NAME: AAMINA KHAN ROLLNO:033 Msc(DSAI)

Practical-2: Subquery-join operations on Relational Schema

Date:-20/12/2024 **Submission Date:- 03/12/2024**

Write-up: -

- DBMS Architecture
- DML
- DCL
- ERD
- Components of ERD
- Relationships in ERD
- SUBQUERY and Types of Subquery
- GROUPBY and HAVING
- Join and Types of join

1.USING (practical 1)

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

```
SELECT COUNT(*) AS customer_count
   -> FROM customer
   -> WHERE grade > (SELECT AVG(grade) FROM customer WHERE city = 'Bangalore');
+----------------+
| customer_count |
+----------------+
|              0 |
+----------------+
1 row in set (0.76 sec)
```

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

```
 SELECT saleman_id, COUNT(DISTINCT customer_id) AS customer_count
   -> FROM `order`
   -> GROUP BY saleman_id
   -> HAVING customer_count > 1;
+------------+----------------+
| saleman_id | customer_count |
+------------+----------------+
|       NULL |              2 |
|       5001 |              2 |
|       5002 |              2 |
+------------+----------------+
3 rows in set (0.17 sec)
```

3. List all salesmen and indicate those who have and don’t have customers in their cities

(Use UNION operation.)

```
SELECT s.saleman_id, 'Has Customers' AS status
   -> FROM salesman s
   -> JOIN `order` o ON s.saleman_id = o.saleman_id
   -> JOIN customer c ON o.customer_id = c.customer_id
   -> WHERE s.city = c.city
   -> GROUP BY s.saleman_id
   -> UNION
   -> -- Salesmen without customers in their cities
   -> SELECT s.saleman_id, 'No Customers' AS status
   -> FROM salesman s
   -> WHERE NOT EXISTS (
   ->     SELECT 1
   ->     FROM `order` o
   ->     JOIN customer c ON o.customer_id = c.customer_id
   ->     WHERE s.saleman_id = o.saleman_id AND s.city = c.city
   -> );
+------------+--------------+
| saleman_id | status       |
+------------+--------------+
|       5001 | Has Customers |
|       5006 | Has Customers |
|       5002 | No Customers  |
|       5003 | No Customers  |
|       5005 | No Customers  |
|       5007 | No Customers  |
+------------+--------------+
6 rows in set (0.35 sec)
```

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.saleman_id,
   ->        o.customer_id,
   ->        o.order_date,
   ->        o.purch_amt AS highest_purchase_amount
   -> FROM `order` o
   -> WHERE o.purch_amt = (
   ->     SELECT MAX(purch_amt)
   ->     FROM `order`
   ->     WHERE order_date = o.order_date
   -> );
Query OK, 0 rows affected (0.28 sec)
mysql> SELECT * FROM highest_order_salesman;
+------------+-------------+------------+-----------------------+
| saleman_id | customer_id | order_date | highest_purchase_amount |
+------------+-------------+------------+-----------------------+
|       5002 |        3005 | 2016-10-05 |                150.50 |
|       NULL |        3009 | 2016-10-10 |               2480.40 |
|       NULL |        3009 | 2016-08-17 |                110.50 |
|       5001 |        3007 | 2016-07-27 |               2400.60 |
```

```
|       5001 |        3002 | 2016-09-10 |                5760.00 |
|       5002 |        3008 | 2016-06-27 |                 250.45 |
+------------+-------------+------------+------------------------+
6 rows in set (0.01 sec)
```

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

```
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)

Write SQL queries to
1. List the titles of all movies directed by 'Hitchcock'.

```
SELECT MOVIES.MOV_TITLE
   -> FROM MOVIES
   -> JOIN DIRECTOR ON MOVIES.DIR_ID = DIRECTOR.DIR_ID
   -> WHERE DIRECTOR.DIR_NAME = 'HITCHCOCK';
+-----------+
| MOV_TITLE |
+-----------+
| AKASH     |
+-----------+
1 row in set (0.01 sec)
```

