ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS184853

ANALISIS PEMANFAATAN PLATFORM DARING PADA UMKM BIDANG KESEHATAN DI SURABAYA

UTILIZATION ANALYSIS OF ONLINE PLATFORM ON SMEs IN HEALTH SECTOR IN SURABAYA

ALDY SYAH DAVIQ RAMADHAN
0521 16 4000 0088

Dosen Pembimbing
Rully Agus Hendrawan, S.Kom, M.Eng

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

TUGAS AKHIR - IS184853

ANALISIS PEMANFAATAN PLATFORM DARING PADA UMKM BIDANG KESEHATAN DI SURABAYA

ALDY SYAH DAVIQ RAMADHAN
0521 16 4000 0088

Dosen Pembimbing
Rully Agus Hendrawan, S.Kom, M.Eng

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

UNDERGRADUATE THESIS - IS184853

UTILIZATION ANALYSIS OF ONLINE PLATFORM ON SMEs IN HEALTH SECTOR IN SURABAYA

ALDY SYAH DAVIQ RAMADHAN
0521 16 4000 0088

Supervisor
Rully Agus Hendrawan, S.Kom, M.Eng

INFORMATION SYSTEM DEPARTMENT
Information Technology and Communication Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

ANALISIS PEMANFAATAN PLATFORM DARING PADA UMKM BIDANG KESEHATAN DI SURABAYA

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer (S.Kom)
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas (ELECTICS)
Institut Teknologi Sepuluh Nopember

Oleh

Aldy Syah Daviq Ramadhan
05211640000088

Surabaya, 18 Agustus 2020

Kepala Departemen Sistem Informasi

Dr. Mudjahidin, ST., MT.
NIP. 197010102003121001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

ANALISIS PEMANFAATAN PLATFORM DARING PADA UMKM BIDANG KESEHATAN DI SURABAYA

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

ALDY SYAH DAVIQ RAMADHAN
NRP. 05211640000088

Disetujui Tim Penguji : Tanggal Ujian:
Periode Wisuda : September 2020

Rully Agus Hendrawan, S.Kom., M.Eng	**(Pembimbing I)**
Mahendrawati ER, ST, M.Sc, Ph.D.	**(Penguji I)**
Erma Suryani, S.T., M.T., Ph.D.	**(Penguji II)**

Halaman ini sengaja dikosongkan

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Aldy Syah Daviq Ramadhan
NRP	: 05211640000088
Tempat / Tanggal lahir	: Surabaya / 04 Januari 1998
Fakultas / Departemen	: FTEIC / Sistem Informasi
Nomor Telp / Hp / Email	: 081234116244 / aldyboting@gmail.com

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul:

ANALISIS PEMANFAATAN PLATFORM DARING PADA UMKM BIDANG KESEHATAN DI SURABAYA

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi p*lag*iarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku. Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, 30 Mei 2020

Aldy Syah Daviq Ramadhan

NRP.05211640000088

Halaman ini sengaja dikosongkan

ANALISIS PEMANFAATAN PLATFROM DARING PADA UMKM BIDANG KESEHATAN DI SURABAYA

Nama Mahasiswa	**: Aldy Syah Daviq Ramadhan**
NRP	**: 0521 16 4000 0088**
Departemen	**: Sistem Informasi FTIK-ITS**
Pembimbing I	**: Rully Agus Hendrawan, S.Kom., M.Eng**

ABSTRAK

Saat ini dunia berada pada masa pandemi Covid-19. Banyak sektor yang terkena dampak dari pandemi ini. Sektor tersebut adalah sektor ekonomi, Kesehatan, dan teknologi. Surabaya adalah salah satu daerah di Indonesia yang terkena dampak dari pandemik. Banyak aktivitas produktif yang sekarang dilakukan dengan pola bekerja dari rumah (work from home). Hal ini mendorong bisnis di Kota Surabaya untuk mendayagunakan platform *daring dengan lebih maksimal. Platform daring dapat didefinisikan sebagai layanan yang tersedia di Internet termasuk toko daring, mesin pencari, media sosial, outlet konten kreatif, toko aplikasi, layanan komunikasi, sistem pembayaran, dan banyak lagi. UMKM (Usaha Mikro Kecil dan Menengah) adalah salah satu contoh pihak yang dapat memanfaatkan perkembangan teknologi serta* platform *daring.*

Tujuan dari penelitian tugas akhir ini adalah untuk mengetahui bagaimana tingkat pendayagunaan UMKM bidang kesehatan di Surabaya dalam mendayagunakan platform *daring. Selain itu, tujuan lain dari penelitian ini untuk memvisualisasikan data UMKM bidang kesehatan di Surabaya dalam menggunakan platform daring. Pada penelitian ini dilakukan proses web scraping untuk mendapatkan data tentang penggunaan* platform *daring oleh UMKM bidang kesehatan, serta dirancang*

visualisasi data untuk dilakukan analisis secara deskriptif terhadap data yang didapatkan.

Hasil penelitian menunjukkan bahwa produk yang paling banyak dijual oleh UMKM daring adalah essential oil dan hand sanitizer. Selain itu, didapatkan informasi bahwa tren kenaikan jumlah UMKM yang menggunakan platform daring adalah mendekati pola tren eksponensial. Jumlah UMKM yang memiliki jalur non daring dalam melakukan proses jual beli adalah sebanyak 5462 UMKM dengan rata-rata review per UMKM sebanyak 10 dan rata-rata skor rating sebesar 4.46 dari 5.dari segi kuantitas, pemanfaatan platform *daring oleh UMKM bidang kesehatan sudah sangat baik. Dibuktikan dengan adanya 19770 toko daring pada e-commerce dan banyaknya rata-rata jumlah katalog produk pada toko, yakni sebanyak lebih dari 200 produk. Namun secara kualitas, pemanfaatan* platform *daring oleh UMKM bidang kesehatan masih kurang. Hal ini dibuktikan minimnya jumlah follower pada toko daring serta kurang optimalnya pemanfaatan fitur yang tersedia pada e-commerce.*

Kata Kunci: UMKM, Kesehatan, Platform Daring, Web Scraping, Visualisasi Data

UTILIZATION ANALYSIS OF ONLINE PLATFORM ON SMEs IN HEALTH SECTOR IN SURABAYA

Name	**: Aldy Syah Daviq Ramadhan**
NRP	**: 0521 16 4000 0088**
Department	**: Information System FTIK-ITS**
Supervisor I	**: Rully Agus Hendrawan, S.Kom., M.Eng**

ABSTRACT

At present the world is in the Covid-19 pandemic. Many sectors were affected by this pandemic. The sectors are the economic, health and technology. Surabaya is one of the regions in Indonesia affected by the pandemic. Many productive activities are now carried out by working from home. This encourages businesses in the city of Surabaya to maximize the use of the online platforms. Online platforms can be defined as services available on the Internet including online stores, search engines, social media, creative content outlets, application stores, communication services, payment systems, and more. SMEs (Micro, Small and Medium Enterprises) is one example of parties who can take advantage of technological developments and online platforms*.*

The purpose of this research is to find out how is the maturity level of the health sector SMEs in Surabaya in utilizing online platforms*. In addition, another objective of this research is to get an idea of how SMEs in the health sector in Surabaya make use of the various online* platforms *available. In this study a web scraping process was carried out to obtain data on the use of online* platforms *by the SMEs health sector, as well as designing a data visualization for descriptive analysis of the data obtained.*

The results showed that the products most widely sold by online SMEs were essential oil and hand sanitizers. In addition, data

showed that the upward trend in the number of SMEs using online platforms is approaching exponential trend patterns. The number of SMEs that have offline channels is 5462 SMEs with an average review per SMEs of 10 and the average rating score of 4.46 out of 5. in terms of quantity, the use of online platforms by the SME sector in health has been very good . Evidenced by the existence of 19770 online stores in e-commerce and the average number of product catalogs in stores, which is more than 200 products. But in quality, the use of online platforms by the SME health sector is still lacking. This is evidenced by the low number of followers in online stores and the suboptimal use of features available in e-commerce.

Keywords: SMEs, Health, Online Platform, Web Scraping, Data Visualisation

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Dengan mengucapkan rasa syukur kepada Allah Yang Maha Pengasih serta Maha Penyayang, maka atas izin-Nya penulis dapat menyelesaikan buku yang sederhana ini dengan judul

"ANALISIS PEMANFAATAN PLATFORM DARING PADA UMKM BIDANG KESEHATAN DI SURABAYA". Tugas Akhir ini dapat diselesaikan tentu dengan dukungan, saran, dan doa dari rekan-rekan yang senantiasa hadir untuk penulis. Secara khusus, penulis mengucapkan terima kasih yang sedalam-dalamnya kepada:

1. Allah SWT, yang selalu membimbing serta memberikan kasihNya kepada penulis.
2. Mudjahidin, ST, MT. selaku Kepala Departemen Sistem Informasi ITS Surabaya.
3. Rully Agus Hendrawan, S.Kom, M.Eng selaku dosen pembimbing yang telah mencurahkan segenap tenaga, waktu dan pikiran dalam penelitian ini, serta memberikan motivasi yang membangun.
4. Erma Suryani, ST, MT, Ph.D dan Mahendrawathi ER. S.T., M.Sc., Ph.D. selaku dosen penguji yang telah memberikan kritik dan saran yang membuat kualitas penelitian ini lebih baik lagi.
5. Segenap dosen dan karyawan Departemen Sistem Informasi.
6. Orang tua penulis, yang tiada hentinya mendoakan dan memberikan dukungan kepada penulis.
7. Pihak lainnya yang berkontribusi dalam tugas akhir yang tidak dapat penulis sebutkan satu per satu.

Penyusunan tugas akhir ini masih jauh dari kata sempurna, untuk itu penulis menerima segala kritik dan saran yang membangun sebagai upaya menjadi lebih baik lagi ke depannya. Semoga buku tugas akhir ini dapat memberikan manfaat untuk pembaca.

Surabaya, 2 Juni 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab ini akan menjelaskan tentang pendahuluan pengerjaan tugas akhir yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, dan manfaat yang akan diperoleh dari penelitian tugas akhir ini.

1.1 Latar Belakang

Saat ini dunia berada pada masa pandemi Covid-19 [1]. Banyak sektor yang terkena dampak dari pandemi ini. Sektor tersebut adalah sektor ekonomi, Kesehatan, dan teknologi. Pada sektor ekonomi sendiri, pandemik ini diprediksi akan memberikan pukulan pada tingkat pertumbuhan ekonomi Indonesia sampai pada level minus 0.4 persen [2]. Pada sektor teknologi, penggunaan platform setidaknya untuk kepentingan pembelajaran daring dan belanja daring telah mengalami peningkatan. Pada Bulan April 2020, Indihome mencatat setidaknya ada peningkatan 40 persen terhadap pengguna internet dibandingkan bulan sebelumnya [3]. Permintaan terhadap kebutuhan pada sektor kesehatan pun jelas meningkat pula [4].

Surabaya adalah salah satu daerah di Indonesia yang terkena dampak dari pandemik. Sekolah dan kampus diliburkan, aktivitas kerja dilakukan dengan pola bekerja dari rumah (*work from home*), dan berbagai macam dampak lainnya [5]. Hal ini mendorong bisnis di Kota Surabaya untuk mendayagunakan platform daring dengan lebih maksimal.

Platform daring dapat didefinisikan sebagai layanan yang tersedia di Internet termasuk toko daring, mesin pencari, media sosial, outlet konten kreatif, toko aplikasi, layanan komunikasi, sistem pembayaran, dan banyak lagi [6]. Dewasa ini, pemanfataan platform daring terhadap kepentingan bisnis sudah marak, baik untuk membantu proses jual beli, pemasaran, maupun hubungan pelanggan. UMKM (Usaha Mikro Kecil dan Menengah) adalah salah satu contoh pihak yang dapat memanfaatkan perkembangan teknologi serta platform daring.

Pada tugas akhir ini, peneliti ingin melakukan analisis pemanfaatan platform daring oleh UMKM pada bidang kesehatan, khususnya adalah UMKM pada Kota Surabaya.

1.2 Rumusan Masalah

Rumusan masalah dari pembuatan tugas akhir ini sebagai berikut:

1. Bagaimana hasil proporsi data UMKM bidang kesehatan di Surabaya dalam pendayagunaan platform daring?
2. Bagaimana persebaran data UMKM bidang Kesehatan di Surabaya dalam pendayagunaan platform daring?

1.3 Batasan Permasalahan

Batasan masalah pada tugas akhir ini sebagai berikut.

1. Data yang dikumpulkan merupakan data hasil *web scraping* toko kategori kesehatan di Surabaya
2. Data didapatkan dari Google My Business, Tokopedia, Shopee, dan Instagram
3. Data yang didapat digunakan untuk keperluan *descriptive analytics*

1.4 Tujuan

Tujuan dari penelitian tugas akhir ini adalah sebagai berikut.

1. Memvisualisasikan data UMKM bidang kesehatan di Surabaya dalam bentuk *dashboard*
2. Mengetahui tingkat pendayagunaan platform daring oleh UMKM bidang kesehatan di Surabaya dalam bentuk analisis secara deskriptif

1.5 Manfaat

Berikut manfaat yang diharapkan dari pengerjaan tugas akhir ini:

1. Secara Teori
   Visualisasi data serta analisis secara deskriptif yang dihasilkan dapat menjadi bahan dan referensi untuk analisis lebih lanjut.

2. Secara Praktis
   Memberikan informasi mengenai kondisi UMKM bidang kesehatan di Surabaya dalam mendayagunakan platform daring dalam bentuk *dashboard*.

1.6 Target Luaran

Target luaran dalam penelitian ini adalah sebagai berikut.

1. Laporan tugas akhir
2. Jurnal mahasiswa ITS
3. *Dashboard* mengenai kondisi UMKM bidang kesehatan di Surabaya dalam mendayagunakan platform daring

1.7 Relevansi

Tugas akhir ini termasuk dalam topik pengetahuan 'bisnis digital' yang berkontribusi pada tujuan penelitian Laboratorium Sistem Enterprise yaitu untuk meningkatkan ekselensi operasional, memandu pertumbuhan organisasi dan produktivitas individu; Untuk mengekspos, mempromosikan, dan membangun rantai bisnis yang kuat. Pada **Gambar 1.1** Kerangka Kerja Riset Laboratorium Sistem Enterprise digambarkan tujuan penelitian dan topik pengetahuan di Laboratorium Sistem Enterprise.

Gambar 1.1 Kerangka Kerja Riset Laboratorium Sistem *Enterprise*

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka terdiri dari landasan-landasan yang akan digunakan dalam penelitian tugas akhir ini. Adapun cakupan dari tinjauan pustaka meliput penelitian-penelitian sebelumnya, kajian pustaka, dan metode yang digunakan selama pengerjaan.

2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian yang memiliki topik yang hampir serupa dengan penelitian ini. Adapun penelitian-penelitian tersebut akan dijelaskan pada **Tabel 2.1.**

Tabel 2.1 Penelitian Terkait

Penelitian 1 [7]	
Judul Penelitian	Exploiting web scraping in a collaborative filtering- based approach to web advertising
Penulis	E. Vargiu, M. Urru
Tahun	2012
Deskripsi Umum Penelitian	Pada penelitian tersebut dilakukan web scraping akan suatu website dapat menampilkan iklan sesuai dengan preferensi pengunjung web Penelitian ini digunakan sebagai bahan referensi dalam mengetahui secara teknis kegiatan *web scrapping*
Penelitian 2 [8]	
Judul Penelitian	Rancang Bangun Analisis Tren Produk Pada Situs Pasar

	Online Berdasarkan Ranking Penjualan
Penulis	Arbintoro Mas
Tahun	2019
Deskripsi Umum Penelitian	Pada penelitian ini dilakukan rancang bangun aplikasi untuk melakukan analisis tren produk berdasarkan ranking penjualan. Data yang digunakan oleh aplikasi didapatkan dari proses web scraping pada sebuah *marketplace*. Penelitian ini digunakan sebagai bahan referensi dalam mengetahui secara teknis kegiatan *web scrapping*, khsusunya pada sebuah *marketplace*
Penelitian 3 [9]	
Judul Penelitian	E-commerce product review sentiment classification based on a naïve Bayes continuous learning framework
Penulis	Feng Xua, Zhenchun Panb, Rui Xia
Tahun	2020
Deskripsi Umum Penelitian	Pada penelitian ini dilakukan klasifikasi menggunakan naïve Bayes continuous

	learning framework pada data review dengan teknik *sentiment analysis*. Pada penelitian ini didapatkan pandangan mengenai salah satu contoh bentuk analisis terhadap data toko, dalam hal ini adalah data produk.
Penelitian 4 [10]	
Judul Penelitian	Pengaruh Strategi Pemasaran Online terhadap Peningkatan Laba UMKM
Penulis	Ira Setiawati, Penta Widyartati
Tahun	2017
Deskripsi Umum Penelitian	Pada penelitian ini dilakukan survei terhadap pelaku UMKM terkait dengan dampak pemasaran melalui platform daring. Dan didapatkan kesimpulan bahwa pemasaran melalui platform daring membawa pengaruh positif terhadap peningkatan laba. Pada penelitian ini didapatkan pandangan mengenai perspektif lain dalam meneliti kaitan platform daring dengan UMKM, sehingga bisa menjadi referensi.

2.2 Studi Literatur

Pada bagian ini akan dibahas mengenai studi literatur yang digunakan oleh peneliti.

2.2.1 Usaha Mikro Kecil Menengah (UMKM)

UMKM merupakan singkatan dari Usaha Mikro, Kecil, dan Menengah yang dapat didefinisikan sebagai entitas perusahaan yang dimiliki dan dikelola oleh seseorang atau sekelompok kecil orang dengan jumlah kekayaan dan pendapatan tertentu. UMKM memiliki beberapa karakteristik dari aspek finansial, antara lain memiliki modal kurang dari 20 juta, membutuhkan dana Rp 5 juta untuk melakukan satu putaran usaha, memiliki aset maksimum Rp 600 juta, dan omzet tahunan sebesar kurang dari Rp 1 miliar. Selain itu, UMKM juga memiliki karakteristik dari aspek komoditas yang dihasilkan, yaitu kualitas produk yang belum terstandar, desain produk terbatas, jenis produk terbatas, kapasitas produk terbatas, dan bahan baku yang kurang terstandar [11].

Terdapat tiga jenis usaha pada UMKM, yaitu usaha mikro, usaha kecil, dan usaha menengah. Penggolongan perusahaan tersebut didasarkan pada ukuran usaha yang sedang dijalankan. Dalam segi jumlah karyawan, perusahaan dapat dikatakan usaha mikro jika memiliki karyawan 10 orang, usaha kecil dengan 30 orang, dan usaha menengah dengan 300 orang [12].

UMKM sangat berperan penting dalam kegiatan pembangunan ekonomi nasional karena dapat menyerap tenaga kerja, mendistribusikan hasil-hasil pembangunan. UMKM dapat bergerak pada berbagai sektor bisnis seperti perdagangan, industri rumah tangga, pertanian, perkebunan, peternakan, perikanan dan jasa. Namun secara spesifik UMKM yang akan dibahas pada tugas akhir ini adalah sektor kesehatan.

2.2.1.1 UMKM Bidang Kesehatan

UMKM bidang kesehatan adalah UMKM yang melakukan proses pemberian layanan dalam hal kesehatan. UMKM bidang

kesehatan yang dimaksud pada penelitian ini adalah UMKM yang memberikan layanan dan atau menjual produk dengan kategori seperti kesehatan seksual, suplemen makanan, alat medis, dan perawatan diri.

2.2.2 Analitik Bisnis

Analitik Bisnis adalah proses mengubah sekumpulan data menjadi pengetahuan baru melalui analisis untuk digunakan dalam pengambilan keputusan dan pemecahan masalah yang efektif sehingga dapat menghasilkan tindakan kompetitif untuk menciptakan nilai [13]. Proses ini melibatkan keterampilan, teknologi, aplikasi dan praktik dalam penyelidikan berulang terus menerus dari kinerja bisnis masa lalu untuk mendapatkan pemahaman yang lebih baik dan mendorong perencanaan bisnis [14].

Jika dilihat dari segi kegunaan, analitik bisnis dapat diklasifikasikan menjadi tiga pola pikir, yaitu analitik deskriptif, analitik prediktif, dan analisis preskriptif. Ketiga pola pikir tersebut memiliki tujuan spesifik yang berbeda dalam melakukan analitik bisnis [15]. Pada tugas akhir ini, pola pikir yang akan digunakan yaitu analitik deskriptif, dimana analisis dilakukan bertujuan memberikan deskripsi atau informasi mengenai kinerja masa lalu atau keadaan bisnis dan lingkungannya dengan menggunakan data yang disimpan dalam *database* atau gudang data. Analisis ini dapat membantu perusahaan untuk mendapatkan wawasan dari data historis dengan pelaporan, kartu skor *(scorecard)*, dan pengelompokan [16]. Dalam analitik deskriptif, visualisasi merupakan salah satu komponen yang penting karena dapat mengembangkan wawasan yang kuat dalam pelaporan sehingga membantu perusahaan untuk melihat fakta-fakta seperti, apa yang telah terjadi, di mana, dan seberapa sering terjadi.

2.2.3 Web Scrapping

Web scrapping merupakan salah satu metode yang digunakan dalam pengumpulan data. *Web scrapping* atau yang juga dikenal sebagai ekstraksi web, adalah teknik untuk mengekstraksi data

dari World Wide Web (WWW) dan menyimpannya ke sistem *file* atau basis data untuk pengambilan atau analisis di kemudian hari. Secara umum, data *web* diambil dengan menggunakan protokol bernama Hyper-text Transfer Protocol (HTTP) atau dengan menggunakan *browser* yang dapat dilakukan secara manual oleh pengguna atau otomatis menggunakan bot/*web scraper*. Secara teknis, terdapat dua modul penting dari program *web scraping* yakni modul untuk menyusun permintaan HTTP, seperti Urllib2 atau selenium dan satu lagi untuk mengurai dan mengekstraksi informasi dari kode HTML mentah, seperti BeautifulSoup atau Pyquery.

Web Scraping dapat digunakan untuk berbagai skenario, seperti *contact scraping*, pemantauan / perbandingan perubahan harga, pengumpulan ulasan produk, pengumpulan daftar real estat, pemantauan data cuaca, deteksi perubahan situs web, dan integrasi data web [17]. Pada penelitian ini, web scraping akan digunakan untuk melakukan ekstraksi data pada platform daring yang digunakan oleh UMKM bidang kesehatan di Surabaya untuk disimpan dalam sebuah basis data.

2.2.4 Visualisasi Data

Visualisasi data merupakan proses merepresentasikan data secara grafis atau gambar dengan jelas, akurat dan efektif. Beberapa teknik umum yang digunakan untuk memvisualisasikan suatu data yaitu dengan menggunakan grafik garis, grafik batang, *pie chart*, dan *scatter plot*. Dengan adanya visualisasi data ini, komunikasi ide-ide kompleks dapat dilakukan dengan lebih efisien dan jelas serta konsep yang disampaikan dapat bersifat *universal*. Hal ini yang menyebabkan visualisasi data sangat berperan dalam berbagai sektor seperti kesehatan, ilmu alam, *fraud detection*, dan berbagai sektor lainnya [18].

2.2.5 Entity Resolution

Entity resolution adalah proses yang secara probabilistik mengidentifikasi beberapa hal berdasarkan pada serangkaian

petunjuk ambigu [19]. Dalam penelitian ini, akan digunakan entity resolution berdasarkan uji similaritas.

2.2.6 Metric

Metric adalah ukuran penilaian kuantitatif yang biasa digunakan untuk menilai, membandingkan, dan melacak kinerja atau tingkat keberhasilan [20]. *Metric* dirancang dengan mengacu pada tujuan bisnis. Contoh *metric* yang biasa digunakan adalah seperti jumlah penjualan, ROI, *customer satisfaction* rate, dan lain-lain [21]. Pada penelitian ini, *metric* dirancang dengan mengacu pada tujuan analisis. Kumpulan *Metric* yang telah dirancang kemudian digunakan untuk menyusun sebuah *dashboard*.

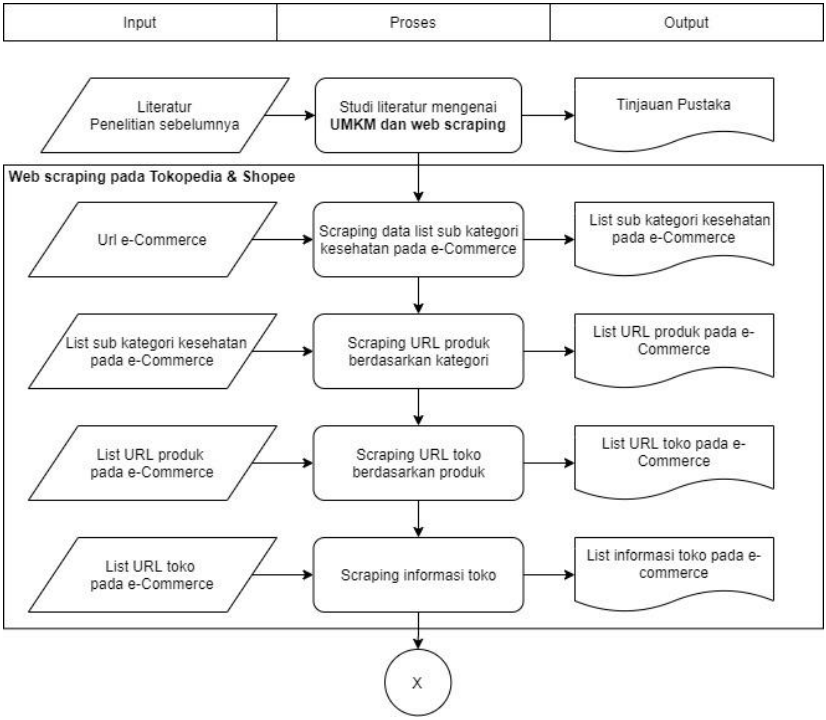
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

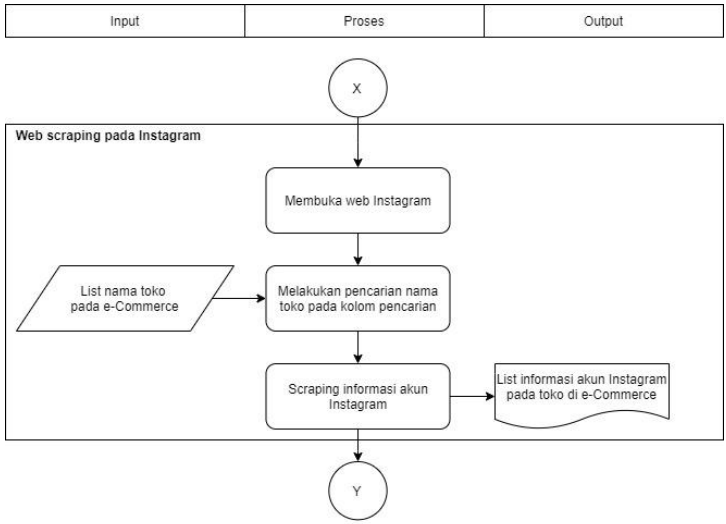
Pada bab ini dijelaskan metodologi yang digunakan sebagai panduan untuk menyelesaikan penelitian tugas akhir.

3.1 Diagram Metodologi

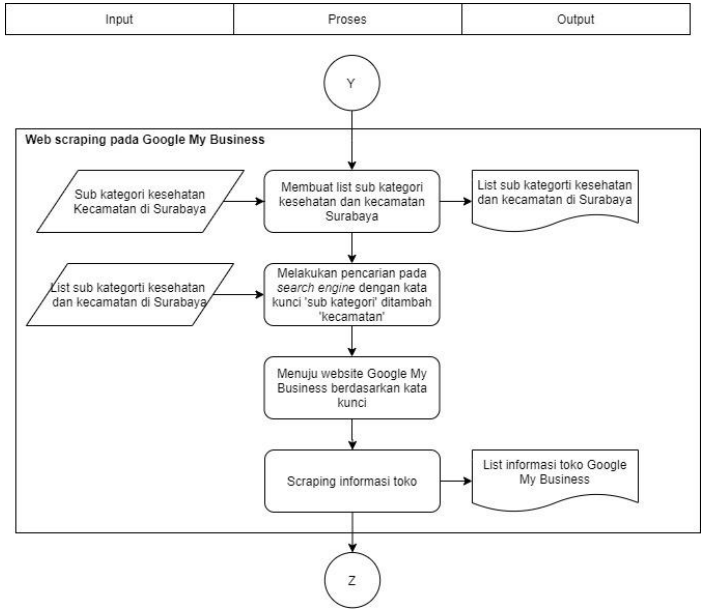
Sub-bab ini menjelaskan tahapan dan proses yang dilakukan dalam penelitian. **Gambar 3.1** sampai dengan **Gambar 3.5** merupakan diagram metodologi penelitian ini.



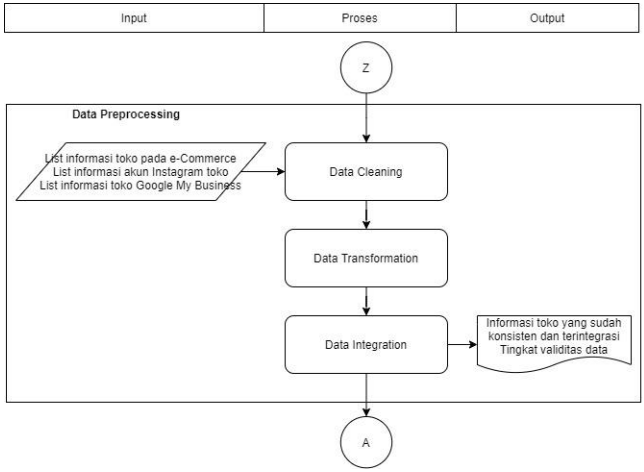
Gambar 3.1 Metodologi Penelitian Bagian 1



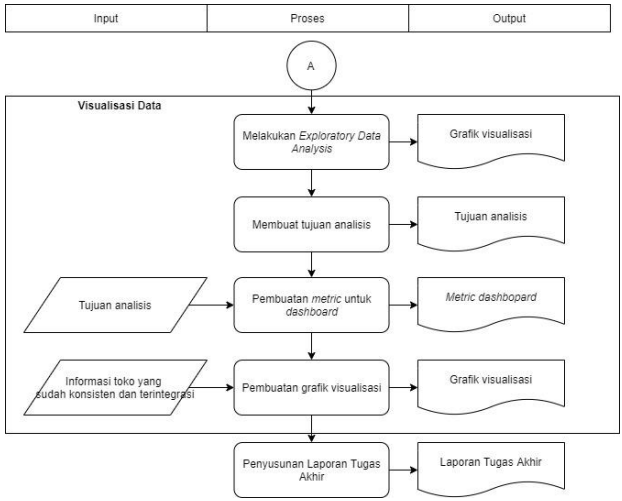
Gambar 3.2 Metodologi Penelitian Bagian 2



Gambar 3.3 Metodologi Penelitian Bagian 3



Gambar 3.4 Metodologi Penelitian Bagian 4



Gambar 3.5 Metodologi Penelitian Bagian 5

3.2 Uraian Metodologi

Bagian ini merupakan penjelasan-penjelasan uraian dari metodologi pengerjaan tugas akhir. Adapun uraian tersebut sebagai berikut.

3.2.1 Studi Literatur

Studi literatur merupakan tahap awal yang perlu dilakukan agar proses penelitian dapat berjalan sesuai dengan ilmu serta teori yang semestinya. Pada penelitian ini, studi literatur dilakukan dengan membahas topik lebih mendalam tentang UMKM dan *web-scrapping*. Seperti yang tertera pada Gambar 3.1, *output* dari tahap ini adalah tinjauan pustaka yang dapat digunakan sebagai landasan untuk proses-proses berikutnya.

3.2.2 Web Scraping pada Tokopedia dan Shopee

Tahap ini berfokus pada pengumpulan data dengan menggunakan teknik *web-scraping* pada platform Tokopedia dengan domain *tokopedia.com* dan Shopee dengan domain *shopee.co.id*. Tahap ini terbagi menjadi empat sub-tahap yaitu *scraping* data *list* sub-kategori kesehatan pada e-commerce, *scraping* URL produk berdasarkan kategori, *scraping* URL toko berdasarkan produk dan *scraping* informasi toko. Hasil akhir yang didapatkan pada tahap ini adalah daftar informasi toko pada e-commerce Tokopedia dan Shopee.

3.2.3 Web Scraping pada Instagram

Tahap ini berfokus pada pengumpulan data dengan menggunakan teknik *web-scraping* pada platform Instagram. Tahap ini terbagi menjadi tiga sub-tahap, yakni membuka web Instagram yang dapat diakses pada domain *instagram.com*, melakukan pencarian nama toko pada kolom pencarian yang disesuaikan pada list nama toko di e-commerce, dan scraping informasi akun Instagram tersebut. Hasil akhir yang didapatkan pada tahap ini yakni daftar informasi akun Instagram pada toko di e-commerce.

3.2.4 Web Scraping pada Google My Business

Tahap ini berfokus pada pengumpulan data dengan menggunakan teknik *web-scraping* pada platform Google My Business. Dalam melakukan web-scraping, tahap ini terbagi menjadi empat sub-tahap yaitu membuat daftar sub-kategori Kesehatan dan kecamatan di Surabaya, melakukan pencarian data pada mesin pencari dengan menggunakan kata kunci ‘sub kategori’ ditambah ‘kecamatan’, membuka website Google My Business berdasarkan kata kunci

tersebut, dan *scraping* informasi toko yang diperlukan. Hasil akhir tahap ini adalah daftar informasi toko dari Google My Business.

3.2.5 Data Pre-Processing

Tahap *pre-processing* data dilakukan dengan tujuan untuk mempersiapkan data agar siap divisualisasi pada tahap selanjutnya. Tahap ini terbagi menjadi tiga langkah, *yaitu data cleaning, data transformation,* dan *data integration*. Pembersihan data atau *data cleaning* merupakan proses penghapusan data-data yang tidak digunakan. Sedangkan transformasi data atau *data transformation* merupakan proses transformasi data menjadi bentuk data yang diinginkan. Setelah data sudah tersedia dalam bentuk yang sesuai, data dihimpun dalam satu kesatuan dan dilakukan uji validitas pada proses *data integration*. Hasil akhir tahap ini adalah informasi toko yang sudah konsisten dan terintegrasi dengan baik.

3.2.6 Visualisasi Data

Dengan mengacu pada Gambar 3.5, tahap ini dimulai dengan melakukan EDA (*exploratory data analysis*). Proses EDA dilakukan dengan tujuan untuk mendapatkan gambaran mengenai kondisi dari UMKM bidang kesehatan secara deskriptif. Setelah melakukan EDA dan mendapatkan gambaran dalam bentuk grafik, selanjutnya dilakukan pembuatan tujuan analisis.

Tujuan analisis digunakan untuk menjadi dasar dalam pembuatan *metric*. Metric digunakan untuk menentukan informasi apa saja yang akan ditampilkan sebagai bagian dari *dashboard* yang hendak dibangun. Jenis grafik yang digunakan beragam karena disesuaikan dengan jenis data yang hendak divisualisasikan. Sebagai contoh, untuk data yang berhubungan dengan letak geografis, visualisasi dalam bentuk map akan digunakan untuk merepresentasikan data tersebut.

