Coding challenge

SQL

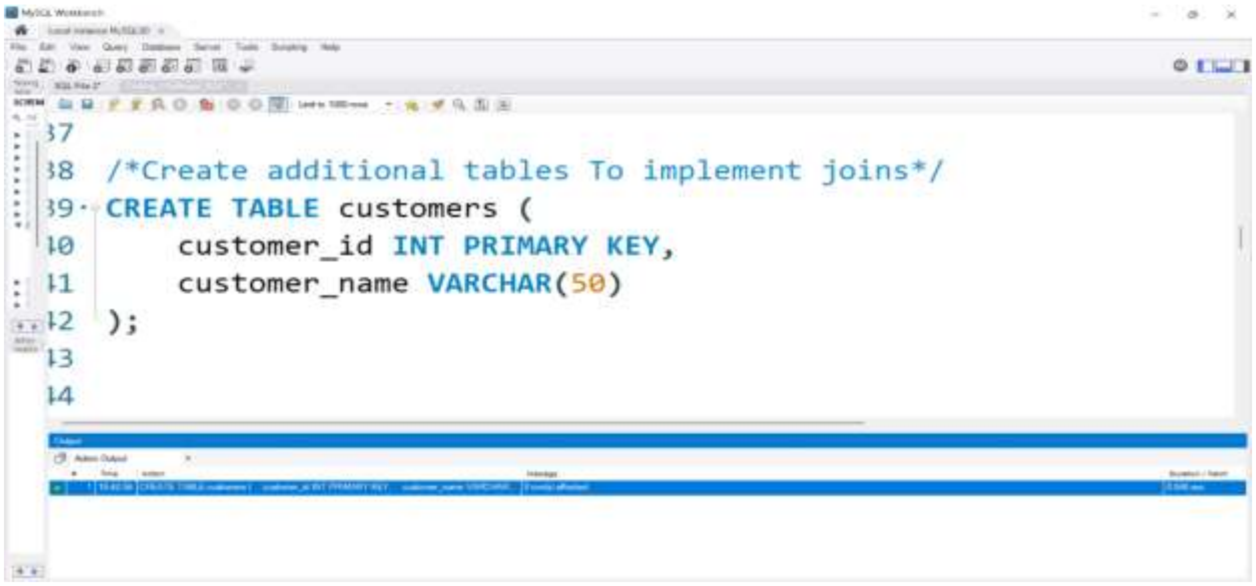
Name: Rohan Vinayak Chaudhari

Batch: Data Engineering 1

<u>Question2:</u> Execute all the join with examples.

1)Creating Extra table to implement join:

Created Extra table to connect with the order table and perform joins operation.

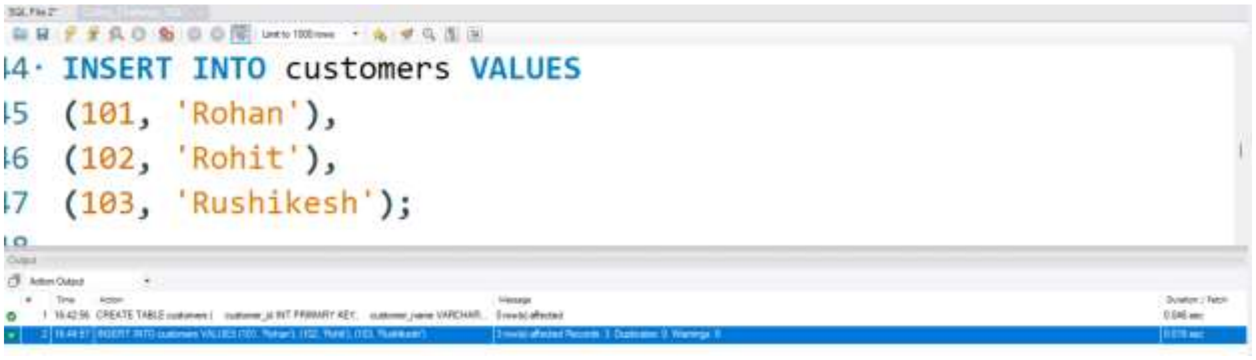


```sql
/*Create additional tables To implement joins*/
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(50)
);
```

```
1  16:42:56  CREATE TABLE customers (  customer_id INT PRIMARY KEY,  customer_name VARCHAR...  0 row(s) affected  0.046 sec
```

