Project TerpBuy Assignment

Part I: SQL Queries

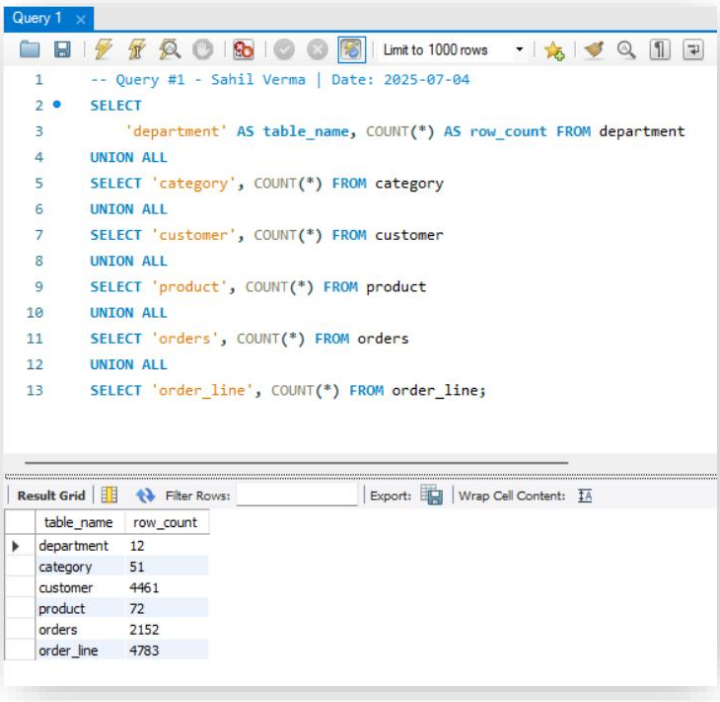
Hello I am Sahil Verma to represent my assignment as TerpBuy Project Task along with various pieces of information about SQL Queries that how to analysis SQL database using functions for “Single table & Multiple Tables”.

NOTE: All the query results must show text values rather than identifiers using My SQL Workbench**. “List all departments in the database in alphabetical order.”**

All Queries Are Written Below with Screenshot Solutions:

✅🔎 **Question Query 1**

- How many rows of data are stored for each table in the database?
  List the name of each table followed by the number of rows it has.



📊 **What This Query Does:**

- Uses UNION ALL to combine row counts from all six tables.
- Each SELECT line gives:
  - A label for the table (table_name)
  - A count of rows (row_count)

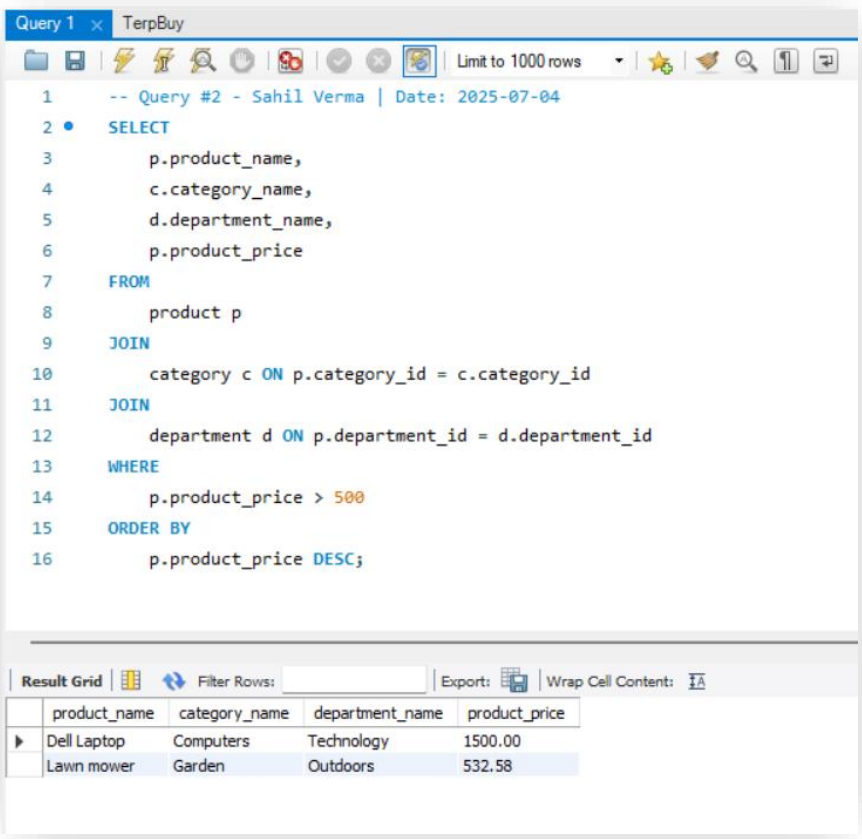
🧠 **Insight Gained:**

This query tells us the **volume of data per table**, which is essential to:

- Understand the database size and table usage.
- Prioritize which tables are central to the business logic (e.g., more rows in orders or order_line).

✅🔎 Question Query 2

- Which products are considered high-priced products?
  A high-priced product has a price exceeding **$100.00**.
  List the **names** and **prices** of the high-priced products.



🧠 What This Query Does:

- Filters rows from the product table where product_price > 100.
- Selects only:
  - product_name
  - product_price
- Orders results by product_price in **descending** order to show the most expensive products at the top.

