LAPORAN FINAL PROJECT DATA LAKEHOUSE

Oleh :

Hans Christian Cakrawangsa (5026231130)

DEPARTEMEN SISTEM INFORMASI

FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

EXECUTIVE SUMMARY

Dalam laporan ini menyajikan tentang implementasi sistem Data Lakehouse berbasis Python dan PostgreSQL untuk mendukung analisis bisnis perusahaan AdventureWorks secara terintegrasi. Sistem ini dirancang untuk dapat mengelola dan menganalisis data dari berbagai sumber yang tidak terstruktur dan terstruktur, seperti file CSV sensor suhu gudang, PDF laporan market share kompetitor, serta teks sosial media.

Tahapan dalam Pipeline ini mencakup proses Ingest → Analyze → Staging → Structure → Data Warehouse, dengan menekankan :

- Tidak adanya perintah DDl untuk menjamin keamanan struktur database
- Tidak ada proses pembersihan data dalam warehouse
- Penggunaan pendekatan append-only dan penerapan logika SCD (Slowly Changing Dimension) untuk menjaga integritas data historis.

Melalui pendekatan ini, proyek ini berhasil menyediakan kerangka kerja secara analitik yang fleksibel dan siap dikembangkan lebih lanjut dalam arsitektur dengan skala besar berbasis cloud.

1. Latar Belakang

Pada era digital saat ini, data telah menjadi aset strategis yang dapat mendorong pengambilan keputusan yang lebih cepat, tepat, dan berbasis pada fakta. Perusahaan modern, termasuk AdventureWorks, dihadapkan pada tantangan mengelola data dari berbagai sumber, format, dan kecepatan. Mulai dari sensor IoT (suhu gudang), laporan keuangan tahunan, hingga opini publik di media sosial.

Pendekatan tradisional berbasis Data Warehouse (DW) memiliki kekuatan dalam menyimpan data yang telah terstruktur rapi dan terstandarisasi. Namun, DW memiliki keterbatasan ketika harus menangani data tidak terstruktur, seperti file PDF, teks mentah, atau data streaming. Di sisi lain, data lake dapat unggul dalam menampung data dalam format mentah dan berbagai jenis, namun sering kali tidak memiliki struktur yang cukup untuk analisis bisnis strategis.

Untuk mengatasi kekurangan dari kedua pendekatan tersebut, diteraokanlah konsep Data Lakehouse yaitu integrasi antara fleksibilitas data lake dan struktur analitik dari data warehouse. Dalam proyek ini, sistem Data Lakehouse dikembangkan untuk menggabungkan, membersihkan, dan menganalisis data dari berbagai format, yaitu diantaranya :

1. File CSV sebagai hasil pembacaan sensor suhu gudang.
2. Dokumen PDF dapat berupa laporan pangsa pasar tahunan para kompetitor Adventureworks.
3. Data teks dari media sosial (tweet) yang mencerminkan persepsi konsumen.

Data tersebut kemudian diproses melalui pipeline ETL (Extract, Transform, Load) tanpa perintah DDL (Data Definition Language) sehingga tidak mengubah skema data secara destruktif. Data dapat dimuat kedalam stagging area, diolah, dan dimasukkan kedalam skema Star Schema di Data Warehouse, untuk mendukung analisis visual dan strategis.

Dengan membangun arsitektur Data Lakehouse, AdventureWorks diharapkan dapat memperoleh wawasan menyeluruh terhadap kondisi pasar, kinerja gudang, dan persepsi publik secara real time dan historis, sehingga mampu meningkatkan daya saing dan kualitas pengambilan keputusan bisnisnya.

2. Tujuan

Tujuan dari proyek ini adalah untuk merancang dan membangun sistem Data Lakehouse yang mampu mengintegrasikan berbagai jenis data, baik terstruktur maupun tidak terstruktur kedapam satu kerangka kerja analitik yang dapat mendukung pengambilan keputusan bisnis secara strategis di lingkungan perusahaan AdventureWorks.

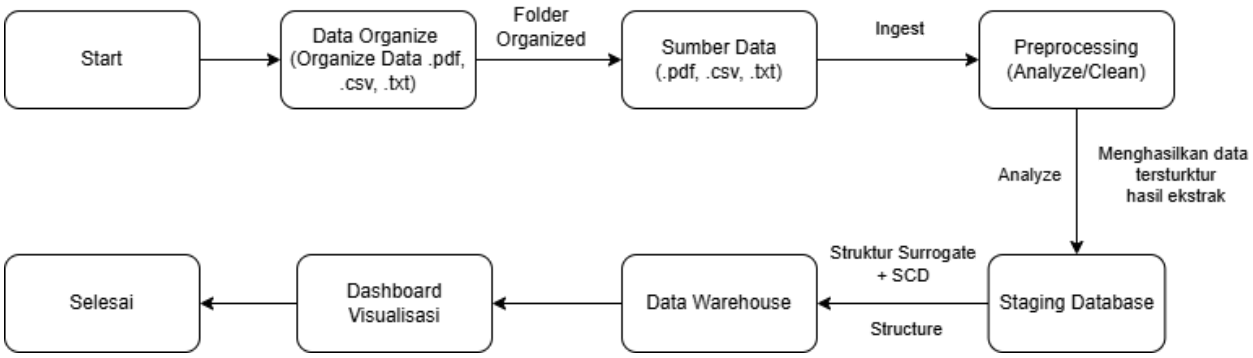
Secara spesifik, proyek ini memiliki sasaran, yaitu :

1. Menyediakan alur ETL yang aman dan non-destruktif
    a. Menjalankan proses ingest, analisis, staging, transformasi, dan pemuatan data ke dalam data warehouse tanpa menggunakan DDL (Data Definition Language) seperti CREATE, DROP, atau TRUNCATE.
    b. Menjamin bahwa tidak ada data warehouse yang dibersihkan, tetapi hanya bersifat append atau update berdasarkan logika dimensi historis (SCD).
2. Menggabungkan berbagai format sumber data
    a. Mengelola data CSV dari sensor suhu gudang.
    b. Mengekstrak informasi dari file PDF berupa laporan tahunan.
    c. Memanfaatkan teks sosial media dalam bentuk tweet.
3. Membangun skema data warehouse yang efisien
    a. Mendesain Star Schema yang mencakup tabel fakta dan dimensi
    b. Menyediakan basis data analitik yang dapat diakses secara mudah untuk keperluan visualisasi dan pelaporan.
4. Menerapkan prinsip-prinsip Data Lakehouse modern
    a. Mengintegrasikan data lake untuk ingest berbagai jenis data dan kekuatan data warehouse unutk analisis OLAP.
    b. Menerapkan pendekaran scalabel, modular, dan reproducible untuk pipeline data, sesuai dengan kebutuhan dunia nyata yang terus berkembang.

3. Arsitektur Sistem

Sistem Data Lakehouse yang dikembangkan dalam proyek ini dirancang untuk dapat memproses, mengintegrasikan, dan menganalisis data dari berbagai jenis sumber, dengan tetap menjaga struktur data warehouse untuk keperluan analitik. Arsitektur ini menggabungkan pendekatan Data Lake yang fleksibel dan Data Warehouse yang lebih terstruktur dalam satu alur pipeline yang utuh dan efisien.

Berikut adalah tahapan utama dalam arsitektur sistem :



1. Sumber data :
    a. Warehouse_temp_sensor.csv = data suhu gudang
    b. Market_share_report = laporan market share kompetitor
    c. Adventureworks_structured_150_tweets.txt = data tweet untuk wordcloud.
2. Staging Area :
    a. Tempat penyimpanan sementara hasil parsing dan pembersihan data.
    b. Tidak ada struktur DDL dibuat disini
3. Data Warehouse:
    a. Menggunakan Star Schema :
        i. Tabel fakta : fact_competitor_share
        ii. Dimensi : dim_competitor, dim_time
    b. Pemuatan data dilakukan dengan append (tanpa replace/truncate)
4. ETL Processing (structure.py):
    a. Menangani logika transofrmasi, mapping surrogate key, dan SCD Type 1

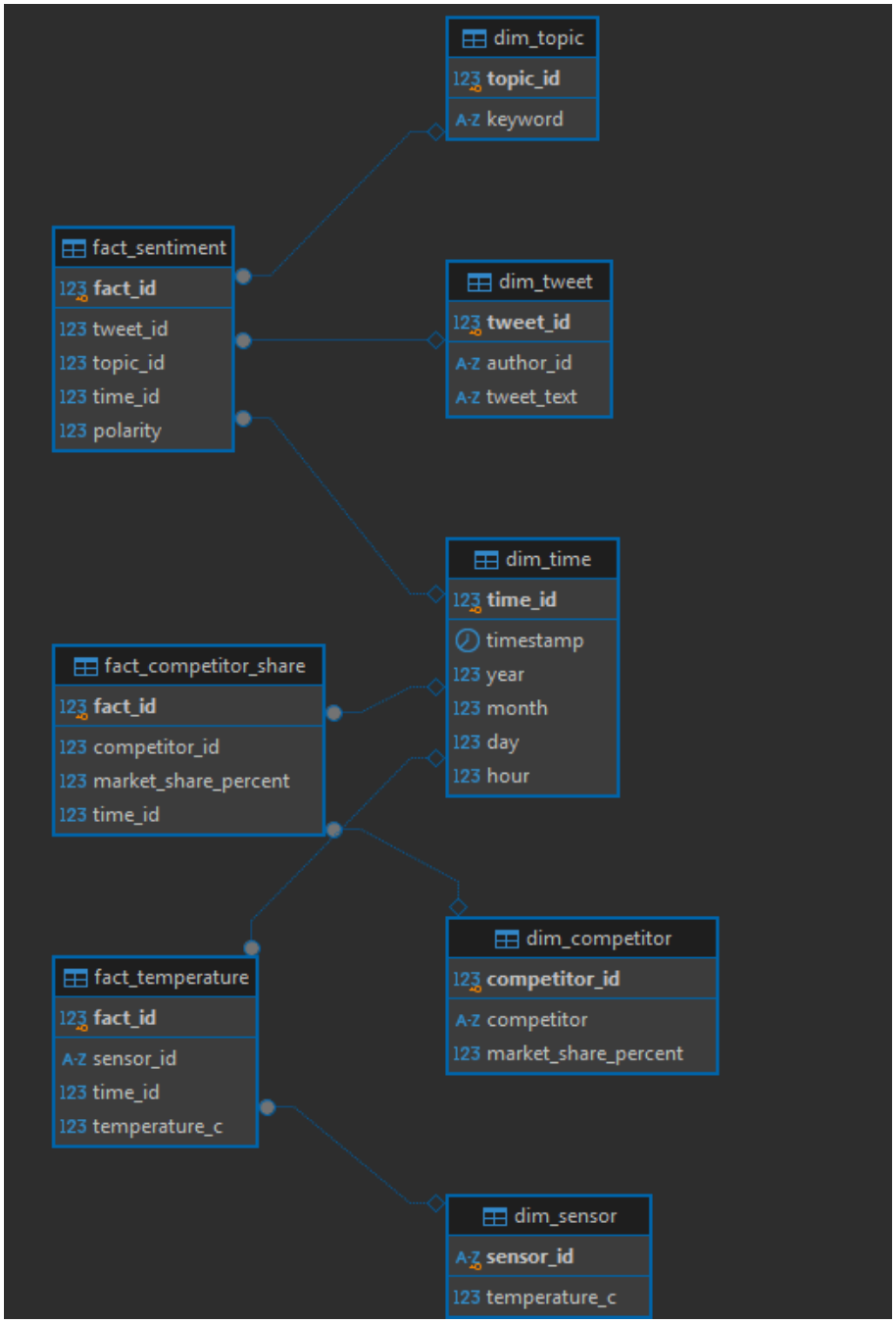
    b. Tidak ada CREATE, DROP, atau TRUNCATE

5. Visualisasi (Dashboard)
    a. Menggunakan matplotlib untuk :
        i. Grafik horizontal market share.
        ii. Tren suhu gudang.
        iii. Word cloud opini publik.

