

**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:**



1. 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

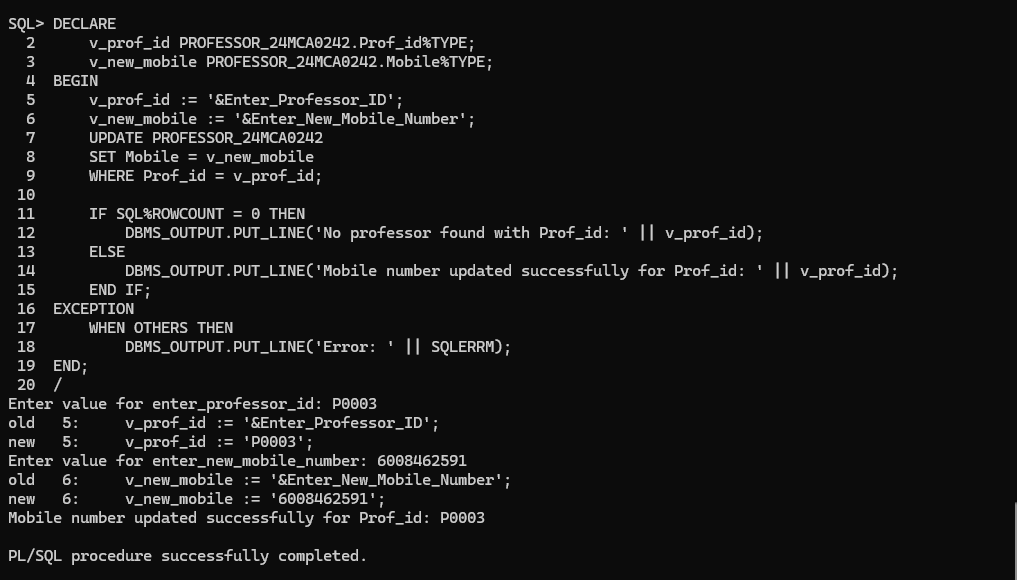
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

**OUTPUT:**



1. 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

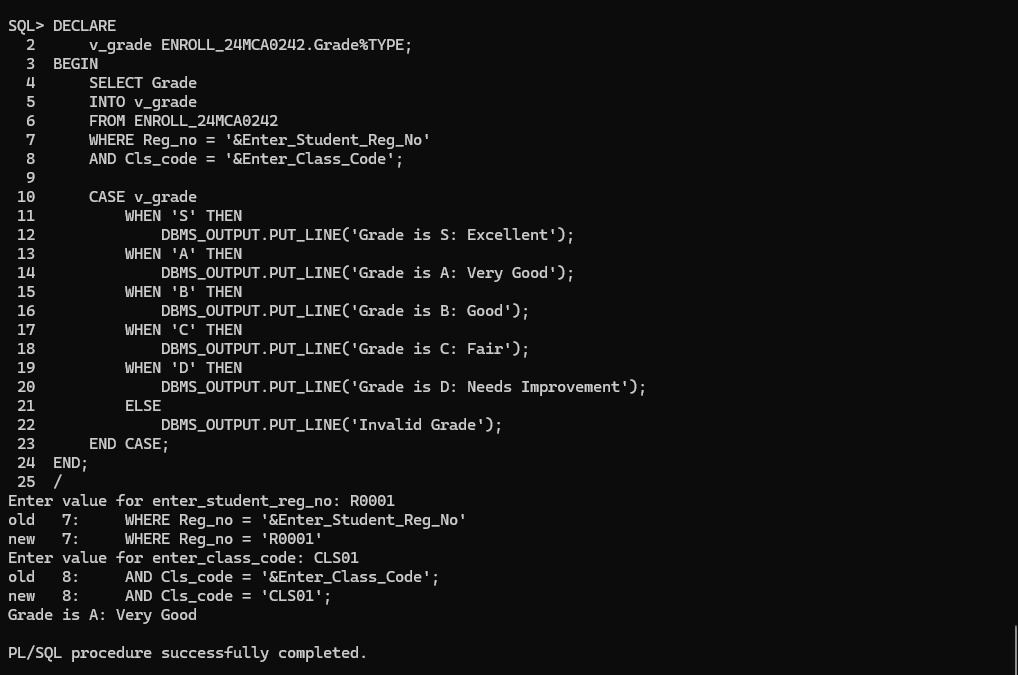
DBMS\_OUTPUT.PUT\_LINE('Invalid Grade');

