Soumya Ranjan **Division B** **Roll No.- 46**

PRACTICAL - 3

Aim: Implementation of ORDBMS using ADT(Abstract Data Types), References, etc.

ORDBMS [Object-Relational Database Management System]

An **object relational database management system** (ORDBMS) is a database management system that is similar to a relational database, except that it has an object-oriented database model. This system supports objects, classes and inheritance in database schemas and query language.

- PostgreSQL.
- Informix by IBM.
- SQL Server by Microsoft.
- Greenplum Database by Pivotal Software.

Advantages of ORDBMSs

There are following advantages of ORDBMSs:

Reuse and Sharing:

The main advantages of extending the Relational data model come from reuse and sharing. Reuse comes from the ability to extend the DBMS server to perform standard functionality centrally, rather than have it coded in each application.

Increased Productivity:

ORDBMS provides increased productivity both for the developer and for the, end user

Use of experience in developing RDBMS:

Another obvious advantage is that the extended relational approach preserves the significant body of knowledge and experience that has gone into developing relational applications. This is a significant advantage, as many organizations would find it prohibitively expensive to change. If the new functionality is designed appropriately, this approach should allow organizations to take advantage of the new extensions in an evolutionary way without losing the benefits of current database features and functions.

Disadvantages of ORDBMSs

The ORDBMS approach has the obvious disadvantages of complexity and associated increased costs. Further, there are the proponents of the relational approach that believe the· essential simplicity' and purity of the relational model are lost with these types of extension.

ORDBMS vendors are attempting to portray object models as extensions to the relational model with some additional complexities. This potentially misses the point of object orientation, highlighting the large semantic gap between these two technologies. Object applications are simply not as data-centric as relational-based ones.

Abstract Data Types (ADT):-

By using abstract data types, which are user-defined types, together with various routines, you can uniquely define and use data with complex structures and perform operations on such data. When you define a column as having an abstract data type, you can conceptualize and model its data based on object-oriented concepts. In addition, by applying object oriented software development techniques, you can reduce the workload for database design, UAP development, and maintenance

REF Function:-

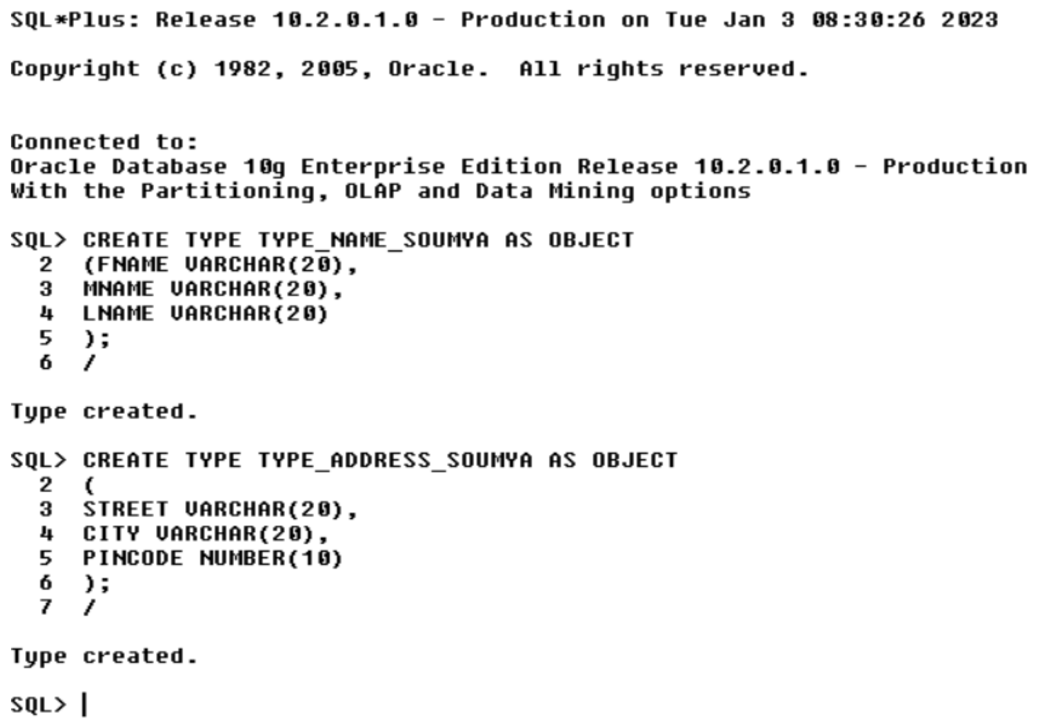
In Oracle PL/SQL, REF data types are pointers that uniquely identify a piece of data as an object. A reference can be established between an existent valid object and a table or type attribute using the REF pointer data type. An attribute referring to a nonexistent object leads to "dangling" situation. Note that a NULL object reference is different from a Dangling Reference. To insert data into a ref column, the REF function is used to get an object instance reference.