4. Desain Schema

Untuk mendukung kebutuhan analitik, sistem Data Lakehouse ini menggunakan pendekatan Star Schema yang umum digunakan dalam data warehouse. Skema ini dapat memungkinkan integrasi data secara efisien serta mempercepat proses query dalam analisis OLAP (Online Analytical Processing).

Desain Struktur Star Schema yaitu :



Tabel-tabel utama dalam Star Schema :

A. Tabel Dimensi

a. `dim_topic`

Berisi data topik / keyword dan topic_id untuk mengklasifikasi tweet.

b. `dim_tweet`

Untuk menyimpan metadata dan informasi dari tweet : tweet_id, author_id, tweet_text.

c. `dim_time`

Untuk menyimpan data waktu dari masing-masing data dari .pdf, csv, dan .txt. Kolom terdiri dari `timestamp`, `year`, `month`, `day`, `hour`

d. `dim_competitor`

Untuk menyimpan info kompetitor dan `market_share_percent` terakhir (bisa diduplikasi juga di fact untuk historisasi)

e. `dim_sensor`

Untuk menyimpan data sensor temperatur gudang dalam bentuk `sensor_id` dan `temperature_c`

B. Tabel Fact

a. `fact_sentiment`

Untuk mencatat hasil analisis sentiment dari tweet terhadap topik tertentu dalam waktu tertentu.

Relasi

i. `tweet_id` → `dim_tweet`

ii. `topic_id` → `dim_topic`

iii. `time_id` → `dim_time`

iv. `polarity`

b. fact_competitor_share

Untuk mencatat pangsa pasar kompetitor berdasarkan waktu.

Relasi

i. competitor_id → dim_competitor

ii. market_share_percent → dim_competitor

iii. time_id → dim_time

c. fact_temperature

Untuk mencatat pangsa pasar kompetitor berdasarkan waktu.

Relasi

i. sensor_id → dim_sensor

ii. temperature_c → dim_sensor

iii. time_id → dim_time

5. ETL & SCD Logic

1. Ingest

Lakukan Ingest dan Analyze untuk melakukan ekstraksi data dari .csv, .pdf, dan txt.

```
# INGEST: Ambil & Simpan Data Mentah dari CSV, PDF, TXT (Pure
Ingestion)

import os
import shutil
import time
import concurrent.futures
from pathlib import Path
from tqdm import tqdm
import pandas as pd
import fitz  # PyMuPDF

def copy_file(src: Path, dst: Path) -> bool:
    """Copy file with error handling and progress
tracking."""
```

```
    try:
        dst.parent.mkdir(parents=True, exist_ok=True)
        shutil.copy2(src, dst)
        return True
    except Exception as e:
        print(f"Error copying {src}: {e}")
        return False

def extract_pdf_text(pdf_path: Path, txt_path: Path) -> bool:
    """Extract text from PDF and save to text file."""
    try:
        text_parts = []
        with fitz.open(pdf_path) as doc:
            for page in tqdm(doc, desc="Extracting PDF
pages", unit="page"):
                text_parts.append(page.get_text())

        txt_path.write_text('\n'.join(text_parts),
encoding='utf-8')
        return True
    except Exception as e:
        print(f"Error processing PDF {pdf_path}: {e}")
        return False

def extract_csv_text(csv_path: Path, txt_path: Path) -> bool:
    """Extract text from CSV and save to text file with
progress bar."""
    try:
        print(f"\nProcessing CSV {csv_path.name}...")
        # Read CSV in chunks with progress bar
        chunk_size = 1000  # Process 1000 rows at a time
        chunks = []
        with tqdm(desc="Reading CSV chunks", unit="rows") as
pbar:
            for chunk in pd.read_csv(csv_path,
chunksize=chunk_size):
                chunks.append(chunk)
                pbar.update(len(chunk))
```

```
        # Combine chunks
        df = pd.concat(chunks, ignore_index=True)

        # Convert to text
        text = df.to_csv(index=False)

        # Write with progress bar
        with tqdm(total=len(text), desc="Writing CSV text",
unit="chars") as pbar:
            with txt_path.open('w', encoding='utf-8') as f:
                for i in range(0, len(text), 10000):  # Write
in chunks of 10000 chars
                    f.write(text[i:i+10000])
                    pbar.update(10000)

        return True
    except Exception as e:
        print(f"Error processing CSV {csv_path}: {e}")
        return False

def extract_txt_text(txt_path: Path, txt_path_out: Path) ->
bool:
    """Copy TXT file to new location with _raw suffix and
progress bar."""
    try:
        print(f"\nProcessing TXT {txt_path.name}...")

        # Read file size
        file_size = txt_path.stat().st_size

        # Read and write with progress bar
        with txt_path.open('r', encoding='utf-8') as src, \
             txt_path_out.open('w', encoding='utf-8') as dst,
\
             tqdm(total=file_size, desc="Copying TXT file",
unit="B", unit_scale=True) as pbar:

            chunk_size = 1024  # 1KB chunks
            while True:
```

```
                chunk = src.read(chunk_size)
                if not chunk:
                    break
                dst.write(chunk)
                pbar.update(len(chunk))

        return True
    except Exception as e:
        print(f"Error processing TXT {txt_path}: {e}")
        return False

def main():
    start_time = time.time()
    print(" Starting data ingestion process...")

    # Get the data_lake directory path
    DATA_LAKE_DIR = Path(__file__).parent  # This file is in
data_lake directory

    # Define source and destination paths relative to
DATA_LAKE_DIR
    operations = [
        # (source, destination, is_pdf)

("data_lake/data_lake/adventureworks/organized/warehouse_temp
_sensor.csv",
"adventureworks/organized/warehouse_temp_sensor.csv", False),

("data_lake/data_lake/adventureworks/organized/market_share_r
eport.pdf",
"adventureworks/organized/market_share_report.pdf", True),

("data_lake/data_lake/adventureworks/organized/adventureworks
_structured_150_tweets.txt",
"adventureworks/organized/adventureworks_structured_150_tweet
s.txt", False),
    ]

    # Process files in parallel
```

```
    with concurrent.futures.ThreadPoolExecutor() as executor:
        futures = []
        for src_rel, dst_rel, is_pdf in operations:
            src = DATA_LAKE_DIR.parent / src_rel  # Go up one
level to account for the nested data_lake directory
            dst = DATA_LAKE_DIR / dst_rel

            # Ensure source file exists
            if not src.exists():
                print(f"Warning: Source file not found:
{src}")
                continue

            # Submit copy operation
            future = executor.submit(copy_file, src, dst)
            futures.append((future, dst, is_pdf))

        # Process results and handle file extraction
        for future, dst, is_pdf in futures:
            if not future.result():
                continue

            # Create raw text version in the same directory
            txt_path = dst.parent / f"{dst.stem}_raw.txt"
            if dst.suffix.lower() == '.pdf':
                extract_pdf_text(dst, txt_path)
            elif dst.suffix.lower() == '.csv':
                extract_csv_text(dst, txt_path)
            elif dst.suffix.lower() == '.txt':
                extract_txt_text(dst, txt_path)

    total_time = time.time() - start_time
    print(f"\n Semua file berhasil diproses dalam
{total_time:.2f} detik")

if __name__ == "__main__":
    main()
```

Pada proses ini menyangkup proses :