3.2.7 Penyusunan Tugas Akhir

Penyusunan buku tugas akhir dilakukan setelah mendapatkan hasil akhir penelitian. Tujuan penyusunan buku tugas akhir ini agar hasil penelitian dapat didokumentasikan dengan baik sehingga dapat menjadi rujukan atau referensi penelitian selanjutnya.

Halaman ini sengaja dikosongkan.

BAB IV
PERANCANGAN

Pada bab ini akan dibahas mengenai seluruh proses perancangan *scraping* data, *pre-process* data, serta visualisasi dan analisis data.

4.1 Rancangan Web Scraping

Web scraping yang hendak dilakukan pada penelitian ini dilakukan pada beberapa platform yakni Tokopedia dan Shopee, Instagram, dan Google My Business. Data yang dihimpun setidaknya terdapat 2 jenis berdasarkan fungsinya.

Jenis yang pertama adalah data yang bertujuan untuk menjadi alat bantu dalam mencari data utama. Yang kedua adalah data utama dimana merupakan data yang menyimpan informasi yang akan digunakan dalam *descriptive analytics*. Program *scraper* yang digunakan pada penelitian ini adalah program yang dirancang oleh Ubai Yusuf Siraj Samudera [22] dan disesuaikan berdasarkan kebutuhan peneliti.

4.2 Data Pre-Processing

Pada tahap *data pre-processing*, data dan informasi yang didapatkan dari proses *web scraping* diproses agar konsistensi data baik dan bisa digunakan untuk tujuan analisis dan visualisasi. Adapun beberapa langkah pre-processing yang hendak dilakukan antara lain adalah *data cleaning*, *data transformation*, dan *data integration*. Kode program yang dijalankan pada proses *data pre-processing* adalah menggunakan kode program yang dirancang oleh Ubai Yusuf Siraj Samudera [22]. Berikut adalah detail dari proses *data pre-processing*.

4.2.1 Data Cleaning

Proses ini menghasilkan *output* berupa keadaan data dimana tidak ditemukan lagi redundansi data. Mengisi nilai-nilai *null* dengan nilai tertentu juga akan dilakukan pada tahap ini.

4.2.2 Data Transformation

Proses ini menghasilkan *output* berupa keadaan data dimana sudah terjadi perubahan tipe data pada beberapa kolom. Perubahan tersebut antara lain seperti perubahan nilai 'K' pada *follower* menjadi 1000 dan perubahan-perubahan lainnya sesuai dengan kebutuhan. Ekspektasi yang diharapkan dengan diadakannya proses ini adalah agar informasi pada setiap kolomnya dapat dimaksimalkan untuk proses analisis lebih lanjut.

4.2.3 Data Integration

4.2.3.1 Integration

Proses terakhir dari *pre-processing* adalah melakukan integrasi pada beberapa tabel data yang dimiliki. Proses integrasi dilakukan secara berurutan. Rancangan integrasi yang dilakukan adalah seperti pada **Tabel 4.1.**

Tabel 4.1 Rancangan proses data integration

Urutan	Sumber Data	Data yang akan Diintegrasi	Keterangan
1	Tokopedia	Instagram_Tokopedia	Toko daring pada Tokopedia dengan tambahan informasi mengenai platform Instagram
2	Shopee	Instagram_Shopee	Toko daring pada Shopee dengan tambahan informasi mengenai platform Instagram
3	Tokopedia	Shopee	Toko daring secara umum

Urutan	Sumber Data	Data yang akan Diintegrasi	Keterangan
4	Google Business	Tokopedia dan Shopee	Toko secara keseluruhan, yakni toko yang dapat dijangkau melali jalur daring dan tidak daring

4.2.3.2 Data Validation Testing

Setelah data diintegrasikan, selanjutnya akan dilakukan pengujian terhadap validitas data yang telah dimiliki. Dalam melakukan pengujian, penelitian ini menggunakan rancangan langkah-langkah pengujian yang disusun oleh penelitian milik Syafrie Dwi Faisal [23].

4.3 Rancangan Visualisasi Data

Terdapat beberapa hal yang dilakukan pada proses ini, antara lain adalah melakukan EDA (*Exploratory Data Analysis*), lalu menentukan tujuan analisis, penyusunan *metric*, dan terakhir adalah melakukan visualisasi data.

4.3.1 Exploratory Data Analysis

Proses EDA dilakukan dengan tujuan untuk mendapatkan gambaran mengenai kondisi dari data yang telah didapatkan melalui proses *web scraping*. Proses EDA dilakukan dengan cara membuat sebuah visualisasi data dalam bentuk grafik menggunakan Microsoft Power BI. Hasil dari EDA kemudian digunakan untuk menentukan tujuan analisis yang selanjutnya akan diturunkan menjadi *metric* untuk merancang sebuah *dashboard*.

4.3.2 Penyusunan *Metric*

Setelah dilakukan *pre-processing* dan EDA, kemudian diperlukan penyusunan *metric* untuk membantu dalam melakukan *descriptive analytics*. Dalam merancang *metrics*,

peneliti terlebih dulu perlu untuk merancang tujuan analisis yang ingin dilakukan. Tuuan analisis dirancang dengan mengacu pada tujuan penelitian dan dengan melihat pertimbangan kondisi data yang mengacu pada hasil dari proes EDA. Berikut ini merupakan tujuan analisis yang ingin dilakukan berdasarkan platform yang diamati seperti pada **Tabel 4.2**.

Tabel 4.2 Tujuan Analisis yang Ingin Dilakukan

Platform	**Tujuan**	**Indikator**
Google My Business	Untuk mengetahui intensitas aktivitas jual beli pada toko secara non daring	• Jumlah toko • Rata-rata durasi jam operasional • Rata-rata skor rating • Rata-rata jumlah review
Tokopedia & Shopee	Untuk mengetahui kuantitas toko di bidang kesehatan yang sudah memanfaatkan platform daring	• Jumlah toko • Rata-rata kategori produk • Rata-rata jumlah produk yang dijual
	Untuk mengetahui kualitas toko di biang kesehatan dalam memanfaatkan platform daring	• Rata-rata jumlah follower • Rata-rata skor rating • Presentase toko yang memanfaatkan fitur *star seller*

Dari tujuan serta indikator yang telah dibuat, kemudian diturunkan menjadi rancangan metric. Rancangan *metrics* dapat dilihat pada **Tabel 4.3**.

Tabel 4.3 Rancangan *Metrics* Visualisasi

Platform	***Metrics***	**Atribut**
	Jumlah toko	Indeks

Platform	***Metrics***	**Atribut**
Google My Business	Rata-rata review	Review
	Rata-rata jumlah jam kerja	Operational_hours
	Rata-rata rating	Rating
	Korelasi rating dengan Jam kerja	Rating, Operational_hours
Daring (Tokopedia dan Shopee)	Jumlah toko	Indeks
	Rata-rata jumlah katalog produk	Active_product/numOf_product
	Rata-rata jumlah follower	Follower
	Rata-rata review	Review
	Rata-rata rating	Rating
	Jumlah toko berdasarkan jenis toko	Seller_type
	Korelasi rating dengan jumlah produk yang dijual	Rating, product_sold
	Modus dari nama produk yang dijual	Name pada data ‘URL produk’
	Modus dari kategori produk	Product_category
Umum	Jumlah toko	Indeks
	Rata-rata review	Review

4.3.3 Visualisasi Data

Visualisasi data dilakukan untuk membuat sebuah *dashboard*. Proses ini dapat dilakukan dengan menggunakan berbagai media, salah satunya adalah grafik. Grafik akan dirancang dengan menggunakan *software* Microsoft Power BI. Perancangan visualisasi yang akan digunakan untuk menggambarkan data sesuai dengan rancangan matriks yang dibuat. Rancangan visualisasi dapat dilihat pada **Tabel 4.4**.

Tabel 4.4 Rancangan Visualisasi Data

Platform	***Metrics***	**Jenis Visualisasi**
Non-Daring	Jumlah toko	*Card, bar chart, map*
	Rata-rata review	*Card*
	Rata-rata jumlah jam kerja	*Card*
	Rata-rata rating	*Gauge*
	Korelasi rating dengan Jam kerja	*Scatter chart*
Daring	Jumlah toko	*Card, map*
	Rata-rata jumlah katalog produk	*Card*
	Rata-rata jumlah follower	*Card*
	Rata-rata review	*Card*
	Rata-rata rating	*Gauge*
	Jumlah toko berdasarkan jenis toko	*Donut chart*
	Korelasi rating dengan jumlah produk yang dijual	*Bar Chart*
	Mode dari layanan pengiriman	*Wordcloud*
	Mode dari kategori produk	*Wordcloud*
Umum	Jumlah toko	*Card*
	Rata-rata review	*Pie Chart*

BAB V
IMPLEMENTASI

Pada bab ini dijelaskan hasil dan pembahasan pada penelitian Tugas Akhir. Proses dari pengolahan data dan analisis akan disintesis dan dibahas hasilnya pada bagian ini.

5.1 Web Scraping

Web scraping dilakukan sebagai proses awal untuk mendapatkan data yang selanjutnya akan dianalisis. Proses *web scraping* dilakukan secara otomatis dengan cara melakukan eksekusi terhadap program *scraper*. Berikut ini merupakan spesifikasi dari program *scraper* seperti pada **Tabel 5.1**.

Tabel 5.1 Spesifikasi Program *Scraper* [22]

Jenis Spesifikasi	Spesifikasi
Bahasa	Python
Library yang digunakan	• Pandas • Selenium • Pyvirtualdisplay • Time • Chromedriver
Bahasa kueri	Xpath

Program lalu dieksekusi pada 3 perangkat berbeda dengan detail spesifikasi perangkat seperti pada **Tabel 5.2**.

Tabel 5.2 Spesifikasi Perangkat untuk Eksekusi Program *Scraper*

Jenis Perangkat	Sistem Operasi	Spesifikasi
Laptop	Windows 10	Inter Core i7 RAM 8.00 GB
Server	Windows 10	Common KVM Processor RAM 8.00 GB
Server	Ubuntu Server	Intel Xeon E7-8870 v2 RAM 16.00 GB

Program *scraper* akan melakukan pengambilan data terhadap 4 platform yakni Tokopedia, Shopee, Instagram, dan Google My Business. Secara umum, penggunaan *library* adalah sebagai berikut:

- Selenium digunakan untuk menjalankan *web browser* secara otomatis. Browser yang digunakan di sini adalah Google Chrome dengan menggunakan Chromedriver
- Chromedriver adalah *driver* yang dipanggil oleh Selenium untuk menjalankan *web browser* secara otomatis
- Pyvirtualdisplay adalah *library* yang digunakan agar program *scraper* dapat dijalankan pada Ubuntu server. *Library* ini bertugas untuk memberikan tampilan semu untuk Ubuntu server agar dapat membuka *web browser*
- Pandas digunakan untuk mengelola data-data yang hendak dikumpulkan dari proses *scraping*
- Time digunakan untuk menghitung durasi eksekusi program *scraper*

Detail mengenai bagaimana program *scraper* dieksekusi pada masing-masing platform adalah sebagai berikut.

5.1.1 Tokopedia

Terdapat 4 proses utama dalam melalukan *web scraping* pada Tokopedia. Adapun proses yang dilakukan pada *scraping* Tokopedia adalah sebagai berikut.

5.1.1.1 Sub-Kategori Kesehatan Tokopedia

Proses ini bertujuan untuk mendapatkan list kategori dan sub kategori kesehatan pada Tokopedia. Untuk mendapatkan kategori, diperlukan interaksi pada elemen yang ada pada halaman utama Tokopedia. Berikut merupakan kode yang digunakan untuk memunculkan list kategori pada Tokopedia seperti pada **Kode 5.1.** Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A**.

Kode 5.1 Interaksi Elemen untuk Memunculkan Kategori Tokopedia [22]

```
WebDriverWait(driver, 3).until(EC.visibility_of_element_lo-
cated((By.XPATH, '//*[@class="css-vk082c"]')))
ActionChains(driver).move_to_element(wait_kategori_button).perform()

WebDriverWait(driver, 3).until(EC.visibility_of_element_lo-
cated((By.XPATH, '//*[@class="css-me46ht"]')))
```

Setelah list kategori dan sub kategori muncul pada halaman *website*, langkah selanjutnya adalah mengambil informasi dari elemen-elemen yang ada. Pengambilan data menggunakan bahas kueri xpath yang dapat dilihat seperti pada **Tabel 5.3** berikut.

Tabel 5.3 Daftar Kueri Xpath untuk Kategori Tokopedia [22]

Nama Kolom	***Xpath***	**Tipe Elemen**
Kategori	`//*[@class="css-1qaqbbz"]`	*Text*
url_kategori	`//*[@class="css-1nykm5o"]`	*Href*

5.1.1.2 Informasi Produk Kesehatan Tokopedia

Setelah mendapatkan URL kategori, langkah selanjutnya adalah untuk mendapatkan URL produk. Tahap awal adalah membuka URL kategori yang didapatkan dari proses sebelumnya. Pada halaman tersebut akan ditemukan katalog produk dimana terdapat beberapa data yang akan diambil setelah melakukan interaksi dengan elemen. Pengambilan data menggunakan bahas kueri xpath yang dapat dilihat seperti pada **Tabel 5.4** berikut.

Tabel 5.4 Daftar Kueri Xpath untuk Produk Tokopedia [22]

Nama Kolom	***Xpath***	**Tipe Elemen**
Kategori	`//*[@class="css-bk6tzz e1nlzfl3"]`	*Text*
Nama_produk	`//*[@class="css-1bjwylw"]`	*Text*

url_produk	//*[@class="css-bk6tzz e1nlzfl3"]/a	*Href*

Katalog produk disusun sedemikian rupa sehingga hanya menampilkan beberapa produk dalam satu halaman. Sehingga perlu dilakukan interaksi terhadap sebuah elemen agar dapat bergeser ke halaman produk berikutnya. Berikut adalah potongan kode yang bertujuan untuk menggeser ke halaman katalog produk berikutnya seperti pada **Kode 5.2**. Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A**.

Kode 5.2 Interaksi Elemen untuk Menggeser Halaman Katalog Produk Tokopedia [22]

```
next_page = driver.
find_elements_by_xpath('//i[@class="css-98hn3t e19tp72t1"]')
next_cant_click_element = driver.
find_element_by_xpath('//*[@Class="css-13vsjvx e19tp72t1"]')
next_page[0].click()
```

5.1.1.3 URL Toko Kesehatan Tokopedia

Setelah mendapatkan URL produk, langkah selanjutnya adalah melakukan *pre-process* terhadap URL produk untuk mendapatkan URL toko. Pada *pre-processing* ini setidaknya terdapat 2 proses yakni proses *splitting* (pengambilan URL toko dari URL produk) dan proses menghilangkan data yang redundan. Berikut adalah potongan kode untuk melakukan proses *splitting* seperti pada **Kode 5.3**.

Kode 5.3 Proses Splitting untuk Mendapatkan URL Toko Tokopedia [22]

```
df_url_toko_notpromo["link"] = df_url_produk_not-
promo["link"].str.rsplit("/", 1).str[-2]
list_toko_promo.append(url.rsplit("/",1)[-2])
```

Setelah dilakukan proses *splitting*, selanjutnya dilanjutkan pada proses menghilangkan data redundan. Berikut adalah potongan kode untuk menghilangkan data redundan seperti pada **Kode 5.4**.

Kode 5.4 Proses Menghilangkan URL Toko Redundan [22]

```
list_toko_promo_unique = list(set(list_toko_promo))
df_url_toko_all = df_url_toko_promo_unique.append(df_url_toko_notpromo)
df_url_toko_all_unique = df_url_toko_all.drop_duplicates('link')
```

Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A**.

5.1.1.4 Informasi Toko Kesehatan Tokopedia

Setelah mendapatkan URL toko, selanjutnya dilakukan proses untuk mendapatkan informasi mengenai toko yang mana merupakan informasi utama pada proses *web scraping* Tokopedia. Seperti pada rancangan, untuk mendapatkan informasi toko, perlu dilakukan beberapa interaksi terhadap elemen. 3 interaksi utama yang perlu dilakukan adalah melalukan hovering terhadap elemen untuk memunculkan reputation point, memunculkan tampilan info toko, dan memunculkan tampilan statistik toko. Berikut merupakan potongan kode yang dapat dilihat pada **Kode 5.5**. Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A**.

Kode 5.5 Interaksi Elemen untuk Memunculkan Informasi Toko Tokopedia [22]

```
#hovering reputation point
point_hover_element = driver.find_element_by_xpath(
       '//*[@class="css-ypd15i-unf-tooltip"]')
ActionChains(driver).move_to_element(
       point_hover_element).perform()

#memunculkan info toko
info_buttons = driver.find_elements_by_xpath(
       '//*[@class="css-rhf1fq-unf-btn e1ggruw00"]')
driver.execute_script("arguments[0].click();",
       info_buttons[0])

#memunculkan statistik toko
statistic_element = driver.find_elements_by_xpath(
       '//*[@class="css-16z1w0p-unf-heading e1qvo2ff6"]')
driver.execute_script("arguments[0].click();",
       statistic_element[0])
```

Pengambilan data menggunakan bahas kueri xpath yang dapat dilihat seperti pada **Tabel 5.5.**

Tabel 5.5 Daftar Kueri Xpath untuk Informasi Toko Tokopedia [22]

Nama Kolom	***Xpath***	**Tipe Elemen**
Link	-	-
Name	`//*[@class="css-1yw0bav"]/span[1]`	*Text*
seller_type	`//*[@class="css-1paptb0-unf-heading e1qvo2ff6"]`	*Text*
City	`//*[@class="css-1k56vr7"]/li[2]/p`	*Text*
Followers	`//*[@class="css-jsut4p-unf-heading e1qvo2ff6"]`	*Text*
reputation_point	`//*[@class="css-1ac807k e1hrfe840"]`	*Text*
product_sold	`//*[@class="css-lzwncz-unf-heading e1qvo2ff2"]`	*Text*
Rating	`//*[@class="css-rfs3ih-unf-heading e1qvo2ff2"]`	*Text*
Review	`//*[@class="css-1s96mum-unf-heading e1qvo2ff6"]`	*Text*
numOf_storefront	`//*[@class="css-z2vy7e"]`	*Count*
Storefront	`//*[@class="css-17mrx6g"]`	*Text*

Owner	`//*[@class="css-1ag3tdd-unf-heading e1qvo2ff4"]`	*Text*
Description	`//*[@class="css-f4j8r2-unf-heading e1qvo2ff8"]`	*Text*
Address	`//*[@class="css-f4j8r2-unf-heading e1qvo2ff8"][1]`	*Text*
Since	`//*[@class="css-1c0g9ad e1ufc1ph0"]/p[3]`	*Text*
numOf_delivery_services	`//*[@class="css-2ibdj3"]`	*count*
delivery_services	`//*[@class="css-15n97o5 e1ufc1ph0"]`	*Text*

5.1.2 Shopee

Terdapat 4 proses besar yang harus dilakukan. Program *scraper* harus dapat melakukan pengambilan data terkait dengan kategori dan sub kategori kesehatan pada Shopee, URL produk, URL toko, dan informasi toko yang menjual produk kesehatan. Adapun proses yang dilakukan pada *scraping* Shopee adalah sebagai berikut.

5.1.2.1 Sub-Kategori Kesehatan Shopee

Dalam rangka mendapatkan sub kategori kesehatan pada Shopee, program *scraper* perlu masuk kedalam tampilan utama pada link *https://shopee.co.id/Kesehatan-cat.14780*. Setelah itu, dilakukan interaksi terhadap elemen tertentu untuk menampilkan keseluruhan list sub kategori kesehatan pada Shopee. Berikut merupakan potongan kode konversi dari rancangan interaksi elemen seperti pada **Kode 5.6**. Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A.**

Kode 5.6 Interaksi Elemen untuk Memunculkan Sub Kategori Kesehatan [22]

```
arrow_down_button = driver.find_elements_by_xpath(
       '//*[@class="stardust-icon stardust-icon-arrow-down"]')
arrow_down_button[0].click()
```

Pengambilan data menggunakan bahas kueri xpath yang dapat dilihat seperti pada **Tabel 5.6**.

Tabel 5.6 Daftar Kueri Xpath Kategori Shopee [22]

Nama Kolom	*Xpath*	Tipe Elemen
kategori	1. //*[@class="shopee-category-list__main-category__link"] 2. //*[@class="shopee-category-list__sub-category"]	*Text*
url_kategori		*Href*

5.1.2.2 Informasi Produk Kesehatan Shopee

Setelah mendapatkan URL kategori, langkah selanjutnya adalah mendapatkan URL produk. Tahap awal yang dilakukan adalah membuka URL kategori yang didapatkan dari proses sebelumnya. Pada halaman tersebut akan ditemukan katalog produk dimana terdapat beberapa data yang akan diambil setelah melakukan interaksi dengan elemen. Pengambilan data menggunakan bahas kueri xpath yang dapat dilihat seperti pada **Tabel 5.7**.

Tabel 5.7 Daftar Kueri Xpath Informasi Produk Shopee [22]

Nama Kolom	*XPath*	Tipe Elemen
Kategori	-	-
Title	//*[@class="O6wiAW"]	*Text*

Nama Kolom	*XPath*	Tipe Elemen
url_produk	`//*[@class="col-xs-2-4 shopee-search-item-result__item"]/div/a`	*Href*

Katalog produk disusun sedemikian rupa sehingga hanya menampilkan beberapa produk dalam satu halaman. Sehingga perlu dilakukan interaksi terhadap sebuah elemen agar dapat bergeser ke halaman katalog produk berikutnya. Berikut adalah potongan kode untuk menggeser halaman katalog seperti pada **Kode 5.7**. Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A.**

Kode 5.7 Interaksi Elemen untuk Menggeser Halaman Katalog Produk Shopee [22]

```
while True:
next_page = driver.find_elements_by_xpath(
'//button[@class="shopee-button-outline shopee-mini-page-controller__next-btn"]')

if (next_page):
        (get data)

        next_page[0].click()
else:
        break
```

5.1.2.3 URL Toko Kesehatan Shopee

Setelah mendapatkan URL produk, selanjutnya adalah dilakukan pengambilan data terkait URL toko yang ada pada halaman produk. Untuk mendapatkan URL toko, tentunya perlu dilakukan interaksi terhadap elemen tertentu. Berikut merupakan potongan kode untuk mendapatkan URL toko seperti pada **Kode 5.8.** Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A**.

Kode 5.8 Interaksi Elemen untuk Mendapatkan URL Toko Shopee [22]

```
url_element = driver.find_elements_by_xpath('//*[@class="_1jOO4S"]/a')
```

Kumpulan URL toko kemudian disimpan dan dihilangkan URL yang redundan.

5.1.2.4 Informasi Toko Kesehatan Shopee

Informasi toko kesehatan Shopee didapatkan dengan cara melakukan interaksi dengan beberapa elemen. Pengambilan data menggunakan bahasa kueri xpath yang dapat dilihat seperti pada **Tabel 5.8.** Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A.**

Tabel 5.8 Daftar Kueri Xpath Informasi Toko Shopee [22]

Nama Kolom	***XPath***	**Tipe Elemen**
Link	-	-
Name	`//*[@class="section-seller-overview-horizontal__portrait-name"]`	*Text*
City	`//*[@class="_3amru2"]`	*text*
seller_type	`//*[@class="ofs-header__page-name"]`	*text*
product_category	`//*[@class="shopee-category-list__body"]`	*text*
numOf_product	`//div[@class="section-seller-overview__item-text" and contains(.,"produk")]/div[2]`	*text*
chat_performance	`//div[@class="section-seller-overview__item-text" and contains(.,"Performa chat")]/div[2]`	*text*
Followers	`//div[@class="section-seller-overview__item-`	*text*

Nama Kolom	*XPath*	Tipe Elemen
	`text" and contains(.,"pengikut")]/div[2]`	
rating_review	`//div[@class="section-seller-overview__item-text" and contains(.,"penilaian")]/div[2]`	*text*
Since	`//div[@class="section-seller-overview__item-text" and contains(.,"penilaian")]/div[2]`	*text*
Voucher	`//*[@class="shop-page__section shop-page__vouchers"]`	*boolean*
numOf_media	`//*[@class="image-carousel__dots"]/div`	*count*
Description	`//div[contains(@class,"shop-page-shop-description")]`	*text*

5.1.3 Instagram

Hal yang perlu dilakukan untuk *scraping* pada Instagram adalah dengan membuka link https://www.instagram.com/instagram. Berikut ini merupakan potongan kode untuk memasukkan kata kunci kedalam kolom pencarian pada Instagram (lihat **Kode 5.9**). Untuk potongan kode secara utuh dapat dilihat pada **LAMPIRAN A**.

Kode 5.9 Interaksi Elemen untuk Memasukkan Kata Kunci Instagram [22]

```
search_element = driver.find_element_by_xpath(
'//*[@class="LWmhU _0aCwM"]/input')
def searching():
    search_element.clear()
    search_element.send_keys(str(store))
    print('store ' + str(index) + ': ' + str(store))
```

Selanjutnya data mengenai akun Instagram akan disimpan untuk digunakan pada proses berikutnya.

5.1.4 Google My Business

Google My Business memerlukan data pendukung untuk dapat melakukan pencarian pada mesin pencari. Detail dari prosesnya adalah sebagai berikut.

5.1.4.1 Data Pendukung Scraping Google My Business

Data pendukung disusun dengan melakukan pencarian pada internet terkait dengan sub kategori yang relevan dengan kategori kesehatan. Berikut ini merupakan contoh data pendukung yang didapatkan untuk melakukan *scraping* Google My Business (lihat **Gambar 5.1** dan **Gambar 5.2**).

kategori
adult day care center
adult foster care service
alternative medicine practitioner
career guidance service
child care agency
child health care centre
child health care centre
chinese medicine clinic
chinese medicine store
community health centre
day care center

Gambar 5.1 Contoh Sub Kategori Kesehatan untuk Google My Business

kecamatan
tegalsari
simokerto
genteng
bubutan
sukolilo
gubeng
gunung anyar
tambaksari
mulyorejo

Gambar 5.2 Contoh Kecamatan di Surabaya untuk Google My Business

Selanjutnya, dilakukan penggabungan antar sub kategori dengan kecamatan dengan menggunakan potongan kode sebagai berikut.

Kode 5.10 Kode untuk Menggabungkan Sub Kategori dengan Kecamatan [22]

```
list_keyword = []
for kategori in list_kategori_gb:
    for kecamatan in list_kecamatan:
        list_keyword.append(str(kategori) + ' ' +
        str(kecamatan))
```

5.1.4.2 Informasi Toko Google My Business

Proses *scraping* dimulai dengan membuka *https://www.google.com* dan lalu memasukkan kata kunci yang sudah disusun seperti potongan kode berikut (lihat **Kode 5.11**).

Kode 5.11 Kode untuk Memasukkan Kata Kunci ke Mesin Pencari [22]

```
for keyword in list_keyword:
        search_element = driver.find_element_by_xpath(
        '//*[@class="gLFyf gsfi"]')

        search_element.clear()
        search_element.send_keys(keyword)
        search_element.send_keys(Keys.RETURN)
```

Ketika kata kunci menemukan hasil toko pada Google My Business, program *scraper* akan membawa *web browser* menuju laman Google My Business. Disitulah proses pengambilan informasi toko dilakukan. Pengambilan data menggunakan bahasa kueri xpath yang dapat dilihat seperti pada **Tabel 5.9.**

Tabel 5.9 Daftar Kueri Xpath Informasi Toko Google My Business [22]

Nama Kolom	*XPath*	Tipe Elemen
Name	`//*[@class="SPZz6b"]/div/span`	*text*
Website	`//*[@class="QqG1Sd"]/a`	*href*
Rating	`//*[@class="Ob2kfd"]/div/span[1]`	*text*
numOf_google_review	`//*[@class="hqzQac"]/span/a/span`	*text*
Price	`//*[@class="YhemCb"][0]`	*text*
Category	`//*[@class="YhemCb"][1]`	*text*
Address	`//*[@class="LrzXr"]`	*text*
operational_hours	`//div[@class="zloOqf PZPZlf"]/div/div/div[1]/div[2]/table/tbody`	*text*
Phone	`//*[@class="LrzXr zdqRlf kno-fv"]`	*text*

Nama Kolom	*XPath*	Tipe Elemen
Instagram	`//*[@class="PZPZlf kno-vrt-t"]`	*href*

5.2 Data Pre-Processing

Sub-bab ini menjelaskan tentang langkah-langkah yang dilakukan pada data sebelum data divisualisasi. Penjelasan mengenai pra-proses yang dilakukan dibagi menjadi tiga bagian yaitu pada platform Tokopedia, Shopee, Instagram, Google My Business, dan integrasi data.

5.2.1 Tokopedia

Terdapat dua proses pada *data pre-processing*, yakni *data cleaning* dan *data transformation*.

Proses *data cleaning* dilakukan dengan menghapus duplikasi data pada kolom link serta melakukan normalisasi dan pembersihan data pada kolom ‘reputation_point, review, city, dan address’.

Proses *data transformation* dilakukan dengan melakukan beberapa perubahan seperti mengganti nan dengan nilai tertentu, merubah nilai ‘K’ dan ‘M’ menjadi 1000 dan 1000000, ekstraksi nilai pada suatu kolom menjadi kolom baru, dan proses lainnya. Detail mengenai keseluruhan *pre-process* dalam dilihat pada

Tabel 5.10.

Tabel 5.10 Pre-process Data Tokopedia

Nama Kolom	*Preprocess*
Name	Tidak ada *preprocess*
Link	Menghapus duplikasi data
seller_type	Mengganti *nan* dengan “Regular seller”

Nama Kolom	*Preprocess*
City	*Filter* area Surabaya
Followers	Mengonversikan nilai ‘k’ dan ‘m’ menjadi *int*
reputation_point	Normalisasi dan membersihkan data
product_sold	- Mengkonvensi nilai ‘k’ dan ‘m’ menjadi *int* Mengganti nilai kosong menjadi nol (0)
Rating	Tidak ada *preprocess*
Review	Normalisasi dan membersihkan data
Owner	Mengganti *value nan* dengan “*no owner information*”
Description	Ekstraksi informasi menjadi atribut baru, yaitu COD, phone, website dan Instagram
Address	- Normalisasi dan membersihkan data - Mengonversikan atribut menjadi kecamatan dan kelurahan
Since	Memisahkan atau *split* data menjadi atribut *monthly_joined* & *year_joined*
numOf_delivery_services	Tidak ada *preprocess*
delivery_services	Tidak ada *preprocess*
numOf_satisfied	Mengubah tipe data menjadi *int*
numOf_neutral	Mengubah tipe data menjadi *int*
numOf_unsatisfied	Mengubah tipe data menjadi *int*

Nama Kolom	*Preprocess*
process_speed	Tidak ada *preprocess*
numOf_storefront	Tidak ada *preprocess*
Storefront	Tidak ada *preprocess*
active_product	- Menghapus baris data yang kosong - Mengubah tipe data menjadi *int*
success_transaction	Mengubah tipe data menjadi *int*
COD **(hasil ekstrasi)**	Tidak ada *preprocess*
phone **(hasil ekstrasi)**	Mengonversikan *value* atribut *phone* agar sesuai dengan standard
website **(hasil ekstrasi)**	Tidak ada *preprocess*
instagram **(hasil ekstrasi)**	Tidak ada *preprocess*
kecamatan **(hasil ekstrasi)**	Tidak ada *preprocess*
kelurahan **(hasil ekstrasi)**	Tidak ada *preprocess*

5.2.2 Shopee

Terdapat dua proses pada *data pre-processing*, yakni *data cleaning* dan *data transformation*.

Proses *data cleaning* dilakukan dengan menghapus duplikasi data pada kolom link serta melakukan normalisasi dan pembersihan data pada kolom ‘name dan product_category’.

Proses *data transformation* dilakukan dengan melakukan beberapa perubahan seperti mengganti nan dengan nilai tertentu, merubah nilai ‘RB’ menjadi 1000, ekstraksi nilai pada suatu kolom menjadi kolom baru, dan proses lainnya. Detail mengenai keseluruhan *pre-process* dalam dilihat pada **Tabel 5.11.**

Tabel 5.11 Pre-process Data Shopee

Nama Kolom	***Preprocess***
Link	- Menghilangkan redundansi data - Menjadikan sebagai index
Name	Menghapus *string* yang tidak memiliki standard *ascii*
City	Filter area Surabaya dengan menggunakan data kecamatan dan kelurahan, serta *string* “sby” dan “”
seller_type	Mengganti *nan* dengan “Regular seller”
product_category	Menghilangkan string ‘Semua Produk\n’
numOf_product	- Mengonversi nilai ‘RB’ menjadi *int* - Merubah tipe data menjadi *int*
chat_performance	- Memisahkan atau *split* data menjadi atribut *chat_performance_unit* - Konversi data persentase menjadi *int*
Followers	- Mengkonvensi nilai ‘RB’ menjadi *int*
rating_review	- Memisah/*split* data menjadi atribut *rating* dan *review* - Merubah tipe data menjadi *int* dan *float*
Since	Tidak ada *preprocess*
Voucher	Tidak ada *preprocess*
numOf_media	Tidak ada *preprocess*

Nama Kolom	***Preprocess***
Description	Ekstraski informasi menjadi atribut baru, yaitu *COD*, *phone*, *website* dan *Instagram*
COD **(hasil ekstraksi)**	Tidak ada *preprocess*
Phone **(hasil ekstraksi)**	Mengonversi isi atribut *phone* agar sesuai dengan standard nomor
Website **(hasil ekstraksi)**	Tidak ada *preprocess*
Instagram **(hasil ekstraksi)**	Tidak ada *preprocess*
Chat_performance_unit **(hasil ekstraksi)**	Tidak ada *preprocess*

5.2.3 Instagram

Terdapat dua proses pada *data pre-processing*, yakni *data cleaning* dan *data transformation*.

Proses *data cleaning* dilakukan dengan melakukan normalisasi dan pembersihan data pada kolom store, account, dan description

Pada data Instagram, transformasi data dilakukan dengan mengubah nilai menjadi *lowercase* pada kolom 'City'. Detail mengenai keseluruhan atribut dengan mekanisme *pre-process* yang dijalankan pada data Instagram dapat dilihat pada **Tabel 5.12.**

Tabel 5.12 Pre-process Data Instagram

Nama Kolom	***Preprocess***
Store	Menghapus *string* yang tidak memiliki standard *ascii*
City	Merubah nilai menjadi *lowercase*
Account	Menghapus *string* yang tidak memiliki standard *ascii*

Nama Kolom	*Preprocess*
Description	Menghapus *string* yang tidak memiliki standard *ascii*
Link	Tidak ada *preprocess*

5.2.4 Google My Business

Terdapat dua proses pada *data pre-processing*, yakni *data cleaning* dan *data transformation*.

Proses ini dilakukan dengan menghapus duplikasi data pada kolom 'Name' serta melakukan normalisasi dan pembersihan data pada beberapa kolom lainnya seperti 'numOf_google_review, Phone, Tokopedia, dan Shopee'.

