18CSC303J - DATABASE MANAGEMENT SYSTEMS

SEMESTER – VI

2021 – 2022 (EVEN)

Name : Aryan Jalla

Register No : RA1911003010729

Branch : CSE

Section : C2

DEPARTMENT OF COMPUTING TECHNOLOIGIES

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)

S.R.M. NAGAR, KATTANKULATHUR – 603 203
CHENGALPATTU DISTRICT

DEPARTMENT OF COMPUTATING TECHNOLIGIES COLLEGE OF ENGINEERING & TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

S.R.M. NAGAR, KATTANKULATHUR

BONAFIDE CERTIFICATE

Register No.: RA1911003010729

Certified to be the bonafide record of work done by Aryan Jalla of **CSE, B.Tech.** Degree course in the Practical of **18CSC303J - DATABASE MANAGEMENT SYSTEMS** in **SRM IST**, Kattankulathur during the academic year **2021 - 2022**.

Staff In-Charge **Head of the Department**

Date:

Submitted for University Examination held on _____ at **SRM IST**, Kattankulathur.

Date:

Internal Examiner I **Internal Examiner II**

CONTENTS

Experiment 1: SQL BASIC AND DDL COMMANDS

Aim:

To write SQL queries to execute basic SQL commands.

Code:

```
SQL> CREATE TABLE books(bookid NUMBER, name VARCHAR2(100), authorid NUMBER);

Table created.

SQL> INSERT INTO books VALUES(2, 'The Near East', 9);

1 row created.

SQL> INSERT INTO books VALUES(4, 'The Freakshow Murders', 2);

1 row created.

SQL> INSERT INTO books VALUES(6, 'Hard Times', 10);

1 row created.

SQL> SELECT * FROM books;

    BOOKID
----------
NAME
--------------------------------------------------------------------------------
  AUTHORID
----------
         2
The Near East
         9


         4
The Freakshow Murders
         2


    BOOKID
----------
NAME
--------------------------------------------------------------------------------
```

```
  AUTHORID
----------


         6
Hard Times
        10
SQL> SELECT name, authorid FROM books;
NAME
--------------------------------------------------------------------------------
  AUTHORID
----------
The Near East
         9


The Freakshow Murders
         2


Hard Times
        10



SQL> UPDATE books SET authorid = 11 WHERE bookid = 6;
1 row updated.
SQL> SELECT * FROM books;
    BOOKID
----------
NAME
--------------------------------------------------------------------------------
  AUTHORID
----------
         2
The Near East
         9
```

```
         4
The Freakshow Murders
         2


    BOOKID
----------
NAME
--------------------------------------------------------------------------------
  AUTHORID
----------


         6
Hard Times
        11
SQL> DELETE FROM books WHERE authorid = 11;
1 row deleted.
SQL> SELECT * FROM books;
    BOOKID
----------
NAME
--------------------------------------------------------------------------------
  AUTHORID
----------
         2
The Near East
         9


         4
The Freakshow Murders
         2


    BOOKID
----------
NAME
```

```
--------------------------------------------------------------------------------

 AUTHORID

----------
```

```
SQL> ALTER TABLE books ADD pagecount NUMBER;

Table altered.

SQL> DESC books;
 Name                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOKID                                    NUMBER
 NAME                                      VARCHAR2(100)
 AUTHORID                                  NUMBER
 PAGECOUNT                                 NUMBER

SQL> ALTER TABLE books MODIFY (name VARCHAR2(200));

Table altered.

SQL> DESC books;
 Name                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOKID                                    NUMBER
 NAME                                      VARCHAR2(200)
 AUTHORID                                  NUMBER
 PAGECOUNT                                 NUMBER

SQL> ALTER TABLE books DROP (pagecount);

Table altered.

SQL> DESC books;
 Name                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOKID                                    NUMBER
 NAME                                      VARCHAR2(200)
 AUTHORID                                  NUMBER
```

```
SQL> ALTER TABLE books RENAME TO books1;

Table altered.

SQL> DESC books1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOKID                                    NUMBER
 NAME                                      VARCHAR2(200)
 AUTHORID                                  NUMBER

SQL> DROP TABLE books1;

Table dropped.

SQL> DESC books1;

ERROR:

ORA-04043: object books1 does not exist

SQL> spool off
```

Result: Thus the basic SQL queries were successfully executed and verified

Experiment 2: SQL DML COMMANDS

Aim:

To write the SQL queries using DDL Commands with and without constraints.

Code:

SQL> CREATE TABLE customers(customer_id INTEGER, sale_date DATE, sale_amount INTEGER, salesperson VARCHAR(200), order_id VARCHAR(200));

Table created.

SQL> SELECT * FROM customers;

no rows selected

SQL> INSERT INTO customers VALUES(1001, TO_DATE('2020-05-23','YYYY-MM-DD'), 1200, 'Raj K', 1001);

1 row created.

SQL> INSERT INTO customers VALUES(1001, TO_DATE('2020-05-22','YYYY-MM-DD'), 1200, 'MK', 1002);

1 row created.

SQL> INSERT INTO customers VALUES(1002, TO_DATE('2020-05-23','YYYY-MM-DD'), 1200, 'Malika Rakesh', 1003);

1 row created.

SQL> INSERT INTO customers VALUES(1003, TO_DATE('2020-05-22','YYYY-MM-DD'), 1200, 'Malika Rakesh', 1004);

1 row created

SQL> SELECT * FROM customers;

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1001 23-MAY-20 1200

Raj K

1001

1001 22-MAY-20 1200

MK

1002

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1002 23-MAY-20 1200

Malika Rakesh

1003

1003 22-MAY-20 1200

Malika Rakesh

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1004

SQL> UPDATE customers SET salesperson = 'Mallika Rakesh' WHERE customer_id = 1002

1 row updated.

