S.NO	PROGRAMS LIST
1	Write a SQL queries to demonstrate DDL & DML commands with Master Child Table's relationships.
2	Write a SQL queries to perform SELECT, PROJECT, SORTING and SET OPERATORS in a table's.
3	Write a SQL program to demonstrate AGGREGATE functions.
4	Write SQL queries for implementing SQL JOINS with given Tables. • Inner join(=,>,<) • Outer join(left outer, right outer, full outer)
5	Create types Locations,Person,Qualification And Create An Object Type Department which Contains Information About An Employee with Above Types
6	Demonstrate The use of V ARRAYS in SQL.
7	Write a Pl/SQL Program To Count the number of records using CURSORS.
8	Write a PL/SQL code to calculate the total salary of first N records of employee table. The value of N is passed to CURSOR AS PARAMETER
9	Write a PL/SQL program to print the Electricity Bill using cursors in the format given below. Electricity bill for the month of…….. …… Meter no category A/C/I/D Consumer Name Address Current Reading Previous reading Billed units unit cost Arrears Total Amount Payable Last Date without fine Last Date with fine
10	Design PL/SQL program to calculate ranks for EMCET/ICET (use correct rules) using CURSORS and generate SQL *PLUS Report for it with ranks and hall ticket numbers.
11	Write a code in PL/SQL to implement a Trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.
12	Write a PL/SQL program to find the factorial number using FUNCTIONS.
13	Write a PL/SQL program for OVERDRAFT Exception for a sample banking application.
14	Design a statistical PACKAGE with FUNCTIONS. Mean, Standard Deviation and use it in PL/SQL program.

15	Write a PL/SQL program to demonstrate STORED PROCEDURE.

PROGRAM 1 – SQL COMMANDS WITH MASTER CHILD

TABLE'S INPUT:

1. Creating Master Table:

SQL> create table dept(deptno number(3) primary key, dname varchar(10) not null, loc varchar(10) null); Table created.

SQL>descdept;

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(3)
DNAME	NOT NULL	VARCHAR2(20)
LOC		VARCHAR2(15)

2. Inserting Values Into Master Table:

SQL> insert into dept values(54,'production','block-4'); 1 row created.

SQL> insert into dept values(27,'sales','block-6'); 1 row created.

SQL> insert into dept values(90,'marketing','block-9'); 1 row created.

3. Selecting All Rows From Master Table:

SQL> select * from dept;

DEPTNO	DNAME	LOC
54	production	block-4
27	sales	block-6
90	marketing	block-9

4. Creating CHILD Table:

SQL> create table emp(empno number(4), enamevarchar(10) not null, desgvarchar(10), mgr number(4),hiredate date, sal number(7,2), deptno number(4)references dept(deptno)); Table created.

SQL>descemp;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME	NOT NULL	VARCHAR2(10)
DESG		VARCHAR2(10)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
DEPTNO		NUMBER(4)

5. Inserting Values Into Child Table:

```
SQL> insert into emp
values(&empno,'&ename','&desg',&mgr,'&hiredate',&salary,&deptno); Enter value for
empno: 101
Enter value for ename: raj
Enter value for desg: clerk
Enter value for mgr: 1234
Enter value for hiredate: 01-jul-2024
Enter value for salary:
42000 Enter value for
deptno: 27
old   1: insert into emp values(&empno,'&ename','&desg',&mgr,'&hiredate',&salary,&deptno)
new   1: insert into emp values(101,'rani','clerk',1234,'01-jul-
2024',42000,27) 1 row created.

SQL> /
Enter value for empno: 102
Enter value for ename: raghu
Enter value for desg: manager
Enter value for mgr: 2345
Enter value for hiredate: 26-mar-1997
Enter value for salary:
57000 Enter value for
deptno: 54
old   1: insert into emp
values(&empno,'&ename','&desg',&mgr,'&hiredate',&salary,&deptno) new   1:
insert into emp values(102,'hari','manager',2345,'26-mar-1997',57000,54) 1 row
created.
SQL> /
Enter value for empno: 103
Enter value for ename: seema
Enter value for desg: manager
Enter value for mgr: 3456
Enter value for hiredate: 19-dec-2000
Enter value for salary:
50000 Enter value for
deptno: 90
old   1: insert into emp
values(&empno,'&ename','&desg',&mgr,'&hiredate',&salary,&deptno) new   1:
insert into emp values(103,'uma','manager',3456,'19-dec-2000',50000,90) 1 row
created.
SQL> /
Enter value for empno: 104
Enter value for ename: ranga
Enter value for desg: manager
```

Enter value for mgr: 4567
Enter value for hiredate: 19-dec-200
Enter value for salary:
50000 Enter value for
deptno: 90
old 1: insert into emp
values(&empno,'&ename','&desg',&mgr,'&hiredate',&salary,&deptno) new 1:
insert into emp values(104,'noortaj','manager',4567,'19-dec-200',50000,90) 1 row
created.

6. Selecting All Rows From Child Table:

SQL> select * from emp;

EMPNO	ENAME	DESG	MGR	HIREDATE	SAL	DEPTNO
101	raj	clerk	1234	01-JUL-24	42000	27
102	raghu	manager	2345	26-MAR-97	57000	54
103	seema	manager	3456	19-DEC-00	50000	90
104	ranga	manager	4567	19-DEC-00	50000	90

OUTPUT:

➢ **Query's Evaluation:**

1. Adding primary key to emp table through "alter" query . SQL> alter table emp add primary key(empno); Table altered.

2. Add a new column to emp table

SQL> alter table emp add commnumber(7,2); Table altered.

SQL>descemp;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME	NOT NULL	VARCHAR2(10)
DESG		VARCHAR2(10)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
DEPTNO		NUMBER(4)
COMM		NUMBER(7,2)

3. updating Commission values to all rows in emp table. SQL> update emp set comm=emp.sal*0.25; 5 rows updated.

SQL> select * from emp;

EMPNO	ENAME	DESG	MGR	HIREDATE	SAL	DEPTNO	COMM
101	rajclerk	1234		01-JUL-24	42000	27	10500
102	raghumanager	2345		26-MAR-97	57000	54	14250
103	seema	manager	3456	19-DEC-00	50000	90	12500
104	ranga	manager	4567	19-DEC-00	50000	90	12500

4. Dropping commission column from emp table. SQL>alter tableemp drop column comm; Table altered.

SQL>descemp;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME	NOT NULL	VARCHAR2(10)
DESG		VARCHAR2(10)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
DEPTNO		NUMBER(4).