END CASE;

END;

/

**OUTPUT:**



1. 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);

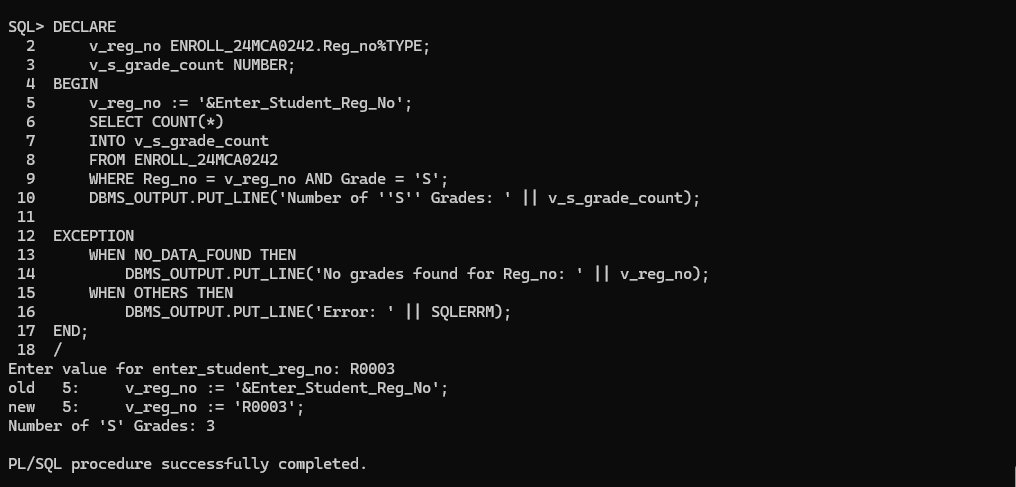
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

**OUTPUT:**



1. 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:**



1. 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

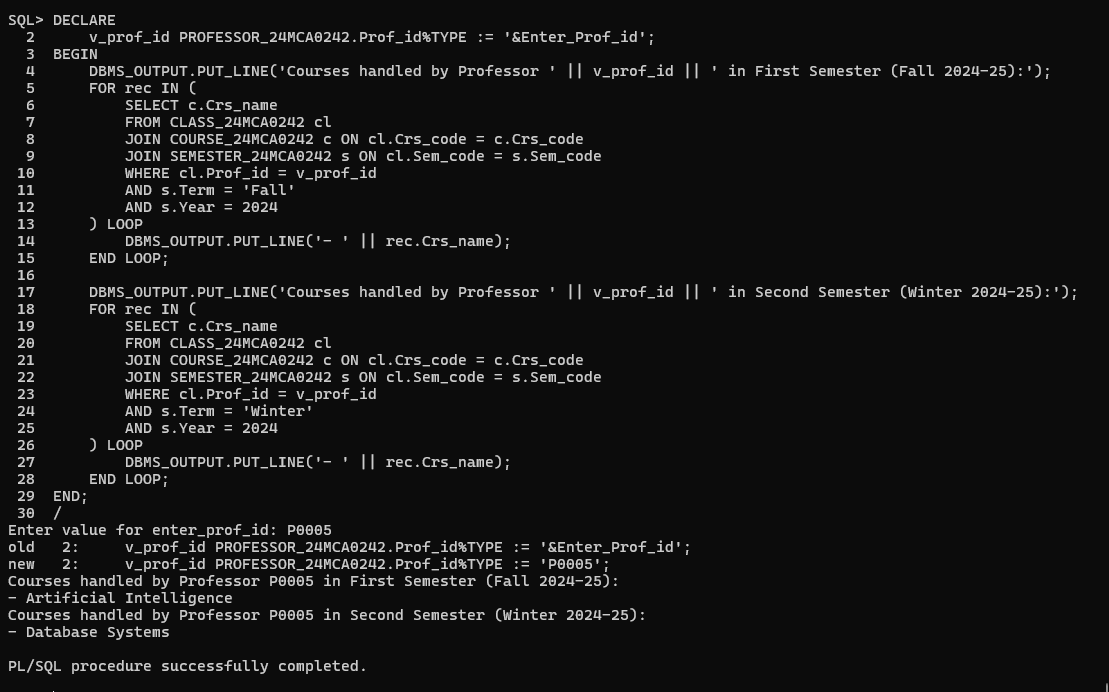
DBMS\_OUTPUT.PUT\_LINE('- ' || rec.Crs\_name);

