SQL SESSION NOTES:

SQL commands are categorized into four main types:

a) Data Definition Language (DDL)

DDL commands are used to define and modify database structures (schemas, tables, etc.).

CREATE – Creates a new database object (table, view, index, etc.).

ALTER – Modifies an existing database object (add/remove columns, change data type).

DROP – Deletes an object permanently from the database.

TRUNCATE – Removes all rows from a table without logging individual row deletions.

RENAME – Changes the name of an existing database object.

b) Data Manipulation Language (DML)

DML commands deal with data manipulation (insertion, modification, and deletion).

INSERT – Adds new records to a table.

UPDATE – Modifies existing records in a table.

DELETE – Removes specific records from a table.

MERGE – Performs an upsert (combination of insert, update, delete based on conditions).

c) Data Query Language (DQL)

DQL commands are used to retrieve data from the database.

SELECT – Fetches data from one or more tables with optional filtering, sorting, and aggregation.

d) Data Control Language (DCL)

DCL commands are used for access control and permissions.

GRANT – Provides specific privileges to users.

REVOKE – Removes privileges from users.

e) Transaction Control Language (TCL)

TCL commands manage transactions in a database.

COMMIT – Saves changes permanently.

ROLLBACK – Undoes changes since the last commit.

Joins in SQL

Joins are used to retrieve data from multiple tables based on related columns.

a) INNER JOIN

Returns only matching rows from both tables.

b) LEFT JOIN (LEFT OUTER JOIN)

Returns all rows from the left table and matching rows from the right table; unmatched rows from the right table are returned as NULL.

c) RIGHT JOIN (RIGHT OUTER JOIN)

Returns all rows from the right table and matching rows from the left table; unmatched rows from the left table are returned as NULL.

d) FULL JOIN (FULL OUTER JOIN)

Returns all rows when there is a match in either table; non-matching rows contain NULL.

e) CROSS JOIN

Produces a Cartesian product where each row from the first table is paired with every row from the second table.

f) SELF JOIN

A table joins itself based on a related column.

g) NATURAL JOIN

Joins two tables based on columns with the same name and data type.

SQL Keywords and Operators

a) EXISTS

Checks for the existence of rows that meet a subquery condition.

Returns TRUE if the subquery contains at least one row.

b) ANY

Compares a value with a set of values.

Returns TRUE if the condition matches any value in the subquery.

c) ALL

Compares a value with all values in a set.

Returns TRUE only if the condition holds for every value in the subquery.

d) DISTINCT

Eliminates duplicate values from query results.

e) GROUP BY

Groups rows based on specified columns, usually used with aggregate functions.

f) HAVING

Filters grouped records based on aggregate function conditions.

g) ORDER BY

Sorts query results in ascending (ASC) or descending (DESC) order.

h) UNION & UNION ALL

UNION combines result sets and removes duplicates.

UNION ALL combines result sets but retains duplicates.

Aggregate Functions in SQL

Aggregate functions perform calculations on multiple rows and return a single result. They are commonly used with the GROUP BY clause to summarize data.

1. COUNT()

Returns the number of rows in a table or the number of non-null values in a column.

Syntax:

SELECT COUNT(column_name) FROM table_name;

2. SUM()

Calculates the total sum of a numeric column.

Syntax:

SELECT SUM(column_name) FROM table_name;

3. AVG()

Returns the average (mean) value of a numeric column.

Syntax:

SELECT AVG(column_name) FROM table_name;

4. MIN()

Returns the smallest value in a column.

Syntax:

SELECT MIN(column_name) FROM table_name;

5. MAX()

Returns the largest value in a column.

Syntax:

SELECT MAX(column_name) FROM table_name;

Subqueries in SQL

A subquery is a query nested inside another SQL query. It is used to retrieve data that will be used in the main query. Subqueries are often used in WHERE, SELECT, FROM, and HAVING clauses.

Types of Subqueries

- Single-row Subquery
- Multi-row Subquery
- Correlated Subquery
- Nested Subquery
- Scalar Subquery

1. Single-row Subquery

Returns a single value (one row, one column).

Used with comparison operators like =, >, <, >=, <=.

Example:

SELECT name, salary

FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

2. Multi-row Subquery

Returns multiple rows (one column).

Used with IN, ANY, ALL operators.

Using IN

Checks if a value exists in the subquery result.

Example:

```
SELECT name, department
FROM employees
WHERE department IN (SELECT department FROM departments WHERE location = 'India');
```

Using ANY

Compares a value with any value from the subquery.

Example:

```
SELECT name, salary
FROM employees
WHERE salary > ANY (SELECT salary FROM employees WHERE department = 'HR');
```

Using ALL

Compares a value with all values from the subquery.

Example:

```
SELECT name, salary
FROM employees
WHERE salary > ALL (SELECT salary FROM employees WHERE department = 'HR');
```

3. Correlated Subquery

The inner query is executed for each row of the outer query.

Uses values from the outer query.

Example:

```
SELECT name, salary
FROM employees e1
WHERE salary > (SELECT AVG(salary) FROM employees e2 WHERE e1.department = e2.department);
```

4. Nested Subquery

A subquery inside another subquery.

Example:

```
SELECT name

FROM employees

WHERE department_id = (SELECT department_id FROM departments WHERE location = (SELECT
location_id FROM locations WHERE city = 'Bangalore'));
```

5. Scalar Subquery

Returns a single value and can be used in the SELECT clause.

Example:

```
SELECT name,

    (SELECT MAX(salary) FROM employees) AS highest_salary

FROM employees;
```