DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 1 DATE:

SQL Single row functions

Single row functions can be categorized into four types, single row functions are applied for each row and produces individual output for each row.

1. Number functions
2. Character functions
3. Date functions
4. Conversion functions

Dual Table: dual is a default table, created at the time of oracle installation.

Regd. No:

Number functions

ABS (): Absolute is the measure of the magnitude of a value. This function returns absolute value is always a positive number.

```
select abs(-9) from dual;
```
Output: 9

CEIL (): This function returns largest integer greater than or equal to n.

```
select ceil(18.23) from dual;
```
Output: 19

FLOOR (): This function returns the smallest integer equal to or less than n.

```
select floor(18.23) from dual;
```
Output: 18

SQRT (): This function gives the square root of the given value n.

```
Select sqrt(576) from dual;
```
Output: 24

MOD (): This function gives the remainder when m is divided by n.

```
select mod(17,5) from dual;
```
Output: 2

POWER (): This function gives the value of m raised to the power of n.

```
select power(3,3) from dual;
```
Output: 27

ROUND (): This function rounds the number to the given number of digits of precision

```
select round(14.5264,2) from dual;
```
Output: 14.53

TRUNC (): This function truncates the decimal portion. This function truncates (deletes) m decimal to n decimal places.

```
Select trunc(10.10998998,4) from dual;
```
Output: 10.1099

LEAST (): This function returns least integer from a set of integers.

```
select least(5,8,1,95,72,48,22,8958,2) from dual;
```
Output: 1

GREATEST (): This function returns GREATEST integer from a set of integers.

```
select greatest(5,8,1,95,72,48,22,8958,2) from dual
```
Output: 8958

Regd. No:

CHARACTER FUNCTIONS

INITCAP ( ): This function returns the string with first letter of each word in uppercase.
Syntax: **INITCAP (string1)**

Select initcap('andhra prasad') from dual;
Output: Andhra Prasad

LOWER ( ): This function returns the string in lower case.
Syntax: **LOWER (string1)**

select lower('THE PEN IS MIGHTIER THAN THE SWORD') from dual;
Output: the pen is mightier than the sword

UPPER ( ): This function returns the string in upper case.
Syntax: **UPPER (string1)**

select upper('the pen is mightier than the sword') from dual;
Output: THE PEN IS MIGHTIER THAN THE SWORD

CONCAT ( ): This function returns a string by appending string1 with string2.
Syntax: **CONCAT (string1, string2)**

Select concat('hello','every one') from dual;
Output: helloevery one

LENGTH ( ): This function gives length of the given string.
Syntax: **LENGTH (string)**

select length('Fortune favors the bold') from dual;
Output: 23

SUBSTR ( ): This function returns a portion of a string beginning at the character position.
Syntax: SUBSTR (STRING, POSITION, OFFSET)

select substr('Theres no such thing as a free lunch',10,5) from dual;
Output: such

INSTR ( ): This function returns Nth occurrence of string2 (first character position) in string1. In string1 characters index begins from 0. This function starts searching from Mth position.
Syntax: INSTR (STRING1, STRING2, M, N)

select instr('Theres no such thing as a free lunch','re',1,1) from dual;
Output: 4

select instr('Theres no such thing as a free lunch','re',1,2) from dual;
Output: 28

TRANSLATE ( ): This function returns a string after replacing some set of characters into another set.
Syntax: TRANSLATE (MAIN STRING, FROM_STRING, TO_STRING)

select translate('delhi is the capital of india','i','a') from dual;
Output: delha as the capatal of andaa

Regd. No:

select translate('Theres no such thing as a free lunch','res','12@') from dual;
Output: Th212@ no @uch thing a@ a f122 lunch

LPAD ( ): This function returns a string as output after padding string2 to the left side of string1 to n length.
Syntax: LPAD (STRING1, N, STRING2)
Select lpad('india',20,'$') from dual;
Output: $$$$$$$$$$$$$$$india

RPAD ( ): This function returns a string as output after padding string2 to the right side of string1 to n length.
Syntax: RPAD (STRING1, N, STRING2)
Select rpad('india',20,'&') from dual;
Output: india&&&&&&&&&&&&&&&

LTRIM ( ): This function returns a string as output after trim string2 from left side up to the string1 which is not in set.
Syntax: LTRIM (STRING1, STRING2)
Select ltrim('abcxyzabcxyz','abc') from dual;
Output: xyzabcxyz

RTRIM ( ): This function returns a string as output after trim string2 from right side up to the string1 which is not in set.
Syntax: RTRIM (STRING1, STRING2)
Select rtrim('abcxyzabcxyz','xyz') from dual;
Output: abcxyzabc

Regd. No:

DATE FUNCTIONS

SYSDATE: This function returns current date of system.

```
Select sysdate from dual;
```
Result: 10-JUN-22

ADD_MONTHS (): This function returns date d plus n months, i.e adds n months to the given date d.

Syntax: ADD_MONTHS (DATE, NO_OF_MONTHS)

```
Select add_months('15-aug-1947',12) from dual;
```
Result: 15-AUG-48

```
Select add_months('01-may-2017',15) from dual;
```
Result: 01-AUG-18

MONTHS_BETWEEN (): This function returns difference between given two dates.

Syntax: MONTHS_BETWEEN (DATE1, DATE2)

```
Select months_between('19-SEP-16','17-MAY-16') from dual;
```
Result: 4.064516

```
Select months_between('19-FEB-16','17-MAY-16') from dual;
```
Result: -2.9354839

NEXT_DAY (): This function returns the date of the next weekday from the date specified.

Syntax: NEXT_DAY (DATE, 'WEEKDAY')

```
Select next_day('15-aug-1947','sun') from dual;
```
Result: 17-AUG-47

```
Select next_day('25-jul-17','sun') from dual;
```
Result: 30-JUL-17

LAST_DAY (): This function returns the date of the last day of the month.

Syntax: LAST_DAY (DATE)

```
Select last_day('15-aug-1947') from dual;
```
Result: 31-AUG-47

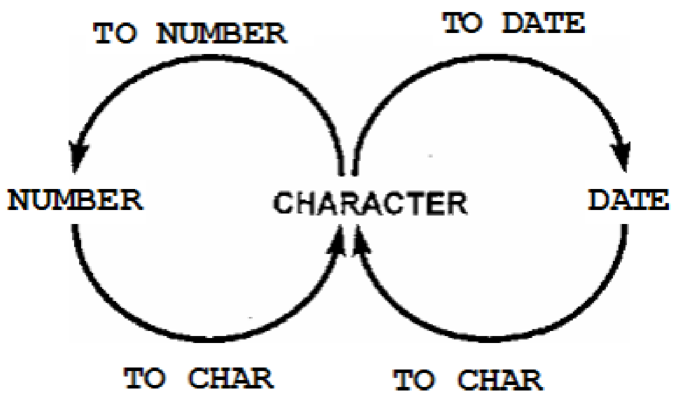
```
Select last_day('22-apr-2017') from dual;
```
Result: 30-APR-17

Regd. No:

CONVERSION FUNCTIONS

SQL provides three functions to convert a value from one data type to another

1. TO_CHAR (number | date [, 'fmt']): Converts a number or a date value to a VARCHAR2 character string with format model fmt.
2. TO_NUMBER (char [, 'fmt']): Converts a character string containing digits to a number with the optional format model fmt.
3. TO_DATE (char [, 'fmt']): Converts a character string representing a date to a date value according to the fmt specified (If fmt is omitted, format is DD-MONYY.)



TO_CHAR ():

Select to_char(30000,'$99999') from dual;
Result: $30000

Select to_char(sysdate,'day,month year') from dual;
Result: friday ,june twenty twenty-two

select to_char(sysdate,'dd,mon yyyy') from dual;
Result: 10,jun 2022

TO_DATE ():

Select to_date('25 january,17') from dual;
Result: 25-JAN-17

Select to_date('2 january,17') from dual;
Result: 02-JAN-17

TO_NUMBER ():

Select to_number('1210.72','9999.99') from dual;
Result: 1210.72

Regd. No: | | | | | | | | | |

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 2 **DATE:**

Queries using operators in SQL

SQL Operators

List of operators used in SQL.

