Link to dataset: **https://www.kaggle.com/datasets/uom190346a/e-commerce-customer-behavior-dataset?resource=download**

Project Overview: This "E-commerce Customer Behavior" dataset simulates or reflects records of customer information and purchase activity on an online retail platform. It typically includes each customer's demographic details (such as Gender, Age, City), spending behavior (like Total Spend, Items Purchased), and engagement indicators (Days Since Last Purchase, Membership Type). It also captures customer satisfaction data, including an Average Rating, a Satisfaction Level, and whether a Discount was applied to their purchases.

Here's a quick rundown of what each column represents:

- Customer ID: A unique identifier for each customer.
- Gender: The customer's reported gender.
- Age: The customer's age (in years).
- City: The city in which the customer resides (useful for geographic analysis).
- Membership Type: The tier or plan the customer is enrolled in (e.g., Basic, Premium).
- Total Spend: Total amount of money spent by the customer (e.g., in dollars).
- Items Purchased: The total number of items the customer has bought.
- Average Rating: A numerical score reflecting the customer's rating of products or services (e.g., out of 5).
- Discount Applied: A boolean-like indicator (0 or 1) showing whether a discount was used.
- Days Since Last Purchase: How many days have passed since this customer last bought something.
- Satisfaction Level: A qualitative label indicating the customer's overall satisfaction (e.g., Low, Medium, High, Very High).

Because it includes demographic, transactional, and satisfaction data, this dataset is ideal for exploring key e-commerce metrics like:

- Spending behavior by membership tier or location
- Customer segmentation (e.g., high vs. low spenders)
- Correlation between discounts and ratings
- Frequency of purchases (via days since last purchase)
- Relationships between satisfaction level, items purchased, and total spend

In short, this dataset allows you to do a well-rounded SQL-driven analysis of consumer habits and satisfaction factors in an e-commerce context.

1) Check the First 10 Rows

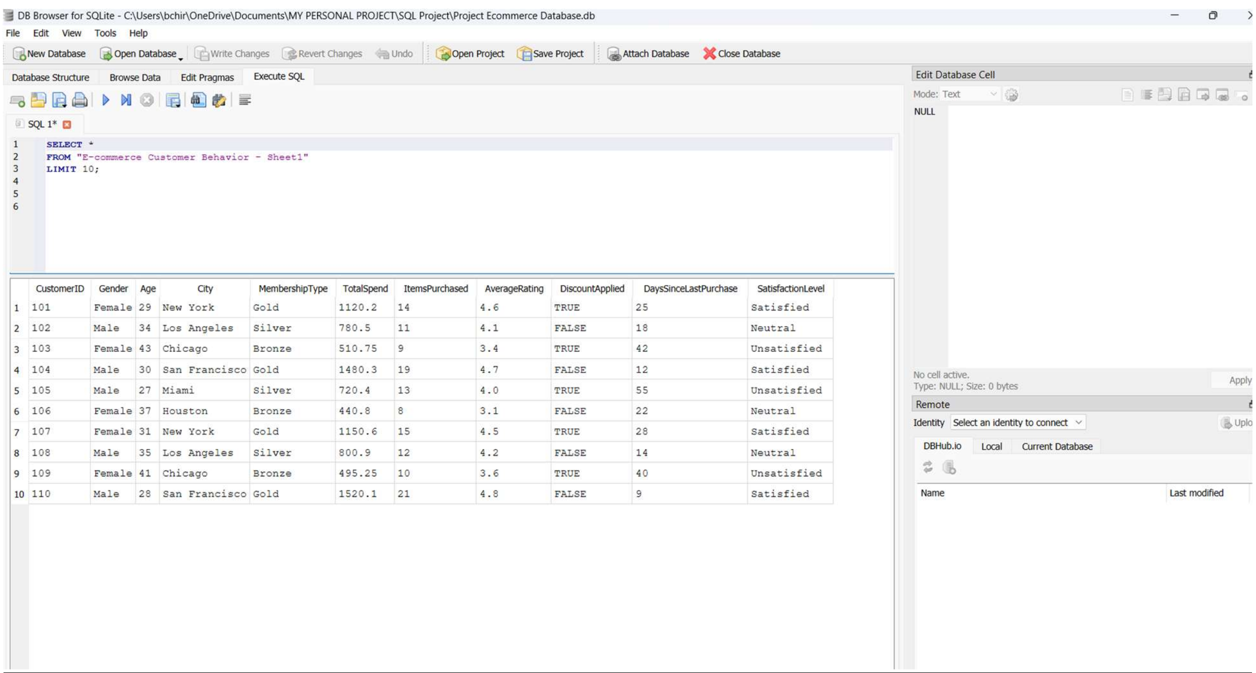
QUERY= SELECT *

FROM "E-commerce Customer Behavior - Sheet1"

LIMIT 10;

Purpose: Quickly verify that the data imported correctly. You can see the first few rows, check column formatting, and confirm that nothing looks obviously broken or misaligned.

