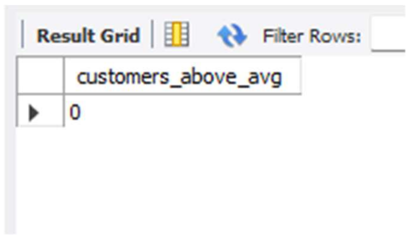
Practical 2 Subquery-join operations on Relational Schema

Class: MSc. DSAI Roll No.:L005

1. Count the customers with grades above Bangalore's average

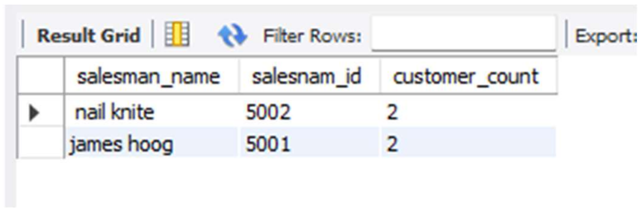
```
SELECT COUNT(*) AS customers_above_avg
FROM customer
WHERE grade > (
    SELECT AVG(grade)
    FROM customer
    WHERE city = 'Bangalore'
);
```



```
customers_above_avg
0
```

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

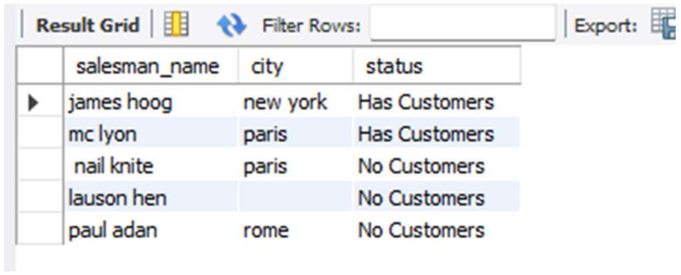
```
SELECT s.name AS salesman_name,
s.salesman_id, COUNT(c.customer_id) AS
customer_count
FROM salesmans s
JOIN customer c ON s.salesman_id =
c.salesman_id
GROUP BY s.name, s.salesman_id
HAVING COUNT(c.customer_id) > 1;
```



	salesman_name	salesnam_id	customer_count
▶	nail knite	5002	2
	james hoog	5001	2

3. **List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation).**

```
SELECT s.name AS salesman_name, s.city,
'Has Customers' AS status
FROM salesmans s
JOIN customer c ON s.salesman_id =
c.salesman_id AND s.city = c.city

UNION

SELECT s.name AS salesman_name, s.city, 'No
Customers' AS status
FROM salesmans s
WHERE s.salesman_id NOT IN (
    SELECT salesman_id
    FROM customer
    WHERE customer.city = s.city
);
```



	salesman_name	city	status
▶	james hoog	new york	Has Customers
	mc lyon	paris	Has Customers
	nail knite	paris	No Customers
	lauson hen		No Customers
	paul adan	rome	No Customers

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

```
CREATE VIEW highest_order_salesman AS
SELECT
    o.order_date,
    o.purch_amt AS highest_order,
    c.customer_name,
    s.name AS salesman_name
FROM orders o
JOIN customer c ON o.customer_id =
c.customer_id
JOIN salesmans s ON o.salesman_id =
s.salesman_id
WHERE o.purch_amt = (
    SELECT MAX(purch_amt)
    FROM orders o2
    WHERE o2.order_date = o.order_date
);
```

5. **Demonstrate the DELETE operation by removing a salesman with ID 1000. All his orders must also be deleted.**

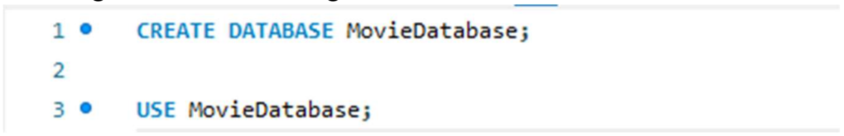
```
ALTER TABLE orders
ADD CONSTRAINT fk_salesman_order
FOREIGN KEY (salesman_id) REFERENCES
salesman(salesman_id)
ON DELETE CASCADE;

DELETE FROM salesman
WHERE salesman_id = 1000;
```

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

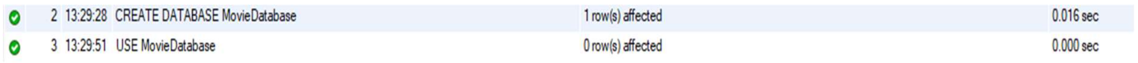
Consider the schema for Movie Database:

```
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)
```

Creating database and using it:



