**PONDICHERRY UNIVERSITY**

**(A Central university)**



**SCHOOL OF ENGINEERING AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE**

**M.Sc. Computer Science**

NAME : ASISH SUBAS A

REG. NO. : 23370008

SEMESTER : II - Semester

SUBJECT : CSSC 424 – DATABASE SYSTEM LAB

**PONDICHERRY UNIVERSITY**

**(A Central university)**



**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE**

**M.Sc. Computer Science** PRACTICAL LAB RECORD

**BONAFIDE CERTIFICATE**

This is to certify that this is a Bonafide record of practical work done by **ASISH SUBAS A**, having Reg. No. **23370008** 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),

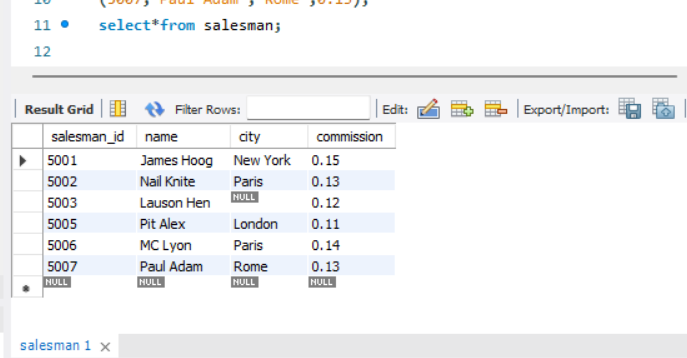
(5002,"Nail Knite","Paris",0.13),

(5005,"Pit Alex","London",0.11),

(5006,"MC Lyon","Paris",0.14),

(5003,"Lauson Hen",null,0.12),

(5007,"Paul Adam","Rome",0.13);



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),

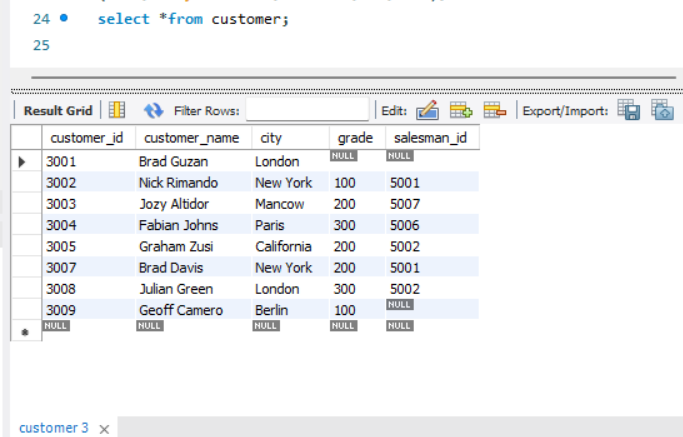
(3004,"Fabian Johns","Paris",300,5006),

(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);



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),

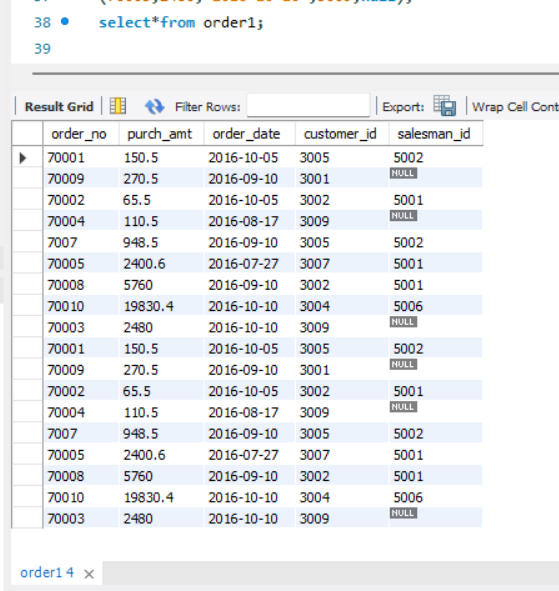
(7007,948.5,"2016-09-10",3005,5002),

(70005,2400.6,"2016-07-27",3007,5001),

(70008,5760,"2016-09-10",3002,5001),

(70010,19830.43,"2016-10-10",3004,5006),

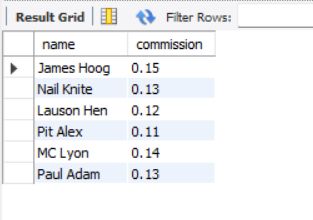
(70003,2480,"2016-10-10",3009,null);



Query 1

• Display name and commission of all the salesmen.

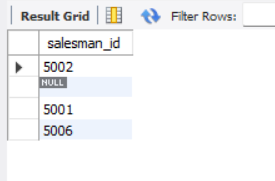
select name,commission from salesman;



Query 2

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

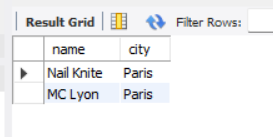
select distinct salesman\_id from order1;



Query 3

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

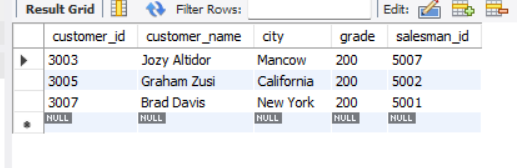
select name,city from salesman where city="paris";



Query 4

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

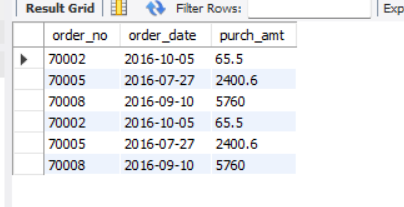
select \* from customer where grade=200;



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.