DEREF Function:-

DEREF returns the object reference of argument expr, where expr must return a REF to an object. If you do not use this function in a query, then Oracle Database returns the object ID of the REF instead.

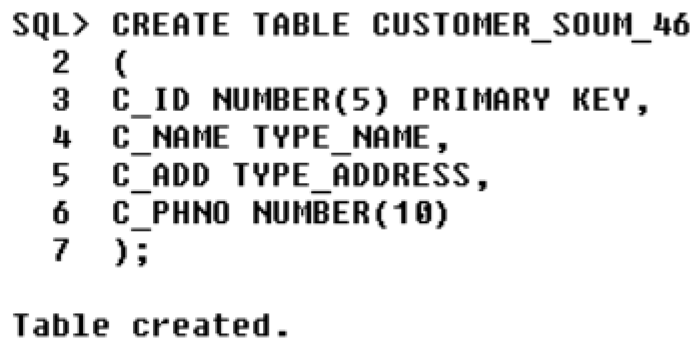
A] Abstract Data Type :

Creating type: type_name_soum_46 as object:



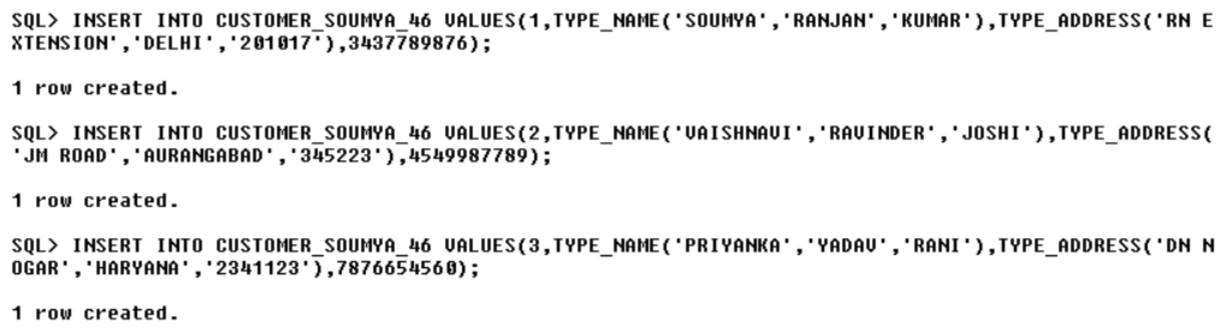
```
SQL*Plus: Release 10.2.0.1.0 - Production on Tue Jan 3 08:30:26 2023

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> CREATE TYPE TYPE_NAME_SOUMYA AS OBJECT
  2  (FNAME VARCHAR(20),
  3  MNAME VARCHAR(20),
  4  LNAME VARCHAR(20)
  5  );
  6  /

Type created.

SQL> CREATE TYPE TYPE_ADDRESS_SOUMYA AS OBJECT
  2  (
  3  STREET VARCHAR(20),
  4  CITY VARCHAR(20),
  5  PINCODE NUMBER(10)
  6  );
  7  /

Type created.

SQL> |
```

Creating table customer_soum_46:



