1. What is DML in SQL?

DML (Data Manipulation Language) is a subset of SQL that allows you to manipulate data in a database. It includes commands to insert, update, delete, and retrieve data from tables.

2. Name all DML commands in SQL.

The main DML commands in SQL are:

- **INSERT** – Adds new records to a table.
- **UPDATE** – Modifies existing records in a table.
- **DELETE** – Removes specific records from a table.
- **SELECT** – Retrieves data from a table (though often categorized under DQL - Data Query Language).

3. How do you add a new row into a table?

To insert a new row into a table, use the INSERT INTO statement.

Example:

INSERT INTO employees (id, name, age, department)

VALUES (1, 'John Doe', 30, 'HR');

This adds a new record to the employees table.

4. How do you update an existing row in a table?

To modify existing data, use the UPDATE statement along with the WHERE condition to specify which record(s) to change.

Example:

UPDATE employees

SET age = 31, department = 'Finance'

WHERE id = 1;

This updates the employee with id = 1 to set their age to 31 and change their department to "Finance."

5. What is the difference between DELETE and TRUNCATE?

Feature	DELETE	TRUNCATE
Removes specific rows?	Yes (can use WHERE)	No (removes all rows)
Can be rolled back?	Yes (if inside a transaction)	No (cannot be rolled back in most databases)
Speed	Slower (logs each row deletion)	Faster (resets the table instantly)
Resets Auto-Increment ID?	No	Yes
Triggers execution?	Yes	No

Example:

- **DELETE:**

  DELETE FROM employees WHERE department = 'HR';

  (Removes only employees from the "HR" department.)

- **TRUNCATE:**
  TRUNCATE TABLE employees;
  (Removes all records from the employees table but keeps the table structure.)

6. What happens if you try to insert data into a table without specifying all columns?

- If the **unspecified columns have default values**, those values will be used.
- If a column **does not allow NULL values** and has no default value, the insert will fail.

Example:

INSERT INTO employees (id, name) VALUES (2, 'Alice');

If age and department have default values or allow NULLs, the query works.

If not, it throws an error.

7. Use the DELETE statement with a WHERE condition to remove a specific record.

Example:

How do you delete a specific record from a table?

DELETE FROM employees WHERE id = 2;

This deletes the employee whose id is 2.

8. What is a transaction in SQL?

A transaction is a sequence of SQL operations that are executed as a single unit. It ensures data consistency and follows the ACID properties (Atomicity, Consistency, Isolation, Durability).

Example:

- If a bank transfer involves debiting one account and crediting another, both must happen together.
- If one fails, the entire transaction should be undone.

BEGIN TRANSACTION;

UPDATE accounts SET balance = balance - 500 WHERE account_id = 1;

UPDATE accounts SET balance = balance + 500 WHERE account_id = 2;

COMMIT;

9. What is the purpose of the ROLLBACK statement?

The ROLLBACK statement undoes changes made in the current transaction if an error occurs or the user decides to cancel the operation.

Example:

BEGIN TRANSACTION;

UPDATE employees SET salary = salary + 500 WHERE id = 3;

ROLLBACK;

If something goes wrong, ROLLBACK cancels the update, and the salary remains unchanged.

10. What does COMMIT do in a transaction?

The COMMIT statement saves all changes made in the current transaction permanently to the database.

Example:

BEGIN TRANSACTION;

UPDATE employees SET salary = salary + 500 WHERE id = 3;

COMMIT;

After COMMIT, the salary change is permanent, and it **cannot** be undone with ROLLBACK.

11. Write a query to insert 5 new employees into an Employees table.

To insert multiple records at once, use the INSERT INTO statement with multiple VALUES sets.

Example:

INSERT INTO Employees (EmployeeID, Name, Age, Department, Salary)

VALUES

(1, 'John Doe', 30, 'HR', 50000),

(2, 'Alice Smith', 28, 'Finance', 55000),

(3, 'Bob Johnson', 35, 'IT', 60000),

(4, 'Emily Davis', 26, 'Marketing', 48000),

(5, 'Michael Brown', 40, 'Sales', 65000);

This will insert 5 employees into the Employees table.

12. How do you update the salary of an employee based on their EmployeeID?

Use the UPDATE statement with a WHERE condition to specify the employee whose salary should be updated.

Example:

UPDATE Employees SET Salary = 70000 WHERE EmployeeID = 3;

This updates the salary of the employee with EmployeeID = 3 to 70,000.

13. Write an SQL query to delete all orders placed before 2023.

Use the DELETE statement with a WHERE condition to remove orders placed before 2023.

Example:

DELETE FROM Orders WHERE OrderDate < '2023-01-01';

- This deletes all records where the OrderDate is before January 1, 2023.
- Ensure OrderDate is stored in a proper **DATE or DATETIME** format.

14. How do you rollback a transaction after updating multiple records?

To rollback a transaction, you must first begin the transaction, perform the updates, and then decide whether to ROLLBACK (cancel changes) or COMMIT (save changes).

Example:

BEGIN TRANSACTION;

UPDATE Employees SET Salary = Salary + 5000 WHERE Department = 'IT';

UPDATE Employees SET Salary = Salary + 3000 WHERE Department = 'Finance';

ROLLBACK;

- If there is an issue or you change your mind, ROLLBACK will **undo all salary updates** made in this transaction.
- If you want to **save the changes**, replace ROLLBACK; with COMMIT;.

