Department of Computer Science

Advanced Database Techniques Journal

MSc Computer Science (Part 1) Semester I

Name: AVINASH KAURAN
Roll No: CS22006

R.D. & S. H. National College & S. W.A. Science College

Bandra, Mumbai – 400050.

Department of Computer Science

CERTIFICATE

This is to certify that Mr./Ms. **AVINASH RAJKUMAR KAURAN** of **Msc..CS** class (**I** Semester) has satisfactorily completed **8** Practicals, in the subject of **Advance Database Technologies** as a part of M.Sc. Degree Course in Computer Science during the academic year 20 22 – 20 23 .

Date of Certification:

Faculty Incharge

Head,
Department Computer Science

Signature of Examiner

AVINASH KAURAN:CS22006

INDEX

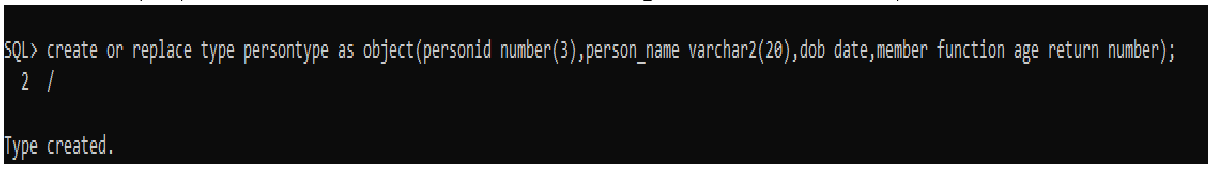
PRACTICAL NO: 01

Part A:

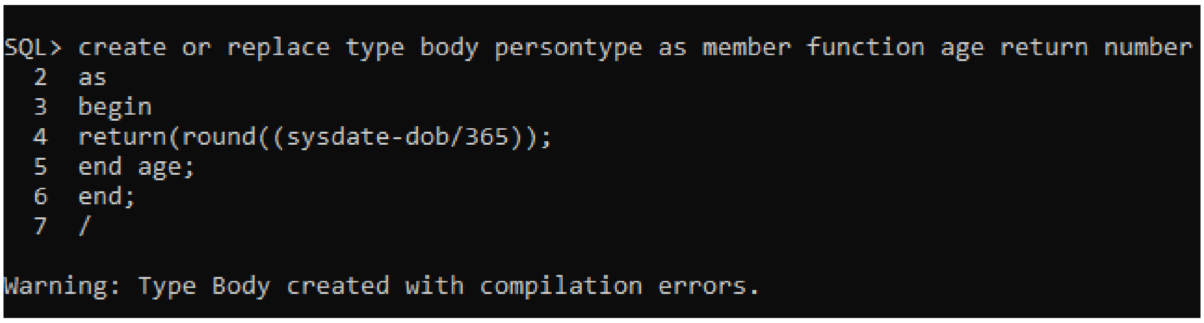
- Create the following types using OODBMS:
- PersonType (Per_id, Per_name, DOB)
- DOB is the date of birth of a person. Include a method to find the age of a person.
- BankType (account_no, per_id, balance)

Create the appropriate tables. Insert 5 tuples and fire the following queries:
(i) Find name and age of a person
ii)Find all names with account number.
iii) Find the names with balance more than 12,000

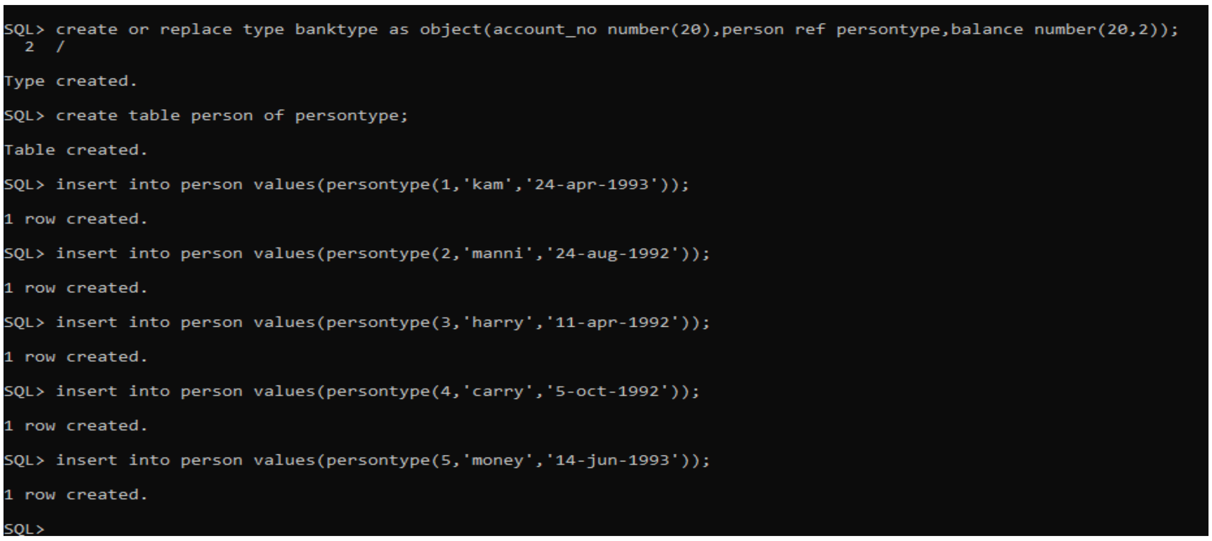
create or replace type persontype as object(personid number(3), person_name varchar2(20), dob date, member function age return number);



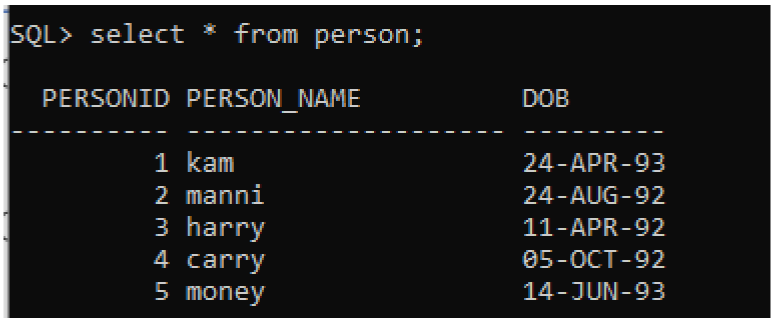
```
SQL> create or replace type persontype as object(personid number(3),person_name varchar2(20),dob date,member function age return number);
  2  /

Type created.
```



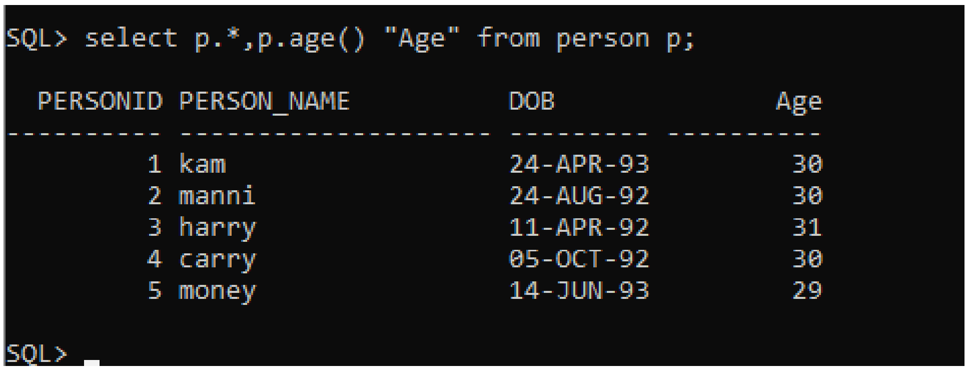
```
SQL> create or replace type body persontype as member function age return number
  2  as
  3  begin
  4  return(round((sysdate-dob/365));
  5  end age;
  6  end;
  7  /

Warning: Type Body created with compilation errors.
```



```
SQL> create or replace type banktype as object(account_no number(20),person ref persontype,balance number(20,2));
  2  /

Type created.

SQL> create table person of persontype;

Table created.

SQL> insert into person values(persontype(1,'kam','24-apr-1993'));

1 row created.

SQL> insert into person values(persontype(2,'manni','24-aug-1992'));

1 row created.

SQL> insert into person values(persontype(3,'harry','11-apr-1992'));

1 row created.

SQL> insert into person values(persontype(4,'carry','5-oct-1992'));

1 row created.

SQL> insert into person values(persontype(5,'money','14-jun-1993'));

1 row created.

SQL>
```

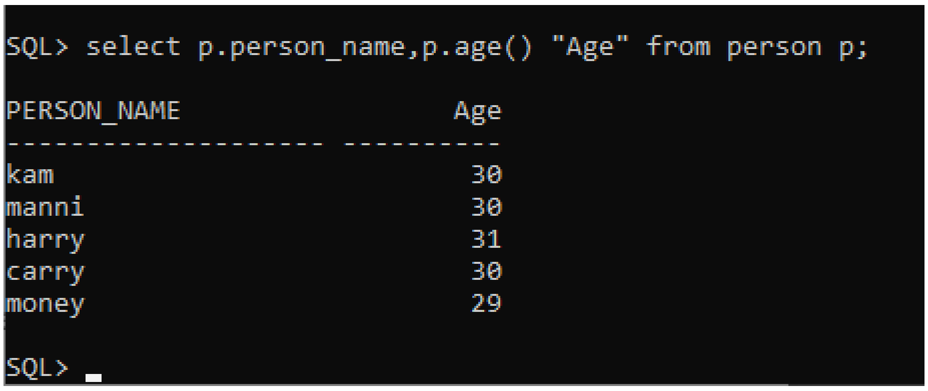


```
SQL> select * from person;

  PERSONID PERSON_NAME          DOB
---------- -------------------- ---------
         1 kam                  24-APR-93
         2 manni                24-AUG-92
         3 harry                11-APR-92
         4 carry                05-OCT-92
         5 money                14-JUN-93
```

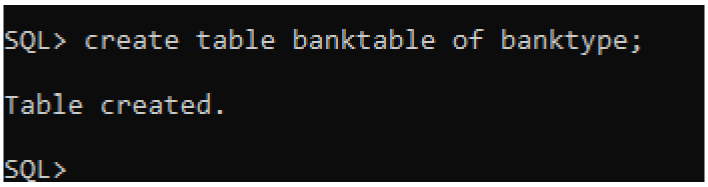


```
SQL> select p.*,p.age() "Age" from person p;

  PERSONID PERSON_NAME          DOB              Age
---------- -------------------- --------- ----------
         1 kam                  24-APR-93         30
         2 manni                24-AUG-92         30
         3 harry                11-APR-92         31
         4 carry                05-OCT-92         30
         5 money                14-JUN-93         29

SQL> _
```

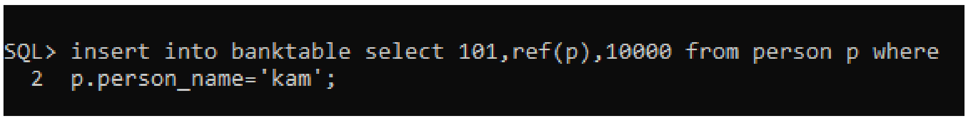
(i) Find name and age of a person



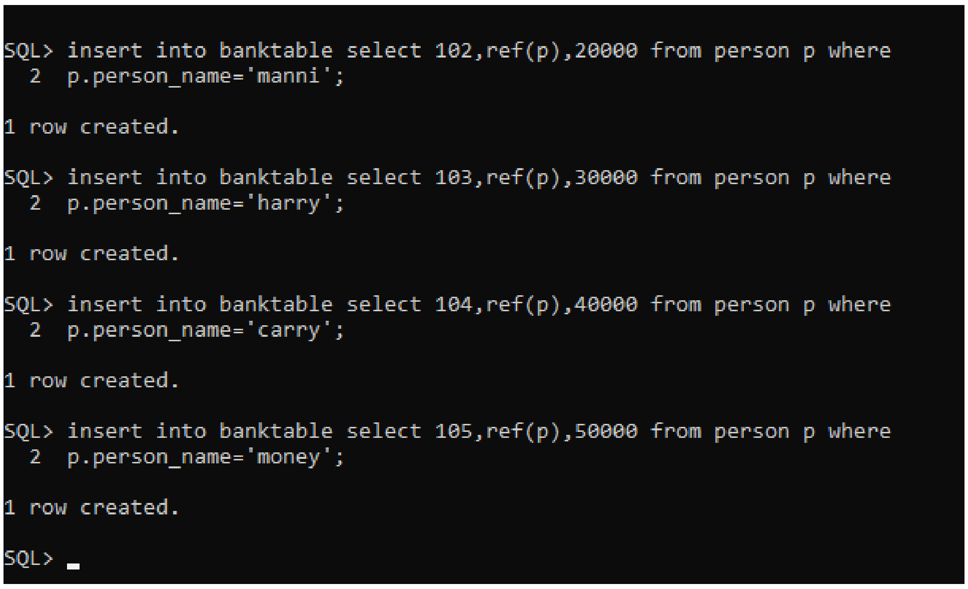
```
SQL> select p.person_name,p.age() "Age" from person p;

PERSON_NAME                 Age
-------------------- ----------
kam                          30
manni                        30
harry                        31
carry                        30
money                        29

SQL> _
```



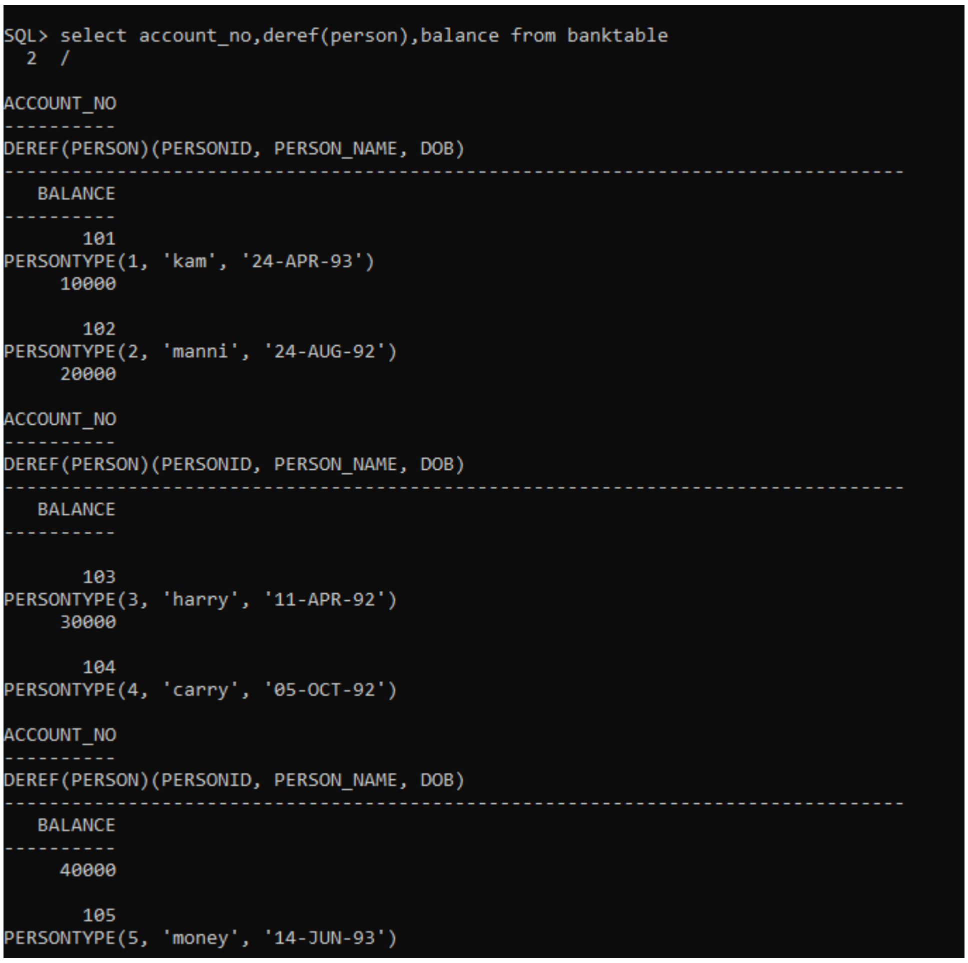
```
SQL> create table banktable of banktype;

Table created.

SQL>
```



```
SQL> insert into banktable select 101,ref(p),10000 from person p where
  2  p.person_name='kam';
```

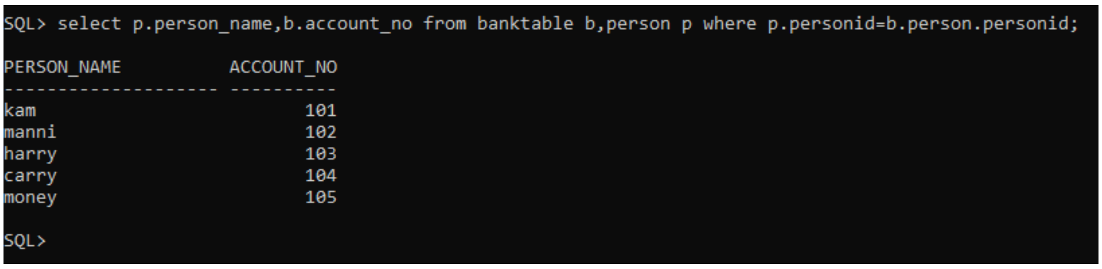


```
SQL> insert into banktable select 102,ref(p),20000 from person p where
  2  p.person_name='manni';

1 row created.

SQL> insert into banktable select 103,ref(p),30000 from person p where
  2  p.person_name='harry';

1 row created.

SQL> insert into banktable select 104,ref(p),40000 from person p where
  2  p.person_name='carry';

1 row created.

SQL> insert into banktable select 105,ref(p),50000 from person p where
  2  p.person_name='money';

1 row created.

SQL> _
```



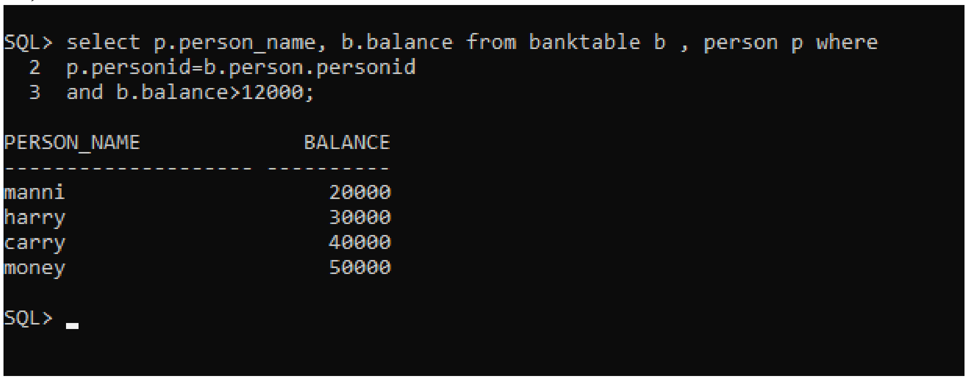
```
SQL> select account_no,deref(person),balance from banktable
  2  /

ACCOUNT_NO
----------
DEREF(PERSON)(PERSONID, PERSON_NAME, DOB)
--------------------------------------------------------------------------------
   BALANCE
----------
       101
PERSONTYPE(1, 'kam', '24-APR-93')
     10000

       102
PERSONTYPE(2, 'manni', '24-AUG-92')
     20000

ACCOUNT_NO
----------
DEREF(PERSON)(PERSONID, PERSON_NAME, DOB)
--------------------------------------------------------------------------------
   BALANCE
----------

       103
PERSONTYPE(3, 'harry', '11-APR-92')
     30000

       104
PERSONTYPE(4, 'carry', '05-OCT-92')

ACCOUNT_NO
----------
DEREF(PERSON)(PERSONID, PERSON_NAME, DOB)
--------------------------------------------------------------------------------
   BALANCE
----------
     40000

       105
PERSONTYPE(5, 'money', '14-JUN-93')
```

ii)Find all names with account number



```
SQL> select p.person_name,b.account_no from banktable b,person p where p.personid=b.person.personid;

PERSON_NAME          ACCOUNT_NO
-------------------- ----------
kam                         101
manni                       102
harry                       103
carry                       104
money                       105

SQL>
```

iii) Find the names with balance more than 12,000

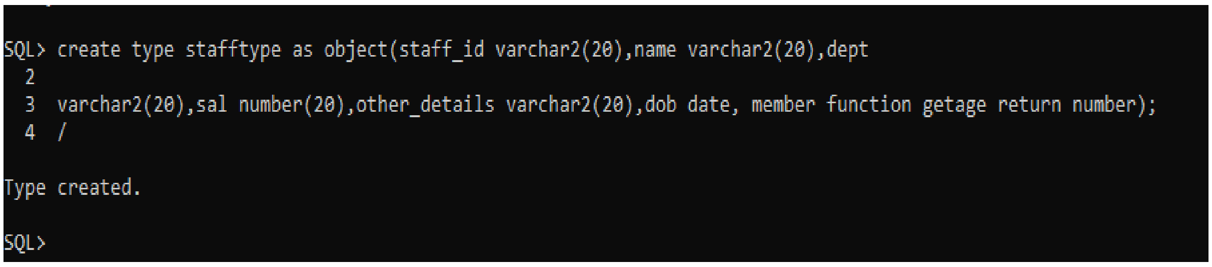


```
SQL> select p.person_name, b.balance from banktable b , person p where
  2  p.personid=b.person.personid
  3  and b.balance>12000;

PERSON_NAME             BALANCE
-------------------- ----------
manni                     20000
harry                     30000
carry                     40000
money                     50000

SQL>
```

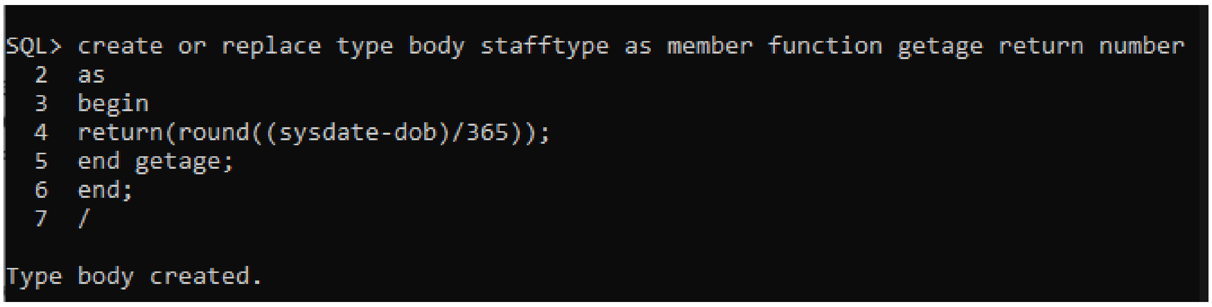
Part. B.
Using object oriented databases, create the following types
a)Staff(staff id,name, dept, sal, other detail, dob,GetAge)
b)Depttype(dept id,name, location, emp)
Next create the following tables:
Stafftable of Staff
Depttable of Depttype nested with emp and store it in reldept

Fire following queries:
1.List all the staff name from the stafftable.
2.List the age of the all staff name.
3.List the staff name, id, dept, sal, age from the stafftable.
4.List the staff id, dept name from the depttype table.
5.Count the no. of name for given dept name='manager'.
6.List the all record of given staff name='jigi'.
7.Count the dept name from table depttype.

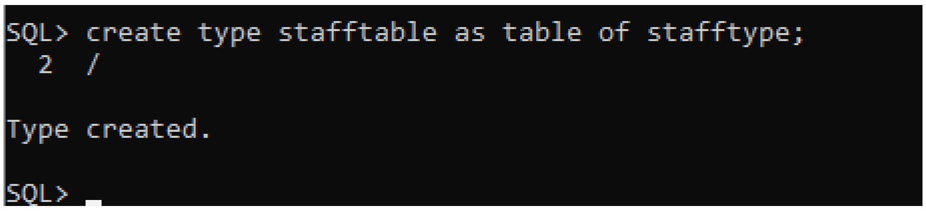
create type stafftype as object(staff_id varchar2(20),name varchar2(20),dept varchar2(20),sal number(20),other_details varchar2(20),dob date, member function getage return number);



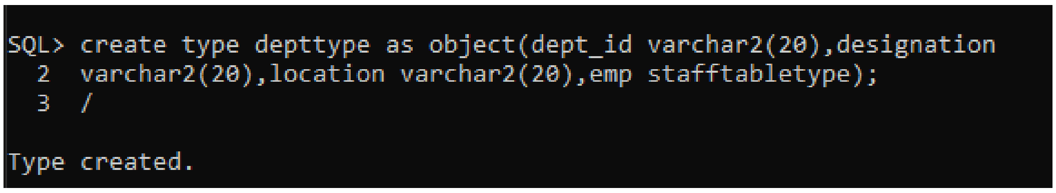
```
SQL> create type stafftype as object(staff_id varchar2(20),name varchar2(20),dept

     varchar2(20),sal number(20),other_details varchar2(20),dob date, member function getage return number);
     /

Type created.

SQL>
```



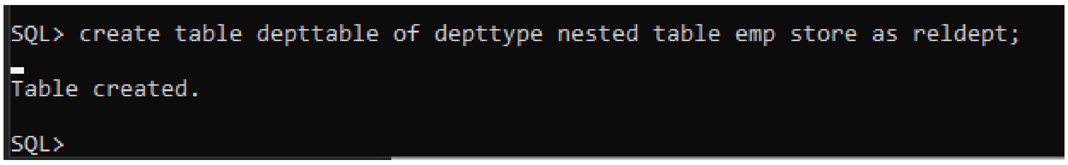
```
SQL> create or replace type body stafftype as member function getage return number
     as
     begin
     return(round((sysdate-dob)/365));
     end getage;
     end;
     /

Type body created.
```



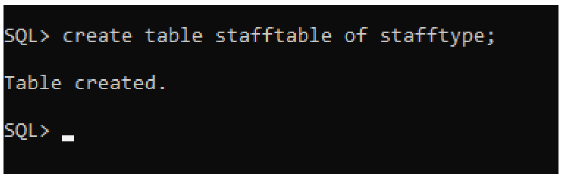
```
SQL> create type stafftable as table of stafftype;
     /

Type created.

SQL>
```



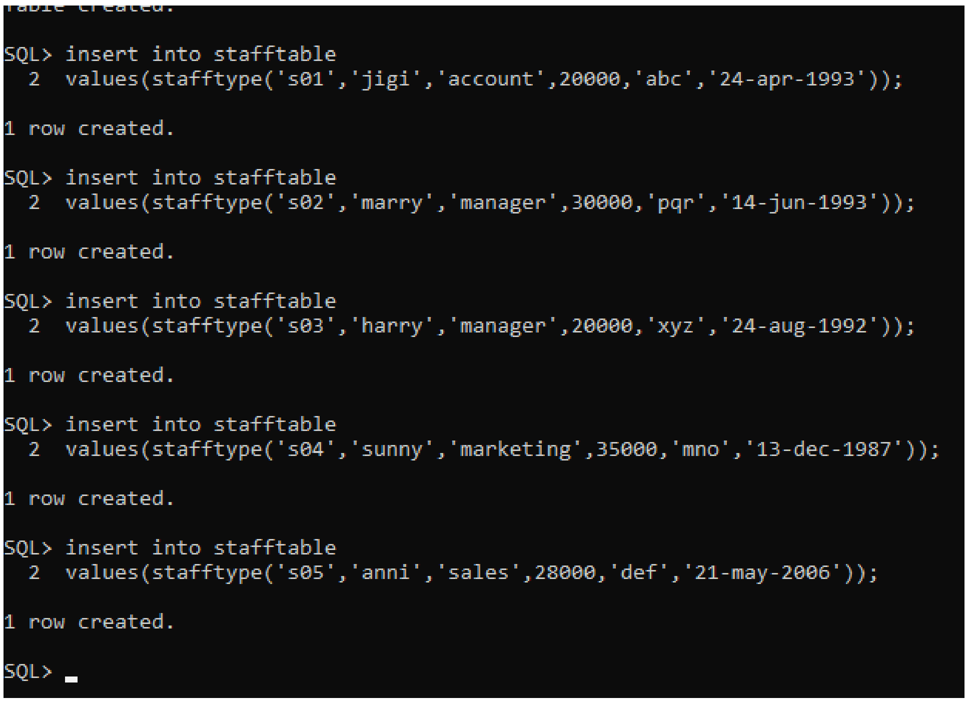
```
SQL> create type depttype as object(dept_id varchar2(20),designation
     varchar2(20),location varchar2(20),emp stafftabletype);
     /

Type created.
```



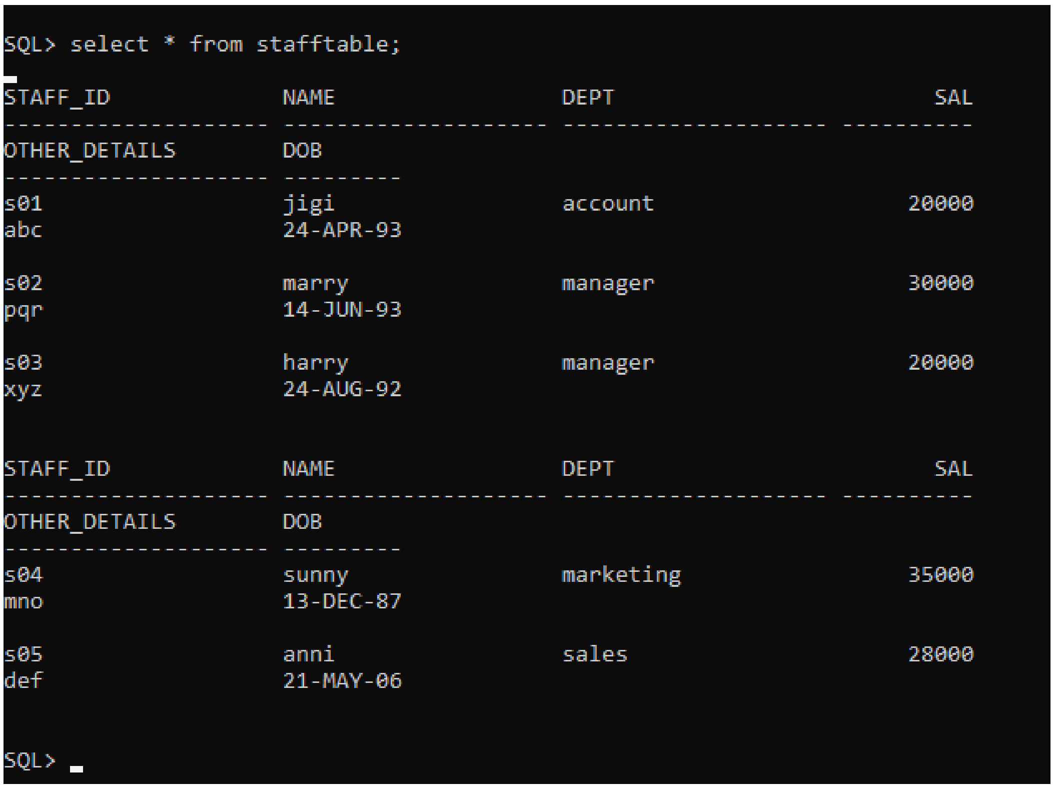
```
SQL> create table depttable of depttype nested table emp store as reldept;

Table created.

SQL>
```



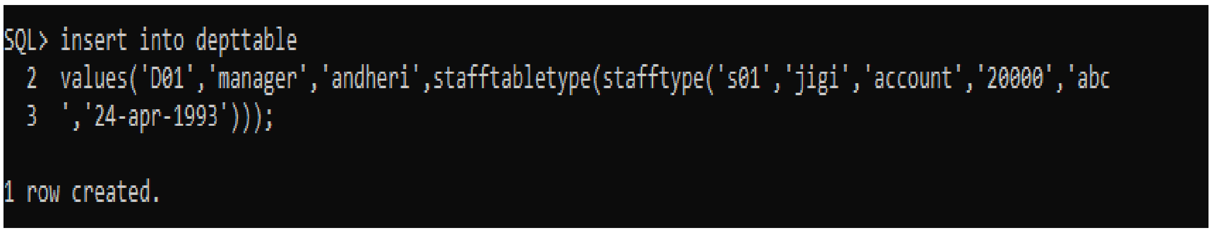
```
SQL> create table stafftable of stafftype;

Table created.

SQL>
```



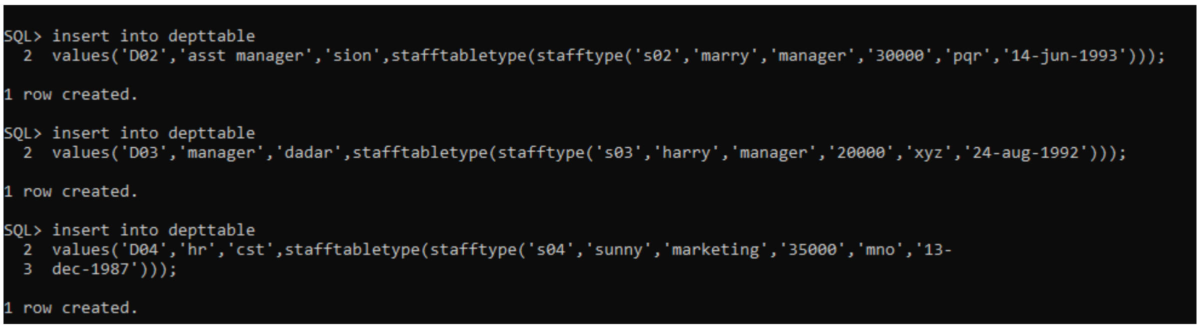
```
SQL> insert into stafftable
  2  values(stafftype('s01','jigi','account',20000,'abc','24-apr-1993'));

1 row created.

SQL> insert into stafftable
  2  values(stafftype('s02','marry','manager',30000,'pqr','14-jun-1993'));

1 row created.

SQL> insert into stafftable
  2  values(stafftype('s03','harry','manager',20000,'xyz','24-aug-1992'));

1 row created.

SQL> insert into stafftable
  2  values(stafftype('s04','sunny','marketing',35000,'mno','13-dec-1987'));

1 row created.

SQL> insert into stafftable
  2  values(stafftype('s05','anni','sales',28000,'def','21-may-2006'));

1 row created.

SQL>
```



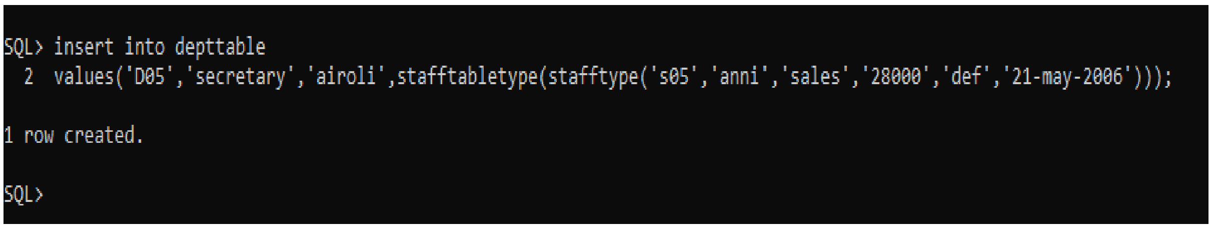
```
SQL> select * from stafftable;

STAFF_ID             NAME                 DEPT                        SAL
-------------------- -------------------- -------------------- ----------
OTHER_DETAILS        DOB
-------------------- ---------
s01                  jigi                 account                   20000
abc                  24-APR-93

s02                  marry                manager                   30000
pqr                  14-JUN-93

s03                  harry                manager                   20000
xyz                  24-AUG-92


STAFF_ID             NAME                 DEPT                        SAL
-------------------- -------------------- -------------------- ----------
OTHER_DETAILS        DOB
-------------------- ---------
s04                  sunny                marketing                 35000
mno                  13-DEC-87

s05                  anni                 sales                     28000
def                  21-MAY-06


SQL>
```



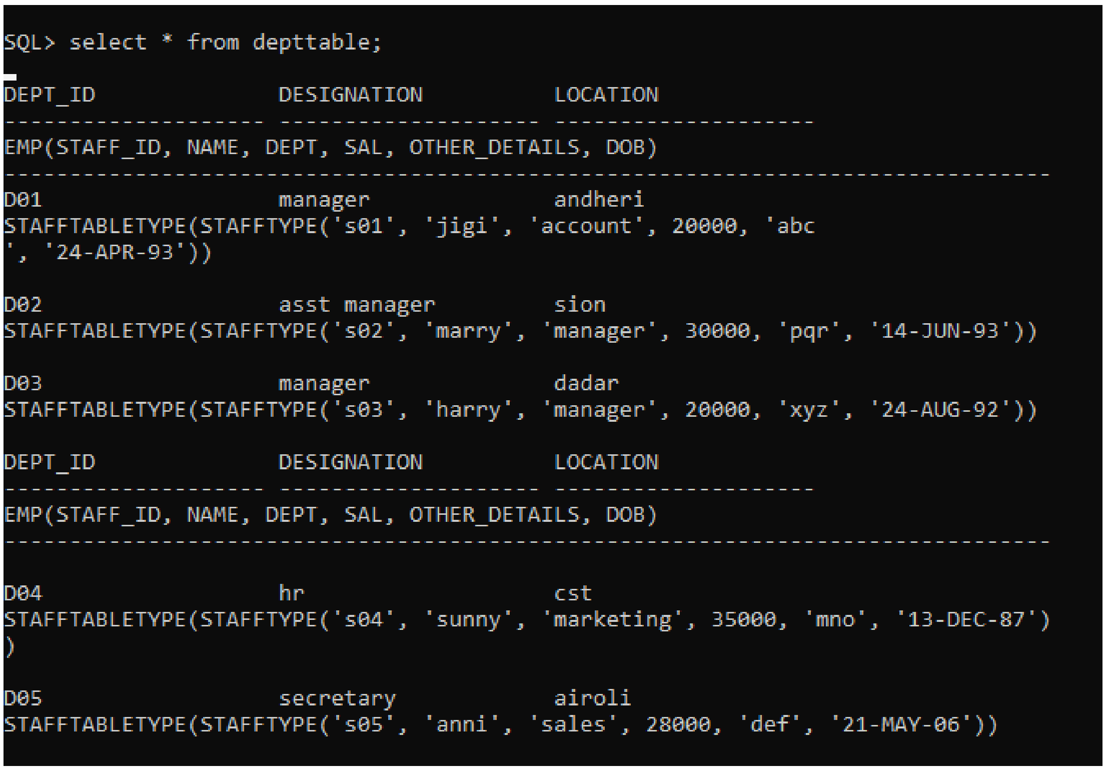
```
SQL> insert into depttable
  2  values('D01','manager','andheri',stafftabletype(stafftype('s01','jigi','account','20000','abc
  3  ','24-apr-1993')));

1 row created.
```



```
SQL> insert into depttable
  2  values('D02','asst manager','sion',stafftabletype(stafftype('s02','marry','manager','30000','pqr','14-jun-1993')));

1 row created.

SQL> insert into depttable
  2  values('D03','manager','dadar',stafftabletype(stafftype('s03','harry','manager','20000','xyz','24-aug-1992')));

1 row created.

SQL> insert into depttable
  2  values('D04','hr','cst',stafftabletype(stafftype('s04','sunny','marketing','35000','mno','13-
  3  dec-1987')));

1 row created.
```

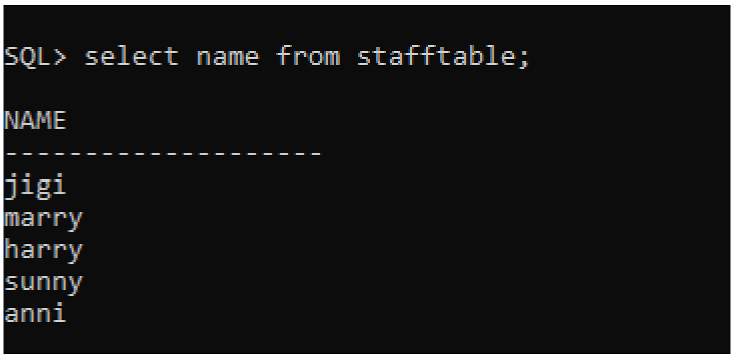


```
SQL> insert into depttable
  2  values('D05','secretary','airoli',stafftabletype(stafftype('s05','anni','sales','28000','def','21-may-2006')));

1 row created.

SQL>
```



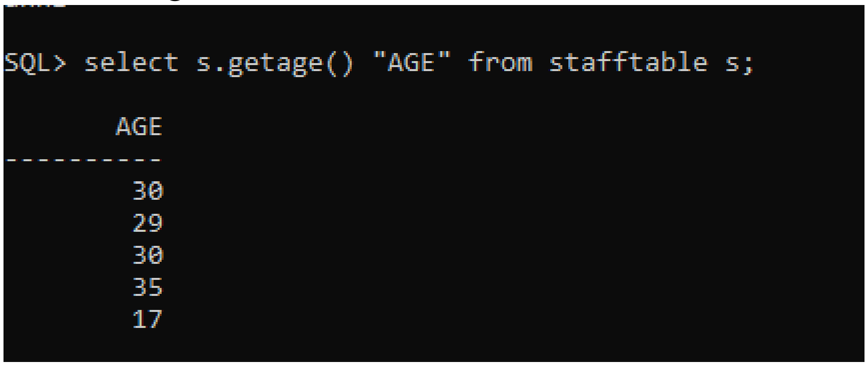
```
SQL> select * from depttable;

DEPT_ID              DESIGNATION          LOCATION
-------------------- -------------------- --------------------
EMP(STAFF_ID, NAME, DEPT, SAL, OTHER_DETAILS, DOB)
--------------------------------------------------------------------------------
D01                  manager              andheri
STAFFTABLETYPE(STAFFTYPE('s01', 'jigi', 'account', 20000, 'abc
', '24-APR-93'))

D02                  asst manager         sion
STAFFTABLETYPE(STAFFTYPE('s02', 'marry', 'manager', 30000, 'pqr', '14-JUN-93'))

D03                  manager              dadar
STAFFTABLETYPE(STAFFTYPE('s03', 'harry', 'manager', 20000, 'xyz', '24-AUG-92'))

DEPT_ID              DESIGNATION          LOCATION
-------------------- -------------------- --------------------
EMP(STAFF_ID, NAME, DEPT, SAL, OTHER_DETAILS, DOB)
--------------------------------------------------------------------------------

D04                  hr                   cst
STAFFTABLETYPE(STAFFTYPE('s04', 'sunny', 'marketing', 35000, 'mno', '13-DEC-87')
)

D05                  secretary            airoli
STAFFTABLETYPE(STAFFTYPE('s05', 'anni', 'sales', 28000, 'def', '21-MAY-06'))
```

1.List all the staff name from the stafftable.



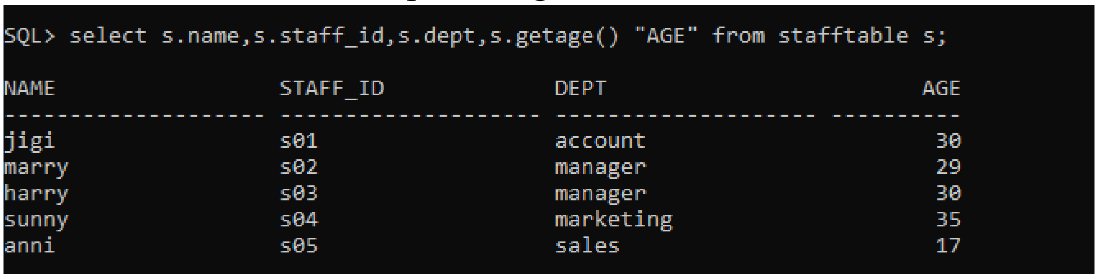
```
SQL> select name from stafftable;

NAME
--------------------
jigi
marry
harry
sunny
anni
```

2.List the age of the all-staff name.



