

PONDICHERRY UNIVERSITY
(A Central university)

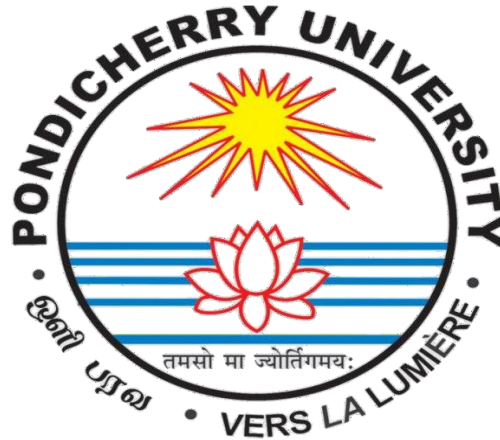


SCHOOL OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
M.Sc. Integrated Computer Science

NAME	:	ASWANTH E K
REG. NO.	:	23370009
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. Integrated Computer Science

PRACTICAL LAB RECORD

BONAFIDE CERTIFICATE

This is to certify that this is a Bonafide record of practical work done by **ASWANTH EK** ,
having Reg. No. **23370009** semester - II from the month February 2024 to June 2024.

FACULTY IN-CHARGE

SUBMITTED FOR THE PRACTICAL EXAM HELD ON:

INTERNAL EXAMINER

EXTERNAL EXAMINER

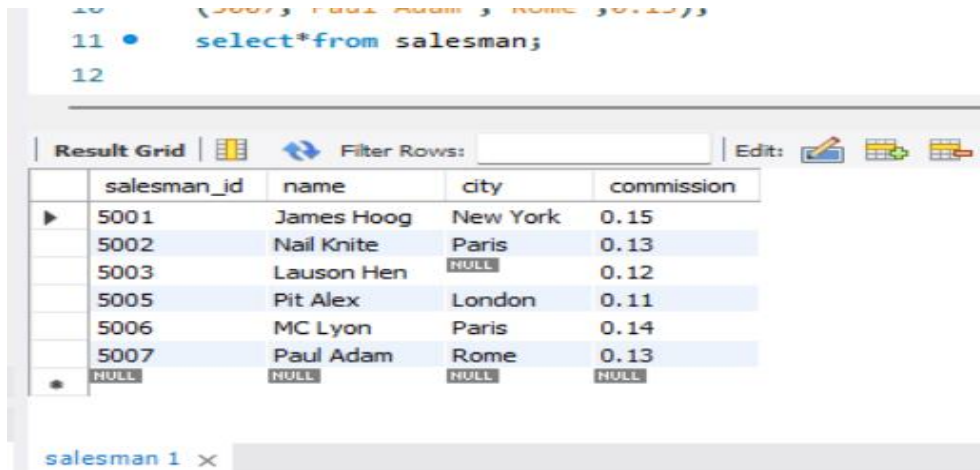
INDEX

EX. No	DATE	TITLE	PAGE	SIGNATURE
1		Experiment 1	4	
2		Experiment 2	13	
3		Experiment 3	15	
4		Experiment 4	17	
5		Experiment 5	22	
6		Experiment 6	25	
7		Experiment 7	28	
8		Experiment 8	31	
9		Experiment 9	35	
10		Experiment 10	39	
11		Experiment 11	49	

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);
```



Result Grid

	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 x

```
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

```

	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

customer 3 x

```

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",3004,5006),
(70003,2480,"2016-10-10",3009,null);

```

