Final Project SI206 Report

Benjamin Riela & Ashley Newman

Goals for your project

We had many goals at the start of our project. First, we wanted to create a connection between the data that we collected that would be meaningful and useful for observation. Our initial plan was to use restaurant data from around the country to calculate the popularity of different restaurants by different neighborhoods and compare these ratings to aggregate data on income and prices in the areas. Our goal was to make a connection between these things through the visualizations. However, we discovered almost immediately that this was not feasible, so we switched our topic and created new goals for ourselves. Our new goals were to make connections between happiness scores in cities that were known for being happier and the temperatures and weather conditions in those cities. Our goal was to make connections between these things to see the correlation between weather and happiness levels. Our main focus for this project was to overall learn a lot about these topics. We knew these were important topics in the real world and wanted to become proficient in many of them by the end of this project. We also set goals to complete many of the project requirements like 2 or more visualizations and 1 API and 1 website.

Goals that were achieved

We achieved many of our goals. We were successfully able to calculate our data between weather conditions and happiness levels. We made successful visualizations that allowed us to make conclusions about our data. We also were successful in our goal of switching our topic to something more workable and interesting. In the end, we were very successful in completing our goal of understanding the topics in this project. We learned web scraping, how to use SQL and pull information from databases, and how to use, understand, and extract data from APIs. We now feel moderately comfortable using all of these topics. If we ever come across a situation like this in an internship or future project, we feel like we would be able to use skills from this project in the future, and that was our main goal of the project. We also completed our general project goals by having 5 visualizations, 2 APIs, and 1 website in the end.

Problems you faced

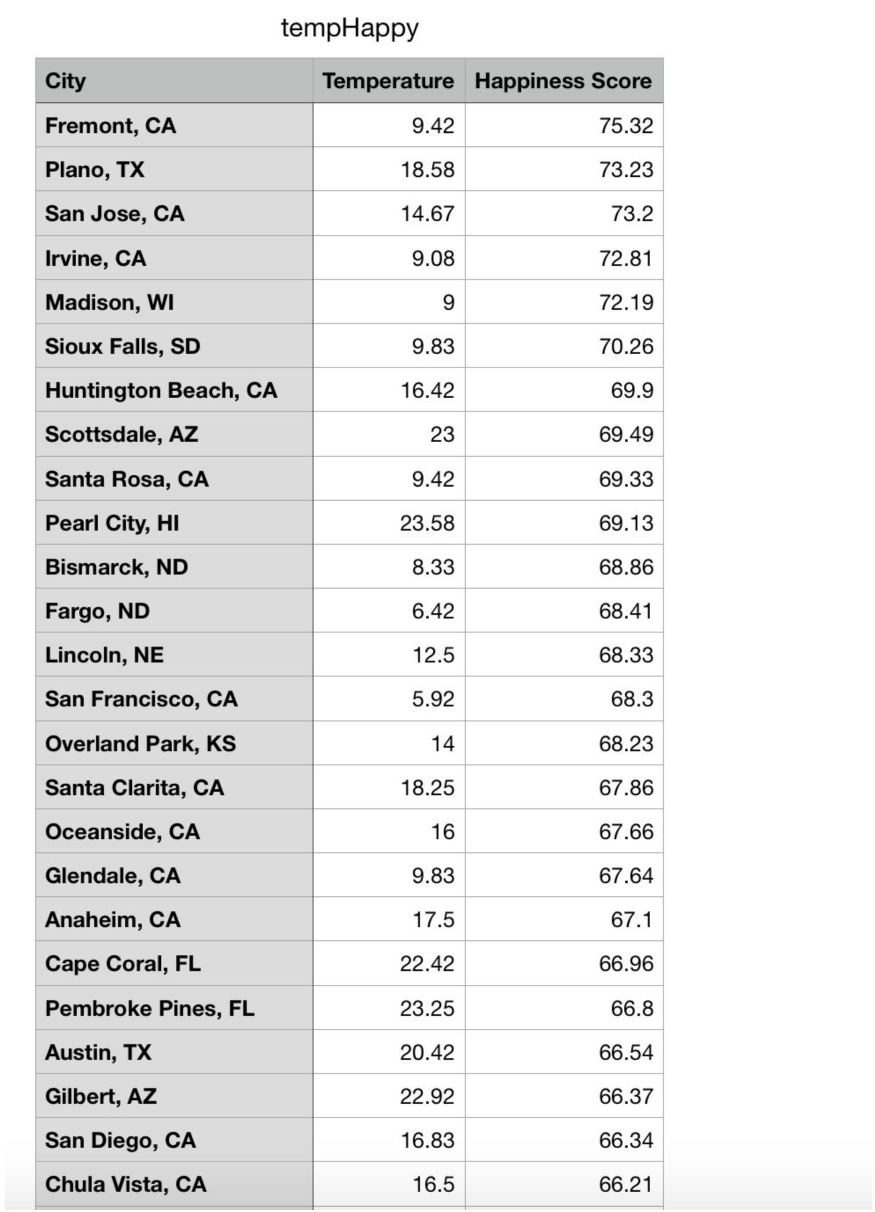
We faced many problems at the start of the project. Our first major issue was that the Zomato API would not allow us to access data after one use. We were fairly new at understanding APIs and felt discouraged by this initial problem, so we switched to the Yelp API. This worked only until we realized the data we were extracting from it was not connecting to the website we were using properly. This led to a couple of days where we had to regroup and reevaluate how to do this project. We then decided to switch our idea to measure happiness levels in cities to the temperatures and weather conditions in those cities. We started off by using the free OpenWeather API. This, however, did not work because we wanted historical data from past years, not current data from that day or hour. We then had to switch our API again and finally found one that worked: Meteostat. This API allowed us to view data from the past and extract it easily. Once we found this, we faced minimal problems. We discovered that we needed another API (MapQuest Geocoding) to get the latitude and longitude for cities in order to use the

