


Getting Started with SQL - Introduction



<https://github.com/learn-co-curriculum/dsc-getting-started-sql-intro>  <https://github.com/learn-co-curriculum/dsc-getting-started-sql-intro/issues/new/choose>

Introduction

In this section, you'll learn about SQL, which stands for Structured Query Language. It has been around since the 1970s and there are many different dialects of the language including MySQL, SQLite, and PostgreSQL, to name a few.

Each of these dialects has particularities such as specific functions or keywords for that specific implementation. All of them, however, have the same basic structures including keywords like **SELECT** for querying databases, and the same general database architecture.

The Structure of SQL Databases

A SQL **database** is a high-level container containing one or more **tables**. The database can be a continuously-running server that receives requests and returns responses (MySQL, PostgreSQL) or it can be a specially-formatted file on disk (SQLite). We will primarily use SQLite in these lessons because it is lightweight and portable (and therefore useful for educational purposes), but it's important to be aware that other more "heavy duty" types of databases are available and are used more often in industry settings.

A SQL table contains tabular data (data stored in rows and columns), similar to a CSV. Every table has a defined set of columns, and then we store any number of what we refer to as 'records' as rows in our database. A record is just information referring to one specific entity. For instance, if you had a table called "people" you could imagine a structure like this:

id	name	age	email
1	Bob	29	bob@flatironschool.com
2	James	28	james@flatironschool.com
3	John	28	john@flatironschool.com

Each column has a name, and each row contains the corresponding information about a person. Unlike a CSV, a SQL table can also enforce the data types of the columns, which are described in the table **schema**. The schema for this table might look something like this:

```
CREATE TABLE people (  
  id PRIMARY KEY,
```



Help

```
age INTEGER,  
email TEXT  
);
```

Connecting to and Querying SQL Databases

As a data scientist, your primary use case of SQL will be querying data stored within databases. To do this, you connect to the database with some sort of tool.

- Most SQL databases have an associated **command-line interface** where you can write SQL queries without any additional languages or tools. For SQLite, this is called `sqlite3` ([documentation here](https://sqlite.org/cli.html) → [_](https://sqlite.org/cli.html)).
- You can also connect through a different **coding language such as Python**. For SQLite, the Python module is called `sqlite3` ([documentation here](https://docs.python.org/3/library/sqlite3.html) → [_](https://docs.python.org/3/library/sqlite3.html)). (Yes, the command-line interface and Python module have the same name.)
- One other approach is using a **GUI (graphical user interface)** tool. For SQLite, a good one is DB Browser for SQLite ([documentation here](https://sqlitebrowser.org/) → [_](https://sqlitebrowser.org/)).

Once you're connected to the database, you can then read, write, update, and delete data from its tables. These commands are called **queries** and are written using the SQL language.

(Similar to other kinds of file permissions, you might only have the ability to perform some of these actions but not others, e.g. read information from the database but not delete information.)

To retrieve data from one or more tables you usually use a `SELECT` statement in your query.

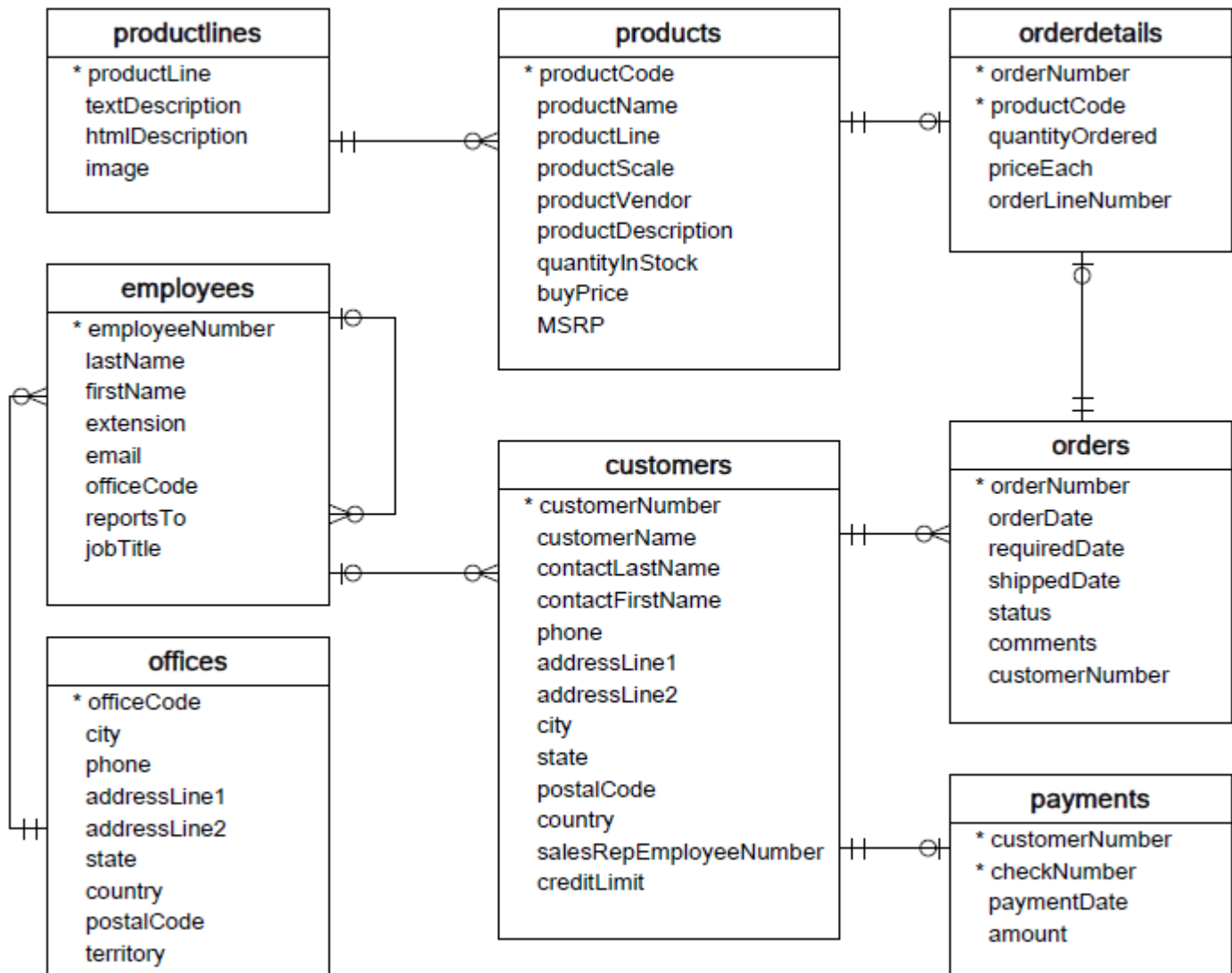
A simple query would look something like this:

```
SELECT col1, col2, col3  
FROM table  
WHERE records match criteria  
LIMIT 100;
```