```

38 • select*from order1;
39

```

	order_no	purch_amt	order_date	customer_id	salesman_id
▶	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 x

Query 1

- Display name and commission of all the salesmen.

select name,commission from salesman;

Result Grid			Filter Rows:
	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;

Result Grid		Filter Rows:
	salesman_id	
▶	5002	
	NULL	
	5001	
	5006	

Query 3

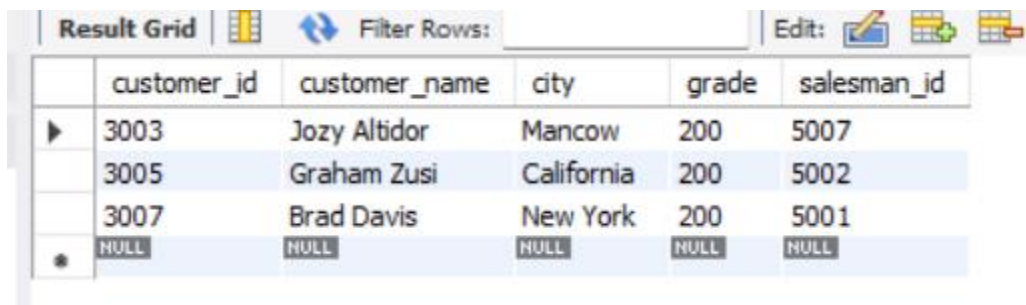
- Display names and city of salesman, who belongs to the city of Paris.

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;

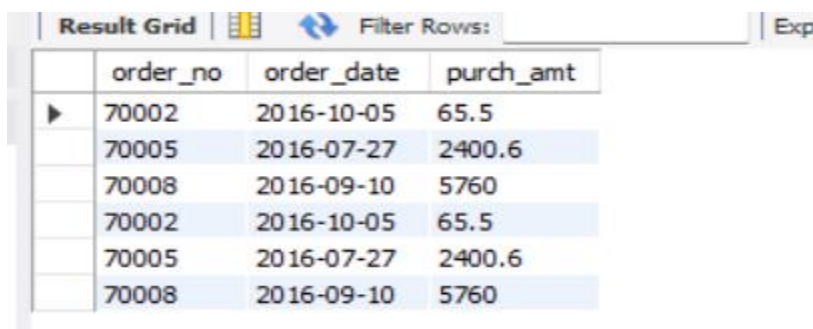


The screenshot shows a 'Result Grid' with a toolbar at the top containing icons for 'Filter Rows', 'Edit', and other functions. The grid displays the results of the query 'select * from customer where grade=200;'. The columns are 'customer_id', 'customer_name', 'city', 'grade', and 'salesman_id'. There are four rows: three with data and one with NULL values.

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

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;



The screenshot shows a 'Result Grid' with a toolbar at the top containing icons for 'Filter Rows', 'Exp', and other functions. The grid displays the results of the query 'select order_no,order_date,purch_amt from order1 where salesman_id=5001;'. The columns are 'order_no', 'order_date', and 'purch_amt'. There are six rows, showing three distinct orders repeated twice.

	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;



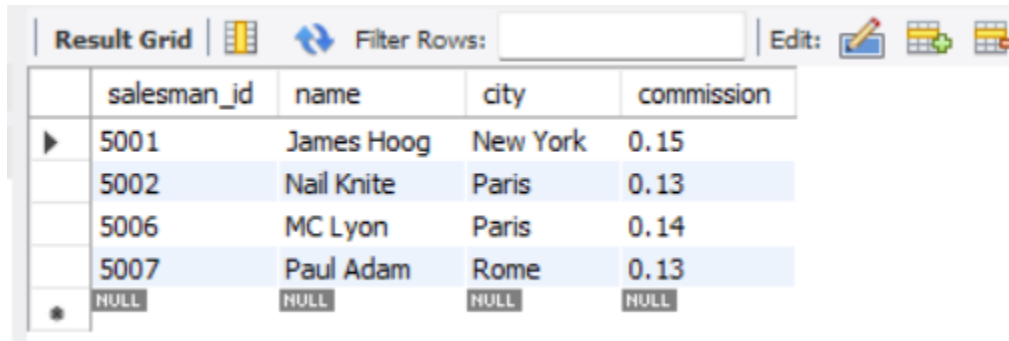
The screenshot shows a 'Result Grid' with a toolbar at the top containing icons for 'Filter Rows', 'Edit', and other functions. The grid displays the results of the query 'select*from customer where city='New York' or not grade>100;'. The columns are 'customer_id', 'customer_name', 'city', 'grade', and 'salesman_id'. There are four rows: three with data and one with NULL values.

	customer_id	customer_name	city	grade	salesman_id
▶	3002	Nick Rimando	New York	100	5001
	3007	Brad Davis	New York	200	5001
	3009	Geoff Camero	Berlin	100	NULL
•	NULL	NULL	NULL	NULL	NULL

Query 7 (table: salesman)

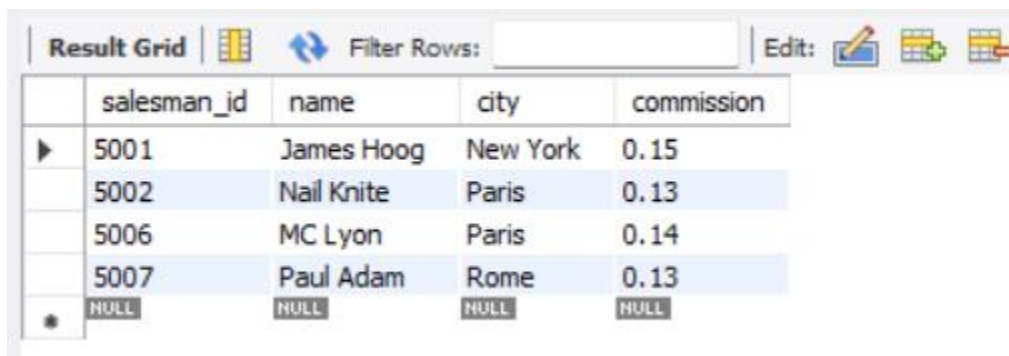
- Find those salesmen with all information who gets the commission within a range of 0.12 and 0.14.

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



	salesman_id	name	city	commission
▶	5001	James Hoog	New York	0.15
	5002	Nail Krite	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);`

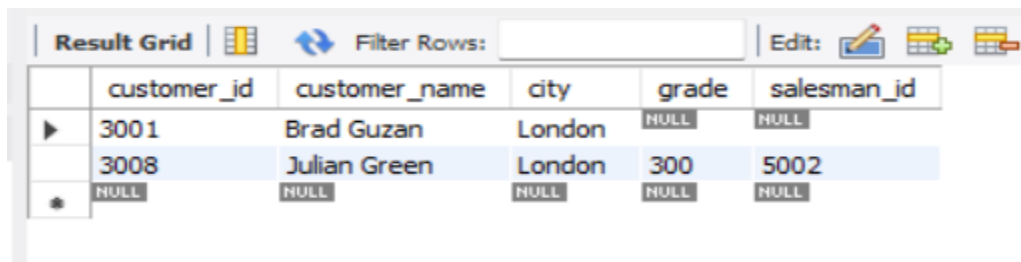


	salesman_id	name	city	commission
▶	5001	James Hoog	New York	0.15
	5002	Nail Krite	Paris	0.13
	5006	MC Lyon	Paris	0.14
	5007	Paul Adam	Rome	0.13
•	NULL	NULL	NULL	NULL

Query 8 (table: customer)

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

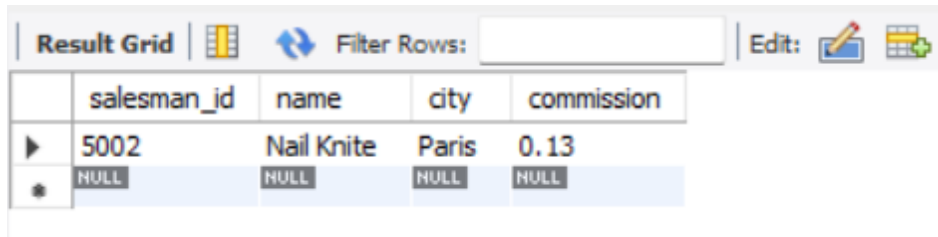
`select*from customer where customer_name like '%n';`



	customer_id	customer_name	city	grade	salesman_id
▶	3001	Brad Guzan	London	NULL	NULL
	3008	Julian Green	London	300	5002
•	NULL	NULL	NULL	NULL	NULL

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%';

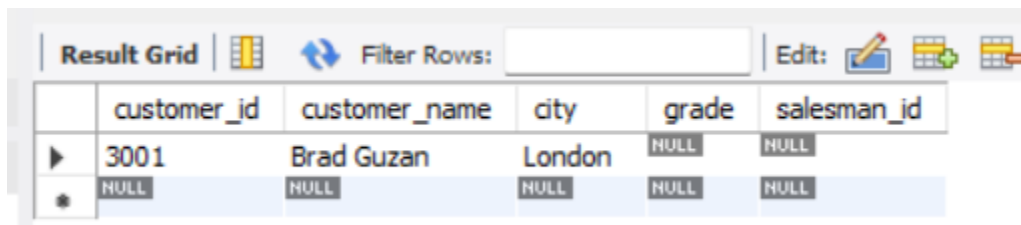


	salesman_id	name	city	commission
▶	5002	Nail Knite	Paris	0.13
✱	NULL	NULL	NULL	NULL

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;

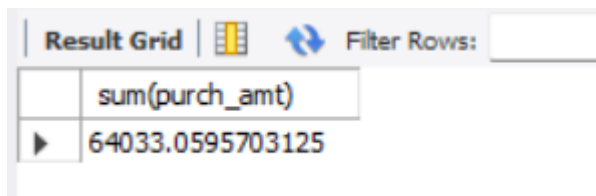


	customer_id	customer_name	city	grade	salesman_id
▶	3001	Brad Guzan	London	NULL	NULL
✱	NULL	NULL	NULL	NULL	NULL

Query 11 (table: orders)

- Find the total purchase amount of all orders.

select sum(purch_amt) from order1;



	sum(purch_amt)
▶	64033.0595703125

Query 12 (table: orders)

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

select count(salesman_id) from customer;

Result Grid		Filter Rows:
	count(salesman_id)	
▶	6	

select count(distinct salesman_id) from order1;

Result Grid		Filter Rows:
	count(distinct salesman_id)	
▶	3	

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;

Result Grid		Filter Rows:
	city	max(grade)
▶	London	300
	New York	200
	Mancow	200
	Paris	300
	California	200
	Berlin	100

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:		Export:
	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;
```

Result Grid			Filter Rows:		Export:
	salesman_id	max(purch_amt)			

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;
```

	customer_id	order_date	MAX(purch_amt)
▶	3007	2016-07-27	2400.6
	3002	2016-09-10	5760
	3004	2016-10-10	19830.4
	3009	2016-10-10	2480

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';
```

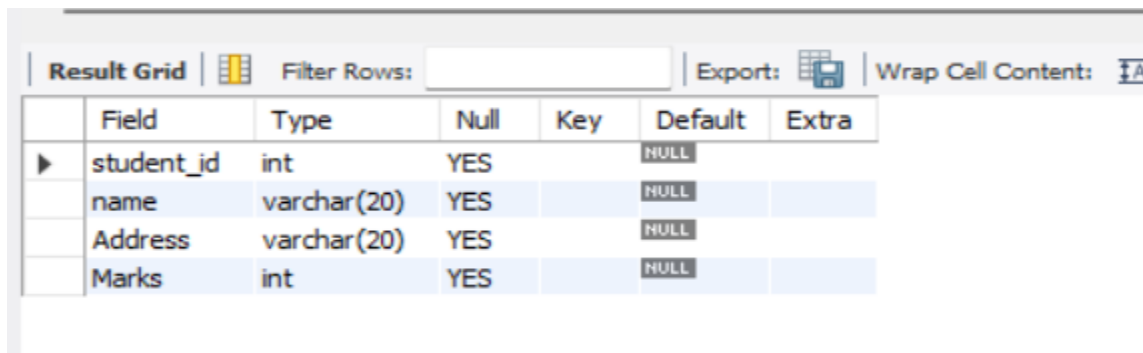
	count(*)
▶	0

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;
```



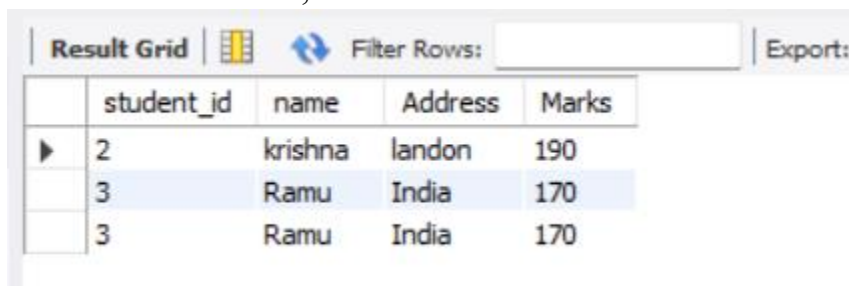
	Field	Type	Null	Key	Default	Extra
▶	student_id	int	YES		NULL	
	name	varchar(20)	YES		NULL	
	Address	varchar(20)	YES		NULL	
	Marks	int	YES		NULL	

-- 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','krishna','london','90');  
insert into student(student_id,name,Address,Marks) values('3','Ramu','India','70');
```

-- Display student table

```
select*from student;
```



	student_id	name	Address	Marks
▶	2	krishna	london	190
	3	Ramu	India	170
	3	Ramu	India	170

-- Display trigger

show triggers;

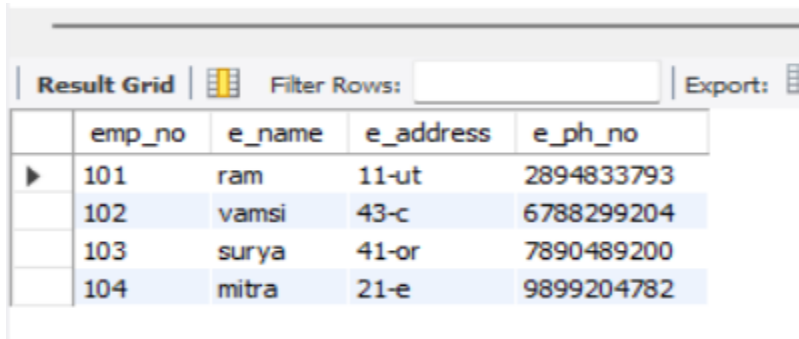
sql_mode	Definer	character_set_client	collation_connection	Database Collation
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

Trigger	Event	Table	Statement	Timing	Created	sql_mode
student_trigger	INSERT	student	set new.Marks=new.Marks+100	BEFORE	2024-06-11 23:16:11.98	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...