1. Arithmetic Operators
2. Character Operator
3. Comparison Operators
4. Logical Operators
5. Set Operators

Levels of Precedence of the Oracle Database Lite SQL Operators

Precedence Level	SQL Operator
1	Unary + - arithmetic operators
2	* / arithmetic operators
3	Binary + - arithmetic operators, || character operators
4	All comparison operators
5	NOT logical operator
6	AND logical operator
7	OR logical operator

Arithmetic Operators

Division (/)

SELECT SAL / 10 FROM EMP;

Multiplication (*)

SELECT SAL * 5 FROM EMP;

Addition (+)

SELECT SAL + 200 FROM EMP;

Subtraction (-)

SELECT SAL - 100 FROM EMP;

Character Operator

Concatenation (||): Concatenates character strings

SELECT 'Name of the employee is: ' || ENAME FROM EMP;

Regd. No: | | | | | | | | | | |

Comparison Operators

Comparison operators used in conditions that compare one expression with another. The result of a comparison can be TRUE, FALSE, or UNKNOWN.

Operator	Description	Example
=	Equality test.	SELECT ENAME "Employee" FROM EMP WHERE SAL = 1500;
!=, ^=, <>	Inequality test.	SELECT ENAME FROM EMP WHERE SAL != 5000;
>	Greater than test.	SELECT ENAME "Employee", JOB "Title" FROM EMP WHERE SAL > 3000;
<	Less than test.	SELECT * FROM EMP WHERE SAL< 3000;
>=	Greater than or equal to test.	SELECT empno,ename FROM EMP WHERE SAL>= 2000;
<=	Less than or equal to test.	SELECT ENAME FROM EMP WHERE SAL <= 1500;
IN	"Equivalent to any member of" test. Equivalent to "=ANY".	SELECT empno,ename,job FROM EMP WHERE ENAME IN ('SMITH', 'WARD');
NOT IN	Equivalent to "!=ANY". Evaluates to FALSE if any member of the set is NULL.	SELECT deptno,dname FROM DEPT WHERE LOC NOT IN ('NEW YORK', 'DALLAS');
ANY/ SOME	Compares a value to each value in a list or returned by a query. Must be preceded by =, !=, >, <, <= or >=. Evaluates to FASLE if the query returns no rows.	SELECT deptno,dname FROM DEPT WHERE LOC = SOME ('BOSTON','CHICAGO');
ALL	Compares a value with every value in a list or returned by a query. Must be preceded by =, !=, >, <, <= or >=. Evaluates to TRUE if the query returns no rows.	SELECT ename,sal,job FROM emp WHERE sal>= ALL (1400, 3000);
[NOT]	[Not] greater than or equal to *x* and less	SELECT ENAME, JOB FROM EMP WHERE SAL BETWEEN 3000 AND

Operator	Description	Example
BETWEEN *x*and *y*	than or equal to *y*.	5000;
x [NOT] LIKE *y*	TRUE if *x* does [not] match the pattern *y*. Within *y*, the character "%" matches any string of zero or more characters except null. The character "_" matches any single character.	1. LIKE 'S%' 2. LIKE '%A' 3. LIKE 'R_ _ _' 4. LIKE '_ _ _ I' SELECT ename,sal FROM EMP WHERE ENAME LIKE '%E%';
IS [NOT] NULL	Tests for nulls. This is the only operator that should be used to test for nulls.	SELECT empno,ename,sal FROM EMP WHERE COMM IS NULL; SELECT * FROM EMP WHERE COMM IS NOT NULL AND SAL > 1500;
EXISTS	TRUE if a sub-query returns at least one row.	SELECT empno,enam,sal FROM EMP WHERE EXISTS (SELECT ENAME FROM EMP WHERE MGR IS NULL);

Logical Operators

Operator	Description	Example
NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN.	SELECT ename,job FROM EMP WHERE NOT (job IS NULL) SELECT empno,ename,sal,job FROM EMP WHERE NOT (sal BETWEEN 1000 AND 2000)
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise returns UNKNOWN.	SELECT empno,ename,sal FROM EMP WHERE job='CLERK' AND deptno=10
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE. Otherwise, returns UNKNOWN.	SELECT empno,ename,mgr,comm FROM emp WHERE job='CLERK' OR deptno=10 SELECT empno,ename,nvl2(mgr,mgr,0),nvl2(comm,comm,0) FROM emp WHERE job='CLERK' OR deptno=10

Regd. No:

Set Operators

Set operators which combine the results of two queries into a single result.

Operator	Description	Example

UNION	Returns all distinct rows selected by either query.	SELECT ename,sal,job FROM EMP WHERE SAL <= 1000
		
		UNION
		
		SELECT ename,sal,job FROM EMP WHERE SAL >= 2500;

UNION ALL	Returns all rows selected by either query, including all duplicates.	SELECT ename,sal,job FROM EMP WHERE SAL <= 1000
		
		UNION ALL
		
		SELECT ename,sal,job FROM EMP WHERE SAL >= 2500;

INTERSECT	Returns all distinct rows selected by both queries.	SELECT ename,sal,job FROM EMP WHERE SAL <= 1500
		
		INTERSECT
		
		SELECT ename,sal,job FROM EMP WHERE SAL >= 1200;

MINUS	Returns all distinct rows selected by the first query but not the second.	SELECT ename,sal,job FROM EMP WHERE SAL >= 1200
		
		MINUS
		
		SELECT ename,sal,job FROM EMP WHERE SAL <= 1500;

Regd. No:

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 3 DATE:

Employee Database

Create the Following Tables and insert the Data in them

DEPT

DEPTNO	NAME	LOCATION
10	Accounting	122
20	Research	124
30	Sales	123
40	Operations	167
12	Research	122
13	Sales	122
14	Operations	122
23	Sales	124
24	Operations	124
34	Operations	123
43	Sales	167

LOCATION

LCODE	NAME
122	Kakinada
124	Hyderabad
123	Bangalore
167	Vijayawada

JOB

JCODE	NAME
667	Clerk
668	Staff
669	Analyst
671	Vice President
672	President

Regd. No:

EMPLOYEE

EMP NO	ENAME	JOB	MGR_NO	HIREDATE	SALARY	COMMISSION	DEPTNO
1	Venkat	672		01-02-06	1200000	10000	40
2	Nirmala	671	1	02-04-07	800000	50000	20
3	Pradeep	669	1	10-10-05	1000000		40
4	Srinivas	669	1	08-05-05	1000000		30
5	Krishna	668	2	09-10-05	500000	20000	12
6	Deepa	668	3	09-09-07	600000		23
7	Keerthi	668	4	05-06-06	600000		24
8	Aravind	671	1	21-01-06	800000	600000	30
9	Srikanth	668	8	18-11-06	400000	500000	34
10	Suresh	667	3	12-12-08	120000		23
11	Rahul	667	9	11-03-08	80000		30
12	Kumar	667	4	16-03-08	120000		20

Procedure:

- We create the table using CREATE TABLE command.
- Specify the column name and its datatype.

//CREATION OF TABLE NAMED Location

```
CREATE TABLE Location (LCode number (3) PRIMARY KEY,
                       Name varchar2 (10)
);
```

//CREATION OF TABLE NAMED Job

```
CREATE TABLE Job (JCode number (3) PRIMARY KEY,
                  Name varchar2 (10)
);
```

//CREATION OF TABLE NAMED Dept

```
CREATE TABLE Dept (Deptno number (2) PRIMARY KEY,
                   Name varchar2 (10),
                   Location number (3)   REFERENCES    LOCATION (LCODE)
);
```

Regd. No: | | | | | | | | | | |

//CREATION OF TABLE NAMED Employee

```
CREATE TABLE Employee(Empno number(2) PRIMARY KEY, Ename varchar2(10),
                      Job number(3) REFERENCES   JOB(JCODE),
                      Mgr_no number(2) REFERENCES EMPLOYEE(EMPNO),
                      Hiredate date,
                      Salary number(10),
                      Commission number(8),
                      Deptno number(2) REFERENCES   DEPT(DEPTNO)
);
```

Insertion of Data in the *JOB* Table

CODE	NAME
667	Clerk
668	Staff
669	Analyst
671	Manager
672	President

Procedure:

- We insert the data into table using INSERT INTO command.
- We have to insert the data in the same order of columns we created the table.

