Data Warehouse (DWH) and ETL Pipeline Project

Project Overview

This document outlines the design, implementation, and analysis of a Data Warehouse (DWH) project. The project focuses on integrating sales, customer behavior, and delivery datasets into a comprehensive DWH, supported by an ETL pipeline built with Python, and visualizing the data using Power BI.

Business Objectives

The main objectives of this project are:
- To create a centralized Data Warehouse for sales, customer behavior, and delivery data.
- To implement an ETL pipeline for data extraction, transformation, and loading into the DWH.
- To develop interactive and insightful Power BI dashboards for data analysis and visualization.
- To support business decision-making with comprehensive data insights.

Project Architecture

The project architecture consists of the following components:
- Data Sources: Various CSV files and databases containing sales, customer behavior, and delivery data.
- ETL Pipeline: A Python-based ETL process to extract, transform, and load data into the DWH.
- Data Warehouse: A PostgreSQL database designed to store integrated and processed data.

- Data Visualization: Power BI dashboards for analyzing and visualizing the data.

Tools and Technologies:

- ETL Tool: Python
- Database: PostgreSQL
- Visualization Tool: Power BI

Data cleaning :

The point of this process is to drop missing values or fill in empty cells to get a more complete dataset with fewer errors

Code :

```
import pandas as pd
import os



#customer behavior
file_path= ''
df=pd.read_csv(file_path)




df.dropna( inplace=True)

df.drop(columns=['Search_Result_Exploration',
'Customer_Reviews_Importance', 'Add_to_Cart_Browsing',
                 'Cart_Completion_Frequency', 'Cart_Abandonment_Factors',
'Saveforlater_Frequency',
                 'Review_Left', 'Review_Reliability',
'Review_Helpfulness',
                 'Personalized_Recommendation_Frequency',
'Recommendation_Helpfulness',
```

