**PRACTICAL N0 – 2**

**Aim:** Subquery-join operations on Relational Schema

**USING (practical 1)**

1. **Count the customers with grades above Bangalore’s average.**

**Code:**

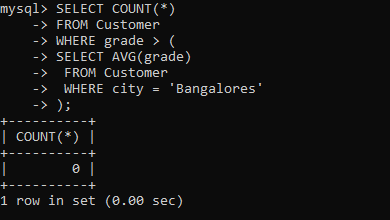
SELECT COUNT(\*)

FROM Customer WHERE grade > ( SELECT AVG(grade) FROM Customer

WHERE city = 'Bangalores'

);

**Output:**



1. **Find the name and numbers of all salesmen who had more than one customer.**

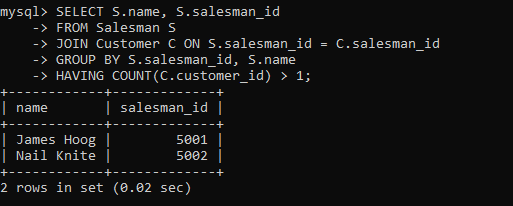
**Code:**

SELECT S.name, S.salesman\_id FROM Salesman S

JOIN Customer C ON S.salesman\_id = C.salesman\_id GROUP BY S.salesman\_id, S.name

HAVING COUNT(C.customer\_id) > 1;

**Output:**



1. **List all salesmen and indicate those who have and don’t have**

**customers in their cities (Use UNION operation.)**

**Code:**

SELECT S.salesman\_id, S.name, 'Has Customers' AS customer\_status FROM Salesman S

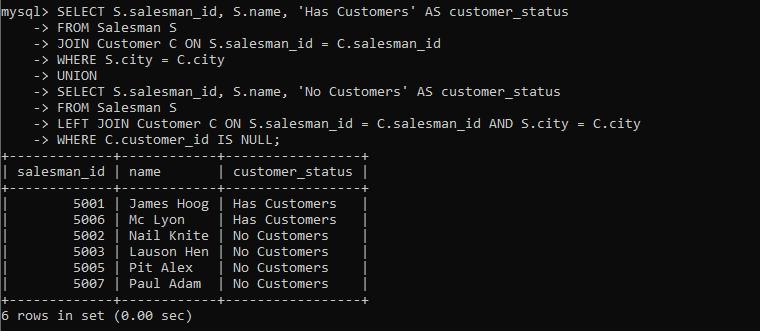
JOIN Customer C ON S.salesman\_id = C.salesman\_id WHERE S.city = C.city

UNION

SELECT S.salesman\_id, S.name, 'No Customers' AS customer\_status FROM Salesman S

LEFT JOIN Customer C ON S.salesman\_id = C.salesman\_id AND S.city = C.city WHERE C.customer\_id IS NULL;

**Output:**



1. **Create a view that finds the salesman who has the customer with the highest order of a day.**

**Code:**

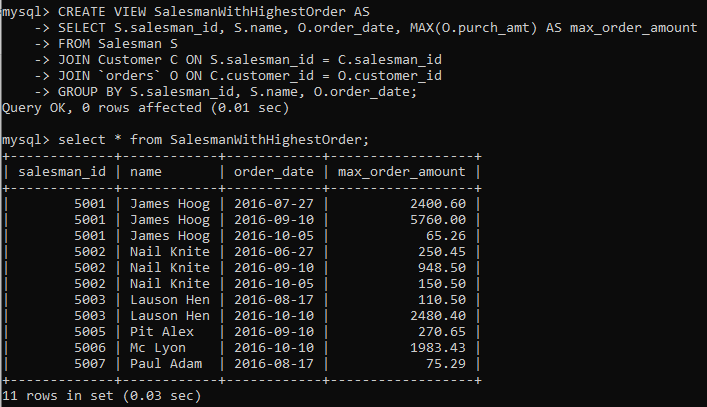
CREATE VIEW SalesmanWithHighestOrder AS

SELECT S.salesman\_id, S.name, O.order\_date, MAX(O.purch\_amt) AS max\_order\_amount

FROM Salesman S

JOIN Customer C ON S.salesman\_id = C.salesman\_id JOIN `orders` O ON C.customer\_id = O.customer\_id GROUP BY S.salesman\_id, S.name, O.order\_date; select \* from SalesmanWithHighestOrder;