Use Case: Helps you do a quick “sanity check” before diving deeper into analysis.



```
SELECT *
FROM "E-commerce Customer Behavior - Sheet1"
LIMIT 10;
```

	CustomerID	Gender	Age	City	MembershipType	TotalSpend	ItemsPurchased	AverageRating	DiscountApplied	DaysSinceLastPurchase	SatisfactionLevel
1	101	Female	29	New York	Gold	1120.2	14	4.6	TRUE	25	Satisfied
2	102	Male	34	Los Angeles	Silver	780.5	11	4.1	FALSE	18	Neutral
3	103	Female	43	Chicago	Bronze	510.75	9	3.4	TRUE	42	Unsatisfied
4	104	Male	30	San Francisco	Gold	1480.3	19	4.7	FALSE	12	Satisfied
5	105	Male	27	Miami	Silver	720.4	13	4.0	TRUE	55	Unsatisfied
6	106	Female	37	Houston	Bronze	440.8	8	3.1	FALSE	22	Neutral
7	107	Female	31	New York	Gold	1150.6	15	4.5	TRUE	28	Satisfied
8	108	Male	35	Los Angeles	Silver	800.9	12	4.2	FALSE	14	Neutral
9	109	Female	41	Chicago	Bronze	495.25	10	3.6	TRUE	40	Unsatisfied
10	110	Male	28	San Francisco	Gold	1520.1	21	4.8	FALSE	9	Satisfied

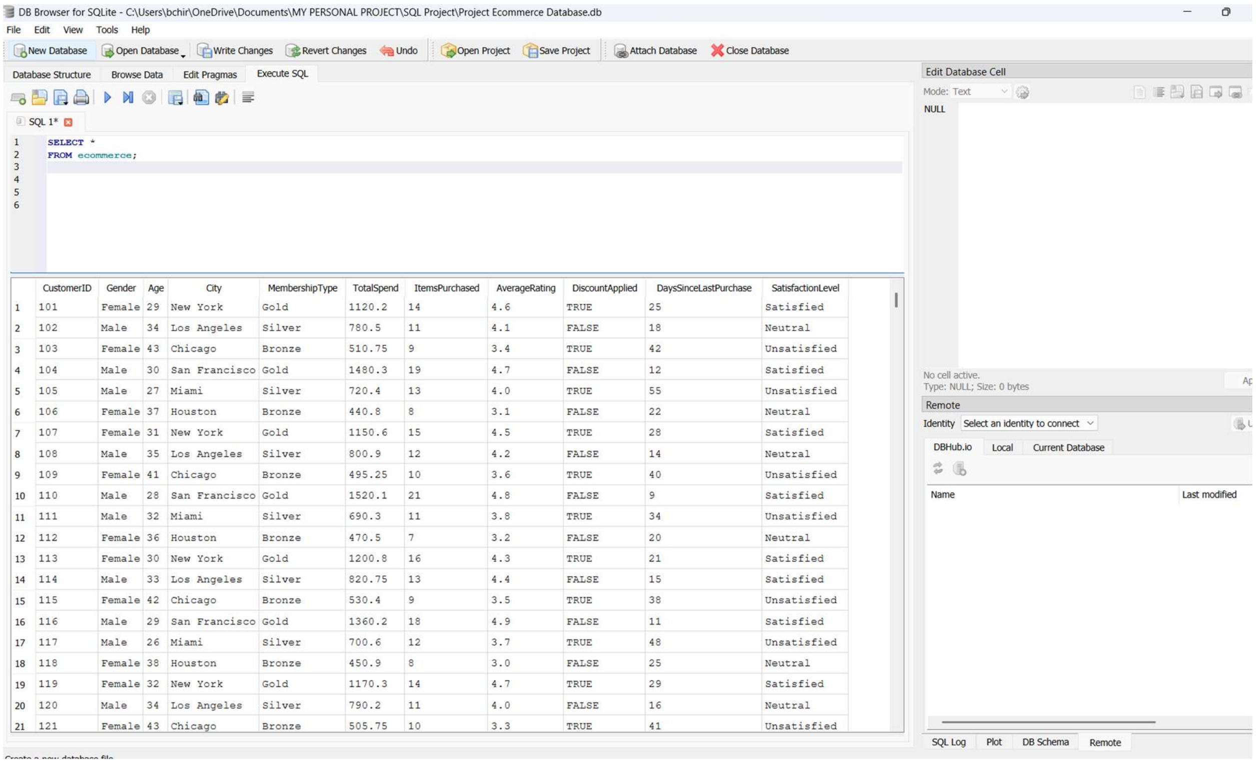
2) Renaming the Table

QUERY= ALTER TABLE "E-commerce Customer Behavior - Sheet1"

RENAME TO ecommerce;

Purpose: Because the table name has spaces and hyphens, quoting it all the time can be cumbersome. Renaming it to ecommerce (or something simpler) makes queries cleaner.

Use Case: Optional convenience step. If you don't mind quoting the original name, you can skip it. But typically, analysts rename tables for ease of use.