1. Mengambil dan menyimpan data mentah
2. Melakukan ekstraksi data dari .csv, .pdf, dan .txt

2. Analyze

```
# ✅ ANALYZE: Membersihkan & Menstrukturkan Data Mentah ke
Staging Database

"""
Tujuan:
- Parsing isi file (PDF → Teks → DataFrame)
- Validasi tipe data dan handling missing values
- Cleaning teks, normalisasi data (contoh: lowercase, hapus
karakter khusus)
- Deteksi entitas penting (contoh: nama perusahaan, sentimen)
- Simpan hasil analisis ke database staging
"""

import pandas as pd
import fitz  # PyMuPDF
from sqlalchemy import create_engine, exc
import re
from pathlib import Path
from datetime import datetime
import logging

# Konfigurasi logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s -
%(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# ---------- KONFIGURASI PATH & DATABASE ---------- #
BASE_DIR = Path(__file__).parent
DATA_DIR = BASE_DIR / "data_lake" / "adventureworks" /
"organized"
OUTPUT_DIR = BASE_DIR / "data_lake" / "adventureworks" /
"processed"
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)
```

```python
# Konfigurasi database
DB_CONFIG = {
    'dbname': 'staging',
    'user': 'postgres',
    'password': 'chriscakra15',
    'host': 'localhost',
    'port': '5432'
}

def get_database_connection():
    """Membuat koneksi ke database"""
    try:
        conn_str =
f"postgresql://{DB_CONFIG['user']}:{DB_CONFIG['password']}@{D
B_CONFIG['host']}:{DB_CONFIG['port']}/{DB_CONFIG['dbname']}"
        return create_engine(conn_str)
    except Exception as e:
        logger.error(f"Gagal terhubung ke database: {e}")
        raise

def process_csv_file(csv_path):
    """Memproses file CSV"""
    try:
        logger.info(f"Memproses file CSV: {csv_path}")
        df = pd.read_csv(csv_path)

        # Handle warehouse temperature sensor data
        if 'warehouse_temp_sensor' in str(csv_path):
            # Pastikan kolom yang diperlukan ada
            required = ['timestamp', 'sensor_id',
'temperature_c']
            missing = [col for col in required if col not in
df.columns]
            if missing:
                logger.error(f"Kolom yang diperlukan tidak
ditemukan: {missing}")
                return None, None

            # Pastikan tipe data sesuai
```

```
            df['timestamp'] = pd.to_datetime(df['timestamp'],
errors='coerce')
            df['temperature_c'] =
pd.to_numeric(df['temperature_c'], errors='coerce')

            # Hapus baris dengan data yang tidak valid
            df = df.dropna(subset=['timestamp', 'sensor_id',
'temperature_c'])

        # Simpan ke CSV yang sudah diproses
        output_path = OUTPUT_DIR /
f"processed_{csv_path.name}"
        df.to_csv(output_path, index=False)

        return df, output_path
    except Exception as e:
        logger.error(f"Gagal memproses file CSV {csv_path}:
{e}")
        return None, None

def process_txt_file(txt_path):
    """Memproses file teks (tweet data)"""
    try:
        logger.info(f"Memproses file TXT: {txt_path}")

        # Baca dan parse data tweet
        df = pd.read_csv(
            txt_path,
            sep='\t',
            header=None,
            names=[
                'tweet_id',
                'tweet_text',
                'timestamp',
                'user_location',
                'sentiment',
                'matched_product'
            ]
        )
```

```
        # Validasi kolom yang diperlukan
        required = ['tweet_id', 'tweet_text', 'timestamp',
'user_location', 'sentiment', 'matched_product']
        missing = [col for col in required if col not in
df.columns]
        if missing:
            logger.error(f"Kolom yang diperlukan tidak
ditemukan: {missing}")
            return None, None

        # Cleaning dan validasi data
        df['timestamp'] = pd.to_datetime(df['timestamp'],
errors='coerce')
        df['tweet_text'] = df['tweet_text'].str.strip()
        df['user_location'] =
df['user_location'].fillna('Unknown').str.strip()
        df['sentiment'] =
df['sentiment'].str.lower().str.strip()
        df['matched_product'] =
df['matched_product'].fillna('Unknown').str.strip()

        # Hapus baris dengan data yang tidak valid
        df = df.dropna(subset=['tweet_id', 'tweet_text',
'timestamp', 'sentiment'])

        # Simpan ke CSV yang sudah diproses
        output_path = OUTPUT_DIR /
f"tweet_analysis_{datetime.now().strftime('%Y%m%d')}.csv"
        df.to_csv(output_path, index=False)

        return df, output_path
    except Exception as e:
        logger.error(f"Gagal memproses file TXT {txt_path}:
{e}")
        return None, None

def process_market_share_pdf(pdf_path):
    """
```

```
    Memproses file PDF laporan market share dan mengekstrak
data:
    - competitor
    - market_share_percent
    - time_period
    - extraction_date
    """
    try:
        logger.info(f"Memproses laporan market share:
{pdf_path}")

        def extract_table_from_pdf(path):
            """Mengekstrak tabel dari PDF"""
            with fitz.open(path) as doc:
                all_tables = []
                for page in doc:
                    # Dapatkan tabel dari halaman
                    tables = page.find_tables()
                    if tables.tables:
                        all_tables.extend([table.extract()
for table in tables])
            return all_tables

        def find_market_share_table(tables):
            """Mencari tabel yang berisi data market share"""
            for table in tables:
                # Cari header yang sesuai
                headers = [str(cell).lower() for row in
table[:1] for cell in row]
                header_text = ' '.join(headers)

                # Cari header yang mengandung kata kunci
                if any(keyword in header_text for keyword in
['time periode','competitor','market share']):
                    return table
            return None

        def parse_table_data(table):
            """Memproses data tabel menjadi DataFrame"""
```

```
            # Dapatkan header dan cari indeks kolom yang
dibutuhkan
            headers = [str(cell).lower().strip() for cell in
table[0]]

            # Cari indeks kolom yang sesuai
            period_col = next((i for i, h in
enumerate(headers) if 'period' in h or 'time' in h), None)
            comp_col = next((i for i, h in enumerate(headers)
if 'competitor' in h or 'company' in h), None)
            share_col = next((i for i, h in
enumerate(headers) if 'market' in h and 'share' in h), None)

            if None in (period_col, comp_col, share_col):
                return pd.DataFrame()

            # Proses setiap baris data
            data = []
            for row in table[1:]:  # Lewati header
                if len(row) > max(period_col, comp_col,
share_col):
                    try:
                        period = str(row[period_col]).strip()
                        competitor =
str(row[comp_col]).strip()
                        market_share =
float(str(row[share_col]).replace('%', '').strip())

                        if competitor and market_share > 0:
                            data.append({
                                'time_periode': period,
                                'competitor': competitor,
                                'market_share_percent':
market_share
                            })
                    except (ValueError, IndexError):
                        continue

            return pd.DataFrame(data)
```

```
        # Ekstrak dan proses tabel
        tables = extract_table_from_pdf(pdf_path)
        if not tables:
            logger.warning("Tidak ada tabel yang ditemukan
dalam PDF")
            return None, None

        market_share_table = find_market_share_table(tables)
        if market_share_table is None:
            logger.warning("Tabel market share tidak
ditemukan dalam PDF")
            return None, None

        df = parse_table_data(market_share_table)

        if df.empty:
            logger.warning("Tidak ada data market share yang
berhasil diekstraksi dari tabel")
            return None, None

        # Pastikan kolom yang diperlukan ada
        required_columns = ['competitor',
'market_share_percent', 'time_periode']
        for col in required_columns:
            if col not in df.columns:
                logger.error(f"Kolom {col} tidak ditemukan
dalam data yang diekstrak")
                return None, None

        # Tambahkan extraction_date
        df['extraction_date'] =
datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        # Simpan ke CSV
        output_path = OUTPUT_DIR /
f"market_share_report_{datetime.now().strftime('%Y%m%d_%H%M%S
')}.csv"
        df.to_csv(output_path, index=False)
```

```
        logger.info(f"Data market share berhasil disimpan ke:
{output_path}")

        return df, output_path

    except Exception as e:
        logger.error(f"Gagal memproses laporan market share:
{e}", exc_info=True)
        return None, None

def load_to_database(df, table_name):
    """Memuat data ke database staging"""
    try:
        engine = get_database_connection()

        # Pastikan kolom yang diperlukan ada
        required_columns = {
            'staging_market_share_report': ['time_periode',
'competitor', 'market_share_percent', 'extraction_date'],
            'staging_warehouse_temp_sensor': ['timestamp',
'sensor_id', 'temperature_c'],
            'staging_external_sentiment': ['tweet_id',
'tweet_text', 'timestamp', 'user_location', 'sentiment',
'matched_product']
        }

        # Validasi kolom yang diperlukan
        if table_name in required_columns:
            # Tambahkan extraction_date untuk market share
report jika belum ada
            if table_name == 'staging_market_share_report'
and 'extraction_date' not in df.columns:
                df['extraction_date'] =
datetime.now().strftime('%Y-%m-%d %H:%M:%S')

            missing_cols = [col for col in
required_columns[table_name] if col not in df.columns]
            if missing_cols:
                logger.error(f"Kolom yang diperlukan tidak
```

```
ditemukan: {missing_cols}")
                return False

        # Pastikan tipe data sesuai
        if 'market_share_percent' in df.columns:
            df['market_share_percent'] =
pd.to_numeric(df['market_share_percent'], errors='coerce')
            df = df.dropna(subset=['market_share_percent'])

        # Load ke database
        with engine.connect() as conn:
            # Gunakan if_exists='append' untuk menambahkan
data ke tabel yang sudah ada
            df.to_sql(
                name=table_name,
                con=conn,
                if_exists='append',
                index=False,
                method='multi',
                chunksize=1000
            )

            # Commit transaksi
            conn.commit()

        logger.info(f"Berhasil memuat {len(df)} baris ke
tabel {table_name}")
        return True

    except Exception as e:
        logger.error(f"Gagal memuat data ke database: {e}",
exc_info=True)
        if 'conn' in locals():
            conn.rollback()
        return False

def main():
    try:
        logger.info("Memulai proses analisis data...")
```

```
        # Pastikan direktori data ada
        if not DATA_DIR.exists():
            logger.error(f"Direktori data tidak ditemukan:
{DATA_DIR}")
            return

        # Proses file sensor suhu gudang
        sensor_file = DATA_DIR / "warehouse_temp_sensor.csv"
        if sensor_file.exists():
            try:
                logger.info(f"Memproses file sensor:
{sensor_file}")
                df_sensor, _ = process_csv_file(sensor_file)

                # Load to database
                if load_to_database(df_sensor,
"staging_warehouse_temp_sensor"):
                    logger.info("✅ Data sensor berhasil
dimuat ke staging_warehouse_temp_sensor")
                else:
                    logger.error("❌ Gagal memuat data sensor
ke database")
            except Exception as e:
                logger.error(f"Gagal memproses file sensor:
{e}", exc_info=True)
        else:
            logger.warning(f"File sensor tidak ditemukan:
{sensor_file}")

        # Proses file tweet
        tweet_file = DATA_DIR /
"adventureworks_structured_150_tweets.txt"
        if tweet_file.exists():
            try:
                logger.info(f"Memproses file tweet:
{tweet_file}")
                df_tweet, _ = process_txt_file(tweet_file)
```

```
                # Load to database
                if load_to_database(df_tweet,
"staging_external_sentiment"):
                    logger.info("✅ Data tweet berhasil
dimuat ke staging_external_sentiment")
                else:
                    logger.error("❌ Gagal memuat data tweet
ke database")
            except Exception as e:
                logger.error(f"Gagal memproses file tweet:
{e}", exc_info=True)
        else:
            logger.warning(f"File tweet tidak ditemukan:
{tweet_file}")

        # Proses file market share PDF
        market_share_file = DATA_DIR /
"market_share_report.pdf"
        if market_share_file.exists():
            try:
                logger.info(f"Memproses file market share:
{market_share_file}")
                df_market_share, _ =
process_market_share_pdf(market_share_file)

                # Load to database
                if df_market_share is not None and
load_to_database(df_market_share,
"staging_market_share_report"):
                    logger.info("✅ Data market share
berhasil dimuat ke staging_market_share_report")
                else:
                    logger.error("❌ Gagal memuat data market
share ke database")
            except Exception as e:
                logger.error(f"Gagal memproses file market
share: {e}", exc_info=True)
        else:
            logger.warning(f"File market share tidak
```

```
ditemukan: {market_share_file}")

        logger.info("✅ Proses analisis data selesai")

    except Exception as e:
        logger.error(f"Terjadi kesalahan dalam proses
analisis: {e}", exc_info=True)
        raise

if __name__ == "__main__":
    main()
```

Pada proses ini mencakup proses :

1. Preprocessing data untuk cleaning dan parsing
2. Data dimasukkan ke dalam tabel staging yaitu terdapat : staging_external_sentiment, staging_market_share_report, staging_warehouse_temp_sensor.
3. Memasukkan data baru hasil ekstraksi dalam bentuk CSV dan menyimpannya dalam folder processed.

