LAPORAN UJIAN AKHIR SEMESTER

DATA WAREHOUSE PADA DATA PENJUALAN MOBIL TOYOTA BEKAS

Disusun untuk Memenuhi Ujian Akhir Semester
Mata Kuliah Data Warehouse

Dosen Pengampu :

Erza Sofian, S.Kom., M.Sc.

Universitas Ary Ginanjar

Oleh :

Agus Saputra Hamzah	(2310120018)
Zinedine Daffa Izaaz	(2310120014)
Muhammad Abyan Al Wafi Effendy	(2310120024)

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS ARY GINANJAR

2026

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, penulis dapat menyelesaikan Laporan Ujian Akhir Semester (UAS) Mata Kuliah Data Warehouse ini dengan baik dan tepat waktu.

Laporan ini disusun sebagai salah satu bentuk pemenuhan tugas akademik pada mata kuliah Data Warehouse. Adapun laporan ini membahas penerapan konsep data warehouse, proses analisis data berbasis OLAP, hingga penyusunan laporan hasil analisis dalam bentuk dokumen PDF menggunakan bahasa pemrograman Python.

Penulis menyadari bahwa penyusunan laporan ini tidak terlepas dari bantuan, bimbingan, serta dukungan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada bapak Erza Sofian, S.Kom., M.Sc. selaku dosen pengampu yang telah memberikan arahan dan ilmu selama perkuliahan, serta kepada semua pihak yang turut membantu dalam penyelesaian laporan ini.

Penulis menyadari bahwa laporan ini masih memiliki keterbatasan dan kekurangan. Oleh sebab itu, penulis mengharapkan kritik dan saran yang membangun demi perbaikan dan pengembangan di masa mendatang. Semoga laporan ini dapat memberikan manfaat dan menambah wawasan, khususnya dalam penerapan konsep Data Warehouse dan analisis data.

Jakarta, 30 Januari 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dan meningkatnya volume data telah mendorong organisasi untuk mengelola data secara lebih terstruktur dan sistematis. Salah satu pendekatan yang banyak digunakan dalam pengelolaan dan analisis data adalah data warehouse, yaitu sistem yang dirancang untuk mengintegrasikan data dari berbagai sumber guna mendukung proses analisis dan pengambilan keputusan.(Gusti Ngurah Agung Trisna Putra et al., n.d.)

Dalam industri otomotif, khususnya pada pasar mobil bekas, data memiliki peran yang sangat penting. Penentuan harga dan nilai jual mobil bekas dipengaruhi oleh berbagai faktor, seperti model kendaraan, tahun produksi, harga, jenis transmisi, jarak tempuh (mileage), jenis bahan bakar, pajak kendaraan (road tax), efisiensi bahan bakar (mpg), serta kapasitas mesin. Pengolahan data tersebut secara manual menjadi tidak efisien ketika jumlah data semakin besar, sehingga diperlukan sistem yang mampu mengolah dan menganalisis data secara optimal.

Pada proyek ini digunakan dataset Toyota Used Cars Market Insights yang diperoleh dari Kaggle. Dataset ini berisi data mobil Toyota yang dipasarkan di pasar mobil bekas dengan berbagai atribut yang relevan untuk dianalisis. Dataset tersebut sangat sesuai untuk diterapkan dalam konsep data warehouse karena memiliki volume data yang cukup besar serta atribut yang beragam.

Dalam penyusunan data warehouse, proyek ini menerapkan konsep ETL (Extract, Transform, Load). Tahap *extract* dilakukan dengan mengambil dataset dari Kaggle, tahap *transform* dilakukan melalui proses data cleaning dan normalisasi data menggunakan Python, sedangkan tahap *load* dilakukan dengan menyiapkan data ke dalam struktur yang siap untuk dianalisis. Selanjutnya, proses analisis data dilakukan menggunakan konsep OLAP (Online Analytical Processing) melalui penerapan query relasi dan fungsi agregasi untuk memperoleh insight dari data yang tersedia.

Hasil analisis data kemudian disajikan dalam bentuk laporan otomatis berformat PDF menggunakan Python, sehingga informasi yang dihasilkan dapat disajikan secara terstruktur, mudah dipahami, dan profesional. Dengan demikian, proyek ini diharapkan mampu memberikan gambaran nyata mengenai penerapan konsep data warehouse, ETL, dan OLAP dalam analisis data pasar mobil bekas Toyota.

1.2 Rumusan Masalah

1. Bagaimana proses data cleaning dan normalisasi data pada dataset yang digunakan?
2. Bagaimana perancangan dan penerapan query relasi antar tabel dalam sistem data warehouse?
3. Bagaimana penerapan analisis OLAP menggunakan fungsi agregasi untuk memperoleh insight dari data mobil bekas Toyota?

4. Bagaimana cara menghasilkan laporan hasil analisis data dalam format PDF menggunakan Python sesuai ketentuan yang ditetapkan?

1.3 Tujuan

1. Menjelaskan dan menerapkan proses data cleaning dan normalisasi data pada dataset yang digunakan agar data memiliki kualitas yang baik dan siap untuk dianalisis.
2. Merancang dan menerapkan query relasi antar tabel dalam sistem data warehouse guna mendukung proses pengolahan dan analisis data.
3. Menerapkan analisis OLAP menggunakan fungsi agregasi untuk memperoleh insight yang berkaitan dengan karakteristik dan pola data mobil bekas Toyota.
4. Menghasilkan laporan hasil analisis data dalam format PDF secara otomatis menggunakan Python sesuai dengan ketentuan yang telah ditetapkan.

BAB II

LANDASAN TEORI

2.1 Data Cleaning

Data cleaning merupakan proses awal dalam pengolahan data yang bertujuan untuk meningkatkan kualitas data sebelum digunakan pada tahap analisis. Proses ini meliputi kegiatan pemeriksaan struktur data, penanganan data duplikat, penghapusan nilai yang tidak valid, serta standarisasi format data. Data yang tidak bersih dapat menyebabkan kesalahan dalam proses analisis dan menghasilkan insight yang tidak akurat.

Dalam konteks data warehouse, data cleaning menjadi bagian penting dari proses ETL (Extract, Transform, Load), khususnya pada tahap *transform*. Dengan melakukan data cleaning, data yang digunakan menjadi lebih konsisten, akurat, dan siap untuk diproses lebih lanjut seperti normalisasi dan agregasi data.

2.2 Normalisasi Data

Normalisasi data merupakan teknik perancangan basis data yang bertujuan untuk mengurangi redundansi dan menghindari terjadinya anomali data. Proses normalisasi dilakukan dengan memecah data ke dalam beberapa tabel berdasarkan ketergantungan fungsional antar atribut. Normalisasi umumnya dilakukan melalui beberapa tahapan yang dikenal sebagai bentuk normal, yaitu First Normal Form (1NF), Second Normal Form (2NF), dan Third Normal Form (3NF).

Pada sistem data warehouse, normalisasi sering digunakan pada tahap awal perancangan struktur data untuk memastikan integritas dan konsistensi data. Dengan menerapkan normalisasi hingga bentuk normal ketiga (3NF), data dapat disusun ke dalam tabel dimensi dan tabel fakta yang saling terhubung melalui primary key dan foreign key. Struktur ini mendukung efisiensi penyimpanan data serta memudahkan proses pengolahan dan analisis data. (Franciska Mey Dina, 2022)

2.3 Agregasi Data

Agregasi data merupakan proses pengolahan data dengan cara mengelompokkan dan merangkum data berdasarkan dimensi tertentu untuk menghasilkan informasi yang lebih bermakna. Proses agregasi umumnya dilakukan menggunakan fungsi-fungsi agregat seperti SUM, AVG, COUNT, MIN, dan MAX. Agregasi data menjadi komponen utama dalam analisis OLAP (Online Analytical Processing).

Dalam data warehouse, agregasi digunakan untuk menganalisis data dari berbagai sudut pandang, misalnya berdasarkan waktu, jenis produk, atau kategori tertentu. Dengan adanya agregasi data, pengguna dapat memperoleh insight seperti tren penjualan, rata-rata harga, serta distribusi data secara multidimensi. Oleh karena itu, agregasi data berperan penting dalam mendukung pengambilan keputusan berbasis data.

BAB III

METODOLOGI

3.1 Sumber Data

Dataset yang digunakan dalam laporan ini diperoleh dari platform Kaggle dengan judul *Toyota Used Cars Market Insights*. Dataset ini berisi data kendaraan merek Toyota yang dipasarkan di pasar mobil bekas dan mencakup berbagai atribut penting yang relevan untuk analisis data. Dengan jumlah record berisi 6.738 record data yang merepresentasikan kendaraan merek Toyota yang dipasarkan di pasar mobil bekas.

