MySQL: Flow-Control Statements

Tushar B. Kute,
http://tusharkute.com

MySQL Function

- A stored function in MySQL is a set of SQL statements that perform some task/operation and return a single value.
- It is one of the types of stored programs in MySQL. When you will create a stored function, make sure that you have a CREATE ROUTINE database privilege.
- Generally, we used this function to encapsulate the common business rules or formulas reusable in stored programs or SQL statements.

MySQL Function

- The stored function is almost similar to the procedure in MySQL, but it has some differences that are as follows:
  - The function parameter may contain only the IN parameter but can't allow specifying this parameter, while the procedure can allow IN, OUT, INOUT parameters.
  - The stored function can return only a single value defined in the function header.
  - The stored function may also be called within SQL statements.
  - It may not produce a result set.

MySQL Function

- DELIMITER $$
  CREATE FUNCTION fun_name(fun_parameter(s))
  RETURNS datatype
  [NOT] {Characteristics}
  fun_body;

MySQL Function

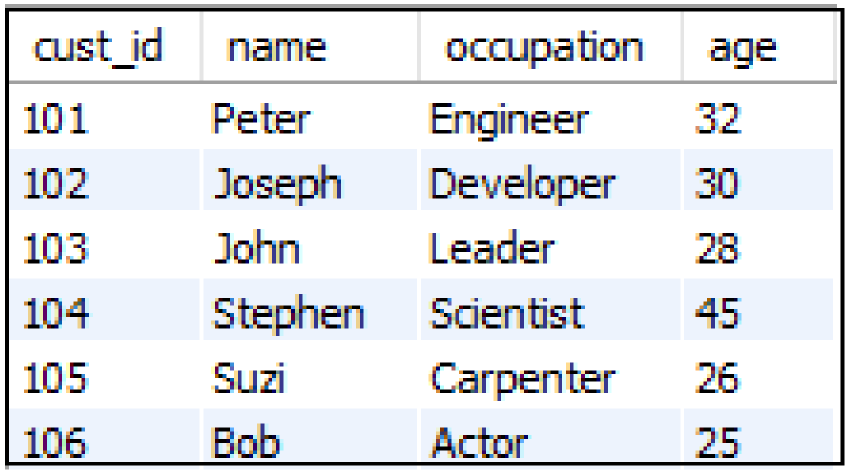
- fun_name
  - It is the name of the stored function that we want to create in a database. It should not be the same as the built-in function name of MySQL.
- fun_parameter
  - It contains the list of parameters used by the function body. It does not allow to specify IN, OUT, INOUT parameters.
- datatype
  - It is a data type of return value of the function. It should any valid MySQL data type.

MySQL Function

- characteristics
  - The CREATE FUNCTION statement only accepted when the characteristics (DETERMINISTIC, NO SQL, or READS SQL DATA) are defined in the declaration.
- fun_body
  - This parameter has a set of SQL statements to perform the operations. It requires at least one RETURN statement.
  - When the return statement is executed, the function will be terminated automatically. The function body is given below: BEGIN -- SQL statements END $$ DELIMITER

Example:

- Let us understand how stored function works in MySQL through the example.
- Suppose our database has a table named "customer" that contains the following data:



cust_id	name	occupation	age
101	Peter	Engineer	32
102	Joseph	Developer	30
103	John	Leader	28
104	Stephen	Scientist	45
105	Suzi	Carpenter	26
106	Bob	Actor	25

Example:

```
DELIMITER $$
CREATE FUNCTION Customer_Occupation(
    age int
)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE customer_occupation VARCHAR(20);
    IF age > 35 THEN
        SET customer_occupation = 'Scientist';
    ELSEIF (age <= 35 AND
            age >= 30) THEN
        SET customer_occupation = 'Engineer';
    ELSEIF age < 30 THEN
        SET customer_occupation = 'Actor';
    END IF;
    -- return the customer occupation
    RETURN (customer_occupation);
END$$
DELIMITER;
```