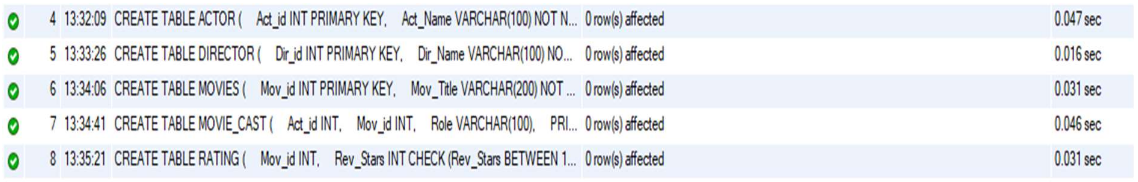
```
CREATE DATABASE MovieDatabase;

USE MovieDatabase;
```



```
2  13:29:28  CREATE DATABASE MovieDatabase    1 row(s) affected    0.016 sec
3  13:29:51  USE MovieDatabase                0 row(s) affected    0.000 sec
```

Creating required tables:

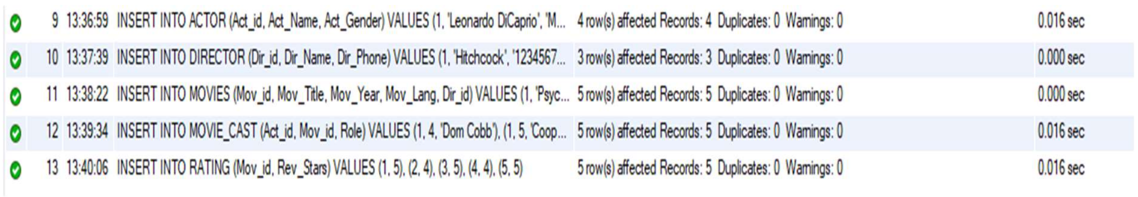
```
CREATE TABLE ACTOR (
    Act_id INT PRIMARY KEY,
    Act_Name VARCHAR(100) NOT NULL,
    Act_Gender VARCHAR(10)
);

CREATE TABLE DIRECTOR (
    Dir_id INT PRIMARY KEY,
    Dir_Name VARCHAR(100) NOT NULL,
    Dir_Phone VARCHAR(15)
);

CREATE TABLE MOVIES (
    Mov_id INT PRIMARY KEY,
    Mov_Title VARCHAR(200) NOT NULL,
    Mov_Year INT,
    Mov_Lang VARCHAR(50),
    Dir_id INT,
    FOREIGN KEY (Dir_id) REFERENCES
DIRECTOR(Dir_id)
);

CREATE TABLE MOVIE_CAST (
    Act_id INT,
    Mov_id INT,
    Role VARCHAR(100),
    PRIMARY KEY (Act_id, Mov_id),
    FOREIGN KEY (Act_id) REFERENCES
ACTOR(Act_id),
    FOREIGN KEY (Mov_id) REFERENCES
MOVIES(Mov_id)
);

CREATE TABLE RATING (
    Mov_id INT,
    Rev_Stars INT CHECK (Rev_Stars BETWEEN
1 AND 5),
    PRIMARY KEY (Mov_id, Rev_Stars),
    FOREIGN KEY (Mov_id) REFERENCES
MOVIES(Mov_id)
);
```



```
4  13:32:09  CREATE TABLE ACTOR (   Act_id INT PRIMARY KEY,   Act_Name VARCHAR(100) NOT N...  0 row(s) affected  0.047 sec
5  13:33:26  CREATE TABLE DIRECTOR (   Dir_id INT PRIMARY KEY,   Dir_Name VARCHAR(100) NO...  0 row(s) affected  0.016 sec
6  13:34:06  CREATE TABLE MOVIES (   Mov_id INT PRIMARY KEY,   Mov_Title VARCHAR(200) NOT ...  0 row(s) affected  0.031 sec
7  13:34:41  CREATE TABLE MOVIE_CAST (   Act_id INT,   Mov_id INT,   Role VARCHAR(100),   PRI...  0 row(s) affected  0.046 sec
8  13:35:21  CREATE TABLE RATING (   Mov_id INT,   Rev_Stars INT CHECK (Rev_Stars BETWEEN 1...  0 row(s) affected  0.031 sec
```

