**Project 2 Report – AirBnB Austin – Foreigners Team**

A team effort by Kavya Shabnavees, Yan Xu, Michel Maniguet, and Vladislav Titov.

**Introduction**

The purpose of this project was to analyze Air BnB listings data from the Austin TX area, so as to answer questions such as a traveler to the area might have as they seek lodging. Answering questions such as price variation and availability of listings by location would be of value to travelers. This is to be accomplished by visualizing different aspects of this data source.

**Approach**

The project used several different technical capabilities to implement this project.

1. SQL: Database for storage and user query
2. Flask: back end support, local server
3. HTML: user interface, front end general purpose
4. JavaScript: user interface, front end general purpose
5. Leaflet: user interface, front end, geographical data visualization
6. ChartsJS/Plotly: user interface, graphical data visualization
7. AJAX: Data tables/sorting

After loading and reviewing the data, it was cleaned and analyzed as described below.

**Strategy**

***Step 1) Extraction***

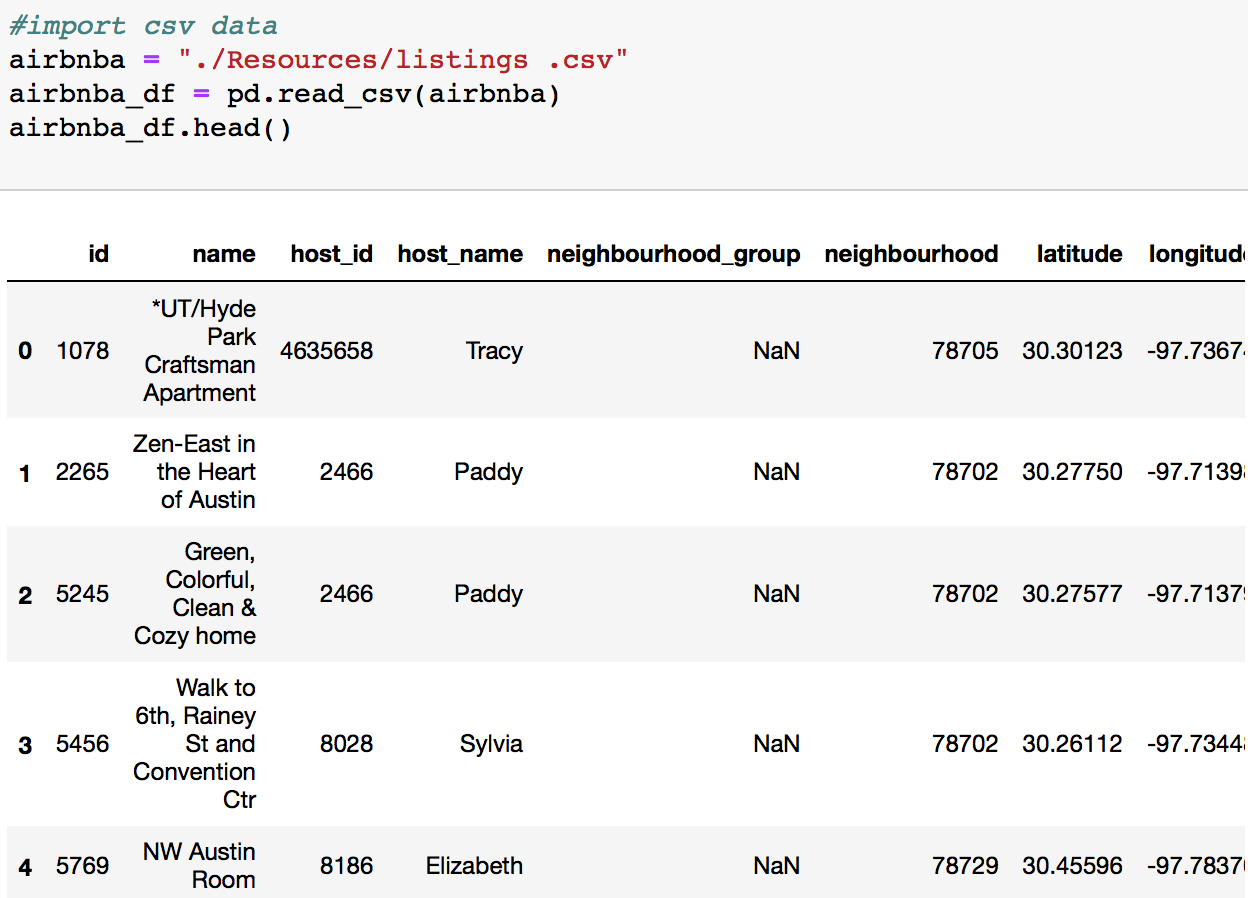
The data was obtained from <http://insideairbnb.com/get-the-data.html>. The dataset is as of May 14, 2019 and contains 11,792 records. Information includes but not limited to latitude and longitude of the listing, price, room type and number of reviews. The dataset was downloaded in the comma separated value (csv) format and was cleaned in Pandas.

