User Defined Functions & Window Functions

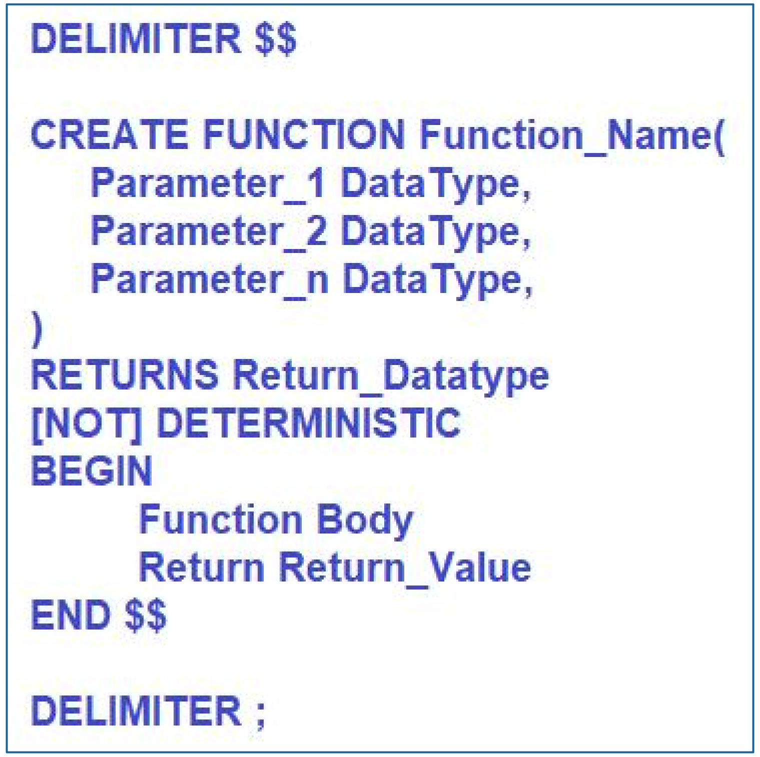
User Defined Functions

- Functions are blocks of codes to perform specific tasks and return the result. User defined functions are basic building blocks of a program.

- Functions can be classified into two categories, namely, system-defined functions and user-defined functions. The functions which are developed by user at the time of writing a program are called user defined functions. Thus, user defined functions are functions developed by user.

- The function which is defined by the user is called a user-defined function. MySQL user-defined functions may or may not have parameters that are optional, but it always returns a single value that is mandatory. The returned value which is return by the MySQL Function can be of any valid MySQL data type.

Syntax:



```
DELIMITER $$

CREATE FUNCTION Function_Name(
    Parameter_1 DataType,
    Parameter_2 DataType,
    Parameter_n DataType,
)
RETURNS Return_Datatype
[NOT] DETERMINISTIC
BEGIN
        Function Body
        Return Return_Value
END $$

DELIMITER ;
```

UDF Explanation

- First, we need to specify the name of the user-defined function that you want to create after the CREATE FUNCTION statement.

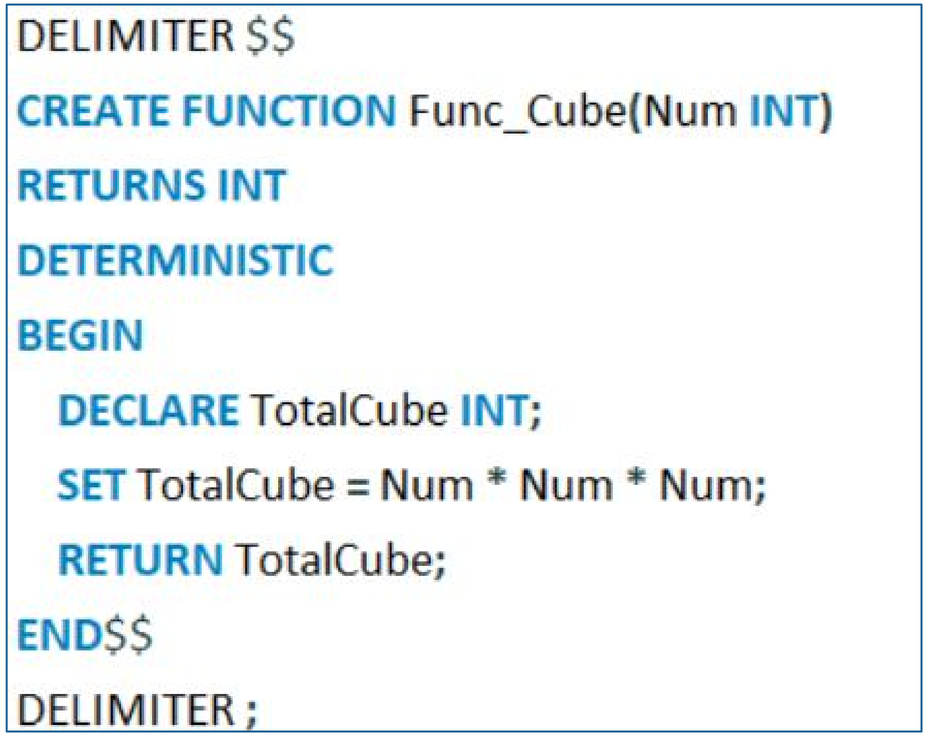
- Second, list all the input parameters of the user-defined function inside the parentheses followed by the function name. By default, all the parameters are IN parameters. The point that you need to remember is, you cannot specify IN, OUT or INOUT modifiers to the parameters in MySQL.

