DBS311 Lab 7

1. Write a store procedure called *Get_Fact* that gets an integer number *n* and calculates and displays its factorial.

   Example:

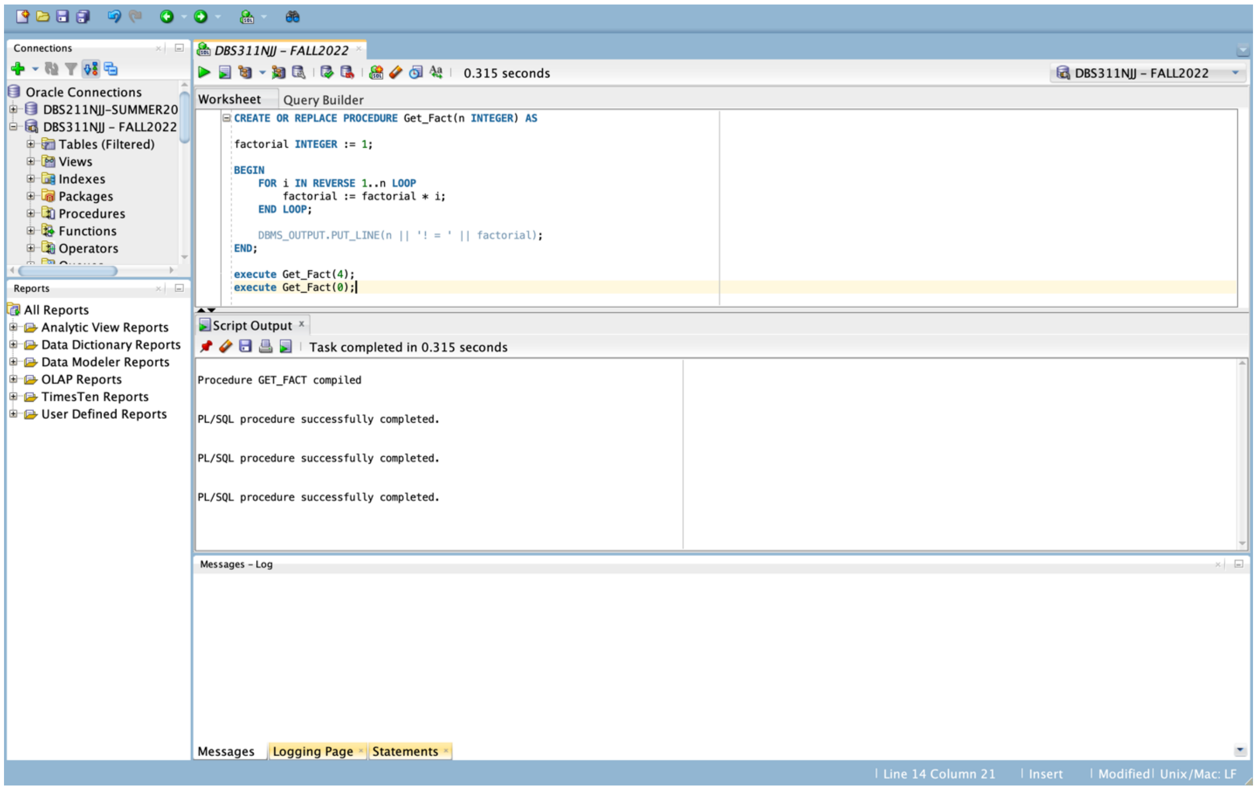
   0! = 1
   2! = fact(2) = 2 * 1 = 1
   3! = fact(3) = 3 * 2 * 1 = 6
   . . .
   n! = fact(n) = n * (n-1) * (n-2) * . . . * 1

   Show your testing with 2 different integers.

```
CREATE OR REPLACE PROCEDURE Get_Fact(n INTEGER) AS

factorial INTEGER := 1;

BEGIN
    FOR i IN REVERSE 1..n LOOP
        factorial := factorial * i;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE(n || '! = ' || factorial);
END;
/
```

TESTING:

```
execute Get_Fact(4);
4! = 24

execute Get_Fact(0);
0! = 1
```



```
CREATE OR REPLACE PROCEDURE Get_Fact(n INTEGER) AS

factorial INTEGER := 1;

BEGIN
    FOR i IN REVERSE 1..n LOOP
        factorial := factorial * i;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE(n || '! = ' || factorial);
END;

execute Get_Fact(4);
execute Get_Fact(0);
```

Task completed in 0.315 seconds

```
Procedure GET_FACT compiled


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

`Question 2 next page`

2. The company wants to calculate the employees’ annual salary:
   The first year of employment, the amount of salary is his/her base salary (shown under column Salary).
   Every year after that, the salary increases by 5%.
   Write a stored procedure named *Calculate_Salary* which gets an Employee ID and for that employee calculates the salary based on the number of years the employee has been working in the company. (Use a loop construct to calculate the salary).
   The procedure calculates and prints the Name and Annual Salary.

   Sample output:
   First Name: first_name
   Last Name: last_name
   Annual Salary: $99,999

   If the employee does not exist, the procedure displays a proper message.

   Show your testing with an invalid ID and the other one with valid ID.