3) Basic Aggregate Queries

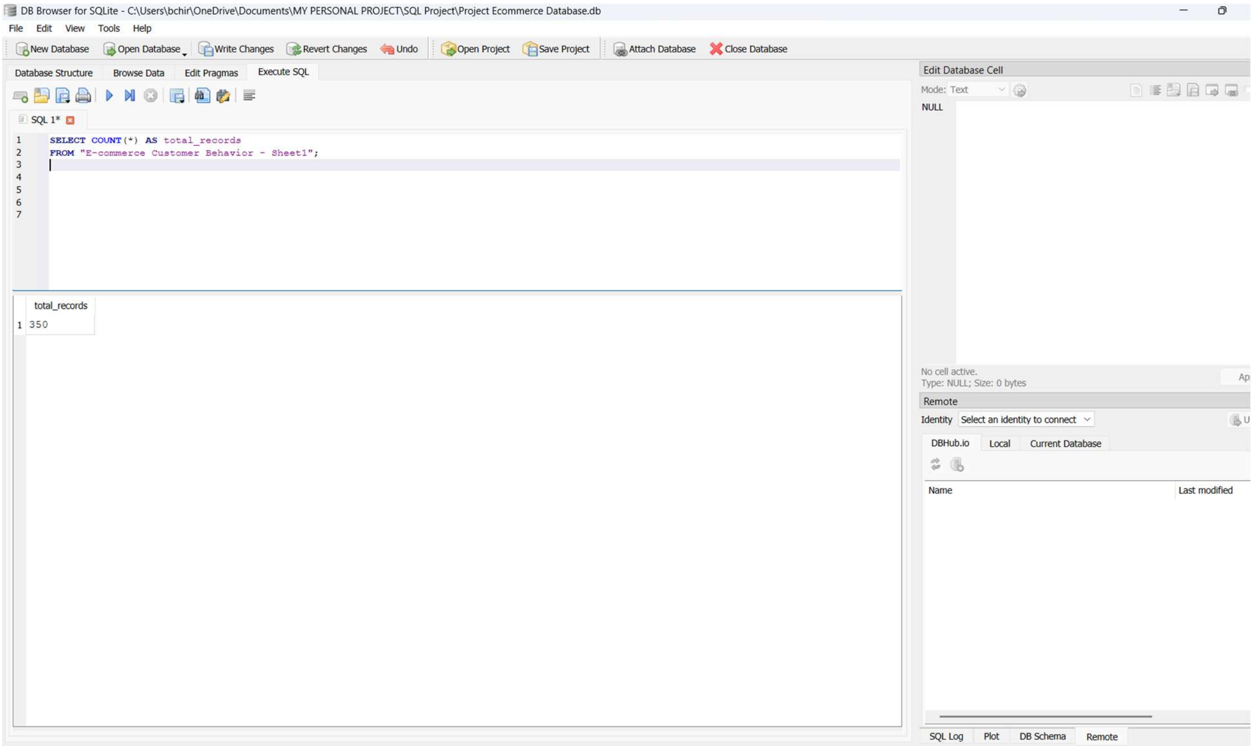
3A) Count Rows

QUERY= SELECT COUNT(*) AS total_rows

FROM "E-commerce Customer Behavior - Sheet1";

What it does: Returns how many records (rows) are in the table. This could be how many customers or transactions you have, depending on how the data is structured.

Why it's useful: Quick measure of the dataset size. Also checks if the number of rows matches what you expect from your CSV or documentation



Shows the total number of rows in the dataset.

