LAB PROGRAM -1

1) Write SQL procedure to illustrate DDL Commands.

AIM : Illustrating create table, alter table and drop table commands using Student Table.

PROCEDUR

Step 1: Creating student table.

```
SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4),
     S_NAME VARCHAR2(20), S_ADDR VARCHAR2(30));
```

Step 2: Displaying table structure.

```
SQL> DESCRIBE STUDENT_101;
```

Name	Null?	Type
S_NO		NUMBER(4)
S_NAME		VARCHAR2(20)
S_ADDR		VARCHAR2(30)

Step 3: Changing (Modifying) column's data characteristics.

```
SQL> ALTER TABLE STUDENT_101 MODIFY(S_NO NUMBER(6));
SQL> DESCRIBE STUDENT_101;
```

Name	Null?	Type
S_NO		NUMBER(6)
S_NAME		VARCHAR2(20)
S_ADDR		VARCHAR2(30)

Step 4: Adding a column.

```
SQL> ALTER TABLE STUDENT_101 ADD(S_PHONE NUMBER(10));
SQL> DESCRIBE STUDENT_101;
```

Name	Null?	Type
S_NO		NUMBER(6)
S_NAME		VARCHAR2(20)
S_ADDR		VARCHAR2(30)
S_PHONE		NUMBER(10)

Step 5: Adding PRIMARY KEY to a column.

```
SQL> ALTER TABLE STUDENT_101 ADD PRIMARY KEY(S_NO);
SQL> DESCRIBE STUDENT_101;
```

Name	Null?	Type
S_NO	Not Null	NUMBER(6)
S_NAME		VARCHAR2(20)
S_ADDR		VARCHAR2(30)
S_PHONE		NUMBER(10)

Step 6: Removing (Dropping) a table.

```
SQL> DROP TABLE STUDENT_101;
SQL> SELECT * FROM TAB;
```

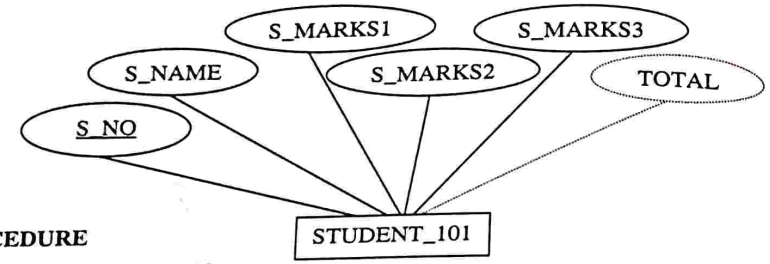
TNAME	TABTYPE CLUSTERID
BONUS	TABLE
DEPT	TABLE
EMP	TABLE
SALGRADE	TABLE

LAB PROGRAM - 2

2) Write SQL procedure to illustrate DML Commands.

AIM : Illustrating insert , update, select and delete commands using Student Table.

ER-DIAGRAM



PROCEDURE

Step 1: Creating student table.

```
SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4) PRIMARY KEY ,
     S_NAME VARCHAR2(20), S_MARKS1 NUMBER(3), S_MARKS2
     NUMBER(3), S_MARKS3 NUMBER(3), TOTAL NUMBER(3) );
```

Step 2 : Inserting 5 rows into STUDENT_101 table.

```
SQL> INSERT INTO STUDENT_101 VALUES(101,'RAVI',66,88,77,0);
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(102,'HAMEED',56,78,67,0);
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(103,'THARUN',76,58,77,0);
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(104,'PHANI',66,68,87,0);
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(105,'ANITHA',86,67,57,0);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3 : Selecting (Displaying) all rows from the STUDENT_101 table.

```
SQL> SELECT * FROM STUDENT_101;
```

S_NO	S_NAME	S_MARKS1	S_MARKS2	S_MARKS3	TOTAL
101	RAVI	66	88	77	0
102	HAMEED	56	78	67	0
103	THARUN	76	58	77	0
104	PHANI	66	68	87	0
105	ANITHA	86	67	57	0

Step 4: Updating column TOTAL.

```
SQL> UPDATE STUDENT_101 SET TOTAL=S_MARKS1+S_MARKS2+S_MARKS3;
```

5 rows updated.

Step 5: Selecting (Displaying) all rows with updated TOTAL column from the STUDENT_101 table.

```
SQL> SELECT * FROM STUDENT_101;
```

S_NO	S_NAME	S_MARKS1	S_MARKS2	S_MARKS3	TOTAL
101	RAVI	66	88	77	231
102	HAMEED	56	78	67	201
103	THARUN	76	58	77	211
104	PHANI	66	68	87	221
105	ANITHA	86	67	57	210

Step 6 : Removing (Deleting) a row from the table.

```
SQL> DELETE FROM STUDENT_101 WHERE S_NO=105;
```

1 row deleted.

Step 6 : Selecting (Displaying) rows after deleting a row from the STUDENT_101 table.

```
SQL> SELECT * FROM STUDENT_101;
```

S_NO	S_NAME	S_MARKS1	S_MARKS2	S_MARKS3	TOTAL
101	RAVI	66	88	77	231
102	HAMEED	56	78	67	201
103	THARUN	76	58	77	211
104	PHANI	66	68	87	221

LAB PROGRAM -3

3) Write SQL procedure to illustrate different operators available in SQL.

AIM : Illustrating different operators available in SQL on Student table.

PROCEDURE

Step 1: Creating student table.

```
SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4) PRIMARY KEY ,
     S_NAME VARCHAR2(20), S_MARKS1 NUMBER(3), S_MARKS2
     NUMBER(3), S_MARKS3 NUMBER(3), TOTAL NUMBER(3) );
```

Step 2 : Inserting 5 rows into STUDENT_101 table.

```
SQL> INSERT INTO STUDENT_101 VALUES(101,'RAVI',66,88,77,0);
1 row created.
SQL> INSERT INTO STUDENT_101 VALUES(102,'HAMEED',56,78,67,0);
1 row created.
SQL> INSERT INTO STUDENT_101 VALUES(103,'THARUN',76,58,77,0);
1 row created.
SQL> INSERT INTO STUDENT_101 VALUES(104,'PHANI',26,68,87,0);
1 row created.
SQL> INSERT INTO STUDENT_101 VALUES(105,'ANITHA',86,67,57,0);
1 row created.
SQL> COMMIT;
Commit complete.
```

Step 3: Updating column TOTAL.

