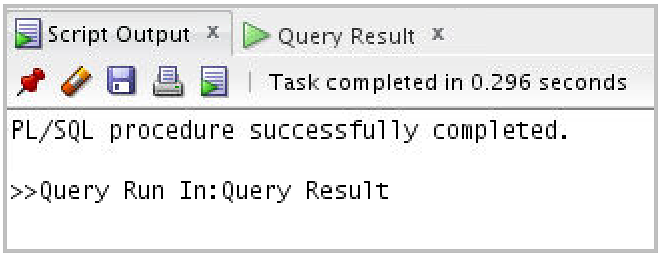
Practices for Lesson 9: Handling Exceptions

Chapter 9

Practice 9-1: Handling Predefined Exceptions

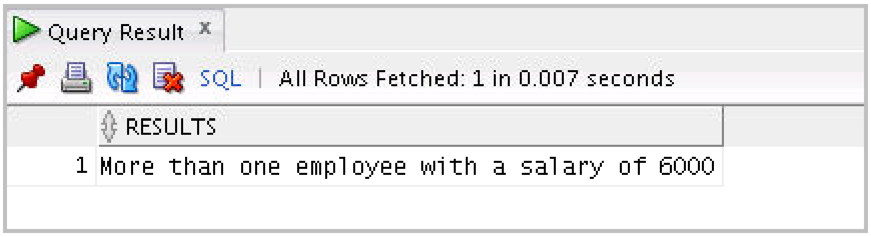
In this practice, you write a PL/SQL block that applies a predefined exception to process only one record at a time. The PL/SQL block selects the name of the employee with a given salary value.

1. Execute the command in the `lab_06_01.sql` file to re-create the `messages` table.
2. In the declarative section, declare two variables: `v_ename` of type `employees.last_name` and `v_emp_sal` of type `employees.salary`. Initialize the latter to 6000.
3. In the executable section, retrieve the last names of employees whose salaries are equal to the value in `v_emp_sal`. If the salary entered returns only one row, insert the employee's name and salary amount into the `MESSAGES` table.
   Note: Do not use explicit cursors.
4. If the salary entered does not return any rows, handle the exception with an appropriate exception handler and insert the message "No employee with a salary of <*salary*>" into the `MESSAGES` table.
5. If the salary entered returns multiple rows, handle the exception with an appropriate exception handler and insert the message "More than one employee with a salary of <*salary*>" into the `MESSAGES` table.
6. Handle any other exception with an appropriate exception handler and insert the message "Some other error occurred" into the `MESSAGES` table.
7. Display the rows from the `MESSAGES` table to check whether the PL/SQL block has executed successfully. The output is as follows:



Task completed in 0.296 seconds

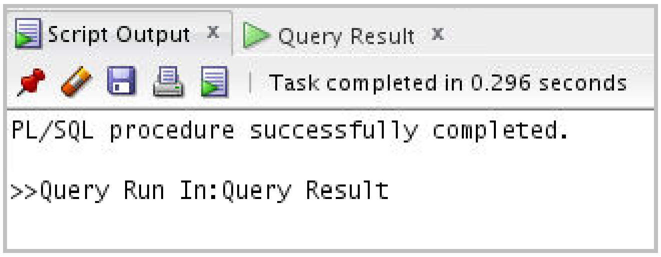
```
PL/SQL procedure successfully completed.

>>Query Run In:Query Result
```



All Rows Fetched: 1 in 0.007 seconds

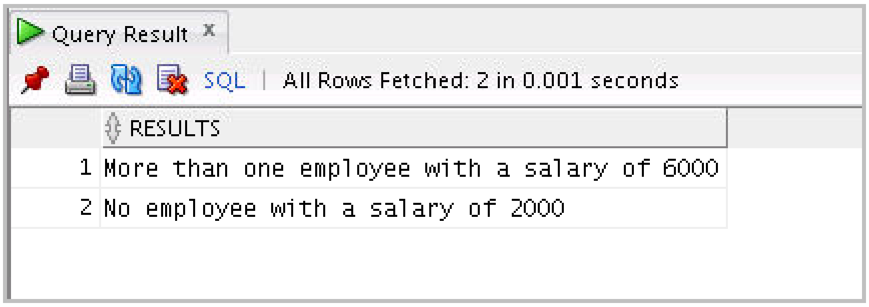
	RESULTS
1	More than one employee with a salary of 6000

8. Change the initialized value of `v_emp_sal` to 2000 and re-execute. The output is as follows:



Task completed in 0.296 seconds

```
PL/SQL procedure successfully completed.

>>Query Run In:Query Result
```



```
All Rows Fetched: 2 in 0.001 seconds

  RESULTS
1 More than one employee with a salary of 6000
2 No employee with a salary of 2000
```

Solution 9-1: Handling Predefined Exceptions

In this practice, you write a PL/SQL block that applies a predefined exception to process only one record at a time. The PL/SQL block selects the name of the employee with a given salary value.

