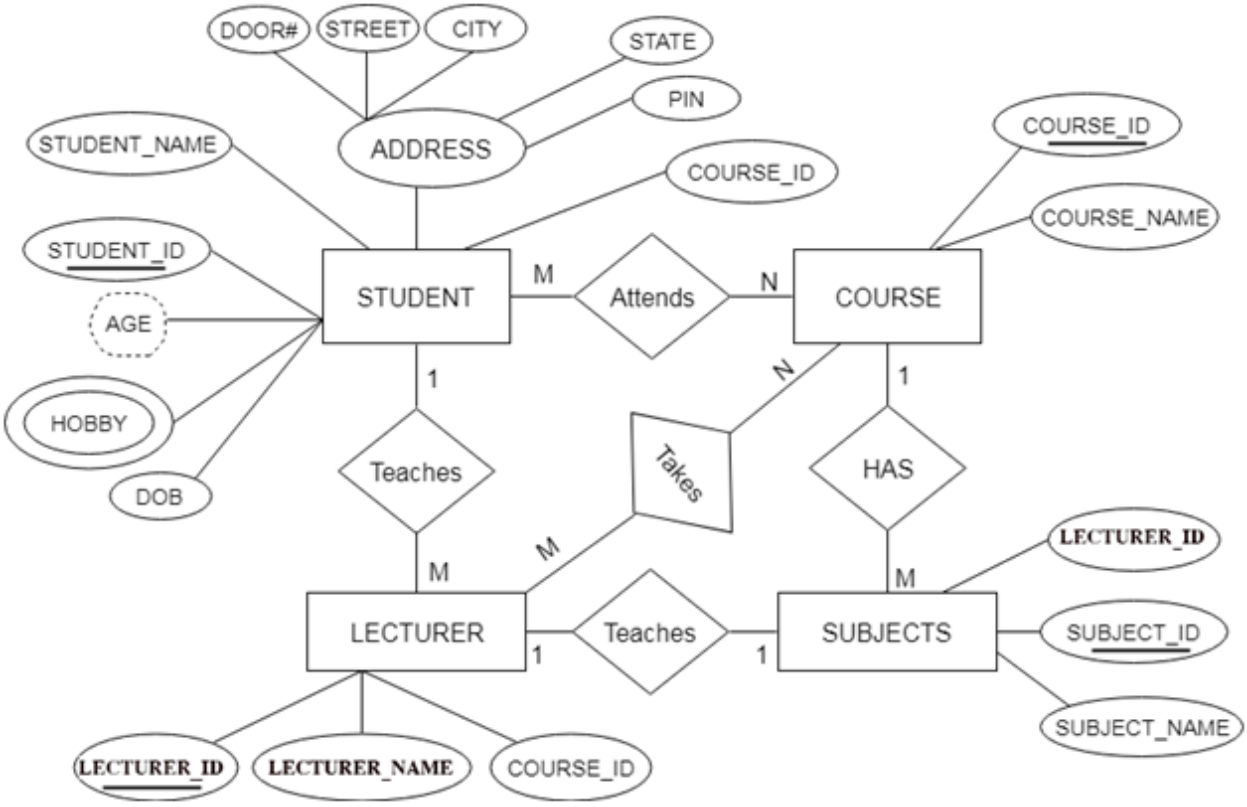
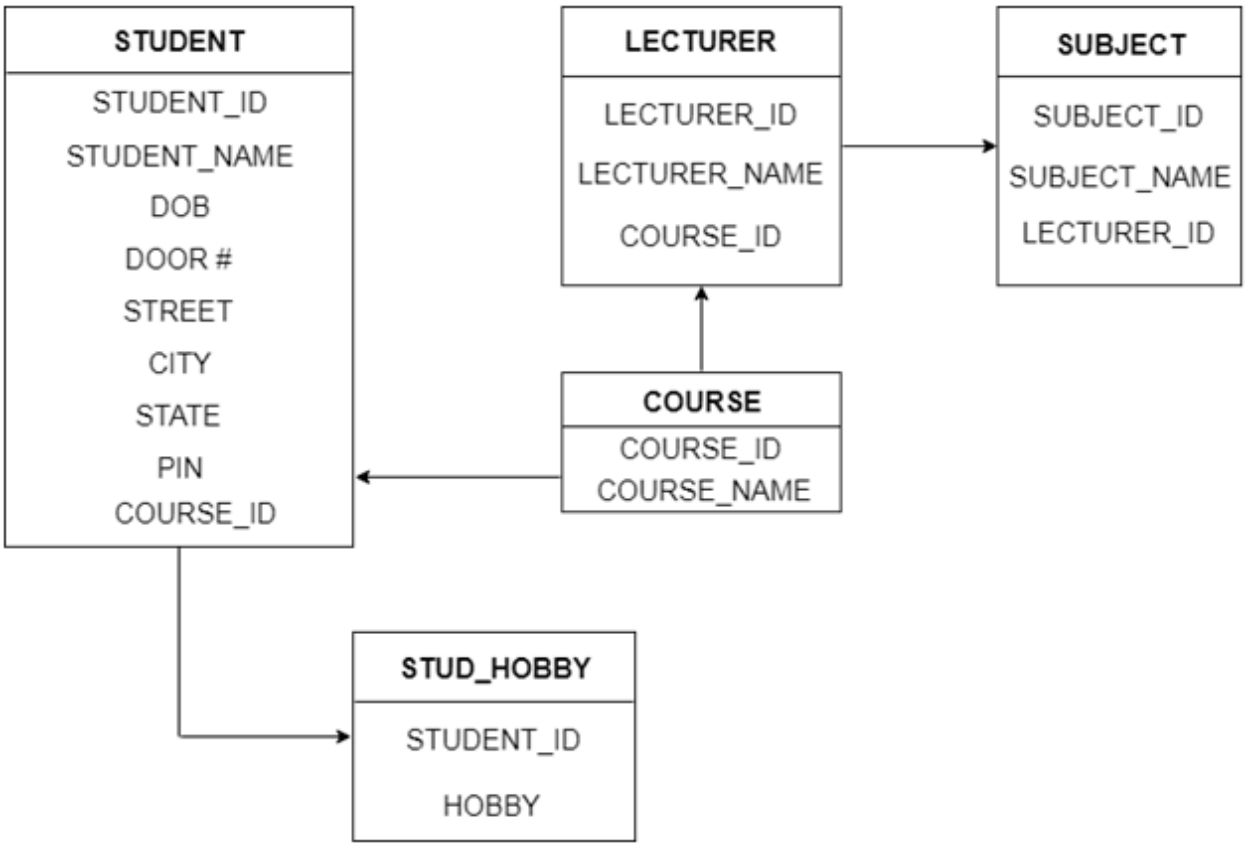
ER Diagram