```
SQL> UPDATE STUDENT_101 SET TOTAL=S_MARKS1+S_MARKS2+S_MARKS3;
```

5 rows updated.

Step 4: Adding new columns AVG and RESULT into STUDENT_101 table.

```
SQL> ALTER TABLE STUDENT_101 ADD(AVG NUMBER(5,2),
RESULT VARCHAR2(6));
```

Table altered.

Step 5: Updating column AVG.

```
SQL>  UPDATE STUDENT_101 SET AVG=TOTAL/3;
```

5 rows updated.

Step 6: Updating column RESULT.

```
SQL> UPDATE STUDENT_101 SET RESULT='PASS' WHERE S_MARKS1>=35
     AND   S_MARKS2>=35 AND S_MARKS3>=35;
```

4 rows updated.

```
SQL> UPDATE STUDENT_101 SET RESULT='FAIL' WHERE S_MARKS1<35 OR
      S_MARKS2<35 OR S_MARKS3<35;
```

1 row updated.

Step 7 : Selecting (Displaying) all rows from the STUDENT_101 table.

```
SQL> SELECT * FROM STUDENT_101;
```

S_NO	S_NAME	S_MARKS1	S_MARKS2	S_MARKS3	TOTAL	AVG	RESULT
101	RAVI	66	88	77	231	77	PASS
102	HAMEED	56	78	67	201	67	PASS
103	THARUN	76	58	77	211	70.33	PASS
104	PHANI	66	68	87	221	60.33	FAIL
105	ANITHA	86	67	57	210	70	PASS

Step 8: Display S_NO,S_NAME and AVG of all passed students who got above 75% marks.

```
SQL> SELECT S_NO,S_NAME,AVG FROM STUDENT_101 WHERE AVG>=75;
```

S_NO	S_NAME	AVG
101	RAVI	77

Step 9: Display S_NO,S_NAME and AVG of all passed students who got above 60% and below 75% marks.

```
SQL> SELECT S_NO,S_NAME,AVG FROM STUDENT_101 WHERE AVG>=60 AND
     AVG<=75;
```

S_NO	S_NAME	AVG
102	HAMEED	67
103	THARUN	70.33
105	ANITHA	70

Step 10: Display the name of the students having 'H' as second letter in their name.

```
SQL> SELECT S_NAME FROM STUDENT_101 WHERE S_NAME LIKE '_H%';
```

SNAME
THARUN
PHANI

Step 11: Display the Students Number and Names of the students whose S_NO between 102 and 104.

```
SQL> SELECT S_NO,S_NAME FROM STUDENT_101 WHERE S_NO BETWEEN
     102 AND 104;
```

S_NO	S_NAME
102	HAMEED
103	THARUN
104	PHANI

Step 12: Display the Students Number and Names of the students whose S_NO not equal to 102 and 104.

```
SQL> SELECT S_NO,S_NAME FROM STUDENT_101 WHERE S_NO NOT
     IN(102,104);
```

S_NO	S_NAME
101	RAVI
103	THARUN
105	ANITHA

LAB PROGRAM - 4

4) Write SQL procedure to illustrate aggregate functions.

AIM : Illustrating aggregate functions on Employee table.

PROCEDURE

Step 1: Creating Employee table.

```
SQL> CREATE TABLE EMPLOYEE_101(EMP_NO    NUMBER(4),EMP_NAME
       VARCHAR2(20), EMP_DOJ    DATE,EMP_CITY   VARCHAR2(15),
       EMP_SALARY     NUMBER(9,2));
```

Step 2: inserting rows into EMPLOYEE_101 table.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(111,'RAZAQ','01-JAN-1990',
     'ANANTAPURAMU', 20000);
```

1 row created.

```
SQL>INSERT INTO EMPLOYEE_101 VALUES(112,'RAMESH',
 '02-FEB-1989', 'HINDUPUR',25000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(113,'RADHIKA',
'04-MAR-1985', 'DHARMAVARAM',18000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(114,'RAMANI',
'07-FEB-1979', 'ANANTAPURAMU',15000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(115,'RAJESH',
'06-APR-1980', 'HUNDUPUR',17000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(116,'RAJU','05-MAY-2000',
     'ANANTAPURAMU',20000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(117,'RAJESH',
'03-JUN-2005', 'DHARMAVARAM',16000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(118,'REKHA','02-DEC-2010',
     'HINDUPUR',19000);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3: Selecting (Displaying) all rows from the EMPLOYEE_101 table.

```
SQL> SELECT * FROM EMPLOYEE_101;
```

EMP_NO	EMP_NAME	EMP_DOJ	EMP_CITY	EMP_SALARY
111	RAZAQ	1-Jan-90	ANANTAPURAMU	20000
112	RAMESH	2-Feb-89	HINDUPUR	25000
113	RADHIKA	4-Mar-85	DHARMAVARAM	18000
114	RAMANI	7-Feb-79	ANANTAPURAMU	15000
115	RAJESH	6-Apr-80	HUNDUPUR	17000
116	RAJU	5-May-00	ANANTAPURAMU	20000
117	RAJESH	3-Jun-05	DHARMAVARAM	16000
118	REKHA	2-Dec-10	HINDUPUR	19000

8 rows selected.

Step 4: Performing aggregate functions on the column EMP_SALARY.

```
SQL> SELECT COUNT(EMP_SALARY) FROM EMPLOYEE_101;
```

COUNT(EMP_SALARY)
8

```
SQL> SELECT COUNT(*) FROM EMPLOYEE_101;
```

COUNT(*)
8

```
SQL> SELECT SUM(EMP_SALARY) FROM EMPLOYEE_101;
```

SUM(EMP_SALARY)
50000

```
SQL> SELECT AVG(EMP_SALARY) FROM EMPLOYEE_101;
```

AVG(EMP_SALARY)
18750

```
SQL> SELECT MIN(EMP_SALARY) FROM EMPLOYEE_101;
```

MIN(EMP_SALARY)
15000

```
SQL> SELECT MAX(EMP_SALARY) FROM EMPLOYEE_101;
```

MAX(EMP_SALARY)
25000

LAB PROGRAM - 5

5) Write SQL procedure to illustrate grouping data.

AIM: Illustrating grouping data on Employee table.

PROCEDURE

Step 1: Creating Employee table.