1. Execute the command in the `lab_06_01.sql` file to re-create the `MESSAGES` table.
2. In the declarative section, declare two variables: `v_ename` of type `employees.last_name` and `v_emp_sal` of type `employees.salary`. Initialize the latter to 6000.

```
DECLARE
  v_ename     employees.last_name%TYPE;
  v_emp_sal   employees.salary%TYPE := 6000;
```

3. In the executable section, retrieve the last names of employees whose salaries are equal to the value in `v_emp_sal`. If the salary entered returns only one row, insert the employee’s name and the salary amount into the `MESSAGES` table.

   Note: Do not use explicit cursors.

```
BEGIN
  SELECT last_name
  INTO       v_ename
  FROM       employees
  WHERE      salary = v_emp_sal;
  INSERT INTO messages (results)
  VALUES (v_ename || ' - ' || v_emp_sal);
```

4. If the salary entered does not return any rows, handle the exception with an appropriate exception handler and insert the message “No employee with a salary of <*salary*>” into the `MESSAGES` table.

```
EXCEPTION
  WHEN no_data_found THEN
    INSERT INTO messages (results)
    VALUES ('No employee with a salary of '||
            TO_CHAR(v_emp_sal));
```

5. If the salary entered returns multiple rows, handle the exception with an appropriate exception handler and insert the message “More than one employee with a salary of <*salary*>” into the `MESSAGES` table.

```
WHEN too_many_rows THEN
    INSERT INTO messages (results)
    VALUES ('More than one employee with a salary of '||
             TO_CHAR(v_emp_sal));
```

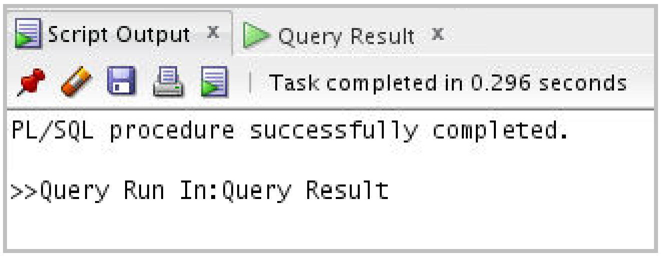
6. Handle any other exception with an appropriate exception handler and insert the message “Some other error occurred” into the MESSAGES table.

```
WHEN others THEN
    INSERT INTO messages (results)
    VALUES ('Some other error occurred.');
END;
```

7. Display the rows from the MESSAGES table to check whether the PL/SQL block has executed successfully.

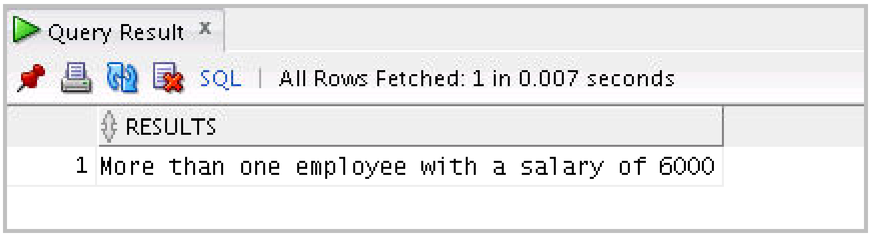
```
/
SELECT * FROM messages;
```

The output is as follows:



Task completed in 0.296 seconds

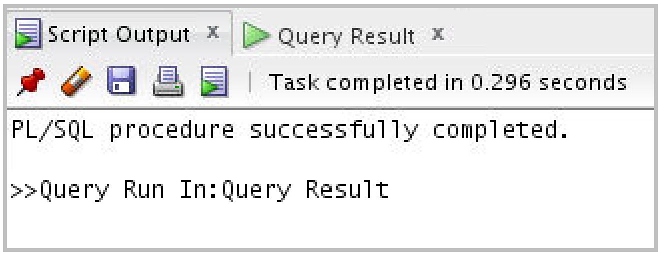
```
PL/SQL procedure successfully completed.

>>Query Run In:Query Result
```



All Rows Fetched: 1 in 0.007 seconds

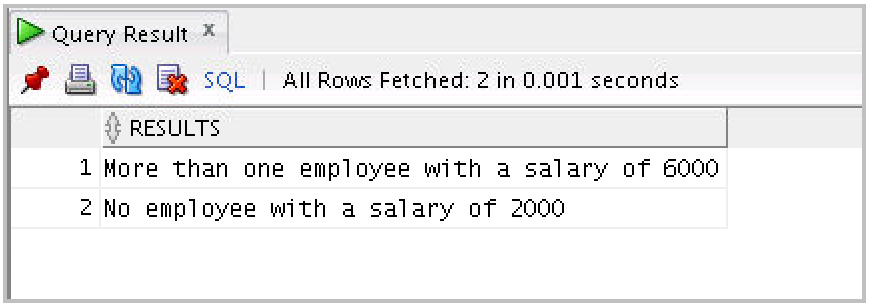
	RESULTS
1	More than one employee with a salary of 6000

8. Change the initialized value of v_emp_sal to 2000 and re-execute. The output is as follows:



Task completed in 0.296 seconds

```
PL/SQL procedure successfully completed.

>>Query Run In:Query Result
```



All Rows Fetched: 2 in 0.001 seconds

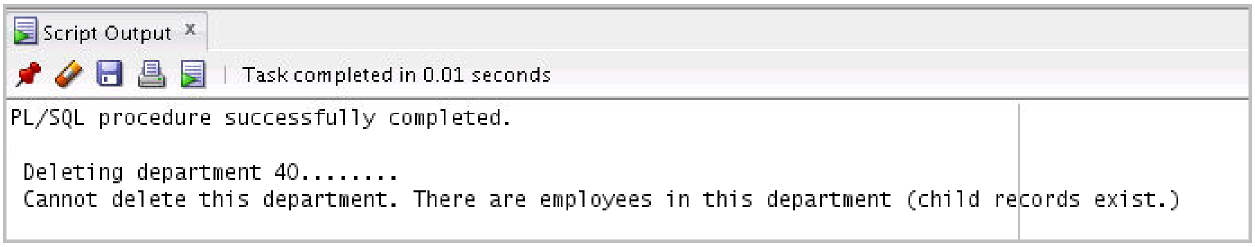
	RESULTS
1	More than one employee with a salary of 6000
2	No employee with a salary of 2000

Practice 9-2: Handling Standard Oracle Server Exceptions

In this practice, you write a PL/SQL block that declares an exception for the Oracle Server error `ORA-02292 (integrity constraint violated – child record found)`. The block tests for the exception and outputs the error message.

1. In the declarative section, declare an exception `e_childrecord_exists`. Associate the declared exception with the standard Oracle Server error `–02292`.
2. In the executable section, display "Deleting department 40...." Include a `DELETE` statement to delete the department with the `department_id` 40.
3. Include an exception section to handle the `e_childrecord_exists` exception and display the appropriate message.

The sample output is as follows:



```
PL/SQL procedure successfully completed.

 Deleting department 40........
 Cannot delete this department. There are employees in this department (child records exist.)
```

Solution 9-2: Handling Standard Oracle Server Exceptions

In this practice, you write a PL/SQL block that declares an exception for the Oracle Server error `ORA-02292 (integrity constraint violated – child record found)`. The block tests for the exception and outputs the error message.

1. In the declarative section, declare an exception `e_childrecord_exists`. Associate the declared exception with the standard Oracle Server error `–02292`.

```
SET SERVEROUTPUT ON
DECLARE
   e_childrecord_exists EXCEPTION;
   PRAGMA EXCEPTION_INIT(e_childrecord_exists, -02292);
```

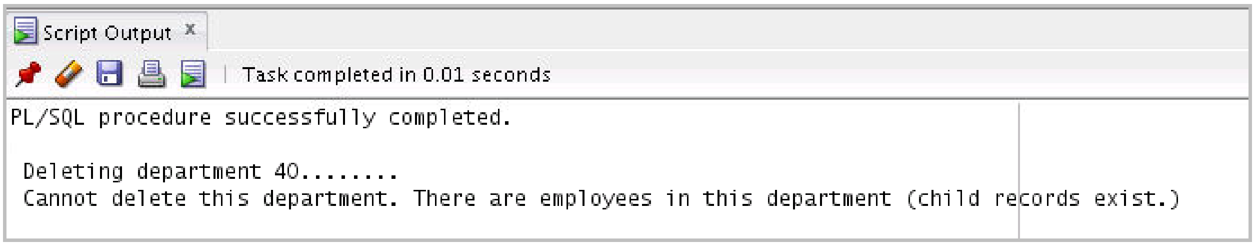
2. In the executable section, display "Deleting department 40...." Include a `DELETE` statement to delete the department with `department_id` 40.

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(' Deleting department 40........');
   delete from departments where department_id=40;
```

3. Include an exception section to handle the `e_childrecord_exists` exception and display the appropriate message.

```
EXCEPTION
   WHEN e_childrecord_exists THEN
   DBMS_OUTPUT.PUT_LINE(' Cannot delete this department. There are
employees in this department (child records exist.) ');
END;
```

The sample output is as follows:



```
PL/SQL procedure successfully completed.

 Deleting department 40........
 Cannot delete this department. There are employees in this department (child records exist.)
```