BAB III

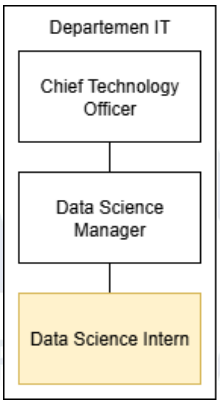
PELAKSANAAN KERJA

3.1. Kedudukan dan Koordinasi

Pada pelaksanaan kerja magang ini, penulis ditempatkan di Departemen IT (Information Technology) PT Matahari Putra Prima Tbk. Departemen ini dipimpin oleh seorang *Chief Technology Officer* (CTO) yang bertanggung jawab penuh terhadap pengembangan dan pengelolaan sistem teknologi serta infrastruktur data perusahaan. Penulis bekerja di bawah arahan langsung tim Data Science untuk mendukung inisiatif pengelolaan data perusahaan.

3.1.1. Kedudukan

Berdasarkan struktur organisasi divisi tempat penulis bekerja, penulis menempati posisi sebagai *Data Science Intern* yang berada langsung di bawah supervisi *Data Science Manager*. Posisi ini dapat dilihat secara visual pada Gambar 3.1 berikut:



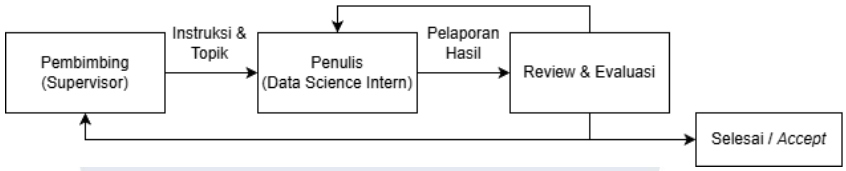
Gambar 3.1 Struktur Organisasi Divisi IT PT Matahari Putra Prima

Lingkup tanggung jawab penulis meliputi seluruh rangkaian proses analisis data, mulai dari tahap pra-pemrosesan (*data cleaning*) hingga penerapan teknik analitik tingkat lanjut. Secara spesifik, penulis bertugas melakukan analisis diagnostik untuk memahami pola perilaku

pelanggan serta mengimplementasikan algoritma *clustering* untuk keperluan segmentasi pasar.

3.1.2. Koordinasi

Koordinasi pekerjaan selama magang dilakukan dengan sistem laporan dengan *supervisor* (*Data Science Manager*), di mana segala bentuk penugasan, arahan teknis, dan evaluasi hasil pekerjaan dilakukan secara langsung antara penulis dengan *supervisor*. Alur koordinasi ini digambarkan dalam alur berikut:



Gambar 3.2 Alur Koordinasi Pekerjaan

Berdasarkan Gambar 3.2, mekanisme koordinasi pekerjaan diawali dengan tahap pemberian tugas, di mana *supervisor* memberikan instruksi kerja berupa topik analisis spesifik atau kebutuhan fitur *dashboard* yang perlu dikembangkan. Instruksi tersebut kemudian ditindaklanjuti oleh penulis pada tahap pelaksanaan teknis melalui proses eksplorasi data, *coding*, dan analisis mendalam. Setelah hasil awal diperoleh, proses berlanjut ke tahap *review* dan evaluasi untuk mempresentasikan prototipe *dashboard* atau hasil analisis tersebut. Apabila masih terdapat kekurangan, penulis diwajibkan melakukan revisi dan perbaikan model sesuai arahan mentor. Siklus koordinasi ini diakhiri dengan tahap finalisasi, di mana penulis menyerahkan dokumen final atau hasil akhir proyek kepada *supervisor* sebagai tanda bahwa tugas telah disetujui (*ACC*) dan siap untuk digunakan.

3.2 Tugas yang Dilakukan

Berisi tabel hal-hal yang penulis lakukan selama menjalankan program.

Tabel 3.1 Detail Pekerjaan yang Dilakukan

No.	Kegiatan	Timeline Pelaksanaan
1	Mengeksplorasi dan memahami data transaksi : - Mempelajari arsitektur data *Medallion* dan infrastruktur TI perusahaan. - Melakukan pembersihan data sampel transaksi harian di *Jupyter Notebook.*	1 Sept – 14 Sept 2025 (Week 1-2)
2	Membangun dan melatih model *clustering* : - Menjalankan algoritma *K-Means* dan seleksi model RFM untuk segmentasi pelanggan. - Melakukan *fine-tuning* dan migrasi kode Python ke sistem *Streamlit.*	15 Sept – 5 Okt 2025 (Week 3-5)
3	Menyusun visualisasi perilaku pelanggan dan produk : - Menganalisis kinerja tiap departemen dan SKU melalui *Product Insight*. - Memetakan pola belanja konsumen pada halaman *Behavioral Pattern*.	6 Okt – 19 Okt 2025 (Week 6-7)
4	Membuat laporan ringkasan bisnis dan performa toko : - Menghitung KPI utama seperti total pendapatan, pertumbuhan pelanggan, dan frekuensi transaksi. - Melakukan optimasi kode dan penerapan sistem caching untuk performa aplikasi.	20 Okt – 26 Okt 2025 (Week 8)
5	Mengembangkan analisis keranjang belanja (*Market Basket Analysis*) : - Menemukan pola asosiasi produk menggunakan algoritma *Apriori*. - Merancang fitur simulasi untuk strategi paket penjualan (*bundling*).	27 Okt – 2 Nov 2025 (Week 9)
6	Melakukan diagnosis mendalam pada kinerja produk : - Mengklasifikasikan status produk menggunakan metode ABC-XYZ. - Mengimplementasikan fitur *What-If Analysis* untuk kalkulasi profit.	3 Nov – 9 Nov 2025 (Week 10)

7	Melaksanakan pengujian sistem dan penyusunan dokumentasi : - Melakukan *refactoring* kode dan *stress test* pada sistem. - Menyusun dokumentasi dan manual penggunaan aplikasi secara lengkap.	10 Nov – 30 Nov 2025 (Week 11-13)
8	Memigrasikan dashboard indikator kinerja (*KPI Membership*) : - Memindahkan visualisasi pelanggan dari *Looker Studio* ke *Streamlit*. - Menerapkan sistem autentikasi keamanan dengan fungsi bawaan *Streamlit*	1 Des – 7 Des 2025 (Week 14)

3.3 Uraian Pelaksanaan Kerja

3.3.1. Proses Pelaksanaan

Pelaksanaan magang berfokus pada pengembangan sistem analitik ritel terintegrasi yang dimulai dengan pemodelan *Machine Learning* (*Clustering* & *RFM*) dan kalkulasi *Market Basket Analysis* di sisi backend. Hasil analisis tersebut kemudian dikonstruksi menjadi dashboard interaktif berbasis Streamlit yang terintegrasi dengan database *ClickHouse* untuk mendukung performa data real-time. Tahap pengembangan dilanjutkan dengan implementasi fitur tingkat lanjut berupa simulasi profit dan mekanisme caching untuk stabilitas sistem, serta ditutup dengan pengerjaan halaman tambahan berupa migrasi dashboard operasional dari *Looker Studio* ke *Streamlit*.

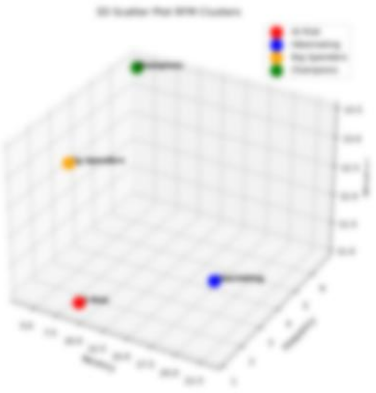
3.3.1.1 Data Exploration & Understanding

Pada awal pelaksanaan magang, difokuskan untuk memahami infrastruktur secara teknis dan konteks data yang ada di perusahaan. Terdapat beberapa tools yang cukup sering digunakan dalam pengolahan data seperti *Visual Studio Code*, *Jupyter Notebook*, maupun library *Python* seperti *pandas*, *numpy*, dan lainnya. Selain dalam hal teknis, dilakukan juga studi ataupun

proses pemahaman tentang arsitektur data *Medallion*. Pemahaman ini berfokus pada struktur data yang digunakan dalam lingkungan perusahaan sebelum memahami data aktual.

Gambar 3.3 Tampilan Sampel Data Transaksi di Jupyter Notebook

Untuk pemahaman data awal, dilakukan eksplorasi menggunakan sampel data transaksi dalam bentuk format Excel sebagai gambaran kegiatan operasional untuk lingkup waktu satu hari. Proses pengecekan data kembali dilakukan melalui Jupyter Notebook seperti pada Gambar 3.3. Dari hasil eksekusi data sampel dan melalui proses *cleaning,* ditemukan volume transaksi harian tergolong cukup besar, mencapai sekitar 600.000 baris data dengan struktur kolom yang mencakup informasi waktu, identitas pelanggan, nilai transaksi, dan sebagainya.

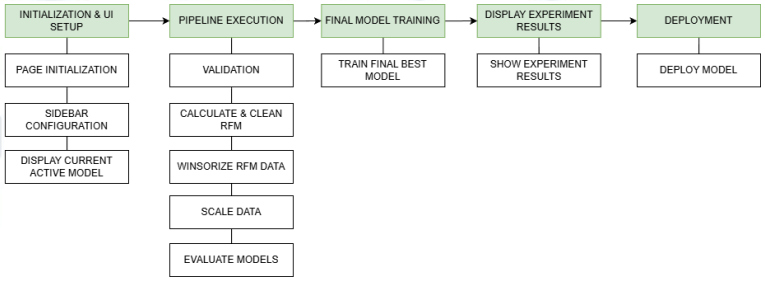


Gambar 3.4 Visualisasi Distribusi RFM Transaksi Harian

Berdasarkan pola yang ditemukan pada sampel data, analisis dilanjutkan dengan melihat distribusi perilaku belanja yang tervisualisasi pada Gambar 3.4. Grafik *RFM* (*Recency, Frequency, Monetary*) tersebut memperlihatkan variasi yang signifikan dalam frekuensi dan nominal belanja antar pelanggan yang yang jelas dalam data transaksional. Hal ini dapat digunakan menjadi informasi yang akan dikembangkan menjadi model clustering pada tahap selanjutnya.

3.3.1.2 Dashboard Model Clustering

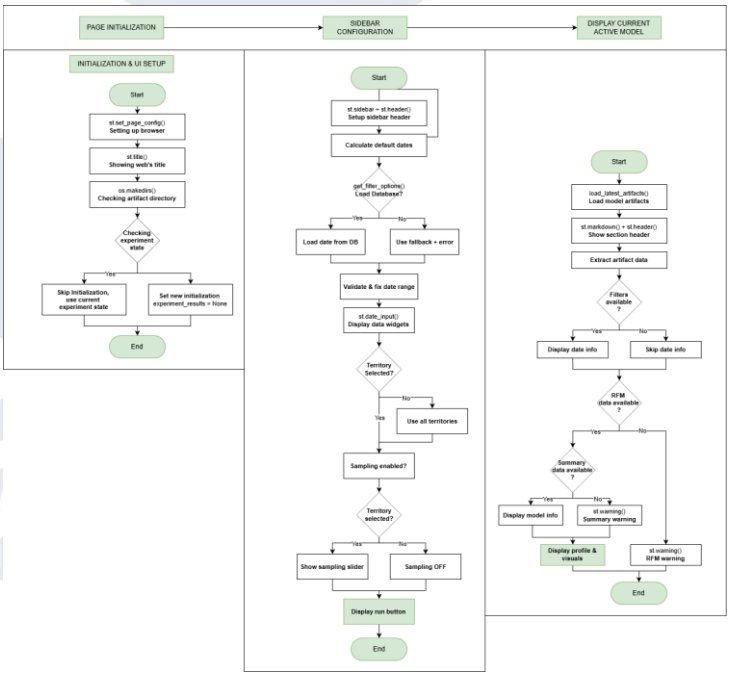
Halaman pertama ini digunakan untuk awalan sistem atau dashboard lainnya. Tujuan utamanya adalah untuk membuat segmentasi atau pengelompokan masing-masing pelanggan secara otomatis dengan memanfaatkan metode analisis RFM atau *Recency, Frequency, Monetary* dan penerapan algoritma clustering melalui *machine learning*. Secara teknis, dashboard ini berjalan pada halaman pertama atau 01_MOD_model_selection.py. Halaman ini berguna untuk para *user* atau pengguna terlebih tim teknis maupun bisnis untuk melatih dan memilih model segmentasi tanpa perlu menulis kode yang terlalu teknikal. Hasil dari segmentasi itu berguna sebagai patokan data yang digunakan untuk laporan dan simulasi lainnya.



Gambar 3.5 Alur Utama Fase Dashboard Model Clustering

Secara keseluruhan sepreti yang terlihat pada Gambar 3.5, alur halaman 1 dijalankan melalui lima fase utama. Proses diawali dengan *Initialization & UI Setup*, yaitu tahap ketika kondisi awal

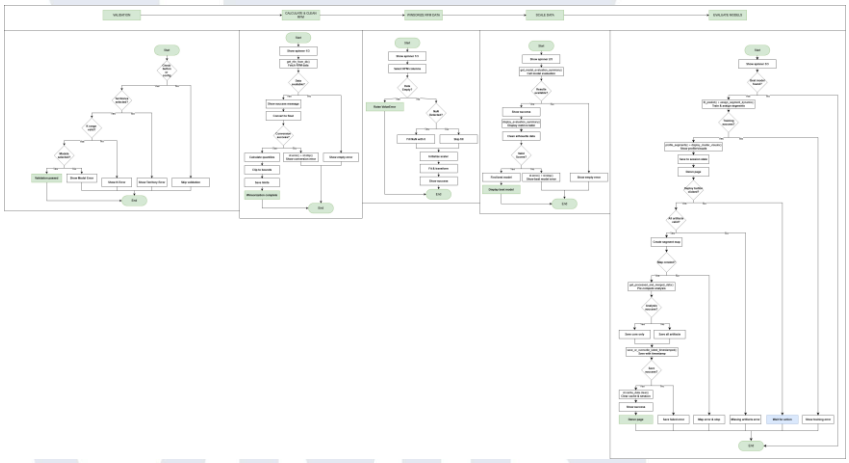
aplikasi disiapkan, konfigurasi *sidebar* dibuat, dan model aktif saat ini yang ditampilkan sebagai standar. Setelah itu, sistem memasuki *Pipeline Execution*, yang berisi rangkaian validasi data, perhitungan dan pembersihan *RFM* (termasuk proses *Winsorize* dan *Scale*), serta evaluasi berbagai kombinasi *model clustering*. *Model* terbaik yang terpilih kemudian diproses pada tahap *Final Model Training*, di mana model dilatih menggunakan data transaksi dan diberi label segmen bisnis (seperti *Champions* atau *At Risk*) melalui *function assign_segment_dynamic()*. Hasil dari *training* tersebut ditampilkan di tahap *Display Experiment Results*, dengan visualisasi dalam bentuk 2D maupun 3D. Terakhir, siklus ditutup dengan fase *Deployment*, ketika model terbaik disimpan secara permanen bersama data yang sudah melalui proses pengolahan di penyimpanan, dan siap digunakan oleh halaman laporan atau simulasi lainnya.