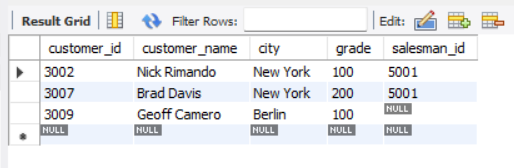
select order\_no,order\_date,purch\_amt from order1 where salesman\_id=5001;



Query 6 (table: customer)

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

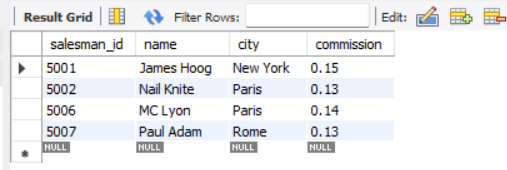
select\*from customer where city='New York' or not grade>100;



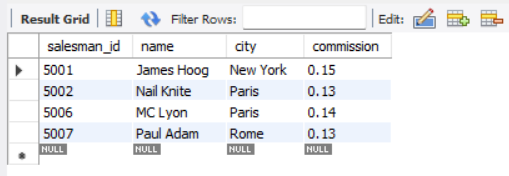
Query 7 (table: salesman)

• 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);



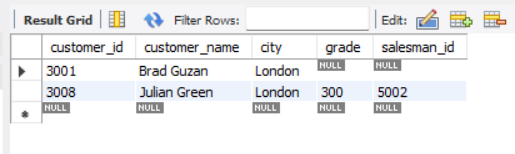
select\*from salesman where(commission between 0.12 and 0.14);



Query 8 (table: customer)

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

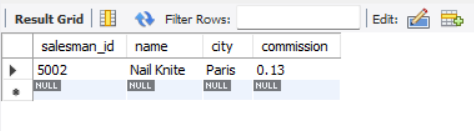
select\*from customer where customer\_name like '%n';



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.

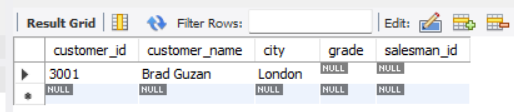
select\*from salesman where name like 'n\_\_l%';



Query 10 (table: customer)

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

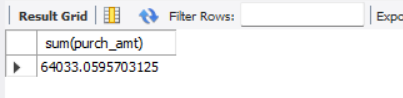
select\*from customer where grade is Null;



Query 11 (table: orders)

• Find the total purchase amount of all orders.

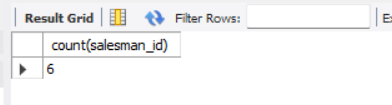
select sum(purch\_amt) from order1;



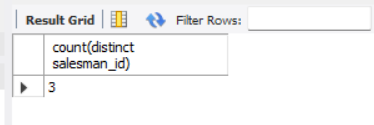
Query 12 (table: orders)

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

select count(salesman\_id) from customer;



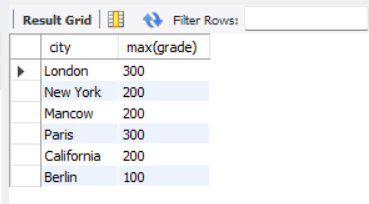
select count(distinct salesman\_id) from order1;



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;

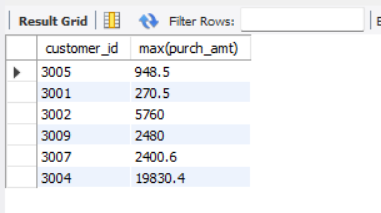


Query 14 (table: orders)

• 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;

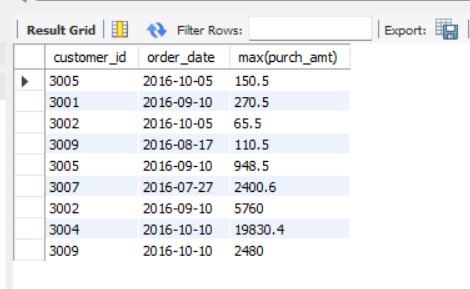


Query 15 (table: orders)

• 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;

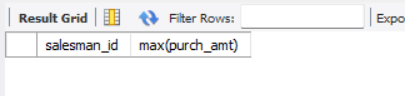


Query 16 (table: orders)

• 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;

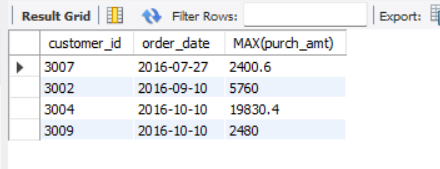


Query 17 (table: orders)

• 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

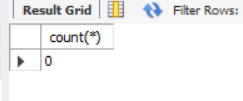
group by customer\_id, order\_date having max(purch\_amt) > 2000.00;



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;

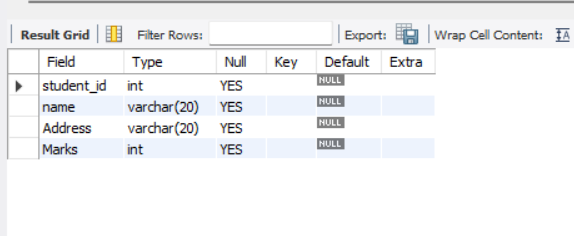
use trigger1;

-- Create student table

create table student(student\_id integer null,name varchar(20),Address varchar(20),Marks integer(10));

-- Describe student table

desc student;



-- create trigger

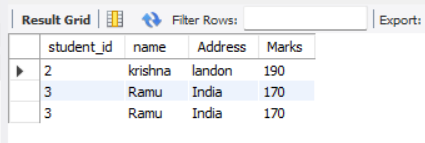
create trigger student\_trigger before insert on student for each row set new.Marks=new.Marks+100;

insert into student(student\_id,name,Address,Marks) values('2','krishna','landon','90');

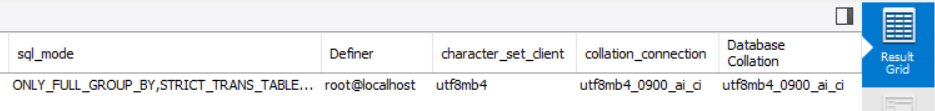
insert into student(student\_id,name,Address,Marks) values('3','Ramu','India','70');

