# BAB III
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinas

Bagian dalam kerja magang yang di laksanakan di Tong Tji Tea Indonesia dan posisi yang tersedia bagi peserta magang adalah Business Intelligence Engineer (BI Engineer). Peran berada di bawah divisi IT yang befokus dalam pengelolaan dan memanfaatkan data lalu mengvisualisasikannya dengan tujuan bisa mendukung pengambilan keputusan berbasis data. Selama magang, peserta yang melakukan magang bekerja secara individu tetapi tetap dibawah bimbingan Supervisor Head Data Engineer (Raymond) yang berhubungan dengan pengembangan system. Penugasan yang diberikan untuk posisi BI Engineer di dalam kerja magang yang dilakukan yaitu:

1. Melakukan ekstraksi data.
2. Merancang struktur database.
3. Mengambangkan dashboard interaktif.
4. Membuat query SQL
5. Menerapkan control akses berbasis role (Admin, HO, User).
6. Menganalisis pola data pembelian dan retur.

## 3.2 Tugas yang dilakukan

Kegiatan kerja magang yang dilakukan di PT Tong Tji Indonesia berfokus pada pengelolaan data transaksi pembelian perusahaan serta pengembangan sistem visualisasi data berbasis web. Aktivitas utama yang dikerjakan meliputi pengolahan file CSV yang berisi data transaksi dari sistem internal perusahaan, kemudian melakukan proses pembersihan data dan penyesuaian struktur agar sesuai dengan skema tabel di dalam database PostgreSQL.

Setelah data siap, proses selanjutnya adalah melakukan import data ke dalam sistem database menggunakan query SQL. Langkah tersebut bertujuan untuk menjaga integritas data dan memastikan database dapat digunakan sebagai sumber utama dalam analisis dan pelaporan.

Dalam mendukung kebutuhan user internal perusahaan, dikembangkan sebuah dashboard interaktif menggunakan framework Streamlit. Dashboard tersebut dirancang untuk menampilkan berbagai informasi penting terkait aktivitas pembelian, seperti grafik total pembelian, jumlah item, performa supplier, serta pergerakan stok barang. Dashboard juga dilengkapi dengan fitur filtering berdasarkan entitas, lokasi, dan periode transaksi, serta sistem role-based access control yang membedakan hak akses antara admin, user HO, dan user biasa

*Tabel 3. 1 detail pekerjaan magang yang dilakukan*

| No | Pekerjaan yang dilakukan | Waktu (Minggu-ke) |
|---|---|---|
| 1 | Pengenalan dengan lingkungan kerja | 1 |
| 2 | Pembuatan tabel dan mengimport data menuju postgresql | 1, 2, 4, 8, 10, 11 |
| 3 | Membuat tampilan login dan sistem role untuk login | 1, 2 |
| 4 | Membuat Tampilan home | 1, 2 |
| 5 | Membuat Tampilan user sanagement | 3, 4, 6, 10, 13 |
| 6 | Membuat dashboard pembelian entity | 1, 2, 3 |
| 7 | Membuat dashboard pembelian retur | 4, 5, 6, 7, |
| 8 | Membuat dashboard pembelian stock | 8, 9, 13 |
| 9 | Membuat dashboard penjualan | 10, 11, 12 |

Tabel di atas menjelaskan mengenai tugas, uraian dan kerja mahasiswa magang di instansi/perusahaan tempat kerja magang.

### 3.2.1 Pembuatan Tabel dan Impor Data dari CSV

Sebelum membangun visualisasi data dalam dashboard, langkah awal yang dilakukan adalah mengimpor data transaksi yang masih berupa file CSV ke dalam database PostgreSQL. PostgreSQL sendiri didukung oleh tools manajemen basis data, yaitu pgAdmin 4.

Proses dimulai dengan pembuatan beberapa tabel yang sesuai dengan struktur data yang dibutuhkan menggunakan pgAdmin 4. Setelah tabel-tabel siap, data dari file CSV diimpor ke dalam masing-masing tabel tersebut. File CSV tersebut disediakan langsung oleh pihak PT Tong Tji Tea Indonesia dalam bentuk data dummy, mengingat data asli perusahaan bersifat rahasia dan tidak dapat

disebarluaskan, termasuk kepada peserta magang. Setelah proses impor berhasil, data yang telah masuk ke dalam database kemudian digunakan sebagai dasar untuk proses query, analisis, dan pembuatan visualisasi data pada tahap pengembangan dashboard. Berikut adalah daftar tabel yang telah dibuat dan diimpor dari file CSV:

A. Tabel entitymanagement

*Tabel 3. 2 Tabel entitymanagement*

| Field | Type | Key |
|---|---|---|
| id | INT | PK |
| entity | INT | |

Tabel menyimpan informasi entitas (entity) yang dapat dikaitkan dengan pengguna. Entity di sini merepresentasikan unit bisnis atau cabang perusahaan yang memiliki akses terhadap data tertentu dalam dashboard. Tabel berfungsi sebagai penghubung antara user dan entitas yang diizinkan untuk dilihat.

B. Tabel userlogin

*Tabel 3. 3 tabel userlogin*

| Field | Type | Key |
|---|---|---|
| id | INT | PK |
| username | VARCHAR | |
| email | VARCHAR | |
| password | TEXT | |
| createdadd | TIMESTAMP | |
| role | VARCHA | |

Tabel berisi informasi pengguna sistem seperti username, email, password (yang telah di-hash), dan role. Data dalam tabel digunakan untuk proses autentikasi login serta pengaturan hak akses pengguna (Admin, User)

C. Tabel initem

*Tabel 3. 4 tabel initem*

| Field | Type | Key |
|---|---|---|
| ItemID | VARCHAR | PK |
| Description | VARCHAR | |
| ShortName | VARCHAR | |
| UnitSetID | VARCHAR | |
| InventoryUnit | VARCHAR | |
| ItemGroupID | INT | FK |
| ItemModelID | INT | FK |

Tabel menyimpan data barang atau item yang tersedia di perusahaan. Setiap baris mewakili satu item lengkap dengan deskripsinya, unit yang digunakan, serta referensi ke kelompok dan model item. Tabel menjadi sumber utama saat menampilkan informasi produk dalam dashboard.

D. Tabel initemgroup

*Tabel 3. 5 tabel initemgorup*

| Field | Type | Key |
|---|---|---|
| ItemGroupID | INT | PK |
| Description | VARCHAR | |
| ShortDesc | VARCHAR | |

Tabel berfungsi untuk mengelompokkan item berdasarkan grup atau kategori besar, seperti jenis produk atau distributor. Relasi membantu pengguna dalam menyaring dan menganalisis data pembelian berdasarkan grup item.

E. Tabel initemmodel

*Tabel 3. 6 tabel initemmodel*

| Field | Type | Key |
|---|---|---|
| ItemModelID | INT | PK |
| Description | VARCHAR | |

Tabel berisi informasi model dari item. Model ini penting untuk klasifikasi lebih detail dan digunakan dalam analisis pembelian berdasarkan jenis produk.

F. Tabel pxentity

*Tabel 3. 7 tabel pxentity*

| Field | Type | Key |
|---|---|---|
| EntityID | VARCHAR | PK |
| Description | VARCHAR | |

Tabel menyimpan daftar entity atau unit bisnis perusahaan. Entity dapat berupa cabang, divisi, atau outlet, dan menjadi bagian penting dalam memfilter data transaksi berdasarkan lokasi bisnis yang relevan.

G. Tabel pxsite

*Tabel 3. 8 tabel pxsite*

| Field | Type | Key |
|---|---|---|
| EntityID | VARCHAR | PK |
| Description | VARCHAR | |

Tabel berisi informasi tentang lokasi atau site tempat transaksi dilakukan. Digunakan sebagai referensi lokasi fisik operasional, terutama dalam data pembelian dan penjualan.

H. Tabel pxperiod

*Tabel 3. 9 tabel pxperiod*

| Field | Type | Key |
|---|---|---|
| PeriodID | INT | PK |
| Description | VARCHAR | |
| StartDate | DATE | |
| EndDate | DATE | |

Tabel menyimpan data periode dalam bentuk tanggal awal dan akhir serta deskripsi periode (biasanya format bulan-tahun). Periode digunakan untuk mengelompokkan transaksi berdasarkan waktu pelaporan.

I. Tabel rptransdtl

*Tabel 3. 10 tabel rptransdtl*

| Field | Type | Key |
|---|---|---|
| PeriodID | INT | PK |
| SiteID | VARCHAR | FK |
| Sys | INT | FK |
| LineNo | INT | FK |
| ItemID | VARCHAR | FK |
| QT | NUM | |
| UnitPrice | NUM | |
| NetAmt | NUM | |

Tabel mencatat detail transaksi pembelian per item: mencakup jumlah (QT), harga, dan nilai total (NetAmt). Informasi digunakan dalam analisis pembelian pada dashboard.

J. Tabel rptranshdr

*Tabel 3. 11 tabe rptranshdr*

| Field | Type | Key |
|---|---|---|
| EntityID | VARCHAR | PK |
| SiteID | VARCHAR | FK |
| PeriodID | INT | FK |
| SuppID | INT | FK |
| Sys | INT | FK |
| TrDate | DATE | |
| TrNo | VARCHAR | |
| TrManualRef | VARCHAR | |
| NetAmt | NUM | |
| TransCode | SMALLINT | |
| CustID | VARCHAR | |

Tabel menyimpan informasi utama dari transaksi seperti tanggal transaksi, supplier, dan kode transaksi (TransCode). Tabel dikombinasikan dengan rptransdtl untuk menghasilkan laporan pembelian dan retur secara lengkap.

K. Table appsupplier

*Tabel 3. 12 tabel appsupplier*

| Field | Type | Key |
|---|---|---|
| SuppID | INT | PK |
| SuppName | VARCHAR | |

Tabel menyimpan daftar supplier atau distributor yang bekerja sama dengan perusahaan. Digunakan untuk analisis performa supplier serta pengelompokan transaksi berdasarkan pemasok.

L. Tabel arareae

*Tabel 3. 13 tabel arareae*

| Field | Type | Key |
|---|---|---|
| AreaID | VARCHAR | PK |
| Description | VARCHAR | |
| ParentID | VARCHAR | FK |

Berisi data area atau wilayah yang digunakan untuk klasifikasi geografis pelanggan atau cabang. Penting dalam analisis penjualan atau distribusi berdasarkan wilayah.

M. Tabel arcustomer

*Tabel 3. 14 tabelarcustomer*

| Field | Type | Key |
|---|---|---|
| CustID | VARCHAR | PK |
| AreaID | VARCHAR | FK |
| SalesRepID | VARCHAR | FK |
| GroupAID | INT | FK |

Tabel berisi informasi pelanggan, termasuk area tempat mereka berada, dan representatif sales-nya. Data mendukung fitur penjualan berbasis customer entity dalam dashboard.

N. Tabel arcustomergroupa

*Tabel 3. 15 tabel acustomergroupa*

| Field | Type | Key |
|---|---|---|
| GroupAID | INT | PK |
| Description | VARCHAR | |

Tabel digunakan untuk mengelompokan pelanggan ke dalam grup tertentu, seperti, Retail, Rescaf. Data membantu dalam analisis penjualan berdasarkan segmen pelanggan.

### 3.2.2 Membuat Dashboard BI bersistem Login

Sistem memiliki halaman login yang dibangun menggunakan Streamlit untuk memastikan bahwa hanya pengguna yang memiliki kredensial yang diperlukan dapat mengakses dashboard. Proses login dimulai dengan memverifikasi username dan password yang dimasukkan pengguna. Password yang disimpan di database telah dienkripsi menggunakan bcrypt untuk memastikan data pengguna aman. Saat pengguna menekan tombol login, sistem akan mengambil data dari tabel userlogin, mencocokkan username yang diberikan, dan memverifikasi password dengan membandingkannya dengan hash yang disimpan di database. Jika berhasil, status sesi Streamlit akan diperbarui untuk menyimpan status login, nama pengguna, dan data entitas pengguna.

Agar sistem dapat membedakan hak akses antar pengguna. Dibuat modul utilitas tambahan dalam file *dashboard_utils.py* yang melacak peran masing-masing pengguna agar sistem dapat membedakan hak akses antar pengguna. Modul tersebut memiliki fungsi seperti *is_admin()* untuk mengecek apakah pengguna adalah administrator berdasarkan peran yang tersimpan di tabel

*userlogin*. Dengan menggunakan st.session_state, seluruh data role dan status login pengguna disimpan secara dinamis, Sehingga proses pengaturan hak akses pada setiap halaman dashboard dilakukan dengan cepat. Metode tersebut membuat dashboard menampilkan berbagai konten sesuai dengan peran pengguna yang sedang aktif.

#### 3.2.2.1 Menghubungkan Streamlit dengan Database PostgreSQL

Dalam pengembangan sistem dashboard berbasis web, pemilihan framework atau platform pengembangan memegang peranan yang sangat penting karena akan berdampak langsung terhadap kecepatan pengembangan, fleksibilitas sistem, kemudahan deployment, serta performa aplikasi yang dihasilkan. Pada proyek, framework Streamlit dipilih sebagai platform utama dalam membangun seluruh aplikasi dashboard. Streamlit merupakan framework open-source berbasis bahasa pemrograman Python yang secara khusus dirancang untuk memudahkan pengembangan aplikasi berbasis data, seperti dashboard analitik, machine learning interface, serta aplikasi monitoring secara real-time. Dengan Streamlit, pengembang hanya dapat berkonsentrasi pada logika pengolahan data dan visualisasinya. Berbeda dengan framework web konvensional seperti Django atau Flask yang memerlukan pemahaman tambahan tentang konsep routing, template HTML, CSS, dan JavaScript. Dengan demikian, framework web sangat cocok untuk pengembangan sistem berbasis data yang cepat dan ringan.

