Cycle Sheet

--CYCLE SHEET 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

DECLARE

reg STUDENT.Reg\_no%type := '&reg\_no'; name STUDENT.Sname%type;

dob STUDENT.DoB%type; add STUDENT.Address%type;

mobile STUDENT.Mobile%type; email STUDENT.Email%type;

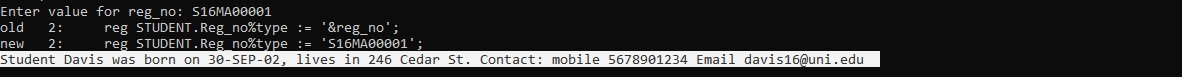
BEGIN

SELECT Sname, DoB, Address, Mobile, Email INTO name, dob, add, mobile, email FROM STUDENT WHERE Reg\_no = reg;

dbms\_output.put\_line('Student '||name||' was born on '||dob||', lives in '||add||'. Contact: mobile '||mobile||' Email '||email);

END;

/



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

DECLARE

id PROFESSOR.Prof\_id%type := '&ID';

mob PROFESSOR.Mobile%type := '&Mobile'; BEGIN

UPDATE PROFESSOR SET Mobile = mob WHERE Prof\_id = id; END;

/



--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.

BEGIN

FOR rec IN (

SELECT s.Sname, c.Crs\_name, e.Grade FROM STUDENT s

JOIN ENROLL e ON s.Reg\_no = e.Reg\_no

JOIN CLASS cls ON e.Cls\_code = cls.Cls\_code

JOIN COURSE c ON cls.Crs\_code = c.Crs\_code WHERE s.Reg\_no = 'S17MTech002'

) LOOP

IF rec.Grade = 'S' THEN

dbms\_output.put\_line(rec.Sname||' '||rec.Crs\_name||' Excellent'

);

ELSIF rec.Grade = 'A' THEN

dbms\_output.put\_line(rec.Sname||' '||rec.Crs\_name||' Excellent'

);

ELSIF rec.Grade = 'B' THEN

dbms\_output.put\_line(rec.Sname||' '||rec.Crs\_name||' Good');

ELSIF rec.Grade = 'C' THEN

dbms\_output.put\_line(rec.Sname||' '||rec.Crs\_name||' Fair'); ELSE

dbms\_output.put\_line(rec.Sname||' '||rec.Crs\_name||' Fair'); END IF;

END LOOP;

END;

/



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

DECLARE

reg STUDENT.Reg\_no%TYPE := '&Reg\_no'; -- Student register number name STUDENT.Sname%TYPE;

c NUMBER(2); BEGIN

-- Retrieve the student's name SELECT Sname INTO name

FROM STUDENT

WHERE Reg\_no = reg;

-- Count the number of 'S' grades SELECT COUNT(\*) INTO c

FROM ENROLL e

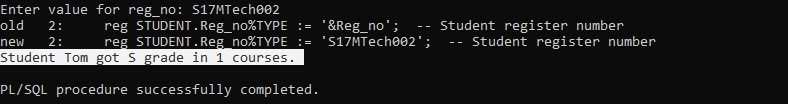
WHERE e.Reg\_no = reg AND e.Grade = 'S';

-- Output the result

dbms\_output.put\_line('Student ' || name || ' got S grade in ' || c || ' courses.');

END;

/



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

BEGIN

dbms\_output.put\_line('Student studying in first semester (Fall 2016) are: '); FOR rec IN (

SELECT s.Reg\_no, s.Sname FROM STUDENT s

JOIN ENROLL e ON s.Reg\_no = e.Reg\_no

JOIN CLASS cls ON e.Cls\_code = cls.Cls\_code

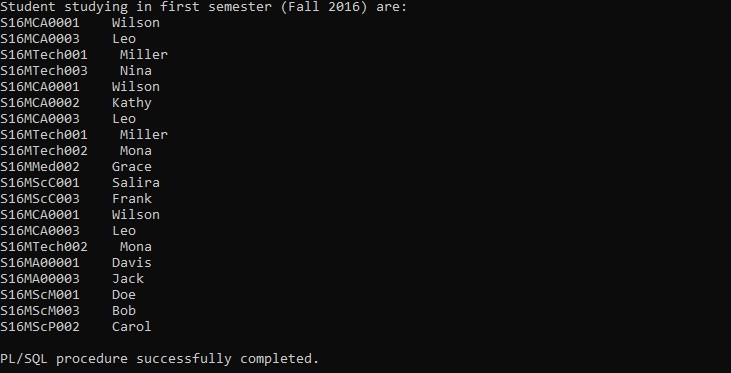
JOIN SEMESTER sem ON cls.Sem\_code = sem.Sem\_code WHERE sem.Year = '2016'AND sem.Term LIKE 'Fall'

) LOOP

dbms\_output.put\_line(rec.Reg\_no||' '||rec.Sname); END LOOP;

END;

/



--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.

