















the differences between IF and CASE statements in MySQL using a table format. Here's a concise comparison:

| Aspect | IF | CASE |

|-------- |---- |----- |

| Purpose | Simple conditional logic | More complex conditional logic |

| Syntax | IF(condition, value\_if\_true, value\_if\_false) | CASE WHEN condition THEN result [WHEN ... THEN ...] [ELSE result] END |

| Number of conditions | Limited to one condition | Can handle multiple conditions |

| Readability | Good for simple conditions | Better for multiple conditions |

| Performance | Generally faster for simple conditions | May be slower due to multiple evaluations |

| Flexibility | Less flexible | More flexible, can use different types of conditions |

| Use case | Best for binary outcomes | Best for multiple possible outcomes |

# **Searching in SQL using LIKE**

### **Descriptions :**

The LIKE operator in SQL is used for pattern matching through simple text patterns. This operator is often used in a WHERE clause to search for a specified pattern in a column.

### **Syntax :**

SELECT column1, column2, ...

FROM table\_name

WHERE columnN LIKE pattern;

### **Patterns Search :**

//Exact match

SELECT \* FROM tableName WHERE column1 LIKE 'value';

//Starts with a pericular char / value

SELECT \* FROM tableName WHERE column1 LIKE 'abc%';

//Ends with a perticular char

SELECT \* FROM tableName WHERE column1 LIKE '%abc';

//Starts with and end with a char

SELECT \* FROM tableName WHERE column1 LIKE 'abc%abc';

//Char is present at specfic index

SELECT \* FROM tableName WHERE column1 LIKE '\_a%';

//If anywhere the pattern is present

SELECT \* FROM tableName WHERE column1 LIKE '%abc%';

1. Match any single character:

```sql

SELECT \* FROM products WHERE product\_name LIKE 'a\_c';

```

Sample data:

- abc

- adc

- a2c

- ac

- abbc

Output:

- abc

- adc

- a2c

2. Match a range of characters:

```sql

SELECT \* FROM products WHERE product\_name LIKE '[a-d]%';

```

Sample data:

- apple

- banana

- cherry

- date

- elderberry

Output:

- apple

- banana

- cherry

- date

3. Match characters not in a range:

```sql

SELECT \* FROM products WHERE product\_name LIKE '[^a-d]%';

```

Sample data:

- apple

- elderberry

- fig

- grape

- banana

Output:

- elderberry

- fig

- grape

4. Match multiple possible characters:

```sql

SELECT \* FROM products WHERE product\_name LIKE '[aeiou]%';

```

Sample data:

- apple

- orange

- banana

- elderberry

- ice cream

Output:

- apple

- orange

- ice cream

5. Match a pattern multiple times:

```sql

SELECT \* FROM products WHERE product\_name LIKE '%[0-9]%[0-9]%';

```

Sample data:

- product1

- item23

- thing456

- stuff7

- object89

Output:

- item23

- thing456

- object89

6. Combine multiple conditions:

```sql

SELECT \* FROM products WHERE product\_name LIKE 'a%' AND product\_name LIKE '%z';

```

Sample data:

- apple juice

- amazing

- buzz

- aztec

Output:

- amazing

- aztec

7. Use wildcards in the middle:

```sql

SELECT \* FROM products WHERE product\_name LIKE 'a%b%c';

```

Sample data:

- abc

- abbc

- acbc

- axyzbmnc

- abcde

Output:

- abc

- abbc

- axyzbmnc

8. Match a specific number of characters:

```sql

SELECT \* FROM products WHERE product\_name LIKE '\_\_\_';

```

Sample data:

- cat

- dog

- mouse

- elephant

- ant

Output:

- cat

- dog

- ant

9. Combine LIKE with other SQL clauses:

```sql

SELECT \* FROM products WHERE product\_name LIKE '%2023%' ORDER BY product\_name DESC;

```

Sample data:

- Model 2023X

- 2023 Edition

- Version 2023

- 2022 Model

- New 2023 Line

Output:

- Version 2023

- New 2023 Line

- Model 2023X

- 2023 Edition

10. Use ESCAPE to match special characters:

```sql

SELECT \* FROM products WHERE product\_name LIKE '10\%%' ESCAPE '\';

```

Sample data:

- 10% off

- 10% discount

- 10 percent

- 100% pure

Output:

- 10% off

- 10% discount

-- Assume we have a table 'words' with a column 'word'

-- 1. Match exactly 3 characters

SELECT word FROM words WHERE word LIKE '\_\_\_';

-- Output: cat, dog, rat, bat

-- 2. Match 4-letter words ending with 'at'

SELECT word FROM words WHERE word LIKE '\_\_\_at';

-- Output: chat, flat

-- 3. Match 5-letter words starting with 'a'

SELECT word FROM words WHERE word LIKE 'a\_\_\_\_';

-- Output: apple, arrow

-- 4. Match words with 'o' as the second letter

SELECT word FROM words WHERE word LIKE '\_o%';

