PONDICHERRY UNIVERSITY
(A Central university)

SCHOOL OF ENGINEERING AND
TECHNOLOGY DEPARTMENT OF
COMPUTER SCIENCE
M.Sc. Computer Science

NAME : FATHIMA RASHEED

REG. NO. : 23370020

SEMESTER : 1st - Semester

SUBJECT : CSSC 424 – DATABASE SYSTEM LAB

PONDICHERRY UNIVERSITY

SCHOOL OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this is a Bonafide record of practical work done by FATHIMA RASHEED, having Reg. No. 23370020 semester - 1 from the month February 2024 to June 2024.

FACULTY IN-CHARGE

SUBMITTED FOR THE PRACTICAL EXAM HELD ON: _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

INDEX

EX. No	DATE	TITLE	PAGE	SIGNATURE
1	07/03/24	Experiment 1	4	
2	20/03/24	Experiment 2	13	
3	27/03/24	Experiment 3	15	
4	03/04/24	Experiment 4	17	
5	03/04/24	Experiment 5	27	
6	03/04/24	Experiment 6	30	
7	10/04/24	Experiment 7	33	
8	10/04/24	Experiment 8	36	

9	10/04/24	Experiment 9	39	
10	17/04/24	Experiment 10	47	
11	17/04/24	Experiment 11	63	
12	15/05/24	Experiment 12	69	

EXPERIMENT 1

SQL Practice 1

```
create
database
vamsi; use
vamsi;
create table salesman(salesman_id int primary key,name
        varchar(30),city varchar(30),commission float);
insert into salesman (salesman_id,name,city,commission)
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);
```

```
11 •  select*from salesman;
12
```

Result Grid | Filter Rows: | Edit:

salesman_id	name	city	commission
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

salesman 1 ×

create table customer(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);

```
24 •    select *from customer;
25
```

Result Grid | Filter Rows: | Edit:

customer_id	customer_name	city	grade	sale
3001	Brad Guzan	London	NULL	NULL
3002	Nick Rimando	New York	100	500
3003	Jozy Altidor	Mancow	200	500
3004	Fabian Johns	Paris	300	500
3005	Graham Zusi	California	200	500
3007	Brad Davis	New York	200	500
3008	Julian Green	London	300	500
3009	Geoff Camero	Berlin	100	NULL
NULL	NULL	NULL	NULL	NULL

customer 3 ×

```
create table order1(order_no int,purch_amt float,order_date
        date,customer_id int,salesman_id int);
insert into
order1(order_no,purch_amt,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",30
04,5006),
(70003,2480,"2016-10-10",3009,n
ull);
```

```
38 •    select*from order1;
39
```

Result Grid | Filter Rows: | Export:

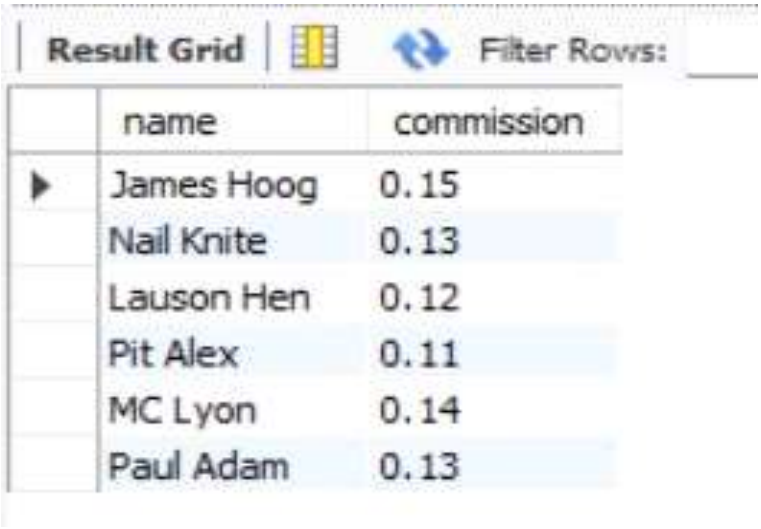
order_no	purch_amt	order_date	customer_id	salesm
70001	150.5	2016-10-05	3005	5002
70009	270.5	2016-09-10	3001	NULL
70002	65.5	2016-10-05	3002	5001
70004	110.5	2016-08-17	3009	NULL
7007	948.5	2016-09-10	3005	5002
70005	2400.6	2016-07-27	3007	5001
70008	5760	2016-09-10	3002	5001
70010	19830.4	2016-10-10	3004	5006
70003	2480	2016-10-10	3009	NULL
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	19830.4	2016-10-10	3004	5006
70003	2480	2016-10-10	3009	NULL

order1 4 ×

Query 1

- Display name and commission of all the salesmen.

select name,commission from salesman;



	name	commission
▶	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 2

- Retrieve salesman id of all salesmen from orders table without any repeats.

select distinct salesman_id from order1;

	salesman_id
▶	5002
	NULL
	5001
	5006

Query 3

- Display names and city of salesman, who belongs to the city of Paris.

select name,city from salesman where city="paris";

Result Grid | Filter Rows:

name	city
Nail Knite	Paris
MC Lyon	Paris

Query 4

- Display all the information for those customers with a grade of 200.

select * from customer where grade=200;

Result Grid | Filter Rows:

customer_id	customer_name	city	grade	s
3003	Jozy Altidor	Mancow	200	5(
3005	Graham Zusi	California	200	5(
3007	Brad Davis	New York	200	5(
NULL	NULL	NULL	NULL	NU

Query 5

- Display the order number, order date and the purchase amount for order(s) which will be delivered by the salesman with ID 5001.

