--------LECTURE- 11--------

📌 What are MySQL Functions?

Think of MySQL functions like small tools inside SQL that help us perform operations on data.

or

MySQL functions are built-in commands that perform calculations or processing on data and return results automatically.

📌 STRING FUNCTIONS in MySQL

Used to work with text/words/characters inside a column.

---

1. CONCAT()

Used to join/merge two or more strings together.

🔷 Syntax:

```
CONCAT(string1, string2, ...)
```

🔷 Example:

```
SELECT CONCAT(first_name, ' ', last_name) AS Full_Name
FROM students;
```

📌 Output example → Amit Sharma

---

2. LENGTH()

Used to count number of characters in a string.

🔷 Syntax:

```
LENGTH(string)
```

🔷 Example:

```
SELECT full_name, LENGTH(full_name) AS name_length
FROM students;
```

📌 Output → Neha Singh → 10 letters

---

3. UPPER()

Converts text to uppercase.

🔷 Syntax:

```
UPPER(string)
```

🔷 Example:

```
SELECT UPPER(full_name) FROM students;
```

📌 Output → ROHAN VERMA

---

4. LOWER()

Converts text to lowercase.

🔷 Syntax:

LOWER(string)

🔷 Example:

SELECT LOWER(full_name) FROM students;

📌 Output → priya patel

---

5. TRIM()

Removes extra spaces from both sides of text.

🔷 Syntax:

TRIM(string)

🔷 Example:

SELECT TRIM(' Hello SQL ');

📌 Output → Hello SQL (spaces removed)

Variants:

LTRIM(string) -- removes space from LEFT

RTRIM(string) -- removes space from RIGHT

---

6. REPLACE()

Replaces a part of text with another text.

🔷 Syntax:

REPLACE(string, 'old', 'new')

🔷 Example:

SELECT REPLACE('I love SQL', 'SQL', 'MySQL');

📌 Output → I love MySQL

OR inside table:

SELECT REPLACE(product, 'Mob', 'Mobile') FROM sales;

---

7. SUBSTRING()

Extracts a portion of text.

🔷 Syntax:

SUBSTRING(string, start_position, length)

🔷 Example:

SELECT SUBSTRING('Database', 1, 4);

📌 Output → Data

---

8. LEFT() & RIGHT()

Used to take characters from left or right side.

🔷 Syntax:

LEFT(string, number_of_characters)

RIGHT(string, number_of_characters)

Example:

SELECT LEFT(full_name, 3) FROM students; -- first 3 letters

SELECT RIGHT(full_name, 3) FROM students; -- last 3 letters

9. REVERSE()

Reverses the string.

Syntax:

REVERSE(string)

Example:

SELECT REVERSE('SQL');

📌 Output → LQS

Quick Table Summary 🧠

Function	Work
CONCAT()	join strings
LENGTH()	count characters
UPPER()	convert to capitals
LOWER()	convert to small letters
TRIM()	remove extra spaces
REPLACE()	replace text
SUBSTRING()	extract part of string
LEFT()/RIGHT()	take characters from side
REVERSE()	string reversed

⚡ Mathematical Functions in MySQL

Used to perform **calculations on numbers**.

---

1. ABS()

Returns **absolute value** (removes negative sign).

Syntax:

ABS(number)

Example:

SELECT ABS(-15); -- 15

SELECT ABS(marks - 70) FROM students;

📌 Used when you want **difference without minus**.

---

2. ROUND()

Rounds a number to a specified decimal position or nearest 10/100.

Syntax:

ROUND(number, decimal_places)

Examples:

SELECT ROUND(12.567, 2); -- 12.57

SELECT ROUND(256, -2); -- 300 (nearest hundred)

📌 -2 means nearest 100, -1 means nearest 10.

---

3. CEIL() / CEILING()

Rounds **up to the nearest whole number**.

Syntax:

CEIL(number)

Example:

```
SELECT CEIL(12.1);      -- 13
SELECT CEIL(7.9);       -- 8
```

📌 Always goes **up**, even if decimal is small.

---

4. FLOOR()

Rounds **down to nearest whole number**.

Syntax:

FLOOR(number)

Example:

```
SELECT FLOOR(12.9);     -- 12
SELECT FLOOR(7.1);      -- 7
```

📌 Always goes **down**.