END LOOP;

END;

/

**OUTPUT:**



1. 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.');

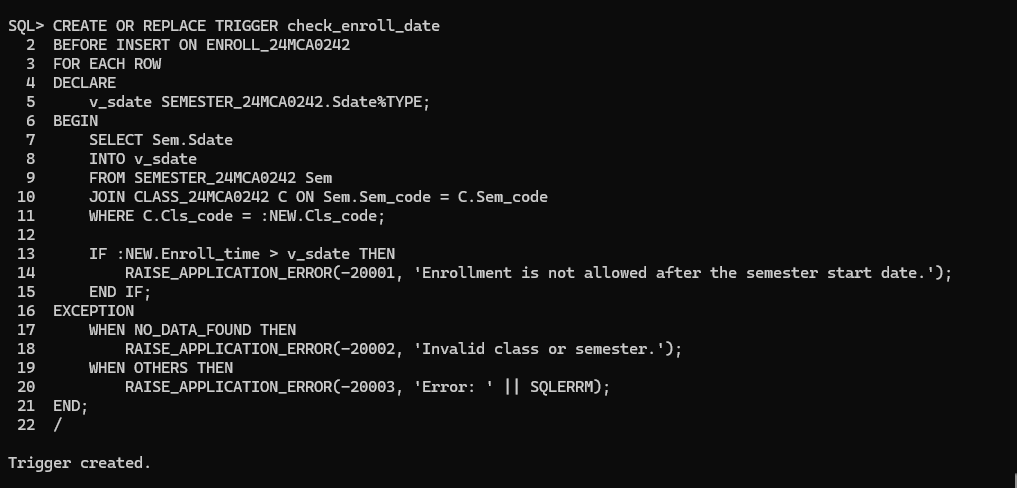
WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Error: ' || SQLERRM);

END;

/

**OUTPUT:**



**TESTING THE TRIGGER:**

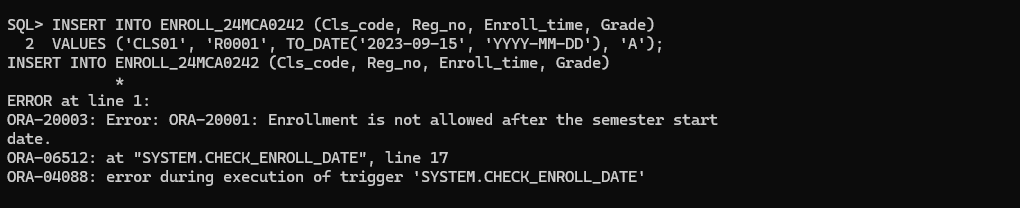
INSERT INTO ENROLL\_24MCA0242 (Cls\_code, Reg\_no, Enroll\_time, Grade)

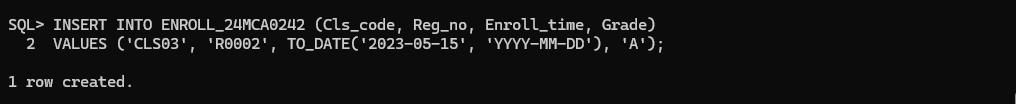
VALUES ('CLS01', 'R0001', TO\_DATE('2023-09-15', 'YYYY-MM-DD'), 'A');