Sebelum data dapat divisualisasikan melalui aplikasi dashboard yang dibangun dengan Streamlit**,** hal pertama yang harus dilakukan adalah menghubungkan aplikasi tersebut dengan database PostgreSQL yang sebelumnya telah dikelola melalui pgAdmin 4. Aplikasi web dapat secara langsung mengakses data yang tersimpan di tabel database berkat koneksi. Untuk melakukan,modul *database.py* dibuat, yang berisi fungsi *connect_db()*. Untuk membuka koneksi ke database, library psycopg2 digunakan oleh fungsi. Fungsi tersebut menetapkan parameter koneksi seperti host, dbname, user, password, dan port yang sesuai dengan konfigurasi database local. Kemudian modul dipanggil ke berbagai bagian aplikasi lain, seperti login, pengambilan data transaksi, dan halaman visualisasi. Dengan demikian, proses pengolahan dan penyajian data dapat dilakukan secara

real-time dan terintegrasi secara langsung dengan sistem database yang digunakan oleh perusahaan.

```python
import psycopg2

def connect_db():
    """Mengatur koneksi ke database PostgreSQL."""
    return psycopg2.connect(
        host="localhost",
        dbname="Sales_db_restore",
        user="postgres",
        password="1234",
        port="5432"
    )
```

*Gambar 3. 1 Connect database streamlit*

### 3.2.2.2 Membuat Halaman Login dengan Sistem Autentikasi

Setelah membangun koneksi ke database, langkah berikutnya adalah membuat sistem autentikasi. Sistem akan mengatur akses pengguna ke dashboard, dilakukan dengan membuat file *login.py* yang berfungsi sebagai halaman login. File tersebut memiliki fungsi penting *login_page()*, yang digunakan untuk menunjukkan form login kepada pengguna.

```python
def login_page():
    initialize_session_state()
```

*Gambar 3. 2 Function login page code*

Pengguna akan diminta untuk memasukkan username dan password mereka, yang kemudian diverifikasi ke dalam tabel *userlogin* database PostgreSQL. Verifikasi dilakukan dengan membandingkan password yang dimasukkan dengan password yang telah dihash menggunakan *bcrypt*.

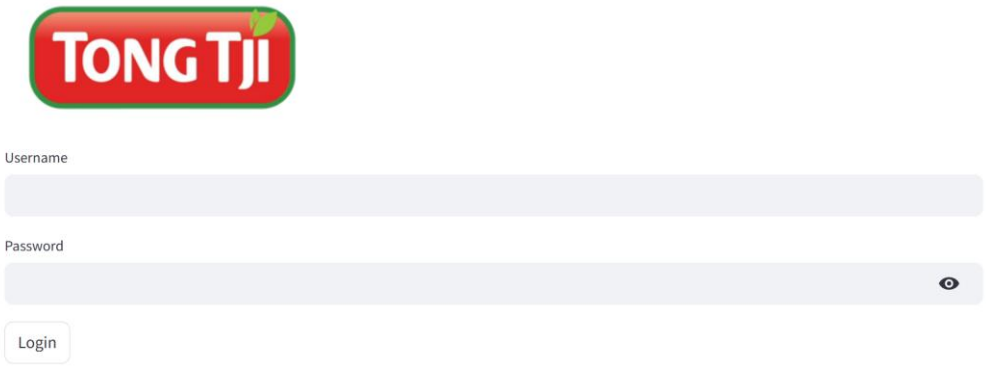
```
if user:
    stored_hash = user[3]
    try:
        if bcrypt.checkpw(password.encode('utf-8'), stored_hash.encode('utf-8')):
            return user
    except ValueError:
        return None
return None
```

*Tabel 3. 3 Function hash code*

Pada halaman login juga ditampilkan logo perusahaan sebagai elemen visual tambahan, dan jika login berhasil maka data user seperti username dan entity akan disimpan ke dalam session_state untuk digunakan di halaman dashboard. Bila login gagal, sistem akan menampilkan notifikasi kesalahan. Code tersebut juga memanggil fungsi initialize_session_state() dari modul dashboard_utils untuk memastikan variabel session sudah siap sebelum digunakan. Berikut merupakan cuplikan kode untuk halaman login:

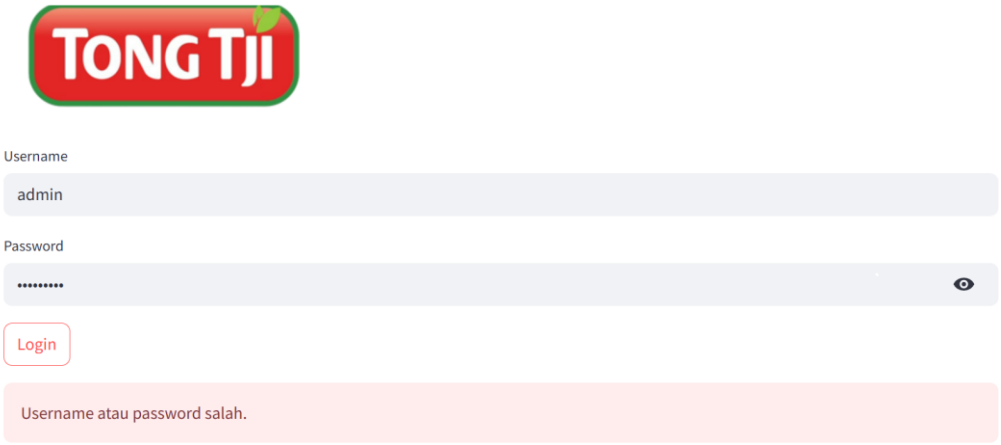
```
if not st.session_state.logged_in:
    img = Image.open("picture/tongtji1.png")
    img = img.resize((300, int(300 * img.height / img.width)))
    st.image(img)

    username = st.text_input("Username")
    password = st.text_input("Password", type="password")
    if st.button("Login"):
        user = check_login(username, password)
        if user:
            st.session_state.logged_in = True
            st.session_state.username = user[1]
            st.session_state.entity = get_user_entity(user[1])
            st.success(f"Selamat datang, {user[1]}!")
            st.rerun()
        else:
            st.error("Username atau password salah.")
```

*Gambar 3. 4 Tampilan login code*




*Tabel 3. 5 Tampilan login*

Gambar dibawah merupakan gambar jika user salah menginput Username dan Password.




*Gambar 3. 6 Tampilan login jika gagal login*

Selain login, file tersebut menggunakan fungsi *initialize_session_state()* dari file dashboard_utils.py untuk mengatur logika session state. Fungsitersebut memastikan bahwa variabel seperti logged_in, username, dan user_entity sudah tersedia di dalam session state sebelum digunakan, untuk mencegah kesalahan saat aplikasi berjalan. Dalam file *dashboard.py*, sistem akan menjalankan fungsi *show_dashboard()* untuk menampilkan halaman utama dashboard sesuai dengan hak akses pengguna. Namun, jika login gagal, sistem akan menampilkan pesan error untuk memberi tahu pengguna bahwa username atau password yang mereka masukkan salah.

File *login.py* sangat penting untuk melindungi akses sistem dan membantu membedakan pengguna menurut peran atau entitas mereka sebelum mereka dapat mengakses dan menggunakan fitur di dashboard.

### 3.2.2.3 Mengatur Stake dan Hak Akses Login

Fungsi pendukung di file *dashboard_utils.py* sangat penting untuk pengaturan state session dan pembagian hak akses pengguna berdasarkan peran mereka. Fungsi tersebut sangat penting untuk sistem login karena memastikan aplikasi tahu siapa yang sedang login dan apa hak aksesnya.

Fungsi *initialize_session_state()*. Fungsi tersebut dipanggil pada awal aplikasi untuk memastikan variabel session seperti l*ogged_in, username,* dan *user_entity* sudah ada sebelum digunakan. Jika belum ada, maka akan di-set dengan nilai default. Beretujuan untuk mencegah error ketika session belum diinisialisasi tapi sudah dipanggil, yang bisa menyebabkan crash atau bug pada aplikasi Streamlit.

```
def initialize_session_state():
    """Inisialisasi session state untuk login."""
    if "logged_in" not in st.session_state:
        st.session_state.logged_in = False
    if "username" not in st.session_state:
        st.session_state.username = None
    if "user_entity" not in st.session_state:
        st.session_state.user_entity = None
```

*Gambar 3. 7 Funtion state untuk login*

Selain itu, fungsi *is_admin()* yang digunakan untuk memastikan apakah user yang sedang login memiliki role sebagai Admin. Fungsi melakukan query langsung ke tabel userlogin berdasarkan username yang tersimpan di *session state*, lalu memastikan apakah role-nya adalah "Admin". Jika itu benar, maka nilai True akan dikembalikan, dan fitur admin khusus seperti manajemen user akan dapat diakses. Secara keseluruhan, *dashboard_utils.py* adalah fondasi dari sistem

otorisasi pengguna yang dipakai di seluruh aplikasi, karena semua pengecekan role atau state login bergantung pada fungsi-fungsi

```python
def is_admin():
    """Mengecek apakah pengguna saat ini adalah admin."""
    if "logged_in" in st.session_state and st.session_state.logged_in:
        conn = connect_db()
        cursor = conn.cursor()
        cursor.execute("SELECT role FROM userlogin WHERE username = %s", (st.session_state.username,))
        result = cursor.fetchone()
        conn.close()
        if result and result[0] == "Admin":
            return True
    return False
```

*Gambar 3. 8 Funtion untuk mengecek role Admin*

### 3.2.3 Dashboard

Sistem dashboard mulai dikembangkan sebagai antarmuka pengguna utama. Dengan desain Streamlit, dashboard berfungsi sebagai pusat navigasi untuk berbagai fitur yang disediakan. Hal tersebut mencakup halaman rumah, manajemen pengguna, pembelian stock, retur, dan tampilan data pembelian. Fitur yang ditampilkan berbasis peran, yang berarti disesuaikan dengan peran atau hak akses pengguna.

Tampilan tersebut terdiri dari kode program yang ada di file dashboard.py. Di dalamnya terdapat fungsi show_dashboard(), yang bertanggung jawab untuk menyiapkan layout halaman dan mengatur struktur menu. Setelah layout dikonfigurasi *dengan st.set_page_config()*, logo dan judul perusahaan ditampilkan di bagian atas dan sidebar. Logo dapat dimuat dan disesuaikan ukurannya dengan menggunakan pustaka PIL.

```python
def show_dashboard():
    st.set_page_config(layout="wide")
    st.title("Dashboard")
```

*Gambar 3. 9 Code untuk menampilkan dashboard*

Di bagian sidebar, digunakan komponen menu dari library streamlit_antd_components untuk membuat navigasi yang interaktif. Menu yang tersedia diatur berdasarkan role pengguna. Misalnya, semua pengguna akan melihat menu “Home” dan “Purchase” (dengan submenu “Pembelian Entity” dan “Pembelian Retur”), tetapi hanya user dengan role **Admin** atau **HO** yang bisa melihat menu tambahan seperti “Pembelian Stock” dan “User Management”.

```
import streamlit as st
import streamlit_antd_components as sac
```

*Gambar 3. 10 Import streamlit*

Bergantung pada pilihan pengguna, setiap menu akan memanggil fungsi tampilan yang berbeda. Submenu lainnya, seperti *show_retur(), show_purchase_stock()*, dan *show_user_management(),* masing-masing memanggil halaman visualisasi atau manajemen data tertentu, akan menampilkan konten dari *show_home()* jika pengguna memilih "Rumah" dan fungsi *show_purchase()* akan menampilkan data pembelian yang sesuai diinginkan oleh User .

```
if selected_menu == 'Home':
    show_home()
elif selected_menu == 'Pembelian Entity':
    show_purchase(df)
elif selected_menu == 'Pembelian Retur':
    show_retur(df)
elif selected_menu == 'Pembelian Stock':
    if is_ho_user() or is_admin():
        show_pembelian_stock(df)
    else:
        st.error("Kamu tidak punya akses ke halaman ini.")
elif selected_menu == 'User Management' and admin_status:
    show_user_management()
```

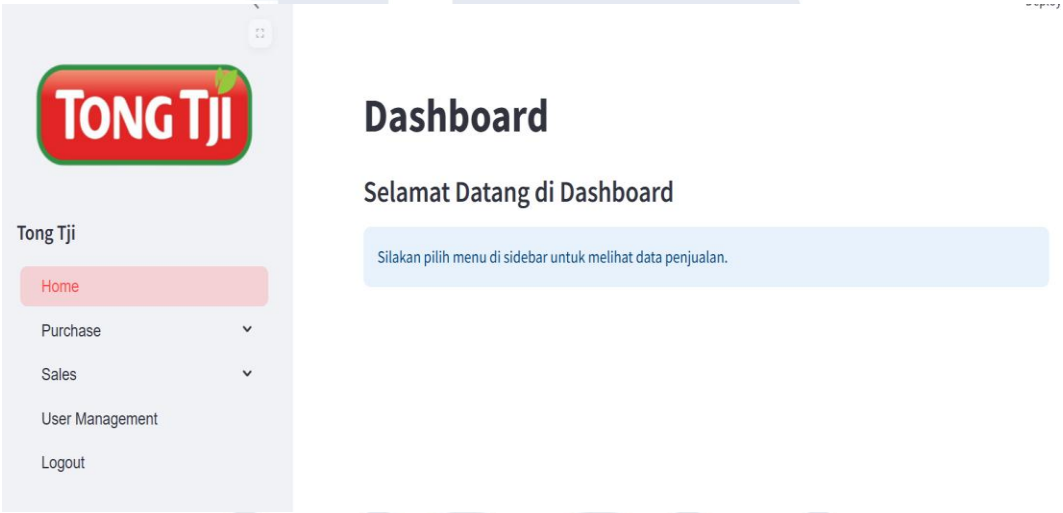
*Gambar 3. 11 Code untuk tampilan menu*

Dengan menggunakan fungsi *load_data()* dari *data_loader.py*, data transaksi yang ditampilkan pada dashboard akan difilter secara otomatis sesuai dengan entitas pengguna yang sedang login. Hanya pengguna dengan hak akses tertentu,

seperti admin, akan dapat melihat data secara keseluruhan tanpa filter. Selain itu, dashboard memiliki fitur logout, yang akan menghapus status login dan mereset session menggunakan st.session_state.

```
username = st.session_state.get('username')
user_entity = get_user_entity(username) if username else None
admin_status = is_admin()
df = load_data() if admin_status else load_data(entity_filter=user_entity)
```

*Gambar 3. 12 Code membuat user entity dan admin status*



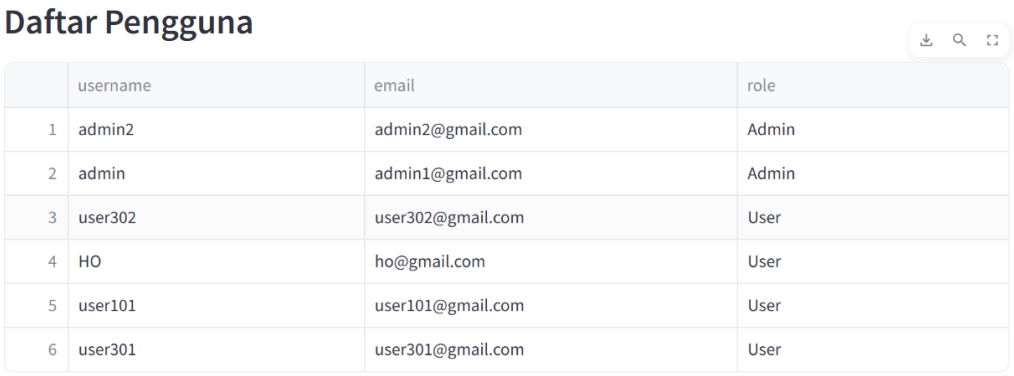

*Gambar 3. 13 Tampilan dashboard setelah login*

### 3.2.3 User Management

Fitur Manajemen merupakan sebuah komponen yang penting dalam pengembangan sistem dashboard. Dikembangkan untuk memberikan administrator otoritas penuh untuk mengatur akses pengguna ke sistem, sehingga hanya user tertentu yang dapat mengakses sistem sesuai dengan perannya masing-masing. Selain itu, mengelola data user secara terus-menerus membuat administrasi lebih mudah, terutama ketika ada perubahan dalam staff atau pembaruan hak akses dalam operasi bisnis. Fungsi Manajemen Pengguna dirancang untuk melakukan beberapa tugas penting, seperti menambahkan pengguna baru, memperbarui data pengguna, dan menghapus pengguna dari

sistem. Tampilan antarmuka aplikasi berbasis Streamlit memungkinkan kontrol interaktif atas semua proses.