-- Output: dog, cook, movie

-- 5. Match 6-letter words with 'e' as the third letter

SELECT word FROM words WHERE word LIKE '\_\_e\_\_\_';

-- Output: cheese, beaver

-- 6. Match words starting and ending with 'a', 5 letters long

SELECT word FROM words WHERE word LIKE 'a\_\_\_a';

-- Output: alpha, arena

-- 7. Match 4-letter words with 'a' as the second letter and 't' as the last

SELECT word FROM words WHERE word LIKE '\_a\_t';

-- Output: fact, part

-- 8. Match words with exactly two characters between 'b' and 'r'

SELECT word FROM words WHERE word LIKE 'b\_\_r%';

-- Output: bear, beer, board

-- 9. Match 7-letter words ending with 'ing'

SELECT word FROM words WHERE word LIKE '\_\_\_\_ing';

-- Output: running, jumping

-- 10. Match words starting with 'pre' and having exactly 6 letters

SELECT word FROM words WHERE word LIKE 'pre\_\_\_';

-- Output: prefix, preset

-- 11. Match 5-letter palindromes (same forward and backward)

SELECT word FROM words WHERE word LIKE '\_a\_a\_';

-- Output: radar, kayak

-- 12. Match words with 'q' as the second-to-last letter

SELECT word FROM words WHERE word LIKE '%q\_';

-- Output: equal, liquid

-- 13. Match 6-letter words with vowels in 2nd and 5th positions

SELECT word FROM words WHERE word LIKE '\_[aeiou]\_\_[aeiou]\_';

-- Output: people, reason

-- 14. Match words starting and ending with the same letter, 5 letters long

SELECT word FROM words WHERE word LIKE '\_\_\_\_' AND LEFT(word, 1) = RIGHT(word, 1);

-- Output: level, rotor

-- 15. Match 7-letter words with double letters in the middle

SELECT word FROM words WHERE word LIKE '\_\_\_' AND SUBSTRING(word, 4, 1) = SUBSTRING(word, 5, 1);

-- Output: bookkeep, lessness

-- 16. Match words with exactly three vowels

SELECT word FROM words WHERE

(LENGTH(word) - LENGTH(REPLACE(LOWER(word), 'a', ''))) +

(LENGTH(word) - LENGTH(REPLACE(LOWER(word), 'e', ''))) +

(LENGTH(word) - LENGTH(REPLACE(LOWER(word), 'i', ''))) +

(LENGTH(word) - LENGTH(REPLACE(LOWER(word), 'o', ''))) +

(LENGTH(word) - LENGTH(REPLACE(LOWER(word), 'u', ''))) = 3;

-- Output: beautiful, audience

-- 17. Match words where the first and last two letters are the same

SELECT word FROM words WHERE LEFT(word, 2) = RIGHT(word, 2);

-- Output: murmur, cancan

-- 18. Match 8-letter words with 's' in the middle

SELECT word FROM words WHERE word LIKE '\_\_\_s\_\_\_\_';

-- Output: disaster, blossom

-- 19. Match words that start and end with a consonant, 6 letters long

SELECT word FROM words WHERE word LIKE '[^aeiou]\_\_\_[^aeiou]' AND LENGTH(word) = 6;

-- Output: stupid, rhythm

-- 20. Match words with alternating consonants and vowels, 6 letters long

SELECT word FROM words WHERE word RLIKE '^[^aeiou][aeiou][^aeiou][aeiou][^aeiou][aeiou]$';

-- Output: banana, kotoko

### **Examples :**

#### **Give all the mobile where Ram is 6 gb**

SELECT \* FROM flipkart\_mobiles WHERE Memory LIKE "6%";

#### **Give all the list of Mobile Where Color Contains Black**

SELECT \* FROM flipkart\_mobiles WHERE Color LIKE "%Black%";

#### **Give all the mobile where Model ends with "3"**

SELECT \* FROM flipkart\_mobiles WHERE Model LIKE "%3";

#### **Give all the mobile where second word in Brand name contains "Q" in it**

SELECT \* FROM flipkart\_mobiles WHERE Brand LIKE "\_Q%";

# **Aggregate Function in SQL**

### **Description :**

Aggregate functions perform a calculation on a set of values and return a single value. They are used extensively in SQL to summarize data.

### **Types :**

#### **SUM**

* Description: Calculates the total sum of a numeric column.
* SQL Syntax: SELECT SUM(column\_name) FROM table\_name;
* Example: To calculate the total salary of all employees: SELECT SUM(salary) FROM employees;

#### **AVG**

* Description: Calculates the average value of a numeric column.
* SQL Syntax: SELECT AVG(column\_name) FROM table\_name;
* Example: To find the average salary: SELECT AVG(salary) FROM employees;

#### **COUNT**

* Description: Returns the number of rows that matches a specified criterion.
* SQL Syntax: SELECT COUNT(column\_name) FROM table\_name;
* Example: To count the number of employees: SELECT COUNT(employee\_id) FROM employees;

#### **MIN and MAX**

* Description: Returns the minimum or maximum value in a set.
* SQL Syntax: SELECT MIN(column\_name) FROM table\_name; or SELECT MAX(column\_name) FROM table\_name;
* Example: To find the lowest and highest salary: SELECT MIN(salary), MAX(salary) FROM employees;

# **String Function in SQL**

### **Description :**

String functions in SQL are built-in functions that perform operations on string data types. These functions can manipulate character strings, perform operations based on the content of a string, convert formats, and much more.