Proses transformasi pada data dari Google My Business dilakukan dengan beberapa perubahan seperti mengganti normalisasi data, ekstraksi informasi pada suatu kolom menjadi kolom baru, dan proses lainnya. Detail mengenai keseluruhan atribut dengan mekanisme *pre-process* yang dijalankan dapat dilihat pada

Tabel 5.13**.**

Tabel 5.13 Pre-process Data Google My Business

Nama Kolom	*Preprocess*
Name	Menghapus data yang duplikat
Website	Ekstrasi informasi yang menjadi atribut baru : Tokopedia, Shopee dan instagram
Rating	Tidak ada *preprocess*
numOf_google_review	Normalisasi dan membersihkan data
Price	Mengganti data yang kosong menjadi 'no price'

Nama Kolom	***Preprocess***
Category	Menghapus kategori yang tidak sesuai
Address	- Normalisasi dan membersihkan data - Konversi atribut menjadi kecamatan dan kelurahan
Operational_hours	Konversi atribut menjadi *operational_hours_duration*
Phone	Normalisasi dan membersihkan data
Instagram	Menambahkan data hasil ekstraksi informasi
Tokopedia	Normalisasi data
Shopee	Normalisasi data
Operational_hours_duration	Tidak ada *preprocess*
Kecamatan	Tidak ada *preprocess*
Kelurahan	Tidak ada *preprocess*

5.2.5 Data Integration

Pada proses ini, dilakukan intregasi terhadap beberapa data. *Data integration* dilakukan dengan menggunakan metode *entity resolution* dengan dasar perhitungan menggunakan tingkat kemiripan. Detail mengenai integrasi adalah sebagai berikut. Kode program yang dibangun untuk menunjang proses integrasi data terlampir pada **LAMPIRAN A.**

5.2.5.1 Integrasi e-Commerce dengan Instagram

Integrasi data pada proses ini adalah mengintegrasikan data toko Shopee dengan data Instagram dan data toko Tokopedia dengan data Instagram. Perbandingan mengenai tingkat kemiripan dilakukan dengan cara membandingkan nama toko pada *e-*

Commerce dengan nama akun pada Instagram. Berikut detail mengenai uji kemiripan yang dilakukan seperti pada **Tabel 5.14.**

Tabel 5.14 Uji Kemiripan e-Commerce terhadap Instagram

Kolom e-Commerce	Kolom Instagram	Sistem Seleksi	Treeshold Kemiripan
Name	Account	Top 1	80%

5.2.5.2 Integrasi antar e-Commerce

Integrasi data yang dilakukan pada proses ini adalah mengintegrasikan data antar e-Commerce. Perbandingan mengenai tingkat kemiripan dilaukan dengan cara membandingkan nomor telfon, website, Instagram, dan nama toko. Berikut detail mengenai uji kemiripan yang dilakukan seperti pada **Tabel 5.15.**

Tabel 5.15 Uji Kemiripan anta e-Commerce

Prioritas	Kolom Tokopedia	Kolom Shopee	Sistem Seleksi	Treeshold Kemiripan
1	Phone	Phone	Top 1	96%
2	Website	Website		86%
3	instagram	Instagram	Top 1	83%
4	Name	Name		90%

Uji kemiripan dilakukan dengan menggunakan prioritas. Dimana apabila pada prioritas terkecil toko sudah dapat memenuhi *treeshold* kemiripan, maka toko akan digabungkan.

5.2.5.3 Integrasi e-Commerce dengan Google My Business

Integrasi data yang dilakukan pada proses ini adalah mengintegrasikan data e-Commerce yang telah dijelaskan pada sub-bab 5.2.5.2 dengan data toko *offline* yang didapatkan dari platform Google My Business. Perbandingan mengenai tingkat kemiripan dilaukan dengan cara membandingkan nomor telfon,

website, Instagram, dan nama toko. Berikut detail mengenai uji kemiripan yang dilakukan seperti pada **Tabel 5.16.**

Tabel 5.16 Uji Kemiripan antara Google My Business dengan e-Commerce

Prioritas	Kolom Google My Business	Kolom Tokopedia	Kolom Shopee	Sistem Seleksi	Treeshold Kemiripan
1	Tokopedia	link	-	Top 1	100%
1	Shopee	-	link		100%
2	Phone	Phone	Phone		96%
3	Website	Website	Website		86%
4	Instagram	instagram	Instagram		83%
5	Name	Name	Name		90%

5.2.5.4 Data Validation Testing

Dalam melakukan *data validation testing*, peneliti perlu terlebih dahulu untuk menentukan jumlah sampel minimal yang dapat merepresentasikan populasi data. Dalam hal ini, peneliti menggunakan metode sampling sistematik (lihat **Persamaan 5.1**) [24].

$$n = \frac{NPQ}{(N-1)D+PQ} \quad \textbf{(Persamaan 5.1)}$$

dimana, $$D = \left(\frac{B}{Z_{1-\alpha/2}} \right)^2$$

Pada metode sampling sistematis, rumus yang digunakan adalah menggunakan taksiran parameter proporsi dimana nilai P dan Q diasumsikan sebesar 0,5. D merupakan nilai konstanta dari perbandingan antara B atau batas kesalahan taksiran sebesar 0,1 dengan nilai Z. Pada penelitian kali ini, derajat error yang digunakan adalah sebesar 10%, sehingga nilai $Z_{1-\alpha/2}$ adalah sebesar 1,64.

Pada *data validation testing*, dilakukan sebanyak 3 kali validasi terhadap 3 *dataset* yang berbeda. Adapun detail dari *dataset* tersebut dapat dilihat seperti pada tabel.

Tabel 5.17 *Dataset* yang Digunakan untuk *Data Validation Testing*

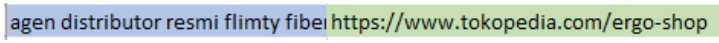
Dataset	***SumberDataset***	**Tujuan**
Akun Tokopedia yang Memiliki Akun Shopee	*Data integration* antar *e-Commerce*	Untuk mengetahui tingkat validitas kesesuaian akun toko daring antar *e-Commerce*
Toko Google My Business yang Memiliki akun Tokopedia	*Data integration* Google My Business dengan *e-commerce*	Untuk mengetahui tingkat validitas kesesuaian toko pada Google My Business dengan akun Tokopedia yang dipasangkan
Toko Google My Business yang Memiliki akun Shopee	*Data integration* Google My Business dengan *e-commerce*	Untuk mengetahui tingkat validitas kesesuaian toko pada Google My Business dengan akun Shopee yang dipasangkan

Berikut ini adalah detail jumlah populasi dan sampel setelah dilakukan perhitungan terhadap masing-masing *dataset* seperti pada tabel.

Tabel 5.18 Jumlah Populasi dan Sampel pada *Dataset*

Dataset	**Jumlah Populasi**	**Jumlah Sampel**
Akun Tokopedia yang Memiliki Akun Shopee	1313	64
Toko Google My Business yang Memiliki akun Tokopedia	186	50
Toko Google My Business yang Memiliki akun Shopee	32	32

Proses selanjutnya adalah melakukan pengecekan terhadap sampel. Berikut ini merupakan salah satu contoh proses pengecekan terhadap salah satu toko yang menjadi sampel (lihat **Gambar 5.3**).



Gambar 5.3 Nama Toko pada Google My Business dan *Link* Tokopedia

Pada **Gambar 5.3**, setelah dilakukan uji similaritas, didapatkan bahwa *link* Tokopedia tersebut cocok dengan toko pada Google My Business. Selanjutnya dilakukan validasi dengan mengunjungi alamat Tokopedia, dan membandingkan beberapa atribut yang tertera pada Tokopedia dengan atribut yang dimiliki oleh Google My Business. Demonstrasi ini ditunjukkan pada **Gambar 5.4**.

(a) Tokopedia (b) Google My Business

Gambar 5.4 Perbandingan Alamat Toko pada (a) Tokopedia dan (b) Google My Business

Seperti yang terlihat pada **Gambar 5.4**, alamat toko antara Tokopedia dan Google My Business telah sesuai. Oleh sebab itu, dapat dikatakan bahwa *link* Tokopedia tersebut valid, karena sesuai dengan toko pada Google My Business. Setelah didapatkan data perbandingan, kemudian dilakukan proses perhitungan nilai *precision* dengan persamaan seperti pada **Persamaan 5.2** [25]. *Precision* digunakan untuk menggambarkan kualitas proses *data integration* dalam mengintegrasikan antar *dataset*.

$$Precision = \frac{TP}{TP+FP} \qquad \textbf{(Persamaan 5.2)}$$

5.3 Visualisasi Data

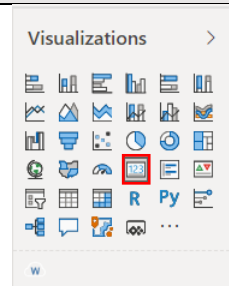
Pada sub-bab ini dijelaskan mengenai proses pembentukan visualisasi data dengan menggunakan *tools* Microsoft Power BI. Pembentukan tersebut didasarkan pada rancangan visualisasi yang tertera pada **Tabel 4.4**. Berikut merupakan rincian

implementasi untuk tiap jenis grafik yang dibangun yakni *Card, Gauge, Scatter Plot, Bar Chart, donut chart, Wordcloud* dan *Map*.

5.3.1 Card

Grafik visualisasi data berupa *card* digunakan untuk menunjukkan suatu data bersifat numerik secara ringkas dan jelas. Dalam penelitian ini, salah satu implementasi *card* digunakan untuk menampilkan informasi jumlah toko. Dengan bantuan PowerBI, visualisasi ini dapat dibangun dengan langkah yang ditunjukkan pada **Tabel 5.19**.

Tabel 5.19 Implementasi Visualisasi Card

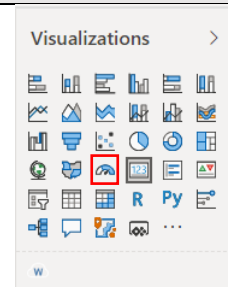
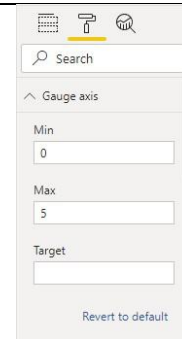
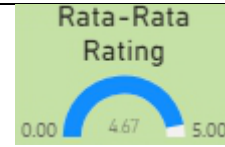
Langkah	***Screenshot***
Pada bagian *Visualizations*, pilih visualisasi card	
Masukkan *field* yang hendak ditampilkan. Dalam hal ini, visualisasi card digunakan untuk menampilkan data terkait jumlah toko	
Visualisi akan terlihat seperti pada gambar	

5.3.2 Gauge

Gauge dalam visualisasi data merupakan jenis grafik yang mematerialisasi dimana skala yang digunakan mewakili metrik, *pointer* mewakili dimensi, dan sudut *pointer* merepresentasikan

nilai. Grafik ini biasa digunakan untuk menampilkan ketercapaian dari sebuah indikator dan membandingkan interval. Implementasi *gauge* digunakan untuk menampilkan informasi mengenai rata-rata rating. Visualisasi ini dapat dibangun dengan langkah yang ditunjukkan pada **Tabel 5.20.**

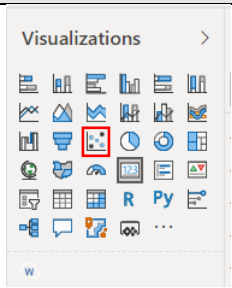
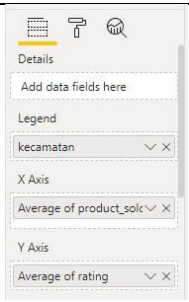
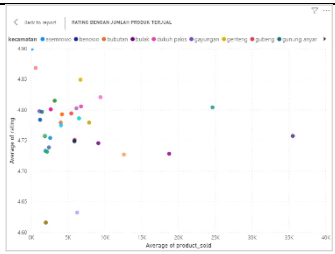
Tabel 5.20 Implementasi Visualisasi Gauge

Langkah	***Screenshot***
Pada bagian *Visualizations*, pilih visualisasi Gauge	
Masukkan *value* yang ingin ditampilkan. Dalam hal ini adalah rata-rata rating.	
Atur gauge axis agar nilai minimal dan maksimal sesuai dengan yang diharapkan. Dalam hal ini nilai minimal yang digunakan adalah 0 dan nilai maksimal sebesar 5.	
Visualisi akan terlihat seperti pada gambar	

5.3.3 Scatter Chart

Scatter Chart atau dapat disebut dengan diagram pencar adalah grafik visualisasi yang digunaknan untuk menunjukkan hubungan (korelasi) antar dua variabel. Dalam penelitian ini, salah satu implementasi *scatter chart* digunakan adalah pada variabel rating dengan jumlah produk terjual. Dengan bantuan PowerBI, visualisasi ini dapat dibangun dengan langkah yang ditunjukkan pada **Tabel 5.21.**

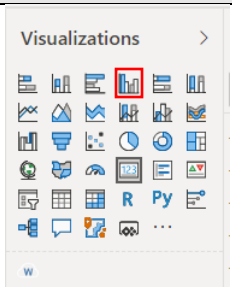
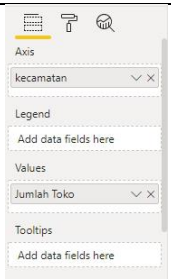
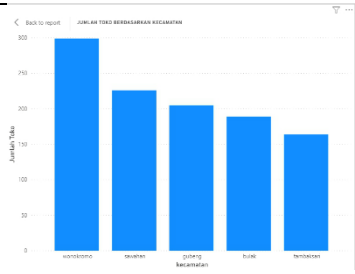
Tabel 5.21 Implementasi Visualisasi Scatter Plot

Langkah	***Screenshot***
Pada bagian *Visualizations*, pilih visualisasi Scatter plot	
Masukkan *legend*, *x-axis*, dan *y-axis* yang hendak ditampilkan. Dalam hal ini, visualisasi scatter plot digunakan untuk menampilkan data dengan *legend* berupa variabel kecamatan, garis *x-axis* berupa rata-rata penjualan produk, dan *y-axis* berupa rata-rata *rating*.	
Visualisi akan terlihat seperti pada gambar	

5.3.4 Bar Chart

Bar chart atau diagram batang merupakan grafik berbentuk batang yang biasa digunakan untuk memvisualisasikan perbandingan data. Salah satu contoh penerapan bar chart pada penelitian ini adalah untuk menampilkan informasi persebaran jumlah toko berdasarkan kecamatan. Dengan bantuan PowerBI, visualisasi ini dapat dibangun dengan langkah yang ditunjukkan pada **Tabel 5.22**.

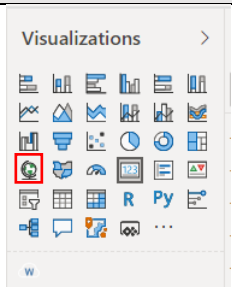
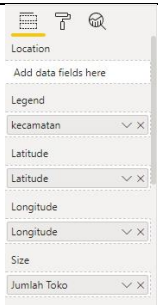
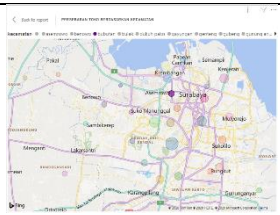
Tabel 5.22 Implementasi Visualisasi Bar Chart

Langkah	***Screenshot***
Pada bagian *Visualizations*, pilih visualisasi bar chart	
Masukkan *axis* dan *values* yang hendak ditampilkan. Dalam hal ini, visualisasi digunakan untuk menampilkan jumlah toko dengan *axis* variabel kecamatan.	
Visualisi akan terlihat seperti pada gambar	

5.3.5 Map

Visualisasi *map* digunakan untuk menunjukkan data dengan unsur geografis sehingga data dapat diinterpretasikan dengan baik. Dalam penelitian ini, salah satu implementasi *map* digunakan untuk menampilkan informasi persebaran toko berdasarkan kecamatan. Dengan bantuan PowerBI, visualisasi ini dapat dibangun dengan langkah yang ditunjukkan pada **Tabel 5.23**.

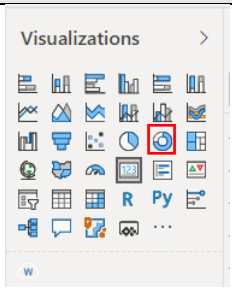
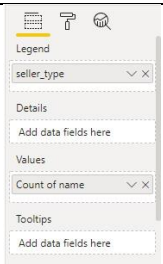
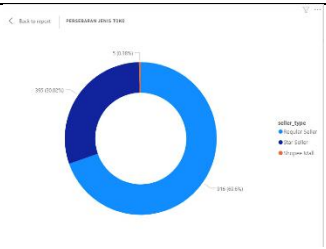
Tabel 5.23 Implementasi Visualisasi Gauge

Langkah	***Screenshot***
Pada bagian *Visualizations*, pilih visualisasi map	
Masukkan *legend*, *latitude*, *longitude*, dan *size* visualisasi yang hendak ditampilkan. Dalam hal ini, *legend* yang digunakan adalah variabel kecamatan dengan variabel 'latitude' dan 'longitude' disertai ukuran atau *size* berupa jumlah toko.	
Visualisi akan terlihat seperti pada gambar	

5.3.6 Donut Chart

Donut chart merupakan variasi dari *pie chart* yang memiliki lubang pada tengah grafik. Visualisasi ini berguna untuk menunjukkan persentase pembagian dari keseluruhan nilai dalam data. Pada penelitian ini, salah satu implementasi *donut chart* digunakan untuk menampilkan informasi persebaran jenis toko. Dengan bantuan PowerBI, visualisasi ini dapat dibangun dengan langkah yang ditunjukkan pada **Tabel 5.24.**

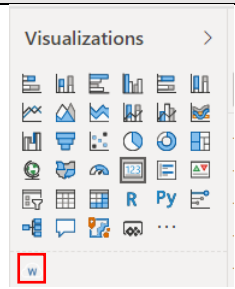
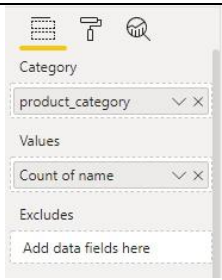
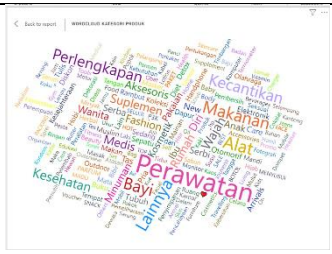
Tabel 5.24 Implementasi Visualisasi Donut Chart

Langkah	***Screenshot***
Pada bagian *Visualizations*, pilih visualisasi donut chart	
Masukkan *legend* dan *values* yang hendak ditampilkan. Dalam hal ini, variabel 'seller_type' digunakan sebagai *legend*, dan jumlah 'name' digunakan sebagai *values*.	
Visualisi akan terlihat seperti pada gambar	

5.3.7 Wordcloud

Wordcloud digunakan dalam memvisualisasikan kata-kata dengan menyoroti frekuensi kata yang digunakan dalam teks tertulis. Dalam penelitian ini, salah satu implementasi *wordcloud* digunakan untuk menampilkan informasi kategori produk. Dengan bantuan PowerBI, visualisasi ini dapat dibangun dengan langkah yang ditunjukkan pada **Tabel 5.25**.

Tabel 5.25 Implementasi Visualisasi Wordcloud

Langkah	***Screenshot***
Pada bagian *Visualizations*, pilih visualisasi word cloud. Jika pilihan word could tidak muncul, maka visualisasi ini perlu di install terlebih dahulu pada PowerBI	
Masukkan *category* dan *values* yang hendak ditampilkan. Dalam hal ini, variabel 'product_category' sebagai *category* dan jumlah 'name' menjadi nilai pada *values*.	
Visualisi akan terlihat seperti pada gambar	

BAB VI
HASIL DAN PEMBAHASAN

Bab ini membahas tentang hasil yang didapatkan dari penelitian yang dilakukan disertai pembahasan atas hasil tersebut.

6.1 Hasil Data

Pada sub-bab ini dijelaskan mengenai data yang didapatkan dari pengumpulan data dengan menggunakan teknik web-*scraping*. Penjelasan akan dibagi menjadi empat bagian sesuai dengan platform yang digunakan yaitu data Tokopedia, Shopee, Instagram, dan Google My Business.

6.1.1 Tokopedia

Terdapat beberapa jenis data yang diperoleh dari hasil *scraping* pada Tokopedia, antara lain Sub-Kategori Kesehatan, Informasi Produk Kesehatan, URL Toko Kesehatan, dan Informasi Toko Kesehatan. Hasil dari data akan terlampir pada **0**. Berikut merupakan rincian dari data-data tersebut.

6.1.1.1 Informasi Sub-Kategori Kesehatan Tokopedia

Jumlah kategori dan sub kategori kesehatan pada Tokopedia berjumlah **74**. Berikut merupakan pratinjau dari kategori dan sub kategori kesehatan pada Tokopedia (lihat **Gambar 6.1)**

kategori	link
alat laboratorium	https://www.tokopedia.com/p/kesehatan/p
alat pelangsing	https://www.tokopedia.com/p/kesehatan/p
alat penunjang gerak	https://www.tokopedia.com/p/kesehatan/p
alat pijat	https://www.tokopedia.com/p/kesehatan/p
alat terapi	https://www.tokopedia.com/p/kesehatan/p
behel	https://www.tokopedia.com/p/kesehatan/p
diaper dewasa	https://www.tokopedia.com/p/kesehatan/p
disposable consumable	https://www.tokopedia.com/p/kesehatan/p
earmuff	https://www.tokopedia.com/p/kesehatan/p
peralatan p3k	https://www.tokopedia.com/p/kesehatan/p
peralatan rumah sakit emergency	https://www.tokopedia.com/p/kesehatan/p
seragam medis	https://www.tokopedia.com/p/kesehatan/p
termometer	https://www.tokopedia.com/p/kesehatan/p

Gambar 6.1 Pratinjau Kategori Kesehatan pada Tokopedia

6.1.1.2 Informasi Produk Kesehatan Tokopedia

Proses ini menghasilkan informasi mengenai URL produk. Jumlah URL produk yang didapatkan adalah sebanyak **227963**

produk. Berikut merupakan pratinjau dari informasi produk yang didapatkan. (lihat **Gambar 6.2**)

Column1	kategori	title	link
227963	salep otot	Tiger Balm Active Muscle Rub Pain Relief Cream 60g Pereda Nyeri Otot	https://ta.tokopedia
227962	salep otot	Krem Heritage Singapore Rheuma Salve Cream cepat penghilang rasa sakit	https://ta.tokopedia
227961	salep otot	SALEP BL ORIGINAL / Cream BL / Pi Kang Wang Cream Gatal, Jerawat	https://ta.tokopedia
227960	salep otot	Counterpain 30 gram	https://ta.tokopedia
227959	salep otot	Obat Salep Wasir	https://ta.tokopedia
227958	salep otot	Cauter Kauter Pulpen Polpen Pensil Kulit Kelamin	https://www.tokope
227957	salep otot	Salep SF PKS Pi Kang Shuang Pikangsuang Phi Kang Suang Biru	https://www.tokope
227956	salep otot	PI KANG SHUANG/SALEP GATAL/SALEP KULIT GATAL/SALEP MULTIFUNGSI	https://www.tokope

Gambar 6.2 Pratinjau Data Produk Kesehatan pada Tokopedia

6.1.1.3 Informasi URL Toko Kesehatan Tokopedia

Pada proses ini, didapatkan sebanyak **20391** URL toko unik. Berikut merupakan pratinjau dari informasi URL toko yang didapatkan. (lihat **Gambar 6.3)**

1	Column1	Column2
2	link	
20370	https	//www.tokopedia.com/lunoxstore-1
20371	https	//www.tokopedia.com/twentystore-1
20372	https	//www.tokopedia.com/toengmarket
20373	https	//www.tokopedia.com/romeomart
20374	https	//www.tokopedia.com/elzadashop
20375	https	//www.tokopedia.com/ceugunk14
20376	https	//www.tokopedia.com/best16
20377	https	//www.tokopedia.com/ibnuhafiz
20378	https	//www.tokopedia.com/eudora-store
20379	https	//www.tokopedia.com/kajucaci
20380	https	//www.tokopedia.com/tigabintangracer
20381	https	//www.tokopedia.com/owwidshop
20382	https	//www.tokopedia.com/evanderjaya84
20383	https	//www.tokopedia.com/vistaniaga
20384	https	//www.tokopedia.com/ulihume
20385	https	//www.tokopedia.com/lion-star
20386	https	//www.tokopedia.com/sheva30

Gambar 6.3 Informasi URL Toko Kesehatan di Tokopedia

6.1.1.4 Informasi Toko Kesehatan Tokopedia

Dari **20391** URL toko yang didapatkan, kemudian dilakukan pengambilan informasi toko secara lebih lengkap. Berikut ini merupakan pratinjau dari hasil *scraping* informasi toko pada Tokopedia. (lihat **Gambar 6.4**)

name	link	seller_type	city
Liabeauty	https://www.tokop	Power Merchant	Kota Surabaya
Laristarshop	https://www.tokop	Power Merchant	Kota Surabaya
Fera_allshop	https://www.tokop	Power Merchant	Kota Surabaya
citybeautyshop	https://www.tokop	Power Merchant	Kota Surabaya
wijayaaaa	https://www.tokop	Power Merchant	Kota Surabaya
WerehouseDSS	https://www.tokop	Power Merchant	Kota Surabaya
coeyz	https://www.tokop	Power Merchant	Kota Surabaya
Bawang123	https://www.tokop	Power Merchant	Kota Surabaya
Beauuuti.ful	https://www.tokop		Tambaksari
Ailershop	https://www.tokop	Power Merchant	Kota Surabaya
adik laris88	https://www.tokop	Power Merchant	Kota Surabaya
Fitri Beauty Care	https://www.tokop	Power Merchant	Kota Surabaya
bagauthetic	https://www.tokop	Power Merchant	Kota Surabaya
Organic5	https://www.tokop	Power Merchant	Kota Surabaya
solagratia777	https://www.tokop	Power Merchant	Kota Surabaya
BearWithUs	https://www.tokop	Power Merchant	Kota Surabaya
Dxtra collection	https://www.tokop	Power Merchant	Kota Surabaya
SILA HERBAL PASUTI	https://www.tokop		Kota Surabaya

Gambar 6.4 Pratinjau Informasi Toko pada Tokopedia

6.1.2 Shopee

Terdapat beberapa jenis data yang diperoleh dari hasil *scraping* pada Shopee, antara lain Sub-Kategori Kesehatan, Informasi Produk Kesehatan, URL Toko Kesehatan, dan Informasi Toko Kesehatan. Hasil dari data akan terlampir pada **0**. Berikut merupakan rincian dari data-data tersebut.

6.1.2.1 Sub-Kategori Kesehatan Shopee

Jumlah kategori dan sub kategori kesehatan pada Shopee berjumlah **5 kategori**. Berikut merupakan pratinjau dari kategori dan sub kategori kesehatan pada Shopee (lihat **Gambar 6.5**).

A	B	C	D	E	F	G	H
	kategori	link					
0	kesehatar	https://shopee.co.id/Kesehatan-cat.14780					
1	kesehatar	https://shopee.co.id/Kesehatan-Seksual-cat.14780.14795					
2	suplemen	https://shopee.co.id/Suplemen-Makanan-cat.14780.14783					
3	alat medi	https://shopee.co.id/Alat-Medis-cat.14780.14798					
4	perawatar	https://shopee.co.id/Perawatan-Diri-cat.14780.14789					

Gambar 6.5 Pratinjau Kategori Kesehatan pada Shopee

6.1.2.2 Informasi Produk Kesehatan Shopee

Dari kategori yang dimiliki, dilakukan pencarian produk terhadap kategori sehingga menemukan sebanyak **7409 produk**. Pratinjau mengenai informasi produk dapat dilihat pada **Gambar 6.6**

Column1	kategori	title	link
7409	perawatan diri	Hot In Koyo AromaTherapy 10's	https://shopee.co.id/Hot-I
7408	perawatan diri	SYNALTEN CREAM	https://shopee.co.id/SYNA
7407	perawatan diri	jabet javabet original nasa - obat diabetes herbal system nano	https://shopee.co.id/jabet-
7406	perawatan diri	Aroma Terapi (Minyak Angin) Bidara Ruqyah Tsumma Tawakkal / FreshCare	https://shopee.co.id/Arom
7405	perawatan diri	Krim delapan-delapan	https://shopee.co.id/Krim-
7404	perawatan diri	Softex Comfort Slim Pembalut Wing 23 cm 8s	https://shopee.co.id/Softe
7403	perawatan diri	Miconazole Cream 10 gr	https://shopee.co.id/Micor
7402	perawatan diri	[BJ] PASTA GIGI NASA (PGN) Perlindungan Gigi dan Gusi	https://shopee.co.id/-BJ-P
7401	perawatan diri	PIGEON TOOTHPASTE FOR CHILDREN / PASTA GIGI ANAK PIGEON 45g	https://shopee.co.id/PIGE
7400	perawatan diri	Dettol antiseptic cair 95 ml...original	https://shopee.co.id/Detto
7399	perawatan diri	BIOAQUA one spring Krim Pembesar / Pengencang Payudara	https://shopee.co.id/BIOA
7398	perawatan diri	NCX nasa original / hilangi keputihan / promil program hamil / obat nyeri haid	https://shopee.co.id/NCX-
7397	perawatan diri	Pepsodent herbal 190gr	https://shopee.co.id/Peps
7396	perawatan diri	Listerine Antiseptic Mouthwash - 100ml	https://shopee.co.id/Lister
7395	perawatan diri	handsanitizer A&G hand moisturizer 100ml, richardson, nuvo, antis	https://shopee.co.id/hands
7394	perawatan diri	rose v - masalah kewanitaan	https://shopee.co.id/rose-

Gambar 6.6 Pratinjau Informasi Produk Kesehatan di Shopee

6.1.2.3 URL Toko Kesehatan Shopee

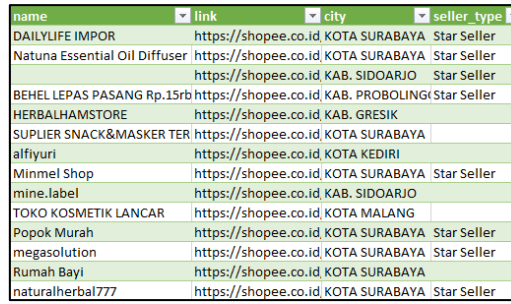
Pada proses ini, dilakukan ekstraksi data URL Toko pada halaman produk, lalu dilakukan *pre-processing* untuk menghilangkan redundansi data. Jumlah URL toko yang didapatkan pada proses ini adalah sebanyak **3764 toko**. Pratinjau mengenai informasi URL toko dapat dilihat pada **Gambar 6.7**.

link
https://shopee.co.id/dailylife.id
https://shopee.co.id/natunaessential
https://shopee.co.id/mbakat0n
https://shopee.co.id/rl.casemurah
https://shopee.co.id/ilhamnandakurnia
https://shopee.co.id/jajaneenaksby
https://shopee.co.id/alfiyuri
https://shopee.co.id/rionbabyshop
https://shopee.co.id/gosyencookies
https://shopee.co.id/lapakkosmetikmalang
https://shopee.co.id/karina1294

Gambar 6.7 Pratinjau URL Toko Kesehatan di Shopee

6.1.2.4 Informasi Toko Kesehatan Shopee

URL toko yang didapatkan dari proses sebelumnya kemudian menjadi *input* untuk proses ini. Pada proses ini, dilakukan pengambilan data terkait dengan informasi toko. Berikut merupakan pratinjau dari informasi toko yang didapatkan (lihat **Gambar 6.8**).



name	link	city	seller_type
DAILYLIFE IMPOR	https://shopee.co.id	KOTA SURABAYA	Star Seller
Natuna Essential Oil Diffuser	https://shopee.co.id	KOTA SURABAYA	Star Seller
	https://shopee.co.id	KAB. SIDOARJO	Star Seller
BEHEL LEPAS PASANG Rp.15rb	https://shopee.co.id	KAB. PROBOLING(	Star Seller
HERBALHAMSTORE	https://shopee.co.id	KAB. GRESIK	
SUPLIER SNACK&MASKER TER	https://shopee.co.id	KOTA SURABAYA	
alfiyuri	https://shopee.co.id	KOTA KEDIRI	
Minmel Shop	https://shopee.co.id	KOTA SURABAYA	Star Seller
mine.label	https://shopee.co.id	KAB. SIDOARJO	
TOKO KOSMETIK LANCAR	https://shopee.co.id	KOTA MALANG	
Popok Murah	https://shopee.co.id	KOTA SURABAYA	Star Seller
megasolution	https://shopee.co.id	KOTA SURABAYA	Star Seller
Rumah Bayi	https://shopee.co.id	KOTA SURABAYA	
naturalherbal777	https://shopee.co.id	KOTA SURABAYA	Star Seller

Gambar 6.8 Pratinjau Informasi Toko Kesehatan di Shopee

6.1.3 Instagram

Pada proses ini dilakukan pencarian akun Instagram berdasarkan kata kunci nama toko pada Shopee dan Tokopedia sehingga menghasilkan **128439** dan **235603** akun Instagram. Hasil dari data akan terlampir pada **0**. Berikut merupakan pratinjau dari akun Instagram yang didapat. (lihat **Gambar 6.9**).

store	city	account
rakioss	Surabaya	#rakiss
rakioss	Surabaya	Rakiss Glam Hub
rakioss	Surabaya	rak.issac
rakioss	Surabaya	rakiss3157
rakioss	Surabaya	rakisslelion
rakioss	Surabaya	rakio.ss
rakioss	Surabaya	rakisslove
rakioss	Surabaya	rakissglam
rakioss	Surabaya	rock_rakiss
rakioss	Surabaya	rakissm_1_2_3
rakioss	Surabaya	rakissa_mimo
rakioss	Surabaya	queen_rakiss
rakioss	Surabaya	rakis_seytu
rakioss	Surabaya	rakiosssylla
rakioss	Surabaya	rakiss
rakioss	Surabaya	rakissd
rakioss	Surabaya	rakiss4154
rakioss	Surabaya	rakibul8614
rakioss	Surabaya	rakissrock
rakioss	Surabaya	rak.issac0
rakioss	Surabaya	rakissa.thomas
rakioss	Surabaya	rakisspammm
rakioss	Surabaya	rakisshaiva

Gambar 6.9 Pratinjau Akun Instagram Berdasarkan Kata Kunci Nama Toko

6.1.4 Google My Business

Pada proses ini, informasi toko mengenai Google My business didapatkan dengan cara pencarian menggunakan kata kunci. Hasil yang didapatkan adalah sebanyak **142529 toko**. Namun, karena pencarian toko didasarkan pada pencarian dengan menggunakan kata kunci, maka informasi yang didapatkan sangat rawan tidak sesuai dengan kebutuhan. Oleh sebab itu, setelah dilakukan *pre-process* terhadap informasi toko Google My Business, didapatkan toko yang sesuai dengan kebutuhan adalah sebanyak **5534 toko**. Hasil dari data akan terlampir pada **0**. Berikut merupakan pratinjau dari informasi toko sebelum dan sesudah dilakukan *pre-process* (lihat **Gambar 6.10** dan **Gambar 6.11**).

name	website	rating	numOf_google_review
Worldwide Adult Daycare	http://www.worldwideadultdaycare.com/	4.4	8 Google reviews
Alpine Adult Day Care	http://www.alpineadultdaycare.com/	4.6	10 Google reviews
McCoy Adult Day Care	http://www.mccoyadultdaycare.com/	4.8	21 Google reviews
SSM Health Adult Day Health Center	https://www.ssmhealth.com/locations/location-details/adult-day-health-center?utm_id=gmb-wi-5&hgcrm_channel=sms_text&hgcrm_source=healthgrades&hgcrm_agency=client&hgcrm_campaignid=1642&hgcrm_tacticid=2664&hgcrm_trackingsetid=4067	3	2 Google reviews
Adult Day Services		5	1 Google review
Ebenezer Minneapolis Adult Day Program	http://www.ebenezercares.org/ebenezer-minneapolis-adult-day-program.html	5	1 Google review
Buffalo Adult Day Center	http://buffaloadultdaycenter.org/		
Community Adult Day Center	http://communityadultdaycenter.org/	4.8	5 Google reviews

Gambar 6.10 Pratinjau Data Toko Google My Business Hasil *Scraping*

name	website	rating	numOf_google_review
guyub rukun	website not found		
kemuning	website not found		
buah hati	website not found		
the nursery "dian"	https://tempat-penitipan-anak-dian.business.site/?utm_source=gmb&utm_medium=referral		
bekam barokah	website not found	5	2
ariadi hipnoterapi surabaya	https://ariadi-hipnoterapi-surabaya.business.site/	3.3	3
agen ck & assikha surabaya	website not found	5	10
herbiotic-100 agen resmi surabaya	website not found	5	4
pusat hipnoterapi surabaya	http://www.hypnoterapisurabaya.com/	5	1
herbal clinic "tefaron"	http://kliniktefaron.blogspot.co.id/2010/06/hemerrhoids.html	4.5	2
herbai alami	https://herbai-alami.business.site/?utm_source=googlemybusiness&utm_medium=referral		
griya teraphy "bekam"	website not found		
oxybaric clinic	http://www.oxibaric.com/	4.5	2

Gambar 6.11 Pratinjau Data Toko Google My Busines Setelah Pre-Process

6.1 Data Integration

Pada data integration, dilakukan integrasi seperti pada tahap rancanngan dan hasil. Intergasi dilakukan dalam 4 tahap, yakni integrasi *e-commerce* dengan Instagram sebanyak dua kali,

integrasi antar *e-commerce*, dan integrasi *e-commerce* dengan Google My Business. Hasil dari data akan terlampir pada **0**.