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:**





1. 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

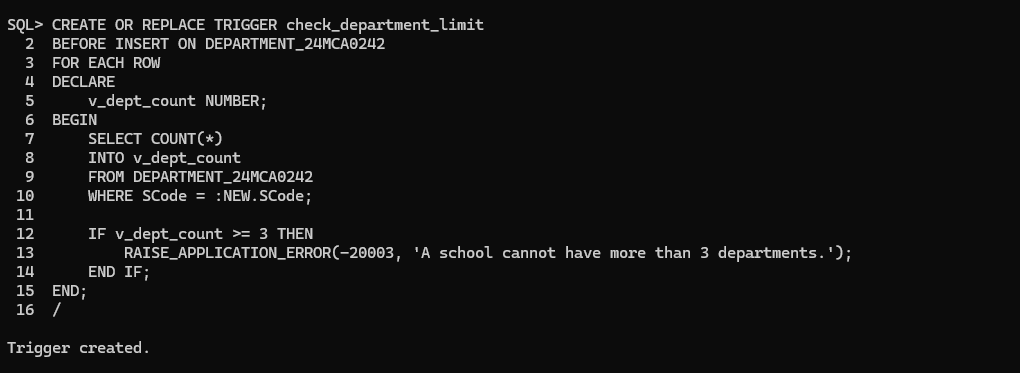
        RAISE\_APPLICATION\_ERROR(-20003, 'A school cannot have more than 3 departments.');

    END IF;

END;

/

**OUTPUT:**

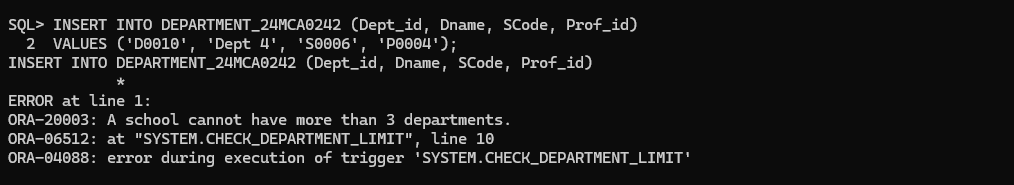


**TESTING THE TRIGGER:**

INSERT INTO DEPARTMENT\_24MCA0242 (Dept\_id, Dname, SCode, Prof\_id)

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

**TESTING OUTPUT:**



1. 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

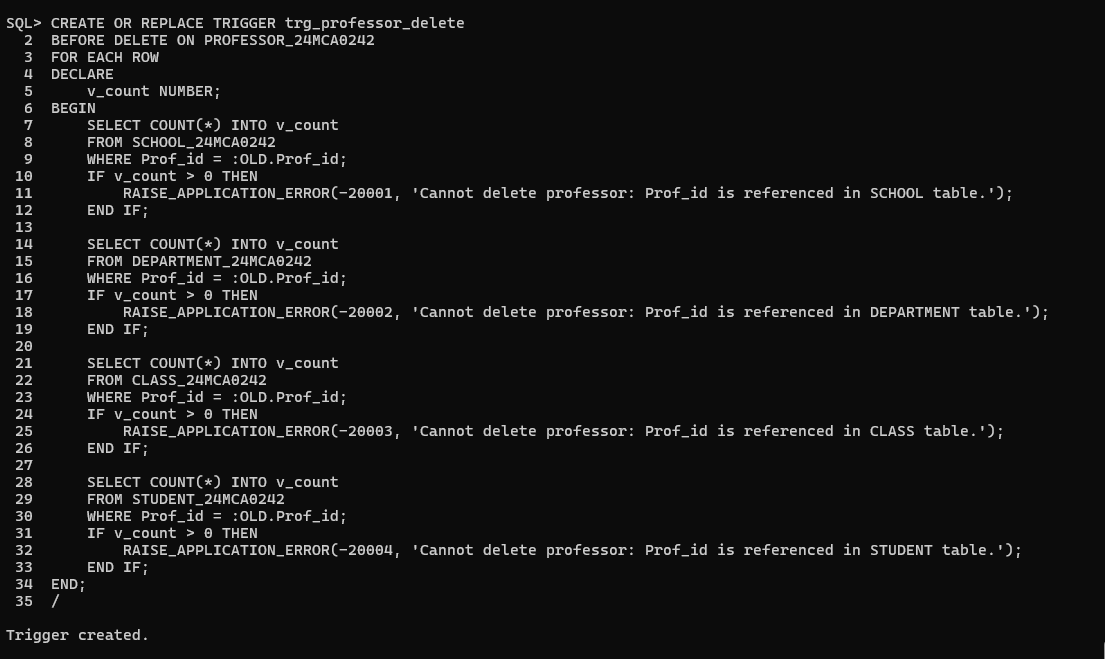
RAISE\_APPLICATION\_ERROR(-20004, 'Cannot delete professor: Prof\_id is referenced in STUDENT table.');