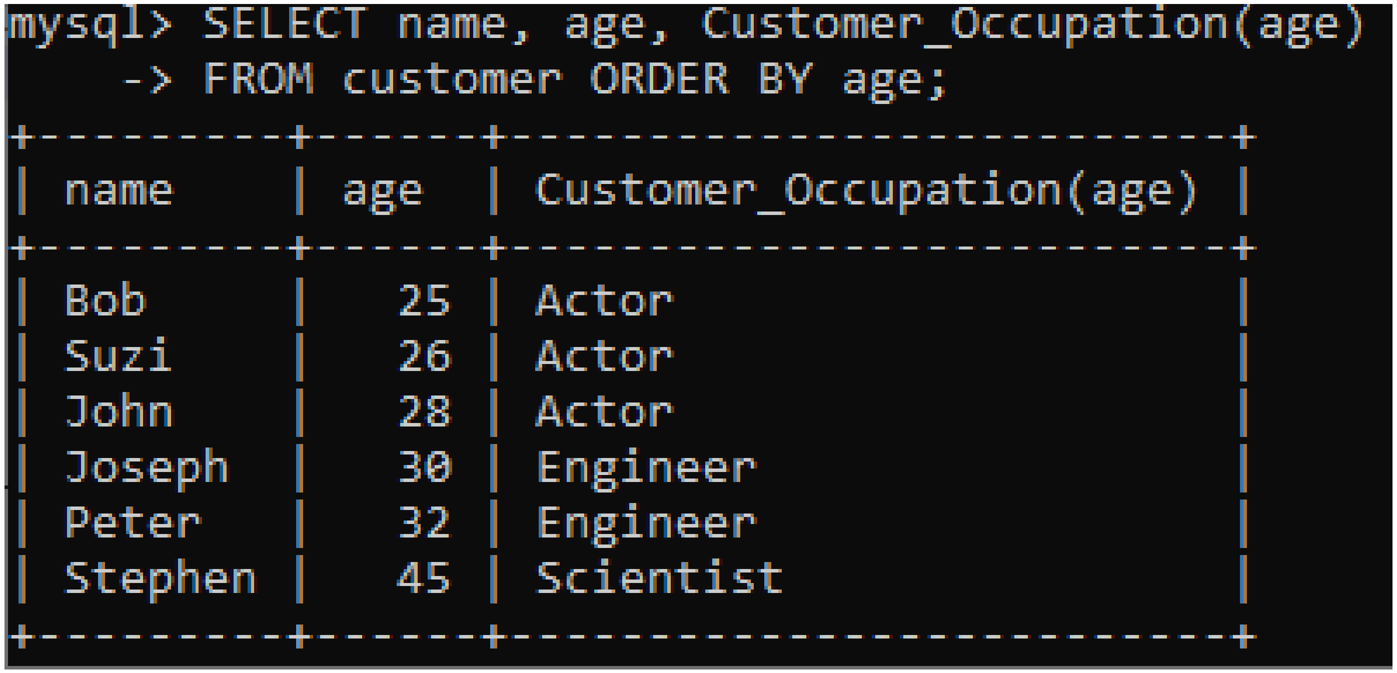
Example:

- Now, we are going to see how stored function is called with the SQL statement.
- The following statement uses customer_occupation stored function to get the result:

```
SELECT name, age, Customer_Occupation(age)
FROM customer ORDER BY age;
```

- It will give the output as below.

Example:


```
mysql> SELECT name, age, Customer_Occupation(age)
    -> FROM customer ORDER BY age;
+----------+------+--------------------------+
| name     | age  | Customer_Occupation(age) |
+----------+------+--------------------------+
| Bob      |   25 | Actor                    |
| Suzi     |   26 | Actor                    |
| John     |   28 | Actor                    |
| Joseph   |   30 | Engineer                 |
| Peter    |   32 | Engineer                 |
| Stephen  |   45 | Scientist                |
+----------+------+--------------------------+
```

Loop

- Similar to other programming languages MySQL provides support for the flow control statements such as IF, CASE, ITERATE, LEAVE LOOP, WHILE, and REPEAT.
- You can use these statements in the stored programs (procedures), and RETURN in stored functions. You can use one Flow Control Statement with in another.
- The LOOP is a compound MySQL statement which is used to execute a single or set of statements repeatedly.

