A Simple Guide to SQL Programming

Kristoffer Larsen

**Table of Contents**

A. History of SQL……………………………………………1

B. Functionality of SQL……………………………………...4

C. How to Program with SQL………………………………..5

1. How to Download SQLite

2. Basic Terms

3. CREATE

4. INSERT

5.SELECT

6. DELETE & DROP

7. ALTER  
 8. Advanced Statements

9. Advanced Theory

D. SQL Injections…………………………………………….17

1. Overview
2. Uses and Current events
3. Using SQL Injection
4. Finding Vulnerabilities

E. Applications and Beyond………………………………….18

F. Resources…………………………………………….….…20

**History of SQL**

SQL was created in the 1970s by Edgar F. Codd, a software engineer. It started when he made a relational database system that evolved into SQL. Edgar Codd worked at IBM (International Business Machines) a well known software engineering business, which has produced computers used in trajectory calculations in NASA to simple calculators. In the 1970s major software companies like IBM were developing database systems, it was sort of a race to develop the first functioning modern database management system, or DBMS for short. Edgar Codd had been working on relational style database systems, a relational style database system that looks for relationships between data in a database.

There are numerous types of database styles including hierarchical which like its name suggest stores data in a node fashion like branches on a tree, network which is very similar to relational but is very complicated and looks at patterns between individual data points, object-oriented which is very similar to relational as it assigns attributes to data sets like the color of an object, and finally, the most widely used style is relational which not only stores data but use data stored in tables to find trends, patterns, and relationships inside. Edgar Codd first created Sequel, a relational style DBM. Unfortunately the name was trademarked by another company so they were forced to change the name; they chose structured query language or SQL for short. After releasing this DBMS language multiple other companies began working on their own software using the SQL language made by Edgar Codd. Microsoft made one of the most popular programs named MySQL in the 1990s. Afterwards people began making other database softwares with the most popular being MySQL, which has many features and is arguably the best for professional and business use. Almost all large companies, websites, and large online games use MySQL to store data such as passwords and usernames. Other prevalent programs are SQLite which has numerous remakes of it, such as the standard SQLite, or remakes made by companies like Mozilla, the owners of FireFox. SQLite is free and open source unlike MySQL which requires a monthly fee and SQLite is easy to download but doesn’t have the ability to fully scale out, and it isn’t recommended to be used with large traffic websites but it is a great tool for learning SQL or as a intro into SQL. Although there are numerous remakes database programs that use the language of SQL there are some that use a branch off of SQL called NoSQL which is very similar to SQL but it has more functionality and ability to scale, examples include MongoDB a very prominent and growing database program using NoSQL.