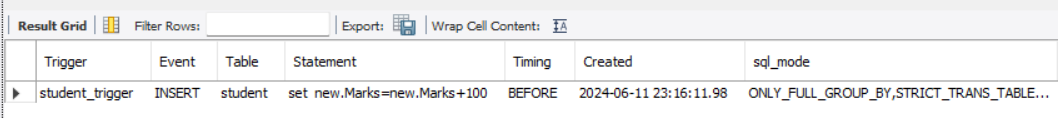
-- Display student table

select\*from student;



-- Display trigger

show triggers; 



**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

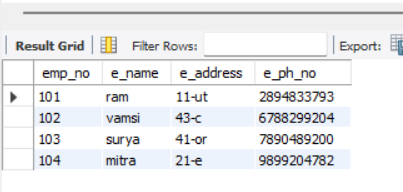
select\*from employees;

end $$

DELIMITER ;

-- Call procedure

call get\_employees();



-- create procedures with parameters

DELIMITER $$

create procedure finds\_employees (in id int)

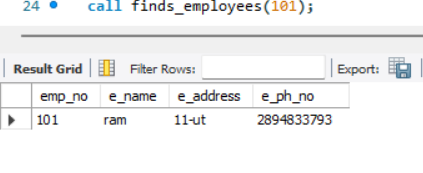
begin

select\*from employees ;

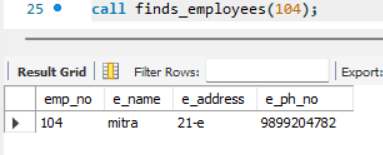
end $$

DELIMITER ;

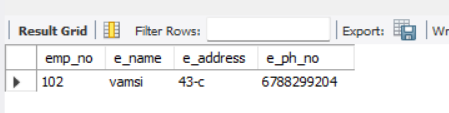
call finds\_employees(101);



call finds\_employees(104);



call finds\_employees(102);



**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'),

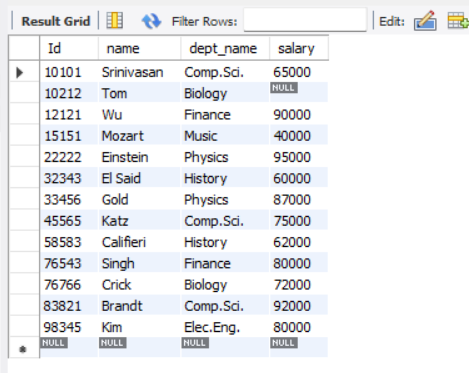
('76543','Singh','Finance','80000'),

('76766','Crick','Biology','72000'),

('83821','Brandt','Comp.Sci.','92000'),

('98345','Kim','Elec.Eng.','80000');

select\*from instructor;



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'),

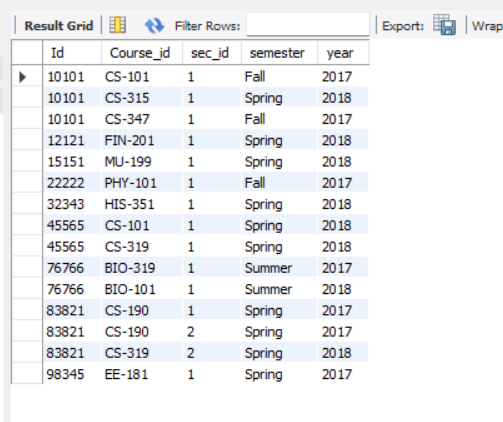
('83821','CS-190','1','Spring','2017'),

('83821','CS-190','2','Spring','2017'),

('83821','CS-319','2','Spring','2018'),

('98345','EE-181','1','Spring','2017');

select\*from teaches;



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

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



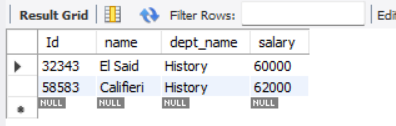
4. Delete this tuple from instructor ('10211', 'Smith', 'Biology', 66000)

delete from instructor where Id=10211;



5. Select tuples from instructor where dept\_name = ‘History’

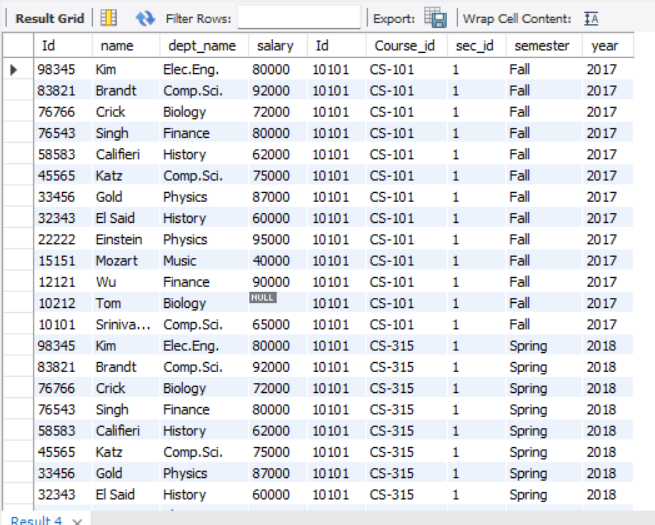
select\*from instructor where dept\_name='History';

