**How to Take Your SQL from Zero to Data Scientist Level — Part 1/3**

**Set up SQL and execute your first SELECT query**



Photo by [Dominik Martin](https://unsplash.com/@dominikmartin?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/step?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

Even though **SQL** is more than 50 years old, it is still the primary language used in most data science teams to retrieve (big) data from a database. Looking at the [Stack Overflow 2022 Developer Survey](https://survey.stackoverflow.co/2022/#most-popular-technologies-language-prof), there were 3,424 professionals with a current job as a data scientist or machine learning specialist. Of them, 58% have worked with SQL in the past year. For comparison, 70% have worked with R and 87% with Python, with 40% using all three in the past year. In other words, 6 out of 10 data science teams will expect you to use SQL in your role.

In this three-part series, I have created a guide for aspiring data scientists that want to learn SQL at the standards of a good data science team. The guide will be structured as follows:

**Part 1:**

* Introduction to SQL
* Install SQL and load your first demo database
* Run your first SELECT query!

**Coming Soon:**

* **Part 2:** Learn all the basic and advanced SQL queries used within a data science team and practice them using the demo database
* **Part 3:** Query optimization, data science coding best practices and plan for training

**1. Introduction to SQL**



Photo by [Dose Media](https://unsplash.com/@dose?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/study?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

**1.1. What is SQL?**

SQL stands for Structured Query Language and is used for manipulating data in a relational database (often you will hear the term “I wrote a query” referring to SQL code)

**1.2. What is a relational database?**

At its core, a relational database is simply a collection of tables (in computer science tables are called **relations**, hence the name relational database). There are other database objects, but as a data scientist, more than 90% of your time will be devoted to tables.

* Within a table, each row must be unique. In other words, the combination of column values for a given row cannot be found in any other row. For that purpose, we usually create a surrogate column that will be unique by design for each table row. This column is called the **Primary Key** of a table
* Within a table, each cell must contain only one value. So you cannot have within a cell the first name and address of a customer. These must be split into different columns

**1.3. What are some examples of tables in a relational database?**

From my experience working as a senior analyst for an e-commerce/retail consultancy and as a data scientist for mobile games, you will find three main tables in almost any relational database.

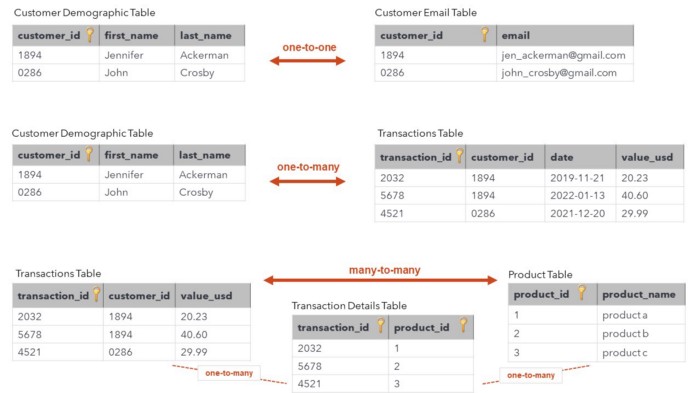
* **Customer** or **User** table (customer\_id, first name, last name, account created date, address, etc.)
* **Transactions** table (transaction\_id, purchase timestamp, transaction value, etc.)
* **Product** table (product\_id, product name, product department, etc.)

**1.4. What types of SQL statements will I write as a data scientist?**

As a data scientist, you will mainly employ SQL to extract data from the database (using the **SELECT** statement). The data will then be loaded in R, Python, or Excel to perform your analysis. Usually, you would want to combine information from multiple tables. So you might want to find the customers that purchased product A in the last year. Sometimes, although rarely, you might alter or create tables in the database (like a table with the scores of your model).

**1.5. How are tables linked with each other?**

An essential concept of any relational database is how the tables are associated with each other. There are three types of associations or **relationships**, as shown below (the primary key columns have a key icon next to them). In order to link the tables, we use the shared columns.