### **Types :**

#### **UPPER & LOWER: Converts a string to uppercase or lowercase.**

* SQL Syntax: SELECT UPPER(column\_name), LOWER(column\_name) FROM table\_name;
* Example: Convert employee names to uppercase: SELECT UPPER(name) FROM employees;

#### **LENGTH: Returns the length of a string.**

* SQL Syntax: SELECT LENGTH(column\_name) FROM table\_name;
* Example: Find the length of each employee's name: SELECT LENGTH(name) FROM employees;

#### **SUBSTRING: Extracts a substring from a string.**

* SQL Syntax: SELECT SUBSTRING(column\_name, start, length) FROM table\_name;
* Example: Extract the first 5 characters of each employee's name: SELECT SUBSTRING(name, 1, 5) FROM employees;

#### **CONCAT: Concatenates two or more strings.**

* SQL Syntax: SELECT CONCAT(string1, string2, ...) FROM table\_name;
* Example: Combine first and last names: SELECT CONCAT(first\_name, ' ', last\_name) FROM employees;

#### **REPLACE: Replaces occurrences of a specified string.**

* SQL Syntax: SELECT REPLACE(column\_name, 'old\_string', 'new\_string') FROM table\_name;
* Example: Replace "Inc" with "LLC" in a company name column: SELECT REPLACE(company\_name, 'Inc', 'LLC') FROM companies;

#### **TRIM: Removes leading and trailing spaces from a string.**

* SQL Syntax: SELECT TRIM(column\_name) FROM table\_name;
* Example: Trim spaces from employee names: SELECT TRIM(name) FROM employees;

#### **LEFT , RIGHT : Extracts a given number of characters from the left or right side of a string.**

SELECT LEFT('Hello World', 5);

SELECT RIGHT('Hello World', 5);

# **Numeric Function in SQL**

Numeric functions in SQL are built-in functions designed to perform operations on numeric data types. These functions can execute mathematical calculations, format numbers, and perform various other operations related to numerical data in SQL queries.

### **Types :**

#### **ABS: Returns the absolute value.**

* SQL Syntax: SELECT ABS(column\_name) FROM table\_name;
* Example: Find the absolute value of balance changes: SELECT ABS(balance\_change) FROM accounts;

#### **POWER, SQRT: Calculates the power of a number, or its square root.**

* SQL Syntax: SELECT POWER(column\_name, exponent), SQRT(column\_name) FROM table\_name;
* Example: Calculate the square and square root of a number: SELECT POWER(value, 2), SQRT(value) FROM numbers;

#### **CEIL, FLOOR: Rounds a number up to the nearest integer, or down to the nearest integer.**

* SQL Syntax: SELECT CEIL(column\_name), FLOOR(column\_name) FROM table\_name;
* Example: Round up and down employee salaries: SELECT CEIL(salary), FLOOR(salary) FROM employees;

#### **ROUND: Rounds a number to a specified number of decimal places.**

* SQL Syntax: SELECT ROUND(column\_name, decimals) FROM table\_name;
* Example: Round a product's price: SELECT ROUND(price, 2) FROM products;

#### **MOD: Returns the remainder of a division.**

* SQL Syntax: SELECT MOD(numerator, denominator) FROM table\_name;
* Example: Find the remainder of employee ID divided by 5: SELECT MOD(employee\_id, 5) FROM employees;

#### **EXP, SIGN, RAND: Calculates the exponential of a number, its sign, or a random number.**

* SQL Syntax: SELECT EXP(column\_name), SIGN(column\_name), RAND() FROM table\_name;
* Example: Generate random numbers for lottery: SELECT RAND() FROM dual;

#### **POWER**

Raises a number to the power of another number.

SELECT POWER(2, 3);

# **Date Function in SQL**

### **Descriptions :**

Date functions in SQL are built-in functions that allow you to manipulate and extract information from date and time values. These functions are crucial for performing operations such as calculating differences between dates, formatting dates into a specific style, extracting parts of a date (like the day, month, or year), and much more

### **Examples :**

#### **NOW**

SELECT NOW() AS current\_datetime;

-- Result: 'YYYY-MM-DD HH:MM:SS'

#### **CURDATE()**

SELECT CURDATE();

SELECT CURDATE() AS current\_date;

-- Result: 'YYYY-MM-DD'

#### **CURTIME()**

SELECT CURTIME() AS current\_time;

-- Result: 'HH:MM:SS'