```
SQL> CREATE TABLE EMPLOYEE_101(EMP_NO    NUMBER(4),EMP_NAME
     VARCHAR2(20), EMP_DOJ    DATE,EMP_CITY   VARCHAR2(15),
     EMP_SALARY     NUMBER(9,2));
```

Step 2: inserting rows into EMPLOYEE_101 table.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(111,'RAZAQ','01-JAN-1990',
     'ANANTAPURAMU', 20000);
```

1 row created.

```
SQL>INSERT INTO EMPLOYEE_101 VALUES(112,'RAMESH', '02-FEB-1989',
     'HINDUPUR',25000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(113,'RADHIKA',
'04-MAR-1985', 'DHARMAVARAM',18000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(114,'RAMANI',
'07-FEB-1979', 'ANANTAPURAMU',15000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(115,'RAJESH',
'06-APR-1980', 'HUNDUPUR',17000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(116,'RAJU','05-MAY-2000',
     'ANANTAPURAMU',20000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(117,'RAJESH',
'03-JUN-2005', 'DHARMAVARAM',16000);
```

1 row created.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(118,'REKHA','02-DEC-2010',
     'HINDUPUR',19000);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3: Selecting (Displaying) all rows from the EMPLOYEE_101 table.

```
SQL> SELECT * FROM EMPLOYEE_101;
```

EMP_NO	EMP_NAME	EMP_DOJ	EMP_CITY	EMP_SALARY
111	RAZAQ	1-Jan-90	ANANTAPURAMU	20000
112	RAMESH	2-Feb-89	HINDUPUR	25000
113	RADHIKA	4-Mar-85	DHARMAVARAM	18000
114	RAMANI	7-Feb-79	ANANTAPURAMU	15000
115	RAJESH	6-Apr-80	HUNDUPUR	17000
116	RAJU	5-May-00	ANANTAPURAMU	20000
117	RAJESH	3-Jun-05	DHARMAVARAM	16000
118	REKHA	2-Dec-10	HINDUPUR	19000

8 rows selected.

Step 4: Displaying number of employees working in each city.

```
SQL> SELECT EMP_CITY,COUNT(EMP_NO) FROM EMPLOYEE_101 GROUP BY
     EMP_CITY;
```

EMP_CITY	COUNT(EMP_NO)
ANANTAPURAMU	3
HINDUPUR	2
DHARMAVARAM	3

Step 5: Displaying rows in ascending order.

```
SQL> SELECT * FROM EMPLOYEE_101 ORDER BY EMP_NAME;
```

EMP_NO	EMP_NAME	EMP_DOJ	EMP_CITY	EMP_SALARY
113	RADHIKA	4-Mar-85	DHARMAVARAM	18000
115	RAJESH	6-Apr-80	HUNDUPUR	17000
117	RAJESH	3-Jun-05	DHARMAVARAM	16000
116	RAJU	5-May-00	ANANTAPURAMU	20000
114	RAMANI	7-Feb-79	ANANTAPURAMU	15000
112	RAMESH	2-Feb-89	HINDUPUR	25000
111	RAZAQ	1-Jan-90	ANANTAPURAMU	20000
118	REKHA	2-Dec-10	HINDUPUR	19000

8 rows selected.

Step 6 : Displaying which cities employees taking salary more than 50000.

```
SQL> SELECT EMP_CITY,SUM(EMP_SALARY) FROM EMPLOYEE_101 GROUP BY
     EMP_CITY HAVING SUM(EMP_SALARY)>50000;
```

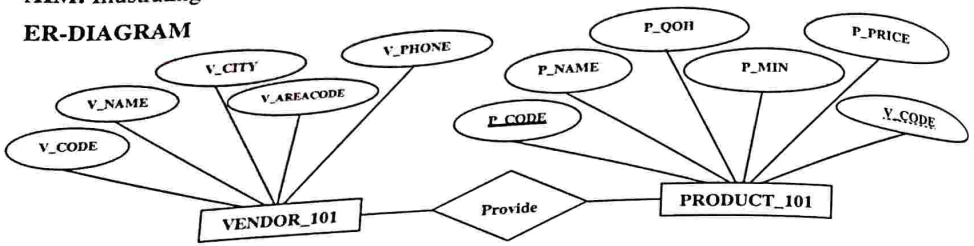
EMP_CITY	SUM(EMP_SALARY)
ANANTAPURAMU	55000
HINDUPUR	61000

LAB PROGRAM -6

6) Write SQL procedure to illustrate Virtual Tables (Views).

AIM: Illustrating Virtual Tables using VENDOR and PRODUCT tables.

ER-DIAGRAM



PROCEDURE

Step 1: Creating Vendor table.

```
SQL> CREATE TABLE VENDOR_101(V_CODE NUMBER(5),V_NAME
     VARCHAR2(20),V_CITY VARCHAR2(15),V_AREACODE VARCHAR2(4),
     V_PHONE NUMBER(10), PRIMARY KEY(V_CODE));
```

Table created.

Step 2: inserting rows into table VENDOR_101.

```
SQL> INSERT INTO VENDOR_101 VALUES (101, 'KAREEM',
     'ANANTAPURAMU', 'ATP', 9999888877);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (102,
     'KAMAL','ANANTAPURAMU', 'ATP', 8899889977);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (103,'KIRAN','TADIPATRI',
     'TDP', 8877997799);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(104,'KAJAL', 'TADIPARTI',
     'TDP',7373839399);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(105,'KESAV','HINDUPUR',
     'HDP',7374565656);
```

1 row created.

```
SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR',
     'HDP',7473656565);
```


1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3: Creating Product table.

```
SQL> CREATE TABLE PRODUCT_101(P_CODE VARCHAR2(10) PRIMARY
KEY,P_NAME VARCHAR2(20), P_QOH NUMBER(4),P_PRICE NUMBER(8,2),
P_MIN NUMBER(2), V_CODE NUMBER(5) REFERENCES VENDOR_101);
```

Table created.