SQL> UPDATE customers SET salesperson = 'Mallika Rakesh' WHERE customer_id = 1003;

1 row updated.

SQL> DELETE FROM customers WHERE salesperson = 'Mallika Rakesh';

2 rows deleted.

SQL> SELECT * FROM customers;

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1001 23-MAY-20 1200

Raj K

1001

1001 22-MAY-20 1200

MK

1002

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

SQL> DESC customers;

Name	Null?	Type
CUSTOMER_ID		NUMBER(38)
SALE_DATE		DATE
SALE_AMOUNT		NUMBER(38)
SALESPERSON		VARCHAR2(200)
ORDER_ID		VARCHAR2(200)

SQL> spool off

Result:

Thus the SQL queries using DDL Commands with and without constraints were successfully executed and verified.

Experiment 3: SQL DCL & TCL COMMANDS

Aim: :- To write SQL queries to execute different DCL and TCL commands.

Code:

```
SQL> CREATE TABLE customers(customer_id INTEGER, sale_date DATE, sale_amount INTEGER,
salesperson VARCHAR(200), order_id VARCHAR(200));

Table created.

SQL> INSERT INTO customers VALUES(1006, TO_DATE('2020-03-04', 'YYYY-MM-DD'), 3200, 'DL', '1008');

1 row created.

SQL> GRANT SELECT, INSERT ON customers TO   RA1911003010742;

Grant succeeded.

SQL> GRANT SELECT, INSERT ON customers TO   RA1911003010742;

Grant succeeded.

SQL> GRANT ALL ON customers TO   RA1911003010742;

Grant succeeded.

SQL> REVOKE ALL ON customers FROM   RA1911003010742;

Revoke succeeded.

SQL> GRANT SELECT ON customers TO public;

Grant succeeded.

SQL> REVOKE SELECT ON customers FROM public;

Revoke succeeded.

SQL> REVOKE SELECT, INSERT ON customers FROM   RA1911003010742;

Revoke succeeded.

SQL> Commit

Commit complete.

SQL> SELECT * FROM customers;

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

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

ORDER_ID

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

      1006 04-MAR-20        3200
```

DL

1008

SQL> DELETE FROM customers WHERE salesperson = 'DL';

1 row deleted.

SQL> SELECT * FROM customers;

no rows selected

SQL> Rollback;

Rollback complete.

SQL> SELECT * FROM customers;

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1006 04-MAR-20 3200

DL

1008

SQL> SAVEPOINT sp1;

Savepoint created.

SQL> DELETE FROM customers WHERE salesperson = 'DL'

1 row deleted.

SQL> SELECT * FROM customers;

no rows selected

SQL> Rollback TO sp1;

Rollback complete.

SQL> SELECT * FROM customers;

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1006 04-MAR-20 3200

DL

1008

SQL> INSERT INTO customers VALUES(1001, TO_DATE('2020-05-22', 'YYYY-MM-DD'), '1200', 'MK', '1002');

1 row created.

SQL> INSERT INTO customers VALUES(1002, TO_DATE('2020-05-23', 'YYYY-MM-DD'), '1200', 'RAKESH', '1003');

1 row created.

SQL> SELECT * FROM customers;

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1006 04-MAR-20 3200

DL

1008

1001 22-MAY-20 1200

MK

1002

CUSTOMER_ID SALE_DATE SALE_AMOUNT

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

SALESPERSON

ORDER_ID

1002 23-MAY-20 1200

RAKESH

1003

SQL> spool off

Result: Thus the DCL and TCL commands are used to modify or manipulate data records present in the customer database tables.

Experiment 4: SQL INBULIT FUNCTIONS

Aim:

To perform Inbuilt Functions in SQL

Code:

```
SQL> CREATE TABLE salesman(id INT, name VARCHAR(255), city VARCHAR(255), commission FLOAT);

Table created.

SQL> INSERT INTO salesman VALUES(5001, 'James Hoog', 'New York', 0.15);

1 row created.

SQL> INSERT INTO salesman VALUES(5002, 'Nail Knite', 'Paris', 0.13);

1 row created.

SQL> INSERT INTO salesman VALUES(5005, 'Pit Alex', 'London', 0.11);

1 row created.

SQL> SELECT ASCII(name) FROM salesman;

ASCII(NAME)

-----------

        74

        78

        80

SQL> SELECT CHR(65) FROM salesman;

C

-

A

A

A

SQL> SELECT CONCAT ('name', 'city') name_city FROM salesman;

NAME_CIT

--------

namecity

namecity

namecity

SQL> SELECT CONCAT (name, city) name_city FROM salesman;

NAME_CITY
```

```
--------------------------------------------------------------------------------

James HoogNew York

Nail KniteParis

Pit AlexLondon

SQL> SELECT INITCAP(name) FROM salesman;

INITCAP(NAME)

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

James Hoog

Nail Knite

Pit Alex

SQL> SELECT INSTR(name, 'Jam') FROM salesman;

INSTR(NAME,'JAM')

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

             1

             0

             0

SQL> SELECT INSTR(name, 'Jam') Jam_Pos FROM salesman;

  JAM_POS

----------

       1

       0

       0

SQL> SELECT INSTR(name, 'a') FROM salesman;

INSTR(NAME,'A')

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

             2

             2

             0

SQL> SELECT city, LENGTH(CITY) AS str_len FROM salesman;

CITY

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

  STR_LEN

----------

New York
```

8

Paris

5

London

6

SQL> SELCT LOWER(CITY) FROM salesman;

SP2-0734: unknown command beginning "SELCT LOWE..." - rest of line ignored.

SQL> SELECT LOWER(CITY) FROM salesman;

LOWER(CITY)

---

new york

paris

london

SQL> SELECT UPPER(city) FROM salesman;

UPPER(CITY)

NEW YORK

PARIS

LONDON

SQL> SELECT LPAD(name, 10, '.') FROM salesman;

LPAD(NAME,10,'.')