#### **YEAR : Extract year from date**

SELECT YEAR('2023-09-09') AS extracted\_year;

-- Result: 2023

#### **MONTH : Extract month from date**

SELECT MONTH('2023-09-09') AS extracted\_month;

-- Result: 9

#### **DAY : Extract day from date**

SELECT DAY('2023-09-09') AS extracted\_day;

-- Result: 9

#### **Date addition**

SELECT DATE\_ADD('2023-09-09', INTERVAL 1 DAY) AS added\_date;

-- Result: '2023-09-10'

#### **Date subtraction**

SELECT DATE\_SUB('2023-09-09', INTERVAL 1 MONTH) AS subtracted\_date;

-- Result: '2023-08-09'

#### **Date difference in days**

SELECT DATEDIFF('2023-09-20', '2023-09-10') AS date\_difference;

-- Result: 10

#### **Date formatting**

SELECT DATE\_FORMAT('2023-09-09', '%Y/%m/%d') AS formatted\_date;

-- Result: '2023/09/09'

#### **Current week number**

SELECT WEEK(NOW()) AS current\_week\_number;

-- Result: Current week number

#### **Current day of the week (0 = Sunday, 1 = Monday, ..., 6 = Saturday)**

SELECT DAYOFWEEK(NOW()) AS current\_day\_of\_week;

-- Result: Current day of the week

#### **Number of days in a month**

SELECT DAY(LAST\_DAY('2023-09-09')) AS days\_in\_month;

-- Result: Number of days in September 2023

#### **Extract : MONTH , YEAR , DAY , QUARTER , WEEK , HOUR , MINUTE**

* Retrieves a specific part from a date or time value.
* SQL Syntax: SELECT EXTRACT(YEAR FROM column\_name) FROM table\_name;
* Example: To extract the year from a date of birth column: SELECT EXTRACT(YEAR FROM dob) FROM employees;

## **Instructor Activity | Code Implementation | Examples**

* Explain about Date Function and show them practical examples in Workbench.

## **Student Activity**

* How can you add and substract date using Date Functions

# **System defined vs User Defined Functions**

### **System Defined Functions:**

* System-defined functions are built-in functions provided by the SQL database management system (DBMS).
* These functions are pre-defined and cover a wide range of functionalities, including string manipulation, numeric calculations, date and time operations, and more.
* Functions provided by the SQL database management system, such as NOW(), UPPER(), SUM(), etc.

### **User Defined Functions :**

* User-defined functions are functions that you can create in the database according to your specific requirements.
* UDFs allow encapsulating complex operations into a single function that can then be reused in multiple SQL queries, enhancing modularity, readability, and maintainability of the database and its associated applications.

# **Conditional Statements in SQL**

### **Description :**

Conditional statements in SQL are used to execute different SQL actions based on specific conditions. These statements enhance the control flow in SQL programming, allowing for more dynamic and flexible data manipulation and querying.

### **Using IF**

#### **Syntax :**

SELECT IF( expression , if true , if false) AS result

FROM tableName;

#### **Example :**

SELECT name, IF(age >= 18, 'Adult', 'Minor') AS age\_group

FROM users;

10 different `IF()` function examples with the correct table name `richperson`.

**### Example Queries with `IF()`**

1. **\*\*Wealth Status\*\***

```sql

SELECT

full\_name,

IF(wealth > 1000000, 'Billionaire', 'Not Billionaire') AS wealth\_status

FROM

richperson;

```

2. **\*\*Age Group\*\***

```sql

SELECT

full\_name,

IF(age < 18, 'Minor', IF(age <= 65, 'Adult', 'Senior')) AS age\_group

FROM

richperson;

```

3. **\*\*Country of Residence\*\***

```sql

SELECT

full\_name,

IF(country\_of\_residence = 'USA', 'United States', 'Other Country') AS country\_name

FROM

richperson;

```

4. **\*\*Gender Description\*\***

```sql

SELECT

full\_name,

IF(gender = 'M', 'Male', 'Female') AS gender\_description

FROM

richperson;

```

5. **\*\*Industry Type\*\***

```sql

SELECT

full\_name,

IF(industry = 'Technology', 'Tech', 'Other') AS industry\_type

FROM

richperson;

```

6. **\*\*Life Expectancy Level\*\***

```sql

SELECT

full\_name,

IF(life\_expectancy > 80, 'High', IF(life\_expectancy >= 70, 'Moderate', 'Low')) AS life\_expectancy\_level

FROM

richperson;

```

7. **\*\*Tax Bracket\*\***

```sql

SELECT

full\_name,

IF(tax\_rate > 30, 'High Tax', IF(tax\_rate >= 15, 'Moderate Tax', 'Low Tax')) AS tax\_bracket

FROM

richperson;

```

8. **\*\*GDP Status\*\***

```sql

SELECT

full\_name,

IF(gdp\_country > 500000, 'High GDP', IF(gdp\_country >= 100000, 'Moderate GDP', 'Low GDP')) AS gdp\_status

FROM

richperson;