DECLARE

PID PROFESSOR.Prof\_id%TYPE; BEGIN

PID := '&pid';

dbms\_output.put\_line('Courses handled by '|| PID || ' in first and second semester (Fall 2016, Winter 2016) are: ');

FOR rec IN (

SELECT UNIQUE c.Crs\_code, c.Crs\_name FROM COURSE c

JOIN CLASS cls ON cls.Crs\_code = c.Crs\_code

JOIN (SELECT p1.Prof\_id FROM PROFESSOR p1 WHERE p1.Prof\_id = PID) p ON

p.Prof\_id = cls.Prof\_id

JOIN SEMESTER sem ON cls.Sem\_code = sem.Sem\_code

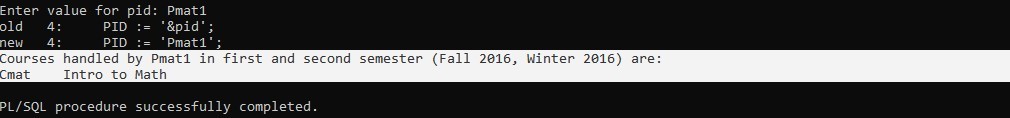
WHERE sem.Year = '2016'AND (sem.Term LIKE 'Fall' OR sem.Term LIKE 'Winter')

) LOOP

dbms\_output.put\_line(rec.Crs\_code||' '||rec.Crs\_name); END LOOP;

END;

/



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

CREATE OR REPLACE TRIGGER check\_enrollment\_date BEFORE INSERT ON ENROLL

FOR EACH ROW DECLARE

sem\_start\_date DATE; BEGIN

-- Get the semester start date based on the class code SELECT s.Sdate

INTO sem\_start\_date FROM SEMESTER s

JOIN CLASS c ON s.Sem\_code = c.Sem\_code WHERE c.Cls\_code = :NEW.Cls\_code;

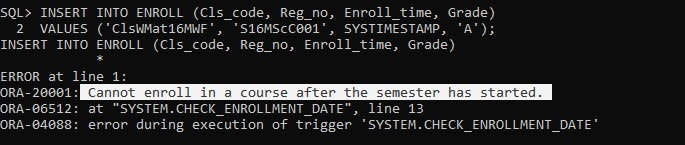
-- Check if the current date is after the semester start date IF SYSDATE > sem\_start\_date THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Cannot enroll in a course after the semester has started.');

END IF; END;

/



INSERT INTO ENROLL (Cls\_code, Reg\_no, Enroll\_time, Grade) VALUES ('ClsWMat16MWF', 'S16MScC001', SYSTIMESTAMP, 'A');

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

CREATE OR REPLACE TRIGGER check\_department\_count BEFORE INSERT OR UPDATE ON DEPARTMENT

FOR EACH ROW DECLARE

dept\_count NUMBER; BEGIN

-- Count the number of departments associated with the school code in the current record

SELECT COUNT(\*)

INTO dept\_count FROM DEPARTMENT

WHERE SCode = :NEW.SCode;

-- If the number of departments is already 3, prevent the insertion IF dept\_count >= 3 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Cannot add more than 3 departments to the school.');

END IF; END;

/



--10. Write a PL/SQL program to interchange the department of Professor Pmat1 and Pphy1.

