Day_10
Date: 07-08-2024
Assignments

Before the Start of Assignments, we have to Create a database with Some Tables inside it and Insert the values inside it So I have created the Database separately for that you can View the File named: "Asg_1-6_Database_Creation-Initial_Steps" which has Images of all steps I have performed still U'h Listing few steps here Also

Ans:

- ❖ **Creation Of The Database**

➔ **Commands:**

```
$ Sudo -i
$ Password:
# mysql

> SHOW DATABASES;

> CREATE DATABASE Innventory_Management;

> SHOW DATABASES;

> USE  Innventory_Management;

> - - CREAATING TABLES

> CREATE TABLE Customers ( CustomerID INT, Name VARCHAR(100),
Address VARCHAR(255), City VARCHAR(100), Pincode VARCHAR(10), Country
VARCHAR(100) );

> CREATE TABLE Employees ( EmployeeID INT, fName VARCHAR(50), lName
VARCHAR(50), Email VARCHAR(100), City VARCHAR(100), DepartmentID INT,
Salary DECIMAL(10, 2) );

> CREATE TABLE Departments ( DepartmentID INT, Department_Name
VARCHAR(100) );
```

> CREATE TABLE Orders (OrderID INT, CustomerID INT, EmployeeID INT, OrderDate DATE, ShippingID INT, OrderValue Double());

> CREATE TABLE Shipments (ShipmentID INT, OrderID INT, ShippingDate DATE, ShippingStatus VARCHAR(50));

> - - INSERTING VALUES INTO TABLES

> INSERT INTO Customers (CustomerID, Name, Address, City, Pincode, Country) VALUES (1, 'John Doe', '123 Elm St', 'Springfield', '62701', 'USA'), (2, 'Jane Smith', '456 Oak St', 'Lincoln', '68508', 'USA');

> INSERT INTO Employees (EmployeeID, fName, lName, Email, City, DepartmentID, Salary) VALUES (1, 'Michael', 'Smith', 'michael.smith@example.com', 'Chicago', 1, 60000.00), (2, 'Linda', 'Jones', 'linda.jones@example.com', 'Naperville', 2, 65000.00);

> INSERT INTO Departments (DepartmentID, Department_Name) VALUES (1, 'Sales'), (2, 'Marketing');

> INSERT INTO Orders (OrderID, CustomerID, EmployeeID, OrderDate, ShippingID OrderValue) VALUES (1, 1, 1, '2024-01-15', 1), (2, 2, 2, '2024-02-20', 2, 2800.00);

> INSERT INTO Shipments (ShipmentID, OrderID, ShippingDate, ShippingStatus) VALUES (1, 1, '2024-01-16', 'Delivered'), (2, 2, '2024-02-21', 'Shipped');

Till Here I have Done With the Initial Steps for Database Creation For the Assignments

From Below I have been started and Noted the solving assignments:

Assignment 1: Write a SELECT query to retrieve all columns from a 'customers' table, and modify it to return only the customer name and email address for customers in a specific city.

Ans:

- ❖ Step 1: Retrieve All Columns:
  - ➢ Code:

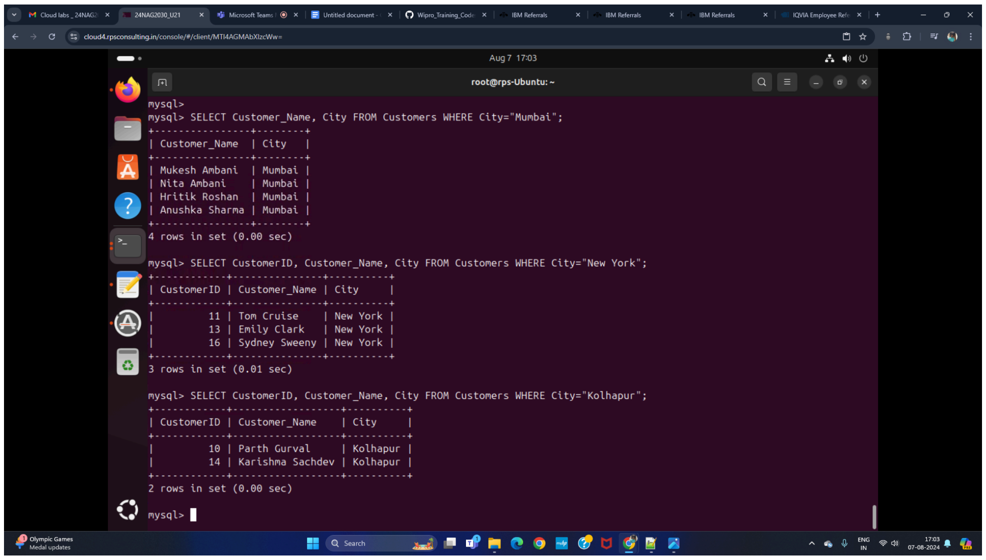
```
SELECT * FROM Customers;
```

- ❖ This will return all columns and rows from the Customers table.

