DECLARE

> v\_lname VARCHAR2 (15);

> BEGIN

> SELECT last\_name INTO v\_lname

> FROM employees

> WHERE first\_name = 'John';

> DBMS\_OUTPUT.PUT\_LINE ('Last name is :' || v\_lname);

> END;

8507: ORA-01422: exact fetch returns more than requested number of rows

8507: ORA-06512: at line 4

The command failed.

You can handle such exceptions in your PL/SQL block so that your program completes successfully. For example:

Command> DECLARE

> v\_lname VARCHAR2 (15);

> BEGIN

> SELECT last\_name INTO v\_lname

> FROM employees

> WHERE first\_name = 'John';

> DBMS\_OUTPUT.PUT\_LINE ('Last name is :' || v\_lname);

> EXCEPTION

> WHEN TOO\_MANY\_ROWS THEN

> DBMS\_OUTPUT.PUT\_LINE (' Your SELECT statement retrieved multiple

> rows. Consider using a cursor.');

> END;

> /

Your SELECT statement retrieved multiple rows. Consider using a cursor.

PL/SQL procedure successfully completed.

-=-=-=-

***Using the ZERO\_DIVIDE predefined exception***

In this example, a PL/SQL program attempts to divide by 0. The ZERO\_DIVIDE predefined exception is used to trap the error in an exception-handling routine.

Command> DECLARE v\_invalid PLS\_INTEGER;

> BEGIN

> v\_invalid := 100/0;

> EXCEPTION

> WHEN ZERO\_DIVIDE THEN

> DBMS\_OUTPUT.PUT\_LINE ('Attempt to divide by 0');

> END;

> /

Attempt to divide by 0

PL/SQL procedure successfully completed.

=-=-=-=-=-=

***Example 4-2 Using RAISE statement to trap user-defined exception***

In this example, the department number 500 does not exist, so no rows are updated in the departments table. The RAISE statement is used to explicitly raise an exception and display an error message, returned by the SQLERRM built-in function, and an error code, returned by the SQLCODE built-in function. Use the RAISE statement by itself within an exception handler to raise the same exception again and propagate it back to the calling environment.

Command> DECLARE

> v\_deptno NUMBER := 500;

> v\_name VARCHAR2 (20) := 'Testing';

> e\_invalid\_dept EXCEPTION;

> BEGIN

> UPDATE departments

> SET department\_name = v\_name

> WHERE department\_id = v\_deptno;

> IF SQL%NOTFOUND THEN

> RAISE e\_invalid\_dept;

> END IF;

> ROLLBACK;

> EXCEPTION

> WHEN e\_invalid\_dept THEN

> DBMS\_OUTPUT.PUT\_LINE ('No such department');

> DBMS\_OUTPUT.PUT\_LINE (SQLERRM);

> DBMS\_OUTPUT.PUT\_LINE (SQLCODE);

> END;

> /

No such department

User-Defined Exception

1

PL/SQL procedure successfully completed.

The command succeeded.

—=-=-=-=-=-

***Example 4-3 Using the RAISE\_APPLICATION\_ERROR procedure***

This example attempts to delete from the employees table where last\_name=Patterson. The RAISE\_APPLICATION\_ERROR procedure raises the error, using error number -20201.

Command> DECLARE

> v\_last\_name employees.last\_name%TYPE := 'Patterson';

> BEGIN

> DELETE FROM employees WHERE last\_name = v\_last\_name;

> IF SQL%NOTFOUND THEN

> RAISE\_APPLICATION\_ERROR (-20201, v\_last\_name || ' does not exist');

> END IF;

> END;

> /

8507: ORA-20201: Patterson does not exist

8507: ORA-06512: at line 6

The command failed.

——===========-=-=-=-=-=-=-=-=-

Below are the few predefined exceptions

**Exception**

**Error Code**

**Exception Reason**

ACCESS\_INTO\_NULL

ORA-06530

Assign a value to the attributes of uninitialized objects

CASE\_NOT\_FOUND

ORA-06592

None of the 'WHEN' clause in CASE statement satisfied and no 'ELSE' clause is specified

COLLECTION\_IS\_NULL

ORA-06531

Using collection methods (except EXISTS) or accessing collection attributes on a uninitialized collections

CURSOR\_ALREADY\_OPEN

ORA-06511

Trying to open a cursor which is already opened

DUP\_VAL\_ON\_INDEX

ORA-00001

Storing a duplicate value in a database column that is a constrained by unique index

INVALID\_CURSOR

ORA-01001

Illegal cursor operations like closing an unopened cursor

INVALID\_NUMBER

ORA-01722

Conversion of character to a number failed due to invalid number character

NO\_DATA\_FOUND

ORA-01403

When 'SELECT' statement that contains INTO clause fetches no rows.

ROW\_MISMATCH

ORA-06504

When cursor variable data type is incompatible with the actual cursor return type

SUBSCRIPT\_BEYOND\_COUNT

ORA-06533

Referring collection by an index number that is larger than the collection size

SUBSCRIPT\_OUTSIDE\_LIMIT

ORA-06532

Referring collection by an index number that is outside the legal range (eg: -1)

TOO\_MANY\_ROWS

ORA-01422

When a 'SELECT' statement with INTO clause returns more than one row

VALUE\_ERROR

ORA-06502

Arithmetic or size constraint error (eg: assigning a value to a variable that is larger than the variable size)

ZERO\_DIVIDE

ORA-01476

Dividing a number by '0'

DECLARE

Sample\_exception EXCEPTION;

PROCEDURE nested\_block

IS

BEGIN

Dbms\_output.put\_line(‘Inside nested block’);

Dbms\_output.put\_line(‘Raising sample\_exception from nested block’);

RAISE sample\_exception;

EXCEPTION

WHEN sample\_exception THEN

Dbms\_output.put\_line (‘Exception captured in nested block. Raising to main block’);

RAISE,

END;

BEGIN

Dbms\_output.put\_line(‘Inside main block’);

Dbms\_output.put\_line(‘Calling nested block’);

Nested\_block;

EXCEPTION

WHEN sample\_exception THEN

Dbms\_output.put\_line (‘Exception captured in main block');

END:

/

**Code Explanation:**

**• Code line 2**: Declaring the variable 'sample\_exception' as EXCEPTION type.

**• Code line 3**: Declaring procedure nested\_block.

**• Code line 6**: Printing the statement "Inside nested block".

**• Code line 7:**Printing the statement "Raising sample\_exception from nested block."

**• Code line 8:**Raising the exception using 'RAISE sample\_exception'.

**• Code line 10:**Exception handler for exception sample\_exception in the nested block.

**• Code line 11:**Printing the statement 'Exception captured in nested block. Raising to main block'.

**• Code line 12:** Raising the exception to main block (propagating to the main block).

**• Code line 15:**Printing the statement "Inside the main block".

**• Code line 16:** Printing the statement "Calling nested block".

**• Code line 17:** Calling nested\_block procedure.

**• Code line 19:**Exception handler for sample\_exception in the main block.

**• Code line 20:** Printing the statement "Exception captured in the main block."