**Output:**



1. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted**

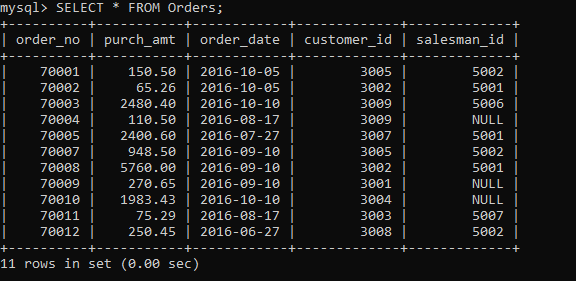
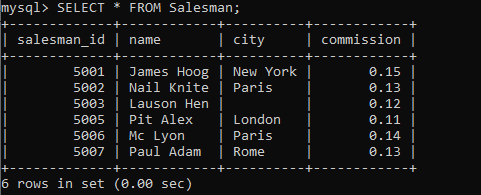
**Code:**

DELETE FROM salesman WHERE salesman\_id = 1000; SELECT \* FROM Salesman;

SELECT \* FROM Orders;

**Output:**



****

**2. Design ERD for the following schema and execute the following Queries on it:**

**Consider the schema for Movie Database: ACTOR (Act\_id, Act\_Name, Act\_Gender) DIRECTOR (Dir\_id, Dir\_Name, Dir\_Phone)**

**MOVIES (Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id) MOVIE\_CAST (Act\_id, Mov\_id, Role)**

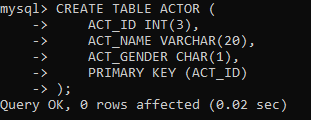
**RATING (Mov\_id, Rev\_Stars) Code:**

CREATE TABLE ACTOR ( ACT\_ID INT (3),

ACT\_NAME VARCHAR (20),

ACT\_GENDER CHAR (1), PRIMARY KEY (ACT\_ID));

**Output:**

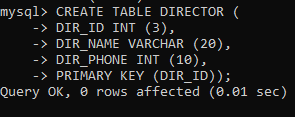


CREATE TABLE DIRECTOR ( DIR\_ID INT (3),

DIR\_NAME VARCHAR (20),

DIR\_PHONE INT (10), PRIMARY KEY (DIR\_ID));

**Output:**



**Code:**

CREATE TABLE MOVIES ( MOV\_ID INT (4),

MOV\_TITLE VARCHAR (25),

MOV\_YEAR INT (4),

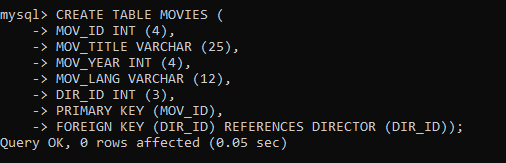
MOV\_LANG VARCHAR (12),

DIR\_ID INT (3),

PRIMARY KEY (MOV\_ID),

FOREIGN KEY (DIR\_ID) REFERENCES DIRECTOR (DIR\_ID));

**Output:**



CREATE TABLE MOVIE\_CAST ( ACT\_ID INT (3),

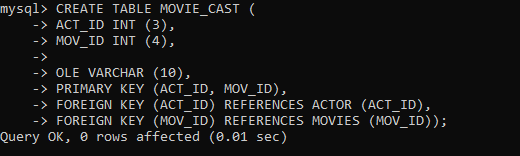
MOV\_ID INT (4),

OLE VARCHAR (10),

PRIMARY KEY (ACT\_ID, MOV\_ID),

FOREIGN KEY (ACT\_ID) REFERENCES ACTOR (ACT\_ID), FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

**Output:**



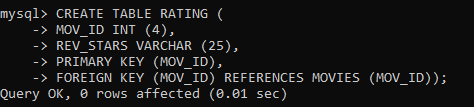
**Code:**

CREATE TABLE RATING ( MOV\_ID INT (4),

REV\_STARS VARCHAR (25), PRIMARY KEY (MOV\_ID),

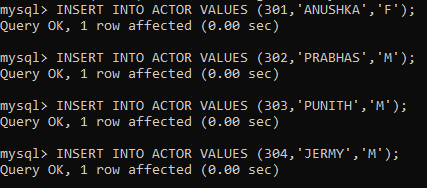
FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

**Output:**



INSERT INTO ACTOR VALUES (301,'ANUSHKA','F'); INSERT INTO ACTOR VALUES (302,'PRABHAS','M'); INSERT INTO ACTOR VALUES (303,'PUNITH','M'); INSERT INTO ACTOR VALUES (304,'JERMY','M');

**Output:**

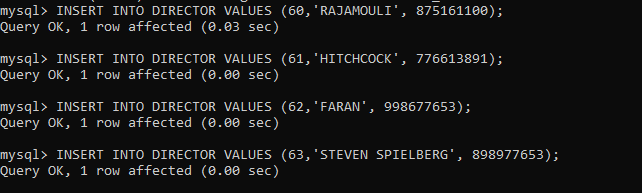


**Code:**

INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 875161100); INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 776613891); INSERT INTO DIRECTOR VALUES (62,'FARAN', 998677653);

INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG', 898977653);

**Output:**



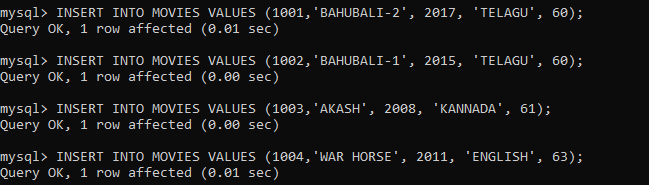
INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELAGU', 60);

INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELAGU', 60);

INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);

INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);

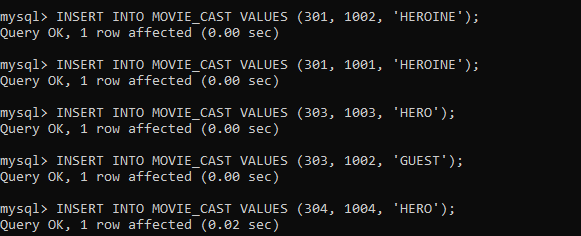
**Output:**



**Code:**

INSERT INTO MOVIE\_CAST VALUES (301, 1002, 'HEROINE'); INSERT INTO MOVIE\_CAST VALUES (301, 1001, 'HEROINE'); INSERT INTO MOVIE\_CAST VALUES (303, 1003, 'HERO'); INSERT INTO MOVIE\_CAST VALUES (303, 1002, 'GUEST'); INSERT INTO MOVIE\_CAST VALUES (304, 1004, 'HERO');

**Output:**



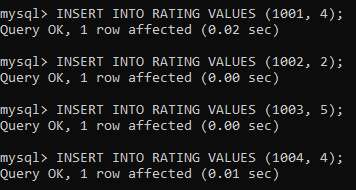
INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);

**Output:**



Write SQL queries to

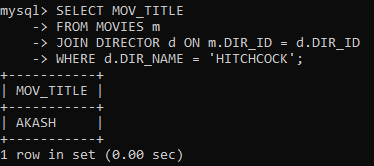
1. **List the titles of all movies directed by ‘Hitchcock’.**

**Code:**

SELECT MOV\_TITLE FROM MOVIES m

JOIN DIRECTOR d ON m.DIR\_ID = d.DIR\_ID WHERE d.DIR\_NAME = 'HITCHCOCK';

**Output:**



1. **Find the movie names where one or more actors acted in two or more movies.**

**Code:**

SELECT DISTINCT m.MOV\_TITLE FROM MOVIES m

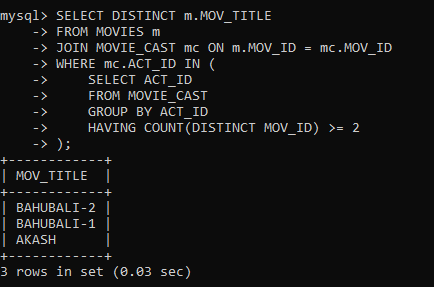
JOIN MOVIE\_CAST mc ON m.MOV\_ID = mc.MOV\_ID WHERE mc.ACT\_ID IN (

SELECT ACT\_ID FROM MOVIE\_CAST GROUP BY ACT\_ID

HAVING COUNT(DISTINCT MOV\_ID) >= 2

);

**Output:**



1. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**

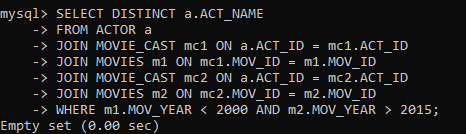
**Code:**

SELECT DISTINCT a.ACT\_NAME FROM ACTOR a

JOIN MOVIE\_CAST mc1 ON a.ACT\_ID = mc1.ACT\_ID JOIN MOVIES m1 ON mc1.MOV\_ID = m1.MOV\_ID JOIN MOVIE\_CAST mc2 ON a.ACT\_ID = mc2.ACT\_ID JOIN MOVIES m2 ON mc2.MOV\_ID = m2.MOV\_ID

WHERE m1.MOV\_YEAR < 2000 AND m2.MOV\_YEAR > 2015;

**Output:**



1. **Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**

**Code:**

SELECT m.MOV\_TITLE, r.REV\_STARS, ( SELECT MAX(r1.REV\_STARS)

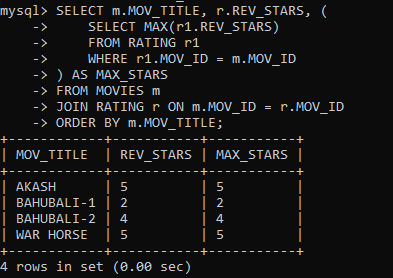
FROM RATING r1

WHERE r1.MOV\_ID = m.MOV\_ID

) AS MAX\_STARS FROM MOVIES m

JOIN RATING r ON m.MOV\_ID = r.MOV\_ID ORDER BY m.MOV\_TITLE;