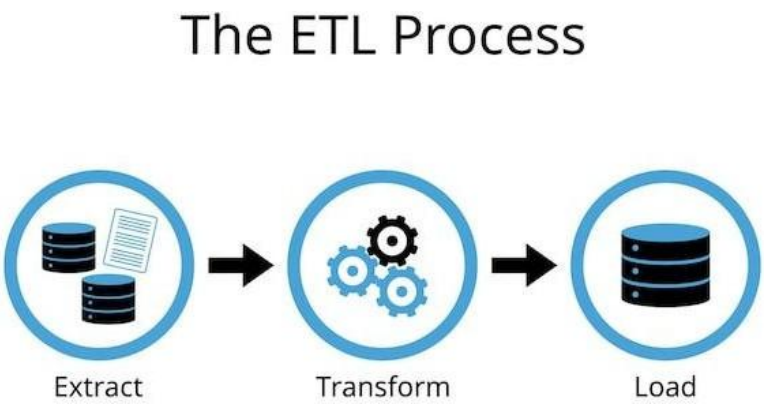
Atribut yang terdapat dalam dataset meliputi model kendaraan, tahun produksi, harga, jenis transmisi, jarak tempuh (mileage), jenis bahan bakar, pajak kendaraan (road tax), efisiensi bahan bakar (miles per gallon/mpg), serta kapasitas mesin. Dataset ini dipilih karena memiliki karakteristik data yang beragam dan representatif untuk diterapkan dalam konsep data warehouse, khususnya pada proses ETL dan analisis OLAP. Adapun recor data yang didapatkan dapat dilihat pada gambar dibawah ini.

model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
Auris	2015	12250	Automatic	50920	Hybrid	0	76.4	1.8
Auris	2016	13000	Automatic	38409	Hybrid	0	78.5	1.8
Auris	2016	14960	Automatic	28121	Hybrid	0	70.6	1.8
Auris	2013	12200	Automatic	29243	Hybrid	0	74.3	1.8
Auris	2013	5995	Manual	79304	Petrol	145	49.6	1.6
Auris	2015	10995	Automatic	53042	Hybrid	0	72.4	1.8
Auris	2015	13000	Automatic	38876	Hybrid	0	70.6	1.8
Auris	2014	7498	Manual	30805	Petrol	125	51.4	1.3
Auris	2015	12495	Automatic	39964	Hybrid	0	78.5	1.8
Auris	2016	9985	Manual	27600	Petrol	30	58.9	1.2
Auris	2016	9985	Manual	27600	Petrol	30	58.9	1.2
Auris	2015	9225	Manual	42588	Petrol	125	52.3	1.2
Auris	2017	12495	Manual	16589	Petrol	145	52.3	1.2
Auris	2013	13995	Automatic	12208	Hybrid	0	70.6	1.8
Auris	2016	10995	Manual	19252	Petrol	30	58.9	1.2
Auris	2016	8995	Manual	53042	Diesel	20	67.3	1.6
Auris	2016	12495	Automatic	33487	Hybrid	0	78.5	1.8
Auris	2017	13495	Manual	4897	Petrol	145	52.3	1.2
Avensis	2012	8750	Automatic	40796	Petrol	205	42.2	1.8
Avensis	2017	12498	Manual	24705	Petrol	145	47.1	1.8
Avensis	2017	12698	Manual	21431	Petrol	145	47.1	1.8
Avensis	2015	9998	Manual	19363	Diesel	20	67.3	1.6
Avensis	2016	9295	Manual	56501	Diesel	20	67.3	1.6
Avensis	2017	11998	Manual	27196	Petrol	145	47.1	1.8
Avensis	2016	11503	Manual	63956	Diesel	30	62.8	2.0
Avensis	2015	8776	Manual	57995	Diesel	20	67.3	1.6
Avensis	2013	7498	Manual	57480	Diesel	30	62.8	2.0
Avensis	2016	11495	Manual	40889	Diesel	20	67.3	1.6
Avensis	2017	11795	Manual	37983	Diesel	145	67.3	1.6
Avensis	2016	10995	Manual	55086	Diesel	20	67.3	1.6
Avensis	2018	12998	Automatic	35507	Petrol	145	44.1	1.8
Avensis	2016	9995	Manual	61300	Diesel	20	67.3	1.6
Avensis	2018	14495	Manual	31174	Diesel	145	61.4	2.0
Avensis	2017	13998	Manual	11855	Petrol	145	47.1	1.8
Avensis	2018	13695	Manual	37191	Diesel	145	67.3	1.6
Avensis	2018	12995	Manual	32236	Diesel	145	67.3	1.6
Avensis	2016	10895	Manual	30879	Diesel	20	67.3	1.6

Gambar 1 : Dataset Toyota Used Cars Market Insight
Sumber : sumberdata

Dataset *Toyota Used Cars Market Insights* dipilih karena memiliki struktur data yang kaya, variatif, serta sesuai untuk dilakukan analisis multidimensi guna menghasilkan insight yang bermanfaat terkait kondisi pasar mobil bekas Toyota. Struktur dataset awal masih bersifat data mentah (*raw data*), sehingga memungkinkan adanya permasalahan seperti nilai kosong, data tidak konsisten, maupun redundansi data. Oleh karena itu, dataset ini perlu melalui tahapan ETL sebelum digunakan dalam proses analisis data lebih lanjut.

3.2 Prose ETL



Gambar 2 : Proses ETL

Sumber : freepik.com

ETL (*Extract, Transform, Load*) merupakan salah satu proses utama dalam pengembangan data warehouse yang berfungsi untuk mengintegrasikan data dari berbagai sumber ke dalam suatu sistem penyimpanan terpusat yang siap digunakan untuk analisis. Proses ETL bertujuan untuk memastikan data yang digunakan memiliki kualitas yang baik, konsisten, dan sesuai dengan kebutuhan analisis.

3.2.1 Extract

Tahap Extract merupakan tahap awal dalam proses ETL, yaitu proses pengambilan data dari sumber data. Pada laporan ini, proses extract dilakukan dengan mengunduh dataset *Toyota Used Cars Market Insights* dari Kaggle.

Dataset yang telah diunduh kemudian diimpor ke dalam lingkungan pemrograman Python menggunakan library yang mendukung pengolahan data. Tahap ini bertujuan untuk memastikan bahwa data dapat dibaca dan diproses pada tahap selanjutnya.

3.2.2 Transform

Tahap Transform bertujuan untuk mengolah data mentah menjadi data yang bersih, konsisten, dan terstruktur. Pada laporan ini, proses transform dilakukan melalui dua tahapan utama, yaitu data cleaning dan normalisasi tabel.

1. Data Cleaning Menggunakan Phyton

Proses data cleaning dilakukan terlebih dahulu menggunakan Python untuk meningkatkan kualitas data. Tahapan data cleaning meliputi:

a) Pengecekan nilai kosong,
b) Penghapusan data duplikat,
c) Standarisasi nama kolom,
d) Pembersihan nilai kategorikal tidak relevan,
e) Penanganan nilai tidak logis, serta penanganan outlier.

Dengan dilakukannya proses data cleaning, diperoleh dataset yang lebih akurat dan siap digunakan pada tahap selanjutnya. Adapun tahapan yang dilakukan sebagai berikut :

1) Import Library dan Membaca Dataset

```
import numpy as np
```

```
df = pd.read_csv(
    r"C:\Users\LENOVO\Downloads\datapenjualanmobil.csv",
    sep=';'
)

df.head()
```

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	GT86	2016	16000	Manual	24089	Petrol	265	36.2	2.0
1	GT86	2017	15995	Manual	18615	Petrol	145	36.2	2.0
2	GT86	2015	13998	Manual	27469	Petrol	265	36.2	2.0
3	GT86	2017	18998	Manual	14736	Petrol	150	36.2	2.0
4	GT86	2017	17498	Manual	36284	Petrol	145	36.2	2.0

2) Pemeriksaan Struktur Awal Data
   - Mengecek informasi dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6738 entries, 0 to 6737
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   model         6738 non-null   object
 1   year          6738 non-null   int64
 2   price         6738 non-null   int64
 3   transmission  6738 non-null   object
 4   mileage       6738 non-null   int64
 5   fuelType      6738 non-null   object
 6   tax           6738 non-null   int64
 7   mpg           6738 non-null   float64
 8   engineSize    6738 non-null   float64
dtypes: float64(2), int64(4), object(3)
memory usage: 473.9+ KB
```

   - Mengecek jumlah nilai kosong

```
df.isnull().sum()
```

```
model           0
year            0
price           0
transmission    0
mileage         0
fuelType        0
tax             0
mpg             0
engineSize      0
dtype: int64
```

3) Penanganan Data Duplikat
Pada proses ini mengidentifikasi jumlah data duplikat, menghapus data duplikat, dan memastikan kembali bahwa duplikasi telah dihilangkan.

```
df.duplicated().sum()
```

```
39
```

```
df = df.drop_duplicates()
```

```
df.duplicated().sum()
```

```
0
```

4) Standarisasi Nama Kolom
Proses ini menghapus spasi yang berlebih dan mengubah nama kolom menjadi huruf kecil.