Meteostat API. Once we got this, things began to run smoothly. Every so often we would struggle with an error in our code, but a quick Google search would always solve the problem. We also had some difficulty installing plotly, so we decided to use matplotlib for our visualizations. One last problem we encountered was with the Meteostat API. We were unable to access all of the data from it so we decided to extract as much as we could and make note of the fact that there could have been more for us to access if we potentially paid for it or did extensive research on how to find it.

Meteostat: https://dev.meteostat.net/docs/#about-meteostat
Note: All temperature data is in degrees Celsius and all precipitation data is in millimeters.

Your file that contains the calculations from the data in the database
Below are images of all of the csv files for our calculations from the data in the database. The actual files can be found in the zipped project folder.

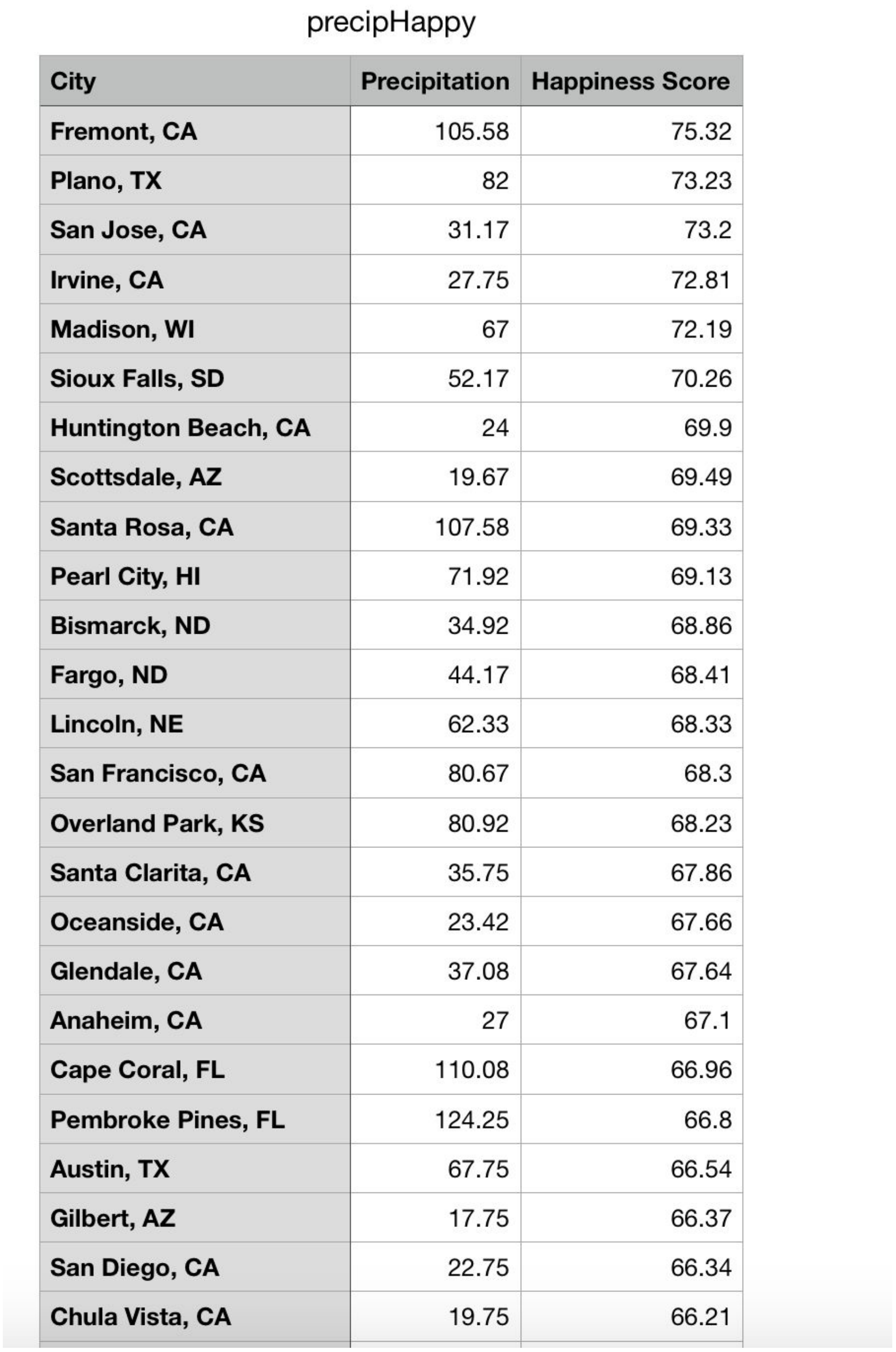
tempHappy.csv



tempHappy

City	Temperature	Happiness Score
Fremont, CA	9.42	75.32
Plano, TX	18.58	73.23
San Jose, CA	14.67	73.2
Irvine, CA	9.08	72.81
Madison, WI	9	72.19
Sioux Falls, SD	9.83	70.26
Huntington Beach, CA	16.42	69.9
Scottsdale, AZ	23	69.49
Santa Rosa, CA	9.42	69.33
Pearl City, HI	23.58	69.13
Bismarck, ND	8.33	68.86
Fargo, ND	6.42	68.41
Lincoln, NE	12.5	68.33
San Francisco, CA	5.92	68.3
Overland Park, KS	14	68.23
Santa Clarita, CA	18.25	67.86
Oceanside, CA	16	67.66
Glendale, CA	9.83	67.64
Anaheim, CA	17.5	67.1
Cape Coral, FL	22.42	66.96
Pembroke Pines, FL	23.25	66.8
Austin, TX	20.42	66.54
Gilbert, AZ	22.92	66.37
San Diego, CA	16.83	66.34
Chula Vista, CA	16.5	66.21

