Querying Relational Data using SQL

Understand the hierarchy of your database

Before you get started, it's important to become accustomed to your database and its hierarchy. If you have multiple databases of data, you'll need to zero in on the location of the data you want to work with.

For example, let's pretend we're working with multiple databases about people in the United States. Type in the query "SHOW DATABASES;". Our results may show that you have a couple of databases for different locations, including one for New England.

Within your database, you'll have different tables containing the data you want to work with. Using the same example above, let's say we want to find out which information is contained in one of the databases. If we use the query "SHOW TABLES in NewEngland;", we'll find that we have tables for each state in New England: people_connecticut, people_maine, people_massachusetts, people_newhampshire, people_rhodeisland, and people_vermont.

Finally, you need to find out which fields are in the tables. Fields are the specific pieces of data that you can pull from your database. For example, if you want to pull someone's address, the field name may not just be "address" -- it may be separated into address_city, address_state, address_zip. In order to figure this out, use the query "Describe people_massachusetts;". That will provide a list of all of the data that you can pull using SQL.

Let's do a quick review of the hierarchy using our New England example:

- Our database is: NewEngland.
- Our tables within that database are: people_connecticut, people_maine, people_massachusetts, people_newhampshire, people_rhodeisland, and people_vermont.
- Our fields within the people_massachusetts table include: address_city, address_state, address_zip, hair_color, first_name, and last_name.

Now, to learn how to write a simple SQL query, let's use the following example:

Who are the people who have red hair in Massachusetts and were born in 2003 organized in alphabetical order?

SELECT

SELECT chooses the fields that you want displayed in your chart. This is the specific piece of information that you want to pull from your database. In the example above, we want to find the *people* who fit the rest of the criteria.

Here is our SQL query:

SELECT
first_name,
 last_name

FROM

FROM pinpoints the table that you want to pull the data from. In the earlier section, we found that there were six tables for each of the six states in New England: people_connecticut, people_maine, people_massachusetts, people_newhampshire, people_rhodeisland, and people_vermont. Because we're looking for people in Massachusetts specifically, we'll pull data from that specific table.

Here is our SQL query:

SELECT
 first_name,
 last_name
FROM
people_massachusetts

WHERE

WHERE allows you to filter your query to be more specific. In our example, we want to filter our query to include only people with red hair who were born in 2003. Let's start with the red hair filter.

Here is our SQL query:

```
SELECT
 first_name,
 last_name
FROM
 people_massachusetts
WHERE
hair_color = "red"
```

hair_color could have been part of your initial SELECT statement if you'd wanted to look at all of the people in Massachusetts along with their specific hair color. But if you want to filter to see *only* people with red hair, you can do so in the WHERE statement.

AND

AND allows you to add additional criteria to your WHERE statement. Remember, we want to filter by people who had red hair in addition to people who were born in 2003. Since our WHERE statement is taken up by the red hair criteria, how can we filter by a specific year of birth as well?

That's where the AND statement comes in. In this case, the AND statement is a date property -- but it doesn't necessary have to be. (Note: Be to check the format of your dates with your product team to make sure it is in the correct format.)

Here is our SQL query:

```
SELECT
 first_name,
 last_name
FROM
people_massachusetts
WHERE
hair_color = "red"
AND
```

birth_date BETWEEN '2003-01-01' AND '2003-12-31'

ORDER BY

When you create SQL queries, you shouldn't have to export the data to Excel. The calculation and organization should be done within the query. That's where the "ORDER BY" and "GROUP BY" functions come in. First, we'll look at our SQL queries with the ORDER BY and then GROUP BY functions, respectively. Then, we'll take a brief look at the difference between the two.

Your ORDER BY clause will allow you to sort by any of the fields that you have specified in the SELECT statement. In this case, let's order by last name.