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

Result Grid | Filter Rows:

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

Query 6 (table: customer)

- Display all the customers, who are either belongs to the city New York or not had a grade above 100.

select*from customer where city='New York' or not grade>100;

Result Grid | Filter Rows:

customer_id	customer_name	city	grade
3002	Nick Rimando	New York	100
3007	Brad Davis	New York	200
3009	Geoff Camero	Berlin	100
NULL	NULL	NULL	NULL

Query 7 (table: salesman)

- Find those salesmen with all information who gets the

commission within a range of 0.12 and 0.14.

select*from salesman where (0.12<commission>0.14);

Result Grid | Filter Rows: | Edi

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5006	MC Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
NULL	NULL	NULL	NULL

select*from salesman where(commission between 0.12 and 0.14);

Result Grid | Filter Rows: | Edit

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5006	MC Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
NULL	NULL	NULL	NULL

Query 8 (table: customer)

- Find all those customers with all information whose names are ending with the letter 'n'.

select*from customer where customer_name like '%n';

Query 9 (table: salesmen)

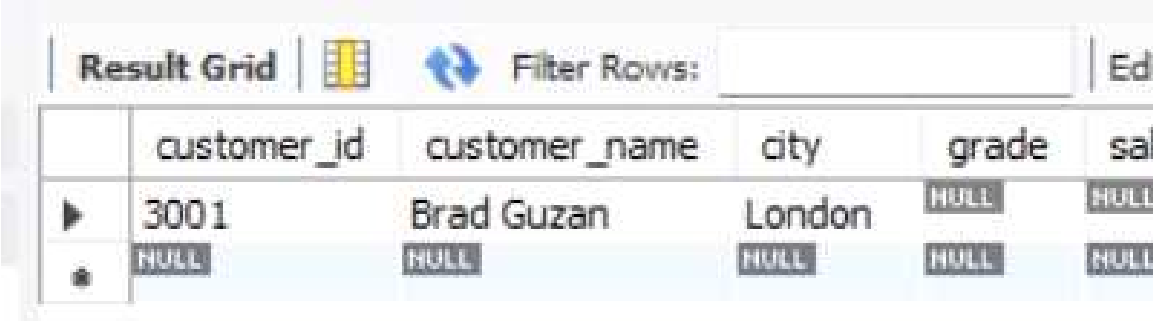
- Find those salesmen with all information whose name containing the 1st character is 'N' and the 4th character is 'l' and rests may be any character.

select*from salesman where name like 'n_l%';

Query 10 (table: customer)

- Find that customer with all information who does not get any grade except NULL.

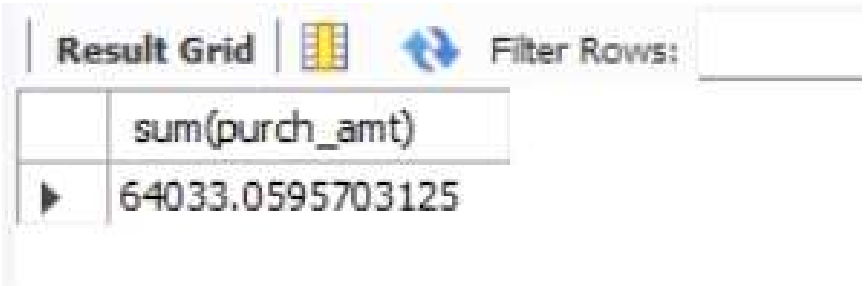
select*from customer where grade is Null;



Query 11 (table: orders)

- Find the total purchase amount of all orders.

select sum(purch_amt) from order1;



Query 12 (table: orders)

- Find the number of salesman currently listing for all of their customers.