The data was downloaded as a csv file.

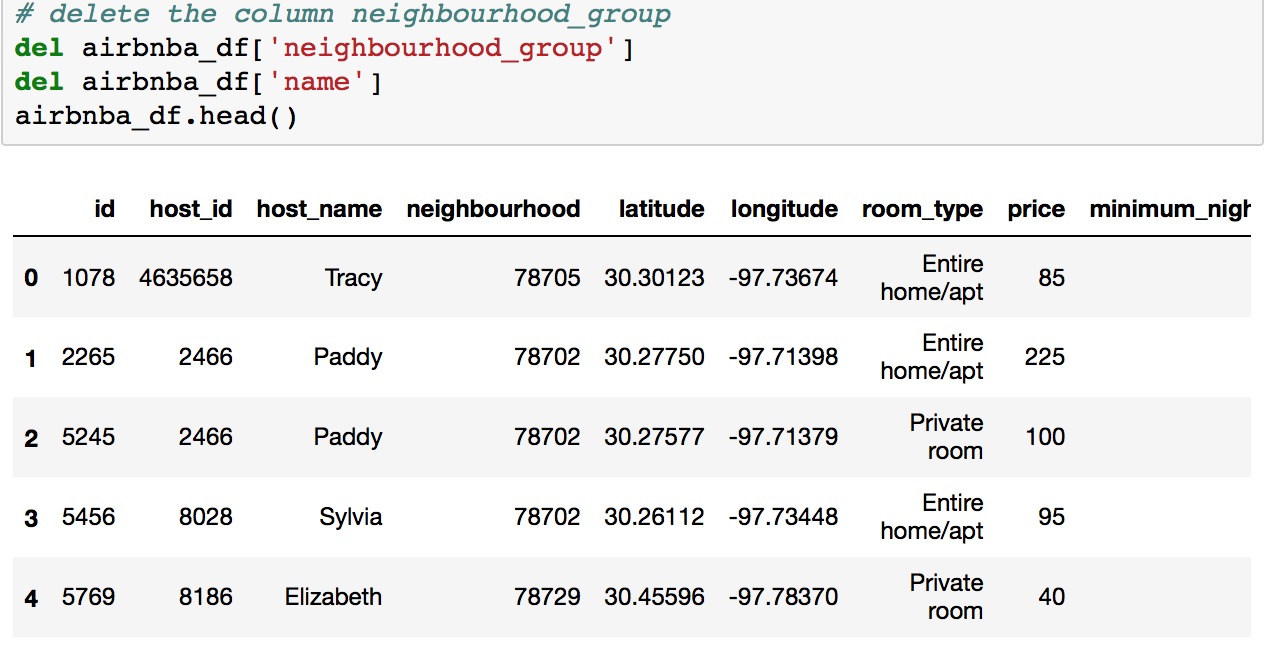
***Step 2) Transformation***

The updated csv file was read into Pandas using the Jupyter Notebook as a separate dataframe and the following manipulation procedures were performed:

