

Experiment 1

AIM: To write a program for SQL Data Definition Language Commands on sample exercise

Program:

SQL> connect

Enter user-name: system

Enter password: admin

Connected.

SQL> create table emp(id number(10),name varchar(10));

Table created.

SQL> desc emp;

Name	Null?	Type
ID		NUMBER(10)
NAME		VARCHAR2(10)

SQL> alter table emp add(dept varchar(10));

Table altered.

SQL> desc emp;

Name	Null?	Type
ID		NUMBER(10)
NAME		VARCHAR2(10)
DEPT		VARCHAR2(10)

SQL> alter table emp modify dept varchar(20);

Table altered.

SQL> desc emp;

Name	Null?	Type

ID		NUMBER(10)
NAME		VARCHAR2(10)
DEPT		VARCHAR2(20)

SQL> alter table emp drop column dept;

Table altered.

SQL> desc emp;

Name	Null?	Type

ID		NUMBER(10)
NAME		VARCHAR2(10)

SQL> alter table emp rename to emp1;

Table altered.

SQL> desc emp1;

Name	Null?	Type

ID		NUMBER(10)
NAME		VARCHAR2(10)

SQL> desc emp2;

Name	Null?	Type

ID		NUMBER(10)
NAME		VARCHAR2(10)
DEPT		VARCHAR2(10)

SQL> drop table emp2;

Table dropped.

```
SQL> select * from emp2;  
select * from emp2  
      *
```

ERROR at line 1:

ORA-00942: table or view does not exist

```
SQL> select * from emp1;
```

ID	NAME	DEPT
1	aaa	cse
2	aaa	cse
3	aaa	ece
4	aaa	cse
5	aaa	cse

```
SQL> truncate table emp1;
```

Table truncated.

```
SQL> select * from emp1;
```

no rows selected

```
SQL> desc emp1;
```

Name	Null?	Type
ID		NUMBER(10)
NAME		VARCHAR2(10)
DEPT		VARCHAR2(10)

```
SQL> drop table emp1;
```

Table dropped.

```
SQL> select * from emp1;
```

```
select * from emp1
      *
```

ERROR at line 1:

ORA-00942: table or view does not exist

```
SQL> desc emp1;
```

ERROR:

ORA-04043: object emp1 does not exist

Output:

Result: Hence, the above query has been implemented successfully.

Experiment 2

AIM: To write a program for SQL Data Manipulation Language Commands on sample exercise

Program:

```
SQL> select * from emp1;
```

no rows selected

```
SQL> alter table emp1 add(dept varchar(10));
```

Table altered.

```
SQL> desc emp1;
```

Name	Null?	Type

ID		NUMBER(10)
NAME		VARCHAR2(10)
DEPT		VARCHAR2(10)

```
SQL> insert into emp1(id,name,dept) values (1,'aaa','cse');
```

1 row created.

```
SQL> select * from emp1;
```

ID	NAME	DEPT

1	aaa	cse

```
SQL> insert into emp1 values (2,'bbb','cse');
```

1 row created.

```
SQL> select * from emp1;
```

ID	NAME	DEPT
----	------	------

```

-----
1 aaa      cse
2 bbb      cse

```

```

SQL> insert into emp1 values(&id,&name,&dept);
Enter value for id: 3
Enter value for name: ccc
Enter value for dept: cse
old 1: insert into emp1 values(&id,&name,&dept')
new 1: insert into emp1 values(3,'ccc','cse')

```

1 row created.

```

SQL> select * from emp1;

```

```

      ID NAME      DEPT
-----
1 aaa      cse
2 bbb      cse
3 ccc      cse

```

```

SQL> insert into emp1 values(&id,&name,&dept);
Enter value for id: 4
Enter value for name: ddd
Enter value for dept: cse
old 1: insert into emp1 values(&id,&name,&dept')
new 1: insert into emp1 values(4,'ddd','cse')

```

1 row created.

```

SQL> /
Enter value for id: 5
Enter value for name: eee
Enter value for dept: cse
old 1: insert into emp1 values(&id,&name,&dept')
new 1: insert into emp1 values(5,'eee','cse')

```

1 row created.

SQL> select * from emp1;

ID	NAME	DEPT
1	aaa	cse
2	bbb	cse
3	ccc	cse
4	ddd	cse
5	eee	cse

SQL> update emp1 set name='BBB' where id=2;

1 row updated.

SQL> select * from emp1;

ID	NAME	DEPT
1	aaa	cse
2	BBB	cse
3	ccc	cse
4	ddd	cse
5	eee	cse

SQL> update emp1 set name='CCC',dept='ece' where id=3;

1 row updated.

SQL> select * from emp1;

ID	NAME	DEPT
1	aaa	cse
2	BBB	cse
3	CCC	ece
4	ddd	cse
5	eee	cse

```
SQL> update emp1 set name='aaa';
```

5 rows updated.

```
SQL> select * from emp1;
```

ID	NAME	DEPT
1	aaa	cse
2	aaa	cse
3	aaa	ece
4	aaa	cse
5	aaa	cse

```
SQL> create table emp2(id number(10),name varchar(10),dept varchar(10));
```

Table created.

```
SQL> insert into emp2 select * from emp1;
```

5 rows created.

```
SQL> select * from emp2;
```

ID	NAME	DEPT
1	aaa	cse
2	aaa	cse
3	aaa	ece
4	aaa	cse
5	aaa	cse

```
SQL> delete from emp2 where id=1;
```

1 row deleted.

```
SQL> select * from emp2;
```

ID	NAME	DEPT
----	------	------


```

-----
      2 aaa      cse
      3 aaa      ece
      4 aaa      cse
      5 aaa      cse

```

SQL> delete from emp2 where dept='cse';

3 rows deleted.

SQL> select * from emp2;

```

      ID NAME      DEPT
-----
      3 aaa      ece

```

SQL> delete from emp2;

1 row deleted.

SQL> select * from emp2;

no rows selected

SQL> select * from emp2;

no rows selected

Output:

Result: Hence, the above query has been implemented successfully.

Experiment 3

AIM:To write a program on SQL Data Control Language Commands and Transaction control commands to the sample exercises

Program:

SQL> connect

Enter user-name: system

Enter password:

Connected.

SQL> create table class1(id number(10),name varchar(10));

Table created.

SQL> insert into class1 values(5,'rahul');

1 row created.

SQL> update class1 set name='raj' where id=5;

1 row updated.

