**SQL Clauses**

## **1. GROUP BY**

* SQL GROUP BY statement is used to arrange identical data into groups. The GROUP BY statement is used with the SQL SELECT statement.
* The GROUP BY statement follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.
* The GROUP BY statement is used with aggregation function.

**Syntax**

SELECT column

FROM table\_name

WHERE conditions

GROUP BY column

ORDER BY column

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PRODUCT** | **COMPANY** | **QTY** | **RATE** | **COST** |
| Item1 | Com1 | 2 | 10 | 20 |
| Item2 | Com2 | 3 | 25 | 75 |
| Item3 | Com1 | 2 | 30 | 60 |
| Item4 | Com3 | 5 | 10 | 50 |
| Item5 | Com2 | 2 | 20 | 40 |
| Item6 | Cpm1 | 3 | 25 | 75 |
| Item7 | Com1 | 5 | 30 | 150 |
| Item8 | Com1 | 3 | 10 | 30 |
| Item9 | Com2 | 2 | 25 | 50 |
| Item10 | Com3 | 4 | 30 | 120 |

**Sample table:**

**Example:**

1. SELECT COMPANY, COUNT(\*)
2. FROM PRODUCT\_MAST
3. GROUP BY COMPANY;

**Output:**

Com1 5

Com2 3

Com3 2

## **2. HAVING**

* HAVING clause is used to specify a search condition for a group or an aggregate.
* Having is used in a GROUP BY clause. If you are not using GROUP BY clause then you can use HAVING function like a WHERE clause.

**Syntax:**

SELECT column1, column2

FROM table\_name

WHERE conditions

GROUP BY column1, column2

HAVING conditions

ORDER BY column1, column2;

**Example:**

SELECT COMPANY, COUNT(\*)

FROM PRODUCT\_MAST

GROUP BY COMPANY

HAVING COUNT(\*)>2;

**Output:**

Com1 5

Com2 3

## **3. ORDER BY**

* The ORDER BY clause sorts the result-set in ascending or descending order.
* It sorts the records in ascending order by default. DESC keyword is used to sort the records in descending order.

**Syntax:**

SELECT column1, column2

FROM table\_name

WHERE condition

ORDER BY column1, column2... ASC|DESC;

**Where**

**ASC:** It is used to sort the result set in ascending order by expression.

**DESC:** It sorts the result set in descending order by expression.

### **Example: Sorting Results in Ascending Order**

**Table:**

**CUSTOMER**

|  |  |  |
| --- | --- | --- |
| **CUSTOMER\_ID** | **NAME** | **ADDRESS** |
| 12 | Kathrin | US |
| 23 | David | Bangkok |
| 34 | Alina | Dubai |
| 45 | John | UK |
| 56 | Harry | US |

Enter the following SQL statement:

SELECT \*

FROM CUSTOMER

ORDER BY NAME;

**Output:**

|  |  |  |
| --- | --- | --- |
| **CUSTOMER\_ID** | **NAME** | **ADDRESS** |
| 34 | Alina | Dubai |
| 23 | David | Bangkok |
| 56 | Harry | US |
| 45 | John | UK |
| 12 | Kathrin | US |

### **Example: Sorting Results in Descending Order**

Using the above CUSTOMER table

SELECT \*

FROM CUSTOMER

ORDER BY NAME DESC;

**Output:**

|  |  |  |
| --- | --- | --- |
| **CUSTOMER\_ID** | **NAME** | **ADDRESS** |
| 12 | Kathrin | US |
| 45 | John | UK |
| 56 | Harry | US |
| 23 | David | Bangkok |
| 34 | Alina | Dubai |