---

5. POWER() / POW()

Used to calculate **number raised to power**.

Syntax:

POWER(number, power_value)

Example:

```
SELECT POWER(2, 3);     -- 8 (2³)
SELECT POWER(5, 2);     -- 25 (5²)
```

📌 Very useful in math/statistics formulas.

---

6. SQRT()

Returns **square root** of a number.

Syntax:

SQRT(number)

Example:

```
SELECT SQRT(64);       -- 8
SELECT SQRT(49);       -- 7
```

📌 Opposite of POWER with 2.

---

7. MOD()

Returns **remainder after division**.

Syntax:

MOD(number, divisor)

Example:

```
SELECT MOD(10, 3);     -- 1
SELECT MOD(25, 5);     -- 0
```

📌 Useful for:

- checking even/odd

```
SELECT MOD(7,2);  -- 1 -> odd
```

SELECT MOD(8,2); -- 0 -> even

---

8. GREATEST() & LEAST()

Find **largest** or **smallest** among multiple values.

Syntax:

GREATEST(val1, val2, val3,...)

LEAST(val1, val2, val3,...)

Example:

SELECT GREATEST(12, 9, 18); -- 18

SELECT LEAST(12, 9, 18); -- 9

---

9. RAND()

Returns a **random number between 0 and 1**.

Syntax:

RAND()

Example:

SELECT RAND();

📌 Useful for random student/product selection.

Quick Summary Table 😍

Function	Purpose
ABS()	absolute (no minus)
ROUND()	round to decimal/100/10
CEIL()	round **up**
FLOOR()	round **down**
POWER()/POW()	number^power
SQRT()	square root
MOD()	remainder
GREATEST()	highest value
LEAST()	lowest value
RAND()	random number

DATE & TIME FUNCTIONS in MySQL

Used to work with **dates, time, day, month, years**, difference between dates, formatting, etc.

---

1. CURDATE()

Returns **today's date (current date)**.

Syntax:

SELECT CURDATE();

📌 Output example → 2025-01-21

---

2. CURRENT_DATE()

Same as CURDATE()

SELECT CURRENT_DATE();

---

3. NOW()

Returns **current date + time** (timestamp).

SELECT NOW();

📌 Example → 2025-01-21 14:35:20

---

4. CURRENT_TIME()

Returns **current time only**.

SELECT CURRENT_TIME();

---

5. DAY(), MONTH(), YEAR()

Extracts **day / month / year** from a date.

SELECT DAY(date_column);

SELECT MONTH(date_column);

SELECT YEAR(date_column);

Example:

SELECT DAY('2024-12-25'); -- 25

SELECT MONTH('2024-12-25'); -- 12

SELECT YEAR('2024-12-25'); -- 2024

---

6. DATE()

Extracts only date from datetime.

SELECT DATE(NOW());

---

7. TIME()

Extracts time from datetime.

SELECT TIME(NOW());

---

8. DAYNAME()

Returns **day of week as name**.

SELECT DAYNAME('2024-02-10'); -- Saturday

---

9. MONTHNAME()

Returns **month name**.

SELECT MONTHNAME(sale_date);

📌 Output → January / February / etc.

---

10. WEEKDAY()

Returns weekday number (0 = Monday, 6 = Sunday)

SELECT WEEKDAY('2024-02-10'); -- 5

---

11. DAYOFWEEK()

Returns weekday number (1 = Sunday, 7 = Saturday)

SELECT DAYOFWEEK('2024-02-10'); -- 7

Used earlier to check **Weekend/Weekday**.

---

12. DATEDIFF()

Find **difference in days** between two dates.

SELECT DATEDIFF(CURDATE(), sale_date);

📌 Output → number of days passed

---

13. TIMESTAMPDIFF()

Difference in **YEAR / MONTH / DAY / HOUR** etc.

SELECT TIMESTAMPDIFF(YEAR, dob, CURDATE()) AS age;

Can also be used like:

TIMESTAMPDIFF(MONTH, date1, date2)

TIMESTAMPDIFF(DAY, date1, date2)

---

14. DATE_FORMAT()

Format date display style.

Syntax:

DATE_FORMAT(date, 'format')

Example:

SELECT DATE_FORMAT(sale_date, '%d-%b-%Y');

📌 Output → 12-Feb-2024

Format Codes:

Code	Output
%d	Day (01–31)
%m	Month number
%b	Short month name (Jan, Feb)
%M	Full month name
%y	Year (2 digit)
%Y	Year (4 digit)

---

15. ADDDATE() / DATE_ADD()

Add days/months/years to a date.