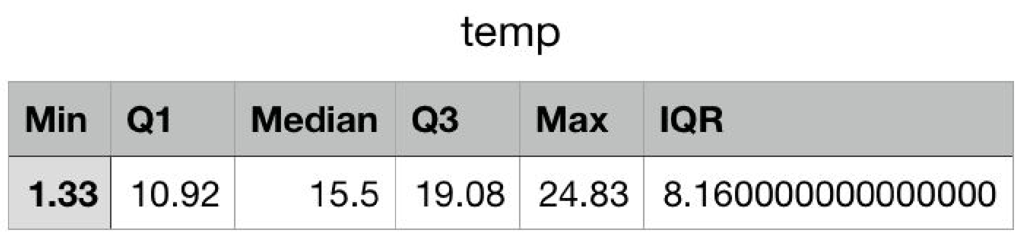
precipHappy.csv



precipHappy

City	Precipitation	Happiness Score
Fremont, CA	105.58	75.32
Plano, TX	82	73.23
San Jose, CA	31.17	73.2
Irvine, CA	27.75	72.81
Madison, WI	67	72.19
Sioux Falls, SD	52.17	70.26
Huntington Beach, CA	24	69.9
Scottsdale, AZ	19.67	69.49
Santa Rosa, CA	107.58	69.33
Pearl City, HI	71.92	69.13
Bismarck, ND	34.92	68.86
Fargo, ND	44.17	68.41
Lincoln, NE	62.33	68.33
San Francisco, CA	80.67	68.3
Overland Park, KS	80.92	68.23
Santa Clarita, CA	35.75	67.86
Oceanside, CA	23.42	67.66
Glendale, CA	37.08	67.64
Anaheim, CA	27	67.1
Cape Coral, FL	110.08	66.96
Pembroke Pines, FL	124.25	66.8
Austin, TX	67.75	66.54
Gilbert, AZ	17.75	66.37
San Diego, CA	22.75	66.34
Chula Vista, CA	19.75	66.21

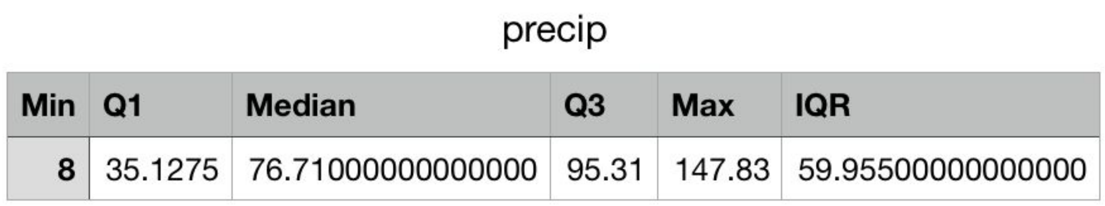
temp.csv



temp

Min	Q1	Median	Q3	Max	IQR
1.33	10.92	15.5	19.08	24.83	8.160000000000000

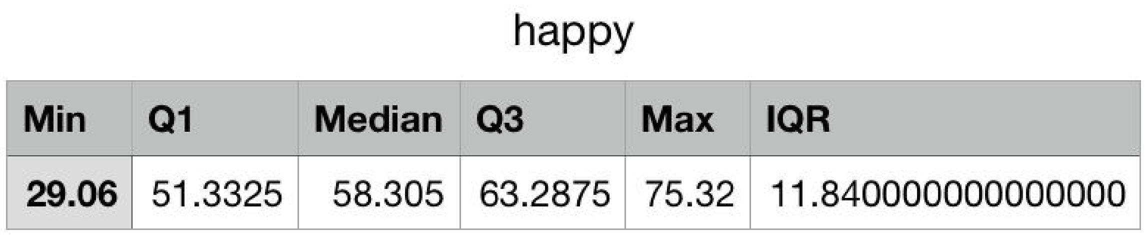
precip.csv



precip

Min	Q1	Median	Q3	Max	IQR
8	35.1275	76.71000000000000	95.31	147.83	59.95500000000000

