SQL for Data Science

Author, Mayank Tripathi

A Data Science Foundation White Paper

March 2020

---

www.datascience.foundation

Data Science Foundation
Data Science Foundation, Atlantic Business Centre, Atlantic Street, Altrincham, WA14 5NQ
Tel: 0161 926 3641 Email: admin@datascience.foundation Web: www.datascience.foundation
Registered in England and Wales 4th June 2015, Registered Number 9624670

SQL For Data Science / Machine Learning

SQL is one of the most requested skills in Data Science. Let's find out how it can be used in Data Processing and Machine Learning. Today I'd like to talk about why SQL plays such an important role in data science and discuss some of its uses. SQL or the Structured Query Language is a set of instructions used to interact with a relational database. The interaction could be for managing / saving / capturing the data or retrieving (querying) the data. Relational Databases are formed by collections of two-dimensional tables (e.g. Datasets, Excel Spreadsheets). Each of these tables is then formed by a fixed number of columns and any possible number of rows. Although it is not the only language out there, SQL is the only language that most relational databases understand.

I am planning to write another article on various databases and will update you on this soon.

Whenever you interact with such a relational database, the software translates your commands into an SQL statement that the database knows how to interpret. Why is this important? Well most of the data stored by organizations is within relational databases and SQL is the language you use to pull or query data from the databases.

Role of SQL in Data Science

The primary use of SQL is in the early phase of your data analysis;

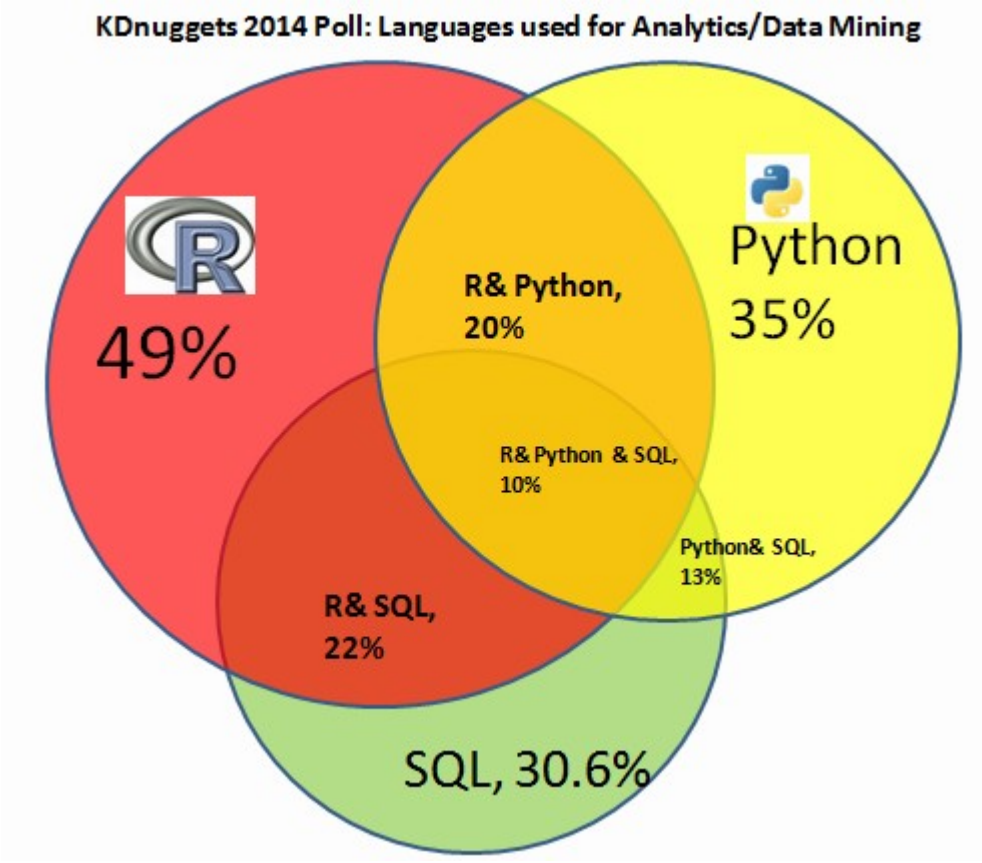
- Get a data set and start to investigate the data,
- Visualize it,
- Identify missing values,
- Incorrect formatting,
- etc.

