Names:

Jonathan Ong – jso464

Prarthana Neotia – pn4322

Final Project Report

A chronological order of challenges encountered is one of the more comprehensive ways of logging the challenges and learning curves experienced this semester, and one of the most interesting ones was picking our database for the first laboratory assignment. A database elaborating on the different zip codes in New York City, and the different groups of demographics that resided within them, was considered. These groups consisted of percentages of the population belonging to each gender, the ethnicities or racial classifications of the different sub groups, citizenship status, etc. It was soon recognized that while the dataset was robust, interesting and informative, the survey results used to build the csv file had not hit upon any characteristics of the residential arrangements that would allow us to derive trends, or group the categories into tables and build junction tables between them.

The focus was then narrowed and we zeroed in on a dataset that elaborates on the different universities across the United States of America, with further information on the majors pursued in each of these universities, the rates of degree completions, past statistics of SAT scores made by enrolled students, the ethnicities of the students, the duration of the university programs, etc. As college students with university related exposure related to The University of Texas, data collected extensively from across the nation that ranged from the diverse backgrounds of its students to the financial debt each university put them in, or the percentages of different majors pursued at colleges, was of great interest to us. This dataset allowed us to build multiple attributes - tables that housed related information. The most elaborate table, College, logs data on the institution names, city of location,

average student debt, duration of college degree programs, etc. Junction table College_Flag was made between the table Flag and College, junction table College_Major was made between the table Major and College, and junction table College_Race was made between the table Race and College. The significance of building the conceptual and logical models seemed trivial at first, but the skeletal foundation of thought they formed became quickly apparent as we progressed through our lab and even through the professor's lectures.

An initial attempt at outlining the logical model involved us segregating the colleges based on public and private, and then rearranging the respective attributes from what is currently the College table, for each college under the Private and Public tables. The underlying issue might have substantialized before the reader's eyes by now – there was a repetition of every attribute (percentages of each major pursued, demographics of students in each university, whether the degree pursued in the public/private university was two years or four years in duration, etc.) in both tables (Public and Private). It was pointed out that the model, as it appeared now, was not only overly convoluted but also redundant. With easily accessible and insightful guidance from our TA, the logical and conceptual models were restructured to comprise the College table, which serves as the bottom, fundamental layer of our current logical structure. Condensing the two repetitive tables mentioned earlier into a single list of attributes immediately simplified the model. The junction tables all draw from College, and each junction table draws from a separate, smaller table, which contains the identification number for the featured characteristic of that junction table.

This lab assignment also required us to describe each attribute, which compelled us to individually evaluate the significance of each attribute of our dataset that was included in the logical table. It also allowed us to recognize that while some pieces of information might be provided in the dataset might not be presented in the best format they were significant enough to not be ignored. For example, we found two separate columns in the original csv file – one of which confirmed or denied that the university in question provided degrees that were four years in duration. The other one confirmed or denied whether the degree provided by the university in question was less than four years in question. As we went through each variable included in our model and hence derived tables, we realized that combining the two mentioned columns into a single attribute that provides information on whether the degrees at the university under consideration is four or less than years in duration would not only make it possible to use these columns in our model but also be a more efficient way of presenting crucial information. Some fundamental toying with Excel allowed us to create this attribute and use 'iclevel' in our models, that would form the foundational base of all data queries going forward. Generating SQL queries to generate the tables and develop the foreign and primary key relationships was the next step, one that was a great way to implement the theoretical discussions on primary and foreign keys held in class thus far. It greatly helped flesh out the precise functioning of keys, and how they served as connections and points of reference between tables. Both my partner and I had taken a course on Data Visualization the previous semester, a course that delved fairy deep into SQL queries and advanced data mining and manipulation. We believe that the course

taught by Professor Cohen this Fall went over the basics of SQL before digging our nails deep into more complex waters, which helped build a strong foundational understanding of this data mining language – one which I believe had been in lacking so until this semester. By the time we progressed to SQL injections, we had covered simpler functions like Group By, Order By, and numerous other filters. This allowed us to wrap our head around the broader concept of SQL injections, and not only recognize why it is poses a significant threat but also successfully create barriers that would protect our data from troublemakers.