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

```
SELECT MOVIES.MOV_TITLE
   -> FROM MOVIES
   -> JOIN MOVIE_CAST ON MOVIES.MOV_ID = MOVIE_CAST.MOV_ID
   -> JOIN ACTOR ON MOVIE_CAST.ACT_ID = ACTOR.ACT_ID
   -> GROUP BY MOVIES.MOV_TITLE
   -> HAVING COUNT(DISTINCT MOVIE_CAST.MOV_ID) >= 2;
Empty set (0.02 sec)
```

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

```
mysql> SELECT DISTINCT ACTOR.ACT_NAME
   -> FROM ACTOR
   -> JOIN MOVIE_CAST ON ACTOR.ACT_ID = MOVIE_CAST.ACT_ID
   -> JOIN MOVIES ON MOVIE_CAST.MOV_ID = MOVIES.MOV_ID
```

```
   -> WHERE (MOVIES.MOV_YEAR < 2000) AND (ACTOR.ACT_ID IN (
   ->     SELECT ACTOR.ACT_ID
   ->     FROM ACTOR
   ->     JOIN MOVIE_CAST ON ACTOR.ACT_ID = MOVIE_CAST.ACT_ID
   ->     JOIN MOVIES ON MOVIE_CAST.MOV_ID = MOVIES.MOV_ID
   ->     WHERE MOVIES.MOV_YEAR > 2015
   -> ));
Empty set (0.04 sec)
```

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 MOVIES.MOV_TITLE, MAX(RATING.REV_STARS) AS HIGHEST_RATING
   -> FROM MOVIES
   -> JOIN RATING ON MOVIES.MOV_ID = RATING.MOV_ID
   -> GROUP BY MOVIES.MOV_TITLE
   -> HAVING COUNT(RATING.REV_STARS) > 0
   -> ORDER BY MOVIES.MOV_TITLE;
+------------+----------------+
| MOV_TITLE  | HIGHEST_RATING |
+------------+----------------+
| AKASH      |              5 |
| BAHUBALI-1 |              2 |
| BAHUBALI-2 |              4 |
| WAR HORSE  |              4 |
+------------+----------------+
4 rows in set (0.00 sec)
```

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

```
UPDATE RATING
   -> SET REV_STARS = '5'
   -> WHERE MOV_ID IN (
   ->     SELECT MOV_ID
   ->     FROM MOVIES
   ->     JOIN DIRECTOR ON MOVIES.DIR_ID = DIRECTOR.DIR_ID
   ->     WHERE DIRECTOR.DIR_NAME = 'STEVEN SPIELBERG'
   -> );
Query OK, 1 row affected (0.21 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

IF YOU WANT TO CEHCK THAT THE UPDATION IS DONE IN THAT TABLE USE THIS CODE :

```
SELECT MOVIES.MOV_TITLE, RATING.REV_STARS
   -> FROM MOVIES
   -> JOIN DIRECTOR ON MOVIES.DIR_ID = DIRECTOR.DIR_ID
   -> JOIN RATING ON MOVIES.MOV_ID = RATING.MOV_ID
   -> WHERE DIRECTOR.DIR_NAME = 'STEVEN SPIELBERG';
+-----------+-----------+
| MOV_TITLE | REV_STARS |
+-----------+-----------+
```

```
| WAR HORSE |         5 |
+-----------+-----------+
1 row in set (0.00 sec)
```

