Mithibai College Department of Computer Science

Msc(Data Sci and AI)

Practical-2: Subquery-join operations on Relational Schema

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

1 USING (practical 1)

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

mysql> SELECT S.name

- -> 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;



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

mysql> SELECT S.name AS salesman_name, S.city AS salesman_city, 'Has Customers in City' AS status

- -> FROM salesman S
- -> JOIN customer C ON S.salesman id = C.salesman id AND S.city = C.city
- ->
- -> UNION
- ->
- -> -- Salesmen without customers in their city
- -> SELECT S.name AS salesman_name, S.city AS salesman_city, 'No Customers in City' AS status
 - -> FROM salesman S
 - -> WHERE NOT EXISTS (

- -> SELECT 1
- -> FROM customer C
- -> WHERE S.salesman id = C.salesman id AND S.city = C.city
- ->);

+		tt
salesman_name	salesman_city	status
James Hoog Mc Lyon Nail Knite Lauson Hen Pit Alex Paul Adam	New York Paris Paris London Rome	Has Customers in City Has Customers in City No Customers in City No Customers in City No Customers in City No Customers in City
6 rows in set (0	.02 sec)	++

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

mysql> CREATE VIEW TopSalesmanPerDay AS

- -> SELECT
- -> S.salesman id,
- -> S.name AS salesman_name,
- -> C.customer id,
- -> C.customer name,
- -> O.order date,
- -> O.purch amt
- -> FROM
- -> orders O
- -> JOIN
- -> customer C ON O.customer id = C.customer id
- -> JOIN
- -> salesman S ON C.salesman id = S.salesman id
- -> WHERE
- -> O.purch amt = (

- -> SELECT MAX(O1.purch_amt)
- -> FROM orders O1
- -> WHERE O1.order date = O.order date
- ->);

Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM TopSalesmanPerDay;

t	 salesman_name	ti customer id	customer name	t order date	l purch amt l
+		+		+	
5001	James Hoog	3002	Nick Rimando	2016-09-10	5760
	Nail Knite			2016-10-05	150.5
•	James Hoog		Brad Davis	2016-07-27	2400.6
5002	Nail Knite	3008	Julian Green	2016-06-27	250.45
4 rows in set ((0.00 sec)			•	-

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

mysql> DELETE FROM salesman WHERE salesman_id = 1000;

Query OK, 0 rows affected (0.01 sec)

Since no salesman has an id 1000 thus no rows were affected.

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)

Table creation:

1] Actor

mysql> CREATE TABLE ACTOR(ACT_ID INT, ACT_NAME VARCHAR(20), ACT_GENDER CHAR(1), PRIMARY KEY(ACT_ID));

Query OK, 0 rows affected (0.08 sec)

mysql> describ	pe actor;	·	LI		·
Field	Туре	Null	Key	Default	Extra
	int varchar(20) char(1)	NO YES YES	PRI	NULL NULL NULL	
rows in set	(0.02 sec)	+	++		++

mysql> INSERT INTO ACTOR VALUES(301, 'ANUSHKA', 'F'),

- -> (302,'PRABHAS','M'),
- -> (303,'PUNITH','M'),
- -> (304,'JERMY','M');

Query OK, 4 rows affected (0.01 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> select * from actor;

```
| ACT_ID | ACT_NAME | ACT_GENDER |
| 301 | ANUSHKA | F |
| 302 | PRABHAS | M |
| 303 | PUNITH | M |
| 304 | JERMY | M |
| 4 rows in set (0.00 sec)
```

2] Director

mysql> CREATE TABLE DIRECTOR(DIR_ID INT PRIMARY KEY, DIR_NAME VARCHAR(20), DIR PHONE BIGINT);

Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO DIRECTOR VALUES(60, 'RAJAMOULI', 8751611001),

- -> (61,'HITCHCOCK', 7766138911),
- -> (62,'FARAN', 9986776531),
- -> (63,'STEVEN SPIELBERG', 8989776530);

Query OK, 4 rows affected (0.01 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> select * from director;

```
mysql> select * from director;

+-----+

| DIR_ID | DIR_NAME | DIR_PHONE |

+-----+

| 60 | RAJAMOULI | 8751611001 |

| 61 | HITCHCOCK | 7766138911 |

| 62 | FARAN | 9986776531 |

| 63 | STEVEN SPIELBERG | 8989776530 |

+-----+

4 rows in set (0.00 sec)
```

