Task 1

CREATE TABLE Studentinfo (

STU\_ID INT PRIMARY KEY,

STU\_NAME VARCHAR(255),

DOB DATE,

PHONE\_NO VARCHAR(15),

EMAIL\_ID VARCHAR(255),

ADDRESS VARCHAR(255)

);

-- Create Coursesinfo table

CREATE TABLE Coursesinfo (

COURSE\_ID INT PRIMARY KEY,

COURSE\_NAME VARCHAR(255),

COURSE\_INSTRUCTOR\_NAME VARCHAR(255)

);

-- Create Enrollmentinfo table with foreign key constraints

CREATE TABLE Enrollmentinfo (

ENROLLMENT\_ID INT PRIMARY KEY,

STU\_ID INT,

COURSE\_ID INT,

ENROLL\_STATUS VARCHAR(20), -- You can use an enum or foreign key to a status table for more control

FOREIGN KEY (STU\_ID) REFERENCES Studentinfo(STU\_ID),

FOREIGN KEY (COURSE\_ID) REFERENCES Coursesinfo(COURSE\_ID)

);

INSERT INTO Studentinfo (STU\_ID, STU\_NAME, DOB, PHONE\_NO, EMAIL\_ID, ADDRESS)

VALUES

(1, 'Namit Goyal' , '1995-05-12', '6876543205', 'namit.goyal@gmail.com', '111, Lajpat Nagar, Indore'),

(2, 'Abhilasha Gour', '1998-08-25', '8749804640' , 'abhilasha.gour@gmail.com', '444, Avantika Nager, Dewas'),

(3, 'Rahul Patel', '1997-03-10', '8664643109' , 'rahul.patel@gmail.com', '888, Lake Road, Bangalore');

INSERT INTO Coursesinfo (COURSE\_ID, COURSE\_NAME, COURSE\_INSTRUCTOR\_NAME)

VALUES

(101, 'Computer Science', 'Dr. Neha Verma'),

(102, 'Electrical Engineering', 'Prof. Rajesh Singh'),

(103, 'Economics', 'Dr. Abhilasha Singh');

INSERT INTO Enrollmentinfo (ENROLLMENT\_ID, STU\_ID, COURSE\_ID, ENROLL\_STATUS)

VALUES

(501, 1, 101, 'Enrolled'),

(502, 2, 102, 'Enrolled'),

(503, 3, 101, 'Enrolled'),

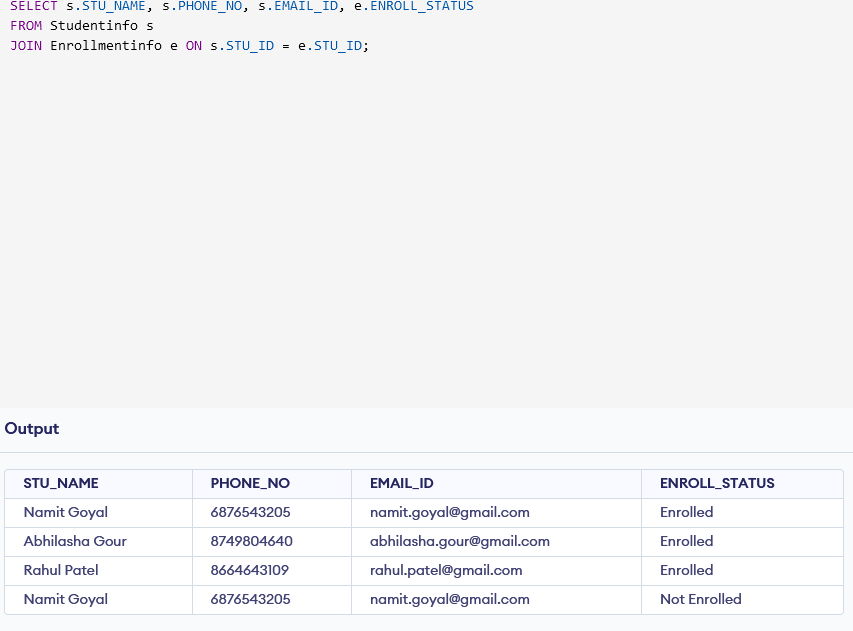
(504, 1, 103, 'Not Enrolled');