Gambar 3.6 Alur Penyiapan Tampilan dan Model Aktif

Pada fase *Initialization & UI Setup*, tahap awal ini, dilakukan inisialisasi ruang lingkup sistem aplikasi sesuai dengan Gambar

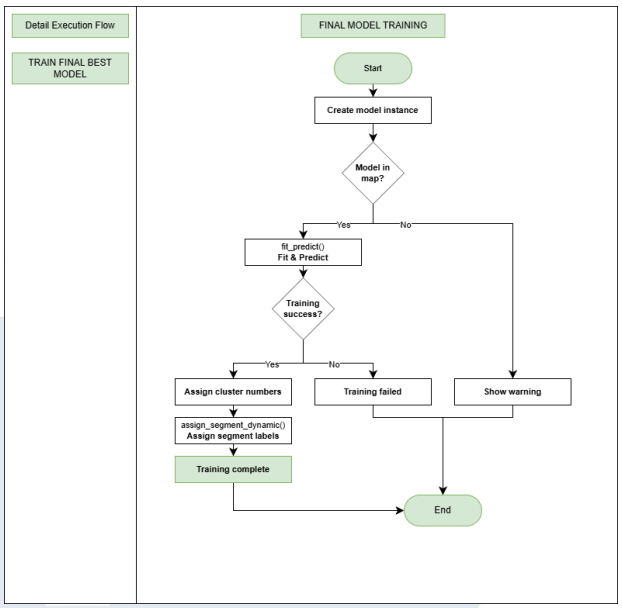
3.6. Pada tahapan pertama, sistem memuat kebutuhan seperti *library* yang dibutuhkan dan mengatur *layout* halaman agar proporsional dan mudah dilihat. Pada bagian kiri tampilan atau *sidebar*, sistem akan secara dinamis mengambil data langsung melalui *database* untuk menampilkan pilihan-pilihan *filter*. Hal ini membuat pengguna atau *user* bisa langsung memilih periode waktu, toko, maupun pilihan jenis algoritma dan metode-metode skala data untuk analisis. Sistem di awal tidak langsung membuat model baru setiap masuk ke halaman, akan tetapi menampilkan "*Current Active Model*" atau model yang sedang aktif untuk mengetahui metrik performanya, sehingga pengguna memiliki batasan atau *benchmark* sebelum melakukan eksperimen baru.



Gambar 3.7 Alur Eksekusi Pipeline dan Evaluasi Model

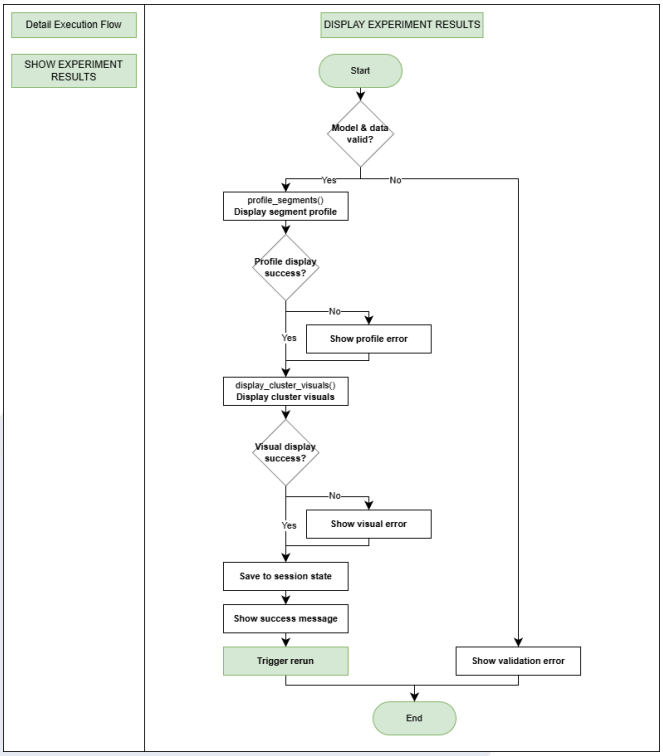
Pada fase *Pipeline Execution*, saat tombol "*Run Clustering*" ditekan, sistem akan bekerja sesuai dengan inisialisasi pada Gambar 3.7. Langkah pertama adalah melakukan validasi atau pengecekan *input* untuk memastikan pengaturan sudah benar dan aman. Setelah validasi selesai, sistem akan mengambil data transaksi dan menghitung nilai *RFM*. Supaya hasilnya akurat, data akan dibersihkan terlebih dulu dari nilai ekstrem (*Winsorizing*) dan dibuat setara dalam skalanya (*Scaling*).

Terakhir, sistem melakukan evaluasi terhadap beberapa model sekaligus untuk mencari model dengan skor yang paling bagus.



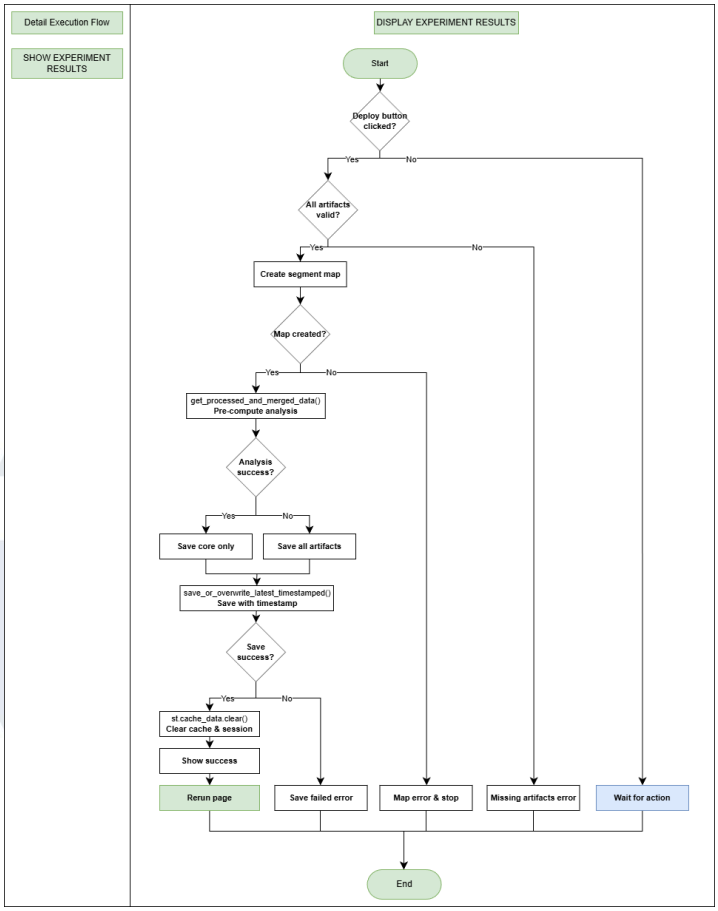
Gambar 3.8 Alur Pelatihan Model Final dan Pemberian Label

Setelah menemukan model dengan konfigurasi terbaik, di fase *Final Model Training*, sistem kemudian masuk ke tahap *training* terakhir seperti alur pada Gambar 3.8. Proses jalannya algoritma adalah dengan mempelajari pola-pola data pelanggan dan membagi ke beberapa kelompok. Dikarenakan *output* asli berupa angka-angka di dalam klaster, sistem akan menjalankan *function* khusus yakni *assign_segment_dynamic()* yang berguna untuk memberikan label atau nama yang mudah dipahami secara bisnis atau general seperti kelompok "*Champions*", "*Loyal*", atau "*At Risk*", dan sebagainya.



Gambar 3.9 Tampilan Hasil Eksperimen dan Visualisasi Sementara

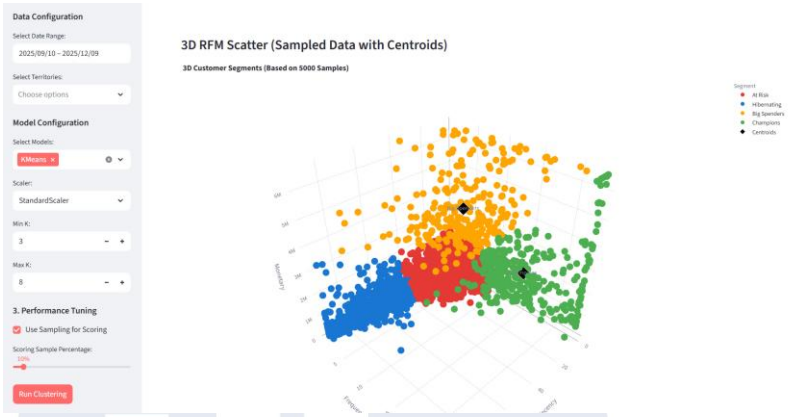
Di fase *Display Experiment Results*, sebelum model di-*deploy* atau benar-benar digunakan, pengguna dapat melihat hasil eksperimennya dengan konfigurasi terlebih dahulu yang di jelaskan di Gambar 3.9. Sistem akan memperlihatkan hasil eksperimen serta visualisasi grafik 3 dimensi yang berguna untuk memperlihatkan bagaimana persebaran antar kelompok pelanggan. Dan di tahap ini, hasil dari eksperimen konfigurasi tersebut disimpan secara sementara di *session state* atau memori di dalam aplikasi *streamlit* untuk tujuan menunggu persetujuan user dengan konfigurasi terbaru.



Gambar 3.10 Alur Proses Deployment dan Penyimpanan Model

Di fase *Deployment*, momen ini merupakan tahap akhir yang menentukan model sesuai dengan alur Gambar 3.10. Pada saat tombol *deploy* ditekan, sistem akan melakukan pengecekan terakhir untuk memastikan komponen data sudah lengkap. Setelah itu, sistem akan menggabungkan data transaksi yang sudah ada (sesuai dengan pilihan konfigurasi model) dengan label segmen pelanggan yang baru dibuat (*pre-compute analysis*). Kemudian seluruh isi dari data ini, mulai dari model, pembagian segmentasi, hingga data yang sudah diproses, akan kemudian disimpan secara permanen dalam penyimpanan dengan timestamp. Sistem kemudian melakukan refresh pada halaman dan menandakan bahwa model yang sudah di-*deploy* tersebut

menjadi model yang aktif dan dapat digunakan untuk tahapan selanjutnya.



Gambar 3.11 Tampilan Pengaturan dan Visualisasi 3D

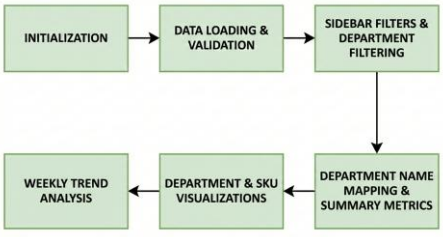
Terlihat di Gambar 3.11, memperlihatkan beberapa pilihan pengaturan pada sidebar, di mana pengguna dapat menentukan parameter model secara mandiri tanpa perlu melakukan *coding*. Terlihat juga di bagian kanan yang menampilkan hasil eksperimen yang memberi informasi tentang informasi tiap kelompok dalam bentuk grafik 3 dimensi dengan faktor RFM atau *Recency, Frequency, Monetary*

Penerapan model ini mengubah data transaksi mentah menjadi data dengan wawasan yang bisa dipakai. Contohnya, pelanggan yang masuk segmen "Champions" (yang baru belanja, sering, dan banyak) bisa dapat diberikan promo khusus. Sebaliknya, segmen "At Risk" bisa diprioritaskan untuk diajak kembali lagi melakukan aktivitas belanja. Sehingga membuat proses marketing dapat tepat secara sasaran dan membawa dampak yang lebih efisien. Data segmen tersebutlah yang akan menjadi patokan data utama untuk halaman-halaman selanjutnya untuk memberi gambaran dalam hasil penjualan.

3.3.1.3 Dashboard Product & Customer Insight

Halaman kedua ini merupakan kelanjutan dari halaman pertama dan berfokus pada fase *reporting* untuk melakukan analisis data transaksi yang sudah dikelompokkan berdasarkan segmen pelanggan. Tujuan utama dari halaman ini adalah memberikan informasi mengenai kinerja dari produk-produk yang ada dan pola perilaku pelanggan. Dashboard ini terbagi menjadi dua halaman utama yakni *04_RPT_product_insight.py* dan *05_RPT_behavioral_pattern.py*. Kedua halaman ini berfungsi sebagai alat bantu untuk membuat keputusan yang lebih baik, merencanakan strategi penjualan maupun melakukan promosi yang terfokus.

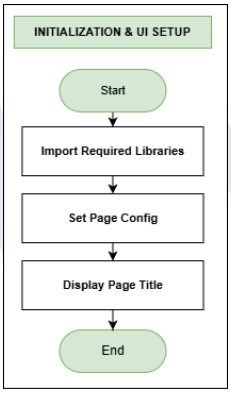
Halaman *Product Insight* atau *04_rpt_product_insight.py*. difokuskan untuk melakukan analisis dengan fokus utama pada produk dan departemen. Halaman ini bertujuan untuk membantu pengguna melakukan analisis kinerja setiap departemen dan SKU dengan menampilkan pendapatan dan penjualan, membandingkan departemen mana yang paling menguntungkan, dan melacak tren pendapatan mingguan untuk departemen dengan terbaik.



Gambar 3.12 Alur Utama Dashboard Product

Secara garis besar, alur pada Halamn *Product Insight* dijalankan melalui enam fase berurutan untuk memberikan gambaran lengkap mengenai performa produk. Proses dimulai dari tahap *Initialization*, yaitu penyiapan awal aplikasi dan

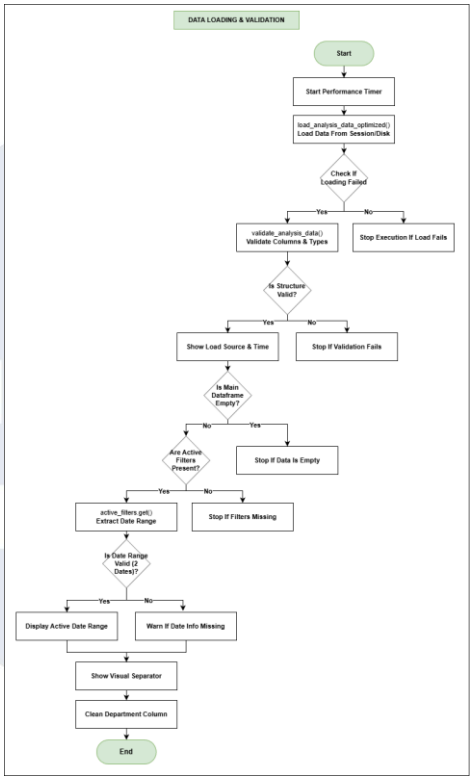
pemanggilan *library* yang diperlukan. Setelah itu, sistem masuk ke *Data Loading & Validation* untuk mengambil data dari *session state* atau *disk* sekaligus memeriksa kelengkapannya. Ketika data sudah siap, masuk ke tahap *Sidebar Filters & Department Filtering* yang membuat *filter* diterapkan, termasuk logika untuk memilih *Top N Department* berdasarkan *revenue*. Data yang sudah terkena *filter* kemudian diproses ke *Department Name Mapping & Summary Metrics*, yaitu tahap yang mengubah kode departemen menjadi nama yang mudah dipahami diikuti dengan indeks utama sepertu *Total Sales dan Total SKU*. Selanjutnya, hasil analisis ditampilkan dalam *Department & SKU Visualizations*, yang mencakup visualisasi seperti *Top SKU*, kontribusi promo, dan grafik pendapatan per departemen. Terakhir adalah fase *Weekly Trend Analysis*, yang menampilkan bagaimana pendapatan mingguan untuk departemen dengan performa yang terbaik.