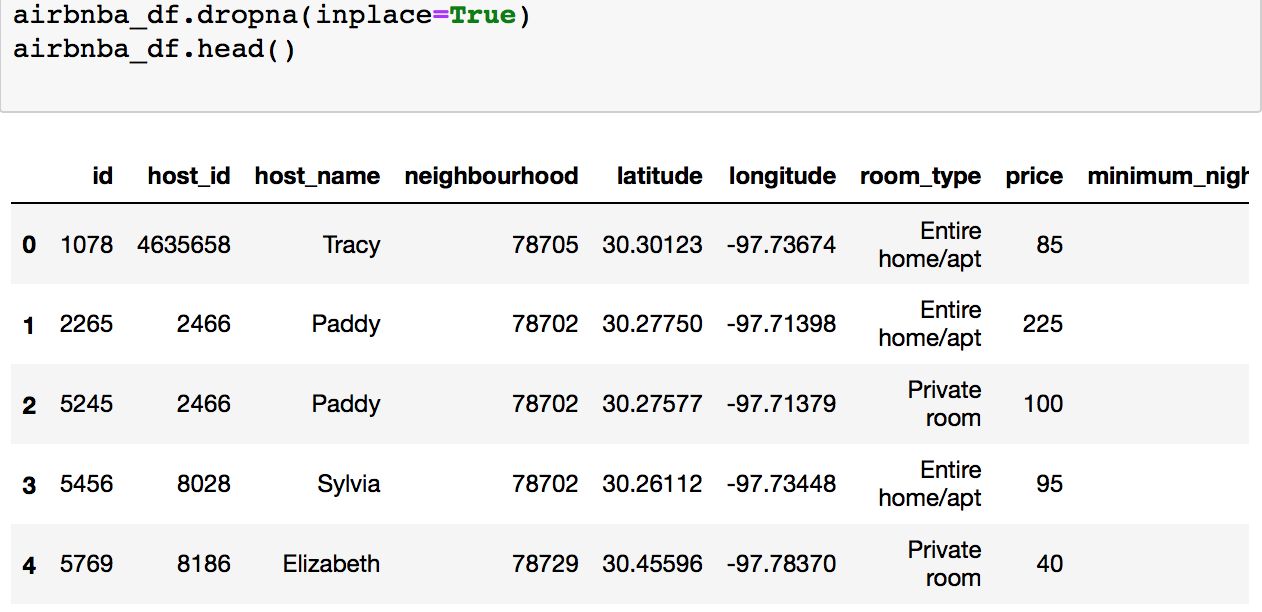
Read in files and view headers



Deletion of columns containing irrelevant, duplicated or otherwise unusable information: neighborhood\_group, hostname.

