ORDER BY & GROUP BY

In this lesson, we will learn about the ORDER BY and GROUP BY clauses.

WE'LL COVER THE FOLLOWING

The ORDER BY clause

The SQL **ORDER BY** clause is used to sort the data of one or more columns in ascending or descending order. Some databases sort the query results in ascending order by default.

Syntax

The basic syntax of the `ORDER BY` clause is as follows:

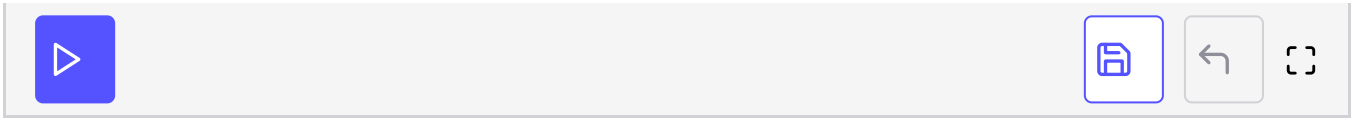
```
SELECT column-list

FROM table_name

WHERE condition

ORDER BY column1, column2, .. columnN;
```

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort is in the column-list.

Example

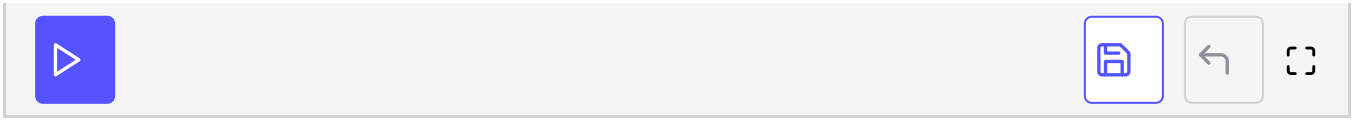
We will sort the CUSTOMERS table in ascending order by the `NAME` column:

```
SELECT *
FROM CUSTOMERS
ORDER BY NAME;
```



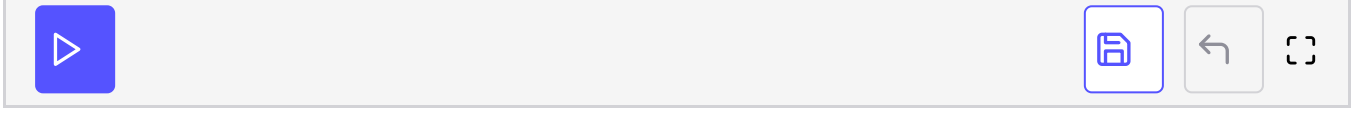
Now let's say we want to sort the list according to `NAME` but in descending order. The code below depicts this:

```
SELECT *
FROM CUSTOMERS
ORDER BY NAME DESC;
```



If we don't write anything after the `ORDER BY` clause, then the column will be sorted in ascending order by default. However, we can also specify that we want to sort the list in ascending order by using the `ASC` keyword. The code below depicts this:

```
SELECT *
FROM CUSTOMERS
ORDER BY NAME ASC;
```



The GROUP BY clause

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

Syntax

The basic syntax of a `GROUP BY` clause is shown below. The `GROUP BY` clause must follow the conditions in the `WHERE` clause and must precede the `ORDER BY`

clause if one is used.

```
SELECT column1, column2 ... columnN

FROM table_name

WHERE conditions

GROUP BY column1, column2 ... columnN

ORDER BY column1, column2 ... columnN;
```

Example

Consider the CUSTOMERS table below but with a few changes:

ID	NAME	AGE	ADDRESS	SALARY
1	Mark	32	Texas	50,000
2	Mark	23	LA	77,000
3	John	25	NY	65,000
4	Emily	23	Ohio	20,000
5	John	31	Arizona	54,000
6	Bill	25	Chicago	75,000
7	Bill	28	Florida	31,000
8	Emily	29	Michigan	43,000
9	Tom	27	Washington	35,000
10	Jane	22	Texas	45,0000

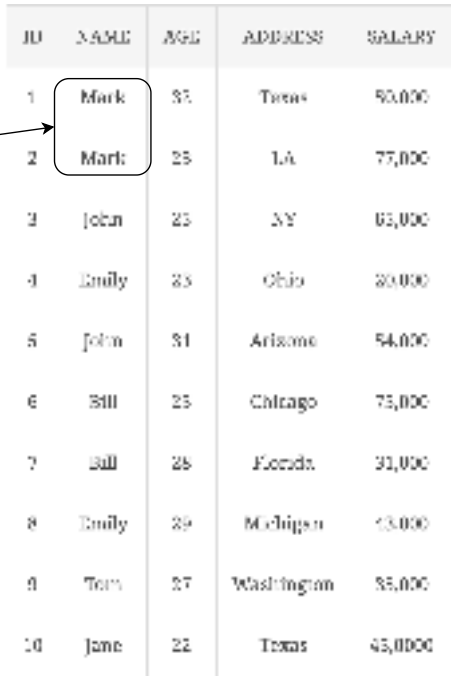
As you can see, there are duplicate names in the table above.

If you want to know the total amount earned by customers with the same name, then the `GROUP BY` query will return the following result:

As we want to sum the salaries of people with same names. We will apply the GROUP BY query on the NAME column so we can group together the people with same name.

So these two people with the name Mark will be grouped together and we will simply add their salaries

The same will be done for the different groups of people with same names.



ID	NAME	AGE	ADDRESS	SALARY
1	Mark	32	Texas	50,000
2	Mark	25	LA	77,000
3	John	23	NY	65,000
4	Emily	33	Ohio	80,000
5	John	31	Arizona	54,000
6	Bill	23	Chicago	75,000
7	Bill	26	Florida	31,000
8	Emily	29	Michigan	63,000
9	Tom	27	Washington	35,000
10	Jane	22	Texas	45,0000

As we mentioned in the previous slide, each group of people with the same name will have their salaries totalled. These totals are presented in a seperate column.

NAME	SUM(SALARY)
Bill	106000.00
Emily	63000.00
Jane	45000.00
John	119000.00
Mark	127000.00
Tom	35000.00

The code for the `GROUP BY` query is written below:

```
SELECT NAME, SUM(SALARY)
FROM CUSTOMERS
GROUP BY NAME
ORDER BY NAME;
```

In **line 3**, the `GROUP BY` statement groups the customers based on their names and then the `SUM()` function is applied over the SALARY column so we get the total salary per customer group.

Quick quiz!

Q

What will be the ouput of the following query?

```
SELECT NAME, MIN(AGE)
FROM CUSTOMERS
GROUP BY NAME;
```

COMPLETED 0%

1 of 1

In the next lesson, we will learn about the HAVING clause.