Gambar 3.13 Alur Inisialisasi Halaman dan Pengaturan Awal

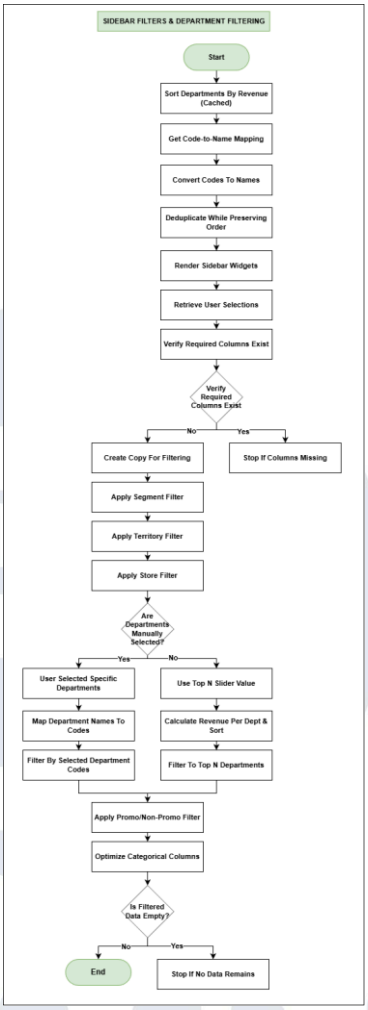
Pada fase awal *Initialization*, seperti yang terlihat di Gambar 3.13, proses dimulai dengan mengimpor *library* yang diperlukan seperti *streamlit* dan *plotly*, dilanjutkan dengan membuat nilai bawaan yang akan dilanjutkan selanjutnya diikuti dan beberapa konfigurasi untuk grafik. Fungsi st.set_page_config() kemudian dipanggil untuk konfigurasi website, termasuk pemanggilan judul

untuk halaman "Product & Department Insights" diikuti layout visualisasi sehingga bisa lebih maksimal. Di ujung fase ini, ditampilkan judul halaman utama di tampilan utama.



Gambar 3.14 Alur Pemuatan Data Teroptimasi dan Validasi Struktur

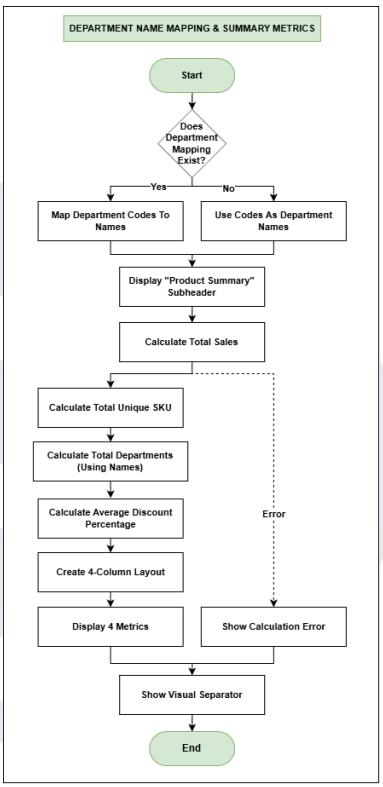
Pada fase *Data Loading & Validation*, Gambar 3.14 menunjukkan tahap pada saat sistem mengambil data yang dibutuhkan. Di bagian ini, lebih dahulu dipanggil fungsi *load_analysis_data_optimized()* untuk mencoba memuat data dari *session state*, dan jika tidak tersedia, barulah membaca dari *file* agar proses tetap cepat. Setelah data telah terbaca, fungsi *validate_analysis_data()* terpanggil untuk melakukan validasi apakah struktur datanya lengkap, termasuk kolom penting seperti *transaction_value* dan *order_id*. Apabila valid, sistem akan menampilkan waktu loading dan mengambil rentang tanggal dari filter yang tersimpan. Namun jika datanya kosong atau kolom penting hilang, proses akan langsung berhenti.



Gambar 3.15 Alur Logika Filter Sidebar dan Seleksi Top N Departemen

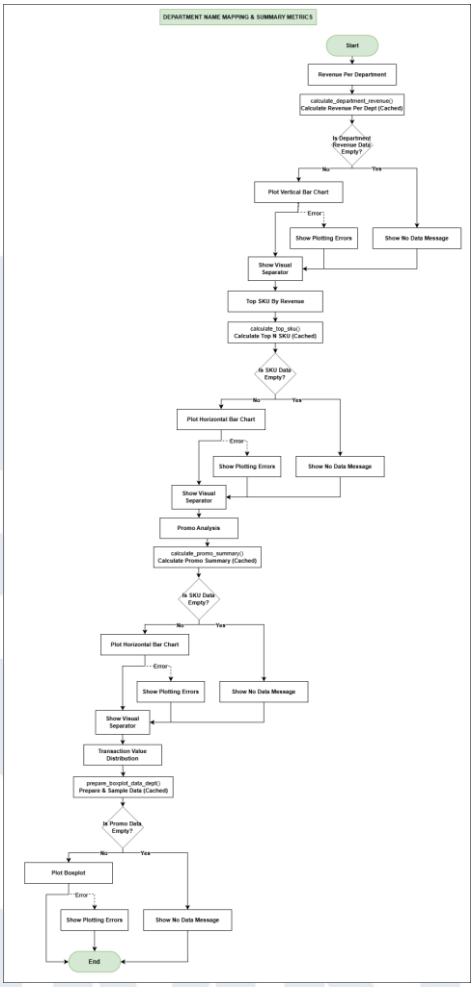
Pada fase *Sidebar Filters & Department Filtering*, terlihat di Gambar 3.15 berguna untuk mengatur pengguna melalui sidebar dengan menggunakan filter yang lebih fleksibel. Dijalankan *get_sorted_departments()* yang digunakan untuk mengurutkan daftar departemen dari pendapatan paling tinggi ke paling rendah. Setelah itu, *sidebar* menampilkan *filter* seperti wilayah, dan toko. Bagian yang membedakan fase ini adalah pada proses pemilihan departemen, apabila pengguna tidak memilih departemen secara manual, sistem akan secara otomatis menggunakan nilai dari *slider* "*Top N*" yang ada di *sidebar* untuk membatasi analisis hanya pada sejumlah departemen teratas sesuai *slider* tersebut.

Dengan begitu, fokus akan tetap pada departemen yang paling berpengaruh terhadap pendapatan atau *revenue*.



Gambar 3.16 Alur Departemen dan Perhitungan Metrik Ringkasan

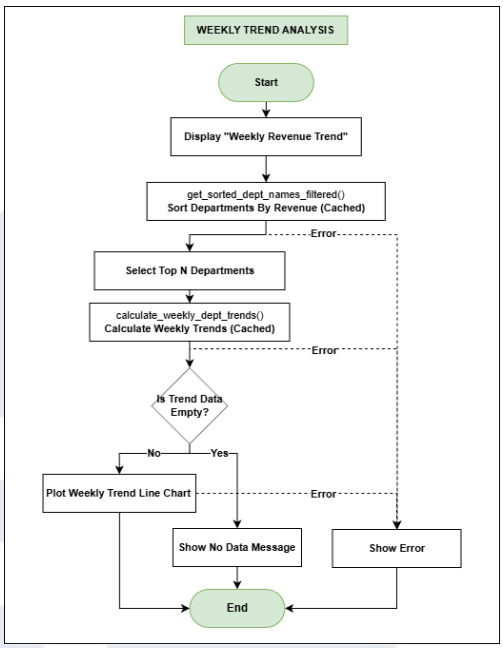
Pada fase *Department Name Mapping & Summary Metrics*, setelah data difilter seperti pada Gambar 3.16, sistem mengubah data yang sebelumnya bersifat teknis menjadi format yang lebih mudah dipahami dan mulai menghitung penilaian atau *KPI* utama. Pertama, sistem akan mengecek apakah tersedia kode departemen, jika ada, kode departemen tersebut akan diganti atau di-*mapping* dengan nama departemen yang lebih jelas. Setelah itu, empat metrik utama dihitung, mencakup total penjualan, jumlah SKU unik, jumlah departemen yang ikut dianalisis, dan rata-rata persentase diskon. Semua hasil ini kemudian ditampilkan dalam empat kolom di bagian atas dashboard sebagai bagian dari "*Product Summary*".



Gambar 3.17 Alur Visualisasi Pendapatan, SKU, dan Distribusi Nilai

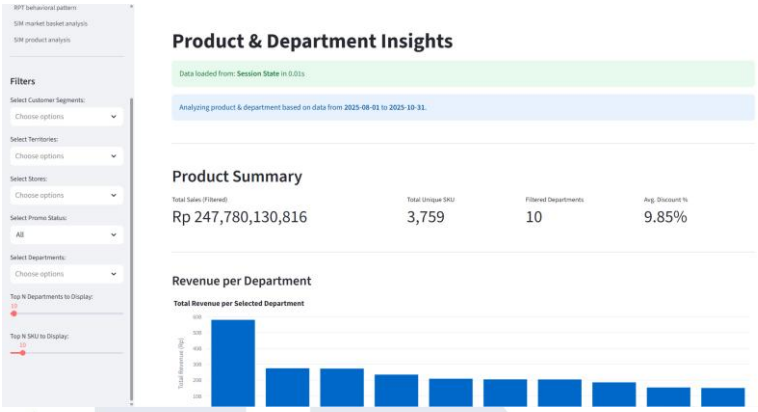
Fase *Department & Sku Visualizations*, di Gambar 3.17 ini, dijalankan empat fungsi secara berurutan untuk memberi informasi performa produk secara menyeluruh. Pertama, fungsi *calculate_department_revenue()* digunakan untuk menampilkan pendapatan per departemen dalam bentuk *bar chart*. Setelah itu, *calculate_top_sku()* berguna untuk menampilkan daftar SKU terlaris melalui *bar chart*. Kemudian, dilanjutkan dengan perbandingan kontribusi dari penjualan antara produk *promo* dan produk *non-promo* menggunakan *bar chart*. Terakhir, sistem menampilkan persebaran nilai transaksi tiap departemen

menggunakan *boxplot*, sehingga pola pembelian belanja pelanggan dapat terlihat lebih jelas.



Gambar 3.18 Alur Analisis Tren Pendapatan Mingguan per Departemen

Fase *Weekly Trend Analysis*, seperti di Gambar 3.18 merupakan fase terakhir. Ditampilkan tren pendapatan mingguan untuk melihat pola pergerakan dari waktu ke waktu. Dengan fungsi *get_sorted_dept_names_filtered(),* data kembali difilter untuk mencegah display data yang terlalu banyak. Setelah itu, *calculate_weekly_dept_trends()* dipangil untuk membuat data transaksi ke format mingguan untuk departemen yang terpilih. Hasil akhirnya ditampilkan sebagai *line chart* dengan beberapa garis, sehingga bisa dilakukan perbandingan apakah pendapatan tiap departemen sedang naik, turun, atau stabil dalam periode waktu yang sama.

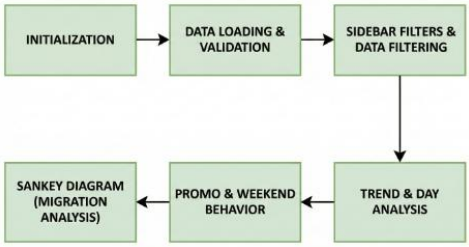


Gambar 3.19 Visualisasi Tampilan Dashboard Product Insight

Pada penerapan *dashboard Product Insight*, seperti ditunjukkan pada Gambar 3.19, difokuskan memang untuk membuat data transaksi yang sifatnya angka biasa menjadi visualisasi yang memberi informasi bagaimana kinerja suatu produk atau departemen sehingga lebih mudah dipahami. Halaman ini dapat berguna bagi tim bisnis, maupun pihak toko, untuk melihat bagaimana hasil dari setiap produk yang dijual. Analisis produk melalui berbagai visualisasi dapat memberi gambaran jelas tentang produk mana yang benar-benar mendorong pendapatan serta seberapa efektif promosi dijalankan. Di satu sisi, grafik garis untuk tren mingguan ditampilkan untuk membantu mengungkap pola musiman lebih tepat, sehingga stok untuk produk dapat disiapkan dengan lebih optimal.

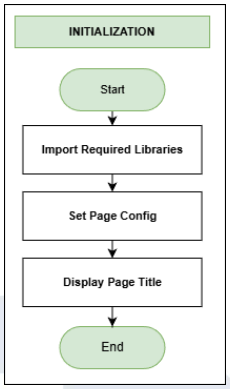
Setelah melakukan analisis pada kinerja departemen dan produk, langkah berikutnya adalah memahami siapa pelanggan yang membeli produk tersebut dan bagaimana pola belanjanya. Oleh karena itu, analisis dilanjutkan ke *05_RPT_behavioral_pattern.py*. Jika sebelumnya berfokus pada apa yang dijual (*Product*) dan seberapa besar pendapatannya (*Revenue*), maka halaman selanjutnya akan membahas tentang siapa pelanggan yang berbelanja (*Customer*) dan bagaimana mereka melakukannya (*Behavior*).

Halaman Behavioral Pattern atau *05_RPT_behavioral_pattern.py* berfokus pada analisis perilaku pelanggan untuk memahami "kapan" dan "bagaimana" pelanggan berbelanja, serta mengetahui bagaimana status mereka sebagai pelanggan dari waktu ke waktu. Halaman ini menjawab pertanyaan strategis seperti hari apa yang paling sibuk, seberapa sensitif segmen tertentu terhadap promosi, dan bagaimana pelanggan berpindah antar segmen.



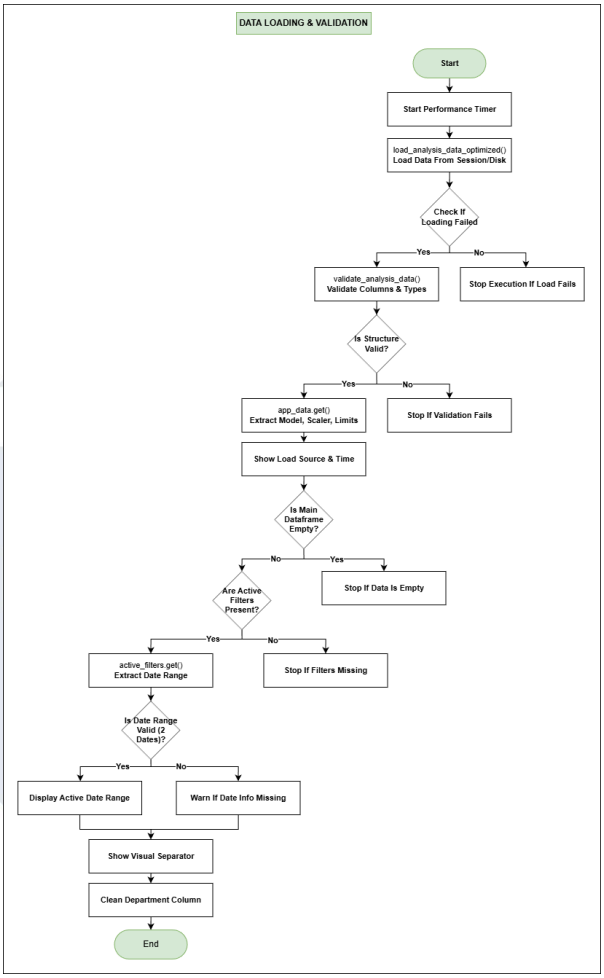
Gambar 3.20 Alur Utama Dashboard Behavioral Pattern

Seperti pada Gambar 3.20, besar, alur kerja untuk Halaman *Behavioral Pattern* ini terdiri dari enam fase utama untuk menganalisis perilaku konsumen. Proses dimulai dari fase *Initialization*, yaitu tahap penyiapan awal aplikasi. Setelah itu, sistem masuk ke *Data Loading & Validation*, yang tidak hanya memuat data transaksi tetapi juga mengambil komponen *model RFM* yang dihasilkan pada halaman pertama karena komponen ini diperlukan untuk analisis migrasi pelanggan. Dengan data tersebut, aplikasi mengaktifkan *Sidebar Filters & Data Filtering* untuk membuat analisis lebih terperinci, sesuai kebutuhan pengguna. Tahap berikutnya adalah *Trend & Day Analysis*, yang berfokus pada pola transaksi berdasarkan waktu, diikuti oleh *Promo & Weekend Behavior* untuk melakukan penilaian respons pelanggan pada promosi dan perilaku belanja di weekend. Terakhir, fase *Sankey Diagram (Migration Analysis)* yang digunakan untuk visualisasi perpindahan pelanggan antar segmen waktu ke waktu.