James Hoog

Nail Knite

..Pit Alex

SQL> SELECT RPAD(city, 20, '.') FROM salesman;

RPAD(CITY,20,'.')

New York.....

Paris.....

London.....

```
SQL> INSERT INTO salesman VALUES(5001, 'James Walt', 'Paris', 0.13);
1 row created.
SQL> SELECT ABS(commission) Abs FROM salesman;
       ABS
----------
       .13
SQL> SELECT ACOS(commission) Arc_cosine FROM salesman;
ARC_COSINE
----------
1.44042735
SQL> SELECT ASIN(commission) Arc_sine FROM salesman;
  ARC_SINE
----------
 .13036898
SQL> SELECT ATAN(commission) Arc_tan FROM salesman;
   ARC_TAN
----------
.129275004
SQL> SELECT CEIL(commission) FROM salesman;
CEIL(COMMISSION)
----------------
              1
SQL> SELECT FLOOR(commission) FROM salesman;
FLOOR(COMMISSION)
-----------------
              0
SQL> SELECT MOD(commission, 1) FROM salesman;
MOD(COMMISSION,1)
-----------------
             .13
SQL> SELECT POWER(commission, 12) FROM salesman;
POWER(COMMISSION,12)
--------------------
         2.3298E-11
```

```
SQL> SELECT ROUND(commission, 1) FROM salesman;

ROUND(COMMISSION,1)
-------------------
                .1
SQL> SELECT TRUNC(commission, 2) FROM salesman;

TRUNC(COMMISSION,2)
-------------------
                .13
SQL> SELECT TRUNC(commission, 1) FROM salesman;

TRUNC(COMMISSION,1)
-------------------
                .1
SQL> SELECT TRUNC(commission, 5) FROM salesman;

TRUNC(COMMISSION,5)
-------------------
                .13
SQL> SELECT TRUNC(commission, -1) FROM salesman;

TRUNC(COMMISSION,-1)
--------------------
                  0
SQL> SELECT TRUNC(commission, -2) FROM salesman;

TRUNC(COMMISSION,-2)
--------------------
                  0
SQL> CREATE TABLE dates(one DATE, two DATE);

Table created.

SQL> INSERT INTO dates VALUES(TO_DATE('2020-12-11', 'YYYY-MM-DD'), TO_DATE('2000-11-21', 'YYYY-
MM-DD'))

1 row created.

SQL> SELECT * FROM dates;

ONE       TWO
--------- ---------
11-DEC-20 21-NOV-00

SQL> SELECT ADD_MONTHS(one, 6) FROM dates;
```

ADD_MONTH

11-JUN-21

SQL> SELECT MONTHS_BETWEEN(one, two) FROM dates;

MONTHS_BETWEEN(ONE,TWO)

240.677419

SQL> SELECT ROUND(one) FROM dates;

ROUND(ONE

11-DEC-20

SQL> SELECT ROUND(one, 'YYYY') FROM dates

ROUND(ONE

01-JAN-21

SQL> SELECT ROUND(one, 'MM') FROM dates;

ROUND(ONE

01-DEC-20

SQL> SELECT TRUNC(one, 'MM') FROM dates;

TRUNC(ONE

01-DEC-20

SQL> SELECT NEXT_DAY(one, 'MON') FROM dates;

NEXT_DAY(

14-DEC-20

SQL> SELECT NEXT_DAY(one, 'WED') FROM dates;

NEXT_DAY(

16-DEC-20

SQL> SYSDATE;

SP2-0042: unknown command "SYSDATE" - rest of line ignored.

SQL> SELECT SYSDATE FROM salesman;

no rows selected

SQL> SELECT LAST_DAY(one) FROM dates;

LAST_DAY(

31-DEC-20

Result: The Inbuilt Functions have been successfully performed and tested.

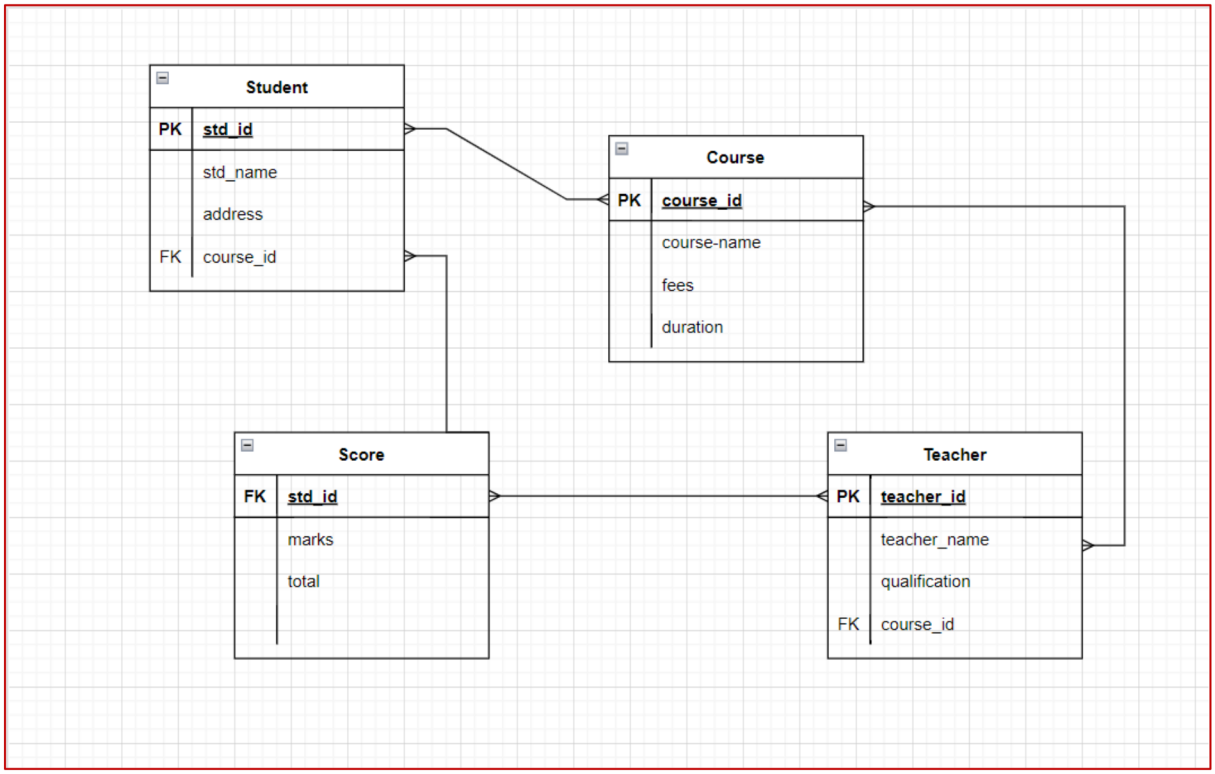
Experiment 5: CONSTRUCT AN ER-DIAGRAM FOR A PARTICULAR SCENARIO