Selain itu, pada tahap ini juga dilakukan proses *data validation* testing. Berikut ini merupakan hasil dari *data validation testing* yang digambarkan dengan nilai *precision*. Nilai dari masing-masing *dataset* yang dapat dilihat pada **Tabel 6.1**.

Tabel 6.1 Nilai *Precision* dari *Dataset*

Dataset	**TP**	**FP**	***Precision***
Akun Tokopedia yang Memiliki Akun Shopee	37	27	58%
Toko Google My Business yang Memiliki akun Tokopedia	23	27	46%
Toko Google My Business yang Memiliki akun Shopee	22	10	69%

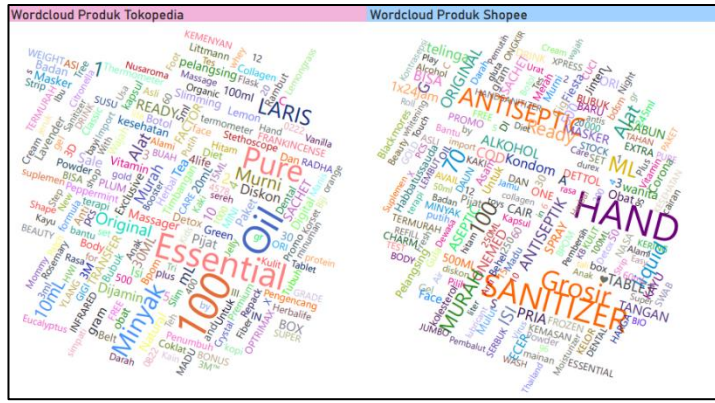
Dari beberapa nilai *precision* tersebut, didapat skor rata-rata 58%. Hasil tersebut menggambarkan kualitas proses *data integration* yang dilakukan.

6.2 Hasil Visualisasi

Pada sub-bab ini dijelaskan mengenai hasil visualisasi data yang telah dibangun. Berikut ini merupakan hasil dari visualisasi data yang telah dilakukan.

6.2.1 Exploratory Data Analysis

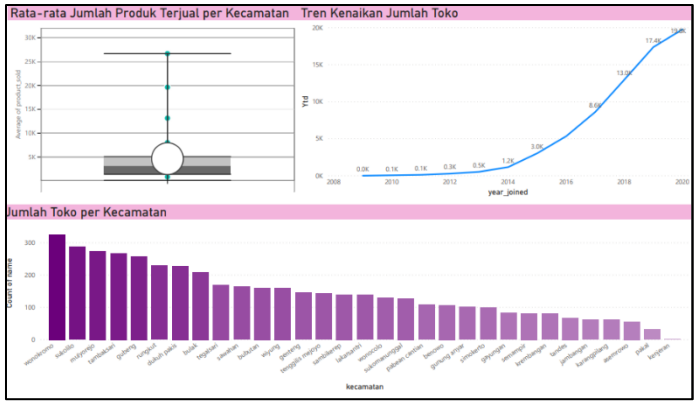
Berikut ini merupakan *wordcloud* yang menunukkan perbandingan dari produk kesehatan yang dijual pada Tokopedia dan Shopee (lihat **Gambar 6.12**). Pada grafik tersebut terlihat bahwa produk kesehatan yang dijual pada Tokopedia lebih banyak merupakan produk berupa ‘essential oil’, sedangkan produk yang banyak dijual pada Shopee lebih bayak merupakan produk berupa ‘hand sanitizer’.



Gambar 6.12 Wordcloud Perbandingan Produk Kesehatan pada Tokopedia dan Shopee

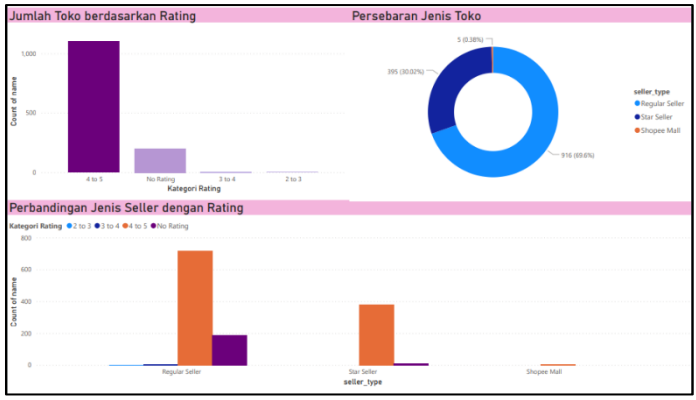
Berikut ini merupakan hasil dari proses EDA pada data Tokopedia (lihat **Gambar 6.13**). Dari gambar tersebut didapatkan informasi bagaimana tren kenaikan jumlah toko dimana bentuk grafiknya hampir mendekati grafik eksponensial. Artinya, semakin hari akan semakin banyak UMKM bidang kesehatan yang hendak mendayagunakan platform daring khsusunya adalah Tokopedia.

Fakta lain yang didapatkan adalah mengenai bagaimana persebaran toko yang ada di Surabaya, dimana Kecamatan Wonokromo dan Sukolilo menjadi kecamatan dengan jumlah toko terbanyak. Namun, grafik ini tidak bisa menampilkan keseluruhan toko, karena mayoritas toko pada Tokopedia tidak menyertakan alamat kecamatan. Rata-rata penjualan produk berdasarkan kecamatan adalah 4651 produk yang dapat dilihat pada grafik *box plot*. Meskipun begitu, terdapat data *outlier* seperti Kecamatan Tambaksari yang memiliki rata-rata penjualan produk sebesar 26673 produk.



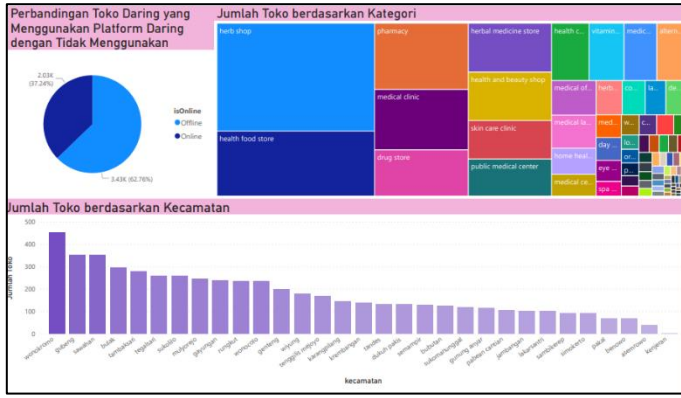
Gambar 6.13 Hasil Proses EDA terhadap Data Tokopedia

Berikut ini merupakan hasil dari proses EDA pada data Shopee (lihat **Gambar 6.14**). Dari grafik 'jumlah toko berdasarkan rating', didapatkan informasi bahwa 1103 atau sekitar 30 persen dari toko yang menjual produk kesehatannya di Shopee, sudah mendapatkan rating diatas 4. Dan dari grafik 'persebaran jenis toko', didapatkan informasi bahwa mayoritas toko yang menjual produk kesehatan adalah merupakan toko dengan jenis *regular seller*. Fakta lain yang didapatkan adalah bahwa tidak ada toko dengan kategori *star seller* dan *shopee mall* yang memiliki rating dibawah 4.



Gambar 6.14 Hasil Proses EDA terhadap Data Shopee

Berikut ini merupakan hasil dari proses EDA pada data Google My Business (lihat **Gambar 6.15**). Dari grafik ‘jumlah toko berdasarkan kecamatan’, didapatkan informasi bahwa Kecamatan Wonokromo menjadi kecamatan dengan jumlah toko terbanyak. Selain itu, jika jumlah toko dilihat dari segi kategori, toko dengan dengan kategori ‘herb shop’ menjadi toko dengan jumlah terbanyak.

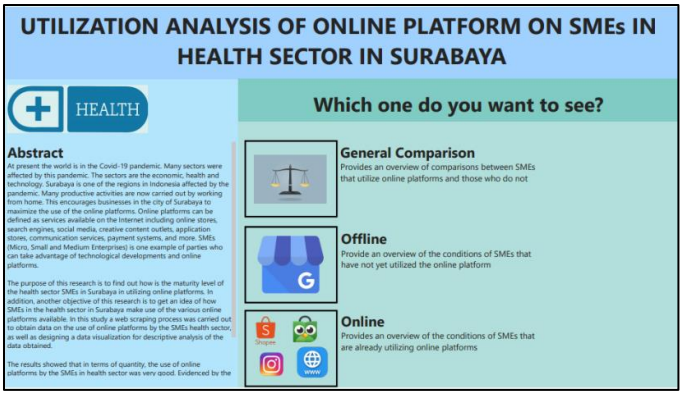


Gambar 6.15 Hasil Proses EDA pada Data Google My Business

Toko pada Google My Business adalah toko yang memiliki jalur non daring. Fakta yang bisa didapatkan dari data Google My Business adalah mengenai pemanfaatan platform daring dari toko yang memiliki jalur non daring. Jika dilihat pada *pie chart* yang pada pada **Gambar 6.15**, didapatkan informasi bahwa 37 persen atau sekitar 2000 toko dari 5400 toko yang ada, sudah memanfaatkan platform daring untuk menunjang kegiatan bisnis.

6.2.2 Dashboard

Dashboard disusun sesuai dengan *metric* yang telah dirancang. Halaman utama dari *dashboard* dapat dilihat seperti pada **Gambar 6.16.**



Gambar 6.16 Halaman Utama Dashboard

Pada bagian *general comparison* (lihat **Gambar 6.17**), diberikan visualisasi mengenai perbandingan secara umum mengenai kondisi toko pada bidang kesehatan dalam memanfaatkan platform daring. Data menunjukkan bahwa era sekarang banyak toko yang memanfaatkan platform daring dalam memfasilitasi kegiatan jual beli tanpa harus memiliki lapak fisik. Hal ini dibuktikan dengan tingginya jumlah toko daring yang ada (19770 toko) dibanding dengan jumlah toko non-daring (5462 toko). Dan jika dilihat, 37 % dari toko yang memiliki jalur non-daring juga memanfaatkan platform daring dalam meningkatkan daya saing tokonya.

Gambar 6.17 Dashboard bagian 'General Comparison'

Pada bagian toko daring Shopee (lihat **Gambar 6.18**), diberikan visualisasi mengenai kondisi toko bidang kesehatan yang menggunakan Shopee untuk membantu kegiatan jual beli. Jumlah toko bidang kesehatan di Surabaya yang menggunakan Shopee adalah sebanyak 1316 toko. Dari toko tersebut, kebanyakan telah menjual produk dengan kategori 'perawatan', 'perlengkapan', 'makanan', 'kecantikan', dan 'suplemen' (dilihat dari kata yang sering muncul pada *wordcloud*). Artinya secara implisit didapatkan informasi bahwa toko-toko daring tersebut tidak hanya menjual produk-produk kesehatan saja. Rata-rata jumlah katalog produk pada Shopee adalah 280 produk, yang mana merupakan angka yang cukup besar.

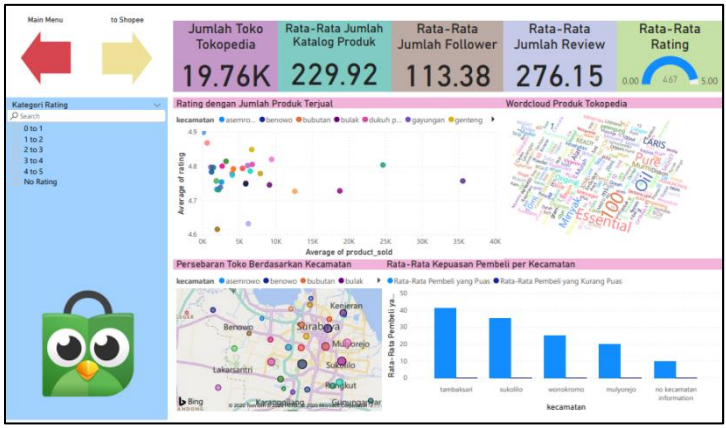
Rata-rata jumlah follower toko kesehatan tersebut adalah 5800 follower. Mayoritas toko tersebut adalah *regular seller*. Sehingga, secara implisit didapatkan informasi bahwa toko-toko tersebut belum begitu memaksimalkan pemanfaatan platform Shopee, karena melihat dari fakta follower yang cukup banyak, namun tidak banyak diantara toko yang mengambil layanan *star seller* (hanya 30 persen dari jumlah toko secara keseluruhan).

Gambar 6.18 Dashboard untuk Data Toko Daring (Shopee)

Pada bagian toko daring Tokopedia (lihat **Gambar 6.19**), diberikan visualisasi mengenai kondisi toko bidang kesehatan

yang menggunakan Tokopedia untuk membantu kegiatan jual beli. Jumlah toko bidang kesehatan di Surabaya yang menggunakan Tokopedia adalah sebanyak 19760 toko. Dengan banyaknya jumlah toko tersebut, rata-rata follower pada semua toko hanya berkisar 113 follower saja. Rata-rata jumlah katalog produk pada Tokopedia adalah 229 produk, yang mana merupakan angka yang cukup besar.

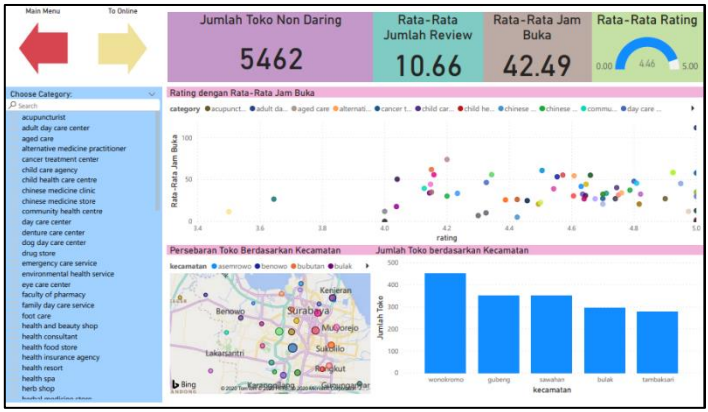
Fakta menarik lainnya adalah bagaimana korelasi antara rating dengan jumlah produk yang dijual. Semakin sedikit produk yang dijual oleh toko tersebut, semakin bagus rating tokonya. Selain itu, kebanyakan dari produk yang dijual pada Tokopedia adalah produk berupa 'essential oil'.



Gambar 6.19 Dashboard untuk Data Toko Daring (Tokopedia)

Pada bagian toko non-daring (lihat **Gambar 6.20**), diberikan visualisasi mengenai kondisi toko bidang kesehatan yang melakukan aktivitas jual beli secara non-daring. Jumlah toko bidang kesehatan di Surabaya pada kategori ini adalah sebanyak 5462 toko. Fakta yang didapatkan, persebaran toko non-daring kesehatan adalah paling banyak berada di kawasan Surabaya Selatan, yakni Kecamatan Wonokromo dan Sawahan. Rata-rata jumlah jam buka toko kesehatan non daring di Surabaya dalam seminggu adalah 42.49 jam.

Fakta lainnya adalah rata-rata jumlah *review* pada setiap toko adalah sebanyak 10 *review*, dengan rata-rata skor rating adalah 4.46 dari 5. Selain itu, didapatkan informasi bahwa tidak ada korelasi yang kuat antara *rating* dengan jumlah jam buka.



Gambar 6.20 Dashboard untuk Data Toko Non Daring

BAB VII
PENUTUP

Bab ini membahas kesimpulan penelitian yang telah dilakukan dan saran yang diusulkan. Kesimpulan ini diharapkan dapat menjawab rumusan masalah penelitian dan saran untuk penelitian serupa di masa mendatang

7.1 Kesimpulan

Proses yang telah dilalui dalam penelitian ini adalah pengambilan data menggunakan teknik *web scraping*, melakukan *pre-processing* terhadap data yang didapat, hingga dilakukan proses integrasi data. Tidak hanya itu, data yang telah diintegrasi juga telah diuji validitasnya dan digambarkan dalam bentuk skor *precision*. Data yang telah terintegrasi selanjutnya dilakukan EDA (*Exploratory Data Analysis*), menentukan *metric,* hingga membuat visualisasi data dalam bentuk *dashboard.*

Penelitian ini memiliki tujuan utama yakni adalah untuk mengetahui bagaimana tingkat pendayagunaan plaftom daring oleh UMKM bidang kesehatan di Surabaya. Selain itu, penelitian ini juga memiliki tujuan untuk memvisualisasikan data UMKM bidang kesehatan di Surabaya.

Dalam melakukan penelitian, ada pembelajaran yang didapatkan. Berikut ini merupakan pembelajaran yang didapatkan, antara lain:

1. Setiap platform daring seperti Tokopedia, Shopee, dan Google My Business memiliki ketersediaan data yang berbeda-beda. Sehingga perlu dilakukan penyesuaian terhadap *metric* yang hendak disusun.
2. Keterbatasan terhadap data fakta yang bisa dieksplor, sehingga sulit untuk dilakukan analisis. Tindak lanjut yang peneliti lakukan adalah mencari data fakta yang paling relevan terhadap *metric* yang telah disusun.

Data yang telah didapat dari proses *web scraping* serta dilakukan integrasi memiliki tingkat validitas sebesar 57%

(didapatkan dari rata-rata skor *precision*). Berikut ini merupakan hasil dari penelitian yang dilakukan, antara lain:

1. UMKM bidang kesehatan adalah UMKM yang menjual produk dan layanan kesehatan, dimana produk yang paling banyak dijual pada toko daring Tokopedia dan Shopee adalah produk 'essential oil' dan 'hand sanitizer'.
2. Tren kenaikan jumlah toko pada platform daring meningkat mendekati tren eksponensial. Hal ini dapat dibuktikan dengan grafik tren kenaikan jumlah toko pada toko daring Tokopedia.
3. UMKM bidang kesehatan yang melakukan aktivitas melalui jalur non daring adalah sebanyak 5462 toko. Rata-rata jumlah *review* pada setiap tokonya adalah sebanyak 10 *review* dan rata-rata skor *rating* 4.46 dari 5. Selain itu, rata-rata jam buka UMKM bidang kesehatan yang melakukan aktivitas melalui jalur non daring adalah 42 jam dalam seminggu.
4. UMKM sudah memanfaatkan platform daring untuk menjalankan aktivitas jual belinya, bahkan tanpa harus memiliki lapak secara fisik. Hal ini dibuktikan dengan besarnya jumlah toko daring (19770 toko) dibandingkan dengan jumlah toko non daring (5462 toko). Jika dilihat, 37 % dari toko yang memiliki jalur non-daring juga memanfaatkan platform daring dalam meningkatkan daya saing tokonya. Selain itu, rata-rata jumlah katalog produk yang dijual pada toko daring baik Tokopedia maupun Shopee adalah lebih dari 200 produk. Artinya, pemanfaatan platform daring oleh UMKM bidang kesehatan, secara kuantitas sudah sangat baik.
5. Jika dilihat dari segi kualitas mengenai bagaimana UMKM bidang kesehatan memanfaatkan penggunaan platform daring, ternyata belum cukup baik. Hal ini dibuktikan dengan minimnya rata-rata *followers* yang dimiliki oleh toko tersebut (pada *e-commerce* Tokopedia). Di samping itu, ada toko-toko yang memiliki *follower* cukup tinggi, namun belum memaksimalkan fitur yang *e-commerce* tawarkan seperti fitur *star seller*.

Hal ini dibuktikan dengan jumlah toko dengan kategori *regular seller* yang masih banyak yakni 69.6 persen.

7.2 Saran

Dalam pengerjaan tugas akhir, terdapat beberapa saran yang diharapkan dapat bermanfaat untuk pengembangan penelitian ke depan yang dijelaskan sebagai berikut.

1. Perlu dilakukan analisis secara prediktif dan atau perskriptif mengenai bagaimana pemanfaatan platform daring oleh UMKM bidang kesehatan di Surabaya
2. Perlu diadakan penelitian mengenai *threshold* dari uji similaritas untuk mengukur validitas agar dapat menghasilkan data dengan kuantitas dan tingkat validitas yang baik.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] "WHO Director-General's opening remarks at the media briefing on COVID-19 - 11 March 2020." https://www.who.int/dg/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020 (accessed May 21, 2020).

[2] R. Falefi and A. Purwoko, "Impact of COVID- 19 ' s Pandemic on the Economy of Indonesia," pp. 1147–1156, 2020.

[3] "Pengguna Internet Kala WFH Corona Meningkat 40 Persen di RI," Apr. 09, 2020. https://www.cnnindonesia.com/teknologi/20200408124947-213-491594/pengguna-internet-kala-wfh-corona-meningkat-40-persen-di-ri (accessed May 21, 2020).

[4] G. Grasselli, A. Pesenti, and M. Cecconi, "Critical Care Utilization for the COVID-19 Outbreak in Lombardy, Italy: Early Experience and Forecast During an Emergency Response," *JAMA*, vol. 323, no. 16, pp. 1545–1546, Apr. 2020, doi: 10.1001/jama.2020.4031.

[5] Z. Zahrotunnimah, "Langkah Taktis Pemerintah Daerah Dalam Pencegahan Penyebaran Virus Corona Covid-19 di Indonesia," *SALAM J. Sos. dan Budaya Syar-i*, vol. 7, no. 3, 2020, doi: 10.15408/sjsbs.v7i3.15103.

[6] "What is an 'online platform'?," in *An Introduction to Online Platforms and Their Role in the Digital Transformation*, OECD, 2019, pp. 19–26.

[7] E. Vargiu and M. Urru, "Exploiting web scraping in a collaborative filtering- based approach to web advertising," *Artif. Intell. Res.*, vol. 2, no. 1, pp. 44–54, 2012, doi: 10.5430/air.v2n1p44.

[8] A. Mas, "RANCANG BANGUN ANALISIS TREN PRODUK PADA SITUS PASAR ONLINE BERDASARKAN RANKING PENJUALAN," ITS,

2019.

[9] F. Xu, Z. Pan, and R. Xia, “E-commerce product review sentiment classification based on a naïve Bayes continuous learning framework,” *Inf. Process. Manag.*, no. August 2019, p. 102221, 2020, doi: 10.1016/j.ipm.2020.102221.

[10] I. Setiawati and P. Widyartati, “Pengaruh Strategi Pemasaran Online terhadap Peningkatan Laba Umkm,” *Strateg. Komun. Pemasar.*, no. 20, pp. 1–5, 2017, [Online]. Available: file:///C:/Users/BAYU/Downloads/Documents/263-760-1-PB.pdf.

[11] H. A. Sarwono, “Profil Bisnis Usaha Mikro, Kecil Dan Menengah (Umkm),” *Bank Indones. dan LPPI*, pp. 1–135, 2015.

[12] M. Ayyagari, T. Beck, and A. Demirguc-Kunt, “Small and medium enterprises across the globe,” *Small Bus. Econ.*, vol. 29, no. 4, pp. 415–434, 2007, doi: 10.1007/s11187-006-9002-5.

[13] R. Ahmad, R. Khan, A. Nadeem, and A. Ali, “Business Analytics: A Framework,” vol. 10, 2019.

[14] M. J. Beller and A. Barnett, “Next Generation Business Analytics,” 2009.

[15] N. Raden, “The Foundations of Analytics:Visualization, Interactivity and Utility. The ten principles of Enterprise Analytics,” *Spotfire, Inc*, 2010.

[16] R. Saxena and A. Srinivasan, *Business Analytics-A Practitioner’s Guide*. Springer, 2013.

[17] B. Zhao, “Web Scraping,” *Encycl. Big Data*, no. December, 2020, doi: 10.1007/978-3-319-32001-4.

[18] D. S. Ebert, J. M. Favre, and R. Peikert, “Data visualization,” *Comput. Graph.*, vol. 26, no. 2, pp. 207–208, 2002, doi: 10.1016/S0097-8493(02)00051-1.

[19] J. R. Talburt, *Entity Resolution and Information Quality*. 2011.

[20] "Metrics Definition." https://www.investopedia.com/terms/m/metrics.asp (accessed Jun. 09, 2020).

[21] "7 Types Of Metrics - Simplicable." https://simplicable.com/new/types-of-metrics (accessed Jun. 09, 2020).

[22] U. Yusuf, "ANALISIS PEMANFAATAN PLATFORM DARING PADA USAHA MIKRO KECIL MENENGAH BIDANG MAKANAN DAN MINUMAN DI SURABAYA," Institut Teknologi Sepuluh Nopember, 2020.

[23] S. D. Faisal, "ANALISIS PEMANFAATAN KANAL DIGITAL DALAM PROSES PENJUALAN PADA UMKM KATEGORI FASHION MUSLIM DI SURABAYA," Institut Teknologi Sepuluh Nopember, 2020.

[24] R. L. Scheaffer, *Elementary Survey Sampling: 7th (Seventh) Edition*, 7th ed. Cengage Learning, 2011.

[25] "How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification." https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/ (accessed Jun. 01, 2020).

Halaman ini sengaja dikosongkan.

LAMPIRAN A. KODE PROGRAM

Pada lampiran ini ditunjukkan kode-kode program yang dijalankan dalam melakukan *web-scraping* serta pra-proses data.

a. Kode *Scraper*

Pada bagian ini dijelaskan kode yang digunakan dalam proses *web-scraping* untuk masing-masing platform yakni Tokopedia, Shopee, Instagram dan Google My Business.

1. Tokopedia

Tabel A.1 Kode Program *Scraper* untuk Sub Kategori Tokopedia

Scraper Sub Kategori Tokopedia	
Fungsi	Melakukan *web-sraping* untuk mencari daftar sub-kategori di Tokopedia
Bahasa	Python

```
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionCh
ains
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions a
s EC
import time
from selenium.webdriver.firefox.options import Options

df = pd.DataFrame(columns = ['kategori','link'])

list_kategori = []
list_url = []

# ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/ALDYSYAH/OneDrive - Institut Teknologi Sepul
uh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
#driver = webdriver.Chrome(executable_path='C:/Users/ALDYSYA
H/OneDrive - Institut Teknologi Sepuluh Nopember/Semester 8/
TA/Scraping/TA-Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()
#driver Firefox headless
# options = Options()
# options.headless = True
```

Scraper Sub Kategori Tokopedia

```
# # fp = webdriver.FirefoxProfile('F:/Temp')
# driver = webdriver.Firefox(options=options, executable_pat
h=r'D:\web driver\geckodriver.exe')

driver.get('https://www.tokopedia.com/')

time.sleep(10)

wait_kategori_button = WebDriverWait(driver, 3).until(EC.vis
ibility_of_element_located((By.XPATH, '//*[@class="css-
vk082c"]')))
ActionChains(driver).move_to_element(wait_kategori_button).p
erform()

WebDriverWait(driver, 3).until(EC.visibility_of_element_loca
ted((By.XPATH, '//*[@class="css-me46ht"]')))

kategori_element = driver.find_elements_by_xpath('//*[@class
="css-me46ht"]')
for kategori in kategori_element:
    if 'Kesehatan' in kategori.text:
        ActionChains(driver).move_to_element(kategori).perfo
rm()

subkategori_element = driver.find_elements_by_xpath('//*[@cl
ass="css-1qaqbbz"]')
for subkategori in subkategori_element:
    list_kategori.append(subkategori.text)
    list_url.append(subkategori.get_attribute('href'))

subkategori_element = driver.find_elements_by_xpath('//*[@cl
ass="css-1nykm5o"]')
for subkategori in subkategori_element:
    list_kategori.append(subkategori.text)
    list_url.append(subkategori.get_attribute('href'))

driver.quit()

list_kategori = [each_string.lower() for each_string in list
_kategori]

df['kategori'] = list_kategori
df['link'] = list_url

#save to csv
df.to_csv('C:/Users/ALDYSYAH/OneDrive - Institut Teknologi S
epuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/kategori.csv')
```

Tabel A.2 Kode Program *Scraper* untuk URL Produk Tokopedia

Scraper URL Produk Tokopedia	
Fungsi	Melakukan *web-sraping* untuk memperoleh URL Produk di Tokopedia
Bahasa	Python

```
#Importing packages
from selenium import webdriver
import pandas as pd
from selenium.webdriver.firefox.options import Options
from datetime import datetime

startTime = datetime.now()

produk = pd.DataFrame(columns = ['kategori','title','link'])

df_kategori = pd.read_csv('C:/Users/ALDYSYAH/OneDrive - Inst
itut Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/kategori.csv',index_col=0)
list_kategori = df_kategori['kategori'].tolist()
list_url_kategori = df_kategori['link'].tolist()

#array for save kategori, title, and url product
list_kategori_produk = []
list_titles_produk = []
list_url_produk = []

#driver Firefox headless
options = Options()
options.headless = True
# fp = webdriver.FirefoxProfile('F:/Temp')
driver = webdriver.Firefox(options=options, executable_path=
r'C:/Users/ALDYSYAH/OneDrive - Institut Teknologi Sepuluh No
pember/Semester 8/TA/Scraping/TA-Scrapping/geckodriver.exe')


#loop for all page in category
index = 0
for url in list_url_kategori:

    driver.get(str(url) + '?page=1&fcity=252')
    print(str(str(url) + '?page=1&fcity=252'))
    while True:
        driver.implicitly_wait(3)
        next_page = driver.find_elements_by_xpath('//i[@clas
s="css-98hn3t e19tp72t1"]')

        if(next_page):
            url_element = driver.find_elements_by_xpath('//*
[@class="css-bk6tzz e1nlzfl3"]/a')
```

Scraper URL Produk Tokopedia

```
            title_element = driver.find_elements_by_xpath('/
/*[@class="css-bk6tzz e1nlzfl3"]//*[@class="css-1bjwylw"]')
            for title in title_element:
                list_titles_produk.append(title.text)
            for url in url_element:
                list_url_produk.append(url.get_attribute('hr
ef'))
                list_kategori_produk.append(list_kategori[in
dex])
            try:
                next_cant_click_element = driver.find_elemen
t_by_xpath('//*[@Class="css-13vsjvx e19tp72t1"]')
                break
            except:
                pass
            next_page[0].click()
        else:
            break
    index+=1

driver.quit()

#save array to dataframe
produk['kategori'] = list_kategori_produk
produk['title'] = list_titles_produk
produk['link'] = list_url_produk

#save to csv
produk.to_csv('C:/Users/ALDYSYAH/OneDrive - Institut Teknolo
gi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/url_produk_tokped_kesehatan.csv')

exec_time = datetime.now() - startTime
print(exec_time)
```

Tabel A.3 Kode Program *Scraper* untuk URL Toko Tokopedia

Scraper URL Toko pada Tokopedia	
Fungsi	Melakukan *pre-processing* URL Produk menjadi URL toko
Bahasa	Python

```
import pandas as pd
# from urllib.parse import urlparse

df_url_produk = pd.read_csv('C:/Users/ALDYSYAH/OneDrive - In
stitut Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/Aldy/url_produk_tokped_kesehatan.csv',in
dex_col=0)
list_url_produk = df_url_produk['link'].tolist()

df_url_produk_promo = df_url_produk[df_url_produk['link'].st
r.contains("ta.tokopedia.com/promo")]
list_url_produk_promo = df_url_produk_promo['link'].tolist()

#create dataframe for notpromo product link
df_url_produk_notpromo = df_url_produk[~df_url_produk.link.s
tr.contains("ta.tokopedia.com/promo")]

df_url_toko_notpromo = pd.DataFrame(columns=['link'])
#get toko link
df_url_toko_notpromo["link"] = df_url_produk_notpromo["link"
].str.rsplit("/", 1).str[-2]

#preprocess promo link
list_toko_promo = []
for url in list_url_produk_promo:
    url = url[url.find('https%3A%2F%2F'):url.find('%3Fsrc%3D
topads')]
    url = url.replace('%3A%2F','')
    url = url.replace('https%2F','https://')
    url = url.replace('%2F','/')
    #get url toko promo
    list_toko_promo.append(url.rsplit("/",1)[-2])

#create dataframe for unique promo toko link
list_toko_promo_unique = list(set(list_toko_promo))
df_url_toko_promo_unique = pd.DataFrame({'link':list_toko_pr
omo_unique})
df_url_toko_promo_unique.to_csv('C:/Users/ALDYSYAH/OneDrive
- Institut Teknologi Sepuluh Nopember/Semester 8/TA/Scraping
/TA-
Scrapping/Tokopedia/Aldy/url_toko_promo_tokopedia_kesehatan_
unique.csv')
print('jumlah toko makanan minuman di surabaya yang telah me
nggunakan fitur promo pada platform tokopedia: ' + str(len(l
ist_toko_promo_unique)))
```

Scraper URL Toko pada Tokopedia

```
#append notpromo toko to promo toko dataframe
df_url_toko_all = df_url_toko_promo_unique.append(df_url_tok
o_notpromo)

#get unique all toko link
df_url_toko_all_unique = df_url_toko_all.drop_duplicates('li
nk')
print('jumlah toko makanan minuman di surabaya pada platform
 tokopedia: ' + str(len(df_url_toko_all_unique)))

#save all unique toko link to csv
df_url_toko_all_unique.to_csv('C:/Users/ALDYSYAH/OneDrive -
Institut Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/T
A-
Scrapping/Tokopedia/Aldy/url_toko_all_tokopedia_kesehatan_un
ique.csv,index=False)
```