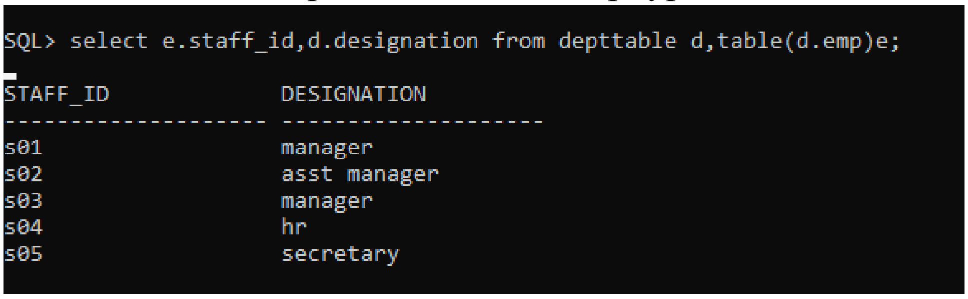
```
SQL> select s.getage() "AGE" from stafftable s;

       AGE
----------
        30
        29
        30
        35
        17
```

3.List the staff name, id, dept, sal, age from the stafftable.



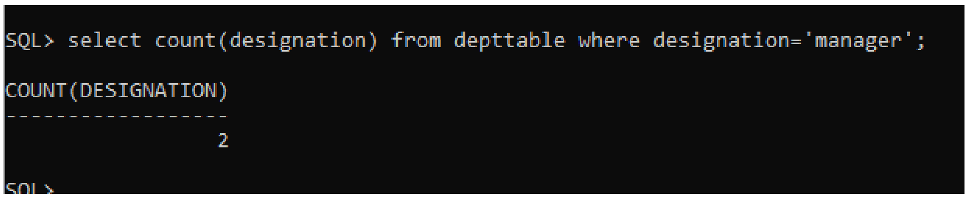
```
SQL> select s.name,s.staff_id,s.dept,s.getage() "AGE" from stafftable s;

NAME                 STAFF_ID             DEPT                        AGE
-------------------- -------------------- -------------------- ----------
jigi                 s01                  account                      30
marry                s02                  manager                      29
harry                s03                  manager                      30
sunny                s04                  marketing                    35
anni                 s05                  sales                        17
```

4.List the staff id, dept name from the depttype table.



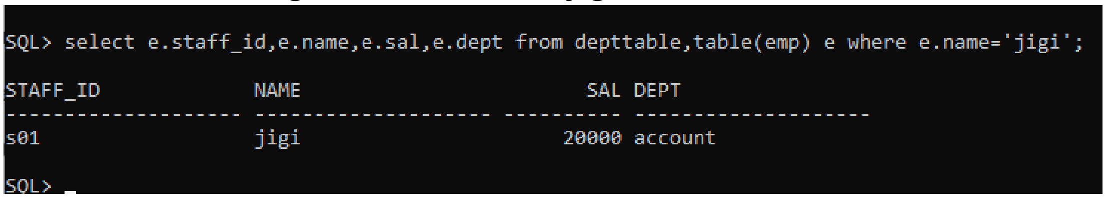
```
SQL> select e.staff_id,d.designation from depttable d,table(d.emp)e;

STAFF_ID             DESIGNATION
-------------------- --------------------
s01                  manager
s02                  asst manager
s03                  manager
s04                  hr
s05                  secretary
```

5.Count the no. of name for given dept name=’manager’.



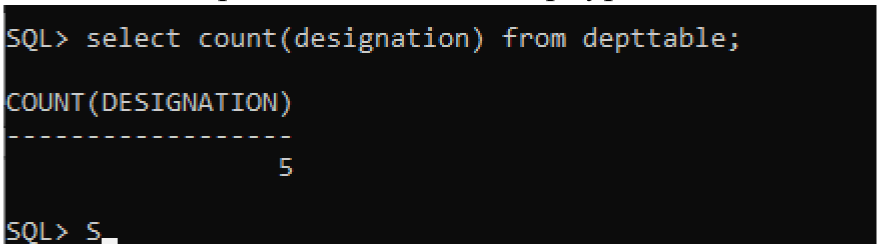
```
SQL> select count(designation) from depttable where designation='manager';

COUNT(DESIGNATION)
------------------
                 2

SQL>
```

6.List all record of given staff name='jigi'.



```
SQL> select e.staff_id,e.name,e.sal,e.dept from depttable,table(emp) e where e.name='jigi';

STAFF_ID             NAME                        SAL DEPT
-------------------- -------------------- ---------- --------------------
s01                  jigi                      20000 account

SQL> _
```

7.Count the dept name from table depttype.

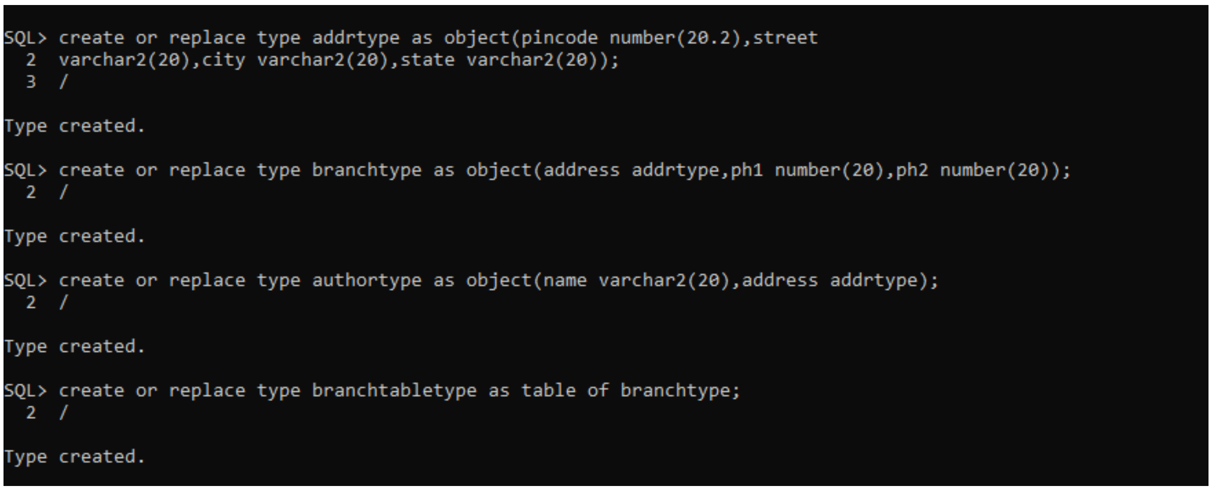


```
SQL> select count(designation) from depttable;

COUNT(DESIGNATION)
------------------
                 5

SQL> S_
```

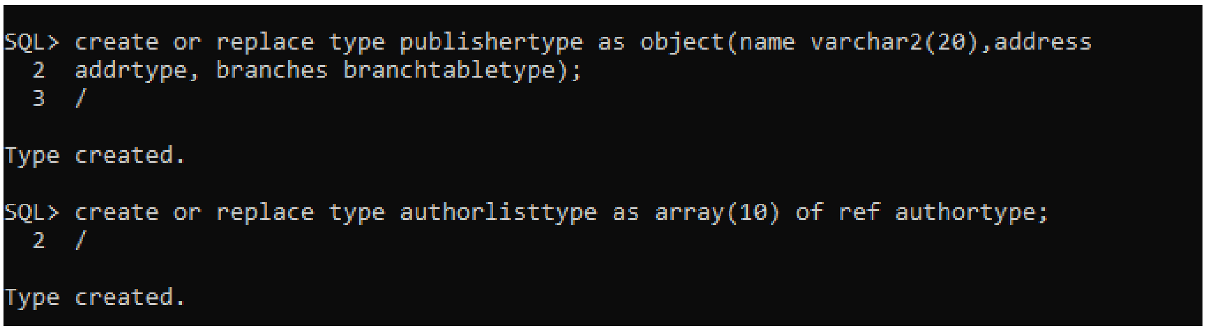
Part.C.
Using object oriented databases, create the following types:
a)addrtype(pincode, street, city, state)
b)branchtype(address, phone1, phone2)
c)authortype(name, addr)
d)publishertype(name, addr, branches)
e)authorlisttype as varray, which is reference to authortype

Next create the following tables:
f)branchtabletype of branchtype
g)authors of authortype
h)books(title, year, publisher by ref publishertype, authors)
i)publishers of publisher type

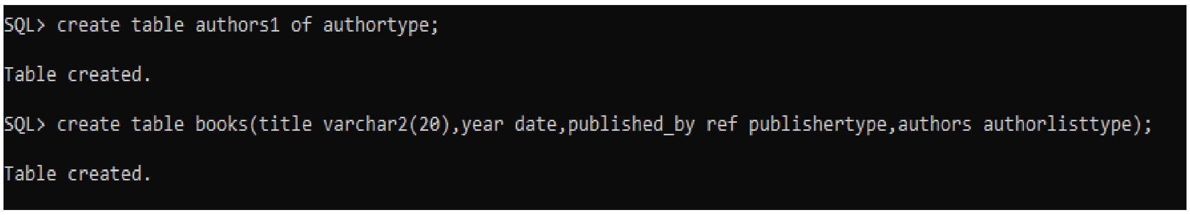
Fire following queries:
1.List all of the authors that have the same pin code as their publisher.
2.List all books that have 2 or more authors.
3.List all authors who have published more than one book.
4.Name the title of the book that has the most authors.



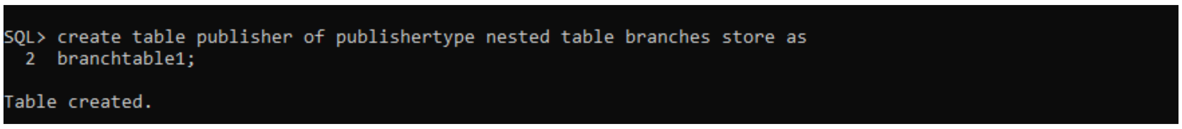
```
SQL> create or replace type addrtype as object(pincode number(20.2),street
  2  varchar2(20),city varchar2(20),state varchar2(20));
  3  /

Type created.

SQL> create or replace type branchtype as object(address addrtype,ph1 number(20),ph2 number(20));
  2  /

Type created.

SQL> create or replace type authortype as object(name varchar2(20),address addrtype);
  2  /

Type created.

SQL> create or replace type branchtabletype as table of branchtype;
  2  /

Type created.
```



```
SQL> create or replace type publishertype as object(name varchar2(20),address
  2  addrtype, branches branchtabletype);
  3  /

Type created.

SQL> create or replace type authorlisttype as array(10) of ref authortype;
  2  /

Type created.
```

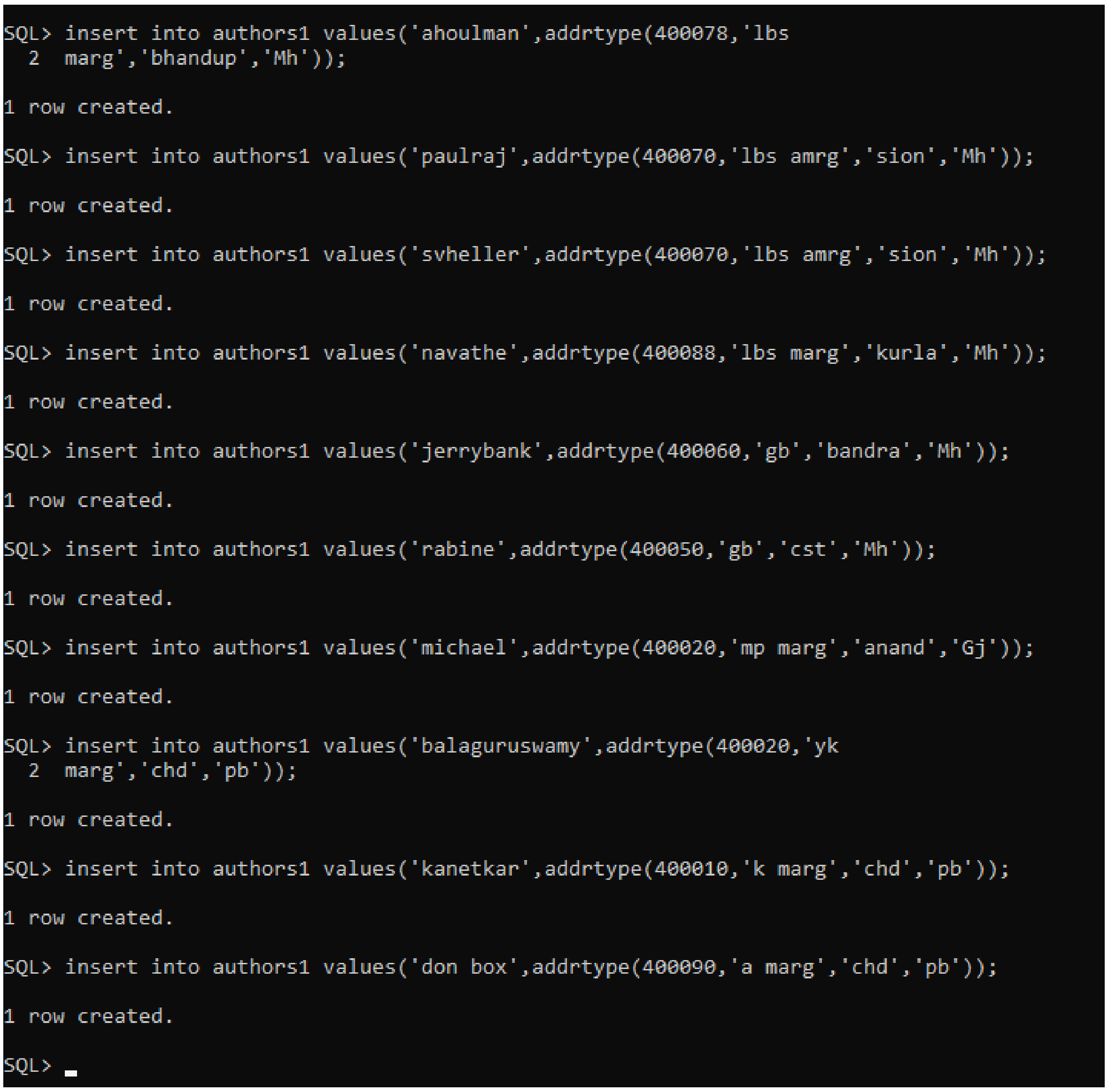


```
SQL> create table authors1 of authortype;

Table created.

SQL> create table books(title varchar2(20),year date,published_by ref publishertype,authors authorlisttype);

Table created.
```



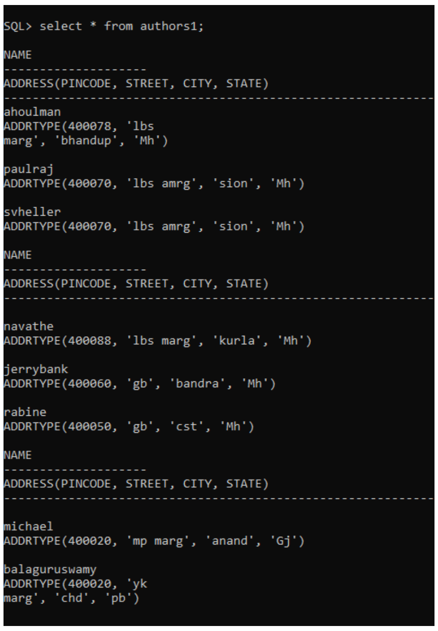
```
SQL> create table publisher of publishertype nested table branches store as
  2  branchtable1;

Table created.
```

Inserting Values in author's table

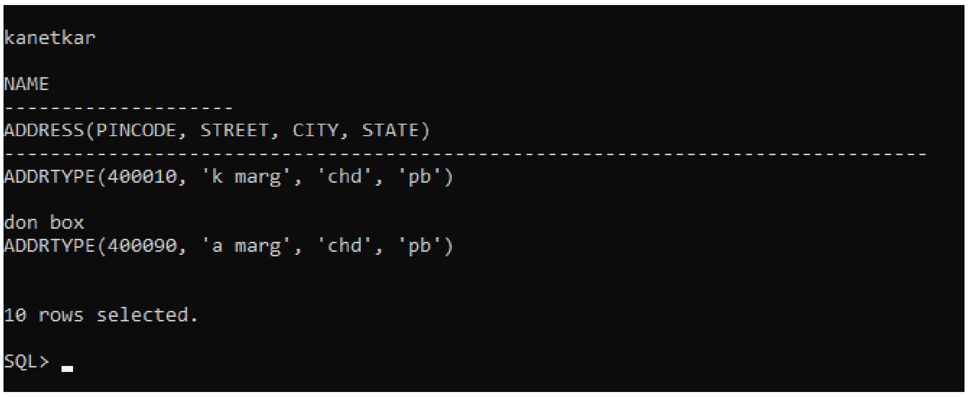


```
SQL> insert into authors1 values('ahoulman',addrtype(400078,'lbs
  2  marg','bhandup','Mh'));

1 row created.

SQL> insert into authors1 values('paulraj',addrtype(400070,'lbs amrg','sion','Mh'));

1 row created.

SQL> insert into authors1 values('svheller',addrtype(400070,'lbs amrg','sion','Mh'));

1 row created.

SQL> insert into authors1 values('navathe',addrtype(400088,'lbs marg','kurla','Mh'));

1 row created.

SQL> insert into authors1 values('jerrybank',addrtype(400060,'gb','bandra','Mh'));

1 row created.

SQL> insert into authors1 values('rabine',addrtype(400050,'gb','cst','Mh'));

1 row created.

SQL> insert into authors1 values('michael',addrtype(400020,'mp marg','anand','Gj'));

1 row created.

SQL> insert into authors1 values('balaguruswamy',addrtype(400020,'yk
  2  marg','chd','pb'));

1 row created.

SQL> insert into authors1 values('kanetkar',addrtype(400010,'k marg','chd','pb'));

1 row created.

SQL> insert into authors1 values('don box',addrtype(400090,'a marg','chd','pb'));

1 row created.

SQL> _
```

select * from authors1;

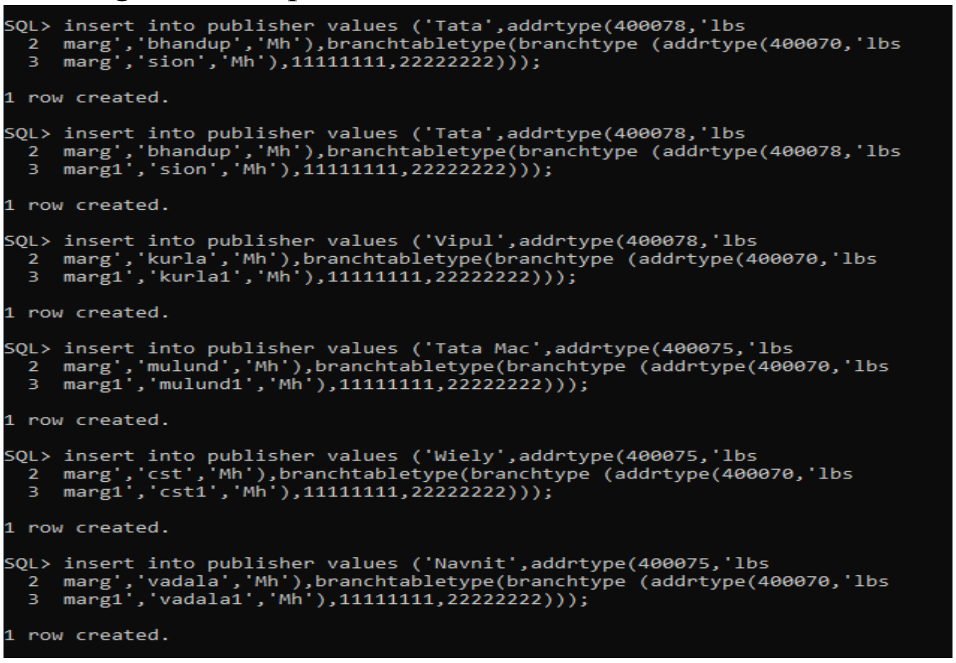


```
SQL> select * from authors1;

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
ahoulman
ADDRTYPE(400078, 'lbs
marg', 'bhandup', 'Mh')

paulraj
ADDRTYPE(400070, 'lbs amrg', 'sion', 'Mh')

svheller
ADDRTYPE(400070, 'lbs amrg', 'sion', 'Mh')

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------

navathe
ADDRTYPE(400088, 'lbs marg', 'kurla', 'Mh')

jerrybank
ADDRTYPE(400060, 'gb', 'bandra', 'Mh')

rabine
ADDRTYPE(400050, 'gb', 'cst', 'Mh')

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------

michael
ADDRTYPE(400020, 'mp marg', 'anand', 'Gj')

balaguruswamy
ADDRTYPE(400020, 'yk
marg', 'chd', 'pb')
```



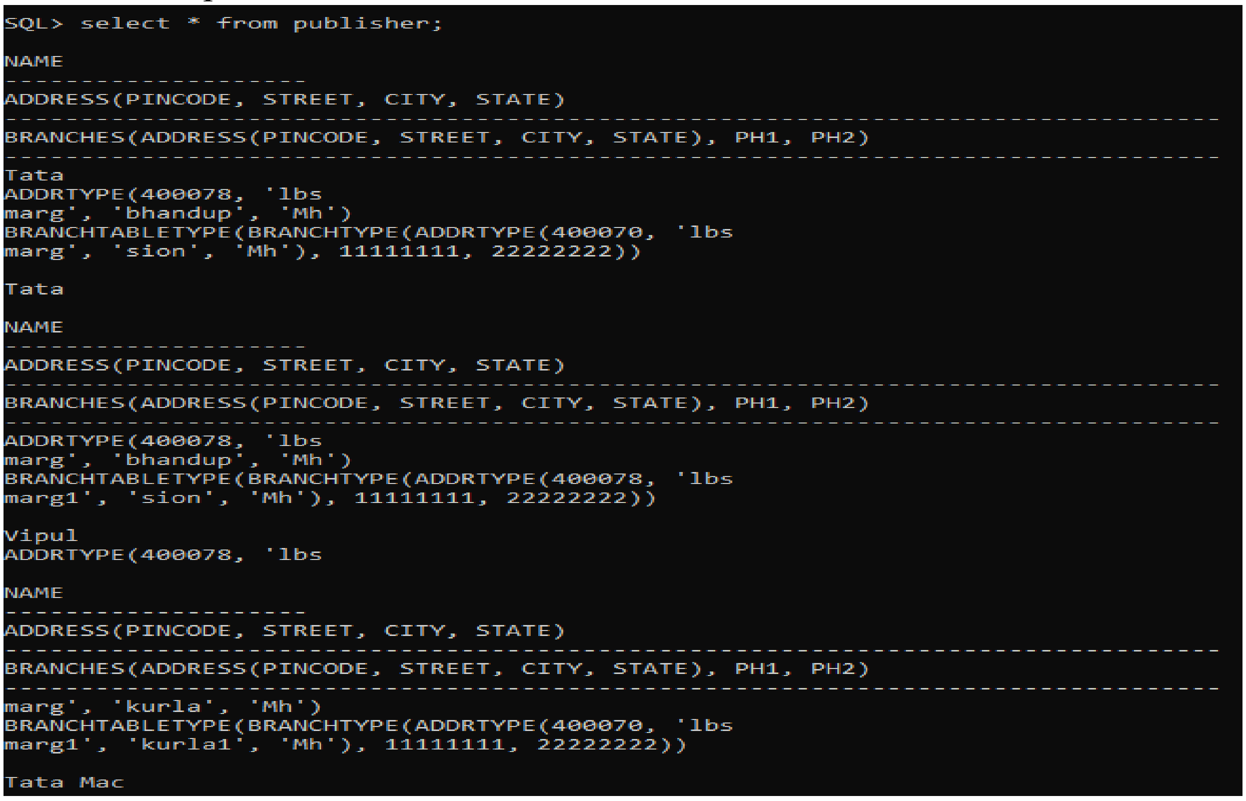
```
kanetkar

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
ADDRTYPE(400010, 'k marg', 'chd', 'pb')

don box
ADDRTYPE(400090, 'a marg', 'chd', 'pb')


10 rows selected.

SQL>
```

Inserting Values in publisher's table

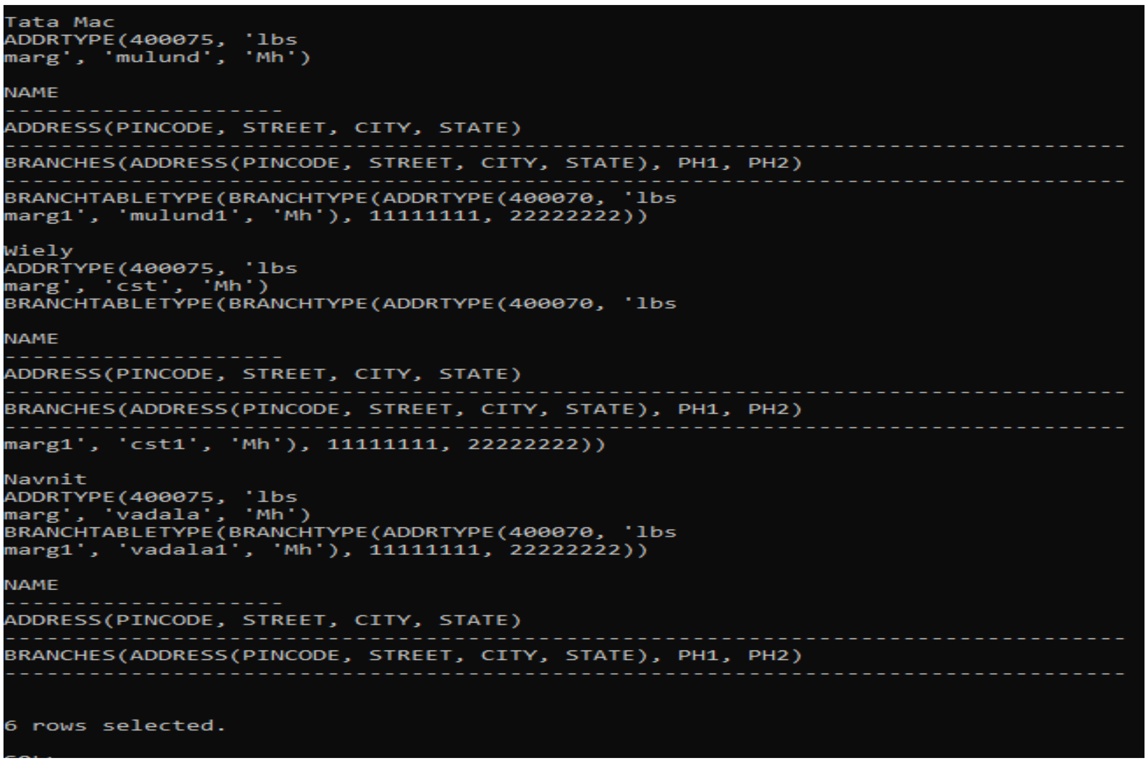


```
SQL> insert into publisher values ('Tata',addrtype(400078,'lbs
  2  marg','bhandup','Mh'),branchtabletype(branchtype (addrtype(400070,'lbs
  3  marg','sion','Mh'),11111111,22222222)));

1 row created.

SQL> insert into publisher values ('Tata',addrtype(400078,'lbs
  2  marg','bhandup','Mh'),branchtabletype(branchtype (addrtype(400078,'lbs
  3  marg1','sion','Mh'),11111111,22222222)));

1 row created.

SQL> insert into publisher values ('Vipul',addrtype(400078,'lbs
  2  marg','kurla','Mh'),branchtabletype(branchtype (addrtype(400070,'lbs
  3  marg1','kurla1','Mh'),11111111,22222222)));

1 row created.

SQL> insert into publisher values ('Tata Mac',addrtype(400075,'lbs
  2  marg','mulund','Mh'),branchtabletype(branchtype (addrtype(400070,'lbs
  3  marg1','mulund1','Mh'),11111111,22222222)));

1 row created.

SQL> insert into publisher values ('Wiely',addrtype(400075,'lbs
  2  marg','cst','Mh'),branchtabletype(branchtype (addrtype(400070,'lbs
  3  marg1','cst1','Mh'),11111111,22222222)));

1 row created.

SQL> insert into publisher values ('Navnit',addrtype(400075,'lbs
  2  marg','vadala','Mh'),branchtabletype(branchtype (addrtype(400070,'lbs
  3  marg1','vadala1','Mh'),11111111,22222222)));

1 row created.
```

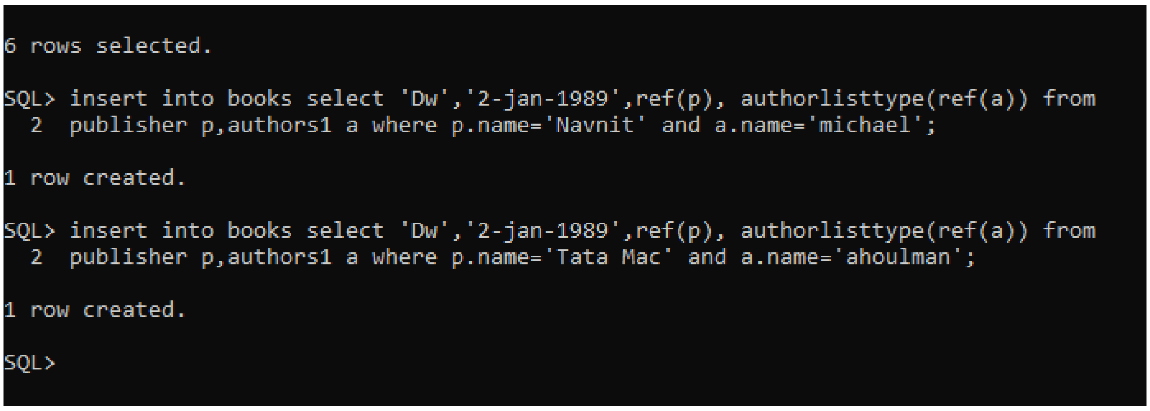
select * from publisher;



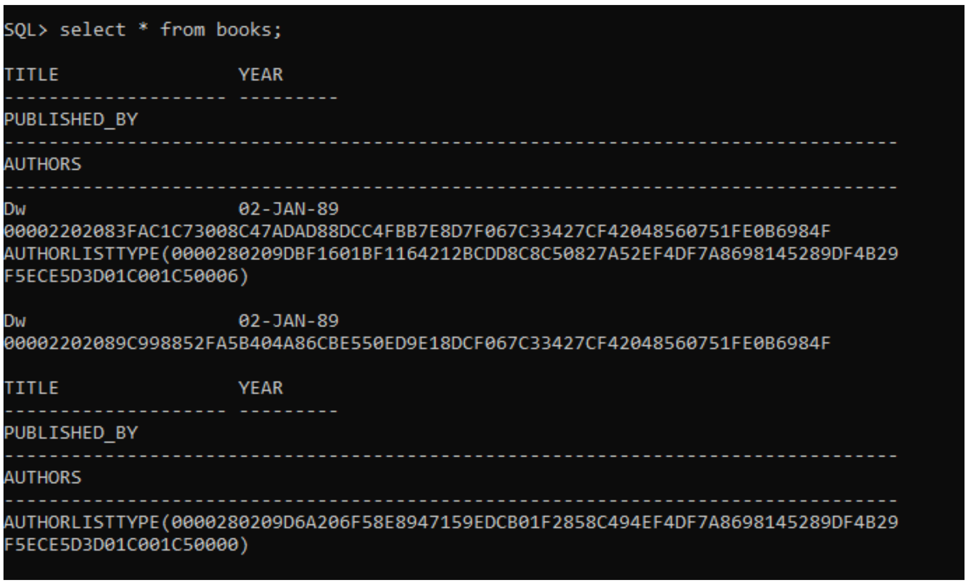
```
SQL> select * from publisher;

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
BRANCHES(ADDRESS(PINCODE, STREET, CITY, STATE), PH1, PH2)
--------------------------------------------------------------------------------
Tata
ADDRTYPE(400078, 'lbs
marg', 'bhandup', 'Mh')
BRANCHTABLETYPE(BRANCHTYPE(ADDRTYPE(400070, 'lbs
marg', 'sion', 'Mh'), 11111111, 22222222))

Tata

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
BRANCHES(ADDRESS(PINCODE, STREET, CITY, STATE), PH1, PH2)
--------------------------------------------------------------------------------
ADDRTYPE(400078, 'lbs
marg', 'bhandup', 'Mh')
BRANCHTABLETYPE(BRANCHTYPE(ADDRTYPE(400078, 'lbs
marg1', 'sion', 'Mh'), 11111111, 22222222))

Vipul
ADDRTYPE(400078, 'lbs

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
BRANCHES(ADDRESS(PINCODE, STREET, CITY, STATE), PH1, PH2)
--------------------------------------------------------------------------------
marg', 'kurla', 'Mh')
BRANCHTABLETYPE(BRANCHTYPE(ADDRTYPE(400070, 'lbs
marg1', 'kurla1', 'Mh'), 11111111, 22222222))

Tata Mac
```



```
Tata Mac
ADDRTYPE(400075, 'lbs
marg', 'mulund', 'Mh')

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
BRANCHES(ADDRESS(PINCODE, STREET, CITY, STATE), PH1, PH2)
--------------------------------------------------------------------------------
BRANCHTABLETYPE(BRANCHTYPE(ADDRTYPE(400070, 'lbs
marg1', 'mulund1', 'Mh'), 11111111, 22222222))

Wiely
ADDRTYPE(400075, 'lbs
marg', 'cst', 'Mh')
BRANCHTABLETYPE(BRANCHTYPE(ADDRTYPE(400070, 'lbs

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
BRANCHES(ADDRESS(PINCODE, STREET, CITY, STATE), PH1, PH2)
--------------------------------------------------------------------------------
marg1', 'cst1', 'Mh'), 11111111, 22222222))

Navnit
ADDRTYPE(400075, 'lbs
marg', 'vadala', 'Mh')
BRANCHTABLETYPE(BRANCHTYPE(ADDRTYPE(400070, 'lbs
marg1', 'vadala1', 'Mh'), 11111111, 22222222))

NAME
--------------------
ADDRESS(PINCODE, STREET, CITY, STATE)
--------------------------------------------------------------------------------
BRANCHES(ADDRESS(PINCODE, STREET, CITY, STATE), PH1, PH2)
--------------------------------------------------------------------------------


6 rows selected.
```

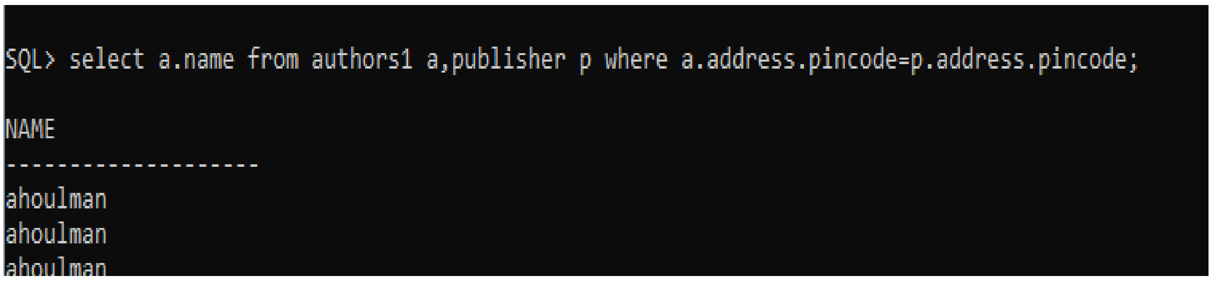


```
6 rows selected.

SQL> insert into books select 'Dw','2-jan-1989',ref(p), authorlisttype(ref(a)) from
  2  publisher p,authors1 a where p.name='Navnit' and a.name='michael';

1 row created.

SQL> insert into books select 'Dw','2-jan-1989',ref(p), authorlisttype(ref(a)) from
  2  publisher p,authors1 a where p.name='Tata Mac' and a.name='ahoulman';

1 row created.

SQL>
```



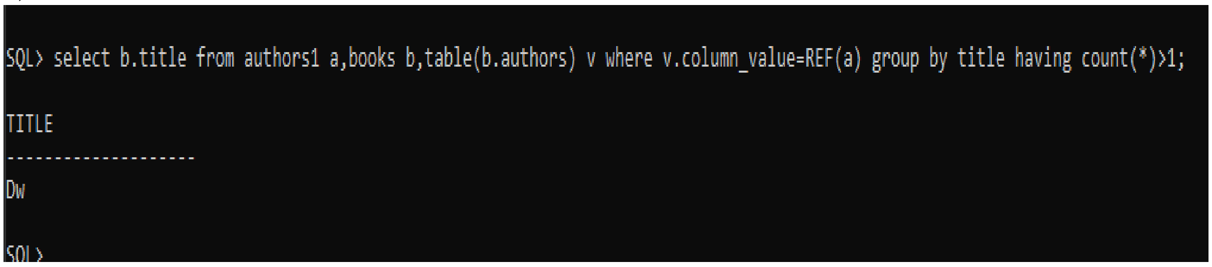
```
SQL> select * from books;

TITLE                YEAR
-------------------- ---------
PUBLISHED_BY
--------------------------------------------------------------------------------
AUTHORS
--------------------------------------------------------------------------------
Dw                   02-JAN-89
00002202083FAC1C73008C47ADAD88DCC4FBB7E8D7F067C33427CF42048560751FE0B6984F
AUTHORLISTTYPE(0000280209DBF1601BF1164212BCDD8C8C50827A52EF4DF7A8698145289DF4B29
F5ECE5D3D01C001C50006)

Dw                   02-JAN-89
00002202089C998852FA5B404A86CBE550ED9E18DCF067C33427CF42048560751FE0B6984F

TITLE                YEAR
-------------------- ---------
PUBLISHED_BY
--------------------------------------------------------------------------------
AUTHORS
--------------------------------------------------------------------------------
AUTHORLISTTYPE(0000280209D6A206F58E8947159EDCB01F2858C494EF4DF7A8698145289DF4B29
F5ECE5D3D01C001C50000)
```

a) List all of the authors that have the same pin code as their publisher.



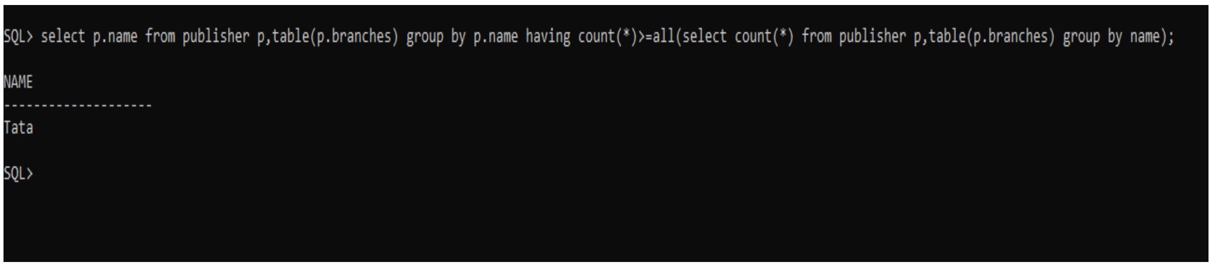
```
SQL> select a.name from authors1 a,publisher p where a.address.pincode=p.address.pincode;

NAME
--------------------
ahoulman
ahoulman
ahoulman
```

b) List all books that have 2 or more authors.



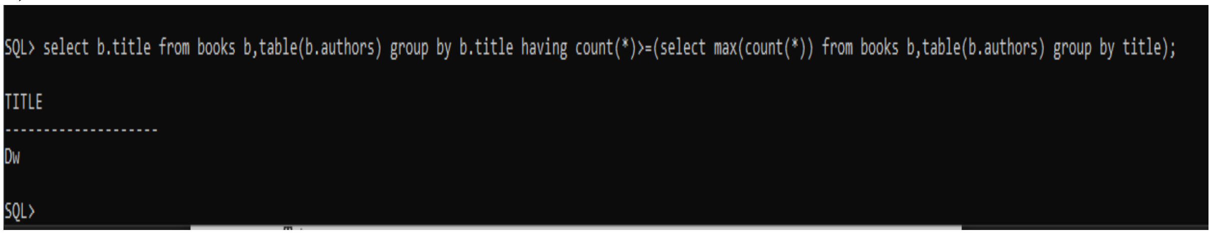
```
SQL> select b.title from authors1 a,books b,table(b.authors) v where v.column_value=REF(a) group by title having count(*)>1;

TITLE
-------------------
Dw

SQL>
```

c) List the name of the publisher that has the most branches.



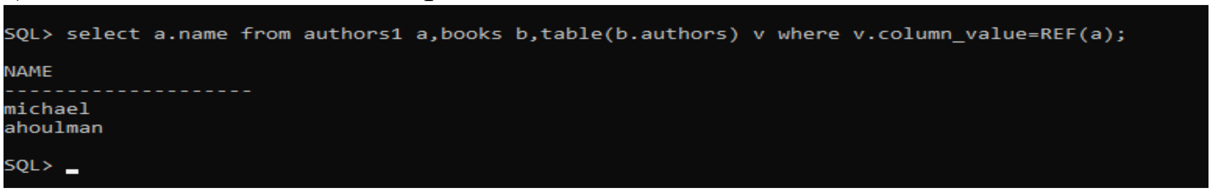
```
SQL> select p.name from publisher p,table(p.branches) group by p.name having count(*)>=all(select count(*) from publisher p,table(p.branches) group by name);

NAME
-------------------
Tata

SQL>
```

d) Name the title of the book that has the most authors.



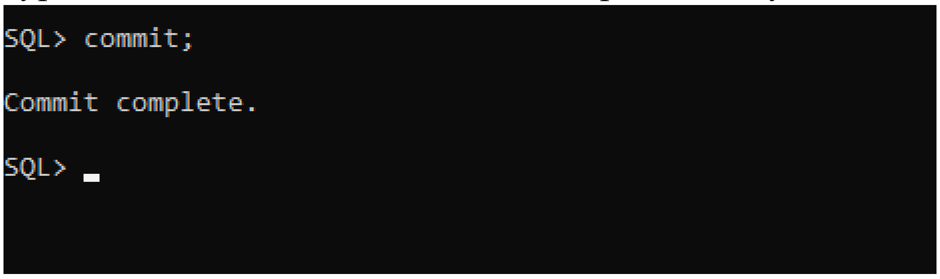
```
SQL> select b.title from books b,table(b.authors) group by b.title having count(*)>=(select max(count(*)) from books b,table(b.authors) group by title);

TITLE
-------------------
Dw

SQL>
```

e) List all authors who have published more than one book.



```
SQL> select a.name from authors1 a,books b,table(b.authors) v where v.column_value=REF(a);

NAME
--------------------
michael
ahoulman

SQL>
```

Type command **commit** as It lets a user permanently save all the changes made.



```
SQL> commit;

Commit complete.

SQL>
```

Practical 02

Temporal Database:

Temporal databases, in the broadest sense, encompass all database applications that require some aspect of time when organizing their information. Hence, they provide a good example to illustrate the need for developing a set of unifying concepts for application developers to use.

Time Representation, Calendars and Time Dimensions

For temporal databases, time is considered to be an ordered sequence of points in some granularity that is determined by the application.

The temporal data types include:

DATE (specifying Year, Month, and Day as YYYY-MM-DD)

TIME (specifying Hour, Minute, and Second as HH: MM: SS)

TIMESTAMP (specifying a Date/Time combination, with options for including

sub second divisions if they are needed)

INTERVAL (a relative time duration, such as 10 days or 250 minutes). PERIOD (an anchored time duration with a fixed starting point, such as the 10-day period from January 1, 2009, to January 10,2009, inclusive).

Types of Temporal Database

- Uni-Temporal

A uni-temporal database has one axis of time, either the validity range or the system time range.

- Bi-Temporal

A bi-temporal database has two axes of time, valid time and transaction time.