Step 4: Inserting rows into the table PRODUCT_101.

```
SQL> INSERT INTO PRODUCT_101
VALUES('PCR','PROCESSOR',50,3200,10,101);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('MB','MOTHER
BOARD',35,2800,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('HD_1TB','HARD
DISK',30,4200,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('LED_18','LED
MONITORS',40,5800,10,106);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101
     VALUES('KB_MS','KEYBOARD_MOUSE_COMBO', 100,670,20,104);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101
     VALUES('PRTR','PRINTER',50,7000,10,101);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 5: Creating Virtual Table(View) to show V_CODE, V_NAME, P_NAME, P_PRICE, P_QOH, V_PHONE .

```
SQL>CREATE VIEW PROD_VIEW AS SELECT VENDOR_101.V_CODE, V_NAME,
P_NAME, P_PRICE, P_QOH, V_PHONE FROM VENDOR_101,PRODUCT_101
WHERE VENDOR_101.V_CODE=PRODUCT_101.V_CODE;
```

view created.

Step 6: Displaying rows from virtual table PROD_VIEW.

```
SQL> SELECT * FROM PROD_VIEW;
```

V_CODE	V_NAME	P_NAME	P_PRICE	P_QOH	V_PHONE
101	KAREEM	PROCESSOR	3200	50	9999888877
103	KIRAN	MOTHER BOARD	2800	35	8877997799
103	KIRAN	HARD DISK	4200	30	8877997799
106	KUMAR	LED MONITORS	5800	40	7473656565
104	KAJAL	KEYBOARD_MOUSE_COMBO	670	100	7373839399
101	KAREEM	PRINTER	7000	50	9999888877

6 rows selected.

LAB PROGRAM -7

7) Write SQL procedure to illustrate Relational Set Operators.

AIM: *Illustrating Relational Set Operators (UNION, UNION ALL, INTERSECT, MINUS) on VENDOR and PRODUCT tables.*

PROCEDURE

Step 1: Creating Vendor table.

```
SQL> CREATE TABLE VENDOR_101(V_CODE NUMBER(5),V_NAME
     VARCHAR2(20),V_CITY VARCHAR2(15),V_AREACODE VARCHAR2(4),
     V_PHONE NUMBER(10), PRIMARY KEY(V_CODE));
```

Table created.

Step 2: inserting rows into table VENDOR_101.

```
SQL> INSERT INTO VENDOR_101 VALUES (101, 'KAREEM',
     'ANANTAPURAMU', 'ATP', 9999888877);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (102,
     'KAMAL','ANANTAPURAMU', 'ATP', 8899889977);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (103,'KIRAN','TADIPATRI',
     'TDP', 8877997799);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(104,'KAJAL', 'TADIPARTI',
     'TDP',7373839399);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(105,'KESAV','HINDUPUR',
     'HDP',7374565656);
```

1 row created.

```
SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR',
     'HDP',7473656565);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3 : Creating Product table.

```
SQL> CREATE TABLE PRODUCT_101(P_CODE VARCHAR2(10) PRIMARY
     KEY,P_NAME VARCHAR2(20), P_QOH NUMBER(4),P_PRICE
```

```
NUMBER(8,2), P_MIN NUMBER(2), V_CODE NUMBER(5) REFERENCES
VENDOR_101);
```

Table created.

Step 4: Inserting rows into the table PRODUCT_101.

```
SQL> INSERT INTO PRODUCT_101
VALUES('PCR','PROCESSOR',50,3200,10,101);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('MB','MOTHER BOARD',
35,2800,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('HD_1TB','HARD DISK',
30,4200,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('LED_18','LED MONITORS',
40,5800,10,106);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101
     VALUES('KB_MS','KEYBOARD_MOUSE_COMBO', 100,670,20,104);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101
     VALUES('PRTR','PRINTER',50,7000,10,101);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 5: Implementing Relational Set operators.

1. UNION

```
SQL> SELECT V_CODE FROM VENDOR_101 UNION SELECT V_CODE FROM
PRODUCT_101;
```

V_CODE
101
102
103
104
105
106

6 rows selected.

2. UNION ALL

```
SQL> SELECT V_CODE FROM VENDOR_101 UNION ALL SELECT V_CODE
  FROM PRODUCT_101;
```

V_CODE
101
102
103
104
105
106
101
103
103
106
104
101

12 rows selected.

3. INTERSECT

```
SQL> SELECT V_CODE FROM VENDOR_101 INTERSECT SELECT V_CODE
  FROM PRODUCT_101;
```

V_CODE
101
103
104
106

4 rows selected.

4. MINUS

```
SQL> SELECT V_CODE FROM VENDOR_101 MINUS SELECT V_CODE FROM
  PRODUCT_101;
```

V_CODE
102
105

LAB PROGRAM -8

8) Write SQL Procedure to illustrate Joins.

AIM : Illustrating SQL Join operations on VENDOR and PRODUCT tables.

PROCEDURE

Step 1: Creating Vendor table.

```
SQL> CREATE TABLE VENDOR_101(V_CODE NUMBER(5),V_NAME
     VARCHAR2(20),V_CITY VARCHAR2(15),V_AREACODE VARCHAR2(4),
     V_PHONE NUMBER(10), PRIMARY KEY(V_CODE));
```

Table created.

Step 2: inserting rows into table VENDOR_101.

```
SQL> INSERT INTO VENDOR_101 VALUES (101, 'KAREEM',
     'ANANTAPURAMU', 'ATP', 9999888877);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (102,
     'KAMAL','ANANTAPURAMU', 'ATP', 8899889977);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (103,'KIRAN','TADIPATRI',
     'TDP', 8877997799);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(104,'KAJAL', 'TADIPARTI',
     'TDP',7373839399);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(105,'KESAV','HINDUPUR',
     'HDP',7374565656);
```

1 row created.

