EX.NO : 7

SET OPERATORS AND JOINS

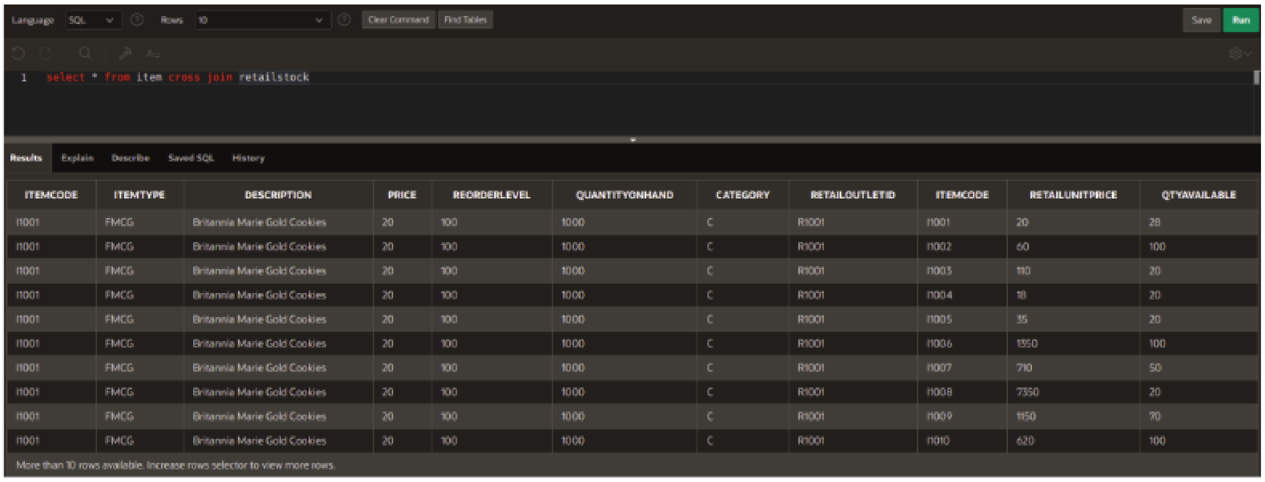
Aim :

To execute queries based on set operators and joins.

Test Cases (Joins) :

1. .Perform Cross JOIN on item and Retailstock. Write the Difference between Cross Join and Inner Join

select * from item cross join retailstock



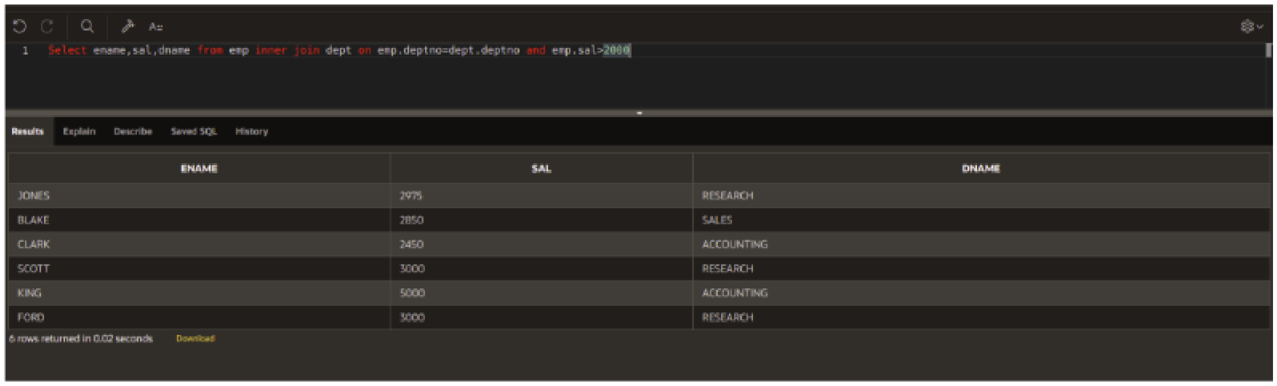
```
select * from item cross join retailstock
```

ITEMCODE	ITEMTYPE	DESCRIPTION	PRICE	REORDERLEVEL	QUANTITYONHAND	CATEGORY	RETAILOUTLETID	ITEMCODE	RETAILUNITPRICE	QTYAVAILABLE
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1001	20	28
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1002	60	100
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1003	110	20
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1004	18	20
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1005	35	20
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1006	1350	100
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1007	710	50
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1008	7350	20
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1009	1150	70
I1001	FMCG	Britannia Marie Gold Cookies	20	100	1000	C	R1001	I1010	620	100

More than 10 rows available. Increase rows selector to view more rows.

2. Display name and salary of the employees,who are drwing salary more than 2000. Along with that display the department names in which they are working.Solve this using INNER JOIN

Select ename,sal,dname from emp inner join dept on emp.deptno=dept.deptno and emp.sal>2000

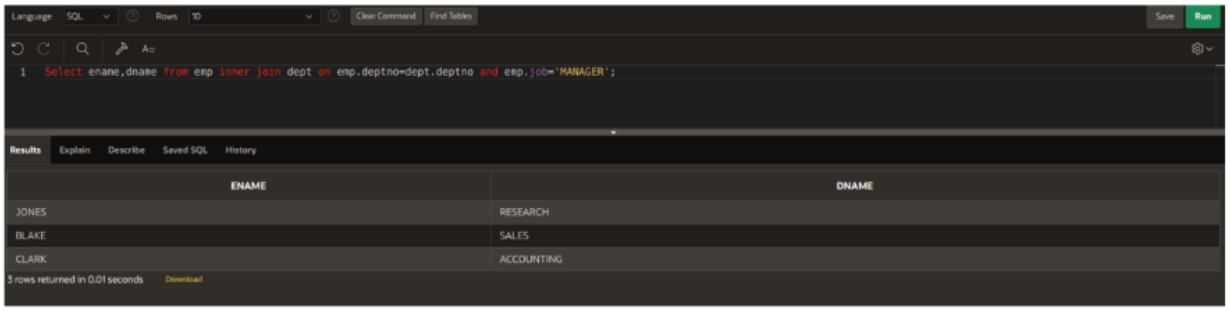


```
Select ename,sal,dname from emp inner join dept on emp.deptno=dept.deptno and emp.sal>2000
```

ENAME	SAL	DNAME
JONES	2975	RESEARCH
BLAKE	2850	SALES
CLARK	2450	ACCOUNTING
SCOTT	3000	RESEARCH
KING	5000	ACCOUNTING
FORD	3000	RESEARCH

6 rows returned in 0.02 seconds

3. Display the name of employees and their department names who are managers.Solve this using INNER JOIN.
Select ename,dname from emp inner join dept on emp.deptno=dept.deptno and emp.job='MANAGER'

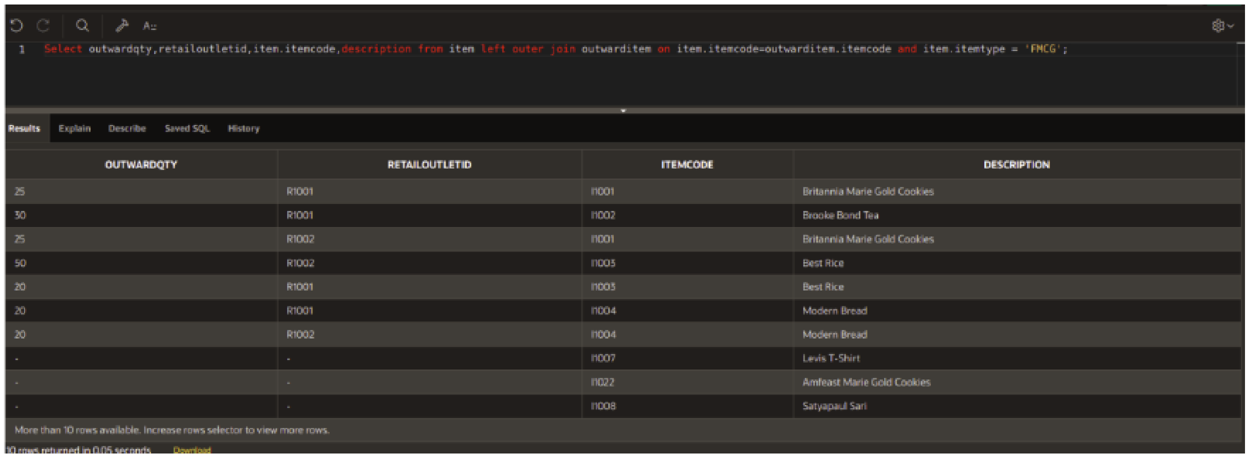


```
Select ename,dname from emp inner join dept on emp.deptno=dept.deptno and emp.job='MANAGER';
```

ENAME	DNAME
JONES	RESEARCH
BLAKE	SALES
CLARK	ACCOUNTING

3 rows returned in 0.01 seconds

4. Display itemcode,description of all items of type ‘FMCG’ along with outwardqty and reatiloutletid where the items have been moved. Solve this using Left outer join. **Select outwardqty,retailoutletid,item.itemcode,description from item left outer join outwarditem on item.itemcode=outwarditem.itemcode and item.itemtype = 'FMCG'**



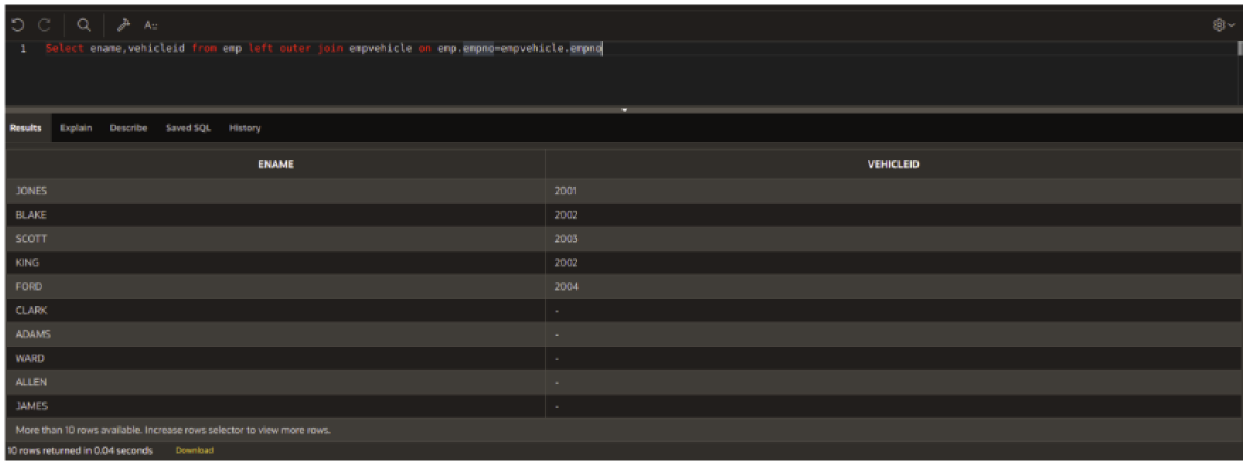
```
Select outwardqty,retailoutletid,item.itemcode,description from item left outer join outwarditem on item.itemcode=outwarditem.itemcode and item.itemtype = 'FMCG';
```

OUTWARDQTY	RETAILOUTLETID	ITEMCODE	DESCRIPTION
25	R1001	I1001	Britannia Marie Gold Cookies
30	R1001	I1002	Brooke Bond Tea
25	R1002	I1001	Britannia Marie Gold Cookies
50	R1002	I1003	Best Rice
20	R1001	I1003	Best Rice
20	R1001	I1004	Modern Bread
20	R1002	I1004	Modern Bread
-	-	I1007	Levis T-Shirt
-	-	I1022	Amfeast Marie Gold Cookies
-	-	I1008	Satyapaul Sari

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.05 seconds

5. For each employee,identify the vehicle owned by them. Displayename and vehicleid for the same. Display name of employees even if they don’t own any vehicle. Solve this using Left outer join.
Select ename,vehicleid from emp left outer join empvehicle on emp.empno=empvehicle.empno



```sql
Select ename,vehicleid from emp left outer join empvehicle on emp.empno=empvehicle.empno
```

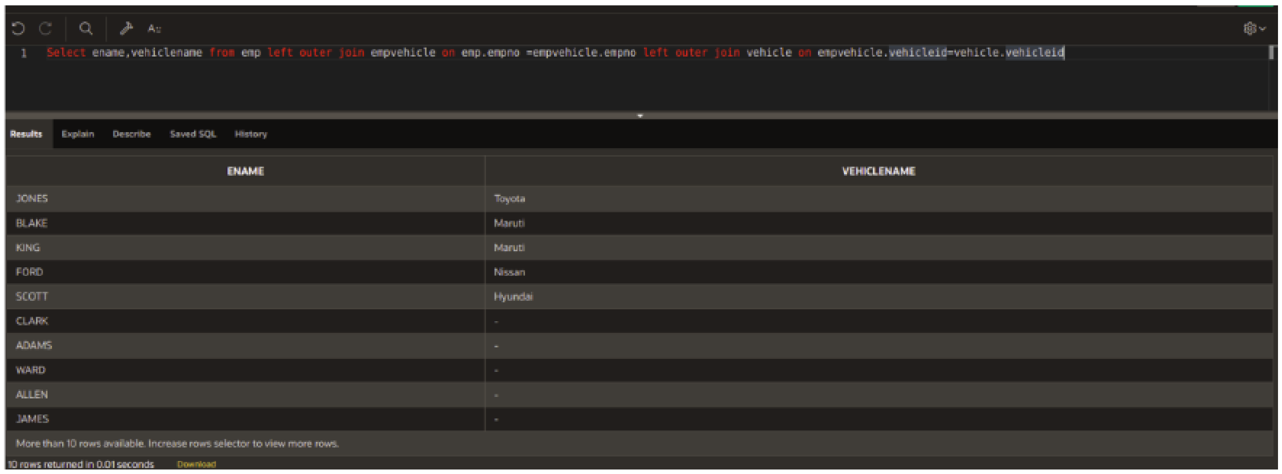
ENAME	VEHICLEID
JONES	2001
BLAKE	2002
SCOTT	2003
KING	2002
FORD	2004
CLARK	-
ADAMS	-
WARD	-
ALLEN	-
JAMES	-

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.04 seconds Download

6. For each employee, identify the vehicle owned by them. Displayename and vehiclename for the same. Display name of employees even if they don't own any vehicle. Solve this using Left outer join.

Select ename,vehiclename from emp left outer join empvehicle on emp.empno = empvehicle.empno left outer join vehicle on empvehicle.vehicleid=vehicle.vehicleid



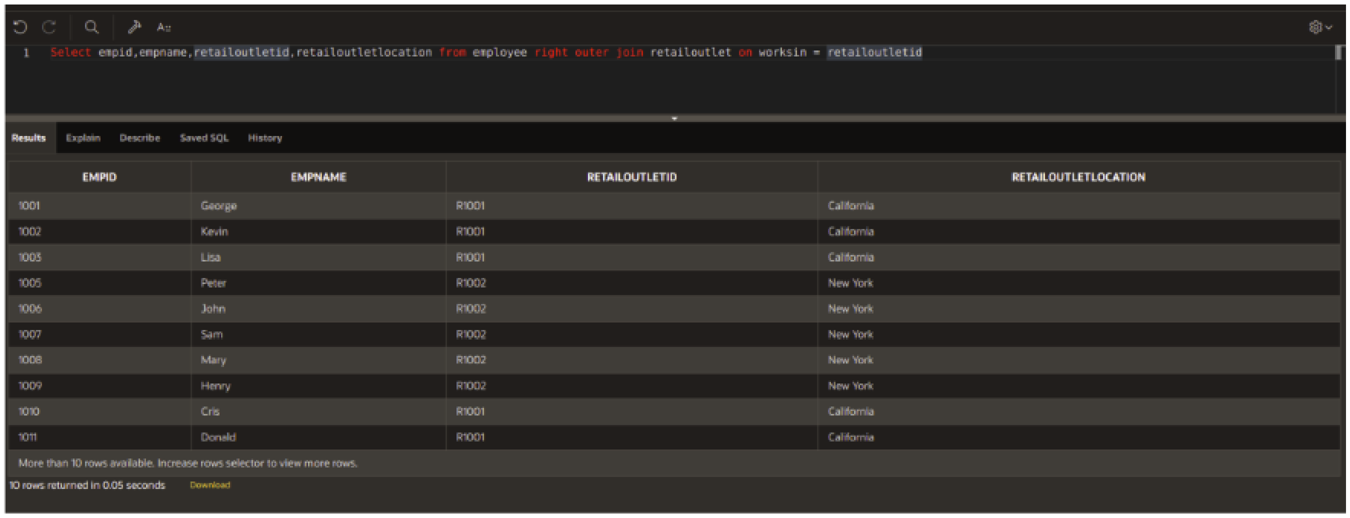
```sql
Select ename,vehiclename from emp left outer join empvehicle on emp.empno =empvehicle.empno left outer join vehicle on empvehicle.vehicleid=vehicle.vehicleid
```

ENAME	VEHICLENAME
JONES	Toyota
BLAKE	Maruti
KING	Maruti
FORD	Nissan
SCOTT	Hyundai
CLARK	-
ADAMS	-
WARD	-
ALLEN	-
JAMES	-

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download

7. For every retail outlet, identify the employees working in it.. Display empid,empname,retailoutletid and their retailoutletlocation.Solve this using Right outer join. **Select empid,empname,retailoutletid,retailoutletlocation from employee right outer join retailoutlet on worksin = retailoutletid**



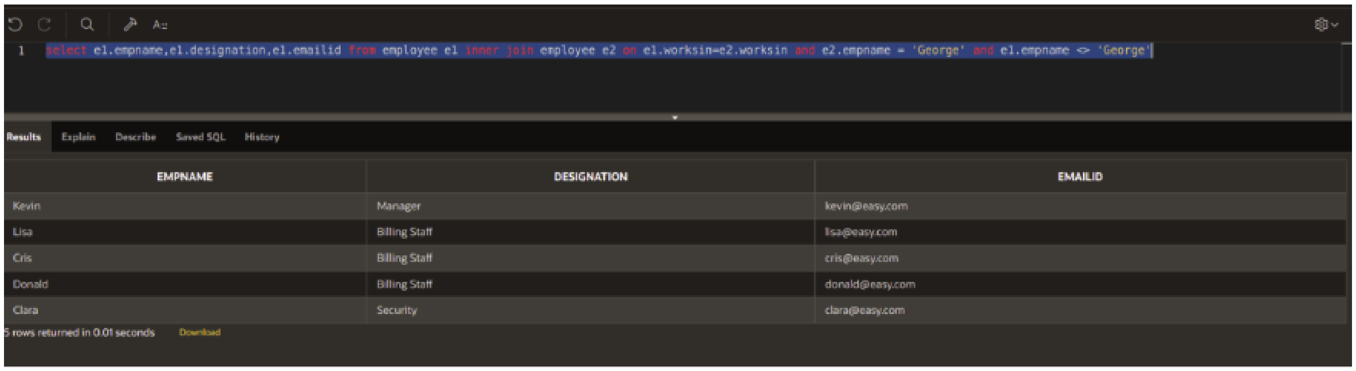
```
Select empid,empname,retailoutletid,retailoutletlocation from employee right outer join retailoutlet on worksin = retailoutletid
```

EMPID	EMPNAME	RETAILOUTLETID	RETAILOUTLETLOCATION
1001	George	R1001	California
1002	Kevin	R1001	California
1003	Lisa	R1001	California
1005	Peter	R1002	New York
1006	John	R1002	New York
1007	Sam	R1002	New York
1008	Mary	R1002	New York
1009	Henry	R1002	New York
1010	Cris	R1001	California
1011	Donald	R1001	California

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.05 seconds

8. Retrieve employee name,designation and email id of those employees who work in the same retail outlet where George works. Do not display the record of George in the result. **select e1.empname,e1.designation,e1.emailid from employee e1 inner join employee e2 on e1.worksin=e2.worksin and e2.empname = 'George' and e1.empname <> 'George'**



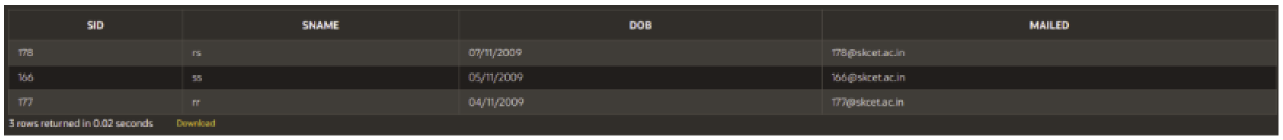
```
select e1.empname,e1.designation,e1.emailid from employee e1 inner join employee e2 on e1.worksin=e2.worksin and e2.empname = 'George' and e1.empname <> 'George'
```

EMPNAME	DESIGNATION	EMAILID
Kevin	Manager	kevin@easy.com
Lisa	Billing Staff	lisa@easy.com
Cris	Billing Staff	cris@easy.com
Donald	Billing Staff	donald@easy.com
Clara	Security	clara@easy.com

5 rows returned in 0.01 seconds

<u>Test Cases (Joins) :</u>

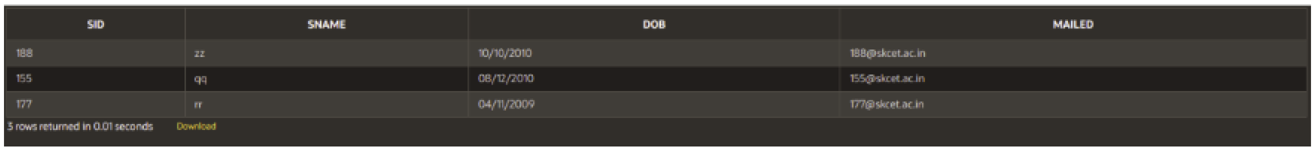
create table student1(sid varchar2(6), sname varchar2(30),dob date, mailed varchar2(30)); insert into student1 values ('177','rr','04/11/2009','177@skcet.ac.in');
insert into student1 values ('166','ss','05/11/2009','166@skcet.ac.in');
insert into student1 values ('178','rs','07/11/2009','178@skcet.ac.in');
select * from student1;



SID	SNAME	DOB	MAILED
178	rs	07/11/2009	178@skcet.ac.in
166	ss	05/11/2009	166@skcet.ac.in
177	rr	04/11/2009	177@skcet.ac.in

3 rows returned in 0.02 seconds

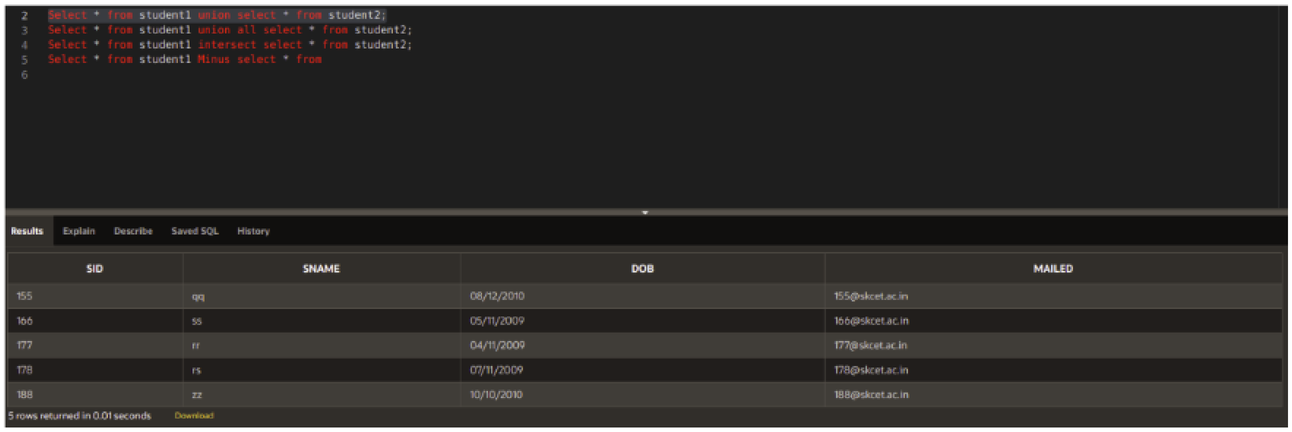
```
create table student2(sid varchar2(6), sname varchar2(30),dob date, mailed
varchar2(30)); insert into student2 values
('177','rr','04/11/2009','177@skcet.ac.in');
insert into student2 values ('155','qq','08/12/2010','155@skcet.ac.in');
insert into student2 values ('188','zz','10/10/2010','188@skcet.ac.in');
select * from student2;
```



SID	SNAME	DOB	MAILED
188	zz	10/10/2010	188@skcet.ac.in
155	qq	08/12/2010	155@skcet.ac.in
177	rr	04/11/2009	177@skcet.ac.in

3 rows returned in 0.01 seconds Download

```
Select * from student1 union select * from student2;
```

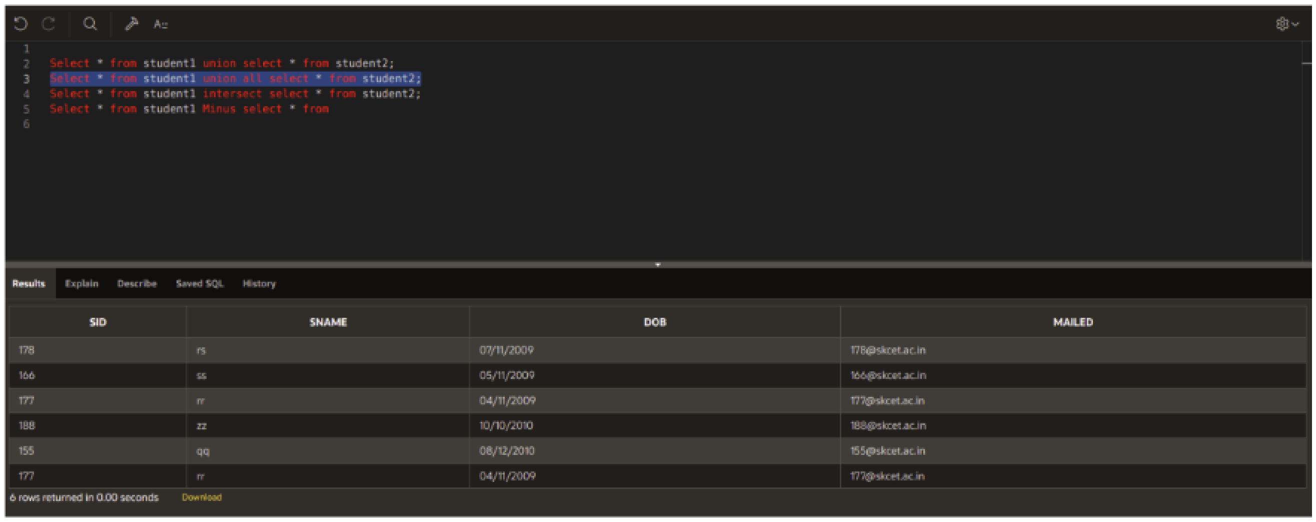


```
Select * from student1 union select * from student2;
Select * from student1 union all select * from student2;
Select * from student1 intersect select * from student2;
Select * from student1 Minus select * from
```

