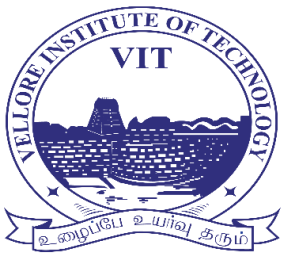

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND INFORMATION SYSTEMS

FALL SEMESTER 2024-2025

PMCA503P – DATABASE SYSTEMS LAB

CYCLESHEET – PL/SQL

SUBMITTED ON: 03 – OCT - 2024

SUBMITTED BY-

AKASH KUMAR BANIK

PROGRAM: MCA

REGISTER No.: 24MCA0242

CYCLESHEET – PL/SQL

Consider the following relational database schema for teaching-learning process in a university.

(Source: Database Systems – Coronel & Morris)

PROFESSOR(Prof_id, Prof_name, Email, Mobile, Specialty, Dept_id)

SCHOOL(SCode, Scl_name, Prof_id, Location)

DEPARTMENT(Dept_id, Dname, SCode, Prof_id)

COURSE(Crs_code, Crs_name, Description, Credits, Hours)

CLASS(Cls_code, Slot, Stime, Etime, Crs_code, Prof_id, Room_no, Sem_code, Day_of_week)

SEMESTER(Sem_code, Term, Year, Sdate, Edate)

STUDENT(Reg_no, Sname, Address, DoB, Email, Mobile, Dept_id, Prof_id)

ENROLL(Cls_code, Reg_no, Enroll_time, Grade)

STUDENT_VISA(Reg_no, Visa_status)

PROGRAMME(Prog_code, Prog_name, Prog_preamble, Scode, Dept_id)

The primary keys are underlined and foreign keys are self-explanatory. The Dept_id column in professor table stands for the department the professor belongs to and Prof_id column in the school table stands for the professor who chairs the school, the same column in the department table stands for the professor who heads the department, the domain of Term column in semester table is {Winter, Fall}.

CYCLESHEET – PL/SQL

1. Write a PL/SQL block to get the student register number and print the student details such as sname, address, dob, email and mobile number.

CODE:

```
DECLARE
   v_reg_no STUDENT_24MCA0242.Reg_no%TYPE;
   v_sname STUDENT_24MCA0242.Sname%TYPE;
   v_address STUDENT_24MCA0242.Address%TYPE;
   v_dob STUDENT_24MCA0242.DoB%TYPE;
   v_email STUDENT_24MCA0242.Email%TYPE;
   v_mobile STUDENT_24MCA0242.Mobile%TYPE;
BEGIN
   v_reg_no := '&Enter_Student_Reg_No';
   SELECT Sname, Address, DoB, Email, Mobile
   INTO v_sname, v_address, v_dob, v_email, v_mobile
   FROM STUDENT_24MCA0242 WHERE Reg_no = v_reg_no;
   DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_sname);
   DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);
   DBMS_OUTPUT.PUT_LINE('Date of Birth: ' || v_dob);
   DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
   DBMS_OUTPUT.PUT_LINE('Mobile: ' || v_mobile);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No student found with Reg_no: ' || v_reg_no);
   WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

OUTPUT:

```
SQL> DECLARE
  2      v_reg_no STUDENT_24MCA0242.Reg_no%TYPE;
  3      v_sname STUDENT_24MCA0242.Sname%TYPE;
  4      v_address STUDENT_24MCA0242.Address%TYPE;
  5      v_dob STUDENT_24MCA0242.DoB%TYPE;
  6      v_email STUDENT_24MCA0242.Email%TYPE;
  7      v_mobile STUDENT_24MCA0242.Mobile%TYPE;
  8  BEGIN
  9      v_reg_no := '&Enter_Student_Reg_No';
 10      SELECT Sname, Address, DoB, Email, Mobile
 11      INTO v_sname, v_address, v_dob, v_email, v_mobile
 12      FROM STUDENT_24MCA0242 WHERE Reg_no = v_reg_no;
 13
 14      DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_sname);
 15      DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);
 16      DBMS_OUTPUT.PUT_LINE('Date of Birth: ' || v_dob);
 17      DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
 18      DBMS_OUTPUT.PUT_LINE('Mobile: ' || v_mobile);
 19  EXCEPTION
 20      WHEN NO_DATA_FOUND THEN
 21          DBMS_OUTPUT.PUT_LINE('No student found with Reg_no: ' || v_reg_no);
 22      WHEN OTHERS THEN
 23          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 24  END;
 25  /
Enter value for enter_student_reg_no: R0004
old   9:      v_reg_no := '&Enter_Student_Reg_No';
new   9:      v_reg_no := 'R0004';
Student Name: Pooja Reddy
Address: Hyderabad, India
Date of Birth: 15-FEB-96
Email: pooja.reddy@student.edu
Mobile: 9000652000

