In **PL/SQL**, **exception handling** is used to **gracefully manage runtime errors**. If an error occurs in the BEGIN block, control is passed to the EXCEPTION block, allowing you to **handle the error without crashing** the program.

**🔹 Structure of PL/SQL with Exception Handling (oracle syntax)**

BEGIN

-- Executable statements

EXCEPTION

WHEN exception\_name THEN

-- Exception handling statements

WHEN OTHERS THEN

-- Handle all other exceptions

END;

## 🔹 ****Types of Exceptions****

1. **Predefined exceptions** (automatically raised by PL/SQL):
   * NO\_DATA\_FOUND
   * ZERO\_DIVIDE
   * TOO\_MANY\_ROWS
   * INVALID\_NUMBER
   * DUP\_VAL\_ON\_INDEX, etc.
2. **User-defined exceptions** (declared by the user using EXCEPTION keyword).

## 🔹 ****Example: Predefined Exception (****NO\_DATA\_FOUND****)****

sql

CopyEdit

DECLARE

v\_emp\_name employees.emp\_name%TYPE;

BEGIN

SELECT emp\_name INTO v\_emp\_name

FROM employees

WHERE emp\_id = 999; -- assume this ID doesn't exist

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_emp\_name);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employee found with that ID');

END;

🧠 Here, if the employee ID doesn't exist, NO\_DATA\_FOUND is triggered, and the program doesn’t crash.

## 🔹 ****Example: Handling Division by Zero (****ZERO\_DIVIDE****)****

sql

CopyEdit

DECLARE

num NUMBER := 10;

denom NUMBER := 0;

result NUMBER;

BEGIN

result := num / denom;

DBMS\_OUTPUT.PUT\_LINE('Result: ' || result);

EXCEPTION

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Cannot divide by zero');

END;

## 🔹 ****Example: User-Defined Exception****

sql

CopyEdit

DECLARE

insufficient\_balance EXCEPTION; -- user-defined exception

balance NUMBER := 500;

withdraw\_amount NUMBER := 1000;

BEGIN

IF withdraw\_amount > balance THEN

RAISE insufficient\_balance;

ELSE

balance := balance - withdraw\_amount;

DBMS\_OUTPUT.PUT\_LINE('Withdrawal successful. Balance: ' || balance);

END IF;

EXCEPTION

WHEN insufficient\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient balance');

END;

## 🔹 ****WHEN OTHERS Clause (Catch-All)****

sql

CopyEdit

BEGIN

-- Some code that may raise an exception

EXCEPTION

WHEN OTHERS THEN

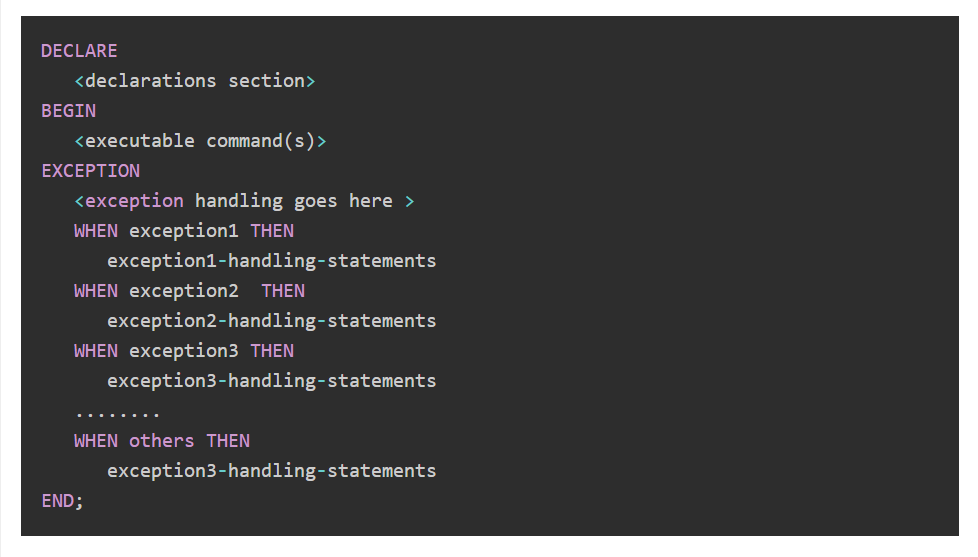
DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

END;

🧠 SQLERRM returns the error message of the exception.