3. Structure

```
# Struktur ETL Data Warehouse (Sensor + Tweet + Competitor)
import pandas as pd
from sqlalchemy import create_engine
from datetime import datetime
from sqlalchemy import text
import tkinter as tk
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import
FigureCanvasTkAgg
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

# Koneksi database - sebaiknya dipindahkan ke file
konfigurasi terpisah
STAGGING_DB =
```

```
'postgresql://postgres:chriscakra15@localhost:5432/staging'
DWH_DB =
'postgresql://postgres:chriscakra15@localhost:5432/Adventurew
orksDW'
engine_stag = create_engine(STAGGING_DB)
engine_dwh = create_engine(DWH_DB)

def check_table_exists_and_has_data(engine, table_name,
schema='dwh'):
    """Memeriksa apakah tabel ada dan memiliki data"""
    try:
        df = pd.read_sql(f"SELECT * FROM
{schema}.{table_name} LIMIT 1", con=engine)
        if df.empty:
            print(f"WARNING: Tabel {table_name} kosong!")
            return False
        return True
    except:
        print(f"WARNING: Tabel {table_name} tidak
ditemukan!")
        return False

def check_staging_data():
    """Memeriksa ketersediaan data di staging database"""
    print("\nMemeriksa data di staging database...")
    tables = ['staging_warehouse_temp_sensor',
'staging_market_share_report', 'staging_external_sentiment']
    for table in tables:
        try:
            df = pd.read_sql(f"SELECT * FROM {table} LIMIT
1", con=engine_stag)
            if df.empty:
                print(f"WARNING: Tabel staging {table}
kosong!")
            else:
                print(f"Data tersedia: Tabel staging {table}
memiliki data")
        except:
            print(f"WARNING: Tabel staging {table} tidak
```

```
ditemukan!")

def check_warehouse_data():
    """Memeriksa ketersediaan data di data warehouse"""
    print("\nMemeriksa data di warehouse...")
    tables = ['dim_sensor', 'dim_time', 'dim_tweet',
'dim_competitor', 'fact_temperature', 'fact_sentiment',
'fact_competitor_share']
    for table in tables:
        check_table_exists_and_has_data(engine_dwh, table)

def load_dim_competitor():
    """Memuat data competitor dari
staging_market_share_report ke dim_competitor"""
    try:
        print("\nMemuat data ke dim_competitor...")

        # Query untuk mengambil data unik dari staging
        query = """
        SELECT DISTINCT
            competitor,
            market_share_percent
        FROM staging_market_share_report
        WHERE competitor IS NOT NULL
        """

        # Baca data dari staging
        df = pd.read_sql(query, con=engine_stag)

        if df.empty:
            print("Tidak ada data competitor di staging.")
            return

        # Tambahkan kolom competitor_id
        df['competitor_id'] = range(1, len(df) + 1)

        # Pilih dan urutkan kolom sesuai dengan struktur
dim_competitor
        df = df[['competitor_id', 'competitor',
```

```
'market_share_percent']]

        # Simpan ke dim_competitor
        df.to_sql(
            'dim_competitor',
            con=engine_dwh,
            schema='dwh',
            if_exists='append',
            index=False
        )

        print(f"Berhasil memuat {len(df)} data competitor ke
dim_competitor")

    except Exception as e:
        print(f"Gagal memuat data ke dim_competitor:
{str(e)}")
        raise

def load_fact_competitor_share():
    """Memuat data market share competitor ke
fact_competitor_share"""
    try:
        print("\nMemuat data ke fact_competitor_share...")

        # Ambil staging dari DB staging
        df_staging = pd.read_sql("SELECT * FROM
staging_market_share_report", con=engine_stag)
        if df_staging.empty:
            print("❗ Tidak ada data di
staging_market_share_report.")
            return

        # Ambil dimensi dari DWH
        df_comp = pd.read_sql("SELECT competitor_id,
competitor FROM dwh.dim_competitor", con=engine_dwh)
        df_time = pd.read_sql("SELECT time_id,
timestamp::date AS extraction_date FROM dwh.dim_time",
con=engine_dwh)
```

```
        # Gabungkan
        df_merged = df_staging.merge(df_comp,
on='competitor', how='inner')
        df_merged = df_merged.merge(df_time,
on='extraction_date', how='inner')

        # Ambil kolom yang diperlukan
        df_final = df_merged[['competitor_id',
'market_share_percent', 'time_id']]

        if df_final.empty:
            print("❗ Data setelah merge tidak ditemukan.
Pastikan competitor dan extraction_date cocok.")
            return

        # Simpan ke DWH
        df_final.to_sql(
            'fact_competitor_share',
            con=engine_dwh,
            schema='dwh',
            if_exists='append',
            index=False
        )

        print(f"✅ Berhasil menambahkan {len(df_final)} data
ke fact_competitor_share.")

    except Exception as e:
        print(f"❌ Gagal memuat data ke
fact_competitor_share: {str(e)}")
        raise




def load_dim_tweet():
    """Memuat data tweet ke dalam dimensi"""
    print("\nMemuat data ke dim_tweet...")
```

```
    df = pd.read_sql("SELECT DISTINCT tweet_id, tweet_text
FROM staging_external_sentiment", con=engine_stag)
    df['author_id'] = 'unknown'
    df = df[['tweet_id', 'author_id', 'tweet_text']]

    existing = pd.read_sql("SELECT tweet_id FROM
dwh.dim_tweet", con=engine_dwh)
    df = df[~df['tweet_id'].isin(existing['tweet_id'])]

    if df.empty:
        print("Tidak ada tweet baru.")
        return

    df.to_sql('dim_tweet', con=engine_dwh, schema='dwh',
if_exists='append', index=False)
    print(f"{len(df)} tweet berhasil dimasukkan.")

def load_dim_topic():
    """Memuat data topik tweet"""
    print("\nMemuat data ke dim_topic...")
    df = pd.read_sql("SELECT DISTINCT matched_product FROM
staging_external_sentiment", con=engine_stag)
    df['keyword'] = df['matched_product']
    df = df[['keyword']].dropna().drop_duplicates()

    existing = pd.read_sql("SELECT keyword FROM
dwh.dim_topic", con=engine_dwh)
    df = df[~df['keyword'].isin(existing['keyword'])]

    if df.empty:
        print("Tidak ada topik baru.")
        return

    df.to_sql('dim_topic', con=engine_dwh, schema='dwh',
if_exists='append', index=False)
    print(f"{len(df)} topik dimasukkan.")

def load_fact_sentiment():
    """Memuat data sentimen ke dalam fact table"""
```

```
    try:
        print("\nMemuat data ke fact_sentiment...")

        # Create fact_sentiment table if it doesn't exist
        create_table_sql = """
        CREATE TABLE IF NOT EXISTS dwh.fact_sentiment (
            fact_id SERIAL PRIMARY KEY,
            tweet_id VARCHAR(50) NOT NULL,
            topic_id INT,
            time_id INT,
            polarity INT,
            FOREIGN KEY (tweet_id) REFERENCES
dwh.dim_tweet(tweet_id),
            FOREIGN KEY (topic_id) REFERENCES
dwh.dim_topic(topic_id),
            FOREIGN KEY (time_id) REFERENCES
dwh.dim_time(time_id)
        )
        """
        with engine_dwh.connect() as connection:
            connection.execute(text(create_table_sql))
            connection.commit()

        # Get sentiment data from staging
        df = pd.read_sql("""
            SELECT tweet_id, sentiment, timestamp,
matched_product
            FROM staging_external_sentiment
        """, con=engine_stag)

        if df.empty:
            print("Tidak ada data sentimen baru yang
ditemukan di staging.")
            return

        # Map sentiment to polarity
        df['polarity'] = df['sentiment'].map({'positive': 1,
'negative': -1, 'neutral': 0}).fillna(0)
```

```
        # Get dimension mappings
        tweet_map = pd.read_sql("SELECT tweet_id FROM
dwh.dim_tweet", con=engine_dwh)
        topic_map = pd.read_sql("SELECT topic_id, keyword
FROM dwh.dim_topic", con=engine_dwh)

        # Get time_id from dim_time
        time_map = pd.read_sql("""
            SELECT time_id, timestamp::date as date
            FROM dwh.dim_time
        """, con=engine_dwh)

        # Filter only tweets that exist in dim_tweet
        df = df[df['tweet_id'].isin(tweet_map['tweet_id'])]

        # Merge with topic and time dimensions
        df = df.merge(topic_map, left_on='matched_product',
right_on='keyword', how='left')
        df['date'] = pd.to_datetime(df['timestamp']).dt.date
        df = df.merge(time_map, on='date', how='left')

        # Select and rename columns to match the fact table
        fact_data = df[['tweet_id', 'topic_id', 'time_id',
'polarity']].dropna()

        if not fact_data.empty:
            # Insert only new records
            existing_records = pd.read_sql(
                "SELECT tweet_id, topic_id, time_id FROM
dwh.fact_sentiment",
                con=engine_dwh
            )

            if not existing_records.empty:
                # Create a composite key for comparison
                existing_records['composite_key'] =
existing_records.astype(str).apply('_'.join, axis=1)
                fact_data['composite_key'] =
fact_data.astype(str).apply('_'.join, axis=1)
```

```
                fact_data =
fact_data[~fact_data['composite_key'].isin(existing_records['
composite_key'])]
                fact_data = fact_data.drop('composite_key',
axis=1)

            if not fact_data.empty:
                # Insert new records
                fact_data.to_sql(
                    'fact_sentiment',
                    con=engine_dwh,
                    schema='dwh',
                    if_exists='append',
                    index=False
                )
                print(f"Berhasil menambahkan {len(fact_data)}
data sentimen baru")
            else:
                print("Tidak ada data sentimen baru yang
perlu ditambahkan")
        else:
            print("Tidak ada data yang memenuhi syarat untuk
dimasukkan ke fact_sentiment")

    except Exception as e:
        print(f"Error saat memuat data ke fact_sentiment:
{str(e)}")
        raise

def load_fact_temperature():
    """Memuat data suhu ke dalam fact table"""
    try:
        print("\nMemulai proses load data ke
fact_temperature...")

        # Hanya ambil data yang belum ada di fact_temperature
        query = """
        SELECT s.sensor_id, s.temperature, s.timestamp
        FROM staging_warehouse_temp_sensor s
```

```
        LEFT JOIN dwh.fact_temperature ft ON s.sensor_id =
ft.sensor_id
            AND s.timestamp = ft.timestamp
        WHERE ft.sensor_id IS NULL
        """
        df = pd.read_sql(query, con=engine_stag)

        if df.empty:
            print("Tidak ada data suhu baru yang ditemukan.")
            return

        # Dapatkan time_id untuk setiap timestamp
        df['date'] = pd.to_datetime(df['timestamp']).dt.date
        time_map = pd.read_sql("SELECT time_id, date FROM
dwh.dim_time", con=engine_dwh)
        df = df.merge(time_map, on='date', how='left')

        # Siapkan data untuk dimasukkan
        fact_data = df[['sensor_id', 'temperature',
'timestamp', 'time_id']]

        fact_data.to_sql('fact_temperature', con=engine_dwh,
schema='dwh',
                         if_exists='append', index=False)

        print(f"Berhasil memuat {len(fact_data)} data suhu
baru ke fact_temperature")

    except Exception as e:
        print(f"Error saat memuat data ke fact_temperature:
{str(e)}")

def load_fact_competitor():
    """Memuat data competitor dari
staging_market_share_report ke fact_competitor"""
    try:
        print("\n=== Memulai pemuatan data fact_competitor
===")
```

```
        # Pastikan tabel fact_competitor ada
        create_table_sql = """
        CREATE TABLE IF NOT EXISTS dwh.fact_competitor (
            fact_id SERIAL PRIMARY KEY,
            competitor_id INTEGER NOT NULL,
            time_id INTEGER NOT NULL,
            market_share_percent NUMERIC(10,2) NOT NULL,
            extraction_date TIMESTAMP,
            FOREIGN KEY (competitor_id) REFERENCES
dwh.dim_competitor(competitor_id),
            FOREIGN KEY (time_id) REFERENCES
dwh.dim_time(time_id),
            UNIQUE(competitor_id, time_id)
        )
        """

        with engine_dwh.connect() as connection:
            connection.execute(text(create_table_sql))
            connection.commit()
        print("Tabel fact_competitor siap digunakan.")

        # Debug: Hitung jumlah data di setiap tabel
        with engine_stag.connect() as conn:
            staging_count = conn.execute(text("SELECT
COUNT(*) FROM staging_market_share_report")).scalar()
            print(f"Jumlah data di
staging_market_share_report: {staging_count}")

        with engine_dwh.connect() as conn:
            comp_count = conn.execute(text("SELECT COUNT(*)
FROM dwh.dim_competitor")).scalar()
            time_count = conn.execute(text("SELECT COUNT(*)
FROM dwh.dim_time")).scalar()
            print(f"Jumlah data di dim_competitor:
{comp_count}")
            print(f"Jumlah data di dim_time: {time_count}")

        # Query untuk mengambil data dari staging dan
menggabungkannya dengan dimensi
```

```
        query = """
        WITH competitor_data AS (
            SELECT
                dc.competitor_id,
                smsr.market_share_percent,
                dt.time_id,
                smsr.extraction_date
            FROM staging_market_share_report smsr
            JOIN dwh.dim_competitor dc ON
TRIM(LOWER(smsr.competitor)) = TRIM(LOWER(dc.competitor))
            JOIN dwh.dim_time dt ON smsr.time_periode =
dt.time_periode
            WHERE NOT EXISTS (
                SELECT 1
                FROM dwh.fact_competitor fc
                WHERE fc.competitor_id = dc.competitor_id
                AND fc.time_id = dt.time_id
            )
        )
        SELECT
            competitor_id,
            market_share_percent,
            time_id,
            extraction_date
        FROM competitor_data
        """

        # Baca data yang akan dimasukkan
        df = pd.read_sql(query, con=engine_stag)

        if df.empty:
            print("\nTidak ada data baru yang akan dimasukkan
ke fact_competitor")
            print("Kemungkinan penyebab:")
            print("1. Data sudah ada di fact_competitor")
            print("2. Tidak ada kecocokan antara data di
staging dengan dim_competitor/dim_time")
            print("3. Data di dim_time belum diisi dengan
benar")
```

```
            # Debug: Cek contoh data di staging
            sample_staging = """
            SELECT DISTINCT
                competitor,
                time_periode,
                extraction_date
            FROM staging_market_share_report
            ORDER BY extraction_date DESC
            LIMIT 5
            """
            print("\nContoh data di
staging_market_share_report:")
            print(pd.read_sql(sample_staging,
con=engine_stag))

            # Debug: Cek contoh data di dim_competitor
            sample_comp = """
            SELECT
                competitor_id,
                competitor,
                LENGTH(TRIM(competitor)) as len_trimmed,
                LENGTH(competitor) as len_original
            FROM dwh.dim_competitor
            LIMIT 5
            """
            print("\nContoh data di dim_competitor:")
            print(pd.read_sql(sample_comp, con=engine_dwh))

            # Debug: Cek contoh data di dim_time
            sample_time = """
            SELECT
                time_id,
                time_periode,
                year,
                month,
                day
            FROM dwh.dim_time
            ORDER BY time_periode DESC
```

```
            LIMIT 5
            """
            print("\nContoh data di dim_time:")
            print(pd.read_sql(sample_time, con=engine_dwh))

            return

        # Masukkan data ke fact_competitor
        print(f"\nMenambahkan {len(df)} baris data ke
fact_competitor...")

        # Insert data ke fact_competitor
        with engine_dwh.connect() as conn:
            # Gunakan SQL langsung untuk insert
            insert_sql = """
            INSERT INTO dwh.fact_competitor
                (competitor_id, time_id,
market_share_percent, extraction_date)
            VALUES
                (:competitor_id, :time_id,
:market_share_percent, :extraction_date)
            ON CONFLICT (competitor_id, time_id) DO NOTHING
            """

            # Eksekusi untuk setiap baris
            result = conn.execute(
                text(insert_sql),
                df[['competitor_id', 'time_id',
'market_share_percent',
'extraction_date']].to_dict('records')
            )
            conn.commit()

            print(f"✅ Berhasil menambahkan {result.rowcount}
baris data ke fact_competitor")

            # Tampilkan ringkasan data yang baru ditambahkan
            summary = pd.read_sql(
