EXPERIMENT 1

DATE :

1. Create the following Relation (Tables) with primary key integrity constraint

```
-- create
CREATE TABLE instructor (
  ID INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  dept_name TEXT NOT NULL,
  salary INTEGER NOT NULL
);

-- insert
INSERT INTO instructor (ID, name, dept_name, salary) VALUES
(10101, 'Srinivasan', 'Comp. Sci.', 65000),
(12121, 'Wu', 'Finance', 90000),
(15151, 'Mozart', 'Music', 40000),
(22222, 'Einstein', 'Physics', 95000),
(32343, 'El Said', 'History', 60000),
(33456, 'Gold', 'Physics', 87000),
(45565, 'Katz', 'Comp. Sci.', 75000),
(58583, 'Califieri', 'History', 6200),
(76543, 'Singh', 'Finance', 80000),
(76766, 'Crick', 'Biology', 72000),
(83821, 'Brandt', 'Comp. Sci.', 92000),
(98345, 'Kim', 'Elec. Eng', 80000);
-- fetch
SELECT * FROM instructor;
```

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	6200
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng	80000

2. Create the following Relation (Tables) teaches

```
CREATE TABLE teaches (
  ID int NOT NULL,
```

```
  course_id varchar(255) NOT NULL,
  sec_id int NOT NULL,
  semester varchar(255) NOT NULL,
  year int NOT NULL,
  FOREIGN KEY (ID) REFERENCES instructor(ID)
);

INSERT INTO teaches (ID, course_id, sec_id, semester, year) VALUES
(10101, 'CS-101', 1, 'Fall', 2017),
(10101, 'CS-315', 1, 'Spring', 2018),
(10101, 'CS-347', 1, 'Fall', 2017),
(12121, 'FIN-201', 1, 'Spring', 2018),
(15151, 'MU-199', 1, 'Spring', 2015),
(22222, 'PHY-101', 1, 'Fall', 2017),
(32343, 'HIS-351', 1, 'Spring', 2018),
(45565, 'CS-101', 1, 'Spring', 2018),
(45565, 'CS-319', 1, 'Spring', 2018),
(76766, 'BIO-101', 1, 'Summer', 2017),
(76766, 'BIO-301', 1, 'Summer', 2018),
(83821, 'CS-190', 1, 'Spring', 2017),
(83821, 'CS-190', 2, 'Spring', 2017),
(83821, 'CS-319', 2, 'Spring', 2018),
(98345, 'EE-181', 1, 'Spring', 2017);

SELECT * FROM teaches;
```

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2015
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

3. Insert following additional tuple in instructor ('10211', 'Smith', 'Biology', 66000)
INSERT INTO instructor VALUES ('10211', 'Smith', 'Biology', 66000);

SELECT * FROM instructor;

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
10211	Smith	Biology	66000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	6200
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng	80000

4. Delete this tuple from instructor ('10211', 'Smith', 'Biology', 66000)
DELETE FROM instructor WHERE ID=10211;
SELECT * FROM instructor;

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	6200
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng	80000

5. Select tuples from instructor where dept_name = ‘History’
SELECT * FROM instructor where dept_name='History';

ID	name	dept_name	salary
32343	El Said	History	60000
58583	Califieri	History	6200

