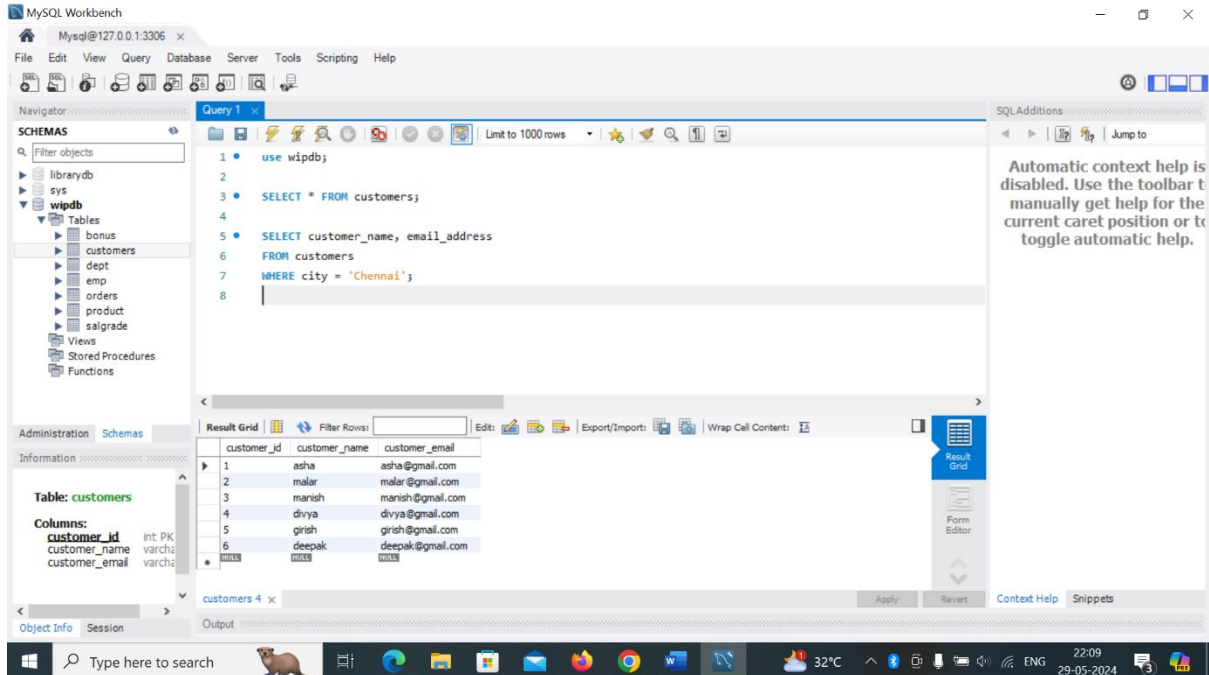


# SQL

**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**



The screenshot shows the MySQL Workbench interface. The 'Query Editor' window displays the following SQL query:

```
1 use wipdb;
2
3 SELECT * FROM customers;
4
5 SELECT customer_name, email_address
6 FROM customers
7 WHERE city = 'Chennai';
8
```

The 'Result Grid' window shows the results of the query, displaying columns: customer\_id, customer\_name, and customer\_email. The results are as follows:

customer_id	customer_name	customer_email
1	asha	asha@gmail.com
2	malar	malar@gmail.com
3	manish	manish@gmail.com
4	divya	divya@gmail.com
5	girish	girish@gmail.com
6	deepak	deepak@gmail.com

The 'Navigator' window on the left shows the database structure, including the 'wipdb' database and its tables: bonus, customers, dept, emp, orders, product, and salgrade. The 'Table: customers' is selected, and its columns are listed: customer\_id (int PK), customer\_name (varchar), and customer\_email (varchar).