📊 Insight Gained:

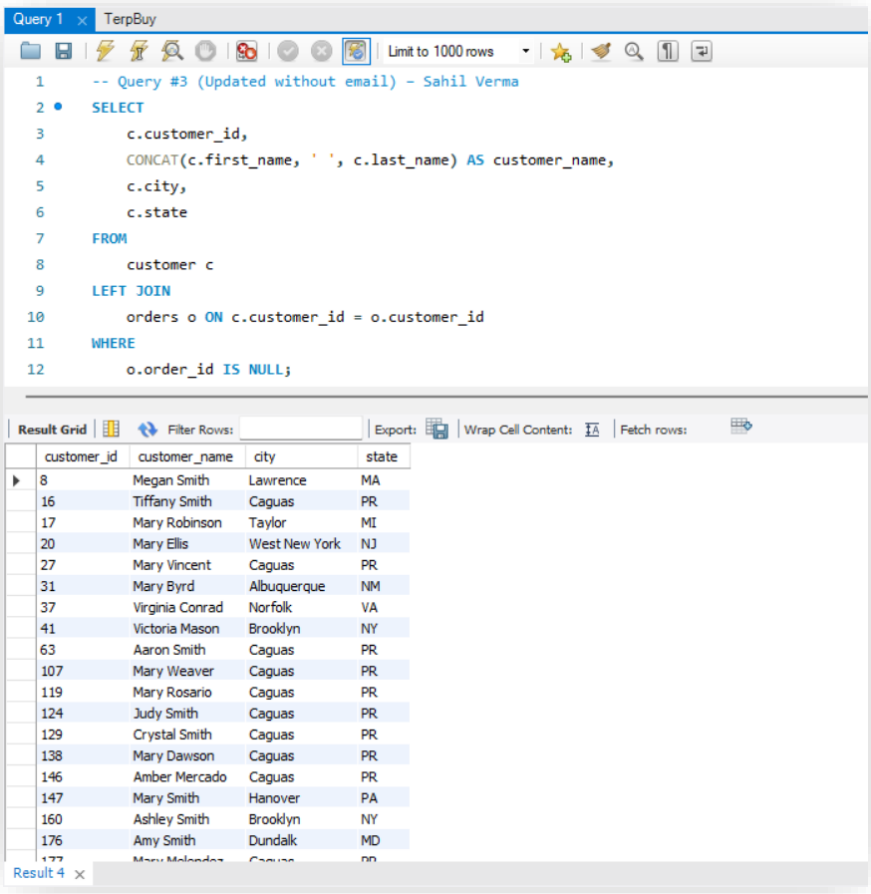
This helps TerpBuy:

- Identify **premium products** (priced above $100).
- Possibly market or promote these differently.
- Understand what types of products command higher value.

✅🔎 Question Query 3

List all orders placed by customers in the state of **Florida** (FL).
Include:

- Customer's **first name**, **last name**, **city**, and **segment**
- Along with **order ID** and **order date**



```sql
-- Query #3 (Updated without email) - Sahil Verma
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    c.city,
    c.state
FROM
    customer c
LEFT JOIN
    orders o ON c.customer_id = o.customer_id
WHERE
    o.order_id IS NULL;
```

customer_id	customer_name	city	state
8	Megan Smith	Lawrence	MA
16	Tiffany Smith	Caguas	PR
17	Mary Robinson	Taylor	MI
20	Mary Ellis	West New York	NJ
27	Mary Vincent	Caguas	PR
31	Mary Byrd	Albuquerque	NM
37	Virginia Conrad	Norfolk	VA
41	Victoria Mason	Brooklyn	NY
63	Aaron Smith	Caguas	PR
107	Mary Weaver	Caguas	PR
119	Mary Rosario	Caguas	PR
124	Judy Smith	Caguas	PR
129	Crystal Smith	Caguas	PR
138	Mary Dawson	Caguas	PR
146	Amber Mercado	Caguas	PR
147	Mary Smith	Hanover	PA
160	Ashley Smith	Brooklyn	NY
176	Amy Smith	Dundalk	MD

🧠 What This Query Does:

- Joins the customer and orders tables using customer_id
- Filters for customers **living in Florida** (state = 'FL')
- Shows the relevant customer info and order details
- Orders results by order_date to show chronology

📊 Insight Gained:

This gives TerpBuy:

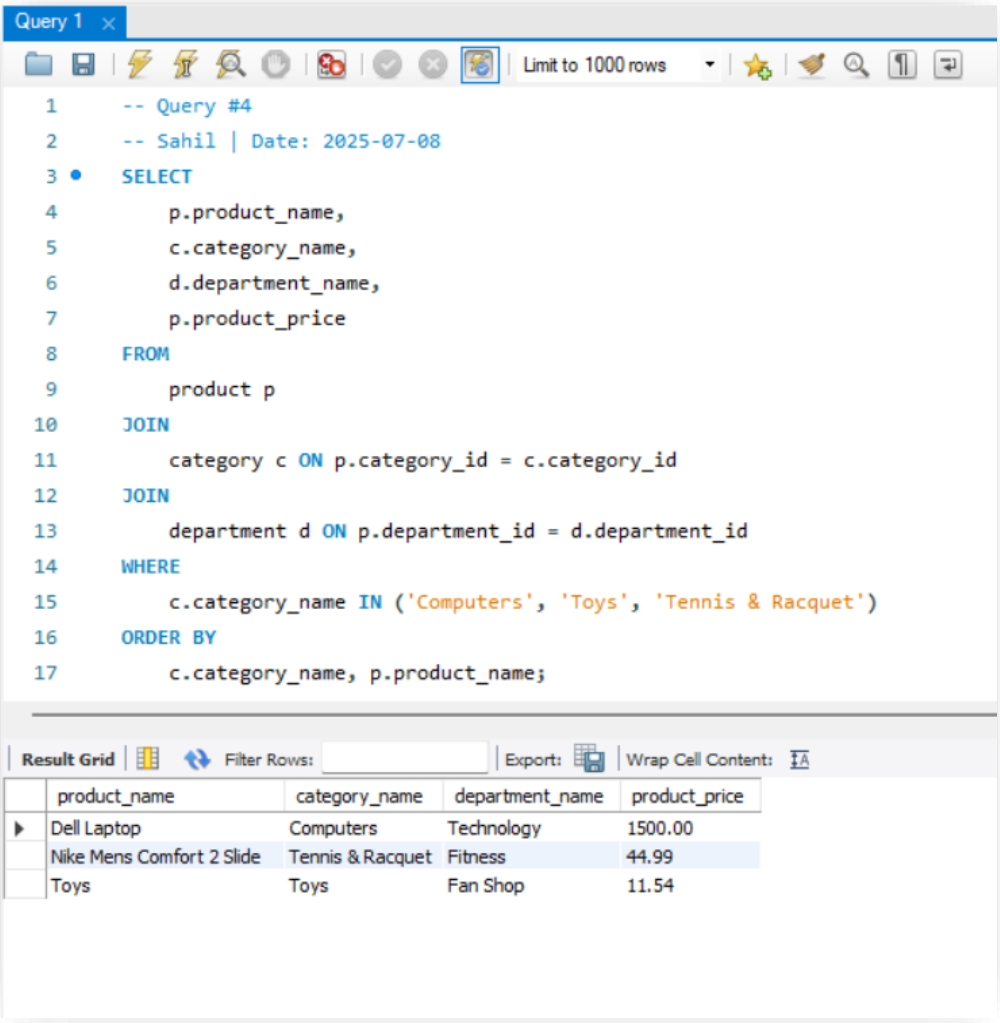
- A list of **all sales activity** in the Florida market
- Useful for **regional marketing**, tracking **sales performance**, or **targeting** specific customer segments in FL

✅🔎 Question Query 4

List all products that fall in one of the following categories:
'Computers', 'Toys', 'Tennis & Racquet'

Include:

- Product **name**
- **Category**
- **Department**
- **Price**



🧠 What This Query Does:

- Joins the product table with category and department to replace IDs with text values.
- Filters for just the three categories listed.
- Returns readable details for decision-making.

📊 Insight Gained:

This helps TerpBuy:

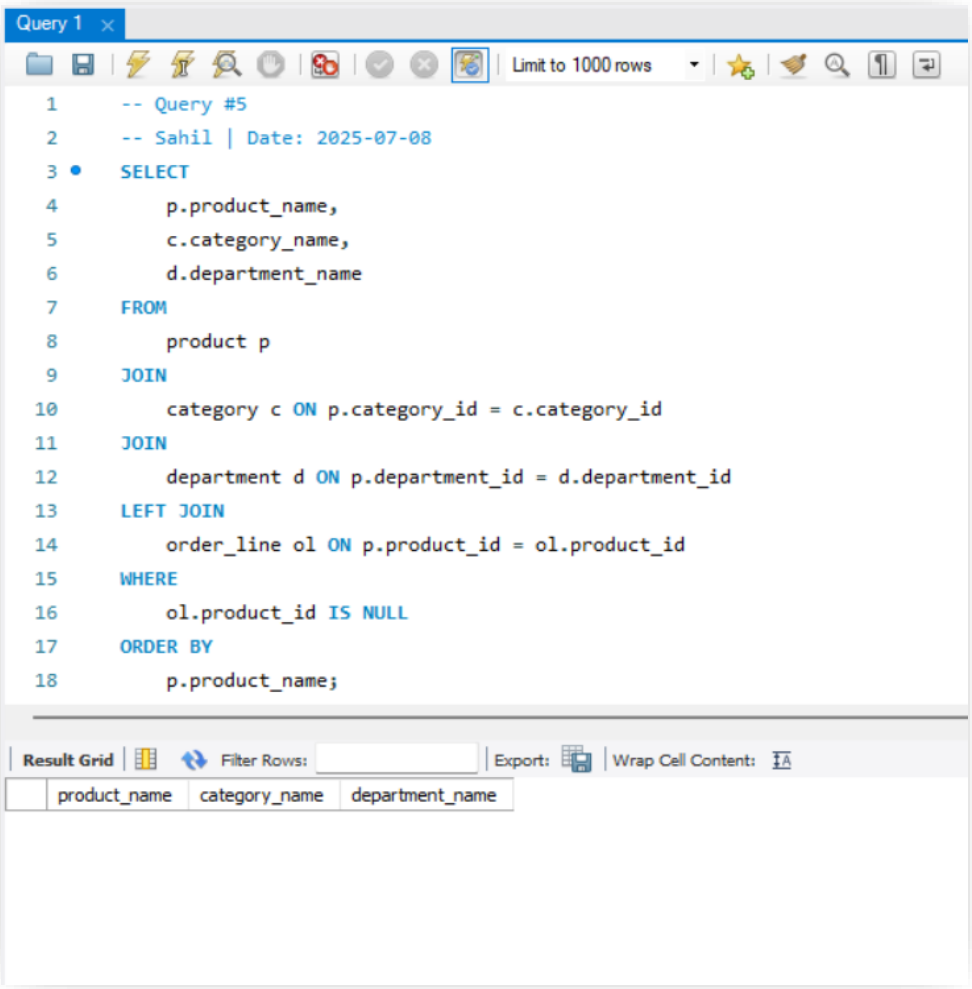
- Identify all products in **specific strategic categories**.
- See their **department context** and **pricing**.
- Make decisions for **marketing**, **bundle deals**, or **stock optimization**.

