

Name: Mansi Rane

Email ID: mansirane190601@gmail.com

Batch Date: 23/06/2025

Course Name: Data analytics (SQL)

Objective:

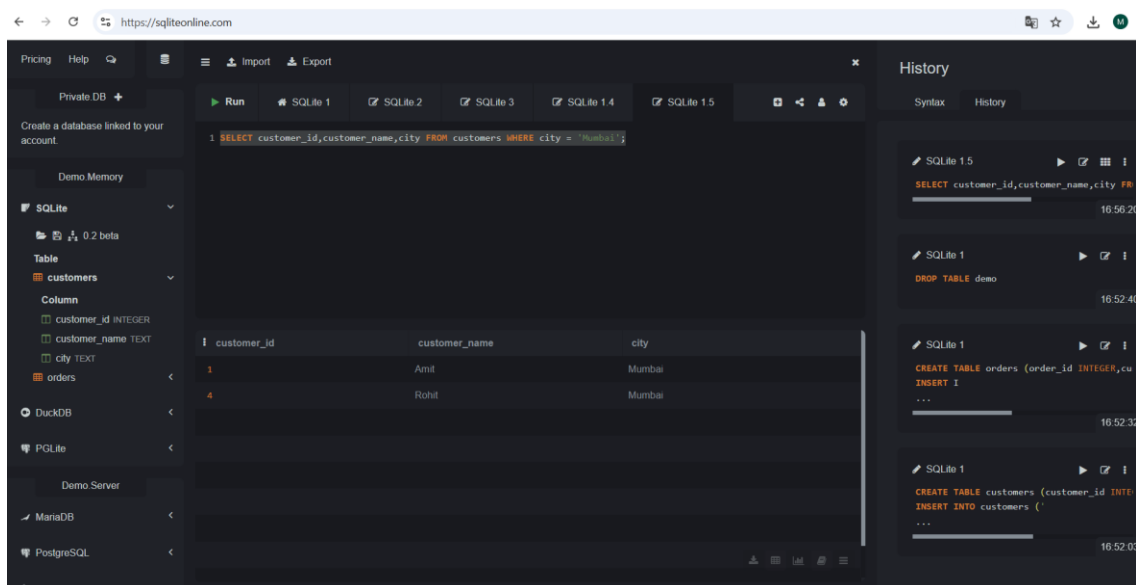
The objective of this project is to analyse sales data using SQL queries in SQLite and extract meaningful insights.

Dataset Description:

- **Tables used: customers, orders**
- **Data source: CSV file**
- **Tool: SQLite**

1) Retrieve customers belonging to a city Mumbai.

```
SELECT customer_id, customer_name, city  
FROM customers  
WHERE city = 'Mumbai';
```



2) Display orders with order amount greater than 3000

SELECT order_id, customer_id, amount

FROM orders

WHERE amount > 3000;

The screenshot shows the SQLiteOnline web interface. The SQL query entered is: `1 SELECT order_id,order_date,amount FROM orders WHERE amount>3000;`. The results are displayed in a table with the following data:

order_id	order_date	amount
101	2024-01-15	4500
103	2024-03-05	6000
105	2024-05-18	7000
106	2024-06-25	4000

The interface also shows a sidebar with database options (SQLite, DuckDB, PGlite) and a history panel on the right.

3) List customers whose names start with 'A'

SELECT customer_id, customer_name

FROM customers

WHERE customer_name LIKE 'A%';

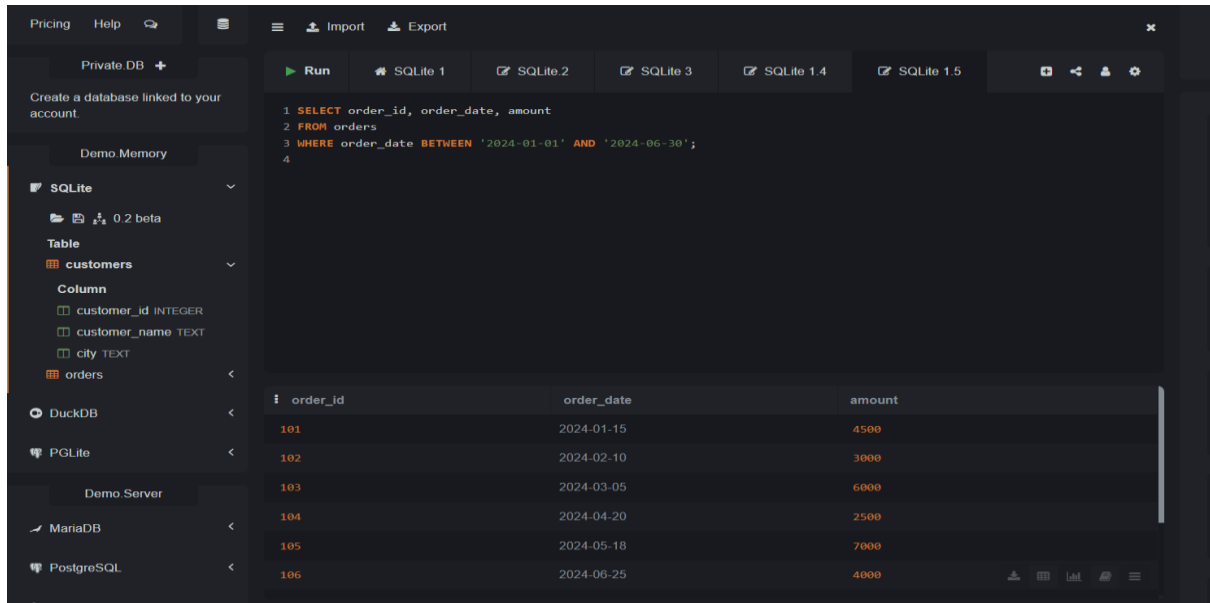
The screenshot shows the SQLiteOnline web interface. The SQL query entered is: `1 SELECT customer_name,customer_name FROM customers WHERE customer_name LIKE 'A%';`. The results are displayed in a table with the following data:

customer_name	customer_name
Amit	Amit
Anjali	Anjali
Aakash	Aakash

The interface also shows a sidebar with database options (SQLite, DuckDB, PGlite) and a history panel on the right.

4) Display orders within a specific date range

```
SELECT order_id, order_date, amount  
FROM orders  
WHERE order_date BETWEEN '2024-01-01' AND '2024-06-30';
```



The screenshot shows a database client interface with a dark theme. On the left, there is a sidebar with a tree view showing database connections: Private DB, Demo Memory, SQLite (0.2 beta), DuckDB, PGLite, Demo Server, MariaDB, PostgreSQL, and MS SQL. The 'SQLite' section is expanded, showing a table named 'orders' with columns: customer_id (INTEGER), customer_name (TEXT), city (TEXT), and amount (INTEGER). The main area displays a SQL query in a text editor:

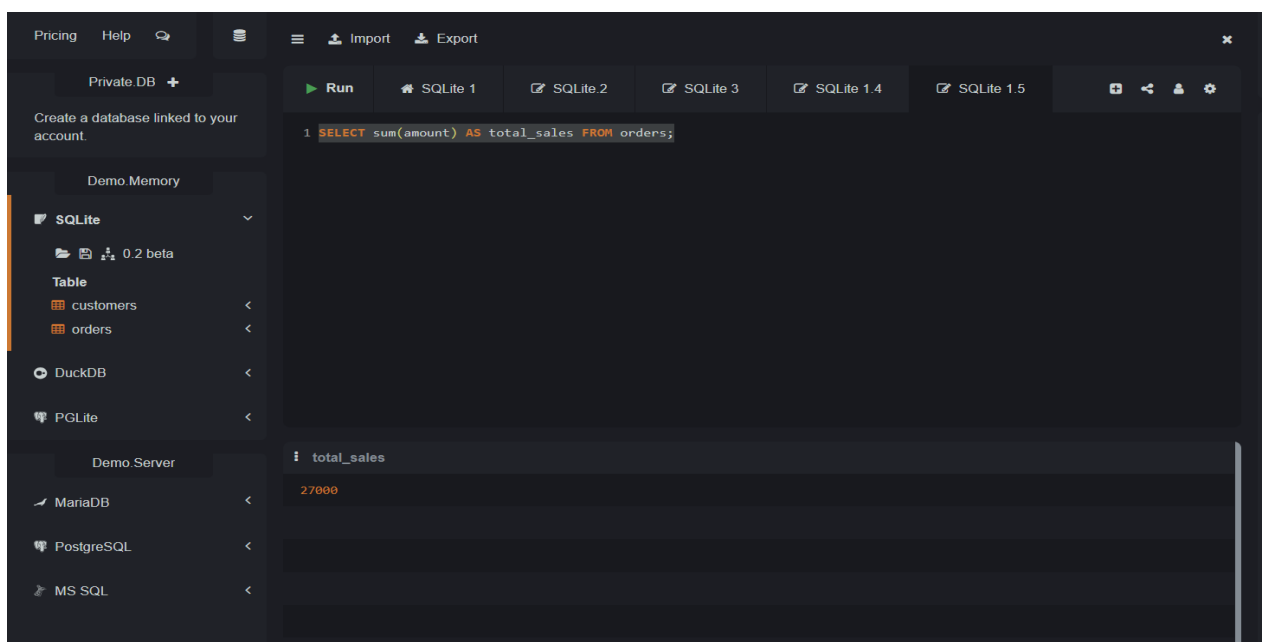
```
1 SELECT order_id, order_date, amount  
2 FROM orders  
3 WHERE order_date BETWEEN '2024-01-01' AND '2024-06-30';  
4
```