5. Deleting particular record from emp table SQL> delete from emp where empno=1234;

1 row deleted.

6. Renaming old column name to new column name SQL> alter table emp rename column ename to empname;

Table altered.

SQL>descemp;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
EMPNAME	NOT NULL	VARCHAR2(10)
DESG		VARCHAR2(10)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
DEPTNO		NUMBER(4)

SQL>
commit;
Commit
complete.

PROGRAM 2 – SELECTION, PROJECTION ,SET OPERATIONS & SORTINGINPUT:

1. Creating Employees Table:

SQL> create table employees (employee_idvarchar (20) primary key, employee_namevarchar(100), department varchar(50), salary decimal(10, 2)); Table created.

SQL>desc employees;

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	VARCHAR2(20)
EMPLOYEE_NAME		VARCHAR2(100)
DEPARTMENT		VARCHAR2(50)
SALARY		NUMBER(10,2)

2. Inserting sample data into "Employees" Table:

SQL> INSERT INTO employees VALUES (1, 'Doe', 'IT', 60000.00); 1 row created.

SQL> INSERT INTO employees VALUES (2, 'Smith', 'HR', 55000.00); 1 row created.

SQL> INSERT INTO employees VALUES(3, 'Johnson', 'IT', 70000.00); 1 row created.

SQL> INSERT INTO employees VALUES (4, 'Brown', 'Sales', 48000.00); 1 row created.

SQL> INSERT INTO employees VALUES (5, 'Lee', 'IT', 62000.00); 1 row created.

3.Selecting all rows from "Employees" Table

SQL> select * from employees;

EMPLOYEE_ID	EMPLOYEE_NAME	DEPARTMENT	SALARY
1	Doe	IT	60000.00
2	Smith	HR	55000.00
3	Johnson	IT	70000.00
4	Brown	Sales	48000.00
5	Lee	IT	62000.00

4. Creating Managers Table:

SQL> create table managers (manager_id number(5) primary key,manager_namevarc
har(100), department varchar(50),salary

decimal(10, 2)); Table created.

SQL>desc managers;

Name	Null?	Type
MANAGER_ID	NOT NULL	NUMBER(5)
MANAGER_NAME		VARCHAR2(100)
DEPARTMENT		VARCHAR2(50)
SALARY		NUMBER(10,2)

5. Inserting sample data into "Managers" Table:

SQL> INSERT INTO managers VALUES(101, 'Alice ', 'HR',

80000.00); 1 row created.

SQL> INSERT INTO managers VALUES (102, 'Bob', 'Finance',

90000.00); 1 row created.

SQL> INSERT INTO managers VALUES (103, 'Catherine', 'IT',
85000.00); 1 row created.

6. Selecting all rows from "Managers" Table

SQL> select * from managers;

MANAGER_ID	MANAGER_NAME	DEPARTMENT	SALARY
101	Alice	HR	80000
102	Bob	Finance	90000
103	Catherine	IT	85000

OUTPUT:

➢ **Query's Evaluation:**

1.Selection: Select employees from 'employees' table who have a salary greater than 50000

SQL> SELECT employee name, department, salary FROM employees WHERE salary > 50000;

EMPLOYEE_NAME	DEPARTMENT	SALARY
Doe	IT	60000
Smith	HR	55000
Johnson	IT	70000
Lee	IT	62000

2.Projection:Select only 'employee name' and 'department' columns from 'managers' table

SQL> SELECT manager name, department FROM managers;

MANAGER_NAMEDEPARTMENT

Alice	HR
Bob	Finance
Catherine	IT

3.Sorting: Select all employees from 'employees' table and sort them by 'salary' in descending order

SQL> SELECT * FROM employees ORDER BY salary DESC;

EMPLOYEE_ID	EMPLOYEE_NAME	DEPARTMENT	SALARY
3	Johnson	IT	70000
5	Lee	IT	62000
1	Doe	IT	60000
2	Smith	HR	55000
4	Brown	Sales	48000

4.Set Operators: Union of employees and managers to get all unique names across both tables SQL> SELECT employee_name, department FROM employees UNION SELECT manager_name, department FROM managers;

EMPLOYEE_NAME	DEPARTMENT
Alice	HR
Bob	Finance
Catherine	IT
Lee	IT
Brown	Sales
Smith	HR
Doe	IT
Johnson	IT

5.Logical Operators: Select employees from 'employees' table who are either in 'IT' department or have a salary greater than 60000

SQL> SELECT * FROM employees WHERE department = 'IT' OR salary > 60000;

EMPLOYEE_ID	EMPLOYEE_NAME	DEPARTMENT	SALARY
1	Doe	IT	60000
3	Johnson	IT	70000
5	Lee	IT	62000

PROGRAM 3 – AGGREGATE FUNCTIONS AND GROUP BY & HAVING CLAUSE

INPUT:

1. Creating Sales Table:

SQL> CREATE TABLE sales (transaction_id number(5) PRIMARY KEY, product_name VARCHAR(100), category VARCHAR(50),quantity number(5),unit_price number (10,2),transaction_date DATE); Table created.

SQL>desc sales;

Name	Null?	Type
TRANSACTION_ID	NOT NULL	NUMBER(5)
PRODUCT_NAME		VARCHAR2(100)
CATEGORY		VARCHAR2(50)
QUANTITY		NUMBER(5)
UNIT_PRICE		NUMBER(10,2)
TRANSACTION_DATE		DATE

2.Insert sample data into 'sales' table

SQL> INSERT INTO sales VALUES(1, 'PC', 'Electronics', 31,200.00, '23-jan-2024'); 1 row created.
SQL> INSERT INTO sales VALUES(2, 'Stool', 'Furniture', 5, 80.00, '12-jan-2023'); 1 row created.
SQL> INSERT INTO sales VALUES (3, 'Smart Fan ', 'Electronics', 2, 800.00, '15-jan-2023'); 1 row created.
SQL> INSERT INTO sales VALUES (4, 'Desk', 'Furniture', 1, 150.00, '18-jan-2023'); 1 row created.
SQL> INSERT INTO sales VALUES (5, 'Speaker', 'Electronics', 4, 150.00, '20-jan-2023'); 1 row created.

