**3. Practice SQL Data Definition Language (DDL) Commands**

**3.1. Table creation and alteration**

**1. Create all 5 tables based on the schema provided.**

**Table Name: Departments**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Dept\_ID | VARCHAR2(4) | PRIMARY KEY, starts with 'D' |
| Dept\_Name | VARCHAR2(50) | NOT NULL, UNIQUE |
| Building | VARCHAR2(30) |  |
| Number\_of\_Classrooms | NUMBER(3) | CHECK(Number\_of\_Classrooms >= 0) |

**CODE:**

CREATE TABLE Departments (

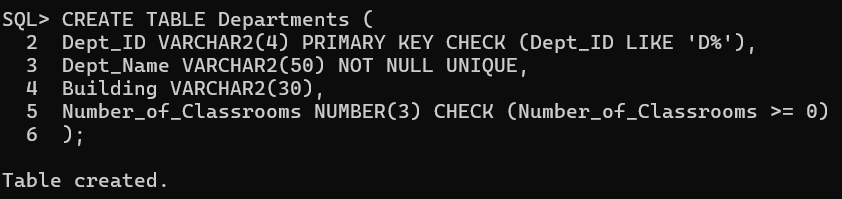
Dept\_ID VARCHAR2(4) PRIMARY KEY CHECK (Dept\_ID LIKE 'D%'),

Dept\_Name VARCHAR2(50) NOT NULL UNIQUE,

Building VARCHAR2(30),

Number\_of\_Classrooms NUMBER(3) CHECK (Number\_of\_Classrooms >= 0) );

**OUTPUT:**



**Table Name: Professors**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Prof\_ID | VARCHAR2(5) | PRIMARY KEY, starts with 'P' |
| Prof\_Name | VARCHAR2(50) | NOT NULL |
| Dept\_ID | VARCHAR2(4) | FOREIGN KEY REFERENCES Departments(Dept\_ID) |
| Experience\_Years | NUMBER(2) | CHECK (Experience\_Years >= 0) |

**CODE:**

CREATE TABLE Professors

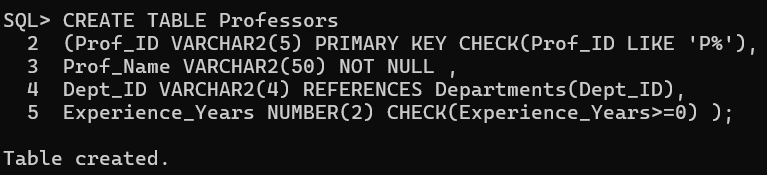
(Prof\_ID VARCHAR2(5) PRIMARY KEY CHECK(Prof\_ID LIKE 'P%'),

Prof\_Name VARCHAR2(50) NOT NULL ,

Dept\_ID VARCHAR2(4) REFERENCES Departments(Dept\_ID),

Experience\_Years NUMBER(2) CHECK(Experience\_Years>=0) );

**OUTPUT:**



**Table Name: Courses**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Course\_ID | VARCHAR2(6) | PRIMARY KEY |
| Course\_Name | VARCHAR2(100) | NOT NULL |
| Dept\_ID | VARCHAR2(4) | FOREIGN KEY REFERENCES Departments(Dept\_ID) |
| Prof\_ID | VARCHAR2(5) | FOREIGN KEY REFERENCES Professors(Prof\_ID) |
| Credits | NUMBER(1) | CHECK (Credits BETWEEN 1 AND 5) |
| Student\_Count | NUMBER(4) | CHECK (Student\_Count >= 0) |

**CODE:**

CREATE TABLE Courses

(Course\_ID VARCHAR2(6) PRIMARY KEY ,

Course\_Name VARCHAR2(100) NOT NULL ,

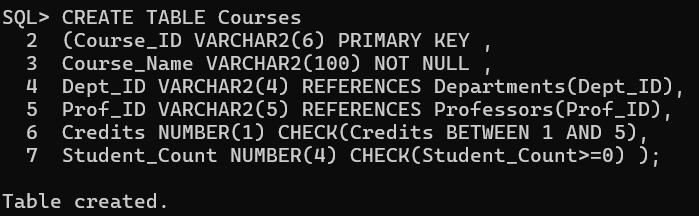
Dept\_ID VARCHAR2(4) REFERENCES Departments(Dept\_ID),

Prof\_ID VARCHAR2(5) REFERENCES Professors(Prof\_ID),

Credits NUMBER(1) CHECK(Credits BETWEEN 1 AND 5),

Student\_Count NUMBER(4) CHECK(Student\_Count>=0) );

**OUTPUT:**



**Table Name: Students**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Student\_ID | VARCHAR2(6) | PRIMARY KEY, starts with 'S' |
| Student\_Name | VARCHAR2(50) | NOT NULL |
| Dept\_ID | VARCHAR2(4) | FOREIGN KEY REFERENCES Departments(Dept\_ID) |
| DOB | DATE |  |

**CODE:**

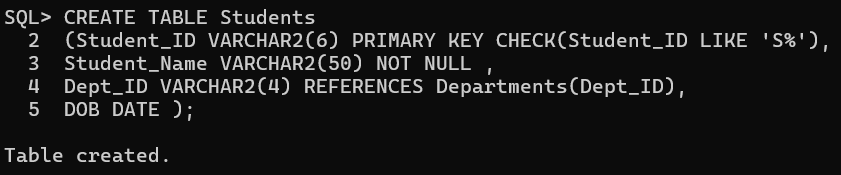
CREATE TABLE Students

(Student\_ID VARCHAR2(6) PRIMARY KEY CHECK(Student\_ID LIKE 'S%'),

Student\_Name VARCHAR2(50) NOT NULL ,

Dept\_ID VARCHAR2(4) REFERENCES Departments(Dept\_ID),

DOB DATE );

**OUTPUT:** 

**Table Name: Enrollments**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Student\_ID | VARCHAR2(6) | FOREIGN KEY REFERENCES Students(Student\_ID) |
| Course\_ID | VARCHAR2(6) | FOREIGN KEY REFERENCES Courses(Course\_ID) |
| Semester | VARCHAR2(6) | e.g., 'Sem1', 'Sem2' |
| Marks | NUMBER(5,2) | CHECK (MARKS >= 0 AND MARKS < 100) |
| PRIMARY KEY | (Student\_ID, Course\_ID) |  |

**CODE:**

CREATE TABLE Enrollments

(Student\_ID VARCHAR2(6) REFERENCES Students(Student\_ID),

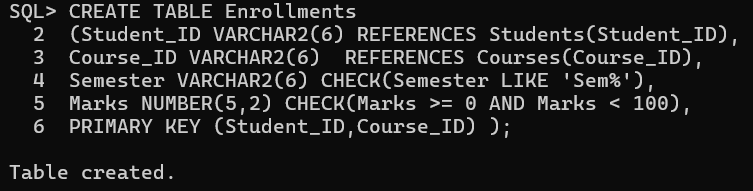
Course\_ID VARCHAR2(6)  REFERENCES Courses(Course\_ID),

Semester VARCHAR2(6) CHECK(Semester LIKE 'Sem%'),

Marks NUMBER(5,2) CHECK(Marks >= 0 AND Marks < 100),

PRIMARY KEY (Student\_ID,Course\_ID) );

**OUTPUT:**

****

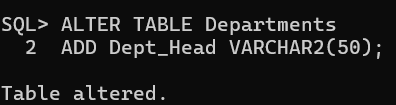
**2. Add a column Dept\_Head (varchar2(50)) to Departments.**

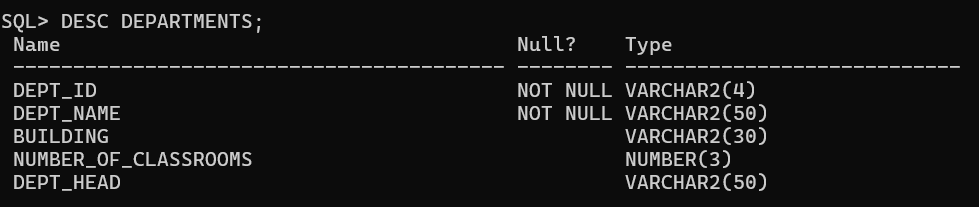
**CODE:**

ALTER TABLE Departments

ADD Dept\_Head VARCHAR2(50);

**OUTPUT:**





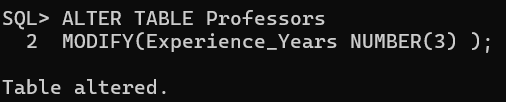
**3. Change size of Experience\_Years in Professors to NUMBER(3).**

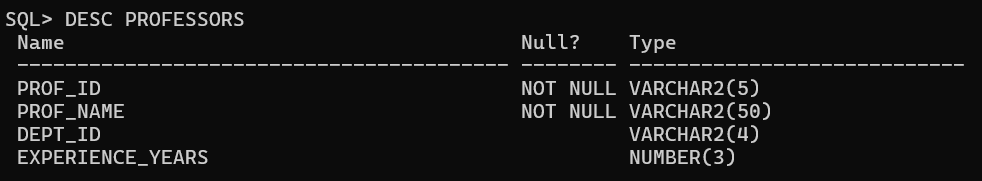
**CODE:**

ALTER TABLE Professors

MODIFY(Experience\_Years NUMBER(3) );

**OUTPUT:**



****

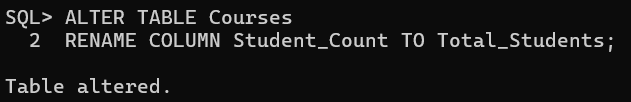
**4. Rename Student\_Count to Total\_Students in Courses.**

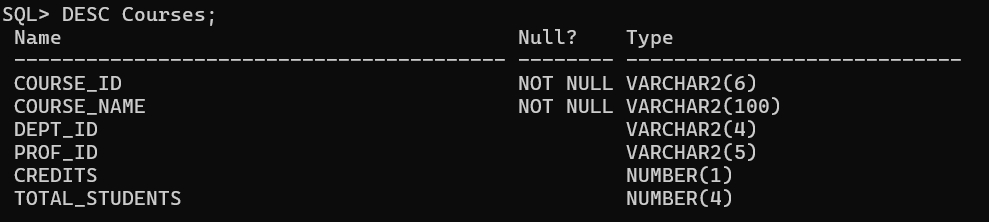
**CODE:**

ALTER TABLE Courses

RENAME COLUMN Student\_Count TO Total\_Students;

**OUTPUT:**

****

****

**5. Drop and recreate the Enrollments table.**

**CODE:**

DROP TABLE Enrollments;

CREATE TABLE Enrollments

(Student\_ID VARCHAR2(6) REFERENCES Students(Student\_ID),

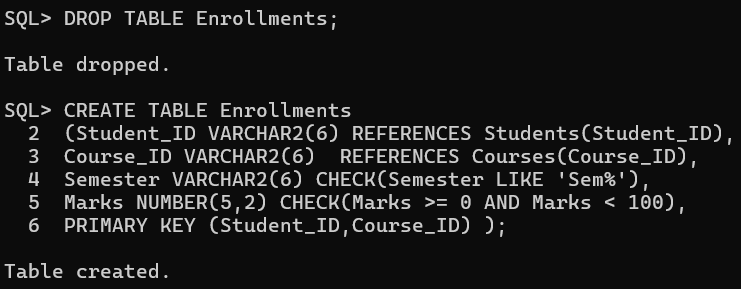
Course\_ID VARCHAR2(6)  REFERENCES Courses(Course\_ID),