- ❖ Step 2: Retrieve Specific Columns for a Specific City:
  - ➢ Code:

```
SELECT Customer_Name, City
FROM Customers
WHERE City = 'Mumbai';
```

- ❖ Step 3: Result is:



```
mysql>
mysql> SELECT Customer_Name, City FROM Customers WHERE City="Mumbai";
+---------------+--------+
| Customer_Name | City   |
+---------------+--------+
| Mukesh Ambani | Mumbai |
| Nita Ambani   | Mumbai |
| Hritik Roshan | Mumbai |
| Anushka Sharma | Mumbai |
+---------------+--------+
4 rows in set (0.00 sec)

mysql> SELECT CustomerID, Customer_Name, City FROM Customers WHERE City="New York";
+------------+---------------+----------+
| CustomerID | Customer_Name | City     |
+------------+---------------+----------+
|         11 | Tom Cruise    | New York |
|         13 | Emily Clark   | New York |
|         16 | Sydney Sweeny | New York |
+------------+---------------+----------+
3 rows in set (0.01 sec)

mysql> SELECT CustomerID, Customer_Name, City FROM Customers WHERE City="Kolhapur";
+------------+------------------+----------+
| CustomerID | Customer_Name    | City     |
+------------+------------------+----------+
|         10 | Parth Gurval     | Kolhapur |
|         14 | Karishma Sachdev | Kolhapur |
+------------+------------------+----------+
2 rows in set (0.00 sec)

mysql>
```

Assignment 2: Craft a query using an INNER JOIN to combine 'orders' and 'customers' tables for customers in a specified region, and a LEFT JOIN to display all customers including those without orders.

Ans:

- To complete this assignment, I have done two Steps:
  - I have used an INNER JOIN to combine the orders and customers tables to show details for customers in a specified region.
  - I have used a LEFT JOIN to display all customers, including those without orders.

- **Step 1: INNER JOIN to Combine orders and customers**
  - **Code:**

```
SELECT o.OrderID, c.CustomerID, c.Customer_Name, c.City,
o.OrderDate

FROM Orders o

INNER JOIN Customers c ON o.CustomerID = c.CustomerID

WHERE c.City = 'Mumbai';
```

  - **Description:**

    SELECT o.OrderID, c.CustomerID, c.Customer_Name, c.City, o.OrderDate: Specifies the columns to retrieve from both tables.

    FROM Orders o: Indicates that the base table is Orders and aliased as o.

    INNER JOIN Customers c ON o.CustomerID = c.CustomerID: Performs an inner join between Orders and Customers based on the CustomerID field.

    WHERE c.City = 'Mumbai': Filters the results to include only those customers located in Mumbai.

- **Step 2: LEFT JOIN to Display All Customers Including Those Without Orders**
  - **Code:**

SELECT c.CustomerID, c.Customer_Name, c.City, o.OrderID, o.OrderDate

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;