Task.1.3.a

SELECT s.STU\_NAME, s.PHONE\_NO, s.EMAIL\_ID, e.ENROLL\_STATUS

FROM Studentinfo s

JOIN Enrollmentinfo e ON s.STU\_ID = e.STU\_ID;



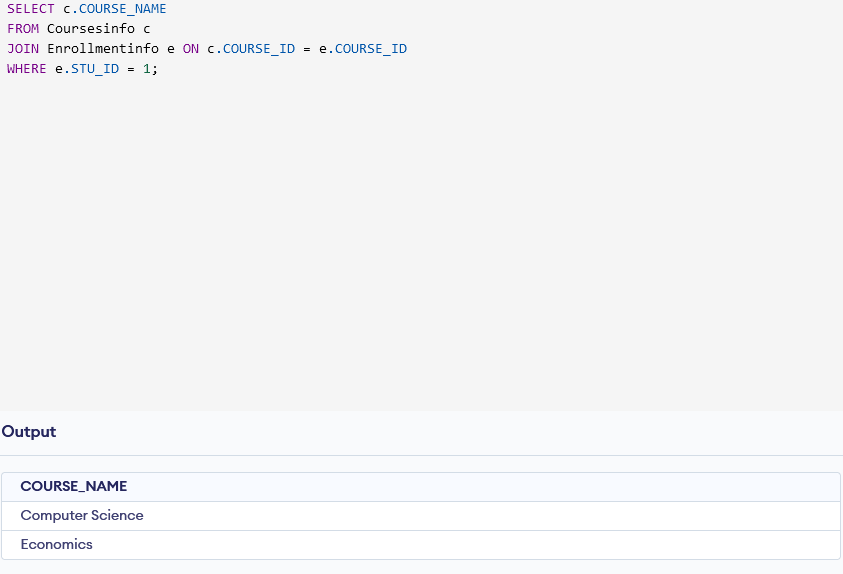
Task1.3.b

SELECT c.COURSE\_NAME

FROM Coursesinfo c

JOIN Enrollmentinfo e ON c.COURSE\_ID = e.COURSE\_ID

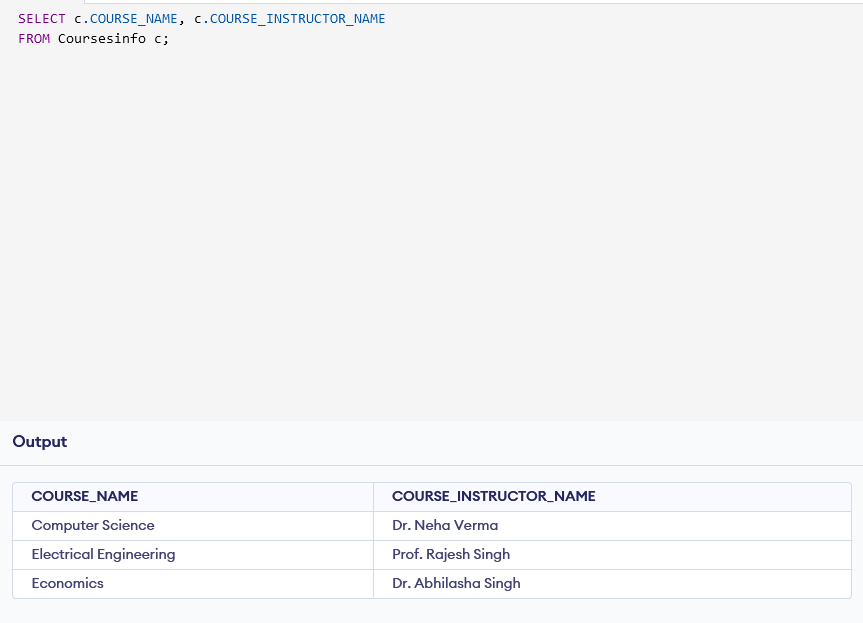
WHERE e.STU\_ID = 1;



Task1.3.c

SELECT c.COURSE\_NAME, c.COURSE\_INSTRUCTOR\_NAME

FROM Coursesinfo c;