END IF;

END;

/

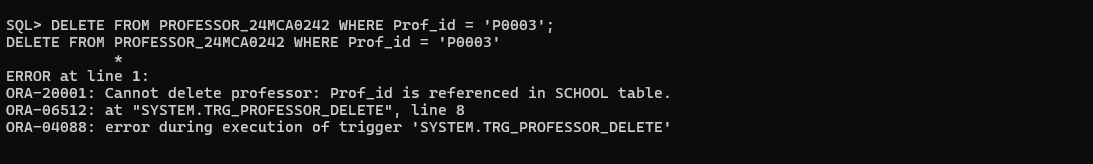
**OUTPUT:**



**TESTING THE TRIGGER:**

DELETE FROM PROFESSOR\_24MCA0242 WHERE Prof\_id = 'P0003';

**TESTING OUTPUT:**



1. 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.');

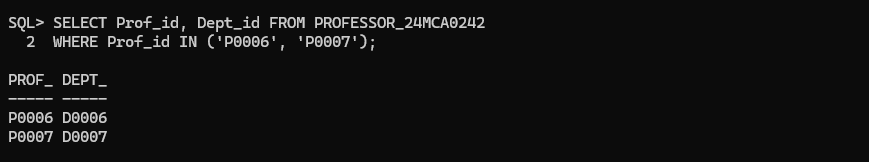
WHEN OTHERS THEN

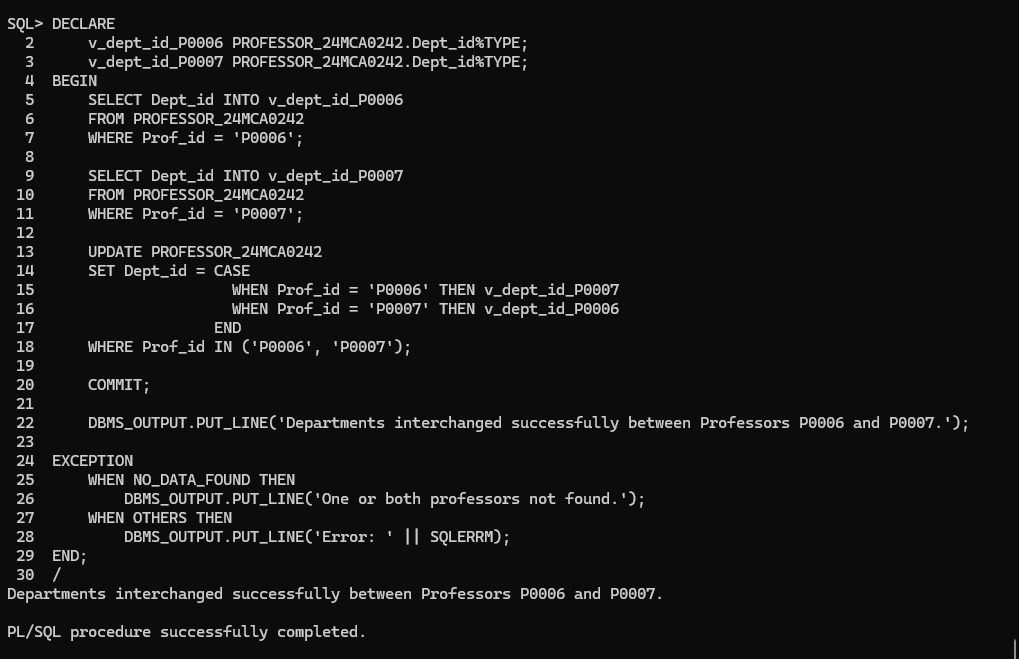
DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

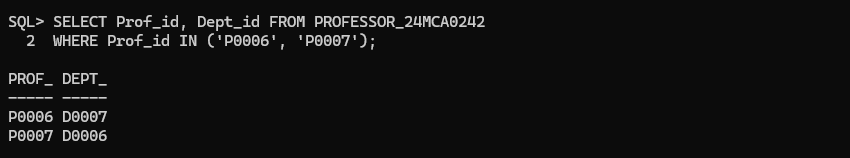
END;

/

**OUTPUT:**







1. 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;