3. Selecting all rows from “Sales” table.

SQL> select * from sales;

TRANSACTION_ID	PRODUCT_NAME	CATEGORY	QUANTITY	UNIT_PRICE	TRANSACTION_DATE
1	PC	Electronics	3	1200	23-JAN-24
2	Stool	Furniture	5	80	12-JAN-23
3	Smart Fan	Electronics	2	800	15-JAN-23
4	Desk	Furniture	1	150	18-JAN-23
5	Speaker	Electronics	4	150	20-JAN-23

OUTPUT:

➢ **Query's Evaluation:**

1. COUNT: Count the number of transactions

```
SQL> SELECT COUNT(*) AS total_transactions FROM sales;
TOTAL_TRANSACTIONS
------------------
                 5
```

2. SUM: Calculate the total quantity of products sold

```
SQL> SELECT SUM(quantity) AS total_quantity_sold FROM sales;
TOTAL_QUANTITY_SOLD
-------------------
                  15
```

3. AVG: Calculate the average unit price of products sold

```
SQL> SELECT AVG(unit_price) AS average_unit_price FROM sales;

AVERAGE_UNIT_PRICE
--------
--------
--
476
```

4. MAX: Find the highest unit price among products

```
SQL> SELECT MAX(unit_price) AS highest_unit_price FROM sales;

HIGHEST_UNIT_PRICE
----------
--------
1200
```

5. MIN: Find the lowest unit price among products

```
SQL> SELECT MIN(unit_price) AS lowest_unit_price FROM sales;

LOWEST_UNIT_PRICE
-----------------
               80
```

6. GROUP BY: Calculate total sales amount per category

```
SQL> SELECT category, SUM(quantity * unit_price) AS total_sales_amount
FROM sale s GROUP BY category;
CATEGORY                                TOTAL_SALES_AMOUNT
--------------------------------------------------- ------------------
 Electronics                                    5800
 Furniture                                       550
```

7. HAVING: Show categories with total sales amount greater than 1000

```
SQL> SELECT category, SUM(quantity * unit_price) AS total_sales_amount  FROM sales
GROUP BY category
```

HAVING SUM(quantity * unit_price) > 1000;

CATEGORY	TOTAL_SALES_AMOUNT
Electronics	5800

8. DISTINCT: List distinct categories of products sold

SQL> SELECT DISTINCT category FROM sales;

CATEGORY

---------- Electronics

Furniture

9. Nested Aggregate Functions: Find the average quantity of products sold per category

SQL> SELECT category, AVG(quantity) AS average_quantity_sold FROM sales GROUP BY category;

CATEGORY	AVERAGE_QUANTITY_SOLD
---------- Electronics	3
Furniture	3

10. Combination of Aggregate Functions: Calculate total transactions, total quantity sold, and average unit price

SQL> SELECT COUNT(*) AS total_transactions, SUM(quantity) AS total_quantity_sold, AVG(unit_price) AS average_unit_price FROM sales;

TOTAL_TRANSACTIONS	TOTAL_QUANTITY_SOLD	AVERAGE_UNIT_PRICE
5	15	476

PROGRAM 4 – JOIN

OPERATIONSINPUT:

1. Creating dept1 Table:

SQL> create table dept1(enonumber(3), enamevarchar(10), deptno number(3),sal number(7,2), locvarchar(5), mob number(10), desgvarchar(5)); Table created.

SQL>desc dept1;

Name	Null?	Type
---------- ENO	NUMBER(3)	
ENAME		VARCHAR2(10)
DEPTNO		NUMBER(3)
SAL		NUMBER(7,2)
LOC		VARCHAR2(5)
MOB		NUMBER(10)
DESG		VARCHAR2(5)

2.Inserting sample data into dept1 table

SQL> insert into dept1 values(101,'kohili',01,45000 ,'hyd',9968335420, 'mgr');

1 row created .

SQL> insert into dept1 values(102,'dhawan',02,50000 ,'bngl',9836254107, 'gmgr');

1 row created.

SQL> insert into dept1 values(103,'sachin',03,60000 ,'bngl',7896554986, 'amgr');

1 row created.

SQL> insert into dept1 values(104,'dravid',04,55000 ,'tpt',8919090999, 'mgr');

1 row created.

3. Selecting all rows from dept1table

SQL> select * from dept1;

ENO	ENAME	DEPTNO	SAL	LOC	MOB	DESG
101	kohili	01	45000	hyd	9968335420	mgr
102	dhawan	02	50000	bngl	9836254107	gmgr
103	sachin	03	60000	tpt	7896554986	amgr
104	dravid	04	55000	chnnai	8919090999	mgr

4. Creating Emp1Table:

SQL> create table emp1(enonumber(3), enamevarchar(10), deptno number(3),sal number(6), locvarchar(5), mob number(10), desgvarchar(5));

Table created.

SQL>desc emp1;

Name	Null?	Type
ENO		NUMBER(3)
ENAME		VARCHAR2(10)
DEPTNO		NUMBER(3)
SAL		NUMBER(6)
LOC		VARCHAR2(5)
MOB		NUMBER(10)
DESG		VARCHAR2(5)

5.Inserting sample data into 'emp1' table

SQL> insert into emp1 values(111,'rajani',01,50000 ,'bngl',9703288493, 'gmgr');

1 row created.

SQL> insert into emp1 values(112,'shobana',02,45000 ,'hyd',9705185164, 'mgr');

1 row created.

SQL> insert into emp1 values(113,'aswani',03,55000 ,'bngl',7330229747, 'amgr');

1 row created.

SQL> insert into emp1 values(114,'suhasini',04,60000 ,'hyd',9989808323, 'clerk');

1 row created.

4. Selecting all rows from 'Departments' table:

SQL> select * from emp1;

ENO	ENAME	DEPTNO	SAL	LOC	MOB	DESG
111	rajani	01	50000	bngl	9703288493	gmgr
112	shobana	02	45000	hyd	9705185164	mgr
113	ashwani	03	55000	bngl	7330229747	amgr
114	suhasini	04	60000	hyd	9989808323	clerk

OUTPUT:

➢ **Query's Evaluation:**

1. Selecting records from the table emp1 where the salary is greater than the salary of a specific employee from the table dept1.

SQL> select * from emp1 where sal>(select sal from dept01 where ename='rajani');

ENO	ENAME	DEPTNO	SAL	LOC	MOB	DESG
111	rajani	01	50000	bngl	9703288493	gmgr
113	aswani	03	55000	bngl	7330229747	amgr
114	suhasini	04	60000	hyd	9989808323	clerk

2. To retrieve records from the table emp1 where the desg (designation) matches the designation of an employee named 'raju' in the dept1 table.

