

## EXPERIMENT-1

1. Creating a Table :

```
SQL-CSE510>CREATE TABLE parts(  
2  part_id NUMBER PRIMARY KEY,  
3  part_name VARCHAR2(50) NOT NULL,  
4  unique_code VARCHAR2(20) NOT NULL,  
5  manufactured_date DATE NOT NULL,  
6  cost NUMBER(10,2) NOT NULL  
7  );
```

Table created.

2. Describing the attributes :

```
SQL-CSE510>DESC parts;
```

Name	Null?	Type
PART_ID	NOT NULL	NUMBER
PART_NAME	NOT NULL	VARCHAR2(50)
UNIQUE_CODE	NOT NULL	VARCHAR2(20)
MANUFACTURED_DATE	NOT NULL	DATE
COST	NOT NULL	NUMBER(10,2)

3. Altering the table :

```
SQL-CSE510>ALTER TABLE parts  
2  ADD quantity NUMBER NOT NULL;
```

Table altered.

4. Describing the attributes :

```
SQL-CSE510>DESC parts;
```

Name	Null?	Type
PART_ID	NOT NULL	NUMBER
PART_NAME	NOT NULL	VARCHAR2(50)
UNIQUE_CODE	NOT NULL	VARCHAR2(20)
MANUFACTURED_DATE	NOT NULL	DATE
COST	NOT NULL	NUMBER(10,2)
QUANTITY	NOT NULL	NUMBER

5. Altering the table :

```
SQL-CSE510>ALTER TABLE parts  
2  MODIFY manufactured_date DATE;
```

Table altered.

## 6. Describing the table :

```
SQL-CSE510>DESC parts;
```

Name	Null?	Type
PART_ID	NOT NULL	NUMBER
PART_NAME	NOT NULL	VARCHAR2(50)
UNIQUE_CODE	NOT NULL	VARCHAR2(20)
MANUFACTURED_DATE	NOT NULL	DATE
COST	NOT NULL	NUMBER(10,2)
QUANTITY	NOT NULL	NUMBER

## 7. Altering the table :

```
SQL-CSE510>ALTER TABLE parts
2 DROP COLUMN manufactured_date;
```

```
Table altered.
```

## 8. Describing the table :

```
SQL-CSE510>DESC parts;
```

Name	Null?	Type
PART_ID	NOT NULL	NUMBER
PART_NAME	NOT NULL	VARCHAR2(50)
UNIQUE_CODE	NOT NULL	VARCHAR2(20)
COST	NOT NULL	NUMBER(10,2)
QUANTITY	NOT NULL	NUMBER

## 9. Creating another table :

```
SQL-CSE510>CREATE TABLE boats(
2 boat_id NUMBER PRIMARY KEY,
3 boat_name VARCHAR2(50) NOT NULL
4 );
```

```
Table created.
```

## 10. Dropping the table :

```
SQL-CSE510>DROP TABLE boats;
```

```
Table dropped.
```

## 11. Error due to dropping :

```
SQL-CSE510>DESC boats;
```

```
ERROR:
```

```
ORA-04043: object boats does not exist
```

12. Inserting the values :

```
SQL-CSE510>INSERT INTO parts VALUES (101, 'XAT', 'XHJ45689023K', 999.89, 98);  
1 row created.
```

13. Truncating the table :

```
SQL-CSE510>TRUNCATE TABLE parts;  
  
Table truncated.  
  
SQL-CSE510>select * from PARTS;  
  
no rows selected
```

**CONCLUSION: CONCLUSION:**

SQL queries to CREATE TABLES for various databases using DDL commands (i.e., CREATE, ALTER, DROP, TRUNCATE) is successfully completed.

## EXPERIMENT -2

1. Login time:

```

C:\Users\SRIT-S1-10>cd desktop
The system cannot find the path specified.

C:\Users\SRIT-S1-10>cd desktop

C:\Users\SRIT-S1-10\Desktop>cd CSE-510

C:\Users\SRIT-S1-10\Desktop\CSE-510>SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

SQL*Plus: Release 11.2.0.2.0 Production on Thu Oct 5 10:47:02 2023

Copyright (c) 1982, 2014, Oracle. All rights reserved.

ERROR:
ORA-12514: TNS:listener does not currently know of service requested in connect
descriptor

Enter user-name: CSE510
Enter password:

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> _

```

2. Table creation of name person:

```

CSE510@XE 05-OCT-23>run
1 CREATE TABLE person(
2 person_id NUMBER PRIMARY KEY,
3 First_name VARCHAR2(50) NOT NULL,
4 Last_name VARCHAR2(50) NOT NULL,
5 phone_no NUMBER(10) NOT NULL
6* )

Table created.

CSE510@XE 05-OCT-23>DESC person;

```

Name	Null?	Type
PERSON_ID	NOT NULL	NUMBER
FIRST_NAME	NOT NULL	VARCHAR2(50)
LAST_NAME	NOT NULL	VARCHAR2(50)
PHONE_NO	NOT NULL	NUMBER(10)

3. Insertion of rows using INSERT command:

```

CSE510@XE 05-OCT-23>INSERT INTO person(person_id,first_name,last_name,phone_no) VALUES(1,'Suresh','Krishna',7382790163);
1 row created.

CSE510@XE 05-OCT-23>INSERT INTO person(person_id,first_name,last_name,phone_no) VALUES(2,'JHON','Cristofer',8548585678);
1 row created.

CSE510@XE 05-OCT-23>INSERT INTO person(person_id,first_name,last_name,phone_no) VALUES(3,'RAM','SHANKAR',8548585678);
1 row created.

```

4.

CA Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

CSE510@XE 05-OCT-23>SELECT \* FROM person;

PERSON_ID	FIRST_NAME	LAST_NAME	PHONE_NO
1	Suresh	Krishna	7382790163
2	JHON	Cristofer	8548585678
3	RAM	SHANKAR	8548585678

CSE510@XE 05-OCT-23>SET LIN200

CSE510@XE 05-OCT-23>SELECT \* FROM person;

PERSON_ID	FIRST_NAME	LAST_NAME	PHONE_NO
1	Suresh	Krishna	7382790163
2	JHON	Cristofer	8548585678
3	RAM	SHANKAR	8548585678

