TITLE: PL/SQL ASSESMENT

AUTHOR: RAMAPRABA J

CREATED DATE: 14-07-2024

Question 1: Create a Procedure to Insert Employee Data Write a PL/SQL procedure named insert_employee to insert employee data into the EMPLOYEES table:

Table structure: EMPLOYEES (EMP_ID NUMBER, EMP_NAME VARCHAR2(100), DEPARTMENT VARCHAR2(50), SALARY NUMBER)

```
CREATE TABLE EMPLOYEES (EMP_ID     NUMBER PRIMARY KEY, EMP_NAME
VARCHAR2(100), DEPARTMENT VARCHAR2(50), SALARY   NUMBER);

CREATE OR REPLACE PROCEDURE insert_employee (
   p_emp_id     IN EMPLOYEES.EMP_ID%TYPE,
   p_emp_name   IN EMPLOYEES.EMP_NAME%TYPE,
   p_department IN EMPLOYEES.DEPARTMENT%TYPE,
   p_salary     IN EMPLOYEES.SALARY%TYPE
) AS
BEGIN
   INSERT INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY)
   VALUES (p_emp_id, p_emp_name, p_department, p_salary);
END insert_employee;
/

BEGIN
insert_employee(101, 'John Doe', 'Engineering', 75000);
END;
/
```

Question 2: Create a Procedure to Update Employee Salary Write a PL/SQL procedure named update_salary to update an employee's salary based on their current salary:

If the current salary is less than 5000, increase it by 10%.

If the current salary is between 5000 and 10000, increase it by 7.5%.

If the current salary is more than 10000, increase it by 5%.

```
CREATE OR REPLACE PROCEDURE update_salary (
    p_emp_id IN EMPLOYEES.EMP_ID%TYPE
) AS
    v_current_salary EMPLOYEES.SALARY%TYPE;
    v_new_salary EMPLOYEES.SALARY%TYPE;
BEGIN
    SELECT SALARY INTO v_current_salary FROM EMPLOYEES WHERE EMP_ID = p_emp_id;
   IF v_current_salary < 5000 THEN
        v_new_salary := v_current_salary * 1.10;  -- Increase by 10%
    ELSIF v_current_salary BETWEEN 5000 AND 10000 THEN
        v_new_salary := v_current_salary * 1.075;  -- Increase by 7.5%
    ELSE
        v_new_salary := v_current_salary * 1.05;  -- Increase by 5%
    END IF;
    UPDATE EMPLOYEES
    SET SALARY = v_new_salary
    WHERE EMP_ID = p_emp_id;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE ('Employee ' || p_emp_id || ' salary updated to ' || v_new_salary);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE ('Employee ID ' || p_emp_id || ' not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('Error: ' || SQLERRM);
END update_salary;
/

BEGIN
    update_salary(101);
END;
/
```

Question 3: Use a Cursor to Display Employee Names Write a PL/SQL block using a cursor to fetch and display all employee names from the EMPLOYEES table.

```
DECLARE
   v_emp_name EMPLOYEES.EMP_NAME%TYPE;
   CURSOR emp_cursor IS
      SELECT EMP_NAME FROM EMPLOYEES;
BEGIN
   OPEN emp_cursor;
   LOOP
      FETCH emp_cursor INTO v_emp_name;
      EXIT WHEN emp_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE ('Employee Name: ' || v_emp_name);
   END LOOP;
   CLOSE emp_cursor;
END;
/
```

Question 4: Create a View for Employees with High Salary Write a SQL statement to create a view named high_salary_employees that displays employees earning more than 10000.

```
CREATE OR REPLACE VIEW high_salary_employees AS
   SELECT *
   FROM EMPLOYEES
   WHERE SALARY >=10000;
```

Question 5: Create a Function to Calculate Bonus Write a PL/SQL function named calculate_bonus to calculate the bonus based on an employee's salary:

Employees earning less than 5000 get a bonus of 10% of their salary.

Employees earning between 5000 and 10000 get a bonus of 7.5% of their salary.

Employees earning more than 10000 get a bonus of 5% of their salary.

```
CREATE OR REPLACE FUNCTION calculate_bonus (
   p_salary IN NUMBER
) RETURN NUMBER
```