3B) Average and Min/Max Age

QUERY= SELECT

AVG(Age) AS avg_age,

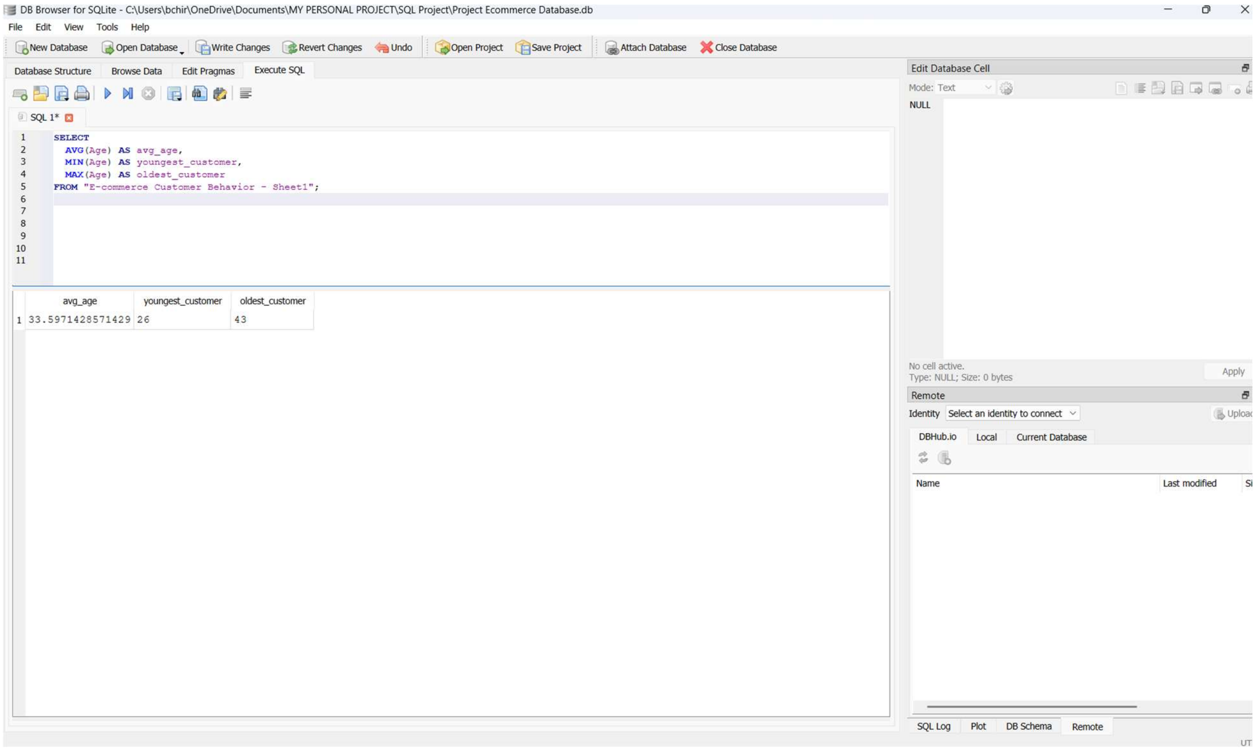
MIN(Age) AS youngest_customer,

MAX(Age) AS oldest_customer

FROM "E-commerce Customer Behavior - Sheet1";

What it does: Shows the **average age**, along with the **minimum** (youngest) and **maximum** (oldest) ages in the dataset.

Why it's useful: Gives a demographic snapshot—are most customers young adults, or older? Helps tailor marketing or product decisions based on typical customer age.



3C) Total Spend by Membership Type

QUERY= SELECT

"Membership Type",

SUM("Total Spend") AS total_spend,

COUNT(*) AS count_customers,

ROUND(AVG("Total Spend"), 2) AS avg_spend

FROM "E-commerce Customer Behavior - Sheet1"

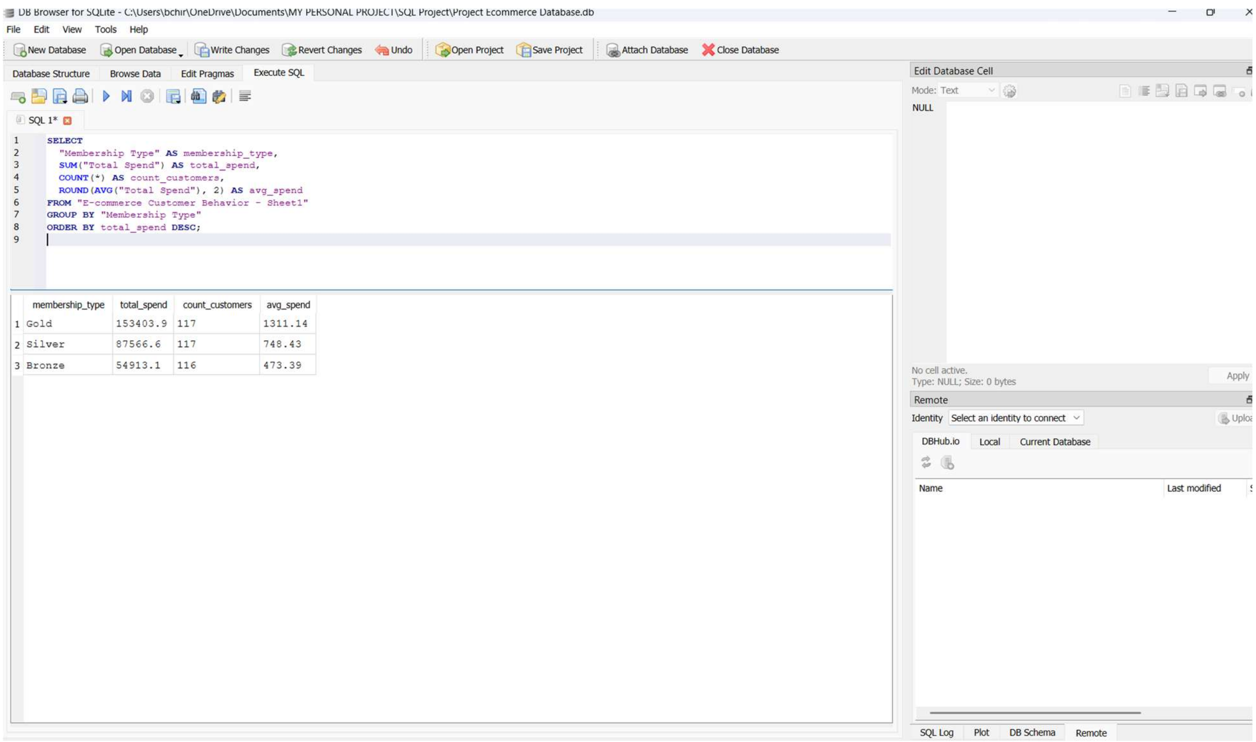
GROUP BY "Membership Type"

ORDER BY total_spend DESC;

What it does: Groups the data by membership level (like for instance Basic, Premium) and calculates:

- **Total spend** across all members in that tier
- **Number of customers** in that tier
- **Average spend** (rounded to 2 decimals)

Why it's useful: Identifies which membership tier generates the most revenue and how spending differs across tiers. You might see if a premium tier yields higher spend per customer.



