# 4. IMPLEMENTASI SISTEM

Pada bab ini, akan dibahas mengenai implementasi dari desain sistem yang telah dibuat pada bab sebelumnya. Implementasi sistem ini mencakup penggunaan perangkat lunak yang digunakan, library yang digunakan, sintaksis kode, proses cleaning & preprocessing data, pengujian model algoritma yang digunakan, dan penerapan pembuatan sistem berupa website. Website ini dapat menerima input file Excel, mengidentifikasi hasil analisis keranjang belanja, dan memberikan wawasan baru seputar penjualan di toko. Tabel 4.1 merupakan daftar implementasi desain ke dalam segmen program.

Tabel 4.1

Implementasi Desain ke Dalam Segmen Program

| **Gambar Bab 3** | **Segmen Program** | **Keterangan** |
|---|---|---|
| 3.1, | 4.16, 4.44, 4.45, 4.46 | Integrasi Flask dan Website |
| 3.2, 3.10, 3.11 | 4.18, 4.19, 4.20, 4.21 | Registrasi dan Login Akun |
| 3.3 , 3.12 | 4.22, 4.23, 4.24, 4.42, 4.43 | Kelola Toko dan Akun |
| 3.4, 3.13 | 4.2, 4.3, 4.4, 4.5, 4.6, 4.25, 4.26, 4.27 | Upload File |
| 3.5, 3.15 | 4.7, 4.8, 4.28, 4.29, 4.30, 4.31 | Analisis MBA |
| 3.6, 3.7, 3.8, 3.16 | 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.32, 4.33, 4.34, 4.35, 4.36, 4.37 | Scraping dan Bundling Produk |
| 3.14 | 4.38, 4.39, 4.40, 4.41 | Fetch Dashboard Data |

## 4.1. Implementasi Perangkat Lunak yang Digunakan

Penerapan data mining pada sistem dalam skripsi ini berbentuk sebuah website yang dibuat dengan menggunakan bahasa pemrograman Python, versi Python 3.12.3. Dalam pembuatan website ini, kami menggunakan library Flask, yang berfungsi sebagai antarmuka untuk aplikasi website analisis. Python juga digunakan dalam proses pengelolaan data, termasuk pembuatan itemset keranjang belanja dan rekomendasi bundling. Dengan menggunakan Python dan Flask, kami dapat mengintegrasikan proses data mining ke dalam sistem website dengan lebih mudah dan efisien.

### 4.2. Program Python

#### 4.2.1. Library yang Digunakan pada Program Python

Visual Studio Code berperan sebagai Integrated Development Environment (IDE) untuk Python yang digunakan untuk mengedit teks pemrograman Python, melakukan pengelolaan data, uji coba algoritma, serta instalasi library. Segmen program ini bertujuan untuk mengimpor library dan melakukan persiapan awal sebelum melakukan pemrosesan produksi hasil analisis keranjang belanja. Berikut adalah penggunaan library pada model:

```
Import pandas as pd
from mlxtend.frequent_patterns import fpgrowth, association_rules
from mlxtend.preprocessing import TransactionEncoder
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.chrome.options import Options
```

Segmen Program 4.1 Instalasi Library pada Program Scraping Python

- Library pandas digunakan untuk melakukan manipulasi dan analisis data dalam format DataFrame, yang sangat berguna untuk memuat dataset dari file Excel dan mengelola dataset yang digunakan untuk identifikasi dan analisis data laporan toko.
- Modul mlxtend.frequent_patterns digunakan untuk menemukan pola-pola item yang sering muncul bersama dalam dataset transaksi (frequent itemsets) dengan algoritma fpgrowth. Fungsi association_rules digunakan untuk menghasilkan aturan asosiasi dari frequent itemsets yang ditemukan, berguna dalam analisis keranjang belanja dan data mining.
- Fungsi TransactionEncoder adalah untuk mengonversi dataset transaksi (misalnya daftar item yang dibeli) menjadi format yang bisa diproses oleh algoritma seperti fpgrowth. Mengubah data transaksi menjadi bentuk yang cocok untuk analisis data mining.

#### 4.2.2. Proses Data File Laporan Pesanan

```
#!C:\Users\liman\AppData\Local\Programs\Python\Python311\python.exe
def process_orders_data(filename):
    # Read the Excel file into a DataFrame
    combined_data = pd.read_excel(filename, sheet_name=1, header=6)
    df = pd.DataFrame(combined_data)
```

```
    columns = [
        "Nomor Invoice",
        "Nama Produk",
        "Tanggal Pembayaran",
        "Status Terakhir",
        "Jumlah Produk Dibeli",
        "Harga Jual (IDR)",
    ]
    df2 = df[columns]
    # Filter out rows where 'Status Terakhir' contains the word 'dibatalkan'
    df_filtered = df2[
        ~df2["Status Terakhir"].str.contains("dibatalkan", case=False,
na=False)
    ]


    df_filtered = df_filtered.copy()


    # Convert 'Tanggal Pembayaran' to datetime format
    df_filtered["Tanggal Pembayaran"] = pd.to_datetime(
        df_filtered["Tanggal Pembayaran"], format="%d-%m-%Y %H:%M:%S",
errors="coerce"
    )


    if df_filtered["Tanggal Pembayaran"].isnull().any():
        print("Conversion failed for some entries in 'Tanggal Pembayaran'.")


    # Reformat the 'Tanggal Pembayaran' to the desired string format
    df_filtered["Tanggal Pembayaran"] = df_filtered["Tanggal
Pembayaran"].dt.strftime(
        "%Y-%m-%d %H:%M:%S"
    )
    return df_filtered
```

Segmen Program 4.2 Fungsi untuk Memproses File Laporan Pesanan

Metode `read_excel` dari library Pandas digunakan untuk memuat dan mengakses file data dalam format Excel yang terletak di alamat yang ditentukan ke dalam sebuah DataFrame agar dapat digunakan untuk analisis keranjang belanja dan pembeli. Fungsi ini dirancang khusus untuk melakukan pengolahan file laporan pesanan toko, di mana langkah pertama adalah melakukan pemfilteran data untuk mengambil data pesanan yang sukses dilakukan. Selanjutnya, beberapa kolom data yang relevan dipilih untuk digunakan dalam analisis keranjang belanja. Fungsi ini akan mengembalikan hasil data yang sudah diproses yang nantinya akan digunakan dalam fungsi lain.

### 4.2.3. Mengupload Hasil Data yang Diproses

```
def upload_orders_data(df, shop_id, db_config):
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()

    insert_query = """
    INSERT INTO orders_report (invoice, nama_produk, tgl_payment, quantity,
gross_revenue, shop_id)
    VALUES (%s, %s, %s, %s, %s, %s)
    """
    try:
        for index, row in df.iterrows():
            cursor.execute(
                insert_query,
                (
                    row["Nomor Invoice"],
                    row["Nama Produk"],
                    row["Tanggal Pembayaran"],
                    row["Jumlah Produk Dibeli"],
                    row["Harga Jual (IDR)"],
                    shop_id,
                ),
            )
```

```
        conn.commit()
        print("Data inserted successfully.")

    except Exception as e:
        print(f"Error occurred: {e}")
        conn.rollback()  # Rollback changes in case of error

    finally:
        cursor.close()
        conn.close()
```

Segmen Program 4.3 Fungsi untuk Mengupload Hasil Data

Dengan memanfaatkan modul mysql.connector, fungsi ini menerima tiga parameter input: hasil data yang sudah difilter, shop_id atau id toko, dan konfigurasi basis data. Selanjutnya, program akan terhubung dengan basis data dan dapat memasukkan satu per satu baris dari data filter ke dalam basis data agar dapat diakses dengan mudah.

#### 4.2.4. Memproses File Laporan Pembeli

```
def upload_buyers_data(filename, shop_id, db_config):
    df = pd.read_excel(filename, engine="openpyxl")
    date_range = df.iloc[
        1, 0
    ]  # Assuming the date range is in the first row, first column
    dates = date_range.split(" - ")
    # Convert the strings to datetime objects
    start_date_obj = datetime.strptime(dates[0], "%d/%m/%Y")
    month = start_date_obj.month
    year = start_date_obj.year

    df2 = pd.read_excel(filename, skiprows=5)
    data_columns = ["Kota", "Pembeli", "Pesanan", "Barang", "Nilai Pesanan"]
    df2 = df2[data_columns]
    df2["Nilai Pesanan"] = df2["Nilai Pesanan"].replace("[Rp,]", "",
regex=True)
```

```
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()


    insert_query = """
    INSERT INTO buyers (city_name, buyers_count, orders_count, products_sold,
total_revenue, shop_id, month, year)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
    """


    for index, row in df2.iterrows():
        cursor.execute(
            insert_query,
            (
                row["Kota"],
                row["Pembeli"],
                row["Pesanan"],
                row["Barang"],
                row["Nilai Pesanan"],
                shop_id,
                month,
                year,
            ),
        )


    conn.commit()
    cursor.close()
    conn.close()
```

Segmen Program 4.4 Fungsi untuk Memproses File Laporan Pembeli

Dengan memanfaatkan modul mysql.connector, fungsi ini menerima tiga parameter input: nama file yang diunggah, shop_id atau id toko, dan konfigurasi basis data. Selanjutnya, program akan terhubung dengan basis data dan dapat memasukkan satu per satu baris dari data filter ke dalam basis data agar dapat diakses dengan mudah.