Semester VARCHAR2(6) CHECK(Semester LIKE 'Sem%'),

Marks NUMBER(5,2) CHECK(Marks >= 0 AND Marks < 100),

PRIMARY KEY (Student\_ID,Course\_ID) );

**OUTPUT:**

****

**4. Practice SQL Data Manipulation Language (DML) Commands**

**4.1 Insertion, Deletion, Update**

**1. Insert the sample data into all five tables.**

**Departments:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dept\_ID** | **Dept\_Name** | **Building** | **Number\_of\_Classrooms** |
| D01 | Computer Science | Tech Block | 10 |
| D02 | Electrical Engg. | Power House | 8 |
| D03 | Mechanical Engg. | Mech Block | 6 |

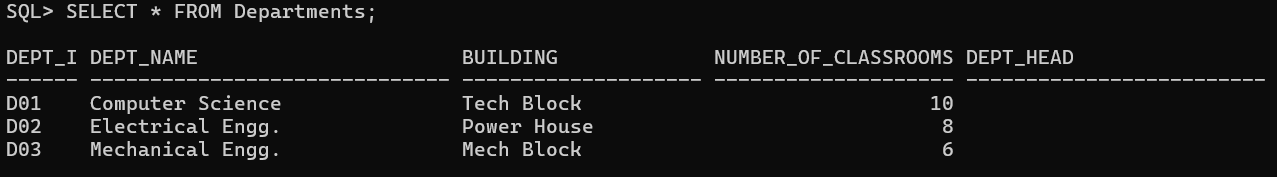
**CODE:**

INSERT INTO Departments VALUES('D01','Computer Science','Tech Block',10,NULL);

INSERT INTO Departments VALUES('D02', 'Electrical Engg.','Power House',8,NULL);

INSERT INTO Departments VALUES('D03', 'Mechanical Engg.','Mech Block',6,NULL);

**OUTPUT:**

****

**Professors:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prof\_ID** | **Prof\_Name** | **Dept\_ID** | **Experience\_Years** |
| P1001 | Dr. Meera Nair | D01 | 12 |
| P1002 | Dr. Arjun Rao | D02 | 9 |
| P1003 | Dr. Kavita Singh | D01 | 7 |
| P1004 | Dr. Raj Malhotra | D03 | 15 |

**CODE:**

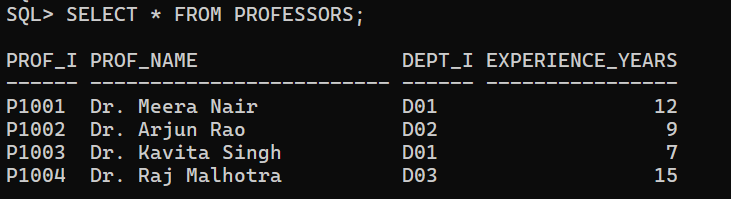
INSERT INTO Professors VALUES('P1001', 'Dr. Meera Nair','D01',12);

INSERT INTO Professors VALUES('P1002', 'Dr. Arjun Rao','D02',9);

INSERT INTO Professors VALUES('P1003', 'Dr. Kavita Singh','D01,7);

INSERT INTO Professors VALUES('P1004', 'Dr. Raj Malhotra','D03',15);

**OUTPUT:**

****

**Courses:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course\_ID** | **Course\_Name** | **Dept\_ID** | **Prof\_ID** | **Credits** | **Student\_Count** |
| CSE101 | Data Structures | D01 | P1001 | 4 | 2 |
| CSE201 | Operating Systems | D01 | P1003 | 3 | 1 |
| EEE101 | Circuit Theory | D02 | P1002 | 4 | 1 |
| ME101 | Thermodynamics | D03 | P1004 | 3 | 1 |

**CODE:**

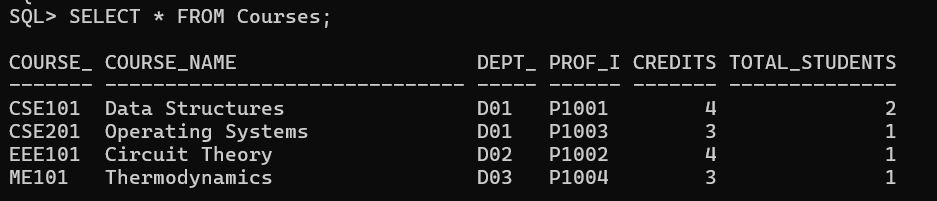
INSERT INTO Courses VALUES('CSE101','Data Structures','D01', 'P1001',4,2);

INSERT INTO Courses VALUES('CSE201','Operating Systems','D01', 'P1003',3,1);

INSERT INTO Courses VALUES('EEE101','Circuit Theory','D02', 'P1002',4,1);

INSERT INTO Courses VALUES('ME101','Thermodynamics','D03', 'P1004',3,1);

**OUTPUT:**

****

**Students:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student\_ID** | **Student\_Name** | **Dept\_ID** | **DOB** |
| S0001 | Anjali Sharma | D01 | 2003-05-14 |
| S0002 | Ravi Kumar | D02 | 2002-11-20 |
| S0003 | Nisha Verma | D03 | 2003-02-02 |
| S0004 | Aman Sheikh | D01 | 2002-07-25 |

**CODE:**

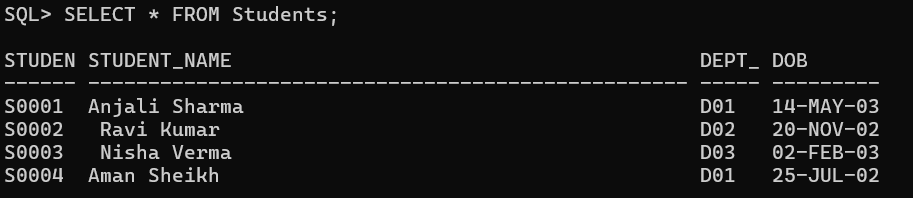
INSERT INTO Students VALUES('S0001','Anjali Sharma','D01',TO\_DATE('2003-05-14', 'YYYY-MM-DD'));

INSERT INTO Students VALUES('S0002',' Ravi Kumar','D02',TO\_DATE('2002-11-20', 'YYYY-MM-DD'));

INSERT INTO Students VALUES('S0003',' Nisha Verma','D03',TO\_DATE('2003-02-02', 'YYYY-MM-DD'));

INSERT INTO Students VALUES('S0004','Aman Sheikh','D01',TO\_DATE('2002-07-25', 'YYYY-MM-DD'));

**OUTPUT:**

****

**Enrollments:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student\_ID** | **Course\_ID** | **Semester** | **Marks** |
| S0001 | CSE101 | Sem1 | 88.0 |
| S0001 | CSE201 | Sem2 | 76.5 |
| S0002 | EEE101 | Sem1 | 81.0 |
| S0003 | ME101 | Sem1 | 93.0 |
| S0004 | CSE101 | Sem1 | 68.5 |

**CODE:**

INSERT INTO Enrollments VALUES('S0001','CSE101','Sem1',88.0);

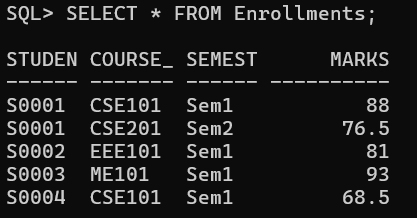
INSERT INTO Enrollments VALUES('S0001','CSE201','Sem2',76.5);

INSERT INTO Enrollments VALUES('S0002','EEE101','Sem1',81.0);

INSERT INTO Enrollments VALUES('S0003','ME101','Sem1',93.0);

INSERT INTO Enrollments VALUES('S0004','CSE101','Sem1',68.5);

**OUTPUT:**

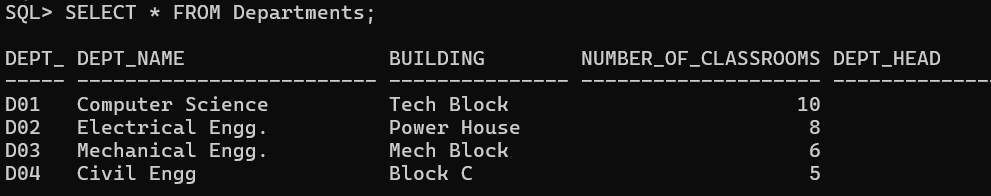
****

**2. Insert a new department: (‘D04’, ‘Civil Engg’, ‘Block C’, 5).**

**CODE:**

INSERT INTO Departments VALUES ('D04', 'Civil Engg', 'Block C', 5,NULL);

**OUTPUT:**

****

**3. Create a new table high\_achievers containing students who scored more than 85 in any course.**

**CODE:**

CREATE TABLE high\_achievers AS

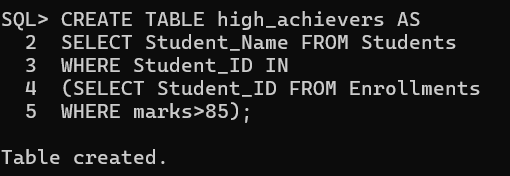
SELECT Student\_Name FROM Students

WHERE Student\_ID IN

(SELECT Student\_ID FROM Enrollments

WHERE marks>85);

**OUTPUT:**

****

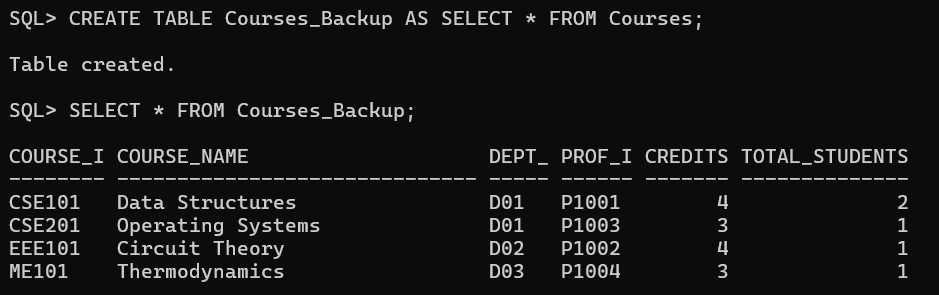
**4. Create a backup table Courses\_Backup with all data from Courses and Professors\_Backup with all the data from professors.**

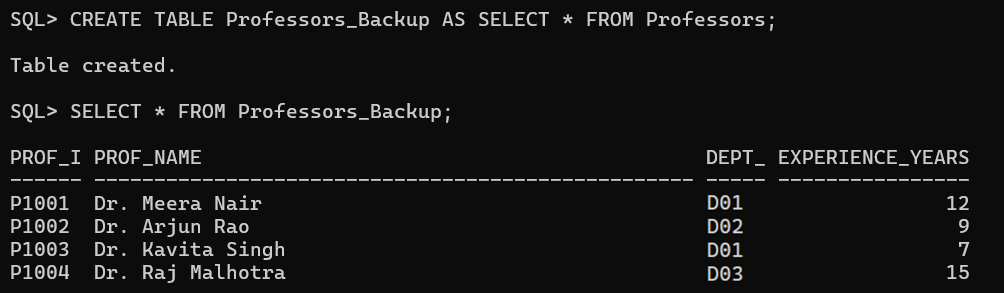
**CODE:**

CREATE TABLE Courses\_Backup AS SELECT \* FROM Courses;

CREATE TABLE Professors\_Backup AS SELECT \* FROM Professors;

