LEARN EVERYTHING AI

SQL GUIDE

SQL Crash Course

FOR MORE DETAILS VISIT:
www.learneverythingai.com

Land your dream data science job with "Learn Everything AI" Data Science program. Also enhance your resume, prepare for techincal interview and learn career growth hack from industry experts.

FOR MORE DETAILS VISIT:
www.learneverythingai.com

Chapter 1. SQL Crash Course

This short chapter is intended to quickly get you up to speed on basic SQL terminology and concepts.

What Is a Database?

Let's start with the basics. A *database* is a place to store data in an organized way. There are many ways to organize data, and as a result, there are many databases to choose from. The two categories that databases fall into are *SQL* and *NoSQL*.

SQL

SQL is short for *Structured Query Language*. Imagine you have an app that remembers all of your friend's birthdays. SQL is the most popular language you would use to talk to that app.

> *English: "Hey app. When is my husband's birthday?"*
> *SQL:* `SELECT * FROM birthdays WHERE person = 'husband';`

SQL databases are often called *relational databases* because they are made up of relations, which are more commonly referred to as tables. Many tables connected to each other make up a database. Figure 1-1 shows a picture of a relation in a SQL database.

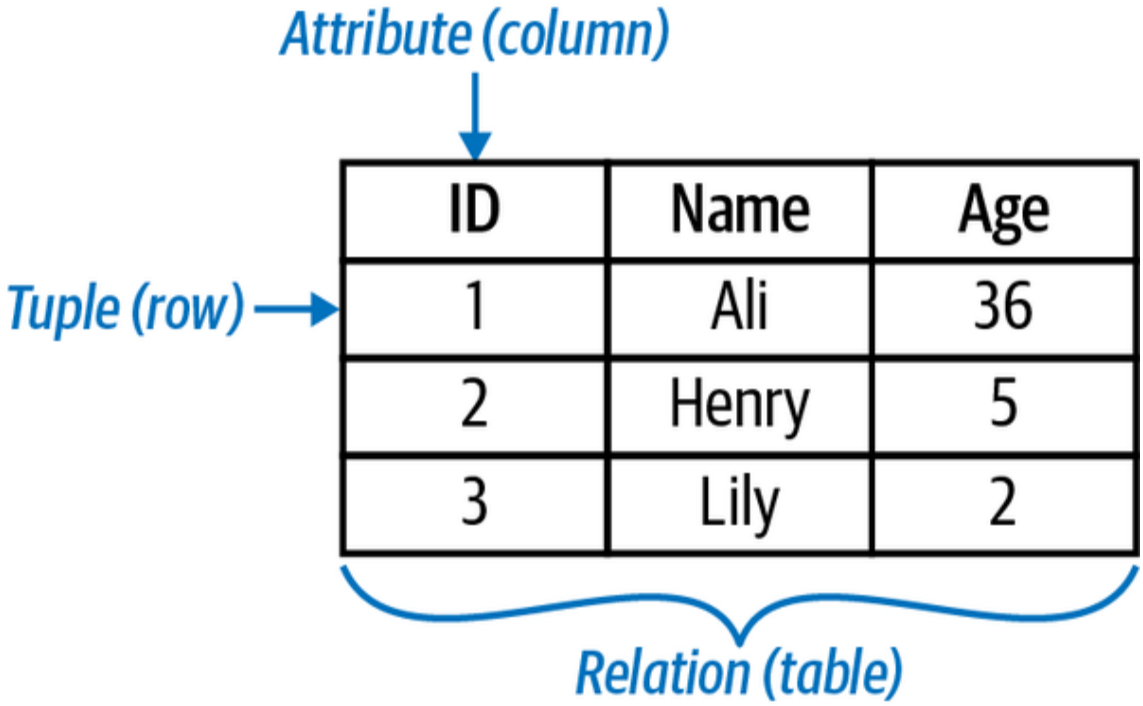

Figure 1-1. A relation (also known as a table) in a SQL database

The main thing to note about SQL databases is that they require predefined *schemas*. You can think of a schema as the way that data in a database is organized or structured. Let's say you'd like to create a table. Before loading any data into the table, the structure of the table must first be decided on, including things like what columns are in the table, whether those columns hold integer or decimal values, etc.

There comes a time, though, when data cannot be organized in such a structured way. Your data may have varying fields or you may need a more effective way of storing and accessing a large amount of data. That's where NoSQL comes in.

NoSQL

NoSQL stands for *not only SQL*. It will not be covered in detail in this book, but I wanted to point it out because the term has grown a lot in popularity since the 2010s and it's important to understand there are ways to store data beyond just tables.

NoSQL databases are often referred to as *non-relational databases*, and they come in all shapes and sizes. Their main characteristics are that they have dynamic schemas (meaning the schema doesn't have to be locked in up front) and they allow for horizontal scaling (meaning the data can spread across multiple machines).