Iterate

- The ITERATE statement is used to restart the LOOP, REPEAT or, WHILE statements.
- It cannot be used outside these statements.
- Syntax:

  ITERATE label
- Where, label is the label of the LOOP or, REPEAT or, WHILE statement.

Iterate

```
DELIMITER //
CREATE FUNCTION Sample (bonus INT)
  RETURNS INT
  BEGIN
    DECLARE income INT;
    SET income = 0;
    myLabel: LOOP
      SET income = income + bonus;
      IF income < 10000 THEN
        ITERATE myLabel;
      END IF;
      LEAVE myLabel;
    END LOOP myLabel;
  RETURN income;
END; //
Query OK, 0 rows affected (0.41 sec)
mysql> DELIMITER ;
```

Iterate

```
delimiter //
CREATE procedure proc()
BEGIN
  DECLARE val INT default 15;
  DECLARE res VARCHAR(255) default '';
  label: LOOP
    IF val < 0 THEN
      LEAVE label;
    END IF;
    SET res = CONCAT(res, val, ',');
    SET val = val –1;
    ITERATE label;
  END LOOP;
  SELECT res;
END//
Delimiter ;
```

Leave

- The LEAVE statement in MySQL is used to exit the LOOP, REPEAT, WHILE statements or, BEGIN…END statements.
- It cannot be used outside these statements.
- Syntax:

  LEAVE label
- Where, label is the label of the LOOP or, REPEAT or, WHILE statement.

Leave

```
mysql> Delimiter //
mysql> CREATE PROCEDURE demo()
  BEGIN
    DECLARE num INT;
    DECLARE str VARCHAR(50);
    SET num = 1;
    SET str = '';
    label: LOOP
    SET num = num + 1;
    IF num > 16 THEN
      LEAVE label;
    END IF;
    IF (num mod 2) THEN
      ITERATE label;
    ELSE
      SET str = CONCAT(str, num, ',');
    END IF;
  END LOOP;
  SELECT str;
  END //
```

In-built functions

- ASCII( )
  - The ASCII() function returns the ASCII value for the specific character.
- CHARACTER_LENGTH( )
  - The CHAR_LENGTH() function return the length of a string (in characters).
- CONCAT( )
  - The CONCAT() function adds two or more expressions together.

In-built functions

- CONCAT_WS()
  - It adds two or more expressions together with a separator.
  - Syntax: CONCAT_WS(separator, expression1, expression2, expression3,...)
- FIELD( )
  - The FIELD() function returns the index position of a value in a list of values.
  - This function performs a case-insensitive search.
  - Note: If the specified value is not found in the list of values, this function will return 0. If value is NULL, this function will return 0.
  - SELECT FIELD("q", "s", "q", "l");

In-built functions

- FIND_IN_SET( )
  - The FIND_IN_SET() function returns the position of a string within a list of strings.
  - FIND_IN_SET(string, string_list)
  - SELECT Find_in_set("l", "s,q,l");
- FORMAT()
  - It formats a number to a format like "#,###,###.##", rounded to a specified number of decimal places, then it returns the result as a string.
  - SELECT format(marks,1) from student;

In-built functions

- INSERT()
  - It inserts a string within a string at the specified position and for a certain number of characters.
  - INSERT(string, position, number, string2)
  - SELECT insert(name,3,1,'x') from student;
- INSTR()
  - It returns the position of the first occurrence of a string in another string.
  - This function performs a case-insensitive search.
  - SELECT instr(name,'an') from student;

In-built functions

- INSERT()
  - It inserts a string within a string at the specified position and for a certain number of characters.
  - INSERT(string, position, number, string2)
  - SELECT insert(name,3,1,'x') from student;
- INSTR()
  - It returns the position of the first occurrence of a string in another string.
  - This function performs a case-insensitive search.
  - SELECT instr(name,'an') from student;

