**The most general difference between procedures and functions is that they are invoked differently and for different purposes:**

1. A procedure does not return a value. Instead, it is invoked with a CALL statement to perform an operation such as modifying a table or processing retrieved records.
2. A function is invoked within an expression and returns a single value directly to the caller to be used in the expression.
3. You cannot invoke a function with a CALL statement, nor can you invoke a procedure in an expression.

**Syntax for routine creation differs somewhat for procedures and functions:**

1. Procedure parameters can be defined as input-only, output-only, or both. This means that a procedure can pass values back to the caller by using output parameters. These values can be accessed in statements that follow the CALL statement. Functions have only input parameters. As a result, although both procedures and functions can have parameters, procedure parameter declaration differs from that for functions.
2. Functions return value, so there must be a RETURNS clause in a function definition to indicate the data type of the return value. Also, there must be at least one RETURN statement within the function body to return a value to the caller. RETURNS and RETURN do not appear in procedure definitions.
   * To invoke a stored procedure, use the CALL statement. To invoke a stored function, refer to it in an expression. The function returns a value during expression evaluation.
   * A procedure is invoked using a CALL statement, and can only pass back values using output variables. A function can be called from inside a statement just like any other function (that is, by invoking the function's name), and can return a scalar value.
   * Specifying a parameter as IN, OUT, or INOUT is valid only for a PROCEDURE. For a FUNCTION, parameters are always regarded as IN parameters.
3. If no keyword is given before a parameter name, it is an IN parameter by default. **Parameters for stored functions are not preceded by IN, OUT, or INOUT.** All function parameters are treated as IN parameters.

**To define a stored procedure or function, use CREATE PROCEDURE or CREATE FUNCTION respectively:**

CREATE PROCEDURE proc\_name ([parameters]) [characteristics] routine\_body CREATE FUNCTION func\_name ([parameters]) RETURNS data\_type // diffrent [characteristics] routine\_body

A MySQL extension for stored procedure (not functions) is that a procedure can generate a result set, or even multiple result sets, which the caller processes the same way as the result of a SELECT statement. However, the contents of such result sets cannot be used directly in expression.

**Stored routines** (referring to both stored procedures and stored functions) **are associated with a particular database, just like tables or views.** When you drop a database, any stored routines in the database are also dropped.

**Stored procedures and functions do not share the same namespace.** It is possible to have a procedure and a function with the same name in a database.

**In Stored procedures dynamic SQL can be used but not in functions or triggers.**

SQL prepared statements (PREPARE, EXECUTE, DEALLOCATE PREPARE) can be used in stored procedures, but not stored functions or triggers. Thus, stored functions and triggers cannot use Dynamic SQL (where you construct statements as strings and then execute them). [(Dynamic SQL in MySQL stored routines)](https://dba.stackexchange.com/questions/321/dynamic-sql-in-mysql-stored-routines/1906#1906)

**Some more interesting differences between FUNCTION and STORED PROCEDURE:**

1. (*This point is* [*copied from a blogpost*](https://stackoverflow.com/questions/3744209/mysql-stored-procedure-vs-function-which-would-i-use-when#comment44782440_13510102)*.*) Stored procedure is precompiled execution plan where as functions are not. Function Parsed and compiled at runtime. Stored procedures, Stored as a pseudo-code in database i.e. compiled form.
2. (*I'm not sure for this point.*)
3. Stored procedure has the security and reduces the network traffic and also we can call stored procedure in any no. of applications at a time. [reference](http://phptechnicalgroups.blogspot.in/2012/10/difference-between-mysql-function-and.html)
4. Functions are normally used for computations where as procedures are normally used for executing business logic.
5. Functions Cannot affect the state of database (Statements that do explicit or implicit commit or rollback are disallowed in function) Whereas Stored procedures Can affect the state of database using commit etc.
6. refrence: [J.1. Restrictions on Stored Routines and Triggers](http://www.cs.duke.edu/csl/docs/mysql-refman/restrictions.html#routine-restrictions)
7. Functions can't use [FLUSH](http://dev.mysql.com/doc/refman/5.0/en/flush.html) statements whereas Stored procedures can do.
8. Stored functions cannot be recursive Whereas Stored procedures can be. Note: Recursive stored procedures are disabled by default, but can be enabled on the server by setting the max\_sp\_recursion\_depth server system variable to a nonzero value. See [Section 5.2.3, “System Variables”](http://www.cs.duke.edu/csl/docs/mysql-refman/database-administration.html#server-system-variables), for more information.
9. Within a stored function or trigger, it is not permitted to modify a table that is already being used (for reading or writing) by the statement that invoked the function or trigger. Good Example: [How to Update same table on deletion in MYSQL?](https://stackoverflow.com/questions/13487010/how-to-update-same-table-on-deletion-in-mysql)

**Note**: that although some restrictions normally apply to stored functions and triggers but not to stored procedures, those restrictions do apply to stored procedures if they are invoked from within a stored function or trigger. For example, although you can use FLUSH in a stored procedure, such a stored procedure cannot be called from a stored function or trigger.