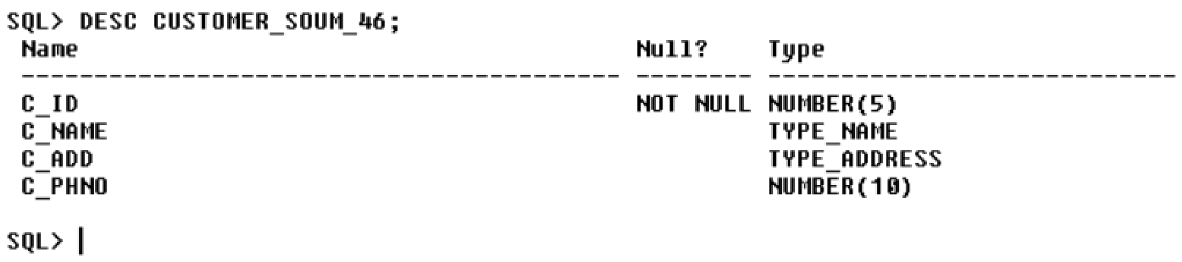
```
SQL> CREATE TABLE CUSTOMER_SOUM_46
  2  (
  3  C_ID NUMBER(5) PRIMARY KEY,
  4  C_NAME TYPE_NAME,
  5  C_ADD TYPE_ADDRESS,
  6  C_PHNO NUMBER(10)
  7  );

Table created.
```

Inserting the records in the customer_soum_46 table:



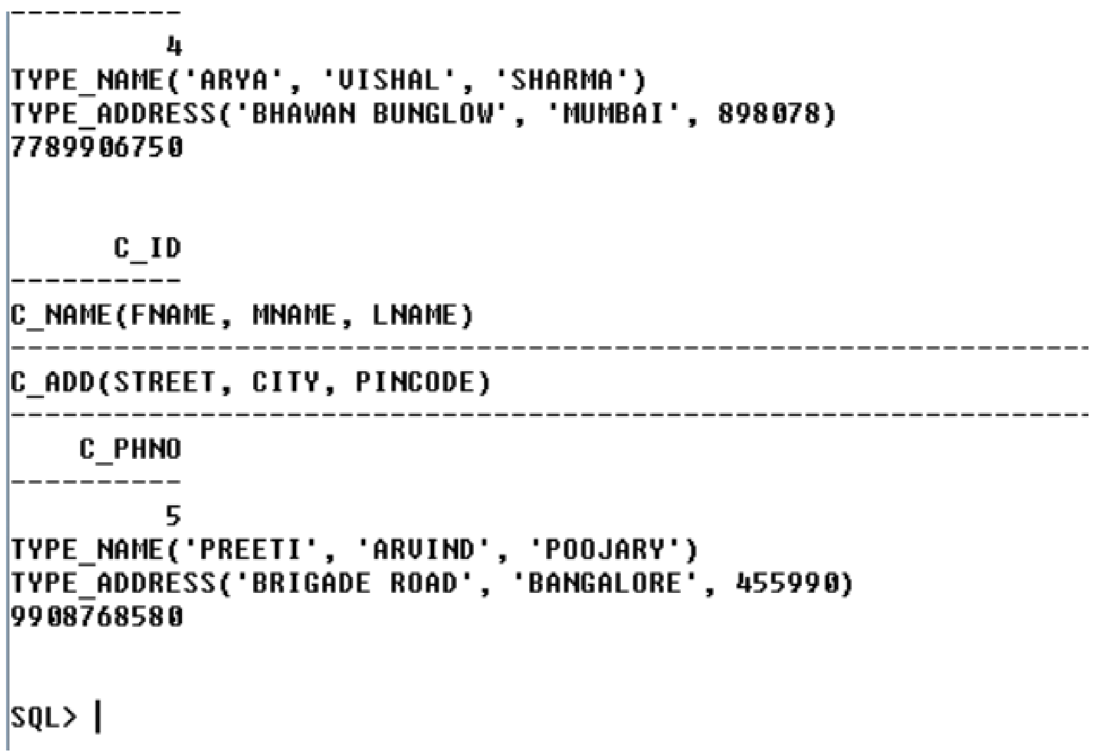
```
SQL> INSERT INTO CUSTOMER_SOUMYA_46 VALUES(1,TYPE_NAME('SOUMYA','RANJAN','KUMAR'),TYPE_ADDRESS('RN E
XTENSION','DELHI','201017'),3437789876);

1 row created.

SQL> INSERT INTO CUSTOMER_SOUMYA_46 VALUES(2,TYPE_NAME('VAISHNAVI','RAVINDER','JOSHI'),TYPE_ADDRESS(
'JM ROAD','AURANGABAD','345223'),4549987789);

1 row created.

SQL> INSERT INTO CUSTOMER_SOUMYA_46 VALUES(3,TYPE_NAME('PRIYANKA','YADAV','RANI'),TYPE_ADDRESS('DN N
OGAR','HARYANA','2341123'),7876654560);

1 row created.
```

