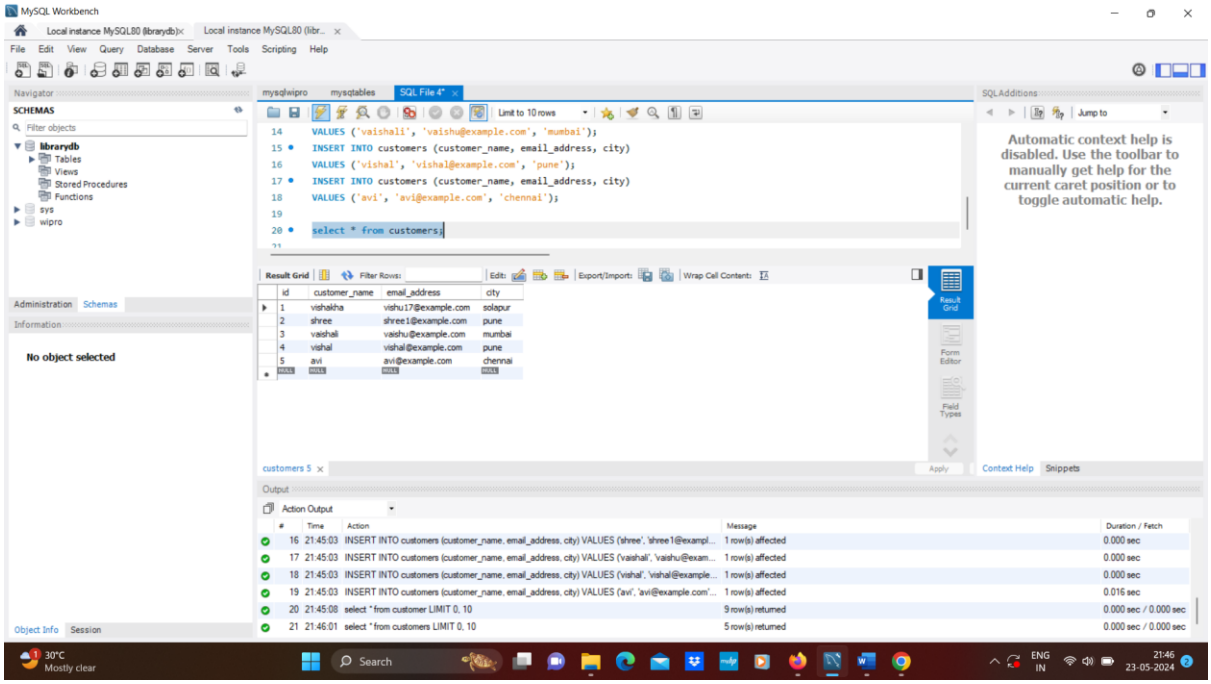
Assignment 1: Write a SELECT query to retrieve all columns from a 'customers' table, and modify it to return only the customer name and email address for customers in a specific city.

