Project 2 Report – AirBnB Austin – Foreigners Team

Step 1) Extraction

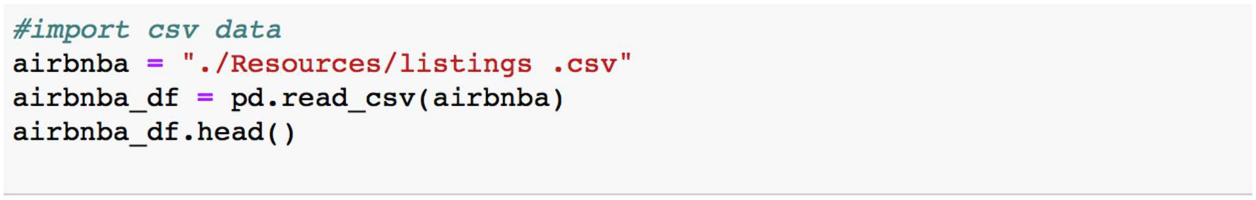
The data was obtained from http://insideairbnb.com/get-the-data.html. The dataset is as of May 14, 2019 and contains 11,792 records. Information includes but not limited to latitude and longitude of the listing, price, room type and number of reviews. The dataset was downloaded in the comma separated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

The data was either downloaded as a csv file.

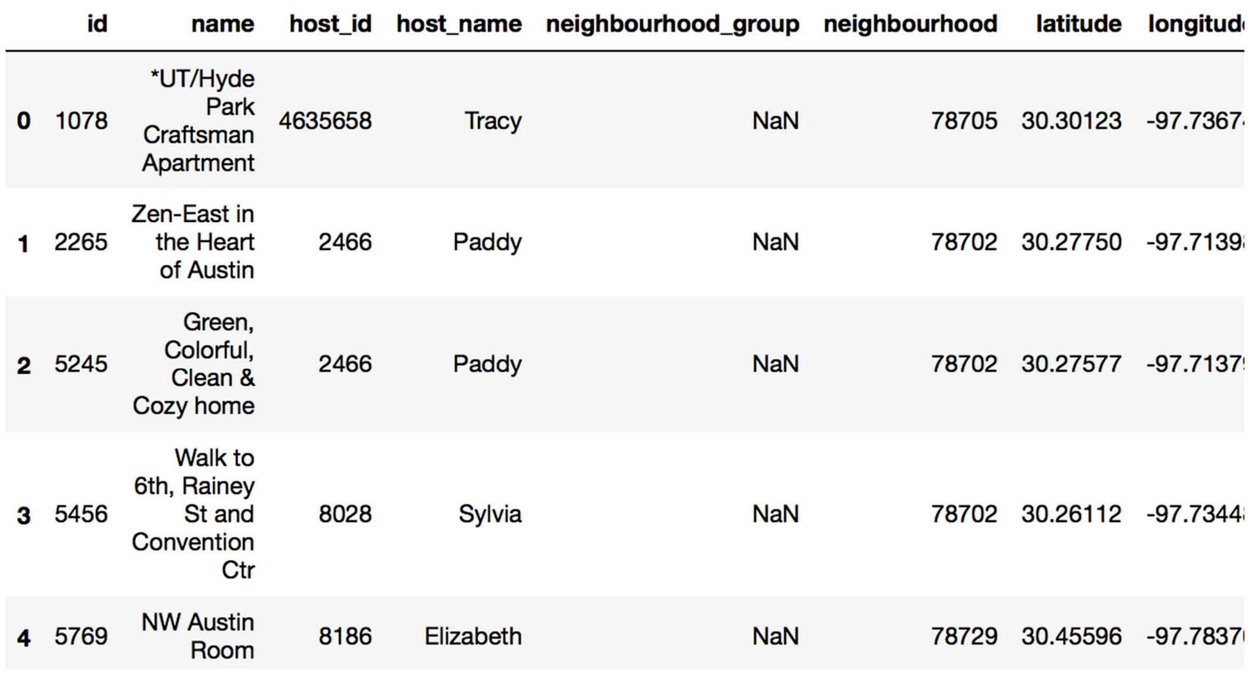
Step 2) Transformation

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

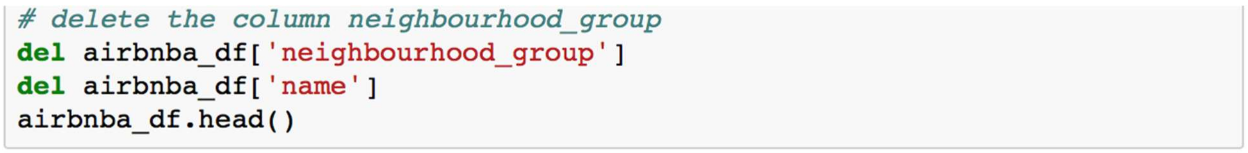


```
#import csv data
airbnba = "./Resources/listings .csv"
airbnba_df = pd.read_csv(airbnba)
airbnba_df.head()
```

