SI 206 Final Project Report

https://github.com/ericlzg/SI206Final.git

I. TOPIC

- Shared Fleet Devices, Bus Stops, and Census Tract of Washington DC

II. GROUP INFORMATION

A. **Team:** "Not another bus project"
B. **Group Members:** Mongkoloudom Chhiv, Eric Li, Sean Kim

III. GOALS

A. **Original Goal:** Originally we planned on pulling data from 3 apis: Census data, bike and scooter data, and bus stop data in order to layer them in a geographic map and visualize distance of shared fleet
B. **Achieved Goal:** We created visualizations of the shared fleet, bus stops, and census tracts. However, those graphs alone were not enough for us to get a proper judgment of the distance between the objects. Therefore, through our calculation, we calculated the distance between a sharefleet to a census tract, to potentially see some patterns when we do take the calculation file into a deeper data analysis effort.

IV. PROBLEMS

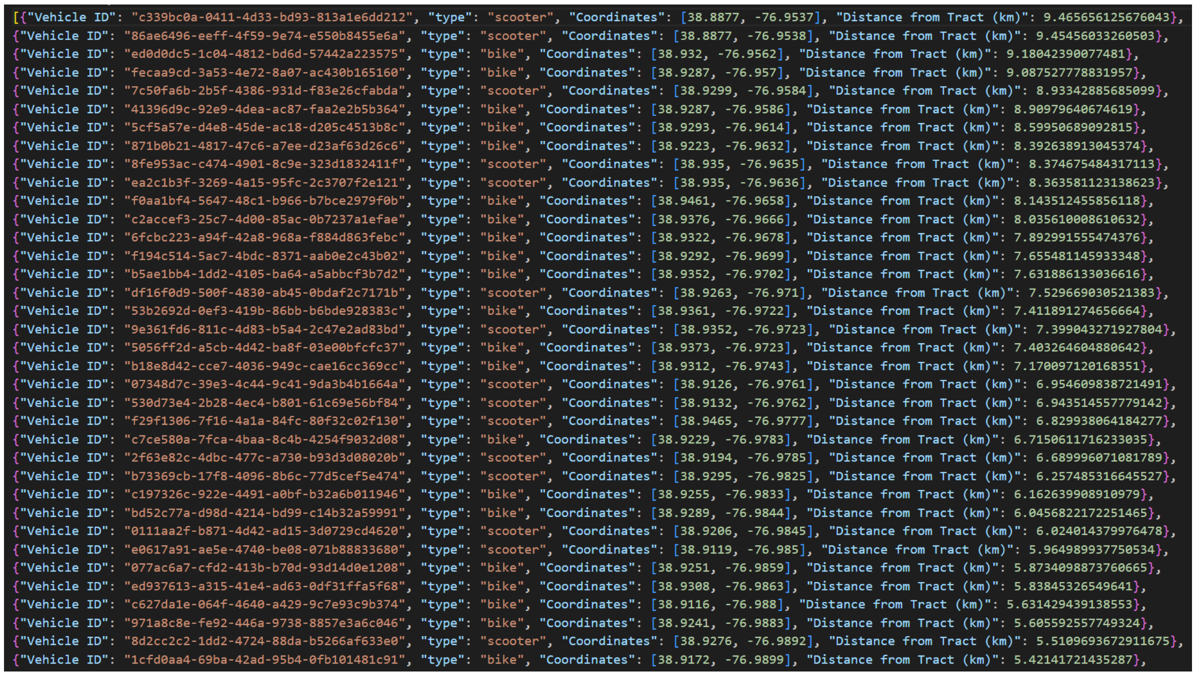
A. **Trying to map out the boundary of Washington DC**
   - It was difficult to locate data on the boundary
   - When the data was found, there were multiple formats with unclear documentation
   - The main issue came from running file on wrong directory leads to file not being found

B. **Trying to figure out what calculations to do**

- We knew we could do something with lat and lon to measure distance, but we got overwhelmed with deciding whether we should do with each individual bus stop or vehicle or census tract