3] Movies

mysql> CREATE TABLE MOVIES(MOV_ID INT, MOV_TITLE VARCHAR(25), MOV_YEAR INT,

- -> MOV LANG VARCHAR(12), DIR ID INT, PRIMARY KEY (MOV ID),
- -> FOREIGN KEY(DIR ID) REFERENCES DIRECTOR(DIR ID));

Query OK, 0 rows affected (0.07 sec)

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

- -> (1002, 'BAHUBALI-1', 2015, 'TELAGU', 60),
- -> (1003,'AKASH', 2008, 'KANNADA', 61),
- -> (1004,'WAR HORSE', 2011, 'ENGLISH', 63);

Query OK, 4 rows affected (0.04 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> select * from movies;

mysql> sel	lect * from mo	ovies;	.	·
MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
:	BAHUBALI-2 BAHUBALI-1 AKASH WAR HORSE	2015 2008	TELAGU TELAGU KANNADA ENGLISH	60 60 61 63
4 rows in	set (0.00 sed	:)		-

4] Movie_cast

mysql> CREATE TABLE MOVIE_CAST(ACT_ID INT, MOV_ID INT, ROLE VARCHAR(10),

- -> PRIMARY KEY(ACT ID, MOV ID),
- -> FOREIGN KEY (ACT ID) REFERENCES ACTOR (ACT ID),
- -> FOREIGN KEY (MOV ID) REFERENCES MOVIES (MOV ID));

Query OK, 0 rows affected (0.11 sec)

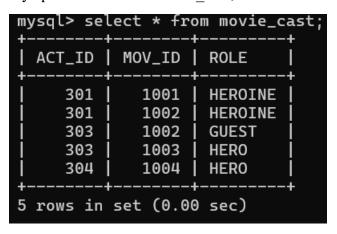
mysql> INSERT INTO MOVIE_CAST VALUES(301, 1002, 'HEROINE'),

- -> (301, 1001, 'HEROINE'),
- -> (303, 1003, 'HERO'),
- -> (303, 1002, 'GUEST'),
- -> (304, 1004, 'HERO');

Query OK, 5 rows affected (0.01 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> select * from movie cast;



5] Rating

mysql> CREATE TABLE RATING(MOV_ID INT, REV_STARS VARCHAR(25), PRIMARY KEY(MOV_ID),

-> FOREIGN KEY(MOV ID) REFERENCES MOVIES(MOV ID));

Query OK, 0 rows affected (0.08 sec)

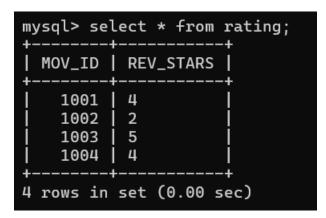
mysql> INSERT INTO RATING VALUES(1001, 4),

- -> (1002, 2),
- -> (1003, 5),
- -> (1004, 4);

Query OK, 4 rows affected (0.01 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> select * from rating;



Write SQL queries to

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

mysql> SELECT MOV TITLE FROM MOVIES

- -> JOIN DIRECTOR ON MOVIES.DIR ID = DIRECTOR.DIR ID
- -> WHERE DIR NAME = 'HITCHCOCK';

```
+-----+

| MOV_TITLE |

+-----+

| AKASH |

+-----+

1 row in set (0.00 sec)
```