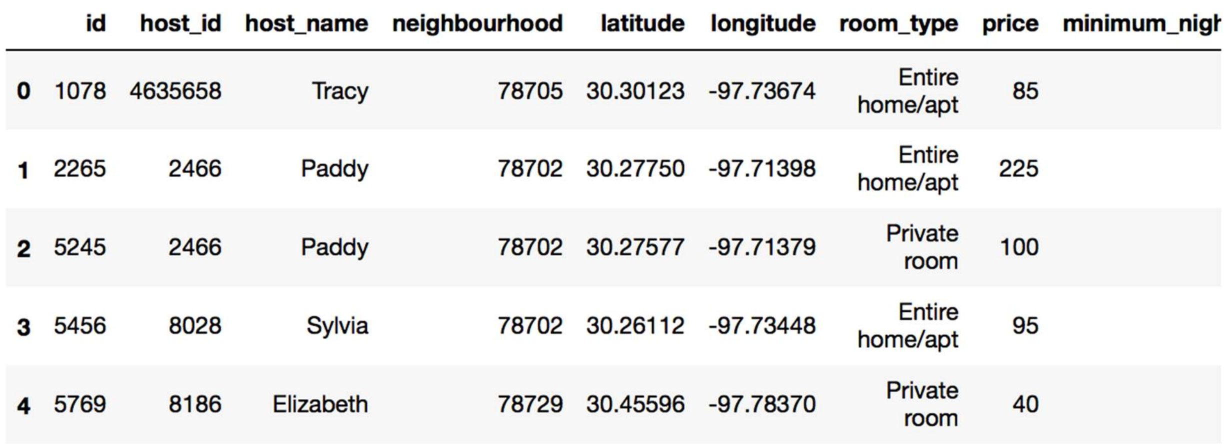


	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitud
0	1078	*UT/Hyde Park Craftsman Apartment	4635658	Tracy	NaN	78705	30.30123	-97.7367
1	2265	Zen-East in the Heart of Austin	2466	Paddy	NaN	78702	30.27750	-97.7139
2	5245	Green, Colorful, Clean & Cozy home	2466	Paddy	NaN	78702	30.27577	-97.7137
3	5456	Walk to 6th, Rainey St and Convention Ctr	8028	Sylvia	NaN	78702	30.26112	-97.7344
4	5769	NW Austin Room	8186	Elizabeth	NaN	78729	30.45596	-97.7837

Delete columns

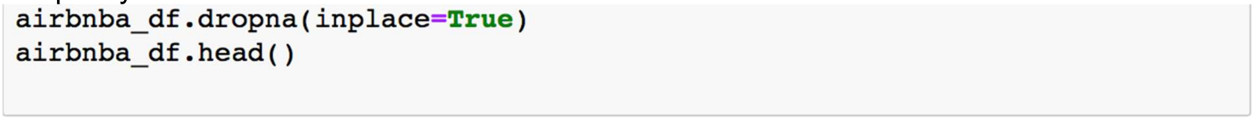


```
# delete the column neighbourhood_group
del airbnba_df['neighbourhood_group']
del airbnba_df['name']
airbnba_df.head()
```