Sample Queries:

```
INSERT INTO Job VALUES (667,'Clerk');
INSERT INTO Job VALUES (668,'Staff');
INSERT INTO Job VALUES (669,'Analyst');
INSERT INTO Job VALUES (671,'Manager');
INSERT INTO Job VALUES (672,'President');
```

Insertion of Data in the *LOCATION* Table

CODE	NAME
122	Kakinada
124	Hyderabad
123	Bangalore
167	Vijayawada

Sample:

```
INSERT INTO location VALUES (122,'Kakinada');
INSERT INTO location VALUES (124,'Hyderabad');
INSERT INTO location VALUES (123,'Bangalore');
INSERT INTO location VALUES (167,'Vijayawada');
```

<u>Insertion of Data in the *Dept* Table</u>

DEPTNO	NAME	LOCATION
10	Accounting	122
20	Research	124
30	Sales	123
40	Operations	167
12	Research	122
13	Sales	122
14	Operations	122
23	Sales	124
24	Operations	124
34	Operations	123
43	Sales	167

Sample Queries:

```
INSERT INTO dept VALUES (10,'Accounting',122);

INSERT INTO dept VALUES (20,'Research',124);

INSERT INTO dept VALUES (30,'Sales',123);

INSERT INTO dept VALUES (40,'Operations',167);

INSERT INTO dept VALUES (12,'Research',122);

INSERT INTO dept VALUES (13,'Sales',122);

INSERT INTO dept VALUES (14,'Operations',122);

INSERT INTO dept VALUES (23,'Sales',124);

INSERT INTO dept VALUES (24,'Operations',124);

INSERT INTO dept VALUES (34,'Operations',123);

INSERT INTO dept VALUES (43,'Sales',167);
```

Regd. No:

Insertion of Data in the *Employee* Table

EMP NO	ENAME	JOB	MGR_NO	HIREDATE	SALARY	COMMISSION	DEPTNO
1	Venkat	672		01-02-06	1200000	10000	40
2	Nirmala	671	1	02-04-07	800000	50000	20
3	Pradeep	669	1	10-10-05	1000000		40
4	Srinivas	669	1	08-05-05	1000000		30
5	Krishna	668	2	09-10-05	500000	20000	12
6	Deepa	668	3	09-09-07	600000		23
7	Keerthi	668	4	05-06-06	600000		24
8	Aravind	671	1	21-01-06	800000	600000	30
9	Srikanth	668	8	18-11-06	400000	500000	34
10	Suresh	667	3	12-12-08	120000		23
11	Rahul	667	9	11-03-08	80000		30
12	Kumar	667	4	16-03-08	120000		20

Sample Queries:

```
INSERT INTO employee VALUES (1,'Venkat',672,null,'01-feb-2006',1200000,10000,40);

INSERT INTO employee VALUES (2,'Nirmala',671,1,'02-apr-2007',800000,50000,20);

INSERT INTO employee VALUES (3,'Pradeep',669,1,'10-oct-2005',1000000,null,40);

INSERT INTO employee VALUES (4,'Srinivas',669,1,'08-may-2005',1000000,null,30);

INSERT INTO employee VALUES (5,'Krishna',668,2,'09-oct-2005',500000,20000,12);

INSERT INTO employee VALUES (6,'Deepa',668,3,'09-sep-2007',600000,null,23);

INSERT INTO employee VALUES (7,'Keerthi',668,4,'05-jun-2006',600000,null,24);

INSERT INTO employee VALUES (8,'Aravind',671,1,'21-jan-2006',800000,600000,30);

INSERT INTO employee VALUES (9,'Srikanth',668,8,'18-nov-2006',400000,500000,34);

INSERT INTO employee VALUES (10,'Suresh',667,3,'12-dec-2008',120000,null,23);

INSERT INTO employee VALUES (11,'Rahul',667,9,'11-mar-2008',80000,null,30);

INSERT INTO employee VALUES (12,'Kumar',667,4,'16-mar-2008',120000,null,20);
```

Regd. No:

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 4 DATE:

Create These Tables

```
create table Sailors(
        sid       int not null constraint sailors_pk primary key,
        sname     varchar2(20),
        rating    int,
        age       decimal(4,1)
);
create table Boats(
        bid       int not null constraint boat_pk primary key,
        bname     varchar2(20),
        color     varchar2(20)
);
create table Reserves(
        sid       int,
        bid       int,
        day       date,
        primary key (sid,bid,day),
        foreign key (sid) references Sailors(sid)
                ON DELETE CASCADE,
        foreign key (bid) references Boats(bid)
                ON DELETE CASCADE
);
```

Insert the following data in those tables

SAILORS			
SID	**SNAME**	**RATING**	**AGE**
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Dunkon	9	40
85	Ardhar	3	25.5
95	Bob	3	63.5

BOATS		
BID	**BNAME**	**COLOR**
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

RESERVES		
SID	**BID**	**DAY**
22	101	10-10-1998
22	102	10-10-1998
22	103	10-08-1998
22	104	10-07-1998
31	102	11-10-1998
31	103	11-06-1998
31	104	11-12-1998
64	101	09-05-1998
64	102	09-08-1998
74	103	09-08-1998

Regd. No:

Inserting the Data into the Sailors Table

Sample Queries for inserting the data into the Sailors Table

```
insert into Sailors  values(22,'Dustin',7,45.0);
insert into Sailors  values(29, 'Brutus', 1, 33);
insert into Sailors  values(31, 'Lubber', 8, 55.5);
insert into Sailors  values(32, 'Andy', 8, 25);
insert into Sailors  values(58, 'Rusty', 10, 35);
insert into Sailors  values(64, 'Horatio', 7, 35);
insert into Sailors  values(71,'Zorba', 10, 16);
insert into Sailors  values(74, 'Dunkon', 9, 40);
insert into Sailors  values(85, 'Ardhar', 3, 25.5);
insert into Sailors  values(95, 'Bob', 3, 63.5);
```

Inserting the Data into the Boats Table

Sample Queries for inserting the data in Boats Table

```
insert into Boats  values(101,'Interlake','blue');
insert into Boats  values(102, 'Interlake', 'red');
insert into Boats  values(103, 'Clipper', 'green');
insert into Boats  values(104, 'Marine', 'red');
```

Inserting the Data into the Reserves Table

Sample Queries for inserting the data in Reserves Table

```
insert into Reserves values(22, 101,'10-OCT-98');
insert into Reserves values(22, 102,'10-OCT-98');
insert into Reserves values(22, 103,'10-AUG-98');
insert into Reserves values(22, 104,'10-JUL-98');
insert into Reserves values(31, 102,'11-OCT-98');
insert into Reserves values(31, 103,'11-JUN-98');
insert into Reserves values(31, 104,'11-DEC-98');
insert into Reserves values(64, 101,'09-MAY-98');
insert into Reserves values(64, 102,'09-AUG-98');
insert into Reserves values(74, 103,'09-AUG-98');
```

Regd. No:

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 5 DATE:

Queries using aggregate functions

AGGREGATE/GROUP FUNCTIONS

SQL supports five aggregate operators which can be applied on any column, say A, of a relation.

1. COUNT (DISTINCT A): The number of (unique) values in A column.
2. SUM (DISTINCT A): The sum of all (unique) values in A column.
3. AVG (DISTINCT A): The Average of all (unique) values in A column.
4. MAX (A): The maximum values in A column.
5. MIN (A): The minimum values in A column.

COUNT ():

The Count function returns the no. of rows returned by a query.

Syntax:

```
SELECT COUNT(Column_name)
FROM table_name
WHERE Condition;
```

SUM():

The sum function adds the column values in a query.

Syntax:

```
SELECT SUM(Column_name)
FROM table_name
WHERE condition;
```

AVG():

AVG() Function is used to calculate average value of set of values.

Syntax:

```
SELECT AVG(Column_name)
FROM table_name
WHERE condition;
```

MAX function:

This function is used to find min values from a set of values.

Syntax:

```
SELECT MAX(Column_name)
FROM table_name
WHERE condition;
```

MIN Function:

This function is used to find min value from a set of values.

Syntax:

```
SELECT MIN(Colmun_name)
```

Regd. No:

FROM table_name
WHERE Condition;

SQL Queries;

1. Display the total number of employee working in the company.
   Query:
   SELECT count(*)
   FROM employee
   Result:

COUNT(*)
12

2. Display the total salary being paid to all employees.
   Query:
   SELECT sum(salary)
   FROM employee
   Result:

SUM(SALARY)
7220000

3. Display the maximum salary from emp table.
   Query:
   SELECT max(salary)
   FROM employee
   Result:

MAX(SALARY)
1200000

4. Display the minimum salary from emp table.
   Query:
   SELECT min(salary)
   FROM employee
   Result:

MIN(SALARY)
80000

5. Display the average salary from emp table.
   Query:
   SELECT avg(salary)
   FROM employee

Result:

AVG(SALARY)
601666.6666666667

6. Display the maximum salary being paid to CLERK.
   Query:
   ```
   SELECT max(salary)
   FROM employee
   WHERE job = (SELECT jcode
                FROM job
                WHERE name = 'Clerk')
   ```
   Result:

MAX(SALARY)
120000

7. Display the maximum salary being paid to depart number 20.
   Query:
   ```
   SELECT max(salary)
   FROM employee
   WHERE deptno = 20
   ```
   Result:

MAX(SALARY)
800000

8. Display the average salary drawn by MANAGERS.
   Query:
   ```
   SELECT avg(salary)
   FROM employee
   WHERE job = (SELECT jcode
         FROM job
         WHERE name = 'Manager')
   ```
   Result:

AVG(SALARY)
800000

9. Display the total salary drawn by ANALYST working in depart number 40.
   Query:
   ```
   SELECT sum(salary)
   FROM employee
   WHERE job = (SELECT jcode
                FROM job
   ```

```
                    WHERE name = 'Analyst') AND
          deptno = 40
```

Result:

SUM(SALARY)
1000000

10. Display the names of all the employees who are working in depart number 20.

    Query:

```
SELECT ename
FROM employee
WHERE deptno = 20
```

Result:

ENAME
Nirmala
Kumar

11. Display the employee no, salary and total salary for all the employees

    Query:

```
SELECT empno,salary,salary+commission Totalsalary
FROM employee
```

Result:

EMPNO	SALARY	TOTALSALARY
1	1200000	1210000
2	800000	850000
3	1000000	-
4	1000000	-
5	500000	520000
6	600000	-
7	600000	-
8	800000	1400000
9	400000	900000
10	120000	-
11	80000	-
12	120000	-

12. Display the names of all the employees whose job code is 667 and drawing a salary more than 100000.

    Query:

```
SELECT ename
```

```
FROM employee
WHERE job = 667 and salary > 100000
```

Result:

ENAME
Suresh
Kumar

13. Display the employee number and name who are earning comm.

Query:

```
SELECT empno,ename
FROM employee
WHERE commission is not null
```

Result:

EMPNO	ENAME
1	Venkat
2	Nirmala
5	Krishna
8	Aravind
9	Srikanth

14. Display the employee number and name who do not earn any comm.

Query:

```
SELECT empno,ename
FROM employee
WHERE commission is null
```

Result:

EMPNO	ENAME
3	Pradeep
4	Srinivas
6	Deepa
7	Keerthi
10	Suresh
11	Rahul
12	Kumar

15. Display the names of employees who are working as Clerks, Salesman or Analyst and drawing a salary more than 250000.

Query:

```
SELECT ename
FROM employee
```

Regd. No:

```
WHERE job in (SELECT jcode
                    FROM job
                    WHERE name in ('Clerk' ,'Salesman','Analyst'))and
          salary > 250000
```

Result:

ENAME
Pradeep
Srinivas

16. Display the names of the employees who are working in the company for the past 15 years.

Query:

```
SELECT ename
FROM employee
WHERE TO_CHAR(sysdate,'YYYY')-TO_CHAR(hiredate,'YYYY')>=15;
```

Result:

ENAME
Venkat
Nirmala
Pradeep
Srinivas
Krishna
Deepa
Keerthi
Aravind
Srikanth

17. Display the names of employees working in depart number 10 or 20 or 40 or employees working as CLERKS, SALESMAN or ANALYST.

Query:

```
SELECT ename
FROM employee
WHERE deptno =ANY(10,20,40)
OR job =ANY(SELECT jcode
                 FROM job
                  WHERE name in ('Clerk','Salesman','Analyst'))
```

Result:

ENAME
Venkat
Nirmala
Pradeep
Srinivas
Suresh

Regd. No:

Rahul
Kumar

18. Display the names of employees whose name starts with alphabet S.

Query:

```
SELECT ename
FROM employee
WHERE ename LIKE 'S%'
```

Result:

ENAME
Srinivas
Srikanth
Suresh

19. Display the Employee names for employees whose name ends with Alphabet S.

Query:

```
SELECT ename
FROM employee
WHERE ename LIKE '%s'
```

Result:

ENAME
Srinivas

20. Display the names of employees whose names have second alphabet A in their names.

Query:

```
SELECT ename
FROM employee
WHERE ename LIKE '_a%'
```

Result:

ENAME
Rahul

21. select the names of the employee whose names is exactly five Characters in length.

Query:

```
SELECT ename
FROM employee
WHERE ename LIKE '_____'
```

Result:

ENAME
Deepa
Rahul
Kumar

Regd. No:

22. Display the names of the employee who are not working as MANAGERS.

Query:

```
SELECT ename
FROM employee
WHERE mgr_no IS NULL
```

Result:

ENAME
Venkat

23. Display the names of the employee who are not working as SALESMAN OR CLERK OR ANALYST.

Query:

```
SELECT ename
FROM employee
WHERE job NOT IN (SELECT jcode
                  FROM job
                  WHERE name in ('Clerk' ,'Salesman','Analyst'))
```

Result:

ENAME
Venkat
Nirmala
Krishna
Deepa
Keerthi
Aravind
Srikanth

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 6 DATE:

Queries using group by, having, order by classes, sub queries and co-related sub queries. Transaction control language commands: Commit, Rollback, and Savepoint.

Group by clause:

Using group by, we can create groups of related information. Columns used in select clause must be used with group by clause, otherwise it was not a group by expression.

Having clause:

This will work as where clause which can be used only with group by clause because of absence of where clause in group by.

Order by clause:

This will be used to ordering the columns data (ascending or descending).
ASC (default) and DESC – specify the ordering of values, either ascending or descending

Sub queries and co-related sub queries

1. Deptname and No. of employees in each department

   Query

   ```
   SELECT name, count(*)
   FROM employee NATURAL JOIN dept
   GROUP BY name,deptno
   ```

   Result:

NAME	COUNT(*)
Research	2
Operations	1
Operations	2
Sales	3
Sales	2
Research	1
Operations	1

2. No.of employees in each department that has location in Hyderabad and have more than 1 employee

   Query

   ```
   SELECT deptno,count(*)
   FROM employee
   WHERE deptno in (SELECT deptno
                    FROM dept
                    WHERE location = (SELECT lcode
                                      FROM location
   ```

Regd. No:

```
                                WHERE name = 'Hyderabad'))
GROUP BY deptno
HAVING count(*) > 1
```

Result:

DEPTNO	COUNT(*)
20	2
23	2

3. No.of employees for each job name

   Query

```
SELECT count(*) NOEMP,j.name JNAME
FROM employee e,job j
WHERE j.jcode=e.job
GROUP BY j.name
```

Result:

NOEMP	JNAME
2	Manager
2	Analyst
4	Staff
3	Clerk
1	President

4. Deptname, count of employees of those dept that have a average salary > 40000

   Query

```
SELECT name,count(*) NOEMP
FROM employee NATURAL JOIN dept
GROUP BY deptno,name
HAVING avg(salary)>40000
```

Result:

NAME	NOEMP
Operations	1
Operations	1
Research	1
Research	2
Sales	3
Operations	2
Sales	2

5. For each jobcode give total salary

   Query

```
SELECT job,sum(salary) TOTAL
FROM employee
```

Regd. No:

```
GROUP BY job
```

Result:

JOB	TOTAL
672	1200000
671	1600000
669	2000000
667	320000
668	2100000

6. Sum of salary for clerks for each dept

Query

```
SELECT deptno,sum(salary) TOTAL
FROM employee
WHERE job = (SELECT jcode
                FROM job
                WHERE name = 'Clerk')
GROUP BY deptno
```

Result:

DEPTNO	TOTAL
30	80000
20	120000
23	120000

7. Display the name of the employee who earns highest salary.

Query :

```
SELECT ename
FROM employee
WHERE salary = (SELECT MAX(salary)
                   FROM employee)
```

Result:

ENAME
Venkat

8. Display the employee number and name for employee working as clerk and earning highest salary among clerks.

Query

```
SELECT empno,ename
FROM employee
WHERE job =(SELECT jcode
               FROM job
               WHERE name='Clerk') AND
                     salary =(SELECT max(salary)
                                 FROM employee
```

Regd. No:

```
                                    WHERE job=(SELECT jcode
                                               FROM job
                                               WHERE name ='Clerk'))
```

Result:

EMPNO	ENAME
10	Suresh
12	Kumar

9. Display the names of employee whose job role is ‘Staff’ and earns salary more than the highest salary of any ‘Clerk’.

   Query

```
SELECT ename
FROM employee
WHERE job = (SELECT jcode
             FROM job
             WHERE name = 'Staff') and
      salary > any (SELECT salary
                    FROM employee
                    WHERE job = (SELECT jcode
                                 FROM job
                                 WHERE name = 'Clerk'))
```

   Result:

ENAME
Krishna
Deepa
Keerthi
Srikanth

10. Display the names of clerks who earn a salary more than the lowest salary of any employee whose job role is Staff.

   Query

```
SELECT ename
FROM employee
WHERE job = (SELECT jcode
             FROM job
             WHERE name = 'clerk') AND
      salary > (SELECT min(salary)
                FROM employee
                WHERE job = (SELECT jcode
                             FROM job
                             WHERE name = 'staff'))
```

   Result:

   no data found

Regd. No:

11. Display the names of the employees who earn highest salary in their respective departments.

Query

```
SELECT ename
FROM employee e
where salary =(SELECT max(salary)
               FROM employee
               WHERE deptno = e.deptno)
```

Result:

ENAME
Venkat
Nirmala
Srinivas
Krishna
Deepa
Keerthi
Srikanth

12. Display the names of the employees who earn highest salaries in their respective job groups

Query

```
SELECT ename
FROM employee e
WHERE salary =(SELECT max(salary)
               FROM employee
               WHERE job = e.job)
```

Result:

ENAME
Venkat
Nirmala
Pradeep
Srinivas
Deepa
Keerthi
Aravind
Suresh
Kumar

13. Display the employee names who are working in 'Operations' department.

Query

```
SELECT ename
FROM employee
WHERE deptno in (SELECT deptno
                 FROM dept
```

```
                        WHERE name = 'Operations')
```

Result:

ENAME
Venkat
Pradeep
Keerthi
Srikanth

14. Display the employee names who are working in Kakinada

Query

```
SELECT ename
FROM employee
WHERE deptno in (SELECT deptno
                        FROM dept
                        WHERE location = (SELECT lcode
                                                FROM location
                                                WHERE name = 'Kakinada'))
```

Result:

ENAME
Krishna

15. Display the names of employees from department number 30 with salary greater than that of any employee working in other department.

Query

```
SELECT ename
FROM employee
WHERE deptno = 30 AND
        salary > any (SELECT salary
                        FROM employee
                        WHERE deptno != 30)
```

Result:

ENAME
Srinivas
Aravind

16. Display the names of the employees from department number 30 with salary greater than that of all employee working in other departments.

Query

```
SELECT ename
FROM employee
WHERE deptno = 30 AND
        salary > ALL (SELECT salary
```

Regd. No:

```
                    FROM employee
                    WHERE deptno != 30)
```

Result:

no data found

17. Display the names of the employees from department number 40 with salary greater than that of all employee working in other departments.

Query

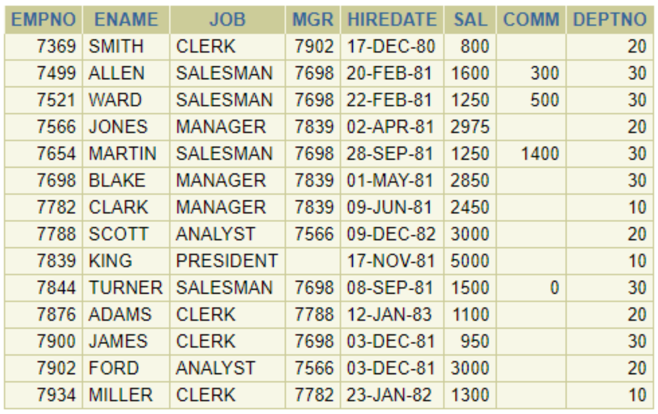
```
SELECT ename
FROM employee
WHERE deptno = 40 AND
        salary > ALL (SELECT salary
                    FROM employee
                    WHERE deptno != 40)
```

Result:

ENAME
Venkat

Transaction control language commands: Commit, Rollback, and Savepoint.

1. select * from emp;



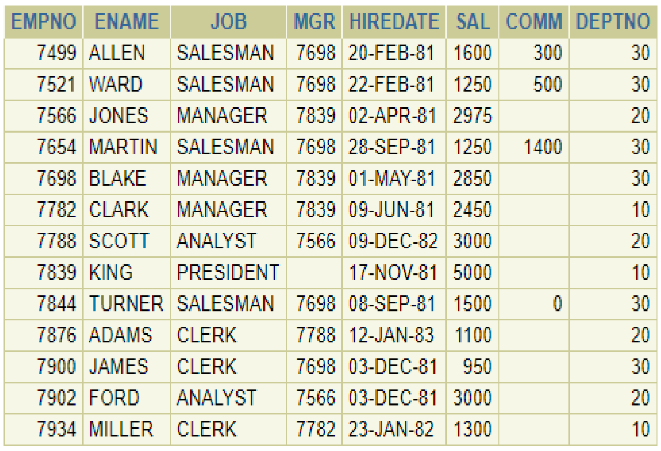
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Result: 14 rows selected.

2. delete from emp where empno=7369;

   Result: 1 row deleted.

3. select * from emp;



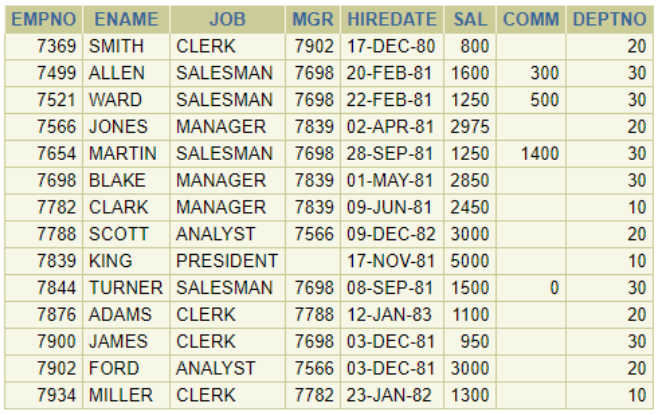
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Result: 13 rows selected.

4. rollback;

   Result: rollback completed.

5. select * from emp;



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Result: 14 rows selected.

6. delete from emp where empno<=7550;

   Result: 3 rows deleted.

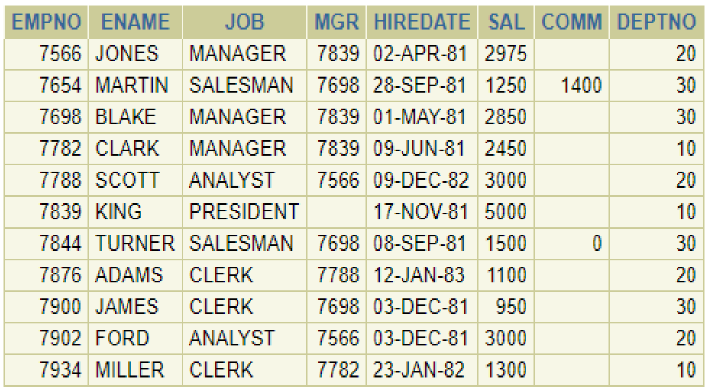
7. commit;

   Result: commit completed.

8. rollback;

   Result: rollback completed.

9. select * from emp;



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

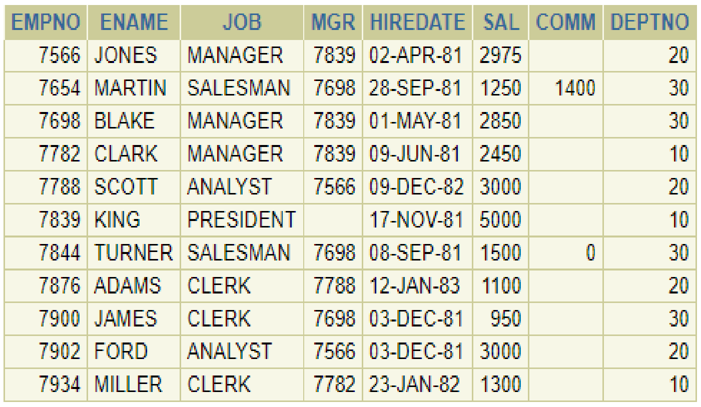
Result: 11 rows selected.

10. insert into emp values(7369,'smith','clear',7902,'17-DEC-1980',800,null,20);

    Result: 1 row inserted.

Regd. No:

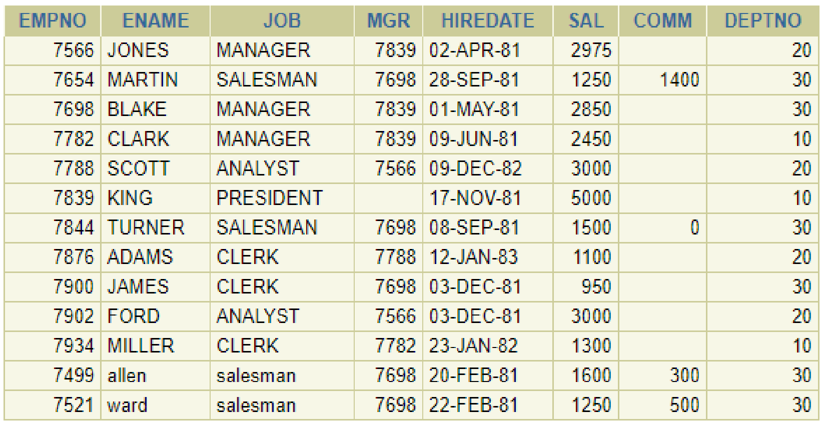
11. insert into emp values(7499,'allen','salesman',7698,'20-FEB-1981',1600,300,30);
    Result: 1 row inserted.
12. insert into emp values(7521,'ward','salesman',7698,'22-FEB-1981',1250,500,30);
    Result: 1 row inserted.
13. savepoint firstrow;
    Result: savepoint created.
14. delete from emp where empno=7369;
    Result: 1 row deleted.
15. savepoint secondrow;
    Result: savepoint created.
16. delete from emp where empno=7499;
    Result: 1 row deleted.
17. savepoint thirdrow;
    Result: savepoint created.
18. delete from emp where empno=7521;
    Result: 1 row deleted.
19. select * from emp;



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Result: 11 rows selected.