Checking the structure of the table:



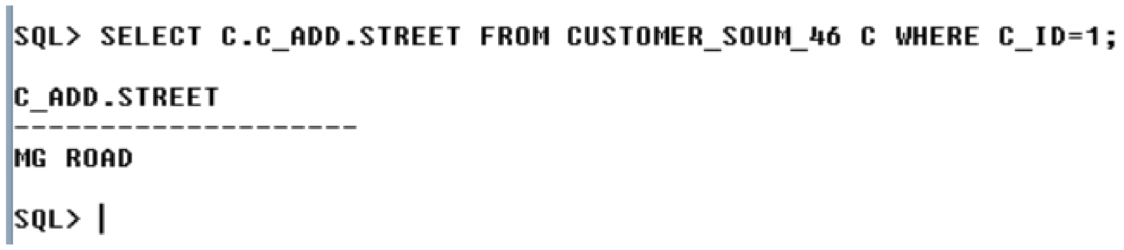
```
SQL> DESC CUSTOMER_SOUM_46;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 C_ID                                      NOT NULL NUMBER(5)
 C_NAME                                             TYPE_NAME
 C_ADD                                              TYPE_ADDRESS
 C_PHNO                                             NUMBER(10)

SQL> |
```

Displaying all records:



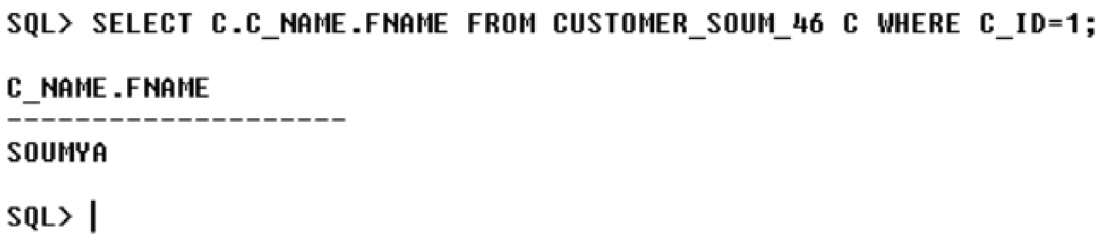
```
----------
         4
TYPE_NAME('ARYA', 'VISHAL', 'SHARMA')
TYPE_ADDRESS('BHAWAN BUNGLOW', 'MUMBAI', 898078)
7789906750


      C_ID
----------
C_NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------------------
C_ADD(STREET, CITY, PINCODE)
--------------------------------------------------------------------------------
    C_PHNO
----------
         5
TYPE_NAME('PREETI', 'ARVIND', 'POOJARY')
TYPE_ADDRESS('BRIGADE ROAD', 'BANGALORE', 455990)
9908768580


SQL> |
```

Displaying only street of the customer c_id=1;