6. Find the Cartesian product instructor x teaches.
SELECT * FROM instructor CROSS JOIN teaches;

ID	name	dept_name	salary	ID	course_id	sec_id	semester	year
98345	Kim	Elec. Eng	80000	10101	CS-101	1	Fall	2017
83821	Brandt	Comp. Sci.	92000	10101	CS-101	1	Fall	2017
76766	Crick	Biology	72000	10101	CS-101	1	Fall	2017
76543	Singh	Finance	80000	10101	CS-101	1	Fall	2017
58583	Califieri	History	6200	10101	CS-101	1	Fall	2017
45565	Katz	Comp. Sci.	75000	10101	CS-101	1	Fall	2017
33456	Gold	Physics	87000	10101	CS-101	1	Fall	2017
32343	El Said	History	60000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
98345	Kim	Elec. Eng	80000	10101	CS-315	1	Spring	2018
83821	Brandt	Comp. Sci.	92000	10101	CS-315	1	Spring	2018
76766	Crick	Biology	72000	10101	CS-315	1	Spring	2018
76543	Singh	Finance	80000	10101	CS-315	1	Spring	2018
58583	Califieri	History	6200	10101	CS-315	1	Spring	2018
45565	Katz	Comp. Sci.	75000	10101	CS-315	1	Spring	2018
33456	Gold	Physics	87000	10101	CS-315	1	Spring	2018
32343	El Said	History	60000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
98345	Kim	Elec. Eng	80000	10101	CS-347	1	Fall	2017
83821	Brandt	Comp. Sci.	92000	10101	CS-347	1	Fall	2017
76766	Crick	Biology	72000	10101	CS-347	1	Fall	2017
76543	Singh	Finance	80000	10101	CS-347	1	Fall	2017
58583	Califieri	History	6200	10101	CS-347	1	Fall	2017
45565	Katz	Comp. Sci.	75000	10101	CS-347	1	Fall	2017
33456	Gold	Physics	87000	10101	CS-347	1	Fall	2017
32343	El Said	History	60000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
98345	Kim	Elec. Eng	80000	12121	FIN-201	1	Spring	2018
83821	Brandt	Comp. Sci.	92000	12121	FIN-201	1	Spring	2018
76766	Crick	Biology	72000	12121	FIN-201	1	Spring	2018
76543	Singh	Finance	80000	12121	FIN-201	1	Spring	2018
58583	Califieri	History	6200	12121	FIN-201	1	Spring	2018
45565	Katz	Comp. Sci.	75000	12121	FIN-201	1	Spring	2018
33456	Gold	Physics	87000	12121	FIN-201	1	Spring	2018
32343	El Said	History	60000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
98345	Kim	Elec. Eng	80000	15151	MU-199	1	Spring	2015
83821	Brandt	Comp. Sci.	92000	15151	MU-199	1	Spring	2015
76766	Crick	Biology	72000	15151	MU-199	1	Spring	2015
76543	Singh	Finance	80000	15151	MU-199	1	Spring	2015
58583	Califieri	History	6200	15151	MU-199	1	Spring	2015
45565	Katz	Comp. Sci.	75000	15151	MU-199	1	Spring	2015
33456	Gold	Physics	87000	15151	MU-199	1	Spring	2015
32343	El Said	History	60000	15151	MU-199	1	Spring	2015
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2015
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2015
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2015
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2015
98345	Kim	Elec. Eng	80000	22222	PHY-101	1	Fall	2017
83821	Brandt	Comp. Sci.	92000	22222	PHY-101	1	Fall	2017
76766	Crick	Biology	72000	22222	PHY-101	1	Fall	2017
76543	Singh	Finance	80000	22222	PHY-101	1	Fall	2017
58583	Califieri	History	6200	22222	PHY-101	1	Fall	2017
45565	Katz	Comp. Sci.	75000	22222	PHY-101	1	Fall	2017
33456	Gold	Physics	87000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	22222	PHY-101	1	Fall	2017
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017

15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
98345	Kim	Elec. Eng	80000	32343	HIS-351	1	Spring	2018
83821	Brandt	Comp. Sci.	92000	32343	HIS-351	1	Spring	2018
76766	Crick	Biology	72000	32343	HIS-351	1	Spring	2018
76543	Singh	Finance	80000	32343	HIS-351	1	Spring	2018
58583	Califieri	History	6200	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	32343	HIS-351	1	Spring	2018
33456	Gold	Physics	87000	32343	HIS-351	1	Spring	2018
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
22222	Einstein	Physics	95000	32343	HIS-351	1	Spring	2018
15151	Mozart	Music	40000	32343	HIS-351	1	Spring	2018
12121	Wu	Finance	90000	32343	HIS-351	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	32343	HIS-351	1	Spring	2018
98345	Kim	Elec. Eng	80000	45565	CS-101	1	Spring	2018
83821	Brandt	Comp. Sci.	92000	45565	CS-101	1	Spring	2018
76766	Crick	Biology	72000	45565	CS-101	1	Spring	2018
76543	Singh	Finance	80000	45565	CS-101	1	Spring	2018
58583	Califieri	History	6200	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
33456	Gold	Physics	87000	45565	CS-101	1	Spring	2018
32343	El Said	History	60000	45565	CS-101	1	Spring	2018
22222	Einstein	Physics	95000	45565	CS-101	1	Spring	2018
15151	Mozart	Music	40000	45565	CS-101	1	Spring	2018
12121	Wu	Finance	90000	45565	CS-101	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	45565	CS-101	1	Spring	2018
98345	Kim	Elec. Eng	80000	45565	CS-319	1	Spring	2018
83821	Brandt	Comp. Sci.	92000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	45565	CS-319	1	Spring	2018
76543	Singh	Finance	80000	45565	CS-319	1	Spring	2018
58583	Califieri	History	6200	45565	CS-319	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
33456	Gold	Physics	87000	45565	CS-319	1	Spring	2018
32343	El Said	History	60000	45565	CS-319	1	Spring	2018
22222	Einstein	Physics	95000	45565	CS-319	1	Spring	2018
15151	Mozart	Music	40000	45565	CS-319	1	Spring	2018
12121	Wu	Finance	90000	45565	CS-319	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	45565	CS-319	1	Spring	2018
98345	Kim	Elec. Eng	80000	76766	BIO-101	1	Summer	2017
83821	Brandt	Comp. Sci.	92000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76543	Singh	Finance	80000	76766	BIO-101	1	Summer	2017
58583	Califieri	History	6200	76766	BIO-101	1	Summer	2017
45565	Katz	Comp. Sci.	75000	76766	BIO-101	1	Summer	2017
33456	Gold	Physics	87000	76766	BIO-101	1	Summer	2017
32343	El Said	History	60000	76766	BIO-101	1	Summer	2017
22222	Einstein	Physics	95000	76766	BIO-101	1	Summer	2017
15151	Mozart	Music	40000	76766	BIO-101	1	Summer	2017
12121	Wu	Finance	90000	76766	BIO-101	1	Summer	2017
10101	Srinivasan	Comp. Sci.	65000	76766	BIO-101	1	Summer	2017
98345	Kim	Elec. Eng	80000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	76766	BIO-301	1	Summer	2018
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
76543	Singh	Finance	80000	76766	BIO-301	1	Summer	2018
58583	Califieri	History	6200	76766	BIO-301	1	Summer	2018
45565	Katz	Comp. Sci.	75000	76766	BIO-301	1	Summer	2018
33456	Gold	Physics	87000	76766	BIO-301	1	Summer	2018
32343	El Said	History	60000	76766	BIO-301	1	Summer	2018
22222	Einstein	Physics	95000	76766	BIO-301	1	Summer	2018
15151	Mozart	Music	40000	76766	BIO-301	1	Summer	2018
12121	Wu	Finance	90000	76766	BIO-301	1	Summer	2018
10101	Srinivasan	Comp. Sci.	65000	76766	BIO-301	1	Summer	2018
98345	Kim	Elec. Eng	80000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
76766	Crick	Biology	72000	83821	CS-190	1	Spring	2017
76543	Singh	Finance	80000	83821	CS-190	1	Spring	2017
58583	Califieri	History	6200	83821	CS-190	1	Spring	2017
45565	Katz	Comp. Sci.	75000	83821	CS-190	1	Spring	2017
33456	Gold	Physics	87000	83821	CS-190	1	Spring	2017
32343	El Said	History	60000	83821	CS-190	1	Spring	2017
22222	Einstein	Physics	95000	83821	CS-190	1	Spring	2017
15151	Mozart	Music	40000	83821	CS-190	1	Spring	2017
12121	Wu	Finance	90000	83821	CS-190	1	Spring	2017
10101	Srinivasan	Comp. Sci.	65000	83821	CS-190	1	Spring	2017
98345	Kim	Elec. Eng	80000	83821	CS-190	2	Spring	2017

