HINDI VIDYA PRACHAR SAMITI'S

RAMNIRANJAN JHUNJHUNWALA COLLEGE

GHATKOPAR (WEST) MUMBAI 400086

CERTIFICATE

DEPARTMENT OF DATA SCIENCE AND ARTIFICIAL INTELLIGENCE

This is to certify Mr. Krish Kevin Kounder

of BSC Data science and Artificial Intelligence,

Roll No 10132 Has successfully completed the practical of Advance Database Management system during the Academic Year 2025 - 2026.

DATE: _____

Prof. Sujata kotian

PROF-IN-CHARGE

INDEX

SR.NO	PRACTICAL NAME	SIGN
1	Perform Left Join and Right Join using SQL.	
2	Perform Full Join and Cross Join using SQL	
3	Perform transaction including COMMIT, ROLLBACK using SQL query.	
4	Perform Save point transaction using SQL query.	
5	Write an SQL query for lost update problem.	
6	Write an SQL query for temporary inconsistency or dirty read problem.	
7	Write an SQL query for unrepeatable read problem.	
8	Write PL/SQL query for variables and identifier.	
9	Perform SQL query to show the implementation of sub queries.	
10	Write PL/SQL query to show the implementation of Global and Local Variables.	
11	Write PL/SQL query to assign SQL query Results to PL/SQL variables.	
12	Write PL/SQL program for IF THEN statement and IF THEN ELSE statement.	
13	Write PL/SQL program for IF THEN statement and IF THEN ELSE statement.	
14	Write PL/SQL statement for PL/SQL program for CASE statement.	
15	Write PL/SQL query to Implement arithmetic operations.	
16	Write PL/SQL query to implement Relational Operator.	
17	Write PL/SQL query to shoe the implementation of LIKE Operator.	
18	Write PL/SQL query to implement BETWEEN Operators.	
19	Write PL/SQL query for implementation of IN AND IS NULL Operator.	
20	Write PL/SQL query to implement Logical Operations	
21	Write PL/SQL for Implementation of Operation Precedence.	
22	Write PL/SQL query for Basic loop and While loop.	
23	Write PL/SQL query for FOR LOOP statement and REVERSE FOR LOOP statement.	
24	Write PL/SQL query to show the implementation of PL/SQL strings.	
25	Write PL/SQL Query to show the implementation of PL/SQL arrays.	
26	Write PL/SQL query for procedure with IN, OUT, IN OUT Parameters.	
27	Write PL/SQL query to Implement Function.	
28	Write PL/SQL query to implement IMPLICIT cursor and EXPLICIT cursor.	
29	Write PL/SQL query to show the implementation of PL/SQL Records.	
30	Write PL/SQL Query to show the implementation of PL/SQL exception.	
31	Write PL/SQL query to show the implementation of PL/SQL Trigger.	
32	Perform MONGODB CRUD Operations.	

Practical no 1: - Perform Left Join and Right Join using SQL

```
mysql> use nadeem;
Database changed
mysql> create table toy(toy_id int primary key,toy_name char(50),cat_id int);
Query OK, 0 rows affected (0.04 sec)

mysql> insert into toy values(1,'ball',3);
Query OK, 1 row affected (0.01 sec)

mysql> insert into toy values(2,'spring',null);
Query OK, 1 row affected (0.01 sec)

mysql> insert into toy values(3,'mouse',1);
Query OK, 1 row affected (0.01 sec)

mysql> insert into toy values(4,'mouse',4);
Query OK, 1 row affected (0.01 sec)

mysql> insert into toy values(5,'ball',1);
Query OK, 1 row affected (0.01 sec)

mysql> select* from toy;
+--------+----------+--------+
| toy_id | toy_name | cat_id |
+--------+----------+--------+
|      1 | ball     |      3 |
|      2 | spring   |   NULL |
|      3 | mouse    |      1 |
|      4 | mouse    |      4 |
|      5 | ball     |      1 |
+--------+----------+--------+
5 rows in set (0.00 sec)

mysql> create table cat(cat_id int,cat_name char(50));
Query OK, 0 rows affected (0.04 sec)

mysql> insert into cat values(1,'kitty')
    -> ;
Query OK, 1 row affected (0.01 sec)

mysql> insert into cat values(2,'hugo');
Query OK, 1 row affected (0.01 sec)

mysql> insert into cat values(3,'sam');
Query OK, 1 row affected (0.01 sec)

mysql> insert into cat values(4,'misty');
Query OK, 1 row affected (0.01 sec)

mysql> select * from cat;
+--------+----------+
| cat_id | cat_name |
+--------+----------+
|      1 | kitty    |
|      2 | hugo     |
|      3 | sam      |
|      4 | misty    |
+--------+----------+
4 rows in set (0.00 sec)
```

```
mysql> select* from toy JOIN cat ON toy.cat_id=cat.cat_id;
+--------+----------+--------+--------+----------+
| toy_id | toy_name | cat_id | cat_id | cat_name |
+--------+----------+--------+--------+----------+
|      1 | ball     |      3 |      3 | sam      |
|      3 | mouse    |      1 |      1 | kitty    |
|      4 | mouse    |      4 |      4 | misty    |
|      5 | ball     |      1 |      1 | kitty    |
+--------+----------+--------+--------+----------+
4 rows in set (0.00 sec)

mysql> select* from toy LEFT JOIN cat ON toy.cat_id=cat.cat_id;
+--------+----------+--------+--------+----------+
| toy_id | toy_name | cat_id | cat_id | cat_name |
+--------+----------+--------+--------+----------+
|      1 | ball     |      3 |      3 | sam      |
|      2 | spring   |   NULL |   NULL | NULL     |
|      3 | mouse    |      1 |      1 | kitty    |
|      4 | mouse    |      4 |      4 | misty    |
|      5 | ball     |      1 |      1 | kitty    |
+--------+----------+--------+--------+----------+
5 rows in set (0.00 sec)

mysql> select* from toy RIGHT JOIN cat ON toy.cat_id=cat.cat_id;
+--------+----------+--------+--------+----------+
| toy_id | toy_name | cat_id | cat_id | cat_name |
+--------+----------+--------+--------+----------+
|      5 | ball     |      1 |      1 | kitty    |
|      3 | mouse    |      1 |      1 | kitty    |
|   NULL | NULL     |   NULL |      2 | hugo     |
|      1 | ball     |      3 |      3 | sam      |
|      4 | mouse    |      4 |      4 | misty    |
+--------+----------+--------+--------+----------+
5 rows in set (0.00 sec)
```