15. What happens if an UPDATE statement does not have a WHERE clause?

If an UPDATE statement does not include a WHERE clause, **all rows** in the table will be updated.

Example:

UPDATE Employees SET Salary = 80000;

This sets the salary of every employee in the table to 80,000, which is usually not intended and can cause data loss. Always use a WHERE clause unless you want to update all records.

16. How do you insert a row with a NULL value in a column?

To insert a NULL value into a column, explicitly specify NULL in the VALUES list.

Example:

INSERT INTO Employees (EmployeeID, Name, Age, Department, Salary)

VALUES (6, 'Sophia Wilson', NULL, 'HR', 50000);

- Here, the Age column is set to NULL for employee Sophia Wilson.
- This works only if the column allows NULL values.

17. Write an SQL query to increase the price of all products by 10%.

Use the UPDATE statement with a mathematical operation.

Example:

UPDATE Products SET Price = Price * 1.10;

- This increases the price of **every product** by **10%**.
- If Price = 100, after the update, it becomes 110.

18. How do you use a transaction to transfer money between two accounts?

To transfer money safely, use a transaction to ensure both operations (debit and credit) happen together. If one fails, the entire transaction should be rolled back.

Example:

BEGIN TRANSACTION;

UPDATE Accounts

SET Balance = Balance - 1000

WHERE AccountID = 1; -- Debit 1000 from Account 1

UPDATE Accounts

SET Balance = Balance + 1000

WHERE AccountID = 2; -- Credit 1000 to Account 2

COMMIT; -- Save the transaction

If the second UPDATE fails (e.g., Account 2 doesn't exist), use ROLLBACK; instead of COMMIT; to cancel the transaction.

19. How do you ensure that deleting a customer also deletes their orders?

Use foreign key constraints with ON DELETE CASCADE so that when a customer is deleted, all their associated orders are also removed automatically.

Example:

CREATE TABLE Orders (

OrderID INT PRIMARY KEY, CustomerID INT, OrderDate DATE, FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE);

Now, if a customer is deleted from the Customers table, all their orders in the Orders table will be automatically deleted.

20. What is the effect of SET AUTOCOMMIT = OFF; in a transaction?

By default, SQL automatically commits every query. If you set AUTOCOMMIT = OFF, SQL will wait for an explicit COMMIT; before saving changes.

Effect of SET AUTOCOMMIT = OFF;

- Queries won't be committed automatically.
- You must manually use COMMIT; to save or ROLLBACK; to undo changes.

Example:

SET AUTOCOMMIT = OFF;

UPDATE Employees SET Salary = 60000 WHERE EmployeeID = 1;

-- Change is NOT saved yet

COMMIT; -- Now the change is saved

21. Design a database for an online shopping system and insert sample records.

// Refer Folder #ShopSystem For Tables and Queries

22. Write a query to update an employee's department if they change jobs.

UPDATE Staff SET Department = 'Human Resources'   WHERE EmployeeID = 3;

(This assumes a Staff table with EmployeeID and Department columns.)

23. How do you delete all students who have not registered for any courses?

DELETE FROM Students

WHERE StudentID NOT IN (SELECT DISTINCT StudentID FROM Registrations);

(This assumes a Students table and a Registrations table that links students to courses.)

24. Create an inventory system where a transaction updates product stock levels.

// Refer Folder #InventorySystem For Tables and Queries

25. Implement a banking transaction where money is withdrawn from one account and deposited into another.

BEGIN TRANSACTION;

UPDATE BankAccounts
SET Balance = Balance - 2000
WHERE AccountNumber = 'A12345';

UPDATE BankAccounts
SET Balance = Balance + 2000
WHERE AccountNumber = 'B67890';

COMMIT;

26. Write an SQL query to delete duplicate records from a table.

DELETE FROM Employees
WHERE EmployeeID NOT IN (SELECT MIN(EmployeeID)
FROM Employees GROUP BY Email
);
(Deletes duplicate employees keeping only one record per unique email.)

27. How do you update a student's marks after re-evaluation?

```
UPDATE ExamResults
SET Marks = 85
WHERE StudentID = 2023 AND SubjectID = 'Math101';
```

28. Write an SQL query to delete all customers who have not made a purchase in 2 years.

// Refer Folder #CustTransction For Tables and Queries for Q.28,29

```
DELETE FROM Clients WHERE ClientID NOT IN (
    SELECT DISTINCT ClientID FROM Transactions
    WHERE TransactionDate >= DATE_SUB(CURDATE(), INTERVAL 2 YEAR)
);
```

29. Create an SQL script to insert new sales transactions and commit them only if all inserts succeed.

```
BEGIN TRANSACTION;

INSERT INTO Sales (SaleID, ProductID, Quantity, SaleDate)
VALUES (7001, 305, 2, '2025-03-14');

INSERT INTO Sales (SaleID, ProductID, Quantity, SaleDate)
VALUES (7002, 310, 3, '2025-03-14');

COMMIT;
```

30. Explain how transactions help in maintaining consistency in financial databases.

Transactions ensure data consistency and accuracy by following the ACID properties:

- **Atomicity** → Ensures all steps succeed or none do (e.g., money transfer).
- **Consistency** → Maintains database integrity (e.g., prevents negative balances).
- **Isolation** → Ensures transactions don't interfere with each other.
- **Durability** → Once committed, changes are permanent (e.g., purchase records).

Example:

If a bank transfer happens and the system crashes after money is deducted but before being credited, a transaction ensures the operation rolls back, preventing data loss.