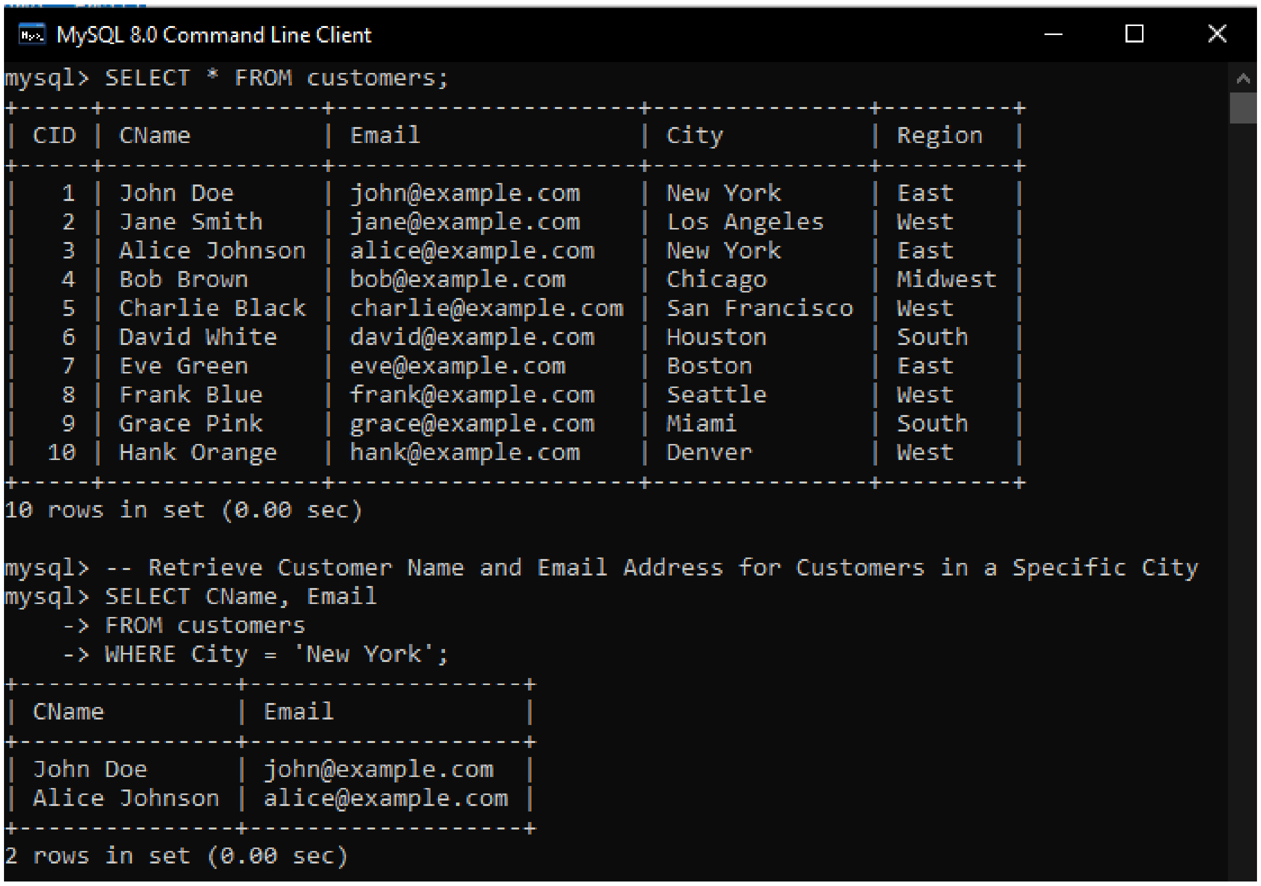
Day 10 Assignment

