**DBS501 - Fall 2020**

**DBS501 - Group No: 3**

**Assignment #1**

**Group Members:**

1. Antonio Bennett
2. Elisa Ng Li
3. Dennis Lam
4. Mohammed Suleiman Mohamed Al-Falahy

October 20th, 2020

**DBS501, FALL 20, ASSIGNMENT ONE, due on Monday, Nov 02nd by 7pm**

***e-mail the ANSWER as an ATTACHMENT, it must contain both Codes and Outputs***

**Your code must encompass ALL Test cases given here.**

1) Write the PL/SQL block that will add a new row to the Departments table like shown:

**Department\_id**  Highest value of all Id’s increased by 50

**Department\_name**  Testing

**Manager\_id**  One with most people under supervision (do NOT assume it is a President, prove it with the code)

**Location\_id**  You will be asked to input the City name where you want to place your new department. And that value will be converted to its related Location\_id.

You will then display that new row and later undo the insert.

Your input value should be a VALID city name without any department. You need also to code for following cases: There is already ONE department in that city, there is MORE THAN ONE department there and the city is NOT listed in the Locations table.

**You do NOT need to use cursor for this question,** Here are the outputs:

Top of Form

|  |  |  |
| --- | --- | --- |
| Please provide the valid city without department: |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **DEPARTMENT\_ID** | **DEPARTMENT\_NAME** | **MANAGER\_ID** | **LOCATION\_ID** |
| 320 | Testing | 100 | 1100 |

**Rollback complete.**

Top of Form

|  |  |  |
| --- | --- | --- |
| Please provide the valid city without department: |  |  |

**This city already contains department: Toronto  
PL/SQL procedure successfully completed.**

**no rows selected   
Rollback complete.**

Top of Form

|  |  |  |
| --- | --- | --- |
| Please provide the valid city without department: |  |  |

**This city has MORE THAN ONE department: Seattle  
PL/SQL procedure successfully completed.  
no rows selected   
Rollback complete.**

Top of Form

|  |  |  |
| --- | --- | --- |
| Please provide the valid city without department: |  |  |

**This city is NOT listed: Belgrade  
PL/SQL procedure successfully completed.  
no rows selected   
Rollback complete.**

**Code:**

SET PAGESIZE 1000;

SET SERVEROUTPUT ON;

SET VERIFY OFF;

ACCEPT i\_city PROMPT 'Please provide the valid city without department: ';

DECLARE

v\_city VARCHAR(100) := '&i\_city';

v\_depart\_id NUMBER;

v\_depart\_name VARCHAR(100) := 'Testing';

v\_manager\_id NUMBER;

v\_location\_id NUMBER;

v\_depart\_count NUMBER;

v\_message VARCHAR(100);

BEGIN

--DBMS\_OUTPUT.PUT\_LINE('Debug: v\_city, ' || v\_city);

SELECT MAX(department\_id) + 50 INTO v\_depart\_id

FROM departments;

--DBMS\_OUTPUT.PUT\_LINE('Debug: v\_depart\_id, ' || v\_depart\_id);

--DBMS\_OUTPUT.PUT\_LINE('Debug: v\_depart\_name, ' || v\_depart\_name);

SELECT MANAGER\_ID

INTO v\_manager\_id

FROM EMPLOYEES

GROUP BY MANAGER\_ID

HAVING COUNT(MANAGER\_ID) = (

SELECT MAX(MAXCOUNT)

FROM (

SELECT MANAGER\_ID, COUNT(MANAGER\_ID) MAXCOUNT

FROM EMPLOYEES

GROUP BY MANAGER\_ID));

--DBMS\_OUTPUT.PUT\_LINE('Debug: v\_manager\_id, ' || v\_manager\_id);

SELECT LOCATION\_ID

INTO v\_location\_id

FROM LOCATIONS

WHERE CITY = v\_city;

--DBMS\_OUTPUT.PUT\_LINE('Debug: v\_location\_id, ' || v\_location\_id);

SELECT COUNT(LOCATION\_ID)

INTO v\_depart\_count

FROM DEPARTMENTS

WHERE LOCATION\_ID = v\_location\_id;

--DBMS\_OUTPUT.PUT\_LINE('Debug: v\_depart\_count, ' || v\_depart\_count);

IF v\_depart\_count < 1 THEN

INSERT INTO DEPARTMENTS

VALUES (v\_depart\_id, v\_depart\_name, v\_manager\_id, v\_location\_id);