2)Inserting data in Customers table:

Inserted data in the customers table

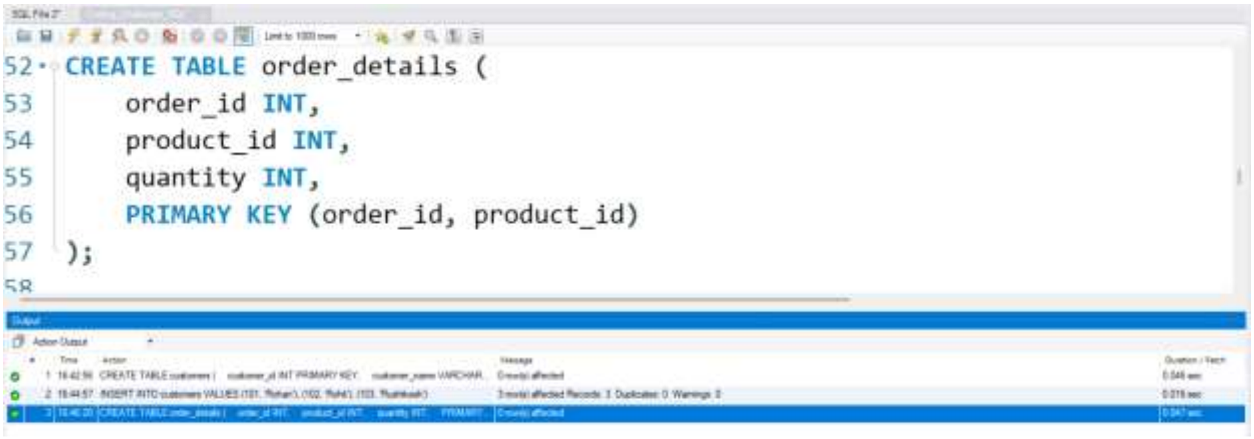


```sql
INSERT INTO customers VALUES
(101, 'Rohan'),
(102, 'Rohit'),
(103, 'Rushikesh');
```

```
#  Time      Action                                                                   Message                                                  Duration / Fetch
1  16:42:56  CREATE TABLE customers (  customer_id INT PRIMARY KEY,  customer_name VARCHAR...  0 row(s) affected                                        0.046 sec
2  16:44:57  INSERT INTO customers VALUES (101, 'Rohan'), (102, 'Rohit'), (103, 'Rushikesh')    3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.016 sec
```

3)Creating 3rd table

Created Extra table to connect with the order & customer table table and perform joins operation.