If you want to see all the ratings in the RATING table (to confirm the updates),

```
SELECT MOVIES.MOV_TITLE, RATING.REV_STARS
    -> FROM MOVIES
    -> JOIN RATING ON MOVIES.MOV_ID = RATING.MOV_ID
    -> ORDER BY MOVIES.MOV_TITLE;
+------------+-----------+
| MOV_TITLE  | REV_STARS |
+------------+-----------+
| AKASH      |         5 |
| BAHUBALI-1 |         2 |
| BAHUBALI-2 |         4 |
| WAR HORSE  |         5 |
+------------+-----------+
4 rows in set (0.00 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

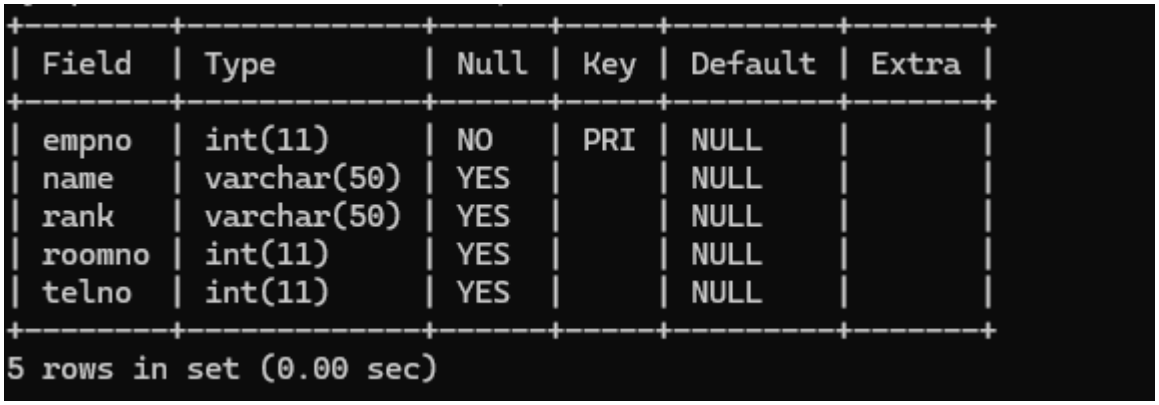
ADVISING

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

```
mysql> CREATE TABLE students (
    -> stno INT PRIMARY KEY,
    -> name VARCHAR(50),
    -> addr VARCHAR(255),
    -> city VARCHAR(50),
    -> state VARCHAR(2),
    -> zip VARCHAR(10));
Query OK, 0 rows affected (1.05 sec)
```

```
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| stno  | int(11)      | NO   | PRI | NULL    |       |
| name  | varchar(100) | YES  |     | NULL    |       |
| addr  | varchar(100) | YES  |     | NULL    |       |
| city  | varchar(50)  | YES  |     | NULL    |       |
| state | varchar(20)  | YES  |     | NULL    |       |
| zip   | int(11)      | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

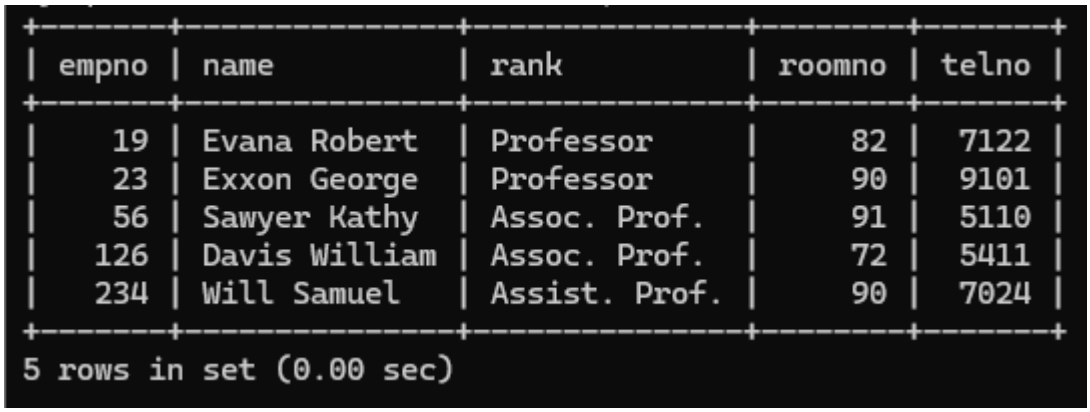
```
INSERT INTO students (stno, name, addr, city, state, zip) VALUES
    -> (1011, 'Edwards P. David', '10 Red Rd.', 'Newton', 'MA', '020159'),
    -> (2415, 'Grogan A. Mary', '8 Walnut St.', 'Malden', 'MA', '02148'),
    -> (2661, 'Mixon Leatha', '100 School St.', 'Brookline', 'MA', '02146'),
    -> (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');
Query OK, 9 rows affected (0.37 sec)
Records: 9  Duplicates: 0  Warnings: 0

CREATE TABLE INSTRUCTORS (
    -> empno INT PRIMARY KEY,
    -> name VARCHAR(50),
    -> `rank` VARCHAR(20),
    -> roomno VARCHAR(10),
    -> telno VARCHAR(15));
Query OK, 0 rows affected (1.43 sec)
```



