Lab 7 (Stored Procedures/Iterative Statements)

**Submission**

***ON Blackboard***

Create a new Worksheet in SQL Developer. Save the file as L06\_ID#\_LASTNAME.txt

Your submission needs to be commented and include the question and the solutions in Word document

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 parameter
* 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 The company wants to calculate what the new buy price of each retail product would be:

Write a stored procedure named ***calculate\_newbuyprice*** which calculates the new buying price by multiplying the percentage increase for each othe productline given in the table. (Use CASE statements for each productline and FOR loop construct for each retail product).

|  |  |
| --- | --- |
| PRODUCTLINE | PERCENT OF BUYPRICE INCREASE |
| Trucks and Buses | 1.1 |
| Vintage Cars | 1.5 |
| Classic Cars | 1.2 |
| Ships | 1.3 |
| Motorcycles | 1.2 |
| Trains | 1.6 |
| Planes | 1.7 |

The procedure calculates and prints the product name, old buy price and new price.

Sample output:

Exception block should handle all error that might occur with custom messages for “If the buy price is 0” product does not exist, the procedure displays a proper message.

**CREATE OR REPLACE PROCEDURE calculate\_newbuyprice AS**

**v\_product\_name retailproducts.productname%TYPE;**

**v\_old\_buy\_price retailproducts.buyprice%TYPE;**

**v\_new\_buy\_price NUMBER;**

**v\_percentage NUMBER;**

**CURSOR product\_cursor IS**

**SELECT productname, buyprice**

**FROM retailproducts;**

**BEGIN**

**dbms\_output.put\_line(**

**RPAD('Product Name', 25) ||**

**RPAD('Old Buy Price', 25) ||**

**RPAD('New Buy Price', 25)**

**);**

**dbms\_output.put\_line(**

**LPAD('-', 25, '-') ||**

**LPAD('-', 