ELSIF v\_depart\_count = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('This city already contains department: ' || v\_city);

ELSE

DBMS\_OUTPUT.PUT\_LINE('This city has MORE THAN ONE department: ' || v\_city);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('This city is NOT listed: ' || v\_city);

END;

/

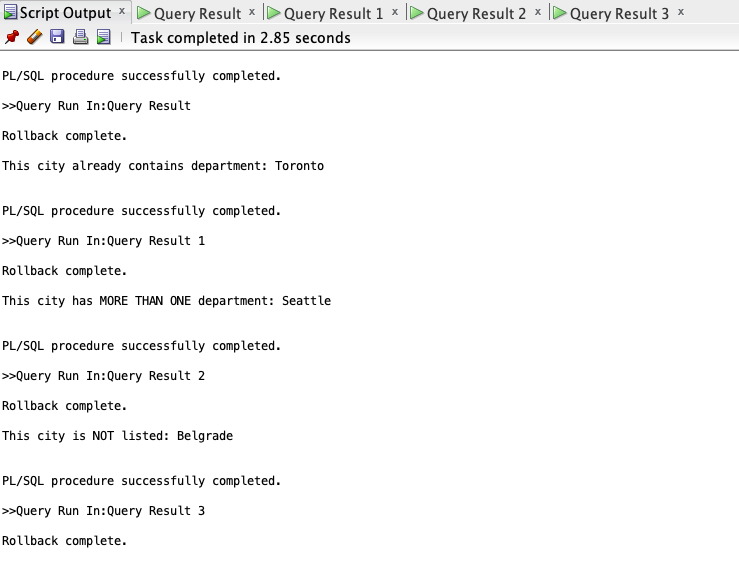
SELECT \*

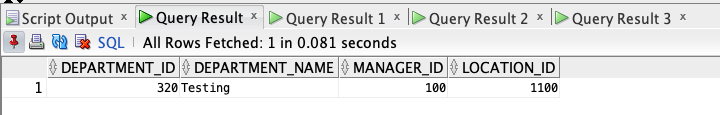
FROM DEPARTMENTS

WHERE DEPARTMENT\_ID = 320;

ROLLBACK

**Output:**

****

****

2) Write the PL/SQL block that will as input accept the start of the Course Description (one or more words) that is **definitely** used as a PREREQUISITE course. Then for each section of all courses that have this course (or courses) as a Prerequisite, determine the number of students registered. If this number is equal to or greater than 7, raise the user-defined exception and display message like “There are too many students for that section”. Otherwise, display how many students are in that section. **Use For Cursor Loop and Exception Handler as well.** Make sure that your program is able to process all sections. Also, you need to deal with the situations where input points to a valid course that is NO prerequisite at all and also if input points to a non-existing course. Here are the possible outputs:

Top of Form

|  |  |  |
| --- | --- | --- |
| Enter the beginning of the Course Description in UPPER case: |  |  |

Bottom of Form

**There are 5 students for section ID 85   
There are 6 students for section ID 86   
There are too many students for section 87   
^^^^^^^^^^^^^^^^^^^^^^^^^^   
There are 5 students for section ID 88   
There are too many students for section 89   
^^^^^^^^^^^^^^^^^^^^^^^^^^   
There are 4 students for section ID 90   
There are 2 students for section ID 91   
There are 4 students for section ID 92   
There are 0 students for section ID 93   
There are 0 students for section ID 98   
There are 4 students for section ID 146   
There are too many students for section 147   
^^^^^^^^^^^^^^^^^^^^^^^^^^   
There are 5 students for section ID 148   
There are 1 students for section ID 149   
There are 3 students for section ID 150   
There are 2 students for section ID 151   
PL/SQL procedure successfully completed.**

Top of Form

|  |  |  |
| --- | --- | --- |
| Enter the beginning of the Course Description in UPPER case: |  |  |

Bottom of Form

**There is NO PREREQUISITE course that starts on: UNIX. Try again.   
PL/SQL procedure successfully completed.**

Top of Form

|  |  |  |
| --- | --- | --- |
| Enter the beginning of the Course Description in UPPER case: |  |  |

Bottom of Form

**There is NO VALID course that starts on: SPORT. Try again.   
PL/SQL procedure successfully completed**.

**Code:**

SET SERVEROUTPUT ON

SET VERIFY OFF

ACCEPT input PROMPT 'Enter the beginning of the Course Description in UPPER case: ';

