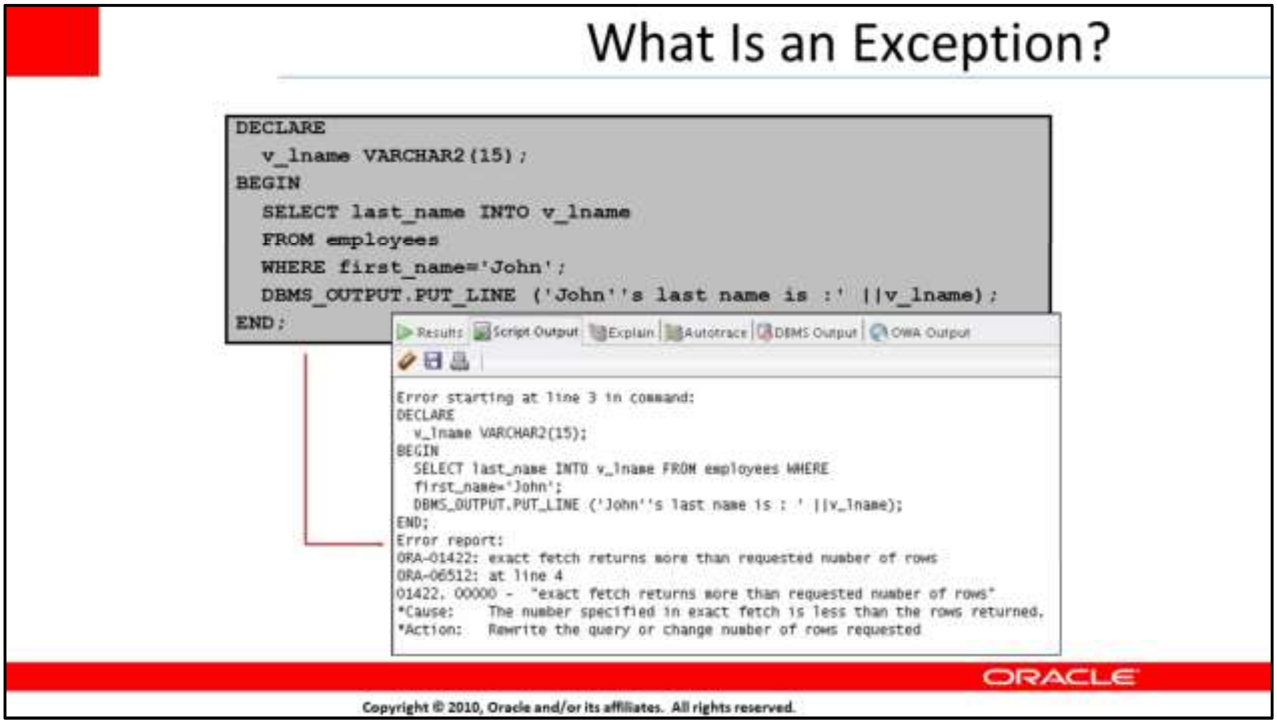


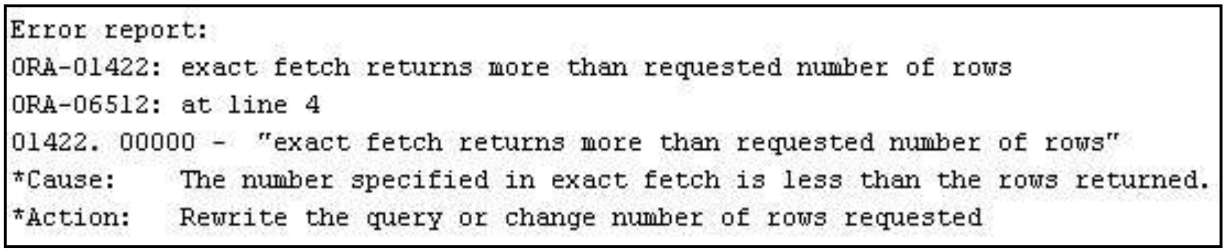
What Is an Exception?

```
DECLARE
  v_lname VARCHAR2(15);
BEGIN
  SELECT last_name INTO v_lname
  FROM employees
  WHERE first_name='John';
  DBMS_OUTPUT.PUT_LINE ('John''s last name is :' ||v_lname);
END;
```

```
Error starting at line 3 in command:
DECLARE
  v_lname VARCHAR2(15);
BEGIN
  SELECT last_name INTO v_lname FROM employees WHERE
  first_name='John';
  DBMS_OUTPUT.PUT_LINE ('John''s last name is : ' ||v_lname);
END;
Error report:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 4
01422. 00000 -  "exact fetch returns more than requested number of rows"
*Cause:    The number specified in exact fetch is less than the rows returned.
*Action:   Rewrite the query or change number of rows requested
```