✅🔎 Question Query 5

TerpBuy is considering reducing its product offerings.
Which **products have not yet been sold?**

Include:

- Product **name**
- **Category**
- **Department**



🧠 What This Query Does:

- Starts with all products.
- Performs a LEFT JOIN to order_line (where product sales are logged).
- Filters to **only those products with no match** (i.e., not sold).
- Joins with category and department for full readable context.

📊 Insight Gained:

This gives TerpBuy:

- A **clear list of unsold inventory**.
- Opportunity to **reduce, replace, or promote** low-performing products.
- A better understanding of **category-wise underperformers**.

✅🔎 Question Query 6

List the names of all **cities from where orders are shipped**.
For each city, find the **number of orders where shipping was delayed**.

✅ Sort the list of cities from the **highest to lowest** number of delayed shipping orders.

🔍 Definition:

Shipping is **delayed** if:
actual_shipping_days > scheduled_shipping_days

Query 1

Limit to 1000 rows

```sql
-- Query #6
-- Sahil | Date: 2025-07-08
SELECT
    o.order_city,
    COUNT(*) AS delayed_orders
FROM
    orders o
WHERE
    o.actual_shipping_days > o.scheduled_shipping_days
GROUP BY
    o.order_city
ORDER BY
    delayed_orders DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_city	delayed_orders
Bangalore	51
Mumbai	45
Pune	41
Delhi	37
Chennai	32
Surat	31
Visakhapatnam	30
Hyderabad	29
Gorakhpur	27
Ajmer	26
Agra	25
Kanpur	24
Jaipur	24
Aurangabad	21
Ludhiana	21
Raipur	20

Result 3

🧠 What This Query Does:

- Filters only those orders that were shipped **late**.
- Groups results by order_city.
- Counts the number of delayed orders for each city.