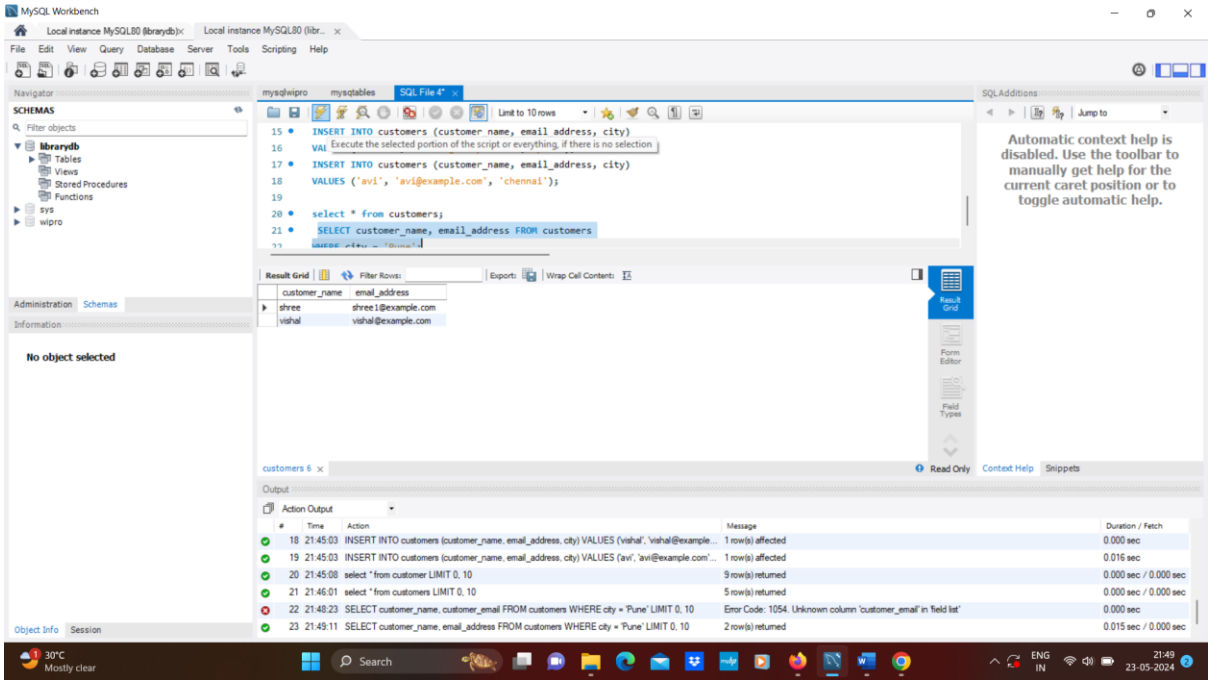
1. SELECT query to retrieve all columns from a 'customers'

SELECT * FROM customers;



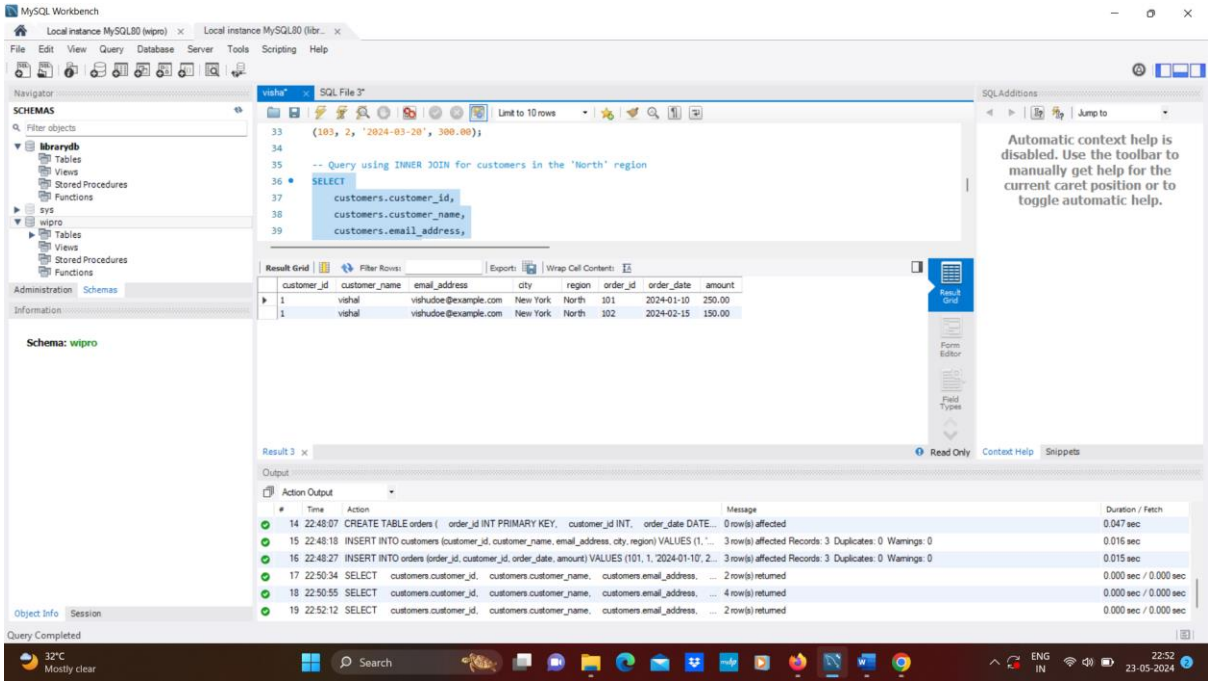
2. modify it to return only the customer name and emailaddress for customers in a specific city.

