SI 206 Final Project Report

Jordan Shefman
Mark Qian
GitHub Link: https://github.com/jshef16/SI-206-Final-Project.git

Project Goals

For our project, it was important to us that we were able to demonstrate all of the skills that we have learned over the course of the semester. While doing so, we aimed to learn about an interesting topic that we were both passionate about. The goal for this project was to check all of the boxes on the rubric. Additionally, it was a good experience to build a full project from scratch, rather than being presented with some starter code.

Goals Achieved

We believe that we have achieved all of these goals. We were able to build a report about different E-Bike companies and which countries in the world that they are located in. Throughout the process, we were able to take advantage of the CityBike API as well as using BeautifulSoup to obtain data from *The Most Bicycle-Friendly Cities of 2019* likes from Copenhagenize. After obtaining the raw data, we were able to present it in a few different ways, including CSVs and graphical visualization. Most importantly, we have become comfortable with receiving an assignment and planning out every step. This is a very important skill to have in the industry, when you are not given starter code.

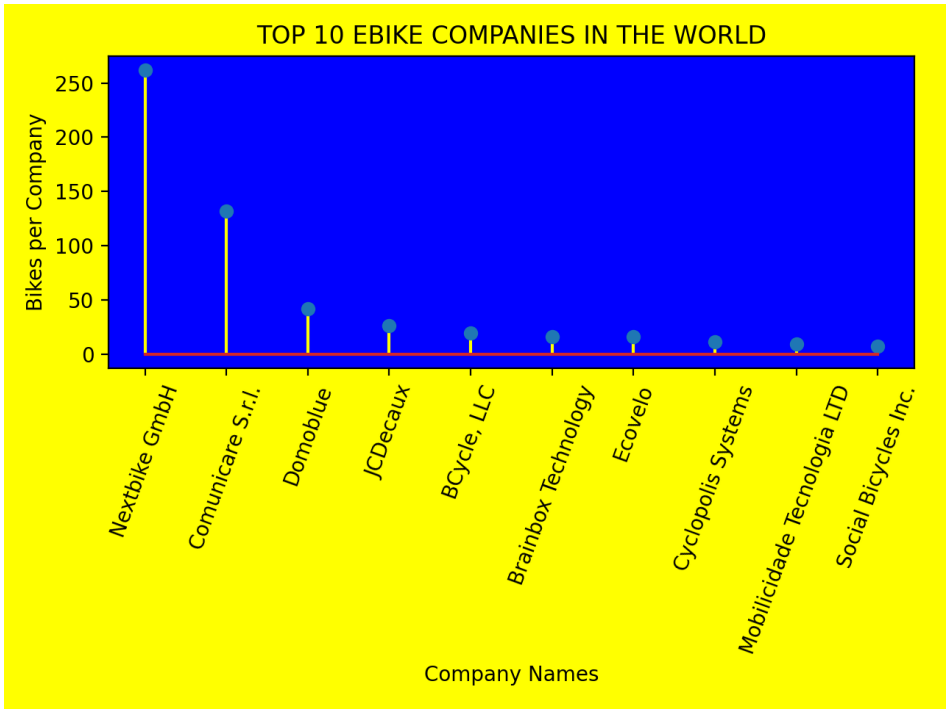
Problems Faced

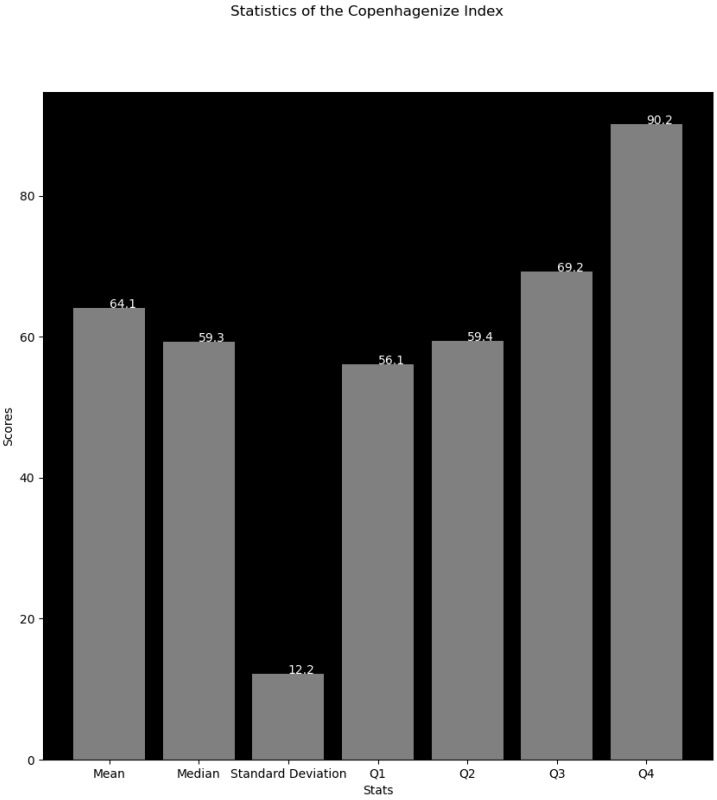
The biggest problem that we faced was our conflicting schedules. It was difficult for us to find time to work together, so we had to collaborate remotely. We created miniature deadlines in order to stay on track and finish on time. The other problem we faced was with MatPlotLib. We found it difficult to come up with plots that would represent our data, not copy each other, and still remain different from the plots that were presented to us in class. However, we were able to overcome both of these problems.