5. Creation of table "discounts":

CA Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```

CSE510@XE 05-OCT-23>run
1 CREATE TABLE discounts(
2 discount_id NUMBER PRIMARY KEY,
3 discount_name VARCHAR2(50) NOT NULL,
4 offer NUMBER(4,2),
5 start_date DATE NOT NULL,
6 end_date DATE NOT NULL,
7 check(end_date>start_date)
8* )

```

Table created.

CSE510@XE 05-OCT-23>DESC discounts

Name	Null?	Type
DISCOUNT_ID	NOT NULL	NUMBER
DISCOUNT_NAME	NOT NULL	VARCHAR2(50)
OFFER		NUMBER(4,2)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE

6. INERT rows using INSERT command:

CA Select Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

CSE510@XE 05-OCT-23>INSERT INTO discounts VALUES (1,'Diwali sales',15.5,DATE'2023-11-10',DATE'2023-11-15');

1 row created.

CSE510@XE 05-OCT-23>INSERT INTO discounts VALUES (2,'NEW YEAR sales',12,DATE'2023-12-28',DATE'2024-01-05');

1 row created.

CA Select Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

CSE510@XE 05-OCT-23>SELECT \* FROM discounts;

DISCOUNT_ID	DISCOUNT_NAME	OFFER	START_DAT	END_DATE
1	Diwali sales	15.5	10-NOV-23	15-NOV-23
2	NEW YEAR sales	12	28-DEC-23	05-JAN-24

## 7. "ORIGINAL\_BILL" table creation:

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```
CSE510@XE 05-OCT-23>CREATE TABLE original_bill(
2  product_no NUMBER PRIMARY KEY,
3  product_name VARCHAR2(110) NOT NULL,
4  quantity NUMBER NOT NULL,
5  cost NUMBER(10,2) NOT NULL
6 );
```

Table created.

```
CSE510@XE 05-OCT-23>DESC original_bill;
```

Name	Null?	Type
PRODUCT_NO	NOT NULL	NUMBER
PRODUCT_NAME	NOT NULL	VARCHAR2(110)
QUANTITY	NOT NULL	NUMBER
COST	NOT NULL	NUMBER(10,2)

## 8. Insertion of multiple rows using INSERT ALL statement:

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```
CSE510@XE 05-OCT-23>INSERT ALL
2  INTO original_bill(product_no,product_name,quantity,cost) VALUES(1,'ABC',2,29.9)
3  INTO original_bill(product_no,product_name,quantity,cost) VALUES(2,'AB34',5,89)
4  INTO original_bill(product_no,product_name,quantity,cost) VALUES(3,'JK98',1,75)
5  INTO original_bill(product_no,product_name,quantity,cost) VALUES(4,'KL77',2,99)
6  INTO original_bill(product_no,product_name,quantity,cost) VALUES(5,'NM87',1,50)
7  SELECT * FROM dual;
```

5 rows created.

```
CSE510@XE 05-OCT-23>SELECT * FROM original_bill;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	2	29.9
2	AB34	5	89
3	JK98	1	75
4	KL77	2	99
5	NM87	1	50

## 9. "copy\_bill" table creation:

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```
CSE510@XE 05-OCT-23>CREATE TABLE copy_bill(
2  product_no NUMBER PRIMARY KEY,
3  product_name VARCHAR2(50) NOT NULL,
4  quantity NUMBER NOT NULL,
5  cost NUMBER(10,2) NOT NULL
6 );
```

Table created.

```
CSE510@XE 05-OCT-23>INSERT INTO copy_bill
2  SELECT * FROM original_bill;
```

5 rows created.

```
CSE510@XE 05-OCT-23>DESC copy_bill;
```

Name	Null?	Type
PRODUCT_NO	NOT NULL	NUMBER
PRODUCT_NAME	NOT NULL	VARCHAR2(50)
QUANTITY	NOT NULL	NUMBER
COST	NOT NULL	NUMBER(10,2)

## 10. Using INSERT INTO SELECT COMMAND :

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```
CSE510@XE 05-OCT-23>INSERT INTO copy_bill
2  SELECT * FROM original_bill;
```

5 rows created.

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	2	29.9
2	AB34	5	89
3	JK98	1	75
4	KL77	2	99
5	NM87	1	50

#### 11. Updating one row using UPDATE command:

```
C:\ Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1
```

```
COST
```

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	2	29.9
2	AB34	5	89
3	JK98	1	75
4	KL77	2	99
5	NM87	1	50

```
CSE510@XE 05-OCT-23>UPDATE copy_bill
```

```
2 SET cost=70
```

```
3 WHERE product_no=1;
```

```
1 row updated.
```

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill WHERE product_no=1;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	2	70

#### 12. Updating multiple columns using UPDATE COMMAND:

```
C:\ Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1
```

```
CSE510@XE 05-OCT-23>UPDATE copy_bill
```

```
2 SET quantity=5,COST=33.99
```

```
3 WHERE product_no=1;
```

```
1 row updated.
```

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill WHERE product_no=1;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	5	33.99

## 13. Updating all rows in a table using UPDATE COMMAND:

```
C:\> Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1
```

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	5	33.99
2	AB34	5	89
3	JK98	1	75
4	KL77	2	99
5	NM87	1	50

```
CSE510@XE 05-OCT-23>UPDATE copy_bill SET cost=cost*1.5;
```

```
5 rows updated.
```

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	5	50.99
2	AB34	5	133.5
3	JK98	1	112.5
4	KL77	2	148.5
5	NM87	1	75

## 14. Deleting a row using DELETE command:

```
C:\> Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1
```

```
5 rows updated.
```

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
1	ABC	5	50.99
2	AB34	5	133.5
3	JK98	1	112.5
4	KL77	2	148.5
5	NM87	1	75

```
CSE510@XE 05-OCT-23>DELETE FROM copy_bill WHERE cost=50.99;
```

```
1 row deleted.
```

```
CSE510@XE 05-OCT-23>SELECT * FROM copy_bill;
```

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
2	AB34	5	133.5
3	JK98	1	112.5
4	KL77	2	148.5
5	NM87	1	75



## 15. Deleting multiple rows:

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

CSE510@XE 05-OCT-23>SELECT \* FROM copy\_bill;

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
2	AB34	5	133.5
3	JK98	1	112.5
4	KL77	2	148.5
5	NM87	1	75

CSE510@XE 05-OCT-23>DELETE FROM copy\_bill WHERE cost>120;

2 rows deleted.

CSE510@XE 05-OCT-23>SELECT \* FROM copy\_bill;

PRODUCT_NO	PRODUCT_NAME	QUANTITY	COST
3	JK98	1	112.5
5	NM87	1	75

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

CSE510@XE 05-OCT-23>CREATE TABLE dept(  
 2 dep\_id NUMBER PRIMARY KEY,  
 3 dep\_name VARCHAR2(10) NOT NULL,  
 4 dep\_description VARCHAR2(100) NOT NULL  
 5 );

Table created.