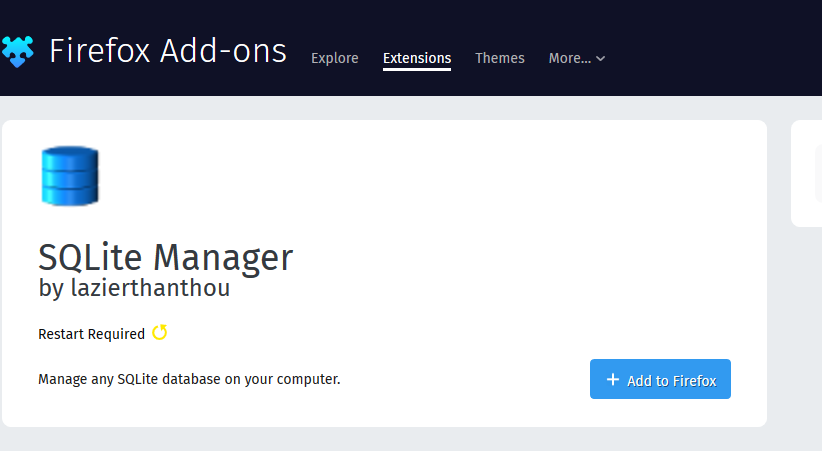
**Functionality of SQL**

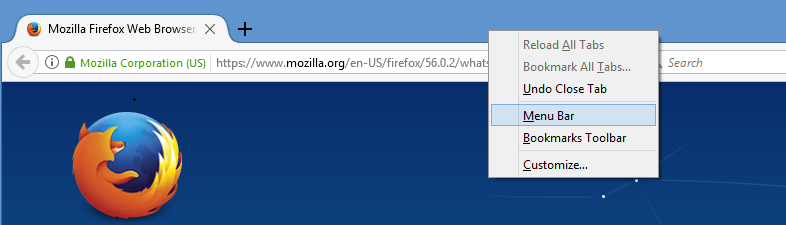
SQL is a popular database language, the language used in SQL resembles java and the syntax also matches that of java. Unlike other programming languages SQL doesn’t allow for creation of software, programs, or even video games. SQL is often used in combination with other languages like Java, PHP, Python, and HTML. SQL is used when you want to find relationships between data. SQL sorts data into tables. The first row states the type of data, for example the amount of calories, the type of cereal, the amount of sugar, or however the data is sorted. The rows after the first row are entries for data. Columns are categories of data. The most important part of SQL software is the command area or the execute SQL, this is the most important part of any SQL software as it allows you to type the SQL statements and queries. SQL is used in conjunction with Websites and games. Before I talk about how to code in the next section it is important to note that SQL is a very important tool. If you want to practice hacking, then having a background in SQL can be very useful. Also if you want to make your own website it's important to know how to use SQL.

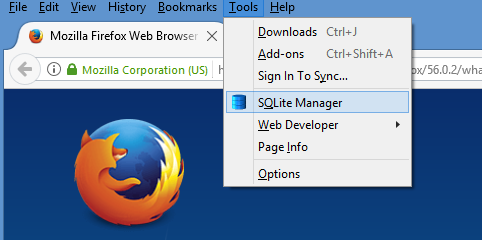
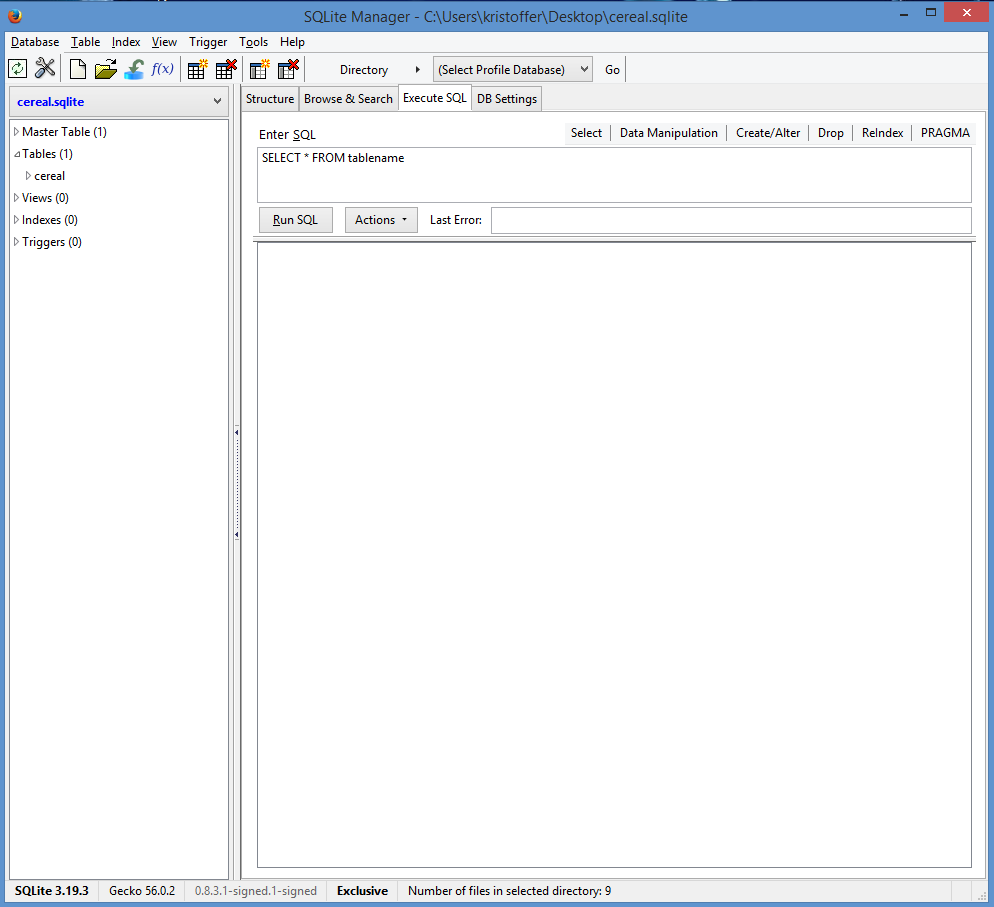
**How to Program using SQL**