DECLARE

CURSOR getInfo IS SELECT SECTION\_ID FROM SECTION

WHERE course\_no IN (

SELECT course\_no FROM COURSE

WHERE prerequisite IN (

SELECT course\_no FROM COURSE

WHERE REGEXP\_LIKE(description, '^(&input\s)', 'i')

)

) ORDER BY section\_id;

CURSOR checkCourse IS SELECT \* FROM COURSE

WHERE REGEXP\_LIKE(description, '^(&input\s)', 'i');

sec\_rec getInfo%ROWTYPE;

course\_rec checkCourse%ROWTYPE;

total NUMBER(5) := 0;

errID NUMBER(5);

too\_many\_stud EXCEPTION;

no\_preReq EXCEPTION;

not\_Valid EXCEPTION;

BEGIN

OPEN getInfo;

FETCH getInfo INTO sec\_rec;

IF getInfo%NOTFOUND THEN

OPEN checkCourse;

FETCH checkCourse into course\_rec;

IF checkCourse%NOTFOUND THEN

RAISE not\_Valid;

ELSE

RAISE no\_preReq;

END IF;

CLOSE checkCourse;

CLOSE getInfo;

ELSE

CLOSE getInfo;

FOR i in getInfo LOOP

SELECT Count(Student\_id) INTO total from enrollment

where section\_id = i.section\_id;

IF total < 7 THEN

dbms\_output.put\_line('There are ' || total || ' students for section ID ' || i.section\_id);

ELSE

BEGIN

RAISE too\_many\_stud;

EXCEPTION

WHEN too\_many\_stud THEN

dbms\_output.put\_line('There are too many students for section ' || i.section\_id);

dbms\_output.put\_line('^^^^^^^^^^^^^^^^^^^^^^^^^^');

END;

END IF;

END LOOP;

END IF;

EXCEPTION

WHEN no\_preReq THEN

dbms\_output.put\_line('There is NO PREREQUISITE course that starts on: ' || UPPER('&input') || '. Try again.');

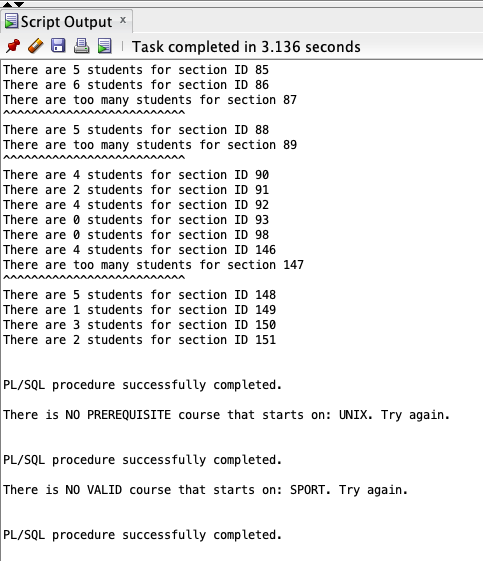
WHEN not\_Valid THEN

dbms\_output.put\_line('There is NO VALID course that starts on: ' || UPPER('&input') || '. Try again.');

END;

/

**Output:**

****

3) Write the PL/SQL block that will as input accept the start of the Course Description (one or more words) that possibly has a PREREQUISITE course. Then, it will display for each course that starts like the provided input the following components: Course Number, Description and Cost followed by the prerequisite Course Number, Description **and Cost**.. **Use For Cursor Loop. You should create a User-Defined Nested Record of four components, where the fourth component is Record again**. Also, you need to deal with the situations where your input points to a valid course that has NO prerequisites at all and also if your input points to a non-existing course. Here are the possible outputs:

Top of Form

|  |  |  |
| --- | --- | --- |
| Enter the beginning of the Course description in UPPER case: |  |  |

**Course: 144 - Database Design   
Cost: 1195   
Prerequisite: 420 - Database System Principles   
Prerequisite Cost: 1195   
========================================   
Course: 420 - Database System Principles   
Cost: 1195   
Prerequisite: 25 - Intro to Programming   
Prerequisite Cost: 1195   
========================================   
PL/SQL procedure successfully completed.**

Top of Form

|  |  |  |
| --- | --- | --- |
| Enter the beginning of the Course description in UPPER case: |  |  |

**There is NO prerequisite course for any course that starts on OPERATING. Try again.   
PL/SQL procedure successfully completed.**

Top of Form

|  |  |  |
| --- | --- | --- |
| Enter the beginning of the Course description in UPPER case: |  |  |

