Experiment No: 9	Demonstrate SQL commands to implement Set operators and LIKE operator with suitable examples.

Aim: Implement Set operators and LIKE operator with suitable examples.

Procedure:

A query can be defined as questioning about the data in the database.

The basic syntax of SQL query can be defined in the following manner.

SELECT <select_list or column_list>
FROM <table_list>
WHERE <qualification>

Q1: Compute increments for the ratings of persons who have sailed two different boats on the same day.

```
SQL> select s.sname,s.rating+1 As rating from sailors s, reserves r1,
        reserves r2  where s.sid=r1.sid AND s.sid=r2.sid AND r1.day=r2.day
        AND r1.bid<>r2.bid;
```

Output:

```
SNAME RATING
-------------------- ---------
dustin 4
dustin 4
```

Q2: Find the names of sailors who have reserved a red or a green boat.

```
SQL> select s.sname from sailors s, reserves r,boats b where s.sid=r.sid
        AND  r.bid=b.bid AND (b.color='red' OR b.color='green')
```

Output:

```
SNAME
--------------------
dustin
dustin
dustin
lubber
lubber
lubber
horatio
horatio

8 rows selected.
```

Q3: find the all sids of sailors who have rating 10 or have reserved boat 104.

```
SQL> select s.sid from sailors s where s.rating=10 union select r.sid from
        reserves  r where r.bid=104;
```

Output:

```
      SID
      ------
  22
  31
  74
```

Q4: Find the ages of sailors whose name begins and ends with B and has at least three characters.

SQL> SELECT S.age FROM Sailors S WHERE S.sname LIKE `B %B';

Output:

```
      AGE
      ------
      63.5
```

Q5: Find the sids of all sailors who have reserved red boats but not green boats.

SQL> SELECT S.sid FROM Sailors S, Reserves R, Boats B WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = `red'
EXCEPT
SELECT S2.sid FROM Sailors S2, Reserves R2, Boats B2 WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = `green';

Output:

```
      SID
      ------
      22
      31
      45
```

Q6: Find all sid's of sailors who have a rating of 10 or have reserved boat 104.

SQL> SELECT S.sid FROM Sailors S WHERE S.rating = 10 UNION
SELECT R.sid FROM Reserves R WHERE R.bid = 104;

Output:

```
      SID
      ------
      22
      31
      45
```

Result: Hence, set operators and LIKE operators are demonstrated with suitable examples successfully.

Experiment No: 10	Demonstrate SQL commands to implement aggregate operators and views with suitable examples.

Aim: Implement aggregate operators and views with suitable examples.

Procedure:

Aggregate operators: In addition to simply retrieving data, we often want to perform some computation or summarization. SQL allows the use of arithmetic expressions. We now consider a powerful class of constructs for computing aggregate values such as MIN and SUM.

1. Count: COUNT following by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of the column.
Syntax: `COUNT (Column name)`
Example: `SELECT COUNT (Sal) FROM emp;`

2. SUM: SUM followed by a column name returns the sum of all the values in that column. ***Syntax:*** `SUM (Column name)`
Example: `SELECT SUM (Sal) From emp;`

3. AVG: AVG followed by a column name returns the average value of that column values. ***Syntax:*** `AVG (n1,n2..)`
Example: `Select AVG(10, 15, 30) FROM DUAL;`

4. MAX: MAX followed by a column name returns the maximum value of that column. ***Syntax:*** `MAX (Column name)`
Example: `SELECT MAX (Sal) FROM emp;`

```
SQL> select deptno,max(sal) from emp group by deptno;

 DEPTNO MAX(SAL)
 ------ --------
 10 5000
 20 3000
 30 2850

SQL> select deptno,max(sal) from emp group by deptno having
     max(sal)<3000;

DEPTNO MAX(SAL)
----- --------
30 2850
```

5. MIN: MIN followed by column name returns the minimum value of that column. ***Syntax:*** `MIN (Column name)`
Example : `SELECT MIN (Sal) FROM emp;`

```
SQL> select deptno,min(sal) from emp group by deptno having
     min(sal)>1000;
```

```
 DEPTNO MIN(SAL)


----- --------
 10 1300
```

VIEW: In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it doest not store in the database. View is a query stored as an object.

Syntax: `CREATE VIEW view_name AS SELECT set of fields FROM relation_name WHERE (Condition)`

1. Example:

```
      SQL>CREATE VIEW employee AS SELECT
           empno,ename,job FROM EMP WHERE job =
           'clerk';
      View created.
 SQL> SELECT * FROM EMPLOYEE;
      EMPNO ENAME JOB
      ---- ------ -------
      7369 SMITH CLERK
      7876 ADAMS CLERK
      7900 JAMES CLERK
      7934 MILLER CLERK
```

2.Example:

```
      CREATE VIEW [Current Product List] AS
      SELECT ProductID,ProductName
      FROM Products
      WHERE Discontinued=No
```

DROP VIEW: This query is used to delete a view , which has been already created. ***Syntax:*** `DROP VIEW View_name;`

Example : `SQL> DROP VIEW EMPLOYEE;`

```
        View dropped
```

Result: Hence, Aggregate operators and Views are executed with suitable examples successfully.