                """
```

```
                SELECT
                    dc.competitor,
                    dt.time_periode,
                    fc.market_share_percent,
                    fc.extraction_date
                FROM dwh.fact_competitor fc
                JOIN dwh.dim_competitor dc ON
fc.competitor_id = dc.competitor_id
                JOIN dwh.dim_time dt ON fc.time_id =
dt.time_id
                ORDER BY dt.time_periode DESC,
fc.market_share_percent DESC
                LIMIT 5
                """,
                con=engine_dwh
            )

            if not summary.empty:
                print("\nRingkasan data terbaru di
fact_competitor:")
                print(summary)

    except Exception as e:
        print(f"❌ Error saat memuat data ke fact_competitor:
{str(e)}")
        import traceback
        traceback.print_exc()
        raise

def create_and_populate_dim_time():
    """Membuat dan mengisi tabel dim_time dengan data
tanggal"""
    try:
        print("\nMemulai pembuatan dan pengisian tabel
dim_time...")

        create_table_sql = text("""
        CREATE TABLE IF NOT EXISTS dwh.dim_time (
            time_id INT NOT NULL UNIQUE,
```

```
            timestamp TIMESTAMP NOT NULL,
            year INT NOT NULL,
            month INT NOT NULL,
            day INT NOT NULL,
            hour INT NOT NULL
        )
        """)

        # Use a connection to execute the SQL
        with engine_dwh.connect() as connection:
            connection.execute(create_table_sql)
            connection.commit()
        print("Tabel dim_time berhasil dibuat/ditemukan.")

        # Tentukan rentang tanggal yang ingin diisi (contoh:
5 tahun terakhir)
        end_date = datetime.now().date()
        start_date = end_date - pd.DateOffset(years=5)
        date_range = pd.date_range(start=start_date,
end=end_date, freq='D')

        # Buat DataFrame untuk dim_time dengan format yang
diminta
        dim_time = pd.DataFrame({
            'timestamp': date_range,
            'year': date_range.year,
            'month': date_range.month,
            'day': date_range.day,
            'hour': 0  # Default hour to 0 since we're only
dealing with daily data
        })

        # Generate time_id as integer in format YYYYMMDD
        dim_time['time_id'] = (dim_time['year'].astype(str) +

dim_time['month'].astype(str).str.zfill(2) +

dim_time['day'].astype(str).str.zfill(2)).astype(int)
```

```
        # Reorder columns to put time_id first
        dim_time = dim_time[['time_id', 'timestamp', 'year',
'month', 'day', 'hour']]

        # Hanya masukkan tanggal yang belum ada
        existing_dates = pd.read_sql("SELECT time_id FROM
dwh.dim_time", con=engine_dwh)
        if not existing_dates.empty:
            dim_time =
dim_time[~dim_time['time_id'].isin(existing_dates['time_id'])
]

        if not dim_time.empty:
            dim_time.to_sql('dim_time', con=engine_dwh,
schema='dwh',
                            if_exists='append', index=False)
            print(f"Berhasil menambahkan {len(dim_time)}
tanggal baru ke dim_time")
        else:
            print("Tidak ada tanggal baru yang perlu
ditambahkan ke dim_time")

    except Exception as e:
        print(f"Error saat mengisi dim_time: {str(e)}")
        raise

def load_dim_sensor():
    """Memuat data sensor ke dalam dimensi"""
    try:
        print("\nMemulai proses load data ke dim_sensor...")

        # Create dim_sensor table if it doesn't exist
        create_table_sql = """
        CREATE TABLE IF NOT EXISTS dwh.dim_sensor (
            sensor_id VARCHAR(50) PRIMARY KEY,
            temperature_c FLOAT
        )
        """
        with engine_dwh.connect() as connection:
```

```
            connection.execute(text(create_table_sql))
            connection.commit()
        print("Tabel dim_sensor siap digunakan.")

        # Get all sensors and temperature from
staging_warehouse_temp_sensor
        query = """
        SELECT sensor_id, temperature_c
        FROM staging_warehouse_temp_sensor
        """

        # Read sensor data from staging
        df = pd.read_sql(query, con=engine_stag)

        if not df.empty:
            # Insert all sensors into dim_sensor
            # Using to_sql with method='multi' for better
performance
            df.to_sql('dim_sensor',
                      con=engine_dwh,
                      schema='dwh',
                      if_exists='append',
                      index=False,
                      method='multi')
            print(f"Berhasil menambahkan {len(df)} data
sensor ke dim_sensor")
        else:
            print("Tidak ada data sensor yang ditemukan di
staging")

    except Exception as e:
        print(f"Error saat memuat data ke dim_sensor:
{str(e)}")
        raise

def create_fact_temperature_table(engine_dwh):
    """Creates the fact_temperature table if it doesn't
exist."""
    from sqlalchemy import text
```

```
    create_table_sql = text("""
    CREATE TABLE IF NOT EXISTS dwh.fact_temperature (
        fact_id SERIAL PRIMARY KEY,
        sensor_id VARCHAR REFERENCES
dwh.dim_sensor(sensor_id),
        time_id INT REFERENCES dwh.dim_time(time_id),
        temperature_c FLOAT,
        UNIQUE (sensor_id, time_id)  -- Prevents duplicate
entries for same sensor and time
    )
    """)

    create_index_sql = text("""
    CREATE INDEX IF NOT EXISTS idx_fact_temp_sensor_time ON
dwh.fact_temperature(sensor_id, time_id);
    """)

    with engine_dwh.connect() as conn:
        conn.execute(create_table_sql)
        conn.execute(create_index_sql)
        conn.commit()
    print("Tabel fact_temperature berhasil
dibuat/diperbarui.")

def populate_fact_temperature(engine_dwh, process_date=None):
    """
    Populates fact_temperature by joining dim_sensor and
dim_time.

    Parameters:
    - engine_dwh: SQLAlchemy engine
    - process_date: Date in 'YYYY-MM-DD' format. If None,
uses current date.
    """
    from sqlalchemy import text
    from datetime import datetime, timedelta

    try:
```

```
        # Set the processing date
        if process_date is None:
            process_date = datetime.now().date()
        else:
            process_date = datetime.strptime(process_date,
'%Y-%m-%d').date()

        print(f"\nMemproses data suhu untuk tanggal:
{process_date}")

        # Insert query that joins dim_sensor and dim_time to
get correct time_id for each reading
        insert_query = text("""
        INSERT INTO dwh.fact_temperature (sensor_id, time_id,
temperature_c)
        SELECT
            ds.sensor_id,
            dt.time_id,
            ds.temperature_c
        FROM dwh.dim_sensor ds
        JOIN dwh.dim_time dt ON
            EXTRACT(YEAR FROM dt.timestamp) = dt.year AND
            EXTRACT(MONTH FROM dt.timestamp) = dt.month AND
            EXTRACT(DAY FROM dt.timestamp) = dt.day AND
            EXTRACT(HOUR FROM dt.timestamp) = dt.hour
        WHERE ds.temperature_c IS NOT NULL
        AND DATE(dt.timestamp) = :target_date
        AND NOT EXISTS (
            SELECT 1
            FROM dwh.fact_temperature ft
            WHERE ft.sensor_id = ds.sensor_id
            AND ft.time_id = dt.time_id
        )
        """)

        with engine_dwh.connect() as conn:
            # Execute the insert query
            result = conn.execute(
                insert_query,
```

```
                {'target_date': process_date}
            )
            conn.commit()

            if result.rowcount > 0:
                print(f"✅ Berhasil menambahkan
{result.rowcount} data suhu baru")
            else:
                print("ℹ️ Tidak ada data suhu baru yang perlu
ditambahkan")

            # Verifikasi data yang sudah ada
            check_query = text("""
            SELECT COUNT(*)
            FROM dwh.fact_temperature ft
            JOIN dwh.dim_time dt ON ft.time_id = dt.time_id
            WHERE dt.year = :year AND dt.month = :month AND
dt.day = :day
            """)

            count_result = conn.execute(
                check_query,
                {'year': process_date.year, 'month':
process_date.month, 'day': process_date.day}
            ).scalar()

            print(f"📊 Total data suhu untuk tanggal
{process_date}: {count_result} record")

            # Tampilkan ringkasan data yang baru ditambahkan
            if result.rowcount > 0:
                summary_query = text("""
                SELECT
                    MIN(ft.temperature_c) as min_temp,
                    MAX(ft.temperature_c) as max_temp,
                    AVG(ft.temperature_c) as avg_temp,
                    COUNT(DISTINCT dt.time_id) as
unique_timestamps
                FROM dwh.fact_temperature ft
```

```
                JOIN dwh.dim_time dt ON ft.time_id =
dt.time_id
                WHERE dt.year = :year AND dt.month = :month
AND dt.day = :day
                """)

                summary = conn.execute(
                    summary_query,
                    {'year': process_date.year, 'month':
process_date.month, 'day': process_date.day}
                ).fetchone()

                print(f"🌡️  Statistik Suhu:")
                print(f"   - Rata-rata:
{summary.avg_temp:.2f}°C")
                print(f"   - Minimum:
{summary.min_temp:.2f}°C")
                print(f"   - Maksimum:
{summary.max_temp:.2f}°C")
                print(f"   - Jumlah timestamp unik:
{summary.unique_timestamps}")

    except Exception as e:
        print(f"Error saat memproses data suhu: {str(e)}")
        raise

# Example usage:
# 1. First, create the table (run once)
# create_fact_temperature_table(engine_dwh)

# 2. Then populate data (can be run daily)
# populate_fact_temperature(engine_dwh)  # For current date
# populate_fact_temperature(engine_dwh, '2023-06-23')  # For
specific date

def run_etl():
    """Menjalankan seluruh proses ETL"""
    print("Memulai proses ETL...")
```

```
    # Periksa data di staging
    check_staging_data()

    # Buat dan isi dim_time jika belum ada
    create_and_populate_dim_time()

    # Load data ke dimensi
    load_dim_competitor()
    load_dim_tweet()
    load_dim_topic()
    load_dim_sensor()

    # Load data ke fact tables
    load_fact_competitor_share()
    load_fact_sentiment()
    load_fact_temperature()
    populate_fact_temperature(engine_dwh)

    print("\nProses ETL selesai!")

if __name__ == "__main__":
    run_etl()

# Create a new figure for the dashboard with 2 rows and 1
column
fig_dashboard = Figure(figsize=(15, 18))
gs = fig_dashboard.add_gridspec(3, 1, height_ratios=[1, 1,
1.5])

# Horizontal Bar Chart untuk Market Share
ax_market_share = fig_dashboard.add_subplot(gs[1])
try:
    query = """
    SELECT competitor, market_share_percent AS market_share
    FROM staging_market_share_report
    WHERE competitor IS NOT NULL
    ORDER BY market_share DESC
    LIMIT 10
    """
```

```
    df_market = pd.read_sql(query, con=engine_stag)

    if not df_market.empty:
        bars = ax_market_share.barh(
            df_market['competitor'],
            df_market['market_share'],
            color='skyblue'
        )

        for bar in bars:
            width = bar.get_width()
            ax_market_share.text(
                width + 0.5,
                bar.get_y() + bar.get_height() / 2,
                f'{width:.2f}%',
                va='center'
            )

        ax_market_share.set_title('Top 10 Market Share by
Competitor', fontsize=14)
        ax_market_share.set_xlabel('Market Share (%)')
        ax_market_share.invert_yaxis()
        ax_market_share.grid(True, linestyle='--', alpha=0.7)
    else:
        ax_market_share.text(0.5, 0.5, 'Tidak ada data
tersedia',
                             ha='center', va='center',
transform=ax_market_share.transAxes)

except Exception as e:
    print(f"❌ Error dalam visualisasi market share: {e}")
    ax_market_share.text(0.5, 0.5, 'Error saat ambil data',
                         ha='center', va='center',
transform=ax_market_share.transAxes)
```

```
# Temperature Plot (Top subplot)
ax_temp = fig_dashboard.add_subplot(gs[0])

# Update the temperature plot to use
staging_warehouse_temp_sensor directly
try:
    # Query temperature data directly from
staging_warehouse_temp_sensor
    query = """
    SELECT
        timestamp,
        temperature_c as temperature
    FROM staging_warehouse_temp_sensor
    ORDER BY timestamp
    """
    df_temp = pd.read_sql(query, con=engine_stag)

    if not df_temp.empty:
        # Convert timestamp to datetime if it's not already
        df_temp['timestamp'] =
pd.to_datetime(df_temp['timestamp'])

        # Plot temperature over time
        ax_temp.plot(df_temp['timestamp'],
df_temp['temperature'],
                     marker='o', linestyle='-',
color='tab:red')

        # Customize the plot
        ax_temp.set_title('Temperature Trends Over Time',
fontsize=14, pad=20)
        ax_temp.set_xlabel('Date and Time', fontsize=12)
        ax_temp.set_ylabel('Temperature (°C)', fontsize=12)
        ax_temp.grid(True, linestyle='--', alpha=0.7)

        # Rotate x-axis labels for better readability
        plt.setp(ax_temp.get_xticklabels(), rotation=45,
ha='right')
```

```
    else:
        ax_temp.text(0.5, 0.5, 'No temperature data
available',
                     horizontalalignment='center',
                     verticalalignment='center',
                     transform=ax_temp.transAxes)
        ax_temp.set_xticks([])
        ax_temp.set_yticks([])

except Exception as e:
    print(f"Error generating temperature plot: {e}")
    ax_temp.text(0.5, 0.5, 'Error loading temperature data',
                 horizontalalignment='center',
                 verticalalignment='center',
                 transform=ax_temp.transAxes)
    ax_temp.set_xticks([])
    ax_temp.set_yticks([])

# Word Cloud (Bottom subplot)
ax_wc = fig_dashboard.add_subplot(gs[2])
try:
    # Fetch tweet data from staging_external_sentiment
    tweet_query = """
    SELECT tweet_text
    FROM staging_external_sentiment
    WHERE tweet_text IS NOT NULL
    """

    df_tweets = pd.read_sql(tweet_query, con=engine_stag)

    if not df_tweets.empty and 'tweet_text' in
df_tweets.columns:
        # Combine all tweets into a single string
        text = ' '.join(tweet for tweet in
df_tweets['tweet_text'].dropna())

        if text.strip():  # Only generate word cloud if
there's text
            # Generate word cloud
```

```
            wordcloud = WordCloud(
                width=800,
                height=400,
                background_color='white',
                max_words=200,
                contour_width=3,
                contour_color='steelblue'
            ).generate(text)

            # Display the word cloud
            ax_wc.imshow(wordcloud, interpolation='bilinear')
            ax_wc.axis('off')
            ax_wc.set_title('Word Cloud of Tweets',
fontsize=14, pad=20)
        else:
            ax_wc.text(0.5, 0.5, 'No tweet text available',
                     horizontalalignment='center',
                     verticalalignment='center',
                     transform=ax_wc.transAxes)
            ax_wc.set_xticks([])
            ax_wc.set_yticks([])
    else:
        ax_wc.text(0.5, 0.5, 'No tweet data available',
                 horizontalalignment='center',
                 verticalalignment='center',
                 transform=ax_wc.transAxes)
        ax_wc.set_xticks([])
        ax_wc.set_yticks([])

except Exception as e:
    print(f"Error generating word cloud: {e}")
    ax_wc.text(0.5, 0.5, 'Error loading tweet data',
              horizontalalignment='center',
              verticalalignment='center',
              transform=ax_wc.transAxes)
    ax_wc.set_xticks([])
    ax_wc.set_yticks([])

# Adjust layout and display
```

```
fig_dashboard.subplots_adjust(hspace=0.6)
fig_dashboard.text(0.5, 0.67, '-'*100, ha='center',
va='center', fontsize=8, color='gray', alpha=0.3)
fig_dashboard.text(0.5, 0.33, '-'*100, ha='center',
va='center', fontsize=8, color='gray', alpha=0.3)
fig_dashboard.tight_layout()

# Create and display the Tkinter window
root = tk.Tk()
root.title("Data Warehouse Visualization")
root.geometry("1200x1000")

# Create a canvas and add it to the Tkinter window
canvas = FigureCanvasTkAgg(fig_dashboard, master=root)
canvas.draw()
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

# Add a status bar
status_bar = tk.Label(
    root,
    text=f"Dashboard loaded successfully | Data last updated:
{datetime.now().strftime('%Y-%m-%d %H:%M:%S')}",
    bd=1,
    relief=tk.SUNKEN,
    anchor=tk.W
)
status_bar.pack(side=tk.BOTTOM, fill=tk.X)

# Run the Tkinter event loop
root.mainloop()
```