#### 4.2.5. Memasukkan Riwayat Detail File yang Sudah Diupload

```
def insert_log(old, new, file_type, shop_id, db_config):
    conn = mysql.connector.connect(**db_config)
    cursor = conn.cursor()

    insert_query = """
    INSERT INTO files_log (old_filename, renamed_filename, file_type,
shop_id)
    VALUES (%s, %s, %s, %s)
    """

    cursor.execute(insert_query, (old, new, file_type, shop_id))

    conn.commit()
    cursor.close()
    conn.close()
```

Segmen Program 4.5 Fungsi untuk Menyimpan Riwayat File Teruploa

Fungsi ini bertujuan untuk memasukkan histori file yang sudah diunggah oleh pengguna untuk menghindari adanya input file ganda.

#### 4.2.6. Inisialisasi Pemanggilan Fungsi Program untuk Proses dan Upload Data

```
db_config = {
    "host": "localhost",
    "port": 3306,
    "user": "melissa",
    "password": "1234",
    "database": "dataAnalytics",
}
if __name__ == "__main__":
    if len(sys.argv) > 1:
        value_from_flask = sys.argv[1]
        values = value_from_flask.split(",")
        old_file, new_name, file_type, shop_id = values
        file_path = os.path.join(
```

```
            r"D:\\xampp\\htdocs\\migrate\\app\\uploads\\", new_name
        )

        try:
            if file_type == "order":
                df = process_orders_data(file_path)
                upload_orders_data(df, shop_id, db_config)
            elif file_type == "pembeli":
                upload_buyers_data(file_path, shop_id, db_config)

            insert_log(old_file, new_name, file_type, shop_id, db_config)

        except Exception as e:
            logging.error("An error occurred", exc_info=True)
            print(f"An error occurred: {e}")

    else:
        # Return a zero exit code if everything executed successfully
        sys.exit(0)
```

Segmen Program 4.6 Inisialisasi Pemanggilan Fungsi Program

Pada segmen ini, dideklarasikan konfigurasi basis data yang nantinya akan terhubung dengan program. Jika program menerima argumen input dari suatu fungsi di `app.py`, maka program akan memecah argumen tersebut dan menyimpan datanya sesuai dengan urutan variabel yang ditentukan. Setelah itu, program akan melakukan pengecekan tipe file yang diunggah oleh pengguna untuk menentukan alur berjalannya program.

#### 4.2.7. Generate Table Hasil Data di HTML

```
def generate_html_table(filtered_rules, shop_id):

    html_output = f"<br><table id='mbaResult' class='table table-
striped'><tr><th onclick='sortTable(0)'>Support</th><th
onclick='sortTable(1)'>Lift</th> <th
onclick='sortTable(2)'>Confidence</th><th onclick='sortTable(3)'>Set
Produk</th></tr>"


    for index, row in filtered_rules.iterrows():
```

```
        spt = format_value(row["support"])
        lift1 = format_value(row["lift"])
        itemsets_str = row["itemsets"]
        conf = format_value(row["confidence"])

        html_output +=
f"<tr><td>{spt}</td><td>{lift1}</td><td>{conf}</td><td>{itemsets_str}</td></t
r>"

    html_output += "</table>"
    html_output += f"<br><br><div id='btns'><button class='btn btn-primary'
type='button' onclick='generate({shop_id})'>Generate Bundling
Recommendation</button></div>"

    return html_output.encode("utf-8")
```

Segmen Program 4.7 Generate Table Hasil Data di HTML

Pada fungsi ini, input parameter yang diperlukan adalah hasil data analisis keranjang belanja yang nilai minimum supportnya ditentukan oleh pengguna. Data tersebut kemudian akan ditampilkan kembali di halaman HTML.

#### 4.2.8. Inisialisasi Program Analisis Keranjang Belanja

```
db_config = {
                "host": "localhost",
                "port": 3306,
                "user": "melissa",
                "password": "1234",
                "database": "dataAnalytics",
}
if len(sys.argv) > 1:
    values_from_php = sys.argv[1]
    values = values_from_php.split(",")
    support, start_date, end_date, shop_id = values
    support = float(support)
```

```
    conn = mysql.connector.connect(**db_config)
    cur = conn.cursor()
    sql = "SELECT * FROM orders_report WHERE tgl_payment BETWEEN %s AND %s
AND shop_id = %s"
    start_date = start_date.strip()
    end_date = end_date.strip()
    cur.execute(sql, (start_date, end_date, shop_id))
    rows = cur.fetchall()

    if len(rows) == 0:
        print('<h3 style="color:red">No data found</h3>')
    else:
        df = pd.DataFrame(rows)
        df.columns = [
            "order_id",
            "invoice",
            "nama_produk",
            "quantity",
            "tgl_payment",
            "gross_revenue",
            "shop_id",
        ]

        selected_columns = ["invoice", "nama_produk"]
        combined_data = df[selected_columns]

        grouped_df = (
            combined_data.groupby("invoice")["nama_produk"].apply(list).reset
_index()
        )

        grouped_df.columns = ["invoice", "nama_produk"]
```

```
        dataset = grouped_df.values.tolist()
        dataset_without_invoice = grouped_df["nama_produk"].values.tolist()


        te = TransactionEncoder()
        te_ary =
te.fit(dataset_without_invoice).transform(dataset_without_invoice)
        df_encoded = pd.DataFrame(te_ary, columns=te.columns_)


        frequent_itemsets = fpgrowth(df_encoded, min_support=support,
use_colnames=True)
        rules = association_rules(frequent_itemsets, metric="lift",
min_threshold=0)


        # Filter the rules based on support, lift, and confidence
        filtered_rules = rules[
            (rules['support'] >= support) &
            (rules['lift'] >= 0) &
            (rules['confidence'] >= 0)  # Set your desired confidence
threshold
        ]


        # Create the itemsets column
        filtered_rules["itemsets"] = (
            filtered_rules["antecedents"].apply(lambda x: " ||
".join(list(x)))
            + " => "
            + filtered_rules["consequents"].apply(lambda x: " ||
".join(list(x)))
        )


        print(generate_html_table(filtered_rules, shop_id))


    cur.close()
    conn.close()
```

Segmen Program 4.8 Inisialisasi Program Analisis Keranjang Belanja

Pada segmen ini, program menerima argumen input dari salah satu fungsi yang ada di `app.py` untuk memulai program. Salah satu input yang diberikan oleh pengguna adalah nilai support minimum suatu set produk. Jika hasil support dari suatu set produk lebih rendah daripada nilai support minimum, set produk tersebut akan secara otomatis tereliminasi. Hasil analisis keranjang belanja ini akan dimasukkan langsung ke dalam database dan ditampilkan di HTML dengan menggunakan fungsi `generate_html_table`.

### 4.2.9. Mengambil Rule Bundling MBA dari Database dan Output ke Tabel HTML

```
def printSets(shop_id):

    sql = """

    SELECT id, antecedent, consequent, support, confidence, lift,
date_created

    FROM bundleItems

    JOIN bundles ON bundleItems.bundle_id = bundles.bundle_id

    WHERE bundles.shop_id = %s ORDER BY bundleItems.bundle_id DESC

    """

    conn = mysql.connector.connect(**db_config)

    cursor = conn.cursor()

    cursor.execute(sql, (shop_id,))

    rows = cursor.fetchall()


    html_out = f"<div style='height: 350px; overflow-y: scroll;'><table
class='table table-striped'><tr><th>Set
Produk</th><th>Support</th><th>Confidence</th><th>Lift</th><th>Tanggal
Dibuat</th><th>Detail</th></tr>"


    id_list = []

    for row in rows:

        if f"{row[0]}" not in id_list:

            id_list.append(f"{row[0]}")

            html_out += f"<tr><td>{row[1]} =>
{row[2]}</td><td>{row[3]}</td><td>{row[4]}</td><td>{row[5]}</td><td>{row[6]}<
/td><td><button class='btn btn-primary' type='button'
onclick='getDetail({row[0]})'>Detail</button></td></tr>"
```

```
    html_out += f"</table></div>"
    cursor.close()
    conn.close()
    return html_out.encode("utf-8")

import sys

if len(sys.argv) > 1:
    value_from_php = sys.argv[1]
    shop_id = value_from_php
    printSets(shop_id)
```

Segmen Program 4.9 Mengambil dan Menampilkan Hasil Bundling

Pada segmen ini, program akan melakukan inisialisasi atau pemanggilan fungsi untuk proses scraping produk dari halaman website Tokopedia sesuai dengan query namaProduk. Nantinya hasil scraping produk ini akan ditampilkan di halaman HTML terkait untuk dilakukan bundling per produk.