happy.csv

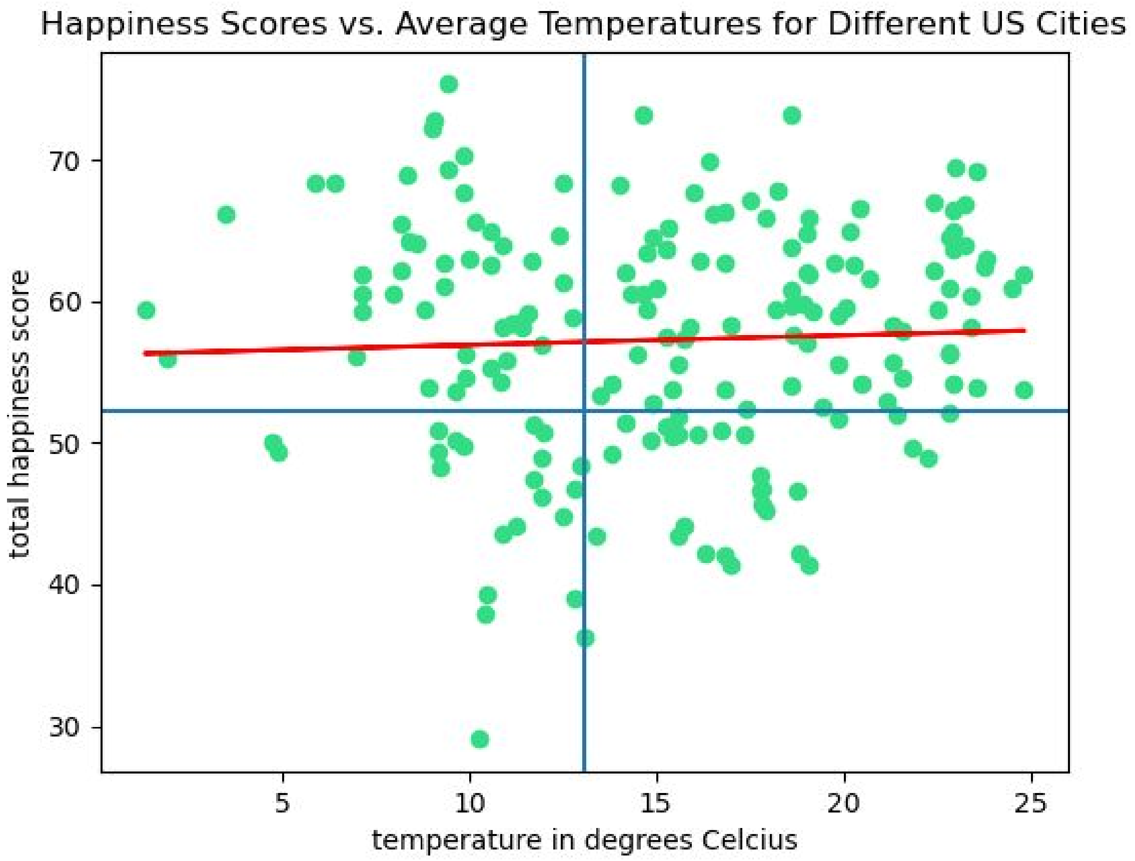


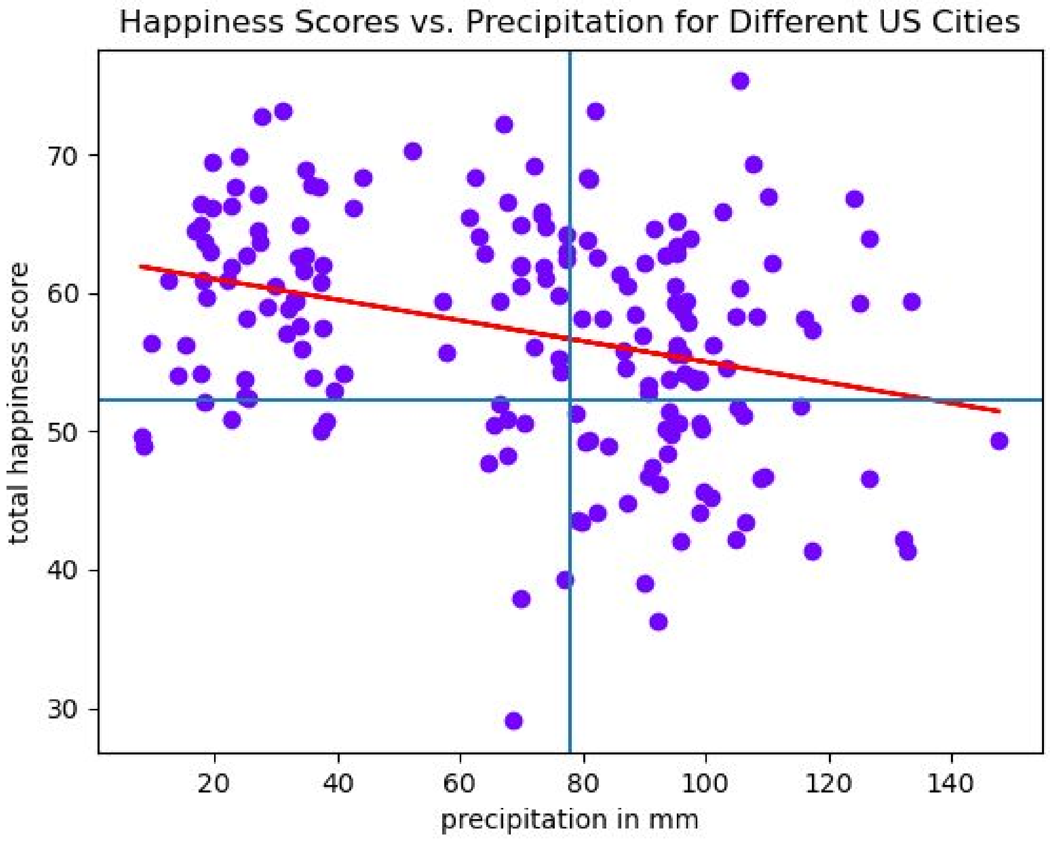
happy

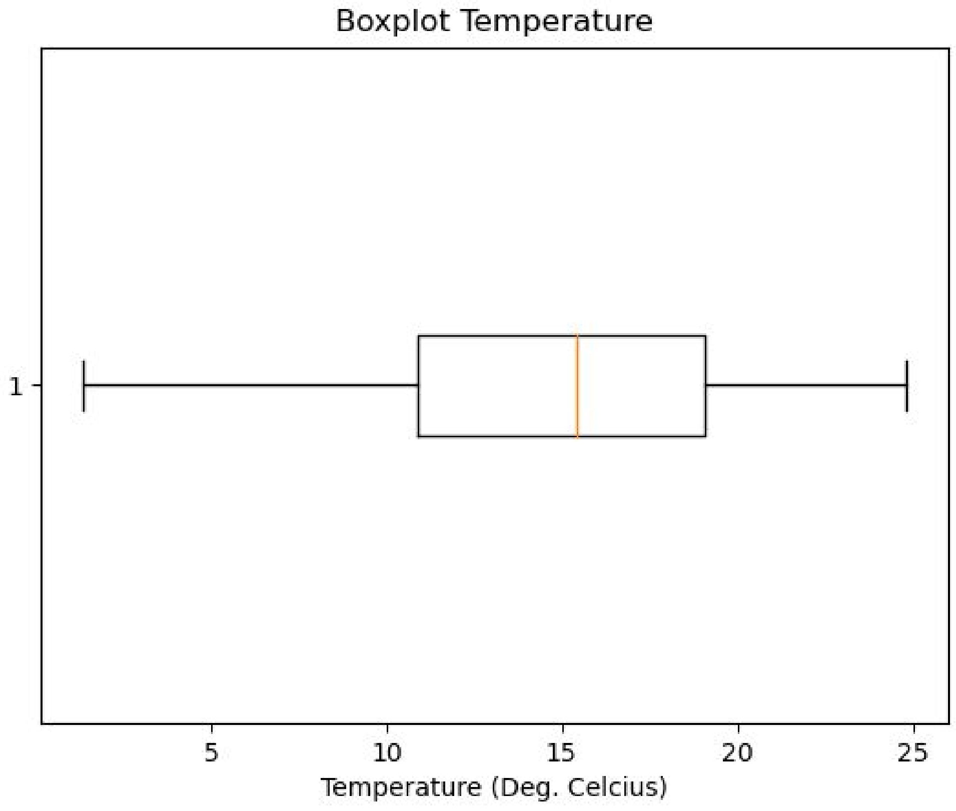
Min	Q1	Median	Q3	Max	IQR
29.06	51.3325	58.305	63.2875	75.32	11.840000000000000

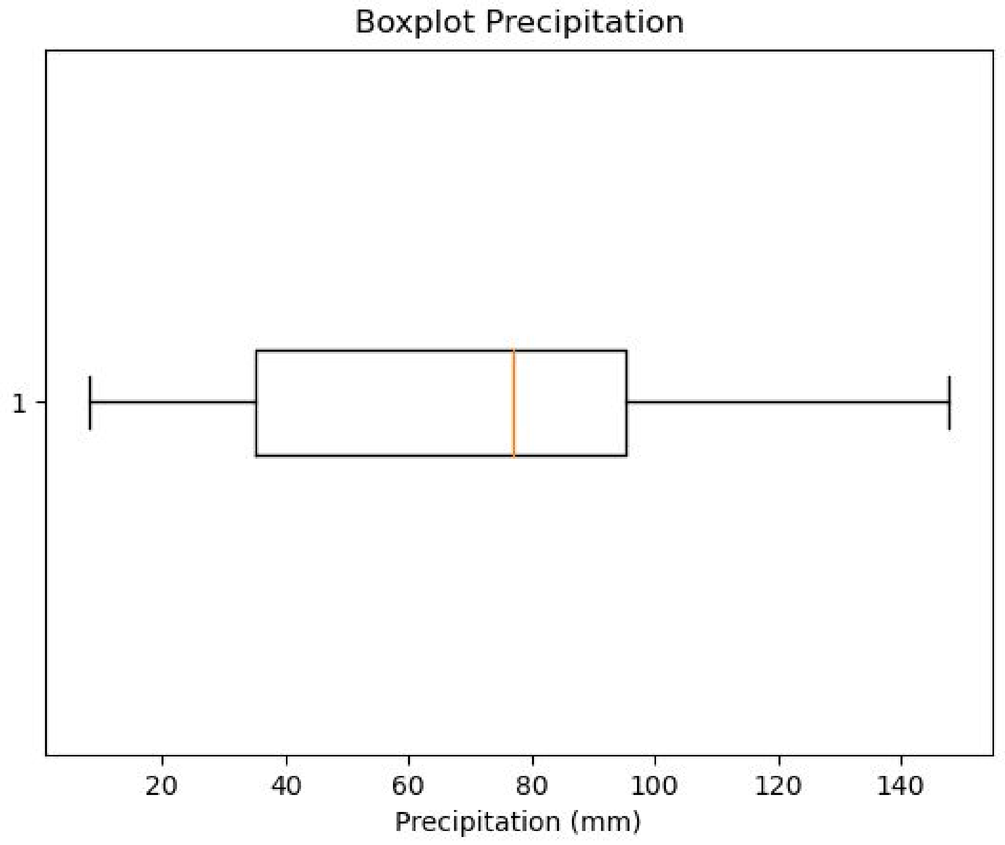
The visualization that you created
The following 5 images are the visualizations that we created. We made two scatterplots comparing each city's average happiness score to its average temperature and average precipitation. Then, we found the line of best fit for each to view the correlation between