CSE510@XE 05-OCT-23>INSERT ALL  
 2 INTO dept VALUES (01,'CSE','Computer Science')  
 3 INTO dept VALUES (02,'CSM','Machine Learning')  
 4 INTO dept VALUES (03,'CSD','Data Science')  
 5 INTO dept VALUES (04,'ECE','Electronics communication')  
 6 INTO dept VALUES (05,'EEE','Electrial')  
 7 INTO dept VALUES (06,'CE','Civil')  
 8 INTO dept VALUES (07,'ME','Mechanical')  
 9 SELECT \* FROM dual ;

7 rows created.

CSE510@XE 05-OCT-23>SELECT \* FROM dept;

DEP_ID	DEP_NAME	DEP_DESCRIPTION
1	CSE	Computer Science
2	CSM	Machine Learning
3	CSD	Data Science
4	ECE	Electronics communication
5	EEE	Electrial
6	CE	Civil
7	ME	Mechanical

7 rows selected.

CSE510@XE 05-OCT-23>\_

16. Using select statement to retrieve data in a single column:

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```
CSE510@XE 05-OCT-23>SELECT dep_name from dept;

DEP_NAME
-----
CSE
CSM
CSD
ECE
EEE
CE
ME

7 rows selected.
```

17. Using select statement to retrieve data in multiple column:

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```
CSE510@XE 05-OCT-23>SELECT dep_name,dep_description from dept;

DEP_NAME  DEP_DESCRIPTION
-----
CSE       Computer Science
CSM       Machine Learning
CSD       Data Science
ECE       Electronics communication
EEE       Electrial
CE        Civil
ME        Mechanical

7 rows selected.
```

18. Using "SELECT \* FROM " statement to retrieve data in multiple column:

Command Prompt - SQLPLUS cse510/password@0.0.0.0:1521/XEPDB1

```
7 rows selected.

CSE510@XE 05-OCT-23>SELECT * FROM dept;

  DEP_ID DEP_NAME  DEP_DESCRIPTION
-----
      1  CSE      Computer Science
      2  CSM      Machine Learning
      3  CSD      Data Science
      4  ECE      Electronics communication
      5  EEE      Electrial
      6  CE       Civil
      7  ME       Mechanical

7 rows selected.
```

**CONCLUSION:** SQL queries to MANIPULATE TABLES for various databases using DML commands (i.e., INSERT, SELECT, UPDATE, DELETE) is successfully completed.

## EXPERIMENT-3

1.

```
SQL-CSE510>CREATE TABLE students(
2  ID NUMBER(10) PRIMARY KEY,
3  name VARCHAR2(50) ,
4  gender CHAR,
5  mobile_no NUMBER(10),
6  dept VARCHAR2(5)
7 );
```

Table created.

2.

```
SQL-CSE510>DESC students;
```

Name	Null?	Type
ID	NOT NULL	NUMBER(10)
NAME		VARCHAR2(50)
GENDER		CHAR(1)
MOBILE_NO		NUMBER(10)
DEPT		VARCHAR2(5)

3.

```
SQL-CSE510>SELECT * FROM students;
```

ID	NAME	G	MOBILE_NO	DEPT
510	Raju	M	7648982567	CSE
339	Suresh	M	7839265709	CSM
289	Krishna	M	6289106653	EEE
501	Alex	M	9286470178	CSE
145	Harsha	M	7459026841	ECE
505	Aravind	M	8468464937	CSE

6 rows selected.

4.

```
SQL-CSE510>CREATE VIEW std AS SELECT id,name,dept FROM students;
```

View created.

5.

```
SQL-CSE510>CREATE VIEW cse_std AS SELECT id,name,gender,dept FROM students WHERE dept='CSE';
```

View created.

6.

```
SQL-CSE510> SELECT * FROM cse_std;
```

ID	NAME	G	DEPT
510	Raju	M	CSE
501	Alex	M	CSE
505	Aravind	M	CSE

7.

```
SQL-CSE510>SELECT * FROM std;
```

ID	NAME	DEPT
510	Raju	CSE
339	Suresh	CSM
289	Krishna	EEE
501	Alex	CSE
145	Harsha	ECE
505	Aravind	CSE

6 rows selected.

8.

```
SQL-CSE510>INSERT INTO std VALUES (509,'Baba','CSE');
```

1 row created.

9.

```
SQL-CSE510>SELECT * FROM cse_std;
```

ID	NAME	G	DEPT
510	Raju	M	CSE
501	Alex	M	CSE
505	Aravind	M	CSE
509	Baba		CSE

10.

```
SQL-CSE510>UPDATE cse_std SET name='Balaji' WHERE ID=510;
```

1 row updated.

11.

```
SQL-CSE510>SELECT * FROM cse_std;
```

ID	NAME	G	DEPT
510	Balaji	M	CSE
501	Alex	M	CSE
505	Aravind	M	CSE
509	Baba		CSE

12.

```
SQL-CSE510>DELETE FROM cse_std WHERE id=501;
```

```
1 row deleted.
```

13.

```
SQL-CSE510>SELECT * FROM cse_std;
```

ID	NAME	G	DEPT
510	Balaji	M	CSE
505	Aravind	M	CSE
509	Baba		CSE

14.

```
SQL-CSE510>CREATE VIEW eee_std AS SELECT ID,NAME,mobile_no FROM students WHERE dept='EEE';
```

```
View created.
```

15.

```
SQL-CSE510>SELECT * FROM eee_std;
```

ID	NAME	MOBILE_NO
289	Krishna	6289106653

16.

```
SQL-CSE510> DELETE FROM eee_std;
```

```
1 row deleted.
```

17.

```
SQL-CSE510>SELECT * FROM eee_std;
```

```
no rows selected
```

18.

```
SQL-CSE510>DROP VIEW eee_std;  
  
View dropped.  
  
SQL-CSE510>SELECT * FROM eee_std;  
SELECT * FROM eee_std  
                *  
ERROR at line 1:  
ORA-00942: table or view does not exist
```

**CONCLUSION:**

SQL queries to create VIEWS for various databases (i.e., CREATE VIEW, UPDATE VIEW, ALTER VIEW, and DELETE VIEW) is successfully completed.

## EXPERIMENT-4

1.

```
SQL-CSE510>CREATE TABLE instructor(  
2 ins_id NUMBER(10) PRIMARY KEY,  
3 ins_name VARCHAR2(25) NOT NULL,  
4 dep_name VARCHAR2(10) NOT NULL,  
5 salary NUMBER(10,0)  
6 );
```

Table created.

```
SQL-CSE510>CREATE TABLE department(  
2 dep_id NUMBER(10) PRIMARY KEY,  
3 dep_name VARCHAR2(10) NOT NULL,  
4 building VARCHAR2(10) NOT NULL,  
5 budget NUMBER(10)  
6 );
```

Table created.