### 4.2.10. Iterasi Tiap Kata dalam Suatu Keyword Produk

```
def generate_phrases(product_name, start_index=4):
    words = product_name.split()
    phrases = []
    pool_products = []

    # Ensure the start_index is within the bounds of the words list
    if start_index > len(words):
        start_index = len(words)

    # Generate phrases starting from the specified index
    for i in range(start_index, len(words) + 1):
        phrase = ' '.join(words[:i])
        # print(phrase)
        phrases.append(phrase)
```

```
    for phrase in phrases:
        scraped_data = scrape_data(phrase)
        if scraped_data is not None:
            pool_products.extend(scraped_data)
    logging.info(len(pool_products))
    return pool_products
```

Segmen Program 4.10 Iterasi Tiap Kata dalam Suatu Keyword Produk

Fungsi ini dirancang untuk melakukan scraping produk di Tokopedia berdasarkan keyword produk yang diberikan. Keyword produk dapat terdiri dari beberapa kata, dan fungsi akan melakukan iterasi mulai dari indeks keempat untuk scraping produk. Pendekatan ini dilakukan untuk memastikan bahwa hasil scraping produk yang diperoleh lebih dari satu.

### 4.2.11. Pembuatan Kelas Produk untuk Objek Hasil Scraping

```
class Product:
    # Constructor method (__init__)
    def __init__(self, name, price, rating, sold, href, src):
        # Attributes
        self.name = name
        self.price = price
        self.rating = rating
        self.sold = sold
        self.href = href
        self.picSRC = src

    def display_info(self):
        return f"<tr><td><img class='prd-name'
src='{self.picSRC}'></td><td><a href='{self.href}'>{self.name}</a></td><td>Rp
{self.price}</td><td>{self.rating}</td><td>{self.sold}</td><td><input
type='radio' onclick='bindSet(event)' class='options' id='{self.name}'
name='{self.name}' value='{self.price}'></td></tr>"

    def return_raw(self):
        return
f"{self.picSRC},{self.href},{self.name},{self.price},{self.rating},{self.sold
}"
```

```
    def __eq__(self, other):
        if isinstance(other, Product):
            return self.name == other.name and self.href == other.href
        return False

    def __hash__(self):
        return hash((self.name, self.href))
```

Segmen Program 4.11 Pembuatan Kelas Produk untuk Objek Hasil Scraping

Kelas Produk bertujuan untuk menyimpan data hasil scraping produk sebagai objek dengan beberapa atributnya. Objek Produk ini akan difilter untuk mendapatkan 10 besar hasil rekomendasi harga terbaik.

#### 4.2.12. Fungsi Scraping Data sesuai Keyword Produk

```
def scrape_data(prdNm):
    # Set up Chrome options
    options = Options()
    options.add_argument("--enable-javascript")
    # options.add_argument("--headless")
    options.add_argument(
        "user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    )
    options.add_argument("--window-position=1920,1080")

    driver = webdriver.Chrome(options=options)

    varProd_dict = {"st": "", "q": prdNm}
    encoded_data = urlencode(varProd_dict)
    base_url = "https://www.tokopedia.com/search?"
    url = base_url + encoded_data

    try:
```

```
        # Navigate to the search URL
        driver.get(url)
        time.sleep(10)

        # Wait for the dropdown button to be clickable
        dropdown_button = WebDriverWait(driver, 5).until(
            EC.element_to_be_clickable((By.XPATH, "//button[@data-
unify='Select']"))
        )
        dropdown_button.click()

        # Find and click the option for "Ulasan"
        ulasan_option = WebDriverWait(driver, 3).until(
            EC.element_to_be_clickable((By.XPATH, "//button[@data-item-
text='Ulasan']"))
        )
        ulasan_option.click()
        time.sleep(5)

        lazy_elements = WebDriverWait(driver, 8).until(
            EC.presence_of_all_elements_located((By.CLASS_NAME,
"IOLazyloading"))
        )
        last_height = driver.execute_script("return
document.body.scrollHeight")

        for _ in range(10):  # Adjust range for sufficient scrolling
            driver.execute_script("window.scrollBy(0, 500);")
            time.sleep(3)  # Adjust wait time as needed

            # Move to lazy-loaded elements
            actions = ActionChains(driver)
            for element in lazy_elements:
                try:
```

```
                    actions.move_to_element(element).perform()
                    time.sleep(1)  # Adjust wait time as needed
                except Exception as e:
                    break

            # Calculate new scroll height
            new_height = driver.execute_script("return
document.body.scrollHeight")
            if new_height == last_height:
                break
            last_height = new_height

        # Parse the HTML content
        html_content = driver.page_source
        soup = BeautifulSoup(html_content, "html.parser")
        price = soup.select('div.prd_link-product-price[data-
testid="spnSRPProdPrice"]')
        name = soup.select('div.prd_link-product-name[data-
testid="spnSRPProdName"]')
        href = soup.select("a.pcv3__info-content")
        rate = soup.select("span.prd_rating-average-text")
        soldNu = soup.select("span.prd_label-integrity")
        pic_cons = soup.find_all("div", class_="pcv3_img_container")
        products = []

        # Extract product information
        for x, y, z, m, n, o in zip(name, price, rate, soldNu, href,
pic_cons):
            first_three_words = prdNm.lower().split()[:3]
            if all(word in x.text.lower() for word in first_three_words) and
convert_sold(m.text.strip()) > 10:
                img_tag = o.find("img")
                if img_tag:
                    img_url = img_tag["src"]
```

```
                    newProd = Product(
                        x.text.strip(),
                        re.sub(r"\D", "", y.text.strip()),
                        z.text.strip(),
                        convert_sold(m.text.strip()),
                        n["href"],
                        img_url,
                    )
                    if newProd not in products:
                        products.append(newProd)

        return products if products else []

    except Exception as e:
        # print(f"An exception occurred: {e}")
        pass

    driver.quit()
    return []  # Return an empty list if no products are found or an error
occurs
```

Segmen Program 4.12 Fungsi Scraping Data sesuai Keyword Produk

Fungsi ini bertujuan untuk melakukan scraping data menggunakan library Selenium dan modul WebDriver. Pertama-tama, dilakukan inisialisasi WebDriver dan konfigurasi perilaku agar data produk yang terdapat pada halaman HTML Tokopedia dapat ditampilkan dengan baik. Selanjutnya, HTML akan melewati proses parsing dengan Beautiful Soup untuk mengekstrak elemen yang dibutuhkan. Setiap hasil scraping akan dideklarasikan sebagai sebuah objek produk dan akan disimpan dalam sebuah array. Array ini kemudian akan digabungkan dengan hasil scraping lainnya dari keyword produk yang sama.

### 4.2.13. Filter Produk Hasil Scraping

```
def filter_result(products):
    if len(products) > 0:
        total_price = sum(int(product.price) for product in products)
```

```
        average_price = total_price / len(products) * 0.7
    else:
        average_price = 0

    filtered = [product for product in products if int(product.price) >=
average_price]

    sorted_products = sorted(
        filtered, key=lambda x: (x.sold, x.rating, int(x.price)),
reverse=True
    )

    return sorted_products[:10]
```

Segmen Program 4.13 Filter Produk Hasil Scraping

Setelah sebuah keyword produk selesai di-scrapping, array yang berisikan banyak objek produk tersebut akan melewati proses filtrasi data untuk mengambil 10 data terbaik berdasarkan rata-rata harga produk.

