# Data Warehouse Project

**submitted to**

Prof. Wesam Ahmed

**Faculty of Computer Science**
**Hurghada University**

**2nd January 2026**

# Table of Contents

# 1. Introduction

## 1.1 Project Overview

This project focuses on designing and implementing a Data Warehouse for retail sales analysis. The main objective is to transform operational sales data into a dimensional model that supports analytical queries and interactive dashboards. A Star Schema was selected to simplify reporting and improve query performance.

## 1.2 Team Members

- Hend Ahmed Haroun (Team Leader)
- Hala Mohammed Maher
- Samar El-Ameer Mohammed
- Mohrael John Wageeh
- Sameer Abbas Mahmoud

## 1.3 Business Objectives

- Build ETL pipeline for data cleaning
- Design and implement Star Schema
- Create interactive Power BI dashboards
- Generate business insights from sales data

## 1.4 System Requirements

The system is designed to analyze retail sales data across products, customers, regions, time, payment methods, and delivery status. The main fact identified is sales transactions, while dimensions include Product, Customer, Date, Region, Payment, and Delivery.

## 1.5 Technologies Used

- Python: Pandas, numpy, pyodbc, sqlalchemy (Data Extracting, Cleaning and Loading).
- SQL Server: Database and Data Warehouse Management.
- Power BI: Data Visualization.

### 1.6 Data Source

- Dataset: Retail Sales & Customer Analytics.
- Source: https://www.kaggle.com/datasets/ajinkyachintawar/sales-and-customer-behaviour-insights

# 2. Dimensional Model Design

### 2.1 Facts and Dimensions Description

The data warehouse follows a Star Schema design with a central Fact_Sales table connected to multiple dimension tables. This structure simplifies queries and improves reporting efficiency. The Fact_Sales table stores measurable business data such as quantity, price, and total sales amount, while the dimension tables provide descriptive context for analysis.
The model consists of a central Fact_Sales table connected to multiple dimension tables:
Dim_Customer, Dim_Product, Dim_Date, Dim_Region, Dim_Payment, Dim_Delivery.
Each dimension is linked to the fact table using foreign keys
**Dimension Tables:**