Tabel A.4 Kode Program *Scraper* untuk Informasi Toko Tokopedia

Scraper Informasi Toko pada Tokopedia	
Fungsi	Melakukan *web-sraping* pada Tokopedia untuk mendapatkan informasi toko
Bahasa	Python

```
#Importing packages
from selenium import webdriver
import pandas as pd
import time
# from selenium.webdriver.firefox.options import Options
from datetime import datetime

startTime = datetime.now()

#read file
df_url_toko = pd.read_csv('C:/Users/ALDYSYAH/OneDrive - Inst
itut Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/Aldy/url_toko_all_tokopedia_kesehatan_un
ique.csv')

#hapus row berdasarkan index yang dirasa perlu dihapus
#df_url_toko = df_url_toko.drop(df_url_toko.index[0:972])

#make dataframe to list
list_url_toko = df_url_toko['link'].tolist()

#make dataframe to save informasi toko
toko = pd.DataFrame(columns = ['Nama','Kota','Pemilik','Desk
ripsi','Alamat','Since','Follower','Terjual','Rating','Ulasa
n'])
```

Scraper Informasi Toko pada Tokopedia

```
# #ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
# chrome_options.add_argument("--headless")
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/ALDYSYAH/OneDrive - Institut Teknologi Sepul
uh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()

#loop for every 'toko' link
for url in list_url_toko:
    try:
        #get url
        print(url)
        driver.get(url)
        driver.implicitly_wait(2)

        #get nama toko
        name_element = driver.find_elements_by_xpath('//*[@c
lass="css-1yw0bav"]')
        name = name_element[0].text

        #get kota
        kota_element = driver.find_elements_by_xpath('//*[@c
lass="css-1k56vr7"]/li[2]/p')
        kota = kota_element[0].text

        #get jumlah follower
        try:
            follower_element = driver.find_elements_by_xpath
('//*[@class="css-jsut4p-unf-heading e1qvo2ff6"]')
            follower = follower_element[0].text
        except:
            follower = None

        #get jumlah terjual
        try:
            sold_element = driver.find_elements_by_xpath('//
*[@class="css-lzwncz-unf-heading e1qvo2ff2"]')
            sold = sold_element[0].text
        except:
            sold = None

        #get rating
        try:
            rating_element = driver.find_elements_by_xpath('
//*[@class="css-rfs3ih-unf-heading e1qvo2ff2"]')
            rating = rating_element[0].text
```

Scraper Informasi Toko pada Tokopedia

```
        except:
            rating = None

        #get review
        try:
            review_element = driver.find_elements_by_xpath('
//*[@class="css-1s96mum-unf-heading e1qvo2ff6"]')
            review = review_element[0].text
        except:
            review = None

        info_buttons = driver.find_elements_by_xpath('//*[@c
lass="css-rhf1fq-unf-btn e1ggruw00"]')
        driver.execute_script("arguments[0].click();", info_
buttons[0])
        time.sleep(2)

        owner_element = driver.find_elements_by_xpath('//*[@
class="css-1ag3tdd-unf-heading e1qvo2ff4"]')
        pemilik = owner_element[0].text

        try:
            #get deskripsi dan alamat
            description_element = driver.find_elements_by_xp
ath('//*[@class="css-f4j8r2-unf-heading e1qvo2ff8"]')
            deskripsi = description_element[0].text
        except:
            deskripsi = None

        alamat = description_element[1].text

        #get buka sejak
        since_element = driver.find_elements_by_xpath('//*[@
class="css-1hjz1j2-unf-heading e1qvo2ff8"]')
        since = since_element[0].text
        df= pd.DataFrame({"Nama":[name],"Kota":[kota],
                          "Pemilik":[pemilik], "Deskripsi":[d
eskripsi], "Alamat":[alamat], "Since":[since],
"Follower":[follower], "Sold":[sold],
"Rating":[rating], "Review":[review]})
        toko = pd.concat([toko,df], sort=False)
        #save to csv

    except:
        continue
toko.to_csv('C:/Users/ALDYSYAH/OneDrive - Institut Teknologi
 Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/Aldy/info_toko_tokopedia_kesehatan.csv')
```

Scraper Informasi Toko pada Tokopedia

```
driver.quit()

exec_time = datetime.now() - startTime
print(exec_time)
```

2. Shopee

Tabel A.5 Kode Program *Scraper* untuk Sub Kategori Shopee

Scraper Sub Kategori Shopee	
Fungsi	Melakukan *web-sraping* pada Shopee untuk mendapatkan list kategori
Bahasa	Python

```
import pandas as pd
from selenium import webdriver
import time

df = pd.DataFrame(columns = ['kategori','link'])

list_kategori = []
list_url = []

# ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/user01/OneDrive - Institut Teknologi Sepuluh
 Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()

driver.get('https://shopee.co.id/Kesehatan-cat.14780/')
driver.implicitly_wait(2)
arrow_down_button = driver.find_elements_by_xpath('//*[@clas
s="stardust-icon stardust-icon-arrow-down"]')
#arrow_down_button[0].click()
time.sleep(2)

main_kategori_element = driver.find_elements_by_xpath('//*[@
class="shopee-category-list__main-category__link"]')
list_kategori.append(main_kategori_element[0].text)
list_url.append(main_kategori_element[0].get_attribute('href
'))

kategori_element = driver.find_elements_by_xpath('//*[@class
="shopee-category-list__sub-category"]')
for kategori in kategori_element:
    list_kategori.append(kategori.text)
    list_url.append(kategori.get_attribute('href'))
```

Scraper Sub Kategori Shopee

```
driver.quit()

list_kategori = [each_string.lower() for each_string in list
_kategori]

df['kategori'] = list_kategori
df['link'] = list_url

#save to csv
df.to_csv('C:/Users/user01/OneDrive - Institut Teknologi Sep
uluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/kategori_shopee.csv')
```

Tabel A.6 Kode Program *Scraper* untuk URL Produk Shopee

Scraper URL Produk pada Shopee	
Fungsi	Melakukan *web-sraping* pada Shopee untuk mendapatkan URL produk Shopee
Bahasa	Python

```
#Importing packages
from selenium import webdriver
import pandas as pd
from selenium.webdriver.firefox.options import Options
from datetime import datetime

startTime = datetime.now()

produk = pd.DataFrame(columns = ['kategori','title','link'])

df_kategori = pd.read_csv('C:/Users/user01/OneDrive - Instit
ut Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/kategori_shopee.csv',index_col=0)
list_kategori = df_kategori['kategori'].tolist()
list_url_kategori = df_kategori['link'].tolist()

#array for save kategori, title, and url product
list_kategori_produk = []
list_titles_produk = []
list_url_produk = []

# #ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/user01/OneDrive - Institut Teknologi Sepuluh
 Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()
```

Scraper URL Produk pada Shopee

```
#loop for all page in category
index = 0
for url in list_url_kategori:
    driver.get(str(url) + '?locations=Jawa%2520Timur')
    print(str(url) + '?locations=Jawa%2520Timur')
    while True:
        driver.implicitly_wait(4)
        next_page = driver.find_elements_by_xpath('//button[
@class="shopee-button-outline shopee-mini-page-
controller__next-btn"]')

        if (next_page):
            url_element = driver.find_elements_by_xpath('//*
[@class="col-xs-2-4 shopee-search-item-
result__item"]/div/a')
            title_element = driver.find_elements_by_xpath('/
/*[@class="O6wiAW"]')
            #get text judul produk
            for title in title_element:
                list_titles_produk.append(title.text)
                print(list_titles_produk)
            #get text url
            for url in url_element:
                list_url_produk.append(url.get_attribute('hr
ef'))
                list_kategori_produk.append(list_kategori[in
dex])

            next_page[0].click()
        else:
            break
    index+=1

driver.quit()

#save array to dataframe
produk['kategori'] = list_kategori_produk
produk['title'] = list_titles_produk
produk['link'] = list_url_produk

#save to csv
produk.to_csv('C:/Users/user01/OneDrive - Institut Teknologi
 Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/url_produk_shopee_kesehatan.csv')

exec_time = datetime.now() - startTime
print(exec_time)
```

Tabel A.7 Kode Program *Scraper* untuk URL Toko Shopee

Scraper URL Toko pada Shopee	
Fungsi	Melakukan *web-sraping* pada Shopee untuk mendapatkan URL toko Shopee
Bahasa	Python

```
#Importing packages
from selenium import webdriver
import pandas as pd
# from selenium.webdriver.firefox.options import Options
from datetime import datetime

startTime = datetime.now()

#read file
df_produk = pd.read_csv('C:/Users/user01/OneDrive - Institut
 Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/url_produk_shopee_kesehatan.csv',index
_col=0)

#make dataframe to list
list_url_produk = df_produk['link'].tolist()

list_url_toko = pd.DataFrame(columns=['link'])

# #ChromeDriver
chrome_options = webdriver.ChromeOptions()
#chrome_options.add_argument("--incognito")
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/user01/OneDrive - Institut Teknologi Sepuluh
 Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()

for url in list_url_produk:
    try:
        print(url)
        driver.get(url)
        driver.implicitly_wait(3)
        #get url toko
        url_element = driver.find_elements_by_xpath('//*[@cl
ass="_1jOO4S"]/a')
        df = pd.DataFrame({"link":[url_element[0].get_attrib
ute('href')]})
        list_url_toko=pd.concat([list_url_toko,df], sort=Fal
se)
    except:
        continue

driver.quit()
```

Scraper URL Toko pada Shopee

```
print('saving...')
#save array to dataframe
#df_toko = pd.DataFrame({'link':list_url_toko})
list_url_toko.to_csv('C:/Users/user01/OneDrive - Institut Te
knologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/url_toko_shopee_kesehatan.csv',index=F
alse)
exec_time = datetime.now() - startTime
print(exec_time)
```

Tabel A.8 Kode Program *Scraper* untuk Informasi Toko Shopee

Scraper Informasi Toko pada Shopee	
Fungsi	Melakukan *web-sraping* pada Shopee untuk mendapatkan informasi toko Shopee
Bahasa	Python

```
#Importing packages
from selenium import webdriver
import pandas as pd
# from selenium.webdriver.firefox.options import Options
from datetime import datetime
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions a
s EC

startTime = datetime.now()

#read file
df_url_toko = pd.read_csv('C:/Users/user01/OneDrive - Instit
ut Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/url_toko_shopee_kesehatan2.csv')

#make dataframe to list
list_url_toko = df_url_toko['link'].tolist()

#ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
prefs = {"profile.managed_default_content_settings.images":
2}
chrome_options.add_experimental_option("prefs", prefs)
# chrome_options.add_argument("--headless")
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/user01/OneDrive - Institut Teknologi Sepuluh
 Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()

#make dataframe to save informasi toko
toko = pd.DataFrame(columns = ['name','link','city','seller_
type','product_category','numOf_product','chat_performance',
'followers','rating_review','since','voucher','numOf_media',
'description'])

# toko = toko.drop(toko.index[1378:1394])

#loop for every 'toko' link
index=0

for url in list_url_toko:
```

Scraper Informasi Toko pada Shopee

```
    try:
        #get url
        driver.get(url)
        # WebDriverWait(driver, 5).until(EC.presence_of_elem
ent_located((By.XPATH, '//*[@id="section-seller-overview-
horizontal__portrait-name"]')))
        driver.implicitly_wait(5)
        print(str(index) + ' : ' + url)
        try:
            #get nama toko
            name_element = driver.find_elements_by_xpath('//
*[@class="section-seller-overview-horizontal__portrait-
name"]')
            name = name_element[0].text
        except:
            name = None

        link = url

        driver.implicitly_wait(1)
        try:
            #get kota
            city_element = driver.find_element_by_xpath('//*
[@class="_3amru2"]')
            city = city_element.text
        except:
            city = None
        try:
            #get seller type
            seller_type_element = driver.find_element_by_xpa
th('//*[@class="ofs-header__page-name"]')
            seller_type = 'Shopee Mall'
        except:
            try:
                seller_type_element = driver.find_element_by
_xpath('//*[@class="horizontal-badge shopee-preferred-
seller-badge horizontal-badge--no-triangle"]')
                seller_type = 'Star Seller'
            except:
                seller_type = None
        try:
            product_category_element = driver.find_elements_
by_xpath('//*[@class="shopee-category-list__body"]')
            product_category = product_category_element[0].t
ext
        except:
            product_category = None
        try:
           #get jumlah produk yang dijual
```

Scraper Informasi Toko pada Shopee

```
            numOf_product_element = driver.find_elements_by_x
path('//div[@class="section-seller-overview__item-
text" and contains(.,"produk")]/div[2]')
            numOf_product = numOf_product_element[0].text
        except:
            numOf_product = None
        try:
            #get performa chat
            chat_performance_element = driver.find_elements_
by_xpath('//div[@class="section-seller-overview__item-
text" and contains(.,"Performa chat")]/div[2]')
            chat_performance = chat_performance_element[0].t
ext
        except:
            chat_performance = None
        try:
            #get jumlah follower
            followers_element = driver.find_elements_by_xpat
h('//div[@class="section-seller-overview__item-
text" and contains(.,"pengikut")]/div[2]')
            followers = followers_element[0].text
        except:
            followers = None
        try:
            #get penilaian
            rating_review_element = driver.find_elements_by_
xpath('//div[@class="section-seller-overview__item-
text" and contains(.,"penilaian")]/div[2]')
            rating_review = rating_review_element[0].text
        except:
            rating_review = None
        try:
            #get bergabung
            since_element = driver.find_elements_by_xpath('/
/div[@class="section-seller-overview__item-
text" and contains(.,"bergabung")]/div[2]')
            since = since_element[0].text
        except:
            since = None
        try:
            #get voucher
            voucher_element = driver.find_elements_by_xpath(
'//*[@class="shop-page__section shop-page__vouchers"]')
            if voucher_element:
                voucher = True
            else:
                voucher = False
        except:
            voucher = False
```

Scraper Informasi Toko pada Shopee

```
        try:
            #get jumlah media
            numOf_media_element = driver.find_elements_by_xp
ath('//*[@class="image-carousel__dots"]/div')
            numOf_media = len(numOf_media_element)
        except:
            numOf_media = None
        try:
            #get tentang toko
            description_element = driver.find_elements_by_xp
ath('//div[contains(@class,"shop-page-shop-description")]')
            description = description_element[0].text
        except:
            description = None
        toko.loc[len(toko)] = [name,link,city,seller_type,pr
oduct_category, numOf_product,chat_performance,followers,
rating_review,since,voucher,numOf_media, description]
        index+=1
    except:
        name=city=seller_type=product_category=numOf_product
=chat_performance=followers=rating_review=since=voucher=numO
f_media=description=None
        toko.loc[len(toko)] = [name,link,city,seller_type,pr
oduct_category,numOf_product,chat_performance,followers,
rating_review,since,voucher,numOf_media, description]
        index+=1

driver.quit()

#save to csv
toko.to_csv('C:/Users/user01/OneDrive - Institut Teknologi S
epuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/info_toko_shopee_kesehatan2.csv',index
=False)

exec_time = datetime.now() - startTime
print(exec_time)
```

3. Instagram

Tabel A.9 Kode Program *Scraper* Instagram berdasarkan Tokopedia

Scraper Akun Instagram Berdasarkan Tokopedia	
Fungsi	Melakukan *web-sraping* pada Instagram dengan kata kunci nama toko pada Tokopedia

Scraper Akun Instagram Berdasarkan Tokopedia

Bahasa | Python

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import pandas as pd
import time

#read file
df_toko = pd.read_csv('C:/Users/user01/OneDrive - Institut T
eknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/Aldy/info_toko_tokopedia_kesehatan1.csv'
,index_col=0)
df_toko['name'] = df_toko['name'].str.encode('ascii', 'ignor
e').str.decode('ascii')

list_store = df_toko['name'].tolist()
list_city = df_toko['city'].tolist()

# #ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
prefs = {"profile.managed_default_content_settings.images":
2}
chrome_options.add_experimental_option("prefs", prefs)
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/user01/OneDrive - Institut Teknologi Sepuluh
 Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()

driver.implicitly_wait(5)
browser = driver.get('https://www.instagram.com/instagram/')

try:
    notnow_element = driver.find_element_by_xpath('//*[@clas
s="aOOlW   HoLwm "]')
    notnow_element.click()
except:
    pass

toko = pd.DataFrame(columns = ['store','city','account','des
cription','link'])

index=0
search_element = driver.find_element_by_xpath('//*[@class="X
TCLo x3qfX "]')
for store in list_store:
    #SEARCHs
    search_element.clear()
    account=description=link=None
```

Scraper Akun Instagram Berdasarkan Tokopedia

```
    search_element.send_keys(str(store))
    print('searching: ' + str(store))
    driver.implicitly_wait(0)
    time.sleep(1)
    result_element = driver.find_elements_by_xpath('//a[@cla
ss="yCE8d  "]')
    print('found result: ' + str(len(result_element)))
    if result_element:
        for result in result_element:
            try:
                account_element = result.find_elements_by_xp
ath('.//*[@class="Ap253"]')
                account = account_element[0].text
            except:
                account = None
            try:
                description_element = result.find_elements_b
y_xpath('.//*[@class="Fy4o8"]')
                description = description_element[0].text
            except:
                description = None
            try:
                link = result.get_attribute('href')
            except:
                link = None
            toko.loc[len(toko)] = [store,list_city[index],ac
count,description,link]
            toko.to_csv('C:/Users/user01/OneDrive - Institut
 Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/Aldy/instagram_info_toko_tokopedia_keseh
atan1.csv')
    else:
        account=description=link=None
        toko.loc[len(toko)] = [store,list_city[index],accoun
t,description,link]
        toko.to_csv('C:/Users/user01/OneDrive - Institut Tek
nologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Tokopedia/Aldy/instagram_info_toko_tokopedia_keseh
atan1.csv')

    index+=1
```

Tabel A.10 Kode Program Scraper Instagram berdasarkan Shopee

Scraper Akun Instagram Berdasarkan Shopee	
Fungsi	Melakukan *web-sraping* pada Instagram dengan kata kunci nama toko pada Shopee
Bahasa	Python

```
from selenium import webdriver
```

Scraper Akun Instagram Berdasarkan Shopee

```
from selenium.webdriver.common.keys import Keys
import pandas as pd
import time
#from pyvirtualdisplay import Display

#read file
df_toko = pd.read_csv('C:/Users/user01/OneDrive - Institut T
eknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/info_toko_shopee_kesehatan1.csv')
df_toko['name'] = df_toko['name'].str.encode('ascii', 'ignor
e').str.decode('ascii')

list_store = df_toko['name'].tolist()
list_city = df_toko['city'].tolist()

# #ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
prefs = {"profile.managed_default_content_settings.images":
2}
chrome_options.add_experimental_option("prefs", prefs)
# chrome_options.add_argument("--headless")
driver = webdriver.Chrome(options=chrome_options, executable
_path='C:/Users/user01/OneDrive - Institut Teknologi Sepuluh
 Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Chromedriver/chromedriver.exe')
driver.maximize_window()

# #open driver
# options = Options()
# options.headless = True
# driver = webdriver.Firefox(options=options, executable_pat
h=r'C:\web driver\geckodriver.exe')

driver.implicitly_wait(5)
browser = driver.get('https://www.instagram.com/instagram/')

try:
    notnow_element = driver.find_element_by_xpath('//*[@clas
s="aOOlW   HoLwm "]')
    notnow_element.click()
except:
    pass
toko = pd.DataFrame(columns = ['store','city','account','des
cription','link'])

index=0
search_element = driver.find_element_by_xpath('//*[@class="X
TCLo x3qfX "]')
```

Scraper Akun Instagram Berdasarkan Shopee

```
for store in list_store:
    #SEARCHs
    search_element.clear()
    account=description=link=None
    search_element.send_keys(str(store))
    print('searching: ' + str(store))
    driver.implicitly_wait(0)
    time.sleep(1)
    result_element = driver.find_elements_by_xpath('//a[@cla
ss="yCE8d  "]')
    print('found result: ' + str(len(result_element)))
    if result_element:
        for result in result_element:
            try:
                account_element = result.find_elements_by_xp
ath('.//*[@class="Ap253"]')
                account = account_element[0].text
            except:
                account = None
            try:
                description_element = result.find_elements_b
y_xpath('.//*[@class="Fy4o8"]')
                description = description_element[0].text
            except:
                description = None
            try:
                link = result.get_attribute('href')
            except:
                link = None
            toko.loc[len(toko)] = [store,list_city[index],ac
count,description,link]
            toko.to_csv('C:/Users/user01/OneDrive - Institut
 Teknologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/instagram_info_toko_shopee_kesehatan1.
csv')
    else:
        account=description=link=None
        toko.loc[len(toko)] = [store,list_city[index],accoun
t,description,link]
        toko.to_csv('C:/Users/user01/OneDrive - Institut Tek
nologi Sepuluh Nopember/Semester 8/TA/Scraping/TA-
Scrapping/Shopee/Aldy/instagram_info_toko_shopee_kesehatan1.
csv')

    index+=1
```

4. Google My Business

Tabel A.11 Kode Program Scraper untuk Membuat Kata Kunci

Scraper untuk Membuat Kata Kunci Google My Business	
Fungsi	Membuat kata kunci untuk Google My Business
Bahasa	Python

```
import pandas as pd
from datetime import datetime

startTime = datetime.now()

df_kategori = pd.read_csv('D:/Documents/Kuliah/TA/scraper/fi
le/kategori.csv',index_col=0)
list_kategori = df_kategori['kategori'].tolist()

df_kecamatan = pd.read_csv('D:/Documents/Kuliah/TA/scraper/f
ile/kecamatan.csv',index_col=0)
list_kecamatan = df_kecamatan['kecamatan'].tolist()

df_keyword = pd.DataFrame(columns = ['keyword'])

for kategori in list_kategori:
    list_keyword = []
    for kecamatan in list_kecamatan:
        list_keyword.append('toko ' + str(kategori) + ' keca
matan ' + str(kecamatan))
    df_keyword['keyword'] = list_keyword
    df_keyword.to_csv('D:/Documents/Kuliah/TA/scraper/Google
 Business/keyword/keyword_'+str(kategori)+'.csv')

exec_time = datetime.now() - startTime
print(exec_time)
```

Tabel A.12 Kode Program Scraper untuk Instagram

Scraper Informasi Toko pada Google My Business	
Fungsi	Melakukan *web-sraping* pada Google My Business untuk mendapatkan informasi toko
Bahasa	Python

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import pandas as pd
from datetime import datetime
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions a
s EC
```

Scraper Informasi Toko pada Google My Business

```
import os
from pyvirtualdisplay import Display

startTime = datetime.now()
print('start time: ' + str(startTime))

df_kategori = pd.read_csv('/root/makananminuman/file/kategor
i.csv',index_col=0)
list_kategori = df_kategori['kategori'].tolist()

df_kecamatan = pd.read_csv('/root/makananminuman/file/kecama
tan.csv',index_col=0)
list_kecamatan = df_kecamatan['kecamatan'].tolist()

toko = pd.DataFrame(columns = ['Nama Toko','Website','Rating
','Jumlah Google Review','Alamat','Jam Operasional','Phone']
)

list_keyword = []
for kategori in list_kategori:
    for kecamatan in list_kecamatan:
        list_keyword.append('toko ' + str(kategori) + ' keca
matan ' + str(kecamatan))

#start display
display = Display(visible=0, size=(1366, 768))
display.start()

# #ChromeDriver
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--incognito")
chrome_options.add_argument('--no-sandbox')
driver = webdriver.Chrome(options=chrome_options)
driver.maximize_window()

# driver.maximize_window()
driver.get('https://www.google.com/')
print('opening google.com')
WebDriverWait(driver, 3).until(EC.visibility_of_element_loca
ted((By.XPATH, '//div[@id="SIvCob"]/a')))

WebDriverWait(driver, 2).until(EC.visibility_of_element_loca
ted((By.XPATH, '//*[@class="gLFyf gsfi"]')))
search_element = driver.find_element_by_xpath('//*[@class="g
LFyf gsfi"]')

for keyword in list_keyword:
    #search
    search_element.send_keys(str(keyword))
```

Scraper Informasi Toko pada Google My Business

```
    search_element.send_keys(Keys.RETURN)
    print('searching: ' + keyword)

    driver.implicitly_wait(3)

    #click more places
    try:
        morePlaces_element = driver.find_elements_by_xpath('
//*[@class="zkIadb"]')
        morePlaces_element[0].click()
        WebDriverWait(driver, 3).until(EC.visibility_of_elem
ent_located((By.XPATH, '//*[@class="VkpGBb"]/a')))
        # time.sleep(2)
    except:
        pass

    print('getting data')
    next_page = True
    page=1
    while next_page:
        print(keyword)
        print('page: '+str(page))
        #click every toko
        # WebDriverWait(driver, 3).until(EC.visibility_of_el
ement_located((By.XPATH, '//*[@class="VkpGBb"]/a')))
        container_element = driver.find_elements_by_xpath('/
/*[@class="VkpGBb"]/a[1]')

        store=1
        for container in container_element:
            driver.execute_script("arguments[0].click();", c
ontainer)
            print('click toko '+str(store))
            nama = None
            website = None
            rating = None
            jumlahGoogleReview = None
            alamat = None
            hours = None
            phone = None
            linkTokopedia = None
            linkShopee = None
            instagram = None


            time.sleep(2)

            #get nama toko
            try:
```

Scraper Informasi Toko pada Google My Business

```
                nama_element = driver.find_elements_by_xpath
('//*[@class="SPZz6b"]/div/span')
                nama = nama_element[0].text
            except:
                nama = None

            #get website
            try:
                websiteButton_element = driver.find_elements
_by_xpath('//*[@class="QqG1Sd"]/a')
                websiteButton = websiteButton_element[0].tex
t
                if 'Website' in websiteButton:
                    website = websiteButton_element[0].get_a
ttribute('href')
            except:
                website = None

            #get rating
            try:
                rating_element = driver.find_elements_by_xpa
th('//*[@class="Ob2kfd"]/div/span[1]')
                rating = rating_element[0].text
            except:
                rating = None

            #get jumlah Google review
            try:
                jumlahGoogleReview_element = driver.find_ele
ments_by_xpath('//*[@class="hqzQac"]/span/a/span')
                jumlahGoogleReview = jumlahGoogleReview_elem
ent[0].text
            except:
                jumlahGoogleReview = None

            #get alamat
            try:
                alamat_element = driver.find_elements_by_xpa
th('//*[@class="LrzXr"]')
                alamat = alamat_element[0].text
            except:
                alamat = None

            #get jam buka-tutup
            try:
                info_element = driver.find_elements_by_xpath
('//div[@class="zloOqf PZPZlf"]')
                divClick_element = info_element[1].find_elem
ents_by_xpath('./div/div/div[1]/div[1]')
```

Scraper Informasi Toko pada Google My Business

```
                driver.execute_script("arguments[0].click();
", divClick_element[0])
                time.sleep(2)

                hours_element = info_element[1].find_element
s_by_xpath('./div/div/div[1]/div[2]/table/tbody')
                hours = hours_element[0].text
            except:
                hours = None

            #get phone
            try:
                phone_element = driver.find_elements_by_xpat
h('//*[@class="LrzXr zdqRlf kno-fv"]')
                phone = phone_element[0].text
            except:
                phone = None

            toko.loc[len(toko)] = [nama,website,rating,jumla
hGoogleReview,alamat,hours,phone]
            toko.to_csv('/root/makananminuman/googleBusiness
/googleBusiness_makananMinuman_Surabaya.csv')
            store+=1

        clear = lambda: os.system('clear') #on Linux System
        clear()

        next_element = driver.find_elements_by_id('pnnext')
        if (next_element):
            next_element[0].click()
            page+=1
            time.sleep(3)
        else:
            next_page = False

    search_element = driver.find_element_by_xpath('//*[@clas
s="gsfi"]')
    search_element.clear()

driver.quit()

# toko.to_csv('D:/Documents/Kuliah/TA/scraper/Google Busines
s/googleBusiness_kesehatan_Surabaya.csv')

exec_time = datetime.now() - startTime
print('end time: ' + str(datetime.now()))
print('running time: ' + exec_time)
```

b. Kode Pra-Proses Data

Pada bagian ini dijelaskan kode yang digunakan dalam pra-proses data.