EXPERIMENT 3

PROCEDURES:-

```
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,'ram','11-ut',2894833793);
insert into employees values(102,'vamsi','43-c',6788299204);
insert into employees values(103,'surya','41-or',7890489200);
insert into employees values(104,'mitra','21-e',9899204782);
-- Create procedures without parameters
DELIMITER $$
create procedure get_employees ()
begin
select*from employees;
end $$
DELIMITER ;
-- Call procedure
call get_employees();
```



The screenshot shows a database interface with a 'Result Grid' tab. It displays a table with four columns: emp_no, e_name, e_address, and e_ph_no. There are four rows of data, each with a blue arrow icon to its left. The data is as follows:




	emp_no	e_name	e_address	e_ph_no
▶	101	ram	11-ut	2894833793
	102	vamsi	43-c	6788299204
	103	surya	41-or	7890489200
	104	mitra	21-e	9899204782

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

DELIMITER ;




call finds_employees(101);

```
24 • call finds_employees(101);
```

Result Grid	 Filter Rows: <input type="text"/>	Export:  		
	emp_no	e_name	e_address	e_ph_no
▶	101	ram	11-ut	2894833793

call finds_employees(104);

```
25 • call finds_employees(104);
```

Result Grid	 Filter Rows: <input type="text"/>	Export: 		
	emp_no	e_name	e_address	e_ph_no
	104	mitra	21-e	9899204782

call finds_employees(102);

Result Grid	Filter Rows:	Export:	Wr
emp_no	e_name	e_address	e_ph_no
102	vamsi	43-c	6788299204

EXPERIMENT 4

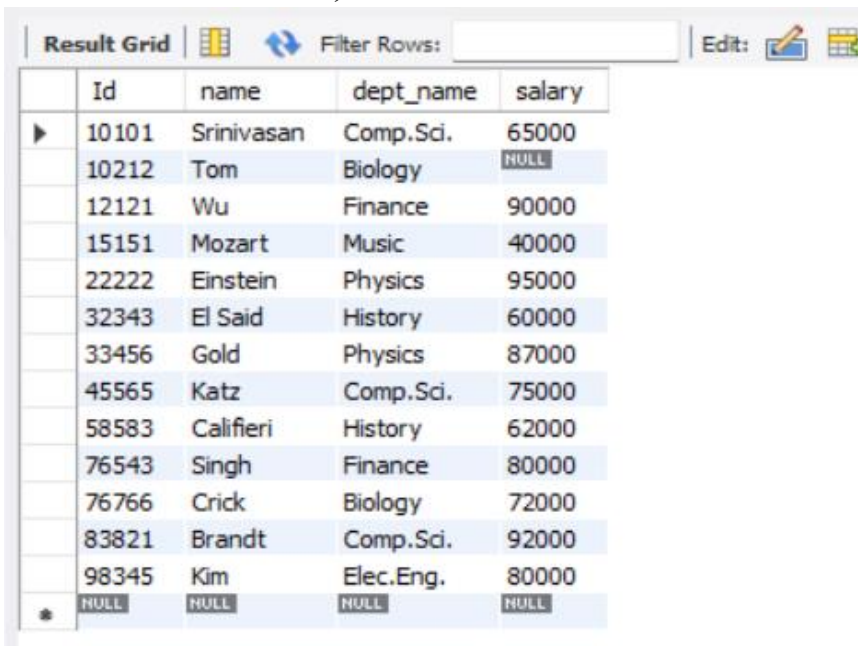
1. Create the following Relation (Tables) with primary key integrity constraint

-- create

```
create table instructor(Id int primary key,name varchar(20),dept_name  
varchar(20),salary integer);
```

```
insert into instructor 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');
```

```
select*from instructor;
```



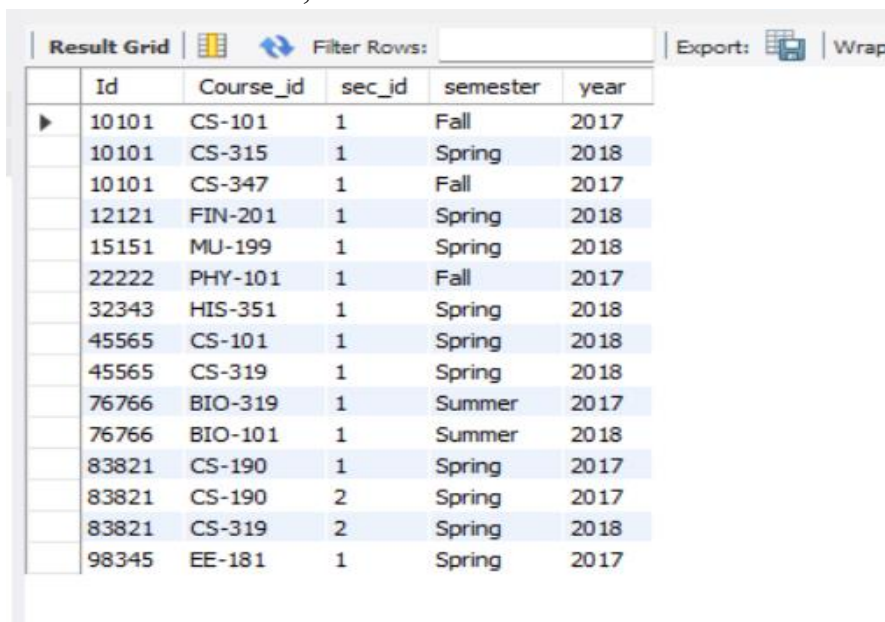
	Id	name	dept_name	salary
▶	10101	Srinivasan	Comp.Sci.	65000
	10212	Tom	Biology	NULL
	12121	Wu	Finance	90000
	15151	Mozart	Music	40000
	22222	Einstein	Physics	95000
	32343	El Said	History	60000
	33456	Gold	Physics	87000
	45565	Katz	Comp.Sci.	75000
	58583	Califieri	History	62000
	76543	Singh	Finance	80000
	76766	Crick	Biology	72000
	83821	Brandt	Comp.Sci.	92000
	98345	Kim	Elec.Eng.	80000
*	NULL	NULL	NULL	NULL

2. Create the following Relation (Tables) teaches

```
create table teaches(Id integer, Course_id varchar(20), sec_id integer, semester  
varchar(20), year integer);
```

```
insert into teaches 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','2018'),  
('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-319','1','Summer','2017'),  
('76766','BIO-101','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;
```



The screenshot shows a database interface with a 'Result Grid' tab selected. The grid displays the data from the 'teaches' table. The columns are 'Id', 'Course_id', 'sec_id', 'semester', and 'year'. The data is as follows:

	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	2018
	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-319	1	Summer	2017
	76766	BIO-101	1	Summer	2018
	83821	CS-190	1	Spring	2017
	83821	CS-190	2	Spring	2017
	83821	CS-319	2	Spring	2018
	98345	EE-181	1	Spring	2017

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

insert into instructor value('10211','Smith','Biology','66000');

102 00:20:21 insert into instructor value('10211','Smith','Biology','66000') 1 row(s) affected 0.047 sec

4. Delete this tuple from instructor ('10211', 'Smith', 'Biology', 66000)

delete from instructor where Id=10211;

103 00:21:48 delete from instructor where Id=10211 1 row(s) affected 0.000 sec

5. Select tuples from instructor where dept_name = 'History'

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	NULL	NULL

6. Find the Cartesian product instructor x teaches.

select*from instructor cross join teaches;

select *from instructor ,teaches;

105 00:22:24 select*from instructor cross join teaches LIMIT 0, 1000 195 row(s) returned 0.015 sec / 0.000 sec

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	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
	10212	Tom	Biology	72000	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
	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

