

**UJIAN AKHIR SEMESTER
KECERDASAN BISNIS**

[DOC_TYPE : TECHNICAL REPORT]

**IMPLEMENTASI
DATA LAKEHOUSE**

**" Optimasi Keputusan Mandi Menggunakan Data Lakehouse dan
Prescriptive Analytic "**

Optimasi Keputusan Sehari-hari Menggunakan Prescriptive Analytics

RINGKASAN TEKNIS: Laporan ini mengimplementasikan sistem pendukung keputusan mandi berbasis Data Lakehouse. Integrasi data aktivitas dan lingkungan diolah melalui untuk menghasilkan rekomendasi preskriptif.

TIM ENGINEERING

ANDRA BRAPAPUTRA AKBAR SALEH
NIM: 2310817210001

MUHAMMAD AZWIN HAKIM
NIM: 2310817310012

PROGRAM STUDI TEKNOLOGI INFORMASI
SEMESTER GANJIL 2025

Halaman Persembahan

Karya ini dipersembahkan sebagai hasil dari proses pembelajaran akademik dalam memahami dan menerapkan konsep kecerdasan bisnis berbasis data. Laporan ini merepresentasikan upaya penulis dalam mengintegrasikan pengetahuan konseptual dan praktik teknis ke dalam perancangan serta implementasi sistem pendukung keputusan berbasis data.

Penyusunan laporan ini tidak terlepas dari peran dosen pengampu mata kuliah Kecerdasan Bisnis yang telah memberikan arahan, pemahaman konseptual, serta lingkungan pembelajaran yang mendorong pengembangan pemikiran kritis dan sistematis. Melalui proses perkuliahan dan diskusi akademik, penulis memperoleh landasan teoretis yang menjadi dasar dalam penyusunan dan implementasi sistem yang dibahas dalam laporan ini.

Karya ini juga mencerminkan semangat kolaborasi dan tanggung jawab bersama dalam menyelesaikan permasalahan secara terstruktur. Seluruh proses perancangan sistem dan penyusunan laporan dilakukan secara setara, terkoordinasi, dan berorientasi pada pencapaian tujuan akademik yang telah ditetapkan. Pengalaman ini menjadi bagian penting dalam pembentukan sikap profesional, kedisiplinan, serta kemampuan bekerja secara kolaboratif dalam konteks pengembangan sistem berbasis data.

Selain sebagai pemenuhan tugas akademik, laporan ini diharapkan dapat memberikan manfaat sebagai bahan pembelajaran dan referensi awal dalam memahami penerapan arsitektur data lakehouse dan analitik preskriptif pada skala sederhana. Penulis menyadari bahwa karya ini masih memiliki keterbatasan, namun diharapkan dapat menjadi pijakan untuk pengembangan pengetahuan dan keterampilan yang lebih lanjut di masa mendatang.

Daftar Isi

HALAMAN PERSEMBAHAN	I
DAFTAR ISI.....	II
DAFTAR GAMBAR.....	IV
DAFTAR TABEL.....	VI
BAB I.....	7
1.1. LATAR BELAKANG.....	7
1.2. <i>PROJECT SCOPE</i>	8
BAB II	9
2.1 URGENSI PERANCANGAN SISTEM BERBASIS DATA LAKEHOUSE.....	9
2.2 PRINSIP ARSITEKTUR DATA LAKEHOUSE YANG DIGUNAKAN	9
2.3 ALAT IMPLEMENTASI	10
2.4 PERANCANGAN ARSITEKTUR DATA LAKEHOUSE.....	10
2.5 DATASET YANG DIBUTUHKAN DAN DIKUMPULKAN	12
2.5.1 <i>Database Terstruktur (SQL)</i>	12
2.5.2 <i>Data Semi-Terstruktur (Google Sheets)</i>	13
2.5.3 <i>Dataset Eksternal (API Lingkungan)</i>	13
2.6 SCEDULING.....	15
2.7 PERANCANGAN ZONA PENYIMPANAN DATA	16
2.7.1 <i>Raw Zone</i>	16
2.7.2 <i>Clean Zone</i>	16
2.7.3 <i>Curated Zone</i>	17
2.8 PERANCANGAN PIPELINE ELT DAN ETL	17
2.8.1 <i>Tahap Ingest Data (Extract dan Load)</i>	19
2.8.2 <i>Tahap Transform (ETL) pada ELT</i>	20
2.9 PERANCANGAN SERVING LAYER.....	26
2.9.1 <i>Konsep Serving Layer dalam Arsitektur Data Lakehouse</i>	26
2.9.2 <i>Tujuan Perancangan Serving Layer pada Studi Kasus</i>	26
2.9.3 <i>Arsitektur dan Komponen Serving Layer</i>	27
2.9.4 <i>Tujuan Perancangan Serving Layer pada Studi Kasus</i>	27
2.9.5 <i>Posisi Serving Layer dalam Alur Keseluruhan Sistem</i>	28
2.10 PERANCANGAN FRONT END (VISUALISASI DAN OLAP).....	28
2.11 PERANCANGAN FRONT END ANALISIS PRESKRIFTIF.....	31

BAB III.....	32
3.1. DEVELOPMENT ENVIRONMENT	32
3.2. IMPLEMENTASI DATA STORAGE	34
3.2.1. <i>Konfigurasi Arsitektur Medallion (MinIO)</i>	34
3.3. IMPLEMENTASI PEMROSESAN DATA (ELT PIPELINE)	35
3.3.1. <i>Raw Layer: Akuisisi Data Heterogen</i>	35
3.3.2. <i>Silver Layer: Transformasi dan Delta Lake</i>	38
3.3.3. <i>Gold Layer: Logika Analisis Preskriptif</i>	42
3.3.4. <i>Serving Layer (Neon DB)</i>	46
3.4. IMPLEMENTASI ORKESTRASI SISTEM	49
3.5. IMPLEMENTASI VISUALISASI DASHBOARD.....	50
3.5.1 <i>Indikator Utama Keputusan (Key Performance Indicator)</i>	50
3.5.2 <i>Visualisasi Keputusan dan Faktor Penentu</i>	52
3.5.3 <i>Analisis Tren dan Riwayat Historis</i>	53
BAB IV	57
4.1. PERSIAPAN LINGKUNGAN SISTEM	57
4.2. INISIALISASI DAN MENJALANKAN INFRASTRUKTUR SISTEM.....	58
4.3. KONFIGURASI AWAL DAN AKSES LAYANAN.....	58
4.4. EKSEKUSI PIPELINE DATA LAKEHOUSE.....	59
4.5. EKSEKUSI PIPELINE DATA LAKEHOUSE.....	61
4.6. VISUALISASI DAN HASIL ANALISIS PRESKRIFTIF	61
BAB V	63
5.1. INSIGHT IMPLEMENTASI	63
5.2. INSIGHT DARI DASHBOARD.....	63
DAFTAR PUSTAKA	64
LAMPIRAN	65

Daftar Gambar

<i>Gambar 2.1 Rancangan Arsitektur Data Lakehouse.....</i>	11
<i>Gambar 2.2 Relasi Tabel Aktivitas.....</i>	12
<i>Gambar 2.3 Rancangan Pipeline ELT dan ETL</i>	18
<i>Gambar 2.4 Tahap Ingest pada ELT</i>	19
<i>Gambar 2.5 Proses Transform (ETL) Data Ke Clean Zone</i>	20
<i>Gambar 2.6 Proses Transform (ETL) Data Ke Curated Zone</i>	23
<i>Gambar 2.7 Diagram Alur Pengambilan Metadata Delta Lake</i>	23
<i>Gambar 2.8 Schema yang akan digunakan oleh Dashboard</i>	27
<i>Gambar 2.9 Posisi serving layer dalam Arsitektur Data Lakehouse</i>	28
<i>Gambar 2.10 Rancangan Front End</i>	29
<i>Gambar 2.11 Rancangan Front End Preskriptif.....</i>	31
<i>Gambar 3.12 Konfigurasi Environment Airflow.....</i>	32
<i>Gambar 3.13 Konfigurasi Layanan MinIO</i>	32
<i>Gambar 3.14 Konfigurasi Metabase dan PostgreSQL</i>	33
<i>Gambar 3.15 Environment Docker</i>	34
<i>Gambar 3.16 Otomasi Pembuatan Bucket MinIO.....</i>	34
<i>Gambar 3.17 Bucket MinIO</i>	35
<i>Gambar 3.18 Sumber Data Sheet</i>	35
<i>Gambar 3.19 Fungsi Membaca Sheet</i>	36
<i>Gambar 3.20 Ingest Api</i>	36
<i>Gambar 3.21 Ingest SQL</i>	37
<i>Gambar 3.22 Bucket raw-zone MinIO</i>	37
<i>Gambar 3.23 Bucket raw-zone MinIO</i>	38
<i>Gambar 3.24 Cleaning Sheet.....</i>	38
<i>Gambar 3.25 Normalisasi Data</i>	39
<i>Gambar 3.26 Cleaning SQL.....</i>	40
<i>Gambar 3.27 Menyimpan Ke Clean Bucket</i>	40
<i>Gambar 3.28 Menyimpan Ke Clean Bucket</i>	41
<i>Gambar 3.29 Bucket clean-zone MinIO.....</i>	41
<i>Gambar 3.30 Bucket clean-zone MinIO.....</i>	41
<i>Gambar 3.31 Fungsi Membaca Parquet</i>	42
<i>Gambar 3.32 Fungsi Membaca Parquet</i>	42
<i>Gambar 3.33 Logic Aktivitas Terakhir.....</i>	43
<i>Gambar 3.34 Logic Skor Kotor.....</i>	43
<i>Gambar 3.35 Skor Bau Badan.....</i>	44
<i>Gambar 3.36 Logic Skor Final.....</i>	45
<i>Gambar 3.37 Menyimpan Hasil Logic.....</i>	45
<i>Gambar 3.38 Bucket curated-zone MinIO.....</i>	46
<i>Gambar 3.39 Load_aktivitas_to_neon.py</i>	47
<i>Gambar 3.40 Load_prescriptive_to_sql.py</i>	48
<i>Gambar 3.41 Load_riwayat_mandi_to_neon.py</i>	48
<i>Gambar 3.42 loading ke NeonDB</i>	49
<i>Gambar 3.43 Apache Airflow DAG Graph View.....</i>	50
<i>Gambar 3.44 Query SQL</i>	50
<i>Gambar 3.45 Query SQL</i>	51

<i>Gambar 3.46 Query SQL</i>	51
<i>Gambar 3.47 Query SQL</i>	51
<i>Gambar 3.48 Hasil Visual.....</i>	51
<i>Gambar 3.49 Query SQL</i>	52
<i>Gambar 3.50 Query SQL</i>	52
<i>Gambar 3.51 Hasil Visual.....</i>	52
<i>Gambar 3.52 Query SQL</i>	53
<i>Gambar 3.53 Hasil Visual.....</i>	53
<i>Gambar 3.54 Query SQL</i>	54
<i>Gambar 3.55 Hasil Visual.....</i>	54
<i>Gambar 3.56 Query SQL</i>	55
<i>Gambar 3.57 Hasil Visual.....</i>	55
<i>Gambar 3.58 Query SQL</i>	55
<i>Gambar 3.59 Hasil Visual.....</i>	56
<i>Gambar 4.60 Struktur Direktori Project</i>	57
<i>Gambar 4.61 Container Apache Airflow, MinIO, PostgreSQL.....</i>	58
<i>Gambar 4.62 Halaman Object Storage MinIO</i>	59
<i>Gambar 4.63 Apache Airflow Dag</i>	60
<i>Gambar 4.64 Halaman Graph Apache Airflow</i>	60
<i>Gambar 4.65 Database PostgreSQL NeonDB.....</i>	60
<i>Gambar 4.66 Hasil Eksekusi Pipeline di curated-zone</i>	61
<i>Gambar 4.67 Halaman Dashboard Metabase</i>	62

Daftar Tabel

<i>Tabel 2.1 Alat Implementasi</i>	10
<i>Tabel 2.2 Kategori Lingkungan Aktivitas</i>	12
<i>Tabel 2.3 Daftar Aktivitas.....</i>	12
<i>Tabel 2.4 Catatan Aktivitas.....</i>	13
<i>Tabel 2.5 Log Mandi</i>	13
<i>Tabel 2.6 Contoh Data API BMKG</i>	13
<i>Tabel 2.7 Contoh Data API AQICN</i>	14
<i>Tabel 2.8 Jadwal Pengambilan Data</i>	15
<i>Tabel 2.9 Contoh Data Raw Zone Aktivitas Harian</i>	22
<i>Tabel 2.10 Contoh Data Clean Zone Master Aktivitas Harian</i>	22
<i>Tabel 2.11 Contoh Data Curated Zone Hasil Prescriptive</i>	25
<i>Tabel 4.12 Perintah Kloning Repotori.....</i>	57
<i>Tabel 4.13 Perintah Untuk Menjalankan Sistem</i>	58
<i>Tabel 4.14 Perintah Untuk Restore Dashboard Metabase</i>	61

BAB I

1.1. Latar Belakang

Kebersihan diri merupakan aspek fundamental kesehatan yang seringkali dikelola secara intuitif tanpa parameter yang terukur. Keputusan untuk mandi umumnya hanya didasarkan pada kebiasaan waktu atau perasaan subjektif semata. Padahal, kebutuhan tubuh untuk dibersihkan bersifat sangat dinamis dan dipengaruhi oleh akumulasi variabel kompleks, mulai dari intensitas aktivitas fisik, kondisi cuaca, hingga paparan polutan lingkungan. Pendekatan konvensional yang mengabaikan variabel objektif ini seringkali tidak akurat.

Ketidakakuratan tersebut menimbulkan dua dampak negatif utama. Pertama, risiko kesehatan dermatologis akibat *over-washing*. Jika seseorang mandi saat tubuh sebenarnya masih bersih, paparan sabun dan air yang tidak perlu dapat merusak *stratum corneum* dan mengganggu keseimbangan mikrobioma kulit (Skowron et al., 2021). Kedua, dampak lingkungan berupa pemborosan air. Studi menyoroti bahwa aktivitas mandi berkontribusi signifikan terhadap konsumsi air rumah tangga, dan efisiensi sangat bergantung pada pemantauan serta visualisasi data penggunaan akhir atau *end-use* (Otaki et al., 2020). Tanpa indikator yang jelas kapan tubuh benar-benar "kotor", penggunaan air menjadi tidak efisien.

Untuk mengobjektifikasi keputusan ini, diperlukan integrasi data dari berbagai dimensi kehidupan sehari-hari yang saat ini masih terfragmentasi (*siloed*). Dimensi historis seperti riwayat mandi dan log aktivitas tersimpan dalam basis data (SQL). Dimensi preferensi pribadi, seperti jadwal harian dan *scoring* rutinitas manual, dikelola dalam *spreadsheet* (Google Sheets). Dimensi cuaca *real-time* (suhu dan hujan) tersedia melalui API publik (BMKG). Selain itu, terdapat dimensi kualitas lingkungan fisik berupa data tingkat polusi udara (Air Quality Index/AQI). Data ini krusial karena paparan partikel debu dan polutan saat beraktivitas di luar ruangan secara langsung mempengaruhi akumulasi kotoran pada kulit, yang diperoleh dari sumber data eksternal.

Tantangan teknis muncul dalam mengintegrasikan keempat sumber data dengan format heterogen tersebut (Terstruktur dan Semi-terstruktur). Pendekatan tradisional seringkali gagal menangani variasi aliran data ini secara efisien. Oleh karena itu, penerapan arsitektur Data Lakehouse menjadi solusi yang urgen. Secara fundamental, *Data Lakehouse* merupakan evolusi arsitektur yang menggabungkan fleksibilitas penyimpanan berbiaya rendah dari *data lake* dengan kemampuan manajemen data dan integritas transaksional (ACID) dari *data warehouse* tradisional (Armbrust et al., 2021). Tidak seperti pendekatan sebelumnya yang memisahkan data ke dalam dua repositori berbeda (*silo*), arsitektur ini mampu mengatasi keterbatasan tersebut dengan mendukung pengelolaan berbagai tipe data secara simultan dalam satu platform analitik yang terpadu (Harby & Zulkernine, 2022).

Sistem *Smart Hygiene Decision Support System* ini dirancang untuk memberikan analisis preskriptif. Sistem akan mengolah seluruh data tersebut untuk menghasilkan keputusan cerdas: apakah pengguna harus segera mandi karena akumulasi aktivitas fisik dan paparan polusi udara tinggi, atau menunda mandi karena kondisi tubuh masih terjaga kebersihannya berdasarkan preferensi yang telah dipetakan.

1.2. Project Scope

Agar pengembangan sistem tetap terarah dan terukur, ruang lingkup penelitian dan pembangunan *Smart Hygiene Decision Support System* ini dibatasi pada aspek-aspek berikut:

1. Sumber Data Terintegrasi Sistem hanya akan mengintegrasikan data dari empat sumber spesifik untuk mewakili dimensi internal dan eksternal pengguna, yaitu:
 - a. **Data Internal (SQL Database):** Mengelola data transaksional historis berupa kegiatan aktivitas tersedia dengan jenis kegiatan.
 - b. **Data Log (Google Sheets):** Mengelola data semi-terstruktur yang berisi log mandi, dan log aktivitas.
 - c. **Data Lingkungan Meteorologis (API BMKG):** Mengambil data cuaca *real-time* (suhu udara dan kelembapan) untuk mengestimasi laju penguapan keringat.
 - d. **Data Lingkungan Fisik (API Kualitas Udara/AQI):** Mengambil data indeks kualitas udara (termasuk PM2.5/PM10) sebagai indikator paparan polusi dan debu saat pengguna beraktivitas di luar ruangan.
2. Arsitektur Data Lakehouse Implementasi teknis difokuskan pada penerapan arsitektur *Data Lakehouse* yang menggabungkan kapabilitas penyimpanan data heterogen (*data lake*) dengan manajemen struktur data (*data warehouse*). Fokus utama adalah proses *Ingestion*, penyimpanan (*Storage*), dan transformasi data dari keempat sumber di atas ke dalam satu repositori terpusat.
3. Sistem dirancang untuk menghasilkan analisis deskriptif (visualisasi kondisi saat ini) dan analisis preskriptif berupa rekomendasi biner atau bertingkat (misalnya: "Segera Mandi", "Bisa Ditunda", atau "Tidak Perlu Mandi") berdasarkan ambang batas (*threshold*) skor kebersihan yang telah dihitung sistem.
4. Skala Implementasi Sistem ini dibangun sebagai *prototype* untuk penggunaan skala individu, yang bertujuan untuk menguji efektivitas algoritma pengambilan keputusan kebersihan dan efisiensi frekuensi mandi.

BAB II

2.1 Urgensi Perancangan Sistem Berbasis Data Lakehouse

Perkembangan sistem analitik modern ditandai dengan meningkatnya kebutuhan integrasi data yang berasal dari berbagai sumber dengan karakteristik yang berbeda. Data operasional yang tersimpan dalam basis data relasional, data semi terstruktur dari spreadsheet, serta data eksternal dari layanan API menghadirkan tantangan tersendiri dalam hal konsistensi, fleksibilitas skema, dan kesiapan analitik. Penelitian terdahulu menunjukkan bahwa arsitektur data warehouse konvensional memiliki keterbatasan dalam menangani variasi format data dan perubahan kebutuhan analitik, sementara pendekatan data lake murni sering menghadapi permasalahan kualitas data dan kurangnya mekanisme manajemen metadata yang kuat (Ait Errami et al., 2023).

Sebagai respons terhadap keterbatasan tersebut, arsitektur data lakehouse dikembangkan dengan menggabungkan fleksibilitas penyimpanan data mentah dari data lake dan kemampuan manajemen data yang terstruktur dari data warehouse. Pendekatan ini memungkinkan penyimpanan dan pengolahan data heterogen dalam satu sistem terintegrasi tanpa memerlukan pemisahan fisik antara data mentah dan data analitik (Harby, 2025).

Dalam penelitian ini, sistem dan dashboard keputusan mandi dirancang sebagai sistem analitik preskriptif yang mengintegrasikan data historis log mandi, aktivitas harian, serta data lingkungan eksternal seperti cuaca dan kualitas udara. Karakteristik data yang heterogen, bertambah secara kontinu, dan bergantung pada dimensi waktu menuntut arsitektur penyimpanan yang fleksibel sekaligus mendukung pengelolaan data analitik secara berkelanjutan. Literatur mencatat bahwa arsitektur data lakehouse dikembangkan untuk mengintegrasikan data mentah dan data terkurasi dalam satu ekosistem analitik dengan tetap menjaga keterlacakkan data sumber (Ait Errami et al., 2023).

Urgensi penggunaan data lakehouse pada sistem ini terletak pada kebutuhan menjaga konsistensi hasil analitik preskriptif ketika data bertambah atau proses analitik perlu disesuaikan. Dashboard membutuhkan data terkurasi yang siap dikonsumsi, sementara sistem tetap memerlukan akses terhadap data mentah untuk evaluasi dan penyesuaian logika keputusan. Pendekatan lakehouse menyediakan kerangka penyimpanan data berlapis dan pengelolaan metadata yang lebih terstruktur dibandingkan data lake murni, sehingga mendukung keberlanjutan analitik tanpa rekonstruksi pipeline secara menyeluruh (Harby, 2025).

2.2 Prinsip Arsitektur Data Lakehouse yang Digunakan

Arsitektur data lakehouse pada penelitian ini dirancang berdasarkan prinsip utama unified storage, schema flexibility, dan analytical readiness. Prinsip unified storage memungkinkan seluruh data dari berbagai sumber disimpan dalam satu sistem penyimpanan tanpa pemisahan fisik antara data lake dan data warehouse, sehingga mengurangi redundansi data dan kompleksitas pipeline (Harby, 2025).

Prinsip schema flexibility memungkinkan data mentah dimasukkan ke sistem tanpa harus didefinisikan skemanya secara ketat pada tahap awal, sehingga mendukung ingestion data eksternal seperti API cuaca atau API kualitas udara. Skema baru diterapkan secara bertahap pada fase transformasi untuk memastikan data siap digunakan dalam analitik (Ait Errami et al., 2023).

Prinsip analytical readiness memastikan bahwa data yang telah melalui proses kurasi dapat langsung digunakan oleh engine analitik tanpa perlu replikasi atau migrasi tambahan, yang merupakan salah satu keunggulan utama lakehouse dibandingkan arsitektur tradisional (Harby, 2025).

2.3 Alat Implementasi

Berdasarkan implementasi pada kode program, berikut adalah daftar tools beserta fungsinya:

Tabel 2.1 Alat Implementasi

Tool	Fungsi
Google Sheets	Sumber data input untuk aktivitas harian pengguna.
NeonDB	Database PostgreSQL <i>cloud-native</i> yang digunakan sebagai sumber data dan <i>serving layer</i> .
Python	Bahasa pemrograman utama untuk <i>extract, load, transform</i> dan Logika Preskriptif.
Visual Studio Code	<i>Integrated Development Environment</i> (IDE) untuk pengembangan kode.
MinIo	<i>Object Storage</i> yang berfungsi sebagai Data Lake untuk menyimpan data mentah.
Delta Lake	Format penyimpanan pada Data Lake yang memungkinkan fitur ACID <i>transactions</i> dan <i>time travel</i> .
Metabase	<i>Business Intelligence</i> (BI) tool untuk visualisasi data dan dashboard.
Airflow	<i>Orchestrator</i> untuk mengatur jadwal dan alur kerja (DAGs) seluruh pipeline data.
Docker	Kontainerisasi untuk menjalankan layanan Airflow, MinIO, dan Metabase secara terintegrasi.

2.4 Perancangan Arsitektur Data Lakehouse

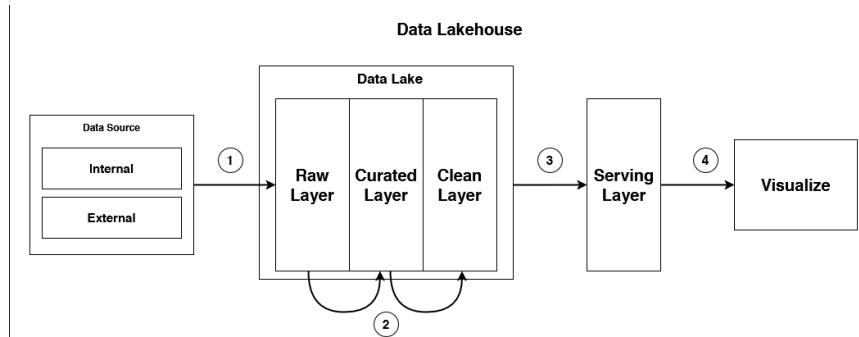
Arsitektur data lakehouse pada penelitian ini terdiri dari lapisan sumber data, lapisan ingest, lapisan penyimpanan terpusat, dan lapisan analitik. Seluruh data dari database SQL, Google Sheets, dan API eksternal dikumpulkan ke dalam satu sistem penyimpanan lakehouse yang mendukung data mentah dan data terkurasi secara bersamaan.