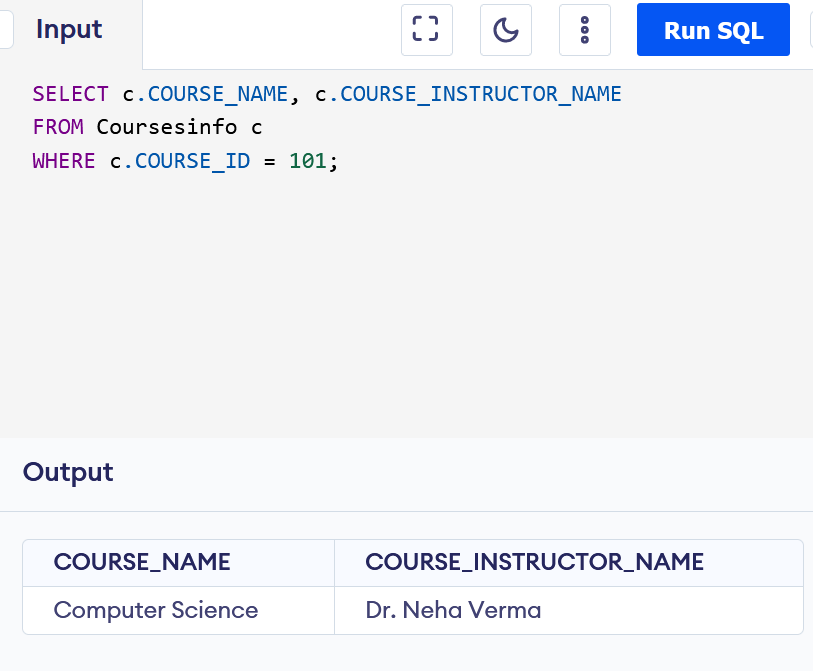


Task1.3.d

SELECT c.COURSE\_NAME, c.COURSE\_INSTRUCTOR\_NAME

FROM Coursesinfo c

WHERE c.COURSE\_ID = 101;

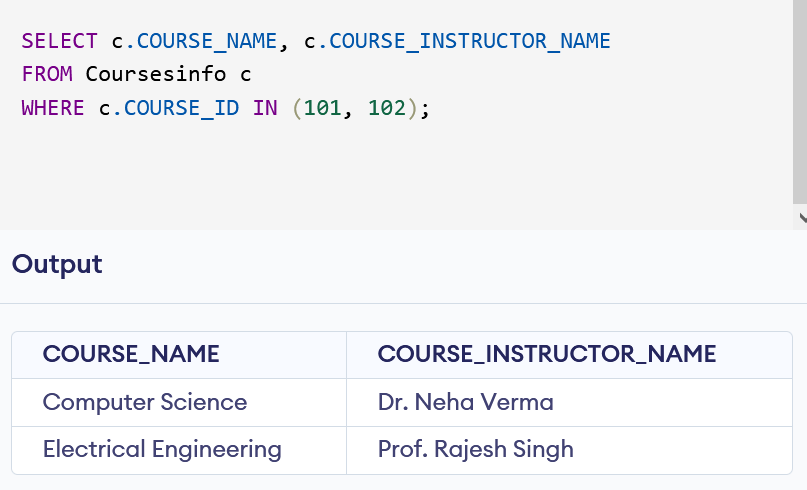


Task1.3.e

SELECT c.COURSE\_NAME, c.COURSE\_INSTRUCTOR\_NAME

FROM Coursesinfo c

WHERE c.COURSE\_ID IN (101, 102);



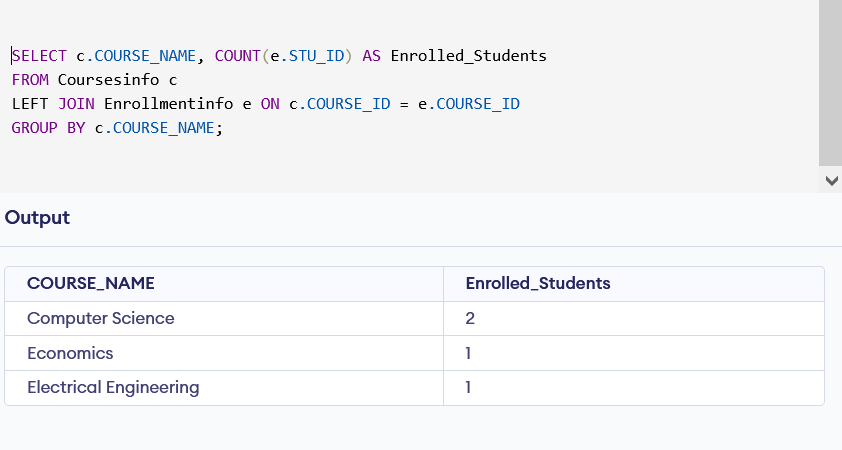
Task.1.4.a

SELECT c.COURSE\_NAME, COUNT(e.STU\_ID) AS Enrolled\_Students

FROM Coursesinfo c

LEFT JOIN Enrollmentinfo e ON c.COURSE\_ID = e.COURSE\_ID

GROUP BY c.COURSE\_NAME;