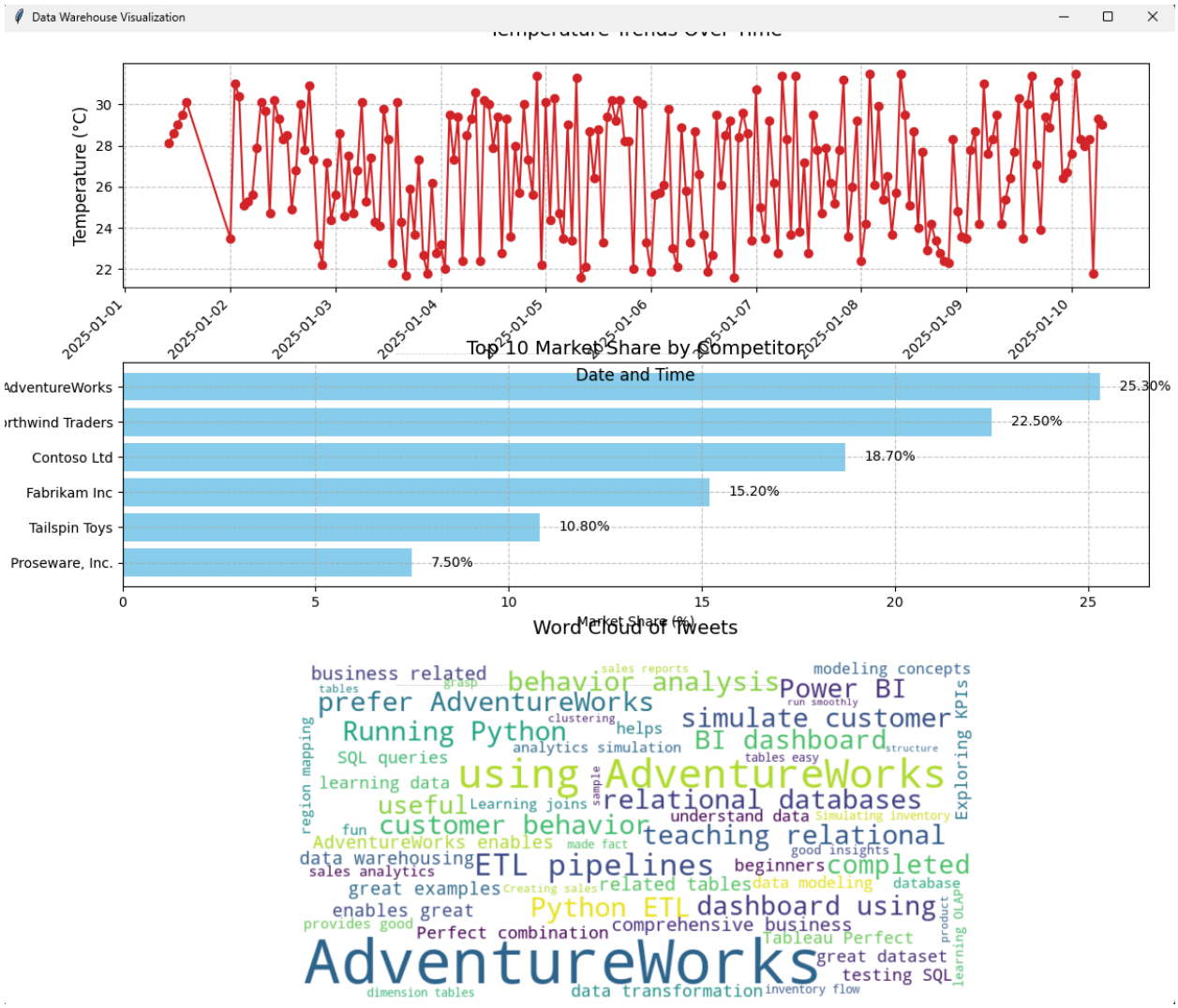
Pada Structure ini terdapat :