```

9. **\*\*Primary Education Enrollment Status\*\***

```sql

SELECT

full\_name,

IF(g\_primary\_ed\_enroll > 90, 'High Enrollment', IF(g\_primary\_ed\_enroll >= 50, 'Moderate Enrollment', 'Low Enrollment')) AS primary\_ed\_enrollment\_status

FROM

richperson;

```

10. **\*\*Consumer Price Index (CPI) Status\*\***

```sql

SELECT

full\_name,

IF(cpi\_country > 50, 'High CPI', IF(cpi\_country >= 20, 'Moderate CPI', 'Low CPI')) AS cpi\_status

FROM

richperson;

```

**### Explanation**

- Each query demonstrates the use of the `IF()` function to apply different conditions and return corresponding results.

- These queries will evaluate the conditions specified and return the appropriate label or value based on the data in your `richperson` table.

### **Using CASE**

#### **Syntax :**

CASE expression

WHEN value1 THEN result1

WHEN value2 THEN result2

...

ELSE default\_result

END

#### **Example :**

Type 1

SELECT name,

CASE

WHEN age < 18 THEN 'Minor'

WHEN age >= 18 AND age < 65 THEN 'Adult'

ELSE 'Senior'

END AS age\_group

FROM users;

Type 2

SELECT OrderID,

CASE Status

WHEN 'P' THEN 'Pending'

WHEN 'O' THEN 'Ongoing'

WHEN 'C' THEN 'Completed'

ELSE 'Unknown'

END AS OrderStatus

FROM Orders;

**10 different examples using the `CASE` statement with your table named `richperson`. Each example will demonstrate a different use case for the `CASE` statement.**