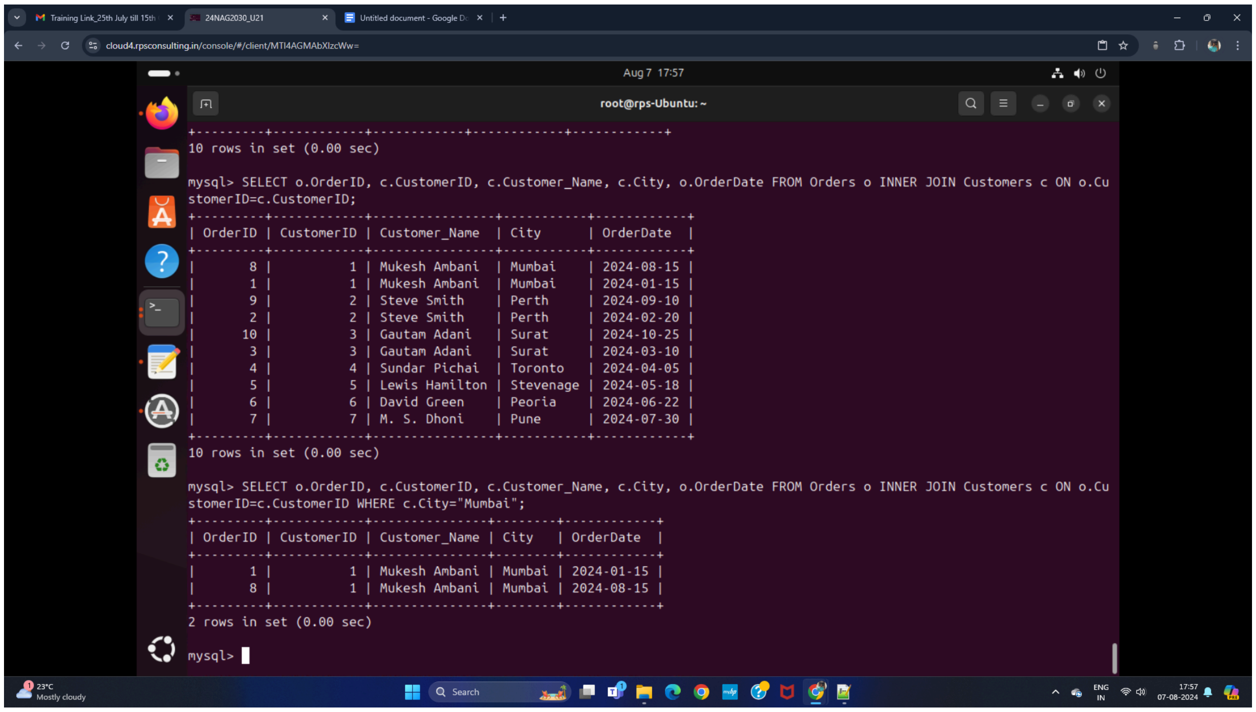
➢ **Description:**

SELECT c.CustomerID, c.Customer_Name, c.City, o.OrderID, o.OrderDate: Specifies the columns to retrieve from both tables. If a customer does not have any orders, OrderID and OrderDate will be NULL.

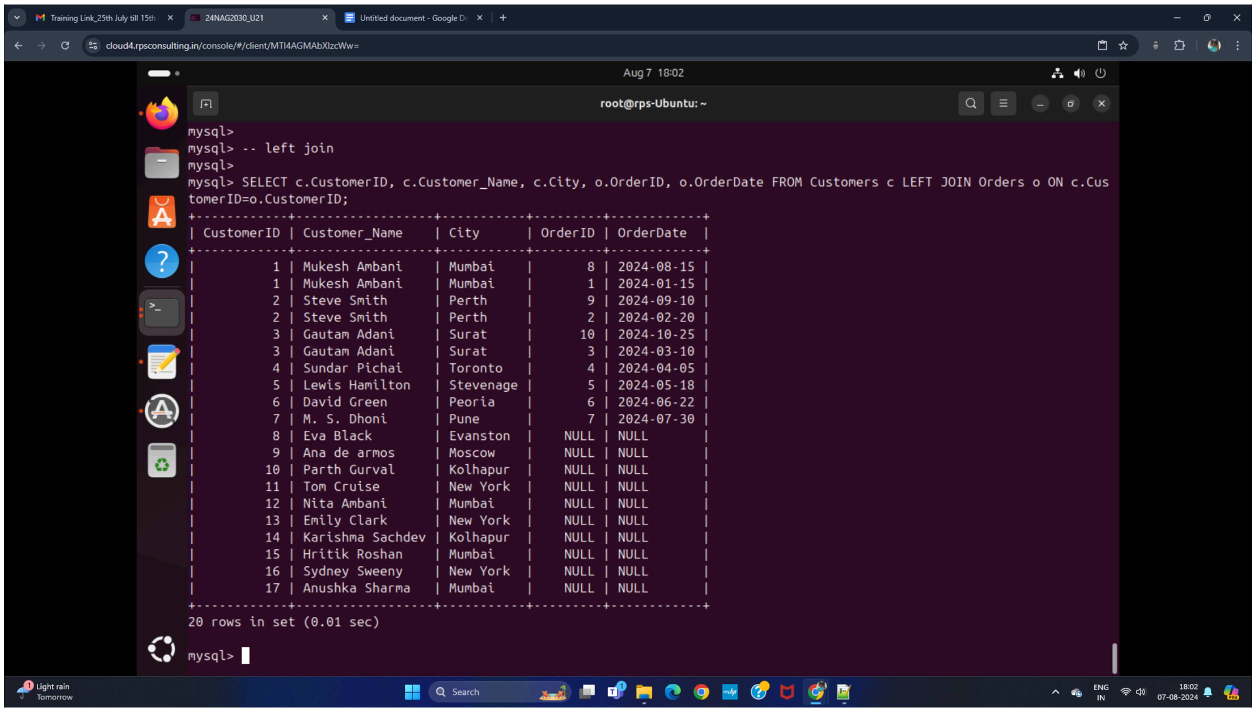
FROM Customers c: Indicates that the base table is Customers and aliased as c.