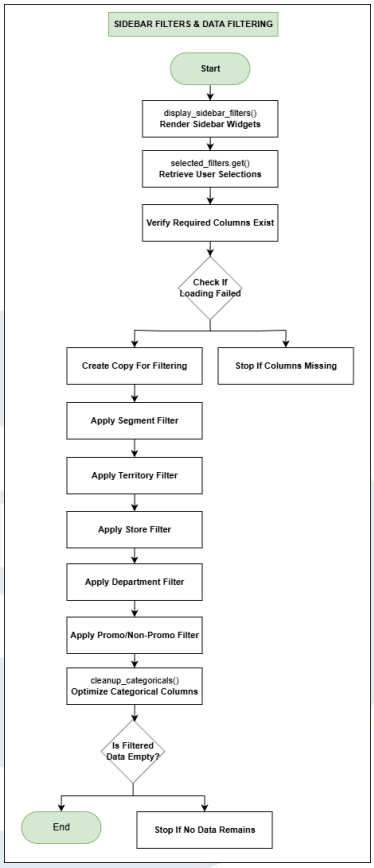
Gambar 3.21 Alur Inisialisasi Halaman Analisis Perilaku

Proses *Initialization* pada tahap awal seperti Gambar 3.21 ini dipanggil untuk menyiapkan dasar aplikasi sebelum melakukan analisis yang lebih mendalam. Pada fase ini, sistem memanggil *library* untuk visualisasi yang diperlukan seperti *plotly*, yang dipakai untuk membangun *diagram sankey*, kemudian mengatur tampilan halaman dengan judul "*Behavioral & Migration Analysis*". Dilakukan juga proses yang memastikan untuk fitur *session state* sudah siap digunakan untuk menyimpan variabel yang akan digunakan nantinya, terutama untuk rentang waktu saat menganalisis perpindahan segmen pelanggan.



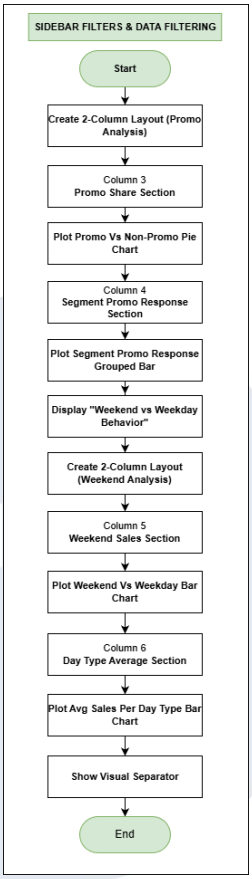
Gambar 3.22 Alur Pemuatan Data Transaksi dan Model RFM

Fase *Data Loading & Validation* di Gambar 3.22 terlihat untuk mengambil data transaksi melalui fungsi *load_analysis_data_optimized(),* kemudian diambil juga komponen model *machine learning* (seperti *scaler*, *limits*, dan *segment map*) dari *artifacts* yang tersimpan. Pengecekan kemudian dilakukan secara teliti, dan tidak hanya pada data transaksi, tetapi juga pada kolom *'Department'*. Fase ini ditandai dengan konfirmasi baik data, maupun komponen *model* segmentasi tersedia secara lengkap.



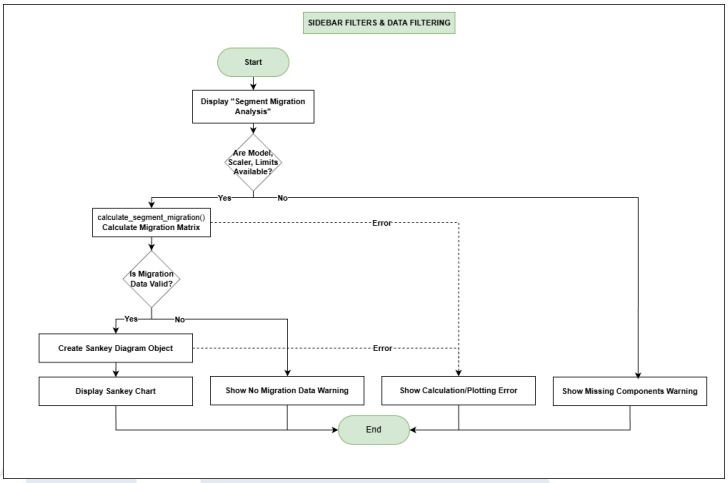
Gambar 3.23 Alur Mekanisme Filter Data dan Pembersihan Kategori

Pada *Sidebar Filters & Data Filtering*, seperti di Gambar 3.23, digambarkan fase ketika pengguna mulai menentukan analisis menggunakan *filter*. Pada tahap ini, sidebar menampilkan berbagai pilihan filter seperti wilayah, toko, departemen, dan status promo. Setelah pengguna memilih pilihan *filter* yang diinginkan, sistem akan mengambil data, kemudian menerapkan semua *filter* tersebut satu per satu. Di bagian akhir, fungsi *cleanup_categoricals()* dijalankan untuk membersihkan kategori yang sudah tidak terpakai atau kosong dari proses *filter*, sehingga visualisasi yang muncul nanti tetap akurat dan rapi.



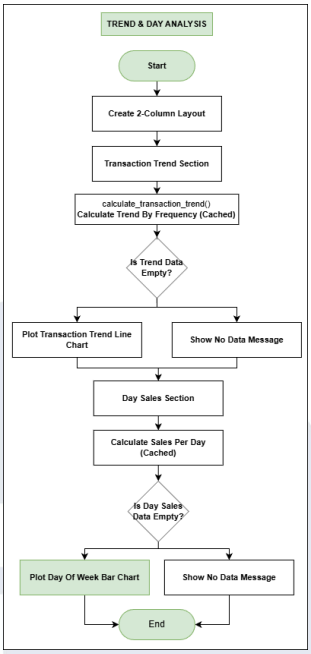
Gambar 3.24 Alur Perhitungan Tren Waktu dan Aktivitas Harian

Pada Gambar 3.24, fase *Trend & Day Analysis* digambarkan fase analisis bagi pola transaksi pelanggan menjadi dua tampilan utama. Pada kolom pertama, fungsi *calculate_transaction_trend()* digunakan untuk membuat grafik garis atau *line graph* yang menunjukkan perubahan jumlah transaksi sesuai rentang waktu yang dipilih, sehingga pola naik atau turun aktivitas belanja dapat terlihat jelas. Di kolom kedua, dijalankan fungsi *calculate_day_sales()* untuk menghitung total penjualan per-hari dalam seminggu, lalu menampilkannya dalam bentuk grafik batang atau *bar chart* harian sehingga mempermudah mengetahui tingkat belanja tertinggi.



Gambar 3.25 Alur Analisis Promo dan Perbandingan Akhir Pekan

Fase *Promo & Weekend Behavior*, pada Gambar 3.25, ini diperlihatkan bagaimana pelanggan berbelanja saat ada promo maupun saat memasuki akhir pekan atau weekend. Tampilan dibagi menjadi 3 bagian utama, Pertama, fungsi *calculate_promo_share()* menghitung seberapa besar distribusi transaksi yang berasal dari promo, lalu hasilnya ditampilkan dalam bentuk *pie chart*. Kedua, *calculate_promo_segment_response()* dipanggil untuk melihat bagaimana transaksi per-segmentasi pelanggan bereaksi pada promo, yang divisualisasikan dengan *grouped bar chart*. Ketiga, dilakukan pembandingan pola belanja antara akhir pekan (*weekend*) dan hari kerja (*weekday*), serta menampilkannya sebagai *bar chart* agar perbedaannya dapat lebih jelas dilihat.



Gambar 3.26 Alur Visualisasi Migrasi Segmentasi Pelanggan

Fase terakhir, *Sankey Diagram (Migration Analysis)* di Gambar 3.26, yang juga merupakan bagian paling kompleks, berfokus pada memvisualisasikan perpindahan segmen pelanggan dari waktu ke waktu. Dimulai dari sistem yang akan terlebih dahulu memastikan bahwa seluruh komponen model *RFM* tersedia. Dan apabila semuanya lengkap, fungsi *get_monthly_migration_data_cached()* digunakan untuk menghitung kembali posisi segmen setiap pelanggan pada berbagai bulan sesuai *filter*. Setelah itu, pengguna dapat memilih periode waktu yang ingin dianalisis, dan sistem akan menampilkan *sankey diagram* interaktif melalui *display_multi_stage_migration()*. Diagram ini menunjukkan aliran perpindahan pelanggan antar segmen dari waktu ke waktu.

Gambar 3.27 Visualisasi Tampilan Dashboard Behavioral Pattern

Halaman Behavioral Pattern berguna untuk mengetahui lebih dalam mengenai kebiasaan pelanggan, sesuatu yang tidak bisa dilihat hanya dari angka penjualan, tetapi juga dengan tampilan visualisasi seperti yang terlihat pada Gambar 3.26. Dilakukan analisis secara pola waktu, baik harian maupun mingguan, serta perbandingan antara hari kerja dan akhir pekan, dapat membantu tim operasional menyesuaikan jadwal staf maupun jam operasional toko secara lebih efisien.

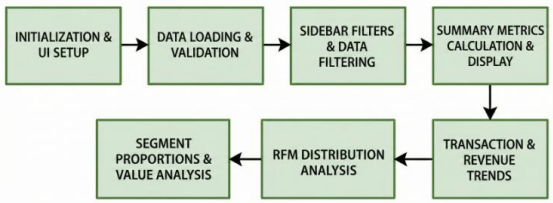
Bagian yang tergolong penting adalah *diagram sankey* yang menggambarkan perpindahan segmen pelanggan dari waktu ke waktu. Visualisasi ini menjadi alat yang sangat berharga bagi tim terkait karena memungkinkan melakukan deteksi lebih awal terhadap pelanggan yang mulai bergerak ke segmen "*At Risk*" atau yang justru meningkat menjadi "*Loyal*". Dengan informasi tersebut, strategi perusahaan dapat diterapkan dengan lebih tepat sasaran.

3.3.1.4 Dashboard Overview & Store Performance

Setelah memahami bagaimana kinerja produk dan kebiasaan dari pelanggan di halaman sebelumnya, halaman ini akan lebih difokuskan pada informasi yang sifatnya lebih *general* dan terfokuskan pada toko atau cabang operasional. *Dashboard* ini mencakup dua halaman yang terbagi ke *Dashboard Overview* atau *02_RPT_overview_distribution.py* untuk memberikan informasi mengenai kinerja bisnis secara general, dan *Dashboard*

Store Performance atau *03_RPT_store_performance.py* untuk menilai bagaimana kinerja per toko.

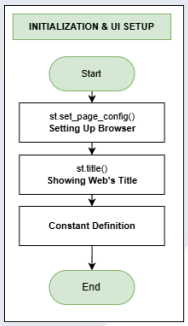
Halaman *Dashboard Overview* atau *02_RPT_overview_distribution.py* dirancang sebagai halaman yang memberikan gambaran yang lebih general pada kinerja transaksi yang terjadi. Tujuan utamanya adalah untuk mengetahui bagaimana kinerja faktor yang penting di perusahaan *(Key Performance Indicators/KPI)*, contohnya seperti total pendapatan, jumlah pertumbuhan pelanggan, serta seberapa sering transaksi terjadi. Dari metrik tersebut pula, halaman ini berguna untuk mengetahui bagaimana distribusi *RFM (Recency, Frequency, Monetary)*, yang dapat memberikan informasi mengenai fluktuasi aktivitas pelanggan pada pola transaksi atau pembelanjaan.



Gambar 3.28 Alur Utama Dashboard Behavioral Pattern

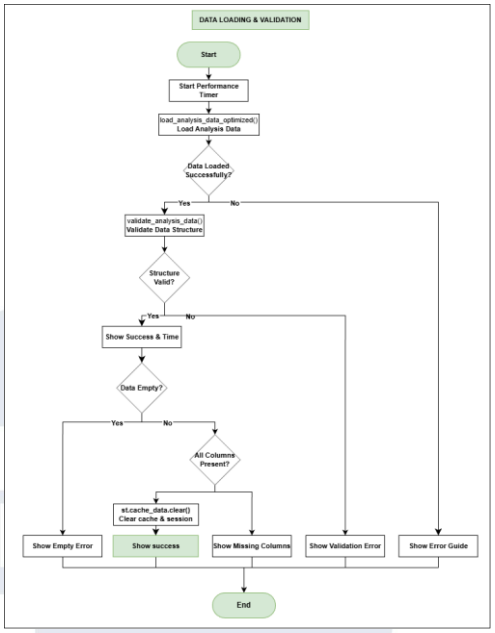
Secara garis besar seperti pada gambar 3.28, alur untuk Halaman *Dashboard Overview* dijalankan melalui tujuh fase utama yang berurutan untuk memberikan informasi tentang kinerja bisnis. Proses dimulai dari *Initialization & UI Setup*, yaitu tahap penyiapan *layout* halaman dan judul utama. Setelah itu, masuk ke *Data Loading & Validation* untuk mengakses data analisis dari *cache* atau *disk* dan memastikan bahwa struktur datanya valid. Setelah data diverifikasi, aplikasi mengaktifkan *Sidebar Filters & Data Filtering* yang berguna untuk *filtering* berdasarkan pilihan pengguna seperti segmen, wilayah, atau toko. Data yang sudah melewati *filter* kemudian diproses dalam fase

Summary Metrics Calculation & Display, yang menghitung indeks keseluruhan total dan rata-rata. Proses dilanjutkan dengan *Transaction & Revenue Trends* untuk menganalisis perkembangan transaksi dan pendapatan dari waktu ke waktu. Analisis diperluas pada fase *RFM Distribution Analysis*, yang menunjukkan sebaran kualitas pelanggan berdasarkan skor *RFM*. Siklus ini ditutup dengan *Segment Proportions & Value Analysis*, yang menampilkan komposisi segmen pelanggan serta rata-rata nilai transaksi masing-masing segmen.



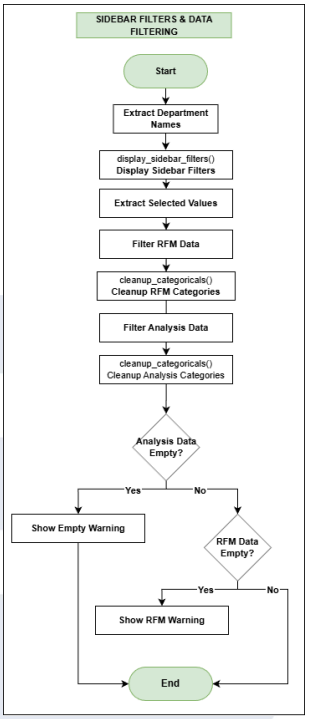
Gambar 3.29 Alur Inisialisasi Halaman dan Pengaturan Tampilan Awal

Pada fase *Initialization & UI Setup*, melalui Gambar 3.29 menunjukkan tahap pertama yang dimulai dengan sistem yang mengimpor library yang diperlukan, kemudian menjalankan *st.set_page_config()* untuk membuat tampilan *browser* ke *mode wide*. Setelah itu, judul halaman ditampilkan dan *performance timer* yang berguna untuk menghitung lama proses dijalankan dimulai untuk mencatat waktu pengisian data. Dengan langkah-langkah tersebut, halaman aplikasi akan menerima data dan interaksi pengguna dari filter yang ada.



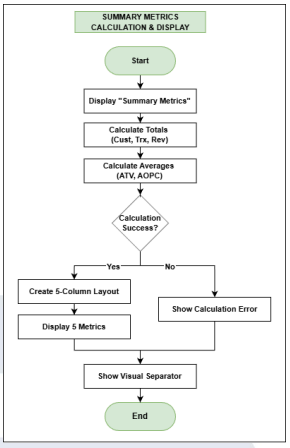
Gambar 3.30 Alur Pemuatan Data dan Validasi Struktur Data Analisis

Kemudian dari tahap pertama, fase lanjutan yaitu *Data Loading & Validation*, yang terlihat di Gambar 3.30 berfokus pada pengambilan data model yang telah dideploy dari tahapan paling pertama (*dashboard clustering*). Sistem memanggil *function load_analysis_data_optimized()* untuk mengakses data dari *session state* atau *disk*, dengan memfokuskan kecepatan melalui sisa *cache*. Data yang berhasil diambil kemudian melewati *function validate_analysis_data()* yang berguna untuk memastikan kolom dan struktur data lengkap. Jika valid, sistem menampilkan sumber pemuatan dan waktu yang dibutuhkan, namun jika terjadi kegagalan validasi atau data kosong, eksekusi akan segera dihentikan.