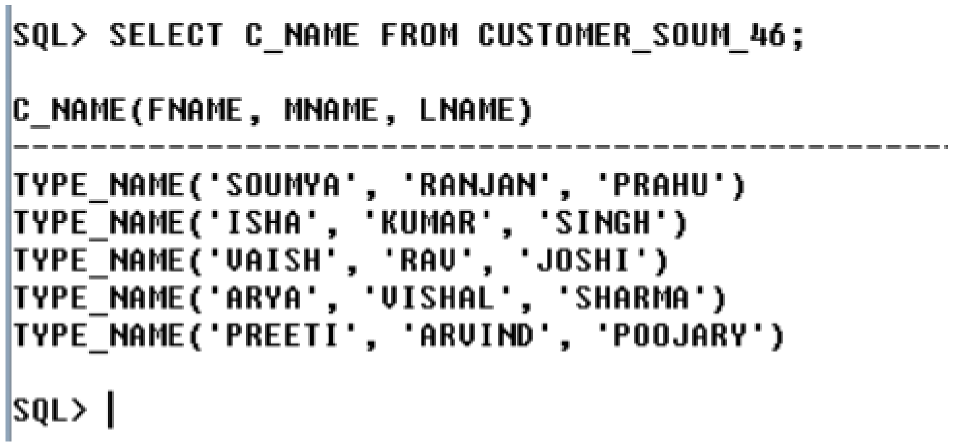


```
SQL> SELECT C.C_ADD.STREET FROM CUSTOMER_SOUM_46 C WHERE C_ID=1;

C_ADD.STREET
--------------------
MG ROAD

SQL> |
```

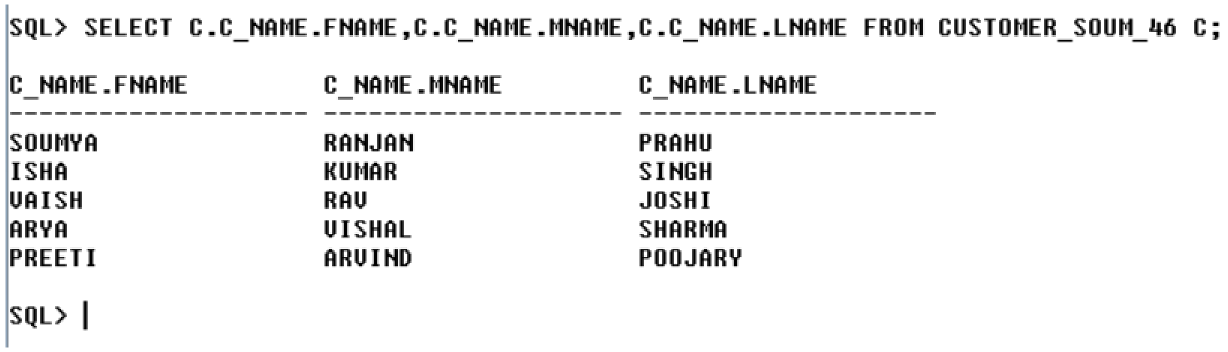
Displaying only FNAME of the customer C_ID=1;



```
SQL> SELECT C.C_NAME.FNAME FROM CUSTOMER_SOUM_46 C WHERE C_ID=1;

C_NAME.FNAME
--------------------
SOUMYA

SQL> |
```

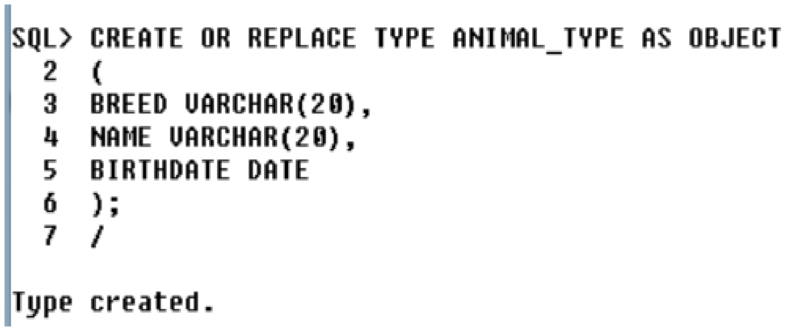


```
SQL> SELECT C_NAME FROM CUSTOMER_SOUM_46;

C_NAME(FNAME, MNAME, LNAME)
------------------------------------------------
TYPE_NAME('SOUMYA', 'RANJAN', 'PRAHU')
TYPE_NAME('ISHA', 'KUMAR', 'SINGH')
TYPE_NAME('VAISH', 'RAV', 'JOSHI')
TYPE_NAME('ARYA', 'VISHAL', 'SHARMA')
TYPE_NAME('PREETI', 'ARVIND', 'POOJARY')

SQL> |
```