```
SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR',
     'HDP',7473656565);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3 : Creating Product table.

```
SQL> CREATE TABLE PRODUCT_101(P_CODE VARCHAR2(10) PRIMARY
     KEY,P_NAME VARCHAR2(20), P_QOH NUMBER(4),P_PRICE
     NUMBER(8,2), P_MIN NUMBER(2), V_CODE NUMBER(5) REFERENCES
     VENDOR_101);
```


Table created.

Step 4: Inserting rows into the table PRODUCT_101.

```
SQL> INSERT INTO PRODUCT_101
     VALUES('PCR','PROCESSOR',50,3200,10,101);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('MB','MOTHER
     BOARD',35,2800,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('HD_1TB','HARD
     DISK',30,4200,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('LED_18','LED
     MONITORS',40,5800,10,106);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101
     VALUES('KB_MS','KEYBOARD_MOUSE_COMBO', 100,670,20,104);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101
     VALUES('PRTR','PRINTER',50,7000,10,101);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 5: Performing join operations.

1. **Natural Join (Inner Join)**

Code for Oracle 8

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101  V,
PRODUCT_101  P WHERE V.V_CODE=P.V_CODE;
```

Code for Oracle 9i, 10g

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101  V,
PRODUCT_101;
```

V_CODE	V_NAME	P_NAME
101	KAREEM	PROCESSOR
103	KIRAN	MOTHER BOARD
103	KIRAN	HARD DISK
106	KUMAR	LED MONITORS

104	KAJAL	KEYBOARD_MOUSE_COMBO
101	KAREEM	PRINTER

6 rows selected.

2. Cross Join

Code for Oracle8

```
SQL> SELECT * FROM VENDOR_101,PRODUCT_101;
```

Code for Oracle 9i, 10g

```
SQL> SELECT * FROM VENDOR_101 CROSS JOIN PRODUCT_101;
```

It displays 36 rows (product of VENDER_101 and PRODUCT_101).

3. Left Outer Join

Code for Oracle8

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101  V,
   PRODUCT_101  P WHERE V.V_CODE=P.V_CODE(+);
```

Code for Oracle 9i,10g

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101  V
   LEFT  OUTER  JOIN PRODUCT_101  P  ON  V.V_CODE=P.V_CODE;
```

V_CODE	V_NAME	P_NAME
101	KAREEM	PROCESSOR
101	KAREEM	PRINTER
102	KAMAL	
103	KIRAN	MOTHER BOARD
103	KIRAN	HARD DISK
104	KAJAL	KEYBOARD_MOUSE_COMBO
105	KESAV	
106	KUMAR	LED MONITORS

8 rows selected.

4. Right Outer Join

Code for Oracle8

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101  V,
   PRODUCT_101  P WHERE V.V_CODE(+)=P.V_CODE;
```

Code for Oracle 9i,10g

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101  V
   RIGHT  OUTER  JOIN PRODUCT_101  P  ON  V.V_CODE=P.V_CODE;
```


V_CODE	V_NAME	P_NAME
101	KAREEM	PROCESSOR
103	KIRAN	MOTHER BOARD
103	KIRAN	HARD DISK
106	KUMAR	LED MONITORS
104	KAJAL	KEYBOARD_MOUSE_COMBO
101	KAREEM	PRINTER

6 rows selected.

5. Full Outer Join

Code for Oracle8

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101 V,
   PRODUCT_101 P WHERE V.V_CODE=P.V_CODE(+) UNION ALL SELECT
   V.V_CODE,V_NAME,P_NAME FROM VENDOR_101 V, PRODUCT_101 P
   WHERE V.V_CODE(+)=P.V_CODE;
```

Code for Oracle 9i,10g

```
SQL> SELECT  V.V_CODE,V_NAME,P_NAME FROM VENDOR_101  V  FULL
   OUTER  JOIN PRODUCT_101  P  ON  V.V_CODE=P.V_CODE;
```

V_CODE	V_NAME	P_NAME
101	KAREEM	PROCESSOR
101	KAREEM	PRINTER
102	KAMAL	
103	KIRAN	MOTHER BOARD
103	KIRAN	HARD DISK
104	KAJAL	KEYBOARD_MOUSE_COMBO
105	KESAV	
106	KUMAR	LED MONITORS
101	KAREEM	PROCESSOR
103	KIRAN	MOTHER BOARD
103	KIRAN	HARD DISK
106	KUMAR	LED MONITORS
104	KAJAL	KEYBOARD_MOUSE_COMBO
101	KAREEM	PRINTER

LAB PROGRAM -9

9) Write SQL procedure to illustrate Sub Queries.

AIM: Illustrating Sub Queries using Vendor and Product tables.

PROCEDURE

Step 1: Creating Vendor table.

```
SQL> CREATE TABLE VENDOR_101(V_CODE NUMBER(5),V_NAME
     VARCHAR2(20),V_CITY VARCHAR2(15),V_AREACODE VARCHAR2(4),
     V_PHONE NUMBER(10), PRIMARY KEY(V_CODE));
```

Table created.

Step 2: inserting rows into table VENDOR_101.

```
SQL> INSERT INTO VENDOR_101 VALUES (101, 'KAREEM',
     'ANANTAPURAMU', 'ATP', 9999888877);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (102,
     'KAMAL','ANANTAPURAMU', 'ATP', 8899889977);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES (103,'KIRAN','TADIPATRI',
     'TDP', 8877997799);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(104,'KAJAL', 'TADIPARTI',
     'TDP',7373839399);
```

1 row created.

```
SQL> INSERT INTO VENDOR_101 VALUES(105,'KESAV','HINDUPUR',
     'HDP',7374565656);
```

1 row created.

```
SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR',
    'HDP',7473656565);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3 : Creating Product table.

```
SQL> CREATE TABLE PRODUCT_101(P_CODE VARCHAR2(10) PRIMARY
     KEY,P_NAME VARCHAR2(20), P_QOH NUMBER(4),P_PRICE
     NUMBER(8,2), P_MIN NUMBER(2), V_CODE NUMBER(5) REFERENCES
     VENDOR_101);
```


Table created.

Step 4: Inserting rows into the table PRODUCT_101.

