

# Airbnb Scraping

CS 4624 Multimedia, Hypertext, and Information Access

Final Report

Virginia Tech, Blacksburg VA 24061

13 May 2020

Instructor: Dr. Edward A. Fox

Client: Florian Zach

Team: Yu Wang , Baokun Huang , Han Liu, Vinh Pham, Alexander Nikolov



VIRGINIA TECH™

# Table of Contents

## Contents

Table of Figures.....	4
Table of Tables .....	6
1.Abstract.....	7
2.Introduction .....	8
3.Requirements.....	10
3.1 Data .....	10
3.2 Graphs .....	10
3.3 Website .....	11
3.4 Automation .....	11
4.Design.....	12
4.1 Web Scraping Module .....	12
4.2 Data Visualization Module .....	13
4.3 Web Representation Module.....	14
5.Implementation .....	17
5.1 Data Collection .....	17
5.2 Visualization .....	21
5.2.1 Visualization for Austria and Virginia .....	22
5.2.2 Visualization for Austria States and Virginia Counties .....	25
5.3 Website .....	31
6.Testing/Evaluation/Assessment .....	32
7.Users' Manual .....	34
8.Developer's Manual .....	36
8.1.Docker Instructions .....	36
8.1.1.Running Docker .....	37
8.1.2.Running a Survey with Docker.....	37
8.1.3.Exporting Data and Accessing the Database with Docker .....	39

8.2.Non-Docker Instructions .....	39
8.2.1.Setup.....	40
8.2.2.Getting the script.....	47
8.2.3.Setting up the Database .....	47
8.2.4.Configuration .....	50
8.2.5.Installing lxml and psychopg2.....	51
8.2.6.Running the Script .....	52
8.2.7.Exporting and Viewing the Data.....	55
8.3.Methodology.....	56
8.3.1.User Goals.....	56
8.3.2.Tasks and Subtasks .....	57
8.3.3.Implementation-based Services.....	64
8.3.4.Workflows.....	68
8.4.Files and Folders.....	69
8.4.1.Root Directory .....	69
8.4.2.The “docker” Folder .....	72
8.4.3.The “data” Folder .....	73
8.4.4.The “postgresql” Folder.....	73
9.Lessons Learned.....	74
10.Acknowledgments .....	78
11.References .....	79

# Table of Figures

Figure 1: Graph Page.....	16
Figure 2: Bounding Box Initial Step.....	18
Figure 3: Bounding Box Second Step .....	18
Figure 4: Bounding Box Steps (continued).....	19
Figure 5: A Geo Map for Room Types' Distribution in Austria .....	22
Figure 6: A Geo Map for Number of Reviews' Distribution in Austria .....	23
Figure 7: A Geo Map for Availability Distribution in Austria .....	23
Figure 8: A Geo Map for Price Distribution in Austria .....	24
Figure 9: A Histogram Chart for Price Distribution in Austria.....	24
Figure 10: A Pie Chart for Instant-Bookable Listings in Austria .....	25
Figure 11: A Pie Chart for Superhost Listings in Austria .....	25
Figure 12: A Geo Map for Room Types' Distribution in Salzburg .....	26
Figure 13: A Geo Map for Availability Distribution in Salzburg .....	26
Figure 14: A Geo Map for Number of Reviews' Distribution in Salzburg .....	27
Figure 15: A Geo Map for Prices' Distribution in Salzburg .....	27
Figure 16: A Histogram Chart for Price Distribution in Salzburg .....	28
Figure 17: Pie Chart for Instant-Bookable Listings (Salzburg) .....	28
Figure 18: Pie Chart for Superhost Listings (Salzburg).....	29
Figure 19: A Word Cloud Map for User Reviews in Salzburg (Austria).....	29
Figure 20: A Scatter Plot of Hosts' Properties in Salzburg .....	30
Figure 21: A Scatter Plot of Price vs Number of Reviews in Salzburg.....	30
Figure 22: Website Layout .....	31
Figure 23: Airbnb Website Listing.....	33
Figure 24: Country Selection.....	34
Figure 25: State Selection .....	34
Figure 26: County Selection .....	35
Figure 27: PostgreSQL Download Page.....	40
Figure 28: PostgreSQL Windows Download Page.....	41
Figure 29: PostgreSQL Version Download Page .....	41
Figure 30: PostgreSQL Installer .....	42
Figure 31: PostgreSQL Installer Finish Page.....	42
Figure 32: Stack Builder Wizard .....	43
Figure 33: Stack Builder Wizard - Application Choice.....	43
Figure 34: PostGIS Installer .....	44
Figure 35: PostGIS Setup Box.....	44
Figure 36: Anaconda Download Page .....	45

Figure 37: Anaconda Installer for Windows .....	45
Figure 38: Anaconda Installing.....	46
Figure 39: Finding Path to Anaconda and Python .....	46
Figure 40: The pgAdmin4 Dashboard .....	47
Figure 41: pgAdmin4 - Right-Clicking on 'Databases' .....	48
Figure 42: pgAdmin4 - Database Creation.....	48
Figure 43: Applying the Database Schema .....	50
Figure 44: Downloading lxml .....	51
Figure 45: Downloading psycopg2.....	52
Figure 46: Testing Database Connection .....	52
Figure 47: Update search_area Message From: python airbnb.py -asa.....	52
Figure 48: Connecting to the Database with psycopg2.....	53
Figure 49: Finding the Bounding Box for Blacksburg.....	53
Figure 50: Updating the search_area.....	54
Figure 51: Message Displayed From: python airbnb.py -asv.....	54
Figure 52: Message Displayed From: python airbnb.py -sb .....	55
Figure 53: Viewing Data Through pgAdmin4.....	56
Figure 54: Process of Fixing Existing Code .....	57
Figure 55: Process of Updating the Code .....	58
Figure 56: Process of Making the Code Automatically Scrape .....	59
Figure 57: Process of Retrieving the Data.....	60
Figure 58: Process of Plotting Geo Maps.....	61
Figure 59: Process of Plotting Charts, Maps, and Tables.....	62
Figure 60: Display the Visualizations on the Website.....	63
Figure 61: Design of the Website.....	67
Figure 62: Timeline .....	74

# Table of Tables

Table 1: Web Scraper Listing Data .....	32
Table 2: Model Data.....	32
Table 3: Problems and Solutions .....	76

# 1. Abstract

Inside Airbnb is a project by Murray Cox, a digital storyteller, who visualized Airbnb data that was scraped by author and coder Tom Slee. The website offers scraped Airbnb data for select cities around the world; historically data is also available.

We were tasked with creating visualizations with listing data over Virginia and Austria to see what impact Airbnb was having on the communities in each respective region. The choice was Virginia and Austria because our team was familiar with both regions, with parts of our team being familiar with Virginia and other parts being familiar with Austria. The eventual goal is to expand past analysis of these 2 regions and expand further to say the rest of the United States. Since July 2019, Tom Slee has abandoned the script<sup>2</sup> to collect data. To collect data on Virginia and Austria, we needed to update the script to collect more recent data.

We began inspecting the script and found it was not collecting as much data as it once was. This was almost certainly due to Airbnb's website layout changing over time (a common nature of websites). After finding out how the script worked, we eventually found out the various problems related to the script and updated it to the new Airbnb website design. Doing so, we were able to get even more data than we thought possible such as calendar and review data. From there, we were able to begin our data collection process.

During all the time fixing the script, our team was making mock visualizations to be displayed on a website for easy viewability. Once data collection was complete, the data was transferred over to be used for these mock visualizations. We visualized many things such as how many listings a single host had, how many listings were in a given county, etc. The main visualization created was to see where all the listings for Airbnb were on the map. We displayed this on a map. We also made maps to visualize availability, prices, and the number of reviews. Further, we created pie charts and histograms to represent Superhosts, instantly bookable listings, and price distributions.

We expect that in the future the script and the data collected and visualized will be used by both future CS Students working on subsequent iterations of the project as well as Dr. Zach himself, our client.

## 2.Introduction

Airbnb has always been quite a controversial business. It is like a hotel combined with Uber. Airbnb has grown tremendously recently to where Airbnb just recently beat Hilton in US consumer spending<sup>1</sup>. However, although Airbnb is so popular, it does not face the same amount of regulation and taxing that traditional hotels face because it is classified as a sharing room service. This is a problem especially when it comes to safety and health regulation. Airbnb could be putting a lot of people at risk because nobody is regulating the properties to make sure that they are safe for the public. It is hard to research more about Airbnb listings since Airbnb purposefully hides their data, but we can get around this by web scraping.

Currently, there exists collected data from Tom Slee<sup>2</sup> that is displayed on Inside Airbnb's Website<sup>3</sup>. This website displays many visualizations to show how Airbnb is impacting local cities. It visualizes listings information such as where the listings are, how many listings there are, availability of those listings, and more. Although this data is very useful to know, the data is currently limited to only the largest cities around the globe. This can be a problem as large cities may not be representative of the greater whole.

Over the past few years, big cities put regulations into place that limit short-term rentals to ensure that units stay on the housing market for year-long rent, to stop the increase in rents (less supply means higher prices). Simultaneously, cities want to tax short-term rentals similarly to taxes on hotels: every guest pays a transient occupancy tax (aka hotel tax, bed tax, overnight tax) that short-term rental hosts often should pay, but oftentimes skirt the taxman. These measures are easier to implement for big cities - some of whom are listed on Inside Airbnb. However, for smaller - often rural - municipalities, this is difficult, particularly as they do not have the relevant data.

Hence, while current Inside Airbnb data is useful for big cities to get a first idea of the short-term rental market in their jurisdiction, smaller cities and towns do not have this information. However, such smaller municipalities can also benefit from this data.

The goal is to scrape Airbnb data for the state of Virginia and Austria. Tourism is an important economic factor for both Virginia and Austria and they are close in size: Virginia



(42,774.2 sq mi), Austria (32,386 sq mi). Both areas have metro and rural mountainous areas, but only Virginia has a coastline.

With this data, we can more accurately see how Airbnb is affecting both the communities in Virginia and Austria as a whole. Additionally, we can expand on this data and move forward to exploring how Airbnb affects the rest of the USA.

## 3. Requirements

### 3.1 Data

Following is a list of data that our client required us to get. This was specified at the beginning of the project before knowing what data was able to be obtained.

- Number of properties
  - This is how many Airbnb listings that a single host owns.
- Mean review score per property
- Number of reviews per property
- Mean daily rate per property
- Additional property fees (e.g., cleaning fee)
- Geolocation clusters - used zip code boundary maps to identify into which zipcode (and thus city, county) a property falls
- (Stretch Goal) Top 50 words in reviews by region, possibly by county (Virginia) or Federal state (Austria) AFTER word stemming and truncating

We were able to obtain all of the required data for this project. However, as we learned more and more about the project, we worked with our client extensively to find out if newly found data would be useful to him. He elected for the approach, “the more the merrier”, so we essentially tried to get as much data as we could on top of the above requirements. Our client wanted to learn as much about the properties as possible and the above list only shows what he is MAINLY concerned with.

### 3.2 Graphs

Two main tasks are related to the graph implementations. The first one is to show how the distributions of Airbnb houses change over time among all the counties in Virginia and Austria. In addition, looking for possible correlations existing in the data is another task. By

visualizing the data with different types of charts, maps, and tables, we can get more insights into the data.

### 3.3 Website

Since the goal for this project is to scrape the data from the website and generate visualizations of these data, the usage of the website is mainly about the display and organization of the graphs by location. The user should be able to quickly access a single graph by searching from bigger places to smaller places.

Furthermore, for usability, the website should be simple in layout. A complex structure will affect the user experience. On the other hand, since the graphs may contain lots of information and the space on the website is limited, there should be a way for the users to zoom in on the graphs and yet have all details of the graphs maintained.

The website should also update the graphs based on changes in the new datasets, which means that when the datasets change, the graphs on the website will change correspondingly. If there exists a county that does not have any data in an updating cycle, we should leave it empty.

### 3.4 Automation

The main goal of the project is for the clients to easily analyze the data. They should not spend too much time studying how to use the software and the technology necessary to automate the current script. However, it would be ideal to have a more automatic system to update the most current data on the website and in our datasets in a seamless fashion. Our ideal goal is to automatically scrape the data once a week, and update the dataset as well.

## 4.Design

Three modules have been involved in the Airbnb project: Web Scraping Module, Data Visualization Module, and Web Representation Module. These modules work sequentially to retrieve the data from the Airbnb official website, represent the data by visualizing it into various types of charts, maps, and tables, and display the visualizations on our website.

### 4.1 Web Scraping Module

Web scraping is the process of extracting information from a website through a programmed script that primarily finds the information in the HTML of the website being scraped.

When starting the project, we were starting with a pre-existing web scraping script that we knew fulfilled our tasks. The only problem with this script was that it was outdated and did not fulfill the scope of what we wanted to know about Airbnb listings. Thus, we decided that to improve the existing script, we must first understand the script.

To understand the script, initially we needed to go about setting the script up on our computers. The existing script already depended on the use of a database and Python 3.4 or later. The use of a database was completely up to our own choice to be used in tandem with the script, but we decided on using PostgreSQL 12.2. We decided upon using PostgreSQL because this was the database of choice for the user of the existing script. To increase the likelihood of everything working, PostgreSQL was chosen for compatibility concerns. Additionally, PostgreSQL was chosen because it is considered a relatively low-difficulty database to be working with and some of our team members had a little experience using it. PostgreSQL also has a PostGIS extension which would be useful for geographic visualization purposes. Almost the whole team knew how to use SQL, and learning to use PostgreSQL with the script seemed like the obvious choice.

For running the script, Python 3.4 or later was needed. On top of that, the script had a number of extensions and dependencies that also needed to be downloaded. To get over this hurdle, we found that Anaconda, a package including Python and a number of other packages, would be of use. In particular, the existing script required the use of lxml and psycopg2. Both of

these could be obtained using Anaconda. This would make running the script and getting everything set up even simpler.

Going beyond the setup, we also had to design the data processing itself. The existing script left some room for flexibility. There were a couple of ways we could have gone about this. In particular, we considered using something like Selenium to scan the HTML of the webpage. This would be quite complicated. Selenium would be necessary since the Airbnb webpages are dynamic, so traditional web scraping methods would not work. Additionally, this method would be very fragile as page layouts tend to change quite frequently.

Another approach would have been to use network requests. In this approach we could visit the Airbnb webpage, use a network scanning tool such as the Google Chrome developer tools, and then analyze the network requests that Airbnb uses to display listing data. This is the approach that we will be taking. Although initially finding the data in this fashion may be a little difficult, once we figure out the request URL that Airbnb uses, we should be able to recreate the requests using a package such as the Requests package in Python. This method would, in addition to being easier, also be more resilient to changes on Airbnb's side.

The last piece for data collection involves how we request the data and how we will save the data. For how we will get data, Airbnb has a map on their website. On this map we are able to get listings by zooming in or zooming out and moving around the map. Thus, we can take advantage of this map by "simulating" the coordinates of where we want listings to be displayed on the map via specifying a bounding box. However, there is also a limit to how many listings can be displayed on the map at a given time. Thus, as a workaround, we can get listings by continuously zooming in on different areas on the maps for Virginia and Austria.

## 4.2 Data Visualization Module

Data visualization is the act of displaying extracted data in a variety of aesthetic ways including but not limited to graphs, charts, maps, representations, etc. We not only produced charts for Virginia and Austria as a whole, but also have created charts for states in Austria and counties in Virginia.

To create visualizations, a variety of choices are available to choose from, such as charts, scatter plots, maps, and time series. Each of them has its own focus on how to present extracted data.

a. Geopandas with Shapefiles

One of the major tasks in the visualization section is to create distribution maps of Airbnb houses for Virginia counties and Austria. Geopandas, a powerful Python package, achieves our goal by plotting the latitudes and longitudes of Airbnb houses in the data as dots on shape maps from shapefiles.

b. NLTK Toolkits

Word cloud maps are produced to make an analysis of the reviews from users. Several NLTK tools have been applied during the process, such as stop lists, word frequency, and the removal of people's names.

c. Interactive Charts

Some visualizations produced by Pyecharts (a Python package) are interactive, which allows users to cancel or add elements by clicking on the buttons at the top of the graphs.

### 4.3 Web Representation Module

To achieve the requirement of letting the choices determine the graphs on a certain page, we designed a selection mechanism for users to easily select the target county. The website will display the graphs based on the choices. There will be three boxes. Each box will stand for a range of geography, for example the first box will stand for the country, the second box will stand for the state, and the third box will stand for the county. In addition, the second box will only show the state in the country chosen from the first box and the third box will only show the county in the state chosen from the second box. Instead of a list of counties, this hierarchy structure will provide users a clear way to access target counties.

Based on the graphs we generated, the smallest area of the United States will be the county and the smallest area of Austria will be the state. So if a user chooses Austria, the third box will only display the state chosen in the second box.

On the other hand, we will have a special option to let the user access the graph of the whole Austria and whole Virginia. This option is the last option in each drop box. For example, there will be a choice called "Virginia" in the third box if the user chooses Virginia in the second box and there will be a choice called "WholeAustria" in the second box if the users choose Austria in the first box.

Once the users choose a specific county, the website will change the graphs based on the choice.



Graph

Graph

Graph

Graph

Graph

*Figure 1: Graph Page*

As in **Figure 1**, five to seven graphs will display. Each graph is loaded from our graph database, which means once the new data is scraped and the new graph is generated, the new graphs will be automatically loaded into this page<sup>4</sup>.



# 5.Implementation

## 5.1 Data Collection

The Data Collection process was implemented paying attention to two core parts: the act of collecting and the data itself. In regard to collecting, we were not starting from scratch. Since we started with Tom Slee's previous code<sup>4</sup>, we decided to build upon it<sup>5</sup> instead of starting from scratch. This is because we decided to run the script to see what its base functionality is. Additionally, since we were not well versed in web scraping, the script provided a lot of existing functionality. Most of the functionality was in getting individual listing data and pages of listings. This proved to be very useful as it was quite complicated to implement a search for listings ourselves.

We get the data by doing a recursive bounding box search. So we initially start with user-entered coordinates that make up a square around an area. This area is then recursively searched by breaking up that box into 4 more boxes. The areas keep breaking down until we reach the base case of not finding any listings. This is where the boxes stop breaking down. As you can imagine, recursively, this process yields every page of listings and hence every individual listing thereafter.

As an example, let's focus on how we obtain data for a listing in Washington DC, initially stemming from a search for Virginia.