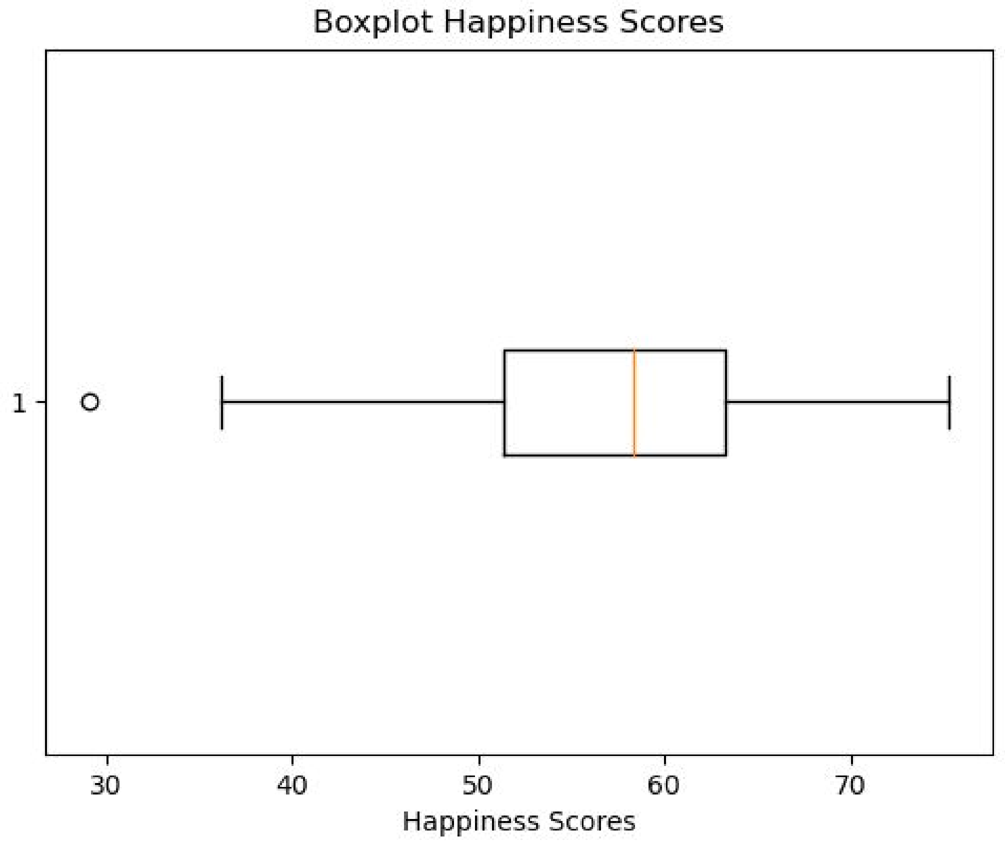
happiness and temperatures and happiness and precipitation for these cities. The next three visualizations are boxplots of data regarding happiness score, average temperature, and average precipitation. These visualizations can be found in the zipped project folder.




Happiness Scores vs. Precipitation for Different US Cities
total happiness score
precipitation in mm
30
40
50
60
70
20
40
60
80
100
120
140


Boxplot Temperature
1
5
10
15
20
25
Temperature (Deg. Celcius)


Boxplot Precipitation
1
20
40
60
80
100
120
140
Precipitation (mm)


Boxplot Happiness Scores
1
30
40
50
60
70
Happiness Scores

Conclusions:

Heading into the project, we thought there would be a weak to minimum positive correlation of the happiness score and average temperature. That is, we believed that cities with warmer average temperatures would report higher average happiness scores, and cities with colder average temperatures would report lower average happiness scores. We were surprised to see that there was no trend when plotting these values in ‘tempHappyScatterplot.png’, as the correlation coefficient was 0.04.

For precipitation, we expected a weak negative correlation of happiness score and average precipitation. That is, we believed that cities with higher average precipitation measures would report lower average happiness scores, and that cities with lower average precipitation measures would report higher average happiness scores. Our belief was confirmed in our analysis, as you can see a weak negative correlation between these two datasets in ‘precipHappyScatterplot.png’. The correlation for this figure was -0.30.

Instructions for running your code

1. Open up the zipped file. Make sure the only things in the project folder are files that end in “.py”, the README, the pdf of the Report, and the two folders 'calculations' and 'visualizations', both of which should be empty before beginning to run the code. The only files needed to start running the code are the python files. There should be no database, images, csv files, or anything else not mentioned above in the folder. Make sure they are deleted if they are in there before running the code.
2. Make sure anaconda is fully installed on your computer. This will ensure that python is fully up to date as well as matplotlib. Also, make sure SQLite Database Browser Portable is installed so that you can view the database once you create it. The API and website do not need to be installed before running the code.
3. Run the file “final.py”. This should be run 7-8 times. It may take a while each time you run it because a lot of data is being extracted every time. You will know everything is finished running once all 5 graphs have come up on your screen and the correlations have been displayed in the terminal. When you begin running the code, it will create a database called “finalProjectDatabase.db” and will create the tables “Temperatures”, “WeatherData", “HappyData”, and “LatLongData” within the database. You will now be able to see the finalProjectDatabase.db file in the project folder. With SQLite Database Browser Portable installed, you will be able to view the tables within the database. At the

end, there should be 182 rows in all of the tables except the Temperatures table. This one will only have 168 rows.