**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.**

The screenshot displays the MySQL Workbench interface. The 'Query Editor' window contains the following SQL code:

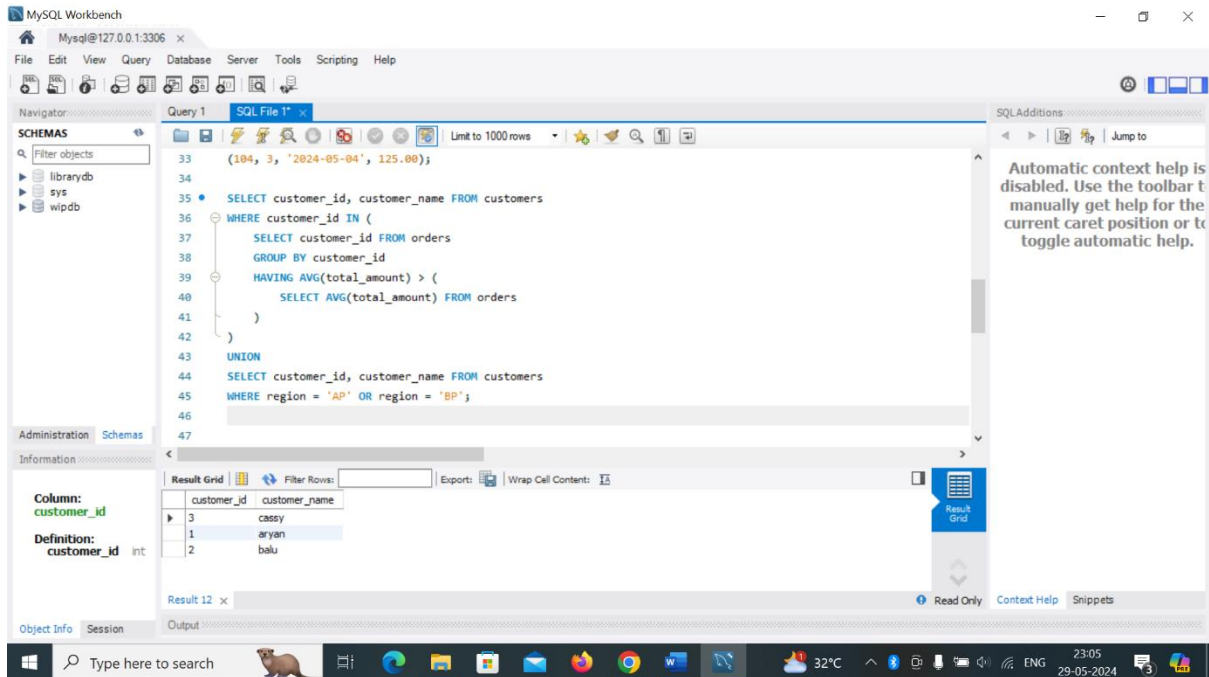
```
1 use wipdb;
2
3 SELECT * FROM customers;
4
5 SELECT customer_name, email_address
6 FROM customers
7 WHERE city = 'Chennai';
8
```

The 'Result Grid' shows the output of the second query, displaying a table with 6 rows and 3 columns: customer\_id, customer\_name, and customer\_email.

customer_id	customer_name	customer_email
1	asha	asha@gmail.com
2	malar	malar@gmail.com
3	manish	manish@gmail.com
4	divya	divya@gmail.com
5	girish	girish@gmail.com
6	deepak	deepak@gmail.com

The 'Navigator' pane on the left shows the database schema, including tables like bonus, customers, dept, emp, orders, product, and salgrade. The 'Information' pane at the bottom left provides details about the 'customers' table, including its columns and data types.

**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**



The screenshot shows the MySQL Workbench interface. The main window displays a SQL query in the 'Query 1' tab. The query is as follows:

```
33 (104, 3, '2024-05-04', 125.00);
34
35 SELECT customer_id, customer_name FROM customers
36 WHERE customer_id IN (
37     SELECT customer_id FROM orders
38     GROUP BY customer_id
39     HAVING AVG(total_amount) > (
40         SELECT AVG(total_amount) FROM orders
41     )
42 )
43 UNION
44 SELECT customer_id, customer_name FROM customers
45 WHERE region = 'AP' OR region = 'BP';
46
47
```