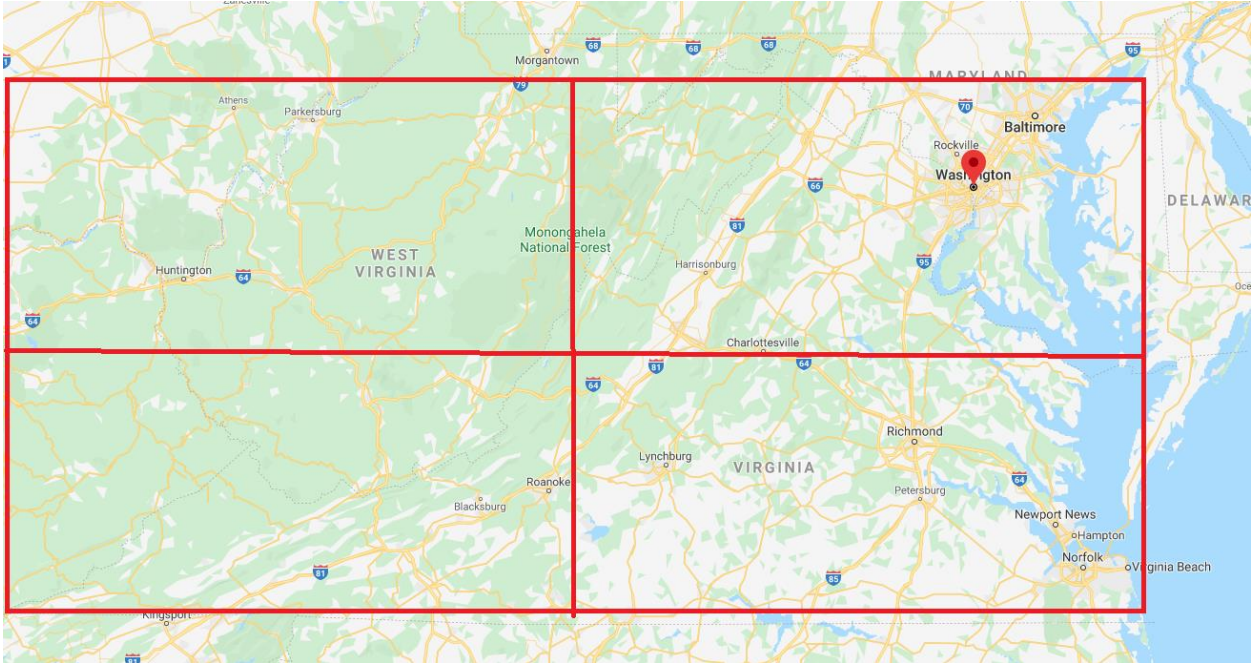


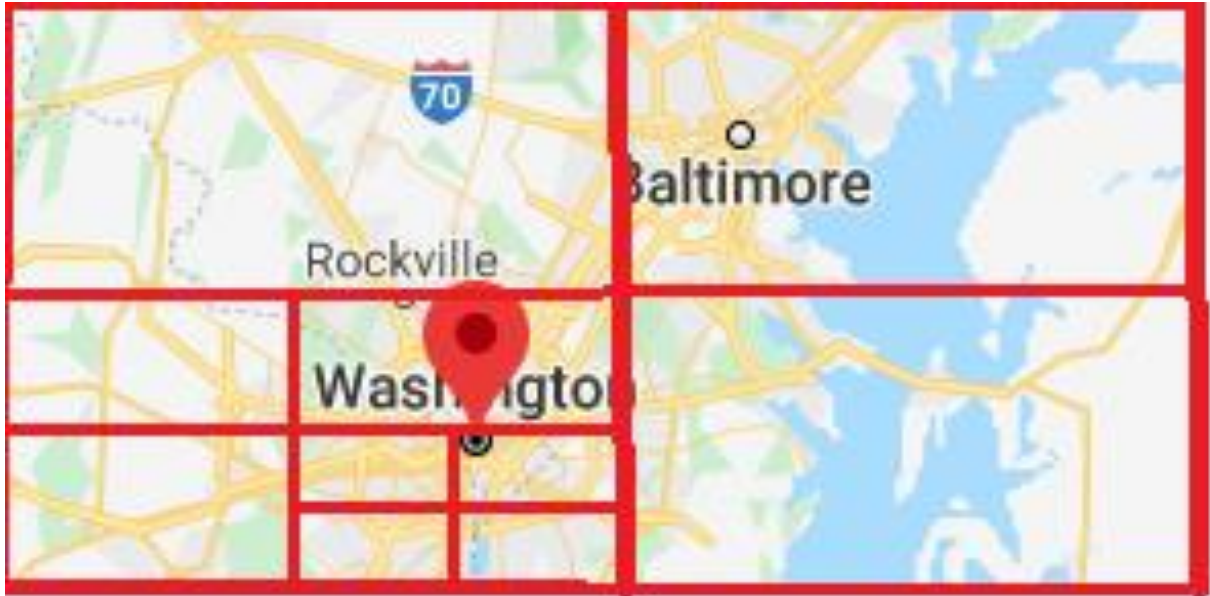
Figure 2: Bounding Box Initial Step

We first specify a bounding box around Virginia shown in **Figure 2**. This initial, specified box is depicted above as the outer red box surrounding all of Virginia. Every listing in this outer box will be searched. This box is then split into 4 boxes, where we then search each of the 4 boxes. Once the script reaches the box on the top right, we search that box.



Figure 3: Bounding Box Second Step

The box gets split into 4 more boxes, shown in **Figure 3**, where each of the boxes will then be searched. For example, once we reach the box on the top right, we search the box and split the box into 4 more boxes.



*Figure 4: Bounding Box Steps (continued)*

We continually and recursively keep doing this process where we visit a box, split that box into 4 more boxes, and visit the 4 boxes created. This is depicted **Figure 4**. At each step, we save any listings Airbnb returns for the region and we continue for as long as we keep finding listings.

Through getting each individual listing, we get the listing's 'ID'. This ID is essential as it is how we distinguish between any listings, and links all the data for that listing. We decided to do it this way because linking review data, calendar data, and listing data -- all for one listing (because it is just for one listing) -- is intuitive. However, we split the 3 different types of data into 3 different tables because each data can have a drastically different number of rows, i.e., many rows belong to a single listing for review and calendar data, while only a single row is used per listing for listing data. Now that we know how data collection is implemented, what is the data we are getting?

The data we are getting is broken down into three tables: listing data, review data, and calendar data. For listing data, we get:

- The room's ID
- The host's ID
- The room type, e.g., entire home, apartment, hotel, room in a home
- Listing address
- Number of reviews
- The rating
- How many guests can stay in the room
- Number of bedrooms
- Number of bathrooms
- Price per night (and currency)
- Minimum number of stays required to book
- The location of the room (longitude and latitude)
- Name of the listing
- How many days of the year is the listing currently available to be booked.
- If the listing is instant-bookable
  - This means a user can book the room instantly without having to go through any interviews with the host.
- If the host is a 'super host'
  - Super hosts have high ratings throughout their listings.

For review data we get:

- Listing ID
  - Used to identify what room this belongs to
- Review ID
  - Used to identify the individual review
- Date review was created
- Reviewer ID
- Reviewer Name
- Rating

- Comments made by the reviewer.

This data is for every single review found.

For calendar data we get:

- Listing ID
  - Used to identify what room this belongs to
- Date
- Price of the room on this day
  - Room prices vary from day-to-day.
- Minimum nights needed to book for this day
- Maximum nights you can book for this day
- Availability of this room on the day.

This data is collected for every calendar day (in a year) for a listing that was found.

Calendar data, survey data, and review data all are collected with respect to a survey ID. Every time we collect data for a user-entered boxed area, we assign a survey ID to it that we also tell the user. It is the user's responsibility to remember the survey\_id in regards to what they intend the box to be used for. We cannot know what the user would use the survey ID for, so we require the user to remember the purpose of each survey. In this way, if a user conducts a search for, say, a box around all of Virginia, the user must remember that the box is for Virginia. We cannot know what SPECIFICALLY the user is searching for just from coordinates alone.

## 5.2 Visualization

All the visualizations are produced by Python packages, such as Matplotlib, Geopandas, and Pyecharts. Four steps were involved to produce those graphs.

- Choose the proper package based on the data.
- Select several fields in the data which are related.
- Normalize the selected data to a proper format.

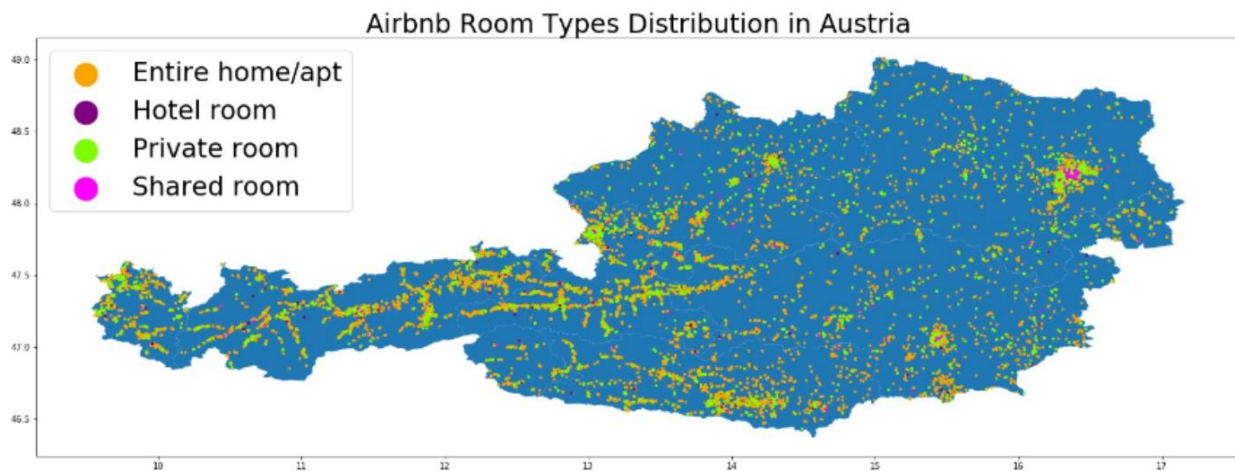
- Determine the type of chart and plot it.

### 5.2.1 Visualization for Austria and Virginia

The visualization contains several topics for both Austria and Virginia:

- A geomap for room distribution
- A geomap for the number of reviews' distribution
- A geomap for availability days in a year distribution
- A geomap for prices distribution
- A histogram chart for prices
- A pie chart for Superhost listings
- A pie chart for instantly bookable rooms

We will show all the visuals for all the areas in Austria as examples to illustrate.



*Figure 5: A Geo Map for Room Types' Distribution in Austria*

The distribution of four room types (Entire home/apt, Hotel room, Private room, Shared room) is plotted in a Geomap in **Figure 5**. Each different color represents one type of room.

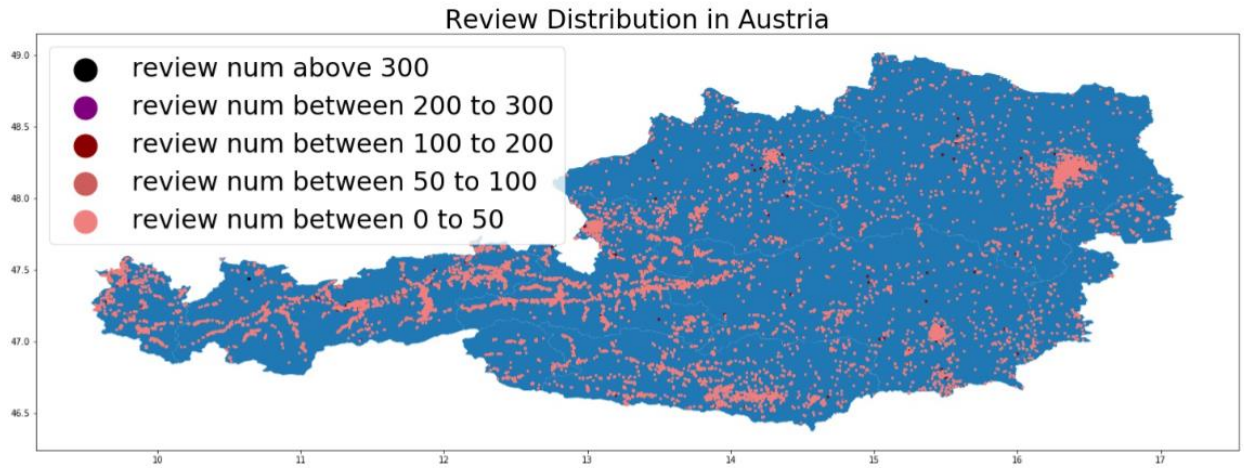


Figure 6: A Geo Map for Number of Reviews' Distribution in Austria

The distribution of how many reviews that each Airbnb house can get is plotted in a Geomap in **Figure 6**. Each different color represents one category of the number of reviews.

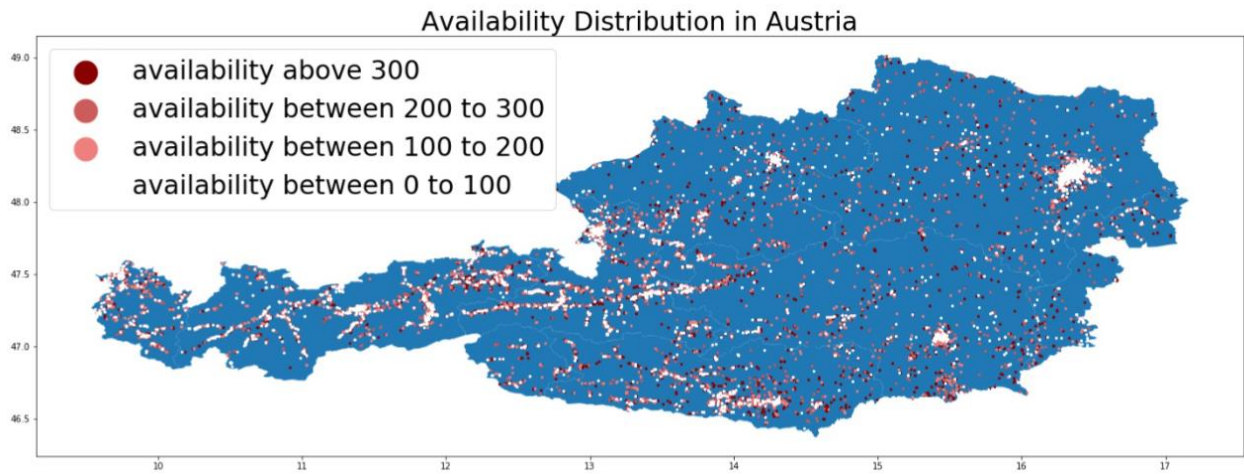


Figure 7: A Geo Map for Availability Distribution in Austria

The distribution of how many days in a year that each Airbnb house is available is plotted in a Geomap in **Figure 7**. Each different color represents one category of the number of reviews.

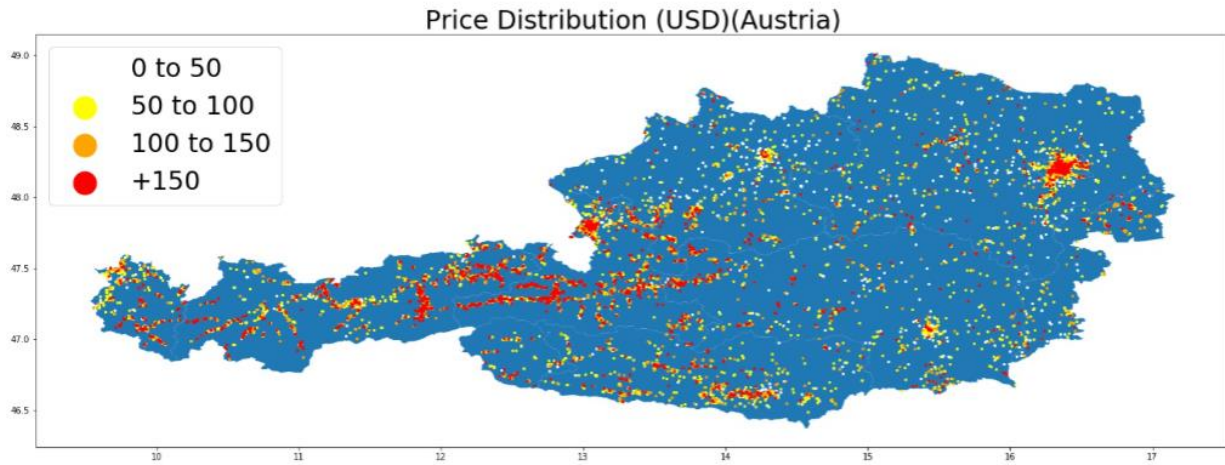


Figure 8: A Geo Map for Price Distribution in Austria

The price distribution is plotted in a Geomap in **Figure 8**. Each different color represents one category of prices.

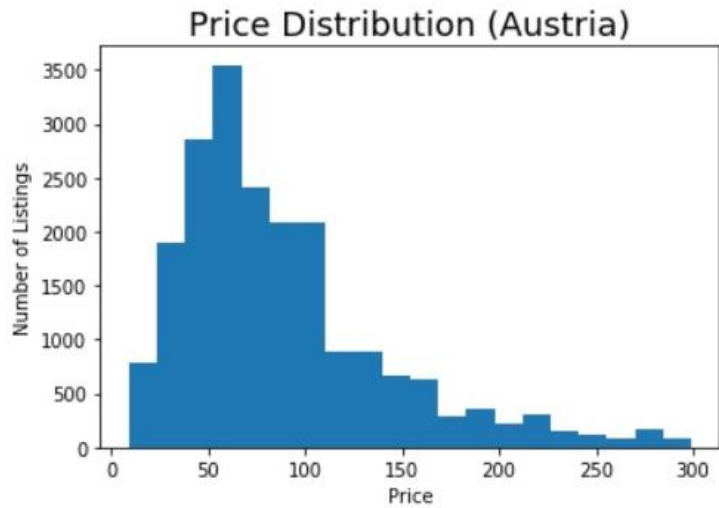


Figure 9: A Histogram Chart for Price Distribution in Austria

**Figure 9** shows a histogram chart for price distribution and the number of corresponding listings in Austria.





*Figure 10: A Pie Chart for Instant-Bookable Listings in Austria*

**Figure 10** shows a pie chart for presenting the relationship between instant-bookable listings and non-instant-bookable listings in Austria



*Figure 11: A Pie Chart for Superhost Listings in Austria*

**Figure 11** shows a pie chart for presenting the relationship between superhost listings and non-superhost listings in Austria

## 5.2.2 Visualization for Austria States and Virginia Counties

The visualization contains several topics for each Austria state and Virginia county:

- A geomap for room distribution
- A geomap for the number of reviews' distribution
- A geomap for availability days in a year distribution
- A geomap for prices distribution
- A histogram chart for prices
- A pie chart for Superhost listings
- A pie chart for instantly bookable rooms
- A word cloud map for reviews

- A scatter plot for the number of houses versus the number of people who own that many

We will choose Salzburg, which is one of the states in Austria, as an example to display the visualizations.

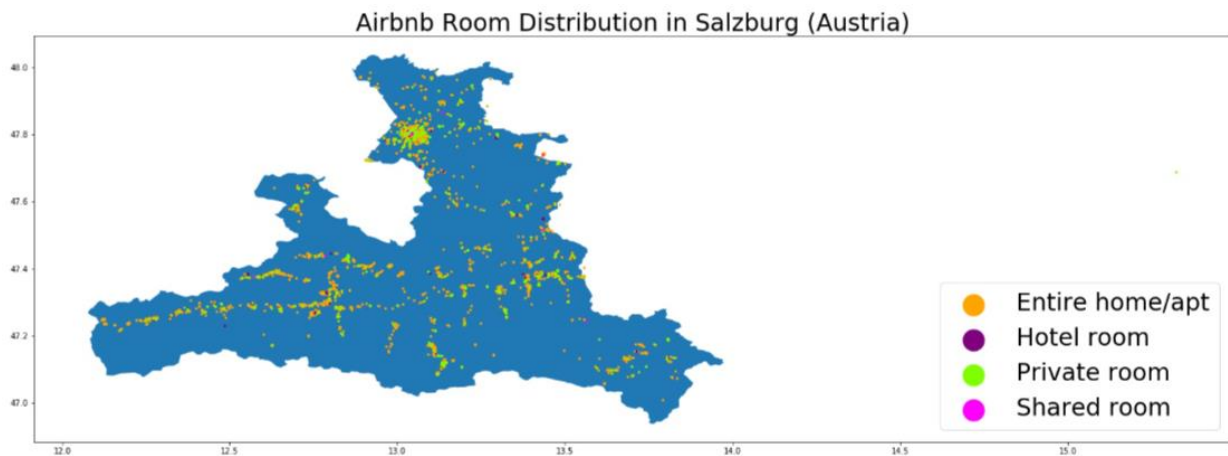


Figure 12: A Geo Map for Room Types' Distribution in Salzburg

The distribution of four room types in Salzburg (Entire home/apt, Hotel room, Private room, Shared room) is plotted in a Geomap in **Figure 12**. Each different color represents one type of room.

