STORED PROCEDURES :

– A **procedure** is a PL/SQL block which performs one or more specific task.

– Procedures do not return a value directly; mainly used to perform an action.

– Stored Procedure increases the performance of the applications. Once stored procedures are created, they are compiled and stored in the database.

– Stored procedure reduces the traffic between application and database server.

– A procedure is called a **recursive stored procedure** when it calls itself. Most database systems support recursive stored procedures. But, **it is not supported well in MySQL**.

Creating a Procedure:

– A procedure is created with the **CREATE OR REPLACE PROCEDURE** statement.

```
CREATE [OR REPLACE] PROCEDURE procedure_name
    [ (parameter [,parameter]) ]

IS
    [declaration_section]

BEGIN
    executable_section

[EXCEPTION
    exception_section]

END [procedure_name];
```

Example

Procedure Code:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `procedure_1`(
    IN employeeID INT,
    OUT empSal INT,
```

```
    OUT manageID INT
)
BEGIN
        SELECT empSalary, managerid
    INTO empSal,manageID
    FROM employees
    WHERE empid = employeeID;
END
```

In Query:

```
call procedure_1(106,@employeeSalary,@managerID);
select @employeeSalary as Employee_Salary, @managerID as Manager_ID;
```

	Employee_Salary	Manager_ID
▶	6000	101

Dropping a Procedure:

Syntax:

```
DROP PROCEDURE procedure_name;
```

LOOP:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `addNext5EvenNumbers`(in i int, out sum
int)
BEGIN
        declare x int;
   set x = 0;
   set sum = i;
   loop1:  LOOP
                set x = x+1;
                IF  x > 10 THEN
                        LEAVE  loop1;
                END  IF;
      SET  i = i + 1;
                IF  (i mod 2) THEN
                        ITERATE  loop1;
                ELSE
                        set sum = sum+i;
```

```
            END  IF;
        END LOOP;
END
```

Output:

```
14
15 •  call addNext5EvenNumbers(1, @sum);
16 •  select @sum as sum;
17
```

Result Grid | Filter Rows: | Export:

sum
31

WHILE LOOP:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Procedure_Factorial`(
        IN num int
)
BEGIN
   DECLARE result INT;
   DECLARE i INT;
   SET result = 1;
   SET i = 1;
   WHILE i <= num DO
   SET result = result * i;
   SET i = i + 1;
   END WHILE;
   SELECT num AS Number1, result as Factorial;

END
```

Output:

Number 1	Factorial
7	5040

Switch Case:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `setShippingCountryById`
(
        in id int,
        out sCountry varchar(40),
        out fname varchar(50)
)
BEGIN
        Declare customerCountry  varchar(20);
        select country,firstName
        into customerCountry , fname
        from customer
        where customer.id = id;

        Case customerCountry

        when 'USA' then
        set sCountry = 'USA';

        when 'Switzerland' then
        set sCountry = 'Switzerland';

        when 'Brazil' then
        set sCountry = customerCountry;

        when 'UK' then
        set sCountry = customerCountry;

        else
          set  sCountry = India;
        end case;
END
```

Call procedure :

```
call setShippingCountryById(16,@shippingCountry,@fname);
select @shippingCountry as shippingCountry, @fname as firstName;
```

Output:

	shippingCountry	firstName
▶	UK	Elizabeth

Multiple OUT

Name: getCustomerById

The name of the routine is parsed automatically fro
statement. The DDL is parsed automatically while

DDL:

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `getCustomerById`(
2      IN id INT,
3      OUT fname VARCHAR(25),
4      OUT city VARCHAR(25)
5  )
6  BEGIN
7  SELECT customer.FirstName, customer.City
8   INTO fname, city
9  FROM customer
10 WHERE customer.Id = id;
11 END
```

Query 1 | getCustomerById - Routine | errorHandling - Routine

Limit to 1000 rows

```
1  Call getCustomerById(6,@firstName,@city);
2  SELECT @firstName as firstName, @city as City;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

firstName	City
Hanna	Mannheim

Exception Handling Example

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `errorHandling`(
   IN customerID INT
)
BEGIN
   // Exit if the duplicate key occurs
   DECLARE EXIT HANDLER FOR 1062
   BEGIN
         SELECT CONCAT('Duplicate key (',customerID,') occurred') AS message;
   END;

   // Insert a new row into the SupplierProducts
   INSERT INTO customer(Id)
   VALUES(customerID);

   // Return the products supplied by the supplier id
   SELECT COUNT(*)
   FROM customer
   WHERE Id = customerID;

END
```

Query 1 | getCustomerById - Routine | errorHandling - Routine | errorHandling2 - Routine

Limit to 1000 rows

```
1  Call getCustomerById(6,@firstName,@city);
2  SELECT @firstName as firstName, @city as City;
3
4  call errorHandling(20);
5  call errorHandling(94);
6  select * from customer;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Id	FirstName	LastName	City	Country	Phone
84	Mary	Saveley	Lyon	France	78.32.54.86
85	Paul	Henriot	Reims	France	26.47.15.10
86	Rita	Müller	Stuttgart	Germany	0711-020361
87	Pirkko	Koskitalo	Oulu	Finland	981-443655
88	Paula	Parente	Resende	Brazil	(14) 555-8122
89	Karl	Jablonski	Seattle	USA	(206) 555-4112
90	Matti	Karttunen	Helsinki	Finland	90-224 8858
91	Zbyszek	Piestrzen...	Warszawa	Poland	(26) 642-7012
92	Craig	Smith	NULL	NULL	NULL
94	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

customer 15

Output

Action Output

#	Time	Action
40	11:44:05	Apply changes to errorHandling2
41	11:45:20	Apply changes to errorHandling
42	11:45:50	Apply changes to errorHandling
43	11:46:22	call errorHandling(20)
44	11:46:46	call errorHandling(94)
45	11:46:59	select * from customer LIMIT 0, 1000

CURSORS

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `createEmailList`(
	INOUT emailList varchar(4000)
)
BEGIN
	DECLARE finished INTEGER DEFAULT 0;
	DECLARE emailAddress varchar(100) DEFAULT "";

	DEClARE curEmail
		CURSOR FOR
			SELECT empEmail FROM employees;

	DECLARE CONTINUE HANDLER
      FOR NOT FOUND SET finished = 1;

	SET emailList = "";
	OPEN curEmail;

	getEmail: LOOP
		FETCH curEmail INTO emailAddress;
		IF finished = 1 THEN
			LEAVE getEmail;
		END IF;
		SET emailList = CONCAT(emailAddress,";",emailList);
	END LOOP getEmail;
	CLOSE curEmail;
END
```

Functions

```
CREATE DEFINER=`root`@`localhost` FUNCTION `getCount`() RETURNS int
    READS SQL DATA
    DETERMINISTIC
BEGIN

declare numberOfRecords int default 0;
select count(id) as count_of_records into numberOfRecords from customer;
RETURN numberOfRecords;

END
```