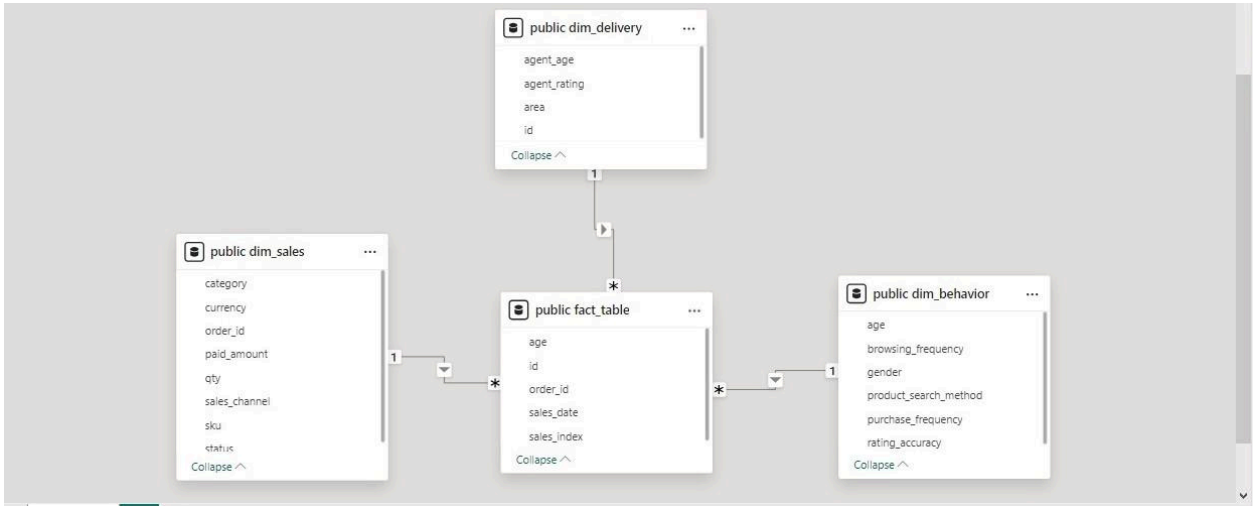
```
                'Shopping_Satisfaction', 'Service_Appreciation',
'Improvement_Areas'], inplace=True)


customer_behavior_dtype_conversions = {
    'index': 'int64',
    'Order ID': 'int64',
    'Date': 'datetime64[ns]',
    'Status': 'category',
    'Fulfilment': 'category',
    'Sales Channel': 'category',
    'SKU': 'category',
    'Category': 'category',
    'Size': 'category',
    'ASIN': 'category',
    'Qty': 'int64',
    'currency': 'category',
    'Amount': 'float64'
}

for column, dtype in customer_behavior_dtype_conversions.items():
    if column in df.columns:
        try:
            df[column] = df[column].astype(dtype)
        except ValueError as e:
            print(f"Error converting column {column} to {dtype}: {e}")



print(df.dtypes)

print(df)

os.makedirs('' , exist_ok=True)
cleaned_file_path = ''
df.to_csv(cleaned_file_path, index=False)
print(f"Cleaned dataset saved to {cleaned_file_path}")
```

```
#sales
sales_path= ''
df=pd.read_csv(sales_path)
print(df.dtypes)



df.dropna( inplace=True)

df.drop (columns=['ship-service-level','Style','Courier
Status','ship-city','ship-state','ship-postal-code',
        'ship-country','promotion-ids','B2B','fulfilled-by','Unnamed:
22'], inplace=True)

os.makedirs('' , exist_ok=True)
cleaned_file = ''
df.to_csv(cleaned_file, index=False)
print(f"Cleaned dataset saved to {cleaned_file}")

#delivery
delivery_path= ''
df=pd.read_csv(delivery_path)
print(df.dtypes)

df.dropna( inplace=True)

df.drop(columns=
['Store_Latitude','Store_Longitude','Drop_Latitude','Drop_Longitude','Pick
up_Time','Weather'
        ,'Traffic','Vehicle','Delivery_Time'] , inplace=True)




print(df)

os.makedirs('' , exist_ok=True)
cleaned_file = ''
df.to_csv(cleaned_file, index=False)
print(f"Cleaned dataset saved to {cleaned_file}")
```

Data Warehouse Design:

Schema Design:

The DWH follows a star schema design, with fact tables connected to dimension tables.



Staging tables sql script :

```
CREATE TABLE staging_sales (
    index TEXT,
    Order_ID TEXT,
    Date timesatmp,
    Status varchar(200),
    Fulfilment  varchar(200),
    Sales_Channel varchar(200),
    SKU varchar(200),
    Category varchar(200),
    Size varchar(200),
```

```
    ASIN varchar(200),
    Qty int,
    Currency varchar(200),
    Amount numeric
);

CREATE TABLE staging_delivery (
    Order_ID varchar(200),
    Agent_Age int ,
    Agent_Rating numeric,
    Order_Date date,
    Order_Time TEXT,
    Area varchar(200),
    Category  varchar(200)
);

CREATE TABLE staging_behavior (
    Timestamp Timestamp,
    Age int ,
    Gender varchar(200),
    Purchase_Frequency varchar(200),
    Purchase_Categories varchar(200),
    Browsing_Frequency varchar(200),
    Product_Search_Method varchar(200),
    Personalized_Recommendation_Frequency  varchar(200),
    Rating_Accuracy TEXT
);
```