V. CALCULATION FILE

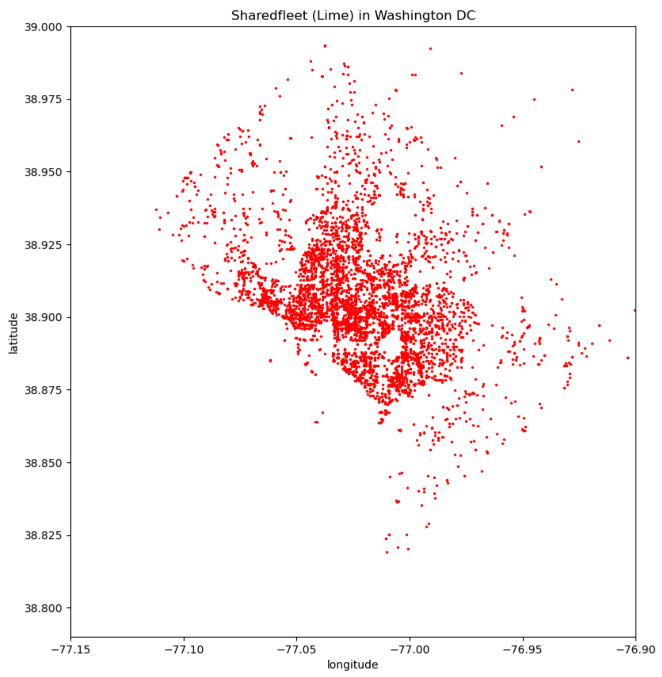
- **This file contains data on shared fleet devices from Vehicles table that join the bike type information from Types table, while calculating distance from the vehicle to a census track using their respective lat lon**