- Tri-Temporal

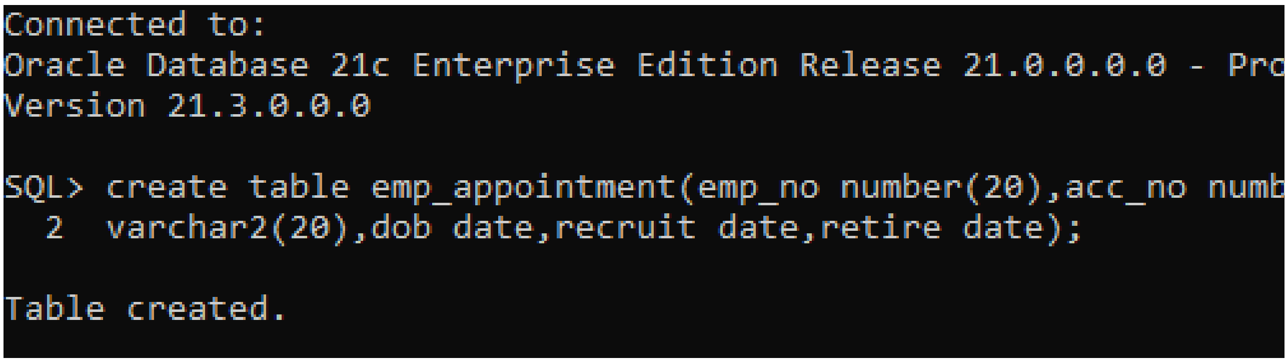
A tri-temporal database has three axes of time valid time, transaction time and decision time.

A.) Aim: Create a table emp_appointment, which stores the account number, name, dob and valid time say, (Recruitment date and Retirement date). Insert 10 records. Also create the trigger to calculate retirement date.

Execute following queries:

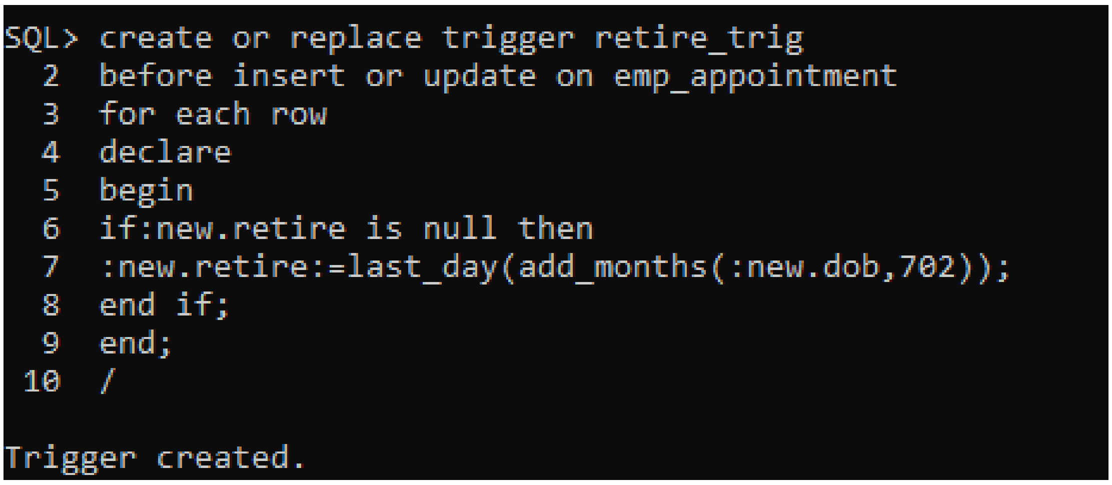
• Find all the employee who join the company on 24-oct-2000.
• Find all employee who join the company on 31-mar-2024.

Create Table



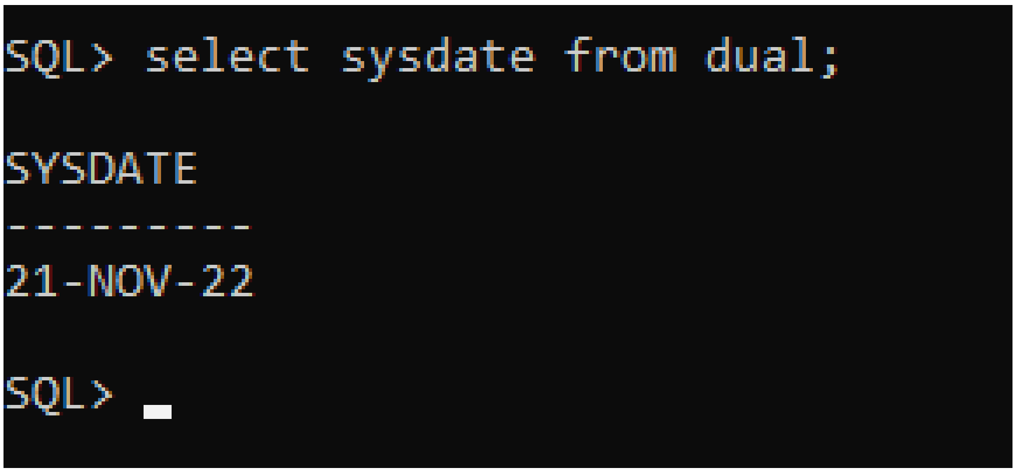
```
Connected to:
Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Pro
Version 21.3.0.0.0

SQL> create table emp_appointment(emp_no number(20),acc_no numb
  2  varchar2(20),dob date,recruit date,retire date);

Table created.
```

Create Trigger



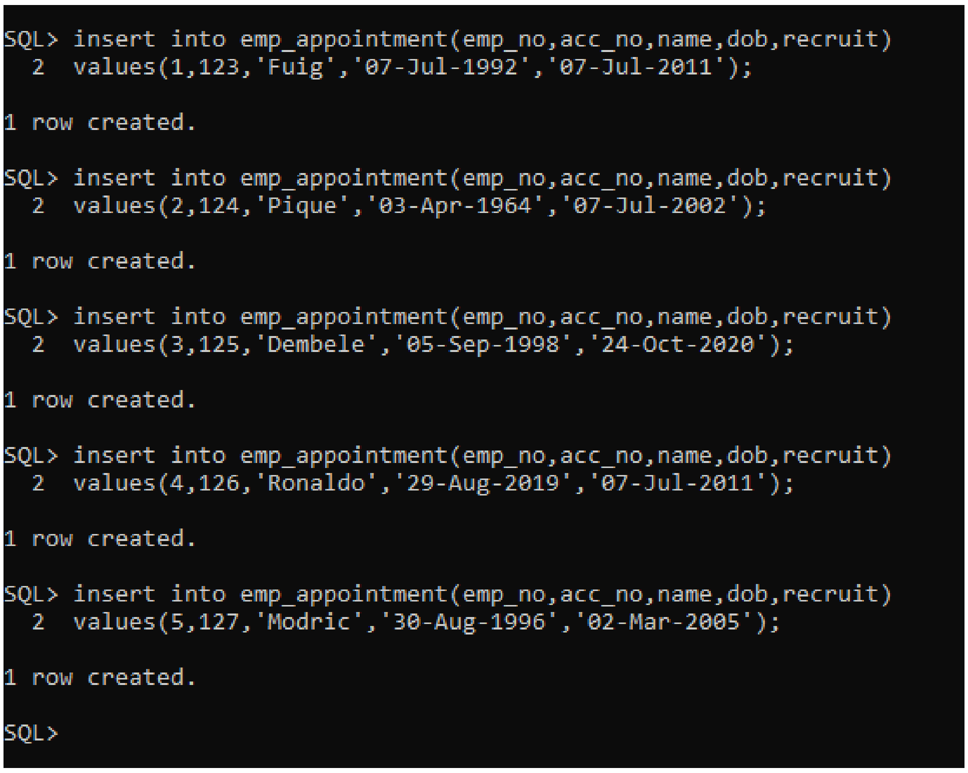
```
SQL> create or replace trigger retire_trig
  2  before insert or update on emp_appointment
  3  for each row
  4  declare
  5  begin
  6  if:new.retire is null then
  7  :new.retire:=last_day(add_months(:new.dob,702));
  8  end if;
  9  end;
 10  /

Trigger created.
```

Check the date format.

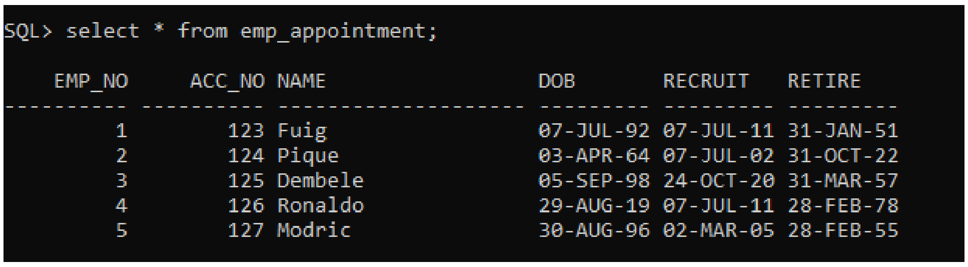


```
SQL> select sysdate from dual;

SYSDATE
---------
21-NOV-22

SQL> _
```

Insert Values

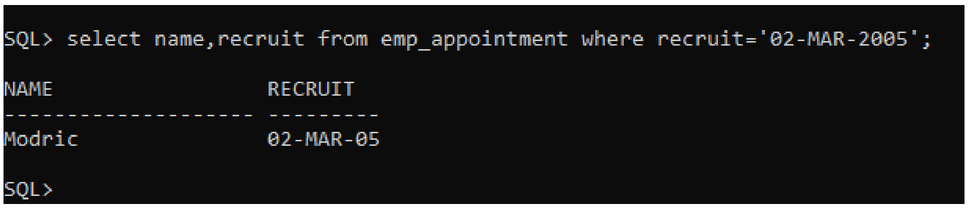


```
SQL> insert into emp_appointment(emp_no,acc_no,name,dob,recruit)
  2  values(1,123,'Fuig','07-Jul-1992','07-Jul-2011');

1 row created.

SQL> insert into emp_appointment(emp_no,acc_no,name,dob,recruit)
  2  values(2,124,'Pique','03-Apr-1964','07-Jul-2002');

1 row created.

SQL> insert into emp_appointment(emp_no,acc_no,name,dob,recruit)
  2  values(3,125,'Dembele','05-Sep-1998','24-Oct-2020');

1 row created.

SQL> insert into emp_appointment(emp_no,acc_no,name,dob,recruit)
  2  values(4,126,'Ronaldo','29-Aug-2019','07-Jul-2011');

1 row created.

SQL> insert into emp_appointment(emp_no,acc_no,name,dob,recruit)
  2  values(5,127,'Modric','30-Aug-1996','02-Mar-2005');

1 row created.

SQL>
```



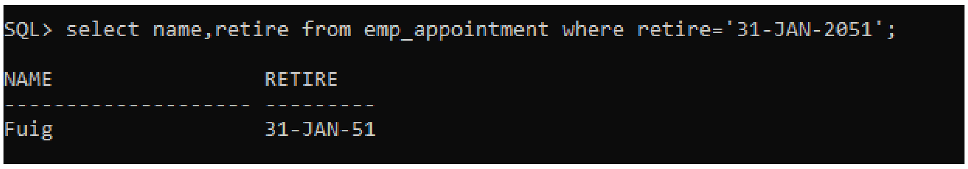
```
SQL> select * from emp_appointment;

    EMP_NO     ACC_NO NAME                 DOB       RECRUIT   RETIRE
---------- ---------- -------------------- --------- --------- ---------
         1        123 Fuig                 07-JUL-92 07-JUL-11 31-JAN-51
         2        124 Pique                03-APR-64 07-JUL-02 31-OCT-22
         3        125 Dembele              05-SEP-98 24-OCT-20 31-MAR-57
         4        126 Ronaldo              29-AUG-19 07-JUL-11 28-FEB-78
         5        127 Modric               30-AUG-96 02-MAR-05 28-FEB-55
```

a) Find all the employee who join the company on 2-mar-2005.



```
SQL> select name,recruit from emp_appointment where recruit='02-MAR-2005';

NAME                 RECRUIT
-------------------- ---------
Modric               02-MAR-05

SQL>
```

b) Find all the employee who will retire on 31-Jan-2051.



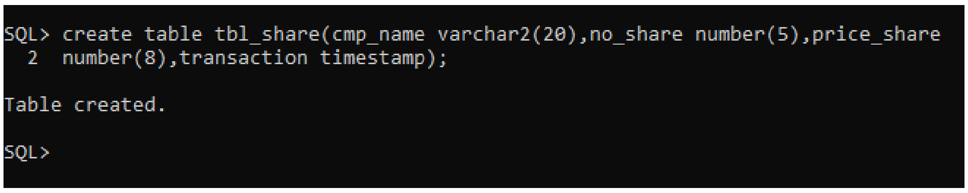
```
SQL> select name,retire from emp_appointment where retire='31-JAN-2051';

NAME                 RETIRE
-------------------- ---------
Fuig                 31-JAN-51
```

B) Aim: Create a table tbl_share, which stores the, name of company, number of shares, and price per share at transaction time. Insert 10 records.

Execute following queries:

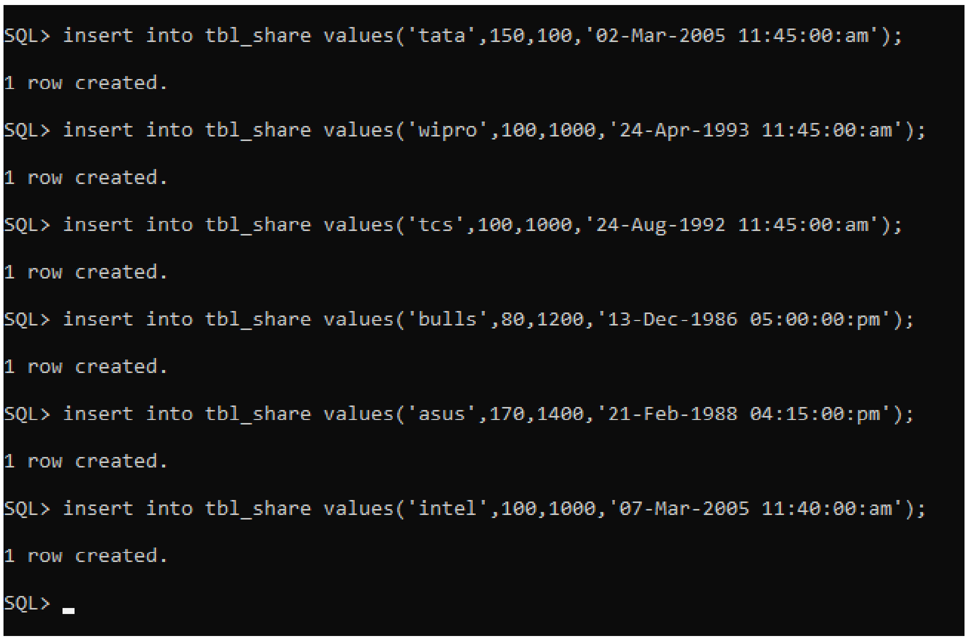
- Find all the names of company whose share is more than Rs.100 at 11:45:00 AM.
- Find the name of the company which has the highest share price at 11:45:00 AM.

Create Table

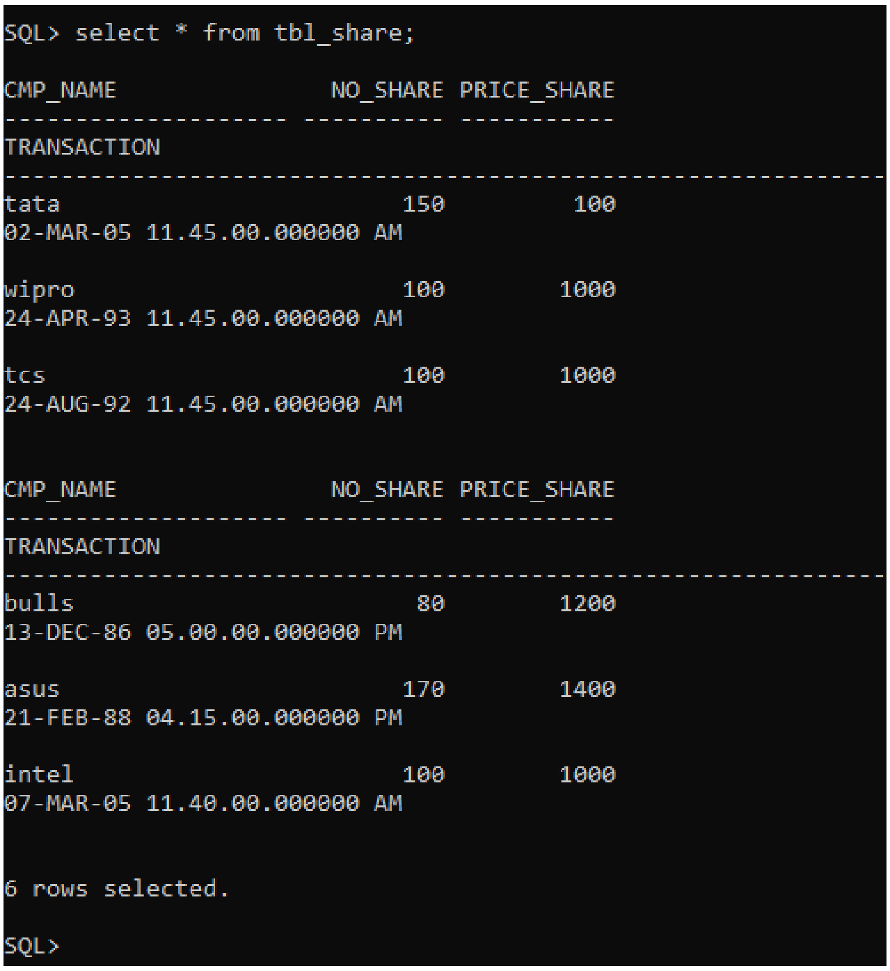


```
SQL> create table tbl_share(cmp_name varchar2(20),no_share number(5),price_share
  2  number(8),transaction timestamp);

Table created.

SQL>
```

Insert Values



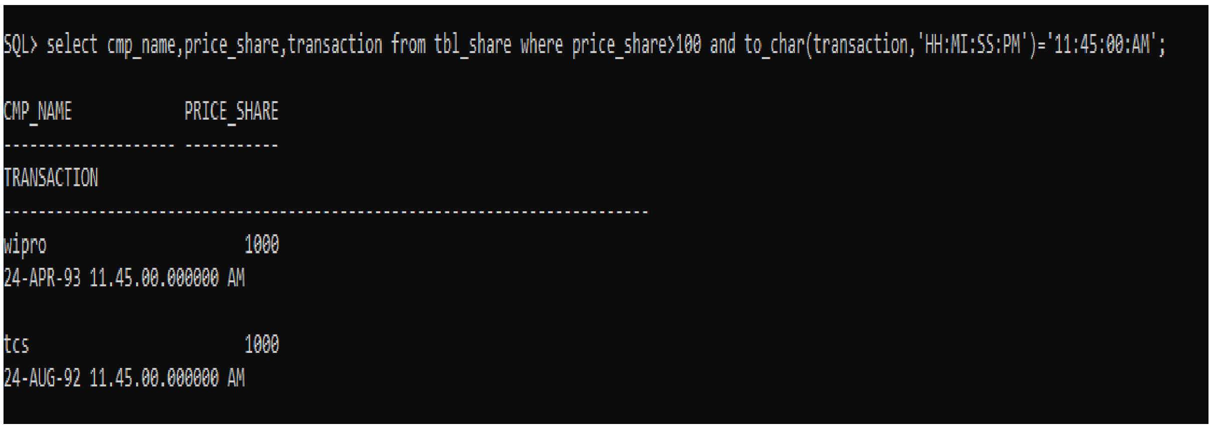
```
SQL> insert into tbl_share values('tata',150,100,'02-Mar-2005 11:45:00:am');

1 row created.

SQL> insert into tbl_share values('wipro',100,1000,'24-Apr-1993 11:45:00:am');

1 row created.

SQL> insert into tbl_share values('tcs',100,1000,'24-Aug-1992 11:45:00:am');

1 row created.

SQL> insert into tbl_share values('bulls',80,1200,'13-Dec-1986 05:00:00:pm');

1 row created.

SQL> insert into tbl_share values('asus',170,1400,'21-Feb-1988 04:15:00:pm');

1 row created.

SQL> insert into tbl_share values('intel',100,1000,'07-Mar-2005 11:40:00:am');

1 row created.

SQL> 
```



```
SQL> select * from tbl_share;

CMP_NAME               NO_SHARE PRICE_SHARE
-------------------- ---------- -----------
TRANSACTION
---------------------------------------------------------------
tata                        150         100
02-MAR-05 11.45.00.000000 AM

wipro                       100        1000
24-APR-93 11.45.00.000000 AM

tcs                         100        1000
24-AUG-92 11.45.00.000000 AM


CMP_NAME               NO_SHARE PRICE_SHARE
-------------------- ---------- -----------
TRANSACTION
---------------------------------------------------------------
bulls                        80        1200
13-DEC-86 05.00.00.000000 PM

asus                        170        1400
21-FEB-88 04.15.00.000000 PM

intel                       100        1000
07-MAR-05 11.40.00.000000 AM


6 rows selected.

SQL>
```

a) Find all the names of a company whose share price is more than Rs. 100 at 11:45 AM

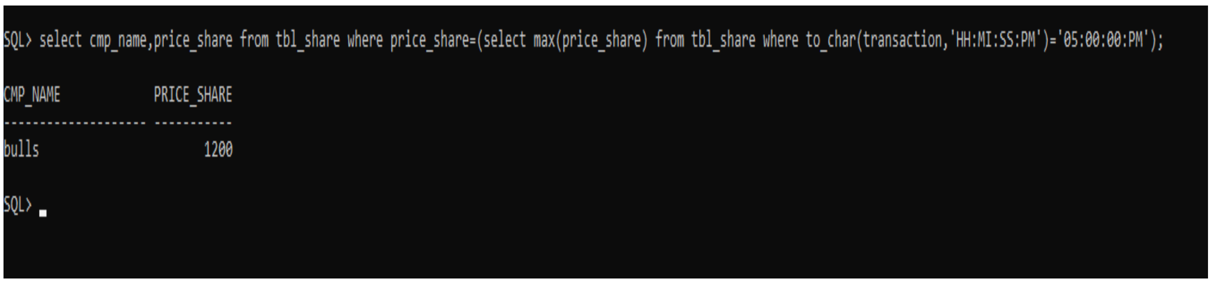
SQL> select cmp_name,price_share,transaction from tbl_share where price_share>100 and to_char(transaction,'HH:MI:SS:PM')='11:45:00:AM';



```
SQL> select cmp_name,price_share,transaction from tbl_share where price_share>100 and to_char(transaction,'HH:MI:SS:PM')='11:45:00:AM';

CMP_NAME             PRICE_SHARE
-------------------- ----------
TRANSACTION
---------------------------------------------------------------------------
wipro                      1000
24-APR-93 11.45.00.000000 AM

tcs                        1000
24-AUG-92 11.45.00.000000 AM
```

b) Find the Name of the company which has highest share price at 5:00PM

SQL> select cmp_name, price_share from tbl_share where price_share=(select max(price_share) from tbl_share where to_char(transaction,'HH:MI:SS:PM')='05:00:00:PM');



```
SQL> select cmp_name,price_share from tbl_share where price_share=(select max(price_share) from tbl_share where to_char(transaction,'HH:MI:SS:PM')='05:00:00:PM');

CMP_NAME             PRICE_SHARE
-------------------- -----------
bulls                       1200

SQL>
```

Practical 3

Part A

AIM: Create a table employee having dept_id as number data type and employee_spec as XML datatype(XML_Type). The employee_spec is a schema with attributes emp_id, name, email, acc_no, managerEmail, dataOf Joining. Insert 10 tuples into the employee table. Fire the following queries on the XML database.

What is XML Database?

XML Database is used to store huge amounts of information in the XML format. As the use of XML is increasing in every field, it is required to have a secured place to store the XML documents. The data stored in the database can be queried using XQuery, serialised, and exported into a desired format.

XML Database
Types

There are two major types of **XML** databases

XML-enabled

Native XML (NXD)

XML Enabled Database

XML enabled database is nothing but the extension provided for the conversion of XML documents. This is a relational database, where data is stored in tables consisting of rows and columns. The tables contain a set of records, which in turn consist of fields.

Native Xml Database

Native XML database is based on the container rather than table format. It can store large amounts of XML documents and data. Native XML database is queried by the XPath-expressions.

Native XML database has an advantage over the XML-enabled database. It is highly capable to store, query and maintain the XML document than XML-enabled database.

Queries

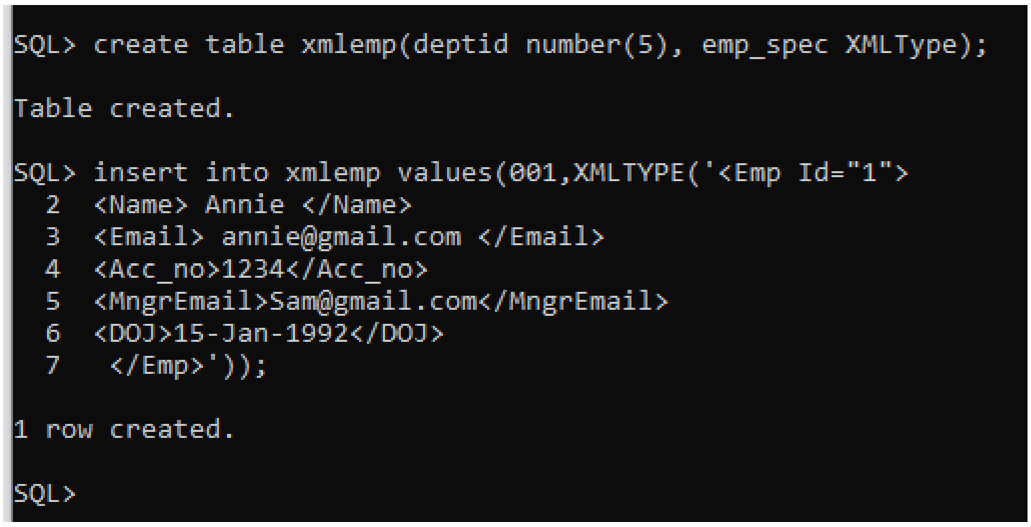
a) Retrieve the names of employee.

b) Retrieve the acc_no of employees.

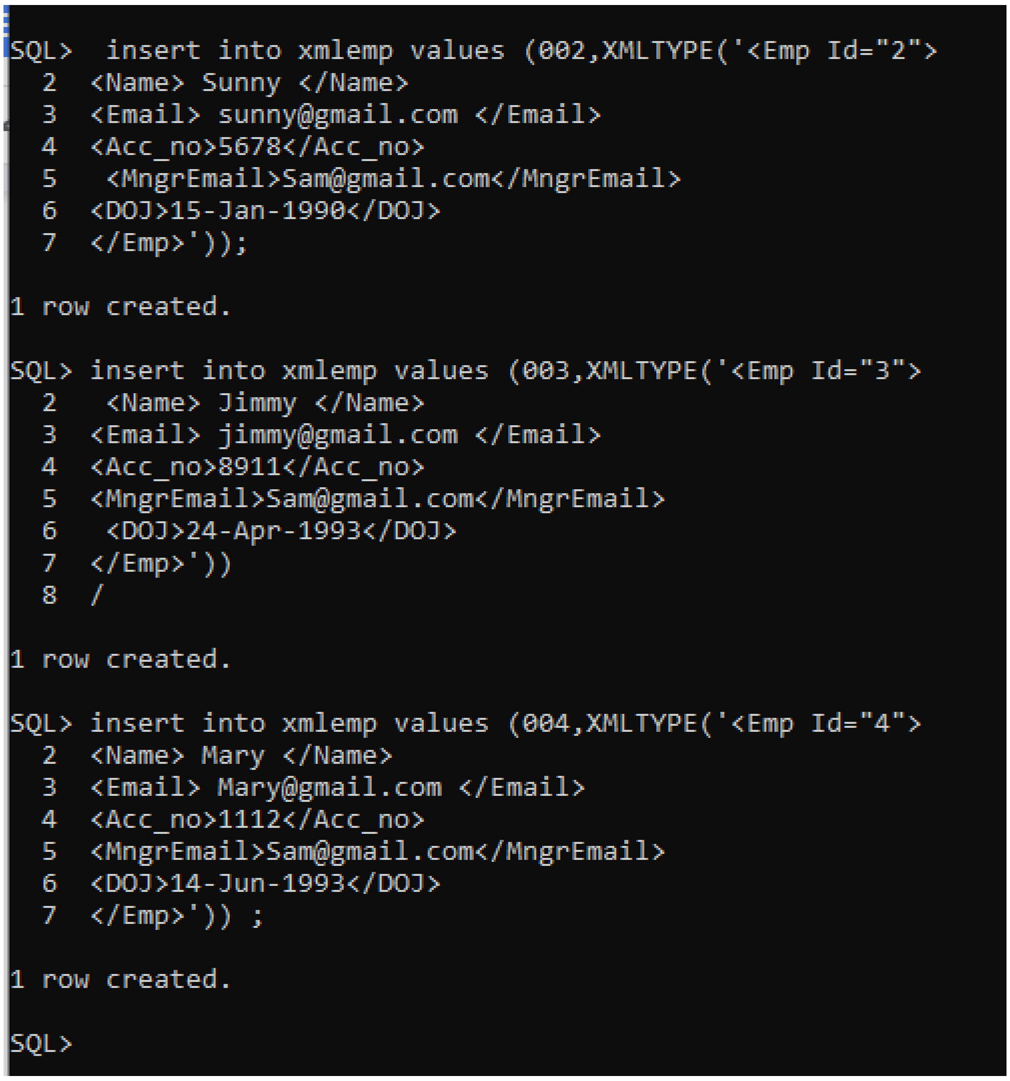
c) Retrieve the names, acc_no, email of employees.

d) Update the 3rd from the table and display the name of an employee.

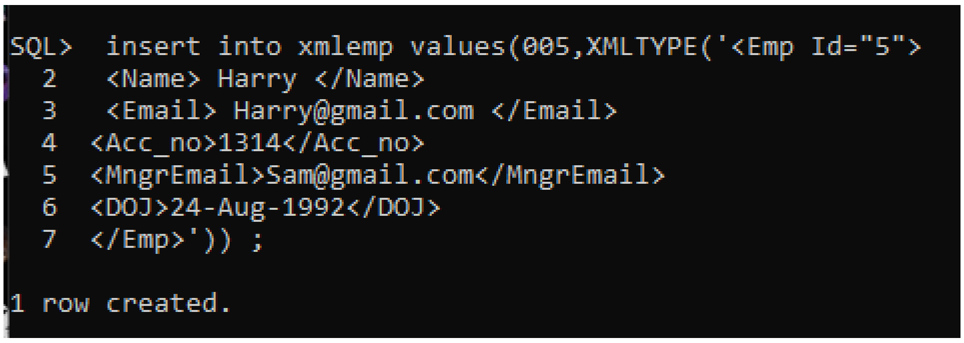
e) Delete 4th record from the table.



```
SQL> create table xmlemp(deptid number(5), emp_spec XMLType);

Table created.

SQL> insert into xmlemp values(001,XMLTYPE('<Emp Id="1">
  2  <Name> Annie </Name>
  3  <Email> annie@gmail.com </Email>
  4  <Acc_no>1234</Acc_no>
  5  <MngrEmail>Sam@gmail.com</MngrEmail>
  6  <DOJ>15-Jan-1992</DOJ>
  7   </Emp>'));

1 row created.

SQL>
```

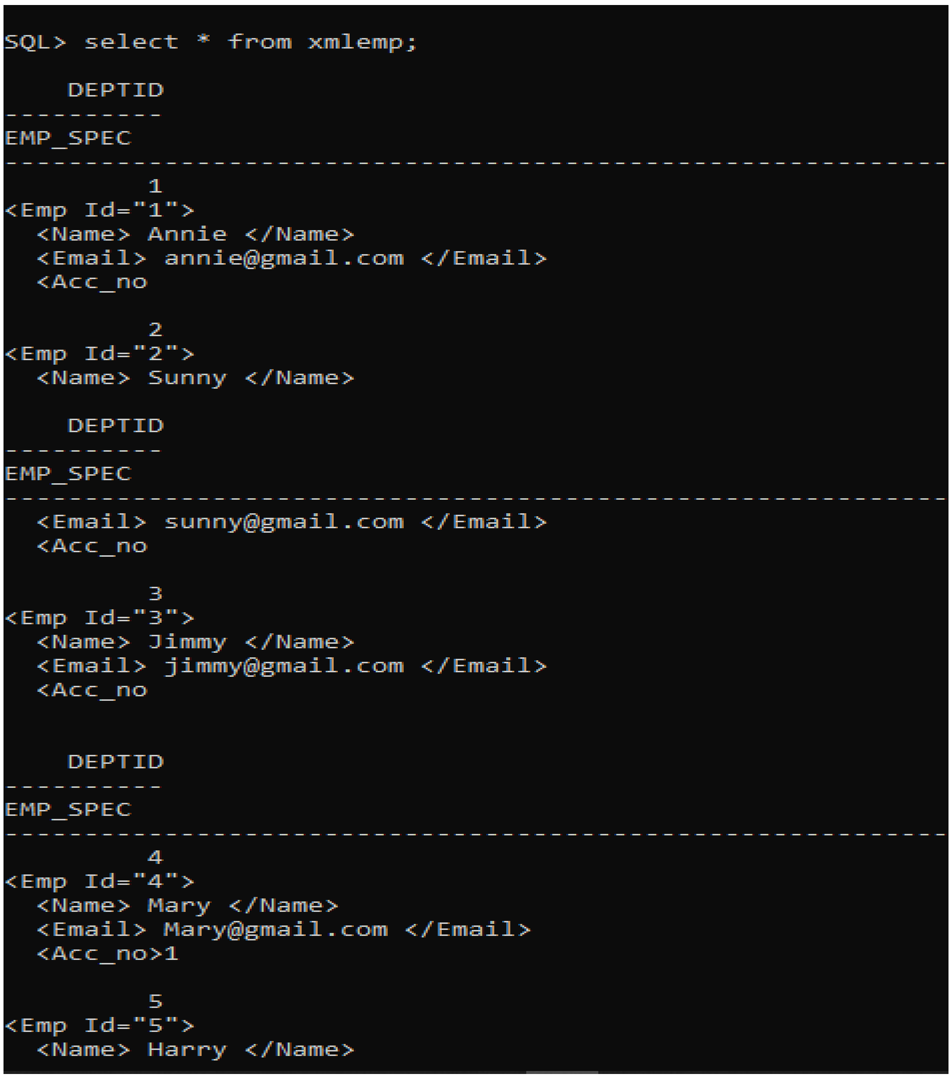


```
SQL>  insert into xmlemp values (002,XMLTYPE('<Emp Id="2">
  2  <Name> Sunny </Name>
  3  <Email> sunny@gmail.com </Email>
  4  <Acc_no>5678</Acc_no>
  5   <MngrEmail>Sam@gmail.com</MngrEmail>
  6  <DOJ>15-Jan-1990</DOJ>
  7  </Emp>'));

1 row created.

SQL> insert into xmlemp values (003,XMLTYPE('<Emp Id="3">
  2   <Name> Jimmy </Name>
  3  <Email> jimmy@gmail.com </Email>
  4  <Acc_no>8911</Acc_no>
  5  <MngrEmail>Sam@gmail.com</MngrEmail>
  6   <DOJ>24-Apr-1993</DOJ>
  7  </Emp>'))
  8  /

1 row created.

SQL> insert into xmlemp values (004,XMLTYPE('<Emp Id="4">
  2  <Name> Mary </Name>
  3  <Email> Mary@gmail.com </Email>
  4  <Acc_no>1112</Acc_no>
  5  <MngrEmail>Sam@gmail.com</MngrEmail>
  6  <DOJ>14-Jun-1993</DOJ>
  7  </Emp>')) ;

1 row created.

SQL>
```

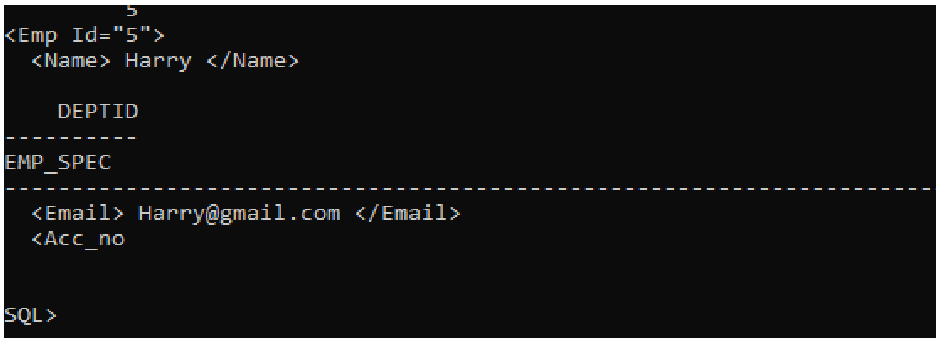


```
SQL>  insert into xmlemp values(005,XMLTYPE('<Emp Id="5">
      <Name> Harry </Name>
      <Email> Harry@gmail.com </Email>
     <Acc_no>1314</Acc_no>
     <MngrEmail>Sam@gmail.com</MngrEmail>
     <DOJ>24-Aug-1992</DOJ>
     </Emp>')) ;

1 row created.
```

select * from xmlemp;

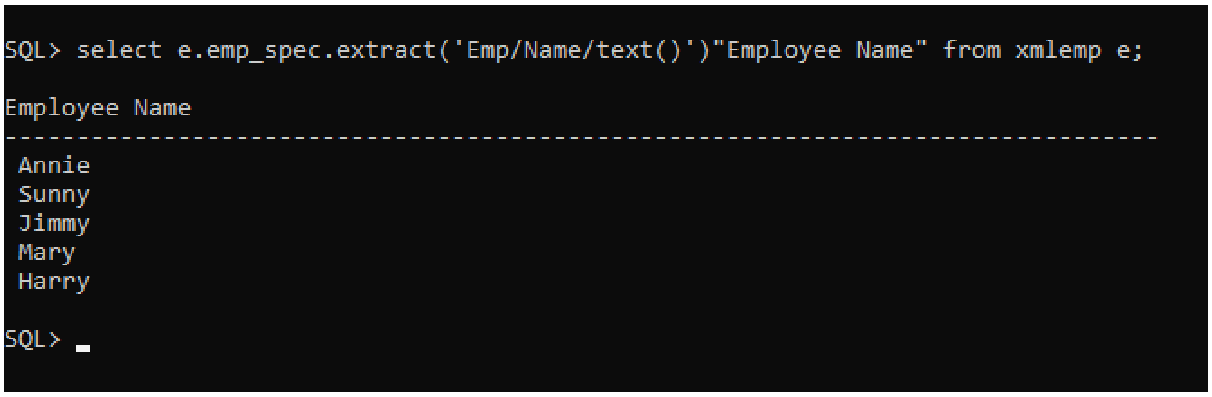


```
SQL> select * from xmlemp;

    DEPTID
----------
EMP_SPEC
------------------------------------------------------------
         1
<Emp Id="1">
  <Name> Annie </Name>
  <Email> annie@gmail.com </Email>
  <Acc_no

         2
<Emp Id="2">
  <Name> Sunny </Name>

    DEPTID
----------
EMP_SPEC
------------------------------------------------------------
  <Email> sunny@gmail.com </Email>
  <Acc_no

         3
<Emp Id="3">
  <Name> Jimmy </Name>
  <Email> jimmy@gmail.com </Email>
  <Acc_no


    DEPTID
----------
EMP_SPEC
------------------------------------------------------------
         4
<Emp Id="4">
  <Name> Mary </Name>
  <Email> Mary@gmail.com </Email>
  <Acc_no>1

         5
<Emp Id="5">
  <Name> Harry </Name>
```



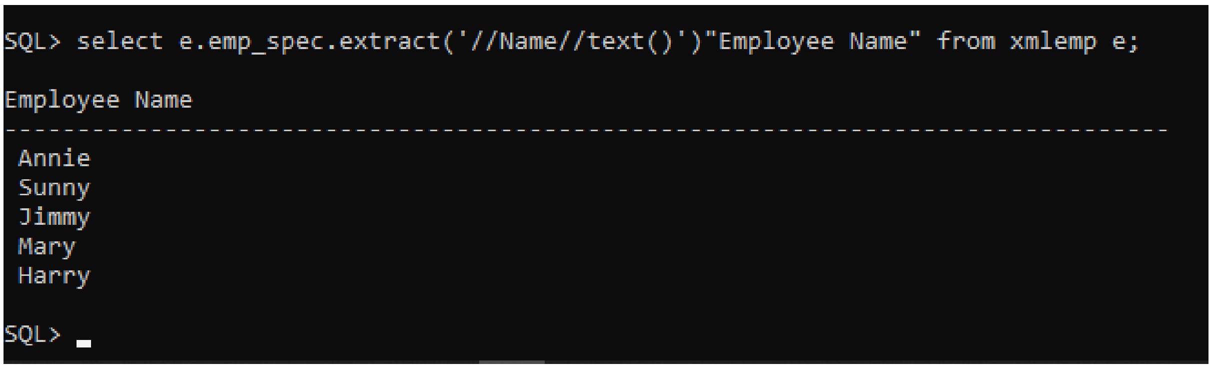
```
         5
<Emp Id="5">
  <Name> Harry </Name>

    DEPTID
----------
EMP_SPEC
------------------------------------------------------------
  <Email> Harry@gmail.com </Email>
  <Acc_no


SQL>
```

a) Retrieve the names of employee with single slash.



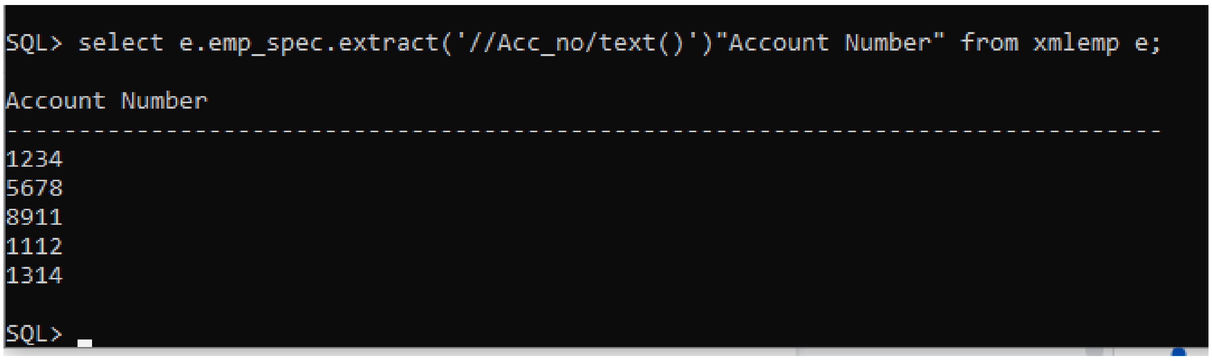
```
SQL> select e.emp_spec.extract('Emp/Name/text()')"Employee Name" from xmlemp e;

Employee Name
--------------------------------------------------------------------------------
 Annie
 Sunny
 Jimmy
 Mary
 Harry

SQL>
```

b) Retrieve the names of employee with double slash.