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

mysql> 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
- ->);

```
+----+
| MOV_TITLE |
+-----+
| BAHUBALI-2 |
| BAHUBALI-1 |
| AKASH |
+-----+
3 rows in set (0.00 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 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;

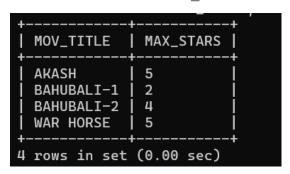
Since, no actor satisfy both the conditions; thus we get empty set as our output.

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.

mysql> SELECT M.MOV TITLE, MAX(R.REV STARS) AS MAX STARS

- -> FROM MOVIES M
- -> JOIN RATING R ON M.MOV ID = R.MOV ID
- -> GROUP BY M.MOV TITLE

-> ORDER BY M.MOV TITLE;

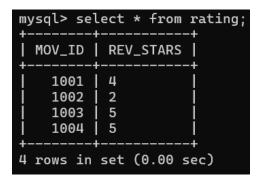


5. Update rating of all movies directed by 'Steven Spielberg' to 5.

mysql> 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 DIR_NAME = 'STEVEN SPIELBERG');

mysql> select * from rating;



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

				STUDEN	TS				
stno	name		add	r		city		state	zip
1011	Edwa	rds P. D	avid 10	Red Rd	10	Newton		MA	02159
2415		an A. Ma		Valnut S		Malden		MA	02148
2661		Leatha		School		Brooklin	e	MA	02146
2890		ne Sand		Cass Ro		Boston	200	MA	02122
3442		k Roland		Beacon		Nashua		NH	03060
3566		e Richard		Park St		Brooklin		MA	02146
4022		Lorraine		teacon S		Boston	~	MA	02125
5544		ings Jerr		Pleasan		Boston		MA	02113
5571		Jerry		Jain Rd		Provider		RI	0290
0011	Dewis	Jerry		aam rea		1 Tovider	ice.	141	02000
			01	NSTRUCT	ORS				
	empno	name			room	no	telno		
	019	Evar	is Robert	Prof	essor	82		7122	ĺ
	023	Exx	on George	Prof	essor	90	- 1	9101	
	056		ver Kathy		c. Prof.			5110	
	126		is William		c. Prof.		- 1	5411	1
	234		Samuel				- 1	7024	
	204		Danie	71007	DU. 1101	50	_	1024	
	_			COURS	ES				
	138	cno	cname			cr	Ca		
		cs110	Introduct				12		
		cs210		Computer Programming Computer Architecture		4	10		
		cs240				3		10	
	1 3	cs310	Data Stru			3	60		
	1.5	cs350	Higher Le			3	50		
		cs410	Software	Enginee	ring	3	40		
		св460	Graphics	34057000	20000	3	30)	
		GR	ADES						
stno	empno	cna	sem	year	grade				
	019	cs110	Fall	2001	40				
1011			Fall	2001					
2661	019	cs110			80				
2661 3566	019	cs110	Fall	2001	9.5				
2661 3566 5544	019 019	cs110 cs110	Fall Fall	2001 2001	95 100				
2661 3566 5544 1011	019 019 023	cs110 cs110 cs110	Fall Fall Spring	2001 2001 2002	95 100 75				
2661 3566 5544 1011 4022	019 019 023 023	cs110 cs110 cs110 cs110	Fall Fall Spring Spring	2001 2001 2002 2002	95 100 75 60				
2661 3566 5544 1011 4022 3566	019 019 023 023 019	cs110 cs110 cs110 cs110 cs240	Fall Fall Spring Spring Spring	2001 2001 2002 2002 2002	95 100 75 60 100				
2661 3566 5544 1011 4022 3566 5571	019 019 023 023 019 019	cs110 cs110 cs110 cs110 cs240 cs240	Fall Fall Spring Spring Spring Spring	2001 2001 2002 2002 2002 2002	95 100 75 60 100 50				
2661 3566 5544 1011 4022 3566 5571 2415	019 019 023 023 019 019 019	cs110 cs110 cs110 cs110 cs240 cs240 cs240	Fall Fall Spring Spring Spring Spring Spring	2001 2001 2002 2002 2002 2002 2002	95 100 75 60 100 50				LOCALO
2661 3566 5544 1011 4022 3566 5571 2415 3442	019 019 023 023 019 019 019 234	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410	Fall Fall Spring Spring Spring Spring Spring Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60				VISING
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571	019 019 023 023 019 019 019 234 234	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs410	Fall Fall Spring Spring Spring Spring Spring Spring Spring Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80			stno	empr
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011	019 019 023 023 019 019 019 234 234 019	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs410 cs210	Fall Fall Spring Spring Spring Spring Spring Spring Spring Fall	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80			stno 1011	empr 019
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2661	019 019 023 023 019 019 019 234 234 019 019	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs410 cs210	Fall Fall Spring Spring Spring Spring Spring Spring Spring Fall Fall	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80 90 70			stno 1011 2415	019 019
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2661 3566	019 019 023 023 019 019 019 234 234 019 019	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs410 cs210 cs210	Fall Fall Spring Spring Spring Spring Spring Spring Spring Fall Fall	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80 90 70			stno 1011 2415 2661	019 019 023
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2661 3566 5571	019 019 023 023 019 019 019 234 234 019 019 019	cs110 cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs240 cs410 cs210 cs210 cs210	Fall Fall Spring Spring Spring Spring Spring Spring Spring Fall Fall Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 80 90 70 90 85			1011 2415 2661 2890	019 019 023 023