SID	SNAME	DOB	MAILED
155	qq	08/12/2010	155@skcet.ac.in
166	ss	05/11/2009	166@skcet.ac.in
177	rr	04/11/2009	177@skcet.ac.in
178	rs	07/11/2009	178@skcet.ac.in
188	zz	10/10/2010	188@skcet.ac.in

5 rows returned in 0.01 seconds Download

```
Select * from student1 union all select * from student2;
```

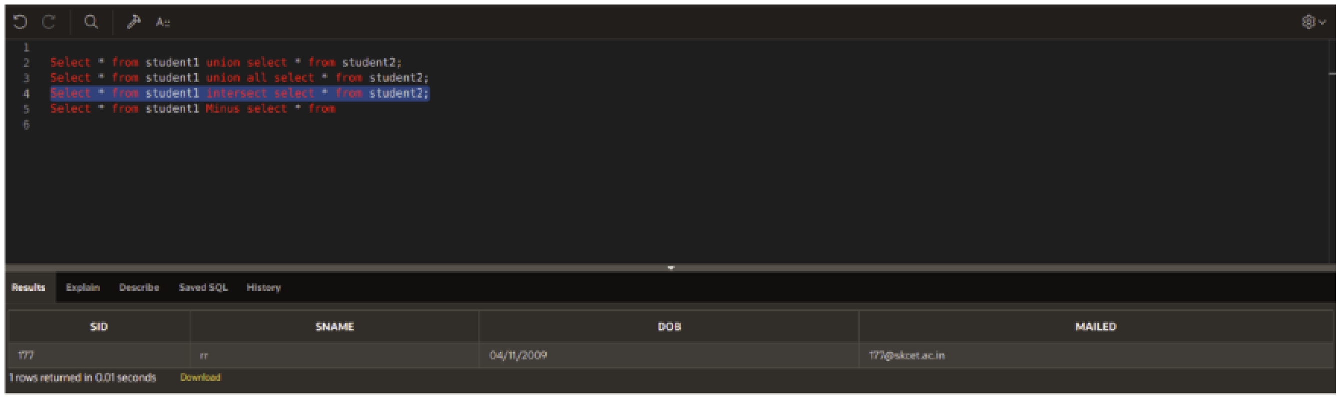


```
Select * from student1 union select * from student2;
Select * from student1 union all select * from student2;
Select * from student1 intersect select * from student2;
Select * from student1 Minus select * from
```

SID	SNAME	DOB	MAILED
178	rs	07/11/2009	178@skcet.ac.in
166	ss	05/11/2009	166@skcet.ac.in
177	rr	04/11/2009	177@skcet.ac.in
188	zz	10/10/2010	188@skcet.ac.in
155	qq	08/12/2010	155@skcet.ac.in
177	rr	04/11/2009	177@skcet.ac.in

6 rows returned in 0.00 seconds Download

Select * from student1 intersect select * from student2;

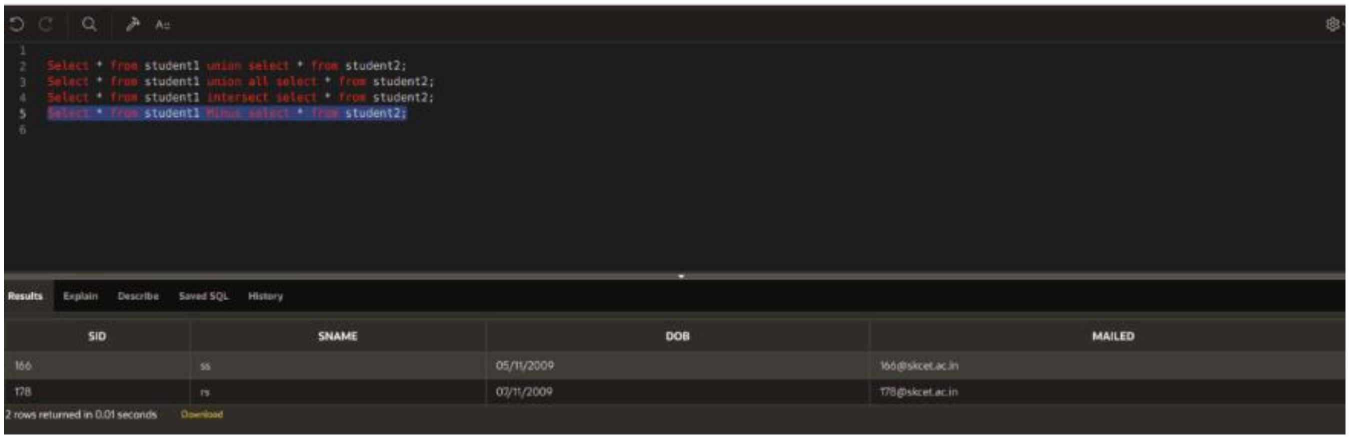


```
Select * from student1 union select * from student2;
Select * from student1 union all select * from student2;
Select * from student1 intersect select * from student2;
Select * from student1 Minus select * from
```

SID	SNAME	DOB	MAILED
177	rr	04/11/2009	177@skcet.ac.in

1 rows returned in 0.01 seconds   Download

Select * from student1 Minus select * from student2;



```
Select * from student1 union select * from student2;
Select * from student1 union all select * from student2;
Select * from student1 intersect select * from student2;
Select * from student1 Minus select * from student2;
```

SID	SNAME	DOB	MAILED
166	ss	05/11/2009	166@skcet.ac.in
178	rs	07/11/2009	178@skcet.ac.in

2 rows returned in 0.01 seconds   Download

<u>RESULT:</u>

Thus join operations are executed successfully.

EX.NO: 8

IMPLEMENTATION OF VIRTUAL TABLES USING VIEWS

Objectives:

1. To implement VIEW operations using SQL.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Understand the use of virtual table and provide a restricted view to the data in the table.
- Summarize data from various tables which can be used to generate reports.

Problem Statement:

To perform VIEW the following commands are executed:

1. Create view
2. Inserting Rows into a View
3. Deleting Rows from a View
4. Update View
5. Delete View

System and Software tools required:

Software Required: Oracle

Operating System: WINDOWS 2000 / XP / NT OR LINUX OR UBUNTU

Computers Required: Minimum Requirement: Pentium III or Pentium IV with 256 RAM and 40 GB hard disk

Description:

Views are virtual tables that allow users to structure data in a way that users find intuitive. It restricts access to data so the user can see and (sometimes) modify exactly what they need. A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed.

Creating Views:

Syntax

create view view_name <as> select column1, column2.....from table_name where [condition];

Inserting Rows into a View:

Rows of data can be inserted into a view. The same rules that apply to the insert command in DML are applied here.

Deleting Rows from a View / Delete view:

Rows of data can be deleted from a view. The same rules that apply to the delete command in DML are applied here.

Updating a View:

The same rules that apply to the update command in DML are applied

here. A view can be updated under certain conditions:

1. The SELECT clause may not contain the keyword DISTINCT.
2. The SELECT clause may not contain set operators.
3. The SELECT clause may not contain an order by group by or having clause.
4. The FROM clause may not contain multiple tables.
5. The WHERE clause may not contain sub queries.

Sample Coding and Execution of the Program:

1. create view emp_7 as select employee_id,address from employee7;

2. update emp_7 set address='ssnagar' where employee_id='14pc02';

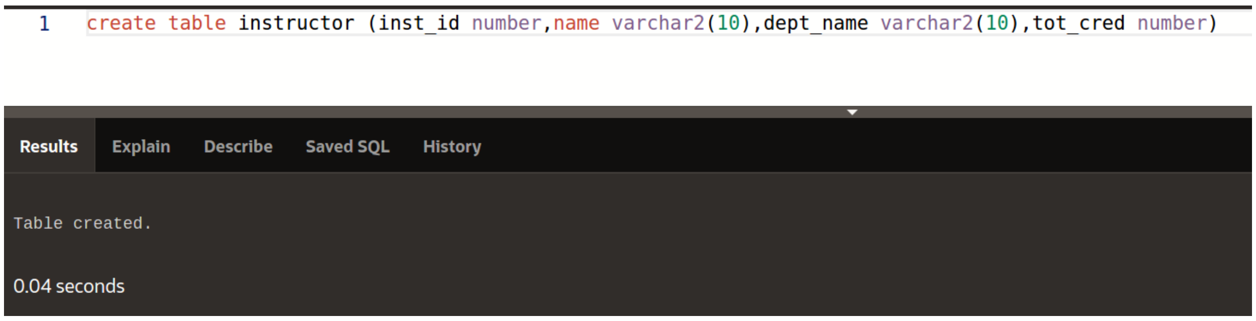
Sample Output:

```
1.  EMPLOYEE_ID           ADDRESS
         --------------------       --------------------

          14pc01                    kknagar

          14pc02                     amman nagar

          14pc03                     nnnagar

          14pc02                     amman nagar

          14pc04                     rrnagar

          14pc06                     ggnagar

2. EMPLOYEE_ID           ADDRESS
       --------------------        --------------------
          14pc01                    kknagar
          14pc02                    ssnagar
          14pc03                  nnnagar

          14pc02                  ssnagar

          14pc04                    rrnagar

          14pc06                  ggnagar

          14pc07                  grnagar
```

Test Cases:

1.Create a relation instructor with following fields instructor_idnumber,name varchar2(10), dept_name varchar2(10),tot_cred number(10)

create table instructor (inst_idnumber,name varchar2(10),dept_name varchar2(10),tot_cred number)

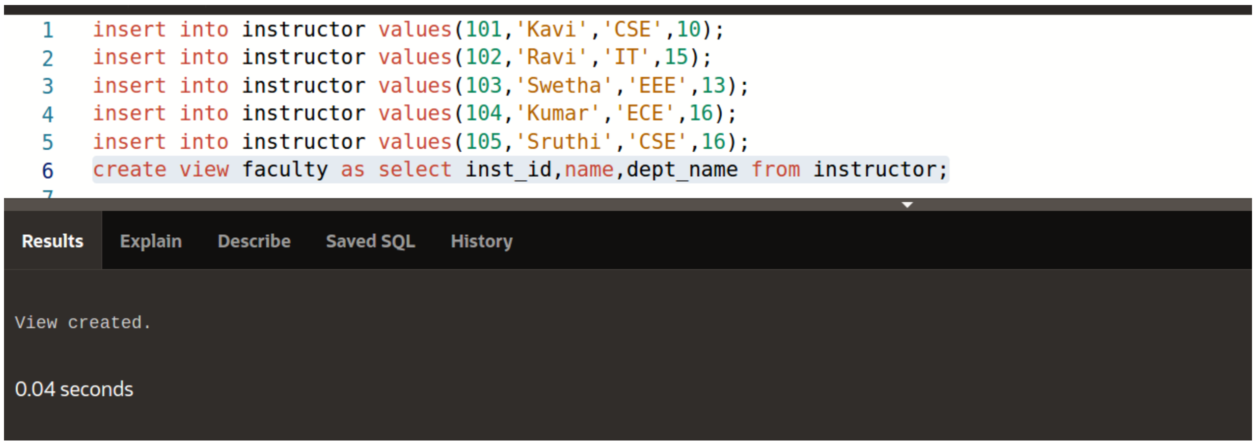


```
create table instructor (inst_id number,name varchar2(10),dept_name varchar2(10),tot_cred number)
```

```
Table created.

0.04 seconds
```

2.Insert 5 values into the instructor relation. Create a view in the name of faculty by having the fields instructor_id,name,dept_name from the instructor table.

insert into instructor values(101,'Kavi','CSE',10);
insert into instructor values(102,'Ravi','IT',15);
insert into instructor values(103,'Swetha','EEE',13);
insert into instructor values(104,'Kumar','ECE',16);
insert into instructor values(105,'Sruthi','CSE',16);
create view faculty as select inst_id,name,dept_name from instructor;

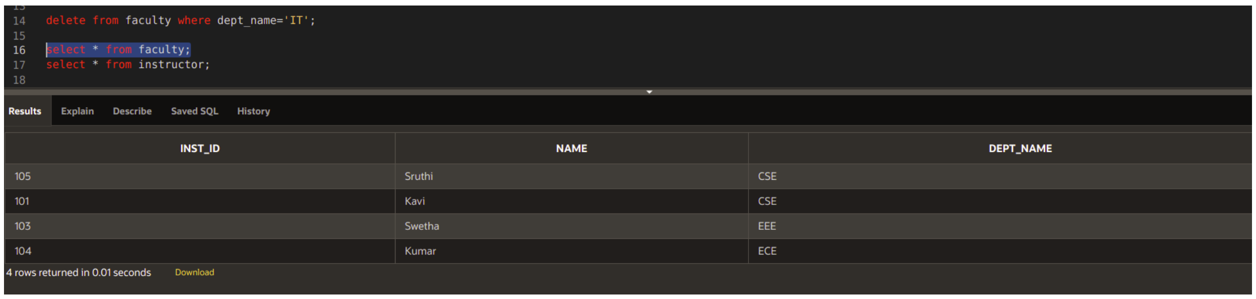


```
insert into instructor values(101,'Kavi','CSE',10);
insert into instructor values(102,'Ravi','IT',15);
insert into instructor values(103,'Swetha','EEE',13);
insert into instructor values(104,'Kumar','ECE',16);
insert into instructor values(105,'Sruthi','CSE',16);
create view faculty as select inst_id,name,dept_name from instructor;
```

```
View created.

0.04 seconds
```

3.Retrieve the record from both instructor and faculty relation

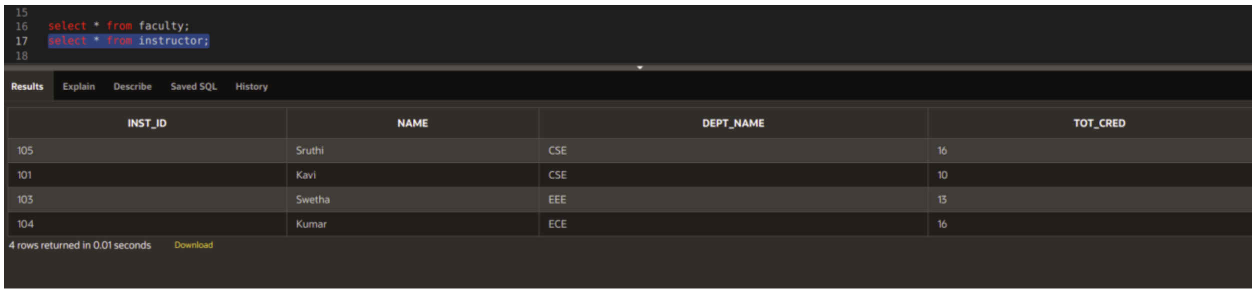
select * from faculty;
select * from instructor;



```
delete from faculty where dept_name='IT';

select * from faculty;
select * from instructor;
```

INST_ID	NAME	DEPT_NAME
105	Sruthi	CSE
101	Kavi	CSE
103	Swetha	EEE
104	Kumar	ECE

4 rows returned in 0.01 seconds



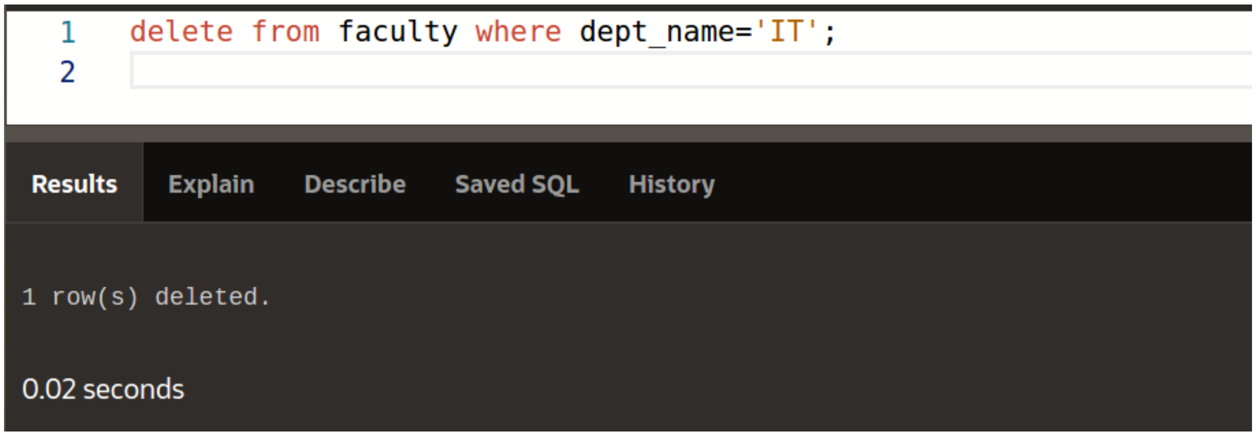
```
select * from faculty;
select * from instructor;
```

INST_ID	NAME	DEPT_NAME	TOT_CRED
105	Sruthi	CSE	16
101	Kavi	CSE	10
103	Swetha	EEE	13
104	Kumar	ECE	16

4 rows returned in 0.01 seconds

4.Delete a record from faculty relation having dept_name as IT

delete from faculty where dept_name='IT';



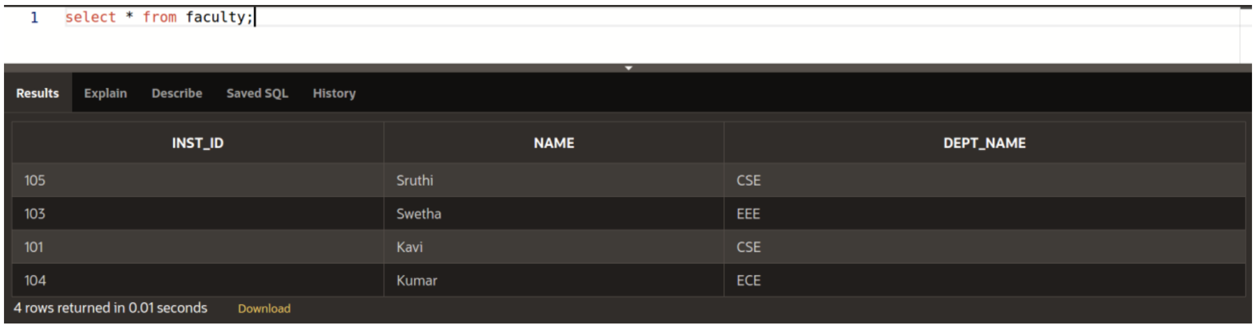
```
delete from faculty where dept_name='IT';
```

```
1 row(s) deleted.
```

0.02 seconds

5.Retrieve the record from both instructor and faculty relation

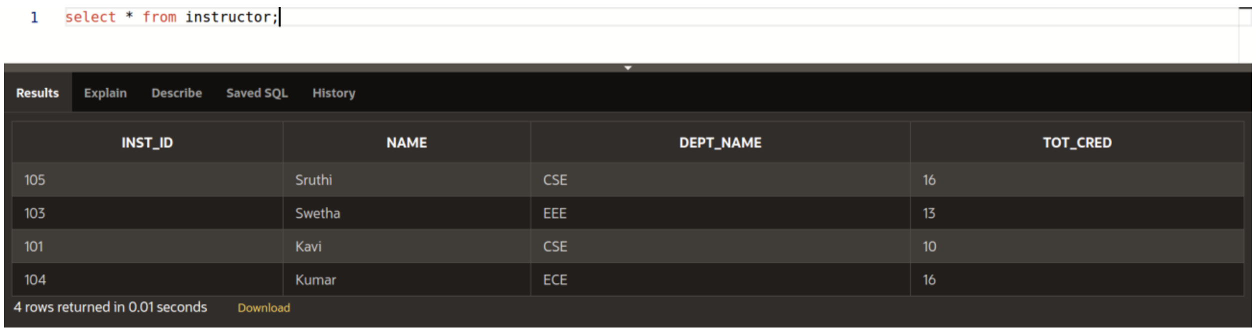
select * from faculty;
select * from instructor;



```
select * from faculty;
```

INST_ID	NAME	DEPT_NAME
105	Sruthi	CSE
103	Swetha	EEE
101	Kavi	CSE
104	Kumar	ECE

4 rows returned in 0.01 seconds



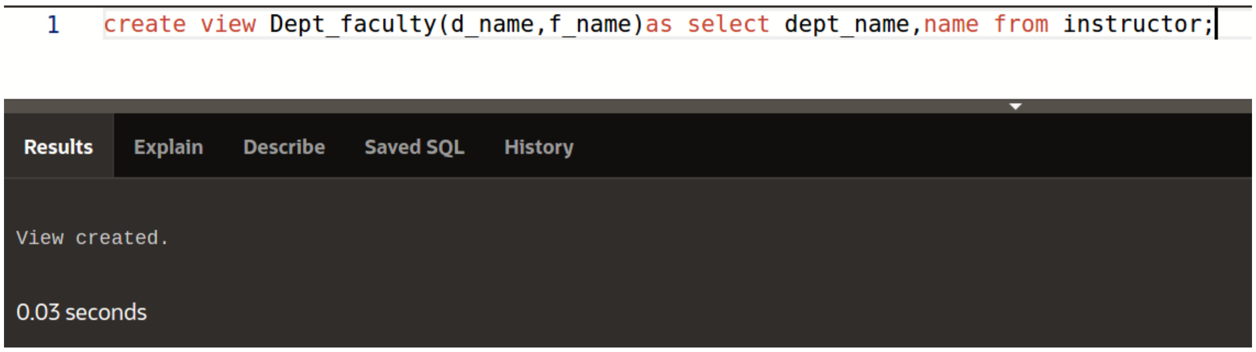
```
select * from instructor;
```

INST_ID	NAME	DEPT_NAME	TOT_CRED
105	Sruthi	CSE	16
103	Swetha	EEE	13
101	Kavi	CSE	10
104	Kumar	ECE	16

4 rows returned in 0.01 seconds

6.Create a view in the name of dept_faculty(d_name,f_name) as select dept_name,name from instructor.

create view Dept_faculty(d_name,f_name)as select dept_name,name from instructor;



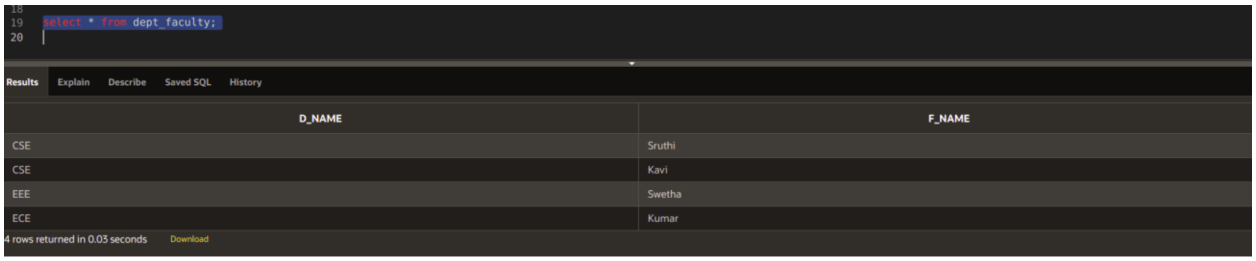
```
create view Dept_faculty(d_name,f_name)as select dept_name,name from instructor;
```

```
View created.
```

0.03 seconds

7.Retrieve the records from dept_faculty

select * from dept_faculty;



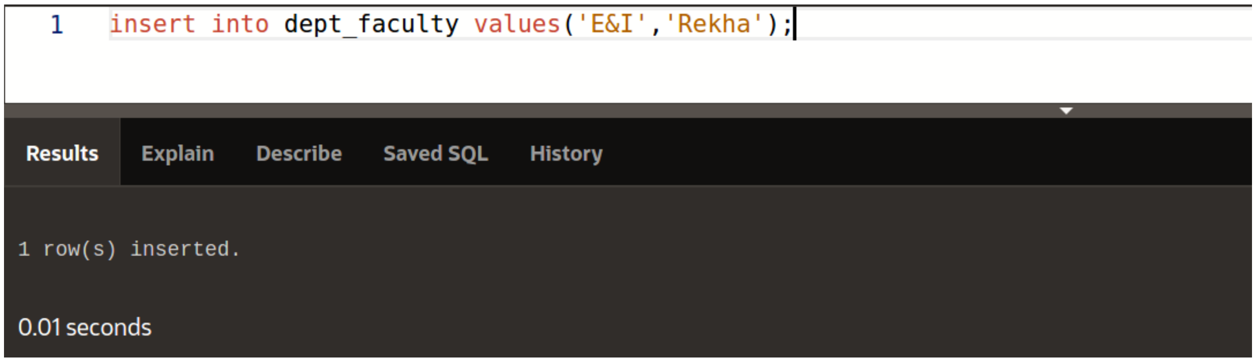
```
select * from dept_faculty;
```

D_NAME	F_NAME
CSE	Sruthi
CSE	Kavi
EEE	Swetha
ECE	Kumar

4 rows returned in 0.03 seconds

8.Insert values into the view relation dept_faculty . Retrieve the records from dept_faculty

insert into dept_faculty values('E&I','Rekha');

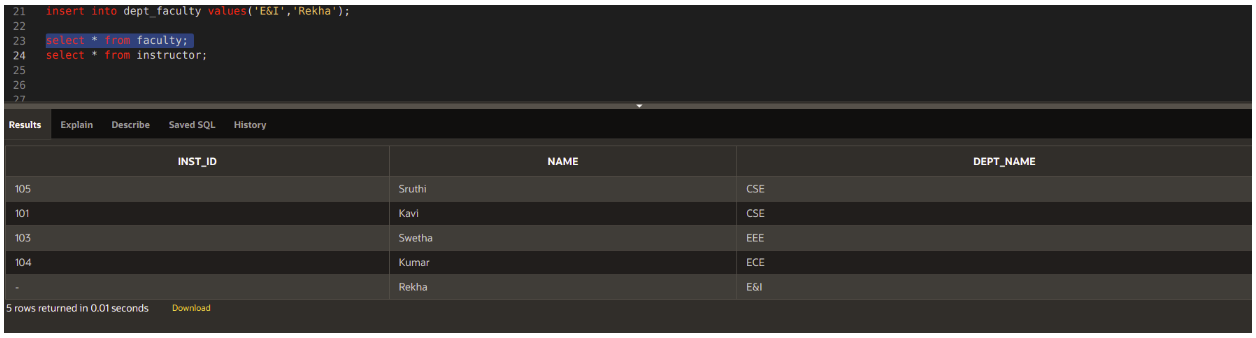


```
insert into dept_faculty values('E&I','Rekha');
```

```
1 row(s) inserted.

0.01 seconds
```

