Aim: SQL Exception

Objectives:

The objectives for learning MySQL exceptions are to understand how exceptions handle errors and unexpected situations during database operations. Exceptions allow developers to define custom error messages and control the flow of execution when something goes wrong, such as during invalid data input or constraint violations. The goal is to learn how to manage errors effectively, ensuring that the database remains stable and providing useful feedback to users or applications.

Tools Used:

- MySQL Workbench

Concept:

Exceptions in SQL are mechanisms used to handle errors that occur during database operations. When an error or unexpected condition arises, exceptions allow the database to catch the error, display a custom message, or perform specific actions, such as rolling back changes. This helps ensure that the database remains stable, consistent, and provides meaningful feedback without disrupting the normal flow of operations. Exceptions are crucial for managing errors like constraint violations, invalid data, or failed transactions.

Example:

<u>Continue Exception handling</u>

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `exceptionHandling1`()
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
```

Problem Statement:

Scenario (For Question 1 and 2)

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

Insert 4 to 5 values in student table

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)

Hint: Use continue statement and observe the output.

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)

Hint: Use exit statement and observe the output.

Scenario (for Question 3 and 4)

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

3) Write a procedure which will insert the value in SupplierProducts table if then value inserted are new, throw an exception for duplicate value insertion. And also show the count of rows. Hint: Use continue statement and observe the output.

4) Write a procedure which will insert the value in SupplierProducts table if then value inserted are new, throw an exception for duplicate value insertion. Hint: Use exit statement and observe the output

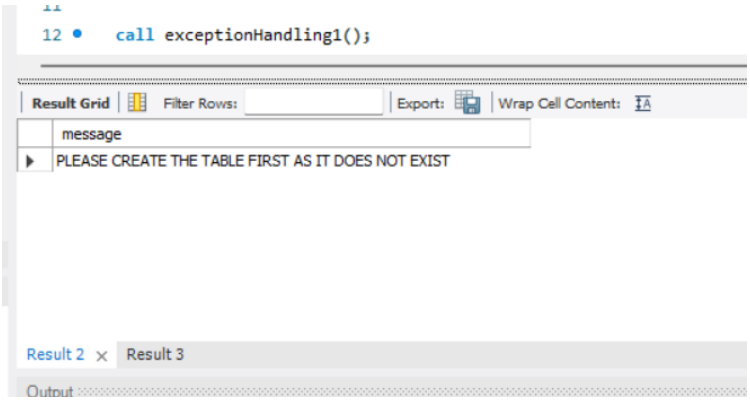
Solution:

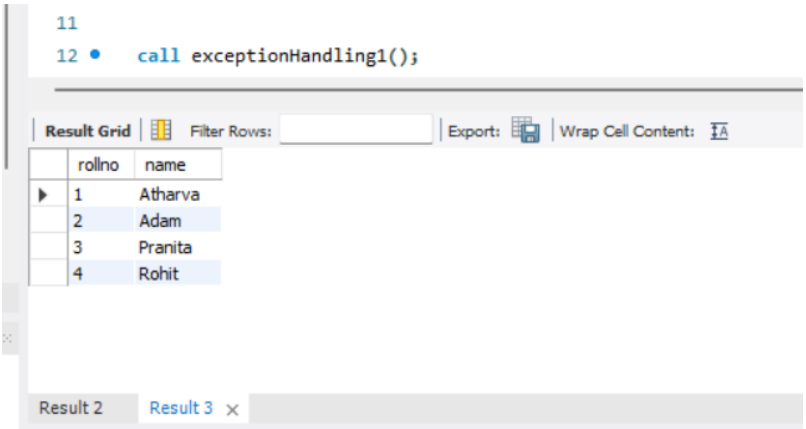
Initial Table Creation:

```
create database College;
use College;
create table student (rollno int, name varchar(50));
select * from student;
insert into student values
(1,'Atharva'),
(2,'Adam'),
(3,'Pranita'),
(4,'Rohit');
```

1)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `exceptionHandling1`()
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
```





2)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `exceptionHandling2`()
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
```