```
SQL> INSERT INTO PRODUCT_101
VALUES('PCR','PROCESSOR',50,3200,10,101);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('MB','MOTHER BOARD',
35,2800,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('HD_1TB','HARD DISK',
30,4200,10,103);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('LED_18','LED MONITORS',
40,5800,10,106);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101 VALUES('KB_MS',
'KEYBOARD_MOUSE_COMBO', 100,670,20,104);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_101
VALUES('PRTR','PRINTER',50,7000,10,101);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 5: Implementing Sub Queries.

1. Display vendors who are providing products.

```
SQL> SELECT V_CODE,V_NAME,V_PHONE FROM VENDOR_101 WHERE V_CODE
IN(SELECT V_CODE FROM PRODUCT_101);
```

V_CODE	V_NAME	V_PHONE
101	KAREEM	9999888877
103	KIRAN	8877997799
104	KAJAL	7373839399
106	KUMAR	7473656565

4 rows selected.

2. Display products with a price greater than or equal to the average product price.

```
SQL> SELECT P_NAME,P_PRICE FROM PRODUCT_101 WHERE P_PRICE >=
(SELECT AVG(P_PRICE) FROM PRODUCT_101);
```

P_NAME	P_PRICE
HARD DISK	4200
LED MONITORS	5800
PRINTER	7000

3 rows selected.

3. Delete the products that are provided by vendors with area code 'HDP'.

```
SQL> DELETE FROM PRODUCT_101 WHERE V_CODE IN (SELECT V_CODE
FROM VENDOR_101 WHERE V_AREACODE='HDP');
```

1 row deleted.

```
SQL> SELECT * FROM PRODUCT_101;
```

P_CODE	P_NAME	P_QOH	P_PRICE	P_MIN	V_CODE
PCR	PROCESSOR	50	3200	10	101
MB	MOTHER BOARD	35	2800	10	103
HD_1TB	HARD DISK	30	4200	10	103
KB_MS	KEYBOARD_MOUSE_COMBO	100	670	20	104
PRTR	PRINTER	50	7000	10	101

5 rows selected.

LAB PROGRAM - 10

10) Write about SQL predefined functions.

AIM: *Illustrating SQL predefined functions.*

PROCEDURE

Step 1: *Number functions.*

1. ABS

```
SQL> SELECT ABS(-500) FROM DUAL;
```

ABS(-500)
500

2. CEIL

```
SQL> SELECT CEIL(123.55) FROM DUAL;
```

CEIL(123.55)
124

3. FLOOR

```
SQL> SELECT FLOOR(123.55) FROM DUAL;
```

FLOOR(123.55)
123

4. MOD

```
SQL> SELECT MOD(10,5) FROM DUAL;
```

MOD(10,5)
0

```
SQL> SELECT MOD(10,3) FROM DUAL;
```

MOD(10,5)
1

5. POWER

```
SQL> SELECT POWER(2,4) FROM DUAL;
```

POWER(2,4)
16

6. ROUND

```
SQL> SELECT ROUND(123.55,0),ROUND(123.55,1) FROM DUAL;
```

ROUND(123.55,0)	ROUND(123.55,1)
124	123.6

7. SQRT

```
SQL> SELECT SQRT(64) FROM DUAL;
```

SQRT(64)
8

8. TRUNC

```
SQL> SELECT TRUNC(123.55,0),TRUNC(123.55,1) FROM DUAL;
```

TRUNC(123.55,0)	TRUNC(123.55,1)
123	123.5

Step 2: Character functions.

1. INITCAP

```
SQL> SELECT INITCAP('anantapuramu') FROM DUAL;
```

INITCAP('ANANTAPURAMU')
Anantapuramu

2. LENGTH

```
SQL> SELECT LENGTH('ANANTAPURAMU') FROM DUAL;
```

LENGTH('ANANTAPURAMU')
12

3. SUBSTR

```
SQL> SELECT SUBSTR('ANANTAPURAMU',3,3) FROM DUAL;
```

SUBSTR
ANT

4. LOWER

```
SQL> SELECT LOWER('ANANTAPURAMU') FROM DUAL;
```

LOWER('ANANT APURAMU')
anantapuramu

5. UPPER

```
SQL> SELECT UPPER('anantapuramu') FROM DUAL;
```

UPPER('ANANT APURAMU')
ANANTAPURAMU

Step 3: Date functions.

1. ADD_MONTHS

```
SQL> SELECT ADD_MONTHS('15-JAN-2014',5) FROM DUAL;
```

ADD_MONTH
15-JUN-14

2. LAST_DAY

```
SQL> SELECT LAST_DAY('15-JAN-2014') FROM DUAL;
```

LAST_DAY
31-JAN-14

3. MONTHS_BETWEEN

```
SQL> SELECT MONTHS_BETWEEN('15-MAR-2014','15-JAN-2014')
FROM DUAL;
```

MONTHS_BETWEEN('15-MAR-2014','15-JAN-2014')
2

4. NEXT_DAY

```
SQL> SELECT NEXT_DAY('20-JAN-2014','MON') FROM DUAL;
```

NEXT_DAY
27-JAN-14

LAB PROGRAM -11

11) Write a PL/SQL block of code to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in a table AREAS.

AIM : Creating PL/SQL block of code to insert rows into AREAS table.

PROCEDURE

Step 1: Creating AREAS table.

```
SQL> CREATE TABLE AREAS(RADIUS NUMBER(5),AREA NUMBER(10,2));
```

Table created.

Step 2: Writing PL/SQL code.

```
DECLARE
     PI CONSTANT NUMBER(4,2):=3.14;
     RADIUS NUMBER(5);
     AREA NUMBER(14,2);
BEGIN
     RADIUS :=3;
     WHILE RADIUS <=7
     LOOP
          AREA := PI*POWER(RADIUS,2);
          INSERT INTO AREAS VALUES(RADIUS, AREA);
          RADIUS:=RADIUS+1;
     END LOOP;
END;
/
```

PL/SQL procedure successfully completed.

Step 3: Displaying rows from the table AREAS.

```
SQL> SELECT * FROM AREAS;
```

RADIUS	AREA
3	28.26
4	50.24
5	78.5
6	113.04
7	153.86

LAB PROGRAM -12

12) Write PL/SQL code to illustrate Triggers using Student table.