Sistem dashboard dilengkapi dengan fitur yang menampilkan daftar seluruh pengguna yang telah terdaftar di dalam database. Fitur tersebut berguna untuk memberikan transparansi terhadap struktur pengguna yang memiliki akses ke sistem, sekaligus sebagai alat bantu bagi administrator dalam memantau dan mengelola akun yang aktif.

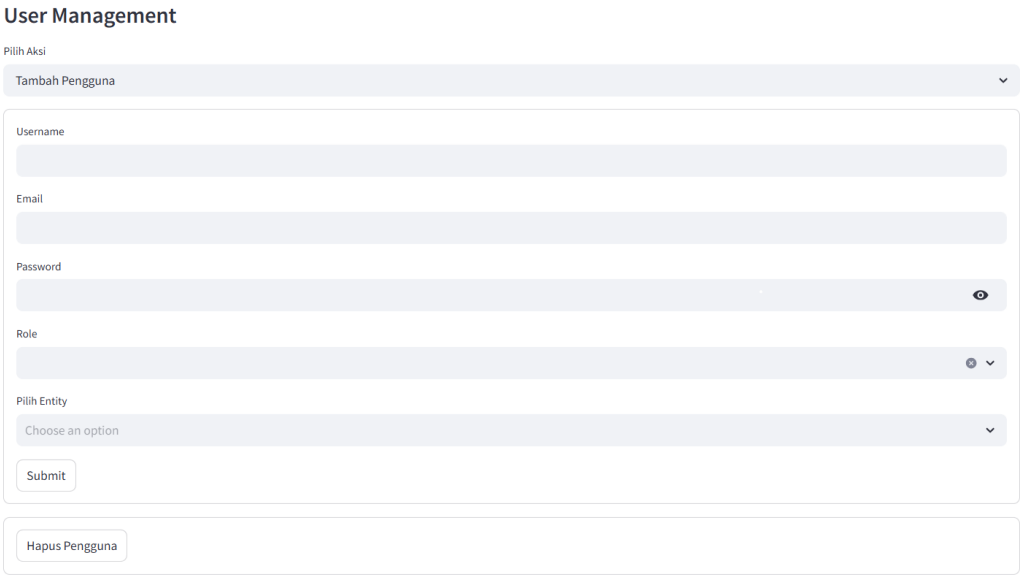



*Gambar 3. 14 Tampilan daftar pengguna*

### 3.2.3.1 Tambah Pengguna

User Management hanya bisa diakses oleh role Admin, role User tidak dapat melihat ataupun mengakses User Manegement. Fitur "Tambah Pengguna" dalam dashboard Streamlit dirancang untuk memungkinkan administrator menambahkan user baru ke dalam sistem. Fitur tersebut terintegrasi langsung dengan database PostgreSQL dan mencakup form input yang terdiri dari informasi penting seperti username, email, password, role, serta entitas yang ingin dikaitkan dengan pengguna tersebut. Saat halaman "*Tambah Pengguna*" dipilih, sistem akan menampilkan formulir interaktif dengan komponen input teks untuk username, email, dan password, serta selectbox untuk memilih peran (role) antara *User* atau *Admin*. Terdapat juga komponen *multiselect* yang menampilkan seluruh daftar entitas *(EntityID)* yang diambil secara real-time dari tabel *pxentity*. Pengguna yang ditambahkan dapat diberikan akses ke satu atau beberapa entitas sesuai

kebutuhan. Proses penyimpanan data pengguna dilakukan dengan pengamanan melalui hashing password menggunakan *library bcrypt*. Password yang dimasukkan pengguna akan dienkripsi terlebih dahulu sebelum disimpan ke tabel *userlogin*. Hal tersebut penting untuk menjaga keamanan kredensial dan menghindari penyimpanan password dalam bentuk teks asli. Setelah data user berhasil dimasukkan ke tabel *userlogin*, sistem secara otomatis akan mengambil *id* user yang baru ditambahkan. ID tersebut digunakan sebagai foreign key untuk menyimpan data entitas ke dalam tabel *entitymanagement*. Dengan cara yang dilakukan, setiap pengguna akan memiliki relasi dengan satu atau lebih entitas sesuai yang dipilih saat proses penambahan. Selain itu, fitur tersebut juga menggunakan *st.session_state* pada *Streamlit* untuk menyimpan sementara input yang dimasukkan pengguna. Tujuannya adalah agar data tidak hilang ketika terjadi rerun pada halaman. Setelah form berhasil di kirim. Sistem akan mengatur *session_state["add_form_submitted"] = True* yang menandakan bahwa proses berhasil dan seluruh field form akan dibersihkan secara otomatis melalui *st.rerun()*.




*Gambar 3. 15 Tampilan tambah pengguna*

```
# Insert user data
hashed_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
cursor.execute(
    "INSERT INTO userlogin (username, email, password, role) VALUES (%s, %s, %s, %s)",
    (username, email, hashed_password.decode('utf-8'), role)
)
conn.commit()
```

*Gambar 3. 16 Function hashed password*

```
if "add_username" not in st.session_state:
    st.session_state["add_username"] = ""
if "add_email" not in st.session_state:
    st.session_state["add_email"] = ""
if "add_password" not in st.session_state:
    st.session_state["add_password"] = ""
if "add_role" not in st.session_state:
    st.session_state["add_role"] = ""
if "add_selected_entities" not in st.session_state:
    st.session_state["add_selected_entities"] = []
```

*Gambar 3. 17 Code untuk tampilan field*

### 3.2.3.2 Update Pengguna

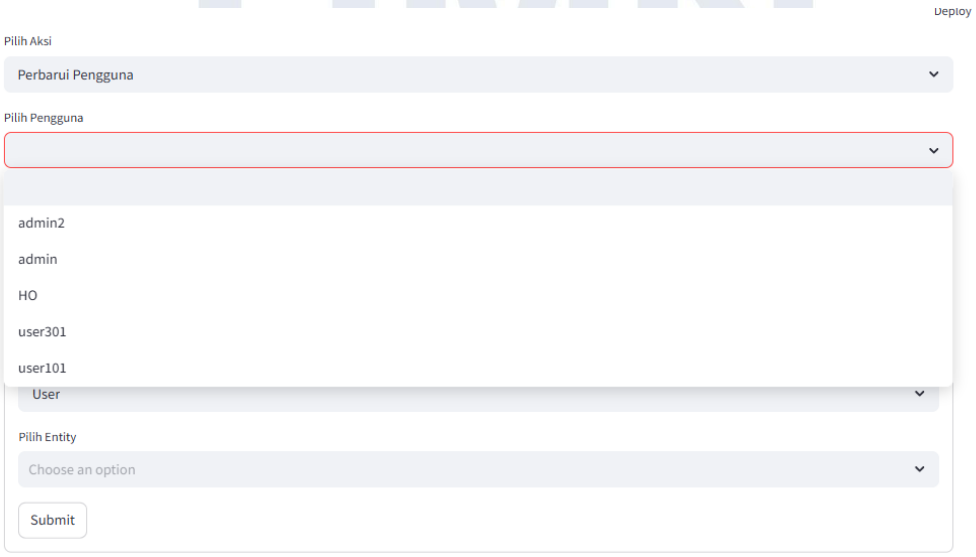
Pada menu Perbarui Pengguna, admin bisa memilih user yang ingin diperbarui informasinya. Form input yang muncul otomatis menampilkan data yang sudah tersimpan di database sebelumnya sebagai nilai default. Admin dapat memperbarui email, password, peran, dan entitas yang terkait dengan user di bagian jika ingin diubah. Sesuai dengan perubahan, sistem akan memperbarui data di tabel *userlogin* dan tabel *entitymanagement*. Penggunaan *st.session_state* juga diterapkan untuk menyimpan status form dan menjaga agar data tetap tersedia selama proses edit berlangsung. Hal tersebut penting untuk menghindari kehilangan input apabila halaman mengalami rerun. Fitur tersebut memegang peran krusial dalam manajemen akun pengguna, khususnya ketika terjadi perubahan struktur organisasi, pergantian tugas, atau restrukturisasi akses yang memerlukan penyesuaian hak akses pengguna secara real-time dan aman.

```
elif action == "Perbarui Pengguna":
    if st.session_state.get("reset_user_update", False):
        st.session_state["reset_user_update"] = False
        st.rerun()
```

*Gambar 3. 18 Code perbarui pengguna*


*Gambar 3. 19 Tampilan perbaharui pengguna 1*

Gambar dibawah merupakan gambar saat Admin memilih *Pilih Pengguna*.




*Gambar 3. 20 Tampilan perbarui pengguna 2*

Gambar dibawah merupakan gambar saat Admin memilih User dibagian *Pilih Pengguna*, Maka semua field akan terisi kecuali password (Optional) dan Admin bisa menghapus/menambah Entity sesuai dengan keperluan User.

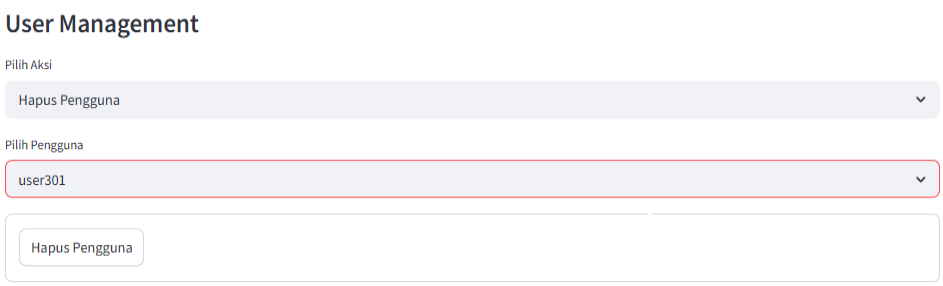

*Gambar 3. 21 Tampilan perbarui pengguna 3*

### 3.2.3.3 Hapus Pengguna

Pada bagian kode tersebut, fitur penghapusan pengguna diimplementasikan menggunakan Streamlit sebagai antarmuka dan PostgreSQL sebagai basis data. Ketika aksi "Hapus Pengguna" dipilih, sistem menampilkan dropdown berisi daftar nama pengguna yang diambil dari DataFrame df_users. Dropdown juga menyertakan opsi kosong di awal sebagai langkah validasi agar tidak langsung melakukan penghapusan tanpa seleksi. Setelah pengguna dipilih dari dropdown, ditampilkan sebuah form yang berisi tombol "Hapus Pengguna". Ketika tombol ditekan, sistem memeriksa apakah pengguna telah dipilih. Jika belum, peringatan akan ditampilkan agar pengguna memilih terlebih dahulu sebelum melanjutkan proses.

```python
# ==================== Hapus Pengguna ====================
elif action == "Hapus Pengguna":
    user_options = [""] + df_users['username'].tolist()
    selected_user = st.selectbox("Pilih Pengguna", user_options, key="delete_user_select")

    if selected_user:
        with st.form("delete_user_form"):
            submit = st.form_submit_button("Hapus Pengguna")

            if submit:
                conn = connect_db()
                cursor = conn.cursor()
                cursor.execute("SELECT id FROM userlogin WHERE username = %s", (selected_user,))
                user_id = cursor.fetchone()[0]

                cursor.execute("DELETE FROM entitymanagement WHERE id = %s", (user_id,))
                cursor.execute("DELETE FROM userlogin WHERE id = %s", (user_id,))
                conn.commit()
                cursor.close()
                conn.close()

                st.success(f"Pengguna {selected_user} berhasil dihapus.")
                st.rerun()
```

*Gambar 3. 22 Code hapus pengguna*




*Gambar 3. 23 Tampilan hapus pengguna*

### 3.2.4 Visualisasi Data

Visualisasi data merupakan bagian penting dari proses analisis karena membantu pengguna memahami informasi yang kompleks dalam bentuk yang lebih mudah dicerna secara visual. Dalam pembuatan dashboard pembelian, berbagai jenis visualisasi digunakan untuk menampilkan data pembelian berdasarkan entitas, distributor, dan model barang yang dibeli. Semua visualisasi dibuat dengan menggunakan pustaka Plotly dan Streamlit.