SQL> savepoint A;

Savepoint created.

SQL> insert into class1 values(6,'ram');

1 row created.

SQL> insert into class1 values(7,'vibhav');

1 row created.

SQL> savepoint B;

Savepoint created.

```
SQL> insert into class1 values(8,'sai');
```

1 row created.

```
SQL> savepoint C;
```

Savepoint created.

```
SQL> select * from class1;
```

ID	NAME
5	raj
6	ram
7	vibhav
8	sai

```
SQL> rollback to B;
```

Rollback complete.

```
SQL> select * from class1;
```

ID	NAME
5	raj
6	ram
7	vibhav

```
SQL> rollback to A;
```

Rollback complete.

```
SQL> select * from class1;
```

ID	NAME
----	------

5 raj

SQL> insert into class1 values(6,'ram');

1 row created.

SQL> insert into class1 values(7,'vibhav');

1 row created.

SQL> insert into class1 values(8,'sai');

1 row created.

SQL> savepoint D;

Savepoint created.

SQL> insert into class1 values(9,'siva');

1 row created.

SQL> commit;

Commit complete.

SQL> select * from class1;

ID	NAME
5	raj
6	ram
7	vibhav
8	sai
9	siva

SQL> rollback to D;

rollback to D

*

ERROR at line 1:

ORA-01086: savepoint 'D' never established

SQL> insert into class1 values(10,'tom');

1 row created.

SQL> savepoint E;

Savepoint created.

SQL> insert into class1 values(11,'sam');

1 row created.

SQL> savepoint F;

Savepoint created.

SQL> rollback to E;

Rollback complete.

SQL> select * from class1;

ID	NAME
5	raj
6	ram
7	vibhav
8	sai
9	siva
10	tom

6 rows selected.

SQL> commit;

Commit complete.

SQL> select * from class1;

ID	NAME
5	raj
6	ram
7	vibhav
8	sai
9	siva
10	tom

6 rows selected.

SQL> rollback to F;

rollback to F

*

ERROR at line 1:

ORA-01086: savepoint 'F' never established

SQL>

Output:

Result: Hence, the above query has been implemented successfully.

Experiment 4

AIM: To write a program on Inbuilt functions in SQL on sample exercise.

Program:

```
SQL> create table stu1(id number(10),name varchar(10),department  
varchar(10),mark1 number(10),mark2 number(10),mark3 number(10));
```

Table created.

```
SQL> insert into stu1  
values(&id,&'name','&department',&mark1,&mark2,&mark3);
```

Enter value for id: 100

Enter value for name: aaa

Enter value for department: cse

Enter value for mark1: 90

Enter value for mark2: 89

Enter value for mark3: 95

```
old 1: insert into stu1  
values(&id,&'name','&department',&mark1,&mark2,&mark3)
```

```
new 1: insert into stu1 values(100,'aaa','cse',90,89,95)
```

1 row created.

```
SQL> /
```

Enter value for id: 101

Enter value for name: bbb

Enter value for department: cse

Enter value for mark1: 88

Enter value for mark2: 89

Enter value for mark3: 90

old 1: insert into stu1

values(&id,&name','&department',&mark1,&mark2,&mark3)

new 1: insert into stu1 values(101,'bbb','cse',88,89,90)

1 row created.

SQL> /

Enter value for id: 102

Enter value for name: ccc

Enter value for department: cse

Enter value for mark1: 90

Enter value for mark2: 88

Enter value for mark3: 87

old 1: insert into stu1

values(&id,&name','&department',&mark1,&mark2,&mark3)

new 1: insert into stu1 values(102,'ccc','cse',90,88,87)

1 row created.

SQL> /

Enter value for id: 103

Enter value for name: ddd

Enter value for department: cse

Enter value for mark1: 75

Enter value for mark2: 80

Enter value for mark3: 85

old 1: insert into stu1

values(&id,'&name','&department',&mark1,&mark2,&mark3)

new 1: insert into stu1 values(103,'ddd','cse',75,80,85)

1 row created.

SQL> select * from stu1;

ID	NAME	DEPARTMENT	MARK1	MARK2	MARK3
100	aaacse	90	89	95	
101	bbbcse	88	89	90	
102	ccccse	90	88	87	
103	dddcse	75	80	85	

SQL> select count(*) from stu1;

COUNT(*)

```
SQL> insert into stu1
values(&id,'&name','&department',&mark1,&mark2,&mark3);
```

Enter value for id: 104

Enter value for name: eee

Enter value for department: ece

Enter value for mark1: 88

Enter value for mark2: 87

Enter value for mark3: 88

old 1: insert into stu1

values(&id,'&name','&department',&mark1,&mark2,&mark3)

new 1: insert into stu1 values(104,'eee','ece',88,87,88)

1 row created.

```
SQL> select count(*) from stu1;
```

COUNT(*)

5

```
SQL> select count(*) from stu1 where department='ece';
```

COUNT(*)

1

```
SQL> select count(*) from stu1 where department='cse';
```

```
COUNT(*)
```

```
-----
```

```
4
```

```
SQL> select count(id) from stu1;
```

```
COUNT(ID)
```

```
-----
```

```
5
```

```
SQL> select count(id) from stu1 where department='cse';
```

```
COUNT(ID)
```

```
-----
```

```
4
```

```
SQL> select count(id) from stu1 where department='ece';
```

```
COUNT(ID)
```

```
-----
```

1

```
SQL> select min(mark1) from stu1;
```

```
MIN(MARK1)
```

```
-----
```

```
75
```

```
SQL> select min(mark2) from stu1;
```

```
MIN(MARK2)
```

```
-----
```

```
80
```

```
SQL> select min(mark3) from stu1;
```

```
MIN(MARK3)
```

```
-----
```

```
85
```

```
SQL> select min(mark1+mark2) from stu1;
```

```
MIN(MARK1+MARK2)
```

```
-----
```

155

```
SQL> select min(mark2+mark3) from stu1;
```

```
MIN(MARK2+MARK3)
```

```
-----
```

165

```
SQL> select min(mark1+mark3) from stu1;
```

```
MIN(MARK1+MARK3)
```

```
-----
```

160

```
SQL> select max(mark1) from stu1;
```

```
MAX(MARK1)
```

```
-----
```