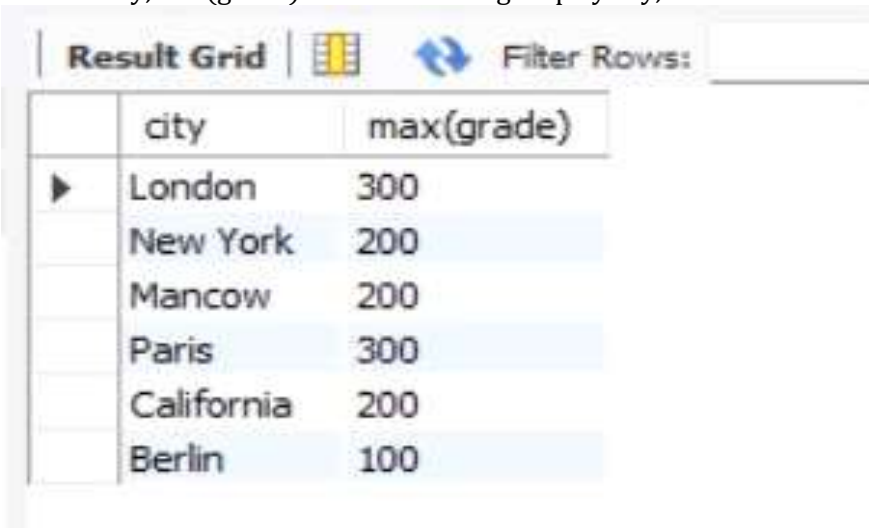
select count(salesman_id) from customer;

select count(distinct salesman_id) from order1;

Query 13 (table: customer)

- Find the highest grade for each of the cities of the customers.

select city,max(grade) from customer group by city;



Query 14 (table: orders)

- Find the highest purchase amount ordered by the each customer with their ID and highest purchase amount.

select customer_id,max(purch_amt) from order1 group by customer_id;

Result Grid | Filter Rows:

customer_id	max(purch_amt)
3005	948.5
3001	270.5
3002	5760
3009	2480
3007	2400.6
3004	19830.4

Query 15 (table: orders)

- Find the highest purchase amount ordered by the each customer on a particular date with their ID, order date and highest purchase amount.

```
select customer_id, order_date, max(purch_amt)
from order1 group by customer_id, order_date;
```

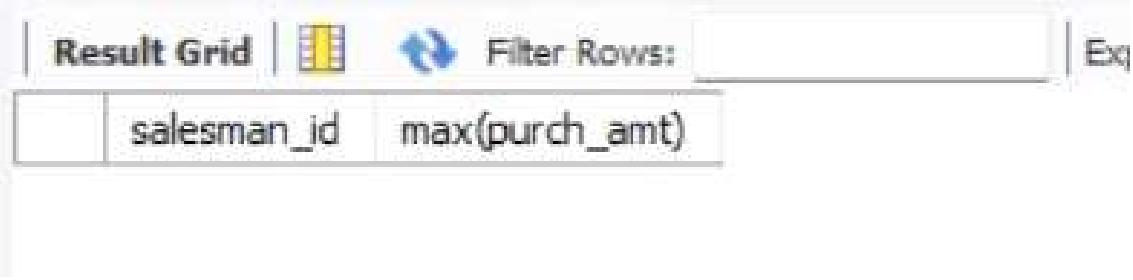
Result Grid | Filter Rows:

customer_id	order_date	max(purch_amt)
3005	2016-10-05	150.5
3001	2016-09-10	270.5
3002	2016-10-05	65.5
3009	2016-08-17	110.5
3005	2016-09-10	948.5
3007	2016-07-27	2400.6
3002	2016-09-10	5760
3004	2016-10-10	19830.4
3009	2016-10-10	2480

Query 16 (table: orders)

- Find the highest purchase amount on a date '2012-08-17' for each salesman with their ID.

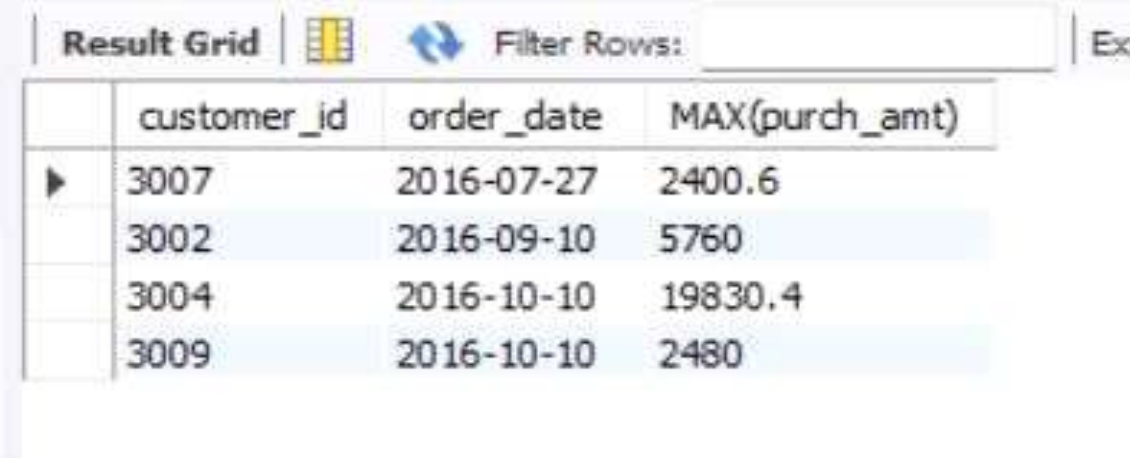
```
select salesman_id, max(purch_amt) from order1
where order_date = '2012-08-17' group by salesman_id;
```



