Experiment No.: 05

Title: To implement aggregate functions with order by, group by, like and having clause.

Batch:A2 **Roll No.:** 16010421063 **Experiment No: 05**

Aim: To implement aggregate functions with order by, group by, like and having clause.

Resources needed: PostgreSQL PgAdmin4

Theory:

The ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns.

```
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

The GROUP BY clause is used in collaboration with the SELECT statement to group together those rows in a table that have identical data. This is done to eliminate redundancy in the output and/or compute aggregates that apply to these groups.

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

```
SELECT column-list
FROM table_name
WHERE [ conditions ]
GROUP BY column1, column2....columnN
ORDER BY column1, column2....columnN
```

The LIKE operator is used to match text values against a pattern using wildcards. If the search expression can be matched to the pattern expression, the LIKE operator will return true, which is 1. There are two wildcards used in conjunction with the LIKE operator:

- The percent sign (%)
- The underscore (_)

The percent sign represents zero, one, or multiple numbers or characters. The underscore represents a single number or character. These symbols can be used in combinations.

If either of these two signs is not used in conjunction with the LIKE clause, then the LIKE acts like the equals operator.

```
SELECT FROM table_name
WHERE column LIKE 'XXXX%'
```

```
or
SELECT FROM table_name
WHERE column LIKE '%XXXX%'
or
SELECT FROM table_name
WHERE column LIKE 'XXXX_'
or
SELECT FROM table_name
WHERE column LIKE '_XXXX'
or
SELECT FROM table_name
WHERE column LIKE '_XXXX_'
```

Here are examples showing WHERE part having different LIKE clause with '%' and '_' operators:

Statement	Description
WHERE SALARY::text LIKE '200%'	Finds any values that start with 200
WHERE SALARY::text LIKE '%200%'	Finds any values that have 200 in any position
WHERE SALARY::text LIKE '_00%'	Finds any values that have 00 in the second and third positions
WHERE SALARY::text LIKE '2_%_%'	Finds any values that start with 2 and are at least 3 characters in length
WHERE SALARY::text LIKE '%2'	Finds any values that end with 2
WHERE SALARY::text LIKE '_2%3'	Finds any values that have a 2 in the second position and end with a 3

WHERE SALARY::text LIKE '2___3'	Finds any values in a five-digit number that start with 2 and end with 3

The HAVING clause allows us to pick out particular rows where the function's result meets some condition.
The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

```
SELECT column1, column2
FROM table1, table2
WHERE [ conditions ]
GROUP BY column1, column2
HAVING [ conditions ]
ORDER BY column1, column2
```

Results: (Queries printout with output)

1. Write 13 queries using ‘order by’, ‘group by’, ‘like’ and ‘having’ clause.
   5 with normal aggregate fun,3 with clauses and aggregate function and 5 with like operator

Example:

```
1. SELECT * FROM COMPANY ORDER BY NAME, SALARY ASC;
2. SELECT NAME, SUM(SALARY) FROM COMPANY GROUP BY NAME;
3. SELECT * FROM COMPANY WHERE AGE::text LIKE '2%';
4. SELECT * FROM COMPANY WHERE ADDRESS  LIKE '%-%';
5. SELECT NAME FROM COMPANY GROUP BY name HAVING count(name) > 1;
```

AGGREGATE

16010421063_Arya on postgres@PostgreSQL 9.6

```
1
2  SELECT  COUNT(no_of_flats) AS total_flats FROM BUILDING
3
```

Data Output | Explain | Messages | History

total_flats bigint
6

16010421063_Arya on postgres@PostgreSQL 9.6

```
1
2  SELECT  AVG(no_of_flats) AS average_flats FROM BUILDING
3
```

Data Output | Explain | Messages | History

average_flats numeric
23.6666666666666667

GROUP BY

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT city,MIN(no_of_flats) AS min_flats_in_city FROM building group by city
2
```