Fact and dim tables sql script :

```
CREATE TABLE DIM_SALES (
    order_id text PRIMARY KEY,
    status varchar(200) ,
    sales_channel varchar(200),
    sku varchar(200),
    category varchar(200),
    qty int,
    currency varchar(200),
    paid_amount numeric
);

INSERT INTO DIM_SALES (order_id, status, sales_channel, sku, category
,qty , currency , paid_amount)
SELECT order_id, status, sales_channel, sku, category ,qty , currency ,
paid_amount
FROM staging_sales
ON CONFLICT (order_id) DO NOTHING;


CREATE TABLE DIM_BEHAVIOR (
    customer_age int PRIMARY KEY,
    gender varchar(50),
    purchase_frequency varchar(500),
    browsing_frequency varchar(500),
    product_search_method varchar(500),
    rating_accuracy text
);


INSERT INTO DIM_BEHAVIOR (age ,
    gender ,
    purchase_frequency ,
    browsing_frequency ,
```

```
    product_search_method ,
    rating_accuracy )
SELECT age ,
    gender ,
    purchase_frequency ,
    browsing_frequency ,
    product_search_method ,
    rating_accuracy
FROM staging_behavior
ON CONFLICT (age) DO NOTHING;

select * from dim_behavior


CREATE TABLE DIM_DELIVERY (
    delivery_id text PRIMARY KEY,
    agent_age int,
    agent_rating numeric ,
    area varchar(200)
);


INSERT INTO DIM_DELIVERY (
    id ,
    agent_age ,
    agent_rating ,
    area )
SELECT id ,
    agent_age ,
    agent_rating ,
    area
FROM staging_delivery

select * from DIM_DELIVERY ;
```

```
CREATE TABLE fact_table (
  sales_index text PRIMARY KEY,
    order_id text REFERENCES DIM_SALES(order_id),
    sales_date text,
    id text REFERENCES DIM_DELIVERY(id),
    age int REFERENCES DIM_BEHAVIOR(age)
 );


--columns names are the same from the raw tables to the staging tables while inserting data
--making data type of columns in staging tables and dim and act tables the same would make insertion of data more smooth

INSERT INTO fact_table (
  sales_index,
  order_id,
  sales_date,
  id,
  age
)
-- Replace NULL with 'NA' (or adjust the value as needed)
SELECT
  COALESCE(staging_sales.sales_index, 'NA'),
  staging_sales.order_id,
  staging_sales.sales_date,
  staging_delivery.order_id,
  staging_behavior.age
FROM staging_sales
FULL OUTER JOIN staging_delivery
  ON staging_sales.order_id = staging_delivery.order_id
FULL OUTER JOIN staging_behavior
  ON staging_sales.category = staging_behavior.category
-- Ensure no duplicate insertions based on primary key (if necessary)
```

WHERE staging_sales.sales_index IS NOT NULL
ON CONFLICT (sales_index) DO NOTHING;

SELECT * FROM fact_table

ETL Pipeline

The ETL process involves three main steps:
1. Extract: Data is extracted from various sources.
2. Transform: Data is cleaned, transformed, and normalized.
3. Load: Data is loaded into the PostgreSQL DWH.

Code :

```
import pandas as pd
from sqlalchemy import create_engine
import logging

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Database info
DB_USER = ''
DB_PASSWORD = ''
DB_HOST = ''
DB_PORT = ''
DB_NAME = ''

# 1 - Extracting process
```