**1-Dim_Customer:**

customer_id (PK)
email
gender
region
loyalty_tier
signup_date

**2-Dim_Product:**

product_id (PK)
product_name
category
base_price
supplier_code
launch_date

**3-Dim_Date:**

date_id (PK)
full_date
day
day_of_week
month
year

**4-Dim_Region:**

region_id (PK)
region_name

**5-Dim_Payment:**

payment_method_id (PK)
payment_method

**6-Dim_Delivery:**

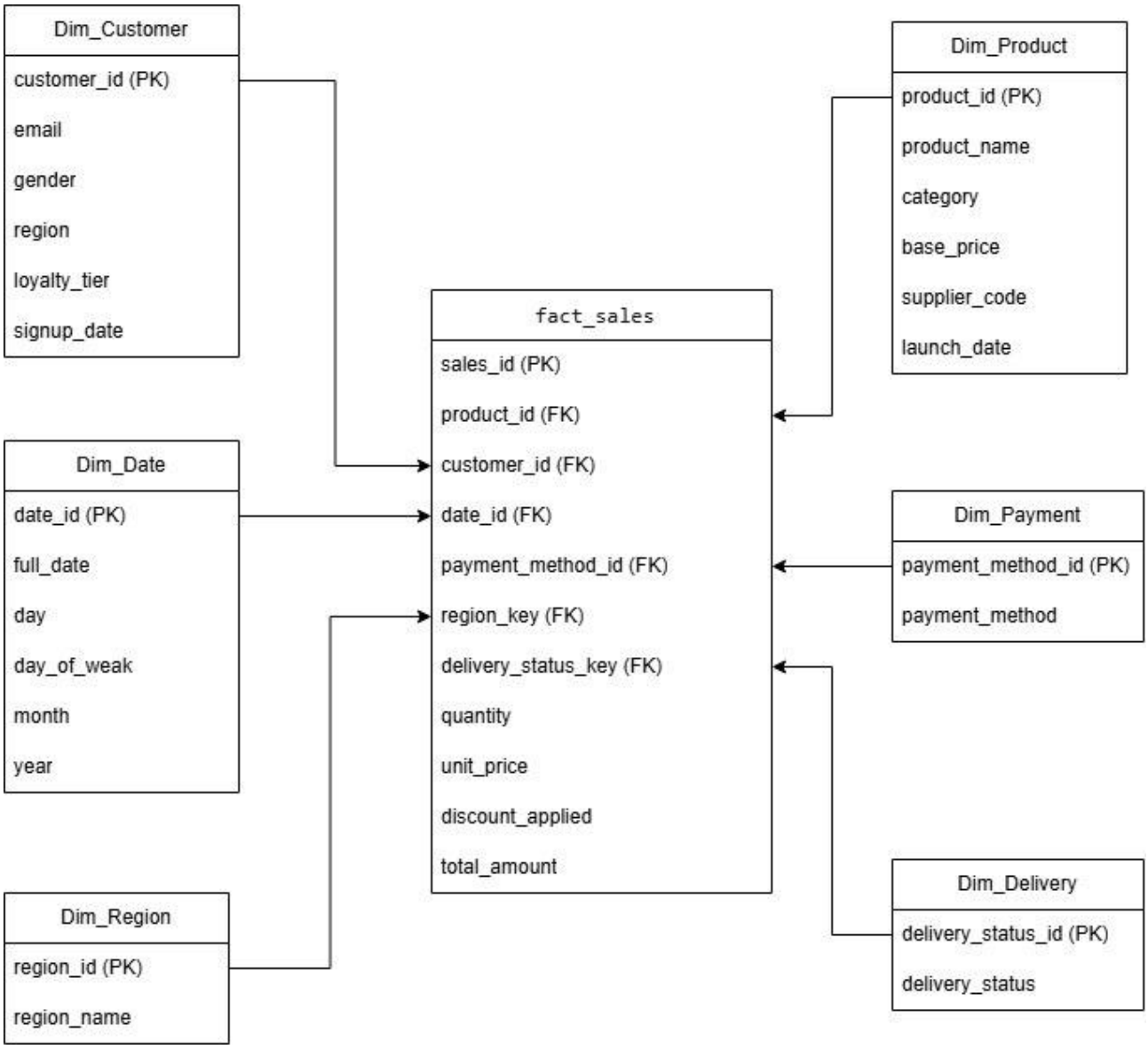
delivery_status_id (PK)
delivery_status

**7-Fact_Sales:**

sales_id (PK)
product_id (FK)
customer_id (FK)
date_id (FK)
payment_method_id (FK)
region_key (FK)
delivery_status_key (FK)
quantity
unit_price
discount_applied
total_amount