20. rollback to secondrow;
    Result: rollback completed.
21. Select * from emp;



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10
7499	allen	salesman	7698	20-FEB-81	1600	300	30
7521	ward	salesman	7698	22-FEB-81	1250	500	30

Result: 13 rows selected.

Regd. No:

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 7 DATE:

Write a PL/SQL Code using Basic Variable, Anchored Declarations, and Usage of Assignment Operation.

Introduction

- The PL/SQL programming language was developed by **Oracle Corporation** in the late 1980s as procedural extension language for SQL.
- PL/SQL is not a stand-alone programming language; it is a tool within the Oracle programming environment.
- PL/SQL is a **block-structured** language that is PL/SQL programs are divided into logical blocks of code.

Structure of a PL/SQL block:

```
DECLARE
        <declaration/initialization of variables>
BEGIN
        <executable statements>
EXCEPTION
        <exception handling code>
END;  ---→ end of the program.
```

Declarations: This section starts with the keyword **DECLARE**. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.

Executable block: This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code

Exception Handling: This section starts with the keyword EXCEPTION. This section is again optional and contains exception(s) that handle errors in the program.

DATA TYPES

1. Number
2. char
3. varchar2
4. date
5. %type →it is to continue the variable data type same as in table

VARIABLE DECLARATION

Syntax for declaring a variable is:

```
variable_name  datatype:=initial_value;
```

Initializing Variables

Regd. No:

```
    a integer := 10;
```

Example:

```
    msg char(20) := ‘hello world’;
```

Control structures

- Conditional statements
- Repetitive statements
- Case statements

Conditional statements

1. Simple IF

Syntax:

```
    IF condition THEN
        Statements;
    END IF;
```

2. IF-ELSE

Syntax:

```
    IF condition THEN
        Statements;
    ELSE
        Statements;
    END IF;
```

3. IF-ELSE LADDER

Syntax:

```
    IF condition THEN
        Statements;
    ELSIF condition THEN
        Statements;
    …
    ELSE
        Statements;
    END IF;
```

4. CASE STATEMENTS

Syntax:

```
    Case selector
        WHEN 'value1' THEN Statements;
        WHEN 'value2' THEN Statements;
        WHEN 'value3' THEN Statements;
        ...
        ELSE Statements; → default case
    END CASE;
```

LOOPS

1. BASIC LOOP

Regd. No:

2. WHILE LOOP
3. FOR LOOP

1. BASIC LOOP

Syntax:

```
LOOP
        Statements;
END LOOP;
```

2. WHILE LOOP

Syntax:

```
WHILE condition LOOP
        Statements;
END LOOP;
```

3. FOR LOOP

Syntax:

```
FOR variable_name IN/IN REVERSE start_value .. end_value
LOOP
        Statements;
END LOOP;     //for j in reverse 10..1
                Loop
                end loop
```

EXCEPTIONS

Error condition during a program execution is called an exception in PL/SQL.

There are two types of exceptions:

- System-defined exceptions
- User-defined exceptions

Syntax:

```
DECLARE
        <Declarations section>
BEGIN
        <Executable Statements>
EXCEPTION
        WHEN exception1 THEN
                exception1-handling-statements;
        WHEN exception2 THEN
                exception2-handling-statements;
        WHEN exception3 THEN
                exception3-handling-statements;
        ........
END;
```

Regd. No:

1. Write a PL/SQL Program to display "Hello World" message

Aim:- To display "Hello World" message

Source Code:-

```
        Set serveroutput on;
        begin
              dbms_output.put_line('Hello World');
        end;
```

Output:-

```
Hello World
```

2. Write a PL/SQL program to find given number is even or odd

Aim:- To find given number is even or odd

Source code:-

```
        Set serveroutput on;
        Declare
                n int:= :n;
        begin
                if mod(n,2)=0 then
                        dbms_output.put_line('Given number is even');
                else
                        dbms_output.put_line('Given  number is odd');
                end if;
        end;
```

Output:-

```
:N 13
Given  number is odd
```

3. Write a PL/SQL program to find biggest of 3 numbers

Aim:- To find biggest of 3 numbers

Source Code:-

```
        Set serveroutput on;
        Declare
                a int:= :a;
                b int:= :b;
                c int:= :c;
        begin
                if (a>b) and (a>c) then
                        dbms_output.put_line('A is big');
                elsif (b>c) then
                        dbms_output.put_line('B is big');
                else
                        dbms_output.put_line('C is big');
                end if;
        end;
```

Output:-

```
:A 13
:B 12
```

Regd. No:

```
:C 10
A is big
```

4. Write a PL/SQL program to find a sum of 1 to n number

Aim:- To find a sum of 1 to n number

Source Code:-

```
        Set serveroutput on;
        Declare
                n int:=:n;
                s int:=0;
                i int:=1;
        begin
                loop
                        if i<=n then
                                s:=s+i;
                                i:=i+1;
                        else
                                Exit;
                        end if;
                end loop;
               dbms_output.put_line('Sum of 1 to ' || n || ' number is ' || s);
        end;
```

Output:-

```
:N 12
Sum of 1 to 12 number is 78
```

<u>Using FOR Loop</u>

```
        Declare
               n int:=:n;
               s int:=0;
               i int;
        begin
               for i in 1..n
               loop
                       s := s+i;
               end loop;
               dbms_output.put_line('Sum of 1 to ' || n || ' number is '|| s);
        end;
```

output:-

```
:N 12
Sum of 1 to 12 number is 78
```

5. Write PL/SQL program to find multiplication table for a given 'n' value

Aim:- To find multiplication table for a given 'n' value

Source Code:-

```
        Set serveroutput on;
        Declare
```

Regd. No: | | | | | | | | | | |

```
                    n int:=:n;
                    res int:=0;
                    i int:=1;
          begin
                    loop
                              res:=(n*i);
                              dbms_output.put_line(n||'*'||i||'='||res);
                              Exit when i=10;
                              i:=i+1;
                    end loop;
          end;
```

output:-

```
:N 12
12*1=12
12*2=24
12*3=36
12*4=48
12*5=60
12*6=72
12*7=84
12*8=96
12*9=108
12*10=120
```

<u>Using FOR Loop</u>

```
          Declare
                    n int:= :n;
                    s int:=0;
                    i int;
          begin
                    for i in 1..10
                    loop
                              dbms_output.put_line(n || ‘ X ‘ || i || ‘ = ‘  || n*i );
                    end loop;
          end;
```

Output:-

```
:N 12
12 X 1 = 12
12 X 2 = 24
12 X 3 = 36
12 X 4 = 48
12 X 5 = 60
12 X 6 = 72
12 X 7 = 84
12 X 8 = 96
12 X 9 = 108
12 X 10 = 120
```

Regd. No:

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 8 **DATE:**

1. Write PL/SQL code to find Factorial of a given number.

Aim:- To find Factorial of a given number.