SELECT DATE_ADD('2024-01-10', INTERVAL 5 DAY); -- +5 days

SELECT DATE_ADD('2024-01-10', INTERVAL 2 MONTH); -- +2 months

---

16. SUBDATE() / DATE_SUB()

Subtract days/months/years.

SELECT DATE_SUB('2024-01-10', INTERVAL 10 DAY); -- -10 days

---

17. LAST_DAY()

Returns **last date of that month**.

SELECT LAST_DAY('2024-02-10'); -- 2024-02-29 (leap year)

---

18. MAKEDATE()

Make date using **year + day number**

SELECT MAKEDATE(2024, 32); -- 2024-02-01 (32nd day of year)

Quick Summary Table

Function	Use
CURDATE(), CURRENT_DATE	today's date
NOW()	date + time
DAY/MONTH/YEAR	extract parts
MONTHNAME(), DAYNAME()	name of month/day
WEEKDAY(), DAYOFWEEK()	day number
DATEDIFF()	days difference
TIMESTAMPDIFF()	diff in years/months/days
DATE_FORMAT()	date formatting
ADDDATE(), DATE_ADD()	add date interval
SUBDATE(), DATE_SUB()	subtract date interval
LAST_DAY()	month end
MAKEDATE()	generate date

ARITHMETIC FUNCTIONS / OPERATORS in SQL

Operator	Meaning	Example Result
+	Addition	10 + 5 = 15
-	Subtraction	10 - 5 = 5
*	Multiplication	10 * 5 = 50
/	Division	10 / 5 = 2
% or MOD()	Modulus (remainder)	10 % 3 = 1

Used mostly to calculate totals, differences, percentages etc.

---

1. Addition (+)

SELECT price + tax AS Total_Amount FROM sales;

Adds two numeric values.

---

2. Subtraction (-)

SELECT marks - 50 AS Marks_Above_50 FROM students;

Subtract values.

---

3. Multiplication (*)

SELECT price * quantity AS Total_Selling_Amount FROM sales;

Used commonly for **total cost** calculation.

---

4. Division (/)

SELECT price / 1000 AS Price_in_Thousands FROM sales;

Used when converting values (like ₹15000 → 15.0)

---

5. Modulus (%) or MOD()

Finds **remainder after division**.

```
SELECT MOD(10,3);    -- 1
SELECT 10 % 3;       -- 1
```

Useful to check **even / odd**:

```
SELECT
  CASE WHEN MOD(number,2)=0 THEN 'Even'
     ELSE 'Odd'
  END AS Type;
```

Common Practical Use Examples

◈ **Calculate Discount Price**

```
SELECT price - (price*10/100) AS Price_After_Discount FROM sales;
```

◈ **Increase salary by 20%**

```
SELECT salary + (salary*0.20) AS New_Salary FROM employees;
```

◈ **Convert Marks to Percentage**

```
SELECT (marks/100)*100 AS Percentage FROM students;
```

◈ **Revenue calculation**

```
SELECT price * quantity AS total_revenue FROM sales;
```

Quick Summary 🧠

Arithmetic	Usage
+	Add values
-	Subtract
*	Multiply
/	Divide
% or MOD()	Remainder

Functions like **ROUND, ABS, SQRT, CEIL, FLOOR, POWER** support arithmetic decisions.

📊 AGGREGATE FUNCTIONS (Very Important in SQL)

Aggregate functions are used to **perform calculations on multiple rows** and return **one result**.

Example use:

- Count total students
- Calculate total sales
- Find highest marks
- Find average salary

1️⃣ COUNT() — Counting values

Where it is used?

To know **how many rows**, **how many people**, **how many products** etc.

Syntax:

```
SELECT COUNT(column_name) FROM table_name;
```

If you want to count everything:
SELECT COUNT(*) FROM table_name;
Example:
SELECT COUNT(*) AS Total_Students FROM students;
Output: 10
Story style:
COUNT() is like counting heads in a classroom.