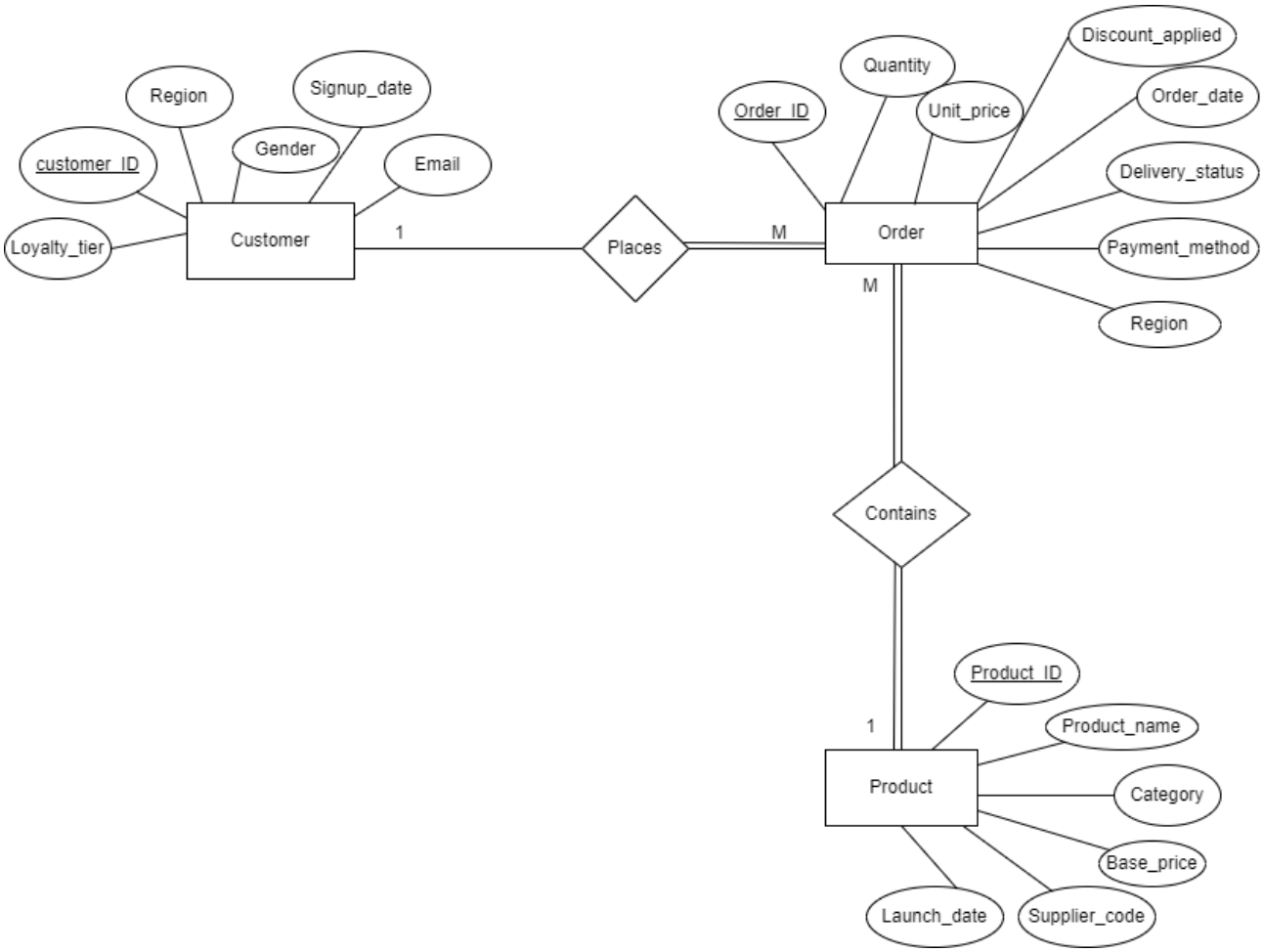
The Fact_Sales table stores measurable business data and links to all dimensions.

## 2.2 Star Schema Diagram



## 2.3 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) represents the logical structure of the Data Warehouse and shows the relationships between the fact table and dimension tables. It illustrates how the Fact_Sales table is connected to the dimension tables through foreign keys




# 3. Data Warehouse Implementation

## 3.1 ETL Process

- Data cleaning and preprocessing using Python.
- Handling missing values and data inconsistencies.
- Transforming data into analytical-friendly formats.