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID: Performs a left join between Customers and Orders based on the CustomerID field. This ensures all customers are included, even those without corresponding orders.

❖ Step 3: Final Output:


Training Link_25th July till 15th
24NAG2030_U21
Untitled document - Google Do
cloud4.rpsconsulting.in/console/#/client/MTI4AGMAbXlzcWw=
Aug 7 17:57
root@rps-Ubuntu: ~
23°C
Mostly cloudy
Search
ENG
IN
17:57
07-08-2024

```
+---------+------------+-----------+-----------+-----------+
10 rows in set (0.00 sec)

mysql> SELECT o.OrderID, c.CustomerID, c.Customer_Name, c.City, o.OrderDate FROM Orders o INNER JOIN Customers c ON o.Cu
stomerID=c.CustomerID;
+---------+------------+----------------+-----------+------------+
| OrderID | CustomerID | Customer_Name  | City      | OrderDate  |
+---------+------------+----------------+-----------+------------+
|       8 |          1 | Mukesh Ambani  | Mumbai    | 2024-08-15 |
|       1 |          1 | Mukesh Ambani  | Mumbai    | 2024-01-15 |
|       9 |          2 | Steve Smith    | Perth     | 2024-09-10 |
|       2 |          2 | Steve Smith    | Perth     | 2024-02-20 |
|      10 |          3 | Gautam Adani   | Surat     | 2024-10-25 |
|       3 |          3 | Gautam Adani   | Surat     | 2024-03-10 |
|       4 |          4 | Sundar Pichai  | Toronto   | 2024-04-05 |
|       5 |          5 | Lewis Hamilton | Stevenage | 2024-05-18 |
|       6 |          6 | David Green    | Peoria    | 2024-06-22 |
|       7 |          7 | M. S. Dhoni    | Pune      | 2024-07-30 |
+---------+------------+----------------+-----------+------------+
10 rows in set (0.00 sec)

mysql> SELECT o.OrderID, c.CustomerID, c.Customer_Name, c.City, o.OrderDate FROM Orders o INNER JOIN Customers c ON o.Cu
stomerID=c.CustomerID WHERE c.City="Mumbai";
+---------+------------+---------------+--------+------------+
| OrderID | CustomerID | Customer_Name | City   | OrderDate  |
+---------+------------+---------------+--------+------------+
|       1 |          1 | Mukesh Ambani | Mumbai | 2024-01-15 |
|       8 |          1 | Mukesh Ambani | Mumbai | 2024-08-15 |
+---------+------------+---------------+--------+------------+
2 rows in set (0.00 sec)

mysql> 
```



```
mysql>
mysql> -- left join
mysql>
mysql> SELECT c.CustomerID, c.Customer_Name, c.City, o.OrderID, o.OrderDate FROM Customers c LEFT JOIN Orders o ON c.Cus
tomerID=o.CustomerID;
+------------+-----------------+-----------+---------+------------+
| CustomerID | Customer_Name   | City      | OrderID | OrderDate  |
+------------+-----------------+-----------+---------+------------+
|          1 | Mukesh Ambani   | Mumbai    |       8 | 2024-08-15 |
|          1 | Mukesh Ambani   | Mumbai    |       1 | 2024-01-15 |
|          2 | Steve Smith     | Perth     |       9 | 2024-09-10 |
|          2 | Steve Smith     | Perth     |       2 | 2024-02-20 |
|          3 | Gautam Adani    | Surat     |      10 | 2024-10-25 |
|          3 | Gautam Adani    | Surat     |       3 | 2024-03-10 |
|          4 | Sundar Pichai   | Toronto   |       4 | 2024-04-05 |
|          5 | Lewis Hamilton  | Stevenage |       5 | 2024-05-18 |
|          6 | David Green     | Peoria    |       6 | 2024-06-22 |
|          7 | M. S. Dhoni     | Pune      |       7 | 2024-07-30 |
|          8 | Eva Black       | Evanston  |    NULL | NULL       |
|          9 | Ana de armos    | Moscow    |    NULL | NULL       |
|         10 | Parth Gurval    | Kolhapur  |    NULL | NULL       |
|         11 | Tom Cruise      | New York  |    NULL | NULL       |
|         12 | Nita Ambani     | Mumbai    |    NULL | NULL       |
|         13 | Emily Clark     | New York  |    NULL | NULL       |
|         14 | Karishma Sachdev | Kolhapur |    NULL | NULL       |
|         15 | Hritik Roshan   | Mumbai    |    NULL | NULL       |
|         16 | Sydney Sweeny   | New York  |    NULL | NULL       |
|         17 | Anushka Sharma  | Mumbai    |    NULL | NULL       |
+------------+-----------------+-----------+---------+------------+
20 rows in set (0.01 sec)

mysql>
```

Assignment 3: Utilize a subquery to find customers who have placed orders above the average order value, and write a UNION query to combine two SELECT statements with the same number of columns.