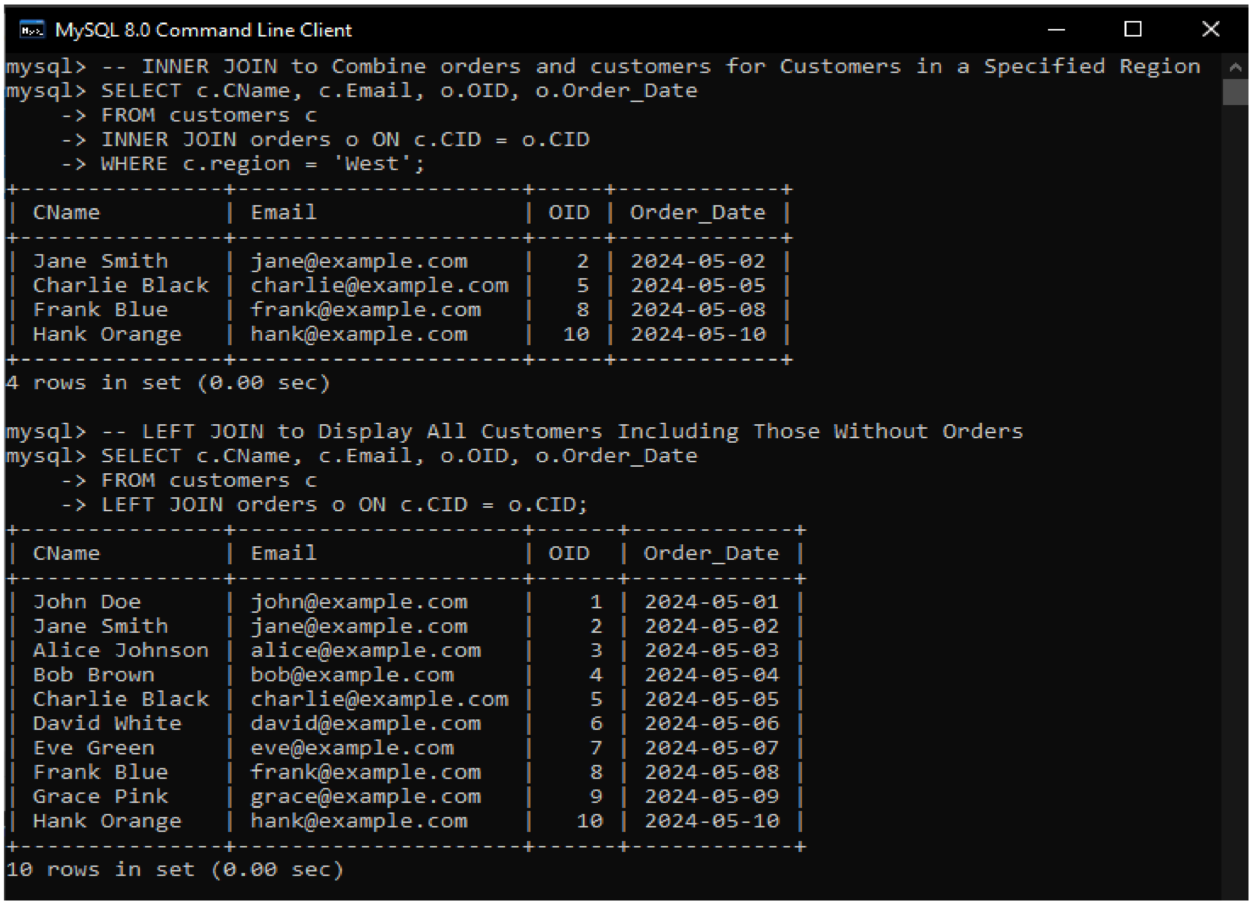
Name : Mehul Anjikhane Email:mehulanjikhane13@gmail.com