**OUTPUT:**

****

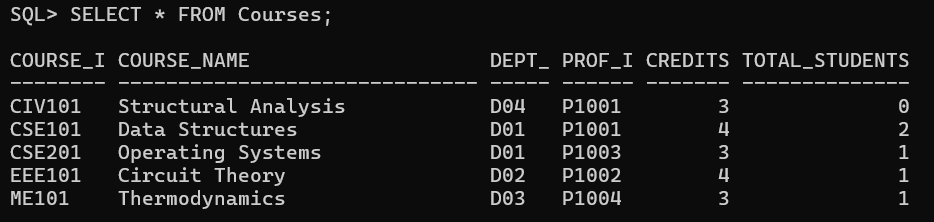


**5. Add a new course ‘CIV101’, ‘Structural Analysis’, under D04, taught by P1001, with 3 credits and 0 students.**

**CODE:**

INSERT INTO Courses VALUES('CIV101','Structural Analysis','D04', 'P1001',3,0);

**OUTPUT:**

****

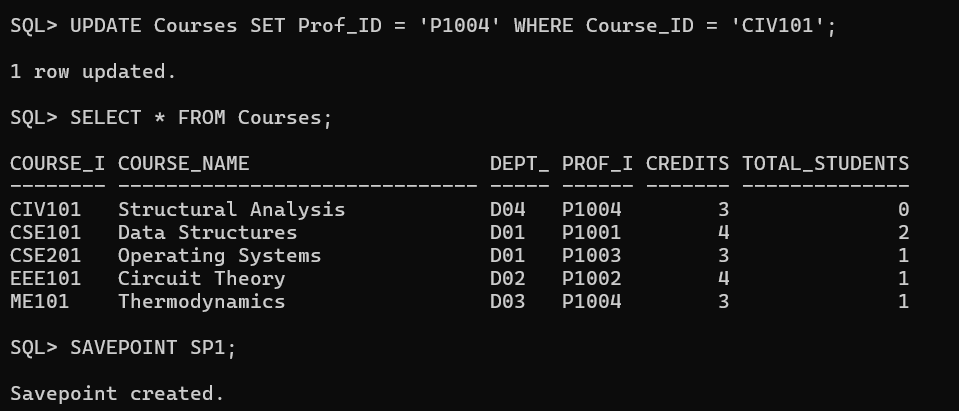
**6. Update professor of ‘CIV101’ to P1004 and savepoint SP1.**

**CODE:**

UPDATE Courses SET Prof\_ID = 'P1004' WHERE Course\_ID = 'CIV101';

SAVEPOINT SP1;

**OUTPUT:**

****

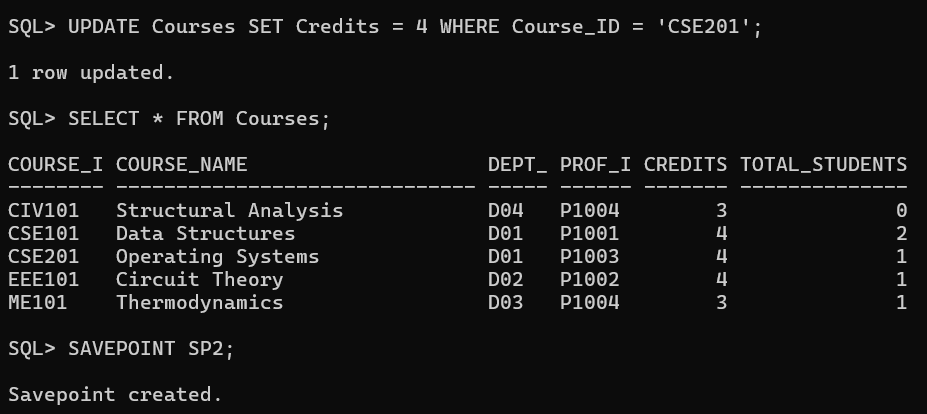
**7. Change credits of ‘CSE201’ to 4 and set savepoint SP2.**

**CODE:**

UPDATE Courses SET Credits = 4 WHERE Course\_ID = 'CSE201';

SAVEPOINT SP2;

**OUTPUT:**

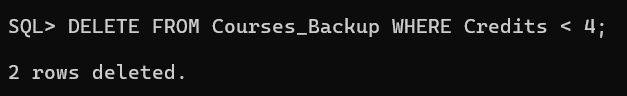
****

**8. Delete all courses from Courses\_Backup that have less than 4 credits.**

**CODE:**

DELETE FROM Courses\_Backup WHERE Credits < 4;

**OUTPUT:**

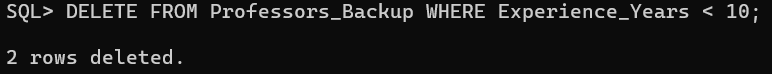
****

**9. Delete all professors from Professors\_Backup with less than 10 years experience.**

**CODE:**

DELETE FROM Professors\_Backup WHERE Experience\_Years < 10;

**OUTPUT:**

****

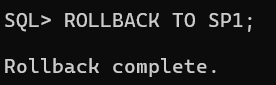
**10. Rollback to SP1 and rename Courses\_Backup to Course\_Master.**

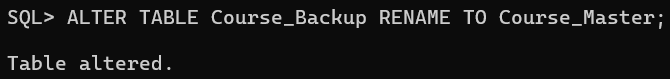
**CODE:**

ROLLBACK TO SP1;

ALTER TABLE Course\_Backup RENAME TO Course\_Master;

**OUTPUT:**

****

****

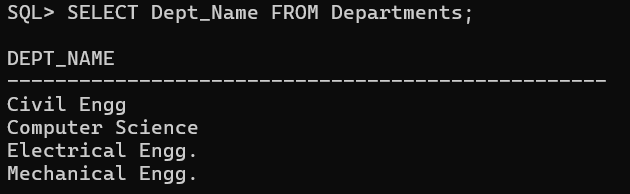
**4.2 Data Retrieval (SELECT)**

**1. List all department names.**

**CODE:**

SELECT Dept\_Name FROM Departments;

**OUTPUT:**

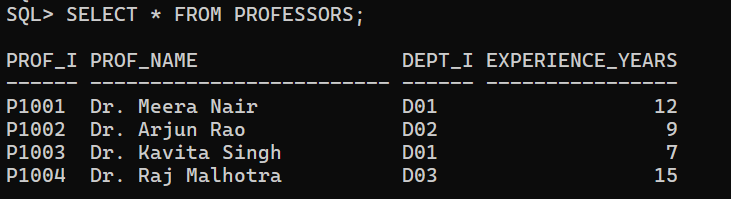
****

**2. Display all data from the Professors table.**

**CODE:**

SELECT \* FROM Professors;

**OUTPUT:**

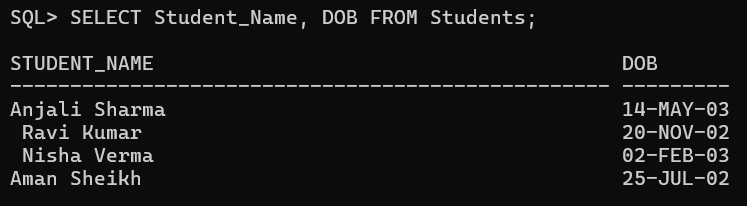
****

**3. List student names and DOBs.**

**CODE:**

SELECT Student\_Name, DOB FROM Students;

**OUTPUT:**

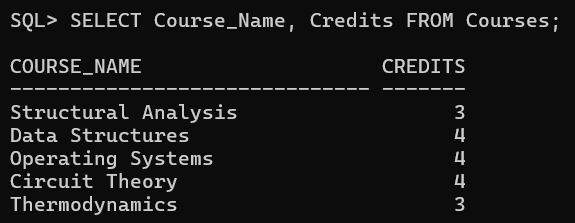
****

**4. List course names and credits.**

**CODE:**

SELECT Course\_Name, Credits FROM Courses;

**OUTPUT:**

****

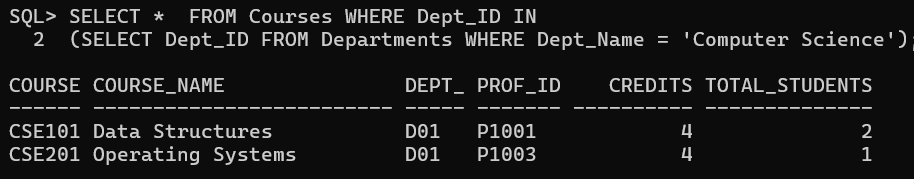
**5. Get courses offered by the ‘Computer Science’ department.**

**CODE:**

SELECT \* FROM Courses WHERE Dept\_ID IN

(SELECT Dept\_ID FROM Departments WHERE Dept\_Name = 'Computer Science');

**OUTPUT:**

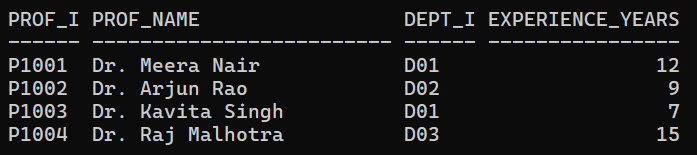
****

**6. List professors whose name starts with ‘Dr’.**

**CODE:**

SELECT \* FROM Professors WHERE Prof\_Name LIKE 'Dr%';

**OUTPUT:**

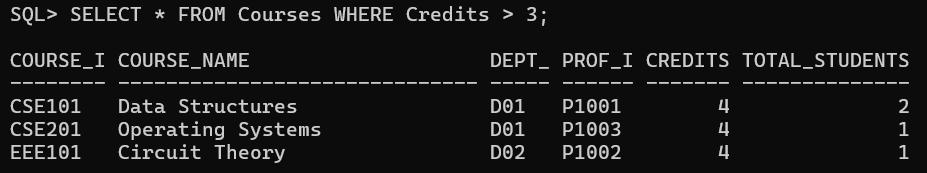
****

**7. List courses with credits more than 3.**

**CODE:**

SELECT \* FROM Courses WHERE Credits > 3;

**OUTPUT:**

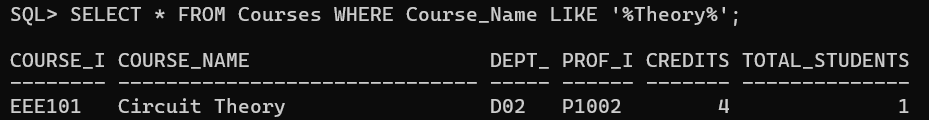
****

**8. Display all courses with “Theory” in their name.**

**CODE:**

SELECT \* FROM Courses WHERE Course\_Name LIKE '%Theory%';

**OUTPUT:**

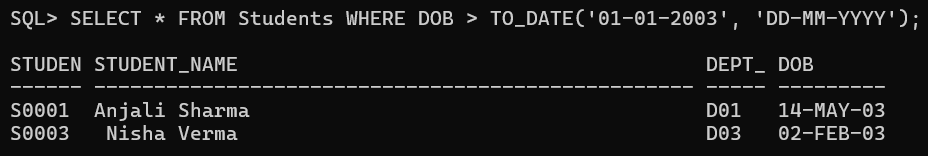
****

**9. List students born after Jan 1, 2003.**

**CODE:**

SELECT \* FROM Students WHERE DOB > TO\_DATE('01-01-2003', 'DD-MM-YYYY');

**OUTPUT:**

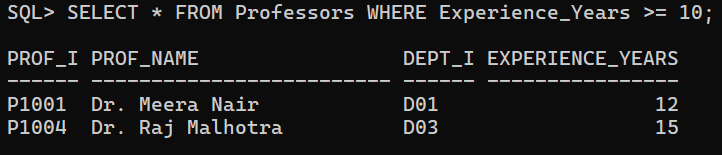
****

**10. Find all professors with 10+ years experience.**

**CODE:**

SELECT \* FROM Professors WHERE Experience\_Years >= 10;