#### 4.2.14. Inisialisasi Proses Scraping

```
if (len(sys.argv) > 1):
    try:
        value_from_php = sys.argv[1]
        sorted_prd = ''
        valuess = value_from_php.split(",")
        bundle_id, shop_id = valuess
        conn = mysql.connector.connect(**db_config)
        cur = conn.cursor()
        sql = "SELECT antecedent, consequent FROM bundleitems WHERE id = %s"
        cur.execute(sql, (bundle_id,))
        rows = cur.fetchall()
        for antecedent, consequent in rows:
            antecedent_items = antecedent.split(" => ")
            if " || " in antecedent_items[0]:
                antecedents = antecedent_items[0].split(" || ")
```

```
            for i in antecedents:
                prdArr.append(i)
        else:
            prdArr.append(antecedent_items[0])

        if " || " in consequent:
            consequents = consequent.split(" || ")
            for i in consequents:
                prdArr.append(i)
        else:
            prdArr.append(consequent)

    for i in prdArr:
        a = generate_phrases(i)
        logging.info(len(a))
        combined_results.extend(a)
        html_string = f'''
        <h5>Result for : {i} </h5>
        <table class="table table-bordered" id="{i}">
            <thead>
                <tr>
                    <th>Gambar</th>
                    <th>Nama Produk</th>
                    <th>Harga</th>
                    <th>Rating</th>
                    <th>Terjual</th>
                    <th>Aksi</th>
                </tr>
            </thead>
            <tbody>'''
        x = filter_result(combined_results, i, db_config)
        print(html_string)
```

```
            for i in x:
                print(i.display_info())
            print('</tbody></table><br><br>')
            combined_results.clear()
    except KeyboardInterrupt:
        print("\nProgram interrupted and stopped.")
```

Segmen Program 4.14 Inisialisasi Proses Scraping dan Output Hasil Data

Pada segmen ini, program akan menginisialisasi atau memanggil fungsi untuk melakukan proses scraping produk dari halaman website Tokopedia berdasarkan query namaProduk. Hasil scraping produk ini akan ditampilkan di halaman HTML terkait untuk dilakukan bundling per produk. Segmen ini sempat dapat berjalan dengan baik untuk melakukan scraping produk di Tokopedia, namun karena Tokopedia memang tidak mengijinkan scraping illegal pada wesbitenya maka proses scraping ini akhirnya terpaksa diberhentikan karena Tokopedia mengubah sistem dan struktur websitenya agar tidak bisa discraping lagi sejak tanggal 28 Juni 2024.

### 4.2.15. Mengambil Informasi Produk dari Suatu Itemset dari Database

```
prdArr = []
if (len(sys.argv) > 1):
    try:
        value_from_php = sys.argv[1]
        sorted_prd = ''
        valuess = value_from_php.split(",")
        bundle_id, shop_id = valuess
        conn = mysql.connector.connect(**db_config)
        cur = conn.cursor()
        sql = "SELECT antecedent, consequent FROM bundleitems WHERE id = %s"
        cur.execute(sql, (bundle_id,))
        rows = cur.fetchall()
        for antecedent, consequent in rows:
            antecedent_items = antecedent.split(" => ")
            if " || " in antecedent_items[0]:
```

```
                antecedents = antecedent_items[0].split(" || ")
                for i in antecedents:
                    prdArr.append(i)
            else:
                prdArr.append(antecedent_items[0])


            if " || " in consequent:
                consequents = consequent.split(" || ")
                for i in consequents:
                    prdArr.append(i)
            else:
                prdArr.append(consequent)


        items = []
        for i in prdArr:
            name = i
            sql = "SELECT DISTINCT quantity, gross_revenue FROM orders_report
WHERE nama_produk = %s"
            cur.execute(sql, (i,))
            rows = cur.fetchall()
            name = i


            # Iterating through the fetched rows and printing quantity and
gross revenue
            for row in rows:
                qty, totalprice = row  # Each row is a tuple containing
(quantity, gross_revenue)
                price = int(totalprice/qty)
                items.append(Product(i, price))
                break


        for i in items :
            html_string = f'''
            <h5>Result for : {i.get_name()} </h5>
```

```
            <table class="table table-bordered" id="{i.get_name()}">
                <thead>
                    <tr>
                        <th>Nama Produk</th>
                        <th>Harga</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tbody>
            '''
            print(html_string)
            print(i.display_info())
            print('</tbody></table><br><br>')
    except KeyboardInterrupt:
        print("\nProgram interrupted and stopped.")
```

Segmen Program 4.15 Mengambil dan Menampilkan Hasil Bundling

Pada segmen ini, program akan melakukan pemanggilan nama produk dari suatu itemset sesuai dengan id dari suatu itemset. Kemudian dilakukan query di database berdasarkan nama produk tersebut untuk mendapat informasi harga produk dan kuantitas terjual. Setelah itu program melakukan perhitungan untuk menemukan harga asli sebuah unit produk dan kemudian membuat objek produk berdasarkan nama produk dan harga per unit tersebut. Data lalu ditampilkan dalam bentuk tabel di halaman HTML. Segmen program ini digunakan sebagai substitusi dari program scraping yang tidak lagi dapat digunakan.

### 4.2.16. Output Informasi Produk dari Rule Bundling ke Tabel HTML

```
def output_table(bundle_id, prdNm, db_config):
    conn = mysql.connector.connect(**db_config)
    cur = conn.cursor()
    sql = "SELECT * FROM scraperesult WHERE bundle_id = %s AND identifier =
%s"
    cur.execute(sql, (bundle_id, prdNm,))
    rows = cur.fetchall()
```

```
    html_string = f'''
        <h5>Result for : {prdNm}</h5>
        <table class="table table-bordered" id="{prdNm}">
            <thead>
                <tr>
                    <th>Gambar</th>
                    <th>Nama Produk</th>
                    <th>Harga</th>
                    <th>Rating</th>
                    <th>Terjual</th>
                    <th>Aksi</th>
                </tr>
            </thead>
            <tbody>
    '''


    for row in rows:
        html_string += f'''
            <tr>
                <td><img src='{row[1]}'></td>
                <td><a href="{row[2]}">{row[3]}</a></td>
                <td>{row[4]}</td>
                <td>{row[5]}</td>
                <td>{row[6]}</td>
                <td><input type='radio' class='options' id='{row[0]}'
name='{row[0]}' value='{row[4]}'></td>
            </tr>
        '''


    html_string += '''
            </tbody>
        </table><br><br>
    '''
```

```
print(html_string)
```

Segmen Program 4.16 Output Hasil Data yang Sudah Difilter kedalam Tabel HTML

Pada segmen ini, program Python akan menampilkan data yang sudah difilter kedalam tabel HTML.

## 4.3. Aplikasi Website Analitik

Segmen program ini bertujuan untuk mengimpor library yang dibutuhkan untuk memproses data, menganalisis data, dan membuat antarmuka web menggunakan Flask dalam pembuatan aplikasi website ini. Berikut adalah penggunaan library yang disertakan:

### 4.3.1. Library yang Digunakan

```
import hashlib
import io
import json
import os
import re
import traceback
import logging
from flask import Flask, render_template, request, redirect, url_for, session, jsonify
from flask_mysqldb import MySQL
import MySQLdb.cursors
import subprocess
import datetime
from datetime import datetime
import random
import uuid
import sys
```

Segmen Program 4.17 Import Library untuk aplikasi Flask

- `from flask import Flask, request, render_template`: Mengimpor modul Flask yang diperlukan, termasuk `Flask` untuk inisialisasi aplikasi, `request` untuk menangani permintaan HTTP, dan `render_template` untuk memuat file template HTML.
- Modul MySQL digunakan untuk mengintegrasikan proyek Flask dengan database MySQL
- Modul MySQLdb.cursors dimanfaatkan untuk navigasi dan manipulasi hasil kueri.

- Library selenium digunakan untuk otomatisasi browser web. Webdriver digunakan untuk mengontrol browser secara otomatis, misalnya untuk menguji aplikasi web atau untuk scraping data dari web.

### 4.3.2. Inisialisasi File App.py Aplikasi Flask

```
app = Flask(__name__)
logging.basicConfig(level=logging.INFO)
app.config['MYSQL_HOST'] = 'localhost'  # Cloud SQL Proxy's IP address
app.config['MYSQL_PORT'] = 3306
app.config['MYSQL_USER'] = 'melissa'
app.config['MYSQL_PASSWORD'] = '1234'
app.config['MYSQL_DB'] = 'dataAnalytics'
mysql = MySQL(app)
UPLOAD_FOLDER = r'D:\\xampp\\htdocs\\migrate\\app\\uploads\\'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

Segmen Program 4.18 Inisialisasi Aplikasi Flask

Berikut adalah contoh kode untuk inisialisasi aplikasi Flask dan inisialisasi direktori untuk tempat sementara dari file yang diunggah:

- `app = Flask(__name__)`: Membuat objek aplikasi Flask. Parameter `__name__` digunakan untuk menentukan nama modul saat ini.
- `UPLOAD_FOLDER = 'temp_upload'`: Menentukan nama direktori untuk menyimpan file yang diunggah. Dalam contoh ini, direktori disebut `temp_upload`.
- `app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER`: Mengatur konfigurasi aplikasi Flask untuk menunjukkan direktori tempat sementara dari file yang diunggah.

### 4.3.3. Fungsi dalam Aplikasi Flask

```
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()
```

Segmen Program 4.19 Fungsi Hash Password

Fungsi hash password dibuat untuk mengubah sebuah string password menjadi bentuk yang terenkripsi dan sulit untuk dikembalikan ke bentuk aslinya. Langkah pertama fungsi akan menambahkan data acak (disebut "salt") ke password asli sebelum di-hash. Salt membuat setiap password hash menjadi unik, meskipun password yang diinput sama. Kemudian akan diterapkan

algoritma hashing SHA-256, pada password yang sudah ditambah salt. Algoritma ini menghasilkan nilai hash yang tetap panjangnya tidak tergantung pada panjang sebuah string input asli.