25, '-') ||**

**LPAD('-', 25, '-')**

**);**

**OPEN product\_cursor;**

**LOOP**

**FETCH product\_cursor into v\_product\_name, v\_old\_buy\_price;**

**EXIT WHEN product\_cursor%notfound;**

**-- Calculate new buy price based on product line percentage increase**

**CASE v\_product\_name**

**WHEN 'Trucks and Buses' THEN v\_percentage := 1.1;**

**WHEN 'Vintage Cars' THEN v\_percentage := 1.5;**

**WHEN 'Classic Cars' THEN v\_percentage := 1.2;**

**WHEN 'Ships' THEN v\_percentage := 1.3;**

**WHEN 'Motorcycles' THEN v\_percentage := 1.2;**

**WHEN 'Trains' THEN v\_percentage := 1.6;**

**WHEN 'Planes' THEN v\_percentage := 1.7;**

**ELSE v\_percentage := 1; -- Default percentage if productline not found**

**END CASE;**

**v\_new\_buy\_price := v\_old\_buy\_price \* v\_percentage;**

**-- Display information**

**dbms\_output.put\_line(**

**RPAD(v\_product\_name, 25) ||**

**RPAD(TO\_CHAR(v\_old\_buy\_price, '99999.99'), 25) ||**

**RPAD(TO\_CHAR(v\_new\_buy\_price, '99999.99'), 25)**

**);**

**END LOOP;**

**CLOSE product\_cursor;**

**EXCEPTION**

**WHEN ZERO\_DIVIDE THEN**

**dbms\_output.put\_line('Error: Buy Price is 0.');**

**WHEN NO\_DATA\_FOUND THEN**

**dbms\_output.put\_line('No products found.');**

**WHEN OTHERS THEN**

**dbms\_output.put\_line('Error: ' || SQLERRM);**

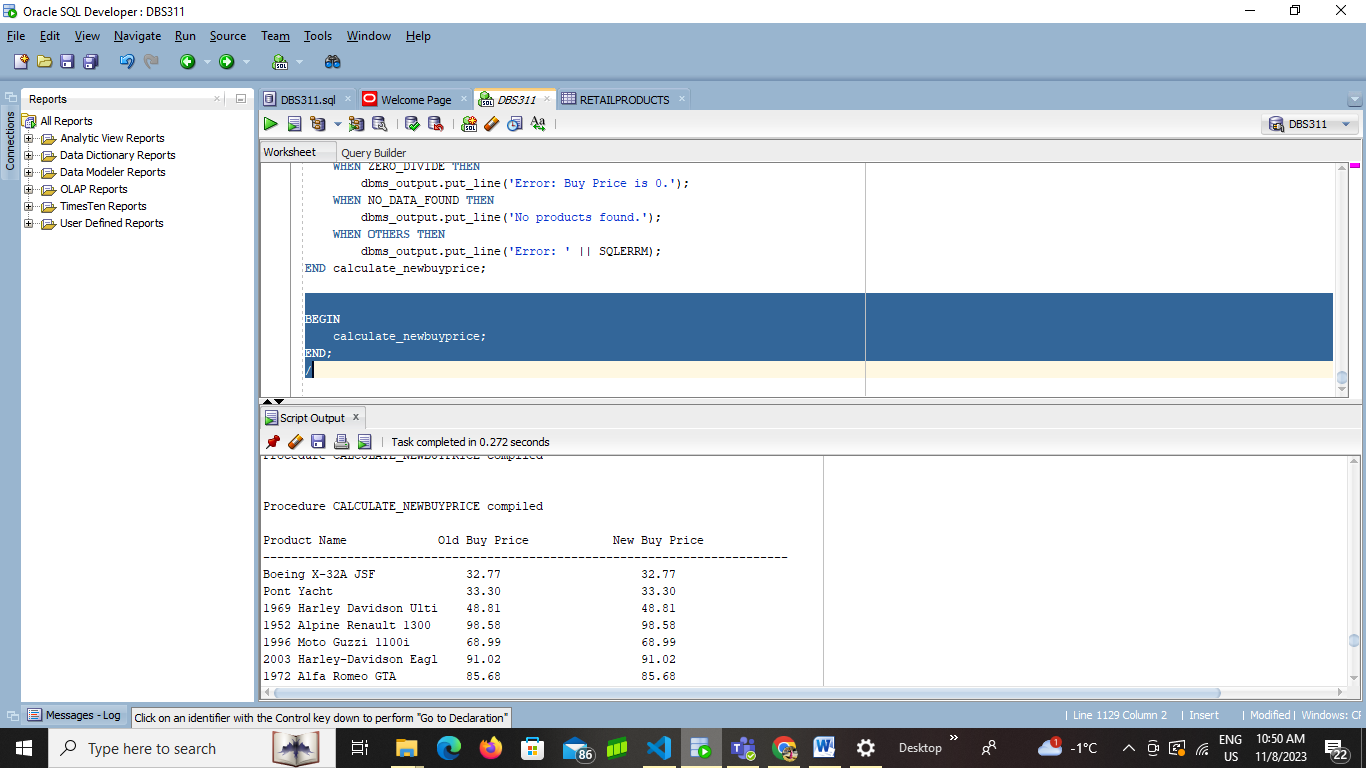
**END calculate\_newbuyprice;**

**BEGIN**

**calculate\_newbuyprice;**

**END;**

**/**



2 Write a stored procedure named **employee\_works\_here** to print the retail employee number, employee Last name and job title.(USING WHILE LOOP)

This is sample output

Employee # Last Name Job title

9999 Able Sales Rep

9998 Hostable Manager

If the value of the job title is null or does not exist, display “no job title”.