**Output:**



1. **Update rating of all movies directed by ‘Steven Spielberg’ to 5.**

**Code:**

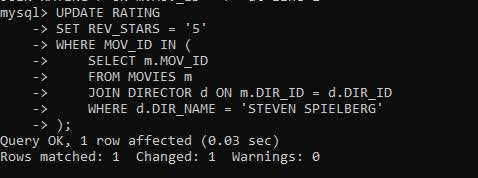
UPDATE RATING SET REV\_STARS = '5'

WHERE MOV\_ID IN ( SELECT m.MOV\_ID FROM MOVIES m

JOIN DIRECTOR d ON m.DIR\_ID = d.DIR\_ID WHERE d.DIR\_NAME = 'STEVEN SPIELBERG'

);

**Output:**



**3. Design ERD for the following schema and execute the following Queries on it:**

**Code:**

CREATE TABLE students ( stno INT PRIMARY KEY, name VARCHAR(50), addr VARCHAR(255), city VARCHAR(50),

state VARCHAR(2), zip VARCHAR(10)

);

CREATE TABLE INSTRUCTORS ( empno INT PRIMARY KEY, name VARCHAR(50),

rank VARCHAR(20),

roomno VARCHAR(10), telno VARCHAR(15)

);

CREATE TABLE COURSES ( cno INT PRIMARY KEY,

cname VARCHAR(50),

cr INT, cap INT

);

CREATE TABLE GRADES (

stno INT, empno INT, cno INT,

sem VARCHAR(10),

year INT, grade INT,

PRIMARY KEY (stno),

FOREIGN KEY (stno) REFERENCES students(stno),

FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno), FOREIGN KEY (cno) REFERENCES COURSES(cno)

);

CREATE TABLE ADVISING (

stno INT, empno INT,

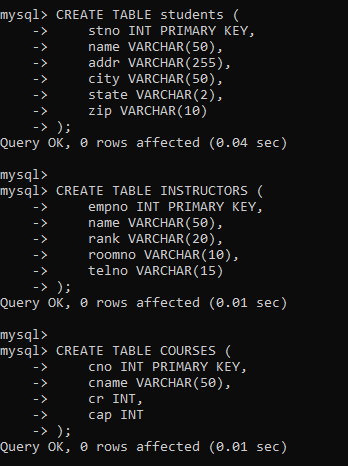
PRIMARY KEY (stno, empno),

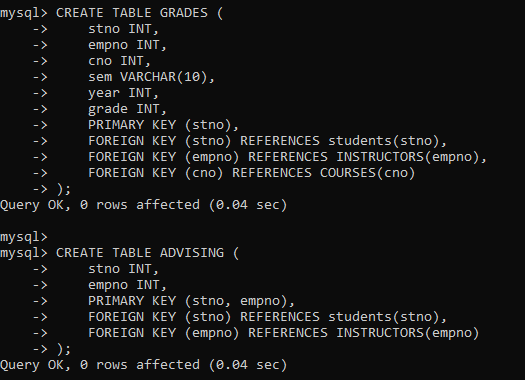
FOREIGN KEY (stno) REFERENCES students(stno),

FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno)

);

**Output:**



****

**Code:**

INSERT INTO COURSES (cno, cname, cr, cap) VALUES

(1, 'Math101', 3, 30),

(2, 'CS210', 4, 25),

(3, 'Physics101', 3, 20);

INSERT INTO students (stno, name) VALUES

(1, 'John Doe'),

(2, 'Jane Smith'),

(3, 'Alice Johnson');

INSERT INTO instructors (empno, name) VALUES

(101, 'Instructor A'),

(102, 'Instructor B'),

(103, 'Instructor C');

INSERT INTO GRADES (stno, empno, cno, sem, year, grade) VALUES

(1, 101, 1, 'Fall', 2021, 85),

(2, 102, 2, 'Fall', 2021, 92),

(3, 103, 3, 'Fall', 2021, 78);

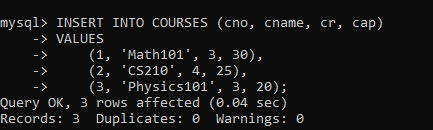
INSERT INTO ADVISING (stno, empno) VALUES