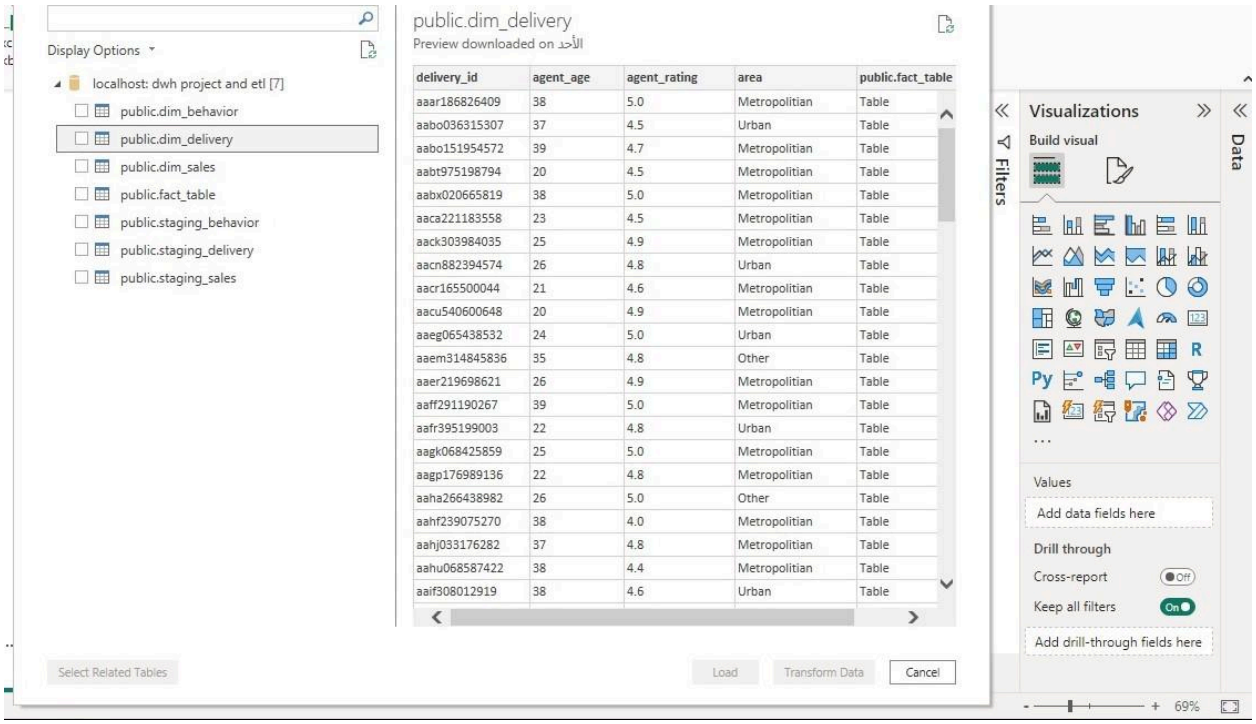
```
file_path = ''

def extract_data(file_path):
    logger.info("Extracting data from %s", file_path)
    try:
        data = pd.read_csv(file_path)
        logger.info("Data extracted successfully.")
        return data
    except Exception as e:
        logger.error("Error extracting data: %s", e)
        raise

# 2 - Transformation process
# Transforming CSV file or editing data types or cleaning

def transform_data(data):
    logger.info("Transforming data.")
    try:
        data = data.dropna()      # Remove rows with missing values
        logger.info("Data transformed successfully.")