SELECT customer_name, email_address FROM customers
WHERE city = 'Pune';



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

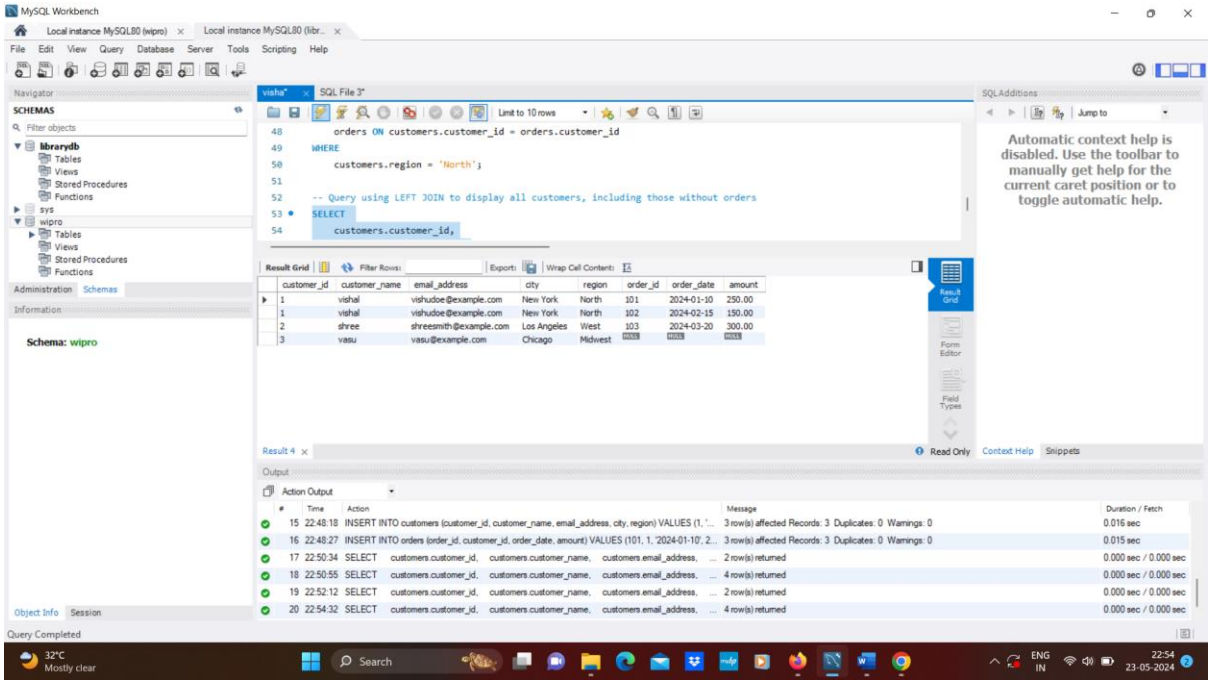
```
SELECT
    customers.customer_id,
    customers.customer_name,
    customers.email_address,
    customers.city,
    customers.region,
    orders.order_id,
    orders.order_date,
    orders.amount
FROM
    customers
INNER JOIN
    orders ON customers.customer_id = orders.customer_id
WHERE
    customers.region = 'North';
```



a LEFT JOIN to display all customers including those without orders

```
SELECT
    customers.customer_id,
```