Pendekatan ini memungkinkan pengolahan data dilakukan secara menyeluruh dalam satu ekosistem tanpa pemisahan fisik antara sistem operasional dan sistem analitik. Penelitian menyatakan bahwa integrasi semacam ini merupakan karakteristik utama arsitektur lakehouse modern dan berkontribusi pada efisiensi serta konsistensi analitik (Ait Errami et al., 2023; Harby, 2025).

Implementasi arsitektur ini didukung oleh dua teknologi utama sebagai fondasi operasional: Docker dan Apache Airflow.

Docker: Digunakan untuk melakukan kontainerisasi seluruh layanan (Airflow, MinIO, Metabase, dan Postgres). Hal ini memastikan bahwa lingkungan pengembangan dan produksi bersifat konsisten, portabel, dan mudah dikelola dalam satu jaringan virtual yang terintegrasi.

Apache Airflow: Bertugas sebagai dirigen atau orkestrator yang mengatur seluruh alur kerja (workflow). Airflow menjamin bahwa setiap tahap, mulai dari pengambilan data hingga analisis akhir, berjalan sesuai urutan dependensi yang benar dan otomatis.



Gambar 2.1 Rancangan Arsitektur Data Lakehouse

1. Ingest :

Proses dimulai dengan penarikan data dari berbagai sumber, baik Internal seperti database operasional maupun External melalui API. Mekanisme ini dapat dijalankan secara batch untuk data historis atau streaming untuk kebutuhan real-time. Seluruh data tersebut kemudian mendarat di Raw Layer di dalam Data Lake dalam format aslinya. Hal ini sangat krusial untuk menjaga data fidelity, sehingga jika terjadi kesalahan proses di masa mendatang, Anda selalu memiliki salinan asli untuk diproses ulang.

2. Transform :

Setelah data terkumpul, proses transformasi dilakukan secara bertahap mengikuti pola Medallion Architecture. Di sini, penggunaan format Parquet menjadi standar karena sifatnya yang *columnar*, sehingga sangat efisien dalam kompresi penyimpanan dan mempercepat kueri pada volume data besar.

Data bergerak dari Raw Layer menuju Curated Layer, di mana dilakukan pembersihan seperti penghapusan duplikasi dan penanganan nilai yang hilang. Tahap akhir dari transformasi ini adalah Clean Layer, di mana data sudah dianggap "matang". Pada lapisan ini, logika bisnis diterapkan dan data dari berbagai sumber digabungkan untuk membentuk satu sumber kebenaran.

3. Model :

Data yang telah bersih di Data Lake kemudian dipindahkan ke Serving Layer melalui proses ETL. Fokus utama di tahap ini adalah Modeling, di mana data disusun menggunakan skema yang dioptimalkan untuk *dashboard*.

4. Analytics :

Tahap final adalah konsumsi data oleh pengguna akhir melalui lapisan Visualize. Alat BI dihubungkan langsung ke Serving Layer untuk mengekstrak nilai dari data. Hasil akhirnya adalah dashboard interaktif dan laporan performa yang memungkinkan pemangku kepentingan melakukan analisis prediktif maupun

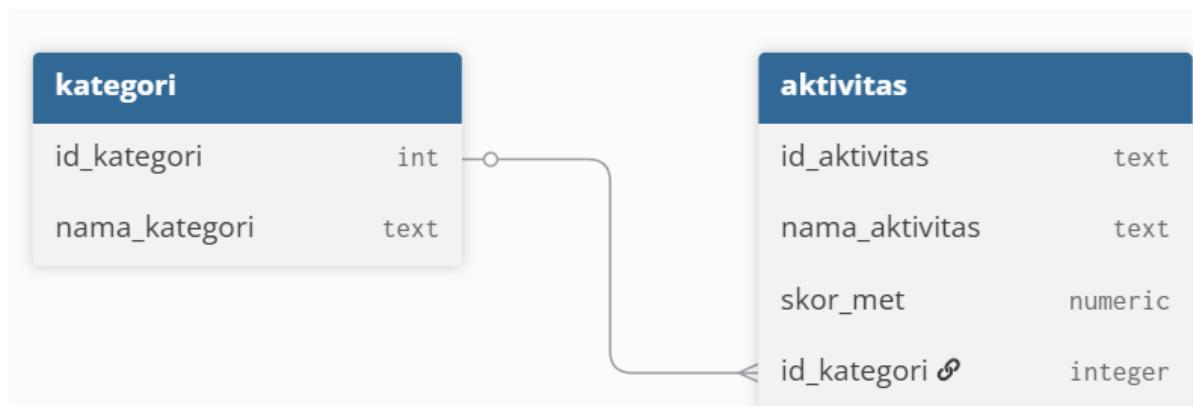
deskriptif. Proses ini mengubah baris data mentah yang kompleks menjadi wawasan visual yang mudah dipahami untuk mendukung keputusan.

2.5 Dataset yang Dibutuhkan dan Dikumpulkan

2.5.1 Database Terstruktur (SQL)

Dataset terstruktur digunakan untuk menyimpan daftar_aktivitas karena data ini bersifat operasional, berulang, dan membutuhkan konsistensi serta integritas data yang tinggi. Basis data relasional dipilih karena mampu merepresentasikan peristiwa aktivitas secara temporal melalui pencatatan waktu dan atribut kondisi tubuh secara sistematis. Literatur menyatakan bahwa data operasional yang bersifat historis paling efektif dikelola menggunakan sistem relasional sebelum diintegrasikan ke platform analitik yang lebih luas (Ait Errami et al., 2023).

Contoh struktur tabel aktivitas dan kategori:



Gambar 2.2 Relasi Tabel Aktivitas

Tabel 2.2 Kategori Lingkungan Aktivitas

id_kategori	nama_kategori
1	Indoor
2	Outdoor

Tabel 2.3 Daftar Aktivitas

Id_aktivitas	nama_aktivitas	Skor_met	Id_kategori
A01	Naik motor	3.5	2
A02	Jalan kaki	3.3	2
A03	lari	10.0	2
A04	gym	6.0	1

Atribut utama pada dataset ini meliputi nama aktivitas, Skor MET (Metabolic Equivalent of Task), dan lingkungan aktivitas tersebut dilakukan. Data tersebut berfungsi sebagai fondasi historis dalam analitik preskriptif karena memungkinkan perhitungan interval waktu sejak mandi terakhir dan analisis pola perilaku mandi.

2.5.2 Data Semi-Terstruktur (Google Sheets)

Dataset semi terstruktur digunakan untuk menyimpan data aktivitas harian dan preferensi pengguna. Spreadsheet dipilih karena fleksibilitasnya dalam pencatatan manual dan kemudahan pembaruan data oleh pengguna. Dalam arsitektur lakehouse modern, spreadsheet dipandang sebagai sumber data semi terstruktur yang sah apabila diproses melalui mekanisme pembersihan dan transformasi yang sistematis (Ait Errami et al., 2023).

a. Catatan Aktivitas

Tabel 2.4 Catatan Aktivitas

timestamp	id_aktivitas	durasi_menit
2025-01-01 08:00	A01	40
2025-01-01 10:30	A02	30
2025-01-01 13:00	A04	120

b. Log Mandi

Tabel 2.5 Log Mandi

Waktu_mandi	Tingkat_kekotoran	Tingkat_bau_badan
2025-12-13 07:00	4	3
2025-12-13 19:30	8	7
2025-12-14 10:30	3	5
2025-12-14 17:50	6	4

Data aktivitas harian berisi jenis aktivitas dan durasi aktivitas yang dilakukan pengguna, sedangkan data preferensi berisi bobot dan ambang batas keputusan yang digunakan dalam perhitungan analitik preskriptif.

2.5.3 Dataset Eksternal (API Lingkungan)

Dataset eksternal digunakan untuk memperoleh data kontekstual yang tidak tersedia dalam sistem internal. Data cuaca dan kualitas udara diperoleh melalui layanan API resmi untuk merepresentasikan kondisi lingkungan di sekitar pengguna. Penelitian menunjukkan bahwa integrasi data eksternal merupakan elemen penting dalam sistem pendukung keputusan karena faktor lingkungan dapat memengaruhi kondisi fisik dan kebutuhan mandi pengguna (Harby, 2025).

A. Contoh Data API BMKG

Tabel 2.6 Contoh Data API BMKG

```
{  
    "datetime": "2025-12-14T00:00:00Z",  
    "t": 26,  
    "tcc": 100,  
    "tp": 0.2,  
    "weather": 60,  
    "weather_desc": "Hujan Ringan",  
    "weather_desc_en": "Light Rain",  
}
```

```

    "wd_deg": 14,
    "wd": "N",
    "wd_to": "S",
    "ws": 8.1,
    "hu": 86,
    "vs": 9999,
    "vs_text": "< 10 km",
    "time_index": "11-12",
    "analysis_date": "2025-12-13T12:00:00",
    "image": "https://api-apps.bmkg.go.id/storage/icon/cuaca/hujan
ringan-pm.svg",
    "utc_datetime": "2025-12-14 00:00:00",
    "local_datetime": "2025-12-14 08:00:00",
    "source": "amandemen"
},

```

B. Contoh Data API AQICN

Tabel 2.7 Contoh Data API AQICN

```

{
  "status": "ok",
  "data": {
    "aqi": 58,
    "idx": -540724,
    "attributions": [
      {
        "url": "https://www.menlhk.go.id/",
        "name": "Kementerian Lingkungan Hidup Dan Kehutanan",
        "station": "kabupaten_banjar"
      },
      {
        "url": "https://waqi.info/",
        "name": "World Air Quality Index Project"
      }
    ],
    "city": {
      "geo": [
        -3.426140069961548,
        114.87999725341797
      ],
      "name": "Kabupaten Banjar",
      "url": "https://aqicn.org/station/@540724",
      "location": "Bincau, Banjar, South Kalimantan, Kalimantan, 70613,
Indonesia"
    },
    "dominentpol": "pm25",
    "iaqi": {
      "co": {
        "v": 0
      },
      "no2": {

```

```

    "v": 5
  },
  "o3": {
    "v": 4
  },
  "pm10": {
    "v": 36
  },
  "pm25": {
    "v": 58
  },
  "so2": {
    "v": 4
  }
},
"time": {
  "s": "2025-12-14 11:00:00",
  "tz": "+08:00",
  "v": 1765681200,
  "iso": "2025-12-14T03:00:00Z"
}
}
}

```

2.6 Scheduling

Tabel 2.8 Jadwal Pengambilan Data

Jenis Data	Frekuensi	Alasan Penjadwalan	Urgensi
Data Cuaca (BMKG API)	Setiap 6 jam	Data cuaca tidak berubah sangat cepat dan bersifat periodik, sehingga interval 6 jam sudah cukup untuk merepresentasikan kondisi lingkungan terkini tanpa membebani API dan infrastruktur.	perubahan cuaca berpengaruh pada perhitungan skor, tetapi tidak membutuhkan pembaruan real-time.
Data Kualitas Udara (AQICN API)	Setiap 6 jam	Nilai AQI bersifat agregat per jam dan relatif stabil dalam rentang waktu pendek, sehingga sinkron dengan jadwal BMKG untuk menjaga konsistensi faktor lingkungan.	berdampak pada skor kebersihan, namun toleran terhadap jeda pembaruan.

Data Aktivitas Manual (Spreadsheet)	Setiap 1 jam	Data ini diinput langsung oleh pengguna, sehingga sistem perlu cukup sering memeriksa perubahan agar rekomendasi mandi tetap relevan dengan aktivitas terbaru.	perubahan aktivitas langsung memengaruhi hasil preskriptif.
Data Log Mandi (Spreadsheet)	Setiap 1 jam	Riwayat mandi menentukan titik awal perhitungan waktu sejak mandi terakhir, sehingga keterlambatan update akan menggeser skor secara signifikan.	data ini menjadi referensi utama penentuan rekomendasi mandi.
Data Aktivitas & Kategori (SQL / NeonDB)	Setiap 1 jam	Data master aktivitas dan kategorinya bersifat structural, tapi data daftar_aktivitas sering berubah jadi tetap perlu disinkronkan secara berkala untuk menjaga konsistensi dengan input pengguna.	menjadi basis integrasi aktivitas pada silver layer.

2.7 Perancangan Zona Penyimpanan Data

Zona penyimpanan data menggunakan MinIO Object Storage, di mana strategi pengorganisasian data menjadi kunci utama untuk menjaga skalabilitas dan keteraturan alur kerja Medallion Architecture. MinIO dipilih karena dukungannya yang optimal terhadap ekosistem Docker, memudahkan orkestrasi layanan dalam satu network yang sama, serta kompatibilitasnya dengan S3 API yang memungkinkan integrasi lancar dengan pustaka Python seperti Boto3 dan Delta Lake.

2.7.1 Raw Zone

Raw zone dirancang untuk menyimpan seluruh data dalam bentuk asli tanpa modifikasi. Praktik ini direkomendasikan dalam literatur lakehouse untuk menjaga data lineage dan memungkinkan validasi ulang terhadap data sumber apabila diperlukan (Ait Errami et al., 2023).

2.7.2 Clean Zone

Clean zone berfungsi sebagai lapisan transisi tempat data melalui proses pembersihan, normalisasi, dan standarisasi. Proses ini penting untuk memastikan kualitas data sebelum digunakan dalam analitik lanjutan, terutama ketika data berasal dari sumber subjektif dan eksternal (Nguyen, 2025).

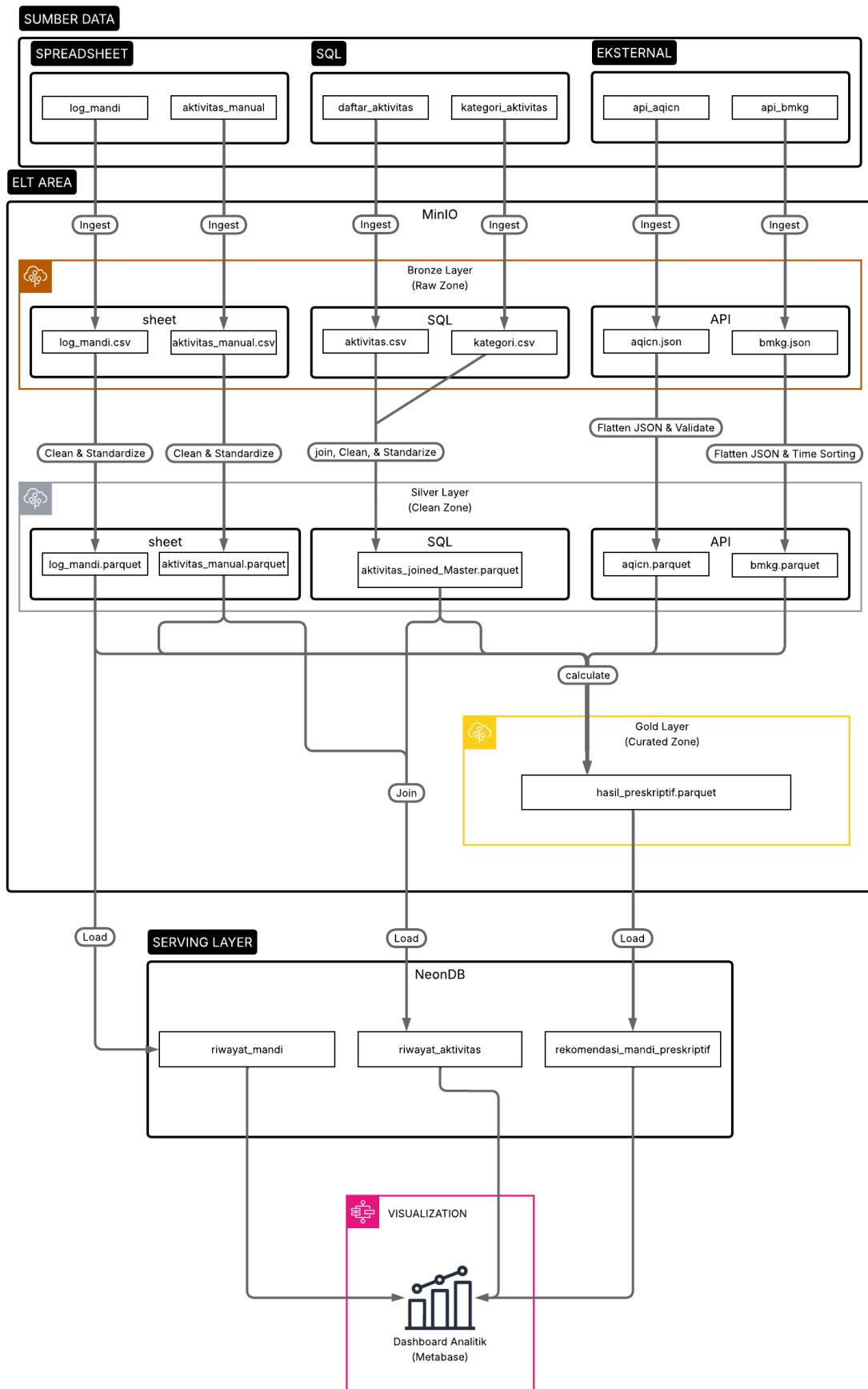
2.7.3 Curated Zone

Curated zone menyimpan data yang telah terintegrasi dan siap digunakan untuk analitik preskriptif. Data pada zona ini disimpan dalam format tabel analitik yang mendukung kueri cepat dan konsistensi transaksi, yang merupakan karakteristik utama dari implementasi lakehouse modern (Harby, 2025).

2.8 Perancangan Pipeline ELT dan ETL

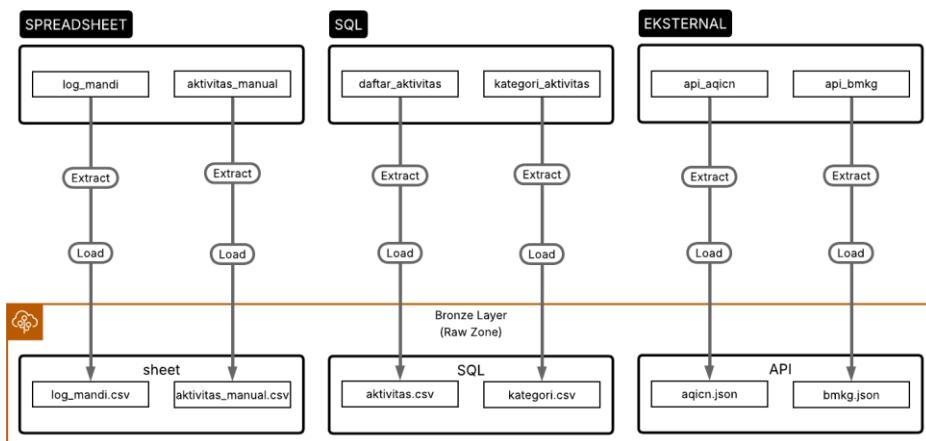
Pendekatan hybrid ELT dan ETL diterapkan karena arsitektur data lakehouse memungkinkan proses transformasi dilakukan pada tahapan yang berbeda sesuai dengan tujuan pengolahan data.

Pendekatan hybrid ini direkomendasikan dalam arsitektur data modern karena memberikan fleksibilitas dalam pengelolaan pipeline serta meminimalkan kebutuhan perubahan ulang terhadap alur pemrosesan ketika kebutuhan analitik berkembang (Ait Errami et al., 2023; Harby, 2025).



Gambar 2.3 Rancangan Pipeline ELT dan ETL

2.8.1 Tahap Ingest Data (Extract dan Load)



Gambar 2.4 Tahap Ingest pada ELT

a. Tahap Extract

Tahap extract pada sistem ini bertujuan untuk mengakuisisi data dari berbagai sumber yang memiliki karakteristik berbeda, yaitu spreadsheet, basis data SQL, serta layanan API eksternal. Sistem dirancang agar setiap sumber data diperlakukan sesuai dengan mekanisme akses yang paling sesuai dengan jenis sumbernya.

Untuk sumber basis data SQL, sistem membangun koneksi langsung ke NeonDB menggunakan SQLAlchemy engine. Melalui koneksi ini, sistem mengeksekusi query terstruktur terhadap tabel pada skema tertentu, seperti tabel `daftar_aktivitas` dan `kategori_aktivitas` pada SQL. Hasil query kemudian dimuat ke dalam objek DataFrame agar dapat diproses secara seragam dengan sumber data lain.

Setelah data berhasil diambil dari database, sistem melakukan serialisasi ke dalam format CSV. Pemilihan format CSV dilakukan karena format ini bersifat ringan, mudah dibaca, dan kompatibel dengan hampir seluruh proses lanjutan di dalam data lakehouse. Setiap file diberi penanda waktu pada nama file, misalnya `aktivitas_20250101_120530.csv`, sehingga setiap proses extract menghasilkan snapshot yang unik dan mudah dilacak berdasarkan waktu pengambilannya.

Untuk sumber API eksternal seperti AQICN dan BMKG, sistem melakukan pemanggilan endpoint API, kemudian mengonversi respons JSON mentah menjadi file JSON tanpa melakukan perubahan struktur. Hal ini memastikan bahwa data yang masuk ke bronze layer benar-benar merepresentasikan kondisi asli dari sumber eksternal.

b. Tahap Extract

Pada tahap load, seluruh hasil extract disimpan ke dalam bucket raw-zone di MinIO yang berfungsi sebagai bronze layer. Penyimpanan dilakukan menggunakan metode `put_object` dari Boto3, dengan pola direktori yang mencerminkan asal data, seperti:

- `sql/aktivitas/aktivitas_20250101_120530.csv`
- `sql/kategori/kategori_20250101_120530.csv`
- `api/bmkg/bmkg_20250101_120530.json`
- `api/aqicn/aqicn_20250101_120530.json`

Struktur direktori ini secara desain sudah menjadi metadata implisit, karena dari path file saja dapat diketahui jenis sumber, nama dataset, dan waktu ingest.

Selain metadata implisit pada path, MinIO juga secara otomatis menyimpan system metadata untuk setiap objek yang diunggah, antara lain:

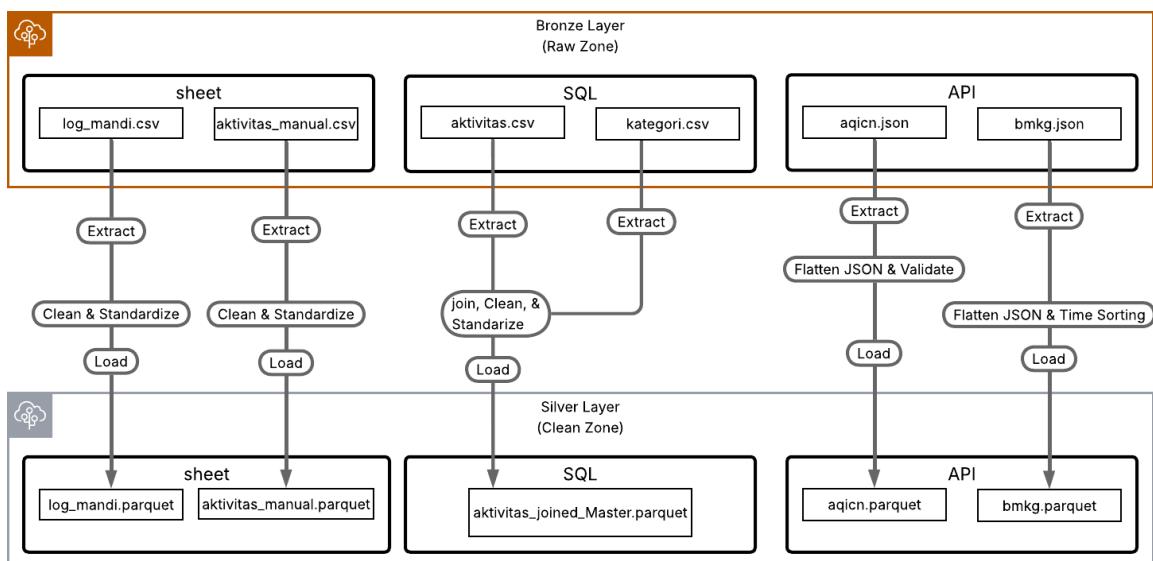
- Key - nama objek lengkap beserta path,
- LastModified - waktu terakhir objek disimpan,
- Size - ukuran file,
- ETag - checksum untuk integritas data,
- ContentType - tipe konten (misalnya text/csv atau application/json).