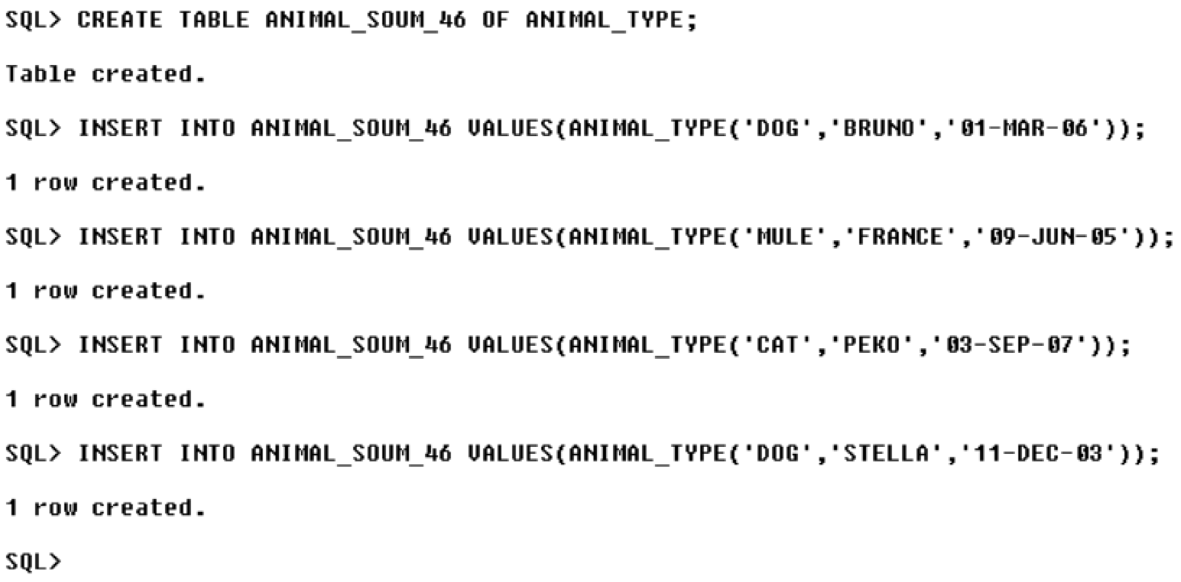


```
SQL> SELECT C.C_NAME.FNAME,C.C_NAME.MNAME,C.C_NAME.LNAME FROM CUSTOMER_SOUM_46 C;

C_NAME.FNAME         C_NAME.MNAME         C_NAME.LNAME
-------------------- -------------------- --------------------
SOUMYA               RANJAN               PRAHU
ISHA                 KUMAR                SINGH
VAISH                RAV                  JOSHI
ARYA                 VISHAL               SHARMA
PREETI               ARVIND               POOJARY

SQL> |
```