1. Data dari staging digabung dengan dimensi melalui surrogate key.
2. Tidak ada DDL, seluruh transformasi dilakukan via insert dan join.
3. Semua proses bersifat append dan aman dari overwrite.
4. Dalam Structure ini juga data dimasukkan ke masing-masing fact yang berhubungan dengan menjaga integritas relasi.

5. Seluruh pemuatan menggunakan `to_sql(..., if_exists='append')`.

6. Dashboard & Visualisasi

Berikut ini adalah hasil visualisasi dashboard grafik dari analisis sentimen dalam bentuk word cloud, persentase pangsa pasar dalam bentuk horizontal barchart, dan grafik garis untuk analisis sensor.



7. Evaluasi & Keunggulan

Berdasarkan hasil implementasi dan pengujian sistem data lakehouse AdventureWorks, berikut adalah hasil evaluasinya :

Aspek	Evaluasi
Ingest dan Integrasi Data	Dapat menggabungkan berbagai sumber data (`.csv`, `.pdf`, `.txt`) ke staging
Preprocessing & Analisis	Proses parsing PDF, normalisasi tweet, dan konversi suhu berjalan stabil
ETL ke DWH	Data berhasil dimuat ke warehouse dengan metode `append`, tanpa DDL atau overwrite
Desain Skema DW	Menggunakan Star Schema yang efisien dan scalable
Penerapan SCD	SCD Type 1 berhasil diterapkan untuk menjaga konsistensi `dim_competitor`
Dashboard Visualisasi	Visual interaktif dan informatif untuk 3 use-case utama

Terdapat keunggulannya yaitu diantaranya, adalah :

1. Tanpa Perintah DDL

   Seluruh proses ETL bebas dari instruksi CREATE, DROP, TRUNCATE, sehingga menjaga integritas struktur database dan cocok untuk sistem produksi yang sensitif terhadap perubahan skema.

2. Append-Only dan Historis

   Tidak ada dilakukannya pembersihan pada data warehouse. Data dimasukkan secara bertahap (append), dengan dukungan time_id untuk analisis historis.

3. Integrasi Multiformat Data

   Dapat menangani dan menggabungkan data dari berbagai format:

   a. CSV (sensor suhu)
   b. PDF (laporan kompetitor)
   c. TXT (tweet opini publik)

4. Modular dan Scalable

   Pipeline dan struktur DW dapat dengan mudah dikembangkan:

   a. Menambahkan lebih banyak tabel fakta (misalnya fact_sales)
   b. Memasukkan data real-time dengan pipeline streaming
   c. Migrasi ke cloud warehouse (BigQuery, Snowflake, dsb.)

5. Visualisasi Komprehensif

   Dashboard mempermudah eksplorasi data melalui:

   a. Grafik tren suhu (monitoring)
   b. Bar chart market share (strategi pasar)
   c. Word cloud opini publik (sentimen & awareness)

8. Kesimpulan

Proyek implementasi sitem Data Lakehouse AdventureWorks ini berhasil membuktikan bahwa penggabungan kekuatan data lake dan data warehouse mampu menciptakan solusi analitik yang fleksibel, teintegrasi, dan efisien untuk mendukung pengambilan keputusan bisnis yang berbasis data.

Melalui pipeline Ingest → Analyze → Staging → Structure → Data Warehouse, sistem ini dapat:

1. Mengelola berbagai format data, baik terstruktur (CSV) maupun tidak terstruktur (PDF dan teks)
2. Melakukan transformasi dan pemuatan data secara bertahap (append-only) tanpa perintah DDL yang merusak struktur
3. Menerapkan pendekatan **Star Schema** dengan pemisahan tabel fakta dan dimensi untuk optimalisasi query OLAP
4. Menyediakan visualisasi yang informatif dalam bentuk **dashboard** untuk monitoring suhu, analisis market share, dan analisis sentimen publik melalui word cloud
5. Menjaga konsistensi data historis melalui **implementasi SCD Type 1** untuk dimensi kompetitor

Dengan pendekatan ini, sistem Data Lakehouse yang dikembangkan tidak hanya memenuhi standar teknis dan prinsip *data governance*, tetapi juga membuka peluang untuk pengembangan lebih lanjut ke arah sistem analitik yang lebih besar dan real-time.

9. Lampiran

- Gambar Tree Sebelum Dilakukan Organize

```
C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\data_lake>tree /f
Folder PATH listing for volume Windows
Volume serial number is B074-6BA6
C:.
└───adventureworks
    ├───files
    │       market_share_report.pdf
    │       market_share_report_raw.txt
    │       tweet_data.csv
    │       warehouse_temp_sensor.csv
    │       warehouse_temp_sensor_raw.txt
    │
    ├───organized
    ├───processed
    └───tweets
            adventureworks_structured_150_tweets.txt
            adventureworks_structured_150_tweets_raw.txt
```

- Gambar Tree sesudah dilakukan Organize

```
Folder PATH listing for volume Windows
Volume serial number is B074-6BA6
C:.
└───adventureworks
    ├───files
    │       market_share_report.pdf
    │       market_share_report_raw.txt
    │       tweet_data.csv
    │       warehouse_temp_sensor.csv
    │       warehouse_temp_sensor_raw.txt
    │
    ├───organized
    │       adventureworks_structured_150_tweets.txt
    │       market_share_report.pdf
    │       warehouse_temp_sensor.csv
    │
    ├───processed
    └───tweets
            adventureworks_structured_150_tweets.txt
            adventureworks_structured_150_tweets_raw.txt
```

- Hasil Run Ingest

```
PS C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake>  c:; cd 'c:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake'; & 'C:\Users
\hansc\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\hansc\.windsurf\extensions\ky1inideteam.python-0.3.5-universal\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '3833' '--' 'C:\Use
rs\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\ingest.py'
 Starting data ingestion process...

Processing CSV warehouse_temp_sensor.csv...
Reading CSV chunks: 205rows [00:00, 3175.44rows/s]
Writing CSV text: 10000chars [00:00, 2163238.95chars/s]
Extracting PDF pages: 100%|[redacted]| 1/1 [00:00<00:00, 12.34page/s]

Processing TXT adventureworks_structured_150_tweets.txt...
Copying TXT file: 100%|[redacted]| 16.1k/16.1k [00:00<00:00, 11.6MB/s]

 Semua file berhasil diproses dalam 0.42 detik
```

- Hasil Run Analyze

```
2025-06-25 14:19:58,234 - INFO - Memproses file tweet: C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\data_lake\adventureworks\organized\adventureworks_structured_150_
tweets.txt
2025-06-25 14:19:58,236 - INFO - Memproses file TXT: C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\data_lake\adventureworks\organized\adventureworks_structured_150_tw
eets.txt
2025-06-25 14:19:58,755 - INFO - Berhasil memuat 150 baris ke tabel staging_external_sentiment
2025-06-25 14:19:58,755 - INFO - ✅ Data tweet berhasil dimuat ke staging_external_sentiment
2025-06-25 14:19:58,757 - INFO - Memproses file market share: C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\data_lake\adventureworks\organized\market_share_report.pdf
2025-06-25 14:19:58,759 - INFO - Memproses laporan market share: C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\data_lake\adventureworks\organized\market_share_report.
pdf
2025-06-25 14:19:59,947 - INFO - Data market share berhasil disimpan ke: C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\data_lake\adventureworks\processed\market_share
_report_20250625_141959.csv
2025-06-25 14:20:00,511 - INFO - Berhasil memuat 6 baris ke tabel staging_market_share_report
2025-06-25 14:20:00,515 - INFO - ✅ Data market share berhasil dimuat ke staging_market_share_report
2025-06-25 14:20:00,516 - INFO - ✅ Proses analisis data selesai
PS C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data lake>
```