**### Example Queries with `CASE`**

**1. \*\*Wealth Status\*\***

**```sql**

**SELECT**

**full\_name,**

**wealth,**

**CASE**

**WHEN wealth > 1000000 THEN 'Billionaire'**

**WHEN wealth > 100000 THEN 'Multi-Millionaire'**

**WHEN wealth > 10000 THEN 'Millionaire'**

**ELSE 'Not a Millionaire'**

**END AS wealth\_status**

**FROM**

**richperson;**

**```**

**2. \*\*Age Group\*\***

**```sql**

**SELECT**

**full\_name,**

**age,**

**CASE**

**WHEN age < 18 THEN 'Minor'**

**WHEN age BETWEEN 18 AND 65 THEN 'Adult'**

**ELSE 'Senior'**

**END AS age\_group**

**FROM**

**richperson;**

**```**

**3. \*\*Country of Residence\*\***

**```sql**

**SELECT**

**full\_name,**

**country\_of\_residence,**

**CASE**

**WHEN country\_of\_residence = 'USA' THEN 'United States'**

**WHEN country\_of\_residence = 'CAN' THEN 'Canada'**

**WHEN country\_of\_residence = 'GBR' THEN 'United Kingdom'**

**ELSE 'Other'**

**END AS country\_name**

**FROM**

**richperson;**

**```**

**4. \*\*Gender Description\*\***

**```sql**

**SELECT**

**full\_name,**

**gender,**

**CASE**

**WHEN gender = 'M' THEN 'Male'**

**WHEN gender = 'F' THEN 'Female'**

**ELSE 'Other'**

**END AS gender\_description**

**FROM**

**richperson;**

**```**

**5. \*\*Industry Type\*\***

**```sql**

**SELECT**

**full\_name,**

**industry,**

**CASE**

**WHEN industry = 'Technology' THEN 'Tech'**

**WHEN industry = 'Finance & Investments' THEN 'Finance'**

**WHEN industry = 'Fashion & Retail' THEN 'Retail'**

**ELSE 'Other'**

**END AS industry\_type**

**FROM**

**richperson;**

**```**

**6. \*\*Life Expectancy Level\*\***

**```sql**

**SELECT**

**full\_name,**

**life\_expectancy,**

**CASE**

**WHEN life\_expectancy > 80 THEN 'High'**

**WHEN life\_expectancy BETWEEN 70 AND 80 THEN 'Moderate'**

**ELSE 'Low'**

**END AS life\_expectancy\_level**

**FROM**

**richperson;**

**```**

**7. \*\*Tax Bracket\*\***

**```sql**

**SELECT**

**full\_name,**

**tax\_rate,**

**CASE**

**WHEN tax\_rate > 30 THEN 'High Tax'**

**WHEN tax\_rate BETWEEN 15 AND 30 THEN 'Moderate Tax'**

**ELSE 'Low Tax'**

**END AS tax\_bracket**

**FROM**

**richperson;**

**```**

**8. \*\*GDP Status\*\***

**```sql**

**SELECT**

**full\_name,**

**gdp\_country,**

**CASE**

**WHEN gdp\_country > 5000000000000 THEN 'High GDP'**

**WHEN gdp\_country BETWEEN 1000000000000 AND 5000000000000 THEN 'Moderate GDP'**

**ELSE 'Low GDP'**

**END AS gdp\_status**

**FROM**

**richperson;**

**```**

**9. \*\*Primary Education Enrollment Status\*\***

**```sql**

**SELECT**

**full\_name,**

**g\_primary\_ed\_enroll,**

**CASE**

**WHEN g\_primary\_ed\_enroll > 90 THEN 'High Enrollment'**

**WHEN g\_primary\_ed\_enroll BETWEEN 50 AND 90 THEN 'Moderate Enrollment'**

**ELSE 'Low Enrollment'**

**END AS primary\_ed\_enrollment\_status**

**FROM**

**richperson;**

**```**

**10. \*\*Consumer Price Index (CPI) Status\*\***

**```sql**

**SELECT**

**full\_name,**

**cpi\_country,**

**CASE**

**WHEN cpi\_country > 50 THEN 'High CPI'**

**WHEN cpi\_country BETWEEN 20 AND 50 THEN 'Moderate CPI'**

**ELSE 'Low CPI'**

**END AS cpi\_status**

**FROM**

**richperson;**

**```**

**### Explanation**

**- \*\*Wealth Status\*\*: Categorizes individuals based on their wealth.**

**- \*\*Age Group\*\*: Classifies individuals into age groups.**

**- \*\*Country of Residence\*\*: Maps country codes to readable country names.**

**- \*\*Gender Description\*\*: Describes gender based on the value.**

**- \*\*Industry Type\*\*: Simplifies industry categories.**

**- \*\*Life Expectancy Level\*\*: Categorizes life expectancy into high, moderate, and low.**

**- \*\*Tax Bracket\*\*: Determines the tax rate category.**

**- \*\*GDP Status\*\*: Categorizes countries based on their GDP.**

**- \*\*Primary Education Enrollment Status\*\*: Evaluates the primary education enrollment percentage.**

**- \*\*Consumer Price Index (CPI) Status\*\*: Categorizes countries based on their Consumer Price Index (CPI).**

**These queries can be run individually to see the results based on your `richperson` table's data. Adjust the conditions as needed to fit your specific requirements.**

**SELECT**

**full\_name,**

**wealth,**

**age,**

**country\_of\_residence,**

**industry,**

**CASE**

**WHEN wealth > 1000000 THEN 'Billionaire'**

**WHEN wealth > 100000 THEN 'Multi-Millionaire'**

**WHEN wealth > 10000 THEN 'Millionaire'**

**ELSE 'Not a Millionaire'**

**END AS wealth\_status,**

**CASE**

**WHEN age < 18 THEN 'Minor'**

**WHEN age BETWEEN 18 AND 65 THEN 'Adult'**

**WHEN age > 65 THEN 'Senior'**

**ELSE 'Unknown'**

**END AS age\_group,**

**CASE**

**WHEN country\_of\_residence = 'USA' THEN 'United States'**

**WHEN country\_of\_residence = 'CAN' THEN 'Canada'**

**WHEN country\_of\_residence = 'GBR' THEN 'United Kingdom'**

**ELSE 'Other'**

**END AS country\_name,**

**CASE**

**WHEN gender = 'M' THEN 'Male'**

**WHEN gender = 'F' THEN 'Female'**

**ELSE 'Other'**

**END AS gender\_description,**

**CASE**

**WHEN industry = 'Technology' THEN 'Tech'**

**WHEN industry = 'Finance' THEN 'Finance'**

**WHEN industry = 'Retail' THEN 'Retail'**

**ELSE 'Other'**

**END AS industry\_type,**

**CASE**

**WHEN life\_expectancy > 80 THEN 'High'**

**WHEN life\_expectancy BETWEEN 70 AND 80 THEN 'Moderate'**

**ELSE 'Low'**

**END AS life\_expectancy\_level,**

**CASE**

**WHEN tax\_rate > 30 THEN 'High Tax'**

**WHEN tax\_rate BETWEEN 15 AND 30 THEN 'Moderate Tax'**

**ELSE 'Low Tax'**

**END AS tax\_bracket,**

**CASE**

**WHEN gdp\_country > 500000 THEN 'High GDP'**

**WHEN gdp\_country BETWEEN 100000 AND 500000 THEN 'Moderate GDP'**

**ELSE 'Low GDP'**

**END AS gdp\_status,**

**CASE**

**WHEN g\_primary\_ed\_enroll > 90 THEN 'High Enrollment'**

**WHEN g\_primary\_ed\_enroll BETWEEN 50 AND 90 THEN 'Moderate Enrollment'**

**ELSE 'Low Enrollment'**

**END AS primary\_ed\_enrollment\_status,**

**CASE**

**WHEN cpi\_country > 50 THEN 'High CPI'**

**WHEN cpi\_country BETWEEN 20 AND 50 THEN 'Moderate CPI'**

**ELSE 'Low CPI'**

**END AS cpi\_status**

**FROM**

**your\_table\_name;**

## **Resources - Official Documentation and Other Resources**

* SQL LIKE - <https://www.w3schools.com/sql/sql_like.asp>
* Aggregate Functions in SQL - <https://www.w3schools.com/sql/sql_aggregate_functions.asp>
* String Functions in SQL - <https://dev.mysql.com/doc/refman/8.3/en/string-functions.html>
* Numeric Functions in SQL - <https://dev.mysql.com/doc/refman/8.3/en/numeric-functions.html>
* Date Functions in SQL - <https://www.w3schools.com/sql/sql_dates.asp>
* Case in SQL - <https://www.w3schools.com/sql/sql_case.asp>