**OUTPUT:**

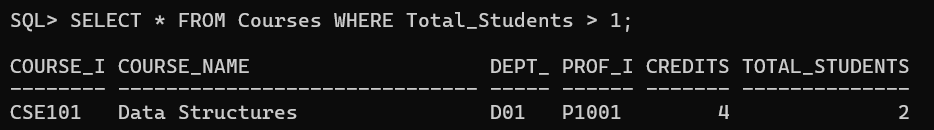


**11. List all courses with more than 1 student.**

**CODE:**

SELECT \* FROM Courses WHERE Total\_Students > 1;

**OUTPUT:**

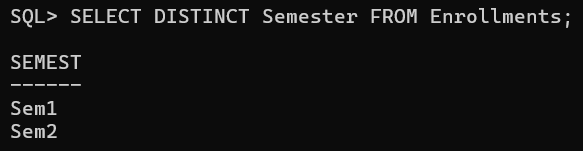
****

**12. Display all distinct semesters from Enrollments.**

**CODE:**

SELECT DISTINCT Semester FROM Enrollments;

**OUTPUT:**

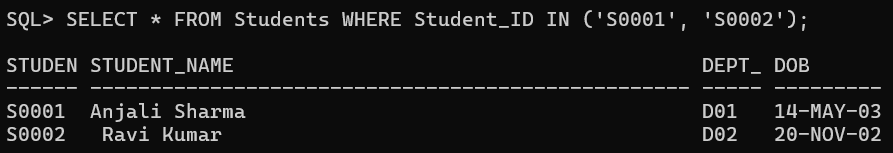
****

**13. Show information of students with ID S0001, S0002.**

**CODE:**

SELECT \* FROM Students WHERE Student\_ID IN ('S0001', 'S0002');

**OUTPUT:**

****

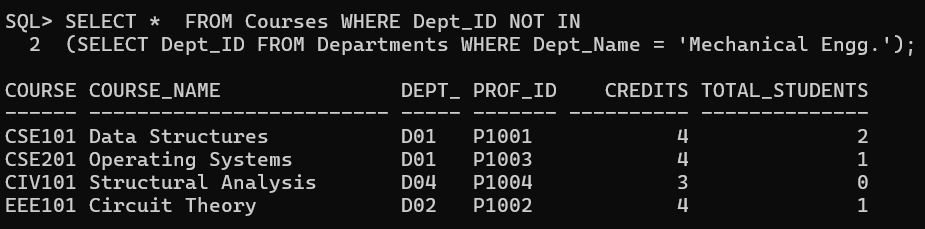
**14. Show all courses not in the ‘Mechanical Engg’ department.**

**CODE:**

SELECT \* FROM Courses WHERE Dept\_ID NOT IN

(SELECT Dept\_ID FROM Departments WHERE Dept\_Name = 'Mechanical Engg.');

**OUTPUT:**



**4.3 SQL Functions**

**1.Use numeric functions like ROUND, MOD, POWER on dummy values.**

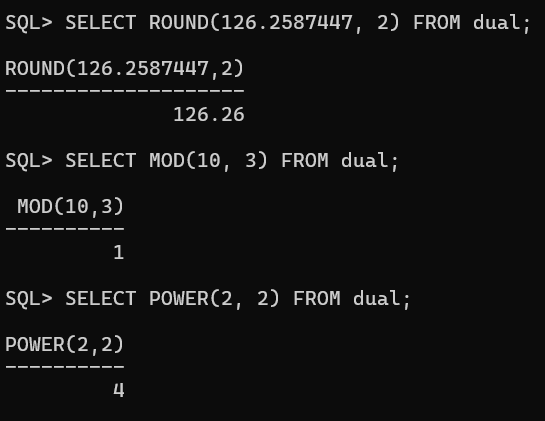
**CODE:**

SELECT ROUND(126.2587447, 2) FROM dual;

SELECT MOD(10, 3) FROM dual;

SELECT POWER(2, 2) FROM dual;

**OUTPUT:**

****

**2. Use string functions like LENGTH, SUBSTR, INSTR, UPPER, LOWER on names in Professors and Students.**

**CODE:**

**For Professors:**

SELECT Prof\_Name,

LENGTH(Prof\_Name) AS Length,

SUBSTR(Prof\_Name, 1, 5) AS Substr,

INSTR(Prof\_Name, 'a') AS Pos,

UPPER(Prof\_Name) AS Uppercase,

LOWER(Prof\_Name) AS Lowercase

FROM Professors;

**For Students:**

SELECT Student\_Name,

LENGTH(Student\_Name) AS Length,

SUBSTR(Student\_Name, 1, 5) AS Substr,

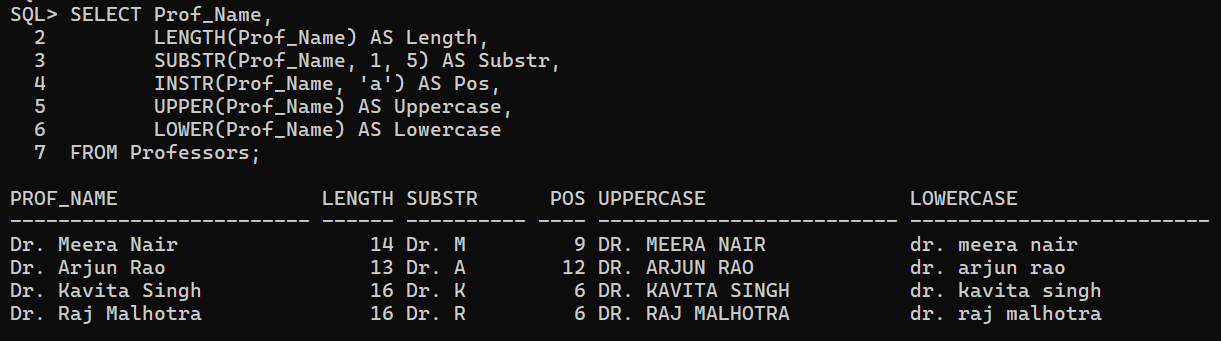
INSTR(Student\_Name, 'a') AS Pos,

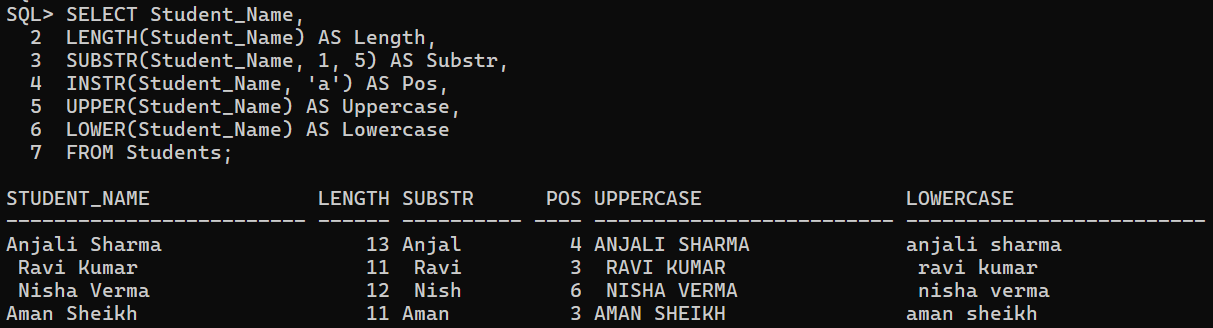
UPPER(Student\_Name) AS Uppercase,

LOWER(Student\_Name) AS Lowercase

FROM Students;

**OUTPUT:**





**3. Use conversion functions on DOB.**

**CODE:**

SELECT Student\_ID, Student\_Name, DOB,

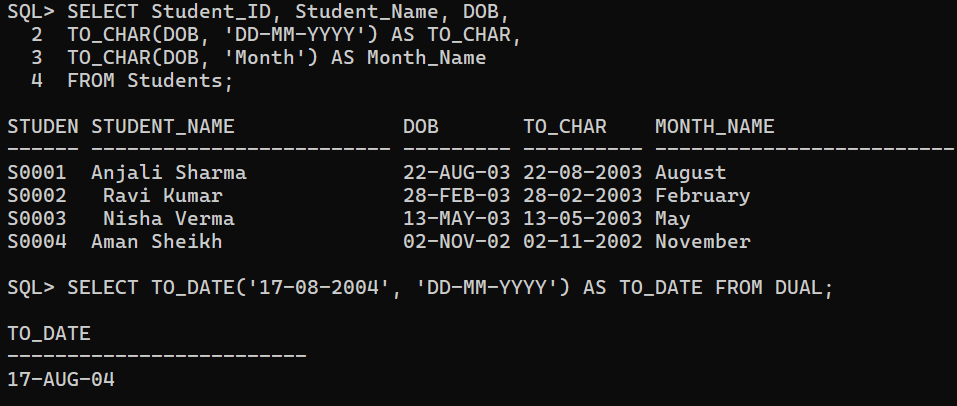
TO\_CHAR(DOB, 'DD-MM-YYYY') AS TO\_CHAR,

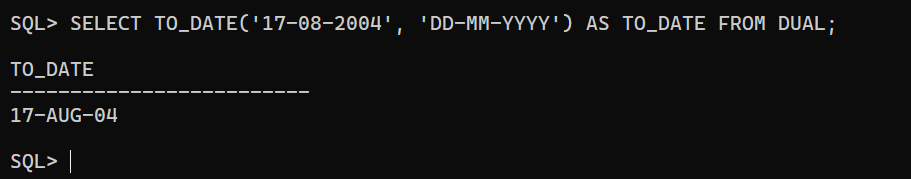
TO\_CHAR(DOB, 'Month') AS Month\_Name

FROM Students;

SELECT TO\_DATE('17-08-2004', 'DD-MM-YYYY') AS TO\_DATE FROM DUAL;

**OUTPUT:**



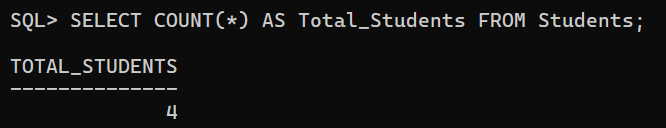
****

**4. Count total number of students.**

**CODE:**

SELECT COUNT(\*) AS Total\_Students FROM Students;

**OUTPUT:**



**5. Find max and min marks in Enrollments as max\_marks, min\_marks.**

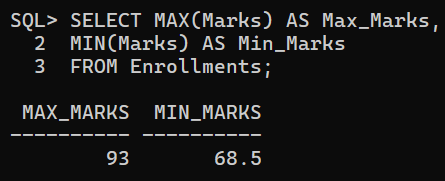
**CODE:**

SELECT MAX(Marks) AS Max\_Marks,

MIN(Marks) AS Min\_Marks

FROM Enrollments;

**OUTPUT:**

****

**6. Count number of students with marks over 75.**

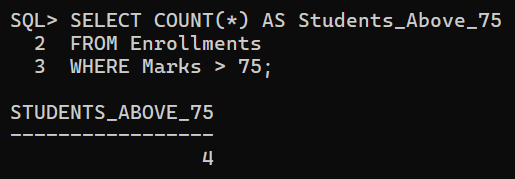
**CODE:**

SELECT COUNT(\*) AS Students\_Above\_75

FROM Enrollments

WHERE Marks > 75;

