Assignment 1:

Data Base for Q#1:

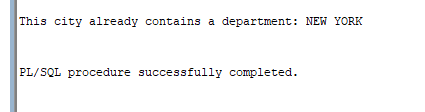
|  |
| --- |
| -- Create the Locations table  CREATE TABLE Locations (  Location\_id NUMBER PRIMARY KEY,  City VARCHAR2(50)  );  -- Insert sample data into Locations  INSERT INTO Locations (Location\_id, City) VALUES (1100, 'New York');  INSERT INTO Locations (Location\_id, City) VALUES (1200, 'Los Angeles');  -- Create the Employees table  CREATE TABLE Employees (  Employee\_id NUMBER PRIMARY KEY,  First\_name VARCHAR2(50),  Last\_name VARCHAR2(50),  Manager\_id NUMBER,  Hire\_date DATE  );  -- Insert sample data into Employees  INSERT INTO Employees (Employee\_id, First\_name, Last\_name, Manager\_id, Hire\_date) VALUES (100, 'Manager', 'Smith', NULL, TO\_DATE('2023-01-15', 'YYYY-MM-DD'));  INSERT INTO Employees (Employee\_id, First\_name, Last\_name, Manager\_id, Hire\_date) VALUES (101, 'John', 'Doe', 100, TO\_DATE('2023-02-10', 'YYYY-MM-DD'));  INSERT INTO Employees (Employee\_id, First\_name, Last\_name, Manager\_id, Hire\_date) VALUES (102, 'Jane', 'Smith', 100, TO\_DATE('2023-03-05', 'YYYY-MM-DD'));  -- Create the Departments table  CREATE TABLE Departments (  Department\_id NUMBER PRIMARY KEY,  Department\_name VARCHAR2(50),  Manager\_id NUMBER,  Location\_id NUMBER,  FOREIGN KEY (Manager\_id) REFERENCES Employees(Employee\_id),  FOREIGN KEY (Location\_id) REFERENCES Locations(Location\_id)  );  -- Insert sample data into Departments  INSERT INTO Departments (Department\_id, Department\_name, Manager\_id, Location\_id) VALUES (320, 'Testing', 100, 1100);  INSERT INTO Departments (Department\_id, Department\_name, Manager\_id, Location\_id) VALUES (321, 'Designing', 100, 1200);  INSERT INTO Departments (Department\_id, Department\_name, Manager\_id, Location\_id) VALUES (322, 'Manufacturing', 100, 1200);  INSERT INTO Departments (Department\_id, Department\_name, Manager\_id, Location\_id) VALUES (320, 'Marketing', 100, 1200); |

Query 1:

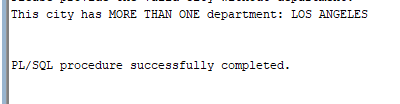
|  |
| --- |
| SET SERVEROUTPUT ON  DECLARE  v\_new\_department\_id NUMBER;  v\_department\_name VARCHAR2(50) := 'Testing';  v\_manager\_id NUMBER;  v\_location\_id NUMBER;  v\_city\_name VARCHAR2(50);  BEGIN  -- Prompt for a valid city name without any department  DBMS\_OUTPUT.PUT('Please provide the valid city without department: ');  DBMS\_OUTPUT.NEW\_LINE;  v\_city\_name := UPPER('&v\_city\_name');  -- Check if the city is not listed in the Locations table  BEGIN  SELECT Location\_id INTO v\_location\_id  FROM Locations  WHERE UPPER(City) = v\_city\_name;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('This city is NOT listed: ' || v\_city\_name);  RETURN;  END;  -- Check if the city already contains one department  SELECT COUNT(\*) INTO v\_manager\_id  FROM Departments  WHERE Location\_id = v\_location\_id;  IF v\_manager\_id = 1 THEN  DBMS\_OUTPUT.PUT\_LINE('This city already contains a department: ' || v\_city\_name);  -- Check if the city contains more than one department  ELSIF v\_manager\_id > 1 THEN  DBMS\_OUTPUT.PUT\_LINE('This city has MORE THAN ONE department: ' || v\_city\_name);  ELSE  -- Find the manager with the most direct reports  SELECT e.Manager\_id INTO v\_manager\_id  FROM Employees e  JOIN Departments d ON e.Employee\_id = d.Manager\_id  WHERE d.Location\_id = v\_location\_id  GROUP BY e.Manager\_id  HAVING COUNT(\*) = (  SELECT MAX(report\_count)  FROM (  SELECT COUNT(\*) AS report\_count  FROM Employees e2  JOIN Departments d2 ON e2.Employee\_id = d2.Manager\_id  WHERE d2.Location\_id = v\_location\_id  GROUP BY e2.Manager\_id  )  );  -- Get the highest existing Department\_id and increase it by 50  SELECT NVL(MAX(Department\_id), 0) + 50 INTO v\_new\_department\_id  FROM Departments;  -- Insert the new department  INSERT INTO Departments (Department\_id, Department\_name, Manager\_id, Location\_id)  VALUES (v\_new\_department\_id, v\_department\_name, v\_manager\_id, v\_location\_id);  -- Display the new row  DBMS\_OUTPUT.PUT\_LINE('DEPARTMENT\_ID ' || v\_new\_department\_id || ' ' || v\_department\_name || ' ' || v\_manager\_id || ' ' || v\_location\_id);  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);  END;  /  SET SERVEROUTPUT OFF |