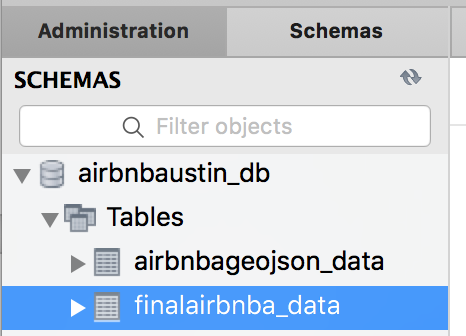


Keep only the rows without non-NA values



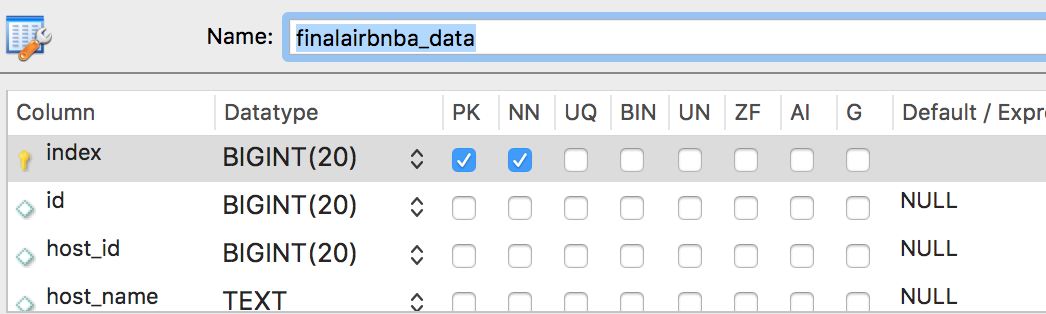
***Step 3) Loading***

Load data and create a table



**Set Primary Key:**

We used index as the primary key.



**Leaflet Overview**

Geographical visual representations were produced using Leaflet. Leaflet is an open-source JavaScript library for interactive maps. This extensive JS library allows rendering of maps with various features that allow extensive integration of user data allowing for significant customization of the user experience. Also important was Leaflet’s compatibility with Mapbox object libraries, which offer mapping functionality vastly extended and superior to Google Maps.

Capabilities such as multiple base maps, overlay maps, placing of markers at individual GPS points, GeoJason support, control utilities and many more features made Leaflet a powerful tool to serve in the visualization of geographic and related data.

The powerful capabilities of Leaflet notwithstanding, the Leaflet library is far from easy to use. I pity the fool who must start a geographical visualization project from scratch just using the online Leaflet documentation. Full capabilities of this package are truly at Mariana depths. Sample code, some available within the library documentation, is a must for reasonably rapid ramp of Leaflet functionality.

**Significant Code Samples & Brief Description**

The following code loads the foundational map object, to which base maps and layers adhere:

var mapptr = L.map("map", {

center: [30.2672,-97.7431], // Austin TX GPS coordinates

zoom: 10

});

The following code loads base map:

var tileStreets L.tileLayer(<map URL+access tokens", {

attribution: "Map data &copy; <a href=\"https://www.openstreetmap.org/\">OpenStreetMap</a> contributors, <a href=\"https://creativecommons.org/licenses/by-sa/2.0/\">CC-BY-SA</a>, Imagery © <a href=\"https://www.mapbox.com/\">Mapbox</a>",

maxZoom: 18,

id: "mapbox.streets",

accessToken: API\_KEY

}).addTo(addToMap);

The following code implements base map selection, overlays and control panel:

L.control.layers(baseMaps, overlayMaps, {

minZoom: 0,

maxZoom: 12,

collapsed: false

})

.addTo(map);

For complete Leaflet documentation, refer to <https://leafletjs.com/>

Some of the available functionality was not implemented in the final release. These include mapping of zip code area boundaries using GeoJson publicly available data objects and the use of layer controls to select data and color keys to explain certain map features. Finally, the deployment of Leaflet. .markerClusterGroup(); to create “market clusters” the locations of AirBNB listings by type or price or rating is probably the most significant enhancement available to this application. This is due to the extremely dense geographical proximity of the AirBnB listings in certain areas. Market clustering would make the shear amount of listing data manageable to the user. This combined with selectable overlays would greatly improve efficiency and enjoyment in the user experience.

**<<NOTE TO VLAD: Boiling this down into simple outline/bullet points summary for the section on Future Improvements/Problems to Address would be good.>>**

**Portability and Standaloneability**

While not completed before the project presentation deadline, extended MySQL/Flask functionality beyond the capability of the application release of that date is highly desirable. Specifically, it was recognized the Python Flask application should be upgraded with the ability to load MySQL, delete old and initialize new, the MySQL database structures necessary for complete functionality. Without this additional functionality, a human user would be required to startup MySQL and initialize the database, all manually, in order to support proper functionality of the Project.

The goal is that for any user pulling the project repository from GitHub, the sole requirement beyond replicating the repository image on their local machine would be to insert the local MySQL account password (and possibly localhost ID). Upon fulfilling these minimal requirements, all the functionality of the AirBnB Austin Project would be functional.