**CREATE OR REPLACE PROCEDURE employee\_works\_here AS**

**v\_employee\_number retailemployees.employeenumber%TYPE;**

**v\_last\_name retailemployees.lastname%TYPE;**

**v\_job\_title retailemployees.jobtitle%TYPE;**

**BEGIN**

**dbms\_output.put\_line(**

**RPAD('Employee #', 15) ||**

**RPAD('Last Name', 15) ||**

**RPAD('Job Title', 30)**

**);**

**dbms\_output.put\_line(**

**LPAD('-', 15, '-') ||**

**LPAD('-', 15, '-') ||**

**LPAD('-', 30, '-')**

**);**

**-- Cursor to fetch employee information**

**FOR employee\_rec IN (**

**SELECT employeenumber, lastname, NVL(jobtitle, 'no job title') AS jobtitle**

**FROM retailemployees**

**)**

**LOOP**

**v\_employee\_number := employee\_rec.employeenumber;**

**v\_last\_name := employee\_rec.lastname;**

**v\_job\_title := employee\_rec.jobtitle;**

**dbms\_output.put\_line(**

**RPAD(v\_employee\_number, 15) ||**

**RPAD(v\_last\_name, 15) ||**

**RPAD(v\_job\_title, 30)**

**);**

**END LOOP;**

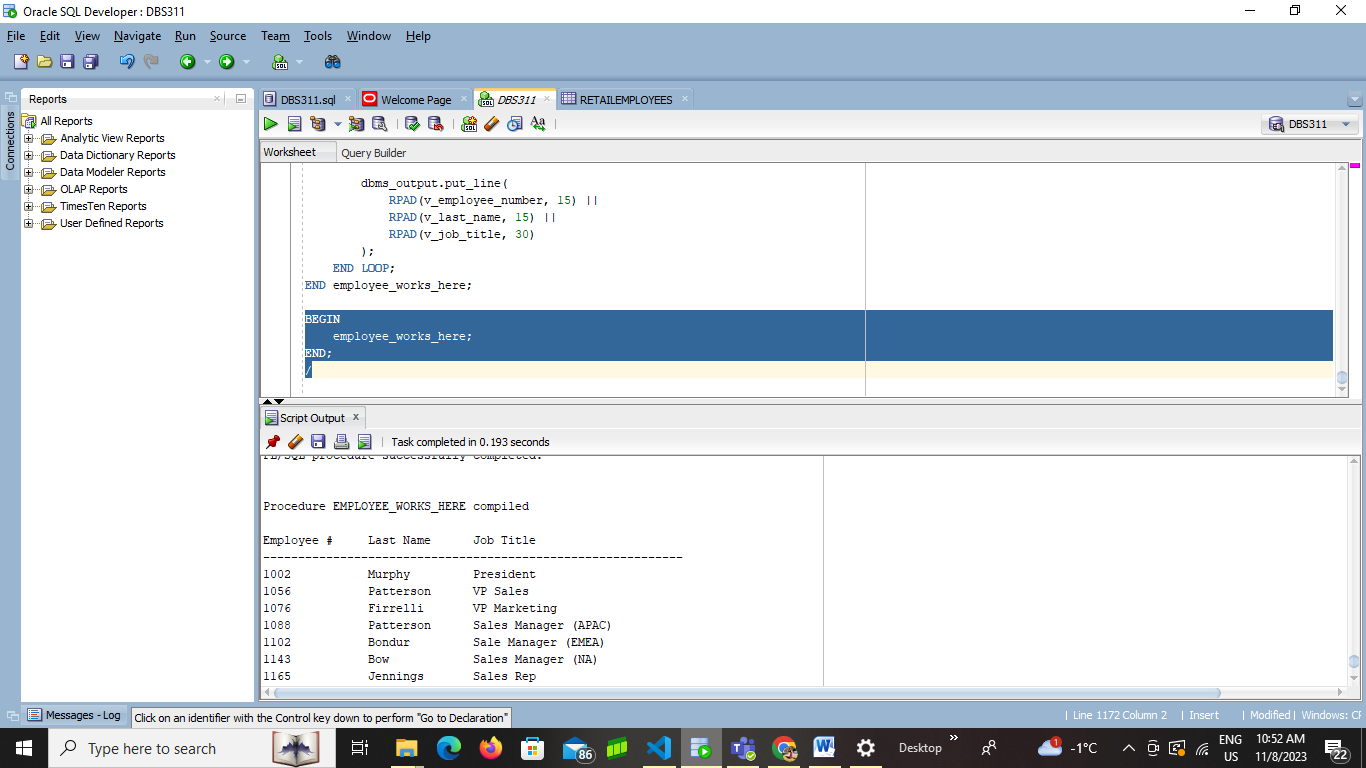
**END employee\_works\_here;**

**BEGIN**

**employee\_works\_here;**

**END;**

**/**



3. Write a stored procedure named CUSTOMER\_credithistory to provide a list of retail customers name, credit limit and credit level using CURSORS.