SQL> select * from emp1 where desg=(select desg from dept1 where ename='raju');

ENO	ENAME	DEPTNO	SAL	LOC	MOB	DESG
111	rajani	01	50000	bngl	9703288493	gmgr

3. Select records from the table emp1 where the desg (designation) is in the list of designations associated with employees named 'ravi' or 'raju' in the dept1 table.

SQL> select * from emp1 where desgin(select desg from dept1 where ename in('rajani','shobana'));

ENO	ENAME	DEPTNO	SAL	LOC	MOB	DESG
112	Shobana	02	45000	hyd	9705185164	mgr
111	rajani	01	50000	bngl	9703288493	gmgr

4.To count the number of employees (emp1) for each designation (desg) that matches between emp1 and dept1, and then ordering the results by desg in ascending order.

SQL> select w.desg, count(*) from emp1 w, dept1 m where w.desg=m.desg group by w.desg order by w.desgasc;

DESG	COUNT(*)
amgr	1
gmgr	1
mgr	2

5.To retrieve employee details (ename, deptno, sal, mob) from the emp1 table where the department number (deptno) of the employee matches the department number in the dept1 table, and where the location (loc) in the dept1 table is 'la', and the salary (sal) of the employee falls within a specified range (between 45000 and 55000).

SQL> select e.ename,e.deptno,e.sal,e.mob from emp1 e, dept1 d where e.deptno=d.deptno and d.loc='la' and e.sal between 45000 and 55000;

ENAME	DEPTNO	SAL	MOB
aswani	03	55000	7330229747

6.To retrieve employee details (ename**,** desg**) along with their department location (**loc**) based on a query involving a subquery.**

SQL> select e.ename,e.desg, d.loc from emp1 e, dept1 d where e.deptno=d.deptno and e.eno in(select eno from dept1);

no rows selected

7. To perform a RIGHT OUTER JOIN between the dept1 and emp1 tables and retrieve all columns from both tables.

SQL> SELECT * FROM dept1 d RIGHT JOIN emp1 e ON d.deptno = e.deptno;

SQL>SELECT * FROM dept1 d LEFT JOIN emp1 e ON d.deptno = e.deptno;

ENO	ENAME	DEPTNO	SAL	LOC	MOB	DESG
101	Kohili	01	45000	hyd	9968335420	mgr
111	rajani	01	50000	bngl	9703288493	gmgr
102	dhawan	02	50000	bngl	9836254107	gmgr
112	shobana	02	45000	hyd	9705185164	mgr
103	sachin	03	60000	tpt	7896554986	amgr
113	ashwani	03	55000	bngl	7330229747	amgr
104	dravid	04	55000	chnnai	8919090999	mgr
114	suhasini	04	60000	hyd	9989808323	clerk

8. To perform a RIGHT OUTER JOIN between the dept1 and emp1 tables and retrieve all columns from both tables.

SQL> SELECT * FROM dept1 d RIGHT JOIN emp1 e ON d.deptno = e.deptno;

ENO	ENAME	DEPTNO	SAL	LOC	MOB	DESG
101	Kohili	01	45000	hyd	9968335420	mgr

111	rajani	01	50000	bngl	9703288493	gmgr
102	dhawan	02	50000	bngl	9836254107	gmgr
112	shobana	02	45000	hyd	9705185164	mgr
103	sachin	03	60000	tpt	7896554986	amgr
113	ashwani	03	55000	bngl	7330229747	amgr
104	dravid	04	55000	chnnai	8919090999	mgr
114	suhasini	04	60000	hyd	9989808323	clerk

PROGRAM 5 – Creating Object Types

INPUT:

1. Creating a user-defined object types using the CREATE TYPEstatement.

SQL> create type location as object(drnovarchar(15), street varchar(10), city varchar(10), pincode number(6));
2 /
Type created.
SQL> create type person as object(fnamevarchar(10), mnamevarchar(10), lnamevarchar(10));
2 /
Type created.
SQL> create type qualification as object(hqvarchar(10), yopvarchar(4), per number(5,2));
2 /
Type created.
SQL> create type dept2 as object(empno number(5), name person, address location, edu qualification);
2 /
Type created.

2. Creating a table emp2 based on the structure of an existing table dept2. SQL> create table emp2 of dept2; Table created.

3. Inserting some records into the table.

SQL> insert into emp2values(dept2(&empno,person1('&fname','&mname','&lname'),location('&drno','&street','&city',&pi ncode),qualification('&hq','&yop',&per)));
Enter value for empno: 1234
Enter value for fname: RAMA
Enter value for mname: KRISHNA
Enter value for lname: REDDY
Enter value for drno: 14-6/2
Enter value for street: SBI COLONY
Enter value for city: TPT
Enter value for pincode: 517234
Enter value for hq: MCA
Enter value for yop: 2019
Enter value for per: 81 old 1: insert into emp2values(dept2(&empno,person('&fname','&mname','&lname'),location('&drno','&street','&city',&pi ncode),qualification('&hq','&yop',&per)))
new 1: insert into emp2values(dept2(1234,person('RAMA','KRISHNA','REDDY'),location('14-6/2','SBI

```
COLONY','TPT',517234),qualification('MCA','2
019',81))) 1 row created.

SQL> /
Enter value for empno: 2345
Enter value for fname: MANOJ
Enter value for mname: KUMAR
Enter value for lname: RAJU
Enter value for drno: 16-4/2
Enter value for street: KSR COLONY
Enter value for city: TPT
Enter value for pincode: 517501
Enter value for hq: BTECH
Enter value for yop:
2016 Enter value
for per: 87
old 1: insert into emp2
values(dept2(&empno,person('&fname','&mname','&lname'),location('&drno','&street','&city'
,&pincode)
,qualification('&hq','&yop',&per)))
new 1: insert into emp2 values(dept2(2345,person('MANOJ','KUMAR','RAJU'),location('16-
4/2','KSR COLONY','TPT',517501),qualification('BTECH','2016',87))) 1 row created.

SQL> /
Enter value for empno: 7865
Enter value for fname: SASI
Enter value for mname: DAHUR
Enter value for lname: KUMAR
Enter value for drno: 45-9/1
Enter value for street: NGP CIRCLE
Enter value for city: MPL
Enter value for pincode: 517110
Enter value for hq: MBA
Enter value for yop:
2011 Enter value
for per: 66
old 1: insert into emp2
values(dept2(&empno,person('&fname','&mname','&lname'),location('&drno','&street','&city'
,&pincode)
,qualification('&hq','&yop',&per)))
new 1: insert into emp2 values(dept2(7865,person('SASI','DAHUR','KUMAR'),location('45-
9/1','NGP CIRCLE','MPL',517110),qualification('MBA','2011',66))) 1 row created.
```