```
CREATE OR REPLACE PROCEDURE Calculate_Salary (empID
employees.employee_id%type) AS

emp employees%rowtype;
newSalary employees.salary%type;
yearsWorked INTEGER;

BEGIN
    SELECT * INTO emp
    FROM employees
    WHERE employee_id = empID;

    newSalary := emp.salary; --starting salary
    yearsWorked := trunc(MONTHS_BETWEEN(SYSDATE, emp.hire_date)
/ 12); --trunc so we dont round up an extra year

    FOR year IN 1..yearsWorked LOOP
        newSalary := newSalary * 1.05;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('First Name: ' || emp.first_name);
    DBMS_OUTPUT.PUT_LINE('Last Name: ' || emp.last_name);
    DBMS_OUTPUT.PUT_LINE('Annual Salary: $' ||
to_char(newSalary, '$99,999'));

    EXCEPTION WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee (ID: ' || empID || ')
does not exist.');

END;
/
```

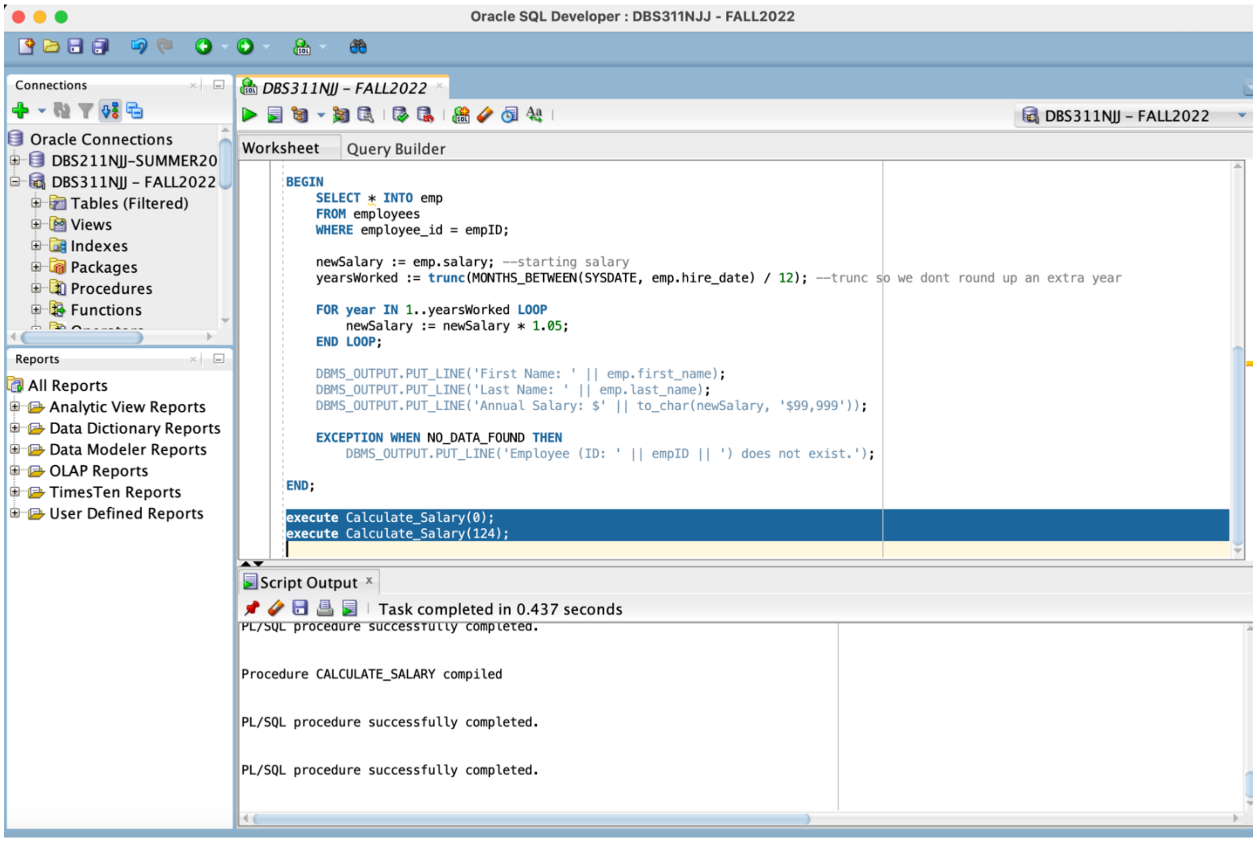
TESTING:

execute Calculate_Salary(0);

