Customer Shopping Behaviour Analysis

End-to-End Data Analytics & Business Intelligence Project

Author: Sai Aman Teppala

Email: amansaileo@gmail.com

LinkedIn: https://www.linkedin.com/in/saiamanteppala/

GitHub: https://github.com/amansai1/

Project Type: Portfolio Project

Tools: Python, PostgreSQL, Docker, SQL, VS Code, Power BI

Year: 2026

1. Project Overview

This project focuses on analysing customer shopping behaviour using a full end-to-end data analytics pipeline. The objective was to clean and transform raw customer data, store it in a relational database, perform analytical queries, and build an interactive dashboard to derive business insights.

The project demonstrates practical skills in **Python, PostgreSQL, SQL, Docker, Power BI, and data visualization**, following an industry-style workflow.

2. Problem Statement

Businesses need a deeper understanding of customer purchasing patterns, subscription behaviour, demographics, and product performance to make informed decisions. This project aims to answer key business questions such as:

- Who are the most valuable customer segments?
- Do subscribed customers spend more?
- Which products and categories generate the most revenue?
- How does age, gender, and shipping type affect sales?

3. Technology Stack

- **Programming Language:** Python
- **Data Analysis:** Pandas, NumPy
- **Database:** PostgreSQL 15
- **Database Containerization:** Docker
- **SQL Development:** Visual Studio Code (SQLTools Extension)
- **Visualization Tool:** Power BI Desktop (Windows VM on macOS)
- **Notebook Environment:** Jupyter Notebook
- **Operating System:** macOS (host), Windows 11 ARM (VMware Fusion)

4. Environment Setup

4.1 Python & Jupyter Notebook

- Anaconda distribution installed on macOS
- Jupyter Notebook used for data cleaning and transformation
- Libraries used: pandas, numpy, sqlalchemy, psycopg2

4.2 PostgreSQL using Docker

PostgreSQL was deployed using Docker to ensure a reproducible and isolated database environment.

```
docker run -d \
  --name postgres \
  -e POSTGRES_USER=admin \
  -e POSTGRES_PASSWORD=admin123 \
  -e POSTGRES_DB=customer_behavior \
  -p 5432:5432 \
  postgres:15
```

4.3 Database Connectivity

- SQLAlchemy was used to connect Python to PostgreSQL
- DataFrames were loaded directly into PostgreSQL tables

```
from sqlalchemy import create_engine

engine = create_engine(
    "postgresql+psycopg2://admin:admin123@localhost:5432/customer_behavior")
```

5. Data Cleaning & Transformation (Python)

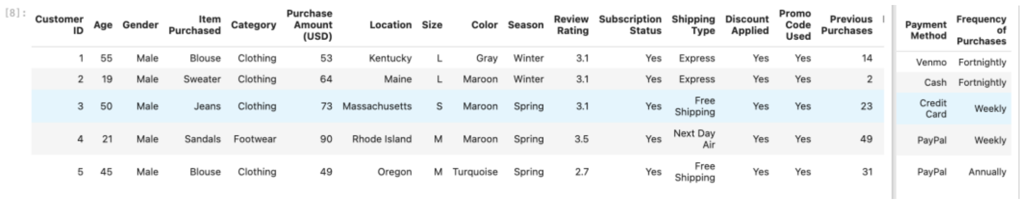
The raw dataset was cleaned and transformed in Jupyter Notebook. Key steps included:

- Handling missing values
- Correcting data types
- Creating derived columns such as age_group
- Standardizing categorical values

We began with data preparation and cleaning in Python:

Data Loading: Imported the dataset using pandas.

Initial Exploration: Used df.info() to check structure and .describe() for summary statistics.



[8]:

Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used	Previous Purchases	Payment Method	Frequency of Purchases
1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter	3.1	Yes	Express	Yes	Yes	14	Venmo	Fortnightly
2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter	3.1	Yes	Express	Yes	Yes	2	Cash	Fortnightly
3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring	3.1	Yes	Free Shipping	Yes	Yes	23	Credit Card	Weekly
4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring	3.5	Yes	Next Day Air	Yes	Yes	49	PayPal	Weekly
5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring	2.7	Yes	Free Shipping	Yes	Yes	31	PayPal	Annually

Missing Data Handling: Checked for null values and imputed missing values in the Review Rating column using the median rating of each product category.

Column Standardization: Renamed columns to **snake case** for better readability and Documentation.

Feature Engineering:

- Created **age_group** column by binning customer ages.
- Created **purchase_frequency_days** column from purchase data.

Data Consistency Check: Verified if discount_applied and promo_code_used were redundant; dropped promo_code_used.

Database Integration: Connected Python script to PostgreSQL and loaded the cleaned DataFrame into the database for SQL analysis.

After cleaning, the final DataFrame was loaded into PostgreSQL:

df.to_sql("customer", engine, if_exists="replace", index=False)

```
Data successfully loaded into table 'customer' in database 'customer_behavior'.
```

6. Database Schema

- **Database:** customer_behavior
- **Schema:** public
- **Table:** customer

Key columns include:

- customer_id
- age, age_group
- gender
- item_purchased
- category

- purchase_amount
- subscription_status
- discount_applied
- shipping_type
- review_rating
- previous_purchases

7. SQL Analysis Using VS Code

In addition to Python and Power BI, SQL was used extensively for analytical querying. All queries were written and executed using **Visual Studio Code** with the **SQLTools PostgreSQL driver**.

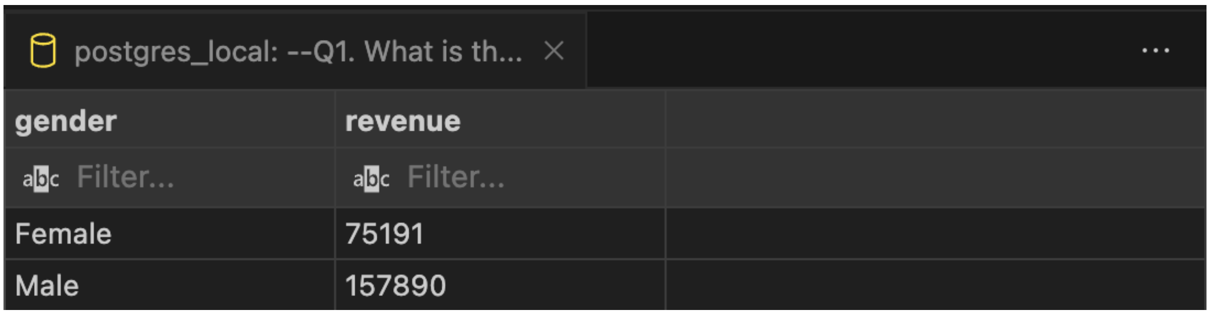
7.1 SQL Development Environment

- IDE: Visual Studio Code
- Database: PostgreSQL (Docker)
- Table: customer

7.2 Business Questions & SQL Queries

Q1. Total Revenue by Gender

```
SELECT gender, SUM(purchase_amount) AS revenue
FROM customer
GROUP BY gender;
```