Inserting values in tables

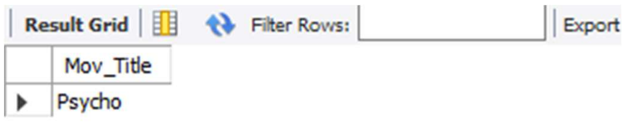
```
INSERT INTO ACTOR (Act_id, Act_Name,
Act_Gender) VALUES
(1, 'Leonardo DiCaprio', 'Male'),
(2, 'Kate Winslet', 'Female'),
(3, 'Morgan Freeman', 'Male'),
(4, 'Tom Hanks', 'Male');

INSERT INTO DIRECTOR (Dir_id, Dir_Name,
Dir_Phone) VALUES
(1, 'Hitchcock', '1234567890'),
(2, 'Steven Spielberg', '9876543210'),
(3, 'Christopher Nolan', '4561237890');

INSERT INTO MOVIES (Mov_id, Mov_Title,
Mov_Year, Mov_Lang, Dir_id) VALUES
(1, 'Psycho', 1960, 'English', 1),
(2, 'Jaws', 1975, 'English', 2),
(3, 'E.T.', 1982, 'English', 2),
(4, 'Inception', 2010, 'English', 3),
(5, 'Interstellar', 2014, 'English', 3);

INSERT INTO MOVIE_CAST (Act_id, Mov_id,
Role) VALUES
(1, 4, 'Dom Cobb'),
(1, 5, 'Cooper'),
(2, 1, 'Marion Crane'),
(3, 2, 'Quint'),
(4, 3, 'Elliott');

INSERT INTO RATING (Mov_id, Rev_Stars)
VALUES
(1, 5),
(2, 4),
(3, 5),
(4, 4),
(5, 5);
```



```
9  13:36:59  INSERT INTO ACTOR (Act_id, Act_Name, Act_Gender) VALUES (1, 'Leonardo DiCaprio', 'M...  4 row(s) affected Records: 4  Duplicates: 0  Warnings: 0  0.016 sec
10 13:37:39  INSERT INTO DIRECTOR (Dir_id, Dir_Name, Dir_Phone) VALUES (1, 'Hitchcock', '1234567...  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0  0.000 sec
11 13:38:22  INSERT INTO MOVIES (Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id) VALUES (1, 'Psyc...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.000 sec
12 13:39:34  INSERT INTO MOVIE_CAST (Act_id, Mov_id, Role) VALUES (1, 4, 'Dom Cobb'), (1, 5, 'Coop...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.016 sec
13 13:40:06  INSERT INTO RATING (Mov_id, Rev_Stars) VALUES (1, 5), (2, 4), (3, 5), (4, 4), (5, 5)     5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.016 sec
```

1. List the titles of all movies directed by 'Hitchcock'.

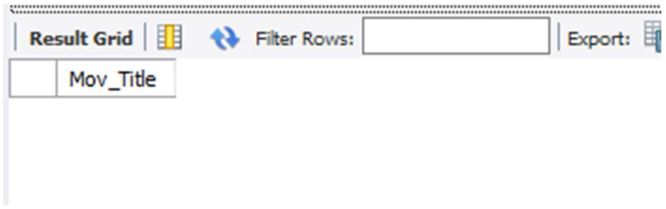
```
SELECT Mov_Title
FROM MOVIES
JOIN DIRECTOR ON MOVIES.Dir_id =
DIRECTOR.Dir_id
WHERE Dir_Name = 'Hitchcock';
```



Mov_Title
Psycho

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

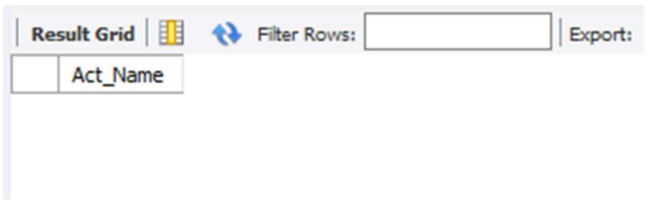
```
SELECT DISTINCT M.Mov_Title
FROM MOVIES M
JOIN MOVIE_CAST MC ON M.Mov_id = MC.Mov_id
WHERE MC.Act_id IN (
    SELECT Act_id
    FROM MOVIE_CAST
    GROUP BY Act_id, Mov_id
    HAVING COUNT(DISTINCT Mov_id) > 1
);
```



Mov_Title

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

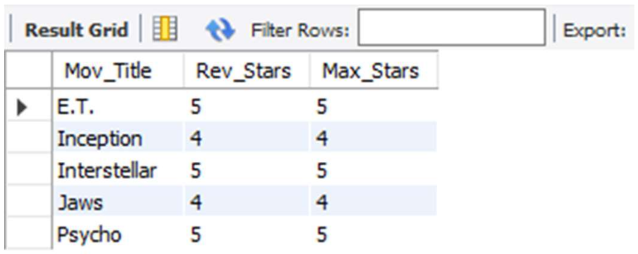
```
SELECT DISTINCT A.Act_Name
FROM ACTOR A
JOIN MOVIE_CAST MC1 ON A.Act_id =
MC1.Act_id
JOIN MOVIES M1 ON MC1.Mov_id = M1.Mov_id
JOIN MOVIE_CAST MC2 ON A.Act_id =
MC2.Act_id
JOIN MOVIES M2 ON MC2.Mov_id = M2.Mov_id
WHERE M1.Mov_Year < 2000 AND M2.Mov_Year >
2015;
```



	Act_Name

4. **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.**

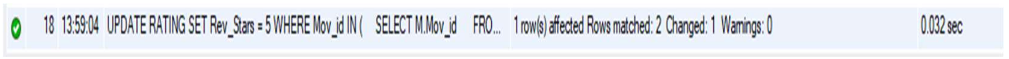
```
SELECT M.Mov_Title, R.Rev_Stars,
MAX(R.Rev_Stars) OVER (PARTITION BY
M.Mov_id) AS Max_Stars
FROM MOVIES M
JOIN RATING R ON M.Mov_id = R.Mov_id
ORDER BY M.Mov_Title;
```


Result Grid | Filter Rows: | Export:

Mov_Title	Rev_Stars	Max_Stars
E.T.	5	5
Inception	4	4
Interstellar	5	5
Jaws	4	4
Psycho	5	5