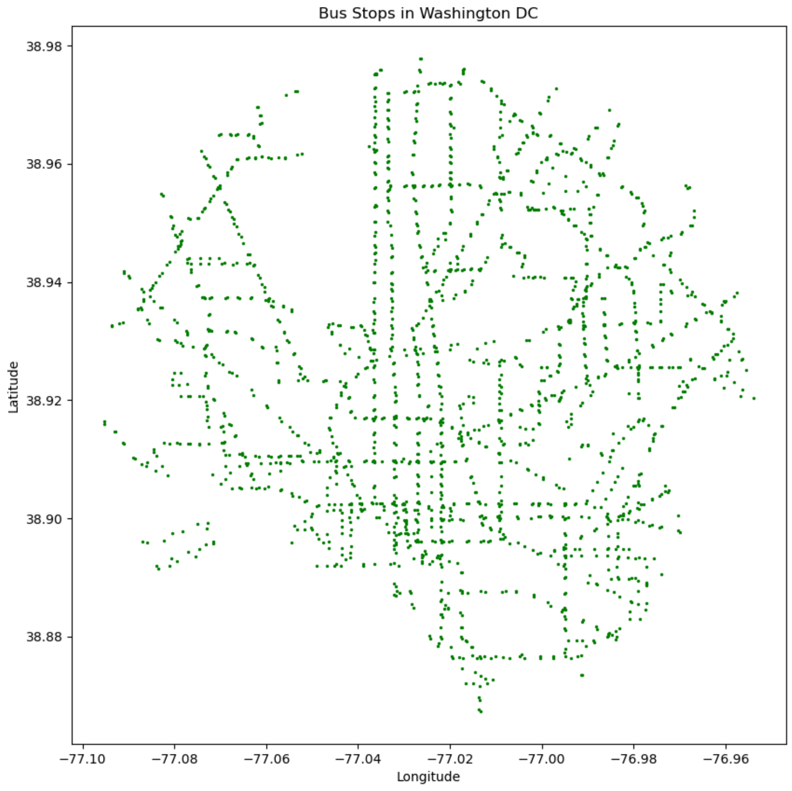
```
[{"Vehicle ID": "c339bc0a-0411-4d33-bd93-813a1e6dd212", "type": "scooter", "Coordinates": [38.8877, -76.9537], "Distance from Tract (km)": 9.465656125676043},
{"Vehicle ID": "86ae6496-eeff-4f59-9e74-e550b8455e6a", "type": "scooter", "Coordinates": [38.8877, -76.9538], "Distance from Tract (km)": 9.45456033260503},
{"Vehicle ID": "ed0d0dc5-1c04-4812-bd6d-57442a223575", "type": "bike", "Coordinates": [38.932, -76.9562], "Distance from Tract (km)": 9.18042390077481},
{"Vehicle ID": "fecaa9cd-3a53-4e72-8a07-ac430b165160", "type": "bike", "Coordinates": [38.9287, -76.957], "Distance from Tract (km)": 9.087527778831957},
{"Vehicle ID": "7c50fa6b-2b5f-4386-931d-f83e26cfabda", "type": "scooter", "Coordinates": [38.9299, -76.9584], "Distance from Tract (km)": 8.93342885685099},
{"Vehicle ID": "41396d9c-92e9-4dea-ac87-faa2e2b5b364", "type": "bike", "Coordinates": [38.9287, -76.9586], "Distance from Tract (km)": 8.90979640674619},
{"Vehicle ID": "5cf5a57e-d4e8-45de-ac18-d205c4513b8c", "type": "bike", "Coordinates": [38.9293, -76.9614], "Distance from Tract (km)": 8.59950689092815},
{"Vehicle ID": "871b0b21-4817-47c6-a7ee-d23af63d26c6", "type": "bike", "Coordinates": [38.9223, -76.9632], "Distance from Tract (km)": 8.392638913045374},
{"Vehicle ID": "8fe953ac-c474-4901-8c9e-323d1832411f", "type": "scooter", "Coordinates": [38.935, -76.9635], "Distance from Tract (km)": 8.374675484317113},
{"Vehicle ID": "ea2c1b3f-3269-4a15-95fc-2c3707f2e121", "type": "scooter", "Coordinates": [38.935, -76.9636], "Distance from Tract (km)": 8.363581123138623},
{"Vehicle ID": "f0aa1bf4-5647-48c1-b966-b7bce2979f0b", "type": "bike", "Coordinates": [38.9461, -76.9658], "Distance from Tract (km)": 8.143512455856118},
{"Vehicle ID": "c2accef3-25c7-4d00-85ac-0b7237a1efae", "type": "bike", "Coordinates": [38.9376, -76.9666], "Distance from Tract (km)": 8.035610008610632},
{"Vehicle ID": "6fcbc223-a94f-42a8-968a-f884d863febc", "type": "bike", "Coordinates": [38.9322, -76.9678], "Distance from Tract (km)": 7.892991555474376},
{"Vehicle ID": "f194c514-5ac7-4bdc-8371-aab0e2c43b02", "type": "bike", "Coordinates": [38.9292, -76.9699], "Distance from Tract (km)": 7.655481145933348},
{"Vehicle ID": "b5ae1bb4-1dd2-4105-ba64-a5abbcf3b7d2", "type": "bike", "Coordinates": [38.9352, -76.9702], "Distance from Tract (km)": 7.631886133036616},
{"Vehicle ID": "df16f0d9-500f-4830-ab45-0bdaf2c7171b", "type": "scooter", "Coordinates": [38.9263, -76.971], "Distance from Tract (km)": 7.529669030521383},
{"Vehicle ID": "53b2692d-0ef3-419b-86bb-b6bde928383c", "type": "bike", "Coordinates": [38.9361, -76.9722], "Distance from Tract (km)": 7.411891274656664},
{"Vehicle ID": "9e361fd6-811c-4d83-b5a4-2c47e2ad83bd", "type": "scooter", "Coordinates": [38.9352, -76.9723], "Distance from Tract (km)": 7.399043271927804},
{"Vehicle ID": "5056ff2d-a5cb-4d42-ba8f-03e00bfcfc37", "type": "bike", "Coordinates": [38.9373, -76.9723], "Distance from Tract (km)": 7.403264604880642},
{"Vehicle ID": "b18e8d42-cce7-4036-949c-cae16cc369cc", "type": "bike", "Coordinates": [38.9312, -76.9743], "Distance from Tract (km)": 7.170097120168351},
{"Vehicle ID": "07348d7c-39e3-4c44-9c41-9da3b4b1664a", "type": "scooter", "Coordinates": [38.9126, -76.9761], "Distance from Tract (km)": 6.954609838721491},
{"Vehicle ID": "530d73e4-2b28-4ec4-b801-61c69e56bf84", "type": "scooter", "Coordinates": [38.9132, -76.9762], "Distance from Tract (km)": 6.943514557779142},
{"Vehicle ID": "f29f1306-7f16-4a1a-84fc-80f32c02f130", "type": "scooter", "Coordinates": [38.9465, -76.9777], "Distance from Tract (km)": 6.829938064184277},
{"Vehicle ID": "c7ce580a-7fca-4baa-8c4b-4254f9032d08", "type": "bike", "Coordinates": [38.9229, -76.9783], "Distance from Tract (km)": 6.715061171623303 5},
{"Vehicle ID": "2f63e82c-4dbc-477c-a730-b93d3d08020b", "type": "scooter", "Coordinates": [38.9194, -76.9785], "Distance from Tract (km)": 6.689996071081789},
{"Vehicle ID": "b73369cb-17f8-4096-8b6c-77d5cef5e474", "type": "scooter", "Coordinates": [38.9295, -76.9825], "Distance from Tract (km)": 6.257485316645527},
{"Vehicle ID": "c197326c-922e-4491-a0bf-b32a6b011946", "type": "bike", "Coordinates": [38.9255, -76.9833], "Distance from Tract (km)": 6.162639908910979},
{"Vehicle ID": "bd52c77a-d98d-4214-bd99-c14b32a59991", "type": "bike", "Coordinates": [38.9289, -76.9844], "Distance from Tract (km)": 6.0456822172251465},
{"Vehicle ID": "0111aa2f-b871-4d42-ad15-3d0729cd4620", "type": "scooter", "Coordinates": [38.9206, -76.9845], "Distance from Tract (km)": 6.024014379976478},
{"Vehicle ID": "e0617a91-ae5e-4740-be08-071b88833680", "type": "scooter", "Coordinates": [38.9119, -76.985], "Distance from Tract (km)": 5.964989937750534},
{"Vehicle ID": "077ac6a7-cfd2-413b-b70d-93d14d0e1208", "type": "bike", "Coordinates": [38.9251, -76.9859], "Distance from Tract (km)": 5.8734098873760665},
{"Vehicle ID": "ed937613-a315-41e4-ad63-0df31ffa5f68", "type": "bike", "Coordinates": [38.9308, -76.9863], "Distance from Tract (km)": 5.83845326549641},
{"Vehicle ID": "c627da1e-064f-4640-a429-9c7e93c9b374", "type": "bike", "Coordinates": [38.9116, -76.988], "Distance from Tract (km)": 5.63142943913855 3},
{"Vehicle ID": "971a8c8e-fe92-446a-9738-8857e3a6c046", "type": "bike", "Coordinates": [38.9241, -76.9883], "Distance from Tract (km)": 5.605592557749324},
{"Vehicle ID": "8d2cc2c2-1dd2-4724-88da-b5266af633e0", "type": "scooter", "Coordinates": [38.9276, -76.9892], "Distance from Tract (km)": 5.5109693672911675},
{"Vehicle ID": "1cfd0aa4-69ba-42ad-95b4-0fb101481c91", "type": "bike", "Coordinates": [38.9172, -76.9899], "Distance from Tract (km)": 5.42141721435287},
```