Field	Type	Null	Key	Default	Extra
empno	int(11)	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
rank	varchar(50)	YES		NULL	
roomno	int(11)	YES		NULL	
telno	int(11)	YES		NULL	

5 rows in set (0.00 sec)

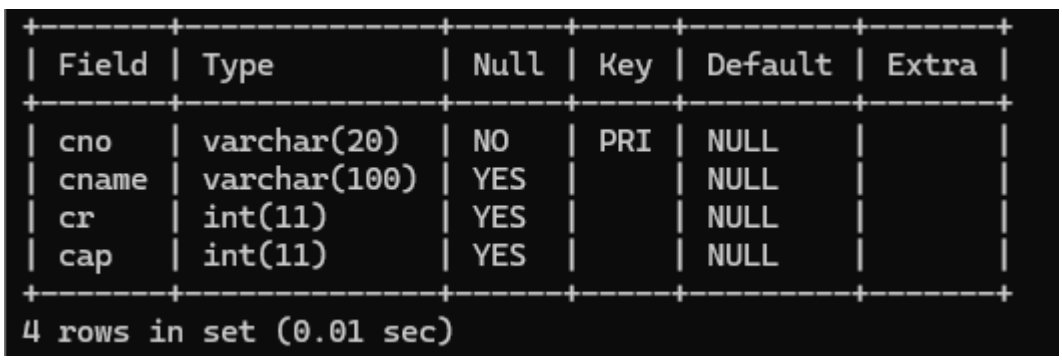
INSERT INTO INSTRUCTORS (empno, name, `rank`, roomno, telno) VALUES
-> ('019', 'Evana Robert', 'Professor', 82, 7122),
-> ('023', 'Exxon George', 'Professor', 90, 9101),
-> ('056', 'Sawyer Kathy', 'Assoc. Prof.', 91, 5110),
-> ('126', 'Davis William', 'Assoc. Prof.', 72, 5411),
-> ('234', 'Will Samuel', 'Assist. Prof.', 90, 7024);
Query OK, 5 rows affected (0.15 sec)
Records: 5 Duplicates: 0 Warnings: 0



empno	name	rank	roomno	telno
19	Evana Robert	Professor	82	7122
23	Exxon George	Professor	90	9101
56	Sawyer Kathy	Assoc. Prof.	91	5110
126	Davis William	Assoc. Prof.	72	5411
234	Will Samuel	Assist. Prof.	90	7024

5 rows in set (0.00 sec)

CREATE TABLE COURSES (
-> cno INT PRIMARY KEY,
-> cname VARCHAR(50),
-> cr INT,
-> cap INT);
Query OK, 0 rows affected (1.16 sec)



Field	Type	Null	Key	Default	Extra
cno	varchar(20)	NO	PRI	NULL	
cname	varchar(100)	YES		NULL	
cr	int(11)	YES		NULL	
cap	int(11)	YES		NULL	

4 rows in set (0.01 sec)

```
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);
Query OK, 7 rows affected (0.11 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

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

7 rows in set (0.00 sec)

