Database Management System Lab

Code: PMDS506P

Digital Assignment 5

Name: Soumyadeep Ganguly

Reg. No.: 24MDT0082

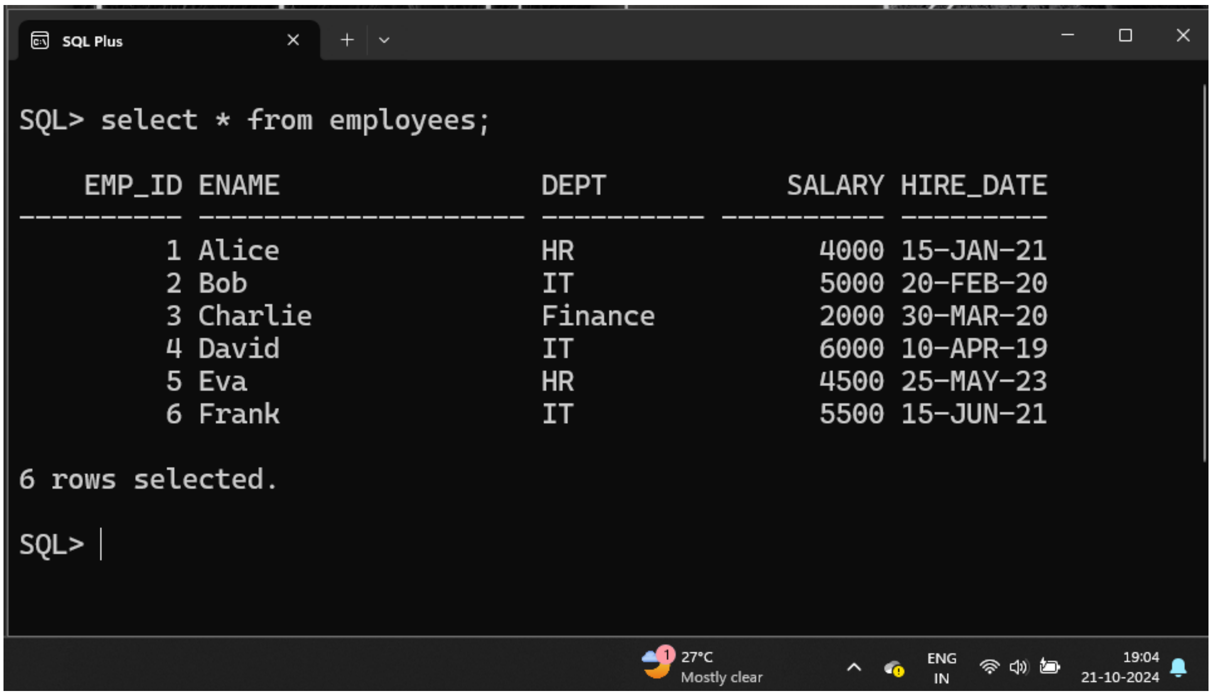
Course: M.Sc in Data Science

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:

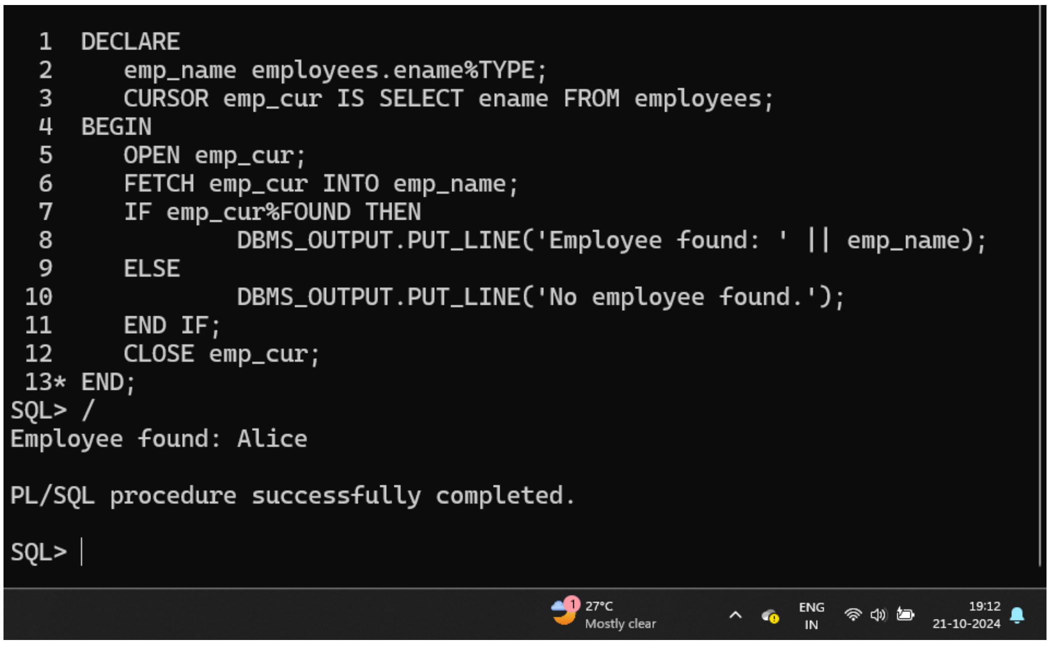
insert into employees values(1, 'Alice', 'HR', 4000, '15-Jan-2021');



```
SQL> select * from employees;

    EMP_ID ENAME                DEPT           SALARY HIRE_DATE
---------- -------------------- ---------- ---------- ---------
         1 Alice                HR               4000 15-JAN-21
         2 Bob                  IT               5000 20-FEB-20
         3 Charlie              Finance          2000 30-MAR-20
         4 David                IT               6000 10-APR-19
         5 Eva                  HR               4500 25-MAY-23
         6 Frank                IT               5500 15-JUN-21

6 rows selected.

SQL>
```

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.**

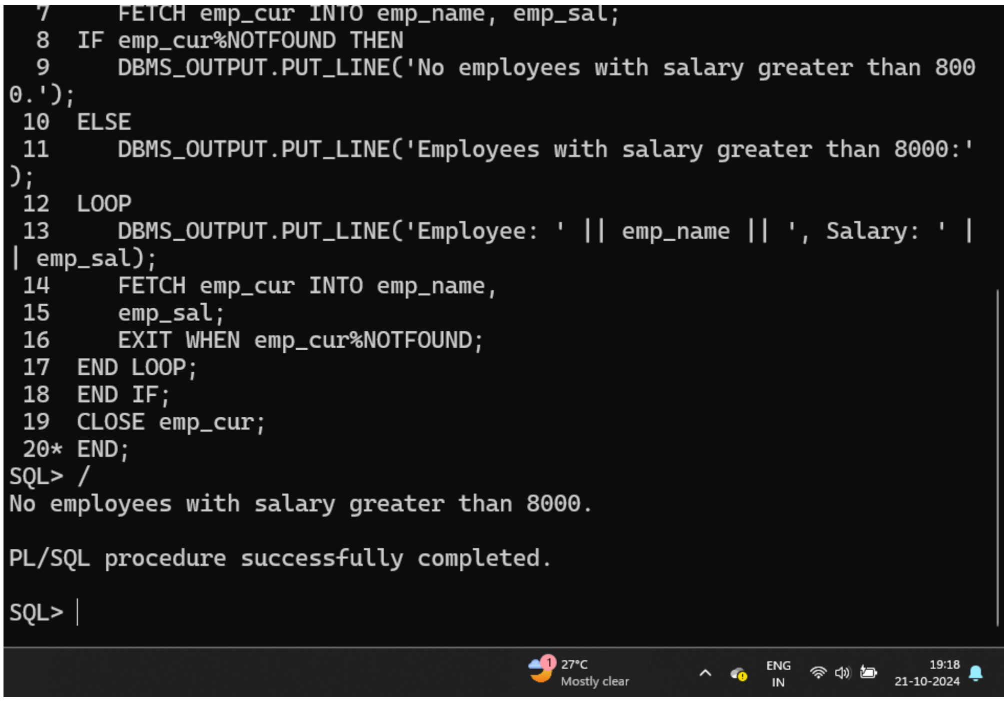
```
DECLARE
        emp_name employees.ename%TYPE;
        CURSOR emp_cur IS SELECT ename FROM employees;
BEGIN
        OPEN emp_cur;
        FETCH emp_cur INTO emp_name;
        IF emp_cur%FOUND THEN
                DBMS_OUTPUT.PUT_LINE('Employee found: ' || emp_name);
        ELSE
                DBMS_OUTPUT.PUT_LINE('No employee found.');
        END IF;
        CLOSE emp_cur;
END;
/
```



```
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;
SQL> /
Employee found: Alice

PL/SQL procedure successfully completed.

SQL>
```