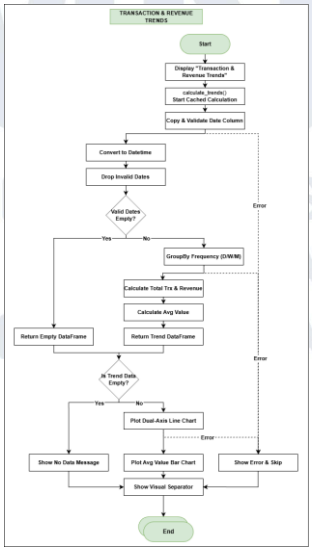
Gambar 3.31 Alur Penyaringan Data Multidimensi Melalui Sidebar

Kemudian di fase ketiga, *Sidebar Filters & Data Filtering* seperti pada Gambar 3.31, data diperkecil sesuai pilihan pengguna di sidebar. Sistem mengambil daftar nama departemen dan menampilkan filter lewat *display_sidebar_filters().* Setelah itu, data *RFM* dan data Analisis disaring satu per satu berdasarkan segmen, wilayah, dan toko yang dipilih. Ketika proses *filtering* selesai, *cleanup_categoricals()* dijalankan untuk membuang kategori yang sudah tidak terpakai agar penggunaan memori lebih efisien. Jika hasil akhirnya ternyata kosong, sistem akan menampilkan peringatan kepada pengguna. dihentikan.



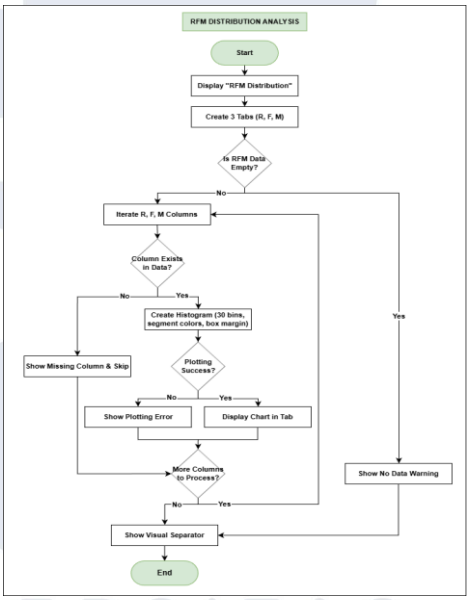
Gambar 3.32 Alur Perhitungan dan Penampilan Metrik Ringkasan Utama

Pada fase *Summary Metrics Calculation & Display*, di Gambar 3.32 dilakukan perhitungan dan ditampilkan metriks utama di bagian atas halaman. Sistem menghitung total pelanggan, total transaksi, dan total pendapatan, serta metrik rata-rata seperti rata-rata nilai transaksi dan rata-rata pesanan per pelanggan. Setelah perhitungan selesai, kelima metrik ini ditampilkan dalam lima kolom yang berbeda, sehingga bisa memberikan ringkasan mengenai kinerja bisnis yang dapat dianalisis.



Gambar 3.33 Alur Analisis Tren Pertumbuhan Transaksi dan Pendapatan

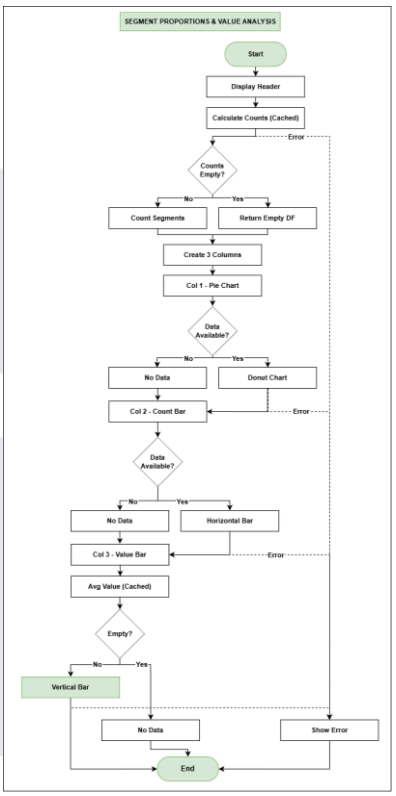
Di fase *Transaction & Revenue Trends*, ditunjukkan bagaimana performa bisnis berubah dari waktu ke waktu. Seperti di Gambar 3.33, digunakan function calculate_trends() yang sudah di-cache untuk mengelompokkan data berdasarkan dari frekuensi waktu, seperti harian, mingguan, atau bulanan. Hasilnya kemudian ditampilkan dalam dual-axis line chart, yang menampilkan total transaksi dan total pendapatan secara bersamaan. Visualisasi ini membantu untuk melihat bagaimana tren kenaikan atau penurunan baik dari segi volume maupun nilai penjualan.



Gambar 3.34 Alur Analisis Distribusi Kualitas Pelanggan (RFM)

Untuk fase *RFM Distribution Analysis*, pada Gambar 3.34 diperlihatkan bagaimana analisis ini memperlihatkan tingkat pelanggan dalam tiga bagian yang terpisah untuk Recency (R), Frequency (F), dan Monetary (M). Di setiap bagian, sistem menampilkan histogram dengan yang diberi warna sesuai segmen pelanggan. Proses ini dijalankan untuk masing-masing komponen RFM, sehingga pengguna dapat melihat sebaran kualitas

pelanggan, misalnya apakah mayoritas pelanggan masih aktif baru-baru ini atau sudah lama tidak melakukan transaksi.



Gambar 3.35 Alur Perhitungan Segmen dan Nilai Transaksi Rata-rata

Kemudian di fase *Segment Proportions & Value Analysis,* seperti Gambar 3.35 menunjukkan pembagian pelanggan dan *revenue* yang dihasilkan. Kemudian dilakukan perhitungan jumlah pelanggan per segmen dan menampilkannya dalam *chart donut* pada kolom pertama. Di kolom berikutnya, dua visualisasi tambahan ditampilkan, yaitu grafik yang menunjukkan jumlah pelanggan per segmen dan grafik yang menunjukkan rata-rata nilai transaksi per segmen. Ketiga visualisasi tersebut memberikan gambaran mengenai pembagian pelanggan serta kontribusi *revenue* yang dihasilkan oleh masing-masing segmen.

Gambar 3.36 Visualisasi Tampilan Tampilan Dashboard Overview

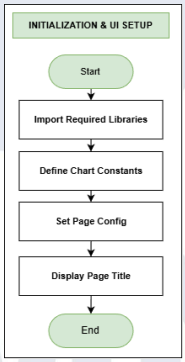
Halaman Overview ini berfungsi sebagai alat yang memberikan ringkasan bagaimana kinerja bisnis seperti yang diperlihatkan pada Gambar 3.36. mengenai visualisasi tren pendapatan dan transaksi dalam satu grafik, yang memudahkan tim terkait untuk mendeteksi pola musiman secara lebih cepat. Selain itu, distribusi metriks singkat juga memberikan informasi bagaimana persebaran pelanggan, sehingga dapat membantu memamahi kondisi bisnis dengan cepat.

Setelah memahami bagaimana kinerja bisnis secara *general* melalui Halaman *Dashboard Overview*, diperlukan pendalaman analisis dari general ke tingkat operasional masing-masing cabang. Kebutuhan ini yang tergambarkan melalui Halaman *Store Performance*, yang fokus analisis terfokuskan pada kinerja spesifik per toko dan wilayah.

Gambar 3.37 Alur Utama Dashboard Store Performance

Pada Gambar 3.37, alur kerja Halaman *Store Performance* terdiri dari enam fase yang digunakan untuk menganalisis kinerja setiap cabang. Prosesnya dimulai dari tahap *Initialization* dan *Data Loading & Validation*, yaitu ketika aplikasi membangun

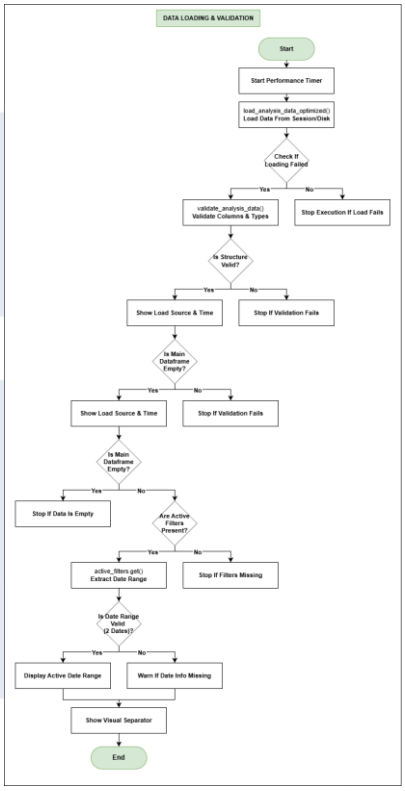
dan mempersiapkan ruang lingkup aplikasi dan mengakses data transaksi yang sudah dikelompokkan dari halaman pertama. Setelah datanya siap dipakai, fase *Sidebar Filters & Data Filtering* membuat pengguna dapat melakukan *filtering* data berdasarkan wilayah dan toko tertentu. Bagian utama ada pada fase *Store Aggregation*, di mana terjadi proses perhitungan metrik per cabang seperti pendapatan, jumlah transaksi, dan nilai transaksi rata-rata. Hasilnya kemudian divisualisasikan pada fase *Trend & Key Metrics Visualizations* melalui grafik tren dan *treemap* untuk membandingkan performa antar toko. Terakhir, fase *Temporal & Segment Analysis* menampilkan visualisasi berupa *heatmap* pendapatan bulanan serta proporsi segmen pelanggan di tiap cabang.



Gambar 3.38 Alur Inisialisasi Halaman dan Pengaturan Tampilan

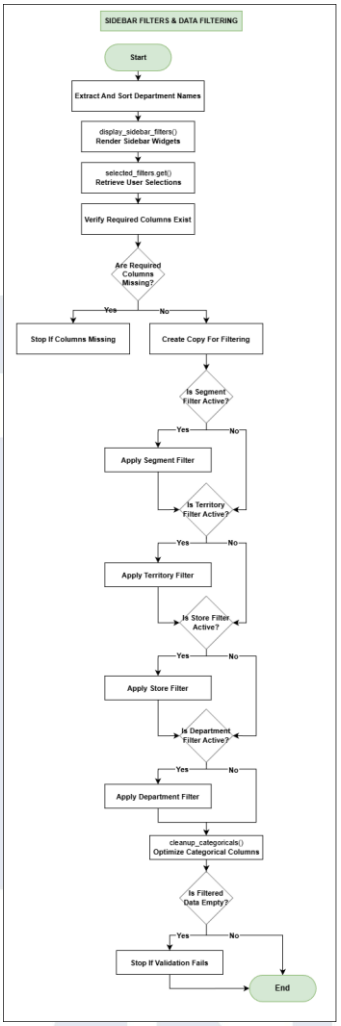
Kemudian dijelaskan proses *Initialization & UI Setup*, sebagaimana terlihat pada Gambar 3.38. Fase ini berguna sebagai awal halaman, di mana dipersiapkan komponen visual sebelum analisis dimulai. Aplikasi mengambil *library* penting, menentukan nilai-nilai dasar seperti tinggi chart dan atauapun warna yang digunakan, serta memanggil *st.set_page_config()* untuk mengaktifkan *wide layout*. Judul halaman ditampilkan, dan struktur dasar *sidebar* disiapkan agar alur analisis berikutnya dapat berjalan tanpa kendala. Fase ini berguna untuk memastikan

seluruh *UI* siap untuk menerima data dengan ukuran besar dan memproses visualisasi yang kompleks tanpa mengganggu performa aplikasi.



Gambar 3.39 Alur Pemuatan Data dan Validasi Struktur

Gambar 3.39 menggambarkan fase *Data Loading & Validation*, yang memastikan data dengan struktur pengelompokan siap digunakan untuk analisis. Pada tahap ini, sistem memulai *Performance Timer* kemudian memanggil fungsi *load_analysis_data_optimized()* untuk mengambil dataset penuh dari *session state* atau *disk*. Setelah data berhasil dimuat, fungsi *validate_analysis_data()* digunakan untuk memastikan kolom-kolom seperti *store name* dan *territory* tersedia. Proses kemudian dilanjutkan dengan aplikasi menampilkan sumber data beserta waktu lama data diakses. Fase ini penting karena akan berguna untuk seluruh fase berikutnya sehingga dapat menggunakan data yang bersih, lengkap, dan bebas dari *error*.



Gambar 3.40 Alur Filtering Data Berdasarkan Lokasi dan Hierarki

Seperti terlihat pada Gambar 3.40, fase *Sidebar Filters & Data Filtering* mengatur data dianalisis sesuai dengan pilihan yang dipilih oleh pengguna. Sistem menggabungkan interaksi melalui *sidebar* sehingga memotong data yang mencakup *filter* segmen, wilayah (*territory*), toko, dan departemen pilihan. Setiap *filter* melakukan pengecekan dengan kolom yang tersedia untuk mencegah data yang tidak konsisten. Setelah proses *filtering* selesai, sistem kemudian memanggil *cleanup_categoricals*() untuk menghapus kategori yang tidak lagi digunakan, sehingga

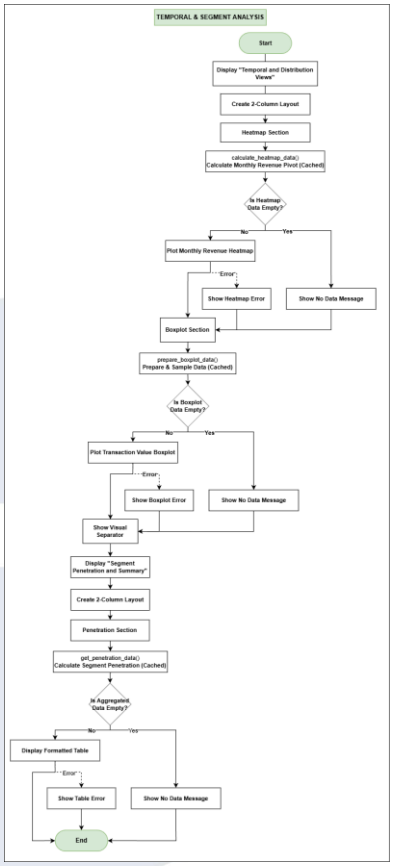
memori lebih efisien dan visualisasi tidak menampilkan kategori kosong pada chart.

Gambar 3.41 Alur Penambahan Metrik Transaksi per Toko

Gambar 3.41 menunjukkan fase *Store Aggregation*, berisikan fase logika perhitungan kinerja dari toko atau cabang. Di fase ini, dijalankan fungsi yang sudah ter-*cache* *aggregate_store_performance()* yang melakukan *group by* berdasarkan nama toko. Tiga metrik inti dihitung secara bersamaan, yaitu *total revenue*, *transaction count*, dan *average ticket size*. Setelah metrik-metrik tersebut dihitung, hasilnya akan diurutkan berdasarkan jumlah dari *revenue* untuk mengetahui toko dengan performa tertinggi. Daftar toko *Top N* kemudian diambil untuk digunakan pada visualisasi tren selanjutnya.