90

```
SQL> select max(mark2) from stu1;
```

```
MAX(MARK2)
```

```
-----
```

```
SQL> select max(mark3) from stu1;
```

```
MAX(MARK3)
```

```
-----
```

```
95
```

```
SQL> select max(mark1+mark2) from stu1;
```

```
MAX(MARK1+MARK2)
```

```
-----
```

```
179
```

```
SQL> select max(mark2+mark3) from stu1;
```

```
MAX(MARK2+MARK3)
```

```
-----
```

```
184
```

```
SQL> select max(mark1+mark3) from stu1;
```

```
MAX(MARK1+MARK3)
```

```
-----
```

```
SQL> select max(mark2) from stu1 where department='cse';
```

```
MAX(MARK2)
```

```
-----
```

```
89
```

```
SQL> select min(mark1) from stu1 where department='cse';
```

```
MIN(MARK1)
```

```
-----
```

```
75
```

```
SQL> select avg(mark1) from stu1;
```

```
AVG(MARK1)
```

```
-----
```

```
86.2
```

```
SQL> select avg(mark2) from stu1;
```

```
AVG(MARK2)
```

```
-----
```

86.6

```
SQL> select avg(mark3) from stu1;
```

AVG(MARK3)

89

```
SQL> select avg(mark3) from stu1 where department='cse';
```

AVG(MARK3)

89.25

```
SQL> select sum(mark1) from stu1;
```

SUM(MARK1)

431

```
SQL> select sum(mark2) from stu1;
```

SUM(MARK2)

433

```
SQL> select sum(mark3) from stu1;
```

```
SUM(MARK3)
```

```
-----
```

445

```
SQL> select sum(mark1+mark2) from stu1;
```

```
SUM(MARK1+MARK2)
```

```
-----
```

864

```
SQL> select sum(mark1+mark3) from stu1;
```

```
SUM(MARK1+MARK3)
```

```
-----
```

876

```
SQL> select sum(mark2+mark3) from stu1;
```

```
SUM(MARK2+MARK3)
```

```
-----
```

878

```
SQL> select sum(mark3) from stu1 where department='cse';
```

```
SUM(MARK3)
```

```
-----
```

```
357
```

```
SQL>
```

Output:















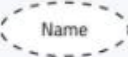





Result: Hence, the above query has been implemented successfully.

Experiment 5

AIM: To Construct a ER Model for the application to be constructed to a Database

Steps for drawing ER Diagram:

1. First, identify the entities in your database. In this case, we have three entities.
2. The second step involves identifying the **relationships** between the selected entities.
3. The third step involves **identifying cardinalities**.
4. The fourth step is **identifying entity attributes**. Make sure that every attribute is mapped to only one entity; assign modifiers for those that belong to more than one.
5. Once you have identified the entities, relationships, cardinalities, and attributes, you can now create your ER diagram. Here's what our sample project will look like when designed using the crow's foot (IE) notation.
6. **Entity:** Entities are represented by **rectangle**. All table of database are treating as entity.
7. **Attributes:** Attributes are represented by **ellipses**. Attributes are properties of entities.

Meaning	Notation	Alternative notation
Entity		
Weak Entity		
Attribute (mandatory)		
Primary identifier attribute		
Partial identifier attribute		
Alternate identifier attribute		
Multi-valued attribute		
Derived attribute		
Composite attribute		
Optional attribute		

Schools Entity : Attributes of Schools are school_id, school_name, school_type, school_description

Students Entity : Attributes of Students are student_id, student_college_id, student_name, student_mobile, student_email, student_username, student_password, student_address

Classes Entity : Attributes of Classes are class_id, class_student_id, class_name, class_room, class_type, class_description

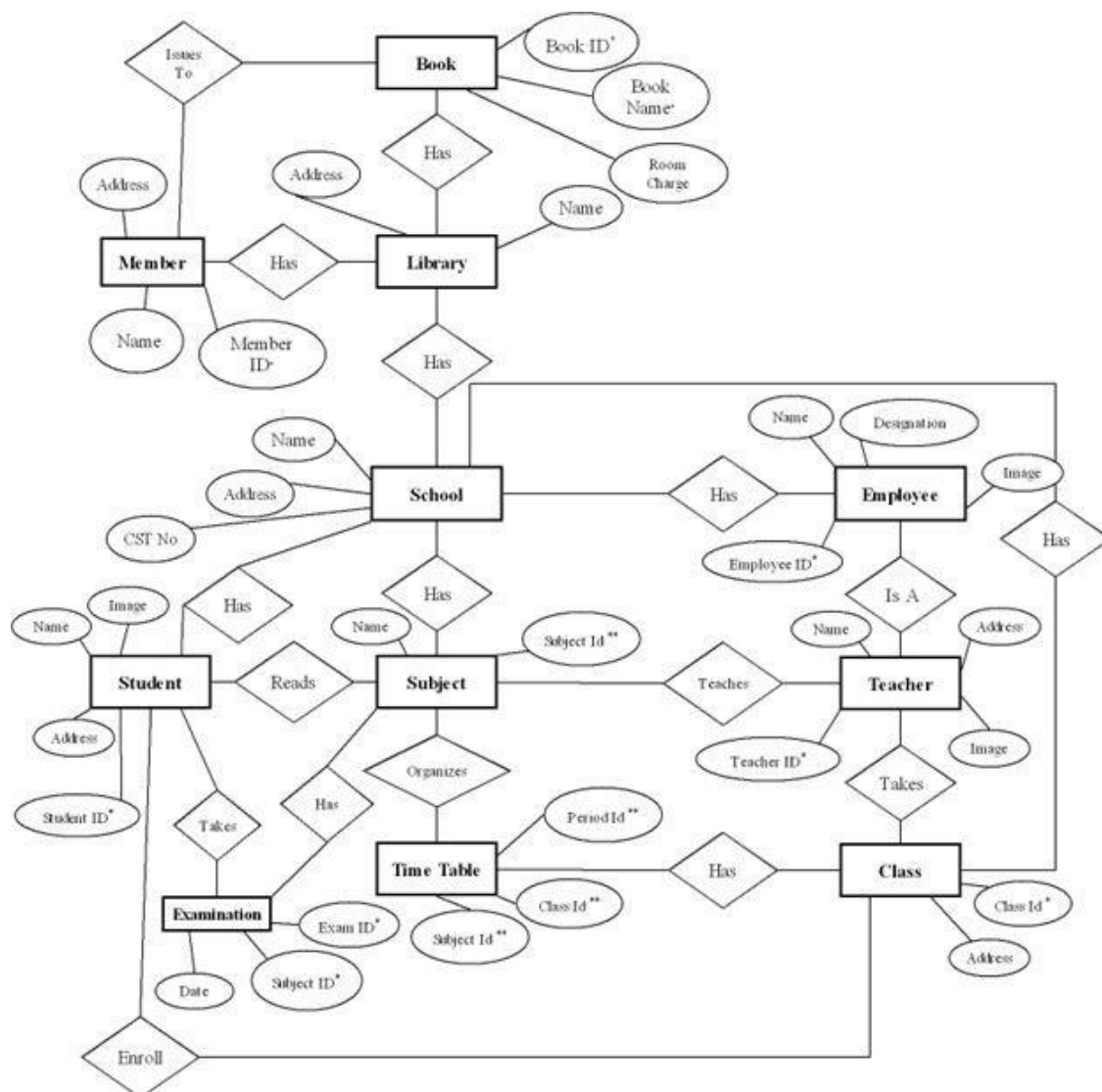
Teachers Entity : Attributes of Teachers are teacher_id, teacher_college_id, teacher_name, teacher_mobile, teacher_email, teacher_username, teacher_password, teacher_address