USE DB101;

-- Query all records from the emp table

SELECT \* FROM emp;

-- Query all records from the flipkart\_mobiles table

SELECT \* FROM flipkart\_mobiles;

-- Query all records from the netflix table

SELECT \* FROM netflix;

-- Give the first 10 rows from the emp table

SELECT \* FROM emp LIMIT 10;

-- Give the first 25 rows from the emp table

SELECT \* FROM emp LIMIT 25;

-- Skip the first 20 rows and return the next 10 rows from the emp table, ordered by salary in descending order

SELECT \* FROM emp ORDER BY salary DESC LIMIT 10 OFFSET 20;

-- Skip the first 20 rows and return the next 10 rows from the emp table, ordered by salary in ascending order

SELECT \* FROM emp ORDER BY salary ASC LIMIT 10 OFFSET 20;

-- Skip the first 20 rows and return the next 10 rows from the emp table, ordered by salary (default is ascending)

SELECT \* FROM emp ORDER BY salary LIMIT 10 OFFSET 20;

-- Query all records from the flipkart\_mobiles table

SELECT \* FROM flipkart\_mobiles;

-- Query all distinct brands from the flipkart\_mobiles table

SELECT DISTINCT brand FROM flipkart\_mobiles;

-- Query all distinct brands and colors from the flipkart\_mobiles table

SELECT DISTINCT brand, color FROM flipkart\_mobiles;

-- Query all records from the netflix table

SELECT \* FROM netflix;

-- Query all unique directors who produced movies in Netflix

SELECT DISTINCT director FROM netflix;

-- Query all unique directors who produced movies in Netflix and belong to India

SELECT DISTINCT director FROM netflix WHERE country = 'India';

USE b35\_db101;

-- Query all records from the flipkart\_mobiles table

SELECT \* FROM flipkart\_mobiles;

-- Give all the mobiles where memory starts with 6 (e.g., 6GB RAM)

SELECT \* FROM flipkart\_mobiles WHERE memory LIKE '6%';

-- Give all the mobiles where color starts with 'M'

SELECT \* FROM flipkart\_mobiles WHERE color LIKE 'M%';

-- Give all the mobiles where color ends with the letter 'd'

SELECT \* FROM flipkart\_mobiles WHERE color LIKE '%d';

-- Give all the mobiles where the model ends with '3'

SELECT \* FROM flipkart\_mobiles WHERE model LIKE '%3';

-- Give all the mobiles where color contains 'White'

SELECT \* FROM flipkart\_mobiles WHERE color LIKE '%White%';

-- Give all the mobiles where color contains 'Black'

SELECT \* FROM flipkart\_mobiles WHERE color LIKE '%Black%';

-- Give all the mobiles where the second character in the brand name is 'N'

SELECT \* FROM flipkart\_mobiles WHERE brand LIKE '\_N%';

USE DB101;

-- Query all records from the emp table

SELECT \* FROM emp;

-- What is the maximum salary of the employees

SELECT MAX(salary) AS max\_salary FROM emp;

-- Query all records from the netflix table

SELECT \* FROM netflix;

-- What is the latest movie added to Netflix

SELECT MAX(date\_added) AS latest\_movie\_date FROM netflix;

-- What is the minimum salary of the employees

SELECT MIN(salary) AS min\_salary FROM emp;

-- What is the total salary of all employees

SELECT SUM(salary) AS total\_salary FROM emp;

-- What is the average salary of the employees

SELECT AVG(salary) AS avg\_salary FROM emp;

-- How many employees are there?

SELECT COUNT(\*) AS total\_employees FROM emp;

-- How many unique employee names are there?

SELECT COUNT(DISTINCT name) AS unique\_employee\_names FROM emp;

-- Find the employee(s) with the maximum salary using a subquery

SELECT \* FROM emp WHERE salary = (SELECT MAX(salary) FROM emp);

-- Count all the employees where gender is female

SELECT COUNT(\*) AS female\_employees FROM emp WHERE gender = 'Female';

-- Count all the employees where gender is male and salary is greater than 50,000

SELECT COUNT(\*) AS high\_earning\_male\_employees FROM emp WHERE gender = 'Male' AND salary > 50000;

-- Find the average salary of male employees

SELECT AVG(salary) AS avg\_male\_salary FROM emp WHERE gender = 'Male';

-- Find the total salary and count of female employees

SELECT SUM(salary) AS total\_female\_salary, COUNT(\*) AS female\_employee\_count FROM emp WHERE gender = 'Female';