DECLARE

-- Declare variables to hold the department IDs of Pmat1 and Pphy1 v\_dept\_pmat1 PROFESSOR.Dept\_id%TYPE;

v\_dept\_pphy1 PROFESSOR.Dept\_id%TYPE; BEGIN

-- Retrieve the current department ID of Professor Pmat1 SELECT Dept\_id INTO v\_dept\_pmat1

FROM PROFESSOR

WHERE Prof\_id = 'Pmat1';

-- Retrieve the current department ID of Professor Pphy1 SELECT Dept\_id INTO v\_dept\_pphy1

FROM PROFESSOR

WHERE Prof\_id = 'Pphy1';

-- Swap the department of Professor Pmat1 with Professor Pphy1 UPDATE PROFESSOR

SET Dept\_id = v\_dept\_pphy1 WHERE Prof\_id = 'Pmat1';

-- Swap the department of Professor Pphy1 with Professor Pmat1 UPDATE PROFESSOR

SET Dept\_id = v\_dept\_pmat1 WHERE Prof\_id = 'Pphy1';

-- Commit the changes COMMIT;

-- Output success message

dbms\_output.put\_line('Departments of Professors Pmat1 and Pphy1 have been interchanged.');

EXCEPTION

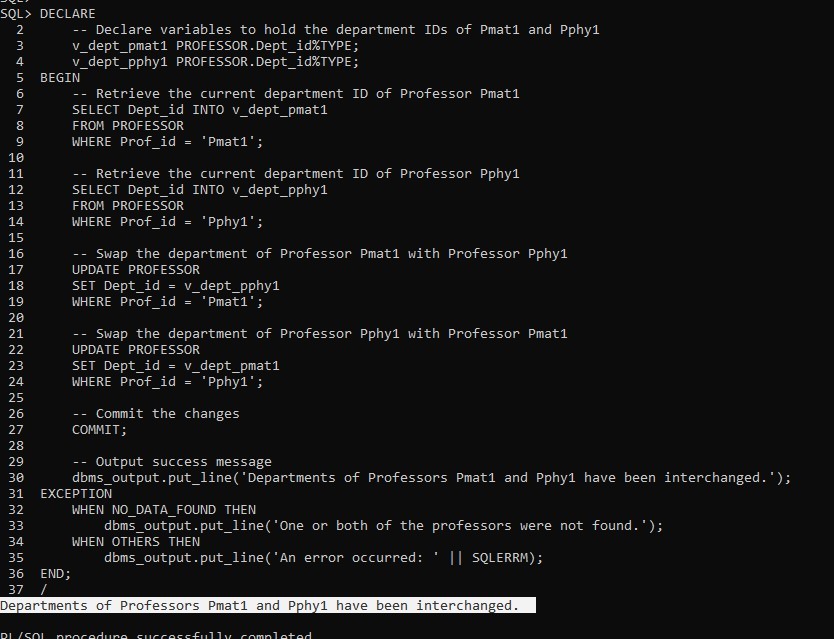
WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('One or both of the professors were not found.'); WHEN OTHERS THEN

dbms\_output.put\_line('An error occurred: ' || SQLERRM);

END;

/



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

CREATE OR REPLACE FUNCTION get\_dept\_head(dept\_id\_in DEPARTMENT.Dept\_id%TYPE) RETURN VARCHAR2

IS

v\_head\_name PROFESSOR.Prof\_name%TYPE; BEGIN

-- Retrieve the name of the head of the department SELECT p.Prof\_name

INTO v\_head\_name FROM PROFESSOR p

JOIN DEPARTMENT d ON p.Prof\_id = d.Prof\_id WHERE d.Dept\_id = dept\_id\_in;

-- Return the name of the head of the department RETURN v\_head\_name;

EXCEPTION

-- Handle case when no head is found WHEN NO\_DATA\_FOUND THEN

RETURN 'No head found for this department';

-- Handle other unexpected errors WHEN OTHERS THEN

RETURN 'An error occurred';

END;

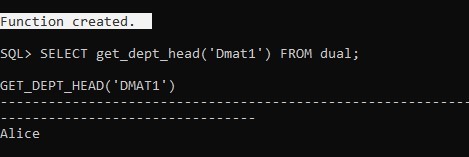
/

SELECT get\_dept\_head('Dmat1') FROM dual;

-- Output:

SELECT get\_dept\_head('Dphy2') FROM dual;

-- Output:



--12. Create a function that displays the age of the student from his DOB. CREATE OR REPLACE FUNCTION calculate\_age(dob\_in STUDENT.DoB%TYPE)

RETURN NUMBER IS

v\_age NUMBER; BEGIN

-- Calculate age using the current date and DOB

v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, dob\_in) / 12);

RETURN v\_age; EXCEPTION

-- Handle any unexpected errors WHEN OTHERS THEN

RETURN NULL; -- Return NULL if there's an error

END;

/

SELECT calculate\_age(TO\_DATE('2000-05-15', 'YYYY-MM-DD')) AS age FROM dual;

