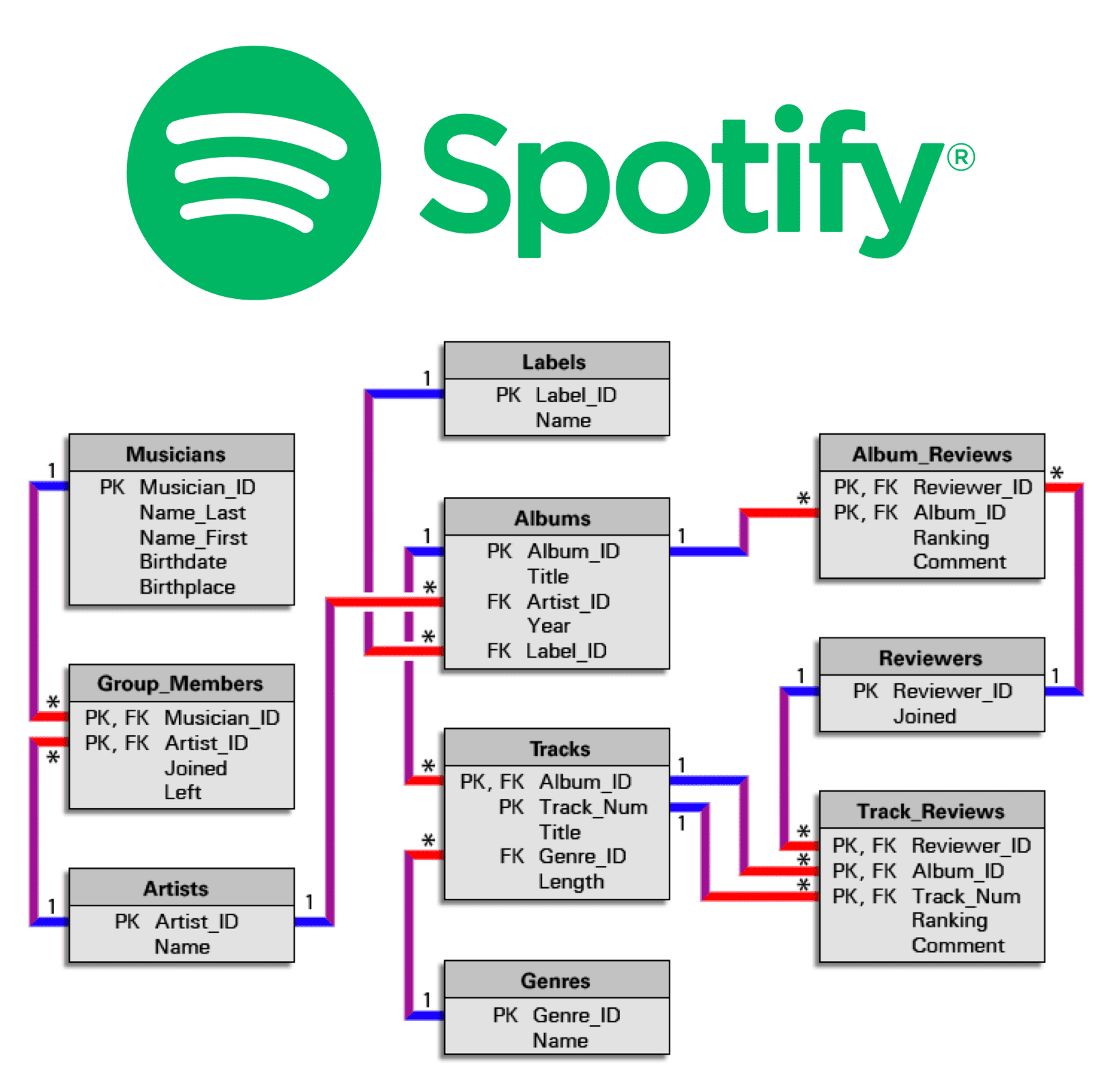
**SQL JOINS Spotify**

So far in the SQL we've only been analyzing data from one table at a time. However, in the real world, companies have databases containing thousands of tables. To combine multiple tables, and analyze their data simultaneously, we can write a **JOIN** SQL query – the focus of this tutorial.