The data warehouse uses three CSV files obtained from a Kaggle dataset as the primary data sources to support analytical reporting.

These CSV files represent operational retail data and contain sales transactions, customer information, and product details.

The three CSV files include:
**Sales CSV file**: Used to store transactional sales data such as quantity, unit price, discounts, and total sales amount.
**Customers CSV file**: Contains customer-related information including customer attributes and behavioral data.
**Products CSV file**: Contains product information such as product name, category, and pricing details.

### 3.1.1 Extract Phase

In the extraction phase, data is collected from the three CSV files obtained from Kaggle.
All extracted data is first loaded into a staging area, which serves as an intermediate storage layer to validate and prepare the data before transformation.

### 3.1.2 Transform Phase

During the transformation phase, data is cleaned and standardized to ensure consistency and quality.

The following operations were performed:

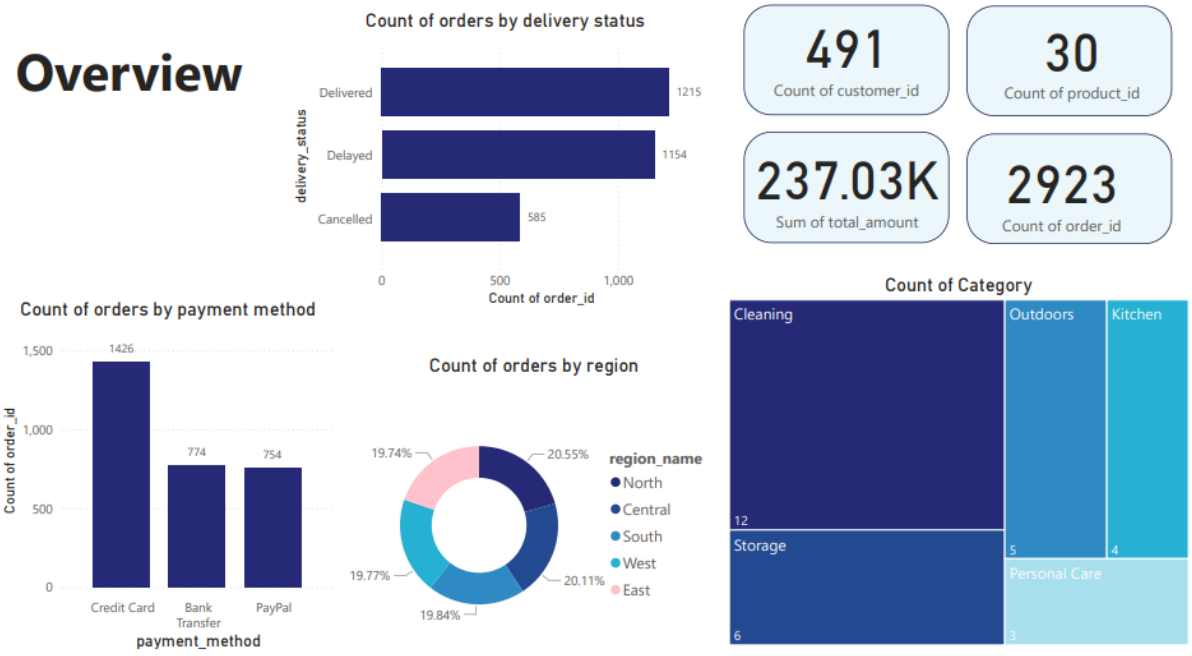
- Removing duplicate records.
- Handling missing or null values.
- Standardizing date formats.
- Converting categorical values (e.g., gender, payment type).
- Generating surrogate keys for dimension tables.
- Ensuring referential integrity between fact and dimension tables.

### 3.1.3 Load Phase

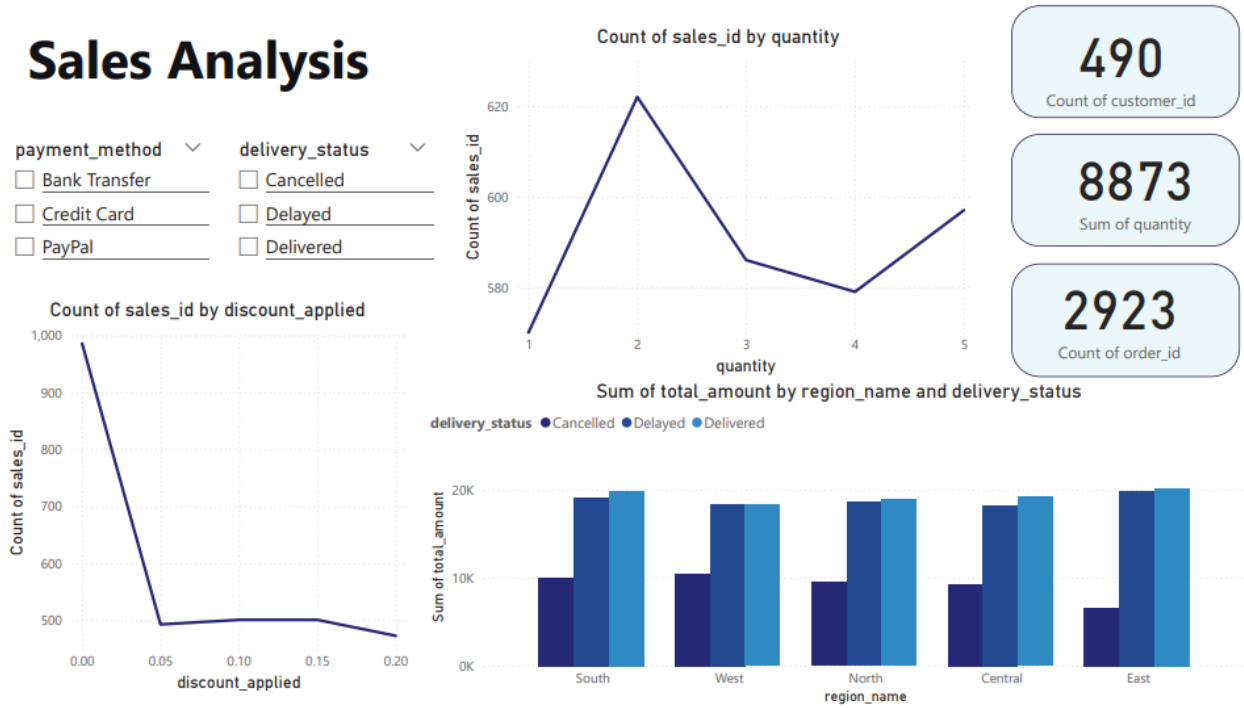
In the loading phase, the transformed data is loaded into the Data Warehouse.
Dimension tables are loaded first, followed by the Fact_Sales table.
Foreign keys are assigned to link fact records with their corresponding dimension records.