```
df.columns = (
    df.columns
      .str.strip()
      .str.lower()
)
```

5) Pembersihan Data Kategorikal
Dengan mengindentifikasi nilai pada kolom transmission dan menghapus data dengan kategori "Other" karena tidak informatif

```
df['transmission'].value_counts()
```

```
transmission
Manual       3826
Automatic    2657
Semi-Auto     254
Other           1
Name: count, dtype: int64
```

```
df = df[df['transmission'] != 'Other']
```

```
df['transmission'].value_counts()
```

```
transmission
Manual       3826
Automatic    2657
Semi-Auto     254
Name: count, dtype: int64
```

6) Validasi Nilai Numerik
Menghapus data dengan engineSize ≤ 0 dan Memfilter nilai `mpg` agar berada pada rentang wajar (10–150)

```
df = df[df['enginesize'] > 0]
```

```
df = df[(df['mpg'] >= 10) & (df['mpg'] <= 150)]
```

7) Penanganan Outlier Pada Harga
Menghitung Q1, Q3, dan IQR dan menghapus data harga yang berada di luar batas IQR

```
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1

df = df[(df['price'] >= Q1 - 1.5 * IQR) &
        (df['price'] <= Q3 + 1.5 * IQR)]
```

8) Finalisasi Dataset
- Mengecek statistik deskriptif akhir (df.describe())

```
df.describe()
```

	year	price	mileage	tax	mpg	enginesize
count	6188.000000	6188.000000	6188.000000	6188.000000	6188.000000	6188.000000
mean	2016.875566	11942.719619	20867.808985	91.340498	63.381884	1.426826
std	1.763547	4850.756871	14181.400990	71.276098	12.138251	0.392377
min	2000.000000	2490.000000	2.000000	0.000000	30.400000	1.000000
25%	2016.000000	8298.000000	9758.750000	0.000000	56.500000	1.000000
50%	2017.000000	10695.000000	18309.500000	135.000000	65.700000	1.500000
75%	2018.000000	14295.000000	29871.000000	145.000000	70.600000	1.800000
max	2020.000000	25000.000000	63475.000000	330.000000	134.500000	3.000000

- Mengecek jumlah nilai unik (df.nunique())

```
df.nunique()
```

```
model            17
year             19
price          1844
transmission      3
mileage        5291
fueltype          4
tax              23
mpg              65
enginesize       12
dtype: int64
```

- Menyimpan dataset bersih ke file CSV baru

```
df.to_csv("datapenjualanmobil_clean_final.csv", index=False)
```

2. Normalisasi Tabel

Setelah dataset melalui tahap pembersihan data (*data cleaning*), langkah selanjutnya adalah melakukan normalisasi data yang dilakukan dalam phyton. Normalisasi bertujuan untuk mengurangi redundansi data, meningkatkan konsistensi, serta membentuk struktur data yang lebih terorganisir sehingga mendukung proses analisis data secara efisien dalam sistem data warehouse.

Pada penelitian ini, proses normalisasi dilakukan dengan memisahkan atribut-atribut kategorikal ke dalam tabel dimensi, sementara data numerik dan transaksi disimpan dalam tabel fakta. Pendekatan ini menghasilkan struktur data yang memenuhi hingga Third Normal Form (3NF) serta dirancang dalam bentuk star schema sederhana. Adapun tahapan yang dilakukan dalam normalisasi tabel sebagai berikut :

1. Pembentukan Tabel Dimensi Model

   Tabel dimensi model dibentuk untuk menyimpan informasi unik mengenai jenis model kendaraan Toyota. Proses pembentukan tabel ini dilakukan dengan menghilangkan duplikasi data pada kolom *model* dan menambahkan atribut model_id sebagai *primary key* yang bersifat unik. Pemisahan tabel dimensi model bertujuan untuk menghindari pengulangan data model pada tabel utama serta mempermudah proses pengelompokan data pada analisis OLAP.

   Kode :

```
# Membuat tabel dimensi: Model

dim_model = df[['model']].drop_duplicates().reset_index(drop=True)

dim_model['model_id'] = dim_model.index + 1

dim_model = dim_model[['model_id', 'model']]

dim_model.head()
```

	model_id	model
0	1	GT86
1	2	Corolla
2	3	RAV4
3	4	Yaris
4	5	Auris

2. Pembentukan Tabel Dimensi Transmisi

   Tabel dimensi transmisi dibuat untuk menyimpan jenis transmisi kendaraan, seperti *manual* dan *automatic*. Data pada kolom *transmission* diekstraksi menjadi tabel tersendiri dengan nilai yang unik dan diberikan atribut transmission_id

sebagai *primary key*. Dengan memisahkan data transmisi ke dalam tabel dimensi, redundansi data dapat diminimalkan dan struktur data menjadi lebih konsisten.

Kode :

```
# Membuat Tabel Dimensi: Transmisi
dim_transmission=
df[['transmission']].drop_duplicates().reset_index(drop=True)

dim_transmission['transmission_id'] = dim_transmission.index + 1

dim_transmission = dim_transmission[['transmission_id',
'transmission']]

dim_transmission.head()
```

	transmission_id	transmission
0	1	Manual
1	2	Automatic
2	3	Semi-Auto

3. Pembentukan Dimensi Tipe Bahan Bakar

Tabel dimensi tipe bahan bakar dibentuk untuk menyimpan informasi mengenai jenis bahan bakar kendaraan, seperti *petrol*, *diesel*, dan *hybrid*. Setiap jenis bahan bakar disimpan satu kali dalam tabel dimensi dengan penambahan fueltype_id sebagai *primary key*. Pemisahan ini mendukung analisis berdasarkan jenis bahan bakar tanpa adanya duplikasi data pada tabel fakta.

Kode :

```
# Membuat Tabel Dimensi: Tipe Bahan Bakar

dim_fueltype =
df[['fueltype']].drop_duplicates().reset_index(drop=True)

dim_fueltype['fueltype_id'] = dim_fueltype.index + 1

dim_fueltype = dim_fueltype[['fueltype_id', 'fueltype']]

dim_fueltype.head()
```

	fueltype_id	fueltype
0	1	Petrol
1	2	Other
2	3	Hybrid
3	4	Diesel

4. Pembentukan Tabel Fact Sales

Tabel fakta penjualan dibentuk dengan menggabungkan dataset utama dengan tabel-tabel dimensi yang telah dibuat sebelumnya. Pada tahap ini, nilai teks pada atribut *model*, *transmission*, dan *fueltype* digantikan dengan foreign key yang mengacu pada tabel dimensi masing-masing. Tabel fakta ini juga dilengkapi dengan atribut sale_id sebagai *primary key* untuk mengidentifikasi setiap transaksi secara unik.

Kode :

```
# Membuat Tabel Fakta: Penjualan

# Kita mengganti teks model, transmisi, dan bahan bakar dengan ID-
nya masing-masing

fact_sales = df.merge(dim_model, on='model') \

        .merge(dim_transmission, on='transmission') \

        .merge(dim_fueltype, on='fueltype')

# Menambahkan sale_id sebagai Primary Key unik untuk setiap
transaksi

fact_sales = fact_sales.reset_index(drop=True)

fact_sales['sale_id'] = fact_sales.index + 1

# Memilih kolom yang diperlukan saja (menggunakan ID sebagai Foreign
Key)

fact_sales = fact_sales[['sale_id', 'model_id', 'year', 'price',
'transmission_id', 'mileage', 'fueltype_id', 'tax', 'mpg',
'enginesize']]

fact_sales.head()
```

	sale_id	model_id	year	price	transmission_id	mileage	fueltype_id	tax	mpg	enginesize
0	1	1	2016	16000	1	24089	1	265	36.2	2.0
1	2	1	2017	15995	1	18615	1	145	36.2	2.0
2	3	1	2015	13998	1	27469	1	265	36.2	2.0
3	4	1	2017	18998	1	14736	1	150	36.2	2.0
4	5	1	2017	17498	1	36284	1	145	36.2	2.0

3.2.3 Load

Tahap load merupakan tahap akhir dalam proses ETL, yaitu proses pemuatan data yang telah melalui tahap transform ke dalam struktur yang siap dianalisis. Data yang telah dimuat selanjutnya digunakan untuk :

- Penerapan query relasi antar tabel
- Pelaksanaan analisis OLAP menggunakan fungsi agregasi
- Penyusunan laporan hasil analisis data dalam format PDF menggunakan Python

BAB IV

HASIL & PEMBAHASAN

4.1 Pembuatan dan Konfigurasi Database PostgreSQL

Setelah proses data cleaning dan normalisasi data selesai dilakukan menggunakan Python, tahap selanjutnya adalah melakukan implementasi database menggunakan PostgreSQL (psql). PostgreSQL digunakan sebagai *Relational Database Management System (RDBMS)* untuk menyimpan data hasil normalisasi dan mendukung proses query relasi serta agregasi data.

Pembuatan database ini bertujuan untuk menyediakan media penyimpanan terpusat yang nantinya akan disinkronkan dengan Python dalam proses analisis OLAP dan pembuatan laporan dalam format PDF. Dengan menggunakan PostgreSQL, data dapat dikelola secara terstruktur dan efisien untuk mendukung proses analisis.

4.1.1 Pembuatan Database

Pada tahap ini, dilakukan pembuatan database baru yang akan digunakan sebagai basis penyimpanan tabel dimensi dan tabel fakta. Database dibuat melalui *psql* dengan perintah sebagai berikut:

```
CREATE DATABSE transaksi_toyota;
```

Database tersebut selanjutnya digunakan untuk menyimpan seluruh tabel hasil normalisasi, yaitu tabel dimensi dan tabel fakta penjualan. Dan pastikan databse berhasil untuk di create dengan hasil digambar.

4.1.2 Input Data Normalisasi Pada Postgresql

Dengan kode berikut :

```
DB_USER = 'postgres'
DB_PASSWORD = '165001'
DB_HOST = 'localhost'
DB_PORT = '5432'
DB_NAME = 'transaksi_toyota'

connection_string =
f"postgresql://{DB_USER}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}
"

engine = create_engine(connection_string)

def push_to_postgresql():
```

```
# Daftar file hasil normalisasi
files_to_upload = {

  'dim_model.csv': 'dim_model',
  'dim_transmission.csv': 'dim_transmission',
  'dim_fueltype.csv': 'dim_fueltype',
  'fact_sales.csv': 'fact_sales'

  }

  try:

     print("--- Memulai proses upload data ---")
     for file_name, table_name in files_to_upload.items():
     if os.path.exists(file_name):

     print(f"Mengunggah {file_name} ke tabel '{table_name}'...")