Aim: :- To draw an ER-diagram for a Restaurant Aggregator Website (e.g. Zomato, swiggy) and Car Insurance Company.

Procedure:

The relationships as follows:

1) Each student has many courses to register and each course can be registered by many students so many-to-many relationship.
2) Each teacher has many courses to teach and each course can be taught by many teachers so many-to-many relationship.
3) Each teacher can update marks of many students and marks of a student can be updated by many teachers so many-to-many relationship.
4) Each student can access only their score and many students can access their scores one-to-many relationship.



Result: Hence the ER diagram for the Restaurant Aggregator and Car Insurance is being constructed using the above-mentioned tool draw.io

Experiment 6: VIEWS

Aim: To study the various SQL view operations on the database.

Code:

```
SQL> CREATE TABLE(
  2  Employee(
  3
SQL> CREATE TABLE EMPLOYEEE(
  2  name VARCHAR(10),
  3  no NUMBER(10),
  4  dept VARCHAR(10),
  5  deptNo NUMBER(5),
  6  date_of_join DATE);
Table created.
SQL> DESC employeee;
 Name                          Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                      VARCHAR2(10)
 NO                                        NUMBER(10)
 DEPT                                      VARCHAR2(10)
 DEPTNO                                    NUMBER(5)
 DATE_OF_JOIN                              DATE
SQL> INSERT INTO employeee VALUES('Ravi', 124, 'ECE', 89, TO_DATE('20-12-2000', 'DD-MM-YYYY'));
1 row created.
SQL> INSERT INTO employeee VALUES('Sai', 122, 'CSE', 20, TO_DATE('2-11-2001', 'DD-MM-YYYY'));
1 row created.

SQL> INSERT INTO employeee VALUES('Ram', 121, 'CSE', 20, TO_DATE('21-11-2000', 'DD-MM-YYYY'));
1 row created.
SQL> INSERT INTO employeee VALUES('John', 100, 'CSE', 20, TO_DATE('16-12-2002', 'DD-MM-YYYY'));
1 row created.
SQL> INSERT INTO employeee VALUES('Reeve', 130, 'MECH', 10, TO_DATE('16-12-2002', 'DD-MM-
YYYY'));
1 row created.
```

```
SQL> CREATE VIEW EmpView as SELECT name, no, dept, deptno, date_of_join from Employeee;

View created.

SQL> DESC EmpView;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                               VARCHAR2(10)
 NO                                                 NUMBER(10)
 DEPT                                               VARCHAR2(10)
 DEPTNO                                             NUMBER(5)
 DATE_OF_JOIN                                       DATE
SQL> SELECT * FROM EmpView;

NAME            NO DEPT         DEPTNO DATE_OF_J
---------- ---------- ---------- ---------- ---------
Ravi           124 ECE             89 20-DEC-00
Sai            122 CSE             20 02-NOV-01
Ram             121 CSE             20 21-NOV-00
John           100 CSE             20 16-DEC-02
Reeve           130 MECH            10 16-DEC-02

SQL> INSERT INTO EmpView VALUES('Emilia', 145, 'MECH', 10, TO_DATE('12-01-2006', 'DD-MM-YYYY'));

1 row created.

SQL> SELECT * FROM EmpView;

NAME            NO DEPT         DEPTNO DATE_OF_J
---------- ---------- ---------- ---------- ---------
Ravi           124 ECE             89 20-DEC-00
Sai            122 CSE             20 02-NOV-01
Ram             121 CSE             20 21-NOV-00
John           100 CSE             20 16-DEC-02
Reeve           130 MECH            10 16-DEC-02
Emilia          145 MECH            10 12-JAN-06

6 rows selected.

SQL> SELECT * FROM Employeee;

NAME            NO DEPT         DEPTNO DATE_OF_J
```

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

Ravi 124 ECE 89 20-DEC-00

Sai 122 CSE 20 02-NOV-01

Ram 121 CSE 20 21-NOV-00

John 100 CSE 20 16-DEC-02

Reeve 130 MECH 10 16-DEC-02

Emilia 145 MECH 10 12-JAN-06

6 rows selected.

SQL> DELETE FROM EmpView where name = 'Emilia';

1 row deleted

SQL> UPDATE Empview SET name = 'Sai Ram' where name = 'Sai';

1 row updated

SQL> SELECT * FROM EmpView;

NAME NO DEPT DEPTNO DATE_OF_J

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

Ravi 124 ECE 89 20-DEC-00

Sai Ram 122 CSE 20 02-NOV-01

Ram 121 CSE 20 21-NOV-00

John 100 CSE 20 16-DEC-02

Reeve 130 MECH 10 16-DEC-02

SQL> CREATE OR REPLACE VIEW Emp_view as SELECT no, name, dept from Employeee;

View created.