76766	Crick	Biology	72000	83821	CS-190	2	Spring	2017
76543	Singh	Finance	80000	83821	CS-190	2	Spring	2017
58583	Califieri	History	6200	83821	CS-190	2	Spring	2017
45565	Katz	Comp. Sci.	75000	83821	CS-190	2	Spring	2017
33456	Gold	Physics	87000	83821	CS-190	2	Spring	2017
32343	El Said	History	60000	83821	CS-190	2	Spring	2017
22222	Einstein	Physics	95000	83821	CS-190	2	Spring	2017
15151	Mozart	Music	40000	83821	CS-190	2	Spring	2017
12121	Wu	Finance	90000	83821	CS-190	2	Spring	2017
10101	Srinivasan	Comp. Sci.	65000	83821	CS-190	2	Spring	2017
98345	Kim	Elec. Eng	80000	83821	CS-319	2	Spring	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
76766	Crick	Biology	72000	83821	CS-319	2	Spring	2018
76543	Singh	Finance	80000	83821	CS-319	2	Spring	2018
58583	Califieri	History	6200	83821	CS-319	2	Spring	2018
45565	Katz	Comp. Sci.	75000	83821	CS-319	2	Spring	2018
33456	Gold	Physics	87000	83821	CS-319	2	Spring	2018
32343	El Said	History	60000	83821	CS-319	2	Spring	2018
22222	Einstein	Physics	95000	83821	CS-319	2	Spring	2018
15151	Mozart	Music	40000	83821	CS-319	2	Spring	2018
12121	Wu	Finance	90000	83821	CS-319	2	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng	80000	98345	EE-181	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	98345	EE-181	1	Spring	2017
76766	Crick	Biology	72000	98345	EE-181	1	Spring	2017
76543	Singh	Finance	80000	98345	EE-181	1	Spring	2017
58583	Califieri	History	6200	98345	EE-181	1	Spring	2017
45565	Katz	Comp. Sci.	75000	98345	EE-181	1	Spring	2017
33456	Gold	Physics	87000	98345	EE-181	1	Spring	2017
32343	El Said	History	60000	98345	EE-181	1	Spring	2017
22222	Einstein	Physics	95000	98345	EE-181	1	Spring	2017
15151	Mozart	Music	40000	98345	EE-181	1	Spring	2017
12121	Wu	Finance	90000	98345	EE-181	1	Spring	2017
10101	Srinivasan	Comp. Sci.	65000	98345	EE-181	1	Spring	2017

7. Find the names of all instructors who have taught some course and the course_id
SELECT i.name, t.course_id FROM instructor i INNER JOIN teaches t on i.ID= t.ID;

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

8. Find the names of all instructors whose name includes the substring “dar”.
SELECT name FROM instructor where name LIKE "%dar%";