The first part of programming with SQL is what program you will use. Each program has a different set up and has different purposes, such as more professional (MySQL) to a more casual (SQLite). I will be using SQLite because it’s one of the simplest yet powerful of all the programs. To download SQLite you can get it from a lot of different places. You can get SQLite from the official SQLite website, download SQLite manager from Firefox add ons, or get it from numerous other websites such as the python version. Each one is roughly the same but I will use the FireFox version as it's very simple to download and use. SQlite from the SQLite official website is a little difficult for beginners as it uses the command line (the place where you access code to talk to the computer). The FireFox SQLite manager won’t allow you to do anything really special. It can import and export datasets, create a couple of tables, and do some SQL statements.

First Open FireFox or download from your current browser. After Downloading FireFox go to the search bar and look up “FireFox SQLite Manager Addon Download”.



After downloading it and restarting FireFox, on the upper part of the browser above the search bar right anywhere and click “open Menu Bar.” 

Afterwards click on tools and click on SQLite Manager.  This will open SQLite manager. It should look something like this… 

Now that there is a DBMS that can be used, it's time to learn the basics of coding in SQL. There are seven basic terms that are a defining feature of SQL: INSERT, DELETE, DROP, SELECT, UPDATE, CREATE, and ALTER. Notice that all these terms are in capital because these terms are constant no matter what type of data you have in your database because doing this makes your code neater when you are looking for terms. Each one of them has a unique purpose when dealing with a Database INSERT adds values into a table, DELETE takeaway values in a table, DROP deletes a table, SELECT searches for queries from tables, UPDATE changes data into table, AND ALTER modifies tables or database.

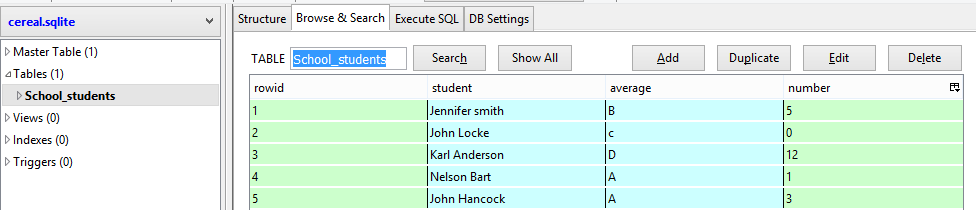
CREATE STATEMENTS

The first tool to learn is CREATE because it is the bread of SQL while SELECT is the butter. CREATE allows people to make the tables used to store data inside a database. The first thing to know about a create statement is that 1) It creates a new table 2) It defines the data that will be put into the table. There are 4 major datatypes in SQLite being: Null, when the values are empty; Integer, a number value; Text, a string of letters; Blob, value stored exactly as inputted (can be letters, numbers, or symbols). Another important statement (constraint) that defines a data type is PRIMARY KEY which basically means that the values in a column can not repeat so no duplicates. The constraint is added between the column name and its data type.

The basic form of a CREATE statement is as follows, CREATE TABLE table\_name ( column data type, column2 datatype, column3 datatype). The first part that will never change regardless of the data you use. Then comes the table’s name. It doesn't matter what the name of the table is, but if you are working with a lot of tables it makes sense to give it a meaningful name. The third part in parentheses is where you put the names of the columns and datatypes that the column follows, for example if the column is the age of customers then you should use integer. Note if you don’t use proper data types for the column it will cause problems when you use SELECT statements.

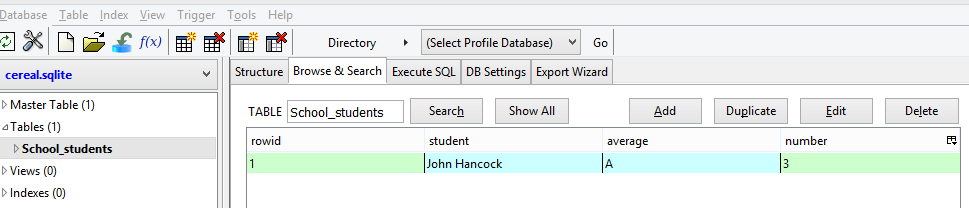
Let's make a table when its name is School\_students, where the columns are student name with datatype text, student average grade with text, and number of missed days with integer. To enter the values you click on the master table on the left and go to execute SQL.