2.

```
SQL-CSE510>INSERT ALL  
2 INTO instructor VALUES (1,'Suresh','cse',40000)  
3 INTO instructor VALUES (2,'Mahesh','csd',37000)  
4 INTO instructor VALUES (3,'Aravind','csm',20000)  
5 INTO instructor VALUES (4,'Jagadeesh','cse',50000)  
6 INTO instructor VALUES (5,'Raju','physics',20000)  
7 INTO instructor VALUES (6,'Somesh','EEE',30000)  
8 INTO instructor VALUES (7,'Ravi','civil',35000)  
9 INTO department VALUES (1,'cse','gandhi',3500000)  
10 INTO department VALUES (2,'csm','b_block',1000000)  
11 INTO department VALUES (3,'ECE','d_block',1500000)  
12 INTO department VALUES (4,'EEE','c_block',2000000)  
13 SELECT * FROM dual;
```

11 rows created.

3.

```
SQL-CSE510>SELECT * FROM INSTRUCTOR;
```

INS_ID	INS_NAME	DEP_NAME	SALARY
1	Suresh	cse	40000
2	Mahesh	csd	37000
3	Aravind	csm	20000
4	Jagadeesh	cse	50000
5	Raju	physics	20000
6	Somesh	EEE	30000
7	Ravi	civil	35000

```
7 rows selected.
```

4.

```
SQL-CSE510>SELECT * FROM department;
```

DEP_ID	DEP_NAME	BUILDING	BUDGET
1	cse	gandhi	3500000
2	csm	b_block	1000000
3	ECE	d_block	1500000
4	EEE	c_block	2000000

5.

```
SQL-CSE510>SELECT dep_name FROM instructor
2 UNION
3 SELECT dep_name FROM department;
```

```
DEP_NAME
-----
cse
csd
csm
physics
EEE
civil
ECE
```

```
7 rows selected.
```



6.

```
SQL-CSE510>SELECT dep_name FROM instructor
2  UNION ALL
3  SELECT dep_name FROM department;
```

```
DEP_NAME
-----
```

```
cse
csd
csm
cse
physics
EEE
civil
cse
csm
ECE
EEE
```

```
11 rows selected.
```

7.

```
SQL-CSE510>SELECT dep_name FROM instructor
2  INTERSECT
3  SELECT dep_name FROM department;
```

```
DEP_NAME
-----
```

```
cse
csm
EEE
```

8.

```
SQL-CSE510>SELECT dep_name FROM instructor
2  MINUS
3  SELECT dep_name FROM department;
```

```
DEP_NAME
-----
```

```
csd
physics
civil
```

9.

```
SQL-CSE510> SELECT dep_name FROM department
2 MINUS
3 SELECT dep_name FROM instructor;
```

```
DEP_NAME
-----
ECE
```

10.

```
SQL-CSE510>SELECT i.ins_name,d.dep_name,d.budget FROM instructor i,department d;
```

INS_NAME	DEP_NAME	BUDGET
Suresh	cse	3500000
Mahesh	cse	3500000
Aravind	cse	3500000
Jagadeesh	cse	3500000
Raju	cse	3500000
Somesh	cse	3500000
Ravi	cse	3500000
Suresh	csm	1000000
Mahesh	csm	1000000
Aravind	csm	1000000
Jagadeesh	csm	1000000

INS_NAME	DEP_NAME	BUDGET
Raju	csm	1000000
Somesh	csm	1000000
Ravi	csm	1000000
Suresh	ECE	1500000
Mahesh	ECE	1500000
Aravind	ECE	1500000
Jagadeesh	ECE	1500000
Raju	ECE	1500000
Somesh	ECE	1500000
Ravi	ECE	1500000
Suresh	EEE	2000000

INS_NAME	DEP_NAME	BUDGET
Mahesh	EEE	2000000
Aravind	EEE	2000000
Jagadeesh	EEE	2000000
Raju	EEE	2000000
Somesh	EEE	2000000
Ravi	EEE	2000000

```
28 rows selected.
```

11.

```
SQL-CSE510>SELECT i.ins_name,d.dep_name,d.budget FROM instructor i CROSS JOIN department d;
```

INS_NAME	DEP_NAME	BUDGET
Suresh	cse	3500000
Mahesh	cse	3500000
Aravind	cse	3500000
Jagadeesh	cse	3500000
Raju	cse	3500000
Somesh	cse	3500000
Ravi	cse	3500000
Suresh	csm	1000000
Mahesh	csm	1000000
Aravind	csm	1000000
Jagadeesh	csm	1000000

INS_NAME	DEP_NAME	BUDGET
Raju	csm	1000000
Somesh	csm	1000000
Ravi	csm	1000000
Suresh	ECE	1500000
Mahesh	ECE	1500000
Aravind	ECE	1500000
Jagadeesh	ECE	1500000
Raju	ECE	1500000
Somesh	ECE	1500000
Ravi	ECE	1500000
Suresh	EEE	2000000

INS_NAME	DEP_NAME	BUDGET
Mahesh	EEE	2000000
Aravind	EEE	2000000
Jagadeesh	EEE	2000000
Raju	EEE	2000000
Somesh	EEE	2000000
Ravi	EEE	2000000

28 rows selected.

12.

```
SQL-CSE510>SELECT i.ins_name,dep_name,d.budget FROM instructor i NATURAL JOIN department d;
```

INS_NAME	DEP_NAME	BUDGET
Suresh	cse	3500000
Aravind	csm	1000000
Jagadeesh	cse	3500000
Somesh	EEE	2000000

#### CONCLUSION:

SQL queries to perform RELATIONAL SET OPERATIONS (i.e., UNION, UNION ALL, INTERSECT, MINUS, CROSS JOIN, NATURAL JOIN) is successfully completed.

## EXPERIMENT-5

1.

```
SQL-CSE510>CREATE TABLE instructors(  
2 id NUMBER PRIMARY KEY,  
3 name VARCHAR2(50) NOT NULL,  
4 salary NUMBER  
5 );
```

Table created.

```
SQL-CSE510>CREATE TABLE departments(  
2 id NUMBER PRIMARY KEY,  
3 dept_name VARCHAR2(50)  
4 );
```

Table created.

2.

```
1 INSERT ALL  
2 INTO instructors VALUES (1,'Ram',70000)  
3 INTO instructors VALUES (2,'Sham',null)  
4 INTO instructors VALUES (3,'Venkat',30000)  
5 INTO departments VALUES (1,'CSE')  
6 INTO departments VALUES (2,'EEE')  
7 INTO departments VALUES (3,'CSM')  
8* SELECT * FROM dual  
SQL-CSE510>/
```

6 rows created.