Types of relationships in a relational database [Image by the author]

**One-to-one:**

* The ***customer demographic*** and the ***customer email*** tables have a one-to-one relationship as they are both unique at the customer\_id column
* The shared **customer\_id** column will be used to link the tables

**One-to-many:**

* The ***customer demographic*** and the ***transactions*** tables have a one-to-many relationship as a customer (customer\_id) could have made more than one transaction (transaction\_id), but each transaction can only be made by a single customer
* The shared **customer\_id** column will be used to link the tables

**Many-to-many:**

* The **transactions** and the **product** tables have a many-to-many relationship as every transaction can have more than one product, and every product can be in more than one transaction
* Relational databases only support one-to-many relationships directly. A many-to-many relationship is implemented by adding a third table between the first and second table to create two one-to-many relationships. The **transaction details** table was added to that purpose (the primary key is the combination of the transaction\_id and product\_id column)

**2. Install SQL and run your first SELECT query!**



Photo by [Amauri Mejía](https://unsplash.com/@amavcinema?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/create?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

So far, we have covered the basics of relational databases and SQL. Next, it will be useful to see how all that comes together with the use of RDBMSs (Relational Database Management Systems). There are many different **RDBMSs**,such as PostgreSQL, SQLite, MongoDB, MySQL, and Microsoft SQL Server (or just SQL Server). On an interesting note, even though the relational model and SQL were invented at IBM, other companies immediately identified it as a game changer leading to Oracle beating them by delivering their RDBMS in the market first. But even though there are a lot, all RDBMSs have a singular main point of value, they **provide an environment that supports both SQL and relational databases.**

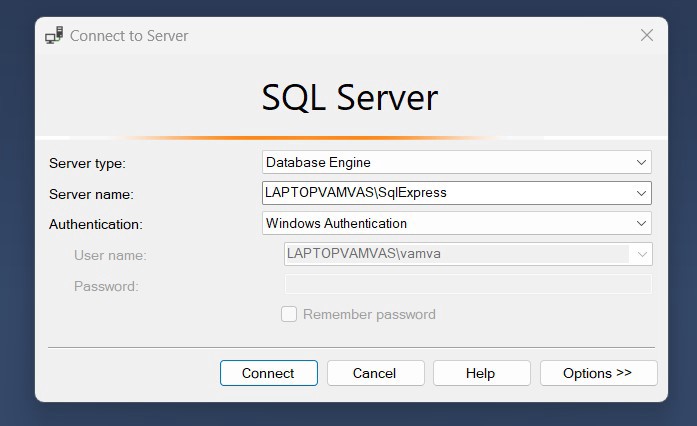
For the purposes of this guide, we will go with SQL Server. To visualize the database and tables and write SQL queries, we will use **SSMS** (SQL Server Management Studio). Similar to the way you use RStudio for R or Spyder for Python, you have SSMS for SQL Server.

**2.1 Install SQL Server (Express Edition)**

* Go to [Microsoft’s download page](https://www.microsoft.com/en-gb/sql-server/sql-server-downloads) and download **SQL Server Express**
* Once downloaded, open the SQL Server Express installer
* Select **Basic** as the installation type
* Select the installation path (I recommend leaving it to the default)
* Follow the installer’s instructions until you finish the installation

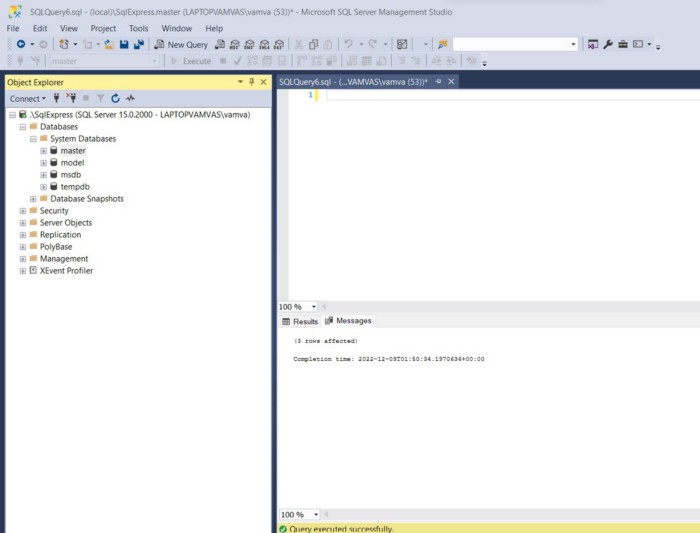
**2.2 Install SSMS**

* Once the SQL Server Express Edition installation finishes, it will prompt you to install SSMS
* Follow the default steps until finished
* Open SSMS and wait until the “Connect to Server” pop-up window appears
* Fill in the pop-up window as in the snapshot below
* if the server name **.\SqlExpress** doesn’t work, use **device name\SqlExpress** (a google query of how to find the device name of your computer in your operating system will give you the result. For example, in my case, it is **LAPTOPVAMVAS\SqlExpress**)