Result 4

Result Grid									
Filter Rows:				Export:		Wrap Cell Content:			
	Id	name	dept_name	salary	Id	Course_id	sec_id	semester	year
	10212	Tom	Biology	NULL	15151	MU-199	1	Spring	2018
	10101	Sriniva...	Comp.Sci.	65000	15151	MU-199	1	Spring	2018
	98345	Kim	Elec.Eng.	80000	22222	PHY-101	1	Fall	2017
	83821	Brandt	Comp.Sci.	92000	22222	PHY-101	1	Fall	2017
	76766	Crick	Biology	72000	22222	PHY-101	1	Fall	2017
	76543	Singh	Finance	80000	22222	PHY-101	1	Fall	2017
	58583	Califieri	History	62000	22222	PHY-101	1	Fall	2017
	45565	Katz	Comp.Sci.	75000	22222	PHY-101	1	Fall	2017
	33456	Gold	Physics	87000	22222	PHY-101	1	Fall	2017
	32343	El Said	History	60000	22222	PHY-101	1	Fall	2017
	22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
	15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
	12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
	10212	Tom	Biology	NULL	22222	PHY-101	1	Fall	2017
	10101	Sriniva...	Comp.Sci.	65000	22222	PHY-101	1	Fall	2017
	98345	Kim	Elec.Eng.	80000	32343	HIS-351	1	Spring	2018
	83821	Brandt	Comp.Sci.	92000	32343	HIS-351	1	Spring	2018
	76766	Crick	Biology	72000	32343	HIS-351	1	Spring	2018
	76543	Singh	Finance	80000	32343	HIS-351	1	Spring	2018
	58583	Califieri	History	62000	32343	HIS-351	1	Spring	2018
	45565	Katz	Comp.Sci.	75000	32343	HIS-351	1	Spring	2018

7. Find the names of all instructors who have taught some course and the course_id

select name, course_id from instructor, teaches where instructor.ID = teaches.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-319
	Crick	BIO-101
	Brandt	CS-190
	Brandt	CS-190
	Brandt	CS-319
	Kim	EE-181

8. Find the names of all instructors whose name includes the substring “dar”.

select * from instructor where name like '%dar%';

108 00:24:31 select from instructor where name like '%da%' LIMIT 0, 1000 0 row(s) returned 0.000 sec / 0.000 sec

Result Grid				
Filter Rows:				
	Id	name	dept_name	salary
*	NULL	NULL	NULL	NULL

9. Find the names of all instructors with salary between 90,000 and 100,000 (that is, $\geq 90,000$ and $\leq 100,000$)

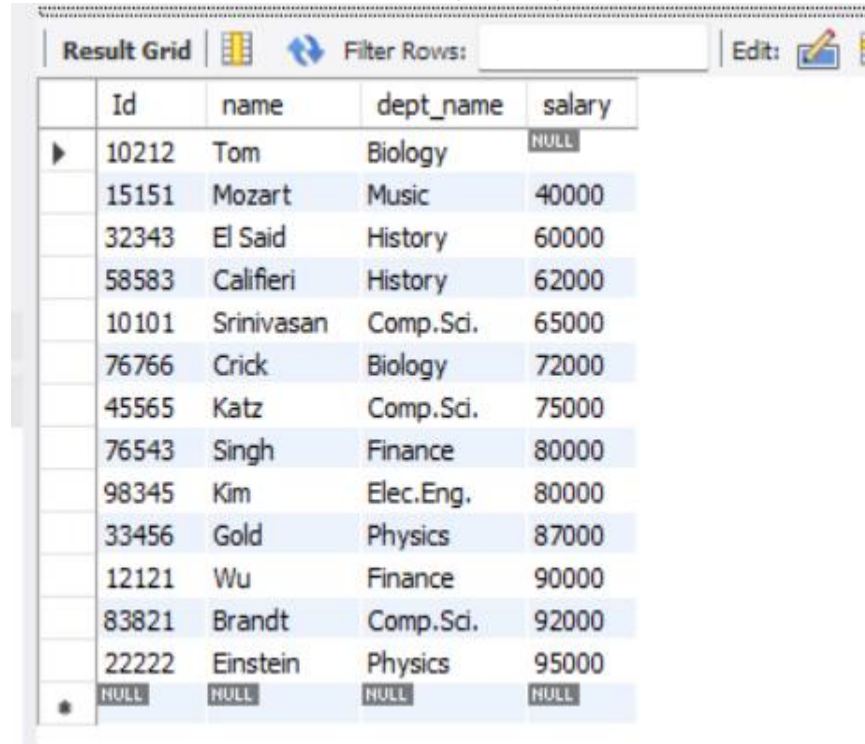
select name from instructor where (salary \geq 90000 and salary \leq 100000);

Result Grid	
Filter	
	name
▶	Wu
	Einstein
	Brandt

EXPERIMENT 5

1. Order the tuples in the instructors relation as per their salary.

`select * from instructor order by salary;`



The screenshot shows a database interface with a 'Result Grid' and a 'Filter Rows' field. The grid displays the 'instructor' table sorted by salary in ascending order. The columns are 'Id', 'name', 'dept_name', and 'salary'. The data is as follows:

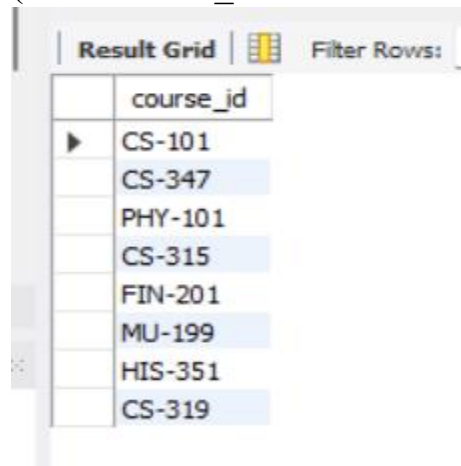
Id	name	dept_name	salary
10212	Tom	Biology	NULL
15151	Mozart	Music	40000
32343	El Said	History	60000
58583	Califieri	History	62000
10101	Srinivasan	Comp.Sci.	65000
76766	Crick	Biology	72000
45565	Katz	Comp.Sci.	75000
76543	Singh	Finance	80000
98345	Kim	Elec.Eng.	80000
33456	Gold	Physics	87000
12121	Wu	Finance	90000
83821	Brandt	Comp.Sci.	92000
22222	Einstein	Physics	95000
NULL	NULL	NULL	NULL

2. Find courses that ran in Fall 2017 or in Spring 2018

`(select course_id from teaches where semester = 'Fall' and year = 2017)`

`union`

`(select course_id from teaches where semester = 'Spring' and year = 2018);`



The screenshot shows a database interface with a 'Result Grid' and a 'Filter Rows' field. The grid displays the result of a SQL query, showing a list of course IDs. The column is 'course_id'. The data is as follows:

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

3. Find courses that ran in Fall 2017 and in Spring 2018

select course_id from teaches where (semester = 'Fall' and year = 2017) and (semester = 'Spring' and year = 2018);

112 01:52:07 select course_id from teaches where (semester = 'Fall' and year = 2017) and (semester = 'Spring' and year = 2018) ... 0 row(s) returned 0.000 sec / 0.000 sec

Result Grid	Filter Rows:
course_id	

4. Find courses that ran in Fall 2017 but not in Spring 2018

select course_id from teaches where (semester = 'Fall' and year = 2017) and not (semester = 'Spring' and year = 2018);

Result Grid	Filter Rows:
course_id	
CS-101	
CS-347	
PHY-101	

5. Insert following additional tuples in instructor :('10211', 'Smith', 'Biology', 66000), ('10212', 'Tom', 'Biology', NULL)
insert into instructor values('10211','Smith','Biology','66000'),
('10212','Tom','Biology',null);

114 01:52:07 insert into instructor values('10211','Smith','Biology','66000'), ('10212','Tom','Biology',null) Error Code: 1062. Duplicate entry '10212' for key 'instructor.PRIMARY' 0.000 sec

6. Find all instructors whose salary is null.

select *from instructor where salary is null;

Result Grid

Filter Rows:

Edit

	Id	name	dept_name	salary
▶	10212	Tom	Biology	NULL
*	NULL	NULL	NULL	NULL

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

`select avg(salary) from instructor where dept_name = 'Comp.Sci.';`

Result Grid	
	avg(salary)
▶	77333.3333

EXPERIMENT 6

1. Find the total number of instructors who teach a course in the Spring 2018 semester.

`select count(distinct ID) from teaches where semester = 'Spring' and year = 2018;`

Result Grid		Filter Rows:
	count(distinct ID)	
▶	6	

2. Find the number of tuples in the teaches relation

`select count(*) from teaches;`

Result Grid		Filter Rows:
	count(*)	
▶	15	

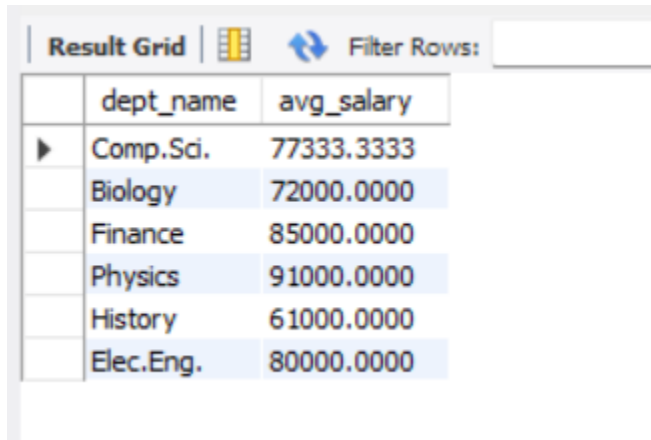
3. Find the average salary of instructors in each department.