PL/SQL procedure successfully completed.
```

2. Write a PL/SQL block the get the professor id and update the mobile number of the professor.

CODE:

```
DECLARE
   v_prof_id PROFESSOR_24MCA0242.Prof_id%TYPE;
   v_new_mobile PROFESSOR_24MCA0242.Mobile%TYPE;
BEGIN
   v_prof_id := '&Enter_Professor_ID';
   v_new_mobile := '&Enter_New_Mobile_Number';
```

```
    UPDATE PROFESSOR_24MCA0242
    SET Mobile = v_new_mobile
    WHERE Prof_id = v_prof_id;
    IF SQL%ROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('No professor found with Prof_id: ' || v_prof_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Mobile number updated successfully for Prof_id: ' ||
v_prof_id);
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

OUTPUT:

```
SQL> DECLARE
  2      v_prof_id PROFESSOR_24MCA0242.Prof_id%TYPE;
  3      v_new_mobile PROFESSOR_24MCA0242.Mobile%TYPE;
  4  BEGIN
  5      v_prof_id := '&Enter_Professor_ID';
  6      v_new_mobile := '&Enter_New_Mobile_Number';
  7      UPDATE PROFESSOR_24MCA0242
  8      SET Mobile = v_new_mobile
  9      WHERE Prof_id = v_prof_id;
 10
 11      IF SQL%ROWCOUNT = 0 THEN
 12          DBMS_OUTPUT.PUT_LINE('No professor found with Prof_id: ' || v_prof_id);
 13      ELSE
 14          DBMS_OUTPUT.PUT_LINE('Mobile number updated successfully for Prof_id: ' || v_prof_id);
 15      END IF;
 16  EXCEPTION
 17      WHEN OTHERS THEN
 18          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 19  END;
 20  /
Enter value for enter_professor_id: P0003
old   5:      v_prof_id := '&Enter_Professor_ID';
new   5:      v_prof_id := 'P0003';
Enter value for enter_new_mobile_number: 6008462591
old   6:      v_new_mobile := '&Enter_New_Mobile_Number';
new   6:      v_new_mobile := '6008462591';
Mobile number updated successfully for Prof_id: P0003

PL/SQL procedure successfully completed.
```

3. Write a PL/SQL procedure to display the message as ‘Excellent’, ‘Good’, and ‘Fair’ depending on the Grade of a student in a course.

CODE:

```
DECLARE
  v_grade ENROLL_24MCA0242.Grade%TYPE;
BEGIN
  SELECT Grade
  INTO v_grade
  FROM ENROLL_24MCA0242
  WHERE Reg_no = '&Enter_Student_Reg_No'
  AND Cls_code = '&Enter_Class_Code';
  CASE v_grade
    WHEN 'S' THEN
      DBMS_OUTPUT.PUT_LINE('Grade is S: Excellent');
    WHEN 'A' THEN
      DBMS_OUTPUT.PUT_LINE('Grade is A: Very Good');
    WHEN 'B' THEN
      DBMS_OUTPUT.PUT_LINE('Grade is B: Good');
    WHEN 'C' THEN
      DBMS_OUTPUT.PUT_LINE('Grade is C: Fair');
    WHEN 'D' THEN
      DBMS_OUTPUT.PUT_LINE('Grade is D: Needs Improvement');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Invalid Grade');
  END CASE;
```

END;