Gambar 3.42 Alur Visualisasi Tren dan Metrik Kinerja Toko

Seperti pada Gambar 3.42, fase *Trend & Key Metrics Visualizations* menghasilkan tiga komponen visual yang memberikan informasi mengenai performa cabang. Pertama, sistem memanggil *calculate_store_trends()* untuk visualisasi tren jumlah transaksi toko-toko *Top N* dalam *line chart*, yang membuat lebih mudah mengetahui pola kenaikan atau penurunan. Kedua, *treemap revenue* yang menampilkan kontribusi pendapatan per toko yang kemudian dikelompokkan berdasarkan wilayah, sehingga perbedaan karakteristik antar masing-masing wilayah dapat terlihat jelas. Ketiga, grafik batang yang digunakan untuk membandingkan total pendapatan secara langsung antar toko untuk mengenali *top performer*



Gambar 3.43 Alur Analisis Sementara dan Komposisi Segmen Pelanggan

Gambar 3.43 menggambarkan fase terakhir, yaitu *Temporal & Segment Analysis*, yang memperlihatkan analisis dimensi waktu dan profil pelanggan. Pada kolom visualisasi pertama, *monthly revenue heatmap* digunakan untuk memberikan informasi pendapatan bulanan per toko, membantu mendeteksi pola musiman, ataupun bagaimana performa antar cabang. Pada kolom kedua, *segment penetration analysis* memberikan gambaran persentase segmen pelanggan di setiap toko, sehingga dapat mengetahui bagaimana kualitas pelanggan di masing-masing lokasi. Analisis ini juga diikuti oleh *transaction value boxplot* yang memberi informasi tentang variasi nilai transaksi per toko, sehingga bisa identifikasi toko yang memiliki konsumen dengan *revenue* ataupun *buying power* yang lebih tinggi.

Gambar 3.44 Visualisasi Tampilan *Dashboard Store Performance*

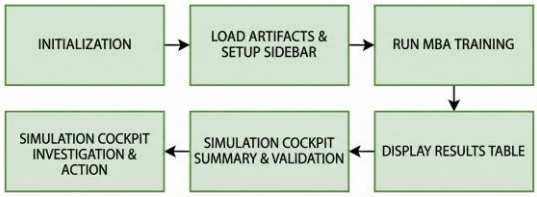
Halaman *Store Performance* memungkinkan evaluasi menggunakan data antar cabang. Seperti yang terlihat pada Gambar 3.44, penggunaan *bar chart* untuk menampilkan toko dengan *revenue* tertinggi memberikan informasi mengenai area yang paling potensial. Selain itu, *treemap* memberikan visualisasi mengenai performa setiap toko dalam suatu wilayah dapat menyiapkan bagaimana promosi yang lebih efektif secara personal bagi pelanggan di wilayah terkait.

Setelah melakukan analisis pada kondisi bisnis secara umum, dan terfokus berdasarkan setiap toko atau wilayah di halaman 3, selanjutnya akan lebih terfokus ke hal yang lebih detail seperti mengetahui produk mana yang sering dibeli bersama-sama dengan tujuan dapat membuat paket *bundling* atau promosi *cross-selling* yang tepat sasaran. Kebutuhan untuk memenuhi strategi *cross-selling* dan menghitung potensi keuntungannya inilah yang akan dibahas pada halaman 4, *Dashboard Market Basket Analysis (MBA).*

3.3.1.5 Dashboard Market Basket Analysis

Halaman *Market Basket Analysis* ini bertujuan untuk menemukan pola asosiasi produk, yaitu *item-item* yang sering

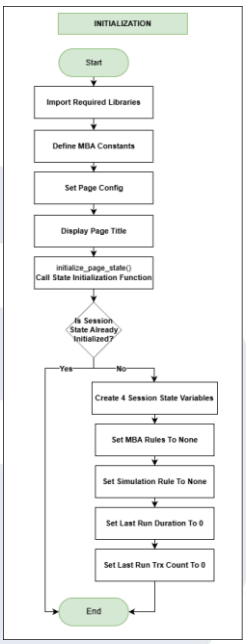
dibeli secara bersama oleh pelanggan. Halaman ini menjalankan algoritma *apriori* pada data transaksi untuk menunjukkan hasil keterkaitan atau *association rules*. Secara spesifik, dashboard ini beroperasi pada halaman *06_SIM_market_basket_analysis.py* dan memiliki dua fungsi utama, yang pertama adalah menampilkan relasi yang ditemukan, dan kedua adalah menyediakan fitur simulasi untuk menghitung potensi pendapatan tambahan dari strategi *cross-selling*. Tujuannya adalah memberikan informasi yang dapat digunakan untuk mengoptimalkan penawaran dan harga.



Gambar 3.45 Alur Utama Dashboard Market Basket Analysis

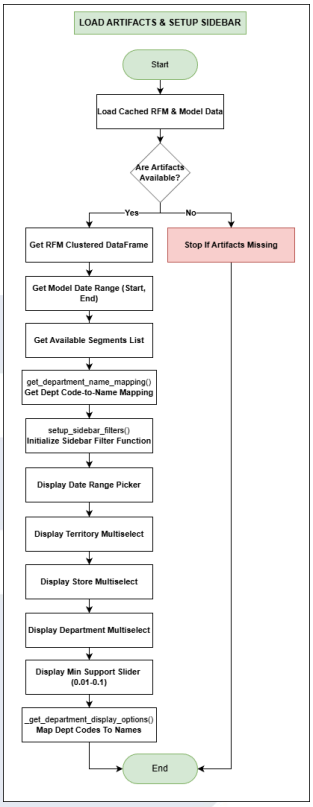
Pada Gambar 3.45, alur kerja Halaman *Market Basket Analysis* terdiri dari enam fase untuk menemukan pola *cross-selling* dan menghitung dampaknya. Proses dimulai dari fase *Initialization*, yaitu menyiapkan setup aplikasi dan membuat *session state* untuk menyimpan hasil dari *market basket analysis*. Setelah itu, pada fase *Load Artifacts & Setup Sidebar*, sistem mengakses *model* yang dibutuhkan dan menampilkan filter seperti tanggal, lokasi, dan segmen pelanggan. Kemudian berlanjut ke fase *Run MBA Training*, yaitu saat algoritma *apriori* yang dijalankan untuk menghasilkan aturan korelasi antar item dari data transaksi. Aturan yang muncul kemudian ditampilkan pada fase *Display Results Table* dalam bentuk tabel. Proses ditutup dengan dua jenis fase simulasi, yaitu *Simulation Cockpit: Summary & Validation* untuk menghitung estimasi *profit* dan *Simulation Cockpit: Investigation & Action* untuk melihat

perilaku segmen serta memberikan rekomendasi produk alternatif untuk strategi *cross-selling*.



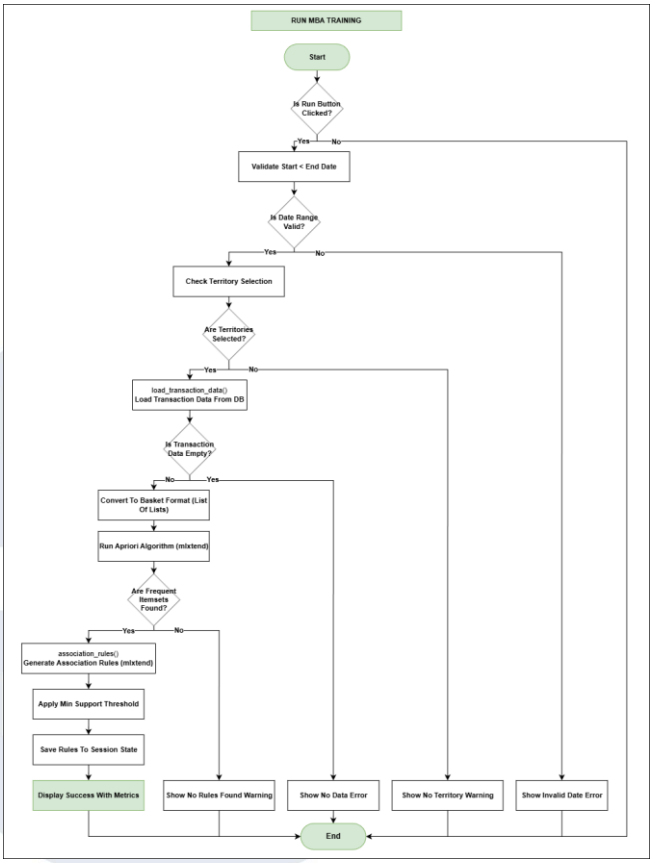
Gambar 3.46 Alur Inisialisasi Halaman dan Variabel Session State MBA

Pada Gambar 3.46 menggambarkan fase pertama, yaitu *Initialization*, yang menjadi awalan saat halaman pertama kali diakses. Pada tahap ini, sistem mengambil *library* yang dibutuhkan dan melakukan inisialisasi variabel *session state* untuk memastikan penyimpanan data nantinya seperti *mba.rules* dapat digunakan dan dalam keadaan kosong. Langkah ini penting untuk dilakukan sehingga mencegah tercampurnya data yang lama dengan data analisis baru.



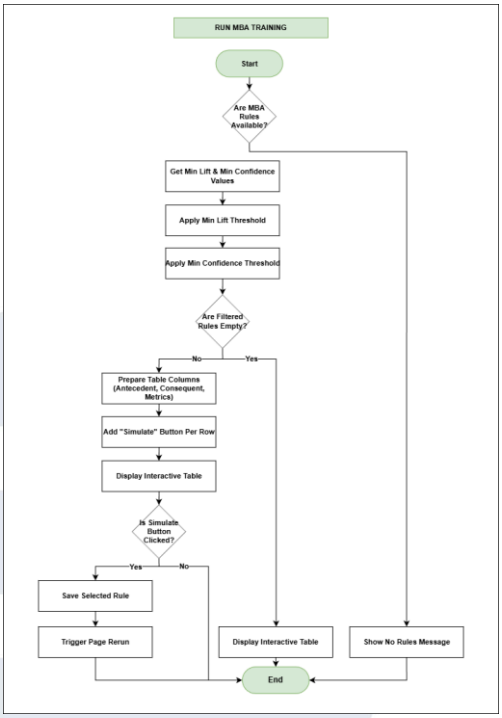
Gambar 3.47 Alur Pemuatan Data RFM dan Filter Sidebar MBA

Gambar 3.47 menunjukkan fase kedua, yaitu *Load Artifacts & Setup Sidebar*, yang berfokus pada persiapan data dan tampilan halaman nantinya. Sistem mengakses artefak data *RFM* yang sudah ada dari penyimpanan agar analisis dapat dikaitkan dengan segmentasi pelanggan. Bersamaan dengan itu, sidebar juga dipanggil untuk menampilkan filter interaktif (seperti rentang tanggal dan wilayah) serta *slider* khusus untuk mengatur parameter *minimum support* atau batas *threshold*.



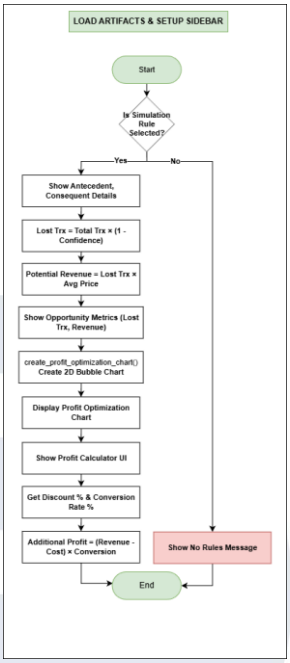
Gambar 3.48 Alur Algoritma Apriori dan Pembentukan Aturan Asosiasi

Gambar 3.48 memvisualisasikan fase inti, yaitu *Run MBA Training*, di mana proses logika algoritma dijalankan. Setelah tombol "*Run*" ditekan, sistem akan menarik data transaksi dari *database*, mengubah strukturnya menjadi format keranjang belanja (*basket format*), dan mengeksekusi algoritma *apriori*. Hasil dari proses ini adalah ditemukannya korelasi antar produk yang valid untuk disimpan secara sementara.



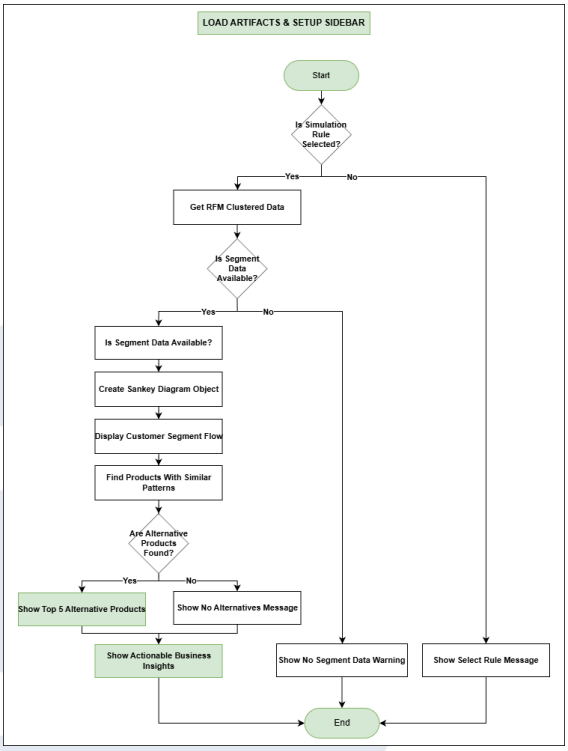
Gambar 3.49 Alur Filterisasi Aturan MBA dan Tampilan Tabel

Gambar 3.49 memberikan gambaran fase keempat, yaitu *Display Results Table*, yang menampilkan hasil temuan algoritma di tampilan halaman. Sistem menampilkan tabel yang berisi daftar aturan produk, dilengkapi dengan slider untuk *filtering* aturan berdasarkan seberapa kuat relasi antar item tersebut (*lift* dan *confidence*). Di setiap baris tabel, terdapat tombol "*Simulate*" yang berfungsi sebagai tombol untuk membawa pengguna ke fitur simulasi *profit*.



Gambar 3.50 Alur Analisis Peluang dan Kalkulasi Keuntungan

Gambar 3.50 memperlihatkan fase kelima, yaitu *Simulation Cockpit: Summary & Validation*, yang aktif setelah sebuah aturan dipilih melalui *button* "*Simulate*. Fase ini berfokus pada kalkulasi *revenue* bisnis, yang memungkinkan logika perhitungan peluang transaksi yang hilang (*lost opportunity*). Fitur utamanya adalah kalkulator yang berguna untuk menghitung *profit* secara dinamis sehingga memungkinkan pengguna untuk memasukkan estimasi diskon dan konversi untuk melihat proyeksi keuntungan tambahan secara instan.



Gambar 3.51 Alur Segmen Pelanggan dan Rekomendasi Produk Alternatif

Gambar 3.51 menggambarkan fase terakhir, yaitu *Simulation Cockpit: Investigation & Action*, yang memberikan informasi strategis untuk mengetahui segmentasi pelanggan dan item yang memiliki korelasi. Melalui visualisasi *diagram sankey*, sistem memberi gambaran segmen pelanggan mana yang paling responsif terhadap paket produk tersebut. Selain itu, sistem juga mengidentifikasi dan menampilkan produk alternatif yang memiliki pola pembelian terkait sebagai rekomendasi cadangan untuk strategi *cross-selling*.

Gambar 3.52 Visualisasi Tampilan Dashboard Market Basket Analysis

Gambar 3.52 memperlihatkan tampilan utama hasil analisis, yaitu tabel *association-rules*, yang memperlihatkan daftar pasangan produk yang terbukti sering dibeli bersamaan. Pada tampilan ini, pengguna dapat melihat detail produk pemicu dan produk pasangannya, lengkap dengan nilai kekuatan hubungan (*lift*). Fitur utamnya adalah tombol "*Simulate*" di sebelah kanan tabel yang berfungsi untuk melakukan simulasi seberapa menguntungkan jika aturan produk tersebut diterapkan.

Gambar 3.53 Visualisasi Tampilan Dashboard Market Basket Analysis 2

Gambar 3.53 menampilkan simulasi, yang menggabungkan kalkulator *revenue* produk, di bagian ini, terlihat fitur input interaktif di mana pengguna dapat memasukkan besar diskon dan target konversi untuk melihat angka estimasi *additional profit*.