Source Code:-

```
declare
        n int;
        i int;
        f number(8):=1;
begin
        n:=:n;
        for i in 1..n loop
                f:=f*i;
        end loop;
        dbms_output.put_line('the factorial value is =' ||f);
end;
/
```

Output:-

```
:N 7
the factorial value is =5040
```

2. Write PL/SQL program to find out the reverse of a given number

Aim:- To find out the reverse of a given number

Source Code:-

```
Declare
        n int:=:n;
        rem int;
        rev int:=0;
begin
        while n!=0 loop
                rem:=mod(n,10);
                rev:=rem+(rev*10);
                n:=trunc(n/10);
        end loop;
        dbms_output.put_line('Reverse Number is:'||rev);
end;
/
```

Output:-

```
:N 1312
Reverse Number is:2131
```

Regd. No:

3. Write a PL/SQL program to find the next prime number for a given n value

Aim:- To find the next prime number for a given n value

Source Code:-

```
Declare
        n int:=&n;
        F int:=1;
begin
        for i in n+1..3000 loop
                for j in 2..trunc(i/2)loop
                        if mod(i,j)=0 then
                                f:=0;
                                exit;
                        end if;
                end loop;
                if f=1 then
                        dbms_output.put_line(i||' is next prime');
                        exit;
                end if;
                f:=1;
        end loop;
end;
/
```

Output:-

```
:N 12
13 is next prime
```

4. Write a PL/SQL program to print all the prime numbers series upto a given number 'n'

Aim:- prime number up to n

Source Code:-

```
Declare
        n int:= :n;  i int:=2;  j int:=2;  count1 int:=0;
begin
        for i in 2..n  loop
                count1:=0;
                for j in 2..trunc(i/2) loop
                        if (mod(i,j)=0) then
                                count1:=1;  exit;
                        end if;
                end loop;
                if count1=0 then
                        dbms_output.put(i||' ');
                end if;
        end loop;
        dbms_output.put_line('');
end;
```

Output:-

```
:N 14
2 3 5 7 11 13
```

Regd. No:

5. Write a PL/SQL program to check the given string is Palindrome

Aim:- To check the given string is Palindrome

Source Code:-

```
Declare
        str1 varchar2(50):= :string;
        str2 varchar2(50);
        len number;
begin
         len:=length(str1);
         for i in reverse 1..len
         loop
                str2:=concat(str2,substr(str1,i,1));
         end loop;
         if str1=str2 then
                dbms_output.put_line('Given String '||str1||' is PALINDROME');
         else
                dbms_output.put_line('Given String '||str1||' is NOT PALINDROME');
         end if;
end;
/
```

Output:-

```
:STRING 1221
Given String 1221 is PALINDROME
```

6. Write PL/SQL program to read one subject marks from the keyboard and wish the student by grade based on marks by student using case statement.

Aim:- To read one subject marks from the keyboard and wish the student by grade based on marks by student using case statement

Source Code:-

```
Declare
M number:=:marks;
begin
        case
                when M<=100 and M>=90 then
                        dbms_output.put_line('Grade is : O');
                when M<90 and M>=80 then
                        dbms_output.put_line('Grade is : S');
                when M<80 and M>=70 then
                        dbms_output.put_line('Grade is : A');
                when M<70 and M>=60 then
                        dbms_output.put_line('Grade is : B');
                when M<60 and M>=50 then
                        dbms_output.put_line('Grade is : C');
                when M<50 and M>=40 then
                        dbms_output.put_line('Grade is : D');
        else
                dbms_output.put_line('--You Are fail--');
```

```
            end case;
end;
/
```

Output:-

```
:MARKS 100
Grade is : O
```

7. Write PL/SQL code to find specific Employee salary for given Empno from EMP table.

Aim:- To find specific Employee salary for given Empno from EMP table.

Source Code:-

```
DECLARE
            VAR_SALARY NUMBER(16);
            VAR_EMPNO NUMBER(16):= :Empno;
BEGIN
              SELECT SALARY INTO VAR_SALARY FROM EMPLOYEE WHERE EMPNO=VAR_EMPNO;
              DBMS_OUTPUT.PUT_LINE('THE EMPLOYEE OF '||VAR_EMPNO||' HAS SALARY
            '||VAR_SALARY);
END;
/
```

Output:-

```
:EMPNO 12
THE EMPLOYEE OF 12 HAS SALARY 120000
```

Regd. No:

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 9 DATE:

1. Write a PL/SQL program to handle divide by zero exception

Aim:-A PL/SQL program to handle divide by zero exception

Source Code:-

```
Declare
        a number:=&a;
        b number:=0;
        c number;
begin
        c:=a/b;
Exception
        When zero_divide then
                dbms_output.put_line('Attempt to divide by zero');
        when others then
                dbms_output.put_line('An Exception is raised in program');
end;
/
```

Output:-

```
:A 10
Attempt to divide by zero
```

2. Write PL/SQL program to create user defined exception and raising and handle the exception property

Aim:-A PL/SQL program to create user defined exception and raising and handle the exception property

Source Code:-

```
Declare
        n number;
        myex exception;
begin
        n:=:n;
        for i in 1..n loop
                dbms_output.put_line('i value is:'||i);
                if (i=n) then
                        raise myex;
                end if;
        end loop;
        exception
                when myex then
                raise_application_error(-20018,'user defined error is raised----');
end;
/
```

Output:-

```
:N 10
ORA-20018: user defined error is raised----
```

Regd. No:

3. Write a PL/SQL program to display a sailor details with sid=31

Aim:- A PL/SQL program to display a sailor details with sid=31

Source Code:-

```
Declare
        v_sid sailors.sid%type:=31;
        v_sname sailors.sname%type;
        v_rating sailors.rating%type;
        v_age sailors.age%type;
begin
        select sname,rating,age into v_sname,v_rating,v_age from sailors where sid=v_sid;
        dbms_output.put_line('Sailor name is: '||v_sname);
        dbms_output.put_line('Sailor rating is: '||v_rating);
        dbms_output.put_line('Sailor age is: '||v_age);
end;
/
```

Output:-

```
Sailor name is: Lubber
Sailor rating is: 8
Sailor age is: 55.5
```

4. Write a PL/SQL program to perform different DML operations on a table

Aim:- A PL/SQL program to perform different DML operations on a table

Source Code:-

```
Declare
        v_rating sailors.rating%type;
Begin
        insert into sailors values(27,'Smith',7.0,25);
        select rating into v_rating from sailors where sid=27;
        dbms_output.put_line('New Sailors rating is:'||v_rating);
        update sailors set rating=rating+2.5 where sid=27;
        select rating into v_rating from sailors where sid=27;
        dbms_output.put_line('New sailor updated rating is:'||v_rating);
        delete from sailors where sname='Smith';
        dbms_output.put_line('Sailor with sname Smith is deleted');
        commit;
end;
/
```

Output:-

```
New Sailors rating is:7
New sailor updated rating is:10
Sailor with sname Smith is deleted
```

5. Write PL/SQL program to display all information about a sailor using % rowtype data type

Aim:-A PL/SQL program to display all information about a sailor using % rowtype data type

Source Code:-

Regd. No:

```
Declare
        v_id sailors.sid%type;
        srow sailors%rowtype;
begin
        v_id:= &v_id;
        select * into srow from sailors where sid=v_id;
        dbms_output.put_line('Sailors name is:'||srow.sname);
        dbms_output.put_line('Sailors rating is:'||srow.rating);
        dbms_output.put_line('Sailors age is:'||srow.age);
exception
        when no_data_found then
        dbms_output.put_line('No sailors with given sid');
end;
/
```

Output:-

```
:V_ID 95
Sailors name is:Bob
Sailors rating is:3
Sailors age is:63.5
```

6. Write PL/SQL program to create an explicit cursor which displays a set of records

Aim:-A PL/SQL program to create an explicit cursor which displays a set of records

Source Code:-

```
Declare
        sr sailors%rowtype;
        cursor sc is
        select *from sailors;
begin
        open sc;
        loop
                fetch sc into sr;
                exit when sc%notfound;
                dbms_output.put_line('Sailor id: '||sr.sid||' Sname is: '||sr.sname);
        end loop;
                dbms_output.put_line('Total rows in the cursor: '||sc%rowcount);
        close sc;
end;
/
```

Output:-

```
Sailor id: 22 Sname is: Dustin
Sailor id: 29 Sname is: Brutus
Sailor id: 31 Sname is: Lubber
Sailor id: 32 Sname is: Andy
Sailor id: 58 Sname is: Rusty
Sailor id: 64 Sname is: Horatio
Sailor id: 71 Sname is: Zorba
Sailor id: 74 Sname is: Dunkon
Sailor id: 85 Sname is: Ardhar
Sailor id: 95 Sname is: Bob
Total rows in the cursor: 10
```

7. Write PL/SQL program to print welcome message after insertion for each row in sailors table using trigger

Aim:-A PL/SQL program to print welcome message after insertion for each row in sailors table using trigger

Source Code:-

```
create or replace trigger trg1 after insert on sailors
for each row
begin
        dbms_output.put_line('---Welcome to new sailors---');
end;
/
```

Output:-

```
Trigger created.
```

Once Trigger Created successfully

Add a Sailor to the database (Later Delete it Also)

```
insert into sailors values(27, 'Smith', 6, 24)
```

Output:

```
---Welcome to new sailors---

1 row(s) inserted
```

8. Write PL/SQL program to convert sailor name to uppercase and print error message when rating below zero and above 10 before insert or update on each row in sailors table using triggers

Aim:-A PL/SQL program to convert sailor name to uppercase and print error message when rating below zero and above 10 before insert or update on each row in sailors table using triggers