```
# Define handler functions
@app.route('/', methods=['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        email = request.form['email']
        password = request.form['password']


        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM users WHERE user_email = %s', (email,))
        account = cursor.fetchone()


        if account:
            hashed_password = account['user_pass']  # Get hashed password from database
            password_hash = hashlib.sha256(password.encode()).hexdigest()  # Hash the provided password


            if hashed_password == password_hash:  # Compare hashed passwords
                session['loggedin'] = True
                session['id'] = account['account_id']
                session['username'] = account['account_name']
                return redirect(url_for('dashboard'))
            else:
                msg = 'Incorrect password!'
        else:
            msg = 'Email not found!'
```

```
    return render_template('login.html', msg=msg)
```

Segmen Program 4.20 Fungsi Login

Fungsi ini digunakan untuk memproses data formulir saat pengguna mencoba masuk ke dalam sistem website. Menerima parameter input dari sebuah form yang ada di halaman HTML login. Jika autentikasi berhasil, sesi khusus untuk pengguna akan dibuat. Jika pengguna salah memasukkan email atau password, akan muncul peringatan dan pengguna diminta untuk mencoba login kembali.

```
def register():
    msg = ''
    try:
        if request.method == 'POST':
            if 'email' in request.form and 'account_name' in request.form and
'password' in request.form:
                username = request.form['account_name']
                password = request.form['password']
                email = request.form['email']
                cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
                cursor.execute('SELECT * FROM users WHERE user_email = %s',
(email,))
                account = cursor.fetchone()
                if account:
                    msg = 'Account already exists!'
                elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
                    msg = 'Invalid email address!'
                elif not re.match(r'[A-Za-z0-9]+', username):
                    msg = 'Username must contain only characters and
numbers!'
                elif not username or not password or not email:
                    msg = 'Please fill out the form!'
                else:
                    hashed_password = hash_password(password)
```

```
                    cursor.execute('INSERT INTO users (user_email,
account_name, user_pass) VALUES (%s, %s, %s)', (email, username,
hashed_password))
                    mysql.connection.commit()
                    msg = 'You have successfully registered!'
                    return render_template('login.html')
            else:
                msg = 'Please fill out the form!'
    except Exception as e:
        logging.error("Exception occurred: %s", e)
        logging.error(traceback.format_exc())
        msg = 'An error occurred during registration. Please try again.'

    return render_template('register.html', msg=msg)
```

Segmen Program 4.21 Fungsi Register

Fungsi ini digunakan untuk memproses data formulir saat pengguna mencoba mendaftar ke dalam sistem website. Menerima parameter input dari sebuah form yang ada di halaman register. Jika berhasil melakukan pendaftaran maka pengguna akan diarahkan ke halaman login. Jika pengguna memasukkan email yang sudah terdaftar di database maka akan muncul peringatan dan pengguna diminta untuk mencoba registrasi kembali.

```
def forgetPass():
    msg = ''
    try:
        if request.method == 'POST':
            logging.debug("Form Data: %s", request.form)
            if 'userEmail' in request.form and 'newPass' in request.form:
                userEmail = request.form['userEmail']
                newpassword = request.form['newPass']
                cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
                cursor.execute('SELECT * FROM users WHERE user_email = %s',
(userEmail,))
                account = cursor.fetchone()
```

```
                if not re.match(r'[^@]+@[^@]+\.[^@]+', userEmail):
                    msg = 'Invalid email address!'
                elif not userEmail or not newpassword:
                    msg = 'Please fill out the form!'
                else:
                    hashed_password = hash_password(newpassword)
                    cursor.execute('UPDATE users SET user_pass = %s WHERE
user_email = %s', (hashed_password, userEmail))
                    mysql.connection.commit()
                    msg = 'Password changed successfully!'
                    return render_template('login.html', msg=msg)
            else:
                msg = 'Please fill out the form!'
    except Exception as e:
        logging.error("Exception occurred: %s", e)
        logging.error(traceback.format_exc())
        msg = 'An error occurred during password reset. Please try again.'

    return render_template('login.html', msg=msg)
```

Segmen Program 4.22 Fungsi Forget Password

Fungsi ini digunakan untuk memproses data formulir saat pengguna mencoba melakukan perubahan password. Menerima parameter input dari sebuah form yang ada di halaman reset password seperti email dan password baru. Jika berhasil melakukan reset password maka pengguna akan diarahkan ke halaman login. Jika pengguna memasukkan email yang tidak terdaftar di database maka akan muncul peringatan dan pengguna diminta untuk mencoba untuk input kembali form reset password.

```
def callShop():
    if 'id' not in session:
        return redirect(url_for('login'))

    userId = session.get('id')
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
```

```
    cursor.execute('SELECT shop_name, shop_id FROM shop WHERE user_id = %s',
(userId,))
    shops = cursor.fetchall()


    if len(shops) == 0:
        msg = "NO SHOP REGISTERED ! ADD NEW SHOP !"
        return jsonify({'message': msg}), 200  # Return JSON response with
message
    else:
        return jsonify(shops), 200  # Return JSON response with shop data
```

Segmen Program 4.23 Fungsi Call Shop

Fungsi ini digunakan untuk melakukan pengecekan di database apakah suatu pengguna sudah memiliki satu toko terdaftar di database atau belum. Apabila belum ada toko terdaftar maka akan muncul peringatan agar pengguna segera membuat sebuah toko baru. Namun jika terdapat toko terdaftar, maka sistem akan mengembalikan hasil kueri ke halaman website untuk ditampilkan datanya.

```
def addShop():
    if 'id' not in session:
        return redirect(url_for('login'))


    userId = session.get('id')
    if request.method == 'POST' and 'shop_name' in request.form and
'marketplace' in request.form:
        shopName = request.form['shop_name'].upper()
        marketplace = request.form['marketplace']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM shop WHERE shop_name = %s AND user_id =
%s', (shopName, userId))
        shop = cursor.fetchone()
        if shop:
            msg = 'Shop name under this account already exists!'
        elif not shopName or not marketplace :
            msg = 'Please fill out the form!'
```

```
        else:
            cursor.execute('INSERT INTO shop (shop_name, marketplace,
user_id) VALUES (%s, %s, %s)', (shopName, marketplace, userId))
            mysql.connection.commit()
            msg = 'You have successfully added a new shop !'
            return render_template('user.html', msg=msg)
```

Segmen Program 4.24 Fungsi Add Shop

Fungsi ini digunakan untuk mendaftarkan sebuah toko baru dari suatu akun. Jika nama toko sudah terdaftar pada akun yang sama maka penambahan toko dianggap gagal. Apabila penambahan toko berhasil maka data toko baru akan tersimpan di database. Kemudian fungsi akan mengembalikan respon berdasarkan string msg ke halaman HTML.

```
def delShop():
    if 'id' not in session:
        return redirect(url_for('login'))

    tableName = ['bundles', 'orders_report', 'shop_report', 'files_log',
'shop']
    if request.method == 'POST' and 'shop_id' in request.form:
        shopId = request.form['shop_id']
        for table in tableName:
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute(f"SELECT * FROM {table} WHERE shop_id = %s",
(shopId,))
            account = cursor.fetchone()
            if account and table == "bundles":
                cursor.execute(f"SELECT bundle_id FROM {table} WHERE shop_id
= %s", (shopId,))
                bundles_result = cursor.fetchall()
                for bundle in bundles_result:
                    cursor.execute(f"DELETE FROM bundleitems WHERE bundle_id
= %s", (bundle['bundle_id'],))
                    mysql.connection.commit()
            elif account:
```

```
                cursor.execute(f"DELETE FROM {table} WHERE shop_id = %s",
(shopId,))
                mysql.connection.commit()
        return '', 204  # Return a no content response to indicate success


    return '', 400  # Return a bad request response if shop_id is missing or
invalid
```

Segmen Program 4.25 Fungsi Delete Shop

Fungsi ini digunakan untuk menghapus sebuah toko yang sudah didaftarkan oleh pengguna. Semua data dari berbagai tabel database yang berkaitan dengan toko ini akan dihapus satu persatu.