Metadata sistem inilah yang kemudian dimanfaatkan kembali pada proses clean-zone, khususnya saat sebuah fungsi digunakan untuk mencari file terbaru berdasarkan nilai LastModified. Dengan mekanisme ini, sistem tidak memerlukan pencatatan waktu ingest di luar MinIO, karena informasi tersebut sudah tersedia secara otomatis sebagai metadata objek.

Melalui desain ini, bronze layer tidak hanya berfungsi sebagai tempat penyimpanan data mentah, tetapi juga sebagai lapisan yang menyimpan informasi asal-usul dan versi data secara natural melalui kombinasi antara struktur direktori dan metadata bawaan MinIO.

2.8.2 Tahap Transform (ETL) pada ELT

2.8.2.1 Tahap Transform Bronze Layer ke Silver Layer



Gambar 2.5 Proses Transform (ETL) Data Ke Clean Zone

a. Tahap Extract Bronze Layer Menggunakan Metadata

Pada tahap extract, sistem tidak membaca data langsung dari nama file statis, melainkan menggunakan object key yang tersimpan di MinIO sebagai penunjuk utama lokasi data mentah. Sebuah fungsi akan dijalankan terlebih dahulu membentuk prefix berbasis nama tabel, misalnya sql/aktivitas/ dan sql/kategori/. Prefix ini merepresentasikan struktur direktori logis pada bronze layer.

Melalui pemanggilan fungsi yang akan dirancang, sistem memperoleh daftar objek beserta metadata sistem seperti Key dan LastModified. Dari seluruh objek yang berada pada prefix tersebut, sistem melakukan penyaringan terhadap nilai Key yang berakhiran .csv. Setelah itu, objek diurutkan berdasarkan metadata LastModified, sehingga sistem dapat mengidentifikasi file dengan waktu ingest paling baru.

Hasil dari proses ini adalah sebuah object key spesifik, misalnya:

```
sql/aktivitas/aktivitas_20250101_120530.csv
```

Object key inilah yang kemudian digunakan sebagai parameter utama pada pemanggilan s3.get_object. Dengan demikian, proses extract sepenuhnya bergantung pada kombinasi antara struktur key dan metadata waktu, bukan pada penamaan file yang dikodekan secara manual. Pendekatan ini memastikan bahwa setiap proses clean selalu bekerja pada versi data terbaru yang tersedia di bronze layer tanpa memerlukan intervensi tambahan.

b. Tahap Transformasi data

Tahap transform dirancang sebagai proses untuk mengubah data mentah dari berbagai sumber menjadi dataset yang bersih, terintegrasi, dan memiliki struktur seragam sebelum dimuat ke silver layer. Meskipun setiap sumber data memiliki format yang berbeda, baik CSV dari spreadsheet, CSV hasil extract SQL, maupun JSON dari API eksternal, seluruh data diproses melalui pola transformasi yang serupa.

Sebagai contoh, pada alur data SQL, dataset `daftar_aktivitas` dan `kategori_aktivitas` digabungkan berdasarkan atribut kunci `id_kategori`. Proses ini menghasilkan tabel master yang tidak hanya memuat informasi aktivitas, tetapi juga konteks kategorinya. Pola integrasi ini juga diterapkan secara konseptual pada sumber lain, seperti:

- Data `log_mandi` dan `aktivitas_manual` dari spreadsheet yang dibersihkan dan distandarisasi agar memiliki format waktu dan skema kolom yang seragam.
- Data AQICN dan BMKG dari API yang di-flatten dari struktur JSON menjadi tabel datar agar dapat disejajarkan dengan dataset lain.

Selain integrasi, tahap transform juga mencakup pembersihan tipe data, terutama pada kolom numerik dan waktu. Nilai yang tidak sesuai format diubah menjadi nilai kosong agar tidak mengganggu proses analitik. Dengan demikian, seluruh dataset, meskipun berasal dari sumber yang heterogen, akan memiliki karakteristik struktural yang sebanding ketika memasuki silver layer.

Transformasi ini menjadikan silver layer sebagai lapisan data yang telah bebas dari inkonsistensi format, redundansi struktural, serta perbedaan skema antar sumber.

c. Tahap Load ke Silver Layer

Tahap load pada silver layer dirancang sebagai proses untuk menyimpan seluruh hasil transformasi ke dalam format yang lebih efisien dan terstandarisasi, yaitu **Parquet berbasis Delta Lake**. Perubahan format dari CSV dan JSON pada bronze layer menjadi Parquet pada silver layer bukan sekadar konversi teknis, melainkan keputusan arsitektural untuk meningkatkan kinerja baca, efisiensi penyimpanan, dan konsistensi skema.

Seluruh dataset, baik yang berasal dari SQL, spreadsheet seperti `log_mandi` dan `aktivitas_manual`, maupun API eksternal seperti AQICN dan BMKG, dimuat ke dalam struktur direktori silver layer dengan pola penamaan yang merepresentasikan jenis data. Contohnya, dataset hasil integrasi aktivitas disimpan sebagai `aktivitas_joined_master.parquet`, sementara data hasil normalisasi dari API disimpan sebagai `aqicn.parquet` dan `bmkg.parquet`.

Pendekatan ini memastikan bahwa setiap entitas data pada silver layer sudah berada dalam bentuk tabel kolumnar yang teroptimasi, sehingga tidak lagi bergantung pada format sumber aslinya. Selain itu, silver layer diperlakukan sebagai lapisan data bersih yang selalu merefleksikan kondisi terbaru dari bronze layer. Oleh karena itu, setiap proses load dilakukan dengan strategi pembaruan penuh, sehingga versi data lama secara sistematis digantikan oleh versi hasil transformasi terkini.

Dengan desain ini, silver layer berfungsi sebagai titik temu seluruh sumber data yang telah distandarisasi, sehingga lapisan berikutnya dapat memanfaatkan dataset Parquet tersebut tanpa perlu kembali menangani perbedaan format atau struktur data mentah.

Contoh perubahan data pada clean zone :

- **Raw Zone:** Aktivitas Harian (sebelum transformasi)

Tabel 2.9 Contoh Data Raw Zone Aktivitas Harian

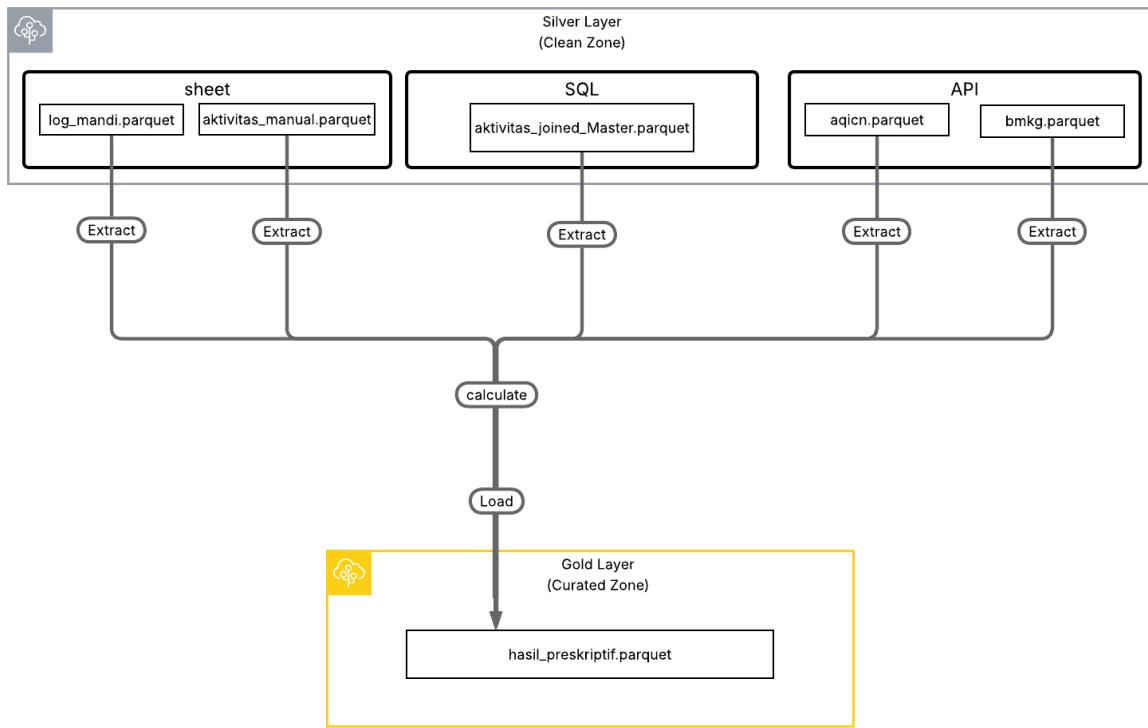
timestamp	id_aktivitas	durasi_menit
2025-01-01 08:00	A01	40

- **Clean Zone:** Aktivitas Harian (sesudah transformasi dengan Daftar Aktivitas)

Tabel 2.10 Contoh Data Clean Zone Master Aktivitas Harian

timestamp	id_aktivitas	nama_aktivitas	skor_met	kategori	durasi_menit
2025-01-01 08:00	A01	Naik motor	3.5	Outdoor	40

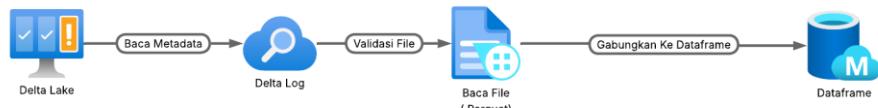
2.8.2.2 Tahap Transform Silver Layer ke Gold Layer



Gambar 2.6 Proses Transform (ETL) Data Ke Curated Zone

Tahap curated zone merupakan tahap akhir dalam pipeline ELT yang digambarkan pada penelitian ini. Pada tahap ini, data tidak hanya dibersihkan, tetapi juga diintegrasikan dan diperkaya untuk membentuk informasi yang mendukung pengambilan keputusan.

a. Tahap Extract Silver Layer Berbasis Metadatas Delta Lake



Gambar 2.7 Diagram Alur Pengambilan Metadatas Delta Lake

Pada tahap extract, sistem tidak lagi membaca file Parquet secara langsung, melainkan memanfaatkan **metadatas internal Delta Lake**. Setiap tabel di silver layer, seperti sheets/catatan_aktivitas, sql/aktivitas_joined_master, api/bmkg, dan api/aqi, dibuka menggunakan objek DeltaTable.

Saat fungsi DeltaTable dipanggil, **Delta Lake membaca file log transaksi _delta_log** yang berada pada direktori tabel tersebut. File log ini berisi informasi penting seperti:

- daftar file Parquet aktif,
- versi snapshot terbaru,
- skema tabel,

- riwayat perubahan data.

Metadata inilah yang memungkinkan sistem untuk selalu mendapatkan versi data terkini tanpa perlu memindai seluruh direktori secara manual. Ketika metode `to_pandas()` dipanggil, Delta Lake menggunakan metadata tersebut untuk menentukan file Parquet mana yang valid pada snapshot terakhir, lalu memuatnya secara otomatis ke dalam DataFrame Pandas.

Dengan mekanisme ini, pengambilan data dari silver layer tidak lagi bergantung pada penamaan file, melainkan pada versi snapshot yang disimpan dalam metadata Delta Lake.

b. Tahap Transform - Pembentukan Skor Preskriptif

• Logika Skor Kotor

Perancangan skor kekotoran diimplementasikan untuk menghasilkan nilai akumulasi kotoran yang dinamis dan berbasis data lingkungan. Proses dimulai dengan inisialisasi nilai awal sebesar nol dan pengecekan validitas data pada jendela aktivitas serta tabel master aktivitas sebelum dilakukan penggabungan data. Sistem kemudian menghitung faktor lingkungan secara otomatis dengan memproses data suhu terkini dari BMKG dan indeks kualitas udara atau AQI melalui rumus berikut:

$$Faktor\ Outdoor = \max(1.0, \left(\frac{Suhu\ Udara}{25} \times 0.6 \right) + \left(\frac{Skor\ AQI}{50} \times 0.4 \right))$$

Penetapan nilai minimal sebesar 1.0 pada rumus tersebut bertujuan agar faktor lingkungan tidak memberikan efek pengurangan pada skor dasar aktivitas. Selanjutnya, setiap baris aktivitas diproses secara individual untuk menentukan skor dasarnya menggunakan perhitungan:

$$Base = Durasi\ Menit \times \frac{Skor\ MET}{10}$$

Apabila aktivitas terdeteksi dilakukan di luar ruangan atau memiliki kategori "outdoor", sistem akan mengalikan base_score dengan faktor_outdoor. Sebaliknya, untuk aktivitas di dalam ruangan, sistem hanya menggunakan nilai base_score saja. Setelah seluruh skor individu selesai dihitung, sistem melakukan akumulasi total dan normalisasi akhir dengan rumus:

$$Skor\ Kekotoran = \min\left(\frac{\sum Skor\ Individu}{15}, 10\right)$$

Langkah penutup ini memastikan bahwa hasil akhir yang ditampilkan pada dashboard memiliki rentang nilai antara 0 hingga 10 poin.

• Logika Skor Bau

Untuk mengukur tingkat ketidaknyamanan aroma tubuh yang dipicu oleh aktivitas fisik, durasi waktu, dan kondisi atmosfer.

$$\begin{aligned} Skor\ Bau &= (Jam\ Sejak\ Mandi \times 0.3) + (Jumlah\ Aktivitas\ Bau \times 0.7) \\ &\quad + (Faktor\ Lembap \times 2) \end{aligned}$$

Jam Sejak Mandi (Bobot 30%): Memberikan 0.3 poin untuk setiap jam yang berlalu.

Jumlah Aktivitas Bau (Bobot 70%): Menambahkan 0.7 poin untuk setiap aktivitas dengan MET > 3.0 atau aktivitas *outdoor* yang memicu keringat.

Faktor Lembap: Menggunakan persentase kelembapan dari BMKG (kelembapan/100) yang berkontribusi hingga maksimal 2.0 poin karena mempercepat pertumbuhan bakteri.

- **Logika Skor Final**

$$Skor\ Final = (Skor\ Kekotoran \times 0.4) + (Skor\ Bau \times 0.4) + (Skor\ AQI \times 0.2)$$

Skor AQI: Skor AQI Menggunakan skor kualitas udara dari AQICN (AQI/50) yang dinormalisasi jadi skor 1-10.

Logika Ambang Batas (*Threshold*): Sistem kemudian membandingkan skor final dengan THRESHOLD = 6.0:

Skor \geq 6.0: Wajib Mandi Sekarang (Kondisi Kritis).

Skor \geq 5.0: Sangat Disarankan (Kondisi Tidak Nyaman).

Skor \geq 4.0: Mandi Bisa Ditunda (Kondisi Masih Oke).

Skor $<$ 4.0: Tidak Perlu Mandi (Kondisi Prima).

Selain itu, terdapat Safety Catch di mana jika Skor Bau \geq 9.0, sistem secara otomatis mengeluarkan perintah wajib mandi tanpa mempertimbangkan parameter lainnya demi menjaga standar higienitas pengguna.

Transformasi data pada curated zone menghasilkan data dengan makna semantik yang lebih tinggi. Data tidak lagi merepresentasikan aktivitas, cuaca, atau mandi secara terpisah, melainkan kondisi kebersihan tubuh pengguna pada waktu tertentu.

c. Load ke Curated Zone

Tahap load dilakukan dengan menyimpan hasil analisis preskriptif ke dalam bucket curated-zone. Seluruh data curated disimpan dalam format Delta Lake, yang secara fisik berupa file Parquet, sehingga tetap mendukung kueri analitik cepat dan konsistensi transaksi.

Contoh data pada curated zone (.parquet) :

Tabel 2.11 Contoh Data Curated Zone Hasil Prescriptive

waktu	skor_Kekotoran	Skor Bau	aqi	jam_sejak_mandi	skor_kebersihan	Rekomendasi
2025-12-14 12:00	5.8	6	1.16	5.5	4.95	Mandi Bisa Ditunda

2.9 Perancangan Serving Layer

2.9.1 Konsep Serving Layer dalam Arsitektur Data Lakehouse

Dalam arsitektur data lakehouse, keberadaan serving layer tidak dapat dipahami sekadar sebagai lapisan tambahan setelah proses pengolahan data selesai, melainkan sebagai titik temu antara data terkuras dan kebutuhan pengambilan keputusan berbasis analitik. Serving layer dirancang untuk menjembatani kompleksitas data analitik dengan antarmuka yang mudah diakses oleh pengguna akhir, khususnya dalam konteks sistem pendukung keputusan dan visualisasi bisnis. Tanpa lapisan ini, data yang telah melalui proses pembersihan dan integrasi hanya akan berhenti sebagai aset teknis, tanpa mampu memberikan nilai praktis bagi proses evaluasi dan penarikan insight.

Secara konseptual, serving layer berfungsi untuk menyajikan data dari curated zone dalam bentuk struktur yang siap dikueri secara efisien, baik melalui mekanisme tabel analitik, view teragregasi, maupun skema dimensional sederhana. Literatur lakehouse menekankan bahwa serving layer berperan penting dalam menjaga konsistensi hasil analitik, karena seluruh dashboard dan laporan bisnis hanya diperbolehkan mengakses data melalui lapisan ini, bukan langsung dari zona penyimpanan sebelumnya. Pendekatan tersebut bertujuan untuk mengurangi risiko inkonsistensi interpretasi data serta meminimalkan duplikasi logika transformasi pada sisi visualisasi.

2.9.2 Tujuan Perancangan Serving Layer pada Studi Kasus

Pada studi kasus yang dikembangkan dalam penelitian ini, perancangan serving layer diarahkan untuk mendukung penyajian hasil analitik terkait perilaku, pola, dan indikasi pengambilan keputusan berbasis data yang telah diproses pada curated zone. Serving layer tidak dirancang untuk melakukan pembersihan atau transformasi data lanjutan, melainkan untuk memastikan bahwa data yang disajikan telah berada pada tingkat kematangan analitik yang stabil dan dapat dikonsumsi secara langsung oleh sistem visualisasi.

Tujuan utama dari perancangan ini adalah memastikan bahwa setiap metrik, indikator, dan agregasi yang ditampilkan pada dashboard memiliki sumber data yang jelas, terstandarisasi, serta dapat ditelusuri kembali ke zona sebelumnya apabila diperlukan proses validasi. Dengan demikian, serving layer berfungsi sebagai lapisan kontrol analitik yang menjaga agar interpretasi data tetap konsisten meskipun visualisasi yang dihasilkan bersifat interaktif dan dinamis.

2.9.3 Arsitektur dan Komponen Serving Layer

rekомендации_mandi_preskriptif		riwayat_aktivitas_dashboard	
waktu_mandi_terakhir	timestamp	timestamp	timestamp
jam_sejak_mandi	double	id_aktivitas	text
skor_kekotoran	double	nama_aktivitas	text
skor_bau	bigint	durasi_menit	bigint
skor_aqi	double	skor_met	double
skor_final	double	nama_kategori	text

Gambar 2.8 Schema yang akan digunakan oleh Dashboard

Serving layer pada sistem ini dirancang menggunakan basis data relasional analitik yang berperan sebagai media penyajian data hasil kurasi. Data dari curated zone dimuat ke dalam serving layer melalui proses extract dan load tanpa transformasi logika tambahan, sehingga struktur dan makna data tetap konsisten dengan hasil pemodelan sebelumnya. Pendekatan ini sejalan dengan prinsip lakehouse modern yang memisahkan tanggung jawab antara pemrosesan data dan penyajian analitik.

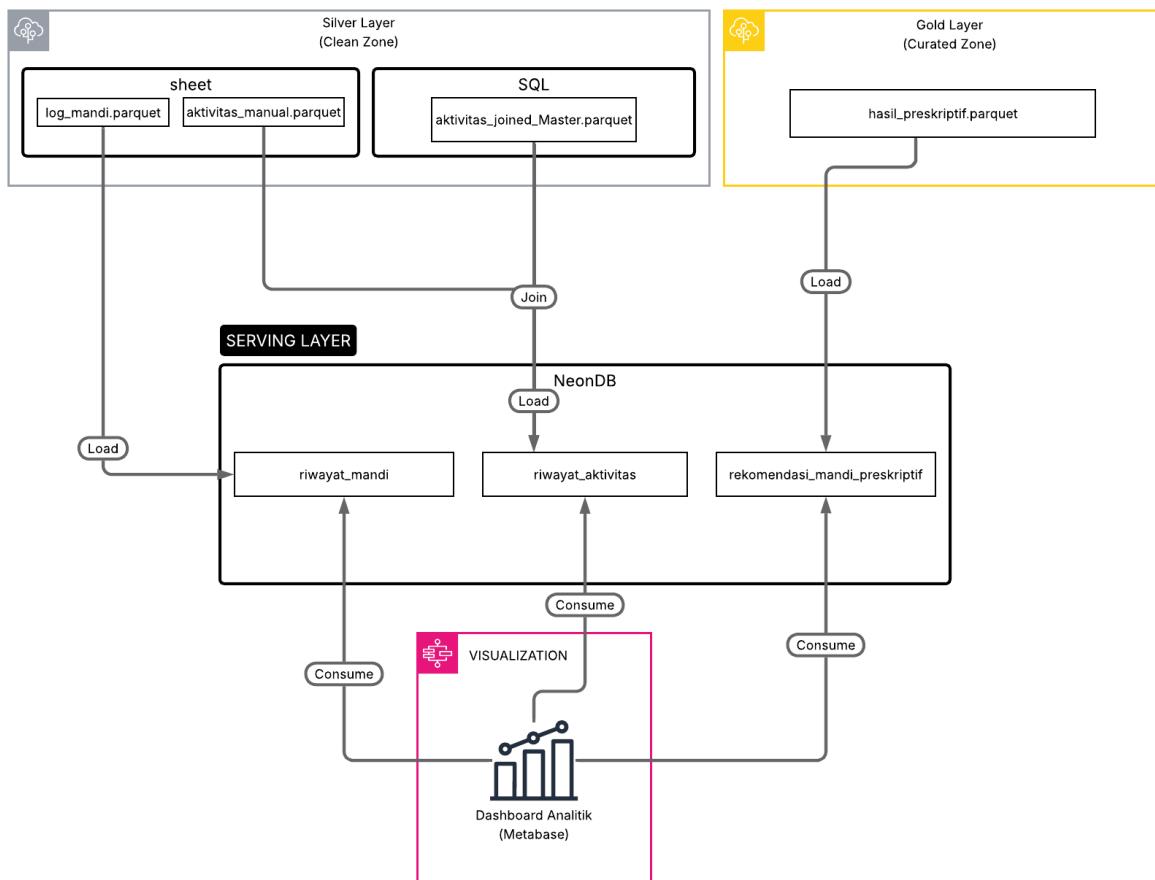
Di dalam serving layer, data disusun dalam bentuk tabel atau view yang merepresentasikan kebutuhan analitik utama, seperti ringkasan aktivitas, distribusi variabel, serta metrik agregat yang relevan dengan tujuan analisis. Penyusunan ini bertujuan untuk mengoptimalkan performa kueri sekaligus menyederhanakan proses eksplorasi data pada sisi dashboard. Dengan struktur tersebut, sistem visualisasi dapat melakukan kueri secara langsung tanpa perlu mengeksekusi logika transformasi kompleks yang berpotensi memperlambat proses analitik.

2.9.4 Tujuan Perancangan Serving Layer pada Studi Kasus

Serving layer memiliki keterkaitan langsung dengan sistem visualisasi yang digunakan dalam penelitian ini, di mana seluruh dashboard hanya diizinkan mengakses data melalui lapisan ini. Pembatasan tersebut bukan sekadar keputusan teknis, melainkan bagian dari strategi tata kelola data untuk menjaga keandalan insight yang dihasilkan. Dengan mengisolasi dashboard dari zona raw dan clean, risiko kesalahan interpretasi akibat data belum tervalidasi dapat diminimalkan secara signifikan.

Selain itu, serving layer memungkinkan penerapan praktik analitik yang lebih terkontrol, seperti penggunaan metrik yang konsisten antar visualisasi dan penghindaran perhitungan ulang yang tidak terstandarisasi. Hal ini menjadi penting ketika dashboard digunakan sebagai alat pendukung evaluasi atau pembuktian temuan analitik, karena seluruh hasil visual yang ditampilkan dapat ditelusuri ke struktur data yang sama.