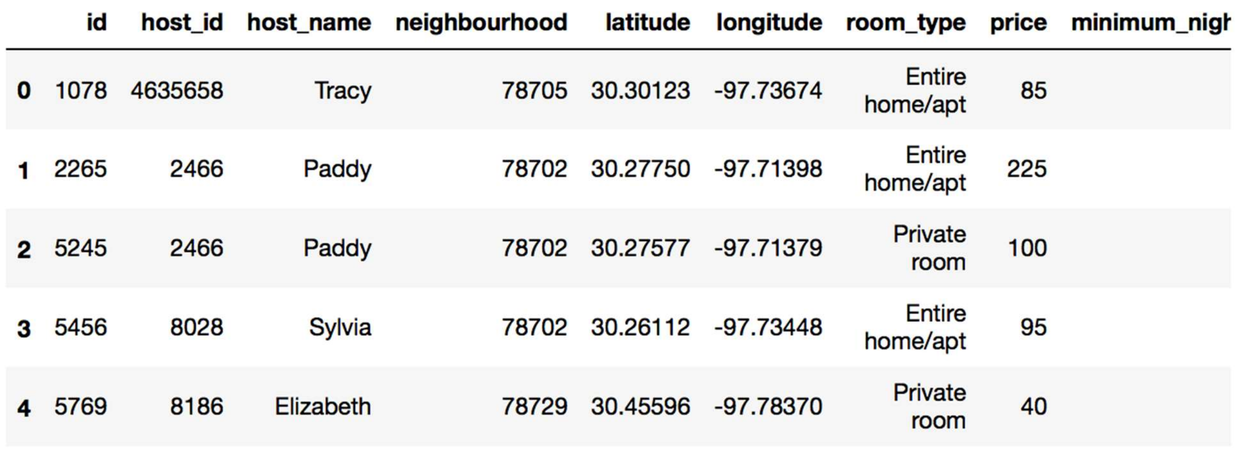


	id	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nigh
0	1078	4635658	Tracy	78705	30.30123	-97.73674	Entire home/apt	85	
1	2265	2466	Paddy	78702	30.27750	-97.71398	Entire home/apt	225	
2	5245	2466	Paddy	78702	30.27577	-97.71379	Private room	100	
3	5456	8028	Sylvia	78702	30.26112	-97.73448	Entire home/apt	95	
4	5769	8186	Elizabeth	78729	30.45596	-97.78370	Private room	40	

Keep only the rows without non-NA values



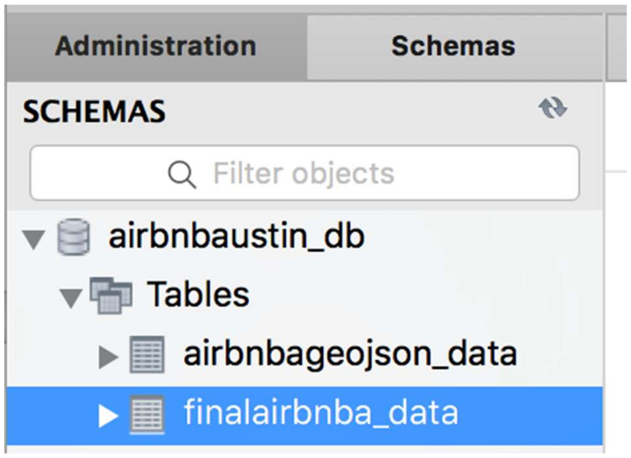
```
airbnba_df.dropna(inplace=True)
airbnba_df.head()
```



	id	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nigh
0	1078	4635658	Tracy	78705	30.30123	-97.73674	Entire home/apt	85	
1	2265	2466	Paddy	78702	30.27750	-97.71398	Entire home/apt	225	
2	5245	2466	Paddy	78702	30.27577	-97.71379	Private room	100	
3	5456	8028	Sylvia	78702	30.26112	-97.73448	Entire home/apt	95	
4	5769	8186	Elizabeth	78729	30.45596	-97.78370	Private room	40	