After you find SQL you should enter in the statement CREATE TABLE School\_students (name text, average\_grade text, missed\_days integer). And on the left side of the screen you should see the table be added. You want the names of the columns to be one word so the program doesn’t confuse it with the table datatype.



INSERT STATEMENTS

INSERT statements are used to insert data points (rows) into the table. The first part of an insert statement is INSERT INTO which begins the statement. This is then followed by the table you want to insert the data into. Followed by the data in order of the column values in parentheses. If we were to insert data into the school student table it would look like this INSERT INTO School\_students VALUES ( 'John Hancock', 'A', 3) The apostrophes signal to the database that the value in between is string, a text value. And if you click on Browse & Search to the left of Execute SQL you will be able to see values in table format.



Insert these values into the table so that when we do SELECT statements there is actually data to be manipulated.

INSERT INTO School\_students VALUES ( 'Jennifer smith', 'B', 5)

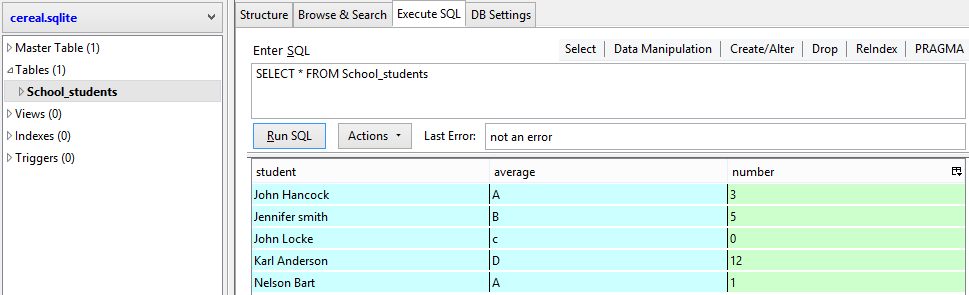
INSERT INTO School\_students VALUES ( 'John Locke', 'c', 0)

INSERT INTO School\_students VALUES ( 'Karl Anderson', 'D', 12)

INSERT INTO School\_students VALUES ( 'Nelson Bart', 'A', 1)

SELECT STATEMENTS

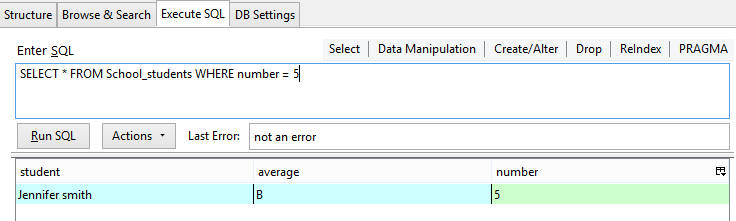
The most used and most important of all the SQL statements and the most complicated is SELECT. It allows the user to find relationships between data by adding constraints. SELECT statements have a basic from which goes as follow SELECT \* FROM table\_name. The astrexic means after the query is executed it will display all the columns returned in the selected table but you can easily add the names of columns you want to the statement. The FROM tells what table or tables the data is coming from. If you add School\_students into the FROM section it will get the school student table from Browse & Search and put it into the Execute SQL tab.



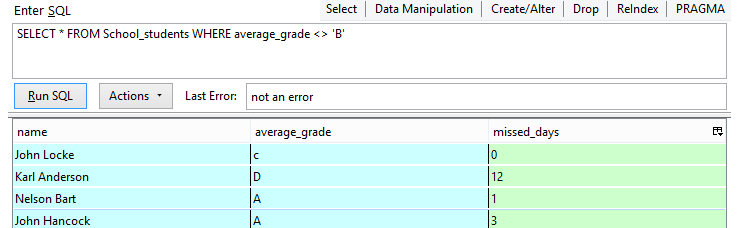
Notice that this is very basic and no constraints have been put on the query, to add a constraint you merely put this statement onto the end, WHERE. This tells the database that you want certains values to be returned where they match a specific criteria.