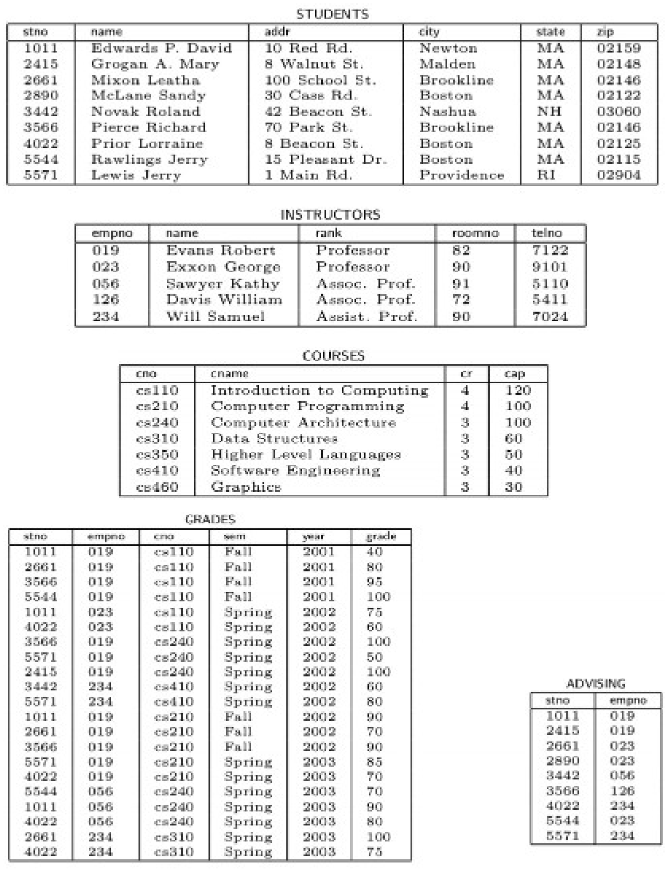
5. **Update the rating of all movies directed by 'Steven Spielberg' to 5.**

```
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'
);
```



```
18  13:59:04  UPDATE RATING SET Rev_Stars = 5 WHERE Mov_id IN (   SELECT M.Mov_id    FRO...  1 row(s) affected Rows matched: 2  Changed: 1  Warnings: 0   0.032 sec
```

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



STUDENTS

stno	name	addr	city	state	zip
1011	Edwards P. David	10 Red Rd.	Newton	MA	02159
2415	Grogan A. Mary	8 Walnut St.	Malden	MA	02148
2661	Mixon Leatha	100 School St.	Brookline	MA	02146
2890	McLane Sandy	30 Cass Rd.	Boston	MA	02122
3442	Novak Roland	42 Beacon St.	Nashua	NH	03060
3566	Pierce Richard	70 Park St.	Brookline	MA	02146
4022	Prior Lorraine	8 Beacon St.	Boston	MA	02125
5544	Rawlings Jerry	15 Pleasant Dr.	Boston	MA	02115
5571	Lewis Jerry	1 Main Rd.	Providence	RI	02904

INSTRUCTORS

empno	name	rank	roomno	telno
019	Evans Robert	Professor	82	7122
023	Exxon George	Professor	90	9101
056	Sawyer Kathy	Assoc. Prof.	91	5110
126	Davis William	Assoc. Prof.	72	5411
234	Will Samuel	Assist. Prof.	90	7024

COURSES

cno	cname	cr	cap
cs110	Introduction to Computing	4	120
cs210	Computer Programming	4	100
cs240	Computer Architecture	3	100
cs310	Data Structures	3	60
cs350	Higher Level Languages	3	50
cs410	Software Engineering	3	40
cs460	Graphics	3	30

GRADES

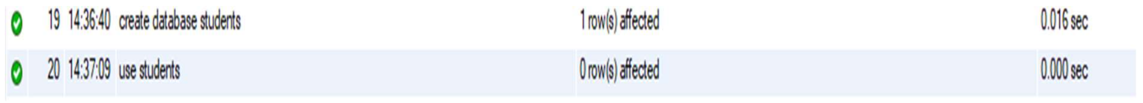
stno	empno	cno	sem	year	grade
1011	019	cs110	Fall	2001	40
2661	019	cs110	Fall	2001	80
3566	019	cs110	Fall	2001	95
5544	019	cs110	Fall	2001	100
1011	023	cs110	Spring	2002	75
4022	023	cs110	Spring	2002	60
3566	019	cs240	Spring	2002	100
5571	019	cs240	Spring	2002	50
2415	019	cs240	Spring	2002	100
3442	234	cs410	Spring	2002	60
5571	234	cs410	Spring	2002	80
1011	019	cs210	Fall	2002	90
2661	019	cs210	Fall	2002	70
3566	019	cs210	Fall	2002	90
5571	019	cs210	Spring	2003	85
4022	019	cs210	Spring	2003	70
5544	056	cs240	Spring	2003	70
1011	056	cs240	Spring	2003	90
4022	056	cs240	Spring	2003	80
2661	234	cs310	Spring	2003	100
4022	234	cs310	Spring	2003	75