Relational Scheme




normalized tables:

Student Table

Attributes:

Student_ID (Primary Key)

Student_Name

Age

DOB

Hobby

Address Table

Attributes:

Door#

Street

City

State

PIN

Student_ID (Foreign Key)

Course Table

Attributes:

Course_ID (Primary Key)

Course_Name

Lecturer Table

Attributes:

Lecturer_ID (Primary Key)

Lecturer_Name

Subjects Table

Attributes:

Subject_ID (Primary Key)

Subject_Name

Attends Table (Relationship between Student and Course)

Attributes:

Student_ID (Foreign Key)

Course_ID (Foreign Key)

Teaches Table (Relationship between Lecturer and Course)

Attributes:

Lecturer_ID (Foreign Key)

Course_ID (Foreign Key)

Takes Table (Relationship between Student and Subjects)

Attributes:

Student_ID (Foreign Key)

Subject_ID (Foreign Key)

. Has Table (Relationship between Course and Subjects)

Attributes:

Course_ID (Foreign Key)

Subject_ID (Foreign Key)

Lecturer_Subjects Table (Relationship between Lecturer and Subjects)

Attributes:

Lecturer_ID (Foreign Key)

Subject_ID (Foreign Key)

Create normalize table:

```
SQL> CREATE TABLE Students_ (
  2      Student_ID INT PRIMARY KEY,
  3      Student_Name VARCHAR(255),
  4      Age INT,
  5      DOB DATE,
  6      Hobby VARCHAR(255)
  7  );

Table created.
```

```
SQL> CREATE TABLE Address (
  2      Door VARCHAR(255),
  3      Street VARCHAR(255),
  4      City VARCHAR(255),
  5      State VARCHAR(255),
  6      PIN VARCHAR(10),
  7      Student_ID INT,
  8      FOREIGN KEY (Student_ID) REFERENCES Students_(Student_ID)
  9  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Courses_ (
  2      Course_ID INT PRIMARY KEY,
  3      Course_Name VARCHAR(255)
  4  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Lecturer (
  2      Lecturer_ID INT PRIMARY KEY,
  3      Lecturer_Name VARCHAR(255)
  4  );

Table created.
```

```
SQL> CREATE TABLE Subjects (
  2      Subject_ID INT PRIMARY KEY,
  3      Subject_Name VARCHAR(255)
  4  );

Table created.
```

```
SQL> CREATE TABLE Attends (
  2      Student_ID INT,
  3      Course_ID INT,
  4      PRIMARY KEY (Student_ID, Course_ID),
  5      FOREIGN KEY (Student_ID) REFERENCES Students_(Student_ID),
  6      FOREIGN KEY (Course_ID) REFERENCES Courses_(Course_ID)
  7  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Teaches (
  2      Lecturer_ID INT,
  3      Course_ID INT,
  4      PRIMARY KEY (Lecturer_ID, Course_ID),
  5      FOREIGN KEY (Lecturer_ID) REFERENCES Lecturer(Lecturer_ID),
  6      FOREIGN KEY (Course_ID) REFERENCES Courses_(Course_ID)
  7  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Takes (
  2      Student_ID INT,
  3      Subject_ID INT,
  4      PRIMARY KEY (Student_ID, Subject_ID),
  5      FOREIGN KEY (Student_ID) REFERENCES Students_(Student_ID),
  6      FOREIGN KEY (Subject_ID) REFERENCES Subjects(Subject_ID)
  7  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Has (
  2      Course_ID INT,
  3      Subject_ID INT,
  4      PRIMARY KEY (Course_ID, Subject_ID),
  5      FOREIGN KEY (Course_ID) REFERENCES Courses_(Course_ID),
  6      FOREIGN KEY (Subject_ID) REFERENCES Subjects(Subject_ID)
  7  );

Table created.

SQL>
```

```
SQL> CREATE TABLE Lecturer_Subjects (
  2      Lecturer_ID INT,
  3      Subject_ID INT,
  4      PRIMARY KEY (Lecturer_ID, Subject_ID),
  5      FOREIGN KEY (Lecturer_ID) REFERENCES Lecturer(Lecturer_ID),
  6      FOREIGN KEY (Subject_ID) REFERENCES Subjects(Subject_ID)
  7  );

Table created.

SQL>
```