Query 17 (table: orders)

- Find the highest purchase amount with their customer ID and order date, for only those customers who have the highest purchase amount in a day is more than 2000.

```
select customer_id, order_date, MAX(purch_amt) from order1
group by customer_id, order_date having max(purch_amt) > 2000.00;
```



Query 18 (table: orders)

- Write a SQL statement that counts all orders for a date

August 17th, 2012.

select count(*) from order1 where order_date = '2012-08-17';

EXPERIMENT 2

TRIGGER:-

-- Source code

```
create database
trigger1; use
trigger1;
-- Create student table
create table student(student_id integer null,name varchar(20),Address
varchar(20),Marks integer(10));
-- Describe
student table desc
student;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Ex
student_id	int	YES		NULL	
name	varchar(20)	YES		NULL	
Address	varchar(20)	YES		NULL	
Marks	int	YES		NULL	

```
-- create trigger
create trigger student_trigger before insert on student for each row set
new.Marks=new.Marks+100;
insert into
student(student_id,name,Address,Marks)
values('2','guru','landon','90');
insert into
student(student_id,name,Address,Marks)
values('3','akaksh','India','70');
-- Display student table
select*from student;
```

Result Grid | Filter Rows:

student_id	name	Address	Marks
2	guru	landon	190
3	akaksh	India	170

sql_mode	Definer	character_set_client	c
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	u

-- Display

trigger show

triggers;

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

	Trigger	Event	Table	Statement	Timing	Created
▶	student_trigger	INSERT	student	set new.Marks=new.Marks+100	BEFORE	2024-06-11 23:1

PROCEDURES:-

EXPERIMENT 3

```
create database procedures;
create table employees(emp_no integer primary key,e_name varchar(20),e_address varchar(20),e_ph_no varchar(20));
-- Insert table values
insert into employees values(101,'guru','11-ut',2894833793); insert into employees values(102,'akash','43-c',6788299204); insert into employees values(103,'khaif','41-or',7890489200); insert into employees values(104,'aniket','21-e',9899204782);
-- Create procedures without parameters DELIMITER $$
create  procedure get_employees () begin
select*from employees; end
$$
DELIMITER ;
-- Call procedure
call get_employees();
```

Result Grid | Filter Rows: | Expor

	emp_no	e_name	e_address	e_ph_no
▶	101	guru	11-ut	2894833793
	102	akash	43-c	6788299204
	103	khaif	41-or	7890489200
	104	aniket	21-e	9899204782

```
-- create procedures with
parameters DELIMITER $$
create  procedure finds_employees (in
id int) begin
select*from employees where emp_id = id;

end $$
DELIMIT
ER ;
```

```
call finds_employees(101);
```

Result Grid | Filter Rows: | Exp

	emp_no	e_name	e_address	e_ph_no
▶	101	guru	11-ut	2894833793

```
call finds_employees(104);
```

Result Grid | Filter Rows: | Exp

	emp_no	e_name	e_address	e_ph_no
▶	104	aniket	21-e	9899204782