There are a couple of SQL queries that are incredibly important with them being:

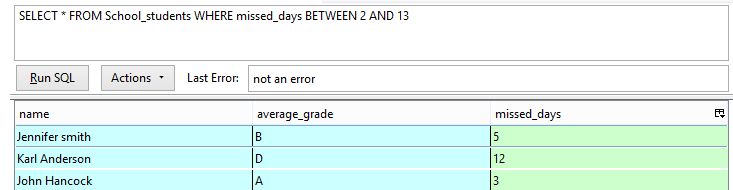
=, <, >, ≤, ≥, =!, BETWEEN, LIKE, and IN. The equal sign is self-explanatory : it sets a condition where it will return values that are equal to the value you set. Let's say you want to return all the information about some kids who have had 5 missed days of school. The statement would look like SELECT \* FROM School\_students WHERE number = 5. Notice that I used an asterisk to show that I wanted all values returned. The Query would bring you back something looking like this.



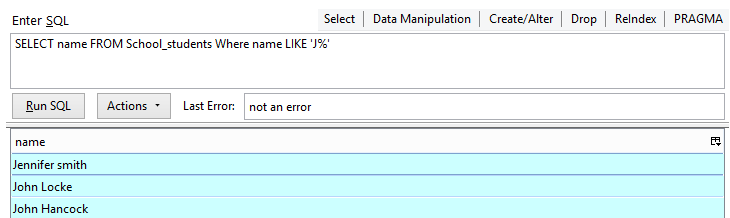
The equal sign can easily be replaced with any other sign that was mentioned above. Obviously there is greater than, less than, greater than or equal to, and less than or equal to. The sign that looks like this =! means not equal to, it will return all values that don't not match the selected one ( it can also be written as <>). For example you want to find students who do not have an average grade of a B. The query would look like SELECT \* FROM School\_students WHERE average\_grade <> ‘B’. The ‘ ‘ signals that the select value is not a number but a string.



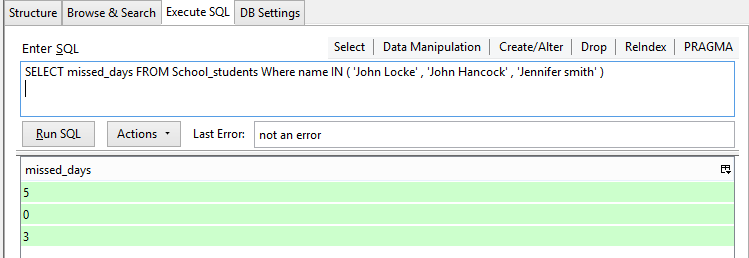
BETWEEN is used when you are looking for values, usually numbers that are between two set values. BETWEEN uses an additional statement piece called AND which is used when you combine two queries regarding one column into one statement. For example you want to find the missing days of a student where their number of missed days is between 2 and 13. The statement would look like SELECT \* FROM School\_students WHERE missed\_days BETWEEN 2 AND 13.



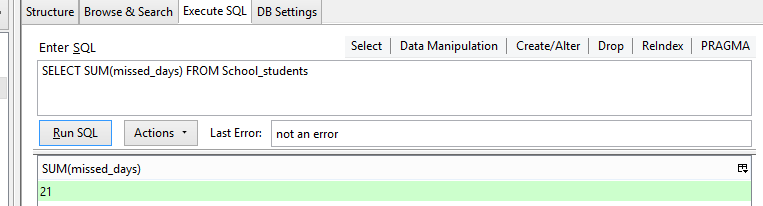
LIKE is a very unique query tool as it primarily deals with strings. It's basically the equal for strings but it can also find patterns in words. It functions by putting in the column name then LIKE, but then it gets confusing as there are two tools that are used in conjunction with LIKE. The % and \_ they are used when you are looking for strings but you don’t know the values of or you are looking for patterns. The % signifies that there are characters, it just means characters. Maybe there are hundreds or only one, it's basically just a placeholder, while \_ means a single character. In a query like this ‘z%’ would mean anything starting with the letter z. Let's say in our database that we want to know the students whose name begins with J. The SELECT statement would look like SELECT name FROM School\_students Where name LIKE 'J%'



IN is the most confusing of all the queries as it allows for you to add another query into your statement called a subquery. But it is most commonly used as the plural operator of like. It will show all values like a couple of values. For example if you wanted to find the absences of multiple students whose names were John Locke, John Hancock, and Jennifer smith. The query would use SELECT missed\_days FROM School\_students Where name IN ( 'John Locke' , 'John Hancock' , 'Jennifer smith' ).