**OUTPUT:**



**4.4 Date Functions**

**1. List student names and their day of birth.**

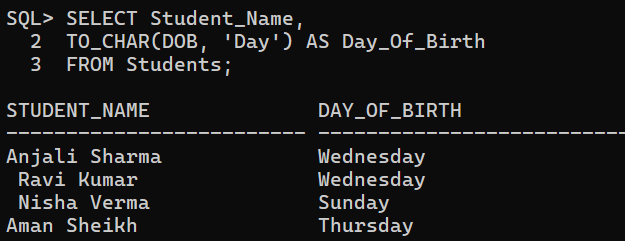
**CODE:**

SELECT Student\_Name,

TO\_CHAR(DOB, 'Day') AS Day\_Of\_Birth

FROM Students;

**OUTPUT:**

****

**2. Format DOBs in ‘DD-Month-YYYY’ format.**

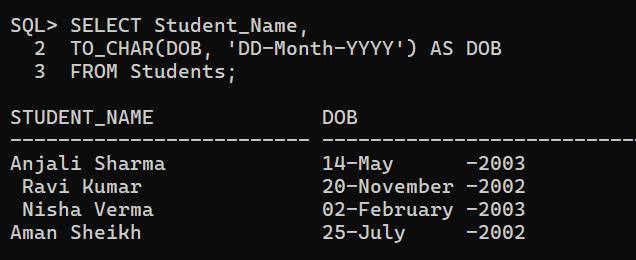
**CODE:**

SELECT Student\_Name,

TO\_CHAR(DOB, 'DD-Month-YYYY') AS DOB

FROM Students;

**OUTPUT:**

****

**3. Show DOBs in ‘DD-MM-YY’ format.**

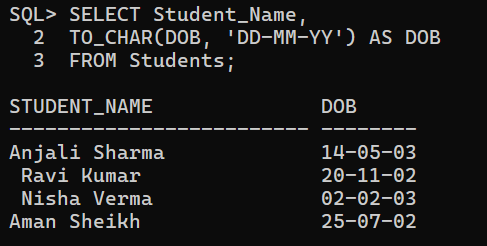
**CODE:**

SELECT Student\_Name,

TO\_CHAR(DOB, 'DD-MM-YY') AS DOB

FROM Students;

**OUTPUT:**

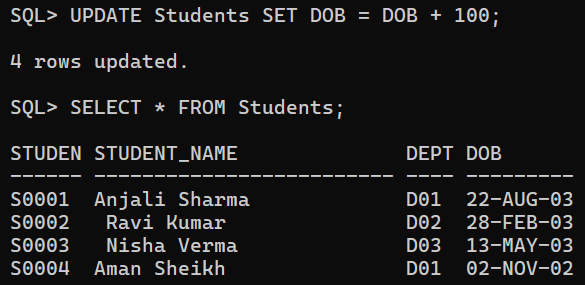
****

**4. Add 100 days to all DOBs.**

**CODE:**

UPDATE Students SET DOB = DOB + 100;

**OUTPUT:**

****

**5. List students born in May.**

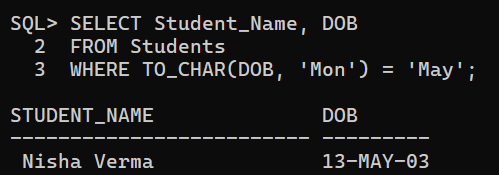
**CODE:**

SELECT Student\_Name, DOB

FROM Students

WHERE TO\_CHAR(DOB, 'Mon') = 'May';

**OUTPUT:**

****

**6. List students born between 2002 and 2003.**

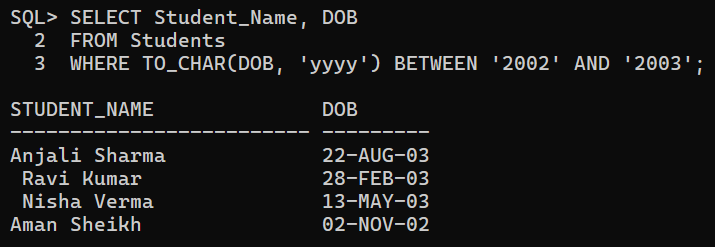
**CODE:**

SELECT Student\_Name, DOB

FROM Students

WHERE TO\_CHAR(DOB, 'yyyy') BETWEEN '2002' AND '2003';

**OUTPUT:**

****

**4.5 Set Operators**

**Create Top\_Courses table:**

CREATE TABLE Top\_Courses (

Course\_Name VARCHAR2(100),

Dept\_Name VARCHAR2(50)

);

**Insert some course-department pairs.**

INSERT INTO Top\_Courses VALUES ('Data Structures', 'Computer Science');

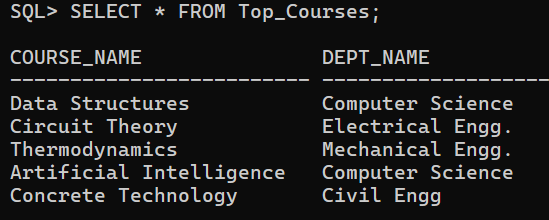
INSERT INTO Top\_Courses VALUES ('Circuit Theory', 'Electrical Engg.');

INSERT INTO Top\_Courses VALUES ('Thermodynamics', 'Mechanical Engg.');

INSERT INTO Top\_Courses VALUES ('Artificial Intelligence', 'Computer Science');

INSERT INTO Top\_Courses VALUES (' Concrete Technology', 'Civil Engg');

**OUTPUT:**



**1. Show unique course names from both Courses and Top\_Courses.**

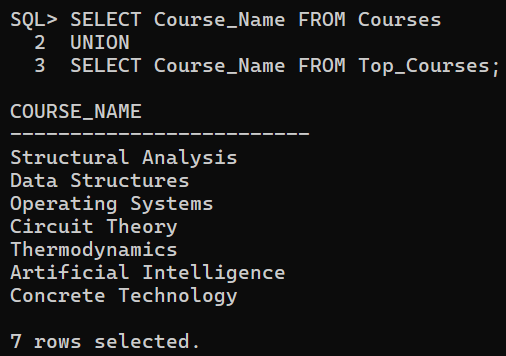
**CODE:**

SELECT Course\_Name FROM Courses

UNION

SELECT Course\_Name FROM Top\_Courses;

**OUTPUT**

****

**2. Show common courses between Courses and Top\_Courses.**

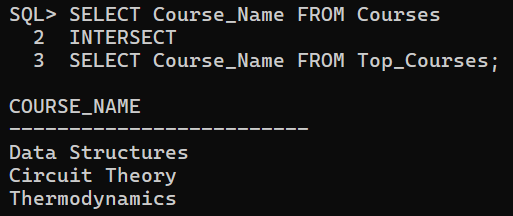
**CODE:**

SELECT Course\_Name FROM Courses

INTERSECT

SELECT Course\_Name FROM Top\_Courses;

**OUTPUT**

****

**3. Show top courses that are not in current courses.**

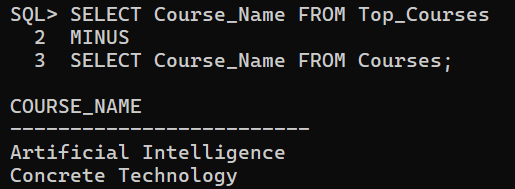
**CODE:**

SELECT Course\_Name FROM Top\_Courses

MINUS

SELECT Course\_Name FROM Courses;

**OUTPUT**

****

**4.Show union all of both.**

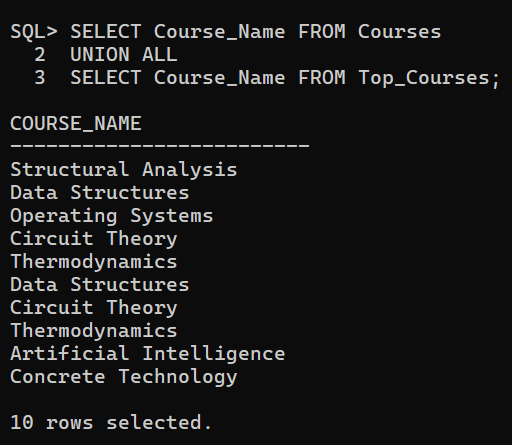
**CODE:**

SELECT Course\_Name FROM Courses

UNION ALL

SELECT Course\_Name FROM Top\_Courses;

**OUTPUT:**



**4.6 Sorting**

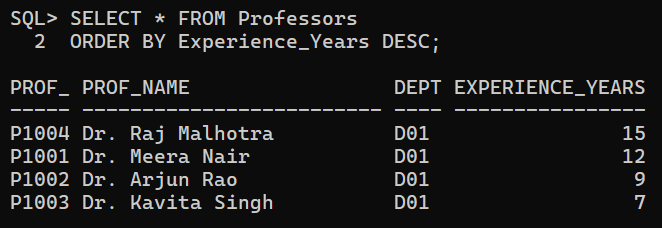
**1. Sort professors by descending experience.**

**CODE:**

SELECT \* FROM Professors

ORDER BY Experience\_Years DESC;

**OUTPUT:**



**4.7 Group By, Having**

**1. Number of students per department (only those >1).**

**CODE:**

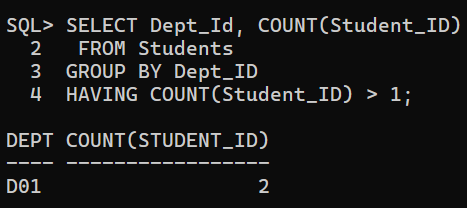
SELECT Dept\_Id, COUNT(Student\_ID)

FROM Students

GROUP BY Dept\_ID

HAVING COUNT(Student\_ID) > 1;

**OUTPUT:**



**2. Departments with avg classroom count >5.**

**CODE:**

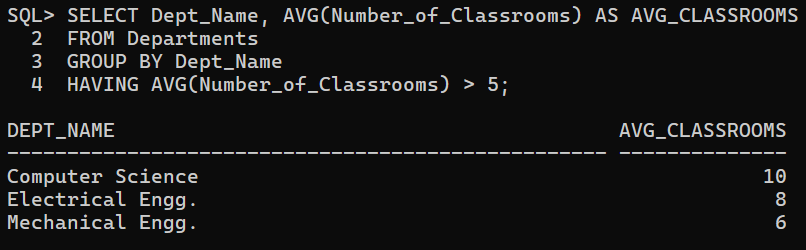
SELECT Dept\_Name, AVG(Number\_of\_Classrooms) AS AVG\_CLASSROOMS

FROM Departments

GROUP BY Dept\_Name

HAVING AVG(Number\_of\_Classrooms) > 5;

**OUTPUT:**



**3. Courses taught by professors with more than 1 course.**

**CODE:**

SELECT c.Course\_Name, p.Prof\_Name

FROM Courses c, Professors p

WHERE c.Prof\_ID = p.Prof\_ID

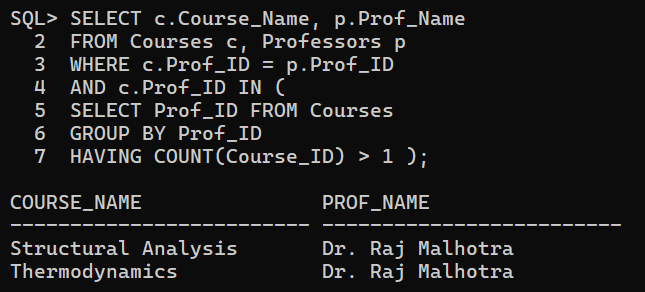
AND c.Prof\_ID IN (

SELECT Prof\_ID FROM Courses

GROUP BY Prof\_ID

HAVING COUNT(Course\_ID) > 1 );

