SQL Day 3: Advanced Queries with Aggregation and Joins

---

1. Aggregating Data

Aggregation functions allow us to perform calculations on a set of values and return a single value. They are commonly used with the GROUP BY clause to summarize data.

Aggregate Functions

Some commonly used aggregate functions in SQL include:

- COUNT(*) – Counts the total number of rows.
- SUM(column_name) – Calculates the total sum of values in a column.
- AVG(column_name) – Computes the average value of a numeric column.
- MAX(column_name) – Returns the highest value in a column.
- MIN(column_name) – Returns the lowest value in a column.

Examples:

```
-- Count the total number of students

SELECT COUNT(*) FROM students;


-- Find the average age of students

SELECT AVG(age) FROM students;
```

GROUP BY (Grouping Data)

The GROUP BY statement is used to arrange identical data into groups. It is often used with aggregate functions.

Example:

```
-- Count the number of students in each grade

SELECT grade, COUNT(*) FROM students GROUP BY grade;
```

HAVING (Filtering Grouped Data)

The HAVING clause filters records after aggregation (unlike WHERE, which filters before aggregation).

Example:

```
-- Show only grades with more than 2 students
SELECT grade, COUNT(*)
FROM students
GROUP BY grade
HAVING COUNT(*) > 2;
```

---

2. Joins (Combining Data from Multiple Tables)

What are Joins?

Joins are used to retrieve data from multiple related tables based on a common column. They help combine meaningful information from different tables.

Types of Joins with Examples

1. INNER JOIN (Matches in Both Tables)

Retrieves records that have matching values in both tables.

Example:

```
SELECT students.name, grades.subject, grades.marks
FROM students
INNER JOIN grades ON students.id = grades.student_id;
```

- Returns students and their grades only if they exist in both tables.

---

2. LEFT JOIN (All Records from Left Table, Matching Right Table)

Returns all records from the left table and only the matching records from the right table. If no match is found, NULL is returned for columns from the right table.

Example:

```
SELECT students.name, grades.subject, grades.marks
FROM students
LEFT JOIN grades ON students.id = grades.student_id;
```

- Includes students even if they don't have any grades.

---

3. RIGHT JOIN (All Records from Right Table, Matching Left Table)

Returns all records from the right table and only matching records from the left table.

Example:

```
SELECT students.name, grades.subject, grades.marks
FROM students
RIGHT JOIN grades ON students.id = grades.student_id;
```

- Includes all subjects with grades, even if some students are missing.

---

4. FULL JOIN (All Records from Both Tables)

Returns all records when there is a match in either table. If no match is found, NULL is returned for missing values. (*Supported in PostgreSQL*)

Example:

```
SELECT students.name, grades.subject, grades.marks
FROM students
FULL JOIN grades ON students.id = grades.student_id;
```

- Includes all students and all subjects, even if some have missing matches.