ADVISING

stno	empno
1011	019
2415	019
2661	023
2890	023
3442	056
3566	126
4022	234
5544	023
5571	234

Creating database and using it:

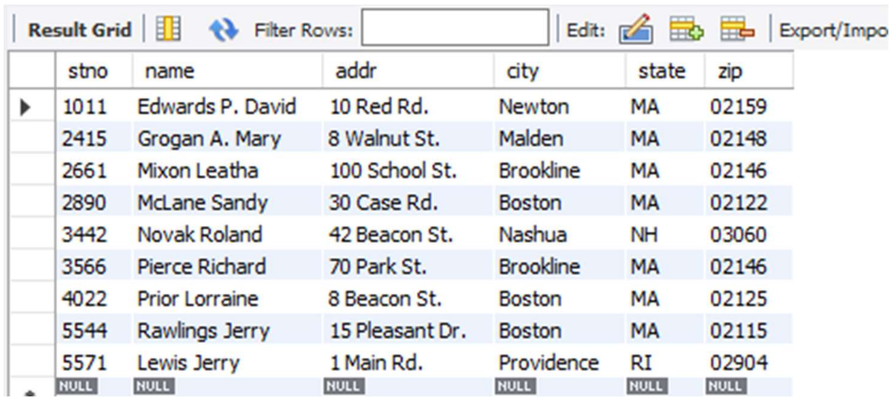
```
create database students;
use students;
```



```
19  14:36:40  create database students  1 row(s) affected  0.016 sec
20  14:37:09  use students              0 row(s) affected  0.000 sec
```

Creating table students and inserting values:

```
CREATE TABLE STUDENTS (

    stno INT PRIMARY KEY,

    name VARCHAR(100),

    addr VARCHAR(100),

    city VARCHAR(50),

    state CHAR(2),
    zip CHAR(5)
);
INSERT INTO STUDENTS (stno, name, addr, city,
state, zip) VALUES
(1011, 'Edwards P. David', '10 Red Rd.',
'Newton', 'MA', '02159'),
(2415, 'Grogan A. Mary', '8 Walnut St.',
'Malden', 'MA', '02148'),
(2661, 'Mixon Leatha', '100 School St.',
'Brookline', 'MA', '02146'),
(2890, 'McLane Sandy', '30 Case Rd.', 'Boston',
'MA', '02122'),
(3442, 'Novak Roland', '42 Beacon St.',
'Nashua', 'NH', '03060'),
(3566, 'Pierce Richard', '70 Park St.',
'Brookline', 'MA', '02146'),
(4022, 'Prior Lorraine', '8 Beacon St.',
'Boston', 'MA', '02125'),
(5544, 'Rawlings Jerry', '15 Pleasant Dr.',
'Boston', 'MA', '02115'),
(5571, 'Lewis Jerry', '1 Main Rd.',
'Providence', 'RI', '02904');
Select * from students;
```

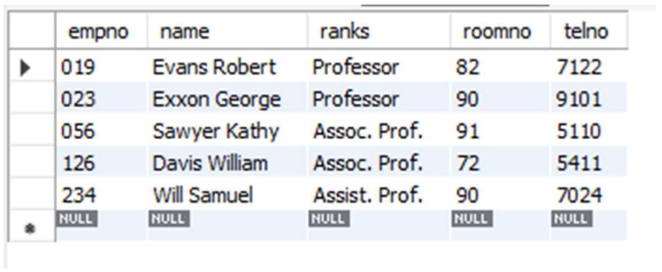


stno	name	addr	city	state	zip
1011	Edwards P. David	10 Red Rd.	Newton	MA	02159
2415	Grogan A. Mary	8 Walnut St.	Malden	MA	02148
2661	Mixon Leatha	100 School St.	Brookline	MA	02146
2890	McLane Sandy	30 Case Rd.	Boston	MA	02122
3442	Novak Roland	42 Beacon St.	Nashua	NH	03060
3566	Pierce Richard	70 Park St.	Brookline	MA	02146
4022	Prior Lorraine	8 Beacon St.	Boston	MA	02125
5544	Rawlings Jerry	15 Pleasant Dr.	Boston	MA	02115
5571	Lewis Jerry	1 Main Rd.	Providence	RI	02904
NULL	NULL	NULL	NULL	NULL	NULL

Creating table instructors and inserting values:

```
CREATE TABLE INSTRUCTORS (
    empno CHAR(3) PRIMARY KEY,
    name VARCHAR(100),
    ranks VARCHAR(50),
    roomno INT,
    telno CHAR(4)
);
INSERT INTO INSTRUCTORS (empno, name, ranks,
roomno, telno) VALUES
('019', 'Evans Robert', 'Professor', 82,
'7122'),
('023', 'Exxon George', 'Professor', 90,
'9101'),
('056', 'Sawyer Kathy', 'Assoc. Prof.', 91,
'5110'),
('126', 'Davis William', 'Assoc. Prof.', 72,
'5411'),
('234', 'Will Samuel', 'Assist. Prof.', 90,
'7024');
```