- Sorts by the **most delayed cities first**.

📊 **Insight Gained:**

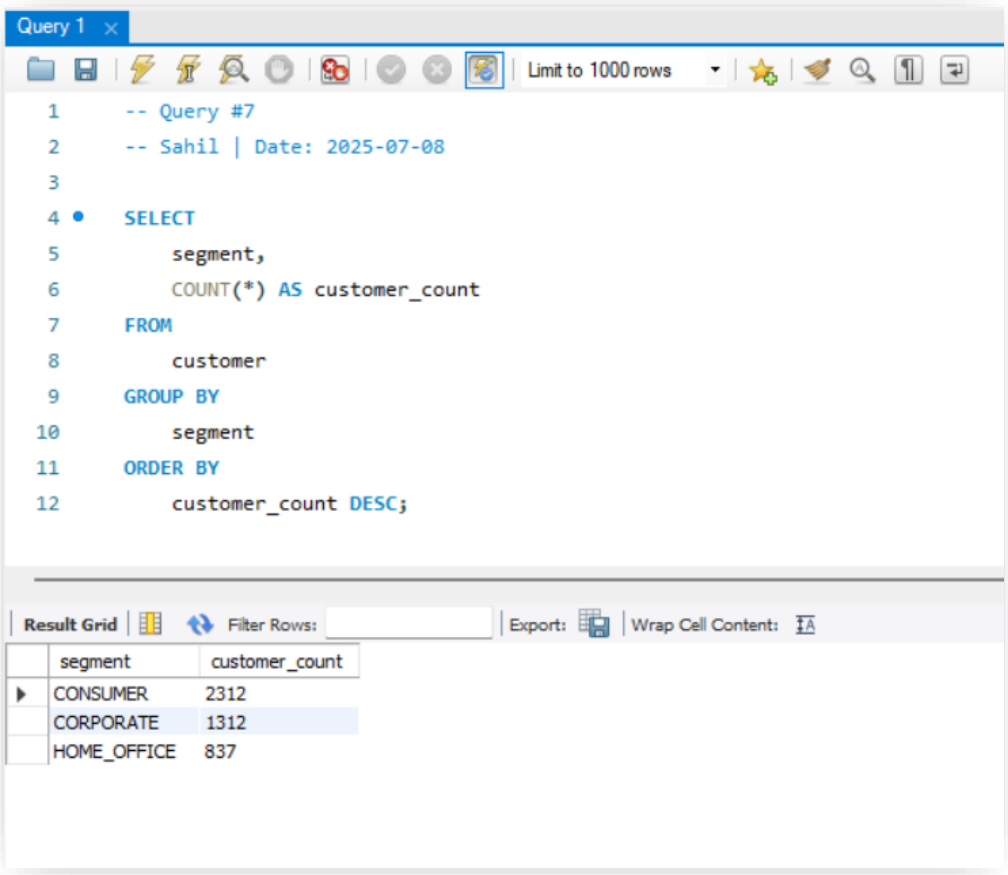
This helps TerpBuy:

- Identify **problematic shipping locations**.
- Analyze whether delays are due to **logistics**, **distance**, or **supplier-side bottlenecks**.
- Improve customer satisfaction by addressing delay hotspots.

✅🔎 **Question Query 7**

How many customers are there in each **segment**? Show the **most popular segment at the top**.

✅ Use a **column alias** in the result.



🧠 **What This Query Does:**

- Groups customers by segment (e.g., Consumer, Corporate, Home Office, etc.).
- Counts how many customers belong to each segment.
- Uses AS customer_count as a **column alias**.
- Sorts to show the **largest segment first**.

📊 **Insight Gained:**

This gives TerpBuy:

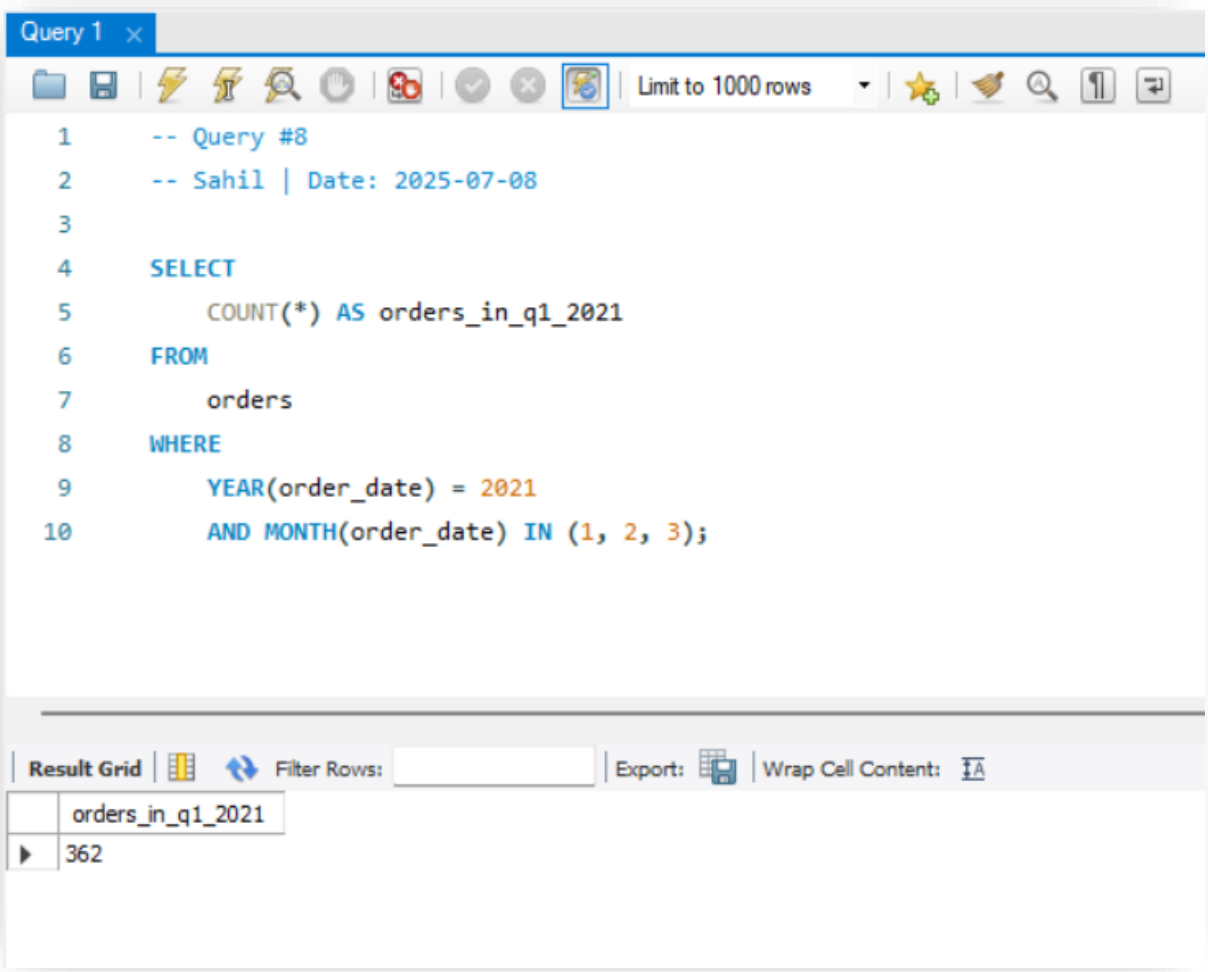
- A snapshot of **which customer segments dominate** the user base.
- Insights to focus marketing or product targeting toward the **largest segment**.

✅ 🔎 **Question Query 8**

How many orders were placed in the **first quarter of 2021**?

🗓️ (Q1 = January, February, March)

✅ Use a **column alias** in the result.



🧠 **What This Query Does:**

- Filters only those orders placed in the **year 2021**.
- Then filters down to the **first quarter**: January, February, March.
- Uses a column alias orders_in_q1_2021 for clarity.

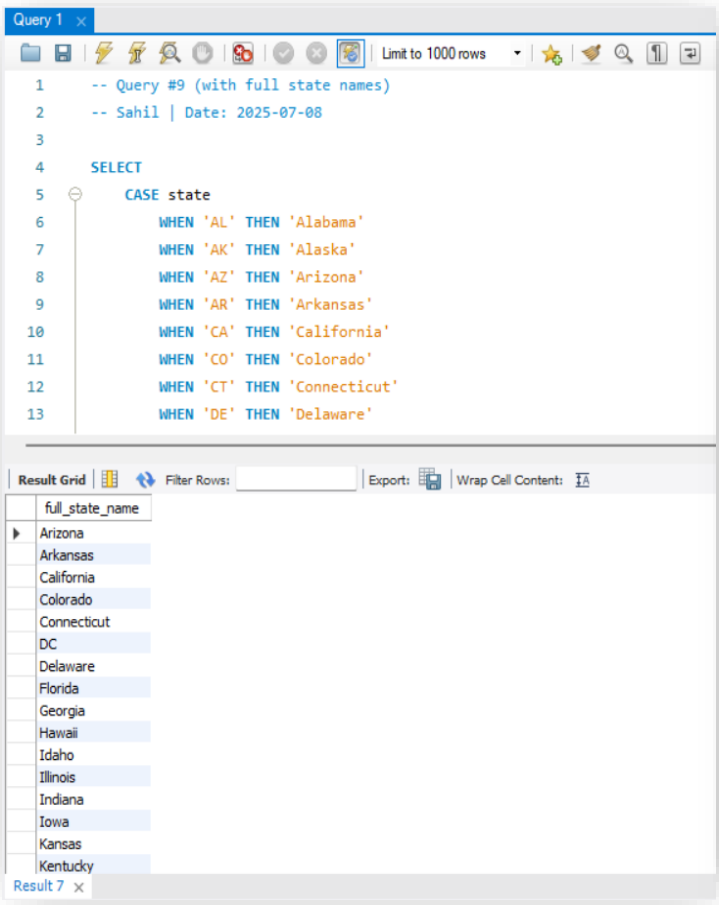
📊 **Insight Gained:**

This gives TerpBuy:

- A clear metric of **sales volume for Q1 2021**.
- Useful for comparing against other quarters or assessing **seasonal performance**.

✅🔎 Question Query 9

List in **alphabetical order** all **states** that support **multiple customer segments**.