```
call finds_employees(102);
```

	emp_no	e_name	e_address	e_ph_no
▶	102	akash	43-c	6788299204

EXPERIMENT 4

DATE :

- 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

- 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

- 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

- 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

- 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

- Find the Cartesian product

instructor x teaches. SELECT * FROM instructor CROSS JOIN teaches;

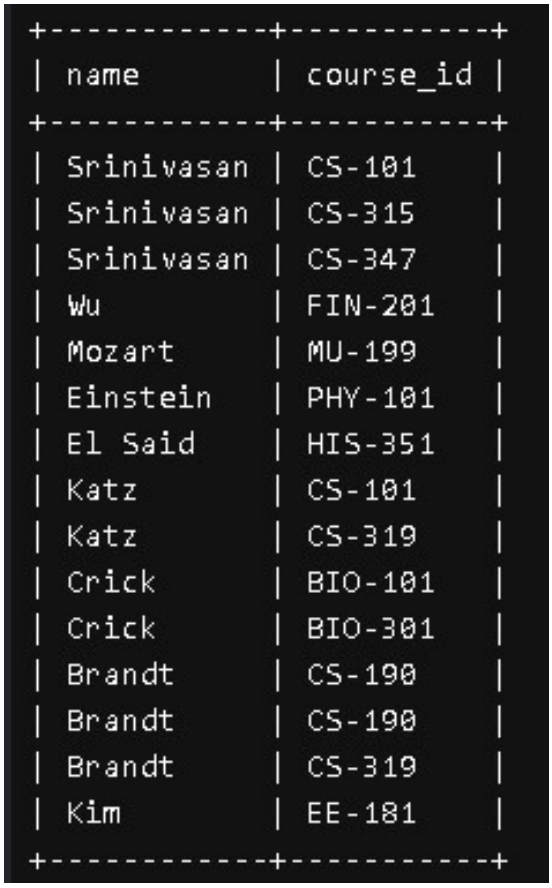
ID	name	dept_name	salary	ID	course_id	sec_id	semester	year
98345	Kim	Elec. Eng	80000	10101	CS-101	1	Fall	2017
83821	Brandt	Comp. Sci.	92000	10101	CS-101	1	Fall	2017
76766	Crick	Biology	72000	10101	CS-101	1	Fall	2017
76543	Singh	Finance	80000	10101	CS-101	1	Fall	2017
58583	Califieri	History	6200	10101	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

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

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

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

- 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

- Find the names of all instructors whose name includes the

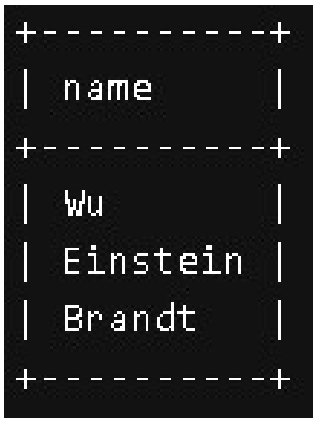
substring “dar”. SELECT name FROM instructor where name

LIKE "%dar%";

- 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 5

- 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

- 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

- 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);

- 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

- Insert following additional tuples in instructor :('10211', 'Smith', 'Biology',

66000), ('10212', 'Tom', 'Biology', NULL)

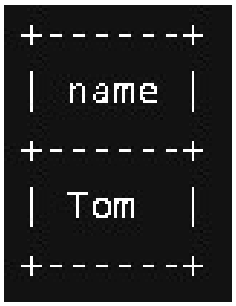
INSERT INTO instructor VALUES ('10211', 'Smith', 'Biology', 66000), ('10212', 'Tom', 'Biology', NULL);

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

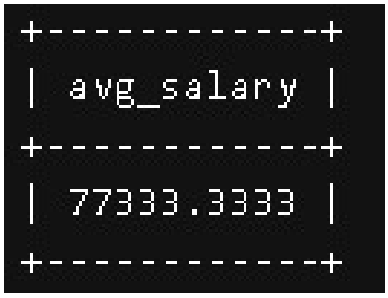
- Find all instructors whose salary is null.

SELECT name FROM instructor WHERE salary IS NULL;



- Find the average salary of instructors in the Computer Science department.

SELECT AVG(salary) AS avg_salary FROM instructor WHERE dept_name='Comp. Sci.';



EXPERIMENT 6

- 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 |
+--------------------+
```

- Find the number of tuples in the

teaches relation SELECT COUNT(*) AS

num_tuples FROM teaches;

```
+-------------+
| num_tuples |
+-------------+
|          15 |
+-------------+
```

- 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

- 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

- 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

- 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

- 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

- 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 7

- 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

- 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

- 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

- 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

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

  GRANT SELECT ON faculty TO new_user;

EXPERIMENT 8

- 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

- 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;

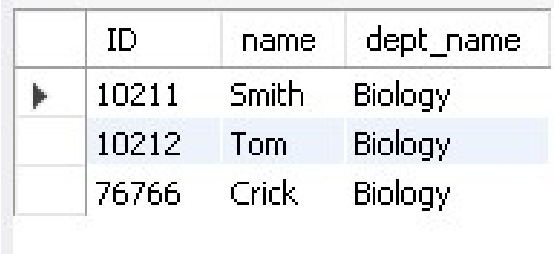
dept_name	total_salary
Comp. Sci.	232000
Finance	170000
Music	40000
Physics	182000
History	66200
Biology	72000
Elec. Eng	80000

- Create a role of student CREATE ROLE student;

- Give select privileges on the view faculty to the role student. GRANT SELECT ON faculty TO student;
- Create a new user and assign her the role of student. CREATE USER fathi@localhost IDENTIFIED BY '1234'; GRANT student TO fathi@localhost;

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

SELECT * FROM faculty WHERE dept_name = 'Biology';



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

- Revoke privileges of the new user
  REVOKE student FROM fathi@localhost;
- Remove the role
  of student.
  DROP ROLE
  student;
- Give select privileges on the view faculty to
  the new user. GRANT SELECT ON faculty TO
  fathi@localhost;

- 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

- Login again as root user
- 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)
```

);

- 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);

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

EXPERIMENT 9

Accessing the database through Python

- Insert following additional tuple in instructor : ('10211', 'Smith', 'Biology', 66000)
- Delete this tuple from instructor : ('10211', 'Smith', 'Biology', 66000)
- Select tuples from instructor where dept_name = 'History'
- Find the Cartesian product instructor x teaches.

- Find the names of all instructors who have taught some course and the course_id
- Find the names of all instructors whose name includes the substring “dar”.
- Find the names of all instructors with salary between 90,000 and 100,000 (that is, ≥ 90,000 and
≤ 100,000)