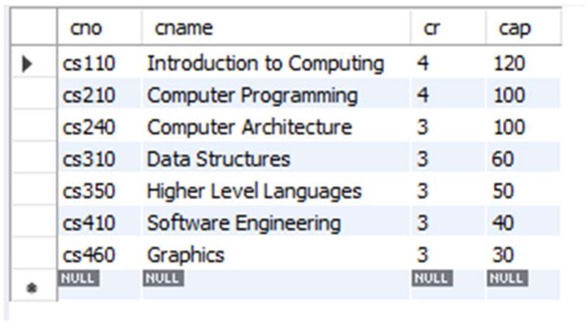
```
select * from instructors;
```



empno	name	ranks	roomno	telno
019	Evans Robert	Professor	82	7122
023	Exxon George	Professor	90	9101
056	Sawyer Kathy	Assoc. Prof.	91	5110
126	Davis William	Assoc. Prof.	72	5411
234	Will Samuel	Assist. Prof.	90	7024
NULL	NULL	NULL	NULL	NULL

Create table courses and insert values:

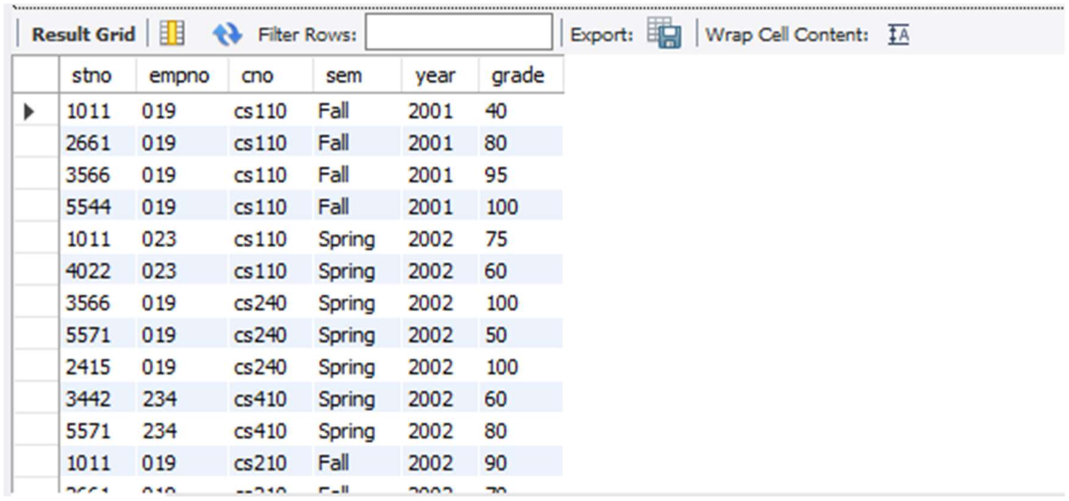
```
CREATE TABLE COURSES (

    cno CHAR(5) PRIMARY KEY,

    cname VARCHAR(100),

    cr INT,

    cap INT

);

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

('cs110', 'Introduction to Computing', 4, 120),

('cs210', 'Computer Programming', 4, 100),

('cs240', 'Computer Architecture', 3, 100),

('cs310', 'Data Structures', 3, 60),

('cs350', 'Higher Level Languages', 3, 50),

('cs410', 'Software Engineering', 3, 40),

('cs460', 'Graphics', 3, 30);

select * from courses;
```



cno	cname	cr	cap
cs110	Introduction to Computing	4	120
cs210	Computer Programming	4	100
cs240	Computer Architecture	3	100
cs310	Data Structures	3	60
cs350	Higher Level Languages	3	50
cs410	Software Engineering	3	40
cs460	Graphics	3	30
NULL	NULL	NULL	NULL