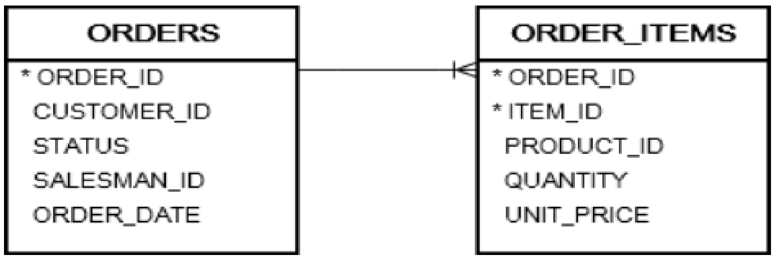
```
IS
    v_bonus NUMBER;
BEGIN
    IF p_salary < 5000 THEN
        v_bonus := p_salary * 0.10;
    ELSIF p_salary BETWEEN 5000 AND 10000 THEN
        v_bonus := p_salary * 0.075;

    ELSE
        v_bonus := p_salary * 0.05;
    END IF;
    RETURN v_bonus;
END calculate_bonus;
/
SELECT EMP_ID, EMP_NAME, SALARY, calculate_bonus(SALARY) AS BONUS
FROM EMPLOYEES;
```

Question 6: Create a Trigger to Log Employee Insertions Write a PL/SQL trigger named log_employee_insert to log whenever an employee is inserted into the EMPLOYEES table.

```
CREATE OR REPLACE TRIGGER log_employee_insert
AFTER INSERT ON EMPLOYEES
FOR EACH ROW
DECLARE
    v_log_message VARCHAR2(100);
BEGIN
    v_log_message := 'Employee inserted: ' || :NEW.EMP_ID || ' - ' || :NEW.EMP_NAME;
    DBMS_OUTPUT.PUT_LINE(v_log_message);
END log_employee_insert;
/
```

Question 7: Consider the orders and order_items tables from the sample database.



A) Create a view that returns the sales revenues by customers. The values of the credit column are 5% of the total sales revenues.

B) Write the PL/SQL query to develop an anonymous block which:

1. Reset the credit limits of all customers to zero.
2. Fetch customers sorted by sales in descending order and give them new credit limits from a budget of 1 million.

A).

```
CREATE VIEW Sales_Revenue_By_Customers AS

SELECT

   o.CUSTOMER_ID,

   SUM(oi.QUANTITY * oi.UNIT_PRICE) AS Total_Sales_Revenue,

   SUM(oi.QUANTITY * oi.UNIT_PRICE) * 0.05 AS Credit

FROM

   ORDERS o

JOIN

   ORDER_ITEMS oi ON o.ORDER_ID = oi.ORDER_ID

GROUP BY

   o.CUSTOMER_ID;
```

B).

```
DECLARE

   CURSOR customer_cursor IS

      SELECT CUSTOMER_ID, Total_Sales_Revenue

      FROM Sales_Revenue_By_Customers

      ORDER BY Total_Sales_Revenue DESC;


   customer_rec customer_cursor%ROWTYPE;

   budget NUMBER := 1000000;
   remaining_budget NUMBER := 1000000;
BEGIN
   UPDATE CUSTOMERS
   SET CREDIT_LIMIT = 0;
   OPEN customer_cursor;
   LOOP
      FETCH customer_cursor INTO customer_rec;
      EXIT WHEN customer_cursor%NOTFOUND;
      IF remaining_budget >= customer_rec.Total_Sales_Revenue * 0.05 THEN
         UPDATE CUSTOMERS
         SET CREDIT_LIMIT = customer_rec.Total_Sales_Revenue * 0.05
         WHERE CUSTOMER_ID = customer_rec.CUSTOMER_ID;
         remaining_budget := remaining_budget - (customer_rec.Total_Sales_Revenue * 0.05);
      ELSE
         UPDATE CUSTOMERS
         SET CREDIT_LIMIT = remaining_budget
         WHERE CUSTOMER_ID = customer_rec.CUSTOMER_ID;
         remaining_budget := 0;
         EXIT;
      END IF;
   END LOOP;
  CLOSE customer_cursor;
END;
/
```

Question 8: Write a program in PL/SQL to show the uses of implicit cursor without using any attribute.

```
DECLARE
   employee_id employees.employee_id%TYPE;
   first_name employees.first_name%TYPE;
   last_name employees.last_name%TYPE;
   email employees.email%TYPE;
   phone_number employees.phone_number%TYPE;
```

```
    hire_date employees.hire_date%TYPE;
    job_id employees.job_id%TYPE;
    salary employees.salary%TYPE;
    commission_pct employees.commission_pct%TYPE;
    manager_id employees.manager_id%TYPE;
    department_id employees.department_id%TYPE;
BEGIN
    -- Loop through all employees using implicit cursor
    FOR rec IN (SELECT * FROM employees) LOOP
        -- Fetch employee details
        employee_id := rec.employee_id;
        first_name := rec.first_name;
        last_name := rec.last_name;
        email := rec.email;
        phone_number := rec.phone_number;
        hire_date := rec.hire_date;
        job_id := rec.job_id;
        salary := rec.salary;
        commission_pct := rec.commission_pct;
        manager_id := rec.manager_id;
        department_id := rec.department_id;

        -- Display employee details
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_id);
        DBMS_OUTPUT.PUT_LINE('First Name: ' || first_name);
        DBMS_OUTPUT.PUT_LINE('Last Name: ' || last_name);
        DBMS_OUTPUT.PUT_LINE('Email: ' || email);
        DBMS_OUTPUT.PUT_LINE('Phone Number: ' || phone_number);
        DBMS_OUTPUT.PUT_LINE('Hire Date: ' || TO_CHAR(hire_date, 'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_id);
        DBMS_OUTPUT.PUT_LINE('Salary: ' || salary);
        DBMS_OUTPUT.PUT_LINE('Commission Pct: ' || commission_pct);
        DBMS_OUTPUT.PUT_LINE('Manager ID: ' || manager_id);
        DBMS_OUTPUT.PUT_LINE('Department ID: ' || department_id);
```

```
   END LOOP;
END;
/
```