VI. VISUALIZATIONS

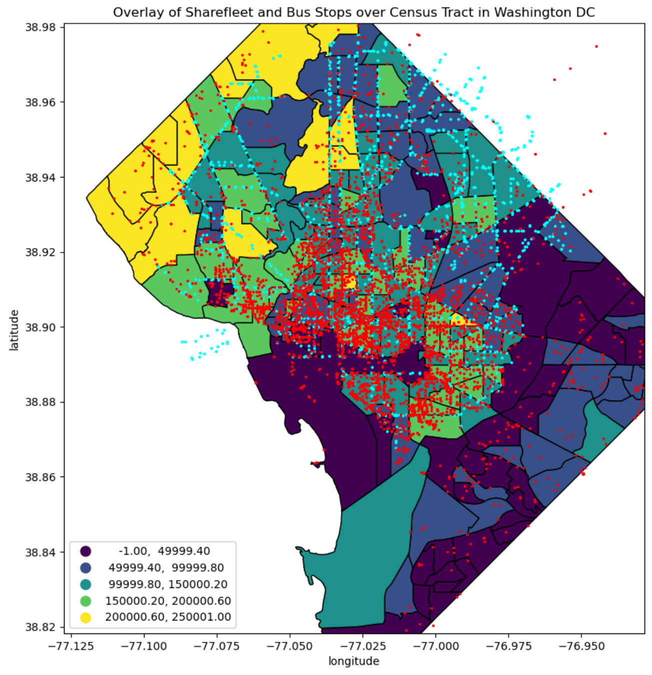
A. Sharedfleet (Lime) in Washington DC



B. Bus Stops in Washington DC



C. Overlay of Sharefleet and Bus Stops over Census Tract in Washington DC



VII. INSTRUCTIONS FOR RUNNING THE CODE

1. **Clone from github repository**
2. **Run all 3 setup files in this specific order:**
   eric.py → sean.py → dom.py
   a. **Check DB Browser (SQLite) to see that 25 items has been added to each of the three tables: Vehicles, bus_stop, Tracts**
   b. **Run all 3 files again.**
   c. **Repeat a-b at least 2 more times**
3. **Run data_processing.py**

