**Project Report**

**Part I - Extract**

For this project we chose a dataset from Kaggle about Mt. Rainier weather and climbing data. This included two CSV files. The first one included information about different routes to the summit on Mt. Rainier, including date of expedition, number of expeditions attempted on that date, number of successes, and the success percentage. The second CSV file included temperature, wind speed, and humidity data for different dates, as well as some other data we decided to drop, such as Battery Voltage Average, Wind Direction Average, and Solare Radiation Average.

The other source we used was a HTML table from summitpost.org (<https://www.summitpost.org/mount-rainier/150291#chapter_6>) that had more detailed information about the routes included in the climbing data CSV file, in addition to several others. This information included: duration to summit, difficulty (alpine grade), elevation gain, maximum grade, and season to approach.

**Part II – T**ransform

There was a fair amount of cleaning and transforming required for this data.

First, we loaded the CSV files into DataFrames using Pandas in a Jupyter Notebook. Then, we dropped unwanted columns (mentioned above). We also renamed columns to make the titles shorter and more intuitive.

When we were looking at the climbing data, we realized there were duplicate rows for certain routes and dates (for example, two trips to Disappointment Cleaver on 10/19/2015 and one to Little Tahoma). We had to figure out how to drop duplicate rows based on two criteria. To do this we used “.drop\_duplicates” using a subset and keeping one instance of a duplicate row. This effectively kept one trip per route per date.

For the table data that was scraped from the web, our initial scrape resulted in multiple tables, so we had to identify the one we needed, so we saved this one using indexing. For this table we dropped unwanted columns and renamed columns to be easier to refer to in SQL.

**Part III - L**oad

After cleaning and transforming the data, we loaded it into a SQL database using pgAdmin. First, we created a climbing data table, a weather table, and a routes table. When we went to import our DataFrames into these tables, we ran into some issues because of how the DataFrame columns were named (using capitals and spaces). We fixed this by using lowercase for all column names, and underscores instead of spaces. This worked well and we successfully loaded everything into SQL.

Once we had all of our tables set up in SQL, we started exploring possible queries that could be run on this new database. First, we wanted to see weather for each trip. We weren’t sure if this would work since we had multiple trips on one day, but only one set of weather data for each day. Using a Left Join effectively mapped weather data to each trip, even if it was on the same day.

Next, we tried to see how we could view data just for a given date. For this we used where, like, and or statements and a wildcard operator so that the query would pull out all dates starting with ‘11’ or ‘12’. This took some experimenting!

The last query we ran we had to set up in two parts. First, we joined our routes table to the climbing statistics data and saved it as a view. Then, to find the success average vs. difficulty rating for different routes, we queried this view, but realized there were many null values in the difficulty rating column. We realized that the table we scraped from the web had a lot more routes on it than the CSV file we downloaded, and that some of the routes had different names in each file (for example: Fuhrer Finger in one, and Fuhrer’s Finger in the other).

Since our climbing data CSV file only had four different routes in it, we decided to rename them based on what the HTML table called them. However, for one route, Little Tahoma, there were several different route names. Some Googling revealed that Little Tahoma is a peak, and it has multiple paths on it that one can take to get to the summit. So, we decided not to rename this one because we couldn’t reliably say what its alternate name is. We went back to our Jupyter Notebook and renamed the routes in the climbing data DataFrame and reloaded it into SQL.

After renaming the routes we were able to avoid all of the null values that appeared the first time, except for those associated with Little Tahoma.

If we had more time and wanted to explore this data further, we could export the query results from pgAdmin into a file that could be plotted using Pandas or Matplotlib in order to better visualize which routes have higher success rates in different seasons, etc.