Experiment No: 11	Demonstrate SQL commands to implement Nested queries with suitable examples.

Aim: Implementation Nested queries with suitable examples.

Procedure:
Nested Query:

A query which consists of another query is called nested query. The syntax of nested query is defined in the following manner. **Query1 (Query2)**

Here,
Query1 denotes outer query and **Query2** denotes inner query. Nested query is implemented with IN and NOT IN operators.

Q1: Find the names of sailors who have not reserved a red boat.

```
SQL> SELECT S.sname FROM Sailors S WHERE S.sid NOT IN ( SELECT
        R.sid  FROM Reserves R WHERE R.bid IN ( SELECT B.bid FROM Boats
        B WHERE B.color  = `red' );
```

Output:

```
SNAME
--------
 dustin
lubber
horatio
horatio
```

Q2: Find the names of sailors who have reserved a red boat.

```
SQL> SELECT S.sname FROM Sailors S WHERE S.sid IN ( SELECT R.sid
        FROM  Reserves R WHERE R.bid IN ( SELECT B.bid FROM Boats B
        WHERE B.color =  `red' );
```

Output:

```
SNAME
-----
brutus
andy
rusty
zorba
art
bob
```

Q3: Find the sid's of sailors who have reserved boat number 103.

```
SQL> SELECT S.sid FROM Sailors S WHERE S.sid IN ( SELECT R.sid
     FROM  Reserves R WHERE R.bid=103);
```

Output:

```
SID
------
22
31
```

Q4: Find the names of sailors who have not reserved boat number 103.

```
SQL> SELECT S.sid FROM Sailors S WHERE S.sid NOT IN ( SELECT R.sid
     FROM  Reserves R WHERE R.bid = 103);
```

Output:

```
SID
------
29
32
58
64
71
74
85
95
```

Q5: Find the names of sailors who have not reserved a red boat.

```
SQL> select s.sname from sailors s where s.sid not in (select r.sid from
     reserves r  where r.bid in (select b.bid from boats b where
     b.color='red'));
```

Output:

```
SNAME
--------
brutus
andy
rusty
zorba
art
bob
6 rows selected.
```

Result: **Hence, Nested queries are executed with suitable examples successfully.**

Experiment No: 12	Demonstrate SQL commands to implement Correlated Nested queries with suitable examples.

Aim: Implementation Correlated Nested queries with suitable examples.

Procedure:

Correlated Nested Query:

A query which consists of another query is called nested query. The inner subquery could depend on the row that is currently being examined in the outer query. The syntax of correlated nested query is defined in the following manner.

Query1 (Query2)

Here,

Query1 denotes outer query and **Query2** denotes inner query. Correlated Nested query is implemented with EXISTS and NOT EXISTS operators.

Q1: Find the names of sailors who have not reserved a red boat.

```
SQL> SELECT S.sname FROM Sailors S WHERE NOT EXISTS (SELECT *
       FROM  Reserves R WHERE EXISTS (SELECT * FROM Boats B WHERE
       S.sid = R.sid AND  R.bid = B.bid AND B.color = `red' );
```

Output:

```
        SNAME
       --------
        dustin
  lubber
  horatio
  horatio
```

Q2: Find the names of sailors who have reserved a red boat.

```
SQL> SELECT S.sname FROM Sailors S WHERE EXISTS (SELECT * FROM
       Reserves  R WHERE EXISTS (SELECT * FROM Boats B WHERE S.sid
       = R.sid AND R.bid =  B.bid AND B.color = `red' );
```

Output:

```
SNAME
-----
brutus
andy
rusty
zorba
art
bob
```

Q3: Find the sid's of sailors who have reserved boat number 103.

```
SQL> SELECT S.sid FROM Sailors S WHERE EXISTS (SELECT R.sid FROM
        Reserves R WHERE S.sid = R.sid AND R.bid=103);
```

Output:

```
SID
------
22
31
```

Q4: Find the names of sailors who have not reserved boat number 103.

```
SQL> SELECT S.sid FROM Sailors S WHERE NOT EXISTS (SELECT R.sid
        FROM  Reserves R WHERE S.sid = R.sid AND R.bid=103);
```

Output:

```
SID
------
29
32
58
64
71
74
85
95
```

Q5: Find sid's of sailors who have not reserved a red boat.

```
SQL> SELECT S.sid FROM Sailors S WHERE NOT EXISTS (SELECT * FROM
        Reserves R WHERE EXISTS (SELECT * FROM Boats B WHERE S.sid =
        R.sid AND  R.bid = B.bid AND B.color = `red' );
```

Output:

```
    SNAME
    --------
    29
    32
    58
    71
    85
    95
4 rows selected.
```

Result: **Hence, Correlated Nested queries are executed with suitable examples successfully.**

Experiment No: 13	Demonstrate SQL commands to implement set comparison operators with suitable examples.