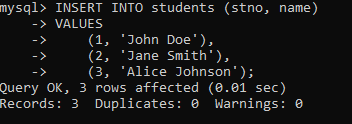
(1, 101),

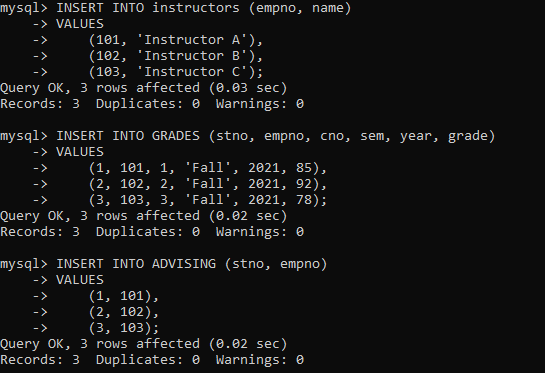
(2, 102),

(3, 103);

**Output:**



****



**For odd rollnumbers(any 10 )**

1. **Find the names of students who took some four-credit courses.**

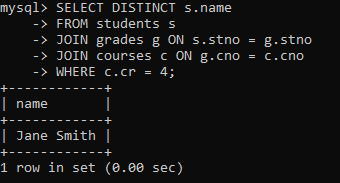
**Code:**

SELECT DISTINCT s.name

FROM students s

JOIN grades g ON s.stno = g.stno JOIN courses c ON g.cno = c.cno WHERE c.cr = 4;

**Output:**



1. **Find the names of students who took every four-credit course. Code:**

SELECT s.name FROM students s WHERE NOT EXISTS (

SELECT 1

FROM courses c

WHERE c.cr = 4 AND NOT EXISTS ( SELECT 1

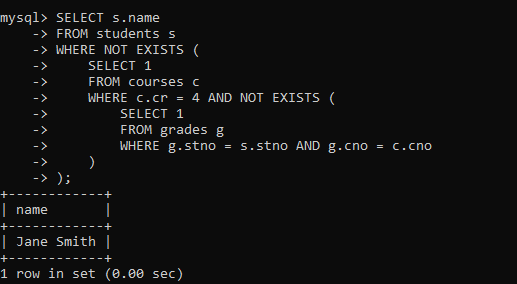
FROM grades g

WHERE g.stno = s.stno AND g.cno = c.cno

)

);

**Output:**



1. **Find the names of students who took a course with an instructor who is also their advisor.**

**Code:**

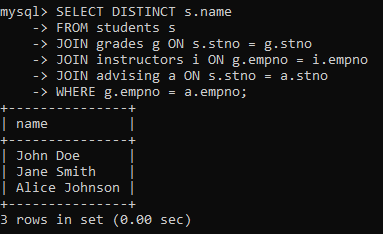
SELECT DISTINCT s.name

FROM students s

JOIN grades g ON s.stno = g.stno

JOIN instructors i ON g.empno = i.empno JOIN advising a ON s.stno = a.stno WHERE g.empno = a.empno;

**Output:**



1. **Find the names of students who took cs210 and cs310. Code:**

SELECT s.name FROM students s WHERE EXISTS (

SELECT 1

FROM grades g

JOIN courses c ON g.cno = c.cno

WHERE s.stno = g.stno AND c.cname = 'cs210'

)

AND EXISTS ( SELECT 1

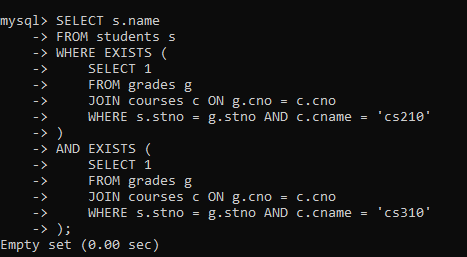
FROM grades g

JOIN courses c ON g.cno = c.cno

WHERE s.stno = g.stno AND c.cname = 'cs310'

);

**Output:**



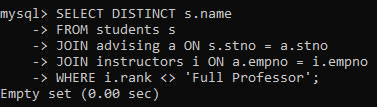
1. **Find the names of all students whose advisor is not a full professor.**

**Code:**

SELECT DISTINCT s.name

FROM students s

JOIN advising a ON s.stno = a.stno

****JOIN instructors i ON a.empno = i.empno WHERE i.rank <> 'Full Professor'; **Output:**

1. **Find instructors who taught students who are advised by another instructor who shares the same room.**

**Code:**

SELECT DISTINCT i1.name

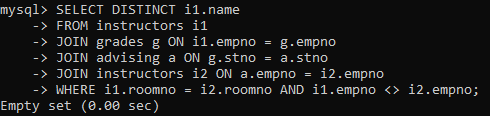
FROM instructors i1

JOIN grades g ON i1.empno = g.empno JOIN advising a ON g.stno = a.stno

JOIN instructors i2 ON a.empno = i2.empno

WHERE i1.roomno = i2.roomno AND i1.empno <> i2.empno;

**Output:**



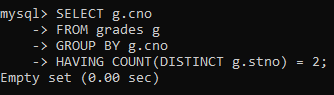
1. **Find course numbers for courses that enroll exactly two students**