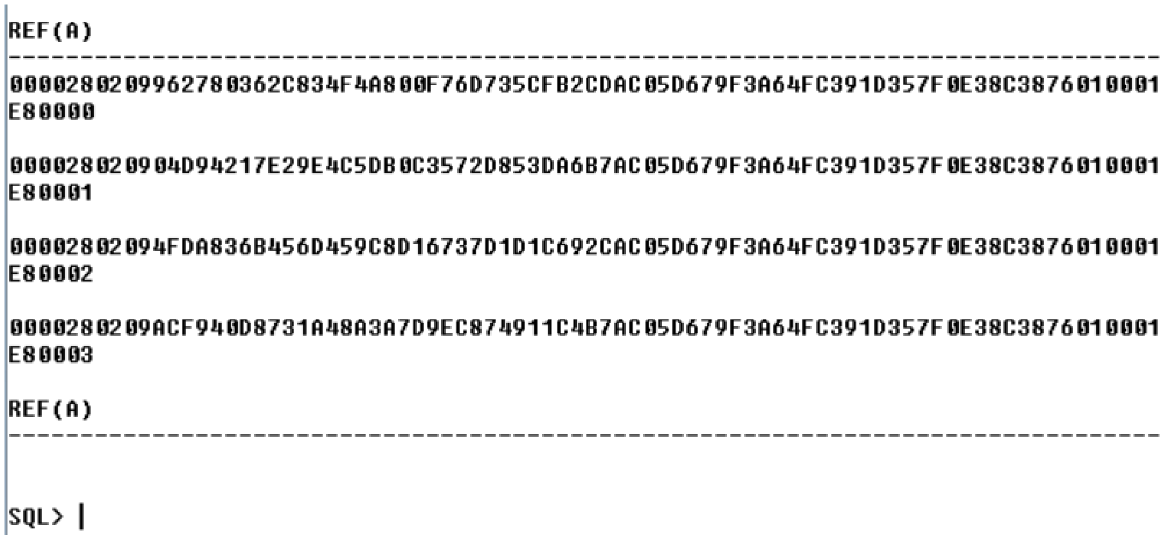
B) REF



```
SQL> CREATE OR REPLACE TYPE ANIMAL_TYPE AS OBJECT
  (
  BREED VARCHAR(20),
  NAME VARCHAR(20),
  BIRTHDATE DATE
  );
  /

Type created.
```