2. **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)**

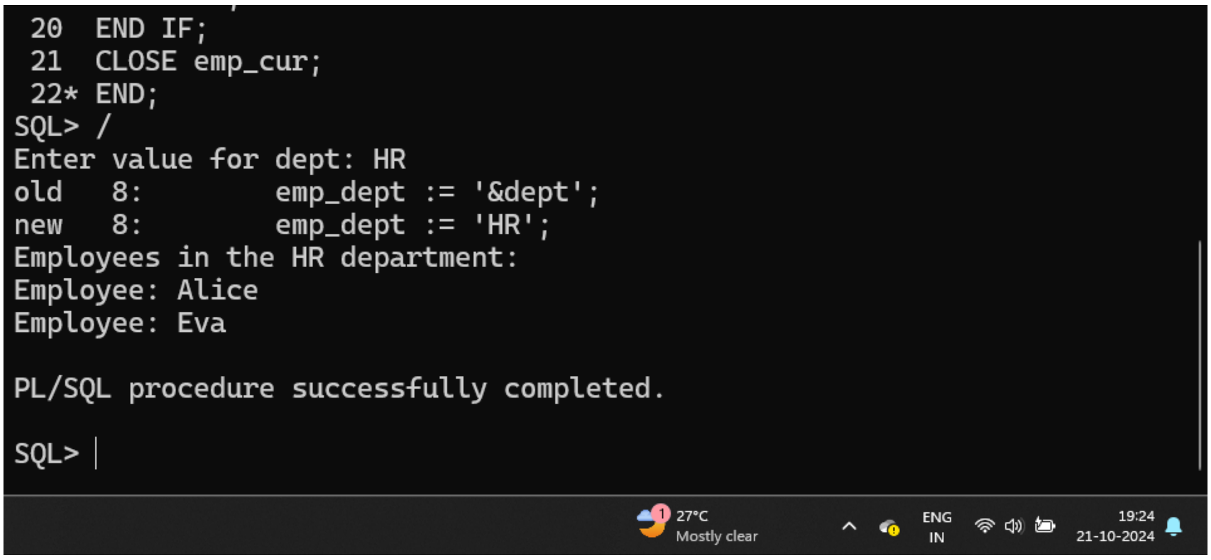
```
DECLARE
        emp_name employees.ename%TYPE;
        emp_sal employees.salary%TYPE;
        CURSOR emp_cur IS SELECT ename, salary FROM employees WHERE salary > 8000;
BEGIN
        OPEN emp_cur;
        FETCH emp_cur INTO emp_name, emp_sal;
IF emp_cur%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No 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;
/
```



```
    FETCH emp_cur INTO emp_name, emp_sal;
IF emp_cur%NOTFOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employees with salary greater than 800
0.');
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;
SQL> /
No employees with salary greater than 8000.

PL/SQL procedure successfully completed.

SQL>
```