Output:

User input: New York



User input: los Angeles



User input: Belgrade

A screenshot of a computer screen

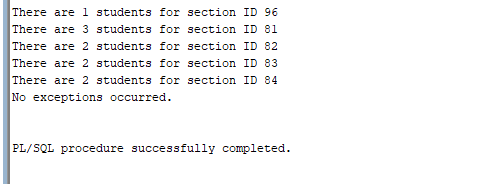
Description automatically generated

Query 2:

|  |
| --- |
| SET SERVEROUTPUT ON  ACCEPT v\_input\_description CHAR PROMPT 'Enter the beginning of the Course Description in UPPER case: ';  DECLARE  v\_search\_description VARCHAR2(50);  v\_has\_exceptions NUMBER := 0;  v\_course\_found NUMBER := 0; -- Variable to track if a course with the given description is found  v\_no\_prereq NUMBER := 1; -- Variable to track if there's no prerequisite  BEGIN  v\_search\_description := UPPER('&v\_input\_description');  -- Initialize a cursor for selecting sections  FOR rec IN (SELECT s.SECTION\_ID, s.COURSE\_NO, s.CAPACITY  FROM SECTION s  WHERE s.COURSE\_NO IN (  SELECT c.PREREQUISITE  FROM COURSE c  WHERE UPPER(c.DESCRIPTION) LIKE v\_search\_description || '%'  ))  LOOP  v\_no\_prereq := 0; -- Reset the variable when a matching course is found  -- Initialize a variable to count the number of students  DECLARE  v\_student\_count NUMBER := 0;  v\_section\_id NUMBER := rec.SECTION\_ID;  BEGIN  -- Use a cursor for loop to iterate through enrollments  FOR enr\_rec IN (SELECT e.STUDENT\_ID  FROM ENROLLMENT e  WHERE e.SECTION\_ID = rec.SECTION\_ID)  LOOP  -- Count the students  v\_student\_count := v\_student\_count + 1;  END LOOP;  -- Check if the student count is greater than or equal to 7  IF v\_student\_count >= 7 THEN  -- Raise the custom exception  DBMS\_OUTPUT.PUT\_LINE('There are more than 7 students for section ' || rec.COURSE\_NO);  DBMS\_OUTPUT.PUT\_LINE('^^^^^^^^^^^^^^^^^^^^^^^^^^');  v\_has\_exceptions := 1;  ELSE  -- Display the student count  DBMS\_OUTPUT.PUT\_LINE('There are ' || v\_student\_count || ' students for section ID ' || v\_section\_id);  END IF;  EXCEPTION  WHEN OTHERS THEN  -- Handle other exceptions  DBMS\_OUTPUT.PUT\_LINE('An error occurred for section ' || v\_section\_id);  END;  END LOOP;  -- Check if there were any exceptions  IF v\_has\_exceptions = 0 THEN  -- No exceptions occurred  DBMS\_OUTPUT.PUT\_LINE('No exceptions occurred.');  END IF;  -- Check if any course with the given description was found  SELECT COUNT(\*) INTO v\_course\_found  FROM COURSE c  WHERE UPPER(c.DESCRIPTION) LIKE v\_search\_description || '%';  -- Check if no prerequisite course is found  IF v\_no\_prereq = 1 AND v\_course\_found <> 0 THEN  DBMS\_OUTPUT.PUT\_LINE('No pre-req of this course.');  END IF;    IF v\_course\_found = 0 THEN  -- No course with the given description found  DBMS\_OUTPUT.PUT\_LINE('No valid course.');  END IF;  END;  /  SET SERVEROUTPUT OFF |

Output:

User input : STRUCTURED



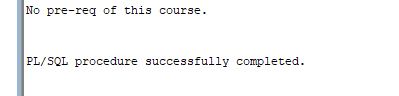
|  |
| --- |
| There are 1 students for section ID 96  There are 3 students for section ID 81  There are 2 students for section ID 82  There are 2 students for section ID 83  There are 2 students for section ID 84  No exceptions occurred.  PL/SQL procedure successfully completed. |