Cources Entity : Attributes of Cources are course_id, course_student_id, course_registration, course_name, course_type, course_year, course_description

Registrations Entity : Attributes of Registrations are registration_id, registration_student_id, registration_course_id, registration_name, registration_type, registration_number, registration_date, registration_description

(Link to draw ER diagram: <https://app.diagrams.net>)

Output:



Result: Hence we have successfully drawn the ER diagram.

Experiment: 6

AIM: To write a program for Nested Queries on sample exercise

Program :

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

no rows selected

```
SQL> desc emp;
```

Name	Null?	Type

ID		NUMBER(10)
NAME		VARCHAR2(10)
AGE		NUMBER(10)
ADDRESS		VARCHAR2(10)
SALARY		NUMBER(10)

```
SQL> insert into emp values(&id,&name,&age,&address,&salary);
```

Enter value for id: 10

Enter value for name: ram

Enter value for age: 33

Enter value for address: chennai

Enter value for salary: 50000

old 1: insert into emp values(&id,&name,&age,&address,&salary)

new 1: insert into emp values(10,'ram',33,'chennai',50000)

1 row created.

```
SQL> /
```

Enter value for id: 20

Enter value for name: raj

Enter value for age: 20

Enter value for address: chennai

Enter value for salary: 20000

old 1: insert into emp values(&id,&name,&age,&address,&salary)

new 1: insert into emp values(20,'raj',20,'chennai',20000)

1 row created.

SQL> /

Enter value for id: 30

Enter value for name: sai

Enter value for age: 26

Enter value for address: mumbai

Enter value for salary: 15000

old 1: insert into emp values(&id,&name,&age,&address,&salary)

new 1: insert into emp values(30,'sai',26,'mumbai',15000)

1 row created.

SQL> /

Enter value for id: 40

Enter value for name: sam

Enter value for age: 27

Enter value for address: hyderabad

Enter value for salary: 20000

old 1: insert into emp values(&id,&name,&age,&address,&salary)

new 1: insert into emp values(40,'sam',27,'hyderabad',20000)

1 row created.

SQL> /

Enter value for id: 50

Enter value for name: tom

Enter value for age: 29

Enter value for address: pune

Enter value for salary: 65000

old 1: insert into emp values(&id,&name,&age,&address,&salary)

new 1: insert into emp values(50,'tom',29,'pune',65000)

1 row created.

SQL> /

Enter value for id: 60

Enter value for name: jerry

Enter value for age: 30

Enter value for address: kochin

Enter value for salary: 85000

old 1: insert into emp values(&id,&name,&age,&address,&salary)

new 1: insert into emp values(60,'jerry',30,'kochin',85000)

1 row created.

SQL> select * from emp;

ID	NAME	AGE	ADDRESS	SALARY
10	ram	33	chennai	50000
20	raj	20	chennai	20000
30	sai	26	mumbai	15000
40	sam	27	hyderabad	20000
50	tom	29	pune	65000
60	jerry	30	kochin	85000

6 rows selected.

SQL> select * from emp;

ID	NAME	AGE	ADDRESS	SALARY
10	ram	33	chennai	12500
20	raj	20	chennai	20000
30	sai	26	mumbai	15000
40	sam	27	hyderabad	20000
50	tom	29	pune	16250
60	jerry	30	hyderabad	25250
70	jack	33	bangalore	60000
80	jill	38	bangalore	65000
90	donald	23	chennai	10000

SQL> select max(salary) from emp;

MAX(SALARY)

65000

SQL> select id,name from emp where salary=65000;

ID NAME

80 jill

SQL> select id,name from emp where salary=(select max(salary) from emp);

ID NAME

80 jill

SQL> select * from emp where salary>(select avg(salary) from emp);

ID NAME	AGE ADDRESS	SALARY
---------	-------------	--------

70 jack	33 bangalore	60000
---------	--------------	-------

80 jill	38 bangalore	65000
---------	--------------	-------

SQL> select * from emp where salary>(select avg(salary) from emp)order by age desc;

ID NAME	AGE ADDRESS	SALARY
---------	-------------	--------

80 jill	38 bangalore	65000
---------	--------------	-------

70 jack	33 bangalore	60000
---------	--------------	-------

SQL> select * from emp where salary<(select avg(salary) from emp)order by age desc;

ID NAME	AGE ADDRESS	SALARY
---------	-------------	--------

10 ram	33 chennai	12500
--------	------------	-------

60 jerry	30 hyderabad	25250
----------	--------------	-------

50 tom	29 pune	16250
--------	---------	-------

40 sam	27 hyderabad	20000
--------	--------------	-------

30 sai	26 mumbai	15000
--------	-----------	-------

90 donald	23 chennai	10000
-----------	------------	-------

20 raj 20 chennai 20000
SQL> select * from emp where salary<=(select avg(salary) from emp);

ID NAME	AGE ADDRESS	SALARY
10 ram	33 chennai	12500
20 raj	20 chennai	20000
30 sai	26 mumbai	15000
40 sam	27 hyderabad	20000
50 tom	29 pune	16250
60 jerry	30 hyderabad	25250
90 donald	23 chennai	10000

7 rows selected.

SQL> select * from emp where salary>=(select avg(salary) from emp);

ID NAME	AGE ADDRESS	SALARY
70 jack	33 bangalore	60000
80 jill	38 bangalore	65000

SQL> insert into emp values(100,'gow',30,'chennai',27100);

1 row created.