If credit \_limit is below $1000 display as “new customers” in credit level, if between 1000 to 5000 then display as” existing customers”. If above $4000 - $10,000 display as “credit approved for new increase” and others as “Waiting for Approval”

**CREATE OR REPLACE PROCEDURE CUSTOMER\_credithistory AS**

**v\_customer\_name retailcustomers.customername%TYPE;**

**v\_credit\_limit retailcustomers.creditlimit%TYPE;**

**v\_credit\_level VARCHAR2(100);**

**CURSOR customer\_cursor IS**

**SELECT customername, creditlimit**

**FROM retailcustomers;**

**BEGIN**

**dbms\_output.put\_line(**

**RPAD('Customer Name', 30) ||**

**RPAD('Credit Limit', 20) ||**

**RPAD('Credit Level', 30)**

**);**

**dbms\_output.put\_line(**

**LPAD('-', 30, '-') ||**

**LPAD('-', 20, '-') ||**

**LPAD('-', 30, '-')**

**);**

**OPEN customer\_cursor;**

**LOOP**

**FETCH customer\_cursor INTO v\_customer\_name, v\_credit\_limit;**

**EXIT WHEN customer\_cursor%NOTFOUND;**

**-- Determine credit level based on credit limit**

**IF v\_credit\_limit < 1000 THEN**

**v\_credit\_level := 'New Customers';**

**ELSIF v\_credit\_limit BETWEEN 1000 AND 5000 THEN**

**v\_credit\_level := 'Existing Customers';**

**ELSIF v\_credit\_limit BETWEEN 4000 AND 10000 THEN**

**v\_credit\_level := 'Credit Approved for Increase';**

**ELSE**

**v\_credit\_level := 'Waiting for Approval';**

**END IF;**

**-- Display information**

**dbms\_output.put\_line(**

**RPAD(v\_customer\_name, 30) ||**

**RPAD(TO\_CHAR(v\_credit\_limit, '99999.99'), 20) ||**

**RPAD(v\_credit\_level, 30)**

**);**

**END LOOP;**

**CLOSE customer\_cursor;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**dbms\_output.put\_line('No customers found.');**