User input: UNIX

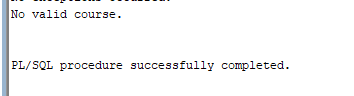
A screenshot of a computer error

Description automatically generated

User input: DP



User input: SPORT

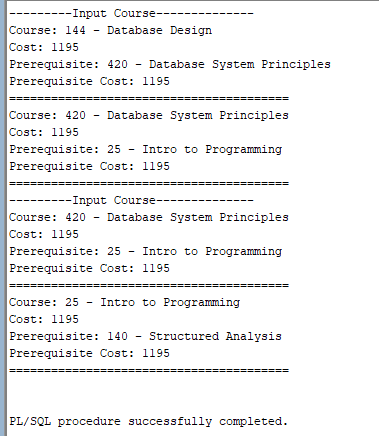


Query 3:

|  |
| --- |
| SET SERVEROUTPUT ON  ACCEPT v\_input\_description CHAR PROMPT 'Enter the beginning of the Course description in UPPER case: ';  DECLARE  v\_search\_description VARCHAR2(100);  BEGIN  -- Assign the user input to the search variable  v\_search\_description := UPPER('&v\_input\_description');  -- Initialize a cursor for selecting course information  FOR rec IN (SELECT c.COURSE\_NO, c.DESCRIPTION, c.COST, c.PREREQUISITE  FROM COURSE c  WHERE UPPER(c.DESCRIPTION) LIKE v\_search\_description || '%')  LOOP  -- Check if any courses were found  IF rec.COURSE\_NO IS NOT NULL THEN  -- Display course information    -- Check if there is a prerequisite, and if so, print its information  IF rec.PREREQUISITE IS NOT NULL THEN  DBMS\_OUTPUT.PUT\_LINE('---------Input Course--------------');  DBMS\_OUTPUT.PUT\_LINE('Course: ' || rec.COURSE\_NO || ' - ' || rec.DESCRIPTION);  DBMS\_OUTPUT.PUT\_LINE('Cost: ' || rec.COST);  DECLARE  v\_prerequisite\_course\_no NUMBER;  v\_prerequisite\_description VARCHAR2(100);  v\_prerequisite\_cost NUMBER;  v\_prerequisite\_prereq NUMBER;  v2\_prerequisite\_course\_no NUMBER;  v2\_prerequisite\_description VARCHAR2(100);  v2\_prerequisite\_cost NUMBER;  v2\_prerequisite\_prereq NUMBER;  BEGIN  SELECT c.COURSE\_NO, c.DESCRIPTION, c.COST, c.prerequisite  INTO v\_prerequisite\_course\_no, v\_prerequisite\_description, v\_prerequisite\_cost, v\_prerequisite\_prereq  FROM COURSE c  WHERE c.COURSE\_NO = rec.PREREQUISITE;  SELECT c.COURSE\_NO, c.DESCRIPTION, c.COST, c.prerequisite  INTO v2\_prerequisite\_course\_no, v2\_prerequisite\_description, v2\_prerequisite\_cost, v2\_prerequisite\_prereq  FROM COURSE c  WHERE c.COURSE\_NO = v\_prerequisite\_prereq;  DBMS\_OUTPUT.PUT\_LINE('Prerequisite: ' || v\_prerequisite\_course\_no || ' - ' || v\_prerequisite\_description);  DBMS\_OUTPUT.PUT\_LINE('Prerequisite Cost: ' || v\_prerequisite\_cost);  DBMS\_OUTPUT.PUT\_LINE('========================================');  DBMS\_OUTPUT.PUT\_LINE('Course: ' || v\_prerequisite\_course\_no || ' - ' || v\_prerequisite\_description );  DBMS\_OUTPUT.PUT\_LINE('Cost: ' || v\_prerequisite\_cost);  DBMS\_OUTPUT.PUT\_LINE('Prerequisite: ' || v2\_prerequisite\_course\_no || ' - ' || v2\_prerequisite\_description);  DBMS\_OUTPUT.PUT\_LINE('Prerequisite Cost: ' || v2\_prerequisite\_cost);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Prerequisite information not found.');  END;  ELSE  DBMS\_OUTPUT.PUT\_LINE('NO prereq for this course found');  END IF;  DBMS\_OUTPUT.PUT\_LINE('========================================');  ELSE  DBMS\_OUTPUT.PUT\_LINE('There is NO VALID course that starts with: ' || v\_search\_description || '. Try again.');  END IF;  END LOOP;    END;  /  SET SERVEROUTPUT OFF |

Output:

User input: DATABASE



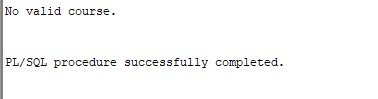
|  |
| --- |
| ---------Input Course--------------  Course: 144 - Database Design  Cost: 1195  Prerequisite: 420 - Database System Principles  Prerequisite Cost: 1195  ========================================  Course: 420 - Database System Principles  Cost: 1195  Prerequisite: 25 - Intro to Programming  Prerequisite Cost: 1195  ========================================  ---------Input Course--------------  Course: 420 - Database System Principles  Cost: 1195  Prerequisite: 25 - Intro to Programming  Prerequisite Cost: 1195  ========================================  Course: 25 - Intro to Programming  Cost: 1195  Prerequisite: 140 - Structured Analysis  Prerequisite Cost: 1195  ========================================  PL/SQL procedure successfully completed. |

User input: DP

A black text on a white background

Description automatically generated

User input: SPORT



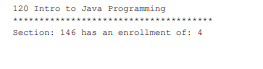
Query 4:

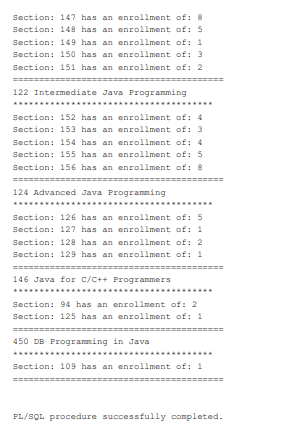
|  |
| --- |
| SET SERVEROUTPUT ON  -- Accept two keywords for course description  ACCEPT keyword1 CHAR PROMPT 'Enter the first keyword within the course description: ';  ACCEPT keyword2 CHAR PROMPT 'Enter the second keyword within the course description: ';  -- Declare variables for course information  DECLARE  v\_keyword1 VARCHAR2(100) := UPPER('&keyword1');  v\_keyword2 VARCHAR2(100) := UPPER('&keyword2');  v\_course\_found BOOLEAN := FALSE;  BEGIN  -- Initialize a cursor for selecting course information  FOR rec\_course IN (SELECT c.COURSE\_NO, c.DESCRIPTION FROM COURSE c  WHERE UPPER(c.DESCRIPTION) LIKE '%' || v\_keyword1 || '%' AND UPPER(c.DESCRIPTION) LIKE '%' || v\_keyword2 || '%')  LOOP  v\_course\_found := TRUE;  -- Display course number and name  DBMS\_OUTPUT.PUT\_LINE(rec\_course.COURSE\_NO || ' ' || rec\_course.DESCRIPTION);  DBMS\_OUTPUT.PUT\_LINE('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');  -- Initialize a child cursor to count enrollments for each section  FOR rec\_section IN (SELECT s.SECTION\_ID, COUNT(\*) AS ENROLLMENTS  FROM SECTION s  LEFT JOIN ENROLLMENT e ON s.SECTION\_ID = e.SECTION\_ID  WHERE s.COURSE\_NO = rec\_course.COURSE\_NO  GROUP BY s.SECTION\_ID  ORDER BY s.SECTION\_ID)  LOOP  -- Display section number and enrollment count  DBMS\_OUTPUT.PUT\_LINE('Section: ' || rec\_section.SECTION\_ID || ' has an enrollment of: ' || rec\_section.ENROLLMENTS);  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('========================================');  END LOOP;  IF v\_course\_found = FALSE THEN  DBMS\_OUTPUT.PUT\_LINE('There is NO course containing these 2 words. Try again.');  END IF;  END;  /  SET SERVEROUTPUT OFF |

Output:

User input 1: JAVA

User input 2: PROGRAM





|  |
| --- |
| 120 Intro to Java Programming  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Section: 146 has an enrollment of: 4  Section: 147 has an enrollment of: 8  Section: 148 has an enrollment of: 5  Section: 149 has an enrollment of: 1  Section: 150 has an enrollment of: 3  Section: 151 has an enrollment of: 2  ========================================  122 Intermediate Java Programming  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Section: 152 has an enrollment of: 4  Section: 153 has an enrollment of: 3  Section: 154 has an enrollment of: 4  Section: 155 has an enrollment of: 5  Section: 156 has an enrollment of: 8  ========================================  124 Advanced Java Programming  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Section: 126 has an enrollment of: 5  Section: 127 has an enrollment of: 1  Section: 128 has an enrollment of: 2  Section: 129 has an enrollment of: 1  ========================================  146 Java for C/C++ Programmers  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Section: 94 has an enrollment of: 2  Section: 125 has an enrollment of: 1  ========================================  450 DB Programming in Java  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Section: 109 has an enrollment of: 1  ========================================  PL/SQL procedure successfully completed. |

User input 1 = INTRO

User input 2 = C

A screenshot of a computer program

Description automatically generated

User input 1 = INTRO

User input 2 = SOCCER

A white background with black text

Description automatically generated