Practical no 2: -Perform Full Join and Cross Join using SQL

1. FULL JOIN

```
mysql> select* from toy LEFT JOIN cat ON toy.cat_id=cat.cat_id UNION  select* from toy RIGHT JOIN cat ON toy.cat_id=cat.cat_id;
+--------+----------+--------+--------+----------+
| toy_id | toy_name | cat_id | cat_id | cat_name |
+--------+----------+--------+--------+----------+
|      1 | ball     |      3 |      3 | sam      |
|      2 | spring   |   NULL |   NULL | NULL     |
|      3 | mouse    |      1 |      1 | kitty    |
|      4 | mouse    |      4 |      4 | misty    |
|      5 | ball     |      1 |      1 | kitty    |
|   NULL | NULL     |   NULL |      2 | hugo     |
+--------+----------+--------+--------+----------+
6 rows in set (0.00 sec)
```

2. CROSS JOIN

```
mysql> select* from toy CROSS JOIN cat;
+--------+----------+--------+--------+----------+
| toy_id | toy_name | cat_id | cat_id | cat_name |
+--------+----------+--------+--------+----------+
|      1 | ball     |      3 |      4 | misty    |
|      1 | ball     |      3 |      3 | sam      |
|      1 | ball     |      3 |      2 | hugo     |
|      1 | ball     |      3 |      1 | kitty    |
|      2 | spring   |   NULL |      4 | misty    |
|      2 | spring   |   NULL |      3 | sam      |
|      2 | spring   |   NULL |      2 | hugo     |
|      2 | spring   |   NULL |      1 | kitty    |
|      3 | mouse    |      1 |      4 | misty    |
|      3 | mouse    |      1 |      3 | sam      |
|      3 | mouse    |      1 |      2 | hugo     |
|      3 | mouse    |      1 |      1 | kitty    |
|      4 | mouse    |      4 |      4 | misty    |
|      4 | mouse    |      4 |      3 | sam      |
|      4 | mouse    |      4 |      2 | hugo     |
|      4 | mouse    |      4 |      1 | kitty    |
|      5 | ball     |      1 |      4 | misty    |
|      5 | ball     |      1 |      3 | sam      |
|      5 | ball     |      1 |      2 | hugo     |
|      5 | ball     |      1 |      1 | kitty    |
+--------+----------+--------+--------+----------+
20 rows in set (0.00 sec)
```

Practical no 3: Perform transaction including COMMIT, ROLLBACK using SQL query.

```
mysql> select*from employees;
+--------+-----------+---------+------------+---------+
| emp_id | emp_name  | rmp_age | city       | income  |
+--------+-----------+---------+------------+---------+
|    101 | Peter     |      32 | New York   |  200000 |
|    102 | Mark      |      32 | California |  300000 |
|    103 | Donald    |      43 | Arizona    | 1000000 |
|    104 | Obama     |      35 | Florida    | 5000000 |
|    105 | Linklon   |      32 | Georgia    |  250000 |
|    106 | Kane      |      45 | Alaska     |  450000 |
|    107 | Adam      |      35 | California | 5000000 |
|    108 | Marcoulam |      40 | Florida    |  350000 |
+--------+-----------+---------+------------+---------+
8 rows in set (0.00 sec)

mysql> create table orders(order_id int,product_name varchar(50),order_num int,order_date date);
Query OK, 0 rows affected (0.04 sec)

mysql> insert into orders values(1,'Laptop',5544,'2020-02-01'),
    -> (2,'Mouse',3322,'2020-02-11'),
    -> (3,'Desktop',2135,'2020-01-05'),
    -> (4,'Mobile',3432,'2020-02-22'),
    -> (5,'Anti-Virus',5648,'2020-03-10');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select*from orders;
+----------+--------------+-----------+------------+
| order_id | product_name | order_num | order_date |
+----------+--------------+-----------+------------+
|        1 | Laptop       |      5544 | 2020-02-01 |
|        2 | Mouse        |      3322 | 2020-02-11 |
|        3 | Desktop      |      2135 | 2020-01-05 |
|        4 | Mobile       |      3432 | 2020-02-22 |
|        5 | Anti-Virus   |      5648 | 2020-03-10 |
+----------+--------------+-----------+------------+
5 rows in set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select max(income) from employees;
+-------------+
| max(income) |
+-------------+
|     5000000 |
+-------------+
```

```
mysql> insert into employees values(111,'Alexander',45,'California',70000);
Query OK, 1 row affected (0.00 sec)

mysql> insert into orders values(6,'Printer',5654,'2020-01-10');
Query OK, 1 row affected (0.00 sec)

mysql> Commit;
Query OK, 0 rows affected (0.01 sec)

mysql> select*from employees;
+--------+----------+---------+-----------+---------+
| emp_id | emp_name | rmp_age | city      | income  |
+--------+----------+---------+-----------+---------+
|    101 | Peter    |      32 | New York  |  200000 |
|    102 | Mark     |      32 | California |  300000 |
|    103 | Donald   |      43 | Arizona   | 1000000 |
|    104 | Obama    |      35 | Florida   | 5000000 |
|    105 | Linklon  |      32 | Georgia   |  250000 |
|    106 | Kane     |      45 | Alaska    |  450000 |
|    107 | Adam     |      35 | California | 5000000 |
|    108 | Marcoulam |      40 | Florida   |  350000 |
|    111 | Alexander |      45 | California |   70000 |
+--------+----------+---------+-----------+---------+
9 rows in set (0.00 sec)

mysql> select*from orders;
+----------+--------------+-----------+------------+
| order_id | product_name | order_num | order_date |
+----------+--------------+-----------+------------+
|        1 | Laptop       |      5544 | 2020-02-01 |
|        2 | Mouse        |      3322 | 2020-02-11 |
|        3 | Desktop      |      2135 | 2020-01-05 |
|        4 | Mobile       |      3432 | 2020-02-22 |
|        5 | Anti-Virus   |      5648 | 2020-03-10 |
|        6 | Printer      |      5654 | 2020-01-10 |
+----------+--------------+-----------+------------+
6 rows in set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from orders;
Query OK, 6 rows affected (0.00 sec)

mysql> select*from orders;
Empty set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select*from orders;
+----------+--------------+-----------+------------+
| order_id | product_name | order_num | order_date |
+----------+--------------+-----------+------------+
|        1 | Laptop       |      5544 | 2020-02-01 |
|        2 | Mouse        |      3322 | 2020-02-11 |
|        3 | Desktop      |      2135 | 2020-01-05 |
|        4 | Mobile       |      3432 | 2020-02-22 |
|        5 | Anti-Virus   |      5648 | 2020-03-10 |
|        6 | Printer      |      5654 | 2020-01-10 |
+----------+--------------+-----------+------------+
6 rows in set (0.00 sec)
```