2.9.5 Posisi Serving Layer dalam Alur Keseluruhan Sistem



Gambar 2.9 Posisi serving layer dalam Arsitektur Data Lakehouse

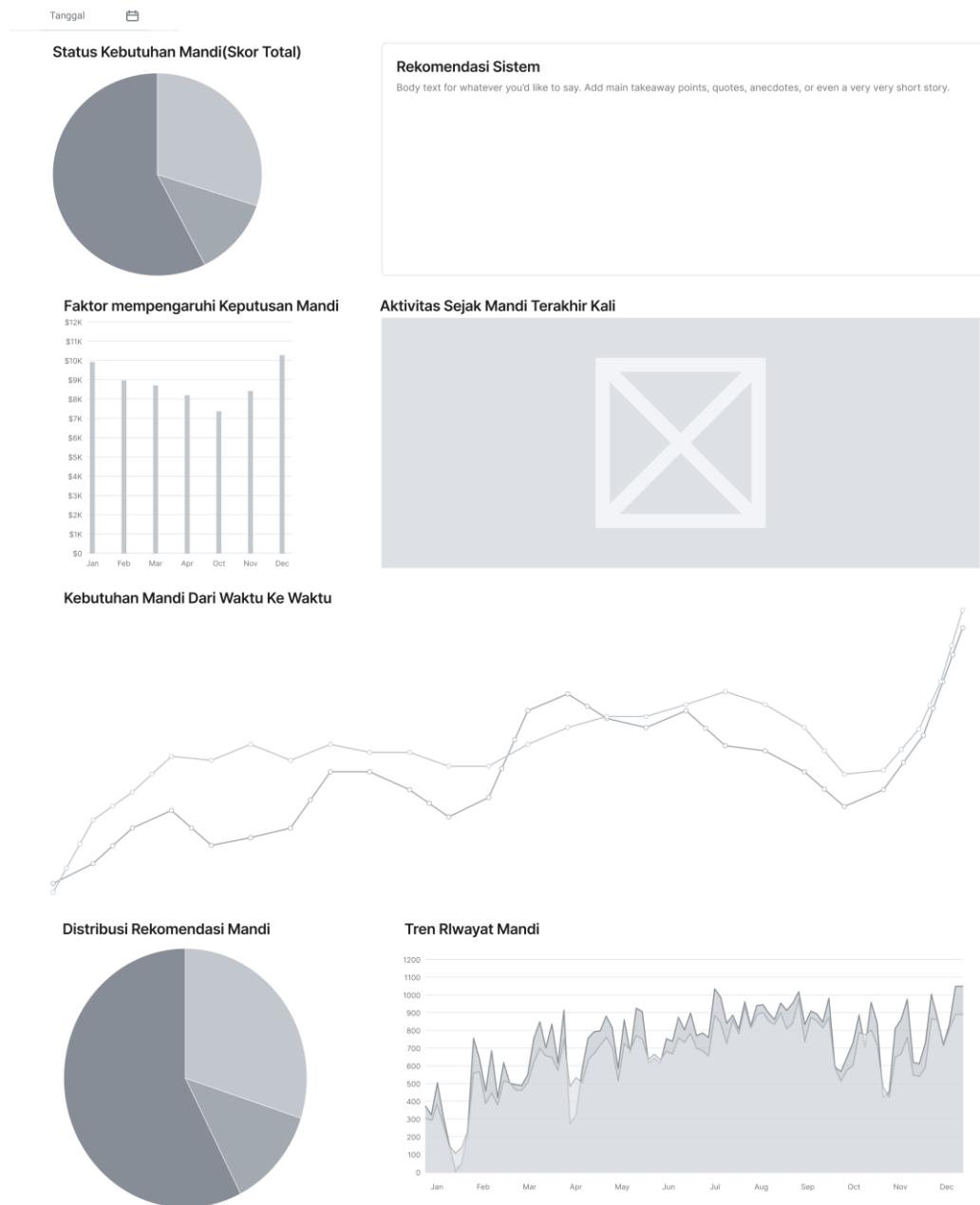
Dalam keseluruhan alur sistem, serving layer menempati posisi akhir sebelum data dikonsumsi oleh pengguna akhir. Lapisan ini menerima data yang telah melalui proses ekstraksi, pembersihan, normalisasi, dan kurasi, kemudian menyajikannya dalam bentuk yang siap dianalisis secara visual. Dengan demikian, serving layer tidak hanya berfungsi sebagai lapisan teknis, tetapi juga sebagai representasi formal dari hasil pengolahan data yang menjadi dasar penarikan kesimpulan.

Keberadaan serving layer memastikan bahwa sistem data lakehouse yang dirancang tidak berhenti pada aspek penyimpanan dan pemrosesan semata, melainkan benar-benar mendukung proses pengambilan keputusan berbasis data secara terstruktur dan dapat dipertanggungjawabkan.

2.10 Perancangan Front End (Visualisasi dan OLAP)

Front end sistem dirancang dalam bentuk dashboard analitik untuk menyajikan hasil perhitungan skor dan rekomendasi keputusan mandi. Dashboard dipilih karena mampu menyajikan informasi secara ringkas, terstruktur, dan mudah dipahami oleh pengguna. Visualisasi data berperan penting dalam meningkatkan pemahaman pengguna terhadap hasil analitik dan dasar pengambilan keputusan (Nguyen, 2025).

Front End dirancang untuk menyajikan hasil analitik preskriptif yang telah dimuat ke serving layer dalam bentuk visual yang mudah dipahami. Seluruh visualisasi bersifat read-only dan tidak melakukan transformasi analitik tambahan, sehingga konsistensi data dengan hasil pipeline ELT dan ETL tetap terjaga.



Gambar 2.10 Rancangan Front End

a. Status Kebutuhan Mandi (Skor Total)

Visual ini menampilkan ringkasan kondisi kebutuhan mandi pengguna berdasarkan skor total yang dihasilkan sistem. Diagram lingkaran digunakan untuk menunjukkan proporsi tingkat kebutuhan mandi dalam suatu periode, sehingga pengguna dapat memahami kondisi kebersihan secara cepat tanpa membaca detail numerik.

b. Rekomendasi Sistem

Komponen ini menyajikan hasil keputusan preskriptif dalam bentuk teks. Rekomendasi diturunkan langsung dari skor kebersihan dan aturan keputusan yang telah diproses pada curated zone, sehingga berfungsi sebagai interpretasi akhir yang mudah dipahami oleh pengguna.

c. Faktor yang Mempengaruhi Keputusan Mandi

Visual ini menampilkan kontribusi relatif faktor-faktor utama, seperti aktivitas, kondisi cuaca, kualitas udara, dan waktu sejak mandi terakhir. Grafik batang digunakan untuk meningkatkan transparansi sistem dan membantu pengguna memahami dasar pengambilan keputusan.

d. Aktivitas Sejak Mandi Terakhir

Visual ini menyajikan ringkasan aktivitas yang dilakukan pengguna setelah mandi terakhir. Tujuannya adalah memberikan konteks perilaku yang menjelaskan perubahan skor kebersihan dan rekomendasi yang dihasilkan oleh sistem.

e. Kebutuhan Mandi dari Waktu ke Waktu

Grafik deret waktu digunakan untuk menampilkan perubahan skor kebutuhan mandi secara temporal. Visual ini membantu pengguna mengamati pola dan tren kebutuhan mandi dalam rentang waktu tertentu.

f. Distribusi Rekomendasi Mandi

Visual distribusi digunakan untuk menunjukkan proporsi rekomendasi mandi dan tidak mandi yang dihasilkan sistem. Komponen ini berfungsi sebagai indikator evaluatif terhadap kecenderungan keputusan sistem.

g. Tren Riwayat Mandi

Visual ini menampilkan pola kebiasaan mandi pengguna dalam jangka waktu yang lebih panjang. Data diambil dari histori rekomendasi yang disimpan secara append pada serving layer, sehingga dapat menggambarkan perubahan perilaku pengguna dari waktu ke waktu.

2.11 Perancangan Front End Analisis Preskriptif

Analisis preskriptif pada sistem ini dirancang untuk menghasilkan rekomendasi tindakan berdasarkan integrasi berbagai faktor data. Front end preskriptif menampilkan hasil akhir berupa rekomendasi seperti mandi segera, mandi dapat ditunda, atau tidak perlu mandi, yang dihitung berdasarkan bobot skor mets pengguna dan kondisi lingkungan.

Integrasi langsung antara data terkuras dan logika preskriptif memungkinkan sistem memberikan rekomendasi yang konsisten dan dapat dijelaskan. Literatur menegaskan bahwa keterpaduan antara arsitektur data dan analitik preskriptif merupakan fondasi utama dalam implementasi sistem pendukung keputusan berbasis data yang efektif (Nguyen, 2025; Harby, 2025).



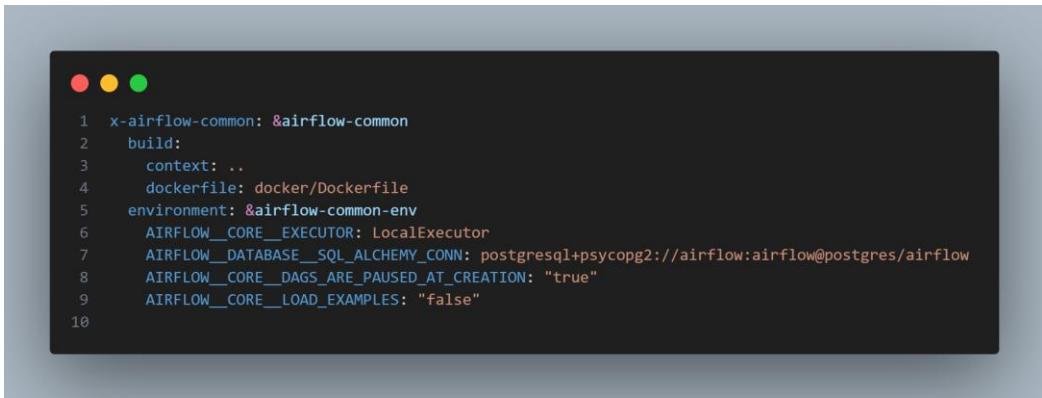
Gambar 2.11 Rancangan Front End Preskriptif

BAB III

3.1. Development Environment

Pengembangan sistem *Data Lakehouse* untuk pemantauan kebersihan diri ini dibangun di atas infrastruktur terisolasi berbasis kontainerisasi menggunakan Docker. Pendekatan ini dipilih untuk menjamin reprodukabilitas (*reproducibility*) lingkungan pengembangan dan kemudahan proses *deployment* antar-platform. Konfigurasi seluruh layanan didefinisikan secara deklaratif dalam berkas docker-compose.yml, yang mengorkestrasikan empat layanan utama

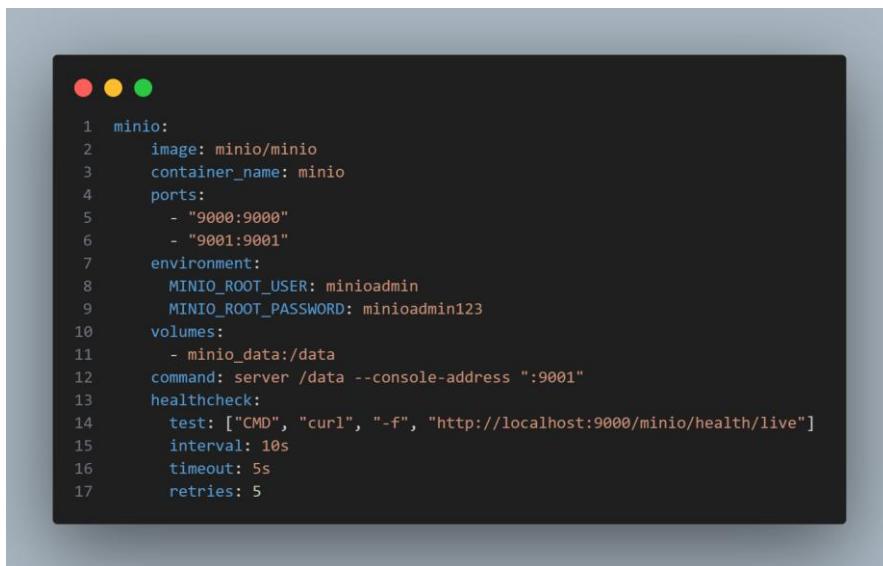
1. **Apache Airflow (Versi 2.7.1):** Berperan sebagai *workflow orchestrator* pusat yang bertugas menjadwalkan, memantau, dan mengeksekusi *pipeline* ELT.



```
1 x-airflow-common: &airflow-common
2   build:
3     context: ..
4     dockerfile: docker/Dockerfile
5     environment: &airflow-common-env
6       AIRFLOW__CORE__EXECUTOR: LocalExecutor
7       AIRFLOW__DATABASE__SQLALCHEMY_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow
8       AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: "true"
9       AIRFLOW__CORE__LOAD_EXAMPLES: "false"
10
```

Gambar 3.12 Konfigurasi Environment Airflow

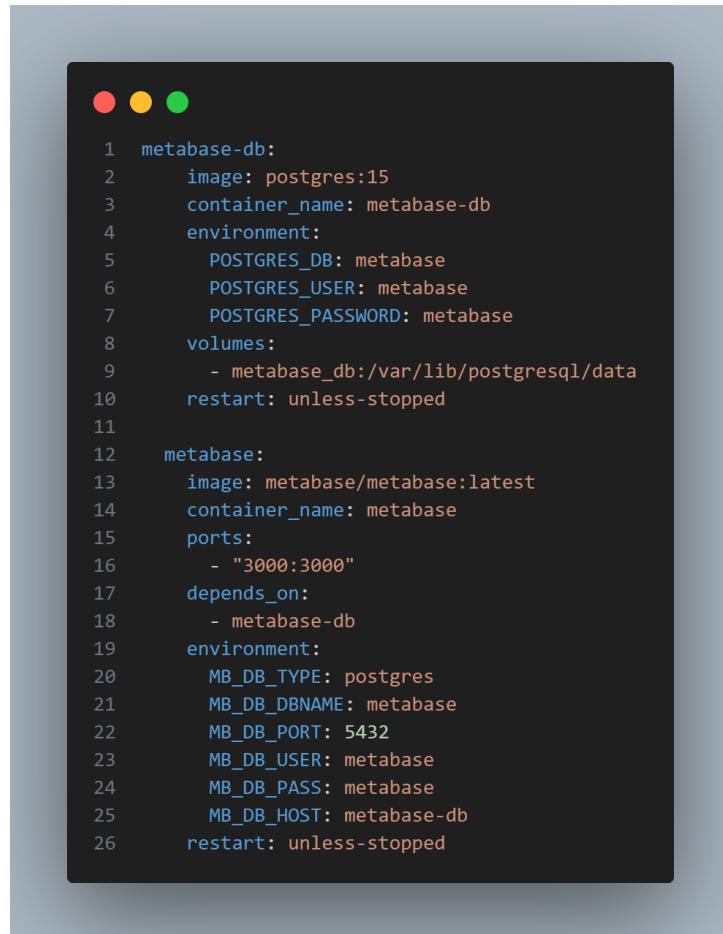
2. **MinIO:** Layanan Object Storage berkinerja tinggi yang kompatibel dengan protokol AWS S3, berfungsi sebagai lapisan penyimpanan utama bagi Data Lake. Layanan ini dikonfigurasi pada port 9000 untuk API dan 9001 untuk konsol manajemen.



```
1 minio:
2   image: minio/minio
3   container_name: minio
4   ports:
5     - "9000:9000"
6     - "9001:9001"
7   environment:
8     MINIO_ROOT_USER: minioadmin
9     MINIO_ROOT_PASSWORD: minioadmin123
10  volumes:
11    - minio_data:/data
12  command: server /data --console-address ":9001"
13  healthcheck:
14    test: ["CMD", "curl", "-f", "http://localhost:9000/minio/health/live"]
15    interval: 10s
16    timeout: 5s
17    retries: 5
```

Gambar 3.13 Konfigurasi Layanan MinIO

3. **PostgreSQL (Neon DB)**: Digunakan sebagai serving layer atau data warehouse untuk menyimpan hasil analisis akhir, sekaligus sebagai backend metadata untuk Apache Airflow.
4. **Metabase**: Alat *Business Intelligence* (BI) yang digunakan untuk membangun antarmuka dasbor visual bagi pengguna akhir.



```
1  metabase-db:
2    image: postgres:15
3    container_name: metabase-db
4    environment:
5      POSTGRES_DB: metabase
6      POSTGRES_USER: metabase
7      POSTGRES_PASSWORD: metabase
8    volumes:
9      - metabase_db:/var/lib/postgresql/data
10   restart: unless-stopped
11
12 metabase:
13   image: metabase/metabase:latest
14   container_name: metabase
15   ports:
16     - "3000:3000"
17   depends_on:
18     - metabase-db
19   environment:
20     MB_DB_TYPE: postgres
21     MB_DB_DBNAME: metabase
22     MB_DB_PORT: 5432
23     MB_DB_USER: metabase
24     MB_DB_PASS: metabase
25     MB_DB_HOST: metabase-db
26   restart: unless-stopped
```

Gambar 3.14 Konfigurasi Metabase dan PostgreSQL

Seluruh logika pemrosesan data diimplementasikan menggunakan bahasa pemrograman Python 3.9, dengan memanfaatkan pustaka khusus seperti pandas untuk manipulasi data tabular, boto3 untuk interaksi dengan layanan *cloud storage*, dan deltalake untuk manajemen format penyimpanan modern yang mendukung transaksi ACID (*Atomicity, Consistency, Isolation, Durability*).

	Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Disk read/v	Actions
□	docker	-	-	-	4.44%	1.93GB / 38.05Gi	25.34%	934KB / 86	⋮ ⚡
□	postgres-1	749da7a095e2	postgres:13	-	0.63%	33.97MB / 7.61Gi	0.44%	0B / 45.4Mi	⋮ ⚡
□	createbuckets	35806152267a	minio/mc	-	0%	0B / 0B	0%	0B / 0B	▷ ⋮ ⚡
□	minio	4ff070c13c5c	minio/minio	9000:9000 ↗ Show all ports (2)	0.05%	95.75MB / 7.61Gi	1.23%	0B / 19.8Mi	⋮ ⚡
□	database	7814c2711d11	database:13	3000:3000 ↗	1.78%	862.3MB / 7.61Gi	11.07%	0B / 17.8Mi	⋮ ⚡
□	airflow-init-1	7e89b0e8159f	docker-airflow	-	0%	0B / 0B	0%	0B / 0B	▷ ⋮ ⚡
□	airflow-sched	015c567f22be	docker-airflow	-	1.86%	441.2MB / 7.61Gi	5.66%	934KB / 3.2	⋮ ⚡
□	airflow-webserver	d409b49fedef	docker-airflow	8080:8080 ↗	0.12%	540.7MB / 7.61Gi	6.94%	0B / 8.19Ki	⋮ ⚡

Gambar 3.15 Environment Docker

Tampilan Docker Desktop atau terminal yang menunjukkan status container (*airflow-scheduler*, *airflow-webserver*, *minio*, *postgres*, *metabase*) dalam keadaan 'Running'.

3.2. Implementasi Data Storage

Arsitektur penyimpanan memisahkan komputasi dan penyimpanan dengan membagi Data Lake ke dalam beberapa zona logika.

3.2.1. Konfigurasi Arsitektur Medallion (MinIO)

Penyimpanan data pada MinIO mengadopsi pola Arsitektur Medallion untuk menjamin kualitas dan integritas data di setiap tahap pemrosesan. Inisialisasi lingkungan penyimpanan dilakukan secara otomatis melalui layanan *createbuckets*, yang memanfaatkan *MinIO Client* (mc) untuk membuat struktur *bucket* yang diperlukan segera setelah layanan utama aktif.

```

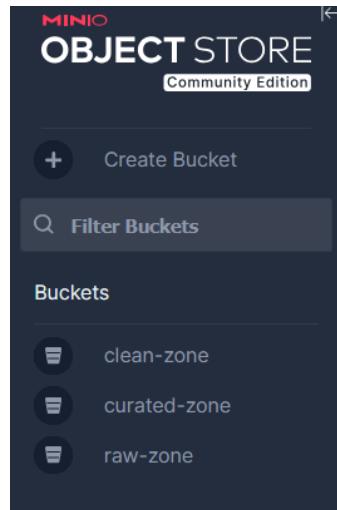
1  createbuckets:
2    image: minio/mc
3    depends_on:
4      minio:
5        condition: service_healthy
6    entrypoint: >
7      /bin/sh -c "
8      /usr/bin/mc alias set myminio http://minio:9000 minioadmin minioadmin123;
9      /usr/bin/mc mb --ignore-existing myminio/raw-zone;
10     /usr/bin/mc mb --ignore-existing myminio/clean-zone;
11     /usr/bin/mc mb --ignore-existing myminio/curated-zone;
12     exit 0;
13   "

```

Gambar 3.16 Otomasi Pembuatan Bucket MinIO

1. **Raw Zone (raw-zone):** Menyimpan data mentah asli dari sumber eksternal tanpa modifikasi.
2. **Clean Zone (clean-zone):** Menyimpan data yang telah dibersihkan, divalidasi, dan dikonversi ke format Delta Lake.

3. **Curated Zone (curated-zone)**: Menyimpan hasil agregasi dan keluaran logika preskriptif dalam format Delta Lake serta log kalkulasi dalam format CSV.



Gambar 3.17 Bucket MinIO

3.3. Implementasi Pemrosesan Data (ELT Pipeline)

Implementasi pemrosesan data dilakukan secara bertahap mengikuti aliran data dari hulu ke hilir, memastikan data mengalami peningkatan nilai guna di setiap lapisannya.

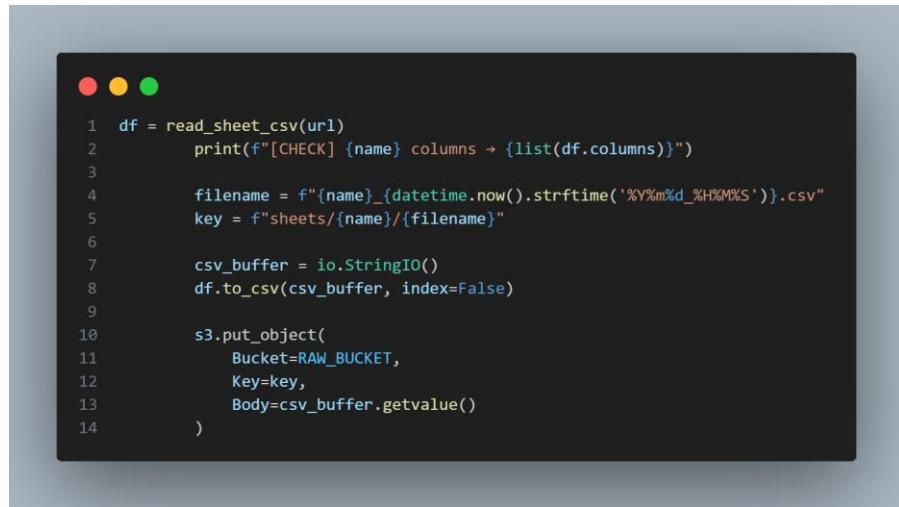
3.3.1. Raw Layer: Akuisisi Data Heterogen

Implementasi pada *Raw Layer* difokuskan pada proses akuisisi data (*ingestion*) dari empat sumber data yang heterogen. Proses ini dijalankan oleh serangkaian skrip Python yang bertugas mengekstraksi data dan menyimpannya ke dalam raw-zone tanpa mengubah konten aslinya untuk keperluan audit.