SSMS Connect to Server [Image by the author]

Once finished, you should see the three panels in SSMS, as shown below. On the left panel, you can see the Object Explorer. We will be connected to a local instance of SQL Server, which is the computer acting as a server (this could also be a cloud server or a physical server if one exists). You can also see the four system databases created in the server by default (with installation). On the top right panel is the area where you will write and execute your SQL queries, and on the bottom right panel, you can also see the results of your queries.

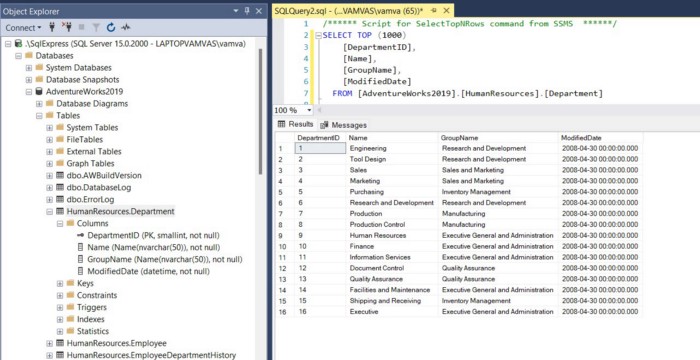


SSMS snapshot after installation [Image by the author]

**2.3 Load AdventureWorks demo database (last step!)**

* Download the AdventureWorks **.bak** file from one of the links provided in the [download backup files section](https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms#download-backup-files) (I recommend selecting the 2019 OLTP file)
* Move the **.bak** file to your SQL Server backup location. This varies depending on your installation location, instance name, and version of SQL Server. For example, the default location for a default instance of SQL Server 2019 is *“C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Backup”*
* In SSMS, right-click **Databases** in Object Explorer and **Restore Database** to launch the Restore Database wizard
* Select **Device** and then select the ellipses (…) to choose a device.
* Select **Add** and then choose the .bak file you recently moved to the backup location
* Select **OK** to confirm your database backup selection and close the Select backup devices window
* Select **OK** to restore your database
* Visit this [page](https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms) from Microsoft for the latest step-by-step guide

And with that last step, you are now the proud owner of a demo relational database hosted on your computer (see the image below). This is a significant milestone as now you have an asset created — the ability to practice SQL queries with actual data anytime you want!



SSMS with AdventureWorks demo database loaded [Image by the author]

On the left panel is our demo database (**AdventureWorks2019**). As you can see, I have expanded it to view all the tables within the database, and I have also expanded the **HumanResources.Department** table to examine the columns within it (I can also see the key next to the **DepartmentID** column indicating it is a primary key for that table).

I also wrote a simple **SELECT** statement on the top right panel of SSMS (as seen in the code chunk below) to extract the top 1,000 rows from the HumanResources.Department table. On the bottom right panel, you can see the results of that query. The table has 16 rows and 4 columns.

SELECT TOP (1000)   
 [DepartmentID],  
 [Name],  
 [GroupName],  
 [ModifiedDate]  
FROM [AdventureWorks2019].[HumanResources].[Department]  
  
--- or use the simpler  
  
SELECT \*  
FROM [AdventureWorks2019].[HumanResources].[Department]

In the next article of this series (coming soon), we will dive in and learn all the basic and advanced SQL queries used within a data science team and practice them using SSMS and our demo database.

I also recommend the following exercises to explore some of the features in SSMS and your newly loaded database:

* Create a new query window by selecting **“New Query”** from the toolbar and changing the database from master to **AdventureWorks2019** (bottom left corner of the toolbar)
* Write a query that selects the CustomerID and TerritoryID from the Sales.Customer table and execute it
* Use the Object Explorer (left panel) to navigate the different tables, columns, and primary keys in the database
* Save your SQL query in your computer (as a .sql file) and load it using SSMS
* Save the results of your query (by right-clicking the results tab of the bottom right panel) in your computer as a .csv file

I hope you found this article helpful and have fun querying!