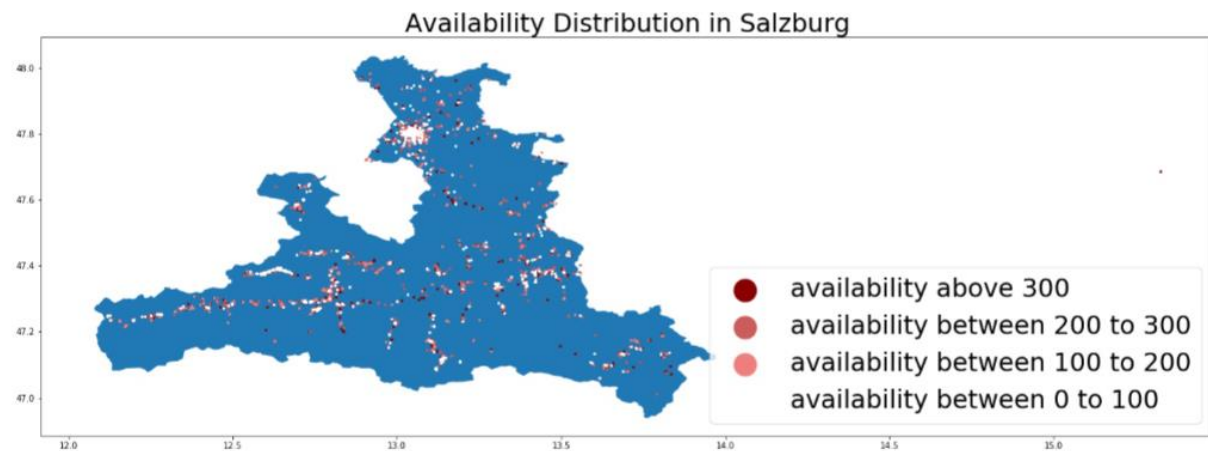


Figure 13: A Geo Map for Availability Distribution in Salzburg

The distribution of how many days in a year that each Airbnb house is available is plotted in a Geomap in **Figure 13**. Each different color represents one category of the number of reviews.

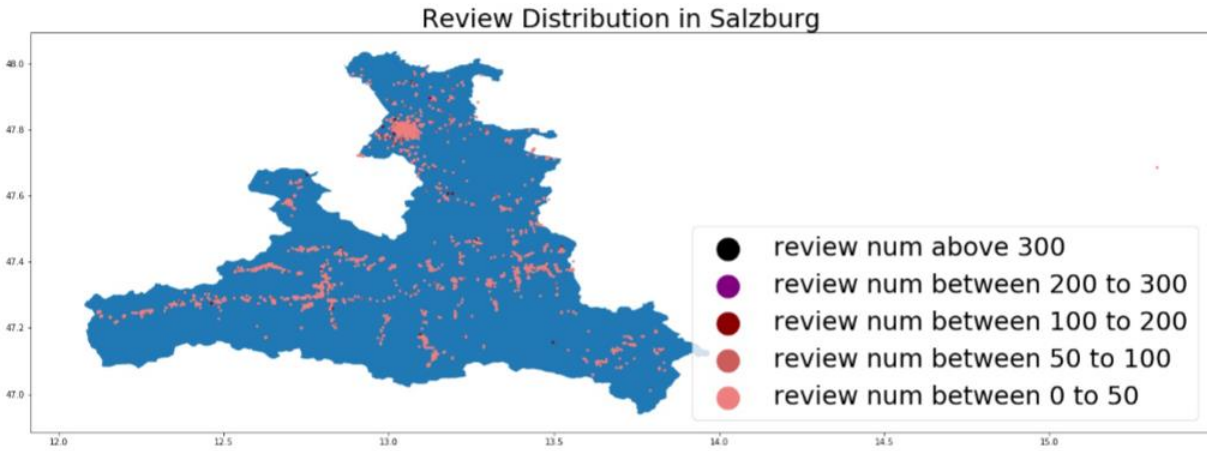


Figure 14: A Geo Map for Number of Reviews' Distribution in Salzburg

The distribution of how many reviews that each Airbnb house can get in Salzburg is plotted in a Geomap in **Figure 14**. Each different color represents one category of the number of reviews.

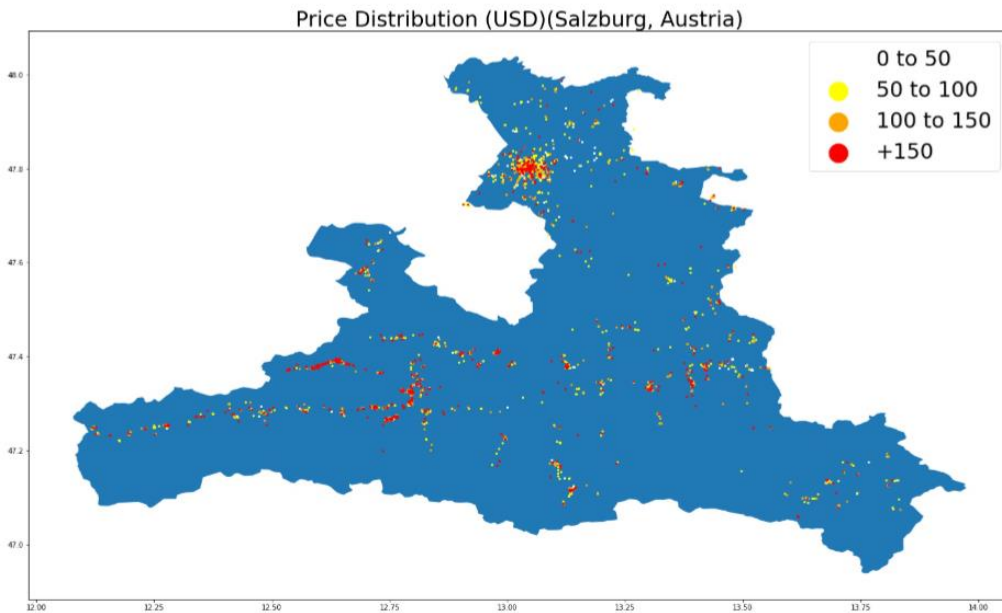


Figure 15: A Geo Map for Prices' Distribution in Salzburg

The price distribution in Salzburg is plotted in a Geomap in **Figure 15**. Each different color represents one category of prices.

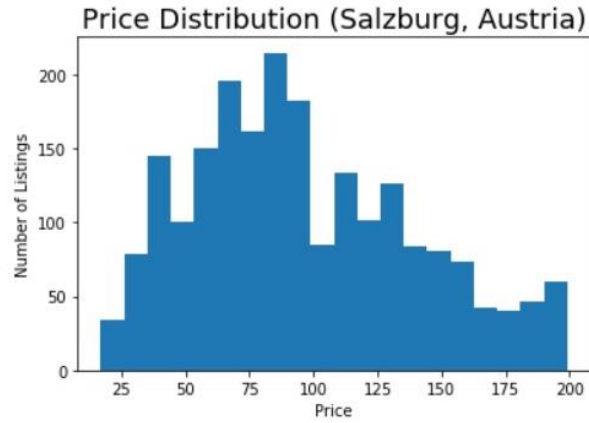


Figure 16: A Histogram Chart for Price Distribution in Salzburg

**Figure 16** shows a histogram chart for price distribution and the number of corresponding listings in Salzburg.

Instant-Bookable Listings (Salzburg, Austria)

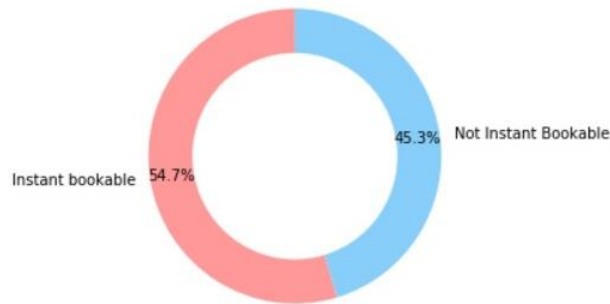


Figure 17: Pie Chart for Instant-Bookable Listings (Salzburg)

**Figure 17** shows a pie chart for presenting the relationship between instant-bookable listings and non-instant-bookable listings in Salzburg

### Superhost Listings (Salzburg, Austria)

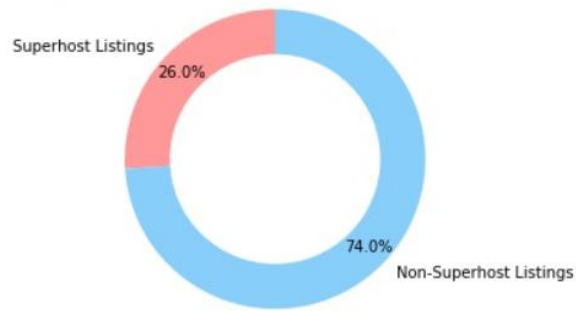


Figure 18: Pie Chart for Superhost Listings (Salzburg)

Figure 18 shows a pie chart for presenting the relationship between superhost listings and non-superhost listings in Salzburg

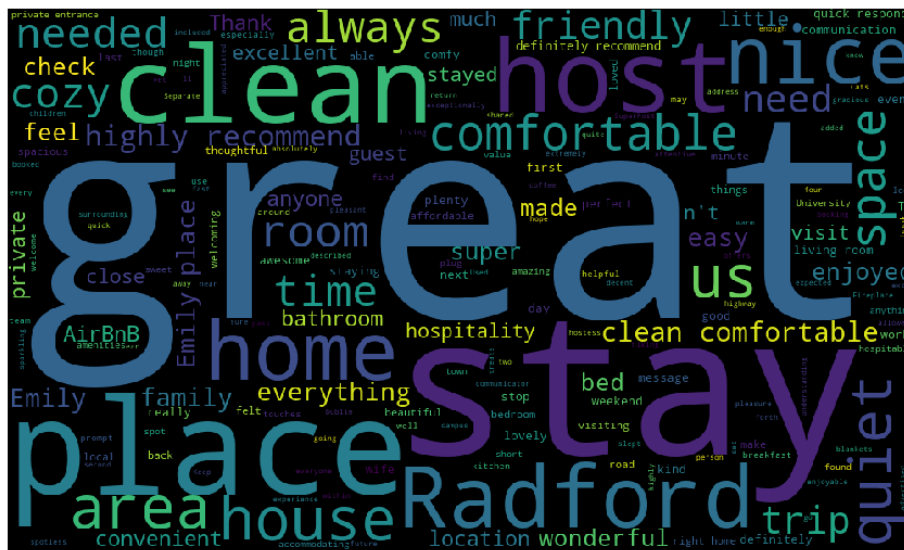


Figure 19: A Word Cloud Map for User Reviews in Salzburg (Austria)

Figure 19 is a word cloud map produced by analyzing user reviews. Words with higher frequency look larger in the graph. It helps those who put up their properties on Airbnb to have a bigger picture of users' feelings about Airbnb in Salzburg.

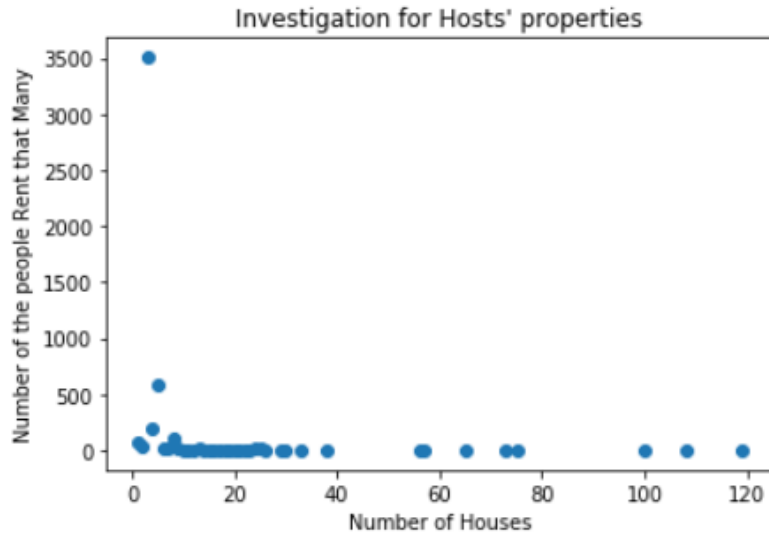


Figure 20: A Scatter Plot of Hosts' Properties in Salzburg

**Figure 20** shows the relationship between the number of houses and the number of people who own that many. It turns out the main range of the number of properties that a person can own is from 1 to 40.



Figure 21: A Scatter Plot of Price vs Number of Reviews in Salzburg

Figure 21 shows the relationship between the prices and the number of reviews. It turns out that Airbnb properties often get more reviews when they have fewer prices.

### 5.3 Website

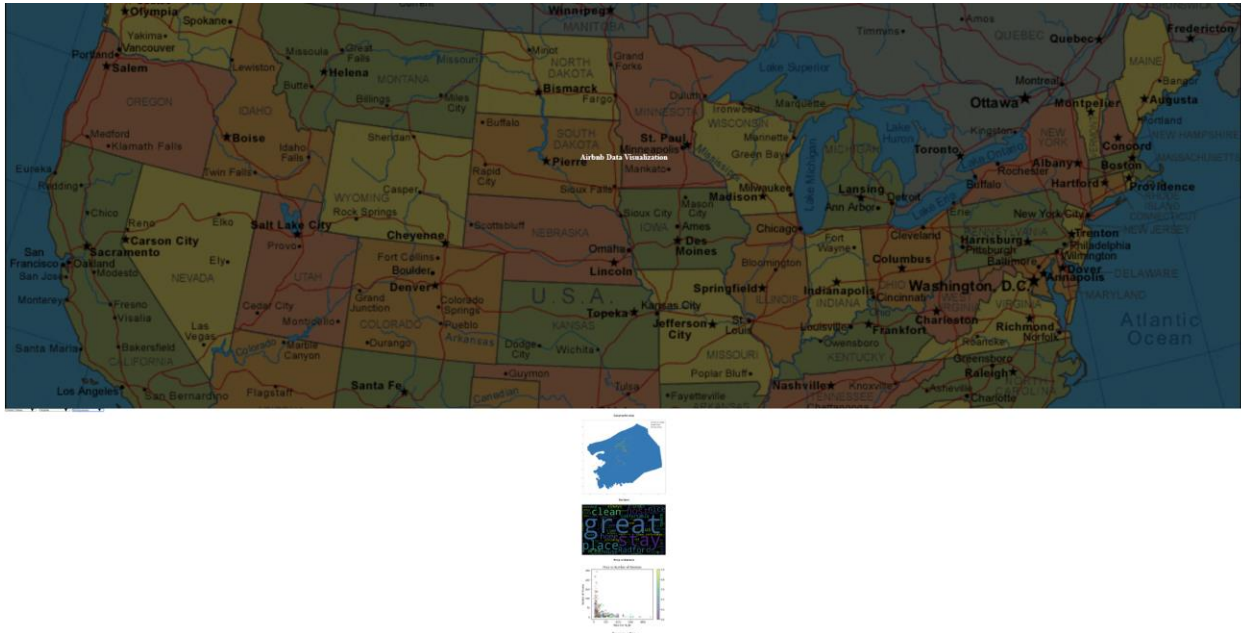


Figure 22: Website Layout

In the implementation, we arrange the layout of the images. As shown in Figure 22, all the graphs are shown in the center and are of similar size. This figure shows a big picture with our title of website. Below the big picture are our three boxes which provide the way to access each graph. Below the three boxes are our graphs of the county or state. The graphs may be dynamic or static.

## 6. Testing/Evaluation/Assessment

room_id	host_id	room_type	latitude	longitude	name	availability_365
25909468	1.86E+08	Private room	37.11399	-80.587	Bedroom number three and living room.	330
26746561	1.74E+08	Entire home/apt	37.13494	-80.5611	â~...Private Bungalow, quick walk to RU/drive to VTâ~...	228
23141176	1.72E+08	Entire home/apt	37.17764	-80.4401	Private Studio-near VT, RU, Aquatic Center & I-81	233
17073936	89581275	Entire home/apt	37.10721	-80.5557	THE RANCH - 10 min to RU, 20 min to Virginia Tech	306

**Table 1:** Web Scraper Listing Data

room_id	host_id	room_type	latitude	longitude	name	availability_365
<b>15883</b>	<b>62142</b>	<b>Hotel room</b>	<b>48.24262</b>	<b>16.42767</b>	<b>b&amp;b near Old Danube river</b>	<b>350</b>
<b>38768</b>	<b>166283</b>	<b>Entire home/apt</b>	<b>48.21823</b>	<b>16.37926</b>	<b>central cityapartement- wifi-nice neighbourhood</b>	<b>193</b>
<b>40625</b>	<b>175131</b>	<b>Entire home/apt</b>	<b>48.18434</b>	<b>16.32701</b>	<b>Near Palace SchÃ¶nbrunn, Apt. 1</b>	<b>341</b>

**Table 2:** Model Data

**Table 1** is an example of the data our web scraper is getting. To test to make sure our data is correct, we look up the ID for the listing and compare the values we obtain for that listing to the listing on the website. Additionally, our visualization team is using sample data from **Table 2**. Thus, it is important for our web scraper to be using the same format as this model data. We then compare the two different data sets. As you can see, **Table 1** and **Table 2** follow the exact same format, allowing for an easy transition from the model data to our scraped data. **Table 1**



does not show every field, as there are many, but to verify if it is correct, we need to compare the listings in **Table 1** to the listings that Airbnb displays on their websites. To accomplish this, consider row 2 from **Table 1**.

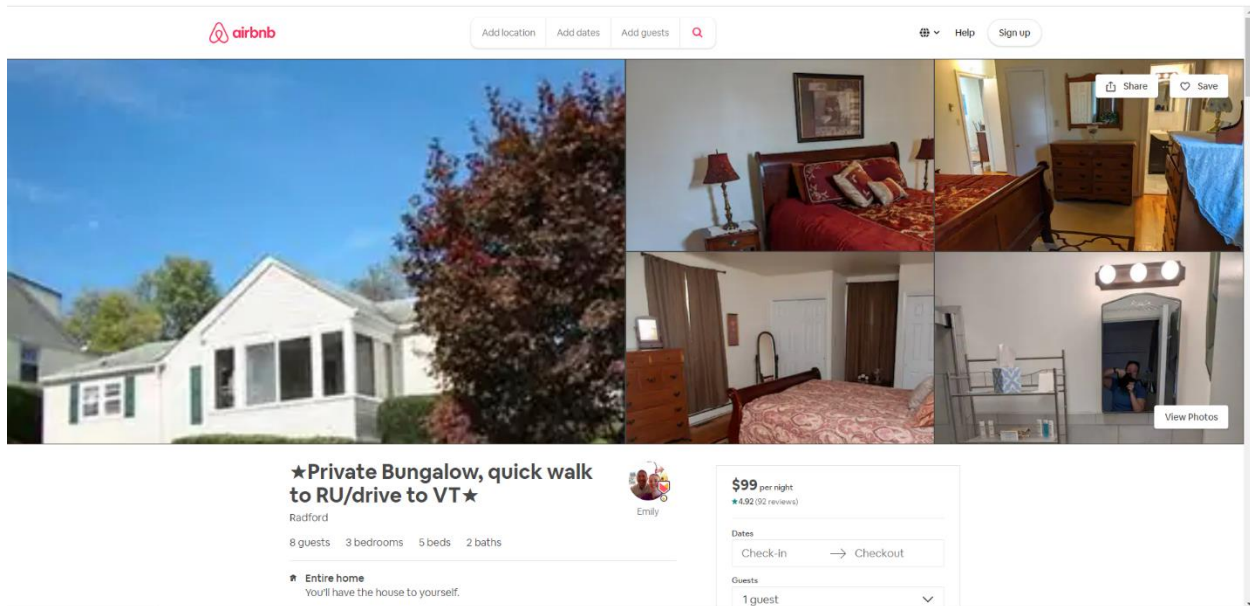


Figure 23: Airbnb Website Listing

If we look at the ID for the listing in row 2 of **Table 1**, we can see its ID is “26746561”. We can then look up this listing using the following URL:

“https://www.airbnb.com/rooms/26746561”

As you can see, we just append the listing ID to the root URL:

“https://www.airbnb.com/rooms/”

This brings up a listing on Airbnb’s website which can be seen in **Figure 23**.

This is how we can cross-reference the listings from the scraper vs. the actual listings that Airbnb is showing. As we can see, the data matches (such as name and room type in **Table 1**).

# 7.Users' Manual

## Websites

For the website, the main function is to let the user select the target county or state. The website will display the related graphs based on their choices. We have these three boxes to implement this function. They are below the big banner on the website.