9. Find the names of all instructors with salary between 90,000 and 100,000 (that is, ≥ 90,000 and ≤ 100,000)
SELECT name FROM instructor where salary>= 90000 AND salary<=100000;

name
Wu
Einstein
Brandt

EXPERIMENT 2

1. Order the tuples in the instructors relation as per their salary.
SELECT * FROM instructor ORDER BY salary;

ID	name	dept_name	salary
58583	Califieri	History	6200
15151	Mozart	Music	40000
32343	El Said	History	60000
10101	Srinivasan	Comp. Sci.	65000
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
76543	Singh	Finance	80000
98345	Kim	Elec. Eng	80000
33456	Gold	Physics	87000
12121	Wu	Finance	90000
83821	Brandt	Comp. Sci.	92000
22222	Einstein	Physics	95000

2. Find courses that ran in Fall 2017 or in Spring 2018
SELECT DISTINCT course_id FROM teaches WHERE (semester='Fall'and year=2017)OR (semester='Spring' and year=2018);

course_id
CS-101
CS-315
CS-347
FIN-201
PHY-101
HIS-351
CS-319

3. Find courses that ran in Fall 2017 and in Spring 2018
SELECT DISTINCT course_id FROM teaches WHERE (semester='Fall'and year=2017) AND (semester='Spring' and year=2018);

4. Find courses that ran in Fall 2017 but not in Spring 2018
SELECT DISTINCT course_id FROM teaches t1 WHERE (t1.semester='Fall'and t1.year=2017) AND NOT EXISTS (SELECT 1 FROM teaches t2 WHERE t2.course_id= t1.course_id AND t2.semester='Spring' AND t2.year=2018);

course_id
CS-347
PHY-101

5. Insert following additional tuples in instructor :('10211', 'Smith', 'Biology', 66000), ('10212', 'Tom', 'Biology', NULL)
INSERT INTO instructor VALUES ('10211', 'Smith', 'Biology', 66000), ('10212', 'Tom', 'Biology', NULL);
SELECT * FROM instructor;

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
10211	Smith	Biology	66000
10212	Tom	Biology	NULL
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	6200
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng	80000

6. Find all instructors whose salary is null.
SELECT name FROM instructor WHERE salary IS NULL;

name
Tom

7. Find the average salary of instructors in the Computer Science department.
SELECT AVG(salary) AS avg_salary FROM instructor WHERE dept_name='Comp. Sci.';

avg_salary
77333.3333

EXPERIMENT 3

1. Find the total number of instructors who teach a course in the Spring 2018 semester.
SELECT COUNT(DISTINCT ID) AS total_instructors FROM teaches WHERE semester='Spring' AND year=2018;

total_instructors
5

2. Find the number of tuples in the teaches relation
SELECT COUNT(*) AS num_tuples FROM teaches;

num_tuples
15

3. Find the average salary of instructors in each department
SELECT dept_name, AVG(salary) as avg_salary FROM instructor GROUP BY dept_name;

dept_name	avg_salary
Comp. Sci.	77333.3333
Biology	69000.0000
Finance	85000.0000
Music	40000.0000
Physics	91000.0000
History	33100.0000
Elec. Eng	80000.0000

4. Find the names and average salaries of all departments whose average salary is greater than 42000
SELECT dept_name, AVG(salary) as avg_salary FROM instructor GROUP BY dept_name HAVING AVG(salary)>42000;

dept_name	avg_salary
Comp. Sci.	77333.3333
Biology	69000.0000
Finance	85000.0000
Physics	91000.0000
Elec. Eng	80000.0000

5. Name all instructors whose name is neither “Mozart” nor Einstein”
SELECT name FROM instructor WHERE name NOT IN ("Mozart","Einstein");

name
Srinivasan
Smith
Tom
Wu
El Said
Gold
Katz
Califieri
Singh
Crick
Brandt
Kim

6. Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.
SELECT l.name FROM instructor l WHERE l.salary > (SELECT salary FROM instructor WHERE dept_name='Biology' AND name="Crick");

name
Wu
Einstein
Gold
Katz
Singh
Brandt
Kim

7. Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.

SELECT l.name FROM instructor l WHERE l.salary > (SELECT max(salary) FROM instructor WHERE dept_name='Biology');

name
Wu
Einstein
Gold
Katz
Singh
Brandt
Kim

8. Find the average instructors' salaries of those departments where the average salary is greater than 42,000
SELECT dept_name, AVG(salary) as average_salary FROM instructor GROUP BY dept_name HAVING AVG(salary)>42000;

dept_name	average_salary
Comp. Sci.	77333.3333
Biology	69000.0000
Finance	85000.0000
Physics	91000.0000
Elec. Eng	80000.0000

EXPERIMENT 4

1. Find all departments where the total salary is greater than the average of the total salary at all departments

   SELECT dept_name, SUM(salary) AS total_salary
   FROM instructor GROUP BY dept_name
   HAVING SUM(salary) > (SELECT AVG(total_salary) FROM (SELECT SUM(salary) AS total_salary
   FROM instructor GROUP BY dept_name) AS avg_salary);

dept_name	total_salary
Comp. Sci.	232000
Finance	170000
Physics	182000

2. List the names of instructors along with the course ID of the courses that they taught

```
SELECT i.name AS instructor_name, t.course_id
FROM instructor i JOIN teaches t ON i.ID = t.ID;
```

instructor_name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

3. List the names of instructors along with the course ID of the courses that they taught. In case, an instructor teaches no courses keep the course ID as null.

```
SELECT i.name AS instructor_name, t.course_id
FROM instructor i LEFT JOIN teaches t ON i.ID = t.ID;
```

instructor_name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Gold	NULL
Katz	CS-101
Katz	CS-319
Califieri	NULL
Singh	NULL
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