/

OUTPUT:

```
SQL> DECLARE
  2      v_grade ENROLL_24MCA0242.Grade%TYPE;
  3  BEGIN
  4      SELECT Grade
  5      INTO v_grade
  6      FROM ENROLL_24MCA0242
  7      WHERE Reg_no = '&Enter_Student_Reg_No'
  8      AND Cls_code = '&Enter_Class_Code';
  9
 10      CASE v_grade
 11          WHEN 'S' THEN
 12              DBMS_OUTPUT.PUT_LINE('Grade is S: Excellent');
 13          WHEN 'A' THEN
 14              DBMS_OUTPUT.PUT_LINE('Grade is A: Very Good');
 15          WHEN 'B' THEN
 16              DBMS_OUTPUT.PUT_LINE('Grade is B: Good');
 17          WHEN 'C' THEN
 18              DBMS_OUTPUT.PUT_LINE('Grade is C: Fair');
 19          WHEN 'D' THEN
 20              DBMS_OUTPUT.PUT_LINE('Grade is D: Needs Improvement');
 21          ELSE
 22              DBMS_OUTPUT.PUT_LINE('Invalid Grade');
 23      END CASE;
 24  END;
 25  /
Enter value for enter_student_reg_no: R0001
old   7:      WHERE Reg_no = '&Enter_Student_Reg_No'
new   7:      WHERE Reg_no = 'R0001'
Enter value for enter_class_code: CLS01
old   8:      AND Cls_code = '&Enter_Class_Code';
new   8:      AND Cls_code = 'CLS01';
Grade is A: Very Good

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL procedure to print the number of ‘S’ grades that a student has obtained.

CODE:

DECLARE

v_reg_no ENROLL_24MCA0242.Reg_no%TYPE;

v_s_grade_count NUMBER;

BEGIN

v_reg_no := '&Enter_Student_Reg_No';

SELECT COUNT(*)

INTO v_s_grade_count

FROM ENROLL_24MCA0242

```
    WHERE Reg_no = v_reg_no AND Grade = 'S';
    DBMS_OUTPUT.PUT_LINE('Number of "S" Grades: ' || v_s_grade_count);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No grades found for Reg_no: ' || v_reg_no);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

OUTPUT:

```
SQL> DECLARE
  2      v_reg_no ENROLL_24MCA0242.Reg_no%TYPE;
  3      v_s_grade_count NUMBER;
  4  BEGIN
  5      v_reg_no := '&Enter_Student_Reg_No';
  6      SELECT COUNT(*)
  7      INTO v_s_grade_count
  8      FROM ENROLL_24MCA0242
  9      WHERE Reg_no = v_reg_no AND Grade = 'S';
 10      DBMS_OUTPUT.PUT_LINE('Number of ''S'' Grades: ' || v_s_grade_count);
 11
 12  EXCEPTION
 13      WHEN NO_DATA_FOUND THEN
 14          DBMS_OUTPUT.PUT_LINE('No grades found for Reg_no: ' || v_reg_no);
 15      WHEN OTHERS THEN
 16          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 17  END;
 18  /
Enter value for enter_student_reg_no: R0003
old   5:      v_reg_no := '&Enter_Student_Reg_No';
new   5:      v_reg_no := 'R0003';
Number of 'S' Grades: 3

PL/SQL procedure successfully completed.
```

5. Write a PL/SQL program to print the regno and student names who are studying in the first semester.

CODE:

```
DECLARE
    CURSOR first_sem_students IS
        SELECT s.Reg_no, s.Sname
```

```
        FROM STUDENT_24MCA0242 s
        JOIN ENROLL_24MCA0242 e ON s.Reg_no = e.Reg_no
        JOIN CLASS_24MCA0242 c ON e.Cls_code = c.Cls_code
        JOIN SEMESTER_24MCA0242 sem ON c.Sem_code = sem.Sem_code
        WHERE sem.Sem_code = 'Fall2024-25'
          AND sem.Term = 'Fall'
          AND sem.Year = 2024;
    v_reg_no STUDENT_24MCA0242.Reg_no%TYPE;
    v_sname STUDENT_24MCA0242.Sname%TYPE;
BEGIN
    OPEN first_sem_students;
    DBMS_OUTPUT.PUT_LINE('Students in First Semester:');
    LOOP
        FETCH first_sem_students INTO v_reg_no, v_sname;
        EXIT WHEN first_sem_students%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Reg_no: ' || v_reg_no || ', Name: ' || v_sname);
    END LOOP;
    CLOSE first_sem_students;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

OUTPUT:

```
SQL> DECLARE
  2      CURSOR first_sem_students IS
  3          SELECT s.Reg_no, s.Sname
  4          FROM STUDENT_24MCA0242 s
  5          JOIN ENROLL_24MCA0242 e ON s.Reg_no = e.Reg_no
  6          JOIN CLASS_24MCA0242 c ON e.Cls_code = c.Cls_code
  7          JOIN SEMESTER_24MCA0242 sem ON c.Sem_code = sem.Sem_code
  8          WHERE sem.Sem_code = 'Fall2024-25'
  9            AND sem.Term = 'Fall'
 10            AND sem.Year = 2024;
 11
 12      v_reg_no STUDENT_24MCA0242.Reg_no%TYPE;
 13      v_sname STUDENT_24MCA0242.Sname%TYPE;
 14  BEGIN
 15
 16      OPEN first_sem_students;
 17      DBMS_OUTPUT.PUT_LINE('Students in First Semester:');
 18
 19      LOOP
 20          FETCH first_sem_students INTO v_reg_no, v_sname;
 21          EXIT WHEN first_sem_students%NOTFOUND;
 22          DBMS_OUTPUT.PUT_LINE('Reg_no: ' || v_reg_no || ', Name: ' || v_sname);
 23      END LOOP;
 24
 25      CLOSE first_sem_students;
 26  EXCEPTION
 27      WHEN OTHERS THEN
 28          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 29  END;
 30  /
Students in First Semester:
Reg_no: R0001, Name: Amit Verma
Reg_no: R0002, Name: Sunita Desai
Reg_no: R0002, Name: Sunita Desai
Reg_no: R0003, Name: Rahul Singh
Reg_no: R0004, Name: Pooja Reddy
Reg_no: R0005, Name: Anil Kumar

PL/SQL procedure successfully completed.
```

6. Write a PL/SQL program to find out what are all the courses that a professor has handled in the semester 1 and 2.

CODE:

```
DECLARE
   v_prof_id PROFESSOR_24MCA0242.Prof_id%TYPE := '&Enter_Prof_id';
BEGIN
   DBMS_OUTPUT.PUT_LINE('Courses handled by Professor ' || v_prof_id || ' in First Semester (Fall 2024-25):');
   FOR rec IN (
      SELECT c.Crs_name
      FROM CLASS_24MCA0242 cl
      JOIN COURSE_24MCA0242 c ON cl.Crs_code = c.Crs_code
      JOIN SEMESTER_24MCA0242 s ON cl.Sem_code = s.Sem_code
```

```
        WHERE cl.Prof_id = v_prof_id
        AND s.Term = 'Fall'
        AND s.Year = 2024
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('- ' || rec.Crs_name);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Courses handled by Professor ' || v_prof_id || ' in Second Semester (Winter 2024-25):');
    FOR rec IN (
        SELECT c.Crs_name
        FROM CLASS_24MCA0242 cl
        JOIN COURSE_24MCA0242 c ON cl.Crs_code = c.Crs_code
        JOIN SEMESTER_24MCA0242 s ON cl.Sem_code = s.Sem_code
        WHERE cl.Prof_id = v_prof_id
        AND s.Term = 'Winter'
        AND s.Year = 2024
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('- ' || rec.Crs_name);
    END LOOP;
END;
/
```

OUTPUT:

```
SQL> DECLARE
  2      v_prof_id PROFESSOR_24MCA0242.Prof_id%TYPE := '&Enter_Prof_id';
  3  BEGIN
  4      DBMS_OUTPUT.PUT_LINE('Courses handled by Professor ' || v_prof_id || ' in First Semester (Fall 2024-25):');
  5      FOR rec IN (
  6          SELECT c.Crs_name
  7          FROM CLASS_24MCA0242 cl
  8          JOIN COURSE_24MCA0242 c ON cl.Crs_code = c.Crs_code
  9          JOIN SEMESTER_24MCA0242 s ON cl.Sem_code = s.Sem_code
 10          WHERE cl.Prof_id = v_prof_id
 11          AND s.Term = 'Fall'
 12          AND s.Year = 2024
 13      ) LOOP
 14          DBMS_OUTPUT.PUT_LINE('- ' || rec.Crs_name);
 15      END LOOP;
 16
 17      DBMS_OUTPUT.PUT_LINE('Courses handled by Professor ' || v_prof_id || ' in Second Semester (Winter 2024-25):');
 18      FOR rec IN (
 19          SELECT c.Crs_name
 20          FROM CLASS_24MCA0242 cl
 21          JOIN COURSE_24MCA0242 c ON cl.Crs_code = c.Crs_code
 22          JOIN SEMESTER_24MCA0242 s ON cl.Sem_code = s.Sem_code
 23          WHERE cl.Prof_id = v_prof_id
 24          AND s.Term = 'Winter'
 25          AND s.Year = 2024
 26      ) LOOP
 27          DBMS_OUTPUT.PUT_LINE('- ' || rec.Crs_name);
 28      END LOOP;
 29  END;
 30  /
Enter value for enter_prof_id: P0005
old   2:      v_prof_id PROFESSOR_24MCA0242.Prof_id%TYPE := '&Enter_Prof_id';
new   2:      v_prof_id PROFESSOR_24MCA0242.Prof_id%TYPE := 'P0005';
Courses handled by Professor P0005 in First Semester (Fall 2024-25):
- Artificial Intelligence
Courses handled by Professor P0005 in Second Semester (Winter 2024-25):
- Database Systems