`select dept_name, avg(salary) as avg_salary from instructor group by dept_name;`

Result Grid			Filter Rows:
	dept_name	avg_salary	
▶	Comp.Sci.	77333.3333	
	Biology	72000.0000	
	Finance	85000.0000	
	Music	40000.0000	
	Physics	91000.0000	
	History	61000.0000	
	Elec.Eng.	80000.0000	

4. Find the names and average salaries of all departments whose average salary is greater than 42000

```
select dept_name, avg(salary) as avg_salary from instructor group by dept_name  
having avg(salary) > 42000;
```

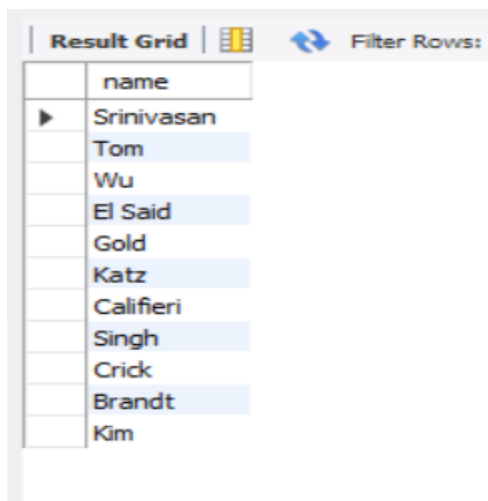


The screenshot shows a 'Result Grid' with a 'Filter Rows' button. The grid contains two columns: 'dept_name' and 'avg_salary'. The data is as follows:

dept_name	avg_salary
Comp.Sci.	77333.3333
Biology	72000.0000
Finance	85000.0000
Physics	91000.0000
History	61000.0000
Elec.Eng.	80000.0000

5. Name all instructors whose name is neither “Mozart” nor Einstein”

```
select distinct name from instructor where name not in ('Mozart', 'Einstein');
```





The screenshot shows a 'Result Grid' with a 'Filter Rows' button. The grid contains one column: 'name'. The data is as follows:

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



6. Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.

```
select name from instructor where salary > some (select salary from instructor where  
dept_name = 'Biology');
```

Result Grid			 Filter Rows:
	name		
▶	Wu		
	Einstein		
	Gold		
	Katz		
	Singh		
	Brandt		
	Kim		

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



`select name from instructor where salary > all (select max(salary) from instructor where dept_name = 'Biology');`

Result Grid			 Filter Rows
	name		
▶	Wu		
	Einstein		
	Gold		
	Katz		
	Singh		
	Brandt		
	Kim		

8. Find the average instructors' salaries of those departments where the average salary is greater than 42,000

`select dept_name, avg_salary from (select dept_name, avg(salary) from instructor group by dept_name) as dept_avg(dept_name, avg_salary) where avg_salary > 42000;`

Result Grid

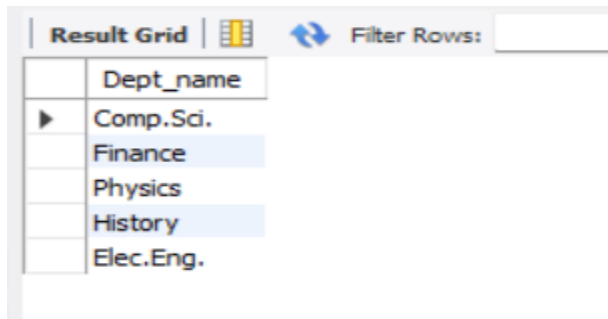


Filter Rows:

	dept_name	avg_salary
▶	Comp.Sci.	77333.3333
	Biology	72000.0000
	Finance	85000.0000
	Physics	91000.0000
	History	61000.0000
	Elec.Eng.	80000.0000

EXPERIMENT 7

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

select Dept_name from instructor group by Dept_name having sum(Salary) > (select avg(Salary) from instructor);

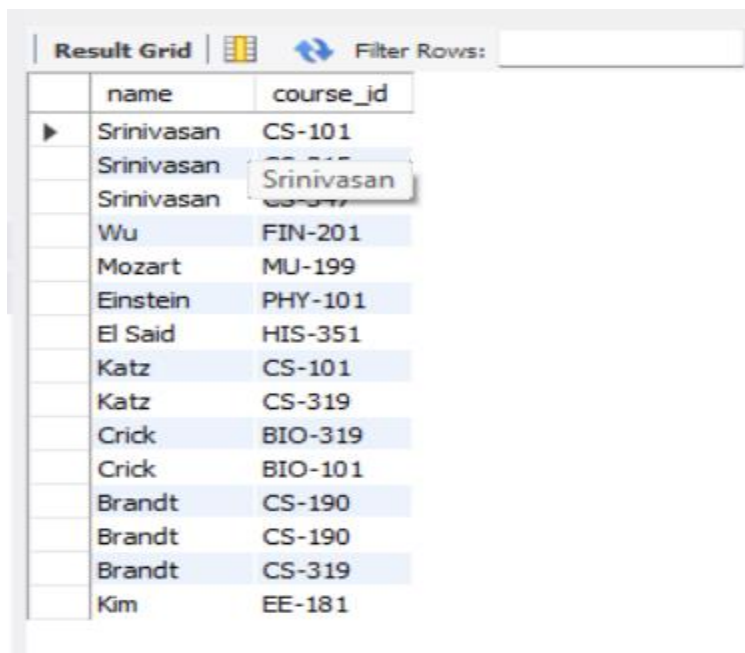


The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The table has one column, 'Dept_name', and five rows of data. The first row is expanded, showing a right-pointing triangle icon to its left.

Dept_name
Comp.Sci.
Finance
Physics
History
Elec.Eng.

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

select instructor.name ,teaches.course_id from instructor join teaches on instructor.Id=teaches.Id;



The screenshot shows a 'Result Grid' window with a 'Filter Rows' input field. The table has two columns, 'name' and 'course_id', and 17 rows of data. The first row is expanded, showing a right-pointing triangle icon to its left. A tooltip is visible over the 'Srinivasan' name in the second row.

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

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

select instructor.name ,teaches.course_id from instructor left join teaches on instructor.Id=teaches.Id;

Result Grid	Filter Rows:
name	course_id
Srinivasan	CS-347
Srinivasan	CS-315
Srinivasan	CS-101
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-101
Crick	BIO-319
Brandt	CS-319
Brandt	CS-190
Brandt	CS-190
Kim	EE-181

4. Create a view of instructors without their salary called faculty
create view faculty as select Id, name, dept_name from instructor;

128 01:52:08 create view faculty as select Id, name, dept_name from instructor

Error Code: 1050. Table 'faculty' already exists

0.000 sec

select*from faculty;

Result Grid			
Filter Rows:			
	Id	name	dept_name
▶	10101	Srinivasan	Comp.Sci.
	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.

5. Give select privileges on the view faculty to the new user.
create user vamsi@localhost identified by 'vamsi3434';

grant select on faculty to vamsi@localhost;

132	01:52:08	grant select on faculty to vamsi@localhost	0 row(s) affected	0.000 sec
-----	----------	--	-------------------	-----------

show grants for vamsi@localhost;

Result Grid	
Filter Rows:	
Export:	
▶	Grants for vamsi@localhost
	GRANT USAGE ON *.* TO `vamsi`@`localhost`

EXPERIMENT 8

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

create view faculty as select Id, name, dept_name from instructor;

128 01:52:08 create view faculty as select Id, name, dept_name from instructor

Error Code: 1050. Table 'faculty' already exists

0.000 sec

select*from faculty;

Result Grid			
Filter Rows:			
	Id	name	dept_name
▶	10101	Srinivasan	Comp.Sci.
	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.