Aim: Implementation of set comparison operators with suitable examples.

Procedure:

(Q1) Find sailors whose rating is better than some sailor called Horatio.

```
SQL> SELECT S.sid FROM Sailors S WHERE S.rating > ANY (SELECT
        S2.rating  FROM Sailors S2 WHERE S2.sname = `Horatio');
```

Output:

```
SID
------
31
32
58
71
```

(Q2) Find sailors whose rating is better than every sailor called Horatio.

```
SQL> SELECT S.sid FROM Sailors S WHERE S.rating > ALL (SELECT
        S2.rating  FROM Sailors S2 WHERE S2.sname = `Horatio');
```

Output:

```
SID
------
58
71
```

(Q3) Find the sailors with the highest rating.

```
SQL> SELECT S.sid FROM Sailors S WHERE S.rating >= ALL (SELECT
        S2.rating  FROM Sailors S2);
```

Output:

```
SID
------
58
71
```

Result: **Hence, set comparison operators are executed with suitable examples successfully.**

Experiment No: 14	Demonstrate SQL commands to implement Group By and Having clause with suitable examples.

Aim: Implementation of Group By and Having clause with suitable examples.

Procedure:

The basic syntax of SQL query involves 3 clauses are SELECT, FROM and WHERE. But, extended SQL query involves another set of clauses in the following manner:

SELECT <select_list or column_list>
FROM <table_list>
WHERE <qualification>
GROUP BY <group_list>
HAVING <group_qualification>
ORDER BY <order_list> [asc/desc]

(Q1) Find the age of the youngest sailor for each rating level.

SQL> SELECT S.rating, MIN (S.age) FROM Sailors S GROUP BY S.rating;

Output:

```
        RATING MIN(S.AGE)
        ------ ------
        1 33
        8 25.5
        7 35
        3 25.5
        10 16
        9 35
   6 rows selected
```

(Q2) Find the age of the youngest sailor who is eligible to vote (i.e., is at least 18 years old) for each rating level with at least two such sailors.

SQL> SELECT S.rating, MIN (S.age) AS minage FROM Sailors S WHERE S.age >= 18 GROUP BY S.rating HAVING COUNT (*) > 1;

Output:

```
        RATING MIN(S.AGE)
        ------ ------
        8 25.5
        3 25.5
        10 16
   3 rows selected
```

(Q3) *For each red boat, find the number of reservations for this boat.*

```
SQL> SELECT B.bid, COUNT (*) AS sailorcount FROM Boats B, Reserves R
        WHERE  R.bid = B.bid GROUP BY B.bid HAVING B.color = `red';
```

Output:

```
        BID Count(*)
        ------ ------
        102 3
        104 3
  2 rows selected
```

(Q4) *Find the average age of sailors for each rating level that has at least two sailors.*

```
SQL> SELECT S.rating, AVG (S.age) AS avgage FROM Sailors S GROUP BY
        S.rating  HAVING COUNT (*) > 1;
```

Output:

```
        RATING AVGAGE
        ------ ------
        8 34.5
        3 35.5
        10 28
  3 rows selected
```

(Q5) *Find the average age of sailors who are of voting age (i.e., at least 18 years old) for each rating level that has at least two sailors.*

```
SQL> SELECT S.rating, AVG (S.age ) AS avgage FROM Sailors S WHERE S.
        age >=  18 GROUP BY S.rating HAVING 1 < ( SELECT COUNT (*) FROM
        Sailors S2 WHERE  S.rating = S2.rating );
```

Output:

```
        RATING AVGAGE
        ------ ------
        8 34.5
        3 35.5
        10 28
  3 rows selected
```

(Q6) *Find those ratings for which the average age of sailors is the minimum overall ratings.*

```
SQL> SELECT S.rating FROM Sailors S WHERE AVG (S.age) = (SELECT
        MIN (AVG  (S2.age)) FROM Sailors S2 GROUP BY S2.rating );
```

Output:

```
      RATING
        ------
        3
        8
  2 rows selected
```

(Q7) List out ages of oldest sailors for each rating level in ascending order.

```
SQL> SELECT max (S.age), s.rating from sailors s group by s.rating order
        by  s.rating;
```

Output:

```
max(s.age) rating
------------- --------
33 1
63.5 3
45 7
55.5 8
35 9
35 9
6 rows selected.
```

(Q8) Find ages of youngest sailor for each rating level with two such sailors in descending order.

```
SQL> SELECT min (S.age), s.rating from sailors s group by s.rating having
        count(*)>1 order by s.rating desc;
```

Output:

```
Min(s.age) rating
------------- --------
 16 10
25.5 8
35 7
25.5 3

4 rows selected.
```

Result: Hence, Group by and Having clauses are executed with suitable examples successfully.