```sql
-- Query #9 (with full state names)
-- Sahil | Date: 2025-07-08

SELECT
    CASE state
        WHEN 'AL' THEN 'Alabama'
        WHEN 'AK' THEN 'Alaska'
        WHEN 'AZ' THEN 'Arizona'
        WHEN 'AR' THEN 'Arkansas'
        WHEN 'CA' THEN 'California'
        WHEN 'CO' THEN 'Colorado'
        WHEN 'CT' THEN 'Connecticut'
        WHEN 'DE' THEN 'Delaware'
        WHEN 'WI' THEN 'Wisconsin'
        WHEN 'WV' THEN 'West Virginia'
        WHEN 'WY' THEN 'Wyoming'
        ELSE state
    END AS full_state_name
FROM
    customer
GROUP BY
    state
HAVING
    COUNT(DISTINCT segment) > 1
ORDER BY
    full_state_name;
```

full_state_name
Arizona
Arkansas
California
Colorado
Connecticut
DC
Delaware
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky

🧠 What This Query Does:

- Groups customers by state
- Uses COUNT(DISTINCT segment) to find how many **unique segments** exist in each state
- Filters to only those states with **more than one segment**
- Sorts the list **alphabetically**

📊 Insight Gained:

This helps TerpBuy:

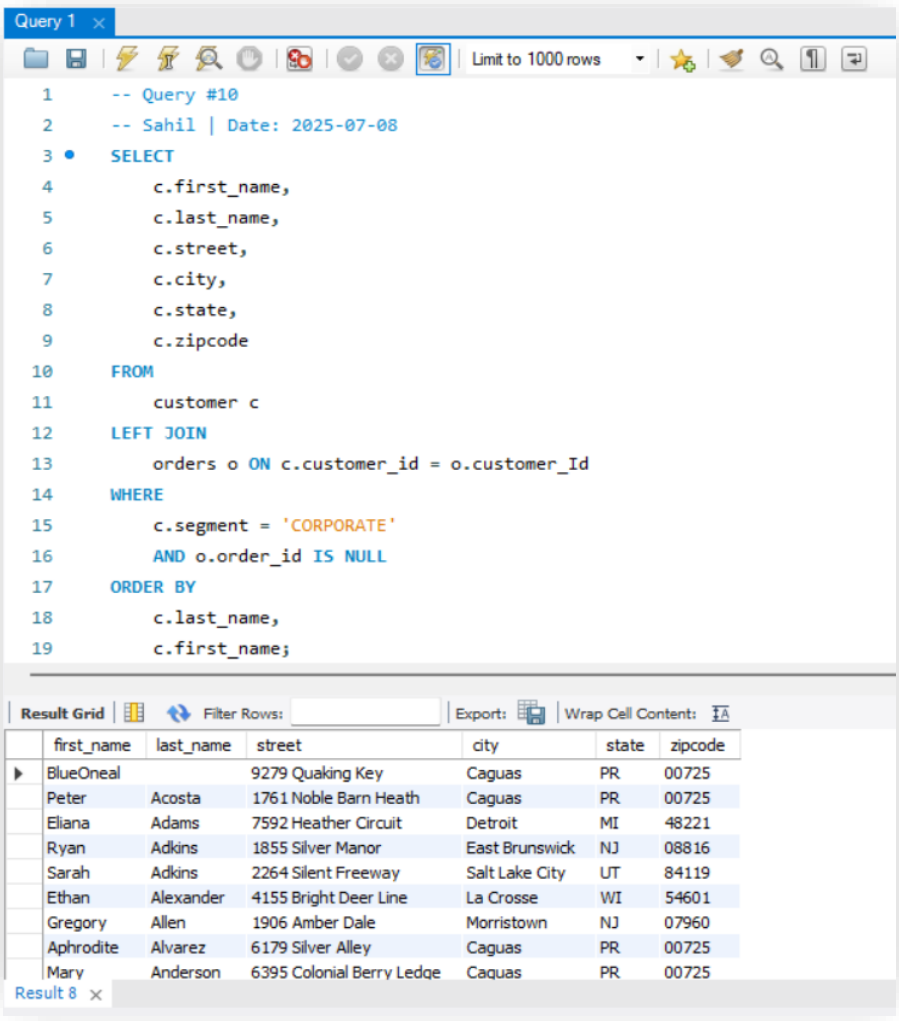
- Discover states with **diverse customer bases**
- Target marketing efforts in states that cater to **both corporate and consumer clients**
- Plan **multi-segment promotions** where both types coexist

✅ 🔎 **Question Query 10**

To help the commercial sales department with its marketing,
Find all **customers in the corporate segment** who have **not placed any orders** Include:

- First name
- Last name
- Street
- City
- State
- Zip code

✅ Sort by **last name**, then **first name**



🧠 **What This Query Does:**

- Filters for **Corporate segment** customers only
- Performs a LEFT JOIN with orders
- Uses o.order_id IS NULL to find **those who haven't placed any orders**
- Outputs clean customer contact info for **outreach**
- Sorted by name for easy lookup

📊 **Insight Gained:**

This gives TerpBuy:

- A **qualified lead list** for the commercial sales team