A screenshot of a terminal window showing a snippet of Python code. The code defines a dictionary named 'URLS' containing two items: 'aktivitas_manual' and 'log_mandi'. Both items have their values set to URLs from Google Sheets, using f-strings. The code is as follows:

```
1 URLs = {  
2     "aktivitas_manual": f"https://docs.google.com/spreadsheets/d/{SHEET_ID}/gviz/tq?tqx=out:csv&sheet=aktivitas_manual",  
3     "log_mandi": f"https://docs.google.com/spreadsheets/d/{SHEET_ID}/gviz/tq?tqx=out:csv&sheet=log_mandi",  
4 }
```

Gambar 3.18 Sumber Data Sheet



```
1 df = read_sheet_csv(url)
2     print(f"[CHECK] {name} columns → {list(df.columns)}")
3
4 filename = f"{name}_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
5 key = f"sheets/{name}/{filename}"
6
7 csv_buffer = io.StringIO()
8 df.to_csv(csv_buffer, index=False)
9
10 s3.put_object(
11     Bucket=RAW_BUCKET,
12     Key=key,
13     Body=csv_buffer.getvalue()
14 )
```

Gambar 3.19 Fungsi Membaca Sheet

Pada proses akuisisi data dari Google Sheets, sistem menggunakan skrip Python untuk menarik data aktivitas manual dan preferensi pengguna. Secara teknis, skrip memanggil URL spreadsheet yang telah diformat untuk ekspor CSV, kemudian data tersebut dibaca ke dalam dataframe Pandas. Alih-alih menyimpan file secara fisik di penyimpanan lokal yang bersifat sementara, data dialirkan melalui io.StringIO ke dalam memori, lalu diunggah langsung ke bucket MinIO menggunakan perintah s3.put_object. Metode ini meningkatkan efisiensi proses karena meminimalkan operasi input/output pada disk.



```
1 resp = requests.get(AQICN_URL)
2 data = resp.json()
3
4 key = f"api/aqicn/aqicn_{datetime.now().strftime('%Y%m%d_%H%M%S')}.json"
5
6 s3.put_object(
7     Bucket=RAW_BUCKET,
8     Key=key,
9     Body=json.dumps(data, indent=2)
10 )
11
```

Gambar 3.20 Ingest Api

Data yang berasal dari Web API, seperti informasi cuaca dari BMKG dan kualitas udara dari AQICN, sistem mempertahankan format JSON. Langkah ini sangat krusial karena data API seringkali memiliki struktur hierarkis atau bersarang (nested) yang kompleks. Dengan menyimpan format asli JSON ke dalam jalur folder api/, integritas struktur data tetap terjaga sebelum nantinya diparsing menjadi tabel relasional di tahap Bronze atau Silver. Hal ini memungkinkan sistem untuk tetap fleksibel jika di masa depan terdapat perubahan skema dari penyedia API.

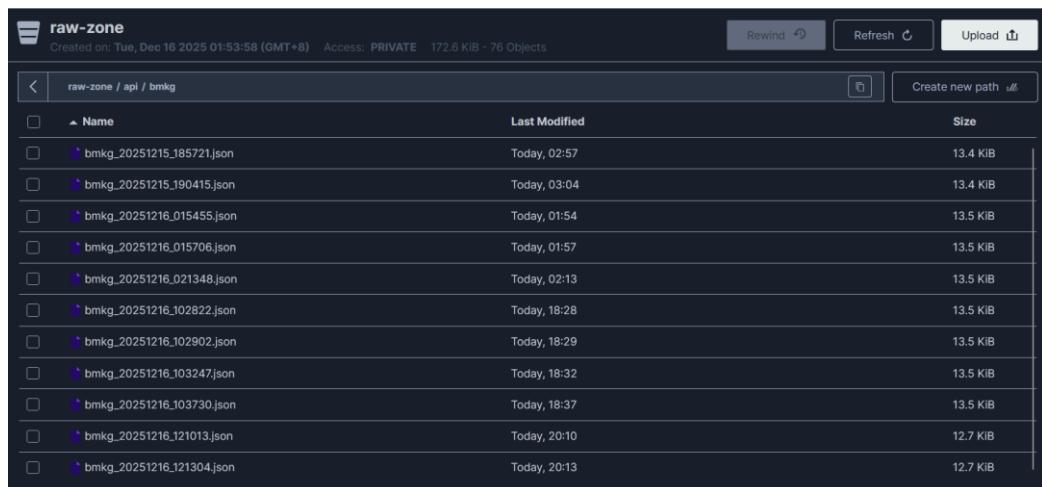
```

1  query = f'SELECT * FROM "Aktivitas"."{name}"'
2      df = pd.read_sql(query, engine)
3
4      if df.empty:
5          print(f"⚠️ Tabel {name} kosong, melewati proses upload.")
6          continue
7
8      csv_buffer = io.BytesIO()
9      df.to_csv(csv_buffer, index=False, encoding='utf-8')
10     csv_buffer.seek(0)
11
12     timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
13     object_key = f'sql/{name}/{name}_{timestamp}.csv'
14
15     s3.put_object(
16         Bucket="raw-zone",
17         Key=object_key,
18         Body=csv_buffer
19     )
20

```

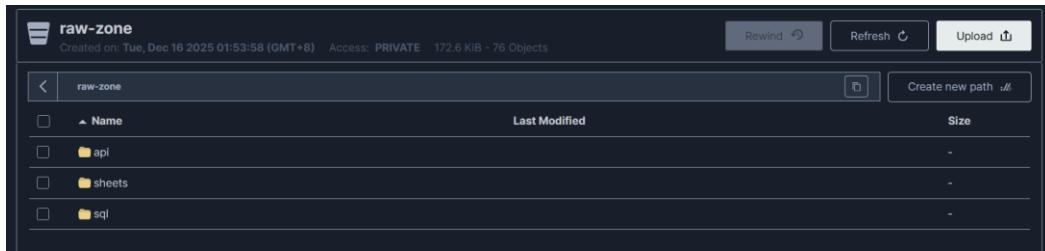
Gambar 3.21 Ingest SQL

Data dari Database Operasional, ekstraksi dilakukan dengan kueri SQL melalui pustaka SQLAlchemy. Data yang berhasil ditarik kemudian dikonversi menjadi format CSV di dalam memori menggunakan io.BytesIO. Penggunaan format CSV untuk data SQL ditujukan untuk standarisasi data tabular, sehingga memudahkan proses integrasi dengan data yang berasal dari Google Sheets. Semua hasil ekstraksi ini diarahkan ke jalur folder sql/ di dalam bucket raw-zone.



Gambar 3.22 Bucket raw-zone MinIO

Seluruh objek yang disimpan dalam Raw Layer mengikuti skema penamaan file yang ketat dengan menyertakan penanda waktu hingga satuan detik, seperti pada contoh berkas bmkg_20251216_121304.json. Strategi penamaan ini memastikan terpenuhinya prinsip idempotensi, di mana eksekusi skrip yang dilakukan berulang kali tidak akan menimpa data yang sudah ada, melainkan menciptakan versi baru. Hal ini memberikan kemampuan bagi sistem untuk memiliki rekam jejak sejarah data yang lengkap dan memudahkan proses pemulihan data jika diperlukan.



Gambar 3.23 Bucket raw-zone MinIO

Secara keseluruhan, struktur folder yang terorganisir di dalam MinIO mulai dari kategori api, sheets, hingga sql memungkinkan otomatisasi pada tahap ELT berikutnya untuk mengenali dan mengambil data secara spesifik berdasarkan sumber dan waktu pengambilannya.

3.3.2. Silver Layer: Transformasi dan Delta Lake

Implementasi pada Silver Layer berfokus pada pembersihan, normalisasi, dan pengintegrasian data yang sebelumnya telah dikumpulkan di Raw Layer. Pada tahap ini, data mengalami peningkatan nilai guna melalui proses standarisasi nama kolom, penyesuaian tipe data, dan deduplikasi. Aspek teknis terpenting pada lapisan ini adalah penggunaan format penyimpanan Delta Lake (berbasis Parquet) di dalam bucket clean-zone. Delta Lake memungkinkan sistem untuk menegakkan skema data (schema enforcement) dan mendukung transaksi ACID, yang menjamin struktur data tetap konsisten dan terhindar dari korupsi meskipun terjadi kegagalan sistem saat penulisan.

```

1 # 1. Identifikasi File Terbaru dari Raw Zone
2 path_aktivitas = get_latest_csv(RAW_BUCKET, "sheets/aktivitas_manual/")
3 path_mandi = get_latest_csv(RAW_BUCKET, "sheets/log_mandi/")
4
5 print(f"🕒 File ditemukan: \n - {path_aktivitas} \n - {path_mandi}")
6
7 # 2. Load Data & Normalisasi Nama Kolom
8 df_aktivitas = normalize_columns(read_csv_from_s3(RAW_BUCKET, path_aktivitas))
9 df_mandi = normalize_columns(read_csv_from_s3(RAW_BUCKET, path_mandi))
10
11 # --- PROSES CLEANING: Aktivitas Manual ---
12 print("⚡ Cleaning: aktivitas_manual...")
13 if "timestamp" in df_aktivitas.columns:
14     df_aktivitas["timestamp"] = pd.to_datetime(df_aktivitas["timestamp"], errors="coerce")
15
16 if "durasi_menit" in df_aktivitas.columns:
17     df_aktivitas["durasi_menit"] = pd.to_numeric(df_aktivitas["durasi_menit"], errors="coerce")
18
19 # Hapus baris yang tidak memiliki timestamp valid agar data Silver bersih
20 df_aktivitas = df_aktivitas.dropna(subset=["timestamp"])
21
22 # --- PROSES CLEANING: Log Mandi ---
23 print("⚡ Cleaning: log_mandi...")
24 # Menangani kolom 'waktu_mandi' dari source dan meragamkannya menjadi 'timestamp'
25 if "waktu_mandi" in df_mandi.columns:
26     df_mandi["timestamp"] = pd.to_datetime(df_mandi["waktu_mandi"], errors="coerce")
27
28 for col in ["tingkat_kekotoran", "tingkat_bau_badan"]:
29     if col in df_mandi.columns:
30         df_mandi[col] = pd.to_numeric(df_mandi[col], errors="coerce")
31
32 df_mandi = df_mandi.dropna(subset=["timestamp"])
33
34 # 3. Simpan ke Clean Zone (Delta Lake)
35 save_to_delta(df_aktivitas, CLEAN_BUCKET, "sheets/catatan_aktivitas")
36 save_to_delta(df_mandi, CLEAN_BUCKET, "sheets/log_mandi")
37

```

Gambar 3.24 Cleaning Sheet

Proses transformasi pertama dilakukan pada data Google Sheets, di mana skrip secara otomatis mengidentifikasi file terbaru dari folder sheets/aktivitas_manual/ dan sheets/log_mandi/ di Raw Layer. Langkah pembersihan melibatkan normalisasi nama kolom menggunakan fungsi normalize_columns serta konversi tipe data yang ketat. Kolom waktu dikonversi menjadi format datetime dan kolom numerik seperti durasi_menit dipastikan bertipe data angka dengan parameter errors="coerce" untuk menangani data yang tidak valid. Baris yang tidak memiliki timestamp valid dihapus untuk memastikan hanya data berkualitas yang tersimpan di Silver Layer.



```
1 df = pd.DataFrame([{
2     "datetime": data.get("time", {}).get("s"),
3     "aqi": data.get("aqi"),
4     "pm25": data.get("iaqi", {}).get("pm25", {}).get("v"),
5     "pm10": data.get("iaqi", {}).get("pm10", {}).get("v"),
6     "dominant_pollutant": data.get("dominentpol")
7 })
8
9 df["datetime"] = pd.to_datetime(df["datetime"])
```

Gambar 3.25 Normalisasi Data

Pada data yang bersumber dari Web API, transformasi dilakukan dengan teknik flattening untuk mengubah struktur JSON yang bersarang menjadi DataFrame tabular. Skrip mengekstraksi metrik spesifik seperti nilai aqi, kadar polutan pm25 dan pm10, serta informasi waktu dari objek JSON mentah. Setelah ekstraksi, kolom waktu diseragamkan ke tipe datetime menggunakan pd.to_datetime untuk memudahkan sinkronisasi data lingkungan dengan aktivitas pengguna pada tahap analisis lanjut.

```

1 # 1. Ambil path file terbaru dari raw-zone
2     key_aktivitas = get_latest_csv(RAW_BUCKET, "aktivitas")
3     key_kategori = get_latest_csv(RAW_BUCKET, "kategori")
4
5     print(f"[RAW] Menggabungkan: {key_aktivitas} dan {key_kategori}")
6
7 # 2. Baca data dari Minio
8 obj_akt = s3.get_object(Bucket=RAW_BUCKET, Key=key_aktivitas)
9 obj_kat = s3.get_object(Bucket=RAW_BUCKET, Key=key_kategori)
10
11 df_aktivitas = pd.read_csv(obj_akt["Body"])
12 df_kategori = pd.read_csv(obj_kat["Body"])
13
14 # 3. Join Tabel berdasarkan id_kategori
15 # Ini akan menggabungkan nama_aktivitas dengan nama_kategori (Indoor/Outdoor)
16 df_clean = pd.merge(
17     df_aktivitas,
18     df_kategori,
19     on="id_kategori",
20     how="left"
21 )
22
23 # 4. Pembersihan Tipe Data (Ensuring MET is numeric)
24 df_clean["skor_met"] = pd.to_numeric(df_clean["skor_met"], errors="coerce")

```

Gambar 3.26 Cleaning SQL

Transformasi pada data SQL melibatkan operasi penggabungan (join) untuk memperkaya informasi. Data aktivitas mentah digabungkan dengan tabel master kategori menggunakan metode left join berdasarkan kunci id_kategori. Proses ini memungkinkan penambahan konteks pada setiap catatan aktivitas, seperti klasifikasi apakah kegiatan tersebut dilakukan di dalam ruangan (Indoor) atau luar ruangan (Outdoor). Selain itu, dilakukan pembersihan tipe data pada kolom skor_met untuk memastikan akurasi perhitungan beban fisik pengguna.

```

1 path = f"s3://{CLEAN_BUCKET}/api/aqi"
2     print(f"  Menyimpan Delta: {path}")
3
4     write_deltalake(
5         path,
6         df,
7         mode="overwrite",
8         storage_options=storage_options
9     )

```

Gambar 3.27 Menyimpan Ke Clean Bucket

```

1 path = f"s3://{CLEAN_BUCKET}/sql/aktivitas_joined_master"
2 print(f"💾 Menyimpan Tabel Gabungan ke Delta: {path}")
3
4 write_deltalake(
5     path,
6     df_clean,
7     mode="overwrite",
8     schema_mode="overwrite",
9     storage_options=storage_options
10 )

```

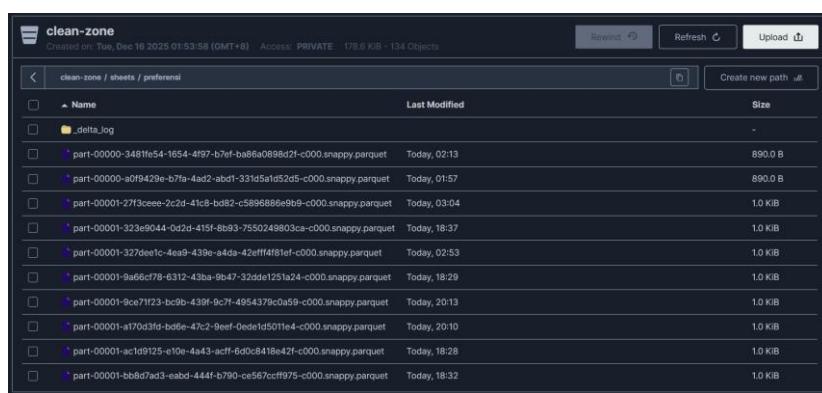
Gambar 3.28 Menyimpan Ke Clean Bucket

Tahap akhir dari Silver Layer adalah penyimpanan data yang telah dibersihkan ke dalam format Delta. Menggunakan fungsi `write_deltalake` dengan mode `overwrite`, sistem memperbarui dataset di clean-zone secara atomik. Penggunaan opsi `storage_options` memastikan koneksi ke MinIO tetap aman selama proses penulisan.



Gambar 3.29 Bucket clean-zone MinIO

Secara struktural, bucket clean-zone tetap terorganisir ke dalam sub-folder `api`, `sheets`, dan `sql` untuk menjaga ketertelusuran sumber data. Organisasi ini memudahkan komponen analitik lainnya untuk mengonsumsi data yang sudah bersih tanpa perlu melakukan pemrosesan ulang.

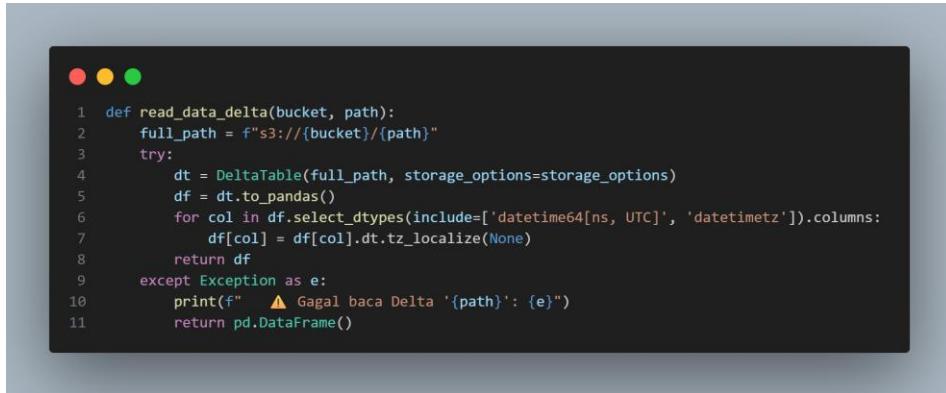


Gambar 3.30 Bucket clean-zone MinIO

Data disimpan dalam bentuk file Parquet yang terkompresi (.snappy.parquet) dan didampingi oleh folder metadata `_delta_log`. Folder log ini menyimpan catatan transaksi yang memungkinkan fitur time travel dan memastikan bahwa pembaca data selalu mendapatkan versi dataset terbaru yang konsisten.

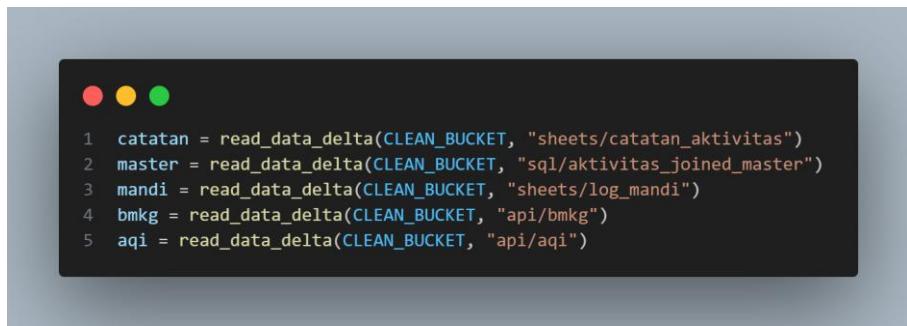
3.3.3. Gold Layer: Logika Analisis Preskriptif

Implementasi *Gold Layer* merupakan inti dari kecerdasan sistem ini, di mana logika analisis preskriptif diterapkan. Seluruh data bersih dari berbagai sumber (aktivitas, cuaca, kualitas udara) digabungkan untuk menghasilkan keputusan otomatis.



```
1 def read_data_delta(bucket, path):
2     full_path = f"s3://{bucket}/{path}"
3     try:
4         dt = DeltaTable(full_path, storage_options=storage_options)
5         df = dt.to_pandas()
6         for col in df.select_dtypes(include=['datetime64[ns, UTC]', 'datetimetz']).columns:
7             df[col] = df[col].dt.tz_localize(None)
8         return df
9     except Exception as e:
10        print(f"⚠️ Gagal baca Delta '{path}': {e}")
11        return pd.DataFrame()
```

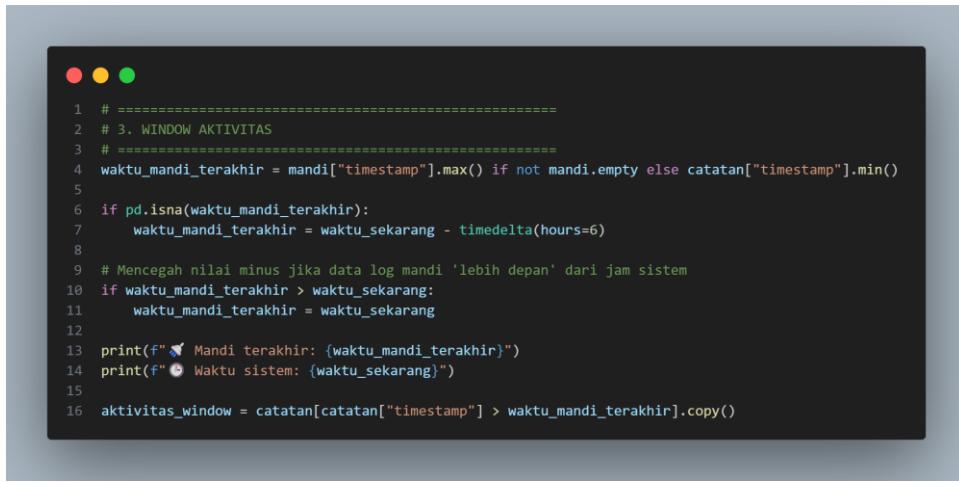
Gambar 3.31 Fungsi Membaca Parqueet



```
1 catatan = read_data_delta(CLEAN_BUCKET, "sheets/catatan_aktivitas")
2 master = read_data_delta(CLEAN_BUCKET, "sql/aktivitas_joined_master")
3 mandi = read_data_delta(CLEAN_BUCKET, "sheets/log_mandi")
4 bmkg = read_data_delta(CLEAN_BUCKET, "api/bmkg")
5 aqi = read_data_delta(CLEAN_BUCKET, "api/aqi")
```

Gambar 3.32 Fungsi Membaca Parqueet

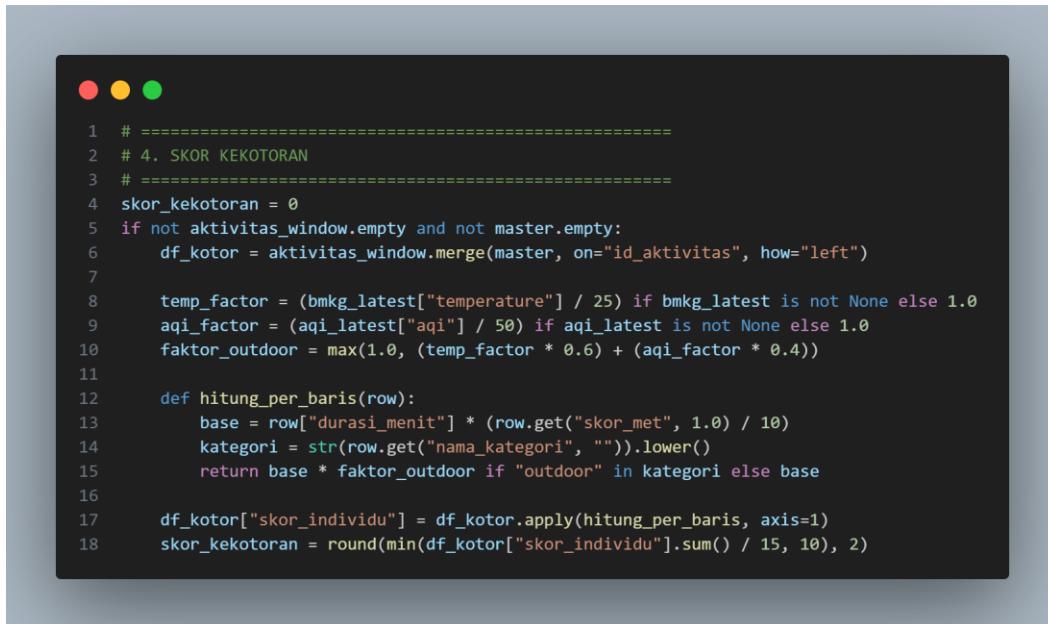
Proses dimulai dengan memuat kembali dataset dari clean-zone menggunakan fungsi `read_data_delta`. Fungsi ini dirancang khusus untuk membaca format Delta Lake dan mengonversinya menjadi DataFrame Pandas, sembari melakukan normalisasi zona waktu pada kolom datetime untuk memastikan konsistensi perhitungan. Data yang dimuat mencakup catatan aktivitas manual, data master kategori, log mandi, serta data cuaca (BMKG) dan kualitas udara (AQI).



```
1 # =====
2 # 3. WINDOW AKTIVITAS
3 # =====
4 waktu_mandi_terakhir = mandi["timestamp"].max() if not mandi.empty else catatan["timestamp"].min()
5
6 if pd.isna(waktu_mandi_terakhir):
7     waktu_mandi_terakhir = waktu_sekarang - timedelta(hours=6)
8
9 # Mencegah nilai minus jika data log mandi 'lebih depan' dari jam sistem
10 if waktu_mandi_terakhir > waktu_sekarang:
11     waktu_mandi_terakhir = waktu_sekarang
12
13 print(f"⌚ Mandi terakhir: {waktu_mandi_terakhir}")
14 print(f"🕒 Waktu sistem: {waktu_sekarang}")
15
16 aktivitas_window = catatan[catatan["timestamp"] > waktu_mandi_terakhir].copy()
```