Question 9: Write a program in PL/SQL to create a cursor displays the name and salary of each employee in the EMPLOYEES table whose salary is less than that specified by a passedin parameter value.

```
DECLARE

   CURSOR emp_cursor (salary_limit NUMBER) IS
      SELECT first_name, last_name, salary
      FROM employees
      WHERE salary < salary_limit;

   v_first_name employees.first_name%TYPE;
   v_last_name employees.last_name%TYPE;
   v_salary employees.salary%TYPE;

   v_salary_limit NUMBER := 60000; -- You can change this value as needed

BEGIN
   OPEN emp_cursor(v_salary_limit);
   LOOP
      FETCH emp_cursor INTO v_first_name, v_last_name, v_salary;
      EXIT WHEN emp_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('First Name: ' || v_first_name || ', Last Name: ' || v_last_name || ',
Salary: ' || v_salary);
   END LOOP;
   CLOSE emp_cursor;
END;
/
```

Question 10: Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE OR REPLACE TRIGGER prevent_duplicate_email
BEFORE INSERT OR UPDATE ON employees
FOR EACH ROW
DECLARE
   v_count INTEGER;
   duplicate_email EXCEPTION;
BEGIN
   SELECT COUNT(*)
   INTO v_count
   FROM employees
   WHERE email = :NEW.email
   AND employee_id != :NEW.employee_id;
   IF v_count > 0 THEN
      RAISE duplicate_email;
   END IF;
EXCEPTION
   WHEN duplicate_email THEN
      RAISE_APPLICATION_ERROR (-20001, 'Duplicate email address found: ' || :NEW.email);
END;
/
```

Question 11:Write a PL/SQL procedure for selecting some records from the database using some parameters as filters.
Consider that we are fetching details of employees from ib_employee table where salary is a parameter for filter.

```
CREATE OR REPLACE PROCEDURE GetEmployeesBySalary(
   p_salary IN NUMBER
) AS
BEGIN
```

```
   FOR rec IN (
       SELECT employee_id, first_name, last_name, email, phone_number, hire_date, job_id, salary,
commission_pct, manager_id, department_id
       FROM employee
       WHERE salary = p_salary
   ) LOOP
       DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.employee_id);
       DBMS_OUTPUT.PUT_LINE('First Name: ' || rec.first_name);
       DBMS_OUTPUT.PUT_LINE('Last Name: ' || rec.last_name);
       DBMS_OUTPUT.PUT_LINE('Email: ' || rec.email);
       DBMS_OUTPUT.PUT_LINE('Phone Number: ' || rec.phone_number);
       DBMS_OUTPUT.PUT_LINE('Hire Date: ' || rec.hire_date);
       DBMS_OUTPUT.PUT_LINE('Job ID: ' || rec.job_id);
       DBMS_OUTPUT.PUT_LINE('Salary: ' || rec.salary);
       DBMS_OUTPUT.PUT_LINE('Commission Pct: ' || rec.commission_pct);
       DBMS_OUTPUT.PUT_LINE('Manager ID: ' || rec.manager_id);
       DBMS_OUTPUT.PUT_LINE('Department ID: ' || rec.department_id);
       DBMS_OUTPUT.PUT_LINE('-----------------------------');
   END LOOP;
END;
/

BEGIN
   GetEmployeesBySalary(50000);
END;
/
```

Question 12: Write PL/SQL code block to increment the employee's salary by 1000 whose employee_id is 102

```
DECLARE
   v_new_salary EMPLOYEE.SALARY%TYPE;
BEGIN

   SELECT SALARY INTO v_new_salary
```

```
    FROM EMPLOYEE
    WHERE E_ID = 102;
    v_new_salary := v_new_salary + 1000;

    UPDATE EMPLOYEE
    SET SALARY = v_new_salary
    WHERE E_ID = 102;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Salary updated for employee ID 102.');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with ID 102.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```