Practical No.4: -Perform Save point transaction using SQL query.

```
mysql> create table nadeem(id int,name varchar(33),Age int,address varchar(40),salary int);
Query OK, 0 rows affected (0.04 sec)

mysql> insert into nadeem values (1,'Hamilton',23,'Australia',34000),
    -> (2,'Warner',34,'England',22000),
    -> (3,'Martin',28,'China',25000),
    -> (4,'Twinkle',30,'Turkey',50000),\
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
mysql> insert into nadeem values (1,'Hamilton',23,'Australia',34000),
    -> (2,'Warner',34,'England',22000),
    -> (3,'Martin',28,'China',25000),
    -> (4,'Twinkle',30,'Turkey',50000),
    -> (5,'Tinu',32,'Nepal',45000),
    -> (6,'Michal',31,'Bhutan',20000);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select*from nadeem;
+------+----------+------+-----------+--------+
| id   | name     | Age  | address   | salary |
+------+----------+------+-----------+--------+
|    1 | Hamilton |   23 | Australia |  34000 |
|    2 | Warner   |   34 | England   |  22000 |
|    3 | Martin   |   28 | China     |  25000 |
|    4 | Twinkle  |   30 | Turkey    |  50000 |
|    5 | Tinu     |   32 | Nepal     |  45000 |
|    6 | Michal   |   31 | Bhutan    |  20000 |
+------+----------+------+-----------+--------+
6 rows in set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> savepoint sp1;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from employees where id=1;
ERROR 1054 (42S22): Unknown column 'id' in 'where clause'
mysql> delete from nadeem where id=1;
Query OK, 1 row affected (0.00 sec)

mysql> savepoint sp2;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from nadeem where id=2;
Query OK, 1 row affected (0.00 sec)

mysql> rollback savepoint sp2;
ERROR 1064 (42000): You have an error in your SQL syntax; ch
mysql> rollback to savepoint sp2;
Query OK, 0 rows affected (0.00 sec)

mysql> select*from nadeem;
+------+----------+------+---------+--------+
| id   | name     | Age  | address | salary |
+------+----------+------+---------+--------+
|    2 | Warner   |   34 | England |  22000 |
|    3 | Martin   |   28 | China   |  25000 |
|    4 | Twinkle  |   30 | Turkey  |  50000 |
|    5 | Tinu     |   32 | Nepal   |  45000 |
|    6 | Michal   |   31 | Bhutan  |  20000 |
+------+----------+------+---------+--------+
5 rows in set (0.00 sec)
```

Practical no 5: Write an SQL query for lost update problem.

```
mysql> create database accounts;
Query OK, 1 row affected (0.01 sec)

mysql> use accounts;
Database changed
mysql> create table accounts(id int primary key,balance int);
Query OK, 0 rows affected (0.06 sec)

mysql> insert into accounts values(1,1000);
Query OK, 1 row affected (0.01 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT BALANCE FROM accounts WHERE id =1;
+---------+
| BALANCE |
+---------+
|    1000 |
+---------+
1 row in set (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT BALANCE FROM accounts WHERE id =1;
+---------+
| BALANCE |
+---------+
|    1000 |
+---------+
1 row in set (0.00 sec)
```

```
mysql> UPDATE accounts SET balance =1100 WHERE id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> UPDATE accounts SET balance =1050 WHERE id=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM ACCOUNTS;
+----+---------+
| id | balance |
+----+---------+
|  1 |    1050 |
+----+---------+
1 row in set (0.00 sec)
```

Practical no 6: -Write an SQL query for temporary inconsistency or dirty read problem

```
mysql> create table accounts3(id int primary key,balance int);
Query OK, 0 rows affected (0.04 sec)

mysql> insert into accounts3 values(1,1000);
Query OK, 1 row affected (0.01 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT BALANCE FROM accounts3 WHERE id=1;
+---------+
| BALANCE |
+---------+
|    1000 |
+---------+
1 row in set (0.00 sec)

mysql> UPDATE accounts3 SET BALANCE=BALANCE+50 WHERE id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT BALANCE FROM ACCOUNTS3 WHERE ID =1;
+---------+
| BALANCE |
+---------+
|    1050 |
+---------+
1 row in set (0.00 sec)
```

```
mysql> UPDATE ACCOUNTS3 SET BALANCE =BALANCE+100 WHERE ID=1
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM ACCOUNTS3;
+----+---------+
| id | balance |
+----+---------+
|  1 |    1150 |
+----+---------+
1 row in set (0.00 sec)
```

Practical no 7: -Write an SQL query for unrepeatable read problem

```
mysql> create table accounts4(id int primary key,balance int);
Query OK, 0 rows affected (0.04 sec)

mysql> insert into accounts4 values(1,1000);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM ACCOUNTS4;
+----+---------+
| id | balance |
+----+---------+
|  1 |    1000 |
+----+---------+
1 row in set (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT BALANCE FROM ACCOUNTS4 WHERE ID=1;
+---------+
| BALANCE |
+---------+
|    1000 |
+---------+
1 row in set (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT BALANCE FROM ACCOUNTS4 WHERE ID =1;
+---------+
| BALANCE |
+---------+
|    1000 |
+---------+
1 row in set (0.00 sec)

mysql> UPDATE ACCOUNTS4 SET BALANCE =BALANCE+100 WHERE ID=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM ACCOUNTS4;
+----+---------+
| id | balance |
+----+---------+
|  1 |    1100 |
+----+---------+
1 row in set (0.00 sec)
```