SQL> select * from emp where salary<=(select avg(salary) from emp);

ID NAME	AGE ADDRESS	SALARY
10 ram	33 chennai	12500
20 raj	20 chennai	20000
30 sai	26 mumbai	15000
40 sam	27 hyderabad	20000
50 tom	29 pune	16250
60 jerry	30 hyderabad	25250
90 donald	23 chennai	10000
100 gow	30 chennai	27100

8 rows selected.

```
SQL> select * from emp where salary>=(select avg(salary) from emp);
```

ID	NAME	AGE	ADDRESS	SALARY
70	jack	33	bangalore	60000
80	jill	38	bangalore	65000

```
SQL> select * from emp where id in(select id from emp where salary>20000);
```

ID	NAME	AGE	ADDRESS	SALARY
10	ram	33	chennai	50000
50	tom	29	pune	65000
60	jerry	30	kochin	85000

```
SQL> select * from emp where id in(select id from emp where salary>=20000);
```

ID	NAME	AGE	ADDRESS	SALARY
10	ram	33	chennai	50000
20	raj	20	chennai	20000
40	sam	27	hyderabad	20000
50	tom	29	pune	65000
60	jerry	30	kochin	85000

```
SQL> update emp set salary=salary*0.25 where age in(select age from emp where age>=29);
```

3 rows updated.

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

ID	NAME	AGE	ADDRESS	SALARY
10	ram	33	chennai	12500

20 raj	20 chennai	20000
30 sai	26 mumbai	15000
40 sam	27 hyderabad	20000
50 tom	29 pune	16250
60 jerry	30 kochin	21250

6 rows selected.

SQL> update emp set salary=salary*10 where age not in(select age from emp where age>=30);

5 rows updated.

SQL> select * from emp;

ID	NAME	AGE	ADDRESS	SALARY
10	ram	33	chennai	12500
20	raj	20	chennai	200000
30	sai	26	mumbai	150000
40	sam	27	hyderabad	200000
50	tom	29	pune	162500
60	jerry	30	hyderabad	25250
70	jack	33	bangalore	60000
80	jill	38	bangalore	65000
90	donald	23	chennai	100000
100	gow	30	chennai	27100

SQL> select * from emp where salary<any(select max(salary) from emp);

ID	NAME	AGE	ADDRESS	SALARY
10	ram	33	chennai	12500
30	sai	26	mumbai	150000
50	tom	29	pune	162500
60	jerry	30	hyderabad	25250
70	jack	33	bangalore	60000
80	jill	38	bangalore	65000
90	donald	23	chennai	100000

100 gow 30 chennai 27100

SQL> select * from emp where salary=all(select salary from emp where salary<=20000);

ID NAME	AGE ADDRESS	SALARY
10 ram	33 chennai	12500

SQL> select * from emp where salary>all(select salary from emp where salary<=20000);

ID NAME	AGE ADDRESS	SALARY
20 raj	20 chennai	200000
30 sai	26 mumbai	150000
40 sam	27 hyderabad	200000
50 tom	29 pune	162500
60 jerry	30 hyderabad	25250
70 jack	33 bangalore	60000
80 jill	38 bangalore	65000
90 donald	23 chennai	100000
100 gow	30 chennai	27100

SQL> delete from emp where age in(select age from emp where age>=30);

2 rows deleted.

SQL> select * from emp;

ID NAME	AGE ADDRESS	SALARY
20 raj	20 chennai	20000
30 sai	26 mumbai	15000
40 sam	27 hyderabad	20000
50 tom	29 pune	16250

VIEWS

SQL> connect

Enter user-name: system

Enter password:

Connected.

SQL> create taable s(sid number(10),sname varchar(10),address varchar(10));

create taable s(sid number(10),sname varchar(10),address varchar(10))

*

ERROR at line 1:

ORA-00901: invalid CREATE command

SQL> create table s(sid number(10),sname varchar(10),address varchar(10));

Table created.

SQL> insert into s values(1,'ram','chennai');

1 row created.

SQL> insert into s values(2,'raj','chennai');

1 row created.

```
SQL> insert into s values(3,'siva','pune');
```

1 row created.

```
SQL> insert into s values(4,'selvam','bangalore');
```

1 row created.

```
SQL> select * from s;
```

SID	SNAME	ADDRESS
1	ram	chennai
2	raj	chennai
3	siva	pune
4	selvam	bangalore

```
SQL> create table s1(sid number(10),sname varchar(10),marks number(10),age  
varchar(10));
```

Table created.

```
SQL> insert into s1 values(1,'ram',90,20);
```

1 row created.

```
SQL> insert into s1 values(2,'raj',90,21);
```

1 row created.

```
SQL> insert into s1 values(3,'siva',80,19);
```

1 row created.

```
SQL> insert into s1 values(4,'selvam',88,19);
```

1 row created.

```
SQL> select * from s1;
```

SID	SNAME	MARKS	AGE
1	ram	90	20
2	raj	90	21
3	siva	80	19
4	selvam	88	19

```
SQL> create view sv as select sname,address from s where sid<4;
```

View created.

```
SQL> select * from sv;
```

SNAME	ADDRESS
-------	---------

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

ram	chennai
-----	---------

raj	chennai
-----	---------

siva	pune
------	------

```
SQL> create view sv1 as select s.sname,s.address,s1.marks from s,s1 where  
s.sid=s1.sid;
```

View created.

```
SQL> select * from sv1;
```

SNAME	ADDRESS	MARKS
-------	---------	-------

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

ram	chennai	90
-----	---------	----

raj	chennai	90
-----	---------	----

siva	pune	80
------	------	----

selvam	bangalore	88
--------	-----------	----


```
SQL> drop view sv1;
```

View dropped.

```
SQL> select * from sv1;
```

```
select * from sv1
```

```
        *
```

ERROR at line 1:

ORA-00942: table or view does not exist

Output:

Result: Hence the above query has been implemented successfully.

Experiment 7

AIM: To write a program for Join Queries on sample exercise.