OUTPUT:

1. Selecting all rows from emp2table

```
SQL> SELECT * FROM EMP2;
EMPNONAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------------------
ADDRESS(DRNO, STREET, CITY, PINCODE)
--------------------------------------------------------------------------------
EDU(HQ, YOP, PER)
--------------------------------------------------------------------------------
--------       1234
PERSON('RAMA', 'KRISHNA', 'REDDY')
LOCATION('14-6/2', 'SBI COLONY', 'TPT', 517234)
QUALIFICATION('MCA', '2019', 81)
    EMPNO
----------
NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------------------
ADDRESS(DRNO, STREET, CITY, PINCODE)
--------------------------------------------------------------------------------
EDU(HQ, YOP, PER)
--------------------------------------------------------------------------------
--------       2345
PERSON('MANOJ', 'KUMAR', 'RAJU')
LOCATION('16-4/2', 'KSR COLONY', 'TPT', 517501)
QUALIFICATION('BTECH', '2016', 87)
    EMPNO
----------
NAME(FNAME, MNAME, LNAME)
--------------------------------------------------------------------------------
ADDRESS(DRNO, STREET, CITY, PINCODE)
--------------------------------------------------------------------------------
EDU(HQ, YOP, PER)
--------------------------------------------------------------------------------
--------       7865
PERSON('SASI', 'DAHUR', 'KUMAR')
LOCATION('45-9/1', 'NGP CIRCLE', 'MPL', 517110)
QUALIFICATION('MBA', '2011', 66)
SQL>
COMMIT;
Commit
complete.
```

PROGRAM 6– V ARRAYS

INPUT:

1. Create a user-defined object type project with ARRAYS.

SQL> create type project as object(pno number(5), title varchar(15), cost number(10,2));

2 /

Type created.

SQL> create type projectlist as varray(60)

of project; 2 /

Type created.

2. Creating a table "company".

SQL> create table company(cmpid number(10), name varchar(15), projects

projectlist); Table created.

3. Inserting some records into the table.

SQL> insert into company values(01,'act',projectlist(project(1,'a11',5000),

project(2,'ball',6000), project(3,'call',7000))); 1 row created.

SQL> insert into company values(02,'bat',projectlist(project(4,'doll',14000),

project(6,'fall',24000), project(8,'grill',32000))); 1 row created.

SQL> insert into company

values(03,'anchor',projectlist(project(12,'dell',25000),project(14,'lenovo',24000),

project(16,'acer',40000))); 1 row created.

OUTPUT:

1. Selecting all rows from "company"table

```
SQL> select * from  company;

    CMPID        NAME              PROJECTS(PNO, TITLE, COST)
--------------------------------------------------------------------
----------       01 act
PROJECTLIST(PROJECT(1, 'all', 5000), PROJECT(2, 'ball', 6000), PROJECT(3, 'call', 7000))

      02  dbc
PROJECTLIST(PROJECT(4, 'doll', 14000), PROJECT(6, 'fall', 24000),
PROJECT(8, 'grill', 32000))

       03 anchor
PROJECTLIST(PROJECT(12, 'dell', 25000), PROJECT(14, 'lenovo', 24000), PROJECT(16, 'acer'

    CMPID NAME
---------- ---------------
PROJECTS(PNO, TITLE, COST)
--------------------------------------------------------------------
---------- , 40000))


SQL> commit;

Commit complete.
```

PROGRAM 7 – CURSORS

INPUT:

1. Creating a table "employee".

SQL> create table employee(enonumber(6) primary key,
enamevarchar(20),desgvarchar(15), sal number(10,2), doj date)); Table created.
SQL>desc emp7;

Name	Null?	Type
ENO	NOT NULL	NUMBER(6)
ENAME		VARCHAR2(20)
DESG		VARCHAR2(15)
SAL		NUMBER(10,2)
DOJ		DATE

3. Inserting some records into the table.

SQL> insert into employee values(10001,'candy','clerk',22000,'5-jun-2015');

1 row created.

SQL> insert into employee values(1001,'sandy','manager',55000,'25-jun-2004');

1 row created.

4. Selecting all rows from employee table

SQL> select * from employee;

ENO	ENAME	DESG	SAL	DOJ
10001	candy	clerk	22000	05-JUN-15
1001	sandy	manager	55000	25-JUN-04

PL/SQL block to count records using cursors

SQL> set serveroutput on;

SQL> edit employee.sql

SQL>
decl
are
tot
num

```
ber;
begi
n
select count(*)into tot from employee;
dbms_output.put_line('total number of
records'|| tot);
end;
  /
```

OUTPUT:

```
total number of records2

PL/SQL procedure successfully completed.
```

PROGRAM 8 – CURSORS AS A PARAMETER

INPUT:

1. Creating a table "employee1".

SQL> create table employee1(enonumber(5) primary key, enamevarchar(20),

desgvarchar(15), sal number(10,2), doj date)); Table created.

SQL>desc emp8;

Name	Null?	Type
ENO	NOT NULL	NUMBER(5)
ENAME		VARCHAR2(20)
DESG		VARCHAR2(15)
SAL		NUMBER(10,2)
DOJ		DATE

3. Inserting some records into the table.

SQL> insert into employee1 values(12345,'ranga','hr',80000,'15-dec-2010');

1 row created.

SQL> insert into employee1 values(34567,'raju','clerk',34000,'15-dec-2015');

1 row created.

SQL> insert into employee1 values(23456,'ramesh','mng',34000,'15-dec-2015');

1 row created.