While we had spent some time using SourceTree for commits to Github all through the course of data visualization this past Spring, using Github Desktop compelled us to systematically and meticulously log the content of each commit. This, along with the issue tracker we were instructed to maintain by Professor Cohen, allowed us to reflect back and comb through each project on receiving our grades, and lay our fingers on errors, without any obstacles. The interface of Github Destktop is also relatively more user-friendly than SourceTree's, and turned out to be a useful nugget of information for future reference.

Fixing past errors was the first task on Lab 2, which turned out to simply be the abridgement of the logical models that was discussed earlier. Once we implemented the TA's instructions on how to go about condensing our tables and displaying the data in a manner that was more comprehensive and not as redundant as our initial set of models, it became clearer how the implementation of the following tasks on the assignment was possible. Were we to stick to the initial model, the run time of the Python code to follow would have been excessive and the

system would be going through superfluous cycles to produce a rather convoluted representation of our university statistics.

Using Python to generate SQL queries was initially a confusing concept, and using relatively unfamiliar functions brought with it long hours of debugging. It quickly became clear, however, why writing SQL queries through Python allowed the programmer to generate more generic and flexible code. Mining data from the overwhelmingly large csv file by generating queries via an OOP language, simultaneously creating and importing the tables sketched out in the Logical model allowed us to explore the waters of navigating through csv files using Python. Our experience in this OOP language so far had been restricted to writing blocks of code that accomplished an immediate and relatively simpler objective. These objectives revolved around mathematical concepts, and understanding the logic behind the mathematical puzzle was a bigger part of the project than was navigating through the different functionalities of Python. While that helped learn the art of critical thinking and organizing your thoughts in a structured fashion before beginning to write code to accomplish said objectives, the assignments in this course allowed us to build a deeper understanding of how csv files are structured. The checks (NOT NULL, or special characters, for example) that one must remember to use when writing SQL queries became more intuitive to us as we bugged and debugged repeatedly.

One of the obstacles encountered when understanding how to read in data from the csv file using Python and present it in the tables sketched out in our model was the transposition of data from the dataset and into the Logical model's

attributes. Wrapping our heads around extrapolating consecutive columns of data and selectively re – arranging them into smaller tables involved not only deep thinking about what the data really portrayed but also about converting numbers from Excel cells into Python lists that were more accessible.

To elaborate further, it is impossible to individually point at specific chunks of data in a csv file through Python, but it is possible instead to convert those columns of data into lists. This allows the programmer to implement conditions on the data 'cells' or list indices. For example, lists allow the programmer to ensure there are no cells with NULL value being interpreted falsely while said attribute is being imported into the table.

Once the tables were created and imported via Python, the run time was recorded for each block of import code. This segment of the lab assignment allowed us to not only evaluate the efficiency of our Python code but also the magnitude of data we were manipulating, which further presented selected information in logically separate tables that allowed a new pair of eyes to quickly sift through and understand everything the dataset had to portray.

The third lab assignment centered around ameliorating our understanding of the different queries that allow the programmer to analyze the data presented in the tables and draw trends, present joined from two different tables based on a significant, common attribute. Most importantly, this segment of our third lab assignment allowed us to better understand just how powerful the Group By and Order By clauses were, and their extensive use in portraying specific aspects about collected data.

We were instructed to use an amalgam of filters, clauses and joins, which refined our SQL skills while also allowing us to toy with our tables and zoom in on certain factors within our dataset. Our queries filtered out certain noticeable aspects of universities across the nation. For example, one of them filtered out all the universities, either public, private nonprofit, or private for profit, that offered degrees that were four years in duration. Yet another one selected the maximum average SAT scores for predominantly those universities that granted certificate-degrees, associate's-degrees, or bachelor's degrees, depending on user input. This allowed an analysis of whether the average quality of students entering institutions offering different degrees differs.

Understanding Inner and Outer Joins were another important take away from this project. Joins went from being an abstract concept to understand from a textbook to a visualization of what common attribute two tables could be joined on and why that would be useful. Left outer joins selectively filter out only the common rows from the right table that match with the left, while retaining all rows from the left table. It became easier to understand the distinction between Inner and Outer joins when we used a Left Outer Join to list median debt at graduation for colleges with a majority of the students being a certain major, and another Left Outer Join to list colleges with only one race, and further list the city and the state of that college. Inner joins, where only those rows from both tables are extracted that find a match on the common attribute that the join is executed on, were used to join tables related to flags with the parent table to list colleges with flags with the number of culture or gender – based flags ordered by number of flags in descending order.

