Documentation of Decisions

We decided to convert the CSV files to Latin 1 because UTF-8 could not read certain symbols in the city names.

We selected only the columns we needed in order to efficiently compare and join data across the 3 tables: the cities, the countries, and the yearly values of average temperature for each city from the Average Temperature of Cities CSV; and the latitudes and longitudes, country codes, and populations from cities15000.csv.

We separated the Celsius and Fahrenheit temperatures, previously listed in one column, into separate columns to make them easier to read. We used the split function where the “/n was prompted in the column to split at the break for the new line in Python to accomplish this. We then worked to remove the parentheses from the Fahrenheit column to make it look a bit better. We did this by deleting the current Fahrenheit column, converting the Celsius column to a numeric data type, and inserting the new Fahrenheit column as a calculation from the Celsius column. We did this because our attempts to use “str.replace” did not work.

We decided to get to work removing duplicates from the Average Temperature of Cities data frame. We utilized a for loop to locate the duplicates by initializing an empty list and then appending any duplicates into the list. After the loop we run, our returned output was zero, indicating no duplicates. Once we tested for duplicates using the cities, country code, latitude, and longitude columns, we found that there were no duplicates. We repeated the process with the cities1500 data frame.

Cities1500 uses country codes instead of country names, and as we discovered several of these codes were missing. We sorted out all entries with a null country code value as well as other null values. We then went back to the Average Temperature of Cities data frame and searched for null values just to make sure that we didn’t miss anything. We decided to remove any entry that had null values for latitude, longitude, or population as we considered those values too important for what we wanted to accomplish with the project.

For the missing country codes, we attempted to replace the null values manually. We created a list out of the cities with null values that we named “null\_cities.” We very quickly ran into problems, however, and eventually decided that it would be better to drop those cities and move on. We used the “null\_cities” list to filter out and then drop the cities with null country code values.

Our reference table of country codes had both the 2 digit and 3-digit codes for reference, so we decided to do run a for loop for our large dataset of cities to verify that they all in fact used the 2-digit code to keep data integrity. We used a len function on the country code to see if any values were different than 2 digits and the output came back with a zero value, so no alternate codes were used.

We then returned to the “all\_df” data frame to ensure that all of the country codes were the proper, two-digit codes that we had decided to keep and not the three-digit codes for the sake of keeping our data uniform.

We then proceeded to load our data into a SQL database.

We belatedly realized that we needed to covert our column names to all lower case for them to work in SQL, so we proceeded to do that using the “.rename” function and “str.lower” function.

In the process of preparing the panda data frames for SQL, we discovered a row where all of the information was off by one column, with no city name listed. We decided to drop that row since the information in that row was not uniform with any of the other rows.