Insert 5 rows in each table:

```
SQL> INSERT INTO Students_ (Student_ID, Student_Name, Age, DOB, Hobby) VALUES (1, 'Alice', 20, TO_DATE('2003-01-01', 'YYYY-MM-DD'), 'Reading');
1 row created.
SQL> INSERT INTO Students_ (Student_ID, Student_Name, Age, DOB, Hobby) VALUES (2, 'Bob', 21, TO_DATE('2002-02-02', 'YYYY-MM-DD'), 'Swimming');
1 row created.
SQL> INSERT INTO Students_ (Student_ID, Student_Name, Age, DOB, Hobby) VALUES (3, 'Charlie', 22, TO_DATE('2001-03-03', 'YYYY-MM-DD'), 'Hiking');
1 row created.
SQL> INSERT INTO Students_ (Student_ID, Student_Name, Age, DOB, Hobby) VALUES (4, 'David', 23, TO_DATE('2000-04-04', 'YYYY-MM-DD'), 'Gaming');
1 row created.
SQL> INSERT INTO Students_ (Student_ID, Student_Name, Age, DOB, Hobby) VALUES (5, 'Eva', 24, TO_DATE('1999-05-05', 'YYYY-MM-DD'), 'Painting');
1 row created.
SQL>
```

```
SQL> INSERT INTO Address (Door, Street, City, State, PIN, Student_ID) VALUES ('12A', 'Main St', 'Springfield', 'IL', '62701', 1);
1 row created.
SQL> INSERT INTO Address (Door, Street, City, State, PIN, Student_ID) VALUES ('34B', '2nd Ave', 'Chicago', 'IL', '60601', 2);
1 row created.
SQL> INSERT INTO Address (Door, Street, City, State, PIN, Student_ID) VALUES ('56C', '3rd Blvd', 'Naperville', 'IL', '60540', 3);
1 row created.
SQL> INSERT INTO Address (Door, Street, City, State, PIN, Student_ID) VALUES ('78D', '4th St', 'Evanston', 'IL', '60201', 4);
1 row created.
SQL> INSERT INTO Address (Door, Street, City, State, PIN, Student_ID) VALUES ('90E', '5th Ave', 'Peoria', 'IL', '61602', 5);
1 row created.
```

```
SQL> INSERT INTO Courses_ (Course_ID, Course_Name) VALUES (101, 'Mathematics');
1 row created.
SQL> INSERT INTO Courses_ (Course_ID, Course_Name) VALUES (102, 'Physics');
1 row created.
SQL> INSERT INTO Courses_ (Course_ID, Course_Name) VALUES (103, 'Chemistry');
1 row created.
SQL> INSERT INTO Courses_ (Course_ID, Course_Name) VALUES (104, 'Biology');
1 row created.
SQL> INSERT INTO Courses_ (Course_ID, Course_Name) VALUES (105, 'Computer Science');
1 row created.
```

```
SQL> INSERT INTO Lecturer (Lecturer_ID, Lecturer_Name) VALUES (201, 'Dr. Smith');

1 row created.

SQL> INSERT INTO Lecturer (Lecturer_ID, Lecturer_Name) VALUES (202, 'Dr. Johnson');

1 row created.

SQL> INSERT INTO Lecturer (Lecturer_ID, Lecturer_Name) VALUES (203, 'Dr. Williams');

1 row created.

SQL> INSERT INTO Lecturer (Lecturer_ID, Lecturer_Name) VALUES (204, 'Dr. Brown');

1 row created.

SQL> INSERT INTO Lecturer (Lecturer_ID, Lecturer_Name) VALUES (205, 'Dr. Jones');

1 row created.

SQL>
```

```
SQL> INSERT INTO Subjects (Subject_ID, Subject_Name) VALUES (301, 'Algebra');