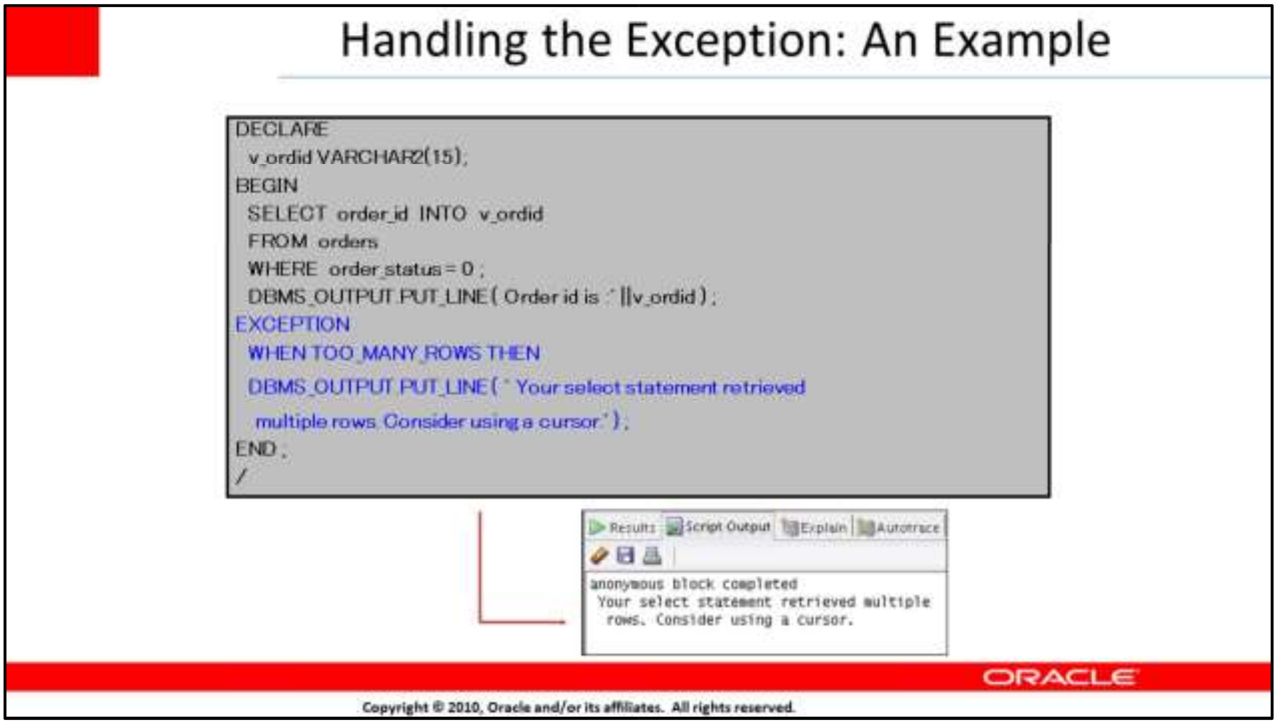
What Is an Exception?

Consider the example shown in the slide. There are no syntax errors in the code, which means that you must be able to successfully execute the anonymous block. The SELECT statement in the block retrieves the last name of John. However, you see the following error report when you execute the code:



```
Error report:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 4
01422. 00000 -  "exact fetch returns more than requested number of rows"
*Cause:    The number specified in exact fetch is less than the rows returned.
*Action:   Rewrite the query or change number of rows requested
```

The code does not work as expected. You expected the SELECT statement to retrieve only one row; however, it retrieves multiple rows. Such errors that occur at run time are called exceptions. When an exception occurs, the PL/SQL block is terminated. You can handle such exceptions in your PL/SQL block.