```
def checkFiles():
    if 'id' not in session:
        return redirect(url_for('login'))


    if 'shop_id' in request.form:
        shop_id = request.form['shop_id']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT COUNT(*) AS num_files FROM files_log WHERE
shop_id = %s', (shop_id,))
        result = cursor.fetchone()
        if result['num_files'] == 0:
            return jsonify({'message': 'No files found for this shop.'}), 200
        else:
            return jsonify({'message': 'Files found for this shop.', 'files':
result['num_files']}), 200
```

Segmen Program 4.26 Fungsi Check Files

Fungsi ini digunakan melakukan pengecekan terhadap files dari suatu toko. Apabila belum ada file yang diupload maka fungsi akan mengembalikan respon message ke halaman HTML.

```
def loadLog():
    if 'id' not in session:
```

```
        return redirect(url_for('login'))


    shop_id = request.form['shop_id']
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELECT * FROM files_log WHERE shop_id = %s', (shop_id,))
    files = cursor.fetchall()
    if len(files) == 0:
        return jsonify({'message': 'No files found for this shop.'}), 200
    else:
        return jsonify({'message': 'Files found for this shop.', 'files':
files}), 200
```

Segmen Program 4.27 Fungsi Load Log

Fungsi ini digunakan melakukan pengecekan terhadap files dari suatu toko. Apabila suatu toko sudah mengupload beberapa file, maka daftar file akan dipanggil dari database untuk ditampilkan di tabel HTML. Namun jika belum ada file yang diupload maka fungsi akan mengembalikan respon message ke halaman HTML.

```
import logging
import subprocess


def upload():
    files = request.files.getlist('xlsx_file')
    logging.info(files)
    messages = []


    type_file = request.form['file_type']
    shop_id = request.form['shopName']


    for file in files:
        if file.filename == '':
            messages.append('No selected file')
            continue
```

```
        current_datetime = datetime.now().strftime('%d-%m-%y-%H-%M')
        filename, file_extension = os.path.splitext(file.filename)
        new_filename = f"{type_file}_{current_datetime}{file_extension}"
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], new_filename)
        logging.info(file_path)
        file.save(file_path)

        values_string = ','.join([file.filename, new_filename, type_file,
shop_id])

        python_script_path = r"D:\xampp\htdocs\migrate\app\upload.py"
        py_exec =
r"C:\\Users\\liman\\AppData\\Local\\Programs\\Python\\Python311\\python.exe"

        command = f'"{py_exec}" "{python_script_path}" "{values_string}"'
        subprocess.run(command, shell=True)
        logging.info("run py")

    final_message = ' | '.join(messages)
    return render_template('files.html', xxx=final_message)
```

Segmen Program 4.28 Fungsi Upload

Fungsi ini digunakan untuk menerima dan menangani parameter input dari form HTML berupa sebuah file atau beberapa file sekaligus dan tipe file yang dipilih. File akan disimpan sementara didalam folder uploads dalam direktori yang sama dengan fungsi ini. Kemudian parameter dikirim dengan bentuk argument ke script Python upload.py. Argumen yang diterima akan digunakan oleh upload.py untuk melakukan proses pembacaan dan upload data kedalam database.

```
def generateMBA():
    if request.method == 'POST':
        support = request.form['support']
        start_date = request.form['dates']
```

```
        end_date = request.form['dates2']
        shop_id = request.form['shopName']

        try:
            # Convert date strings to datetime objects
            start_timestamp = datetime.strptime(start_date, '%Y-%m-%d')
            end_timestamp = datetime.strptime(end_date, '%Y-%m-%d')

            # Check if start date is greater than or equal to end date
            if start_timestamp >= end_timestamp:
                return render_template('mba.html', msg="Error: Start date
must be before end date.")

        except ValueError:
            return render_template('mba.html', msg="Error: Invalid date
format. Please use YYYY-MM-DD.")

        # Concatenate values into a single string separated by a delimiter
        values_string = ','.join([support, start_date, end_date, shop_id])

        python_script_path = r"D:\\xampp\\htdocs\\migrate\\app\\sqlQuery.py"
        py_exec =
r"C:\\Users\\liman\\AppData\\Local\\Programs\\Python\\Python311\\python.exe"
        command = [py_exec, python_script_path, values_string]

        try:
            output = subprocess.check_output(command,
stderr=subprocess.STDOUT, text=True)
            return render_template('mba.html', msg=output.strip())
        except subprocess.CalledProcessError as e:
            return render_template('mba.html', msg=f"Error executing script:
{e.output}")

    return render_template('mba.html')
```

Segmen Program 4.29 Fungsi Generate MBA

Fungsi ini digunakan untuk menerima dan menangani parameter input dari form HTML berupa nama toko, minimum support, rentang waktu data transaksi untuk analisis MBA. Kemudian parameter dikirim dengan bentuk argument ke script Python sqlQuery.py . Argumen yang diterima akan digunakan oleh sqlQuery.py untuk melakukan proses pembacaan data dari database sesuai rentang waktu dan melakukan proses analisis MBA berdasarkan minimum support terkait. Hasil analisis akan ditampilkan di halaman HTML dalam bentuk tabel.

```
def insertBundle(shop_id, data):
    try:
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        for row in data:
            print(row)
            new_bundle = """
            INSERT INTO bundles (bundles_name, shop_id)
            VALUES (%s, %s);
            """

            num = random.randint(1, 100)
            current_datetime = datetime.now()
            stringRand = f"bundle-{num}-
{current_datetime.strftime('%Y%m%d%H%M%S')}"

            cursor.execute(new_bundle, (stringRand, shop_id))
            bundle_id = cursor.lastrowid

            itemsets = row["Set Produk"]
            antecedent, consequent = itemsets.split(" => ")

            support = row["Support"]
            lift = row["Lift"]
            confidence = row["Confidence"]
```

```
            insertItem(
                cursor,
                bundle_id,
                antecedent,
                consequent,
                support,
                confidence,
                lift,
            )

        # Commit changes to the database
        mysql.connection.commit()

        return True  # Indicate success
    except mysql.connection.Error as err:
        print(f"Error: {err}")
        return False  # Indicate failure

    finally:
        # Close cursor
        cursor.close()
```

Segmen Program 4.30 Fungsi Insert Bundle

Fungsi ini digunakan untuk menyimpan sebuah rule bundling hasil analisis MBA kedalam database sebagai entitas induk. Setiap rule bundling memiliki id bundle yang berbeda. Nantinya id dari bundle ini akan menjadi foreign key dari produk yang tersimpan di bundle items, dimana 1 id bundle dapat memiliki banyak produk. Memanggil fungsi insert item untuk melakukan input data produk ke database dengan iterasi hingga akhir data.

```
def insertItem(cursor, bundle_id, antecedent, consequent, support,
confidence, lift):
    try:
        insert_query = """
```

```
        INSERT INTO bundleitems (bundle_id, antecedent, consequent, support,
confidence, lift)
        VALUES (%s, %s, %s, %s, %s, %s)
        """
        cursor.execute(
            insert_query, (bundle_id, antecedent, consequent, support,
confidence, lift)
        )
    except mysql.connection.Error as err:
        print(f"Error inserting item: {err}")
```

Segmen Program 4.31 Fungsi Insert Item

Fungsi ini digunakan untuk menyimpan produk dari suatu id bundle. Menerima parameter nama produk antecedent, nama produk consequent, support, confidence, lift dan id bundle sebagai foreign key.

```
def handleBundling():
    data = request.form['jsonData']
    shop_id = request.form['shop_id']
    bundle_items = json.loads(data) # Assuming bundle items directly in the
JSON data

    if not shop_id or not bundle_items:
        return jsonify({'error': 'Missing shop_id or bundle_items'}), 400

    success = insertBundle(shop_id, bundle_items)
    if success:
        return jsonify({'message': 'Bundle inserted successfully'})
    else:
        return jsonify({'error': 'Failed to insert bundle'}), 500
```

Segmen Program 4.32 Fungsi Handle Bundling

Fungsi ini digunakan untuk menangani data JSON yang dikirim dari HTML terkait rule bundling hasil analisis MBA yang sudah difilter. Data JSON akan dibaca dan diproses dengan memanggil fungsi insert bundle.