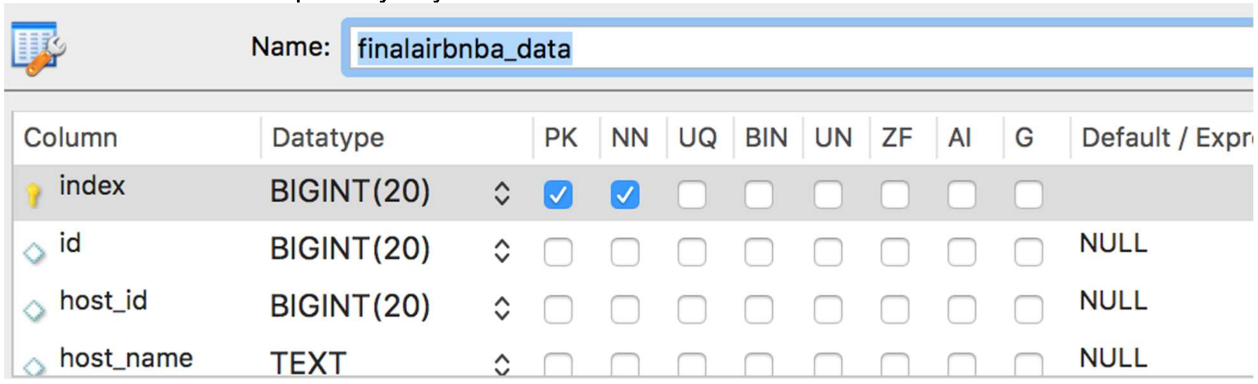
Step 3) Loading

Load data and create a table



Set Primary Key:

We used index as the primary key.



Name: finalairbnba_data

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expr
index	BIGINT(20)	✓	✓							
id	BIGINT(20)									NULL
host_id	BIGINT(20)									NULL
host_name	TEXT									NULL

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.