**OUTPUT:**



**4.8 Subqueries**

**1. Find names of students enrolled in any course.**

**CODE:**

SELECT Student\_Name

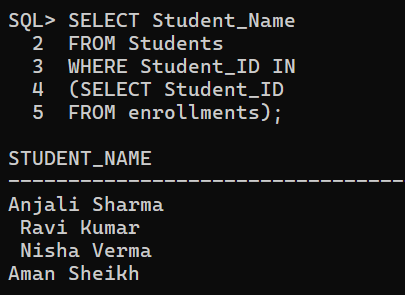
FROM Students

WHERE Student\_ID IN

(SELECT Student\_ID

FROM enrollments);

**OUTPUT:**

****

**2. Find department of student S0001.**

**CODE:**

SELECT Dept\_ID, Dept\_Name

FROM Departments

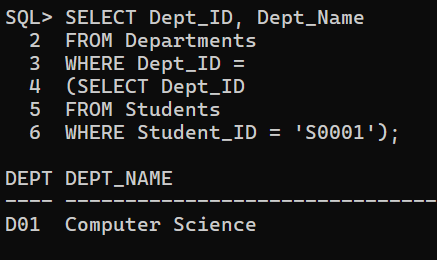
WHERE Dept\_ID =

(SELECT Dept\_ID

FROM Students

WHERE Student\_ID = 'S0001');

**OUTPUT:**



**3. Departments where student count > 1.**

**CODE:**

SELECT Dept\_ID, Dept\_Name

FROM Departments

WHERE Dept\_ID IN

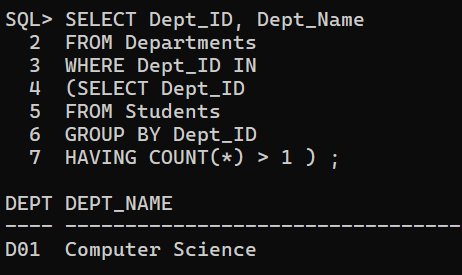
(SELECT Dept\_ID

FROM Students

GROUP BY Dept\_ID

HAVING COUNT(\*) > 1 ) ;

**OUTPUT:**

****

**4. List courses taught by professors with >10 years experience.**

**CODE:**

SELECT Course\_ID, Course\_Name

FROM Courses

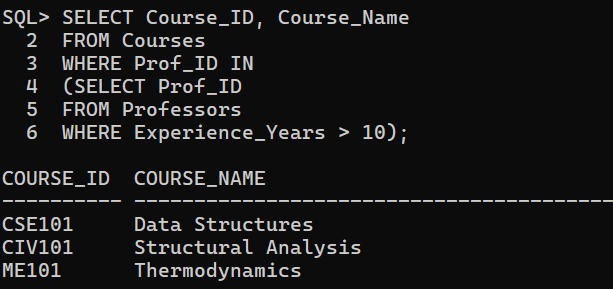
WHERE Prof\_ID IN

(SELECT Prof\_ID

FROM Professors

WHERE Experience\_Years > 10);

**OUTPUT:**

****

**5. List students in departments with at least 1 professor.**

**CODE:**

SELECT Student\_Name

FROM Students

WHERE Dept\_ID IN

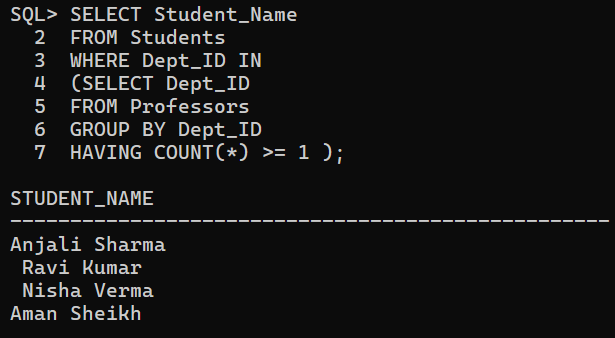
(SELECT Dept\_ID

FROM Professors

GROUP BY Dept\_ID

HAVING COUNT(\*) >= 1 );

**OUTPUT:**



**6. Find the professors with exp less than the average experience of all the professors.**

**CODE:**

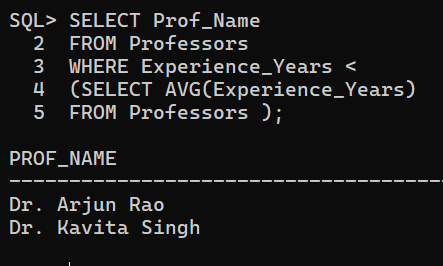
SELECT Prof\_Name

FROM Professors

WHERE Experience\_Years <

(SELECT AVG(Experience\_Years)

FROM Professors );

**OUTPUT:**

**4.9 NOT EXISTS**

**1. Departments where every professor has > 5 years experience.**

**CODE:**

SELECT D.Dept\_Name, D.Dept\_ID

FROM Departments D

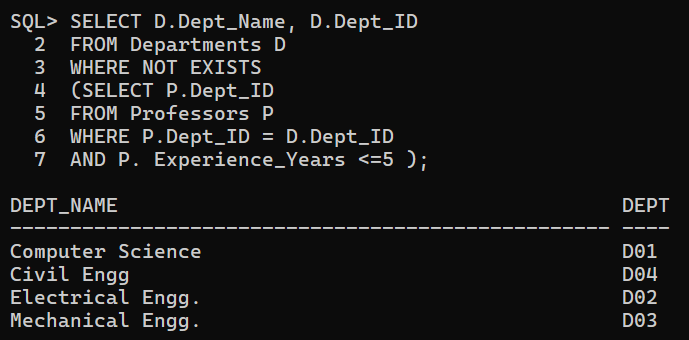
WHERE NOT EXISTS

(SELECT P.Dept\_ID

FROM Professors P

WHERE P.Dept\_ID = D.Dept\_ID

AND P. Experience\_Years <=5 );

**OUTPUT:  
**

**2. Departments with no students.**

**CODE:**

SELECT D.Dept\_Name, D.Dept\_ID

FROM Departments D

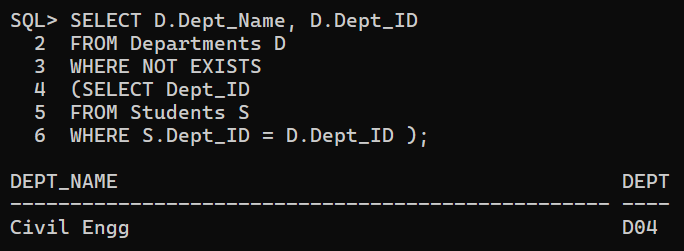
WHERE NOT EXISTS

(SELECT Dept\_ID

FROM Students S

WHERE S.Dept\_ID = D.Dept\_ID );

**OUTPUT:**

****

**4.10 Correlated Subqueries**

**1.** **Students with marks above average.**

**CODE:**

SELECT S.Student\_Name, E.Course\_ID, E.Marks

FROM Students S, Enrollments E

WHERE S.Student\_ID = E.Student\_ID

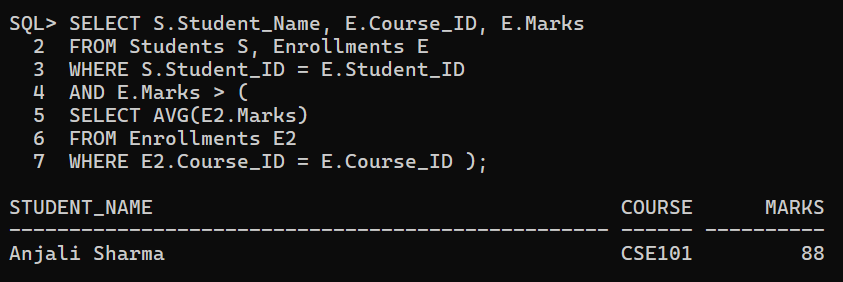
AND E.Marks > (

SELECT AVG(E2.Marks)

FROM Enrollments E2

WHERE E2.Course\_ID = E.Course\_ID );

**OUTPUT:**

****

**2. Students with DOB later than any student from 'Electrical Engg'.**

**CODE:**

SELECT S.Student\_Name, S.DOB

FROM Students S

WHERE EXISTS (

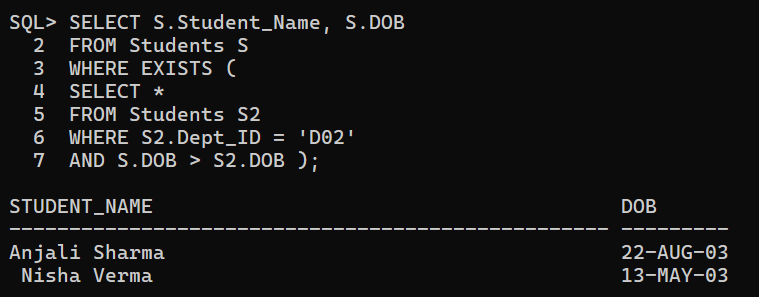
SELECT \*

FROM Students S2

WHERE S2.Dept\_ID = 'D02'

AND S.DOB > S2.DOB );

**OUTPUT:**

****

**4.11 JOINING Tables**

**1. List student names along with their department name.**

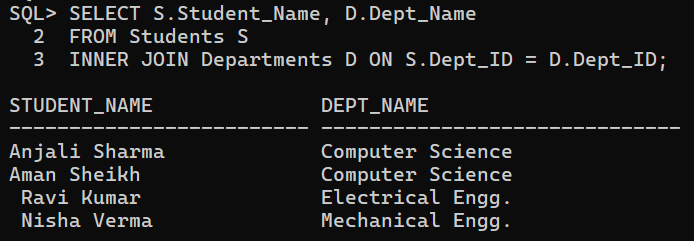
**CODE:**

SELECT S.Student\_Name, D.Dept\_Name

FROM Students S

INNER JOIN Departments D ON S.Dept\_ID = D.Dept\_ID;

**OUTPUT:**



**2. Courses and number of students enrolled.**

**CODE:**

SELECT C.Course\_ID, C.Course\_Name, COUNT(E. Student\_ID) AS COUNT

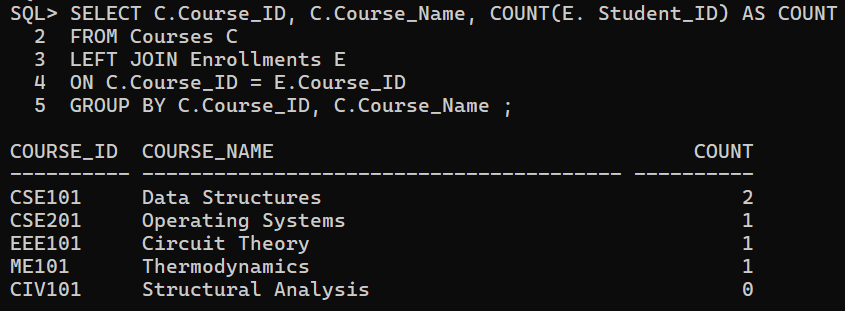
FROM Courses C

LEFT JOIN Enrollments E

ON C.Course\_ID = E.Course\_ID

GROUP BY C.Course\_ID, C.Course\_Name ;