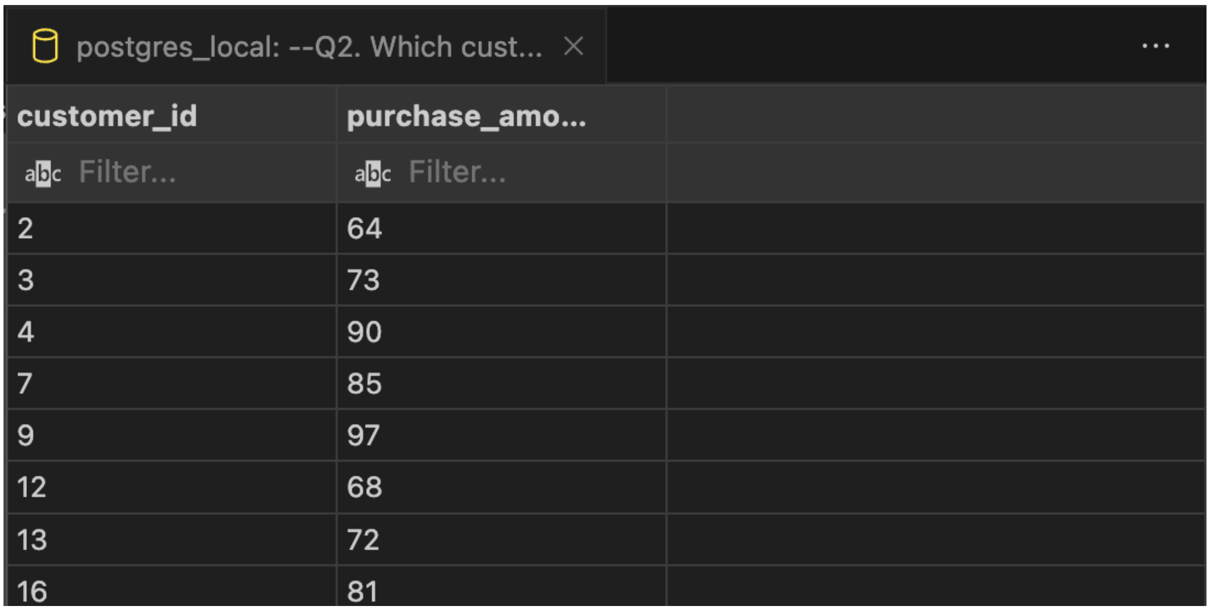


gender	revenue
Female	75191
Male	157890

Q2. Customers Using Discounts but Spending Above Average

```
SELECT customer_id, purchase_amount
FROM customer
WHERE discount_applied = 'Yes'
```

AND purchase_amount >= (SELECT AVG(purchase_amount) FROM customer);

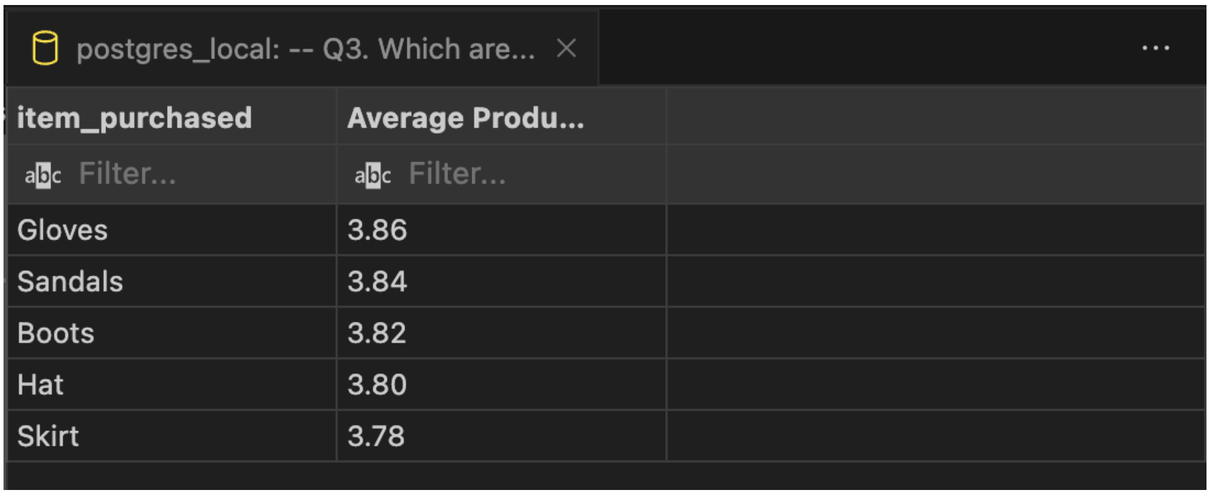


postgres_local: --Q2. Which cust...

customer_id	purchase_amo...
Filter...	Filter...
2	64
3	73
4	90
7	85
9	97
12	68
13	72
16	81

Q3. Top 5 Products by Average Review Rating

```
SELECT item_purchased,
       ROUND(AVG(review_rating::numeric), 2) AS "Average Product Rating"
FROM customer
GROUP BY item_purchased
ORDER BY AVG(review_rating) DESC
LIMIT 5;
```



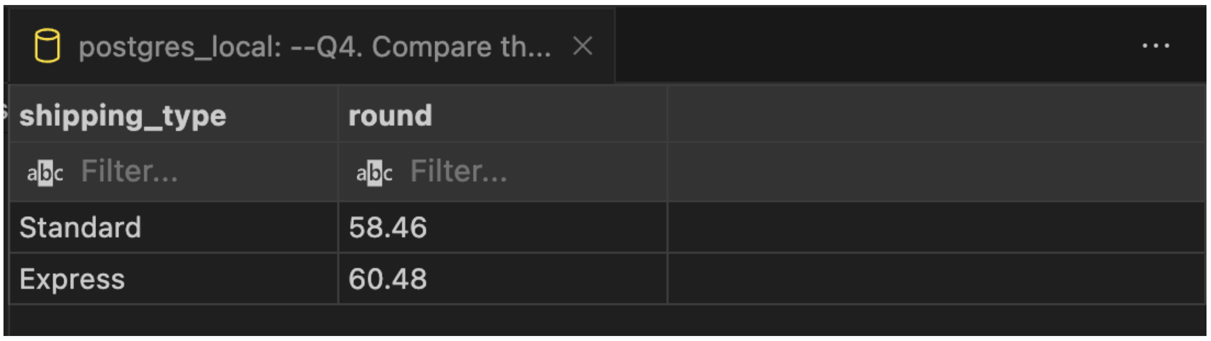
postgres_local: -- Q3. Which are...

item_purchased	Average Produ...
Filter...	Filter...
Gloves	3.86
Sandals	3.84
Boots	3.82
Hat	3.80
Skirt	3.78