**Flask Endpoints & Description defined in app.py file**

Flask server base URL: http://127.0.0.1:5000

# Flask Routes

#################################################

# approute that renders index page

@app.route("/")

def index():

return render\_template("index.html")

@app.route("/zipchart")

def chartjs():

return render\_template("chartjs.html")

# app route for getting zip codes as a list for populating dropdown

@app.route("/zipcodes")

def zips():

"""Return a list of unique zipcodes."""

data = {

"zips": df.neighbourhood.unique().tolist()}

return jsonify(data)

# app route to get data for populating listings per room\_type when a zip code is selected

@app.route("/zipcode/listings/<zipdata>")

def zipcodelistings(zipdata):

# gets all the data for a zipcode that matches with the selection

filtered\_df = df[df.neighbourhood == int(zipdata)]

zip\_df = filtered\_df.groupby("room\_type")["room\_type"].count()

jsondump = zip\_df.to\_json()

return jsondump

# app route for getting avg price for each room type for a zipcode selected

@app.route("/zipcode/price/<zipcode>")

def zipcodeprice(zipcode):

filtered\_df = df[df.neighbourhood == (int(zipcode))]

price\_df = round(filtered\_df.groupby("room\_type")["price"].mean(),2)

jsondump = price\_df.to\_json()

return jsondump

# app route for chart.js data

@app.route("/zipchart/<zipdata>")

def zipchart(zipdata):

filtered\_df = df[df.neighbourhood == (int(zipdata))]

data = {

"Roomtype\_Listings": filtered\_df.groupby("room\_type")["room\_type"].count().to\_json(),

"Roomtype\_avgprice": round(filtered\_df.groupby("room\_type")["price"].mean(),2).to\_json(),

"Roomtype\_Reviews": filtered\_df.groupby("room\_type")["number\_of\_reviews"].sum().to\_json(),

}

return jsonify(data)

# app route for returning leaflet map data jsonified

@app.route("/leaflet/<zipcode>")

def lmap(zipcode):

filtered\_df = df[df.neighbourhood == (int(zipcode))]

return filtered\_df.to\_json()

# app route for rendering leaflet map

@app.route("/leafletmap")

def getmap():

return render\_template("leaflet.html")

# app route for rendering leaflet map

@app.route("/stackedchartjs")

def getstackmapdata():

mean\_df = round(df.groupby(["neighbourhood","room\_type"])["price"].mean(),2)

unstack\_data = mean\_df.unstack(fill\_value=0)

stack\_data = unstack\_data.rename(columns={"Entire home/apt": "Entire\_home\_apt","Private room": "Private\_room","Shared room":"Shared\_room"}).to\_json()

return stack\_data

# app route for rendering Data Table

@app.route("/dtable")

def table():

return render\_template("table.html")

