

**Aim:** Demonstrate the use of PL/SQL Exceptions and Records.

**Objectives:**

- To work with Exceptional handling of PL/SQL.
- To understand the concepts of Exception Handling.

**Tools Used:** MySQL Workbench

**Concepts:** Exceptions are abnormal conditions that can occur during the execution of the program. When an error occurs inside a stored procedure, it is important to handle it appropriately, such as continuing or exiting the current code block's execution and issuing a meaningful error message. MySQL provides an easy way to define handlers that handle from general conditions such as warnings or exceptions to specific conditions e.g., specific error codes.

**Exceptions are of two types:**

- **System Defined Exception** – Any issue with the code the issue is shown by default.
- **User Defined Exception.** - User defines when the issue has to be shown.

**Exception Syntax:**

DECLARE

<declarations section> BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

WHEN exception1 THEN exception1-handling-statements

WHEN exception2 THEN exception2-handling-statements

WHEN exception3 THEN exception3-handling-statements

.....

WHEN others THEN exception3-handling-statements

END;

**Assignment on Exception Handling****Scenario (For Question 1 and 2)**

Create a database college. Create a table student (rollno, name)

Insert 4 to 5 values in student table

```
create database college2;
use college2;
create table student (rollno int , name varchar(20));
insert into student values ('1' ,"Shradhha"),('2' ,"Shreya"),('3' ,"Sejal"),('4' ,"Shruti"),('5'
,"Saniya");
select * from student;
```

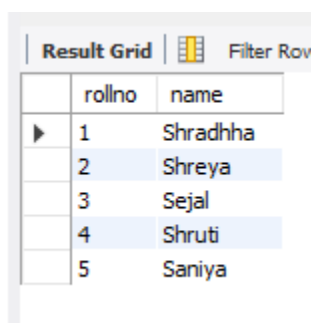
**Problem Statement :**

**1) Write a procedure which will handle the exception for selecting a data from test table (which is not present in college database) and selecting a data from student table (which is present in the college database)**

**Solution :**

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `exception1`()
BEGIN
DECLARE CONTINUE HANDLER FOR 1146
SELECT 'PLEASE CREATE THE TABLE FIRST AS IT DOES NOT EXIST' MESSAGE;
SELECT * FROM TEST;
SELECT * FROM student;
END
```

call exception1();



	rollno	name
▶	1	Shradhha
	2	Shreya
	3	Sejal
	4	Shruti
	5	Saniya

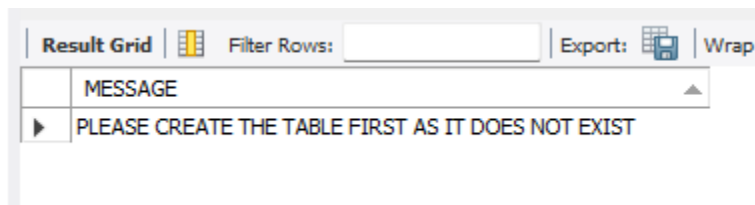
**Problem Statement :**

2) Write a procedure which will handle the exception for selecting a data from test table (which is not present in college database) and selecting a data from student table (which is present in the college database)

**Solution :**

```
CREATE DEFINER='root'@'localhost' PROCEDURE `exception2`()
BEGIN
DECLARE EXIT HANDLER FOR 1146
SELECT 'PLEASE CREATE THE TABLE FIRST AS IT DOES NOT EXIST' MESSAGE;
SELECT * FROM TEST;
SELECT * FROM student;
END
```

call exception2();

**Scenario (for Question 3 and 4)**

**Create a database Flipkart. Create a table SupplierProducts(supplierId, productId)**  
Make supplierId and productId a combined primary key.

```
Create database Flipkart;
USE Flipkart;
Create table SupplierProducts(supplierId int , productId int , constraint
pk_supplierProductID primary key (supplierId,productId)) ;
```

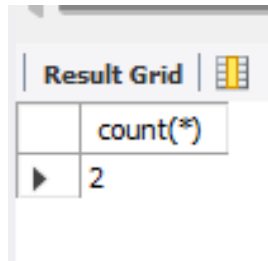
**Problem Statement :**

3) Write a procedure which will insert the value in SupplierProducts table if the value inserted are new, throw an exception for duplicate value insertion. And also show the count of rows.

**Solution :**

```
CREATE DEFINER='root'@'localhost' PROCEDURE `exception3`(in sid int, in pid int)
BEGIN
DECLARE CONTINUE HANDLER FOR 1062
Select 'DUPLICATE VALUE HAS INSERTED' MESSAGE;
INSERT INTO SupplierProducts( supplierId, productId) values (sid, pid);
select count(*) from SupplierProducts where supplierId=sid;
END
```

```
call exception3(1, 101);
call exception3(2, 201);
call exception3(2, 301);
```



	count(*)
▶	2

**Problem Statement :**

**4) Write a procedure which will insert the value in SupplierProducts table if the value inserted are new, throw an exception for duplicate value insertion.**

**Solution :**

```
CREATE DEFINER='root'@'localhost' PROCEDURE `exception3`(in sid int, in pid int)
BEGIN
DECLARE EXIT HANDLER FOR 1062
Select 'DUPLICATE VALUE HAS INSERTED' MESSAGE;
INSERT INTO SupplierProducts( supplierId, productId) values (sid, pid);
select count(*) from SupplierProducts where supplierId=sid;
END
```

```
call exception4(1, 101);
call exception4(2, 201);
```

call exception4(2, 301);

Result Grid		Filter Rows:
	MESSAGE	
▶	DUPLICATE VALUE HAS INSERTED	

**Observation :**

In this practical , I understand how to use the exception handling with the help of stored procedures. And I also learnt the difference between exit and continue actions which helps us to handle exceptions efficiently.