3D) Distribution by City

QUERY= SELECT

City,

COUNT(*) AS number_of_customers,

ROUND(AVG("Total Spend"), 2) AS avg_spend

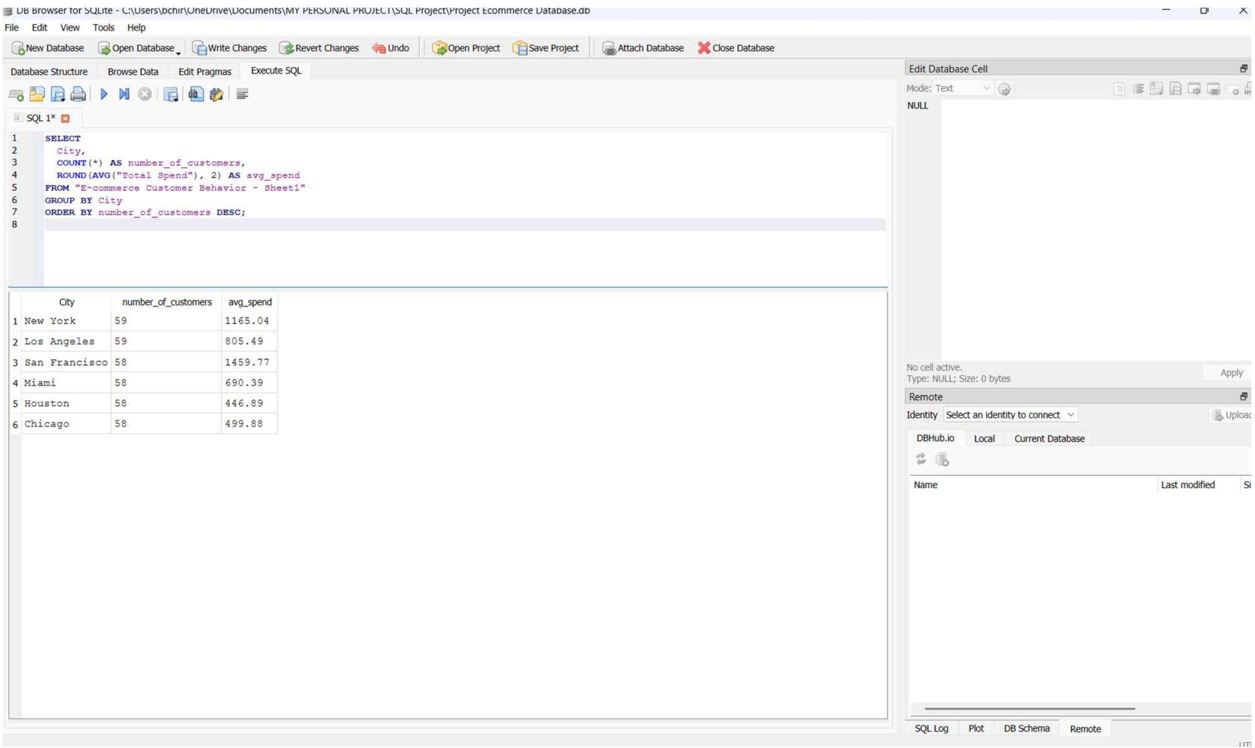
FROM "E-commerce Customer Behavior - Sheet1"

GROUP BY City

ORDER BY number_of_customers DESC;

What it does: Groups rows by **City**, shows how many customers (COUNT(*)) are from each city, and the **average** total spend in each city.

Why it's useful: Lets you identify where your largest customer bases are, and compare whether certain cities have higher or lower average spends.