SQL> SELECT * FROM Emp_View;

N NAME DEPT

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

124 Ravi ECE

122 Sai Ram CSE

121 Ram CSE

100 John CSE

130 Reeve MECH

SQL> CREATE OR REPLACE VIEW Emp_view_CSE as SELECT * from Employeee WHERE dept = 'CSE';

View created.

SQL> SELECT * FROM Emp_View_CSE;

NAME NO DEPT DEPTNO DATE_OF_J

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

Sai Ram 122 CSE 20 02-NOV-01

Ram 121 CSE 20 21-NOV-00

John 100 CSE 20 16-DEC-02

SQL> CREATE OR REPLACE VIEW Emp_view_date(ID, Employeename, JOIN_DATE) as SELECT no, name, date_of_join from Employeee;

View created

SQL> SELECT * FROM Emp_View_date;

ID EMPLOYEENA JOIN_DATE

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

124 Ravi 20-DEC-00

122 Sai Ram 02-NOV-01

121 Ram 21-NOV-00

100 John 16-DEC-02

130 Reeve 16-DEC-02

SQL> CREATE OR REPLACE VIEW Emp_view_date as SELECT no "ID", name "EMP_NAME", date_of_join "JOIN_DATE" from Employeee;

View created.

SQL> SELECT * FROM Emp_View_date;

ID EMP_NAME JOIN_DATE

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

124 Ravi 20-DEC-00

122 Sai Ram 02-NOV-01

121 Ram 21-NOV-00

100 John 16-DEC-02

130 Reeve 16-DEC-02

SQL> CREATE OR REPLACE VIEW Emp_view_date as SELECT E.no "ID", E.name "EMP_NAME", E.date_of_join "JOIN_DATE" from Employeee E;

View created.

SQL> SELECT * FROM Emp_View_date;

ID EMP_NAME JOIN_DATE

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

124 Ravi 20-DEC-00

122 Sai Ram 02-NOV-01

121 Ram 21-NOV-00

100 John 16-DEC-02

130 Reeve 16-DEC-02

SQL> spool off

Result:

Thus the SQL views have been executed successfully

Experiment 7: JOINS

Aim: To study the various Join operations on the database.

Code:

```
SQL> CREATE TABLE orders(O_id number(5), Orderno number(5), P_Id number(3));

Table created.

SQL> DESC orders;
 Name                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 O_ID                                  NUMBER(5)
 ORDERNO                               NUMBER(5)
 P_ID                                  NUMBER(3)
SQL> INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id);
Enter value for o_id: 1
Enter value for orderno: 77895
Enter value for p_id: 3
old   1: INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id)
new   1: INSERT INTO orders VALUES(1, 77895, 3)
1 row created.
SQL> INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id);
Enter value for o_id: 2
Enter value for orderno: 44678
Enter value for p_id: 3
old   1: INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id)
new   1: INSERT INTO orders VALUES(2, 44678, 3
1 row created.
SQL> INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id);
Enter value for o_id: 3
Enter value for orderno: 22456
Enter value for p_id: 1
old   1: INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id)
new   1: INSERT INTO orders VALUES(3, 22456, 1)
```

1 row created.

SQL> INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id);

Enter value for o_id: 4

Enter value for orderno: 24562

Enter value for p_id: 1

old 1: INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id)

new 1: INSERT INTO orders VALUES(4, 24562, 1)

1 row created.

SQL> INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id);

Enter value for o_id: 5

Enter value for orderno: 34764

Enter value for p_id: 15

old 1: INSERT INTO orders VALUES(&O_id, &Orderno, &P_Id)

new 1: INSERT INTO orders VALUES(5, 34764, 15)

1 row created.

SQL> SELCT * FROM orders;

SP2-0734: unknown command beginning "SELCT * FR..." - rest of line ignored.

SQL> SELECT * FROM orders;

O_ID	ORDERNO	P_ID
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

SQL> CREATE TABLE persons(p_Id number(5), lastname varchar2(15), firstname varchar2(15), Address varchar2(20), city varchar2(10));

Table created.

SQL> INSERT INTO persons VALUES(&p_Id, '&lastname', '&firstname','&address', '&city');

Enter value for p_id: 1

Enter value for lastname: Hansen

```
Enter value for firstname: Ola
Enter value for address: Timoteivn 10
Enter value for city: sadnes
old   1: INSERT INTO persons VALUES(&p_Id, '&lastname', '&firstname','&address', '&city')
new   1: INSERT INTO persons VALUES(1, 'Hansen', 'Ola','Timoteivn 10', 'sadnes')
1 row created.
SQL> INSERT INTO persons VALUES(&p_Id, '&lastname', '&firstname','&address', '&city');
Enter value for p_id: 2
Enter value for lastname: Svenson
Enter value for firstname: Tove
Enter value for address: Borgn 23
Enter value for city: Sandnes
old   1: INSERT INTO persons VALUES(&p_Id, '&lastname', '&firstname','&address', '&city')
new   1: INSERT INTO persons VALUES(2, 'Svenson', 'Tove','Borgn 23', 'Sandnes')
1 row created.
SQL> INSERT INTO persons VALUES(&p_Id, '&lastname', '&firstname','&address', '&city');
Enter value for p_id: 3
Enter value for lastname: Pettersen
Enter value for firstname: Kari
Enter value for address: Storgt 20
Enter value for city: Stavanger
old   1: INSERT INTO persons VALUES(&p_Id, '&lastname', '&firstname','&address', '&city')
new   1: INSERT INTO persons VALUES(3, 'Pettersen', 'Kari','Storgt 20', 'Stavanger')
1 row created.
SQL> SELECT * from persons;

    P_ID LASTNAME       FIRSTNAME      ADDRESS            CITY
---------- --------------- --------------- -------------------- ----------
         1 Hansen          Ola            Timoteivn 10        sadnes
         2 Svenson         Tove           Borgn 23            Sandnes
         3 Pettersen       Kari           Storgt 20           Stavanger
```

```
SQL> select persons.lastname, persons.firstname, orders.orderno
  2  FROM persons
  3  LEFT JOIN orders
  4  ON persons.p_Id = orders.p_Id
  5  ORDER by persons.lastname;
LASTNAME       FIRSTNAME        ORDERNO
-------------- --------------- ----------
Hansen         Ola              22456
Hansen         Ola              24562
Pettersen      Kari             44678
Pettersen      Kari             77895
Svenson        Tove
SQL> SELECT persons.lastname, persons.firstname, orders.orderno
  2  FROM persons
  3  FULL OUTER JOIN orders
  4  ON persons.p_Id = orders.p_Id
  5  ORDER by persons.lastname;
LASTNAME       FIRSTNAME        ORDERNO
-------------- --------------- ----------
Hansen         Ola              22456
Hansen         Ola              24562
Pettersen      Kari             44678
Pettersen      Kari             77895
Svenson        Tove
                                34764
6 rows selected.
SQL> SELECT persons.lastname, persons.firstname, orders.orderno FROM persons
  2  RIGHT JOIN orders
  3  on persons.p_Id = orders.p_id
  4  order by persons.lastname;
LASTNAME       FIRSTNAME        ORDERNO
```