- Masuk staging_external_sentiment

select * from staging_external_sentiment ses | Enter a SQL expression to filter results (use Ctrl+Space)

	tweet_id	tweet_text	timestamp	user_location	sentiment	matched_product
1	1	Creating sales reports using AdventureWorks sample data.	2024-06-03 19:55:00.000	Yogyakarta	positive	Touring Bike
2	2	AdventureWorks made fact and dimension tables easy to grasp.	2024-06-01 18:14:00.000	Jakarta	neutral	Helmet
3	3	AdventureWorks has customer and region mapping for BI.	2024-06-02 13:49:00.000	Singapore	negative	Helmet
4	4	Running Python ETL pipelines with AdventureWorks.	2024-06-02 09:32:00.000	Yogyakarta	negative	Road Tire
5	5	Using AdventureWorks to simulate customer behavior analysis.	2024-06-04 14:12:00.000	Yogyakarta	negative	Touring Bike
6	6	AdventureWorks enables great examples in data warehousing.	2024-06-04 20:40:00.000	Bandung	neutral	Road Tire
7	7	AdventureWorks is a great dataset for testing SQL queries!	2024-06-02 03:47:00.000	Bandung	negative	Unknown
8	8	AdventureWorks enables great examples in data warehousing.	2024-06-05 22:33:00.000	Medan	positive	Helmet
9	9	Just completed my Power BI dashboard using AdventureWorks.	2024-06-02 22:44:00.000	Surabaya	positive	Touring Bike
10	10	Just completed my Power BI dashboard using AdventureWorks.	2024-06-01 23:07:00.000	Surabaya	neutral	Unknown
11	11	Just completed my Power BI dashboard using AdventureWorks.	2024-06-02 20:21:00.000	Singapore	negative	Mountain Bike
12	12	Using AdventureWorks to simulate customer behavior analysis.	2024-06-04 23:54:00.000	Singapore	positive	Unknown
13	13	AdventureWorks + Tableau = Perfect combination for beginners.	2024-06-03 07:10:00.000	Bandung	neutral	Helmet
14	14	AdventureWorks is great for learning data transformation.	2024-06-01 13:45:00.000	Yogyakarta	positive	Bike Frame
15	15	Using AdventureWorks to simulate customer behavior analysis.	2024-06-02 01:45:00.000	Singapore	neutral	Road Tire
16	16	AdventureWorks is useful for sales analytics simulation.	2024-06-04 08:28:00.000	Surabaya	positive	Unknown
17	17	Using AdventureWorks to simulate customer behavior analysis.	2024-06-04 06:20:00.000	Surabaya	neutral	Helmet
18	18	Exploring KPIs from AdventureWorks for dashboards.	2024-06-03 04:43:00.000	Singapore	neutral	Unknown

- Masuk staging_market_share_report

select * from staging_market_share_report smsr | Enter a SQL expression to filter results (use Ctrl+Space)

	time_periode	competitor	market_share_percent	extraction_date
1	2024	AdventureWorks	25.3	2025-06-25
2	2024	Contoso Ltd	18.7	2025-06-25
3	2024	Fabrikam Inc	15.2	2025-06-25
4	2024	Northwind Traders	22.5	2025-06-25
5	2024	Tailspin Toys	10.8	2025-06-25
6	2024	Proseware, Inc.	7.5	2025-06-25

- Masuk staging_warehouse_temp_sensor

select * from staging_market_share_report smsr | Enter a SQL expression to filter results (use Ctrl+Space)

	time_periode	competitor	market_share_percent	extraction_date
1	2024	AdventureWorks	25.3	2025-06-25
2	2024	Contoso Ltd	18.7	2025-06-25
3	2024	Fabrikam Inc	15.2	2025-06-25
4	2024	Northwind Traders	22.5	2025-06-25
5	2024	Tailspin Toys	10.8	2025-06-25
6	2024	Proseware, Inc.	7.5	2025-06-25

- Masuk Folder Processed setelah dilakukan analyze

```
C:\Users\hansc\OneDrive - Institut Teknologi Sepuluh Nopember\Semester 4 SISFOR\DLH\data_lake\data_lake>tree /f
Folder PATH listing for volume Windows
Volume serial number is B074-6BA6
C:.
└───adventureworks
    ├───files
    │       market_share_report.pdf
    │       market_share_report_raw.txt
    │       tweet_data.csv
    │       warehouse_temp_sensor.csv
    │       warehouse_temp_sensor_raw.txt
    │
    ├───organized
    │       adventureworks_structured_150_tweets.txt
    │       market_share_report.pdf
    │       warehouse_temp_sensor.csv
    │
    ├───processed
    │       market_share_report_20250625_141959.csv
    │       processed_warehouse_temp_sensor.csv
    │       tweet_analysis_20250625.csv
    │
    └───tweets
            adventureworks_structured_150_tweets.txt
            adventureworks_structured_150_tweets_raw.txt
```

- Hasil setelah menjalankan Structure

```
Memproses data suhu untuk tanggal: 2025-06-25
✅ Berhasil menambahkan 205 data suhu baru
📊 Total data suhu untuk tanggal 2025-06-25: 205 record
🌡️ Statistik Suhu:
   - Rata-rata: 26.74°C
   - Minimum: 21.60°C
   - Maksimum: 31.50°C
   - Jumlah timestamp unik: 1

Proses ETL selesai!
```

- Masuk dim_time

select * from dwh.dim_time dt | Enter a SQL expression to filter results (use Ctrl+Space)

	time_id	timestamp	year	month	day	hour
1	20,200,625	2020-06-25 00:00:00.000	2,020	6	25	0
2	20,200,626	2020-06-26 00:00:00.000	2,020	6	26	0
3	20,200,627	2020-06-27 00:00:00.000	2,020	6	27	0
4	20,200,628	2020-06-28 00:00:00.000	2,020	6	28	0
5	20,200,629	2020-06-29 00:00:00.000	2,020	6	29	0
6	20,200,630	2020-06-30 00:00:00.000	2,020	6	30	0
7	20,200,701	2020-07-01 00:00:00.000	2,020	7	1	0
8	20,200,702	2020-07-02 00:00:00.000	2,020	7	2	0
9	20,200,703	2020-07-03 00:00:00.000	2,020	7	3	0
10	20,200,704	2020-07-04 00:00:00.000	2,020	7	4	0
11	20,200,705	2020-07-05 00:00:00.000	2,020	7	5	0
12	20,200,706	2020-07-06 00:00:00.000	2,020	7	6	0
13	20,200,707	2020-07-07 00:00:00.000	2,020	7	7	0
14	20,200,708	2020-07-08 00:00:00.000	2,020	7	8	0
15	20,200,709	2020-07-09 00:00:00.000	2,020	7	9	0
16	20,200,710	2020-07-10 00:00:00.000	2,020	7	10	0
17	20,200,711	2020-07-11 00:00:00.000	2,020	7	11	0
18	20,200,712	2020-07-12 00:00:00.000	2,020	7	12	0
19	20,200,713	2020-07-13 00:00:00.000	2,020	7	13	0

- Masuk dim_tweet

	tweet_id	author_id	tweet_text
1	131	unknown	Simulating inventory flow with AdventureWorks tables.
2	136	unknown	AdventureWorks provides good insights for learning OLAP.
3	18	unknown	Exploring KPIs from AdventureWorks for dashboards.
4	109	unknown	AdventureWorks made fact and dimension tables easy to grasp.
5	40	unknown	AdventureWorks has customer and region mapping for BI.
6	20	unknown	AdventureWorks helps me understand data modeling concepts.
7	15	unknown	Using AdventureWorks to simulate customer behavior analysis.
8	23	unknown	I prefer AdventureWorks for teaching relational databases.
9	57	unknown	AdventureWorks has comprehensive business-related tables.
10	44	unknown	ETL pipelines run smoothly using AdventureWorks structure.
11	55	unknown	Using AdventureWorks to simulate customer behavior analysis.
12	56	unknown	Just completed my Power BI dashboard using AdventureWorks.
13	146	unknown	Using AdventureWorks to simulate customer behavior analysis.
14	47	unknown	AdventureWorks made fact and dimension tables easy to grasp.
15	8	unknown	AdventureWorks enables great examples in data warehousing.
16	79	unknown	AdventureWorks product tables are useful for clustering.
17	32	unknown	AdventureWorks is useful for sales analytics simulation.
18	4	unknown	Running Python ETL pipelines with AdventureWorks.
19	89	unknown	Exploring KPIs from AdventureWorks for dashboards.

- Masuk dim_topic

	topic_id	keyword
1	1	Touring Bike
2	2	Bike Frame
3	3	Mountain Bike
4	4	Helmet
5	5	Unknown
6	6	Road Tire

- Masuk dim_sensor

select *from dwh.dim_sensor | Enter a SQL expression to

	sensor_id	temperature_c
1	sensor_1	28.1
2	sensor_2	28.6
3	sensor_3	29
4	sensor_4	29.5
5	sensor_5	30.1
6	sensor_6	23.5
7	sensor_7	31
8	sensor_8	30.4
9	sensor_9	25.1
10	sensor_10	25.3
11	sensor_11	25.6
12	sensor_12	27.9
13	sensor_13	30.1
14	sensor_14	29.7
15	sensor_15	24.7
16	sensor_16	30.2
17	sensor_17	29.3
18	sensor_18	28.3
19	sensor_19	28.5

- Masuk dim_competitor

select *from dwh.dim_competitor dc | Enter a SQL expression to filter results (use Ctrl+S

	competitor_id	competitor	market_share_percent
1	1	Fabrikam Inc	15.2
2	2	Tailspin Toys	10.8
3	3	Proseware, Inc.	7.5
4	4	AdventureWorks	25.3
5	5	Northwind Traders	22.5
6	6	Contoso Ltd	18.7

- Masuk fact_sentiment

select *from dwh.fact_sentiment fs2 | Enter a SQL expression to filter results (use Ctrl+Space)

	fact_id	tweet_id	topic_id	time_id	polarity
1	1	1	1	20,240,603	1
2	2	2	4	20,240,601	0
3	3	3	4	20,240,602	-1
4	4	4	6	20,240,602	-1
5	5	5	1	20,240,604	-1
6	6	6	6	20,240,604	0
7	7	7	5	20,240,602	-1
8	8	8	4	20,240,605	1
9	9	9	1	20,240,602	1
10	10	10	5	20,240,601	0
11	11	11	3	20,240,602	-1
12	12	12	5	20,240,604	1
13	13	13	4	20,240,603	0
14	14	14	2	20,240,601	1
15	15	15	6	20,240,602	0
16	16	16	5	20,240,604	1
17	17	17	4	20,240,604	0
18	18	18	5	20,240,603	0
19	19	19	6	20,240,601	0

- Masuk fact_competitor_share

select * from dwh.fact_competitor_share fcs | Enter a SQL expression to filter results (use Ctrl+Space)

	fact_id	competitor_id	market_share_percent	time_id
1	1	4	25.3	20,250,625
2	2	6	18.7	20,250,625
3	3	1	15.2	20,250,625
4	4	5	22.5	20,250,625
5	5	2	10.8	20,250,625
6	6	3	7.5	20,250,625

- Masuk fact_temperature

select * from dwh.fact_temperature ft | Enter a SQL expression to filter results (use Ctrl+Sp

	fact_id	sensor_id	time_id	temperature_c
1	1	sensor_98	20,250,625	22
2	2	sensor_126	20,250,625	30.7
3	3	sensor_104	20,250,625	25.7
4	4	sensor_43	20,250,625	22.3
5	5	sensor_24	20,250,625	30.9
6	6	sensor_198	20,250,625	27.6
7	7	sensor_191	20,250,625	23.9
8	8	sensor_58	20,250,625	29.4
9	9	sensor_187	20,250,625	23.5
10	10	sensor_39	20,250,625	24.3
11	11	sensor_173	20,250,625	23.6
12	12	sensor_13	20,250,625	30.1
13	13	sensor_178	20,250,625	31
14	14	sensor_177	20,250,625	24.2
15	15	sensor_38	20,250,625	27.4
16	16	sensor_15	20,250,625	24.7
17	17	sensor_183	20,250,625	25.4
18	18	sensor_32	20,250,625	24.6

- Hasil Visualisasi Dashboard