Q4. Average Purchase Amount by Shipping Type

```
SELECT shipping_type,
       ROUND(AVG(purchase_amount), 2)
FROM customer
WHERE shipping_type IN ('Standard', 'Express')
```

```
GROUP BY shipping_type;
```

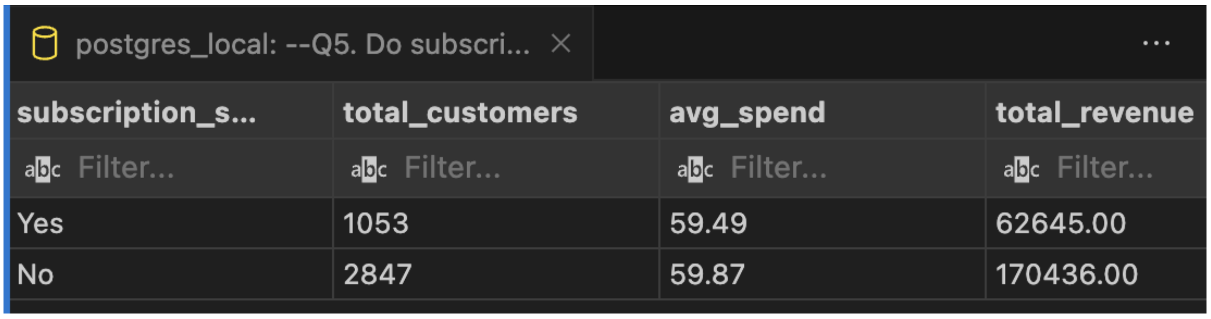


postgres_local: --Q4. Compare th...

shipping_type	round
Standard	58.46
Express	60.48

Q5. Subscriber vs Non-Subscriber Spending

```
SELECT subscription_status,
       COUNT(customer_id) AS total_customers,
       ROUND(AVG(purchase_amount), 2) AS avg_spend,
       ROUND(SUM(purchase_amount), 2) AS total_revenue
FROM customer
GROUP BY subscription_status
ORDER BY total_revenue DESC;
```



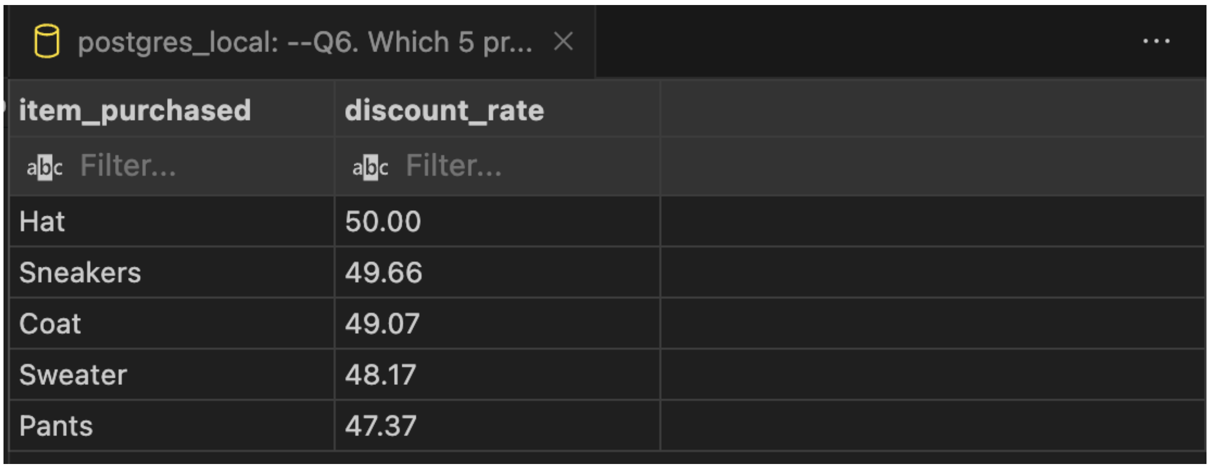
postgres_local: --Q5. Do subscri...

subscription_s...	total_customers	avg_spend	total_revenue
Yes	1053	59.49	62645.00
No	2847	59.87	170436.00

Q6. Products with Highest Discount Usage

```
SELECT item_purchased,
       ROUND(100.0 * SUM(CASE WHEN discount_applied = 'Yes' THEN 1 ELSE 0 END) /
COUNT(*), 2)
       AS discount_rate
FROM customer
GROUP BY item_purchased
ORDER BY discount_rate DESC
```

```
LIMIT 5;
```

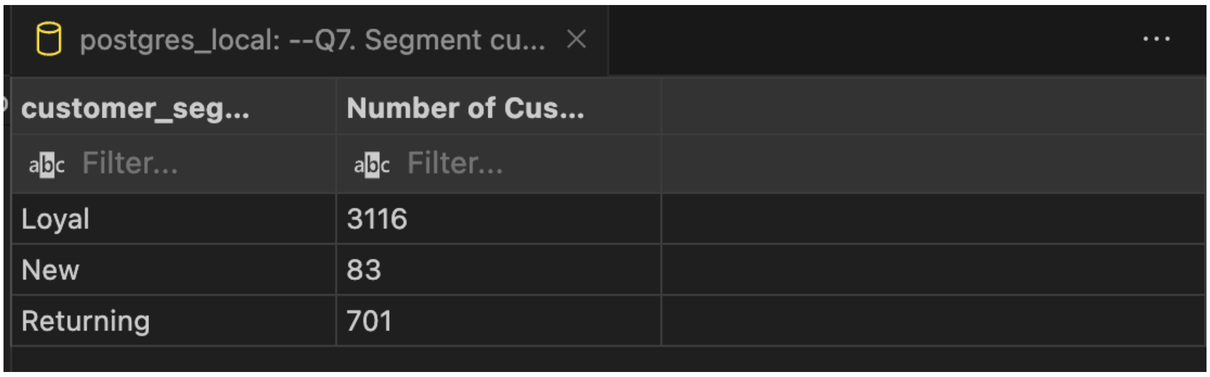


postgres_local: --Q6. Which 5 pr...

item_purchased	discount_rate
Hat	50.00
Sneakers	49.66
Coat	49.07
Sweater	48.17
Pants	47.37

Q7. Customer Segmentation

```
WITH customer_type AS (
   SELECT customer_id,
         CASE
            WHEN previous_purchases = 1 THEN 'New'
            WHEN previous_purchases BETWEEN 2 AND 10 THEN 'Returning'
            ELSE 'Loyal'
         END AS customer_segment
   FROM customer
)
SELECT customer_segment, COUNT(*)
FROM customer_type
GROUP BY customer_segment;
```