1 row created.

SQL> INSERT INTO Subjects (Subject_ID, Subject_Name) VALUES (302, 'Mechanics');

1 row created.

SQL> INSERT INTO Subjects (Subject_ID, Subject_Name) VALUES (303, 'Organic Chemistry');

1 row created.

SQL> INSERT INTO Subjects (Subject_ID, Subject_Name) VALUES (304, 'Genetics');

1 row created.

SQL> INSERT INTO Subjects (Subject_ID, Subject_Name) VALUES (305, 'Data Structures');

1 row created.

SQL>
```

```
SQL> INSERT INTO Attends (Student_ID, Course_ID) VALUES (1, 101);

1 row created.

SQL> INSERT INTO Attends (Student_ID, Course_ID) VALUES (2, 102);

1 row created.

SQL> INSERT INTO Attends (Student_ID, Course_ID) VALUES (3, 103);

1 row created.

SQL> INSERT INTO Attends (Student_ID, Course_ID) VALUES (4, 104);

1 row created.

SQL> INSERT INTO Attends (Student_ID, Course_ID) VALUES (5, 105);

1 row created.

SQL>
```

```
SQL> INSERT INTO Teaches (Lecturer_ID, Course_ID) VALUES (201, 101);

1 row created.

SQL> INSERT INTO Teaches (Lecturer_ID, Course_ID) VALUES (202, 102);

1 row created.

SQL> INSERT INTO Teaches (Lecturer_ID, Course_ID) VALUES (203, 103);

1 row created.

SQL> INSERT INTO Teaches (Lecturer_ID, Course_ID) VALUES (204, 104);

1 row created.

SQL> INSERT INTO Teaches (Lecturer_ID, Course_ID) VALUES (205, 105);

1 row created.

SQL>
```

```
SQL> INSERT INTO Takes (Student_ID, Subject_ID) VALUES (1, 301);

1 row created.

SQL> INSERT INTO Takes (Student_ID, Subject_ID) VALUES (2, 302);

1 row created.

SQL> INSERT INTO Takes (Student_ID, Subject_ID) VALUES (3, 303);

1 row created.

SQL> INSERT INTO Takes (Student_ID, Subject_ID) VALUES (4, 304);

1 row created.

SQL> INSERT INTO Takes (Student_ID, Subject_ID) VALUES (5, 305);

1 row created.
```

```
SQL> INSERT INTO Has (Course_ID, Subject_ID) VALUES (101, 301);

1 row created.

SQL> INSERT INTO Has (Course_ID, Subject_ID) VALUES (102, 302);

1 row created.

SQL> INSERT INTO Has (Course_ID, Subject_ID) VALUES (103, 303);

1 row created.

SQL> INSERT INTO Has (Course_ID, Subject_ID) VALUES (104, 304);

1 row created.

SQL> INSERT INTO Has (Course_ID, Subject_ID) VALUES (105, 305);

1 row created.

SQL>
```

```
SQL> INSERT INTO Lecturer_Subjects (Lecturer_ID, Subject_ID) VALUES (201, 301);

1 row created.

SQL> INSERT INTO Lecturer_Subjects (Lecturer_ID, Subject_ID) VALUES (202, 302);

1 row created.

SQL> INSERT INTO Lecturer_Subjects (Lecturer_ID, Subject_ID) VALUES (203, 303);

1 row created.

SQL> INSERT INTO Lecturer_Subjects (Lecturer_ID, Subject_ID) VALUES (204, 304);

1 row created.

SQL> INSERT INTO Lecturer_Subjects (Lecturer_ID, Subject_ID) VALUES (205, 305);

1 row created.

SQL>
```