2.Create a view of department salary totals

create view department_salary_totals as select Dept_name, sum(Salary) as TotalSalary from instructor group BY Dept_name;

135 02:31:22 create view department_salary_totals as select Dept_name, sum(Salary) as TotalSalary from instructor group BY D...

Error Code: 1050. Table 'department_salary_totals' already exists

0.000 sec

select*from department_salary_totals;

Result Grid			Filter Rows:
	Dept_name	TotalSalary	
▶	Comp.Sci.	232000	
	Biology	72000	
	Finance	170000	
	Music	40000	
	Physics	182000	
	History	122000	
	Elec.Eng.	80000	

3.Create a role of student
create role 'student';

✓ 137 02:31:22 create role 'student' 0 row(s) affected 0.000 sec

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

✓ 138 02:31:22 grant select on faculty to student 0 row(s) affected 0.016 sec

5.Create a new user and assign her the role of student.

create user krishna@localhost identified by 'krishna3434';

show grants for krishna@localhost;

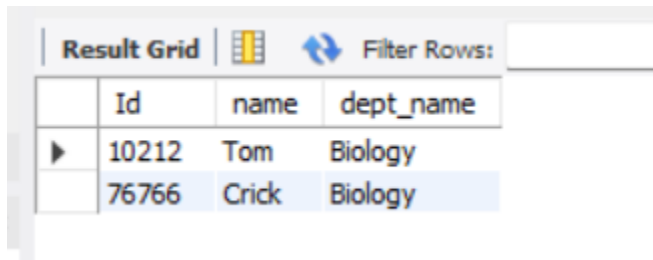
Result Grid			Filter Rows:	Export:
	Grants for krishna@localhost			
▶	GRANT USAGE ON *.* TO 'krishna'@'localhost'			
	GRANT SELECT ON `lab`,`faculty` TO 'krishna...			

6.Login as this new user and find all instructors in the Biology department.

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

✓ 145 02:44:15 GRANT ALL PRIVILEGES ON student.* TO vamsi@localhost 0 row(s) affected 0.015 sec

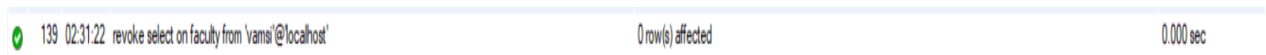
SELECT * FROM faculty WHERE dept_name = 'Biology';



	Id	name	dept_name
▶	10212	Tom	Biology
	76766	Crick	Biology

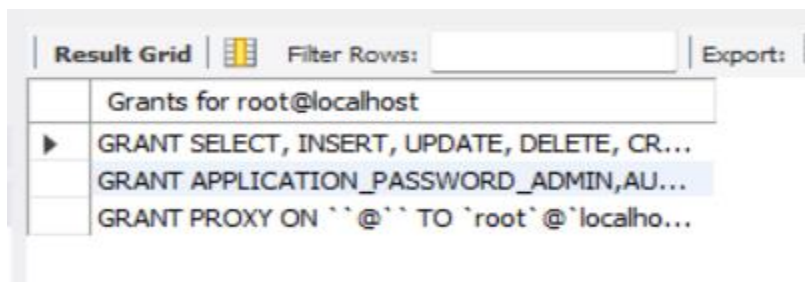
7.Revoke privileges of the new user

revoke select on faculty from 'vamsi'@'localhost';



✓	139	02:31:22	revoke select on faculty from 'vamsi'@'localhost'	0 row(s) affected	0.000 sec
---	-----	----------	---	-------------------	-----------

show grants;



	Grants for root@localhost
▶	GRANT SELECT, INSERT, UPDATE, DELETE, CR...
	GRANT APPLICATION_PASSWORD_ADMIN,AU...
	GRANT PROXY ON ``@`` TO 'root'@'localho...

8.Remove the role of student.

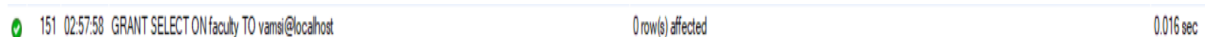
drop role 'student';



✓	141	02:31:22	drop role 'student'	0 row(s) affected	0.000 sec
---	-----	----------	---------------------	-------------------	-----------

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

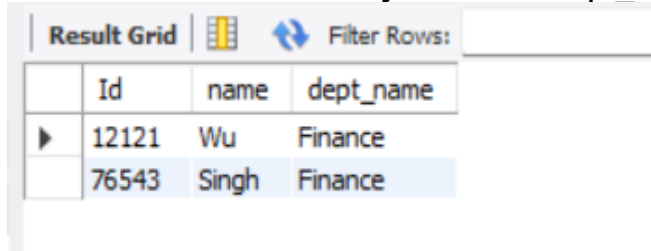
GRANT SELECT ON faculty TO vamsi@localhost;



✓	151	02:57:58	GRANT SELECT ON faculty TO vamsi@localhost	0 row(s) affected	0.016 sec
---	-----	----------	--	-------------------	-----------

10.Login as this new user and find all instructors in the finance department.

SELECT * FROM faculty WHERE dept_name = 'Finance';



The screenshot shows a database interface with a 'Result Grid' tab. It displays the results of a query. At the top, there are icons for a grid, a table, and a filter, followed by a 'Filter Rows:' input field. The table has four columns: 'Id', 'name', and 'dept_name'. There are two data rows. The first row has '12121' in the 'Id' column, 'Wu' in the 'name' column, and 'Finance' in the 'dept_name' column. The second row has '76543' in the 'Id' column, 'Singh' in the 'name' column, and 'Finance' in the 'dept_name' column.

	Id	name	dept_name
▶	12121	Wu	Finance
	76543	Singh	Finance

11.Login again as root user

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

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

148 02:49:30 CREATE TABLE teaches2 (ID INT NOT NULL, course_id VARCHAR(255) NOT NULL, sec_id INT NOT NULL, ... 0 row(s) affected 0.063 sec

13.Create index ID column of teaches. Compare the difference in time to obtain query results with or without index.

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

149 02:50:11 CREATE INDEX idx_ID ON teaches (ID) 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.078 sec

14.Drop the index to free up the space.

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

150 02:50:43 DROP INDEX idx_ID ON teaches 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.016 sec

EXPERIMENT 9

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, $\geq 90,000$ and $\leq 100,000$)

SOURCE CODE:-

```
import mysql.connector

conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='mysql@k2c89snw',
    database='lab'
)

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,))
```

```
results = cursor.fetchall()
```

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

```
# 4
```

```
cartesian_query = """
SELECT * FROM instructor, teaches
"""
```

```
cursor.execute(cartesian_query)
```

```
results = cursor.fetchall()
```

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

```
# 5
```

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

```
# Execute the query
cursor.execute(query)
```

```
# Fetch the results
results = cursor.fetchall()
```

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

```
# 6
```

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

```
cursor.execute(query)
```

```
results = cursor.fetchall()
```

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

```
# 7
```

```

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

cursor.execute(query)

results = cursor.fetchall()

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

conn.commit()

cursor.close()
conn.close()

```

OUTPUT:

OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\nalag> & C:/Users/nalag/AppData/Local/Programs/Python/Python312/python.exe "d:/Msc 2nd year/ADBMS/exp6.py"
(32343, 'El Said', 'History', 60000)
(58583, 'Califieri', 'History', 62000)
(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)
(10212, 'Tom', 'Biology', None, 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)
(10212, 'Tom', 'Biology', None, 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)

```

```

(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)
(10212, 'Tom', 'Biology', None, 98345, 'EE-181', 1, 'Spring', 2017)
(10101, 'Srinivasan', 'Comp.Sci.', 65000, 98345, 'EE-181', 1, 'Spring', 2017)
('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-319')
('Crick', 'BIO-101')
('Brandt', 'CS-190')
('Brandt', 'CS-319')
('Kim', 'EE-181')
Wu

```

```

(10212, 'Tom', 'Biology', None, 98345, 'EE-181', 1, 'Spring', 2017)
(10101, 'Srinivasan', 'Comp.Sci.', 65000, 98345, 'EE-181', 1, 'Spring', 2017)
('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-319')
('Crick', 'BIO-101')
('Brandt', 'CS-190')
('Brandt', 'CS-319')
('Kim', 'EE-181')
Wu
Einstein
Brandt

```

EXPERIMENT 10

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='mysql@k2c89snw',
    database='lab'
)

cursor = conn.cursor()
```

```

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

cursor.execute(order_by_salary_query)

results = cursor.fetchall()

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

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

cursor.execute(courses_in_spring_or_fall)

results = cursor.fetchall()

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

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

cursor.execute(courses_in_spring_and_fall)

results = cursor.fetchall()

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

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

cursor.execute(course_in_fall_only)

