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.

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

WHILE LOOP:

END

Output:

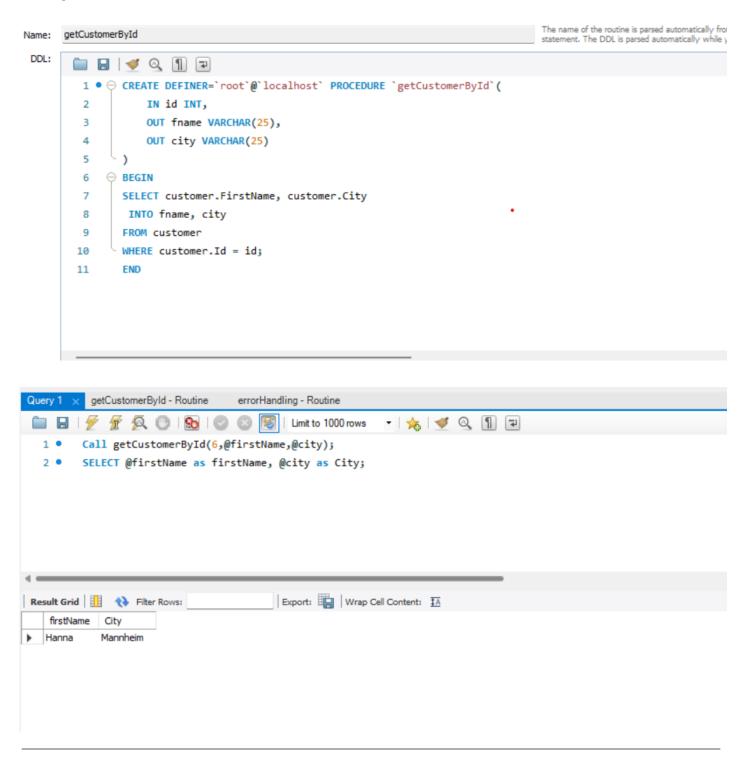


Switch Case:

```
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;
        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

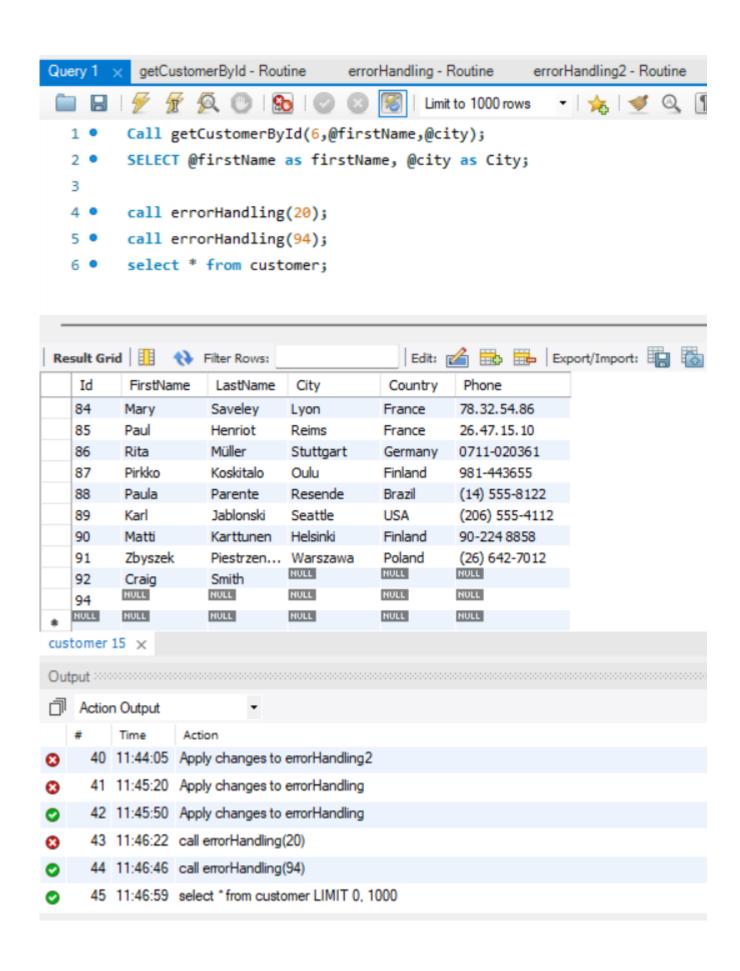
Multiple OUT



Exception Handling Example

END

```
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;
```



CURSORS

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