- Third, specify the data type of the return value in the RETURNS statement, which can be any valid MySQL data type.

- Fourth, specify if the function is deterministic or not using the DETERMINISTIC keyword. It is optional. If we don’t specify DETERMINISTIC or NOT DETERMINISTIC, by default. MySQL uses the NOT DETERMINISTIC option. A deterministic function in MySQL always returns the same result for the same input parameters whereas a non-deterministic function returns different results for the same input parameters.
- Fifth, write the code in the body of the user-defined function within the BEGIN & END block. Inside the body section, you need to specify at least one RETURN statement. The RETURN statement returns a value to the calling programs. Whenever the RETURN statement is reached, the execution of the stored function is terminated immediately.

UDF Example

Below function is used to return the cube of any number



```
DELIMITER $$
CREATE FUNCTION Func_Cube(Num INT)
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE TotalCube INT;
   SET TotalCube = Num * Num * Num;
   RETURN TotalCube;
END$$
DELIMITER ;
```

Window Functions

- Windows function in MySQL helps to solve a query problem. The operation is performed on a set of query rows while keeping the number of rows intact.
- The window functions allow you to solve query problems in new, easier ways and with better performance.
- Aggregate functions summarize the data from multiple rows into a single result row, but window functions, which also operate on a subset of rows, do not reduce the number of rows returned by the query.

Syntax:

```
window_function (expression) OVER ( [partition_definition] [order_defenition] )
```

where,

• window_function:	The name of the window function
• Expression:	A field on which function needs to be performed
• partition_definition:	Used to divide or break the rows into partitions
• order_definition:	Used to specify the order of the rows within a partition

Window Functions

- Windows function in MySQL helps to solve a query problem. The operation is performed on a set of query rows while keeping the number of rows intact.

- The window functions allow you to solve query problems in new, easier ways and with better performance.

- Aggregate functions summarize the data from multiple rows into a single result row, but window functions, which also operate on a subset of rows, do not reduce the number of rows returned by the query.

Syntax:

```
window_function (expression) OVER ( [partition_definition] [order_defenition] )
```

where,

window_function:	The name of the window function
Expression:	A field on which function needs to be performed
partition_definition:	Used to divide or break the rows into partitions
order_definition:	Used to specify the order of the rows within a partition

Window Functions - Example

Que. Show the total sales of all products for each year

Without window function

```
SELECT Year, Product, SUM(Sale) AS
Total_Sales
FROM Sales
GROUP BY Year
ORDER BY Product;
```

Using window function

```
 SELECT Year, Product,SUM(Sale)
OVER(PARTITION BY Year) AS Total_Sales
 FROM Sales;`
```

Window Functions Types

Aggregate Functions

- SUM
- AVG
- MIN
- MAX
- COUNT

Ranking Functions

- RANK
- DENSE_RANK
- PERCENT_RANK
- ROW_NUMBER
- CUME_DIST

Analytical Functions

- NTILE
- LAG
- LEAD
- FIRST_VALUE
- LAST_VALUE
- NTH_VALUE

ROW_NUMBER()	Used to insert the row numbers.
RANK()	Provides every row with rank value that may or may not be consecutive in nature
DENSE_RANK()	Will assign consecutive rank numbers for each group.
PERCENT_RANK()	Calculates the percentile of a row within the small result set and will return a value ranging from 0 to 1 for every row
CUME_DIST()	Returns a value that represents the number of rows with values less than or equal to (<=)the current row's value divided by the total number of rows.
NTILE()	Helps to split the result set rows into a specified number of groups based upon the partition_by and order_by clauses provided.

LAG()	Returns the value to the previous rows in a sorted and partitioned result set.
LEAD()	Will return the values ahead, in the partitioned and sorted result set.
FIRST_VALUE()	Returns the first row among a partitioned sorted result set.
LAST_VALUE()	Returns the last row among a partitioned sorted result set.
NTH_VALUE()	Returns the Nth row among a partitioned sorted result set.