3.

```
SQL-CSE510>SELECT * FROM instructors;
```

ID	NAME	SALARY
1	Ram	70000
2	Sham	
3	Venkat	30000

4.

```
SQL-CSE510>SELECT * FROM department;
```

DEP_ID	DEP_NAME	BUILDING	BUDGET
1	cse	gandhi	3500000
2	csn	b_block	1000000
3	ECE	d_block	1500000
4	EEE	c_block	2000000

5.

```
SQL-CSE510>SELECT * FROM instructors
2 WHERE
3 salary IS NULL;
```

ID	NAME	SALARY
2	Sham	

6.

```
SQL-CSE510>SELECT * FROM instructors
2 WHERE
3 salary BETWEEN 10000 AND 80000;
```

ID	NAME	SALARY
1	Ram	70000
3	Venkat	30000

7.

```
SQL-CSE510>SELECT * FROM instructors
2 WHERE
3 name LIKE 'R%';
```

ID	NAME	SALARY
1	Ram	70000

8.

```
SQL-CSE510>SELECT * FROM instructors
2  WHERE
3  name LIKE '___';
```

ID	NAME	SALARY
1	Ram	70000

9.

```
SQL-CSE510>SELECT * FROM instructors
2  WHERE
3  salary IN(10000,30000,20000);
```

ID	NAME	SALARY
3	Venkat	30000

10.

```
SQL-CSE510>SELECT * FROM instructors
2  WHERE
3  EXISTS(SELECT * FROM departments WHERE instructors.id=departments.id);
```

ID	NAME	SALARY
1	Ram	70000
2	Sham	
3	Venkat	30000

CONCLUSION:

SQL queries to perform SPECIAL OPERATIONS (i.e. ISNULL, BETWEEN, LIKE, IN, EXISTS) is successfully completed

## EXPERIMENT-6

1.

```
SQL-CSE510>CREATE TABLE student(  
2  roll_no NUMBER PRIMARY KEY,  
3  name VARCHAR2(50) NOT NULL,  
4  dept_name VARCHAR2(10) NOT NULL  
5  );
```

Table created.

```
SQL-CSE510>CREATE TABLE blocks(  
2  dept_name VARCHAR2(10) PRIMARY KEY,  
3  block_name VARCHAR2(20) NOT NULL  
4  );
```

Table created.

2.

```
1  INSERT ALL  
2  INTO student VALUES (505,'Aravind','CSE')  
3  INTO student VALUES (411,'Rani','EEE')  
4  INTO student VALUES (310,'Raju','ECE')  
5  INTO student VALUES (509,'Baba','CSM')  
6  INTO blocks VALUES ('CSE','C-BLOCK')  
7  INTO blocks VALUES ('CSM','B-BLOCK')  
8  INTO blocks VALUES ('EEE','A-BLOCK')  
9* SELECT * FROM dual  
SQL-CSE510>/
```

7 rows created.

3.

```
SQL-CSE510>SELECT * FROM student;
```

ROLL_NO	NAME	DEPT_NAME
505	Aravind	CSE
411	Rani	EEE
310	Raju	ECE
509	Baba	CSM

4.

```
SQL-CSE510>SELECT * FROM blocks;
```

DEPT_NAME	BOLCK_NAME
CSE	C-BLOCK
CSM	B-BLOCK
EEE	A-BLOCK

5.

```
SQL-CSE510>SELECT * FROM student
2 JOIN blocks ON
3 student.dept_name=blocks.dept_name;
```

ROLL_NO	NAME	DEPT_NAME	DEPT_NAME	BOLCK_NAME
505	Aravind	CSE	CSE	C-BLOCK
411	Rani	EEE	EEE	A-BLOCK
509	Baba	CSM	CSM	B-BLOCK

6.

```
SQL-CSE510>SELECT * FROM student JOIN blocks
2 USING(dept_name);
```

DEPT_NAME	ROLL_NO	NAME	BOLCK_NAME
CSE	505	Aravind	C-BLOCK
EEE	411	Rani	A-BLOCK
CSM	509	Baba	B-BLOCK

7.

```
SQL-CSE510>SELECT * FROM student
2 LEFT OUTER JOIN blocks ON
3 student.dept_name=blocks.dept_name;
```

ROLL_NO	NAME	DEPT_NAME	DEPT_NAME	BOLCK_NAME
505	Aravind	CSE	CSE	C-BLOCK
411	Rani	EEE	EEE	A-BLOCK
310	Raju	ECE		
509	Baba	CSM	CSM	B-BLOCK

8.

```
SQL-CSE510>SELECT * FROM student
2 RIGHT OUTER JOIN blocks ON
3 student.dept_name=blocks.dept_name;
```

ROLL_NO	NAME	DEPT_NAME	DEPT_NAME	BOLCK_NAME
505	Aravind	CSE	CSE	C-BLOCK
411	Rani	EEE	EEE	A-BLOCK
509	Baba	CSM	CSM	B-BLOCK



9.

```
SQL-CSE510>SELECT * FROM student
2  FULL OUTER JOIN blocks
3  ON student.dept_name=blocks.dept_name;
```

ROLL_NO	NAME	DEPT_NAME	DEPT_NAME	BLOCK_NAME
505	Aravind	CSE	CSE	C-BLOCK
411	Rani	EEE	EEE	A-BLOCK
310	Raju	ECE		
509	Baba	CSM	CSM	B-BLOCK

CONCLUSION:

SQL queries to perform JOIN OPERATIONS (i.e., CONDITIONAL JOIN, EQUI JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN) is successfully completed.

## EXPERIMENT-7

1.

```
SQL-CSE510>CREATE TABLE employee(  
2 ID NUMBER PRIMARY KEY,  
3 name VARCHAR2(50) NOT NULL,  
4 gender CHAR NOT NULL,  
5 salary NUMBER(10,2) NOT NULL  
6 );
```

Table created.

2.

```
SQL-CSE510>INSERT ALL  
2 INTO employee VALUES (1,'RAJU','M',90000)  
3 INTO employee VALUES (2,'Balaji','M',95000)  
4 INTO employee VALUES (3,'Aravind','M',80000)  
5 INTO employee VALUES (4,'Abhilash','M',100000)  
6 INTO employee VALUES (5,'Rani','F',85000)  
7 INTO employee VALUES (6,'Pinky','F',85000)  
8 SELECT * FROM dual;
```

6 rows created.

3.

```
SQL-CSE510>SELECT * FROM employee;
```

ID	NAME	G	SALARY
1	RAJU	M	90000
2	Balaji	M	95000
3	Aravind	M	80000
4	Abhilash	M	100000
5	Rani	F	85000
6	Pinky	F	85000

6 rows selected.