## Syntax for Exception Handling

The general syntax for exception handling is as follows. Here you can list down as many exceptions as you can handle. The default exception will be handled using ***WHEN others THEN*** −



MYSQL-

### ✅ Exception Handling in **MySQL**

MySQL does **not** have full-fledged exception handling like Oracle’s PL/SQL, but it **does support error handling inside stored procedures** using:

* **DECLARE ... HANDLER**
* **SIGNAL** (to raise errors manually)

## 🔹 Syntax: Declaring Handlers in MySQL

sql

CopyEdit

DECLARE handler\_type HANDLER

FOR condition

statement;

### 🔸 handler\_type can be:

* CONTINUE – continue execution after the error
* EXIT – exit the block or procedure after the error
* UNDO – not supported in MySQL

### 🔸 condition can be:

* SQLEXCEPTION – handles all SQL errors
* SQLWARNING – handles warnings
* Specific error codes (e.g., 1062 for duplicate entry)

## 🧠 ****Example 1: Handling Division by Zero****

sql

CopyEdit

DELIMITER $$

CREATE PROCEDURE handle\_division()

BEGIN

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

BEGIN

SELECT 'An error occurred!' AS message;

END;

DECLARE result INT;

SET result = 10 / 0; -- will throw error

SELECT result;

END$$

DELIMITER ;

-- Call the procedure

CALL handle\_division();

🟡 **Explanation**:

* If 10 / 0 throws an error, control goes to the handler, and "An error occurred!" is shown.

## 🧠 ****Example 2: Handling Duplicate Entry (Error 1062)****

sql

CopyEdit

DELIMITER $$

CREATE PROCEDURE insert\_customer()

BEGIN

DECLARE CONTINUE HANDLER FOR 1062

BEGIN

SELECT 'Duplicate entry!' AS message;

END;

INSERT INTO customers(id, name) VALUES (1, 'John');

END$$

DELIMITER ;

CALL insert\_customer();

## 🔹 Raising Custom Errors: SIGNAL

You can manually **raise errors** using SIGNAL.

### 🔸 Example: Raise custom error if input is invalid

sql

CopyEdit

DELIMITER $$

CREATE PROCEDURE validate\_age(IN user\_age INT)

BEGIN

IF user\_age < 18 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Age must be 18 or above';

ELSE

SELECT 'Valid age' AS status;

END IF;

END$$

DELIMITER ;

CALL validate\_age(16); -- This will raise the error

## 🔸 Summary: Exception Handling in MySQL

| **Feature** | **Supported** |
| --- | --- |
| DECLARE HANDLER | ✅ Yes |
| SIGNAL | ✅ Yes (MySQL 5.5+) |
| Specific exception types | ✅ Yes (via error codes or conditions) |
| TRY...CATCH blocks | ❌ No |
| Nested exception blocks | ❌ No |

## 🔹 ****Frequently Used Error Codes****

| **Error Code** | **SQLSTATE** | **Meaning** | **Use in HANDLER / SIGNAL** |
| --- | --- | --- | --- |
| 1062 | 23000 | Duplicate entry (violates unique) | DECLARE HANDLER FOR 1062 |
| 1048 | 23000 | Column cannot be null | DECLARE HANDLER FOR 1048 |
| 1054 | 42S22 | Unknown column in field list | DECLARE HANDLER FOR 1054 |
| 1146 | 42S02 | Table does not exist | DECLARE HANDLER FOR 1146 |
| 1216 | 23000 | Foreign key constraint fails | DECLARE HANDLER FOR 1216 |
| 1049 | 42000 | Unknown database | DECLARE HANDLER FOR 1049 |
| 1364 | HY000 | Field doesn't have a default value | DECLARE HANDLER FOR 1364 |
| 1045 | 28000 | Access denied for user | DECLARE HANDLER FOR 1045 |
| 1451 | 23000 | Cannot delete/update parent row | DECLARE HANDLER FOR 1451 |
| 1452 | 23000 | Cannot add/update child row | DECLARE HANDLER FOR 1452 |

## 🔹 Special Condition Handlers

| **Condition** | **Description** | **Use like this** |
| --- | --- | --- |
| SQLEXCEPTION | Any SQL error (runtime errors) | DECLARE EXIT HANDLER FOR SQLEXCEPTION |
| SQLWARNING | Any warning (non-fatal errors) | DECLARE CONTINUE HANDLER FOR SQLWARNING |
| NOT FOUND | No rows found (useful in cursors) | DECLARE CONTINUE HANDLER FOR NOT FOUND |

## 🔹 Example: Using Error Code in Handler

sql

CopyEdit

DELIMITER $$

CREATE PROCEDURE insert\_customer()

BEGIN

DECLARE CONTINUE HANDLER FOR 1062

BEGIN

SELECT 'Duplicate entry error!' AS message;

END;

INSERT INTO customers(id, name) VALUES (1, 'Alice');

END$$

DELIMITER ;