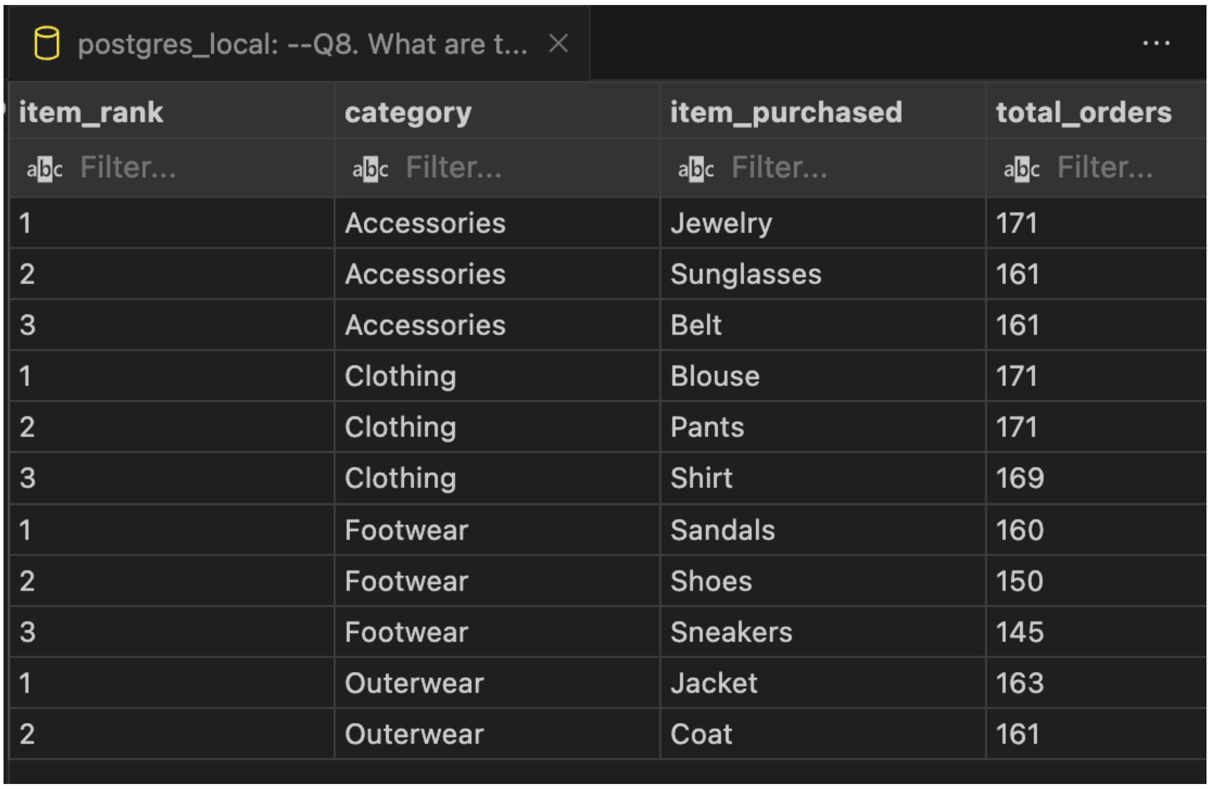
postgres_local: --Q7. Segment cu...

customer_seg...	Number of Cus...
Loyal	3116
New	83
Returning	701

Q8. Top 3 Products per Category

```
WITH item_counts AS (
   SELECT category, item_purchased,
         COUNT(customer_id) AS total_orders,
         ROW_NUMBER() OVER (PARTITION BY category ORDER BY COUNT(*)
DESC) AS item_rank
```

```
    FROM customer
    GROUP BY category, item_purchased
)
SELECT * FROM item_counts WHERE item_rank <= 3;
```

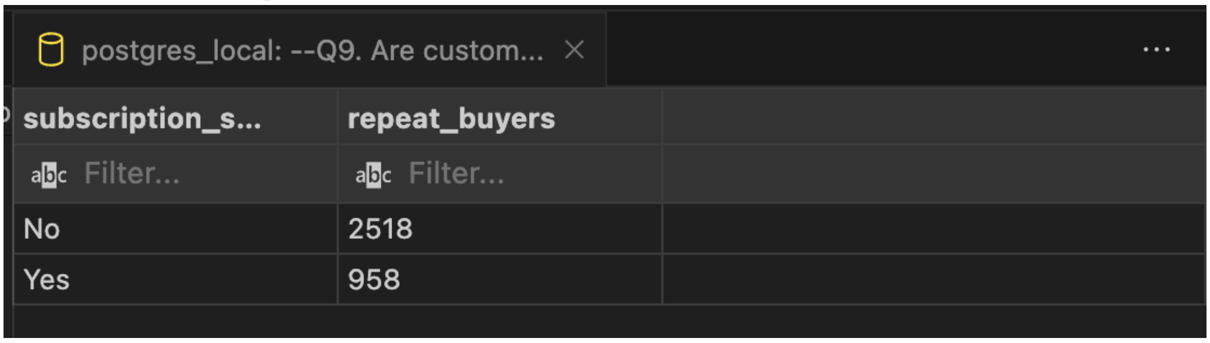


postgres_local: --Q8. What are t...

item_rank	category	item_purchased	total_orders
Filter...	Filter...	Filter...	Filter...
1	Accessories	Jewelry	171
2	Accessories	Sunglasses	161
3	Accessories	Belt	161
1	Clothing	Blouse	171
2	Clothing	Pants	171
3	Clothing	Shirt	169
1	Footwear	Sandals	160
2	Footwear	Shoes	150
3	Footwear	Sneakers	145
1	Outerwear	Jacket	163
2	Outerwear	Coat	161

Q9. Subscription Status of Repeat Buyers

```
SELECT subscription_status, COUNT(customer_id)
FROM customer
WHERE previous_purchases > 5
GROUP BY subscription_status;
```

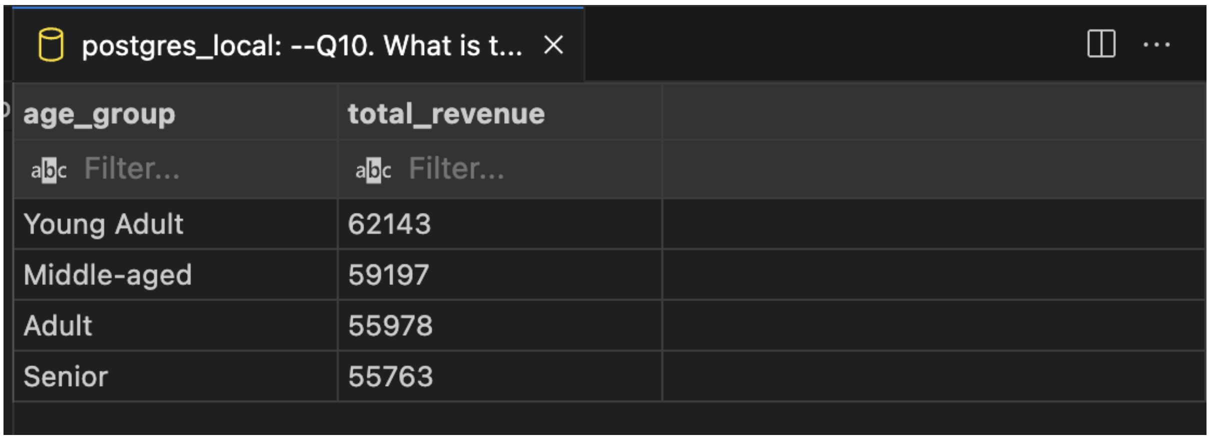


postgres_local: --Q9. Are custom...