**OUTPUT:**

****

**3. List course name and professor name.**

**CODE:**

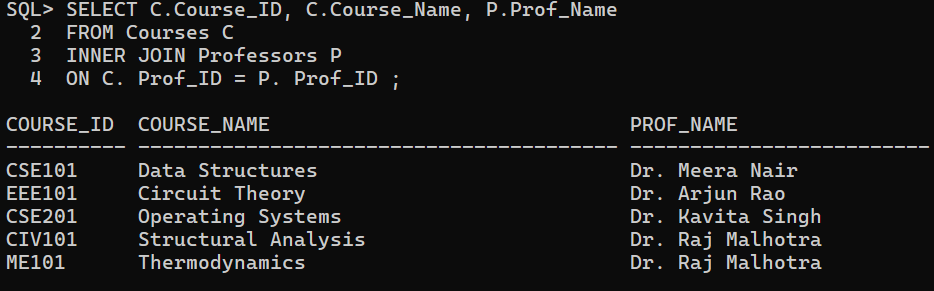
SELECT C.Course\_ID, C.Course\_Name, P.Prof\_Name

FROM Courses C

INNER JOIN Professors P

ON C. Prof\_ID = P. Prof\_ID ;

**OUTPUT:**

****

**4. Departments with more than one professor.**

**CODE:**

SELECT D.Dept\_Name, COUNT(P.Prof\_ID) AS Total\_Professors

FROM Departments D

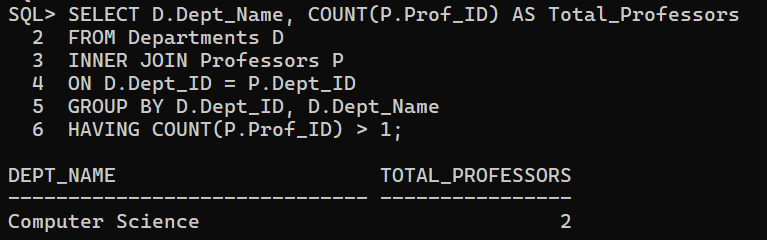
INNER JOIN Professors P

ON D.Dept\_ID = P.Dept\_ID

GROUP BY D.Dept\_ID, D.Dept\_Name

HAVING COUNT(P.Prof\_ID) > 1;

**OUTPUT:**

****

**5. List students enrolled in courses with more than 3 credits.**

**CODE:**

SELECT S.Student\_ID, S.Student\_Name, C.Course\_ID, C.Credits

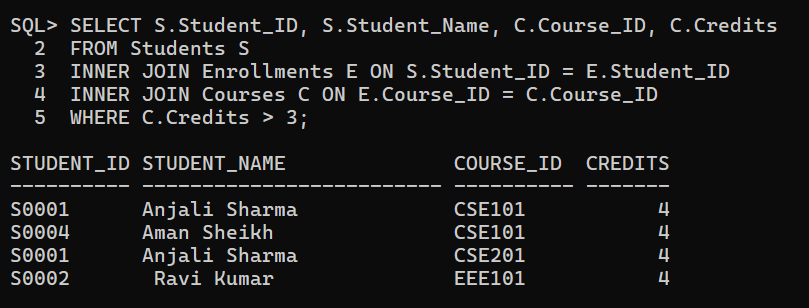
FROM Students S

INNER JOIN Enrollments E ON S.Student\_ID = E.Student\_ID

INNER JOIN Courses C ON E.Course\_ID = C.Course\_ID

WHERE C.Credits > 3;

**OUTPUT:**

****

**5. PL/SQL Section**

**5.1 Basic Blocks**

**1. Accept a course ID and show number of students enrolled.**

**CODE:**

SET SERVEROUTPUT ON;

DECLARE

v\_course\_id Courses.Course\_ID%TYPE;

v\_student\_count NUMBER;

BEGIN

v\_course\_id := '&course\_id';

SELECT COUNT(\*)

INTO v\_student\_count

FROM Enrollments

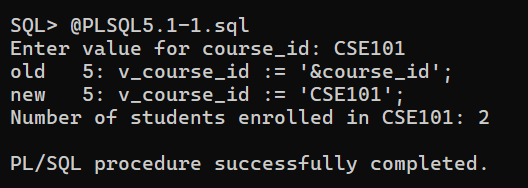
WHERE Course\_ID = v\_course\_id;

DBMS\_OUTPUT.PUT\_LINE('Number of students enrolled in ' || v\_course\_id || ': ' || v\_student\_count);

END;

/

**OUTPUT:**



**2. If department 'Biotech' doesn't exist, insert it with ‘D05’.**

**CODE:**

SET SERVEROUTPUT ON;

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM Departments

WHERE Dept\_Name = 'Biotech';

IF v\_count = 0 THEN

INSERT INTO Departments (Dept\_ID, Dept\_Name, Building, Number\_of\_Classrooms)

VALUES ('D05', 'Biotech', 'Bio Block', 5);

DBMS\_OUTPUT.PUT\_LINE('Department Biotech inserted.');

ELSE

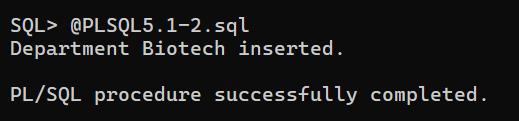
DBMS\_OUTPUT.PUT\_LINE('Department Biotech already exists.');

END IF;

END;

/

**OUTPUT:**



**3. Calculate total students in ‘Computer Science’.**

**CODE:**

SET SERVEROUTPUT ON;

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM Departments

WHERE Dept\_Name = 'Biotech';

IF v\_count = 0 THEN

INSERT INTO Departments (Dept\_ID, Dept\_Name, Building, Number\_of\_Classrooms)

VALUES ('D05', 'Biotech', 'Bio Block', 5);

DBMS\_OUTPUT.PUT\_LINE('Department Biotech inserted.');

ELSE

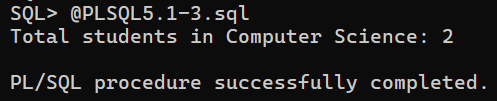
DBMS\_OUTPUT.PUT\_LINE('Department Biotech already exists.');

END IF;

END;

/

**OUTPUT:**

****

**5.2 Cursors**

**1. Insert a new student and show row count.**

**CODE:**

SET SERVEROUTPUT ON;

DECLARE

v\_rows NUMBER;

BEGIN

INSERT INTO Students (Student\_ID, Student\_Name, Dept\_ID, DOB)

VALUES ('S0005', 'Rahul Dev', 'D02', DATE '2003-09-15');

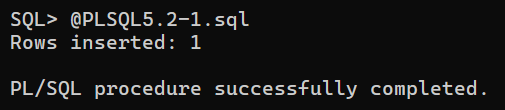
v\_rows := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Rows inserted: ' || v\_rows);

END;

/

**OUTPUT:**

****

**2. Update marks and show affected row count.**

**CODE:**

SET SERVEROUTPUT ON;

DECLARE

v\_rows NUMBER;

BEGIN

UPDATE Enrollments

SET Marks = Marks + 2

WHERE Semester = 'Sem1';

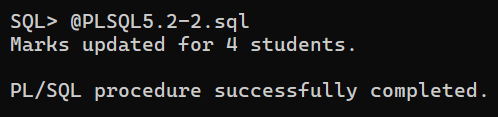
v\_rows := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Marks updated for ' || v\_rows || ' students.');

END;

/

**OUTPUT:**

****

**3. List students with marks above 80 (Explicit Cursor).**

**CODE:**

SET SERVEROUTPUT ON;

DECLARE

CURSOR c1 IS

SELECT s.Student\_Name, e.Marks

FROM Students s, Enrollments e

WHERE s.Student\_ID = e.Student\_ID

AND e.Marks > 80;

BEGIN

FOR rec IN c1

LOOP

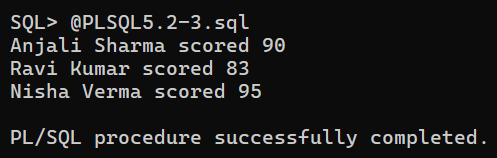
DBMS\_OUTPUT.PUT\_LINE(rec.Student\_Name || ' scored ' || rec.Marks);

END LOOP;

END;

/

**OUTPUT:**

****

**4. List all courses taught by P1001 (Explicit Cursor).**

**CODE:**

SET SERVEROUTPUT ON;

DECLARE

CURSOR c1 IS

SELECT Course\_Name

FROM Courses

WHERE Prof\_ID = 'P1001';

BEGIN

FOR rec IN c1

LOOP

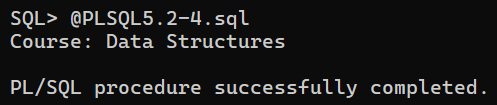
DBMS\_OUTPUT.PUT\_LINE('Course: ' || rec.Course\_Name);

END LOOP;

END;

/

**OUTPUT:**

****

**5.3 Procedures**

**1. Procedure: Accept student ID and return number of enrolled courses.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE student\_course\_count(

p\_student\_id IN VARCHAR2

) AS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM Enrollments

WHERE Student\_ID = p\_student\_id;

DBMS\_OUTPUT.PUT\_LINE('Student ' || p\_student\_id || ' is enrolled in ' || v\_count || ' courses.');

END;

/

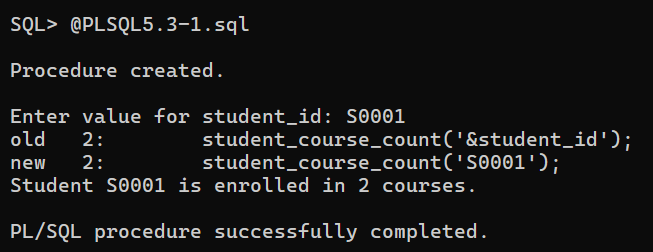
BEGIN

student\_course\_count('&student\_id');

END;

/

**OUTPUT:**

****

**2. Procedure: Insert new course if department and professor exist.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE add\_course(

p\_course\_id IN VARCHAR2,

p\_course\_name IN VARCHAR2,

p\_dept\_id IN VARCHAR2,

p\_prof\_id IN VARCHAR2,

p\_credits IN NUMBER

) AS

v\_dept NUMBER;

v\_prof NUMBER;

BEGIN

-- Check if department exists

SELECT COUNT(\*) INTO v\_dept FROM Departments WHERE Dept\_ID = p\_dept\_id;

-- Check if professor exists

SELECT COUNT(\*) INTO v\_prof FROM Professors WHERE Prof\_ID = p\_prof\_id;

IF v\_dept > 0 AND v\_prof > 0 THEN

INSERT INTO Courses (Course\_ID, Course\_Name, Dept\_ID, Prof\_ID, Credits, Total\_Students)

VALUES (p\_course\_id, p\_course\_name, p\_dept\_id, p\_prof\_id, p\_credits, 0);

DBMS\_OUTPUT.PUT\_LINE('Course ' || p\_course\_name || ' added successfully.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Department or Professor not found. Cannot add course.');

END IF;

END;

/

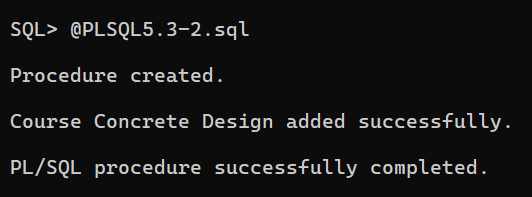
BEGIN

add\_course('CIV102', 'Concrete Design', 'D04', 'P1004', 3);

END;

/

**OUTPUT:**

****

**3. Procedure: Update course credits and log to new table.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE TABLE Course\_Log (

Course\_ID VARCHAR2(6),

Old\_Credits NUMBER(1),

New\_Credits NUMBER(1),

Update\_Date DATE

);

CREATE OR REPLACE PROCEDURE update\_course\_credits(

p\_course\_id IN VARCHAR2,

p\_new\_credits IN NUMBER

) AS

v\_old NUMBER;