**Code:**

SELECT g.cno FROM grades g GROUP BY g.cno

HAVING COUNT(DISTINCT g.stno) = 2;

**Output:**



1. **Find the names of all students for whom no other student lives in the same city.**

**Code:**

SELECT s1.name FROM students s1 WHERE NOT EXISTS (

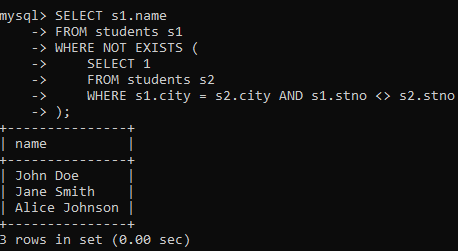
SELECT 1

FROM students s2

WHERE s1.city = s2.city AND s1.stno <> s2.stno

);

**Output:**



1. **Find course numbers of courses taken by students who live in Boston and which are taught by an associate professor.**

**Code:**

SELECT DISTINCT g.cno

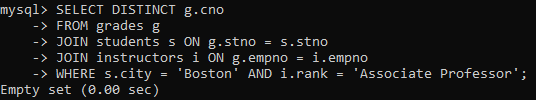
FROM grades g

JOIN students s ON g.stno = s.stno

JOIN instructors i ON g.empno = i.empno

WHERE s.city = 'Boston' AND i.rank = 'Associate Professor';

**Output:**



1. **Find the telephone numbers of instructors who teach a course taken by any student who lives in Boston.**

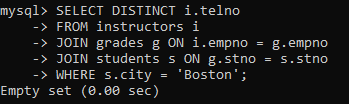
**Code:**

SELECT DISTINCT i.telno

FROM instructors i

JOIN grades g ON i.empno = g.empno JOIN students s ON g.stno = s.stno WHERE s.city = 'Boston';

**Output:**



1. **Find names of students who took every course taken by Richard Pierce.**

**Code:**

SELECT s.name FROM students s WHERE NOT EXISTS (

SELECT 1

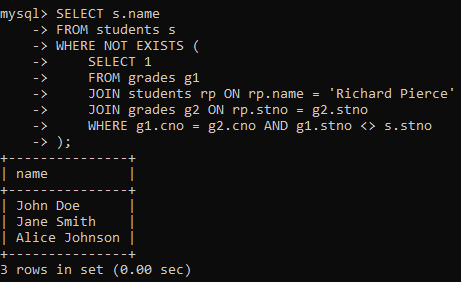
FROM grades g1

JOIN students rp ON rp.name = 'Richard Pierce' JOIN grades g2 ON rp.stno = g2.stno

WHERE g1.cno = g2.cno AND g1.stno <> s.stno

);

**Output:**



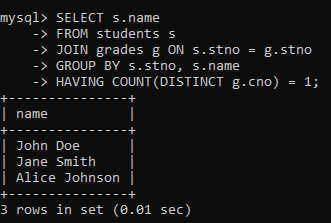
1. **Find the names of students who took only one course. Code:**

SELECT s.name FROM students s

JOIN grades g ON s.stno = g.stno GROUP BY s.stno, s.name

HAVING COUNT(DISTINCT g.cno) = 1;

**Output:**

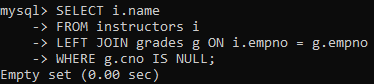


1. **Find the names of instructors who teach no course. Code:**

SELECT i.name FROM instructors i

LEFT JOIN grades g ON i.empno = g.empno WHERE g.cno IS NULL;

**Output:**



1. **Find the names of the instructors who taught only one course during the spring semester of 2001.**

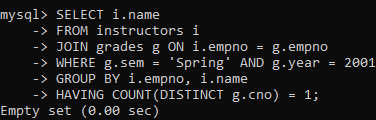
**Code:**

SELECT i.name FROM instructors i

JOIN grades g ON i.empno = g.empno WHERE g.sem = 'Spring' AND g.year = 2001 GROUP BY i.empno, i.name

HAVING COUNT(DISTINCT g.cno) = 1;

**Output:**



***For even rollnumbers(any 10)***

1. **Find the names of students who took only four-credit courses. Code:**

SELECT s.name FROM students s

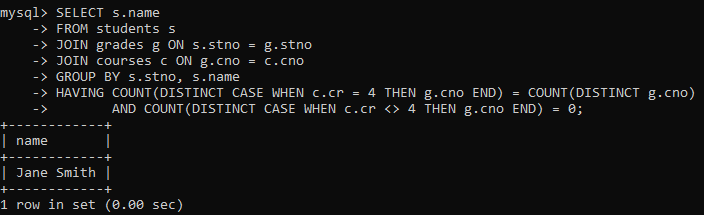
JOIN grades g ON s.stno = g.stno JOIN courses c ON g.cno = c.cno GROUP BY s.stno, s.name