In-built functions

- LCASE() / LOWER()
  - It converts a string to lower-case.
- LEFT()
  - It extracts a number of characters from a string (starting from left).
  - select left(name, 3) as name from student;
- RIGHT()
  - It extracts a number of characters from a string (starting from right).
  - select right(name, 3) as name from student;
-

In-built functions

- LENGTH()
  - It returns the length of a string (in bytes).
- LOCATE() / POSITION()
  - It 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.
  - This function performs a case-insensitive search.
  - LOCATE(substring, string, start)
  - select locate('il',name,2) as name from student;

In-built functions

- LPAD()
  - It left-pads a string with another string, to a certain length.
  - LPAD(string, length, lpad_string)
  - select lpad(name,10,'-') as name from student;
- RPAD()
  - It right-pads a string with another string, to a certain length.
  - RPAD(string, length, lpad_string)
  - select rpad(name,10,'-') as name from student;

In-built functions

- LTRIM()
  - It removes leading spaces from a string.
- RTRIM()
  - It removes trailing spaces from a string.
- MID() / SUBSTR() / SUBSTRING()
  - It extracts a substring from a string (starting at any position).
  - MID(string, start, length)
  - select mid(name,1,3) as name from student;

In-built functions

- REPEAT()
  - It repeats a string as many times as specified.
  - select repeat(name,2) as name from student;
- REPLACE()
  - It replaces all occurrences of a substring within a string, with a new substring.
  - Note: This function performs a case-sensitive replacement.
  - REPLACE(string, substring, new_string)
  - select replace(name,'i','ee') as name from student;