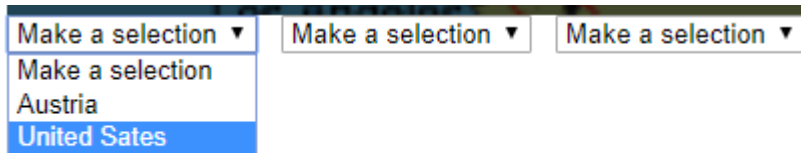


Figure 24: Country Selection

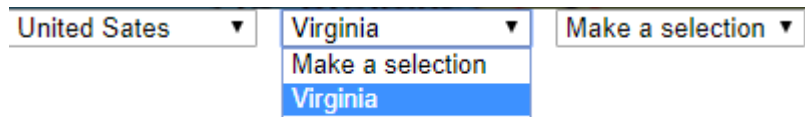


Figure 25: State Selection

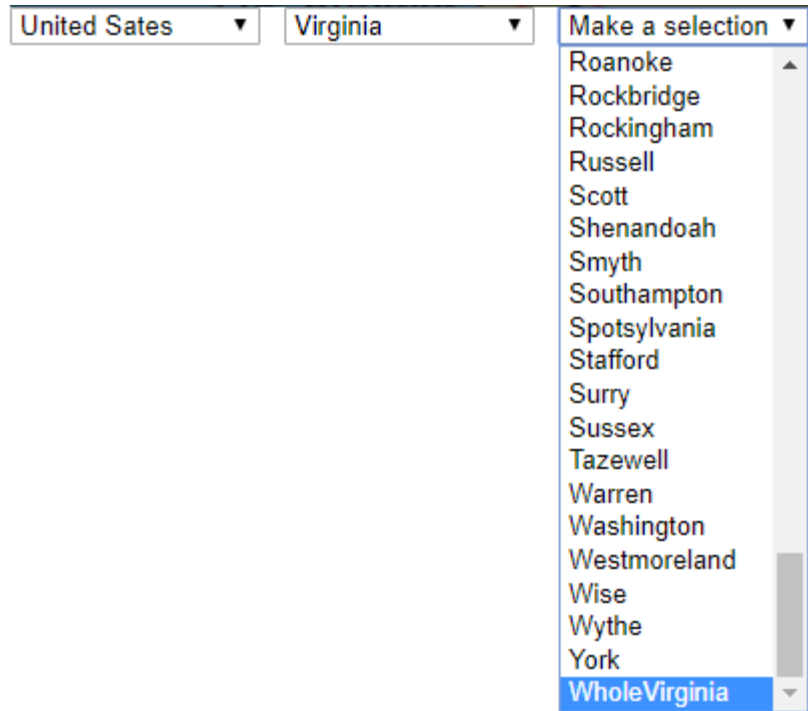


Figure 26: County Selection

As shown in **Figure 24**, each box is a drop-down box where users can drop down the choices by clicking the box. And every box stands for a range of areas. In the first box, we have Austria and the United States. Once the user selects a country, the second box will only display the states of the selected country. For example, in **Figure 24**, the user selects the United States. In **Figure 25**, the second box will only show the states in the United States. If the user selects Austria, the second box will only show the states in Austria. As shown in **Figure 26**, the third box will only display the county of Virginia.

We have a special option in the second and third box for users to access the graphs of larger areas such as the whole of Austria and the whole of Virginia. As shown in **Figure 26**, if the user chooses the last option, “WholeVirginia”, in the third box, the website will only display the graphs of the whole of Virginia. On the other hand, if the user chooses Austria in the first box, the ‘WholeAustria’ option will appear on the second box.

# 8. Developer's Manual

## Environment

- Airbnb.com
  - Data source
- Python 3.6
  - Data collection script
  - Data Visualization
- PostgreSQL / Excel
  - Data storage
- HTML 5
  - Front End

## Script:

The script we maintained and fixed mainly scraped the data from Airbnb. The data scraped includes reviews, dates, prices, room type, and so on. This data can be used to support tourism research for future studies. Users can look at both the data and visualizations to conduct a great analysis of how Airbnb affects local tourism over time.

To use the script, the first thing to do is to set up the environment in which to run it! Firstly, pull our GitHub<sup>5</sup>. You can use Windows 10, Mac, or Linux. There are two ways to run the script: using Docker to configure everything automatically or just configuring the database and script yourself manually. The question is, which one to use? Well, like always it comes down to personal preference and what you already know, so first let us analyze how to use the script with Docker.

## 8.1. Docker Instructions

To use the script with Docker, you can use either Mac, Linux, or preferably Windows 10 Pro, Enterprise, or Education. This is to run Docker Desktop which makes installing Docker very easy. Windows 10 Home does work as well, but the installation is quite tricky as now you have

to install Docker Toolbox instead. To install Docker Toolbox, visit its installation page<sup>6</sup>. Mac can also run Docker Desktop. To install Docker Desktop, visit Docker Desktop's Installation page<sup>7</sup>. Once you have Docker and pulled the repository, continue as follows.

### 8.1.1. Running Docker

First, using a terminal, change your directory to 'airbnb/docker' with 'airbnb' being the root of the repository you pulled.

Once in the docker folder, run the following command to start up the two Docker containers (database and survey script). The -d starts them in the background.

```
docker-compose up -d
```

To stop running the containers:

```
docker-compose stop
```

To stop running the containers and RESET all data stored and progress that the container made run:

```
docker-compose down
```

### 8.1.2. Running a Survey with Docker

First access the database with:

```
docker exec -it airbnbcollector-db /bin/bash
```

This puts you into the database container.

On initial startup, you need to apply the database schema. This creates all the tables needed.

To apply the database schema, type the following command:

```
psql -U airbnb airbnb < sql/schema_current.sql
```

Type 'exit' to exit the container.

Run this to access the script:

```
docker exec -it airbnbcollector /bin/bash
```

NOTE: This command is very similar to the command for accessing the database, but it is different!

This puts you into the container. Then cd into collector:

```
cd collector
```

This puts you into the folder with the script. Proceed to use the script as normal.

1. `python airbnb.py -asa "county_name_here"`

Go into database and:

```
UPDATE search_area
```

```
SET bb_n_lat = ?, bb_s_lat = ?, bb_e_lng = ?, bb_w_lng = ?
```

```
WHERE search_area_id = ?
```

Note: you can get coordinates using Klokantech's bounding box tool.<sup>8</sup>

The bottom left corner has a "copy and paste" section.

Use CSV formatting: Order goes: West, South, East, North

search\_area\_id should be shown upon using command from "1."

2. `python airbnb.py -asv "same_city_name"`

3. `python airbnb.py -sb [surveynumber]`

For command 3 do not include brackets but for commands 1 and 2 make sure the name is in quotes.

The script then continues to run!

### 8.1.3. Exporting Data and Accessing the Database with Docker

You can access the database by running:

```
psql -U airbnb airbnb
```

Proceed to update the survey\_area and export to CSV.

To export to CSV, run the following command:

```
\copy (SELECT * FROM room where survey_id = [surveynumber]) to 'data/name_of_file.csv' csv header
```

```
\copy (SELECT * FROM calendar where survey_id = [surveynumber]) to 'data/name_of_file.csv' csv header
```

```
\copy (SELECT * FROM reviews where survey_id = [surveynumber]) to 'data/name_of_file.csv' csv header
```

This will export the data into a CSV file in the data folder.

To quit out of psql, type \q

Then you can exit out of the container by typing exit.

## 8.2. Non-Docker Instructions

If you do not want to use Docker, you will have to manually download PostgreSQL and Anaconda. The following are instructions on how to set up the script without Docker:

## 8.2.1.Setup

### 8.2.1.1.Installing PostgreSQL

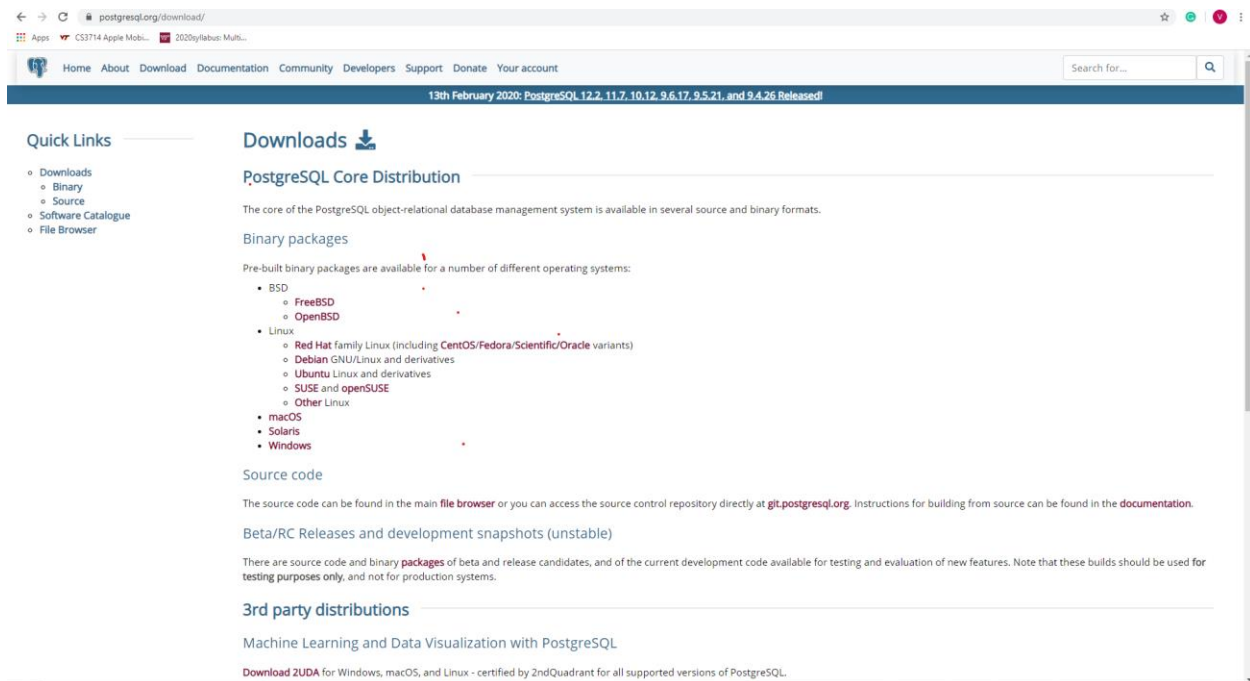


Figure 27: PostgreSQL Download Page

First visit the PostgreSQL download page<sup>9</sup> to download PostgreSQL. The download page is shown in **Figure 27**. Then click on the respective link for your operating system. We used Windows 10.



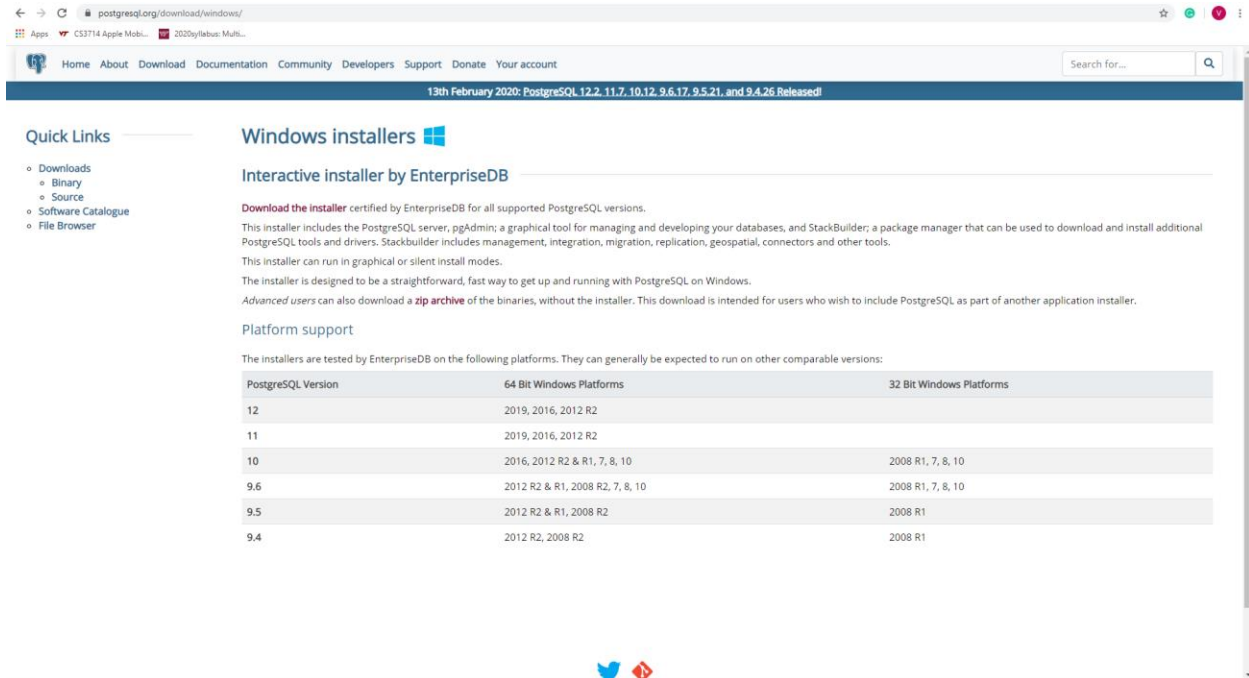


Figure 28: PostgreSQL Windows Download Page

Shown in **Figure 28**, for Windows, there is an installer. Click the link “Download the Installer”.

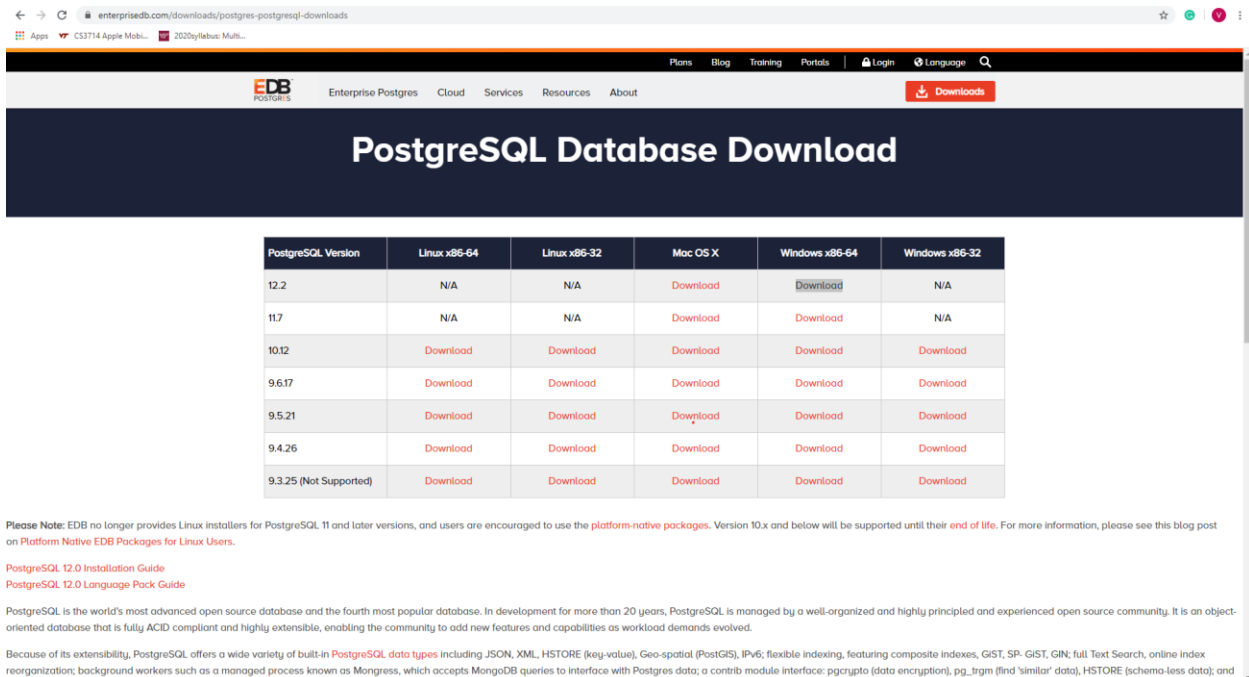


Figure 29: PostgreSQL Version Download Page

Then download the latest version; at the time this report was written, for Windows and Mac, it is 12.2. This is shown in **Figure 29**.

The download should then begin upon clicking the link. Then run the downloaded file to begin installation.

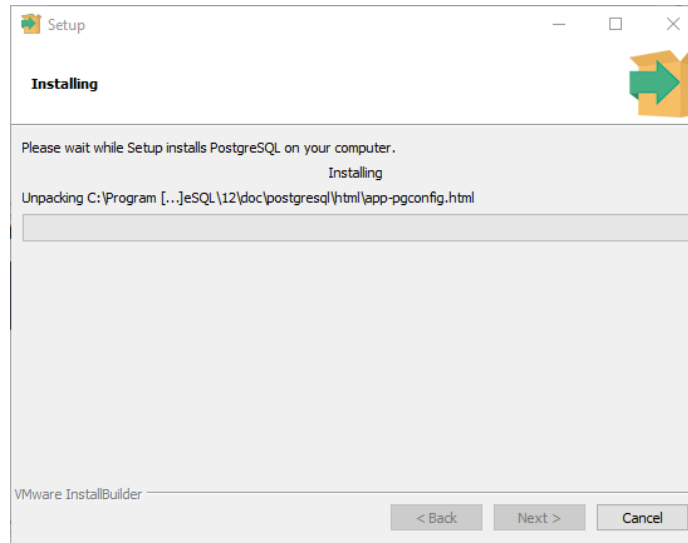


Figure 30: PostgreSQL Installer

Once the installer is open, continue pressing next until you get to the screen in **Figure 30**. If this is your first time installing PostgreSQL it will ask you to set a username and password. Make sure you remember what the username and password you choose is! You will have to use it later! For this example my username and password is both: postgres

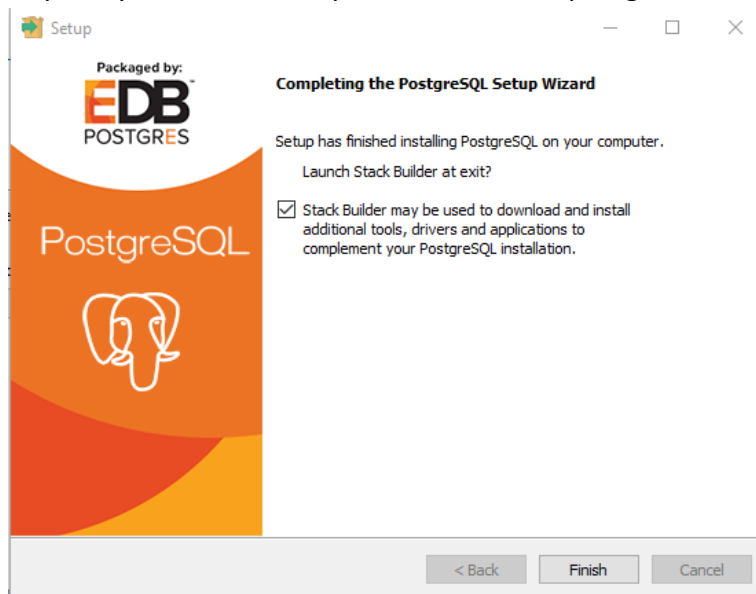


Figure 31: PostgreSQL Installer Finish Page

Upon completing the installation, make sure to leave Stack Builder checked. See **Figure 31**. Then press "Finish".