-- Intro to subquery example

-- First, find the maximum salary

-- Second, compare this salary in the main query

-- Find the employee(s) with the maximum salary using a subquery

SELECT \* FROM emp WHERE salary = (SELECT MAX(salary) FROM emp);

-- Get current date and time

SELECT NOW(); -- Current date and time

-- Get current date

SELECT CURDATE(); -- Current date

-- Get current time

SELECT CURTIME(); -- Current time

-- Query all records from the richperson table

SELECT \* FROM richperson;

-- Give full\_name and year of birth for all rich persons

SELECT full\_name, YEAR(birth\_date) FROM richperson;

-- Give full\_name and month of birth for all rich persons

SELECT full\_name, MONTH(birth\_date) FROM richperson;

SELECT full\_name, MONTHNAME(birth\_date) FROM richperson;

-- Give all the rich persons who were born in January

SELECT full\_name FROM richperson WHERE MONTH(birth\_date) = 1;

-- Give all the rich person names and the day and month they were born

SELECT full\_name, DAY(birth\_date) AS day\_of\_birth, MONTH(birth\_date) AS month\_of\_birth FROM richperson;

-- Date operations

-- 1. Date Addition -> DATE\_ADD("YYYY-MM-DD", INTERVAL no MONTH/DAY/YEAR)

SELECT birth\_date, DATE\_ADD(birth\_date, INTERVAL 1 MONTH) AS new\_date FROM richperson;

-- 2. Date Subtraction -> DATE\_SUB("YYYY-MM-DD", INTERVAL no MONTH/DAY/YEAR)

SELECT birth\_date, DATE\_SUB(birth\_date, INTERVAL 1 MONTH) AS new\_date FROM richperson;

-- 3. Date Difference -> DATEDIFF(date1, date2)

SELECT birth\_date, DATEDIFF(NOW(), birth\_date) AS days\_difference FROM richperson;

-- 4. Timestamp Difference -> TIMESTAMPDIFF(unit, datetime\_expr1, datetime\_expr2)

SELECT birth\_date, TIMESTAMPDIFF(YEAR, birth\_date, NOW()) AS years\_difference FROM richperson;

USE DB101;

-- String Functions: LOWER, UPPER, LENGTH, SUBSTRING, CONCAT, REPLACE, TRIM, LEFT, RIGHT

-- Data Types: varchar, text, int, bigint

-- Query all records from the flipkart\_mobiles table

SELECT \* FROM flipkart\_mobiles;

-- Round the ratings to the nearest integer

SELECT id, ROUND(rating) AS rounded\_rating, rating FROM flipkart\_mobiles;

-- Get the floor value (rounding down) of the ratings

SELECT id, FLOOR(rating) AS floor\_rating, rating FROM flipkart\_mobiles;

-- Get the ceiling value (rounding up) of the ratings

SELECT id, CEIL(rating) AS ceil\_rating, rating FROM flipkart\_mobiles;

-- Calculate the power of 2 for the ratings

SELECT id, POWER(rating, 2) AS squared\_rating, rating FROM flipkart\_mobiles;

-- Query the Brand column from the flipkart\_mobiles table

SELECT Brand FROM flipkart\_mobiles;

-- Convert all mobile brand names to lowercase

SELECT LOWER(Brand) AS lower\_brand FROM flipkart\_mobiles;

-- Convert all model names to uppercase

SELECT UPPER(model) AS upper\_model FROM flipkart\_mobiles;

-- Find the length of each model name

SELECT model, LENGTH(model) AS model\_length FROM flipkart\_mobiles;

-- Get all the products from Apple (case-insensitive search)

SELECT \* FROM flipkart\_mobiles WHERE LOWER(Brand) = 'apple';

-- Query all records from the richperson table

SELECT \* FROM richperson;

-- Find the full name and length of each rich person

SELECT full\_name, LENGTH(full\_name) AS name\_length FROM richperson;

-- Query all records from the emp table

SELECT \* FROM emp;

-- Create a new column dynamically containing name and department name of emp

SELECT CONCAT(name, ' - ', department) AS name\_department FROM emp;

-- Get a substring of the brand starting from the 4th character

SELECT SUBSTRING(Brand, 4) AS brand\_substr FROM flipkart\_mobiles;

-- Give the first 5 characters from email in uppercase

SELECT UPPER(LEFT(email, 5)) AS email\_prefix FROM emp;

-- Give the last 3 characters from email

SELECT RIGHT(email, 3) AS email\_suffix FROM emp;

-- Trim spaces from a string

SELECT TRIM(' space string ') AS trimmed\_string;

-- Concatenate name and email with a space in between

SELECT CONCAT(name, ' ', email) AS name\_email FROM emp;

-- Replace the name 'Felicle' with 'CHANGED NAME' where name is 'Felicle'

SELECT REPLACE(name, name, 'CHANGED NAME') AS changed\_name FROM emp WHERE name = 'Felicle';