Gambar 3.33 Logic Aktivitas Terakhir

Sistem kemudian menentukan cakupan waktu analisis melalui mekanisme Window Aktivitas. Logika ini mencari waktu mandi terakhir pengguna; jika data kosong, sistem secara otomatis mengambil jendela waktu 6 jam ke belakang sebagai basis observasi. Variabel aktivitas_window inilah yang menjadi dasar perhitungan beban kotoran dan bau badan yang terakumulasi sejak mandi terakhir hingga waktu saat ini.



```
1 # =====
2 # 4. SKOR KEKOTORAN
3 # =====
4 skor_kekotoran = 0
5 if not aktivitas_window.empty and not master.empty:
6     df_kotor = aktivitas_window.merge(master, on="id_aktivitas", how="left")
7
8     temp_factor = (bmkg_latest["temperature"] / 25) if bmkg_latest is not None else 1.0
9     aqi_factor = (aqi_latest["aqi"] / 50) if aqi_latest is not None else 1.0
10    faktor_outdoor = max(1.0, (temp_factor * 0.6) + (aqi_factor * 0.4))
11
12    def hitung_per_baris(row):
13        base = row["durasi_minit"] * (row.get("skor_met", 1.0) / 10)
14        kategori = str(row.get("nama_kategori", "")).lower()
15        return base * faktor_outdoor if "outdoor" in kategori else base
16
17    df_kotor["skor_individu"] = df_kotor.apply(hitung_per_baris, axis=1)
18    skor_kekotoran = round(min(df_kotor["skor_individu"].sum() / 15, 10), 2)
```

Gambar 3.34 Logic Skor Kotor

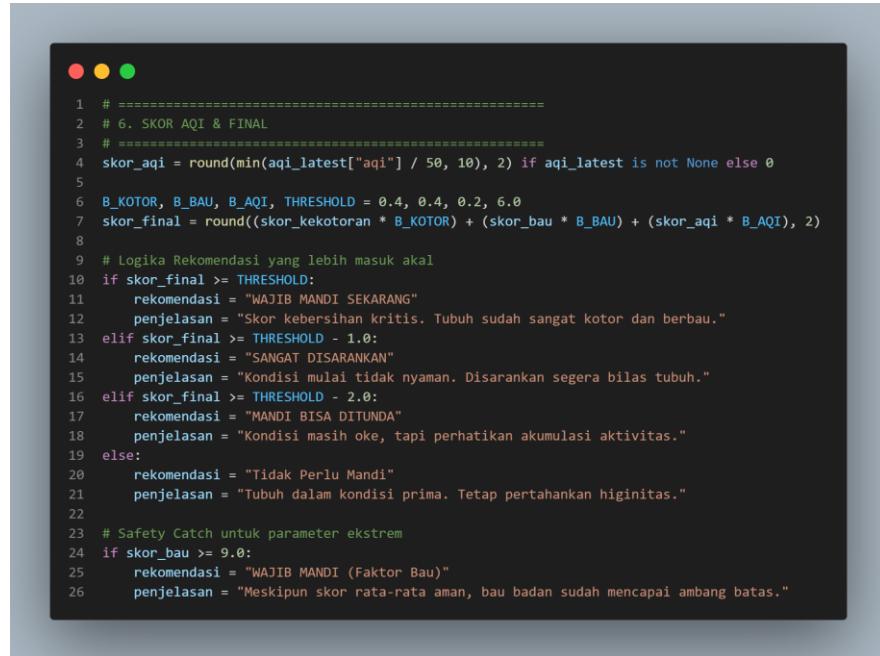
Logika perhitungan skor kekotoran dalam potongan kode ini diimplementasikan untuk menghasilkan nilai akumulasi kotoran yang dinamis dan berbasis data lingkungan. Proses dimulai dengan inisialisasi nilai awal sebesar nol dan pengecekan validitas data pada jendela aktivitas serta tabel master aktivitas sebelum dilakukan penggabungan data. Sistem kemudian menghitung faktor lingkungan secara otomatis dengan memproses data suhu terkini dari BMKG dan indeks kualitas udara atau AQI. Selanjutnya, setiap baris aktivitas diproses secara individual untuk menentukan skor dasarnya menggunakan perhitungan yang sudah dirancang



```
1 # =====
2 # 5. SKOR BAU BADAN (DISESUAIKAN TANPA BOBOT_BAU)
3 # =====
4
5 # 1. Hitung jam sejak mandi (Safety check agar tidak minus)
6 jam_sejak_mandi = (waktu_sekarang - waktu_mandi_terakhir).total_seconds() / 3600
7 if jam_sejak_mandi < 0: jam_sejak_mandi = 0
8
9 # 2. Filter aktivitas yang memicu bau
10 # Kita asumsikan aktivitas bau adalah yang Skor MET > 3 (Aktivitas sedang-berat)
11 # atau yang mengandung kata 'outdoor' di kategorinya.
12 aktivitas_analisis = aktivitas_window.merge(master, on="id_aktivitas", how="left")
13
14 def filter_bau(row):
15     # Kriteria 1: MET Tinggi (biasanya > 3.0 mulai berkeringat)
16     met_tinggi = row.get("skor_met", 0) > 3.0
17     # Kriteria 2: Lokasi Outdoor
18     is_outdoor = "outdoor" in str(row.get("nama_kategori", "")).lower()
19     return met_tinggi or is_outdoor
20
21 # Terapkan filter
22 aktivitas_bau = aktivitas_analisis[aktivitas_analisis.apply(filter_bau, axis=1)]
23 jumlah_aktivitas_bau = len(aktivitas_bau)
24
25 # 3. Faktor kelembapan (Max skor kontribusi = 2.0)
26 faktor_lembap = bmkg_latest["humidity"] / 100 if bmkg_latest is not None else 0.5
27
28 # 4. Rumus Final Skor Bau (Disesuaikan)
29 skor_bau = (
30     (jam_sejak_mandi * 0.3) +          # 10 jam = 3.0 poin
31     (jumlah_aktivitas_bau * 0.7) +      # 1 aktivitas berkeringat = 0.7 poin
32     (faktor_lembap * 2)                # Kelembapan lingkungan
33 )
34
35 # Batasi maksimal 10
36 skor_bau = round(min(skor_bau, 10), 2)
```

Gambar 3.35 Skor Bau Badan

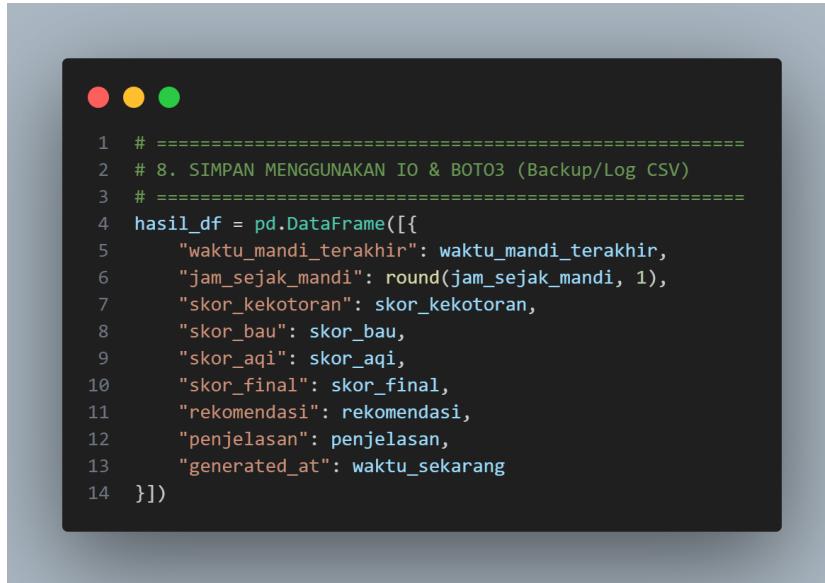
Untuk mengukur tingkat ketidaknyamanan aroma tubuh yang dipicu oleh aktivitas fisik, durasi waktu, dan kondisi atmosfer.



```
1 # =====
2 # 6. SKOR AQI & FINAL
3 # =====
4 skor_aqi = round(min(aqi_latest["aqi"] / 50, 10), 2) if aqi_latest is not None else 0
5
6 B_KOTOR, B_BAU, B_AQI, THRESHOLD = 0.4, 0.4, 0.2, 6.0
7 skor_final = round((skor_kekotoran * B_KOTOR) + (skor_bau * B_BAU) + (skor_aqi * B_AQI), 2)
8
9 # Logika Rekomendasi yang lebih masuk akal
10 if skor_final >= THRESHOLD:
11     rekomendasi = "WAJIB MANDI SEKARANG"
12     penjelasan = "Skor kebersihan kritis. Tubuh sudah sangat kotor dan berbau."
13 elif skor_final >= THRESHOLD - 1.0:
14     rekomendasi = "SANGAT DISARANKAN"
15     penjelasan = "Kondisi mulai tidak nyaman. Disarankan segera bilas tubuh."
16 elif skor_final >= THRESHOLD - 2.0:
17     rekomendasi = "MANDI BISA DITUNDA"
18     penjelasan = "Kondisi masih oke, tapi perhatikan akumulasi aktivitas."
19 else:
20     rekomendasi = "Tidak Perlu Mandi"
21     penjelasan = "Tubuh dalam kondisi prima. Tetap pertahankan higinitas."
22
23 # Safety Catch untuk parameter ekstrem
24 if skor_bau >= 9.0:
25     rekomendasi = "WAJIB MANDI (Faktor Bau)"
26     penjelasan = "Meskipun skor rata-rata aman, bau badan sudah mencapai ambang batas."
```

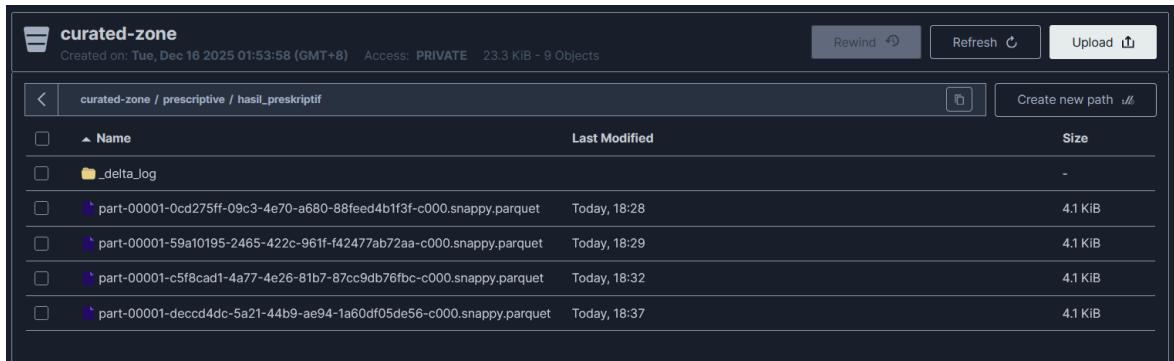
Gambar 3.36 Logic Skor Final

Sistem menggabungkan seluruh indikator menggunakan metode *Weighted Scoring* untuk menentukan keputusan akhir.



```
1 # =====
2 # 8. SIMPAN MENGGUNAKAN IO & BOTO3 (Backup/Log CSV)
3 # =====
4 hasil_df = pd.DataFrame([{
5     "waktu_mandi_terakhir": waktu_mandi_terakhir,
6     "jam_sejak_mandi": round(jam_sejak_mandi, 1),
7     "skor_kekotoran": skor_kekotoran,
8     "skor_bau": skor_bau,
9     "skor_aqi": skor_aqi,
10    "skor_final": skor_final,
11    "rekomendasi": rekomendasi,
12    "penjelasan": penjelasan,
13    "generated_at": waktu_sekarang
14 }])
```

Gambar 3.37 Menyimpan Hasil Logic



Gambar 3.38 Bucket curated-zone MinIO

Hasil kalkulasi, rekomendasi, dan penjelasan tersebut disusun ke dalam sebuah DataFrame final. Data ini kemudian disimpan secara permanen ke dalam curated-zone dengan format Delta Lake. Struktur folder di zona ini memungkinkan akses cepat bagi aplikasi frontend untuk menampilkan saran kesehatan kepada pengguna secara instan.

3.3.4. Serving Layer (Neon DB)

Tahap akhir dari arsitektur penyimpanan adalah *Serving Layer* yang menggunakan basis data PostgreSQL dari Neon DB sebagai tujuan akhir data setelah diproses di *Curated Zone*. Lapisan ini berfungsi sebagai media penyimpanan terstruktur untuk hasil analisis preskriptif yang telah divalidasi sehingga siap dikonsumsi secara efisien oleh alat visualisasi seperti Metabase.

A screenshot of a terminal window with a dark background. The window title bar has three colored circles (red, yellow, green). The terminal displays a Python script named 'Load_aktivitas_to_neon.py'. The code is color-coded: comments are in green, strings are in blue, and variables/keywords are in white. The script performs several steps: 1. Reads data from 'Clean Zone' (Google Sheets and SQL) into DataFrames. 2. Checks if the 'catatan' DataFrame is empty and exits if it is. 3. Converts the 'timestamp' column to datetime. 4. Merges the 'catatan' DataFrame with a 'master' DataFrame on the 'id_aktivitas' column using a left join. 5. Cleans the merged DataFrame by selecting specific columns ('timestamp', 'id_aktivitas', 'nama_aktivitas', 'durasi_menit', 'skor_met', 'nama_kategori'). 6. Prints a message indicating the number of rows to be loaded. 7. Establishes a connection to a Neon database using SQLAlchemy. 8. Writes the cleaned DataFrame to the 'riwayat_aktivitas_dashboard' table in the database, using 'replace' mode to ensure synchronization with the Lakehouse.

Gambar 3.39 Load_aktivitas_to_neon.py

Skrip Load_aktivitas_to_neon.py menangani pemindahan riwayat aktivitas lengkap untuk kebutuhan dashboard visualisasi. Skrip ini membaca data dari Clean Zone yang mencakup catatan aktivitas dan data master hasil join SQL. Melalui operasi merge di tingkat kode, data diperkaya dengan informasi kategori seperti klasifikasi lokasi indoor atau outdoor serta skor MET. Berbeda dengan skrip sebelumnya, proses pengunggahan ke tabel riwayat_aktivitas_dashboard menggunakan mode replace. Hal ini dilakukan agar tabel pada dashboard selalu sinkron dengan versi penuh riwayat yang ada di Lakehouse secara akurat dan terhindar dari duplikasi data aktivitas lama.

```

1 # Path sumber data di Curated Zone
2 path_curated = f"s3://{CURATED_BUCKET}/prescriptive_hygiene"
3
4 dt = DeltaTable(path_curated, storage_options=storage_options)
5 df_hasil = dt.to_pandas()
6
7 if not df_hasil.empty:
8     # Pastikan kolom waktu valid
9     if "generated_at" in df_hasil.columns:
10         df_hasil["generated_at"] = pd.to_datetime(df_hasil["generated_at"])
11
12     engine = create_engine(DATABASE_URL)
13
14     # Nama tabel baru: rekomendasi_mandi_preskriptif
15     # Menggunakan 'append' agar histori skor tersimpan terus ke bawah
16     df_hasil.to_sql(
17         "rekomendasi_mandi_preskriptif",
18         engine,
19         if_exists="append",
20         index=False
21     )
22     print("✅ Berhasil: Data dimuat ke tabel [rekomendasi_mandi_preskriptif].")
23 else:
24     print("⚠️ Data di Curated Zone kosong.")

```

Gambar 3.40 Load_prescriptive_to_sql.py

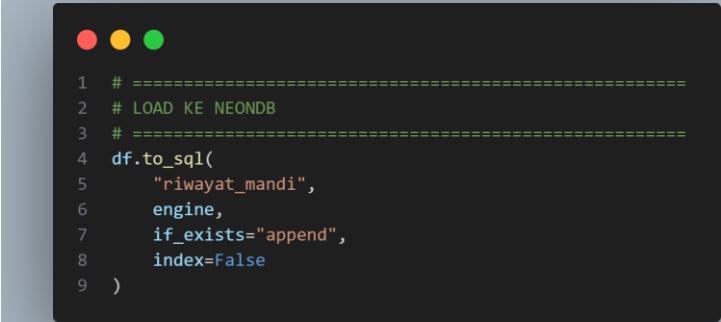
Load_prescriptive_to_sql.py bertugas menarik hasil keputusan otomatis dari jalur penyimpanan s3://curated-zone/prescriptive_hygiene. Dalam prosesnya, skrip ini memastikan kolom generated_at dikonversi menjadi format *datetime* yang valid sebelum dimuat ke tabel rekomendasi_mandi_preskriptif. Strategi penulisan yang digunakan adalah append, yang bertujuan agar seluruh rekam jejak histori skor dan rekomendasi tersimpan secara berkelanjutan tanpa menimpa data yang sudah ada, sehingga memungkinkan analisis tren jangka panjang bagi pengguna.

```

1 # =====
2 # AMBIL CSV TERBARU DARI RAW ZONE
3 # =====
4 def get_latest_csv(bucket, prefix):
5     response = s3.list_objects_v2(Bucket=bucket, Prefix=prefix)
6     if "Contents" not in response:
7         raise FileNotFoundError("Tidak ada file log_mandi di raw-zone")
8
9     csv_files = [o for o in response["Contents"] if o["Key"].endswith(".csv")]
10    if not csv_files:
11        raise FileNotFoundError("File CSV log_mandi tidak ditemukan")
12
13    latest = sorted(csv_files, key=lambda x: x["LastModified"], reverse=True)[0]
14    return latest["Key"]
15
16 key = get_latest_csv(RAW_BUCKET, PREFIX)
17 print(f"[RAW] log_mandi pakai → {key}")

```

Gambar 3.41 Load_riwayat_mandi_to_neon.py



```

1 # =====
2 # LOAD KE NEONDB
3 # =====
4 df.to_sql(
5     "riwayat_mandi",
6     engine,
7     if_exists="append",
8     index=False
9 )

```

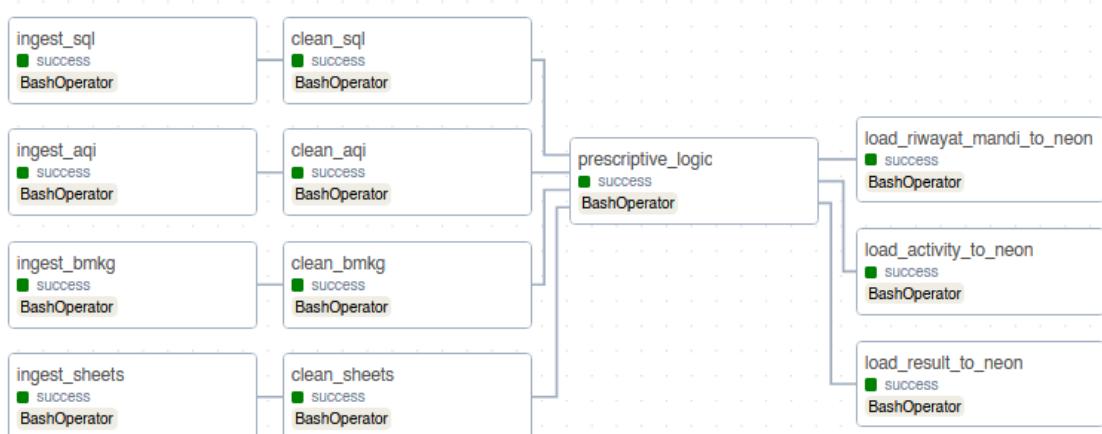
Gambar 3.42 loading ke NeonDB

Tahap terakhir diselesaikan oleh skrip Load_riwayat_mandi_to_neon.py yang mengelola integritas log kebersihan pengguna. Skrip ini secara otomatis mengidentifikasi file CSV terbaru di bucket raw-zone dengan menggunakan fungsi get_latest_csv. Setelah data dimuat ke DataFrame dan kolom waktunya divalidasi, data tersebut dipindahkan ke tabel riwayat_mandi menggunakan koneksi SQLAlchemy yang dikonfigurasi dengan parameter sslmode=require guna menjamin keamanan transmisi data ke layanan cloud. Secara keseluruhan, Serving Layer ini menyediakan fondasi data yang aman, terstruktur, dan performan, yang memungkinkan alat visualisasi seperti Metabase untuk menyajikan wawasan higienitas secara responsif berdasarkan data yang telah dikurasi.

3.4. Implementasi Orkestrasi Sistem

Apache Airflow diimplementasikan sebagai pusat kendali otomatisasi sistem. Seluruh skrip Python yang telah didefinisikan di atas disusun sebagai tugas-tugas (*tasks*) di dalam sebuah DAG (*Directed Acyclic Graph*) bernama hygiene_lakehouse_pipeline.

Dalam DAG tersebut, alur eksekusi diatur secara sekuensial dengan dependensi yang ketat: proses dimulai dari *Ingestion*, dilanjutkan dengan *Cleaning*, kemudian *Prescriptive Logic*, dan diakhiri dengan *Loading* ke database. Mekanisme ini memastikan pengguna mendapatkan rekomendasi yang selalu diperbarui setiap harinya tanpa intervensi manual.



Gambar 3.43 Apache Airflow DAG Graph View

Gambar 3.11 merupakan visualisasi *Graph View* dari DAG tersebut yang memperlihatkan alur kerja data yang terbagi menjadi empat tahap utama. Indikator "success" berwarna hijau pada setiap kotak mengonfirmasi bahwa seluruh rangkaian tugas telah berhasil dieksekusi tanpa galat.

1. **Tahap Ingestion (Akuisisi Data):** Alur dimulai di kolom paling kiri dengan empat tugas yang berjalan secara paralel untuk mempercepat waktu pemrosesan. Tugas tersebut meliputi ingest_sql (ekstraksi database), ingest_aqi (API kualitas udara), ingest_bmkg (API cuaca), dan ingest_sheets (Google Sheets).
2. **Tahap Cleaning (Transformasi):** Setiap tugas akuisisi memiliki pasangan pembersihannya masing-masing, seperti clean_sql, clean_aqi, clean_bmkg, dan clean_sheets. Pada tahap ini, data mentah diubah menjadi format Delta Lake yang terstandarisasi di dalam *clean-zone*.
3. **Tahap Prescriptive Logic (Inti Analisis):** Seluruh jalur transformasi bermuara pada satu titik temu di tengah, yaitu tugas prescriptive_logic. Tugas ini bertindak sebagai *bottleneck* yang hanya akan berjalan setelah keempat sumber data di atas dinyatakan siap. Di sinilah algoritma utama menggabungkan variabel aktivitas, cuaca, dan polusi untuk menghasilkan keputusan otomatis.
4. **Tahap Loading (Penyajian):** Rangkaian diakhiri dengan pemuatan data ke database Neon untuk kebutuhan visualisasi. Tahap ini mencakup load_riwayat_mandi_to_neon, load_activity_to_neon untuk detail aktivitas, serta load_result_to_neon untuk menyimpan hasil akhir skor dan rekomendasi preskriptif.

3.5. Implementasi Visualisasi Dashboard

Sebagai antarmuka akhir bagi pengguna (end-user), sistem visualisasi diimplementasikan menggunakan Metabase yang terhubung langsung dengan basis data Neon berbasis PostgreSQL. Dashboard ini dirancang untuk menyajikan hasil analisis preskriptif yang dihasilkan oleh sistem secara ringkas, intuitif, dan mudah ditindaklanjuti. Seluruh visualisasi pada dashboard bersumber dari data serving layer yang telah melalui proses kurasi dan pemrosesan pada arsitektur Data Lakehouse.

3.5.1 Indikator Utama Keputusan (Key Performance Indicator)

Indikator utama pada dashboard diperoleh dengan melakukan agregasi langsung terhadap tabel hasil rekomendasi preskriptif. Proses pengolahan meliputi perhitungan jumlah total rekomendasi, persentase rekomendasi dengan status Wajib Mandi, rata-rata skor kebutuhan mandi, serta rata-rata durasi waktu sejak mandi terakhir.