```
def callFilteredBundlings():
    # Check if user is logged in (assuming you have session handling logic)
    if 'id' not in session:
        return redirect(url_for('login'))

    # Get the shop_id from the POST request
    shop_id = request.form['shop_id']

    # Call the print_sets function to generate HTML content
    sql = """
    SELECT id, antecedent, consequent, support, confidence, lift,
date_created
    FROM bundleitems
    JOIN bundles ON bundleitems.bundle_id = bundles.bundle_id
    WHERE bundles.shop_id = %s ORDER BY bundleitems.bundle_id DESC
    """
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute(sql, (shop_id,))
    rows = cursor.fetchall()

    html_out = f"<div style='height: 350px; overflow-y: scroll;'><table
class='table table-
striped'><tr><th>Itemsets</th><th>Support</th><th>Confidence</th><th>Lift</th
><th>Created At</th><th>Detail</th></tr>"

    for row in rows:
        html_out += f"<tr><td>{row['antecedent']} =>
{row['consequent']}</td><td>{row['support']}</td><td>{row['confidence']}</td>
<td>{row['lift']}</td><td>{row['date_created']}</td><td><button class='btn
btn-primary' type='button'
onclick='getDetail({row['id']})'>Detail</button></td></tr>"

    html_out += "</table></div>"
    # Return the HTML content as a response
```

```
    return html_out.strip()
```

Segmen Program 4.33 Fungsi Call Filtered Bundlings

Fungsi ini digunakan untuk memanggil data rule bundling hasil analisis MBA yang sudah difilter dan disimpan kedalam database dengan parameter id toko. Data akan ditampilkan dalam bentuk tabel di halaman HTML.

```
def bundlingDetail():
     if request.method == "POST":
        bundle_id = request.form['bundle_id']
        shop_id = request.form['shop_id']

        # Concatenate values into a single string separated by a delimiter
        values_string = ','.join([bundle_id, shop_id])

        python_script_path = r"D:\\xampp\\htdocs\\migrate\\app\\scraping.py"
        py_exec =
r"C:\\Users\\liman\\AppData\\Local\\Programs\\Python\\Python311\\python.exe"

        # Construct the command to execute
        command = f'"{py_exec}" "{python_script_path}" "{values_string}"'
        try:
            # Execute the command and capture the output
            output = subprocess.check_output(command, shell=True, text=True)
            return output.strip()
        except subprocess.CalledProcessError as e:
            return f"Error executing script: {e.output}"
```

Segmen Program 4.34 Fungsi Bundling Detail

Fungsi ini digunakan untuk melakukan scraping data produk di website Tokopedia dengan parameter id bundle dan id toko yang akan diteruskan ke script Python scraping.py. Scraping.py akan melakukan proses scraping dari tiap produk didalam rule bundle dan menampilkan hasil scraping dalam bentuk tabel ke halaman HTML.

```
def saveBundlingSets():
    if 'id' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        data = request.form.get('products')
        products = json.loads(data)
        group_id = uuid.uuid4().hex[:6]

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        for product in products:
            sql = '''
            INSERT INTO savedbundlings (product_name, normal_subtotal,
discount, discounted_price, quantity, unitPrice, items_id, shop_id, group_id)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
            '''
            cursor.execute(sql, (
                product['product_name'],
                product['normal_subtotal'],
                product['discount'],
                product['discounted_price'],
                product['quantity'],
                product['unitPrice'],
                product['items_id'],
                product['shop_id'],
                group_id
            ))

        mysql.connection.commit()
        cursor.close()
```

```
        return jsonify({'status': 'success', 'message': 'Products saved
successfully'})
```

Segmen Program 4.35 Fungsi Save Bundling Sets

Fungsi ini digunakan untuk menyimpan data produk hasil scraping yang dikombinasikan pengguna dari suatu rule bundling kedalam database dengan uuid spesial untuk setiap set. Suatu rule bundling dapat memiliki banyak variasi bundling.

```
def callSavedBundling():
    if 'id' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        shop_id = request.form['shop_id']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM savedbundlings WHERE shop_id = %s',
(shop_id))
        results = cursor.fetchall()
        if len(results) != 0:
            return jsonify({'message': 'Data bundlings found for this shop.',
'response': results})
```

Segmen Program 4.36 Fungsi Call Saved Bundling

Fungsi ini digunakan untuk memanggil detail tiap set produk bundling yang sudah disimpan di database kedalam suatu tabel di HTML.

```
def updateSavedBundling():
    if 'id' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        data = request.form.get('products')
        products = json.loads(data)

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
```

```
        for product in products:
            sql = '''
            UPDATE savedbundlings
            SET
                discount = %s,
                discounted_price = %s,
                quantity = %s
            WHERE itemset_id = %s
            '''

            cursor.execute(sql, (
                product['discount'],
                product['discounted_price'],
                product['quantity'],
                product['itemset_id'],
            ))

        mysql.connection.commit()
        cursor.close()

        return jsonify({'status': 'success', 'message': 'Products updated
successfully'})
```

Segmen Program 4.37 Update Saved Bundling

Fungsi ini digunakan untuk melakukan update data di database terhadap produk bundling seperti kuantitas tiap produk, promosi dan harga akhir bundling secara keseluruhan.

```
def delSaved():
    if 'id' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        group_id = request.form['group_id']
```

```
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute('DELETE FROM savedbundlings WHERE group_id = %s',
(group_id,))

        mysql.connection.commit()

        cursor.close()

        return jsonify({'status': 'success', 'message': 'Itemset deleted
successfully'})
```

Segmen Program 4.38 Delete Saved Bundling

Fungsi ini digunakan menghapus suatu bundling yang pernah disimpan oleh user dari database.

```
def callBuyersDataMonthly():

    if 'id' not in session:

        return redirect(url_for('login'))


    if request.method == "POST":

        shop_id = request.form['shop_id']

        month_periode = request.form['month']

        year_periode = request.form['year']

        # logging.info(shop_id, month_periode, year_periode)


        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute('SELECT city_name, buyers_count, orders_count,
products_sold, total_revenue FROM buyers WHERE shop_id = %s AND month = %s
AND year = %s', (shop_id, month_periode, year_periode))

        results = cursor.fetchall()

        if len(results) != 0:

            return jsonify({'message': 'Data found for this shop.',
'response': results})

        else:

            return redirect(url_for('dashboard'))
```

Segmen Program 4.39 Call Buyers Data Monthly

Fungsi ini digunakan untuk memanggil data laporan pembeli dari database dengan parameter bulan dan tahun. Data akan dikembalikan ke halaman HTML dalam bentuk JSON.

```
def callBuyersDataYearly():
    if 'id' not in session:
        return redirect(url_for('login'))


    if request.method == "POST":
        shop_id = request.form['shop_id']
        year_periode = request.form['year']
        # logging.info(shop_id, month_periode, year_periode)


        sql = '''
        SELECT
            city_name,
            SUM(buyers_count) AS total_buyers,
            SUM(orders_count) AS total_orders,
            SUM(products_sold) AS total_products_sold,
            SUM(total_revenue) AS total_revenue
        FROM
            buyers
        WHERE
            shop_id = %s
            AND year = %s
        GROUP BY
            city_name;
        '''
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute(sql, (shop_id, year_periode))
        results = cursor.fetchall()
        if len(results) != 0:
            return jsonify({'message': 'Data found for this shop.',
'response': results})
        else:
            return redirect(url_for('dashboard'))
```

Segmen Program 4.40 Call Buyers Data Yearly

Fungsi ini digunakan untuk memanggil data laporan pembeli dari database dengan parameter tahun. Data akan dikembalikan ke halaman HTML dalam bentuk JSON.

```
def callSalesDataMonthly():
    if 'id' not in session:
        return redirect(url_for('login'))

    if request.method == "POST":
        shop_id = request.form['shop_id']
        month_periode = request.form['month']
        year_periode = request.form['year']
        # logging.info(shop_id, month_periode, year_periode)

        sql = '''
        SELECT
            DAY(tgl_payment) AS day_of_month,
            MONTH(tgl_payment) AS month,
            SUM(gross_revenue) AS total_gross_revenue
        FROM
            orders_report
        WHERE
            shop_id = %s
            AND MONTH(tgl_payment) = %s
            AND YEAR(tgl_payment) = %s
        GROUP BY
            DAY(tgl_payment), MONTH(tgl_payment);
        '''
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute(sql, (shop_id, month_periode, year_periode))
        results = cursor.fetchall()
        if len(results) != 0:
```

```
            return jsonify({'message': 'Data found for this shop.',
'response': results})
        else:
            return redirect(url_for('dashboard'))
```

Segmen Program 4.41 Call Sales Data Monthly

Fungsi ini digunakan untuk memanggil data laporan pesanan dari database dengan parameter bulan dan tahun. Data akan dikembalikan ke halaman HTML dalam bentuk JSON.