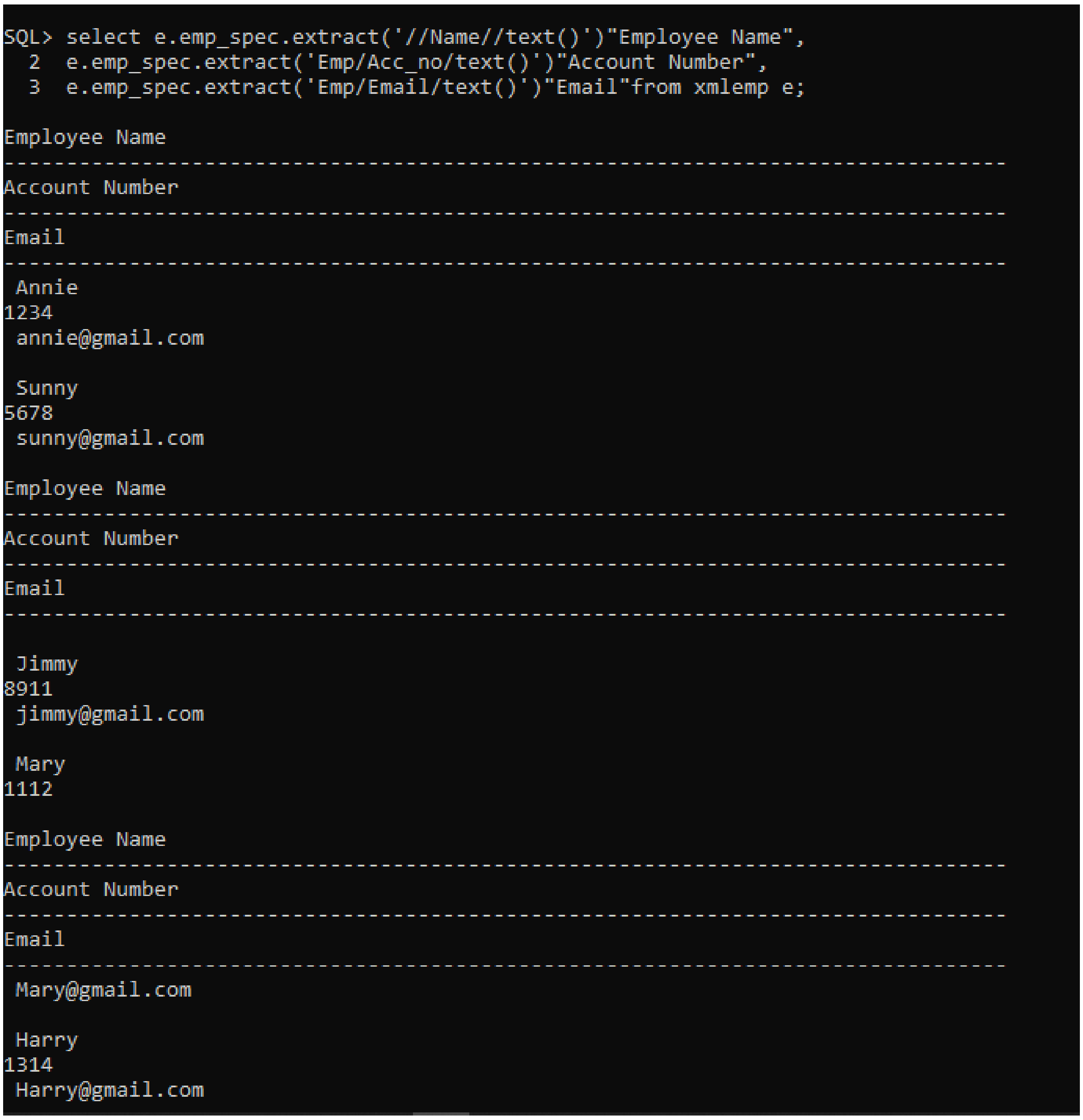
```
SQL> select e.emp_spec.extract('//Name//text()')"Employee Name" from xmlemp e;

Employee Name
--------------------------------------------------------------------------------
 Annie
 Sunny
 Jimmy
 Mary
 Harry

SQL>
```

c) Retrieve the acc_no of employees.



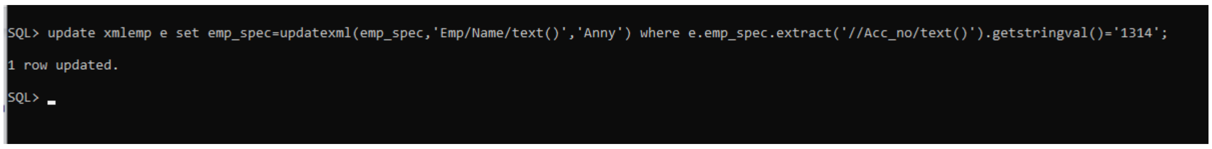
```
SQL> select e.emp_spec.extract('//Acc_no/text()')"Account Number" from xmlemp e;

Account Number
--------------------------------------------------------------------------------
1234
5678
8911
1112
1314

SQL>
```

d) Retrieve the names, acc_no, email of employees.



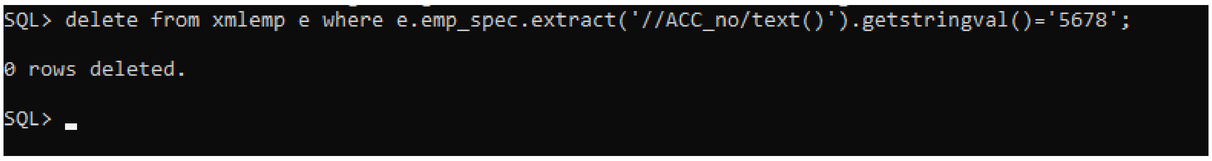
```
SQL> select e.emp_spec.extract('//Name//text()')"Employee Name",
  2  e.emp_spec.extract('Emp/Acc_no/text()')"Account Number",
  3  e.emp_spec.extract('Emp/Email/text()')"Email"from xmlemp e;

Employee Name
--------------------------------------------------------------------------------
Account Number
--------------------------------------------------------------------------------
Email
--------------------------------------------------------------------------------
 Annie
1234
 annie@gmail.com

 Sunny
5678
 sunny@gmail.com

Employee Name
--------------------------------------------------------------------------------
Account Number
--------------------------------------------------------------------------------
Email
--------------------------------------------------------------------------------

 Jimmy
8911
 jimmy@gmail.com

 Mary
1112

Employee Name
--------------------------------------------------------------------------------
Account Number
--------------------------------------------------------------------------------
Email
--------------------------------------------------------------------------------
 Mary@gmail.com

 Harry
1314
 Harry@gmail.com
```

e) Update the 3rd record from the table and display the name of an employee.



```
SQL> update xmlemp e set emp_spec=updatexml(emp_spec,'Emp/Name/text()','Anny') where e.emp_spec.extract('//Acc_no/text()').getstringval()='1314';

1 row updated.

SQL>
```

f) Delete 4th record from the table.

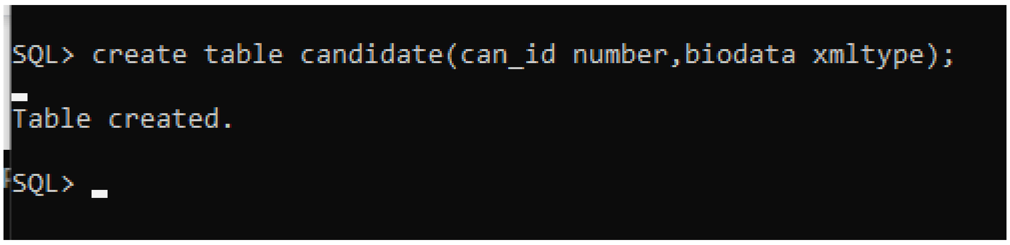


```
SQL> delete from xmlemp e where e.emp_spec.extract('//ACC_no/text()').getstringval()='5678';

0 rows deleted.

SQL>
```

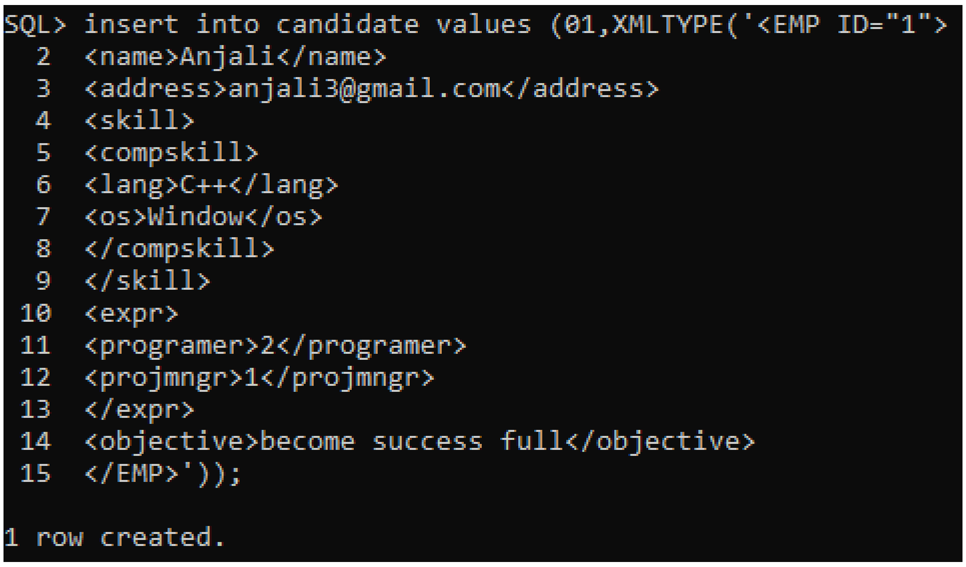
Part B

Create a table candidate having cand_id as varchar2 datatype and biodata as XML datatype (XML type). The biodata is a schema with attributes
Name, address, skill - compskill - 1) language 2) networking, expr - 1) prog 2) prjmgr, objectives.
Fire the following queries on XML database

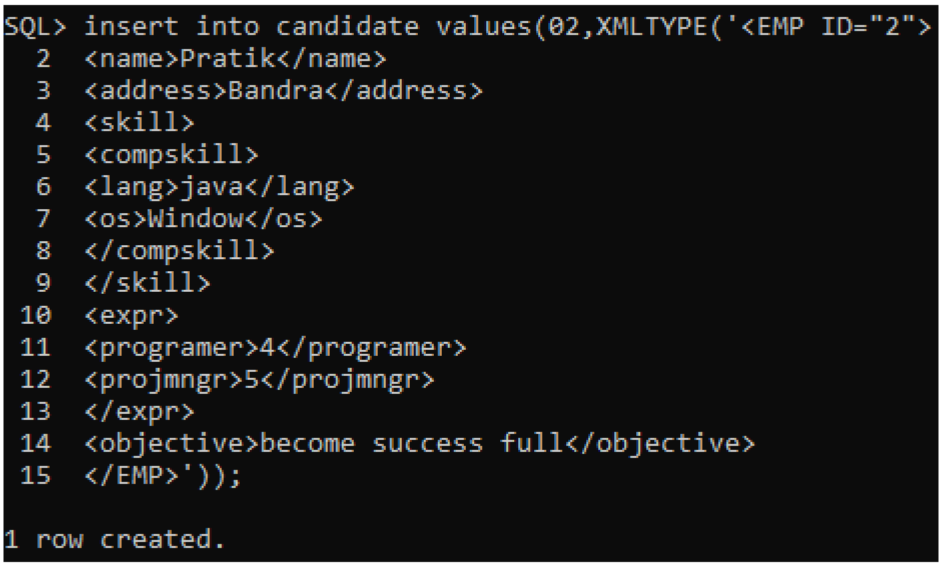
a) Display candidate name who is good in java and having experience more than 5 years
b) Display candidate having project manager level experience
c) Display name and skill of all candidates
d) Delete record for address = Worli
e) Update experience of a particular candidate



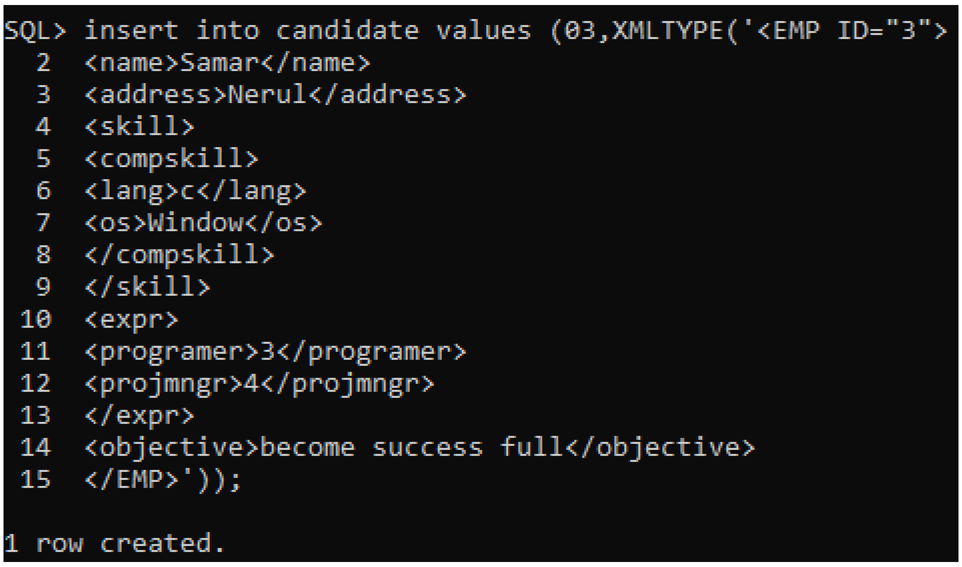
```
SQL> create table candidate(can_id number,biodata xmltype);

Table created.

SQL> _
```



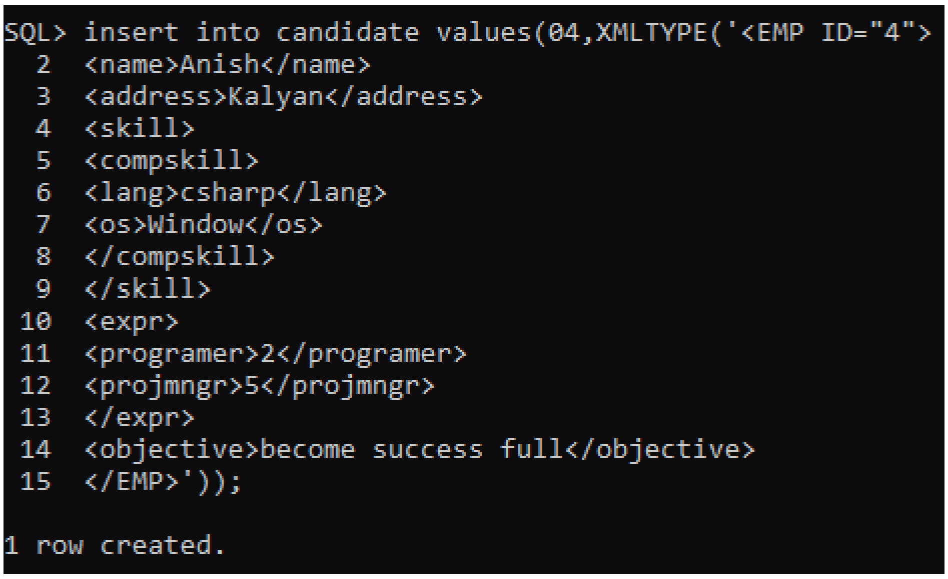
```
SQL> insert into candidate values (01,XMLTYPE('<EMP ID="1">
  2  <name>Anjali</name>
  3  <address>anjali3@gmail.com</address>
  4  <skill>
  5  <compskill>
  6  <lang>C++</lang>
  7  <os>Window</os>
  8  </compskill>
  9  </skill>
 10  <expr>
 11  <programer>2</programer>
 12  <projmngr>1</projmngr>
 13  </expr>
 14  <objective>become success full</objective>
 15  </EMP>'));

1 row created.
```



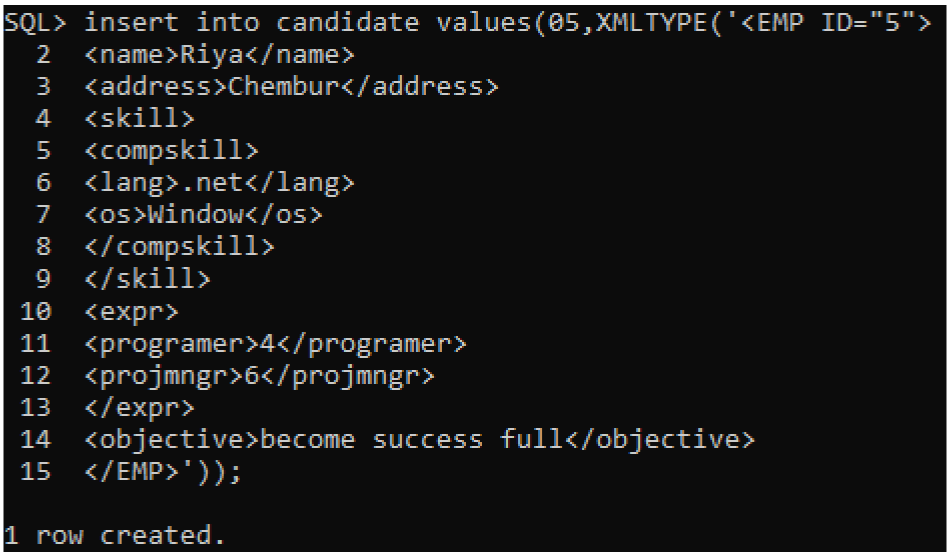
```
SQL> insert into candidate values(02,XMLTYPE('<EMP ID="2">
  2  <name>Pratik</name>
  3  <address>Bandra</address>
  4  <skill>
  5  <compskill>
  6  <lang>java</lang>
  7  <os>Window</os>
  8  </compskill>
  9  </skill>
 10  <expr>
 11  <programer>4</programer>
 12  <projmngr>5</projmngr>
 13  </expr>
 14  <objective>become success full</objective>
 15  </EMP>'));

1 row created.
```



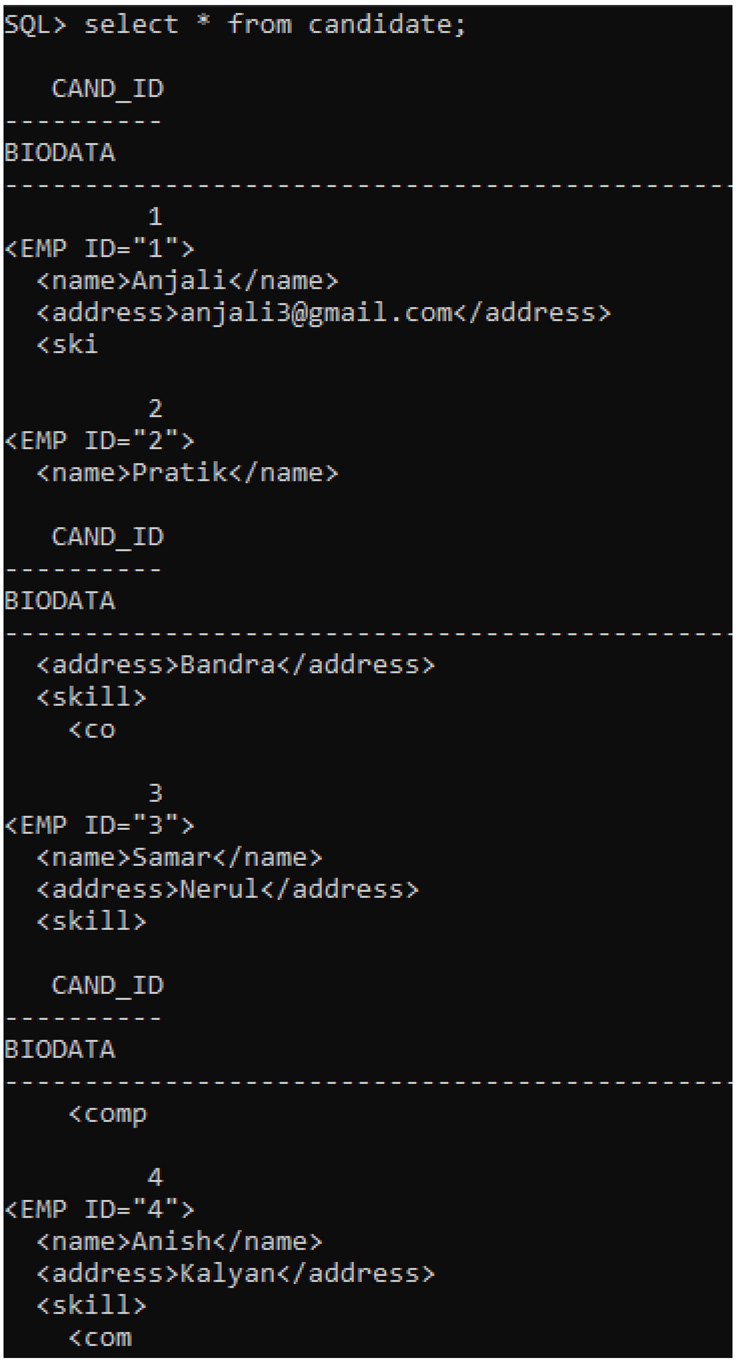
```
SQL> insert into candidate values (03,XMLTYPE('<EMP ID="3">
<name>Samar</name>
<address>Nerul</address>
<skill>
<compskill>
<lang>c</lang>
<os>Window</os>
</compskill>
</skill>
<expr>
<programer>3</programer>
<projmngr>4</projmngr>
</expr>
<objective>become success full</objective>
</EMP>'));

1 row created.
```



```
SQL> insert into candidate values(04,XMLTYPE('<EMP ID="4">
<name>Anish</name>
<address>Kalyan</address>
<skill>
<compskill>
<lang>csharp</lang>
<os>Window</os>
</compskill>
</skill>
<expr>
<programer>2</programer>
<projmngr>5</projmngr>
</expr>
<objective>become success full</objective>
</EMP>'));

1 row created.
```

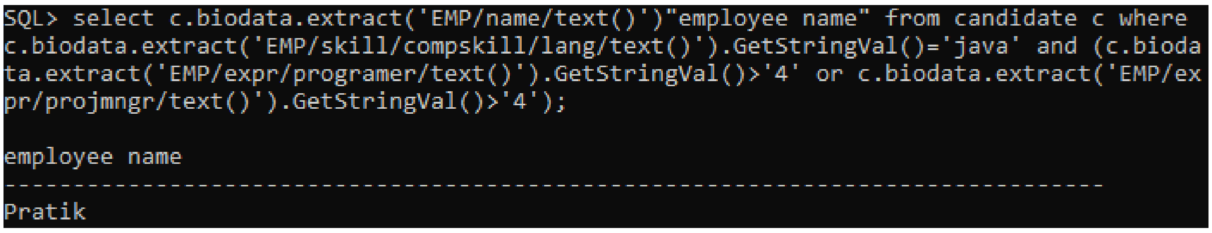


```
SQL> insert into candidate values(05,XMLTYPE('<EMP ID="5">
<name>Riya</name>
<address>Chembur</address>
<skill>
<compskill>
<lang>.net</lang>
<os>Window</os>
</compskill>
</skill>
<expr>
<programer>4</programer>
<projmngr>6</projmngr>
</expr>
<objective>become success full</objective>
</EMP>'));

1 row created.
```



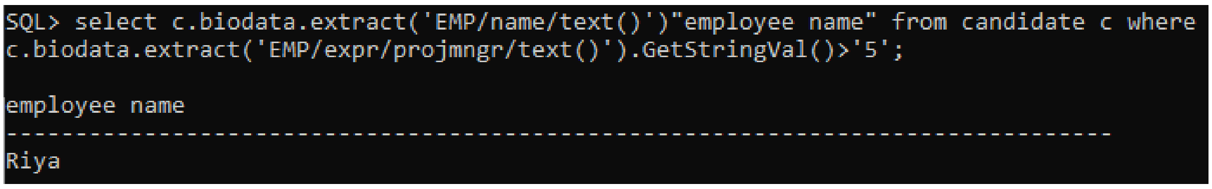
```
SQL> select * from candidate;

   CAND_ID
----------
BIODATA
----------------------------------------------
         1
<EMP ID="1">
  <name>Anjali</name>
  <address>anjali3@gmail.com</address>
  <ski

         2
<EMP ID="2">
  <name>Pratik</name>

   CAND_ID
----------
BIODATA
----------------------------------------------
  <address>Bandra</address>
  <skill>
    <co

         3
<EMP ID="3">
  <name>Samar</name>
  <address>Nerul</address>
  <skill>

   CAND_ID
----------
BIODATA
----------------------------------------------
    <comp

         4
<EMP ID="4">
  <name>Anish</name>
  <address>Kalyan</address>
  <skill>
    <com
```

a) Display candidate name who is good in java and having experience more than 4 years.



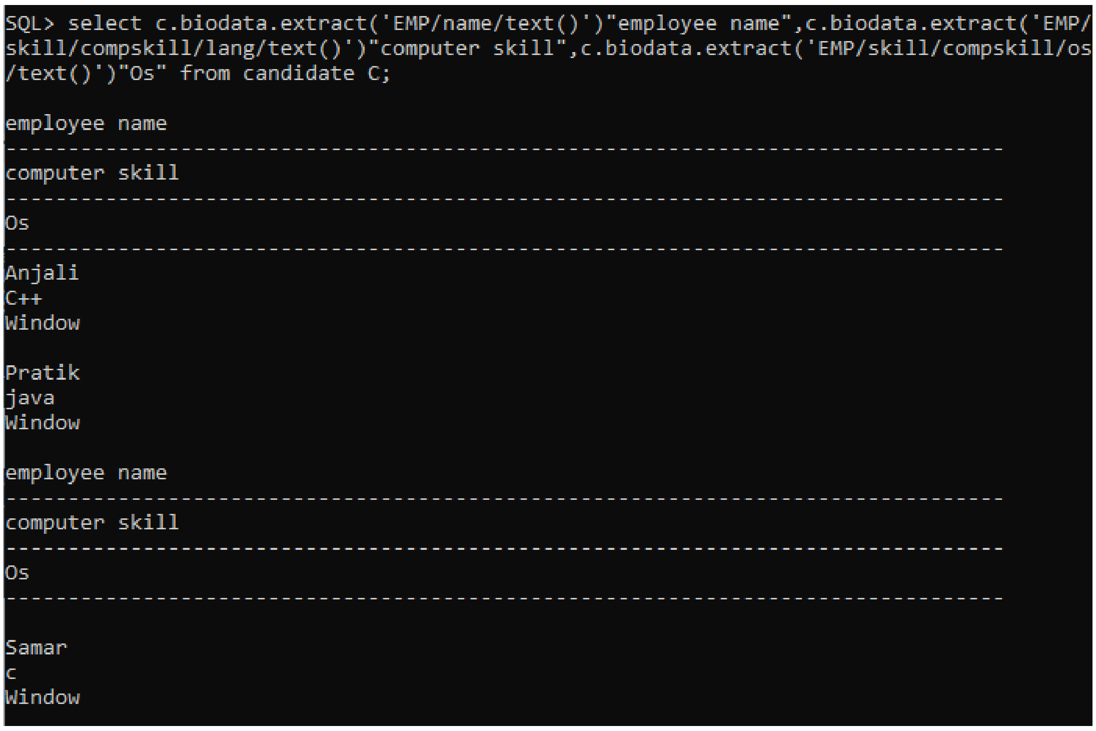
```
SQL> select c.biodata.extract('EMP/name/text()')"employee name" from candidate c where
c.biodata.extract('EMP/skill/compskill/lang/text()').GetStringVal()='java' and (c.bioda
ta.extract('EMP/expr/programer/text()').GetStringVal()>'4' or c.biodata.extract('EMP/ex
pr/projmngr/text()').GetStringVal()>'4');

employee name
--------------------------------------------------------------------------------
Pratik
```

b) Display candidate having project manager level **experience**

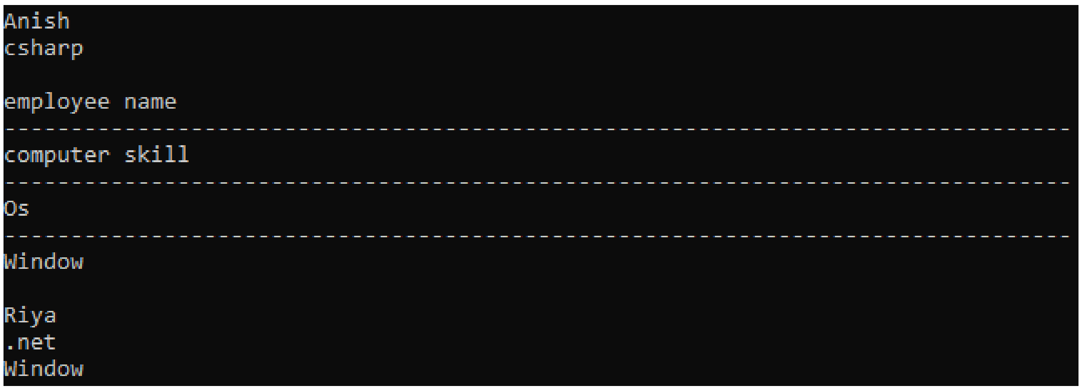


```
SQL> select c.biodata.extract('EMP/name/text()')"employee name" from candidate c where
c.biodata.extract('EMP/expr/projmngr/text()').GetStringVal()>'5';

employee name
--------------------------------------------------------------------------------
Riya
```

c) Display name and skill **of all candidates**

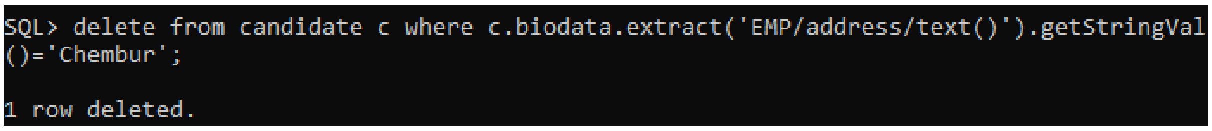


```
SQL> select c.biodata.extract('EMP/name/text()')"employee name",c.biodata.extract('EMP/
skill/compskill/lang/text()')"computer skill",c.biodata.extract('EMP/skill/compskill/os
/text()')"Os" from candidate C;

employee name
--------------------------------------------------------------------------------
computer skill
--------------------------------------------------------------------------------
Os
--------------------------------------------------------------------------------
Anjali
C++
Window

Pratik
java
Window

employee name
--------------------------------------------------------------------------------
computer skill
--------------------------------------------------------------------------------
Os
--------------------------------------------------------------------------------

Samar
c
Window
```

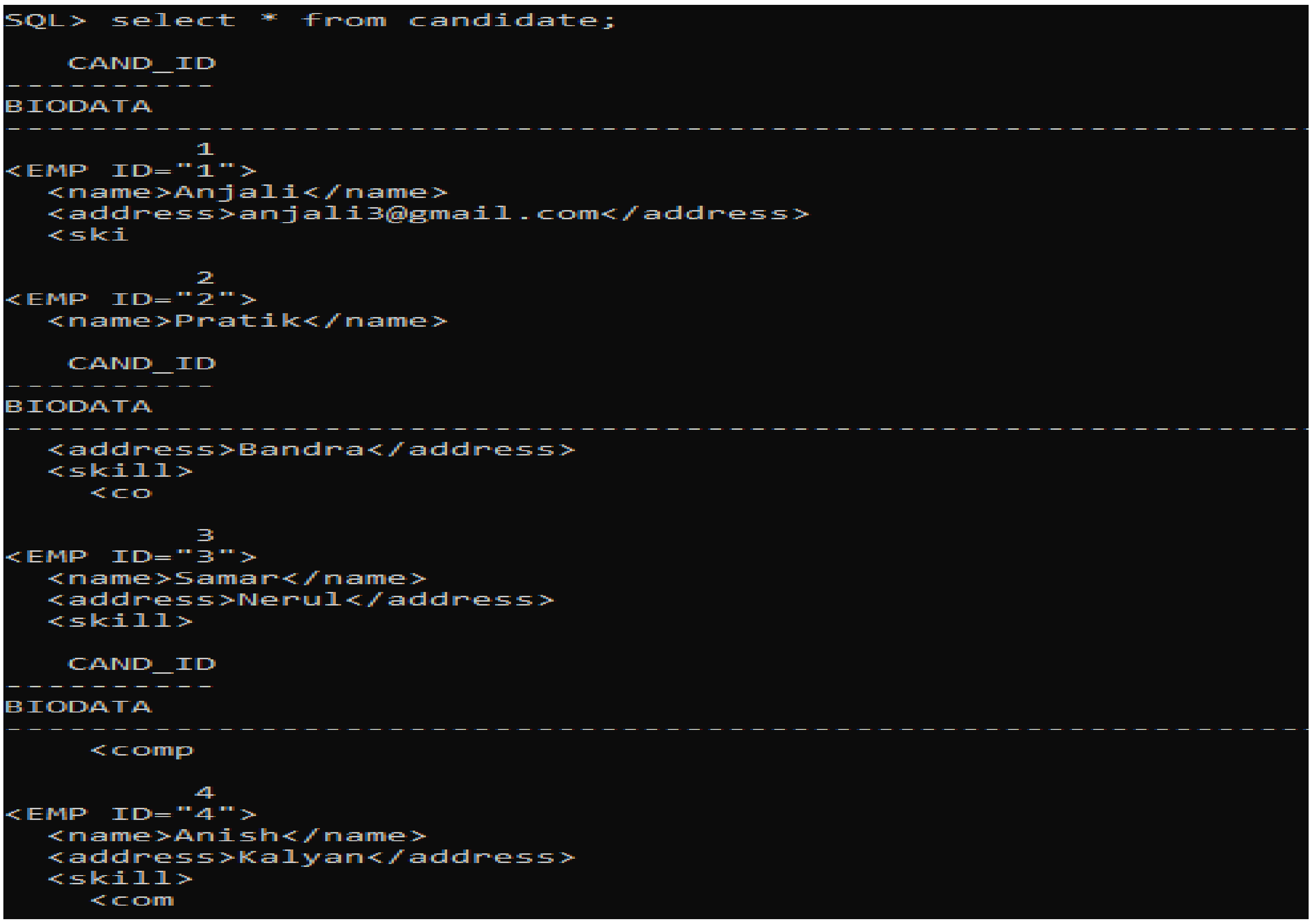


```
Anish
csharp

employee name
--------------------------------------------------------------------------------
computer skill
--------------------------------------------------------------------------------
Os
--------------------------------------------------------------------------------
Window

Riya
.net
Window
```

d) Delete record for address = Chembur

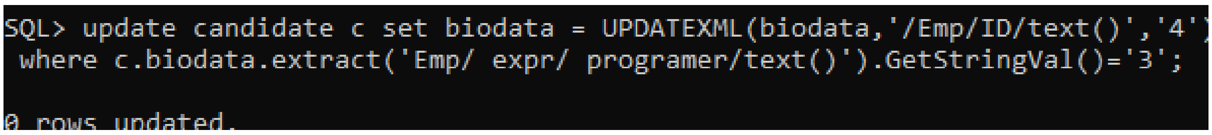


```
SQL> delete from candidate c where c.biodata.extract('EMP/address/text()').getStringVal
()='Chembur';

1 row deleted.
```



```
SQL> select * from candidate;

   CAND_ID
----------
BIODATA
------------------------------------------------------------------------------
         1
<EMP ID="1">
  <name>Anjali</name>
  <address>anjali3@gmail.com</address>
  <ski

         2
<EMP ID="2">
  <name>Pratik</name>

   CAND_ID
----------
BIODATA
------------------------------------------------------------------------------
  <address>Bandra</address>
  <skill>
    <co

         3
<EMP ID="3">
  <name>Samar</name>
  <address>Nerul</address>
  <skill>

   CAND_ID
----------
BIODATA
------------------------------------------------------------------------------
    <comp

         4
<EMP ID="4">
  <name>Anish</name>
  <address>Kalyan</address>
  <skill>
    <com
```

e) Update experience of a particular candidate



```
SQL> update candidate c set biodata = UPDATEXML(biodata,'/Emp/ID/text()','4'
 where c.biodata.extract('Emp/ expr/ programer/text()').GetStringVal()='3';

0 rows updated.
```

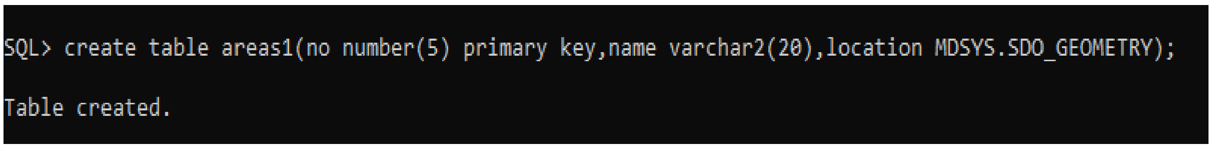
Practical 4

Aim: Create a spatial database table that stores the number, name and location, which consists of four different areas say ABC, PQR, MNO and XYZ.

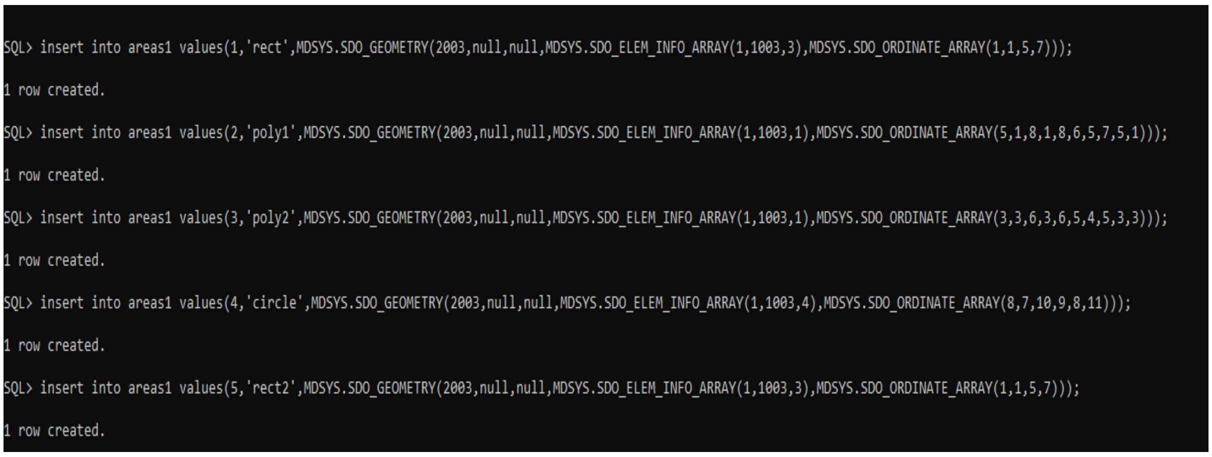
What is a Spatial Database?
Spatial data is associated with geographic locations such as cities, towns etc. A spatial database is optimised to store and query data representing objects. These are the objects which are defined in a geometric space.

Characteristics of Spatial Database
A spatial database system has the following characteristics
It is a database system
It offers spatial data types (SDTs) in its data model and query language.
It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

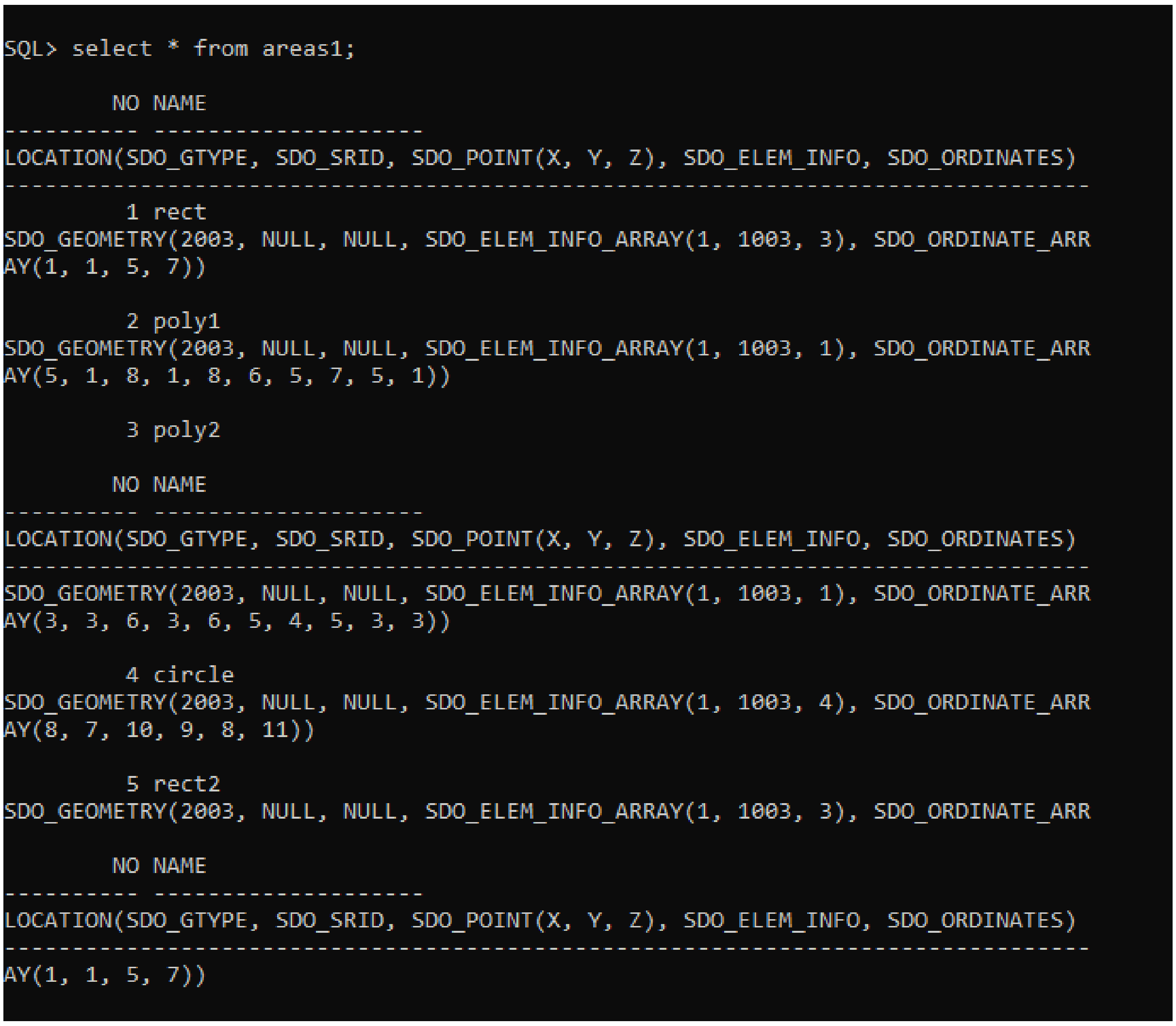
Queries:
1. Find the topological intersection of two geometries
2. Find whether two geometric figures are equivalent to each other
3. Find the areas of all different locations
4. Find the area of only one location



```
SQL> create table areas1(no number(5) primary key,name varchar2(20),location MDSYS.SDO_GEOMETRY);

Table created.
```

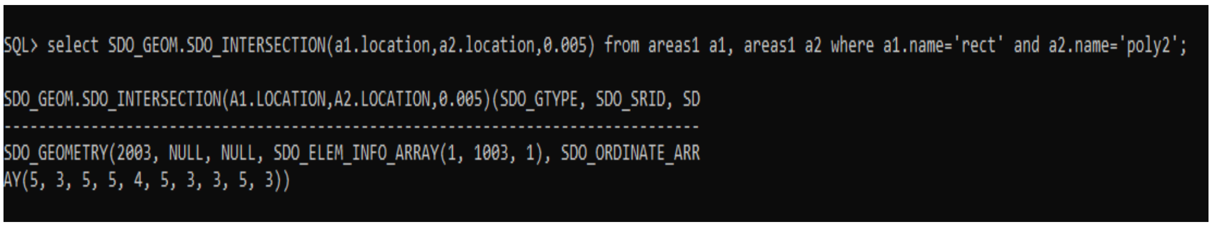


```
SQL> insert into areas1 values(1,'rect',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),MDSYS.SDO_ORDINATE_ARRAY(1,1,5,7)));

1 row created.

SQL> insert into areas1 values(2,'poly1',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY(5,1,8,1,8,6,5,7,5,1)));

1 row created.

SQL> insert into areas1 values(3,'poly2',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY(3,3,6,3,6,5,4,5,3,3)));

1 row created.

SQL> insert into areas1 values(4,'circle',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,4),MDSYS.SDO_ORDINATE_ARRAY(8,7,10,9,8,11)));

1 row created.

SQL> insert into areas1 values(5,'rect2',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),MDSYS.SDO_ORDINATE_ARRAY(1,1,5,7)));

1 row created.
```



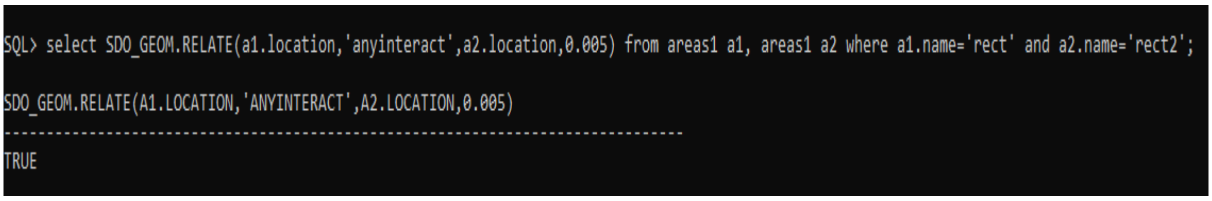
```
SQL> select * from areas1;

        NO NAME
---------- --------------------
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
         1 rect
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 3), SDO_ORDINATE_ARR
AY(1, 1, 5, 7))

         2 poly1
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(5, 1, 8, 1, 8, 6, 5, 7, 5, 1))

         3 poly2

        NO NAME
---------- --------------------
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(3, 3, 6, 3, 6, 5, 4, 5, 3, 3))

         4 circle
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 4), SDO_ORDINATE_ARR
AY(8, 7, 10, 9, 8, 11))

         5 rect2
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 3), SDO_ORDINATE_ARR

        NO NAME
---------- --------------------
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
AY(1, 1, 5, 7))
```

a) Find the topological intersection of two geometries



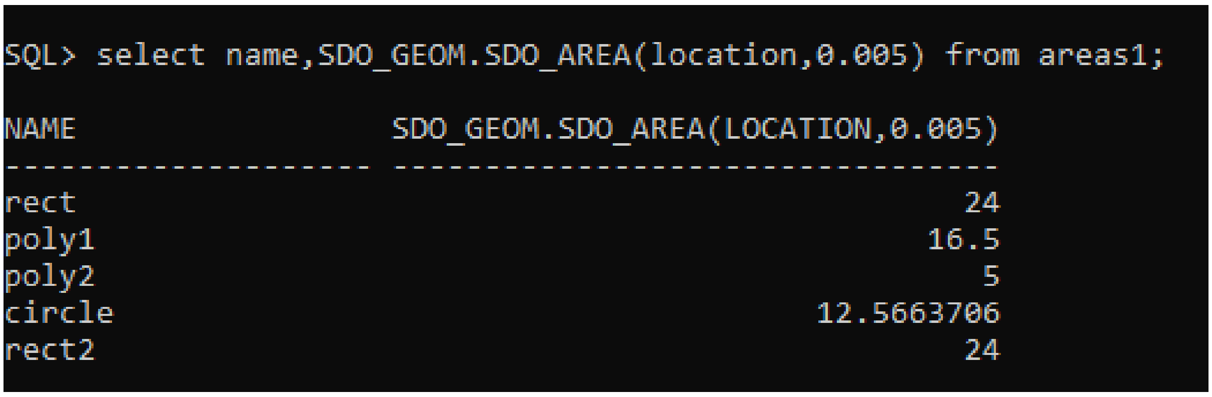
```
SQL> select SDO_GEOM.SDO_INTERSECTION(a1.location,a2.location,0.005) from areas1 a1, areas1 a2 where a1.name='rect' and a2.name='poly2';

SDO_GEOM.SDO_INTERSECTION(A1.LOCATION,A2.LOCATION,0.005)(SDO_GTYPE, SDO_SRID, SD
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(5, 3, 5, 5, 4, 5, 3, 3, 5, 3))
```

b)Find whether two geometric figures are equivalent to each other



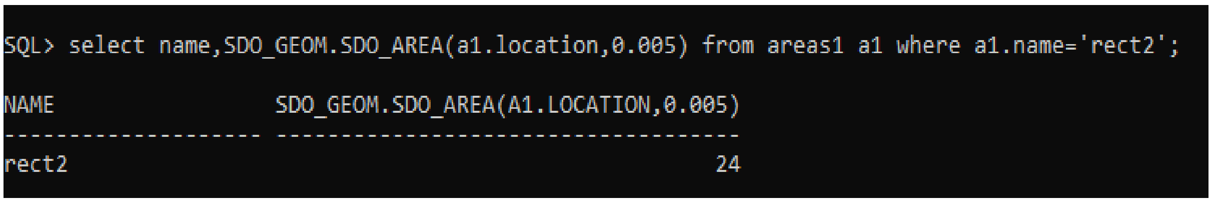
```
SQL> select SDO_GEOM.RELATE(a1.location,'anyinteract',a2.location,0.005) from areas1 a1, areas1 a2 where a1.name='rect' and a2.name='rect2';

SDO_GEOM.RELATE(A1.LOCATION,'ANYINTERACT',A2.LOCATION,0.005)
--------------------------------------------------------------------------------
TRUE
```

c) Find the areas of all different locations



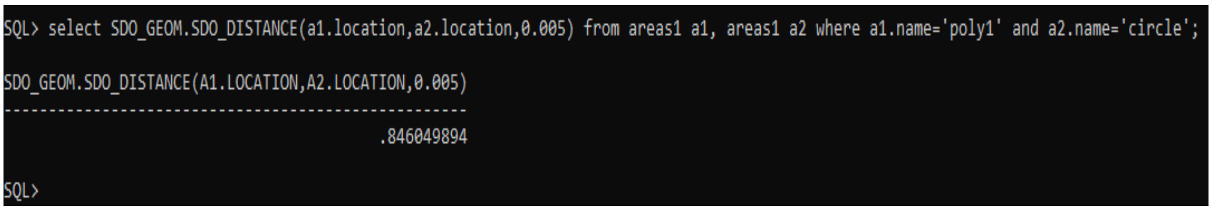
```
SQL> select name,SDO_GEOM.SDO_AREA(location,0.005) from areas1;

NAME                 SDO_GEOM.SDO_AREA(LOCATION,0.005)
-------------------- ---------------------------------
rect                                                24
poly1                                             16.5
poly2                                                5
circle                                      12.5663706
rect2                                               24
```

d)Find the area of only one location



```
SQL> select name,SDO_GEOM.SDO_AREA(a1.location,0.005) from areas1 a1 where a1.name='rect2';

NAME                 SDO_GEOM.SDO_AREA(A1.LOCATION,0.005)
-------------------- ------------------------------------
rect2                                                  24
```

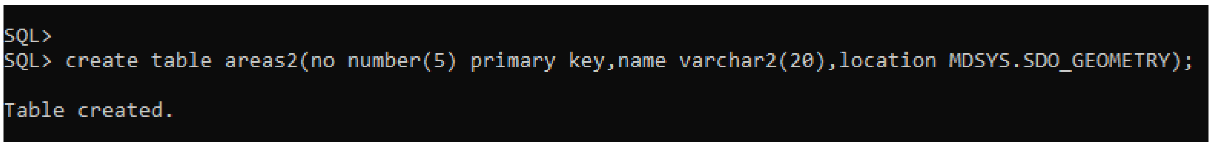
e) Find the distance between two geometries.