        return data
    except Exception as e:
        logger.error("Error transforming data: %s", e)
        raise

# 3 - Loading process
# Loading data into the database

def load_data(data, table_name):
    logger.info("Loading data into %s table.", table_name)
    try:
        engine =
create_engine(f'postgresql://{DB_USER}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{
DB_NAME}')
        with engine.connect() as connection:
            data.to_sql(table_name, connection, if_exists='replace',
index=False)
        logger.info("Data loaded successfully.")
    except Exception as e:
        logger.error("Error loading data: %s", e)
```

```
        raise

def main():
    # ETL process
    file_path = ''
    table_name = 'sales_report'

    try:
        # Extract
        data = extract_data(file_path)

        # Transform
        data = transform_data(data)

        # Load
        load_data(data, table_name)

        logger.info("ETL process completed successfully.")
    except Exception as e:
        logger.error("ETL process failed: %s", e)

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

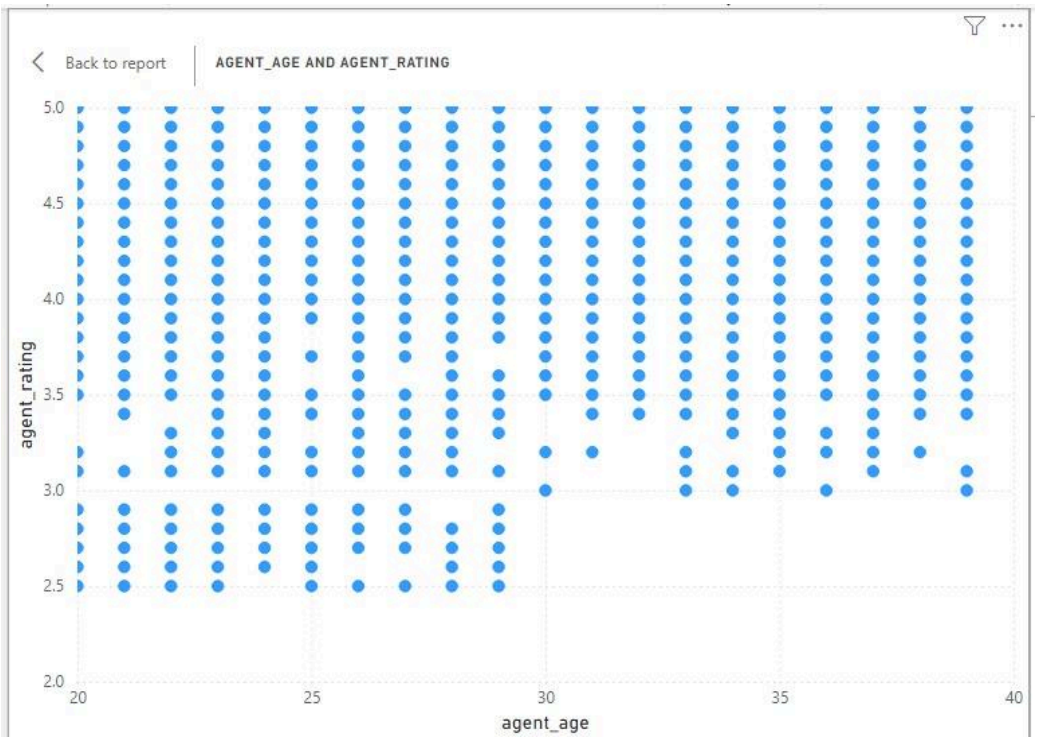
Data Visualization with Power BIe.

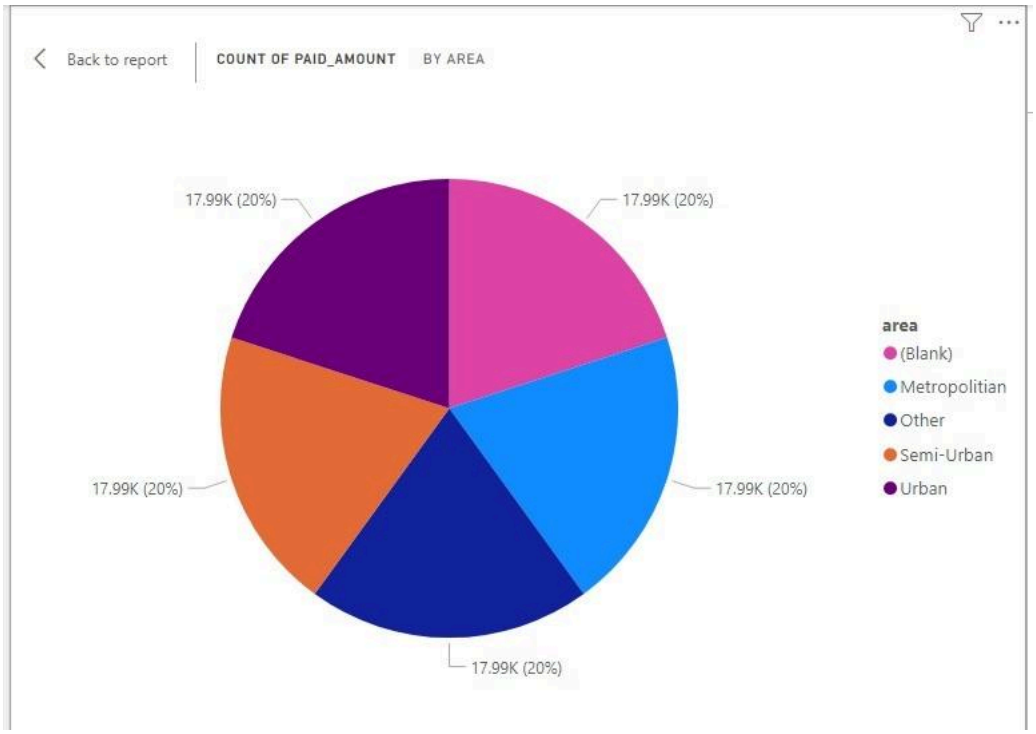
Power BI Implementation