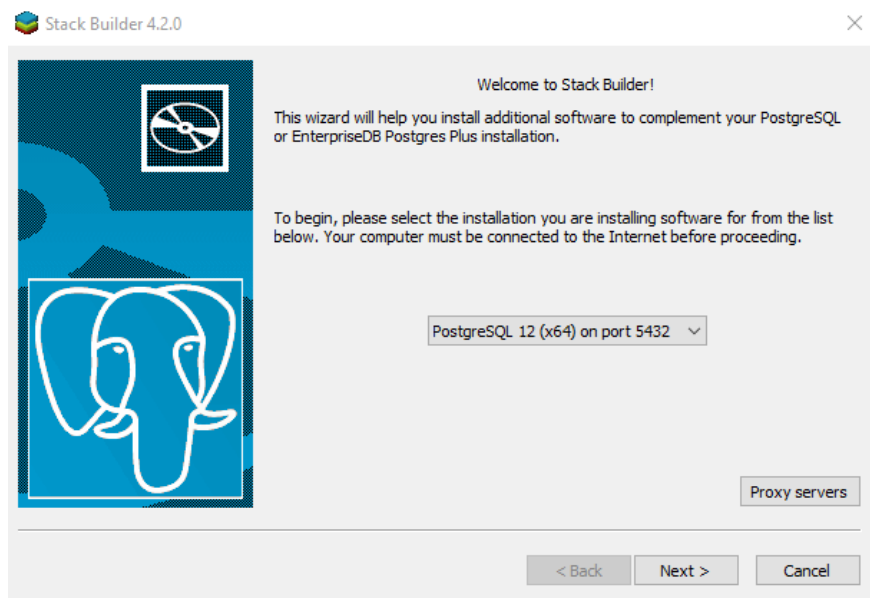


Figure 32: Stack Builder Wizard

Select PostgreSQL 12 under the drop-down menu and press next as shown in **Figure 32**.

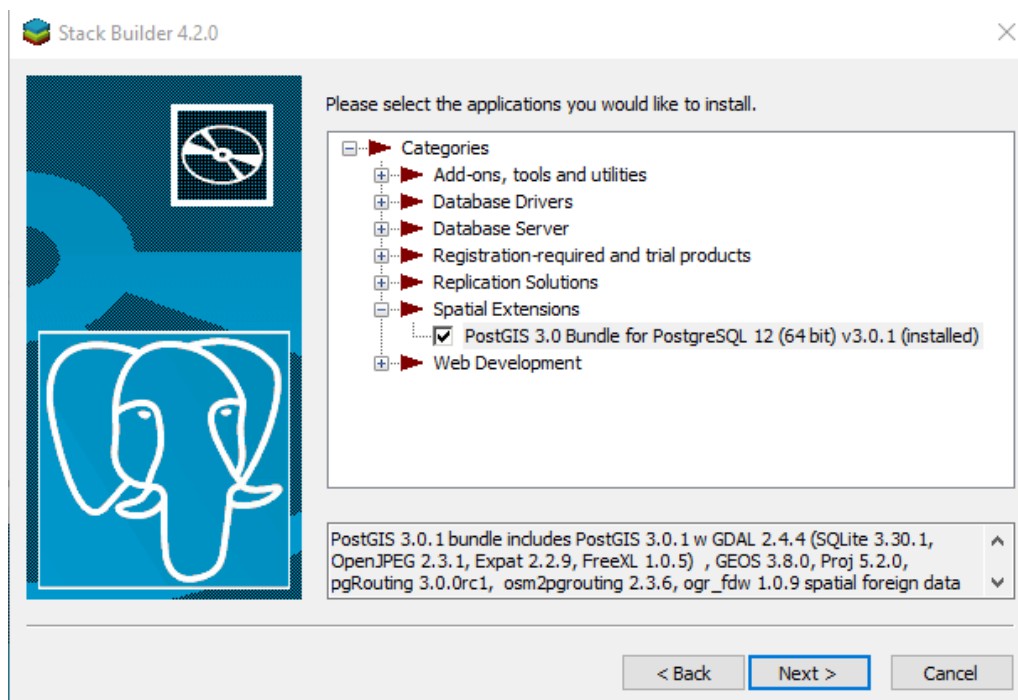


Figure 33: Stack Builder Wizard - Application Choice

On the next screen, make sure to check the PostGIS 3.0 Bundle as shown in **Figure 33**.

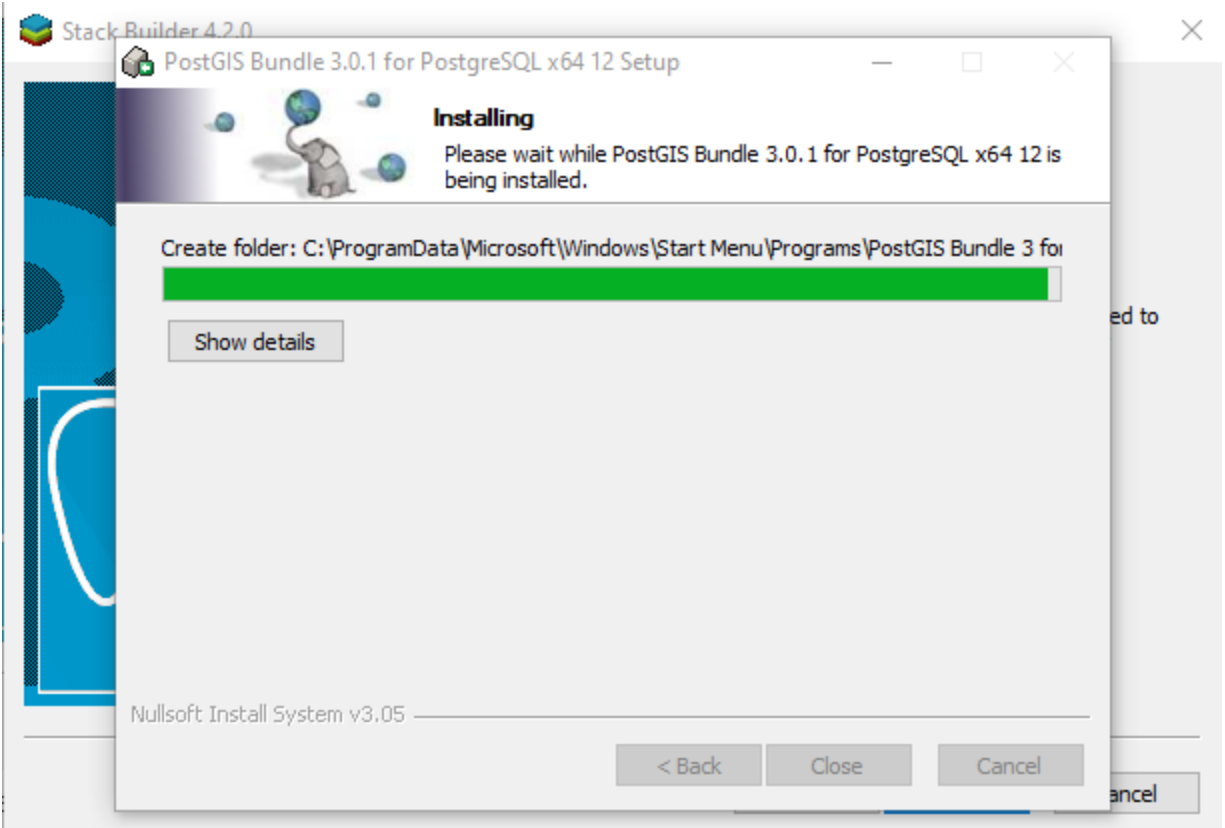


Figure 34: PostGIS Installer

Continue pressing next and agreeing to the terms of service until you reach the installation screen. The installation screen is shown in **Figure 34**.

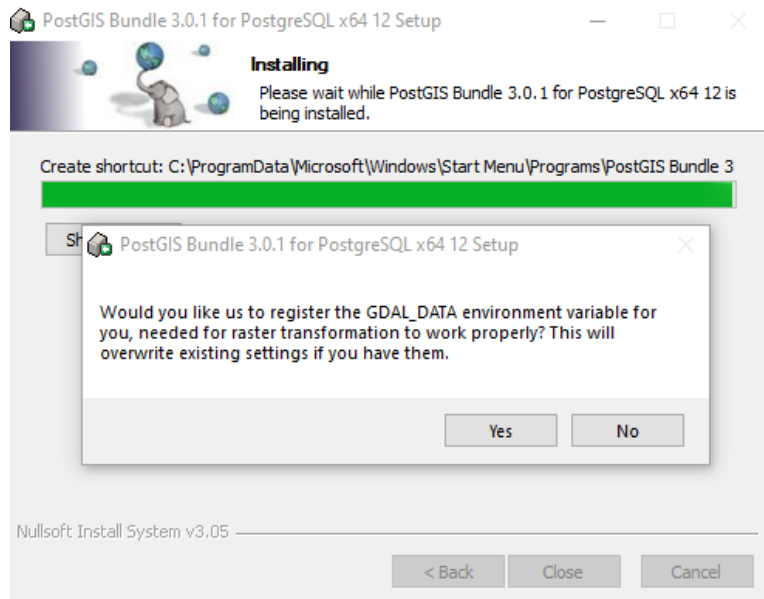


Figure 35: PostGIS Setup Box

Continue pressing yes for the boxes that appear as shown in **Figure 35**. Then the installation is complete!

## 8.2.1.2. Installing Anaconda

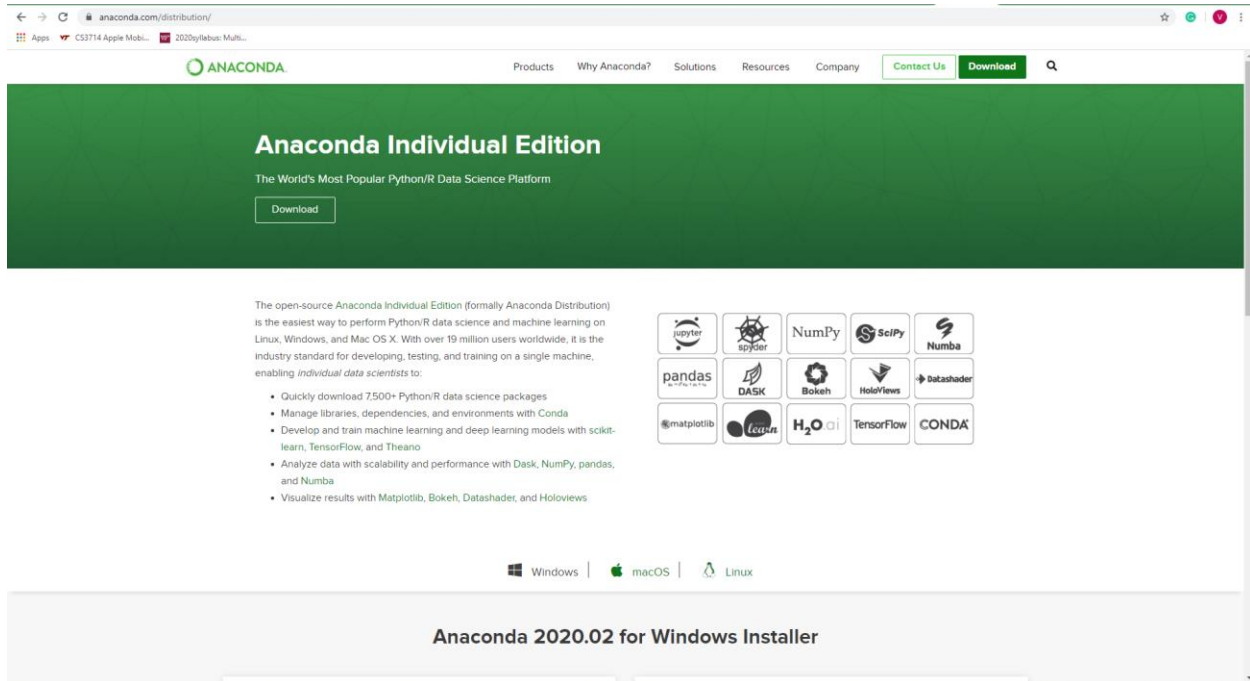


Figure 36: Anaconda Download Page

Visit the Anaconda's Website<sup>10</sup> shown in **Figure 36**. Download Anaconda Individual Edition.

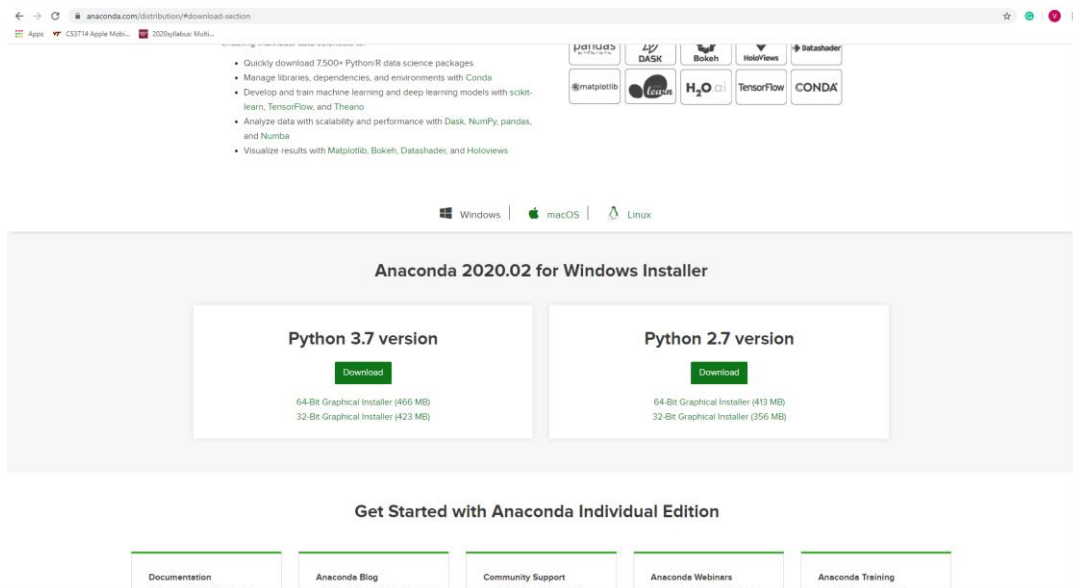


Figure 37: Anaconda Installer for Windows

Download version 3.7 shown in **Figure 37**.

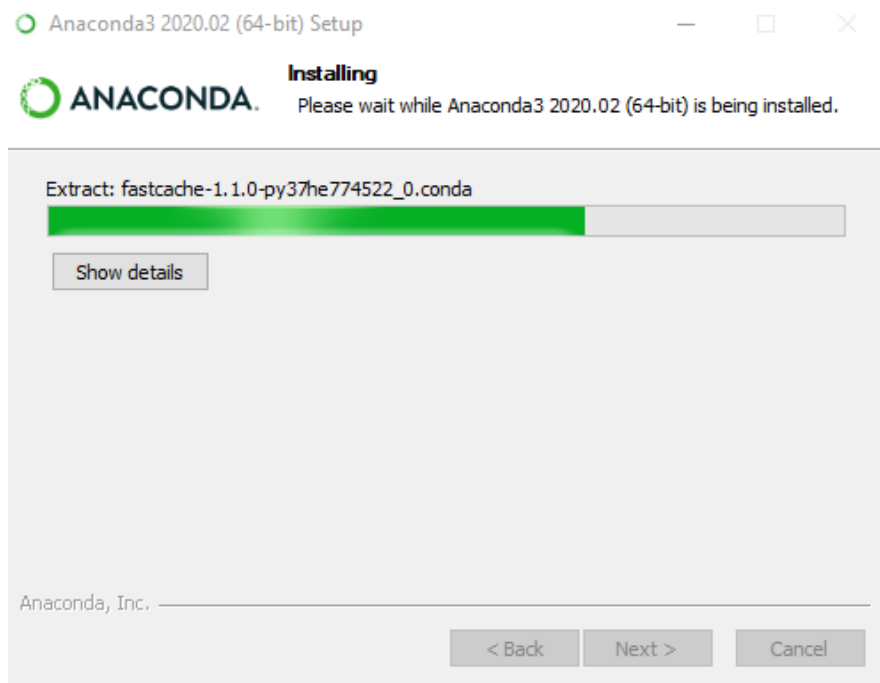


Figure 38: Anaconda Installing

You will be given an option to add Anaconda to your path. Please check it. Continue pressing next and agree to the terms of service until it begins installing. **Figure 38** shows what it looks like when Anaconda is installing. Then just continue pressing next and the installation is complete!

### Optional: Adding Anaconda to path Manually.

To do this. Open Anaconda Prompt and type “where conda” and “where python”. This is shown in **Figure 39**.

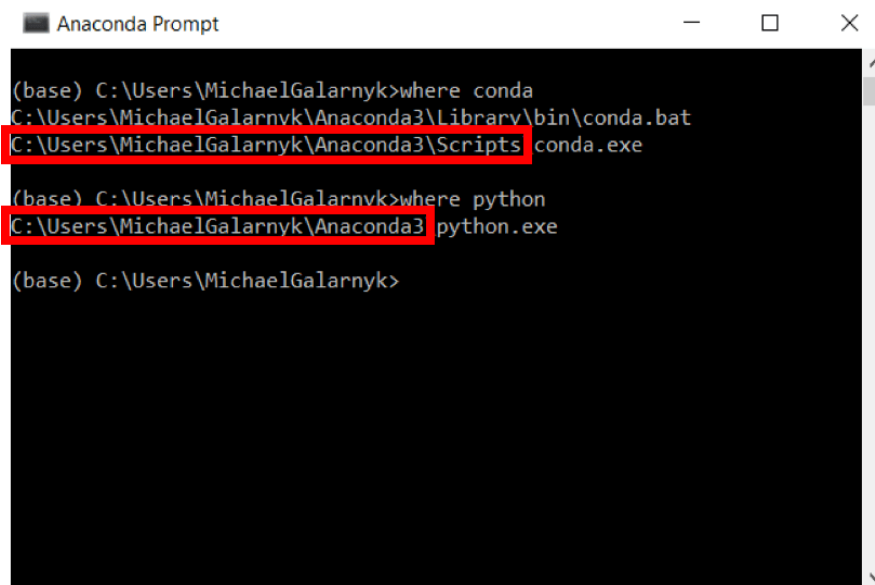


Figure 39: Finding Path to Anaconda and Python

Then you need to add the links to the path. On Windows 10, if you follow this: computer => properties => advanced system settings=> Environment Variables => System Variables> select PATH, you actually get the option to add a new row.

### 8.2.2. Getting the script

You can use git to get our repository<sup>5</sup> onto your computer:

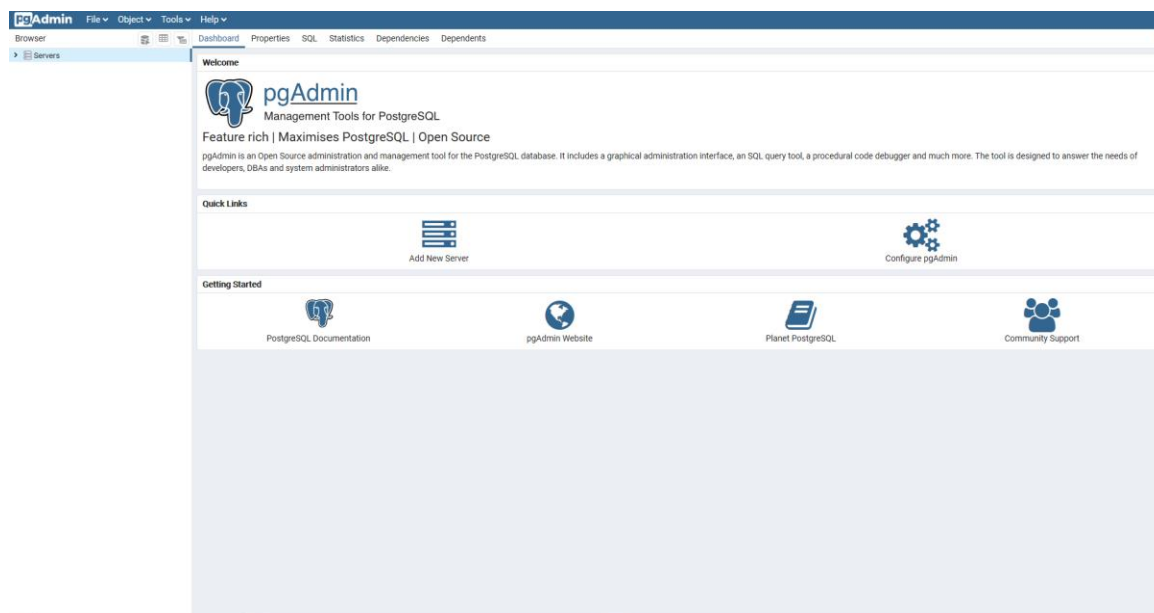
Inside a folder where you would like to keep the repository, using the terminal, enter the following command:

```
git clone https://code.vt.edu/florian/airbnb.git
```

The terminal may ask you for your login credentials for code.vt.edu. Go ahead and enter that in as well.