**WHEN OTHERS THEN**

**dbms\_output.put\_line('Error: ' || SQLERRM);**

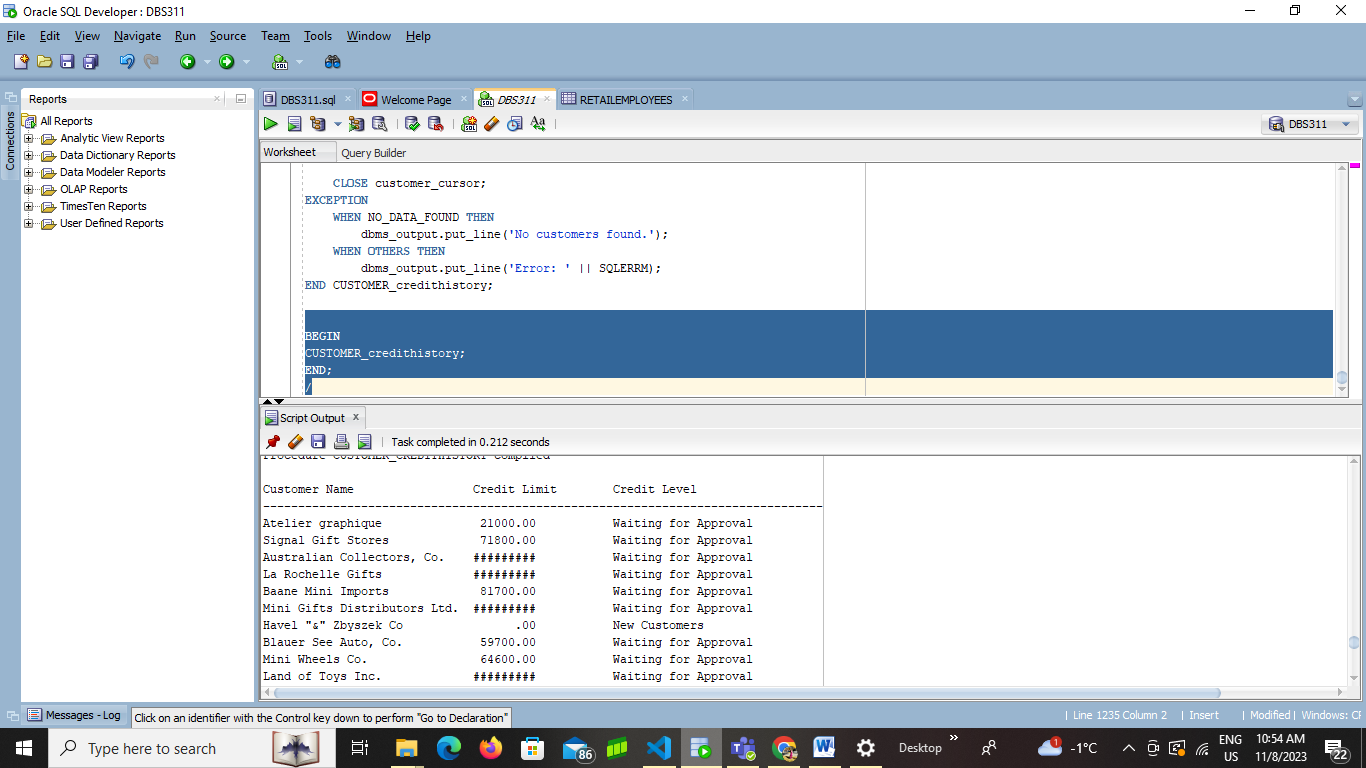
**END CUSTOMER\_credithistory;**

**BEGIN**

**CUSTOMER\_credithistory;**

**END;**

**/**



4.Write a PL/SQL stored procedure named order\_status to provide a list of retail ORDERS with order id and status using WHILE loop structure.