Handling the Exception: An Example

```
DECLARE
 v_ordid VARCHAR2(15);
BEGIN
 SELECT order_id INTO v_ordid
 FROM orders
 WHERE order_status = 0 ;
 DBMS_OUTPUT.PUT_LINE( Order id is :' ||v_ordid ) ;
EXCEPTION
 WHEN TOO_MANY_ROWS THEN
 DBMS_OUTPUT.PUT_LINE( ' Your select statement retrieved
  multiple rows. Consider using a cursor.' ) ;
END ;
/
```

```
anonymous block completed
 Your select statement retrieved multiple
  rows. Consider using a cursor.
```

Handling the Exception: An Example

You have previously learned how to write PL/SQL blocks with a declarative section (beginning with the DECLARE keyword) and an executable section (beginning and ending with the BEGIN and END keywords, respectively).

For exception handling, you include another optional section called the exception section.

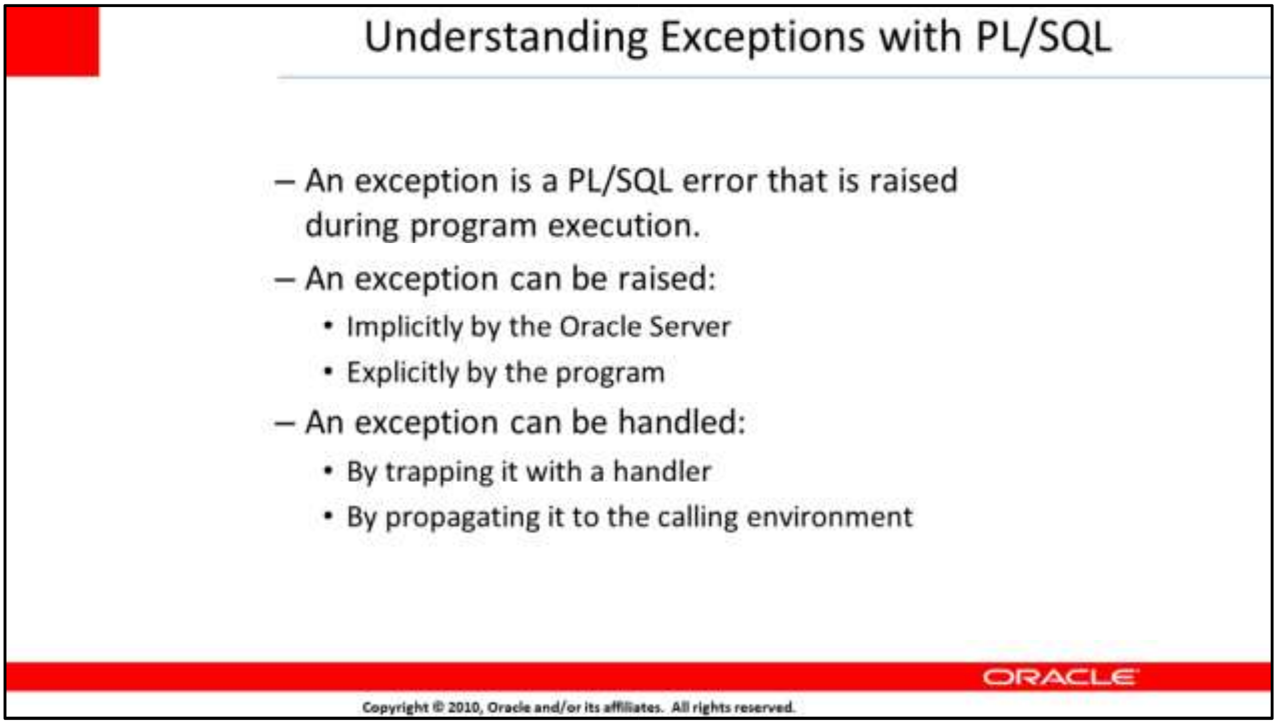
This section begins with the EXCEPTION keyword.

If present, this must be the last section in a PL/SQL block.

Example

In the example in the slide, the code from the previous slide is rewritten to handle the exception that occurred. The output of the code is shown in the slide as well.