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



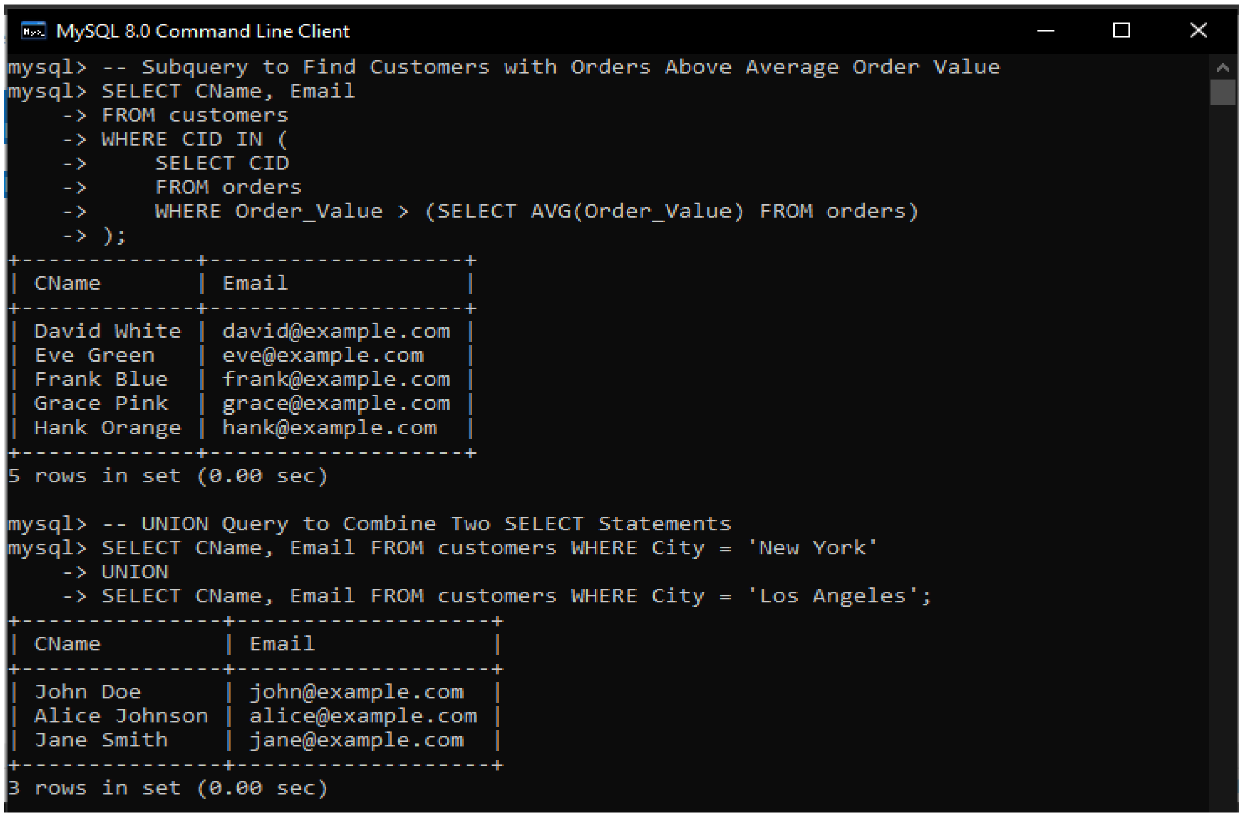
```
mysql> SELECT * FROM customers;
+-----+---------------+---------------------+---------------+---------+
| CID | CName         | Email               | City          | Region  |
+-----+---------------+---------------------+---------------+---------+
|   1 | John Doe      | john@example.com    | New York      | East    |
|   2 | Jane Smith    | jane@example.com    | Los Angeles   | West    |
|   3 | Alice Johnson | alice@example.com   | New York      | East    |
|   4 | Bob Brown     | bob@example.com     | Chicago       | Midwest |
|   5 | Charlie Black | charlie@example.com | San Francisco | West    |
|   6 | David White   | david@example.com   | Houston       | South   |
|   7 | Eve Green     | eve@example.com     | Boston        | East    |
|   8 | Frank Blue    | frank@example.com   | Seattle       | West    |
|   9 | Grace Pink    | grace@example.com   | Miami         | South   |
|  10 | Hank Orange   | hank@example.com    | Denver        | West    |
+-----+---------------+---------------------+---------------+---------+
10 rows in set (0.00 sec)

mysql> -- Retrieve Customer Name and Email Address for Customers in a Specific City
mysql> SELECT CName, Email
    -> FROM customers
    -> WHERE City = 'New York';
+---------------+-------------------+
| CName         | Email             |
+---------------+-------------------+
| John Doe      | john@example.com  |
| Alice Johnson | alice@example.com |
+---------------+-------------------+
2 rows in set (0.00 sec)
```

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 these without orders.