subscription_s...	repeat_buyers
Filter...	Filter...
No	2518
Yes	958

Q10. Revenue by Age Group

```
SELECT age_group, SUM(purchase_amount) AS total_revenue
FROM customer
GROUP BY age_group
ORDER BY total_revenue DESC;
```

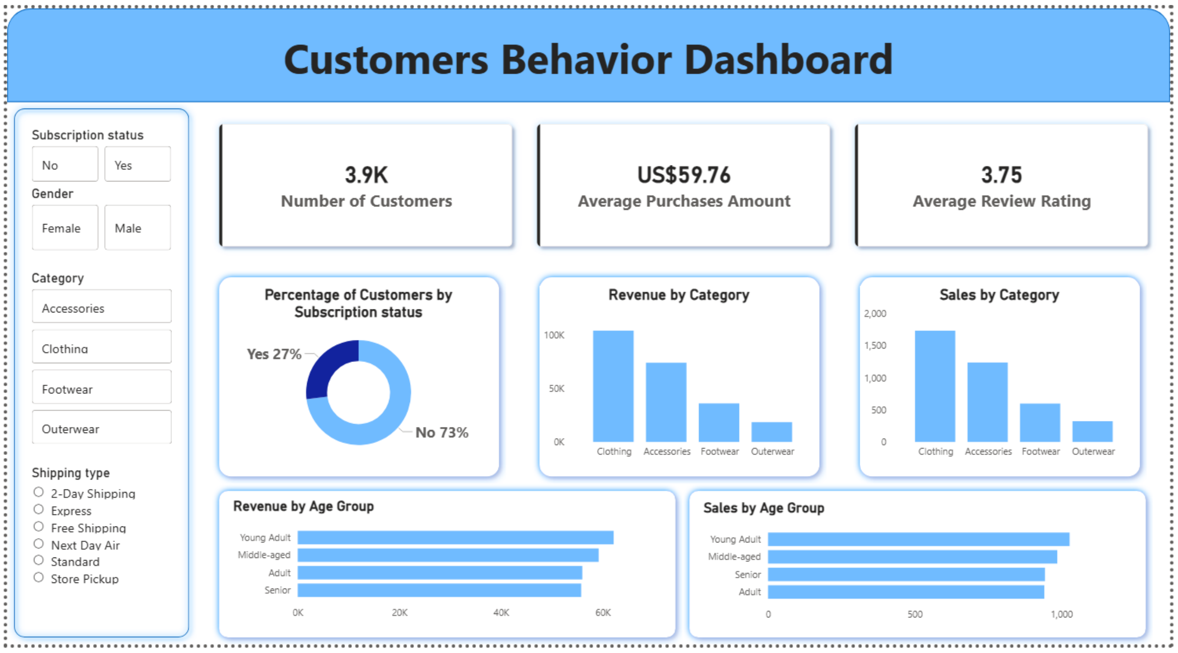


postgres_local: --Q10. What is t...

age_group	total_revenue
Filter...	Filter...
Young Adult	62143
Middle-aged	59197
Adult	55978
Senior	55763

8. Data Visualization with Power BI

Power BI Desktop was installed on a **Windows 11 ARM virtual machine (VMware Fusion)** due to macOS limitations.



Dashboard Features:

- KPI Cards:
  - Number of Customers

    - Average Purchase Amount
    - Average Review Rating
- Donut Chart: Subscription Status Distribution
- Bar Charts:
    - Revenue by Category
    - Sales by Category
    - Revenue by Age Group
    - Sales by Age Group
- Interactive Slicers:
    - Subscription Status
    - Gender
    - Category
    - Shipping Type

The final dashboard provides interactive insights into customer behavior and purchasing trends.

9. Key Insights

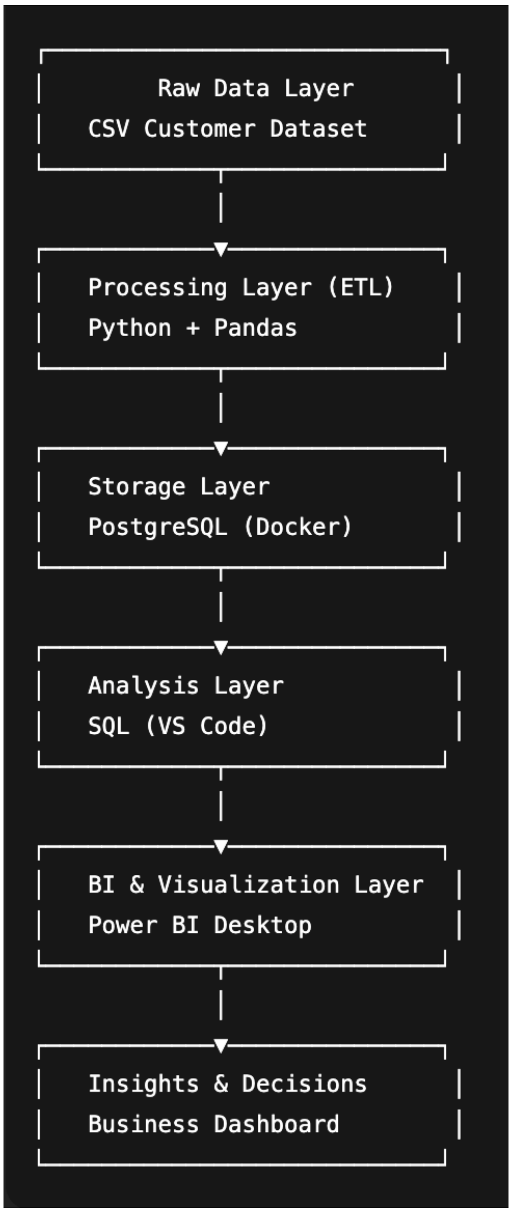
- Subscribed customers generate higher total revenue
- Clothing category dominates both revenue and sales volume
- Young Adults and Middle-aged customers contribute the most revenue
- Discounts are heavily used on specific products
- Faster shipping does not always imply higher spending

10. Data Pipeline Architecture

This project follows a structured end-to-end data pipeline that transforms raw customer data into actionable business insights. The pipeline is modular, scalable, and closely aligned with real-world data analytics workflows.

