

**Name: Mansi Rane**

**Email ID: [mansirane190601@gmail.com](mailto: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';
```

The screenshot shows the SQLiteOnline.com web application interface. On the left, there's a sidebar with database connections (Private DB, Demo Memory, SQLite, DuckDB, PGSQL, MariaDB, PostgreSQL, MS SQL) and a history section. The main area has tabs for Run, SQLite 1, SQLite 2, SQLite 3, SQLite 4, and SQLite 1.5. The Run tab contains the SQL query: `SELECT customer_id, customer_name, city FROM customers WHERE city = 'Mumbai';`. Below the query, the results are displayed in a table:

	customer_id	customer_name	city
1		Amit	Mumbai
4		Rohit	Mumbai

On the right, there's a History panel showing previous queries and their results. The queries listed are:

- SQLite 1.5: `SELECT customer_id, customer_name, city FROM customers WHERE city = 'Mumbai';` (executed at 16:56:20)
- SQLite 1: `DROP TABLE demo` (executed at 16:52:40)
- SQLite 1: `CREATE TABLE orders (order_id INTEGER, cu...` (executed at 16:52:32)
- SQLite 1: `CREATE TABLE customers (customer_id INTE...` (executed at 16:52:03)

## 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.com interface. The left sidebar lists databases: 'Private DB', 'Demo Memory' (SQLite 0.2 beta), 'Table' (customers, orders), 'DuckDB', 'PGLite', 'Demo Server' (MariaDB, PostgreSQL). The main area has tabs for 'Run' (SQLite 1, 2, 3, 4, 5) and 'History'. The 'Run' tab contains the SQL query: `SELECT order_id,order_date,amount FROM orders WHERE amount>3000;`. The results table shows four rows:

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 'History' tab on the right shows previous queries:

- SQLite 1.5: `SELECT order_id,order_date,amount FROM orders WHERE amount>3000;` (17:00:42)
- SQLite 1.5: `SELECT customer_id,customer_name,city FROM customers` (16:56:20)
- SQLite 1: `DROP TABLE demo` (16:52:40)
- SQLite 1: `CREATE TABLE orders (order_id INTEGER, customer_id INTEGER, order_date TEXT, amount REAL);` (16:52:32)
- SQLite 1: `CREATE TABLE customers (customer_id INTEGER, customer_name TEXT, city TEXT);` (16:52:32)

## 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.com interface. The left sidebar lists databases: 'Private DB', 'Demo Memory' (SQLite 0.2 beta), 'Table' (customers, orders), 'DuckDB', 'PGLite', 'Demo Server' (MariaDB, PostgreSQL). The main area has tabs for 'Run' (SQLite 1, 2, 3, 4, 5) and 'History'. The 'Run' tab contains the SQL query: `SELECT customer_name, customer_name FROM customers WHERE customer_name LIKE 'A%';`. The results table shows three rows:

customer_name	customer_name
Amit	Amit
Anjali	Anjali
Aakash	Aakash

The 'History' tab on the right shows previous queries:

- SQLite 1.5: `SELECT customer_name, customer_name FROM customers WHERE customer_name LIKE 'A%';` (17:03:43)
- SQLite 1.5: `SELECT order_id,order_date,amount FROM orders WHERE amount>3000;` (17:00:42)
- SQLite 1.5: `SELECT customer_id,customer_name,city FROM customers` (16:56:20)
- SQLite 1: `DROP TABLE demo` (16:52:40)
- SQLite 1: `CREATE TABLE orders (order_id INTEGER, customer_id INTEGER, order_date TEXT, amount REAL);` (16:52:32)
- SQLite 1: `CREATE TABLE customers (customer_id INTEGER, customer_name TEXT, city TEXT);` (16:52:32)

#### 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 interface with a dark theme. On the left, there's a sidebar with various database connections listed under "Demo Memory". Under "SQLite", there are tables "customers" and "orders". The "orders" table has columns: order\_id, order\_date, and amount. The results of the query are displayed in a table:

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 a database interface with a dark theme. On the left, there's a sidebar with various database connections listed under "Demo Memory". Under "SQLite", there are tables "customers" and "orders". The results of the query are displayed in a table:

total_sales
27000

## 6) Calculate average order value

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