PL/SQL procedure successfully completed.
```

7. Implement and test a trigger to ensure that a student cannot enroll in a course after the semester has started.

CODE:

```
CREATE OR REPLACE TRIGGER check_enroll_date

BEFORE INSERT ON ENROLL_24MCA0242

FOR EACH ROW

DECLARE

   v_sdate SEMESTER_24MCA0242.Sdate%TYPE;

BEGIN

   SELECT Sem.Sdate

   INTO v_sdate

   FROM SEMESTER_24MCA0242 Sem

   JOIN CLASS_24MCA0242 C ON Sem.Sem_code = C.Sem_code
```

```
    WHERE C.Cls_code = :NEW.Cls_code;
    IF :NEW.Enroll_time > v_sdate THEN
        RAISE_APPLICATION_ERROR(-20001, 'Enrollment is not allowed after the semester start date.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Invalid class or semester.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20003, 'Error: ' || SQLERRM);
END;
/
```

OUTPUT:

```
SQL> CREATE OR REPLACE TRIGGER check_enroll_date
  2  BEFORE INSERT ON ENROLL_24MCA0242
  3  FOR EACH ROW
  4  DECLARE
  5      v_sdate SEMESTER_24MCA0242.Sdate%TYPE;
  6  BEGIN
  7      SELECT Sem.Sdate
  8      INTO v_sdate
  9      FROM SEMESTER_24MCA0242 Sem
 10      JOIN CLASS_24MCA0242 C ON Sem.Sem_code = C.Sem_code
 11      WHERE C.Cls_code = :NEW.Cls_code;
 12
 13      IF :NEW.Enroll_time > v_sdate THEN
 14          RAISE_APPLICATION_ERROR(-20001, 'Enrollment is not allowed after the semester start date.');
 15      END IF;
 16  EXCEPTION
 17      WHEN NO_DATA_FOUND THEN
 18          RAISE_APPLICATION_ERROR(-20002, 'Invalid class or semester.');
 19      WHEN OTHERS THEN
 20          RAISE_APPLICATION_ERROR(-20003, 'Error: ' || SQLERRM);
 21  END;
 22  /

Trigger created.
```

TESTING THE TRIGGER:

```
INSERT INTO ENROLL_24MCA0242 (Cls_code, Reg_no, Enroll_time, Grade)
VALUES ('CLS01', 'R0001', TO_DATE('2023-09-15', 'YYYY-MM-DD'), 'A');
```

INSERT INTO ENROLL_24MCA0242 (Cls_code, Reg_no, Enroll_time, Grade)

VALUES ('CLS03', 'R0002', TO_DATE('2023-05-15', 'YYYY-MM-DD'), 'A');

TESTING OUTPUT:

```
SQL> INSERT INTO ENROLL_24MCA0242 (Cls_code, Reg_no, Enroll_time, Grade)
  2  VALUES ('CLS01', 'R0001', TO_DATE('2023-09-15', 'YYYY-MM-DD'), 'A');
INSERT INTO ENROLL_24MCA0242 (Cls_code, Reg_no, Enroll_time, Grade)
            *
ERROR at line 1:
ORA-20003: Error: ORA-20001: Enrollment is not allowed after the semester start
date.
ORA-06512: at "SYSTEM.CHECK_ENROLL_DATE", line 17
ORA-04088: error during execution of trigger 'SYSTEM.CHECK_ENROLL_DATE'
```

```
SQL> INSERT INTO ENROLL_24MCA0242 (Cls_code, Reg_no, Enroll_time, Grade)
  2  VALUES ('CLS03', 'R0002', TO_DATE('2023-05-15', 'YYYY-MM-DD'), 'A');

1 row created.
```

8. Implement and test a trigger to ensure that number of departments in a school cannot exceed three.

CODE:

CREATE OR REPLACE TRIGGER check_department_limit

BEFORE INSERT ON DEPARTMENT_24MCA0242

FOR EACH ROW

DECLARE

v_dept_count NUMBER;

BEGIN

SELECT COUNT(*)

INTO v_dept_count

FROM DEPARTMENT_24MCA0242

WHERE SCode = :NEW.SCode;

IF v_dept_count >= 3 THEN

RAISE_APPLICATION_ERROR(-20003, 'A school cannot have more than 3 departments.');

END IF;

END;

/

OUTPUT:

```
SQL> CREATE OR REPLACE TRIGGER check_department_limit
  2  BEFORE INSERT ON DEPARTMENT_24MCA0242
  3  FOR EACH ROW
  4  DECLARE
  5      v_dept_count NUMBER;
  6  BEGIN
  7      SELECT COUNT(*)
  8      INTO v_dept_count
  9      FROM DEPARTMENT_24MCA0242
 10      WHERE SCode = :NEW.SCode;
 11
 12      IF v_dept_count >= 3 THEN
 13          RAISE_APPLICATION_ERROR(-20003, 'A school cannot have more than 3 departments.');
 14      END IF;
 15  END;
 16  /

Trigger created.
```

TESTING THE TRIGGER:

INSERT INTO DEPARTMENT_24MCA0242 (Dept_id, Dname, SCode, Prof_id)

VALUES ('D0010', 'Dept 4', 'S0006', 'P0004');

TESTING OUTPUT:

```
SQL> INSERT INTO DEPARTMENT_24MCA0242 (Dept_id, Dname, SCode, Prof_id)
  2  VALUES ('D0010', 'Dept 4', 'S0006', 'P0004');
INSERT INTO DEPARTMENT_24MCA0242 (Dept_id, Dname, SCode, Prof_id)
            *
ERROR at line 1:
ORA-20003: A school cannot have more than 3 departments.
ORA-06512: at "SYSTEM.CHECK_DEPARTMENT_LIMIT", line 10
ORA-04088: error during execution of trigger 'SYSTEM.CHECK_DEPARTMENT_LIMIT'
```

9. Write a trigger to subside referential integrity constraint. (Choose tables of your choice)

CODE:

CREATE OR REPLACE TRIGGER trg_professor_delete

BEFORE DELETE ON PROFESSOR_24MCA0242

```
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_count
   FROM SCHOOL_24MCA0242
   WHERE Prof_id = :OLD.Prof_id;
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Cannot delete professor: Prof_id is referenced in SCHOOL table.');
   END IF;
   SELECT COUNT(*) INTO v_count
   FROM DEPARTMENT_24MCA0242
   WHERE Prof_id = :OLD.Prof_id;
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20002, 'Cannot delete professor: Prof_id is referenced in DEPARTMENT table.');
   END IF;
   SELECT COUNT(*) INTO v_count
   FROM CLASS_24MCA0242
   WHERE Prof_id = :OLD.Prof_id;
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20003, 'Cannot delete professor: Prof_id is referenced in CLASS table.');
   END IF;
```

```
    SELECT COUNT(*) INTO v_count
    FROM STUDENT_24MCA0242
    WHERE Prof_id = :OLD.Prof_id;
    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20004, 'Cannot delete professor: Prof_id is referenced in STUDENT table.');
    END IF;
END;
/
```

OUTPUT:

```
SQL> CREATE OR REPLACE TRIGGER trg_professor_delete
  2  BEFORE DELETE ON PROFESSOR_24MCA0242
  3  FOR EACH ROW
  4  DECLARE
  5      v_count NUMBER;
  6  BEGIN
  7      SELECT COUNT(*) INTO v_count
  8      FROM SCHOOL_24MCA0242
  9      WHERE Prof_id = :OLD.Prof_id;
 10      IF v_count > 0 THEN
 11          RAISE_APPLICATION_ERROR(-20001, 'Cannot delete professor: Prof_id is referenced in SCHOOL table.');
 12      END IF;
 13
 14      SELECT COUNT(*) INTO v_count
 15      FROM DEPARTMENT_24MCA0242
 16      WHERE Prof_id = :OLD.Prof_id;
 17      IF v_count > 0 THEN
 18          RAISE_APPLICATION_ERROR(-20002, 'Cannot delete professor: Prof_id is referenced in DEPARTMENT table.');
 19      END IF;
 20
 21      SELECT COUNT(*) INTO v_count
 22      FROM CLASS_24MCA0242
 23      WHERE Prof_id = :OLD.Prof_id;
 24      IF v_count > 0 THEN
 25          RAISE_APPLICATION_ERROR(-20003, 'Cannot delete professor: Prof_id is referenced in CLASS table.');
 26      END IF;
 27
 28      SELECT COUNT(*) INTO v_count
 29      FROM STUDENT_24MCA0242
 30      WHERE Prof_id = :OLD.Prof_id;
 31      IF v_count > 0 THEN
 32          RAISE_APPLICATION_ERROR(-20004, 'Cannot delete professor: Prof_id is referenced in STUDENT table.');
 33      END IF;
 34  END;
 35  /