By adding the EXCEPTION section of the code, the PL/SQL program does not terminate abruptly. When the exception is raised, the control shifts to the exception section and all the statements in the exception section are executed. The PL/SQL block terminates with normal, successful completion



Understanding Exceptions with PL/SQL

- An exception is a PL/SQL error that is raised during program execution.
- An exception can be raised:
  - Implicitly by the Oracle Server
  - Explicitly by the program
- An exception can be handled:
  - By trapping it with a handler
  - By propagating it to the calling environment

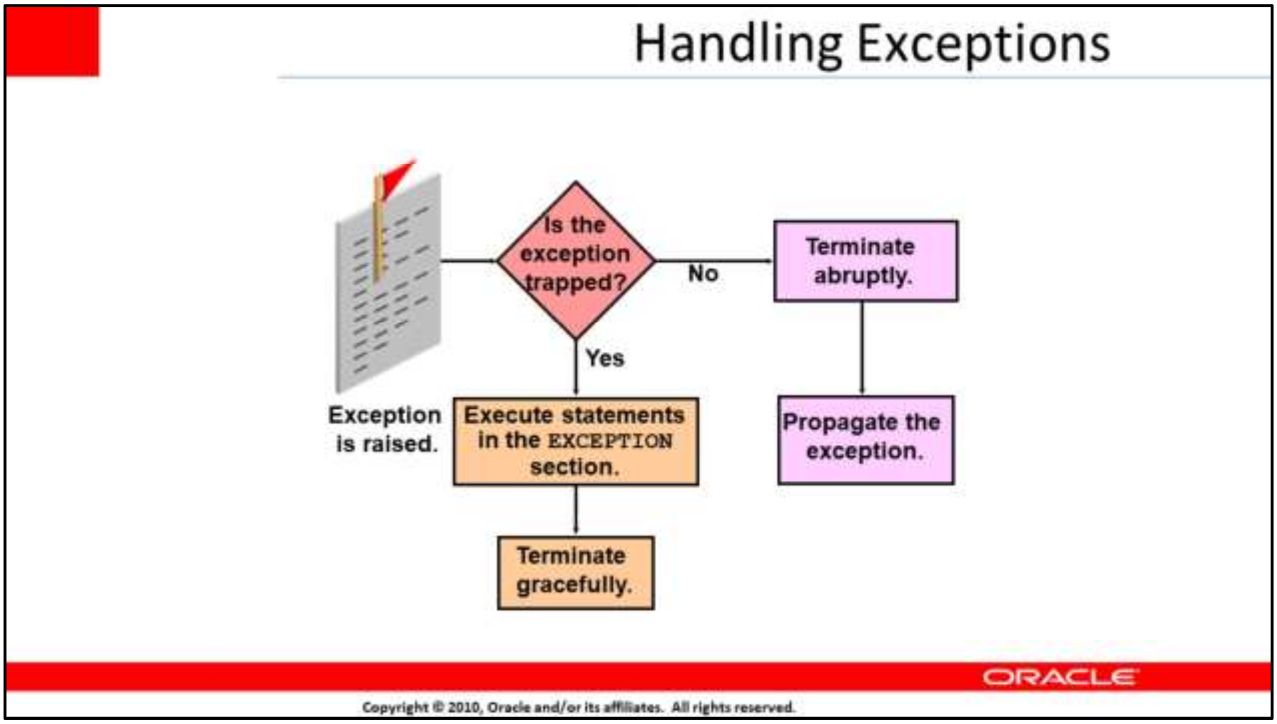
Understanding Exceptions with PL/SQL

An exception is an error in PL/SQL that is raised during the execution of a block. A block always terminates when PL/SQL raises an exception, but you can specify an exception handler to perform final actions before the block ends.

Two Methods for Raising an Exception

An Oracle error occurs and the associated exception is raised automatically. For example, if the ORA-01403 error occurs when no rows are retrieved from the database in a SELECT statement, PL/SQL raises the NO_DATA_FOUND exception. These errors are converted into predefined exceptions.

Depending on the business functionality your program implements, you may have to explicitly raise an exception. You raise an exception explicitly by issuing the RAISE statement in the block. The raised exception may be either user-defined or predefined. There are also some non-predefined Oracle errors. These errors are any standard Oracle errors that are not predefined. You can explicitly declare exceptions and associate them with the non-predefined Oracle errors.



Handling Exceptions

Exception is raised.

Is the exception trapped?

No

Terminate abruptly.

Propagate the exception.

Yes

Execute statements in the EXCEPTION section.

Terminate gracefully.