# 4. Visualization & Dashboard

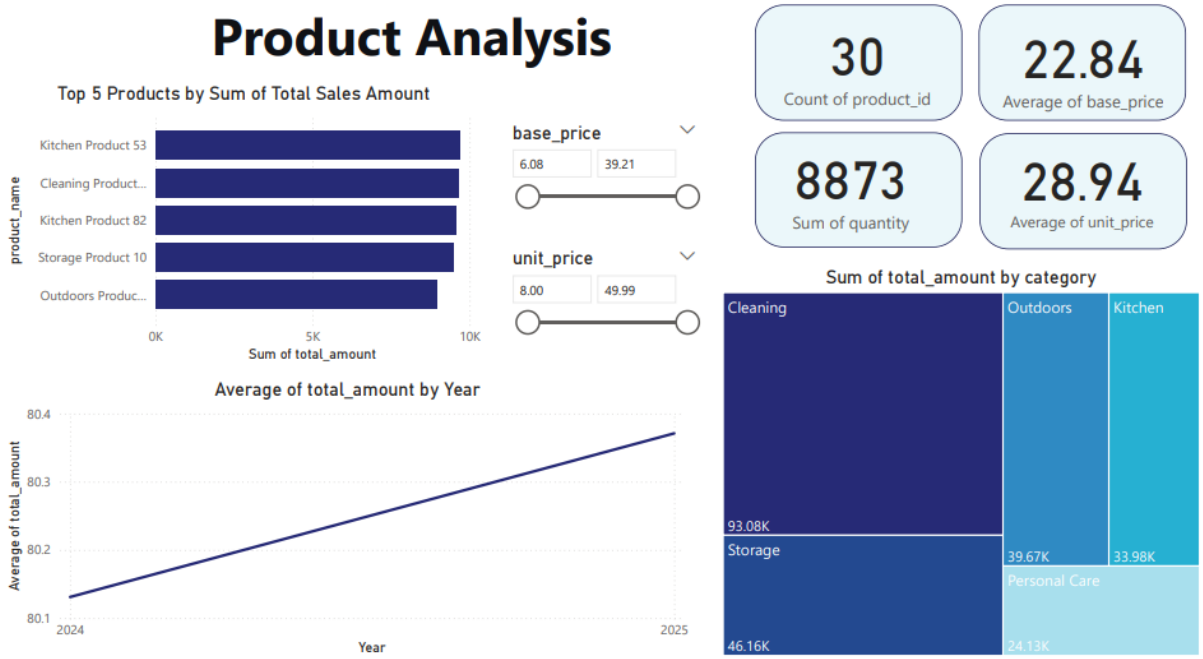
## 4.1 Overview Dashboard



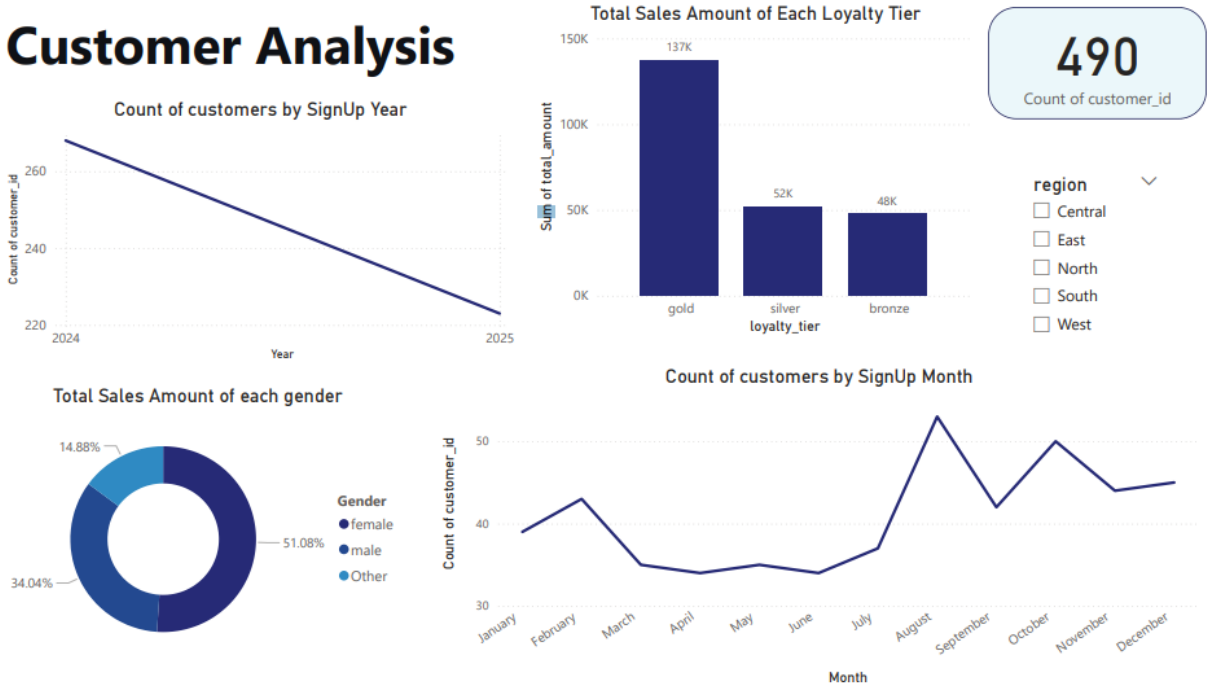

## 4.2 Sales Dashboard



## 4.3 Product Dashboard




## 4.4 Customer Dashboard



### 4.5 Key Visualization

The dashboards include the following key visualizations:

- Total Sales KPI.
- Top Selling Products.
- Sales by Region.
- Sales Trends over Time.
- Payment Method Distribution.

These dashboards allow users to analyze data dynamically and gain actionable insights.

### 4.6 Tools Used

The Data Warehouse was connected to *Power BI* to build interactive dashboards that support decision-making and data exploration.

## 5. Challenges & Solutions

### 5.1 Challenges Encountered

During the implementation of the Data Warehouse, several challenges were encountered:

- Inconsistent data formats across different sources.
- Missing and null values.
- Complex joins between multiple tables.

### 5.2 Solutions Implemented

The following solutions were applied to address these challenges:

- Data cleaning and preprocessing using Python.
- Use of surrogate keys in dimension tables.
- Adoption of a Star Schema design to simplify joins and improve query performance.

# 6. Results & Recommendation

## 6.1 Results Achieved

The implemented Data Warehouse enables efficient data analysis and reporting. It supports informed decision-making and provides a scalable foundation for future extensions.

## 6.2 Key Insights & Analysis

- Total revenue reached 237K from 2,923 orders, with an average order value of around 80.
- Sales are evenly distributed across regions, each contributing close to 20% of total orders.
- Cleaning is the top-performing category, generating the highest revenue.
- The dataset includes 490 customers, with Gold being the dominant loyalty tier.
- Most orders are successfully delivered, and Credit Card is the most used payment method.
- The majority of purchases involve 1–3 items, with limited reliance on high discounts.