data retrieval:

1) SELECT Student_ID, Student_Name, Age, DOB, Hobby

FROM Students_

WHERE Age > 21

ORDER BY Student_Name;

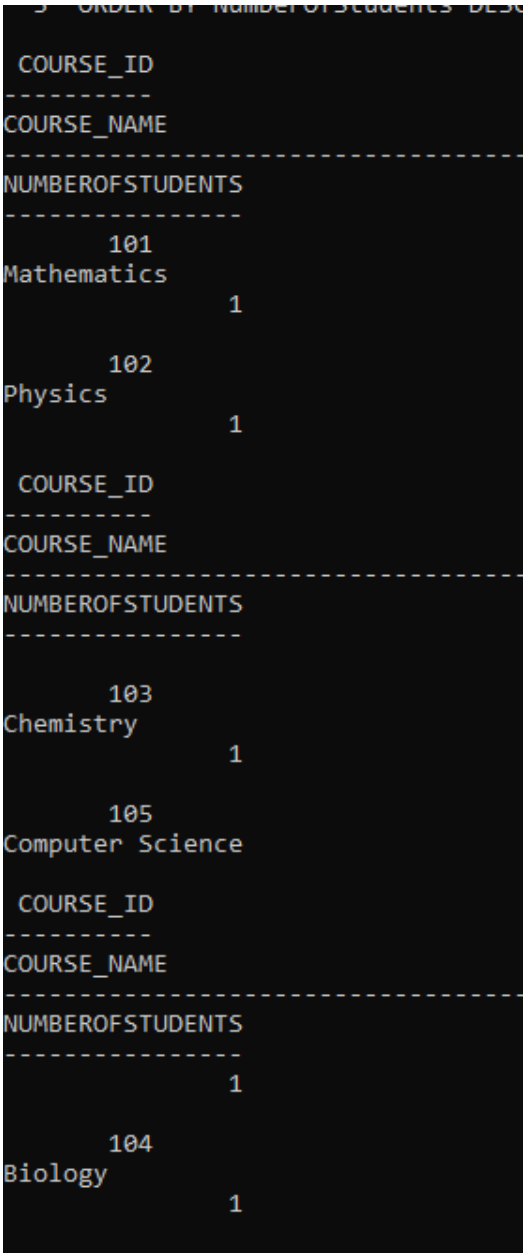

2) SELECT a.Course_ID, c.Course_Name, COUNT(a.Student_ID) AS NumberOfStudents

FROM Attends a

JOIN Courses_ c ON a.Course_ID = c.Course_ID

GROUP BY a.Course_ID, c.Course_Name

ORDER BY NumberOfStudents DESC;




