DBMS QUESTIONS

Deepthi Tabitha Bennet (20S009)

1. Write a PL/SQL procedure to calculate the incentive achieved according to the specific sale limit.

Sample Output:

Sale achieved: 45000, incentive: 1800.
Sale achieved: 36000, incentive: 800.
Sale achieved: 28000, incentive: 500.

```
/*-----------------------------------------------------------------------*/
CREATE TABLE department(
dep_id VARCHAR2(50),
employee_id VARCHAR2(50)
);
/*-----------------------------------------------------------------------*/
DECLARE
   input number;
   PROCEDURE test1 (sal_achieve  NUMBER)
   IS
      incentive NUMBER := 0;
   BEGIN
      IF sal_achieve > 44000 THEN
         incentive := 1800;
      ELSIF sal_achieve > 32000 THEN
         incentive := 800;
      ELSE
         incentive := 500;
      END IF;
   DBMS_OUTPUT.NEW_LINE;
   DBMS_OUTPUT.PUT_LINE ('Sale achieved : ' || sal_achieve || ', incentive : ' || incentive
|| '.');
   END test1;

BEGIN
  input := :input;
  test1(input);
END;
/*-----------------------------------------------------------------------*/
```

NOT NEEDED

Anonymous Block → DECLARE, BEGIN, END
Procedure Block → IS, BEGIN, END

optional, just for formatting output

Anonymous Block,
to call the procedure, with a value
from user-input

calling the procedure

2. Write a PL/SQL program to count the number of employees in a specific department and check whether these departments have any vacancies or not. If any vacancies, how many vacancies are in that department.

Sample Output:

Enter value for new_dep_id: 20
old 6: get_dep_id := '&new_dep_id';
new 6: get_dep_id := '20';
The employees are in the department 20 is: 2
There are 43 vacancies in department 20

```
/*----------------------------------------------------------------------*/
CREATE TABLE department(
dep_id VARCHAR2(50),
employee_id VARCHAR2(50)
);
/*----------------------------------------------------------------------*/
INSERT INTO department(dep_id, employee_id) VALUES ('D1', 'E2');
INSERT INTO department(dep_id, employee_id) VALUES ('D3', 'E1');
INSERT INTO department(dep_id, employee_id) VALUES ('D1', 'E2');
INSERT INTO department(dep_id, employee_id) VALUES ('D2', 'E3');
/*----------------------------------------------------------------------*/
SELECT * FROM department;
/*----------------------------------------------------------------------*/
DECLARE
   DID varchar2(50);
   emp_count number;
   capacity number;
   vacancy number;
BEGIN
   DID := :DID;
   capacity := :capacity;

   SELECT COUNT(dep_id)  INTO emp_count FROM department WHERE (dep_id =
DID);
   dbms_output.put_line('Number of Employees : ' || emp_count);

   vacancy := capacity-emp_count;

   dbms_output.put_line('Number of Vacancies : ' || vacancy);
   RETURN;
END;
/*----------------------------------------------------------------------*/
```

Insert one by one

user input

math function

3. Write a program in PL/SQL Function to print the total count of prime numbers between ranges (like 1 to 50).

```
/*-----------------------------------------------------------------------*/
DECLARE
   i NUMBER(3);
   j NUMBER(3);
   end_number NUMBER;
BEGIN
   end_number := :end_number;
   dbms_output.Put_line('The prime numbers are:');
   dbms_output.new_line;

   i := 2;
   LOOP
      j := 2;
      LOOP
         EXIT WHEN( ( MOD(i, j) = 0 ) OR ( j = i ) );
         j := j + 1;
      END LOOP;
      IF( j = i )THEN
        dbms_output.Put(ill'   ');
      END IF;
      i := i + 1;
      exit WHEN i = end_number;
   END LOOP;
dbms_output.new_line;
END;
/*-----------------------------------------------------------------------*/
```

4. Write a PL/SQL program to Create trigger to increase the Salary of the employee by 10% bonus for new employee during insertion.

```
/*-------------------------------------------------------------------*/
CREATE TABLE Employee(
  EmployeeID number NOT NULL,
  Name varchar2(50) NOT NULL,
  Salary number NOT NULL
);
/*-------------------------------------------------------------------*/
CREATE OR REPLACE TRIGGER display_salary_changes
AFTER INSERT ON Employee
DECLARE
   emp_count NUMBER;
BEGIN
   SELECT count(EmployeeID) INTO emp_count FROM Employee;
   UPDATE Employee SET Salary = Salary + (Salary/10) WHERE EmployeeID =
emp_count;
END;
/*-------------------------------------------------------------------*/
// triggering the trigger
/*-------------------------------------------------------------------*/
INSERT INTO Employee VALUES (1, 'Amanda', 25000);
/*-------------------------------------------------------------------*/
INSERT INTO Employee VALUES (2, 'John', 10000);
/*-------------------------------------------------------------------*/
INSERT INTO Employee VALUES (3, 'Alice', 50000);
/*-------------------------------------------------------------------*/
SELECT * FROM Employee;
/*-------------------------------------------------------------------*/
```

increment by 10%

After each insert, make sure to display the table

IMPORTANT

insert in this order, one by one

this is a comment, don't run it :)