Tabel A.13 Kode *pre-process* data Tokopedia

Pra-proses data Tokopedia	
Fungsi	Melakukan pembersihan, transformasi pada data Tokopedia
Bahasa	Python

```
import pandas as pd
import numpy as np
import re
# from datetime import datetime

####### LOAD DATA #########
df_toko_1 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\info_toko_tokopedia_kesehatan1.csv',index_col=0)
df_toko_2 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\info_toko_tokopedia_kesehatan2.csv',index_col=0)
df_toko_3 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\info_toko_tokopedia_kesehatan3.csv',index_col=0)
#df_toko_5 = pd.read_csv('D:\\Documents\\Kuliah\\TA\\scraper
\\Google Business\\googleBusiness_tokped_5.csv',index_col=0)

df_toko = df_toko_1.append(df_toko_2)
df_toko = df_toko.append(df_toko_3)

df_kelurahan = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\scraper\file
\kelurahan.csv',index_col=0)
#lowercase kecamatan dan kelurahan
df_kelurahan['kecamatan'] = df_kelurahan['kecamatan'].str.lo
wer()
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.lo
wer()
#replace word 'kecamatan'
df_kelurahan['kecamatan'] = df_kelurahan.kecamatan.str.repla
ce('kecamatan ','')
df_kelurahan['kecamatan'] = df_kelurahan.kecamatan.str.repla
ce('kecamatan','')
#replace word 'kelurahan'
df_kelurahan['kelurahan'] = df_kelurahan.kelurahan.str.repla
ce('kelurahan ','')
df_kelurahan['kelurahan'] = df_kelurahan.kelurahan.str.repla
ce('kelurahan','')
```

```
list_kecamatan = df_kelurahan['kecamatan'].tolist()
list_kelurahan = df_kelurahan['kelurahan'].tolist()

###### DROP DUPLICATE #######
df_toko = df_toko.drop_duplicates('link')

####### PREPROCESS SELLER_TYPE #######print('PREPROCESS SELL
ER_TYPE')
df_toko.seller_type.replace(np.nan, 'Regular Seller', regex=
True, inplace=True)

####### PREPROCESS CITY #######
print('PREPROCESS CITY')

#Drop NaN city
df_toko = df_toko[df_toko['city'].notna()]
#Lowercase city
df_toko['city'] = df_toko['city'].str.lower()
#Load file kelurahan

list_filter_city = ['surabaya','sby']
for kecamatan in list_kecamatan:
    list_filter_city.append(kecamatan)

df_toko['city'] = df_toko['city'].str.replace('gununganyar',
 'gunung anyar')
df_toko['city'] = df_toko['city'].str.replace('pabean cantik
an', 'pabean cantian')

#drop if city not surabaya
df_toko_notSurabaya = df_toko[~df_toko.city.str.contains('|'
.join(list_filter_city))]
df_toko = df_toko[df_toko.city.str.contains('|'.join(list_fi
lter_city))]
df_toko = df_toko.reset_index(drop=True)

####### PREPROCESS FOLLOWERS #######
print('PREPROCESS FOLLOWERS')

#replace nan with 0
df_toko.followers.replace(np.nan, '0', regex=True, inplace=T
rue)
#lowercase follower/s
df_toko['followers'] = df_toko['followers'].str.lower()
df_toko['followers'] = df_toko['followers'].str.replace('fol
lowers', '')
df_toko['followers'] = df_toko['followers'].str.replace('fol
lower', '')

for row in range(0,len(df_toko.index)):
```

```
    if 'k' in df_toko.loc[row,'followers']:
        df_toko.loc[row,'followers'] = df_toko.loc[row,'foll
owers'].replace('k', '')
        df_toko.loc[row,'followers'] = int(float(df_toko.loc
[row,'followers'])*1000)
    elif 'm' in df_toko.loc[row,'followers']:
        df_toko.loc[row,'followers'] = df_toko.loc[row,'foll
owers'].replace('m', '')
        df_toko.loc[row,'followers'] = int(float(df_toko.loc
[row,'followers'])*1000000)

df_toko['followers'] = df_toko['followers'].astype(int)

####### PREPROCESS REPUTATION_POINTS #######
print('PREPROCESS REPUTATION_POINTS')

#replace nan with 0
df_toko.reputation_point.replace(np.nan, '0', regex=True, in
place=True)
#lowercase follower/s
df_toko['reputation_point'] = df_toko['reputation_point'].st
r.lower()
df_toko['reputation_point'] = df_toko['reputation_point'].st
r.replace('points', '')
df_toko['reputation_point'] = df_toko['reputation_point'].st
r.replace('point', '')
df_toko['reputation_point'] = df_toko['reputation_point'].st
r.replace('.', '')

df_toko['reputation_point'] = df_toko['reputation_point'].as
type(int)

####### PREPROCESS PRODUCT_SOLD #######print('PREPROCESS PRO
DUCT_SOLD')

#replace nan with 0
df_toko.product_sold.replace(np.nan, '0', regex=True, inplac
e=True)
#lowercase follower/s
df_toko['product_sold'] = df_toko['product_sold'].str.lower(
)

for row in range(0,len(df_toko.index)):
    if 'k' in df_toko.loc[row,'product_sold']:
        df_toko.loc[row,'product_sold'] = df_toko.loc[row,'p
roduct_sold'].replace('k', '')
        df_toko.loc[row,'product_sold'] = int(float(df_toko.
loc[row,'product_sold'])*1000)
    elif 'm' in df_toko.loc[row,'product_sold']:
        df_toko.loc[row,'product_sold'] = df_toko.loc[row,'p
roduct_sold'].replace('m', '')
```

```
        df_toko.loc[row,'product_sold'] = int(float(df_toko.
loc[row,'product_sold'])*1000000)

df_toko['product_sold'] = df_toko['product_sold'].astype(int
)



######### PREPROCESS RATING #################
print('PREPROCESS RATING')
df_toko['rating'] = df_toko['rating'].astype(float)



######### PREPROCESS REVIEW #################
print('PREPROCESS REVIEW')

df_toko.review.replace(np.nan,'0',regex=True,inplace=True)

df_toko['review'] = df_toko['review'].str.lower()

df_toko['review'] = df_toko['review'].str.replace('(','')
df_toko['review'] = df_toko['review'].str.replace(')','')
df_toko['review'] = df_toko['review'].str.replace('ulasan','
')

df_toko['review'] = df_toko['review'].astype(int)



######### PREPROCESS OWNER #################
print('PREPROCESS OWNER')
df_toko.owner.replace(np.nan,'no owner information',regex=Tr
ue,inplace=True)

######### PREPROCESS DESCRIPTION #################
print('PREPROCESS DESCRIPTION')

df_toko.review.replace(np.nan,'no description',regex=True,in
place=True)

df_toko['description'] = df_toko['description'].str.encode('
ascii', 'ignore').str.decode('ascii')
df_toko['description'] = df_toko['description'].str.lower()
df_toko['description'].fillna('no description', inplace=True
)

df_toko_description = df_toko['description']

list_deskripsi_toko = df_toko_description.tolist()
```

```
########## INFORMATION EXTRACTION ##################
print('INFORMATION EXTRACTION')
############# CASH ON DELIVERY ############################
df_toko['COD'] = df_toko['description'].str.contains("cod")

searchfor = ['no cod', 'tidak menerima cod','tidak terima co
d', 'tidak cod']
df_toko_noCOD = df_toko[df_toko['description'].str.contains(
'|'.join(searchfor))]
df_toko_noCOD.loc['COD'] = False

df_toko.set_index('link', inplace=True)
df_toko.update(df_toko_noCOD.set_index('link'))
df_toko = df_toko.reset_index()
################################

df_kodepos = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Inst
itut Teknologi Sepuluh Nopember\Semester 8\TA\scraper\file\k
odepos_surabaya.csv',index_col=0)
df_kodepos['kodepos'] = df_kodepos['kodepos'].astype(str)
list_kodepos = df_kodepos['kodepos'].tolist()

for index in range(0,len(list_deskripsi_toko)):
    for kodepos in list_kodepos:
        if 'surabayaa' in list_deskripsi_toko[index]:
            continue
        elif 'surabaya' in list_deskripsi_toko[index] or 'sb
y' in list_deskripsi_toko[index]:
            list_deskripsi_toko[index] = list_deskripsi_toko
[index].replace(kodepos,'')

#find phone number
# phoneNumberRegex =  re.compile(r'(\()?(\+62|62|0)(\d{2,3})
?\)?[ .-]?\d{2,4}[ .-]?\d{2,4}[ .-]?\d{2,21}')
# phoneNumberRegex =  re.compile(r'(\()?(\+62|62|0)(\d{2,3})
?\)?[ .-]?\d{2,4}[ .-]?\d{2,4}[ .-]?\d{2,4}')
# phoneNumberRegex =  re.compile(r'(\+62 ((\d{3}([ -
]\d{3,})([- ]\d{4,})?)|(\d+)))|(\(\d+\) \d+)|\d{3}( \d+)+|(\
d+[ -]\d+)|\d+')
phoneNumberRegex  = re.compile(r'\+?[0-9]+')

#official website
websiteRegex = re.compile(r'(http(s)?:\/\/.)?(www\.)?[-a-zA-
Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-
9@:%_\+.~#?&//=]*)')

#instagram account
```

```
instagramRegex = re.compile(r'(?:^|[^\w])(?:@)([A-Za-z0-
9_](?:(?:[A-Za-z0-9_]|(?:\.(?!\.))){0,28}(?:[A-Za-z0-
9_]))?)')

index=0
for info in list_deskripsi_toko:
    # print(index)
    #extract phone number
    phone_text = phoneNumberRegex.search(str(info))
    #extract website
    websiteText = websiteRegex.search(str(info))
    #extract instagram
    list_instagram = instagramRegex.findall(str(info))
    try:
        phone_check = True
        phone_index = 0
        list_phone = []
        while phone_check:
            # print(index)
            phone = phone_text.group(0)
            if len(phone)>1:
                list_phone.append(phone[0:2].replace('62', '
0')+phone[2:len(phone)])
            else:
                list_phone.append(phone)
            all_phone = ' '.join(list_phone)
            df_toko.loc[index,'phone'] = all_phone
            list_deskripsi_toko[index] = list_deskripsi_toko
[index].replace(phone,'',1)
            phone_text = phoneNumberRegex.search(list_deskri
psi_toko[index])
            phone_index+=1
    except:
        if phone_index == 0:
            df_toko.loc[index,'phone'] = None
    try:
        website = websiteText.group(0)
        if not '@' in website:
            if website.startswith("www") or website.startswi
th("http"):
                df_toko.loc[index,'website'] = website
            elif website.endswith(".com") or website.endswit
h(".id") or website.endswith("/"):
                df_toko.loc[index,'website'] = website
    except:
        df_toko.loc[index,'website'] = None
    try:
        # list_instagram = instagram.group(0)
        list_instagram = list(dict.fromkeys(list_instagram))

        instagram = '\n'.join(list_instagram)
```

```
        df_toko.loc[index,'instagram'] = instagram
    except:
        df_toko.loc[index,'instagram'] = None
    index+=1

# df_toko.phone.replace(np.nan,'no phone', regex=True, inpla
ce=True)
# df_toko.website.replace(np.nan,'no website', regex=True, i
nplace=True)
# df_toko.instagram.replace('','no instagram', regex=True, i
nplace=True)



######### PREPROCESS ADDRESS #################
print('PREPROCESS ADDRESS')

df_toko['address'] = df_toko['address'].str.encode('ascii',
'ignore').str.decode('ascii')

df_toko['address'] = df_toko['address'].str.lower()

df_toko['address'] = df_toko['address'].str.replace('gununga
nyar', 'gunung anyar')
df_toko['address'] = df_toko['address'].str.replace('pabean
cantikan', 'pabean cantian')

df_key_address = pd.read_excel(r'C:\Users\ALDYSYAH\OneDrive
- Institut Teknologi Sepuluh Nopember\Semester 8\TA\scraper\
file\key_address.xlsx')

list_key = df_key_address['key'].tolist()
list_kecamatan_key = df_key_address['kecamatan'].tolist()
list_kelurahan_key = df_key_address['kelurahan'].tolist()

list_filter_area=list_filter_city
for area in list_key:
    list_filter_area.append(area)
list_filter_area.append('no shop location yet')

df_toko_unaddressed = df_toko[~df_toko.address.str.contains(
'|'.join(list_filter_area))]
df_toko = df_toko[df_toko.address.str.contains('|'.join(list
_filter_area))]

df_toko = df_toko.reset_index(drop=True)

list_address = df_toko['address'].tolist()
list_kecamatan_toko = []
list_kelurahan_toko = []
```

```
for address in list_address:
    index=0
    bol_kecamatan=True
    for kelurahan in list_kelurahan:
        if bol_kecamatan:
            if list_kecamatan[index] in address:
                list_kecamatan_toko.append(list_kecamatan[in
dex])
                bol_kecamatan=False
        if kelurahan in address:
            if bol_kecamatan:
                list_kecamatan_toko.append(list_kecamatan[in
dex])
            list_kelurahan_toko.append(kelurahan)
            break
        index+=1
        if index==len(list_kelurahan):
            index_key=0
            for key in list_key:
                if key in address:
                    if bol_kecamatan:
                        list_kecamatan_toko.append(list_keca
matan_key[index_key])
                    list_kelurahan_toko.append(list_keluraha
n_key[index_key])
                    break
                index_key+=1
                if index_key == len(list_key):
                    if bol_kecamatan:
                        list_kecamatan_toko.append('no kecam
atan information')
                    list_kelurahan_toko.append('no kelurahan
 information')
df_toko['kecamatan'] = list_kecamatan_toko
df_toko['kelurahan'] = list_kelurahan_toko



############# PREPROCESS SINCE #################
print('PREPROCESS SINCE')
df_toko[['month_joined','year_joined']] = df_toko.since.str.
split(expand=True)

for row in range(0,len(df_toko.index)):
    try:
        # df_toko.loc[row,'month_joined'] = datetime.strptim
e(df_toko.loc[row,'month_joined'],'%B').month
        df_toko.loc[row,'year_joined'] = int(df_toko.loc[row
,'year_joined'])
    except:
```

```
        pass



# df_toko.dtypes
######## PREPROCESS NUMOF_SATISFACTION #################
print('PREPROCESS NUMOF_SATISFACTION')
df_toko.numOf_satisfied.replace(np.nan,0, regex=True, inplac
e=True)
df_toko.numOf_neutral.replace(np.nan,0, regex=True, inplace=
True)
df_toko.numOf_unsatisfied.replace(np.nan,0, regex=True, inpl
ace=True)

df_toko['numOf_satisfied'] = df_toko['numOf_satisfied'].asty
pe(int)
df_toko['numOf_neutral'] = df_toko['numOf_neutral'].astype(i
nt)
df_toko['numOf_unsatisfied'] = df_toko['numOf_unsatisfied'].
astype(int)

######## PREPROCESS ACTIVE_PRODUCT #################
print('PREPROCESS ACTIVE_PRODUCT')
df_toko = df_toko[df_toko['active_product'].notna()]
df_toko['active_product'] = df_toko['active_product'].astype
(int)

######## PREPROCESS SUCCESS_TRANSACTION ########
print('PREPROCESS SUCCESS_TRANSACTION')
df_toko['success_transaction'] = df_toko['success_transactio
n'].astype(int)

############ PREPROCESS PHONE #################
print('PREPROCESS PHONE')

# df_toko['phone'] = df_toko.phone.str.replace('(','')
# df_toko['phone'] = df_toko.phone.str.replace(')','')
df_toko['phone'] = df_toko.phone.str.replace('\+62','0')
df_toko['phone'] = df_toko.phone.str.replace(' ', '')
df_toko['phone'] = df_toko.phone.str.replace('+', '')
# df_toko['phone'] = df_toko.phone.str.replace('.', '')
df_toko['phone'] = df_toko['phone'].astype(str)

list_phone = df_toko['phone'].tolist()
list_description = df_toko['description'].tolist()

# # df_phone  = pd.DataFrame({'description':list_description
})
# df_phone  = pd.DataFrame({'description':list_description,'
phone':list_phone})
```

```
# df_phone = df_phone[~df_phone.phone.str.contains('None')]
# df_phone = df_phone[~df_phone.phone.str.contains('nan')]
list_operator = ['031',
                 '0811','0812','0813','0821','0822','0852','
0853','0823','0851',
                 '0814','0815','0816','0855','0856','0857','
0858',
                 '0817','0818','0819','0859','0877','0878',
                 '0838','0831','0832','0833',
                 '0895','0896','0897','0898','0899',
                 '0881','0882','0883','0884','0885','0886','
0887','0888','0889',
                 '0828',
                 '0868',
                 '0810','0820','0824','0825','0826','0827','
0829','0830','0834','0835','0836',
                 '0837','0839','0840','0841','0842','0843','
0844','0845','0846','0847','0848',
                 '0849','0850','0854','0860','0861','0862','
0863','0864','0865','0866','0867',
                 '0869','0870','0871','0872','0873','0874','
0875','0876','0879',
                 '0891','0892','0893','0894','0901','0902']

# index=0
# for phone in list_phone:
#     if phone.startswith('62'):
#         phone[0:2].replace('62','0')
#         list_phone[index] = phone
#     index+=1

index=0
for phone in list_phone:
    print(index)
    if len(list_phone[index]) > 13:
        list_number_operator = re.findall('|'.join(list_oper
ator),list_phone[index])
        list_phone_temp = []
        phone_temp = list_phone[index]
        index_operator=0
        index_plus = 0
        same_operator = False
        print('list_number_operator: '+str(len(list_number_o
perator)))
        if len(list_number_operator) == 0:
            phone_temp = 'nan'
            list_phone_temp.append(phone_temp)
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
```

```
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        elif len(list_number_operator) == 1:
            start_index_phone = phone_temp.find(list_number_
operator[index_operator])
            phone_temp = phone_temp[start_index_phone:len(ph
one_temp)]
            if len(phone_temp) > 13:
                if phone_temp.startswith('031'):
                    list_phone_temp.append(phone_temp[0:10])
                else:
                    list_phone_temp.append(phone_temp[0:12])
            elif len(phone_temp) >= 10:
                list_phone_temp.append(phone_temp[0:len(phon
e_temp)])
            else:
                list_phone_temp.append('nan')
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        elif len(list_number_operator) > 1:
            phone_slice = list_phone[index]
            for number_operator in list_number_operator:
                print('index_operator: '+ str(index_operator
))
                print(number_operator)
                print('phone_slice: '+phone_slice)
                start_index_phone = phone_slice.find(number_
operator)
                try:
                    end_index_phone = phone_slice[4:].find(l
ist_number_operator[index_operator+1])+4
                except:
                    end_index_phone  = len(phone_slice)
                phone_temp = phone_slice[start_index_phone:e
nd_index_phone]
                print('phone_temp: ' + phone_temp)
                # print(phone_temp[start_index_phone:end_ind
ex_phone])
                if len(phone_temp) < 10:
                    index_plus = 2
                    same_operator = True
                    while len(phone_temp) < 10:
                        try:
                            end_index_phone = phone_slice[4:
].find(list_number_operator[index_operator+index_plus])+4
                        except:
```

```
                            end_index_phone = len(phone_slic
e)
                        phone_temp = phone_slice[start_index
_phone:end_index_phone]
                        print('phone_temp: '+phone_temp)
                        if end_index_phone >= len(phone_slic
e):
                            break
                        index_plus+=1
                elif len(phone_temp) > 13:
                    list_phone_temp.append(phone_temp[0:12])
                elif len(phone_temp) >= 10 and len(phone_tem
p) <= 13:
                    if len(phone_temp)==13:
                        if phone_temp.startswith(('0811','08
12','0813','0821','0823','0852','0853','0859','0877','0878',
'0857','0858','0814','0815','0896','0897','0898','0899','088
1','0882','0883','0884','0885','0886','0887','0817','0818','
0819','0855','0856','0816')):
                            list_phone_temp.append(phone_tem
p[0:12])
                    elif len(phone_temp)==10:
                        if phone_temp.startswith(('0811','08
12','0813','0821','0823','0852','0853','0859','0877','0878',
'0857','0858','0814','0815','0896','0897','0898','0899','088
1','0882','0883','0884','0885','0886','0887','0817','0818','
0819','0855','0856','0816')):
                            list_phone_temp.append(phone_sli
ce[0:12])
                if len(phone_temp) < 10:
                    phone_temp='nan'
                    list_phone_temp.append(phone_temp)
                elif len(phone_temp) >= 10 and len(phone_tem
p) <= 13:
                    print('phone_temp: '+phone_temp)
                    if phone_temp.startswith('031'):
                        list_phone_temp.append(phone_temp[0:
10])
                    else:
                        if len(phone_temp)==10:
                            if phone_temp.startswith(('0811'
,'0817','0818','0819','0855','0856','0816')):
                                list_phone_temp.append(phone
_temp)
                        elif len(phone_temp)==13:
                            if phone_temp.startswith(('0811'
,'0812','0813','0821','0823','0852','0853','0859','0877','08
78','0857','0858','0814','0815','0896','0897','0898','0899',
'0881','0882','0883','0884','0885','0886','0887','0817','081
8','0819','0855','0856','0816')):
```

```
                                list_phone_temp.append(phone
_temp[0:12])
                        else:
                            list_phone_temp.append(phone_tem
p)
                elif len(phone_temp) > 13:
                    if phone_temp.startswith('031'):
                        list_phone_temp.append(phone_temp[0:
10])
                    else:
                        list_phone_temp.append(phone_temp[0:
12])
                phone_slice = phone_slice[end_index_phone:le
n(list_phone[index])]
                print('phone_slice: ' + phone_slice)
                if not same_operator:
                    index_operator+=1
                else:
                    if len(phone_slice) > 13:
                        if phone_temp.startswith('031'):
                            list_phone_temp.append(phone_tem
p[0:10])
                        else:
                            list_phone_temp.append(phone_sli
ce[0:12])
                    elif len(phone_slice) < 10:
                        list_phone_temp.append('nan')
                    index_operator+=index_plus
                if index_operator >= len(list_number_operato
r):
                    break
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        else:
            phone_temp = 'nan'
            list_phone_temp.append(phone_temp)
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
    elif len(list_phone[index]) >= 10:
        list_number_operator = re.findall('|'.join(list_oper
ator),list_phone[index])
        list_phone_temp = []
        phone_temp = list_phone[index]
```

```
        index_operator=0
        if len(list_number_operator) == 0:
            phone_temp = 'nan'
            list_phone_temp.append(phone_temp)
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        else:
            start_index_phone = phone_temp.find(list_number_
operator[index_operator])
            phone_temp = phone_temp[start_index_phone:len(ph
one_temp)]
            if phone_temp.startswith('031'):
                list_phone_temp.append(phone_temp[0:10])
            else:
                list_phone_temp.append(phone_temp[0:12])
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
    elif len(list_phone[index]) < 10:
        list_phone_temp = []
        phone_temp = 'nan'
        list_phone_temp.append(phone_temp)
        for item in list_phone_temp:
            if item == 'nan':
                list_phone_temp.remove(item)
        list_phone_temp = list(dict.fromkeys(list_phone_temp
))
        list_phone[index] = '\n'.join(list_phone_temp)
    index+=1

df_phone  = pd.DataFrame({'phone':list_phone,'description':l
ist_description})
df_phone['phone'].replace('', 'no phone', inplace=True)
df_phone['phone'].replace('\nnan','',inplace=True)
df_phone['phone'].replace('nan', 'no phone', inplace=True)

df_toko['phone'] = df_phone['phone']

df_description_nol = df_phone[df_phone.description.str.conta
ins('nol')]
df_phone_62 = df_phone[df_phone.phone.str.contains('62')]
```

```
############# PREPROCESS WEBSITE ###########
df_toko['website'].fillna('no website',inplace=True)
# df_toko['website'] = df_toko[df_toko.website.str.replace('
nan','no website')]
df_toko['website'].replace('', 'no website', inplace=True)

list_website=df_toko['website'].tolist()

df_toko = df_toko.reset_index(drop=True)

list_instagram=[]
for index,website in enumerate(list_website):
    if 'instagram' in website:
        list_instagram.append(website)
        df_toko.loc[index,'website'] = 'no website'
    else:
        if not str(df_toko.loc[index,'instagram']) == 'nan':
            list_instagram.append(df_toko.loc[index,'instagr
am'])
        else:
            list_instagram.append('no instagram')

############# PREPROCESS INSTAGRAM ###########
df_toko['instagram'].fillna('no instagram',inplace=True)
# df_toko['instagram'] = df_toko[df_toko.instagram.str.repla
ce('nan','no instagram')]
df_toko['instagram'].replace('', 'no instagram', inplace=Tru
e)

############# REARRANGE COLUMN ORDER #########
list(df_toko.columns.values)
df_toko = df_toko[['name',
  'link',
  'seller_type',
  'city',
  'followers',
  'reputation_point',
  'product_sold',
  'rating',
  'review',
  'owner',
  'description',
  'address',
  'kecamatan',
  'kelurahan',
  'month_joined',
  'year_joined',
```

```
  'numOf_delivery_services',
  'delivery_services',
  'numOf_satisfied',
  'numOf_neutral',
  'numOf_unsatisfied',
  'process_speed',
  'numOf_storefront',
  'storefront',
  'active_product',
  'success_transaction',
  'COD',
  'phone',
  'website',
  'instagram']]


################ EXPORT TO CSV ##############
df_toko.to_csv(r'C:\Users\ALDYSYAH\OneDrive - Institut Tekno
logi Sepuluh Nopember\Semester 8\TA\Scraping\Result\Tokopedi
a\4 - Tokopedia_processed.csv')
```

Tabel A.14 Kode *pre-process* data Shopee

Pra-proses data Shopee	
Fungsi	Melakukan pembersihan, transformasi pada data Shopee
Bahasa	Python

```
import pandas as pd
import numpy as np
import re
# from datetime import datetime

################### LOAD DATA ####################
df_toko_1 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Shopee\info_toko_shopee_kesehatan1.csv')
df_toko_2 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Shopee\info_toko_shopee_kesehatan2.csv')

#df_toko_3 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Inst
itut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Resul
t\Tokopedia\info_toko_tokopedia_kesehatan3.csv',index_col=0)
#df_toko_5 = pd.read_csv('D:\\Documents\\Kuliah\\TA\\scraper
\\Google Business\\googleBusiness_tokped_5.csv',index_col=0)

df_toko = df_toko_1.append(df_toko_2)
```

```
df_kelurahan = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\scraper\file
\kelurahan.csv',index_col=0)
#lowercase kecamatan dan kelurahan
df_kelurahan['kecamatan'] = df_kelurahan['kecamatan'].str.lo
wer()
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.lo
wer()
#replace word 'kecamatan'
df_kelurahan['kecamatan'] = df_kelurahan.kecamatan.str.repla
ce('kecamatan ','')
df_kelurahan['kecamatan'] = df_kelurahan.kecamatan.str.repla
ce('kecamatan','')
#replace word 'kelurahan'
df_kelurahan['kelurahan'] = df_kelurahan.kelurahan.str.repla
ce('kelurahan ','')
df_kelurahan['kelurahan'] = df_kelurahan.kelurahan.str.repla
ce('kelurahan','')

list_kecamatan = df_kelurahan['kecamatan'].tolist()
list_kelurahan = df_kelurahan['kelurahan'].tolist()



################## DROP DUPLICATE #################
df_toko = df_toko.drop_duplicates('link')



################ PREPROCESS CITY #################
print('PREPROCESS CITY')

#Drop NaN city
df_toko = df_toko[df_toko['city'].notna()]
#Lowercase city
df_toko['city'] = df_toko['city'].str.lower()
#Load file kelurahan

# list_kecamatan = list(dict.fromkeys(list_kecamatan))
list_filter_city = ['surabaya','sby']
for kecamatan in list_kecamatan:
    list_filter_city.append(kecamatan)

# df_toko['city'] = df_toko['city'].str.replace('gununganyar
', 'gunung anyar')
# df_toko['city'] = df_toko['city'].str.replace('pabean cant
ikan', 'pabean cantian')

#drop if city not surabaya
```

```
df_toko_notSurabaya = df_toko[~df_toko.city.str.contains('|'
.join(list_filter_city))]
df_toko = df_toko[df_toko.city.str.contains('|'.join(list_fi
lter_city))]
df_toko = df_toko.reset_index(drop=True)


############ PREPROCESS SELLER_TYPE #################
print('PREPROCESS SELLER_TYPE')
df_toko.seller_type.replace(np.nan, 'Regular Seller', regex=
True, inplace=True)


############ PREPROCESS PRODUCT_CATEGORY #################
print('PREPROCESS PRODUCT_CATEGORY')
df_toko.product_category.replace('Semua Produk\n', '', regex
=True, inplace=True)


############ PREPROCESS NUMOF_PRODUCT #################
print('PREPROCESS NUMOF_PRODUCT')
df_toko.numOf_product.replace(',', '.', regex=True, inplace=
True)

for row in range(0,len(df_toko.index)):
    if 'RB' in df_toko.loc[row,'numOf_product']:
        df_toko.loc[row,'numOf_product'] = df_toko.loc[row,'
numOf_product'].replace('RB', '')
        df_toko.loc[row,'numOf_product'] = int(float(df_toko
.loc[row,'numOf_product'])*1000)

# df_toko.numOf_product.replace('.0', '', regex=True, inplac
e=True)
df_toko['numOf_product'] = df_toko['numOf_product'].astype(f
loat)
df_toko['numOf_product'] = df_toko['numOf_product'].astype(i
nt)

############ PREPROCESS CHAT_PERFORMANCE #################
print('PREPROCESS CHAT_PERFORMANCE')
df_toko['chat_performance'] = df_toko['chat_performance'].st
r.replace('...%', '0%')
df_toko['chat_performance'] = df_toko['chat_performance'].st
r.replace('\(hitungan ', '')
df_toko['chat_performance'] = df_toko['chat_performance'].st
r.replace(')', '')

df_toko[['chat_performance_percentage','chat_performance_uni
t']] = df_toko.chat_performance.str.split(expand=True)
```

```
for row in range(0,len(df_toko.index)):
    try:
        if df_toko.loc[row,'chat_performance_percentage']=='
0%':
            df_toko.loc[row,'chat_performance_percentage'] =
 np.nan
            df_toko.loc[row,'chat_performance_unit'] = np.na
n
        else:
            df_toko.loc[row,'chat_performance_percentage'] =
 float(df_toko.loc[row,'chat_performance_percentage'].strip(
'%'))/100
        # df_toko.loc[row,'chat_performance_unit'] = int(df_
toko.loc[row,'year_joined'])
    except:
        pass


############ PREPROCESS FOLLOWERS #################
print('PREPROCESS FOLLOWERS')


df_toko.followers.replace(np.nan, '0', regex=True, inplace=T
rue)

df_toko.followers.replace(',', '.', regex=True, inplace=True
)
# df_toko.followers.replace('.0', '', regex=True, inplace=Tr
ue)

for row in range(0,len(df_toko.index)):
    if 'RB' in df_toko.loc[row,'followers']:
        df_toko.loc[row,'followers'] = df_toko.loc[row,'foll
owers'].replace('RB', '')
        df_toko.loc[row,'followers'] = int(float(df_toko.loc
[row,'followers'])*1000)

df_toko['followers'] = df_toko['followers'].astype(float)
df_toko['followers'] = df_toko['followers'].astype(int)

######## PREPROCESS RATING_REVIEW #################
print('PREPROCESS RATING_REVIEW')
df_toko['rating_review'] = df_toko['rating_review'].str.repl
ace('(', '')
df_toko['rating_review'] = df_toko['rating_review'].str.repl
ace(' Penilaian', '')
df_toko['rating_review'] = df_toko['rating_review'].str.repl
ace(')','')
df_toko['rating_review'] = df_toko['rating_review'].str.repl
ace(',','.')
```

```
df_toko[['rating','review']] = df_toko.rating_review.str.spl
it(expand=True)


#replace rating 0 with nan
df_toko.rating.replace('0',np.nan, regex=True, inplace=True)

df_toko['review'] = df_toko['review'].astype(str)
df_toko.review.replace('None','nan', inplace=True)

for row in range(0,len(df_toko.index)):
    if 'RB' in df_toko.loc[row,'review']:
        df_toko.loc[row,'review'] = df_toko.loc[row,'review'
].replace('RB', '')
        df_toko.loc[row,'review'] = int(float(df_toko.loc[ro
w,'review'])*1000)

df_toko['rating'] = df_toko['rating'].astype(float)
df_toko['review'] = df_toko['review'].astype(float)


######### PREPROCESS DESCRIPTION #################
print('PREPROCESS DESCRIPTION')

df_toko.review.replace(np.nan,'no description',regex=True,in
place=True)

df_toko['description'] = df_toko['description'].str.encode('
ascii', 'ignore').str.decode('ascii')
df_toko['description'] = df_toko['description'].str.lower()
df_toko['description'].fillna('no description', inplace=True
)

df_toko_description = df_toko['description']

list_deskripsi_toko = df_toko_description.tolist()


########## INFORMATION EXTRACTION #################
print('INFORMATION EXTRACTION')
############# CASH ON DELIVERY ##############################
df_toko['COD'] = df_toko['description'].str.contains("cod")

searchfor = ['no cod', 'tidak menerima cod','tidak terima co
d', 'tidak cod']
df_toko_noCOD = df_toko[df_toko['description'].str.contains(
'|'.join(searchfor))]
df_toko_noCOD.loc['COD'] = False
```

```
df_toko.set_index('link', inplace=True)
df_toko.update(df_toko_noCOD.set_index('link'))
df_toko = df_toko.reset_index()
################################

# df_kodepos = pd.read_csv('D:\\Documents\\Kuliah\\TA\\scrap
er\\file\\kodepos_surabaya.csv',index_col=0)
# df_kodepos['kodepos'] = df_kodepos['kodepos'].astype(str)
# list_kodepos = df_kodepos['kodepos'].tolist()

# for index in range(0,len(list_deskripsi_toko)):
#     for kodepos in list_kodepos:
#         if 'surabayaa' in list_deskripsi_toko[index]:
#             continue
#         elif 'surabaya' in list_deskripsi_toko[index] or '
sby' in list_deskripsi_toko[index]:
#             list_deskripsi_toko[index] = list_deskripsi_to
ko[index].replace(kodepos,'')

#find phone number
# phoneNumberRegex =  re.compile(r'(\()?(\+62|62|0)(\d{2,3})
?\)?[ .-]?\d{2,4}[ .-]?\d{2,4}[ .-]?\d{2,21}')
phoneNumberRegex =  re.compile(r'(\()?(\+62|62|0)(\d{2,3})?\
)?[ .-]?\d{2,4}[ .-]?\d{2,4}[ .-]?\d{2,4}')
# phoneNumberRegex =  re.compile(r'(\+62 ((\d{3}([ -
]\d{3,})([- ]\d{4,})?)|(\d+)))|(\(\d+\) \d+)|\d{3}( \d+)+|(\
d+[ -]\d+)|\d+')
# phoneNumberRegex  = re.compile(r'\+?[0-9]+')

#official website
websiteRegex = re.compile(r'(http(s)?:\/\/.)?(www\.)?[-a-zA-
Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-
9@:%_\+.~#?&//=]*)')

#instagram account
instagramRegex = re.compile(r'(?:^|[^\w])(?:@)([A-Za-z0-
9_](?:(?:[A-Za-z0-9_]|(?:\.(?!\.))){0,28}(?:[A-Za-z0-
9_]))?)')

index=0
for info in list_deskripsi_toko:
    # print(index)
    #extract phone number
    phone_text = phoneNumberRegex.search(str(info))
    #extract website
    websiteText = websiteRegex.search(str(info))
    #extract instagram
    list_instagram = instagramRegex.findall(str(info))
    try:
        phone_check = True
```

```
        phone_index = 0
        list_phone = []
        while phone_check:
            # print(index)
            phone = phone_text.group(0)
            if len(phone)>1:
                list_phone.append(phone[0:2].replace('62', '
0')+phone[2:len(phone)])
            else:
                list_phone.append(phone)
            all_phone = ' '.join(list_phone)
            df_toko.loc[index,'phone'] = all_phone
            list_deskripsi_toko[index] = list_deskripsi_toko
[index].replace(phone,'',1)
            phone_text = phoneNumberRegex.search(list_deskri
psi_toko[index])
            phone_index+=1
    except:
        if phone_index == 0:
            df_toko.loc[index,'phone'] = None
    try:
        website = websiteText.group(0)
        if not '@' in website:
            if website.startswith("www") or website.startswi
th("http"):
                df_toko.loc[index,'website'] = website
            elif website.endswith(".com") or website.endswit
h(".id") or website.endswith("/"):
                df_toko.loc[index,'website'] = website
    except:
        df_toko.loc[index,'website'] = None
    try:
        # list_instagram = instagram.group(0)
        list_instagram = list(dict.fromkeys(list_instagram))

        instagram = '\n'.join(list_instagram)
        df_toko.loc[index,'instagram'] = instagram
    except:
        df_toko.loc[index,'instagram'] = None
    index+=1

# df_toko.phone.replace(np.nan,'no phone', regex=True, inpla
ce=True)
# df_toko.website.replace(np.nan,'no website', regex=True, i
nplace=True)
# df_toko.instagram.replace('','no instagram', regex=True, i
nplace=True)
```

```
############ PREPROCESS PHONE #################
print('PREPROCESS PHONE')

df_toko['phone'] = df_toko.phone.str.replace('(','')
df_toko['phone'] = df_toko.phone.str.replace(')','')
df_toko['phone'] = df_toko.phone.str.replace('\+62','0')
df_toko['phone'] = df_toko.phone.str.replace(' ', '')
df_toko['phone'] = df_toko.phone.str.replace('+', '')
df_toko['phone'] = df_toko.phone.str.replace('.', '')
df_toko['phone'] = df_toko.phone.str.replace('-', '')
df_toko['phone'] = df_toko['phone'].astype(str)

list_phone = df_toko['phone'].tolist()
list_description = df_toko['description'].tolist()

# # df_phone  = pd.DataFrame({'description':list_description
})
# df_phone  = pd.DataFrame({'description':list_description,'
phone':list_phone})

# df_phone = df_phone[~df_phone.phone.str.contains('None')]
# df_phone = df_phone[~df_phone.phone.str.contains('nan')]
list_operator = ['031',
                 '0811','0812','0813','0821','0822','0852','
0853','0823','0851',
                 '0814','0815','0816','0855','0856','0857','
0858',
                 '0817','0818','0819','0859','0877','0878',
                 '0838','0831','0832','0833',
                 '0895','0896','0897','0898','0899',
                 '0881','0882','0883','0884','0885','0886','
0887','0888','0889',
                 '0828',
                 '0868',
                 '0810','0820','0824','0825','0826','0827','
0829','0830','0834','0835','0836',
                 '0837','0839','0840','0841','0842','0843','
0844','0845','0846','0847','0848',
                 '0849','0850','0854','0860','0861','0862','
0863','0864','0865','0866','0867',
                 '0869','0870','0871','0872','0873','0874','
0875','0876','0879',
                 '0891','0892','0893','0894','0901','0902']

# index=0
# for phone in list_phone:
#     if phone.startswith('62'):
#         phone[0:2].replace('62','0')
#         list_phone[index] = phone
#     index+=1
```

```
index=0
for phone in list_phone:
    print(index)
    if len(list_phone[index]) > 13:
        list_number_operator = re.findall('|'.join(list_oper
ator),list_phone[index])
        list_phone_temp = []
        phone_temp = list_phone[index]
        index_operator=0
        index_plus = 0
        same_operator = False
        print('list_number_operator: '+str(len(list_number_o
perator)))
        if len(list_number_operator) == 0:
            phone_temp = 'nan'
            list_phone_temp.append(phone_temp)
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        elif len(list_number_operator) == 1:
            start_index_phone = phone_temp.find(list_number_
operator[index_operator])
            phone_temp = phone_temp[start_index_phone:len(ph
one_temp)]
            if len(phone_temp) > 13:
                if phone_temp.startswith('031'):
                    list_phone_temp.append(phone_temp[0:10])
                else:
                    list_phone_temp.append(phone_temp[0:12])
            elif len(phone_temp) >= 10:
                list_phone_temp.append(phone_temp[0:len(phon
e_temp)])
            else:
                list_phone_temp.append('nan')
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        elif len(list_number_operator) > 1:
            phone_slice = list_phone[index]
            for number_operator in list_number_operator:
                print('index_operator: '+ str(index_operator
))
                print(number_operator)
                print('phone_slice: '+phone_slice)
```

```
                start_index_phone = phone_slice.find(number_
operator)
                try:
                    end_index_phone = phone_slice[4:].find(l
ist_number_operator[index_operator+1])+4
                except:
                    end_index_phone  = len(phone_slice)
                phone_temp = phone_slice[start_index_phone:e
nd_index_phone]
                print('phone_temp: ' + phone_temp)
                # print(phone_temp[start_index_phone:end_ind
ex_phone])
                if len(phone_temp) < 10:
                    index_plus = 2
                    same_operator = True
                    while len(phone_temp) < 10:
                        try:
                            end_index_phone = phone_slice[4:
].find(list_number_operator[index_operator+index_plus])+4
                        except:
                            end_index_phone = len(phone_slic
e)
                        phone_temp = phone_slice[start_index
_phone:end_index_phone]
                        print('phone_temp: '+phone_temp)
                        if end_index_phone >= len(phone_slic
e):
                            break
                        index_plus+=1
                elif len(phone_temp) > 13:
                    list_phone_temp.append(phone_temp[0:12])
                elif len(phone_temp) >= 10 and len(phone_tem
p) <= 13:
                    if len(phone_temp)==13:
                        if phone_temp.startswith(('0811','08
12','0813','0821','0823','0852','0853','0859','0877','0878',
'0857','0858','0814','0815','0896','0897','0898','0899','088
1','0882','0883','0884','0885','0886','0887','0817','0818','
0819','0855','0856','0816')):
                            list_phone_temp.append(phone_tem
p[0:12])
                    elif len(phone_temp)==10:
                        if phone_temp.startswith(('0811','08
12','0813','0821','0823','0852','0853','0859','0877','0878',
'0857','0858','0814','0815','0896','0897','0898','0899','088
1','0882','0883','0884','0885','0886','0887','0817','0818','
0819','0855','0856','0816')):
                            list_phone_temp.append(phone_sli
ce[0:12])
                if len(phone_temp) < 10:
                    phone_temp='nan'
```

```
                    list_phone_temp.append(phone_temp)
                elif len(phone_temp) >= 10 and len(phone_tem
p) <= 13:
                    print('phone_temp: '+phone_temp)
                    if phone_temp.startswith('031'):
                        list_phone_temp.append(phone_temp[0:
10])
                    else:
                        if len(phone_temp)==10:
                            if phone_temp.startswith(('0811'
,'0817','0818','0819','0855','0856','0816')):
                                list_phone_temp.append(phone
_temp)
                        elif len(phone_temp)==13:
                            if phone_temp.startswith(('0811'
,'0812','0813','0821','0823','0852','0853','0859','0877','08
78','0857','0858','0814','0815','0896','0897','0898','0899',
'0881','0882','0883','0884','0885','0886','0887','0817','081
8','0819','0855','0856','0816')):
                                list_phone_temp.append(phone
_temp[0:12])
                        else:
                            list_phone_temp.append(phone_tem
p)
                elif len(phone_temp) > 13:
                    if phone_temp.startswith('031'):
                        list_phone_temp.append(phone_temp[0:
10])
                    else:
                        list_phone_temp.append(phone_temp[0:
12])
                phone_slice = phone_slice[end_index_phone:le
n(list_phone[index])]
                print('phone_slice: ' + phone_slice)
                if not same_operator:
                    index_operator+=1
                else:
                    if len(phone_slice) > 13:
                        if phone_temp.startswith('031'):
                            list_phone_temp.append(phone_tem
p[0:10])
                        else:
                            list_phone_temp.append(phone_sli
ce[0:12])
                    elif len(phone_slice) < 10:
                        list_phone_temp.append('nan')
                    index_operator+=index_plus
                if index_operator >= len(list_number_operato
r):
                    break
            for item in list_phone_temp:
```

```
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        else:
            phone_temp = 'nan'
            list_phone_temp.append(phone_temp)
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
    elif len(list_phone[index]) >= 10:
        list_number_operator = re.findall('|'.join(list_oper
ator),list_phone[index])
        list_phone_temp = []
        phone_temp = list_phone[index]
        index_operator=0
        if len(list_number_operator) == 0:
            phone_temp = 'nan'
            list_phone_temp.append(phone_temp)
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
        else:
            start_index_phone = phone_temp.find(list_number_
operator[index_operator])
            phone_temp = phone_temp[start_index_phone:len(ph
one_temp)]
            if phone_temp.startswith('031'):
                list_phone_temp.append(phone_temp[0:10])
            else:
                list_phone_temp.append(phone_temp[0:12])
            for item in list_phone_temp:
                if item == 'nan':
                    list_phone_temp.remove(item)
            list_phone_temp = list(dict.fromkeys(list_phone_
temp))
            list_phone[index] = '\n'.join(list_phone_temp)
    elif len(list_phone[index]) < 10:
        list_phone_temp = []
        phone_temp = 'nan'
        list_phone_temp.append(phone_temp)
        for item in list_phone_temp:
            if item == 'nan':
                list_phone_temp.remove(item)
```

```
        list_phone_temp = list(dict.fromkeys(list_phone_temp
))
        list_phone[index] = '\n'.join(list_phone_temp)
    index+=1

df_phone  = pd.DataFrame({'phone':list_phone,'description':l
ist_description})
df_phone['phone'].replace('', 'no phone', inplace=True)
# df_phone['phone'] = df_phone.phone.str.replace('\nnan','')

df_toko['phone'] = df_phone['phone']

df_description_nol = df_phone[df_phone.description.str.conta
ins('nol')]
df_phone_62 = df_phone[df_phone.phone.str.contains('62')]



############# PREPROCESS WEBSITE ###########
df_toko['website'].fillna('no website',inplace=True)
# df_toko['website'] = df_toko[df_toko.website.str.replace('
nan','no website')]
df_toko['website'].replace('', 'no website', inplace=True)

list_website=df_toko['website'].tolist()

list_instagram=[]
for index,website in enumerate(list_website):
    if 'instagram' in website:
        list_instagram.append(website)
        df_toko.loc[index,'website'] = 'no website'
    else:
        if not str(df_toko.loc[index,'instagram']) == 'nan':
            list_instagram.append(df_toko.loc[index,'instagr
am'])
        else:
            list_instagram.append('no instagram')

# df_toko['tokopedia'] = list_tokopedia
# # [df_toko.website.str.contains('instagram')]
# df_toko['shopee'] = list_shopee
# [df_toko.website.str.contains('tokopedia')]
df_toko['instagram'] = list_instagram
# df_toko['website'][df_toko.website.str.contains('shopee')]

############# PREPROCESS WEBSITE ###########
df_toko['instagram'].fillna('no instagram',inplace=True)
# df_toko['instagram'] = df_toko[df_toko.instagram.str.repla
ce('nan','no instagram')]
```

```
df_toko['instagram'].replace('', 'no instagram', inplace=Tru
e)

############# REARRANGE COLUMN ORDER #########
list(df_toko.columns.values)
df_toko = df_toko[['name',
  'link',
  'seller_type',
  'city',
  'followers',
  'rating',
  'review',
  'chat_performance_percentage',
  'chat_performance_unit',
  'since',
  'numOf_product',
  'product_category',
  'voucher',
  'numOf_media',
  'description',
  'COD',
  'phone',
  'website',
  'instagram']]

################ EXPORT TO CSV ##############
df_toko.to_csv(r'C:\Users\ALDYSYAH\OneDrive - Institut Tekno
logi Sepuluh Nopember\Semester 8\TA\Scraping\Result\Shopee\4
 - Shopee_Processed.csv')
```