Below the query editor, the results are displayed in a table with the following data:

order_id	order_date	amount
101	2024-01-15	4500
102	2024-02-10	3000
103	2024-03-05	6000
104	2024-04-20	2500
105	2024-05-18	7000
106	2024-06-25	4000

5) Calculate total sales amount

```
SELECT SUM(amount) AS total_sales  
FROM orders;
```



The screenshot shows the same database client interface. The SQL query in the text editor is:

```
1 SELECT sum(amount) AS total_sales FROM orders;
```

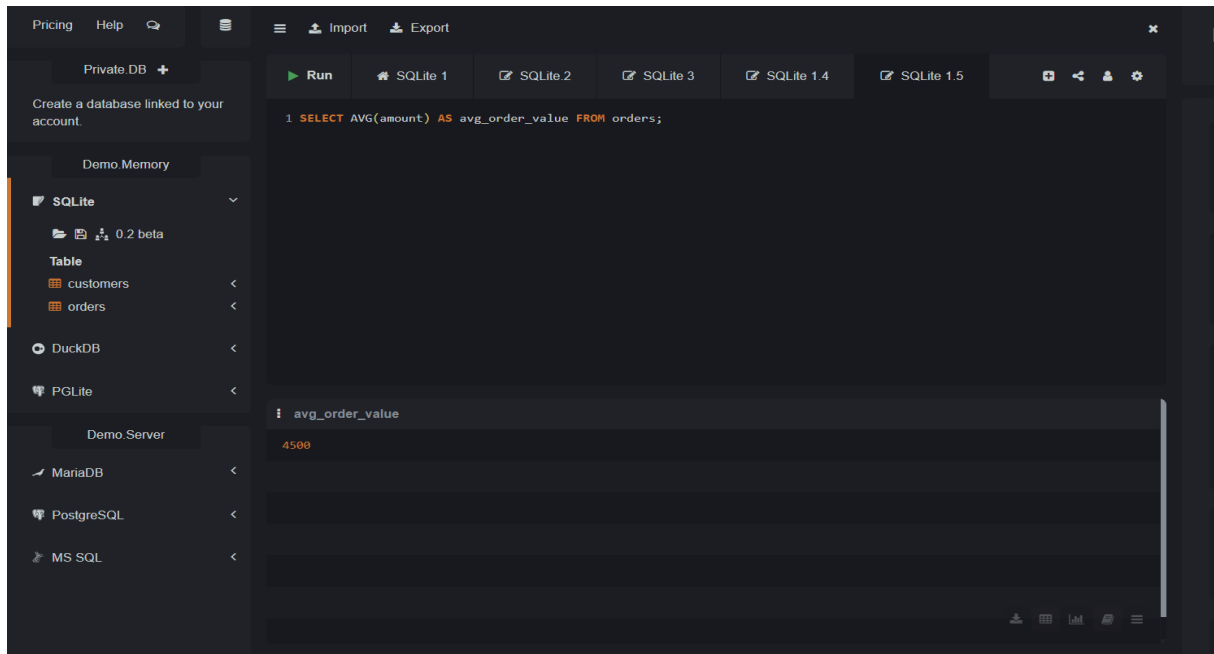
The results are displayed in a table with the following data:

total_sales
27000

6) Calculate average order value

```
SELECT AVG(amount) AS avg_order_value
```

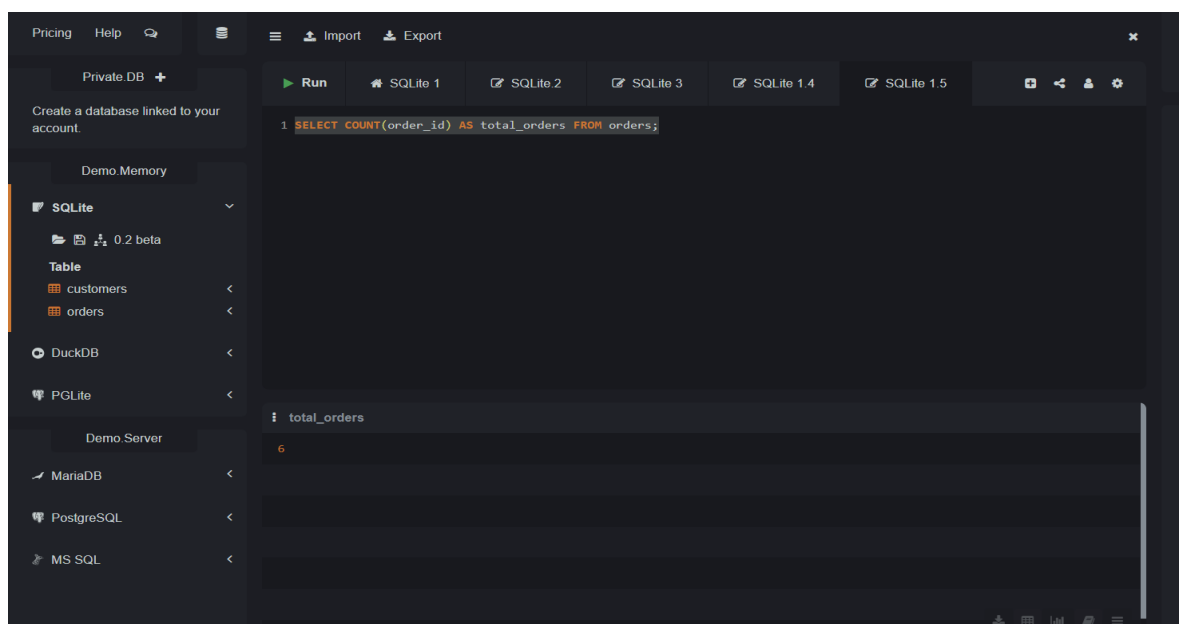
```
FROM orders;
```



7) Count total number of orders

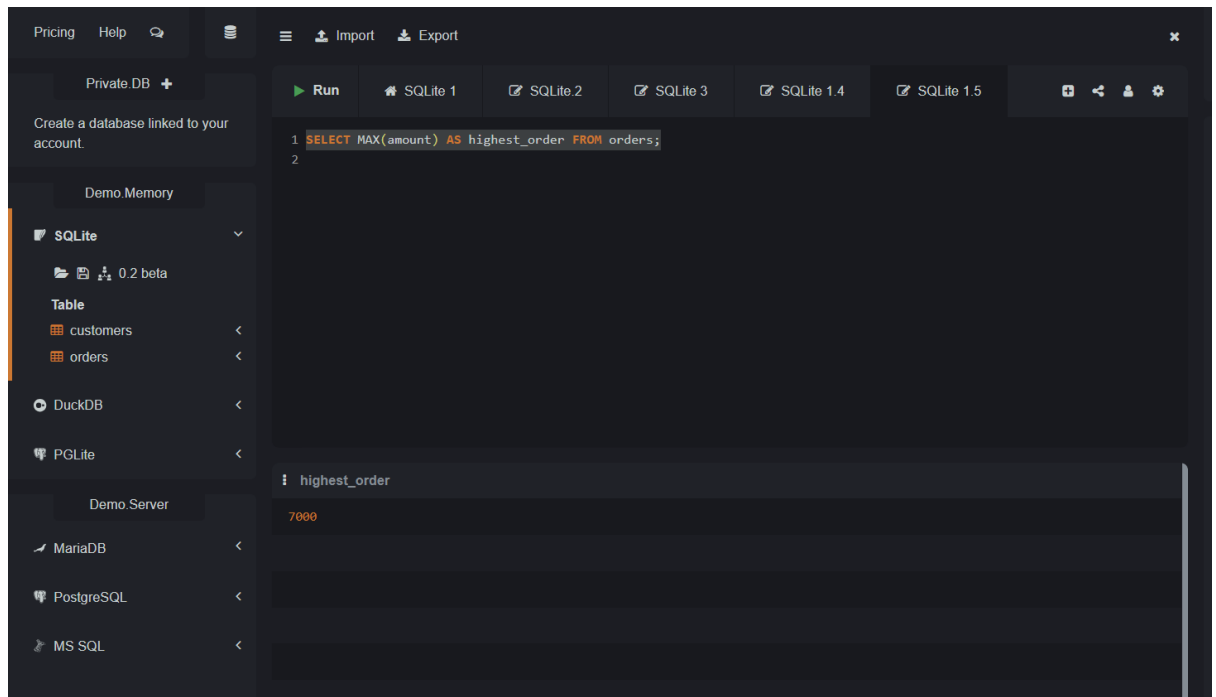
```
SELECT COUNT (order_id) AS total_orders
```

```
FROM orders;
```



