PONDICHERRY UNIVERSITY

(A Central university)

SCHOOL OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

M.Sc. Computer Science

NAME	:	ELITAM SHIRISHA
REG. NO.	:	23370019
SEMESTER	:	II- Semester
SUBJECT	:	CSSC 424 – DATABASE SYSTEM LAB

PONDICHERRY UNIVERSITY

(A Central university)

SCHOOL OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

M.Sc. Computer Science

PRACTICAL LAB RECORD

BONAFIDE CERTIFICATE

This is to certify that this is a Bonafide record of practical work done by **ELITAM SHIRISHA**, having Reg. No. **23370019** semester - II from the month February 2024 to June 2024.

FACULTY IN-CHARGE

SUBMITTED FOR THE PRACTICAL EXAM HELD ON: _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

EXPERIMENT 1

```
create database customer;
create table customer(
CNO varchar(30) primary key,
NAME VARCHAR(30),
ADDRESS VARCHAR(30),
CITY VARCHAR(30)
);
INSERT INTO customer(CNO,NAME,ADDRESS,CITY)
VALUES
("C1","Rahat","Thapar Nagar","Patiala"),
("C2","Rhuhi","Tagore Nagar","Jalandhar"),
("C3","Chatan","Dharamapura","Qadian"),
("C4","Pooja","GNDU","Amritsar");
```

--Query 1:

select *from customer;

Result Grid | Filter Rows: | Edit:

CNO	NAME	ADDRESS	CITY
C1	Rahat	Thapar Nagar	Patiala
C2	Rhuhi	Tagore Nagar	Jalandhar
C3	Chatan	Dharamapura	Qadian
C4	Pooja	GNDU	Amritsar
NULL	NULL	NULL	NULL

```
create table customer_loan(
CNO varchar(30),
LNO varchar(30),
AMOUNT INT
);
INSERT INTO customer_loan(CNO,LNO,AMOUNT)
VALUES
("C1","L1","10000"),
("C2","L1","10000"),
("C3","L2","15000"),
("C3","L3","25000"),
("C4","L4","35000");
--Query 2:
select *from customer_loan;
```

Result Grid Filter Rows:

CNO	LNO	AMOUNT
C1	L1	10000
C2	L1	10000
C3	L2	15000
C3	L3	25000
C4	L4	35000

EXPERIMENT 2

```
create database siri;
use siri;

create table salesman(
salesman_id int primary key,
name varchar(30),
city varchar(30),
comission float
);
insert into salesman (salesman_id,name,city,comission)
values
(5001,"James Hoog","New York",0.15),
(5002,"Nail Knite","Paris",0.13),
(5005,"Pit Alex","London",0.11),
(5006,"MC Lyon","Paris",0.14),
(5003,"Lauson Hen",null,0.12),
(5007,"Paul Adam","Rome",0.13);
```

--Query 1:

```
select*from salesman;
```

Result Grid | Filter Rows: | Ed

salesman_id	name	city	comission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5003	Lauson Hen	NULL	0.12
5005	Pit Alex	London	0.11
5006	MC Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
NULL	NULL	NULL	NULL

```
create table customer1(
customer_id int,
customer_name varchar(30),
city varchar(30),
grade int,
salesman_id int,
primary key (customer_id),
foreign key (salesman_id) references salesman (salesman_id)
);
insert into customer1(customer_id,customer_name,city,grade,salesman_id)
values
(3002,"Nick Rimando","New York",100,5001),
(3005,"Graham Zusi","California",200,5002),
(3001,"Brad Guzan","London",null,null),
(3004,"Fabian Johns","Paris",300,5006),
(3007,"Brad Davis","New York",200,5001),
(3009,"Geoff Camero","Berlin",100,null),
(3008,"Julian Green","London",300,5002),
(3003,"Jozy Altidor","Mancow",200,5007);
```