- Opportunities to send **exclusive promotions**, follow-ups, or **onboarding offers**
- The ability to **activate dormant customers** with zero sales so far

✅ 🔎 **Question Query 11**

The product **Nike Men's Free 5.0+ Running Shoe** has been recalled.

TerpBuy wants to offer **one discount coupon per customer** who **purchased it**.

Find all **orders** that included this product, and return:

- Customer **first name**, **last name**
- **Street**, **state**, **zip code**
- **Order date**

✅ Do **not repeat customers** — only **one row per customer**

Query 1

Limit to 1000 rows

```sql
-- Query #11
-- Sahil | Date: 2025-07-08
SELECT DISTINCT
    c.first_name,
    c.last_name,
    c.street,
    c.state,
    c.zipcode,
    o.order_date
FROM
    customer c
JOIN
    orders o ON c.customer_id = o.customer_Id
JOIN
    order_line ol ON o.order_id = ol.order_id
JOIN
    product p ON ol.product_id = p.product_id
WHERE
    p.product_name = 'Nike Mens Free 5.0+ Running Shoe'
ORDER BY
    c.last_name,
    c.first_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	street	state	zipcode	order_date
Mary	Adkins	379 Honey Maze	PR	00725	2019-02-22
Mary	Aguilar	1320 Silver Embers Villas	CA	91764	2018-12-01
Linda	Allen	1173 Iron Terrace	PR	00725	2019-02-04
Elizabeth	Andrews	6864 Tawny Bluff Forest	NJ	08540	2018-11-10
Stephanie	Archer	2749 Dusty Orchard	PR	00725	2018-12-30
Mary	Arnold	1757 Grand Blossom Subdivision	PR	00725	2018-12-29
Mary	Barnes	7910 Cotton Branch Highway	PR	00725	2018-12-12

Result 9

🧠 **What This Query Does:**

- Joins customer, orders, order_line, and product tables
- Filters only the product named **Nike Mens Free 5.0+ Running Shoe**
- Selects one row **per customer** using DISTINCT
- Gives all needed shipping and contact info
- Sorted by name for easy referencing

📊 **Insight Gained:**

This helps TerpBuy:

- Quickly **identify impacted customers** from the recall
- Send out **targeted discount coupons**
- Avoid duplicates by ensuring only **one entry per customer**