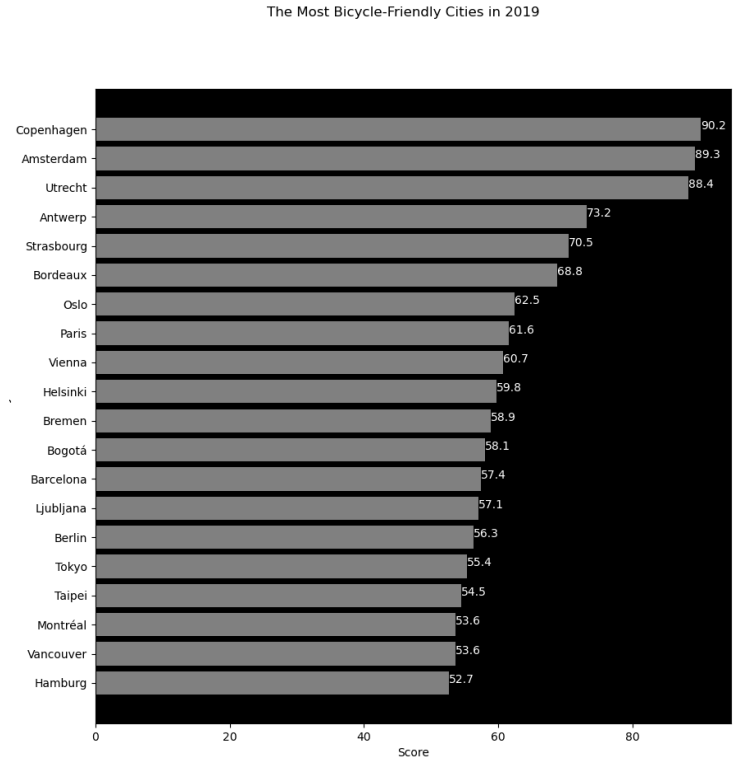
Output File Sample

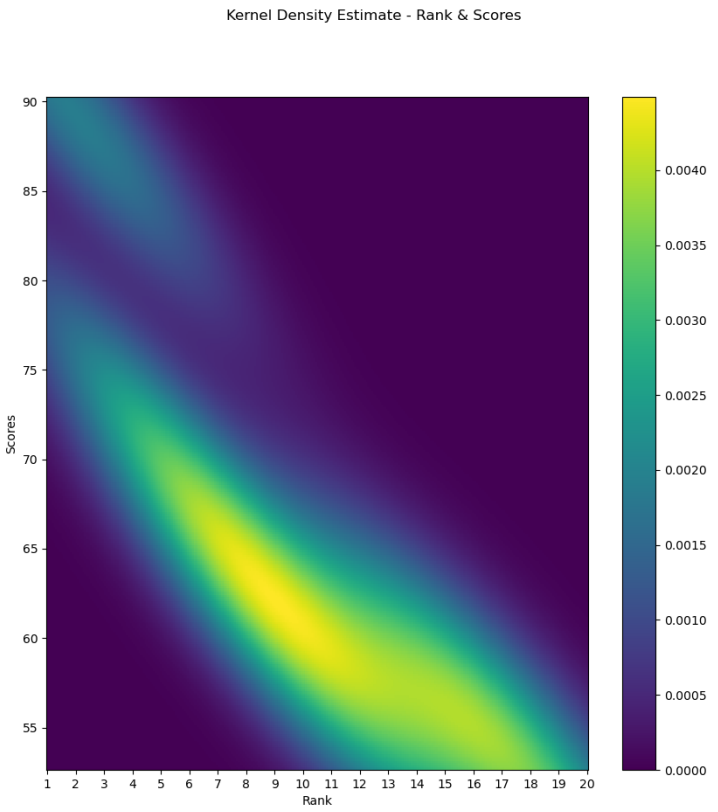
```
CITY,COUNTRY,NUMBER OF BIKES
Moscow,RU,1
Antequera,ES,1
Igoumenitsa,GR,1
Lecce,IT,1
Ioannina,GR,1
Karditsa,GR,1
Manerba del Garda,IT,1
Kavala,GR,1
Kinouria,GR,1
Komotini,GR,1
Patra,GR,1
Marciana Marina,IT,1
Aranda de Duero,ES,1
Rethymno,GR,1
Samos,GR,1
Novara,IT,1
Nicosia,CY,2
Mani,GR,1
Irakleio,GR,1
Thessaloniki,GR,1
Athens,GR,1
Padova,IT,1
São Paulo,BR,2
Aranjuez,ES,1
A Coruña,ES,1
Ferrol – Narón,ES,1
Curtin University, Perth, WA,
Melbourne, AU,AU,1
Ferrara,IT,1
Montréal, QC,CA,1
Parma,IT,1
Toronto, ON,CA,1
Hamilton, ON,CA,1
Ottawa, ON,CA,1
Badajoz,ES,1
Ann Arbor, MI,US,1
Pavia,IT,1
Perugia,IT,1
Austin, TX,US,1
Chattanooga, TN,US,1
Piacenza,IT,1
Pinerolo,IT,1
Portland, OR,US,1
Reggio Emilia,IT,1
San Ramon, CA,US,1
Baeza,ES,1
Boise, ID,US,1
Boulder, CO,US,1
Santa Monica, CA,US,1
Rimini,IT,1
Fort Lauderdale, FL,US,1
Milwaukee, WI,US,1
Buffalo, NY,US,1
Washington, DC,US,1
Charlotte, NC,US,1
Saluzzo,IT,1
Benidorm,ES,2
Cincinnati, OH,US,1
Savigliano,IT,1