MySQL 8.0 Command Line Client

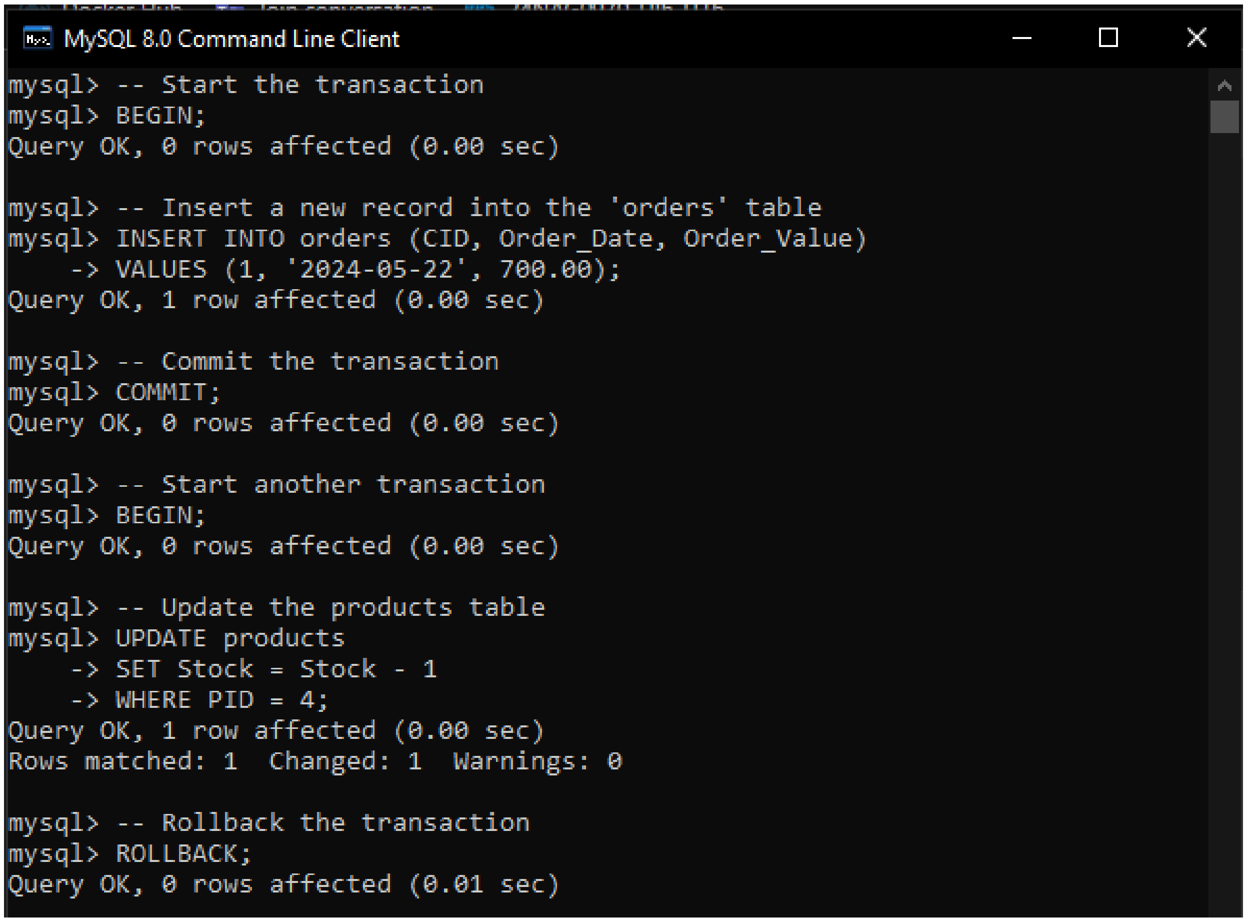
```
mysql> -- INNER JOIN to Combine orders and customers for Customers in a Specified Region
mysql> SELECT c.CName, c.Email, o.OID, o.Order_Date
    -> FROM customers c
    -> INNER JOIN orders o ON c.CID = o.CID
    -> WHERE c.region = 'West';
+---------------+---------------------+-----+------------+
| CName         | Email               | OID | Order_Date |
+---------------+---------------------+-----+------------+
| Jane Smith    | jane@example.com    |   2 | 2024-05-02 |
| Charlie Black | charlie@example.com |   5 | 2024-05-05 |
| Frank Blue    | frank@example.com   |   8 | 2024-05-08 |
| Hank Orange   | hank@example.com    |  10 | 2024-05-10 |
+---------------+---------------------+-----+------------+
4 rows in set (0.00 sec)

mysql> -- LEFT JOIN to Display All Customers Including Those Without Orders
mysql> SELECT c.CName, c.Email, o.OID, o.Order_Date
    -> FROM customers c
    -> LEFT JOIN orders o ON c.CID = o.CID;
+---------------+---------------------+------+------------+
| CName         | Email               | OID  | Order_Date |
+---------------+---------------------+------+------------+
| John Doe      | john@example.com    |    1 | 2024-05-01 |
| Jane Smith    | jane@example.com    |    2 | 2024-05-02 |
| Alice Johnson | alice@example.com   |    3 | 2024-05-03 |
| Bob Brown     | bob@example.com     |    4 | 2024-05-04 |
| Charlie Black | charlie@example.com |    5 | 2024-05-05 |
| David White   | david@example.com   |    6 | 2024-05-06 |
| Eve Green     | eve@example.com     |    7 | 2024-05-07 |
| Frank Blue    | frank@example.com   |    8 | 2024-05-08 |
| Grace Pink    | grace@example.com   |    9 | 2024-05-09 |
| Hank Orange   | hank@example.com    |   10 | 2024-05-10 |
+---------------+---------------------+------+------------+
10 rows in set (0.00 sec)
```