4. Create a view of instructors without their salary called faculty
   CREATE VIEW faculty AS SELECT ID, name, dept_name
   FROM instructor;
   SELECT * FROM faculty;

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music
22222	Einstein	Physics
32343	El Said	History
33456	Gold	Physics
45565	Katz	Comp. Sci.
58583	Califieri	History
76543	Singh	Finance
76766	Crick	Biology
83821	Brandt	Comp. Sci.
98345	Kim	Elec. Eng

5. Give select privileges on the view faculty to the new user.

   GRANT SELECT ON faculty TO new_user;

EXPERIMENT 5

1. Create a view of instructors without their salary called faculty

   CREATE VIEW faculty1 AS SELECT ID, name, dept_name
   FROM instructor;
   SELECT * FROM faculty1;

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music
22222	Einstein	Physics
32343	El Said	History
33456	Gold	Physics
45565	Katz	Comp. Sci.
58583	Califieri	History
76543	Singh	Finance
76766	Crick	Biology
83821	Brandt	Comp. Sci.
98345	Kim	Elec. Eng

2. Create a view of department salary totals

   CREATE VIEW department_salary_totals AS SELECT dept_name, SUM(salary) AS total_salary FROM instructor GROUP BY dept_name;

SELECT * FROM department_salary_totals;

```
+------------+--------------+
| dept_name  | total_salary |
+------------+--------------+
| Comp. Sci. |       232000 |
| Finance    |       170000 |
| Music      |        40000 |
| Physics    |       182000 |
| History    |        66200 |
| Biology    |        72000 |
| Elec. Eng  |        80000 |
+------------+--------------+
```

3. Create a role of student

   CREATE ROLE student;

4. Give select privileges on the view faculty to the role student.

   GRANT SELECT ON faculty TO student;

5. Create a new user and assign her the role of student.
   CREATE USER guru@localhost IDENTIFIED BY '1234';
   GRANT student TO guru@localhost;

6. Login as this new user and find all instructors in the Biology department.
   GRANT ALL PRIVILEGES ON student.* TO guru@localhost;

   SELECT * FROM faculty WHERE dept_name = 'Biology';

	ID	name	dept_name
▶	10211	Smith	Biology
	10212	Tom	Biology
	76766	Crick	Biology

7. Revoke privileges of the new user
   REVOKE student FROM guru@localhost;

8. Remove the role of student.
   DROP ROLE student;

9. Give select privileges on the view faculty to the new user.
   GRANT SELECT ON faculty TO guru@localhost;

10. Login as this new user and find all instructors in the finance department.
    SELECT * FROM faculty WHERE dept_name = 'Finance';

	ID	name	dept_name
▶	12121	Wu	Finance
	76543	Singh	Finance

11. Login again as root user

12. Create table teaches2 with same columns as teaches but with additional constraint that that semester is one of fall, winter, spring or summer

```
CREATE TABLE teaches2 (
  ID INT NOT NULL,
  course_id VARCHAR(255) NOT NULL,
  sec_id INT NOT NULL,
  semester VARCHAR(255) NOT NULL CHECK (semester IN ('Fall', 'Winter', 'Spring',
'Summer')),
  year INT NOT NULL,
  FOREIGN KEY (ID) REFERENCES instructor(ID)
);
```

13. Create index ID column of teaches. Compare the difference in time to obtain query results with or without index.
    CREATE INDEX idx_ID ON teaches (ID);