3) SELECT s.Student_ID, s.Student_Name, c.Course_Name, l.Lecturer_Name

FROM Students_ s

JOIN Attends a ON s.Student_ID = a.Student_ID

JOIN Courses_ c ON a.Course_ID = c.Course_ID

JOIN Teaches t ON c.Course_ID = t.Course_ID

JOIN Lecturer l ON t.Lecturer_ID = l.Lecturer_ID;

```
STUDENT_ID
----------
STUDENT_NAME
--------------------------------------------------------------------------------
COURSE_NAME
--------------------------------------------------------------------------------
LECTURER_NAME
--------------------------------------------------------------------------------
         3
Charlie
Chemistry
Dr. Williams


STUDENT_ID
----------
STUDENT_NAME
--------------------------------------------------------------------------------
COURSE_NAME
--------------------------------------------------------------------------------
LECTURER_NAME
--------------------------------------------------------------------------------
         4
David
Biology
Dr. Brown


STUDENT_ID
----------
STUDENT_NAME
--------------------------------------------------------------------------------
COURSE_NAME
--------------------------------------------------------------------------------
LECTURER_NAME
--------------------------------------------------------------------------------
         5
Eva
Computer Science
Dr. Jones
```

4) SELECT s.Student_Name

FROM Students_ s

JOIN Attends a ON s.Student_ID = a.Student_ID

WHERE a.Course_ID = (

  SELECT a.Course_ID

  FROM Attends a

  GROUP BY a.Course_ID

  ORDER BY COUNT(a.Student_ID) DESC

  FETCH FIRST 1 ROWS ONLY

);

```
SQL> SELECT s.Student_Name
  2  FROM Students_ s
  3  JOIN Attends a ON s.Student_ID = a.Student_ID
  4  WHERE a.Course_ID = (
  5      SELECT a.Course_ID
  6      FROM Attends a
  7      GROUP BY a.Course_ID
  8      ORDER BY COUNT(a.Student_ID) DESC
  9      FETCH FIRST 1 ROWS ONLY
 10  );

STUDENT_NAME
--------------------------------------------------------------------------------
Eva

SQL>
```