4.

```
SQL-CSE510>SELECT SUM(salary) FROM employee;
```

```
SUM(SALARY)  
-----  
535000
```

5.

```
SQL-CSE510>SELECT AVG(salary) FROM employee;

AVG(SALARY)
-----
89166.6667
```

6.

```
SQL-CSE510>SELECT COUNT(salary) FROM employee;

COUNT(SALARY)
-----
6
```

7.

```
SQL-CSE510>SELECT MIN(salary) FROM employee;

MIN(SALARY)
-----
80000
```

8.

```
SQL-CSE510>SELECT MAX(salary) FROM employee;

MAX(SALARY)
-----
100000
```

CONCLUSION:

SQL queries to perform AGGREGATE OPERATIONS (i.e., SUM, COUNT, AVG, MIN, MAX) is successfully completed.

## EXPERIMENT-8

1.

```
SQL-CSE510>CREATE TABLE names(  
2  first_name VARCHAR2(30  
3  ) NOT NULL,  
4  LAST_name VARCHAR2(30) NOT NULL  
5  );  
  
Table created.
```

2.

```
1  INSERT ALL  
2  INTO names VALUES ('Antony','Robert')  
3  INTO names VALUES ('Mark','Antony')  
4  INTO names VALUES ('Stuart','Smart')  
5  INTO names VALUES ('Rakesh','k')  
6* select * from dual  
SQL-CSE510>/  
  
4 rows created.
```

3.

```
SQL-CSE510>SELECT LOWER(first_name) FROM names;  
  
LOWER(FIRST_NAME)  
-----  
antony  
mark  
stuart  
rakesh
```

4.

```
SQL-CSE510>SELECT UPPER(first_name) FROM names;  
  
UPPER(FIRST_NAME)  
-----  
ANTONY  
MARK  
STUART  
RAKESH
```

5.

```
SQL-CSE510>SELECT INITCAP(first_name) FROM names;

INITCAP(FIRST_NAME)
-----
Antony
Mark
Stuart
Rakesh
```

6.

```
SQL-CSE510>SELECT CONCAT(first_name,last_name) FROM names;

CONCAT(FIRST_NAME, LAST_NAME)
-----
AntonyRobert
MarkAntony
StuartSmart
Rakeshk
```

7.

```
SQL-CSE510>SELECT SUBSTR(first_name,1,4) FROM names;

SUBSTR(FIRST_NAM
-----
Anto
Mark
Stua
Rake
```

8.

```
SQL-CSE510>SELECT LENGTH(first_name) FROM names;

LENGTH(FIRST_NAME)
-----
6
4
6
6
```

9.

```
SQL-CSE510>SELECT INSTR(first_name,'Ma') FROM names;

INSTR(FIRST_NAME,'MA')
-----
                0
                1
                0
                0
```

10.

```
SQL-CSE510>SELECT TRIM(' ' FROM first_name) FROM names;

TRIM(' 'FROMFIRST_NAME)
-----
Antony
Mark
Stuart
Rakesh
```

11.

```
SQL-CSE510>SELECT ROUND(11.111,2) FROM dual;

ROUND(11.111,2)
-----
          11.11
```

12.

```
SQL-CSE510>SELECT MOD(11,2) FROM dual;

MOD(11,2)
-----
          1
```

13.

```
SQL-CSE510>SELECT SYSDATE FROM dual;

SYSDATE
-----
08-DEC-23
```

14.

```
SQL-CSE510>SELECT MONTHS_BETWEEN(SYSDATE,'08-DEC-2024') FROM dual;  
MONTHS_BETWEEN(SYSDATE,'08-DEC-2024')  
-----  
-12
```

15.

```
SQL-CSE510>SELECT ADD_MONTHS(SYSDATE,12) FROM dual;  
ADD_MONTH  
-----  
08-DEC-24
```

16.

```
SQL-CSE510>SELECT NEXT_DAY(SYSDATE,'MONDAY') FROM dual;  
NEXT_DAY(  
-----  
11-DEC-23
```

17.

```
SQL-CSE510>SELECT LAST_DAY(SYSDATE) FROM dual;  
LAST_DAY(  
-----  
31-DEC-23
```

18.

```
SQL-CSE510>SELECT CURRENT_TIMESTAMP(3) FROM dual;  
CURRENT_TIMESTAMP(3)  
-----  
08-DEC-23 11.11.14.747 PM +05:30
```

CONCLUSION:

SQL queries to perform ORACLE BUILT-IN FUNCTIONS (i.e., DATE, TIME) is successfully completed.

## EXPERIMENT-9

1.

```
SQL_CSE-510>CREATE TABLE stud(  
2 ID NUMBER PRIMARY KEY,  
3 first_name VARCHAR2  
4 (25) NOT NULL,  
5 last_name VARCHAR2(25) NOT NULL  
6 );
```

Table created.

2.

```
SQL_CSE-510>INSERT INTO stud VALUES (111,'ROBERT','JUNIOR');
```

1 row created.

```
SQL_CSE-510>INSERT INTO stud VALUES (111,'HARRY','HARRY');  
INSERT INTO stud VALUES (111,'HARRY','HARRY')
```

\*

ERROR at line 1:

ORA-00001: unique constraint (C##510.SYS\_C008365) violated

3.

```
1 CREATE TABLE orders(  
2 id NUMBER PRIMARY KEY,  
3 order_num NUMBER NOT NULL,  
4 stud_id NUMBER REFERENCES stud(id)  
5* )
```

```
SQL_CSE-510>/
```

Table created.

4.

```
SQL_CSE-510>INSERT INTO orders VALUES (11,2,111);
```

1 row created.

```
SQL_CSE-510>INSERT INTO orders VALUES (2011,7,112);  
INSERT INTO orders VALUES (2011,7,112)
```

\*

ERROR at line 1:

ORA-02291: integrity constraint (C##510.SYS\_C008368) violated - parent key not found



5.

```
1 CREATE TABLE employees(  
2   id NUMBER PRIMARY KEY,  
3   name VARCHAR2(50) NOT NULL,  
4   e_mail VARCHAR2(50) UNIQUE  
5* )  
SQL_CSE-510>/  
  
Table created.
```

6.

```
SQL_CSE-510>INSERT INTO employees VALUES (501,'Ramesh','Ramesh510@gmail.com');  
  
1 row created.  
  
SQL_CSE-510>INSERT INTO employees VALUES (1505,'RAMESH K','Ramesh510@gmail.com');  
INSERT INTO employees VALUES (1505,'RAMESH K','Ramesh510@gmail.com')  
*  
ERROR at line 1:  
ORA-00001: unique constraint (C##510.SYS_C008371) violated
```

7.

```
1 CREATE TABLE order1(  
2   id NUMBER PRIMARY KEY,  
3   product_name VARCHAR2(50) NOT NULL,  
4   quantity NUMBER  
5* )  
SQL_CSE-510>/  
  
Table created.
```

8.

```
SQL_CSE-510>INSERT INTO order1 VALUES (1,'ABCD',98);  
  
1 row created.  
  
SQL_CSE-510>INSERT INTO order1 VALUES (4,'',98);  
INSERT INTO order1 VALUES (4,'',98)  
*  
ERROR at line 1:  
ORA-01400: cannot insert NULL into ("C##510"."ORDER1"."PRODUCT_NAME")
```

9.

```
SQL_CSE-510>CREATE TABLE parts1(  
2  part_id NUMBER PRIMARY KEY,  
3  part_name VARCHAR2(50) NOT NULL,  
4  buy_price NUMBER(9,2) CHECK(buy_price>0)  
5  );
```

Table created.

