Queries for a Single Table

Let's look at basic 5 clauses of the SELECT statement. (Select, From, Where, Order By, and Limit) SELECT is really the only one that is required.

But they do need to be used in order as they are shown here for them to work properly. Let's look at what each of them does.

Here we see a query using all 5 of the clauses. Below the query we see the result set or the data that is returned when the query runs.

SELECT describes the columns in the result set or values that will be returned when queried. This is where you will list all the fields or columns that you want to show up in the result set. You separate each column with a comma. Here we want first name and last name to show up .

FROM tells what table you are getting the data from. This is always required if you are getting data from a database table. In this example, we want the data to come from the artist table.

WHERE is where you can place criteria for what data you want from the table, to filter the rows down to only those that match a condition you state. Here we only want names of artists from Italy.

ORDER BY will sort the rows in the order you specify (like alphabetizing, or sorting biggest to smallest, etc.) Our results will be alphabetical by last name a to z.

LIMIT tells how many rows of the result set to show. Here we are limiting the result set to the first two rows that are returned. So we see our result set shows two artist: Leonardo da Vinci and Michelangelo Simoni. Their first and last names from the artist table, they are both from Italy and they are sorted by last name alphabetically and only two rows show up.

Let's look more closely at the SELECT clause. There are different ways to code columns with SELECT. You can reference all columns of a table with the asterisk *, or list columns names (like we saw in our example), use can also use calculated columns and functions. as part of the SELECT clause.

When you use the asterisk every column in the table will be returned. It is OK to use an asterisk but once your project is developed and in production you should probably avoid it.

You can also list each column separated with commas. Then only those columns listed will show up in the result set.

You can show calculations in the SELECT clause as well. Here we are using the date of death minus the date of birth to get the age of the artist at their death. The AS keyword is followed with an alias name for the column header. If we left out the AS and alias name then the column header would have just been whatever the calculation was in our case 'dod-dob'. So the alias is a nice way to make sure we have a more meaningful column header.

SELECT statement calculations recognize all these Arithmetic Operators. If you end up using more than one operator in a single calculation be aware of the order just like we learned in elementary school, there is an order of precedence. The 3rd column here shows the precedence from right to left as the calculation is processed.

Functions can also be a part of the SELECT clause. Here is an example of a function called CONCAT that takes groups of strings and in this case a calculation and joins them together. There are many different types of functions. We learn about these functions in a different lesson.

If you end up having a query with a result set that has repeated values you can use the DISTINCT keyword to eliminate the duplicate rows included in the result set.

Here is a query getting all the countries of the artists and if they are local or not. We can see that some of the result set rows are duplicates.

Here is the same query with the keyword DISTINCT after the SELECT keyword. Now you see DISTINCT prevents duplicate rows from being included in the result set.

The WHERE clause will filter the rows from the result set according to the conditions you give. This enables you to see just the rows you need.

Comparison operators are used in the WHERE clause. Comparison Operators do just what they say, they help you compare one value or string to another. Are they equal, is one bigger or comes later in the alphabet, or smaller or are they not equal to each other. Then according to the results of the conditions only those rows in the result set will show up.

Let's take a look at these comparison operators in action.

In this first example using the equal sign, the WHERE clause read WHERE country = "Italy" so only the artist's first and last name will show up if the country is equal to Italy. Italy is another attribute in each row of the artist table, even though we don't show the column in the SELECT statement we can still filter using country. So Leonardo and Michelangelo are both from Italy.

Let's look at the greater than and less than sign. In the example on the left, the WHERE clause states, WHERE dob > 1606, so only the artist will show up if their

date of birth is greater than or after the year 1606. The example on the right adds the equal sign to the greater than sign now you can see all artist whose date of birth is after **or on** the year 1606. So Rembrandt now shows up as well because he was born in 1606, before it was only those born after 1606.

On the left, if we use the less than sign we are saying show everyone that was born before 1606. With the less than and equal sign we are saying show everyone that was born before **or on** 1606. So again Rembrandt will show up.