```
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()
```

```
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,
(76766, 'Crick', 'Biology', 72000, 83821, 'CS-319', 2, 'Sp
(76543, 'Singh', 'Finance', 80000, 83821, 'CS-319', 2, 'Sp
(58583, 'Califieri', 'History', 62000, 83821, 'CS-319', 2,
(45565, 'Katz', 'Comp. Sci.', 75000, 83821, 'CS-319', 2, '
(33456, 'Gold', 'Physics', 87000, 83821, 'CS-319', 2, 'Spr
(32343, 'El Said', 'History', 60000, 83821, 'CS-319', 2, '
(22222, 'Einstein', 'Physics', 95000, 83821, 'CS-319', 2,
(15151, 'Mozart', 'Music', 40000, 83821, 'CS-319', 2, 'Spr
(12121, 'Wu', 'Finance', 90000, 83821, 'CS-319', 2, 'Sprin
(10101, 'Srinivasan', 'Comp. Sci.', 65000, 83821, 'CS-319'
(98345, 'Kim', 'Elec. Eng', 80000, 98345, 'EE-181', 1, 'Sp
(83821, 'Brandt', 'Comp. Sci.', 92000, 98345, 'EE-181', 1,
(76766, 'Crick', 'Biology', 72000, 98345, 'EE-181', 1, 'Sp
(76543, 'Singh', 'Finance', 80000, 98345, 'EE-181', 1, 'Sp
(58583, 'Califieri', 'History', 62000, 98345, 'EE-181', 1,
(45565, 'Katz', 'Comp. Sci.', 75000, 98345, 'EE-181', 1, '
(33456, 'Gold', 'Physics', 87000, 98345, 'EE-181', 1, 'Spr
(32343, 'El Said', 'History', 60000, 98345, 'EE-181', 1, '
(22222, 'Einstein', 'Physics', 95000, 98345, 'EE-181', 1,
(15151, 'Mozart', 'Music', 40000, 98345, 'EE-181', 1, 'Spr
(12121, 'Wu', 'Finance', 90000, 98345, 'EE-181', 1, 'Sprin
(10101, 'Srinivasan', 'Comp. Sci.', 65000, 98345, 'EE-181'


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
```

EXPERIMENT 10

- Order the tuples in the instructors relation as per their salary.
- Find courses that ran in Fall 2017 or in Spring 2018
- Find courses that ran in Fall 2017 and in Spring 2018
- Find courses that ran in Fall 2017 but not in Spring 2018
- Insert following additional tuples in instructor ('10211', 'Smith', 'Biology', 66000) ('10212', 'Tom', 'Biology', NULL
- Find all instructors whose salary is null.
- Find the average salary of instructors in the Computer Science department.
- Find the total number of instructors who teach a course in the Spring 2018 semester.
- Find the number of tuples in the teaches relation
- Find the average salary of instructors in each department
- Find the names and average salaries of all departments whose average salary is greater than 42000
- Name all instructors whose name is neither “Mozart” nor Einstein”.
- Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.
- Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.
- Find the average instructors’ salaries of those departments where the average salary is greater than 42,000.
- Find all departments where the total salary is greater than the average of the total salary at all departments
- List the names of instructors along with the course ID of the courses that they taught.
- 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.

```
    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:")
```

```
('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')


Question18:
('Srinivasan', 'CS-101')
('Srinivasan', 'CS-315')
('Srinivasan', 'CS-347')
('Smith', None)
('Tom', None)
('Wu', 'FIN-201')
('Mozart', 'MU-199')
('Einstein', 'PHY-101')
('El Said', 'HIS-351')
('Gold', None)
('Katz', 'CS-101')
('Katz', 'CS-319')
('Califieri', None)
('Singh', None)
('Crick', 'BIO-101')
('Crick', 'BIO-301')
('Brandt', 'CS-190')
('Brandt', 'CS-190')
('Brandt', 'CS-319')
```

EXPERIMENT 11

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

```
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 fathi@localhost IDENTIFIED BY '1234'
```

```
"""

cursor.execute(new_role)

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

cursor.execute(grant_user)

#  Revoke privileges of the new user
revoke_user = """
REVOKE student FROM fathi@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 fathi@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)
```

```
PS C:\Users\D A GURUPRIYAN\Downloads\ADBMS> & "c
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 12

SQL*Plus: Release 21.0.0.0.0 - Production on Wed May 15 10:51:44 2024

Version 21.3.0.0.0

Enter user-name: system

Enter password:

Last Successful login time: Wed May 15 2024 10:29:18 +05:30

Connected to:

Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

Version 21.3.0.0.0

SQL> create type addr_ty as object

- (street varchar2(60),
- city varchar2(30),
- state char(2),
- zip varchar(9));
- /

Type created.

SQL> CREATE TYPE person_ty AS OBJECT

- (name varchar2(25),
- address addr_ty);

4 /

Type created.

SQL> CREATE TYPE emp_ty AS OBJECT

- (empt_id varchar2(9),
- person person_ty);

4 /

Type created.