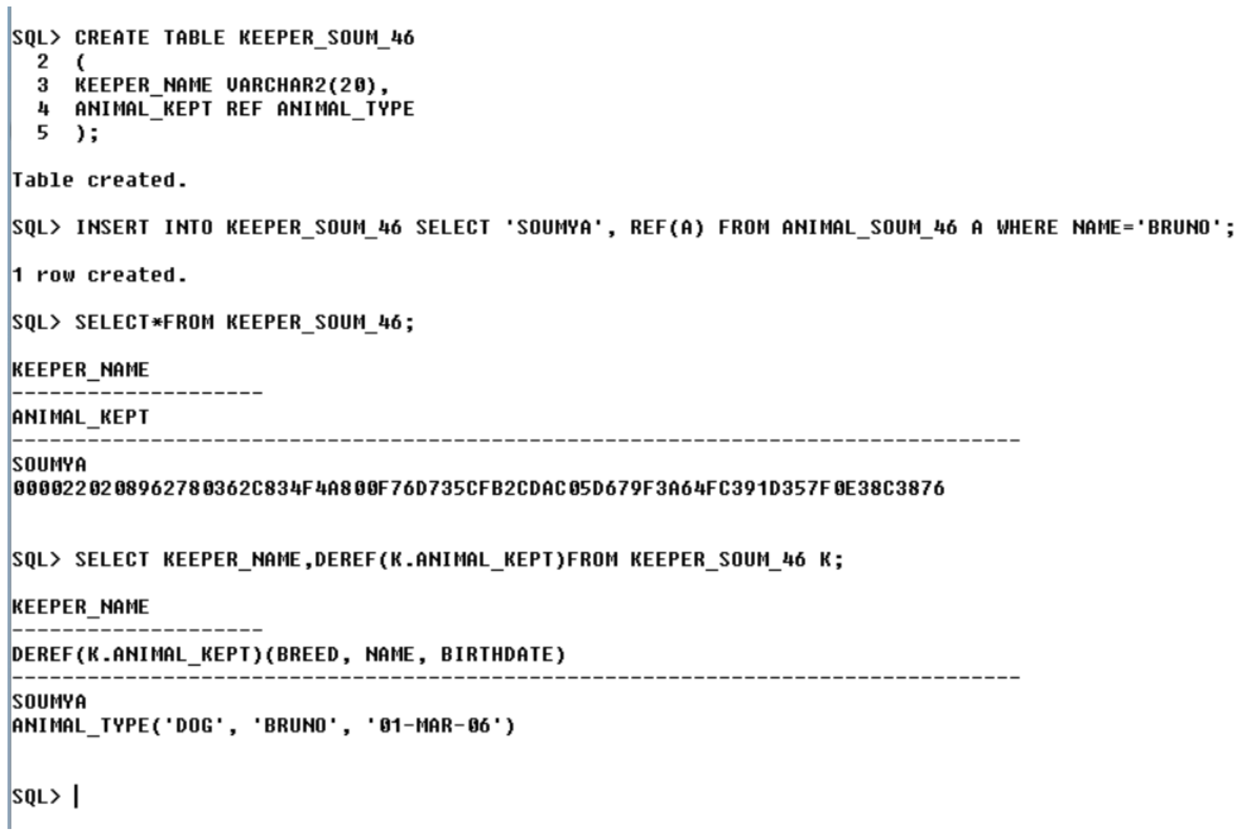


```
SQL> CREATE TABLE ANIMAL_SOUM_46 OF ANIMAL_TYPE;

Table created.

SQL> INSERT INTO ANIMAL_SOUM_46 VALUES(ANIMAL_TYPE('DOG','BRUNO','01-MAR-06'));

1 row created.

SQL> INSERT INTO ANIMAL_SOUM_46 VALUES(ANIMAL_TYPE('MULE','FRANCE','09-JUN-05'));

1 row created.

SQL> INSERT INTO ANIMAL_SOUM_46 VALUES(ANIMAL_TYPE('CAT','PEKO','03-SEP-07'));

1 row created.

SQL> INSERT INTO ANIMAL_SOUM_46 VALUES(ANIMAL_TYPE('DOG','STELLA','11-DEC-03'));

1 row created.

SQL>
```



```
REF(A)
--------------------------------------------------------------------------------
00002802099627803 62C834F4A800F76D735CFB2CDAC05D679F3A64FC391D357F0E38C3876010001
E80000

0000280209 04D94217E29E4C5DB0C3572D853DA6B7AC05D679F3A64FC391D357F0E38C3876010001
E80001

000028020 94FDA836B456D459C8D16737D1D1C692CAC05D679F3A64FC391D357F0E38C3876010001
E80002

0000280209ACF940D8731A48A3A7D9EC874911C4B7AC05D679F3A64FC391D357F0E38C3876010001
E80003

REF(A)
--------------------------------------------------------------------------------


SQL> |
```

C) DEREF:



```
SQL> CREATE TABLE KEEPER_SOUM_46
  2  (
  3  KEEPER_NAME VARCHAR2(20),
  4  ANIMAL_KEPT REF ANIMAL_TYPE
  5  );

Table created.

SQL> INSERT INTO KEEPER_SOUM_46 SELECT 'SOUMYA', REF(A) FROM ANIMAL_SOUM_46 A WHERE NAME='BRUNO';

1 row created.

SQL> SELECT*FROM KEEPER_SOUM_46;

KEEPER_NAME
--------------------
ANIMAL_KEPT
--------------------------------------------------------------------------------
SOUMYA
0000220208962780362C834F4A800F76D735CFB2CDAC05D679F3A64FC391D357F0E38C3876


SQL> SELECT KEEPER_NAME,DEREF(K.ANIMAL_KEPT)FROM KEEPER_SOUM_46 K;

KEEPER_NAME
--------------------
DEREF(K.ANIMAL_KEPT)(BREED, NAME, BIRTHDATE)
--------------------------------------------------------------------------------
SOUMYA
ANIMAL_TYPE('DOG', 'BRUNO', '01-MAR-06')


SQL> |
```

Conclusion:

ORDBMS using ADT(Abstract Data Types), Reference, Dereference is implemented successfully.