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2661 3566 5571 4022	019 019 023 023 019 019 019 234 234 019 019 019	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs210 cs210 cs210 cs210 cs210	Fall Fall Spring Spring Spring Spring Spring Spring Spring Fall Fall Fall Spring Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80 90 70 90 85 70			1011 2415 2661 2890 3442	019 019 023 023 056
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2661 3566 5566 4022 5544	019 019 023 023 019 019 019 234 234 019 019 019 019 019	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs410 cs210 cs210 cs210 cs210 cs210 cs210	Fall Fall Spring Spring Spring Spring Spring Spring Fall Fall Spring Spring Spring Fall Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80 90 70 90 85 70			stno 1011 2415 2661 2890 3442 3566	019 019 023 023 056 126
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2661 3566 5571 4022 5544 1011	019 019 023 023 019 019 019 234 234 019 019 019 019 019 056	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs210 cs210 cs210 cs210 cs210 cs210 cs210 cs240	Fall Fall Spring Spring Spring Spring Spring Spring Fall Fall Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80 90 70 90 85 70 90			stno 1011 2415 2661 2890 3442 3566 4022	019 019 023 023 056 126 234
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2666 5571 4022 5544 1011 4022	019 019 023 023 019 019 019 234 234 219 019 019 019 019 019 056	es110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs210 cs210 cs210 cs210 cs210 cs210 cs240 cs240 cs240	Fall Fall Spring Spring Spring Spring Spring Spring Fall Fall Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80 90 70 90 85 70 70 90 85			stno 1011 2415 2661 2890 3442 3566 4022 5544	019 019 023 023 056 126 234 023
2661 3566 5544 1011 4022 3566 5571 2415 3442 5571 1011 2661 3566 5571 4022 5544 1011	019 019 023 023 019 019 019 234 234 019 019 019 019 019 056	cs110 cs110 cs110 cs110 cs240 cs240 cs240 cs410 cs210 cs210 cs210 cs210 cs210 cs210 cs210 cs240	Fall Fall Spring Spring Spring Spring Spring Spring Fall Fall Spring	2001 2001 2002 2002 2002 2002 2002 2002	95 100 75 60 100 50 100 60 80 90 70 90 85 70 90			stno 1011 2415 2661 2890 3442 3566 4022	019 019 023 023 056 126 234

Table creation:

1] Students

mysql> create table students(stno int primary key, name varchar(100), addr varchar(100),

-> city varchar(50), state varchar(20), zip int);

Query OK, 0 rows affected (0.01 sec)

mysql> describe students;

+ Field	Туре	+ Null	 Key	Default	Extra
stno name addr city state zip	int(11) varchar(100) varchar(100) varchar(50) varchar(20) int(11)	NO YES YES YES YES YES YES	PRI	NULL NULL NULL NULL NULL NULL	
+	set (0.01 sec	+)	+	١١	·+

mysql> insert into students 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 (0.00 sec)

Records: 9 Duplicates: 0 Warnings: 0

stno	name	addr	city	state	zip
1011	Edwards P. David	10 Red Rd.	Newton	MA	20159
2415	Grogan A. Mary	8 Walnut St.	Malden	MA	2148
2661	Mixon Leatha	100 School St.	Brookline	MA	2146
2890	McLane Sandy	30 Case Rd.	Boston	MA	2122
3442	Novak Roland	42 Beacon St.	Nashua	NH	3060
3566	Pierce Richard	70 Park St.	Brookline	MA	2146
4022	Prior Lorraine	8 Beacon St.	Boston	MA	2125
5544	Rawlings Jerry	15 Pleasant Dr.	Boston	MA	2115
5571	Lewis Jerry	1 Main Rd.	Providence	RI	2904

2] Instructors

mysql> create table instructors(empno int primary key, name varchar(50), ranks varchar(50), roomno int, telno int);

Query OK, 0 rows affected (0.01 sec)

mysql> describe instructors;

mysql> insert into instructors 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 (0.00 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> select * from instructors;

empno	name	rank	roomno	telno
23 56 126	Evana Robert Exxon George Sawyer Kathy Davis William Will Samuel	Professor Professor Assoc. Prof. Assoc. Prof. Assist. Prof.	82 90 91 72 90	7122 9101 5110 5411 7024
5 rows in	n set (0.00 sec)		·	

3] Courses

mysql> create table courses(cno varchar(20) primary key, cname varchar(100), cr int, cap int);

Query OK, 0 rows affected (0.01 sec)

mysql> describe courses;

Field	Туре	+ Null	Key	Default	Extra
cno cname cr cap	varchar(20) varchar(100) int(11) int(11)	NO YES YES YES	PRI	NULL NULL NULL NULL	
4 rows in	set (0.01 sec))			· -

mysql> insert into courses 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 (0.00 sec)

Records: 7 Duplicates: 0 Warnings: 0

mysql> select * from courses;

cno	cname	 cr	+ cap
cs110	Introduction to Computing	4	120
cs210 cs240	Computer Programming Computer Architecture	4 3	100 100
cs310 cs350	Data Structures Higher Level Languages	3 3	60 50
cs410	Software Engineering	3	40
cs460 +	Graphics	3 +	30 +
7 rows in	set (0.00 sec)		

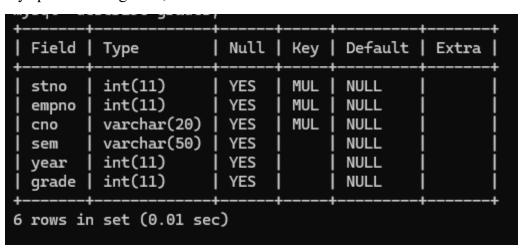
4] Grades

mysql> create table grades(stno int, empno int, cno varchar(20), sem varchar(50), year int,

- -> grade int, foreign key (stno) references students(stno),
- -> foreign key (empno) references instructors(empno),
- -> foreign key (cno) references courses(cno));

Query OK, 0 rows affected (0.01 sec)

mysql> describe grades;



mysql> insert into grades 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),
```