HAVING COUNT(DISTINCT CASE WHEN c.cr = 4 THEN g.cno END) = COUNT(DISTINCT g.cno)

AND COUNT(DISTINCT CASE WHEN c.cr <> 4 THEN g.cno END) =

0;

**Output:**



1. **Find the names of students who took no four-credit courses.**

**Code:**

SELECT s.name FROM students s WHERE NOT EXISTS (

SELECT 1

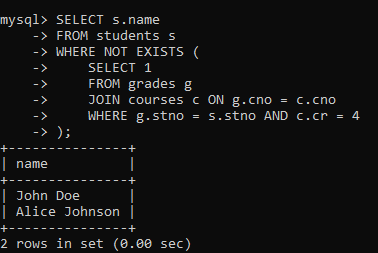
FROM grades g

JOIN courses c ON g.cno = c.cno

WHERE g.stno = s.stno AND c.cr = 4

);

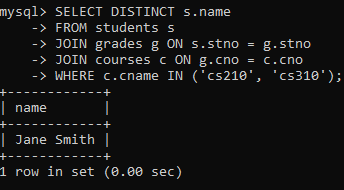
**Output:**



1. **Find the names of students who took cs210 or cs310. Code:**

SELECT DISTINCT s.name

FROM students s

****JOIN grades g ON s.stno = g.stno JOIN courses c ON g.cno = c.cno WHERE c.cname IN ('cs210', 'cs310'); **Output:**

1. **Find names of all students who have a cs210 grade higher than the highest grade given in cs310 and did not take any course with Prof. Evans.**

**Code:**

SELECT DISTINCT s.name

FROM students s

JOIN grades g1 ON s.stno = g1.stno JOIN courses c1 ON g1.cno = c1.cno

WHERE c1.cname = 'cs210' AND g1.grade > ( SELECT MAX(g2.grade)

FROM grades g2

JOIN courses c2 ON g2.cno = c2.cno WHERE c2.cname = 'cs310'

)

AND NOT EXISTS ( SELECT 1

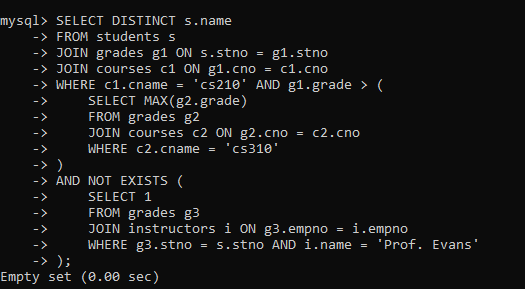
FROM grades g3

JOIN instructors i ON g3.empno = i.empno

WHERE g3.stno = s.stno AND i.name = 'Prof. Evans'

);

**Output:**



1. **Find course numbers for courses that enrol at least two students; solve the same query for courses that enroll at least three students.**

**Code:**

-- For courses with at least two students SELECT g.cno

FROM grades g GROUP BY g.cno

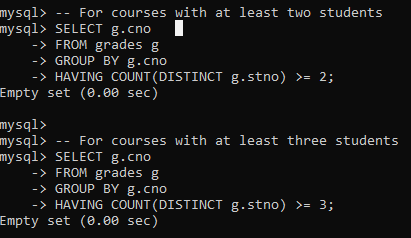
HAVING COUNT(DISTINCT g.stno) >= 2;

-- For courses with at least three students SELECT g.cno

FROM grades g GROUP BY g.cno

HAVING COUNT(DISTINCT g.stno) >= 3;

**Output:**



1. **Find the names of students who obtained the highest grade in cs210.**

**Code:**

SELECT s.name FROM students s

JOIN grades g ON s.stno = g.stno JOIN courses c ON g.cno = c.cno

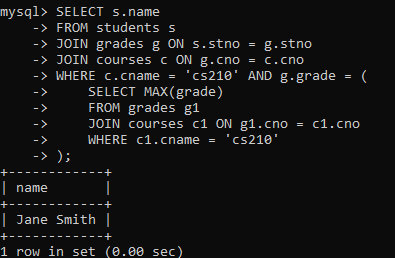
WHERE c.cname = 'cs210' AND g.grade = ( SELECT MAX(grade)

FROM grades g1

JOIN courses c1 ON g1.cno = c1.cno WHERE c1.cname = 'cs210'

);

**Output:**



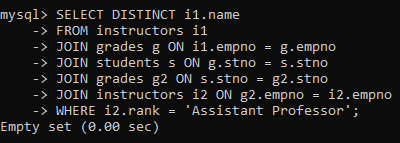
1. **Find the names of instructors who teach courses attended by students who took a course with an instructor who is an assistant professor.**