4. Next, the “cal_vis.py” file will run automatically once all of the data has been collected from the “final.py” file. This file will be called in the last run of "final.py".The “calc_vis.py” file will do the calculations from the data in the database and create the visualizations for the calculations. The calculations will all be saved within the 'calculations' folder in the project folder. The visualizations will all be saved within the 'visualizations' folder in the project folder. You can view the calculations in the csv files within the 'calculations' folder after everything is done running. They will be titled “tempHappy.csv” for the temperature vs. happiness scores data, “precipHappy.csv” for the precipitation vs. happiness scores data, “temp.csv” for the temperature boxplot data, “precip.csv” for the precipitation boxplot data, and “happy.csv” for the happiness scores boxplot data. You can view the visualizations in the png files within the 'visualizations' folder after everything is done running as well. The visualizations will be labelled “tempHappyScatterplot.png”, “precipHappyScatterplot.png”, “tempBoxplot.png”, “precipBoxplot.png”, and “happyBoxplot.png” corresponding to the data in the png files respectively. Plus, when in the 7th run of the project, each visualization will display on your screen once you get to it in the code. You must close out of the image in order to allow the file to continue to run. Once the file is finished running, you will have all of these images and csv files in the folders in the overall project folder and the code will also output the correlation coefficients for the first two visualizations within the terminal.
5. In the end, you can now open finalProjectDatabase.db, any of the csv files, and any of the png images to view the database tables, calculations, and visualizations.

Documentation for each function that you wrote (including input and output for each function)

final.py:

```
- def create_connection(database):
-     '''This function takes in the name of the database. It makes a
  connection to the cursor using the name of the database given. It
  returns the connection variable and cursor variable to allow access
  into the database.'''
- def lat_long_table(cur, conn, lat, long, place):
'''This function takes in the cursor, connection, latitude,
  longitude, and name for a specific city. It creates a
  Latitude/Longitude table in the database and inserts the city,
  latitude, and longitude. It returns nothing. '''

def get_weather_data(cur, conn, location):
'''This function takes in the cursor, connection, and city name.
  First request the geocoding API and include the city name in the
  request. Use json to extract the latitude and longitude from this.
  Then call the lat_long_table function to create the
  latitude/longitude table. Now, call the weather API in order to get
  a json file of the weather information per month. Add each month's
  data to a dictionary (key = month number, value = tuple of
  temperature and precipitation for that month). Return that
  dictionary. '''

def weather_table(data, cur, conn, location, temp_list):
'''This function takes in a dictionary of the weather data, the
  cursor, connection, city name, and list of already seen temperature
  values in the data dictionary. This creates a table for the Weather
  Data and then gets the temperature and precipitation from the data
  dictionary for each month. It takes the average of these 12 months
  and inserts the id for the temperature into the table and the value
  for precipitation into the table along with the city name
  (location). If a new temperature value was found it adds it to the
  temp_list and creates a new id in the Temperatures table. Otherwise,
  it uses an existing id from that table to describe the average
  temperature for the current location being used. It commits to the
  database and returns nothing.'''

def website_prep():
'''This function takes in no parameters. It preps the website
  data to be extracted easily later. It makes a request using the
  happiest places to live url and then uses beautiful soup to find the
  table that we will extract data from in a later function. It returns
  a list of all of the values within the table.'''

def get_website_data(i, row_tags):
```

```python
    '''This function takes in the index for the current city being
looked at and the row_tags data list of the table information from
the website that was found in the website_prep function. It looks at
the specific row (i) in the list (row_tags) to find the overall
rank, city name, total happiness score, emotional/physical score,
income/employment score, and community/environment score. It then
adds these 6 values to a list and returns that list. '''

def website_table(data, cur, conn):
    '''This function takes in the list of data (the 6 values) from
the website, the cursor, and connection. It creates a table for the
website data (HappyData) if one does not already exist. It then
inserts the data into each column of the table. It commits the
values to the database and returns nothing. '''

def get_start_index(cur, conn):
    '''This function takes in the cursor and connection. This
function finds where to start gathering data from in the next round
of data extraction. It creates the HappyData table if it does not
exist already. It then finds all of the cities in the table
currently. If that length is 0, then it returns 0 (we are in the
first round of data extraction from the API/website. If it is
between 1 and 181, then we return the next value to be extracted (if
we ended at 25 extractions, we return 26 to start at the next one).
If we are out of this range, then we are done collecting data and
return -1.'''

def get_temp_lists(cur):
    '''This function takes in the cursor. It creates the
Temperatures table if it does not already exist and selects the
temperature values from it. It then adds them all to a list and
returns that list. This is meant to be used for figuring out the id
of temperatures that will be extracted from the API and added to the
WeatherData table. '''

def main():
    '''The main function takes in no parameters. It calls the
create_connection function on the database name being used. It then
```

```
creates start and stop variables to make sure we don't extract more
than 25 things from the APIs and website. Then it calls website_prep
and get_temps_list. Then, it does a while loop that continues until
around 25 things have been added to the tables. In the loop, we call
get_start_index to get the starting point for data extraction. We
make sure if it is -1, we do the calculations (see other python
file) and then end the running of the code. Then, we call
get_website_data, get_weather_data, weather_table, and website_table
to get the data and add it to the database. Then we add one to our
starting value for counting how many times we have done a data
extraction, and we loop again. This function returns nothing. '''
```

