KJSCE/IT/SYBTECH/SEMIII/DMS/2022-23



**Experiment No.: 05**

**Title:**

To implement aggregate functions with order

by, group by, like and having clauses.

(Autonomous College Affiliated to University of Mumbai)

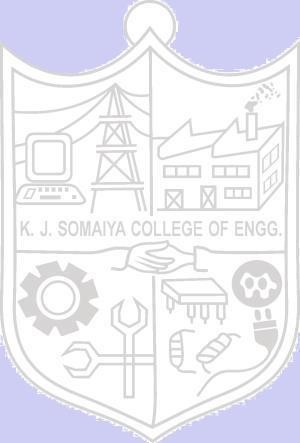
**Batch: A2** **Roll No.: 16010421062** **Experiment No: 05**

**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 | Finds any values in a five-digit number that start with 2 and |
| '2\_\_\_3' | 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

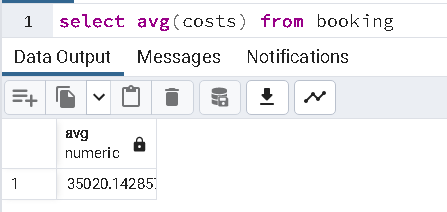
FROM table1, table2

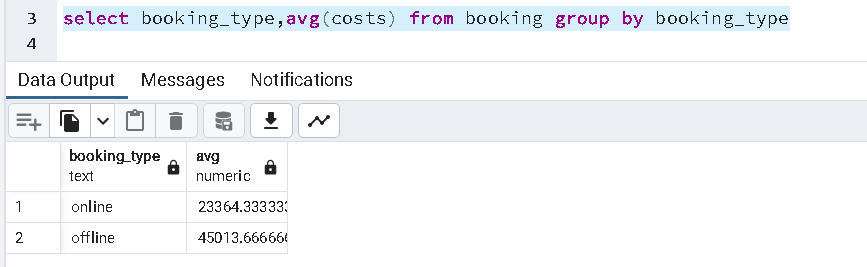
WHERE [ conditions ]

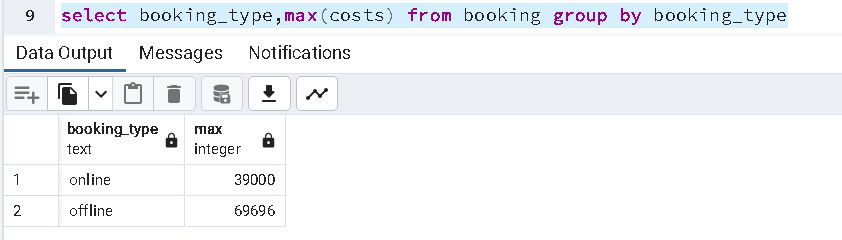
GROUP BY column1, column2 HAVING [ conditions ]

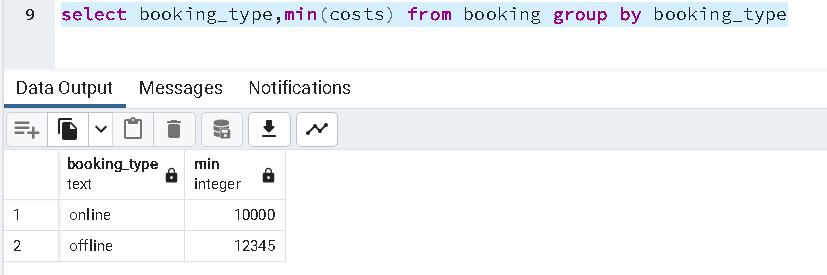
ORDER BY column1, column2

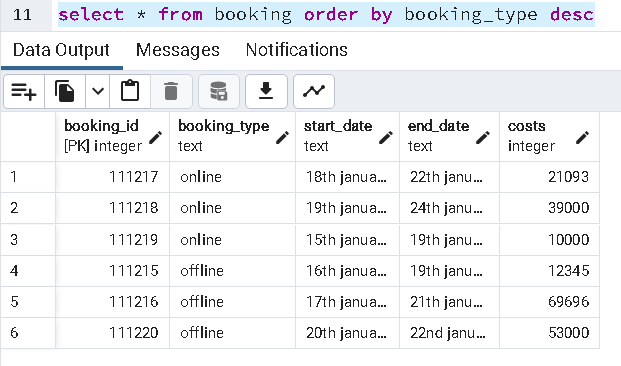
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