New York, NY,US,1
Tampa, FL,US,1
Terni,IT,1
Columbus, OH,US,1
Denver, CO,US,1
Chicago, IL,US,1
Tirano,IT,1
El Paso, TX,US,1
Blanca,ES,1
Fort Worth, TX,US,1
Udine,IT,1
Fargo, ND,US,1
Salt Lake City,US,1
Phoenix, AZ,US,1
Omaha, NE,US,1
Houston, TX,US,1
Boston, MA,US,1
Ustica,IT,1
Philadelphia, PA,US,1
Cieza,ES,1
Indianapolis, IN,US,1
Orlando, FL,US,1
Como,IT,1
Kansas City, MO,US,1
Dayton, OH,US,1
Madison, WI,US,1
Los Angeles, CA,US,1
Ketchum / Sun Valley, ID,US,1
Minneapolis, MN,US,1
La Spezia,IT,1
Don Benito – Villanueva,ES,1
San Antonio, TX,US,1
```

Visualizations









Instructions for Running the Code

To run the code, simply press run. The first several times the code is run, the data will not be accurate, because only 25 lines of data can be added to the database at a time. There were over 600 total lines of data, so only after running the program enough times to get it all in the database will the output be accurate. The only other thing to do while the code is being run is to exit the visualization to allow the rest of the code to execute.