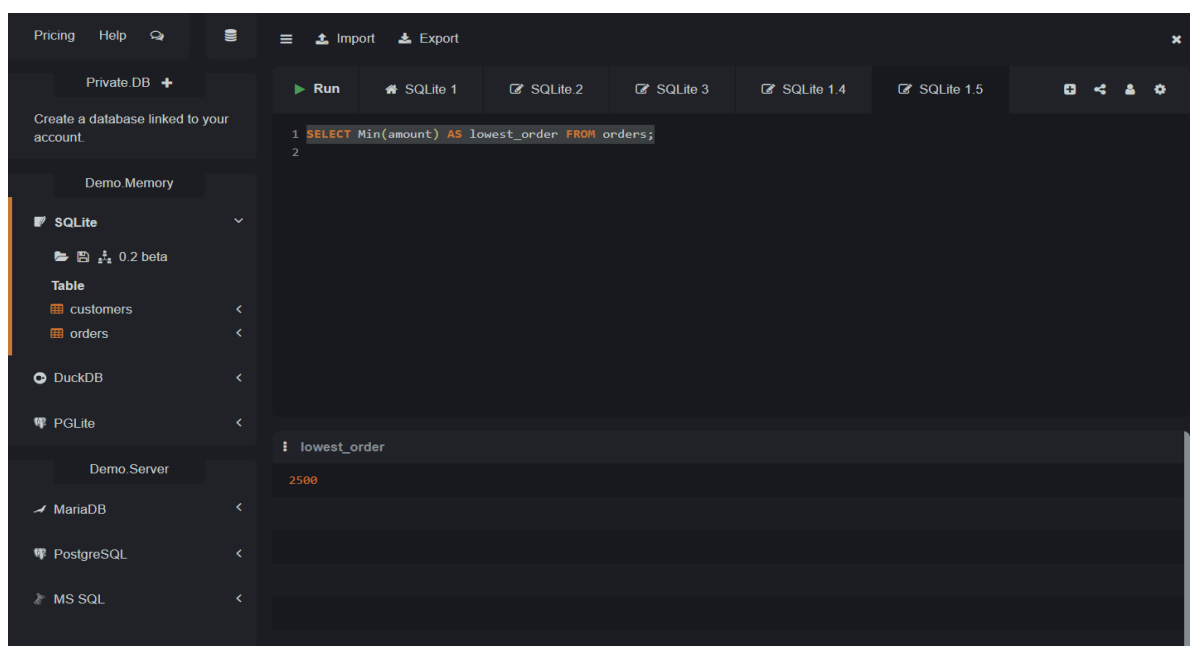
8) Identify the highest order value

```
SELECT MAX(amount) AS highest_order  
FROM orders;
```



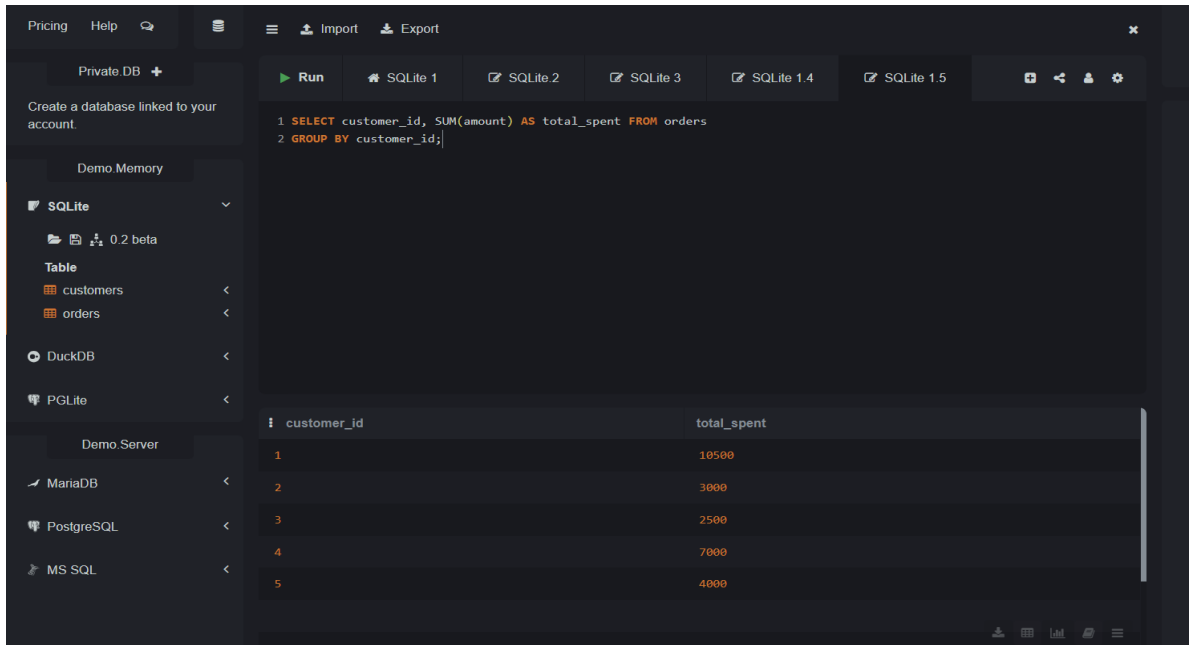
9) Identify the lowest order value

```
SELECT MIN (amount) AS lowest_order  
FROM orders;
```