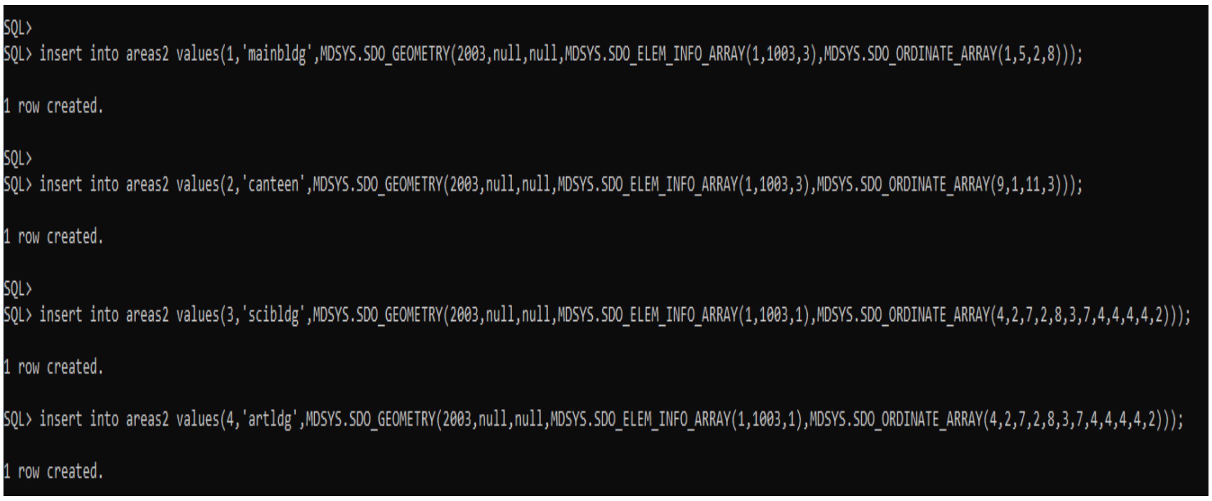
```
SQL> select SDO_GEOM.SDO_DISTANCE(a1.location,a2.location,0.005) from areas1 a1, areas1 a2 where a1.name='poly1' and a2.name='circle';

SDO_GEOM.SDO_DISTANCE(A1.LOCATION,A2.LOCATION,0.005)
----------------------------------------------------
                                          .846049894

SQL>
```

<u>Part-B</u>

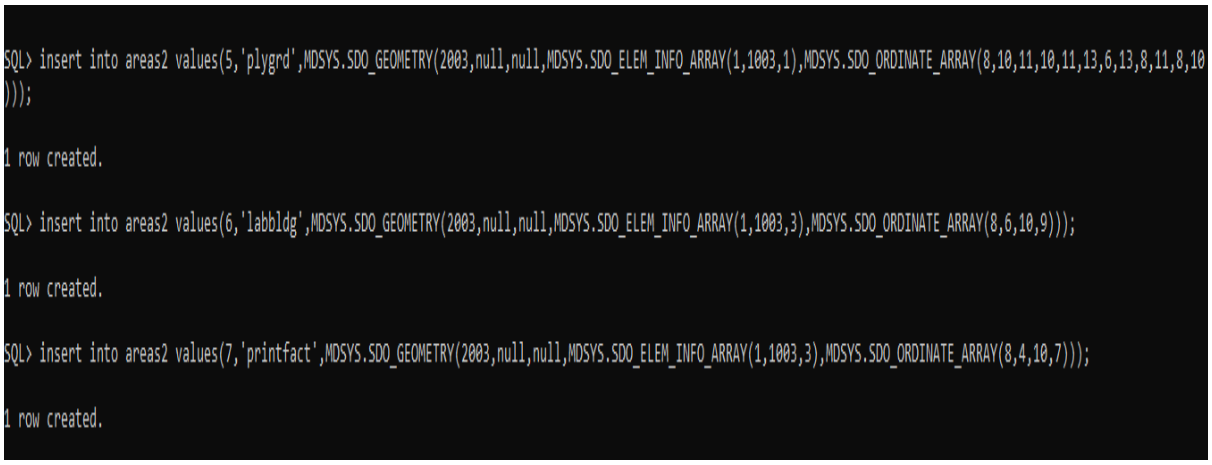
Aim: Create a spatial database according to the following diagram given below.



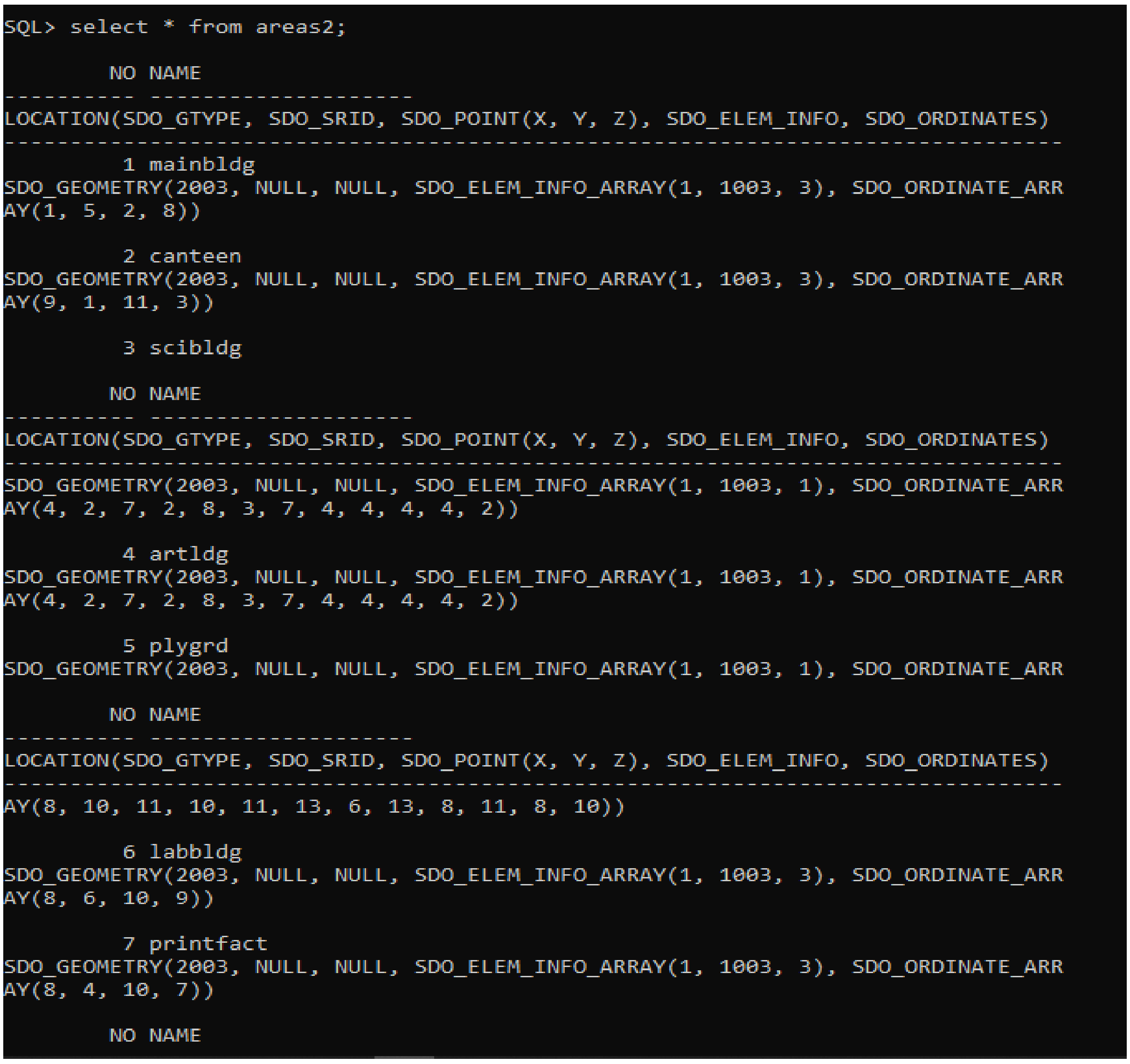
```
SQL>
SQL> create table areas2(no number(5) primary key,name varchar2(20),location MDSYS.SDO_GEOMETRY);

Table created.
```



```
SQL>
SQL> insert into areas2 values(1,'mainbldg',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),MDSYS.SDO_ORDINATE_ARRAY(1,5,2,8)));

1 row created.

SQL>
SQL> insert into areas2 values(2,'canteen',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),MDSYS.SDO_ORDINATE_ARRAY(9,1,11,3)));

1 row created.

SQL>
SQL> insert into areas2 values(3,'scibldg',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY(4,2,7,2,8,3,7,4,4,4,4,2)));

1 row created.

SQL> insert into areas2 values(4,'artldg',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY(4,2,7,2,8,3,7,4,4,4,4,2)));

1 row created.
```

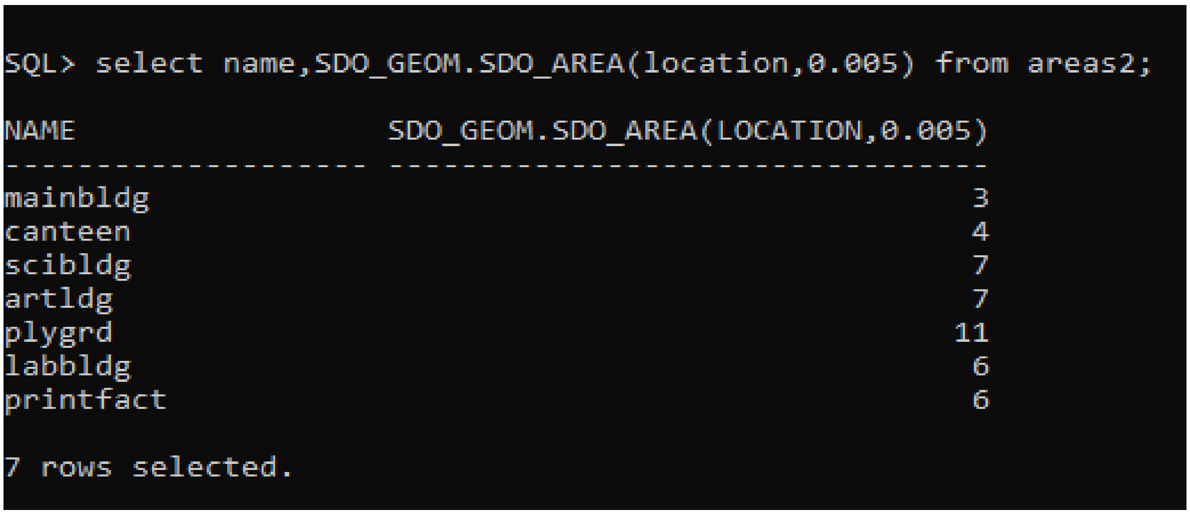


```
SQL> insert into areas2 values(5,'plygrd',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY(8,10,11,10,11,13,6,13,8,11,8,10
)));

1 row created.

SQL> insert into areas2 values(6,'labbldg',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),MDSYS.SDO_ORDINATE_ARRAY(8,6,10,9)));

1 row created.

SQL> insert into areas2 values(7,'printfact',MDSYS.SDO_GEOMETRY(2003,null,null,MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),MDSYS.SDO_ORDINATE_ARRAY(8,4,10,7)));

1 row created.
```



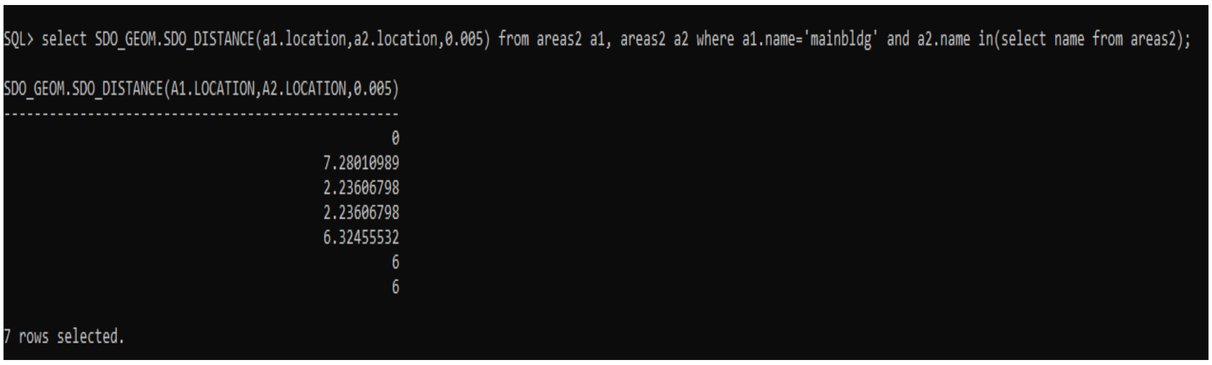
```
SQL> select * from areas2;

        NO NAME
---------- --------------------
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
         1 mainbldg
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 3), SDO_ORDINATE_ARR
AY(1, 5, 2, 8))

         2 canteen
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 3), SDO_ORDINATE_ARR
AY(9, 1, 11, 3))

         3 scibldg

        NO NAME
---------- --------------------
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(4, 2, 7, 2, 8, 3, 7, 4, 4, 4, 4, 2))

         4 artldg
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(4, 2, 7, 2, 8, 3, 7, 4, 4, 4, 4, 2))

         5 plygrd
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR

        NO NAME
---------- --------------------
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
AY(8, 10, 11, 10, 11, 13, 6, 13, 8, 11, 8, 10))

         6 labbldg
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 3), SDO_ORDINATE_ARR
AY(8, 6, 10, 9))

         7 printfact
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 3), SDO_ORDINATE_ARR
AY(8, 4, 10, 7))

        NO NAME
```

A)Display area of each object



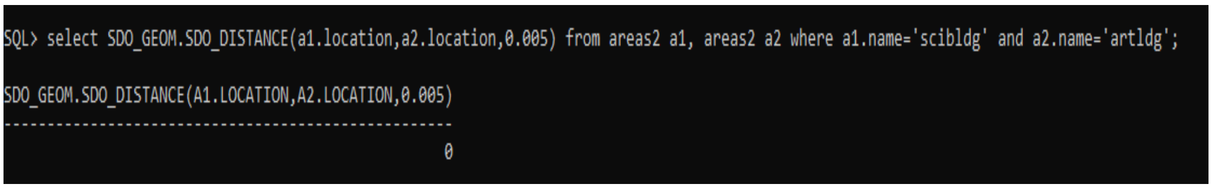
```
SQL> select name,SDO_GEOM.SDO_AREA(location,0.005) from areas2;

NAME                 SDO_GEOM.SDO_AREA(LOCATION,0.005)
-------------------- ---------------------------------
mainbldg                                             3
canteen                                              4
scibldg                                              7
artldg                                               7
plygrd                                              11
labbldg                                              6
printfact                                            6

7 rows selected.
```

b)Find the distance between the main building and all other buildings.



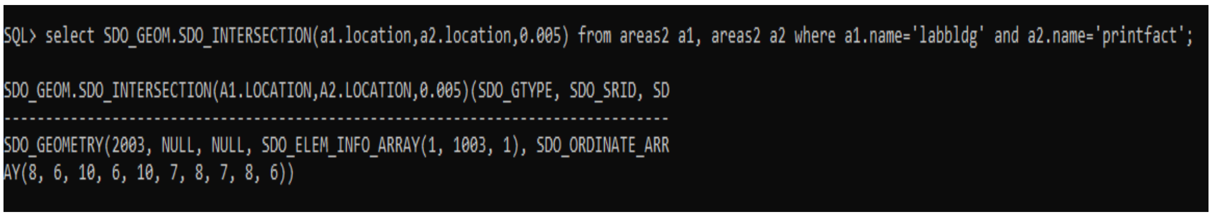
```
SQL> select SDO_GEOM.SDO_DISTANCE(a1.location,a2.location,0.005) from areas2 a1, areas2 a2 where a1.name='mainbldg' and a2.name in(select name from areas2);

SDO_GEOM.SDO_DISTANCE(A1.LOCATION,A2.LOCATION,0.005)
----------------------------------------------------
                                                   0
                                          7.28010989
                                          2.23606798
                                          2.23606798
                                          6.32455532
                                                   6
                                                   6

7 rows selected.
```

c)Find the distance between arts and science building.



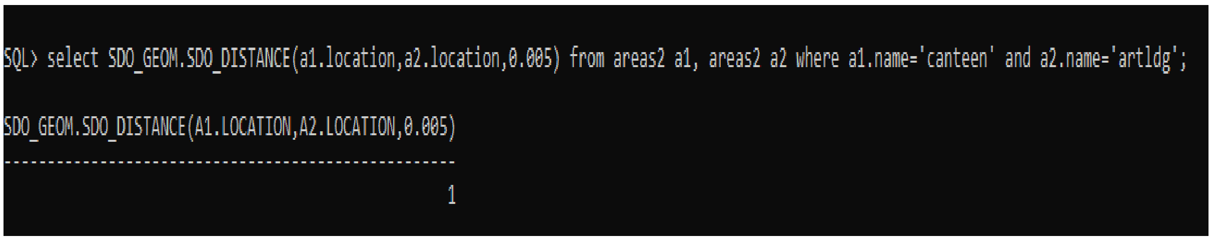
```
SQL> select SDO_GEOM.SDO_DISTANCE(a1.location,a2.location,0.005) from areas2 a1, areas2 a2 where a1.name='scibldg' and a2.name='artldg';

SDO_GEOM.SDO_DISTANCE(A1.LOCATION,A2.LOCATION,0.005)
----------------------------------------------------
                                                   0
```

d)Find the sharing area between lab and print facility.



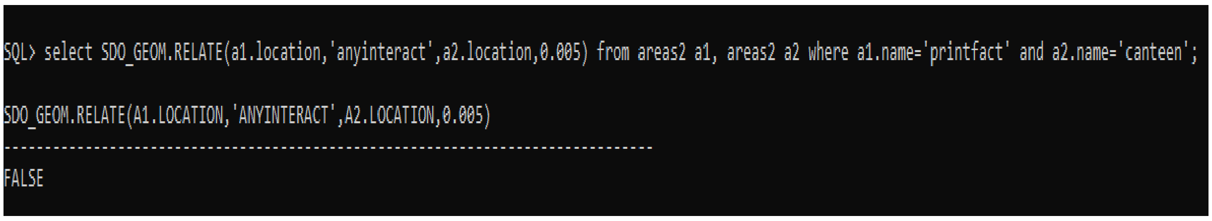
```
SQL> select SDO_GEOM.SDO_INTERSECTION(a1.location,a2.location,0.005) from areas2 a1, areas2 a2 where a1.name='labbldg' and a2.name='printfact';

SDO_GEOM.SDO_INTERSECTION(A1.LOCATION,A2.LOCATION,0.005)(SDO_GTYPE, SDO_SRID, SD
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(8, 6, 10, 6, 10, 7, 8, 7, 8, 6))
```

e)Find the distance between the arts building and canteen.



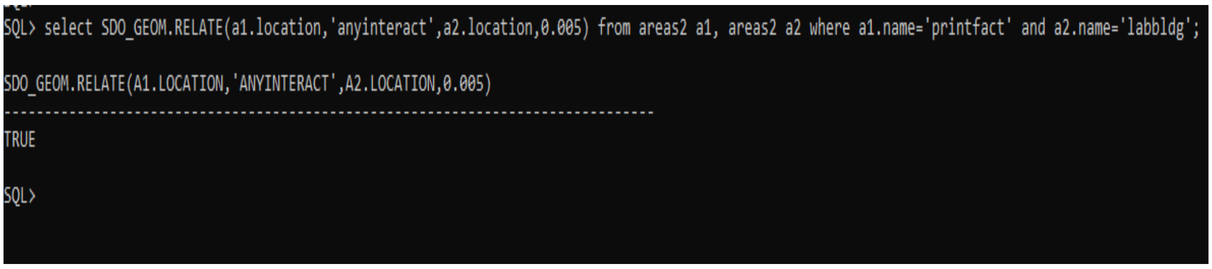
```
SQL> select SDO_GEOM.SDO_DISTANCE(a1.location,a2.location,0.005) from areas2 a1, areas2 a2 where a1.name='canteen' and a2.name='artldg';

SDO_GEOM.SDO_DISTANCE(A1.LOCATION,A2.LOCATION,0.005)
----------------------------------------------------
                                                   1
```

f)Find spatial relationship between print facility and canteen building



```
SQL> select SDO_GEOM.RELATE(a1.location,'anyinteract',a2.location,0.005) from areas2 a1, areas2 a2 where a1.name='printfact' and a2.name='canteen';

SDO_GEOM.RELATE(A1.LOCATION,'ANYINTERACT',A2.LOCATION,0.005)
--------------------------------------------------------------------------------
FALSE
```

g)Is there any spatial relationship between print facility and lab building.



```
SQL> select SDO_GEOM.RELATE(a1.location,'anyinteract',a2.location,0.005) from areas2 a1, areas2 a2 where a1.name='printfact' and a2.name='labbldg';

SDO_GEOM.RELATE(A1.LOCATION,'ANYINTERACT',A2.LOCATION,0.005)
--------------------------------------------------------------------------------
TRUE

SQL>
```

Part C:
Aim: Create a spatial database based on the following information

Create three relations State (region, name) City(centre,region,name) Rivers (name, route)

1. State 'st1' which extends from (10,10),(60,60),(50,10),(10,40)
2. State 'st2' which has two opposite corners situated at(100,50) & (150,20)
3. City 'C1' with centre at (15,35) region is circular with largest road of 10
4. City 'C2' with centre at (22,35) region is circular with largest road of 4.
5. City 'C3' with centre as (55,40) region is point
6. City 'C4' with centre (48,33) which is rectangular with corner situated at (40,30) & (55,15)
7. City 'C5' with centre (120,35) extending from (120,40) to(130,30)
8. River 'r1' with route extending from (15,25) to (52,58)
9. River 'r2' with route extending from (10,30) to (60,45)
10. River 'r3' with route extending from (55,30) to (110,30)

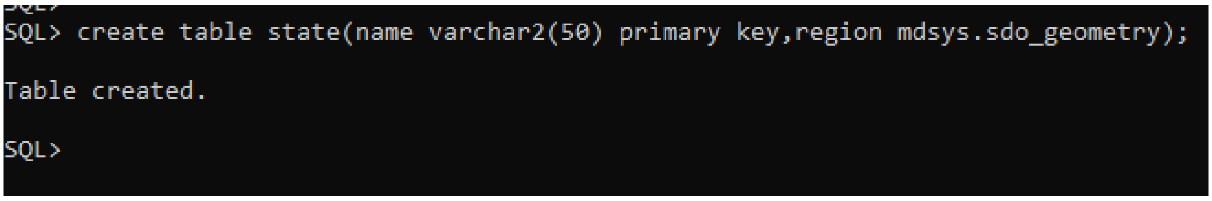
Queries

1. Locate all cities in state 'st1'.
2. Locate all cities in state 'st2'.
3. Locate all cities not more than 10 km from 'c3'.
4. Locate the cities touching city 'c2'
5. Locate a city within 5km from 'r2'.
6. Locate cities intersected by river 'r2'.
7. Find cities intersected by 'r3'.
8. Find the population in every city of state 'st1' if the population per sq.km. is 6..
9. Find distance between two states.

STEPS:

Create the following tables:
State with attributes

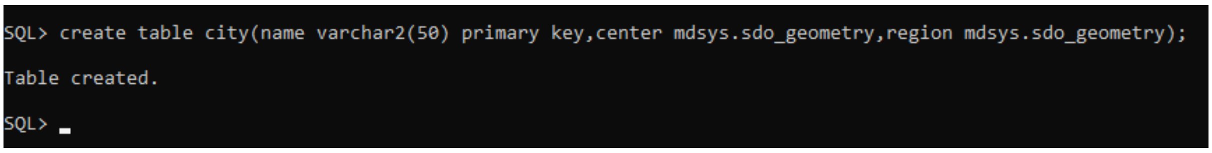
Name of type varchar2 and primary key and region as mdsys.sdo_geometry



```
SQL> create table state(name varchar2(50) primary key,region mdsys.sdo_geometry);

Table created.

SQL>
```

City with attributes

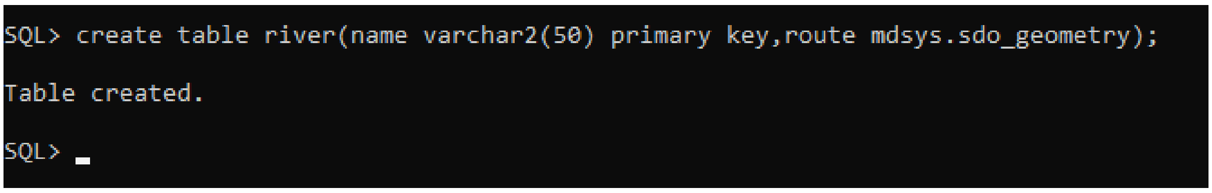
Name of type varchar2 and primary key and region as mdsys.sdo_geometry



```
SQL> create table city(name varchar2(50) primary key,center mdsys.sdo_geometry,region mdsys.sdo_geometry);

Table created.

SQL>
```

River with attributes

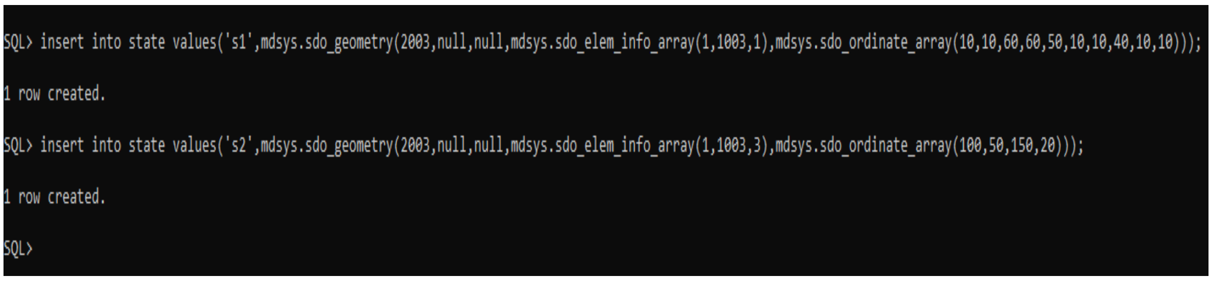
Name **of** type varchar2 and primary key and region as mdsys.sdo_geometry



```
SQL> create table river(name varchar2(50) primary key,route mdsys.sdo_geometry);

Table created.

SQL>
```

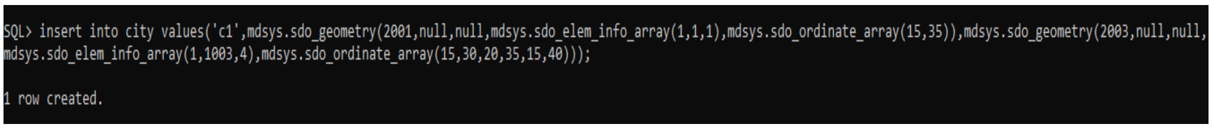
Inserting values

Inserting value in state table

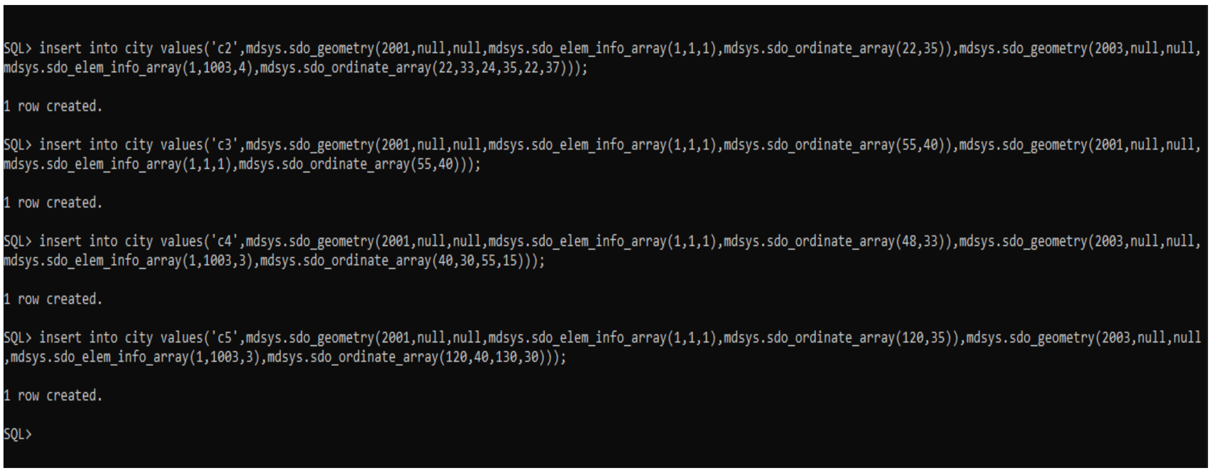


```
SQL> insert into state values('s1',mdsys.sdo_geometry(2003,null,null,mdsys.sdo_elem_info_array(1,1003,1),mdsys.sdo_ordinate_array(10,10,60,60,50,10,10,40,10,10)));

1 row created.

SQL> insert into state values('s2',mdsys.sdo_geometry(2003,null,null,mdsys.sdo_elem_info_array(1,1003,3),mdsys.sdo_ordinate_array(100,50,150,20)));

1 row created.

SQL>
```

Inserting value in city table

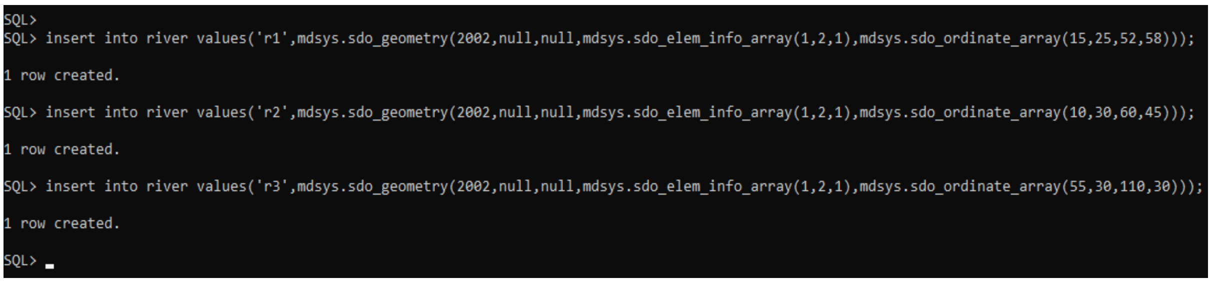


```
SQL> insert into city values('c1',mdsys.sdo_geometry(2001,null,null,mdsys.sdo_elem_info_array(1,1,1),mdsys.sdo_ordinate_array(15,35)),mdsys.sdo_geometry(2003,null,null,
mdsys.sdo_elem_info_array(1,1003,4),mdsys.sdo_ordinate_array(15,30,20,35,15,40)));

1 row created.
```



```
SQL> insert into city values('c2',mdsys.sdo_geometry(2001,null,null,mdsys.sdo_elem_info_array(1,1,1),mdsys.sdo_ordinate_array(22,35)),mdsys.sdo_geometry(2003,null,null,
mdsys.sdo_elem_info_array(1,1003,4),mdsys.sdo_ordinate_array(22,33,24,35,22,37)));

1 row created.

SQL> insert into city values('c3',mdsys.sdo_geometry(2001,null,null,mdsys.sdo_elem_info_array(1,1,1),mdsys.sdo_ordinate_array(55,40)),mdsys.sdo_geometry(2001,null,null,
mdsys.sdo_elem_info_array(1,1,1),mdsys.sdo_ordinate_array(55,40)));

1 row created.

SQL> insert into city values('c4',mdsys.sdo_geometry(2001,null,null,mdsys.sdo_elem_info_array(1,1,1),mdsys.sdo_ordinate_array(48,33)),mdsys.sdo_geometry(2003,null,null,
mdsys.sdo_elem_info_array(1,1003,3),mdsys.sdo_ordinate_array(40,30,55,15)));

1 row created.

SQL> insert into city values('c5',mdsys.sdo_geometry(2001,null,null,mdsys.sdo_elem_info_array(1,1,1),mdsys.sdo_ordinate_array(120,35)),mdsys.sdo_geometry(2003,null,null
,mdsys.sdo_elem_info_array(1,1003,3),mdsys.sdo_ordinate_array(120,40,130,30)));

1 row created.

SQL>
```

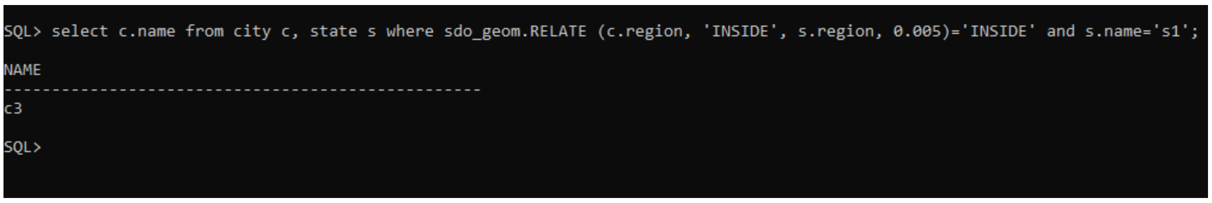
Inserting value into river table:



```
SQL>
SQL> insert into river values('r1',mdsys.sdo_geometry(2002,null,null,mdsys.sdo_elem_info_array(1,2,1),mdsys.sdo_ordinate_array(15,25,52,58)));

1 row created.

SQL> insert into river values('r2',mdsys.sdo_geometry(2002,null,null,mdsys.sdo_elem_info_array(1,2,1),mdsys.sdo_ordinate_array(10,30,60,45)));

1 row created.

SQL> insert into river values('r3',mdsys.sdo_geometry(2002,null,null,mdsys.sdo_elem_info_array(1,2,1),mdsys.sdo_ordinate_array(55,30,110,30)));

1 row created.

SQL> _
```

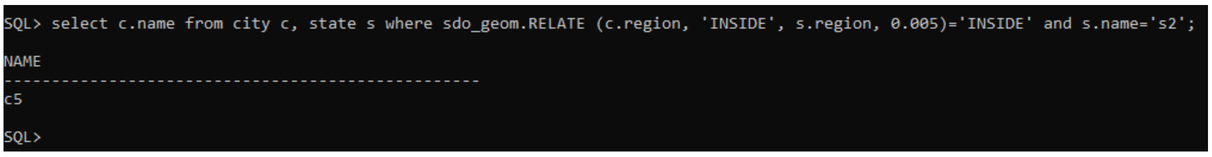
Queries

<u>a)Locate all cities in state ‘s1’</u>



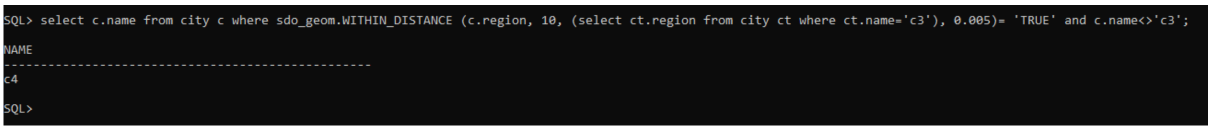
```
SQL> select c.name from city c, state s where sdo_geom.RELATE (c.region, 'INSIDE', s.region, 0.005)='INSIDE' and s.name='s1';

NAME
--------------------------------------------------
c3

SQL>
```

<u>b)Locate all cities in state ‘s2’</u>



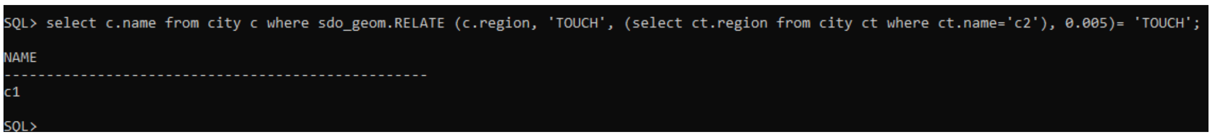
```
SQL> select c.name from city c, state s where sdo_geom.RELATE (c.region, 'INSIDE', s.region, 0.005)='INSIDE' and s.name='s2';

NAME
--------------------------------------------------
c5

SQL>
```

<u>c)Locate all cities not more than 10 km from ‘c3’</u>



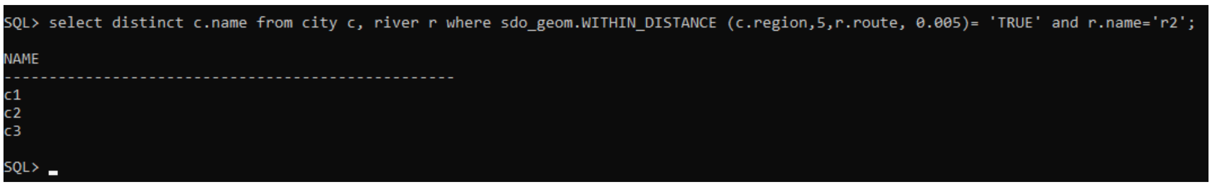
```
SQL> select c.name from city c where sdo_geom.WITHIN_DISTANCE (c.region, 10, (select ct.region from city ct where ct.name='c3'), 0.005)= 'TRUE' and c.name<>'c3';

NAME
--------------------------------------------------
c4

SQL>
```

d)Locate the cities touching to c2



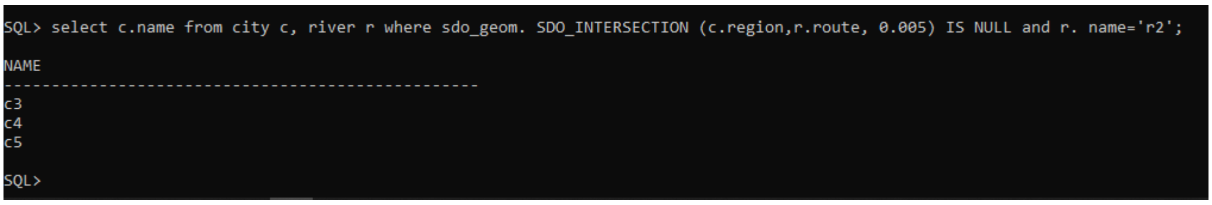
```
SQL> select c.name from city c where sdo_geom.RELATE (c.region, 'TOUCH', (select ct.region from city ct where ct.name='c2'), 0.005)= 'TOUCH';

NAME
--------------------------------------------------
c1

SQL>
```

e)Locate city within 5km from r2



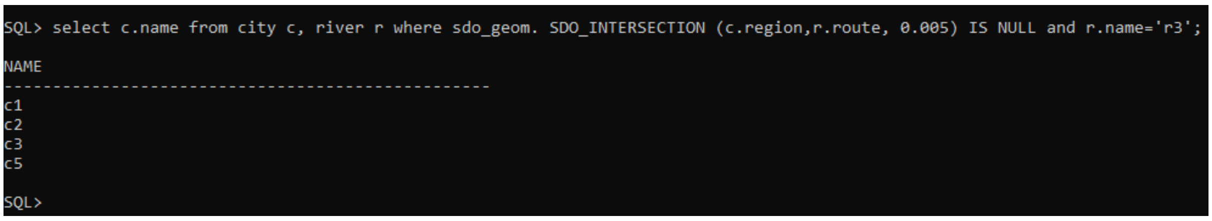
```
SQL> select distinct c.name from city c, river r where sdo_geom.WITHIN_DISTANCE (c.region,5,r.route, 0.005)= 'TRUE' and r.name='r2';

NAME
--------------------------------------------------
c1
c2
c3

SQL>
```

f)Locate cities intersected by river r2



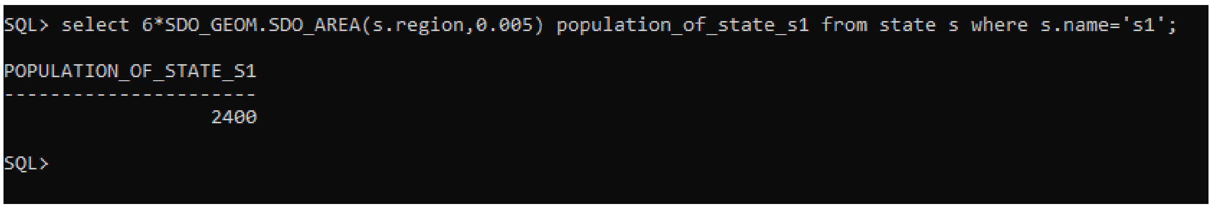
```
SQL> select c.name from city c, river r where sdo_geom. SDO_INTERSECTION (c.region,r.route, 0.005) IS NULL and r. name='r2';

NAME
--------------------------------------------------
c3
c4
c5

SQL>
```

g)Find cities intersected by ‘r3’



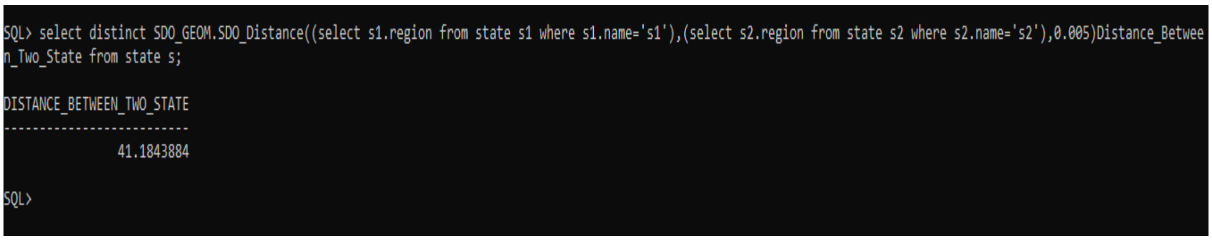
```
SQL> select c.name from city c, river r where sdo_geom. SDO_INTERSECTION (c.region,r.route, 0.005) IS NULL and r.name='r3';

NAME
--------------------------------------------------
c1
c2
c3
c5

SQL>
```

h)Find the population in every city of state 's1' if population per sq.km. is 6.



```
SQL> select 6*SDO_GEOM.SDO_AREA(s.region,0.005) population_of_state_s1 from state s where s.name='s1';

POPULATION_OF_STATE_S1
----------------------
                  2400

SQL>
```

i)Find the distance between two states.



```
SQL> select distinct SDO_GEOM.SDO_Distance((select s1.region from state s1 where s1.name='s1'),(select s2.region from state s2 where s2.name='s2'),0.005)Distance_Betwee
n_Two_State from state s;

DISTANCE_BETWEEN_TWO_STATE
--------------------------
                41.1843884

SQL>
```

Practical – 5

AIM: Distributed Databases

Perform Horizontal
Fragmentation

Create a global conceptual schema Emp(Eno;Ename;Address;Email;Salary) and insert 10 records. Divide Emp into verticall fragments using the condition that Emp1 contains the tuples with salary <= 10,000 and Emp2 with 10,000<salary<=20,000 on two different nodes.