Practical no 8: -Write PL/SQL query for variables and identifier

```
SQL> SET SERVEROUTPUT ON ;
SQL> DECLARE
  2   --variable declaration
  3   message varchar2(20):='Hello,World!';
  4   BEGIN
  5   /*
  6   *PL/SQL executable statement(s)
  7   */
  8   dbms_output.put_line(message);
  9   END;
 10   /
Hello,World!

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> DECLARE
  2     a integer:=10;
  3      b integer:=20;
  4     c integer;
  5      f real;
  6      BEGIN
  7     c:=a+b;
  8      dbms_output.put_line('Value of c:'||c);
  9     f:=70.0/3.0;
 10     dbms_output.put_line('Value of f:'||f);
 11   END;
 12    /
Value of c:30
Value of f:23.333333333333333333333333333333333333

PL/SQL procedure successfully completed.
```

Practical no 9:-Perform SQL query to show the implementation of sub queries

```
mysql> SELECT*FROM EMPLOYEES;
+--------+-----------+---------+------------+---------+
| emp_id | emp_name  | emp_age | city       | income  |
+--------+-----------+---------+------------+---------+
|    101 | Peter     |      32 | Newyork    |  200000 |
|    102 | Mark      |      32 | California |  300000 |
|    103 | Donald    |      40 | Arizona    | 1000000 |
|    104 | Obama     |      35 | Florida    | 5000000 |
|    105 | Linklon   |      32 | Georgia    |  250000 |
|    106 | Kane      |      45 | Alaska     |  450000 |
|    107 | Adam      |      35 | California | 5000000 |
|    108 | Maculam   |      40 | Florida    |  350000 |
|    109 | Brayan    |      32 | Alaska     |  400000 |
|    110 | Stephen   |      40 | Arizona    |  600000 |
|    111 | Alexander |      45 | california |   70000 |
+--------+-----------+---------+------------+---------+
11 rows in set (0.00 sec)

mysql> SELECT*FROM EMPLOYEES WHERE EMP_ID IN(SELECT EMP_ID
    -> FROM EMPLOYEES WHERE INCOME>350000);
+--------+----------+---------+------------+---------+
| emp_id | emp_name | emp_age | city       | income  |
+--------+----------+---------+------------+---------+
|    103 | Donald   |      40 | Arizona    | 1000000 |
|    104 | Obama    |      35 | Florida    | 5000000 |
|    106 | Kane     |      45 | Alaska     |  450000 |
|    107 | Adam     |      35 | California | 5000000 |
|    109 | Brayan   |      32 | Alaska     |  400000 |
|    110 | Stephen  |      40 | Arizona    |  600000 |
+--------+----------+---------+------------+---------+
6 rows in set (0.00 sec)

mysql> SELECT EMP_NAME,CITY,INCOME FROM EMPLOYEES
    -> WHERE INCOME=(SELECT MAX(income)FROM employees);
+----------+------------+---------+
| EMP_NAME | CITY       | INCOME  |
+----------+------------+---------+
| Obama    | Florida    | 5000000 |
| Adam     | California | 5000000 |
+----------+------------+---------+
2 rows in set (0.00 sec)
```

Practical no 10: -Write PL/SQL query to show the implementation of Global and Local Variables

```
Connected.
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  --Global variables
  3  Num3 number:=95;
  4  Num4 number:=85;
  5  BEGIN
  6  dbms_output.put_line('Outer variable num1:'||num3);
  7  dbms_output.put_line('Outer variable num2:'||num4);
  8  DECLARE
  9  Num1 number:=195;
 10  Num2 number:=185;
 11  BEGIN
 12  dbms_output.put_line('Inner variable num1:'||num1);
 13  dbms_output.put_line('Inner variable num2:'||num2);
 14  dbms_output.put_line('Inner variable num2:'||num3);
 15  dbms_output.put_line('Inner variable num2:'||num4);
 16  END;
 17  END;
 18  /
Outer variable num1:95
Outer variable num2:85
Inner variable num1:195
Inner variable num2:185
Inner variable num2:95
Inner variable num2:85

PL/SQL procedure successfully completed.
```

Practical no 11: -Write PL/SQL query to assign SQL query Results to PL/SQL variables

```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE TABLE CUSTOMER(ID INT NOT NULL,
  2  NAME VARCHAR(20) NOT NULL,
  3  AGE INT NOT NULL,ADDRESS CHAR(25),SALARY DECIMAL(18,2),PRIMARY KEY(ID)
  4  );

Table created.

SQL> INSERT INTO CUSTOMER VALUES(1,'Ramesh',32,'Ahmedabad',2000.00);

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(2,'Khilan',25,'Delhi',1500.00);

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(3,'Kaushik',23,'Kota',2000.00);

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(4,'Chaitali',25,'Mumbai',6500.00);

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(5,'Hardik',27,'Bhopal',8500.00);

1 row created.

SQL> INSERT INTO CUSTOMER VALUES(6,'Komal',22,'MP',4500.00);

1 row created.

SQL> DECLARE
  2  c_id customer.id%type :=1;
  3  c_name customer.name%type;
  4  c_addr customer.address%type;
  5  c_sal customer.salary%type;
  6  BEGIN
  7  SELECT name,address,salary INTO c_name,c_addr,c_sal
  8  FROM customer
  9  WHERE id = c_id;
 10  dbms_output.put_line
 11  ('Customer'||c_name||'from'||c_addr||'earns'||c_sal);
 12  END;
 13  /
CustomerRameshfromAhmedabad               earns2000

PL/SQL procedure successfully completed.
```

Practical No 12: - Write PL/SQL program for IF THEN statement and IF THEN ELSE statement

```
SQL> SET SERVEROUTPUT ON;
SQL>  DECLARE
  2     a number(2):=10;
  3    BEGIN
  4      IF(a<20) THEN
  5      dbms_output.put_line('a is less than 20');
  6      END IF;
  7      dbms_output.put_line('value of a is:'||a);
  8      END;
  9     /
a is less than 20
value of a is:10

PL/SQL procedure successfully completed.
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  a number(3):=100;
  3    BEGIN
  4  IF(a<20) THEN
  5  dbms_output.put_line('a is less than 20');
  6  ELSE
  7  dbms_output.put_line('a is not less than 20');
  8  END IF;
  9      dbms_output.put_line('value of a is:'||a);
 10  END;
 11  /
a is not less than 20
value of a is:100

PL/SQL procedure successfully completed.
```

Practical no 13: - Write PL/SQL program for IF THEN ELSE IF statement and NESTED IF THEN ELSE statement.