**Spotify SQL JOIN Case Study**

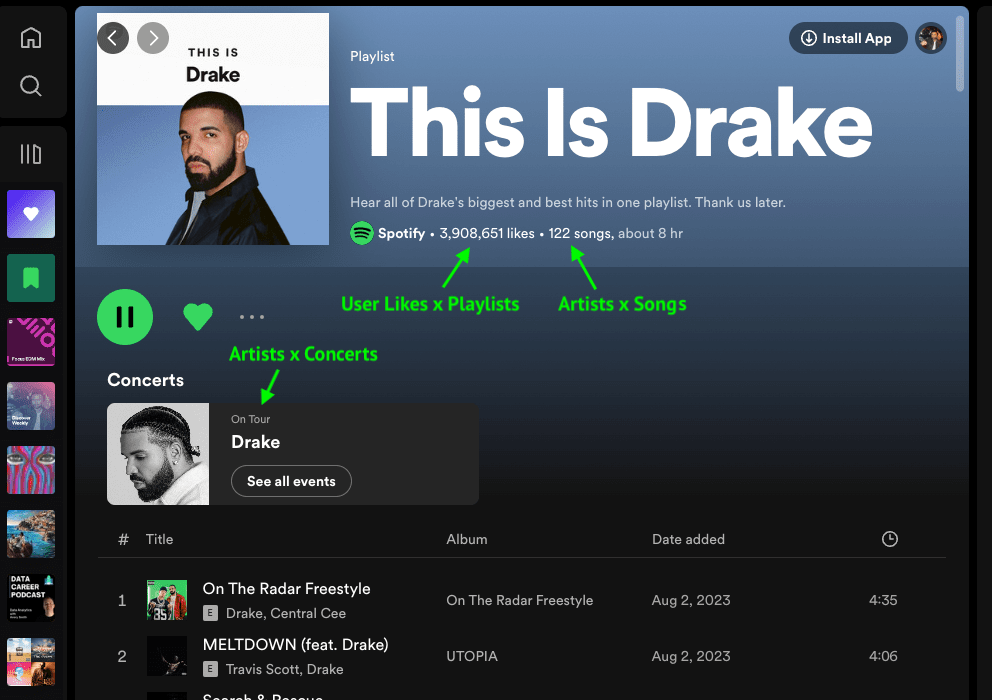
For example, let's pretend we work at Spotify where we have a table for albums, a table for artists, a table to represent users, and the list 🎵 *goes on and on and on*🎵 (sung to the tune of [Don't Stop Believin'](https://open.spotify.com/track/4bHsxqR3GMrXTxEPLuK5ue)).



As a Data Analyst or Data Scientist, you'll almost always be pulling data from multiple tables, which is why the SQL JOIN is crucial.

For example, at Spotify you might analyze Drake 🐐 by writing queries that combine two tables like:

* **JOIN** the **artists** and **concerts** table to analyze upcoming Drake concerts.
* **JOIN** the **user\_likes** and **playlists** table to count the # of likes on the *This Is Drake* playlist.
* **JOIN** the **artists** and **songs** table to rank Drake's top songs by play count.



Now that you understand the real-world use case of a **JOIN**, let's write a specific SQL join query using Spotify data.

**SQL JOIN Example**

Suppose you work at Spotify and want to join information in the **artists** table with information in the **songs** table. Here's the schema for the two tables:

**artists Example Input:**

| **artist\_id** | **artist\_name** | **label\_owner** |
| --- | --- | --- |
| 101 | Ed Sheeran | Warner Music Group |
| 120 | Drake | Warner Music Group |
| 125 | Bad Bunny | Rimas Entertainment |

**songs Example Input:**

| **song\_id** | **artist\_id** | **name** |
| --- | --- | --- |
| 55511 | 101 | Perfect |
| 45202 | 101 | Shape of You |
| 22222 | 120 | One Dance |
| 19960 | 120 | Hotline Bling |

Notice how the two tables have the **artist\_id** column in common:

* In the **artists** table, you have information about each **artist\_id**, like their artist name and what music label they work with.
* in the **songs** table, you have information about each song, which includes which **artist\_id** made that particular song.

Now, let's actually build our SQL JOIN query to combine the **artists** and **songs** table.

**How To Write A SQL JOIN**

There are three key parts to get right when writing a **JOIN** in SQL. First, you need to specify what columns to show. For our **artists** x **songs** example, we'll just use **SELECT \*** for that.

Secondly, you need to specify the names of the two tables we are joining. Since we are joining the **artists** with **songs** table, our query so far is as follows:

**SELECT \***

**FROM artists**

**JOIN songs;**

The third and final in a SQL JOIN query is the **ON** clause, which explains to the RDBMS (Relational Database Management System) how the two tables relate to each other.