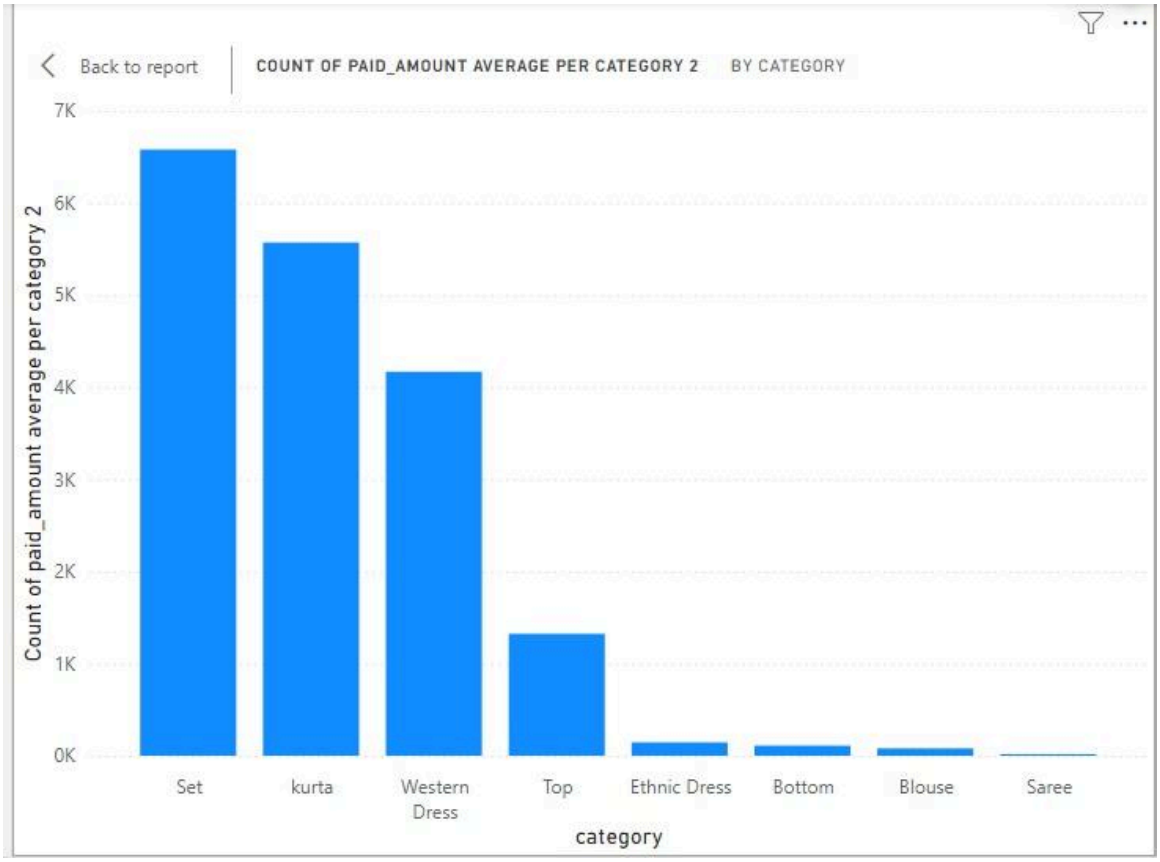
connected Power BI to the PostgreSQL DWH.

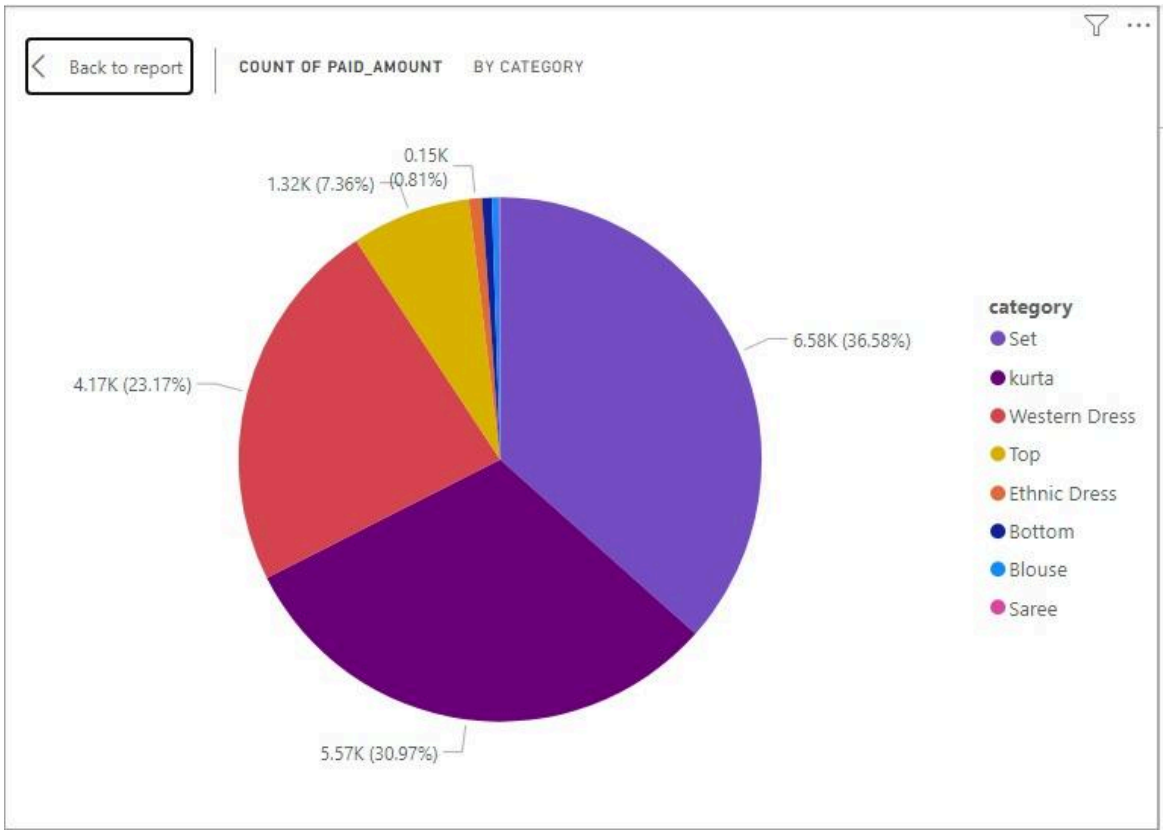


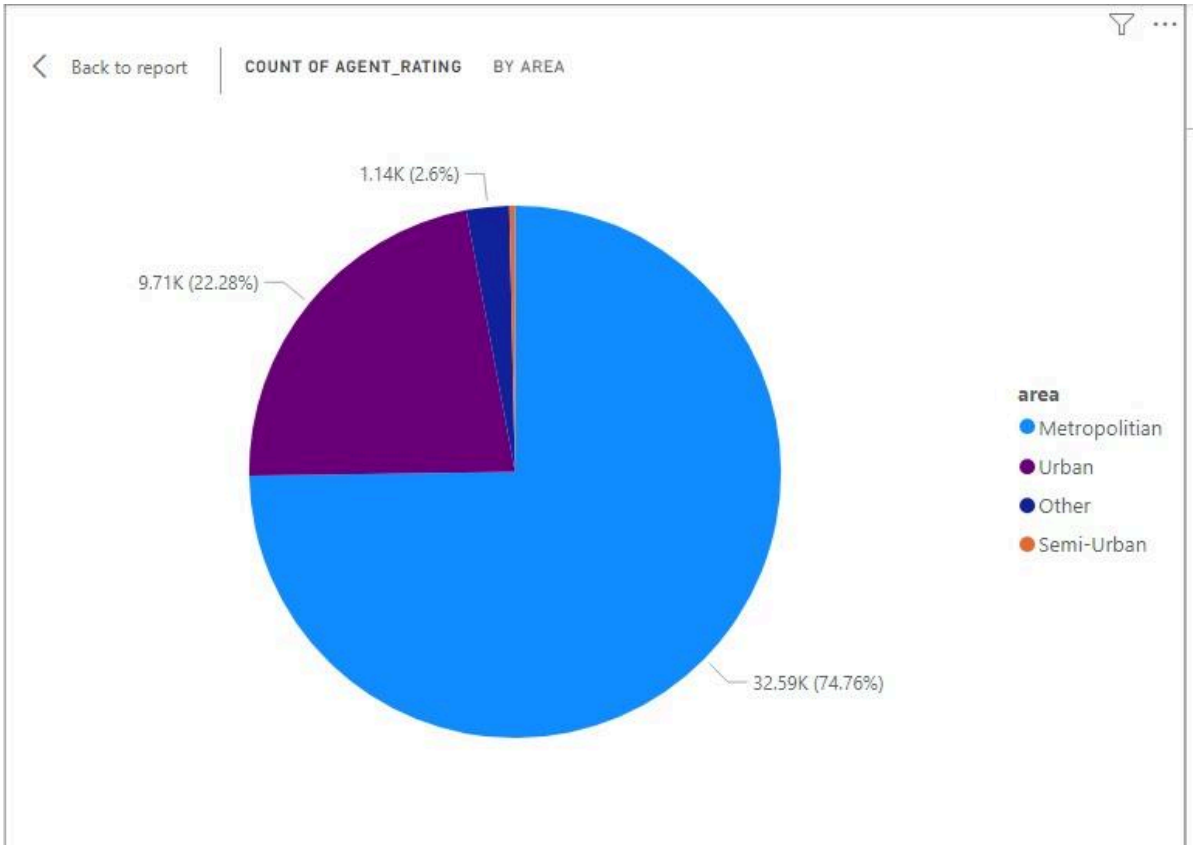
Charts :


Back to report
AGENT_AGE AND AGENT_RATING
agent_rating
5.0
4.5
4.0
3.5
3.0
2.5
2.0
20
25
30
35
40
agent_age


Back to report
COUNT OF PAID_AMOUNT
BY AREA
17.99K (20%)
17.99K (20%)
17.99K (20%)
17.99K (20%)
17.99K (20%)
area
(Blank)
Metropolitian
Other
Semi-Urban
Urban


Back to report
COUNT OF PAID_AMOUNT AVERAGE PER CATEGORY 2
BY CATEGORY
7K
6K
5K
4K
3K
2K
1K
0K
Count of paid_amount average per category 2
Set
kurta
Western Dress
Top
Ethnic Dress
Bottom
Blouse
Saree
category


Back to report
COUNT OF PAID_AMOUNT
BY CATEGORY
0.15K (0.81%)
1.32K (7.36%)
6.58K (36.58%)
4.17K (23.17%)
5.57K (30.97%)
category
Set
kurta
Western Dress
Top
Ethnic Dress
Bottom
Blouse
Saree


Back to report
COUNT OF AGENT_RATING
BY AREA
1.14K (2.6%)
9.71K (22.28%)
32.59K (74.76%)
area
Metropolitian
Urban
Other
Semi-Urban

Conclusion

The DWH and ETL pipeline project successfully integrates sales, customer behavior, and delivery datasets, providing valuable insights through interactive Power BI charts. This implementation supports data-driven decision-making and enhances business analysis capabilities.