Function Documentation

getData()

Input: none

Output: sorted list of best bike friendly cities according to copenhagenize index

Action: uses beautiful soup to scrape data and prints the sorted list

setUpDatabase(db_name)

Input: db_name: string

Output: cur, conn

Action: creates a database with the given name

createBikesTable(cur, conn)

Input: cur, conn
Output: none
Action: creates a bikes table if one does not already exist

createCitiesTable(cur, conn)

Input: cur, conn
Output: none
Action: creates a cities table if one does not already exist

createCountriesTable(cur, conn)

Input: cur, conn
Output: none
Action: creates a countries table if one does not already exist

createCompaniesTable(cur, conn)

Input: cur, conn
Output: none
Action: creates a companies table if one does not already exist

createNamesTable(cur, conn)

Input: cur, conn
Output: none
Action: creates a names table if one does not already exist

readAPI()

Input: none
Output: results: dict
Action: makes a request to the city bikes API. if request is OK, returns results as a dict. If not OK, prints error message and returns none

addBikes(cur, conn, bikes_dict)

Input: cur, conn, bikes_dict
Output: none
Action: adds bikes to the bike table 25 at a time. Looks up city, country, etc in other tables to get their ids, so bike table does not store repeating strings

addCities(cur, conn, bikes_dict)

Input: cur, conn, bikes_dict
Output: none

Action: adds cities and ids to their table

addCountries(cur, conn, bikes_dict)
Input: cur, conn, bikes_dict
Output: none
Action: adds countries and ids to their table

addCompanies(cur, conn, bikes_dict)
Input: cur, conn, bikes_dict
Output: none
Action: adds companies and ids to their table

addNames(cur, conn, bikes_dict)
Input: cur, conn, bikes_dict
Output: none
Action: adds names and ids to their table

getCounts(cur, conn)
Input: cur, conn
Output: res: list of tuples (city, country, count)
Action: computes the counts of of bikes per city

getMaxCount(count_list)
Input: count_list
Output: (max, max_city)
Action: finds the city with the most bikes and outputs as a tuple of the count and the city

bikesByCompany(cur, conn)
Input: cur, conn
Output: matplotlib visualization
Action: computes the number of bikes that each company has in operation and creates a lollipop chart of the top ten companies with the most bikes

get_data()
Input: nothing
Output: city_list
Action: retrieves the data we want from the website and puts them in a tuple list

city_lst(city_list)
Input: city_list

Output: cities
Action: splits the cities from the tuple with their rank and score and puts them in their own list

score_lst(city_list)
Input: city_list
Output: scores
Action: splits the scores from the tuple with their rank and city and puts them in their own list

rank_lst(city_list)
Input: city_list
Output: ranks
Action: splits the ranks from the tuple with their city and score and puts them in their own list

stats(scores)
Input: scores
Output: stats
Action: reads in the scores and computes mean, median, standard deviation, Q1, Q2, Q3, and 100th quantile

city_score_visual(cities, scores)
Input: cities: list, scores: list
Output: matplotlib visualization
Action: creates a horizontal bar chart that demonstrates the scores that each city has received on their bike friendliness

stats_visual(scores_stats)
Input: scores_stats: list
Output: matplotlib visualization
Action: creates a bar chart that demonstrates the stats that were computed in stats()

kernal_estimate_visual(ranks, scores)
Input: ranks: list, scores: list
Output: matplotlib visualization
Action: creates a kernel density estimate plotwith a color bar that demonstrates the relationship between the rank and score

Resources Used

Date	Issue Description	Location of Resource	Result
3/28/22	Needed website to scrape data from	https://copenhagenizeindex.eu	Solved
2/28/22	Needed an API	http://api.citybik.es/v2/	Solved
4/4/22	Needed to find visualization technique not explicitly used in class	https://seaborn.pydata.org/introduction.html	Not solved
4/4/22	Needed to find visualization technique not explicitly used in class	https://matplotlib.org/stable/gallery/index	Solved