IT Department

Built-in Functions in MySQL

Soma Soleiman Zadeh

Database Systems II (IT 216)

Spring 2025 - 2026

Week 4

February 23, 2026

Outline

- Functions
- **Built-In** Functions vs. **User-Defined** Functions
- Some Built-In Functions in MySQL
  - **MySQL String Functions**
  - **MySQL Numeric Functions**
  - **MySQL Date Functions**

Functions

- A **MySQL function** is a **named, reusable sequence of SQL statements**.
- It takes **zero** or **more arguments** as **input**, performs specific operations, and **returns a value**.
- **Functions** can be utilized within **SQL statements**, making them a powerful tool for data manipulation and calculation.

Built-in Functions vs. User-defined Functions

- **Built-in functions** are predefined functions that can be used to perform operations on data at any time.
  - They are usually categorized according to the data types (strings, date, numeric).
- **User-Defined Functions (UDFs)** are one of the most useful features in **MySQL**, allowing users to extend MySQL functionality by creating custom functions.

MySQL String Functions

Function	Description
char_length( )	Returns the length of a string (in characters)
concat( )	Adds two or more expressions together
concat_ws( )	Adds two or more expressions together with a separator
position( )	Returns the position of the first occurrence of a substring in a string
replace( )	Replaces all occurrences of a substring within a string, with a new substring
strcmp( )	Compares two strings
trim( )	Removes spaces from the beginning and end of a string

CHAR_LENGTH( ) Function

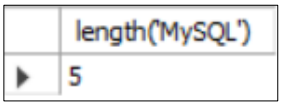
- **CHAR_LENGTH( )** returns the length of a given string in characters (number of characters).

Syntax:

```
CHAR_LENGTH(str)
```

Example:

```
SELECT  CHAR_LENGTH ( 'MySQL' );
```



CONCAT( ) Function

- **CONCAT( )** returns the string that results from concatenating the arguments.

Syntax:

CONCAT(***str1, str2, …***)

Example:

```
SELECT  CONCAT( 'My' , 'S' , 'QL' );
```
→ MySQL

```
SELECT  CONCAT( 'Hello' , NULL , 'ALL');
```
→ NULL

CONCAT( ) Function

Example: Use **CONCAT( )** function to get the following output from the student table.

student

SID	firstName	lastName
1	Sara	Bahram
2	Noor	Ali
3	Ali	Omar

Output

SID	fullName
1	Sara Bahram
2	Noor Ali
3	Ali Omar

Output

```
SELECT  SID, CONCAT( firstName , ' ' , lastName) AS fullName
FROM  student;
```

CONCAT_WS( ) Function

- **CONCAT_WS( )** stands for **C**oncatenate **W**ith **S**eparator and is a special form of **concat( ).**
- **CONCAT_WS( )** returns the string that results from concatenating the arguments with the separator added between them.

Syntax:

CONTACT_WS(***separator, str1, str2, …***)

Example:

SELECT CONTACT_WS('_' , 'Spring' , '2025' , '2026'); → Spring_2025_2026

CONCAT_WS( ) Function

Example: Use **CONCAT_WS( )** function to get the following output from the student table.

student

SID	firstName	lastName
1	Sara	Bahram
2	Noor	Ali
3	Ali	Omar

Output

SID	fullName
1	Sara Bahram
2	Noor Ali
3	Ali Omar

Output

```
SELECT  SID, CONCAT_WS( ' ', firstName , lastName) AS fullName
FROM  student;
```

POSITION( ) Function

- **POSITION( )** returns the position of the first occurrence of a substring in a string.
- If the substring is not found within the original string, this function returns 0.

Syntax:

```
POSITION(substr IN str)
```

Example:

```
SELECT POSITION( 'a' IN 'Database' );
```

1 4 6

2

2

REPLACE( ) Function

- **REPLACE( )** replaces all occurrences of a substring within a string, with a new substring.

Syntax:

```
REPLACE(str , old_substr , new_substr)
```

Example:

```
SELECT REPLACE( 'This is the Fall semester.' , 'Fall' , 'Spring' );
```

This is the Spring Semester.

TRIM( ) Function

- **TRIM( )** function removes leading and trailing spaces from a string.

Syntax:

TRIM(*str*)

Example:

SELECT TRIM(‘ *This is the Spring semester!* ’);

This is the Spring semester!

STRCMP( ) Function

- **STRCMP( )** compares two strings:
  - If **string1 = string2**, the function returns **0**.
  - If **string1 ≠ string2**, the function returns **1** or **-1** according to the sum of ASCII and the position of the characters.

Syntax:

STRCMP(*str1 , str2*)

STRCMP( ) Function

Example:

```
SELECT STRCMP( 'MySQL' , 'MySQL' );
```

0

```
SELECT STRCMP( 'MySQL' , 'Access' );
```

1

```
SELECT STRCMP( 'Access' , 'MySQL' );
```

-1

Some MySQL Numeric Functions

Function	Description
SUM( )	Returns the sum of a set of values
AVG()	Returns the average value of a set of values
MAX( )	Returns the maximum value in a set of values
MIN( )	Returns the minimum value in a set of values
COUNT( )	Returns the number of records returned by a select query

Examples of MySQL Numeric Functions

product

pCode	PName	price
1	A	100
2	B	50
3	C	200
4	D	150

```
SELECT SUM(price) FROM product;   → 500
SELECT AVG(price) FROM product;   → 125
SELECT MIN(price) FROM product;   → 50
SELECT MAX(price) FROM product;   → 200
SELECT COUNT(*) FROM product;     → 4
```

Some MySQL Date Functions

Function	Description
CURRENT_DATE()	Returns the current date.
CURRENT_TIME( )	Returns the current time.
CURRENT_TIMESTAMP( )	Returns the current date and time.
DATEDIFF( )	Returns the number of days between two date values.
DAYNAME( )	Returns the weekday name for a given date.
MONTHNAME()	Returns the name of the month for a given date.
YEAR()	Returns the year part for a given date.

Examples of MySQL Date Functions

```
SELECT CURRENT_DATE( );
```
→ 2026-02-23

```
SELECT CURRENT_TIME( );
```
→ 11:05:45

```
SELECT CURRENT_TIMESTAMP( );
```
→ 2026-02-23 11:05:45

```
SELECT DATEDIFF("2026-02-23" , "2026-01-01");
```
→ 53

Examples of MySQL Date Functions

```
SELECT DAYNAME("2026-02-23");
```
→ Monday

```
SELECT MONTHNAME("2026-02-23");
```
→ February

```
SELECT YEAR("2026-02-23");
```
→ 2026

Let's Try Using Some Functions

Question – Add a new column to the product table named "Days_to_Expiry" that shows the number of days remaining to the expiry date.

- **Step 1 –** Add a new column "DTE" to the Product table (using an ALTER statement).
- **Step 2 –** Update values in the recently added column "DTE" (using an UPDATE statement).

product

pCode	PName	price	expiry_date
1	A	100	2026-06-23
2	B	50	2026-11-01
3	C	200	2026-12-23
4	D	150	2026-02-28

Adding new column "DTE" storing days to expiry of products

product

pCode	PName	price	expiry_date	DTE
1	A	100	2026-06-23	120
2	B	50	2026-11-01	251
3	C	200	2026-12-23	303
4	D	150	2026-02-28	5

Thank You!