✅🔎 Question Query 12

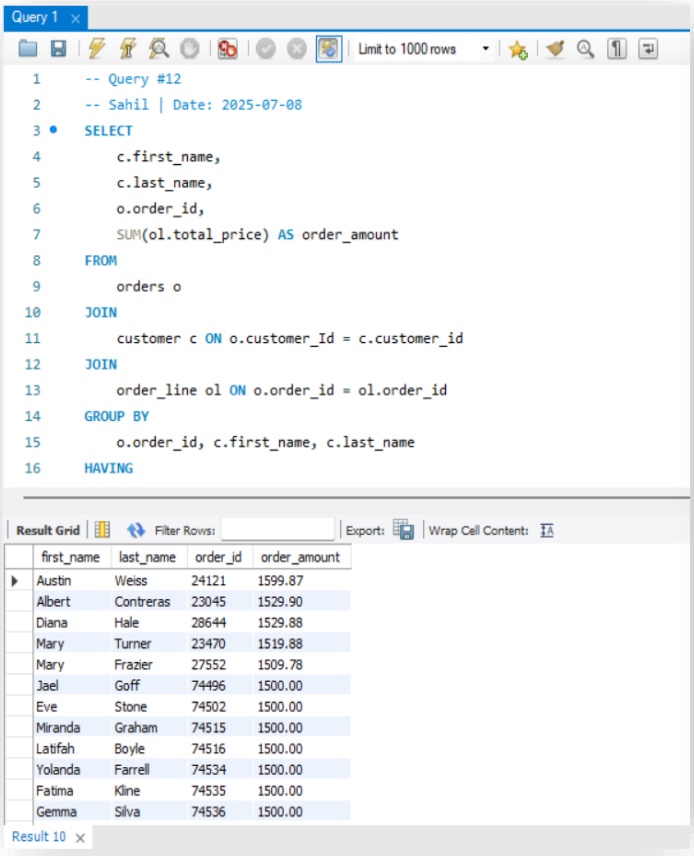
Premium customers are those who placed **orders with amounts greater than the average order amount**.

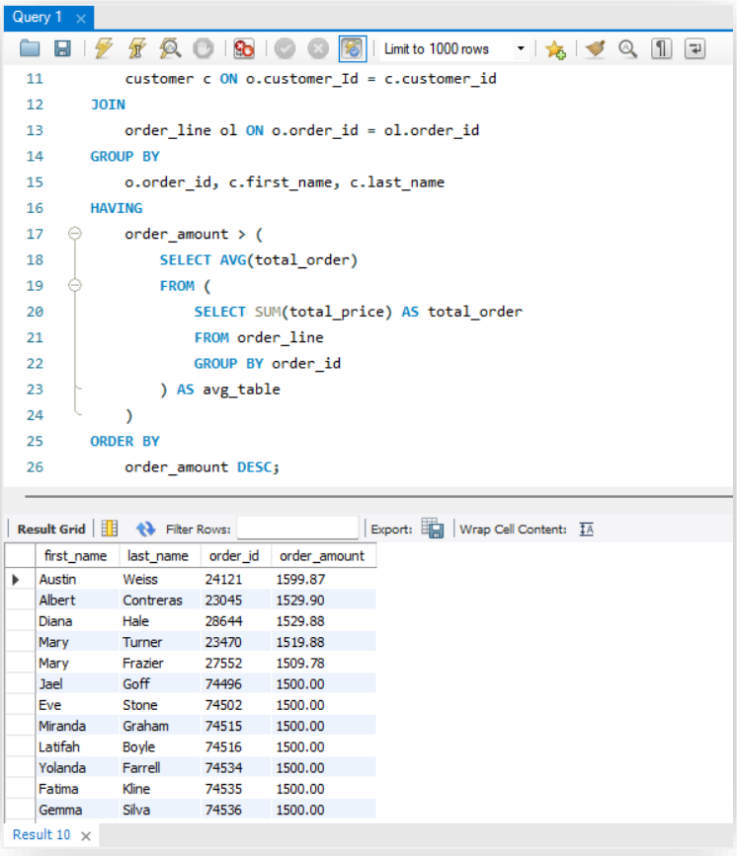
For each such order, find:

- Customer **first name**, **last name**
- The **order amount**

🧠 Prep Step:

- We'll use the order_line table to **calculate total order amounts**.
- Then compute the **average**.
- Return only the orders **above average**.
- Include the customer's name using a join.





🧠 What This Query Does:

- Joins customers, orders, and order lines.
- Calculates total order amount per order.
- Uses a subquery to compute the **average order amount**.
- Filters to keep only **above-average** orders.
- Sorts by the highest order amounts first.

📊 **Insight Gained:**

This identifies **high-spending customers**, which helps TerpBuy:

- Offer loyalty rewards or upsell opportunities.
- Segment premium clients for **targeted VIP campaigns**.
- Understand what price ranges trigger strong spending.

✅ **End Note**

This project has given me the opportunity to apply my SQL skills in a real-world business context. Through a structured approach, I explored TerpBuy's customer behaviors, product performance, shipping operations, and sales trends — all directly from a relational database.

Each query was carefully designed to not only meet technical requirements but also deliver meaningful business insights that management can act upon. From identifying premium customers and unsold inventory, to detecting shipping issues and analyzing customer segmentation, this assignment has shown how powerful structured query logic can be when tied to business strategy.

Working on this project has strengthened my confidence in writing advanced SQL queries, using joins effectively, filtering data with logic, and delivering data-driven conclusions. I've learned the value of not just *extracting* data, but also *understanding and presenting* it in ways that drive action.

I look forward to carrying this experience forward into more complex data projects, and I sincerely thank you for reviewing my work.

Prepared by: Sahil Verma "Data Analyst"
Date: July 21, 2025