SQL allows you to play around with data sets so you can perform these actions. You may ask me why I should use SQL when I can perform all these steps in Python or R or any other programming languages. SQL is just another tool in your data scientist toolbox. But as a Data Scientist, one must be aware of more than one tool. Generally, if you work in a company that stores its data in SQL relational databases, you use SQL to fetch your data, so that you can model / describe / visualize it. Companies are now giving more importance to the value of data and generating millions - trillions of bits of data every day. In order to store such large amounts of data, it is strictly necessary to make use of databases.

Data can, for example, be used to: analyze and solve business problems, make predictions on

market trends and understand customer needs. If you are working with real data (meaning not clean and structured data that you normally get during courses or sample projects, you know that clean data is not the norm. Usually, before you can begin your data analysis process, you need to go through a data munging or wrangling process – this means you need to get the data into a format that you can use for your analysis. It is said that data scientists spend about 80% of their time is spent doing just that! SQL is there to help with this process. One of the main advantages of using SQL, is that when performing operations with data, this is accessed directly (without any need to copy it beforehand). This can considerably speed up workflow executions.

Also refer to below image, though I copied this from KDnuggets.com, the image is from 2014, which say's SQL was uses 30+%, but for sure the percent either remain same or would have increased up. From here you can compare how important is SQL.



Fortunately for those of us who aren't computer programmers yet or don't plan to be, SQL commands are designed to have a syntax like the English language. They normally begin with a command statement describing the action to take, followed by a clause that describes the target of the command (this could be the specific table within a database affected by the command) and finally, a series of clauses that provide additional instructions.

Most of the time, you can read an SQL statement and understand what the command is trying to do.

Just to show it on board.

```
SELECT ID, FIRST_NAME, LAST_NAME
FROM STUDENTS
WHERE LAST_NAME = 'TRIPATHI'
```

Can you guess what this statement will do?

It accesses the students table of the database and Select all records for students whose last name matches to the value given. See? It wasn't that difficult!

Let's talk about some of the potential uses…

- Data Aggregations → Aggregation functions are very useful for understanding the data and presenting it in summary. The widely used aggregate functions are min, max, average, etc.
- Statistical Functions → Functions like distribution fitting can be directly applied on data by using SQL functions. Few are as below.
    - COUNT
    - SUM
    - AVG
    - STDEV
    - STDEVP
    - VAR
    - VARP
- Ranking Functions → Ranking functions are useful to rank values in a data set and doing a Top-N analysis. For example, ranking functions can be used to rank customers within a segment.
    - ROW_NUMBER: Assigns a sequential number to each row in the result set.
    - RANK: Ranks each row in the result set. …
    - DENSE_RANK: Ranks each row in the result set. …
    - NTILE: Divides the result set into the number of groups specified as an argument to the function.
- Grouping the Data → Sometimes, we need to group the data for producing better predictions or results. For example, we can group employees ages into five distinct sets / range to analyze their common traits across each age set / range.

- Windowing Functions → Windowing functions are very useful for any aggregate calculations that involve a range of values or a group of rows. For example, these functions come handy in case of operations over time series data which includes calculations over a fixed or variable window time period.
- Joins → Used to join or club multiple data-set and club it onto a single data-set based on various conditions (if required).

Note: Function may differ based on database server from which SQL is triggered. Some of the frequently used are Microsoft SQL Server; Oracle; MySQL; Postgres SQL; DB2 etc.

The best part of SQL is that one can connect to SQL using Python as well. You must use specific drivers and start using it.

As mentioned above, there are many different SQL databases such as: SQLite, MySQL, Postgres, Oracle and Microsoft SQL Server, Postgres SQL etc. for all these we need some kind of initial setup and requires installation and specific software / hardware requirement.

To get rid of this hassle, Google has provided BigQuery ML. I am also new to this and trying to learn more about it. Once I am more familiar of it will write a brief article on it.

But in-short**BigQuery ML enables users to create and execute machine learning models in BigQuery using standard SQL queries. BigQuery ML democratizes machine learning by enabling SQL practitioners to build models using existing SQL tools and skills. BigQuery ML increases development speed by eliminating the need to move data.**

Refer https://cloud.google.com/bigquery-ml/docs/bigqueryml-intro

You can also refer to one of these excellent articles from KDnuggets https://www.kdnuggets.com/2019/05/7-steps-mastering-sql-data-science-2019-edition.html From here you will have more details of SQL.