SQL> CREATE TABLE EMP_OO

- (full_emp emp_ty);

Table created.

```
SQL> insert into emp_oo values
  2  (emp_ty('100',
    person_ty('Ram',
    addr_ty('1000 TU',
    'Patiala', 'PB', '147001'))));
```

- row created.

```
SQL> insert into emp_oo values
  2    (emp_ty('101',
    person_ty('Sham',
    addr_ty('1001 TU',
  5  'Patiala', 'PB', '147001'))));
```

- row created.

```
SQL> select * from emp_oo;

FULL_EMP(EMPT_ID, PERSON(NAME, ADDRESS(STREET, CITY, STATE, ZIP)))
--------------------------------------------------------------------------------
EMP_TY('100', PERSON_TY('Ram', ADDR_TY('1000 TU', 'Patiala', 'PB', '147001')))
EMP_TY('101', PERSON_TY('Sham', ADDR_TY('1001 TU', 'Patiala', 'PB', '147001')))

SQL> desc emp_oo;
 Name                          Null?    Type
```

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

FULL_EMP EMP_TY

SQL> 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	Ram	Patiala
101	Sham	Patiala

SQL> Update emp_oo e set

- e.full_emp.person.name='Raj'
- where
- e.full_emp.empt_id='100';

- row updated.

SQL> 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

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

- Raj Patiala
- Sham Patiala

SQL> create or replace type newemp_ty as object (firstname varchar2(25),

- lastname varchar2(25), birthdate date,
- member function AGE(birthdate in DATE) return NUMBER)
- /

Type created.

SQL> create or replace type body newemp_ty as

- member function AGE(BirthDate in DATE) return NUMBER is
- begin
- RETURN ROUND(SysDate - birthdate);
- end;
- end;
- /

Type body created.

SQL> create table new_emp_oo

2 (employee newemp_ty);

Table created.

```
SQL> insert into new_emp_oo values
  2  (newemp_ty('Ram', 'Lal', '12-dec-1976'));

1 row created.

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

EMPLOYEE.FIRSTNAME       E.EMPLOYEE.AGE(E.EMPLOYEE.BIRTHDATE)
------------------------  -------------------------------------
Ram                                        17321

SQL> create table new_emp1 of emp_ty;

Table created.

SQL> create type emp_ty1 as object
  2  (empt_id varchar2(9),
  3  person person_ty);
  4  /

Type created.

SQL> create table emp_oo1(full_emp emp_ty1);


Table created.
```

```
SQL> insert into emp_oo1 values
  2       (emp_ty1('101',
  3       person_ty('Sham',
  4       addr_ty('1001 TU',
  5    'Patiala', 'PB', '147001'))));

1 row created.

SQL> insert into new_emp1 values ('100', person_ty('raj', addr_ty('1000 TU', 'Pta', 'Pb',
'147001')));

1 row created.

SQL> select * from new_emp1;

EMPT_ID
---------
PERSON(NAME, ADDRESS(STREET, CITY, STATE, ZIP))
--------------------------------------------------------------------------------
100
PERSON_TY('raj', ADDR_TY('1000 TU', 'Pta', 'Pb', '147001'))

SQL> select ref(p) from new_emp1 p;

REF(P)

--------------------------------------------------------------------------------
```

000028020962310E79DAD541678083F34D04C7597F4FAF0E96224F4E05993B631113268E
D20041B
9

810000

SQL> create type new_dept_oo as object

  2 (depno number(3), dname varchar(20));

  3 /

Type created.

SQL> CREATE TABLE dept_table OF new_dept_oo;

Table created.

SQL> insert into dept_table values(10, 'comp');

1 row created.

SQL> insert into dept_table values(20, 'chem');

1 row created.

SQL> insert into dept_table values(10, 'math');

1 row created.

SQL> select ref(p) from dept_table p;

REF(P)

0000280209E0B2B6CBC62A4509A73B0168855948CE0BD10BC5001F4AD79B080B129E78
F1DF0041 B9

990000

00002802091BDD768FBC6E4197B0D94EE374114CD80BD10BC5001F4AD79B080B129E78
F1DF0041 B9

990001

0000280209F5B9EBEAEDA94A45A9BF32CFD67DAE7D0BD10BC5001F4AD79B080B129E78
F1DF004
1B9

990002

SQL> create table emp_test_fk(

2 empno number(3),

3 name varchar(10),

4 dept ref new_dept_oo);

Table created.