Query OK, 21 rows affected (0.00 sec)

Records: 21 Duplicates: 0 Warnings: 0

mysql> select * from grades;

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

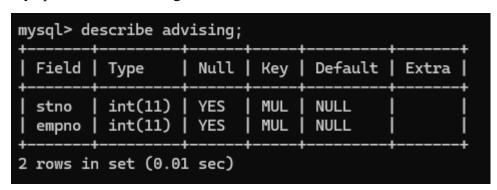
5] Advising

mysql> create table advising(stno int, empno int, foreign key (stno) references students(stno),

-> foreign key (empno) references instructors(empno));

Query OK, 0 rows affected (0.01 sec)

mysql> describe advising;



mysql> insert into advising values(1011, 019),

- -> (2415, 019),
- -> (2661, 023),
- -> (2890, 023),
- -> (3442, 056),

```
-> (3566, 126),
```

$$->$$
 (4022, 234),

$$-> (5544, 023),$$

Query OK, 9 rows affected (0.03 sec)

Records: 9 Duplicates: 0 Warnings: 0

mysql> select * from advising;

```
mysql> select * from advising;

+-----+

| stno | empno |

+----+

| 1011 | 19 |

| 2415 | 19 |

| 2661 | 23 |

| 2890 | 23 |

| 3442 | 56 |

| 3566 | 126 |

| 4022 | 234 |

| 5574 | 23 |

| 5571 | 234 |

+----+
```

For odd rollnumbers(any 10)

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

- -> FROM students S
- -> JOIN grades G ON S.stno = G.stno
- -> JOIN courses C ON G.cno = C.cno
- \rightarrow WHERE C.cr = 4 AND MOD(S.stno, 2) = 1;

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

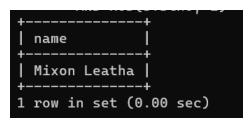
mysql> SELECT DISTINCT S.name

- -> 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
- \rightarrow AND MOD(S.stno, 2) = 1;

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

mysql> SELECT DISTINCT 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'
- \rightarrow AND MOD(S.stno, 2) = 1;



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

- -> FROM students S
- -> JOIN advising A ON S.stno = A.stno
- -> JOIN instructors I ON A.empno = I.empno

- -> WHERE I.ranks != 'Professor'
- \rightarrow AND MOD(S.stno, 2) = 1;

- 5. Find course numbers for courses that enroll exactly two students; mysql> SELECT G.cno
 - -> FROM grades G
 - -> GROUP BY G.cno
 - -> HAVING COUNT(DISTINCT G.stno) = 2;

```
+----+
| cno |
+-----+
| cs310 |
| cs410 |
+-----+
2 rows in set (0.00 sec)
```

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

 mysql> SELECT S.name FROM students S WHERE MOD(S.stno, 2) = 1
 - -> AND NOT EXISTS (
 - -> SELECT 1
 - -> FROM students S2
 - -> WHERE S.city = S2.city
 - -> AND S2.stno != S.stno
 - ->);

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

mysql> 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.ranks = 'Assoc. Prof.';