Perhitungan ini dilakukan menggunakan fungsi agregasi seperti COUNT, AVG, dan ekspresi kondisional pada kueri SQL, sehingga indikator yang ditampilkan merepresentasikan kondisi sistem secara ringkas dalam periode waktu tertentu.

```
1 |SELECT
2 |      count(*) as "Total Rekomendasi Sistem"
3 |FROM
4 |      rekomendasi_mandi_preskriptif
```

Gambar 3.44 Query SQL

```

1 v |SELECT
2 v     ROUND(
3 v         100.0 * SUM(
4 v             CASE WHEN rekomendasi LIKE 'WAJIB MANDI%' THEN 1 ELSE 0 END
5 v         ) / COUNT(*),
6 v         2
7 v     ) AS persen_wajib_mandi
8 v FROM rekomendasi_mandi_preskriptif;
9

```

Gambar 3.45 Query SQL

```

1 v |SELECT
2 v     avg(skor_final)
3 v     FROM
4 v     rekomendasi_mandi_preskriptif
5

```

Gambar 3.46 Query SQL

```

1 v |SELECT
2 v     avg(jam_sejak_mandi)
3 v     FROM
4 v     rekomendasi_mandi_preskriptif
5

```

Gambar 3.47 Query SQL



Gambar 3.48 Hasil Visual

Key Performance Indicator (KPI) yang merangkum efektivitas sistem higinitas Berdasarkan data yang tersaji pada dashboard, Key Performance Indicator (KPI) atau indikator kinerja utama berfungsi untuk merangkum efektivitas sistem rekomendasi higinitas Anda secara keseluruhan. Berikut adalah penjelasan detail mengenai isi KPI tersebut secara umum:

Total Rekomendasi Sistem: Metrik ini menunjukkan akumulasi seluruh keputusan yang telah dikeluarkan oleh mesin analitik, yaitu sebanyak 16 rekomendasi. Angka ini mencerminkan seberapa aktif sistem melakukan pemantauan terhadap kondisi tubuh Anda.

Wajib Mandi (%): Ini merupakan indikator tingkat urgensi yang menunjukkan bahwa 43,75% dari seluruh riwayat data berakhir pada status kritis atau "Wajib Mandi". Tingginya persentase ini menandakan bahwa pengguna sering membiarkan kondisi tubuh mencapai ambang batas kotor sebelum memutuskan untuk mandi.

Rata-Rata Skor Kebutuhan Mandi: Angka 5.21 menunjukkan nilai tengah dari seluruh evaluasi skor kebersihan yang dilakukan. Nilai ini memberikan gambaran bahwa secara umum, kondisi tubuh pengguna sering berada di level menengah (mendekati ambang batas disarankan) dalam siklus pemantauan harian.

Rata-Rata Jam Sejak Mandi: Indikator ini mencatat durasi disiplin pembersihan diri, dengan hasil rata-rata 13.79 jam. Angka yang melebihi 12 jam ini menjadi alasan logis

mengapa persentase "Wajib Mandi" sangat tinggi, karena durasi aktivitas yang panjang secara otomatis meningkatkan akumulasi skor bau dan kekotoran.

3.5.2 Visualisasi Keputusan dan Faktor Penentu

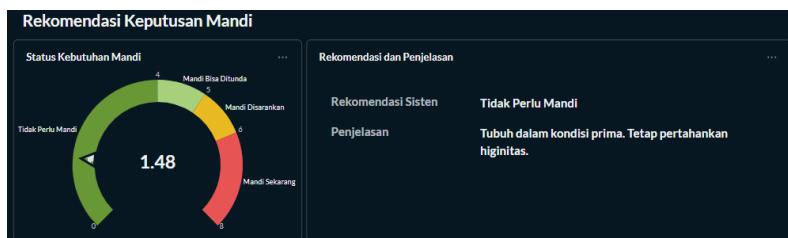
Untuk menampilkan keputusan sistem secara eksplisit, data skor akhir dipetakan ke dalam kategori rekomendasi berdasarkan nilai ambang batas (threshold) yang telah ditentukan pada logika preskriptif. Nilai skor ini kemudian divisualisasikan menggunakan gauge chart guna menunjukkan tingkat urgensi kebutuhan mandi secara intuitif.

```
1 v |SELECT
2     skor_final
3  FROM rekomendasi_mandi_preskriptif
4 ORDER BY generated_at DESC
5 LIMIT 1;
```

Gambar 3.49 Query SQL

```
1 v |SELECT
2     rekomendasi,
3     penjelasan
4  FROM rekomendasi_mandi_preskriptif
5 ORDER BY generated_at DESC
6 LIMIT 1;
```

Gambar 3.50 Query SQL



Gambar 3.51 Hasil Visual

Visualisasi ini berfungsi sebagai indikator utama tingkat kebersihan secara real-time. Saat ini, jarum menunjukkan angka 1,48, yang menempatkan pengguna pada kategori "Tidak Perlu Mandi". Skala ini memiliki pembagian zona warna yang jelas: zona hijau untuk kondisi aman, kuning untuk mulai kotor, dan merah untuk kondisi kritis. Jika angka ini terus naik dan melewati ambang batas 6.0, sistem akan mengubah status menjadi rekomendasi yang lebih mendesak.

Selain itu, pengolahan data juga dilakukan untuk memisahkan kontribusi masing-masing faktor penyusun skor akhir, yaitu faktor kekotoran aktivitas, bau badan, dan kualitas udara. Data setiap faktor dihitung dan ditampilkan dalam bentuk grafik batang untuk memberikan transparansi terhadap proses pengambilan keputusan sistem.

```

1 v SELECT
2     generated_at as "Timestamp",
3     skor_kekotoran as "Skor Kekotoran",
4     skor_bau "Skor Bau Badan",
5     skor_aqi as "Skor AQI"
6 FROM rekomendasi_mandi_preskriptif
7 ORDER BY generated_at DESC
8 LIMIT 1;
9

```

Gambar 3.52 Query SQL



Gambar 3.53 Hasil Visual

Grafik batang ini menguraikan variabel apa saja yang membuat skor Anda meningkat. Pada data tanggal 22 Desember 2025, terlihat bahwa:

Skor Bau Badan: Menjadi penyumbang terbesar dengan nilai menyentuh angka 3.

Skor Kekotoran: Menyumbang beban kotor sebesar 0.5.

Skor AQI: Memiliki pengaruh paling kecil, yakni hanya di angka 1.16. Ini menunjukkan bahwa meskipun Anda mungkin tidak merasa terlalu berdebu, faktor aroma tubuh adalah alasan utama sistem menyarankan Anda untuk mandi.

3.5.3 Analisis Tren dan Riwayat Historis

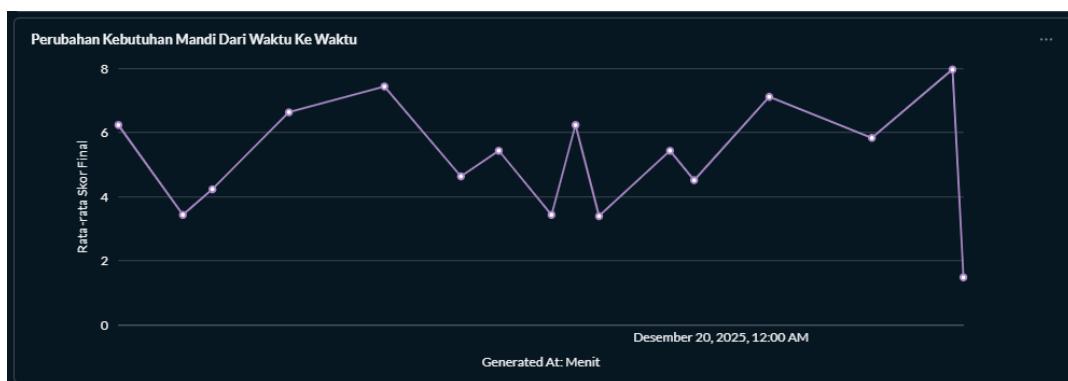
Pengolahan data berbasis waktu dilakukan untuk menganalisis perubahan kebutuhan mandi secara historis. Data skor akhir dikelompokkan berdasarkan dimensi waktu dan dihitung nilai rata-ratanya untuk menghasilkan grafik tren perubahan kebutuhan mandi dari waktu ke waktu..

```

1 v |SELECT
2 v   DATE_TRUNC(
3 v     'minute',
4 v     "public"."rekомендации_мани_preskriptif"."generated_at"
5 v   ) AS "generated_at",
6 v   AVG(
7 v     "public"."rekомендации_мани_preskriptif"."skor_final"
8 v   ) AS "avg"
9 v FROM
10 v   "public"."rekомендации_мани_preskriptif"
11 v GROUP BY
12 v   DATE_TRUNC(
13 v     'minute',
14 v     "public"."rekомендации_мани_preskriptif"."generated_at"
15 v   )
16 v ORDER BY
17 v   DATE_TRUNC(
18 v     'minute',
19 v     "public"."rekомендации_мани_preskriptif"."generated_at"
20 v   ) ASC

```

Gambar 3.54 Query SQL



Gambar 3.55 Hasil Visual

Grafik garis ini memantau fluktuasi skor kebersihan Anda dari waktu ke waktu.

Tren Kenaikan: Skor akan merangkak naik seiring bertambahnya aktivitas dan durasi sejak mandi terakhir.

Titik Penurunan: Terdapat penurunan tajam pada grafik (misalnya di sekitar jam 12:00 PM), yang menandakan adanya aktivitas mandi yang berhasil mereset skor kebersihan kembali ke titik rendah.

Selain itu, data riwayat rekomendasi dan riwayat mandi ditampilkan dalam bentuk tabel untuk menyajikan rekam jejak keputusan sistem dan aktivitas pengguna secara detail. Pengolahan ini memungkinkan pengguna melakukan evaluasi terhadap pola kebiasaan mandi serta konsistensi rekomendasi sistem dalam jangka waktu tertentu.

```

1 v |SELECT
2   "public"."rekомендasi_mandi_preskriptif"."waktu_mandi_terakhir" AS "waktu_mandi_terakhir",
3   "public"."rekомендasi_mandi_preskriptif"."jam_sejak_mandi" AS "jam_sejak_mandi",
4   "public"."rekомендasi_mandi_preskriptif"."skor_kekotoran" AS "skor_kekotoran",
5   "public"."rekомендasi_mandi_preskriptif"."skor_bau" AS "skor_bau",
6   "public"."rekомендasi_mandi_preskriptif"."skor_aqi" AS "skor_aqi",
7   "public"."rekомендasi_mandi_preskriptif"."skor_final" AS "skor_final",
8   "public"."rekомендasi_mandi_preskriptif"."rekомендации" AS "rekомендации",
9   "public"."rekомендasi_mandi_preskriptif"."пояснение" AS "пояснение",
10  "public"."rekомендasi_mandi_preskriptif"."generated_at" AS "generated_at"
11 FROM
12   "public".rekомендации_mandi_preskriptif"
13 ORDER BY
14   "public".rekомендации_mandi_preskriptif.generated_at" DESC
15 LIMIT
16  1048575

```

Gambar 3.56 Query SQL

Riwayat Rekomendasi Sistem						
Generated At	Waktu Mandi Terakhir	Rekomendasi	Jam Sejak Mandi	Skor Kekotoran	Skor Bau	Skor Aqi
Desember 22, 2025, 12:07 PM	Desember 22, 2025, 9:30 AM	Tidak Perlu Mandi	2.6	0.5	3	1.16
Desember 22, 2025, 9:25 AM	Desember 21, 2025, 1:31 PM	WAJIB MANDI (Faktor Bau)	19.6	10	9	1.16
Desember 21, 2025, 1:25 PM	Desember 21, 2025, 12:00 PM	WAJIB MANDI SEKARANG	25.5	10	10	1.16
Desember 20, 2025, 11:57 AM	Desember 19, 2025, 5:22 PM	WAJIB MANDI SEKARANG	18.6	8.2	9	1.16
Desember 19, 2025, 5:19 PM	Desember 19, 2025, 11:25 AM	MANDI BISA DITUNDA	5.9	4.7	6	1.16

Gambar 3.57 Hasil Visual

Visualisasi ini merangkum riwayat keputusan yang telah dikeluarkan oleh sistem dari total 20 rekomendasi yang ada.

```

1 v |SELECT
2   "public"."riwayat_mandi"."waktu_mandi" AS "waktu_mandi",
3   "public"."riwayat_mandi"."tingkat_kekotoran" AS "tingkat_kekotoran",
4   "public"."riwayat_mandi"."tingkat_bau_badan" AS "tingkat_bau_badan"
5 FROM
6   "public".riwayat_mandi"
7 LIMIT
8  1048575

```

Gambar 3.58 Query SQL

Riwayat Mandi			
Waktu Mandi	Tingkat Kekotoran	Tingkat Bau Badan	
Desember 13, 2025, 7:00 AM	4		3
Desember 13, 2025, 7:30 PM	8		7
Desember 14, 2025, 10:30 AM	3		5
Desember 14, 2025, 5:50 PM	6		4
Desember 15, 2025, 12:51 PM	9		7
Desember 16, 2025, 12:32 PM	8		10
Desember 17, 2025, 7:20 AM	4		5
< >		10 baris	

Gambar 3.59 Hasil Visual

BAB IV

4.1. Persiapan Lingkungan Sistem

Tahap awal dalam penggunaan sistem Smart Hygiene Decision Support System adalah melakukan persiapan lingkungan eksekusi. Seluruh komponen sistem dibangun menggunakan pendekatan containerization untuk memastikan konsistensi lingkungan dan kemudahan replikasi pada perangkat lain. Seluruh layanan dikonfigurasi dan dikelola menggunakan Docker dan Docker Compose.

Pengguna diwajibkan memastikan bahwa perangkat telah terpasang perangkat lunak Docker Engine dan Docker Compose. Sistem operasi yang digunakan tidak menjadi batasan selama mendukung virtualisasi kontainer, seperti Windows, Linux, maupun macOS.

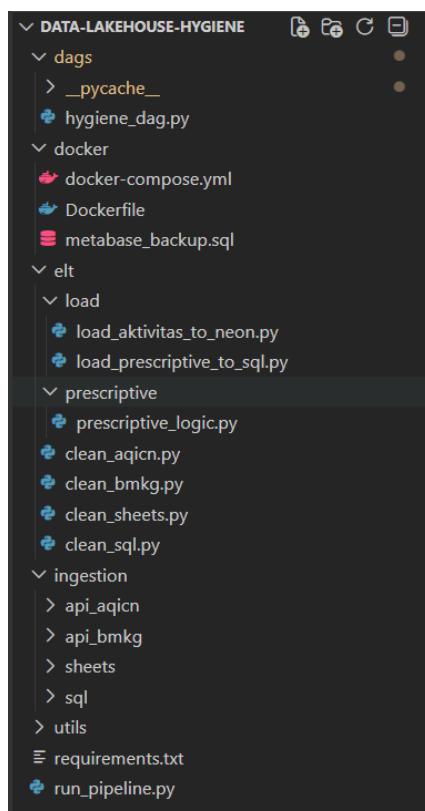
Seluruh source code sistem disimpan dalam satu repositori GitHub yang mencakup konfigurasi infrastruktur, skrip pemrosesan data, serta definisi pipeline orkestrasi. Proses persiapan sistem dimulai dengan melakukan kloning repositori proyek ke dalam direktori lokal pengguna.

Perintah kloning repositori:

Tabel 4.12 Perintah Kloning Repository

```
git clone https://github.com/Andra-Braputra/data-lakehouse-hygiene.git
```

Setelah repositori berhasil diunduh, struktur direktori proyek akan menampilkan folder utama seperti dags, scripts, data, serta berkas `docker-compose.yml` yang menjadi pusat konfigurasi layanan.



Gambar 4.60 Struktur Direktori Project

4.2. Inisialisasi dan Menjalankan Infrastruktur Sistem

Setelah persiapan direktori selesai, langkah berikutnya adalah menjalankan seluruh layanan sistem secara bersamaan. Proses ini dilakukan menggunakan Docker Compose yang akan membangun dan menjalankan container Apache Airflow, MinIO, PostgreSQL (Neon-compatible), dan Metabase.

Perintah berikut digunakan untuk menjalankan sistem:

Tabel 4.13 Perintah Untuk Menjalankan Sistem

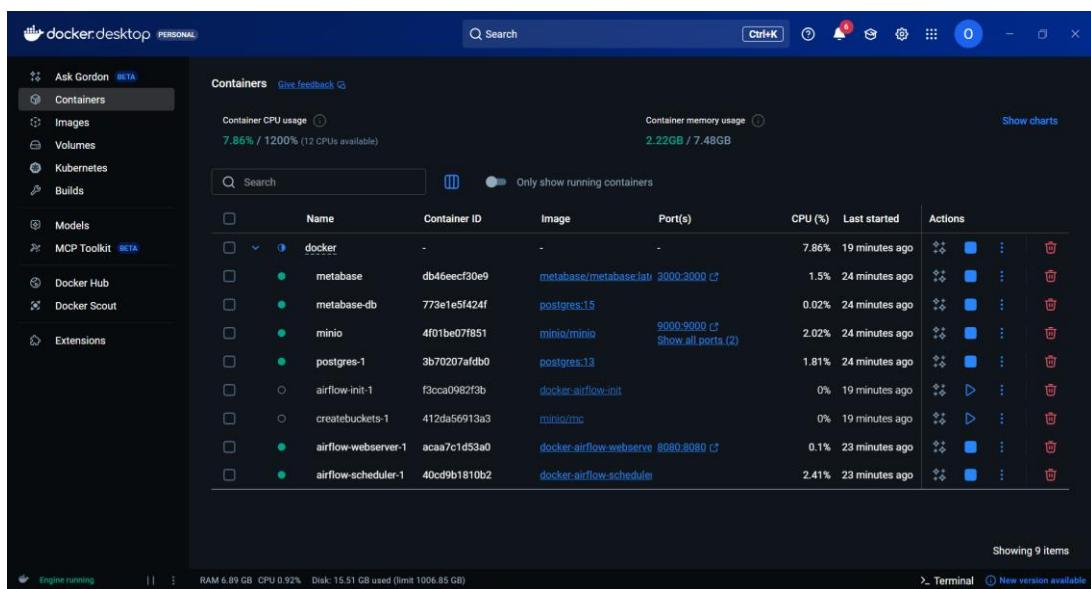
```
docker compose up -d
```

Perintah tersebut akan mengeksekusi seluruh layanan dalam mode background. Pengguna dapat memverifikasi status layanan melalui Docker Desktop atau menggunakan perintah terminal untuk memastikan seluruh container berada dalam kondisi berjalan (running).

Layanan utama yang harus aktif meliputi:

- Airflow Webserver dan Scheduler
- MinIO Object Storage
- PostgreSQL
- Metabase

Jika seluruh container berjalan dengan baik, maka sistem Data Lakehouse siap digunakan.



Gambar 4.61 Container Apache Airflow, MinIO, PostgreSQL

4.3. Konfigurasi Awal dan Akses Layanan

Setelah seluruh layanan aktif, pengguna dapat mengakses masing-masing komponen sistem melalui browser dengan alamat berikut:

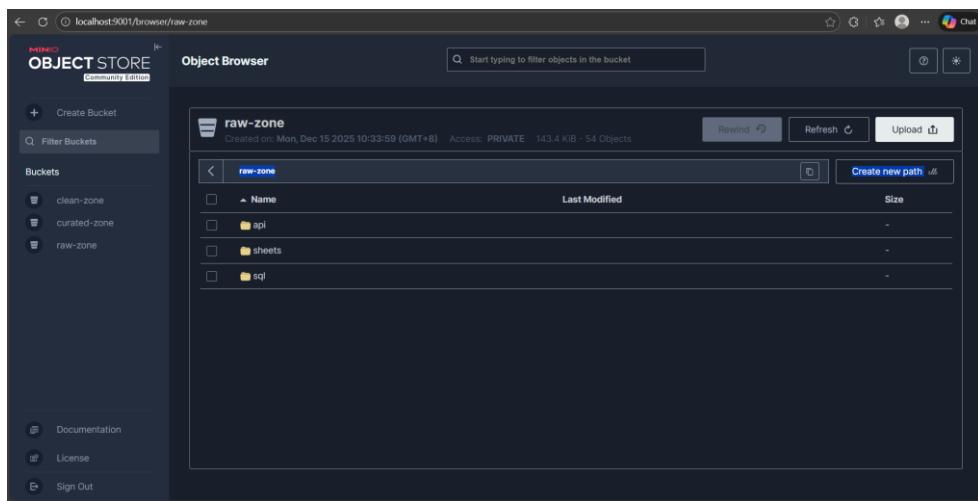
- Apache Airflow Web UI: digunakan untuk memantau dan mengeksekusi pipeline
- MinIO Console: digunakan untuk memverifikasi penyimpanan data pada raw, clean, dan curated zone
- Metabase Dashboard: digunakan untuk melihat hasil visualisasi dan analisis preskriptif

Pada tahap ini, pengguna tidak perlu melakukan konfigurasi manual tambahan karena seluruh variabel lingkungan, kredensial, serta koneksi antar layanan telah didefinisikan di dalam berkas konfigurasi Docker.

Validasi awal dapat dilakukan dengan memastikan bucket MinIO untuk raw, clean, dan curated zone telah tersedia serta database PostgreSQL telah terinisialisasi dengan benar.

Login ke Minio dengan :

- Username : minioadmin
- Password : minioadmin123



Gambar 4.62 Halaman Object Storage MinIO

4.4. Eksekusi Pipeline Data Lakehouse

Setelah sistem siap, proses analisis dimulai dengan menjalankan pipeline data lakehouse. Pipeline ini diorkestrasikan menggunakan Apache Airflow dalam sebuah DAG bernama hygiene_lakehouse_pipeline.

Pengguna dapat menjalankan pipeline secara manual melalui antarmuka Airflow dengan menekan tombol Trigger DAG. Setelah DAG dijalankan, sistem akan mengeksekusi seluruh tahapan secara berurutan sesuai dependensi yang telah dirancang.

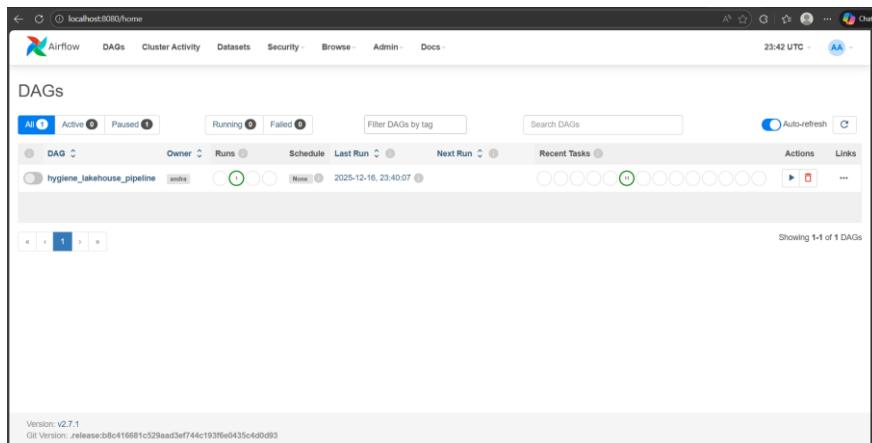
Tahapan eksekusi pipeline meliputi:

1. Akuisisi data dari database SQL, Google Sheets, API BMKG, dan API AQICN.
2. Penyimpanan data mentah ke dalam raw zone tanpa modifikasi.
3. Proses pembersihan dan standarisasi data ke dalam clean zone menggunakan format Delta Lake.
4. Penggabungan data dan penerapan logika analisis preskriptif pada curated zone.
5. Pemuatan hasil akhir ke database PostgreSQL untuk visualisasi.

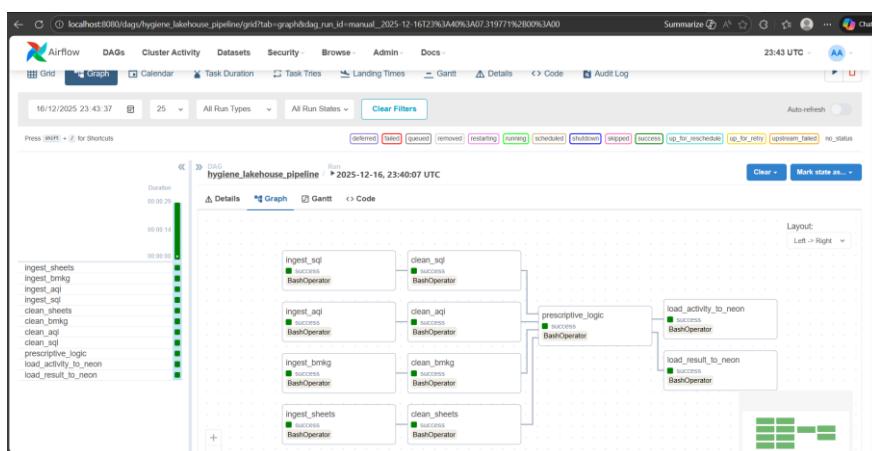
Setiap tahapan hanya akan dieksekusi apabila tahap sebelumnya berhasil, sehingga menjaga konsistensi dan integritas data.