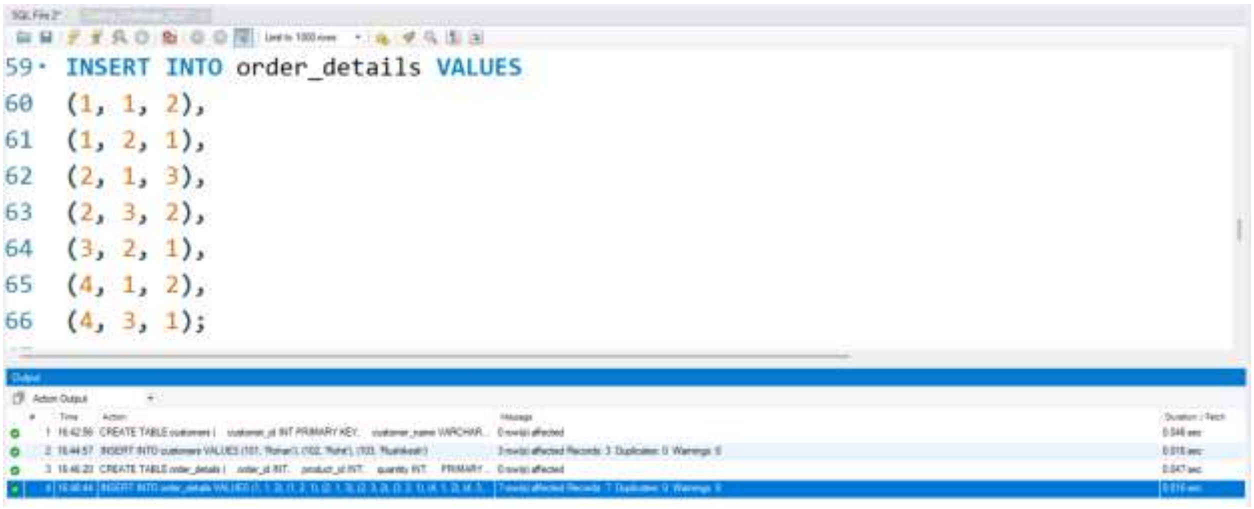


```
CREATE TABLE order_details (
    order_id INT,
    product_id INT,
    quantity INT,
    PRIMARY KEY (order_id, product_id)
);
```

```
Output
Action Output
#  Time      Action                                                                       Message                                                  Duration / Fetch
1  16:42:56  CREATE TABLE customers (   customer_id INT PRIMARY KEY,   customer_name VARCHAR...  0 row(s) affected                                        0.046 sec
2  16:44:57  INSERT INTO customers VALUES (101, 'Rohan'), (102, 'Rohit'), (103, 'Rushikesh')      3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.016 sec
3  16:46:20  CREATE TABLE order_details (   order_id INT,   product_id INT,   quantity INT,   PRIMARY...  0 row(s) affected                              0.047 sec
```

4)Inserting data in order_details:

Inserted data in the order details table.

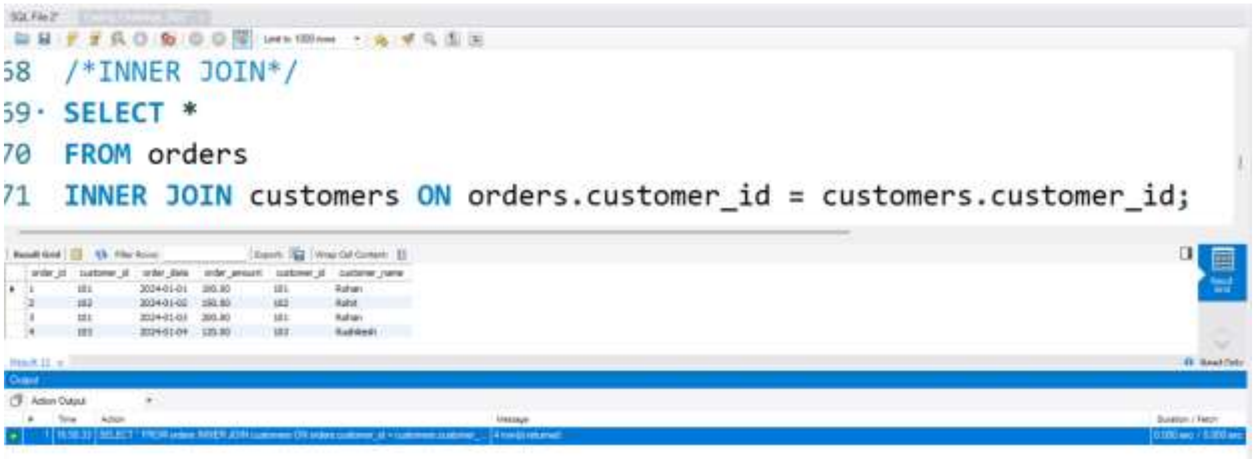


```
INSERT INTO order_details VALUES
(1, 1, 2),
(1, 2, 1),
(2, 1, 3),
(2, 3, 2),
(3, 2, 1),
(4, 1, 2),
(4, 3, 1);
```

```
Output
Action Output
#  Time      Action                                                                       Message                                                  Duration / Fetch
1  16:42:56  CREATE TABLE customers (   customer_id INT PRIMARY KEY,   customer_name VARCHAR...  0 row(s) affected                                        0.046 sec
2  16:44:57  INSERT INTO customers VALUES (101, 'Rohan'), (102, 'Rohit'), (103, 'Rushikesh')      3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.016 sec
3  16:46:20  CREATE TABLE order_details (   order_id INT,   product_id INT,   quantity INT,   PRIMARY...  0 row(s) affected                              0.047 sec
4  16:48:44  INSERT INTO order_details VALUES (1, 1, 2), (1, 2, 1), (2, 1, 3), (2, 3, 2), (3, 2, 1), (4, 1, 2), (4, 3,...  7 row(s) affected Records: 7  Duplicates: 0  Warnings: 0  0.016 sec
```