Important parts that are helpful when using queries are basic math queries. SUM, if the data type is integer it will total all the values in query; AVG, will average the values of integer values in a query; COUNT, it will tally the amount of values that match a query; MAX, finds the maximum value of a query; MIN, finds the minimum value of query; And you can do basic math functions in SQL such as divide (/), multiply (\*), add (+), and subtract (-). For example if you want to find the total number of days missed you would just add SUM(missed\_days) and SQLite will instance find the total of the values. The query will look like SELECT SUM(missed\_days) FROM School\_students.



ALTER STATEMENTS

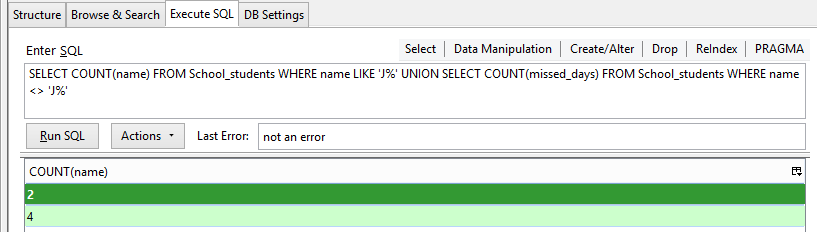
ALTER is used to modify and change existing tables by adding or deleting columns. Alter is almost never used because when a person designs the layout of a table they usually plan out all the values they will ever need to record. But maybe a school in the future didn’t need to record people’s grades. The structure of the statement is as follows: ALTER TABLE then table name, then depending if you want to add or delete a column it could go drop (delete) or add COLUMN then the column name. It’s important to note if you’re adding a column you need to add the data type it uses. If you want to add a column that tells the height in cm of the students the statement would look like ALTER TABLE School\_students ADD height Integer. In the Browse & Search on the far right there will be a black column meaning that the SQL statement went through. ALTER only deals with table aspects not rows, meaning you have to use insert to actually put values in the table.

DELETE & DROP STATEMENTS

DELETE is used to remove rows from tables. It's self explanatory, if you want to remove a record. For example if you had a student that attends your school but later moves to another school. The basic structure is DELETE FROM then the table name followed by what your constraint using WHERE. If you wanted to delete John Hancock from your table for some reason you would use the basic structure looking like DELETE FROM School\_students WHERE name LIKE 'John Hancock'. The statement could use an equal sign but LIKE would do the same. In the Browse & Search John Hancock is gone from the table. DELETE statements are very important when dealing with SQL injections because if someone was to delete everything from a business database such as credit cards, email address, and other personal information the business would be ruined. DROP is the same as DELETE but instead of a single row inside a table DROP deletes the entire table. Its basic structure is DROP TABLE then the table’s name.

Advanced Statements

If you look at the upper right hand corner of the SQLite tool bar there are even more statements that can be used but the one of the most important SQL terms that is rarely covered is UNION and UNION ALL. These terms solve so many problems, perhaps you might have thought why can’t I just use two statements? With UNION and UNION ALL they allow you to combine statements. The difference between UNION and UNION ALL is that if you had two statements using UNION and they return duplicate values UNION would factor out all duplicate values while UNION ALL would return all values regardless of duplicates. This means in a large database it's a lot faster to use UNION ALL because it doesn’t have to calculate duplicates, it just displays the values. For example if you want to find the amount of people whose name began with J and the amount of missed days that people whose name did not begin with J. The statement is complicated but if you break it down it reads as follows SELECT COUNT(name) FROM School\_students WHERE name LIKE 'J%' UNION SELECT COUNT(missed\_days) FROM School\_students WHERE name <> 'J%'. SQLite displays the values in order of the SELECT statement.



Advanced Theory

One of the biggest ideas when dealing with SQL is data normalization. Data normalization refers to the process of making “clean” and “simple” data. It means that there is no redundancy in tables, no columns that have the same values or meanings. This also indirectly means that table size (column wise) is kept to a minimum because the purpose of SQL is to find relationships between data and when the data inside is redundant it makes for bad tables. Not only this but if the data has no correlation which is incredibly rare in the database fields because usually data collected is sorted for example the age of bees and the amount of freshwater in Ireland there is no correlation. This ultimately makes one of the principles of databasing, data needs to be organized to keep integrity. To keep integrity and lower redundancy when working with large data sets the columns inside of the tables are usually split into other tables not only to keep it organized but for security reasons to prevent SQL injections.