3E) Average Rating vs. Discount Usage

QUERY= SELECT

"Discount Applied",

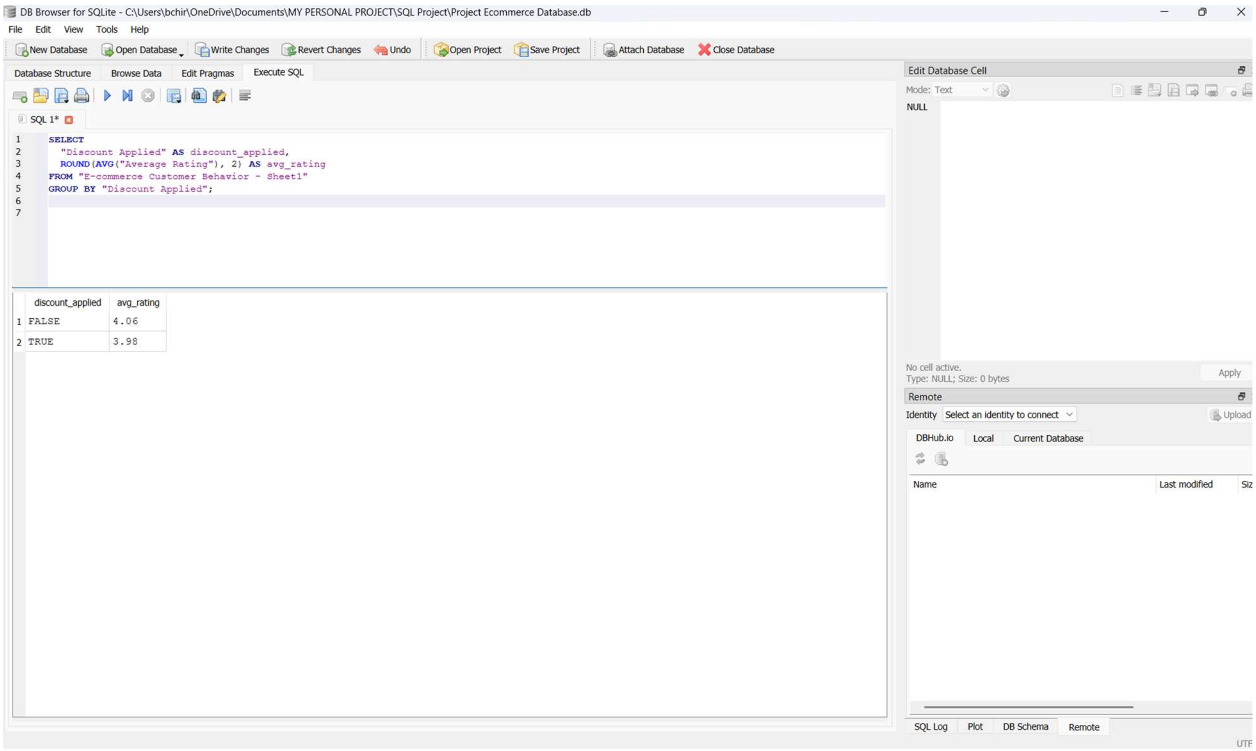
ROUND(AVG("Average Rating"), 2) AS avg_rating

FROM "E-commerce Customer Behavior - Sheet1"

GROUP BY "Discount Applied";

What it does: Splits customers (or transactions) based on whether "Discount Applied" is 0 or 1, then calculates the **average rating** in each group.

Why it's useful: You can see whether offering a discount correlates with higher (or lower) average ratings or satisfaction. Helps decide if discounts improve customer perception.



4) <u>More Advanced Examples</u>

These queries go beyond simple aggregates to explore relationships between columns, segment customers via a CASE statement, etc.

<u>4A) Days Since Last Purchase vs. Satisfaction Level</u>

QUERY= SELECT

"Satisfaction Level",

ROUND(AVG("Days Since Last Purchase"), 1) AS avg_days_since_last_purchase,

COUNT(*) AS total_customers

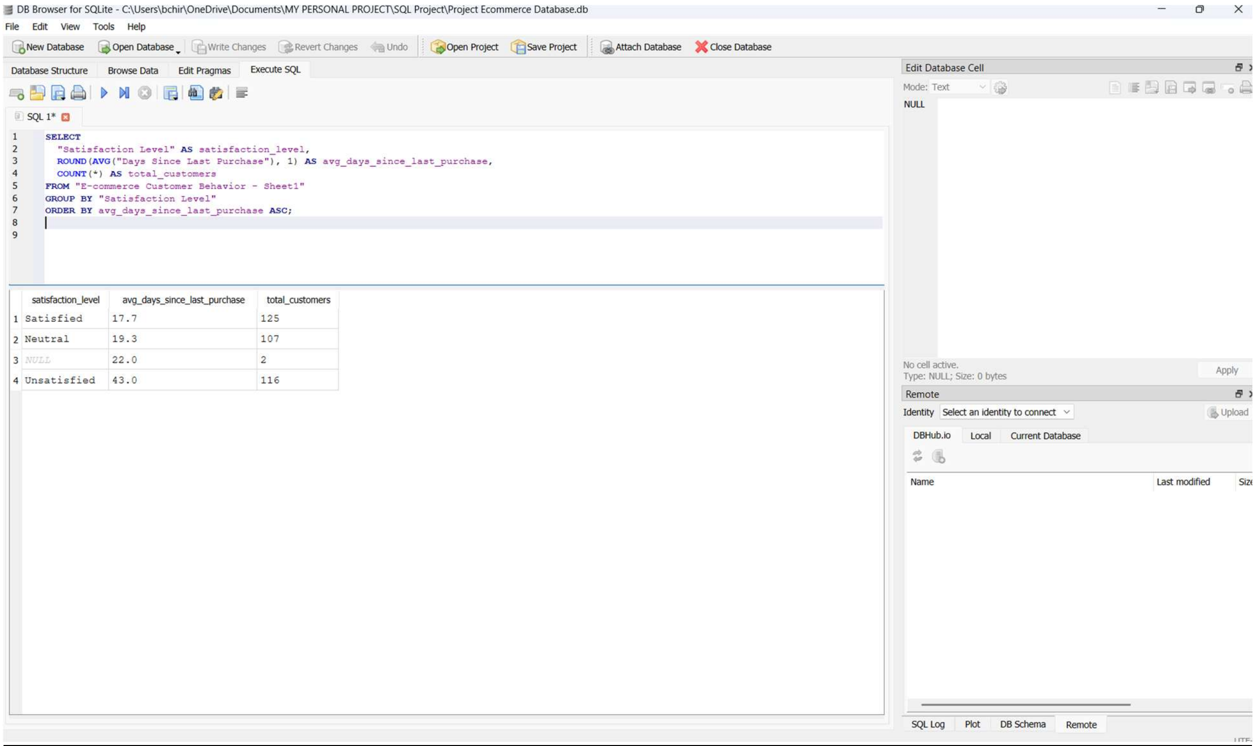
FROM "E-commerce Customer Behavior - Sheet1"