Ans:

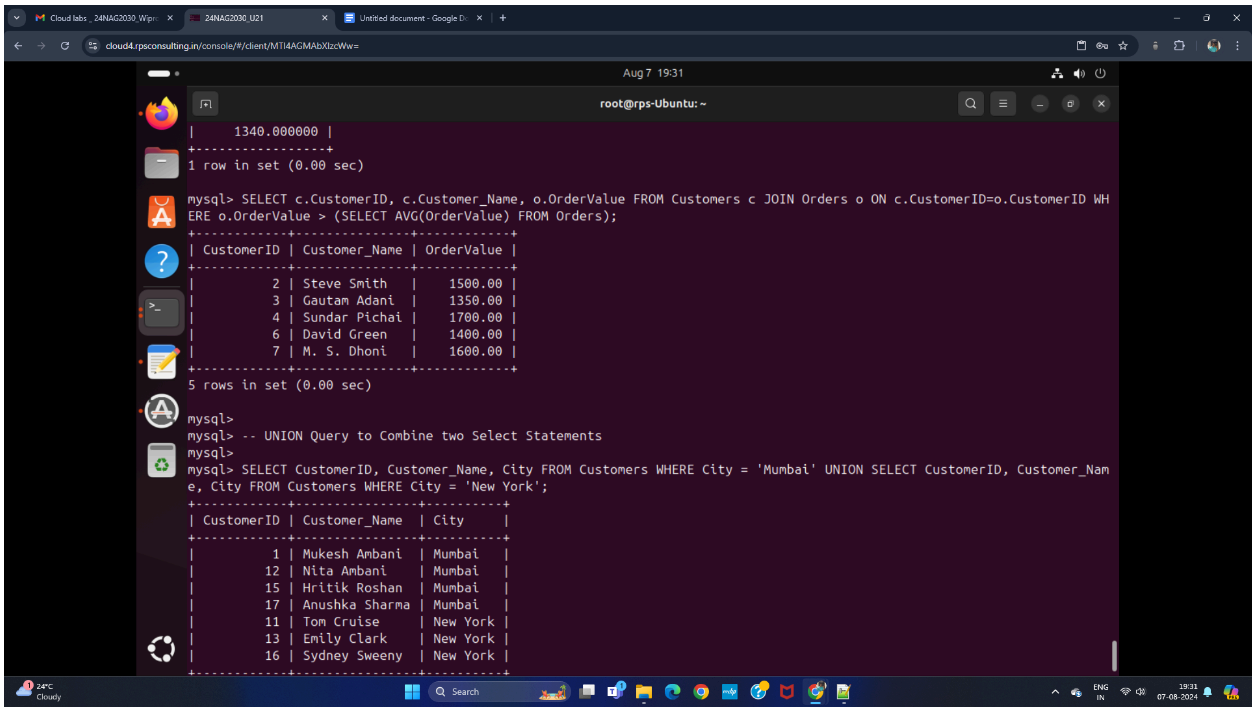
- Here I have to create the subquery to find customers who have placed orders above the average order value.
- First, we have to calculate the average order value. With the column OrderValue in the Orders table.
- Second, have to write a UNION query to combine two SELECT statements with the same number of columns.
- **Step 1: Subquery to find customers who have placed orders above the average order value:**
  - **Code:**

SELECT CustomerID, Customer_Name

FROM Customers

```
WHERE CustomerID IN (
  SELECT CustomerID
   FROM Orders
   WHERE OrderValue > (SELECT AVG(OrderValue) FROM Orders)
);
```