10) Calculate total sales by each customer

```
SELECT customer_id, SUM (amount) AS total_spent
FROM orders
GROUP BY customer_id;
```

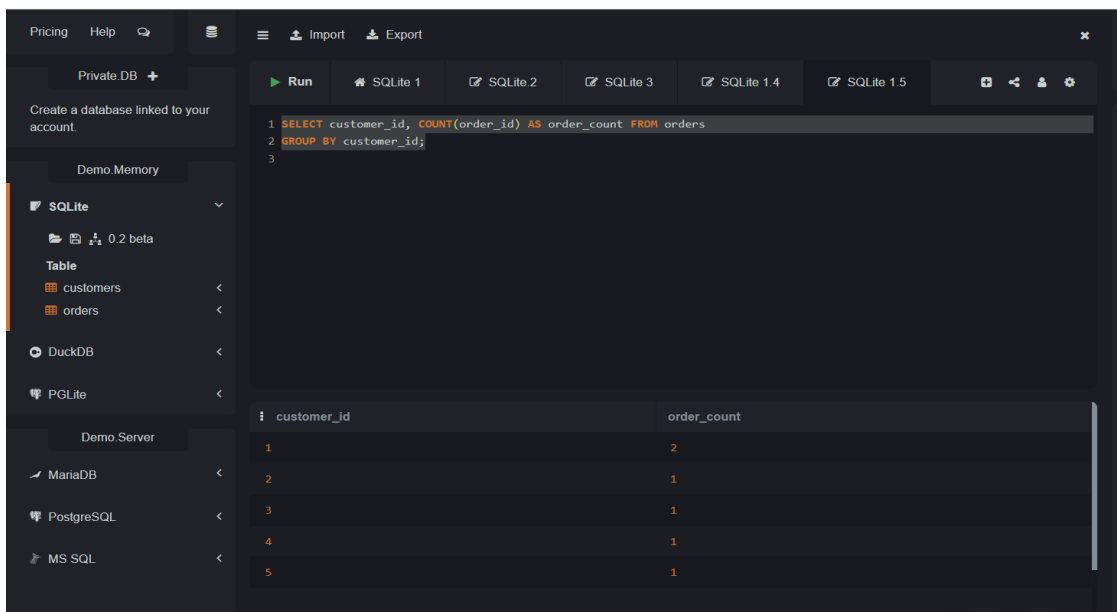


The screenshot shows a database client interface with a sidebar on the left containing a tree view of databases and tables. The main area displays a SQL query and its results. The query is: `1 SELECT customer_id, SUM(amount) AS total_spent FROM orders` and `2 GROUP BY customer_id;`. The results are shown in a table with two columns: `customer_id` and `total_spent`. The data is as follows:

customer_id	total_spent
1	10500
2	3000
3	2500
4	7000
5	4000

11) Calculate number of orders per customer

```
SELECT customer_id, COUNT (order_id) AS order_count
FROM orders
GROUP BY customer_id;
```



The screenshot shows the same database client interface as before, but with a different SQL query. The query is: `1 SELECT customer_id, COUNT(order_id) AS order_count FROM orders` and `2 GROUP BY customer_id;`. The results are shown in a table with two columns: `customer_id` and `order_count`. The data is as follows:

customer_id	order_count
1	2
2	1
3	1
4	1
5	1

12) Identify customers with total spending above 5000

```
SELECT customer_id, SUM (amount) AS total_spent  
FROM orders  
GROUP BY customer_id  
HAVING SUM (amount) > 5000;
```

The screenshot shows a database IDE interface. On the left, there is a sidebar with a tree view containing 'Private.DB +', 'Demo.Memory', 'SQLite' (expanded), 'DuckDB', 'PGLite', and 'Demo.Server'. Under 'SQLite', there are 'Table' entries for 'customers' and 'orders'. The main editor area displays a SQL query:

```
1 SELECT customer_id, SUM(amount) AS total_spent  
2 FROM orders  
3 GROUP BY customer_id  
4 HAVING SUM(amount) > 5000;  
5
```

 Below the query, the results are shown in a table with two columns: 'customer_id' and 'order_count'. The results are as follows:

customer_id	order_count
1	2
2	1
3	1
4	1
5	1

13) Retrieve customer names along with their order amounts

```
SELECT c. customer_name, o. amount
FROM customers c
JOIN orders o
ON c. customer_id = o. customer_id;
```

The screenshot shows a database client interface with a dark theme. On the left, there's a sidebar with 'Private DB +', 'Demo Memory', and 'Demo Server' sections. Under 'Demo Memory', 'SQLite' is selected, showing 'customers' and 'orders' tables. The main area displays a SQL query: `1 SELECT c.customer_name, o.amount`, `2 FROM customers c`, `3 INNER JOIN orders o`, `4 ON c.customer_id = o.customer_id;`. Below the query, the results are shown in a table with two columns: 'customer_name' and 'amount'. The data rows are: Amit (4500), Amit (6000), Neha (3000), Anjali (2500), Rohit (7000), and Aakash (4000).