THEORY:

What is Distributed Database?

A distributed database **is basically** a database that **is** not limited to one **system, it** is spread over different **sites**, i.e**, on** multiple computers or over a network of computers. A distributed database system is located on various sites that don't share physical components. This may be required when **a** particular database needs to be **accessed** by various users globally**. It needs to be** managed such that for the **users** it looks **like** one single **database**.

Distributed Data Storage:

There are 2 **ways in** which data can be stored on different sites. These are: **1. Replication -**
2. Fragmentation -

Fragmentation of relations can be done in two ways:

➤ Horizontal fragmentation **-** Splitting **by rows**

> **Vertical** fragmentation **-** Splitting by columns

What is Horizontal fragmentation?

✓ Horizontal fragmentation **refers** to **the** process **of** dividing a table horizontally **by assigning** each row or (a **group** of **rows) of** relation to one or **more** fragments.

✔ These fragments are then be assigned to different sides in the distributed system.
Some of the **rows** or **tuples** of the table are placed in one **system** and the rest are placed in other **systems**.

✔**The rows that** belong to the horizontal fragments are specified **by** a condition **on one** or more attributes **of** the relation**.**

In relational algebra horizontal fragmentation on **table T, can be** represented as **follows:**

Op*(T*
)

***where*, σ is *relational* algebra operator for selection**

p is the *condition* satisfied *by a horizontal* fragment

Fire the following queries:

(i) Find the salary **of all** employees.

(ii) Find the **Email of** all employees **where** salary=**12,000**.

(iii) Find the employee name and **Email** where employee number is known.

(iv) Find the employee name where salary>20,000.

Open the sql with system/root user.

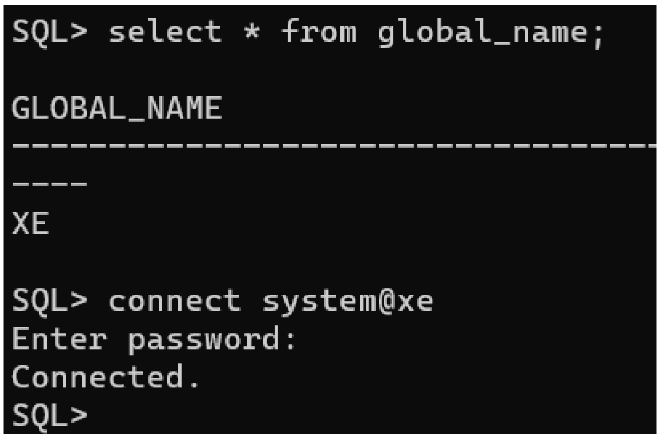
Check global database as:

```
> select * from global_name;
```

Connect to global database first.

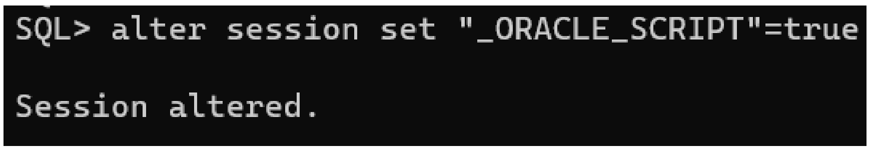
```
SQL>Connect system@xe
Enter password:
```

Connected.



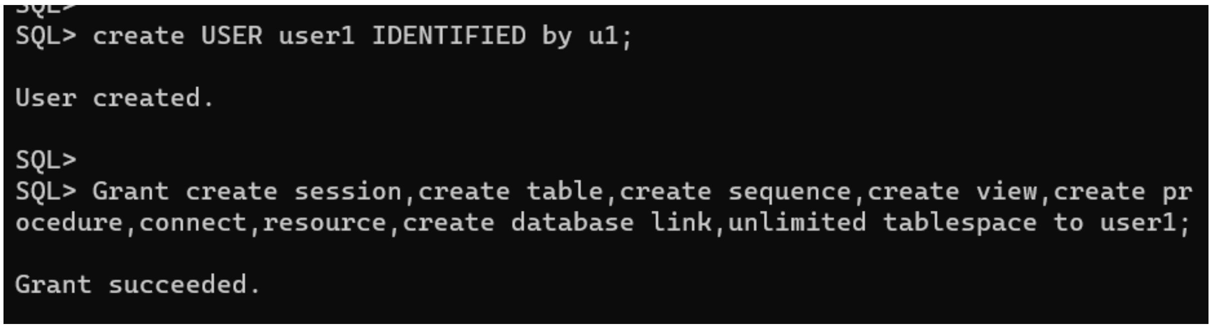
```
SQL> select * from global_name;

GLOBAL_NAME
------------------------------------
----
XE

SQL> connect system@xe
Enter password:
Connected.
SQL>
```

Alter session to get the access to create users
SQL> alter session set "**_ORACLE_SCRIPT**"=true;
Session altered.



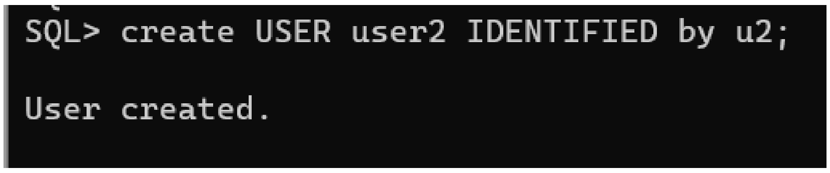
```
SQL> alter session set "_ORACLE_SCRIPT"=true

Session altered.
```

Create different users and grant them various privileges.
SQL> **create** USER **user1** IDENTIFIED **by** u1;
SQL> Grant **create** session, create **table**,create **sequence,**create **view,create** procedure**,**connect,resource**,** create database **link,** unlimited tablespace to user1;

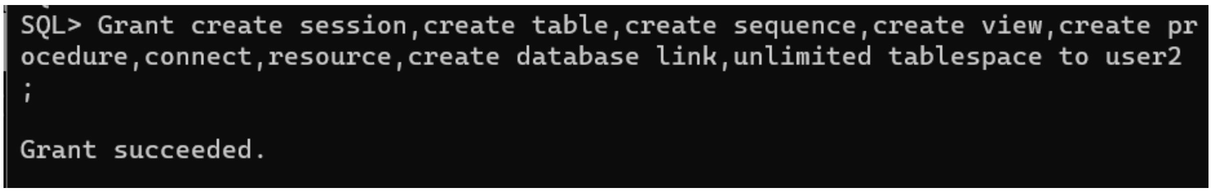


```
SQL> create USER user1 IDENTIFIED by u1;

User created.

SQL>
SQL> Grant create session,create table,create sequence,create view,create pr
ocedure,connect,resource,create database link,unlimited tablespace to user1;

Grant succeeded.
```

SQL> **create** USER user2 IDENTIFIED by **u2;**
SQL> **Grant** create **session,** create table,create sequence,create view**,**create procedure**,**connect,resource**,** create **database link**, **unlimited** tablespace to user2;

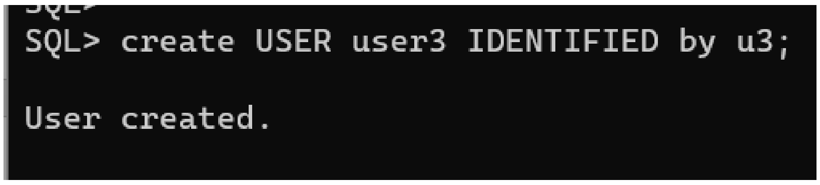


```
SQL> create USER user2 IDENTIFIED by u2;

User created.
```



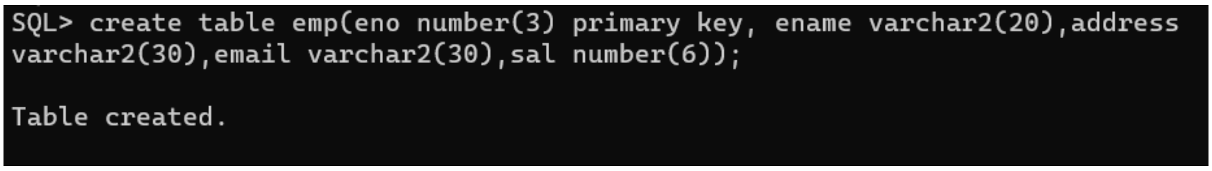
```
SQL> Grant create session,create table,create sequence,create view,create pr
ocedure,connect,resource,create database link,unlimited tablespace to user2
;

Grant succeeded.
```

SQL> create USER user3 IDENTIFIED **by** u3;
SQL> Grant create session,create table,create **sequence**,create view,**create procedure,**connect,resource, create database link, unlimited tablespace to user3;



```
SQL> create USER user3 IDENTIFIED by u3;

User created.
```

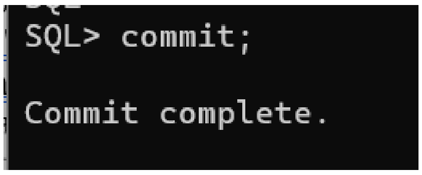
Creating table and inserting 10 data.
SQL> create **table** emp(eno number(3**)** primary key**,**ename varchar2(**20),**address varchar2**(30)**,email varchar2(30**),**sal number(6));



```
SQL> create table emp(eno number(3) primary key, ename varchar2(20),address
varchar2(30),email varchar2(30),sal number(6));

Table created.
```

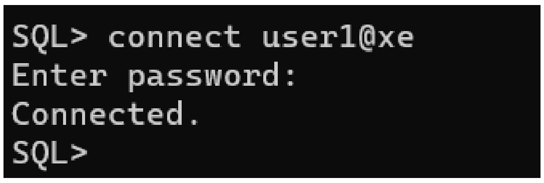
SQL> **insert** into emp values(101,'ram','dadar','ram@gmail.com',10000);
SQL> insert into emp values(102,'tom**','cst**', 'tom**@gmail.com',25000)**;
SQL> insert into emp values(103,'sam','wadala','sam@gmail.com',30000);
SQL> **insert** into emp values(104**,**'sonu', 'wadala','sonu**@gmail.com',12000)**;
SQL> insert into emp values(105**,'monu','**matunga','monu**@gmail.com'**,24000);
SQL> insert **into** emp values(106,'**mona**','sion','mona**@gmail.com',**35000);
SQL> insert into emp values(107,**'sona', 'sion','sona@gmail.com'**,40000);
SQL> insert into emp values(108**,**'harry'**,'kurla'**,'harry@gmail.com',30000**)**;
SQL> insert into emp values(109**,'**marry'**,**'kurla','marry@gmail.com',**60000)**;
SQL> insert into emp values(110,'anni','kurla',**'anni@gmail.com',10000)**;

Perform Commit.
SQL> commit;



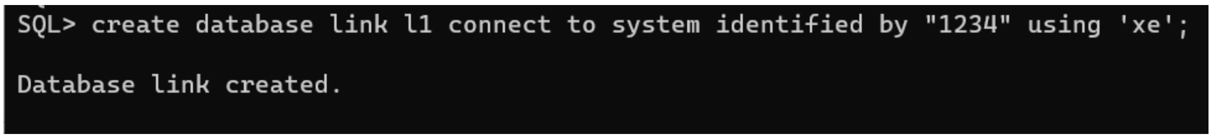
```
SQL> commit;

Commit complete.
```

Now open another sqlplus and login with user1 id and password.
SQL> connect user1@xe



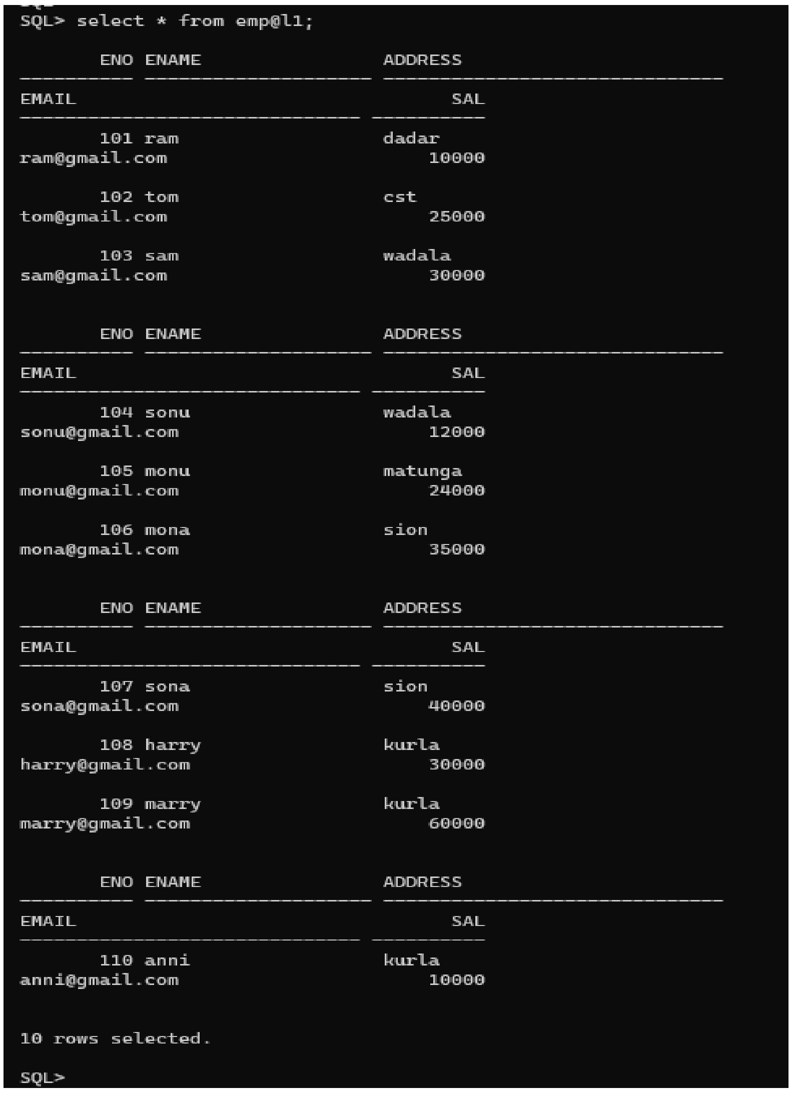
```
SQL> connect user1@xe
Enter password:
Connected.
SQL>
```

Create database link to global database in order to access the data of global database table.
SQL> create database link 11 connect to system identified by system using 'xe';



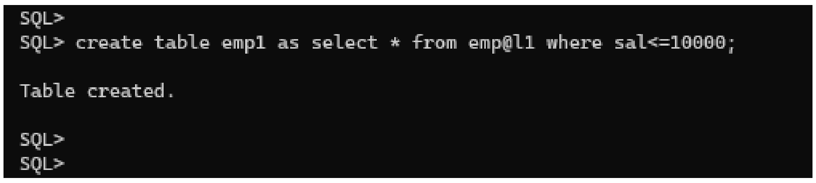
```
SQL> create database link l1 connect to system identified by "1234" using 'xe';

Database link created.
```

Fire select **query to check if the link to databases is functioning**
well.
SQL> select
***from**emp@11;



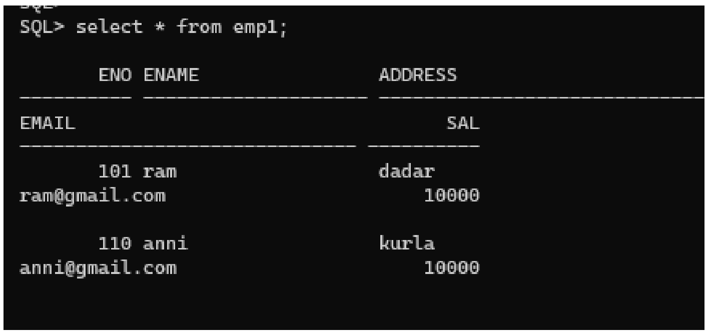
```
SQL> select * from emp@l1;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       101 ram                  dadar
ram@gmail.com                       10000

       102 tom                  cst
tom@gmail.com                       25000

       103 sam                  wadala
sam@gmail.com                       30000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       104 sonu                 wadala
sonu@gmail.com                      12000

       105 monu                 matunga
monu@gmail.com                      24000

       106 mona                 sion
mona@gmail.com                      35000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       107 sona                 sion
sona@gmail.com                      40000

       108 harry                kurla
harry@gmail.com                     30000

       109 marry                kurla
marry@gmail.com                     60000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       110 anni                 kurla
anni@gmail.com                      10000


10 rows selected.

SQL>
```

Create table and insert the data from global database table where all sal<=10000.
SQL> create table emp1 **as select** * from emp@11 where sal<=10000;



```
SQL>
SQL> create table emp1 as select * from emp@l1 where sal<=10000;

Table created.

SQL>
SQL>
```

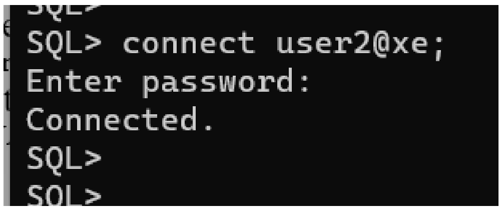
View the inserted data into the created table.
SQL> select * from emp1;



```
SQL> select * from emp1;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                SAL
------------------------------ ----------
       101 ram                  dadar
ram@gmail.com                      10000

       110 anni                 kurla
anni@gmail.com                     10000
```

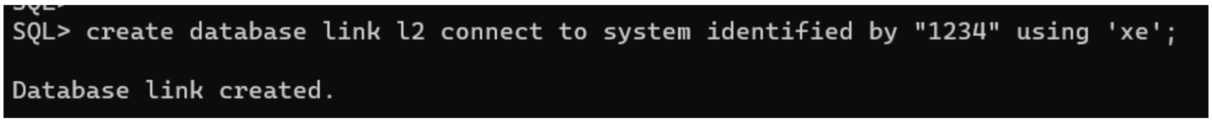
Perform the commit.
SQL> **commit**;
Commit complete.

Now open another sqlplus and login with user2 id and password.
SQL> connect user2@xe



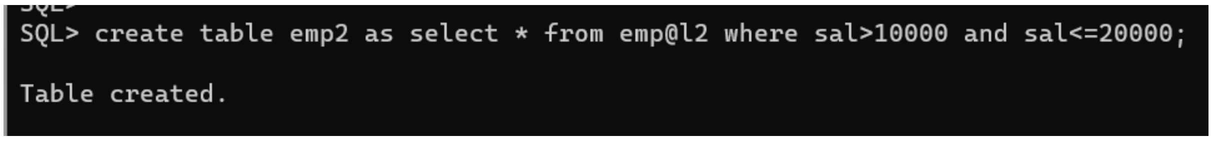
```
SQL> connect user2@xe;
Enter password:
Connected.
SQL>
SQL>
```

Create database link to global database in order to access the data of global database table.
SQL> create database link l22 connect **to** system identified by system using 'xe';



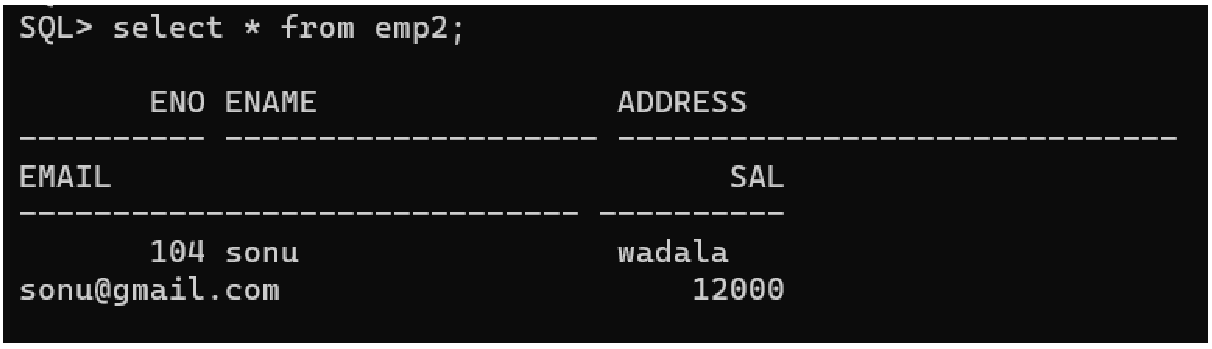
```
SQL> create database link l2 connect to system identified by "1234" using 'xe';

Database link created.
```

Create table and insert the data from global database table where all 10000 < sal<= 20000.
SQL> create **table** emp2 as select * **from** emp@122 where sal**>10000** and sal<=20000;



```
SQL> create table emp2 as select * from emp@l2 where sal>10000 and sal<=20000;

Table created.
```

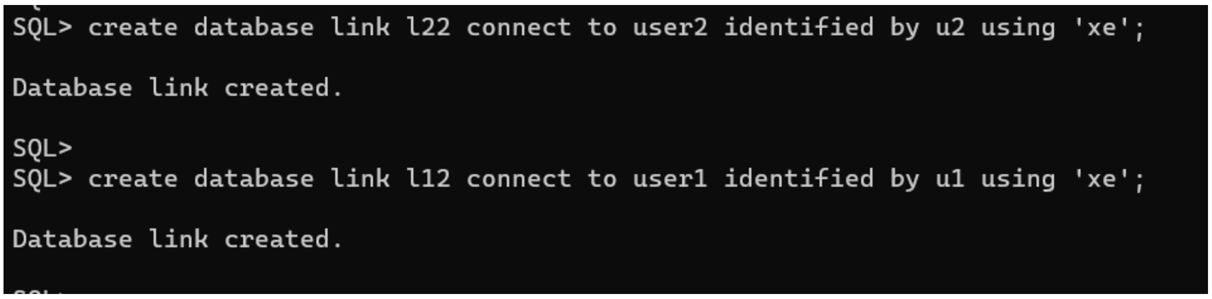
SQL> select * **from** emp2;



```
SQL> select * from emp2;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       104 sonu                 wadala
sonu@gmail.com                      12000
```

Perform the commit.
SQL> commit;
Commit complete.

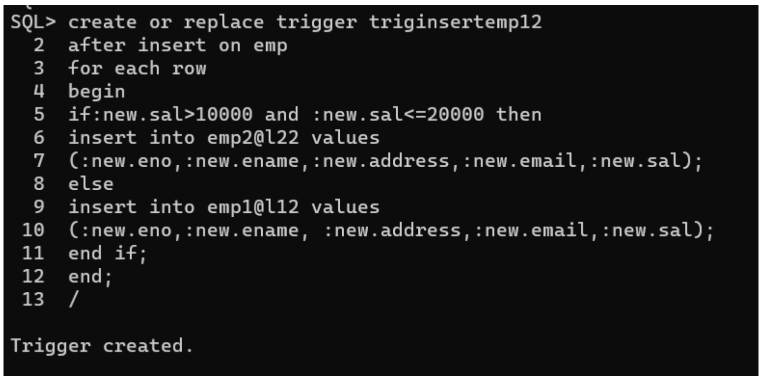
In Central database
Now let's create insert trigger in global database table in order to maintain the consistency of the data in all the distributed tables.
SQL> **create** database link 122 **connect** to user2 identified **by u2** using '**xe**';
SQL> **create** database link 11 connect to user1 identified by u1 using **'**xe';



```
SQL> create database link l22 connect to user2 identified by u2 using 'xe';

Database link created.

SQL>
SQL> create database link l12 connect to user1 identified by u1 using 'xe';

Database link created.
```

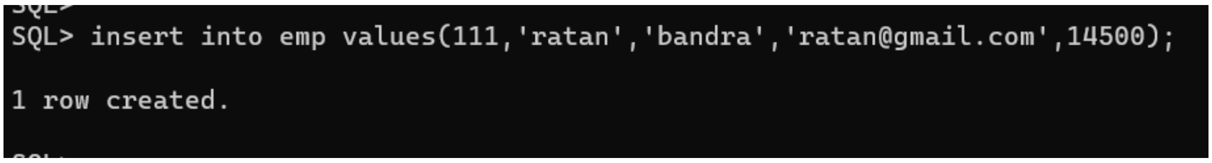
Creating insert trigger

SQL> create or replace trigger triginsertemp12
2 after insert on emp
3 for each row
4 begin
5 **if:**new.sal>10000 and **:new.sal<**=20000 then
6 insert into emp2**@122** values
7 (:new.eno**,:**new.ename**,:**new.address**,**:new.email,**:**new.sal);
8 else
9 insert into emp1**@**l1 values
10 (:new.eno**,:new.ename,** :new.address**,:**new.email,**:**new.sal**)**;
11 end if**;**
12 end;
13/



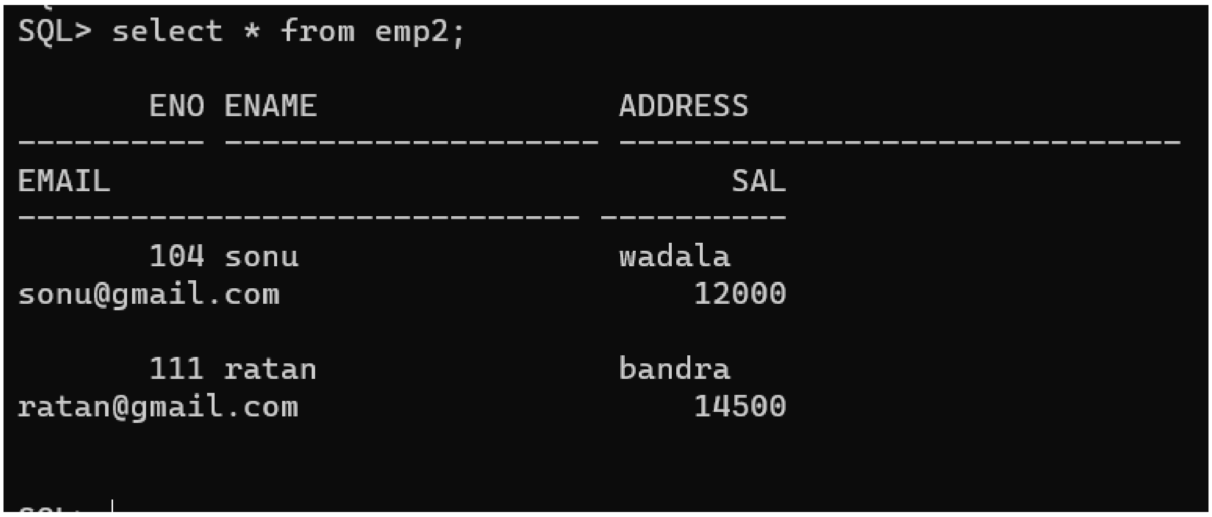
```
SQL> create or replace trigger triginsertemp12
  after insert on emp
  for each row
  begin
  if:new.sal>10000 and :new.sal<=20000 then
  insert into emp2@l22 values
  (:new.eno,:new.ename,:new.address,:new.email,:new.sal);
  else
  insert into emp1@l12 values
  (:new.eno,:new.ename, :new.address,:new.email,:new.sal);
  end if;
  end;
  /

Trigger created.
```

Let's insert one row to the global table and check whether it is being available in the distributed table or not.
SQL> **insert** into emp values(111,'ratan','bandra','ratan@**gmail.com**',**14500)**;
1 row created.



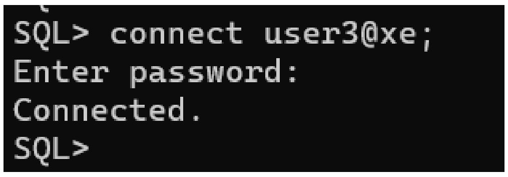
```
SQL> insert into emp values(111,'ratan','bandra','ratan@gmail.com',14500);

1 row created.
```

Perform the commit.
SQL> commit**;**
Commit complete.
Check in table emp2 if this data is being added or not, since the sal>14500 it should be added in emp2.
SQL> select * from emp2;



```
SQL> select * from emp2;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       104 sonu                 wadala
sonu@gmail.com                      12000

       111 ratan                bandra
ratan@gmail.com                     14500
```

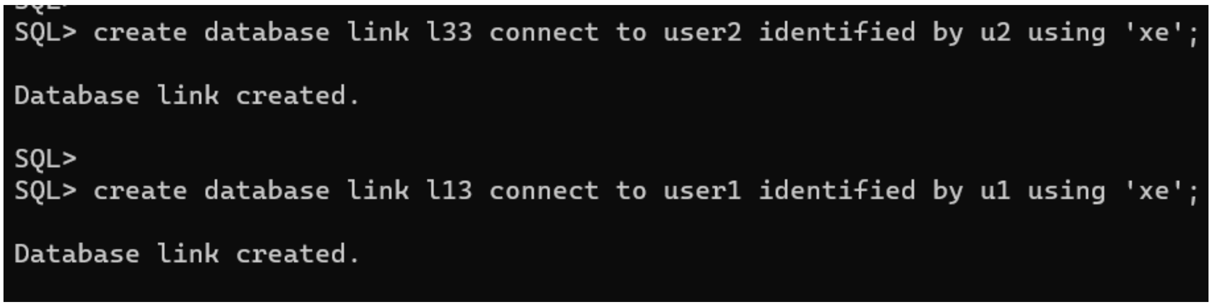
Now open another sqlplus and login with user3 id and password.
SQL> connect user3@**xe**
Connected.



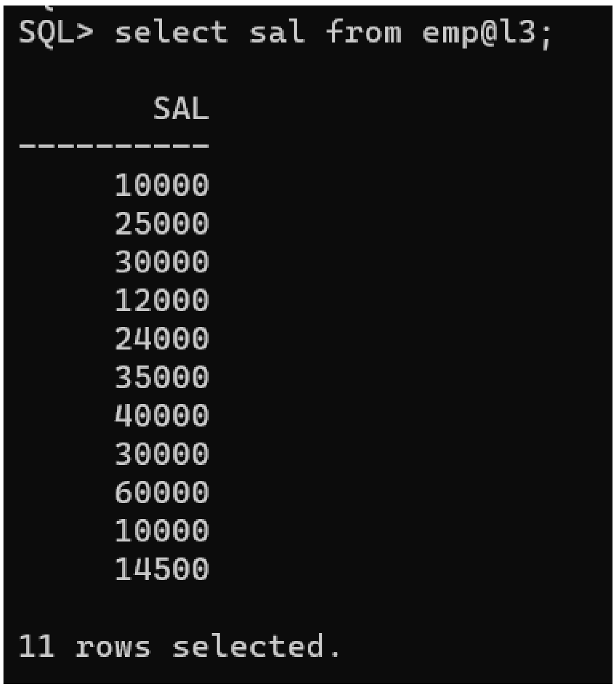
```
SQL> connect user3@xe;
Enter password:
Connected.
SQL>
```

Create database link for all the three tables here in order to access various types of data based on give question

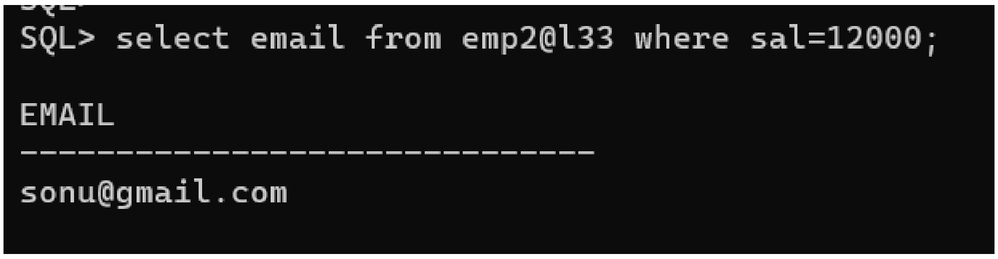
SQL> create database link **10 connect** to system identified by system using **'xe'**;
SQL> create database link **122 connect** to **user2** identified by **u2** using **'xe'**;
SQL> **create database** link 11 connect to user1 identified by **u1** using 'xe';


```
SQL> create database link l33 connect to user2 identified by u2 using 'xe';

Database link created.

SQL>
SQL> create database link l13 connect to user1 identified by u1 using 'xe';

Database link created.
```

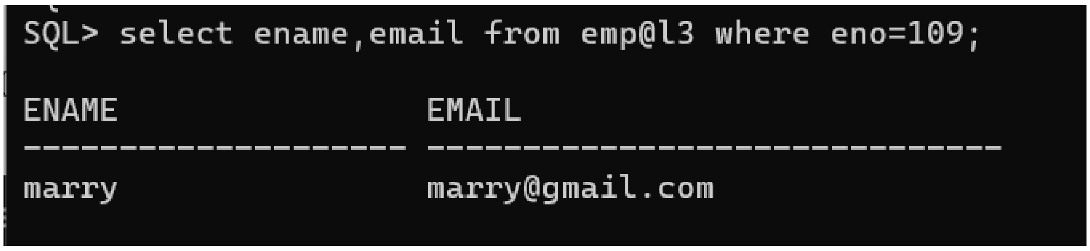
(i) Find the salary of all employees.
SQL> **select** sal from emp@10**;**


```
SQL> select sal from emp@l3;

       SAL
----------
     10000
     25000
     30000
     12000
     24000
     35000
     40000
     30000
     60000
     10000
     14500

11 rows selected.
```

(ii) Find the Email of all employees where salary=12,000.
SQL> select email from emp2@122 where sal=**12000**;

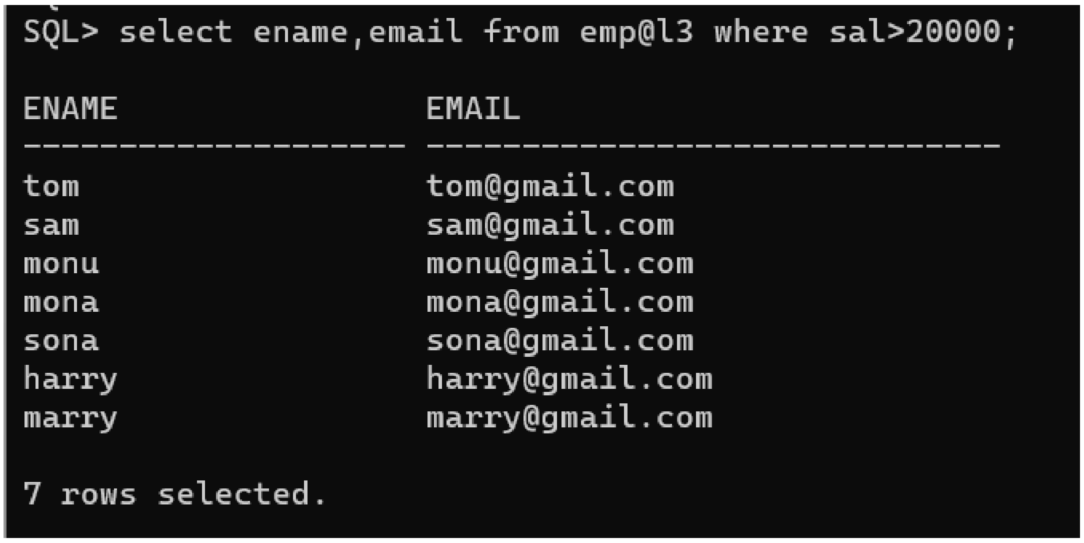

```
SQL> select email from emp2@l33 where sal=12000;

EMAIL
------------------------------
sonu@gmail.com
```

(iii) Find the employee name and Email where employee number is **known**.
SQL> select ename,**email** from emp@10 where eno=109;



```
SQL> select ename,email from emp@l3 where eno=109;

ENAME                EMAIL
-------------------- ------------------------------
marry                marry@gmail.com
```

(iv) Find the employee name whose salary >20000
SQL> select ename from emp@10 where **sal**>20000;



```
SQL> select ename,email from emp@l3 where sal>20000;

ENAME                EMAIL
-------------------- ------------------------------
tom                  tom@gmail.com
sam                  sam@gmail.com
monu                 monu@gmail.com
mona                 mona@gmail.com
sona                 sona@gmail.com
harry                harry@gmail.com
marry                marry@gmail.com

7 rows selected.
```

Practical No:6

Aim: Distributed Database
Perform Vertical Fragmentation

Create a global conceptual schema Emp(Eno;Ename;Address;email;Salary) and insert 10 records. Divide Emp into vertical fragments Emp1(Eno;Ename;Address) and Emp2(Eno;Email;Salary) on two **different nodes**.

THEORY:

What is vertical fragmentation?

✓ Vertical fragmentation refers to **the** process of decomposing a table vertically **by** attributes are **columns**.
In **this** fragmentation**,** some of the attributes are stored in one system and **the** rest are stored in other systems.

✓ This **is because each** site **may** not need **all** columns of a table.
In order to take care of restoration**, each** fragment must contain the **primary** key field(s) in a table.
The fragmentation **should** be in such a manner that **we** can rebuild **a table** from the fragment by taking the natural JOIN operation **and** to make it possible we **need** to **include** a **special** attribute called Tuple**-id to** the schema.

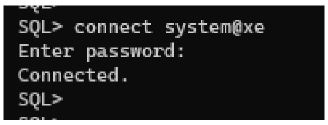
For **this** purpose**, a** user can use any super key. And by this**,** the tuples or **rows** can be linked together. **The** projection is as follows:

***πa1*, a2,..., an**
(T)
where*, π is relational algebra *operator
al...., an are ***the aatriubutes of*** *T*
***T* is the table** (relation)

Fire the following queries:
a) Find the salary of an employee where employee number **is** known.
b) Find the Email where the employee name is known.
c) Find the employee name and email where employee **number is** known.
d) Find the employee name whose salary is **> 2000**.

Open sqlplus with system user and connect to the global database.

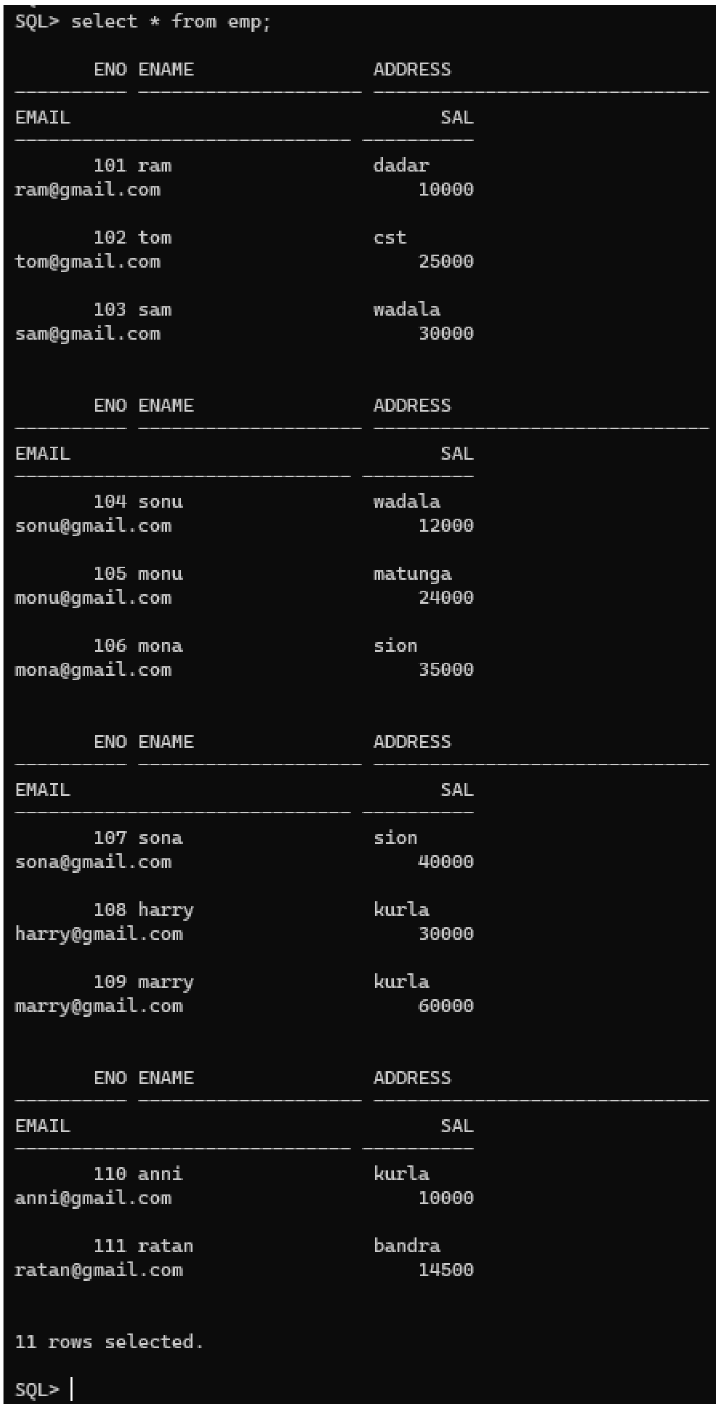
SQL> connect system@xe



```
SQL> connect system@xe
Enter password:
Connected.
SQL>
```

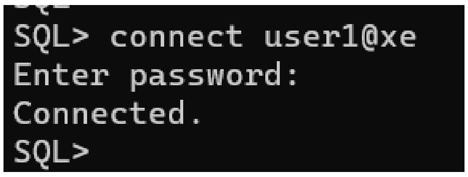
Check if the table already exist. In our case the table already exist.