VIII. CODE DOCUMENTATION

A. **dom.py** (Shared Fleet Devices API)
   1. **def get_data(url)**
      a) Input: API url (str)
      b) Output: JSON-formatted data (dict)
   2. **def database_creation(db_name)**
      a) Input: Database name (str)
      b) Output: Cursor and Connection (tuple)
   3. **def type_table(data, cur, conn)**
      a) Input: JSON data (dict)
      b) Output: A table called “Types” on the database (none)
   4. **def unix_conversion(data)**
      a) Input: JSON data (dict)
      b) Output: EST date time format (str)
   5. **def vehicle_table(data, cur, conn)**
      a) Input: JSON data (dict)
      b) Output: A table called “Vehicles” on the database (none)
   6. **def status_table(cur, conn)**
      a) Input: Cursor and Connection to the database (sqlite3)
      b) Output: A table called “Status” on the database (none)

B. **eric.py** (Census Tract API)
   1. **def get_database()**
      a) Input: api url to census tract tract (str)
      b) Output: json format of the data (list)
   2. **def writedb(data, db)**
      a) Input: json data of the census tract, database (list, str)
      b) Output: the database, containing tracts table (str)
   3. **def modifygeojson(db, infile, outfile)**
      a) Input: database name, input file name, output file name (str, str, str)
      b) Output: output file in geojson, containing tracts. (none)

C. **sean.py** (Bus Stop API)

1. **def get_bus_stop_data(url, headers) ()**
    a. Input: API url, request headers (str, dict)
    b. Output: JSON-formatted data (dict)
2. **def create_bus_stop_database(data, db_name)**
    a. Input: JSON-formatted data, SQLite database name (dict, str)
    b. Output: A table called "bus stops" and inserts data (none)
3. **def retrieve_bus_stop_data(db_name)**
    a. Input: SQLite database name (str)
    b. Output: List of tuples with bus stop data (list)
4. **def create_geo_dataframe_from_db(db_name)**
    a. Input: SQLite database name (str)
    b. Output: GeoDataFrame (gpd.GeoDataFrame)
5. **def plot_bus_stops_from_db(db_name)**
    a. Input: SQLite database name (str)
    b. Output: plots the bus stops on a matplotlib (none)

D. **data_processing.py**

1. **def coordinate(cur, table_name)**
    a) Input: Cursor to the database and table name (str)
    b) Output: (lat,lon) of all the items in a table (list)
2. **def point_creater(cur, table_name)**
    a) Input: Cursor to the database (str)
    b) Output: "Point(lat,lon)" format for GeoDataframe creation (list)
3. **def GeoDataFrame_creation(geometry)**
    a) Input: Points created from def point_creater (str)
    b) Output: GeoDataFrame of the vehicles (list)
4. **def visualize_census(file, ax)**
    a) Input: GeoJSON of the census tract file and ax (str)

        b) Output: Plotting the census tract (none)

    5. **def visualize_dots(gdf_object, ax, color, markersize)**
        a) Input: GeoDataFrame of the vehicles/bus stops, and axis to plot, color (string), markersize (string)
        b) Output: Plotting the position of vehicles/bus stops (graph)
    6. **def calc_distance(coordinate1, coordinate2)**
        a) Input: (tuples) lat, lon of cord1 and lat lon of cord 2
        b) Output: float (distance in km)
    7. **def dist_from_tract(infile, tractname, db, outfile)**
        a) Input: String - GeoJSON file name containing tracts, String - Name of the census tract, database name, output file name.
        b) Output: Json

IX. RESOURCES DOCUMENTATION

A. **ChatGPT:** What libraries/websites to use and their code structure
    1. **import datetime**: to convert UNIX timestamp into readable format (EST date time format)
    2. **Haversine formula**: to calculate distance from just lon and lat of an object

B. **GeoPandas:** Plotting documentation

C. **OpenData.DC:** Washington DC Boundary GeoJSON file