➢ **Result:**



```
|     1340.000000 |
+-----------------+
1 row in set (0.00 sec)

mysql> SELECT c.CustomerID, c.Customer_Name, o.OrderValue FROM Customers c JOIN Orders o ON c.CustomerID=o.CustomerID WH
ERE o.OrderValue > (SELECT AVG(OrderValue) FROM Orders);
+------------+---------------+------------+
| CustomerID | Customer_Name | OrderValue |
+------------+---------------+------------+
|          2 | Steve Smith   |    1500.00 |
|          3 | Gautam Adani  |    1350.00 |
|          4 | Sundar Pichai |    1700.00 |
|          6 | David Green   |    1400.00 |
|          7 | M. S. Dhoni   |    1600.00 |
+------------+---------------+------------+
5 rows in set (0.00 sec)

mysql>
mysql> -- UNION Query to Combine two Select Statements
mysql>
mysql> SELECT CustomerID, Customer_Name, City FROM Customers WHERE City = 'Mumbai' UNION SELECT CustomerID, Customer_Nam
e, City FROM Customers WHERE City = 'New York';
+------------+----------------+----------+
| CustomerID | Customer_Name  | City     |
+------------+----------------+----------+
|          1 | Mukesh Ambani  | Mumbai   |
|         12 | Nita Ambani    | Mumbai   |
|         15 | Hritik Roshan  | Mumbai   |
|         17 | Anushka Sharma | Mumbai   |
|         11 | Tom Cruise     | New York |
|         13 | Emily Clark    | New York |
|         16 | Sydney Sweeny  | New York |
+------------+----------------+----------+
```

❖ Step 2: UNION Query to Combine Two SELECT Statements:

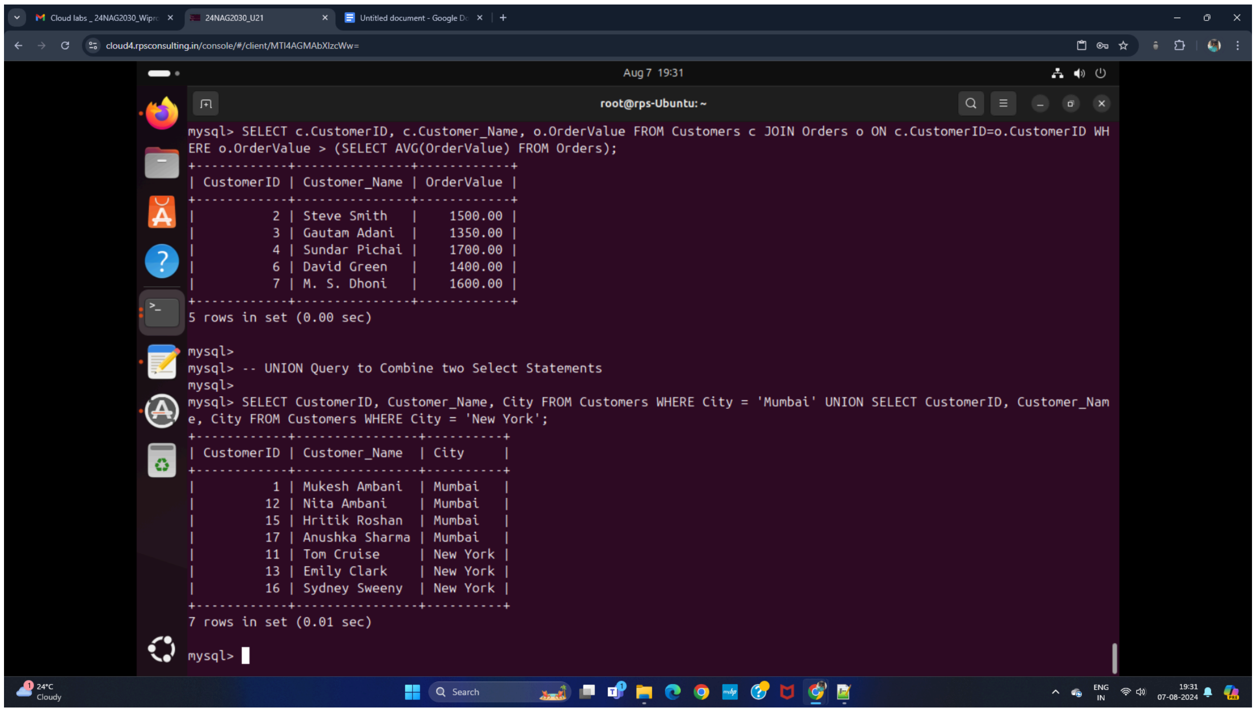
➢ **Code:**

```
SELECT CustomerID, Customer_Name, City

FROM Customers

WHERE City = 'Mumbai'

UNION

SELECT CustomerID, Customer_Name, City
```

FROM Customers

WHERE City = 'New York';