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2      a number(3):=100;
  3        BEGIN
  4      IF(a=10) THEN
  5          dbms_output.put_line('value of a is 10');
  6      ELSIF(a=20) THEN
  7         dbms_output.put_line('value of a is 20');
  8      ELSIF(a=30) THEN
  9          dbms_output.put_line('value of a is 30');
 10     ELSE
 11         dbms_output.put_line('None of the values is matching');
 12     END IF;
 13         dbms_output.put_line('value of a is:'||a);
 14     END;
 15    /
None of the values is matching
value of a is:100

PL/SQL procedure successfully completed.
```

```
SQL>  SET SERVEROUTPUT ON;
SQL>  DECLARE
  2          a number(3):=100;
  3          b number(3):=200;
  4            BEGIN
  5      -- CHECK THE BOOLEAN CONDITION
  6      IF(a=100) THEN
  7      -- IF CONDITION IS TRUE THEN CHECK THE FOLLOWING
  8      IF(b=200) THEN
  9      --IF CONDITION IS TRUE THEN PRINT THE FOLLOWING
 10            dbms_output.put_line('value of a is 100 and b is 200');
 11     END IF;
 12     END IF;
 13            dbms_output.put_line('value of a is:'||a);
 14            dbms_output.put_line('value of b is:'||b);
 15    END;
 16     /
value of a is 100 and b is 200
value of a is:100
value of b is:200

PL/SQL procedure successfully completed.
```

Practical no 14: - PL/SQL program for CASE statement.

```
SQL>  SET SERVEROUTPUT ON;
SQL>   DECLARE
  2      grade char(1):='A';
  3      BEGIN
  4      CASE grade
  5     when 'A' then dbms_output.put_line('Excellent');
  6      when 'B' then dbms_output.put_line('Very good');
  7      when 'C' then dbms_output.put_line('Well done');
  8      when 'D' then dbms_output.put_line('You Passed');
  9      when 'F' then dbms_output.put_line('Better Try again');
 10   else dbms_output.put_line('No such Grade');
 11     END CASE;
 12     END;
 13     /
Excellent

PL/SQL procedure successfully completed.
```

Practical no 15: - Write PL/SQL query to Implement arithmetic operations.

```
SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
  2  dbms_output.put_line(10+5);
  3  dbms_output.put_line(10-5);
  4  dbms_output.put_line(10*5);
  5  dbms_output.put_line(10/5);
  6  dbms_output.put_line(10**5);
  7  END;
  8  /
15
5
50
2
100000
```

Practical no 16: - Write PL/SQL query to implement Relational Operator

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  a number(2):=21;
  3  b number(2):=10;
  4  BEGIN
  5  if (a=b)then
  6  dbms_output.put_line('Line 1 - a is equal to b');
  7  ELSE
  8  dbms_output.put_line('Line 1 - a is not equal to b');
  9  END IF;
 10  IF (a<b) then
 11  dbms_output.put_line('Line 2 - a is less than b');
 12  ELSE
 13  dbms_output.put_line('Line 2 - a is not less than b');
 14  END IF;
 15  IF (a>b) then
 16  dbms_output.put_line('Line 3 - a is greater  than b');
 17  ELSE
 18  dbms_output.put_line('Line 3 - a is not greater  than b');
 19  END IF;
 20  a:=5;
 21  b:=20;
 22  IF (a<=b) THEN
 23  dbms_output.put_line('Line 4 - a is either equal or less  than b');
 24  END IF;
 25  IF (b>=a) THEN
 26  dbms_output.put_line('Line 5 - b is either equal or greater than a');
 27  END IF;
 28  IF (a<>b) THEN
 29  dbms_output.put_line('Line 6 - a is not equal b');
 30  ELSE
 31  dbms_output.put_line('Line 6 - a is equal b');
 32  END IF;
 33  END;
 34  /
Line 1 - a is not equal to b
Line 2 - a is not less than b
Line 3 - a is greater  than b
Line 4 - a is either equal or less  than b
Line 5 - b is either equal or greater than a
Line 6 - a is not equal b
```

Practical no 17: - Write PL/SQL query to shoe the implementation of LIKE Operator

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  PROCEDURE compare (value varchar2, pattern varchar2)is
  3  BEGIN
  4  IF value LIKE pattern THEN
  5  dbms_output.put_line('TRUE');
  6  ELSE
  7  dbms_output.put_line('FALSE');
  8  END IF;
  9  END;
 10  BEGIN
 11  compare('Zara Ali','Z%A_i');
 12  compare('Nuha Ali','Z%A_i');
 13  END;
 14  /
TRUE
FALSE
```

Practical no 18: - Write PL/SQL query to implement BETWEEN Operators

```
SQL> SET SERVEROUTPUT ON;
SQL> x number(2) :=10;
SP2-0734: unknown command beginning "x number(2..." - rest of line ignored.
SQL> end;
SP2-0042: unknown command "end" - rest of line ignored.
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  x number(2) :=10;
  3  BEGIN
  4  IF (x between 5 and 20) THEN
  5  dbms_output.put_line('TRUE');
  6  ELSE
  7  dbms_output.put_line('FALSE');
  8  END IF;
  9  IF (x between 5 and 10) THEN
 10  dbms_output.put_line('TRUE');
 11  ELSE
 12  dbms_output.put_line('FALSE');
 13  END IF;
 14  IF (x between 11 and 20) THEN
 15  dbms_output.put_line('TRUE');
 16  ELSE
 17  dbms_output.put_line('FALSE');
 18  END IF;
 19  END;
 20  /
TRUE
TRUE
FALSE
```

Practical no 19: - Write PL/SQL query for implementation of IN AND IS NULL Operator

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  letter varchar2(1) :='m';
  3  BEGIN
  4  IF (letter in('a','b','c')) THEN
  5  dbms_output.put_line('TRUE');
  6  ELSE
  7  dbms_output.put_line('FALSE');
  8  END IF;
  9  IF (letter in('m','n','o')) THEN
 10  dbms_output.put_line('TRUE');
 11  ELSE
 12  dbms_output.put_line('FALSE');
 13  END IF;
 14  IF (letter is null) THEN
 15  dbms_output.put_line('TRUE');
 16  ELSE
 17  dbms_output.put_line('FALSE');
 18  END IF;
 19  END ;
 20  /
FALSE
TRUE
FALSE
```