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

mysql> 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 |
+-----+
| 9101 |
| 7122 |
| 5110 |
| 7024 |
+-----+
4 rows in set (0.00 sec)
```

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

- -> FROM students S
- -> JOIN grades G ON S.stno = G.stno
- -> GROUP BY S.stno

- -> HAVING COUNT(DISTINCT G.cno) = 1
- \rightarrow AND MOD(S.stno, 2) = 1;

10. Find the names of instructors who teach no course.

mysql> SELECT I.name

- -> FROM instructors I
- -> LEFT JOIN grades G ON I.empno = G.empno
- -> WHERE G.cno IS NULL;

For even rollnumbers (any 10)

- 1. Find the names of students who took no four-credit courses. SELECT DISTINCT S.name
 - -> FROM students S
 - \rightarrow WHERE MOD(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
 - \rightarrow AND C.cr = 4
 - ->);

- 2. Find the names of students who took cs210 or cs310. mysql> SELECT DISTINCT S.name
 - -> FROM students S
 - -> JOIN grades G ON S.stno = G.stno
 - -> WHERE G.cno IN ('cs210', 'cs310')
 - \rightarrow AND MOD(S.stno, 2) = 0;

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

mysql> -- At least two students

mysql> SELECT G.cno

- -> FROM grades G
- -> GROUP BY G.cno
- -> HAVING COUNT(DISTINCT G.stno) >= 2;

mysql> -- At least three students

mysql> SELECT G.cno

- -> FROM grades G
- -> GROUP BY G.cno
- -> HAVING COUNT(DISTINCT G.stno) >= 3;

4. Find the names of students who obtained the highest grade in cs210. mysql> SELECT DISTINCT S.name

```
-> FROM students S
```

```
-> JOIN grades G ON S.stno = G.stno
```

```
-> WHERE G.cno = 'cs210'
```

-> AND G.grade = (SELECT MAX(grade) FROM grades WHERE cno = 'cs210')

```
\rightarrow AND MOD(S.stno, 2) = 0;
```

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

mysql> SELECT DISTINCT I.name

-> FROM instructors I

```
-> JOIN grades G ON I.empno = G.empno
-> WHERE EXISTS (
    SELECT 1
->
    FROM grades G2
    WHERE G2.stno = G.stno
->
    AND EXISTS (
->
->
      SELECT 1
      FROM instructors I2
->
->
      WHERE I2.empno = G2.empno
      AND I2.ranks = 'Assist. Prof.'
->
->
   )
->);
```

6. Find the lowest grade of a student who took a course during the spring of 2003. mysql> SELECT MIN(G.grade)

```
-> FROM grades G
```

-> WHERE G.sem = 'Spring'

```
-> AND G.year = 2003;
```

```
+-----+
| MIN(G.grade) |
+-----+
| 70 |
+-----+
1 row in set (0.01 sec)
```

7. 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). mysql> SELECT DISTINCT S.name

```
-> FROM students S
\rightarrow WHERE MOD(S.stno, 2) = 0
-> AND NOT EXISTS (
    SELECT 1
    FROM courses C
    WHERE NOT EXISTS (
->
      SELECT 1
->
      FROM grades G
->
->
      WHERE G.stno = S.stno
      AND G.cno = C.cno
->
   )
    AND EXISTS (
->
      SELECT 1
->
      FROM grades G2
->
->
      WHERE G2.empno = 019 -- Prof. Evans' empno
      AND G2.cno = C.cno
->
->
   )
->);
name
 Pierce Richard
rows in set (0.00 sec)
```

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

```
-> FROM students S
```

- -> JOIN advising A ON S.stno = A.stno
- \rightarrow WHERE MOD(S.stno, 2) = 0
- -> AND NOT EXISTS (
- -> SELECT 1

- -> FROM grades G
- -> WHERE G.stno = S.stno
- -> AND G.empno = A.empno
- ->);

- 9. Find the highest grade of a student who never took cs110. mysql> SELECT MAX(G.grade)
 - -> FROM grades G
 - -> WHERE G.stno NOT IN (
 - -> SELECT G2.stno
 - -> FROM grades G2
 - -> WHERE G2.cno = 'cs110'
 - ->)
 - \rightarrow AND MOD(G.stno, 2) = 0;

- 10. Find names of courses taken by students who do not live in Massachusetts (MA). mysql> SELECT DISTINCT G.cno
 - -> FROM grades G
 - -> JOIN students S ON G.stno = S.stno
 - -> WHERE S.state != 'MA';

```
+----+
| cno |
+----+
| cs410 |
| cs240 |
| cs210 |
+----+
3 rows in set (0.00 sec)
```