```
Employee (ID: 0) does not exist.
```

execute Calculate_Salary(124);

```
First Name: Kevin
Last Name: Mourgos
Annual Salary: $15,389
```



```
BEGIN
    SELECT * INTO emp
    FROM employees
    WHERE employee_id = empID;

    newSalary := emp.salary; --starting salary
    yearsWorked := trunc(MONTHS_BETWEEN(SYSDATE, emp.hire_date) / 12); --trunc so we dont round up an extra year

    FOR year IN 1..yearsWorked LOOP
        newSalary := newSalary * 1.05;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('First Name: ' || emp.first_name);
    DBMS_OUTPUT.PUT_LINE('Last Name: ' || emp.last_name);
    DBMS_OUTPUT.PUT_LINE('Annual Salary: $' || to_char(newSalary, '$99,999'));

    EXCEPTION WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee (ID: ' || empID || ') does not exist.');

END;

execute Calculate_Salary(0);
execute Calculate_Salary(124);
```

```
PL/SQL procedure successfully completed.


Procedure CALCULATE_SALARY compiled


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Question 3 next page

3. Write the code for the procedure called *Find_Prod_price*, that will search table Products and for a given Product ID will find its Description and display a message (note) regarding its List Price. This note will show *Cheap* for price below $200, *Not Expensive* for price between $200 and $500, otherwise will be *Expensive* (for price higher than $500). You need to take care of the wrong input (Product ID is invalid) as well.
   Use one IN parameter and two OUT parameters, then use PL/SQL block to show your output like (for a given ID of 31):

CPU:LGA2011-3 x 2,Form Factor:EATX,RAM Slots:16,Max RAM: is Not Expensive

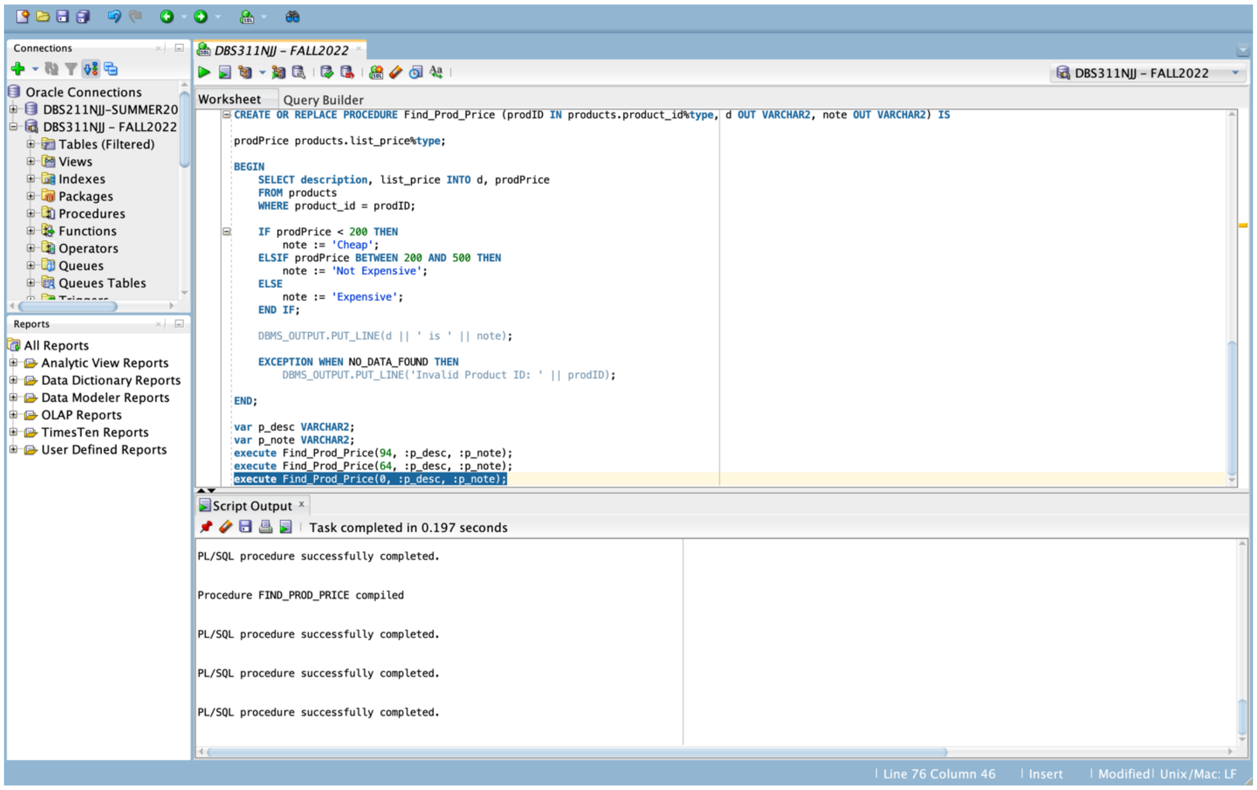
Show your testing with a Cheap, Expensive and Invalid product.

```
CREATE OR REPLACE PROCEDURE Find_Prod_Price (prodID IN
products.product_id%type, d OUT VARCHAR2, note OUT VARCHAR2) IS

prodPrice products.list_price%type;

BEGIN
    SELECT description, list_price INTO d, prodPrice
    FROM products
    WHERE product_id = prodID;

    IF prodPrice < 200 THEN
        note := 'Cheap';
    ELSIF prodPrice BETWEEN 200 AND 500 THEN
        note := 'Not Expensive';
    ELSE
        note := 'Expensive';
    END IF;

    DBMS_OUTPUT.PUT_LINE(d || ' is ' || note);

    EXCEPTION WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Product ID: ' || prodID);
```

```
END;
/
```

