14 Feb 2026 - SQL Test.

1) Create a customer table & Perform INSERT, UPDATE, DELETE, SELECT.

Create Table.

```
CREATE TABLE customer ( customer_id INT PRIMARY KEY,
 name VARCHAR(50), city VARCHAR (50), age INT );
```

insert:

```
INSERT INTO Customer (customer_id, name, city, age)
 VALUES (1, 'Selva', 'chennai', 25);
```

select:

```
SELECT * from Customer ;
```

update:

```
UPDATE Customer
SET City = 'coimbatore'
WHERE Customer_id =1;
```

delete:

```
DELETE FROM Customer
WHERE Customer_id =1;
```

2] Difference between DELETE and TRUNCATE:

DELETE	TRUNCATE
1. Removes specific rows using where	Removes all Rows
2. call rollback	cannot Rollback
3. Slower	Faster
4. keeps table Structure	keeps table structure
5. Identity not reset	Identity reset

3) Difference between WHERE clause & Having clause.

WHERE	HAVING:
1. filters rows before grouping	filter after GROUP By
2. used with SELECT, UPDATE, DELETE	used with aggregate functions
3. Cannot use SUM( ), COUNT( ) directly	used with SUM( ), COUNT( ), AVG( )

Example:

```
SELECT department, count(*)
FROM Employee
WHERE age > 25
GROUP BY department
HAVING COUNT (*) > 2;
```

4) Difference between ALTER, UPDATE, DELETE, and DROP command:

1. ALTER	-	change table structure.
2. UPDATE	-	Modify existing data.
3. DELETE	-	Remove rows
4. DROP	-	Remove table completely.

5] Query for add column, drop column, modify size, and modify data type.

Add column:

```
ALTER TABLE customer
ADD email VARCHAR(50);
```

DROP column :-

```
ALTER TABLE Customer
DROP COLUMN email;
```

Modify size :-

```
ALTER TABLE Customer
MODIFY name VARCHAR(100);
```

Modify datatype :

```
ALTER TABLE Customer
MODIFY age BIGINT;
```

6) List all constraints & explain :-

i) Primary key - unique + Not null

ii) Foreign key - Link two tables

iii) unique key - No duplicat values

iv) Not null - Cannot store NULL

v) check - condition validation

vi) Default - Default value

vii) composite key - Multiple columns as key.

7) Query for primary key, unique, foreign key constraint

Primary + unique :-

```
CREATE TABLE Department ( dept_id INT PRIMARY KEY,
 dept_name VARCHAR (50) UNIQUE );
```

Foreign key :-

```
CREATE TABLE Employee ( emp_id INT PRIMARY KEY, emp_name,
  VARCHAR(50), dep_id INT, constraint fk_dept.
  FOREIGN KEY (dept_id)
  REFERENCES Department (dep_id) );
```

8) create Primary key constraint for an existing table.

```
CREATE TABLE Customer ( customer_id INT, name VARCHAR (50),
City VARCHAR (50) );

ALTER TABLE Customer
ADD Constraint Pk_Customer
PRIMARY KEY (Customer_id);
```

9) Types of Joins with example :-

1. INNER JOIN
2. OUTER JOIN
   * LEFT
   * RIGHT
   * Full.
3. SELF JOIN
4. CROSS JOIN.

1. INNER Joins :-

* Returns only matching rows from both tables

Example :

SELECT * FROM Employee.
INNER JOIN Department
ON Employee.dep_id = Department.dept_id;

2. Outer JOIN :-

Returns matching + non-matching rows.

i) Left Outer Join :-

all rows from left table + matching rows from right table.

Ex :-

SELECT * FROM EmpLoyee
LEFT OUTER JOIN Department
ON Employee.dept_id = Department.dep_id;

ii) Right Outer Join :-

All rows from right table + matching rows from left table.

Ex: SELECT * FROM Employee
RIGHT OUTER JOIN Department
ON Employee.dept-id = Department.dept-id;

iii) Full outer JOIN :-

All rows from both tables :

Ex:- SELECT * FROM Employee
Full OUTER JOIN Department
ON Employee . dept-id = Department . dept - id;

3) SELF JOIN :-

* Joining a table with itself.

Ex :- Employee and Manager in same table.

SELECT A . emp_name AS Employee , B.emp_name . As Manager
FROM Employee A
JOIN Employee B
ON A.manager_id = B. emp-id;

4) CROSS JOIN :-

Returns all combinations of rows.

Ex :- SELECT * FROM Employee
CROSS JOIN· Department :

10] Aggregate Functions with example:.

Functions :

* COUNT ()
* SUM()
* AVG ()
* MIN()
* MAX()

Example :

```
SELECT COUNT (*) AS total_employees,
SUM (salary )AS total_salary,
AVG ( salary) as average_salary,
max (salary) as highest_salary,
MIN ( salary ) as lowest_salary
FROM Employees;
```