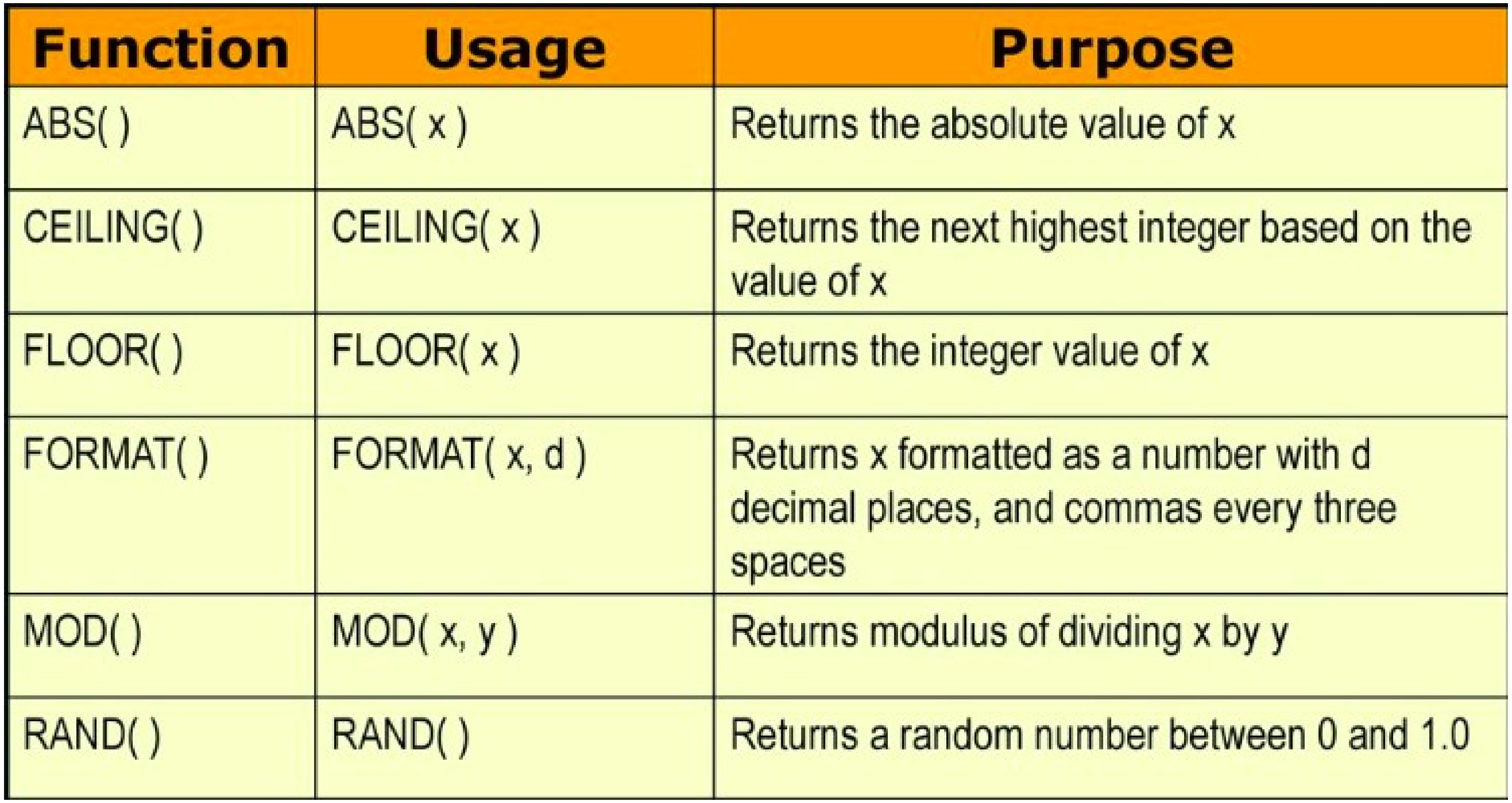
In-built functions

- REVERSE()
  - It reverses a string and returns the result.
  - select reverse(name) as name from student;
- SPACE()
  - It returns a string of the specified number of space characters.
  - select space(4) as sps;
- STRCMP()
  - It compares two strings.
  - STRCMP(string1, string2)
  - select strcmp(name, class) as name from student;