customer_name	amount
Amit	4500
Amit	6000
Neha	3000
Anjali	2500
Rohit	7000
Aakash	4000

14) Display customer name, city, and order date

```
SELECT c. customer_name, c. city, o. order_date
FROM customers c
JOIN orders o
ON c. customer_id = o. customer_id;
```

The screenshot shows the same database client interface as before, but with a different SQL query: `1 SELECT c.customer_name,city, o.order_date`, `2 FROM customers c`, `3 INNER JOIN orders o`, `4 ON c.customer_id = o.customer_id;`. The results table now has three columns: 'customer_name', 'city', and 'order_date'. The data rows are: Amit (Mumbai, 2024-01-15), Amit (Mumbai, 2024-03-05), Neha (Pune, 2024-02-10), Anjali (Delhi, 2024-04-20), Rohit (Mumbai, 2024-05-18), and Aakash (Bangalore, 2024-06-25).

customer_name	city	order_date
Amit	Mumbai	2024-01-15
Amit	Mumbai	2024-03-05
Neha	Pune	2024-02-10
Anjali	Delhi	2024-04-20
Rohit	Mumbai	2024-05-18
Aakash	Bangalore	2024-06-25

15) Calculate total sales by each city

SELECT c. city, SUM(o. amount) AS total_sales

FROM customers c

JOIN orders o

ON c. customer_id = o. customer_id

GROUP BY c. city;

The screenshot shows a database management tool interface with a dark theme. The top menu bar includes 'Pricing', 'Help', 'Import', and 'Export'. Below the menu, there are tabs for 'Private.DB +', 'Demo Memory', and 'Demo.Server'. The 'Demo Memory' tab is active, showing a list of databases: 'SQLite' (0.2 beta), 'Table', 'customers', 'orders', 'DuckDB', 'PGLite', 'MariaDB', 'PostgreSQL', and 'MS SQL'. The 'SQLite' database is selected, and the 'Table' view is active. The SQL query editor displays the following query:

```
1 SELECT c.city, sum(o.amount) AS total_sales
2 FROM customers c
3 INNER JOIN orders o
4 ON c.customer_id = o.customer_id;
```

The query results are displayed in a table with two columns: 'city' and 'total_sales'. The results show that the total sales for Mumbai are 27000.

city	total_sales
Mumbai	27000

16) Identify orders with amount higher than the average order value

```
SELECT order_id, amount
```

```
FROM orders
```

```
WHERE amount > (
```

```
    SELECT AVG(amount) FROM orders
```

```
);
```

The screenshot shows a database client interface with a dark theme. On the left sidebar, under 'Demo.Memory', the 'SQLite' section is expanded, showing a table named 'orders'. The main panel displays a SQL query in a text editor, which is then executed. The results are shown in a table below the query.

```
1 SELECT order_id, amount
2 FROM orders
3 WHERE amount > (
4     SELECT AVG(amount) FROM orders
5 );
```

order_id	amount
103	6000
105	7000

17) Identify customers who have placed at least one order

SELECT customer_id, customer_name

FROM customers

WHERE customer_id IN (

SELECT DISTINCT customer_id FROM orders

);

The screenshot shows a SQL IDE interface with a dark theme. On the left, there is a sidebar with a tree view containing 'Private.DB +', 'Demo.Memory', 'SQLite' (expanded), 'DuckDB', and 'PGLite'. Under 'SQLite', there are 'Table' entries for 'customers' and 'orders'. Below this is 'Demo.Server' with entries for 'MariaDB', 'PostgreSQL', and 'MS SQL'. The main editor area displays a SQL query:

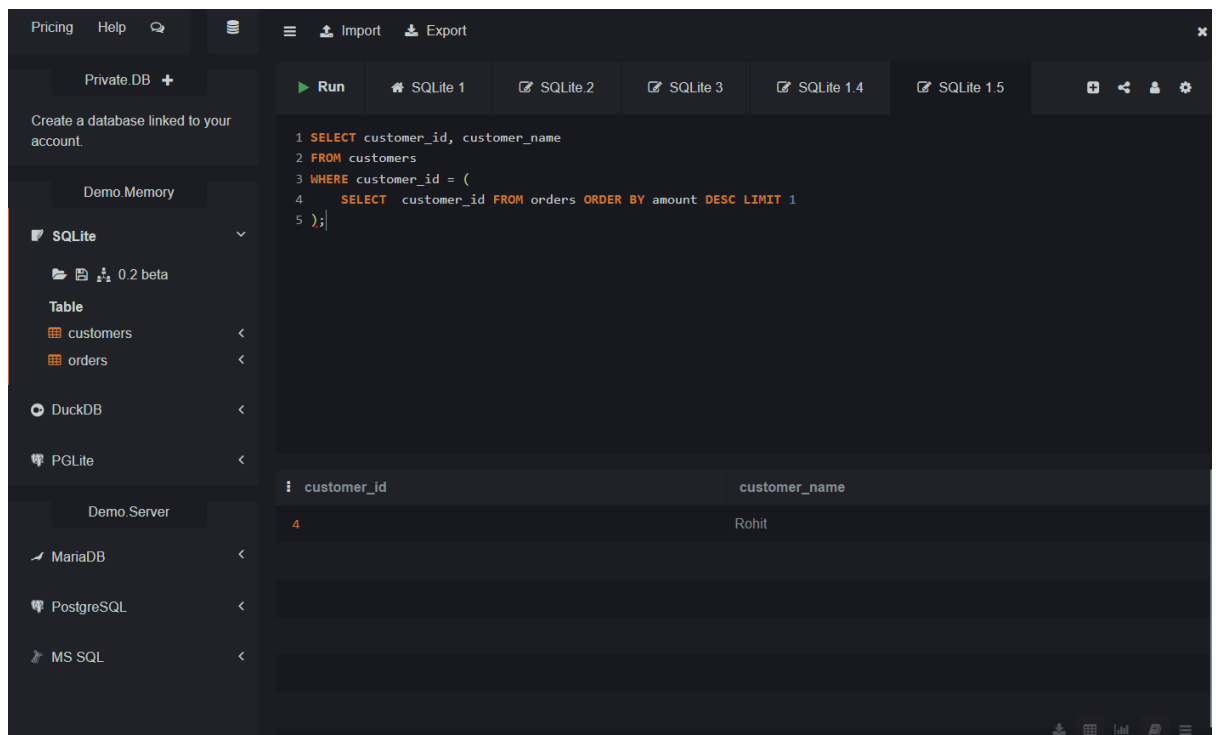
```
1 SELECT customer_id, customer_name
2 FROM customers
3 WHERE customer_id IN (
4   SELECT DISTINCT customer_id FROM orders
5 );
```

 Above the query is a 'Run' button and tabs for 'SQLite 1' through 'SQLite 1.5'. Below the query, the results are shown in a table with two columns: 'customer_id' and 'customer_name'. The table contains five rows of data.

customer_id	customer_name
1	Amit
2	Neha
3	Anjali
4	Rohit
5	Aakash

18) Identify the customer who placed the highest value order

```
SELECT customer_id, customer_name
FROM customers
WHERE customer_id = (
    SELECT customer_id
    FROM orders
    ORDER BY amount DESC
    LIMIT 1
);
```



19) Display all customers along with their order amounts (including customers with no orders).

```
SELECT c. customer_name, o. amount  
  
FROM customers c  
  
LEFT JOIN orders o  
  
ON c. customer_id = o. customer_id;
```

The screenshot shows a database management interface with a dark theme. On the left, there's a sidebar with a tree view containing 'Private DB', 'Demo.Memory', 'SQLite' (expanded), 'DuckDB', and 'PGLite'. Under 'SQLite', there are 'customers' and 'orders' tables. The main area displays a SQL query in a text editor, which has been executed. The results are shown in a table with two columns: 'customer_name' and 'amount'. The results list five customers: Amit (4500), Amit (6000), Neha (3000), Anjali (2500), and Rohit (7000). A sixth row for Aakash with an amount of 4000 is also visible at the bottom of the table.

```
1 SELECT c.customer_name,  
2     o.amount  
3 FROM customers c  
4 LEFT JOIN orders o  
5 ON c.customer_id = o.customer_id;
```

customer_name	amount
Amit	4500
Amit	6000
Neha	3000
Anjali	2500
Rohit	7000
Aakash	4000

20) Display orders having amount greater than the average order amount.

```
SELECT order_id, amount
FROM orders
WHERE amount > (
SELECT AVG(amount) FROM orders
);
```

The screenshot shows a database management tool interface. On the left, there is a sidebar with a tree view containing 'Private.DB +', 'Demo.Memory', 'SQLite' (expanded), 'DuckDB', 'PGLite', and 'Demo.Server'. Under 'SQLite', there are 'Table' entries for 'customers' and 'orders'. The main area displays a SQL query in a dark-themed editor:

```
1 SELECT order_id, amount
2 FROM orders
3 WHERE amount > (
4     SELECT AVG(amount) FROM orders
5 );
```

Below the query editor, the results are shown in a table with two columns: 'order_id' and 'amount'. The results are:

order_id	amount
103	6000
105	7000