--Query 2:

```
select *from customer1;
```

Result Grid | Filter Rows: | Edit:

customer_id	customer_name	city	grade	salesman_id
3001	Brad Guzan	London	NULL	NULL
3002	Nick Rimando	New York	100	5001
3003	Jozy Altidor	Mancow	200	5007
3004	Fabian Johns	Paris	300	5006
3005	Graham Zusi	California	200	5002
3007	Brad Davis	New York	200	5001
3008	Julian Green	London	300	5002
3009	Geoff Camero	Berlin	100	NULL
NULL	NULL	NULL	NULL	NULL

```
create table order1(
order_no int,
purch_no float,
order_date date,
customer_id int,
salesman_id int);
insert into order1(order_no,purch_no,order_date,customer_id,salesman_id)
values
(70001,150.5,"2016-10-05",3005,5002),
(70009,270.5,"2016-09-10",3001,null),
(70002,65.5,"2016-10-05",3002,5001),
(70004,110.5,"2016-08-17",3009,null),
(7007,948.5,"2016-09-10",3005,5002),
(70005,2400.6,"2016-07-27",3007,5001),
(70008,5760,"2016-09-10",3002,5001),
(70010,19830.43,"2016-10-10",3004,5006),
(70003,2480,"2016-10-10",3009,null);
```

--Query 3:

select*from order1;

Result Grid | Filter Rows: | Edit:

customer_id	customer_name	city	grade	salesman_id
3001	Brad Guzan	London	NULL	NULL
3002	Nick Rimando	New York	100	5001
3003	Jozy Altidor	Mancow	200	5007
3004	Fabian Johns	Paris	300	5006
3005	Graham Zusi	California	200	5002
3007	Brad Davis	New York	200	5001
3008	Julian Green	London	300	5002
3009	Geoff Camero	Berlin	100	NULL
NULL	NULL	NULL	NULL	NULL

--Query 4:

select name,comission from salesman;

Result Grid | Filter Rows:

name	comission
James Hoog	0.15
Nail Knite	0.13
Lauson Hen	0.12
Pit Alex	0.11
MC Lyon	0.14
Paul Adam	0.13

--Query 5:

select distinct salesman_id from order1;

Result Grid

salesman_id
5002
NULL
5001
5006
5007

--Query 6:

select name,city from salesman where city="paris

Result Grid Filter Rows

name	city
Nail Knite	Paris
MC Lyon	Paris

--Query 7:

select * from customer1 where grade=200;

Result Grid Filter Rows: Edit:

customer_id	customer_name	city	grade	salesman_id
3003	Jozy Al Jozy Altidor	ncow	200	5007
3005	Graham Zusi	California	200	5002
3007	Brad Davis	New York	200	5001
NULL	NULL	NULL	NULL	NULL

--Query 8:

select order_no,order_date,purch_no from order1 where salesman_id=5001;

Result Grid | Filter Rows:

order_no	order_date	purch_no
70002	2016-10-05	65.5
70005	2016-07-27	2400.6
70008	2016-09-10	5760

--Query 9:

select order_no from order1 where order_date = "2016-10-05" and salesman_id = 5002;

Result Grid | Filter

order_no
70001

--Query 10:

select*from order1 where salesman_id like 5001;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_no	purch_no	order_date	customer_id	salesman_id	amount
70002	65.5	2016-10-05	3002	5001	NULL
70005	2400.6	2016-07-27	3007	5001	NULL
70008	5760	2016-09-10	3002	5001	NULL

EXPERIMENT 3

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);
```

--Query 1:

SELECT * FROM instructor;

Result Grid | Filter Rows:

ID	name	dept_name	salary
10101	Srinivasan	Comp.Sci	65000
10212	Tom	Biology	NULL
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	95000
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
NULL	NULL	NULL	NULL

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);
```

--Query 2:

```
SELECT * FROM teaches;
```

Result Grid Filter Rows:

ID	name	dept_name	salary
10101	Srinivasan	Comp.Sci	65000
10212	Tom	Biology	NULL
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	95000
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
NULL	NULL	NULL	NULL

--Query 3:

INSERT INTO instructor VALUES ('10211', 'Smith', 'Biology', 66000);

Successfully inserted

--Query 4:

DELETE FROM instructor WHERE ID=10211;

Successfully deleted

--Query 5:

SELECT * FROM instructor where dept_name='History';

Result Grid Filter Rows:

ID	name	dept_name	salary
32343	El Said	History	60000
58583	Califieri	History	62000
NULL	NULL	History	NULL

--Query 6:

SELECT * FROM instructor CROSS JOIN teaches;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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	62000	10101	CS-101	1	Fall	2017
45565	Katz	Comp.Sci	75000	10101	CS-101	1	Fall	2017
33456	Gold	Physics	95000	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
10212	Tom	Biology	NULL	10101	CS-101	1	Fall	2017
10211	Smith	Biology	66000	10101	CS-101	1	Fall	2017
10101	Sriniva...	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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	dept_name	salary	ID	Course_id	sec_id	semester	year
76543	Singh	Finance	80000	10101	CS-315	1	Spring	2018
58583	Califieri	History	62000	10101	CS-315	1	Spring	2018
45565	Katz	Comp.Sci	75000	10101	CS-315	1	Spring	2018
33456	Gold	Physics	95000	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
10212	Tom	Biology	NULL	10101	CS-315	1	Spring	2018
10211	Smith	Biology	66000	10101	CS-315	1	Spring	2018
10101	Sriniva...	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	62000	10101	CS-347	1	Fall	2017
45565	Katz	Comp.Sci	75000	10101	CS-347	1	Fall	2017

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	dept_name	salary	ID	Course_id	sec_id	semester	year
33456	Gold	Physics	95000	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
10212	Tom	Biology	NULL	10101	CS-347	1	Fall	2017
10211	Smith	Biology	66000	10101	CS-347	1	Fall	2017
10101	Sriniva...	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	62000	12121	FIN-201	1	Spring	2018
45565	Katz	Comp.Sci	75000	12121	FIN-201	FIN-201	Spring	2018
33456	Gold	Physics	95000	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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	name	dept_name	salary	ID	Course_id	sec_id	semester	year
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
10212	Tom	Biology	NULL	12121	FIN-201	1	Spring	2018
10211	Smith	Biology	66000	12121	FIN-201	1	Spring	2018
10101	Sriniva...	Comp.Sci	65000	12121	FIN-201	1	Spring	2018
98345	Kim	Elec.Eng	80000	15151	MU-199	1	Spring	2018
83821	Brandt	Comp.Sci	92000	15151	MU-199	1	Spring	2018
76766	Crick	Biology	72000	15151	MU-199	1	Spring	2018
76543	Singh	Finance	80000	15151	MU-199	1	Spring	2018
58583	Califieri	History	62000	15151	MU-199	1	Spring	2018
45565	Katz	Comp.Sci	75000	15151	MU-199	1	Spring	2018
33456	Gold	Physics	95000	15151	MU-199	1	Spring	2018
32343	El Said	History	60000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
10212	Tom	Biology	NULL	15151	MU-199	1	Spring	2018

--Query 7:

SELECT i.name, t.course_id FROM instructor i INNER JOIN teaches t on i.ID= t.ID;

Result Grid | Filter Rows:

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
Kim	EE-181

--Query 8:

SELECT name FROM instructor where name LIKE "%dar%";

No table

--Query 9:

SELECT name FROM instructor where salary>= 90000 AND salary<=100000;

Result Grid

name
Wu
Einstein
Gold
Brandt

EXPERIMENT 4

--Query 1:
SELECT * FROM instructor ORDER BY salary;

Result Grid Filter Rows:

ID	name	dept_name	salary
10212	Tom	Biology	NULL
15151	Mozart	Music	40000
32343	El Said	History	60000
58583	Califieri	History	62000
10101	Srinivasan	Comp.Sci	65000
10211	Smith	Biology	66000
76766	Crick	Biology	72000
45565	Katz	Comp.Sci	75000
76543	Singh	Finance	80000
98345	Kim	Elec.Eng	80000
12121	Wu	Finance	90000
83821	Brandt	Comp.Sci	92000
22222	Einstein	Physics	95000
33456	Gold	Physics	95000
NULL	NULL	NULL	NULL

--Query 2:

SELECT DISTINCT course_id FROM teaches WHERE (semester='Fall'and year=2017)OR (semester='Spring' and year=2018);

Result Grid | Filter

course_id
CS-101
CS-347
PHY-101
CS-315
FIN-201
MU-199
HIS-351
CS-319

--Query 3:
SELECT DISTINCT course_id FROM teaches WHERE (semester='Fall'and year=2017) AND (semester='Spring' and year=2018);

