Chapter 6

Introduction to SQL

6.1 What is a SQL? When would I use it?

SQL stands for Structured Query Language. It is a language used mainly for talking to database servers. It's main feature divisions are broken into data description, data access, and user privilege. It is used as the front-end language for many applications such as mysql, postgresql, and oracle. You would use a database, and subsequently SQL whenever you have a large dataset that needs to be accessed easily very frequently or even simultaneously by multiple clients. Examples of when you would use a database could consist of generating reports, data storage, and more recently as backends to websites.

SQL is the standard for relational databases. A relational database stores information in tables made of many rows and columns where a row represents a record and a column represents an attribute. Databases typically provide ways to add, delete, and search records in a table. Typically one column of a table will be considered a "primary key". This primary key should be unique to each record and allows for faster searches on this column.

Example of a Relational Database

Students Table

FName	LName	StudentID	College	Year
Justin	Ennen	83746387	Science	Senior
Dan	Bass	83635563	Management	Junior
Michael	Chazoule	83554752	Mathematics	Junior

Above you can see an example of a relational database which holds data about various students at a college, their names, year of school, etc in a table called Students. This will provide a good example for choosing a primary key. At first you might think it would be a good idea to choose the name of the student as the primary key, however there can be multiple students with similar or even the same name. In this case, it is best by far to choose the student ID as the primary key, since every student has their own unique ID that belongs only to them.

6.2 SQL Language

The first thing to take a look at is the various commonly used data types available to us in SQL. These are outlined in the table below:

Variable	Description
CHARACTER(n)	string with a fixed length of n
INTEGER(n), SMALLINT, BIGINT	integers with various precision, precision n, 5, and 10 respectively
BOOLEAN	true or false valued variable
FLOAT	decimal numerical value, with a mantissa precision of 16
TIME	stores hours minutes and seconds

In addition to the variable types, you should know the various operations you can perform. We will first look at the operations that allow you to edit within a table, and then later briefly look at the operations that allow you to navigate the whole database. The main operations or statements you will use within a table that we are concerned with are SELECT, INSERT, UPDATE, and DELETE.

Statement	Description	Example
SELECT*	used to retrieve records from the database	SELECT column, column2 FROM table WHERE column = value;
UPDATE*	used to change existing records	UPDATE table SET column = value, column2 = value2, etc WHERE column = value;
DELETE*	removes a record or records from a table	DELETE FROM table WHERE column = value;
INSERT INTO	used to add an entry to a table	INSERT INTO table (column1, column2, column3, etc) VALUES (value1, value2, value3, etc);

**It is important to note that these statements do NOT require a where clause to check values, if there is no where clause, then the operation will be applied to the entire table*

As well as each of these operations within a table, there are operations that allow us to manage multiple tables within a database or even multiple databases. These operations are each fairly straight forward and obvious.

Statement	Description
SHOW DATABASES;	lists all accessible databases
SHOW TABLES IN database;	lists all accessible tables in “database”
USE database;	makes “database” the currently active database, operations will assume this database as the basis for other statements
DESCRIBE table;	lists all attributes of “table”
CREATE TABLE tablename (column1 variableType, column2 variableType, column3 variableType, etc);	creates a table with the name tablename and creates columns or attributes to hold each of the specified variables with the given names column1, column2, etc.

6.1 Creating an Example Database

We will next go through the steps of creating the example database about the students we have shown previously. You are encouraged to follow along with these steps to create the database as well. If you would like to do this, one of the easiest, but certainly not the only, SQL programs to set up is MySQL, an open source database application provided by Oracle. You may use any of the MySQL Community Server applications to follow along or use another program of your own choosing as SQL is fairly standardized across all platforms and database applications.

The very first thing we need to do is to create our table. We want it to have columns for the first name, last name, student ID, college, and what their current year is at the university. We also want to specify the student ID as the primary key. We also want to make sure that there is always a supplied name for the student, that is the students name cannot be NULL, or empty. To do these two things we will use constraints, which are descriptors added after the variable type. After we have created the table, we want to make sure there were no mistakes, so we will ask to have our newly created table described to us after we have created it. Therefore our first command will look like this:

```
mysql> CREATE TABLE Students
```

```
(
FName CHARACTER(100) NOT NULL,
LName CHARACTER(100) NOT NULL,
StudentID INTEGER PRIMARY KEY,
College CHARACTER(100),
Year CHARACTER(100)
);
mysql> DESCRIBE Students;
+-----------+-----------+------+-----+---------+-------+
| Field     | Type      | Null | Key | Default | Extra |
+-----------+-----------+------+-----+---------+-------+
| FName     | char(100) | YES  |     | NULL    |       |
| LName     | char(100) | YES  |     | NULL    |       |
| StudentID | int(11)   | YES  |     | NULL    |       |
| College   | char(100) | YES  |     | NULL    |       |
| Year      | char(100) | YES  |     | NULL    |       |
+-----------+-----------+------+-----+---------+-------+
5 rows in set (0.07 sec)
```

After running the describe command, you should now see a table describing the fields of your newly created table saying which values can be null, and which fields are considered any type of key. We have now successfully created our first table! Now we just have to populate it with data. We will do the first together, and leave the last two up to you as exercises.

Our first student, Justin Ennen, is in the college of science, has a student ID of 83746387 and is a senior. Therefore, our insert statement will look like the following:

```
mysql> INSERT INTO Students
VALUES ("Justin", "Ennen", 83746387, "Science", "Senior");
mysql> SELECT * FROM Students;
+--------+-------+-----------+---------+--------+
| FName  | LName | StudentID | College | Year   |
+--------+-------+-----------+---------+--------+
| Justin | Ennen |  83746387 | Science | Senior |
+--------+-------+-----------+---------+--------+
1 row in set (0.00 sec)
```

Below our insert statement you can see we used a select statement, with an asterisk as the column name. The simplest way to think about this is think of the asterisk as meaning "everything". So we are saying select everything, so that we can see the changes we made to the table by adding our new row. For the sake of the rest of the example we will assume you have added the other two entries to the table.

We have just received a request from the registrar at our university for the student ID of Dan Bass. This information is thankfully already in our database, all we have to do is retrieve it from the table! We can do this with a command like the following:

```
mysql> SELECT StudentID
FROM Students
WHERE FName LIKE "Dan" AND LName LIKE "Bass";
+-----------+
| StudentID |
+-----------+
|  83635563 |
+-----------+
1 row in set (0.06 sec)
```

If we typed this correctly, and the entry was in our database, this should have returned the student ID we needed. This operation used few new things. It used the "LIKE" operator which is similar to using the "=" operator for numbers, but with strings, and it used the AND keyword to allow us to use two criteria for the select statement. You can apply these same concepts to the update, and delete statements as well.

Chapter Summary Questions and Exercises

1. Create a statement to insert James into our student database using the INSERT statement. His ID is 6857938 and he is a sophomore. He is in the same college as you are.
2. Use what you have learned about forming SQL statements to now delete James from the database using the DELETE statement. This will work very similarly to the SELECT statement.
3. Now try using the UPDATE statement to change Dan's Student ID to the number 1000000.
4. Take the time to find an SQL connector for your favorite language online. A connector is the library you will include that allows you to easily make connections to and send statements to your SQL database from inside one of your programs.