Tabel A.15 Kode *pre-process* data Instagram

Pra-proses data Instagram	
Fungsi	Melakukan pembersihan, transformasi pada data Instagram
Bahasa	Python

```
import pandas as pd

df_toko_0 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\instagram_info_toko_tokopedia_kesehatan1.csv',ind
ex_col=0)
df_toko_1 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\instagram_info_toko_tokopedia_kesehatan2.csv',ind
ex_col=0)
df_toko_2 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\instagram_info_toko_tokopedia_kesehatan3.csv',ind
ex_col=0)

df_toko = df_toko_0.append(df_toko_1, ignore_index = True)
df_toko = df_toko.append(df_toko_2, ignore_index = True)
df_toko['account'] = df_toko['account'].str.encode('ascii',
'ignore').str.decode('ascii')
df_toko['description'] = df_toko['description'].str.encode('
ascii', 'ignore').str.decode('ascii')
df_toko['store'] = df_toko['store'].str.encode('ascii', 'ign
ore').str.decode('ascii')

df_toko['account'] = df_toko['account'].astype(str)
df_toko['description'] = df_toko['description'].astype(str)
df_toko['store'] = df_toko['store'].astype(str)

df_toko['city'] = df_toko['city'].str.lower()
df_toko['store'] = df_toko['store'].str.lower()

df_kecamatan = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\scraper\file
\kecamatan.csv',index_col=0)
list_kecamatan = df_kecamatan['kecamatan'].tolist()

list_filter_city = ['surabaya','sby']
for place in list_kecamatan:
    list_filter_city.append(place.lower())
df_toko = df_toko[df_toko['city'].notna()]
df_toko_surabaya = df_toko[df_toko.city.str.contains('|'.joi
n(list_filter_city))]
df_toko_surabaya_noHash = df_toko_surabaya[~df_toko_surabaya
.account.str.startswith('#')]
```

```
df_toko_surabaya_noHash.to_csv(r'C:\Users\ALDYSYAH\OneDrive
- Institut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping
\Result\ready_similarity_tokopedia_kesehatan.csv')
```

Tabel A.16 Kode *pre-process* data Google My Business

Pra-proses data Google My Business	
Fungsi	Melakukan pembersihan, transformasi pada data Google My Business
Bahasa	Python

```
import pandas as pd
import numpy as np
from datetime import datetime
from datetime import timedelta

################## LOAD DATA #################
df_toko_1 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDr
ive - Institut Teknologi Sepuluh Nopember\Semester 8\TA\Scra
ping\Result\Google Business\googleBusiness_kesehatan_Surabay
a1.csv',index_col=0)
df_toko_2 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Google Business\googleBusiness_kesehatan_Surabaya2.csv',ind
ex_col=0)
df_toko_3 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Google Business\googleBusiness_kesehatan_Surabaya3.csv',ind
ex_col=0)
df_toko_4 = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Google Business\googleBusiness_kesehatan_Surabaya4.csv',ind
ex_col=0)
#df_toko_5 = pd.read_csv('D:\\Documents\\Kuliah\\TA\\scraper
\\Google Business\\googleBusiness_tokped_5.csv',index_col=0)

df_toko = df_toko_1.append(df_toko_2)
df_toko = df_toko.append(df_toko_3)
df_toko = df_toko.append(df_toko_4)
#df_toko = df_toko.append(df_toko_5)

df_category = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Ins
titut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Resu
lt\Google Business\kategori.csv')
# df_category_tokped = pd.read_csv('D:\\Documents\\Kuliah\\T
A\\scraper\\file\\kategori.csv',index_col=0)
```

Pra-proses data Google My Business

```
list_category = df_category['kategori'].tolist()

############### ADDITIONAL CATEGORY ##########
add_category = ['acupuncturist','pharmacy','orthopedic clini
c', 'cancer treatment','herb shop']
for cat in add_category:
    list_category.append(cat)

############ PREPROCESS KELURAHAN #############
df_kelurahan = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\scraper\file
\kelurahan.csv',index_col=0)
df_kelurahan['kecamatan'] = df_kelurahan['kecamatan'].str.lo
wer()
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.lo
wer()

df_kelurahan['kecamatan'] = df_kelurahan.kecamatan.str.repla
ce('kecamatan ','')
df_kelurahan['kecamatan'] = df_kelurahan.kecamatan.str.repla
ce('kecamatan','')

df_kelurahan['kelurahan'] = df_kelurahan.kelurahan.str.repla
ce('kelurahan ','')
df_kelurahan['kelurahan'] = df_kelurahan.kelurahan.str.repla
ce('kelurahan','')

df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.re
place('kalirungkut', 'kali rungkut')
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.re
place('penjaringansari', 'penjaringan sari')
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.re
place('medoan ayu', 'medokan ayu')
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.re
place('tambak rejo', 'tambakrejo')
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.re
place('waru gunung', 'warugunung')
df_kelurahan['kelurahan'] = df_kelurahan['kelurahan'].str.re
place('karangpilang', 'karang pilang')

list_kecamatan = df_kelurahan['kecamatan'].tolist()
list_kelurahan = df_kelurahan['kelurahan'].tolist()

########## PREPROCESS DUPLICATE NAME #########
df_toko['name'] = df_toko['name'].str.lower()
df_toko['name'] = df_toko['name'].astype(str)
df_toko = df_toko[~df_toko.name.str.contains('hotel')]

# df_toko = df_toko.drop_duplicates('name')
```

Pra-proses data Google My Business

```
df_toko = df_toko.drop_duplicates(subset=['name', 'website',
 'rating', 'numOf_google_review', 'price', 'category', 'addr
ess', 'phone', 'instagram'], keep='first')


############## PREPROCESS CATEGORY ###########
print('PREPROCESS CATEGORY')
df_toko['category'] = df_toko['category'].str.lower()
df_toko['category'] = df_toko['category'].astype(str)

df_toko_uncat = df_toko[~df_toko.category.str.contains('|'.j
oin(list_category))]
df_toko = df_toko[df_toko.category.str.contains('|'.join(lis
t_category))]

list_filter_cat = ['restaurant']
df_toko = df_toko[~df_toko.category.str.contains('|'.join(li
st_filter_cat))]

list_cat_toko = df_toko.category.unique()
df_cat_toko = pd.DataFrame({'category':list_cat_toko})

list_cat_toko_uncat = df_toko_uncat.category.unique()
df_cat_toko_uncat = pd.DataFrame({'category':list_cat_toko_u
ncat})

############## PREPROCESS ADDRESS ############
print('PREPROCESS ADDRESS')
df_toko['address'] = df_toko['address'].str.lower()
df_toko['address'] = df_toko['address'].astype(str)
df_toko['address'] = df_toko['address'].str.replace('sby,',
'surabaya,')

list_filter_city = ['surabaya']
df_toko_notSurabaya = df_toko[~df_toko.address.str.contains(
'|'.join(list_filter_city))]
df_toko = df_toko[df_toko.address.str.contains('|'.join(list
_filter_city))]
list_filter_area = ['sidoarjo','jakarta','blora','bongso wet
an','pasuruan','buduran']
df_toko = df_toko[~df_toko.address.str.contains('|'.join(lis
t_filter_area))]

df_toko['address'] = df_toko.address.str.replace('perak bar.
', 'perak barat')
df_toko['address'] = df_toko.address.str.replace('krembangan
 sel.', 'krembangan selatan')
```

Pra-proses data Google My Business

```
df_toko['address'] = df_toko.address.str.replace('kjw', 'kej
awan')
df_toko['address'] = df_toko.address.str.replace('gn. anyar'
, 'gunung anyar')
df_toko['address'] = df_toko.address.str.replace('perak tim.
', 'perak timur')
df_toko['address'] = df_toko.address.str.replace('perak bar.
', 'perak barat')
df_toko['address'] = df_toko.address.str.replace('ngenden ja
ngkungan', 'nginden jangkungan')
df_toko['address'] = df_toko.address.str.replace('south krem
bangan', 'krembangan selatan')
df_toko['address'] = df_toko.address.str.replace('asem rowo'
, 'asemrowo')
df_toko['address'] = df_toko.address.str.replace('central ru
ngkut', 'rungkut tengah')
df_toko['address'] = df_toko.address.str.replace('north pera
k', 'perak utara')
df_toko['address'] = df_toko.address.str.replace('gn. sari',
 'gunung sari')
df_toko['address'] = df_toko.address.str.replace('jemur sari
', 'jemur wonosari')
df_toko['address'] = df_toko.address.str.replace('lakar sant
ri', 'lakarsantri')
df_toko['address'] = df_toko.address.str.replace('north krem
bangan', 'krembangan utara')
df_toko['address'] = df_toko.address.str.replace('west perak
', 'perak barat')
df_toko['address'] = df_toko.address.str.replace('east perak
', 'perak timur')
df_toko['address'] = df_toko.address.str.replace('tenggelis'
, 'tenggilis')
df_toko['address'] = df_toko.address.str.replace('kemenangan
 utara', 'krembangan utara')
df_toko['address'] = df_toko.address.str.replace('pakuan tra
de center', 'pakuwon trade center')
df_toko['address'] = df_toko.address.str.replace('penjaringa
n tim.', 'penjaringan timur')
df_toko['address'] = df_toko.address.str.replace('kalirungku
t', 'kali rungkut')
df_toko['address'] = df_toko.address.str.replace('monokremba
ngan', 'morokrembangan')
df_toko['address'] = df_toko.address.str.replace('rugkut', '
rungkut')

df_key_address = pd.read_excel(r'C:\Users\ALDYSYAH\OneDrive
- Institut Teknologi Sepuluh Nopember\Semester 8\TA\scraper\
file\key_address.xlsx')
list_key = df_key_address['key'].tolist()
```

Pra-proses data Google My Business

```
list_kecamatan_key = df_key_address['kecamatan'].tolist()
list_kelurahan_key = df_key_address['kelurahan'].tolist()

df_toko = df_toko.reset_index(drop=True)

list_address = df_toko['address'].tolist()
list_kecamatan_toko = []
list_kelurahan_toko = []
for address in list_address:
    index=0
    for kelurahan in list_kelurahan:
        if kelurahan in address:
            list_kecamatan_toko.append(list_kecamatan[index]
)
            list_kelurahan_toko.append(kelurahan)
            break
        index+=1
        if index == len(list_kelurahan):
            index_key=0
            for key in list_key:
                if key in address:
                    list_kecamatan_toko.append(list_kecamata
n_key[index_key])
                    list_kelurahan_toko.append(list_keluraha
n_key[index_key])
                    break
                index_key+=1
                if index_key == len(list_key):
                    list_kecamatan_toko.append(None)
                    list_kelurahan_toko.append(None)
df_toko['kecamatan'] = list_kecamatan_toko
df_toko['kelurahan'] = list_kelurahan_toko



######## PREPROCESS NUMOF_GOOGLE_REVIEW ####################
print('PREPROCESS NUMOF_GOOGLE_REVIEW')
df_toko['numOf_google_review'] = df_toko['numOf_google_revie
w'].str.replace('Google reviews', '')
df_toko['numOf_google_review'] = df_toko['numOf_google_revie
w'].str.replace('Google review', '')
df_toko['numOf_google_review'] = df_toko['numOf_google_revie
w'].str.replace('.', '')
df_toko['numOf_google_review'] = df_toko['numOf_google_revie
w'].str.replace(',', '')
df_toko['numOf_google_review'] = df_toko['numOf_google_revie
w'].astype(float)
```

Pra-proses data Google My Business

```
######### PREPROCESS OPERATIONAL_HOURS ####################
print('PREPROCESS OPERATIONAL_HOURS')
df_toko['operational_hours'] = df_toko['operational_hours'].
astype(str)

list_days = ['Sunday','Monday','Tuesday','Wednesday','Thursd
ay','Friday','Saturday']
df_toko_hours = df_toko['operational_hours']

for day in list_days:
    string = day+' '
    df_toko_hours = df_toko_hours.str.replace(string,'')

df_toko_hours = df_toko_hours.str.replace(',','')
df_toko_hours = df_toko_hours.str.replace('Open 24 hours','0
-24')
df_toko_hours = df_toko_hours.str.replace('Closed','0-0')

df_toko_hours = df_toko_hours.str.split('[\n| |-
]', expand=True).add_prefix('hours_')
df_toko_hours.fillna(value=0,inplace=True)
df_toko_hours = df_toko_hours.replace('nan',0, regex=True)
df_toko_hours = df_toko_hours.astype(str)

#SET AM OR PM
for row in range(0,len(df_toko_hours.index)):
    for col in range(0,len(df_toko_hours.columns)-1):
        if not df_toko_hours.iloc[row,col] == '24':
            if not df_toko_hours.iloc[row,col] == '0':
                if not any(value in df_toko_hours.iloc[row,c
ol] for value in ('AM','PM')):
                    if 'AM' in df_toko_hours.iloc[row,col+1]
:
                        df_toko_hours.iloc[row,col] = df_tok
o_hours.iloc[row,col]+'AM'
                    elif 'PM' in df_toko_hours.iloc[row,col+
1]:
                        df_toko_hours.iloc[row,col] = df_tok
o_hours.iloc[row,col]+'PM'
#CHANGE TO 24 HOURS FORMAT
for row in range(0,len(df_toko_hours.index)):
    for col in range(0,len(df_toko_hours.columns)):
        if 'AM' in df_toko_hours.iloc[row,col]:
            df_toko_hours.iloc[row,col] = df_toko_hours.iloc
[row,col].replace('AM','')
            if '12' in df_toko_hours.iloc[row,col]:
```

Pra-proses data Google My Business

```
                df_toko_hours.iloc[row,col] = df_toko_hours.
iloc[row,col].replace('12','0')
        elif 'PM' in df_toko_hours.iloc[row,col]:
            df_toko_hours.iloc[row,col] = df_toko_hours.iloc
[row,col].replace('PM','')
            if not '12' in df_toko_hours.iloc[row,col]:
                if ':' in df_toko_hours.iloc[row,col]:
                    df_toko_hours.iloc[row,col] = str(int(df
_toko_hours.iloc[row,col][0:df_toko_hours.iloc[row,col].find
(':')])+12)+ df_toko_hours.iloc[row,col][df_toko_hours.iloc[
row,col].find(':'):len(df_toko_hours.iloc[row,col])]
                else:
                    df_toko_hours.iloc[row,col] = str(int(df
_toko_hours.iloc[row,col])+12)
#MEASURE DURATION IN TIMEDELTA
list_duration = []
for row in range(0,len(df_toko_hours.index)):
    print(row)
    duration = timedelta(0)
    for col in range(0,len(df_toko_hours.columns)-1,2):
        if df_toko_hours.iloc[row,col+1] == '24':
            delta = int(df_toko_hours.iloc[row,col])
            delta = pd.Timedelta(delta,unit='h')
        else:
            try:
                df_toko_hours.iloc[row,col] = datetime.strpt
ime(df_toko_hours.iloc[row,col], '%H')
            except:
                df_toko_hours.iloc[row,col] = datetime.strpt
ime(df_toko_hours.iloc[row,col], '%H:%M')
            try:
                df_toko_hours.iloc[row,col+1] = datetime.str
ptime(df_toko_hours.iloc[row,col+1], '%H')
            except:
                df_toko_hours.iloc[row,col+1] = datetime.str
ptime(df_toko_hours.iloc[row,col+1], '%H:%M')
            delta = df_toko_hours.iloc[row,col+1] - df_toko_
hours.iloc[row,col]
        if '-1 day' in str(delta):
            delta = str(delta)[str(delta).find(':')-
2:len(str(delta))]
            delta = datetime.strptime(delta, '%H:%M:%S')-
datetime.strptime('0', '%H')
        duration = duration + delta
    list_duration.append(duration)
#CONVERT TOTAL DURATION FROM TIMEDELTA TO HOURS
list_duration_hours = []
for duration in list_duration:
```

Pra-proses data Google My Business

```
    list_duration_hours.append(duration.days*24 + duration.s
econds/3600)
#SAVE DURATION IN HOURS INTO DATAFRAME
df_toko['operational_hours_duration'] = list_duration_hours



# type(df_toko.loc[0,'phone'])
############# PREPROCESS PHONE ###############
df_toko['phone'] = df_toko['phone'].astype(str)

df_toko['phone'] = df_toko.phone.str.replace('\+62', '0')
df_toko['phone'] = df_toko.phone.str.replace('-', '')
df_toko['phone'] = df_toko.phone.str.replace(' ', '')
df_toko['phone'] = df_toko.phone.str.replace('nan', 'phone n
ot found')
# df_toko.phone.replace(np.nan, 'no phone information', rege
x=True, inplace=True)



######## PREPROCESS WEBSITE,INSTAGRAM AND PRICE ############
###
print('PREPROCESS WEBSITE,INSTAGRAM,SHOPEE,TOKOPEDIA AND PRI
CE')
df_toko.website.replace(np.nan, 'website not found', regex=T
rue, inplace=True)

list_website=df_toko['website'].tolist()

list_tokopedia=[]
list_shopee=[]
list_instagram=[]
for index,website in enumerate(list_website):
    if 'tokopedia' in website:
        list_tokopedia.append(website)
        list_shopee.append('shopee not found')
        if not str(df_toko.loc[index,'instagram']) == 'nan':
            list_instagram.append(df_toko.loc[index,'instagr
am'])
        else:
            list_instagram.append('instagram not found')
        df_toko.loc[index,'website'] = 'website not found'
    elif 'shopee' in website:
        list_tokopedia.append('tokopedia not found')
        list_shopee.append(website)
        if not str(df_toko.loc[index,'instagram']) == 'nan':
            list_instagram.append(df_toko.loc[index,'instagr
am'])
        else:
```

Pra-proses data Google My Business

```
            list_instagram.append('instagram not found')
        df_toko.loc[index,'website'] = 'website not found'
    elif 'instagram' in website:
        list_tokopedia.append('tokopedia not found')
        list_shopee.append('shopee not found')
        list_instagram.append(website)
        df_toko.loc[index,'website'] = 'website not found'
    else:
        list_tokopedia.append('tokopedia not found')
        list_shopee.append('shopee not found')
        if not str(df_toko.loc[index,'instagram']) == 'nan':
            list_instagram.append(df_toko.loc[index,'instagr
am'])
        else:
            list_instagram.append('instagram not found')

df_toko['tokopedia'] = list_tokopedia
# [df_toko.website.str.contains('instagram')]
df_toko['shopee'] = list_shopee
# [df_toko.website.str.contains('tokopedia')]
df_toko['instagram'] = list_instagram
# df_toko['website'][df_toko.website.str.contains('shopee')]

df_toko.tokopedia.replace('http://tokopedia', 'https://www.t
okopedia', regex=True, inplace=True)
df_toko.tokopedia.replace('http://www.tokopedia', 'https://w
ww.tokopedia', regex=True, inplace=True)
df_toko.tokopedia.replace('https://m.tokopedia', 'https://ww
w.tokopedia', regex=True, inplace=True)
df_toko.tokopedia.replace('https://tokopedia.link/KvNs2ABXw1
', 'https://www.tokopedia.com/tokoenom', inplace=True)
df_toko.tokopedia.replace('https://tokopedia.link/Nje5v8L4v4
', 'https://www.tokopedia.com/nysiemsby', inplace=True)
df_toko.tokopedia.replace('https://tokopedia.link/Xriy2sdfxQ
', 'https://www.tokopedia.com/cafepulsa', inplace=True)
df_toko.tokopedia.replace('https://tokopedia.link/dRDYmYVZ3Y
', 'https://www.tokopedia.com/timotius7', inplace=True)
df_toko.tokopedia.replace('https://www.tokopedia.com/bekuind
onesia?source=universe&st=product', 'https://www.tokopedia.c
om/bekuindonesia', inplace=True)
df_toko.tokopedia.replace('https://www.tokopedia.com/nutrifi
talami/paket-asi-booster-nutrifit-
surabaya', 'https://www.tokopedia.com/nutrifitalami', inplac
e=True)
df_toko.tokopedia.replace('https://www.tokopedia.com/rumah-
otaji/otaji-oseng-tuna-
asap', 'https://www.tokopedia.com/rumah-
otaji', inplace=True)
```

Pra-proses data Google My Business

```
df_toko.tokopedia.replace('https://www.tokopedia.link/SpA2lg
NpX3', 'tokopedia not found', inplace=True)

df_toko['shopee'] = df_toko.shopee.str.lower()
df_toko.shopee.replace('http://', 'https://', regex=True, in
place=True)
df_toko.shopee.replace('www.', '', regex=True, inplace=True)
df_toko.shopee.replace('https://shopee.co.id/delimafrozzen1?
v=b7e&smtt=0.0.3', 'https://shopee.co.id/delimafrozzen1', in
place=True)
df_toko.shopee.replace('https://shopee.co.id/healthypleasure
indonesia?smtt=0.0.9', 'https://shopee.co.id/healthypleasure
indonesia', inplace=True)
df_toko.shopee.replace('https://shopee.co.id/jocelynstar?v=4
38&smtt=0.0.3', 'https://shopee.co.id/jocelynstar', inplace=
True)

#################### PRICE ####################
# df_toko.rating.replace(np.nan, 'tidak memiliki rating', re
gex=True, inplace=True)
df_toko.price.replace(np.nan, 'price not found', regex=True,
 inplace=True)
# df_toko.instagram.replace(np.nan, 'no instagram account',
regex=True, inplace=True)

#################### NO DUPLICATE ###############
df_toko_duplicateName = df_toko[df_toko.duplicated(['name'])
]
df_toko = df_toko.drop_duplicates(subset=['name', 'website',
 'address'], keep='last')

list_cat_toko = df_toko.category.unique()
df_cat_toko = pd.DataFrame({'category':list_cat_toko})

########## REARRANGE COLUMN ORDER ############
list(df_toko.columns.values)
df_toko = df_toko[[
 'name',
 'website',
 'rating',
 'numOf_google_review',
 'price',
 'category',
 'address',
 'kecamatan',
 'kelurahan',
 'operational_hours',
 'operational_hours_duration',
 'phone',
```

Pra-proses data Google My Business
'instagram', 'tokopedia', 'shopee']] ################ EXPORT TO CSV ############## df_toko.to_csv(r'C:\Users\ALDYSYAH\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result\Google Business\2 - googleBusiness_processed.csv')

Tabel A.17 Kode integrasi Tokopedia dengan Instagram

Data Integration Tokopedia dengan Instagram	
Fungsi	Melakukan integrasi pada data Tokopedia dan Instagram
Bahasa	Python

```
from difflib import SequenceMatcher
import pandas as pd
# import numpy as np

df_toko = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result\Tokopedia\ready_similarity_tokopedia_makananminuman.csv',index_col=0)
df_toko['account'] = df_toko['account'].astype(str)
df_toko['description'] = df_toko['description'].astype(str)
df_toko['store'] = df_toko['store'].astype(str)

def similar(a, b):
    return SequenceMatcher(None, a.lower(), b).ratio()

df_toko['similarity'] = df_toko.apply(lambda row: similar(row['store'], row['account']), axis=1)
# df_toko.to_csv('D:/Documents/Kuliah/TA/scraper/Tokopedia/similarity_tokopedia_makananminuman_all.csv')

df_toko['link'] = df_toko['link'].astype(str)
df_toko = df_toko[~df_toko.link.str.contains('explore')]

######### GET MAX SIMILARITY EVERY STORE #########
inds = df_toko.groupby(['store'])['similarity'].transform(max) == df_toko['similarity']
df_max_similarity = df_toko[inds]
df_max_similarity.reset_index(drop=True, inplace=True)

# df_max_similarity.to_csv('similarity_tokped_ig.csv')
############ GET MAX SIMILARITY EVERY STORE #########
```

Data Integration Tokopedia dengan Instagram

```
df_max_similarity = df_max_similarity.drop_duplicates(subset
=['store'], keep='first')
df_max_similarity = df_max_similarity[df_max_similarity['sim
ilarity']>=0.8] #dengan confidence setidaknya 63% (258 benar
 dari 408) data saya benar match (True Positive)

# df_max_similarity['similarity'].mean()

# df_explore = df_max_similarity[df_max_similarity.link.str.
contains('explore')]
df_max_similarity.to_csv(r'C:\Users\ALDYSYAH\OneDrive - Inst
itut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Resul
t\Tokopedia\3 - Tokopedia_Instagram_similarity.csv')
```

Tabel A.18 Kode integrasi Tokopedia dengan Instagram

Data Integration Shopee dengan Instagram	
Fungsi	Melakukan integrasi pada data Shopee dan Instagram
Bahasa	Python

```
from difflib import SequenceMatcher
import pandas as pd
# import numpy as np

df_toko = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Institu
t Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result\S
hopee\ready_similarity_shopee_kesehatan.csv',index_col=0)
df_toko['account'] = df_toko['account'].astype(str)
df_toko['description'] = df_toko['description'].astype(str)
df_toko['store'] = df_toko['store'].astype(str)

def similar(a, b):
    return SequenceMatcher(None, a.lower(), b).ratio()

df_toko['similarity'] = df_toko.apply(lambda row: similar(ro
w['store'], row['account']), axis=1)
# df_toko.to_csv('D:/Documents/Kuliah/TA/scraper/Tokopedia/s
imilarity_tokopedia_makananminuman_all.csv')

df_toko['link'] = df_toko['link'].astype(str)
df_toko = df_toko[~df_toko.link.str.contains('explore')]

############## GET MAX SIMILARITY EVERY STORE #########
inds = df_toko.groupby(['store'])['similarity'].transform(ma
x) == df_toko['similarity']
df_max_similarity = df_toko[inds]
df_max_similarity.reset_index(drop=True, inplace=True)
```

Data Integration Shopee dengan Instagram

```
# df_max_similarity.to_csv('similarity_tokped_ig.csv')

############# GET MAX SIMILARITY EVERY STORE #########

df_max_similarity = df_max_similarity.drop_duplicates(subset
=['store'], keep='first')
df_max_similarity = df_max_similarity[df_max_similarity['sim
ilarity']>=0.8] #dengan confidence setidaknya 63% (258 benar
 dari 408) data saya benar match (True Positive)

# df_max_similarity['similarity'].mean()

# df_explore = df_max_similarity[df_max_similarity.link.str.
contains('explore')]
df_max_similarity.to_csv(r'C:\Users\ALDYSYAH\OneDrive - Inst
itut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Resul
t\Shopee\3 - Shopee_Instagram_similarity.csv')
```

Tabel A.19 Integrasi Tokopedia dan Shopee

Data Integration Tokopedia dan Shopee	
Fungsi	Melakukan integrasi pada data *e-commerce*
Bahasa	Python

```
import pandas as pd
import numpy as np
from difflib import SequenceMatcher
# from nltk import ngrams
import math
# from statistics import mean

df_tokped = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\4 - Tokopedia_processed.csv',index_col=0)
df_shopee = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Shopee\4 - Shopee_Processed.csv',index_col=0)

df_ig_tokped = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Res
ult\Tokopedia\3 - Tokopedia_Instagram_similarity.csv',index_
col=0)
df_ig_shopee = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Res
ult\Shopee\3 - Shopee_Instagram_similarity.csv',index_col=0)

del df_ig_tokped['city']
del df_ig_tokped['description']
del df_ig_tokped['link']
```

Data Integration Tokopedia dan Shopee

```
del df_ig_tokped['similarity']

del df_ig_shopee['city']
del df_ig_shopee['description']
del df_ig_shopee['link']
del df_ig_shopee['similarity']

df_ig_tokped.columns = ['name', 'tokped_ig_merged']
df_ig_shopee.columns = ['name', 'shopee_ig_merged']

df_tokped = pd.merge(df_tokped,df_ig_tokped, on='name', how=
'left')
df_shopee = pd.merge(df_shopee,df_ig_shopee, on='name', how=
'left')

df_tokped = df_tokped.reset_index(drop=True)
df_shopee = df_shopee.reset_index(drop=True)

df_tokped.tokped_ig_merged.replace(np.nan,'no instagram',inp
lace=True)
df_shopee.shopee_ig_merged.replace(np.nan,'no instagram',inp
lace=True)

df_tokped.website.replace('https://', '', regex=True, inplac
e=True)
df_tokped.website.replace('http://', '', regex=True, inplace
=True)
df_tokped.website.replace('www.', '', regex=True, inplace=Tr
ue)

df_shopee.website.replace('https://', '', regex=True, inplac
e=True)
df_shopee.website.replace('http://', '', regex=True, inplace
=True)
df_shopee.website.replace('www.', '', regex=True, inplace=Tr
ue)

df_tokped.instagram.replace('https://', '', regex=True, inpl
ace=True)
df_tokped.instagram.replace('http://', '', regex=True, inpla
ce=True)
df_tokped.instagram.replace('www.', '', regex=True, inplace=
True)
df_tokped.instagram.replace('instagram.com', '', regex=True,
 inplace=True)
df_tokped.instagram.replace('/', '', regex=True, inplace=Tru
e)
```

Data Integration Tokopedia dan Shopee

```
df_shopee.instagram.replace('https://', '', regex=True, inpl
ace=True)
df_shopee.instagram.replace('http://', '', regex=True, inpla
ce=True)
df_shopee.instagram.replace('www.', '', regex=True, inplace=
True)
df_shopee.instagram.replace('instagram.com', '', regex=True,
 inplace=True)
df_shopee.instagram.replace('/', '', regex=True, inplace=Tru
e)

df_tokped['name'] = df_tokped['name'].astype(str)
df_shopee['name'] = df_shopee['name'].astype(str)
df_tokped['phone'] = df_tokped['phone'].astype(str)
df_shopee['phone'] = df_shopee['phone'].astype(str)
df_tokped['website'] = df_tokped['website'].astype(str)
df_shopee['website'] = df_shopee['website'].astype(str)
df_tokped['instagram'] = df_tokped['instagram'].astype(str)
df_shopee['instagram'] = df_shopee['phone'].astype(str)

df_tokped['phone'].replace('nan','no phone',inplace=True)
df_shopee['phone'].replace('nan','no phone',inplace=True)

list_link_from_tokped = df_tokped['link'].tolist()
list_website_from_tokped = df_tokped['website'].tolist()
list_phone_from_tokped = df_tokped['phone'].tolist()
list_ig_from_tokped = df_tokped['instagram'].tolist()
list_ig_merged_from_tokped = df_tokped['tokped_ig_merged'].t
olist()
list_name_from_tokped = df_tokped['name'].tolist()

list_link_from_shopee = df_shopee['link'].tolist()
list_website_from_shopee = df_shopee['website'].tolist()
list_phone_from_shopee = df_shopee['phone'].tolist()
list_ig_from_shopee = df_shopee['instagram'].tolist()
list_ig_merged_from_shopee = df_shopee['shopee_ig_merged'].t
olist()
list_name_from_shopee = df_shopee['name'].tolist()

df_tokped['shopee'] = 'match not found'
df_tokped['shopee_similarity'] = 0
df_tokped['shopee_similarity_by'] = 'match not found'
list_shopee_tokped = df_tokped['shopee'].tolist()
list_shopee_similarity = df_tokped['shopee_similarity'].toli
st()
list_shopee_similarity_by = df_tokped['shopee_similarity_by'
].tolist()

def similar(a, b):
```

