[PROG8590-24F-Sec1-Relational](https://conestoga.desire2learn.com/d2l/home/1255613) Databases

NAME: VETRI VELAN ANANDAN

STUDENT ID: 8979936

Assignment No: 5

Date: 19/11/2024

***Question 1: [5 pts] – Anonymous Block***

Write an anonymous block using PL/SQL (program) to load the following details about customers: **First name**, **last name**, **email**, **country name** and **credit limit** and display them.

*Answer:*

DECLARE

-- Variables to hold customer details

fname oehr\_customers.CUST\_FIRST\_NAME%TYPE;

lname oehr\_customers.CUST\_LAST\_NAME%TYPE;

c\_email oehr\_customers.CUST\_EMAIL%TYPE;

c\_countryname oehr\_customers.NLS\_TERRITORY%TYPE;

c\_creditlimit oehr\_customers.CREDIT\_LIMIT%TYPE;

-- Cursor to fetch customer data

CURSOR customer\_cursor IS

SELECT CUST\_FIRST\_NAME, CUST\_LAST\_NAME, CUST\_EMAIL, NLS\_TERRITORY, CREDIT\_LIMIT

FROM oehr\_customers;

BEGIN

-- Loop through the cursor

FOR rec IN customer\_cursor LOOP

-- Assign values to variables

fname := rec.CUST\_FIRST\_NAME;

lname := rec.CUST\_LAST\_NAME;

c\_email := rec.CUST\_EMAIL;

c\_countryname := rec.NLS\_TERRITORY;

c\_creditlimit := rec.CREDIT\_LIMIT;

-- Display customer details

DBMS\_OUTPUT.PUT\_LINE(

'First Name: ' || fname ||

', Last Name: ' || lname ||

', Email: ' || c\_email ||

', Country: ' || c\_countryname ||

', Credit Limit: ' || c\_creditlimit

);

END LOOP;

END;

*Result:*

A screenshot of a computer

Description automatically generated

***Question 2: [5 pts] – Anonymous Block***

Write a PL/SQL program to determine the lowest of three numbers by using Nested-If statement. The numbers will be assigned in the declare section.

*Answer:*

DECLARE

num1 NUMBER := 100; -- First number

num2 NUMBER := 200; -- Second number

num3 NUMBER := 300; -- Third number

lowest NUMBER; -- Variable to store the lowest number

BEGIN

-- Nested-IF to determine the lowest number

IF num1 < num2 THEN

IF num1 < num3 THEN

lowest := num1;

ELSE

lowest := num3;

END IF;

ELSE

IF num2 < num3 THEN

lowest := num2;

ELSE

lowest := num3;

END IF;

END IF;

-- Display the result

DBMS\_OUTPUT.PUT\_LINE('The lowest number is: ' || lowest);

END;

*Result:*

A screenshot of a computer

Description automatically generated

***Question 3: [ 10 pts] – Stored Procedure***

Write a PL/SQL procedure **Insert\_promo** to insert a new row in the table **oehr\_promotions.**

The procedure should take in the promo\_id and the promo\_name as parameters.

The procedure should not allow a duplicate value in the promo\_name. To do so, you should:

- check the non-existence of the new promotion to add.

- Include exception handling

- Notify the user about the status of the insertion (Success/failure)

Write an anonymous block to run the procedure for both scenarios (Success/failure)

*Answer:*

CREATE OR REPLACE PROCEDURE Insert\_promo (

p\_promo\_id IN NUMBER, -- Promotion ID

p\_promo\_name IN VARCHAR2 -- Promotion Name

) IS

-- Declare a variable to check if promo\_name already exists

v\_count NUMBER;

BEGIN

-- Check if the promotion name already exists

SELECT COUNT(\*)

INTO v\_count

FROM oehr\_promotions

WHERE promo\_name = p\_promo\_name;

IF v\_count > 0 THEN

-- If promo\_name already exists, raise an exception

RAISE\_APPLICATION\_ERROR(-20001, 'Error: Promotion name already exists.');

ELSE

-- If promo\_name does not exist, insert the new promotion

INSERT INTO oehr\_promotions (promo\_id, promo\_name)

VALUES (p\_promo\_id, p\_promo\_name);

----COMMIT; //Not using commit as there is no mention to mkae changes to the DB

-- Notify user of success

DBMS\_OUTPUT.PUT\_LINE('Success: Promotion "' || p\_promo\_name || '" inserted successfully.');

END IF;

EXCEPTION

WHEN OTHERS THEN

-- Handle any exceptions

DBMS\_OUTPUT.PUT\_LINE('Failure: ' || SQLERRM);

END;

*Simple test query(to check if the procedure exists):*

BEGIN

-- Test case 1: Insert a new promotion

Insert\_promo(101, 'Summer Sale');

-- Test case 2: Try inserting a duplicate promotion (should raise an error)

Insert\_promo(102, 'Summer Sale');

END;

*Anonymous block to run the procedure for both scenarios (Success/failure):*

DECLARE

v\_promo\_id NUMBER := 101; -- Promo ID for the new promotion

v\_promo\_name VARCHAR2(100) := 'Winter Sale'; -- Promo Name for the new promotion

BEGIN

-- Scenario 1: Try inserting a new promotion

BEGIN

-- Call the procedure to insert the first promotion

Insert\_promo(v\_promo\_id, v\_promo\_name);