#### 3.2.4.1 Data Loader

Untuk memudahkan proses pengambilan dan pengolahan data dari database PostgreSQL ke dalam aplikasi dashboard berbasis Streamlit, dibuatlah

beberapa fungsi data loader yang bertanggung jawab melakukan query data dari berbagai tabel yang relevan. Fungsi-fungsi tersebut berperan sebagai penghubung antara database dengan komponen visualisasi yang akan ditampilkan pada dashboard.

Terdapat fungsi *get_user_entity()* yang digunakan untuk mengambil daftar entitas yang dimiliki oleh masing-masing pengguna. Proses dilakukan dengan cara mengambil ID user dari tabel userlogin berdasarkan username yang sedang login, kemudian mencocokkannya dengan tabel *entitymanagement* untuk mendapatkan entitas yang sesuai. Dengan pendekatan, aplikasi dapat membatasi akses data hanya kepada entitas yang diizinkan untuk masing-masing user, sehingga aspek keamanan dan segmentasi data tetap terjaga.

```python
def get_user_entity(username):
    """Mengambil list entity pengguna berdasarkan username."""
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("""
        SELECT entity
        FROM entitymanagement
        WHERE id = (
            SELECT id FROM userlogin WHERE username = %s
        )
    """, (username,))
    results = cursor.fetchall()
    cursor.close()
    conn.close()
    if results:
        return [row[0] for row in results]
    return []
```

*Gambar 3. 24 Function get user entity code*

Fungsi utama untuk memuat data transaksi pembelian adalah fungsi load_data(). Data yang diambil dari sumber data stok terdiri dari tabel utama seperti *rptranshdr, rptransdtl, pxentity, pxperiod, initem, kelompok initem, model initem, dan vinmovement*. Data yang dimuat dalam query tersebut hanya terdiri dari periode *PeriodID* 83 hingga 94, dan query mengecualikan transaksi dengan *TransCode* bernilai 3 (Data Penjualan). Karena hanya pembelian dengan *TransCode* selain 3 (Data Penjualan) yang relevan untuk analisis pembelian.

Selanjutnya, hasil pengumpulan data diproses dalam bentuk dataframe pandas. Termasuk mengubah tipe data menjadi tanggal dan mengelompokkan periode dalam bentuk bulan-tahun. Selain itu, fungsi tersebut menyediakan fitur filter berdasarkan entitas. Dengan parameter *entity_filter*, data dapat difilter lebih dahulu sebelum dimuat ke aplikasi, sesuai dengan hak akses pengguna.


*Gambar 3. 25 import pandas*

```python
def load_data(entity_filter=None):
    """Mengambil data transaksi dengan filter berdasarkan entity dan periode PeriodID 83-94."""

    if entity_filter is not None and len(entity_filter) == 0:
        columns = ['transaction_date', 'month', 'distributor', 'entity_buyer',
                   'total_harga', 'ItemID', 'QT', 'Inventory_Unit', 'TransCode', 'TrNo']
        return pd.DataFrame(columns=columns)
```

*Gambar 3. 26 Function load data code*

Untuk memenuhi kebutuhan analisis stok, fungsi get_stock_data() disiapkan. Query yang dijalankan cukup rumit, menggunakan konsep Common Table Expression (CTE) untuk mengambil periode berjalan dan periode bulan lalu secara dinamis. Query akan menghasilkan data untuk masing-masing kombinasi ItemID dan EntityID, total penjualan bulan lalu, dan rata-rata penjualan selama dua belas bulan terakhir. Data penjualan hanya memperhitungkan transaksi dengan TransCode bernilai 3, yang menunjukkan penjualan kepada customer. Penghitungan rata-rata penjualan tahunan dilakukan dengan membatasi interval PeriodID 71–82.

```
def get_stock_data():
    """Mengambil stock data sales 1 bulan lalu dan rata-rata 12 bulan terakhir."""
    conn = connect_db()
    query = """
        WITH current_period AS (
            SELECT "PeriodID"
            FROM pxperiod
            WHERE date_trunc('month', "StartDate") = date_trunc('month', current_date)
            LIMIT 1
        ),
        last_month_period AS (
            SELECT "PeriodID"
            FROM pxperiod
            WHERE date_trunc('month', "StartDate") = date_trunc('month', current_date) - INTERVAL '1 month'
            LIMIT 1
        )
```

*Gambar 3. 27 Function get stock data code*

Untuk keperluan analisis penjualan, fungsi *load_sales()* mengambil data penjualan dalam format summary (rekap). Data yang diambil hanya mencakup transaksi dengan *TransCode* bernilai 3 pada periode *PeriodID* 83 hingga 94. Hasil pertanyaan memuat agregasi total nilai penjualan (*TotalSales*), yang dikelompokkan berdasarkan *periode, entity, customer, grup customer, dan area penjualan.*

```
    JOIN rptransdtl dtl ON
        rh."PeriodID" = dtl."PeriodID" AND
        rh."SiteID" = dtl."SiteID" AND
        rh."Sys" = dtl."Sys"
    JOIN pxperiod p ON rh."PeriodID" = p."PeriodID"
    JOIN pxentity e ON rh."EntityID" = e."EntityID"
    JOIN arcustomer c ON rh."CustID" = c."CustID"
    JOIN arcustomergroupa cg ON c."GroupA_ID" = cg."GroupA_ID"
    JOIN arareae ca ON c."AreaID" = ca."AreaID"
    JOIN initem i ON dtl."ItemID" = i."Short_Name"
    JOIN initemgroup ig ON i."ItemGroupID" = ig."ItemGroupID"
    WHERE rh."TransCode" = 3
```

*Tabel 3. 28 Funtion load penjualan*

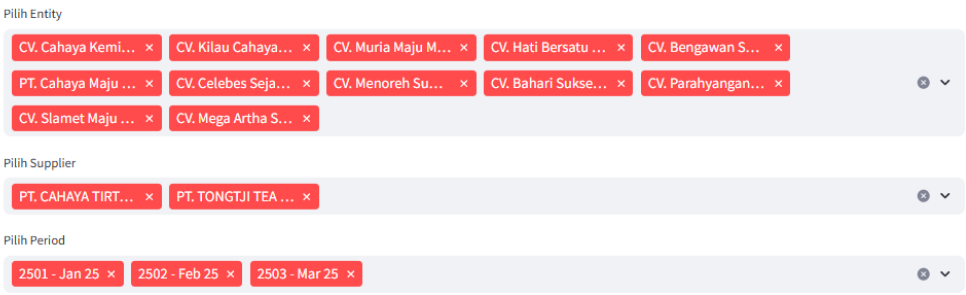
Dashboard dapat menampilkan distribusi penjualan yang lebih sederhana tetapi tetap memberi manajemen banyak informasi dengan hasil data. Setiap kali pengguna melakukan filter data di aplikasi, seluruh proses pemuatan data berjalan secara dinamis. Yang berarti dashboard selalu menampilkan data yang paling baru sesuai dengan kebutuhan analisis pengguna.

### 3.2.4.2 Visualisasi Data Pembelian Entity

Aplikasi dashboard Pembelian Entity memulai visualisasi data pembelian dengan proses filtering data, sehingga pengguna dapat menyesuaikan tampilan

data sesuai kebutuhan. Pengguna dapat memilih filter berdasarkan entitas, distributor (supplier), dan waktu di halaman pembelian.

Pada filter Entity, sistem secara otomatis menampilkan daftar entity yang tersedia berdasarkan data yang dimuat dari tabel *pxentity*. Selain itu, sistem memperhatikan hasil dari fungsi *get_user_entity()*, yang membatasi entity sesuai dengan hak akses masing-masing user, sehingga user hanya dapat memilih entity yang menjadi wewenangnya. filter distributor, sistem menggunakan tabel initemgroup yang digabungkan untuk mendapatkan daftar distributor atau supplier dari kolom distributor yang sudah dipetakan selama proses load data. Setiap distributor yang muncul di filter yang otomatis diambil dari data transaksi pembelian yang tersedia. Filter periode menggunakan mapping dari tabel pxperiod, di mana *PeriodID* diubah menjadi deskripsi periode dalam bentuk bulan-tahun—sehingga pengguna dapat dengan mudah memilih rentang periode yang ingin dianalisis tanpa memahami kode periode database. Data transaksi yang ditampilkan akan otomatis disesuaikan dengan kombinasi filter yang dipilih setiap kali pengguna mengubah pilihan filter.

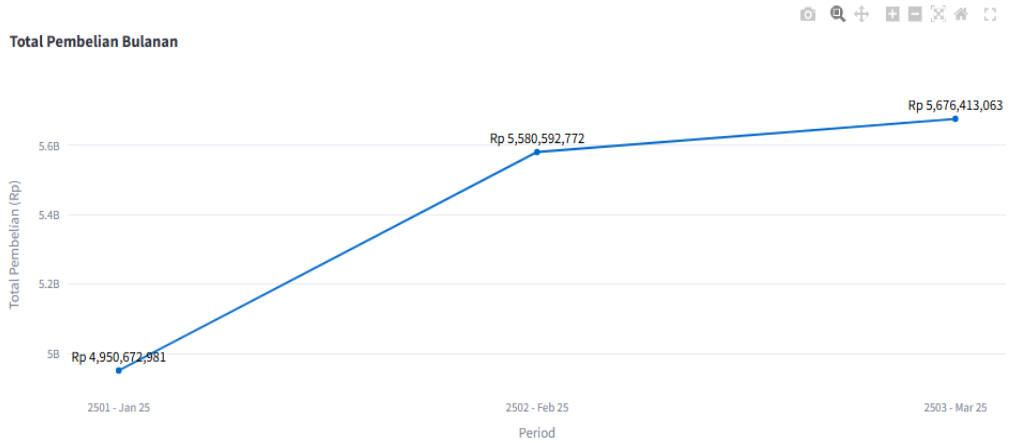



*Gambar 3. 29 Tampilan Filter dashboard pembelian entity*

Dari data yang telah difilter, hanya transaksi dengan kode "*PI*" yang dimasukkan ke dalam perhitungan. Kode "*PI*" menunjukkan bahwa transaksi tersebut merupakan pembelian, sehingga transaksi lain seperti retur atau penyesuaian tidak akan memengaruhi hasil. Setelah itu, sistem menghitung total nilai pembelian setiap bulannya berdasarkan kolom *total_harga*. Hasil penghitungan digunakan untuk membentuk garis tren dalam grafik. Garis tersebut menunjukkan fluktuasi jumlah pembelian dari bulan ke bulan, dengan sumbu X mewakili waktu (dalam format bulan dan tahun) dan sumbu Y menunjukkan total pembelian dalam satuan Rupiah. Agar lebih mudah dibaca, setiap titik pada grafik diberi label angka dalam format mata uang Rupiah, lengkap dengan pemisah ribuan, sehingga pengguna dapat langsung melihat nilai pembelian pada setiap bulannya tanpa harus melakukan interaksi tambahan dengan grafik.

```
# Line Chart untuk Total Pembelian Bulanan
monthly_grouped = (
    df_filtered.groupby('period_description')['total_harga']
    .sum()
    .reset_index()
)

monthly_grouped['harga_text'] = monthly_grouped['total_harga'].apply(lambda x: f"Rp {x:,.0f}")

# Membuat Line Chart
fig_monthly = px.line(
    monthly_grouped,
    x='period_description',
    y='total_harga',
    text='harga_text',
    markers=True,
    labels={"period_description": "Period", "total_harga": "Total Pembelian (Rp)"},
    title="Total Pembelian Bulanan"
)

fig_monthly.update_traces(
    textposition="top center",
    textfont=dict(size=14, color="black"),
    hoverinfo="skip",
    hovertemplate=None
)
st.plotly_chart(fig_monthly, use_container_width=True)

# Box ringkasan total pembelian
```

*Tabel 3. 30 Code line chart total pembelian bulanan*

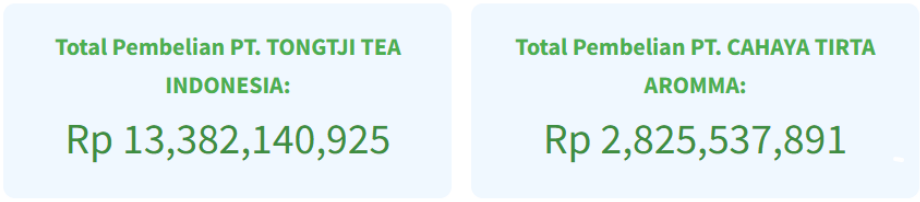



*Gambar 3. 31 Tampilan total pembelian bulanan*

Selanjutnya, kotak ringkasan untuk masing-masing distributor ditampilkan. Kotak menunjukkan jumlah total yang dibeli oleh entitas terhadap supplier tertentu. Kotak disusun secara horizontal dan dirancang untuk menjadi lebih mudah dibaca dengan warna latar dan ukuran font yang mencolok.

```
# Box ringkasan total pembelian
total_pembelian = df_filtered.groupby('distributor')['total_harga'].sum().reset_index()
total_pembelian = total_pembelian.iloc[::-1]  #\
cols = st.columns(len(total_pembelian))
for col, row in zip(cols, total_pembelian.itertuples()):
    with col:
        st.markdown(f"""
            <div style="text-align: center; padding: 20px; background-color: #f0f8ff; border-radius: 10px; font-size: 20px;">
                <span style="font-weight: bold; color: #4caf50;">Total Pembelian {row.distributor}:</span>
                <br>
                <span style="font-size: 36px; color: #3e8e41;">Rp {row.total_harga:,.0f}</span>
            </div>
        """, unsafe_allow_html=True)
```

*Gambar 3. 32 Box ringkasan Code*

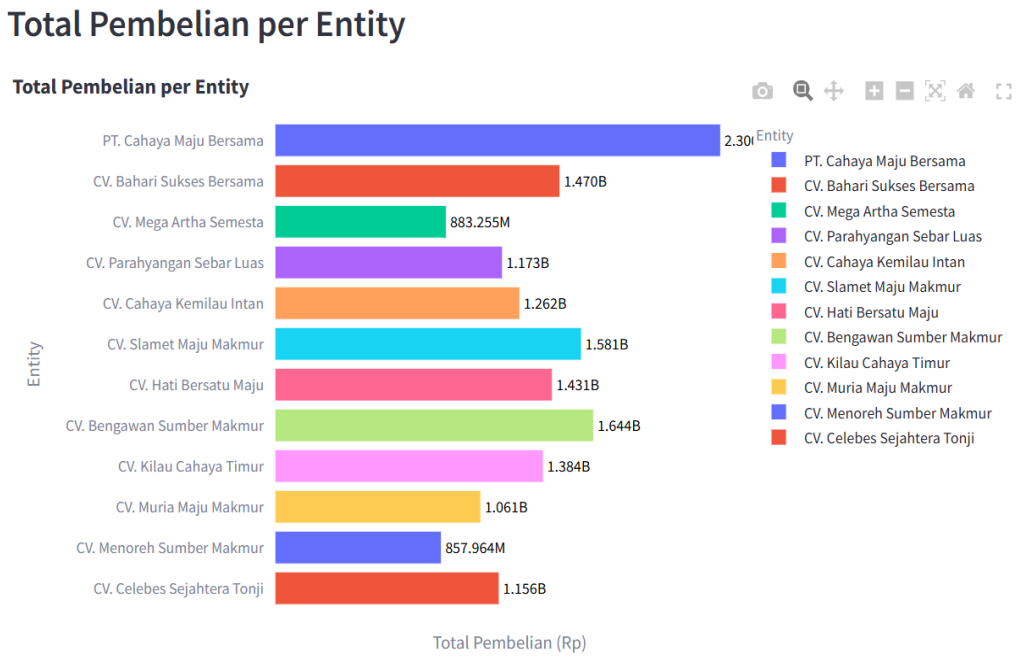



*Gambar 3. 33 Tampilan box ringkasan*

Setelah itu, pengguna dapat melihat bar chart yang menunjukkan total pembelian per entitas. Grafik tersebut menggunakan data yang sudah difilter sebelumnya, dan disusun berdasarkan kolom entity_buyer. Untuk memastikan urutan entitas sesuai dengan struktur organisasi yang ada, data disesuaikan dengan kolom *EntityID* dari tabel pxentity. Bar chart tersebut juga dilengkapi label singkat (seperti "10M", "50M", atau "1B") untuk menunjukkan besaran nilai pembelian, serta label tambahan saat pengguna mengarahkan kursor ke grafik (*hover*) yang menampilkan nilai secara lengkap dalam format Rupiah.