calcs_vis.py:

```
- def setUpDatabase(db_name):
-     '''This function takes in the name of the database and creates a
connection and cursor to it. It returns the cursor and
connection.'''
-
- def calculation(cur):
-     '''This function takes in the cursor as input. Uses SQL
SELECT...FROM...JOIN statement to find where temperature that
results from joining the WeatherData and Temperature tables
together. This list of temperatures is extracted with the fetchall()
command. The uses SELECT statement to extract a list of the
happiness rating scores from the HappyData table. Then performs a
SELECT statement to extract a list of average precipitation figures
from the WeatherData table. Finally, extracts a list of city names
using a SELECT statement to extract city names from the HappyData
table. Returns a tuple of lists for temperature, happiness scores,
precipitation, and city names.'''
-
- def write_csv(x, y, names_city, file_name, headers):
-     '''This function takes in x and y axis data as lists, a list of
city names, the csv file to write to, and the headers to include at
the top of the csv file. Writes to the csv file all of the
information passed in from the x data, y data and city lists and
does not return anything.'''
-
```

```python
def visualization1(temp, happy, city_names):
    '''This function takes in a list of temperatures, a list of
happiness scores, and a list of city names as inputs. Calls the
write_csv function to write to the csv the data from these lists,
and then plots the data on a matplotlib plot. The figure is saved as
'v1.png' and does not return anything.'''

def visualization2(precip, happy, city_names):
    '''This function takes in a list of precipitation data, a list
of happiness scores, and a list of city names as inputs. Calls the
write_csv function to write to the csv the data from these lists,
and then plots the data on a matplotlib plot. The figure is saved as
'v2.png' and does not return anything.'''

def box_and_wiskers(data, x_label, fig_name, title, csv_name):
    '''This function takes in a list of the data of interest we want
to plot, the x_label for the plot, the name of the figure, title,
and the name of the csv file that is desired to save the data into.
Creates a box-and-whiskers plot, which includes the quartiles for
the data and the iqr as well. It then writes to the csv file the
Min, q1, Median, Q3, Max, and IQR data for the data passed into the
function. The function returns nothing.'''

def main():
    '''The main() function takes in no inputs. It calls the
calculations function to return data for temperature, happiness
scores, precipitation, and city names, and uses these to call the
visualization1() and visualization2() functions. Finally, the
box_and_wiskers() function is called on precipitation, temperatures
and happiness scores individually so we can see the box-and-whiskers
plots for each of these lists of data. Main() returns nothing.'''
```

Documentation of all resources used

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
11/19/20	convert curl statement to http request Python	https://stackoverflow.com/questions/37530378/convert-curl-command-to-http-request-in-python	yes- got this to properly make a request
11/23/20	API we wanted to use costs money	https://github.com/meteostat/dev/blob/main/src/api/point/climate.md	yes- this new API does something similar to the one we originally wanted and is free
11/23/20	We needed a way to get latitude and longitude in order to access our weather API	https://developer.mapquest.com/documentation/geocoding-api/	Yes- we were able to use this to access our weather API
11/24/20	struggled with format for JOIN statement	https://runestone.academy/runestone/assignments/doAssignment?assignment_id=51587	yes- I was able to properly format a JOIN command
11/30/20	trouble with finding the index in a list- I thought I could use .find() but that caused errors	https://www.programiz.com/python-programming/methods/list/index	no- I thought this would do the same as .find() where it returns -1 if it's not in the list but that was not the case
11/30/20	trouble with finding the index in a list- I thought I could use .find() but that caused errors	https://stackoverflow.com/questions/9542738/python-find-in-list	yes- I figured out how to find the index properly
11/30/20	solve merge conflict with pulling from GIT	https://opensource.com/article/20/4/git-merge-conflict	yes- was able to get rid of the current code to pull from the updated repository
12/1/20	quadrants in a matplotlib graph	https://www.youtube.com/watch?v=7BNmf2HxX_I	yes- we figured out how to make quadrants
12/1/20	how to properly do join again	https://www.tutorialspoint.com/python_data_access/python_sqlite_join.htm	sort of- figured it out with some help from here but mostly trial and error
12/1/20	getting hex values for color	https://matplotlib.org/3.3.3/tutorials/colors/colors.html	yes- we got cool colors
12/1/20	figuring out how to do box plots	https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/box-plot-review	yes- we learned what we need for this plot
12/1/20	how to make a boxplot horizontal	https://stackoverflow.com/questions/18500011/horizontal-box-plots-in-matplotlib-pandas	yes- we used this code
12/1/20	how to set up a boxplot in	https://matplotlib.org/3.1.1/gall	yes- this helped us format

	matplotlib	ery/pyplots/boxplot_demo_pyplot.html	our code
12/1/20	how to get IQR from matplotlib	https://stackoverflow.com/questions/62330801/how-to-get-min-and-max-values-from-boxplot-in-python	yes- we used this code
12/1/20	how to find the 5 number summary from our boxplots	https://machinelearningmastery.com/how-to-calculate-the-5-number-summary-for-your-data-in-python/	yes- this helped us format our code
12/8/20	How to save visualizations and calculations to proper folder in project folder	https://stackoverflow.com/questions/11373610/save-matplotlib-file-to-a-directory	Yes- helped get me started on how to solve this problem and eventually I was able to