Program:

```
SQL> create table fac(fid number(10),fname varchar(10),address  
varchar(10),age number(10));
```

Table created.

```
SQL> create table fcou(cid number(10),fid number(10));
```

Table created.

```
SQL> insert into fac values(&fid,&fname,&address,&age);
```

Enter value for fid: 1

Enter value for fname: aaa

Enter value for address: chennai

Enter value for age: 30

```
old 1: insert into fac values(&fid,&fname,&address,&age)
```

```
new 1: insert into fac values(1,'aaa','chennai',30)
```

1 row created.

```
SQL> /
```

Enter value for fid: 2

Enter value for fname: bbb

Enter value for address: chennai

Enter value for age: 33

old 1: insert into fac values(&fid,&fname,&address,&age)

new 1: insert into fac values(2,'bbb','chennai',33)

1 row created.

SQL> /

Enter value for fid: 3

Enter value for fname: ccc

Enter value for address: bangalore

Enter value for age: 34

old 1: insert into fac values(&fid,&fname,&address,&age)

new 1: insert into fac values(3,'ccc','bangalore',34)

1 row created.

SQL> /

Enter value for fid: 4

Enter value for fname: ddd

Enter value for address: kochin

Enter value for age: 30

old 1: insert into fac values(&fid,&fname,&address,&age)

new 1: insert into fac values(4,'ddd','kochin',30)

1 row created.

SQL> /

Enter value for fid: 5

Enter value for fname: eee

Enter value for address: hyderabad

Enter value for age: 30

old 1: insert into fac values(&fid,&fname,&address,&age)

new 1: insert into fac values(5,'eee','hyderabad',30)

1 row created.

SQL> /

Enter value for fid: 6

Enter value for fname: fff

Enter value for address: hyderabad

Enter value for age: 29

old 1: insert into fac values(&fid,&fname,&address,&age)

new 1: insert into fac values(6,'fff','hyderabad',29)

1 row created.

SQL> /

Enter value for fid: 7

Enter value for fname: ggg

Enter value for address: chennai

Enter value for age: 33

old 1: insert into fac values(&fid,'&fname','&address',&age)

new 1: insert into fac values(7,'ggg','chennai',33)

1 row created.

SQL> select * from fac;

FID	FNAME	ADDRESS	AGE
1	aaa	chennai	30
2	bbb	chennai	33
3	ccc	bangalore	34
4	ddd	kochin	30
5	eee	hyderabad	30
6	fff	hyderabad	29
7	ggg	chennai	33

7 rows selected.

SQL> insert into fcou values(10,1);

1 row created.

```
SQL> insert into fcou values(10,2);
```

1 row created.

```
SQL> insert into fcou values(20,3);
```

1 row created.

```
SQL> insert into fcou values(30,4);
```

1 row created.

```
SQL> insert into fcou values(20,5);
```

1 row created.

```
SQL> insert into fcou values(20,6);
```

1 row created.

```
SQL> select * from fcou;
```

CID	FID
-----	-----

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

10	1
10	2
20	3
30	4
20	5
20	6

6 rows selected.

SQL> select * from fac;

	FID	FNAME	ADDRESS	AGE

	1	aaa	chennai	30
	2	bbb	chennai	33
	3	ccc	bangalore	34
	4	ddd	kochin	30
	5	eee	hyderabad	30
	6	fff	hyderabad	29
	7	ggg	chennai	33

7 rows selected.

SQL> select fcou.cid,fac.fname,fac.age from fac inner join fcou on
fac.fid=fcou.fid;

CID	FNAME	AGE
-----	-------	-----

```

-----
      10 aaa      30
      10 bbb      33
      20 ccc      34
      30 ddd      30
      20 eee      30
      20 fff      29

```

6 rows selected.

SQL> select fac.fname,fcou.cid from fac left join fcou on fcou.fid=fac.fid;

```

FNAME      CID
-----
aaa         10
bbb         10
ccc         20
ddd         30
eee         20
fff         20
ggg

```

7 rows selected.


```
SQL> select fac.fid,fcou.cid from fac left join fcou on fcou.fid=fac.fid;
```

FID	CID
1	10
2	10
3	20
4	30
5	20
6	20
7	

7 rows selected.

```
SQL> select fac.fid,fcou.cid from fac right join fcou on fcou.fid=fac.fid;
```

FID	CID
1	10
2	10
3	20
4	30
5	20
6	20

6 rows selected.

```
SQL> insert into fcou values(40,7);
```

1 row created.

```
SQL> insert into fcou values(40,8);
```

1 row created.

```
SQL> select fac.fid,fcou.cid from fac right join fcou on fcou.fid=fac.fid;
```

FID	CID
1	10
2	10
3	20
4	30
5	20
6	20
7	40
	40

8 rows selected.

```
SQL> select fac.fid,fcou.cid from fac full join fcou on fcou.fid=fac.fid;
```

FID	CID
1	10
2	10
3	20
4	30
5	20
6	20
7	40
	40

8 rows selected.

```
SQL> select fcou.fid,fcou.cid from fac full join fcou on fcou.fid=fac.fid;
```

FID	CID
1	10
2	10
3	20

4	30
5	20
6	20
7	40
8	40

8 rows selected.

SQL> select fac.fid,fcou.fid from fac full join fcou on fcou.fid=fac.fid;

FID	FID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
	8

8 rows selected.

Output:

Result: Hence the above query has been implemented successfully.

Experiment: 8

AIM: To implement the program for Set Operators & Views.

Program:

```
SQL> create table z(id number(10),name varchar(10));
```

Table created.

```
SQL> insert into z values(1,'a');
```

1 row created.

```
SQL> insert into z values(2,'b');
```

1 row created.

```
SQL> create table z1(id number(10),name varchar(10));
```

Table created.

```
SQL> insert into z1 values(2,'b');
```

1 row created.

```
SQL> insert into z1 values(3,'c');
```

1 row created.

```
SQL> insert into z1 values(4,'d');
```

1 row created.

```
SQL> select * from z;
```

ID	NAME
----	------

1	a
2	b

```
SQL> select * from z1;
```

ID	NAME
----	------

2	b
3	c
4	d

```
SQL> select * from z union select * from z1;
```

ID	NAME
1	a
2	b
3	c
4	d

```
SQL> select * from z union all select * from z1;
```

ID	NAME
1	a
2	b
2	b
3	c
4	d

```
SQL> select * from z intersect select * from z1;
```

ID NAME

2 b

SQL> select * from z minus select * from z1;

ID NAME

1 a

SQL> select * from z1 minus select * from z;

ID NAME

3 c

4 d

OUTPUT:

Result: The above query has been implemented successfully

Experiment: 9

AIM: To write a program on PL/SQL Conditional and Iterative Statements

Program:

Set serveroutput on

SQL> declare

2 a integer:=10;

3 begin

4 dbms_output.put_line(a);

5 end;

6 /

10

PL/SQL procedure successfully completed.