```
CREATE TABLE GRADES (
    ->     stno INT,
    ->     empno INT,
    ->     cno VARCHAR(20),
    ->     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));
Query OK, 0 rows affected (1.42 sec)
```

Field	Type	Null	Key	Default	Extra
stno	int(11)	YES	MUL	NULL	
empno	int(11)	YES	MUL	NULL	
cno	varchar(20)	YES	MUL	NULL	
sem	varchar(50)	YES		NULL	
year	int(11)	YES		NULL	
grade	int(11)	YES		NULL	

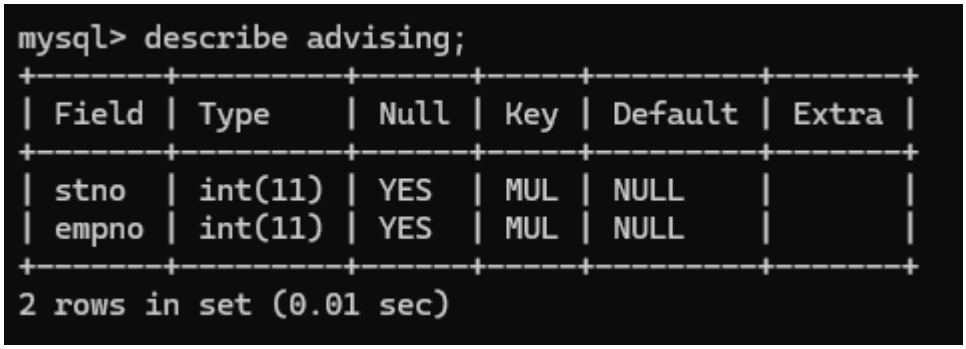
6 rows in set (0.01 sec)

```
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);
Query OK, 21 rows affected (0.12 sec)
Records: 21  Duplicates: 0  Warnings: 0
```

stno	empno	cno	sem	year	grade
1011	19	cs110	Fall	2001	40
2661	19	cs110	Fall	2001	80
3566	19	cs110	Fall	2001	95
5544	19	cs110	Fall	2001	100
1011	23	cs110	Spring	2002	75
4022	23	cs110	Spring	2002	60
3566	19	cs240	Spring	2002	100
5571	19	cs240	Spring	2002	50
2415	19	cs240	Spring	2002	100
3442	234	cs410	Spring	2002	60
5571	234	cs410	Spring	2002	80
1011	19	cs210	Fall	2002	90
2661	19	cs210	Fall	2002	70
3566	19	cs210	Fall	2002	90
5571	19	cs210	Spring	2003	85
4022	19	cs210	Spring	2003	70
5544	56	cs240	Spring	2003	70
1011	56	cs240	Spring	2003	90
4022	56	cs240	Spring	2003	80
2661	234	cs310	Spring	2003	100
4022	234	cs310	Spring	2003	75

21 rows in set (0.00 sec)

```
CREATE TABLE ADVISING (
   -> stno INT,
   -> empno INT,
   -> PRIMARY KEY (stno, empno),
   -> FOREIGN KEY (stno) REFERENCES students(stno),
   -> FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno));
Query OK, 0 rows affected (1.28 sec)
```



```
mysql> describe advising;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| stno  | int(11) | YES  | MUL | NULL    |       |
| empno | int(11) | YES  | MUL | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
INSERT INTO ADVISING (stno, empno)VALUES
   -> (1011, 019),
   -> (2415, 019),
   -> (2661, 023),
   -> (2890, 023),
   -> (3442, 056),
   -> (3566, 126),
   -> (4022, 234),
   -> (5544, 023),
   -> (5571, 234);
Query OK, 9 rows affected (0.07 sec)
Records: 9  Duplicates: 0  Warnings: 0

SELECT * FROM ADVISING;
+------+-------+
| stno | empno |
+------+-------+
| 1011 |    19 |
| 2415 |    19 |
| 2661 |    23 |
| 2890 |    23 |
| 5544 |    23 |
| 3442 |    56 |
| 3566 |   126 |
| 4022 |   234 |
| 5571 |   234 |
+------+-------+
9 rows in set (0.00 sec)
```

For odd rollnumbers(any 10)

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;
+------------------+
| name             |
+------------------+
| Edwards P. David |
| Mixon Leatha     |
| Pierce Richard   |
| Prior Lorraine   |
| Rawlings Jerry   |
| Lewis Jerry      |
+------------------+
6 rows in set (0.07 sec)
```

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

```
SELECT s.name
    -> FROM students s
    -> WHERE NOT EXISTS (
    ->     SELECT c.cno
    ->     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
    ->     )
    -> );
+------------------+
| name             |
+------------------+
| Edwards P. David |
| Mixon Leatha     |
| Pierce Richard   |
| Prior Lorraine   |
+------------------+
4 rows in set (0.04 sec)
```