➢ **Result:**



```
mysql> SELECT c.CustomerID, c.Customer_Name, o.OrderValue FROM Customers c JOIN Orders o ON c.CustomerID=o.CustomerID WH
ERE o.OrderValue > (SELECT AVG(OrderValue) FROM Orders);
+------------+---------------+------------+
| CustomerID | Customer_Name | OrderValue |
+------------+---------------+------------+
|          2 | Steve Smith   |    1500.00 |
|          3 | Gautam Adani  |    1350.00 |
|          4 | Sundar Pichai |    1700.00 |
|          6 | David Green   |    1400.00 |
|          7 | M. S. Dhoni   |    1600.00 |
+------------+---------------+------------+
5 rows in set (0.00 sec)

mysql>
mysql> -- UNION Query to Combine two Select Statements
mysql>
mysql> SELECT CustomerID, Customer_Name, City FROM Customers WHERE City = 'Mumbai' UNION SELECT CustomerID, Customer_Nam
e, City FROM Customers WHERE City = 'New York';
+------------+----------------+----------+
| CustomerID | Customer_Name  | City     |
+------------+----------------+----------+
|          1 | Mukesh Ambani  | Mumbai   |
|         12 | Nita Ambani    | Mumbai   |
|         15 | Hritik Roshan  | Mumbai   |
|         17 | Anushka Sharma | Mumbai   |
|         11 | Tom Cruise     | New York |
|         13 | Emily Clark    | New York |
|         16 | Sydney Sweeny  | New York |
+------------+----------------+----------+
7 rows in set (0.01 sec)

mysql>
```

Description:

1. **Subquery to Find Average Order Value (`AverageOrderValue`)**: This Common Table Expression (CTE) calculates the average order value.
2. **First SELECT**: Retrieves customers who have placed orders with a value above the average.
3. **Second SELECT**: Retrieves customers located in 'Mumbai'.
4. **`UNION`**: Combines the results from the two `SELECT` statements, eliminating duplicate rows.

Assignment 4: Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'orders' table, COMMIT the transaction, then UPDATE the 'products' table, and ROLLBACK the transaction.

Ans:

- ❖ To perform the assignment of actions involving transactions in SQL, I have follow these steps:
  1. **BEGIN a transaction**: Start a new transaction.
  2. **INSERT a new record into the orders table**: Add a new record.
  3. **COMMIT the transaction**: Save the changes made during the transaction.
  4. **UPDATE the products table**: Modify existing records.
  5. **ROLLBACK the transaction**: Revert any changes made after the commit.
- ❖ Code:

```
-- First start with a new transaction

START TRANSACTION;

-- Then Insert a new record into the 'Orders' table

INSERT INTO Orders (CustomerID, EmployeeID, OrderDate, OrderValue,
ShippingID)

VALUES (11, 3, '2024-11-01', 1450.00, 13);

-- Then check the contents of the 'Orders' table to see the new record

SELECT * FROM Orders WHERE CustomerID = 11;

-- Commit the transaction to save changes

COMMIT;

-- Again check the contents of the 'Orders' table after the commit

SELECT * FROM Orders WHERE CustomerID = 11;

-- Again start a new transaction for updating the 'Orders' table

START TRANSACTION;
```

```
-- Update the 'Orders' table

UPDATE Orders

SET OrderValue = 1550.00

WHERE CustomerID = 11;

-- Check the contents of the 'Orders' table to see the update

SELECT * FROM Orders WHERE CustomerID = 11;

-- Rollback the transaction to revert changes

ROLLBACK;

-- Check the contents of the 'Orders' table after the rollback

SELECT * FROM Orders WHERE CustomerID = 11;
```

❖ Working:

1. START TRANSACTION;: Begins a new transaction.
2. INSERT INTO Orders: Adds a new order record.
3. SELECT * FROM Orders WHERE CustomerID = 11;: Views the newly inserted record.
4. COMMIT;: Saves the inserted record.
5. SELECT * FROM Orders WHERE CustomerID = 11;: Verifies that the record is saved.
6. START TRANSACTION;: Begins a new transaction for the update.
7. UPDATE Orders: Updates the OrderValue for the inserted record.
8. SELECT * FROM Orders WHERE CustomerID = 11;: Views the updated record.
9. ROLLBACK;: Reverts the update.
10. SELECT * FROM Orders WHERE CustomerID = 11;: Verifies that the update was rolled back.

Assignment 5: Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.