5)Performing inner join:

Returns only the rows where there is a match in both tables based on the specified condition (orders.customer_id = customers.customer_id).



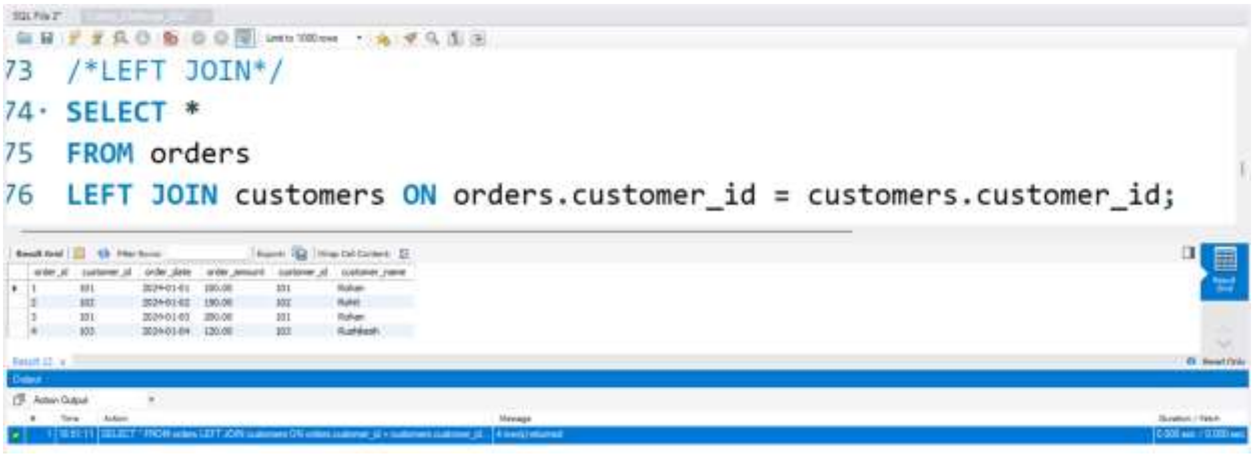
```sql
/*INNER JOIN*/
SELECT *
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;
```

order_id	customer_id	order_date	order_amount	customer_id	customer_name
1	101	2024-01-01	100.00	101	Rohan
2	102	2024-01-02	150.00	102	Rohit
3	101	2024-01-03	200.00	101	Rohan
4	103	2024-01-04	120.00	103	Rushikesh

```
1  18:50:33  SELECT * FROM orders INNER JOIN customers ON orders.customer_id = customers.customer_...   4 row(s) returned   0.000 sec / 0.000 sec
```

6)Performing LEFT JOIN

Returns all rows from the left table (orders) and the matched rows from the right table (customers). If there is no match, NULL values are returned for columns from the right table.