9.Retrieve the record from both instructor and faculty relation

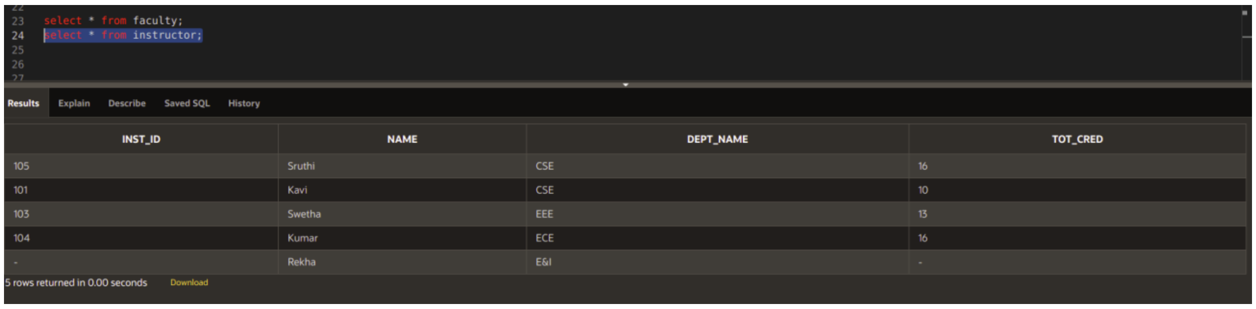
select * from faculty;
select * from instructor;



```
insert into dept_faculty values('E&I','Rekha');

select * from faculty;
select * from instructor;
```

INST_ID	NAME	DEPT_NAME
105	Sruthi	CSE
101	Kavi	CSE
103	Swetha	EEE
104	Kumar	ECE
-	Rekha	E&I

5 rows returned in 0.01 seconds



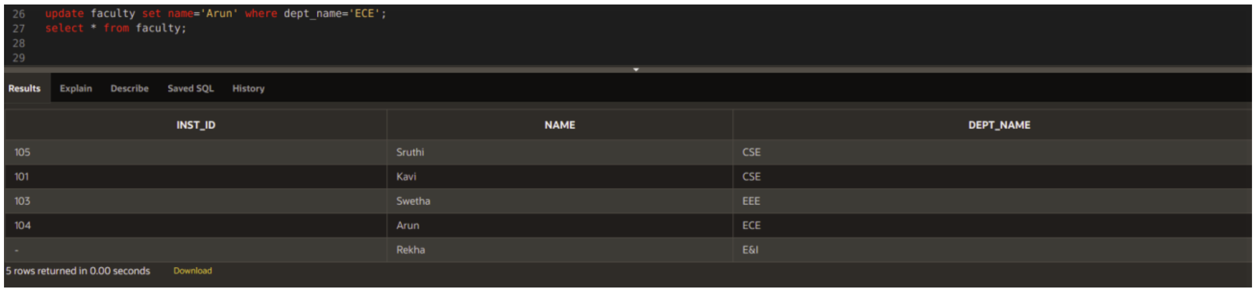
```
select * from faculty;
select * from instructor;
```

INST_ID	NAME	DEPT_NAME	TOT_CRED
105	Sruthi	CSE	16
101	Kavi	CSE	10
103	Swetha	EEE	13
104	Kumar	ECE	16
-	Rekha	E&I	-

5 rows returned in 0.00 seconds

10.Update the name=’arun’ from faculty relation whose dept_name=’ECE’

update faculty set name='Arun' where dept_name='ECE';
select * from faculty;



```
update faculty set name='Arun' where dept_name='ECE';
select * from faculty;
```

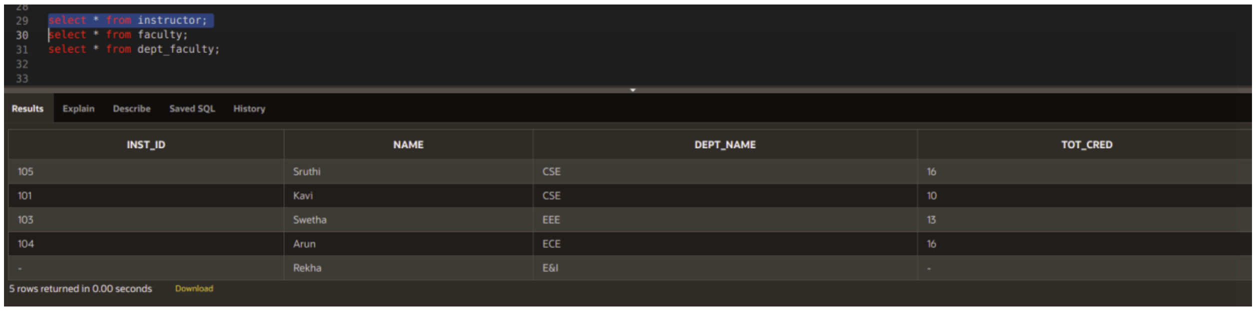
Results Explain Describe Saved SQL History

INST_ID	NAME	DEPT_NAME
105	Sruthi	CSE
101	Kavi	CSE
103	Swetha	EEE
104	Arun	ECE
-	Rekha	E&I

5 rows returned in 0.00 seconds Download

11.Retrieve the record from instructor, faculty&dept_faculty relation

select * from instructor;
select * from faculty;
select * from dept_faculty;

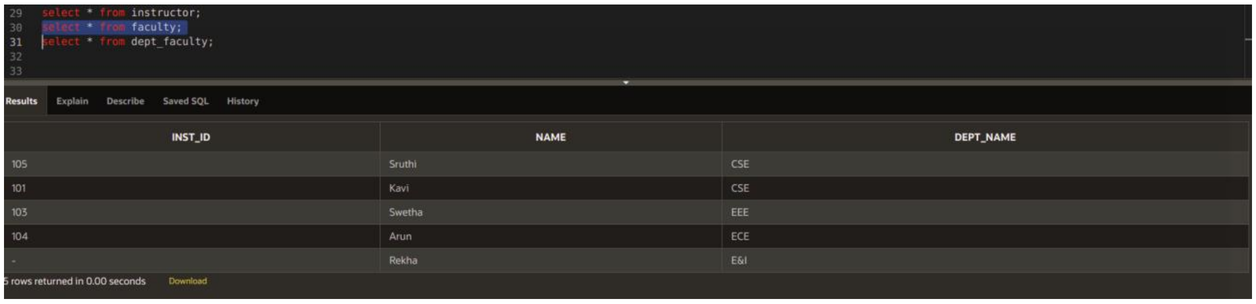


```
select * from instructor;
select * from faculty;
select * from dept_faculty;
```

Results Explain Describe Saved SQL History

INST_ID	NAME	DEPT_NAME	TOT_CRED
105	Sruthi	CSE	16
101	Kavi	CSE	10
103	Swetha	EEE	13
104	Arun	ECE	16
-	Rekha	E&I	-

5 rows returned in 0.00 seconds Download

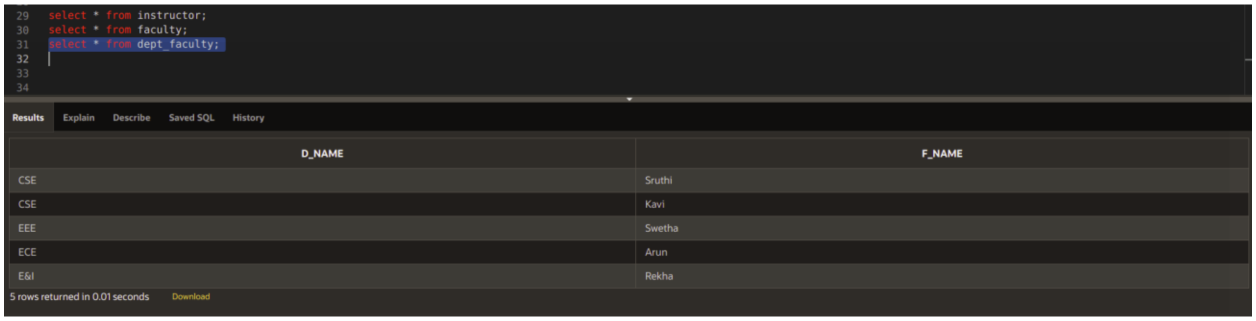


```
select * from instructor;
select * from faculty;
select * from dept_faculty;
```

Results Explain Describe Saved SQL History

INST_ID	NAME	DEPT_NAME
105	Sruthi	CSE
101	Kavi	CSE
103	Swetha	EEE
104	Arun	ECE
-	Rekha	E&I

5 rows returned in 0.00 seconds Download



```
select * from instructor;
select * from faculty;
select * from dept_faculty;
```

Results Explain Describe Saved SQL History

D_NAME	F_NAME
CSE	Sruthi
CSE	Kavi
EEE	Swetha
ECE	Arun
E&I	Rekha

5 rows returned in 0.01 seconds Download

Result:

Thus, views operations are executed successfully.

EX.NO: 09

Practice of named PL/SQL blocks (Procedure, Function)

Aim:

To create PL/SQL programs to implement procedures, functions

Learning Outcomes:

After the completion of this experiment, student will be able to

- Create procedures for different kinds of data by invoking different types of parameters.
- Understand how functions are used in PL/SQL.

Problem Statement:

To implement the various procedures and functions like,

1. IN parameter
2. OUT parameter
3. IN OUT parameter

Procedures:

1. IN: It specifies that a value for the argument must be specified when calling the procedure ie., used to pass values to a sub-program. This is the default parameter.

2. OUT: It specifies that the procedure passes a value for this argument back to it's calling environment after execution ie. used to return values to a caller of the sub-program.

3. IN OUT: It specifies that a value for the argument must be specified when calling the procedure and that procedure passes a value for this argument back to it's calling environment after execution.

PROCEDURES

<u>Syntax:</u>

create or replace procedure <procedure name> (argument **{in,out,inout}** datatype) {is,as}

variable declaration;

constant

declaration; begin

PL/SQL subprogram body;

exception

exception PL/SQL block;

end;

FUNCTIONS

<u>Syntax:</u>

create or replace function <function name> (argument in datatype,……) return datatype {is,as}

variable declaration;

constant declaration;

begin

PL/SQL subprogram body;

exception

exception PL/SQL block;

end;

<u>Sample Coding and Execution of the Program:</u>

1. //Square of a number using 'in out' parameter//

```
SQL> declare
a number(3);
create procedure squarenum(x in out
number)
is
begin
x:=x*x;
```

```
  end;
  begin
  a:=23;
  squarenum(a);
 dbms_output.put_line('square
 of(23):'||a); end;
  /
```

2.//Factorial of a number using functions//

```
SQL> create or replace function fact1


  return    number    is
  fnum
  number(5):=1;        i
  number(3);
 begin
 for   i   in   1..5
 loop
 fnum:=fnum*i
 ; end loop;
 return fnum;
 end;
  /
Function   created.
SQL> declare
  c number(5);
  begin
  c:=fact1()
  ;

dbms_output.put_line('factorial is:'||c); end;
```

Sample Output:

1. square of(23):529

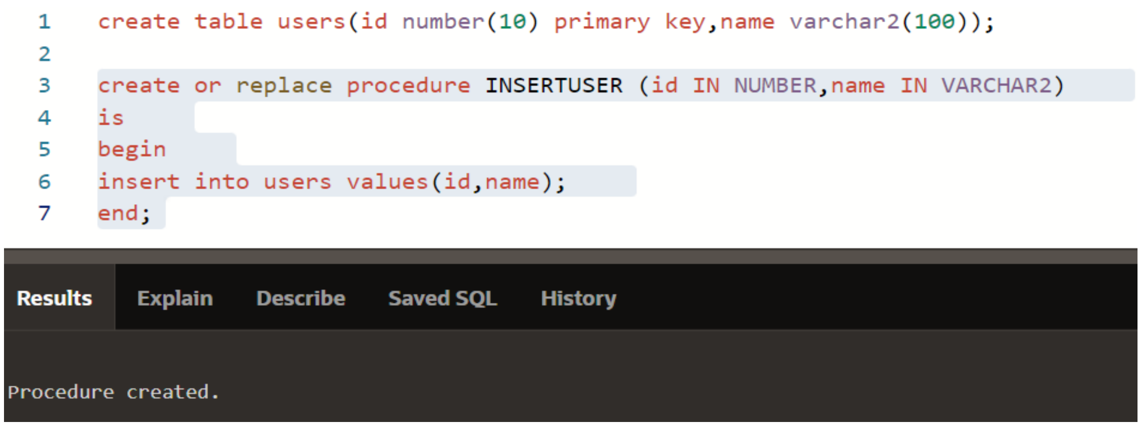
PL/SQL procedure successfully completed.

2. factorial is:120

PL/SQL procedure successfully completed.

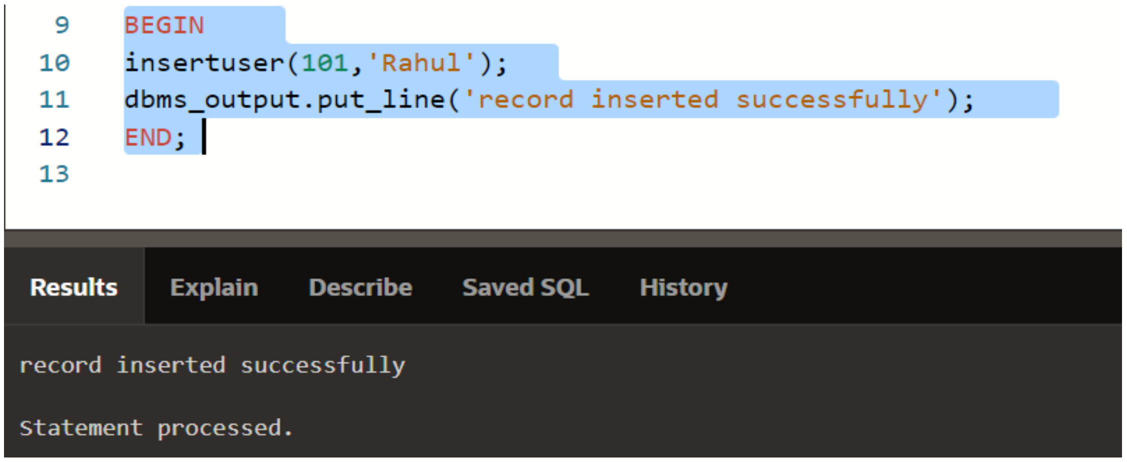
Test Cases:

1. Write a procedure to insert a row in users using "IN" parameters.



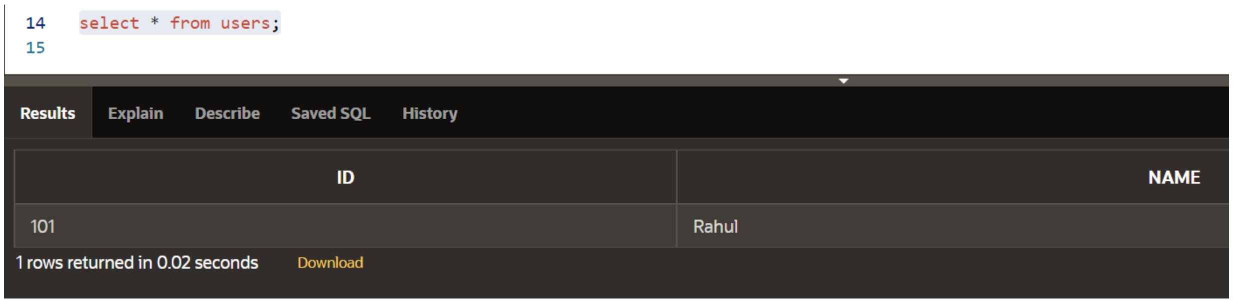
```
create table users(id number(10) primary key,name varchar2(100));

create or replace procedure INSERTUSER (id IN NUMBER,name IN VARCHAR2)
is
begin
insert into users values(id,name);
end;
```

```
Procedure created.
```



```
BEGIN
insertuser(101,'Rahul');
dbms_output.put_line('record inserted successfully');
END;
```

```
record inserted successfully

Statement processed.
```

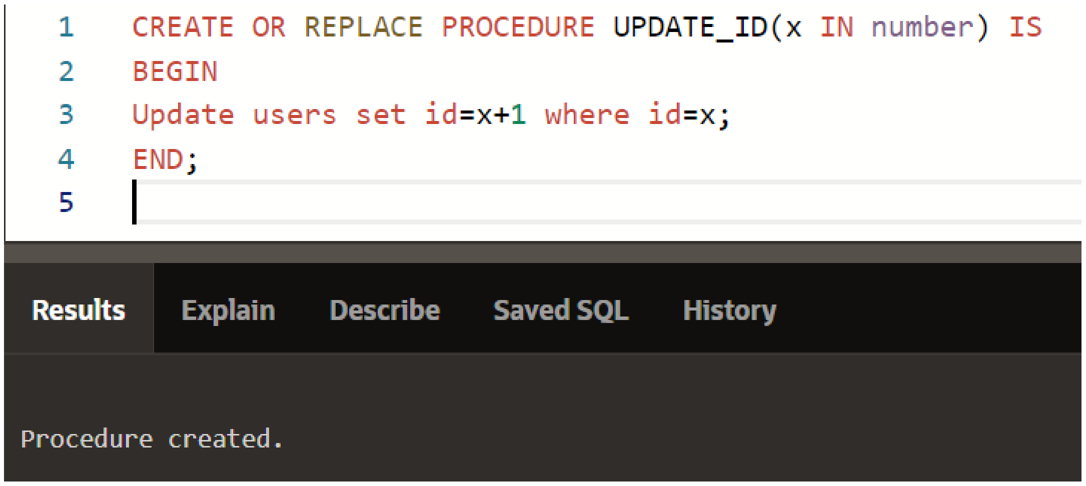


```
select * from users;
```

ID	NAME
101	Rahul

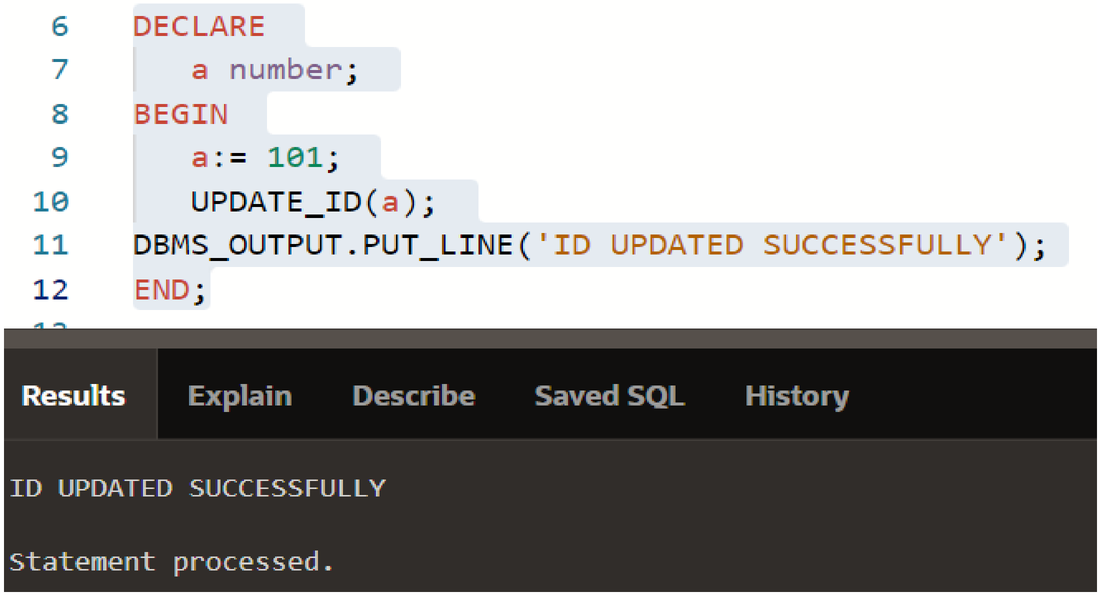
1 rows returned in 0.02 seconds Download

2. Create a stored procedure to update the id value by 1 using "IN" Parameter.



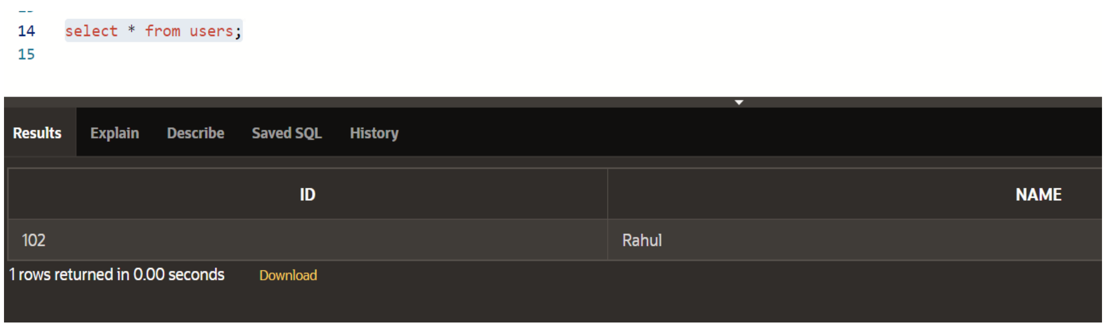
```
CREATE OR REPLACE PROCEDURE UPDATE_ID(x IN number) IS
BEGIN
Update users set id=x+1 where id=x;
END;
```

```
Procedure created.
```



```
DECLARE
    a number;
BEGIN
    a:= 101;
    UPDATE_ID(a);
DBMS_OUTPUT.PUT_LINE('ID UPDATED SUCCESSFULLY');
END;
```

```
ID UPDATED SUCCESSFULLY

Statement processed.
```

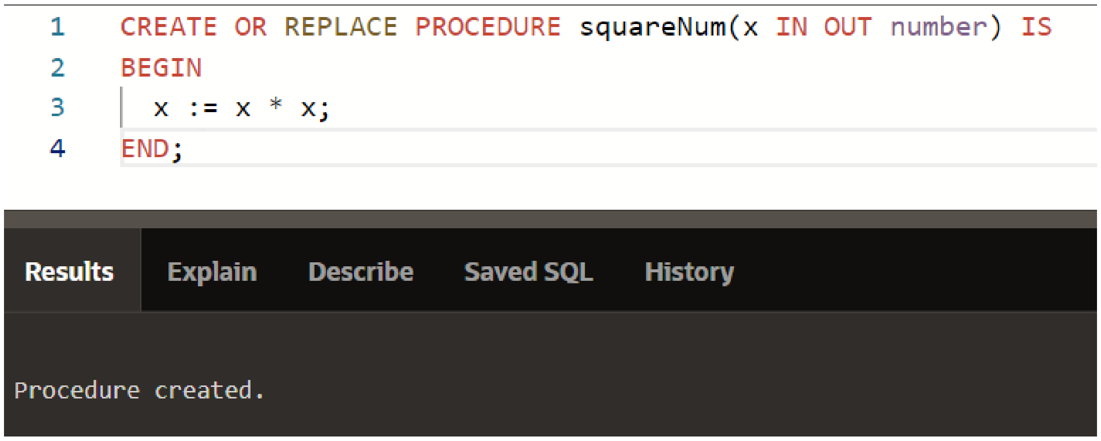


```
--
select * from users;
```

ID	NAME
102	Rahul

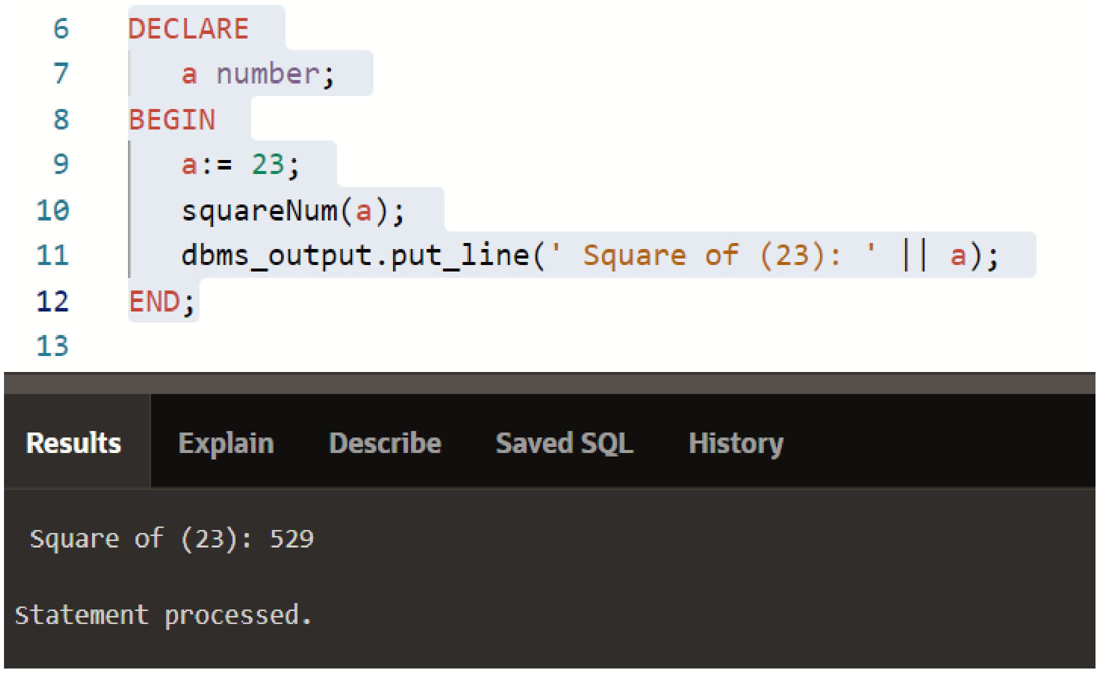
```
1 rows returned in 0.00 seconds    Download
```

3. Create a stored procedure to find the square of the given 'INOUT' parameter.



```
CREATE OR REPLACE PROCEDURE squareNum(x IN OUT number) IS
BEGIN
  x := x * x;
END;
```

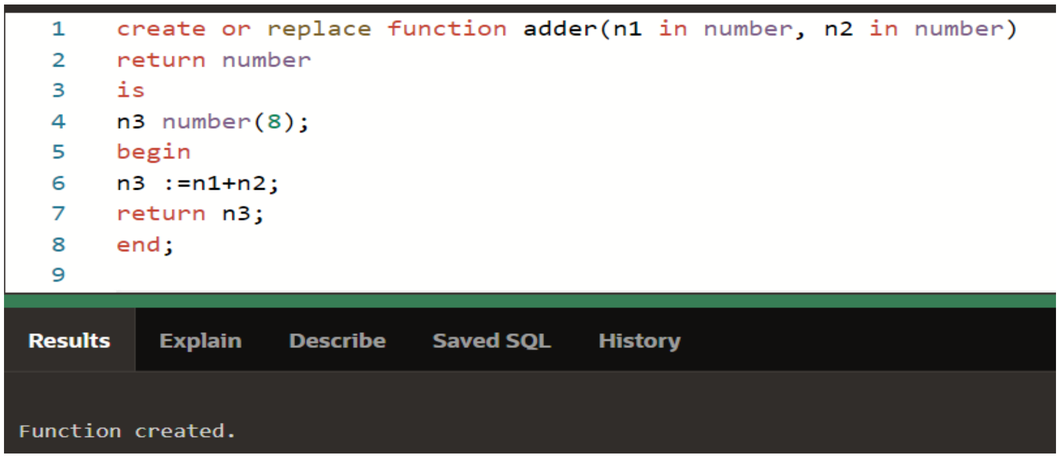
```
Procedure created.
```



```
DECLARE
    a number;
BEGIN
    a:= 23;
    squareNum(a);
    dbms_output.put_line(' Square of (23): ' || a);
END;
```