<u>Flask Endpoints & Description defined in app.py file</u>

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)
```

```python
# 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")
```

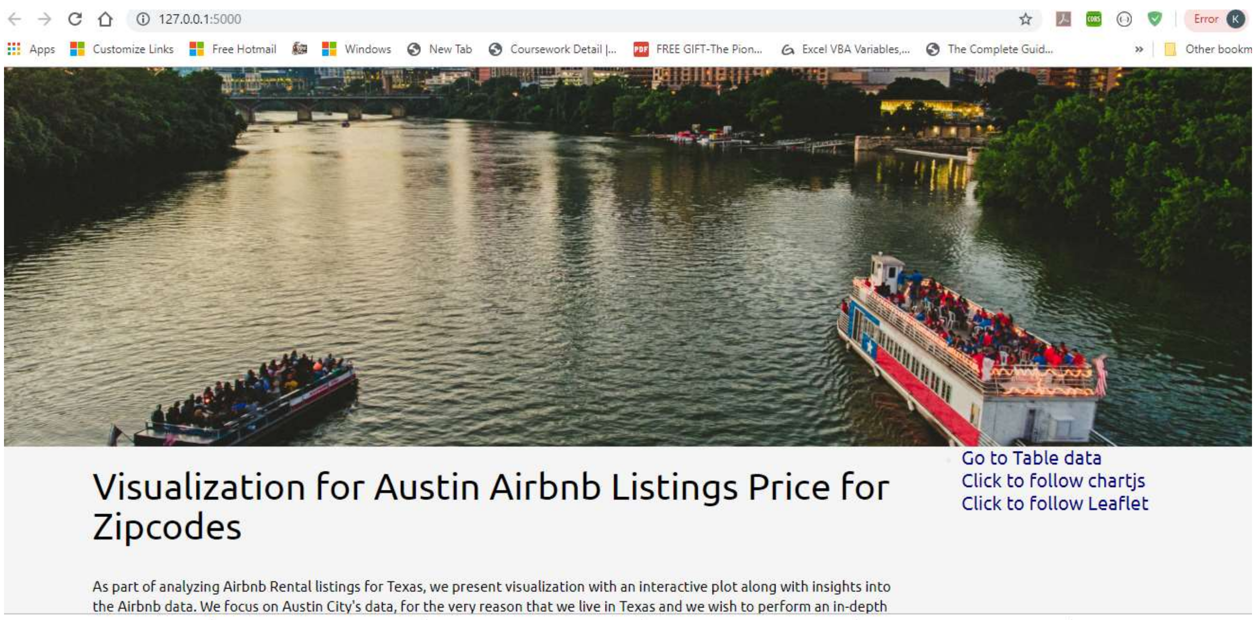
```
# app route for rendering leaflet map
@app.route("/stackedchartjs")
def getstackmapdata():
     mean_df = round(df.groupby(["neighbourhood","room_type"])["price"].mean(),2)
     unstack_data = mean_df.unstack(fill_value=0)
     stack_data = unstack_data.rename(columns={"Entire home/apt":
"Entire_home_apt","Private 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')
```

<u>HTML PAGE:</u>

- **<u>Index.html:</u>**

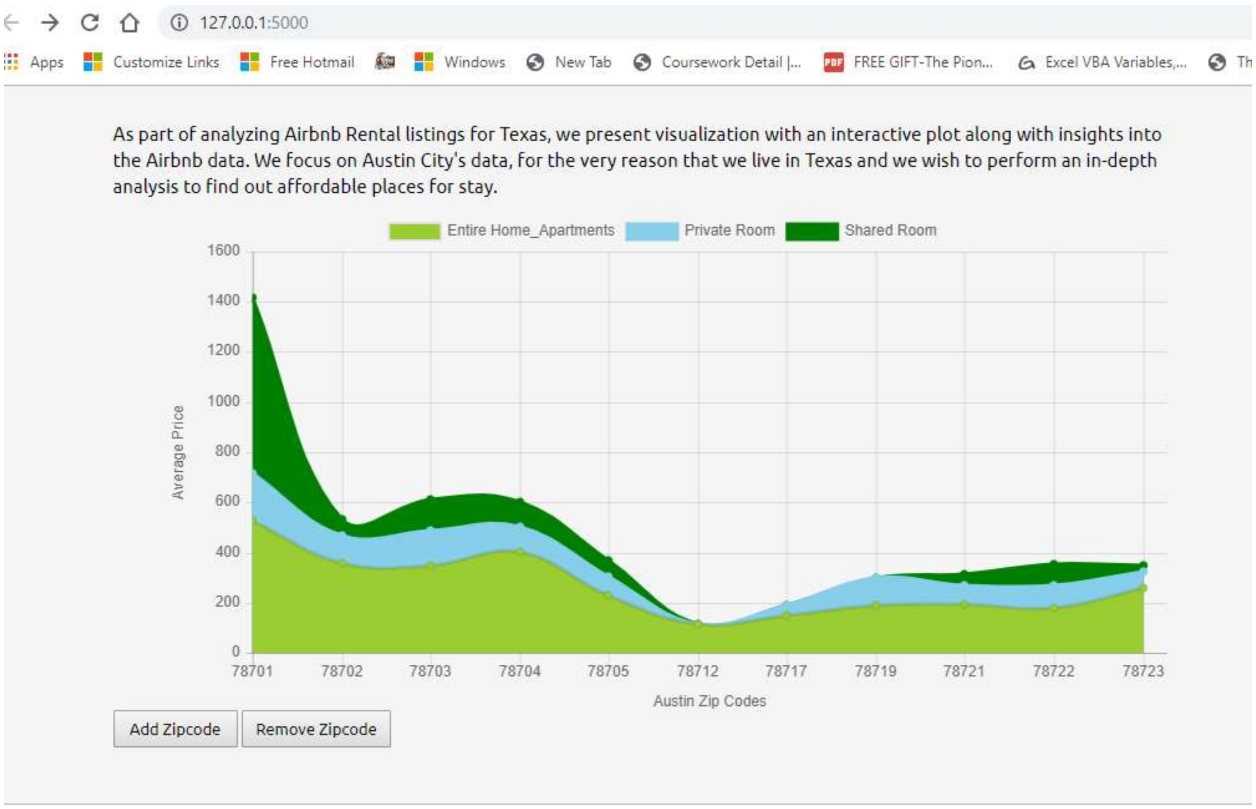


Visualization for Austin Airbnb Listings Price for Zipcodes

As part of analyzing Airbnb Rental listings for Texas, we present visualization with an interactive plot along with insights into the Airbnb data. We focus on Austin City's data, for the very reason that we live in Texas and we wish to perform an in-depth