Refer to https://www.tutorialspoint.com/sql/index.htm to get started with SQL. For Machine Learning or Data Science, you may not need to learn the entire SQL programming language, instead having a basic knowledge will be good enough.

You can reach out to me as well or add a comment in the article / post if have any questions, and I will help you with it.

Basics of SQL

Below are the activities one can perform with SQL.

- SQL can create new databases

- SQL can create new tables in a database
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create functions in a database --> May not be necessary for Data Science / Machine Learning. If time permits or interested then please go ahead and learn, practice this.
- SQL can create stored procedures in a database --> May not be necessary for Data Science / Machine Learning. If time permits or interested then please go ahead and learn, practice this.
- SQL can create packages in a database --> May not be necessary for Data Science / Machine Learning. If time permits or interested then please go ahead and learn, practice this.
- SQL can create views in a database --> May not be necessary for Data Science / Machine Learning. If time permits or interested then please go ahead and learn, practice this.
- SQL can set permissions on tables, procedures, and views --> May not be necessary for Data Science / Machine Learning. If time permits or interested then please go ahead and learn, practice this.

Next term is **RDBMS**, RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables.

A table is a collection of related data entries and it consists of columns and rows.

Look at the "STUDENTS" table, we saw earlier.

SELECT * FROM STUDENTS;

Each table is broken up into smaller entities called fields. The fields in the STUDENTS table consist of an ID (which either can be considered as Student ID or any sequential number for Indexing), First_Name, Last_Name etc.

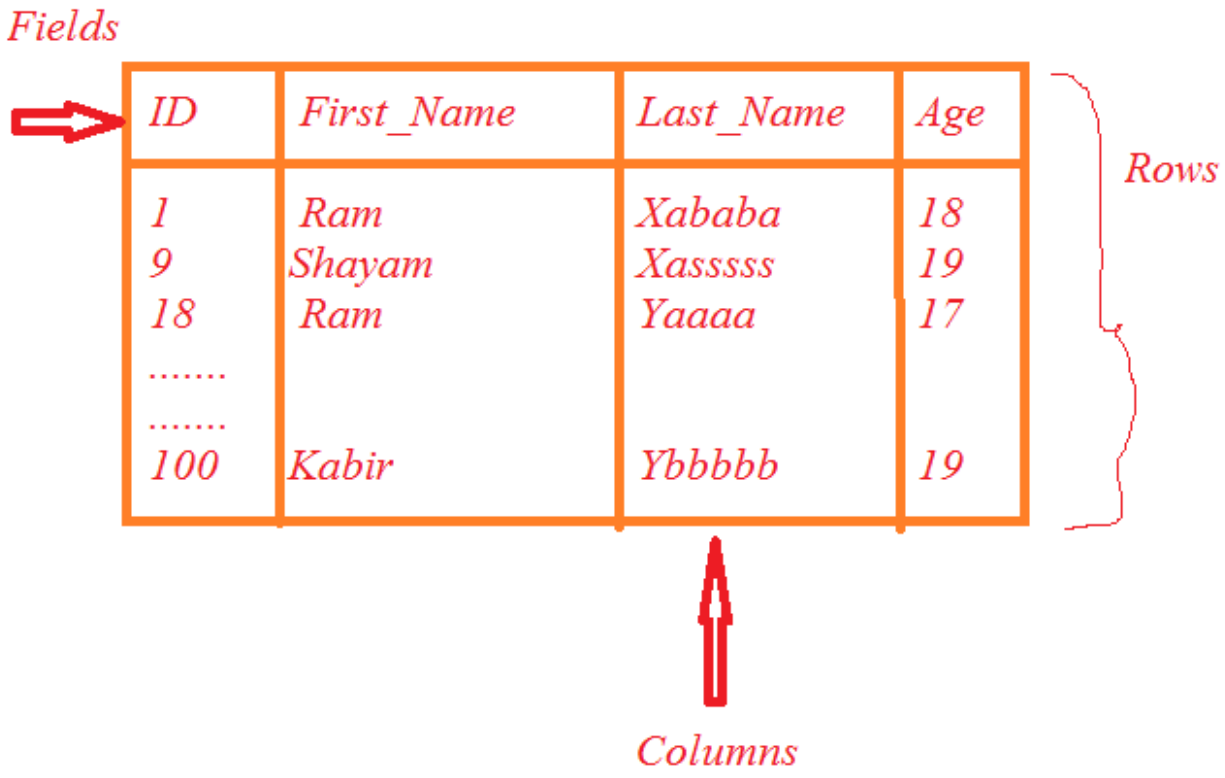
A field is a column in a table that is designed to maintain specific information about every record in the table.