     # Membaca CSV menggunakan Pandas

         df = pd.read_csv(file_name)

     # Memasukkan ke PostgreSQL

        # if_exists='replace' akan membuat tabel baru atau menimpa yang lama

       df.to_sql(table_name, engine, if_exists='replace', index=False)

        print(f"Berhasil mengunggah {table_name}.")
        else:
        print(f"Peringatan: File {file_name} tidak ditemukan.")

      print("\n[SUKSES] Semua data berhasil dimasukkan ke PostgreSQL!")

  except Exception as e:

      print(f"\n[ERROR] Terjadi kesalahan: {e}")

     if __name___ == "__main__":

         push_to_postgresql()
```

Adapun penjelasan dari kode tersebut yaitu :

- Konfigurasi koneksi

```
DB_USER = 'postgres'
DB_PASSWORD = '165001'
DB_HOST = 'localhost'
DB_PORT = '5432'
DB_NAME = 'transaksi_toyota'

connection_string = f"postgresql://{DB_USER}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}"
engine = create_engine(connection_string)
```

Variabel DB: Mendefinisikan kredensial database seperti username, password, host (lokasi server), dan nama database (transaksi_toyota).

Connection String: Ini adalah format standar URL yang digunakan SQLAlchemy untuk mengenali jenis database (PostgreSQL) dan kredensialnya.

Engine: create_engine adalah "pintu gerbang" yang menghubungkan skrip Python Anda dengan database PostgreSQL.

- Struktur Data (Star Schema)

```
def push_to_postgresql():
    # Daftar file hasil normalisasi
    files_to_upload = {
        'dim_model.csv': 'dim_model',
        'dim_transmission.csv': 'dim_transmission',
        'dim_fueltype.csv': 'dim_fueltype',
        'fact_sales.csv': 'fact_sales'
    }
```

Bagian ini menggunakan pendekatan Star Schema (Skema Bintang) dalam data warehousing:

Dimension Tables (dim_): Berisi data deskriptif (nama model, jenis transmisi, jenis bahan bakar).

Fact Table (fact_sales): Berisi data transaksi utama yang biasanya menghubungkan semua tabel dimensi tadi.

- Proses Iterasi dan Pengecekan

```
for file_name, table_name in files_to_upload.items():
    if os.path.exists(file_name):
        print(f"Mengunggah {file_name} ke tabel '{table_name}'...")

        # Membaca CSV menggunakan Pandas
        df = pd.read_csv(file_name)
```

Kode melakukan looping (perulangan) untuk setiap file yang ada di daftar. os.path.exists: Langkah pengamanan untuk memastikan file CSV benar-benar ada di folder sebelum mencoba membacanya. Jika file tidak ada, program tidak akan langsung crash tapi memberikan peringatan.

- Upload ke PostgreSQL

```
df.to_sql(table_name, engine, if_exists='replace', index=False)
```

to_sql: Fungsi dari library Pandas untuk mengirim data langsung ke database.

if_exists='replace': Parameter ini sangat penting. Artinya, jika tabel sudah ada di database, tabel tersebut akan dihapus dan dibuat ulang dengan data yang baru.

index=False: Mencegah Pandas membuat kolom tambahan untuk nomor baris (index) di dalam database.

4.1.3 Menambahkan PrimaryKey & Foreignkey Pada Tabel

Kode menambahkan *primarykey :*

```
ALTER TABLE dim_model ADD PRIMARY KEY (model_id);

ALTER TABLE dim_transmission ADD PRIMARY KEY (transmission_id);

ALTER TABLE dim_fueltype ADD PRIMARY KEY (fueltype_id);

ALTER TABLE fact_sales ADD PRIMARY KEY (sale_id);
```

Tujuan: Menetapkan satu kolom sebagai identitas unik untuk setiap baris di dalam tabel.
Fungsi:

- Menjamin tidak ada data ganda (duplikat) pada kolom tersebut.
- Memastikan kolom tersebut tidak boleh bernilai kosong (NOT NULL).
- Mempercepat proses pencarian data (indexing) oleh database.

Kode menambahkan *foreignkey:*

```
ALTER TABLE fact_sales
ADD CONSTRAINT fk_model FOREIGN KEY (model_id) REFERENCES
dim_model(model_id),
ADD CONSTRAINT fk_transmission FOREIGN KEY (transmission_id)
REFERENCES dim_transmission(transmission_id),
ADD CONSTRAINT fk_fueltype FOREIGN KEY (fueltype_id) REFERENCES
dim_fueltype(fueltype_id);
```

Penjelasan:

- Perintah ini menghubungkan tabel fakta (fact_sales) dengan tabel-tabel dimensi. Inilah inti dari database relasional.
- ADD CONSTRAINT: Memberikan nama pada aturan hubungan tersebut (misal: fk_model). Nama ini berguna jika suatu saat ingin menghapus atau mengubah relasi tersebut.
- FOREIGN KEY (kolom_di_fact): Menentukan kolom di tabel fakta yang akan menjadi "perwakilan" dari tabel lain.
- REFERENCES tabel_tujuan(kolom_tujuan): Menentukan tabel dan kolom induk mana yang menjadi referensi.

4.1.3 Mengecek Struktur Tabel

Setelah proses diatas selesai, langkah selanjutnya untuk tahap analisis lebih lanjut yaitu mengecek struktur tabel.

- Mengecek struktur tabel pada tabel dim_model;

```
transaksi_toyota=# \d dim_model;
              Table "public.dim_model"
  Column  |  Type  | Collation | Nullable | Default
----------+--------+-----------+----------+---------
 model_id | bigint |           | not null |
 model    | text   |           |          |
Indexes:
    "dim_model_pkey" PRIMARY KEY, btree (model_id)
Referenced by:
    TABLE "fact_sales" CONSTRAINT "fk_model" FOREIGN KEY (model_id) REFERENCES dim_model(model_id)
```

- Mengecek struktur tabel pada tabel typefuel;

```
transaksi_toyota=# \d dim_typefuel;
Did not find any relation named "dim_typefuel".
transaksi_toyota=# \d dim_fueltype;
             Table "public.dim_fueltype"
   Column    |  Type  | Collation | Nullable | Default
-------------+--------+-----------+----------+---------
 fueltype_id | bigint |           | not null |
 fueltype    | text   |           |          |
Indexes:
    "dim_fueltype_pkey" PRIMARY KEY, btree (fueltype_id)
Referenced by:
    TABLE "fact_sales" CONSTRAINT "fk_fueltype" FOREIGN KEY (fueltype_id) REFERENCES dim_fueltype(fueltype_id)
```

- Tabel transmission;

```
transaksi_toyota=# \d dim_transmission;
            Table "public.dim_transmission"
    Column      |  Type  | Collation | Nullable | Default
----------------+--------+-----------+----------+---------
 transmission_id | bigint |           | not null |
 transmission    | text   |           |          |
Indexes:
    "dim_transmission_pkey" PRIMARY KEY, btree (transmission_id)
Referenced by:
    TABLE "fact_sales" CONSTRAINT "fk_transmission" FOREIGN KEY (transmission_id) REFERENCES dim_transmission(transmission_id)
```

- Tabel fact sales;

```
transaksi_toyota=# \d fact_sales;
                     Table "public.fact_sales"
     Column      |       Type       | Collation | Nullable | Default
-----------------+------------------+-----------+----------+---------
 sale_id         | bigint           |           | not null |
 model_id        | bigint           |           |          |
 year            | bigint           |           |          |
 price           | bigint           |           |          |
 transmission_id | bigint           |           |          |
 mileage         | bigint           |           |          |
 fueltype_id     | bigint           |           |          |
 tax             | bigint           |           |          |
 mpg             | double precision |           |          |
 enginesize      | double precision |           |          |
Indexes:
    "fact_sales_pkey" PRIMARY KEY, btree (sale_id)
Foreign-key constraints:
    "fk_fueltype" FOREIGN KEY (fueltype_id) REFERENCES dim_fueltype(fueltype_id)
    "fk_model" FOREIGN KEY (model_id) REFERENCES dim_model(model_id)
    "fk_transmission" FOREIGN KEY (transmission_id) REFERENCES dim_transmission(transmission_id)
```

4.1.4 Menampilkan Record Tabel

- Pada Tabel dim_model

