# Assignment no 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.

SELECT \* FROM Customers;  
SELECT name AS customer\_name, email AS customer\_email  
FROM Customers

WHERE city = 'kanpur';

# Assignment no 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.

SELECT c.\*, IFNULL(o.orders, '') AS Orders  
FROM Customers c  
LEFT JOIN orders\_o o ON c.Customers = o.Customers  
WHERE c.region = 'specified\_region'  
LIMIT 0, 1000;

# Assignment no 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

Query 1: Find customers who have placed orders above the average order value using a subquery:

SELECT customer\_id, order\_id, total

FROM orders

WHERE total > (SELECT AVG(total) FROM orders)

This query uses a subquery to calculate the average order value and then selects customers who have placed orders with a total value greater than the average.

Query 2: UNION query to combine two SELECT statements with the same number of columns:

SELECT customer\_id, order\_id, total

FROM orders

WHERE order\_id < 1000

UNION

SELECT customer\_id, order\_id, total

FROM orders

WHERE order\_id > 2000

# Assignment no 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.

1. BEGIN transaction:

BEGIN TRANSACTION;

1. INSERT a new record into the 'orders' table:

INSERT INTO orders (customer\_id, order\_date, total)

VALUES (1, '2024-06-17', 100.00);

1. COMMIT the transaction:

COMMIT;

1. UPDATE the 'products' table:

UPDATE products

SET quantity = quantity - 1

WHERE product\_id = 1;

1. ROLLBACK the transaction (note: this will undo the UPDATE operation):

ROLLBACK;

explanation of each statement:

- BEGIN TRANSACTION starts a new transaction.

- INSERT INTO orders adds a new record to the orders table.

- COMMIT commits the transaction, making the changes permanent.

- UPDATE products updates the quantity of a product.

- ROLLBACK rolls back the transaction, undoing the changes made since the beginning of the transaction.

# 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.

1. BEGIN transaction:

BEGIN TRANSACTION;

1. INSERT into 'orders' and set SAVEPOINT:

INSERT INTO orders (customer\_id, order\_date, total)

VALUES (1, '2024-06-17', 100.00);

SAVEPOINT sp1;

1. INSERT into 'orders' and set SAVEPOINT:

INSERT INTO orders (customer\_id, order\_date, total)

VALUES (2, '2024-06-18', 200.00);

SAVEPOINT sp2;

1. INSERT into 'orders' and set SAVEPOINT:

INSERT INTO orders (customer\_id, order\_date, total)

VALUES (3, '2024-06-19', 300.00);

SAVEPOINT sp3;

1. ROLLBACK to the second SAVEPOINT:

ROLLBACK TO SAVEPOINT sp2;

1. COMMIT the overall transaction:

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.has context menu

Report: Utilizing Transaction Logs for Data Recovery

Transaction logs are a crucial component of database management systems, enabling the consistent recovery of data in the event of unexpected failures or shutdowns. By recording all changes made to the database, transaction logs provide a chronological record of transactions, allowing for precise restoration of data to a consistent state.

Hypothetical Scenario:

Company X's e-commerce database experiences an unexpected shutdown due to a power outage. Upon restarting the system, database administrators discover that the last backup was taken several hours prior, resulting in potential data loss. Fortunately, the transaction log is intact and utilized for data recovery.

Recovery Process:

1. Database administrators replay the transaction log, reapplying all committed transactions since the last backup.

2. The log reveals a series of transactions, including orders placed, inventory updates, and customer information changes.

3. The database is restored to its consistent state, reflecting all changes made up to the point of shutdown.

Thanks to the transaction log, Company X recovers all data modifications made during the critical period, ensuring business continuity and minimizing potential losses.

Context Menu:

- Transaction Log: A chronological record of database transactions.

- Data Recovery: The process of restoring data to a consistent state after a failure.

- Database Consistency: Ensuring data accuracy and integrity.

- Backup: A copy of data created at a specific point in time.

- Replay: Reapplying transactions from the log to restore data.