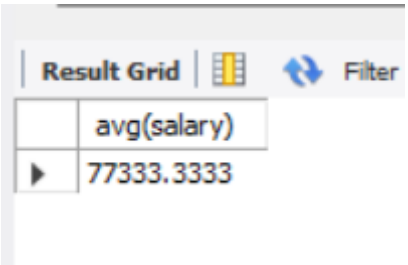
No Table

--Query 4:
select course_id from teaches where (semester ="Fall" and year=2017) and not (semester ="Spring" and year=2018);

Result Grid | Filter Rows:

ID	name	dept_name	salary
10212	Tom	Biology	NULL
NULL	NULL	NULL	NULL

--Query 5:
select avg(salary) from instructor where dept_name="Comp.Sci";



EXPERIMENT 5

--Query 1:

```
SELECT COUNT(DISTINCT ID) AS total_instructors FROM teaches
WHERE semester='Spring' AND year=2018;
```

Result Grid | Filter

	count(distinct ID)
▶	6

--Query 2:

```
relationSELECT COUNT(*) AS
num_tuples FROM teaches;
```

Result Grid

	count(*)
▶	15

15

--Query 3:

```
SELECT dept_name, AVG(salary) as avg_salary FROM instructor GROUP
BY dept_name;
```

Result Grid | Filter Rows:

	dept_name	avg_salary
▶	Comp.Sci	77333.3333
	Biology	69000.0000
	Finance	85000.0000
	Music	40000.0000
	Physics	95000.0000
	History	61000.0000
	Elec.Eng	80000.0000

--Query 4:

```
SELECT dept_name, AVG(salary) as avg_salary FROM instructor GROUP
BY dept_nameHAVING AVG(salary)>42000;
```

Result Grid | Filter Rows:

dept_name	avg_salary
Comp.Sci	77333.3333
Biology	69000.0000
Finance	85000.0000
Music	40000.0000
Physics	95000.0000
History	61000.0000
Elec.Eng	80000.0000

--Query 5:

```
SELECT name FROM instructor WHERE name NOT IN
("Mozart","Einstein");
```

Result Grid

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

--Query 6:

```
SELECT l.name FROM instructor l WHERE l.salary > (SELECT salary
FROM instructorWHERE dept_name='Biology' AND name="Crick");
```

Result Grid

name
Wu
Einstein
Gold
Katz
Singh
Crick
Brandt
Kim

--Query 7:

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

Result Grid

name
Wu
Einstein
Gold
Katz
Singh
Brandt
Kim

--Query 8:

```
SELECT dept_name, AVG(salary) as average_salary FROM
instructor GROUP BY dept_name HAVING AVG(salary)>42000;
```

Result Grid | Filter Rows:

dept_name	avg_salary
Comp.Sci	77333.3333
Biology	69000.0000
Finance	85000.0000
Physics	95000.0000
History	61000.0000
Elec.Eng	80000.0000

EXPERIMENT 6

--Query 1:

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

Result Grid

dept_name
Comp.Sci
Biology
Finance
Physics
History
Elec.Eng

--Query 2:

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

Result Grid | Filter Ro

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
Kim	EE-181

--Query 3:

```
SELECT i.name AS instructor_name, t.course_id

FROM instructor i LEFT JOIN teaches t ON i.ID = t.ID;
```

Result Grid | Filter Rows:

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

Result 23

--Query 4:

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

Result Grid | Filter Rows:

ID	name	dept_name
10101	Srinivasan	Comp.Sci
10211	Smith	Biology
10212	Tom	Biology
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

--Query 5:

GRANT SELECT ON faculty TO new_user;

Successful

EXPERIMENT 7

--Query 1:

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

SELECT * FROM faculty1;

Result Grid | Filter Rows:

ID	name	dept_name
10101	Srinivasan	Comp.Sci
10211	Smith	Biology
10212	Tom	Biology
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

--Query 2:

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;

Result Grid | Filter Row

dept_name	TotalSalary
Comp.Sci	232000
Biology	138000
Finance	170000
Music	40000
Physics	190000
History	122000
Elec.Eng	80000

--Query 3:

CREATE ROLE student;

Successful

--Query 4:

GRANT SELECT ON faculty TO student;

Successful

--Query 5:

CREATE USER raju@localhost
IDENTIFIED BY '1234'; GRANT student TO
raju@localhost;

Successful

--Query 6:

GRANT ALL PRIVILEGES ON student.* TO
raju@localhost;

SELECT * FROM faculty WHERE dept_name = 'Biology';