AIM: *Illustrating triggers using student table.*

PROCEDURE

Step 1: Creating student table.

```
SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4) PRIMARY KEY ,
S_NAME VARCHAR2(20), S_MARKS1 NUMBER(3), S_MARKS2 NUMBER(3),
S_MARKS3 NUMBER(3), TOTAL NUMBER(3),AVERAGE NUMBER(5,2),RESULT
VARCHAR2(6));
```

Table created.

Step 2: Creating trigger to update columns TOTAL, AVERAGE, RESULT after inserting a row.

```
SQL> CREATE OR REPLACE TRIGGER STUDENT_TRG AFTER INSERT ON
     STUDENT_101
BEGIN
UPDATE STUDENT_101 SET TOTAL=S_MARKS1+S_MARKS2+S_MARKS3;
UPDATE STUDENT_101 SET AVERAGE=TOTAL/3;
UPDATE STUDENT_101 SET RESULT='PASS' WHERE S_MARKS1>=35 AND
     S_MARKS2>=35 AND S_MARKS3>=35;
UPDATE STUDENT_101 SET RESULT='FAIL' WHERE S_MARKS1<35 OR
     S_MARKS2<35 OR S_MARKS3<35;
END;
/
```

Trigger created.

Step 3 : Inserting 5 rows into STUDENT_101 table.

```
SQL> INSERT INTO STUDENT_101 VALUES(101,'RAVI',66,88,77,0,0,' ');
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(102,'HAMEED',56,78,67,0,0,' ');
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(103,'THARUN',76,58,77,0,0,' ');
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(104,'PHANI',26,68,87,0,0,' ');
```

1 row created.

```
SQL> INSERT INTO STUDENT_101 VALUES(105,'ANITHA',86,67,57,0,0,' ');
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 4 : Selecting (Displaying) all rows from the STUDENT_101 table.

```
SQL> SELECT * FROM STUDENT_101;
```

S_NO	S_NAME	S_MARKS1	S_MARKS2	S_MARKS3	TOTAL	AVERAGE	RESULT
101	RAVI	66	88	71	231	77	PASS
102	HAMEED	56	78	67	201	67	PASS
103	THARUN	76	58	77	211	70.33	PASS
104	PHANI	26	68	87	181	60.33	FAIL
105	ANITHA	86	67	57	210	70	PASS

LAB PROGRAM -13

13) Write a PL/SQL block of code to create a trigger for the Employee table to display the salary difference between the old values and new values before performing INSERT or UPDATE operations.

AIM : Create a trigger for the Employee table to display the salary differences.

PROCEDURE

Step 1: Creating Employee table.

```
SQL> CREATE TABLE EMPLOYEE_101(EMP_NO   NUMBER(4),EMP_NAME
       VARCHAR2(20), EMP_DOJ   DATE,EMP_CITY  VARCHAR2(15),
       EMP_SALARY     NUMBER(9,2));
```

Step 2: Creating Trigger.

```
SQL> SET SERVER OUT ON;
SQL> CREATE  OR  REPLACE  TRIGGER  EMP_SALARY_CHANGE  BEFORE
       UPDATE OR INSERT ON EMPLOYEE
FOR EACH ROW WHEN (NEW.EMP_NO>0)
DECLARE
     SAL_DIFF NUMBER(8,2);
BEGIN
     SAL_DIFF:= :NEW.EMP_SALARY-:OLD.EMP_SALARY;
     DBMS_OUTPUT.PUT_LINE('EMPLOYEE NO. : '|| :NEW.EMP_NO);
     DBMS_OUTPUT.PUT_LINE('OLD SALARY : ' || :OLD.EMP_SALARY);
     DBMS_OUTPUT.PUT_LINE('NEW SALARY : ' || :NEW.EMP_SALARY);
     DBMS_OUTPUT.PUT_LINE('SALARY DIFFERENCE : ' ||SAL_DIFF);
END;
/
```

Trigger created.

Step 3: inserting a row into EMPLOYEE_101 table.

```
SQL> INSERT INTO EMPLOYEE_101 VALUES(119,'RAJINI','02-MAR-1998',
      'TADIPATRI',25000)
```

EMPLOYEE NO. : 119
OLD SALARY :
NEW SALARY : 25000
SALARY DIFFERENCE :

Step 4: Updating column EMP_SALARY.

```
SQL> UPDATE EMPLOYEE_101  SET  EMP_SALARY=EMP_SALARY+1000 WHERE
     EMP_NO=119;
```

EMPLOYEE NO. : 119
OLD SALARY : 25000
NEW SALARY : 26000
SALARY DIFFERENCE : 1000
1 row updated.

LAB PROGRAM -14

14) Write a PL/SQL Stored Procedure to insert rows into Employee table.

AIM : PL/SQL Stored Procedure to insert rows into Employee table.

PROCEDURE

Step 1: Creating Employee table.

```
SQL> CREATE TABLE EMPLOYEE_102(EMP_NO    NUMBER(4),EMP_NAME
VARCHAR2(20), EMP_DOJ    DATE,EMP_CITY   VARCHAR2(15),  EMP_SALARY
NUMBER(9,2));
```

Step 2: Creating Stored Procedure.

```
SQL> CREATE OR REPLACE PROCEDURE EMP_ADD(E_NO   NUMBER,E_NAME
VARCHAR, E_DOJ   DATE,E_CITY   VARCHAR,  E_ SALARY      NUMBER)
AS
BEGIN
INSERT INTO EMPLOYEE_102 VALUES(E_NO,E_NAME,E_DOJ,
      E_CITY,E_SALARY);
DBMS_OUTPUT.PUT_LINE('Employee '||E_NAME||' is Added.');
END;
/
```

Procedure created.

Step 3: Executing Stored Procedure.

```
SQL> EXEC EMP_ADD(110,'RAMANA','02-MAR-1999', 'TADIPATRI',35000);
```

Employee RAMANA is Added.

PL/SQL procedure successfully completed.

Step 4: Selecting (Displaying) all rows from the EMPLOYEE_102 table.

