Lab 6 – Week 6 (Stored Procedures/Conditional Statements)

Submission

Your submission will be a single text-based SQL file with appropriate header and commenting. Please ensure your file runs when the entire file is executed in SQL Developer.

Create a new Worksheet in SQL Developer. Save the file as L05_ID#_LASTNAME.sql

Your submission needs to be commented and include the question, the solutions.
In this Lab, you create PL/SQL stored procedures to perform the following tasks. As you know, a stored procedure does not return any value. To send values back to the caller, you can use OUT parameters.

A parameter can be

- IN patameter
- OUT parameter
- IN OUT parameter

See the following template:

```
CREATE OR REPLACE procedure_name(arg1 IN/OUT/IN OUT data_type, ...)
AS

BEGIN

  ....
EXCEPTION

WHEN OTHERS

  THEN

      DBMS_OUTPUT.PUT_LINE (Error!');

END procedure_name;
```

For all the stored procedures make sure you handle all exceptions such as

- TOO_MANY_ROWS
- NO_DATA_FOUND
- OTHERS

Besides checking all required exceptions, have the OTHER exception checked just in case any error occurs that has not been anticipated at the time you write the code.

Tasks

1. Write a store procedure that get an integer number and prints

   The number is even.
   If a number is divisible by 2.
   Otherwise, it prints
   The number is odd.

2. Create a stored procedure named *find_employee*. This procedure gets an employee number and prints the following employee information:

   First name
   Last name
   Email
   Phone
   Hire date
   Job title

   The procedure gets a value as the employee ID of type NUMBER.

   See the following example for employee ID 107:

   ```
   First name: Summer
   Last name: Payn
   Email: summer.payne@example.com
   Phone: 515.123.8181
   Hire date: 07-JUN-16
   Job title: Public Accountant
   ```

   The procedure display a proper error message if any error accours.

3. Every year, the company increases the price of all products in one category. For example, the company wants to increase the price (list_price) of products in category 1 by $5. Write a procedure named *update_price_by_cat* to update the price of all products in a given category and the given amount to be added to the current price if the price is greater than 0. The procedure shows the number of updated rows if the update is successful.

   The procedure gets two parameters:

   - category_id IN NUMBER
   - amount NUMBER(9,2)

   To define the type of variables that store values of a table' column, you can also write:

   ```
   vriable_name table_name.column_name%type;
   ```

The above statement defines a variable of the same type as the type of the table' column.

```
category_id products.category_id%type;
```

Or you need to see the table definition to find the type of the category_id column. Make sure the type of your variable is compatible with the value that is stored in your variable.

To show the number of affected rows the update query, declare a variable named `rows_updated` of type NUMBER and use the SQL variable `sql%rowcount` to set your variable. Then, print its value in your stored procedure.

```
Rows_updated := sql%rowcount;
```

`SQL%`*`ROWCOUNT`* stores the number of rows affected by an INSERT, UPDATE, or DELETE.

4. Every year, the company increase the price of products whose price is less than the average price of all products by 1%. (list_price * 1.01). Write a stored procedure named *update_price_under_avg*. This procedure do not have any parameters. You need to find the average price of all products and store it into a variable of the same type. If the average price is less than or equal to $1000, update products' price by 2% if the price of the product is less than the calculated average. If the average price is greater than $1000, update products' price by 1% if the price of the product is less than the calculated average. The query displays an error message if any error occurs. Otherwise, it displays the number of updated rows.

5. The company needs a report that shows three category of products based their prices. The company needs to know if the product price is cheap, fair, or expensive. Let's assume that

   - If the list price is less than
     - (avg_price - min_price) / 2

     The product's price is cheap.
   - If the list price is greater than
     - (max_price - avg_price) / 2

     The product' price is expensive.
   - If the list price is between
     - (avg_price - min_price) / 2
     - and
     - (max_price - avg_price) / 2
     - the end values included

     The product's price is fair.

   Write a procedure named *product_price_report* to show the number of products in each price category:

   The following is a sample output of the procedure if no error occurs:

```
Cheap: 10
Fair: 50
Expensive: 18
```

The values in the above examples are just random values and may not match the real numbers in your result.

The procedure has no parameter. First, you need to find the average, minimum, and maximum prices (list_price) in your database and store them into varibles avg_price, min_price, and max_price.

You need more three varaibles to store the number of products in each price category:

cheap_count
fair_count
exp_count

Make sure you choose a proper type for each variable. You may need to define more variables based on your solution.

Example Submission

```
-- ************************
-- Name: Your Name
-- ID: #########
-- Date: The current date
-- Purpose: Lab 5 DBS311
-- ************************

-- Question 1 – write a brief note about what the question is asking

-- Q1 SOLUTION –
        CREATE OR REPLACE procedure_name(arg1 data_type, ...) AS
        BEGIN
          ....
        EXCEPTION
        WHEN OTHERS
          THEN
             DBMS_OUTPUT.PUT_LINE (Error!');
        END procedure_name;

-- Question 2 –

-- Q2 Solution –
```