DBMS\_OUTPUT.PUT\_LINE('Test Case 1: Success - Promotion "' || v\_promo\_name || '" inserted successfully.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Test Case 1: Failure - ' || SQLERRM);

END;

-- Scenario 2: Try inserting the same promotion again (should raise an error)

BEGIN

-- Try inserting the same promotion again, which should fail

Insert\_promo(v\_promo\_id, v\_promo\_name);

DBMS\_OUTPUT.PUT\_LINE('Test Case 2: Success - Promotion "' || v\_promo\_name || '" inserted successfully.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Test Case 2: Failure - ' || SQLERRM);

END;

END;

*Result:*

Creating the procedure:

A white screen with a black text

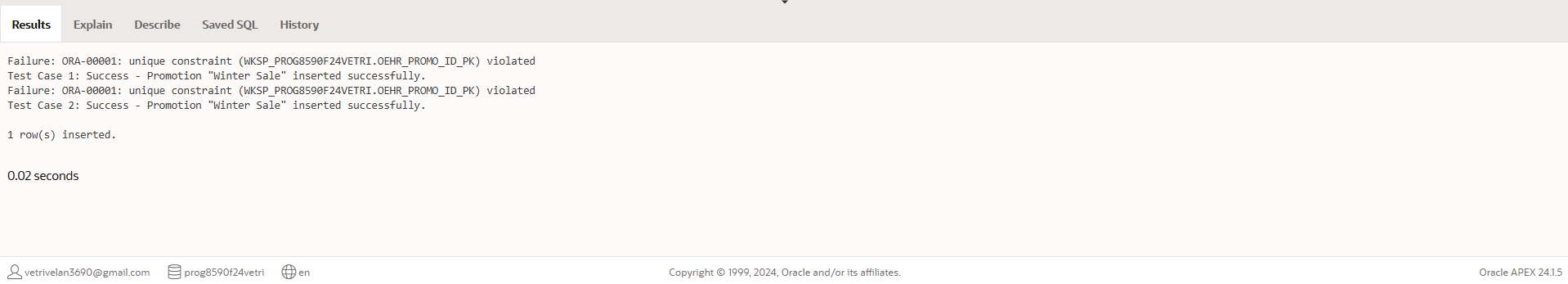
Description automatically generated with medium confidence

Simple test query:

A white screen with a white background

Description automatically generated with medium confidence

Anonymous block for both scenarios (Success/failure):



**Question 4: [ 10 pts] - Functions**

Write a PL/SQL function to increase the salary of a given employee. The program will search using the employee's ID and determine the new salary based on the job id:

- AD\_PRES: increase salary by 10%

- AD\_VP: increase salary by 8%

- IT\_Prog : increase salary by 15%

- Any other class: Nothing

Your program should store the result into a variable.

Include exception handling

Run the program hardcoding in a search for the employee id 110.

\*\*Hint: Use OEHR\_EMPLOYEES table.

*Answer:*

*CREATE OR REPLACE FUNCTION Increase\_Salary(p\_emp\_id IN NUMBER)*

*RETURN VARCHAR2*

*IS*

*-- Declare variables to store employee information*

*v\_job\_id oehr\_employees.JOB\_ID%TYPE;*

*v\_salary oehr\_employees.SALARY%TYPE;*

*v\_new\_salary oehr\_employees.SALARY%TYPE;*

*v\_message VARCHAR2(100);*

*BEGIN*

*-- Fetch the employee's job\_id and salary based on the employee id*

*SELECT JOB\_ID, SALARY*

*INTO v\_job\_id, v\_salary*

*FROM oehr\_employees*

*WHERE EMPLOYEE\_ID = p\_emp\_id;*

*-- Check job ID and calculate new salary based on it*

*IF v\_job\_id = 'AD\_PRES' THEN*

*v\_new\_salary := v\_salary \* 1.10; -- 10% increase*

*v\_message := 'Salary increased by 10%. New salary: ' || v\_new\_salary;*

*ELSIF v\_job\_id = 'AD\_VP' THEN*

*v\_new\_salary := v\_salary \* 1.08; -- 8% increase*

*v\_message := 'Salary increased by 8%. New salary: ' || v\_new\_salary;*

*ELSIF v\_job\_id = 'IT\_Prog' THEN*

*v\_new\_salary := v\_salary \* 1.15; -- 15% increase*

*v\_message := 'Salary increased by 15%. New salary: ' || v\_new\_salary;*

*ELSE*

*v\_new\_salary := v\_salary; -- No change for other jobs*

*v\_message := 'No salary increase for job ' || v\_job\_id || '. Current salary: ' || v\_new\_salary;*

*END IF;*

*-- Return the result message*

*RETURN v\_message;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*-- Handle case where no employee is found with the given ID*

*RETURN 'Error: Employee ID not found.';*

*WHEN OTHERS THEN*

*-- Handle other exceptions*

*RETURN 'Error: ' || SQLERRM;*

*END;*

*The program hardcoding in a search for the employee id 110:*

*DECLARE*

*v\_result VARCHAR2(100);*

*BEGIN*

*-- Call the function with employee ID 110*

*v\_result := Increase\_Salary(110);*

*-- Output the result message*

*DBMS\_OUTPUT.PUT\_LINE(v\_result);*

*END;*

*Result:*

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

---------------Thank you---------------