SQL> declare

2 a varchar(10):='hello';

3 begin

4 dbms_output.put_line(a);

5 end;

6 /

hello

PL/SQL procedure successfully completed.

SQL> declare

2 a integer:=10;

```
3 begin
4 dbms_output.put_line('value of a:'||a);
5 end;
6 /
```

value of a:10

PL/SQL procedure successfully completed.

SQL> declare

```
2 pi constant number:=3.14;
3 begin
4 dbms_output.put_line(pi);
5 end;
6 /
```

3.14

PL/SQL procedure successfully completed.

SQL> declare

```
2 a integer:=10;
3 b integer:=20;
4 c integer;
5 begin
6 c:=a+b;
7 dbms_output.put_line(c);
8 end;
```

9 /

30

IF-ELSE

PL/SQL procedure successfully completed.

SQL> declare

2 a integer:=30;

3 begin

4 if(a<20) then

5 dbms_output.put_line('a is less than 20');

6 else

7 dbms_output.put_line('a is not less than 20');

8 end if;

9 end;

10 /

a is not less than 20

CASE STATEMENT

PL/SQL procedure successfully completed.

SQL> DECLARE

2 grade char(1) := 'A';

3 BEGIN

4 CASE grade

5 when 'A' then dbms_output.put_line('Excellent');

6 when 'B' then dbms_output.put_line('Very good');

7 when 'C' then dbms_output.put_line('Good');

```
8    when 'D' then dbms_output.put_line('Average');
9    when 'F' then dbms_output.put_line('Passed with Grace');
10   else dbms_output.put_line('Failed');
11   END CASE;
12 END;
13 /
```

Excellent

LOOP

PL/SQL procedure successfully completed.

SQL> DECLARE

```
2 i NUMBER := 1;
3 BEGIN
4 LOOP
5 EXIT WHEN i>10;
6 DBMS_OUTPUT.PUT_LINE(i);
7 i := i+1;
8 END LOOP;
9 END;
10 /
1
2
3
4
5
```

6

7

8

9

10

WHILE LOOP

PL/SQL procedure successfully completed.

SQL> DECLARE

2 i INTEGER := 1;

3 BEGIN

4 WHILE i <= 10 LOOP

5 DBMS_OUTPUT.PUT_LINE(i);

6 i := i+1;

7 END LOOP;

8 END;

9 /

1

2

3

4

5

6

7

8

9

10

FOR LOOP

PL/SQL procedure successfully completed.

SQL> DECLARE

2 VAR1 NUMBER;

3 BEGIN

4 VAR1:=10;

5 FOR VAR2 IN 1..10

6 LOOP

7 DBMS_OUTPUT.PUT_LINE (VAR1*VAR2);

8 END LOOP;

9 END;

10 /

10

20

30

40

50

60

70

80

90

100

PL/SQL procedure successfully completed.

OUTPUT:

Result: The above query has been implemented successfully.

Experiment: 10

AIM: To write a program on PL/SQL Procedures on sample exercises.

Program:

```
SQL> create table proc(id number(10),name varchar(10));
```

Table created.

```
SQL> create or replace procedure "INSERTUSER"
```

```
(id IN NUMBER,
```

```
name IN VARCHAR2)
```

```
is
```

```
begin
```

```
insert into proc values(id,name);
```

```
end;
```

```
/
```

Procedure created.

```
SQL> BEGIN
```

```
insertuser(101,'Rahul');
```

```
dbms_output.put_line('record inserted successfully');
```

```
END;
```

```
/
```

```
record inserted successfully
```

PL/SQL procedure successfully completed.


```
SQL> select * from proc;
```

ID	NAME
----	------

101	Rahul
-----	-------

OUTPUT:

Result: The above query has been implemented successfully

Experiment: 11

AIM: To write a program on PL/SQL Functions

Program:

```
SQL> create or replace function adder(n1 in number, n2 in number)
```

```
2 return number
```

```
3 is
```

```
4 n3 number(8);
```

```
5 begin
```

```
6 n3 :=n1+n2;
```

```
7 return n3;
```

```
8 end;
```

```
9 /
```

Function created.

```
SQL> DECLARE
```

```
2 n3 number(2);
```

```
3 BEGIN
```

```
4 n3 := adder(11,22);
```

```
5 dbms_output.put_line('Addition is: ' || n3);
```

```
6 END;
```

```
7 /
```

Addition is: 33

PL/SQL procedure successfully completed.

SQL> DECLARE

2 a number;

3 b number;

4 c number;

5 FUNCTION findMax(x IN number, y IN number)

6 RETURN number

7 IS

8 z number;

9 BEGIN

10 IF x > y THEN

11 z:= x;

12 ELSE

13 Z:= y;

14 END IF;

15

16 RETURN z;

17 END;

18 BEGIN

19 a:= 23;

20 b:= 45;

21

22 c := findMax(a, b);

23 dbms_output.put_line(' Maximum of (23,45): ' || c);

24 END;

25 /

Maximum of (23,45): 45

PL/SQL procedure successfully completed.

SQL> select * from emp;

ID	NAME	AGE	ADDRESS	SALARY
30	sai	26	mumbai	150000
40	sam	27	hyderabad	200000
50	tom	29	pune	40630

SQL> create or replace function totalemp

2 return number is

3 total number(10):=0;

4 begin

5 select count(*) into total from emp;

6 return total;

7 end;

8 /

Function created.

SQL> declare

2 c number(20);

3 begin

4 c:=totalemp();

5 dbms_output.put_line('Total no of emp:'||c);

6 end;

7 /

Total no of emp:3

PL/SQL procedure successfully completed.

SQL> DECLARE

2 num number;

3 factorial number;

4

5 FUNCTION fact(x number)

6 RETURN number

7 IS

8 f number;

9 BEGIN

10 IF x=0 THEN

11 f := 1;

12 ELSE

13 f := x * fact(x-1);

14 END IF;

```
15 RETURN f;
16 END;
17
18 BEGIN
19     num:= 6;
20     factorial := fact(num);
21     dbms_output.put_line(' Factorial ' || num || ' is ' || factorial);
22 END;
23 /
Factorial 6 is 720
```

PL/SQL procedure successfully completed.