### **4.** **Or Clause in SQL**

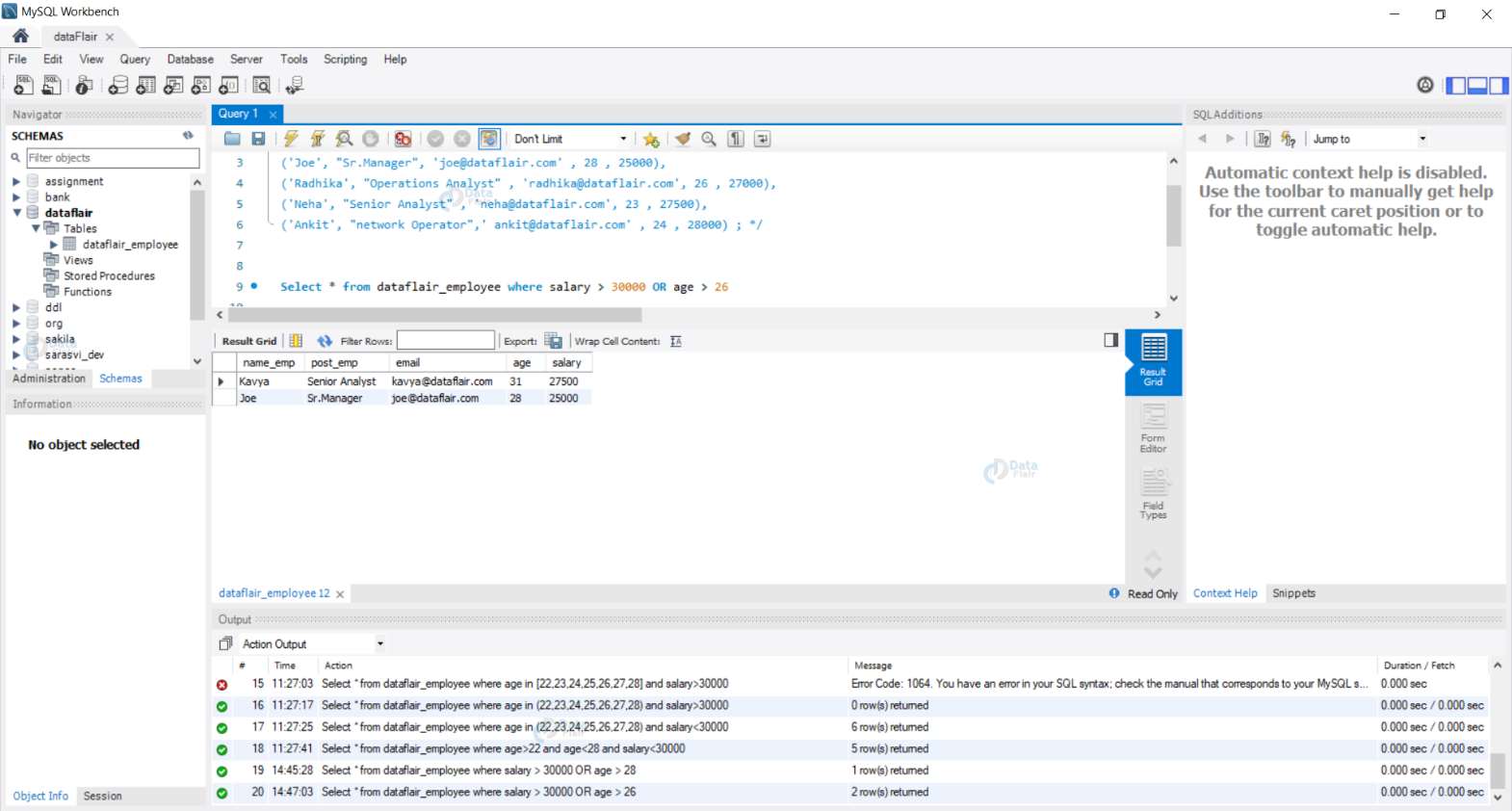
Or clause is beneficial when we need to pass multiple conditions, and we need data that satisfies any one of those specified conditions.

**Rules and Usage:**  
1. Rules:

* When using an Or clause, we need to mention at least two conditions(the result would be at least one of the specified conditions.)

2. Usage:

* Used while transacting and querying a database.
* Used in the update and delete statements.
* Or clause returns a data point when any one of the specified conditions is met.
* **Syntax:**
* SELECT \* FROM tableName WHERE condition1 OR condition2 ;
* **Example 1:** Let us find the employees with age more than 26 or a salary more than 30000
* Select \* from dataflair\_employee where salary > 30000 OR age > 26



### **5. Like Clause in SQL**

LIKE clause is beneficial to find specific patterns in the data. We use specific symbols i.e (%) and (\_). The rules and usage is specified below.

**Rules and Usage:**  
1. Rules:

* % – Represents zero, one, or multiple characters.
* \_ – Represents one single character.

2. Usage:

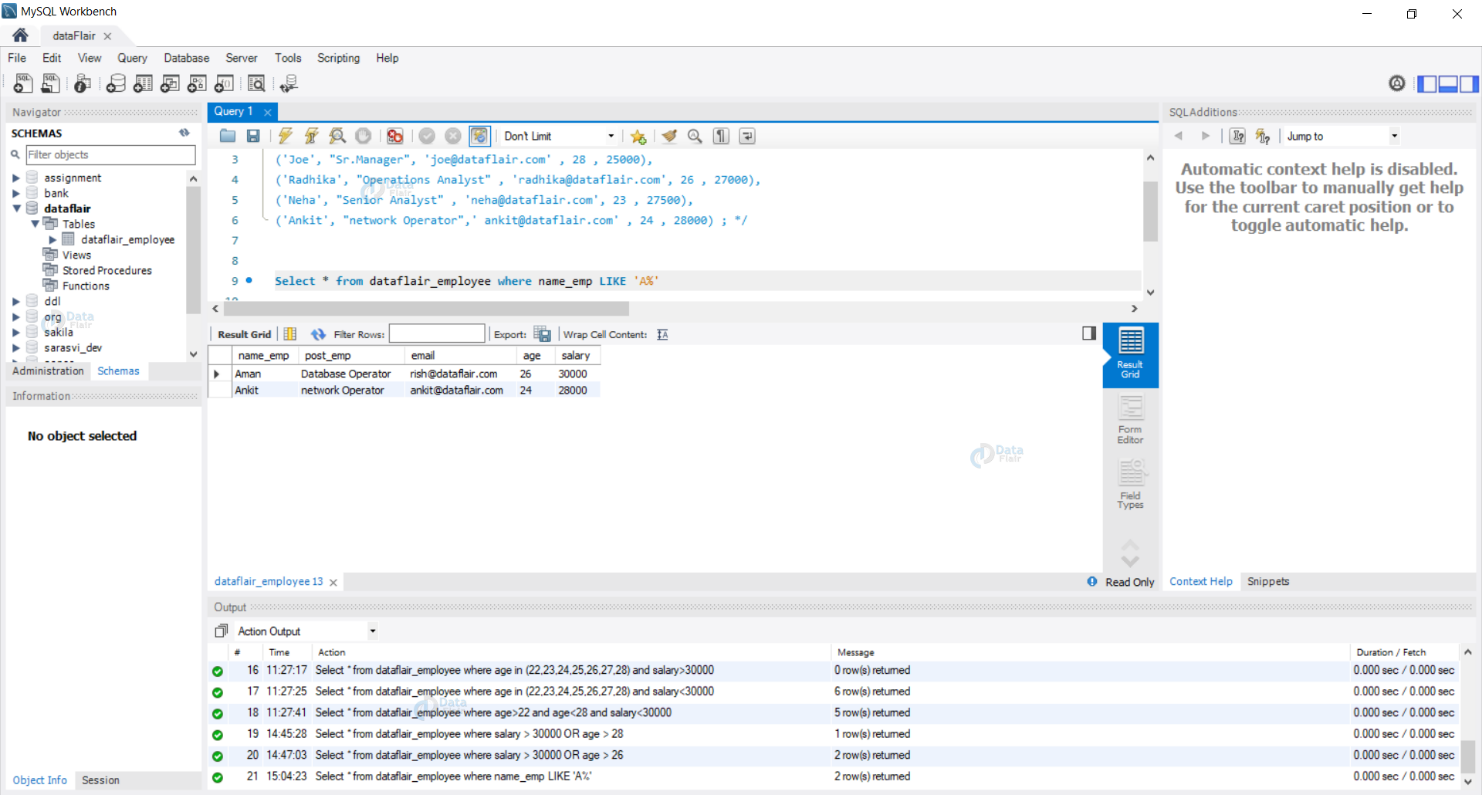
* Used to retrieve data points that satisfy the pattern passed using the like clause.