Here is our SQL query:

```
SELECT
 first_name,
 last_name
FROM
 people_massachusetts
WHERE
 hair_color = "red"
AND
birth_date BETWEEN '2003-01-01' AND '2003-12-31'
ORDER BY
 last_name
;
```

GROUP BY

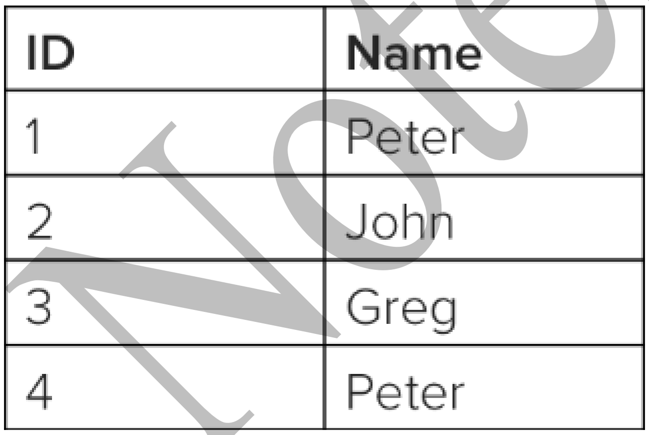
"GROUP BY" is similar to "ORDER BY," but it will aggregate data that has similarities. For example, if you have any duplicates in your data, iyou can use "GROUP BY" to count the number of duplicates in your fields.

Here is your SQL query:

```
SELECT
first_name,
last_name
FROM
people_massachusetts
WHERE
hair_color = "red"
AND
birth_date BETWEEN '2003-01-01' AND '2003-12-31'
GROUP BY
 last_name
;
```

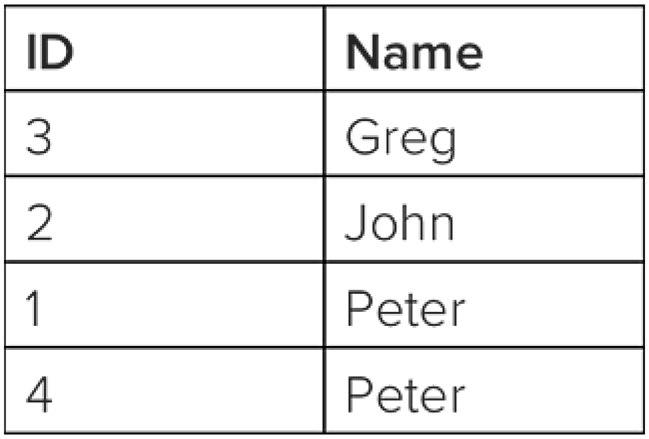
ORDER BY VS. GROUP BY

To clearly show you the difference between an "ORDER BY" statement and a "GROUP BY" statement, let's step outside our Massachusetts example briefly to look at a very simple dataset. Below is a list of four employees' ID numbers and names.



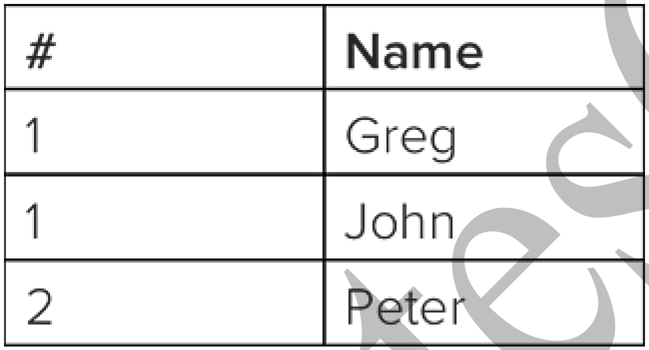
ID	Name
1	Peter
2	John
3	Greg
4	Peter

If we were to use an ORDER BY statement on this list, the names of the employees would get sorted in alphabetical order. The results would look like this:



ID	Name
3	Greg
2	John
1	Peter
4	Peter

If we were to use a GROUP BY statement, the employees would be counted based on the number of times they appeared in the initial table. Note that Peter appeared twice in the initial table. The results would look like this:



#	Name
1	Greg
1	John
2	Peter

With me so far? Okay. Let's return to the SQL query we've been creating about red-haired people in Massachusetts who were born in 2003.

LIMIT

Depending on the amount of data you have in your database, it may take a long time to run the queries. It can be frustrating if you find yourself waiting a long time to run a query that you didn't really want to begin with. If you want to test our query, the LIMIT function is a great one to use because it allows you to limit the number of results you get.

For example, if we suspect there are millions of people who have red hair in Massachusetts, we may want to test out our query using LIMIT before we run it in full to make sure we're getting the information we want. Let's say, for instance, we only want to see the first 100 people.

Here is our SQL query:

```
SELECT
 first_name,
 last_name
FROM
people_massachusetts
WHERE
hair_color = "red"
AND
birth_date BETWEEN '2003-01-01' AND '2003-12-31'
ORDER BY
 last_name
LIMIT
100
;
```

That's it for the basics!