Built-in functions and aggregation

MySQL functions

MySQL has a number of built-in functions that can be used to perform common tasks. They are divided in the following categories:

1. Numeric
2. String
3. Date and time
4. Miscellaneous

Numeric functions

Function	Description	Example
ABS	Absolute value	SELECT ABS(-1)
CEIL	Round up to the nearest integer	SELECT CEIL(1.5)
FLOOR	Round down to the nearest integer	SELECT FLOOR(1.5)
ROUND	Round to the given precision	SELECT ROUND(1.54, 1)
TRUNCATE	Truncate to the given precision	SELECT TRUNCATE(1.54, 1)
RAND	Generate a random number	SELECT RAND()

See more function here.

String functions

Function	Description	Example

Function	Description	Example
LENGTH	Length of the string	SELECT LENGTH('Kattapa')
LOWER	Convert to lowercase	SELECT LOWER('Kattapa')
UPPER	Convert to uppercase	SELECT UPPER('Kattapa')
LTRIM	Trim leading spaces	SELECT LTRIM(' Kattapa')
RTRIM	Trim trailing spaces	SELECT RTRIM('Kattapa ')
TRIM	Trim leading and trailing spaces	SELECT TRIM(' Kattapa ')
SUBSTR	Extract a substring	SELECT SUBSTR('Namma Bengaluru', 1, 3)
LEFT	Extract left substring	SELECT LEFT('Namma Bengaluru', 3)
RIGHT	Extract right substring	SELECT RIGHT('Namma Bengaluru', 3)
LOCATE	Find the position of a substring	SELECT LOCATE('Bengaluru', 'Namma Bengaluru')

See more function here.

Date and time functions

Function	Description	Example
NOW	Current date and time	SELECT NOW()
CURDATE	Current date	SELECT CURDATE()
CURTIME	Current time	SELECT CURTIME()
YEAR	Year of the date	SELECT YEAR('2020-01-01')
MONTH	Month of the date	SELECT MONTH('2020-01-01')
DAY	Day of the date	SELECT DAY('2020-01-01')
DAYNAME	Day of the week	SELECT DAYNAME('2020-01-01')
DAYOFWEEK	Day of the week	SELECT DAYOFWEEK('2020-01-01')
DATE_ADD	Add a date	SELECT DATE_ADD('2020-01-01', INTERVAL 1 DAY)
DATE_SUB	Subtract a date	SELECT DATE_SUB('2020-01-01', INTERVAL 1 DAY)
DATEDIFF	Difference between two dates	SELECT DATEDIFF('2020-01-01', '2020-01-02')

Miscellaneous functions

Function	Description	Example
IFNULL	Replace NULL values	SELECT IFNULL(batch_id, 'NO BATCH')
COALESCE	Replace NULL values recursively	SELECT COALESCE(batch_id, phone, email, first_name)

Function	Description	Example
IF	Conditional expression	SELECT IF(batch_id = 1, 'YES', 'NO')
CASE	Conditional expression	SELECT CASE WHEN batch_id = 1 THEN 'YES' ELSE 'NO' END

Aggregate Functions

> MySQL's aggregate function is used to perform calculations on multiple values and return the result in a single value like the average of all values, the sum of all values, and maximum & minimum value among certain groups of values.

> We mainly use the aggregate functions in databases, spreadsheets and many other data manipulation software packages. In the context of business, different organization levels need different information such as top levels managers interested in knowing whole figures and not the individual details. These functions produce the summarised data from our database. Thus, they are extensively used in economics and finance to represent the economic health or stock and sector performance.

The ISO SQL standard defines the following aggregate functions:

Function	Description
MIN	Return the minimum value
MAX	Return the maximum value
SUM	Return the sum of all values
AVG	Return the average value of the argument
COUNT	Return a count of the number of rows returned

Get the minimum value

Keyword: `MIN` **Syntax**: `SELECT MIN(column_name) FROM table_name`

Getting the minimum value without using the aggregate function could be done by using the `ORDER BY` clause.

```
SELECT
    first_name, iq
FROM
    students
ORDER BY iq
LIMIT 1;
```

This would give the correct answer but not in the case of NULL values. By default, MySQL will return the NULL value at the top of the sorted rows. Special handling will have to be done to get the correct result.

This is why we can use the aggregate function `MIN` to get the minimum value.

```
SELECT
    MIN(iq)
FROM
    students;
```

The aggregate functions ignore the NULL values.

Get the maximum value

Keyword: `MAX` **Syntax**: `SELECT MAX(column_name) FROM table_name`

To get the maximum IQ in students, we can use the aggregate function `MAX`.

```
SELECT
    MAX(iq)
FROM
    students;
```

Get the sum of all values

Keyword: `SUM` **Syntax**: `SELECT SUM(column_name) FROM table_name`

To get the sum of all the IQ in students, we can use the aggregate function `SUM`.

```
SELECT
    SUM(iq)
FROM
    students;
```

Get the average value

Keyword: `AVG` **Syntax**: `SELECT AVG(column_name) FROM table_name`

To get the average IQ in students, we can use the aggregate function `AVG`.

```
SELECT
    AVG(iq)
FROM
    students;
```

Get the number of rows

Keyword: `COUNT` **Syntax**: `SELECT COUNT(column_name) FROM table_name`

To get the number of rows in students, we can use the aggregate function `COUNT`.

```
SELECT
    COUNT(*)
FROM
    students;
```

Using `*` instead of a column name will return the number of rows in the table whereas using a column name will return the number of rows that have a non-null value in a column.

GROUP BY clause

The GROUP BY clause groups a set of rows into a set of summary rows by values of columns or expressions. The GROUP BY clause returns one row for each group. In other words, it reduces the number of rows in the result set

Keyword: `GROUP BY` **Syntax**: `SELECT column1, column2, ... FROM table_name GROUP BY column1, column2, ...`

In this syntax, you place the GROUP BY clause after the FROM and WHERE clauses. After the GROUP BY keywords, you place is a list of comma-separated columns or expressions to group rows.

> MySQL evaluates the GROUP BY clause after the FROM and WHERE clauses and before the HAVING, SELECT, DISTINCT, ORDER BY and LIMIT clauses

Fields that are not encapsulated within an aggregate function and must be included in the GROUP BY Clause at the end of the SQL statement.

For example, the following query will return the number of students in each batch:

```
SELECT
    batch_id, AVG(iq)
FROM
    students
GROUP BY batch_id;
```

You can also use the `HAVING` clause to filter the result set. For example, the following query will return the number of students in each batch where each batches average IQ is greater than or equal to 100:

```
SELECT
    batch_id, AVG(iq)
FROM
    students
```

```
GROUP BY batch_id
HAVING AVG(iq) >= 100;
```

Problems

1. Aggregation - I
2. Aggregation - II
3. Aggregation - III
4. Aggregation - IV