```
SELECT * FROM dim_model;
```

```
transaksi_toyota=# SELECT * FROM dim_model;
 model_id |     model
----------+---------------
        1 |  GT86
        2 |  Corolla
        3 |  RAV4
        4 |  Yaris
        5 |  Auris
        6 |  Aygo
        7 |  C-HR
        8 |  Prius
        9 |  Avensis
       10 |  Verso
       11 |  Hilux
       12 |  PROACE VERSO
       13 |  Land Cruiser
       14 |  Camry
       15 |  Verso-S
       16 |  IQ
       17 |  Urban Cruiser
(17 rows)
```

- Tabel fueltype

```
SELECT * FROM dim_fueltype;
```

```
transaksi_toyota=# SELECT * FROM dim_fueltype;
 fueltype_id | fueltype
-------------+----------
           1 | Petrol
           2 | Other
           3 | Hybrid
           4 | Diesel
(4 rows)
```

- Tabel trransmission

```
SELECT * FROM dim_transmission;
```

```
transaksi_toyota=# SELECT * FROM dim_transmission;
 transmission_id | transmission
-----------------+--------------
               1 | Manual
               2 | Automatic
               3 | Semi-Auto
(3 rows)
```

- Tabel factsales

```
SELECT * FROM fact_sales;
```

```
transaksi_toyota=# SELECT * FROM fact_sales;
 sale_id | model_id | year | price | transmission_id | mileage | fueltype_id | tax |  mpg  | enginesize
---------+----------+------+-------+-----------------+---------+-------------+-----+-------+-----------
       1 |        1 | 2016 | 16000 |               1 |   24089 |           1 | 265 |  36.2 |          2
       2 |        1 | 2017 | 15995 |               1 |   18615 |           1 | 145 |  36.2 |          2
       3 |        1 | 2015 | 13998 |               1 |   27469 |           1 | 265 |  36.2 |          2
       4 |        1 | 2017 | 18998 |               1 |   14736 |           1 | 150 |  36.2 |          2
       5 |        1 | 2017 | 17498 |               1 |   36284 |           1 | 145 |  36.2 |          2
       6 |        1 | 2017 | 15998 |               1 |   26919 |           1 | 260 |  36.2 |          2
       7 |        1 | 2017 | 18522 |               1 |   10456 |           1 | 145 |  36.2 |          2
       8 |        1 | 2017 | 18995 |               1 |   12340 |           1 | 145 |  36.2 |          2
       9 |        1 | 2016 | 13990 |               1 |   37999 |           1 | 265 |  36.2 |          2
      10 |        1 | 2017 | 17990 |               1 |   12597 |           1 | 145 |  36.2 |          2
      11 |        1 | 2017 | 16995 |               1 |   36100 |           1 | 145 |  36.2 |          2
      12 |        1 | 2019 | 23995 |               1 |     995 |           1 | 145 |  33.2 |          2
      13 |        1 | 2018 | 18498 |               1 |   35228 |           1 | 145 |  36.2 |          2
      14 |        1 | 2019 | 23980 |               1 |    1751 |           1 | 145 |  33.2 |          2
      15 |        1 | 2017 | 17995 |               1 |   16444 |           1 | 265 |  36.2 |          2
```

4.2 Agregasi Data Pada Phyton

Setelah data tersimpan dan terstruktur di dalam database PostgreSQL, tahap selanjutnya adalah melakukan agregasi data menggunakan Python. Proses agregasi ini bertujuan untuk memperoleh ringkasan informasi dari data penjualan mobil bekas Toyota yang telah melalui proses pembersihan dan normalisasi.

Agregasi data merupakan bagian penting dari analisis OLAP (Online Analytical Processing), di mana data dianalisis secara multidimensi menggunakan fungsi-fungsi agregat untuk menghasilkan insight yang mendukung pengambilan keputusan.(Agus Wikananda et al., 2025)

4.2.1 Konsep Agregasi Data

Agregasi data adalah proses pengelompokan dan perhitungan data berdasarkan dimensi tertentu dengan menggunakan fungsi agregasi seperti COUNT, SUM, AVG, MIN, dan MAX. Dalam laporan ini, agregasi dilakukan untuk menganalisis data penjualan berdasarkan atribut tertentu, seperti model kendaraan, jenis transmisi, dan tipe bahan bakar. Python digunakan sebagai alat bantu analisis karena memiliki library yang mendukung pengolahan data secara efisien dan terintegrasi dengan database PostgreSQL.

4.2.2 Prose Agregasi Data di Phyton

Proses agregasi data dilakukan dengan cara menghubungkan Python ke database PostgreSQL, kemudian mengeksekusi query SQL yang mengandung fungsi agregasi. Hasil query tersebut selanjutnya diproses menggunakan Python untuk ditampilkan dalam bentuk tabel dan digunakan sebagai bahan pembuatan laporan. Adapun agregasi yang dilakukan dalama laporan tersebut, Adalah sebagai berikut :

a. Agregasi Rata-rata Harga Mobil Berdasarkan Model
   Query :

```
query1 = """
SELECT m.model, ROUND(AVG(s.price), 2) as rata_rata_harga
FROM fact_sales s
JOIN dim_model m ON s.model_id = m.model_id
```

```
GROUP BY m.model
ORDER BY rata_rata_harga DESC;
"""
df_avg_price = pd.read_sql(query1, engine)
df_avg_price.head()
```

	model	rata_rata_harga
0	Camry	24990.00
1	PROACE VERSO	24472.00
2	Land Cruiser	23995.00
3	Corolla	20947.49
4	Hilux	20698.83
5	C-HR	19959.12
6	Prius	19166.74
7	GT86	18138.79
8	RAV4	17355.47
9	Auris	12816.74
10	Verso	12751.99
11	Avensis	11025.46
12	Yaris	10708.10
13	Aygo	7959.75
14	Verso-S	6395.00
15	IQ	4766.80
16	Urban Cruiser	4617.50

Query ini menghubungkan tabel fakta fact_sales dengan tabel dimensi dim_model melalui kolom model_id. Relasi tersebut digunakan untuk mengelompokkan data penjualan berdasarkan model kendaraan. Fungsi agregasi AVG(price) diterapkan untuk menghitung rata-rata harga mobil bekas pada setiap model.

Jenis Operasi OLAP: *Roll-Up*

b. Jumlah Unit Terjual per Tahun
Query :

```
query2 = """
SELECT year, COUNT(sale_id) as total_unit
FROM fact_sales
GROUP  BY  year
ORDER BY year DESC;
"""
df_yearly_sales = pd.read_sql(query2, engine)
df_yearly_sales.head(20)
```

	year	total_unit
0	2020	66
1	2019	1109
2	2018	996
3	2017	1990
4	2016	966
5	2015	489
6	2014	316
7	2013	170
8	2012	27
9	2011	21
10	2010	6
11	2009	14
12	2008	7
13	2007	4
14	2006	2
15	2005	1
16	2004	2
17	2002	1
18	2000	1

Query ini menggunakan tabel fact_sales tanpa relasi tambahan karena atribut tahun sudah tersedia pada tabel fakta. Fungsi agregasi COUNT(sale_id) digunakan untuk menghitung jumlah unit mobil yang terjual pada setiap tahun berdasarkan pengelompokan kolom year.

Jenis Operasi OLAP: *Roll-Up* berdasarkan dimensi waktu

c. Analisis Bahan Bakar (Mileage & Harga)

Query :

```
query3 = """
SELECT f.fueltype,
       ROUND(AVG(s.mileage), 0) as avg_mileage,
       ROUND(AVG(s.price), 2) as avg_price
FROM fact_sales s
JOIN dim_fueltype f ON s.fueltype_id = f.fueltype_id
GROUP BY f.fueltype;
"""
```

	fueltype	avg_mileage	avg_price
0	Hybrid	23849.0	16493.49
1	Other	17093.0	12174.62
2	Diesel	36621.0	13267.22
3	Petrol	18193.0	9728.61

Query ini menghubungkan tabel fact_sales dengan tabel dimensi dim_fueltype melalui fueltype_id. Relasi tersebut digunakan untuk mengelompokkan data penjualan berdasarkan jenis bahan bakar. Fungsi agregasi

AVG(mileage) dan AVG(price) digunakan untuk mengetahui rata-rata jarak tempuh dan harga mobil pada setiap tipe bahanbakar.

Jenis Operasi OLAP: *Roll-up* berdasarkan dimensi bahan bakar

d. Total Pendapatan per Transmisi
Query :

```
query4 = """
SELECT t.transmission, SUM(s.price) as total_pendapatan
FROM fact_sales s
JOIN dim_transmission t ON s.transmission_id = t.transmission_id
GROUP BY t.transmission
ORDER BY total_pendapatan DESC;
"""
```

	transmission	total_pendapatan
0	Automatic	36773560.0
1	Manual	34925671.0
2	Semi-Auto	2202318.0

Query ini mengombinasikan tabel fact_sales dengan tabel dimensi dim_transmission melalui kolom transmission_id. Fungsi agregasi SUM(price) digunakan untuk menghitung total pendapatan penjualan berdasarkan jenis transmisi. Hasilnya diurutkan untuk mengetahui transmisi dengan pendapatan tertinggi.

Jenis Operasi OLAP: *Roll-up* berdasarkan dimensi transmisi

e. Segmentasi Transmisi dan Bahan Bakar
Query :