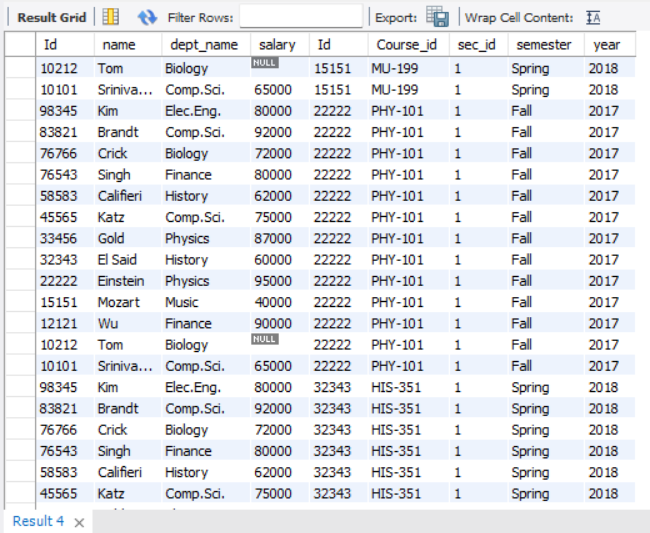


6. Find the Cartesian product instructor x teaches.

select\*from instructor cross join teaches;

select \*from instructor ,teaches;





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

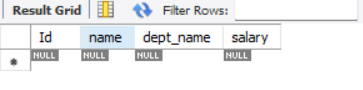
select name, course\_id from instructor, teaches where instructor.ID = teaches.ID;



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

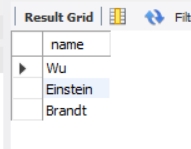
select\*from instructor where name like'%dar%';





9. Find the names of all instructors with salary between 90,000 and 100,000 (that is, ≥ 90,000 and ≤ 100,000)

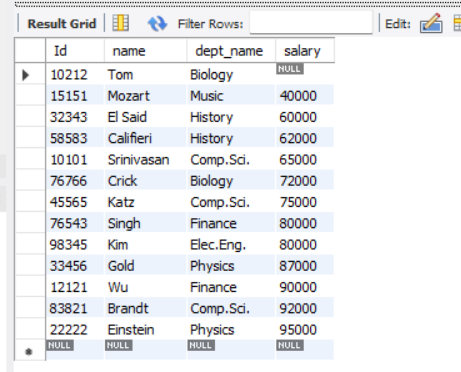
select name from instructor where (salary>=90000 and salary<=100000);



**EXPERIMENT 5**

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

select\*from instructor order by salary;

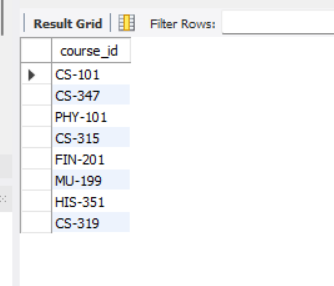


2. Find courses that ran in Fall 2017 or in Spring 2018

(select course\_id from teaches where semester = 'Fall' and year = 2017)

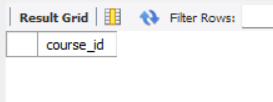
union

(select course\_id from teaches where semester = 'Spring' and year = 2018);



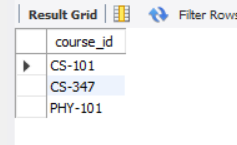
3. Find courses that ran in Fall 2017 and in Spring 2018

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



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);



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

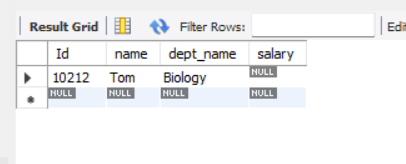
insert into instructor values('10211','Smith','Biology','66000'),

('10212','Tom','Biology',null);



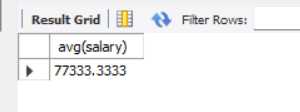
6. Find all instructors whose salary is null.

select \*from instructor where salary is null;



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

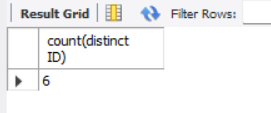
select avg(salary) from instructor where dept\_name = 'Comp.Sci.';



**EXPERIMENT 6**

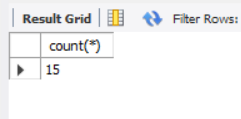
1. Find the total number of instructors who teach a course in the Spring 2018 semester.

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



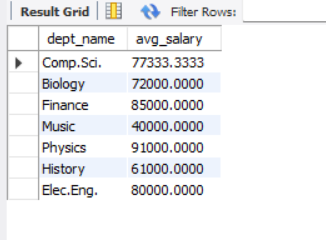
2. Find the number of tuples in the teaches relation

select count(\*) from teaches;



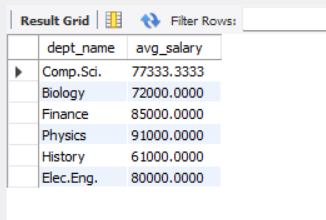
3. Find the average salary of instructors in each department.

select dept\_name, avg(salary) as avg\_salary from instructor group by dept\_name;



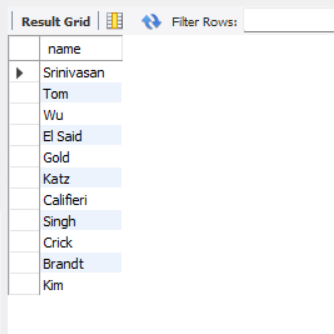
4. Find the names and average salaries of all departments whose average salary is greater than 42000

select dept\_name, avg(salary) as avg\_salary from instructor group by dept\_name having avg(salary) > 42000;



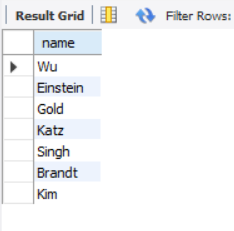
5. Name all instructors whose name is neither “Mozart” nor Einstein”

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



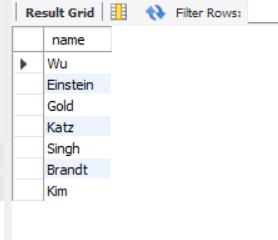
6. Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.

select name from instructor where salary > some (select salary from instructor where dept\_name = 'Biology');