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

```
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 a.empno = i.empno;
+-----------------+
| name            |
+-----------------+
| Edwards P. David |
| Grogan A. Mary   |
| Prior Lorraine   |
| Lewis Jerry      |
+-----------------+
4 rows in set (0.03 sec)
```

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

```
SELECT s.name
    -> FROM students s
    -> JOIN grades g1 ON s.stno = g1.stno
    -> JOIN grades g2 ON s.stno = g2.stno
    -> WHERE g1.cno = 'cs210' AND g2.cno = 'cs310';
+----------------+
| name           |
+----------------+
| Mixon Leatha   |
| Prior Lorraine |
+----------------+
2 rows in set (0.02 sec)
```

5. **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.rank != 'Professor';
+----------------+
| name           |
+----------------+
| Novak Roland   |
| Pierce Richard |
| Prior Lorraine |
| Lewis Jerry    |
+----------------+
4 rows in set (0.00 sec)
```

6. **Find instructors who taught students who are advised by another**

instructor who shares the same room.

```
SELECT DISTINCT i1.name
    -> FROM instructors i1
    -> JOIN grades g ON i1.empno = g.empno
    -> JOIN students s ON g.stno = s.stno
    -> JOIN advising a ON s.stno = a.stno
    -> JOIN instructors i2 ON a.empno = i2.empno
    -> WHERE i1.roomno = i2.roomno AND i1.empno != i2.empno;
+--------------+
| name         |
+--------------+
| Will Samuel  |
| Exxon George |
+--------------+
2 rows in set (0.01 sec)
```

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

```
SELECT g.cno
    -> FROM grades g
    -> GROUP BY g.cno
    -> HAVING COUNT(g.stno) = 2;
+-------+
| cno   |
+-------+
| cs310 |
| cs410 |
+-------+
2 rows in set (0.07 sec)
```

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

```
SELECT s.name
    -> FROM students s
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM students s2
    ->     WHERE s2.city = s.city
    ->     AND s2.stno != s.stno
    -> );
+------------------+
| name             |
+------------------+
| Edwards P. David |
| Grogan A. Mary   |
| Novak Roland     |
| Lewis Jerry      |
+------------------+
```

4 rows in set (0.00 sec)

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

```
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 = 'Assoc. Prof.';
+-------+
| cno   |
+-------+
| cs240 |
+-------+
1 row in set (0.00 sec)
```

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

```
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';
+-------+
| telno |
+-------+
| 7122  |
| 9101  |
| 5110  |
| 7024  |
+-------+
4 rows in set (0.00 sec)
```

11. Find names of students who took every course taken by Richard Pierce.
12. Find the names of students who took only one course.
13. Find the names of instructors who teach no course.
14. Find the names of the instructors who taught only one course during the spring semester of 2001.

<u>For even rollnumbers(any 10)</u>

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

```
SELECT s.name
   -> FROM students s
   -> JOIN grades g ON s.stno = g.stno
   -> JOIN courses c ON g.cno = c.cno
   -> WHERE s.stno % 2 = 0
```

```
    -> GROUP BY s.stno
    -> HAVING SUM(c.cr) = COUNT(c.cr) * 4;
Empty set (0.04 sec)
```

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

```
SELECT s.name
    -> FROM students s
    -> WHERE s.stno % 2 = 0
    -> AND NOT EXISTS (
    ->     SELECT 1
    ->     FROM grades g
    ->     JOIN courses c ON g.cno = c.cno
    ->     WHERE g.stno = s.stno AND c.cr = 4
    -> );
+--------------+
| name         |
+--------------+
| McLane Sandy |
| Novak Roland |
+--------------+
2 rows in set (0.00 sec)
```

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

```
SELECT DISTINCT s.name
    -> FROM students s
    -> JOIN grades g ON s.stno = g.stno
    -> WHERE s.stno % 2 = 0 AND g.cno IN ('cs210', 'cs310');
+----------------+
| name           |
+----------------+
| Pierce Richard |
| Prior Lorraine |
+----------------+
2 rows in set (0.00 sec)
```

4. **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.**