```
SQL> SELECT * FROM EMPLOYEE_102;
```

EMP_NO	EMP_NAME	EMP_DOJ	EMP_CITY	EMP_SALARY
110	RAMANA	02-MAR-99	TADIPATRI	35000

LAB PROGRAM -15

15) Write a PL/SQL Stored Procedure to update Product table.

AIM : PL/SQL Stored Procedure to update column P_DISCOUNT of the PRODUCT table.

PROCEDURE

Step 1: Creating Product table.

```
SQL> CREATE TABLE PRODUCT_102(P_CODE VARCHAR2(10) PRIMARY
     KEY,P_NAME VARCHAR2(20), P_QOH NUMBER(4),P_PRICE
     NUMBER(8,2), P_MIN NUMBER(2), P_DISCOUNT NUMBER(4,2));
```

Table created.

Step 2: Inserting rows into the table PRODUCT_102.

```
SQL> INSERT INTO PRODUCT_102
     VALUES('PCR','PROCESSOR',50,3200,10,0.5);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102
    VALUES('MB','MOTHER BOARD',35,2800,10,0.75);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102
    VALUES('HDD','HARD DISK',30,4200,10,1.2);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102
    VALUES('MNTR','LED MONITORS',40,5800,10,0.8);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102
    VALUES('PRTR','PRINTER',50,7000,10,1);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3: Displaying rows.

```
SQL> SELECT * FROM PRODUCT_102;
```

P_CODE	P_NAME	P_QOH	P_PRICE	P_MIN	P_DISCOUNT
PCR	PROCESSOR	50	3200	10	.5
MB	MOTHER BOARD	35	2800	10	.75
HDD	HARD DISK	30	4200	10	1.2
MNTR	LED MONITORS	40	5800	10	.8
PRTR	PRINTER	50	7000	10	1

Step 4: Creating Stored Procedure to add additional 5% discount for all products when the P_QOH is more than or equal to thrice of the minimum quantity.

```
SQL> CREATE OR REPLACE PROCEDURE PROD_DISCOUNT
AS
BEGIN
UPDATE PRODUCT_102 SET P_DISCOUNT=P_DISCOUNT+0.05 WHERE
    P_QOH>=P_MIN*3;
DBMS_OUTPUT.PUT_LINE(' Table Updated');
END;
/
```

Procedure created.

Step 5: Executing Stored Procedure.

```
SQL> EXEC PROD_DISCOUNT;
```

Table Updated

PL/SQL procedure successfully completed.

Step 6: Displaying rows.

```
SQL> SELECT * FROM PRODUCT_102;
```

P_CODE	P_NAME	P_QOH	P_PRICE	P_MIN	P_DISCOUNT
PCR	PROCESSOR	50	3200	10	.55
MB	MOTHER BOARD	35	2800	10	.8
HDD	HARD DISK	30	4200	10	1.25
MNTR	LED MONITORS	40	5800	10	.85
PRTR	PRINTER	50	7000	10	1.05

LAB PROGRAM -16

16) Write a PL/SQL Stored Procedure to illustrate Cursor.

AIM : PL/SQL Stored Procedure to Cursor using PRODUCT table.

PROCEDURE

Step 1: Creating Product table.

```
SQL> CREATE TABLE PRODUCT_102(P_CODE VARCHAR2(10) PRIMARY
     KEY,P_NAME VARCHAR2(20), P_QOH NUMBER(4),P_PRICE
     NUMBER(8,2), P_MIN NUMBER(2), P_DISCOUNT NUMBER(4,2));
```

Table created.

Step 2: Inserting rows into the table PRODUCT_102.

```
SQL> INSERT INTO PRODUCT_102
    VALUES('PCR','PROCESSOR',50,3200,10,0.5);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102 VALUES('MB','MOTHER
   BOARD',35,2800,10,0.75);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102 VALUES('HDD','HARD
   DISK',30,4200,10,1.2);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102 VALUES('MNTR','LED
   MONITORS',40,5800,10,0.8);
```

1 row created.

```
SQL> INSERT INTO PRODUCT_102
    VALUES('PRTR','PRINTER',50,7000,10,1);
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

Step 3: Displaying rows.

```
SQL> SELECT * FROM PRODUCT_102;
```

P_CODE	P_NAME	P_QOH	P_PRICE	P_MIN	P_DISCOUNT
PCR	PROCESSOR	50	3200	10	.5
MB	MOTHER BOARD	35	2800	10	.75
HDD	HARD DISK	30	4200	10	1.2
MNTR	LED MONITORS	40	5800	10	.8
PRTR	PRINTER	50	7000	10	1

Step 4: Creating Stored Procedure to list all products that have P_QOH is greater than the average of all products P_QOH.

```
SQL>CREATE OR REPLACE PROCEDURE PRO_CURSOR IS
      CODE PRODUCT_102.P_CODE%TYPE;
      NAME PRODUCT_102.P_NAME%TYPE;
      CURSOR P_CURSOR IS
            SELECT P_CODE,P_NAME FROM PRODUCT_102 WHERE
                  P_QOH>=(SELECT AVG(P_QOH) FROM PRODUCT_102);
      BEGIN
      DBMS_OUTPUT.PUT_LINE('PRODUCT WITH P_QOH > AVG(P_QOH)');
      DBMS_OUTPUT.PUT_LINE('-----------------------------------');
      DBMS_OUTPUT.PUT_LINE('PRODUCT_CODE     PRODUCT_NAME');
      DBMS_OUTPUT.PUT_LINE('-----------------------------------');
      OPEN P_CURSOR;
      LOOP
            FETCH P_CURSOR INTO CODE,NAME;
            EXIT WHEN P_CURSOR%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('    '||CODE||'   '||NAME);
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('-----------------------------------');
      DBMS_OUTPUT.PUT_LINE('TOTAL NUMBER OF PRODUCTS PROCESSED:
            '||P_CURSOR%ROWCOUNT);
      CLOSE P_CURSOR;
END;
/
```

Procedure created.

```
SQL> SET SERVEROUT ON;
SQL> EXEC PRO_CURSOR;
```

PRODUCT WITH P_QOH > AVG(P_QOH)

PRODUCT_CODE PRODUCT_NAME

PCR PROCESSOR
PRTR PRINTER

TOTAL NUMBER OF PRODUCTS PROCESSED : 2
PL/SQL procedure successfully completed.