Untuk Login ke Airflow dapat menggunakan :

- Username : admin
- Password : admin



Gambar 4.63 Apache Airflow Dag



Gambar 4.64 Halaman Graph Apache Airflow

	generated_at	activity	start_time	end_time	duration	status
1	2025-12-16 18:37:41.76...	wash_mandibular_teeth	2025-12-16 12:32:45	2025-12-16 12:32:45	0.02	success
2	2025-12-16 18:37:49.95...	wash_mandibular_teeth	2025-12-16 12:32:45	2025-12-16 12:32:45	0.02	success
3	2025-12-16 18:39:27.47...	wash_mandibular_teeth	2025-12-16 12:32:45	2025-12-16 12:32:45	0.02	success
4	2025-12-16 18:39:37.59...	wash_mandibular_teeth	2025-12-16 12:32:45	2025-12-16 12:32:45	0.02	success
5	2025-12-16 18:40:28.53...	wash_mandibular_teeth	2025-12-16 12:32:45	2025-12-16 12:32:45	0.02	success

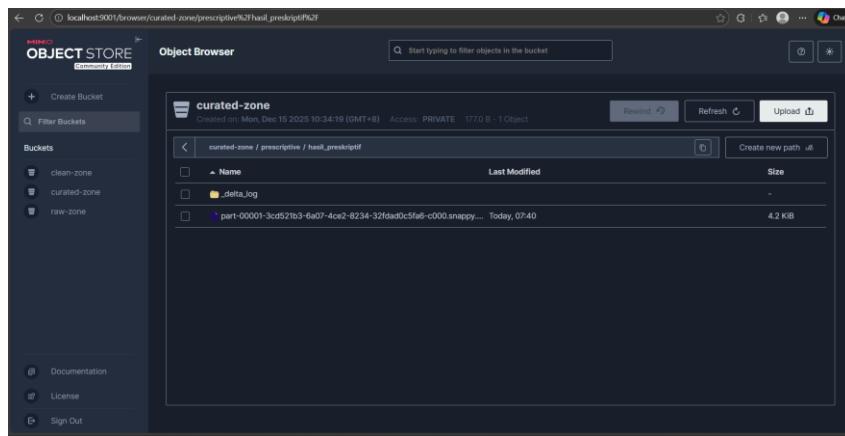
Gambar 4.65 Database PostgreSQL NeonDB

4.5. Eksekusi Pipeline Data Lakehouse

Setelah pipeline selesai dijalankan, pengguna dapat melakukan validasi hasil pemrosesan data pada setiap zona penyimpanan.

Pada raw zone, data tersimpan dalam format asli seperti CSV dan JSON yang dilengkapi timestamp, menandakan keberhasilan proses ingestion. Pada clean zone, data telah tersimpan dalam format Delta Lake dengan struktur skema yang konsisten. Pada curated zone, data telah mengalami agregasi dan pengolahan logika preskriptif.

Validasi ini memastikan bahwa alur ETL dan ELT berjalan sesuai dengan rancangan arsitektur data lakehouse.



Gambar 4.66 Hasil Eksekusi Pipeline di curated-zone

4.6. Visualisasi dan Hasil Analisis Preskriptif

Tahap akhir penggunaan sistem adalah melihat hasil rekomendasi melalui dashboard Metabase. Dashboard ini terhubung langsung dengan database PostgreSQL yang menyimpan hasil analisis dari curated zone.

Sebelum mengunjungi metabase kita perlu melakukan restore database dulu dari file backup yang sudah disediakan. Jalankan perintah berikut setelah “`docker-compose up -d`” di “cmd” folder `docker/` :

Tabel 4.14 Perintah Untuk Restore Dashboard Metabase

```
docker exec -i metabase-db psql -U metabase metabase <
metabase backup utf8.sql
```

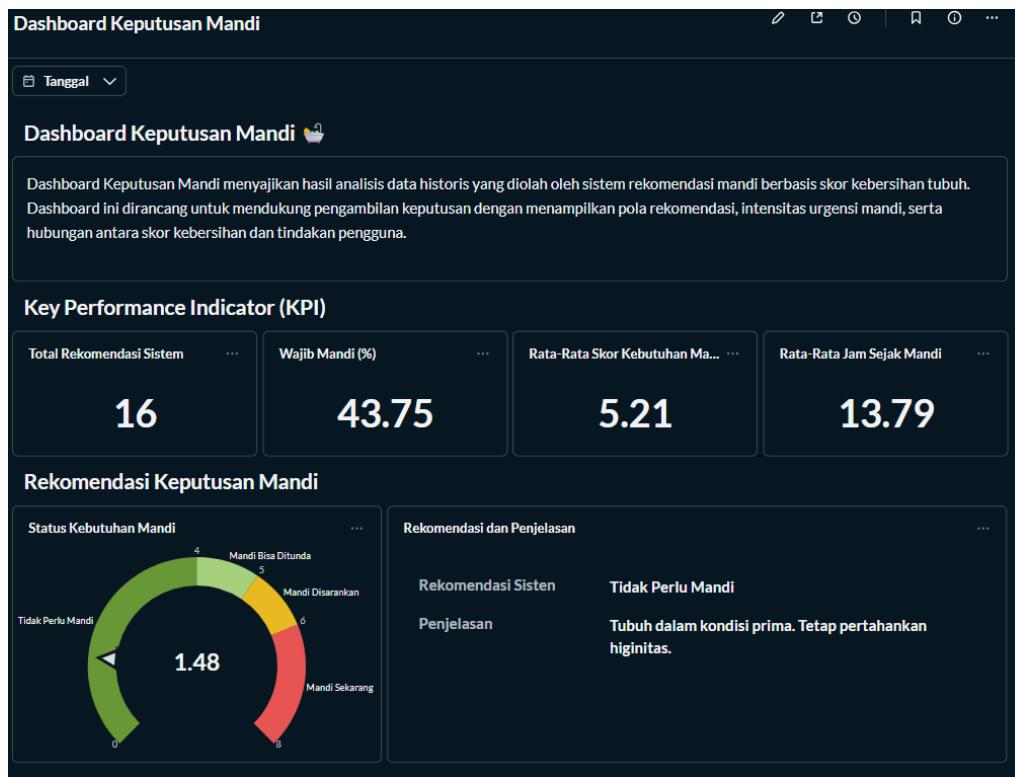
Untuk Login Metabase dapat menggunakan :

- Email : 2310817310012@mhs.ulm.ac.id
- Password : azwin719

Masuk Kebagian Analytic Kami, pilih folder hygien, dan klik dashboard Keputusan Mandi.

Dashboard akan menampilkan skor kebutuhan mandi, status rekomendasi, serta visualisasi faktor-faktor yang memengaruhi keputusan. Pengguna dapat langsung mengetahui apakah sistem merekomendasikan untuk mandi segera, menunda mandi, atau tidak perlu mandi.

Keputusan yang ditampilkan bersifat preskriptif karena sistem tidak hanya menyajikan data historis, tetapi juga memberikan rekomendasi tindakan berdasarkan perhitungan objektif dari berbagai sumber data.



Gambar 4.67 Halaman Dashboard Metabase

BAB V

5.1. Insight Implementasi

Insight yang diperoleh dari proses pembangunan arsitektur Data Lakehouse adalah:

1. Efektivitas Arsitektur Medallion dalam Menjaga Integritas Data: Penerapan zona Raw, Clean, dan Curated menggunakan format Delta Lake terbukti efektif dalam menangani data heterogen (SQL, Google Sheets, dan API). Penggunaan Delta Lake menjamin konsistensi skema dan transaksi ACID, sehingga data terhindar dari kerusakan meskipun terjadi kegagalan saat proses transformasi.
2. Otomasi Pipeline Melalui Orkestrasi: Penggunaan Apache Airflow sebagai orkestrator memungkinkan seluruh alur kerja, mulai dari akuisisi data (ingestion), pembersihan (cleaning), hingga pemuatan ke database (loading)—berjalan secara otomatis dan terjadwal. Hal ini memastikan bahwa sistem selalu memberikan rekomendasi yang aktual tanpa memerlukan intervensi manual setiap harinya.
3. Standarisasi Data dari Sumber Heterogen: Proses transformasi pada Silver Layer berhasil menyatukan data terstruktur (SQL) dan semi-terstruktur (Sheets/JSON API) ke dalam satu format tabular yang seragam. Hal ini sangat krusial agar logika analitik preskriptif dapat menghitung skor secara akurat dengan menggabungkan berbagai dimensi data.

5.2. Insight dari Dashboard

Insight yang diperoleh dari hasil visualisasi pada dashboard adalah:

1. Dominasi Faktor Bau Badan terhadap Keputusan: Melalui grafik kontribusi faktor, terlihat bahwa skor bau badan seringkali menjadi penyumbang terbesar (mencapai angka 3) dibandingkan skor kekotoran fisik atau kualitas udara (AQI). Hal ini memberikan pemahaman bahwa meskipun paparan debu rendah, aroma tubuh tetap menjadi alasan utama sistem menyarankan untuk mandi.
2. Urgensi Higienitas Melalui KPI: Data Key Performance Indicator (KPI) menunjukkan bahwa 43.75% dari seluruh riwayat data berakhir pada status "Wajib Mandi". Hal ini merefleksikan bahwa pengguna cenderung membiarkan kondisi tubuh mencapai ambang batas kotor sebelum memutuskan untuk mandi, yang juga diperkuat oleh rata-rata waktu sejak mandi terakhir yang mencapai 13,79 jam.
3. Visualisasi Preskriptif untuk Pengambilan Keputusan: Penggunaan Gauge Chart dengan skor seperti 1.48 mempermudah pengguna memahami status kebersihan diri secara real-time. Dengan pembagian zona warna (hijau, kuning, merah), pengguna mendapatkan rekomendasi tindakan yang jelas, apakah mandi bisa ditunda atau harus segera dilakukan berdasarkan ambang batas (threshold) yang telah ditetapkan.

Daftar Pustaka

- Ait Errami, S., Hajji, H., Ait El Kadi, K., & Badir, H. (2023). Spatial big data architecture: From Data Warehouses and Data Lakes to the LakeHouse. *Journal of Parallel and Distributed Computing*, 176, 70–79. <https://doi.org/10.1016/j.jpdc.2023.02.007>
- Armbrust, M., Ghodsi, A., Xin, R., Zaharia, M., & Berkeley, U. (2021). *Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics*.
- Harby, A. A. (2022). *From Data Warehouse to Lakehouse: A Comparative Review*. IEEE.
- Harby, A. A., & Zulkernine, F. (2025). Data Lakehouse: A survey and experimental study. *Information Systems*, 127, 102460. <https://doi.org/10.1016/j.is.2024.102460>
- Nguyen, T., Nguyen, H.-T., & Nguyen-Hoang, T.-A. (2025). Data quality management in big data: Strategies, tools, and educational implications. *Journal of Parallel and Distributed Computing*, 200, 105067. <https://doi.org/10.1016/j.jpdc.2025.105067>
- Otaki, Y., & Maeda, A. (2022). Water-Saving Tips With a Visualized Indicator Related to the Environment. *Frontiers in Water*, 4. <https://doi.org/10.3389/frwa.2022.914665>
- Skowron, K., Bauza-kaszewska, J., Kraszewska, Z., Wiktorczyk-kapischke, N., Grudlewska-buda, K., Kwiecińska-piróg, J., Wałecka-zacharska, E., Radtke, L., & Gospodarek-komkowska, E. (2021). Human skin microbiome: Impact of intrinsic and extrinsic factors on skin microbiota. In *Microorganisms* (Vol. 9, Issue 3, pp. 1–20). MDPI AG. <https://doi.org/10.3390/microorganisms9030543>

Lampiran

Lampiran 1.

Link Repositori Github.

<https://github.com/Andra-Braputra/data-lakehouse-hygiene.git>

Lampiran 2.

Referensi Skor METs.

MET Levels of Common Recreational Activities



What is a MET?

MET stands for Metabolic Equivalent

One MET is the amount of energy (calories) your body uses each minute while resting quietly. On average, a man sitting quietly burns 70 calories per hour, and a woman sitting quietly burns about 60 calories per hour. The MET level is higher as the intensity of your activity increases. For example, 2.5 METs is the amount of energy used each minute to walk leisurely, but that goes up to 5 METs when walking very briskly at 4 mph. You are burning 5 times as many calories per minute when walking briskly as when sitting quietly.

Measuring Exercise Intensity

Activity Description	Intensity on a "0-10" Scale*	MET level (Sitting = 1 MET)	Breathing & Heart Rate	How it feels; example
EASY	3-4	Less than 3.0 METs	Minimal increase	Feels easy (e.g., easy walking)
MODERATE	5-6	3.0-6.0 METs	Noticeable increase	Feels fairly easy to somewhat hard (e.g., brisk walking)
VIGOROUS	7-8	Greater than 6.0 METs	Large increase in breathing & heart rate but not out-of-breath	Feels somewhat hard to hard (e.g., jogging, vigorous sports)

*Intensity scale: On a scale of 0–10 where 0 = sitting and 10 = all-out effort

Walk, Jog, Run!

	METs	Calories per 60 mins*
Walking, slowly (stroll)	2.0	145
Walking, 2 mph	2.5	215
Walking, 3 mph (20 min/mile)	3.3	245
Walking, 17 min/mile	3.8	285
Walking, 15 min/mile	5.0	360
Race walking, moderate pace	6.5	465
Hiking up hills	6.9	500
Hiking hills, 12 lb pack	7.5	540
Jogging, 12 min/mile	8.0	575
Running, 10 min/mile	10.0	715
Running, 9 min/mile	11.0	790
Running, 8 min/mile	12.5	855
Running, 7 min/mile	14.0	1000
Running, 6 min/mile	16.0	1145

*Approximation based on 150 lb. person

Ready, Set, Bike!

	METs	Calories per 60 mins*
Stationary cycling, 50 watts	3.0	215
Bicycling, leisurely	3.5	250
Stationary cycling, 100 watts	5.5	395
Bicycling, 12-13 mph	8.0	575
Bicycling, 14-15 mph	10.0	715
Bicycling, 16-19 mph	12.0	860
Bicycling, 20+ mph	16.0	1145

*Approximation based on 150 lb. person

MET Levels

Listed alphabetically
by category of intensity

Light activities (<3 METs)*	METs
Canoeing leisurely	2.5
Croquet	2.5
Dancing, ballroom, slow	2.9
Fishing, standing	2.5
Golf with a cart	2.5
Housework, light	2.5
Playing catch	2.5
Playing a piano	2.5
Sitting quietly	1.0
Stretching exercises, yoga	2.5
Walking, 2 mph	2.5

*Calories burned = up to 215/hour



Moderate activities (3-6 METs)*	METs	Moderate activities (3-6 METs)*	METs
Aerobic dance, low impact	5.0	Jumping on mini tramp	4.5
Archery	3.5	Kayaking	5.0
Badminton	4.5	Mowing lawn, walking	5.5
Baseball or softball	5.0	Raking the lawn	4.0
Basketball, shooting baskets	4.5	Shoveling snow	6.0
Bicycling, leisurely	3.5	Skateboarding	5.0
Bowling	3.0	Skiing downhill, moderate	6.0
Calisthenics, light to moderate	3.5	Snorkeling	5.0
Canoeing, 3 mph	3.0	Snowmobiling	3.5
Chopping wood	6.0	Surfing	6.0
Dancing, aerobic or ballet	6.0	Swimming, moderate pace	4.5
Dancing, modern, fast	4.8	Table tennis	4.0
Fencing	6.0	Tai chi	4.0
Fishing, walking and standing	3.5	Tennis, doubles	5.0
Foot bag, hacky sack	4.0	Trampoline	3.5
Gardening, active	4.0	Volleyball, noncompetitive	3.0
Golf, walking	4.4	Walking, 15 min/mile	5.0
Gymnastics	4.0	Walking, brisk up hills	6.0
Hiking cross country	6.0	Water skiing	6.0
Horseback riding	4.0	Weight lifting, heavy workout	6.0
Ice skating	5.5	Wrestling	6.0

*Calories burned = 215–430/hour



Vigorous activities (>6 METs)*	METs
Aerobic dance	6.5
Aerobic dance, high impact	7.0
Aerobic stepping, 6-8 inches	8.5
Backpacking	7.0
Basketball game	8.0
Bicycling, 12-13 mph	8.0
Bicycling, 20+ mph	16.0
Calisthenics, heavy, vigorous	8.0
Canoeing, 5 mph or portaging	7.0
Fishing in stream with waders	6.5
Football, competitive	9.0
Football, touch/flag	8.0
Frisbee, ultimate	8.0
Hockey, field or ice	8.0
Ice skating, social	7.0
Jogging, 12 min/mile	8.0
Judo/karate/tae kwon do	10.0



Vigorous activities (>6 METs)*	METs
Lacrosse	8.0
Logging/felling trees	8.0
Mountain climbing	8.0
Racquetball	10.0
Racquetball, team	8.0
Roller skating	7.0
Rollerblading, fast	12.0
Rope skipping, slow	8.0
Rope skipping, fast	12.0
Running, 10 min/mile	10.0
Running, 6 min/mile	16.0
Running, 7 min/mile	14.0
Running, 8 min/mile	12.5
Running, 9 min/mile	11.0
Skiing cross country, slow	7.0
Skiing cross country, moderate	8.0
Skiing cross country, racing uphill	16.5



Vigorous activities (>6 METs)*	METs
Skiing cross country, vigorous	9.0
Skiing down hill, vigorous	8.0
Skin diving	12.5
Snow shoeing	8.0
Soccer, casual	7.0
Soccer, competitive	10.0
Swimming laps, fast	10.0
Swimming laps, moderate pace	7.0
Swimming laps, sidestroke	8.0
Swimming recreational	6.0
Tennis	7.0
Volleyball, competitive/beach	8.0
Walking, 11 min/mile	11.0
Walking up stairs	8.0
Water jogging	8.0
Water polo	10.0



*Calories burned = 430+/hour

How many calories is that? You can calculate the number of calories you burn for any activity by using the following equation:

$$\text{Exercise calories} = (\text{MET level of activity} \times 3.5 \times \text{Weight (kg)} \times \text{minutes of activity}) / 200$$

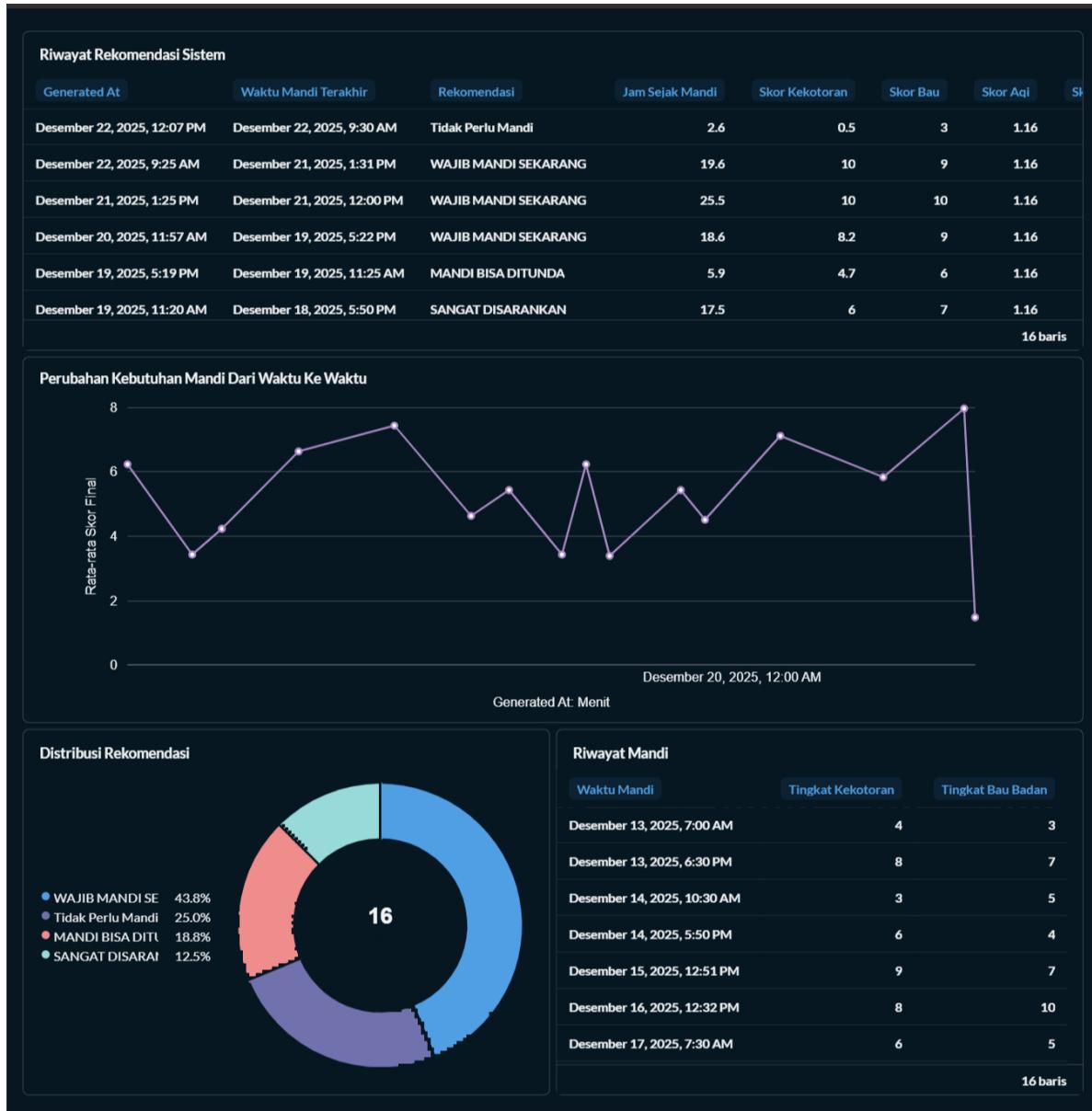
Reference: American College of Sports Medicine. *The Compendium of Physical Activities. ACSM Resource Manual 5th Edition*, 2006.



© 2008 Wellsource, Inc., Clackamas, Oregon. All rights reserved. For personal use only – do not make unauthorized copies.

Written by Don Hall, DrPH, CHES

Halaman Dashboard Analisis Kebutuhan Mandi.



Riwayat Rekomendasi Sistem

Generated At	Waktu Mandi Terakhir	Rekomendasi	Jam Sejak Mandi	Skor Kekotoran	Skor Bau	Skor Aqi	Skor
Desember 22, 2025, 12:07 PM	Desember 22, 2025, 9:30 AM	Tidak Perlu Mandi	2.6	0.5	3	1.16	1.16
Desember 22, 2025, 9:25 AM	Desember 21, 2025, 1:31 PM	WAJIB MANDI SEKARANG	19.6	10	9	1.16	1.16
Desember 21, 2025, 1:25 PM	Desember 21, 2025, 12:00 PM	WAJIB MANDI SEKARANG	25.5	10	10	1.16	1.16
Desember 20, 2025, 11:57 AM	Desember 19, 2025, 5:22 PM	WAJIB MANDI SEKARANG	18.6	8.2	9	1.16	1.16
Desember 19, 2025, 5:19 PM	Desember 19, 2025, 11:25 AM	MANDI BISA DITUNDA	5.9	4.7	6	1.16	1.16
Desember 19, 2025, 11:20 AM	Desember 18, 2025, 5:50 PM	SANGAT DISARANKAN	17.5	6	7	1.16	1.16

16 baris

Perubahan Kebutuhan Mandi Dari Waktu Ke Waktu

Generated At: Menit

Distribusi Rekomendasi

Kategori	Persentase
WAJIB MANDI SEKARANG	43.8%
Tidak Perlu Mandi	25.0%
MANDI BISA DITUNDA	18.8%
SANGAT DISARAI	12.5%

16

Riwayat Mandi

Waktu Mandi	Tingkat Kekotoran	Tingkat Bau Badan
Desember 13, 2025, 7:00 AM	4	3
Desember 13, 2025, 6:30 PM	8	7
Desember 14, 2025, 10:30 AM	3	5
Desember 14, 2025, 5:50 PM	6	4
Desember 15, 2025, 12:51 PM	9	7
Desember 16, 2025, 12:32 PM	8	10
Desember 17, 2025, 7:30 AM	6	5

16 baris