The most popular NoSQL database is *MongoDB*, which is more specifically a document database. Figure 1-2 shows a picture of how data is stored in MongoDB. You'll notice that the data is no longer in a structured table and the number of fields (similar to a column) varies for each document (similar to a row).

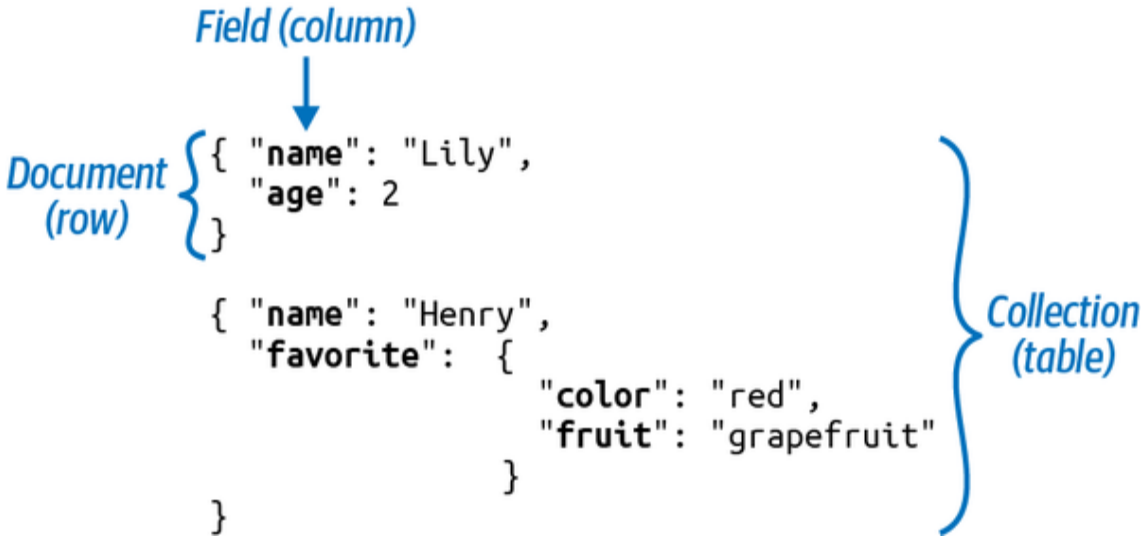

Figure 1-2. A collection (a variant of a table) in MongoDB, a NoSQL database

That all said, the focus of this book is on SQL databases. Even with the introduction of NoSQL, most companies still

store the majority of their data in tables in relational databases.

Database Management Systems (DBMS)

You may have heard terms like *PostgreSQL* or *SQLite*, and be wondering how they are different from SQL. They are two types of *Database Management Systems* (DBMS), which is software used to work with a database.

This includes things like figuring out how to import data and organize it, as well as things like managing how users or other programs access the data. A *Relational Database Management System* (RDBMS) is software that is specifically for relational databases, or databases made up of tables.

Each RDBMS has a different implementation of SQL, meaning that the syntax varies slightly from software to software. For example, this is how you would output 10 rows of data in 5 different RDBMSs:

MySQL, PostgreSQL, and SQLite

```
SELECT * FROM birthdays LIMIT 10;
```

Microsoft SQL Server

```
SELECT TOP 10 * FROM birthdays;
```

Oracle Database

```
SELECT * FROM birthdays WHERE ROWNUM <= 10;
```

GOOGLING SQL SYNTAX

When searching for SQL syntax online, always include the RDBMS you are working with in the search. When I first learned SQL, I could not for the life of me figure out why my copy-pasted code from the internet didn't work and this was the reason!

Do this.

Search: *create table datetime* ***postgresql***

→ Result: `timestamp`

Search: *create table datetime* ***microsoft sql server***

→ Result: `datetime`

Not this.

Search: *create table datetime*

→ Result: syntax could be for any RDBMS

This book covers SQL basics along with the nuances of five popular database management systems: Microsoft SQL Server, MySQL, Oracle Database, PostgreSQL and SQLite.

Some are proprietary, meaning they are owned by a company and cost money to use, and others are open source, meaning they are free for anyone to use. Table 1-1 details the differences between the RDBMSs.