### 8.2.3. Setting up the Database

With the installation of PostgreSQL, you will have installed pgAdmin4 as well. By default it is located in your Programs>PostgreSQL 12 folder. You can also just search the computer for it.



*Figure 40: The pgAdmin4 Dashboard*

**Figure 40** shows what it looks like after pgAdmin4 is opened. Upon first opening it in a while, it will prompt you for a username and password to log in.

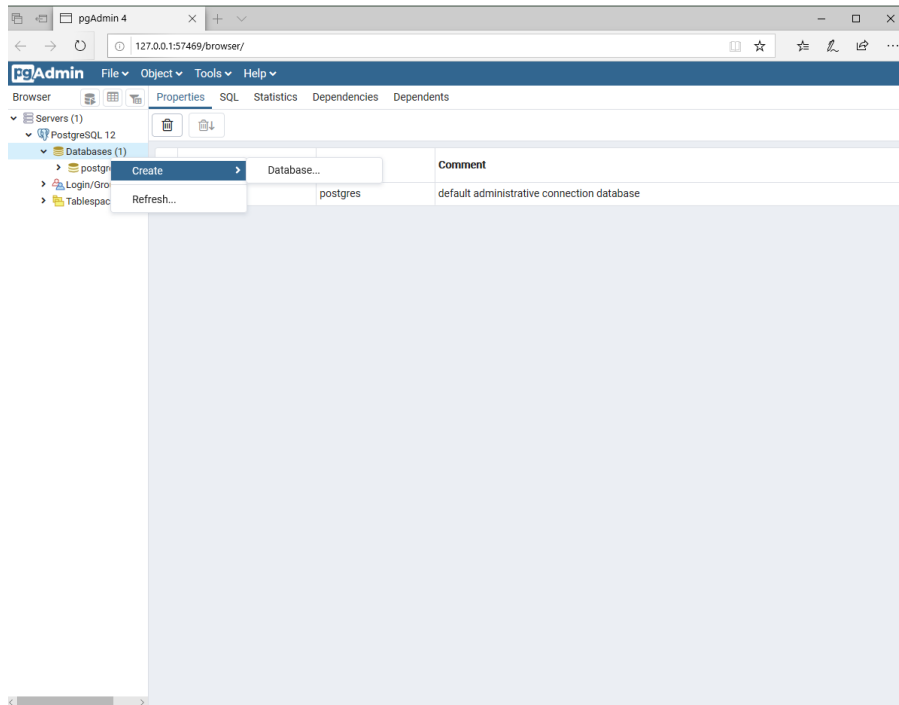


Figure 41: pgAdmin4 - Right-Clicking on 'Databases'

Keep expanding the down arrows, as shown in **Figure 41**, until you see 'Databases', right-click 'Databases'-> then click "Create" then, Database.

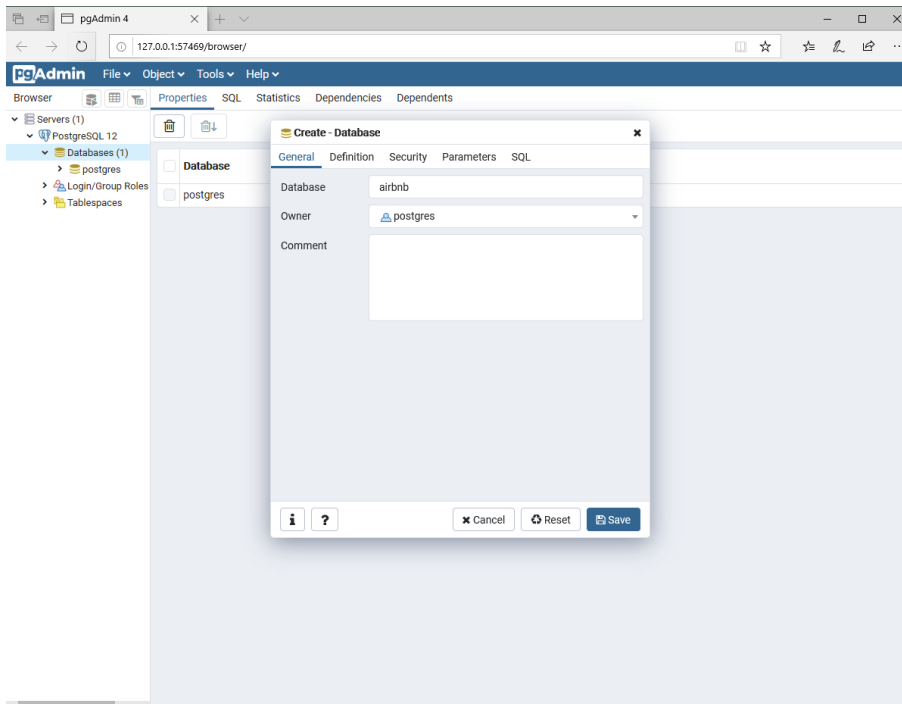


Figure 42: pgAdmin4 - Database Creation

Enter a database name. Then click Save. See **Figure 42** for what the "Create Database" screen looks like.



Next we need to apply the schema to create all the necessary tables in this database. The tables created are as follows:

- Calendar
  - Used to store all calendar related data for a listing such as availability on a day
- City
  - Used to keep track of cities during a search
- Neighborhood
  - Used to keep track of neighborhoods during a search
- Review
  - Used to store all review the reviews for a listing
- Room
  - Used to store all data gained from what's shown on the listing's page
- Schema\_version
  - Used to score the Schema's version being used
- Search\_area
  - Used to store a survey's search area coordinates
- Spatial\_ref\_sys
  - Used to store geographic reference points for better find listings.
- Survey
  - Used to store and track how many surveys were created.
- Survey\_progress\_log
  - Used to log the process of any ongoing surveys.
- Survey\_progress\_log\_bb
  - Used to log the process of how "deep" a search is currently in. This will allow for a search to resume on the last bounding box it was searching on instead of restarting if it crashes.
- Zipcode
  - Used to store zipcodes during a search by zipcodes.
- Zipcode\_US
  - Used to match zipcodes to cities in the US.

This table creation is shown in **Figure 43**.

We can do this by first adding PostgreSQL to the Path. On Windows 10, if you follow this:

```
computer => properties => advanced system settings=> Environment Variables => System Variables> select PATH
```

you get the option to add a new row. Click Edit, add the /bin and /lib folder locations and save changes.

The locations for these folders for me are:

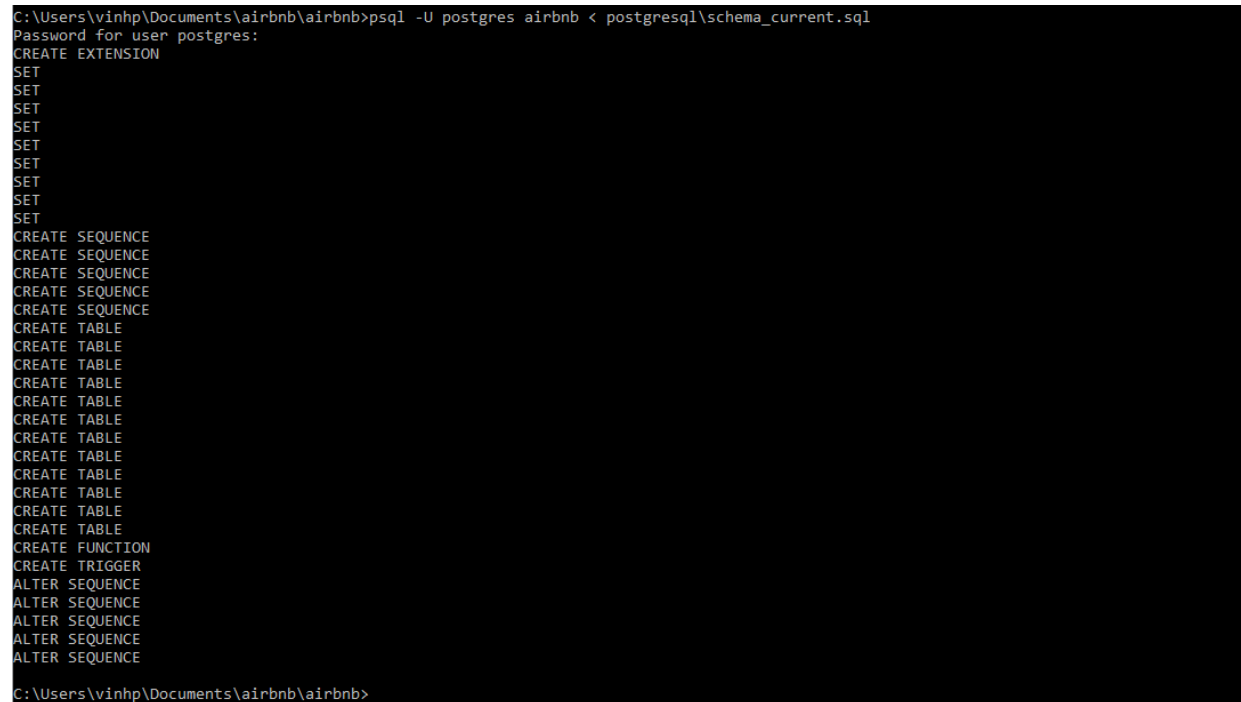
C:\Program Files\PostgreSQL\12\bin

C:\Program Files\PostgreSQL\12\lib

Then, using the Windows terminal, navigate to inside the repository on your computer and enter the command:

```
psql -U postgres airbnb < postgresql/schema_current.sql
```

Here, “postgres” is the username that you are using. You should then see what is in **Figure 43**.



```
C:\Users\vinhp\Documents\airbnb\airbnb>psql -U postgres airbnb < postgresql/schema_current.sql
Password for user postgres:
CREATE EXTENSION
SET
SET
SET
SET
SET
SET
SET
SET
SET
CREATE SEQUENCE
CREATE SEQUENCE
CREATE SEQUENCE
CREATE SEQUENCE
CREATE SEQUENCE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
ALTER SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
ALTER SEQUENCE
C:\Users\vinhp\Documents\airbnb\airbnb>
```

Figure 43: Applying the Database Schema

That’s it for the initial database set up! We will have to return to the PostgreSQL database once more before running the script, but before that, we need to connect the script to the database using the CONFIG file.

#### 8.2.4. Configuration

In the repository, you will see a file named “example.config”. Feel free to change and create a copy of this file and name it whatever your computer’s user’s name is. For example, if your computer’s user’s name is “johns”, name your CONFIG file, “johns.config” In this file you will have to change a few things.

db_host should be set equal to local_host
db_port should be 5432
db_name should be the name of the database you created. Mine is called "airbnb".
db_user should be the username you created. Mine is "postgres"
db_password should be the password for that user.

Nothing else is required to be changed but one may wish to fiddle with the request\_sleep number for a longer/shorter break between requests depending on how frequently Airbnb is blocking your IP. Setting request\_sleep to 10 is pretty safe to not get your IP blocked too often. Airbnb doesn't like people slamming their site with requests, so they will block you if you do not wait in between requests.

### 8.2.5. Installing lxml and psycpg2

You also need lxml. To download this, for Windows, you can use pip install lxml --force -U  
The download for lxml is shown in **Figure 44**.

```
C:\Users\vinhp\Documents\airbnb\airbnb>pip install lxml --force -U
Collecting lxml
  Downloading lxml-4.5.0-cp37-cp37m-win_amd64.whl (3.7 MB)
    |████████████████████████████████████████| 3.7 MB 3.3 MB/s
Installing collected packages: lxml
  Attempting uninstall: lxml
    Found existing installation: lxml 4.5.0
    Uninstalling lxml-4.5.0:
      Successfully uninstalled lxml-4.5.0
Successfully installed lxml-4.5.0
C:\Users\vinhp\Documents\airbnb\airbnb>
```

Figure 44: Downloading lxml

Done for lxml! Easy!

Next we need to install psycpg2.

For Windows, use pip install psycpg2-binary

The download for psycpg2 is shown in **Figure 45**.

```

C:\Users\vinhp\Documents\airbnb\airbnb>pip install psycopg2-binary
Collecting psycopg2-binary
  Downloading psycopg2_binary-2.8.5-cp37-cp37m-win_amd64.whl (1.1 MB)
    |-----| 1.1 MB 3.2 MB/s
Installing collected packages: psycopg2-binary
Successfully installed psycopg2-binary-2.8.5

C:\Users\vinhp\Documents\airbnb\airbnb>

```

Figure 45: Downloading psycopg2

### 8.2.6. Running the Script

Congratulations! You are almost done!

Inside your repository folder again, you can check if you can connect to the database with:  
python airbnb.py -dbp

The result of this command is shown in **Figure 46**.

```

C:\Users\vinhp\Documents\airbnb\airbnb>python airbnb.py -dbp
Connection test succeeded: airbnb@localhost

C:\Users\vinhp\Documents\airbnb\airbnb>

```

Figure 46: Testing Database Connection

Next we need to add a city name for our search; we want to do this using:

python airbnb.py -asa "City Name"

The results of this command is shown in **Figure 47**.

```

C:\Users\vinhp\Documents\airbnb\airbnb>python airbnb.py -asa "test"
Search area test added: search_area_id = 1
Before searching, update the row to add a bounding box, using SQL.
I use coordinates from http://www.mapdevelopers.com/geocode_bounding_box.php.
The update statement to use is:

    UPDATE search_area
    SET bb_n_lat = ?, bb_s_lat = ?, bb_e_lng = ?, bb_w_lng = ?
    WHERE search_area_id = 1

This program does not provide a way to do this update automatically.

C:\Users\vinhp\Documents\airbnb\airbnb>

```

Figure 47: Update search\_area Message From: python airbnb.py -asa

This adds a city to the search\_area table.

Then, we enter back into our database using psql -U postgres airbnb  
**Figure 48** shows what it looks like to enter the database.

```
C:\Users\vinhp\Documents\airbnb\airbnb>psql -U postgres airbnb
Password for user postgres:
psql (12.2)
WARNING: Console code page (437) differs from Windows code page (1252)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

airbnb=#
```

Figure 48: Connecting to the Database with psql

Then we want to use the following query:

```
UPDATE search_area
```

```
SET bb_n_lat = ?, bb_s_lat = ?, bb_e_lng = ?, bb_w_lng = ?
```

```
WHERE search_area_id = ?
```

As you can see in **Figure 47**, our search\_area\_id is 1 and we can get the coordinates using Klokantech's bounding box tool.<sup>8</sup>

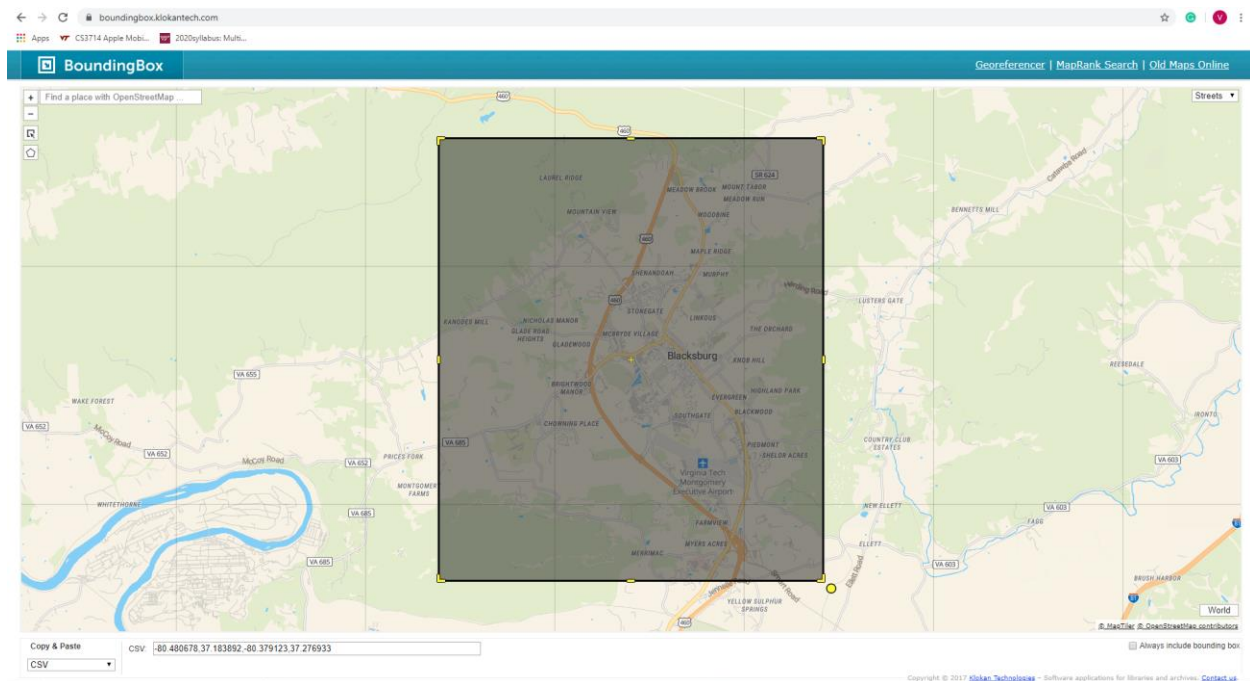


Figure 49: Finding the Bounding Box for Blacksburg

Using the Klokantech's bounding box tool, shown in **Figure 49**, notice the bottom left corner has a "copy and paste" section.

Use CSV formatting: Order goes: West, South, East, then North

Thus, for a survey of Blacksburg, set an area such that our query looks like:

```
UPDATE search_area
```

```
SET bb_w_lng = -80.480678, bb_s_lat = 37.183892, bb_e_lng = -80.379123, bb_n_lat =  
37.276933
```

```
WHERE search_area_id = 1;
```

```
airbnb=# UPDATE search_area  
airbnb=# SET bb_w_lng = -80.480678, bb_s_lat = 37.183892, bb_e_lng = -80.379123, bb_n_lat = 37.276933  
airbnb=# WHERE search_area_id = 1;  
UPDATE 1  
airbnb=#
```

*Figure 50: Updating the search\_area*

If you see UPDATE 1, then you did it correctly. **Figure 50** shows a successful update.

Notice how you are able to change the ordering of setting any compass direction. The order we set the coordinates doesn't have to be north, south, east, then west. We set them in the order of west, south, east, then north because that is how the CSV formatting displays the coordinates. Also, note that the query must end in a semicolon on the last line.

You can exit the database by typing exit or \q

Then we add a survey description for that city after exiting the database:

```
python airbnb.py -asv "City Name"
```

```
C:\Users\vinhp\Documents\airbnb\airbnb> python airbnb.py -asv "test"  
Survey added:  
  
    survey_id=1  
    survey_date=2020-04-18  
    survey_description=test (2020-04-18)  
    search_area_id=1  
C:\Users\vinhp\Documents\airbnb\airbnb>
```

*Figure 51: Message Displayed From: python airbnb.py -asv*

This makes an entry in the survey table, and should give you a survey\_id value as shown in **Figure 51**

Then the last thing to do is... run the script!

Using: `python airbnb.py -sb 1`

Here 1 is the `survey_id` assigned earlier. (You can see this in **Figure 51**.)

```
C:\Users\vinhp\Documents\airbnb\airbnb>python airbnb.py -sb 1
INFO      =====
INFO      Survey 1, for test
INFO      Searching by bounding box, max_zoom=12
INFO      -----
INFO      Rectangle calculated: [37.276933, -80.379123, 37.183892, -80.480678]
INFO      Searching rectangle: zoom factor = 0, node = []
```

*Figure 52: Message Displayed From: `python airbnb.py -sb`*

**Figure 52** shows what it looks like when a survey is just started. The script will keep running until it is finished! It will say “Finished Survey” once done. And it will notify you after every page of listings it comes across.

### 8.2.7. Exporting and Viewing the Data

To export the data to a CSV file:

Access the database: `psql -U postgres airbnb`

Then enter the commands.

```
\copy (SELECT * FROM room where survey_id = [surveynumber]) to 'room_data.csv' csv header
```

```
\copy (SELECT * FROM calendar where survey_id = [surveynumber]) to 'calendar_data.csv' csv header
```

```
\copy (SELECT * FROM reviews where survey_id = [surveynumber]) to 'review_data.csv' csv header
```

`Room_data.csv`, `calendar_data.csv`, and `review_data.csv` will be the file names for its respective table. You can change these to whatever you want but keep the “.csv”

This will export the data into a CSV file in the repository.