GROUP BY "Satisfaction Level"

ORDER BY avg_days_since_last_purchase ASC;

What it does: Groups rows by **Satisfaction Level**, calculates the **average** of "Days Since Last Purchase" in each group, and also shows how many people are in each satisfaction group.

Why it's useful: If highly satisfied customers tend to have fewer days since last purchase, it might mean they buy more frequently. Conversely, dissatisfied customers might be the ones who haven't purchased in a while.



4B) Categorize Total Spend with CASE

QUERY= SELECT

"Customer ID",

Gender,

CASE

WHEN "Total Spend" < 100 THEN 'Low Spender'

WHEN "Total Spend" BETWEEN 100 AND 500 THEN 'Medium Spender'

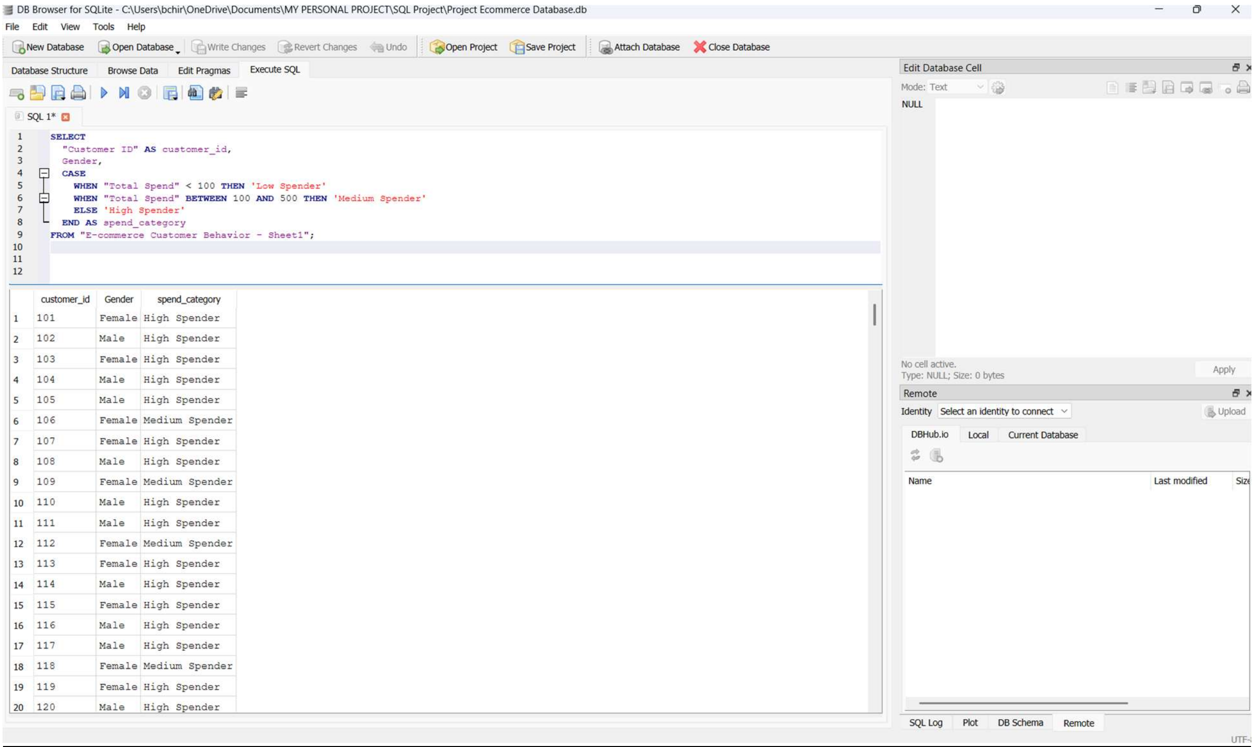
ELSE 'High Spender'

END AS spend_category

FROM "E-commerce Customer Behavior - Sheet1";

What it does: Dynamically assigns each row (customer) to a **"Low," "Medium," or "High"** spender category based on "Total Spend".