5. Write a program in PL/SQL – that calculates total sales by year, the schema contains unit price and quantity.

```
/*————————————————————————————————————————————————————————————*/
CREATE TABLE sales(
id VARCHAR2(50),
unit_price number,
quantity number,
year varchar2(50)
);
/*————————————————————————————————————————————————————————————*/
INSERT ALL
    INTO sales VALUES (1, 200, 5, 2020)
    INTO sales VALUES (2, 250, 25, 2021)
    INTO sales VALUES (3, 300, 10, 2022)
    INTO sales VALUES (4, 500, 90, 2020)
    INTO sales VALUES (5, 1000, 2, 2022)
SELECT * FROM dual;
/*————————————————————————————————————————————————————————————*/
SELECT sum(unit_price), year FROM sales GROUP BY year;
/*————————————————————————————————————————————————————————————*/
DECLARE
    sale_year varchar2(50);
    PROCEDURE Sum_Of_Sales(sale_year varchar2)
        IS
            sale_sum number;
        BEGIN
            SELECT sum(unit_price) INTO sale_sum FROM sales WHERE year = sale_year;
            dbms_output.put_line('year : ' || sale_year);
            dbms_output.put_line('sale : ' || sale_sum);
    END Sum_Of_Sales;

BEGIN
    sale_year := :sale_year;
    Sum_Of_Sales(sale_year);
END;
/*————————————————————————————————————————————————————————————*/
```

Normal SQL Query, not PL/SQL
Just in case it's asked ☺

→ calling the procedure

6. Write a program in PL/SQL to display a table based detail information for the employee of ID 149 from the employees table. (Tables are to be created base)

```
/*————————————————————————————————————————————————————————————————————————————*/
CREATE TABLE department(
dep_id VARCHAR2(50),
employee_id NUMBER,
employee_name VARCHAR2(50),
salary NUMBER
);
/*————————————————————————————————————————————————————————————————————————————*/
INSERT ALL
   INTO department VALUES ('D1', 147, 'Alice', 2500)
   INTO department VALUES ('D2', 148, 'Ashley', 1500)
   INTO department VALUES ('D1', 149, 'Holly', 7500)
   INTO department VALUES ('D1', 150, 'John', 1000)
   INTO department VALUES ('D3', 151, 'Peter', 2000)
SELECT * FROM dual;
/*————————————————————————————————————————————————————————————————————————————*/
DECLARE
   EID NUMBER;
   DID VARCHAR2(50);
   ENAME VARCHAR2(50);
   SAL NUMBER;

   PROCEDURE details(EID varchar2)
      IS
         sale_sum number;
      BEGIN
         SELECT dep_id, employee_name, salary INTO DID, ENAME, SAL FROM
department WHERE employee_ID = EID;
         dbms_output.put_line('Employee ID : ' || EID);
         dbms_output.put_line('Employee Name : ' || ENAME);
         dbms_output.put_line('Department ID : ' || DID);
         dbms_output.put_line('Salary : ' || SAL);
   END details;

BEGIN
   EID := :EID; ⟶ user input
   details(EID); ⟶ calling the procedure
END;
/*————————————————————————————————————————————————————————————————————————————*/
```

7. Write a program in PL/SQL to create an explicit cursor with for loop and display the person who is a clerk and gets salary above 10000.
Use the Schema- department_name, department_id, first_name, last_name, job_name, salary

```
/*————————————————————————————————————————————————————————————————————————————*/
CREATE TABLE department(
    dep_id VARCHAR2(50),
    dep_name VARCHAR2(50),
    employee_id NUMBER,
    jobname VARCHAR2(50),
    firstname VARCHAR2(50),
    lastname VARCHAR2(50),
    salary NUMBER
);
/*————————————————————————————————————————————————————————————————————————————*/
INSERT ALL
    INTO department VALUES ('D1', 'CSE', 1, 'clerk', 'Alice', 'Brown', 25000)
    INTO department VALUES ('D2', 'DS', 2, 'professor', 'Ashley', 'Johnson', 15000)
    INTO department VALUES ('D1', 'CSE', 3, 'professor', 'Holly', 'Simpson', 75000)
    INTO department VALUES ('D1', 'CSE', 4, 'clerk', 'John', 'Fallon', 9000)
    INTO department VALUES ('D3', 'IT', 5, 'clerk', 'Peter', 'Parker', 20000)
SELECT * FROM dual;
/*————————————————————————————————————————————————————————————————————————————*/
SELECT * FROM department;
/*————————————————————————————————————————————————————————————————————————————*/
DECLARE
    CURSOR emp_cur_detail IS
        SELECT dep_id, dep_name, employee_id, jobname, firstname, lastname, salary
            FROM department WHERE  jobname = 'clerk' AND salary > 10000;
BEGIN
    FOR emp_rec IN emp_cur_detail LOOP ⟶ equivalent to FOR LOOP
        dbms_output.put_line('Name: ' || emp_rec.firstname || ' ' || emp_rec.lastname || Chr(9)
                        || 'Department Name: ' || emp_rec.dep_name || Chr(9)
                        || 'Department ID: ' || emp_rec.dep_id ||Chr(9)        prints 9 whitespaces
                        || 'Job Name: ' || emp_rec.jobname || Chr(9)
                        || 'Salary: ' || emp_rec.salary);
    END LOOP;
END;
/*————————————————————————————————————————————————————————————————————————————*/
```