**CREATE OR REPLACE PROCEDURE order\_status AS**

**v\_order\_number retailorders.ordernumber%TYPE;**

**v\_status retailorders.status%TYPE;**

**v\_row\_count NUMBER := 0;**

**BEGIN**

**dbms\_output.put\_line(**

**RPAD('OrderNUMBER', 15) ||**

**RPAD('Status', 30)**

**);**

**dbms\_output.put\_line(**

**LPAD('-', 15, '-') ||**

**LPAD('-', 30, '-')**

**);**

**-- Cursor to fetch order information**

**FOR order\_rec IN (**

**SELECT ordernumber, status**

**FROM retailorders**

**)**

**LOOP**

**v\_order\_number := order\_rec.ordernumber;**

**v\_status := order\_rec.status;**

**dbms\_output.put\_line(**

**RPAD(v\_order\_number, 15) ||**

**RPAD(v\_status, 30)**

**);**

**v\_row\_count := v\_row\_count + 1;**

**END LOOP;**

**-- Check if no rows were fetched**

**IF v\_row\_count = 0 THEN**

**dbms\_output.put\_line('No orders found.');**

**END IF;**

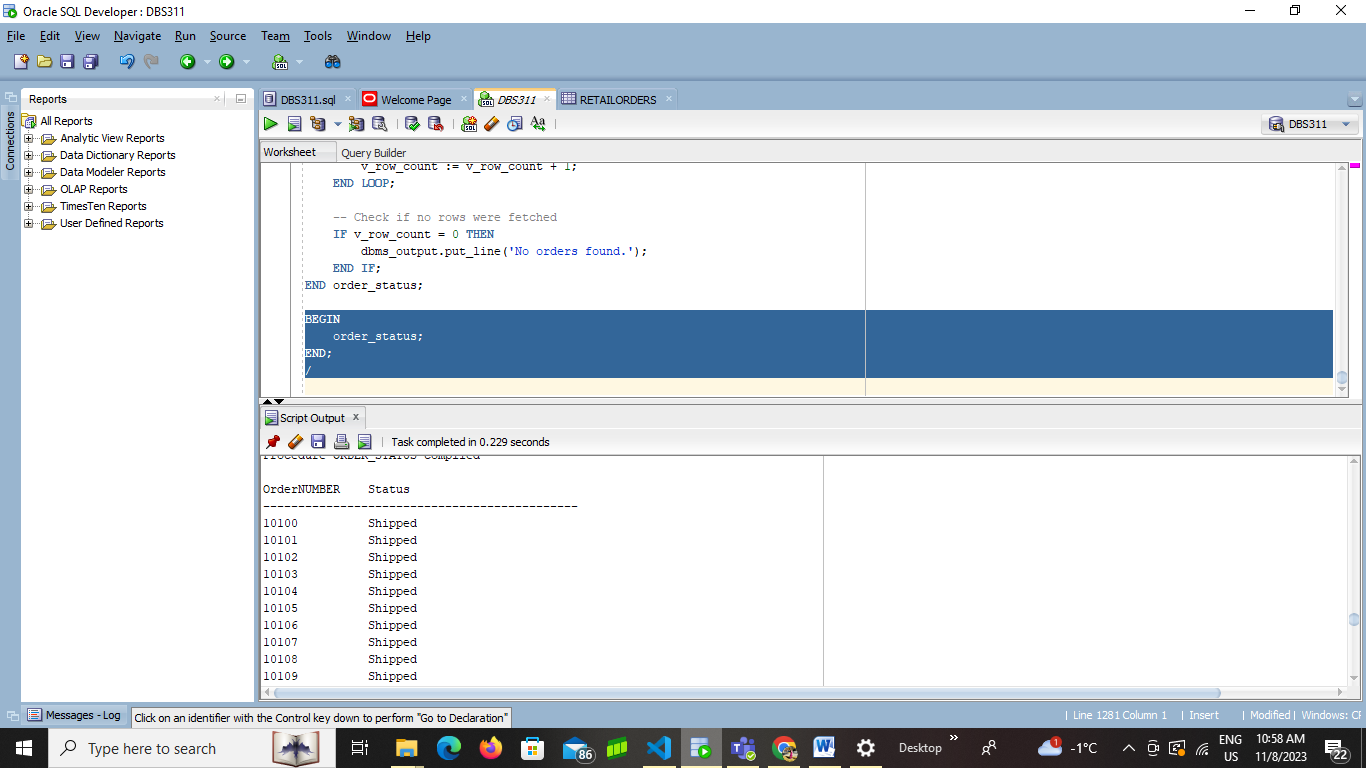
**END order\_status;**

**BEGIN**

**order\_status;**

**END;**

**/**



5 Create a new table OrderdetailsTmp from orderdetails. Add a new column called new\_price. Write a PL/SQL stored procedure named orders\_newprice. Using CURSORS update the new column in the orderdetailstmp table called new\_price (it is the updated price). Fill the new\_price column by the following calculation.

If price of each item is below $50 then add $30, if price of each item is between 50 to 100 then add $50, if it is between 200 to 500 then add $150 else add $200. Remember to display the price of each item, new\_price and ordernumber before the update happens. Use cursors and case statement.