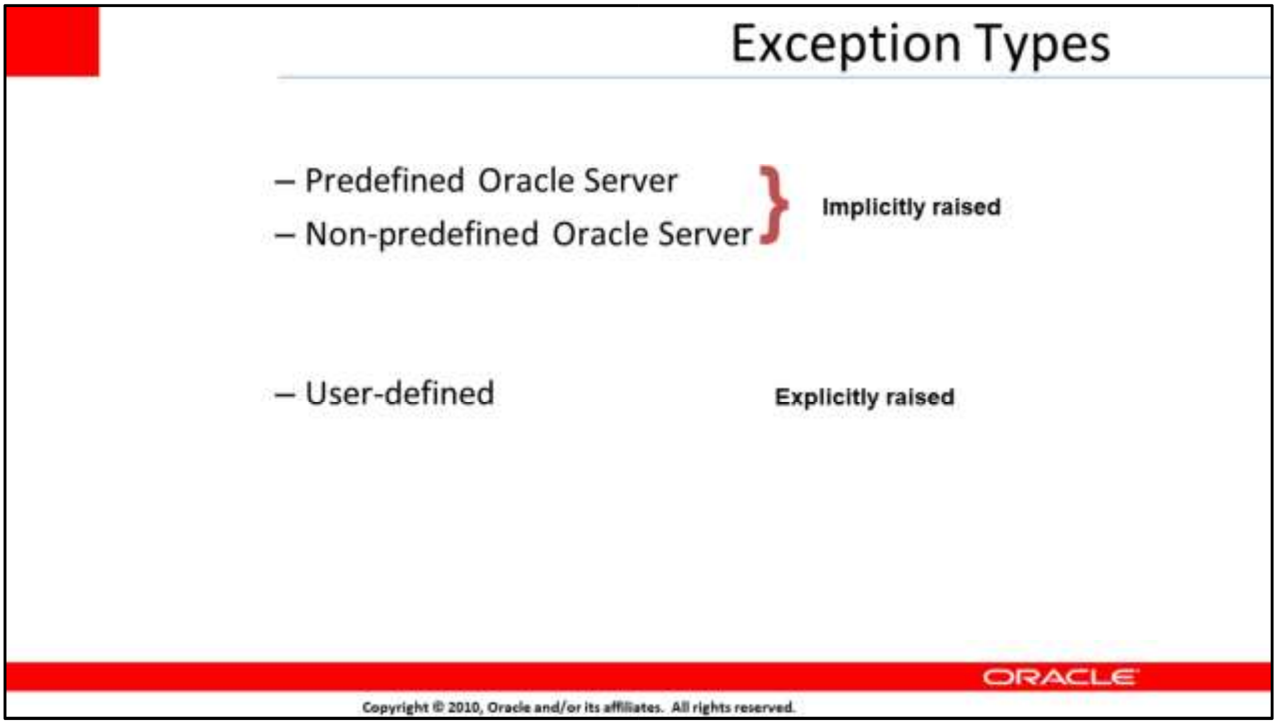
Handling Exceptions

Trapping an Exception

Include an EXCEPTION section in your PL/SQL program to trap exceptions. If the exception is raised in the executable section of the block, processing branches to the corresponding exception handler in the exception section of the block. If PL/SQL successfully handles the exception, the exception does not propagate to the enclosing block or to the calling environment. The PL/SQL block terminates successfully.

Propagating an Exception

If the exception is raised in the executable section of the block and there is no corresponding exception handler, the PL/SQL block terminates with failure and the exception is propagated to an enclosing block or to the calling environment. The calling environment can be any application (such as SQL*Plus that invokes the PL/SQL program).



Exception Types

- Predefined Oracle Server
- Non-predefined Oracle Server

Implicitly raised

- User-defined

Explicitly raised

Exception Types

There are three types of exceptions.

Exception	Description	Directions for Handling
Predefined Oracle Server error	One of approximately 20 errors that occur most often in PL/SQL code	You need not declare these exceptions. They are predefined by the Oracle server and are raised implicitly.
Non-predefined Oracle Server error	Any other standard Oracle Server error	You need to declare these within the declarative section; the Oracle server raises the error implicitly, and you can catch the error in the exception handler.
User-defined error	A condition that the developer determines is abnormal	You need to declare in the declarative section and raise explicitly.