```



```

results = cursor.fetchall()

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

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

cursor.execute(insert_tuples)

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

cursor.execute(select_table)

results = cursor.fetchall()

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

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

cursor.execute(instructor_salary_null)

results = cursor.fetchall()

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

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

cursor.execute(avg_cs_dept)

results = cursor.fetchall()

print("Question7:")

```

```

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

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

cursor.execute(instructors_spring)

results = cursor.fetchall()

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

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

cursor.execute(teaches_count)

results = cursor.fetchall()

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

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

cursor.execute(avg_instructor)

results = cursor.fetchall()

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

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

```

```

cursor.execute(avg_salary_greater)

results = cursor.fetchall()

print("Question11:")
for row in results:
    print(row)
print("\n")

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

cursor.execute(instructor_name)

results = cursor.fetchall()

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

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

cursor.execute(salary_greater)

results = cursor.fetchall()

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

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

cursor.execute(salary_greater_biology)

results = cursor.fetchall()

print("Question14:")
for row in results:
    print(row)

```

```

print("\n")

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

cursor.execute(avg_instructor_greater)

results = cursor.fetchall()

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

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

cursor.execute(department_salary)

results = cursor.fetchall()

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

# List the names of instructors along with the course ID of the courses that they taught
instructor_name_with_courseID = """
SELECT instructor.name, teaches.course_id
FROM instructor
JOIN teaches ON instructor.ID = teaches.ID
"""

cursor.execute(instructor_name_with_courseID)

```

```

results = cursor.fetchall()

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

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

cursor.execute(instructor_name_with_courseID_with_null)

results = cursor.fetchall()

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

```

OUTPUT:-

```

OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nalag> & C:/Users/nalag/AppData/Local/Programs/Python/Python312/python.exe "d:/Msc 2nd year/ADBMS/exp7.py"
Question1:
(10212, 'Tom', 'Biology', None)
(15151, 'Mozart', 'Music', 40000)
(32343, 'El Said', 'History', 60000)
(58583, 'Califieri', 'History', 62000)
(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)

Question2:
('CS-101',)
('CS-315',)
('CS-347',)
('FIN-201',)
('MU-199',)
('PHY-101',)
('HIS-351',)
('CS-319',)

Question3:

```

Question3:

Question4:

```
('CS-347',)  
('PHY-101',)
```

Question5:

```
(10101, 'Srinivasan', 'Comp. Sci.', 65000)  
(10211, 'Smith', 'Biology', 66000)  
(10212, 'Tom', 'Biology', None)  
(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)
```

Question6:

```
('Tom',)
```

Question7:

```
(Decimal('77333.3333'),)
```

Question8:

```
(5,)
```

Question9:

```
(15,)
```

Question10:

```
('Comp. Sci.', Decimal('77333.3333'))  
('Biology', Decimal('69000.0000'))  
('Finance', Decimal('85000.0000'))  
('Music', Decimal('40000.0000'))  
('Physics', Decimal('91000.0000'))  
('History', Decimal('61000.0000'))  
('Elec. Eng', Decimal('80000.0000'))
```

Question11:

```
('Comp. Sci.', Decimal('77333.3333'))  
('Biology', Decimal('69000.0000'))  
('Finance', Decimal('85000.0000'))  
('Physics', Decimal('91000.0000'))  
('History', Decimal('61000.0000'))  
('Elec. Eng', Decimal('80000.0000'))
```

Question12:

```
('Srinivasan',)  
('Smith',)  
('Tom',)  
('Wu',)  
('El Said',)  
('Gold',)  
('Katz',)  
('Califieri',)  
('Singh',)  
('Crick',)  
('Brandt',)  
('Kim',)
```

```

Question13:
('Wu',)
('Einstein',)
('Gold',)
('Katz',)
('Singh',)
('Brandt',)
('Kim',)

Question14:
('Wu',)
('Einstein',)
('Gold',)
('Katz',)
('Singh',)
('Brandt',)
('Kim',)

Question15:
('Comp. Sci.', Decimal('77333.3333'))
('Biology', Decimal('69000.0000'))
('Finance', Decimal('85000.0000'))
('Physics', Decimal('91000.0000'))
('History', Decimal('61000.0000'))
('Elec. Eng', Decimal('80000.0000'))

Question16:
('Comp. Sci.',)
('Biology',)
('Finance',)
('Physics',)

Question17:
('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')

```

```
('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')
('Kim', 'EE-181')
```


EXPERIMENT 11

OODBMS:

SQL*Plus: Release 21.0.0.0.0 - Production on Sat May 18 16:40:59 2024
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Enter user-name: system

Enter password:

Last Successful login time: Fri May 17 2024 11:33:55 +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

```
2 (street varchar(30),  
3 city varchar(30),  
4 state char(10),  
5 zip varchar(10));  
6 /
```

Type created.

SQL> create type person_ty as object

```
2 (name varchar(30),  
3 address addr_ty);  
4 /
```

Type created.

SQL> create type emp_ty as object

```
2 (emp_id varchar(10),person  
3 person_ty);  
4 /
```

Type created.

SQL> create table emp_oo

```
2 (full_emp emp_ty);
```

Table created.

SQL> insert into emp_oo values

```
2 (emp_ty('1001',  
3 person_ty('Krishna',  
4 addr_ty('1001 rt','Vijayawada','AP','52119'))));
```

1 row created.

```
SQL> insert into emp_oo values
  2 (emp_ty('1002',
  3 person_ty('Ajay',
  4 addr_ty('182 ri','ppl','AP','52991'))));
```

1 row created.

```
SQL> insert into emp_oo values
  2 (emp_ty('1003',
  3 person_ty('Vamsi',
  4 addr_ty('104 se','Kalapet','Pondy','14729')));
```

1 row created.

```
SQL> select * from emp_oo;
```

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

```
-----
EMP_TY('1001', PERSON_TY('Krishna', ADDR_TY('1001 rt', 'Vijayawada', 'AP
', '52119'))))
```

```
EMP_TY('1002', PERSON_TY('Ajay', ADDR_TY('182 ri', 'ppl', 'AP      ', '52991')
))
```

```
EMP_TY('1003', PERSON_TY('Vamsi', ADDR_TY('104 se', 'Kalapet', 'Pondy   ', '14
729'))))
```

```
SQL> desc emp_oo;
```

```
Name                               Null?  Type
```

```
-----
FULL_EMP                           EMP_TY
```

```
SQL> select * from emp_oo;
```

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

```
-----
EMP_TY('1001', PERSON_TY('Krishna', ADDR_TY('1001 rt', 'Vijayawada', 'AP
', '52119'))))
```

```
EMP_TY('1002', PERSON_TY('Ajay', ADDR_TY('182 ri', 'ppl', 'AP      ', '52991')
))
```

```
EMP_TY('1003', PERSON_TY('Vamsi', ADDR_TY('104 se', 'Kalapet', 'Pondy   ', '14
729'))))
```

```
SQL> select e.full_emp.empt_id ID,
```

```

2 e.full_emp.person.name NAME,
3 e.full_emp.address.city CITY
4 from emp_oo e;
e.full_emp.address.city CITY
*
```

ERROR at line 3:

ORA-00904: "E"."FULL_EMP"."ADDRESS"."CITY": invalid identifier

```

SQL> select e.full_emp.empt_id ID,
2 e.full_emp.person.name NAME,
3 e.full_emp.address.city CITY from emp_oo e;
e.full_emp.address.city CITY from emp_oo e
*
```

ERROR at line 3:

ORA-00904: "E"."FULL_EMP"."ADDRESS"."CITY": invalid identifier

```

SQL> select e.full_emp.empt_id ID,
2 e.full_emp.person.name NAME,
3 e.full_emp.person.address.city CITY
4 from emp_oo e;
```

ID	NAME	CITY
1001	Krishna	Vijayawada
1002	Ajay	ppl
1003	Vamsi	Kalapet

```

SQL> update emp_oo e set
2 e.full_emp.person.name='Ramu'
3 where
4 e.full_emp.empt_id='1001';
```

1 row updated.

```

SQL> select e.full_emp.empt_id ID, e.full_emp.person.name NAME,
2 e.full_emp.person.address.city CITY
3 from emp_oo e;
```

ID	NAME	CITY
1001	Ramu	Vijayawada
1002	Ajay	ppl
1003	Vamsi	Kalapet

```

SQL> create or replace type newemp_ty as object(firstname varchar(20),
2 lastname varchar(20),birthdate date,
3 member function AGE(BirthDate in Date) return NUMBER)
4 /
```

Type created.

```
SQL> create or replace type body newemp_ty as
  2 member function AGE(BirthDate in DATE) return NUMBER is
  3 begin
  4 RETURN ROUND(SysDate - BirthDate);
  5 /
```

Warning: Type Body created with compilation errors.

```
SQL> drop type body;
drop type body
      *
```