```
query5 = """
SELECT t.transmission, f.fueltype, COUNT(s.sale_id) as jumlah_unit
FROM fact_sales s
JOIN dim_transmission t ON s.transmission_id = t.transmission_id
JOIN dim_fueltype f ON s.fueltype_id = f.fueltype_id
GROUP BY t.transmission, f.fueltype
ORDER BY jumlah_unit DESC;
"""
df_segmentation = pd.read_sql(query5, engine)
df_segmentation.head()
```

	transmission	fueltype	jumlah_unit
0	Manual	Petrol	3323
1	Automatic	Hybrid	1807
2	Automatic	Petrol	401
3	Manual	Diesel	300
4	Semi-Auto	Petrol	214

Query ini melibatkan tabel fakta fact_sales serta dua tabel dimensi, yaitu dim_transmission dan dim_fueltype. Relasi antar tabel digunakan untuk mengelompokkan data berdasarkan kombinasi jenis transmisi dan bahan bakar. Fungsi agregasi COUNT(sale_id) digunakan untuk menghitung jumlah unit pada setiap segmen.

Jenis Operasi OLAP: *Slice & Dice* (kombinasi dua dimensi)

f. Rentang Harga per Model

Query :

```
Query6 = """
SELECT m.model,
       MAX(s.price) as harga_tertinggi,
       MIN(s.price) as harga_terendah,
       (MAX(s.price) - MIN(s.price)) as rentang_harga
FROM fact_sales s
JOIN dim_model m ON s.model_id = m.model_id
GROUP BY m.model
ORDER BY rentang_harga DESC;
"""
df_price_range = pd.read_sql(query6, engine)
df_price_range.head()
```

	model	harga_tertinggi	harga_terendah	rentang_harga
0	RAV4	25000	5195	19805
1	Prius	25000	5750	19250
2	Yaris	19276	2490	16786
3	Auris	19300	3990	15310
4	Hilux	24995	11292	13703

Query ini menghubungkan fact_sales dengan dim_model melalui model_id untuk mengelompokkan data berdasarkan model kendaraan. Fungsi agregasi MAX(price) dan MIN(price) digunakan untuk mengetahui harga tertinggi dan terendah, serta selisihnya sebagai rentang harga setiap model.

Jenis Operasi OLAP: *Roll-up* berdasarkan model

g. Tren Depresiasi (Harga vs Jarak Tempuh)

Query :

```
query7 = """
SELECT year,
       ROUND(AVG(price), 2) as rata_rata_harga,
       ROUND(AVG(mileage), 0) as rata_rata_jarak_tempuh
FROM fact_sales
GROUP BY year
ORDER BY year ASC;
"""
df_depreciation = pd.read_sql(query7, engine)
df_depreciation.head()
```

	year	rata_rata_harga	rata_rata_jarak_tempuh
0	2000	2695.0	21000.0
1	2002	2695.0	21000.0
2	2004	4495.0	44615.0
3	2005	2795.0	48830.0
4	2006	2620.0	55000.0

Query ini menggunakan tabel fact_sales dan mengelompokkan data berdasarkan tahun. Fungsi agregasi AVG(price) dan AVG(mileage) digunakan untuk menganalisis tren perubahan harga dan jarak tempuh rata-rata mobil dari waktu ke waktu.

Jenis Operasi OLAP: *Roll-up* (dimensi waktu)

h. Top 5 Model Paling Irit (mpg)

Query :

```
query8 = """
SELECT
    m.model,
    ROUND(AVG(s.mpg)::numeric, 2) as rata_rata_mpg
FROM fact_sales s
JOIN dim_model m ON s.model_id = m.model_id
GROUP BY m.model
ORDER BY rata_rata_mpg DESC
LIMIT 5;
"""

with engine.connect() as conn:
    df_efficiency = pd.read_sql(query8, conn)
    print(df_efficiency)
```

	model	rata_rata_mpg
0	Prius	83.73
1	Auris	69.30
2	C-HR	66.32
3	Aygo	65.20
4	Corolla	64.83

Query ini mengombinasikan tabel fact_sales dengan tabel dim_model melalui model_id. Fungsi agregasi AVG(mpg) digunakan untuk menghitung efisiensi bahan bakar rata-rata tiap model. Data diurutkan menurun dan dibatasi lima teratas untuk menampilkan model paling irit.

Jenis Operasi OLAP: *Roll-up* + *ranking*

i. Variasi Kapasitas Mesin per Model

Query :

```
query9 = """
SELECT m.model, COUNT(DISTINCT s.enginesize) as variasi_mesin
FROM fact_sales s
JOIN dim_model m ON s.model_id = m.model_id
GROUP BY m.model
ORDER BY variasi_mesin DESC;
"""
df_depreciation = pd.read_sql(query9, engine)
df_depreciation.head(20)
```

	model	variasi_mesin
0	Auris	5
1	Avensis	4
2	Yaris	4
3	C-HR	3
4	Verso	3
5	Corolla	3
6	RAV4	3
7	Hilux	3
8	Prius	2
9	Urban Cruiser	2
10	IQ	1
11	PROACE VERSO	1
12	GT86	1
13	Aygo	1
14	Camry	1
15	Verso-S	1
16	Land Cruiser	1

Query ini menghubungkan tabel fact_sales dengan dim_model melalui model_id untuk mengelompokkan data berdasarkan model kendaraan. Fungsi agregasi COUNT(DISTINCT enginesize) digunakan untuk menghitung jumlah variasi kapasitas mesin yang dimiliki setiap model.

Jenis Operasi OLAP: *Roll-up* berdasarkan model

j. Total Penerimaan Pajak per Tahun

Query :

```
query10 = """
SELECT year, SUM(tax) as total_pajak
FROM fact_sales
GROUP  BY year
ORDER BY year DESC;
"""
df_depreciation = pd.read_sql(query10, engine)
df_depreciation.head(20)
```

	year	total_pajak
0	2020	9560.0
1	2019	159910.0
2	2018	146120.0
3	2017	179135.0
4	2016	24575.0
5	2015	13715.0
6	2014	13055.0
7	2013	11080.0
8	2012	2540.0
9	2011	1695.0
10	2010	215.0
11	2009	620.0
12	2008	1025.0
13	2007	615.0
14	2006	195.0
15	2005	145.0
16	2004	655.0
17	2002	200.0
18	2000	160.0

Query ini menggunakan tabel fact_sales dan mengelompokkan data berdasarkan tahun. Fungsi agregasi SUM(tax) digunakan untuk menghitung total penerimaan pajak kendaraan pada setiap tahun penjualan.

Jenis Operasi OLAP: *Roll-up* berdasarkan dimensi waktu

k. Jumlah Unit dengan Jarak Tempuh Tinggi (>50.000 mil)

Query :

```
query11 = """
SELECT m.model, COUNT(s.sale_id) as jumlah_unit_high_mileage
FROM fact_sales s
```

```
JOIN dim_model m ON s.model_id = m.model_id
WHERE s.mileage > 50000
GROUP BY m.model
ORDER BY jumlah_unit_high_mileage DESC;
"""

df_depreciation = pd.read_sql(query11, engine)
df_depreciation.head(20)
```

	model	jumlah_unit_high_mileage
0	Auris	68
1	Yaris	66
2	RAV4	61
3	Aygo	30
4	Avensis	18
5	Prius	17
6	Hilux	7
7	C-HR	5
8	Verso	4
9	GT86	2
10	Urban Cruiser	1
11	Land Cruiser	1
12	Verso-S	1

Query ini mengombinasikan tabel fact_sales dengan dim_model melalui model_id serta menerapkan kondisi filter pada kolom mileage. Fungsi agregasi COUNT(sale_id) digunakan untuk menghitung jumlah unit mobil dengan jarak tempuh tinggi pada setiap model.

Jenis Operasi **OLAP:** *Slice* (filter berdasarkan mileage)

l. Persentase Nilai Pasar per Model

Query :