3. **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.**

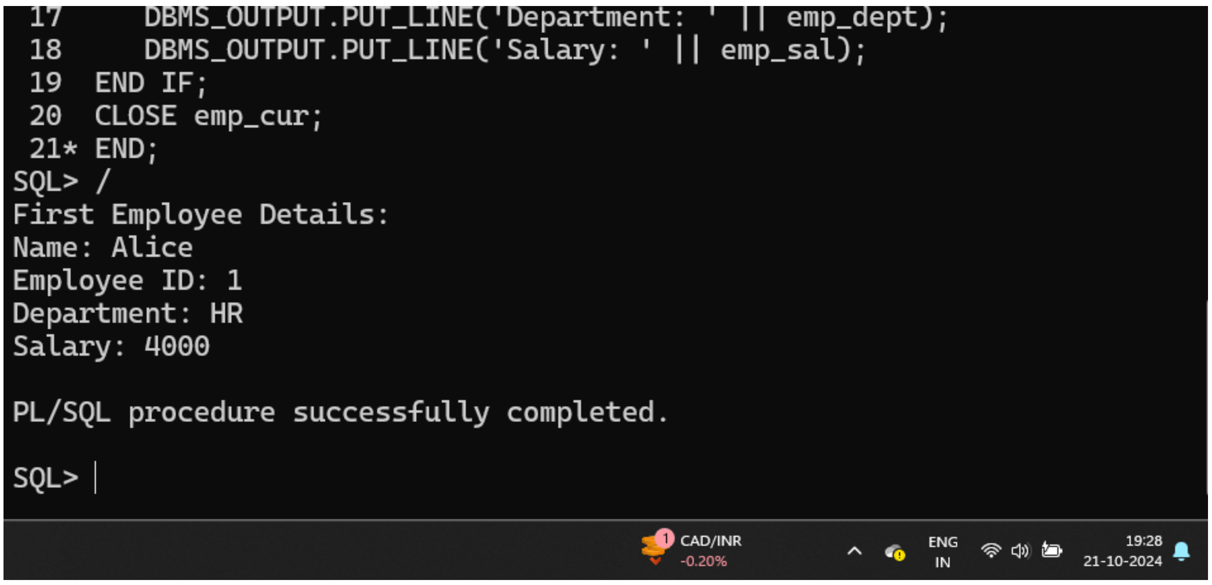
```
DECLARE
        emp_name employees.ename%TYPE;
        emp_dept employees.dept%TYPE;
        CURSOR emp_cur(emp_dept employees.dept%TYPE) IS
        SELECT ename FROM employees WHERE dept =
        emp_dept;
BEGIN
        emp_dept := '&dept';
        OPEN emp_cur(emp_dept);
        FETCH emp_cur 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;
/
```



```
END IF;
CLOSE emp_cur;
END;
SQL> /
Enter value for dept: HR
old   8:         emp_dept := '&dept';
new   8:         emp_dept := 'HR';
Employees in the HR department:
Employee: Alice
Employee: Eva

PL/SQL procedure successfully completed.

SQL>
```

4. 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."