TESTING:

```
var p_desc VARCHAR2;
var p_note VARCHAR2;
execute Find_Prod_Price(94, :p_desc, :p_note);
Series:AV-GP,Type:5400RPM,Capacity:250GB,Cache:8MB is Cheap

execute Find_Prod_Price(64, :p_desc, :p_note);
CPU:G34 x 2,Form Factor:EATX,RAM Slots:16,Max RAM:512GB is
Expensive

execute Find_Prod_Price(0, :p_desc, :p_note);
Invalid Product ID: 0
```



```
CREATE OR REPLACE PROCEDURE Find_Prod_Price (prodID IN products.product_id%type, d OUT VARCHAR2, note OUT VARCHAR2) IS

prodPrice products.list_price%type;

BEGIN
    SELECT description, list_price INTO d, prodPrice
    FROM products
    WHERE product_id = prodID;

    IF prodPrice < 200 THEN
        note := 'Cheap';
    ELSIF prodPrice BETWEEN 200 AND 500 THEN
        note := 'Not Expensive';
    ELSE
        note := 'Expensive';
    END IF;

    DBMS_OUTPUT.PUT_LINE(d || ' is ' || note);

    EXCEPTION WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Product ID: ' || prodID);

END;

var p_desc VARCHAR2;
var p_note VARCHAR2;
execute Find_Prod_Price(94, :p_desc, :p_note);
execute Find_Prod_Price(64, :p_desc, :p_note);
execute Find_Prod_Price(0, :p_desc, :p_note);
```