```
query12 = """
SELECT m.model,
       SUM(s.price) as total_nilai,
       ROUND((SUM(s.price)   *   100.0   /   (SELECT   SUM(price)   FROM
fact_sales))::numeric, 2) as persentase_pasar
FROM fact_sales s
JOIN dim_model m ON s.model_id = m.model_id
GROUP BY m.model
ORDER BY total_nilai DESC;
"""

df_depreciation = pd.read_sql(query12, engine)
```

```
df_depreciation.head(20)
```

	model	total_nilai	persentase_pasar
0	Yaris	22015850.0	29.79
1	Aygo	15378245.0	20.81
2	Auris	8484681.0	11.48
3	C-HR	8482624.0	11.48
4	RAV4	6543014.0	8.85
5	Corolla	4734132.0	6.41
6	Prius	3545847.0	4.80
7	Verso	1338959.0	1.81
8	Hilux	1200532.0	1.62
9	Avensis	1014342.0	1.37
10	GT86	961356.0	1.30
11	PROACE VERSO	97888.0	0.13
12	Camry	24990.0	0.03
13	Land Cruiser	23995.0	0.03
14	IQ	23834.0	0.03
15	Urban Cruiser	18470.0	0.02
16	Verso-S	12790.0	0.02

Query ini menghubungkan tabel fact_sales dengan tabel dim_model untuk mengelompokkan data berdasarkan model kendaraan. Fungsi agregasi SUM(price) digunakan untuk menghitung total nilai penjualan setiap model, kemudian dihitung persentasenya terhadap total nilai pasar seluruh penjualan.

Jenis Operasi OLAP: *Roll-up* + perhitungan kontribusi

4.3 Penyusunan Laporan PDF Menggunakan Python

4.3.1 Tools dan Library yang Digunakan

Penyusunan laporan PDF dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan beberapa library pendukung. Library *pandas* digunakan untuk membaca dan mengelola data hasil agregasi dari database. Koneksi ke database PostgreSQL dilakukan menggunakan *SQLAlchemy*. Untuk pembuatan dokumen PDF digunakan library *reportlab*, khususnya modul *platypus* yang mendukung penyusunan elemen laporan seperti paragraf, tabel, gambar, dan pemisah halaman.

Kode :

```
import pandas as pd
from sqlalchemy import create_engine, text
from reportlab.platypus import (
    SimpleDocTemplate, Paragraph, Table, TableStyle,
    Spacer, Image, PageBreak
)
from reportlab.lib.pagesizes import A4
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib import colors
from datetime import datetime
```

4.3.2 Integrasi Data & Koneksi

Integrasi data dilakukan dengan menghubungkan Python ke database PostgreSQL yang menyimpan data warehouse penjualan mobil Toyota.

Kode :

```
engine = create_engine("postgresql://postgres:165001@localhost:5432/transaksi_toyota")
logo_path = r"C:\Users\LENOVO\Downloads\logo_uag.png"
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
output_pdf = f"Laporan_Toyota_Used_Cars_Olive_{timestamp}.pdf"
```

```
queries = [
    ("Rata-rata Harga per Model", """SELECT m.model, ROUND(AVG(s.price)::numeric, 2) as rata_rata_harga FROM fact_sales s JOIN dim_model m ON s.model_id
    ("Jumlah Unit per Tahun", """SELECT year, COUNT(sale_id) as total_unit FROM fact_sales GROUP BY year ORDER BY year DESC;"""),
    ("Analisis Bahan Bakar (Mileage & Harga)", """SELECT f.fueltype, ROUND(AVG(s.mileage)::numeric, 0) as avg_mileage, ROUND(AVG(s.price)::numeric, 2) as
    ("Total Pendapatan per Transmisi", """SELECT t.transmission, SUM(s.price) as total_pendapatan FROM fact_sales s JOIN dim_transmission t ON s.transmis
    ("Segmentasi Transmisi dan Bahan Bakar", """SELECT t.transmission, f.fueltype, COUNT(s.sale_id) as jumlah_unit FROM fact_sales s JOIN dim_transmissio
    ("Rentang Harga per Model", """SELECT m.model, MAX(s.price) as harga_tertinggi, MIN(s.price) as harga_terendah, (MAX(s.price) - MIN(s.price)) as rent
    ("Tren Depresiasi (Harga vs Jarak Tempuh)", """SELECT year, ROUND(AVG(price)::numeric, 2) as rata_rata_harga, ROUND(AVG(mileage)::numeric, 0) as rata
    ("Top 5 Model Paling Irit (MPG)", """SELECT m.model, ROUND(AVG(s.mpg)::numeric, 2) as rata_rata_mpg FROM fact_sales s JOIN dim_model m ON s.model_id
    ("Variasi Kapasitas Mesin per Model", """SELECT m.model, COUNT(DISTINCT s.enginesize) as variasi_mesin FROM fact_sales s JOIN dim_model m ON s.model_
    ("Total Penerimaan Pajak per Tahun", """SELECT year, SUM(tax) as total_pajak FROM fact_sales GROUP BY year ORDER BY year DESC;"""),
    ("Jumlah Unit High Mileage (>50rb mil)", """SELECT m.model, COUNT(s.sale_id) as jumlah_unit_high_mileage FROM fact_sales s JOIN dim_model m ON s.mode
    ("Persentase Nilai Pasar per Model", """SELECT m.model, SUM(s.price) as total_nilai, ROUND((SUM(s.price) * 100.0 / (SELECT SUM(price) FROM fact_sales
]
```

4.3.3 Proses Pembuatan Laporan dan Struktur PDF

Proses pembuatan laporan PDF dimulai dengan konfigurasi dokumen, meliputi ukuran halaman, margin, serta pengaturan gaya teks. Laporan disusun secara otomatis dengan menambahkan elemen-elemen laporan seperti halaman sampul, judul analisis, dan tabel hasil agregasi. Struktur laporan PDF disusun secara sistematis yang diawali dengan halaman sampul berisi judul laporan dan informasi waktu pencetakan. Setelah itu, laporan menampilkan bagian isi yang terdiri dari beberapa tabel hasil agregasi data.

Kode :

```
doc = SimpleDocTemplate(output_pdf, pagesize=A4, leftMargin=30, rightMargin=30, topMargin=70, bottomMargin=40)
styles = getSampleStyleSheet()
elements = []

title_style = ParagraphStyle('TitleStyle', parent=styles['Title'], alignment=1, fontSize=22, spaceAfter=10)
subtitle_style = ParagraphStyle('SubTitle', parent=styles['Heading2'], alignment=1, fontSize=14, spaceAfter=30, textColor=colors.darkgrey)

# --- HALAMAN COVER ---
elements.append(Spacer(1, 150))
elements.append(Paragraph("<b>LAPORAN PENJUALAN MOBIL TOYOTA BEKAS</b>", title_style))
elements.append(Paragraph("Analisis Data Warehouse (Tema Olive & Grey)", subtitle_style))
elements.append(Spacer(1, 20))
elements.append(Paragraph(f"Dicetak pada: {datetime.now().strftime('%d %B %Y')}", ParagraphStyle('Date', parent=styles['Normal'], alignment=1)))
elements.append(PageBreak())
```

```
with engine.connect() as conn:
    for i, (title, sql) in enumerate(queries, 1):
        elements.append(Paragraph(f"<b>{i}. {title}</b>", styles['Heading2']))
        elements.append(Spacer(1, 12))

        df = pd.read_sql(text(sql), conn)
        data = [df.columns.tolist()] + df.values.tolist()

        # Lebar tabel compact
        col_width = 115

        t = Table(data, colWidths=[col_width]*len(df.columns), hAlign='CENTER', repeatRows=1)
```

```
        # Style Tabel: Olive & Abu-abu
        t.setStyle(TableStyle([
            # Header: Olive Tua (OliveDrab)
            ('BACKGROUND', (0,0), (-1,0), colors.HexColor("#6B8E23")),
            ('TEXTCOLOR', (0,0), (-1,0), colors.whitesmoke),
            ('ALIGN', (0,0), (-1,-1), 'CENTER'),
            ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
            ('FONTSIZE', (0,0), (-1,0), 10),
            ('FONTSIZE', (0,1), (-1,-1), 8),
            ('GRID', (0,0), (-1,-1), 0.5, colors.grey),
            ('ROWBACKGROUNDS', (0, 1), (-1, -1), [colors.HexColor("#F4F7ED"), colors.HexColor("#E9E9E9")]),
            ('VALIGN', (0,0), (-1,-1), 'MIDDLE'),
            ('BOTTOMPADDING', (0,0), (-1,-1), 5),
            ('TOPPADDING', (0,0), (-1,-1), 5),
        ]))

        elements.append(t)
        elements.append(Spacer(1, 25))

        if i % 2 == 0:
            elements.append(PageBreak())

doc.build(elements, onFirstPage=header_footer, onLaterPages=header_footer)
print(f"✅ SUKSES: {output_pdf}")
```

4.4.4 Penyisipan Logo Universitas (UAG)

Sebagai identitas institusi, logo Universitas Ary Ginanjar (UAG) disisipkan pada bagian kanan atas setiap halaman laporan. Penyisipan logo dilakukan melalui pengaturan *header* dokumen menggunakan library *reportlab*. Selain logo, ditambahkan pula nomor halaman pada bagian footer untuk meningkatkan kerapihan.

Kode :