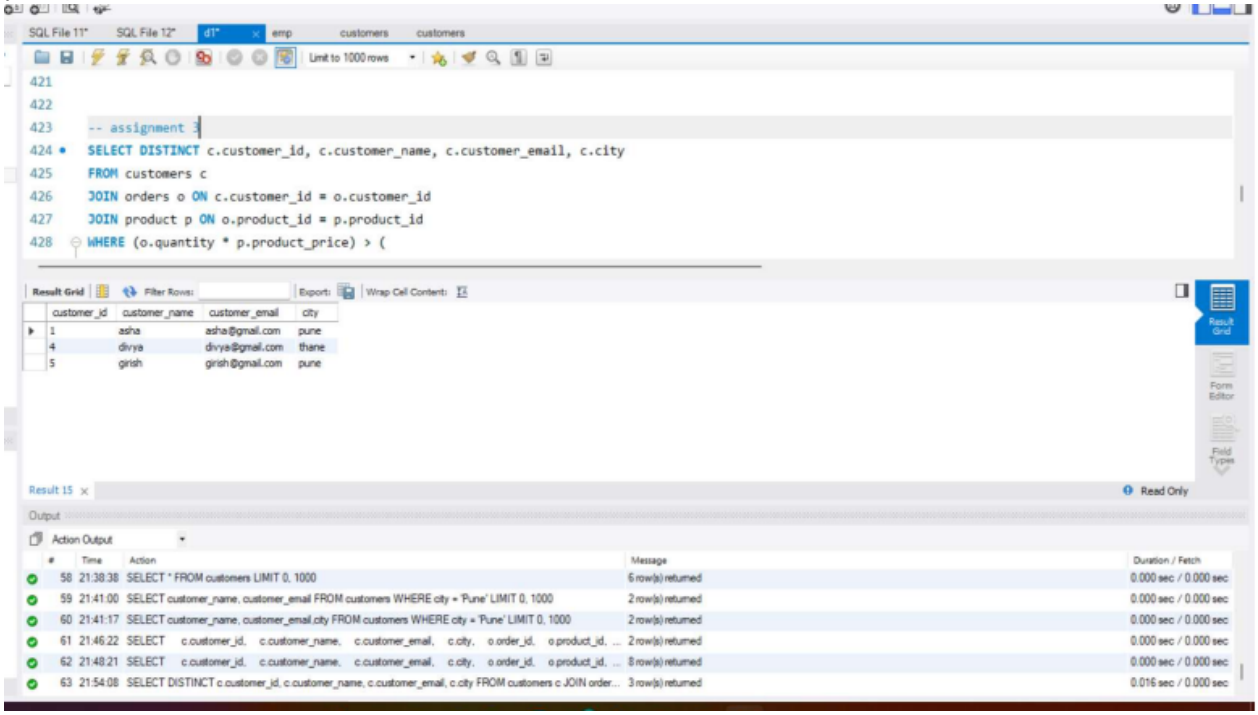
```
    customers.customer_name,
    customers.email_address,
    customers.city,
    customers.region,
    orders.order_id,
    orders.order_date,
    orders.amount
FROM
    customers
LEFT JOIN
    orders ON customers.customer_id = orders.customer_id;
```