Edgar Codd, creator of relational databases and SQL first came up with the ideas for “rules” regarding databases. He wanted to keep everything organized just like tables to make everything efficient so that databases could grow and expand. To make a function database he established some forms to regulate and oversee how data was stored. When databases had no issues with querying, inserting, and updating then the database was normalized. But in the real world it is exhausting to keep a growing database normalized because there will almost always be issues with inserting data for example, what if there is no value for a column making it null.

**SQL Injections**

SQL injections are completely unique from computer security threats like RAT, trojans, and viruses. SQL injections are targeted pieces of SQL meant to extract information, destroy information, or manipulate information inside a database. SQL injections are the most destructive and one of the easiest security threats to block. Injections work by targeting databases, but how do you find a database? Almost every website will have a database. If there is a place to enter passwords, usernames, or basically any information that needs to be stored there is a database.

Well how does it work? It happens in two steps 1. The database needs to be analyzed. This happens by testing the water of the database, entering stuff that would not normally be entered step 2. After the vulnerabilities of a database have been figured out, it's time to interact with the database. Assuming you know the name(s) of the table(s) in the database you can begin working. If you remember from all the statements used they always need to know the name of the table INSERT, SELECT, and DELETE. So the first step is to find out the column names, vulnerabilities, and table names. But every database is going to be unique so it depends on the security and type.

The most basic SQL injection that will work on low security databases is ‘or 1=1--. This works because when you log on to a website the statement that is used will get the password you give it and compare it to the values in a database using SELECT. When a value is brought back then it’s correct if no value returns then it’s false. But the injection adds an unexpected value that most databases won’t be expecting. Most databases won’t expect values like: ‘ ; . , \* ( ) & ^ % $ # @ ! { } | [ ] \ these values are unexpected syntax. In a select statement syntax is one of the most important parts of knowing if something is a string or integer. For example if the database expects you to put in only text with no syntax devices then if you were to put in syntax it would cause a SELECT anomaly. A SELECT anomaly can cause a message to be sent back to a website showing table names, and column names. This means to find table names and column names an anomaly needs to be created, which is (usually) caused by incorrect syntax.

**Applications and Beyond**

SQL is by far one of the most important technological innovations in the last century. It changed how data was stored from being written physically to stored in imaginary tables in micro processing chips. Skills in SQLite can be transferred to any other SQL program like MySQL or Oracle. Most websites are made with SQL and HTML, if you know even a little bit of HTML and SQL you can make functioning websites for money. SQL can also be used with games, if you need to know people's passwords or username. Most popular games have large databases that require a lot of workers to manage. SQL can also be used as a freelancing tool. Many companies and businesses need to have a database set up or they need data to be analyzed. SQL can be used in conjunction with HTML, Java, Javascript, Python, and its skills can be transferred to statistical languages like R.

**RESOURCES**

Crew. NA, HackThisSite.org. “Hack This Site!” *Hack This Site*,

[www.hackthissite.org/missions/realistic/](http://www.hackthissite.org/missions/realistic/)

Eagan, James. “How to Read Datasets.” *Project Datasets*,

[www.perso.telecom-paristech.fr/eagan/class/igr204/datasets](http://www.perso.telecom-paristech.fr/eagan/class/igr204/datasets).

Guru99, director. *What Is Database & SQL?* *YouTube*, 12 July 2013,

[www.youtube.com/watch?v=FR4QIeZaPeM](http://www.youtube.com/watch?v=FR4QIeZaPeM).

Khan, Salman. “Intro to SQL: Querying and Managing Data.” *Khan Academy*, Khan Academy,

[www.khanacademy.org/computing/computer-programming/sql](http://www.khanacademy.org/computing/computer-programming/sql).

Pickett, Patricia. “The History and Importance of SQL.” *The Balance*,

[www.thebalance.com/what-is-sql-and-uses-2071909](http://www.thebalance.com/what-is-sql-and-uses-2071909).

“SQL.” *Codecademy*, [www.codecademy.com/catalog/language/sql](http://www.codecademy.com/catalog/language/sql).

“SQL Injection Cheat Sheet & Tutorial: Vulnerabilities & How to Prevent SQL Injection

Attacks.” *Veracode*, 6 Oct. 2017, [www.veracode.com/security/sql-injection](http://www.veracode.com/security/sql-injection).

Tchibo. “Tchibo/SQL-Tutorial.” *GitHub*, 19 June 2016,

[www.github.com/tthibo/SQL-Tutorial](http://www.github.com/tthibo/SQL-Tutorial).