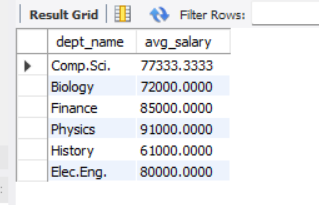
7. Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.

select name from instructor where salary > all (select max(salary) from instructor where dept\_name = 'Biology');



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

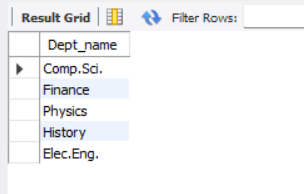
select dept\_name, avg\_salary from (select dept\_name, avg(salary) from instructor group by dept\_name) as dept\_avg(dept\_name, avg\_salary) where avg\_salary > 42000;



**EXPERIMENT 7**

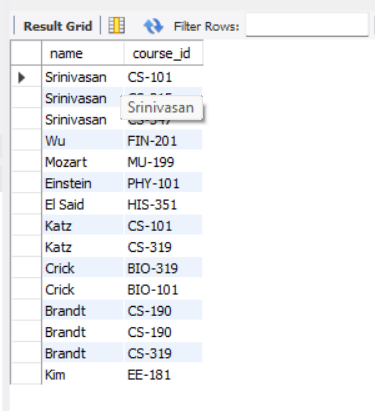
1.Find all departments where the total salary is greater than the average of the total salary at all departments

select Dept\_name from instructor group by Dept\_name having sum(Salary) > (select avg(Salary) from instructor);



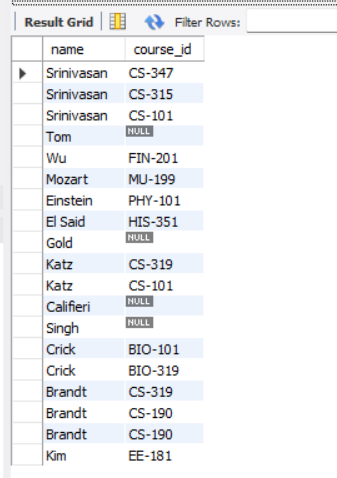
2.List the names of instructors along with the course ID of the courses that they taught

select instructor.name ,teaches.course\_id from instructor join teaches on instructor.Id=teaches.Id;



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;

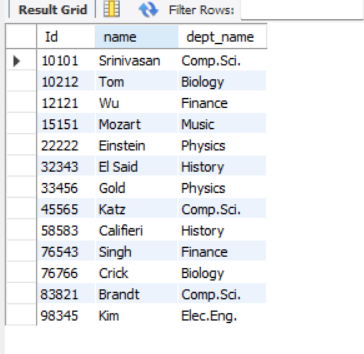


4.Create a view of instructors without their salary called faculty

create view faculty as select Id, name, dept\_name from instructor;



select\*from faculty;



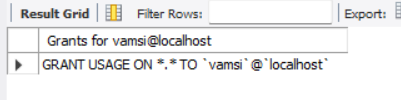
5.Give select privileges on the view faculty to the new user.

create user vamsi@localhost identified by 'vamsi3434';

grant select on faculty to vamsi@localhost;



show grants for vamsi@localhost;



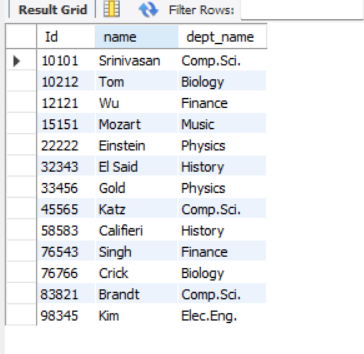
**EXPERIMENT 8**

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

create view faculty as select Id, name, dept\_name from instructor;



select\*from faculty;

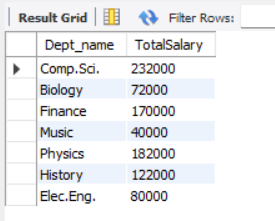


2.Create a view of department salary totals

create view department\_salary\_totals as select Dept\_name, sum(Salary) as TotalSalary from instructor group BY Dept\_name;

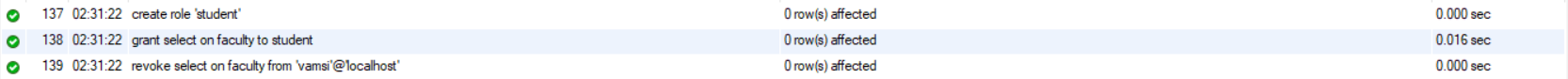


select\*from department\_salary\_totals;



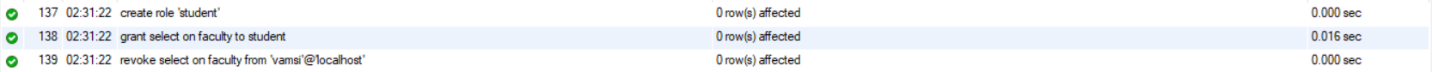
3.Create a role of student

create role 'student';



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

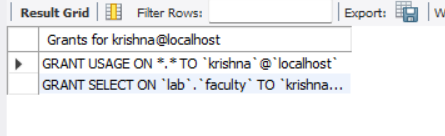
grant select on faculty to student;



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

create user krishna@localhost identified by 'krishna3434';

show grants for krishna@localhost;

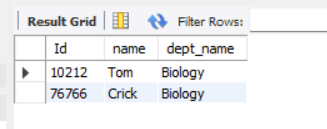


6.Login as this new user and find all instructors in the Biology department.