Ans:

- To complete Assignment 5, I have done the steps as follows:
    1. Begin a transaction.
    2. Insert multiple records into the Orders table, setting a SAVEPOINT after each insert.
    3. Rollback to the second SAVEPOINT to undo the changes made after that point.
    4. Commit the overall transaction to finalize the changes up to the second SAVEPOINT.
- **Code:**

```
-- First start a new transaction

START TRANSACTION;



-- After Insert the first record into the 'Orders' table and set a SAVEPOINT

INSERT INTO Orders (CustomerID, EmployeeID, OrderDate, OrderValue, ShippingID)

VALUES (12, 2, '2024-11-05', 1600.00, 14);

SAVEPOINT sp1;



-- Then Insert the second record into the 'Orders' table and set another SAVEPOINT

INSERT INTO Orders (CustomerID, EmployeeID, OrderDate, OrderValue, ShippingID)

VALUES (13, 3, '2024-11-06', 1700.00, 15);

SAVEPOINT sp2;
```

```
-- Again Insert the third record into the 'Orders' table

INSERT INTO Orders (CustomerID, EmployeeID, OrderDate, OrderValue, ShippingID)

VALUES (14, 1, '2024-11-07', 1800.00, 16);


-- Rollback to the second SAVEPOINT

ROLLBACK TO SAVEPOINT sp2;


-- Commit the transaction to save the changes up to the second SAVEPOINT

COMMIT;


-- Check the contents of the 'Orders' table to confirm the result

SELECT * FROM Orders WHERE CustomerID IN (12, 13, 14);
```

❖ Explanation:

1. START TRANSACTION;: Initiates a new transaction.
2. INSERT INTO Orders ... VALUES ...;: Inserts a new record into the Orders table.
3. SAVEPOINT sp1;: Sets a savepoint named sp1 after the first insert. This allows you to roll back to this point if needed.
4. INSERT INTO Orders ... VALUES ...;: Inserts another record and sets another savepoint named sp2.
5. SAVEPOINT sp2;: Sets a savepoint named sp2.
6. INSERT INTO Orders ... VALUES ...;: Inserts a third record. This record and any subsequent changes can be rolled back if necessary.
7. ROLLBACK TO SAVEPOINT sp2;: Rolls back changes made after the sp2 savepoint. The first two records are kept, while the third record is discarded.
8. COMMIT;: Commits the transaction, finalizing the changes up to the sp2 savepoint.
9. SELECT * FROM Orders WHERE CustomerID IN (12, 13, 14);: Verifies the contents of the Orders table to confirm the result of the transaction.

Assignment 6: Draft a brief report on the use of transaction logs for data recovery and create a hypothetical scenario where a transaction log is instrumental in data recovery after an unexpected shutdown.

Ans:

- ❖ Report on Transaction Logs for Data Recovery:

Transaction Logs: The Lifesavers of Database Recovery

<u>**What are Transaction Logs?**</u>

As a database management system (DBMS) user, have you ever wondered how databases recover from unexpected events like crashes or failures? The answer lies in transaction logs, which record every change made to the database. These logs are essential for ensuring data consistency and availability.

How Do Transaction Logs Work?

Transaction logs perform four critical functions:

1. Recording Changes: They capture detailed information about every operation that modifies the database, including inserts, updates, and deletes.
2. Ensuring Atomicity: Transaction logs ensure that transactions are atomic, meaning either all operations are completed successfully, or none are applied.
3. Durability and Consistency: They support the durability and consistency properties of transactions, guaranteeing that changes are recovered in case of system failures.
4. Recovery Mechanism: In the event of a system crash, transaction logs can be used to recover the database to its last consistent state.

A Real-Life Scenario: Recovering from a System Failure

Imagine you're a database administrator (DBA) for an online retail company. One evening, the database server experiences a sudden power outage, causing an unexpected shutdown. Several transactions were in progress, including updates to inventory levels and customer orders. The server doesn't restart properly, and the database appears to be in an inconsistent state.

Using Transaction Logs for Recovery

To recover the database, I would follow these steps:

1. Find the Last Consistent State: I'd examine the transaction logs to identify the last point at which the database was in a consistent state.
2. Roll Back Incomplete Transactions: I'd use the transaction logs to roll back incomplete transactions that were in progress at the time of the shutdown.
3. Replay Committed Transactions: I'd replay any transactions that were committed before the shutdown to ensure all committed changes are applied.
4. Restore the Database: After rolling back incomplete transactions and applying committed ones, I'd restore the database to a consistent state.
5. Verify Data Integrity: I'd verify the integrity and consistency of the database by checking key tables and performing consistency checks.
6. Restart Operations: Once the recovery is complete, I'd restart the database server and resume normal operations.

Conclusion

- In this scenario, transaction logs played a crucial role in recovering the database from an unexpected shutdown.
- By recording every change made to the database, the logs enabled me to restore the database to its last consistent state, ensuring that all committed transactions were preserved while incomplete ones were rolled back.
- This process highlights the importance of transaction logs in maintaining data integrity and availability in the face of unforeseen disruptions.