SQL> desc emp_test_fk

Name	Null?	Type
EMPNO		NUMBER(3)

```
NAME                                      VARCHAR2(10)
DEPT                                      REF OF NEW_DEPT_OO


SQL> set desc depth 2
SQL> desc emp_test_fk
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(3)
 NAME                                               VARCHAR2(10)
 DEPT                                               REF OF NEW_DEPT_OO
  DEPNO                                             NUMBER(3)
  DNAME                                             VARCHAR2(20)

SQL> insert into emp_test_fk
  2  select 100, 'raj', ref(p) from dept_table p where depno = 10;

2 rows created.

SQL> insert into emp_test_fk
  2  select 101, 'shyam', ref(p) from dept_table p where depno = 20;

1 row created.

SQL> select * from emp_test_fk;

    EMPNO NAME
---------- ----------
```

```
DEPT
--------------------------------------------------------------------------------
     100 raj
0000220208E0B2B6CBC62A4509A73B0168855948CE0BD10BC5001F4AD79B080B129E78F
1DF



     100 raj
0000220208F5B9EBEAEDA94A45A9BF32CFD67DAE7D0BD10BC5001F4AD79B080B129E78
F1DF

     101 shyam
00002202081BDD768FBC6E4197B0D94EE374114CD80BD10BC5001F4AD79B080B129E78F
1DF



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

    EMPNO NAME
--------- ----------
DEREF(E.DEPT)(DEPNO, DNAME)
--------------------------------------------------------------------------------
     100 raj
NEW_DEPT_OO(10, 'comp')

     100 raj
NEW_DEPT_OO(10, 'math')

     101 shyam
```

```
NEW_DEPT_OO(20, 'chem')


SQL> select empno, name, deref(e.dept), deref(e.dept).depno depno,
  2  deref(e.dept).dname dname from emp_test_fk e;

    EMPNO NAME
---------- ----------
DEREF(E.DEPT)(DEPNO, DNAME)

--------------------------------------------------------------------------------
    DEPNO DNAME
---------- --------------------
      100 raj
NEW_DEPT_OO(10, 'comp')
       10 comp

      100 raj
NEW_DEPT_OO(10, 'math')
       10 math

    EMPNO NAME
---------- ----------
DEREF(E.DEPT)(DEPNO, DNAME)
--------------------------------------------------------------------------------
    DEPNO DNAME
---------- --------------------
```

```
      101 shyam
NEW_DEPT_OO(20, 'chem')
       20 chem


SQL> create table emp_table_fk
  2  (employee emp_ty,
  3  dept ref new_dept_oo);

Table created.


SQL> set describe depth 1
SQL> desc emp_table_fk
 Name                          Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE                                 EMP_TY
 DEPT                                     REF OF NEW_DEPT_OO

SQL> set describe depth 2
SQL> desc emp_table_fk
 Name                          Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE                                 EMP_TY
   EMPT_ID                                VARCHAR2(9)
   PERSON                                 PERSON_TY
 DEPT                                     REF OF NEW_DEPT_OO
```

```
  DEPNO                                     NUMBER(3)
  DNAME                                     VARCHAR2(20)

SQL> set describe depth 3
SQL> desc emp_table_fk
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE                                           EMP_TY
  EMPT_ID                                           VARCHAR2(9)
  PERSON                                            PERSON_TY
    NAME                                            VARCHAR2(25)
    ADDRESS                                         ADDR_TY
 DEPT                                               REF OF NEW_DEPT_OO
  DEPNO                                             NUMBER(3)

  DNAME                                             VARCHAR2(20)

SQL> set describe depth 4
SQL> desc emp_table_fk
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE                                           EMP_TY
  EMPT_ID                                           VARCHAR2(9)
  PERSON                                            PERSON_TY
    NAME                                            VARCHAR2(25)
    ADDRESS                                         ADDR_TY
      STREET                                        VARCHAR2(60)
      CITY                                          VARCHAR2(30)
```

```
    STATE                          CHAR(2)
    ZIP                           VARCHAR2(9)
DEPT                               REF OF NEW_DEPT_OO
 DEPNO                              NUMBER(3)
 DNAME                              VARCHAR2(20)

SQL> INSERT INTO emp_table_fk
 2  VALUES (
 3    emp_ty(
 4       100,
 5       person_ty('ram', addr_ty('10 tu', 'pat', 'pb', '147001'))
 6    ),
•   (SELECT REF(P)
•   FROM dept_table P
•   WHERE depno = 10
•  AND ROWNUM = 1)

11  );
```

- row created.

```
SQL> select * from emp_table_fk;

EMPLOYEE(EMPT_ID, PERSON(NAME, ADDRESS(STREET, CITY, STATE, ZIP)))
--------------------------------------------------------------------------------
DEPT
--------------------------------------------------------------------------------
EMP_TY('100', PERSON_TY('ram', ADDR_TY('10 tu', 'pat', 'pb', '147001')))
```

0000220208E0B2B6CBC62A4509A73B0168855948CE0BD10BC5001F4AD79B080B129E78F
1DF

SQL> select e.employee.empt_id id, e.employee.person.name name,

- deref(e.dept), deref(e.dept).depno depno,
- deref(e.dept).dname dname from emp_table_fk e;

ID NAME

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

DEREF(E.DEPT)(DEPNO, DNAME)

DEPNO DNAME

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

100 ram

NEW_DEPT_OO(10, 'comp')

10 comp print(row)