- **<u>Summary Stacked Line chart:</u>**

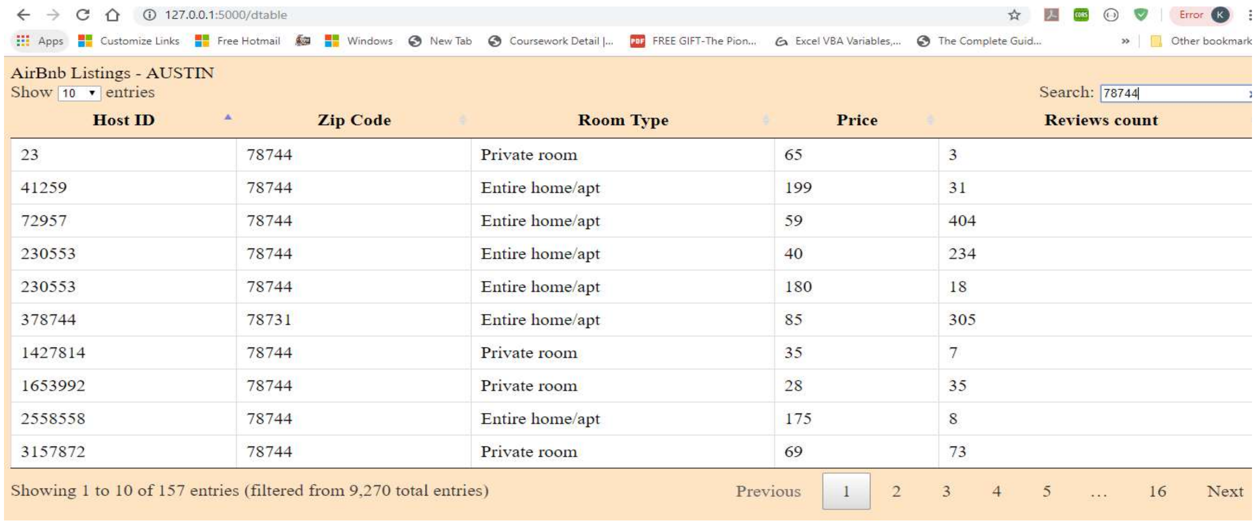
Below chart shows Average Price for each Zipcode. Also, the buttons below the chart enables to add/ remove zipcodes which makes chart more interactive.



As part of analyzing Airbnb Rental listings for Texas, we present visualization with an interactive plot along with insights into the Airbnb data. We focus on Austin City's data, for the very reason that we live in Texas and we wish to perform an in-depth analysis to find out affordable places for stay.

- <u>DataTable (script in app.js)</u>

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



AirBnb Listings - AUSTIN

Show 10 entries

Search: 78744

Host ID	Zip Code	Room Type	Price	Reviews count
23	78744	Private room	65	3
41259	78744	Entire home/apt	199	31