# app route for getting dtable data

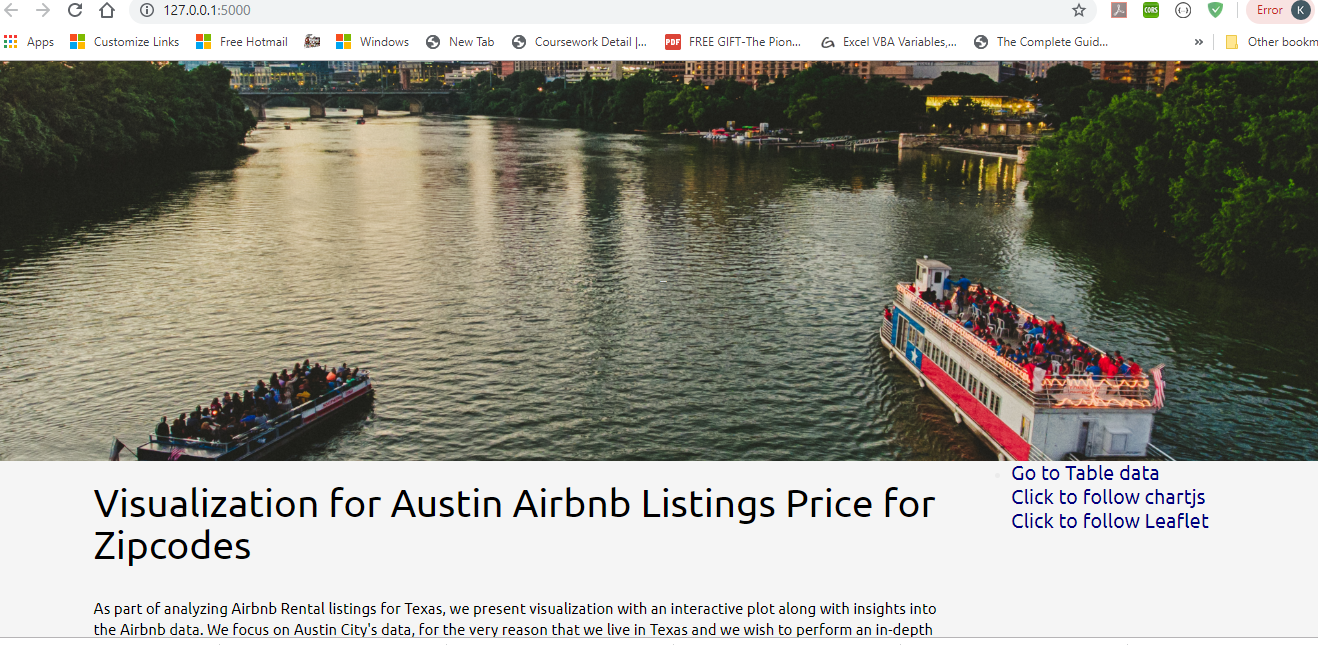
@app.route('/ajax')

def hello\_world(name=None):

return df.to\_json(orient='records')