Data Output | Explain | Messages | History

city character varying	min_flats_in_city integer
delhi	8
mumbai	5

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT city,MAX(no_of_flats) AS max_flats_in_city FROM building group by city
2
```

Data Output | Explain | Messages | History

city character varying	max_flats_in_city integer
delhi	50
mumbai	20

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT city,SUm(no_of_flats) AS sum_of_flats_city FROM building group by city
2
```

Data Output | Explain | Messages | History

city character varying	sum_of_flats_city bigint
delhi	108
mumbai	34

HAVING

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT city,SUm(no_of_flats) AS sum_of_flats_city FROM building group by city HAVING SUM(no_of_flats)>40
2
```

Data Output | Explain | Messages | History

city character varying	sum_of_fl... bigint
delhi	108

Order By

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT city,MAX(no_of_flats) AS max_flats FROM building group by city ORDER BY MAX(no_of_flats)
2
```

Data Output | Explain | Messages | History

city character varying	max_flats integer
mumbai	20
delhi	50

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT city,MAX(no_of_flats) AS max_flats
2  FROM building
3  group by city
4  HAVING MAX(no_of_flats)>30
5  ORDER BY MAX(no_of_flats)
6
```

Data Output | Explain | Messages | History

city character varying	max_flats integer
delhi	50

LIKE

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT * FROM building where bname LIKE 'ar%'
2
```

Data Output | Explain | Messages | History

bname text	no_of_flats integer	city character varying
arya	5	mumbai
ary1	9	mumbai
ary2	20	mumbai
ary5	50	delhi
ary7	50	delhi

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT * FROM building where bname LIKE '_r%'
2
```

Data Output | Explain | Messages | History

bname text	no_of_flats integer	city character varying
arya	5	mumbai
ary1	9	mumbai
ary2	20	mumbai
ary5	50	delhi
ary7	50	delhi

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT * FROM building where city LIKE '%i'
2
```

Data Output | Explain | Messages | History

bname text	no_of_flats integer	city character varying
arya	5	mumbai
ary1	9	mumbai
ary2	20	mumbai
ary5	50	delhi
ary7	50	delhi
Elegance	8	delhi

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT * FROM building where city LIKE '%e%'
2
```

Data Output | Explain | Messages | History

bname text	no_of_flats integer	city character varying
ary5	50	delhi
ary7	50	delhi
Elegance	8	delhi

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT * FROM building where city LIKE '_umb__'
2
```

Data Output | Explain | Messages | History

bname text	no_of_flats integer	city character varying
arya	5	mumbai
ary1	9	mumbai
ary2	20	mumbai

Outcomes:

CO3: Illustrate the concept of security, Query processing, indexing and Normalization for Relational database.

Questions:

Q1 Can you apply like operator on integer values? explain with an example how?
Yes we can do it by casting the integer as string. CAST can be used for the same

16010421063_Arya on postgres@PostgreSQL 9.6

```
1  SELECT * FROM building where CAST(no_of_flats as Varchar) LIKE '%5%'
2
```

Data Output | Explain | Messages | History

bname text	no_of_flats integer	city character varying
arya	5	mumbai
ary5	50	delhi
ary7	50	delhi

Q2 Why aggregate functions are more used with order by, group by and having clauses? Can we change order of these clauses when used in single query.
Aggregate functions are used to summarize the data to get a value to fit in a single column. For example if we group by Department then we cannot show salaries of all the employees instead we summarize the data by getting the average or highest salary.
We cannot change the order of these clauses as we can order the data only after we group it and not vice versa. Same in the case of **having** we cannot change the order of the clause as it used to add a condition on the entire group so first the group has to be made

---

Conclusion:
We implemented group by, having and order by clauses to get meaningful data from the database. Aggregate function were used to get statistical values such as Average, sum, minima, maxima and count

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books:

1. Elmasri and Navathe, “Fundamentals of Database Systems”, 6th Edition, Pearson Education
2. Korth, Slberchatz,Sudarshan, :”Database System Concepts”, 6th Edition, McGraw – Hill.

WebSite:

1. http://www.tutorialspoint.com/postgresql/