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

--Query 7:

REVOKE student FROM
raju@localhost;
Successful

--Query 8:

DROP ROLE
student;
Successful

--Query 9:

GRANT SELECT ON faculty TO
raju@localhost;

Successful

--Query 10:

```
SELECT * FROM faculty WHERE dept_name =
'Finance';
```

ID	name	dept_name
12121	Wu	Finance
76543	Singh	Finance

--Query 11:

Login again as root user

--Query 12:

```
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)

);
```

--Query 13:

```
CREATE INDEX idx_ID ON teaches (ID);
```

--Query 14:

```
DROP INDEX idx_ID ON
teaches;
```

EXPERIMENT 8

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

Source Code:

```
import mysql.connector
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='surya@sql1',
    database='example'
)
cursor = conn.cursor()
# 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,))
```

```python
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
"""
```

```python
cursor.execute(query)

results = cursor.fetchall()

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

conn.commit()

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

Output:

```
PS C:\Users\D A GURUPRIYAN\Downloads\ADBMS> & "c:/Users/D A GURUPRIYAN/Downloads/ADBMS/.venv/Scr
Question 3
(32343, 'El Said', 'History', 60000)
(58583, 'Califieri', 'History', 62000)


Question 4
(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', 62000, 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', 62000, 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', 62000, 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', 62000, 12121, 'FIN-201', 1, 'Spring', 2018)
(45565, 'Katz', 'Comp. Sci.', 75000, 12121, 'FIN-201', 1, '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', 62000, 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', 62000, 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)


Question 5
('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-319')
('Kim', 'EE-181')


Question 6


Question 7
Wu
Einstein
Brandt
```

EXPERIMENT 9

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.

Source Code:

```
import mysql.connector
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='surya@sql1',
    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)
"""
```

```python
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")
```

```python
#  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)
```

```python
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
"""
```

```python
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)
```

```python
results = cursor.fetchall()
print("Question17:")
for row in results:
    print(row)
print("\n")
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")
```

Output:

```
PS C:\Users\D A GURUPRIYAN\Downloads\ADBMS> & "c:/Users/D A GURUPRIYAN/Downl
Question1:
(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')


Question2:
('Comp. Sci.', Decimal('232000'))
('Finance', Decimal('170000'))
('Music', Decimal('40000'))
('Physics', Decimal('182000'))
('History', Decimal('122000'))
('Biology', Decimal('72000'))
('Elec. Eng', Decimal('80000'))
```

EXPERIMENT 10

```
-- query 1
CREATE TYPE addr_ty AS OBJECT
  2  (street    varchar2(60),
  3  city       varchar2(30),
  4  state      char(2),
  5  zip        varchar(9));
  6  /
Type created.

SQL> CREATE TYPE person_ty AS OBJECT
  2     (name   varchar2(25),
  3     address  addr_ty);
  4  /
Type created.

SQL> CREATE TYPE emp_ty AS OBJECT
  2     (empt_id        varchar2(9),
  3     person  person_ty);
  4
  5  /
Type created.

-- query 2
SQL> CREATE TABLE EMP_OO
  2     (full_emp emp_ty);
```

```
-- query 3

-- insert

insert into EMP_OO values( emp_ty('100', person_ty('raju', addr_ty('100
st','Patiala','up','605001'))));

insert into EMP_OO values( emp_ty('101', person_ty('siri', addr_ty('101
st','sire','Blore','105001'))));


-- query 4

-- select

select * from emp_oo;


FULL_EMP(EMPT_ID, PERSON(NAME, ADDRESS(STREET, CITY,
STATE, ZIP)))

--------------------------------------------------------------------------------

EMP_TY('100', PERSON_TY('Raju', ADDR_TY('1000 st', 'Patiala', 'up',
'605001')))

EMP_TY('101', PERSON_TY('siri', ADDR_TY('1001 st', 'sire', 'AP',
'105001')))


select e.full_emp.empt_id ID,e.full_emp.person.name NAME,
e.full_emp.person.address.city CITY from emp_oo e;


ID        NAME                    CITY

--------- ------------------------ ------------------------------

100       Raju                    Patiala

101       siri                    sire
```

```
-- query 5
-- update
update emp_oo e set e.full_emp.person.name = 'Raju' where e.full_emp.empt_id
= '1000';