14. Drop the index to free up the space.
    DROP INDEX idx_ID ON teaches;

EXPERIMENT 6

Accessing the database through Python

1. Insert following additional tuple in instructor : ('10211', 'Smith', 'Biology', 66000)
2. Delete this tuple from instructor : ('10211', 'Smith', 'Biology', 66000)
3. Select tuples from instructor where dept_name = ‘History’
4. Find the Cartesian product instructor x teaches.
5. Find the names of all instructors who have taught some course and the course_id
6. Find the names of all instructors whose name includes the substring “dar”.
7. Find the names of all instructors with salary between 90,000 and 100,000 (that is, ≥ 90,000 and ≤ 100,000)

```
import mysql.connector

conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='root123',
    database='exp6'
)

cursor = conn.cursor()

create_table_query = """
CREATE TABLE instructor (
  ID INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  dept_name VARCHAR(255) NOT NULL,
  salary INT
)
"""
cursor.execute(create_table_query)

insert_query = """
INSERT INTO instructor (ID, name, dept_name, salary) VALUES
(10101, 'Srinivasan', 'Comp. Sci.', 65000),
(12121, 'Wu', 'Finance', 90000),
(15151, 'Mozart', 'Music', 40000),
(22222, 'Einstein', 'Physics', 95000),
(32343, 'El Said', 'History', 60000),
(33456, 'Gold', 'Physics', 87000),
(45565, 'Katz', 'Comp. Sci.', 75000),
(58583, 'Califieri', 'History', 62000),
(76543, 'Singh', 'Finance', 80000),
(76766, 'Crick', 'Biology', 72000),
(83821, 'Brandt', 'Comp. Sci.', 92000),
(98345, 'Kim', 'Elec. Eng', 80000)
"""
cursor.execute(insert_query)
```

```
create_table_query = """
CREATE TABLE teaches (
  ID INT,
  course_id VARCHAR(255),
  sec_id INT,
  semester VARCHAR(255),
  year INT,
  FOREIGN KEY (ID) REFERENCES instructor(ID)
)
"""
cursor.execute(create_table_query)

insert_query = """
INSERT INTO teaches (ID, course_id, sec_id, semester, year) VALUES
(10101, 'CS-101', 1, 'Fall', 2017),
(10101, 'CS-315', 1, 'Spring', 2018),
(10101, 'CS-347', 1, 'Fall', 2017),
(12121, 'FIN-201', 1, 'Spring', 2018),
(15151, 'MU-199', 1, 'Spring', 2015),
(22222, 'PHY-101', 1, 'Fall', 2017),
(32343, 'HIS-351', 1, 'Spring', 2018),
(45565, 'CS-101', 1, 'Spring', 2018),
(45565, 'CS-319', 1, 'Spring', 2018),
(76766, 'BIO-101', 1, 'Summer', 2017),
(76766, 'BIO-301', 1, 'Summer', 2018),
(83821, 'CS-190', 1, 'Spring', 2017),
(83821, 'CS-190', 2, 'Spring', 2017),
(83821, 'CS-319', 2, 'Spring', 2018),
(98345, 'EE-181', 1, 'Spring', 2017)
"""
cursor.execute(insert_query)

# 1
insert_query = """
INSERT INTO instructor (ID, name, dept_name, salary) VALUES
('10211', 'Smith', 'Biology', 66000)
"""
cursor.execute(insert_query)

# 2
tuple_to_delete = ('10211', 'Smith', 'Biology', 66000)

delete_query = "DELETE FROM instructor WHERE ID = %s AND name = %s AND
dept_name = %s AND salary = %s"
cursor.execute(delete_query, tuple_to_delete)

# 3
dept_name = 'History'
```

```
select_query = "SELECT * FROM instructor WHERE dept_name = %s"
cursor.execute(select_query, (dept_name,))

results = cursor.fetchall()

for row in results:
    print(row)

# 4
cartesian_query = """
SELECT * FROM instructor, teaches
"""

cursor.execute(cartesian_query)

results = cursor.fetchall()

for row in results:
    print(row)

# 5
query = """
SELECT DISTINCT instructor.name, teaches.course_id
FROM instructor
JOIN teaches ON instructor.ID = teaches.ID
"""

# Execute the query
cursor.execute(query)

# Fetch the results
results = cursor.fetchall()

# Print the results
for row in results:
    print(row)

# 6
query = """
SELECT name
FROM instructor
WHERE name LIKE '%dar%'
"""

cursor.execute(query)

results = cursor.fetchall()
```

```
for row in results:
    print(row[0])

# 7
query = """
SELECT name
FROM instructor
WHERE salary BETWEEN 90000 AND 100000
"""

cursor.execute(query)

results = cursor.fetchall()

for row in results:
    print(row[0])

conn.commit()

cursor.close()
conn.close()
```

EXPERIMENT 7

1. Order the tuples in the instructors relation as per their salary.
2. Find courses that ran in Fall 2017 or in Spring 2018
3. Find courses that ran in Fall 2017 and in Spring 2018
4. Find courses that ran in Fall 2017 but not in Spring 2018
5. Insert following additional tuples in instructor ('10211', 'Smith', 'Biology', 66000) ('10212', 'Tom', 'Biology', NULL
6. Find all instructors whose salary is null.
7. Find the average salary of instructors in the Computer Science department.
8. Find the total number of instructors who teach a course in the Spring 2018 semester.
9. Find the number of tuples in the teaches relation
10. Find the average salary of instructors in each department