To quit out of `psql` type `\q`

Then you can exit out of the container by typing `exit`

To view the data using pgAdmin 4:

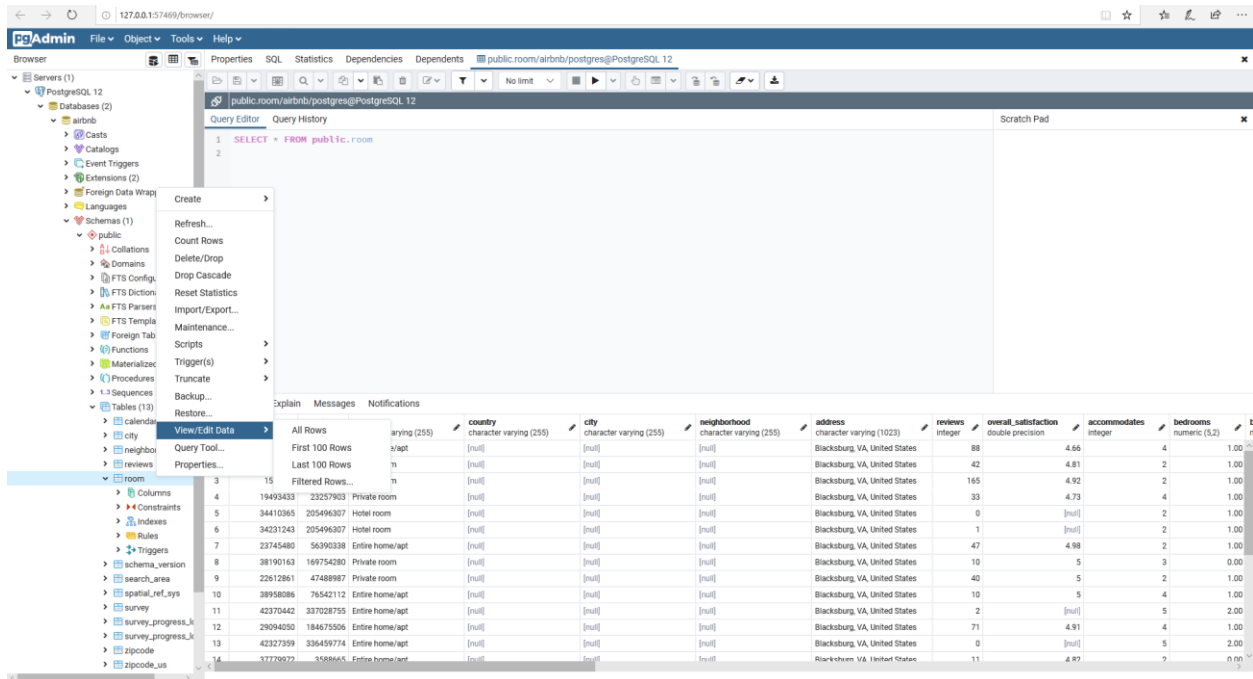


Figure 53: Viewing Data Through pgAdmin4

Figure 53 shows how to view data. Log into pgAdmin4 and in the database, under Schemas, public, tables, right-click the table you want to view and click View/Edit data. The tables of interest are the 'room', 'calendar', and 'reviews' tables. These tables are where scraped data from Airbnb are stored.

## 8.3. Methodology

### 8.3.1. User Goals

#### Persona:

Users: 1) Dr. Zach 2) Future students working on this research.

Dr. Zach: A professor who has a limited CS background.

#### Goals:

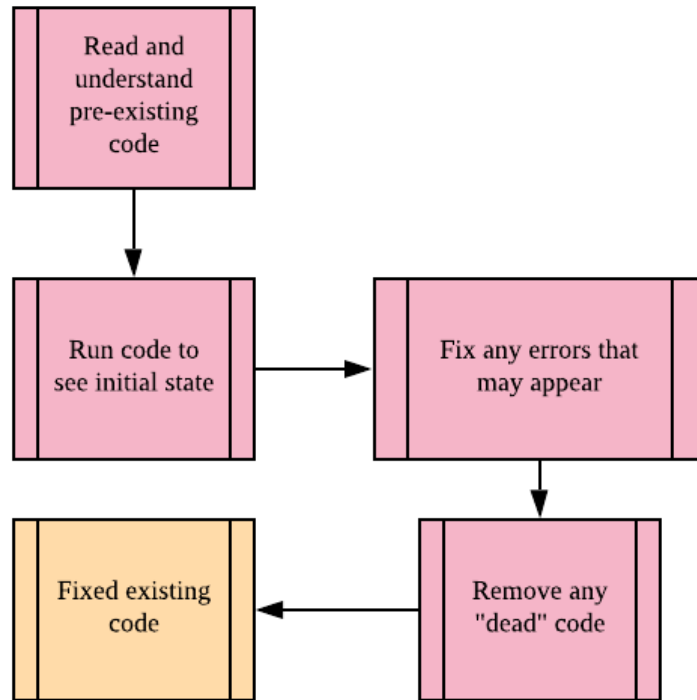
1. Collect data from airbnb website
2. Visualize the retrieved data
3. Build up a Website and display the generated graph.



### 8.3.2.Tasks and Subtasks

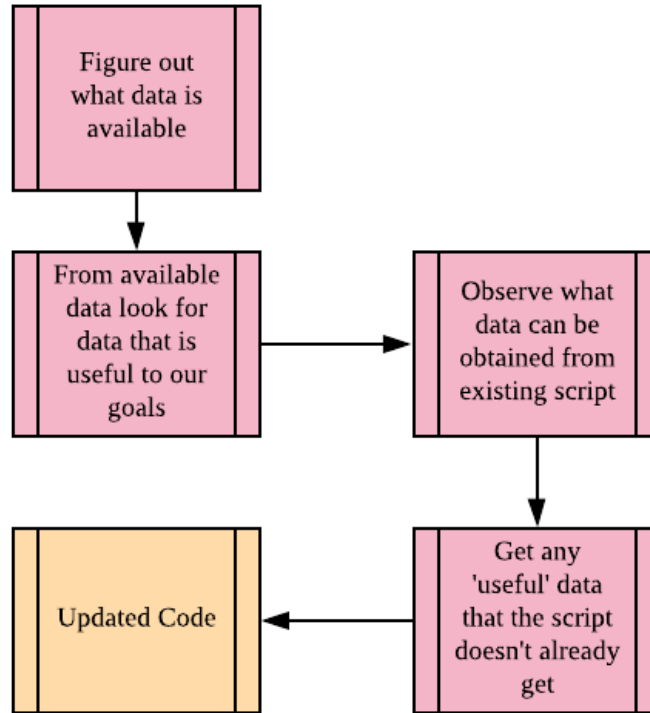
#### 1. Data Collection:

- a. Fix existing code. The process is shown below in **Figure 54**. It depicts the subtasks involved in fixing the existing code.



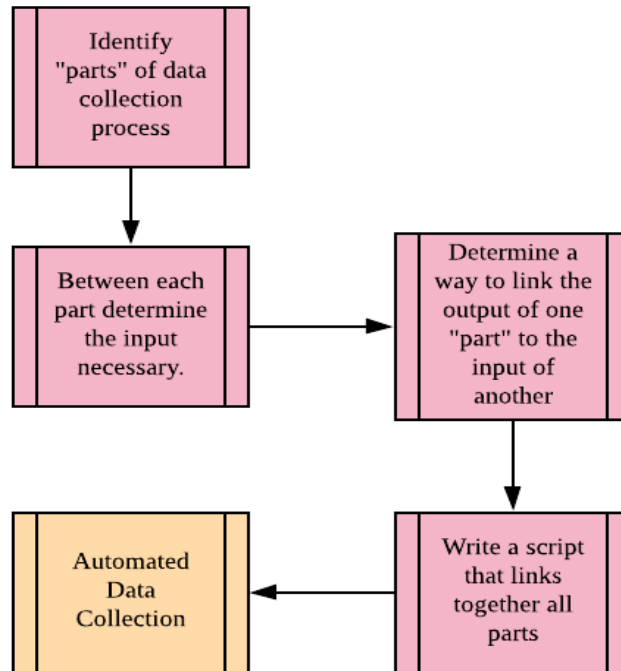
*Figure 54: Process of Fixing Existing Code*

- b. Update the code from Tom Slee's previous code. The process is shown below in **Figure 55**. It depicts the subtasks involved in updating the code.



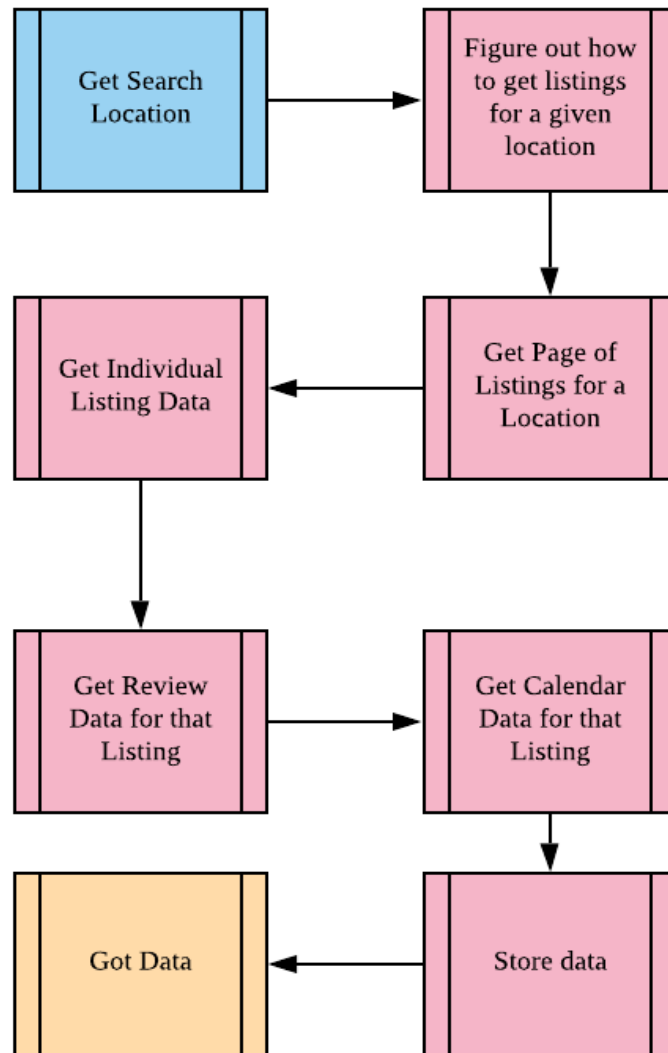
*Figure 55: Process of Updating the Code*

- c. Make the code automatically scrape the new data and store it in the database, to collect enough data to analyze in the future. The process is shown below in **Figure 56**. It depicts the subtasks involved in automating the data collecting code.



*Figure 56: Process of Making the Code Automatically Scrape*

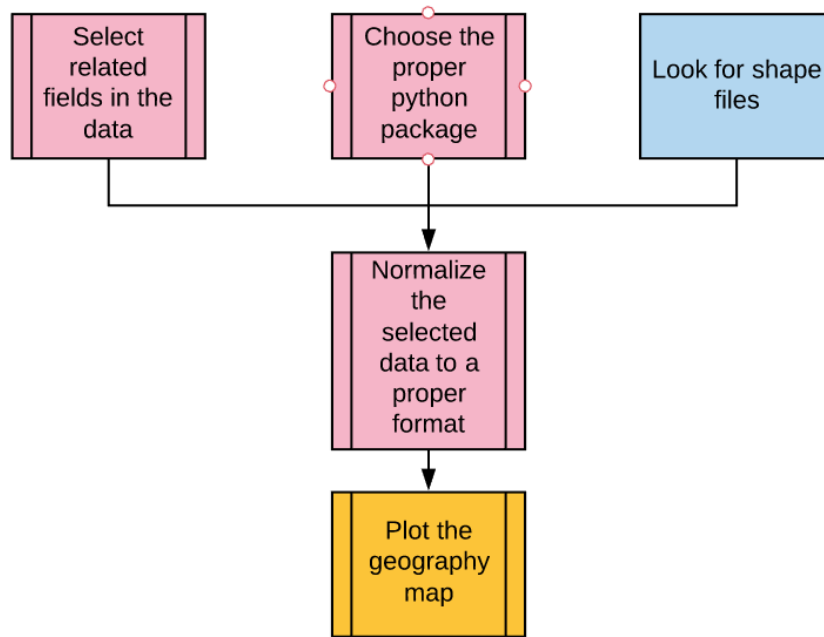
- d. Get the data. The subtasks involved in getting the data is visualized in **Figure 57** below.



*Figure 57: Process of Retrieving the Data*

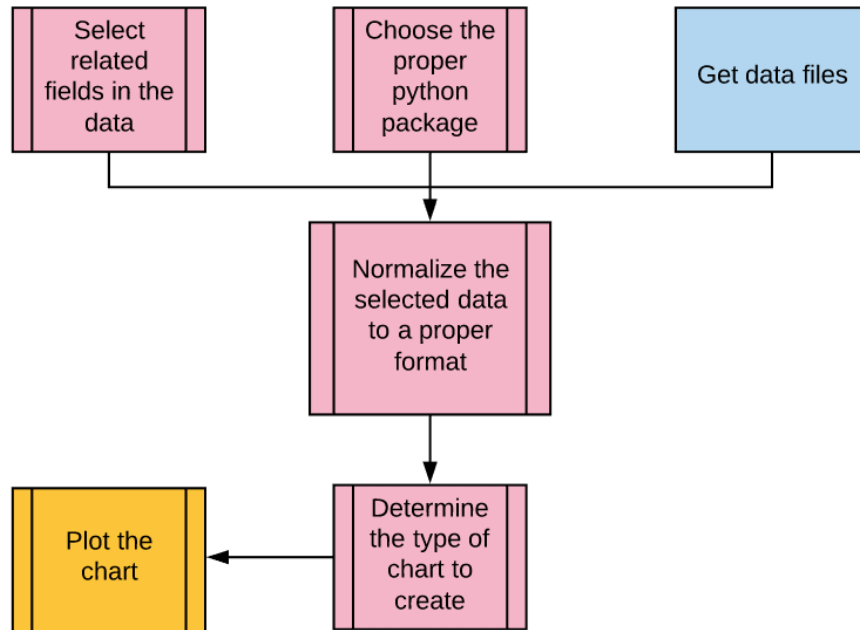
## 2. Visualize the retrieved data.

- a. Look for trends of the data over time, such as the changes of different types of houses' distribution, prices, and the total number of houses over weeks, seasons, and years. **Figure 58** which is shown below introduce the process of plotting a geo map.



*Figure 58: Process of Plotting Geo Maps*

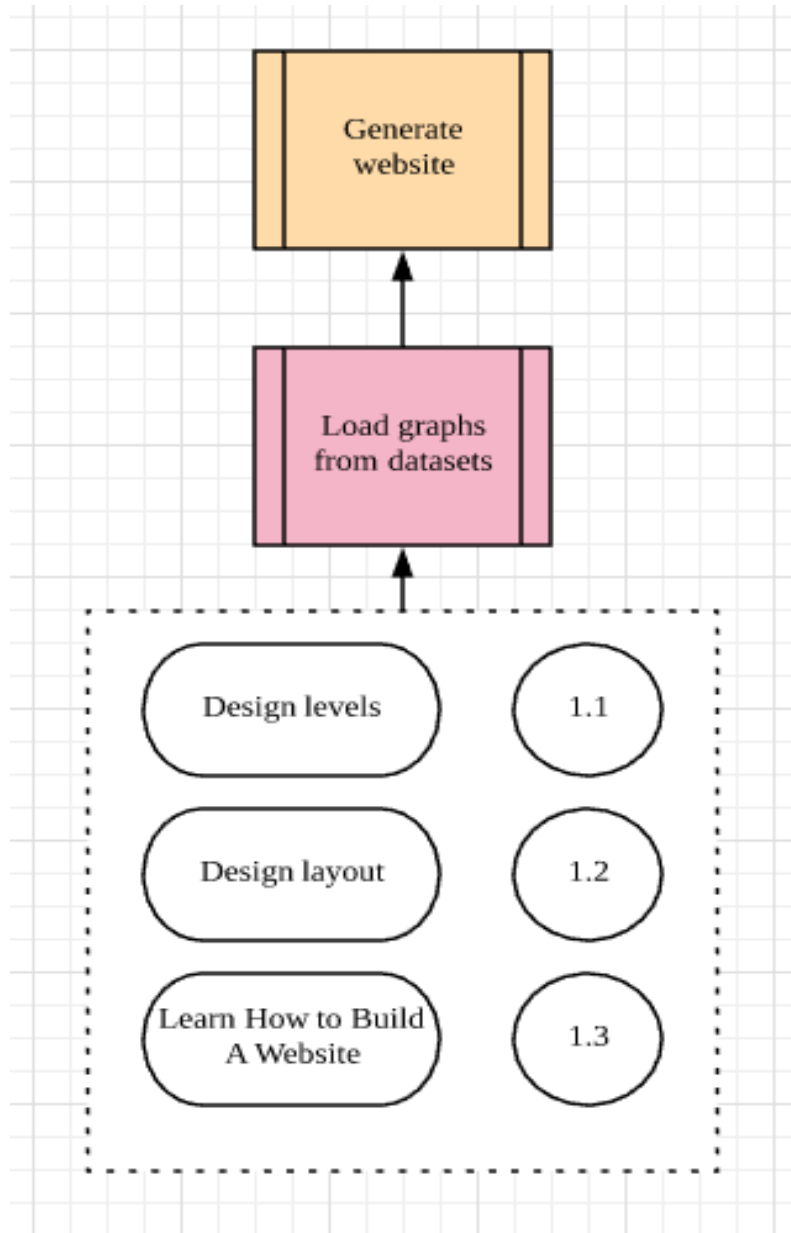
- b. Investigate how different fields of data are correlated with each other to give users inferences. **Figure 59** which is shown below represent the process of plotting chart, maps and tables in our project.



*Figure 59: Process of Plotting Charts, Maps, and Tables*

### 3. Website:

- a. Display visualizations of the data in an aesthetic way on a website where it is easy to navigate different graphs, charts, and maps.



*Figure 60: Display the Visualizations on the Website*

In **Figure 60**, to build the website, we will figure out the levels of the web pages. For each page, we will draw the layout. And then we will build the website based on our design. When the website code is written, we will fill the blank with our graphs. In the end, we will generate the whole website.

To support the goal of examining tweets reporting “outage” during a hurricane, the system needs to support the tasks of:

1) Key phrase matching and sentiment analysis. This task is dependent on the task of 2) doing sentiment analysis on tweets, which is dependent on the system supporting the task of 3) extracting NLP/text-based measures and so on and so forth.

So, the structure of the graph indicates how tasks are dependent on one another. As a result, can derive a sequence of tasks required to accomplish the goal (as seen above).

What you should turn in is: a) a set of figures and b) a short description of each task and subtask.

### 8.3.3. Implementation-based Services

#### 1. Data Collection:

##### Fix Old code

- a. Read Code: For fixing the old code first we need to read the code. To do this nothing is required other than GitHub to try to read and analyze how it works.
- b. Run Code: Input would be a location run the script. To run the code, you can do this by downloading the existing repository. Then you need to install the latest version of PostgreSQL and the latest version of Anaconda. After installing everything, you can configure the script to connect to the database. After this, run the existing schema to create all the necessary tables in the database. You can then run the script by first creating a “survey”. You can do this by entering: `python airbnb.py -asa “city”`. Then you can change the location coordinates that you want to search under in a `search_table` in PostgreSQL. After this you can simply run the script using `python airbnb.py -sb 1`. The script then runs and outputs any listing data to the table. For more detailed information on running the code, see section 8.2.
- c. Error Fix: Occasionally, the script would crash as it had little error handling. To fix this, after running the code, we can analyze the error and manually search through the line number that the error is being thrown at. Since we understand the code from reading the existing code/documentation, fixing whatever error that is thrown should be not difficult.