**CREATE OR REPLACE PROCEDURE orders\_newprice AS**

**v\_order\_number OrderdetailsTmp.ordernumber%TYPE;**

**v\_price\_each OrderdetailsTmp.priceeach%TYPE;**

**v\_new\_price OrderdetailsTmp.new\_price%TYPE;**

**v\_increase NUMBER;**

**CURSOR order\_cursor IS**

**SELECT ordernumber, priceeach**

**FROM OrderdetailsTmp;**

**BEGIN**

**dbms\_output.put\_line(**

**RPAD('Order Number', 20) ||**

**RPAD('Price Each', 20) ||**

**RPAD('New Price', 20)**

**);**

**dbms\_output.put\_line(**

**LPAD('-', 20, '-') ||**

**LPAD('-', 20, '-') ||**

**LPAD('-', 20, '-')**

**);**

**OPEN order\_cursor;**

**LOOP**

**FETCH order\_cursor INTO v\_order\_number, v\_price\_each;**

**EXIT WHEN order\_cursor%NOTFOUND;**

**-- Calculate new price based on the given criteria**

**v\_increase :=**

**CASE**

**WHEN v\_price\_each < 50 THEN 30**

**WHEN v\_price\_each BETWEEN 50 AND 100 THEN 50**

**WHEN v\_price\_each BETWEEN 200 AND 500 THEN 150**

**ELSE 200**

**END;**

**v\_new\_price := v\_price\_each + v\_increase;**

**-- Display information before update**

**dbms\_output.put\_line(**

**RPAD(v\_order\_number, 20) ||**

