**Database Management System Lab**

Code: PMDS506P

**Digital Assignment 5**

**Name: Soumyadeep Ganguly**

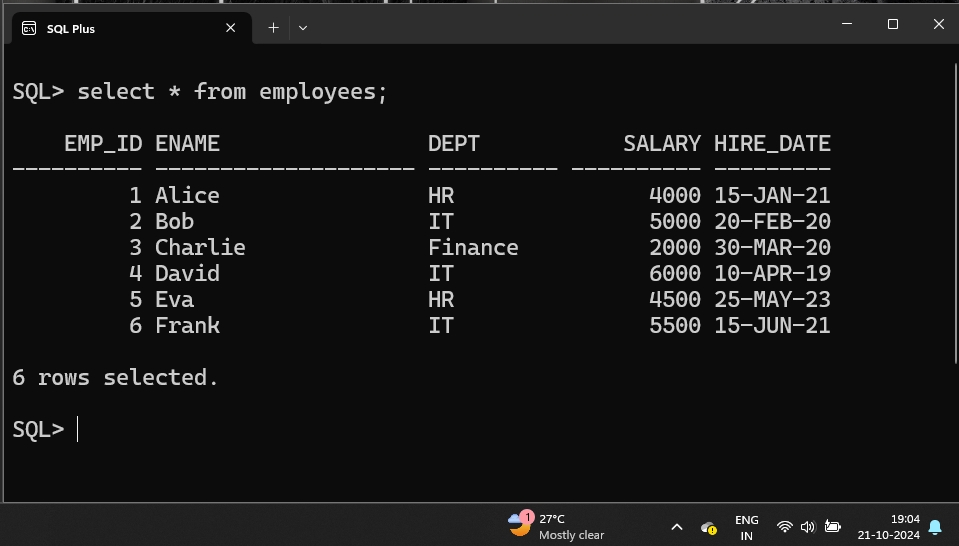
**Reg. No.: 24MDT0082**

**Course: M.Sc in Data Science**

**Q1. Create the following tables and answer the following questions. The employee’s table is,**

create table employees(emp\_id number(6), ename varchar(20), dept varchar(10), salary number(10), hire\_date date);

**Inserting Data:**

****insert into employees values(1, 'Alice', 'HR', 4000, '15-Jan-2021');

1. **Write a PL/SQL block that opens a cursor to fetch employee names. Display the message "Employee found" if any employee name is fetched using the %FOUND attribute.**

DECLARE

emp\_name employees.ename%TYPE;

CURSOR emp\_cur IS SELECT ename FROM employees;

BEGIN

OPEN emp\_cur;

FETCH emp\_cur INTO emp\_name;

IF emp\_cur%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Employee found: ' || emp\_name);

ELSE

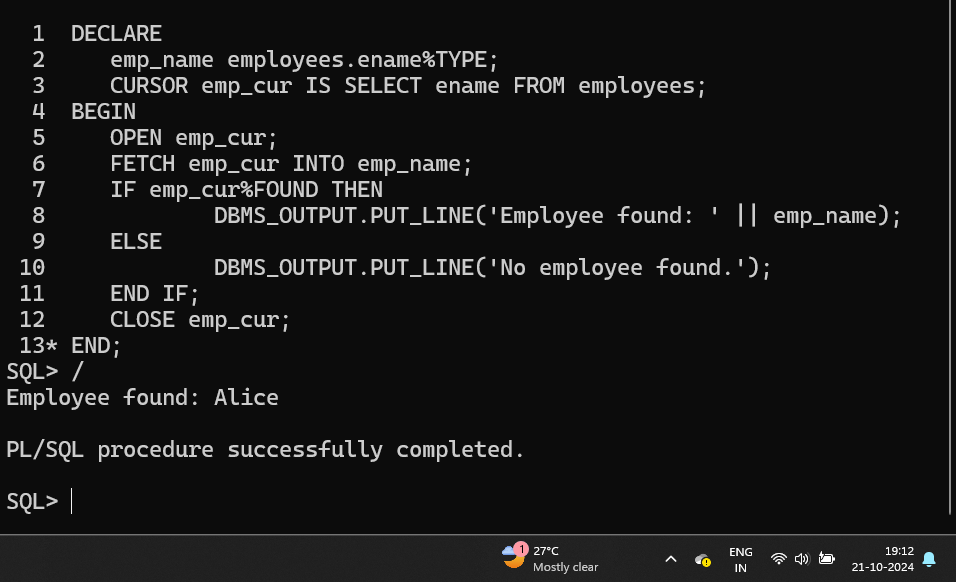
DBMS\_OUTPUT.PUT\_LINE('No employee found.');