The less than and greater than sign used together or the exclamation and equal sign together mean not equal to. Our WHERE clause says WHERE dob is not equal to '1606'. So here we want everyone who is not born in 1606 which is everyone except Rembrandt because he is born in 1606. So he does not show up.

You can combine multiple operators in different ways to filter your results down even further.

When you use the AND between two different conditions, both conditions have to be true for it to show up in the result set. So now the WHERE clause states WHERE country is equal to Italy AND lastname = 'da Vinci'. Not only does the artist have to be from Italy, he also has to have the last name "da Vinci". Both those conditions have to be true. So AND will filter down really narrowly to get just the results that match both conditions. Only da Vinci has that last name and is from Italy.

When you use OR between the same two conditions, now either of the conditions can be true for it to show up in the result set. Everyone from Italy or those with the last name Picasso will show up. So now our filter is not so narrow. Anyone from Italy and any one with the last name Picasso will show up in the result set.

You can also reverse (or negate) your condition with NOT. This will then show everyone who is not from Italy. There is also an order of precedence with AND, OR and NOT as well. NOTs will be evaluated first then AND and then OR. Use parenthesis to make sure they are evaluated in the order you'd like them to be.

There are even more operators that can be used with a WHERE clause to filter down your results. IN, BETWEEN, LIKE and REGULAR EXPRESSIONS.

With the IN operators a test expression is compared to a list of expressions in the IN phrase. If the test expression is equal to one of the expressions in the list, the row will be a part of the result set. Here we see that if country is equal to Italy or France, the row will show up in the result set. It's the same as if we said "WHERE country is equal to Italy AND country is equal to France in a compound operator but the syntax is simpler.

The BETWEEN operator when used in a WHERE clause will allow you to compare a test expression with a range of values. If the value falls within the range the row will be included in the result set. Here are a few example of how that might work. In the example on the left, if the date of birth lands in the range between the year 1500 to the year 1900 it will show up in the result set. The example on the right shows that you can also use it alphabetically. If the last name alphabetically lands between the letter H and the letter W then it will be included.

The LIKE operator matches specific simple string patterns to a test expression. In the first one you are saying if the last name starts with van and the percentage sign says that the van can be followed by any number of characters after it. This matches two different last names, van Gogh and van Rijn show up. Another symbol that can be used with LIKE is the _ underscore character instead of any number of characters like the % percent sign, the _ underscore represents any single character.

The REGEXP (regular expression) operator allows you to create much more complex string patters to test expressions. There are many regular expression symbols that can be used; here are just a few. The top left example is doing the same thing the LIKE example did getting all last names that begin with "van". The carrot symbol represents the beginning of a string. The next one down gets all first names that end in "t" . The dollar sign represents the end of the a string. The last example is saying get any first name that has an "en' or an "an" anywhere in it. So Vincent as and 'en', Rembrant has an 'an' and so forth.

You can even find all nulls with IS NULL or all rows that don't have a null with IS NOT NULL. The example on the left is showing all rows where there is a null for the middle name. The other is showing all the artist that do have a middle name. The middle name is not null or not empty.

We are getting there. 2 more of the 5 clauses to go. There's a lot to know here.

ORDER BY specifies the sort order for the rows in the result set. Most of the time you use a column name to tell how you want it sorted.

In this example the results are alphabetized by last name. The default is to sort in ascending order. If you wanted to reverse the order to descending you have to use the DESC keyword.

This one shows the same query but with the keyword DESC for descending. Now the order is reversed. Those later in the alphabet show up first.

You can even sort on multiple columns. This is sorted by country and then within that country by date of birth.

The LIMIT clause specifies the maximum number of rows that will be returned. Even if there were hundreds of rows returned if you said LIMIT 5 then only the first 5 rows would be returned.

Here, on the left, is a query run with all the rows being that fit the criteria of WHERE returned and then, on the right, if we add LIMIT 4 at the end then only 4 rows would be returned. You might have thousands of results but you only need to see the top 10 or something like that. That is when LIMIT would come in handy.

So there we have the common clauses of getting data from a table.