Task completed in 0.197 seconds

```
PL/SQL procedure successfully completed.


Procedure FIND_PROD_PRICE compiled


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

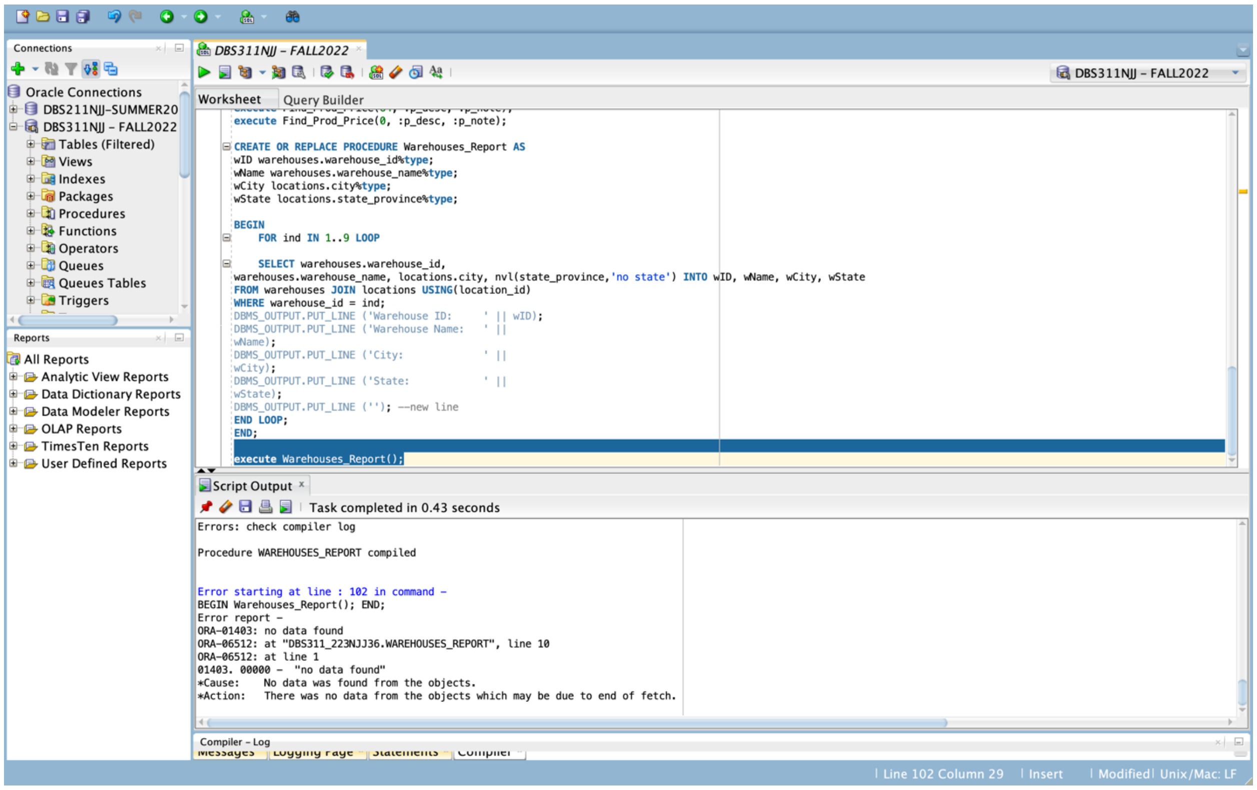
Question 4 next page

4. Write a stored procedure named *Warehouses_Report* to print the warehouse ID, warehouse name, and the city where the warehouse is located in the following format for ALL warehouses:

Warehouse ID:
Warehouse name:
City:
State:

If the value of state does not exist (null), display "no state".
The value of warehouse ID ranges from 1 to 9.
You can use a loop to find and display the information of each warehouse inside the loop. (Use a loop construct to answer this question. **Do not use cursors**.)

Note: Some of the above output displayed may not match exactly with your produced output. This is because the script file supplied to you was modified after creation of this lab requirements.



```
execute Find_Prod_Price(0, :p_desc, :p_note);

CREATE OR REPLACE PROCEDURE Warehouses_Report AS
wID warehouses.warehouse_id%type;
wName warehouses.warehouse_name%type;
wCity locations.city%type;
wState locations.state_province%type;

BEGIN
    FOR ind IN 1..9 LOOP

    SELECT warehouses.warehouse_id,
warehouses.warehouse_name, locations.city, nvl(state_province,'no state') INTO wID, wName, wCity, wState
FROM warehouses JOIN locations USING(location_id)
WHERE warehouse_id = ind;
DBMS_OUTPUT.PUT_LINE ('Warehouse ID:      ' || wID);
DBMS_OUTPUT.PUT_LINE ('Warehouse Name:    ' ||
wName);
DBMS_OUTPUT.PUT_LINE ('City:              ' ||
wCity);
DBMS_OUTPUT.PUT_LINE ('State:             ' ||
wState);
DBMS_OUTPUT.PUT_LINE (''); --new line
END LOOP;
END;

execute Warehouses_Report();
```

Script Output — Task completed in 0.43 seconds

```
Errors: check compiler log

Procedure WAREHOUSES_REPORT compiled


Error starting at line : 102 in command -
BEGIN Warehouses_Report(); END;
Error report -
ORA-01403: no data found
ORA-06512: at "DBS311_223NJJ36.WAREHOUSES_REPORT", line 10
ORA-06512: at line 1
01403. 00000 -  "no data found"
*Cause:    No data was found from the objects.
*Action:   There was no data from the objects which may be due to end of fetch.
```