An error condition during a program execution is called an exception in PL/SQL. PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition. There are two types of exceptions:

- System-defined exceptions
- User-defined exceptions

### Syntax for Exception Handling

The General Syntax for exception handling is as follows. Here you can list down as many as exceptions you want to handle. The default exception will be handled using *WHEN others THEN*:

```sql
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;
```

### Example

Let us write some simple code to illustrate the concept. We will be using the CUSTOMERS table we had created and used in the previous chapters:

```sql
DECLARE
    c_id customers.id%type := 8;
    c_name customers.name%type;
    c_addr customers.address%type;
BEGIN
    SELECT  name, address INTO  c_name, c_addr
    FROM customers
    WHERE id = c_id;
    DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
    DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('No such customer!');
    WHEN others THEN
        dbms_output.put_line('Error!');
END;
```

When the above code is executed at SQL prompt, it produces the following result:

```
No such customer!
```

**PL/SQL procedure successfully completed.**

The above program displays the name and address of a customer whose ID is given. Since there is
no customer with ID value 8 in our database, the program raises the run-time exception
**NO_DATA_FOUND**, which is captured in **EXCEPTION** block.

### Raising Exceptions

Exceptions are raised by the database server automatically whenever there is any internal
database error, but exceptions can be raised explicitly by the programmer by using the command
**RAISE**. Following is the simple syntax of raising an exception:

```plsql
DECLARE
    exception_name EXCEPTION;
BEGIN
    IF condition THEN
        RAISE exception_name;
    END IF;
EXCEPTION
    WHEN exception_name THEN
        statement;
END;
```

You can use above syntax in raising Oracle standard exception or any user-defined exception.
Next section will give you an example on raising user-defined exception, similar way you can raise
Oracle standard exceptions as well.

### User-defined Exceptions

PL/SQL allows you to define your own exceptions according to the need of your program. A user-defined exception must be declared and then raised explicitly, using either a **RAISE** statement or
the procedure **DBMS_STANDARD.RAISE_APPLICATION_ERROR**.

The syntax for declaring an exception is:

```plsql
DECLARE
    my_exception EXCEPTION;
```

### Example:

The following example illustrates the concept. This program asks for a customer ID, when the user
enters an invalid ID, the exception **invalid_id** is raised.

```plsql
DECLARE
    c_id customers.id%type := &cc_id;
    c_name customers.name%type;
    c_addr customers.address%type;
    -- user defined exception
    ex_invalid_id EXCEPTION;
BEGIN
    IF c_id <= 0 THEN
        RAISE ex_invalid_id;
    ELSE
        SELECT name, address INTO c_name, c_addr
        FROM customers
        WHERE id = c_id;
        DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
        DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
    END IF;
EXCEPTION
    WHEN ex_invalid_id THEN
        dbms_output.put_line('ID must be greater than zero!');
    WHEN no_data_found THEN
        dbms_output.put_line('No such customer!');
    WHEN others THEN
        dbms_output.put_line('Error!');
END;
```
When the above code is executed at SQL prompt, it produces the following result:

```
Enter value for cc_id: -6 (let's enter a value -6)
old  2: c_id customers.id%type := &cc_id;
new  2: c_id customers.id%type := -6;
ID must be greater than zero!
PL/SQL procedure successfully completed.
```

### Pre-defined Exceptions

PL/SQL provides many pre-defined exceptions, which are executed when any database rule is violated by a program. For example, the predefined exception NO_DATA_FOUND is raised when a SELECT INTO statement returns no rows. The following table lists few of the important pre-defined exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Oracle Error</th>
<th>SQLCODE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS INTO NULL</td>
<td>06530</td>
<td>-6530</td>
<td>It is raised when a null object is automatically assigned a value.</td>
</tr>
<tr>
<td>CASE NOT FOUND</td>
<td>06592</td>
<td>-6592</td>
<td>It is raised when none of the choices in the WHEN clauses of a CASE statement is selected, and there is no ELSE clause.</td>
</tr>
<tr>
<td>COLLECTION IS NULL</td>
<td>06531</td>
<td>-6531</td>
<td>It is raised when a program attempts to apply collection methods other than EXISTS to an uninitialized nested table or varray, or the program attempts to assign values to the elements of an uninitialized nested table or varray.</td>
</tr>
<tr>
<td>DUP_VAL ON INDEX</td>
<td>00001</td>
<td>-1</td>
<td>It is raised when duplicate values are attempted to be stored in a column with unique index.</td>
</tr>
<tr>
<td>INVALID_CURSOR</td>
<td>01001</td>
<td>-1001</td>
<td>It is raised when attempts are made to make a cursor operation that is not allowed, such as closing an unopened cursor.</td>
</tr>
<tr>
<td>INVALID NUMBER</td>
<td>01722</td>
<td>-1722</td>
<td>It is raised when the conversion of a character string into a number fails because the string does not represent a valid number.</td>
</tr>
<tr>
<td>LOGIN DENIED</td>
<td>01017</td>
<td>-1017</td>
<td>It is raised when a program attempts to log on to the database with an invalid username or password.</td>
</tr>
<tr>
<td>NO DATA FOUND</td>
<td>01403</td>
<td>+100</td>
<td>It is raised when a SELECT INTO statement returns no rows.</td>
</tr>
<tr>
<td>NOT LOGGED ON</td>
<td>01012</td>
<td>-1012</td>
<td>It is raised when a database call is issued without being connected to the database.</td>
</tr>
<tr>
<td>PROGRAM ERROR</td>
<td>06501</td>
<td>-6501</td>
<td>It is raised when PL/SQL has an internal problem.</td>
</tr>
<tr>
<td>ROWTYPE MISMATCH</td>
<td>06504</td>
<td>-6504</td>
<td>It is raised when a cursor fetches value in a variable having incompatible data type.</td>
</tr>
<tr>
<td>SELF IS NULL</td>
<td>30625</td>
<td>-30625</td>
<td>It is raised when a member method is invoked, but the instance of the object type was not initialized.</td>
</tr>
<tr>
<td>Error Type</td>
<td>Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>STORAGE_ERROR</td>
<td>06500</td>
<td>It is raised when PL/SQL ran out of memory or memory was corrupted.</td>
<td></td>
</tr>
<tr>
<td>TOO_MANY_ROWS</td>
<td>01422</td>
<td>It is raised when a SELECT INTO statement returns more than one row.</td>
<td></td>
</tr>
<tr>
<td>VALUE_ERROR</td>
<td>06502</td>
<td>It is raised when an arithmetic, conversion, truncation, or size-constraint error occurs.</td>
<td></td>
</tr>
<tr>
<td>ZERO_DIVIDE</td>
<td>01476</td>
<td>It is raised when an attempt is made to divide a number by zero.</td>
<td></td>
</tr>
</tbody>
</table>