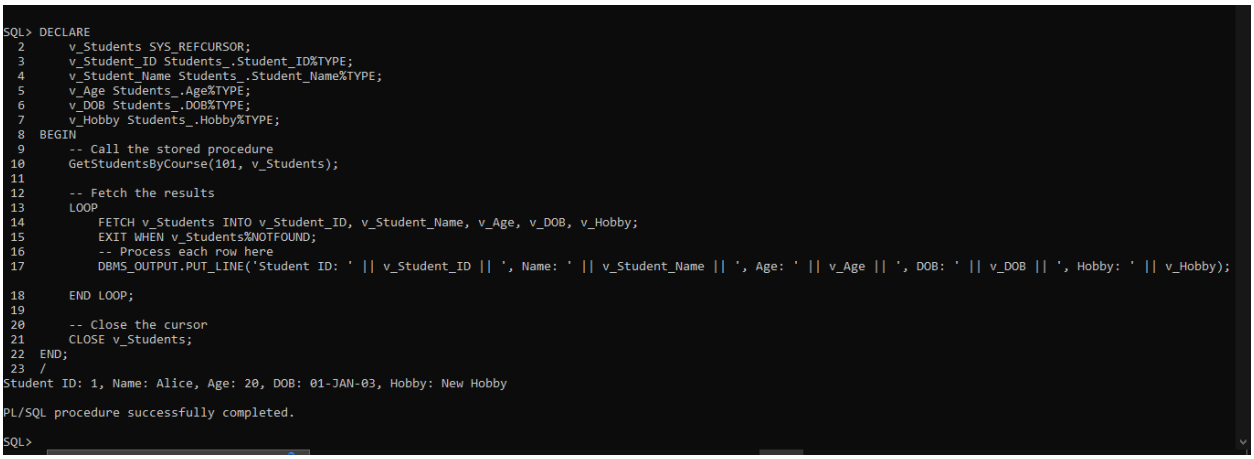
Creating the Procedure

```
CREATE OR REPLACE PROCEDURE GetStudentsByCourse (
   p_Course_ID IN NUMBER,
   o_Students OUT SYS_REFCURSOR
)
AS
BEGIN
   OPEN o_Students FOR
   SELECT s.Student_ID, s.Student_Name, s.Age, s.DOB, s.Hobby
   FROM Students_ s
   JOIN Attends a ON s.Student_ID = a.Student_ID
   WHERE a.Course_ID = p_Course_ID;
END GetStudentsByCourse;
/
DECLARE
   v_Students SYS_REFCURSOR;
```

```
    v_Student_ID Students_.Student_ID%TYPE;
    v_Student_Name Students_.Student_Name%TYPE;
    v_Age Students_.Age%TYPE;
    v_DOB Students_.DOB%TYPE;
    v_Hobby Students_.Hobby%TYPE;
BEGIN
    -- Call the stored procedure
    GetStudentsByCourse(101, v_Students);

    -- Fetch the results
    LOOP
        FETCH v_Students INTO v_Student_ID, v_Student_Name, v_Age, v_DOB, v_Hobby;
        EXIT WHEN v_Students%NOTFOUND;
        -- Process each row here
        DBMS_OUTPUT.PUT_LINE('Student ID: ' || v_Student_ID || ', Name: ' || v_Student_Name
|| ', Age: ' || v_Age || ', DOB: ' || v_DOB || ', Hobby: ' || v_Hobby);
    END LOOP;

    -- Close the cursor
    CLOSE v_Students;
END;
/
```



```
SQL> DECLARE
  2      v_Students SYS_REFCURSOR;
  3      v_Student_ID Students_.Student_ID%TYPE;
  4      v_Student_Name Students_.Student_Name%TYPE;
  5      v_Age Students_.Age%TYPE;
  6      v_DOB Students_.DOB%TYPE;
  7      v_Hobby Students_.Hobby%TYPE;
  8  BEGIN
  9      -- Call the stored procedure
 10      GetStudentsByCourse(101, v_Students);
 11
 12      -- Fetch the results
 13      LOOP
 14          FETCH v_Students INTO v_Student_ID, v_Student_Name, v_Age, v_DOB, v_Hobby;
 15          EXIT WHEN v_Students%NOTFOUND;
 16          -- Process each row here
 17          DBMS_OUTPUT.PUT_LINE('Student ID: ' || v_Student_ID || ', Name: ' || v_Student_Name || ', Age: ' || v_Age || ', DOB: ' || v_DOB || ', Hobby: ' || v_Hobby);

 18      END LOOP;
 19
 20      -- Close the cursor
 21      CLOSE v_Students;
 22  END;
 23  /
Student ID: 1, Name: Alice, Age: 20, DOB: 01-JAN-03, Hobby: New Hobby

PL/SQL procedure successfully completed.

SQL>
```

Update Query Based Stored Procedure

```
CREATE OR REPLACE PROCEDURE UpdateStudentHobby (
   p_Student_ID IN NUMBER,
   p_New_Hobby IN VARCHAR2
)
AS
BEGIN
   UPDATE Students_
   SET Hobby = p_New_Hobby
   WHERE Student_ID = p_Student_ID;

   -- Optional: Check if the update was successful
   IF SQL%ROWCOUNT = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'No student found with the given ID');
   END IF;
END UpdateStudentHobby;
/
BEGIN
   -- Call the stored procedure
```

```
    UpdateStudentHobby(1, 'New Hobby');


    -- If needed, you can add additional logic here to handle the outcome
    DBMS_OUTPUT.PUT_LINE('Hobby updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

```
SQL> BEGIN
  2      -- Call the stored procedure
  3      UpdateStudentHobby(1, 'New Hobby');
  4
  5      -- If needed, you can add additional logic here to handle the outcome
  6      DBMS_OUTPUT.PUT_LINE('Hobby updated successfully.');
  7  EXCEPTION
  8      WHEN OTHERS THEN
  9          DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 10  END;
 11  /
Hobby updated successfully.

PL/SQL procedure successfully completed.

SQL>
```