**There is NO VALID course that starts on: SPORT. Try again.   
PL/SQL procedure successfully completed.**

**Code:**

SET SERVEROUTPUT ON;

SET VERIFY OFF;

ACCEPT description PROMPT 'Enter the beginning of the Course description in UPPER case: ';

DECLARE

CURSOR c1 IS SELECT \* FROM COURSE WHERE PREREQUISITE IS NOT NULL AND UPPER(DESCRIPTION) LIKE UPPER('&description%');

TYPE Prereqcourse IS RECORD(PreCourseno NUMBER, Predescription VARCHAR2(50), Precost NUMBER);

TYPE coursereq IS RECORD(Courseno NUMBER, description VARCHAR2(50), cost NUMBER, Precourse Prereqcourse);

v\_course coursereq;

v\_pre NUMBER(5);

v\_count NUMBER(5);

BEGIN

    SELECT COUNT(\*) INTO v\_pre FROM COURSE WHERE PREREQUISITE IS NOT NULL AND UPPER(DESCRIPTION) LIKE UPPER('&description%');

    SELECT count(\*) INTO v\_count FROM COURSE WHERE UPPER(DESCRIPTION) LIKE UPPER('&description%');

    IF v\_count > 0 THEN

        IF v\_pre > 0 THEN

            FOR cur\_course in c1

                LOOP

                    SELECT COURSE\_NO, DESCRIPTION, COST INTO v\_course.Courseno, v\_course.description, v\_course.cost

                    FROM COURSE WHERE COURSE\_NO LIKE cur\_course.COURSE\_NO;

                    SELECT COURSE\_NO, DESCRIPTION, COST INTO v\_course.Precourse.PreCourseno, v\_course.Precourse.Predescription, v\_course.Precourse.Precost

                    FROM COURSE WHERE COURSE\_NO LIKE cur\_course.PREREQUISITE;

                    DBMS\_OUTPUT.PUT\_LINE('Course: ' || v\_course.Courseno || ' - ' || v\_course.description);

                    DBMS\_OUTPUT.PUT\_LINE('Cost: ' || v\_course.cost);

                    DBMS\_OUTPUT.PUT\_LINE('Prerequisite Course: ' || v\_course.Precourse.PreCourseno || ' - ' || v\_course.Precourse.Predescription);

                    DBMS\_OUTPUT.PUT\_LINE('Prerequisite Cost: ' || v\_course.Precourse.Precost);

                    DBMS\_OUTPUT.PUT\_LINE('======================================== ');

                END LOOP;

            ELSE

                DBMS\_OUTPUT.PUT\_LINE('There is NO prerequisite course that starts on &description. Try again.');

            END IF;

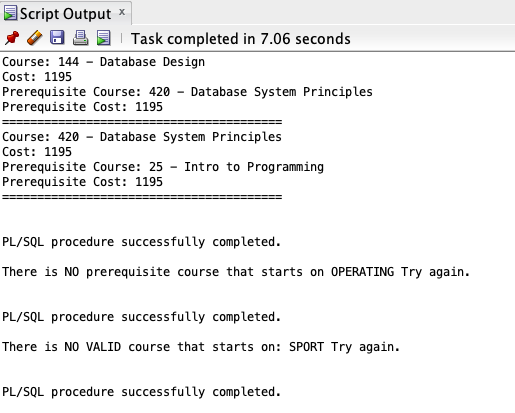
    ELSE

        DBMS\_OUTPUT.PUT\_LINE('There is NO VALID course that starts on: &description. Try again.');

    END IF;

END;

**Output:**



4) Write a PL/SQL block to display the course number and name and then for each section it will show the enrollment number. **Use Nested For Cursor Loops where the parent cursor will give information about each course and the child cursor will count the enrollment for each section. The child cursor will contain ONE input argument.**

You will be prompted to enter TWO key words within the course description and here are your possible outputs:

I) If you entered words JAVA and PROGRAM, the output looks like:

**20 Intro to Java Programming   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Section: 1 has an enrollment of: 4   
Section: 2 has an enrollment of: 8   
Section: 5 has an enrollment of: 3   
Section: 4 has an enrollment of: 1   
Section: 3 has an enrollment of: 5   
Section: 7 has an enrollment of: 2   
122 Intermediate Java Programming   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Section: 1 has an enrollment of: 4   
Section: 2 has an enrollment of: 3   
Section: 4 has an enrollment of: 5   
Section: 5 has an enrollment of: 8   
Section: 3 has an enrollment of: 4   
124 Advanced Java Programming   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Section: 1 has an enrollment of: 5   
Section: 2 has an enrollment of: 1   
Section: 3 has an enrollment of: 2   
146 Java for C/C++ Programmers   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Section: 1 has an enrollment of: 1   
Section: 2 has an enrollment of: 2   
450 DB Programming in Java   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
Section: 1 has an enrollment of: 1   
PL/SQL procedure successfully completed.**