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.


MySQL 8.0 Command Line Client

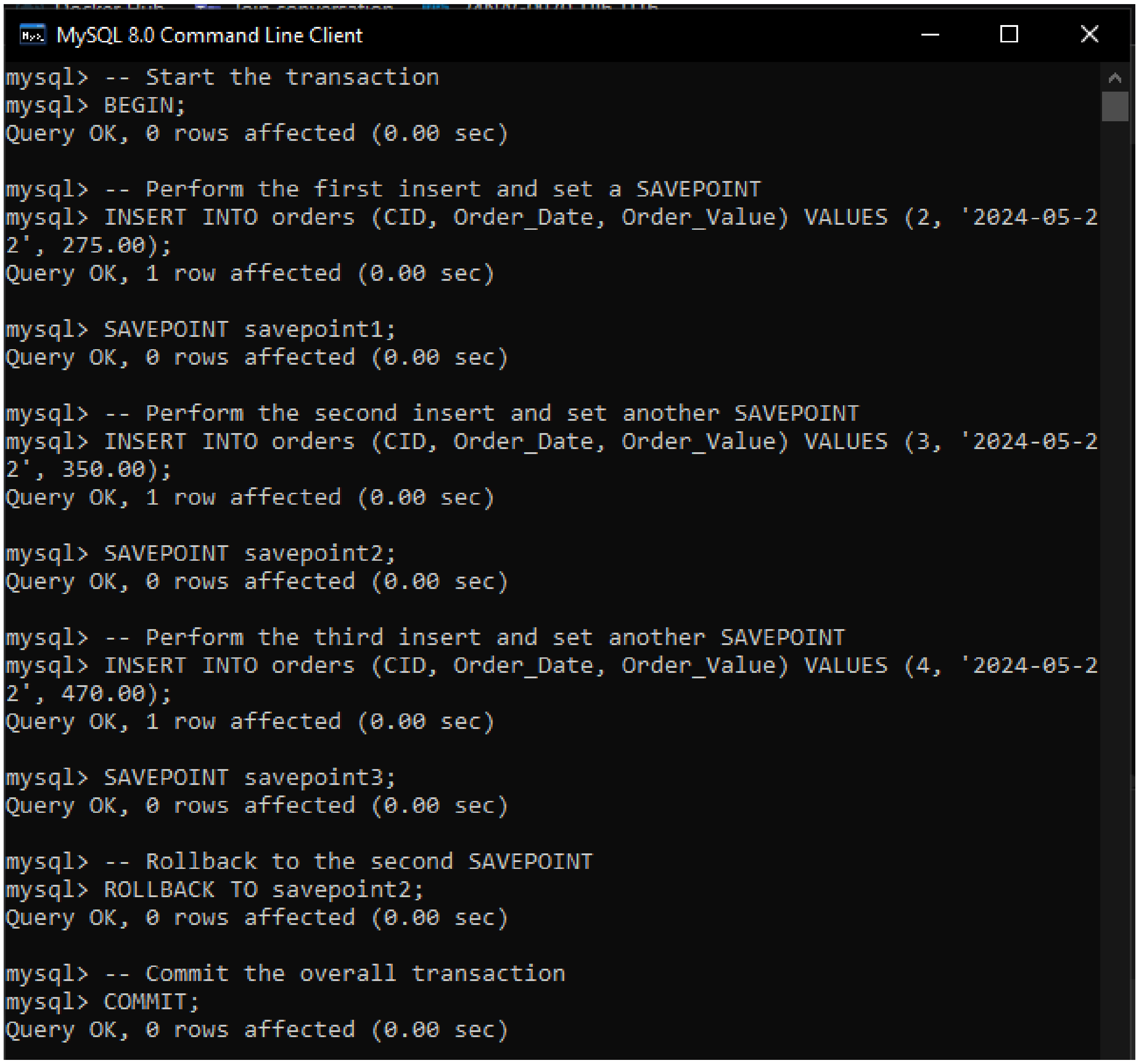
```
mysql> -- Subquery to Find Customers with Orders Above Average Order Value
mysql> SELECT CName, Email
    -> FROM customers
    -> WHERE CID IN (
    ->     SELECT CID
    ->     FROM orders
    ->     WHERE Order_Value > (SELECT AVG(Order_Value) FROM orders)
    -> );
+-------------+-------------------+
| CName       | Email             |
+-------------+-------------------+
| David White | david@example.com |
| Eve Green   | eve@example.com   |
| Frank Blue  | frank@example.com |
| Grace Pink  | grace@example.com |
| Hank Orange | hank@example.com  |
+-------------+-------------------+
5 rows in set (0.00 sec)

mysql> -- UNION Query to Combine Two SELECT Statements
mysql> SELECT CName, Email FROM customers WHERE City = 'New York'
    -> UNION
    -> SELECT CName, Email FROM customers WHERE City = 'Los Angeles';
+---------------+-------------------+
| CName         | Email             |
+---------------+-------------------+
| John Doe      | john@example.com  |
| Alice Johnson | alice@example.com |
| Jane Smith    | jane@example.com  |
+---------------+-------------------+
3 rows in set (0.00 sec)
```