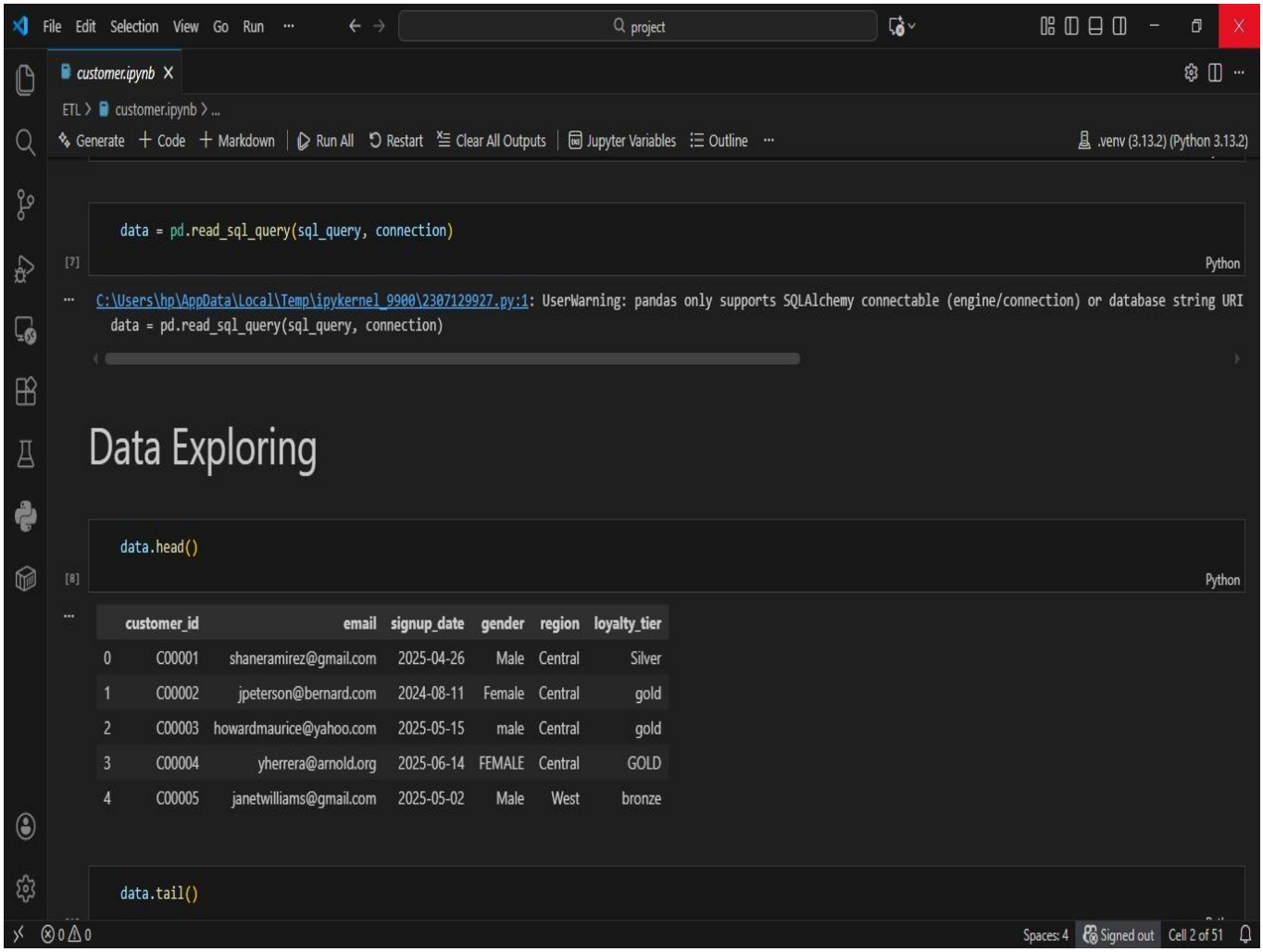
## 6.3 Future Recommendations

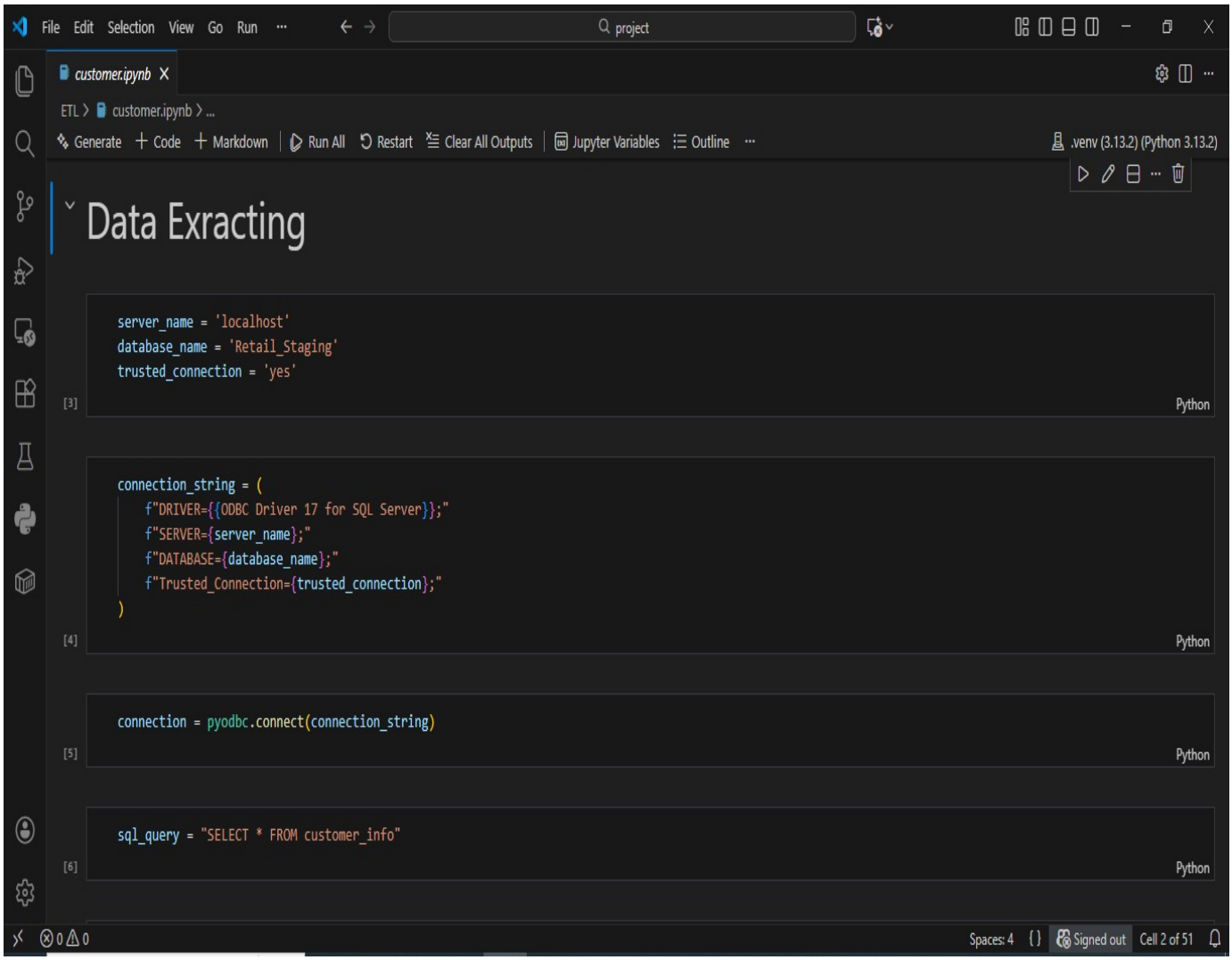
To further enhance the system, the following improvements are recommended:

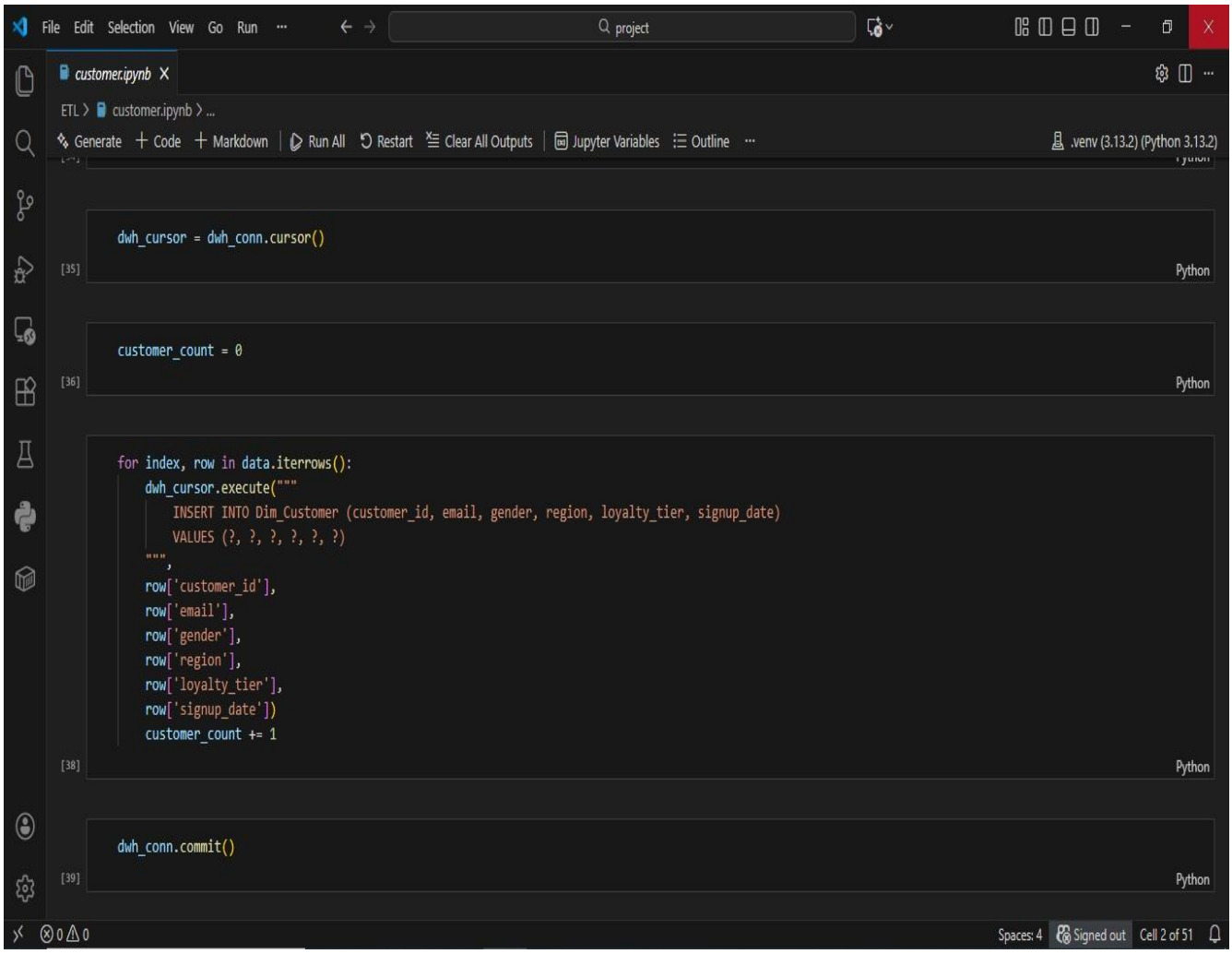
- Adding customer segmentation analysis.
- Automating ETL pipelines.
- Including real-time data updates.