```
def header_footer(canvas, doc):
    canvas.saveState()
    try:
        logo_width = 45
        logo_x = A4[0] - 30 - logo_width
        canvas.drawImage(logo_path, logo_x, A4[1] - 55, width=logo_width, height=45, preserveAspectRatio=True)
    except:
        canvas.drawRightString(A4[0] - 30, A4[1] - 40, "[Logo Missing]")

    canvas.setFont('Helvetica', 8)
    canvas.drawRightString(A4[0] - 30, 20, f"Halaman {doc.page}")
    canvas.restoreState()
```

Adapun hasil dari “Laporan PDF” yang dibuat ini dapat dilihat pada bagian lampiran.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan dan implementasi yang telah dilakukan, dapat disimpulkan bahwa proses pengolahan data melalui tahapan *Extract, Transform, Load* (ETL) berhasil menghasilkan data yang bersih, terstruktur, dan siap dianalisis. Proses *cleaning*, normalisasi, serta pembentukan skema data warehouse memungkinkan data operasional diolah menjadi informasi yang lebih bernilai. Penerapan agregasi data dan analisis *Online Analytical Processing* (OLAP) menggunakan Python dan PostgreSQL mampu menyajikan informasi ringkas dan analitis terkait karakteristik penjualan mobil, pola harga, tren tahunan, serta segmentasi berdasarkan atribut kendaraan. Proses ini membantu dalam memahami data secara multidimensi dan mendukung analisis pengambilan keputusan.

Selain itu, penyusunan laporan dalam bentuk PDF secara otomatis melalui Python berhasil mengintegrasikan hasil analisis ke dalam format dokumentasi yang rapi dan konsisten. Dengan demikian, sistem yang dibangun tidak hanya berfungsi sebagai alat analisis data, tetapi juga sebagai sarana pelaporan yang siap digunakan secara akademik maupun praktis.

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah memberikan dukungan dan kontribusi dalam penyusunan laporan ini. Ucapan terima kasih secara khusus disampaikan kepada dosen pengampu mata kuliah yang telah memberikan bimbingan dan arahan selama proses pengerjaan. Selain itu, penulis juga mengapresiasi pihak-pihak lain yang secara langsung maupun tidak langsung telah membantu sehingga laporan ini dapat diselesaikan dengan baik.

5.2 Saran

Berdasarkan hasil perancangan dan analisis data warehouse yang telah dilakukan, maka saran yang dapat diberikan adalah sebagai berikut:

1. Pengembangan Dimensi Data
   Penelitian selanjutnya disarankan untuk menambahkan dimensi data lain, seperti lokasi penjualan, kondisi kendaraan, serta waktu transaksi yang lebih detail. Penambahan dimensi tersebut diharapkan dapat memperkaya struktur data warehouse sehingga analisis yang dihasilkan menjadi lebih mendalam dan komprehensif.
2. Peningkatan Analisis OLAP
   Analisis OLAP dapat dikembangkan lebih lanjut dengan menerapkan operasi drill-down, roll-up, dan slice secara lebih optimal. Dengan demikian, proses eksplorasi data dapat dilakukan pada berbagai tingkat detail untuk mendukung pengambilan keputusan yang lebih akurat.

DAFTAR PUSTAKA

Agus Wikananda, K., Raka, M., Habibi, N., & Wijaya, A. (2025). Jurnal Sains dan Teknologi Perancangan Data Warehouse Penjualan Minimarket XYZ Menggunakan Kimball Modeling dan Analisis OLAP. *Jurnal Sains Dan Teknologi*, *2*(2), 103–110.

Franciska Mey Dina, D. (2022). Normalisasi Database Rancangan Sistem Penyewaan Buku Berbayar. In *Jurnal Ilmiah Computing Insight* (Vol. 4, Issue 1).

Gusti Ngurah Agung Trisna Putra, I., Nyoman Aditya Mahendra, I., Made Suwija Putra, I., Studi Teknologi Informasi, P., Teknik, F., Udayana, U., Raya Kampus Unud Jimbaran, J., & Kuta Sel, K. (n.d.). Putra, Implementasi ETL Data Warehouse Dengan Konsep Fitur Metadata Dan Cleansing Data Pada Toko Kue IMPLEMENTASI ETL DATA WAREHOUSE DENGAN KONSEP FITUR METADATA DAN CLEANSING DATA. In *Jurnal Sistem Informasi* (Vol. 9, Issue 2).

LAPORAN PENJUALAN MOBIL TOYOTA BEKAS

Analisis Data Warehouse

Dicetak pada: 06 January 2026

1. Rata-rata Harga per Model

model	rata_rata_harga
Camry	24990.0
PROACE VERSO	24472.0
Land Cruiser	23995.0
Corolla	20947.49
Hilux	20698.83
C-HR	19959.12
Prius	19166.74
GT86	18138.79
RAV4	17355.47
Auris	12816.74
Verso	12751.99
Avensis	11025.46
Yaris	10708.1
Aygo	7959.75
Verso-S	6395.0
IQ	4766.8
Urban Cruiser	4617.5

2. Jumlah Unit per Tahun

year	total_unit
2020	66
2019	1109
2018	996
2017	1990
2016	966
2015	489
2014	316
2013	170
2012	27
2011	21
2010	6
2009	14
2008	7
2007	4
2006	2
2005	1
2004	2
2002	1
2000	1

3. Analisis Bahan Bakar (Mileage & Harga)

fueltype	avg_mileage	avg_price
Hybrid	23849.0	16493.49
Other	17093.0	12174.62
Diesel	36621.0	13267.22
Petrol	18193.0	9728.61

4. Total Pendapatan per Transmisi

transmission	total_pendapatan
Automatic	36773560.0
Manual	34925671.0
Semi-Auto	2202318.0

5. Segmentasi Transmisi dan Bahan Bakar

transmission	fueltype	jumlah_unit
Manual	Petrol	3323
Automatic	Hybrid	1807
Automatic	Petrol	401
Manual	Diesel	300
Semi-Auto	Petrol	214
Automatic	Other	88
Automatic	Diesel	47
Manual	Other	4
Manual	Hybrid	3
Semi-Auto	Diesel	1

6. Rentang Harga per Model

model	harga_tertinggi	harga_terendah	rentang_harga
RAV4	25000	5195	19805
Prius	25000	5750	19250
Yaris	19276	2490	16786
Auris	19300	3990	15310
Hilux	24995	11292	13703
GT86	24995	11575	13420
C-HR	25000	11995	13005
Avensis	16495	3495	13000
Aygo	15000	2999	12001
Verso	17995	5995	12000
Corolla	25000	15291	9709
IQ	5995	3495	2500
Verso-S	6995	5795	1200
PROACE VERSO	24990	23950	1040
Urban Cruiser	4995	3995	1000
Land Cruiser	23995	23995	0
Camry	24990	24990	0

7. Tren Depresiasi (Harga vs Jarak Tempuh)

year	rata_rata_harga	rata_rata_jarak_tempuh
2000.0	2695.0	21000.0
2002.0	2695.0	21000.0
2004.0	4495.0	44615.0
2005.0	2795.0	48830.0
2006.0	2620.0	55000.0
2007.0	3467.75	50083.0
2008.0	4304.71	38933.0
2009.0	4573.14	41484.0
2010.0	4879.83	40830.0
2011.0	6048.29	41237.0
2012.0	6841.37	41486.0
2013.0	8384.19	39634.0
2014.0	8841.33	36754.0
2015.0	9658.59	32646.0
2016.0	11653.04	26701.0
2017.0	12090.45	21638.0
2018.0	12204.97	15758.0
2019.0	14452.04	5912.0
2020.0	14963.5	2020.0

8. Top 5 Model Paling Irit (MPG)

model	rata_rata_mpg
Prius	83.73
Auris	69.3
C-HR	66.32
Aygo	65.2
Corolla	64.83

9. Variasi Kapasitas Mesin per Model

model	variasi_mesin
Auris	5
Avensis	4
Yaris	4
C-HR	3
Verso	3
Corolla	3
RAV4	3
Hilux	3
Prius	2
Urban Cruiser	2
IQ	1
PROACE VERSO	1
GT86	1
Aygo	1
Camry	1
Verso-S	1
Land Cruiser	1

10. Total Penerimaan Pajak per Tahun

year	total_pajak
2020.0	9560.0
2019.0	159910.0
2018.0	146120.0
2017.0	179135.0
2016.0	24575.0
2015.0	13715.0
2014.0	13055.0
2013.0	11080.0
2012.0	2540.0
2011.0	1695.0
2010.0	215.0
2009.0	620.0
2008.0	1025.0
2007.0	615.0
2006.0	195.0
2005.0	145.0
2004.0	655.0
2002.0	200.0
2000.0	160.0

11. Jumlah Unit High Mileage (>50rb mil)

model	jumlah_unit_high_mileage
Auris	68
Yaris	66
RAV4	61
Aygo	30
Avensis	18
Prius	17
Hilux	7
C-HR	5
Verso	4
GT86	2
Urban Cruiser	1
Land Cruiser	1
Verso-S	1

12. Persentase Nilai Pasar per Model

model	total_nilai	persentase_pasar
Yaris	22015850.0	29.79
Aygo	15378245.0	20.81
Auris	8484681.0	11.48
C-HR	8482624.0	11.48
RAV4	6543014.0	8.85
Corolla	4734132.0	6.41
Prius	3545847.0	4.8
Verso	1338959.0	1.81
Hilux	1200532.0	1.62
Avensis	1014342.0	1.37
GT86	961356.0	1.3
PROACE VERSO	97888.0	0.13
Camry	24990.0	0.03
Land Cruiser	23995.0	0.03
IQ	23834.0	0.03
Urban Cruiser	18470.0	0.02
Verso-S	12790.0	0.02