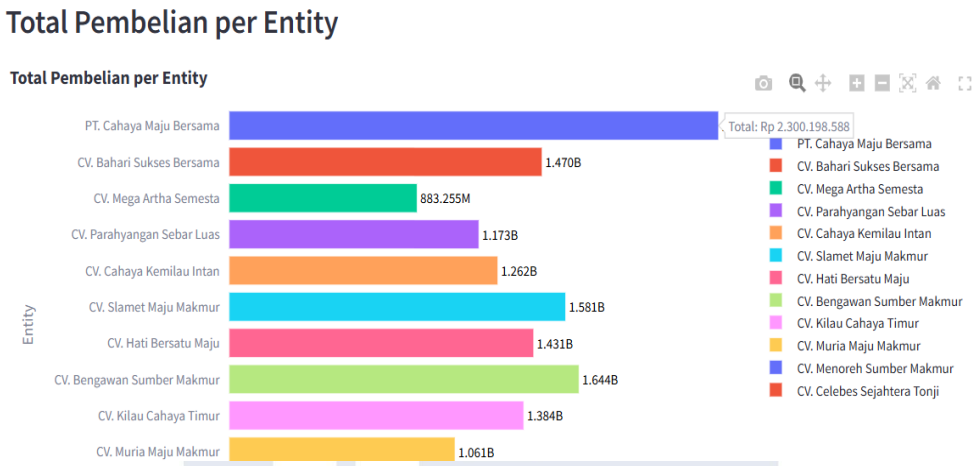
```python
def format_label_singkat(x):
    if x >= 1_000_000_000:
        return f"{int(x) // 1_000_000:,}".replace(',', '.') + "B"
    else:
        return f"{int(x) // 1_000:,}".replace(',', '.') + "M"

def format_label_detail(x):
    return f"Rp {x:,.0f}".replace(',', '.')
```

*Gambar 3. 34 Function harga text*




*Gambar 3. 35 Tampilan pembelian per entity*

Gambar di bawah merupakan gambar jika cursor di hover kearah Grafik bar.




*Gambar 3. 36 Tampilan pembelian per entity saat dihover*

Selain itu, Dashboard menampilkan tabel distribusi kuantitas item berdasarkan dua distributor utama, PT. Tong Tji Tea Indonesia dan PT. Cahaya Tirta Aromma. Tabel tersebut menampilkan jumlah item yang dibeli untuk masing-masing distributor, yang dikategorikan berdasarkan satuan unit. Kuantitas dikonversi secara otomatis sesuai dengan satuannya, misalnya, kg ditampilkan sebagai angka desimal sementara unit lain ditampilkan sebagai angka bulat.

```python
# Filter berdasarkan distributor dengan case-insensitive dan trim spasi
df_dist = df_filtered[df_filtered['distributor'].str.strip().str.upper() == distributor.upper()]

if not df_dist.empty:
    df_grouped = (
        df_dist.groupby(['ItemID', 'Inventory_Unit'])['QT']
        .sum()
        .reset_index()
    )

    df_grouped.rename(columns={'Inventory_Unit': 'Unit'}, inplace=True)
    df_grouped.index = range(1, len(df_grouped) + 1)
    df_grouped['QT'] = df_grouped.apply(
        lambda row: float(row['QT']) if row['Unit'].lower() == 'kg' else int(row['QT']),
        axis=1
    )

    st.dataframe(
        df_grouped[['ItemID', 'Unit', 'QT']],
        use_container_width=True,
        height=350
```

*Gambar 3. 37 Code Filter berdaskan distributor*


| | ItemID | Unit | QT |
|---|---|---|---|
| 1 | 1-2PRM | pcs | 550 |
| 2 | 1-2RCH | pcs | 3970 |
| 3 | 1-2RCM | pcs | 1680 |
| 4 | 1-2TT | pcs | 260 |
| 5 | 1-4PRM | pcs | 8080 |
| 6 | 1-4TT | pcs | 17540 |
| 7 | BT50 | slop | 1020 |
| 8 | BTE100 | pcs | 1850 |
| 9 | BTE100H | pcs | 500 |

| | ItemID | Unit | QT |
|---|---|---|---|
| 1 | TCH10 | pcs | 6040 |
| 2 | TCH3 | pcs | 10464 |
| 3 | TCH3C1 | pcs | 2328 |
| 4 | TFSCH1000 | pcs | 32 |
| 5 | TFSCH250 | pcs | 96 |
| 6 | TFSCH500 | pcs | 64 |
| 7 | TFSGT1000 | pcs | 120 |
| 8 | TFSGT250 | pcs | 416 |
| 9 | TFSGT500 | pcs | 128 |

*Gambar 3. 38 Tampilan pembelian quantity*

Fitur terserbut menampilkan distribusi kuantitas pembelian berdasarkan jenis model item yang tersedia di PT Tong Tji Tea Indonesia. Model item yang digunakan sebagai kategori klasifikasi dalam visualisasi tersebut mencakup empat jenis, yaitu: *Celup, Non Celup, Serbuk, dan POCI*. Klasifikasi berasal dari kolom itemmodel_description pada data transaksi pembelian. Tampilan visualisasi menggunakan layout kolom horizontal yang dibagi menjadi empat bagian, masing-masing mewakili satu kategori item model. Untuk setiap kategori, dilakukan proses penyaringan data (filtering) terhadap DataFrame yang sudah difilter sebelumnya (*df_filtered*). Proses tersebut mencocokkan nama item model dengan data yang tersedia, di mana perbandingan dilakukan secara *case-insensitive* dan mengabaikan spasi tambahan. Salah satu aspek teknis penting dalam bagian tersebut adalah pengolahan tipe data kuantitas. Jika satuan unit adalah "kg", maka nilai kuantitas akan dikonversi menjadi float untuk mencerminkan bobot desimal. Sementara jika unit bukan "kg", maka nilai akan dikonversi menjadi integer agar representasi jumlah item lebih akurat dan mudah dibaca. Hal tersebut dilakukan melalui proses iteratif menggunakan *apply()* dan lambda function. Tabel hasil akhir kemudian ditampilkan menggunakan komponen *st.dataframe()* dari Streamlit

```python
for col, item_model in zip(cols, item_models):
    with col:
        st.markdown(f"### {item_model}")
        df_model = df_filtered[df_filtered['itemmodel_description'].str.strip().str.upper() == item_model.upper()]
        if not df_model.empty:
            df_grouped = (
                df_model.groupby(['ItemID', 'Inventory_Unit'])['QT']
                .sum()
                .reset_index()
            )

            df_grouped.rename(columns={'Inventory_Unit': 'Unit'}, inplace=True)

            df_grouped.index = range(1, len(df_grouped) + 1)
            df_grouped['QT'] = df_grouped.apply(
                lambda row: float(row['QT']) if row['Unit'].lower() == 'kg' else int(row['QT']),
                axis=1
            )

            st.dataframe(
                df_grouped[['ItemID', 'Unit', 'QT']],
                use_container_width=True,
                height=350
            )
        else:
            st.info(f"Tidak ada data untuk {item_model}")
```

*Gambar 3. 39 Code quantity item model*

**Pembelian Quantity Item Model**

**Celup**

| | ItemID | Unit | QT |
|---|---|---|---|
| 1 | BTE100 | pcs | 1850 |
| 2 | BTE100H | pcs | 500 |
| 3 | BTJL | pcs | 20 |
| 4 | BTPRM | pcs | 20 |
| 5 | BTSCH | pcs | 15000 |
| 6 | CL100 | pcs | 6280 |
| 7 | CLBT | pcs | 85500 |
| 8 | CLBTC2 | pcs | 3000 |
| 9 | CLBTE | pcs | 4280 |

**Non Celup**

| | ItemID | Unit | QT |
|---|---|---|---|
| 1 | 1-2PRM | pcs | 550 |
| 2 | 1-2RCH | pcs | 3970 |
| 3 | 1-2RCM | pcs | 1680 |
| 4 | 1-2TT | pcs | 260 |
| 5 | 1-4PRM | pcs | 8080 |
| 6 | 1-4TT | pcs | 17540 |
| 7 | BT50 | slop | 1020 |
| 8 | GT-100 | pcs | 2000 |
| 9 | GT50 | pcs | 6100 |

**Serbuk**

| | ItemID | Unit | QT |
|---|---|---|---|
| 1 | TCH10 | pcs | 6040 |
| 2 | TCH3 | pcs | 10464 |
| 3 | TCH3C1 | pcs | 2328 |
| 4 | TFSCH1000 | pcs | 32 |
| 5 | TFSCH250 | pcs | 96 |
| 6 | TFSCH500 | pcs | 64 |
| 7 | TFSGT1000 | pcs | 120 |
| 8 | TFSGT250 | pcs | 416 |
| 9 | TFSGT500 | pcs | 128 |

**POCI**

| | ItemID | Unit | QT |
|---|---|---|---|
| 1 | PSSPC | pcs | 10 |

*Gambar 3. 40 Tampilan quantity item model*

### 3.2.3.3 Visualisasi Data Pembelian Retur

Dalam pengembangan dashboard Pembelian Retur, sistem juga menyediakan fitur visualisasi untuk memantau data pembelian retur. Fitur Tersebut memungkinkan user untuk menganalisis jumlah transaksi pembelian dan retur secara komprehensif berdasarkan beberapa parameter filter yang tersedia. Pada tahap awal, sistem memberikan beberapa filter yang terdiri dari *Entity, Supplier (Distributor)*, serta Period. Filter tersebut bertujuan agar pengguna dapat menyaring data berdasarkan kebutuhan analisis yang lebih spesifik. Mapping data period diambil langsung dari tabel *pxperiod* sehingga nama period yang muncul mengikuti deskripsi period yang sudah didefinisikan dalam database.ada tahap

awal, sistem menyediakan beberapa filter yang terdiri dari entitas, pemasok (distributor), dan waktu. Tujuan dari filter tersebut adalah agar pengguna dapat menyaring data sesuai dengan kebutuhan analisis yang lebih khusus. Nama waktu yang muncul sesuai dengan deskripsi waktu yang sudah ada dalam database karena mapping data menggunakan tabel *pxperiod*.