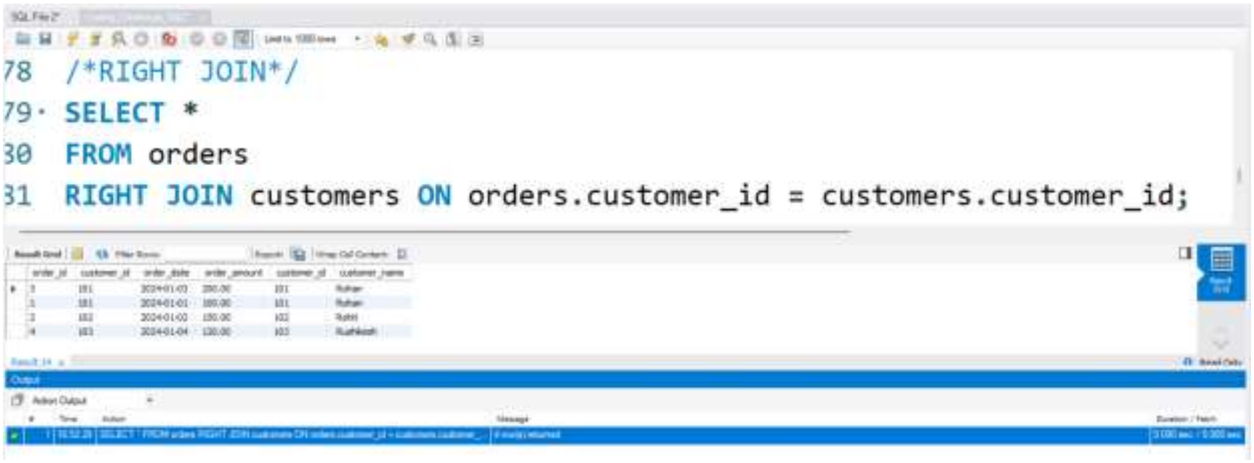
```sql
/*LEFT JOIN*/
SELECT *
FROM orders
LEFT JOIN customers ON orders.customer_id = customers.customer_id;
```

order_id	customer_id	order_date	order_amount	customer_id	customer_name
1	101	2024-01-01	100.00	101	Rohan
2	102	2024-01-02	150.00	102	Rohit
3	101	2024-01-03	200.00	101	Rohan
4	103	2024-01-04	120.00	103	Rushikesh

```
1  18:51:11  SELECT * FROM orders LEFT JOIN customers ON orders.customer_id = customers.customer_id...   4 row(s) returned   0.000 sec / 0.000 sec
```

7)Performing Right join:

Returns all rows from the right table (customers) and the matched rows from the left table (orders). If there is no match, NULL values are returned for columns from the left table.



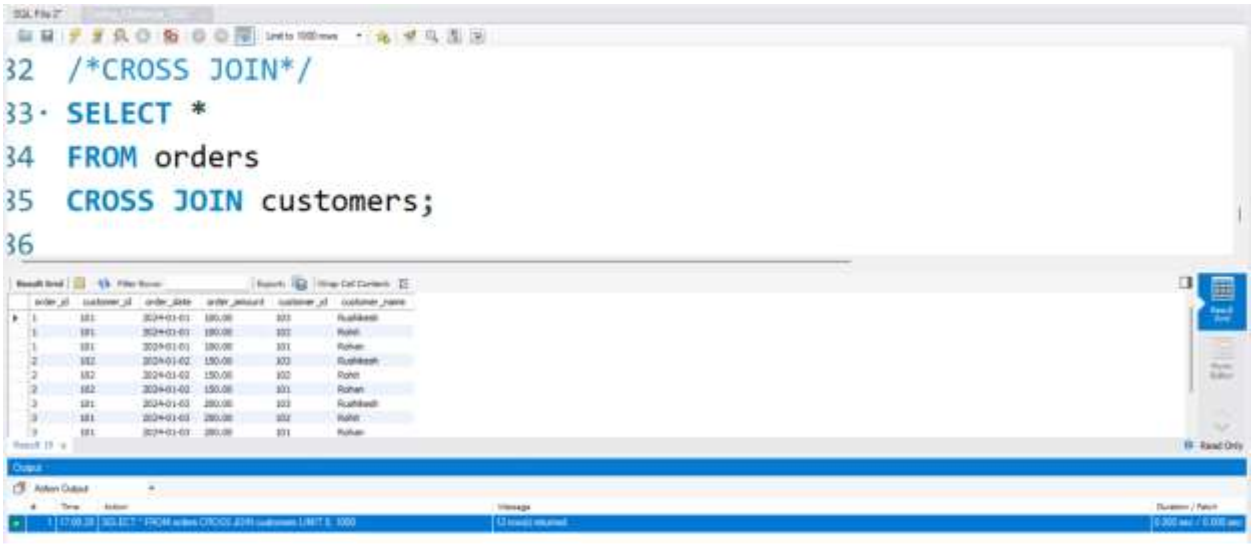
```
/*RIGHT JOIN*/
SELECT *
FROM orders
RIGHT JOIN customers ON orders.customer_id = customers.customer_id;
```