10.1 High-Level Pipeline Flow

Raw Customer Dataset (CSV)
|
▼
Python Data Cleaning & Feature Engineering
(Jupyter Notebook / Pandas)
|
▼
PostgreSQL Database (Docker on macOS)
|
▼
SQL Analysis (VS Code)
|
▼
Power BI Desktop (Windows VM)
|
▼
Interactive Business Dashboard



10.2 Pipeline Stages

1. Data Source Layer

The pipeline begins with a raw CSV dataset containing customer shopping behavior data. This dataset includes demographic information, purchase details, subscription status, and transactional attributes.

2. Data Processing Layer (ETL – Python)

Data preprocessing and feature engineering are performed using Python in Jupyter Notebook. Key steps include handling missing values, standardizing columns, creating derived features such as age groups, and preparing the dataset for database storage.

3. Data Storage Layer (PostgreSQL)

The cleaned dataset is stored in a PostgreSQL 15 database running inside a Docker container. This ensures portability, isolation, and consistency across environments.

4. Data Analysis Layer (SQL)

SQL queries are executed using Visual Studio Code to validate data integrity and answer business questions. This layer includes aggregations, subqueries, common table expressions (CTEs), and window functions.

5. Data Consumption Layer (Power BI)

Power BI Desktop connects to PostgreSQL using an ODBC Unicode driver. Data is imported into Power BI, where DAX measures and calculated columns are created to support analysis.

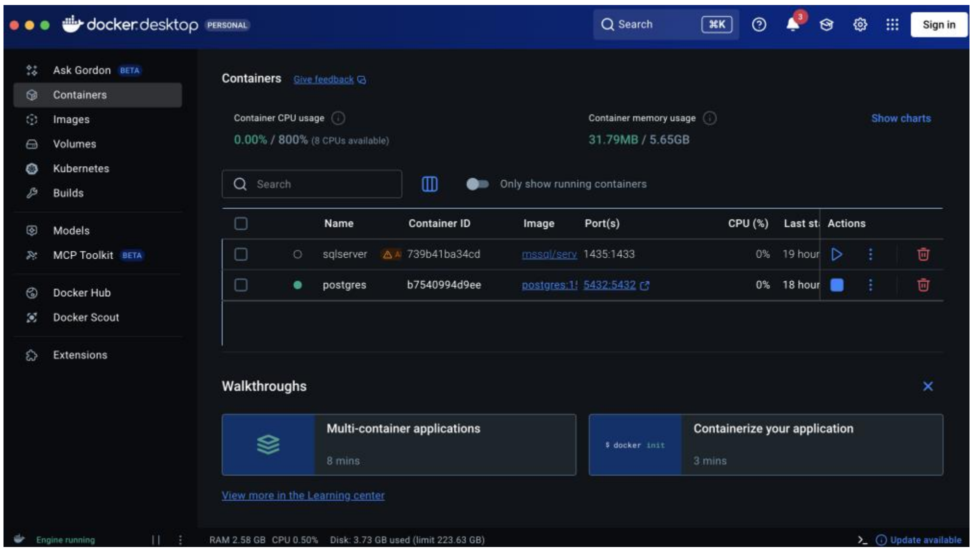
6. Visualization & Insight Layer

The final layer consists of an interactive Power BI dashboard featuring KPIs, charts, and slicers that allow users to explore customer behavior and purchasing trends.

11. Screenshots & Visual References

The following screenshots were captured during different stages of the project to demonstrate implementation, validation, and final outcomes. These visuals support the technical steps described in this report.

11.1 Environment & Database Setup



Containers

Container CPU usage
0.00% / 800% (8 CPUs available)

Container memory usage
31.79MB / 5.65GB

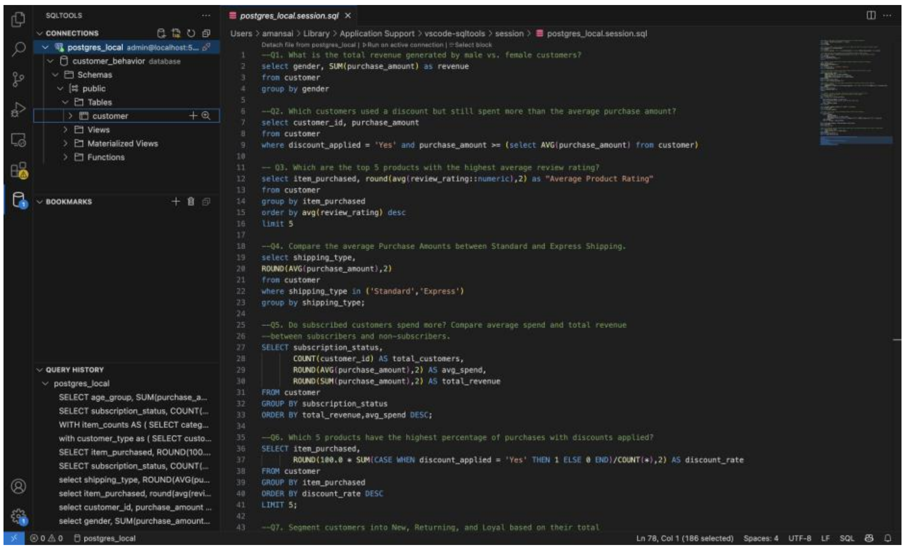
Name	Container ID	Image	Port(s)	CPU (%)	Last st	Actions
sqlserver	739b41ba34cd	mssql/serv	1435:1433	0%	19 hour	
postgres	b7540994d9ee	postgres:1!	5432:5432	0%	18 hour	

RAM 2.58 GB CPU 0.50% Disk: 3.73 GB used (limit 223.63 GB)

- Docker container running PostgreSQL 15 on macOS
- PostgreSQL container port mapping (5432)
- SQLTools connection to PostgreSQL via Visual Studio Code

Purpose: Confirms successful database deployment and connectivity from local development tools.