Source Code:-

```
create or replace  trigger TRG2 before insert or update on sailors
for each row
begin
        if :new.rating < 0 or :new.rating >10 then
                raise_application_error(-20027,'Invalid rating value---');
        end if;
        dbms_output.put_line('Sailor inserted by changing to upper case');
        :new.sname:=upper(:new.sname);
end;
/
```

Output:-

```
Trigger created.
```

Once Trigger Created insert a sailor as below

Regd. No: | | | | | | | | | | |

```
insert into sailors values(27,'Smith',7.0,25);
```
output:-

```
Sailor inserted by changing to upper case
---Welcome to new sailors---

1 row(s) inserted.
```

```
insert into sailors values(13,'lilly',-9,30);
```
output:-

```
ORA-20027: Invalid rating value---
ORA-06512: at "SYSTEM.TRG2", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRG2'
1. insert into sailors values(13,'lilly',-9,30);
```

9. write pl/sql program to demonstrate implicit cursor

Aim: To demonstrate implicit cursor

Source code:

```
set serveroutput on;
set verify off;
DECLARE
        var_rows number(5);
BEGIN
        UPDATE sailors
        SET rating= rating + 1 where sid=31;
        IF SQL%NOTFOUND THEN
                dbms_output.put_line('None of the salaries where updated');
        ELSIF SQL%FOUND THEN
                var_rows := SQL%ROWCOUNT;
                dbms_output.put_line('Salaries for ' || var_rows || ' employees are
updated');
        END IF;
END;
/
```
Output:-

```
Sailor inserted by changing to upper case
Salaries for 1 employees are updated

1 row(s) updated.
```

Regd. No:

DATABASE MANAGEMENT SYSTEMS LAB

EXPERIMENT NO: 10DATE:

Procedure and Functions in PL/SQL

Parameters in Procedure and Functions

In PL/SQL, we can pass parameters to procedures and functions in three ways.

1) **IN type parameter**: These types of parameters are used to send values to stored procedures. This type of parameter is a read only parameter.
2) **OUT type parameter**: The OUT parameters are used to send the OUTPUT from a procedure or a function. This is a write-only parameter i..e, we cannot pass values to OUT parameters while executing the stored procedure, but we can assign values to OUT parameter inside the stored procedure and the calling program can receive this output value.
3) **IN OUT parameter**: These types of parameters are used to send values and get values from stored procedures.
NOTE: If a parameter is not explicitly defined a parameter type, then by default it is an IN type parameter.

Function in PL/SQL

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

General Syntax to create a function is

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
RETURN return_datatype;
IS
Declaration_section
BEGIN
Execution_section
Return return_variable;
EXCEPTION
exception section
Return return_variable;
END;
```

1) **Return Type**: The header section defines the return type of the function. The return datatype can be any of the oracle datatype like varchar, number etc.
2) The execution and exception section both should return a value which is of the datatype defined in the header section.
A function can be executed in the following ways.

Regd. No:

Consider **emp_func** is created

1) Since a function returns a value we can assign it to a variable.

employee_name := emp_func;

If 'employee_name' is of datatype varchar we can store the name of the employee by assigning the return type of the function to it.

2) As a part of a SELECT statement

SELECT emp _func FROM dual;

3) In a PL/SQL Statements like,

dbms_output.put_line(emp _func);

This line displays the value returned by the function.

Stored Procedures

A stored procedure or in simple a proc is a named PL/SQL block which performs one or more specific task.

A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists of declaration section, execution section and exception section similar to a general PL/SQL Block.

A procedure is similar to PL/SQL Block but it is named for repeated usage. A procedure may or may not return any value.

General Syntax to create a procedure is:

```
CREATE [OR REPLACE] PROCEDURE proc_name [list of parameters]
IS
   Declaration section
BEGIN
   Execution section
EXCEPTION
  Exception section
END;
```

IS - marks the beginning of the body of the procedure and code between IS and BEGIN forms the Declaration section.

The syntax within the brackets [ ] indicate they are optional. By using CREATE OR REPLACE together the procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.

Regd. No:

There are two ways to execute a procedure.

1) From the SQL prompt.

```
 EXECUTE [or EXEC] procedure_name;
```

2) Within another procedure – simply use the procedure name.

```
procedure_name;
```

1.Write a PL/SQL program biggest of three by using function

Aim:-A PL/SQL program biggest of three by using function

Source Code:-

```
create or replace function bigthree(X IN number,Y IN number,Z IN number)
return number
is
begin
        if X>Y and X>Z then
                return X;
        elsif Y>Z then
                return Y;
        else
                return Z;
        end if;
end;
/
```

Output:-

```
1. Select bigthree (23,2,3) from dual;
```

Output:-

```
Bigthree(23,2,3)
23
```

```
2.Set serveroutput on;
begin
dbms_output.put_line('big three is:'||bigthree(23,2,3));
end;
/
```

Output:-

```
big three is:23
```

2. Write a PL/SQL program to find factorial of a given number by using function

Aim:-A PL/SQL program to find factorial of a given number by using function

Source Code:-

```
Declare
        a number;
        function fact(n in number)
        return number
        is
```

```
          z number;
                    Begin
                              if n<0 then
                                        z:=-1;
                              elsif n=1 or n=0 then
                                        z:=1;
                              else
                                        z:=1;
                                        for i in 1..n loop
                                                  z:=(z*i);
                                        end loop;
                              end if;
                              return z;
                    end;

begin
          a:=&a;
                    dbms_output.put_line(a||' factorial is: '||fact(a));
end;
/
```

Output:-

```
:A 5
5 factorial is: 120
```

3. Write a PL/SQL program to check a given number is Armstrong or not by using functions

Aim:-A PL/SQL program to check a given number is Armstrong or not by using functions

Source Code:-

```
create or replace function armstrong(n in out number)
return number
is
          rem number;
          arms number:=0;
          temp number;
begin
          temp:=n;
          while n>0 loop
                    rem:=mod(n,10);
                    arms:=arms+power(rem,3);
                    n:=trunc(n/10);
          end loop;
          return arms;
end;
/

Declare
          a number;
          b number;
```

```
        c number;
begin
        a:=&a;
        b:=a;
        c:=armstrong(b);
        if a=c then
                dbms_output.put_line('Given number is armstrong number');
        else
                dbms_output.put_line('Given number is not armstrong number');
        end if;
end;
/
```

Output:-

```
:A 153
Given number is armstrong number
```

4. Write a PL/SQL program to display greetings by using procedure programming

Aim:-A PL/SQL program to display greetings by using procedure programming

Source Code:-

```
create or replace procedure greetings
is
begin
        dbms_output.put_line('**Hello** from PL/SQL procedure programming');
end;
```

Output:-

1. Execute Greetings

Output:- **Hello** from PL/SQL procedure programming

2.

```
   begin
        greetings;
   end;
```

Output:- **Hello** from PL/SQL procedure programming

5. Write a PL/SQL program to find the GCD of two numbers by using procedures

Aim:-A PL/SQL program to find the GCD of two numbers by using procedures

Source Code:-

```
Declare
        a number;
        b number;
        procedure gcd(X in out number,Y in out number)
        is
        dif number;
        begin
                if x<Y then
                        X:=X+Y;
                        Y:=X-Y;
```

Regd. No:

```
                    X:=X-Y;
            end if;
            if x=1 or y=1 then
                    dbms_output.put_line('1');
            elsif mod(X,Y)=0 then
                    dbms_output.put_line(Y);
            else
                    dif:=X-Y;
                    gcd(dif,y);
            end if;
        end;
begin
        a:=:a;
        b:=:b;
        dbms_output.put('GCD of '||a||' and '||b||' is: ');
        gcd(a,b);
        dbms_output.put('');
end;
```

Output:-

```
:A 13
:B 12
GCD of 13 and 12 is: 1
```

6. Write a PL/SQL program to find 1 to n prime numbers by using procedure

Aim:-A PL/SQL program to find 1 to n prime numbers by using procedure

Source Code:-

```
create or replace procedure prime(n in number)
is
flag number:=0;
begin
        dbms_output.put_line('Prime number from 1 to n');
        for i in 2..n loop
                for j in 1..trunc(i/2) loop
                        if mod(i,j)=0 then
                        flag:=flag+1;
                        end if;
                end loop;
                if flag=1 then
                        dbms_output.put_line(i);
                end if;
                flag:=0;
        end loop;
end;
```

Output:-

1.Execute prime(20)

Output:-

```
Prime number from 1 to n
```

Regd. No:

```
2
3
5
7
11
13
17
19
```

2.

```
begin
        prime(30);
end;
/
```

Output:-

```
Prime number from 1 to n
2
3
5
7
11
13
17
19
23
29
```

Regd. No: