Names: Gabrielle Luna and Michael Carr

Instructor: Dr Taneja

Class: CS355

Citi Bike Project

Before we started this implementing code, the table that was given to us had to normalized. We did this by separating the table into two parts. The main reason for doing this was to eliminate any duplicate data. If you look at the original table, you will see that there were 5 columns of data for the start station and 5 columns for the end station. The information presented in each group was not unique to the start stations only or the end stations only but characteristic of a station regardless of whether the station was a start station or end station, so the stations were given their own table (stations). From there, the ‘trip’ table was formed with foreign keys of startStationID and endStationID referencing the stationID on the ‘stations’ table.

To implement the interface, make tables and populate them took a fair amount of research and experimenting. PHP is not a language that is like C++ or JAVA. It can be similar at times but for the most part, there seemed to be some minor inconvenience when implementing the interface, making the tables or populating them. If I told you that a computer was almost thrown out the window I wouldn’t be lying. Most Computer Science majors have a fair bit of experience with HTML and it's not all that hard to get a webpage to function the way you want. Throwing in a new language with database functionality is and was a challenge, to say the least. The way we did it was one person did the code for implementing the interface and one person did the database and queries. Our thinking was to limit each other from messing with the others code to minimize problems and crashing of the database or interface. We seemed to just let each other work and came together at the end to put everything together that came together somewhat seamless.

The PHP interface and all HTML code were done in Visual Studio with WAMP used to check syntax before uploading it to blue to run. The database was initialized on MySql via Blue but all subsequent code was done through a text editor with use of PHP through Chrome web browser. With the CSV file provided, there was no way we would ever get all that data into a single database so a subset of about 163 entities was used. This seemed like an amount that could get a good idea into what trends the data is holding but not too big to confuse us or any person who uses our PHP to run queries. The database was populated with the help of MySql WorkBench where the CSV file was converted to standard SQL code that could be uploaded to MySql on the Blue server. Once both the PHP and database were up, a few tweaks were made to the design and general bugs were fixed.

The biggest challenges we faced but it seems a lot of teams had been too was the connection to blue and how to implement PHP along with SQL and HTML. After finally getting connected to MySql via Blue, having a solid PHP page, making the database and getting it populated a lot of problems were solved by just investigating the internet. There is a vast amount of knowledge out there. We seemed to work good by talking about a plan, being quiet and getting it done.