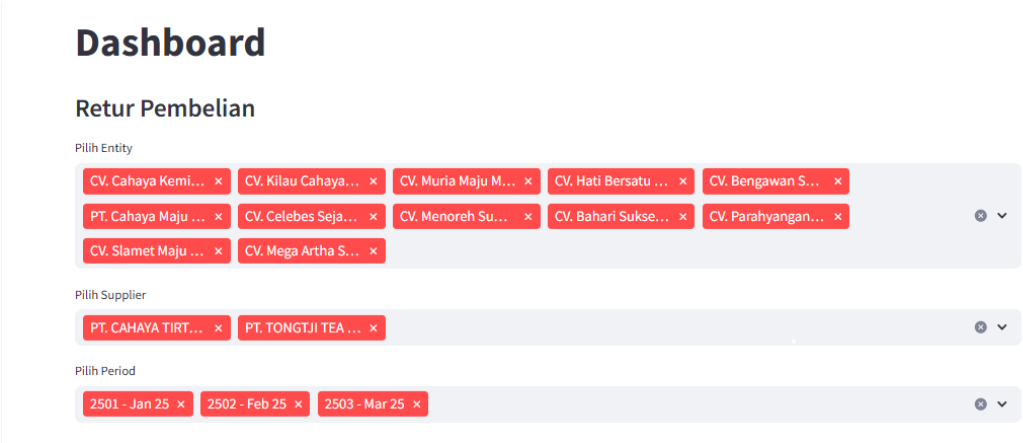
Setelah data difilter, sistem akan menggunakan kode transaksi untuk melakukan proses pemisahan transaksi. Data yang sudah terklasifikasi ditampilkan dalam beberapa bentuk visualisasi: Transaksi dengan kode *TransCode = 1* dianggap sebagai pembelian, dan *TransCode = 2* dianggap sebagai retur pembelian.

```python
def get_period_mapping():
    """Ambil data PeriodID dan Description dari tabel pxperiod"""
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute('SELECT "PeriodID", "Description" FROM pxperiod ORDER BY "PeriodID"')
    results = cursor.fetchall()
    cursor.close()
    conn.close()
    return {period_id: desc for period_id, desc in results}
```

*Gambar 3. 41 Function get period mapping*

```python
#Filter Period
period_mapping = get_period_mapping()
all_period_descriptions = [period_mapping[pid] for pid in sorted(df['PeriodID'].unique()) if pid in period_mapping]
desc_to_periodid = {desc: pid for pid, desc in period_mapping.items()}
selected_period_descriptions = st.multiselect("Pilih Period", options=all_period_descriptions, default=all_period_descriptions)
selected_periods = [desc_to_periodid[desc] for desc in selected_period_descriptions]

df_filtered = df.copy()
if selected_entities:
    df_filtered = df_filtered[df_filtered['entity_buyer'].isin(selected_entities)]
if selected_distributors:
    df_filtered = df_filtered[df_filtered['distributor'].isin(selected_distributors)]
if selected_periods:
    df_filtered = df_filtered[df_filtered['PeriodID'].isin(selected_periods)]
```

*Gambar 3. 42 Code filter dashboard*



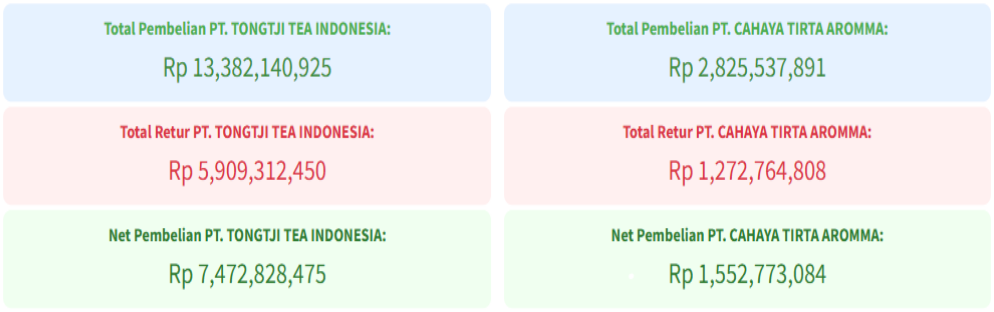

*Gambar 3. 43 Tampilan Filter retur pembelian*

Hasil agregasi Toal pembelian dan retur per bulan ditampilkan dalam bentuk line chart menggunakan *plotly.graph_objects*. Pada visualisasi Pembelian


*Gambar 3. 44 Tampilan retur dan pembelian bulanan*

Visualisasi tersebut menampilkan ringkasan total pembelian, total retur, dan nilai net pembelian untuk setiap distributor. Data dihitung dengan mengelompokkan transaksi berdasarkan kode transaksi, di mana *TransCode* bernilai 1 menandakan pembelian, dan *TransCode* bernilai 2 menandakan retur. Nilai total pembelian dan retur dijumlahkan per distributor, lalu dihitung nilai net pembelian sebagai selisih antara keduanya. Setiap distributor ditampilkan dalam satu kolom yang berisi tiga kotak informasi, yaitu total pembelian dengan latar biru, total retur dengan latar merah, dan net pembelian dengan latar hijau. Seluruh nilai diformat dalam satuan rupiah agar mudah dibaca dan dipahami. Tampilan

visual tersebut menggunakan kombinasi layout kolom *st.columns* dan elemen HTML di Streamlit.




*Gambar 3. 45 Tampilan Box pembelian retur*

Visualisasi tersebut menyajikan perbandingan total nilai pembelian dan retur dalam bentuk diagram batang yang dirancang untuk memudahkan analisis perbandingan dua jenis transaksi utama dalam proses pengadaan. Data yang digunakan merupakan hasil agregasi dari kolom *total_harga*, yang sebelumnya telah diklasifikasikan berdasarkan tipe transaksi menggunakan fungsi *classify_transaction*. Fungsi tersebut membedakan antara dua kategori transaksi utama, yaitu *Pembelian* dan *Retur*, untuk memastikan bahwa masing-masing jenis transaksi dihitung secara akurat sesuai dengan perannya dalam alur keuangan.

Setelah proses klasifikasi selesai, data dikelompokkan menggunakan metode *groupby()* berdasarkan kolom *transaction_type.* Proses tersebut menghasilkan total nilai dari masing-masing jenis transaksi, yang kemudian digunakan sebagai dasar dalam pembuatan grafik. Visualisasi dibuat menggunakan library Plotly, yang memberikan tampilan interaktif dan profesional. Setiap batang pada grafik mewakili total nilai untuk satu jenis transaksi, dengan pewarnaan yang dibedakan secara kontras: hijau digunakan untuk mewakili transaksi pembelian, sedangkan merah digunakan untuk retur. Penggunaan warna bertujuan untuk memberikan identifikasi visual yang cepat dan intuitif bagi pengguna.

Label berupa nilai transaksi ditampilkan secara langsung di atas masing-masing batang dalam format mata uang Rupiah tanpa desimal, sehingga pengguna

dapat dengan mudah membaca dan membandingkan besarnya nilai transaksi. Posisi label ditempatkan di luar batang secara vertikal agar tidak mengganggu tampilan keseluruhan dan tetap menjaga keterbacaan meskipun nilai transaksi besar atau grafik memiliki banyak data.

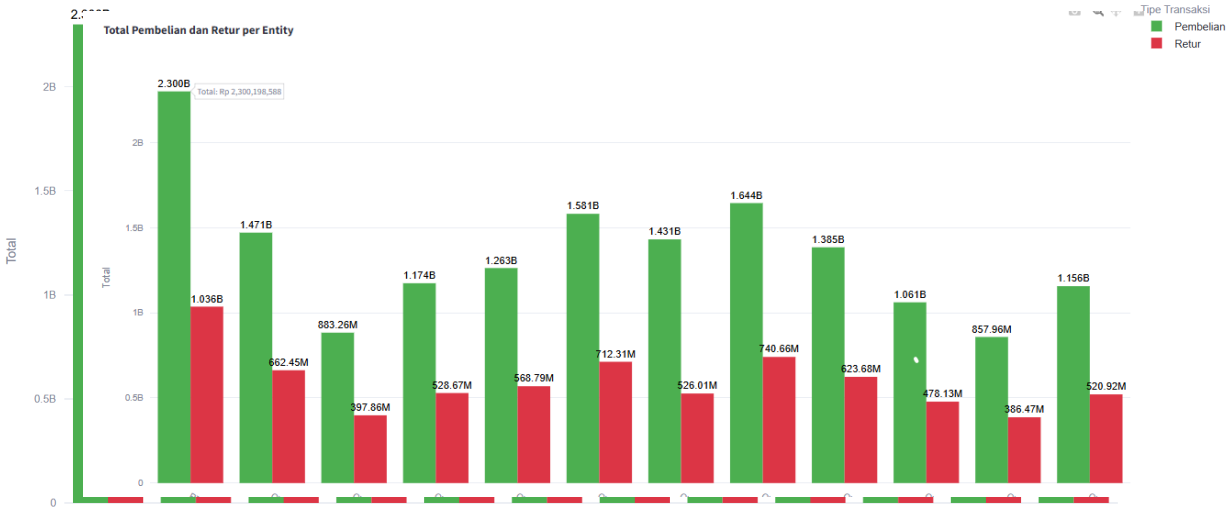
```
# Bar Chart untuk Total Pembelian dan Retur
df_filtered['transaction_type'] = df_filtered.apply(classify_transaction, axis=1)

total_by_type = df_filtered.groupby('transaction_type')['total_harga'].sum().reset_index()
total_by_type = total_by_type[total_by_type['transaction_type'].isin(['Pembelian', 'Retur'])]
```

*Gambar 3. 46 Code total pembelian dan retur*


*Tabel 3. 47 Tampilan total pembelian dan retur*

Visualisasi tersebut dirancang untuk menampilkan perbandingan nilai total pembelian dan retur berdasarkan entitas pembeli yang tercatat dalam sistem transaksi perusahaan. Tujuan utamanya adalah untuk memberikan gambaran yang lebih detail mengenai aktivitas transaksi masing-masing entitas, serta mendukung analisis performa distribusi di berbagai unit atau cabang.

Data yang digunakan diambil dari transaksi yang telah difilter dan diklasifikasikan berdasarkan tipe transaksi, yaitu “Pembelian” dan “Retur”. Selanjutnya, dilakukan agregasi nilai total_harga menggunakan metode *groupby* untuk menghitung total nominal transaksi per entitas dan jenis transaksinya. Nilai-nilai tersebut kemudian diformat ke dalam satuan mata uang rupiah agar lebih mudah dipahami. Agar nama entitas dapat ditampilkan secara informatif dalam grafik, data transaksi digabungkan dengan data referensi dari tabel *pxentity* yang memuat pasangan antara *EntityID* dan deskripsi entitas. Dengan demikian, setiap batang dalam grafik mewakili satu entitas bisnis yang telah dikenali secara formal dalam sistem. Grafik divisualisasikan menggunakan Plotly dalam bentuk diagram batang berkelompok (grouped bar chart), dengan sumbu X menampilkan nama entitas dan sumbu Y menunjukkan total nominal pembelian dan retur. Masing-masing jenis transaksi dibedakan dengan warna yang kontras: hijau untuk pembelian dan merah untuk retur. Di atas setiap batang, terdapat label nilai transaksi dalam format rupiah tanpa desimal untuk memberikan informasi yang jelas dan langsung terlihat. Proses pengurutan entitas dilakukan berdasarkan *EntityID* agar urutan visual mencerminkan struktur organisasi atau urutan cabang yang konsisten. Apabila entitas tidak dapat diurutkan secara numerik karena format data, maka sistem tetap menampilkan seluruh data secara utuh tanpa mengganggu tampilan.




*Gambar 3. 48 Tampilan pembelian dan retur per entity*

```
conn = connect_db()
df_entity_map = pd.read_sql('SELECT "EntityID", "Description" FROM pxentity', conn)
conn.close()
```

*Gambar 3. 49 Code connect description dari entityid*

Visualisasi tersebut menampilkan ringkasan kuantitas pembelian dan retur berdasarkan satuan unit dan jenis produk yang dikelompokkan berdasarkan distributor. Data disusun dalam bentuk tabel interaktif yang dibagi menjadi dua kolom, masing-masing mewakili distributor utama, yaitu PT. Tong Tji Tea Indonesia dan PT. Cahaya Tirta Aromma. Tampilan tersebut memudahkan pengguna dalam membandingkan pola pembelian dan retur antar distributor dalam satu tampilan terstruktur.

Untuk masing-masing distributor, data difilter terlebih dahulu berdasarkan nama distributor yang diubah ke format huruf kapital agar pencocokan nilai menjadi konsisten. Selanjutnya, data transaksi dipisahkan menjadi dua bagian berdasarkan kode transaksi: kode 1 untuk pembelian dan kode 2 untuk retur. Masing-masing jenis transaksi kemudian dikelompokkan menggunakan kolom *ItemID* dan *Inventory_Unit* untuk menghitung total kuantitas (*QT*) yang dibeli maupun dikembalikan.

Tabel akhir yang ditampilkan memuat empat kolom utama, yaitu *ItemID, Unit, Pembelian, dan Retur,* dan ditampilkan secara interaktif menggunakan komponen *st.dataframe()* dari Streamlit. Setiap baris tabel diberikan indeks otomatis yang dimulai dari angka satu agar mudah dilacak dan dibaca oleh pengguna.

```python
# Group pembelian
df_grouped_pi = (
    df_pi.groupby(['ItemID', 'Inventory_Unit'])['QT']
    .sum()
    .reset_index()
    .rename(columns={'QT' : 'Pembelian'})
)

# Group retur
df_grouped_pn = (
    df_pn.groupby(['ItemID', 'Inventory_Unit'])['QT']
    .sum()
    .reset_index()
    .rename(columns={'QT': 'Retur'})
)
```

*Gambar 3. 50 Code pembelian quantity 2*

Pembelian Quantity

PT. TONGTJI TEA INDONESIA

| | ItemID | Unit | Pembelian | Retur |
|---|---|---|---|---|
| 1 | 1-2PRM | pcs | 550 | 275 |
| 2 | 1-2RCH | pcs | 3970 | 1985 |
| 3 | 1-2RCM | pcs | 1680 | 840 |
| 4 | 1-2TT | pcs | 260 | 130 |
| 5 | 1-4PRM | pcs | 8080 | 4040 |
| 6 | 1-4TT | pcs | 17540 | 8770 |
| 7 | BT50 | slop | 1020 | 510 |
| 8 | BTE100 | pcs | 1850 | 925 |
| 9 | BTE100H | pcs | 500 | 250 |

PT. CAHAYA TIRTA AROMMA

| | ItemID | Unit | Pembelian | Retur |
|---|---|---|---|---|
| 1 | TCH10 | pcs | 6040 | 3020 |
| 2 | TCH3 | pcs | 10464 | 5232 |
| 3 | TCH3C1 | pcs | 2328 | 1164 |
| 4 | TFSCH1000 | pcs | 32 | 16 |
| 5 | TFSCH250 | pcs | 96 | 48 |
| 6 | TFSCH500 | pcs | 64 | 32 |
| 7 | TFSGT1000 | pcs | 120 | 60 |
| 8 | TFSGT250 | pcs | 416 | 208 |
| 9 | TFSGT500 | pcs | 128 | 64 |

*Gambar 3. 51 Tampian pembelian quantity dengan ratur*

Visualisasi selanjutnya menyajikan data kuantitas pembelian dan retur yang dikelompokkan berdasarkan kategori item model, seperti *Celup*, *Non Celup*, *Serbuk*, dan *POCI*. Tujuan dari visualisasi tersebut adalah untuk menunjukkan sebaran aktivitas pembelian dan pengembalian barang berdasarkan tipe produk, sehingga dapat memberikan wawasan terhadap kinerja masing-masing kategori produk.