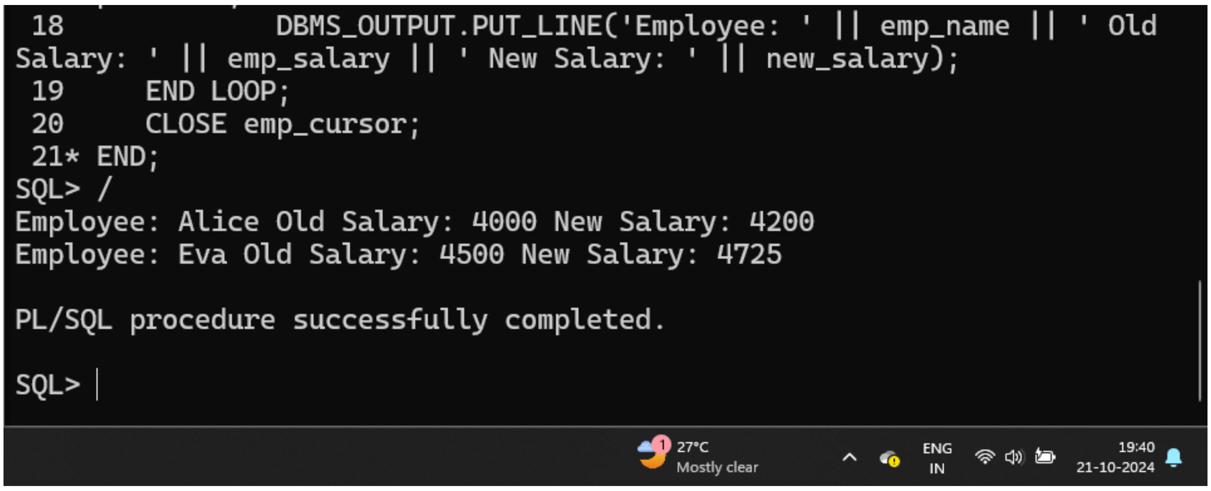
```
DECLARE
    emp_name employees.ename%TYPE;
    id employees.emp_id%TYPE;
    emp_dept employees.dept%TYPE;
    emp_sal employees.salary%TYPE;
CURSOR emp_cur IS SELECT ename, emp_id, dept,
salary FROM employees;
BEGIN
    OPEN emp_cur;
    FETCH emp_cur INTO emp_name, id, emp_dept, emp_sal;
IF emp_cur%NOTFOUND 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;
/
```



```
      DBMS_OUTPUT.PUT_LINE('Department: ' || emp_dept);
      DBMS_OUTPUT.PUT_LINE('Salary: ' || emp_sal);
   END IF;
   CLOSE emp_cur;
   END;
SQL> /
First Employee Details:
Name: Alice
Employee ID: 1
Department: HR
Salary: 4000

PL/SQL procedure successfully completed.

SQL>
```

5. **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.**

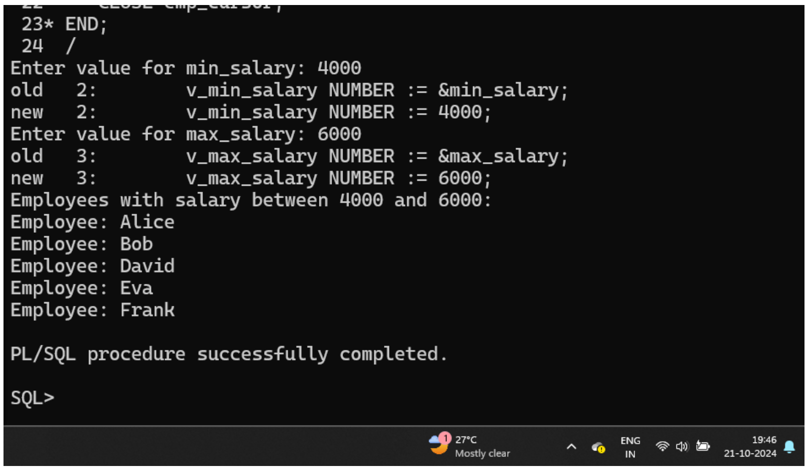
```
DECLARE
    id employees.emp_id%TYPE;
    emp_name employees.ename%TYPE;
    emp_salary employees.salary%TYPE;
    new_salary employees.salary%TYPE;
    CURSOR emp_cursor IS
            SELECT emp_id, ename, salary
            FROM employees
            WHERE dept = 'HR'
            FOR UPDATE OF 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;
/
```



```
               DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_name || ' Old
Salary: ' || emp_salary || ' New Salary: ' || new_salary);
        END LOOP;
        CLOSE emp_cursor;
     END;
SQL> /
Employee: Alice Old Salary: 4000 New Salary: 4200
Employee: Eva Old Salary: 4500 New Salary: 4725

PL/SQL procedure successfully completed.

SQL>
```

6. **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;
/
```



```
END;
/
Enter value for min_salary: 4000
old   2:        v_min_salary NUMBER := &min_salary;
new   2:        v_min_salary NUMBER := 4000;
Enter value for max_salary: 6000
old   3:        v_max_salary NUMBER := &max_salary;
new   3:        v_max_salary NUMBER := 6000;
Employees with salary between 4000 and 6000:
Employee: Alice
Employee: Bob
Employee: David
Employee: Eva
Employee: Frank

PL/SQL procedure successfully completed.

SQL>
```

7. **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.**