4. Selecting all rows from employee1 table

SQL> select * from employee1;

ENO	ENAME	DESG	SAL	DOJ
12345	ranga	hr	80000	15-DEC-10
34567	raju	clerk	34000	15-DEC-15
23456	ramesh	mng	34000	15-DEC-15

PL/SQL block to calculate total salary of “N” employees

```
SQL> set serveroutput on;
SQL> edit employee1.sql;

SQL> declare
totsalnumber:=0; salary
employee1.sal%type;
cursores(n number) is
selectsal from emp8 where
rownum<=n;
begin
openes(&n);
loop fetches
into salary;
exit when
es%notfoun
d;
totsal:=totsa
l+salary; end
loop;
dbms_output.put_line('total salry
is' || totsal); closees; end;
  /
```

OUTPUT:

```
Enter
value  for
n:  3  old
7:    open
es(&n);
new     7:
open
es(3);
Total salry is114000

PL/SQL procedure successfully completed.
```

PROGRAM 9 – ELECTRICITY BILL

INPUT:

1. Creating a table “electricity ” .

SQL> CREATE TABLE electricity_consumers(meter_no VARCHAR2(20),
consumer_nameVARCHAR2(100), address VARCHAR2(200), current_reading NUMBER(10),
previous_reading NUMBER(10), category CHAR(1), arrears_total NUMBER(10),
last_date_no_fine DATE, last_date_with_fine DATE); Table created.
SQL>descelectricity_consumers ;

Name	Null?	Type
METER_NO		VARCHAR2(20)
CONSUMER_NAME		VARCHAR2(100)
ADDRESS		VARCHAR2(200)
CURRENT_READING		NUMBER(10)
PREVIOUS_READING		NUMBER(10)
CATEGORY		CHAR(1)
ARREARS_TOTAL		NUMBER(10)
LAST_DATE_NO_FINE DATE		
LAST_DATE_WITH_FINE DATE		

3. Inserting some records into the table.

SQL> INSERT INTO electricity_consumers VALUES ('M123', 'John Doe',
'123 Main St' , 600, 400, 'A', 0, TO_DATE('2024-07-15', 'YYYY-MM-DD'),
TO_DATE('2024-07-20', ' YYYY-MM-DD')); 1 row created.

4. Selecting all rows from electricity_consumerstable

SQL> select * from electricity_consumers;

```
METER_NO
--------------------
CONSUMER_NAME
--------------------------------------------------------------------------------
ADDRESS
--------------------------------------------------------------------------------
CURRENT_READING PREVIOUS_READING C ARREARS_TOTAL LAST_DATE LAST_DATE
--------------- ---------------- - ------------- --------- ---------
M123
John Doe
123 Main St
 600             400 A           0 15-JUL-24 20-JUL-24
```

PL/SQL program to calculate and print electricity bill

```
SQL>SET SERVEROUTPUT ON;
DECLARE
v_month_yearVARCHAR2(50) := 'July 2024';
v_meter_noVARCHAR2(2
0);
v_consumer_nameVARC
HAR2(100);
v_addressVARCHAR2(20
0); v_current_reading
NUMBER;
v_previous_reading
NUMBER; v_billed_units
NUMBER;
v_unit_costNUMBER :=
10; v_category CHAR(1);
v_arrears_total
NUMBER;
v_amount_payable
NUMBER;
v_last_date_no_fine
DATE;
v_last_date_with_fine DATE;

   CURSOR c_consumers IS
      SELECT meter_no, consumer_name, address, current_reading, previous_reading,
category, arrears_total, last_date_no_fine, last_date_with_fine
      FROM electricity_consumers;
BEGIN
   FOR consumer_rec IN c_consumers LOOP
v_meter_no := consumer_rec.meter_no;
v_consumer_name :=
consumer_rec.consumer_name; v_address :=
consumer_rec.address; v_current_reading :=
consumer_rec.current_reading;
v_previous_reading :=
consumer_rec.previous_reading; v_category :=
consumer_rec.category; v_arrears_total :=
consumer_rec.arrears_total; v_last_date_no_fine
:= consumer_rec.last_date_no_fine;
v_last_date_with_fine :=
consumer_rec.last_date_with_fine;

      -- Calculate billed units
```

```
v_billed_units := v_current_reading - v_previous_reading;

      -- Calculate
amount payable       IF
v_billed_units> 0 THEN
v_amount_payable := v_billed_units * v_unit_cost + v_arrears_total;
      ELSE
v_amount_payable := v_arrears_total;
      END IF;
            -- Print bill in required format
      DBMS_OUTPUT.PUT_LINE('Electricity bill for the month of ' || v_month_year);
      DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');
      DBMS_OUTPUT.PUT_LINE('Meter no: ' || LPAD(v_meter_no, 20) || 'Category: ' ||
v_category);                              DBMS_OUTPUT.PUT_LINE('Consumer    Name:    '    ||
LPAD(v_consumer_name, 40) || 'Address: ' || v_address);
      DBMS_OUTPUT.PUT_LINE('Current Reading: ' || LPAD(v_current_reading, 20) || 'Previous
Reading: ' ||v_previous_reading);
      DBMS_OUTPUT.PUT_LINE('Billed units: ' || LPAD(v_billed_units, 20) || 'Unit Cost: ' ||
v_unit_cost);         DBMS_OUTPUT.PUT_LINE('Arrears Total: ' || LPAD(v_arrears_total, 20) ||
'Amount Payable: ' || v_amount_payable);
      DBMS_OUTPUT.PUT_LINE('Last Date without fine: ' || TO_CHAR(v_last_date_no_fine,
'YYYY-MMDD') || '     Last Date with fine: ' || TO_CHAR(v_last_date_with_fine, 'YYYY-MM-
DD'));        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');
      DBMS_OUTPUT.NEW_LINE;
   END LOOP;
END;
/
```

OUTPUT:

```
Electricity bill for the month of July 2024
------------------------------------------------------------
------ Meter no:              M123Category: A
Consumer Name:                        John DoeAddress: 123 Main St
Current Reading:              600Previous Reading: 400
Billed units:              200Unit Cost: 10
Arrears Total:                 0Amount Payable: 2000
Last Date without fine: 2024-07-15     Last Date with fine: 2024-07-20
------------------------------------------------------------

PL/SQL procedure successfully completed.
```

PROGRAM 10 – ICET RANK CALCULATION

INPUT:

1.Creating a table " ICET" .

SQL> create table icet(htno number(7) primary key, eng number(3), res number(3),arithc number(3), tot number(5),rank number(5)); Table created.

SQL>desc icet;

Name	Null?	Type
HTNO	NOT NULL	NUMBER(7)
ENG		NUMBER(3)
RES		NUMBER(3)
ARITHC		NUMBER(3)
TOT		NUMBER(5)
RANK		NUMBER(5)

3. Inserting some records into the table.

SQL> insert into icet(htno,eng,res,arithc)

values(543,45,56,43); 1 row created.

SQL> insert into icet(htno,eng,res,arithc)

values(653,34,55,60); 1 row created.

SQL> insert into icet(htno,eng,res,arithc)

values(974,33,61,21); 1 row created.

SQL> commit;

Commit complete.