Don't worry if this is confusing now, you'll soon learn what each line does. For now, just notice that:

- Queries start with the `SELECT` clause, followed by what you want to select. If selecting multiple columns, you separate them with a comma.
- Then you specify where that data is being retrieved from the using the `FROM` clause followed by the table name.
- Afterward, you can provide conditions such as filters or limits on the amount of data returned.

Also unlike a CSV, a SQL database can contain multiple related SQL tables. To demonstrate, here is an outline of a database structure:



This kind of diagram is called an **entity relationship diagram (ERD)** because it shows the relationships between tables. It does not give us any information about the specific data stored in the database, but rather the metadata.

In the diagram, each rectangle is a table, with the table name listed at the top. In this case, we have 8 tables:

1. **productlines**
2. **products**
3. **orderdetails**
4. **employees**
5. **offices**
6. **customers**

7

Ⓢ **Help**

8

Below each of the table names, we have a list of the various column names associated with that table. So for example, the `productlines` table has four columns: `productLine` , `textDescription` , `htmlDescription` and `image` .

In addition to the names of the tables and their column names, we have an indication of relationships. For example, data in the `employees` table has some relationship to data on the `offices` table, indicating that an employee may be associated with a specific office location. Likewise, certain orders are associated with certain customers. Lots of real world data is inherently related. For example, students have an association to a course, or ingredients are related to a recipe.

Primary Keys and Foreign Keys


You may also note that some of these column names are preceded by an asterisk (`*`). This indicates that this is the **primary key** for the table. A primary key is a unique identifier for a table. That is, there can only be unique values for this column entry. `lastName` would not be a good choice for a primary key as it's common for people to have the same last names or even `firstName + lastName` pairings. For this reason it is typical for a primary key to have a name reflecting some kind of "number", "code", or "id" — something that is truly unique to that record, which may or may not have any meaning beyond the database itself.

If you look closely, you'll see that the columns that are the primary key for one table can also appear on other tables. This is known as a **foreign key** aka the primary key from a different ("foreign") table. This is the core idea of how data on different tables is associated in a relational database. If you were told a specific `customerNumber` , and then given a list of order data that included the `customerNumber` , you could determine which orders were placed by that customer by matching up the primary and foreign keys.

The lines, circles, arrows, and tick marks are showing different categorizations on exactly how this data is linked.

Technical Interview Practice

Data retrieval is the most foundational and arguably most important skill in a data scientist's toolbox. You can know all the Python and machine learning and statistics in the world, but they're useless if you don't have data to aim those skills at!

Especially for junior data scientists, employers tend to evaluate candidates' data retrieval abilities by asking SQL questions. Keep this in mind as you go through our SQL content, and consider getting additional practice with platforms such as [HackerRank](https://www.hackerrank.com/domains/sql/)  [\(https://www.hackerrank.com/domains/sql/\)](https://www.hackerrank.com/domains/sql/) to keep your skills up.

In this lesson, you got a quick overview of what SQL is and saw an example SQL **SELECT** query and ERD! Remember that there are multiple SQL dialects all with particular differences, but the overall language is generally fairly similar and interchangeable. You also learned that knowledge of SQL is important for job interviews since data retrieval is a foundational part of the data science process.

How do you feel about this lesson?



Have specific feedback?

[Tell us here! \(https://github.com/learn-co-curriculum/dsc-getting-started-sql-intro/issues/new/choose\)](https://github.com/learn-co-curriculum/dsc-getting-started-sql-intro/issues/new/choose)