Practical no 20: - Write PL/SQL query to implement Logical Operations

```
SQL>  SET SERVEROUTPUT ON;
SQL> DECLARE
  2     a boolean := true;
  3       b boolean := false;
  4      BEGIN
  5      IF (a AND b) THEN
  6     dbms_output.put_line('Line 1 - Condition is true');
  7  END IF;
  8      IF (a OR b) THEN
  9      dbms_output.put_line('Line 2 - Condition is true');
 10    END IF;
 11     IF (NOT a) THEN
 12   dbms_output.put_line('Line 3 - a is not true');
 13  ELSE
 14     dbms_output.put_line('Line 3 - a is  true');
 15     END IF;
 16     IF (NOT b) THEN
 17     dbms_output.put_line('Line 4 - b is not true');
 18     ELSE
 19     dbms_output.put_line('Line 4 - b is true');
 20     END IF;
 21    END;
 22  /
Line 2 - Condition is true
Line 3 - a is  true
Line 4 - b is not true
```

Practical no 21: - Write PL/SQL for Implementation of Operation Precedence

```
SQL> SET SERVEROUTPUT ON
SQL>  DECLARE
  2      a number(2) :=20;
  3      b number(2) :=10;
  4     c number(2) :=15;
  5      d number(2) :=5;
  6      e number(2) ;
  7      BEGIN
  8      e := (a+b)*c/d;
  9      dbms_output.put_line('Value of (a+b)*c/d is :'||e);
 10     e := ((a+b)*c)/d;
 11     dbms_output.put_line('Value of ((a+b)*c)/d is :'||e);
 12     e := (a+b)*(c/d);
 13     dbms_output.put_line('Value of (a+b)*(c/d )is :'||e);
 14     e := a+(b*c)/d;
 15     dbms_output.put_line('Value of a+(b*c)/d is :'||e);
 16  END;
 17  /
Value of (a+b)*c/d is :90
Value of ((a+b)*c)/d is :90
Value of (a+b)*(c/d )is :90
Value of a+(b*c)/d is :50

PL/SQL procedure successfully completed.
```

Practical no 22: - Write PL/SQL query for Basic loop and While loop

```
SQL> SET SERVEROUTPUT ON
SQL>  DECLARE
  2  x number :=10;
  3  BEGIN
  4  LOOP
  5  dbms_output.put_line(x);
  6  x:=x+10;
  7  IF x>50 THEN
  8  exit;
  9  END IF;
 10  END LOOP;
 11  -- after exit ,control resumes here
 12  dbms_output.put_line('After Exit x is:'||x);
 13  END;
 14  /
10
20
30
40
50
After Exit x is:60

PL/SQL procedure successfully completed.
```

```
SQL> SET SERVEROUTPUT ON
SQL>  DECLARE
  2  a number(2) :=10;
  3  BEGIN
  4  WHILE a<20 LOOP
  5  dbms_output.put_line('Value of a:'||a);
  6  a:=a+1;
  7  END LOOP;
  8  END;
  9  /
Value of a:10
Value of a:11
Value of a:12
Value of a:13
Value of a:14
Value of a:15
Value of a:16
Value of a:17
Value of a:18
Value of a:19

PL/SQL procedure successfully completed.
```

Practical no 23: - Write PL/SQL query for FOR LOOP statement and REVERSE FOR LOOP statement.

```
SQL> SET SERVEROUTPUT ON
SQL>  DECLARE
  2  a number(2);
  3  BEGIN
  4  FOR  a in 10 .. 20 LOOP
  5  dbms_output.put_line('Value of a:'||a);
  6  END LOOP;
  7  END;
  8  /
Value of a:10
Value of a:11
Value of a:12
Value of a:13
Value of a:14
Value of a:15
Value of a:16
Value of a:17
Value of a:18
```

Practical no 24: - Write PL/SQL query to show the implementation of PL/SQL strings

```
SQL>
SQL> SET SERVEROUTPUT ON
SQL>  DECLARE
  2   name varchar2(20);
  3   company varchar2(30);
  4   introduction clob;
  5   choice char(1);
  6    BEGIN
  7    name:='John Smith';
  8    company:='Infotech';
  9    introduction:='Hello!I"mJohn Smith from Infotech';
 10    choice:='y';
 11    IF choice='y' THEN
 12   dbms_output.put_line(name);
 13   dbms_output.put_line(company);
 14   dbms_output.put_line(introduction);
 15    END IF;
 16   END;
 17    /
John Smith
Infotech
Hello!I"mJohn Smith from Infotech

PL/SQL procedure successfully completed.
```

Practical no 25: - Write PL/SQL Query to show the implementation of PL/SQL arrays

```
SQL> SET SERVEROUTPUT ON
SQL>  DECLARE
  2   type namesarray is VARRAY(5) OF VARCHAR2(10);
  3    type grades IS VARRAY(5) OF INTEGER;
  4    names namesarray;
  5    marks grades;
  6    total integer;
  7    BEGIN
  8   names := namesarray('Kavita','Pritam','Ayan','Rishav','Aziz');
  9    marks:= grades(98,97,78,87,92);
 10  total := names.count;
 11    dbms_output.put_line('Total'||total||'Students');
 12   FOR i in 1 .. total LOOP
 13   dbms_output.put_line('Student:'||names(i)||'
 14    Marks:'||marks(i));
 15   END LOOP;
 16   END;
 17    /
Total5Students
Student:Kavita
  Marks:98
Student:Pritam
  Marks:97
Student:Ayan
  Marks:78
Student:Rishav
  Marks:87
Student:Aziz
  Marks:92

PL/SQL procedure successfully completed.
```

PRACTICAL NO 26: - Write PL/SQL query for procedure with IN ,OUT ,IN OUT Parameters

```
SQL> SET SERVEROUTPUT ON ;
SQL> DECLARE
  2  a number;
  3  b number;
  4  c number;
  5  PROCEDURE findMin(x IN number,y IN number,z OUT number)IS
  6  BEGIN
  7  IF x<y THEN
  8  z:=x;
  9  ELSE
 10  z:=y;
 11  END IF;
 12  END;
 13  BEGIN
 14  a:=23;
 15  b:=45;
 16  findMin(a,b,c);
 17  dbms_output.put_line('Minium of (23,45):'||c);
 18  END;
 19  /
Minium of (23,45):23

PL/SQL procedure successfully completed.
```

```
SQL> SET SERVEROUTPUT ON ;
SQL> DECLARE
  2  a number;
  3  PROCEDURE squareNum(x IN OUT number ) IS
  4  BEGIN
  5  x:=x*x;
  6  END;
  7  BEGIN
  8  a:=23;
  9  squareNum(a);
 10  dbms_output.put_line('Square of (23):'||a);
 11  END;
 12  /
Square of (23):529

PL/SQL procedure successfully completed.
```

