Lab 5. PL/SQL Cursors, Exceptions

Point Distribution:

Option A	
Question Number	Points
7a, 8a	2pt x 2
6c, 8c, 8e	1pt x 3
Other questions	0.9pt x 20
Total	25pt

Option B	
Question Number	Points
1, 2	12.5pt x 2
Total	25pt

Submission:

- If you decide to skip the lab, make sure you submit the **check-off questions** (highlighted with green background) in the text box of the check-off assignment item for this Lab before **Wednesday, noon to Brightspace**.
- **All students are expected to submit your answers to Option A or Option B in a text document with the name <lastname>_<firstname>_lab5.sql or .txt by the due date to Gradescope.**
- Please include both your code and the results in the **.txt** or **.sql** documents for full credits. For detailed requirements, please refer to the "Lab and Homework Submission Guideline."
- You may choose to work with Option A or Option B for credit. Please indicate which part you wish to submit. If you submit both, only Option A will be graded. If you choose to submit Option B, please make sure that you know the answers to Option A.

Objectives:

- Practice implementing implicit PL/SQL Cursors
- Practice implementing explicit PL/SQL Cursors
- Practice implementing PL/SQL Cursor FOR Loops
- Practice handling exceptions
- Familiarization with standard Oracle exceptions

Option A

Explicit Cursors

1a Using the code provided below, create the **print_employee_roster** procedure. The procedure uses an explicit cursor.

Worksheet | Query Builder

```
CREATE OR REPLACE PROCEDURE print_employee_roster
AS
  current_employeeid employee.employeeid%TYPE;
  current_lastname employee.lastname%TYPE;
  current_firstname employee.firstname%TYPE;

  CURSOR all_employees
  IS
    SELECT employeeid, lastname, firstname FROM employee;

BEGIN
  OPEN all_employees;

  FETCH all_employees into current_employeeid, current_lastname, current_firstname;

  WHILE all_employees%FOUND LOOP

      DBMS_OUTPUT.PUT (RPAD(current_employeeid, 15, ' '));
      DBMS_OUTPUT.PUT (RPAD(current_lastname, 30, ' '));
      DBMS_OUTPUT.PUT_LINE (current_firstname);
      FETCH all_employees into current_employeeid, current_lastname, current_firstname;
  END LOOP;

  CLOSE all_employees;
END print_employee_roster;
```

Note: the procedure uses 3 local variables of %TYPE to hold the “fetched” values.

1b Execute the **print_employee_roster** procedure. Include the results in your submission.

2a	Improve the **print_employee_roster** procedure by using a single local variable of %ROWTYPE (instead of the 3 local variables it currently uses) as shown below.
2b	Execute the **print_employee_roster** procedure. Include the results in your submission.
2c	What is the database object that the **current_employee** variable is based upon?

Worksheet | Query Builder

```
CREATE OR REPLACE PROCEDURE print_employee_roster
AS

  CURSOR all_employees
  IS
    SELECT employeeid, lastname, firstname FROM employee;

  current_employee  all_employees%ROWTYPE;
BEGIN
  OPEN all_employees;

  FETCH all_employees into current_employee;

  WHILE all_employees%FOUND LOOP

      DBMS_OUTPUT.PUT (RPAD(current_employee.employeeid, 15, ' '));
      DBMS_OUTPUT.PUT (RPAD(current_employee.lastname, 30, ' '));
      DBMS_OUTPUT.PUT_LINE (current_employee.firstname);
      FETCH all_employees into current_employee;
  END LOOP;

  CLOSE all_employees;
END print_employee_roster;
```

3a	Modify the cursor definition to include concatenation, formatting, and an alias as shown below.

Worksheet | Query Builder

```
CREATE OR REPLACE PROCEDURE print_employee_roster
AS

  CURSOR all_employees
  IS
    SELECT  employeeid,
            lastname || ', ' || firstname as name
    FROM employee;

  current_employee  all_employees%ROWTYPE;
BEGIN
  OPEN all_employees;

  FETCH all_employees into current_employee;

  WHILE all_employees%FOUND LOOP

      DBMS_OUTPUT.PUT (RPAD(current_employee.employeeid, 15, ' '));
      DBMS_OUTPUT.PUT_LINE (current_employee.name);
      FETCH all_employees into current_employee;
  END LOOP;

  CLOSE all_employees;
END print_employee_roster;
```

3b	Execute the **print_employee_roster** procedure. Include the results in your submission.

Cursor FOR Loops

4a Modify the **print_employee_roster** procedure to use a cursor FOR loop as shown below.

Worksheet | Query Builder

```
CREATE OR REPLACE PROCEDURE print_employee_roster
AS

  CURSOR all_employees
  IS
    SELECT  employeeid,
            lastname || ', ' || firstname as name
    FROM employee;

BEGIN

  FOR current_employee IN all_employees LOOP

      DBMS_OUTPUT.PUT (RPAD(current_employee.employeeid, 15, ' '));
      DBMS_OUTPUT.PUT_LINE (current_employee.name);
  END LOOP;

END print_employee_roster;
```

Be certain to remove any unnecessary variables from your procedure.

4b Execute the **print_employee_roster** procedure. Include the results in your submission.

4c Which of the processing steps associated with cursor usage are implicitly handled by the PL/SQL engine?

Parameterized Cursors

5a Modify the **print_employee_roster** procedure to accept a parameter that is subsequently used in the WHERE clause of the cursor to restrict the data retrieved by the query.

Worksheet | Query Builder

```
CREATE OR REPLACE PROCEDURE print_employee_roster
(p_jobtitle IN employee.jobtitle%type)
AS

  CURSOR all_employees
  IS
    SELECT  employeeid,
            lastname || ', ' || firstname as name
    FROM employee
    WHERE UPPER(TRIM(JOBTITLE)) = UPPER(TRIM(p_jobtitle));

BEGIN

  FOR current_employee IN all_employees LOOP

      DBMS_OUTPUT.PUT (RPAD(current_employee.employeeid, 15, ' '));
      DBMS_OUTPUT.PUT_LINE (current_employee.name);
  END LOOP;

END print_employee_roster;
```

5b What is the code necessary to display all 'sales' personnel at Eagle Electronics using the **print_employee_roster** procedure? Include the results in your submission.

5c What is the code necessary to display all 'assembly' personnel at Eagle Electronics using the **print_employee_roster** procedure? Include the results in your submission.

5d What happens if you provide the procedure a non-existent job title? For example: 'student'. Does the procedure crash? Or display nothing? Or what?

Exception Handling

6a Modify the **print_employee_roster** procedure to gracefully handle exceptions in a "generic" manner (e.g., treat all exceptions the same).

Worksheet | Query Builder

```
CREATE OR REPLACE PROCEDURE print_employee_roster
(p_jobtitle IN employee.jobtitle%type)
AS

  CURSOR all_employees
  IS
    SELECT  employeeid,
            lastname || ', ' || firstname as name
    FROM employee
    WHERE UPPER(TRIM(JOBTITLE)) = UPPER(TRIM(p_jobtitle));

BEGIN

  FOR current_employee IN all_employees LOOP

      DBMS_OUTPUT.PUT (RPAD(current_employee.employeeid, 15, ' '));
      DBMS_OUTPUT.PUT_LINE (current_employee.name);
  END LOOP;

EXCEPTION

  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT (SQLCODE);
    DBMS_OUTPUT.PUT (': ');
    DBMS_OUTPUT.PUT_LINE (SUBSTR(SQLERRM, 1, 100));
END print_employee_roster;

```

6b What code is now necessary to display all 'assembly' personnel at Eagle Electronics using the **print_employee_roster** procedure? Include the results in your submission.

6c If an exception was to occur, what would be displayed to the user? You will need to understand what SQLCODE and SQLERRM functions do

Write Your Own Code	
7a	Create a **procedure** named **customer_roster** that accepts 1 parameter: state. The procedure should display the company name, city, state, and contact name (in *Last, First* format) for all customers that are located in the state specified. The procedure should: • Use an **explicit** cursor and a **WHILE** Loop. • Format the output so it is easy to read (e.g., make certain the columns line up and formatting is standard). Handle any exceptions that might occur gracefully, displaying the error number and message to the user.
7b	If we wanted to determine the customers that Eagle Electronics has in Georgia (GA) using the **customer_roster** procedure, what is the code necessary to execute the procedure? Verify your solution works by running it. Include the results in your submission.
8a	Create a **procedure** named **customer_search** that accepts 1 parameter: name. The procedure should display the company name, contact first name, contact last name, and contact title for any customer contact whose last name contains the characters string specified by the user (e.g., use LIKE functionality). The search should be case insensitive and ignore leading and trailing spaces. The procedure should: • Use an **explicit** cursor. • Format the output so it is easy to read (e.g., make certain the columns line up and formatting is standard). Handle any exceptions that might occur gracefully, displaying the error number and message to the user.
8b	If we wanted to use the **customer_search** procedure to identify all customer contacts having a last name containing the character string 'NA', what code is necessary to execute the procedure? Verify your solution works by running it. Include the results in your submission.
8c	Modify the search functionality of the **customer_search** procedure created in 8a. The search should now display the information for customer contacts whose first name or last name contains the specified character string.
8d	If we wanted to use the **customer_search** procedure to identify all customer contacts having a first or last name containing the character string 'na', what code is necessary to execute the procedure? Verify your solution works by running it. Include the results in your submission.

8e	Modify the search functionality of the **customer_search** procedure in 8c to using an **IMPLICIT** cursor with a **FOR** loop.
8f	Repeat 8d with the updated procedure.

Option B

Optional Questions: EAGLE database	
1	Create a **procedure** that would: given an employee first name, last name, and date of birth, find employees that they supervise. Order the supervised employees by last name. The procedure should: • Use first name, last name, date of birth as input to your procedure • Use exception if any part of input is missing • Use exception if such employee doesn't exist (it should not be case sensitive) • Use exception if an employee doesn't supervise anyone • Use exception if date is invalid • Use exceptions for anything else that you can think of. • When you list the supervised employees, use a cursor and a FOR loop (implicit OPEN, FETCH) • Prior to listing the supervised employees, print a message to the screen indicating how many employees are there that match given criteria.
2	Write a **function** that would: given an employee name, return the number of people they supervise. The function should: • Use employee id as input • Use exception if any part of input is missing • Use exception if invalid last name is entered, however, allow for variations in first name, such as initial. • If a user enters a name that corresponds to more than one employee, return the overall sum for all of them, not individual numbers. • Use exceptions for anything else that you can think of. • Use a parameterized cursor and a WHILE loop to display all supervised employees that you can find for a given name. • If an employee with such name doesn't exist, suggest a similarly spelled name that does exist.