END IF;

CLOSE emp\_cur;

END;

/



1. **Create a PL/SQL program that checks if there are any employees with a salary greater than 8000 using a cursor. If none are found, print a message stating "No employees with salary greater than 8000." (using %notfound)**

DECLARE

emp\_name employees.ename%TYPE;

emp\_sal employees.salary%TYPE;

CURSOR emp\_cur IS SELECT ename, salary FROM employees WHERE salary > 8000;

BEGIN

OPEN emp\_cur;

FETCH emp\_cur INTO emp\_name, emp\_sal;

IF emp\_cur%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employees with salary greater than 8000.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Employees with salary greater than 8000:');

LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || emp\_name || ', Salary: ' || emp\_sal);

FETCH emp\_cur INTO emp\_name,

emp\_sal;

EXIT WHEN emp\_cur%NOTFOUND;

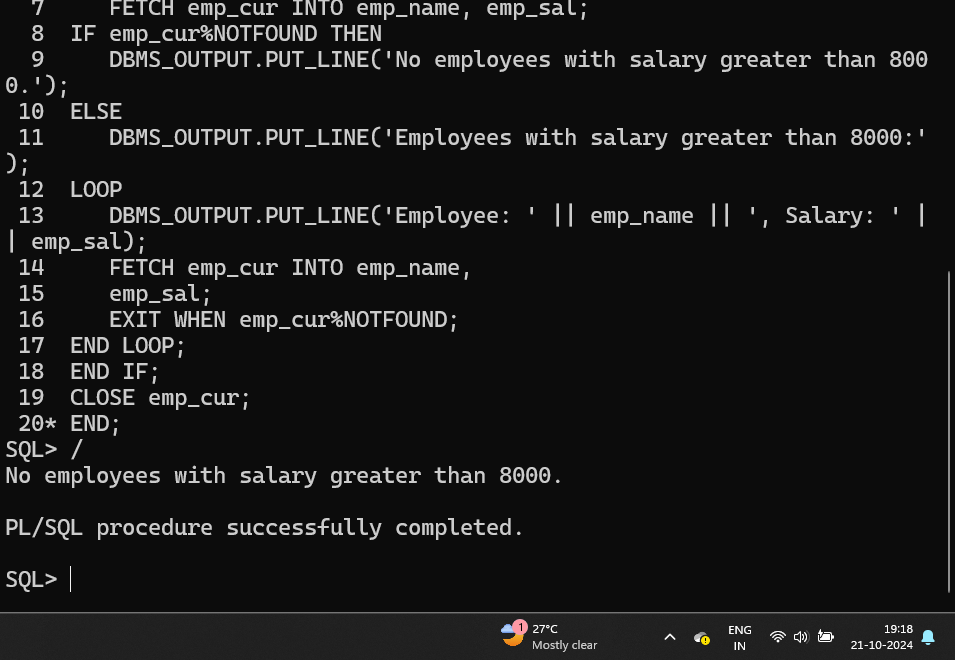
END LOOP;

END IF;

CLOSE emp\_cur;

END;

/



1. **Create a parameterized cursor that accepts a department name as an input. Write a PL/SQL block to fetch and display all employee names belonging to that department.**

DECLARE

emp\_name employees.ename%TYPE;

emp\_dept employees.dept%TYPE;

CURSOR emp\_cur(emp\_dept employees.dept%TYPE) IS

SELECT ename FROM employees WHERE dept =

emp\_dept;

BEGIN

emp\_dept := '&dept';

OPEN emp\_cur(emp\_dept);

FETCH emp\_cur INTO emp\_name;

IF emp\_cur%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employees found in the ' || emp\_dept || ' department.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Employees in the ' || emp\_dept || ' department:');

LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || emp\_name);

FETCH emp\_cur INTO emp\_name;

EXIT WHEN emp\_cur%NOTFOUND;