---

2 SUM() — Total Addition

Where used?
To get **total money**, **total marks**, **total revenue**, etc.
Syntax:
SELECT SUM(column_name) FROM table_name;
Example:
SELECT SUM(price) AS Total_Revenue FROM sales;
Output: 134000
Story style:
SUM() is like adding all shopping amounts from a bill.

---

3 AVG() — Average Value

Where used?
To know **average marks**, **average salary**, **average price** etc.
Syntax:
SELECT AVG(column_name) FROM table_name;
Example:
SELECT AVG(marks) AS Average_Marks FROM students;
Output: 76.50
Story style:
AVG() is like teacher calculating average class performance.

---

4 MAX() — Highest Value

Where used?
To find **top marks**, **highest price**, **max salary**, etc.
Syntax:
SELECT MAX(column_name) FROM table_name;
Example:
SELECT MAX(marks) AS Highest_Marks FROM students;
Output: 92
Story style:
MAX() shows who is topper or most expensive item.

---

5 MIN() — Lowest Value

Where used?
To get **lowest marks**, **minimum salary**, **cheapest item**, etc.
Syntax:
SELECT MIN(column_name) FROM table_name;

Example:

SELECT MIN(marks) AS Lowest_Marks FROM students;

Output: 55

Story style:

MIN() shows who got least or cheapest product.

🧠 **Important Concepts**

✓ **You do NOT need GROUP BY when you want only one result.**

SELECT SUM(price) FROM sales; -- total revenue

SELECT COUNT(*) FROM students; -- total students

✓ **GROUP BY gives separate results category-wise**

SELECT gender, AVG(marks)

FROM students

GROUP BY gender;

Output:

Female → 88

Male → 66

✓ **WHERE vs HAVING**

Uses before grouping	**Uses after grouping**
WHERE	**HAVING**
filters rows	filters groups
cannot use aggregate	used with aggregate

Example:

WHERE marks > 60 -- works before grouping

HAVING AVG(marks)>70 -- works after grouping

Quick Summary Table

Function	Returns
COUNT()	Total count of rows
SUM()	Total of values
AVG()	Average
MAX()	Highest value
MIN()	Lowest value

🎭 CASE WHEN in SQL

CASE works like an IF-ELSE ladder inside SQL.

It checks conditions and returns a value depending on what is true.

Think of CASE like:

If this happens → show this

else if something else → show that

otherwise → show default value

📌 **Syntax (Simple Format)**

CASE

```
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  ELSE default_result
END
```

We use it inside SELECT like a virtual column.

✓ Example 1 — Pass/Fail Based on Marks

```
SELECT full_name, marks,
CASE
   WHEN marks >= 60 THEN 'Pass'
   ELSE 'Fail'
END AS Result
FROM students;
```

full_name	marks	Result
Riya	75	Pass
Arjun	55	Fail

✓ Example 2 — Grade Category

```
SELECT full_name, marks,
CASE
   WHEN marks >= 90 THEN 'Excellent'
   WHEN marks >= 70 THEN 'Good'
   WHEN marks >= 50 THEN 'Average'
   ELSE 'Poor'
END AS Remarks
FROM students;
```

This works like a marks grading system.

✓ Example 3 — Weekend / Weekday

```
SELECT sale_date,
CASE
   WHEN DAYOFWEEK(sale_date) IN (1,7) THEN 'Weekend'
   ELSE 'Weekday'
END AS DayType
FROM sales;
```

It checks the day and returns message accordingly.

✓ Example 4 — Price Classification

```
SELECT product, price,
CASE
   WHEN price > 30000 THEN 'Expensive'
   ELSE 'Affordable'
END AS Category
FROM sales;
```

✓ Example 5 — Product Type using LIKE (Text Matching)

```
SELECT product,
CASE
   WHEN product LIKE '%Mob%' THEN 'Mobile'
   WHEN product LIKE '%Lap%' THEN 'Laptop'
   ELSE 'Others'
END AS Product_Type
FROM sales;
```

🤔 When do we use CASE?

Use-case	Example
Grading	Excellent/Good/Average
Status label	Pass/Fail
Price category	Costly/Cheap
Date based category	Weekend/Weekday
Pattern check	Mobile/Laptop/Others
Numeric ranges	age groups, discount levels

CASE is useful for **classifying, grouping, labeling, conditional reporting**.

Quick Summary for Memory

Keyword	Meaning
WHEN	condition
THEN	output if true
ELSE	output if none true
END	close case expression