Practical no 27: - Write PL/SQL query to Implement Function

```
SQL> SET SERVEROUTPUT ON ;
SQL>  DECLARE
  2      a number;
  3      b number;
  4      c number;
  5      FUNCTION findMax(x IN number,y IN number)
  6      RETURN number
  7      IS
  8      z number;
  9      BEGIN
 10     IF x>y THEN
 11     z:=x;
 12     ELSE
 13     z:=y;
 14     END IF;
 15    RETURN z;
 16     END;
 17     BEGIN
 18     a:=23;
 19     b:=45;
 20    c:= findMax(a,b);
 21    dbms_output.put_line('Maximumof (23,45):'||c);
 22    END;
 23     /
Maximumof (23,45):45

PL/SQL procedure successfully completed.
```

PRACTICAL NO 28: - Write PL/SQL query to implement IMPLICIT cursor and EXPLICIT cursor

- A cursor is a pointer that refer, fetch and processed the rows written by the SQL statement.

```
SQL> CREATE TABLE CUSTOMERs1(ID NUMBER PRIMARY KEY,NAME VARCHAR(50),AGE NUMBER,ADDRESS VARCHAR2(100),SA

Table created.

SQL> INSERT INTO CUSTOMERS1 VALUES(1,'RAMESH',32,'AHMEDABAD',2000.00);

1 row created.

SQL> INSERT INTO CUSTOMERS1 VALUES(2,'KHILAN',25,'DELHI',1500.00);

1 row created.

SQL> INSERT INTO CUSTOMERS1 VALUES(3,'KAUSHIK',23,'KOTA',2000.00);

1 row created.

SQL> INSERT INTO CUSTOMERS1 VALUES(4,'CHAITALI',25,'MUMBAI',6500.00);

1 row created.

SQL> INSERT INTO CUSTOMERS1 VALUES(5,'HARDIK',27,'BHOPAL',8500.00);

1 row created.

SQL> INSERT INTO CUSTOMERS1 VALUES(6,'KOMAL',22,'MP',4500.00);

1 row created.
```

```
SQL> SELECT *FROM CUSTOMERS1;

        ID NAME                                                AGE
---------- -------------------------------------------------- ----------
ADDRESS
--------------------------------------------------------------------------------
    SALARY
----------
         1 RAMESH                                              32
AHMEDABAD
      2000

         2 KHILAN                                              25
DELHI
      1500

        ID NAME                                                AGE
---------- -------------------------------------------------- ----------
ADDRESS
--------------------------------------------------------------------------------
    SALARY
----------

         3 KAUSHIK                                             23
KOTA
      2000

         4 CHAITALI                                            25
MUMBAI

        ID NAME                                                AGE
---------- -------------------------------------------------- ----------
ADDRESS
--------------------------------------------------------------------------------
    SALARY
----------
      6500

         5 HARDIK                                              27
BHOPAL
      8500

         6 KOMAL                                               22

        ID NAME                                                AGE
---------- -------------------------------------------------- ----------
ADDRESS
```

```
SQL> SET SERVEROUTPUT ON ;
SQL> DECLARE
  2    total_rows number(2);
  3     BEGIN
  4     UPDATE CUSTOMERS1
  5      SET salary=salary+500;
  6      IF sql%notfound THEN
  7      dbms_output.put_line('no customers selected');
  8      ELSIF sql%found THEN
  9      total_rows := sql%rowcount;
 10     dbms_output.put_line(total_rows||' customers selected');
 11     END IF;
 12    END;
 13    /
6 customers selected

PL/SQL procedure successfully completed.
```

```
SQL> SET SERVEROUTPUT ON ;
SQL> DECLARE
  2  c_id CUSTOMERS1.id%type;
  3  c_name CUSTOMERS1.name%type;
  4  c_addr CUSTOMERS1.address%type;
  5  CURSOR c_CUSTOMERS1 is
  6  SELECT id ,name,address FROM CUSTOMERS1;
  7  BEGIN
  8  OPEN c_CUSTOMERS1;
  9  LOOP
 10  FETCH c_CUSTOMERS1 into c_id ,c_name,c_addr;
 11  EXIT WHEN c_CUSTOMERS1%notfound;
 12  dbms_output.put_line(c_id||' '||c_name||' '||c_addr);
 13  END LOOP;
 14  CLOSE c_CUSTOMERS1;
 15  END;
 16  /
1 RAMESH AHMEDABAD
2 KHILAN DELHI
3 KAUSHIK KOTA
4 CHAITALI MUMBAI
5 HARDIK BHOPAL
6 KOMAL MP
```

Practical no 29: - Write PL/SQL query to show the implementation of PL/SQL Records

```
SQL> create table customer2(ID NUMBER PRIMARY KEY,NAME VARCHAR(50),AGE NUMBER,ADDRESS VARCHAR2(100),SALARY NUMBER(10,2));

Table created.

SQL> INSERT INTO customer2 VALUES(1,'RAMESH',32,'AHMEDABAD',2000.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(2,'KHILAN',25,'DELHI',1500.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(3,'KAUSHIK',23,'KOTA',2000.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(4,'CHAITALI',25,'MUMBAI',6500.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(5,'HARDIK',27,'BHOPAL',8500.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(6,'KOMAL',22,'MP',4500.00);

1 row created.

SQL> SELECT * FROM customer2;
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  customer_rec customer2%rowtype;
  3  BEGIN
  4  SELECT * into customer_rec
  5  FROM customer2
  6  WHERE id=5;
  7  dbms_output.put_line('Customer ID:'||customer_rec.id);
  8  dbms_output.put_line('Customer Name:'||customer_rec.name);
  9  dbms_output.put_line('Customer Address:'||customer_rec.address);
 10  dbms_output.put_line('Customer Salary:'||customer_rec.salary);
 11  END;
 12  /
Customer ID:5
Customer Name:HARDIK
Customer Address:BHOPAL
Customer Salary:8500
```

