Que 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.

Ans.

```
  select *from customers;

select customer_name, email_address

from customers

where city = 'Pune';
```

Que 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.

Ans.

```
select o.order_id, c.customer_id, c.customer_name, c.email_address, o.order_date from customers  c

inner join orders o  on o.customer_id=c.customer_id

where c.region=”Specified Region”;
```

Que 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.

Ans.

```
SELECT c.customer_name, c.email_address

FROM customers c

WHERE c.customer_id IN (

   SELECT o.customer_id

   FROM orders o

   WHERE o.total_amount > (

      SELECT AVG(total_amount)

      FROM orders

   )

);
```

Que 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.

Ans.

```
BEGIN TRANSACTION;


INSERT INTO orders (customer_id, order_date, total_amount)

VALUES (123, '2024-05-25', 150.00);


COMMIT;


UPDATE products

SET price = price * 1.1  -- Increase price by 10%

WHERE category = 'Electronics';

ROLLBACK;
```

Que. 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.

Ans.

```
BEGIN TRANSACTION;

INSERT INTO orders (customer_id, order_date, total_amount)

VALUES (123, '2024-05-25', 150.00);


SAVEPOINT savepoint1;

INSERT INTO orders (customer_id, order_date, total_amount)

VALUES (456, '2024-05-26', 200.00);


SAVEPOINT savepoint2;


INSERT INTO orders (customer_id, order_date, total_amount)
```

```
VALUES (789, '2024-05-27', 300.00);

ROLLBACK TO SAVEPOINT savepoint2;

COMMIT;
```

Que. 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.

Ans.

Report: Transaction Logs for Data Recovery

Introduction: Transaction logs are crucial for database management systems, recording transactions and enabling data recovery. This report highlights their importance and illustrates their role through a hypothetical scenario.

Importance of Transaction Logs:

1. Continuous Data Protection**:** Logs capture transactions in real-time, ensuring no data loss during failures.
2. Point-in-Time Recovery**:** Allows restoring databases to specific points before failures.
3. Rollback and Rollforward**:** Supports undoing uncommitted transactions and reapplying transactions post-recovery.

Scenario: A retail database crashes during peak hours, causing inventory discrepancies.

Using Transaction Logs:

1. Failure Analysis: Logs help pinpoint the failure's timing.
2. Recovery: Logs guide the restoration process.
3. Transaction Replay**:** Missing transactions are reapplied.
4. Verification**:** Data integrity is confirmed post-recovery.

Conclusion: Transaction logs are indispensable for swift and reliable data recovery, ensuring minimal disruptions in operations following unexpected events.