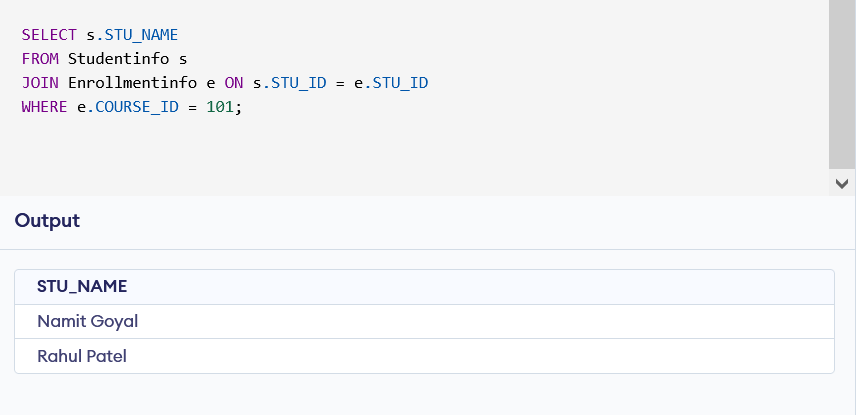
Task 1.4.b

SELECT s.STU\_NAME

FROM Studentinfo s

JOIN Enrollmentinfo e ON s.STU\_ID = e.STU\_ID

WHERE e.COURSE\_ID = 101;



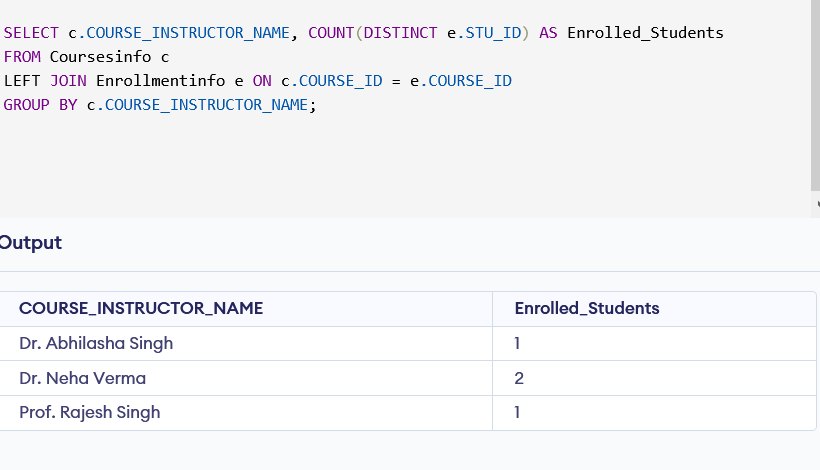
Task1.4.c

SELECT c.COURSE\_INSTRUCTOR\_NAME, COUNT(DISTINCT e.STU\_ID) AS Enrolled\_Students

FROM Coursesinfo c

LEFT JOIN Enrollmentinfo e ON c.COURSE\_ID = e.COURSE\_ID

GROUP BY c.COURSE\_INSTRUCTOR\_NAME;



Task 1.4.d

SELECT s.STU\_NAME, COUNT(e.COURSE\_ID) AS Enrolled\_Courses

FROM Studentinfo s

JOIN Enrollmentinfo e ON s.STU\_ID = e.STU\_ID

GROUP BY s.STU\_NAME

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



4e

SELECT c.COURSE\_NAME, COUNT(e.STU\_ID) AS Enrolled\_Students\_Count

FROM Coursesinfo c

LEFT JOIN Enrollmentinfo e ON c.COURSE\_ID = e.COURSE\_ID

GROUP BY c.COURSE\_NAME

ORDER BY Enrolled\_Students\_Count DESC;