```
FROM orders;
```

The screenshot shows a dark-themed SQLite database interface. On the left, there's a sidebar with a 'Private DB' section, a 'Demo.Memory' section containing a 'SQLite' entry with '0.2 beta' and 'Table' sections for 'customers' and 'orders', and a 'Demo.Server' section with 'MariaDB', 'PostgreSQL', and 'MS SQL'. The main area has tabs for 'Run', 'SQLite 1', 'SQLite 2', 'SQLite 3', 'SQLite 1.4', and 'SQLite 1.5'. A query window contains the SQL command: 'SELECT AVG(amount) AS avg\_order\_value FROM orders;'. Below the query, the result is shown: 'avg\_order\_value' with a value of '4500'. At the bottom, there are several small icons.

## 7) Count total number of orders

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

```
FROM orders;
```

This screenshot is identical to the one above, showing the same interface and the execution of the query 'SELECT COUNT(order\_id) AS total\_orders FROM orders;'. The result is displayed as 'total\_orders' with a value of '6'. The interface includes the same sidebar, tabs, and bottom icons.

## 8) Identify the highest order value

```
SELECT MAX(amount) AS highest_order
```

```
FROM orders;
```

The screenshot shows a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private DB** dropdown with a plus icon.
- Demo Memory** section.
- SQLite** connection selected.
- Table** section showing **customers** and **orders**.
- DuckDB**, **PGLite**, **Demo Server**, **MariaDB**, **PostgreSQL**, **MS SQL** sections.
- Run** button.
- SQLite 1** through **SQLite 1.5** tabs.
- Output Area** showing the query and its result:

```
1 SELECT MAX(amount) AS highest_order FROM orders;
2
3 : highest_order
4 7000
```

## 9) Identify the lowest order value

```
SELECT MIN(amount) AS lowest_order
```

```
FROM orders;
```

The screenshot shows a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private DB** dropdown with a plus icon.
- Demo Memory** section.
- SQLite** connection selected.
- Table** section showing **customers** and **orders**.
- DuckDB**, **PGLite**, **Demo Server**, **MariaDB**, **PostgreSQL**, **MS SQL** sections.
- Run** button.
- SQLite 1** through **SQLite 1.5** tabs.
- Output Area** showing the query and its result:

```
1 SELECT MIN(amount) AS lowest_order FROM orders;
2
3 : lowest_order
4 2500
```

## 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 SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private DB** dropdown.
- Demo Memory** section: **SQLite** (selected), **Table** view showing **customers** and **orders**.
- DuckDB**, **PGLite**, **Demo Server** sections.
- MariaDB**, **PostgreSQL**, **MS SQL** sections.
- Run** button.
- SQLite 1** tab selected.
- SQLite 2**, **SQLite 3**, **SQLite 1.4**, **SQLite 1.5** tabs.
- Customer ID** and **Total Spent** columns in the results table.
- Results table data:

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 a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private.DB** dropdown.
- Demo Memory** section: **SQLite** (selected), **Table** view showing **customers** and **orders**.
- DuckDB**, **PGLite**, **Demo Server** sections.
- MariaDB**, **PostgreSQL**, **MS SQL** sections.
- Run** button.
- SQLite 1** tab selected.
- SQLite 2**, **SQLite 3**, **SQLite 1.4**, **SQLite 1.5** tabs.
- Customer ID** and **Order Count** columns in the results table.
- Results table data:

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 SQLite database interface with the following details:

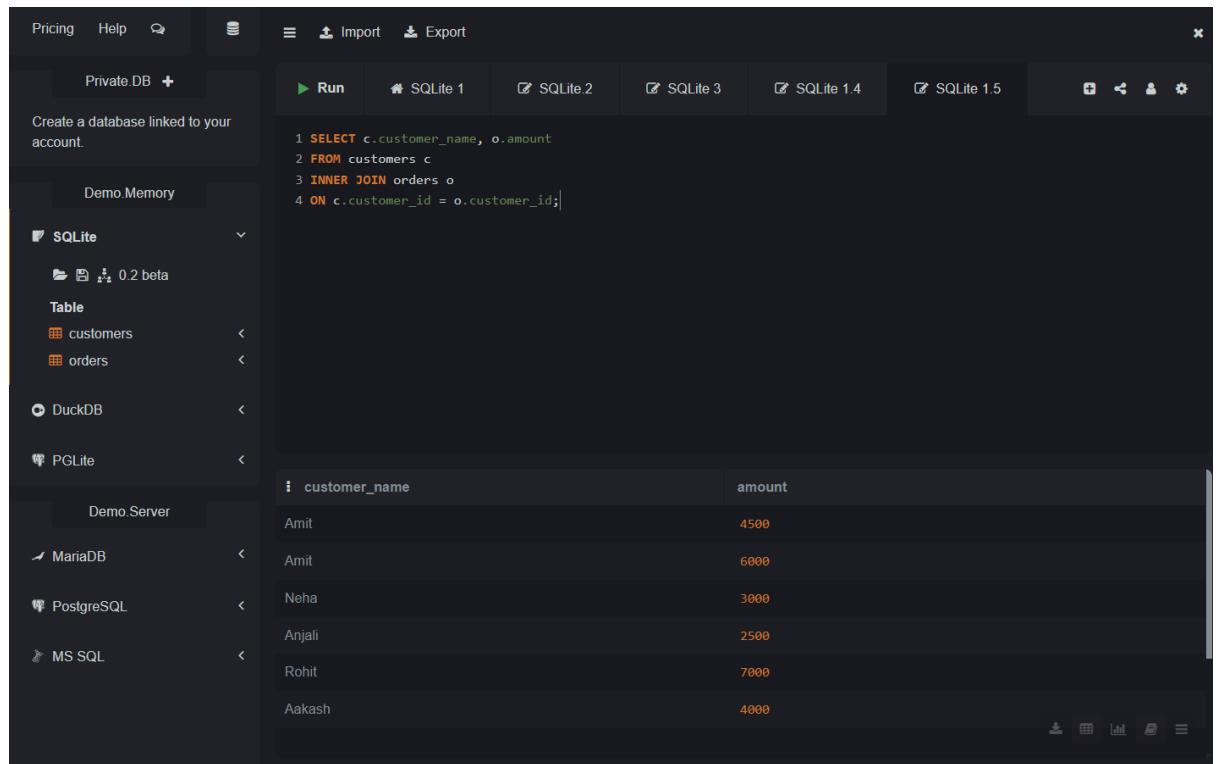
- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private DB +** button and "Create a database linked to your account." text.
- Demo.Memory** database selected.
- SQLite** tab selected, showing version 0.2 beta and tables **customers** and **orders**.
- DuckDB**, **PGLite**, **Demo.Server** (with **MariaDB**, **PostgreSQL**, **MS SQL**), and **PGFitter** tabs are also present.
- Run** button and **SQLite 1** tab selected.
- SQL query entered:

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

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;
```



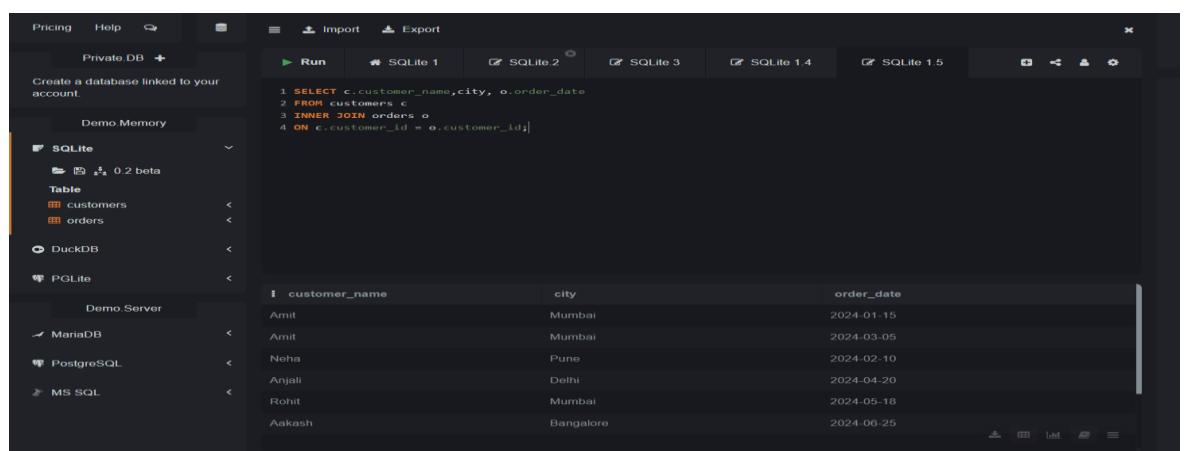
The screenshot shows a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private.DB +** button for creating a database.
- Demo.Memory** database selected.
- SQLite** connection selected.
- Tables**: **customers** and **orders**.
- Run** button is active.
- Results** tab shows the output of the query:

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;
```



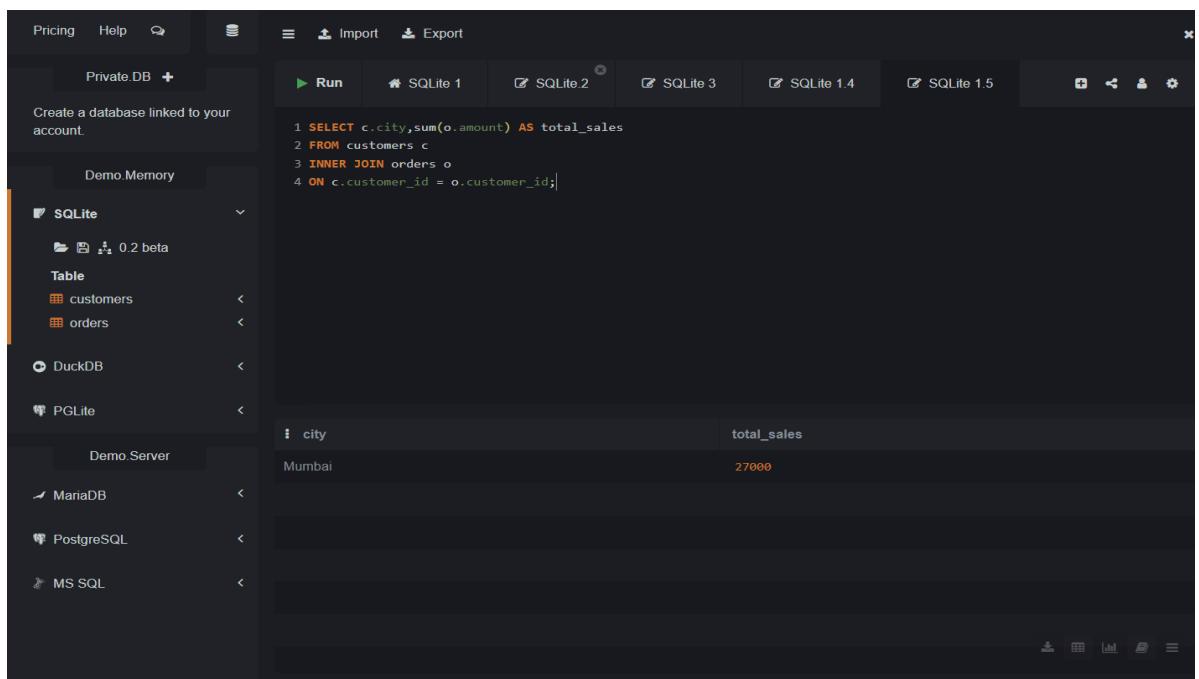
The screenshot shows a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private.DB +** button for creating a database.
- Demo.Memory** database selected.
- SQLite** connection selected.
- Tables**: **customers** and **orders**.
- Run** button is active.
- Results** tab shows the output of the query:

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 SQLite database interface with the following details:

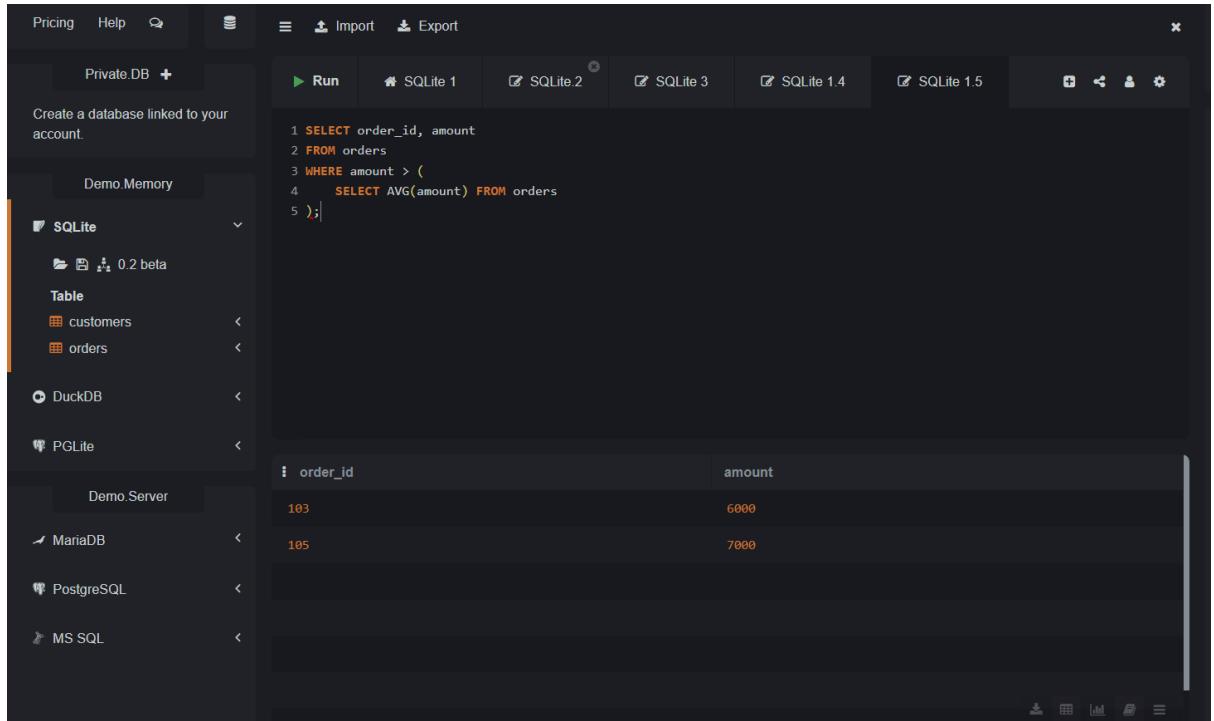
- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private DB +** button on the left.
- Create a database linked to your account.** text.
- Demo Memory** section.
- SQLite** section:
  - Table: **customers** and **orders**.
- DuckDB**, **PGLite**, **Demo Server**, **MariaDB**, **PostgreSQL**, **MS SQL** sections.
- Run** button and tabs for **SQLite 1**, **SQLite 2**, **SQLite 3**, **SQLite 1.4**, **SQLite 1.5**.
- SQL Editor**:

```
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;
```
- Results Table**:

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  
);
```



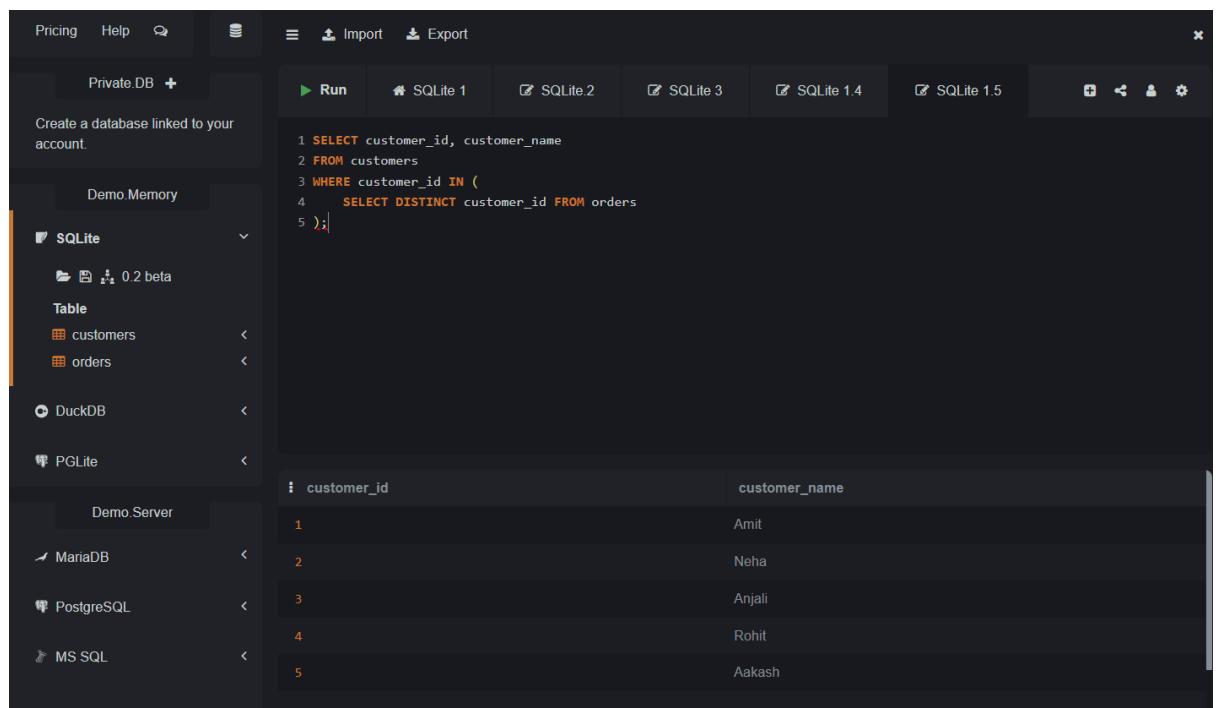
The screenshot shows a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** menu items.
- Run** button.
- SQLite 1** selected in the top navigation bar.
- SQLite 2**, **SQLite 3**, **SQLite 4**, **SQLite 5** options in the top navigation bar.
- Private.DB** and **+ Create a database linked to your account**.
- Demo Memory** section with **SQLite** selected, showing tables **customers** and **orders**.
- DuckDB**, **PGLite**, **Demo Server**, **MariaDB**, **PostgreSQL**, **MS SQL** sections.
- order\_id** and **amount** columns in the results table.
- Results table:

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 SQLite database interface with the following details:

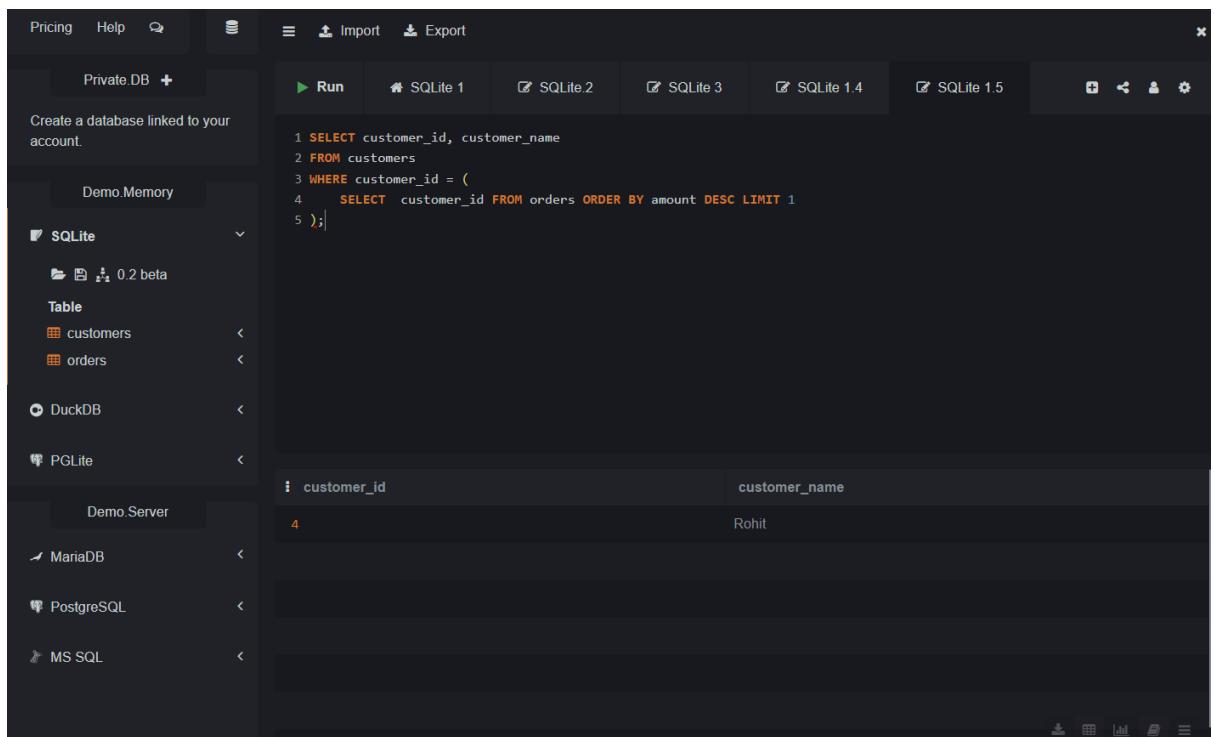
- Pricing**, **Help**, **Import**, **Export** buttons are at the top.
- A sidebar on the left lists databases: **Private.DB** (selected), **Demo Memory**, **SQLite** (selected), **DuckDB**, **PGLite**, **Demo Server**, **MariaDB**, **PostgreSQL**, and **MS SQL**.
- The main area shows the SQL query and its results:

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

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
);
```



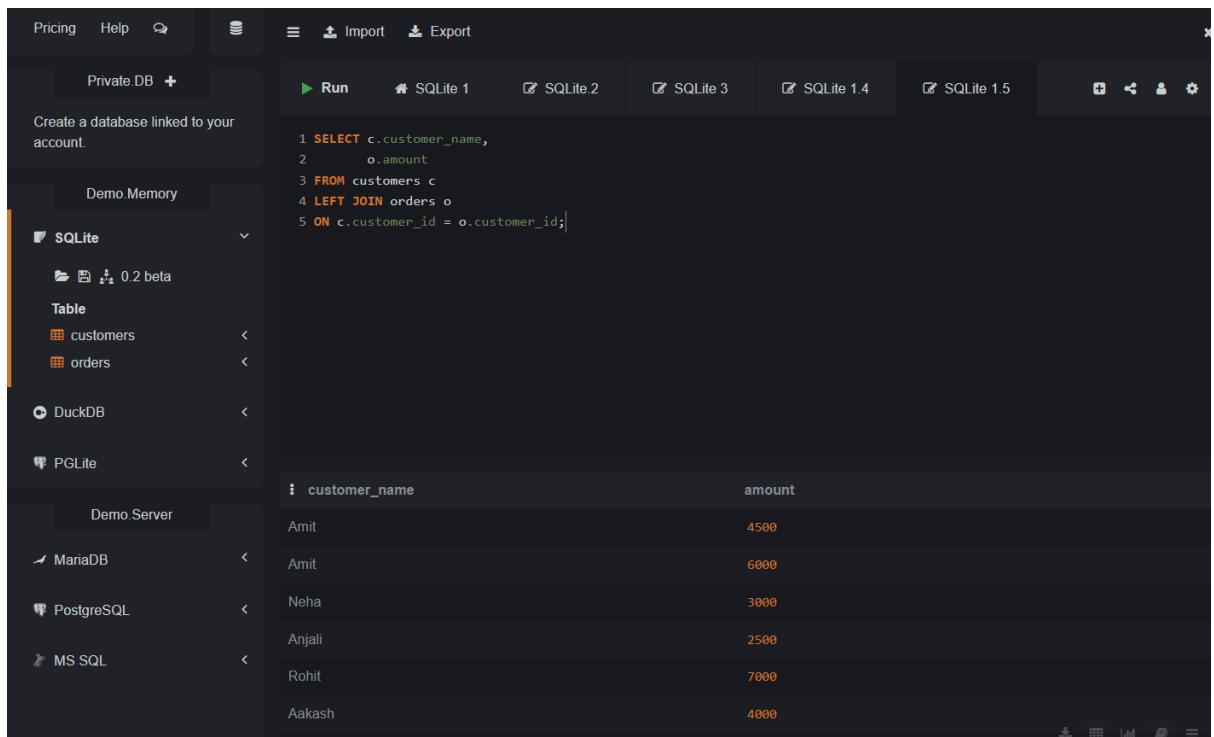
The screenshot shows a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private DB +** button.
- Create a database linked to your account.** input field.
- Demo.Memory** is selected in the dropdown menu.
- SQLite** is selected in the dropdown menu.
- Table** section shows **customers** and **orders**.
- DuckDB** is selected in the dropdown menu.
- PGLite** is selected in the dropdown menu.
- Demo.Server** is selected in the dropdown menu.
- MariaDB** is selected in the dropdown menu.
- PostgreSQL** is selected in the dropdown menu.
- MS SQL** is selected in the dropdown menu.
- Run** button is highlighted.
- SQLite 1** tab is selected.
- SQLite 2**, **SQLite 3**, **SQLite 1.4**, **SQLite 1.5** tabs are available.
- Customer** table results:

customer_id	customer_name
4	Rohit

**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;
```