**Code:**

SELECT DISTINCT i1.name

FROM instructors i1

JOIN grades g ON i1.empno = g.empno JOIN students s ON g.stno = s.stno JOIN grades g2 ON s.stno = g2.stno

****JOIN instructors i2 ON g2.empno = i2.empno WHERE i2.rank = 'Assistant Professor'; **Output:**

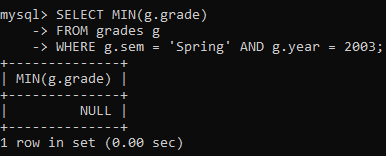
1. **Find the lowest grade of a student who took a course during the spring of 2003.**

**Code:**

SELECT MIN(g.grade) FROM grades g

WHERE g.sem = 'Spring' AND g.year = 2003;

**Output:**



1. **Find the names for students such that if prof. Evans teaches a course, then the student takes that course (although not necessarily with prof. Evans).**

**Code:**

SELECT s.name FROM students s WHERE NOT EXISTS (

SELECT 1

FROM courses c WHERE EXISTS (

SELECT 1

FROM grades g

WHERE g.stno = s.stno AND g.cno = c.cno

) AND EXISTS ( SELECT 1

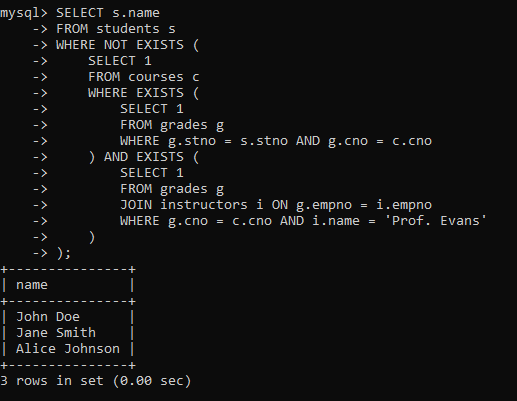
FROM grades g

JOIN instructors i ON g.empno = i.empno WHERE g.cno = c.cno AND i.name = 'Prof. Evans'

)

);

**Output:**



1. **Find the names of students whose advisor did not teach them any course.**

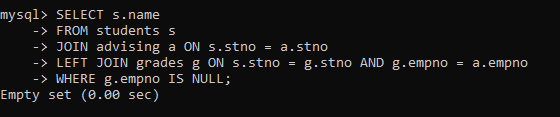
**Code:**

SELECT s.name FROM students s

JOIN advising a ON s.stno = a.stno

LEFT JOIN grades g ON s.stno = g.stno AND g.empno = a.empno WHERE g.empno IS NULL;

**Output:**



1. **Find the names of students who have failed all their courses (failing is defined as a grade less than 60).**

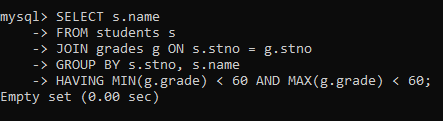
**Code:**

SELECT s.name FROM students s

JOIN grades g ON s.stno = g.stno GROUP BY s.stno, s.name

HAVING MIN(g.grade) < 60 AND MAX(g.grade) < 60;

**Output:**



1. **Find the highest grade of a student who never took cs110.**

**Code:**

SELECT MAX(g.grade) FROM grades g

WHERE g.stno NOT IN (

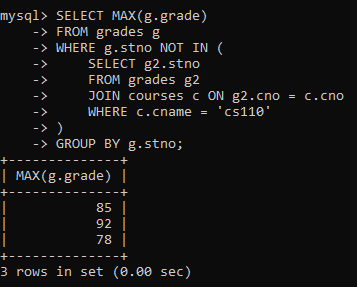
SELECT g2.stno FROM grades g2

JOIN courses c ON g2.cno = c.cno WHERE c.cname = 'cs110'

)

GROUP BY g.stno;

**Output:**

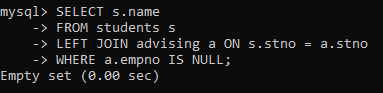
****

1. **Find the names of students who do not have an advisor. Code:**

SELECT s.name FROM students s

LEFT JOIN advising a ON s.stno = a.stno WHERE a.empno IS NULL;

**Output:**

****

1. **Find names of courses taken by students who do not live in Massachusetts (MA).**

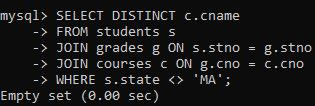
**Code:**

SELECT DISTINCT c.cname

FROM students s

JOIN grades g ON s.stno = g.stno JOIN courses c ON g.cno = c.cno WHERE s.state <> 'MA';

**Output:**

****