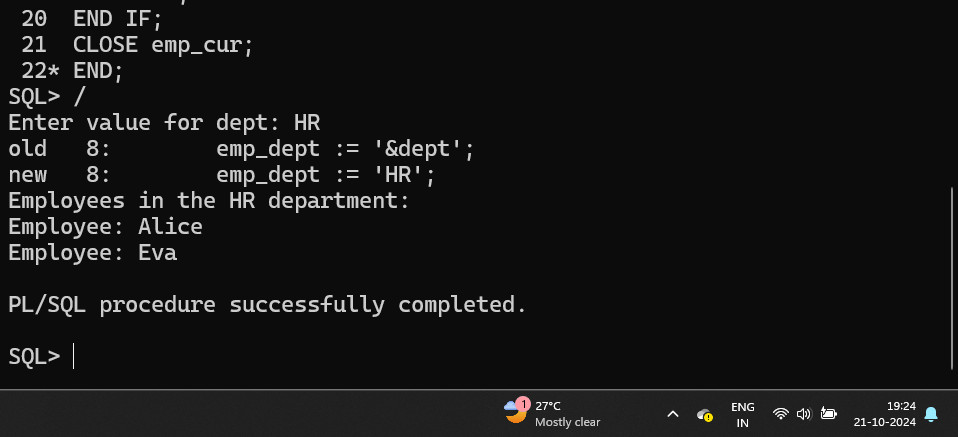
END LOOP;

END IF;

CLOSE emp\_cur;

END;

/



1. **Write a PL/SQL program that uses a cursor to fetch the first employee's details. If no employees are found, display a message "No employees available."**

DECLARE

emp\_name employees.ename%TYPE;

id employees.emp\_id%TYPE;

emp\_dept employees.dept%TYPE;

emp\_sal employees.salary%TYPE;

CURSOR emp\_cur IS SELECT ename, emp\_id, dept,

salary FROM employees;

BEGIN

OPEN emp\_cur;

FETCH emp\_cur INTO emp\_name, id, emp\_dept, emp\_sal;

IF emp\_cur%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employees available.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('First Employee Details:');

DBMS\_OUTPUT.PUT\_LINE('Name: ' || emp\_name);

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || id);

DBMS\_OUTPUT.PUT\_LINE('Department: ' || emp\_dept);

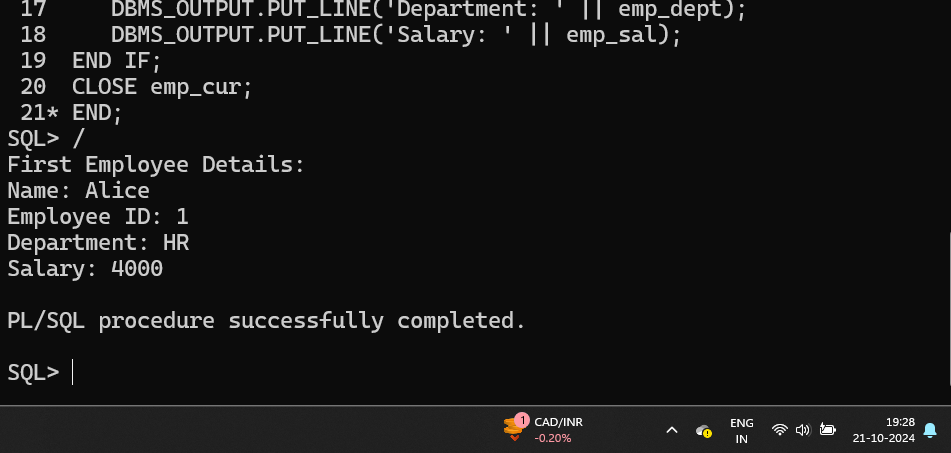
DBMS\_OUTPUT.PUT\_LINE('Salary: ' || emp\_sal);

END IF;

CLOSE emp\_cur;

END;

/



1. **Create a PL/SQL block that uses a cursor to find all employees in a specific department (e.g., 'HR') and increases their salaries by 5%. Display the old and new salaries for each updated employee.**

DECLARE

id employees.emp\_id%TYPE;

emp\_name employees.ename%TYPE;

emp\_salary employees.salary%TYPE;

new\_salary employees.salary%TYPE;

CURSOR emp\_cursor IS

SELECT emp\_id, ename, salary

FROM employees

WHERE dept = 'HR'

FOR UPDATE OF salary;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO id, emp\_name, emp\_salary;

EXIT WHEN emp\_cursor%NOTFOUND;

new\_salary := emp\_salary \* 1.05;