PRACTICAL NO 30: - Write PL/SQL Query to show the implementation of PL/SQL exception

```
SQL> create table customer2(ID NUMBER PRIMARY KEY,NAME VARCHAR(50),AGE NUMBER,ADDRESS VARCHAR2(100),SALARY NUMBER(10,2));

Table created.

SQL> INSERT INTO customer2 VALUES(1,'RAMESH',32,'AHMEDABAD',2000.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(2,'KHILAN',25,'DELHI',1500.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(3,'KAUSHIK',23,'KOTA',2000.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(4,'CHAITALI',25,'MUMBAI',6500.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(5,'HARDIK',27,'BHOPAL',8500.00);

1 row created.

SQL> INSERT INTO customer2 VALUES(6,'KOMAL',22,'MP',4500.00);

1 row created.

SQL> SELECT * FROM customer2;
```

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  c_id customer2.id%type :=8;
  3  c_name customer2.Name%type;
  4  c_addr customer2.address%type;
  5  BEGIN
  6  SELECT name,address INTO c_name,c_addr
  7  FROM customer2
  8  WHERE id=c_id;
  9  DBMS_OUTPUT.PUT_LINE('NAME:'||c_name);
 10  DBMS_OUTPUT.PUT_LINE('ADDRESS:'||c_addr);
 11  EXCEPTION
 12  WHEN no_data_found THEN
 13  dbms_output.put_line('No such Customer!');
 14  WHEN others THEN
 15  dbms_output.put_line('Error!');
 16  END;
 17  /
No such Customer!

PL/SQL procedure successfully completed.
```

Practical no 31: - Write PL/SQL query to show the implementation of PL/SQL Trigger

```
SQL> CREATE TABLE  EMPLOYEE(
  2      ID      NUMBER PRIMARY KEY,
  3      NAME    VARCHAR2(50),
  4      AGE     NUMBER,
  5      ADDRESS VARCHAR2(100),
  6      SALARY  NUMBER(10,2)
  7  );

Table created.

SQL> INSERT INTO EMPLOYEE (ID, NAME, AGE, ADDRESS, SALARY) VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

1 row created.

SQL> INSERT INTO EMPLOYEE (ID, NAME, AGE, ADDRESS, SALARY) VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);

1 row created.

SQL> INSERT INTO EMPLOYEE(ID, NAME, AGE, ADDRESS, SALARY) VALUES (3, 'Kaushik', 23, 'Kota', 2000.00);

1 row created.

SQL> INSERT INTO EMPLOYEE(ID, NAME, AGE, ADDRESS, SALARY) VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00);

1 row created.

SQL> INSERT INTO EMPLOYEE(ID, NAME, AGE, ADDRESS, SALARY) VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00);

1 row created.

SQL> INSERT INTO EMPLOYEE (ID, NAME, AGE, ADDRESS, SALARY) VALUES (6, 'Komal', 22, 'MP', 4500.00);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM  EMPLOYEE;
```

```
SQL> CREATE OR REPLACE TRIGGER display_salary_changes3
  2  BEFORE DELETE OR INSERT OR UPDATE ON EMPLOYEE
  3  FOR EACH ROW
  4  WHEN(NEW.ID>0)
  5  DECLARE
  6  sal_diff number;
  7  BEGIN
  8  sal_diff := :NEW.salary -:OLD.salary;
  9  dbms_output.put_line('Old salary:'||:OLD.salary);
 10  dbms_output.put_line('New salary:'||:NEW.salary);
 11  dbms_output.put_line('Salary difference:'||sal_diff);
 12  END;
 13  /

Trigger created.
```

```
SQL> INSERT INTO EMPLOYEE VALUES(7,'Kriti',22,'HP',7500);
Old salary:
New salary:7500
Salary difference:

1 row created.

SQL> UPDATE EMPLOYEE
  2  SET salary =salary +500
  3  WHERE id=2
  4  ;
Old salary:1500
New salary:2000
Salary difference:500

1 row updated.
```

Practical no 32: - Perform MONGODB CRUD Operations

```
> show dbs
< admin        40.00 KiB
  coffeeshop   80.00 KiB
  config       72.00 KiB
  local       240.00 KiB
  sakshi       64.00 KiB
  students    144.00 KiB
> use nadeem
< switched to db nadeem
> db
< nadeem
> db.createCollection("shaikh")
< { ok: 1 }
> db.shaikh.insertOne({name:"Angela",age:27,});
< {
    acknowledged: true,
    insertedId: ObjectId('68917512d54dd111e7d22044')
  }
> db.shaikh.insertMany([{name:"Angela",age:27},{name:"Dwight",age:30},{name:"Jim",age:29}]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('689175f3d54dd111e7d22045'),
      '1': ObjectId('689175f3d54dd111e7d22046'),
      '2': ObjectId('689175f3d54dd111e7d22047')
    }
  }
```

```
> db.shaikh.find()
< {
    _id: ObjectId('68917512d54dd111e7d22044'),
    name: 'Angela',
    age: 27
  }
  {
    _id: ObjectId('689175f3d54dd111e7d22045'),
    name: 'Angela',
    age: 27
  }
  {
    _id: ObjectId('689175f3d54dd111e7d22046'),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId('689175f3d54dd111e7d22047'),
    name: 'Jim',
    age: 29
  }
> db.shaikh.find({age:{$gt:29}},{name:1,age:1})
< {
    _id: ObjectId('689175f3d54dd111e7d22046'),
    name: 'Dwight',
    age: 30
  }
> db.shaikh.findOne({name:"Jim"})
< {
    _id: ObjectId('689175f3d54dd111e7d22047'),
    name: 'Jim',
    age: 29
  }
```

```
db.shaikh.find({age:{$gt:29}},{name:1,age:1})
{
  _id: ObjectId('689175f3d54dd111e7d22046'),
  name: 'Dwight',
  age: 30
}
db.shaikh.findOne({name:"Jim"})
{
  _id: ObjectId('689175f3d54dd111e7d22047'),
  name: 'Jim',
  age: 29
}
db.shaikh.updateOne({name:"Angela"},{$set:{email:"angela@gmail.com"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
db.shaikh.updateMany({age:{$lt:30}},{$set:{name:"Angela"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
> db.shaikh.deleteOne({age:30})
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.shaikh.deleteMany({age:{$lt:30}})
< {
    acknowledged: true,
    deletedCount: 3
  }
```

```
> db.shaikh.drop()
< true
> show collections
<
> show dbs
< admin        40.00 KiB
  coffeeshop   80.00 KiB
  config      108.00 KiB
  local       240.00 KiB
  sakshi       64.00 KiB
  students    144.00 KiB
> db.dropDatabase()
< { ok: 1, dropped: 'nadeem' }
```