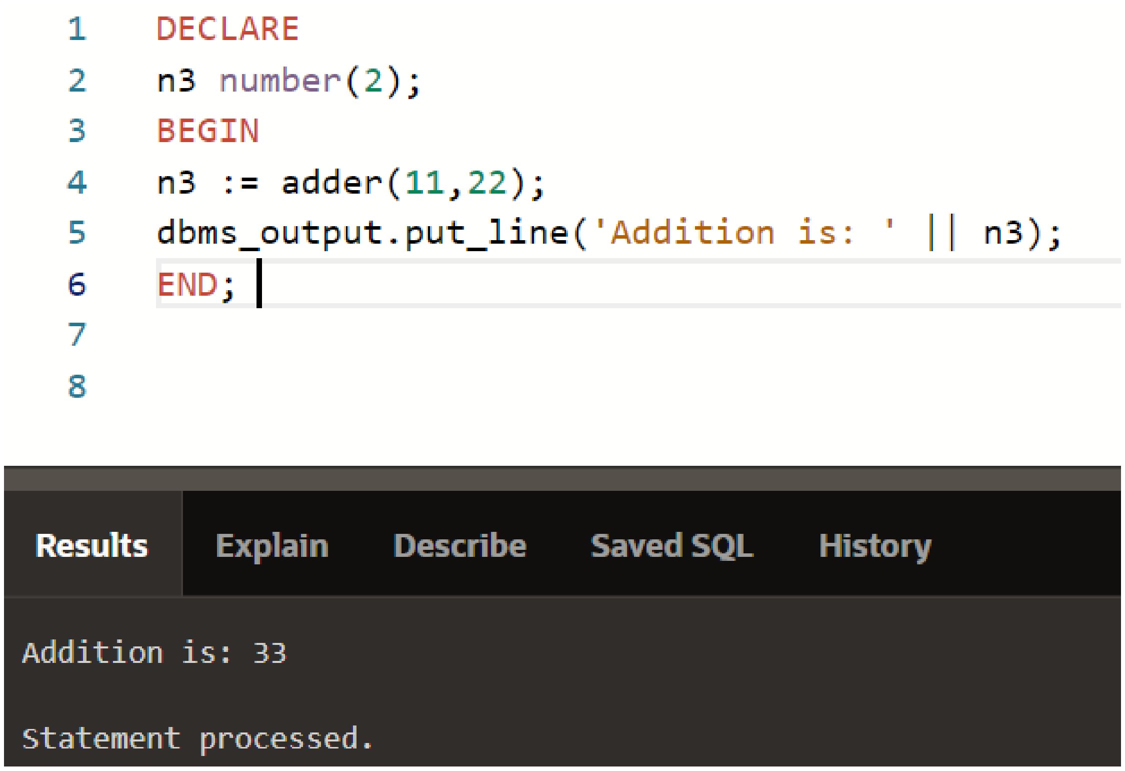
```
Square of (23): 529

Statement processed.
```

4. Create a function for addition of 2 numbers and return the sum.



```
create or replace function adder(n1 in number, n2 in number)
return number
is
n3 number(8);
begin
n3 :=n1+n2;
return n3;
end;
```

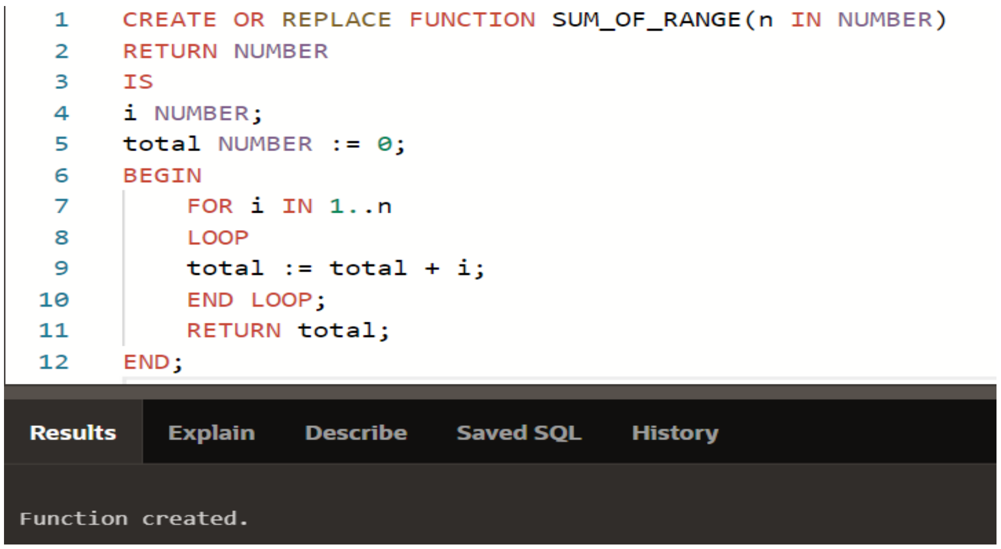
```
Function created.
```



```
DECLARE
n3 number(2);
BEGIN
n3 := adder(11,22);
dbms_output.put_line('Addition is: ' || n3);
END;
```

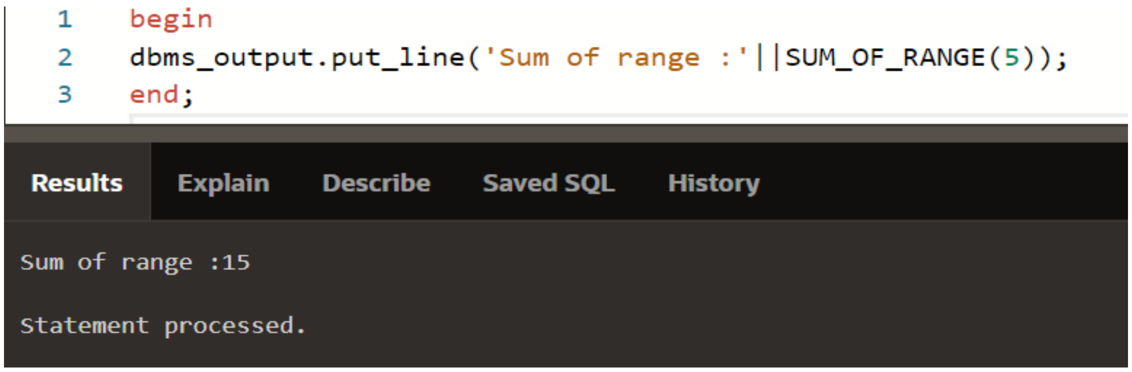
```
Addition is: 33

Statement processed.
```

5. Create a function for finding the sum of n natural numbers using "IN" parameter.



```
CREATE OR REPLACE FUNCTION SUM_OF_RANGE(n IN NUMBER)
RETURN NUMBER
IS
i NUMBER;
total NUMBER := 0;
BEGIN
    FOR i IN 1..n
    LOOP
    total := total + i;
    END LOOP;
    RETURN total;
END;
```

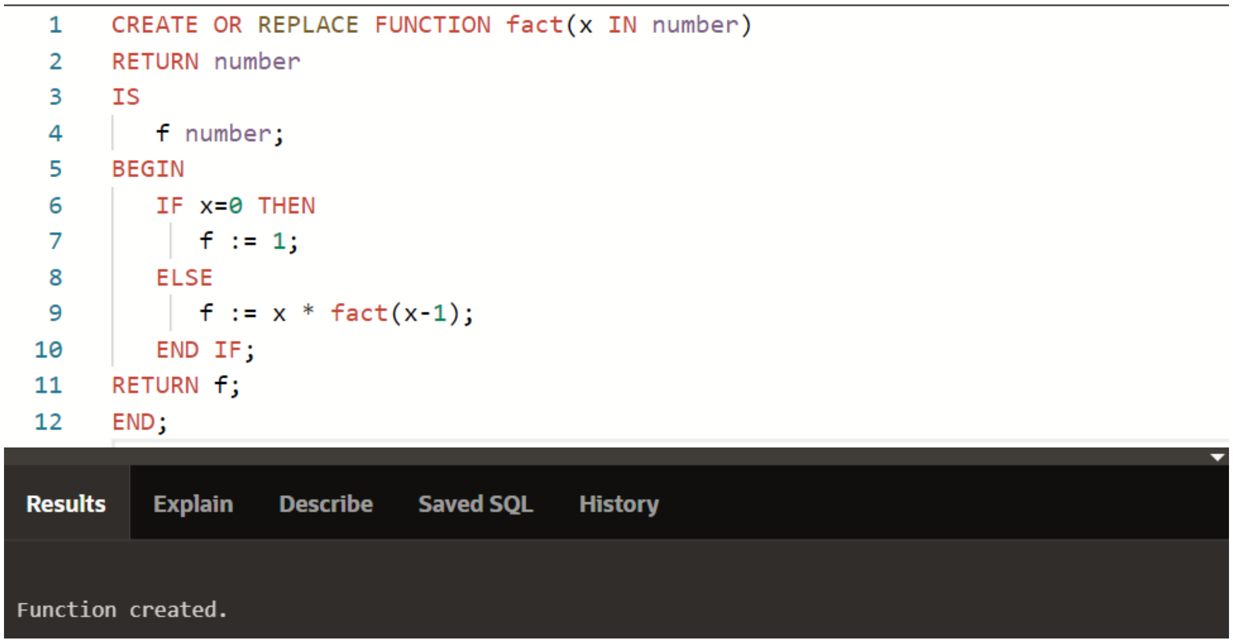
```
Function created.
```



```
begin
dbms_output.put_line('Sum of range :'||SUM_OF_RANGE(5));
end;
```

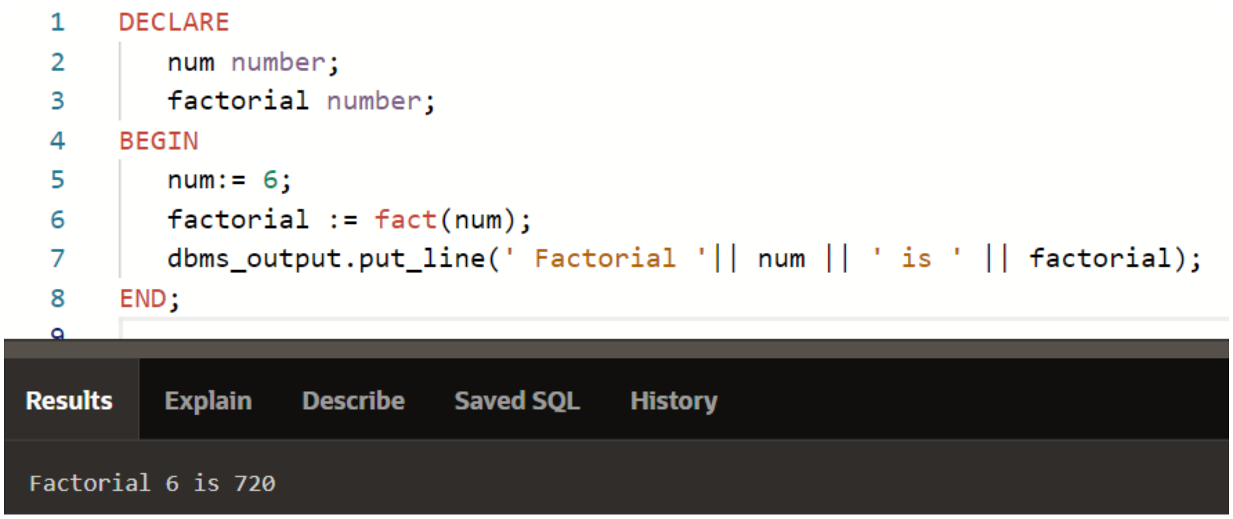
```
Sum of range :15

Statement processed.
```

6. Create a Recursive function to find factorial of a given number.



```sql
CREATE OR REPLACE FUNCTION fact(x IN number)
RETURN number
IS
   f number;
BEGIN
   IF x=0 THEN
      f := 1;
   ELSE
      f := x * fact(x-1);
   END IF;
RETURN f;
END;
```

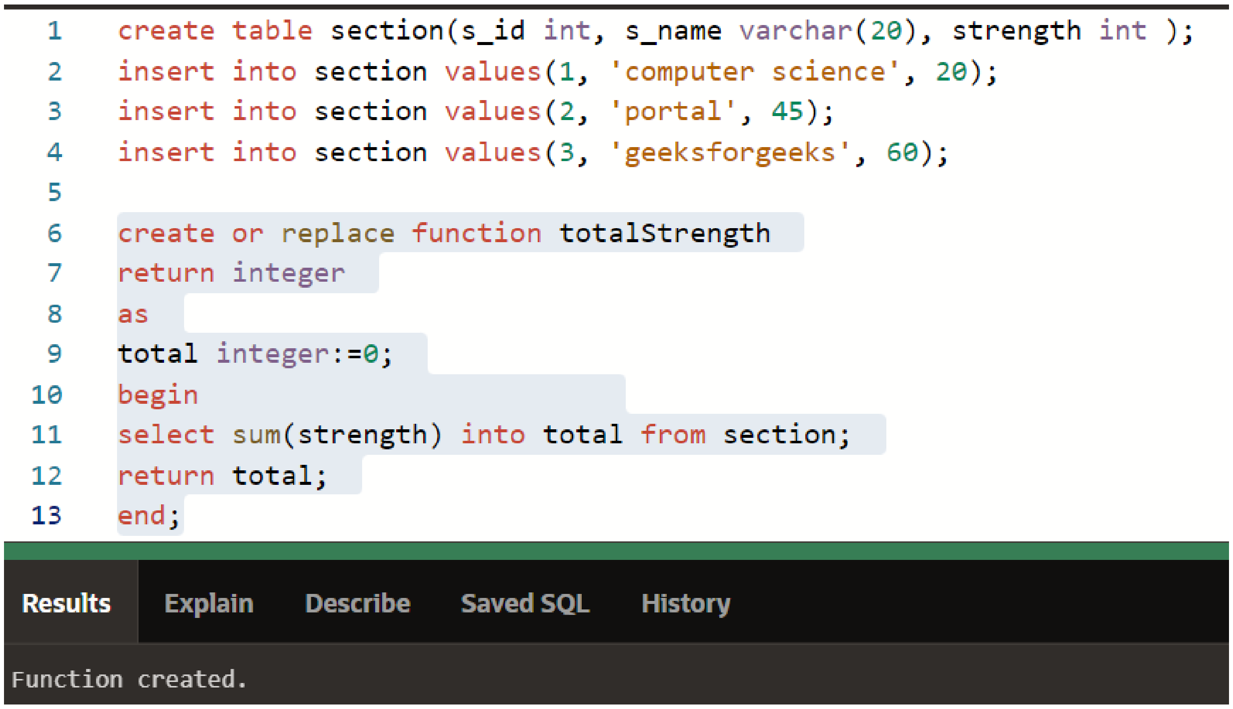
```
Function created.
```



```sql
DECLARE
   num number;
   factorial number;
BEGIN
   num:= 6;
   factorial := fact(num);
   dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
```

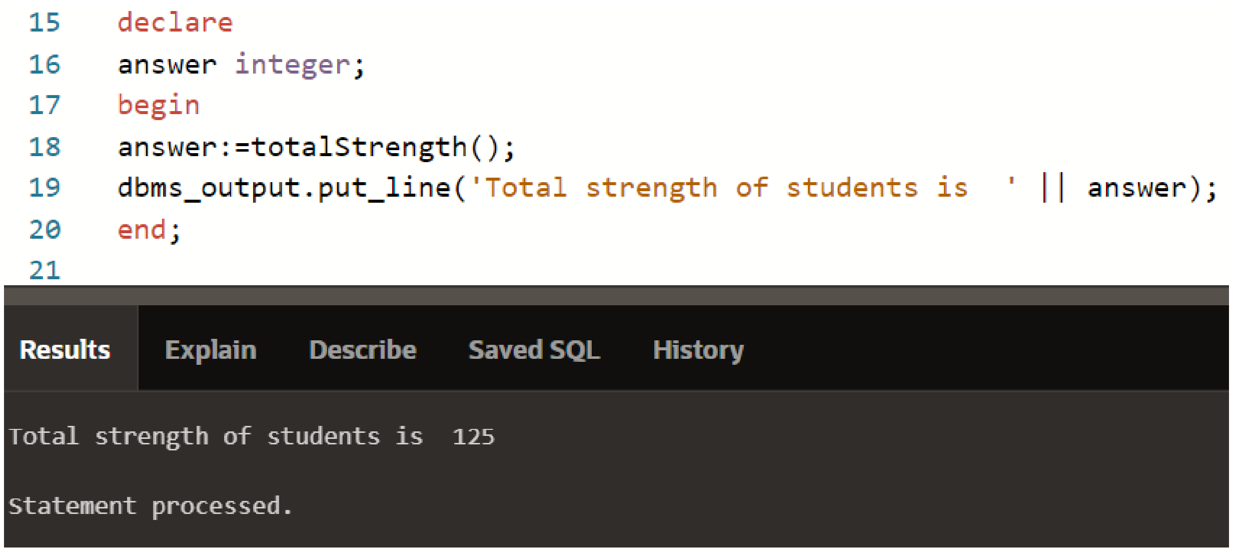
```
Factorial 6 is 720
```

7. Write a function to return total strength of students from section table.



```
create table section(s_id int, s_name varchar(20), strength int );
insert into section values(1, 'computer science', 20);
insert into section values(2, 'portal', 45);
insert into section values(3, 'geeksforgeeks', 60);

create or replace function totalStrength
return integer
as
total integer:=0;
begin
select sum(strength) into total from section;
return total;
end;
```

```
Function created.
```



```
declare
answer integer;
begin
answer:=totalStrength();
dbms_output.put_line('Total strength of students is  ' || answer);
end;
```

```
Total strength of students is  125

Statement processed.
```

Result:

Thus the creation of PL/SQL programs to implement procedures, functions, were successfully executed.

EX.NO: 10

Implementation of Triggers using PL/SQL

Aim:

To create PL/SQL programs to implement Triggers.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Access both the Row level and state level trigger.

TRIGGERS:

A Trigger is a stored procedure that defines an action that the database automatically takes when some database-related event such as insert, update or delete occurs.

TYPES OF TRIGGERS

The various types of triggers are as follows,

Before: It fires the trigger before executing the trigger statement.

After: It fires the trigger after executing the trigger statement.

For each row: It specifies that the trigger fires once per row.

For each statement: This is the default trigger that is invoked. It specifies that the trigger fires once per statement.

VARIABLES USED IN TRIGGERS

1. :new
2. :old

These two variables retain the new and old values of the column updated in the database. The values in these variables can be used in the database triggers for data manipulation.

Syntax:

Create or replace trigger <trg_name> <Before /After> < Insert/Update/Delete>
[of column_name, column_name….]

```
 on
<table_name
>  [for  each
row]  [when
condition]
begin
statements;
 end;
/
```

Sample Coding & Execution of the Program:

1. //Trigger for User Defined Error Message and does not allow update and insert//

Trigger:

```
SQL> create trigger trigg before insert or update of salary on employee7 for each row
   declare
  triggsal employee7.salary%type;
 begin
 select salary into triggsal from employee7 where eid=12;
 if(:new.salary>triggsal or :new.salary<triggsal) then
 raise_application_error(-20100,'Salary has not been changed');
 end if;
 end;
/
Trigger created.
```

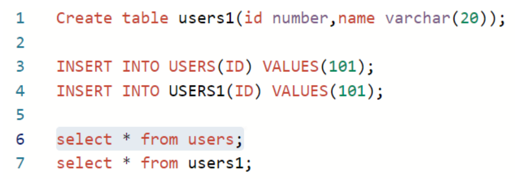
Sample Output:

```
SQL> insert into employee7values ('bbb',16,45000);
insert into employee7 values ('bbb',16,45000)
ERROR at line 1:

ORA-04098: trigger ' employee7.ITTRIGGS' is invalid and failed re-validation
SQL> update employee7 set eid=18 where ename='zzz';
update employee7 set eid=18 where ename='zzz'
```

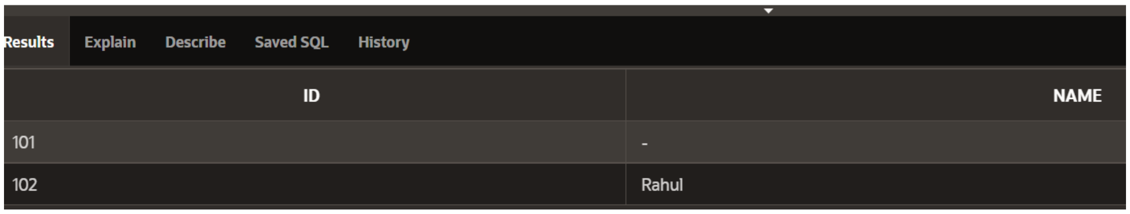
```
               *
ERROR at line 1:
ORA-04298: trigger ' employee7.ITTRIGGS' is invalid and failed re-validation
```

Test Cases:

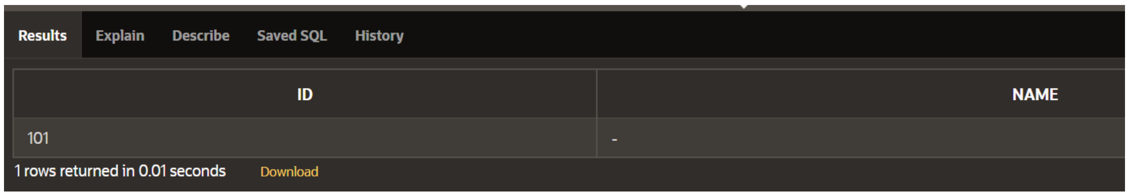
1. If a record is deleted in one table, if the same record is in another table then delete the record in both the tables. Write a trigger for this case.



```
Create table users1(id number,name varchar(20));

INSERT INTO USERS(ID) VALUES(101);
INSERT INTO USERS1(ID) VALUES(101);

select * from users;
select * from users1;
```

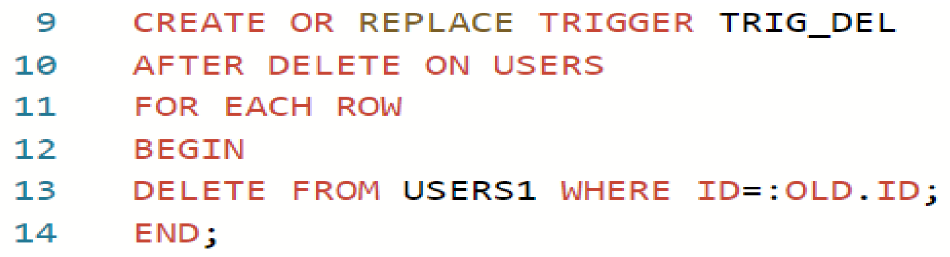


ID	NAME
101	-
102	Rahul

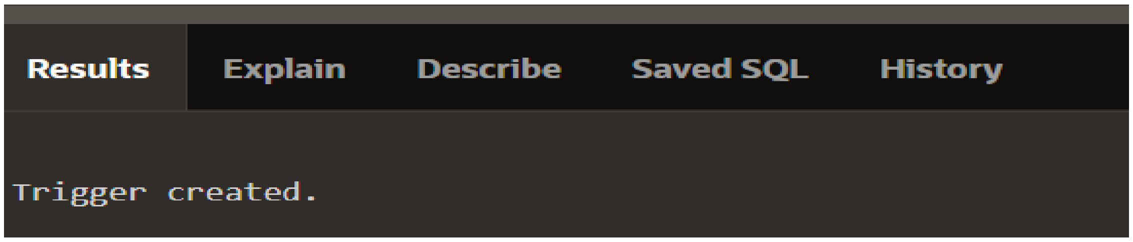


ID	NAME
101	-

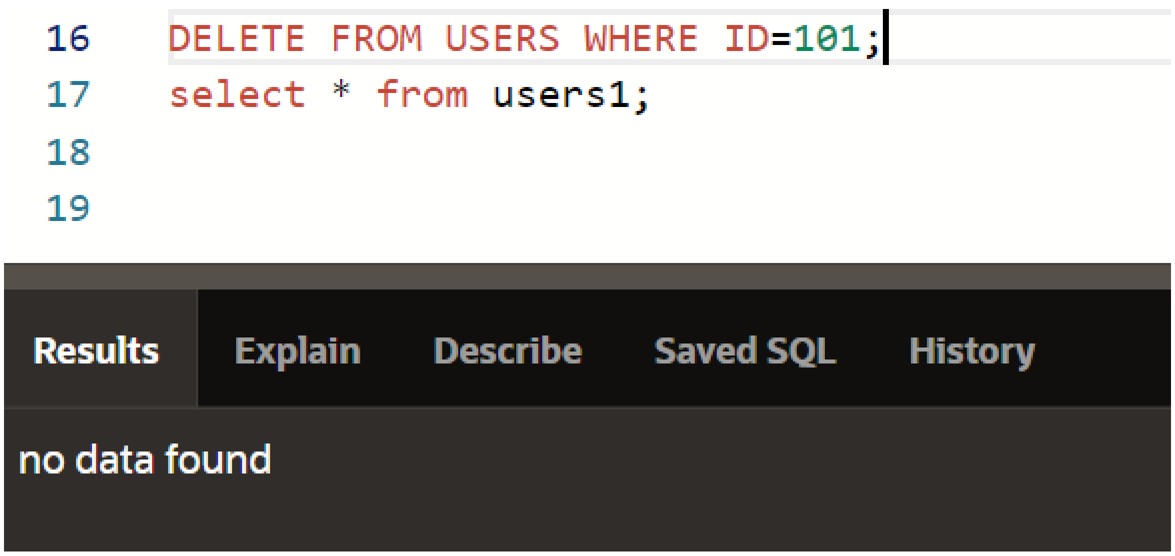
1 rows returned in 0.01 seconds



```
CREATE OR REPLACE TRIGGER TRIG_DEL
AFTER DELETE ON USERS
FOR EACH ROW
BEGIN
DELETE FROM USERS1 WHERE ID=:OLD.ID;
END;
```



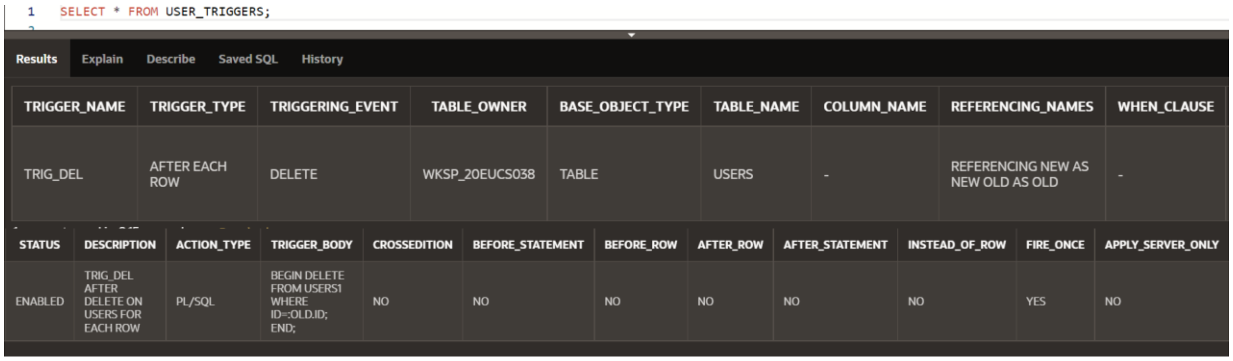
```
Trigger created.
```



```
DELETE FROM USERS WHERE ID=101;
select * from users1;
```

```
no data found
```

2. To view all triggers in workspace created by users.

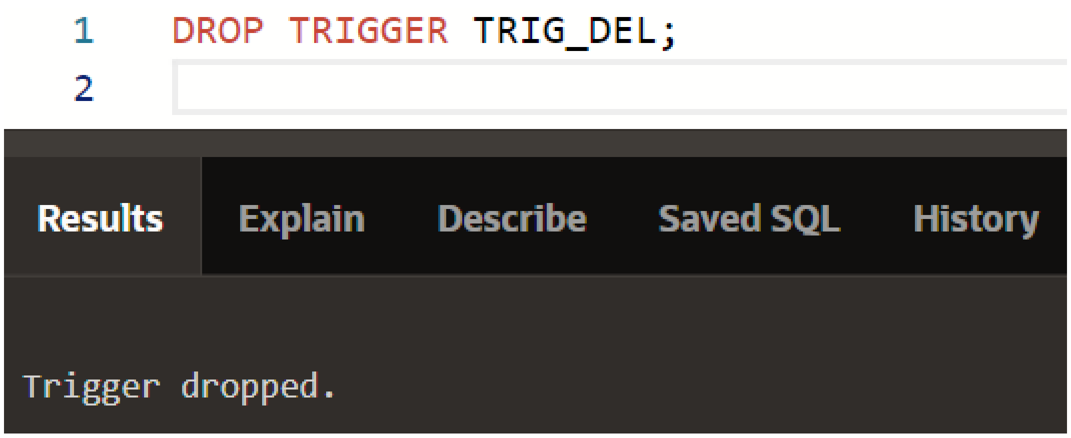


```
SELECT * FROM USER_TRIGGERS;
```

TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_OWNER	BASE_OBJECT_TYPE	TABLE_NAME	COLUMN_NAME	REFERENCING_NAMES	WHEN_CLAUSE
TRIG_DEL	AFTER EACH ROW	DELETE	WKSP_20EUCS038	TABLE	USERS	-	REFERENCING NEW AS NEW OLD AS OLD	-

STATUS	DESCRIPTION	ACTION_TYPE	TRIGGER_BODY	CROSSEDITION	BEFORE_STATEMENT	BEFORE_ROW	AFTER_ROW	AFTER_STATEMENT	INSTEAD_OF_ROW	FIRE_ONCE	APPLY_SERVER_ONLY
ENABLED	TRIG_DEL AFTER DELETE ON USERS FOR EACH ROW	PL/SQL	BEGIN DELETE FROM USERS1 WHERE ID=:OLD.ID; END;	NO	NO	NO	NO	NO	NO	YES	NO

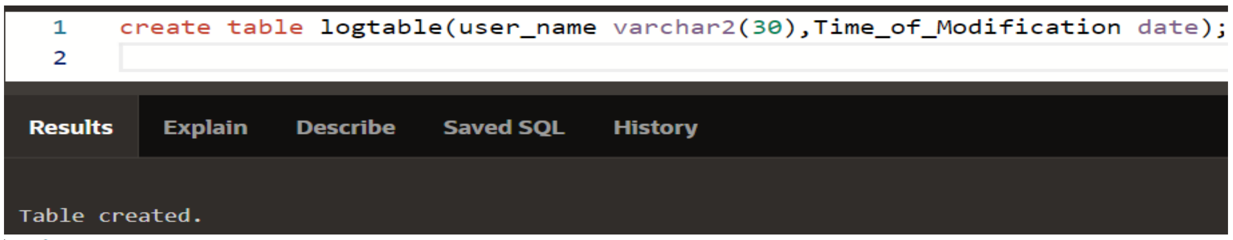
3. To Drop the trigger.



```
DROP TRIGGER TRIG_DEL;
```

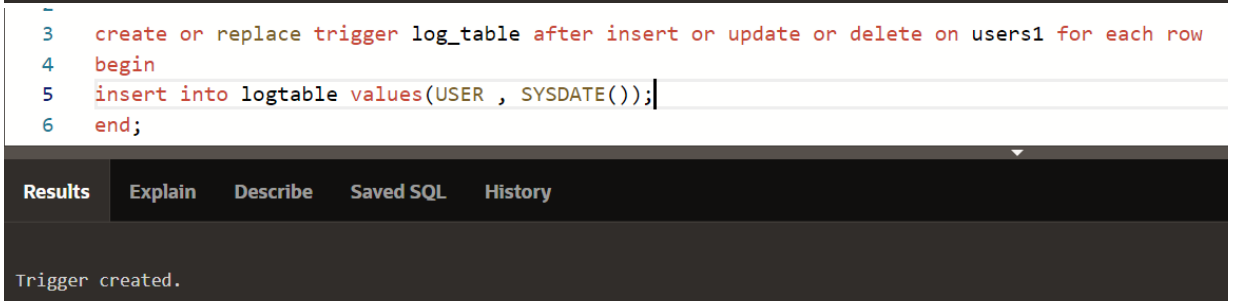
```
Trigger dropped.
```

4. Create an automatic log (username,sysdate) whenever an insertion,updation,deletion happens in an Users1 table.



```sql
create table logtable(user_name varchar2(30),Time_of_Modification date);
```

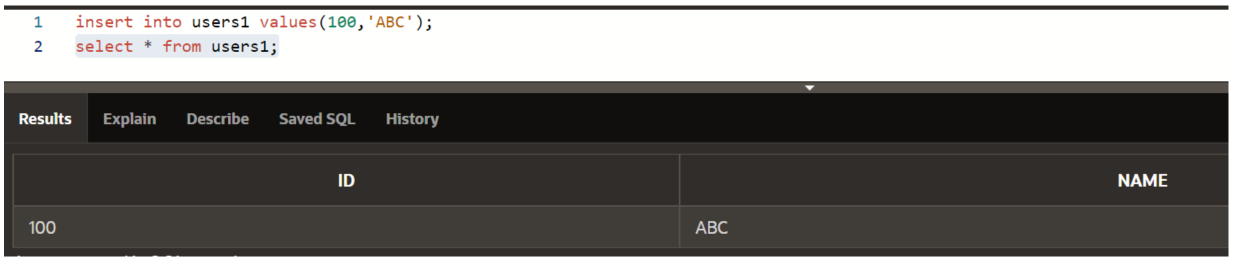
```
Table created.
```



```sql
create or replace trigger log_table after insert or update or delete on users1 for each row
begin
insert into logtable values(USER , SYSDATE());
end;
```

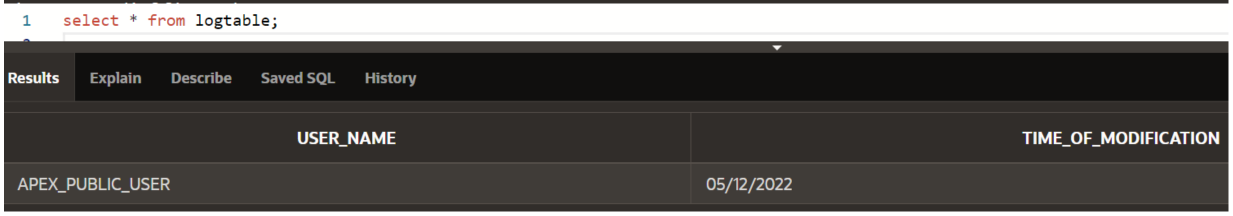
```
Trigger created.
```

insertion:



```sql
insert into users1 values(100,'ABC');
select * from users1;
```

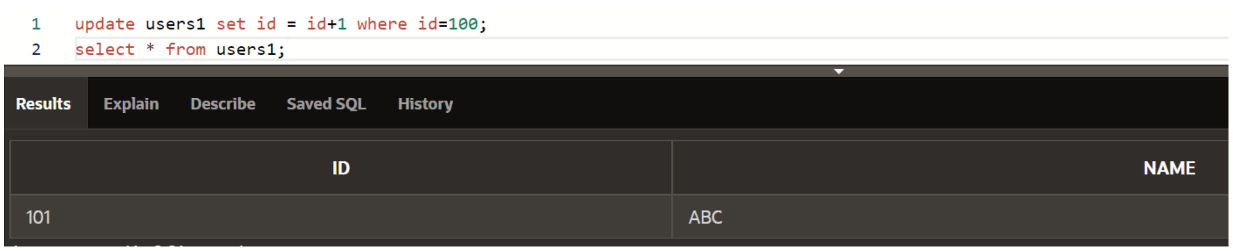
ID	NAME
100	ABC



```sql
select * from logtable;
```

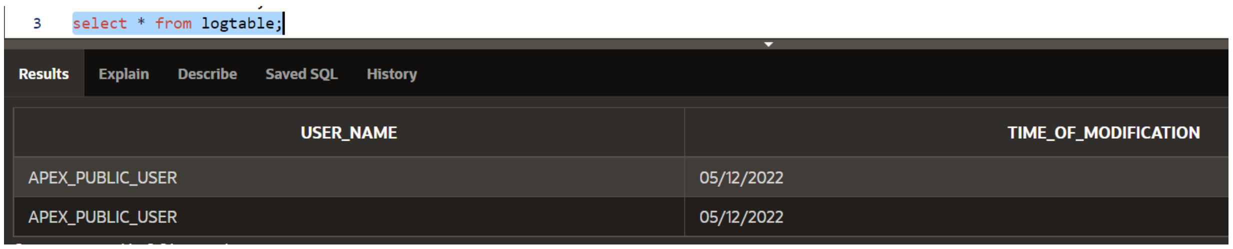
USER_NAME	TIME_OF_MODIFICATION
APEX_PUBLIC_USER	05/12/2022

updation:



```sql
update users1 set id = id+1 where id=100;
select * from users1;
```

ID	NAME
101	ABC



```
select * from logtable;
```

USER_NAME	TIME_OF_MODIFICATION
APEX_PUBLIC_USER	05/12/2022
APEX_PUBLIC_USER	05/12/2022

Result:

Thus the creation of PL/SQL programs to implement Triggers were successfully executed.

EX.NO : 11

Implementation of cursors using PL/SQL

Aim:

To create PL/SQL programs to implement Cursors.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Fetch the single row from database using implicit cursor.
- Fetch the multiple rows from database using explicit cursor.

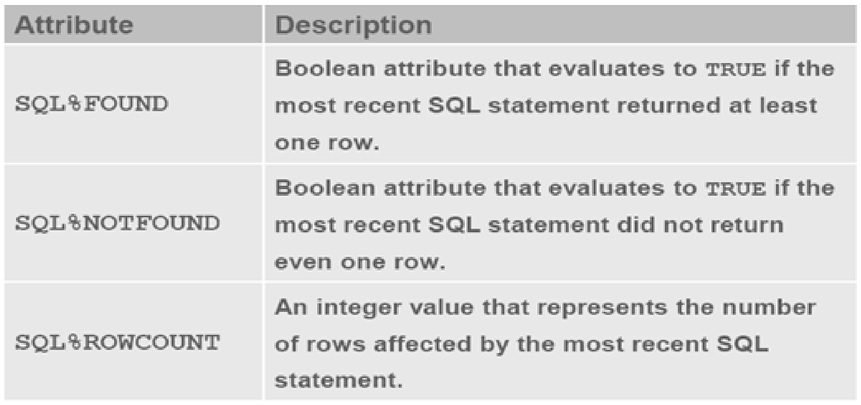
CURSORS:

- The Oracle server allocates a private memory area called a context area to store the data processed by a SQL statement.
- Every context area (and therefore every SQL statement) has a cursor associated with it.

Types of cursor:

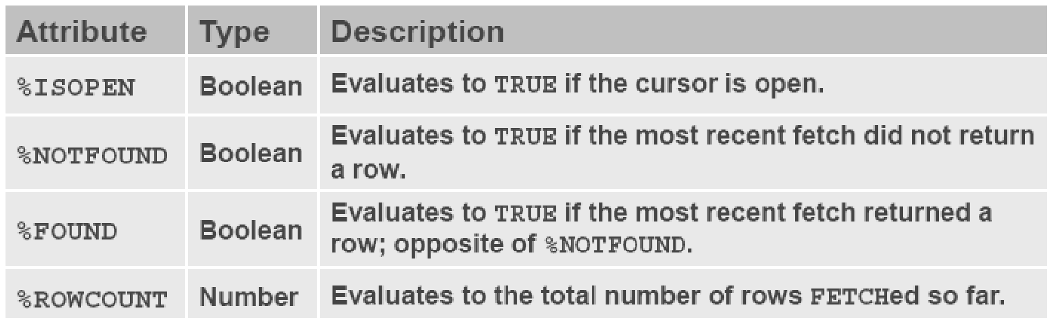
Implicit cursors: Defined automatically by Oracle for all SQL DML statements (INSERT, UPDATE, DELETE, and MERGE), and for SELECT statements that return only one row.
Explicit cursors: Declared by the programmer for queries that return more than one row. You can use explicit cursors to name a context area and access its stored data.

Attributes for Implicit Cursor:



Attribute	Description
SQL%FOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement returned at least one row.
SQL%NOTFOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement did not return even one row.
SQL%ROWCOUNT	An integer value that represents the number of rows affected by the most recent SQL statement.

Attributes for Explicit cursor:



Attribute	Type	Description
%ISOPEN	Boolean	Evaluates to TRUE if the cursor is open.
%NOTFOUND	Boolean	Evaluates to TRUE if the most recent fetch did not return a row.
%FOUND	Boolean	Evaluates to TRUE if the most recent fetch returned a row; opposite of %NOTFOUND.
%ROWCOUNT	Number	Evaluates to the total number of rows FETCHed so far.

Syntax for Explicit cursor:

CURSOR cursor_name IS select_statement;

Cursor_name Is a PL/SQL identifier

Select_statement Is a SELECT statement without an INTO clause

Steps for Using Explicit Cursor:

STEP 1: DECLARE the cursor in the declarative section by naming it and defining the SQL SELECT statement to be associated with it.

STEP 2:OPEN the cursor. This will populate the cursor's active set with the results of the SELECT statement in the cursor's definition. The OPEN statement also positions the cursor pointer at the first row.

STEP 3: FETCH each row from the active set and load the data into variables. After each FETCH, the EXIT WHEN checks to see if the FETCH reached the end of the active set resulting in a data NOTFOUND condition. If the end of the active set was reached, the LOOP is exited.

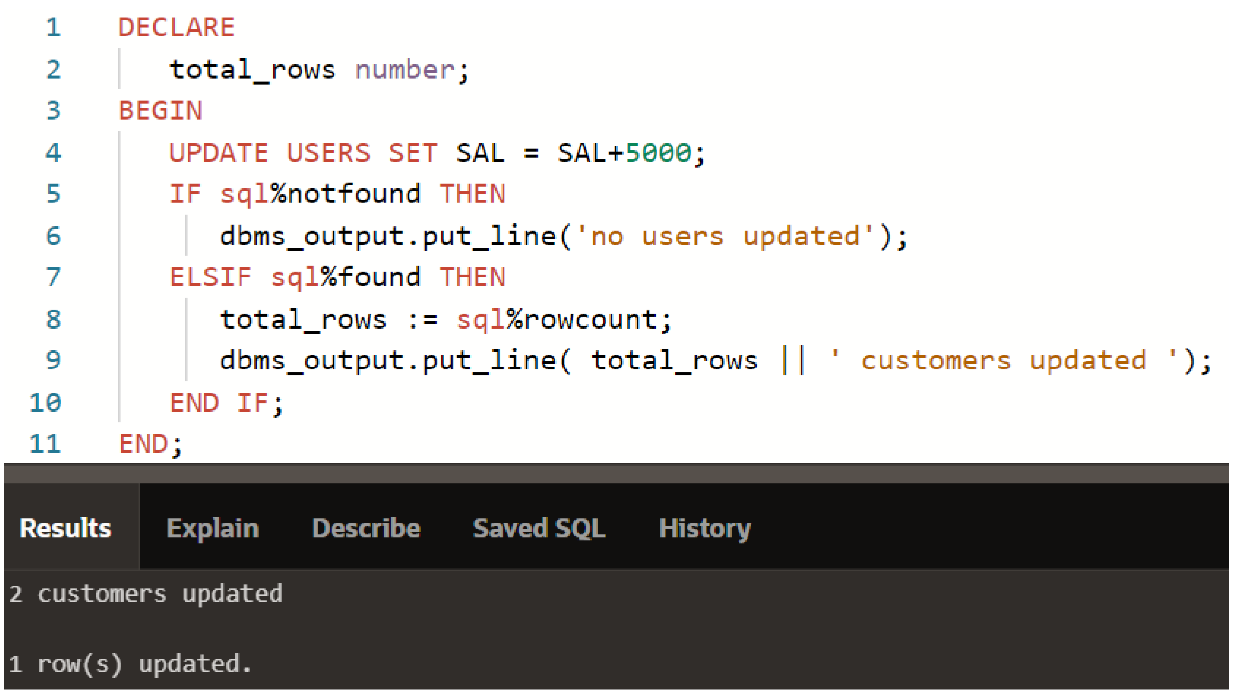
STEP 4:CLOSE the cursor. The CLOSE statement releases the active set of rows. It is now possible to reopen the cursor to establish a fresh active set using a new OPEN statement.

STEPS:

```
DECLARE variables;
 records;
 create a cursor;
 BEGIN
OPEN cursor;
FETCH cursor;
 process the records;
 CLOSE cursor;
END;
```

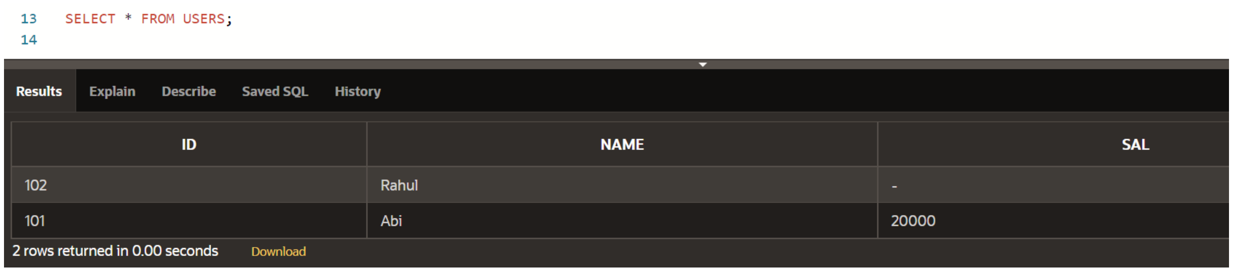
Test Cases:

1. Write an implicit cursor to retrieve the no. of rows affected by the update statement which updates the existing salary by 5000 in users table.



```sql
DECLARE
   total_rows number;
BEGIN
   UPDATE USERS SET SAL = SAL+5000;
   IF sql%notfound THEN
      dbms_output.put_line('no users updated');
   ELSIF sql%found THEN
      total_rows := sql%rowcount;
      dbms_output.put_line( total_rows || ' customers updated ');
   END IF;
END;
```

```
2 customers updated

1 row(s) updated.
```

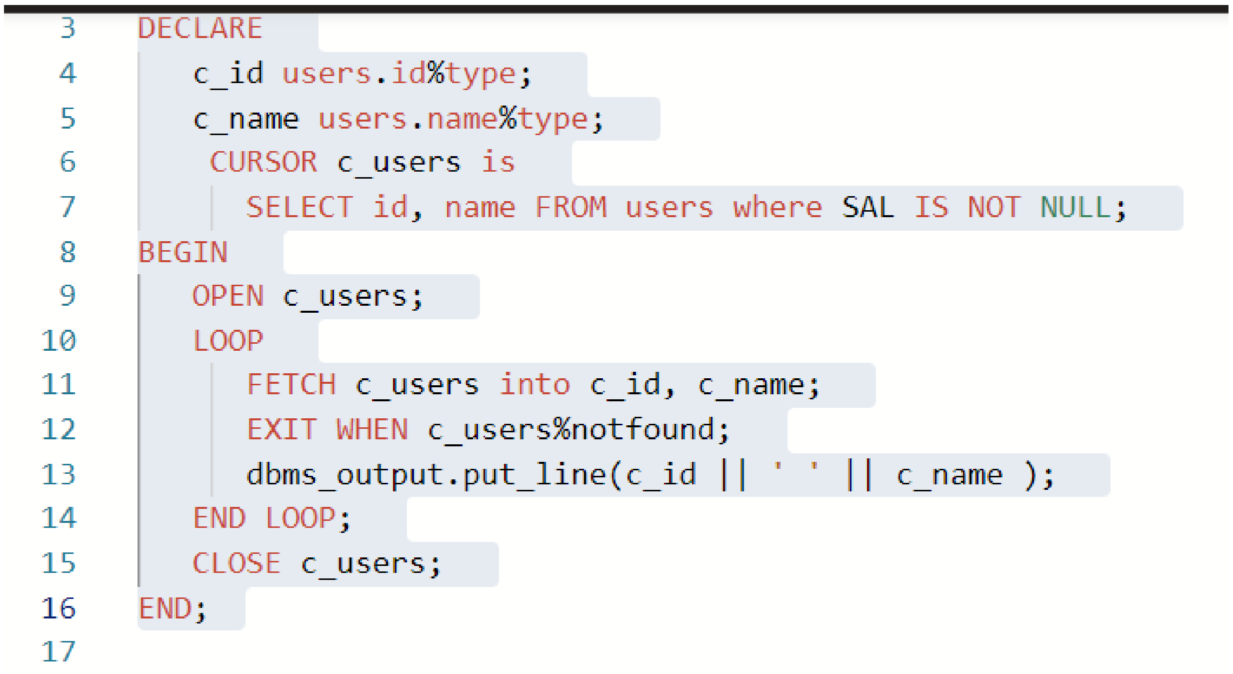


```sql
SELECT * FROM USERS;
```

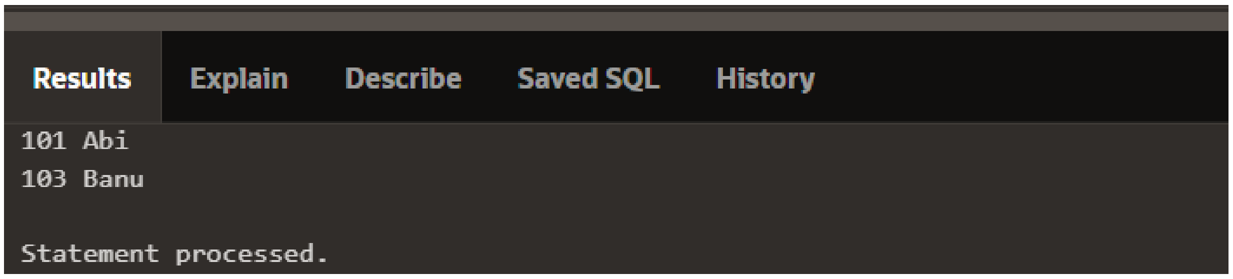
ID	NAME	SAL
102	Rahul	-
101	Abi	20000

2 rows returned in 0.00 seconds

2. Write an Explicit Cursor to retrieve the id and name from users relation if the salary is not null.



```
DECLARE
    c_id users.id%type;
    c_name users.name%type;
     CURSOR c_users is
        SELECT id, name FROM users where SAL IS NOT NULL;
BEGIN
    OPEN c_users;
    LOOP
        FETCH c_users into c_id, c_name;
        EXIT WHEN c_users%notfound;
        dbms_output.put_line(c_id || ' ' || c_name );
    END LOOP;
    CLOSE c_users;
END;
```



```
101 Abi
103 Banu

Statement processed.
```

Result:

Thus the creation of PL/SQL programs to implement Cursors were successfully executed.

Ex.No:12

APPLICATION DEVELOPMENT USING FRONT END TOOLS AND DATABASE CONNECTIVITY

AIM

To design a forms and write a code for Employee System and make a connection withback end using database connectivity.

LEARNING OUTCOMES

After the completion of this experiment, student will be able to

Create databases using SQL
Understand all the RDBMS terms

PROBLEM STATEMENT

To create the database and perform using various commands such as

1. Create Table
2. Insert Command

SYSTEM AND SOFTWARE TOOLS REQUIRED

Software Required: MongoDB

Operating System : WINDOWS 2000 / XP / 7

Computers Required : Minimum Requirement: Pentium III or Pentium IV with 256 RAM and 40 GB hard disk

DESCRIPTION

Table Used: Employee:

NAME	FATHER _NAME	EMP. _No	DOB	SEX	MOTHER TONGUE	CITY	STREET	STATE
Sekar	Moorthy	101	2/2/80	Male	Hindi	Delhi	Clive St	Delhi
Ajith	Arjun	102	23/9/81	Male	English	Banglore	MG St	KA
Anitha	Arun	103	30/10/75	Female	Tamil	Chennai	KKnagar	TN
Kowsi	Maridass	104	20/1/87	Female	Telugu	Hydrabad	Port st	AP

Table creation:

The student database has been reated in Oracle and some rows have been insertedusing the DDL and DML command

To open visual basic:

1) Go to start → all programs→ Microsoft Visual Studio 6.0 → Microsoft Visual Basic6.0 →Click.

To open a new form:

2) While opening it will ask you New project in that click standard exe→ then open→new Form is opened. (Or)

Go to File menu →click new project→new form is opened.

To bring the toolbar:

3) Go to tools menu→click toolbar→tool bar is loaded.

To create a form:

4) From the tool bar drag the text box and label and place it in the form.
5) The number of text box and label depends upon the fields we have in the Table.
6) We can also have command buttons to perform particular action when they areclicked.
7) To view the form we should press shift +F7.

Data control:

Visual Basic provides a set of controls that allow you to display, add edit data in the database with minimal coding. When such controls are used the user need not write code, instead they allow the user to use their properties to access the database. Such controls are known as Data-aware controls. Data controls are a standard control available in the tool box.

Let us consider that we are maintaining a database named emp, which consists of fields like empno, empname, empadd, empphone. The steps to connect the data control to the emp database are:

1. Place the Data control on the form by double clicking on the icon representing theData

   control in the toolbox.

2. Place four text boxes on the form to display the value of the fields empno, empname, empadd

and empphone from the table emp into respectively.

3. Set the connection string property of the Data control to Access. The Connection string property determines the type of the database to access.
4. Set the DatabaseName property to emp. The DatabaseName property determines the name of the database to be opened.
5. Set the RecordSource property of the Data control to empinfo. The RecordSource property determines the name of the table to be accessed.
6. Make the text boxes bound to the Data control by using the DataSource and DataFieldproperties. A control is said to be data-aware when it is bound to a Data control. The DataSource property determines the name of the Data control to which the text box is to be bound. The DataField property determines the name of the field in the table. Set the Name, DataSource and DataField properties of the text boxes as shown in the below table.

Object	Property	Setting
Text1	Name	txtempno
	DataSource	data 1
	DataField	empno
ADODB1	ConnectionString	Provider=MSDAORA.1; User ID=scott;Persist SecurityInfo=false
	Password	Tiger
	RecordSource	empinfo
	UserName	scott

1. Run the application and use the arrow buttons on the data control to navigate through the records in the below screen. You have to write code to add, update, edit and deleterecords in the below screen. Or else press the function key 5 (F5)

SAMPLE CODING AND EXECUTION OF THE PROGRAM

Table Creation:

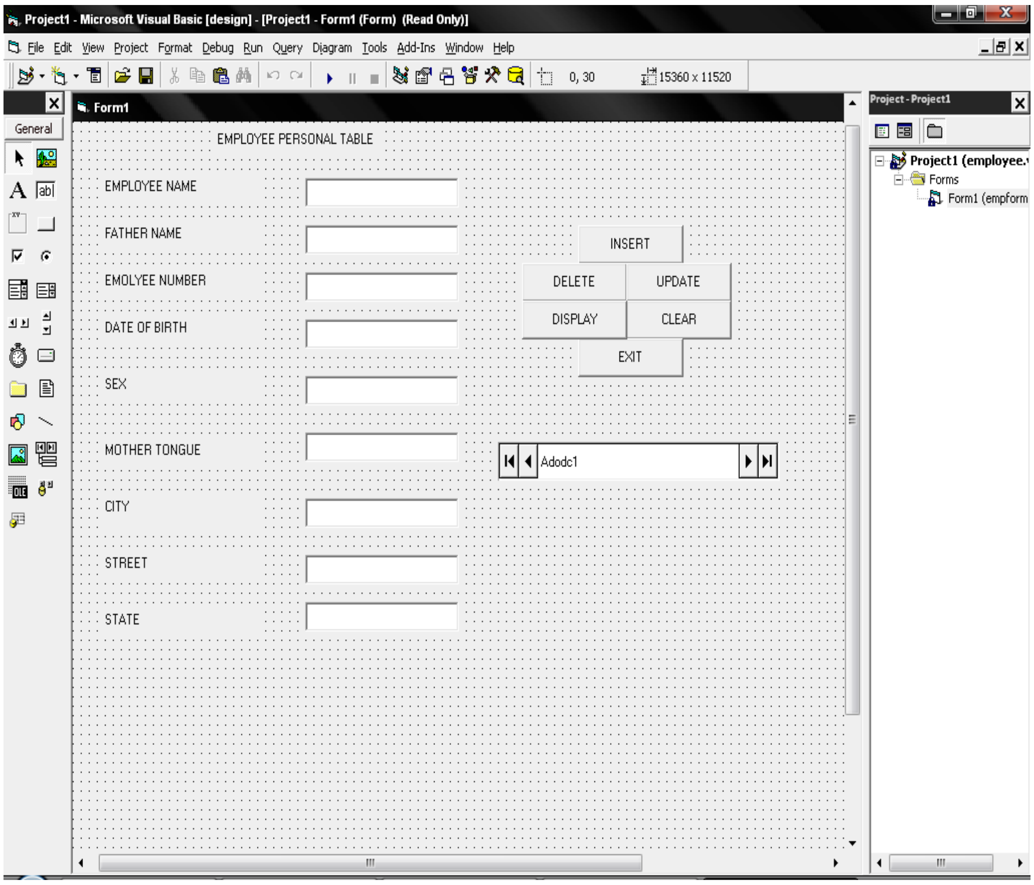
```
SQL> create table employee (name varchar2(20), f_name varchar2(15), emp_no number(5),dob
     date, sex varchar2(5) , m_tong varchar2(10) , city varchar2(10) , street varchar2(10) , state
     varchar2(10));
```

Table created.

Values Are Inserted By

```
SQL>Insert into employee values (‘&name’, ‘&f_name’, &emp_no, ‘&dob’, ‘&sex’,
     ‘&m_tong’, ‘&city’, ‘&street’, ‘&state’);
```

SAMPLE OUTPUT;

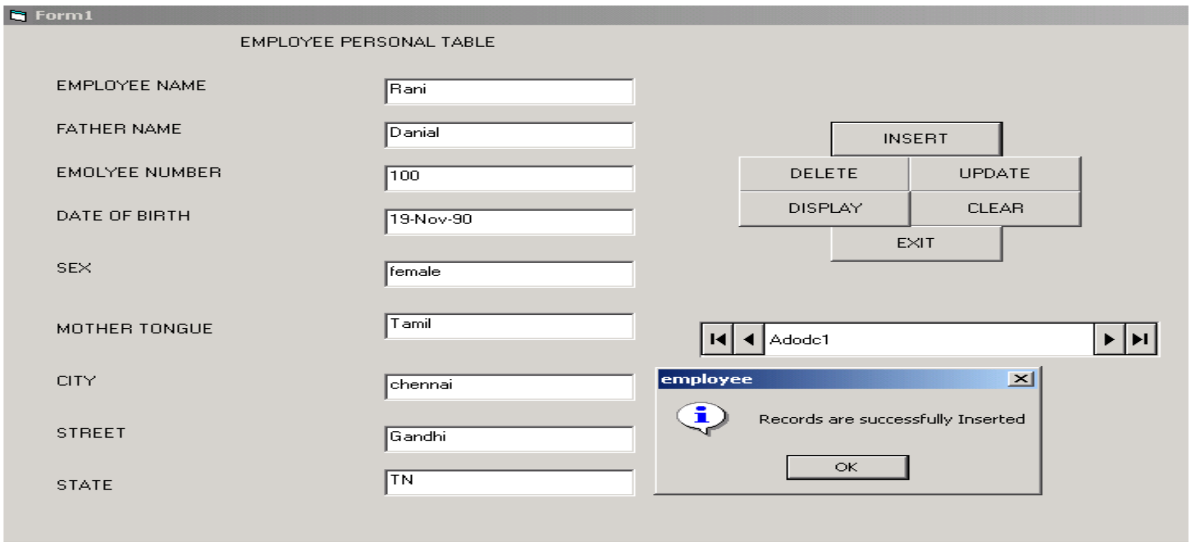


CODING WINDOW:

```
  Private Sub CLEAR_Click ()
Text1.Text    =    ""
    Text2.Text   =   ""
    Text3.Text   =   ""
    Text4.Text   =   ""
    Text5.Text   =   ""
    Text6.Text   =   ""
    Text7.Text   =   ""
    Text8.Text   =   ""
    Text9.Text = ""
  End Sub

      Private Sub DELETE_Click()
          Adodc1.Recordset.DELETE
MsgBox "Records are Deleted successfully", vbInformation
  End Sub

      Private Sub INSERT_Click()
Adodc1.Recordset.AddNew
MsgBox "Records are Inserted successfully", vbInformation
  End Sub

      Private Sub UPDATE_Click()
          Adodc1.Recordset.UPDATE
MsgBox "Records are Updated successfully", vbInformation
  End Sub

Private Sub DISPLAY_Click()
rs.Open " select * from bank where acc_no=" & Text1.Text & " ", Con, adOpenStaticIf
    rs.BOF Then
MsgBox "No Such Record Found", vbInformationElse
MsgBox "Record Found", vbInformation
Text1.Text = rs.Fields("name") Text2.Text =
rs.Fields("f_name") Text3.Text =
rs.Fields("emp_n0") Text4.Text =
rs.Fields("dob") Text5.Text = rs.Fields("sex")
Text6.Text = rs.Fields("m_tong")
    Text7.Text = rs.Fields("city")
    Text8.Text = rs.Fields("street")
    Text9.Text = rs.Fields("state")
End If
End Sub
```

<u>Screen Shots</u>:

1. **<u>Insertion</u>**



Form1

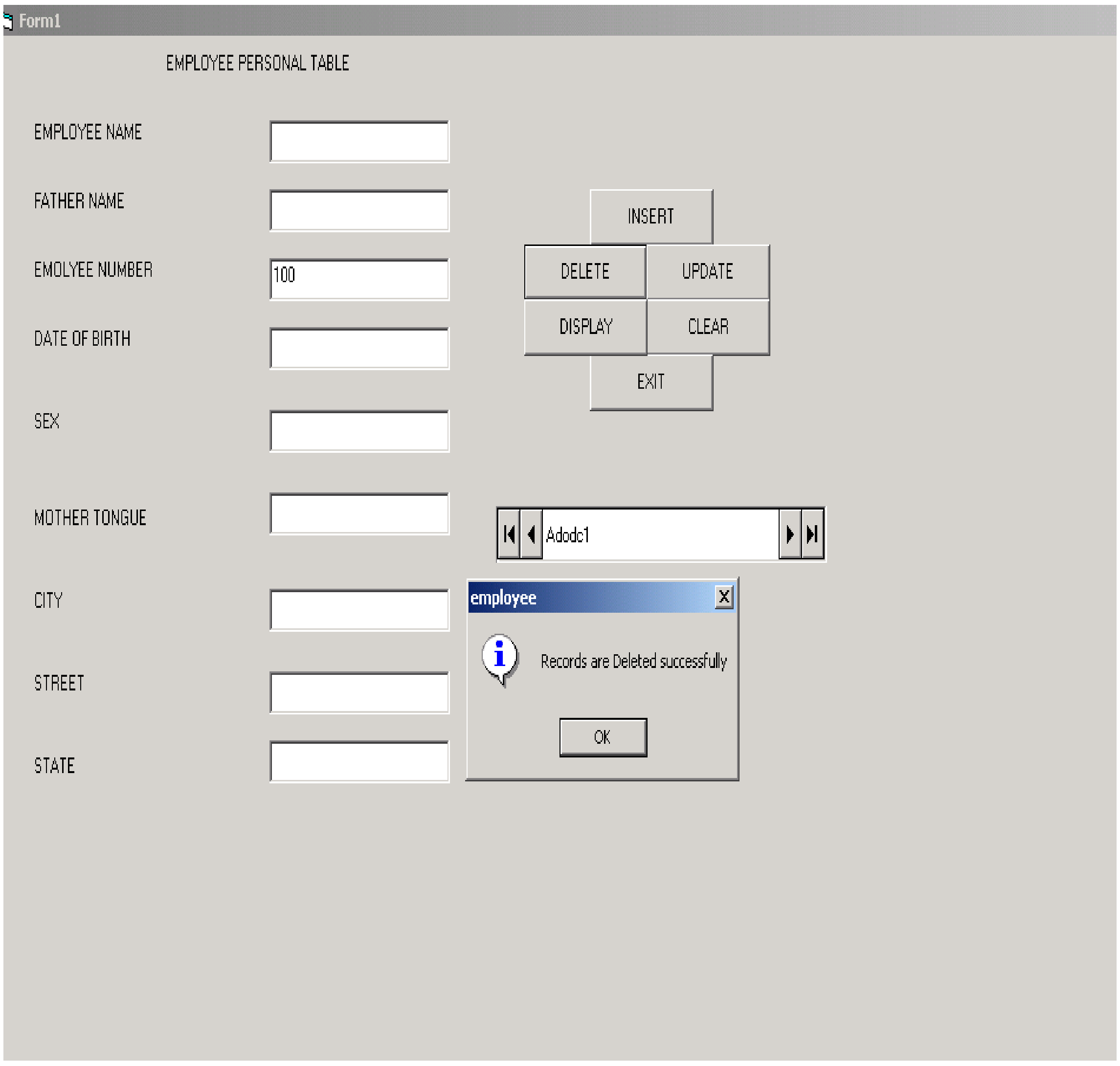
EMPLOYEE PERSONAL TABLE

EMPLOYEE NAME	Rani
FATHER NAME	Danial
EMOLYEE NUMBER	100
DATE OF BIRTH	19-Nov-90
SEX	female
MOTHER TONGUE	Tamil
CITY	chennai
STREET	Gandhi
STATE	TN

employee

Records are successfully Inserted

1. **Deleting**



Form1

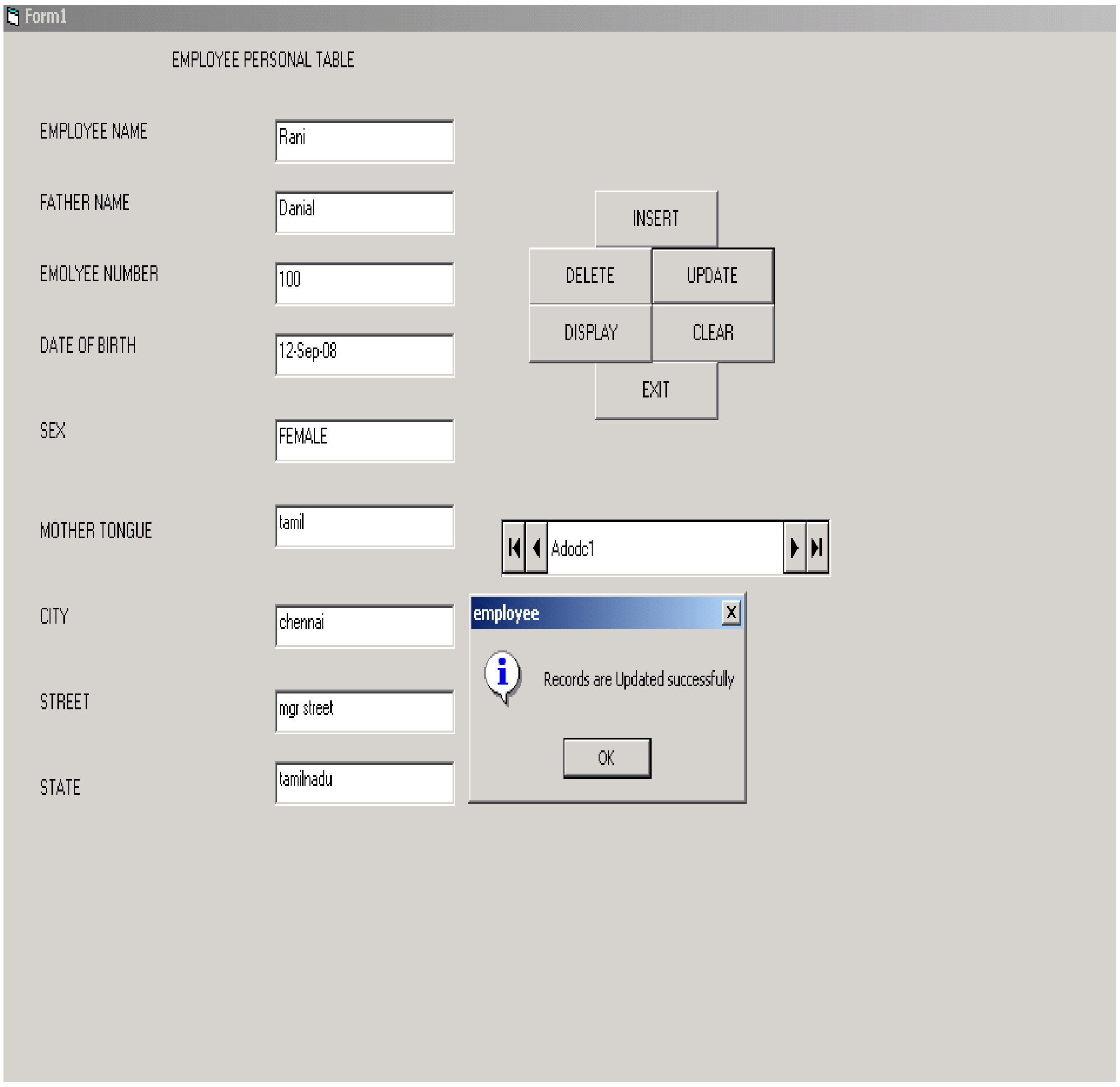
EMPLOYEE PERSONAL TABLE

EMPLOYEE NAME	
FATHER NAME	
EMOLYEE NUMBER	100
DATE OF BIRTH	
SEX	
MOTHER TONGUE	
CITY	
STREET	
STATE	

employee

Records are Deleted successfully

2. **Updating**



Form1

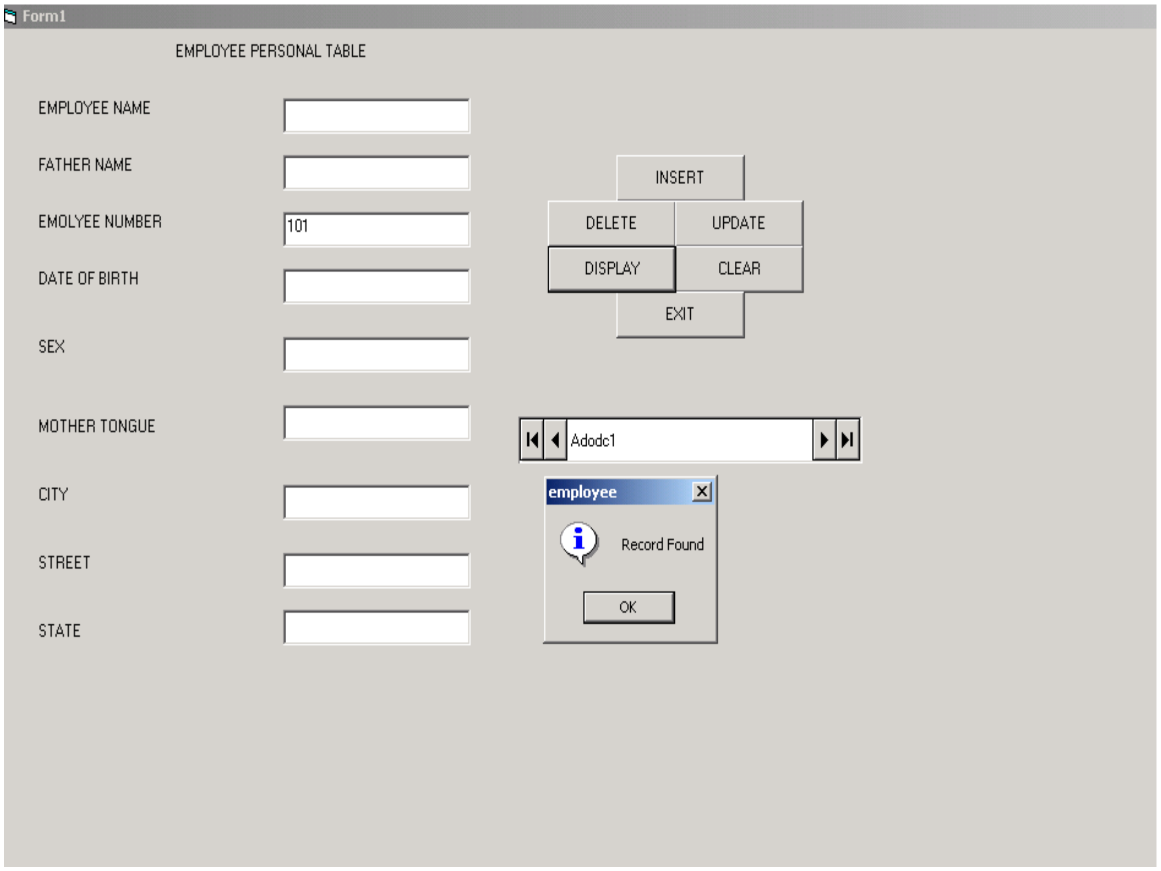
EMPLOYEE PERSONAL TABLE

EMPLOYEE NAME	Rani
FATHER NAME	Danial
EMOLYEE NUMBER	100
DATE OF BIRTH	12-Sep-08
SEX	FEMALE
MOTHER TONGUE	tamil
CITY	chennai
STREET	mgr street
STATE	tamilnadu

employee

Records are Updated successfully

3. Display a particular record


Form1

EMPLOYEE PERSONAL TABLE

EMPLOYEE NAME	
FATHER NAME	
EMOLYEE NUMBER	101
DATE OF BIRTH	
SEX	
MOTHER TONGUE	
CITY	
STREET	
STATE	

employee

Record Found

RESULT:

Thus design and implementation of simple employee application is executed.

Ex.No:13

STUDY OF MONGODB

Aim:

To study the concepts and installation process of MongoDB.

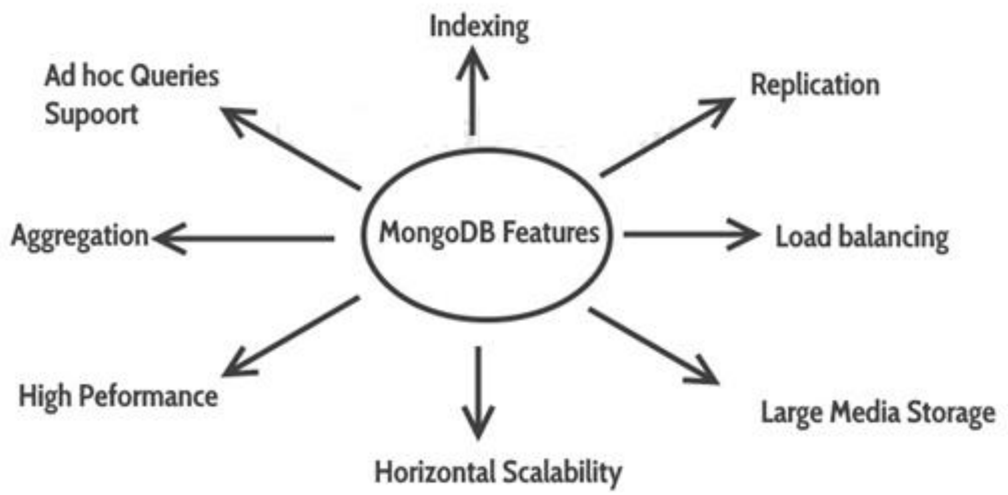
Introduction to MongoDB:

MongoDB is an open source, document oriented database that stores data in form of documents (key and value pairs).

History of MongoDB:

- MongoDB was created by Eliot and Dwight (founders of DoubleClick) in 2007, when they faced scalability issues while working with relational database. The organization that developed MongoDB was originally known as 10gen.
- In Feb 2009, they changed their business model and released MongoDB as an open source Project. The organization changed its name in 2013 and now known as MongoDB Inc.

Features of MongoDB:



1. MongoDB provides **high performance**. Most of the operations in the MongoDB are faster compared to relational databases.
2. MongoDB provides **auto replication** feature that allows you to quickly recover data in case of a failure.
3. Horizontal scaling is possible in MongoDB because of sharing. Sharing is partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved.

Horizontal scaling vs vertical scaling:

Vertical scaling means adding more resources to the existing machine while horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement, on the other hand horizontal scaling is easy to implement. Horizontal scaling database examples: MongoDB, Cassandra etc.

4. **Load balancing**: Horizontal scaling allows MongoDB to balance the load.
5. **High Availability**: Auto Replication improves the availability of MongoDB database.
6. **Indexing:** Index is a single field within the document. Indexes are used to quickly locate data without having to search every document in a MongoDB database. This improves the performance of operations performed on the MongoDB database.

Term Equivalent between RDBMS & MongoDB:

RDBMS	**Mongo DB**
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
MySQL/Oracle	MongoDB
MySQL/SQLplus	MongoDB

Installing MongoDB On Windows:

Step 1 : To install the MongoDB on windows, first download the latest release of MongoDB from http://www.mongodb.org/downloads Make sure you get correct version of MongoDB depending upon your windows version. To get your windows version open command prompt and execute following command:

```
        C:\>wmic os get os architecture
        OSArchitecture
        64-bit
C:\>
```

32-bit versions of MongoDB only support databases smaller than 2GB and suitable only for testing and evaluation purposes.

Step 2: Now extract your downloaded file to c:\ drive or any other location. Make sure name of the extracted folder is
mongodb-win32-i386-[version] or mongodb-win32-x86_64-[version]. Here [version] is the version of MongoDB
download.

Step 3 : Now open command prompt and run the following command

```
C:\>move mongodb-win64-* mongodb
1 dir(s) moved.
C:\>
```

In case you have extracted the mondodb at different location, then go to that path by using command cd
FOLDER/DIR and now run the above given process.
MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So
you need to create this folder using the Command Prompt. Execute the following command sequence

```
C:\>md data
C:\md data\db
```

If you have install the MongoDB at different location, then you need to specify any alternate path for \data\db by setting the path dbpath in mongod.exe.

Step 4 : In command prompt navigate to the bin directory present into the mongodb installation folder. Suppose my installation
folder is D:\set up\mongodb

```
C:\Users\XYZ>d:
D:\>cd "set up"
D:\set up>cd mongodb
D:\set up\mongodb>cd bin
D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
```

This will show waiting for connections message on the console output indicates that the mongod.exe process is
running successfully.
Now to run the mongodb you need to open another command prompt and issue the following command

```
D:\set up\mongodb\bin>mongo.exe
MongoDB shell version: 2.4.6
```

```
connecting to: test
>db.test.save( { a: 1 } )
>db.test.find()
{ "_id" : ObjectId(5879b0f65a56a454), "a" : 1 }
>
```

This will show that mongodb is installed and run successfully. Next time when you run mongodb you need to issue
only commands

```
D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
```

D:\set up\mongodb\bin>mongo.exe.

MongoDB Create Database :

The use Command
MongoDB use DATABASE_NAME is used to create database. The command will create a new database, if it
doesn't exist otherwise it will return the existing database.

Syntax:
Basic syntax of use DATABASE statement is as follows:
use DATABASE_NAME

Example:
If you want to create a database with name <mydb>, then use DATABASE statement would be as follows:

```
>use mydb
switched to db mydb
```

To check your currently selected database use the command db

```
>db
mydb
```

If you want to check your databases list, then use the command show dbs.

```
>show dbs
local 0.78125GB
test 0.23012GB
```

Your created database (mydb) is not present in list. To display database you need to insert at least one document into
it.

```
>db.movie.insert({"name":"database_IT"})
>show dbs
local 0.78125GB
mydb 0.23012GB
test 0.23012GB
```

In MongoDB default database is test. If you didn't create any database then collections will be stored in test database.

<u>Result:</u>

Thus the study of MongoDB has been done successfully.

EX.NO: 14

DOCUMENT DATABASE CREATION USING MONGODB

Aim:

1. To create document in MongoDB

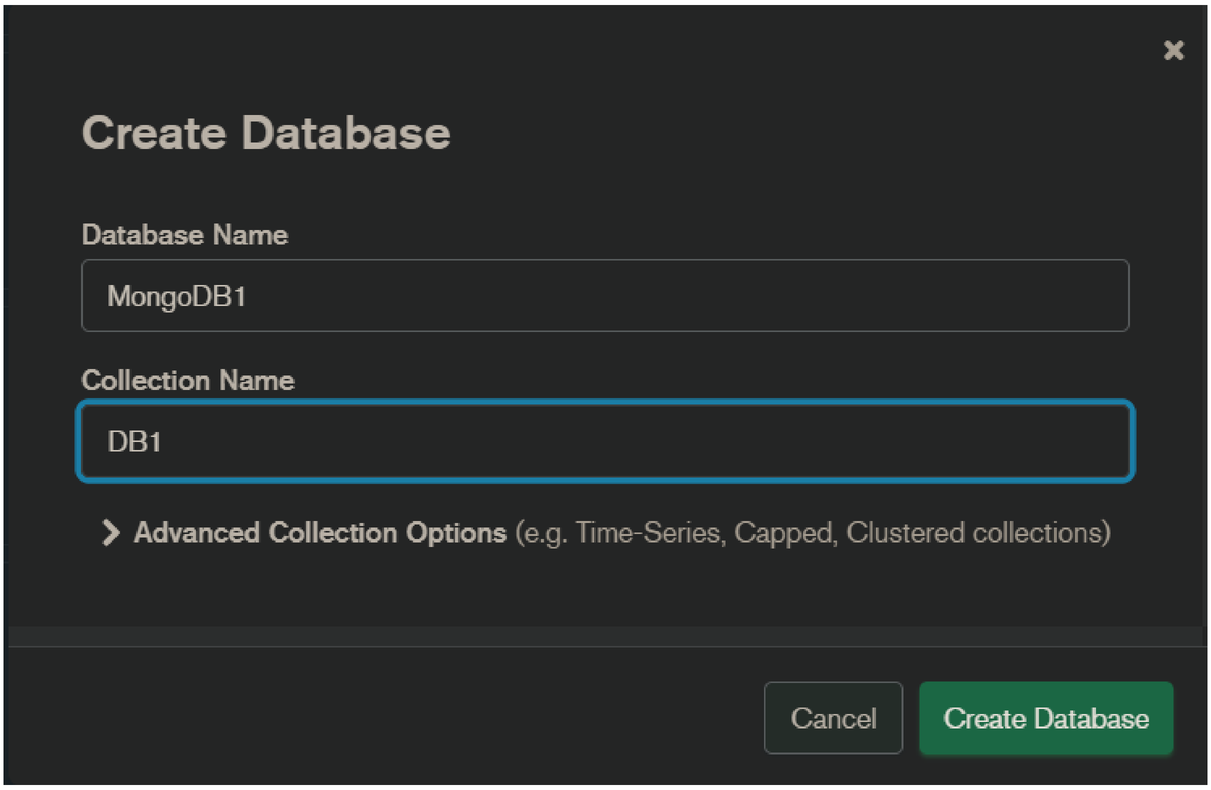
Learning Outcomes:

After the completion of this experiment, student will be able to

- Understand the use of MongoDB
- How to create and use Documents in MongoDB

Problem Statement:

1. Create Database:
   - Open MongoDB Compass and establish the connection and create a new Database

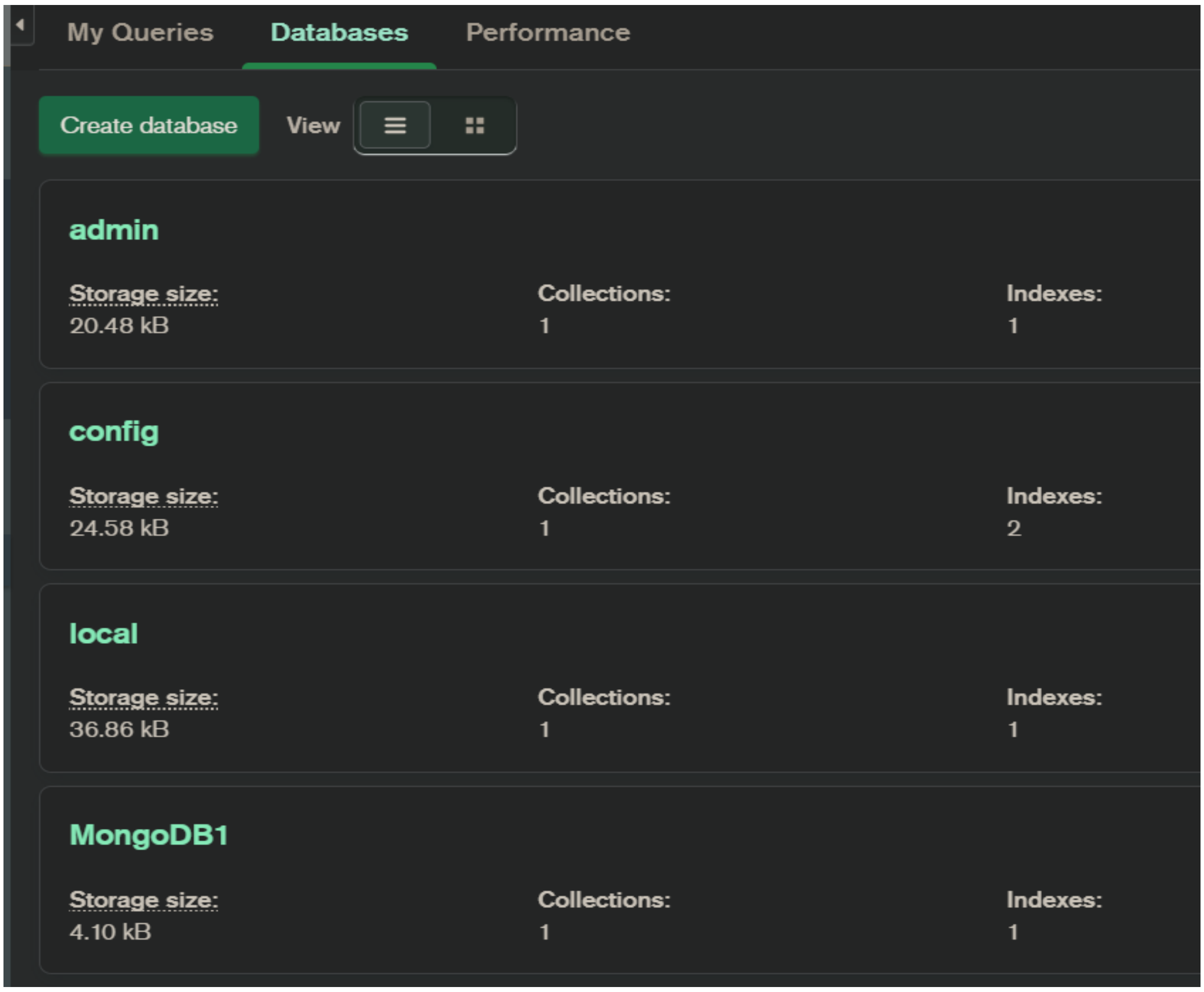


Create Database

Database Name: MongoDB1

Collection Name: DB1

Advanced Collection Options (e.g. Time-Series, Capped, Clustered collections)
Cancel
Create Database



My Queries   Databases   Performance

Create database   View

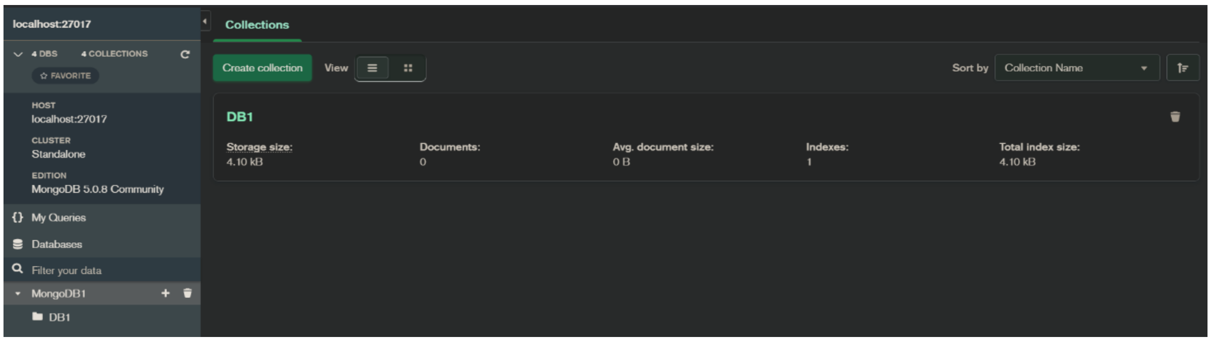
admin
Storage size: 20.48 kB   Collections: 1   Indexes: 1

config
Storage size: 24.58 kB   Collections: 1   Indexes: 2

local
Storage size: 36.86 kB   Collections: 1   Indexes: 1

MongoDB1
Storage size: 4.10 kB   Collections: 1   Indexes: 1

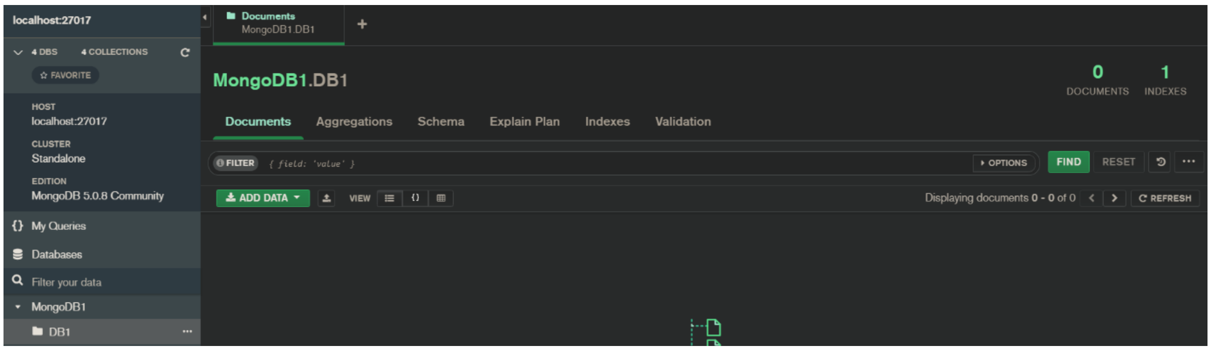
2. Create Collection:



localhost:27017
4 DBS   4 COLLECTIONS
FAVORITE
HOST localhost:27017
CLUSTER Standalone
EDITION MongoDB 5.0.8 Community
My Queries
Databases
Filter your data
MongoDB1
DB1

Collections
Create collection   View   Sort by Collection Name

DB1
Storage size: 4.10 kB   Documents: 0   Avg. document size: 0 B   Indexes: 1   Total index size: 4.10 kB



localhost:27017
4 DBS   4 COLLECTIONS
FAVORITE
HOST localhost:27017
CLUSTER Standalone
EDITION MongoDB 5.0.8 Community
My Queries
Databases
Filter your data
MongoDB1
DB1

Documents MongoDB1.DB1

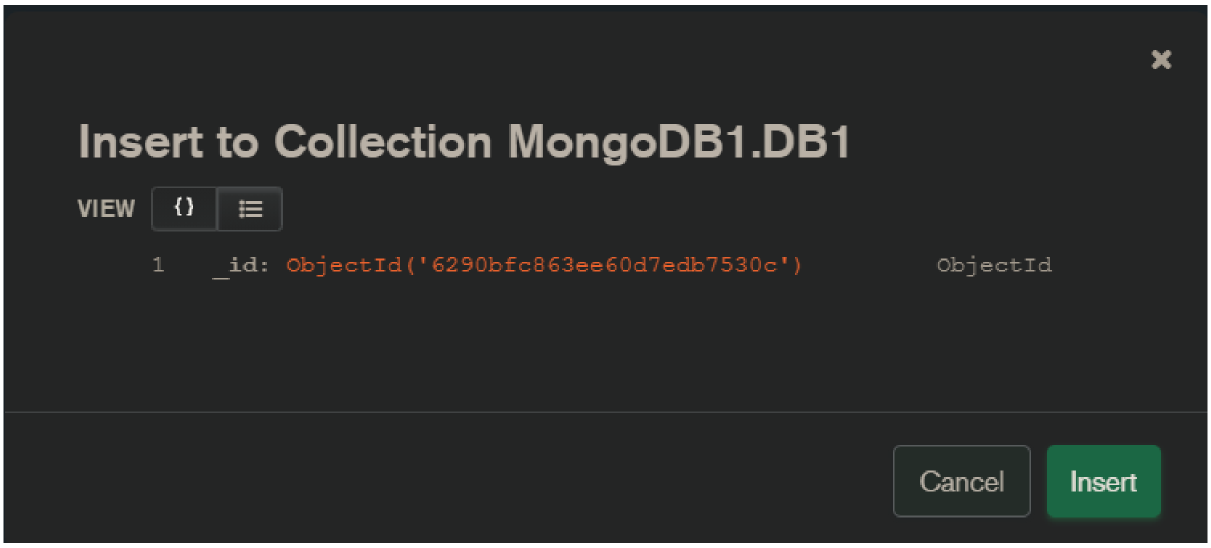
MongoDB1.DB1   0 DOCUMENTS   1 INDEXES

Documents   Aggregations   Schema   Explain Plan   Indexes   Validation

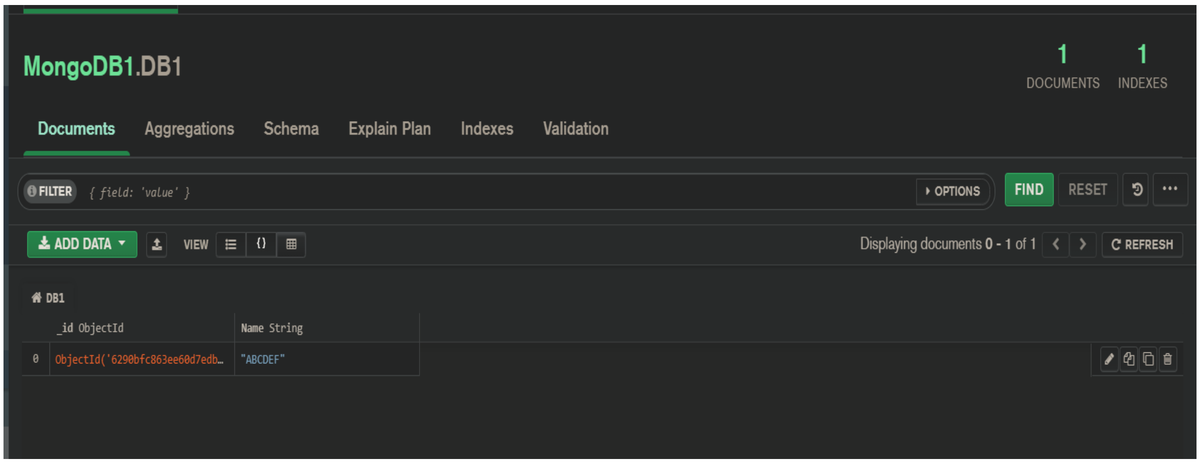
FILTER { field: 'value' }   OPTIONS   FIND   RESET

ADD DATA   VIEW   Displaying documents 0 - 0 of 0   REFRESH

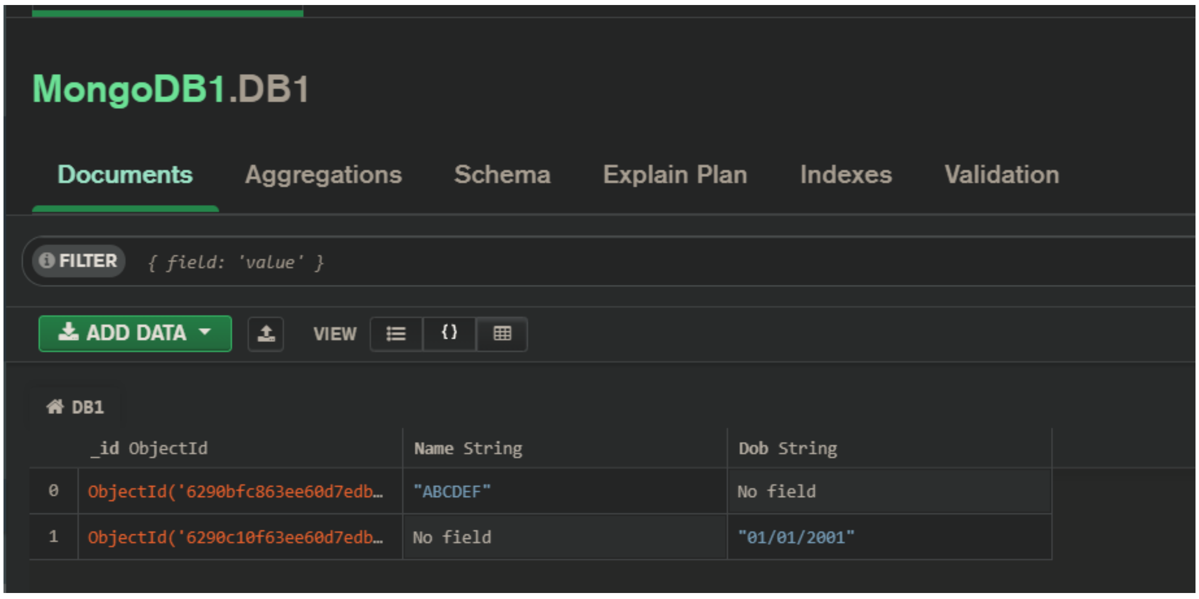
3. Add Documents:



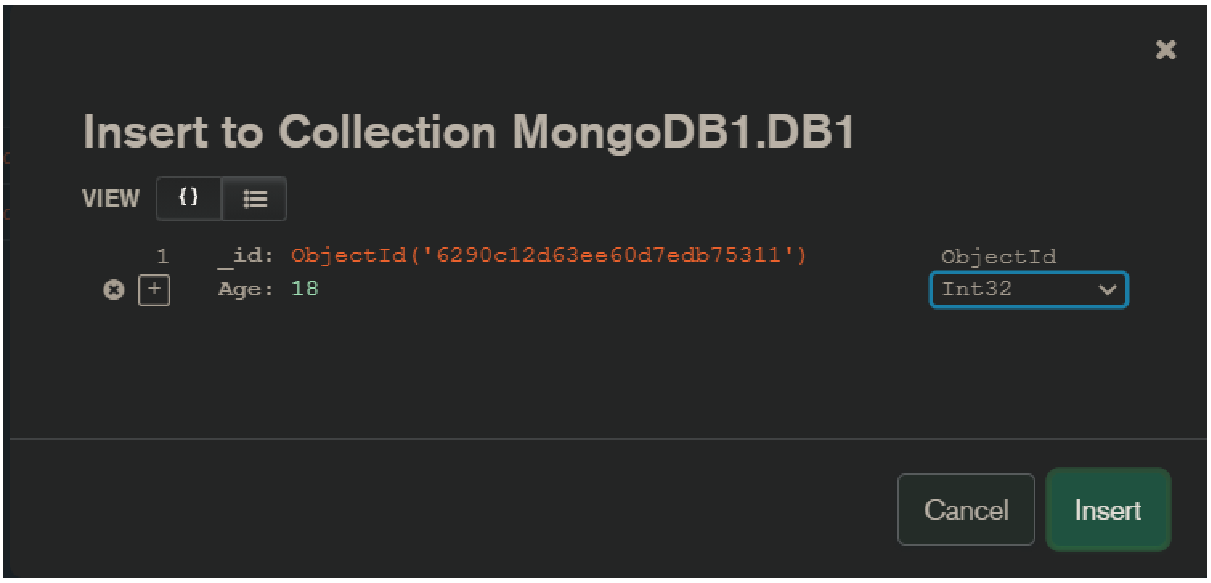
```
1    _id: ObjectId('6290bfc863ee60d7edb7530c')          ObjectId
```



	_id ObjectId	Name String
0	ObjectId('6290bfc863ee60d7edb…	"ABCDEF"

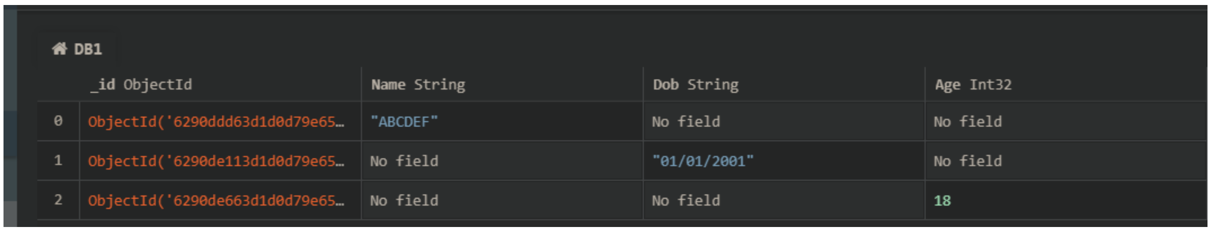


	_id ObjectId	Name String	Dob String
0	ObjectId('6290bfc863ee60d7edb…	"ABCDEF"	No field
1	ObjectId('6290c10f63ee60d7edb…	No field	"01/01/2001"



Insert to Collection MongoDB1.DB1

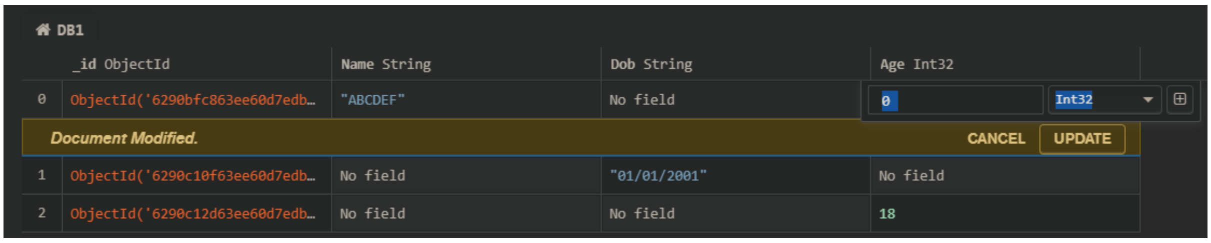
```
_id: ObjectId('6290c12d63ee60d7edb75311')    ObjectId
Age: 18                                      Int32
```



DB1

	_id ObjectId	Name String	Dob String	Age Int32
0	ObjectId('6290ddd63d1d0d79e65…	"ABCDEF"	No field	No field
1	ObjectId('6290de113d1d0d79e65…	No field	"01/01/2001"	No field
2	ObjectId('6290de663d1d0d79e65…	No field	No field	18

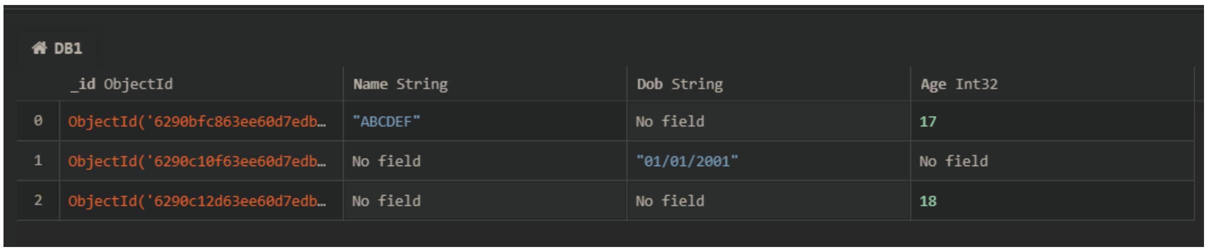
4. Update values:



DB1

	_id ObjectId	Name String	Dob String	Age Int32
0	ObjectId('6290bfc863ee60d7edb…	"ABCDEF"	No field	0 Int32
1	ObjectId('6290c10f63ee60d7edb…	No field	"01/01/2001"	No field
2	ObjectId('6290c12d63ee60d7edb…	No field	No field	18

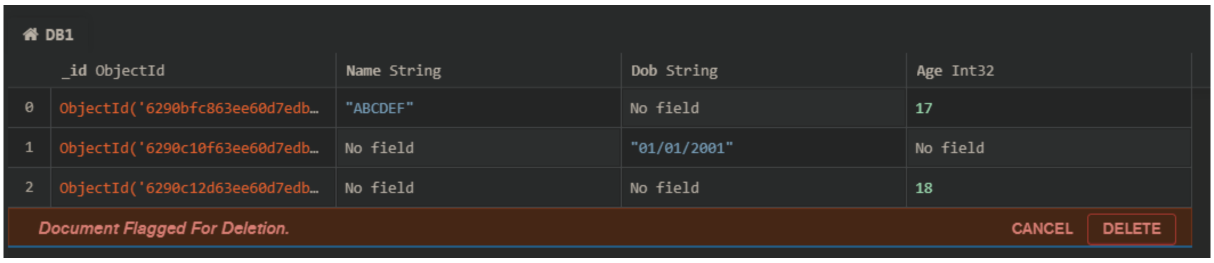
Document Modified. CANCEL UPDATE



DB1

	_id ObjectId	Name String	Dob String	Age Int32
0	ObjectId('6290bfc863ee60d7edb…	"ABCDEF"	No field	17
1	ObjectId('6290c10f63ee60d7edb…	No field	"01/01/2001"	No field
2	ObjectId('6290c12d63ee60d7edb…	No field	No field	18

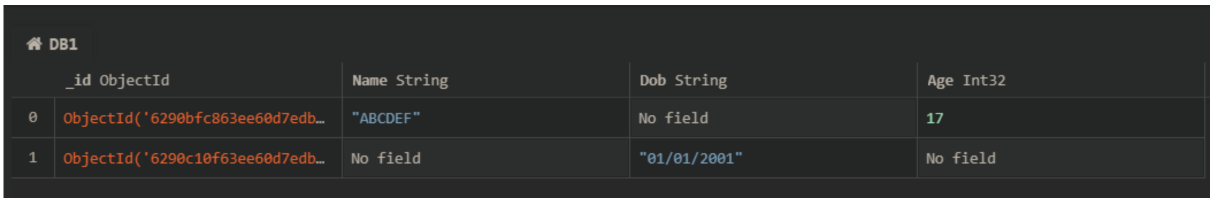
5. Deleting Document:



DB1

	_id ObjectId	Name String	Dob String	Age Int32
0	ObjectId('6290bfc863ee60d7edb…	"ABCDEF"	No field	17
1	ObjectId('6290c10f63ee60d7edb…	No field	"01/01/2001"	No field
2	ObjectId('6290c12d63ee60d7edb…	No field	No field	18

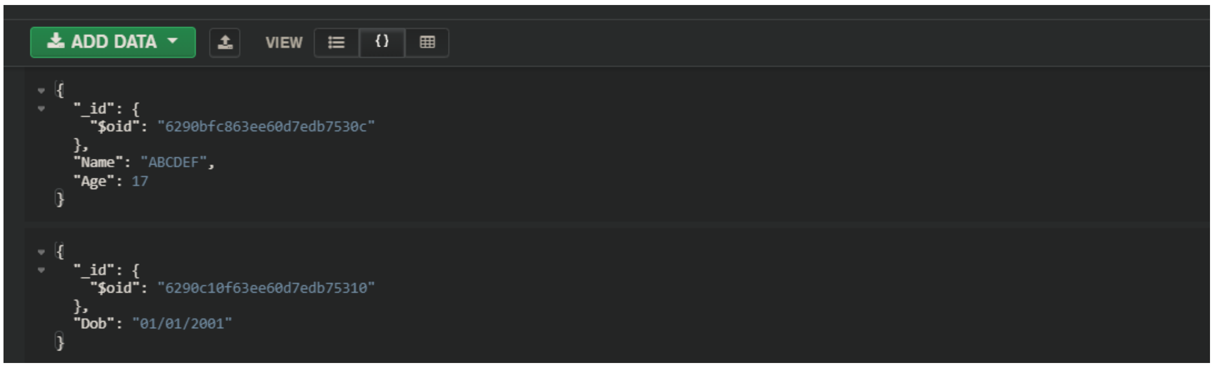
Document Flagged For Deletion. CANCEL DELETE



DB1

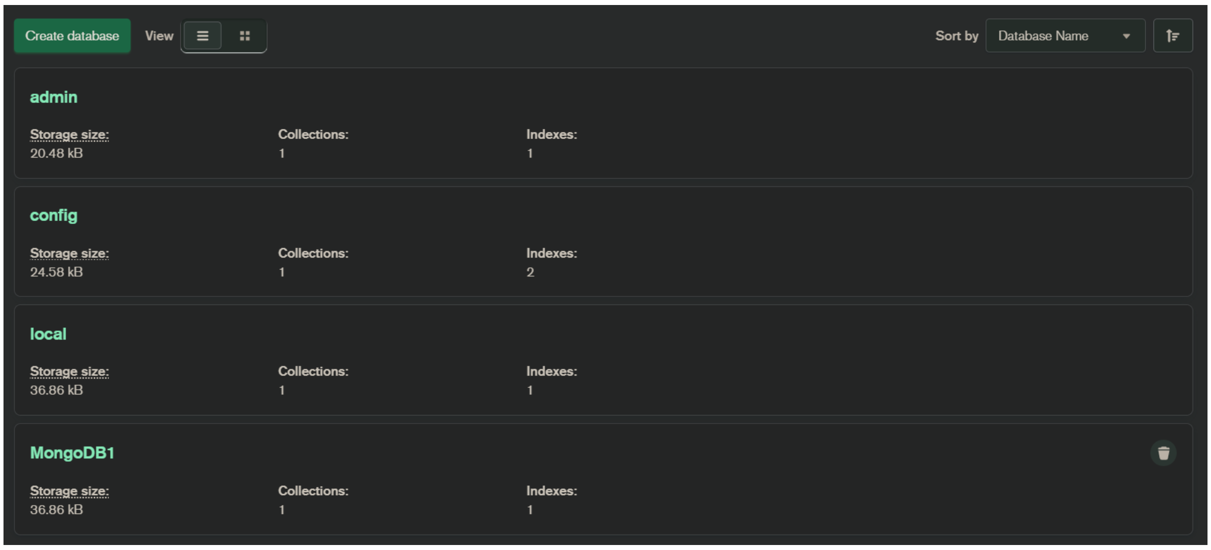
	_id ObjectId	Name String	Dob String	Age Int32
0	ObjectId('6290bfc863ee60d7edb…	"ABCDEF"	No field	17
1	ObjectId('6290c10f63ee60d7edb…	No field	"01/01/2001"	No field

->Query Representation:



```
{
  "_id": {
    "$oid": "6290bfc863ee60d7edb7530c"
  },
  "Name": "ABCDEF",
  "Age": 17
}

{
  "_id": {
    "$oid": "6290c10f63ee60d7edb75310"
  },
  "Dob": "01/01/2001"
}
```

6. Droping Database:

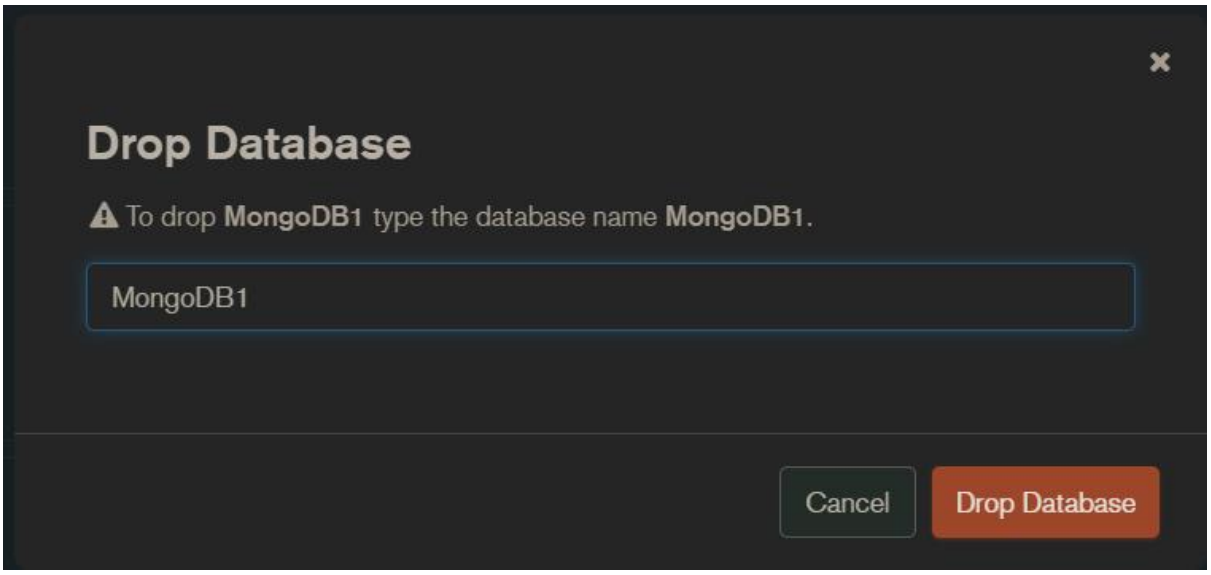


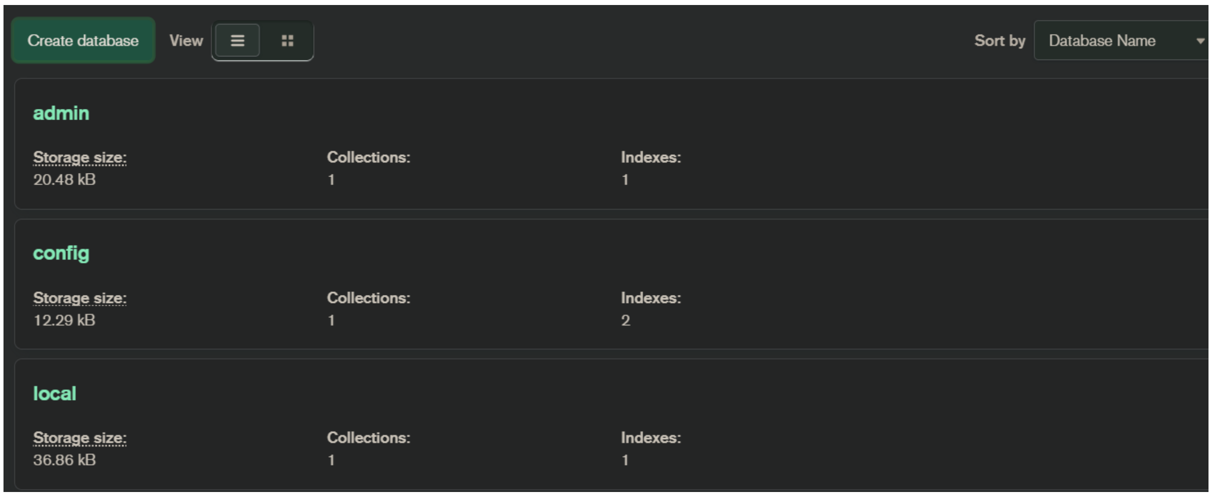
admin
Storage size: 20.48 kB | Collections: 1 | Indexes: 1

config
Storage size: 24.58 kB | Collections: 1 | Indexes: 2

local
Storage size: 36.86 kB | Collections: 1 | Indexes: 1

MongoDB1
Storage size: 36.86 kB | Collections: 1 | Indexes: 1





admin		
Storage size: 20.48 kB	Collections: 1	Indexes: 1
config		
Storage size: 12.29 kB	Collections: 1	Indexes: 2
local		
Storage size: 36.86 kB	Collections: 1	Indexes: 1

Result:

Thus, MongoDB operations are executed successfully.

Ex.No: 15

STUDY OF CLOUD STORAGES

AIM:

To study the cloud storages.

LEARNING OUTCOMES:

After the completion of this experiment, student will be able to:

- Know the cloud storages
- Understand about the storages in cloud
- Able to know the issues in cloud

DESCRIPTION

I. INTRODUCTION

Cloud computing transforms the way in which current enterprises IT infrastructure is constituted and managed through consumable services such as infrastructure, platform, and applications. It will convert the IT infrastructure from a “factory” into a “supply chain” model. It is a type of computing that provides simple, on demand access to pools of highly elastic computing resources. These resources are provided as a service over a network, often the Internet. Cloud enables the consumers of the technology to think of computing as effectively limitless, of minimal cost, and reliable, as well as not to be concerned about how it is constructed, how it works, who operates it, or where it is located. The cloud computing model is enabled by the ongoing standardization of underlying technologies like virtualization, Service Oriented Architecture (SOA), and Web 2.0.

Cloud computing is a style of computing where computing resources are easy to obtain and access, simple to use, cheap, and just work. Cloud is not a point product or a singular technology, but a way to deliver IT resources in a manner that provides self-service, on-demand and pay-per-use consumption. Utilizing cloud delivers time and cost savings. Cloud involves the subscriber and the provider. The service provider can be a company’s internal IT group, a trusted third party or a combination of both. The subscriber is anyone who uses the services. By making data available in the cloud, it can be more easily and ubiquitously accessed, often at much lower cost, increasing its value by enabling opportunities for enhanced collaboration, integration, and
analysis on a shared common platform.

II. KEY TECHNOLOGIES

2. 1Types of Cloud

In a cloud computing system, there's a significant workload shift. Local computers no longer have to do all the heavy lifting when it comes to running applications. The network of computers that make up the cloud handles them instead which leads in reduction of hardware and software demands on the user's side. A typical cloud computing architecture is given in Fig.1. The only thing the user's computer needs to be able to run is the cloud computing systems interface software, which can be as simple as a Web browser, and the cloud's network takes care of the rest. There are three common types of clouds available namely, Private, Public and Hybrid cloud which is represented in Fig. 2. A private cloud is based upon a pool of shared resources, whose access is limited within organizational boundaries. The resources are accessed over a private and secured intranet, and are all owned and controlled by the company's IT organization. In essence, the cloud computing business model is brought and managed in-house to enable shared IT services. A public cloud is a domain where the public Internet is used to obtain cloud services. The resources that make up those services are owned by the respective cloud service. Some examples include Salesforce.com, Google App Engine and Google search, Microsoft Azure, and Amazon Web services such as EC2

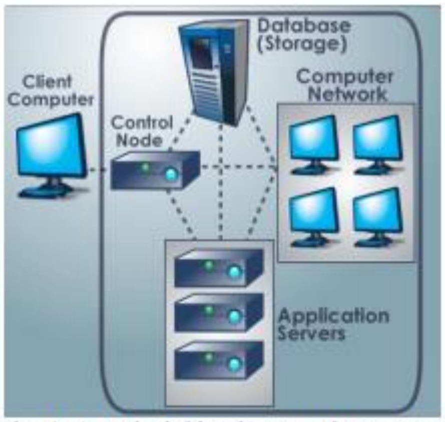


Fig. 1 A typical Cloud computing system

A Hybrid cloud is a combination of private and public clouds, where services from each domain are consumed in an integrated fashion and include an extended relationship with the selected external service providers.

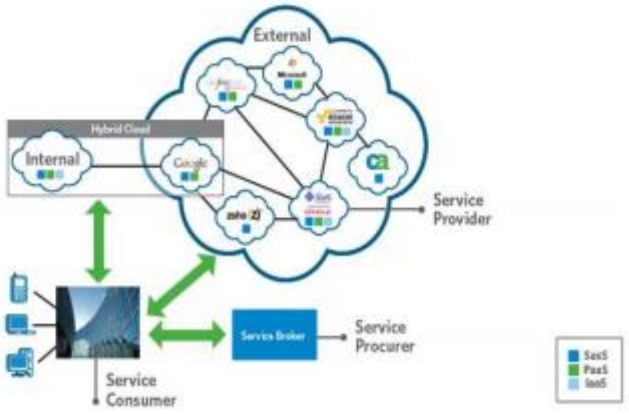


Fig. 2 Three types of Cloud computing model

2.2 Cloud Computing Service Models

Private and Public clouds serve as the backbone for a variety of different cloud computing service models given in Fig.3. Currently the industry has been successfully adopting three common types of cloud computing service models.

Infrastructure as a Service (IaaS), is a service model around servers (compute power), storage capacity, and network bandwidth. Examples include Amazon EC2 and S3, Rackspace, AT&T, and Verizon.

Platform-as-a-Service (PaaS) provides an externally managed platform for building and deploying applications and services. This model typically provides development tools such as databases and development studios for working with the supplied frameworks, as well as the infrastructure to host the built application. Examples include Force.com, Microsoft Azure, and Google App Engine.

Software-as-a-Service (SaaS) is simply having a software system running on a computer that doesn't belong to the customer and isn't on the customer's premises. It is based on the concept of renting an application from a service provider rather than buying, installing and running software yourself.

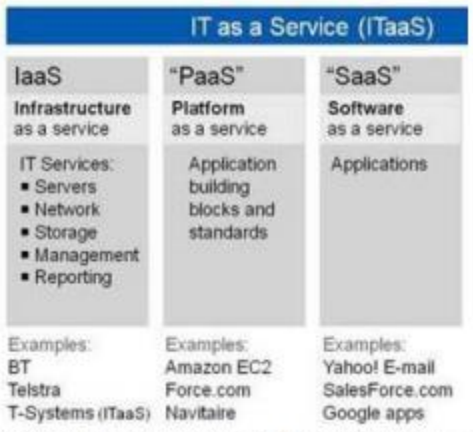


Fig. 3 Three types of Cloud service models

2.3 Key benefits of Cloud Computing

Management Insight, NH, USA, which is a dedicated market research consulting firm, conducted a study on the impact of Cloud services in the market. This study was sponsored by CA Technologies, New York, USA. The statistical data (given in Fig.4 & 5) has revealed the following facts.

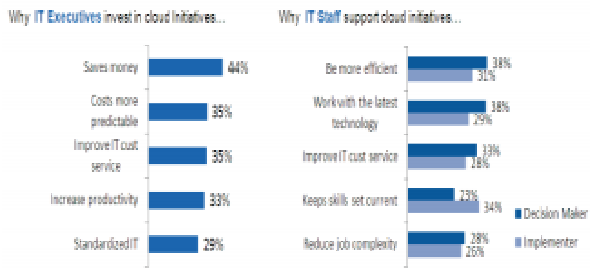


Fig. 4 IT personnel attitude towards the Cloud

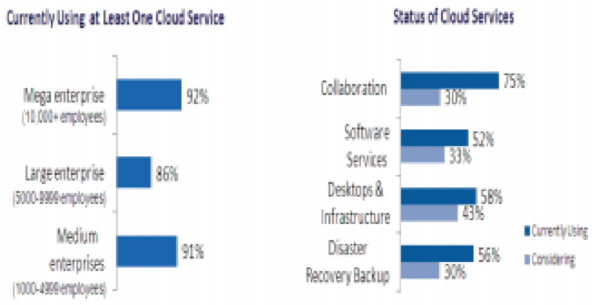


Fig. 5 Usage of Cloud services in the market

Cloud computing offers the following advantages to the enterprises:

i) Lower costs: All resources, including expensive networking equipment, servers, IT personnel, etc. are shared, resulting in reduced costs, especially for small to mid-sized applications.

ii) Shifting Capital Expenses to Operational Expenses: Cloud computing enables companies to shift money from capital expenses to operating expenses, which ultimately allows the enterprise to focus their money and resources on innovation.

iii) Agility: Provisioning on-demand enables faster setup on an as-needed basis. When a project is funded, customer can initiate service, and then if the project is over, they can simply terminate the cloud contract.

iv) Scalability: Many cloud services can smoothly and efficiently scale to handle the growing nature of the business with a more cost effective pay-as-you-go model. This is also known as elasticity.

v) Simplified maintenance: Patches and upgrades are rapidly deployed across the shared infrastructure, as well as the backups.

vi) Diverse platform support: Many cloud computing services offer built-in support for a rich collection of client platforms including browsers, mobile, and more. This diverse platform support enables applications to reach a broader category of users.

vii) Faster development: Cloud computing platforms provide many of the core services that, under traditional development models, would normally be built in house. These services, plus templates and other tools can significantly accelerate the development cycle.

viii) Large scale prototyping / Testing: Cloud computing makes large scale prototyping and load testing much easier. A client can easily spawn 1,000 servers in the cloud to load test your application and then release them as soon as they are done, and then try doing that with owned or corporate servers.

III. CLOUD STORAGE

Rapid data growth and the need to keep it safer and longer will require organizations to integrate how they manage and use their data, from creation to end of life. Now there is an opportunity to store all our data in the internet. Those off-site storages are provided and maintained by the third parties through the Internet which is represented in Fig. 6. Cloud storage offers a large pool of storage was available for use, with three significant attributes: access via Web services APIs on a non persistent network connection, immediate availability of very large quantities of storage, and pay for what you use. It supports rapid scalability.

3.1 Evolution of Cloud Storage

Cloud storage is an offering of cloud computing. Fig. 7 shows the evolution of Cloud Storage based on traditional network storage and hosted storage. Benefit of cloud storage is the access of your data from anywhere. Cloud storage providers provide storage varying from small amount of data to even the entire warehouse of an organization. Subscriber can pay to the cloud storage provider for what they are using and how much they are transferring to the cloud storage.

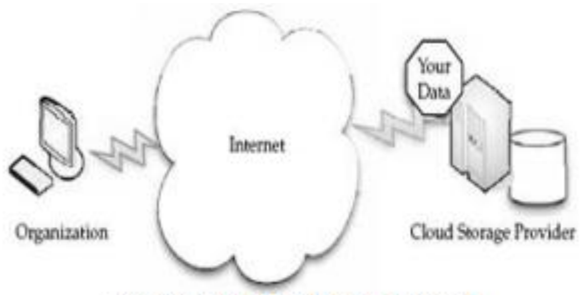


Fig. 6 Simple cloud storage model

Basically the cloud storage subscriber copies the data into any one of the data server of the cloud
storage provider. That copy of data will be made available to all the other data servers of the cloud storage provider featuring redundancy in the availability which ensures that the data of the subscriber is safe even anything goes wrong. Most systems store the same data on servers that use different power supplies.

3.2 Benefits of Cloud storage:

- No need to invest any capital on storage devices.
- No need for technical expert to maintain the storage, backup, replication and importantly disaster management.
- Allowing others to access your data will result with collaborative working style instead of individual work.

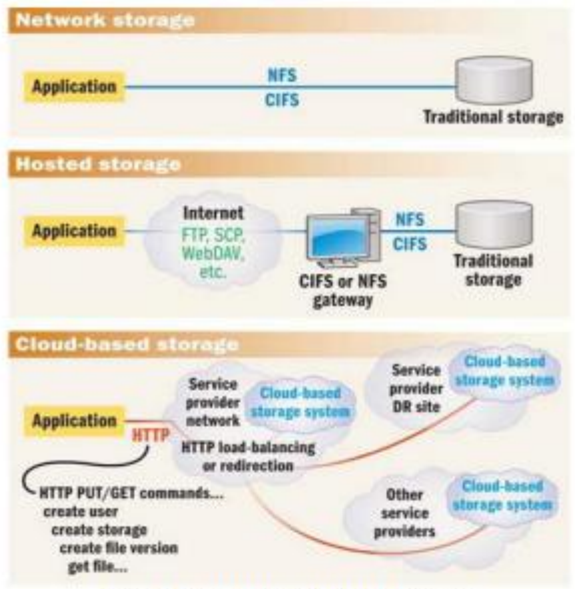
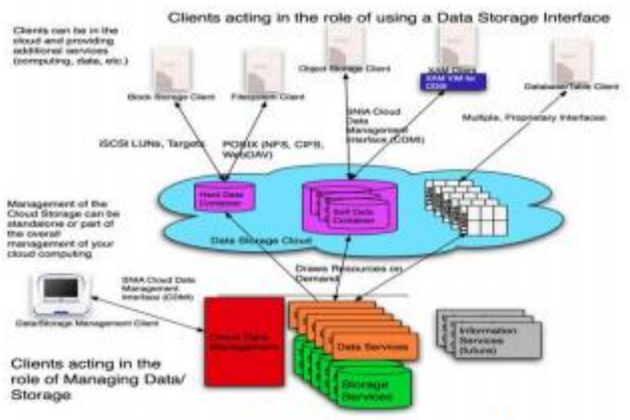


Fig. 7 Evolution of Cloud Storage

3.3 Cloud Storage Reference Model

The appeal of cloud storage is due to some of the same attributes that define other cloud services: pay as you go, the illusion of infinite capacity (elasticity), and the simplicity of use/management. It is therefore important that any interface for cloud storage support these attributes, while allowing for a multitude of business cases and offerings, long into the future. The model created and published by the Storage Networking Industry Association (SNIA) [7,8] shows multiple types of cloud data storage interfaces are able to support both legacy and new applications. All of the interfaces allow storage to be provided on demand, drawn from a pool of resources. The capacity is drawn from a pool of storage capacity provided by storage services. The data services are applied to individual data elements as determined by the data system metadata. Metadata specifies the data requirements on the basis of individual data elements or on groups of data elements (containers).

As shown in Fig. 8, Cloud Data Management Interface (CDMI) is the functional interface that applications will use to create, retrieve, update and delete data elements from the cloud. As part of this interface the client will be able to discover the capabilities of the cloud storage offering and use this interface to manage containers and the data that is placed in them. In addition, metadata can be set on containers and their contained data elements through this interface. It is expected that the interface will be able to be implemented by the majority of existing cloud storage offerings today. This can be done with an adapter to their existing proprietary interface, or by implementing the interface directly. In addition, existing client libraries such as eXtensible Access Method (XAM) can be adapted to this interface as show in Fig. 8.


Clients acting in the role of using a Data Storage Interface
Clients can be in the cloud and providing additional services (computing, data, etc.)
Block Storage Client
Filesystem Client
Object Storage Client
XAM Client
XAM VIM for CDMI
Database/Table Client
iSCSI LUNs, Targets
POSIX (NFS, CIFS, WebDAV)
SNIA Cloud Data Management Interface (CDMI)
Multiple, Proprietary Interfaces
Hard Data Container
Soft Data Container
Data Storage Cloud
Management of the Cloud Storage can be standalone or part of the overall management of your cloud computing
Draws Resources on Demand
SNIA Cloud Data Management Interface (CDMI)
Data/Storage Management Client
Cloud Data Management
Data Services
Storage Services
Information Services (future)
Clients acting in the role of Managing Data/ Storage
Fig. 8 Cloud Storage Reference model

This interface is also used by administrative and management applications to manage containers,
accounts, security access, monitoring/billing information and even for storage that is accessible by other protocols. The capabilities of the underlying storage and data services are exposed so that clients can understand the offering. Conformant cloud offerings may offer a subset of either interface as long as they expose the limitations in the capabilities part of the interface.

<u>3.4 Cloud Storage API</u>

A Cloud Storage Application Programming Interface (API) is a method for access to and utilization of a cloud storage system. The most common of these kinds are REST (Representational State Transfer) although there are others, which are based on SOAP (Simple Object Access Protocol). All these APIs are associated with establishing requests for service via the Internet. REST is a concept widely recognized as an approach to "quality" scalable API design.
One of the most important features of REST is that it is a "stateless" architecture. This means that everything needed to complete the request to the storage cloud is contained in the request, so that a session between the requestor and the storage cloud is not required. It is very important because the Internet is highly latent (it has an unpredictable response time and it is generally not fast when compared to a local area network).

REST is an approach that has very high affinity to the way the Internet works. Traditional file storage access methods that use NFS (network files system) or CIFS (Common Internet File System) do not work over the Internet, because of latency.

Cloud Storage is for files, which, some refer to as objects, and others call unstructured data. Think about the files stored on your PC, like pictures, spreadsheets and documents. These have an extraordinary variability, thus unstructured. The other kind of data is block or structured data. Think data base data, data that feeds transactional system that require a certain guaranteed or low-latency performance. Cloud Storage is not for this use case. Industrial Design Centre (IDC) estimates that approximately 70% of the machine stored data in the world is unstructured, and this is also the fastest growing data type. So, Cloud Storage is storage for files that is easily accessed via the Internet. This does not mean you cannot access Cloud Storage on a private network or LAN, which may also provide access to a storage cloud by other approaches, like NFS or CIFS. It does mean that the primary and preferred access is by a REST API.

REST APIs are language neutral and therefore can be leveraged very easily by developers using any development language they choose. Resources within the system may be acted on through a URL. So, an API is not a "programming language", but it is the way a programming language is used to access a storage cloud.

REST APIs are also about changing the state of resource through representations of those resources. They are not about calling web service methods in a functional sense. The key differences between different Cloud Storage APIs are the URLs defining the resources and the format of the representations. Amazon S3 APIs,

Eucalyptus APIs, Rackspace Cloud Files APIs, Mezeo APIs, Nivanix APIs, Simple Cloud API, along with the standards proposed by the Storage Networking Industry Association (SNIA) Cloud Storage Technical Work Group, and more.

IV. ISSUES IN CLOUD STORAGE

Cloud storage gets the attention of IT managers with its comparatively low cost and ability to easily adjust capacity. Although cloud storage offers reduction in the capital investment cost, customers has to face some of the technical, integration, security and organizational issues at various levels. Also Fig. 9 shows how IT people are hesitating to take up the cloud services.

- **Control over the Data:** Since the data is residing outside the enterprise's infrastructure, it is perceived that the enterprise may loss the control over data. Although the concerns are largely hypothetical and psychological rather than actual, due to the immaturity of cloud services, standards on the delivery of services and their evolving business model, users may have genuine concerns about the service provider's viability and operational processes.
- **Interoperability & Control:** The complexity of using cloud storage is something many customers underestimate "It's not plug-and-play." Each vendor has different access methods, nonstandard APIs that make integrating applications, such as archiving or file shares with cloud storage, difficult and costly. Some vendors provide software clients that implement common network file sharing protocols such as Network File System (NFS) or Common Internet File System (CIFS), but these are proprietary and cannot bridge between different cloud services. The lack of standard protocols for accessing

cloud storage means there is no interoperability between cloud storage providers, greatly complicating the data migration.

- **Performance & Security:** Access to cloud data is obviously limited by network throughput and latency, and despite of drastic improvements in Internet performance, it is still poor in comparison to local network storage. Although some vendors attempt to enhance throughput with various local caching and compression techniques, these don't improve Internet latency. Data security is the biggest issue with cloud storage. If any possibility leakage, both in transfer and within a shared infrastructure, experts agree that using encryption on all data stored in a cloud is essential although depending on the application, this is easier said than done.
- **Suitability of applications:** A kind of more static data, inactive data, such as applications that include online backup and archiving is the best fit for cloud storage. The archiving kind of data works well in the cloud because the data changes less frequently. These data don't require high speed transactional access. Bulk data can be easily compressed using data reduction technologies as well as it can be easily encrypted. Applications like content delivery of rich media such as video, audio, or image files, Web 2.0 applications, user files and email repositories are some examples for best fit into the cloud storage. The applications with low I/O performance and tolerance for low downtime are suitable residents in cloud storage

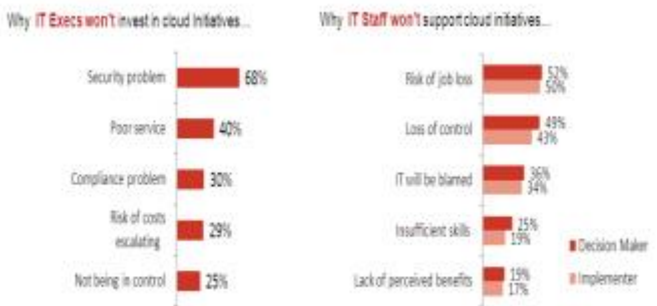


Fig. 9 Reasons why IT personnel hesitate to take up cloud services

RESULT:

Thus the study of cloud storage is done successfully