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

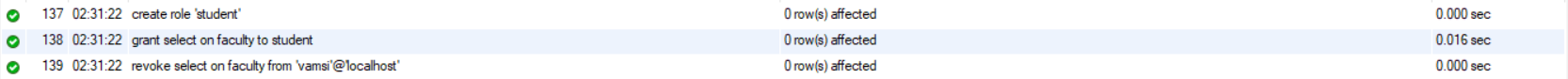


SELECT \* FROM faculty WHERE dept\_name = 'Biology';

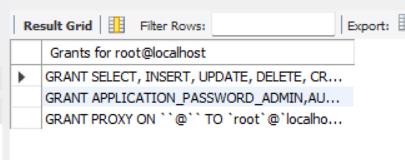


7.Revoke privileges of the new user

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



show grants;



8.Remove the role of student.

drop role 'student';



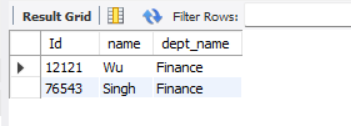
9.Give select privileges on the view faculty to the new user.

GRANT SELECT ON faculty TO vamsi@localhost;



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

SELECT \* FROM faculty WHERE dept\_name = 'Finance';



11.Login again as root user

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

CREATE TABLE teaches2 (

ID INT NOT NULL,

course\_id VARCHAR(255) NOT NULL,

sec\_id INT NOT NULL,

semester VARCHAR(255) NOT NULL CHECK (semester IN ('Fall', 'Winter', 'Spring', 'Summer')),

year INT NOT NULL,

FOREIGN KEY (ID) REFERENCES instructor(ID)

);



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

CREATE INDEX idx\_ID ON teaches (ID);



14.Drop the index to free up the space.

DROP INDEX idx\_ID ON teaches;



**EXPERIMENT 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, ≥ 90,000 and ≤ 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:

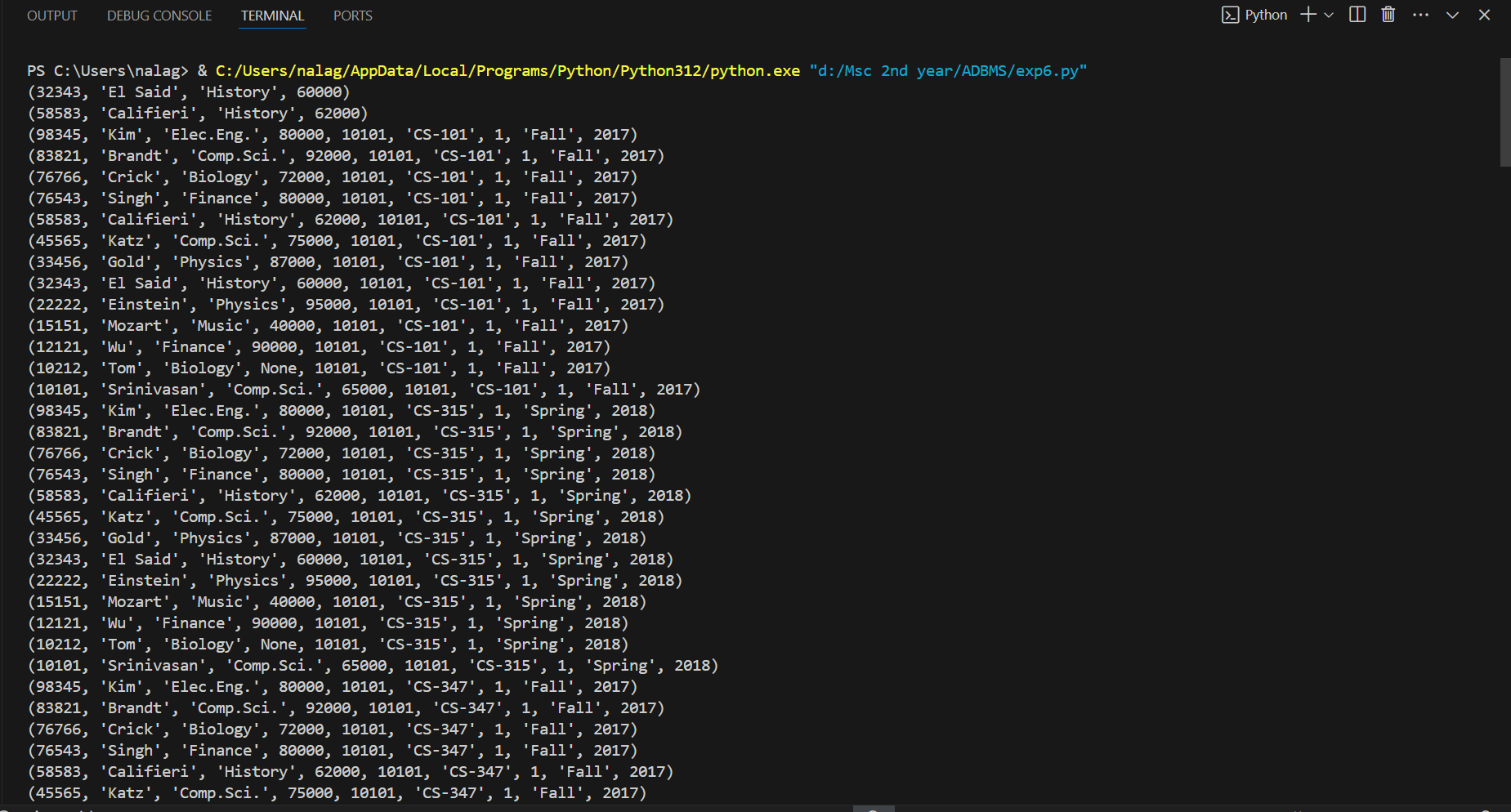
    print(row[0])