-- query 6
-- create new obj with member function
create or replace type newemp_ty as object (firstname varchar2(25),
lastname Varchar2(25), birthdate Date, member function age (birthdate in date)
return number);



-- query 7
create or replace type body newemp_ty as
        member function age(birthdate in date) return number is
        begin
                return round(sysdate - birthdate);
        end;
end;


-- query 8
create table new_emp_oo (employee newemp_ty);


-- query 9
insert into new_emp_oo values(newemp_ty('raju', 'lal','1976-12-12'));
```

-- query 10

select e.employee.firstname, e.employee.age, e.employee.age(e.employee.birthdate) from new_emp_oo e;

-- query 11

create table new_emp1 of emp_ty;

-- query 12

insert into new_emp1 values('102',person_ty('rajitha',addr_ty('100 TU', 'Pta','PB', '147002'))));

-- query 13

select * from new_emp1;

PERSON_TY('rajitha', ADDR_TY('100 TU', 'Pta', 'PB', '147002'))

-- query 14 references

select ref(p) from new_emp1 p;

REF(P)

0000280209E44C561C843C4E90B9AB35A22AD3E8FBAFAB0D508DDF493C87F3A6F19DC6804F0041DC

C90000

-- query 15 implementing the concept of fk

create type new_dept_oo as object (deptno number(3),dname varchar(10));

```
-- query 16
create table dept_table of new_dept_oo;

-- query 17
insert into dept_table values (10,'comp');
insert into dept_table values (20,'chem');
insert into dept_table values (30,'math');

-- query 18
create table emp_test_fk(empno number(3), name varchar2(10), dept ref
new_dept_oo);

-- query 19
set desc depth 2
desc emp_test_fk

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(3)
 NAME                                               VARCHAR2(10)
 DEPT                                               REF OF NEW_DEPT_OO
 DEPTNO                                             NUMBER(3)
 DNAME                                              VARCHAR2(10)
```

-- query 20

insert into emp_test_fk select 100, 'raju', ref(p) from dept_table p where deptno =10;

insert into emp_test_fk select 101, 'siri', ref(p) from dept_table p where deptno = 20;

-- query 21 accessing values

select empno, name, deref(e.dept) from emp_test_fk e;

EMPNO NAME

---------- ----------

DEREF(E.DEPT)(DEPTNO, DNAME)

---

100 raju

NEW_DEPT_OO(10, 'comp')

101 siri

NEW_DEPT_OO(20, 'chem')

select empno, name, deref(e.dept), deref(e.dept).deptno DEPTNO,deref(e.dept).dname DNAME from emp_test_fk e;

EMPNO NAME

---------- ----------

DEREF(E.DEPT)(DEPTNO, DNAME)

DEPTNO DNAME

---------- ----------

100 raju

NEW_DEPT_OO(10, 'comp') 10 comp

101 siri

NEW_DEPT_OO(20, 'chem') 20 chem

EMPNO NAME

---------- ----------

DEREF(E.DEPT)(DEPTNO, DNAME)

DEPTNO DNAME

---------- ----------

-- query 22

create table emp_table_fk (employee emp_ty, dept ref new_dept_oo);

set desc depth 2

-- query 23

insert into emp_table_fk values (emp_ty('100', person_ty('ram', addr_ty('100 st','Patiala','up','605001'))), (select ref(p) from dept_table p where deptno = 10));

-- query 24

select * from em_table_fk;

EMPLOYEE(EMPT_ID, PERSON(NAME, ADDRESS(STREET, CITY, STATE, ZIP)))

DEPT

EMP_TY('100', PERSON_TY('ram', ADDR_TY('100 st', 'Patiala', 'up', '605001')))

00002202088ECB5F5DB94A44CD901A1BACD0D508D64D9EE4FAD8EF44
04B2D19B5A449B8463

```
select e.employee.empt_id ID, e.employee.person.name NAME, deref(e.dept),
deref(e.dept).deptno DEPTNO,deref(e.dept).dname DNAME from
emp_table_fk e;

ID        NAME
--------- -------------------------
DEREF(E.DEPT)(DEPTNO, DNAME)
--------------------------------------------------------------------------------
   DEPTNO DNAME
---------- ----------
100       ram
NEW_DEPT_OO(10, 'comp')
        10 comp
```