BEGIN

SELECT Credits INTO v\_old FROM Courses WHERE Course\_ID = p\_course\_id;

UPDATE Courses

SET Credits = p\_new\_credits

WHERE Course\_ID = p\_course\_id;

INSERT INTO Course\_Log VALUES (p\_course\_id, v\_old, p\_new\_credits, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Credits updated and logged for ' || p\_course\_id);

END;

/

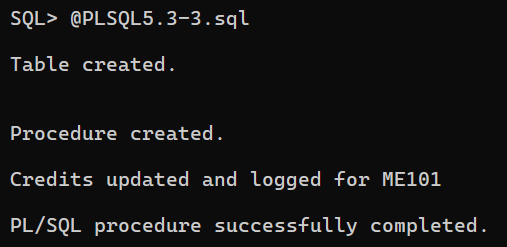
BEGIN

update\_course\_credits('ME101', 4);

END;

/

**OUTPUT:**

****

**5.4 Functions**

**1. Function to return age of student.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION get\_student\_age(p\_student\_id IN VARCHAR2)

RETURN NUMBER

AS

v\_age NUMBER;

v\_dob DATE;

BEGIN

-- Get DOB from table

SELECT DOB INTO v\_dob

FROM Students

WHERE Student\_ID = p\_student\_id;

-- Calculate age

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

RETURN v\_age;

END;

/

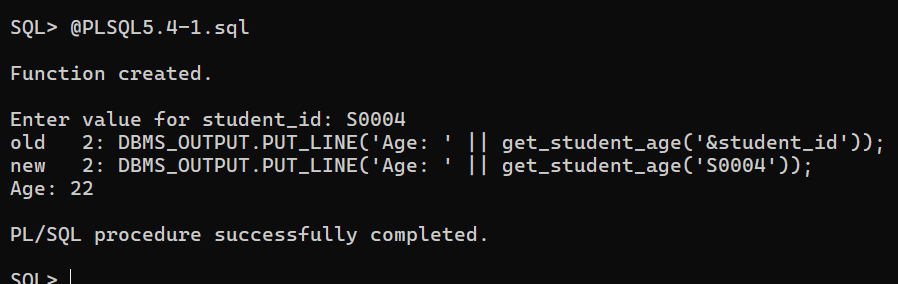
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Age: ' || get\_student\_age('&student\_id'));

END;

/

**OUTPUT:**

****

**2. Function to count courses offered by a department.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION count\_courses\_by\_dept(p\_dept\_id IN VARCHAR2)

RETURN NUMBER

AS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM Courses

WHERE Dept\_ID = p\_dept\_id;

RETURN v\_count;

END;

/

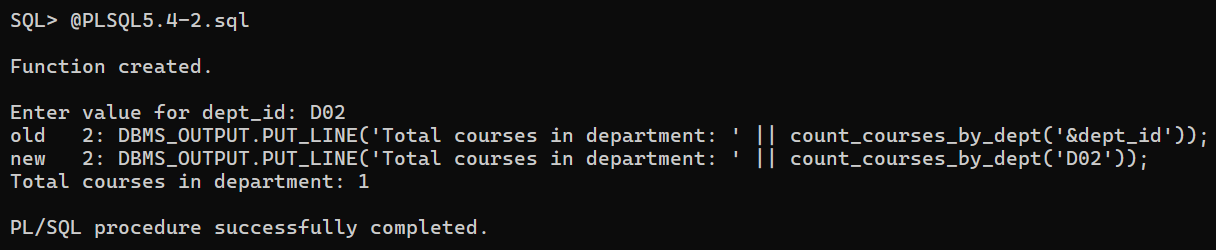
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Total courses in department: ' || count\_courses\_by\_dept('&dept\_id'));

END;

/

**OUTPUT:**

****

**3. Function to check if a professor teaches more than 2 courses.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION teaches\_more\_than\_two(p\_prof\_id IN VARCHAR2)

RETURN VARCHAR2

AS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM Courses

WHERE Prof\_ID = p\_prof\_id;

IF v\_count > 2 THEN

RETURN 'YES';

ELSE

RETURN 'NO';

END IF;

END;

/

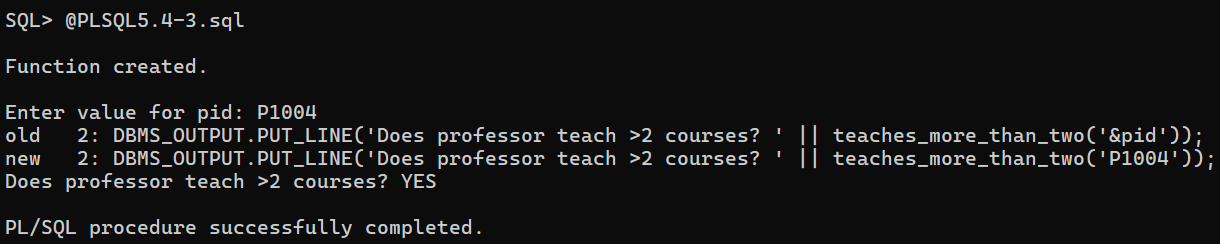
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Does professor teach >2 courses? ' || teaches\_more\_than\_two('&pid'));

END;

/

**OUTPUT:**

****

**5.5 Triggers**

**1. Prevent deletion of courses with more than 2 credits.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER trg\_prevent\_delete\_course

BEFORE DELETE ON Courses

FOR EACH ROW

BEGIN

IF :OLD.Credits > 2 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Cannot delete course with more than 2 credits.');

END IF;

END;

/

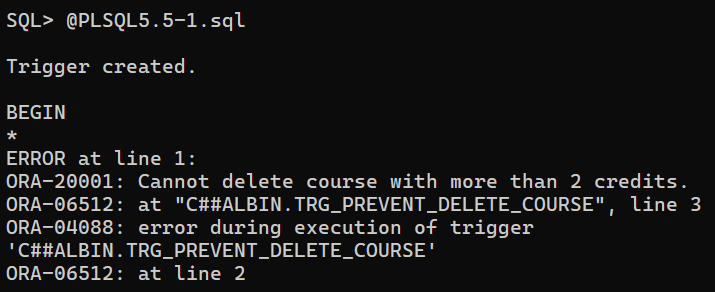
BEGIN

DELETE from Courses where Credits>2;

END;

/

**OUTPUT:**



**2. Log updates to Marks in Enrollments\_Log.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE TABLE Enrollments\_Log (

Student\_ID VARCHAR2(6),

Course\_ID VARCHAR2(6),

Old\_Marks NUMBER(5,2),

New\_Marks NUMBER(5,2),

Update\_Date DATE

);

CREATE OR REPLACE TRIGGER trg\_log\_marks\_update

AFTER UPDATE OF Marks ON Enrollments

FOR EACH ROW

BEGIN

INSERT INTO Enrollments\_Log

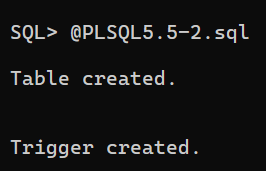
VALUES (:OLD.Student\_ID, :OLD.Course\_ID, :OLD.Marks, :NEW.Marks,

SYSDATE);

END;

/

**OUTPUT:**

****

**3. Prevent duplicate enrollments in a course.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER trg\_prevent\_duplicate\_enroll

BEFORE INSERT ON Enrollments

FOR EACH ROW

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count

FROM Enrollments

WHERE Student\_ID = :NEW.Student\_ID

AND Course\_ID = :NEW.Course\_ID;

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Student already enrolled in this course.');

END IF;

END;

/

BEGIN

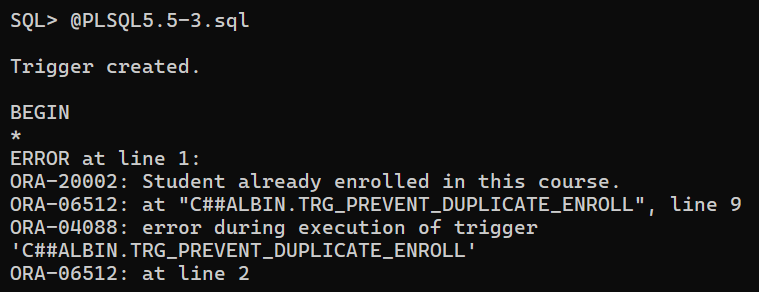
INSERT INTO Enrollments (Student\_ID, Course\_ID, Semester, Marks)

VALUES ('S0001', 'CSE101', 'Sem1', 90);

END;

/

**OUTPUT:**

****

**4. Prevent update / delete being done on Sunday / Saturday.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE TRIGGER trg\_no\_weekend\_changes

BEFORE UPDATE OR DELETE ON Courses

FOR EACH ROW

BEGIN

-- Check if today is Saturday or Sunday

IF TO\_CHAR(SYSDATE,'D') IN ('7','1') THEN

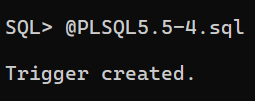
RAISE\_APPLICATION\_ERROR(-20003, 'Updates or deletes not allowed on weekends.');

END IF;

END;

/

**OUTPUT:**

****

**5.6 Views**

**1. View: Students and number of courses they are enrolled in.**

**CODE:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE VIEW Students\_Course\_Count AS

SELECT s.Student\_ID,

s.Student\_Name,

(SELECT COUNT(\*)

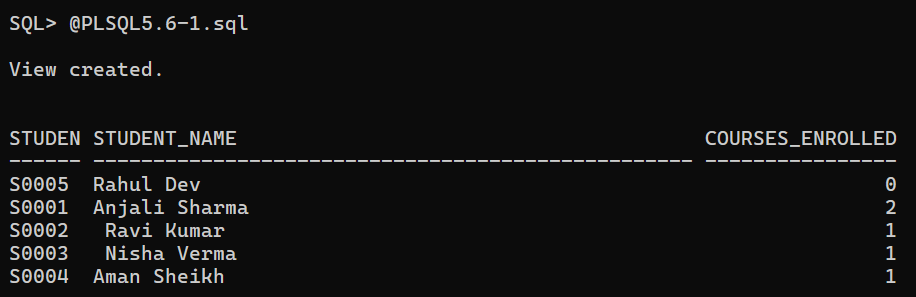
FROM Enrollments e

WHERE e.Student\_ID = s.Student\_ID) AS Courses\_Enrolled

FROM Students s;

SELECT \* FROM Students\_Course\_Count;

**OUTPUT:**

****

**2.View: Courses and average marks of enrolled students (only for courses with >1 student).**

**CODE:**

CREATE OR REPLACE VIEW Courses\_Avg\_Marks AS

SELECT c.Course\_ID,

c.Course\_Name,

AVG(e.Marks) AS Avg\_Marks,

COUNT(e.Student\_ID) AS Student\_Count

FROM Courses c

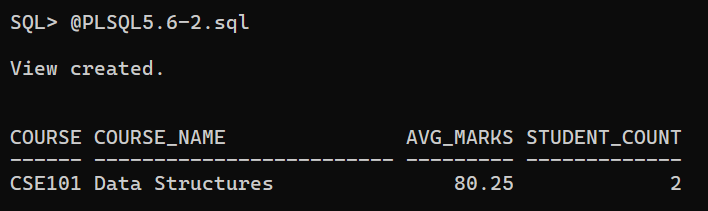
JOIN Enrollments e ON c.Course\_ID = e.Course\_ID

GROUP BY c.Course\_ID, c.Course\_Name

HAVING COUNT(e.Student\_ID) > 1;

SELECT \* FROM Courses\_Avg\_Marks;

**OUTPUT:**

****