4. Selecting all rows from "ICET" table.

SQL> select * from icet;

HTNO	ENG	RES	ARITHC	TOT	RANK
543	45	56	43		
653	34	55	60		
974	33	61	21		

PL/SQL program to calculate and print ICET RANKS

SQL>set serveroutput on

```
SQL> declare cursoricur is select * from icet
        order by tot desc;
        strecicet%rowtype; ricet.rank%type;
        toticet.tot%type; begin openicur;
        r:=0; tot:=0; loop fetchicur into strec;
        exit when icur%notfound;
        strec.tot:=strec.eng+strec.res+strec.a
        rithc; updateicet set
        tot=strec.totwherehtno=strec.htno;
        if tot!=strec.tot then r:=r+1; end if;
        updateicet set rank=r where
        htno=strec.htno; tot:=strec.tot;
        end loop; closeicur; end;
         /
        PL/SQL procedure successfully completed.
```

OUTPUT:

```
SQL> select * from icet;
```

HTNO	ENG	RES	ARITHC	TOT	RANK
543	45	56	43	144	1
653	34	55	60	149	3
974	33	61	21	115	2

PROGRAM 11 – TRIGGERS

1. Creating“ Employee” table.

```
SQL>CREATE TABLE Employee11(emp_idNUMBER(10) PRIMARYKEY,emp_name
VARCHAR(50),emp_salary NUMBER(10));
SQL>desc employee11;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
---------
 EMP_ID                                    NOT NULL NUMBER(10)
 EMP_NAME                                           VARCHAR2(50)
 EMP_SALARY                                         NUMBER(10)
```

2. Inserting a records into“employee” table.

```
SQL>INSERTINTOEmployee11(emp_id,emp_name,emp_salary)VALUES(1,'JohnDoe
',50000); SQL> INSERT INTO
Employee11(emp_id,emp_name,emp_salary)VALUES(2,'roy', 35000); 1 row
created.
SQL> INSERT INTO
Employee11(emp_id,emp_name,emp_salary)VALUES(3,'githa', 55000);
1 row created.
```

3. Selecting all records from “employee” table.

```
SQL> select * from employee11;

    EMP_ID EMP_NAME
EMP_SALARY
---------- -------------------------------------------------- -------
---
1     JohnDoe                                                    50000
2     roy                                                        35000
3 githa                                                       55000
```

OUTPUT:

```
SQL>CREATETABLE Audit_Log (log_idNUMBER(10)  PRIMARYKEY, table_nameVARCHAR2
activity_typeVARCHAR2  (20) ,activity_date TIMESTAMP, user_idVARCHAR2 (50));
```

1.Create"Audit_Log" table

```
                                                                    (100),

// CREATING A SEQUENCE
SQL>CREATE SEQUENCE Audit_Log_SeqSTARTWITH1 INCREMENT BY1;
// CREATING A TRIGGER
SQL>CREATEORREPLACETRIGGER Employee11_Audit_Trigger
AFTERINSERTORUPDATEORDELETEON Employee11
FOR EACH
ROW DECLARE
v_activity_typeVARCHAR2(20);
BEGIN
IF INSERTING THEN
v_activity_type
:='INSERT'; ELSIF
UPDATING THEN
v_activity_type
:='UPDATE'; ELSIF
DELETING THEN
v_activity_type
:='DELETE';
ENDIF;
INSERT    Audit_Log (log_id ,table_name ,activity_type ,activity_date ,user_id
INTO                                                                    )
VALUES Audit_Log_Seq .NEXTVAL ,'Employee   v_activity_type , SYSTIMESTAMP
(                                                                  ,USER);
END;
/
```

2.Insert a new employee into the table.

```
SQL>INSERTINTOEmployee11(emp_id,emp_name,emp_salary)VALUES(5,'Hari',
10000); 1 row created.
```

3. Updating an employee's salary.

```
SQL>UPDATE
Employee11SETemp_salary=55000WHEREemp_id=1; 1
row updated.
```

4.Deleting an employee record from table.

```
SQL>DELETEFROM
Employee11WHEREemp_id=1; 1 row
deleted.
```

5.selecting an employee record from "Audit_log" table. SQL> SELECT * FROM Audit_Log;

LOG_ID TABLE_NAME ACTIVITY_TYPE ACTIVITY_DATE USER_ID

---------- 1 Employee INSERT 07-JUL-24

05.34.53.975000 PM SYSTEM

2 Employee UPDATE 07-JUL-24 05.34.53.975000 PM SYSTEM

3 Employee DELETE 07-JUL-24 05.34.53.975000 PM SYSTEM

PROGRAM 12 – FUNCTIONS

1. Creating factorial number using functions.

```
SQL>create function factorial(n in number)
return number
is
i
n
u
m
b
e
r
;
f
n
u
m
b
e
r
=
1
;
begin
for i
in
1..n
loop
f
:
=
f
```

```
*
i
;

e
n
d

l
o
o
p
;

r
e
t
u
r
n

f
;

e
n
d
;

/
```

Function created.

2.Creating the PL/SQL block to use this function.

```
SQL>Set
serveroutput
on;
SQL>declare n
number(3); fn
number(5);
begin n:=&n;
fn:=factorial(n
);
```

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

OUTPUT:

```
Enter value
for n: 3 old
5:
n:=&n; new
5:
n:=3;
factorial
is:6
PL/SQL procedure successfully completed.
```

PROGRAM 13 – OVERDRAFT EXCEPTION

1. Creating a table to simulate the bank_accounts

```
SQL>CREATE TABLE
bank_accounts( accno
NUMBER(10), cname
VARCHAR2(50),
balance NUMBER(10, 2)
);
SQL>descbank_accounts;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 ACCNO                                     NUMBER(10)
 CNAME                                     VARCHAR2(50)
 BALANCE                                   NUMBER(10,2)
```

2. Insert sample data into the bank_accounts table

```
SQL> INSERT INTO bank_accounts VALUES (101, 'Alice', 1500);

1 row created.

SQL> INSERT INTO bank_accounts VALUES (102, 'Bob', 2500);

1 row created.

SQL> INSERT INTO bank_accounts VALUES (103, 'Charlie', 3500);

1 row created.
```

3.Selecting all rows from “bank_accounts” table

```
SQL> select * from bank_accounts;

    ACCNO CNAME                                              BALANCE
---------- -------------------------------------------------- -
---------       101 Alice
1500
102 Bob                                          2500
103 Charlie                                       3500
SQL> commit;

Commit complete.
```