Trigger created.
```

TESTING THE TRIGGER:

```
DELETE FROM PROFESSOR_24MCA0242 WHERE Prof_id = 'P0003';
```

TESTING OUTPUT:

```
SQL> DELETE FROM PROFESSOR_24MCA0242 WHERE Prof_id = 'P0003';
DELETE FROM PROFESSOR_24MCA0242 WHERE Prof_id = 'P0003'
            *
ERROR at line 1:
ORA-20001: Cannot delete professor: Prof_id is referenced in SCHOOL table.
ORA-06512: at "SYSTEM.TRG_PROFESSOR_DELETE", line 8
ORA-04088: error during execution of trigger 'SYSTEM.TRG_PROFESSOR_DELETE'
```

10. Write a PL/SQL program to interchange the department of Professor P0006 and P0007.

CODE:

```
DECLARE
   v_dept_id_P0006 PROFESSOR_24MCA0242.Dept_id%TYPE;
   v_dept_id_P0007 PROFESSOR_24MCA0242.Dept_id%TYPE;
BEGIN
   SELECT Dept_id INTO v_dept_id_P0006
   FROM PROFESSOR_24MCA0242
   WHERE Prof_id = 'P0006';
   SELECT Dept_id INTO v_dept_id_P0007
   FROM PROFESSOR_24MCA0242
   WHERE Prof_id = 'P0007';
   UPDATE PROFESSOR_24MCA0242
   SET Dept_id = CASE
                  WHEN Prof_id = 'P0006' THEN v_dept_id_P0007
                  WHEN Prof_id = 'P0007' THEN v_dept_id_P0006
                END
   WHERE Prof_id IN ('P0006', 'P0007');
   COMMIT;
```

```
    DBMS_OUTPUT.PUT_LINE('Departments interchanged successfully between Professors P0006 and P0007.');
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('One or both professors not found.');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
  END;
  /
```

OUTPUT:

```
SQL> SELECT Prof_id, Dept_id FROM PROFESSOR_24MCA0242
  2  WHERE Prof_id IN ('P0006', 'P0007');

PROF_ DEPT_
----- -----
P0006 D0006
P0007 D0007
```

```
SQL> DECLARE
  2      v_dept_id_P0006 PROFESSOR_24MCA0242.Dept_id%TYPE;
  3      v_dept_id_P0007 PROFESSOR_24MCA0242.Dept_id%TYPE;
  4  BEGIN
  5      SELECT Dept_id INTO v_dept_id_P0006
  6      FROM PROFESSOR_24MCA0242
  7      WHERE Prof_id = 'P0006';
  8
  9      SELECT Dept_id INTO v_dept_id_P0007
 10      FROM PROFESSOR_24MCA0242
 11      WHERE Prof_id = 'P0007';
 12
 13      UPDATE PROFESSOR_24MCA0242
 14      SET Dept_id = CASE
 15                      WHEN Prof_id = 'P0006' THEN v_dept_id_P0007
 16                      WHEN Prof_id = 'P0007' THEN v_dept_id_P0006
 17                    END
 18      WHERE Prof_id IN ('P0006', 'P0007');
 19
 20      COMMIT;
 21
 22      DBMS_OUTPUT.PUT_LINE('Departments interchanged successfully between Professors P0006 and P0007.');
 23
 24  EXCEPTION
 25      WHEN NO_DATA_FOUND THEN
 26          DBMS_OUTPUT.PUT_LINE('One or both professors not found.');
 27      WHEN OTHERS THEN
 28          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 29  END;
 30  /
Departments interchanged successfully between Professors P0006 and P0007.

PL/SQL procedure successfully completed.
```

```
SQL> SELECT Prof_id, Dept_id FROM PROFESSOR_24MCA0242
  2  WHERE Prof_id IN ('P0006', 'P0007');

PROF_ DEPT_
----- -----
P0006 D0007
P0007 D0006
```

11. Create a function that takes department ID and returns the name of the Head of the department.

CODE:

```
CREATE OR REPLACE FUNCTION get_department_head(p_dept_id IN DEPARTMENT_24MCA0242.Dept_id%TYPE)
RETURN PROFESSOR_24MCA0242.Prof_name%TYPE IS
   v_head_name PROFESSOR_24MCA0242.Prof_name%TYPE;
BEGIN
   SELECT p.Prof_name
   INTO v_head_name
   FROM PROFESSOR_24MCA0242 p
   JOIN DEPARTMENT_24MCA0242 d ON p.Prof_id = d.Prof_id
   WHERE d.Dept_id = p_dept_id;
   RETURN v_head_name;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL;
   WHEN OTHERS THEN
      RETURN 'Error: ' || SQLERRM;
END;
/
```

OUTPUT:

```
SQL> CREATE OR REPLACE FUNCTION get_department_head(p_dept_id IN DEPARTMENT_24MCA0242.Dept_id%TYPE)
  2  RETURN PROFESSOR_24MCA0242.Prof_name%TYPE IS
  3      v_head_name PROFESSOR_24MCA0242.Prof_name%TYPE;
  4
  5  BEGIN
  6      SELECT p.Prof_name
  7      INTO v_head_name
  8      FROM PROFESSOR_24MCA0242 p
  9      JOIN DEPARTMENT_24MCA0242 d ON p.Prof_id = d.Prof_id
 10      WHERE d.Dept_id = p_dept_id;
 11      RETURN v_head_name;
 12
 13  EXCEPTION
 14      WHEN NO_DATA_FOUND THEN
 15          RETURN NULL;
 16      WHEN OTHERS THEN
 17          RETURN 'Error: ' || SQLERRM;
 18  END;
 19  /

Function created.
```

TESTING THE FUNCTION:

```
DECLARE
    v_head_name PROFESSOR_24MCA0242.Prof_name%TYPE;
BEGIN
    v_head_name := get_department_head('D0003');
    IF v_head_name IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Head of Department D0003: ' || v_head_name);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No head found for Department D0003.');
    END IF;
END;
/
```

TESTING OUTPUT:

```
SQL> DECLARE
  2      v_head_name PROFESSOR_24MCA0242.Prof_name%TYPE;
  3  BEGIN
  4      v_head_name := get_department_head('D0003');
  5      IF v_head_name IS NOT NULL THEN
  6          DBMS_OUTPUT.PUT_LINE('Head of Department D0003: ' || v_head_name);
  7      ELSE
  8          DBMS_OUTPUT.PUT_LINE('No head found for Department D0003.');
  9      END IF;
 10  END;
 11  /
Head of Department D0003: Dr. Arjun Mehta

PL/SQL procedure successfully completed.
```

12. Create a function that displays the age of the student from his DOB.

CODE:

```
CREATE OR REPLACE FUNCTION get_student_age(p_reg_no IN STUDENT_24MCA0242.Reg_no%TYPE)
RETURN NUMBER IS
   v_age NUMBER;
BEGIN
   SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, DoB) / 12)
   INTO v_age
   FROM STUDENT_24MCA0242
   WHERE Reg_no = p_reg_no;
   RETURN v_age;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL;
   WHEN OTHERS THEN
      RETURN -1;
END;
/
```

OUTPUT:

```
SQL> CREATE OR REPLACE FUNCTION get_student_age(p_reg_no IN STUDENT_24MCA0242.Reg_no%TYPE)
  2  RETURN NUMBER IS
  3      v_age NUMBER;
  4
  5  BEGIN
  6      SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, DoB) / 12)
  7      INTO v_age
  8      FROM STUDENT_24MCA0242
  9      WHERE Reg_no = p_reg_no;
 10      RETURN v_age;
 11
 12  EXCEPTION
 13      WHEN NO_DATA_FOUND THEN
 14          RETURN NULL;
 15      WHEN OTHERS THEN
 16          RETURN -1;
 17  END;
 18  /

Function created.
```

TESTING THE FUNCTION:

```
DECLARE
    v_age NUMBER;
BEGIN
    v_age := get_student_age('R0001');
    IF v_age IS NOT NULL AND v_age >= 0 THEN
        DBMS_OUTPUT.PUT_LINE('Age of Student R0001: ' || v_age || ' years');
    ELSIF v_age IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('No student found with Reg_no R0001');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Error calculating age');
    END IF;
END;
/
```

TESTING OUTPUT:

```
SQL> DECLARE
  2      v_age NUMBER;
  3  BEGIN
  4      v_age := get_student_age('R0001');
  5
  6      IF v_age IS NOT NULL AND v_age >= 0 THEN
  7          DBMS_OUTPUT.PUT_LINE('Age of Student R0001: ' || v_age || ' years');
  8      ELSIF v_age IS NULL THEN
  9          DBMS_OUTPUT.PUT_LINE('No student found with Reg_no R0001');
 10      ELSE
 11          DBMS_OUTPUT.PUT_LINE('Error calculating age');
 12      END IF;
 13  END;
 14  /
Age of Student R0001: 26 years

PL/SQL procedure successfully completed.
```