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

```
mysql> SELECT * FROM customer;

+-----+--------+------------------+-----------+---------+------+

| Cid | CName  | email            | City      | Pincode | age  |

+-----+--------+------------------+-----------+---------+------+

| 100 | lokesh | lokesh@gmail.com | bangalore |  560068 |   21 |

| 101 | ravi   | ravi@gmail.com   | BANGALORE |  560068 |   22 |

+-----+--------+------------------+-----------+---------+------+

2 rows in set (0.02 sec)


mysql> SELECT name, email FROM customers WHERE city = 'bangalore';

ERROR 1146 (42S02): Table 'wipro.customers' doesn't exist

mysql> SELECT name, email FROM customer WHERE city = 'bangalore';

ERROR 1054 (42S22): Unknown column 'name' in 'field list'

mysql> SELECT cname, email FROM customer WHERE city = 'bangalore';

+--------+------------------+

| cname  | email            |

+--------+------------------+

| lokesh | lokesh@gmail.com |

| ravi   | ravi@gmail.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 those without orders.

```
mysql> SELECT c.CustomerID, c.FirstName, c.LastName, o.OrderID, o.OrderDate, o.TotalAmount

    -> FROM customers c

    -> INNER JOIN orders o ON c.CustomerID = o.CustomerID
    -> WHERE c.City = 'bangalore';
+------------+-----------+----------+---------+------------+-------------+
| CustomerID | FirstName | LastName | OrderID | OrderDate  | TotalAmount |
+------------+-----------+----------+---------+------------+-------------+
|          1 | Alice     | Johnson  |       1 | 2024-05-01 |      720.00 |
|          2 | Bob       | Smith    |       2 | 2024-05-02 |       45.00 |
|          6 | Frank     | Wright   |       6 | 2024-05-06 |       75.00 |
|          7 | Grace     | Hall     |       7 | 2024-05-07 |      200.00 |
|          8 | Hank      | Moore    |       8 | 2024-05-08 |        NULL |
+------------+-----------+----------+---------+------------+-------------+
5 rows in set (0.00 sec)

mysql> SELECT c.CustomerID, c.FirstName, c.LastName, o.OrderID, o.OrderDate, o.TotalAmount
    -> FROM customers c
    -> LEFT JOIN orders o ON c.CustomerID = o.CustomerID;
+------------+-----------+----------+---------+------------+-------------+
| CustomerID | FirstName | LastName | OrderID | OrderDate  | TotalAmount |
+------------+-----------+----------+---------+------------+-------------+
|          1 | Alice     | Johnson  |       1 | 2024-05-01 |      720.00 |
|          2 | Bob       | Smith    |       2 | 2024-05-02 |       45.00 |
|          3 | Charlie   | Brown    |    NULL | NULL       |        NULL |
|          4 | David     | Lee      |    NULL | NULL       |        NULL |
|          5 | Eva       | Green    |    NULL | NULL       |        NULL |
|          6 | Frank     | Wright   |       6 | 2024-05-06 |       75.00 |
|          7 | Grace     | Hall     |       7 | 2024-05-07 |      200.00 |
|          8 | Hank      | Moore    |       8 | 2024-05-08 |        NULL |
|          9 | Ivy       | King     |       9 | 2024-05-09 |        NULL |
|         10 | Jake      | Stone    |      10 | 2024-05-10 |        NULL |
|         11 | John      | Doe      |    NULL | NULL       |        NULL |
+------------+-----------+----------+---------+------------+-------------+
11 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> SELECT c.CustomerID, c.FirstName, c.LastName
    -> FROM customers c
    -> WHERE c.CustomerID IN (
    ->     SELECT o.CustomerID
    ->     FROM orders o
    ->     WHERE o.TotalAmount > (SELECT AVG(TotalAmount) FROM orders)
    -> );
+------------+-----------+----------+
| CustomerID | FirstName | LastName |
+------------+-----------+----------+
|          1 | Alice     | Johnson  |
+------------+-----------+----------+
1 row in set (0.01 sec)

mysql> SELECT FirstName, LastName, Email FROM customers WHERE City = 'bangalore'
    -> UNION
    -> SELECT FirstName, LastName, Email FROM customers WHERE OrderCount > 0;
+-----------+----------+---------------------------+
| FirstName | LastName | Email                     |
+-----------+----------+---------------------------+
| Alice     | Johnson  | alice.johnson@example.com |
| Bob       | Smith    | bob.smith@example.com     |
| Frank     | Wright   | frank.wright@example.com  |
| Grace     | Hall     | grace.hall@example.com    |
| Hank      | Moore    | hank.moore@example.com    |
| Ivy       | King     | ivy.king@example.com      |
| Jake      | Stone    | jake.stone@example.com    |
+-----------+----------+---------------------------+
7 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> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> INSERT INTO orders (OrderID, CustomerID, OrderDate, TotalAmount, Status)
    -> VALUES (11, 1, '2024-05-12', 150.00, 'pending');
ERROR 1062 (23000): Duplicate entry '11' for key 'orders.PRIMARY'
mysql>
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

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

mysql>
mysql> UPDATE products SET Price = Price + 5.00 WHERE ProductID = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 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> BEGIN;

Query OK, 0 rows affected (0.00 sec)


mysql>

mysql> INSERT INTO orders (OrderID, CustomerID, OrderDate, TotalAmount, Status)

    -> VALUES (12, 2, '2024-05-12', 200.00, 'pending');

Query OK, 1 row affected (0.00 sec)


mysql> SAVEPOINT sp1;

Query OK, 0 rows affected (0.00 sec)


mysql>

mysql> INSERT INTO orders (OrderID, CustomerID, OrderDate, TotalAmount, Status)

    -> VALUES (13, 3, '2024-05-13', 300.00, 'pending');

Query OK, 1 row affected (0.00 sec)


mysql> SAVEPOINT sp2;

Query OK, 0 rows affected (0.00 sec)


mysql>

mysql> INSERT INTO orders (OrderID, CustomerID, OrderDate, TotalAmount, Status)

    -> VALUES (14, 4, '2024-05-14', 400.00, 'pending');

Query OK, 1 row affected (0.00 sec)


mysql> SAVEPOINT sp3;

Query OK, 0 rows affected (0.00 sec)


mysql>
mysql> ROLLBACK TO SAVEPOINT sp2;
Query OK, 0 rows affected (0.00 sec)

mysql>
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.

Report:

Transaction logs are an essential feature in database management systems for ensuring data integrity and consistency. These logs record every change made to the database, which allows for the recovery of data in case of unexpected failures.

In case of a sudden shutdown, the transaction log can be used to recover the database by replaying committed transactions and rolling back any uncommitted ones. For example, if an order insertion was in progress when the system crashed, the transaction log will indicate the uncommitted state and roll back the incomplete transaction.

Hypothetical Scenario:

Imagine a scenario where a customer places an order, and the system crashes before the transaction is fully committed. After the server restarts, the transaction log will show that the order was only partially inserted, and the database will roll back the transaction, ensuring that the customer's order is not duplicated or incomplete. The system will also apply all successfully committed transactions to restore the database to its most recent consistent state.