OUTPUT:

Result: The above query has been implemented successfully

Experiment: 12

AIM: To write program on PL/SQL Cursors

Program:

```
SQL> DECLARE
```

```
total_rows number(2);
```

```
BEGIN
```

```
UPDATE emp
```

```
SET salary = salary + 5000;
```

```
IF sql%notfound THEN
```

```
dbms_output.put_line('no customers updated');
```

```
ELSIF sql%found THEN
```

```
total_rows := sql%rowcount;
```

```
dbms_output.put_line( total_rows || ' customers updated ');
```

```
END IF;
```

```
END;
```

```
/
```

```
3 customers updated
```

PL/SQL procedure successfully completed.

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

ID	NAME	AGE	ADDRESS	SALARY

30 sai	26 mumbai	155000
40 sam	27 hyderabad	205000
50 tom	29 pune	45630

EXPLICIT CURSOR

SQL> DECLARE

c_id emp.id%type;

c_name emp.name%type;

c_addr emp.address%type;

CURSOR c_emp is

SELECT id, name, address FROM emp;

begin

open c_emp;

loop

FETCH c_emp into c_id, c_name, c_addr;

EXIT WHEN c_emp%notfound;

dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);

end loop;

close c_emp;

end;

/

30 sai mumbai

40 sam hyderabad

50 tom pune

PL/SQL procedure successfully completed.

OUTPUT:

Result: The above query has been implemented successfully.

Experiment: 13

AIM: To write program on PL/SQL Exception Handling

Program:

```
SQL> DECLARE
```

```
c_id emp.id%type:=10;
```

```
c_name emp.name%type;
```

```
c_addr emp.address%type;
```

```
begin
```

```
SELECT name, address into c_name,c_addr FROM emp where id=c_id;
```

```
dbms_output.put_line('Name:'||c_name);
```

```
dbms_output.put_line('Address:'||c_addr);
```

```
exception
```

```
when no_data_found then dbms_output.put_line('no such customer');
```

```
when others then dbms_output.put_line('error');
```

```
end;
```

```
/
```

```
no such customer
```

PL/SQL procedure successfully completed.

```
SQL> declare
```

```
c_id emp.id%type:=30;
```

```
c_name emp.name%type;
```

```
c_addr emp.address%type;
```

```

begin
SELECT name, address into c_name,c_addr FROM emp where id=c_id;
dbms_output.put_line('Name:'||c_name);
dbms_output.put_line('Address:'||c_addr);
exception
when no_data_found then dbms_output.put_line('no such customer');
when others then dbms_output.put_line('error');
end;
/
Name:sai
Address:mumbai

```

PL/SQL procedure successfully completed.

USER DEFINED

```

SQL> declare
c_id emp.id%type:=&id;
c_name emp.name%type;
c_addr emp.address%type;
ex_invalid_id exception;
begin
if c_id<=0 then
raise ex_invalid_id;
else
select name,address into c_name,c_addr from emp where id=c_id;

```

```
dbms_output.put_line('name:'||c_name);  
dbms_output.put_line('address:'||c_addr);  
end if;  
exception  
when ex_invalid_id then dbms_output.put_line('id should be greater than zero');  
when no_data_found then dbms_output.put_line('no such customer');  
when others then dbms_output.put_line('error');  
end;  
/
```

Enter value for id: -6

```
old 2: c_id emp.id%type:=&id;  
new 2: c_id emp.id%type:=-6;  
id should be greater than zero
```

PL/SQL procedure successfully completed.

SQL> /

Enter value for id: 30

```
old 2: c_id emp.id%type:=&id;  
new 2: c_id emp.id%type:=30;  
name:sai  
address:mumbai
```

PL/SQL procedure successfully completed.

OUTPUT:

Result: The above query has been implemented successfully.

Experiment 14

AIM: To write a program on PL/SQL Trigger

Program:

```
SQL> CREATE OR REPLACE TRIGGER display_salary_changes
  BEFORE DELETE OR INSERT OR UPDATE ON emp
  FOR EACH ROW
  WHEN (NEW.ID > 0)
  DECLARE
    sal_diff number;
  BEGIN
    sal_diff := :NEW.salary - :OLD.salary;
    dbms_output.put_line('Old salary: ' || :OLD.salary);
    dbms_output.put_line('New salary: ' || :NEW.salary);
    dbms_output.put_line('Salary difference: ' || sal_diff);
  END;
/
```

Trigger created.

```
SQL> DECLARE
  total_rows number(2);
  BEGIN
  UPDATE emp
    SET salary = salary + 5000;
  IF sql%notfound THEN
```

```
dbms_output.put_line('no customers updated');  
ELSIF sql%found THEN  
    total_rows := sql%rowcount;  
dbms_output.put_line( total_rows || ' customers updated ');  
END IF;  
END;
```

/

Old salary: 155000

New salary: 160000

Salary difference: 5000

Old salary: 205000

New salary: 210000

Salary difference: 5000

Old salary: 45630

New salary: 50630

Salary difference: 5000

3 customers updated

PL/SQL procedure successfully completed.

Output:

Result: Hence the above program is implemented successfully.

Experiment 15

AIM: To make a project with front-end and back-end using oracle DBMS

Procedure:

1. Consider the database of at least 4 schemas/tables.
2. Plan for attributes that cover all general data types.
3. Plan for constraints of different types (primary keys, foreign keys, unique keys, check, not null etc) with appropriate constraint names.
4. Create appropriate data for the above schemas.
5. Implement the database in Oracle 10g Express Edition creating a user.
6. Plan queries and find the answers (at least 10).
7. Formulate everything in a report which includes:
 - Brief description of the system that you are going to implement in the database
 - Mention schemas with attributes
 - Expected functional dependencies on your schemas.
 - Proof of good database design so that the schemas are in good form (BCNF or 3NF)
 - Schema diagram of the database
 - E-R diagram of the database
 - Snapshots of SQL DDL of all the schemas/tables
 - Snapshots of the instances (data of the populated tables)
 - Query statements in English language, SQL statements and snapshots of the outputs. SQL statements should contain the following :
 - natural join, cross product, outer join, join with using, on
 - Nested sub-queries with clauses (some, all, any, exists, unique etc.)
 - order by, group by, having clauses
 - Use of with clause
 - Use of Functions (string/text, numeric) , set operations (union, intersect, difference/minus)

- Update, delete operations
- Conclusion of the work

Output:

Result: Hence the project was implemented successfully.