

```
CREATE PROCEDURE my_proc_CASE`  
(INOUT no_employees INT, IN salary INT)  
BEGIN  
CASE  
WHEN (salary>10000)  
THEN (SELECT COUNT(job_id) INTO no_employees  
FROM jobs  
WHERE min_salary>10000);  
WHEN (salary<10000)  
THEN (SELECT COUNT(job_id) INTO no_employees  
FROM jobs  
WHERE min_salary<10000);  
ELSE (SELECT COUNT(job_id) INTO no_employees  
FROM jobs WHERE min_salary=10000);  
END CASE;  
END$$
```

Number of employees whose salary greater than 10000 :

To execute

```
CALL my_proc_CASE(@C,10001);
```

```
mysql> SELECT @C;
```

LOOPS

```

CREATE PROCEDURE `my_proc_LOOP` (IN num INT)

BEGIN

DECLARE x INT;

    SET x = 0;

    loop_label: LOOP

        INSERT INTO number VALUES (rand());

        SET x = x + 1;

        IF x >= num

            THEN

                LEAVE loop_label;

            END IF;

    END LOOP;

END$$

```

To execute

```
CALL my_proc_LOOP(3);
```

MySQL: REPEAT Statement

The REPEAT statement executes the statement(s) repeatedly as long as the condition is true. The condition is checked every time at the end of the statements.

```
[begin_label:]
```

REPEAT

statement_list

UNTIL search_condition

END

REPEAT

[end_label]

statement_list: List of one or more statements, each statement terminated by a semicolon(;).

search_condition : An expression.

A REPEAT statement can be labeled.

DELIMITER \$\$

CREATE PROCEDURE my_proc_REPEAT (IN n INT)

BEGI

NSET @sum = 0;

SET @x = 1;

REPEAT

IF mod(@x, 2) = 0

THEN

SET @sum = @sum + @x;

END IF;

SET @x = @x + 1;

UNTIL @x > n

END REPEAT;

```
END $$
```

```
mysql> call my_proc_REPEAT(5);
```

```
SELECT @sum;
```

MySQL: RETURN Statement

The RETURN statement terminates execution of a stored function and returns the value expr to the function caller. There must be at least one RETURN statement in a stored function. There may be more than one if the function has multiple exit points. Here is the syntax :

```
RETURN expr
```

This statement is not used in stored procedures or triggers. The LEAVE statement can be used to exit a stored program of those types.

While Loop

```
DELIMITER $$
```

```
CREATE PROCEDURE my_proc_WHILE(IN n INT)
```

```
BEGIN
```

```
SET @sum = 0;
```

```
SET @x = 1;
```

```
WHILE @x<n
```

```
DO
    IF mod(@x, 2) <> 0 THEN
SET @sum = @sum + @x;
    END IF;
SET @x = @x + 1;
END WHILE;
END$$
```

```
mysql> CALL my_proc_WHILE(5);
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT @sum;
```

ALTER PROCEDURE

This statement can be used to change the characteristics of a stored procedure. More than one change may be specified in an ALTER PROCEDURE statement. However, you cannot change the parameters or body of a stored procedure using this statement; to make such changes, you must drop and re-create the procedure using DROP PROCEDURE and CREATE PROCEDURE. Here is the syntax :

```
ALTER PROCEDURE proc_name [characteristic ...]characteristic:
```

```
COMMENT 'string'
```

```
| LANGUAGE SQL
```

```
| { CONTAINS SQL
```

```
| NO SQL | READS SQL DATA
```

```
| MODIFIES SQL DATA }
```

| SQL SECURITY { DEFINER

| INVOKER }

You must have the ALTER ROUTINE privilege for the procedure. By default, that privilege is granted automatically to the procedure creator. In our previous procedure "my_proc_WHILE" the comment section was empty. To input new comment or modify the previous comment use the following command :

```
mysql> ALTER PROCEDURE my_proc_WHILE
```

```
COMMENT 'Modify Comment';
```

```
>Query OK, 0 rows affected (0.20 sec)
```

You can check the result through SHOW CREATE PROCEDURE command

MySQL: DROP PROCEDURE

This statement is used to drop a stored procedure or function. That is, the specified routine is removed from the server. You must have the ALTER ROUTINE privilege for the routine. (If the automatic_sp_privileges system variable is enabled, that privilege and EXECUTE are granted automatically to the routine creator when the routine is created and dropped from the creator when the routine is dropped

```
DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp_name
```

The IF EXISTS clause is a MySQL extension. It prevents an error from occurring if the procedure or function does not exist. A warning is produced that can be viewed with SHOW WARNINGS. Here is an example:

```
mysql> DROP PROCEDURE new_procedure;  
Query OK, 0 rows affected (0.05 sec)
```

```

delimiter $$
create procedure whileloop()

begin
    declare x int;
    declare str varchar(100);
    set x = 1;
    set str = '' ;
    while x <= 10 do
        set str = concat(str,x,',');
        set x = x + 1;
    end while;
    select str;
end$$
delimiter;

```

```

delimiter $$

create procedure whileloop()
begin
    declare x int;
    declare str varchar(100);
    set x = 1;
    set str = '' ;
    repeat
        set str = concat(str,x,',');
        set x = x + 1;
    until x >=10
    end repeat;
    select str;
end$$
delimiter;

```

```

CREATE PROCEDURE p ()
BEGIN
    DECLARE counter INT DEFAULT 0;
    FOR SELECT a, b FROM t DO
        SET counter = counter + 1;
    END FOR;
    SELECT 'There are ',counter,' rows in t';
END

```