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.


MySQL 8.0 Command Line Client

```
mysql> -- Start the transaction
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Insert a new record into the 'orders' table
mysql> INSERT INTO orders (CID, Order_Date, Order_Value)
    -> VALUES (1, '2024-05-22', 700.00);
Query OK, 1 row affected (0.00 sec)

mysql> -- Commit the transaction
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Start another transaction
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Update the products table
mysql> UPDATE products
    -> SET Stock = Stock - 1
    -> WHERE PID = 4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> -- Rollback the transaction
mysql> ROLLBACK;
Query OK, 0 rows affected (0.01 sec)
```

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.



MySQL 8.0 Command Line Client

```
mysql> -- Start the transaction
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Perform the first insert and set a SAVEPOINT
mysql> INSERT INTO orders (CID, Order_Date, Order_Value) VALUES (2, '2024-05-2
2', 275.00);
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT savepoint1;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Perform the second insert and set another SAVEPOINT
mysql> INSERT INTO orders (CID, Order_Date, Order_Value) VALUES (3, '2024-05-2
2', 350.00);
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT savepoint2;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Perform the third insert and set another SAVEPOINT
mysql> INSERT INTO orders (CID, Order_Date, Order_Value) VALUES (4, '2024-05-2
2', 470.00);
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT savepoint3;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Rollback to the second SAVEPOINT
mysql> ROLLBACK TO savepoint2;
Query OK, 0 rows affected (0.00 sec)

mysql> -- Commit the overall transaction
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

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.

Transaction Logs for Data Recovery

Transaction logs are critical for ensuring data integrity and enabling data recovery in case of system failures. These logs record all changes made to the database, including INSERT, UPDATE, DELETE operations, and transaction control commands like COMMIT and ROLLBACK. By keeping a detailed record of all transactions, transaction logs allow the database to:

- **Recover from Crashes:** In the event of an unexpected shutdown or crash, the database can use the transaction log to recover to the last known consistent state. Any uncommitted transactions can be rolled back, and committed transactions can be replayed to ensure no data loss.
- **Point-in-Time Recovery:** Transaction logs enable point-in-time recovery, allowing the database to be restored to a specific moment before an error or data corruption occurred.

Hypothetical Scenario

Imagine a retail company's database server crashes unexpectedly due to a power outage. The database was handling numerous transactions at the time, including new orders and updates to inventory levels. Upon restarting the server, the database uses the transaction log to:

- **Identify Uncommitted Transactions:** Any transactions that were not committed at the time of the crash are identified and rolled back, ensuring that partial or corrupt data is not retained.
- **Replay Committed Transactions:** Transactions that were committed but not yet written to the main database files are replayed from the log, ensuring that all customer orders and inventory updates are accurately reflected.

This process allows the company to resume operations with confidence that the database is in a consistent and accurate state, minimizing downtime and potential data loss.