Table 1-1. RDBMS comparison table

RDBMS	Owner	Highlights

RDBMS	Owner	Highlights
Microsoft SQL Server	Microsoft	- Popular proprietary RDBMS - Often used alongside other Microsoft products including Microsoft Azure and the .NET framework - Common on the Windows platform - Also referred to as *MSSQL* or *SQL Server*
MySQL	Open Source	- Popular open source RDBMS - Often used alongside web development languages like HTML/CSS/Javascript - Acquired by Oracle, though still open source
Oracle Database	Oracle	- Popular proprietary RDBMS - Often used at large corporations given the amount of features, tools, and support available - Also referred to simply as *Oracle*
PostgreSQL	Open Source	- Quickly growing in popularity - Often used alongside open source technologies like Docker and Kubernetes - Efficient and great for large datasets

RDBMS	Owner	Highlights
SQLite	Open Source	- World’s most used database engine - Common on iOS and Android platforms - Lightweight and great for a small database

NOTE

Going forward in this book:

- Microsoft SQL Server will be referred to as *SQL Server*.
- Oracle Database will be referred to as *Oracle*.

Installation instructions and code snippets for each RDBMS can be found in RDBMS Software in Chapter 2.

A SQL Query

A common acronym in the SQL world is *CRUD*, which stands for “Create, Read, Update, and Delete.” These are the four major operations that are available within a database.

SQL Statements

People who have *read and write access* to a database are able to perform all four operations. They can create and delete tables, update data in tables, and read data from tables. In other words, they have all the power.

They write *SQL statements*, which is general SQL code that can be written to perform any of the CRUD operations. These people often have titles like *database administrator* (DBA) or *database engineer*.

SQL Queries

People who have *read access* to a database are only able to perform the read operation, meaning they can look at data in tables.

They write *SQL queries*, which are a more specific type of SQL statement. Queries are used for finding and displaying data, otherwise known as "reading" data. This action is sometimes referred to as *querying tables*. These people often have titles like *data analyst* or *data scientist*.

The next two sections are a quick-start guide for writing SQL queries, since it is the most common type of SQL code that you'll see. More details on creating and updating tables can be found in Chapter 5.

The SELECT Statement

The most basic SQL query (which will work in any SQL software) is:

```
SELECT * FROM my_table;
```

which says, show me all of the data within the table named `my_table`—all of the columns and all of the rows.

While SQL is case-insensitive (`SELECT` and `select` are equivalent), you'll notice that some words are in all caps and others are not.

- The uppercase words in the query are called *keywords*, meaning that SQL has reserved them to perform some

sort of operation on the data.

- All other words are lowercase. This includes table names, column names, etc.

The uppercase and lowercase formats are not enforced, but it is a good style convention to follow for readability's sake.

Let's go back to this query:

```
SELECT * FROM my_table;
```

Let's say that instead of returning all of the data in its current state, I want to:

- Filter the data
- Sort the data

This is where you would modify the `SELECT` statement to include a few more *clauses*, and the result would look something like this:

```
SELECT *
FROM my_table
WHERE column1 > 100
ORDER BY column2;
```

More details on all of the clauses can be found in Chapter 4, but the main thing to note is this: the clauses must always be listed in the same order.

MEMORIZE THIS ORDER

All SQL queries will contain some combination of these clauses. If you remember nothing else, remember this order!

```
SELECT        -- columns to display
FROM          -- table(s) to pull from
WHERE         -- filter rows
GROUP BY      -- split rows into groups
HAVING        -- filter grouped rows
ORDER BY      -- columns to sort
```

NOTE

The `--` is the start of a comment in SQL, meaning the text after it is just for documentation's sake and the code will not be executed.

For the most part, the `SELECT` and `FROM` clauses are required and all other clauses are optional. The exception is if you are selecting a particular database function, then only the `SELECT` is required.

The classic mnemonic to remember the order of the clauses is:

Sweaty feet will give horrible odors.

If you don't want to think about sweaty feet each time you write a query, here's one that I made up:

Start Fridays with grandma's homemade oatmeal.

Order of Execution

The order that SQL code is executed is not something typically taught in a beginner SQL course, but I'm including it here because it's a common question I received when I taught SQL to students coming from a Python coding background.

A sensible assumption would be that the order that you *write* the clauses is the same order that the computer *executes* the clauses, but that is not the case. After a query is run, this is the order that the computer works through the data:

1. `FROM`
2. `WHERE`
3. `GROUP BY`
4. `HAVING`
5. `SELECT`
6. `ORDER BY`

Compared to the order that you actually write the clauses, you'll notice that the `SELECT` has been moved to the fifth position. The high-level takeaway here is that SQL works in this order:

1. Gathers all of the data with the `FROM`
2. Filters rows of data with the `WHERE`
3. Groups rows together with the `GROUP BY`
4. Filters grouped rows with the `HAVING`
5. Specifies columns to display with the `SELECT`
6. Rearranges the results with the `ORDER BY`

A Data Model

I'd like to spend the final section of the crash course going over a simple *data model* and point out some terms that you'll often hear in fun SQL conversations around the office.

A data model is a visualization that summarizes how all of the tables in a database are related to one another, along with some details about each table. Figure 1-3 is a simple data model of a student grades database.