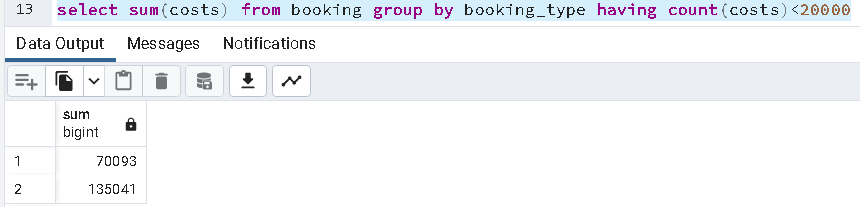


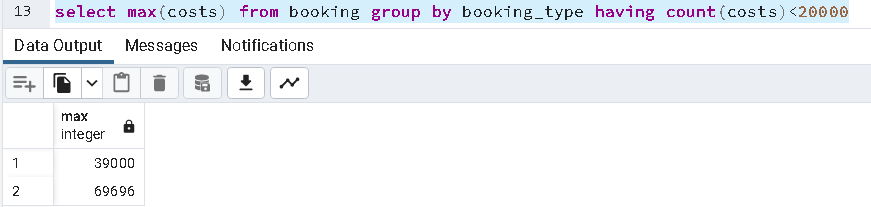


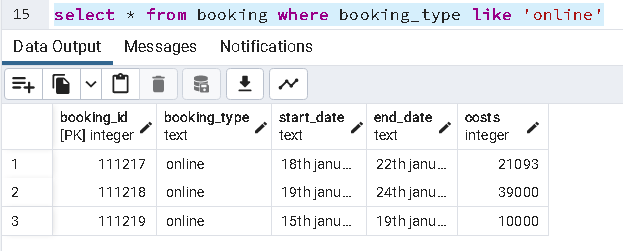


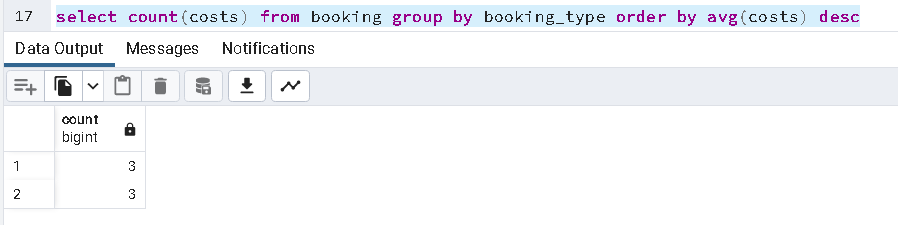












**Example:**

|  |  |
| --- | --- |
| 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; |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **Questions:**

**Q1 Can you apply an operator on integer value? explain with an example how?**

: Yes, we can apply LIKE operator on a integer value, but the search will have to be performed like text match, we can use wild card operator like % and \_ to find the pattern in the integer. E.g.: SELECT \* FROM Customers WHERE CustomerName LIKE 'a%'

**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**

Using the aggregate functions are more used with order by, group by and having clauses as the data which we have it might not have every detail we need to make sense from it, so we use the functions like sum, average, min, max, etc with group by, order by to perform some operations easily. Group by clause groups the databases on the basis of one or more column and aggregates the results. After Grouping the data, you can filter the grouped record using HAVING Clause. HAVING Clause returns the grouped records which match the given conditions.

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Conclusion:**

In completing this experiment, I successfully implemented aggregate functions with order by, group by, like and having clause.

**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/