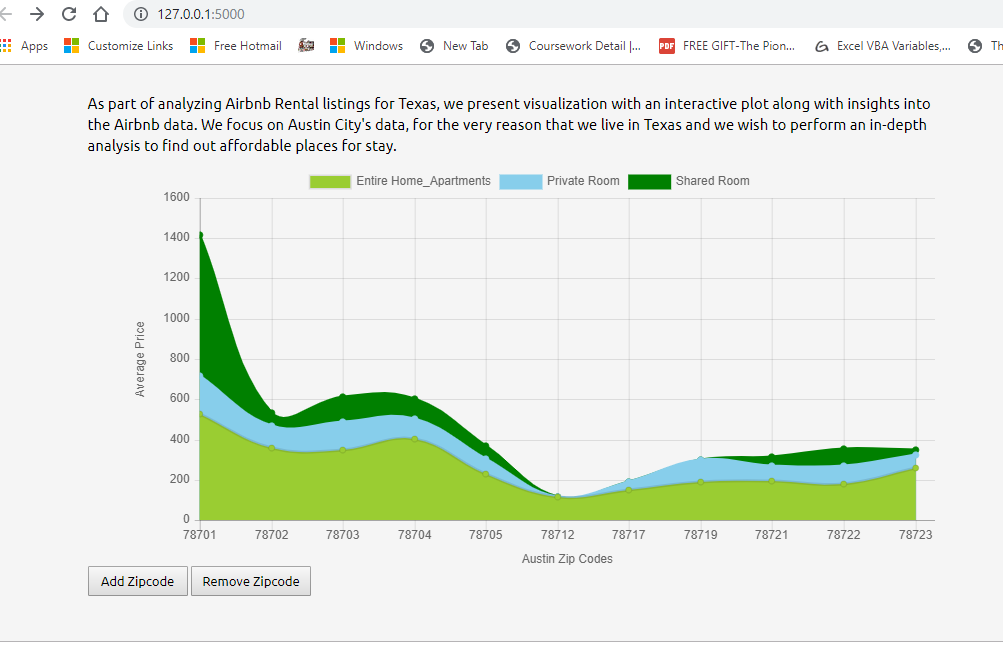
**HTML PAGE:**

* **Index.html:**



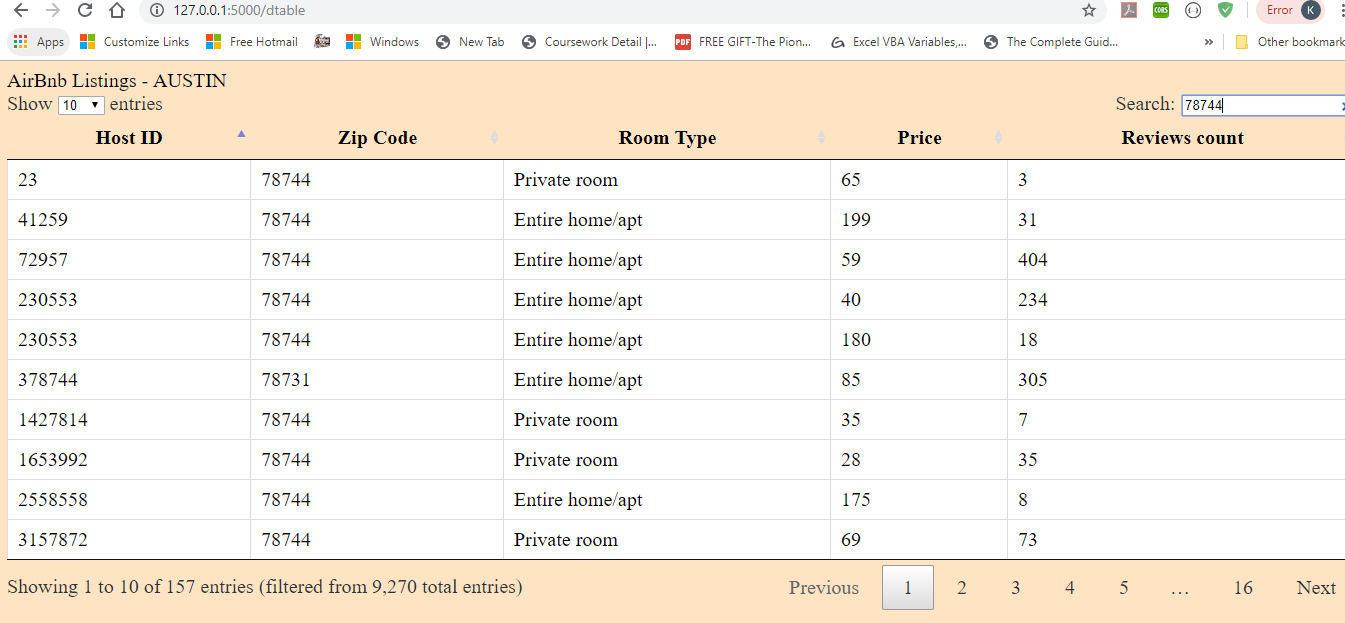
* **Summary Stacked Line chart:**

Below chart shows Average Price for each Zipcode. Also, the buttons below the chart enables to add/ remove zipcodes which makes chart more interactive.