Bersamaan dengan itu, visualisasi *diagram sankey* di tab selanjutnya juga memperlihatkan segmentasi pelanggan dalam penjualan tersebut, membantu pengguna memahami kelompok mana yang paling berpotensi membeli paket *bundling* tersebut.

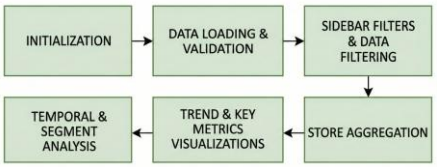
Penerapan dashboard ini, bersifat mengubah data transaksi yang mentah menjadi strategi penjualan yang terukur. Dengan fitur yang membantu menemukan pola pembelian keranjang dan menghitung kalkulasi *revenue* atau *profit*, tim terkait dapat merancang promo berdasarkan landasan data yang kuat, memastikan setiap paket *bundling* yang diluncurkan memiliki target pasar yang jelas dan estimasi keuntungan yang terhitung sebelum dieksekusi ke pasar.

Hasil dari Halaman *Market Basket Analysis* memberikan daftar rekomendasi produk yang potensial untuk dipaketkan (*bundling*). Namun, sebelum strategi tersebut dieksekusi, diperlukan validasi mendalam terhadap kinerja individu produk tersebut. Analisis produk secara spesifik inilah yang dilakukan pada halaman 5, *Dashboard Product Analysis* yang menggunakan metode klasifikasi *ABC-XYZ* untuk menilai status setiap barang secara detail.

3.3.1.6 Dashboard Product Analysis

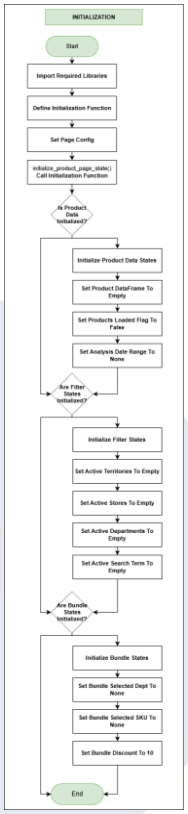
Halaman *Dashboard Product Analysis* atau halaman ke-5 ini difokuskan untuk melakukan penilaian secara mendalam pada kinerja satu produk (*SKU*) secara spesifik. Halaman ini berfungsi untuk memperlihatkan performa produk tersebut menggunakan data historis, bukan dengan sekadar asumsi. Secara teknis, *dashboard* ini juga menggunakan metode klasifikasi *ABC-XYZ* yang digunakan untuk menentukan status produk. Hal tersebut dapat membantu misalnya produk dengan *revenue* tinggi, namun

permintaan tidak stabil, sehingga dapat memberikan simulasi strategi untuk aksi yang dapat dilakukan nantinya.



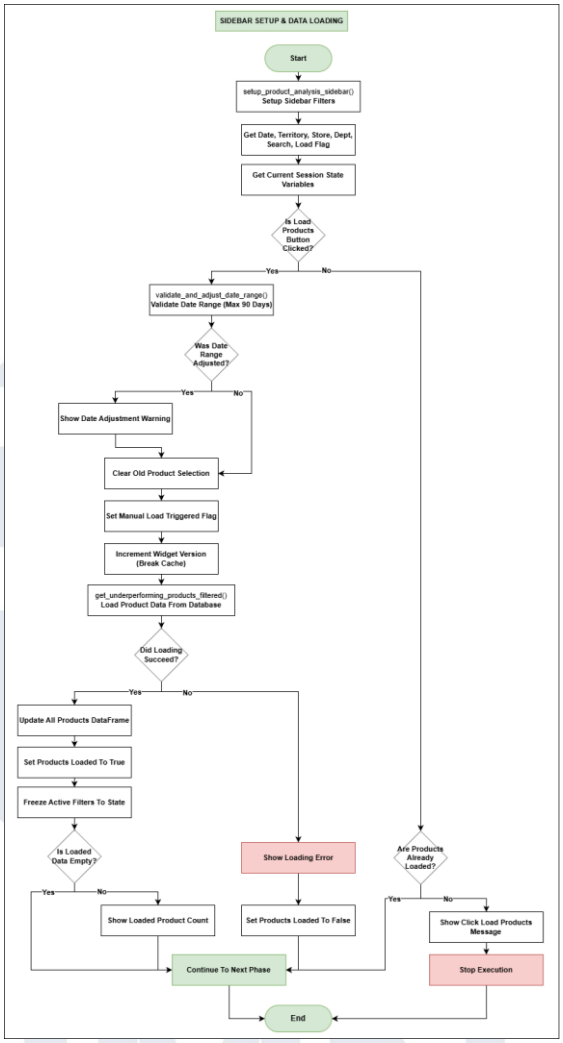
Gambar 3.54 Alur Utama Dashboard Product Analysis

Mengacu pada Gambar 3.54, alur kerja pada halaman ini terdiri dari enam fase utama yang berurutan. Proses dimulai dari *Initialization* untuk persiapan awal aplikasi, dilanjutkan dengan *Sidebar Setup & Data Loading* untuk memuat *list* atau isi dari pilihan produk. Fase berikutnya adalah *Visual Product Discovery*, yang memungkinkan pengguna dapat memilih produk korelasi atau produk *target* melalui *filter*. Setelah produk dipilih, sistem menjalankan *Product Analysis Execution* untuk menampilkan metrik-metrik penting. Hasil analisis kemudian disajikan dalam dua kelompok, yakni visualisasi *tab* 1 & 2 untuk mengetahui kinerja secara umum dan konteks toko, serta *tab* 3, 4, & 5 untuk analisa strategi secara mendalam.



Gambar 3.55 Alur Inisialisasi Halaman dan Reset Variabel

Pada. Gambar 3.55 menggambarkan fase pertama yaitu *Initialization*, yang berfungsi untuk melakukan persiapan analisis data. Pada tahapan ini, sistem tidak hanya mengakses *library*, tetapi juga melakukan pengaturan awal kepada *session state*. Sistem melakukan inisiasi variabel khusus seperti *all_products_df* untuk diisi dengan pilihan produk dan *bundle_selected_sku* untuk menampung hasil simulasi dalam keadaan kosong. Langkah inisialisasi ini penting untuk mencegah terjadinya data yang saling menimpa, atau tercampurnya informasi dari analisis produk sebelumnya ketika perpindahan halaman atau melakukan pencarian baru.

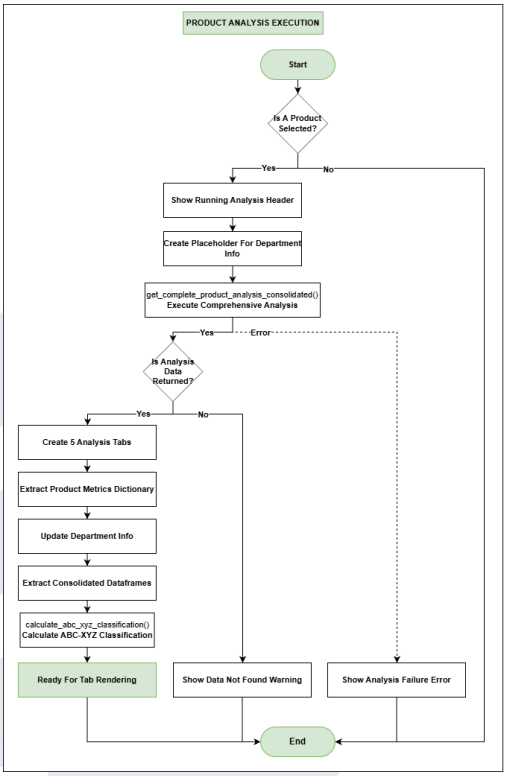


Gambar 3.56 Alur Pemuatan List Produk dan Filter Sidebar

Gambar 3.56 menunjukkan fase kedua yaitu *Sidebar Setup & Data Loading*, di mana sistem mempersiapkan tampilan awal pengguna dan mengambil *list* data untuk pilihan produk. Sistem mengakses daftar produk dari *database* dengan mengambil kolom penting seperti nama *SKU* dan total pendapatan, lalu melakukan konfigurasi *filter* di *sidebar* untuk pilihan wilayah, toko, dan departemen. Selain itu, fase ini juga mencakup proses validasi untuk rentang tanggal di bagian *filter* yang memastikan periode analisis tidak lebih dari batas performa, sehingga proses loading data transaksi dapat tetap berjalan cepat dan responsif.

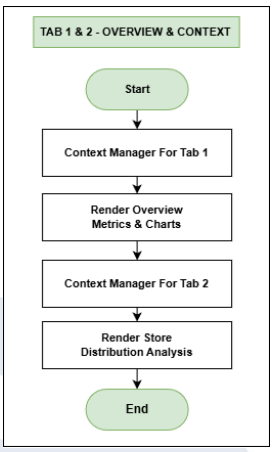
Gambar 3.57 Alur Penemuan Produk Melalui Distribusi Visual

Gambar 3.57 memvisualisasikan fase ketiga yaitu *Visual Product Discovery*, yang dibuat untuk mengatasi kesulitan dalam pencarian satu produk spesifik di antara ribuan *SKU*. Tidak hanya mengandalkan metode pencarian biasa, sistem akan memberikan grafik yang memperlihatkan persebaran distribusi revenue atau *Revenue Distribution Chart* yang memberi tahu posisi setiap produk berdasarkan persentil kinerjanya. Fitur ini berguna untuk menyaring dan memilih produk secara visual, misalnya mem*filter* hanya produk pada lapisan "*Top 20%*" atau "*Bottom 10%*", yang kemudian sistem akan dapat mengurutkan opsi pada *menu dropdown* berdasarkan pendapatan tertinggi untuk memudahkan seleksi produk.



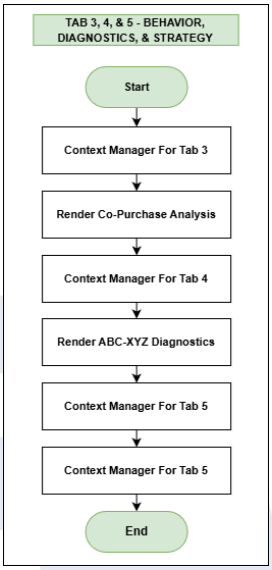
Gambar 3.58 Alur Eksekusi Proses Analisis Produk

Gambar 3.58 merepresentasikan fase utama yaitu *Product Analysis Execution*, di mana beban proses komputasi itu dijalankan setelah produk dipilih. Sistem akan memanggil fungsi bernama *get_complete_product_analysis_consolidated()* yang bertugas untuk menarik seluruh riwayat transaksi secara detail terkait dengan produk tersebut dari *database*. Dalam proses eksekusi, sistem secara akan menghitung berbagai metrik, mulai dari tren penjualan secara harian, margin keuntungan, distribusi toko, hingga pola dari transaksi pelanggan. Proses ini memastikan bahwa semua data yang dibutuhkan untuk lima bagian analisis berikutnya akan tersedia tanpa memerlukan *query* berulang dan dapat memperlambat aplikasi.



Gambar 3.59 Alur Visualisasi Metrik Kinerja General

Gambar 3.59 memperlihatkan fase kelima yaitu *Tab 1 & 2 - Overview & Context*, yang memberikan hasil dari analisis mengenai bagaimana kinerja produk. Pada *Tab* 1 atau *Product Overview*, sistem menampilkan skor kinerja produk secara keseluruhan yang mencakup total pendapatan, total transaksi, rata-rata *basket size*, dan *margin* dalam rentang waktu yang terpilih. Analisis tersebut kemudian diperluas pada *Tab* 2 atau *Performance Context*, di mana sistem membandingkan bagaimana hasil kinerja produk tersebut terhadap rata-rata produk lain di departemen yang sama. Hal ini memungkinkan identifikasi secara cepat bagaimana produk tersebut, dapat laku secara merata atau hanya kuat di lokasi tertentu saja.

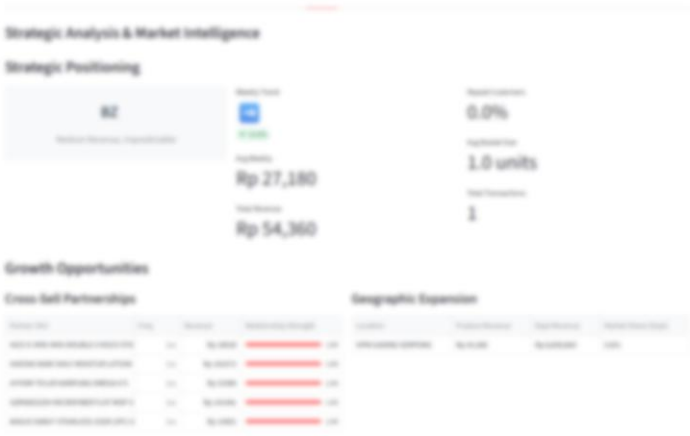


Gambar 3.60 Alur Diagnosa ABC-XYZ dan Simulasi Bundling

Gambar 3.60 menggambarkan fase terakhir dan terlengkap yaitu *Tab 3, 4, & 5 - Behavior, Diagnostics, & Strategy*, yang berfokus pada informasi yang *deep dive*. Sistem melakukan tiga analisis secara spesifik di sini. Pertama, pada Tab 3 atau *Customer Behavior & Cross-Selling*, pengguna dapat memilih produk pendamping yang direkomendasikan oleh sistem berdasarkan korelasi pembelian. Fitur ini bertujuan untuk meningkatkan performa produk dengan menampilkan item yang cenderung dibeli bersamaan oleh pelanggan. Kedua, mengklasifikasikan status produk menggunakan matriks *ABC-XYZ* di *Tab* 4 atau *Diagnostics*, contohnya mengidentifikasi produk kelas C yang penjualannya naik turun. Terakhir, di *Tab* 5 atau *Strategic Analysis* ditampilkan sistem yang tidak hanya memberikan rekomendasi secara biasa, tetapi juga menyediakan kesimpulan secara bersamaan sehingga mengetahui karakteristik produk secara compact dan jelas.

Gambar 3.61 Visualisasi Tampilan Dashboard Product Analysis 1

Gambar 3.61 memperlihatkan tampilan tab 3 atau *Customer Behavior & Cross-Selling*, tempat di mana perhitungan simulasi dilakukan. Fitur utamanya adalah *co-related item* atau *bundling simulator* dan kalkulator yang berguna secara interaktif, yang terlihat di sisi kanan layar. Pengguna dapat memilih departemen dan produk pilihan atau *SKU Partner*, lalu memasukkan persentase diskon. Sistem kemudian dapat secara otomatis melakukan kalkulasi dan menampilkan angka *Estimated Additional Profit*. Ini membuat dashboard tidak hanya menampilkan data secara statis, tetapi juga memberikan skenario paket penjualan yang menguntungkan dan korelasinya secara langsung.



Gambar 3.62 Visualisasi Tampilan Dashboard Product Analysis 2

Gambar 3.62 menampilkan *tab* 5 atau *Strategic Analysis*, yang memberikan ringkasan untuk untuk pengambilan keputusan di skala general. Pada tampilan ini, terlihat bagian *Strategic Positioning* yang langsung memberikan label klasifikasi produk (seperti *AX* atau *BY*) beserta deskripsi strategisnya. Selain itu, terdapat bagian *Store Expansion Potential* yang merekomendasikan daftar toko spesifik yang memiliki potensi pasar, namun belum menjual produk tersebut. Tampilan ini memberikan fungsi dashboard sebagai alat yang mengubah metrik rumit menjadi rekomendasi strategi yang jelas.