SQL> select * **from** emp;



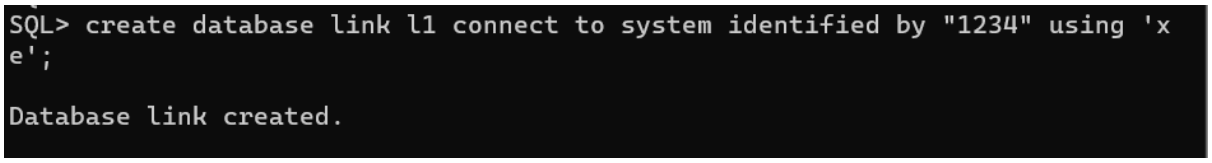
```
SQL> select * from emp;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       101 ram                  dadar
ram@gmail.com                       10000

       102 tom                  cst
tom@gmail.com                       25000

       103 sam                  wadala
sam@gmail.com                       30000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       104 sonu                 wadala
sonu@gmail.com                      12000

       105 monu                 matunga
monu@gmail.com                      24000

       106 mona                 sion
mona@gmail.com                      35000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       107 sona                 sion
sona@gmail.com                      40000

       108 harry                kurla
harry@gmail.com                     30000

       109 marry                kurla
marry@gmail.com                     60000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       110 anni                 kurla
anni@gmail.com                      10000

       111 ratan                bandra
ratan@gmail.com                     14500


11 rows selected.

SQL> |
```

Now open another sqlplus and login with user1 id and password.
SQL> connect user1**@xe**



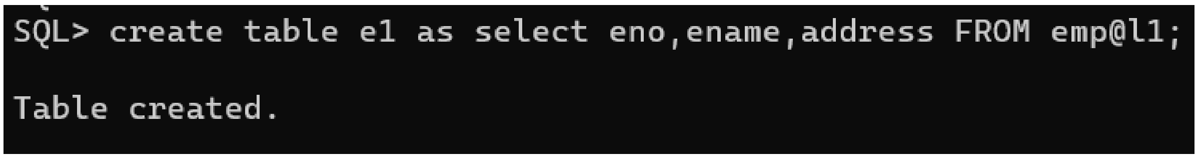
```
SQL> connect user1@xe
Enter password:
Connected.
SQL>
```

Create database link to global database in order to access the data of global database table.
SQL> create database link l1 connect to system identified **by** system using 'xe';



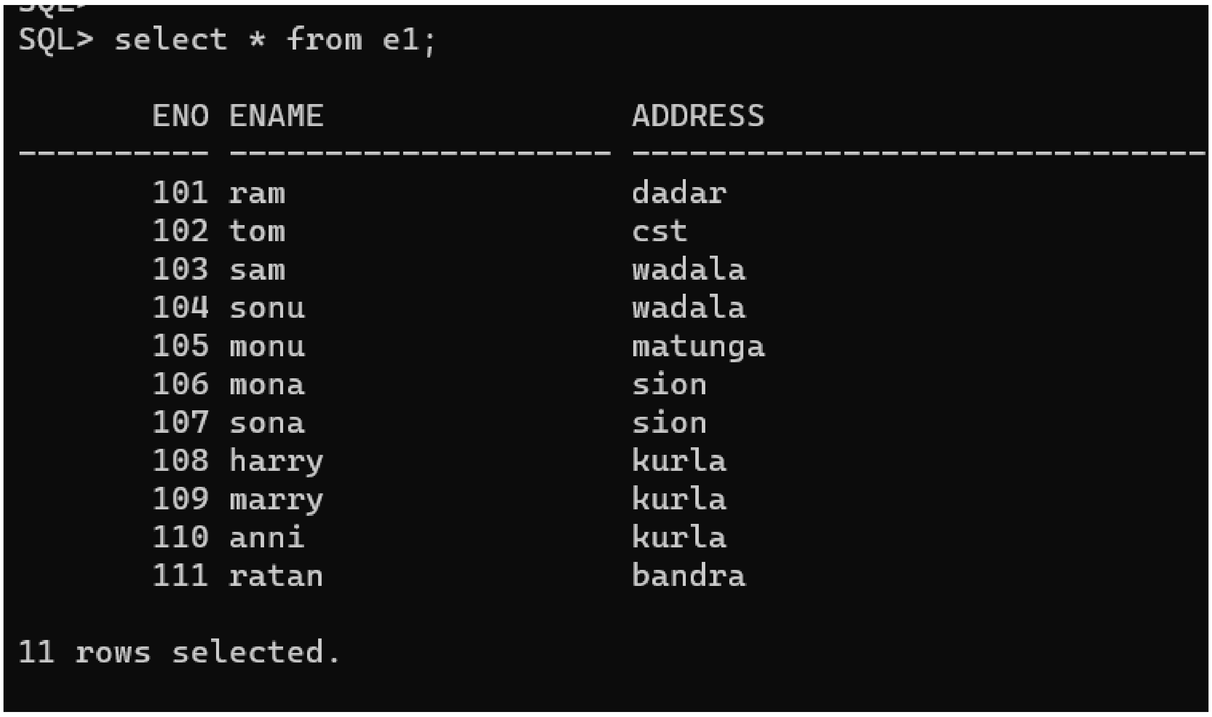
```
SQL> create database link l1 connect to system identified by "1234" using 'x
e';

Database link created.
```

Create table and insert the data from global database table
SQL> create table **e1** as select eno,ename,address from emp**@11**;



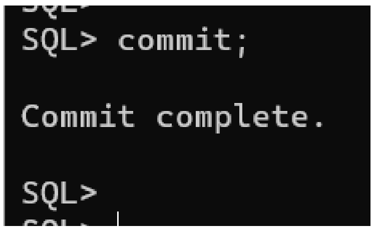
```
SQL> create table e1 as select eno,ename,address FROM emp@l1;

Table created.
```

View the inserted data into the created table.
SQL> select * from e1;



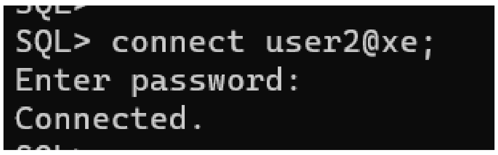
```
SQL> select * from e1;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
       101 ram                  dadar
       102 tom                  cst
       103 sam                  wadala
       104 sonu                 wadala
       105 monu                 matunga
       106 mona                 sion
       107 sona                 sion
       108 harry                kurla
       109 marry                kurla
       110 anni                 kurla
       111 ratan                bandra

11 rows selected.
```

Perform commit.
SQL> **commit**;



```
SQL> commit;

Commit complete.

SQL>
```

Now open another sqlplus and login with user2 id and password.
SQL> connect user2@xe

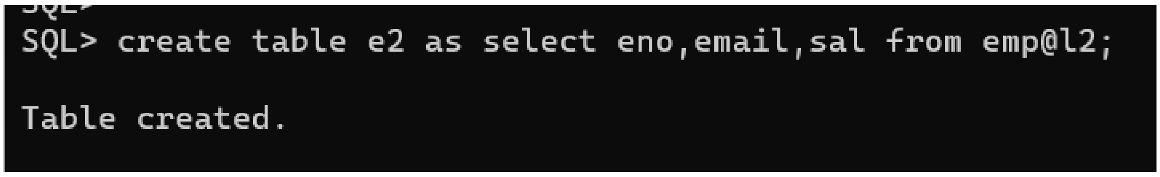


```
SQL> connect user2@xe;
Enter password:
Connected.
```

Create database link to global database in order to access the data of global database table.
SQL> create database link 122 connect to **system** identified by system using'xe';

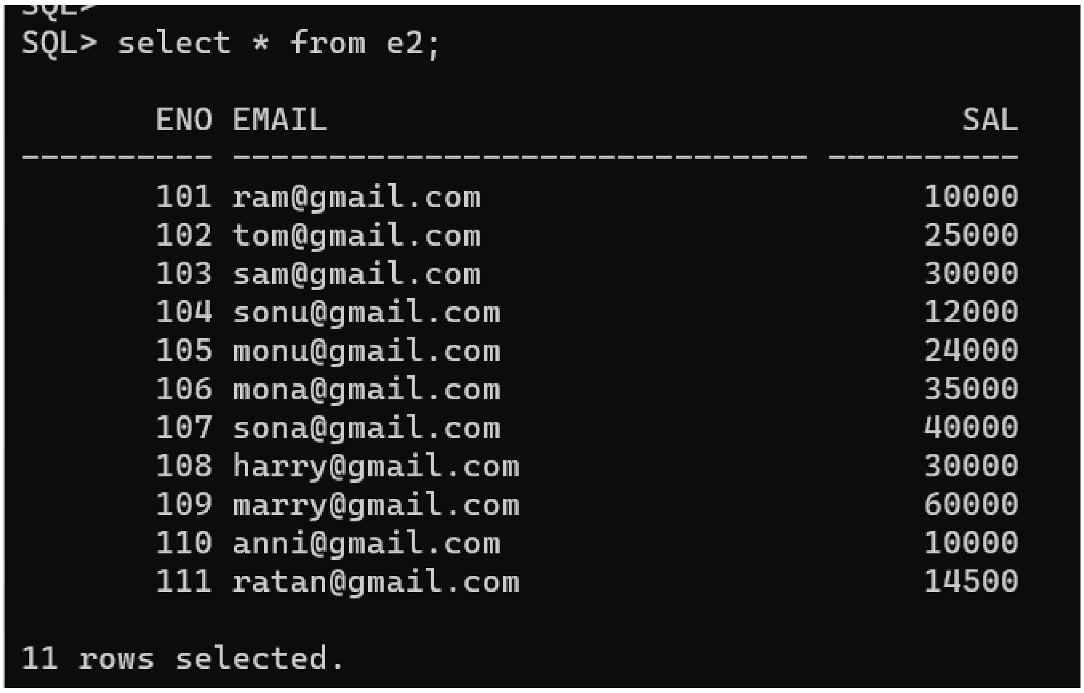
Create table and insert the data from global database table

SQL> create table e2 as select eno,email,sal from emp@**122**;



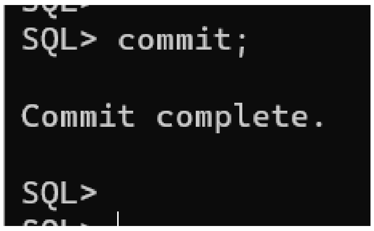
```
SQL> create table e2 as select eno,email,sal from emp@l2;

Table created.
```

View the inserted data into the created table.
SQL> select * from e2;



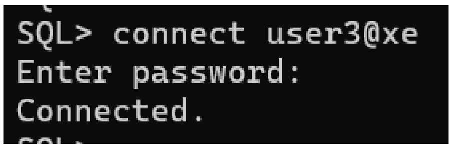
```
SQL> select * from e2;

       ENO EMAIL                                 SAL
---------- ------------------------------ ----------
       101 ram@gmail.com                       10000
       102 tom@gmail.com                       25000
       103 sam@gmail.com                       30000
       104 sonu@gmail.com                      12000
       105 monu@gmail.com                      24000
       106 mona@gmail.com                      35000
       107 sona@gmail.com                      40000
       108 harry@gmail.com                     30000
       109 marry@gmail.com                     60000
       110 anni@gmail.com                      10000
       111 ratan@gmail.com                     14500

11 rows selected.
```

Perform the commit.
SQL> commit;



```
SQL> commit;

Commit complete.

SQL>
```

Now open another sqlplus and login with user3 id and password.
SQL> connect user3@**xe**



```
SQL> connect user3@xe
Enter password:
Connected.
```

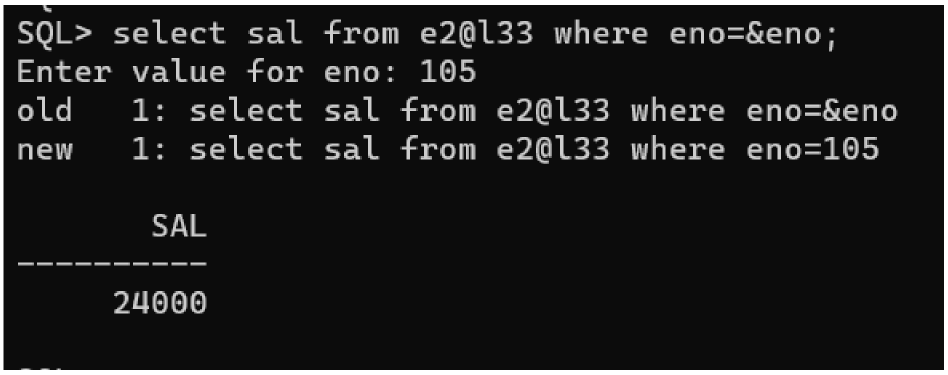
Create database link for **tables here in order to** access **various types of** data based **on give question**

SQL> create database link 111 connect to user1 identified by u1 using'xe';

SQL> create database link **122** connect to user2 identified by **u2** using **'xe';**

(i) Find the salary of an employee where employee number is known.

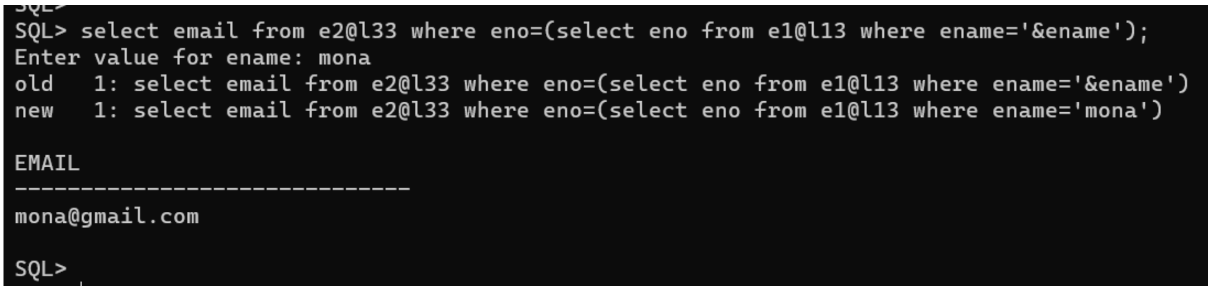
SQL> select sal from **e2**@122 **where** eno=**&**eno**;**



```
SQL> select sal from e2@l33 where eno=&eno;
Enter value for eno: 105
old   1: select sal from e2@l33 where eno=&eno
new   1: select sal from e2@l33 where eno=105

       SAL
----------
     24000
```

(ii) Find the Email where the employee name is known.

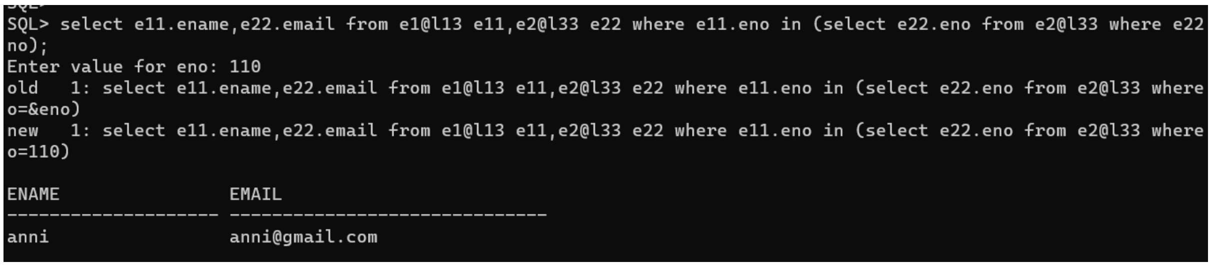
SQL> select email from **e2@122** where eno=(select eno from e1@111 where ename=**'&ename'**);



```
SQL> select email from e2@l33 where eno=(select eno from e1@l13 where ename='&ename');
Enter value for ename: mona
old   1: select email from e2@l33 where eno=(select eno from e1@l13 where ename='&ename')
new   1: select email from e2@l33 where eno=(select eno from e1@l13 where ename='mona')

EMAIL
------------------------------
mona@gmail.com

SQL>
```

(iii) **Find the employee name and email where employee number is known**.

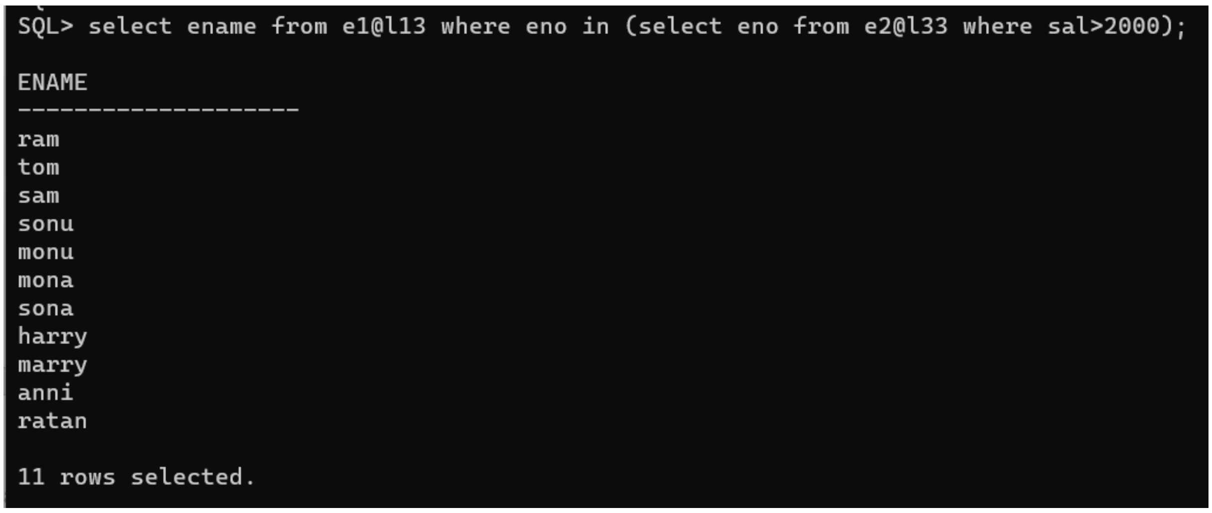
SQL> select e11.ename,e22.email **from e1@111** e11,**e2@l22 e22** where e11.eno in (select e22.eno from e2@122 where e22.eno=**&**eno);



```
SQL> select e11.ename,e22.email from e1@l13 e11,e2@l33 e22 where e11.eno in (select e22.eno from e2@l33 where e22
no);
Enter value for eno: 110
old   1: select e11.ename,e22.email from e1@l13 e11,e2@l33 e22 where e11.eno in (select e22.eno from e2@l33 where
o=&eno)
new   1: select e11.ename,e22.email from e1@l13 e11,e2@l33 e22 where e11.eno in (select e22.eno from e2@l33 where
o=110)

ENAME                EMAIL
-------------------- ------------------------------
anni                 anni@gmail.com
```

(iv) Find the employee name whose salary is > 2000.

SQL> select ename from e1@111 where eno in (select eno **from e2@122** where sal>**2000)**;



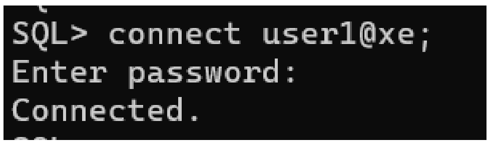
```
SQL> select ename from e1@l13 where eno in (select eno from e2@l33 where sal>2000);

ENAME
--------------------
ram
tom
sam
sonu
monu
mona
sona
harry
marry
anni
ratan

11 rows selected.
```

PRACTICAL 07

Aim: Distributed Database.
Perform Replication
Fragmentation

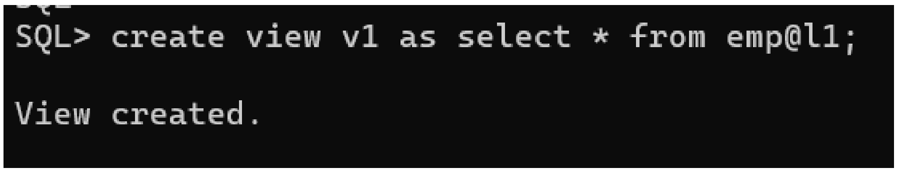
Create a global conceptual schema Emp(Eno;Ename;Address;Email;Salary) and insert 10 records.
Store the replication of Emp into two different nodes.
THEOR
Y:
What is Database
Replication?
Data Replication is the process of storing data in more than one site or node. It is useful in improving the availability **of** data. It **is** simply copying **data from** a database **from** one server to **another** server **so that all** the **users** can share **the** same **data** without any inconsistency. **The** result **is** a distributed database **in** which users can **access** data relevant **to** their tasks without interfering **with** the work **of** others.
Data replication encompasses duplication **of** transactions on an **ongoing basis,** so **that** the replicate **is** in **a** consistently updated state and synchronized with **the source. However, in** data replication data is available at different **locations,** but a particular relation has to reside at only one **location**.
There can **be full** replication**,** in which **the** whole database **is** stored at **every site**. **There** can **also** be partial replication**,** in which some **frequently** used fragment of **the** database are replicated and others are not replicated.
Advantages of full **replication**
High Availability of Data.
✔ Improves the performance for retrieval **of global** queries as the **result** can be obtained locally **from** any **of the** local site.
✓ Faster execution of Queries.
Disadvantages of full replication
Concurrency **is** difficult to **achieve** in full **replication**.
✓ Slow update process **as** a single update must be performed **at** different databases to keep the **copies** consistent.
Fire the following queries:
a) Find **the** salary of all employees.
b) Find the email of all employees **where** salary=15,000.
c) Find the employee name **& email where** employee number **is** known.
d) Find the employee name & address where employee number is known.

Now open another sqlplus and login with user1 id and password.
SQL> **connect** user1@xe


```
SQL> connect user1@xe;
Enter password:
Connected.
```

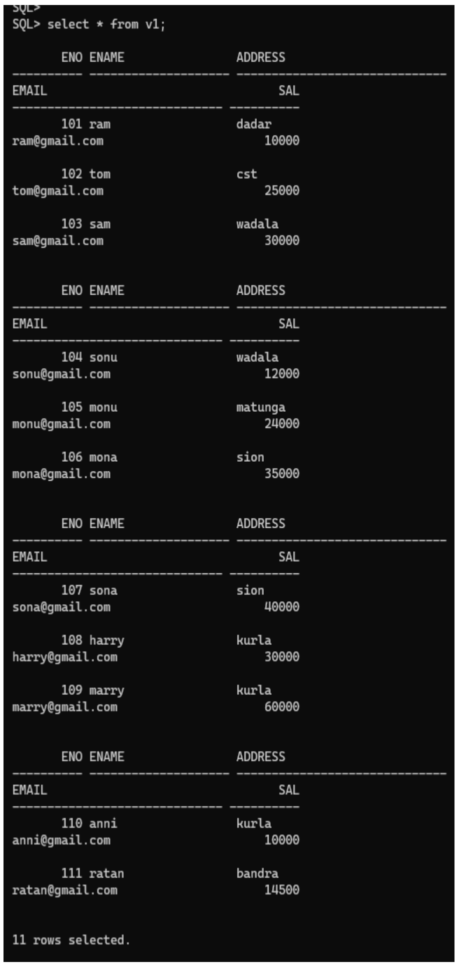
Create database link to global database in order to access the data of global database **table**.

SQL> create database link 11 **connect** to system identified by system using '**xe**';
Create view of global database
SQL> create view v1 as select * from emp@l1;



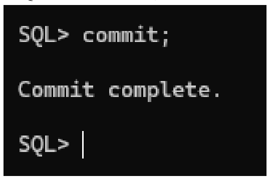
```
SQL> create view v1 as select * from emp@l1;

View created.
```

Check if view has been created successfully.
SQL> select * **from** v1;



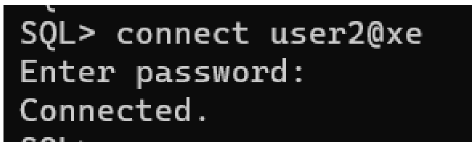
```
SQL> select * from v1;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       101 ram                  dadar
ram@gmail.com                       10000

       102 tom                  cst
tom@gmail.com                       25000

       103 sam                  wadala
sam@gmail.com                       30000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       104 sonu                 wadala
sonu@gmail.com                      12000

       105 monu                 matunga
monu@gmail.com                      24000

       106 mona                 sion
mona@gmail.com                      35000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       107 sona                 sion
sona@gmail.com                      40000

       108 harry                kurla
harry@gmail.com                     30000

       109 marry                kurla
marry@gmail.com                     60000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       110 anni                 kurla
anni@gmail.com                      10000

       111 ratan                bandra
ratan@gmail.com                     14500


11 rows selected.
```

Perform the commit.
SQL> commit;



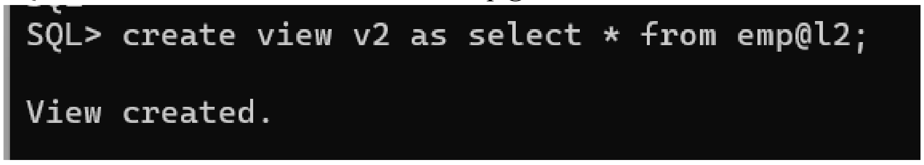
```
SQL> commit;

Commit complete.

SQL>
```

Now open another sqlplus and login with user2 id and password.
SQL> connect user2@xe



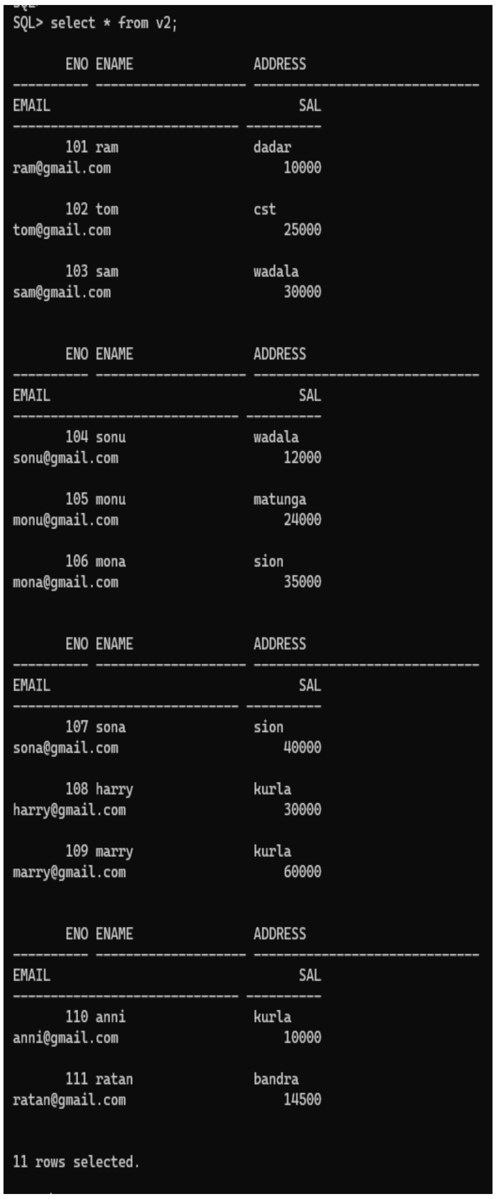
```
SQL> connect user2@xe
Enter password:
Connected.
```

Create database link to global database in order to access **the data of global database** **table**.
SQL> create database link **122** connect to system identified by system using'xe';
Create view **of global database**
SQL> create view v2 **as** select * from emp@122;



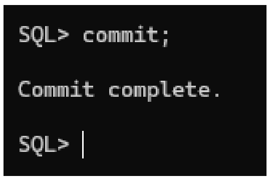
```
SQL> create view v2 as select * from emp@l2;

View created.
```

Check if view has been created successfully.
SQL> **select** * from **v2**;



```
SQL> select * from v2;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       101 ram                  dadar
ram@gmail.com                       10000

       102 tom                  cst
tom@gmail.com                       25000

       103 sam                  wadala
sam@gmail.com                       30000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       104 sonu                 wadala
sonu@gmail.com                      12000

       105 monu                 matunga
monu@gmail.com                      24000

       106 mona                 sion
mona@gmail.com                      35000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       107 sona                 sion
sona@gmail.com                      40000

       108 harry                kurla
harry@gmail.com                     30000

       109 marry                kurla
marry@gmail.com                     60000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       110 anni                 kurla
anni@gmail.com                      10000

       111 ratan                bandra
ratan@gmail.com                     14500


11 rows selected.
```

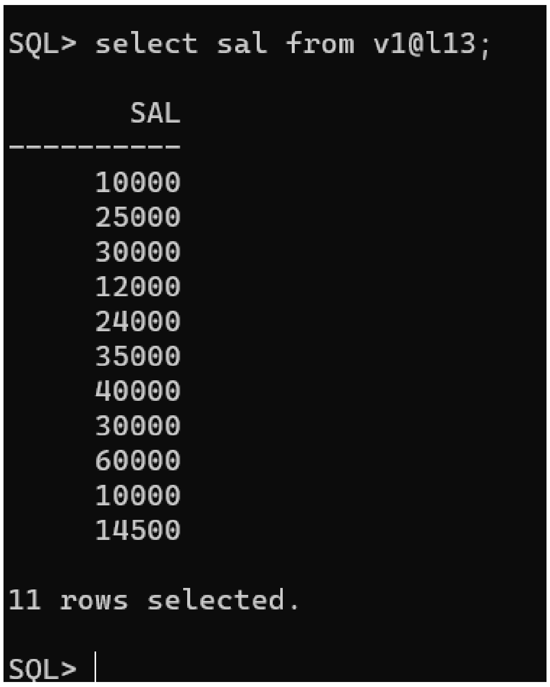
Perform the commit
SQL> commit;



```
SQL> commit;

Commit complete.

SQL> |
```

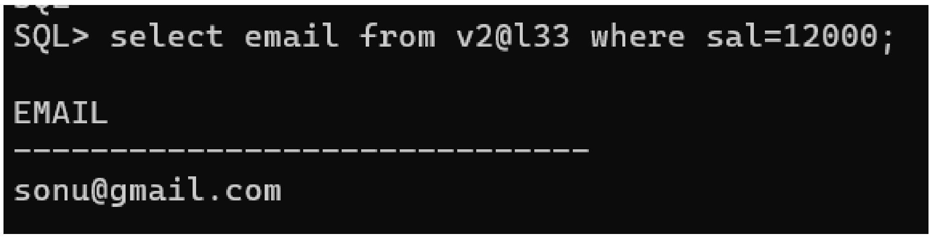
Now open another sqlplus and login with user3 id and password.
SQL> **connect** user3@xe

Create database link for tables here in order to access **various types of data based on give question**
SQL> create **database** link 111 connect to user1 identified **by u1** using'xe';
SQL> create database link 122 connect **to** user2 identified **by** u2 **using** 'xe'**;**
(i) Find the salary of all **employees**.
SQL> select sal from v1@111;



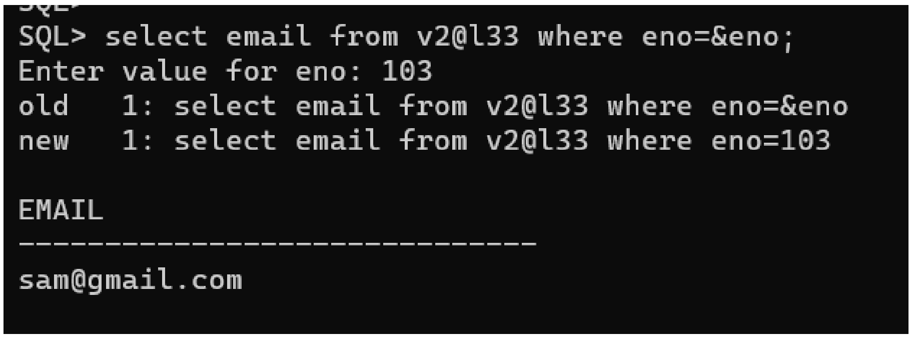
```
SQL> select sal from v1@l13;

       SAL
----------
     10000
     25000
     30000
     12000
     24000
     35000
     40000
     30000
     60000
     10000
     14500

11 rows selected.

SQL> |
```

(ii) Find the email of all employees where salary=12000
SQL> **select** email **from v2**@122 where sal=12000;



```
SQL> select email from v2@l33 where sal=12000;

EMAIL
------------------------------
sonu@gmail.com
```

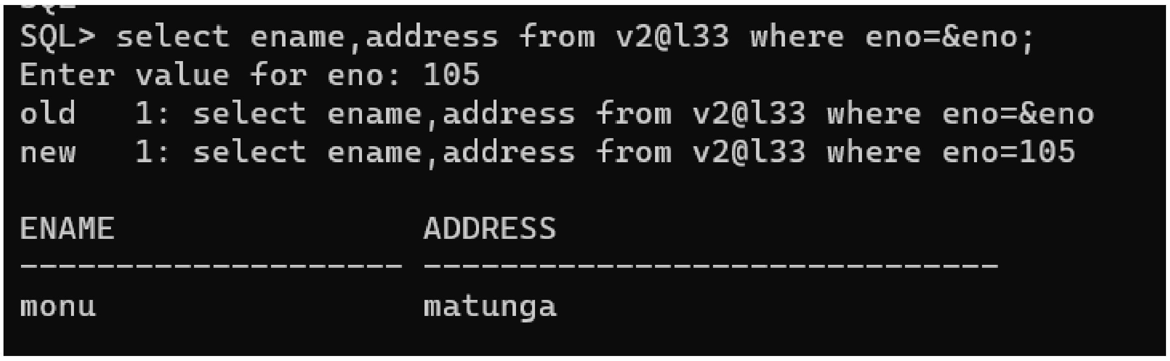
(iii) Find the employee name & email where employee number is known.
SQL> **select** ename,email from v1@111 where eno=**&**eno**;**



```
SQL> select email from v2@l33 where eno=&eno;
Enter value for eno: 103
old   1: select email from v2@l33 where eno=&eno
new   1: select email from v2@l33 where eno=103

EMAIL
------------------------------
sam@gmail.com
```

(iv) Find the employee name & address where employee number is known

SQL> select ename**,**address from **v2@122** where eno=&eno;



```
SQL> select ename,address from v2@l33 where eno=&eno;
Enter value for eno: 105
old   1: select ename,address from v2@l33 where eno=&eno
new   1: select ename,address from v2@l33 where eno=105

ENAME                ADDRESS
-------------------- ------------------------------
monu                 matunga
```

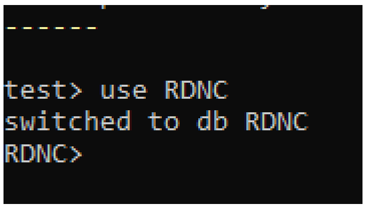
Practical 08

Aim: NoSQL Databases a) MongoDB CRUD b) Import CSV Data in MongoDB c) CouchDB CRUD d) Redis CRUD

• MongoDB database
Performing CRUD in MongoDB

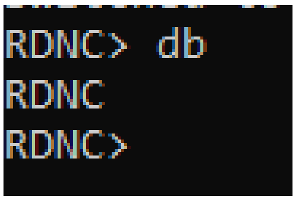
Step 1: To Create Database in MongoDB.

Type Command: use RDNC

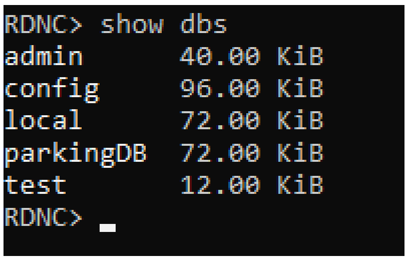

```
------

test> use RDNC
switched to db RDNC
RDNC>
```

Step 2: To display the name of current database:

Type Command: db


```
RDNC> db
RDNC
RDNC>
```

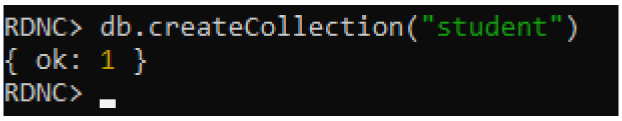
Step 3: To list down all the databases, use the command **show dbs**. This command lists down all the databases and their size on the disk


```
RDNC> show dbs
admin      40.00 KiB
config     96.00 KiB
local      72.00 KiB
parkingDB  72.00 KiB
test       12.00 KiB
RDNC>
```

Since our database has no collection inside it. Our database is not listed

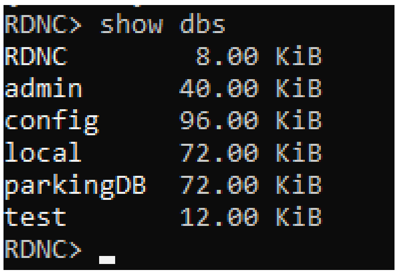
Step 4: To create a collection in the database.

Type command db.createCollection("Student")



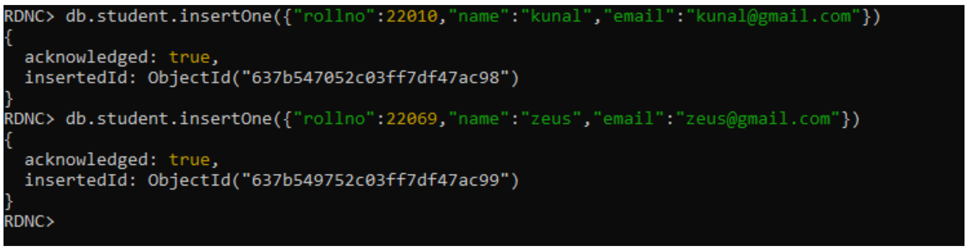
```
RDNC> db.createCollection("student")
{ ok: 1 }
RDNC> _
```

List all the databases again. Now, you will see that our database has also been shown here. Before it was not there.



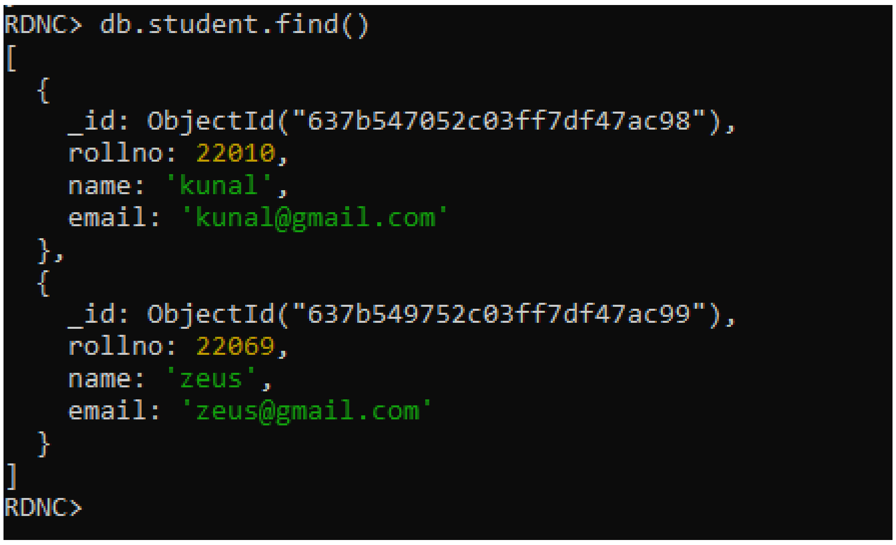
```
RDNC> show dbs
RDNC         8.00 KiB
admin       40.00 KiB
config      96.00 KiB
local       72.00 KiB
parkingDB   72.00 KiB
test        12.00 KiB
RDNC> _
```

Step 5: Insert data to the collection Student



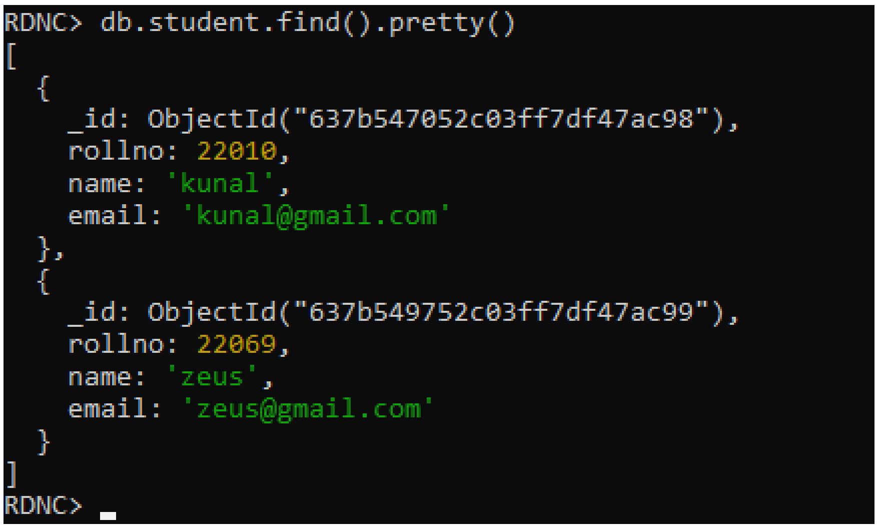
```
RDNC> db.student.insertOne({"rollno":22010,"name":"kunal","email":"kunal@gmail.com"})
{
  acknowledged: true,
  insertedId: ObjectId("637b547052c03ff7df47ac98")
}
RDNC> db.student.insertOne({"rollno":22069,"name":"zeus","email":"zeus@gmail.com"})
{
  acknowledged: true,
  insertedId: ObjectId("637b549752c03ff7df47ac99")
}
RDNC>
```

Step 6: Find the data in Student



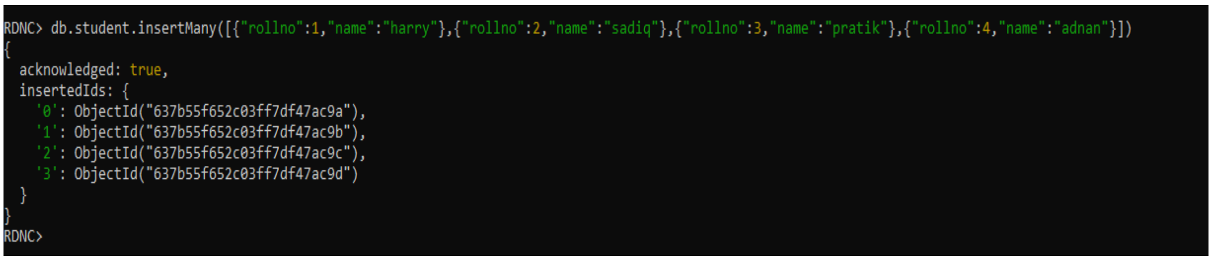
```
RDNC> db.student.find()
[
  {
    _id: ObjectId("637b547052c03ff7df47ac98"),
    rollno: 22010,
    name: 'kunal',
    email: 'kunal@gmail.com'
  },
  {
    _id: ObjectId("637b549752c03ff7df47ac99"),
    rollno: 22069,
    name: 'zeus',
    email: 'zeus@gmail.com'
  }
]
RDNC>
```

Another way of printing data is buy using pretty() function



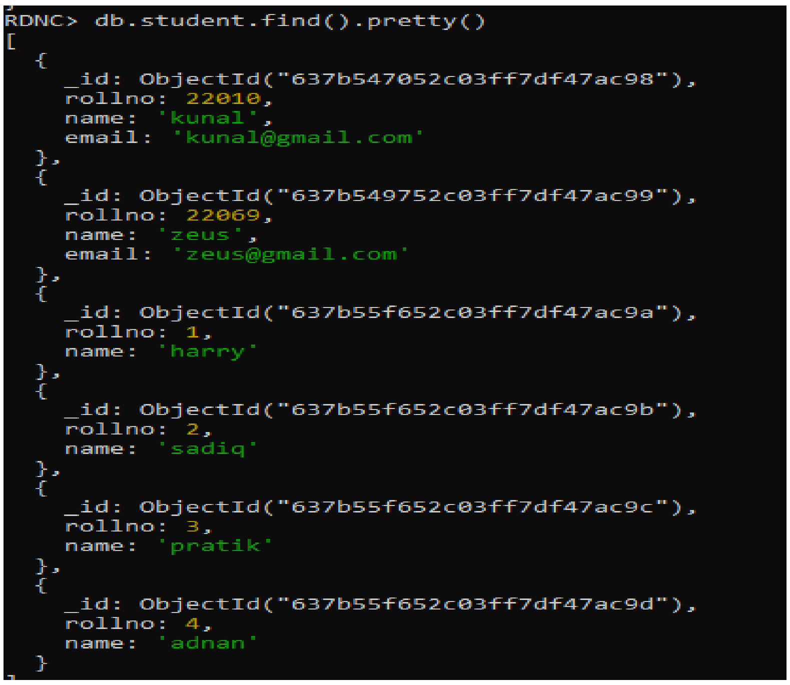
```
RDNC> db.student.find().pretty()
[
  {
    _id: ObjectId("637b547052c03ff7df47ac98"),
    rollno: 22010,
    name: 'kunal',
    email: 'kunal@gmail.com'
  },
  {
    _id: ObjectId("637b549752c03ff7df47ac99"),
    rollno: 22069,
    name: 'zeus',
    email: 'zeus@gmail.com'
  }
]
RDNC> _
```

Step 7: Insert data in the collection with insertMany() method



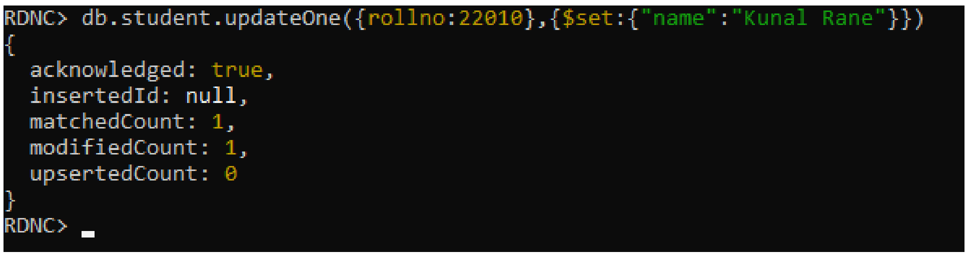
```
RDNC> db.student.insertMany([{"rollno":1,"name":"harry"},{"rollno":2,"name":"sadiq"},{"rollno":3,"name":"pratik"},{"rollno":4,"name":"adnan"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("637b55f652c03ff7df47ac9a"),
    '1': ObjectId("637b55f652c03ff7df47ac9b"),
    '2': ObjectId("637b55f652c03ff7df47ac9c"),
    '3': ObjectId("637b55f652c03ff7df47ac9d")
  }
}
RDNC>
```

Display the inserted data in the collection.