11. Find the names and average salaries of all departments whose average salary is greater than 42000
12. Name all instructors whose name is neither “Mozart” nor Einstein”.
13. Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.
14. Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.
15. Find the average instructors’ salaries of those departments where the average salary is greater than 42,000.
16. Find all departments where the total salary is greater than the average of the total salary at all departments
17. List the names of instructors along with the course ID of the courses that they taught.
18. List the names of instructors along with the course ID of the courses that they taught. In case, an instructor teaches no courses keep the course ID as null.

```
import mysql.connector

conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='root123',
    database='exp6'
)

cursor = conn.cursor()

#  Order the tuples in the instructors relation as per their salary.
order_by_salary_query = """
SELECT * FROM instructor
ORDER BY salary
"""

cursor.execute(order_by_salary_query)

results = cursor.fetchall()

print("Question1:")
for row in results:
    print(row)
print("\n")

# Find courses that ran in Fall 2017 or in Spring 2018
courses_in_spring_or_fall = """
SELECT DISTINCT course_id FROM teaches WHERE (semester='Fall'and year=2017)OR
(semester='Spring' and year=2018)
"""

cursor.execute(courses_in_spring_or_fall)
```

```
results = cursor.fetchall()

print("Question2:")
for row in results:
    print(row)
print("\n")

# Find courses that ran in Fall 2017 and in Spring 2018
courses_in_spring_and_fall = """
SELECT DISTINCT course_id FROM teaches WHERE (semester='Fall'and year=2017)
AND (semester='Spring' and year=2018)
"""

cursor.execute(courses_in_spring_and_fall)

results = cursor.fetchall()

print("Question3:")
for row in results:
    print(row)
print("\n")

# Find courses that ran in Fall 2017 but not in Spring 2018
course_in_fall_only = """
SELECT DISTINCT course_id FROM teaches t1 WHERE (t1.semester='Fall'and
t1.year=2017) AND NOT EXISTS (SELECT 1 FROM teaches t2 WHERE t2.course_id=
t1.course_id AND t2.semester='Spring' AND t2.year=2018)
"""

cursor.execute(course_in_fall_only)

results = cursor.fetchall()

print("Question4:")
for row in results:
    print(row)
print("\n")

#  Insert following additional tuples in instructor
insert_tuples= """
INSERT INTO instructor VALUES ('10211', 'Smith', 'Biology', 66000), ('10212',
'Tom', 'Biology', NULL )
"""

cursor.execute(insert_tuples)

select_table = """
SELECT * FROM instructor
```

```
"""

cursor.execute(select_table)

results = cursor.fetchall()

print("Question5:")
for row in results:
    print(row)
print("\n")

#  Find all instructors whose salary is null.
instructor_salary_null = """
SELECT name FROM instructor WHERE salary IS NULL
"""

cursor.execute(instructor_salary_null)

results = cursor.fetchall()

print("Question6:")
for row in results:
    print(row)
print("\n")

# Find the average salary of instructors in the Computer Science department.
avg_cs_dept = """
SELECT AVG(salary) AS avg_salary FROM instructor WHERE dept_name='Comp. Sci.'
"""

cursor.execute(avg_cs_dept)

results = cursor.fetchall()

print("Question7:")
for row in results:
    print(row)
print("\n")

#  Find the total number of instructors who teach a course in the Spring 2018
semester.
instructors_spring = """
SELECT COUNT(DISTINCT ID) AS total_instructors FROM teaches WHERE
semester='Spring' AND year=2018
"""

cursor.execute(instructors_spring)
```

```
results = cursor.fetchall()

print("Question8:")
for row in results:
    print(row)
print("\n")

# Find the number of tuples in the teaches relation
teaches_count = """
SELECT COUNT(*) AS num_tuples FROM teaches
"""

cursor.execute(teaches_count)

results = cursor.fetchall()

print("Question9:")
for row in results:
    print(row)
print("\n")

# Find the average salary of instructors in each department
avg_instructor = """
SELECT dept_name, AVG(salary) as avg_salary FROM instructor GROUP BY dept_name
"""

cursor.execute(avg_instructor)

results = cursor.fetchall()

print("Question10:")
for row in results:
    print(row)
print("\n")

# Find the names and average salaries of all departments whose average salary
is greater than 42000
avg_salary_greater = """
SELECT dept_name, AVG(salary) as avg_salary FROM instructor GROUP BY dept_name
HAVING AVG(salary)>42000
"""

cursor.execute(avg_salary_greater)

results = cursor.fetchall()

print("Question11:")
for row in results:
```

```
    print(row)
print("\n")

#  Name all instructors whose name is neither “Mozart” nor Einstein”.
instructor_name = """
SELECT name FROM instructor WHERE name NOT IN ("Mozart","Einstein")
"""

cursor.execute(instructor_name)

results = cursor.fetchall()

print("Question12:")
for row in results:
    print(row)
print("\n")

#  Find names of instructors with salary greater than that of some (at least
one) instructor in the Biology department.
salary_greater= """
SELECT l.name FROM instructor l WHERE l.salary > (SELECT salary FROM
instructor WHERE dept_name='Biology' AND name="Crick")
"""

cursor.execute(salary_greater)

results = cursor.fetchall()

print("Question13:")
for row in results:
    print(row)
print("\n")

# Find the names of all instructors whose salary is greater than the salary of
all instructors in the Biology department.
salary_greater_biology = """
SELECT l.name FROM instructor l WHERE l.salary > (SELECT max(salary) FROM
instructor WHERE dept_name='Biology')
"""

cursor.execute(salary_greater_biology)

results = cursor.fetchall()

print("Question14:")
for row in results:
    print(row)
print("\n")
```

```
#  Find the average instructors' salaries of those departments where the
average salary is greater than 42,000.
avg_instructor_greater = """
SELECT dept_name, AVG(salary) as average_salary FROM instructor GROUP BY
dept_name HAVING AVG(salary)>42000
"""

cursor.execute(avg_instructor_greater)

results = cursor.fetchall()

print("Question15:")
for row in results:
    print(row)
print("\n")

#  Find all departments where the total salary is greater than the average of
the total salary at all
department_salary = """
SELECT dept_name
FROM (
    SELECT dept_name, SUM(salary) AS total_salary
    FROM instructor
    GROUP BY dept_name
) AS department_total_salary
WHERE total_salary > (
    SELECT AVG(total_salary)
    FROM (
        SELECT SUM(salary) AS total_salary
        FROM instructor
        GROUP BY dept_name
    ) AS avg_total_salary
)
"""

cursor.execute(department_salary)

results = cursor.fetchall()

print("Question16:")
for row in results:
    print(row)
print("\n")

#  List the names of instructors along with the course ID of the courses that
they taught
instructor_name_with_courseID = """
```

```
SELECT instructor.name, teaches.course_id
FROM instructor
JOIN teaches ON instructor.ID = teaches.ID
"""

cursor.execute(instructor_name_with_courseID)

results = cursor.fetchall()

print("Question17:")
for row in results:
    print(row)
print("\n")

#  List the names of instructors along with the course ID of the courses that they taught. In case, an instructor teaches no courses keep the course ID as null.
instructor_name_with_courseID_with_null = """
SELECT instructor.name, teaches.course_id
FROM instructor
LEFT JOIN teaches ON instructor.ID = teaches.ID
"""

cursor.execute(instructor_name_with_courseID_with_null)

results = cursor.fetchall()

print("Question18:")
for row in results:
    print(row)
print("\n")
```