In-built functions

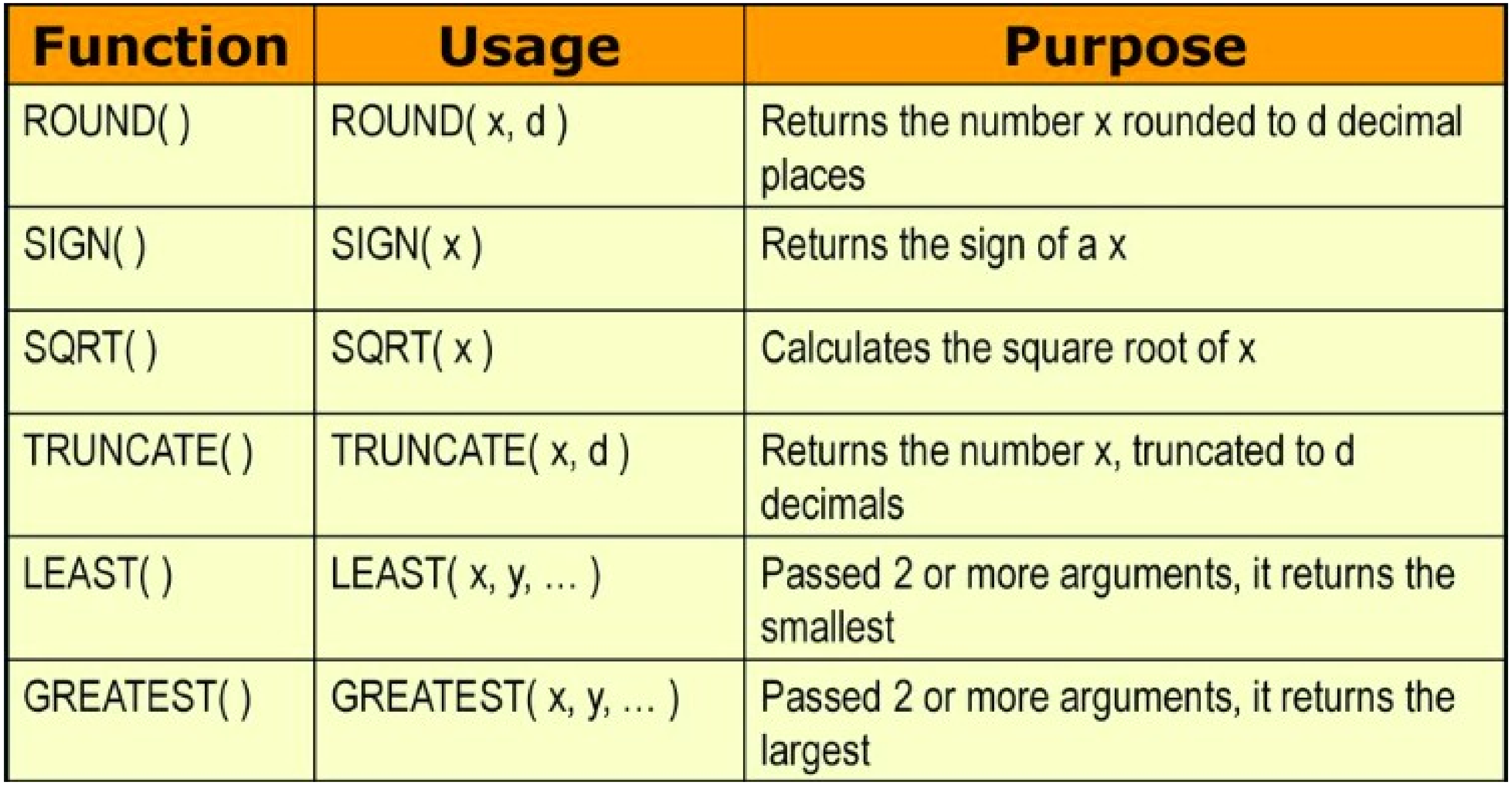
- SUBSTRING_INDEX()
  - It returns a substring of a string before a specified number of delimiter occurs.
  - SUBSTRING_INDEX(string, delimiter, number)
  - select substring_index(name,'a',2) as name from student;
- TRIM()
  - It removes leading and trailing spaces from a string.
- UCASE() / UPPER()
  - It converts a string to upper-case.