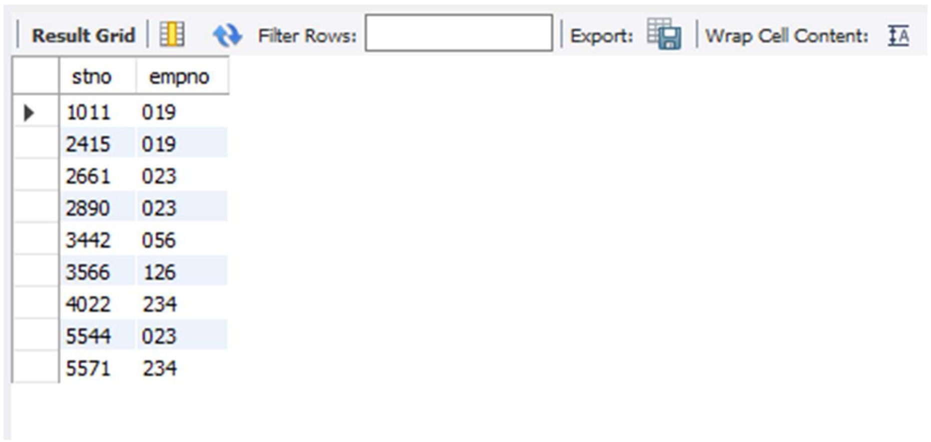
Create table and insert values:

```
CREATE TABLE GRADES (
    stno INT,
    empno CHAR(3),
    cno CHAR(5),
    sem VARCHAR(10),
    year INT,
    grade INT
);

INSERT INTO GRADES (stno, empno, cno, sem,
year, grade) VALUES
(1011, '019', 'cs110', 'Fall', 2001, 40),
(2661, '019', 'cs110', 'Fall', 2001, 80),
(3566, '019', 'cs110', 'Fall', 2001, 95),
(5544, '019', 'cs110', 'Fall',2001, 100),
(1011, '023', 'cs110', 'Spring', 2002, 75),
(4022, '023', 'cs110', 'Spring', 2002, 60),
(3566, '019', 'cs240', 'Spring', 2002, 100),
(5571, '019', 'cs240', 'Spring', 2002, 50),
(2415, '019', 'cs240', 'Spring', 2002, 100),
(3442, '234', 'cs410', 'Spring', 2002, 60),
(5571, '234', 'cs410', 'Spring', 2002, 80),
(1011, '019', 'cs210', 'Fall', 2002, 90),
(2661, '019', 'cs210', 'Fall', 2002, 70),
(3566, '019', 'cs210', 'Fall', 2002, 90),
(5571, '019', 'cs210', 'Spring', 2003, 85),
(4022, '019', 'cs210', 'Spring', 2003, 70),
(5544, '056', 'cs240', 'Spring', 2003, 70),
(1011, '056', 'cs240', 'Spring', 2003, 90),
(4022, '056', 'cs240', 'Spring', 2003, 80),
(2661, '234', 'cs310', 'Spring', 2003, 100),
(4022, '234', 'cs310', 'Spring', 2003, 75);
select * from GRADES;
```



stno	empno	cno	sem	year	grade
1011	019	cs110	Fall	2001	40
2661	019	cs110	Fall	2001	80
3566	019	cs110	Fall	2001	95
5544	019	cs110	Fall	2001	100
1011	023	cs110	Spring	2002	75
4022	023	cs110	Spring	2002	60
3566	019	cs240	Spring	2002	100
5571	019	cs240	Spring	2002	50
2415	019	cs240	Spring	2002	100
3442	234	cs410	Spring	2002	60
5571	234	cs410	Spring	2002	80
1011	019	cs210	Fall	2002	90

```
CREATE TABLE ADVISING (
    stno INT,
    empno CHAR(3)
);

INSERT INTO ADVISING (stno, empno) VALUES
(1011, '019'),
(2415, '019'),
(2661, '023'),
(2890, '023'),
(3442, '056'),
(3566, '126'),
(4022, '234'),
(5544, '023'),
(5571, '234');
select * from ADVISING;
```

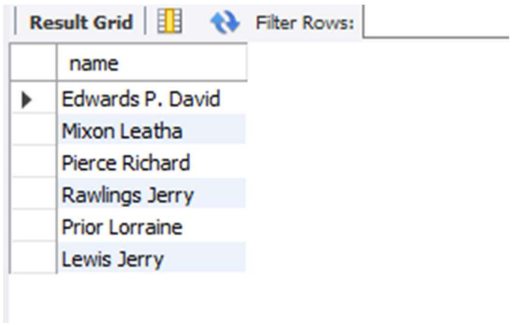


stno	empno
1011	019
2415	019
2661	023
2890	023
3442	056
3566	126
4022	234
5544	023
5571	234

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

   ```
   SELECT DISTINCT s.name
   FROM STUDENTS s
   JOIN GRADES g ON s.stno = g.stno
   JOIN COURSES c ON g.cno = c.cno
   ```

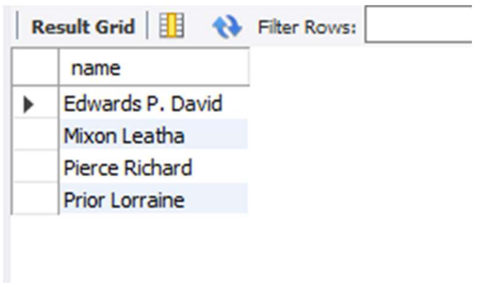
```
WHERE c.cr = 4;
```


Result Grid Filter Rows:

```
name
Edwards P. David
Mixon Leatha
Pierce Richard
Rawlings Jerry
Prior Lorraine
Lewis Jerry
```

2. **Find the names of students who took every four-credit course.**

```
SELECT 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
GROUP BY s.stno, s.name
HAVING COUNT(DISTINCT c.cno) = (SELECT
COUNT(DISTINCT cno) FROM COURSES WHERE cr =
4);
```

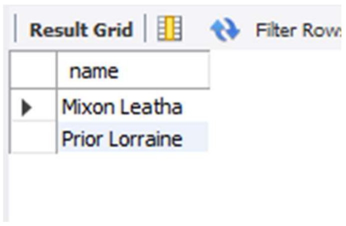

Result Grid Filter Rows:

```
name
Edwards P. David
Mixon Leatha
Pierce Richard
Prior Lorraine
```

3. **Find the names of students who took cs210 and cs310.**

```
SELECT s.name
FROM STUDENTS s
```