order_id	customer_id	order_date	order_amount	customer_id	customer_name
3	101	2024-01-03	200.00	101	Rohan
1	101	2024-01-01	100.00	101	Rohan
2	102	2024-01-02	150.00	102	Rohit
4	103	2024-01-04	120.00	103	Rushikesh

```
1  16:52:28  SELECT * FROM orders RIGHT JOIN customers ON orders.customer_id = customers.customer_...  4 row(s) returned  0.000 sec / 0.000 sec
```

8)Performing Cross join:

Returns the Cartesian product of rows from both tables, resulting in all possible combinations of rows from orders and customers.



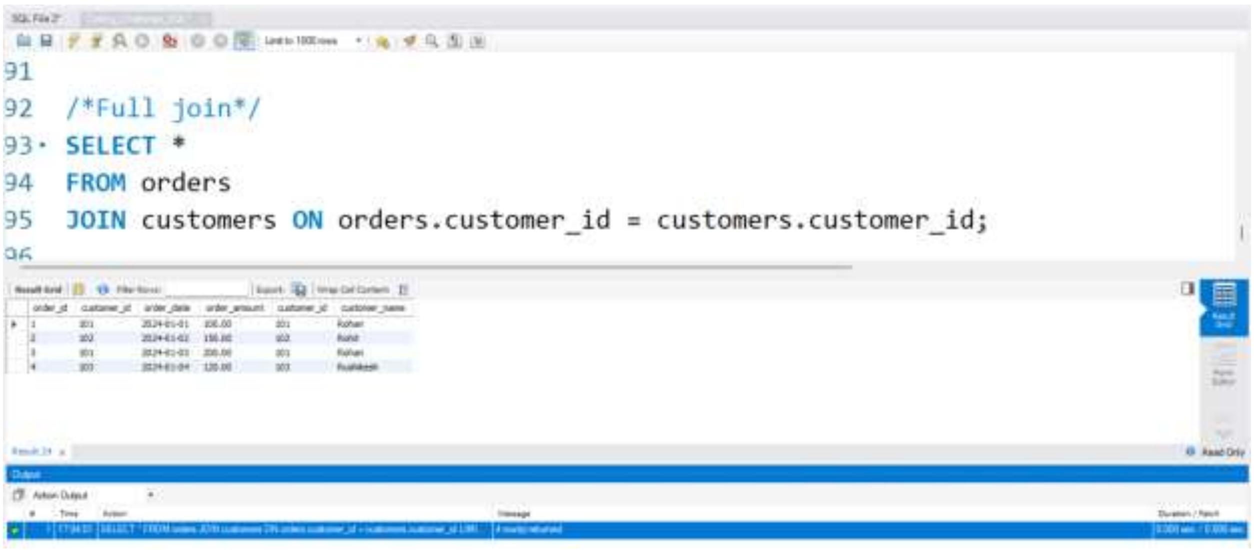
```
/*CROSS JOIN*/
SELECT *
FROM orders
CROSS JOIN customers;
```

order_id	customer_id	order_date	order_amount	customer_id	customer_name
1	101	2024-01-01	100.00	103	Rushikesh
1	101	2024-01-01	100.00	102	Rohit
1	101	2024-01-01	100.00	101	Rohan
2	102	2024-01-02	150.00	103	Rushikesh
2	102	2024-01-02	150.00	102	Rohit
2	102	2024-01-02	150.00	101	Rohan
3	101	2024-01-03	200.00	103	Rushikesh
3	101	2024-01-03	200.00	102	Rohit
3	101	2024-01-03	200.00	101	Rohan

```
1  17:00:28  SELECT * FROM orders CROSS JOIN customers LIMIT 0, 1000  12 row(s) returned  0.000 sec / 0.000 sec
```

9)Performing simple JOIN:

Gives all the result where data gets match with the given condition.