EXPERIMENT 8

1. Create a view of instructors without their salary called faculty
2. Create a view of department salary totals
3. Create a role of student
4. Give select privileges on the view faculty to the role student.
5. Create a new user and assign her the role of student.
6. Revoke privileges of the new user
7. Remove the role of student.
8. Give select privileges on the view faculty to the new user.
9. Create table teaches2 with same columns as teaches but with additional constraint that that semester is one of fall, winter, spring or summer.
10. Create index ID column of teaches. Compare the difference in time to obtain query results with or without index.
11. Drop the index to free up the space.

```
import mysql.connector

conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='root123',
    database='exp6'
)

cursor = conn.cursor()

# Create a view of instructors without their salary called faculty
instructors_view_without_salary = """
CREATE VIEW faculty AS
SELECT ID, name, dept_name
FROM instructor
"""

cursor.execute(instructors_view_without_salary)

display_instructor_view = """
SELECT *
FROM faculty
"""

cursor.execute(display_instructor_view)

results = cursor.fetchall()

print("Question1:")
for row in results:
    print(row)
```

```
print("\n")

#  Create a view of department salary totals
department_salary_view = """
CREATE VIEW department_salary_totals AS SELECT dept_name, SUM(salary) AS
total_salary FROM instructor GROUP BY dept_name
"""

cursor.execute(department_salary_view)

display_department_view="""
SELECT * FROM department_salary_totals;
"""

cursor.execute(display_department_view)

results = cursor.fetchall()

print("Question2:")
for row in results:
    print(row)
print("\n")

#  Create a role of student
role= """
CREATE ROLE 'student';
"""

cursor.execute(role)

# Give select privileges on the view faculty to the role student.
grant_select = """
GRANT SELECT ON faculty TO student;
"""
cursor.execute(grant_select)

#  Create a new user and assign her the role of student.
new_role = """
CREATE USER guru@localhost IDENTIFIED BY '1234'
"""

cursor.execute(new_role)

grant_user = """
GRANT student TO guru@localhost
"""

cursor.execute(grant_user)
```

```
#  Revoke privileges of the new user
revoke_user = """
REVOKE student FROM guru@localhost
"""

cursor.execute(revoke_user)

#  Remove the role of student.
remove_role = """
DROP ROLE student
"""

cursor.execute(remove_role)

# Give select privileges on the view faculty to the new user
select_user = """
GRANT SELECT ON faculty TO guru@localhost
"""

cursor.execute(select_user)

#  Create table teaches2 with same columns as teaches but with additional
constraint that that semester is one of fall, winter, spring or summer.
new_table= """
CREATE TABLE teaches2 (
  ID INT NOT NULL,
  course_id VARCHAR(255) NOT NULL,
  sec_id INT NOT NULL,
  semester VARCHAR(255) NOT NULL CHECK (semester IN ('Fall', 'Winter',
'Spring', 'Summer')),
  year INT NOT NULL,
  FOREIGN KEY (ID) REFERENCES instructor(ID)
)
"""

cursor.execute(new_table)

#  Create index ID column of teaches. Compare the difference in time to obtain
query results with or without index.
create_index = """
CREATE INDEX idx_ID ON teaches (ID)
"""

cursor.execute(create_index)

#  Drop the index to free up the space.
drop_index = """
```

```
DROP INDEX idx_ID ON teaches
"""
cursor.execute(drop_index)
```