Lesson 2-Accessing Data Part 1

Lesson 2 Concepts

1. A question asked of a database is a "query".
2. The database responds to queries with a result set.
3. Structured Query Language (SQL) is the language used in this course to query a database.
4. MySQL is the type of database management system that we send our queries to in this course, though there are many others.
5. We can make sense of our data (and better answer "human" questions) by creating smaller data sets from larger ones.

A Question Asked of a Database is a "Query"

Part of our job as the web developer is to be the intermediary between the human world and the computer world. We need to be able to speak Human to humans, and Computer to computers. When asking "human" questions of a database, a question is called a "query". Query is also used as a verb: "I will query the database to answer that question for you".

The Database Responds to a Query with a Result Set

After receiving and processing your query, the database will return a *result set*, a smaller version of the full data set. This result set does not change the database in any way, it is a temporary answer to your query. In addition to your requested data, the result provides the total number of rows in your result set.

Structured Query Language (SQL) is the Language Used in This Course to Query a Database

SQL (often pronounced as, "sequel") provides the syntax used to formulate our queries. It has built in tools that can give super powers to our queries, however, it has a set structure that we must follow (see below). A poorly formed or ambiguous query will result in an error message.

MySQL is The Type of Database Management System That We Send Our Queries to in This Course

MySQL is an extremely popular, fast, and open-source database management system. It is included in most web hosting packages and is relatively easy to set up and use. There are numerous graphical user interfaces (GUIs) for MySQL, but one of the most useful and powerful is the one that we will use in this class, *phpMyAdmin*. phpMyAdmin is also available in most hosting packages and found under the Database section of the Control Panel interface of a web hosting package. It provides a window to enter your queries into, as well as the ability to import, export, and modify your database and tables. In addition to MySQL, there are other types of database management systems that you will learn about later in this course, and in the C# stream.

We Can Make Sense of Our Data by Creating Smaller Data Sets from Larger Ones

Imagine looking at the full data set for the Amazon website. Trying to find what we were looking for would be practically impossible. By creating smaller versions of the full data set (and removing unwanted information) we can output user-friendly content to a website. It is through our SQL queries that we can create smaller data sets that will better answer the questions of our human users. The structure of an SQL query is flexible, but requires adherence to a set syntax and ordering of the clauses in the full query statement *.

The most basic statement will select all of the information from a particular table:

```
SELECT * FROM employees;
```

In the statement above…

- the keyword, **SELECT**, asks the database to retrieve data
- the asterisk (*****) is a shorthand character for anything/everything
- the **FROM** is a required keyword pointing to the table to search in
- **employees** is the specific name of the table to conduct the search in
- the semi-colon (**;**) is used to end the statement, like a period at the end of a sentence.

Note: although SQL is case-insensitive, it is best practice to type your SQL keywords in uppercase letters as shown above.

We can limit our results further by specifying the columns from the table that we wish to include in our results. In place of the asterisk, provide the column names, separated by commas:

```
SELECT id, last_name, first_name FROM employees;
```

Instead of showing all of data from the employees table, the above query will only provide the data from the first three columns of the database. You are also able to reorder the columns if you wish:

```
SELECT first_name, last_name FROM employees;
```

This allows us to provide results that may be more user-friendly. If you want to combine the data columns together so that first_name and last_name appear in a single column, use the CONCAT() function.

```
SELECT CONCAT(first_name, " ", last_name) FROM
employees;
```

To add a space between the first and last names with double quotes and 1 space followed by a comma.

Along those lines, we have the ability to provide custom names for the columns in our results. This is referred to as giving a column an *alias*, and is performed with the AS keyword:

```
SELECT last_name AS "Staff Member" FROM employees;
```

Note the AS keyword and that the desired heading is in quotes. We can create aliases for any of the selected columns:

```
SELECT id AS "Emp #", last_name AS "Staff" FROM
employees;
```

Though the above syntax limits the columns in the results, the WHERE clause allows us to limit the number of *rows*. If you think about a shopping website, when you click on a category (“automotive”, “MMORPG”, “desserts”, “fiction”, “comedy” et cetera) you are sending a request to the database to provide a result set that only includes rows that belong to that category. This is achieved by specifying conditions in the WHERE clause:

`SELECT last_name FROM employees WHERE role = "Sales";`

Tagged on to the end of the SELECT… FROM section of our statement, we are specifying an inclusion criteria for the result set - "I only want to see rows where this is true". In the case above, the result set would not include the last names of *all* the employees, only the last names for rows where the role column had a value of "Sales" inside of it. The syntax for the WHERE clause is…

… `WHERE name_of_column operator value;`

- `WHERE` is the keyword indicating that a filter should be applied to the results
- `name_of_column` is the exact name of the column where the comparison should occur
- `operator` is the type of comparison to perform. Options include:
  - `=` equal to
  - `!=` not equal to
  - `>` greater than
  - `<` less than
  - `>=` greater than or equal to
  - `<=` less than or equal to
- `value` is what to check in the column for. A string needs to be in quotes, but numbers do not.
- A semi-colon (`;`) to end the statement.

This additional clause limits the result set to include only the rows where the comparison of the value stored in the specified column, and that of the provide value return true.

SELECT Statement Structure Including a WHERE clause

SELECT	Columns separated by commas, or * for all columns.	**FROM**	Name of the table in the database.	**WHERE**	Name of the column to search in.	Comparator = != < > <= >=	Value to compare to the data in the specified column.

Examples

SELECT	last_name	**FROM**	employees	**WHERE**	role	=	"Sales"
SELECT	item	**FROM**	stock_items	**WHERE**	price	>	50
SELECT	id	**FROM**	employees	**WHERE**	role	!=	"Sales"

*NOTE: Although “query” and “statement” may seem to be contradictory terms, the relationship is better understood as follows:

Query/question: “How many employees are full-time?”

Statement: “Show me all of the people in the employees table with a status of full-time.”

SQL: SELECT * FROM employees WHERE status = “’FT”;