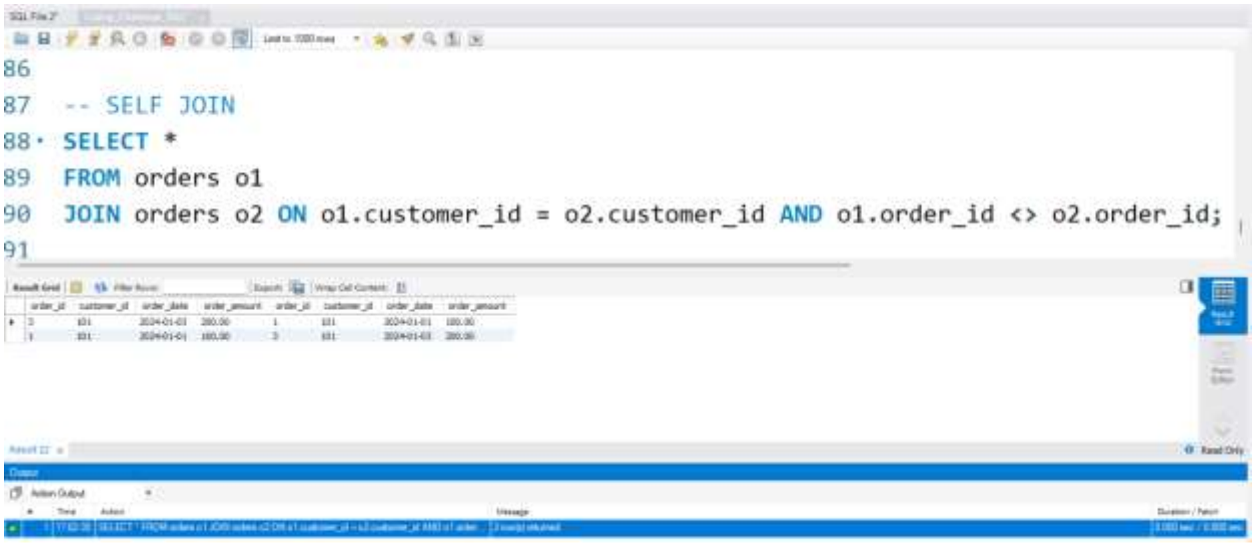
```
/*Full join*/
SELECT *
FROM orders
JOIN customers ON orders.customer_id = customers.customer_id;
```

order_id	customer_id	order_date	order_amount	customer_id	customer_name
1	101	2024-01-01	100.00	101	Rohan
2	102	2024-01-02	150.00	102	Rohit
3	101	2024-01-03	200.00	101	Rohan
4	103	2024-01-04	120.00	103	Rushikesh

```
1 | 17:34:31 | SELECT * FROM orders JOIN customers ON orders.customer_id = customers.customer_id LIMI... | 4 row(s) returned
```

10)Performing Self join:

Joins a table with itself. Here it retrieves rows from orders where the customer_id is the same, but the order_id is different, essentially finding pairs of orders for the same customer.



```
-- SELF JOIN
SELECT *
FROM orders o1
JOIN orders o2 ON o1.customer_id = o2.customer_id AND o1.order_id <> o2.order_id;
```

order_id	customer_id	order_date	order_amount	order_id	customer_id	order_date	order_amount
3	101	2024-01-03	200.00	1	101	2024-01-01	100.00
1	101	2024-01-01	100.00	3	101	2024-01-03	200.00

```
1 | 17:02:38 | SELECT * FROM orders o1 JOIN orders o2 ON o1.customer_id = o2.customer_id AND o1.order_id... | 2 row(s) returned
```