ERROR at line 1:
ORA-02302: invalid or missing type name

```
SQL> drop type body;
drop type body
      *
```

ERROR at line 1:
ORA-02302: invalid or missing type name

```
SQL> create or replace type body newemp_ty as
  2 member function AGE(BirthDate in DATE) return NUMBER is
  3 begin
  4 RETURN ROUND(SysDate - BirthDate);
  5 /
```

Warning: Type Body created with compilation errors.

```
SQL> drop type body
  2 ;
```

*

ERROR at line 2:
ORA-02302: invalid or missing type name

```
SQL> create or replace type body newemp_ty as
  2 member function AGE(BirthDate in DATE) return NUMBER is
  3 begin
  4 RETURN ROUND(SysDate - BirthDate);
  5 /
```

Warning: Type Body created with compilation errors.

```
SQL> create or replace type body newemp_ty as
  2 member function AGE(BirthDate in DATE) return NUMBER is
  3
```

```

4 begin
5 RETURN ROUND(SysDate - BirthDate);
6 /

```

Warning: Type Body created with compilation errors.

```

SQL> create or replace type body newemp_ty as
2  member function AGE(BirthDate in DATE) return NUMBER is
3  begin
4  RETURN ROUND(SysDate - BirthDate);
5  end;
6  end;
7  /

```

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','11-oct-1994'));

```

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                  10813

```

```

SQL> create table new_emp1 of emp_ty;

```

Table created.

```

SQL> create type emp_ty1 as object
2  (empt_id varchar(10),
3  person person_ty);
4  /

```

Type created.

```

SQL> create table emp_oo1
2  (full_emp emp_ty1);

```

Table created.

```

SQL> insert into new_emp1 values('1001',

```

```
2 person_ty('raj',addr_ty('143 tr','vizag',
3 'AP','35402')));
```

1 row created.

```
SQL> select * from new_emp1;
```

EMPT_ID

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

1001
PERSON_TY('raj', ADDR_TY('143 tr', 'vizag', 'AP', '35402'))

```
SQL> select ref(p) from new_emp1 p;
```

REF(P)

00002802092CBF85D6CCC64E378DB40C241BC48A1ECBF894A198384E17AEBAC6B87352B73C0
041DC
E10000

```
SQL> drop type emp_ty1;
```

```
drop type emp_ty1
```

*

ERROR at line 1:

ORA-02303: cannot drop or replace a type with type or table dependents

```
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_dert_oo;
```

```
create table dept_table of new_dert_oo
```

*

ERROR at line 1:

ORA-00902: invalid datatype

```
SQL> create table dept_table of new_dept_oo;
```

Table created.

```
SQL> insert into dept_table values
```

```
2 (10,'com sci');
```

1 row created.

```
SQL> insert into dept_table values(12,'math');
insert into dept_table values(12,'math')
*
```

ERROR at line 1:
ORA-00933: SQL command not properly ended

```
SQL> insert into dept_table values(12,'math');
```

1 row created.

```
SQL> insert into dept_table values(13,'chem');
```

1 row created.

```
SQL> select ref(p) from dept_table p;
```

REF(P)

000028020910F8CD3CD081404F94F75749C43250F11120502A851A457B8CB2C74EFB9CDFF60041
DC
F90000

00002802098AB414E0B94F403CB9CEF071DFEB70EE1120502A851A457B8CB2C74EFB9CDFF600
41DC
F90001

00002802092D6F9946325940C1A898749FD0ADF9831120502A851A457B8CB2C74EFB9CDFF60041
DC
F90002

```
SQL> create table emp_test_fk
  2 (empno number(3),
  3 name varchar(20),
  4 dept rwf new_dept_oo);
dept rwf new_dept_oo)
*
```

ERROR at line 4:
ORA-00907: missing right parenthesis

```
SQL> create table emp_test_fk
  2 (empno number(3),
  3 name varchar(20),
  4 dept ref new_dept_oo);
```

Table created.

SQL> desc emp_test_fk

Name	Null?	Type
EMPNO		NUMBER(3)
NAME		VARCHAR2(20)
DEPT		REF OF NEW_DEPT_OO

SQL> set desc depth 2

SQL> desc emp_test_fk

Name	Null?	Type
EMPNO		NUMBER(3)
NAME		VARCHAR2(20)
DEPT		REF OF NEW_DEPT_OO
DEPNO		NUMBER(3)
DNAME		VARCHAR2(20)

SQL> insert into emp_test_fk

```
2 select 1001,'ram',ref(p) from dept_table p
3 where depno=10;
select 1001,'ram',ref(p) from dept_table p
*
```

ERROR at line 2:

ORA-01438: value larger than specified precision allowed for this column

SQL> insert into emp_test_fk

```
2 select 101,'ram',ref(p) from dept_table p
3 where depno=10;
```

1 row created.

SQL> insert into emp_test_fk

```
2 select 100,'surya',ref(p) from dept_table p
3 where depno=12;
```

1 row created.

SQL> insert into emp_test_fk

```
2 select 103,'sai',ref(p) from dept_table p
3 where depno=13'
4 ;
```

ERROR:

ORA-01756: quoted string not properly terminated

SQL> insert into emp_test_fk

```
2 select 103,'sai',ref(p) from dept_table p
3 where depno=13;
```

1 row created.


```
SQL> select * from emp_test_fk;
```

```
EMPNO NAME
-----
DEPT
-----
101 ram
000022020810F8CD3CD081404F94F75749C43250F11120502A851A457B8CB2C74EFB9CDFF6

100 surya
00002202088AB414E0B94F403CB9CEF071DFEB70EE1120502A851A457B8CB2C74EFB9CDFF6

103 sai
00002202082D6F9946325940C1A898749FD0ADF9831120502A851A457B8CB2C74EFB9CDFF6
```

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

```
EMPNO NAME
-----
DEREF(E.DEPT)(DEPNO, DNAME)
-----
101 ram
NEW_DEPT_OO(10, 'com sci')

100 surya
NEW_DEPT_OO(12, 'math')

103 sai
NEW_DEPT_OO(13, 'chem')
```

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

```
ERROR at line 2:
ORA-00942: table or view does not exist
```

```
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
-----
101 ram
```

```
NEW_DEPT_OO(10, 'com sci')
  10 com sci
```

```
  100 surya
NEW_DEPT_OO(12, 'math')
  12 math
```

EMPNO NAME

DEREF(E.DEPT)(DEPNO, DNAME)

DEPNO DNAME

```
  103 sai
NEW_DEPT_OO(13, 'chem')
  13 chem
```

```
SQL> create table emp_table_fk
  2 (employee emp_ty'
  3 (employee emp_ty,/
  4 ;
(employee emp_ty'
  *
```

ERROR at line 2:
ORA-01756: quoted string not properly terminated

```
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(10)
PERSON		PERSON_TY
DEPT		REF OF NEW_DEPT_OO
DEPNO		NUMBER(3)

DNAME VARCHAR2(20)

SQL> set describe depth 3

SQL> desc emp_test_fk

Name	Null?	Type
EMPNO		NUMBER(3)
NAME		VARCHAR2(20)
DEPT		REF OF NEW_DEPT_OO
DEPNO		NUMBER(3)
DNAME		VARCHAR2(20)

SQL> set describe depth 4

SQL> desc emp_test_fk

Name	Null?	Type
EMPNO		NUMBER(3)
NAME		VARCHAR2(20)
DEPT		REF OF NEW_DEPT_OO
DEPNO		NUMBER(3)
DNAME		VARCHAR2(20)

SQL> insert into emp_table_fk values(
2 emp_ty(121,person_ty('ramu',addr_ty('123 re','pat','pb','37892'))),
3 ;
emp_ty(121,person_ty('ramu',addr_ty('123 re','pat','pb','37892'))),
*)

ERROR at line 2:

ORA-00933: SQL command not properly ended

SQL> insert into emp_table_fk values(
2 emp_ty(121,person_ty('ramu',addr_ty('123 re','pat','pb','37892'))),
3 (select ref(p)
4 from dept_table p
5 where depno=10));

1 row created.

SQL> select * from emp_table_fk;

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

DEPT

EMP_TY('121', PERSON_TY('ramu', ADDR_TY('123 re', 'pat', 'pb', '37892'))
)
000022020810F8CD3CD081404F94F75749C43250F11120502A851A457B8CB2C74EFB9CDFF6

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

```
2 deref(e.dept),deref(e.dept).depno depno,  
3 deref(e.dept).dname dname from emp_table_fk e;
```

```
ID      NAME
```

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

```
-----  
DEPNO DNAME
```

```
-----  
121      ramu  
NEW_DEPT_OO(10, 'com sci')  
10 com sci
```

```
SQL>
```