Data Integration Tokopedia dan Shopee

```
    return SequenceMatcher(None, a.lower(), b.lower()).ratio
()

def intersectionOfWord(a,b):
    intersec = ''
    for char in a:
        if char in b and not char in intersec:
            intersec += char
    return len(intersec)/max(len(a),len(b))

############## SIMILARITY  ################
# df_merge_col = pd.merge(df_gb, df_tokped, left_on='tokoped
ia', right_on='link')

###### TOKOPEDIA ##################
print('checking on tokopedia')

for idx_shopee in range(len(df_shopee)):

    print(idx_shopee)

    phone_check = True
    website_check = True
    ig_check = True
    name_check = True

    #check phone
    if phone_check:
        if not list_phone_from_shopee[idx_shopee] == 'no pho
ne':
            idx_ratio_max = np.nan
            ratio_max = 0
            threshold = 0.96
            for idx_tokped in range(len(df_tokped)):
                similarity = 0
                if not list_phone_from_tokped[idx_tokped] ==
 'no phone':
                    list_phone_test_shopee = list_phone_from
_shopee[idx_shopee].split('\n')
                    list_phone_test_tokped = list_phone_from
_tokped[idx_tokped].split('\n')
                    for idx_split,phone_test_shopee in enume
rate(list_phone_test_shopee):
                        if not phone_test_shopee == 'nan':
                            for idx_split,phone_test_tokped
in enumerate(list_phone_test_tokped):
                                if not phone_test_tokped ==
'nan':
```

Data Integration Tokopedia dan Shopee

```
                                    similarity = similar(pho
ne_test_shopee,phone_test_tokped)
                                    if similarity > ratio_ma
x:
                                        idx_ratio_max = idx_
tokped
                                        ratio_max = similari
ty
            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
                    list_shopee_tokped[idx_ratio_max] = list
_link_from_shopee[idx_shopee]
                    list_shopee_similarity[idx_ratio_max] =
ratio_max
                    list_shopee_similarity_by[idx_ratio_max]
 = 'phone'
                    website_check = False
                    ig_check = False
                    name_check = False

    #check website
    if website_check:
        if not list_website_from_shopee[idx_shopee] == 'no w
ebsite':
            ratio_max = 0
            threshold = 0.86
            idx_ratio_max = np.nan
            for idx_tokped in range(len(df_tokped)):
                similarity = 0
                if list_shopee_tokped[idx_tokped] == 'match
not found':
                    if not list_website_from_tokped[idx_tokp
ed] == 'no website':
                        similarity = similar(list_website_fr
om_shopee[idx_shopee],list_website_from_tokped[idx_tokped])
                        if similarity > ratio_max:
                            idx_ratio_max = idx_tokped
                            ratio_max = similarity
            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
                    list_shopee_tokped[idx_ratio_max] = list
_link_from_shopee[idx_shopee]
                    list_shopee_similarity[idx_ratio_max] =
ratio_max
                    list_shopee_similarity_by[idx_ratio_max]
 = 'website'
                    ig_check = False
                    name_check = False
```

Data Integration Tokopedia dan Shopee

```
    #check ig
    if ig_check:
        idx_ratio_max = np.nan
        ratio_max = 0
        threshold = 0.83
        if not list_ig_from_shopee[idx_shopee] == 'no instag
ram':
            for idx_tokped in range(len(df_tokped)):
                similarity = 0
                if list_shopee_tokped[idx_tokped] == 'match
not found':
                    if not list_ig_from_tokped[idx_tokped] =
= 'no instagram':
                        list_ig_test_shopee = list_ig_from_s
hopee[idx_shopee].split('\n')
                        list_ig_test_tokped = list_ig_from_t
okped[idx_tokped].split('\n')
                        for idx_split,ig_test_shopee in enum
erate(list_ig_test_shopee):
                            if not ig_test_shopee == 'nan':
                                for idx_split,ig_test_tokped
 in enumerate(list_ig_test_tokped):
                                    if not ig_test_tokped ==
 'nan':
                                        similarity = similar
(ig_test_shopee,ig_test_tokped)
                                        if similarity > rati
o_max:
                                            idx_ratio_max =
idx_tokped
                                            ratio_max = simi
larity
                    if not list_ig_merged_from_tokped[idx_to
kped] == 'no instagram':
                        list_ig_test_shopee = list_ig_from_s
hopee[idx_shopee].split('\n')
                        list_ig_test_tokped = list_ig_merged
_from_tokped[idx_tokped].split('\n')
                        for idx_split,ig_test_shopee in enum
erate(list_ig_test_shopee):
                            if not ig_test_shopee == 'nan':
                                for idx_split,ig_test_tokped
 in enumerate(list_ig_test_tokped):
                                    if not ig_test_tokped ==
 'nan':
                                        similarity = similar
(ig_test_shopee,ig_test_tokped)
                                        if similarity > rati
o_max:
```

Data Integration Tokopedia dan Shopee

```
                                            idx_ratio_max =
idx_tokped
                                            ratio_max = simi
larity
        if not list_ig_merged_from_shopee[idx_shopee] == 'no
 instagram':
            for idx_tokped in range(len(df_tokped)):
                similarity = 0
                if list_shopee_tokped[idx_tokped] == 'match
not found':
                    if not list_ig_from_tokped[idx_tokped] =
= 'no instagram':
                        list_ig_test_shopee = list_ig_merged
_from_shopee[idx_shopee].split('\n')
                        list_ig_test_tokped = list_ig_from_t
okped[idx_tokped].split('\n')
                        for idx_split,ig_test_shopee in enum
erate(list_ig_test_shopee):
                            if not ig_test_shopee == 'nan':
                                for idx_split,ig_test_tokped
 in enumerate(list_ig_test_tokped):
                                    if not ig_test_tokped ==
 'nan':
                                        similarity = similar
(ig_test_shopee,ig_test_tokped)
                                        if similarity > rati
o_max:
                                            idx_ratio_max =
idx_tokped
                                            ratio_max = simi
larity
                    if not list_ig_merged_from_tokped[idx_to
kped] == 'no instagram':
                        list_ig_test_shopee = list_ig_merged
_from_shopee[idx_shopee].split('\n')
                        list_ig_test_tokped = list_ig_merged
_from_tokped[idx_tokped].split('\n')
                        for idx_split,ig_test_shopee in enum
erate(list_ig_test_shopee):
                            if not ig_test_shopee == 'nan':
                                for idx_split,ig_test_tokped
 in enumerate(list_ig_test_tokped):
                                    if not ig_test_tokped ==
 'nan':
                                        similarity = similar
(ig_test_shopee,ig_test_tokped)
                                        if similarity > rati
o_max:
```

Data Integration Tokopedia dan Shopee

```
                                                idx_ratio_max =
idx_tokped
                                                ratio_max = simi
larity
        if not math.isnan(idx_ratio_max):
            if ratio_max >= threshold:
                list_shopee_tokped[idx_ratio_max] = list_lin
k_from_shopee[idx_shopee]
                list_shopee_similarity[idx_ratio_max] = rati
o_max
                list_shopee_similarity_by[idx_ratio_max] = '
instagram'
                name_check = False

    #check name
    if name_check:
        idx_ratio_max = np.nan
        ratio_max = 0
        threshold = 0.5
        for idx_tokped in range(len(df_tokped)):
            if list_shopee_tokped[idx_tokped] == 'match not
found':
                # maxthreegrams_sim = maxthreegrams_similari
ty(list_name_from_tokped[idx_tokped],list_name_from_gb[idx_g
b])
                # intersection_sim = intersectionOfWord(list
_name_from_tokped[idx_tokped],list_name_from_gb[idx_gb])
                # if maxthreegrams_sim >= threshold or inter
section_sim >= threshold :
                #     avg_sim = (maxthreegrams_sim+intersect
ion_sim)/2
                similarity = similar(list_name_from_shopee[i
dx_shopee],list_name_from_tokped[idx_tokped])
                if similarity > ratio_max:
                    idx_ratio_max = idx_tokped
                    ratio_max = similarity
        if not math.isnan(idx_ratio_max):
            if ratio_max >= threshold:
                list_shopee_tokped[idx_ratio_max] = list_lin
k_from_shopee[idx_shopee]
                list_shopee_similarity[idx_ratio_max] = rati
o_max
                list_shopee_similarity_by[idx_ratio_max] = '
name'

df_tokped['shopee'] = list_shopee_tokped
df_tokped['shopee_similarity'] = list_shopee_similarity
df_tokped['shopee_similarity_by'] = list_shopee_similarity_b
y
```

Data Integration Tokopedia dan Shopee

```
# df_backup = df_tokped

# for idx_gb in range(len(df_backup)):
#     print(idx_gb)
#     if df_backup.loc[idx_gb,'tokopedia_similarity'] < 0.9
and df_backup.loc[idx_gb,'tokopedia_similarity_by'] == 'name
':
#         df_backup.loc[idx_gb,'tokopedia_similarity'] = 0
#         df_backup.loc[idx_gb,'tokopedia_similarity_by'] =
'match not found'
#     if df_backup.loc[idx_gb,'shopee_similarity'] < 0.9 and
 df_backup.loc[idx_gb,'shopee_similarity_by'] == 'name':
#         df_backup.loc[idx_gb,'shopee_similarity'] = 0
#         df_backup.loc[idx_gb,'shopee_similarity_by'] = 'ma
tch not found'

df_tokped.to_csv(r'C:\Users\ALDYSYAH\OneDrive - Institut Tek
nologi Sepuluh Nopember\Semester 8\TA\Scraping\Result\integr
ated_tokped_shopee.csv',index=False)
```

Tabel A.20 Kode *Data Integration* Google My Business dengan *e-Commerce*

Data Integration Google My Business dengan *e-Commerce*	
Fungsi	Melakukan integrasi pada data Google My Business dengan Tokopedia dan Shopee
Bahasa	Python

```
import pandas as pd
import numpy as np
from difflib import SequenceMatcher
# from nltk import ngrams
import math
# from statistics import mean

df_gb = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Institut
Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result\Goo
gle Business\2 - googleBusiness_processed.csv',index_col=0)
df_tokped = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Tokopedia\4 - Tokopedia_processed.csv',index_col=0)
df_shopee = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - Insti
tut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Result
\Shopee\4 - Shopee_Processed.csv',index_col=0)

df_ig_tokped = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Res
```

Data Integration Google My Business dengan *e-Commerce*

```
ult\Tokopedia\3 - Tokopedia_Instagram_similarity.csv',index_
col=0)
df_ig_shopee = pd.read_csv(r'C:\Users\ALDYSYAH\OneDrive - In
stitut Teknologi Sepuluh Nopember\Semester 8\TA\Scraping\Res
ult\Shopee\3 - Shopee_Instagram_similarity.csv',index_col=0)

del df_ig_tokped['city']
del df_ig_tokped['description']
del df_ig_tokped['link']
del df_ig_tokped['similarity']

del df_ig_shopee['city']
del df_ig_shopee['description']
del df_ig_shopee['link']
del df_ig_shopee['similarity']

df_ig_tokped.columns = ['name', 'tokped_ig_merged']
df_ig_shopee.columns = ['name', 'shopee_ig_merged']

df_tokped = pd.merge(df_tokped,df_ig_tokped, on='name', how=
'left')
df_shopee = pd.merge(df_shopee,df_ig_shopee, on='name', how=
'left')

df_gb = df_gb.reset_index(drop=True)
df_tokped = df_tokped.reset_index(drop=True)
df_shopee = df_shopee.reset_index(drop=True)

df_tokped.tokped_ig_merged.replace(np.nan,'no instagram',inp
lace=True)
df_shopee.shopee_ig_merged.replace(np.nan,'no instagram',inp
lace=True)

df_gb.website.replace('https://', '', regex=True, inplace=Tr
ue)
df_gb.website.replace('http://', '', regex=True, inplace=Tru
e)
df_gb.website.replace('www.', '', regex=True, inplace=True)

df_tokped.website.replace('https://', '', regex=True, inplac
e=True)
df_tokped.website.replace('http://', '', regex=True, inplace
=True)
df_tokped.website.replace('www.', '', regex=True, inplace=Tr
ue)

df_shopee.website.replace('https://', '', regex=True, inplac
e=True)
```

Data Integration Google My Business dengan *e-Commerce*

```
df_shopee.website.replace('http://', '', regex=True, inplace
=True)
df_shopee.website.replace('www.', '', regex=True, inplace=Tr
ue)


df_gb.instagram.replace('https://', '', regex=True, inplace=
True)
df_gb.instagram.replace('http://', '', regex=True, inplace=T
rue)
df_gb.instagram.replace('www.', '', regex=True, inplace=True
)
df_gb.instagram.replace('instagram.com', '', regex=True, inp
lace=True)
df_gb.instagram.replace('/', '', regex=True, inplace=True)

df_tokped.instagram.replace('https://', '', regex=True, inpl
ace=True)
df_tokped.instagram.replace('http://', '', regex=True, inpla
ce=True)
df_tokped.instagram.replace('www.', '', regex=True, inplace=
True)
df_tokped.instagram.replace('instagram.com', '', regex=True,
 inplace=True)
df_tokped.instagram.replace('/', '', regex=True, inplace=Tru
e)

df_shopee.instagram.replace('https://', '', regex=True, inpl
ace=True)
df_shopee.instagram.replace('http://', '', regex=True, inpla
ce=True)
df_shopee.instagram.replace('www.', '', regex=True, inplace=
True)
df_shopee.instagram.replace('instagram.com', '', regex=True,
 inplace=True)
df_shopee.instagram.replace('/', '', regex=True, inplace=Tru
e)

df_gb.insert(0, 'index', df_gb.index)

df_gb['tokopedia'] = df_gb['tokopedia'].astype(str)
df_gb['shopee'] = df_gb['shopee'].astype(str)

df_gb['name'] = df_gb['name'].astype(str)
df_tokped['name'] = df_tokped['name'].astype(str)
df_shopee['name'] = df_shopee['name'].astype(str)
df_gb['phone'] = df_gb['phone'].astype(str)
df_tokped['phone'] = df_tokped['phone'].astype(str)
df_shopee['phone'] = df_shopee['phone'].astype(str)
```

Data Integration Google My Business dengan *e-Commerce*

```
df_gb['website'] = df_gb['website'].astype(str)
df_tokped['website'] = df_tokped['website'].astype(str)
df_shopee['website'] = df_shopee['website'].astype(str)
df_gb['instagram'] = df_gb['instagram'].astype(str)
df_tokped['instagram'] = df_tokped['instagram'].astype(str)
df_shopee['instagram'] = df_shopee['phone'].astype(str)

df_tokped['phone'].replace('nan','no phone',inplace=True)
df_shopee['phone'].replace('nan','no phone',inplace=True)



list_tokopedia_gb = df_gb['tokopedia'].tolist()
list_shopee_gb = df_gb['shopee'].tolist()
list_website_from_gb = df_gb['website'].tolist()
list_phone_from_gb = df_gb['phone'].tolist()
list_ig_from_gb = df_gb['instagram'].tolist()
list_name_from_gb = df_gb['name'].tolist()

list_link_from_tokped = df_tokped['link'].tolist()
list_website_from_tokped = df_tokped['website'].tolist()
list_phone_from_tokped = df_tokped['phone'].tolist()
list_ig_from_tokped = df_tokped['instagram'].tolist()
list_ig_merged_from_tokped = df_tokped['tokped_ig_merged'].t
olist()
list_name_from_tokped = df_tokped['name'].tolist()

list_link_from_shopee = df_shopee['link'].tolist()
list_website_from_shopee = df_shopee['website'].tolist()
list_phone_from_shopee = df_shopee['phone'].tolist()
list_ig_from_shopee = df_shopee['instagram'].tolist()
list_ig_merged_from_shopee = df_shopee['shopee_ig_merged'].t
olist()
list_name_from_shopee = df_shopee['name'].tolist()

df_gb['tokopedia_similarity'] = 0
df_gb['tokopedia_similarity_by'] = 'match not found'
list_tokopedia_similarity = df_gb['tokopedia_similarity'].to
list()
list_tokopedia_similarity_by = df_gb['tokopedia_similarity_b
y'].tolist()

df_gb['shopee_similarity'] = 0
df_gb['shopee_similarity_by'] = 'match not found'
list_shopee_similarity = df_gb['shopee_similarity'].tolist()
list_shopee_similarity_by = df_gb['shopee_similarity_by'].to
list()

def similar(a, b):
```

Data Integration Google My Business dengan *e-Commerce*

```
    return SequenceMatcher(None, a.lower(), b.lower()).ratio
()

def intersectionOfWord(a,b):
    intersec = ''
    for char in a:
        if char in b and not char in intersec:
            intersec += char
    return len(intersec)/max(len(a),len(b))

############## SIMILARITY  ################
# df_merge_col = pd.merge(df_gb, df_tokped, left_on='tokoped
ia', right_on='link')

###### TOKOPEDIA ##################
print('checking on tokopedia')

for idx_tokped in range(len(df_tokped)):

    print(idx_tokped)

    phone_check = True
    website_check = True
    ig_check = True
    name_check = True

    #check link
    for idx_gb in range(len(df_gb)):
        if not list_tokopedia_gb[idx_gb] == 'tokopedia not f
ound':
            if list_link_from_tokped[idx_tokped] == list_tok
opedia_gb[idx_gb]:
                list_tokopedia_gb[idx_gb] = list_link_from_t
okped[idx_tokped]
                list_tokopedia_similarity[idx_gb] = 1
                list_tokopedia_similarity_by[idx_gb] = 'web
link contains tokopedia'
                phone_check = False
                website_check = False
                ig_check = False
                name_check = False
                break

    #check phone
    if phone_check:
        if not list_phone_from_tokped[idx_tokped] == 'no pho
ne':
            idx_ratio_max = np.nan
            ratio_max = 0
```

Data Integration Google My Business dengan *e-Commerce*

```
            threshold = 0.96
            for idx_gb in range(len(df_gb)):
                similarity = 0
                if list_tokopedia_gb[idx_gb] == 'tokopedia n
ot found':
                    if not list_phone_from_gb[idx_gb] == 'no
 phone':
                        list_phone_test = list_phone_from_to
kped[idx_tokped].split('\n')
                        for idx_split,phone_test in enumerat
e(list_phone_test):
                            if not phone_test == 'nan':
                                if not list_phone_from_gb[id
x_gb] == 'no phone':
                                    similarity = similar(pho
ne_test,list_phone_from_gb[idx_gb])
                                    if similarity > ratio_ma
x:
                                        idx_ratio_max = idx_
gb
                                        ratio_max = similari
ty
            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
                    list_tokopedia_gb[idx_ratio_max] = list_
link_from_tokped[idx_tokped]
                    list_tokopedia_similarity[idx_ratio_max]
 = ratio_max
                    list_tokopedia_similarity_by[idx_ratio_m
ax] = 'phone'
                    website_check = False
                    ig_check = False
                    name_check = False

    #check website
    if website_check:
        if not list_website_from_tokped[idx_tokped] == 'no w
ebsite':
            ratio_max = 0
            threshold = 0.86
            idx_ratio_max = np.nan
            for idx_gb in range(len(df_gb)):
                similarity = 0
                if list_tokopedia_gb[idx_gb] == 'tokopedia n
ot found':
                    if not list_website_from_gb[idx_gb] == '
no website':
                        similarity = similar(list_website_fr
om_tokped[idx_tokped],list_website_from_gb[idx_gb])
```

Data Integration Google My Business dengan *e-Commerce*

```
                        if similarity > ratio_max:
                            idx_ratio_max = idx_gb
                            ratio_max = similarity
            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
                    list_tokopedia_gb[idx_ratio_max] = list_
link_from_tokped[idx_tokped]
                    list_tokopedia_similarity[idx_ratio_max]
 = ratio_max
                    list_tokopedia_similarity_by[idx_ratio_m
ax] = 'website'
                    ig_check = False
                    name_check = False

    #check ig
    if ig_check:
        if not list_ig_from_tokped[idx_tokped] == 'no instag
ram':
            idx_ratio_max = np.nan
            ratio_max = 0
            threshold = 0.83
            for idx_gb in range(len(df_gb)):
                similarity = 0
                if list_tokopedia_gb[idx_gb] == 'tokopedia n
ot found':
                    if not list_ig_from_gb[idx_gb] == 'insta
gram not found':
                        list_ig_test = list_ig_from_tokped[i
dx_tokped].split('\n')
                        for idx_split,ig_test in enumerate(l
ist_ig_test):
                            if not ig_test == 'nan':
                                similarity = similar(ig_test
,list_ig_from_gb[idx_gb])
                                if similarity > ratio_max:
                                    idx_ratio_max = idx_gb
                                    ratio_max = similarity

            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
                    list_tokopedia_gb[idx_ratio_max] = list_
link_from_tokped[idx_tokped]
                    list_tokopedia_similarity[idx_ratio_max]
 = ratio_max
                    list_tokopedia_similarity_by[idx_ratio_m
ax] = 'instagram'
                    name_check = False
        if name_check:
```

Data Integration Google My Business dengan *e-Commerce*

```
            if not list_ig_merged_from_tokped[idx_tokped] ==
 'no instagram':
                idx_ratio_max = np.nan
                ratio_max = 0
                threshold = 0.83
                for idx_gb in range(len(df_gb)):
                    similarity = 0
                    if list_tokopedia_gb[idx_gb] == 'tokoped
ia not found':
                        if not list_ig_from_gb[idx_gb] == 'i
nstagram not found':
                            similarity = similar(list_ig_mer
ged_from_tokped[idx_tokped],list_ig_from_gb[idx_gb])
                            if similarity > ratio_max:
                                idx_ratio_max = idx_gb
                                ratio_max = similarity

                if not math.isnan(idx_ratio_max):
                    if ratio_max >= threshold:
                        list_tokopedia_gb[idx_ratio_max] = l
ist_link_from_tokped[idx_tokped]
                        list_tokopedia_similarity[idx_ratio_
max] = ratio_max
                        list_tokopedia_similarity_by[idx_rat
io_max] = 'instagram_merged'
                        name_check = False
    #check name
    if name_check:
        idx_ratio_max = np.nan
        ratio_max = 0
        threshold = 0.9
        for idx_gb in range(len(df_gb)):
            if list_tokopedia_gb[idx_gb] == 'tokopedia not f
ound':
                # maxthreegrams_sim = maxthreegrams_similari
ty(list_name_from_tokped[idx_tokped],list_name_from_gb[idx_g
b])
                # intersection_sim = intersectionOfWord(list
_name_from_tokped[idx_tokped],list_name_from_gb[idx_gb])
                # if maxthreegrams_sim >= threshold or inter
section_sim >= threshold :
                #     avg_sim = (maxthreegrams_sim+intersect
ion_sim)/2
                similarity = similar(list_name_from_tokped[i
dx_tokped],list_name_from_gb[idx_gb])
                if similarity > ratio_max:
                    idx_ratio_max = idx_gb
                    ratio_max = similarity
        if not math.isnan(idx_ratio_max):
```

Data Integration Google My Business dengan *e-Commerce*

```
            if ratio_max >= threshold:
                list_tokopedia_gb[idx_ratio_max] = list_link
_from_tokped[idx_tokped]
                list_tokopedia_similarity[idx_ratio_max] = r
atio_max
                list_tokopedia_similarity_by[idx_ratio_max]
= 'name'



###### SHOPEE ##################
print('checking on shopee')

for idx_shopee in range(len(df_shopee)):

    print(idx_shopee)

    phone_check = True
    website_check = True
    ig_check = True
    name_check = True

    #check link
    for idx_gb in range(len(df_gb)):
        if not list_shopee_gb[idx_gb] == 'shopee not found':
            if list_link_from_shopee[idx_shopee] == list_sho
pee_gb[idx_gb]:
                list_shopee_gb[idx_gb] = list_link_from_shop
ee[idx_shopee]
                list_shopee_similarity[idx_gb] = 1
                list_shopee_similarity_by[idx_gb] = 'website
 link contains shopee'
                phone_check = False
                website_check = False
                ig_check = False
                name_check = False
                break

    #check phone
    if phone_check:
        if not list_phone_from_shopee[idx_shopee] == 'no pho
ne':
            idx_ratio_max = np.nan
            ratio_max = 0
            threshold = 0.96
            for idx_gb in range(len(df_gb)):
                similarity = 0
                if list_shopee_gb[idx_gb] =='shopee not foun
d':
```

Data Integration Google My Business dengan *e-Commerce*

```
                    if not list_phone_from_gb[idx_gb] == 'ph
one not found':
                        list_phone_test = list_phone_from_sh
opee[idx_shopee].split('\n')
                        for idx_split,phone_test in enumerat
e(list_phone_test):
                            if not phone_test == 'nan':
                                if not list_phone_from_gb[id
x_gb] == 'no phone':
                                    similarity = similar(pho
ne_test,list_phone_from_gb[idx_gb])
                                    if similarity > ratio_ma
x:
                                        idx_ratio_max = idx_
gb
                                        ratio_max = similari
ty
            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
                    list_shopee_gb[idx_ratio_max] = list_lin
k_from_shopee[idx_shopee]
                    list_shopee_similarity[idx_ratio_max] =
ratio_max
                    list_shopee_similarity_by[idx_ratio_max]
 = 'phone'
                    website_check = False
                    ig_check = False
                    name_check = False

    #check website
    if website_check:
        if not list_website_from_shopee[idx_shopee] == 'no w
ebsite':
            idx_ratio_max = np.nan
            ratio_max = 0
            threshold = 0.86
            for idx_gb in range(len(df_gb)):
                similarity = 0
                if list_shopee_gb[idx_gb] == 'shopee not fou
nd':
                    if not list_website_from_gb[idx_gb] == '
no website':
                        similarity = similar(list_website_fr
om_shopee[idx_shopee],list_website_from_gb[idx_gb])
                        if similarity > ratio_max:
                            idx_ratio_max = idx_gb
                            ratio_max = similarity
            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
```

Data Integration Google My Business dengan *e-Commerce*

```
                    list_shopee_gb[idx_ratio_max] = list_lin
k_from_shopee[idx_shopee]
                    list_shopee_similarity[idx_ratio_max] =
ratio_max
                    list_shopee_similarity_by[idx_ratio_max]
 = 'website'
                    ig_check = False
                    name_check = False

    #check ig
    if ig_check:
        if not list_ig_from_shopee[idx_shopee] == 'no instag
ram':
            idx_ratio_max = np.nan
            ratio_max = 0
            threshold = 0.83
            for idx_gb in range(len(df_gb)):
                similarity = 0
                if list_shopee_gb[idx_gb] == 'shopee not fou
nd':
                    if not list_ig_from_gb[idx_gb] == 'insta
gram not found':
                        list_ig_test = list_ig_from_shopee[i
dx_shopee].split('\n')
                        for idx_split,ig_test in enumerate(l
ist_ig_test):
                            if not ig_test == 'nan':
                                similarity = similar(ig_test
,list_ig_from_gb[idx_gb])
                                if similarity > ratio_max:
                                    idx_ratio_max = idx_gb
                                    ratio_max = similarity
            if not math.isnan(idx_ratio_max):
                if ratio_max >= threshold:
                    list_shopee_gb[idx_ratio_max] = list_lin
k_from_shopee[idx_shopee]
                    list_shopee_similarity[idx_ratio_max] =
ratio_max
                    list_shopee_similarity_by[idx_ratio_max]
 = 'instagram'
                    name_check = False
        if name_check:
            if not list_ig_merged_from_shopee[idx_shopee] ==
 'no instagram':
                idx_ratio_max = np.nan
                ratio_max = 0
                threshold = 0.83
                for idx_gb in range(len(df_gb)):
                    similarity = 0
```

Data Integration Google My Business dengan *e-Commerce*

```
                    if list_shopee_gb[idx_gb] == 'shopee not
 found':
                        if not list_ig_from_gb[idx_gb] == 'i
nstagram not found':
                            similarity = similar(list_ig_mer
ged_from_shopee[idx_shopee],list_ig_from_gb[idx_gb])
                            if similarity > ratio_max:
                                idx_ratio_max = idx_gb
                                ratio_max = similarity
                if not math.isnan(idx_ratio_max):
                    if ratio_max >= threshold:
                        list_shopee_gb[idx_ratio_max] = list
_link_from_shopee[idx_shopee]
                        list_shopee_similarity[idx_ratio_max
] = ratio_max
                        list_shopee_similarity_by[idx_ratio_
max] = 'instagram_merged'
                        name_check = False

    # check name
    if name_check:
        idx_ratio_max = np.nan
        ratio_max = 0
        threshold = 0.9
        for idx_gb in range(len(df_gb)):
            if list_shopee_gb[idx_gb] == 'shopee not found':
                # maxthreegrams_sim = maxthreegrams_similari
ty(list_name_from_shopee[idx_shopee],list_name_from_gb[idx_g
b])
                # intersection_sim = intersectionOfWord(list
_name_from_shopee[idx_shopee],list_name_from_gb[idx_gb])
                # if maxthreegrams_sim >= threshold or inter
section_sim >= threshold :
                    # avg_sim = (maxthreegrams_sim+intersect
ion_sim)/2
                similarity = similar(list_name_from_shopee[i
dx_shopee],list_name_from_gb[idx_gb])
                if similarity > ratio_max:
                    idx_ratio_max = idx_gb
                    ratio_max = similarity
        if not math.isnan(idx_ratio_max):
            if ratio_max >= threshold:
                list_shopee_gb[idx_ratio_max] = list_link_fr
om_shopee[idx_shopee]
                list_shopee_similarity[idx_ratio_max] = rati
o_max
                list_shopee_similarity_by[idx_ratio_max] = '
name'
```

Data Integration Google My Business dengan *e-Commerce*

```
df_gb['tokopedia'] = list_tokopedia_gb
df_gb['tokopedia_similarity'] = list_tokopedia_similarity
df_gb['tokopedia_similarity_by'] = list_tokopedia_similarity
_by

df_gb['shopee'] = list_shopee_gb
df_gb['shopee_similarity'] = list_shopee_similarity
df_gb['shopee_similarity_by'] = list_shopee_similarity_by

df_gb.to_csv(r'C:\Users\ALDYSYAH\OneDrive - Institut Teknolo
gi Sepuluh Nopember\Semester 8\TA\Scraping\Result\integrasi.
csv')
```

Halaman ini sengaja dikosongkan

LAMPIRAN B . DATA

Pada lampiran ini ditunjukkan data hasil pengumpulan data *(scraping)* dan data yang telah dilakukan pra-proses data. Data dilampirkan dalam bentuk URL untuk mempermudah pembacaan data.

Tabel B.1 Data Hasil *Scraping* dan *Pre-Process*

No	Data	Deskripsi	URL
1	Sub kategori Tokopedia	Hasil dari *web scraping* sub kategori pada Tokopedia	*https://intip.in/SubKategoriTokped*
2	URL produk Tokopedia	Hasil dari *web scraping* URL produk pada Tokopedia	*https://intip.in/UrlProdukTokped*
3	URL toko Tokopedia	Hasil dari *pre-processing* URL produk menjadi URL toko pada Tokopedia	*https://intip.in/UrlTokoTokopedia*
4	Informasi toko Tokopedia	Hasil dari *web scraping* informasi toko pada Tokopedia	*https://intip.in/InfoTokoTokopedia*
5	Sub kategori Shopee	Hasil dari *web scraping* sub kategori pada Shopee	*https://intip.in/SubKategoriShopee*
6	URL produk Shopee	Hasil dari *web scraping* URL produk pada Shopee	*https://intip.in/UrlProdukShopee*
7	URL toko Shopee	Hasil dari *web scraping* URL toko pada Shopee	*https://intip.in/UrlTokoShopee*
8	Informasi toko Shopee	Hasil dari *web scraping* informasi toko pada Shopee	*https://intip.in/InfoTokoShopee*
9	Informasi akun Instagram berdasarkan Tokopedia	Hasil dari *web scraping* informasi akun pada Instagram berdasarkan kata	*https://intip.in/AkunIgTokopedia*

No	Data	Deskripsi	URL
		kunci nama toko pada Tokopedia	
10	Informasi akun Instagram berdasarkan Shopee	Hasil dari *web scraping* informasi akun pada Instagram berdasarkan kata kunci nama toko pada Shopee	*https://intip.in/AkunIgShopee*
11	Kata kunci Google My Business	Hasil dari penggabungan sub kategori kesehatan dan kecamatan	*https://intip.in/SubKategoriKecamatanGb*
12	Informasi toko Google My Business	Hasil dari *web scraping* informasi toko pada Google My Business	*https://intip.in/InfoTokoGb*
13	Integrasi Tokopedia dengan Instagram	Hasil *data integration* Tokopedia dengan Instagram	*https://intip.in/IntegrasiTokpedIg*
14	Integrasi Shopee dengan Instagram	Hasil *data integration* Shopee dengan Instagram	*https://intip.in/IntegrasiShopeeIg*
15	Integrasi Tokopedia dan Shopee	Hasil *data integration* Tokopedia dengan Shopee	*https://intip.in/IntegrasiTokpedShopee*
16	Integrasi Google My Business dengan *e-Commerce*	Hasil *data integration* Google My Business dengan Tokopedia dan Shopee	*https://intip.in/IntegrasiAll*

LAMPIRAN C. VISUALISASI

Pada lampiran ini ditunjukkan visualisasi data yang telah dibuat menggunakan Microsoft Power BI. Visualisasi data dilampirkan dalam bentuk *link.*

Tabel C.1 *Link* untuk Melihat Visualisasi Data

***Link* Visualisasi Data**
https://intip.in/VisualisasiTA

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis lahir di Surabaya pada tanggal 4 Januari 1998 dan merupakan anak pertama dari tiga bersaudara pasangan Bapak Muhammad Syaifudin dan Ibu Diana Marthessa. Penulis menempuh Pendidikan formal di SD Muhammadiyan GKB Gresik, SMP Negri 01 Gresik, dan SMA 1 Gresik.

Pada tahun 2016 penulis melanjutkan Pendidikan jenjang sarjana dengan jalur SBMPTN di Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211640000088. Selama masa perkuliahan penulis aktif mengikuti organisasi mahasiswa, seperti menjadi Ketua Himpunan Mahasiswa Sistem Informasi 2018/2019. Penulis juga aktif dalam kegiatan kepanitiaan, kegiatan sosial masyarakat dan kegiatan sosial politik, seperti ISE, ITS Mengajar For Indonesia, Tim Penyusun Materi Musyawarah Besar V ITS

Pada tahun keempat, penulis memilih untuk fokus di bidang *System Enterprise* khususnya *Enterprise Resource Planning* dan Sistem Pendukung Keputusan. Oleh karena itu, penulis terdaftar sebagai mahasiswa tugas akhir di Laboratorium Sistem Enterprise, Departemen Sistem Informasi, ITS. Penulis dapat dihubungi melalui email aldyboting@gmail.com.