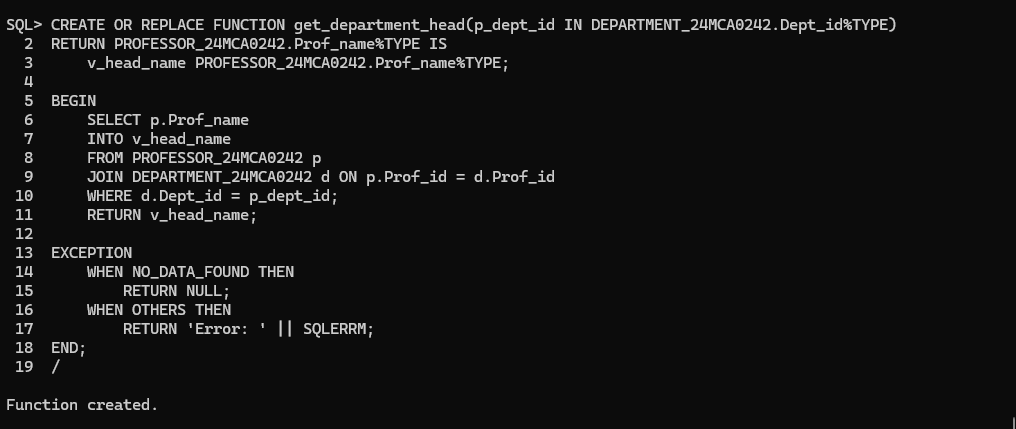
WHEN OTHERS THEN

RETURN 'Error: ' || SQLERRM;

END;

/

**OUTPUT:**



**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

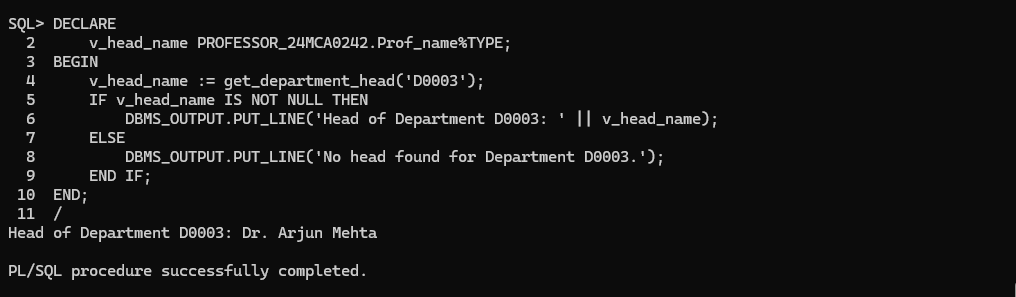
DBMS\_OUTPUT.PUT\_LINE('No head found for Department D0003.');

END IF;

END;

/

**TESTING OUTPUT:**



1. 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;

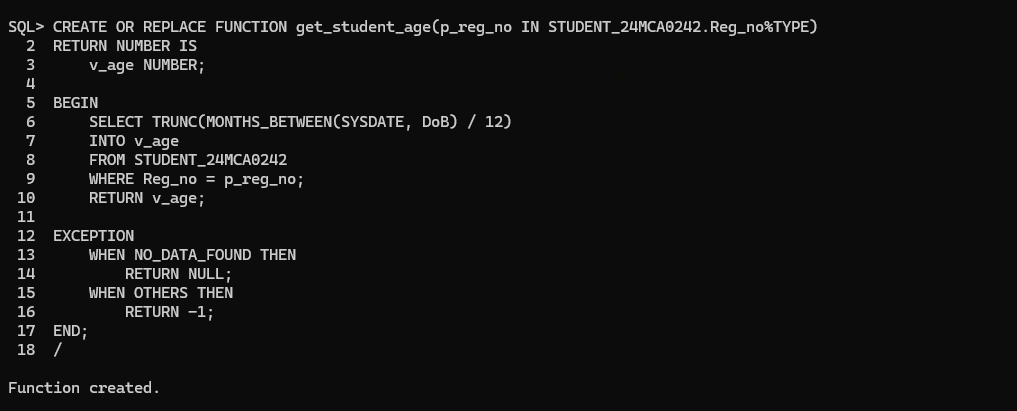
WHEN OTHERS THEN

RETURN -1;

END;

/

**OUTPUT:**



**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:**