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

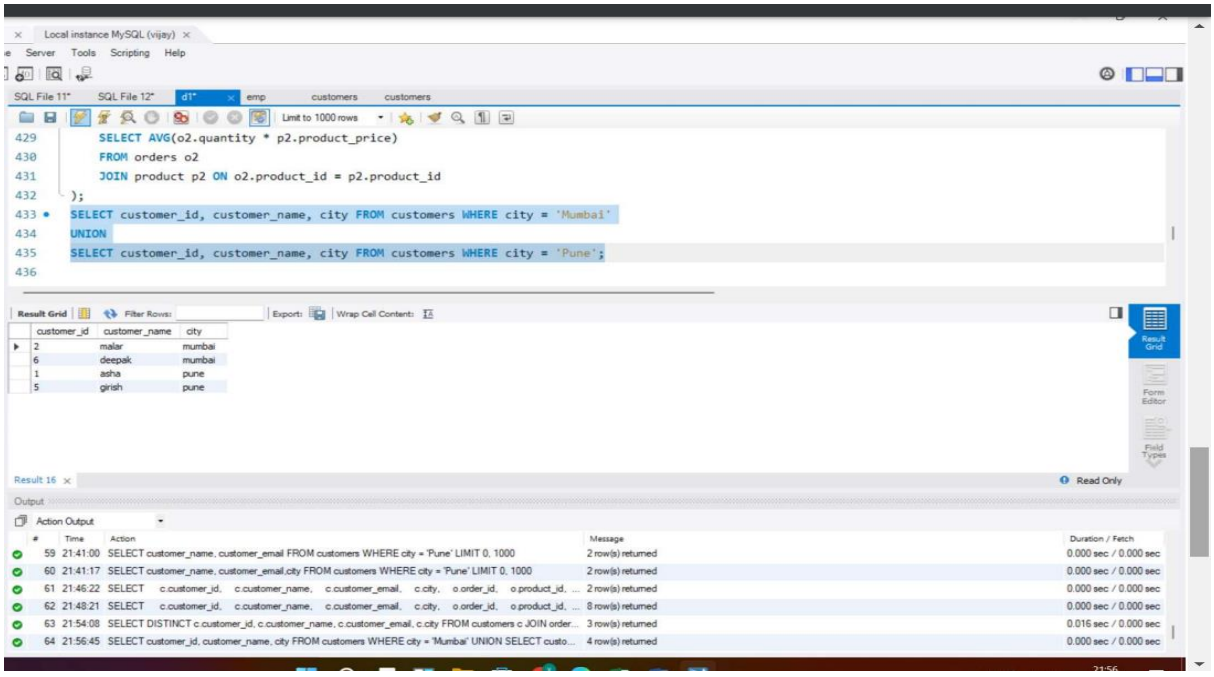
```
SELECT DISTINCT c.customer_id, c.customer_name, c.customer_email, c.city
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN product p ON o.product_id = p.product_id
WHERE (o.quantity * p.product_price) > (
 SELECT AVG(o2.quantity * p2.product_price)
 FROM orders o2
```

JOIN product p2 ON o2.product_id = p2.product_id
);



UNION query to combine two SELECT statements with the same number of columns.

SELECT customer_id, customer_name, city FROM customers WHERE city = 'Mumbai'
UNION
SELECT customer_id, customer_name, city FROM customers WHERE city = 'Pune';



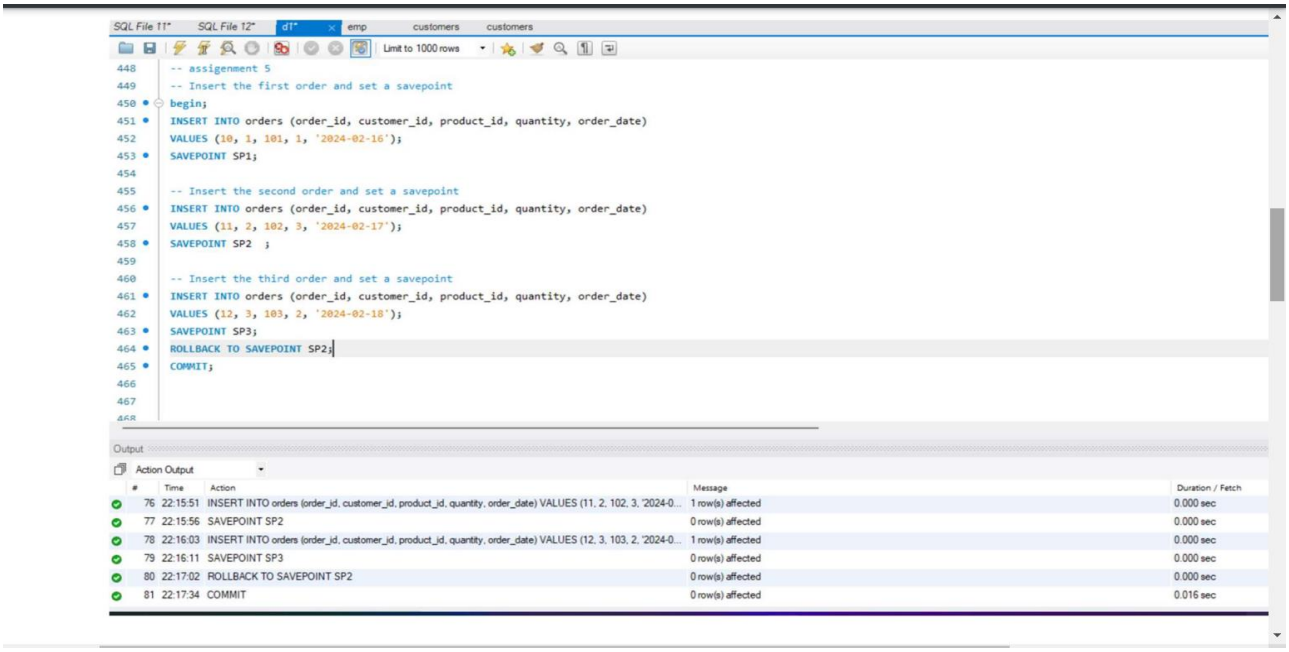
Assignment 4: Compose SQL statements to BEGIN atransaction, INSERT a new record into the 'orders' table,COMMIT the transaction, then UPDATE the 'products' table,and ROLLBACK the transaction.

```
BEGIN;
INSERT INTO orders (order_id, customer_id, product_id, quantity, order_date)
VALUES (8, 1, 101, 2, '2024-02-15');
COMMIT;
BEGIN;
UPDATE product
SET product_price = product_price * 1.1 -- Increasing price by 10%
WHERE product_id = 101;
ROLLBACK;
```