**Syntax :**

SELECT \* FROM tableName WHERE column2 LIKE pattern ;

**Example 1:** Let us find the details of employees whose name starts with A.

Select \* from dataflair\_employee where name\_emp LIKE 'A%' ;



### **6.Limit Clause in SQL**

We use the limit clause when we have a large amount of data in the table. With the help of the limit clause, we can restrict the number of rows our query returns.

**Rules and Usage:**  
1. Rules:

* We need to specify a number after the limit clause.
* Float and exponential values can’t be utilized.

2. Usage:

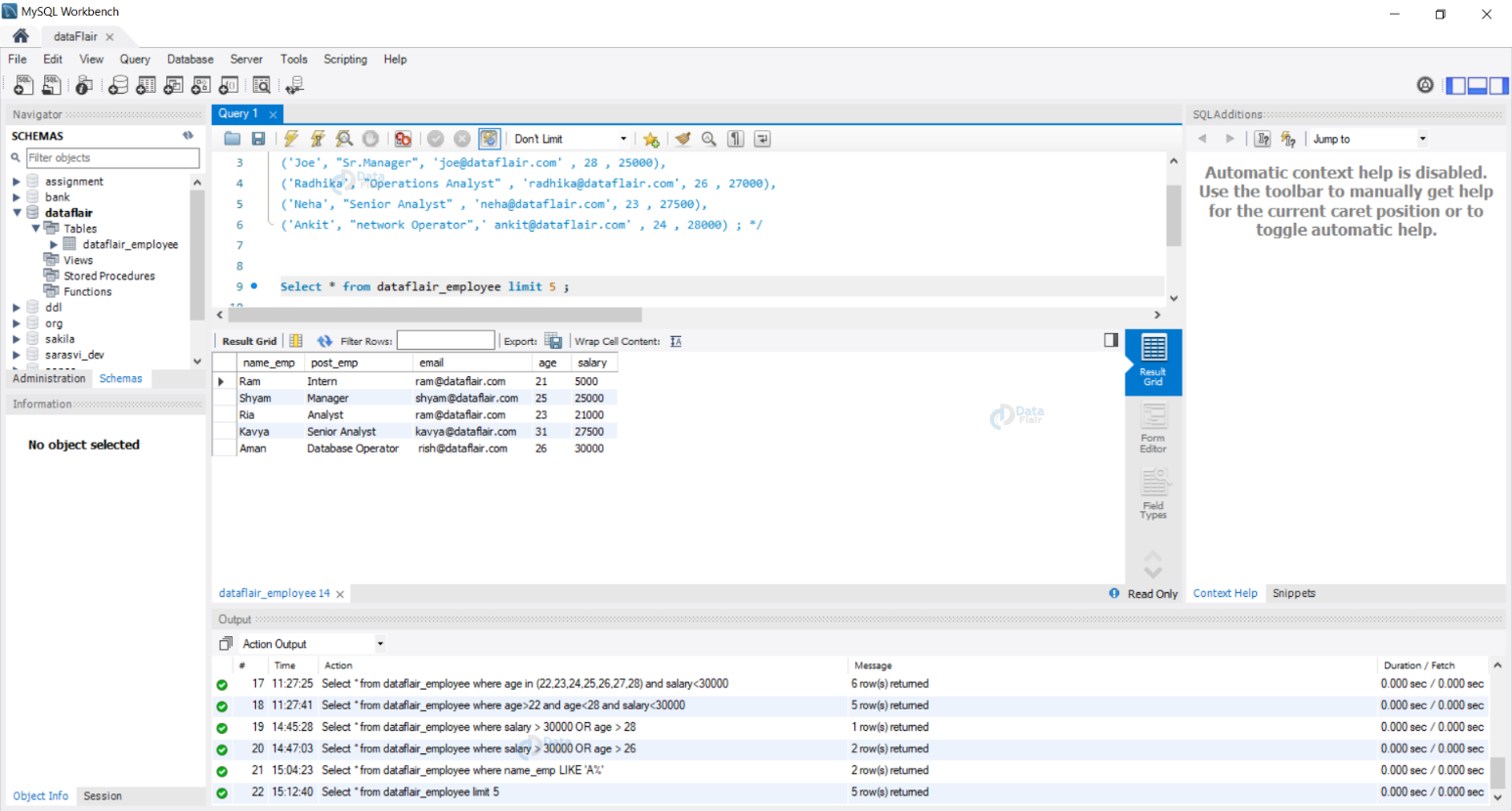
* Used to limit the data that a query would return.