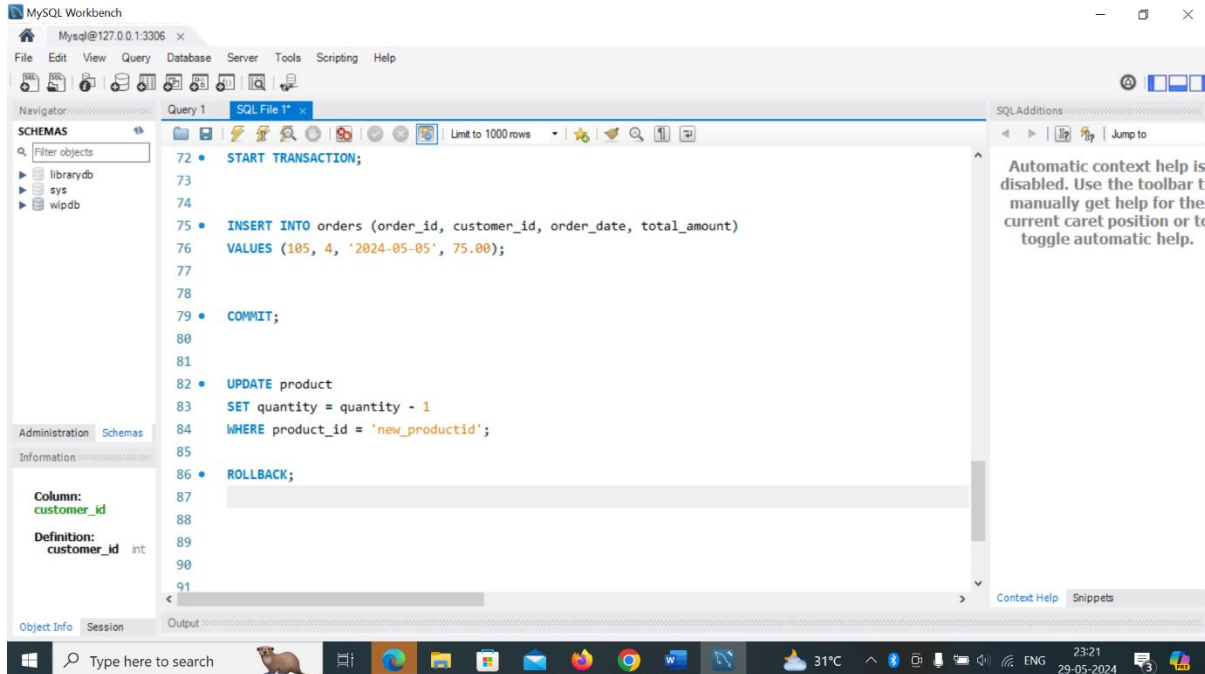
The 'Result Grid' at the bottom shows the results of the query:

customer_id	customer_name
3	cassy
1	aryani
2	belu

The 'Information' panel on the left shows the column definition for 'customer\_id' as an integer.

The 'SQLAdditions' panel on the right contains a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

**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.**



The screenshot shows the MySQL Workbench interface. The 'Query 1' window contains the following SQL code:

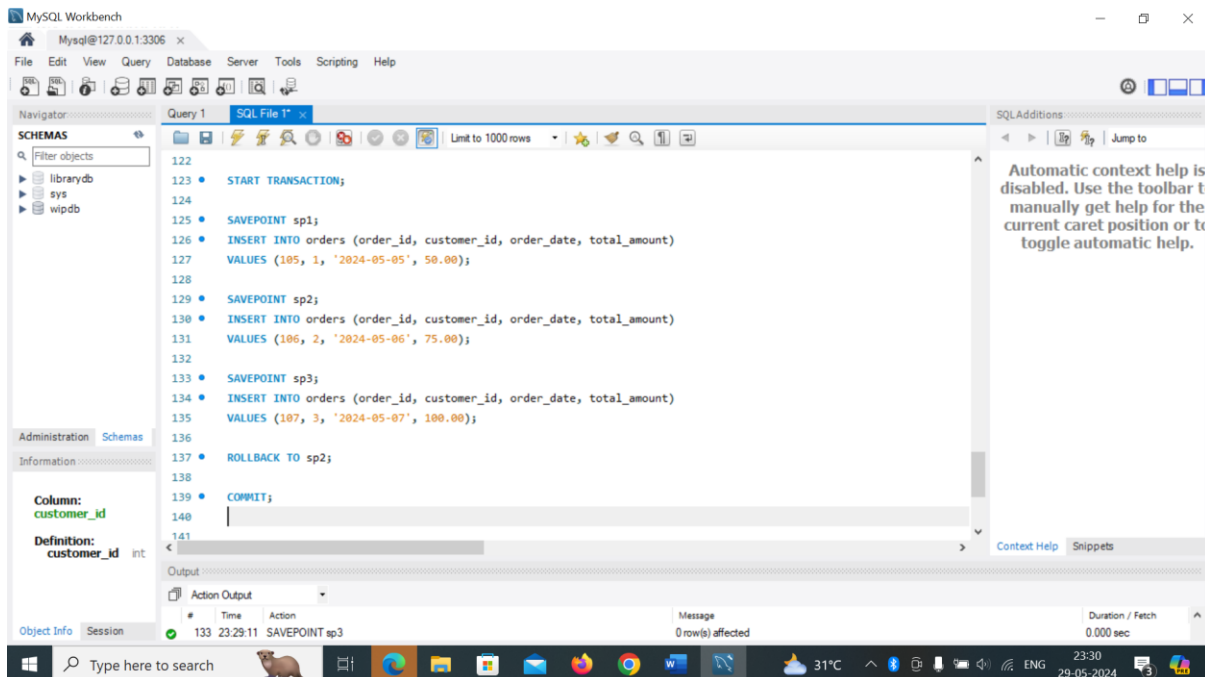
```

72 • START TRANSACTION;
73
74
75 • INSERT INTO orders (order_id, customer_id, order_date, total_amount)
76   VALUES (105, 4, '2024-05-05', 75.00);
77
78
79 • COMMIT;
80
81
82 • UPDATE product
83   SET quantity = quantity - 1
84   WHERE product_id = 'new_productid';
85
86 • ROLLBACK;
87
88
89
90
91