A record, also called a row, is each inpidual entry that exists in a table. For example, there are 100 records in the above Students table. It means there are 100 Students for who we do have

data. A record is a horizontal entity in a table.

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

Below is the image, entire data is saved in Table.



A database most often contains one or more table. Each table is identified by a name (e.g. "Students" or "Qualification", “Certificates”, “Address”, “University”). Tables contain records (rows) with data.

The SQL SELECT Statement → The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

Syntax:

SELECT column1, column2, ... FROM TABLE_NAME

In-case we need to get all columns, then alternatively we can use asterisk sign (*).

SELECT * FROM TABLE_NAME

The SQL WHERE Clause → The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

SELECT column1, column2, ... FROM TABLE_NAME WHERE condition;

Note: The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.!

The SQL INSERT INTO Statement → The INSERT INTO statement is used to insert new records in a table.

INSERT INTO TABLE_NAME (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

Below is one of the easiest approaches to insert the data, there are various other ways to add a record / data. You will come across once you start using the SQL.

The SQL UPDATE Statement → The UPDATE statement is used to modify the existing records in a table.

UPDATE TABLE_NAME

SET column1 = value1, column2 = value2, ...

WHERE condition;

The SQL DELETE Statement → The DELETE statement is used to delete existing records in a table.

DELETE FROM TABLE_NAME WHERE condition;

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

The SQL SELECT TOP Clause → The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

Note: Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses ROWNUM.

MS SQL Server Syntax:

SELECT TOP number|percent column_name(s)

FROM TABLE_NAME

WHERE condition;

Syntax for MySQL:

SELECT column_name(s)

FROM TABLE_NAME

WHERE condition

LIMIT number;

Syntax for Oracle:

SELECT column_name(s)

FROM TABLE_NAME

WHERE ROWNUM <= number;

The SQL MIN() and MAX() Functions → The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

Syntax for MIN()

SELECT MIN(column_name)

FROM TABLE_NAME

WHERE condition;

Syntax for MAX()

SELECT MAX(column_name)

FROM TABLE_NAME

WHERE condition;

The SQL SELECT TOP Clause → The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

Note: Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses ROWNUM.

Syntax for MS SQL Server:

SELECT TOP number|percent column_name(s)

FROM TABLE_NAME

WHERE condition;

Syntax for MySQL:

SELECT column_name(s)

FROM TABLE_NAME

WHERE condition

LIMIT number;

Syntax for Oracle:

SELECT column_name(s)

FROM TABLE_NAME

WHERE ROWNUM <= number;

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!.

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

Syntax for IS NULL:

SELECT column_name(s)

FROM TABLE_NAME

WHERE column_name IS NULL;

Syntax for IS NOT NULL

SELECT column_name(s)

FROM TABLE_NAME

WHERE column_name IS NOT NULL;

The SQL Aggregate functions such as COUNT(), AVG() and SUM() Functions

The COUNT() function returns the number of rows that matches a specified criteria.

The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

Syntax for COUNT()

SELECT COUNT(column_name)

FROM TABLE_NAME

WHERE condition;

Syntax for AVG()

SELECT AVG(column_name)

FROM TABLE_NAME

WHERE condition;

Syntax for SUM()

SELECT SUM(column_name)

FROM TABLE_NAME

WHERE condition;

I hope, by now you have some basic understanding and are more comfortable with SQL.

Note: Not all database systems support all the clauses or syntax. Each database has their own syntax, and they will differ slightly, so I suggest you choose any one database when you start and focus on this until you have become familiar with the way it functions. Then working on other databases will be easy.

Once you have an understanding you will realize how one syntax is differs from other.

About the Data Science Foundation

The Data Science Foundation is a professional body representing the interests of the Data Science Industry. Its membership consists of suppliers who offer a range of big data analytical and technical services and companies and individuals with an interest in the commercial advantages that can be gained from big data. The organisation aims to raise the profile of this developing industry, to educate people about the benefits of knowledge based decision making and to encourage firms to start using big data techniques.

Contact Data Science Foundation

Email:admin@datascience.foundation
Telephone: 0161 926 3641
Atlantic Business Centre
Atlantic Street
Altrincham
WA14 5NQ
web: www.datascience.foundation