## Update Code

1. **Data Available:** Find out what data can be scraped with the existing code. We take the output of the existing code, after running it, and we then use pgAdmin to view the PostgreSQL database. We can then see what columns are filled out and verify that the data being stored is correct by comparing it with Airbnb's results. If we have listing IDs, that is how we will identify the data. If not we need to get that working before we can figure out if the data is accurate.
  2. **Useful Data:** From the data available, we will continue using pgAdmin or to view our database to see what data we can use that would be good for our goals.
  3. **Data Possible:** We can find what data is possible by using network requests. We find what URL Airbnb is using to retrieve its listing data using a tool suite such as the Chrome Developer Tools. From here, we see that the data is in JSON objects which can then be parsed and read by visiting the URL. From here, we know what data can be possibly obtained.
  4. **Useful New Data:** Using the data from Data Possible, we then can manually filter out the data by observing and working with our client to figure out what data he wants to view.
2. **Automation**
    - a. **Identify Parts:** For each part we just need to analyze the function of the script that we have already done in Fix Old Code.
    - b. **Determine Outputs and Inputs:** From identification we can see how there are different outputs and inputs for different parts of the project. This is also intuitive from Fix Old Code.
    - c. **Determine Link:** The link can be determined by finding out where we need to bridge a gap. In our code, this happens to be the area where we need a user to specify coordinates of an area by going to a website to get bounding box coordinates.
    - d. **Linking:** We can link by automating this process of going to the website by scraping that website itself after searching for a given location. This only has to be done once per script and can ideally be done using the same scraping procedure of our project. If need be, we can also create a bash script to automate the script as well, and the user must research all the bounding box coordinates someone wants to try in advance.
  3. **Getting Data**
    - a. **Get page of Listings:** We can get Listing data based off of location using a recursive bounding box approach. The input is the coordinates for a bounding box that the user inputs. This box is then recursively searched to render more

and more listings within a certain area until a box reaches no listings. Once this happens, it does not continue breaking down that box any further.

- b. Get Listing Data: From 'get page of listings', we can then use the listing IDs that are returned to then use a network request to get the data from the source page of each listing.
- c. Get Review Data: Similarly to above, we know the listing ID from 'Get page of listings'. Thus, we can also get the reviews from the review page for that listing. We just parse through the JSON objects of each review.
- d. Get Calendar Data: Similarly to above, we know the listing ID from 'Get page of listings'. Thus, we can also get the calendar availability from the calendar page for that listing. We just parse through the JSON objects of each calendar day.
- e. Save Data: We then save all this data into our PostgreSQL database. We save this using psycopg2. We then call update and insert queries to insert into the database.

## Visualizations

1. Find the trends of the data over time

- a. Input files: CSV files with Airbnb data which are retrieved from websites,  
shapefiles from all counties in Virginia and Australia

Output files: geography maps for the distribution of Airbnb houses

- b. Environments: Jupyter Notebook (Python3.7)
- c. Libraries: Pandas, Descartes, Geopandas.

2. Investigate how different fields of data are correlated with each other to give users inferences.

Input files: CSV files with Airbnb data which are retrieved from websites

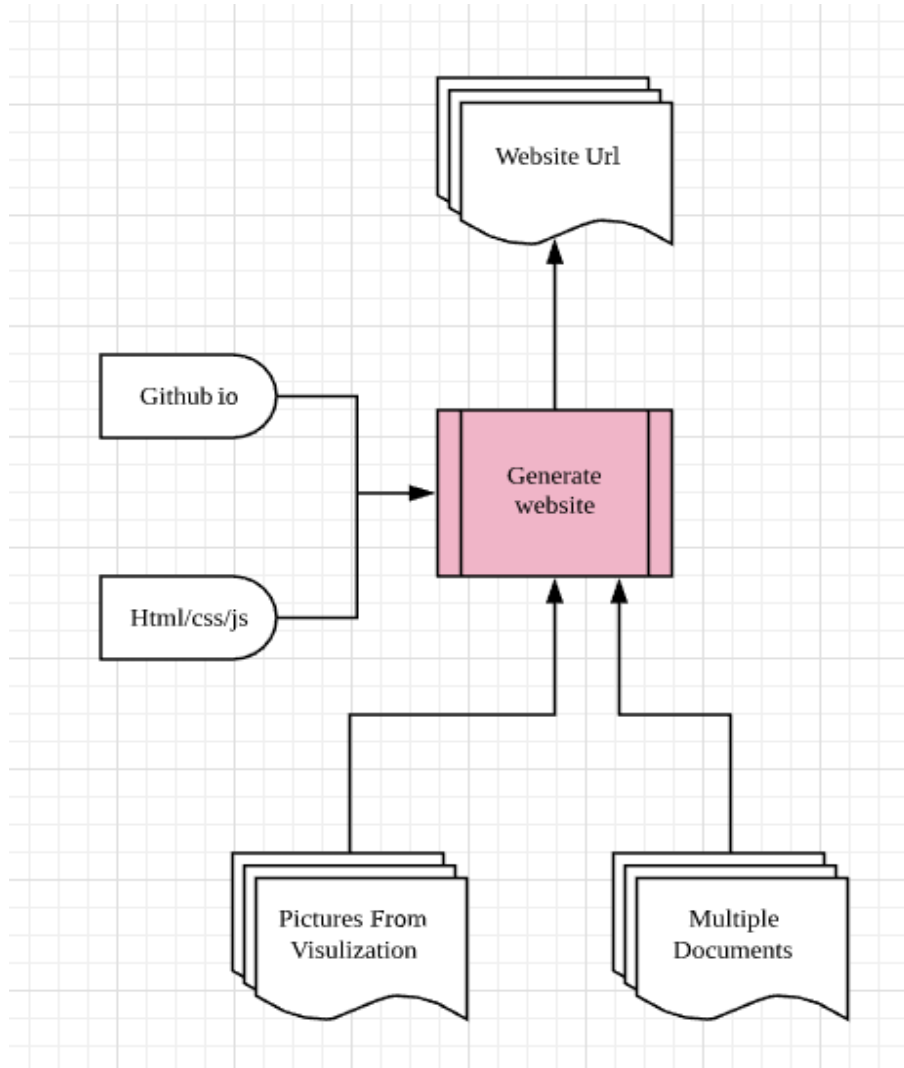
Output files: a variety of charts, such as pie charts, bar charts, scatter plot, and WordCloud maps

Environments: Jupyter Notebook (Python3.7)

Libraries: Pandas, Pycharts, Matplotlib, NLTK,

Website:

1. Display visualizations of the data in an aesthetic way on a website where it is easy to navigate different graphs, charts, and maps.



*Figure 61: Design of the Website*

Input files: pictures from visualization and multiple documents

Output files: the website URL for others to access

Environments: github io which is the one to host the website; the HTML/CSS/JavaScript are the languages to use.

### 8.3.4.Workflows

Goals:

#### 1. Data Collection:

##### 1) Fix Existing Code

- Read Code
- Run Code
- Error Fix

##### 2) Update Code

- Data Available
- Useful Data
- Data Possible
- Useful Possible Data

##### 3) Automation

- Identify Parts
- Determine Output and Input
- Determine Link
- Link

##### 4) Getting Data

- Get Page of Listings
- Get Listing Data
- Get Review Data
- Get Calendar Data
- Store Data

#### 2. Visualize the Retrieved Data

##### 1) Find the trends of the data over time

- Import shapefiles of Virginia and Australia
- Choose the proper Python packages
- Normalize the selected data to a proper format
- Plot the geography map and save it

2) Investigate how different fields of data are correlated with each other to give users inferences

- Import CSV files with Airbnb data
- Select related fields in the data
- Choose the proper Python packages
- Normalize the selected data to a proper format
- Determine the types of chart to make
- Plot the chart and save it

3. Build up Website

- Learn how to build to build website
- Design layout
- Design levels
- Import graphs
- Generate URL

## 8.4.Files and Folders

In this section we will briefly explain the purpose of each file in our repository<sup>5</sup>. The explanations will be general and will explain how a file contributes to the project as a whole. For a more deep, technical description, see the comments inside each file. The script is made up of many smaller scripts. Whenever we refer to “*the script*”, know we are talking about the script as a whole rather than any individual script.

### 8.4.1.Root Directory

The root directory of our repository contains the vast majority of the files in our project. They are listed below:

#### **airbnb.config**

This is a CONFIG file that allows the user to customize the script to work with a database of their choosing. It also allows for more customization on how the script runs such as giving the user the ability to change wait time between requests and how many times the script can zoom into a bounding box.

### **airbnb.py**

This Python script contains the main function. It is the script you will be using for preparing to set up a survey.

### **airbnb\_config.py**

This Python file parses the 'airbnb.config' file. It puts each field of the CONFIG file into a structure which is later called and used in various other portions throughout the script. Whenever the CONFIG file's fields are needed, the structure created using this file is accessed.

### **airbnb\_listing.py**

This Python file is used as a structure to store all the data we collect for any individual Airbnb listing. The file is also used when uploading the data of a listing to the database.

### **airbnb\_s3\_upload.py**

This Python file is used to more systematically export data from the database. It contains some functions to export the data collected in several formats such as CSV and HTML.

### **airbnb\_summaries.py**

This Python file is used to load data to get a quick summary on all the listings collected so far. This helps in displaying survey progress.

### **airbnb\_survey.py**

This Python file contains most of the logic that goes into the process of a survey. The file conducts requests, data storage, and mapping data to surveys. It also contains all the recursive logic for bounding box searches. If the script is no longer collecting some data, check this file to make sure the JSON objects the script is expecting match with the JSON objects Airbnb is sending.

### **airbnb\_ws.py**

This python file handles the requests to Airbnb. It makes repeated requests just in case one request fails and also handles the use of any proxies that were specified in 'airbnb.config'.

### **amenities.json**

This is a simple JSON file that stores a list of JSON objects. Each JSON object is an amenity Airbnb allows a listing to have. This file is used when trying to map the amenity number that we get from Airbnb to its name.

### **Dockerfile**

This file is used in building a Docker image of the project. Use this file any time you want to create a new Docker image such as wanting to use an updated version of any of the Python files in the project.

### **export\_spreadsheet.py**

This Python file contains some useful functions to export a CSV file of the data with various filters.

### **README.md**

This is just a file containing more specific instructions on how to run the script. It also describes in more detail about the history of the script.

### **requirements.txt**

This is a Text Document that is used in the creation of a new docker image. It specifies all the required technologies that are needed by the script.

### **reverse\_geocode.py**

This Python file takes a latitude and longitude and gets a more identifiable name/address. It uses the Google Maps API. If you do not have a key for the API, this file will not work. The file is not totally necessary but can make searches more accurate.

### **schema\_update.py**

This Python file enables for easier updating of existing tables and columns in the database via its schema.

### **survey\_report.py**

This Python file creates a log of every survey and when a survey stops (be it by it finishing or by you manually stopping it), a text file is created containing a log of the progress for the survey. It also contains all the output of the survey from the time it began.

## **8.4.2.The “docker” Folder**

In the folder titled, “docker”, it contains the following files:

### **docker-compose.yml**

This file is used when starting a docker container. In this file, you can specify what Docker image you want to use from Docker Hub. Thus, you can use a more updated version of the script by building a more updated Docker image, uploading that image to Docker Hub, then change this file to use the uploaded image.

### **configs**

This is a shared folder so that you can edit the CONFIG file inside the Docker container.



### **scripts**

This is folder containing “init\_db.sh”. It is a shell script that is used for apply the schema to the database.

### 8.4.3.The “data” Folder

This folder is purely used so that the script has a well-defined place to export data in CSV format. Purely organizational.

### 8.4.4.The “postgresql” Folder

This folder contains all the database related files as listed below:

#### **functions.sql**

This file contains various database functions used in manipulating the database. The functions mainly revolve around keeping track of surveys in the database.

#### **schema\_current.sql**

This is the schema that is applied to the database when initially setting up the script. It essentially creates all the tables and columns that the script expects. For more information on what tables it creates, see section 8.2.3.

## 9. Lessons Learned

### Timeline/Schedule

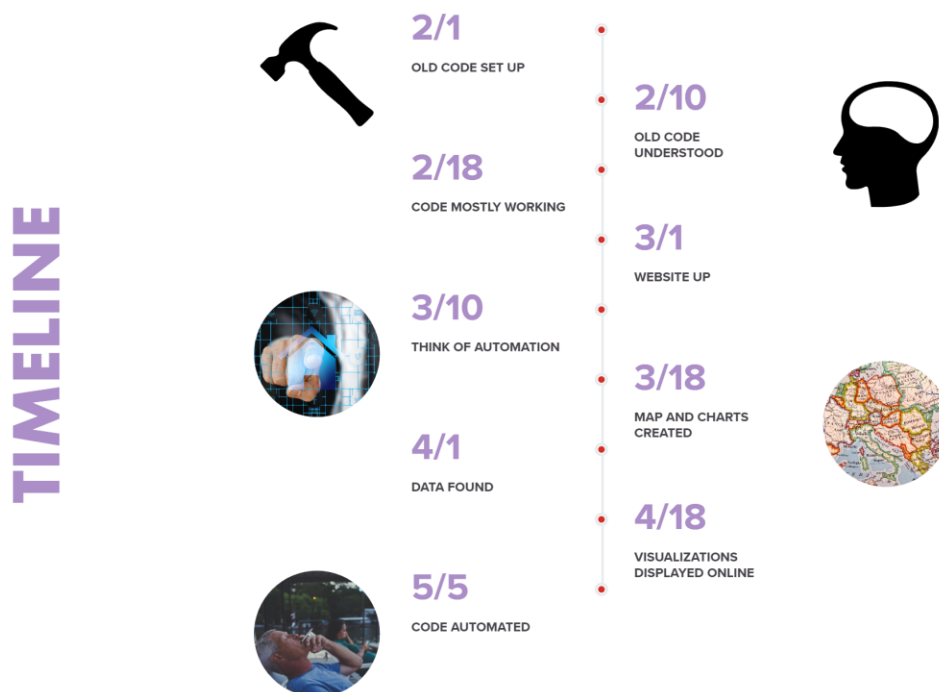


Figure 62: Timeline

Figure 62 shows a visual summary of our timeline.

- February 1st
  - Old code completely set up.
  - Brainstorm for the design of visualizations.
- February 10th
  - Old code abstractly understood.
  - Direction to where bugs and possible fixes can be found.
  - Visualization design decided.
- February 18th
  - Strong knowledge on what needs to be changed with the code.

- The code is mostly working to achieve "descriptive statistics" deliverables.
  - Revert to Plan B if code can't be fixed: Do visualizations of what we can extract using the current code.
- March 1st
  - Code is able to completely achieve "descriptive statistics" for a single survey.
  - Visualization designs tweaked around available data.
- March 10th
  - Brainstorm ways for automating city input.
- March 18th
  - Create a map with data overlaid.
  - Data collected for all Austrian cities.
  - Increase accuracy for data collected in each city.
- April 1st
  - Create bar charts based on the frequency of data categories.
  - Data collected for all Virginia cities.
  - Increase accuracy for data collected in each city.
- April 10th
  - Brainstorm ways for script scheduling.
- April 18th
  - Visualizations using collected data displayed online.
- May 5th
  - Top 50 words found in review per city.
  - Word bank of top 50 words found per city
  - Visualize and automate the code to run and update regularly.

Problems	Solutions
<p>When we try to plot the Airbnb houses as dots on the maps of counties, it doesn't work well if we set the image of a county as a base, and then add another layer of dots on the image. The precision of plotting in this way is pretty low.</p>	<p>We decided to use a Python package called Geopandas which can perfectly match the latitudes and longitudes collected with the actual locations on the shapefiles of all the counties in Virginia and Austria.</p>
<p>We originally wanted to use Tableau as a visualization tool but after implementing some data and mapping it, our client, Dr. Zach, told us of his dislike of Tableau and we also realized that we would eventually have to start paying for it as well.</p>	<p>We decided to display the data using Python and various libraries in order to have more precise charts and maps than before.</p>
<p>Initially none of the team was very adept at web scraping and there was a lot of code to sift through at the start. Thus, we were lacking direction and it was hard to determine what we were going to do and if we should even continue using the web scraper that we were not even sure worked.</p>	<p>We decided to split the team into a visualization team and a data collection team. This split allowed us to be efficient as we knew we would have to make visualizations regardless of whether the script worked or not. Since there was existing data that our client liked, we decided to model our visualizations and web scraper after the existing data. In this way, we ensure, at the very least, we will have new visualizations for the existing data and at the best we will have new visualizations and an updated web scraper.</p>

**Table 3:** Problems and Solutions

**Future work**

There are several areas where we can include improvements. What we hope our finished product will contain is essentially scraping Airbnb for Virginia and Austria and then displaying this information in several charts and maps and finally putting it online onto our website. For starters, in the future the script may be used to collect information from more areas outside of just Virginia and Austria. The script could also be made automatic, rather than having to run the script manually multiple times which can prove to be quite tedious. Displaying the data could also use some areas of enhancements. For starters, in the future the maps could

be made dynamic rather than static, and have the functionality to zoom in and out as well as highlight certain data points and see what information they will contain.

## 10.Acknowledgments

We would like to thank both our client, Dr. Florian Zach, as well as our professor Dr. Fox.

Their contact information is listed below.

### **Florian Zach, Ph.D.**

- florian@vt.edu

### **Edward A. Fox**

- Web pages: homepage <http://fox.cs.vt.edu>
- Personal info
  - Office: 2160G Torgersen Hall
  - Phone: (540) 231-5113
  - Email: fox@vt.edu
- Address: Dept. of CS, 114 McBryde Hall, Mail Code 0106, Virginia Tech, Blacksburg, VA 24061
- Laboratory: [Digital Library Research Laboratory](#)
- Torgersen Hall room 2030, x3615. The facilities of DLRL and support by students working there, are available to the class.

## 11. References

1. Gessner, Kate. "Ahead of IPO, Airbnb's Consumer Sales Surpass Most Hotel Brands." *Second Measure*, 25 Mar. 2019, [secondmeasure.com/datapoints/airbnb-sales-surpass-most-hotel-brands/](https://secondmeasure.com/datapoints/airbnb-sales-surpass-most-hotel-brands/). Accessed 4 May 2020.
2. "Get the Data." Inside Airbnb, [insideairbnb.com/get-the-data.html](https://insideairbnb.com/get-the-data.html). Accessed 4 May 2020.
3. Slee, Tom. "Airbnb-Data-Collection." Github, [github.com/tomslee/airbnb-data-collection](https://github.com/tomslee/airbnb-data-collection). Accessed 4 May 2020.
4. Minkin, Tracey. "We Planned the Perfect Weekend in Virginia Beach." *Coastal Living*, 29 Nov. 2017, [www.coastalliving.com/travel/atlantic/virginia-beach-weekend-getaway](http://www.coastalliving.com/travel/atlantic/virginia-beach-weekend-getaway). Accessed 4 May 2020.
5. Zach, Florian, et al. "GitLab Repository." *GitLab*, Virginia Tech, 26 Apr. 2020, [code.vt.edu/florian/airbnb](https://code.vt.edu/florian/airbnb). Accessed 4 May 2020.
6. "Install Docker Toolbox on Windows." *Docker Docs*, [docs.docker.com/toolbox/toolbox\\_install\\_windows/](https://docs.docker.com/toolbox/toolbox_install_windows/). Accessed 4 May 2020.
7. Docker. (2020). Retrieved from <https://www.docker.com/products/docker-desktop>. Accessed 4 May 2020.
8. "Bounding Box Tool: Metadata Enrichment for Catalogue Records by Visually Selecting Geographic Coordinates (Latitude / Longitude) for Maps." *Bounding Box Tool: Metadata Enrichment for Catalogue Records by Visually Selecting Geographic Coordinates (Latitude / Longitude) for Maps*, [boundingbox.klokantech.com/](https://boundingbox.klokantech.com/). Accessed 4 May 2020.

9. "Downloads." *PostgreSQL*, [www.postgresql.org/download/](http://www.postgresql.org/download/). Accessed 4 May 2020
  
10. "The World's Most Popular Data Science Platform." *Anaconda*, [www.anaconda.com/](http://www.anaconda.com/). Accessed 4 May 2020.