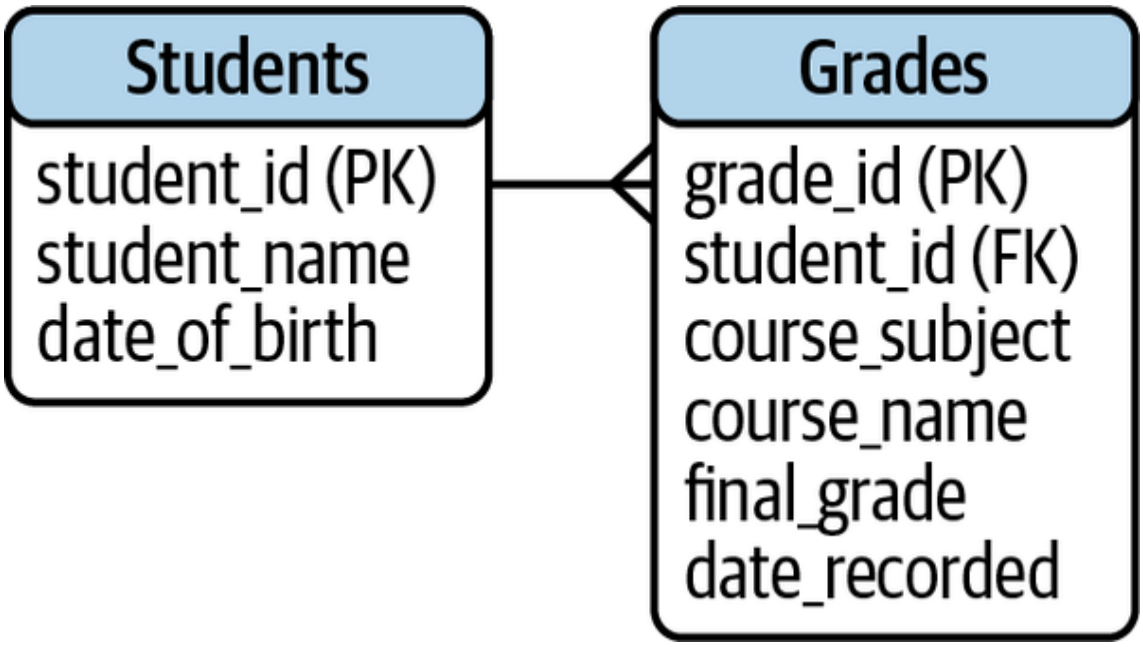

Figure 1-3. A data model of student grades

Table 1-2 lists out the technical terms that describe what's happening in the data model.

Table 1-2. Terms used to describe what's in a data model

Term	Definition	Example

Term	Definition	Example
Database	A database is a place to store data in an organized way.	This data model shows all of the data in the student grades database.
Table	A table is made up of rows and columns. In the data model, they are represented by rectangles.	There are two tables in the student grades database: `Students` and `Grades`.
Column	A table consists of multiple columns, which are sometimes referred to as attributes or fields. Each column contains a particular type of data. In the data model, all of the columns in a table are listed within each rectangle.	In the `Students` table, the columns are `student_id`, `student_name`, and `date_of_birth`.
Primary Key	A *primary key* uniquely identifies each row of data in a table. A primary key can be made up of one or more columns in a table. In a data model, it is flagged as `pk` or with a key icon.	In the `Students` table, the primary key is the `student_id` column, meaning that the `student_id` value is different for each row of data.

Term	Definition	Example
Foreign Key	A *foreign key* in a table refers to a primary key in another table. The two tables can be linked together by the common column. A table can have multiple foreign keys. In a data model, it is flagged as `fk`.	In the `Grades` table, `student_id` is a foreign key, meaning that the values in that column match up with values in the corresponding primary key column in the `Students` table.
Relationship	A *relationship* describes how the rows in one table map to the rows in another table. In a data model, it is represented by a line with symbols at the end points. Common types are one-to-one and one-to-many relationships.	In this data model, the two tables have a one-to-many relationship represented by the fork. One student can have many grades, or one row of the `Students` table maps to multiple rows in the `Grades` table.

More details on these terms can be found in “Creating Tables” in Chapter 5.

You might be wondering why we’re spending so much time reading a data model instead of writing SQL code already! The reason is because you’ll often be writing queries that link up a number of tables, so it’s a good idea to first get familiar with the data model to know how they all connect.

Data models can typically be found in a documentation repository at a company. You may want to print out the data models that you frequently work with—both for easy reference and easy desk decor.

You can also write queries within an RDBMS to look up information contained in a data model, such as the tables in a database, the columns of a table, or the constraints of a table.

AND THAT IS YOUR CRASH COURSE!

The remainder of this book is intended to be a reference book and does not need to be read in order. Please use it to look up concepts, keywords, and standards.