Setiap item model ditampilkan dalam satu kolom tersendiri menggunakan layout paralel (multi-column) yang disusun horizontal melalui komponen *st.columns()* dari *Streamlit*. Untuk setiap kategori, data difilter berdasarkan deskripsi item model dengan menyamakan nilai dalam bentuk huruf kapital dan tanpa spasi tambahan agar proses pencocokan lebih akurat. Data kemudian dibagi

menjadi dua bagian: transaksi dengan kode *TransCode = 1* untuk mencatat pembelian, dan *TransCode = 2* untuk mencatat retur. Masing-masing data transaksi tersebut dikelompokkan menggunakan kombinasi kolom *ItemID dan Inventory_Unit*, lalu dihitung total kuantitas (*QT*) yang tercatat. Hasil pengelompokan diberi label sebagai kolom *Pembelian* dan *Retur* untuk membedakan jenis transaksinya. Tabel akhir yang ditampilkan berisi informasi penting seperti *ItemID*, *Unit*, *Pembelian*, dan *Retur*. Setiap baris diberi indeks numerik otomatis untuk memudahkan identifikasi data. Tabel Quantity Item Model ditampilkan secara interaktif menggunakan *st.dataframe()* dengan tinggi tampilan yang disesuaikan agar dapat menampilkan seluruh baris secara nyaman.

```
# Filter data pembelian dan retur untuk item model ini
df_model_pi = df_filtered[
    (df_filtered['TransCode'] == 1) &
    (df_filtered['itemmodel_description'].str.strip().str.upper() == item_model.upper())
]
df_model_pn = df_filtered[
    (df_filtered['TransCode'] == 2) &
    (df_filtered['itemmodel_description'].str.strip().str.upper() == item_model.upper())
]
```

*Gambar 3. 52 Filter data pembelian dan retur code*

Pembelian Quantity Item Model

Celup

| | ItemID | Unit | Pembelian | Retur |
|---|---|---|---|---|
| 1 | BTE100 | pcs | 1850 | 925 |
| 2 | BTE100H | pcs | 500 | 250 |
| 3 | BTJL | pcs | 20 | 10 |
| 4 | BTPRM | pcs | 20 | 10 |
| 5 | BTSCH | pcs | 15000 | 7500 |
| 6 | CL100 | pcs | 6280 | 3140 |
| 7 | CLBT | pcs | 85500 | 42750 |
| 8 | CLBTC2 | pcs | 3000 | 1500 |
| 9 | CLBTE | pcs | 4280 | 2140 |

Non Celup

| | ItemID | Unit | Pembelian | Retur |
|---|---|---|---|---|
| 1 | 1-2PRM | pcs | 550 | 275 |
| 2 | 1-2RCH | pcs | 3970 | 1985 |
| 3 | 1-2RCM | pcs | 1680 | 840 |
| 4 | 1-2TT | pcs | 260 | 130 |
| 5 | 1-4PRM | pcs | 8080 | 4040 |
| 6 | 1-4TT | pcs | 17540 | 8770 |
| 7 | BT50 | slop | 1020 | 510 |
| 8 | GT-100 | pcs | 2000 | 1000 |
| 9 | GT50 | pcs | 6100 | 3050 |

Serbuk

| | ItemID | Unit | Pembelian | Retur |
|---|---|---|---|---|
| 1 | TCH10 | pcs | 6040 | 3020 |
| 2 | TCH3 | pcs | 10464 | 5232 |
| 3 | TCH3C1 | pcs | 2328 | 1164 |
| 4 | TFSCH1000 | pcs | 32 | 16 |
| 5 | TFSCH250 | pcs | 96 | 48 |
| 6 | TFSCH500 | pcs | 64 | 32 |
| 7 | TFSGT1000 | pcs | 120 | 60 |
| 8 | TFSGT250 | pcs | 416 | 208 |
| 9 | TFSGT500 | pcs | 128 | 64 |

POCI

| | ItemID | Unit | Pembelian | Retur |
|---|---|---|---|---|
| 1 | PSSPC | pcs | 10 | 5 |

*Gambar 3. 53 Tampilan quantity item model retur*

Visualisasi tersebut menampilkan perhitungan persentase retur terhadap total pembelian untuk masing-masing entitas dan periode waktu (bulanan). Tujuan dari visualisasi tersebut adalah untuk mengukur seberapa besar volume retur yang

terjadi dibandingkan dengan nilai pembelian, sehingga perusahaan dapat mengevaluasi efektivitas distribusi serta mengidentifikasi entitas dengan tingkat retur tinggi secara spesifik. Langkah pertama dalam proses adalah memisahkan data transaksi menjadi dua kelompok, yaitu pembelian *TransCode = 1* dan retur *TransCode = 2*. Setelah itu, dibuat dua pivot table terpisah yang masing-masing menghitung total nilai pembelian dan retur berdasarkan kombinasi *entity_buyer* dan *period_description*, yaitu deskripsi periode waktu transaksi (misalnya bulan dan tahun).

Kedua pivot table tersebut kemudian disinkronkan untuk hanya mengambil kolom periode yang sama, sehingga perbandingan dapat dilakukan secara valid. Persentase retur dihitung dengan membagi nilai retur terhadap nilai pembelian pada setiap entitas dan periode, lalu dikalikan dengan 100. Untuk menghindari pembagian dengan nol, nilai pembelian nol diganti dengan *NA* sebelum dilakukan perhitungan, dan hasil akhirnya diisi dengan nol pada nilai yang tidak valid atau kosong. Selain menampilkan persentase retur per periode, sistem juga menghitung rata-rata total retur untuk setiap entitas selama seluruh periode waktu. Nilai rata-rata akan disimpan dalam kolom khusus bernama Total dan ditambahkan juga satu baris rata-rata di bagian bawah tabel untuk menunjukkan rerata seluruh entitas. Untuk meningkatkan keterbacaan, nama entitas digabungkan terlebih dahulu dengan data referensi dari tabel *pxentity*, agar nama yang ditampilkan dalam tabel sesuai dengan label resmi perusahaan. Data kemudian diurutkan berdasarkan *EntityID*, dan ditampilkan menggunakan *st.dataframe()* dalam format persentase dua digit *desimal ({:.2f}%)*. Tampilan akhir dari tabel tersebut membantu pengguna dalam mengidentifikasi pola retur per entitas dan per bulan secara rinci. Entitas yang memiliki persentase retur tinggi secara konsisten dapat menjadi perhatian untuk evaluasi lebih lanjut, seperti pemeriksaan kualitas produk, ketidaksesuaian distribusi, atau pola pemesanan yang tidak efisien.

```
# Hitung rasio: Retur / Pembelian * 100
df_ratio = (pivot_retur[common_columns] / pivot_pembelian[common_columns].replace(0, pd.NA)) * 100
df_ratio = df_ratio.fillna(0)

df_ratio['Total'] = df_ratio.mean(axis=1)

conn = connect_db()
df_entity_map = pd.read_sql('SELECT "EntityID", "Description" FROM pxentity', conn)
conn.close()

df_ratio = df_ratio.reset_index().rename(columns={'entity_buyer': 'Description'})
df_ratio = df_ratio.merge(df_entity_map, on='Description', how='left')
```

*Gambar 3. 54 Code penghitungan presentase retur terhadap pembelian*

| | 2501 - Jan 25 | 2502 - Feb 25 | 2503 - Mar 25 | Total |
|---|---|---|---|---|
| PT. Cahaya Maju Bersama | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Bahari Sukses Bersama | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Mega Artha Semesta | 45.05% | 45.05% | 0.00% | 30.03% |
| CV. Parahyangan Sebar Luas | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Cahaya Kemilau Intan | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Slamet Maju Makmur | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Hati Bersatu Maju | 22.70% | 45.05% | 37.73% | 35.16% |
| CV. Bengawan Sumber Makmur | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Kilau Cahaya Timur | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Muria Maju Makmur | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Menoreh Sumber Makmur | 45.05% | 45.05% | 45.05% | 45.05% |
| CV. Celebes Sejahtera Tonji | 45.05% | 45.05% | 45.05% | 45.05% |
| Total | 43.18% | 45.05% | 40.68% | 42.97% |

*Tabel 3. 55 Tampilan presentase retur terhadap pembelian*

Secara keseluruhan, alur kode berfungsi untuk menyajikan analisis penjualan dan retur secara multi dimensi mulai dari level bulanan, distributor, entity, hingga item model, dengan memanfaatkan kombinasi antara visualisasi interaktif dan penyajian data berbasis tabel.

### 3.2.3.4 Data Visualisasi Penjualan Sales VS Stock

Visualisasi Sales VS Stock menampilkan informasi stok harian (H-1) yang dikaitkan dengan data penjualan sebelumnya, dengan tujuan utama untuk menilai kecukupan persediaan barang berdasarkan permintaan historis. Data yang

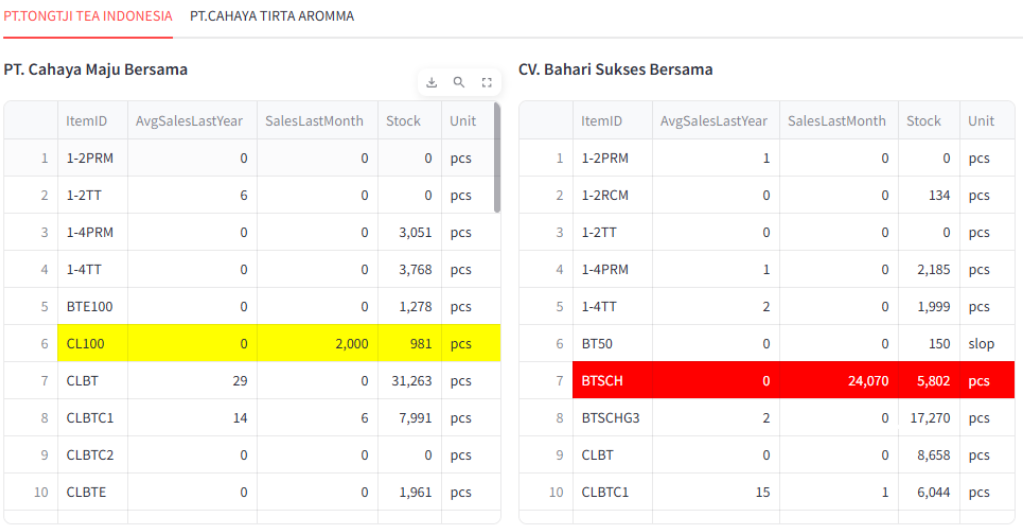
digunakan diperoleh dari fungsi *get_stock_data(),* yang memuat informasi stok terkini, rata-rata penjualan tahunan, penjualan bulan sebelumnya, satuan unit, serta entitas dan distributor yang terkait. Untuk mempermudah pembacaan dan navigasi pengguna, data visualisasi dibagi ke dalam dua tab yang mewakili dua distributor utama.

Tabel yang ditampilkan berisi informasi detail mengenai ID barang (*ItemID*), penjualan rata-rata dalam setahun (*AvgSalesLastYear*), penjualan bulan sebelumnya (*SalesLastMonth*), jumlah stok terkini (Stock), dan satuan unit barang (*Unit*). Untuk meningkatkan keterbacaan, kolom *EntityID* dihilangkan dari tampilan karena sudah direpresentasikan dalam nama entitas yang ditampilkan di atas setiap tabel. Setiap baris dalam tabel diberi penomoran indeks, dan data diurutkan berdasarkan *ItemID* agar sistematis.

Yang menjadi elemen penting dalam visualisasi tersebut adalah penggunaan format pewarnaan otomatis berdasarkan kondisi stok terhadap penjualan terakhir. Hal tersebut dilakukan melalui fungsi *highlight_stock()*, yang secara dinamis memberikan warna latar pada baris data sesuai dengan proporsi stok terhadap penjualan bulan sebelumnya. Jika nilai stok kurang dari 25 persen dari nilai penjualan bulan lalu, baris tersebut akan ditandai dengan warna merah, yang menunjukkan kondisi kritis dan kemungkinan kehabisan stok. Apabila stok berada di antara 25 hingga 50 persen dari penjualan terakhir, baris akan diberi warna kuning sebagai penanda kewaspadaan. Sebaliknya, jika stok mencukupi, maka tidak diberikan warna latar tambahan. Setiap tabel juga diformat agar angka-angka seperti penjualan dan stok ditampilkan dalam format ribuan tanpa desimal. Format tersebut bertujuan untuk memberikan kesan profesional dan memudahkan pembacaan dalam konteks bisnis. Tampilan keseluruhan disusun agar responsif dan informatif, serta memberikan gambaran cepat kepada pengguna mengenai situasi persediaan barang di masing-masing entitas distribusi.

```python
import streamlit as st
from data_loader import get_stock_data

# Penandaan warna untuk stok berdasarkan penjualan
def highlight_stock(row):
    sales = row['SalesLastMonth']
    stock = row['Stock']
    if stock < 0.25 * sales:
        return ['background-color: red; color: white'] * len(row)
    elif stock < 0.5 * sales:
        return ['background-color: yellow'] * len(row)
    else:
        return [''] * len(row)

def show_pembelian_stock(df):
    st.subheader("Stock H-1")
    df_stock = get_stock_data()
```

*Gambar 3. 56 Function hightlight color*




| | ItemID | AvgSalesLastYear | SalesLastMonth | Stock | Unit |
|---|---|---|---|---|---|
| 1 | 1-2PRM | 0 | 0 | 0 | pcs |
| 2 | 1-2TT | 6 | 0 | 0 | pcs |
| 3 | 1-4PRM | 0 | 0 | 3,051 | pcs |
| 4 | 1-4TT | 0 | 0 | 3,768 | pcs |
| 5 | BTE100 | 0 | 0 | 1,278 | pcs |
| 6 | CL100 | 0 | 2,000 | 981 | pcs |
| 7 | CLBT | 29 | 0 | 31,263 | pcs |
| 8 | CLBTC1 | 14 | 6 | 7,991 | pcs |
| 9 | CLBTC2 | 0 | 0 | 0 | pcs |
| 10 | CLBTE | 0 | 0 | 1,961 | pcs |

| | ItemID | AvgSalesLastYear | SalesLastMonth | Stock | Unit |
|---|---|---|---|---|---|
| 1 | 1-2PRM | 1 | 0 | 0 | pcs |
| 2 | 1-2RCM | 0 | 0 | 134 | pcs |
| 3 | 1-2TT | 0 | 0 | 0 | pcs |
| 4 | 1-4PRM | 1 | 0 | 2,185 | pcs |
| 5 | 1-4TT | 2 | 0 | 1,999 | pcs |
| 6 | BT50 | 0 | 0 | 150 | slop |
| 7 | BTSCH | 0 | 24,070 | 5,802 | pcs |
| 8 | BTSCHG3 | 2 | 0 | 17,270 | pcs |
| 9 | CLBT | 0 | 0 | 8,658 | pcs |
| 10 | CLBTC1 | 15 | 1 | 6,044 | pcs |