Numeric Functions



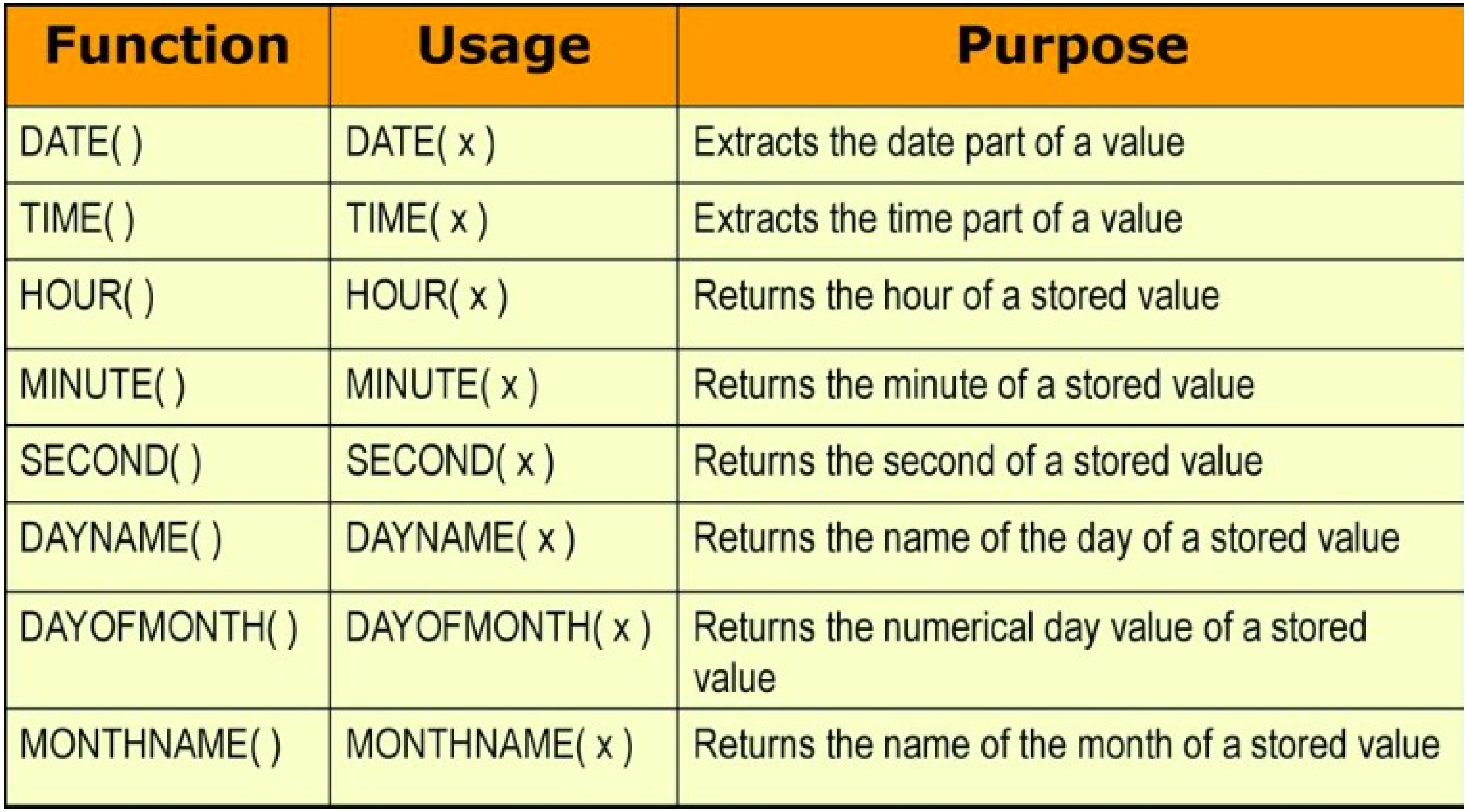
Function	Usage	Purpose
ABS( )	ABS(x)	Returns the absolute value of x
CEILING( )	CEILING(x)	Returns the next highest integer based on the value of x
FLOOR( )	FLOOR(x)	Returns the integer value of x
FORMAT( )	FORMAT(x, d)	Returns x formatted as a number with d decimal places, and commas every three spaces
MOD( )	MOD(x, y)	Returns modulus of dividing x by y
RAND( )	RAND( )	Returns a random number between 0 and 1.0