--- --- ---

Hansen Ola 22456

Hansen Ola 24562

Pettersen Kari 77895

Pettersen Kari 44678

34764

```
SQL> SELECT persons.lastname, persons.firstname, orders.orderno
  2  FROM persons
  3  INNER JOIN orders
  4  ON persons.p_Id = orders.p_Id
  5  ORDER by persons.lastname;
```

LASTNAME FIRSTNAME ORDERNO

--- --- ---

Hansen Ola 22456

Hansen Ola 24562

Pettersen Kari 77895

Pettersen Kari 44678

```
SQL> spool off
```

Result: Thus the joining tables have been executed successfully.

Experiment 8: Nested Query

Aim: To study the various net query operations sucessfully.

Code:

```
SQL> CREATE TABLE emp
  2  (
  3  empno NUMBER,
  4  ename VARCHAR2(255),
  5  job VARCHAR2(20),
  6  sal NUMBER,
  7  deptno NUMBER
  8  );
Table created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7369
Enter value for ename: 'SMITH'
Enter value for job: 'CLERK'
Enter value for sal: 800
Enter value for dept_no: 20
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7369, 'SMITH', 'CLERK', 800, 20)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7900
Enter value for ename: 'JAMES'
Enter value for job: 'CLERK'
Enter value for sal: 950
Enter value for dept_no: 30
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7900, 'JAMES', 'CLERK', 950, 30)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
```

```
Enter value for empno: 7876
Enter value for ename: 'ADAMS'
Enter value for job: 'CLERK'
Enter value for sal: 1100
Enter value for dept_no: 20
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7876, 'ADAMS', 'CLERK', 1100, 20)
1 row created
SQL> SELECT * FROM emp;
    EMPNO
----------
ENAME
------------------------------------------------------------------------------
JOB                   SAL     DEPTNO
-------------------- ---------- ----------
     7369
SMITH
CLERK                  800        20


     7900
JAMES
CLERK                  950        30


    EMPNO
----------
ENAME
------------------------------------------------------------------------------
JOB                   SAL     DEPTNO
-------------------- ---------- ----------
     7876
ADAMS
```

```
CLERK                1100        20
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7521
Enter value for ename: 'WARD'
Enter value for job: 'SALESMAN'
Enter value for sal: 1250
Enter value for dept_no: 10
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7521, 'WARD', 'SALESMAN', 1250, 10)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7654
Enter value for ename: 'MARTIN'
Enter value for job: 'SALESMAN'
Enter value for sal: 1250
Enter value for dept_no: 40
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7654, 'MARTIN', 'SALESMAN', 1250, 40)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7499
Enter value for ename: 'ALLEN'
Enter value for job: 'SALESMAN'
Enter value for sal: 1300
Enter value for dept_no: 30
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7499, 'ALLEN', 'SALESMAN', 1300, 30)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7566
Enter value for ename: 'JONES'
```

Enter value for job: 'MANAGER'

Enter value for sal: 2975

Enter value for dept_no: 20

old 1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)

new 1: INSERT INTO emp VALUES(7566, 'JONES', 'MANAGER', 2975, 20)

1 row created.

SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);

Enter value for empno: 7698

Enter value for ename: 'BLAKE'

Enter value for job: 'MANAGER'

Enter value for sal: 2850

Enter value for dept_no: 10

old 1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)

new 1: INSERT INTO emp VALUES(7698, 'BLAKE', 'MANAGER', 2850, 10)

1 row created.

SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);

Enter value for empno: 7902

Enter value for ename: 'FORD'

Enter value for job: 'ANALYST'

Enter value for sal: 3000

Enter value for dept_no: 50

old 1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)

new 1: INSERT INTO emp VALUES(7902, 'FORD', 'ANALYST', 3000, 50)

1 row created.

SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);

Enter value for empno: 7782

Enter value for ename: 'CLARK'

Enter value for job: 'MANAGER'

Enter value for sal: 2500

Enter value for dept_no: 20

old 1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)

```
new   1: INSERT INTO emp VALUES(7782, 'CLARK', 'MANAGER', 2500, 20)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7788
Enter value for ename: 'SCOTT'
Enter value for job: 'ANALYST'
Enter value for sal: 3000
Enter value for dept_no: 60
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7788, 'SCOTT', 'ANALYST', 3000, 60)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7839
Enter value for ename: 'KING'
Enter value for job: 'PRESIDENT'
Enter value for sal: 5000
Enter value for dept_no: 10
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7839, 'KING', 'PRESIDENT', 5000, 10)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7844
Enter value for ename: 'TURNER'
Enter value for job: 'SALESMAN'
Enter value for sal: 2115
Enter value for dept_no: 40
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7844, 'TURNER', 'SALESMAN', 2115, 40)
1 row created.
SQL> INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no);
Enter value for empno: 7669
```

```
Enter value for ename: 'MILLER'
Enter value for job: 'CLERK'
Enter value for sal: 1300
Enter value for dept_no: 10
old   1: INSERT INTO emp VALUES(&empno, &ename, &job, &sal, &dept_no)
new   1: INSERT INTO emp VALUES(7669, 'MILLER', 'CLERK', 1300, 10)
1 row created.
SQL> SELECT * FROM emp;
    EMPNO
----------
ENAME
------------------------------------------------------------------------------
JOB                   SAL     DEPTNO
------------------- ---------- ----------
     7369
SMITH
CLERK                  800        20
     7900
JAMES
CLERK                  950        30
    EMPNO
----------
ENAME
------------------------------------------------------------------------------
JOB                   SAL     DEPTNO
------------------- ---------- ----------
     7876
ADAMS
CLERK                 1100        20
     7521
WARD
```

EMPNO

ENAME

JOB SAL DEPTNO

--- --- ---

SALESMAN 1250 10

7654

MARTIN

SALESMAN 1250 40

7499

EMPNO

ENAME

JOB SAL DEPTNO

--- --- ---

ALLEN

SALESMAN 1300 30

7566

JONES

MANAGER 2975 20

EMPNO

ENAME

JOB SAL DEPTNO

--- --- ---

7698

BLAKE

MANAGER 2850 10

7902

FORD

ANALYST 3000 50

EMPNO

ENAME

JOB SAL DEPTNO

--- --- ---

7782

CLARK

MANAGER 2500 20

7788

SCOTT

EMPNO

ENAME

JOB SAL DEPTNO

--- --- ---

ANALYST 3000 60

7839

KING

PRESIDENT 5000 10

7844

EMPNO

ENAME

JOB SAL DEPTNO

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

TURNER

SALESMAN 2115 40

7669

MILLER

CLERK 1300 10

14 rows selected.

SQL> ALTER TABLE emp MODIFY (ename VARCHAR2(10));

Table altered.

SQL> SELECT * FROM emp;

EMPNO ENAME JOB SAL DEPTNO

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

7369 SMITH CLERK 800 20

7900 JAMES CLERK 950 30

7876 ADAMS CLERK 1100 20

7521 WARD SALESMAN 1250 10

7654 MARTIN SALESMAN 1250 40

7499 ALLEN SALESMAN 1300 30

7566 JONES MANAGER 2975 20

7698 BLAKE MANAGER 2850 10

7902 FORD ANALYST 3000 50

7782 CLARK MANAGER 2500 20

7788 SCOTT ANALYST 3000 60

EMPNO ENAME JOB SAL DEPTNO

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

7839 KING PRESIDENT 5000 10

7844 TURNER SALESMAN 2115 40

7669 MILLER CLERK 1300 10

14 rows selected

SQL> select ename from emp where sal > (select sal from emp where empno=7566);

ENAME

```
----------

FORD

SCOTT

KING

SQL> SELECT ename FROM EMP WHERE job = ( SELECT job FROM emp WHERE empno=7369) AND
sal>(SELECT sal from emp where empno=7876);

ENAME

----------

MILLER

SQL> select ename, job, sal from emp where sal = (select min(sal) from emp);

ENAME      JOB                       SAL

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

SMITH      CLERK                     800

SQL> select deptno, min(sal) from emp group by deptno having min(sal) > (select min(sal) from emp
where deptno = 20);

   DEPTNO   MIN(SAL)

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

       50       3000

       40       1250

       30        950

       10       1250

       60       3000

SQL> select empno,ename,job from emp where sal<any(select sal from emp where job='CLERK') and
job <>'CLERK';

    EMPNO ENAME      JOB

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

     7521 WARD       SALESMAN

     7654 MARTIN     SALESMAN

SQL> SELECT empno, ename, job

  2   FROM emp

  3   WHERE sal > ALL

  4                ( SELECT AVG(sal)
```

```
  5                   FROM emp
  6                  GROUP BY deptno )
  7  ;
    EMPNO ENAME      JOB
---------- ---------- --------------------
     7839 KING       PRESIDENT
```

SQL> select ename, deptno, sal, comm from employee where (sal, nvl(comm,-1)) in (select sal, nvl(comm,-1) from employee where deptno = 30);

ENAME	DEPTNO	SAL	JOB	SAL
WARD	23	1920	SALESMAN	1250
MARTIN	44	1300	SALESMAN	1250
FORD	46	465	ANALYST	3000
SCOTT	49	825	ANALYST	3000

SQL> SELECT ename,job,sal FROM emp WHERE sal

```
  2  IN (SELECT sal FROM emp where ename='SCOTT' OR ename='WARD');
```

ENAME	JOB	SAL
WARD	SALESMAN	1250
MARTIN	SALESMAN	1250
FORD	ANALYST	3000
SCOTT	ANALYST	3000

SQL> SELECT ename,job,sal FROM emp WHERE sal IN

```
  2  (SELECT sal FROM emp where ename='FORD')
  3  and job IN
  4  (SELECT job FROM emp where ename='FORD');
```