We need to explicitly write in the SQL query that the **artists\_id** column in the **artists** table matches up against the **artist\_id** column in the **songs** table with this clause:

**ON artists.artist\_id = songs.artist\_id**

Putting these three parts together, we get the following SQL JOIN query:

**SELECT \***

**FROM artists**

**JOIN songs**

**ON artists.artist\_id = songs.artist\_id;**

The above query combines info from the **artists** and **songs** table into the result below:

| **artist\_id** | **artist\_name** | **label\_owner** | **song\_id** | **artist\_id** | **name** |
| --- | --- | --- | --- | --- | --- |
| 101 | Ed Sheeran | Warner Music Group | 55511 | 101 | Perfect |
| 101 | Ed Sheeran | Warner Music Group | 45202 | 101 | Shape of You |
| 120 | Drake | Warner Music Group | 22222 | 120 | One Dance |
| 120 | Drake | Warner Music Group | 19960 | 120 | Hotline Bling |
| 125 | Bad Bunny | Rimas Entertainment | 12636 | 125 | Mia |

Now, let's practice writing a **JOIN** on a different dataset.

**Easy SQL JOIN Practice Exercise**

Suppose you work as a Data Scientist at the stock-trading app Robinhood. Assume you're given access to a table called **trades** which contains information about trades placed on the platform, and a table called **users** which has information about a specific user.

Here's what the data looks like in both tables:

**trades Example Input:**

| **order\_id** | **user\_id** | **price** | **quantity** | **status** | **timestamp** |
| --- | --- | --- | --- | --- | --- |
| 100101 | 111 | 9.80 | 10 | Cancelled | 08/17/2022 12:00:00 |
| 100102 | 111 | 10.00 | 10 | Completed | 08/17/2022 12:00:00 |
| 100259 | 148 | 5.10 | 35 | Completed | 08/25/2022 12:00:00 |
| 100264 | 148 | 4.80 | 40 | Completed | 08/26/2022 12:00:00 |
| 100305 | 300 | 10.00 | 15 | Completed | 09/05/2022 12:00:00 |
| 100400 | 178 | 9.90 | 15 | Completed | 09/09/2022 12:00:00 |
| 100565 | 265 | 25.60 | 5 | Completed | 12/19/2022 12:00:00 |

**users Example Input:**

| **user\_id** | **city** | **Email** | **signup\_date** |
| --- | --- | --- | --- |
| 111 | San Francisco | [rrok10@gmail.com](mailto:rrok10@gmail.com) | 08/03/2021 12:00:00 |
| 148 | Boston | [sailor9820@gmail.com](mailto:sailor9820@gmail.com) | 08/20/2021 12:00:00 |
| 178 | San Francisco | [harrypotterfan182@gmail.com](mailto:harrypotterfan182@gmail.com) | 01/05/2022 12:00:00 |
| 265 | Denver | [shadower\_@hotmail.com](mailto:shadower_@hotmail.com) | 02/26/2022 12:00:00 |
| 300 | San Francisco | [houstoncowboy1122@hotmail.com](mailto:houstoncowboy1122@hotmail.com) | 06/30/2022 12:00:00 |

**Write a SQL query to join the trades and users table.**

The output should look something like this:

| **order\_id** | **user\_id** | **quantity** | **status** | **date** | **price** | **user\_id** | **city** | **email** | **signup\_date** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100102 | 111 | 10 | Completed | 08/17/2022 12:00:00 | 10.00 | 111 | San Francisco | [rrok10@gmail.com](mailto:rrok10@gmail.com) | 08/03/2021 12:00:00 |
| 100101 | 111 | 10 | Cancelled | 08/17/2022 12:00:00 | 9.80 | 111 | San Francisco | [rrok10@gmail.com](mailto:rrok10@gmail.com) | 08/03/2021 12:00:00 |
| 100900 | 148 | 50 | Completed | 07/14/2022 12:00:00 | 9.78 | 148 | Boston | [sailor9820@gmail.com](mailto:sailor9820@gmail.com) | 08/20/2021 12:00:00 |
| 100259 | 148 | 35 | Completed | 08/25/2022 12:00:00 | 5.10 | 148 | Boston | [sailor9820@gmail.com](mailto:sailor9820@gmail.com) | 08/20/2021 12:00:00 |
| 100264 | 148 | 40 | Completed | 08/26/2022 12:00:00 | 4.80 | 148 | Boston | [sailor9820@gmail.com](mailto:sailor9820@gmail.com) | 08/20/2021 12:00:00 |
| 100777 | 178 | 60 | Completed | 07/25/2022 17:47:00 | 3.56 | 178 | San Francisco | [harrypotterfan182@gmail.com](mailto:harrypotterfan182@gmail.com) | 01/05/2022 12:00:00 |
| ... | ... |  |  |  |  |  |  |  |  |