Other inner joins were used to list the percentage a certain race makes up of each college in descending order, and finally to list all colleges that have only one race, ordered by the name of the racial category. This exercise made junction tables go from a visual concept on Logical and Conceptual Models to a substantialized table built on a logical common attribute that brings together useful and relevant data from the parent table and the smaller table that pertains specifically to the characteristic of the university that the junction table pertains to. The functionalities of the other queries, for example - create views, executing queries on these created views, order bys, group bys, distinct, where clauses, and aggregated, while fairly simpler to understand in terms of logic became crystal clear after the multiple SQL queries this assignment instructed us to write. The menu to be created arranged all the functionalities of our code into a clear visual list, and this assignment aided in building an understanding the working behind db_connect.py and its usefulness. The final step before committing to Github for this assignment was an initial introduction to the concept of SQL injections and the potential harm they could cause to any python generated SQL query if not armed with sufficient protection. This, along with the guest lecture on SQL injections and hacking conventions, helped build an understanding of their purpose – to mainly exploit vulnerability occurring in the database layer of an application. These snippets of code are injected as user inputs inside queries, and can lead to substantial corruption or even deletion of the database, thereby leading to losses of extensive and valuable information.

The final project, though relatively simpler in terms of the writing and execution of code, was a new concept and introduced us to the method behind using

API clients in Python. Twitter was an interesting choice of Application Program Interface because it is heavily used on a global scale and has constantly changing volumes of activity on specific hashtags. This alters the limits that have to be set on the Python code that restrict the number of Twitter feeds that can be fed in in one read. An overnight gap in running the code and thereby searching Twitter for a specific hashtag not only churned out different results but also lead to the appearance of an error the following morning – one that had not existed the night before. This was because there was a tumultuous amount of activity on that hashtag over 12 hours, which lead to the number of Twitter feeds being read into the MySQL table having exceeded a permissible volume.

This assignment also drew on our understanding of JSON file formats from our course on data visualization from this past Spring semester, and built skills involved with navigating through these files and mining out the required information. One of the main takeaways from this assignment was learning how lists on JSON worked, and how potentially useful they could be when trying to access a specific desired part of a tweet, which when stored in a regular JSON file has multiple components that are worth multiple lines of output and would make the result, if the JSON format is left un tampered with, look incredibly convoluted and impossible to understand. After some debugging and in depth research on how lists in JSON are different from regular Python lists, and the functionalities of lists in JSON, the tweet initially stored in the JSON file was converted to a Python list and then back to JSON. This allowed us to use keys to access only the words that the tweet comprised of, thereby churning out a cleaner output that displayed only the

tweet string and not the other specifications that were stored in the JSON file. The topic for our Twitter search involved retrieving all tweets on hashtags for a specific list of majors from different universities across the nation. This list of majors is the same one as that sketched out in our dataset that has been in use for the past last assignments. Including the foreign key from one of our existing tables into the topic required us to rename the code names for the different majors, since these were a simple combination of alphabets and digits that were a subcategory of a larger attribute in the initial database. These codes allowed for navigation through the database csv file, but were virtually useless when searching Twitter for hashtags on major names. Debugging this issue involved manually renaming every code with the complete major name in words via Python. This allowed for the majors from the database to be matched with the tweets with the major hashtagged on the post to be transferred into the MySQL table. There was a little trouble in determining whether or not a limit should be set on the number of Tweets pulled in, but a limit was set in order to ensure that the file size limit for the backup was not being exceeded. Overall, this course has helped us learn a great deal about the real world threat of SQL injections, the efficiency of generating SQL queries via Python and the strength of this OOP within the database mining space, and ameliorated our understanding of SQL and its different functionalities.

Some issues still remain yet to be explored further – discovering how to retrieve as many Twitter feeds as possible without running into errors, and potentially creating backup files for the same without exceeding file limits.