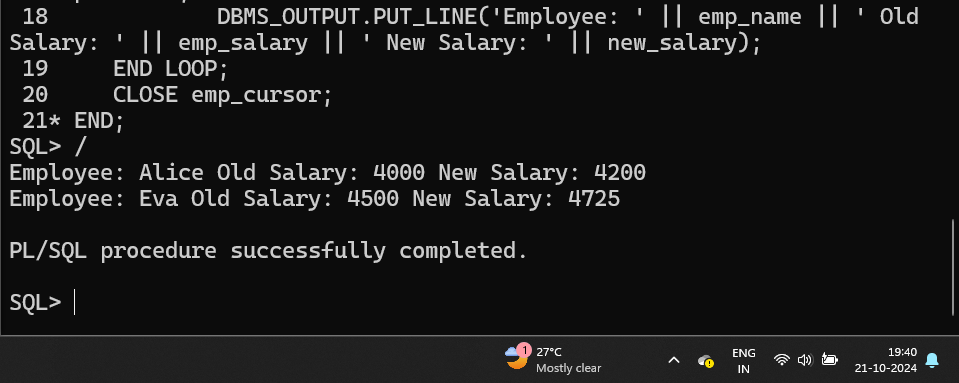
UPDATE employees SET salary = new\_salary WHERE CURRENT OF emp\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || emp\_name || ' Old Salary: ' || emp\_salary || ' New Salary: ' || new\_salary);

END LOOP;

CLOSE emp\_cursor;

END;

/

1. **Write a PL/SQL block that defines a parameterized cursor to fetch employee names with salaries within a specified range (e.g., between 4000 and 6000). Use input parameters for the range and display the names of employees that meet the criteria.**

DECLARE

v\_min\_salary NUMBER := &min\_salary;

v\_max\_salary NUMBER := &max\_salary;

emp\_name employees.ename%TYPE;

CURSOR emp\_cursor(p\_min\_salary NUMBER, p\_max\_salary NUMBER) IS

SELECT ename

FROM employees

WHERE salary BETWEEN p\_min\_salary AND p\_max\_salary;

BEGIN

OPEN emp\_cursor(v\_min\_salary, v\_max\_salary);

FETCH emp\_cursor INTO emp\_name;

IF emp\_cursor%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employees found with salary between ' || v\_min\_salary || ' and ' || v\_max\_salary);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Employees with salary between ' || v\_min\_salary || ' and ' || v\_max\_salary || ':');

LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || emp\_name);

FETCH emp\_cursor INTO emp\_name;

EXIT WHEN emp\_cursor%NOTFOUND;

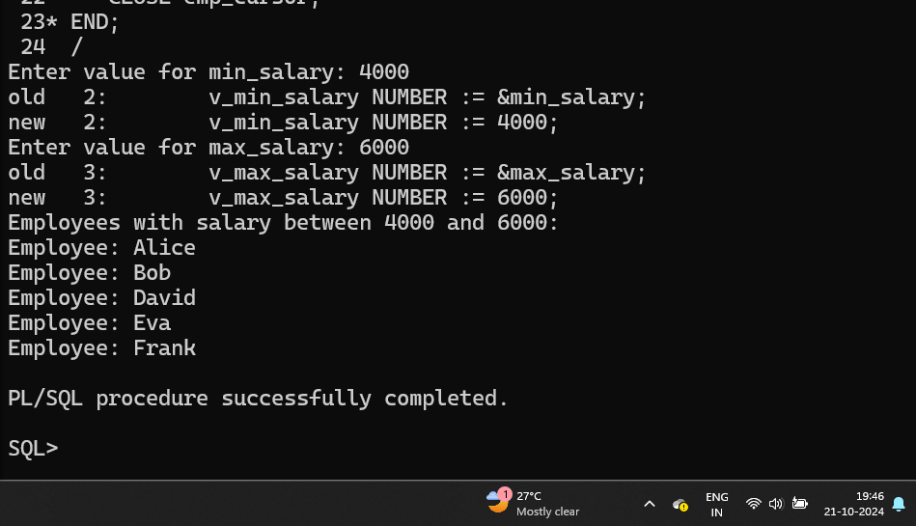
END LOOP;

END IF;

CLOSE emp\_cursor;

END;

/



1. **Explore what are triggers in the context of PL/SQL and give at least three examples for the same.**

**Triggers in PL/SQL:**

A trigger in PL/SQL is a stored procedure that is

automatically executed or fired in response to specific

events on a particular table or view. Triggers are used

to enforce business rules, maintain audit trails, or

perform actions automatically when certain database

events occur.

**Types of Triggers:**

1. **Row-Level Trigger: Executed once for each row affected by the triggering event.**
2. **Statement-Level Trigger: Executed once for the entire SQL statement, regardless of how many rows it affects.**
3. **Before vs. After Triggers:**

* BEFORE triggers fire before the DML (INSERT, UPDATE, DELETE) statement.
* AFTER triggers fire after the DML statement.

**Triggers can be fired by:**

1. **DML events: INSERT, UPDATE, or DELETE.**
2. **DDL events: CREATE, ALTER, or DROP (though DDL triggers are less common in basic business logic).**
3. **Database events: such as logon, logoff, or startup events.**