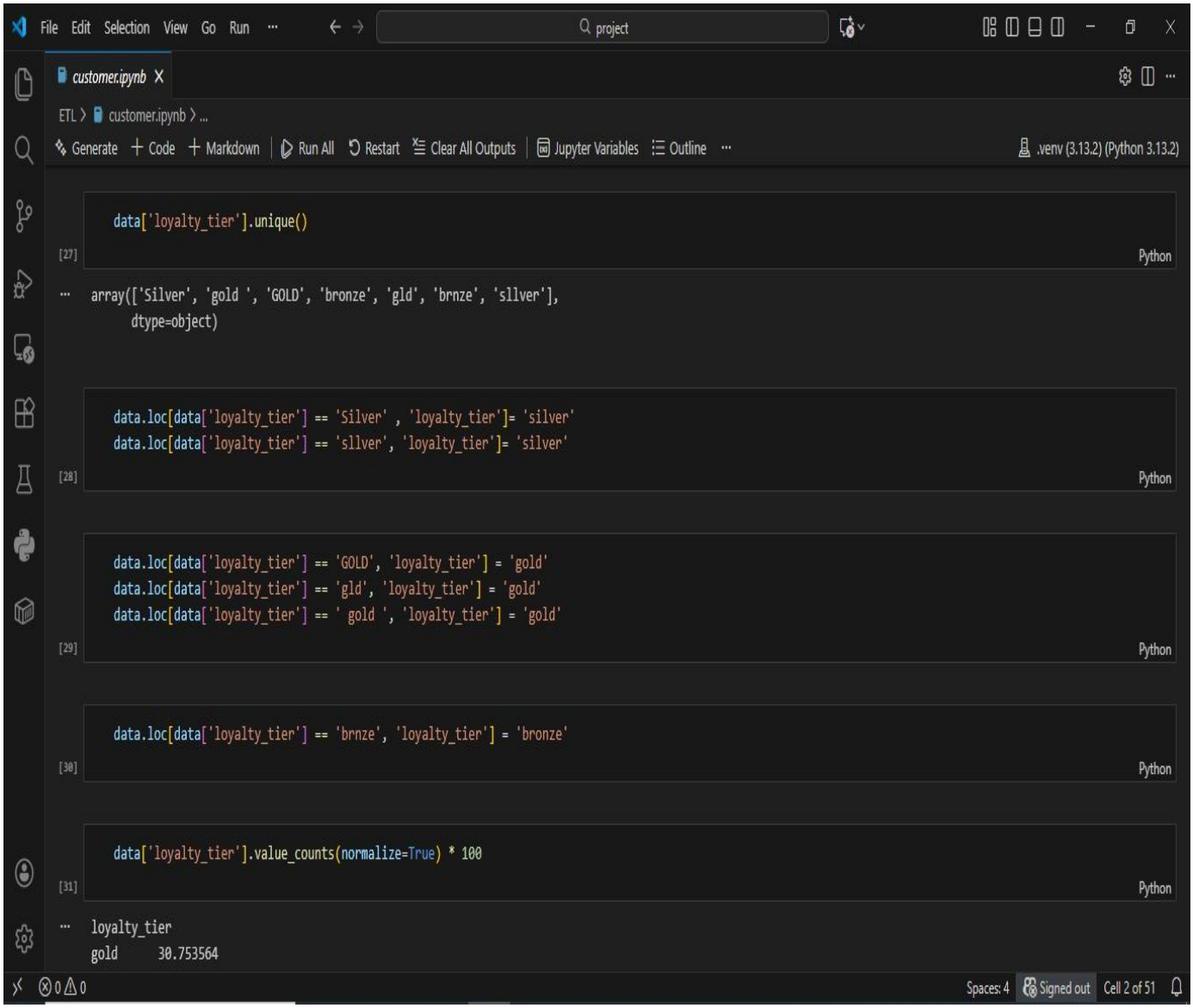
# 7. Conclusion

This project demonstrates the complete lifecycle of a Data Warehousing solution, from requirements analysis and dimensional modeling to ETL implementation and data visualization.
The system provides a scalable and analytical foundation for generating meaningful retail sales insights.
These dashboards allow users to analyze data dynamically and gain actionable insights.


File Edit Selection View Go Run ···
project
customer.ipynb
ETL > customer.ipynb > ...
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ···
.venv (3.13.2) (Python 3.13.2)
data = pd.read_sql_query(sql_query, connection)
[7]
Python
C:\Users\hp\AppData\Local\Temp\ipykernel_9900\2307129927.py:1: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI
data = pd.read_sql_query(sql_query, connection)
Data Exploring
data.head()
[8]
Python
customer_id email signup_date gender region loyalty_tier
0 C00001 shaneramirez@gmail.com 2025-04-26 Male Central Silver
1 C00002 jpeterson@bernard.com 2024-08-11 Female Central gold
2 C00003 howardmaurice@yahoo.com 2025-05-15 male Central gold
3 C00004 yherrera@arnold.org 2025-06-14 FEMALE Central GOLD
4 C00005 janetwilliams@gmail.com 2025-05-02 Male West bronze
data.tail()
Spaces: 4
Signed out
Cell 2 of 51


File Edit Selection View Go Run ···
project
customer.ipynb
ETL > customer.ipynb > ...
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ···
.venv (3.13.2) (Python 3.13.2)
Data Exracting
server_name = 'localhost'
database_name = 'Retail_Staging'
trusted_connection = 'yes'
[3]
Python
connection_string = (
f"DRIVER={{ODBC Driver 17 for SQL Server}};"
f"SERVER={server_name};"
f"DATABASE={database_name};"
f"Trusted_Connection={trusted_connection};"
)
[4]
Python
connection = pyodbc.connect(connection_string)
[5]
Python
sql_query = "SELECT * FROM customer_info"
[6]
Python
0 0
Spaces: 4 { } Signed out Cell 2 of 51


File Edit Selection View Go Run
project
customer.ipynb
ETL > customer.ipynb > ...
Generate Code Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
.venv (3.13.2) (Python 3.13.2)
dwh_cursor = dwh_conn.cursor()
[35]
Python
customer_count = 0
[36]
Python
for index, row in data.iterrows():
dwh_cursor.execute("""
INSERT INTO Dim_Customer (customer_id, email, gender, region, loyalty_tier, signup_date)
VALUES (?, ?, ?, ?, ?, ?)
""",
row['customer_id'],
row['email'],
row['gender'],
row['region'],
row['loyalty_tier'],
row['signup_date'])
customer_count += 1
[38]
Python
dwh_conn.commit()
[39]
Python
0 0
Spaces: 4
Signed out
Cell 2 of 51


File Edit Selection View Go Run
project
customer.ipynb
ETL > customer.ipynb > ...
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
.venv (3.13.2) (Python 3.13.2)
[27]
data['loyalty_tier'].unique()
Python
array(['Silver', 'gold ', 'GOLD', 'bronze', 'gld', 'brnze', 'sllver'],
dtype=object)
[28]
data.loc[data['loyalty_tier'] == 'Silver' , 'loyalty_tier']= 'silver'
data.loc[data['loyalty_tier'] == 'sllver', 'loyalty_tier']= 'silver'
Python
[29]
data.loc[data['loyalty_tier'] == 'GOLD', 'loyalty_tier'] = 'gold'
data.loc[data['loyalty_tier'] == 'gld', 'loyalty_tier'] = 'gold'
data.loc[data['loyalty_tier'] == ' gold ', 'loyalty_tier'] = 'gold'
Python
[30]
data.loc[data['loyalty_tier'] == 'brnze', 'loyalty_tier'] = 'bronze'
Python
[31]
data['loyalty_tier'].value_counts(normalize=True) * 100
Python
loyalty_tier
gold 30.753564
Spaces: 4 Signed out Cell 2 of 51

# Dashboard_Video.mp4

Dashboard_Video.mp4