Assignment 1

Q. 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.

Answer:

Retrieve All Columns:

```
SELECT * FROM customers;
```

Retrieve Customer Name and Email Address for Customers in a Specific City:

```
SELECT customer_name, email_address
FROM customers
WHERE city = 'SpecificCity';
```

Assignment 2

Q. 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.

A

INNER JOIN to Combine 'orders' and 'customers' for Customers in a Specified Region:

```
SELECT orders.*, customers.*
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE customers.region = 'SpecifiedRegion';
```

LEFT JOIN to Display All Customers Including Those Without Orders:

```
SELECT customers.*, orders.*
FROM customers
LEFT JOIN orders ON customers.customer_id = orders.customer_id;
```

Assignment 3

Q. 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.

Answer:

Subquery to Find Customers Who Have Placed Orders Above the Average Order Value:

```
SELECT customer_id, customer_name
FROM customers
WHERE customer_id IN (
   SELECT customer_id
   FROM orders
   GROUP BY customer_id
   HAVING AVG(order_value) > (
      SELECT AVG(order_value)
      FROM orders
   )
);
```

UNION Query to Combine Two SELECT Statements with the Same Number of Columns:

```
SELECT customer_id, customer_name
FROM customers
WHERE region = 'Region1'
UNION
SELECT customer_id, customer_name
FROM customers
WHERE region = 'Region2';
```

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.

BEGIN a Transaction, INSERT a New Record, and COMMIT:

```
BEGIN TRANSACTION;
INSERT INTO orders (order_id, customer_id, order_date, order_value)
VALUES (123, 456, '2024-05-24', 99.99);

COMMIT;
```

UPDATE and ROLLBACK:

```
BEGIN TRANSACTION;
UPDATE products
SET stock_quantity = stock_quantity - 10
WHERE product_id = 789;
ROLLBACK;
```

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.

Transaction with SAVEPOINTs:

```
BEGIN TRANSACTION;

INSERT INTO orders (order_id, customer_id, order_date, order_value)
VALUES (201, 101, '2024-05-24', 150.00);
SAVEPOINT sp1;

INSERT INTO orders (order_id, customer_id, order_date, order_value)
VALUES (202, 102, '2024-05-24', 200.00);
SAVEPOINT sp2;

INSERT INTO orders (order_id, customer_id, order_date, order_value)
VALUES (203, 103, '2024-05-24', 250.00);
SAVEPOINT sp3;

ROLLBACK TO sp2;

COMMIT;
```

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.

Answer:

Transaction Logs Overview:

Purpose: Record all changes made to the database.

Uses: Data recovery, replication, and audit trails.

Scenario: Unexpected shutdown at 3:00 PM, last backup at 2:00 PM.

Steps for Data Recovery:

Restore from Backup: Use the 2:00 PM backup.

Apply Transaction Logs: Replay logs from 2:00 PM to 3:00 PM to redo committed transactions and undo incomplete ones.

Example:

-- Restore from the last backup

RESTORE DATABASE mydatabase FROM DISK = ' D:\Wipro-Assignment\Wipro-Assignment\M4-RDBMSandSQL\ backup_path';

-- Apply transaction logs

RESTORE LOG mydatabase FROM DISK = ' D:\Wipro-Assignment\Wipro-Assignment\M4-RDBMSandSQL\ transaction_log_path';

---END---