```
def callSalesDataYearly():
    if 'id' not in session:
        return redirect(url_for('login'))

    if request.method == "POST":
        shop_id = request.form['shop_id']
        year_periode = request.form['year']
        # logging.info(shop_id, month_periode, year_periode)

        sql = '''
        SELECT
            MONTH(tgl_payment) AS month,
            SUM(gross_revenue) AS total_gross_revenue
        FROM
            orders_report
        WHERE
            shop_id = %s
            AND YEAR(tgl_payment) = %s
        GROUP BY
            MONTH(tgl_payment);
        '''
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute(sql, (shop_id, year_periode))
        results = cursor.fetchall()
        if len(results) != 0:
```

```
            return jsonify({'message': 'Data found for this shop.',
'response': results})
        else:
            return redirect(url_for('dashboard'))
```

Segmen Program 4.42 Call Sales Data Yearly

Fungsi ini digunakan untuk memanggil data laporan pesanan dari database dengan parameter tahun. Data akan dikembalikan ke halaman HTML dalam bentuk JSON.

```
def userData():
    if 'id' not in session:
        return redirect(url_for('login'))

    userId = session.get('id')
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELECT account_name, user_email FROM users WHERE
account_id = %s', (userId,))
    result = cursor.fetchone()

    if result:
        return jsonify({'result': result}), 200
    else:
        return jsonify({'error': 'User data not found'}), 404
```

Segmen Program 4.43 Fungsi User Data

Fungsi ini digunakan untuk memanggil data pengguna untuk ditampilkan di halaman pengaturan akun.

```
def updateProfile():
    if 'id' not in session:
        return redirect(url_for('login'))

    userId = session.get('id')
    if request.method == "POST" and 'usrname' in request.form and 'emailUsr'
in request.form and 'passUsr' in request.form :
```

```
        usrname = request.form['usrname']
        emailUsr = request.form['emailUsr']
        passUsr = request.form['passUsr']
        hashed_pass = hash_password(passUsr)
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT COUNT(*) FROM users WHERE user_email = %s AND
account_id != %s', ((usrname, emailUsr)))
        if cursor.fetchone() != None:
           return render_template('user.html', upd="Email already exists")
        else:
            cursor.execute('UPDATE users SET account_name = %s, user_email =
%s, user_pass = %s WHERE account_id = %s',
                    (usrname, emailUsr, hashed_pass, userId))
            # Commit changes to the database
            mysql.connection.commit()
            return render_template('user.html', upd="User data updated
successfully")
```

Segmen Program 4.44 Fungsi Update Profile

Fungsi ini digunakan untuk melakukan update data akun di database sesuai dengan parameter input dari form HTML.

```
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))
```

Segmen Program 4.45 Fungsi Logout

Fungsi ini digunakan untuk menghancurkan sesi login dari suatu akun dan mengarahkan user ke halaman login.

```
def generic_template_renderer(template_name):
    return render_template(template_name)
```

```
# Define routes and their respective handler functions

routes = [

    {'endpoint': 'login', 'route': '/login', 'methods': ['GET', 'POST'],
'handler': login},

    {'endpoint': 'register', 'route': '/register', 'methods': ['GET',
'POST'], 'handler': register},

    {'endpoint': 'forgetPass', 'route': '/forgetPass', 'methods': ['GET',
'POST'], 'handler': forgetPass},

    {'endpoint': 'userData', 'route': '/userData', 'methods': ['GET'],
'handler': userData},

    {'endpoint': 'updateProfile', 'route': '/updateProfile', 'methods':
['POST'], 'handler': updateProfile},

    {'endpoint': 'callShop', 'route': '/callShop', 'methods': ['GET'],
'handler': callShop},

    {'endpoint': 'callBundlings', 'route': '/callBundlings', 'methods':
['GET'], 'handler': callBundlings},

    {'endpoint': 'callFilteredBundlings', 'route': '/callFilteredBundlings',
'methods': ['POST'], 'handler': callFilteredBundlings},

    {'endpoint': 'callBuyersDataMonthly', 'route': '/callBuyersDataMonthly',
'methods': ['POST'], 'handler': callBuyersDataMonthly},

    {'endpoint': 'callBuyersDataYearly', 'route': '/callBuyersDataYearly',
'methods': ['POST'], 'handler': callBuyersDataYearly},

    {'endpoint': 'callSalesDataMonthly', 'route': '/callSalesDataMonthly',
'methods': ['POST'], 'handler': callSalesDataMonthly},

    {'endpoint': 'callSalesDataYearly', 'route': '/callSalesDataYearly',
'methods': ['POST'], 'handler': callSalesDataYearly},

    {'endpoint': 'callscraperesult', 'route': '/callscraperesult', 'methods':
['POST'], 'handler': callscraperesult},

    {'endpoint': 'addShop', 'route': '/addShop', 'methods': ['GET','POST'],
'handler': addShop},

    {'endpoint': 'delShop', 'route': '/delShop', 'methods': ['POST'],
'handler': delShop},

    {'endpoint': 'saveBundlingSets', 'route': '/saveBundlingSets', 'methods':
['POST'], 'handler': saveBundlingSets},

    {'endpoint': 'callSavedBundling', 'route': '/callSavedBundling',
'methods': ['POST'], 'handler': callSavedBundling},

    {'endpoint': 'updateSavedBundling', 'route': '/updateSavedBundling',
'methods': ['POST'], 'handler': updateSavedBundling},

    {'endpoint': 'delSaved', 'route': '/delSaved', 'methods': ['POST'],
'handler': delSaved},
```

```
    {'endpoint': 'checkFiles', 'route': '/checkFiles', 'methods': ['POST'],
'handler': checkFiles},

    {'endpoint': 'loadLog', 'route': '/loadLog', 'methods': ['POST'],
'handler': loadLog},

    {'endpoint': 'upload', 'route': '/upload', 'methods': ['POST'],
'handler': upload},

    {'endpoint': 'generateMBA', 'route': '/generateMBA', 'methods':
['GET','POST'], 'handler': generateMBA},

    {'endpoint': 'handleBundling', 'route': '/handleBundling', 'methods':
['POST'], 'handler': handleBundling},

    {'endpoint': 'bundlingDetail', 'route': '/bundlingDetail', 'methods':
['POST'], 'handler': bundlingDetail},

    {'endpoint': 'logout', 'route': '/logout', 'methods': ['GET'], 'handler':
logout},

    {'endpoint': 'buyerInsight', 'route': '/buyerInsight', 'methods':
['GET'], 'template': 'buyerInsight.html'},

    {'endpoint': 'user', 'route': '/user', 'methods': ['GET'], 'template':
'user.html'},

    {'endpoint': 'mba', 'route': '/mba', 'methods': ['GET'], 'template':
'mba.html'},

    {'endpoint': 'files', 'route': '/files', 'methods': ['GET'], 'template':
'files.html'},

    {'endpoint': 'bundlings', 'route': '/bundlings', 'methods': ['GET'],
'template': 'bundlings.html'},

    {'endpoint': 'dashboard', 'route': '/dashboard', 'methods': ['GET'],
'template': 'dashboard.html'},

    {'endpoint': 'bundlingMenu', 'route': '/bundlingMenu', 'methods':
['GET'], 'template': 'bundlingMenu.html'},

]


# Add routes dynamically

for route in routes:

    if 'handler' in route:

        app.add_url_rule(route['route'], endpoint=route['endpoint'],
view_func=route['handler'], methods=route['methods'])

    elif 'template' in route:

        app.add_url_rule(route['route'], endpoint=route['endpoint'],
view_func=lambda template_name=route['template']:
generic_template_renderer(template_name), methods=route['methods'])
```

Segmen Program 4.46 Fungsi dan Routing Aplikasi Flask

Pada segmen program ini, terdapat berbagai fungsi Flask yang bertanggung jawab untuk melakukan pemindahan data antara HTML dan database MySQL, serta routing fungsi dan file HTML terkait agar dapat diakses dari antarmuka aplikasi Flask. Beberapa fungsi kunci yang mungkin termasuk dalam segmen program ini adalah:

- Pengaturan routing dalam aplikasi Flask untuk menentukan alamat URL yang akan dihubungkan dengan fungsi tertentu dalam aplikasi.
- Penggunaan fungsi `render_template` untuk memuat file HTML ke dalam aplikasi Flask, sehingga halaman web dapat diakses oleh pengguna.

Dengan menggunakan kombinasi fungsi-fungsi ini, aplikasi Flask dapat menjadi jembatan yang efektif antara antarmuka pengguna HTML dan database MySQL, memungkinkan pengguna untuk berinteraksi dengan aplikasi dan mengakses serta memanipulasi data dengan lancar.

```
if __name__ == '__main__':
    app.run(debug=True)
```

Segmen Program 4.47 Fungsi inisialisasi Flask

Kode ini digunakan untuk menjalankan server Flask. Saat aplikasi ini dijalankan sebagai program utama, server Flask akan mulai untuk melayani permintaan HTTP.