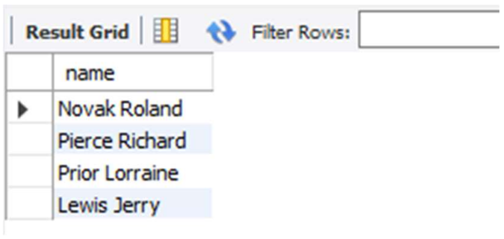
```
JOIN GRADES g ON s.stno = g.stno
WHERE g.cno IN ('cs210', 'cs310')
GROUP BY s.name
HAVING COUNT(DISTINCT g.cno) = 2;
```



```
name
Mixon Leatha
Prior Lorraine
```

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

```
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.ranks != 'Professor';
```

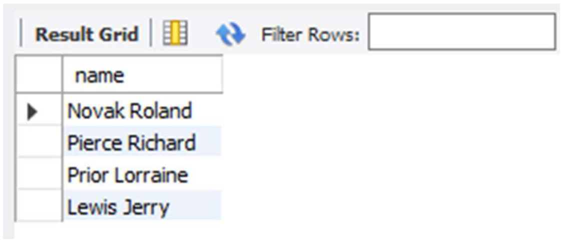


```
name
Novak Roland
Pierce Richard
Prior Lorraine
Lewis Jerry
```

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

```
SELECT DISTINCT i1.name
FROM INSTRUCTORS i1
JOIN COURSES c ON i1.empno = c.empno
JOIN GRADES g ON c.cno = g.cno
JOIN ADVISING a ON g.stno = a.stno
JOIN INSTRUCTORS i2 ON a.empno = i2.empno
```

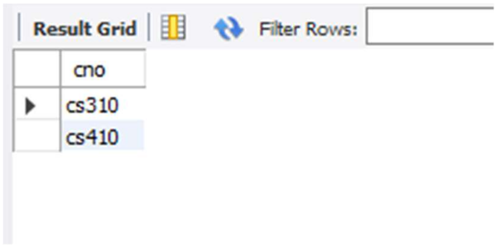
```
WHERE i1.roomno = i2.roomno AND i1.empno !=
i2.empno;
```



name
Novak Roland
Pierce Richard
Prior Lorraine
Lewis Jerry

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

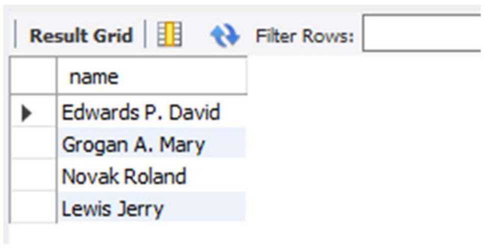
```
SELECT cno
FROM GRADES
GROUP BY cno
HAVING COUNT(DISTINCT stno) = 2;
```



cno
cs310
cs410

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

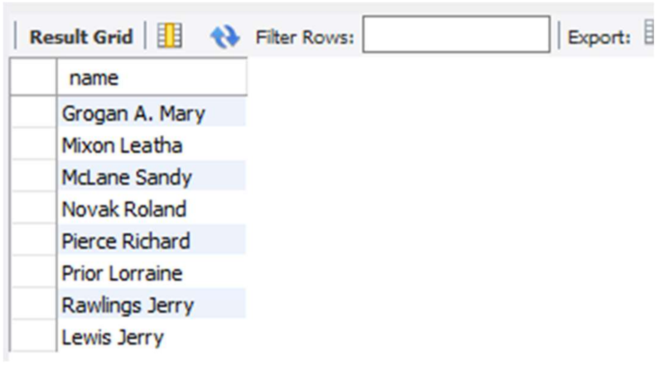
```
SELECT s1.name
FROM STUDENTS s1
WHERE NOT EXISTS (
    SELECT 1
    FROM STUDENTS s2
    WHERE s1.city = s2.city AND s1.stno !=
s2.stno
);
```



```
name
Edwards P. David
Grogan A. Mary
Novak Roland
Lewis Jerry
```

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

```
SELECT s1.name
FROM STUDENTS s1
WHERE NOT EXISTS (
    SELECT g1.cno
    FROM GRADES g1
    JOIN STUDENTS s2 ON g1.stno = s2.stno
    WHERE s2.name = 'Richard Pierce'
    AND g1.cno NOT IN (
        SELECT g2.cno
        FROM GRADES g2
        WHERE g2.stno = s1.stno
    )
);
```

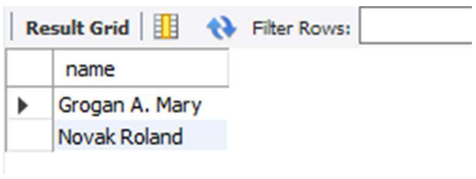


```
name
Grogan A. Mary
Mixon Leatha
McLane Sandy
Novak Roland
Pierce Richard
Prior Lorraine
Rawlings Jerry
Lewis Jerry
```

9. **Find the names of students who took only one course.**

```
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;
```



name
Grogan A. Mary
Novak Roland