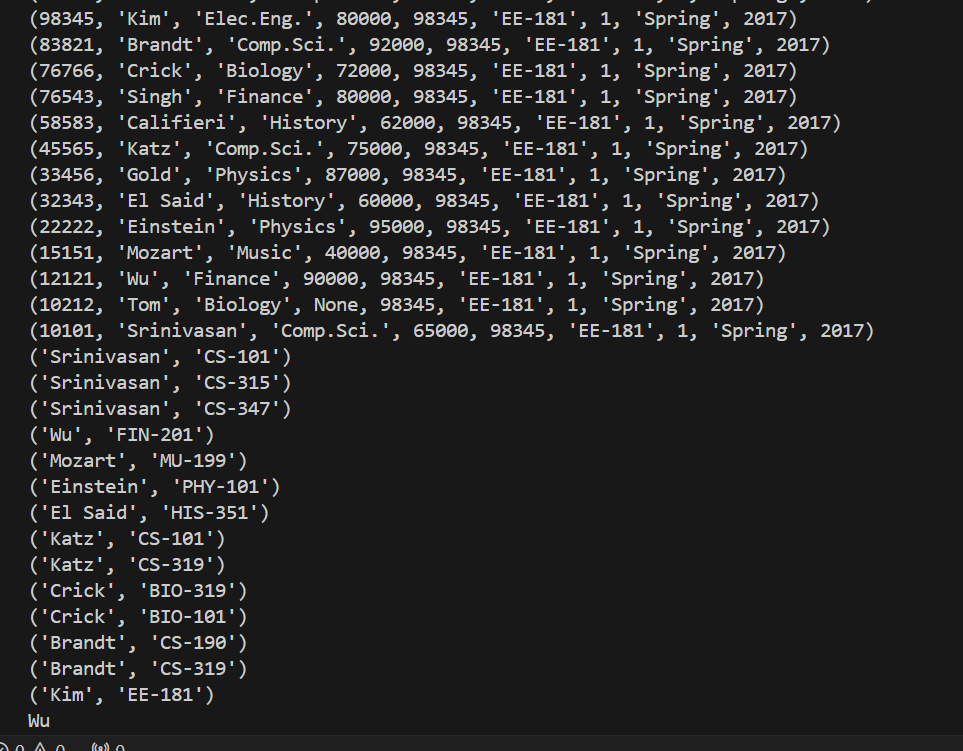
conn.commit()

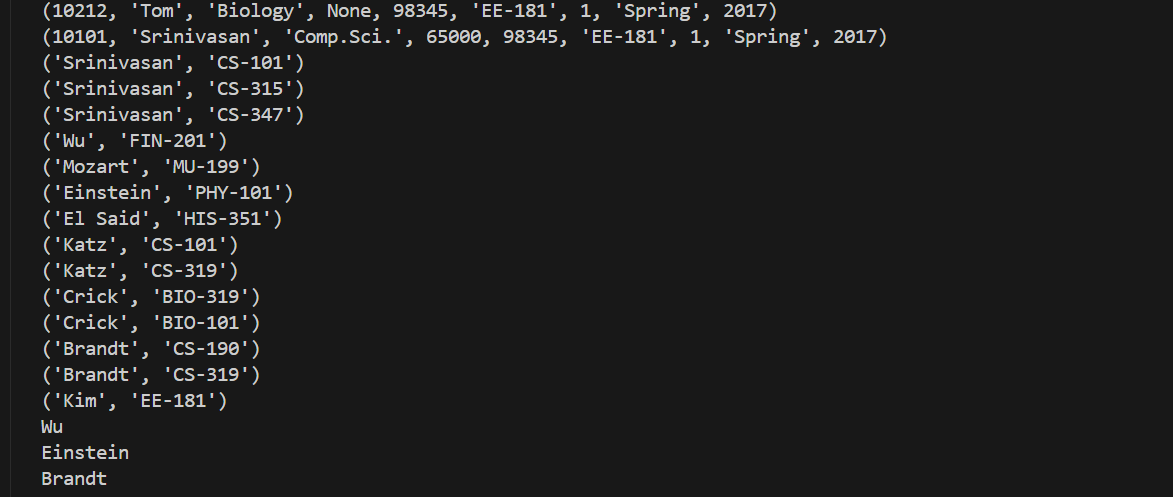
cursor.close()

conn.close()

**OUTPUT:**







**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()

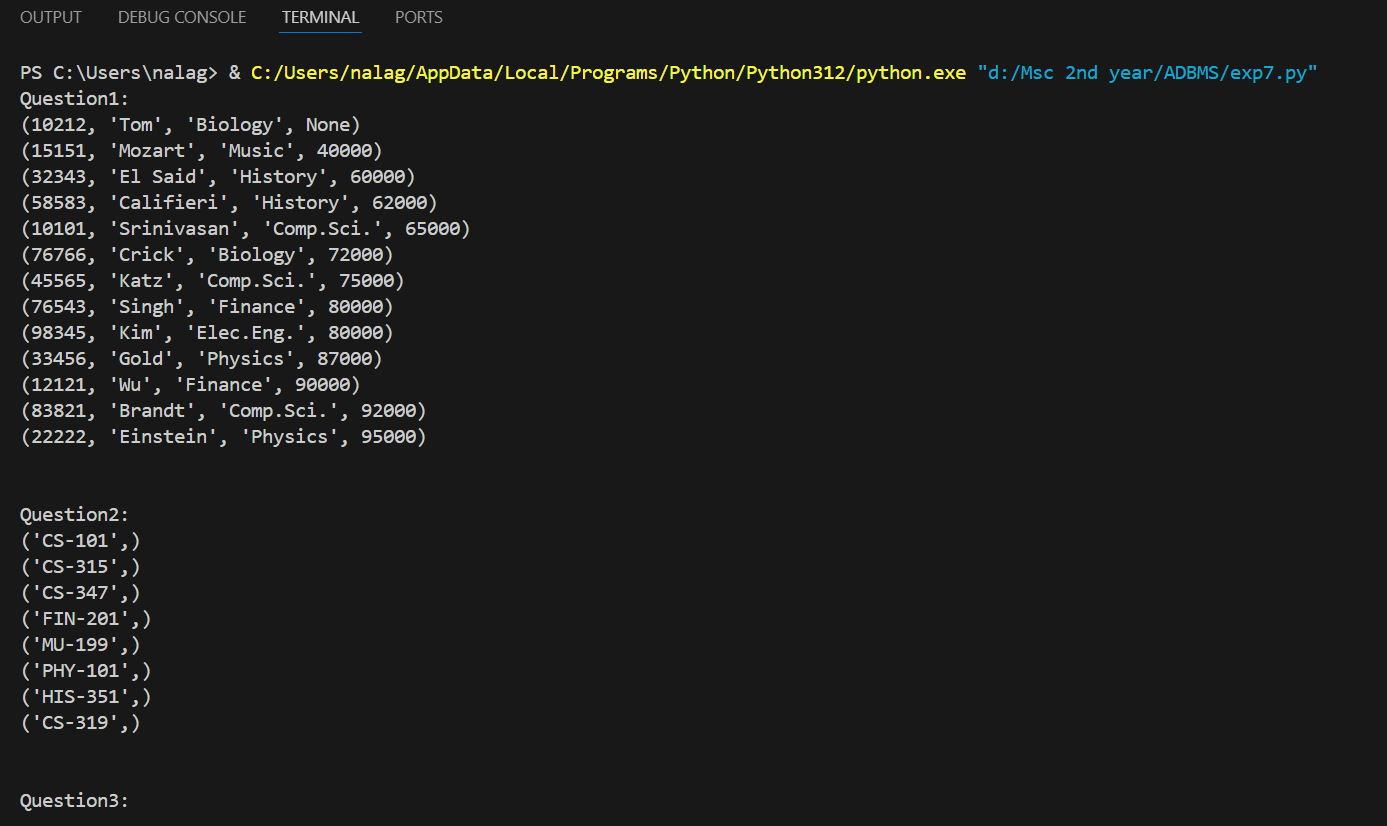
print("Question18:")