10.

```
SQL_CSE-510>INSERT INTO parts1 VALUES (1,'ABCD',788);  
1 row created.  
  
SQL_CSE-510>INSERT INTO parts1 VALUES (2,'ABD',-788);  
INSERT INTO parts1 VALUES (2,'ABD',-788)  
*  
ERROR at line 1:  
ORA-02290: check constraint (C##510.SYS_C008375) violated
```

11.

```
1 CREATE TABLE customers1(  
2  name VARCHAR2(50) NOT NULL,  
3  id NUMBER PRIMARY KEY,  
4  country VARCHAR2(20) DEFAULT 'IND'  
5* )  
SQL_CSE-510>/  
  
Table created.
```

12.

```
SQL_CSE-510>INSERT INTO customers1(name,id,country) VALUES ('Ram',1,'AUS');  
1 row created.  
  
SQL_CSE-510>INSERT INTO customers1(name,id) VALUES ('Raju',2);  
1 row created.
```

13.

```
SQL_CSE-510>SELECT * FROM customers1;
```

NAME	ID	COUNTRY
Ram	1	AUS
Raju	2	IND

#### CONCLUSION:

SQL queries to perform KEY CONSTRAINTS (i.e., PRIMARY KEY, FOREIGN KEY, UNIQUE NOT NULL, CHECK, DEFAULT) is successfully completed.

## EXPERIMENT-10

1.

```
SQL_CSE-510>SET SERVEROUT ON
SQL_CSE-510>SET VERIFY OFF
```

2.

```
SQL_CSE-510>DECLARE
  2  n NUMBER;
  3  fac NUMBER:=1;
  4  n1 NUMBER;
  5  BEGIN
  6  n:=&n;
  7  n1:=n;
  8  WHILE n1>0 LOOP
  9  fac := n1*fac;
 10  n1:=n1-1;
 11  END LOOP;
 12  DBMS_OUTPUT.PUT_LINE('The Factorial of '||n||' is '||fac);
 13  END;
 14  /
Enter value for n: 3
The Factorial of 3 is 6
```

3.

```
SQL_CSE-510>/
Enter value for n: 5
The Factorial of 5 is 120

PL/SQL procedure successfully completed.

SQL_CSE-510>/
Enter value for n: 99
The Factorial of 99 is ~

PL/SQL procedure successfully completed.
```

CONCLUSION:

PL/SQL program for calculating the factorial of a given number is successfully completed.

## EXPERIMENT-11

1.

```
SQL_CSE-510>SET SERVEROUT ON
SQL_CSE-510>SET VERIFY OFF
```

2.

```
SQL_CSE-510>DECLARE
  2  n NUMBER;
  3  flag NUMBER:=1;
  4  g NUMBER;
  5  g1 NUMBER;
  6  BEGIN
  7  n:=&n;
  8  g1:=n;
  9  g:=2;
 10  FOR g IN 2..g1/2
 11  LOOP
 12  IF mod(n,g) = 0
 13  THEN
 14  flag:=0;
 15  EXIT;
 16  END IF;
 17  END LOOP;
 18  IF flag=1
 19  THEN
 20  DBMS_OUTPUT.PUT_LINE(g1||' is a prime number');
 21  ELSE
 22  DBMS_OUTPUT.PUT_LINE(g1||' is not a prime number');
 23  END IF;
 24  END;
 25  /
Enter value for n: 9
9 is not a prime number

PL/SQL procedure successfully completed.
```

3.

```
SQL_CSE-510>/  
Enter value for n: 8  
8 is not a prime number  
  
PL/SQL procedure successfully completed.  
  
SQL_CSE-510>/  
Enter value for n: 7  
7 is a prime number  
  
PL/SQL procedure successfully completed.
```

CONCLUSION:

PL/SQL program for finding the given number is prime number or not is successfully completed.

## EXPERIMENT-12

1.

```
SQL_CSE-510>SET SERVEROUT ON  
SQL_CSE-510>SET VERIFY OFF
```

2.

```
SQL_CSE-510>DECLARE  
  2  first_num NUMBER:=0;  
  3  second_num NUMBER:=1;  
  4  n NUMBER;  
  5  i NUMBER;  
  6  temp NUMBER;  
  7  BEGIN  
  8  n:=&n;  
  9  DBMS_OUTPUT.PUT_LINE('SERIES :');  
10  DBMS_OUTPUT.PUT_LINE(first_num);  
11  DBMS_OUTPUT.PUT_LINE(second_num);  
12  FOR i IN 2..N  
13  LOOP  
14  temp := first_num+second_num;  
15  first_num := second_num;  
16  second_num := temp;  
17  DBMS_OUTPUT.PUT_LINE(temp);  
18  END LOOP;  
19  END;  
20  /  
Enter value for n: 4  
SERIES :  
0  
1  
1  
2  
3  
  
PL/SQL procedure successfully completed.
```

3.

```
SQL_CSE-510>/
Enter value for n: 3
SERIES :
0
1
1
2

PL/SQL procedure successfully completed.

SQL_CSE-510>/
Enter value for n: 5
SERIES :
0
1
1
2
3
5

PL/SQL procedure successfully completed.
```

CONCLUSION:

PL/SQL program for displaying the Fibonacci series up to an integer is successfully completed.

## EXPERIMENT-13

1.

```
SQL_CSE-510>CREATE TABLE sailor1(  
2 id NUMBER PRIMARY KEY,  
3 name VARCHAR2(50) NOT NULL  
4 );
```

Table created.

2.

```
SQL_CSE-510>CREATE OR REPLACE PROCEDURE insertuser(id IN NUMBER,name IN VARCHAR2)  
2 AS  
3 BEGIN  
4 INSERT INTO sailor1 VALUES(id,name);  
5 DBMS_OUTPUT.PUT_LINE('Record inserted successfully');  
6 END;  
7 /
```

Procedure created.