**RPAD(v\_price\_each, 20) ||**

**RPAD(v\_new\_price, 20)**

**);**

**-- Update new\_price column**

**UPDATE OrderdetailsTmp**

**SET new\_price = v\_new\_price**

**WHERE ordernumber = v\_order\_number;**

**END LOOP;**

**CLOSE order\_cursor;**

**dbms\_output.put\_line('Update complete.');**

**EXCEPTION**

**WHEN OTHERS THEN**

**dbms\_output.put\_line('Error: ' || SQLERRM);**

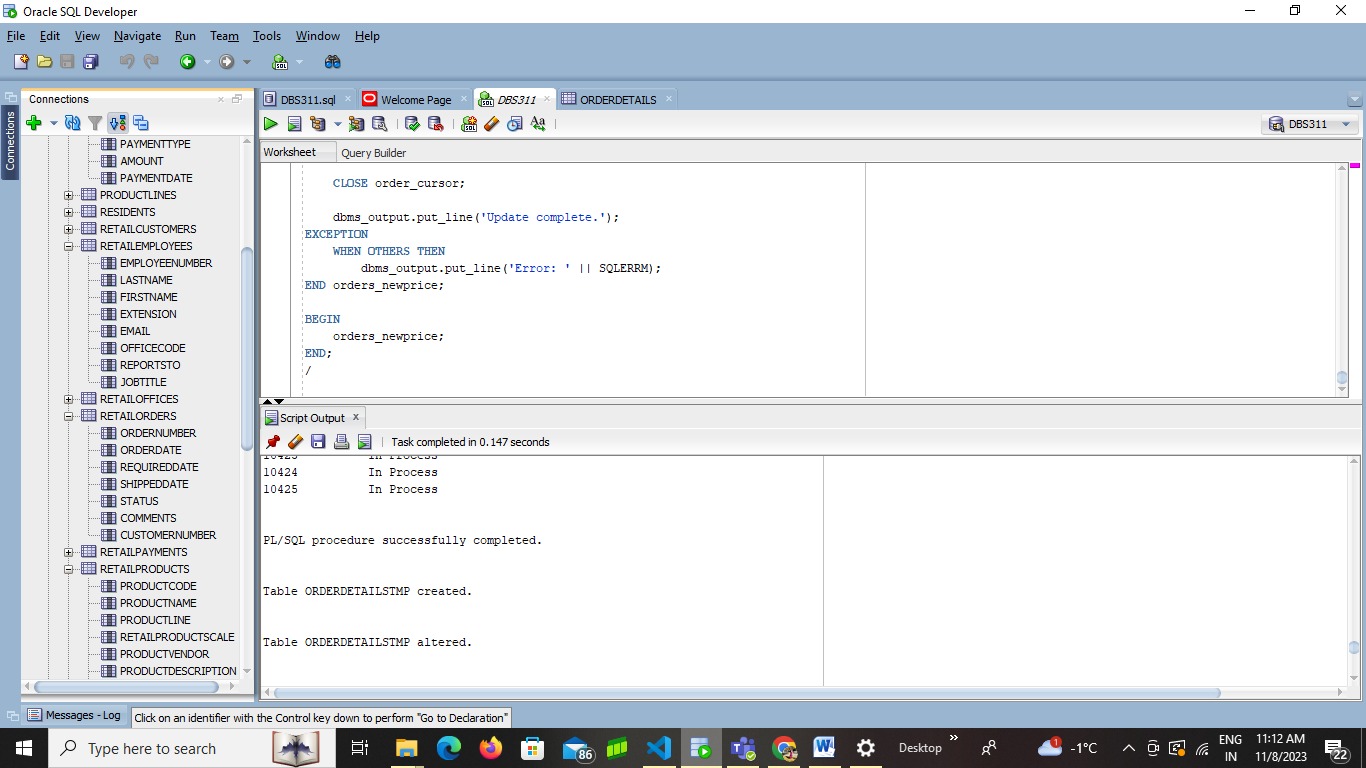
**END orders\_newprice;**

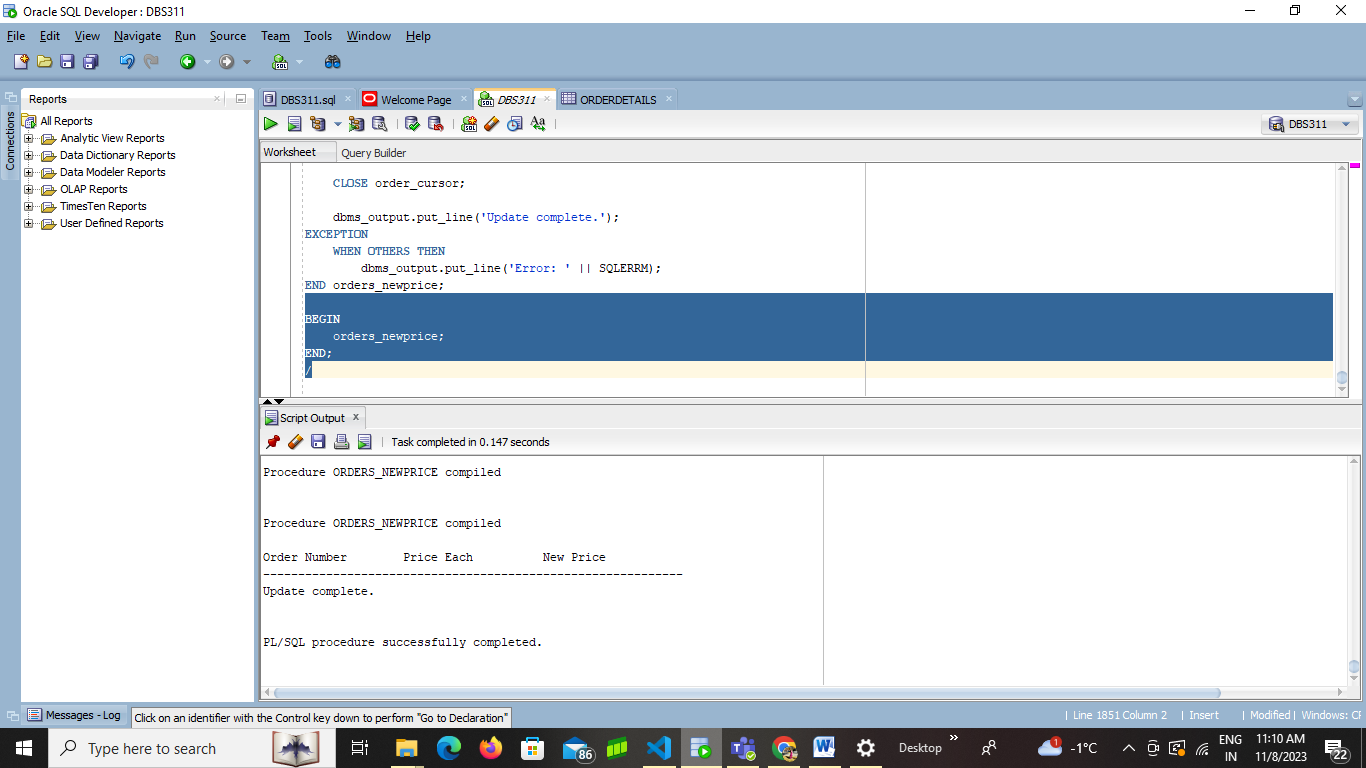
**BEGIN**

**orders\_newprice;**

**END;**

**/**





Go to next page to see sample submission

Example Submission

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
-- Name: Your Name  
-- ID: #########  
-- Date: The current date  
-- Purpose: Lab 7  
-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
-- 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;

SAMPLE OUTPUT  
Begin

Procedurename

End

Connecting to the database TestDatabase.

\*\*\* OUTPUT update\_low\_prices STARTED \*\*\*

Number of updates: 22

--------ENDED---------

Process exited.

Disconnecting from the database TestDatabase.

-- Question 2 –

-- Q2 Solution –