11.2 SQL Analysis (VS Code)



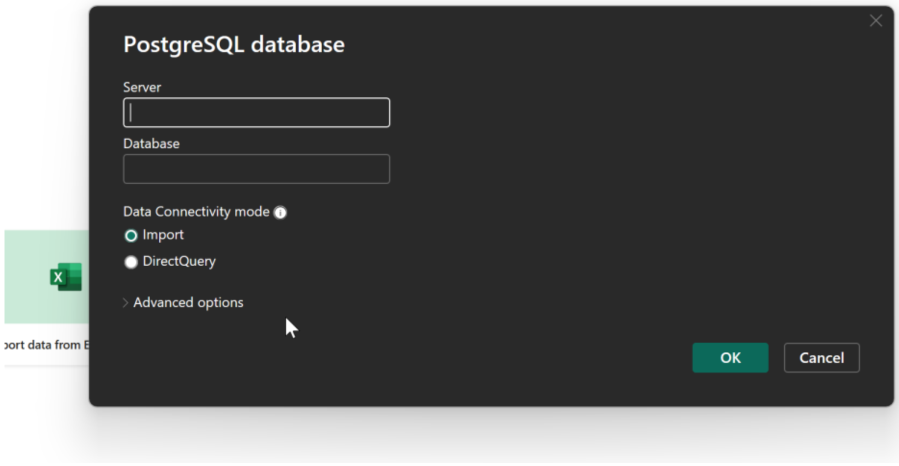
```
--Q1. What is the total revenue generated by male vs. female customers?
select gender, SUM(purchase_amount) as revenue
from customer
group by gender

--Q2. Which customers used a discount but still spent more than the average purchase amount?
select customer_id, purchase_amount
from customer
where discount_applied = 'Yes' and purchase_amount >= (select AVG(purchase_amount) from customer)

-- Q3. Which are the top 5 products with the highest average review rating?
select item_purchased, round(avg(review_rating::numeric),2) as "Average Product Rating"
from customer
group by item_purchased
order by avg(review_rating) desc
limit 5

--Q4. Compare the average Purchase Amounts between Standard and Express Shipping.
select shipping_type,
ROUND(AVG(purchase_amount),2)
from customer
where shipping_type in ('Standard','Express')
group by shipping_type;

--Q5. Do subscribed customers spend more? Compare average spend and total revenue
--between subscribers and non-subscribers.
SELECT subscription_status,
        COUNT(customer_id) AS total_customers,
        ROUND(AVG(purchase_amount),2) AS avg_spend,
        ROUND(SUM(purchase_amount),2) AS total_revenue
FROM customer
GROUP BY subscription_status
ORDER BY total_revenue,avg_spend DESC;

--Q6. Which 5 products have the highest percentage of purchases with discounts applied?
SELECT item_purchased,
        ROUND(100.0 * SUM(CASE WHEN discount_applied = 'Yes' THEN 1 ELSE 0 END)/COUNT(*),2) AS discount_rate
FROM customer
GROUP BY item_purchased
ORDER BY discount_rate DESC
LIMIT 5;

--Q7. Segment customers into New, Returning, and Loyal based on their total
```

- SQL queries executed in Visual Studio Code
- Query results validating business questions (Q1–Q10)
- Use of aggregations, subqueries, CTEs, and window functions

Purpose: Shows analytical querying skills and direct interaction with the relational database.

11.3 Power BI Connectivity

- Power BI PostgreSQL connection dialog



PostgreSQL database

Server:

Database:

Data Connectivity mode: [✓ Import] DirectQuery

When selecting **Get Data → PostgreSQL database** in Power BI Desktop, the following values were entered:

- **Server:** <macOS_host_IP>:5432

Example: 192.168.0.226:5432

Explanation:
Power BI runs inside a Windows VM, while PostgreSQL runs on the macOS host via Docker.
Using localhost would incorrectly point to the Windows VM itself. Therefore, the macOS host IP address was used to allow cross-machine communication.

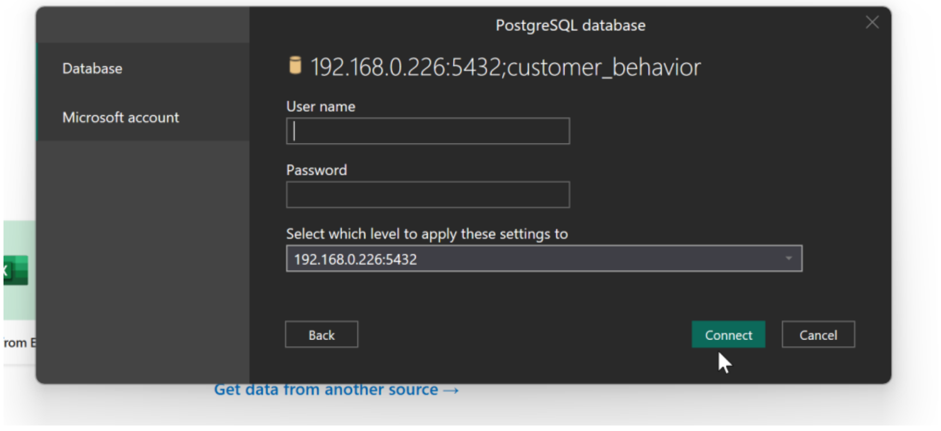
- **Database:** customer_behavior

Explanation:
This is the PostgreSQL database created during Docker container initialization.

- **Data Connectivity Mode:** Import

Explanation:
Import mode was selected to load the data directly into Power BI for faster performance and simplified modeling.

- Authentication and encryption fallback message



PostgreSQL database

Database

Microsoft account

192.168.0.226:5432;customer_behavior

User name

Password

Select which level to apply these settings to

192.168.0.226:5432

After clicking **OK**, Power BI prompted for database credentials. The following values were entered:

- **Username:** admin
- **Password:** admin123
- **Apply settings to:** <macOS_host_IP>:5432

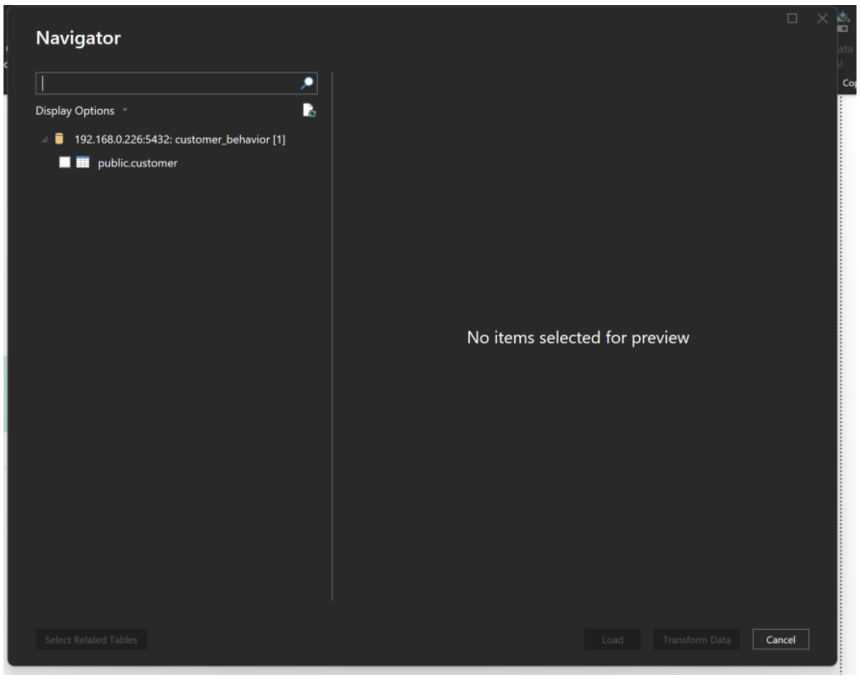
Explanation:
These credentials match the PostgreSQL user and password defined when creating the Docker container. Authentication was performed using **Database credentials**.

Power BI displayed an encryption warning indicating that SSL was not enabled on the PostgreSQL server.