3.

```
1 DECLARE  
2 co NUMBER;  
3 BEGIN  
4 insertuser(11,'RANI');  
5 SELECT COUNT(*) INTO co FROM sailor1;  
6 DBMS_OUTPUT.PUT_LINE(co||' Record is inserted successfully');  
7* END;  
SQL_CSE-510>/
```

PL/SQL procedure successfully completed.

4.

```
1 DECLARE  
2 co NUMBER;  
3 BEGIN  
4 insertuser(10,'Balaji');  
5 SELECT COUNT(*) INTO co FROM sailor1;  
6 DBMS_OUTPUT.PUT_LINE(co||' Record is inserted successfully');  
7* END;  
SQL_CSE-510>/
```

Record inserted successfully  
2 Record is inserted successfully

PL/SQL procedure successfully completed.

CONCLUSION:

PL/SQL program to implement Stored Procedure on table is successfully completed.



## EXPERIMENT-14

1.

```
SQL_CSE-510>CREATE TABLE section(  
2 id NUMBER PRIMARY KEY,  
3 course_name VARCHAR2(20) NOT NULL,  
4 strength NUMBER NOT NULL  
5 );
```

Table created.

2.

```
SQL_CSE-510>INSERT ALL  
2 INTO section VALUES (1,'CSE',50)  
3 INTO section VALUES (2,'CSM',60)  
4 INTO section VALUES (3,'CSD',75)  
5 SELECT * FROM dual;
```

3 rows created.

3.

```
SQL_CSE-510>CREATE OR REPLACE FUNCTION totalstrength RETURN NUMBER  
2 AS  
3 total NUMBER:=0;  
4 BEGIN  
5 SELECT sum(strength) INTO total FROM section;  
6 return total;  
7 END;  
8 /
```

Function created.

4.

```
SQL_CSE-510>DECLARE  
2 answer NUMBER;  
3 BEGIN  
4 answer:=totalstrength();  
5 DBMS_OUTPUT.PUT_LINE('Total strength of students is '||answer);  
6 END;  
7 /
```

Total strength of students is 185

PL/SQL procedure successfully completed.

CONCLUSION:

PL/SQL program to implement Stored Function on table is successfully completed.

## EXPERIMENT-15

1.

```
SQL_CSE-510>CREATE TABLE instruc(  
2 id NUMBER PRIMARY KEY,  
3 name VARCHAR2(50) NOT NULL,  
4 dept_name VARCHAR2(20) NOT NULL,  
5 salary NUMBER(10,2) CHECK(salary>10000)  
6 );
```

Table created.

2.

```
SQL_CSE-510>INSERT ALL  
2 INTO instruc VALUES (1,'Abhi','CSE',50000)  
3 INTO instruc VALUES (2,'Narsimha','CSM',75000)  
4 INTO instruc VALUES (3,'Balaji','CSE',80000)  
5 INTO instruc VALUES (4,'Rani','CSD',47000)  
6 SELECT * FROM dual;
```

4 rows created.

3.

```
SQL_CSE-510>CREATE OR REPLACE TRIGGER display_changes  
2 BEFORE UPDATE ON instruc  
3 FOR EACH ROW  
4 WHEN (NEW.ID = OLD.ID)  
5 DECLARE  
6 sal_diff number;  
7 BEGIN  
8 sal_diff := :NEW.salary - :OLD.salary;  
9 dbms_output.put_line('Old salary: ' || :OLD.salary);  
10 dbms_output.put_line('New salary: ' || :NEW.salary);  
11 dbms_output.put_line('Salary difference: ' || sal_diff);  
12 END;  
13 /
```

Trigger created.

4.

```
SQL_CSE-510>DECLARE
  2  tot_rows NUMBER;
  3  BEGIN
  4  UPDATE instruc
  5  SET salary=salary*1.5;
  6  IF sql%notfound THEN
  7  DBMS_OUTPUT.PUT_LINE('no instructors updated');
  8  ELSIF sql%found THEN
  9  tot_rows:=sql%rowcount;
 10  DBMS_OUTPUT.PUT_LINE(tot_rows||' instructors updated');
 11  END IF;
 12  END;
 13  /
Old salary: 50000
New salary: 75000
Salary difference: 25000
Old salary: 75000
New salary: 112500
Salary difference: 37500
Old salary: 80000
New salary: 120000
Salary difference: 40000
Old salary: 47000
New salary: 70500
Salary difference: 23500
4 instructors updated

PL/SQL procedure successfully completed.
```

CONCLUSION:

PL/SQL program to implement Trigger on table is successfully completed.

## EXPERIMENT-16

1.

```
1 CREATE TABLE customers2(  
2 id NUMBER PRIMARY KEY,  
3 name VARCHAR2(30) NOT NULL,  
4 age NUMBER(3) NOT NULL,  
5 salary NUMBER(10,2) NOT NULL  
6* )  
SQL_CSE-510>/
```

Table created.

2.

```
SQL_CSE-510>DECLARE  
2 tot_rows NUMBER;  
3 BEGIN  
4 UPDATE customers2 SET salary=salary*1.5;  
5 IF sql%notfound THEN  
6 DBMS_OUTPUT.PUT_LINE('No customers updated');  
7 ELSIF sql%found THEN  
8 tot_rows := sql%rowcount;  
9 DBMS_OUTPUT.PUT_LINE(tot_rows||' customers updated');  
10 END IF;  
11 END;  
12 /
```

No customers updated

PL/SQL procedure successfully completed.

3.

```
SQL_CSE-510>INSERT ALL  
2 INTO customers2 VALUES (1,'Bala',22,60000)  
3 INTO customers2 VALUES (2,'Shyam',33,70000)  
4 INTO customers2 VALUES (3,'Charan',23,65000)  
5 INTO customers2 VALUES (4,'Ravi',25,60000)  
6 SELECT * FROM dual;
```

4 rows created.

4.

```
1 DECLARE
2 c_id customers2.id%type;
3 c_name customers2.name%type;
4 c_age customers2.age%type;
5 CURSOR c_customers IS
6 SELECT id,name,age FROM customers2;
7 BEGIN
8 OPEN c_customers;
9 LOOP
10 FETCH c_customers INTO c_id,c_name,c_age;
11 EXIT WHEN c_customers%notfound;
12 DBMS_OUTPUT.PUT_LINE(c_id||' '||c_name||' '||c_age);
13 END LOOP;
14 CLOSE c_customers;
15* END;
SQL_CSE-510>/
1 Bala 22
2 Shyam 33
3 Charan 23
4 Ravi 25

PL/SQL procedure successfully completed.
```

CONCLUSION:

PL/SQL program to implement Cursor on table is successfully completed.