Numeric Functions



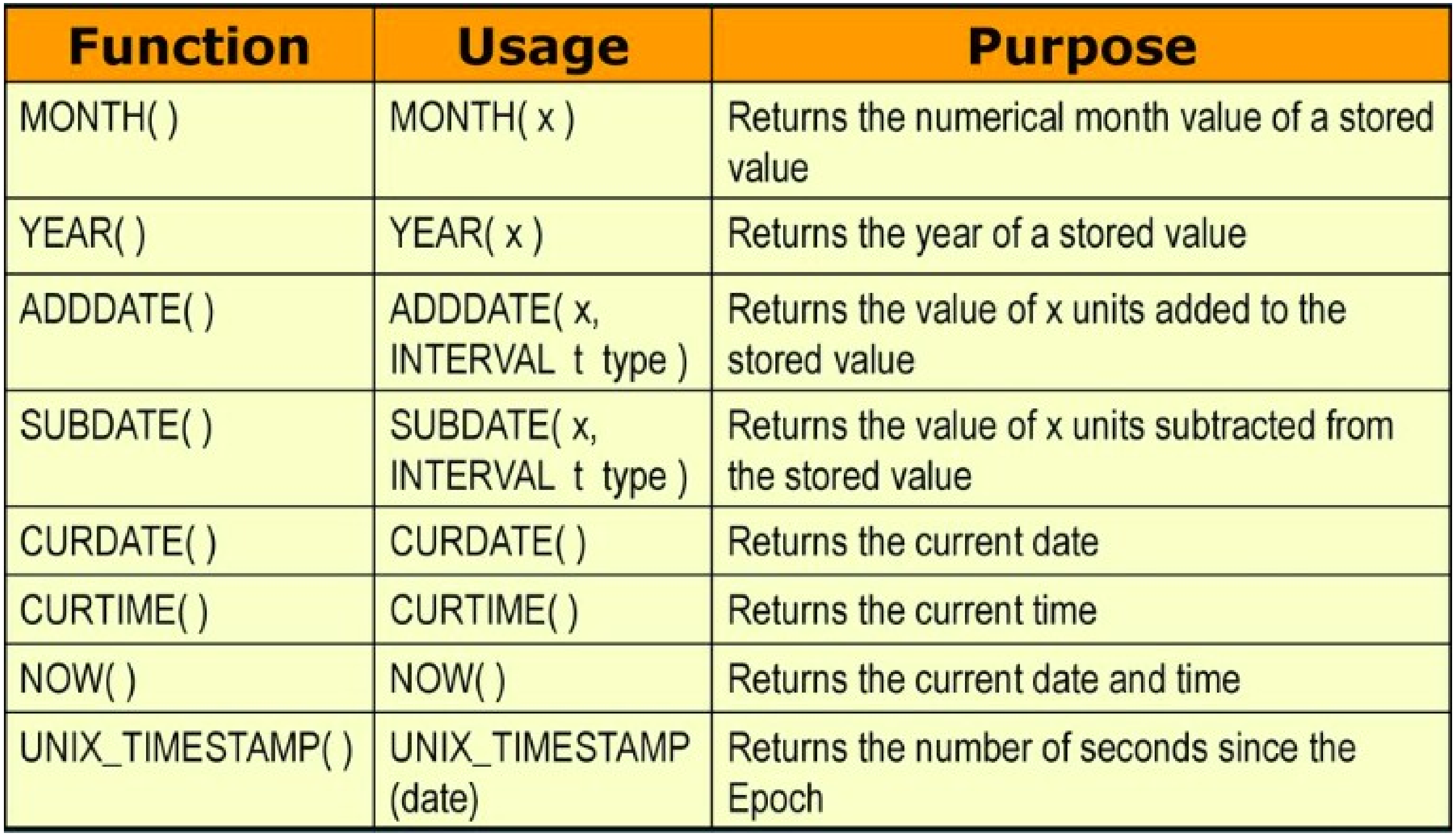
Function	Usage	Purpose
ROUND( )	ROUND(x, d)	Returns the number x rounded to d decimal places
SIGN( )	SIGN(x)	Returns the sign of a x
SQRT( )	SQRT(x)	Calculates the square root of x
TRUNCATE( )	TRUNCATE(x, d)	Returns the number x, truncated to d decimals
LEAST( )	LEAST(x, y, …)	Passed 2 or more arguments, it returns the smallest
GREATEST( )	GREATEST(x, y, …)	Passed 2 or more arguments, it returns the largest

Data/Time Functions



Function	Usage	Purpose
DATE( )	DATE(x)	Extracts the date part of a value
TIME( )	TIME(x)	Extracts the time part of a value
HOUR( )	HOUR(x)	Returns the hour of a stored value
MINUTE( )	MINUTE(x)	Returns the minute of a stored value
SECOND( )	SECOND(x)	Returns the second of a stored value
DAYNAME( )	DAYNAME(x)	Returns the name of the day of a stored value
DAYOFMONTH( )	DAYOFMONTH(x)	Returns the numerical day value of a stored value
MONTHNAME( )	MONTHNAME(x)	Returns the name of the month of a stored value

Data/Time Functions



Function	Usage	Purpose
MONTH( )	MONTH(x)	Returns the numerical month value of a stored value
YEAR( )	YEAR(x)	Returns the year of a stored value
ADDDATE( )	ADDDATE(x, INTERVAL t type)	Returns the value of x units added to the stored value
SUBDATE( )	SUBDATE(x, INTERVAL t type)	Returns the value of x units subtracted from the stored value
CURDATE( )	CURDATE( )	Returns the current date
CURTIME( )	CURTIME( )	Returns the current time
NOW( )	NOW( )	Returns the current date and time
UNIX_TIMESTAMP( )	UNIX_TIMESTAMP (date)	Returns the number of seconds since the Epoch

Thank you

@mitu_skillologies @mITuSkillologies @mitu_group @mitu-skillologies @MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com