```
13 •  call exceptionHandling2();
14
```

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

message
PLEASE CREATE THE TABLE FIRST AS IT DOES NOT EXIST

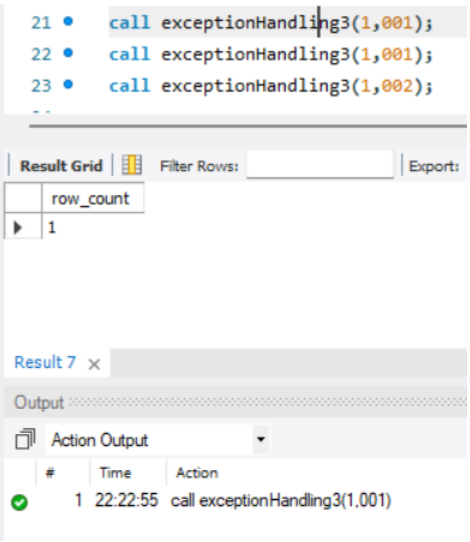
Result 4

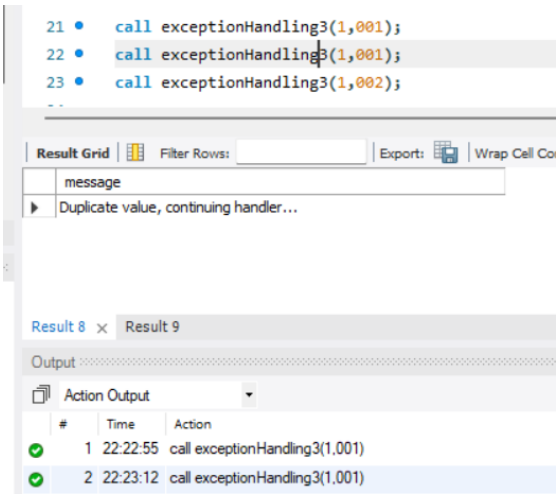
Initial Table Creation:

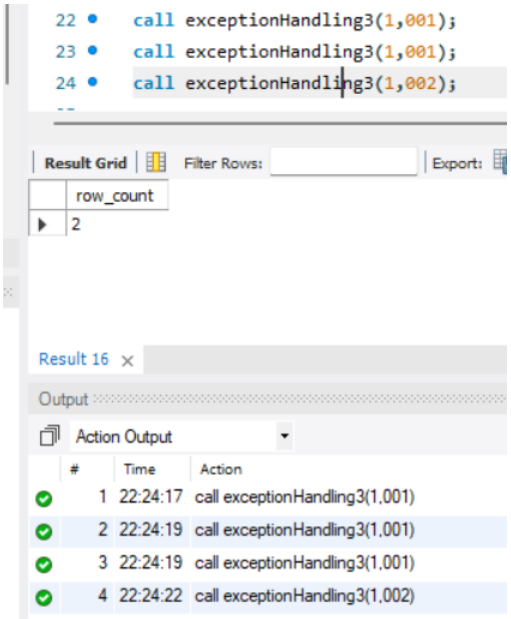
```
create database Flipkart;
use Flipkart;
create table SupplierProducts (supplierId int , productId int, primary key(supplierId,productId));
select * from SupplierProducts;
```

3)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `exceptionHandling3`( in suppId int, in prodId int)
BEGIN
declare continue handler for 1062
select 'Duplicate value, continuing handler...' message;
insert into SupplierProducts value (suppId,prodId);
select count(*) as row_count from SupplierProducts;
END
```

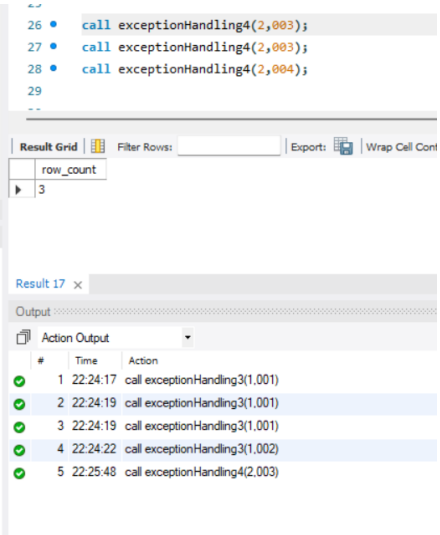


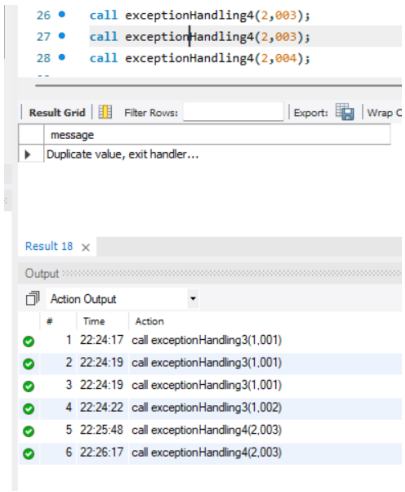


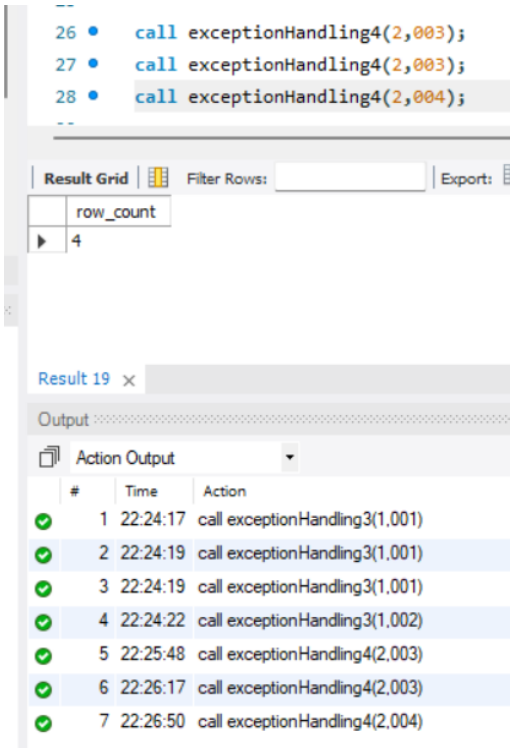


4)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `exceptionHandling4`( in suppId int, in prodId int)
BEGIN
declare exit handler for 1062
select 'Duplicate value, exit handler...' message;
insert into SupplierProducts value (suppId,prodId);
select count(*) as row_count from SupplierProducts
END
```







Observation:

Exception handling in SQL is a vital feature that allows the database to manage errors and unexpected situations during operations. It enables the system to catch and handle errors, ensuring smooth execution by providing meaningful error messages, rolling back transactions, or performing alternative actions. This helps maintain data integrity and prevents system failures by addressing issues like constraint violations or invalid data inputs.