4.Create the PL/SQL block to handle overdraft exception

```
SQL>SET SERVEROUTPUT ON;

SQL>DECLAR
E
acnoNUMBE
R := &acno;
tamt
NUMBER :=
&tamt;   bal
NUMBER;
cnamebank_accounts.cname%TYPE;
BEGIN
   -- Retrieve account balance and customer name
   SELECT balance, cname INTO bal, cname FROM bank_accounts WHERE accno = acno;

   -- Check if sufficient balance is available for withdrawal
   IF bal>= tamt
THEN        --
Perform
withdrawal
bal := bal - tamt;
      UPDATE bank_accounts SET balance = bal WHERE accno = acno;

      -- Output transaction details
      DBMS_OUTPUT.PUT_LINE('Withdrawal of ' || tamt || ' successful for account ' || acno);
DBMS_OUTPUT.PUT_LINE('Remaining balance: ' || bal);
   ELSE
      -- Raise overdraft exception if balance is insufficient
      RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance for
withdrawal');     END IF;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Account number ' || acno || ' not found');

   WHEN TOO_MANY_ROWS THEN
      DBMS_OUTPUT.PUT_LINE('Multiple accounts found for account number ' || acno);

   WHEN VALUE_ERROR THEN
      DBMS_OUTPUT.PUT_LINE('Invalid input value');
```

```
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

OUTPUT:

```
Enter value for acno: 101 old   2:
acnoNUMBER := &acno; new   2:
acno NUMBER := 101; Enter value
for tamt: 1500 old   3:     tamt
NUMBER := &tamt; new   3:
tamt NUMBER := 1500;
Withdrawal of 1500 successful for account 101
Remaining balance: 0
PL/SQL procedure successfully completed.
SQL> /
Enter value for acno: 102 old   2:
acnoNUMBER := &acno; new   2:
acno NUMBER := 102; Enter value
for tamt: 100 old   3:     tamt
NUMBER := &tamt; new   3:
tamt NUMBER := 100;
Withdrawal of 100 successful for account 102
Remaining balance: 2400
PL/SQL procedure successfully completed.
SQL> select * from bank_accounts;
  ACCNO CNAME                                BALANCE
---------- -------------------------------------------------- ----------
101 Alice                                       0
102 Bob                                      2400
103 Charlie                                  3500
```

PROGRAM 14 – STATISTICAL PACKAGE

INPUT:

1. Creating a table "numbers" for statistical package program SQL> create table numbers(num number(3)); Table created.

```
SQL>desc numbers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NUM                                                NUMBER(3)
```

2. Inserting values into the table

```
SQL> insert into numbers values(&num); Enter
value for num: 5
old   1: insert into numbers values(&num)
new   1: insert into numbers values(5) 1
row created.
SQL> /
Enter value for num: 3
old   1: insert into numbers values(&num)
new   1: insert into numbers values(3) 1
row created.
SQL> /
Enter value for num: 2
old   1: insert into numbers values(&num)
new   1: insert into numbers values(2) 1
row created.
SQL> /
Enter value for num: 4
old   1: insert into numbers values(&num)
new   1: insert into numbers values(4) 1
row created.
SQL> /
Enter value for num: 1
old   1: insert into numbers values(&num)
new   1: insert into numbers values(1) 1
row created.

SQL> /
```

Enter value for num: 6
old 1: insert into numbers values(&num)

new 1: insert into numbers values(6) 1

row created.

3. Selecting all records from table

SQL> select * from numbers;

```
      NUM
----------
         5
         3
         2
         4
         1
         6
```

6 rows selected.

//Creating the Package Specification

SQL> create or replace package stats is

function mean return number;

proceduresd(s out number); end stats;

/

Package created.

//Creating the Package Body Specification

SQL> create or replace package body stats is

function mean return number is x

number:=0; c number:=0; mn number(5,2);

cursor cur is select * from numbers;

meancurnumbers%rowtype; begin open

cur; loop fetch cur into meancur;

x:=x+meancur.num; c:=c+1; exit when

cur%not found; end loop; mn:=x/c;

returnmn;

close cur; end mean;

proceduresd(s out number)is

x1 number:=0; x2

number:=0; c number:=0;

cursor cur is select * from numbers;

sdcurnumbers%rowtype; begin

open cur; loop fetch cur into

```
sddcur; x1:=x1+sdcur.num;
x2:=x2+(sdcur.num*sdcur.num);
c:=c+1; exit when cur%notfound;
end loop; s:=sqrt((x2/c)-
((x1*x1)/(c*c))); close cur; endsd;
end stats;
/
Package Body created.
//PL/SQL Program to Display the Mean and Standard Deviation
SQL> set serveroutput on;
SQL> declare s number;
begin
dbms_output.put_line('mean:'||stats.mean); stats.sd(s);
dbms_output.put_line('standard deviation:'||s); end;
 /
```

OUTPUT: mean:3.86

standard deviation:1.807015805810502475427939 16 PL/SQL

procedure successfully completed.

PROGRAM 15STORED PROCEDURES

INPUT:

1. Creating a table "emp14" for Stored Procedures.

SQL> create table emp14(eidnumber(3) primary key, ename varchar(10), desg varchar(10), sal number(10,2));Table created.

SQL>desc emp14;

Name	Null?	Type
EID	NOT NULL	NUMBER(3)
ENAME		VARCHAR2(10)
DESG		VARCHAR2(10)
SAL		NUMBER(10,2)

2. Inserting some records into table "emp14".

SQL> insert into emp14 values(123,'ravi', 'manager',45000); 1

row created.

SQL> insert into emp14 values(124,'naresh','clerk',30000); 1

row created.

SQL> insert into emp14 values(125,'bala','sr clerk',55000); 1

row created.

SQL> insert into emp14 values(126,'nani','g manager',60000); 1

row created.

3. Selecting all records from emp14 table.

SQL> select * from emp14;

EID	ENAME	DESG	SAL
123	ravi	manager	45000
124	naresh	clerk	30000
125	balasr	clerk	55000
126	nani	g manager	60000

Creating a Stored Procedure for updating details in employee table.

//Procedure Creation

SQL>create or replace procedure updateemp(id number, salary in number) as begin

update emp14 set sal=salary+500 whereeid=id; end;

/

Procedure created.

//Procedure Usage

```
SQL>set serveroutput on;
SQL>declare id number;
sal number; begin
id:=&id; sal:=&sal;
updateemp(id,sal);
dbms_output.put_line('employee salary updated'); end;
    /
```

OUTPUT:

```
Enter value for id: 123
old   5:     id:=&id; new
5:     id:=123; Enter value
for sal: 45500 old   6:
sal:=&sal; new   6:
sal:=45500; employee
salary updated
PL/SQL procedure successfully completed.
```