ARCHITECTURE:

W205-Final-Project/

- **setup_scripts/**
  - aggregates.py
  - app.py
  - calculate_score.sql
  - chi_aggregates.py
  - create_tables.sql
  - explore_data.sh
  - format_bike_parking_data.sh
  - requirements.txt
  - setup_env.sh
  - setup_hive.py
  - upload_roiana_scores.sql
  - upload_scores.sh
  - upload_scores.sql
- **data/**
  - chi_bike_data.csv
  - chi_parking_scores.csv
  - score_output_f.csv
  - sf_bike_data.csv
  - sf_parking_scores.csv

W205-Final-Project
Github repository containing the code used to obtain, clean, explore the data and implement the solution for our final project. The repository also contains the final presentation and serving layer for the project.

data/
Folder which contains some of the data used in the analysis.
Four total raw datasets used in this project: 1) Bike rack data for Chicago, 2) Crime data for Chicago, 3) Bike rack data for San Francisco, 4) Crime data for San Francisco

chi_bike_data.csv: Bike parking installment data for Chicago.

chi_parking_scores.csv: Counts of crimes within a 0.1 mile radius of Chicago bike parking locations.

score_output_f.csv: File containing the risk score for each location, with 0 being the safest and 10 being the most dangerous.

sf_bike_data.csv: Bike parking installment data for San Francisco.

sf_parking_scores.csv: Counts of crimes within a 0.1 mile radius of San Francisco bike parking locations.

setup_scripts/

Folder containing the shell, python, SQL scripts which were used to set up the EC2 environment, clean, load and format the data, perform analysis and create the serving layer for our “Park My Bike” application.

Setting up the programming environment

requirements.txt:

setup_env.sh: Installs Hadoop, Spark SQL, Postgres using scripts provided by the UCB W205 course. Also sets up the virtual environment and installs the necessary python packages needed to implement the solution.

setup_hive.py:

Data downloading, cleaning and exploration

create_tables.sql: Script used to explore crime dataset and identify various trends within the data.

explore_data.sh: Code used to download the Chicago and San Francisco crime datasets and load data into the EC2 instance.

format_bike_parking_data.sh: Cleans up the San Francisco bike parking data. Splits the location column into two columns, one which contains the longitude coordinates and another which contains the latitude coordinates.

Analytical component, Merging of data

aggregates.py: Creates schema for loading the San Francisco crime dataset and the bike parking installments dataset. Builds table for areas within a 0.1mile radius of the bike parking locations. Uses a Cartesian production to join the crime data table with the table containing the boundaries for places within a certain distance of the bike parking locations. Counts the total number of crimes near bike parking locations and outputs data to csv file.

chi_aggregates.py: Creates schema for loading the Chicago crime dataset and the bike parking installments datasets Builds table for areas within a 0.1mile radius of the bike parking locations. Uses a Cartesian production to join the crime data table with the table containing the boundaries places within a certain distance of the bike parking locations. Counts the total number of crimes near bike parking locations and outputs data to csv file.

calculate_score.sql: Schema for creating table from csv file which contains the number of crimes near bike parking locations. These queries rank each location by number of crimes which have occurred within the vicinity and creates a score based on these rankings.

Serving Component with Real Time Processing

app.py: Application for receiving user inputted location, obtaining nearest bike parking locations, distance and resulting scores. This script serves these results to the resting API.

upload_roiana_scores.sql

upload_scores.sh

upload_scores.sql