```
SELECT s.name
-> FROM students s
-> JOIN grades g1 ON s.stno = g1.stno
-> JOIN grades g2 ON s.stno = g2.stno
-> JOIN instructors i ON g2.empno = i.empno
-> WHERE s.stno % 2 = 0
-> AND g1.cno = 'cs210' AND g2.cno = 'cs310'
-> AND g1.grade > (SELECT MAX(grade) FROM grades WHERE cno = 'cs310')
-> AND NOT EXISTS (
```

```
    ->     SELECT 1
    ->     FROM grades g3
    ->     JOIN instructors i2 ON g3.empno = i2.empno
    ->     WHERE g3.stno = s.stno AND i2.name = 'Evana Robert'
    -> );
Empty set (0.04 sec)
```

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

```
SELECT g.cno
    -> FROM grades g
    -> WHERE g.stno % 2 = 0
    -> GROUP BY g.cno
    -> HAVING COUNT(DISTINCT g.stno) >= 2;
+-------+
| cno   |
+-------+
| cs110 |
| cs210 |
| cs240 |
+-------+
3 rows in set (0.02 sec)
```

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

```
SELECT g.cno
    -> FROM grades g
    -> WHERE g.stno % 2 = 0
    -> GROUP BY g.cno
    -> HAVING COUNT(DISTINCT g.stno) >= 3;
+-------+
| cno   |
+-------+
| cs110 |
| cs240 |
+-------+
2 rows in set (0.00 sec)
```

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

```
SELECT DISTINCT i.name
    -> FROM instructors i
    -> JOIN grades g ON i.empno = g.empno
    -> WHERE g.stno % 2 = 0
    -> AND EXISTS (
    ->     SELECT 1
    ->     FROM grades g2
    ->     JOIN instructors i2 ON g2.empno = i2.empno
```

```
    ->     WHERE g2.stno = g.stno AND i2.rank = 'Assist. Prof.'
    -> );
+--------------+
| name         |
+--------------+
| Evana Robert |
| Exxon George |
| Sawyer Kathy |
| Will Samuel  |
+--------------+
4 rows in set (0.01 sec)
```

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

```
SELECT MIN(g.grade)
    -> FROM grades g
    -> WHERE g.sem = 'Spring' AND g.year = 2003 AND g.stno % 2 = 0;
+--------------+
| MIN(g.grade) |
+--------------+
|           70 |
+--------------+
1 row in set (0.00 sec)
```

9. **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).**

```
SELECT s.name
    -> FROM students s
    -> WHERE s.stno % 2 = 0
    -> AND NOT EXISTS (
    ->     SELECT 1
    ->     FROM grades g
    ->     JOIN instructors i ON g.empno = i.empno
    ->     WHERE i.name = 'Evana Robert'
    ->     AND NOT EXISTS (
    ->         SELECT 1
    ->         FROM grades g2
    ->         WHERE g2.stno = s.stno AND g2.cno = g.cno
    ->     )
    -> );
+----------------+
| name           |
+----------------+
| Pierce Richard |
| Prior Lorraine |
+----------------+
2 rows in set (0.00 sec)
```

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

```
SELECT s.name
   -> FROM students s
   -> JOIN advising a ON s.stno = a.stno
   -> WHERE s.stno % 2 = 0
   -> AND NOT EXISTS (
   ->     SELECT 1
   ->     FROM grades g
   ->     WHERE g.stno = s.stno AND g.empno = a.empno
   -> );
+----------------+
| name           |
+----------------+
| McLane Sandy   |
| Novak Roland   |
| Pierce Richard |
| Rawlings Jerry |
+----------------+
4 rows in set (0.00 sec)
```

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

```
SELECT s.name
   -> FROM students s
   -> JOIN grades g ON s.stno = g.stno
   -> WHERE s.stno % 2 = 0
   -> GROUP BY s.stno
   -> HAVING MIN(g.grade) < 60;
Empty set (0.02 sec)
```

12. **Find the highest grade of a student who never took cs110.**
13. **Find the names of students who do not have an advisor.**
14. **Find names of courses taken by students who do not live in Massachusetts (MA).**