ENAME	JOB	SAL
FORD	ANALYST	3000
SCOTT	ANALYST	3000

SQL> SELECT ename,job,deptno,sal FROM emp WHERE job IN

```
  2  (SELECT job FROM emp where ename='JONES') and sal>
  3  (SELECT sal FROM emp where ename='FORD')
  4
SQL> SELECT ename,job,deptno,sal FROM emp WHERE job IN
  2  (SELECT job FROM emp where ename='JONES') and sal>
  3  (SELECT sal FROM emp where ename='FORD');
no rows selected
SQL> select ename,empno,deptno from emp s
where exists (select * from emp
where s.empno=mgr)
order by empno;
ENAME      EMPNO                    DEPTNO
---------- -------------------- ----------
FORD       7456                 3000
SCOTT      7751                 3000
SQL> spool off;
```

Result: Thus study the various net query operations successfully

Experiment 9: PL/SQL Conditional and Iterative Statements

Aim: To study the PL/SQL conditional and iterative statements successfully

Code:

```
SQL> set serveroutput on;
SQL> declare
  2  a number;
  3  b number;
  4  c number;
  5  begin
  6  dbms_output.put_line('Enter a: ');
  7  a:=&a;
  8  dbms_output.put_line('Enter b: ');
  9  b:=&b;
 10  dbms_output.put_line('Enter c: ');
 11  c:=&c;
 12  if(a>b) and (a>c) then
 13  dbms_output.put_line('A is MAX');
 14  elsif(b>a) and (b>c) then
 15  dbms_output.put_line('B is MAX');
 16  else
 17  dbms_output.put_line('C is MAX');
 18  end if;
 19  end;
 20  /
Enter value for a: 11
old   7: a:=&a;
new   7: a:=11;
Enter value for b: 22
old   9: b:=&b;
```

```
new   9: b:=22;
Enter value for c: 33
old  11: c:=&c;
new  11: c:=33;
Enter a:
Enter b:
Enter c:
C is MAX
PL/SQL procedure successfully completed.
SQL> declare
  2  n number;
  3  sum1 number default 0;
  4  begin
  5  n:=1;
  6  for n in 1..100
  7  loop
  8         if mod(n,2)=1
  9         then
 10         sum1:=sum1+n;
 11         end if;
 12  end loop;
 13  dbms_output.put_line('Sum = '||sum1);
 14  end;
 15  /
Sum = 2500
PL/SQL procedure successfully completed.
SQL> spool off
```

Result:

Thus study the PL/SQL conditional and iterative statements successfully

Experiment 10: PL/SQL Procedures

Aim:

To study PL/SQL Procedures and to execute sucessfully

Code:

```
DECLARE
  a number;
  b number;
  c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
  IF x < y THEN
    z:= x;
  ELSE
    z:= y;
  END IF;
END;
BEGIN
  a:= 23;
  b:= 45;
  findMin(a, b, c);
  dbms_output.put_line(' Minimum of (23, 45) : ' || c);
END;
/
```

```
Minimum of (23, 45) : 23
PL/SQL procedure successfully completed.
```

Result: Thus study PL/SQL Procedures and to execute sucessfully

Experiment 11: PL/SQL Functions

Aim: To study PL/SQL Functions and execute them successfully

Code:

```
DECLARE
  a number;
  b number;
  c number;
FUNCTION findMax(x IN number, y IN number)
RETURN number
IS
   z number;
BEGIN
  IF x > y THEN
    z:= x;
  ELSE
    Z:= y;
  END IF;
  RETURN z;
END;
BEGIN
  a:= 23;
  b:= 45;
  c := findMax(a, b);
  dbms_output.put_line(' Maximum of (23,45): ' || c);
END;
/
```

```
Maximum of (23,45): 45
PL/SQL procedure successfully completed.
```

Result: Thus study PL/SQL Functions and executed them sucessfully

Experiment 12: PL/SQL Cursors

Aim: To study **PL/SQL Cursors** and execute them successfully.

Code:

```
SQL> declare
2           s_rno cse5.rno%type;
3           s_name cse5.name%type;
4           s_ grade cmarks.grade %type;
7           cursor s cses is
8           select cse5.rno,cse5.name,cmarks.grade from cse5 Inner     join cmarks on
cse5.rno=cmarks.rno where mark<70;
9           begin
10          open s_cse5;
12          loop
13              fetch s_cse5 into s_rno,s_name,s_grade;
15              exit when s_cse%notfound;
17             dbms_output.put_line(s_rno||' '||s_name||' '||s_grade);
18          end loop;
19         close s_cse5;
20
21 end;
22   /
2 rishi b
3 saran A
PL/SQL procedure successfully completed.
```

Result: Thus study PL/SQL Cursors and execute them successfully.

Experiment 13: PL/SQL Exception Handling

Aim: To study and execute PL/SQL Exception Handling successfully

Code:

```
DECLARE
  c_id customers.id%type := &cc_id;
  c_name customerS.Name%type;
  c_addr customers.address%type;
  -- user defined exception
  ex_invalid_id  EXCEPTION;
BEGIN
  IF c_id <= 0 THEN
    RAISE ex_invalid_id;
  ELSE
    SELECT  name, address INTO  c_name, c_addr
    FROM customers
    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 must 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 cc_id: -6 (let's enter a value -6)
old  2: c_id customers.id%type := &cc_id;
```

new 2: c_id customers.id%type := -6;

ID must be greater than zero!

PL/SQL procedure successfully completed

Result:

Thus study and execute PL/SQL Exception Handling successfully

Experiment 14: PL/SQL Triggers

Aim: To study and execute PL/SQL triggers successfully

Code:

```
SQL> create or replace trigger t2
2 before insert on t
3 for each
4 begin
5 :new.total:= :new.mark1 + :new.mark2 + :new.mark3;
6 end;
7 /
Trigger created.
SQL> insert into t (rno, marki, mark2, mark3) values (101,89,57,78);
1 row created.
SQL> select * from t;
```

RNO	MARK1	MARK2	MARK3	TOTAL
1	25	35	45	
2	28	36	49	
3	50	29	60	
101	89	57	78	224

Result: Thus study and execute PL/SQL triggers successfully