**Syntax:**

SELECT \* FROM tableName LIMIT number ;

**Example 1:** Let us view the first 5 rows of data from our table DataFlair\_Employee.

Select \* from dataflair\_employee limit 5 ;



**SQL Aggregate Functions**

* SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
* It is also used to summarize the data.

## **Types of SQL Aggregation Function**

### **1. COUNT FUNCTION**

* COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
* COUNT function uses the COUNT(\*) that returns the count of all the rows in a specified table. COUNT(\*) considers duplicate and Null.

**Syntax**

COUNT(\*)

or

COUNT( [ALL|DISTINCT] expression )

**Sample table:**

**PRODUCT\_MAST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PRODUCT** | **COMPANY** | **QTY** | **RATE** | **COST** |
| Item1 | Com1 | 2 | 10 | 20 |
| Item2 | Com2 | 3 | 25 | 75 |
| Item3 | Com1 | 2 | 30 | 60 |
| Item4 | Com3 | 5 | 10 | 50 |
| Item5 | Com2 | 2 | 20 | 40 |
| Item6 | Cpm1 | 3 | 25 | 75 |
| Item7 | Com1 | 5 | 30 | 150 |
| Item8 | Com1 | 3 | 10 | 30 |
| Item9 | Com2 | 2 | 25 | 50 |
| Item10 | Com3 | 4 | 30 | 120 |

**Example: COUNT()**

SELECT COUNT(\*)

FROM PRODUCT\_MAST;

**Output:**

10

**Example: COUNT with WHERE**

SELECT COUNT(\*)

FROM PRODUCT\_MAST;

WHERE RATE>=20;

**Output:**

7

**Example: COUNT() with DISTINCT**

SELECT COUNT(DISTINCT COMPANY)

FROM PRODUCT\_MAST;

**Output:**

3

**Example: COUNT() with GROUP BY**

SELECT COMPANY, COUNT(\*)

FROM PRODUCT\_MAST

GROUP BY COMPANY;

**Output:**

Com1 5

Com2 3

Com3 2

**Example: COUNT() with HAVING**

SELECT COMPANY, COUNT(\*)

FROM PRODUCT\_MAST

GROUP BY COMPANY

HAVING COUNT(\*)>2;

**Output:**

Com1 5

Com2 3

### **2. SUM Function**

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

**Syntax**

SUM()

or

SUM( [ALL|DISTINCT] expression )

**Example: SUM()**

SELECT SUM(COST)

FROM PRODUCT\_MAST;

**Output:**

670

**Example: SUM() with WHERE**

SELECT SUM(COST)

FROM PRODUCT\_MAST

WHERE QTY>3;

**Output:**

320

**Example: SUM() with GROUP BY**

SELECT SUM(COST)

FROM PRODUCT\_MAST

WHERE QTY>3

GROUP BY COMPANY;

**Output:**

Com1 150

Com2 170

**Example: SUM() with HAVING**

SELECT COMPANY, SUM(COST)

FROM PRODUCT\_MAST

GROUP BY COMPANY

HAVING SUM(COST)>=170;

**Output:**

Com1 335

Com3 170

### **3. AVG function**

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

**Syntax**

AVG()

or

AVG( [ALL|DISTINCT] expression )

**Example:**

SELECT AVG(COST)

FROM PRODUCT\_MAST;

**Output:**

67.00

### **4. MAX Function**

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

**Syntax**

MAX()

or

MAX( [ALL|DISTINCT] expression )

**Example:**

SELECT MAX(RATE)

FROM PRODUCT\_MAST;

**Output:**

30

### **5. MIN Function**

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

**Syntax**

MIN()

or

MIN( [ALL|DISTINCT] expression )

**Example:**

SELECT MIN(RATE)

FROM PRODUCT\_MAST;

**Output:**

10