8. Create a PL/SQL block to increase salary of employees in the department id 50. Use the schema employee_id, first_name, last_name, department_id, salary

```
/*————————————————————————————————————————————————————————————————————————————*/
CREATE TABLE employee(
    dep_id VARCHAR2(50),
    dep_name VARCHAR2(50),
    employee_id NUMBER,
    jobname VARCHAR2(50),
    firstname VARCHAR2(50),
    lastname VARCHAR2(50),
    salary NUMBER
);
/*————————————————————————————————————————————————————————————————————————————*/
INSERT ALL
    INTO employee VALUES ('D1', 'CSE', 1, 'clerk', 'Alice', 'Brown', 25000)
    INTO employee VALUES ('D2', 'DS', 2, 'professor', 'Ashley', 'Johnson', 15000)
    INTO employee VALUES ('D1', 'CSE', 3, 'professor', 'Holly', 'Simpson', 75000)
    INTO employee VALUES ('D1', 'CSE', 4, 'clerk', 'John', 'Fallon', 9000)
    INTO employee VALUES ('D3', 'IT', 5, 'clerk', 'Peter', 'Parker', 20000)
SELECT * FROM dual;
/*————————————————————————————————————————————————————————————————————————————*/
SELECT * FROM employee;
/*————————————————————————————————————————————————————————————————————————————*/
CREATE OR REPLACE PROCEDURE UpdateSalary(DEPTID VARCHAR2, SAL
NUMBER) IS  when specifying parameters, don't use VARCHAR2(50); use VARCHAR2
BEGIN
    UPDATE employee SET Salary = Salary + SAL WHERE dep_id = DEPTID;
END;
/*————————————————————————————————————————————————————————————————————————————*/
DECLARE
    DEPTID VARCHAR2(50);
    SAL NUMBER;
BEGIN
    DEPTID := :DEPTID;
    SAL := :SAL;
    UpdateSalary(DEPTID, SAL);
END;
/*————————————————————————————————————————————————————————————————————————————*/
```

Anonymous Block,
→ to call the procedure, with a value from user-input

9. Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

Sample Output:

Table updated? Yes, incentive = 75

Table updated? Yes, incentive = 150

```
/*————————————————————————————————————————————————————————————————————————————*/
CREATE TABLE Employee(
  EmployeeID number NOT NULL,
  Name varchar2(50) NOT NULL,
  Salary number NOT NULL
);
/*————————————————————————————————————————————————————————————————————————————*/
INSERT ALL
    INTO Employee VALUES (1, 'Amanda', 25000)
    INTO Employee VALUES (2, 'John', 10000)
    INTO Employee VALUES (3, 'Alice', 50000)
SELECT * FROM dual;
/*————————————————————————————————————————————————————————————————————————————*/
DECLARE
    sal_achieve NUMBER;
    target_qty  NUMBER;
    emp_id NUMBER;
    PROCEDURE test1 (sal_achieve  NUMBER, target_qty  NUMBER, emp_id NUMBER)
    IS
        incentive NUMBER := 0;
        updated VARCHAR2(3) := 'No';
    BEGIN
        IF sal_achieve >= (target_qty) THEN incentive := (sal_achieve * 5 / 100);
        UPDATE Employee SET salary = salary + incentive  WHERE EmployeeID = emp_id;
        updated := 'Yes';
        END IF;
        DBMS_OUTPUT.PUT_LINE ('Table updated?  ' || updated || ', ' || 'incentive = ' ||
incentive || '.');
    END test1;

BEGIN
    emp_id := :emp_id;
    sal_achieve := :sal_achieve;
    target_qty := :target_qty;
    test1(sal_achieve, target_qty, emp_id);
END;
```

if they achieve more than the target, award an incentive

Assume incentive is 5% of achieved salary, since it is not given in the qn.

user input