II) If you entered words INTRO and C, the output looks like:

**20 Intro to Computers   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Section: 2 has an enrollment of: 3   
Section: 4 has an enrollment of: 2   
Section: 8 has an enrollment of: 2   
Section: 7 has an enrollment of: 2   
240 Intro to the Basic Language   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Section: 1 has an enrollment of: 12   
Section: 2 has an enrollment of: 1   
PL/SQL procedure successfully completed.**

III) If you entered words INTRO and SOCCER, the output looks like:

**There is NO course containing these 2 words. Try again.   
PL/SQL procedure successfully completed.**

**Code:**

SET SERVEROUTPUT ON;

SET VERIFY OFF;

ACCEPT prog PROMPT 'Enter key1 for the program name: ';

ACCEPT prog2 PROMPT 'Enter key2 for the program name: ';

DECLARE

CURSOR sectioncursor IS

SELECT COURSE\_NO, DESCRIPTION

FROM COURSE

WHERE UPPER(DESCRIPTION) LIKE UPPER('%&prog%')

AND UPPER(DESCRIPTION) LIKE UPPER('%&prog2%')

ORDER BY 1;

CURSOR enrollmentcursor (coursenum SECTION.COURSE\_NO%TYPE) IS

SELECT COUNT(e.STUDENT\_ID) AS NUMENROLLED, s.SECTION\_NO

FROM SECTION s LEFT OUTER JOIN ENROLLMENT e

ON s.SECTION\_ID = e.SECTION\_ID

WHERE s.COURSE\_NO = coursenum

GROUP BY S.SECTION\_NO;

sectionrow sectioncursor%ROWTYPE;

enrollmentrow enrollmentcursor%ROWTYPE;

BEGIN

OPEN sectioncursor;

FETCH sectioncursor INTO sectionrow;

IF sectioncursor%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('There is NO course containing these 2 words. Please try again');

ELSE

CLOSE sectioncursor;

FOR sect IN sectioncursor LOOP

DBMS\_OUTPUT.PUT\_LINE(sect.COURSE\_NO || ' ' || sect.DESCRIPTION);

DBMS\_OUTPUT.PUT\_LINE('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

OPEN enrollmentcursor(sect.COURSE\_NO); --Check if we have rows

FETCH enrollmentcursor INTO enrollmentrow;

IF enrollmentcursor%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('This course has no assigned sections');

ELSE

CLOSE enrollmentcursor;

FOR enroll IN enrollmentcursor(sect.COURSE\_NO) LOOP

DBMS\_OUTPUT.PUT\_LINE('Section: ' || enroll.SECTION\_NO || ' has an enrollment of: ' || enroll.NUMENROLLED);

END LOOP;

END IF;

END LOOP;

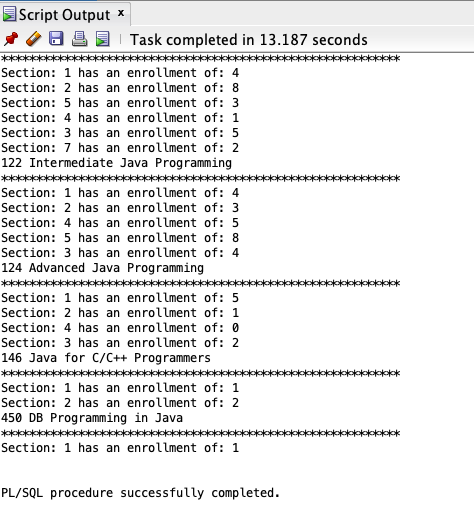
END IF;

END;

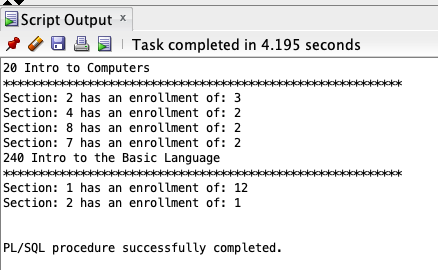
/

**Output:**

JAVA and PROGRAM

****

INTRO and C

****

INTRO and SOCCER