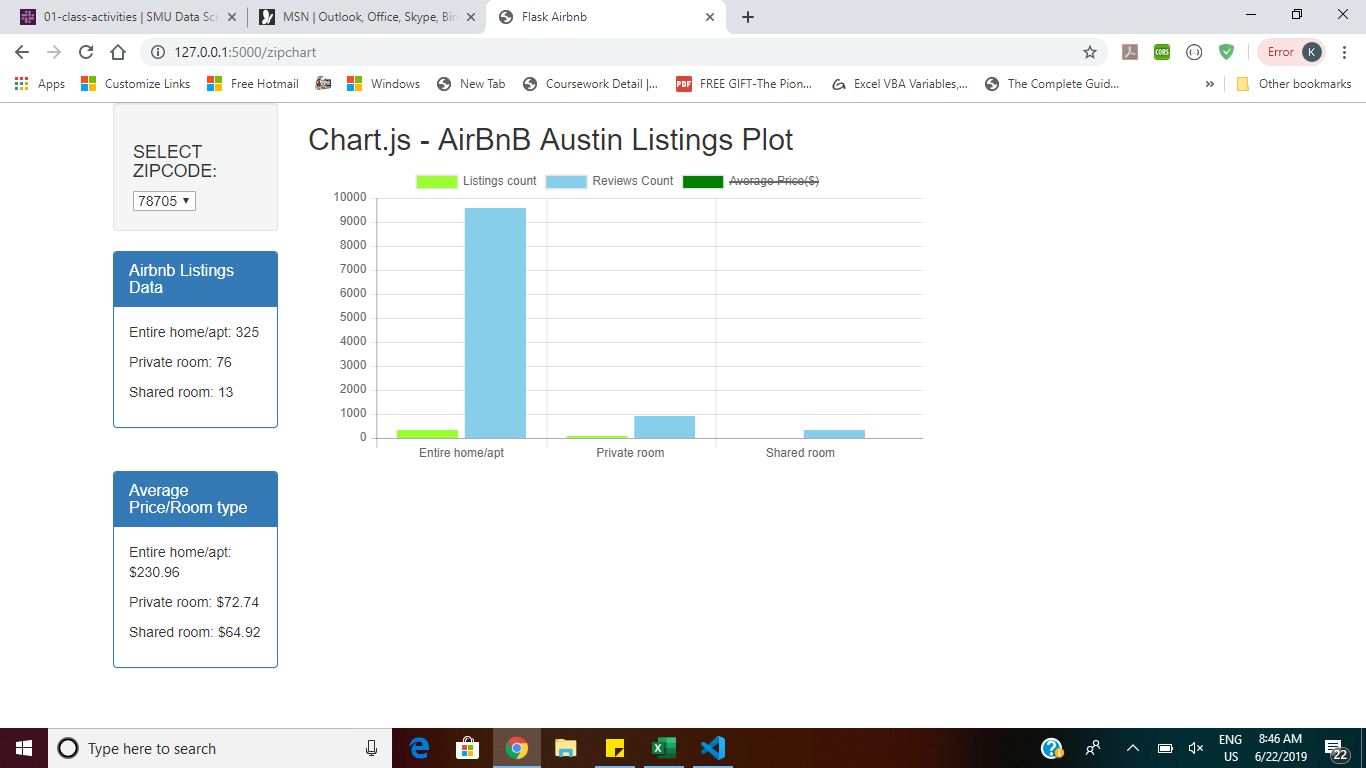
DataTable (script in app.js)

Data Table Ajax plugin is used for its efficient table capabilities including search options.



* Chartjs Bar Chart:

We used Chartjs library for bar chart below to show Listings count, Reviews count and Average Price in dollars for different home types.



**Conclusions:**

1. Geographical overview shows riverfront/downtown area has highest listing density. This is consistent with housing type density (apartments & smaller houses) and market demand (UT-A, theatre district, etc)

A map of a computer

Description automatically generated

2. The majority of listings in Austin are centered around Downtown area and ZIP code 78704 has highest number of listings.

3. Entire home/apartment is the most popular listing type in Austin though Entire home/apartment is the most expensive listing type.

**Limitations of the dataset:**

* The analyzed dataset did not include the average ratings of each listing. The team utilized the number of reviews as an indicator of the quality of the listing and assumed that higher number reviews corresponds with higher rating of the listing. Granted this is a broad assumption.
* If starting the project again we would combine our dataset with another dataset from insideairbnb.com which includes review scores data. Incorporating review scores into analysis would have allow us to achieve more accurate results.

**Appendix**

**Project repository Github link**: <https://github.com/yanzi1210/project2_AirbnbAustin>