- **Action Taken:** Clicked **OK** to proceed without encryption.

Explanation:
Since PostgreSQL was running locally in Docker for development purposes, SSL encryption was not configured. Disabling encryption is acceptable for local and portfolio projects and does not affect data integrity in this context.

- Navigator view showing public.customer table


Navigator

192.168.0.226:5432: customer_behavior [1]

public.customer

No items selected for preview

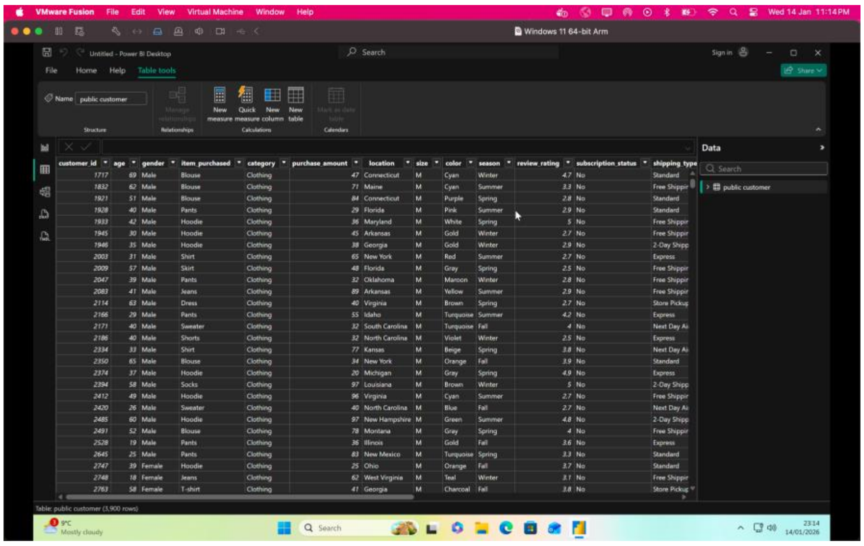
After successful authentication, the **Navigator** window appeared.

- **Database:** customer_behavior
- **Schema:** public
- **Table Selected:** customer

Action Taken:

- Selected the public.customer table
- Clicked **Load**

Explanation:
The customer table contains the cleaned and transformed dataset loaded from Python using SQLAlchemy. Loading this table made the data available in Power BI for modeling and visualization.


VMware Fusion File Edit View Virtual Machine Window Help
Wed 14 Jan 11:14 PM
Windows 11 64-bit Arm
Untitled - Power BI Desktop
Search
Sign in
Share
File Home Help Table tools
Name public customer
Manage relationships
New measure Quick measure New column New table
Mark as date table
Structure Relationships Calculations Calendars
Data
Search
public customer
9°C Mostly cloudy
23:14 14/01/2026

customer_id	age	gender	item_purchased	category	purchase_amount	location	size	color	season	review_rating	subscription_status	shipping_type
1717	69	Male	Blouse	Clothing	47	Connecticut	M	Cyan	Winter	4.7	No	Standard
1832	62	Male	Blouse	Clothing	71	Maine	M	Cyan	Summer	3.3	No	Free Shippir
1921	51	Male	Blouse	Clothing	84	Connecticut	M	Purple	Spring	2.8	No	Standard
1928	40	Male	Pants	Clothing	29	Florida	M	Pink	Summer	2.9	No	Standard
1933	42	Male	Hoodie	Clothing	36	Maryland	M	White	Spring	5	No	Free Shippir
1945	30	Male	Hoodie	Clothing	45	Arkansas	M	Gold	Winter	2.7	No	Free Shippir
1946	35	Male	Hoodie	Clothing	38	Georgia	M	Gold	Winter	2.9	No	2-Day Shipp
2003	31	Male	Shirt	Clothing	65	New York	M	Red	Summer	2.7	No	Express
2009	57	Male	Skirt	Clothing	48	Florida	M	Gray	Spring	2.5	No	Free Shippir
2047	39	Male	Pants	Clothing	32	Oklahoma	M	Maroon	Winter	2.8	No	Free Shippir
2083	41	Male	Jeans	Clothing	89	Arkansas	M	Yellow	Summer	2.9	No	Free Shippir
2114	63	Male	Dress	Clothing	40	Virginia	M	Brown	Spring	2.7	No	Store Pickup
2166	29	Male	Pants	Clothing	55	Idaho	M	Turquoise	Summer	4.2	No	Express
2171	40	Male	Sweater	Clothing	32	South Carolina	M	Turquoise	Fall	4	No	Next Day Ai
2186	40	Male	Shorts	Clothing	32	North Carolina	M	Violet	Winter	2.5	No	Express
2334	33	Male	Shirt	Clothing	77	Kansas	M	Beige	Spring	3.8	No	Next Day Ai
2350	65	Male	Blouse	Clothing	34	New York	M	Orange	Fall	3.9	No	Standard
2374	37	Male	Hoodie	Clothing	20	Michigan	M	Gray	Spring	4.9	No	Express
2394	58	Male	Socks	Clothing	97	Louisiana	M	Brown	Winter	5	No	2-Day Shipp
2412	49	Male	Hoodie	Clothing	96	Virginia	M	Cyan	Summer	2.7	No	Free Shippir
2420	26	Male	Sweater	Clothing	40	North Carolina	M	Blue	Fall	2.7	No	Next Day Ai
2485	60	Male	Hoodie	Clothing	97	New Hampshire	M	Green	Summer	4.8	No	2-Day Shipp
2491	52	Male	Blouse	Clothing	78	Montana	M	Gray	Spring	4	No	Free Shippir
2528	19	Male	Pants	Clothing	36	Illinois	M	Gold	Fall	3.6	No	Express
2645	25	Male	Pants	Clothing	83	New Mexico	M	Turquoise	Spring	3.3	No	Standard
2747	39	Female	Hoodie	Clothing	25	Ohio	M	Orange	Fall	3.7	No	Standard
2748	18	Female	Jeans	Clothing	62	West Virginia	M	Teal	Winter	3.1	No	Free Shippir
2763	58	Female	T-shirt	Clothing	41	Georgia	M	Charcoal	Fall	3.8	No	Store Pickup

Table: public customer (3,900 rows)

Purpose: Confirms successful integration between PostgreSQL (Docker) and Power BI Desktop via ODBC.

12. Conclusion

This project successfully demonstrates an end-to-end data analytics workflow using modern tools and best practices. By integrating Python, SQL, PostgreSQL, Docker, and Power BI, meaningful insights were derived from raw customer data. The approach closely mirrors real-world data analyst and business intelligence workflows.

13. Future Enhancements

- Automate ETL pipelines using Apache Airflow
- Migrate PostgreSQL to a cloud service (AWS RDS / Azure Database)
- Publish the Power BI dashboard to Power BI Service
- Add forecasting and predictive analytics using machine learning
- Implement role-based access and data refresh scheduling