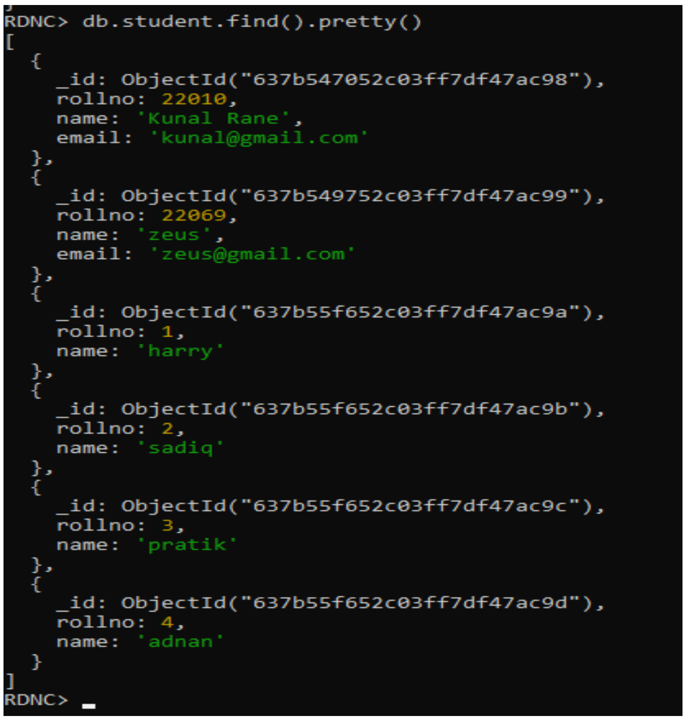
```
RDNC> db.student.find().pretty()
[
  {
    _id: ObjectId("637b547052c03ff7df47ac98"),
    rollno: 22010,
    name: 'kunal',
    email: 'kunal@gmail.com'
  },
  {
    _id: ObjectId("637b549752c03ff7df47ac99"),
    rollno: 22069,
    name: 'zeus',
    email: 'zeus@gmail.com'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9a"),
    rollno: 1,
    name: 'harry'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9b"),
    rollno: 2,
    name: 'sadiq'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9c"),
    rollno: 3,
    name: 'pratik'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9d"),
    rollno: 4,
    name: 'adnan'
  }
```

Step 8: Update the collection students



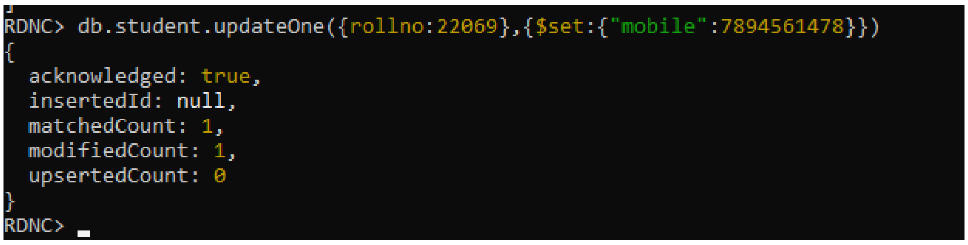
```
RDNC> db.student.updateOne({rollno:22010},{$set:{"name":"Kunal Rane"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
RDNC>
```

Print the data in the collection to check if the data has been updated or not.



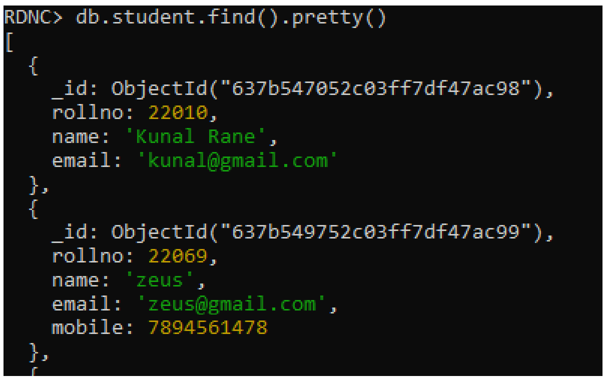
```
RDNC> db.student.find().pretty()
[
  {
    _id: ObjectId("637b547052c03ff7df47ac98"),
    rollno: 22010,
    name: 'Kunal Rane',
    email: 'kunal@gmail.com'
  },
  {
    _id: ObjectId("637b549752c03ff7df47ac99"),
    rollno: 22069,
    name: 'zeus',
    email: 'zeus@gmail.com'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9a"),
    rollno: 1,
    name: 'harry'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9b"),
    rollno: 2,
    name: 'sadiq'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9c"),
    rollno: 3,
    name: 'pratik'
  },
  {
    _id: ObjectId("637b55f652c03ff7df47ac9d"),
    rollno: 4,
    name: 'adnan'
  }
]
RDNC> _
```

Step 9: You can also add the extra data in the existing document with the same: updateOne() method.



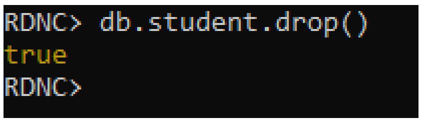
```
RDNC> db.student.updateOne({rollno:22069},{$set:{"mobile":7894561478}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
RDNC> _
```

Check if the data has been added or not



```
RDNC> db.student.find().pretty()
[
  {
    _id: ObjectId("637b547052c03ff7df47ac98"),
    rollno: 22010,
    name: 'Kunal Rane',
    email: 'kunal@gmail.com'
  },
  {
    _id: ObjectId("637b549752c03ff7df47ac99"),
    rollno: 22069,
    name: 'zeus',
    email: 'zeus@gmail.com',
    mobile: 7894561478
  },
```

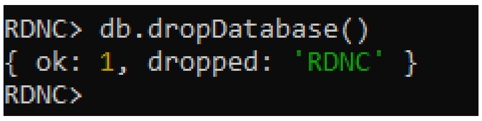
Step 10: To drop the collection in the database type: db.student.drop(). Then type show collections to confirm if our Student collection is dropped or not.



```
RDNC> db.student.drop()
true
RDNC>
```

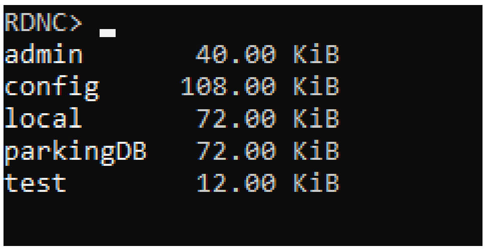
Its Dropped

Step 11: Drop the current database.



```
RDNC> db.dropDatabase()
{ ok: 1, dropped: 'RDNC' }
RDNC>
```

Check if database has been dropped successfully

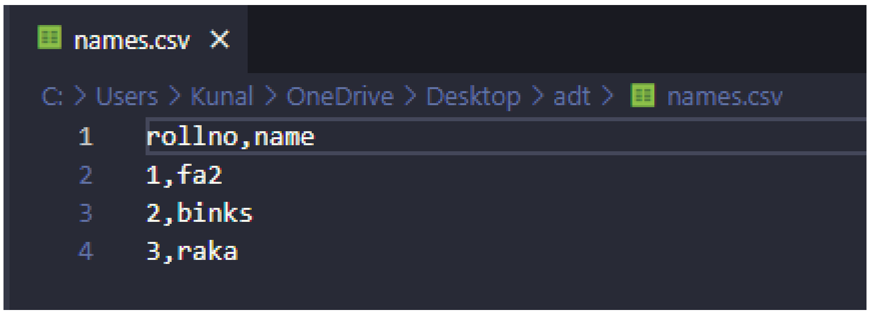


```
RDNC> _
admin        40.00 KiB
config      108.00 KiB
local        72.00 KiB
parkingDB    72.00 KiB
test         12.00 KiB
```

Our Database Dropped Successfully.

B) Import CSV file in MongoDB

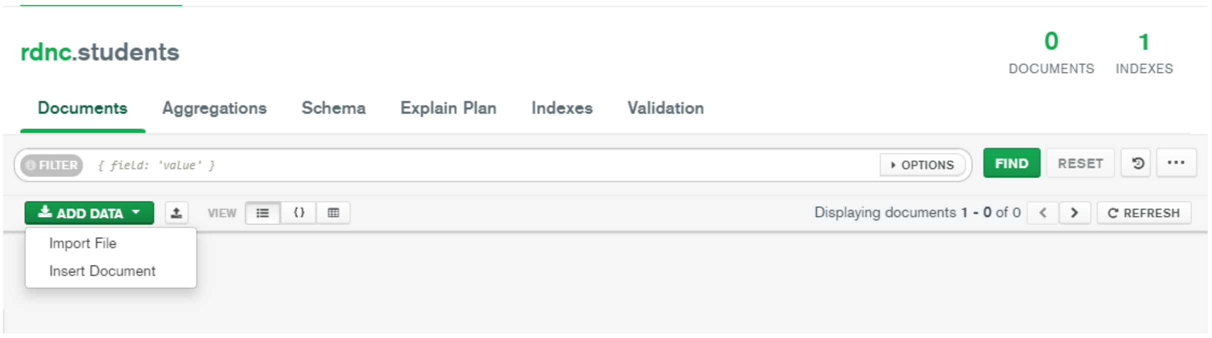
Step 1: First create a file in notepad using specific comma format and save that file as .csv extension



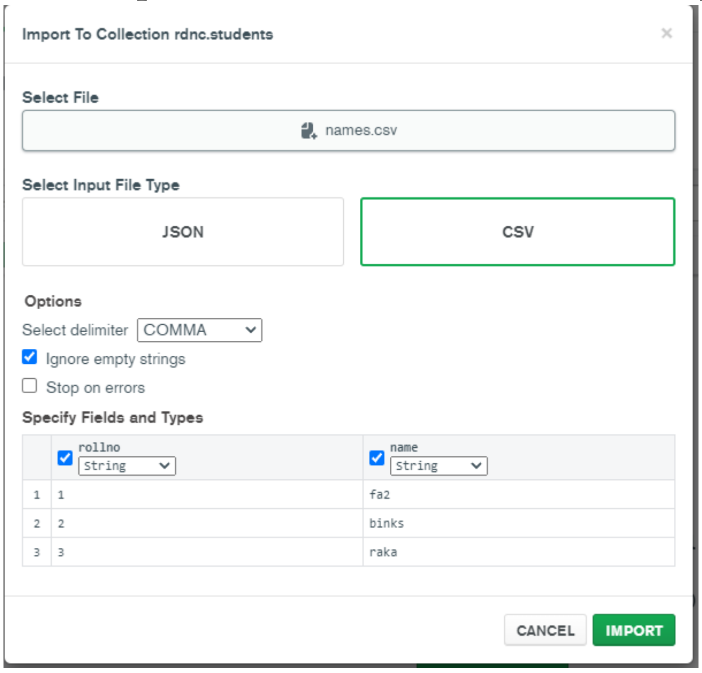
```
rollno,name
1,fa2
2,binks
3,raka
```

Step 2: Open Mongo DB Compass Create a database and then click the

Add Data drop down button and click Import File

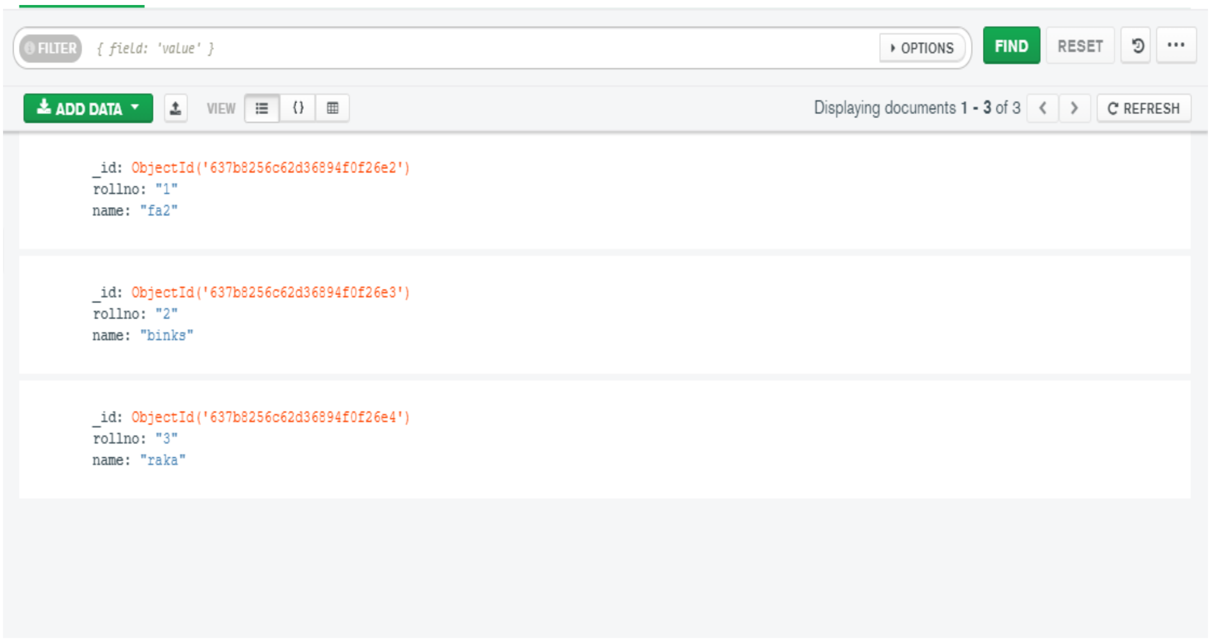


Step 3: After Selecting Import File option, select your saved file and Select input field as CSV and then click Import



	✓ rollno String	✓ name String
1	1	fa2
2	2	binks
3	3	raka

After Clicking Import You can see all the data from the csv file has been added to mongoDB Compass.



```
_id: ObjectId('637b8256c62d36894f0f26e2')
rollno: "1"
name: "fa2"

_id: ObjectId('637b8256c62d36894f0f26e3')
rollno: "2"
name: "binks"

_id: ObjectId('637b8256c62d36894f0f26e4')
rollno: "3"
name: "raka"
```

AIM: Perform the CRUD Operations using

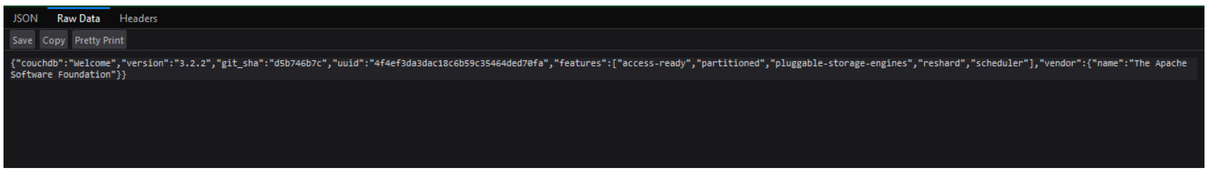
- CouchDB database

Performing CRUD in CouchDB

Step 1: Install CouchDB.

Step 2: Go to the browser and type: http://localhost:5984/

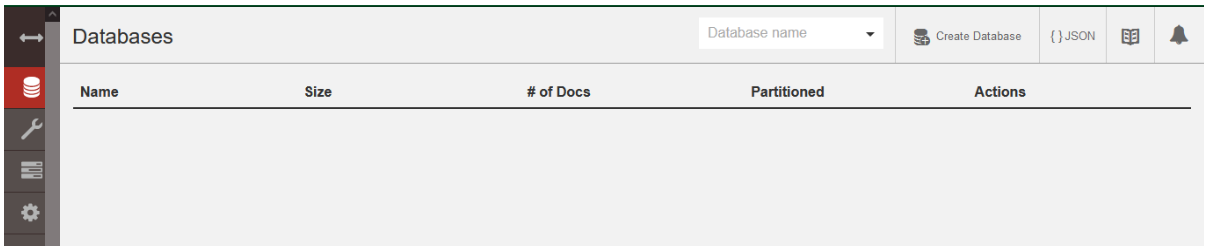
If you get to this screen this means that CouchDB is successfully install and running.



```
{"couchdb":"Welcome","version":"3.2.2","git_sha":"d5b746b7c","uuid":"4f4ef3da3dac18c6b59c35464ded70fa","features":["access-ready","partitioned","pluggable-storage-engines","reshard","scheduler"],"vendor":{"name":"The Apache Software Foundation"}}
```

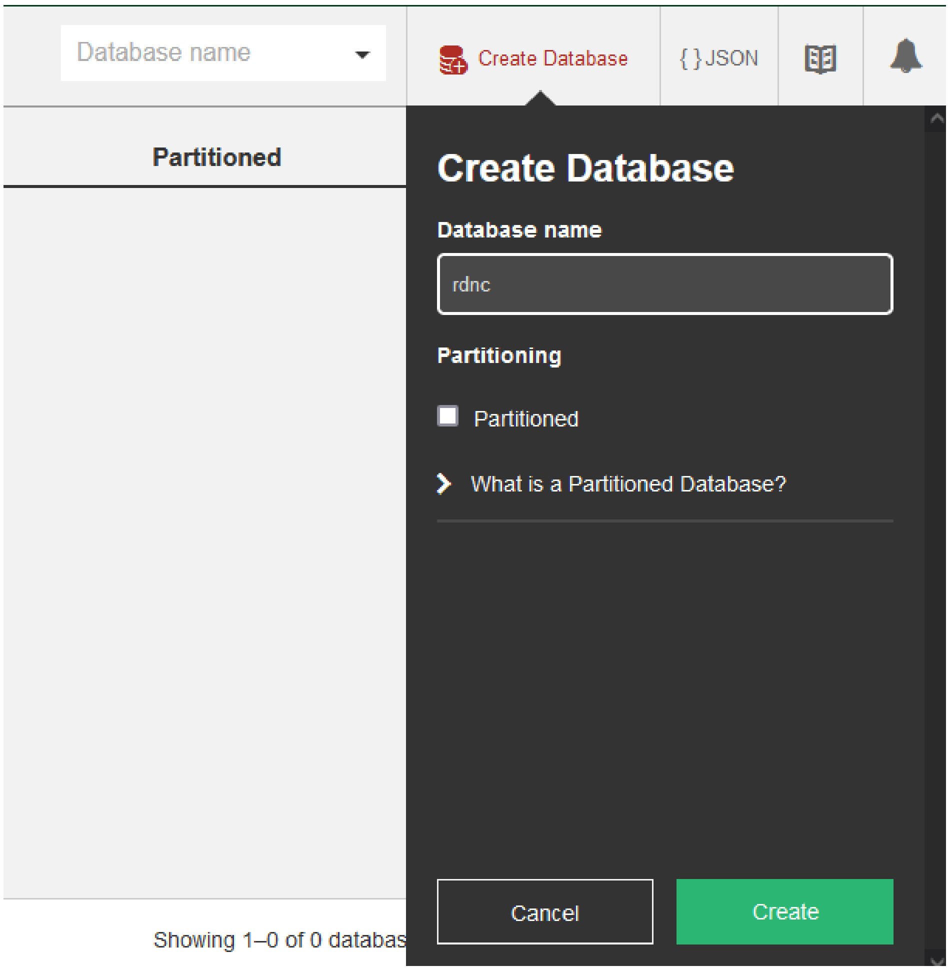
Step 3: Now type http://localhost:5984/_utils and enter your username and password.

After entering the credential, you will see this screen.



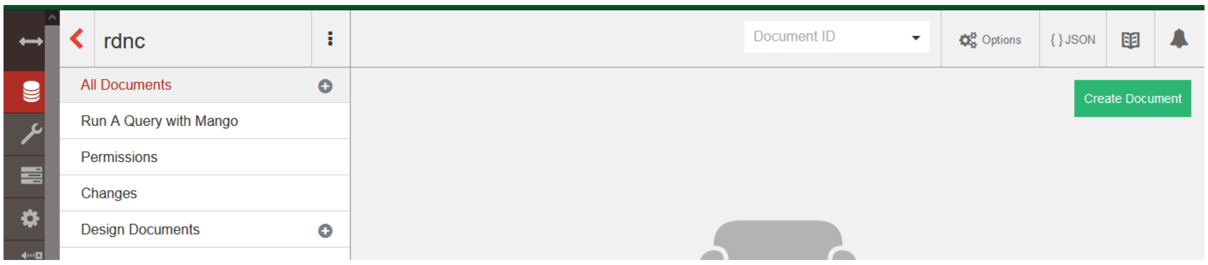
Step 4: At the top right corner there is an option of create database.

Click on the button and it will show the database creation option as given below. Type the name of database you want and then click create button



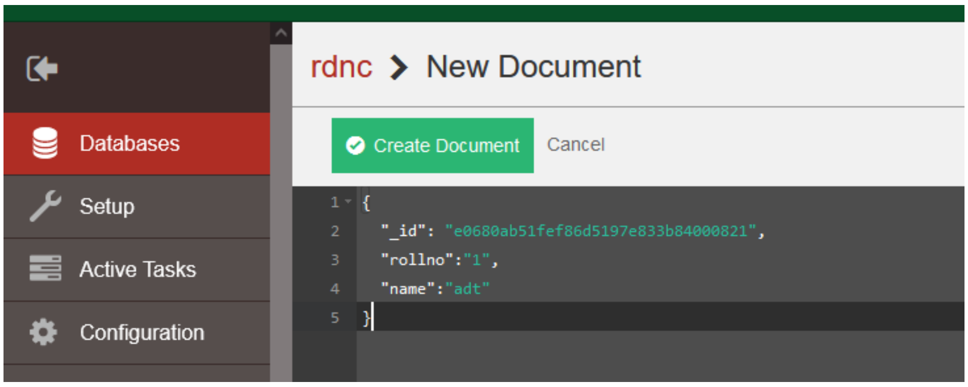
Database Name Should be in small case

Step 5: After the creation of the database the screen Will look like this. At the top right corner there is an option of create document.



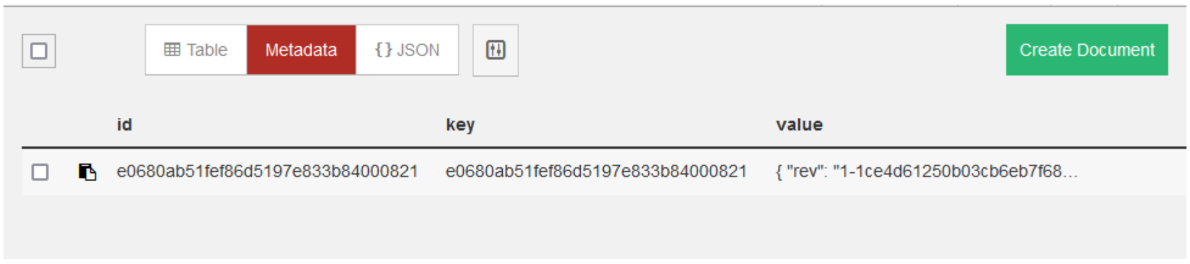
Step 6: Click on the create document button. And, the screen will appear as shown below.

Enter the data of your choice and click on create document button again



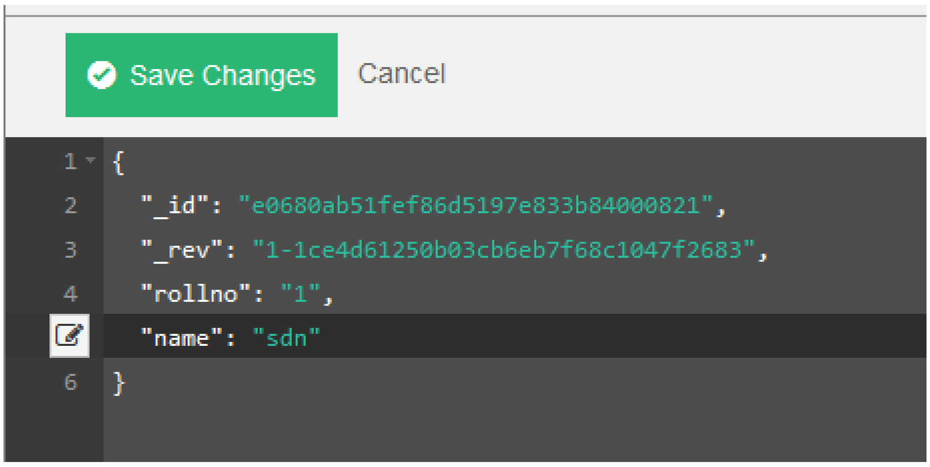
```
{
  "_id": "e0680ab51fef86d5197e833b84000821",
  "rollno":"1",
  "name":"adt"
}
```

Once the data has been created, you will see your inserted data as shown below



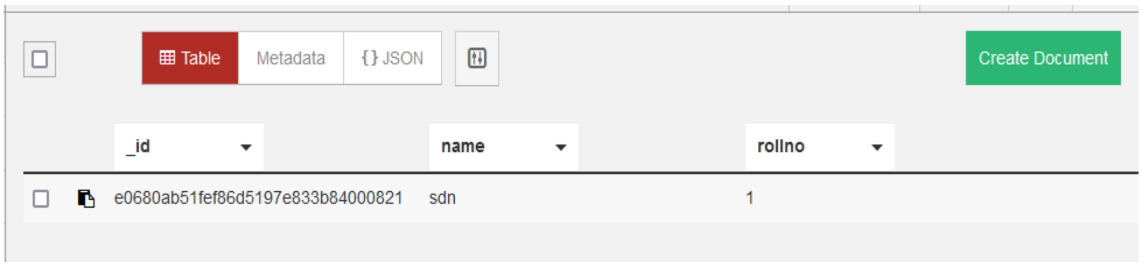
id	key	value
e0680ab51fef86d5197e833b84000821	e0680ab51fef86d5197e833b84000821	{ "rev": "1-1ce4d61250b03cb6eb7f68...

Step 7: You can update the data by clicking on the data and click on the save changes button on the top left corner and your data will be updated.



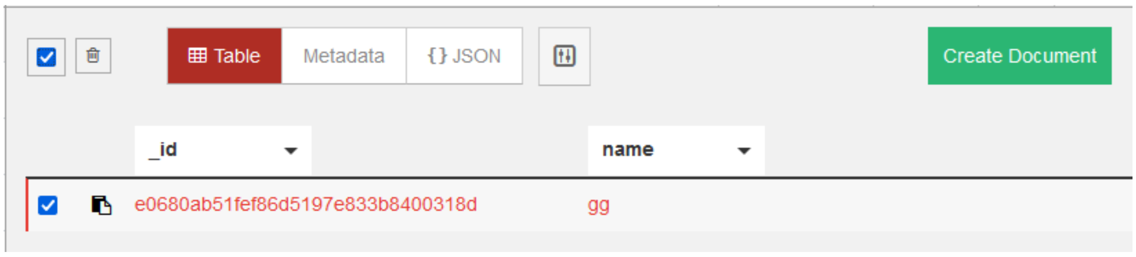
```
{
  "_id": "e0680ab51fef86d5197e833b84000821",
  "_rev": "1-1ce4d61250b03cb6eb7f68c1047f2683",
  "rollno": "1",
  "name": "sdn"
}
```

After updating you can view your data properly by click on the table and JSON view option given on top middle part.



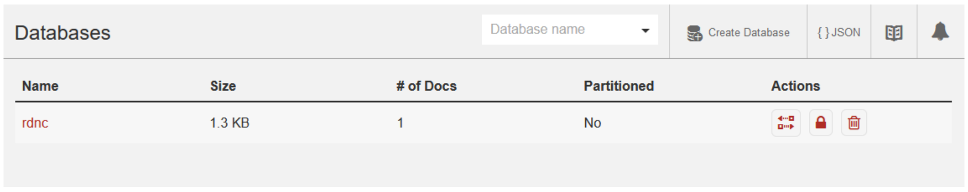
_id	name	rollno
e0680ab51fef86d5197e833b84000821	sdn	1

Step 8: After checking the check box of the row you can delete the data by clicking on the delete icon. When you click on delete icon. It will show you a prompt asking for confirmation. Once you confirm it your data will be deleted



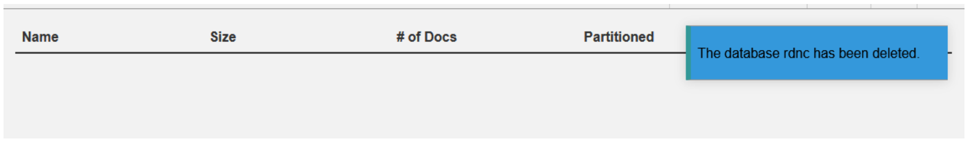
_id	name
e0680ab51fef86d5197e833b8400318d	gg

Step 9: Now this below picture shows the name of your database. You can delete your database by clicking on the delete icon



Name	Size	# of Docs	Partitioned	Actions
rdnc	1.3 KB	1	No	

Step 10: After clicking on the delete icon, you will see the prompt asking for the name of your database, once you type your database name and click on delete button, your database will be deleted. Below picture shows that there is no database in your databases list.

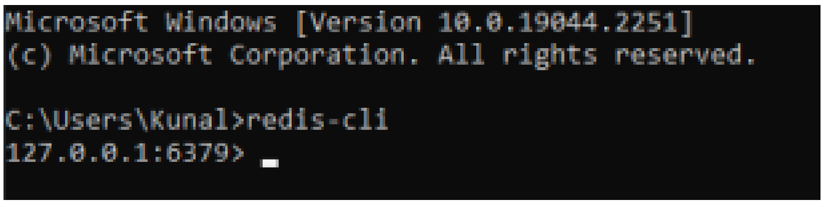


Name	Size	# of Docs	Partitioned

The database rdnc has been deleted.

Aim: Performing CRUD Operations using Redis Database

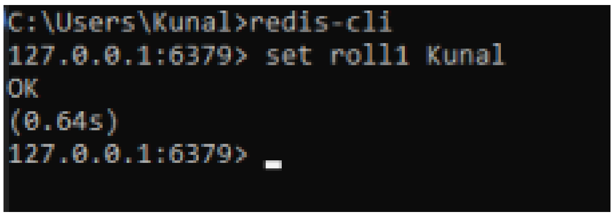
Step 1: Open Redis cli by using command redis-cli in command prompt



```
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kunal>redis-cli
127.0.0.1:6379> _
```

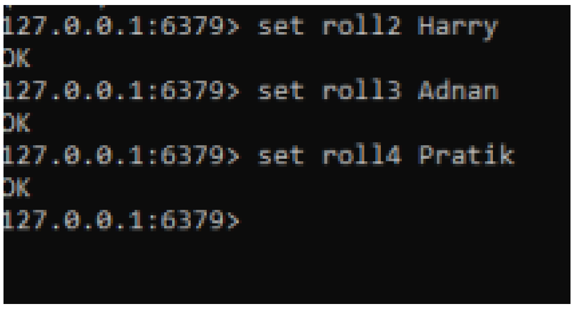
Step 2: Inserting values in Database

To insert values in database use command **set roll1 Kunal.**



```
C:\Users\Kunal>redis-cli
127.0.0.1:6379> set roll1 Kunal
OK
(0.64s)
127.0.0.1:6379> _
```

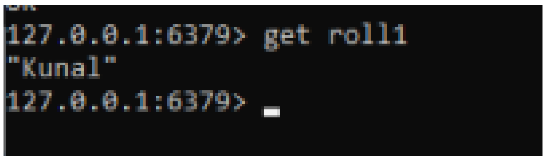
You can enter as many values as you want.



```
127.0.0.1:6379> set roll2 Harry
OK
127.0.0.1:6379> set roll3 Adnan
OK
127.0.0.1:6379> set roll4 Pratik
OK
127.0.0.1:6379>
```

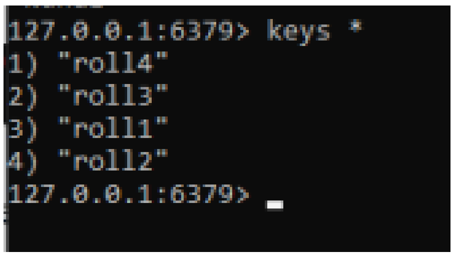
Step 2: View inserted values

To view single key value use command, get roll1.



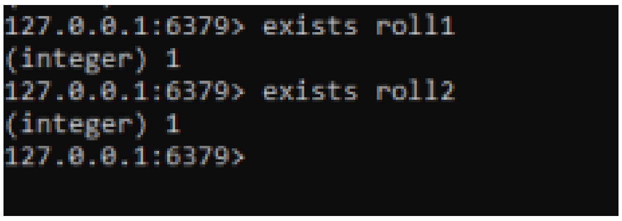
```
127.0.0.1:6379> get roll1
"Kunal"
127.0.0.1:6379> _
```

To get all the value use command, keys *



```
127.0.0.1:6379> keys *
1) "roll4"
2) "roll3"
3) "roll1"
4) "roll2"
127.0.0.1:6379> _
```

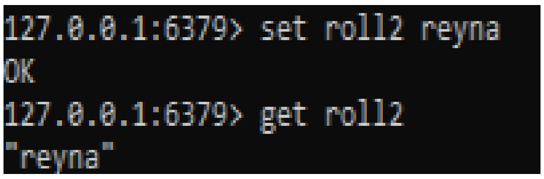
To see existing key value use command **exists roll1**



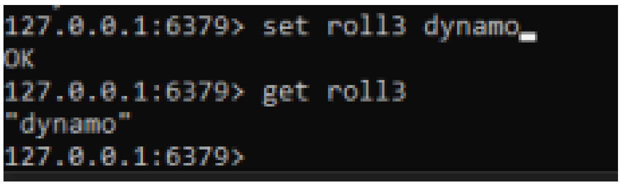
```
127.0.0.1:6379> exists roll1
(integer) 1
127.0.0.1:6379> exists roll2
(integer) 1
127.0.0.1:6379>
```

Step 3: Update values in database

To update values in database use command set roll2 Jack (replace your value with some new value)



```
127.0.0.1:6379> set roll2 reyna
OK
127.0.0.1:6379> get roll2
"reyna"
```

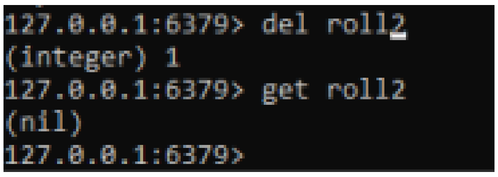


```
127.0.0.1:6379> set roll3 dynamo
OK
127.0.0.1:6379> get roll3
"dynamo"
127.0.0.1:6379>
```

Values has been updated.

Step 4: Delete values in Database

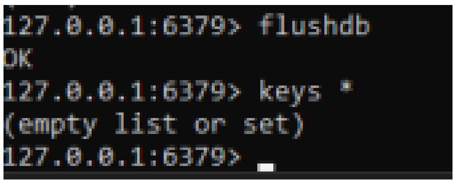
- To delete a single key value use command **del roll2**



```
127.0.0.1:6379> del roll2
(integer) 1
127.0.0.1:6379> get roll2
(nil)
127.0.0.1:6379>
```

Value has been deleted.

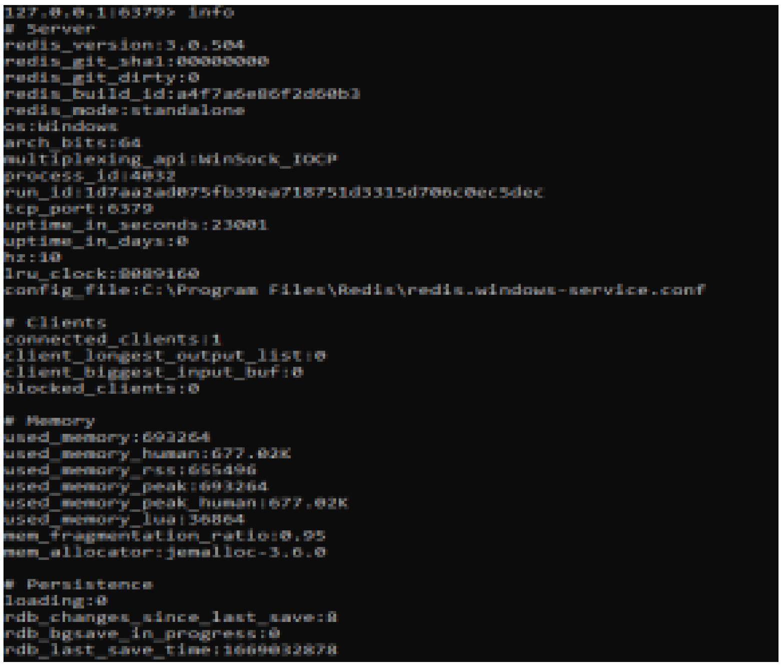
- To delete all key values from the database use command **flushdb**



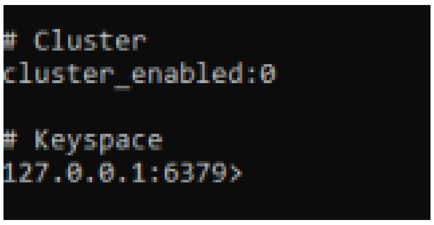
```
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379>
```

All Value has been deleted

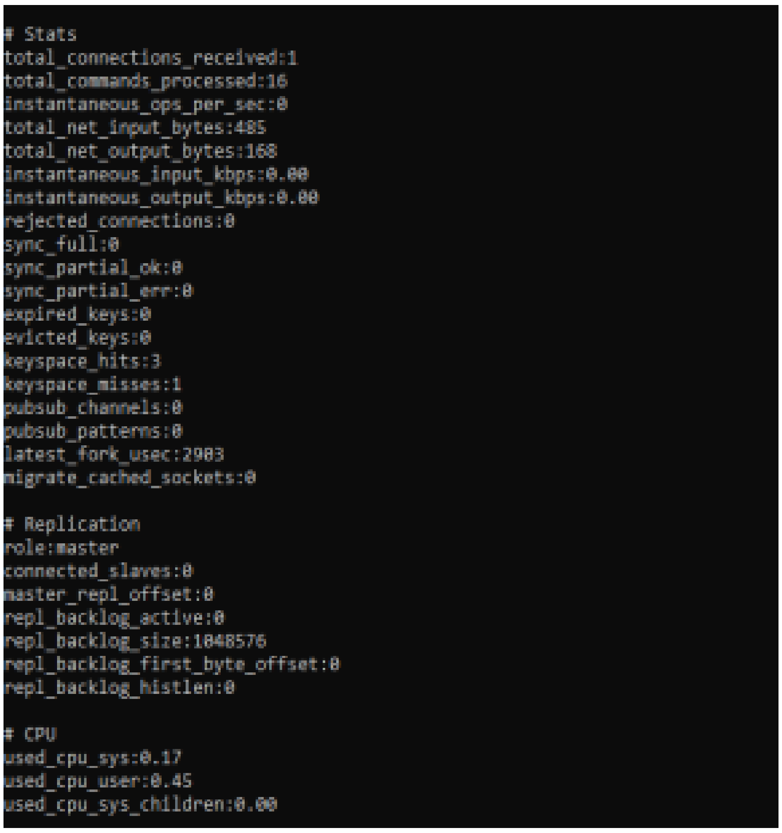
To see server information use command **info**



```
127.0.0.1:6379> info
# Server
redis_version:3.0.504
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:a4f7a6e86f2d60b3
redis_mode:standalone
os:Windows
arch_bits:64
multiplexing_api:WinSock_IOCP
process_id:4032
run_id:1d7aa2ad075fb39ea718751d3315d706c0ec5dec
tcp_port:6379
uptime_in_seconds:23001
uptime_in_days:0
hz:10
lru_clock:8089160
config_file:C:\Program Files\Redis\redis.windows-service.conf

# Clients
connected_clients:1
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0

# Memory
used_memory:693264
used_memory_human:677.02K
used_memory_rss:655496
used_memory_peak:693264
used_memory_peak_human:677.02K
used_memory_lua:36864
mem_fragmentation_ratio:0.95
mem_allocator:jemalloc-3.6.0

# Persistence
loading:0
rdb_changes_since_last_save:0
rdb_bgsave_in_progress:0
rdb_last_save_time:1669032878
```



```
# Cluster
cluster_enabled:0

# Keyspace
127.0.0.1:6379>
```



```
# Stats
total_connections_received:1
total_commands_processed:16
instantaneous_ops_per_sec:0
total_net_input_bytes:485
total_net_output_bytes:168
instantaneous_input_kbps:0.00
instantaneous_output_kbps:0.00
rejected_connections:0
sync_full:0
sync_partial_ok:0
sync_partial_err:0
expired_keys:0
evicted_keys:0
keyspace_hits:3
keyspace_misses:1
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:2903
migrate_cached_sockets:0

# Replication
role:master
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0

# CPU
used_cpu_sys:0.17
used_cpu_user:0.45
used_cpu_sys_children:0.00
```