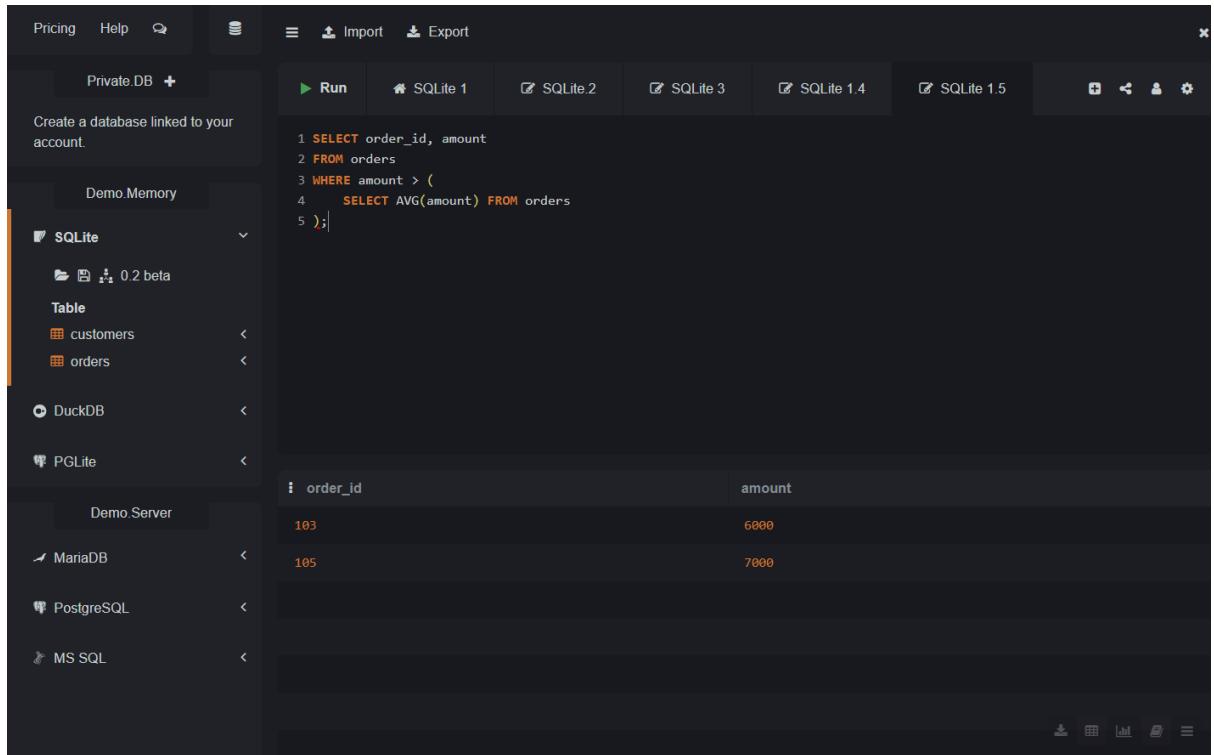
The screenshot shows a SQLite database interface with the following details:

- Pricing**, **Help**, **Import**, **Export** buttons at the top.
- Private.DB +** button.
- Run** button.
- SQLite 1** selected tab.
- SQLite 2**, **SQLite 3**, **SQLite 1.4**, **SQLite 1.5** tabs.
- Demo.Memory** database selected.
- SQLite** connection selected.
- Table** section shows **customers** and **orders** tables.
- DuckDB**, **PGLite**, **Demo Server**, **MariaDB**, **PostgreSQL**, **MS SQL** connections listed.
- customer\_name** and **amount** columns in the result table.
- Results:**

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 SQLite database interface with the following details:

- Toolbar:** Pricing, Help, Import, Export, Run, SQLite 1, SQLite 2, SQLite 3, SQLite 1.4, SQLite 1.5.
- Left Sidebar:** Shows connected databases: Private.DB, Demo.Memory, SQLite (with tables customers and orders), DuckDB, PGLite, Demo.Server, MariaDB, PostgreSQL, MS SQL.
- Query Editor:** Displays the SQL query:

```
1 SELECT order_id, amount
2 FROM orders
3 WHERE amount > (
4     SELECT AVG(amount) FROM orders
5 );
```
- Results Table:** A table showing the results of the query:

order_id	amount
103	6000
105	7000