Why it's useful: Great for segmentation—who are your biggest spenders? You might target "High Spenders" with premium offers or "Low Spenders" with more frequent discounts.



4C) Items Purchased vs. Satisfaction Level

QUERY= SELECT

"Satisfaction Level",

ROUND(AVG("Items Purchased"), 1) AS avg_items

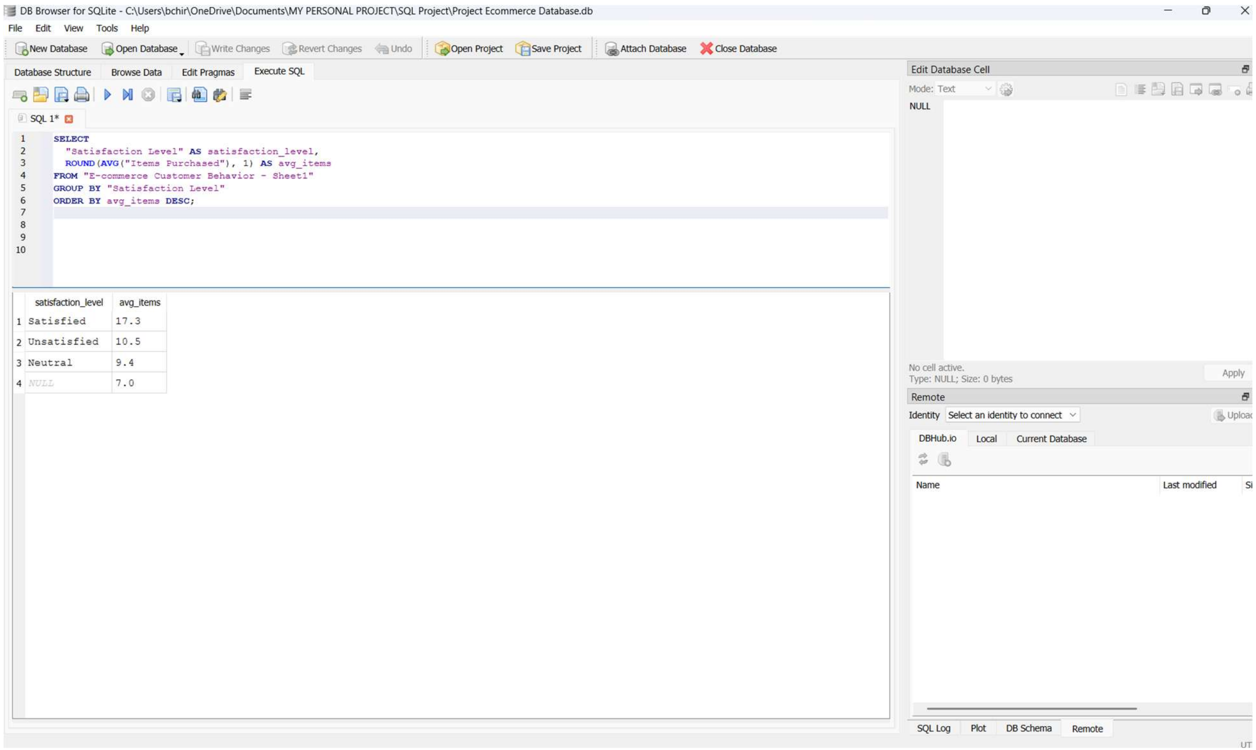
FROM "E-commerce Customer Behavior - Sheet1"

GROUP BY "Satisfaction Level"

ORDER BY avg_items DESC;

What it does: Groups rows by satisfaction level (like "High," "Medium," etc.) and shows the average number of items purchased for each group.

Why it's useful: Helps confirm if the most satisfied customers also buy the most items (which could signal loyalty). Alternatively, you might see if unhappy customers also have smaller purchase quantities.


DB Browser for SQLite - C:\Users\bchir\OneDrive\Documents\MY PERSONAL PROJECT\SQL Project\Project Ecommerce Database.db
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Undo
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1*
SELECT
"Satisfaction Level" AS satisfaction_level,
ROUND(AVG("Items Purchased"), 1) AS avg_items
FROM "E-commerce Customer Behavior - Sheet1"
GROUP BY "Satisfaction Level"
ORDER BY avg_items DESC;
satisfaction_level
avg_items
1 Satisfied 17.3
2 Unsatisfied 10.5
3 Neutral 9.4
4 NULL 7.0
Edit Database Cell
Mode: Text
NULL
No cell active.
Type: NULL; Size: 0 bytes
Apply
Remote
Identity
Select an identity to connect
DBHub.io
Local
Current Database
Name
Last modified
SQL Log
Plot
DB Schema
Remote