*Gambar 3. 57 Tampilan Stock H-1*

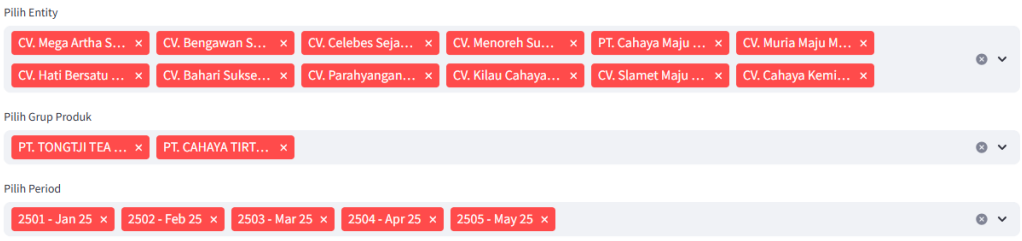
### 3.2.3.5 Viualisasi Penjualan

Bagian Visualisasi Penjualan merupakan implementasi dashboard interaktif yang dirancang untuk menampilkan data penjualan secara menyeluruh dan fleksibel. Dashboard dibangun menggunakan Streamlit sebagai antarmuka pengguna dan *Plotly* sebagai alat visualisasi. Tujuannya adalah untuk memudahkan analisis performa penjualan dari berbagai sudut, baik berdasarkan waktu, entitas perusahaan, grup produk, maupun distributor. Pengguna dapat terlebih dahulu memilih filter yang tersedia, seperti entitas, periode waktu, grup produk, dan distributor. Filter tersebut membantu menyaring data agar hanya

informasi yang relevan yang ditampilkan. Setelah data difilter, dashboard menampilkan rangkuman nilai penjualan total, grafik tren penjualan bulanan,

Filter pada dasboard berfungsi untuk melakukan proses penyaringan data sebelum ditampilkan dalam bentuk visualisasi di dashboard. Filter yang tersedia meliputi entitas perusahaan, grup produk, periode waktu, dan distributor atau supplier. Pengguna dapat memilih satu atau beberapa nilai dari setiap kategori filter melalui komponen *multiselect*, yang secara otomatis menampilkan semua opsi unik dari masing-masing kolom. Filter tersebut bersifat dinamis dan memungkinkan analisis dilakukan secara lebih spesifik sesuai kebutuhan.

Setelah pengguna memilih filter, sistem akan menerapkan kondisi penyaringan pada data utama. Proses tersebut menghasilkan DataFrame baru bernama *df_filtered*, yang hanya berisi data sesuai dengan entitas, periode, grup produk, dan distributor yang dipilih. Data hasil filter inilah yang nantinya digunakan untuk membuat grafik tren, diagram batang, serta tabel-tabel yang ditampilkan dalam dashboard. Penyaringan dilakukan bertujuan agar visualisasi yang ditampilkan menjadi lebih relevan dan fokus.




*Gamber 3.58 Tampilan Filter visualisasi penjualan*

Visualisasi tersebut menampilkan tren penjualan bulanan dalam bentuk grafik garis. Data yang digunakan merupakan hasil pengelompokan berdasarkan periode, kemudian dijumlahkan pada kolom *TotalSales*. Setiap titik pada garis merepresentasikan total penjualan pada satu periode tertentu, lengkap dengan label angka dalam format mata uang Rupiah untuk memperjelas nilainya. Tampilan dibuat menggunakan Plotly dengan desain sederhana dan profesional, serta posisi label ditempatkan di atas titik agar mudah dibaca. Visual tersebut

```python
df_line = df_filtered.groupby('Period_Desc', as_index=False)['TotalSales'].sum().sort_values('Period_Desc')
fig_line = px.line(
    df_line, x='Period_Desc', y='TotalSales', markers=True,
    title='Trend Penjualan Bulanan', template='plotly_white', text='TotalSales',
    labels={'TotalSales': 'Total Penjualan', 'Period_Desc': 'Periode'}
)
fig_line.update_traces(line=dict(width=3), marker=dict(size=8),
    texttemplate='Rp %{y:,.0f}', textposition="top center")
fig_line.update_layout(margin=dict(t=40, b=40, l=40, r=40), xaxis_tickangle=-45)
st.plotly_chart(fig_line, use_container_width=True)
```

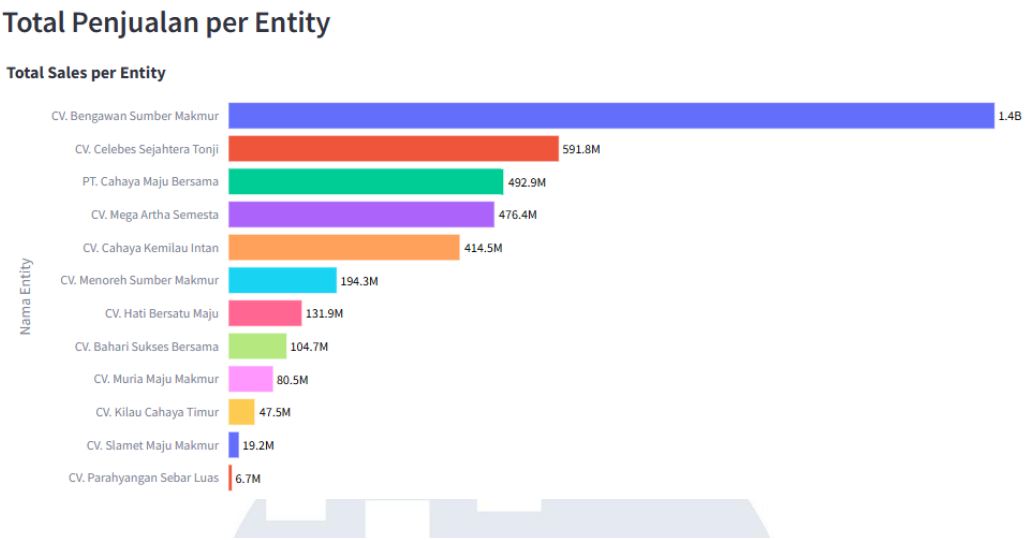
*Gambar 3.59 Code trend Penjualan*

membantu pengguna mengidentifikasi pola naik-turun penjualan secara lebih cepat dan intuitif.


*Gambar 3.60 Tampilan trend Penjualan bulanan*

Visualisasi tersebut menampilkan total penjualan berdasarkan masing-masing entitas dalam bentuk diagram batang horizontal. Data yang digunakan diperoleh dari hasil pengelompokan nilai penjualan (*TotalSales*) berdasarkan kolom *Entity_Name*, kemudian dijumlahkan agar menunjukkan akumulasi total penjualan per entitas. Setelah itu, data diurutkan dari nilai tertinggi ke terendah untuk memudahkan pembacaan.

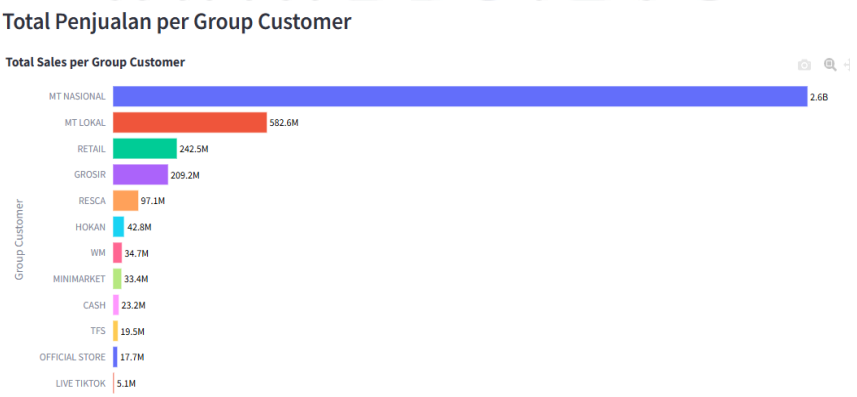
Setiap batang pada grafik merepresentasikan satu entitas, dengan panjang batang mencerminkan besarnya total penjualan. Label angka pada batang ditampilkan dalam format singkat (seperti "2.5M" untuk dua setengah juta) agar lebih ringkas dan mudah dipahami. Selain itu, ketika pengguna mengarahkan kursor ke salah satu batang, akan muncul informasi lengkap dalam format Rupiah tanpa desimal sebagai *tooltip*. Visualisasi tersebut dibuat menggunakan Plotly dengan gaya tampilan putih (*plotly_white*), serta dilengkapi dengan pengaturan margin dan ukuran font agar tampilan lebih rapi dan responsif di dalam dashboard.




*Gambar 3.61 Tampilan trend Penjualan bulanan*

Visualisasi tersebut menyajikan total penjualan berdasarkan Group Customer, yaitu segmen pelanggan yang telah dikelompokkan menurut kategori tertentu, seperti *retail, horeca, atau reseller*. Data yang ditampilkan merupakan hasil agregasi dari kolom *TotalSales*, yang dikelompokkan menggunakan kolom *GroupCustomer*. Nilai total penjualan dari masing-masing grup kemudian diurutkan dari yang tertinggi ke yang terendah untuk memudahkan perbandingan.

Grafik yang digunakan adalah diagram batang horizontal, di mana setiap batang mewakili satu kelompok pelanggan. Panjang batang menunjukkan besarnya nilai penjualan yang dikontribusikan oleh grup tersebut. Untuk memperjelas pembacaan, setiap batang diberi label singkat seperti "1.2M" atau "750K", dan informasi lebih lengkap dalam format Rupiah ditampilkan saat pengguna mengarahkan kursor ke grafik.




*Gambar 3.62 Tampilan total penjualan per group customer*

### 3.3 Kendala yang Ditemukan

Selama Melakukan kegiatan magang di PT. Tong Tji Tea Indonesia, terdapat beberapa kendala yang muncul selama kegiatan magang berlangsung. Berikut merupakan kendala yang ditemukan selama kegiatan magang:

1. Selama proses pengembangan sistem dashboard penjualan berbasis *Streamlit*, ditemukan kendala utama dalam bentuk keterbatasan kualitas data akibat penggunaan data dummy. Meskipun struktur data dummy dirancang menyerupai data rail, beberapa masalah teknis muncul dalam proses pengolahan maupun visualisasi. Permasalahan yang paling sering terjadi adalah keberadaan nilai kosong (null) pada kolom-kolom penting seperti kuantitas pembelian dan deskripsi item, sehingga terjadinya error pada importing data menuju *pgadmin 4*. Nilai null tersebut menyebabkan gangguan saat data diproses menggunakan Pandas maupun ketika divisualisasikan dalam dashboard, karena sistem tidak dapat melakukan agregasi atau rendering grafik dengan benar.
2. Permasalahan tantangan signifikan yang ditemukan berasal dari keterbatasan pemahaman terhadap alat dan teknologi yang digunakan, khususnya pgAdmin 4 sebagai antarmuka manajemen PostgreSQL dan Streamlit sebagai framework pengembangan antarmuka visualisasi data. Keduanya merupakan teknologi utama dalam pengembangan dashboard, namun tidak secara langsung dipelajari dalam kurikulum perkuliahan yang telah ditempuh sebelumnya. Ketidakterbiasaan terhadap pgAdmin 4 menyebabkan hambatan dalam memahami cara membuat dan memodifikasi tabel, mengelola relasi antar tabel, serta melakukan import data dalam format CSV ke dalam database.
3. Kendala umum yang juga ditemukan selama proses kerja adalah kesalahan penulisan kode program (error sintaks) saat membangun fitur-fitur di dalam aplikasi dashboard. Kesalahan sering terjadi saat menulis query SQL di dalam kode Python, menggunakan komponen *Streamlit* seperti *selectbox, form*, dan *session_state*, maupun ketika melakukan pemrosesan

data dengan Pandas. Error yang muncul bisa berupa variabel yang belum didefinisikan, kesalahan penamaan kolom, atau kesalahan logika pemfilteran data.

### 3.4 Solusi atas Kendala yang Ditemukan

Berikut meruapakan hal yang dilakukan agar bisa mengatasi kendala yang dihadapi selama magang berjalan:

1. Untuk mengatasi kendala akibat penggunaan data dummy, dilakukan serangkaian langkah pembersihan dan standarisasi data agar kompatibel dengan sistem pengolahan dan visualisasi. Nilai kosong pada kolom esensial dilengkapi secara manual dengan input yang konsisten secara logis, terutama pada kolom kuantitas dan deskripsi. Langkah tersebut bertujuan untuk menjaga keberlangsungan proses agregasi data dan memastikan seluruh fungsi visualisasi tetap berjalan dengan normal. Langkah berikutnya adalah melakukan normalisasi format data, termasuk mengubah seluruh teks menjadi huruf kapital, menghapus spasi yang tidak diperlukan, dan menyeragamkan istilah pada kategori seperti item model.
2. Untuk pembelajaran mandiri secara bertahap dan terstruktur. Proses dimulai dengan mempelajari dokumentasi resmi dari masing-masing platform, khususnya halaman dokumentasi PostgreSQL dan dokumentasi developer Streamlit. Materi yang dipelajari mencakup cara membuat dan mengelola tabel, menulis *query* SQL, serta melakukan koneksi antar sistem. Selain dokumentasi resmi, berbagai referensi tambahan digunakan, seperti video tutorial, artikel blog teknis, dan forum diskusi komunitas (misalnya Stack Overflow). Materi-materi tersebut membantu dalam mempercepat pemahaman fitur-fitur praktis yang belum tercakup dalam dokumentasi utama. Contohnya, cara menangani error saat import CSV ke pgAdmin 4, atau implementasi multiselect dan form submission dalam Streamlit.
3. Menerapkan kebiasaan uji coba bertahap dan debugging langsung di lingkungan pengembangan. Setiap penambahan fitur dilakukan dalam

bagian kecil terlebih dahulu, lalu diuji secara terpisah untuk memastikan tidak ada error sebelum digabungkan ke dalam sistem utama. Jika error terjadi, pesan kesalahan dari Streamlit atau Python dibaca secara detail untuk melacak sumber masalahnya.