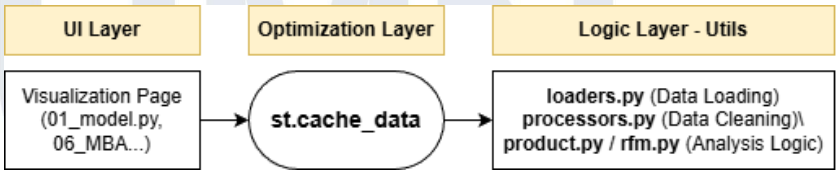
Halaman *Dashboard Product Analysis* ini memberikan arahan bagi setiap item dalam data perusahaan. Melalui *simulator* di *tab* 3, tim operasional dapat mengetahui paket promo melalui riwayat transaksi yang terukur korelasi dan profitnya. Sementara itu, ringkasan strategis di *tab* 5 juga berguna bagi para *shareholder* mengetahui karakteristik produk secara cepat, sehingga mengetahui mana yang memiliki potensial dan harus dievaluasi ulang. Kombinasi ini memastikan bahwa keputusan dari pihak *supply chain* dan *marketing* memiliki tambahan dasar yang lebih kuat dalam proses eksekusi memanfaatkan dari diagnosa data yang presisi.

Setelah seluruh halaman analisis selesai dibangun, mulai dari model segmentasi pelanggan hingga analisis produk mendalam, fokus utama kini berpindah pada perbaikan kualitas dan memastikan sistem dapat terdokumnetasi dengan baik. Aplikasi tidak hanya membutuhkan fitur yang lengkap, tetapi juga infrastruktur code yang bersih dan performa yang dapat berjalan dengan baik saat diakses oleh banyak pengguna. Oleh karena itu, sebelum sistem siap digunakan sepenuhnya, dilakukan serangkaian pengujian mulai dari *End-to-End Testing*, *Stress*

Test, hingga perbaikan *bug*, serta penyusunan *flow* arsitektur yang lengkap. Seluruh proses pemantapan teknis dan administratif ini dikerjakan dalam fase *Quality Assurance & Documentation*.

3.3.1.7 Quality Assurance & Documentation

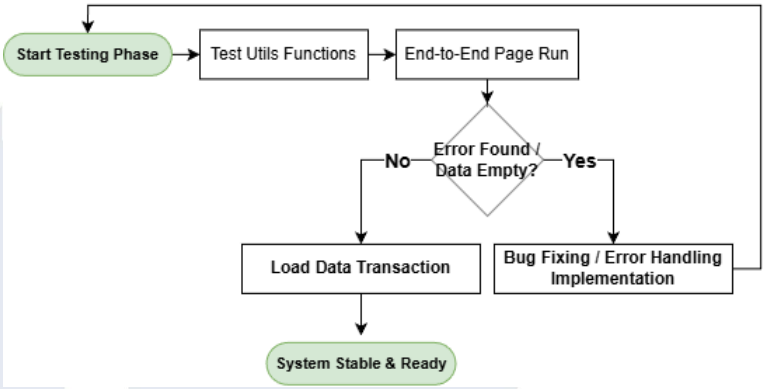
Setelah seluruh dashboard untuk analisis data selesai dikembangkan, mulai dari segmentasi pelanggan pada halaman pertama hingga analisis produk pada halaman terakhir, fokus pekerjaan yang sebelumnya merupakan pengembangan fitur, akan berpindah pada perbaikan dan *improvement* pada arsitektur sistem. Aplikasi dalam penerapan *data science* pada skala industri tidak hanya dinilai berdasarkana seberapa canggih model yang digunakan, tetapi juga dari efisiensi sistem dan kemudahannya dalam maintenance dalam jangka panjang. Oleh karena itu, bagian ini akan membahas fase *Quality Assurance & Documentation*, yaitu rangkaian proses untuk memastikan sistem berjalan stabil, bebas dari bug, efisien secara komputasi, serta penjelasan dokumentasi teknis yang jelas agar dapat digunakan oleh tim perusahaan.



Gambar 3.63 Alur Proses Code Refactoring & Optimization

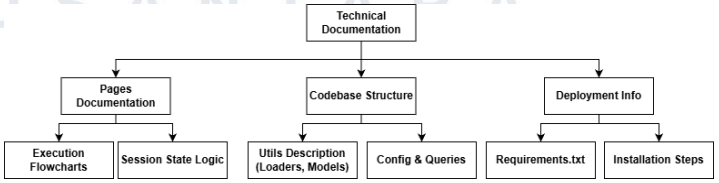
Seperti yang diilustrasikan pada Gambar 3.63, proses refactoring dilakukan dengan memisahkan bagian logika dari bagian tampilan UI. Halaman visualisasi (seperti *01_model.py*) tidak lagi melakukan pemrosesan data secara langsung di file yang sama, melainkan memanggil fungsi yang terintegrasi di lapisan *Logic Layer - Utils* (seperti *loaders.py* dan *processors.py*). Di antara kedua lapisan tersebut, diterapkan lapisan optimasi menggunakan st.cache_data. Fungsi tersebut berguna sebagai

penyimpanan sementara yang menyimpan hasil komputasi berat di memori sementara, sehingga data tidak perlu diproses ulang setiap kali pengguna melakukan interaksi di halaman, yang membuat proses lebih efisien dan memberi beban lebih pada query maupun proses komputasi.



Gambar 3.64 Alur Proses System Testing & Bug Fixing

Gambar 3.64 menggambarkan alur untuk menguji atau testing pada sistem yang diterapkan untuk mengetahui bagaimana proses aplikasi, apakah stabil atau tidak. Proses dimulai dengan menguji unit pada fungsi-fungsi dasar (*Test Utils Functions*), diikuti oleh pengujian pada halaman-halaman yang ada (*End-to-End Page Run*). Apabila ditemukan error atau data kosong selama proses ini, alur akan masuk ke tahap *Bug Fixing* dan perbaikan penanganan error. Siklus ini bersifat mengulang, jadinya sistem hanya dinyatakan stabil dan siap digunakan (*System Stable & Ready*) jika seluruh proses pengujian, termasuk uji beban (*load transaction*), berhasil lewat tanpa kegagalan.



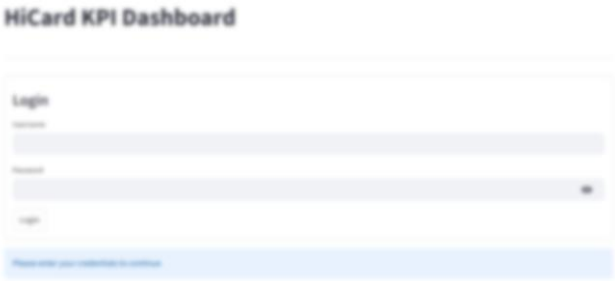
Gambar 3.65 Alur Proses Comprehensive Project Documentation

Untuk mengetahui bagaimana proses lanjutan pada halaman-halaman yang ada, dibuatkan suatu dokumentasi teknis yang terstruktur seperti terlihat pada Gambar 3.65. Dokumentasi ini dipecah menjadi tiga pilar utama. Yang pertama, *Pages Documentation* yang menjelaskan alur eksekusi setiap halaman dan logika *session state*. Kedua, *Codebase Structure* yang membahas secara detil fungsi modul-modul yang ada serta bagaimana konfigurasi *query*. Dan yang terakhir adalah *Deployment Info* yang berisi panduan instalasi dan daftar *library* yang dibutuhkan *(requirements.txt*). Struktur ini dirancang untuk memudahkan tim dalam memahami dan melakukan maintenance sistem dengan lebih proper.

Dengan selesainya tahap untuk pengujian sistem, dan penyusunan dokumentasi teknis, aplikasi kini diharapkan mencapai status stabil dan siap. Seluruh kode halaman aplikasi telah dioptimalkan melalui refactoring, potensi error telah dimitigasi, dan informasi teknis mengenai halaman-halaman analisis tersebut telah dijelaskan dalam bentuk dokumen. Selanjutnya, laporan ini akan melanjutkan ke halaman *Dashboard Streamlit KPI Membership*, yang akan membahas proyek lainnya mengenai proses migrasi dan pembuatan ulang visualisasi dari *Looker Studio* ke *Dashboard Streamlit* mengenai data membership.

3.3.1.8 Dashboard Streamlit KPI Membership

Proyek untuk halaman *Dashboard Streamlit KPI Membership* ini berfokus pada perpindahan visualisasi yang menunjukkan kinerja program loyalitas toko yakni *Hicard*, yang sebelumnya dari platform *Looker Studio* ke aplikasi *Streamlit*. Proses utama dalam proyek ini bukan hanya pada interaktivitas visual-nya, melainkan juga pada aspek keamanan data.



Gambar 3.66 Tampilan Halaman Login dan Sistem Autentikasi

Untuk Gambar 3.66 memperlihatkan bagaimana tampilan autentikasi yang menjadi tampilan utama aplikasi. Sistem ini dibangun menggunakan library streamlit_authenticator untuk menerapkan proses Role-Based Access Control (RBAC). Seperti yang terlihat pada tampilan tersebut, pengguna diwajibkan memasukkan kredensial atau informasi yang valid. Di balik layar, konfigurasi kredensial dan peran pengguna disimpan secara aman dalam file autentikasi yang dibaca menggunakan sistem caching untuk menjaga performa. Fitur ini berguna untuk membatasi halaman-halaman krusial agar hanya dapat diakses oleh pengguna dengan role "*Admin*", sementara pengguna biasa memiliki akses yang terbatas. data.

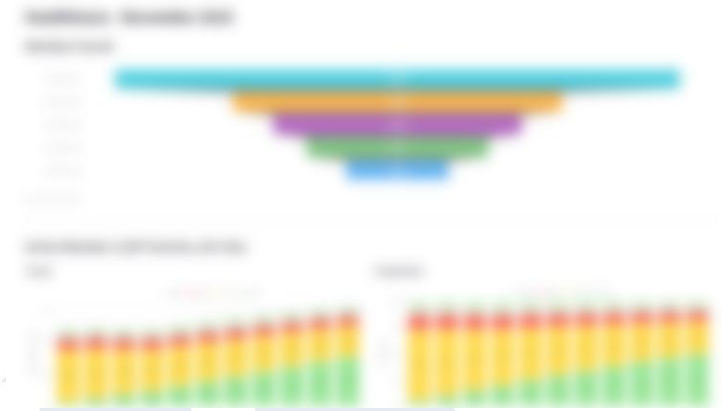
Gambar 3.67 Tampilan Visualisasi Ringkasan pada KPI

Setelah pengguna berhasil melewati proses autentikasi, sistem akan mengarahkan ke halaman pertama. Gambar 3.67 menampilkan visualisasi metrik-metrik utama pada *KPI*

Dashboard, yang memberikan bagaimana status kinerja program loyalitas dengan angka yang terfokus pada 1 bulan terakhir. Pada bagian atas dashboard, terlihat kartu metrik atau scorecards yang berisikan data penting seperti total *Registered Member*, *Sales Contribution*, dan *Customer Lifetime Value* (*CLTV*). Bagian ini dibuat untuk memberikan bagaimana skala informasi secara general atau luas kepada mengenai pertumbuhan anggota dan perbandingan kinerja dari waktu ke waktu.

Gambar 3.68 Tampilan Visualisasi Demografi Member

Untuk memahami karakteristik member yang sudah terdaftar lebih dalam, dashboard ini juga menyediakan analisis secara demografis dan bagaiman siklus dari pelanggan. Gambar 3.68 merepresentasikan visualisasi analisis demografi member, yang di mana memberikan informasi distribusi pelanggan berdasarkan gender, wilayah, dan kelengkapan data profil divisualisasikan. Informasi ini krusial bagi tim terkait untuk menargetkan segmentasi yang tepat.



Gambar 3.69 Tampilan Visualisasi Kesehatan Member

Melengkapi analisis tersebut, Gambar 3.69 menunjukkan bagaimana kondisi dari member. Tampilan ini berfokus pada analisis *Member Funnel* dan segmentasi *RFM* (*Recency, Frequency, Monetary*). Melalui visualisasi ini, pihak terkait dapat mengetahui distribusi dan membedakan antara anggota yang benar-benar aktif (*Loyal*) dengan anggota yang berisiko berhenti atau pergi (*Churn*), di serta di saat yang bersamaan memantau tren anggota berdasarkan tahun bergabung atau visualisasi *cohort analysis*.

Gambar 3.70 Tampilan Visualisasi Sales Breakdown

Halaman *Sales Breakdown* ini berfungsi untuk mengetahui total penjualan menjadi bagian-bagian yang lebih kecil agar mudah dipahami, menggunakan tampilan berupa *tree chart* seperti pada Gambar 3.70. Dengan alat ini, dapat ditelusuri

mengenai penyebab naik-turunnya pendapatan secara spesifik, dan menilai apakah disebabkan oleh pelanggan yang makin sering datang (*Volume Transaksi*) atau karena nilai belanjaan mereka yang makin besar (*Basket Size*). Diagram ini dipecah dari *level member* hingga ke harga barang, dan dilengkapi dengan indikator untuk mengetahui area mana yang tumbuh positif atau justru mengalami penurunan.

Secara keseluruhan, migrasi *dashboard* ke *Streamlit* ini diharapkan dapat meningkatkan keamanan data melalui sistem *login* yang baru, sekaligus memberikan tampilan laporan yang lebih interaktif dibandingkan sebelumnya. Dengan adanya visualisasi lengkap mulai dari ringkasan *KPI* hingga *deep-dive* penjualan, tim terkait dapat menggunakan *dashboard* untuk memantau performa bisnis dan mengambil keputusan strategi yang tepat sasaran.

3.3.2. Kendala yang Ditemukan

Selama proses pengembangan sistem *dashboard*, baik dari *dashboard* analisis, simulasi, maupun konversi dari *dashboard* lainnya, ditemui beberapa tantangan dalam proses magang. Dari segi teknis, terutama dalam mengolah data transaksi, yang jumlahnya sangat besar hingga mencapai jutaan baris. *Volume* data yang besar ini membuat proses perhitungan algoritma yang rumit, seperti *Market Basket Analysis* dan *Clustering*, menjadi beban kerja yang cukup berat bagi sistem. Sehingga, penggunaan memori komputer sering kali mencapai batas maksimal dan waktu pemrosesan data menjadi lebih lama, sehingga penulis harus melakukan berbagai upaya optimasi kode agar *dashboard* tetap terasa ringan dan cepat saat digunakan.

Selain aspek teknis, tantangan lain yang dihadapi adalah proses memahami aturan bisnis ke dalam logika pemrograman. Diperlukan momen untuk memahami secara mendalam bagaimana perusahaan

mendefinisikan istilah-istilah penting seperti *churn rate*, klasifikasi produk, hingga segmentasi pelanggan agar dapat membuat konsep dengan tepat ke dalam kode *Python*. Pada tahap ini diperlukan ketelitian yang mendalam untuk memastikan bahwa hasil analisis yang ditampilkan di dashboard benar-benar akurat, sesuai dengan standar operasional perusahaan, dan tidak menimbulkan hasil data yang kesannya tidak benar atau keliru.

3.3.3. Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala pada kinerja sistem akibat *volume* data yang besar, penulis menerapkan strategi optimasi melalui fitur *caching* pada aplikasi. Fitur ini berguna untuk menyimpan hasil dari perhitungan algoritma yang rumit di memori sementara, sehingga sistem tidak perlu melakukan pemrosesan ulang dari nol setiap kali pengguna melakukan interaksi pada aplikasi. Langkah ini terbukti efektif membuat waktu pemuatan *dashboard* menjadi jauh lebih cepat dan menjaga penggunaan memori komputer lebih efisien. Selain itu, penulis juga melakukan pengambilan data secara terpilih dari *database*, dan memastikan hanya data yang benar-benar dibutuhkan saja yang diproses.

Sementara itu, untuk menjawab tantangan dalam pemahaman bisnis, dilakukan proses validasi data secara bertahap. Untuk menghindari data yang keliru, diperlukan perbandingan antara hasil perhitungan dari kode *Python* yang dibuat dengan laporan manual ataupun mengecek data historis perusahaan untuk memastikan akurasinya. Melakukan diskusi dengan *supervisor* juga dilakukan untuk mengonfirmasi pemahaman dari setiap metrik, sehingga algoritma yang dibangun, seperti dalam segmentasi pelanggan dan juga klasifikasi produk apakah sudah sesuai dengan yang berlaku di perusahaan dan dapat dipercaya untuk pengambilan keputusan.