72957	78744	Entire home/apt	59	404
230553	78744	Entire home/apt	40	234
230553	78744	Entire home/apt	180	18
378744	78731	Entire home/apt	85	305
1427814	78744	Private room	35	7
1653992	78744	Private room	28	35
2558558	78744	Entire home/apt	175	8
3157872	78744	Private room	69	73

Showing 1 to 10 of 157 entries (filtered from 9,270 total entries)

Previous 1 2 3 4 5 … 16 Next

- <u>Chartjs Bar Chart:</u>

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

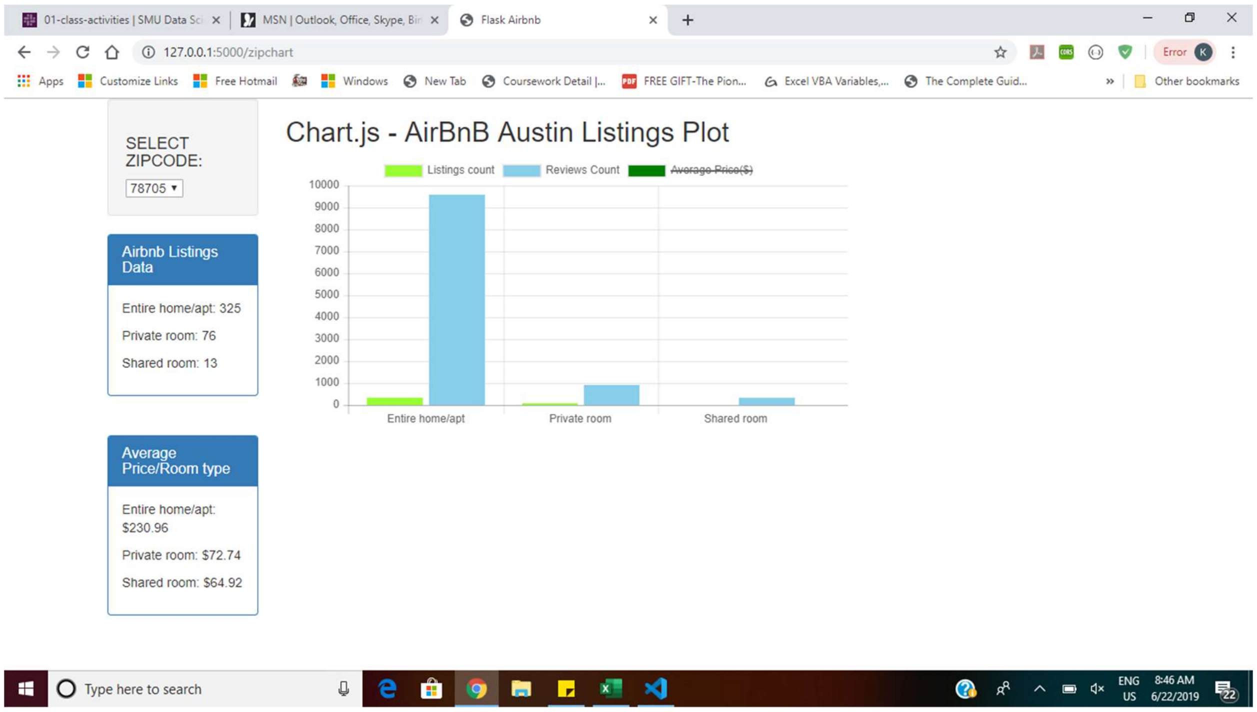


Chart.js - AirBnB Austin Listings Plot

SELECT ZIPCODE:
78705

Airbnb Listings Data

Entire home/apt: 325

Private room: 76

Shared room: 13

Average Price/Room type

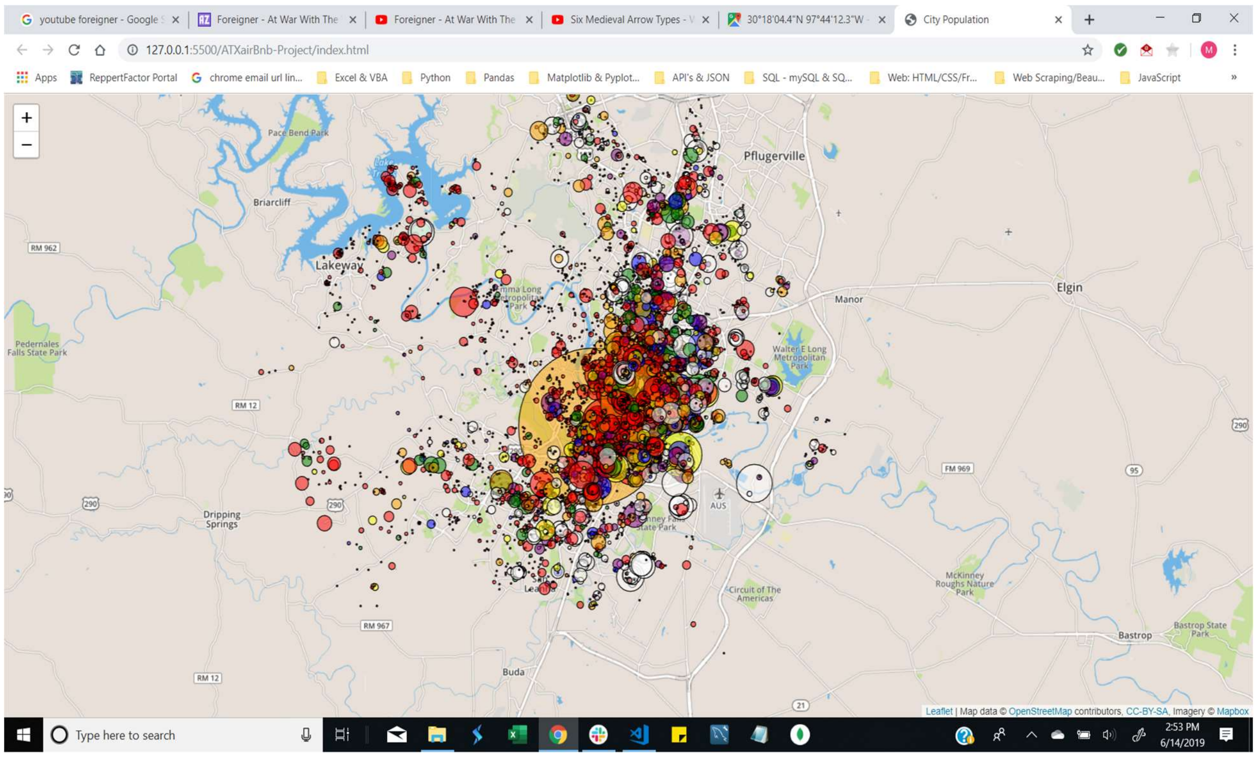
Entire home/apt: $230.96

Private room: $72.74

Shared room: $64.92

<u>Conclusions:</u>

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)



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.