SQL Basic Commands

Retrieving data from a database is one of the main uses of SQL. In SQL, there are two main clauses that are used to pull data: SELECT and FROM. Anytime you want to learn more about a topic, you could use SQL to retrieve information from a database about that topic! For example, if you're deciding between two items to purchase, you can use SQL to retrieve information that helps you compare their features and prices.

SELECT and FROM

The SELECT FROM statement is used to select data from a database. The SELECT clause begins with the SELECT keyword and is followed by specified column names that you want to pull. You can specify as many or as few column names as you want.

This clause is then followed by the FROM clause, which specifies which table in your database you're pulling the columns from. The structure of the SELECT FROM statement is this:

```
SELECT column1, column2, …
FROM table_name;
```

Remember to include the semicolon ; at the end of a statement. Imagine that you're a manager at a company and you're checking three of your employees' logged work hours for the month to make sure that their entries are correct. Say there is a table, August_21_Hours, that has this information, with columns corresponding to every employee name at the company. You can pull your employees' hours logged by using a SELECT FROM statement, shown below.

```
SELECT AndreC, TiffanyL, GraceW
FROM August_21_Hours;
```

As you can see, the SELECT FROM statement is pretty straightforward. Now, what if you want to see the hours logged for every employee at the company? SQL has a shortcut to pull all columns without having to write out each one in the statement: you can use an asterisk in place of the column names.

For this example, the statement would look like this:

```
SELECT *
FROM August_21_Hours;
```

*Using the asterisk * in place of a column name after SELECT is a shortcut to pull all columns from a table.*

LIMIT

If you're just trying to get a sense of the data or output, you may want to only see the first 10, 50, or 100 rows of your query results. In general, if you want to limit the number of records shown in the Data Output, you can use the LIMIT command using the following structure.

```
SELECT column1, column2, …
FROM table_name
LIMIT number of records;
```

Below, the LIMIT command is used to pull just the first 10 records of the table film.

Query Editor | Query History | Scratch Pad

```
SELECT *
FROM film
LIMIT 10;
```

Data Output | Explain | Messages | Notifications

	film_id [PK] integer	title character varying (255)	description text	release_year integer	language_id smallint	original_language_id smallint	rental_duration smallint	rental nume
1	1	ACADEMY DINOSAUR	A Epic Drama of...	2006	1	[null]	6	
2	2	ACE GOLDFINGER	A Astounding E...	2006	1	[null]	3	
3	3	ADAPTATION HOLES	A Astounding R...	2006	1	[null]	7	
4	4	AFFAIR PREJUDICE	A Fanciful Docu...	2006	1	[null]	5	
5	5	AFRICAN EGG	A Fast-Paced D...	2006	1	[null]	6	
6	6	AGENT TRUMAN	A Intrepid Panor...	2006	1	[null]	3	
7	7	AIRPLANE SIERRA	A Touching Sag...	2006	1	[null]	6	
8	8	AIRPORT POLLOCK	A Epic Tale of a ...	2006	1	[null]	6	
9	9	ALABAMA DEVIL	A Thoughtful Pa...	2006	1	[null]	3	
10	10	ALADDIN CALENDAR	A Action-Packe...	2006	1	[null]	6	

Filtering data with comparisons

In the previous lesson, you learned how to use the basic SELECT FROM statement to retrieve columns of data from a table. Though SELECT FROM is a useful statement on its own, you'll often want to make your query more specific. If you're planning an upcoming trip and looking through a database of vacation rentals, for example, you could limit the search to show only those rentals that are less than a certain price per night. Or, if you're shopping for a new computer and you have a table containing all the different computers that you can purchase, you can limit your search to computers with a certain amount of memory.

In SQL, you can add comparison operators to make your SELECT FROM statement include these kinds of specifications.

The WHERE clause

The WHERE clause is used in a SELECT FROM statement to filter records. If you want to pull data based on a certain condition, use WHERE after the FROM clause. Here's the structure for using the WHERE clause:

```
SELECT column1, column2, …
FROM table_name
WHERE condition;
```

The WHERE clause is the basis for any kind of SQL filtering. It indicates some conditions that must be met, and all data that meets those conditions is retrieved. The following operators can be used to specify the condition.

The > operator

The > operator is used to specify that the data retrieved should be greater than a certain amount.

```
SELECT *
FROM payment
WHERE amount > 2.99;
```

The < operator

The < operator is used to specify that the data retrieved should be less than a certain amount.

```
SELECT *
FROM payment
WHERE amount < 2.99;
```

The = operator

The = operator is used to specify that the data retrieved should be equal to a certain amount. If you're working with non-numeric data, the = operator is used to specify that the data retrieved should have a certain name.

```
SELECT *
FROM payment
WHERE amount = 2.99;
```

Or

```
SELECT *
FROM film
WHERE rating = 'PG-13';
```

SQL requires single quotation marks around the text if you're specifying a text value. Numeric values, on the other hand, don't require quotation marks.

The >= operator

The >= operator is used to specify that the data retrieved should be greater than or equal to a certain amount.

```
SELECT *
FROM payment
WHERE amount >= 2.99;
```

The <= operator

The <= operator is used to specify that the data retrieved should be less than or equal to a certain amount.

```
SELECT *
FROM payment
WHERE amount <= 2.99;
```

The <> and != operators

The <> and != operators perform the same function. These operators are used to specify that the data retrieved should not be equal to a certain amount. If you're working with non-numeric data, they are used to specify that the data retrieved should not have the specified name.

```
SELECT *
FROM payment
WHERE amount <> 2.99;
```

In the query above, you could use != in place of <>, and you would still get the same results. Remember, the != and <> operators are interchangeable.

Execute this query. In your Data Output, you won't see any payments of $2.99.

Similarly, if you want to pull records from the table film if the rating isn't R, you could use the following query:

```
SELECT *
FROM film
WHERE rating <> 'R';
```