```

The 'Navigator' panel on the left shows the 'SCHEMAS' section with 'librarydb', 'sys', and 'wipdb' listed. The 'Object Info' panel shows the 'customer\_id' column definition as 'customer\_id int'. The 'Output' panel at the bottom is empty.

**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**



The screenshot shows the MySQL Workbench interface. The 'Query 1' window contains the following SQL code:

```

122
123 • START TRANSACTION;
124
125 • SAVEPOINT sp1;
126 • INSERT INTO orders (order_id, customer_id, order_date, total_amount)
127   VALUES (105, 1, '2024-05-05', 50.00);
128
129 • SAVEPOINT sp2;
130 • INSERT INTO orders (order_id, customer_id, order_date, total_amount)
131   VALUES (106, 2, '2024-05-06', 75.00);
132
133 • SAVEPOINT sp3;
134 • INSERT INTO orders (order_id, customer_id, order_date, total_amount)
135   VALUES (107, 3, '2024-05-07', 100.00);
136
137 • ROLLBACK TO sp2;
138
139 • COMMIT;
140
141

```

The 'Navigator' panel on the left shows the 'SCHEMAS' section with 'librarydb', 'sys', and 'wipdb' listed. The 'Object Info' panel shows the 'customer\_id' column definition as 'customer\_id int'. The 'Output' panel at the bottom shows a message: '133 23:29:11 SAVEPOINT sp3' and '0 row(s) affected'.

**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**

## **Report on the Use of Transaction Logs for Data Recovery**

### **Introduction:**

Transaction logs are vital components of modern database management systems (DBMS). They act as detailed records of every change made to a database, ensuring data consistency and integrity, especially in the face of unexpected system shutdowns or failures.

### **Functionality of Transaction Logs:**

Transaction logs serve a crucial function by recording all database modifications during transactions, including inserts, updates, and deletes. They store both the original and modified data, enabling the DBMS to roll back or replay transactions as needed for data recovery.

### **Use Cases for Data Recovery:**

**Incomplete Transaction Rollback:** If a system crash interrupts a transaction, the DBMS can use transaction logs to roll back the incomplete operation, restoring the database to a consistent state.

**Point-in-Time Recovery:** Transaction logs allow for recovery to a specific moment in time by replaying logged transactions up to the desired timestamp, ensuring data consistency.

**Error Correction:** In cases of erroneous data modifications, transaction logs facilitate the rollback of affected transactions, preserving data accuracy and integrity.

**Recovery from Disk Failures:** Transaction logs aid in reconstructing lost or damaged data due to disk failures by replaying logged transactions onto a new storage medium.

### **Hypothetical Scenario:**

A financial institution's database experiences an unexpected shutdown during critical transactions, such as payroll processing and customer payments. Without transaction logs, this interruption could lead to financial discrepancies and operational disruptions.