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

```
Begin;
INSERT INTO orders (order_id, customer_id, product_id, quantity, order_date)
VALUES (10, 1, 101, 1, '2024-02-16');
SAVEPOINT SP1;
INSERT INTO orders (order_id, customer_id, product_id, quantity, order_date)
VALUES (11, 2, 102, 3, '2024-02-17');
SAVEPOINT SP2;
```

```
INSERT INTO orders (order_id, customer_id, product_id, quantity, order_date)
VALUES (12, 3, 103, 2, '2024-02-18');
SAVEPOINT SP3;
ROLLBACK TO SAVEPOINT SP2;
COMMIT;
```



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

Transaction logs are crucial components of database management systems (DBMS) designed to maintain a continuous record of all changes made to the data within a database. Every transaction that modifies, adds, or deletes data is sequentially logged with complete details regarding the type of change, the data affected, and the time of the transaction. This meticulous recording allows databases not only to maintain data integrity but also to facilitate recovery operations in case of failures.

Key Functions of Transaction Logs

1. Data Recovery: Transaction logs play a vital role in data recovery processes. They ensure that any changes made during incomplete transactions at the time of a system failure can either be rolled back or completed during

system recovery.
2. Audit and Compliance: Transaction logs provide a traceable history of all data interactions, which is critical for auditing and compliance purposes.
3. Replication: In distributed databases, transaction logs are used to replicate data changes across different database systems, ensuring consistency across geographically dispersed infrastructure.

Hypothetical Scenario: Recovery After an Unexpected Shutdown

Scenario Description

Consider a financial services company, "FinCorp," which manages an online trading platform. The database holds critical information such as user profiles, transaction records, and trading histories. One day, due to an unforeseen power outage, the main data center experiences an abrupt shutdown. This incident occurs during a high-volume trading period, and there are multiple transactions related to stock trades in process.

Role of Transaction Logs in Recovery

Upon restoration of power, the DBMS initiates a recovery process. Here's how transaction logs are used:
1. Analysis of Logs: The system starts by analyzing the transaction logs, identifying the last transaction checkpoint where the database was in a consistent state.
2. Redo Operations: Transactions that had been committed prior to the shutdown and were logged after the last checkpoint are replayed. These redo operations ensure that all committed transactions are restored to the database even if they were not fully written to disk before the shutdown.
3. Undo Operations: Transactions that were in progress and not committed at the time of the shutdown are identified. The logs provide the necessary information to reverse (undo) these

transactions, ensuring that the database remains in a consistent state without partial or
corrupt data entries.