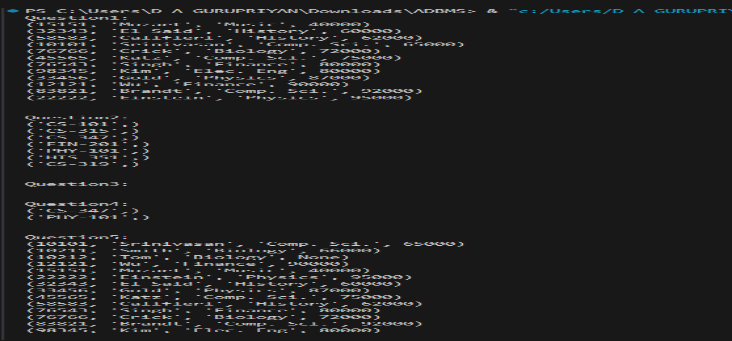
for row in results:

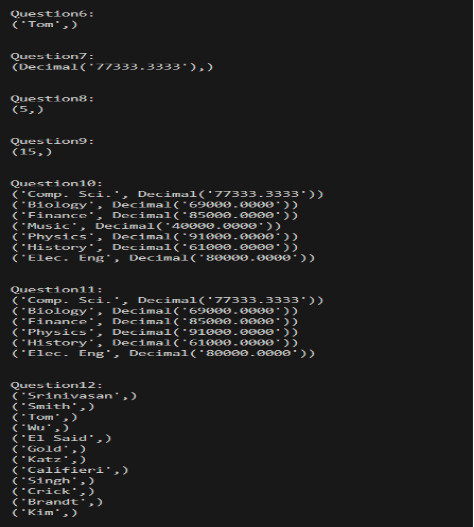
    print(row)

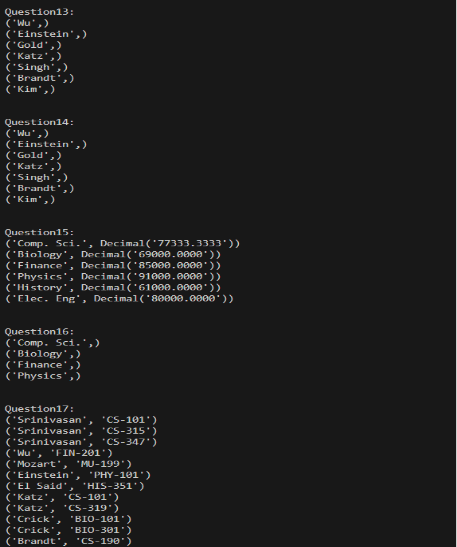
print("\n")

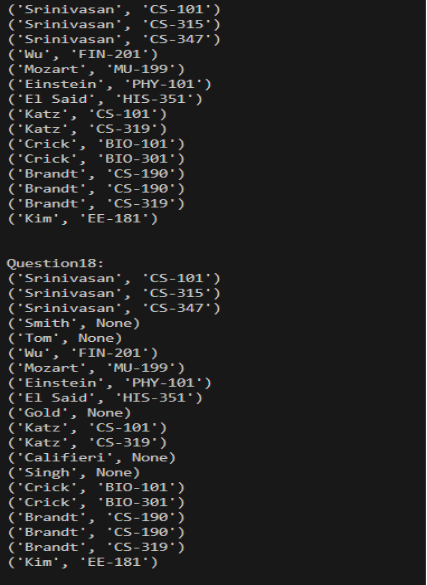
**OUTPUT:-**

****

****

****

****

****

**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 (empt\_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)

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

00002802092CBF85D6CCC64E378DB40C241BC48A1ECBF894A198384E17AEBAC6B87352B73C0041DC

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)

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

000028020910F8CD3CD081404F94F75749C43250F11120502A851A457B8CB2C74EFB9CDFF60041DC

F90000

00002802098AB414E0B94F403CB9CEF071DFEB70EE1120502A851A457B8CB2C74EFB9CDFF60041DC

F90001

00002802092D6F9946325940C1A898749FD0ADF9831120502A851A457B8CB2C74EFB9CDFF60041DC

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>