The GROUP BY clause in SQL allows users to apply functions and perform aggregations over the data by grouping together records with identical values, as specified in the resulting query.

Newcomers to SQL usually struggle to figure out the correct query syntax when using GROUP BY clauses and aggregation functions.

In the next few sections we will go through two fundamental principles that could eventually help you dissolve any confusion you may be having with aggregations and GROUP BY clauses in SQL.

First, let's create an example table that we will be referencing throughout this short tutorial in order to demonstrate a few concepts with the use of actual examples.

```
> SELECT * FROM orders;
+----+-------------+---------+------------+----------+
| id | customer_id | amount  | order_date | store_id |
+----+-------------+---------+------------+----------+
| 1  |     20      |  10.99  | 2022-02-02 |    1     |
| 2  |     138     |  21.85  | 2022-09-10 |    1     |
| 3  |     31      |  9.99   | 2021-12-28 |    2     |
| 4  |     20      |  12.99  | 2022-04-29 |    2     |
| 5  |     89      |  5.00   | 2022-03-03 |    3     |
| 6  |     31      |  4.99   | 2022-11-09 |    1     |
| 7  |     20      |  15.00  | 2022-09-10 |    3     |
| 8  |     15      | 120.00  | 2022-06-07 |    2     |
| 9  |     15      |  32.00  | 2022-04-01 |    2     |
+----+-------------+---------+------------+----------+
```

The GROUP BY clause in SQL

Now the GROUP BY clause is usually applied over categorical columns. Even though it is possible to group records over a column with continuous values, it -usually- doesn't make a lot of sense to do so.

When calling a GROUP BY clause, the query will group records together based on the value of the columns specified in the GROUP BY clause, it will then apply the aggregation function to the individual groups in order to compute the desired result and finally return the results in which we would expect to see one value per group of columns specified in the GROUP BY clause.

Some of the most common aggregation functions

include SUM, AVG, COUNT, MAX and MIN.

Now using our example table, let's assume that we would like to see the maximum order amount per customer and finally sort the result in descending order based on the maximum order . The following query would do the trick:-

```
SELECT
   customer_id,
   MAX(amount) as max_order_amount
FROM
   orders
GROUP BY
   customer_id
ORDER BY
   max_order_amount DESC;
```

And the result will be

```
+-------------+------------------+
| customer_id | max_order_amount |
+-------------+------------------+
|          15 |              120 |
|         138 |            21.85 |
|          20 |               15 |
|          31 |             9.99 |
|          89 |                5 |
+-------------+------------------+
```

What columns to include in GROUP BY clause:-

Now things may get a bit more complicated when we need to include more columns in our SELECT clause. Going forward with the query we have written earlier, now let's assume that we want to see both the minimum and maximum orders per customer, for each individual store.

Let's try to run the following query:-

```
SELECT
   customer_id,
   store_id,
   MIN(amount) as min_order_amount,
   MAX(amount) as max_order_amount
FROM
   orders
```

```
GROUP BY
   customer_id
ORDER BY
   max_order_amount DESC
;
```

Apparently, we have a syntax error that should look like the one reported below:-

ERROR: column "orders.store_id" must appear in the GROUP BY clause or be used in an aggregate function LINE 3:   store_id,   ^ SQL state: 42803 Character: 29

This is because we violated one basic principle of SQL queries that involve GROUP BY clauses. Columns specified in the SELECT statement, must either have an aggregate function applied over them or be included in the GROUP BY clause.

Going back to our example, the error is caused because we did not include staff_id in the GROUP BY clause along with customer_id. On the other hand, amount should not be included in the GROUP BY clause given that it has aggregation functions (MIN and MAX) applied over it.

Columns in the SELECT statement must either have an aggregate function applied over them or be included in the GROUP BY clause

Therefore, in order to fix our query, all we need to do is include store_id in the GROUP BY clause (given that what we want to observe is the max and min order amount per customer and store).

```
SELECT
   customer_id,
   store_id,
   MIN(amount) as min_order_amount,
   MAX(amount) as max_order_amount
FROM
   orders
GROUP BY
   customer_id,
 store_id
ORDER BY
   max_order_amount DESC;
```

Aggregations in WHERE statements and how to use HAVING:-

Going back to our first example, let's assume that we want to calculate the maximum order amount per customer that have spent more than 5.00.

```
SELECT
   customer_id,
   MAX(amount) as max_order_amount
FROM
   orders
WHERE
   MAX(amount) > 5
GROUP BY
   customer_id
ORDER BY
   max_order_amount DESC;
```

If we attempt to run the query above, we will end up with the following error:-

ERROR: aggregate functions are not allowed in WHERE

The problem with the above query is that we've tried to include an aggregation function (i.e. MAX) in a WHERE clause. The syntax of SQL clearly specifies that WHERE statements cannot refer to aggregations specified in the SELECT clause.

But what is the workaround then?

WHERE statements cannot refer to aggregations specified in the SELECT clause
Rather than specifying conditions over an aggregated column within the WHERE clause, we can instead use the HAVING clause that allows us to useGROUP BY clauses followed by filters over aggregation results.

Therefore, we should instead
replace WHERE with HAVING clause, as shown below:-

```
SELECT
   customer_id,
   MAX(amount) as max_order_amount
FROM
   orders
```

```
GROUP BY
    customer_id
HAVING
 MAX(amount) > 5
ORDER BY
    max_order_amount DESC;
```

And the resulting set of records should like the one shared below:-

customer_id	max_order_amount
15	120
138	21.85
20	15
31	9.99

Final Thoughts

Working with GROUP BY clauses as an SQL beginner can sometimes be quite frustrating. In today’s short tutorial we went through two of the most fundamental principles you always need to have in mind when working with aggregations in your SQL queries.

To recap, it’s important to ensure that all the columns specified in SELECT clauses, will either have an aggregation function applied over them (such as SUM, AVG etc.) or are specified in the GROUP BY clause.

Additionally, you need to remember that aggregations (specified in SELECT clause) cannot be reference in WHERE clauses. If you wish to do so, you will need to specify your condition in HAVING clause.