Step 1: Connect to Linux EMR

Connect to Linux EMR by running `ssh yourusername@54.187.148.25` in your terminal/command prompt.

Step 2: Import Airbnb CSV files from GitHub to Linux

This step is to import necessary Airbnb CSV files manually from GibHub by performing the `wget` command. Files can be accessed at https://github.com/anniechen61/CIS4560

Retrieve *la_listings.csv* and *la_reviews.csv* using:

```
wget
https://raw.githubusercontent.com/anniechen61/CIS4560/maste
r/la_listings.csv
```

```
wget
https://raw.githubusercontent.com/anniechen61/CIS4560/maste
r/la_reviews.csv
```

A. Retrieve *sf_listings.csv* and *sf_reviews.csv* using:

```
wget
https://raw.githubusercontent.com/anniechen61/CIS4560/maste
r/sf_listings.csv
```

```
wget
https://raw.githubusercontent.com/anniechen61/CIS4560/maste
r/sf_reviews.csv
```

Step 3: Create directories in Hadoop for Airbnb datasets

By doing this step we are creating directories to store our files.

A. Create a directory for Airbnb using `hdfs dfs -mkdir`:

```
hdfs dfs -mkdir airbnb
```

B. Create two more directories within **airbnb** called **la** and **sf**:

```
hdfs dfs -mkdir airbnb/la

hdfs dfs -mkdir airbnb/sf
```

Step 3: Put files in the correct directories

Before we start editing our data, it needs to be put in the correct directories for organization.

A. First check to see if your files are uploaded using `ls`.

B. Put *la** files in **airbnb/la** directory:

```
hdfs dfs -put la_listings.csv airbnb/la

hdfs dfs -put la_reviews.csv airbnb/la
```

C. Put *sf** files in airbnb/sf directory:

```
hdfs dfs -put sf_listings.csv airbnb/sf

hdfs dfs -put sf_reviews.csv airbnb/sf
```

D. Check if all files are in the designated directory

```
hdfs dfs -ls airbnb/la
```

Your result should look similar to this:

```
Found 2 items
-rw-r--r--   1 mduong11 hadoop    6224964 2019-12-07 02:40 airbnb/la/la_listings.csv
-rw-r--r--   1 mduong11 hadoop   33738359 2019-12-07 02:40 airbnb/la/la_reviews.csv
```

```
hdfs dfs -ls airbnb/sf
```

Your result should look similar to this:

```
Found 2 items
-rw-r--r--   1 mduong11 hadoop    1158361 2019-12-07 03:37 airbnb/sf/sf_listings.csv
-rw-r--r--   1 mduong11 hadoop   11711138 2019-12-07 03:37 airbnb/sf/sf_reviews.csv
```

Step 3: Running Pig Script in Grunt shell

In this step we are going to run Pig script in grunt shell by simply executing the command `pig` in Linux. Open another Terminal and execute the command `pig` in Linux.

A. Load data with schema:

```
lalistings = LOAD
'/user/mduong11/airbnb/la/la_listings.csv' USING
PigStorage(',') AS (
id:chararray,
name:chararray,
host_id:chararray,
host_name:chararray,
neighborhood:chararray,
latitude:double,
longitude:double,
room_type:chararray,
price:double,
minimum_nights:int,
number_of_reviews:int,
last_review:datetime,
reviews_per_month:double);

sflistings = LOAD
'/user/mduong11/airbnb/sf/sf_listings.csv' USING
PigStorage(',') AS (
id:chararray,
name:chararray,
host_id:chararray,
host_name:chararray,
neighborhood:chararray,
latitude:double,
longitude:double,
room_type:chararray,
price:double,
minimum_nights:int,
number_of_reviews:int,
last_review:datetime,
reviews_per_month:double);
```

! this part must be changed to your username !

B. Describe the schema:

```
describe lalistings;
```

Your result should look similar to this:

```
lalistings: {id: chararray,name: chararray,host_id: chararray,host_name:
chararray,neighborhood: chararray,latitude: double,longitude: double,room_type:
chararray,price: double,minimum_nights: int,number_of_reviews: int,last_review:
datetime,reviews_per_month: double}
```

C. Clean *la_listings.csv* and *sf_listings.csv*.

   a. Get rid of records that do not have *reviews_per_month* by using `FILTER`:

   ```
   lahasreviews = FILTER lalistings BY reviews_per_month
   IS NOT NULL;

   sfhasreviews = FILTER sflistings BY reviews_per_month
   IS NOT NULL;
   ```

   b. Get rid of records that do not have location data (longitude and latitude)

   ```
   lahaslongitude = FILTER lahasreviews BY longitude IS
   NOT NULL;

   lahaslatitude = FILTER lahaslongitude BY latitude IS
   NOT NULL;

   sfhaslongitude = FILTER sfhasreviews BY longitude IS
   NOT NULL;

   sfhaslatitude = FILTER sfhaslongitude BY latitude IS
   NOT NULL;
   ```

   c. Dump `lahaslatitude` to check.

   ```
   dump lahaslatitude;
   ```

```
dump sfhaslatitude;
```

Your `dump lahaslatitude` result should look similar to this:

```
…
(38494976,Private LA LOFT with separate bedroom. Very cool,21013529,The
Rosemary Hospitality,Downtown,34.04431,-118.25342,Entire
home/apt,120.0,1,1,2019-09-11T00:00:00.000Z,1.0)
(38520114,Room 4 (Master Bedroom) -
HTC,293579920,Hung,Lancaster,34.675,-118.18812,Private
room,50.0,1,1,2019-09-12T00:00:00.000Z,1.0)
(38525272,Exquisite Hollywood Hills Town-House,53521418,Kathleen,Hollywood
Hills,34.12014,-118.34132,Entire home/apt,650.0,2,1,2019-09-11T00:00:00.000Z,1.0)
(38531041,核桃市独栋别墅单独房间  中文房东 租车服务  迪士尼 环球影城 星光大道 大
吉精品民宿3号,252366287,葵,Walnut,34.01123,-117.86498,Private
room,100.0,1,1,2019-09-13T00:00:00.000Z,1.0)
(38551243,Cozy bright apartment in the best part of
DTLA,177439206,Radouane,Downtown,34.04712,-118.25133,Private
room,75.0,2,1,2019-09-12T00:00:00.000Z,1.0)

…
```

D. Create a new relation with total price, including 14% tax for LA Rental

```
lanew_listings = FOREACH lahaslatitude GENERATE
neighborhood, id, price + (price * 0.14) AS
finalprice:double, latitude, longitude, room_type,
reviews_per_month;
```

E. Group the neighborhood by `finalprice`.

   i. Create a new relation with only neighborhood and `finalprice` from la*new_listings* table.

```
laprice = FOREACH lanew_listings GENERATE finalprice,
neighborhood;
```

   ii. Group price by neighborhood and see results.

```
lapricebyneighborhood = GROUP laprice BY neighborhood;

dump lapricebyneighborhood;
```

   iii. Calculate the average price of each region to compare the most expensive/cheap place to live.

```
latotals = FOREACH lapricebyneighborhood GENERATE
group, AVG(laprice.finalprice) AS lafinalprice;
```

iv. Describe `totals`.

```
describe latotals;
```

Your result should look similar to this:

```
latotals: {group: chararray,lafinalprice: double}
```

v. Sort the data by price and see the top 15 most expensive places to live.

```
lasortedpricedesc = ORDER latotals BY lafinalprice
DESC;

latop15 = LIMIT lasortedpricedesc 15;

dump latop15;
```

Your result should look similar to this:

```
...
(Rolling Hills,2825.2999999999997)
(Malibu,1068.3962068965504)
(Beverly Crest,1052.7947107438017)
(Bel-Air,707.8565853658537)
(Hollywood Hills West,665.9661437908482)
(Athens,655.5)
(Palos Verdes Estates,613.6457142857142)
(Unincorporated Catalina Island,609.6150000000001)
(Unincorporated Santa Monica Mountains,482.3117241379313)
(Avalon,469.44529411764705)
(Encino,407.43983193277313)
(North Whittier,399.0)
(Pacific Palisades,363.8786301369863)
(Rancho Palos Verdes,349.5434042553191)
(Universal City,347.7)
```

E. Store `sortedpricedesc` into airbnb/la directory.

```
store lasortedpricedesc INTO 'airbnb/la/sorted_avg_price'
USING PigStorage(',');
```

Now we are going to repeat steps 3D - 3E for SF using the following commands:

A. Create a new relation with total price, including 14% tax for SF Rental

```
sfnew_listings = FOREACH sfhaslatitude GENERATE
neighborhood, id, price + (price * 0.14) AS
finalprice:double, latitude, longitude, room_type,
reviews_per_month;
```

B. Group the neighborhood by `finalprice`.

vi. Create a new relation with only neighborhood and `finalprice` from sf*new_listings* table.

```
sfprice = FOREACH sfnew_listings GENERATE finalprice,
neighborhood;
```

vii. Group price by neighborhood and see results.

```
sfpricebyneighborhood = GROUP sfprice BY neighborhood;

dump sfpricebyneighborhood;
```

viii. Calculate the average price of each region to compare the most expensive/cheap place to live.

```
sftotals = FOREACH sfpricebyneighborhood GENERATE
group, AVG(sfprice.finalprice) AS sffinalprice;
```

ix. Describe `totals`.

```
describe sftotals;
```

Your result should look similar to this:

```
sftotals: {group: chararray,sffinalprice: double}
```

x. Sort the data by price and see the top 15 most expensive places to live.

```
sfsortedpricedesc = ORDER sftotals BY sffinalprice
DESC;
```

```
sftop15 = LIMIT sfsortedpricedesc 15;

dump sftop15;
```

Your result should look similar to this:

```
(Presidio Heights,589.912)
(Golden Gate Park,397.29)
(Inner Sunset,379.03538461538454)
(Pacific Heights,369.14508196721306)
(Marina,366.5234645669293)
(Russian Hill,350.092131147541)
(South of Market,309.4539884393066)
(Potrero Hill,297.5737278106509)
(Western Addition,281.8229055690073)
(Castro/Upper Market,274.6314285714288)
(Seacliff,268.26000000000005)
(Parkside,267.63599999999997)
(Noe Valley,265.9084337349398)
(Twin Peaks,260.0495454545455)
(Inner Richmond,258.68185430463564)
```

F. Store `sfsortedpricedesc` into airbnb/la directory.

```
store sfsortedpricedesc INTO 'airbnb/sf/sorted_avg_price'
USING PigStorage(',');
```

Step 4: Downloading the files using SFTP

In this step we are going to store `sortedpricedesc` into correct directory and download the file using **sftp**.

A. Go back to Hadoop cluster and run the following hdfs commands to see "/user/*yourusername*/airbnb/la/sorted_avg_price" folder.

```
hdfs dfs -ls airbnb/la/sorted_avg_price
```

Your results should look similar to this:

```
Found 2 items
```

```
-rw-r--r-- 1 ychen148 hadoop 0 2019-12-06 23:22
airbnb/la/sorted_avg_price/_SUCCESS
-rw-r--r-- 1 ychen148 hadoop 7449 2019-12-06 23:22
airbnb/la/sorted_avg_price/part-v004-o000-r-00000
```

```
hdfs dfs -ls airbnb/sf/sorted_avg_price
```

Your results should look similar to this:

```
Found 2 items
-rw-r--r--   1 ychen148 hadoop          0 2019-12-08 19:44
airbnb/sf/sorted_avg_price/_SUCCESS
-rw-r--r--   1 ychen148 hadoop       1102 2019-12-08 19:44
airbnb/sf/sorted_avg_price/part-v004-o000-r-00000
```

B. You can see the content of the output file as follows:

```
hdfs dfs -cat
airbnb/la/sorted_avg_price/part-v004-o000-r-00000
```

Your results should look similar to this:

```
Rolling Hills,2825.2999999999997
Malibu,1068.3962068965504
Beverly Crest,1052.7947107438017
Bel-Air,707.8565853658537
Hollywood Hills West,665.9661437908482
Athens,655.5
Palos Verdes Estates,613.6457142857142
Unincorporated Catalina Island,609.6150000000001
Unincorporated Santa Monica Mountains,482.3117241379313
Avalon,469.44529411764705
Encino,407.43983193277313
North Whittier,399.0
Pacific Palisades,363.8786301369863
Rancho Palos Verdes,349.5434042553191
Universal City,347.7
Manhattan Beach,329.19268965517244
Beverly Hills,319.3294054054057
Beverly Grove,301.75626898047733
...
```

```
hdfs dfs -cat
airbnb/sf/sorted_avg_price/part-v004-o000-r-00000
```

Your results should look similar to this:

```
Presidio Heights,589.912
Golden Gate Park,397.29
Inner Sunset,379.03538461538454
Pacific Heights,369.14508196721306
Marina,366.5234645669293
Russian Hill,350.092131147541
South of Market,309.4539884393066
Potrero Hill,297.5737278106509
Western Addition,281.8229055690073
Castro/Upper Market,274.6314285714288
Seacliff,268.26000000000005
Parkside,267.63599999999997
Noe Valley,265.9084337349398
...
```

! double check this part of your file, it may contain a different name !

C. LA: Download the output file “part-v004-o000-r-00000” as *neighborhoodbyprice.csv* using the following hdfs command:

```
hdfs dfs -get
airbnb/la/sorted_avg_price/part-v004-o000-r-00000
laneighborhoodbyprice.csv
```

SF: Download the output file “part-v004-o000-r-00000” as sf*neighborhoodbyprice.csv* using the following hdfs command:

```
hdfs dfs -get
airbnb/sf/sorted_avg_price/part-v004-o000-r-00000
sfneighborhoodbyprice.csv
```

D. Open another terminal and go to **sftp** to get your la*neighborhoodbyprice.csv and sfneighborhoodbyprice.csv* file.

```
sftp mduong11@54.187.148.25
```

! this part must be changed to your username !

E. Download the file using get in **sftp**.

```
get laneighborhoodbyprice.csv
```

Your results should look similar to this:

```
sftp> get laneighborhoodbyprice.csv
Fetching /home/ychen148/laneighborhoodbyprice.csv to
laneighborhoodbyprice.csv
```

```
/home/ychen148/laneighborhoodbyprice.csv      100% 7449
81.2KB/s   00:00
```

```
get sfneighborhoodbyprice.csv
```

Your results should look similar to this:

```
sftp> get sfneighborhoodbyprice.csv
Fetching /home/ychen148/sfneighborhoodbyprice.csv to
sfneighborhoodbyprice.csv
/home/ychen148/sfneighborhoodbyprice.csv      100% 1102
12.3KB/s   00:00
```

Step 5: Clean *reviews.csv* and Perform JOIN

Step 4 was an exercise to display only neighborhood by price. Now, we want to produce 2 worksheets that joins lanew_listings with lareviews and sfnew_listings with sfreviews. Open your terminal that is in grunt shell and start loading lareviews and sfreviews.

A. Load *la_reviews.csv* files with schema and describe the schema to double check.

```
lareviews = LOAD '/user/mduong11/airbnb/la/la_reviews.csv'
USING PigStorage(',') AS (
date:datetime,
listing_id:chararray,
reviewer_id:chararray,
reviewer_name:chararray);

sfreviews = LOAD '/user/mduong11/airbnb/sf/sf_reviews.csv'
USING PigStorage(',') AS (
date:datetime,
listing_id:chararray,
reviewer_id:chararray,
reviewer_name:chararray);
```

! this part must be changed to your username !

```
describe lareviews;
```

Your results should look similar to this:

lareviews: {date: chararray,listing_id: chararray,reviewer_id: chararray,reviewer_name: chararray}

```
describe sfreviews;
```

Your results should look similar to this:

sfreviews: {date: datetime,listing_id: chararray,reviewer_id: chararray,reviewer_name: chararray}

B. Join `lanew_listings` with `lareviews`.

```
lajoined = JOIN lanew_listings by id, lareviews by
listing_id;
```

C. Clean schema and describe to double check.

```
lacleaned = FOREACH lajoined GENERATE
lanew_listings::neighborhood,
lanew_listings::id,
lanew_listings::finalprice,
lanew_listings::latitude,
lanew_listings::longitude,
lanew_listings::room_type,
lanew_listings::reviews_per_month,
lareviews::date,
lareviews::reviewer_id,
lareviews::reviewer_name;

describe lacleaned;
```

Your results should look similar to this:

```
grunt> describe lacleaned;
lacleaned: {lanew_listings::neighborhood:
chararray,lanew_listings::id:
chararray,lanew_listings::finalprice:
double,lanew_listings::latitude:
double,lanew_listings::longitude:
double,lanew_listings::room_type:
```

```
chararray,lanew_listings::reviews_per_month:
double,lareviews::date: datetime,lareviews::reviewer_id:
chararray,lareviews::reviewer_name: chararray}
```

D. Store `lacleaned` into correct directory and download the file using **sftp**.

```
store lacleaned INTO 'airbnb/la/lajoined' USING
PigStorage(',');
```

Repeat steps 5B - 5D for SF:

A. Join `sfnew_listings` with `sfreviews`.

```
sfjoined = JOIN sfnew_listings by id, sfreviews by
listing_id;
```

B. Clean schema and describe to double check.

```
sfcleaned = FOREACH sfjoined GENERATE
sfnew_listings::neighborhood,
sfnew_listings::id,
sfnew_listings::finalprice,
sfnew_listings::latitude,
sfnew_listings::longitude,
sfnew_listings::room_type,
sfnew_listings::reviews_per_month,
sfreviews::date,
sfreviews::reviewer_id,
sfreviews::reviewer_name;

describe sfcleaned;
```

Your results should look like this:

```
sfcleaned: {sfnew_listings::neighborhood: chararray,sfnew_listings::id:
chararray,sfnew_listings::finalprice: double,sfnew_listings::latitude:
double,sfnew_listings::longitude: double,sfnew_listings::room_type:
chararray,sfnew_listings::reviews_per_month: double,sfreviews::date:
datetime,sfreviews::reviewer_id: chararray,sfreviews::reviewer_name: chararray}
```

C. Store `sfcleaned` into correct directory and download the file using **sftp**.

```
store sfcleaned INTO 'airbnb/sf/sfjoined' USING
PigStorage(',');
```

Step 6: Check for files in directory and download

Similar to Step 4, we are going to double check that the file is in the correct directory and download it using **sftp**.

A. Go back to Hadoop cluster and run the following hdfs commands to see "/user/*yourusername*/airbnb/la/joined" folder.

```
hdfs dfs -ls airbnb/la/lajoined
```

```
hdfs dfs -ls airbnb/sf/sfjoined
```

Your results should look similar to this:

```
Found 2 items
-rw-r--r--   1 ychen148 hadoop          0 2019-12-08 20:05 airbnb/la/lajoined/_SUCCESS
-rw-r--r--   1 ychen148 hadoop   96432384 2019-12-08 20:05
airbnb/la/lajoined/part-v002-o000-r-00000
```

Your results should look similar to this:

```
Found 2 items
-rw-r--r--   1 ychen148 hadoop          0 2019-12-08 20:07 airbnb/sf/sfjoined/_SUCCESS
-rw-r--r--   1 ychen148 hadoop   33370807 2019-12-08 20:07
airbnb/sf/sfjoined/part-v002-o000-r-00000
```

B. You can see the content of the output file as follows:

```
hdfs dfs -cat airbnb/la/joined/part-v004-o000-r-00000
```

! double check this part of your file, it may contain a different name !

C. Download the output file "part-v004-o000-r-00000" as *lajoined.csv* using the following hdfs command:

```
hdfs dfs -get airbnb/la/lajoined/part-v002-o000-r-00000
lajoined.csv

hdfs dfs -get airbnb/sf/sfjoined/part-v002-o000-r-00000
sfjoined.csv
```

D. Go back to **sftp** to get your lajoined.csv file.

```
get lajoined.csv
```

```
Your results should look similar to this:
```

```
sftp> get lajoined.csv
Fetching /home/ychen148/lajoined.csv to lajoined.csv
/home/ychen148/lajoined.csv                   100%   92MB
13.1MB/s   00:07
```

```
get sfjoined.csv
```

```
sftp> get sfjoined.csv
Fetching /home/ychen148/sfjoined.csv to sfjoined.csv
/home/ychen148/sfjoined.csv                        100%   32MB
11.2MB/s   00:02
```

Step 7: Using 3D Maps in Microsoft Excel

In this step, we are going to analyze `lajoined.csv` and `sfjoined.csv` from Step 6 Using the Geo Map feature to display the rental landscape in Los Angeles and San Francisco.

In this step we are going to add headers to *lajoined.csv and sfjoined.csv.*

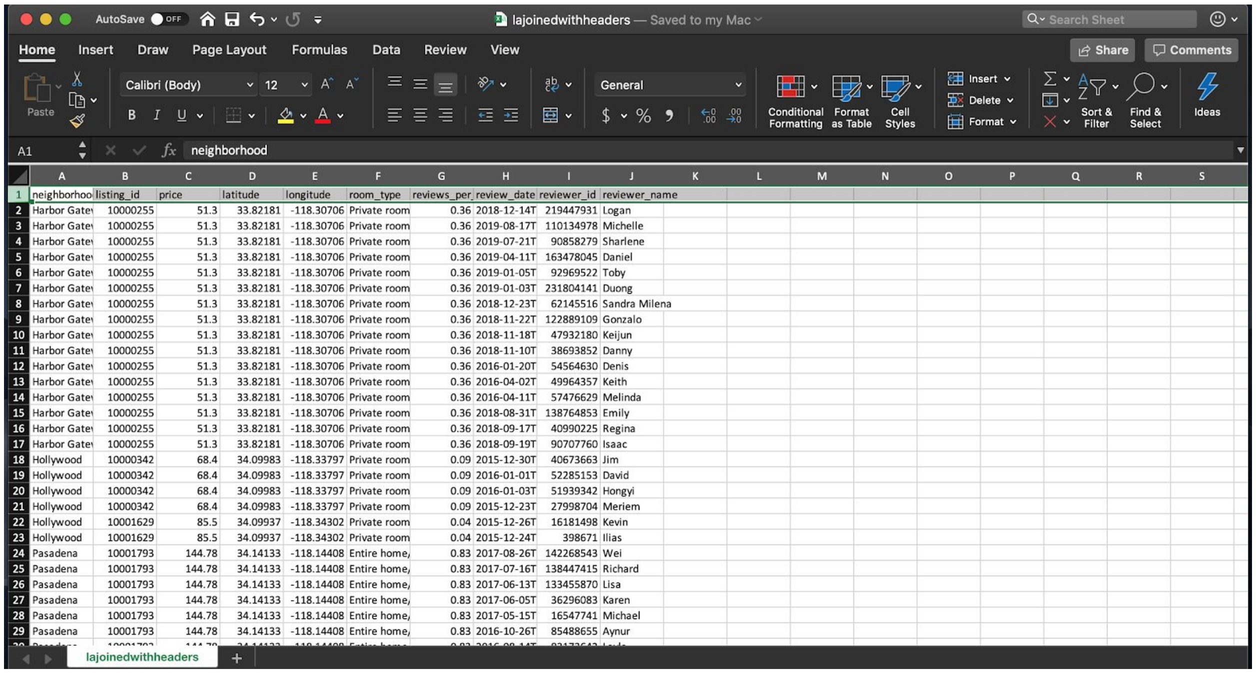
A. Open CSV file, *lajoined.csv*

B. Save file as xslx

C. Insert a row headers(from left to right)

neighborhood
listing_id
price
latitude
longitude

room_type
reviews_per_month
review_date
reviewer_id
reviewer_name


lajoinedwithheaders — Saved to my Mac

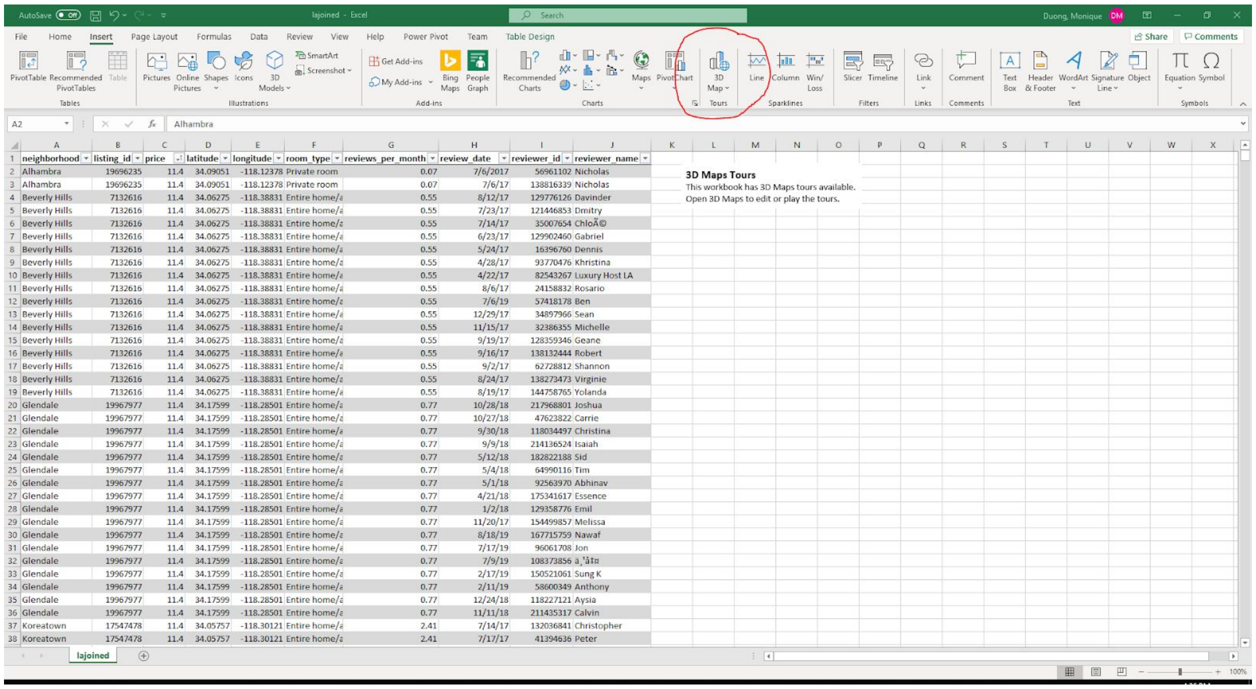
A1 neighborhood

	neighborhoo	listing_id	price	latitude	longitude	room_type	reviews_per	review_date	reviewer_id	reviewer_name
2	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-12-14T	219447931	Logan
3	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2019-08-17T	110134978	Michelle
4	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2019-07-21T	90858279	Sharlene
5	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2019-04-11T	163478045	Daniel
6	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2019-01-05T	92969522	Toby
7	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2019-01-03T	231804141	Duong
8	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-12-23T	62145516	Sandra Milena
9	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-11-22T	122889109	Gonzalo
10	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-11-18T	47932180	Keijun
11	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-11-10T	38693852	Danny
12	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2016-01-20T	54564630	Denis
13	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2016-04-02T	49964357	Keith
14	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2016-04-11T	57476629	Melinda
15	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-08-31T	138764853	Emily
16	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-09-17T	40990225	Regina
17	Harbor Gate\	10000255	51.3	33.82181	-118.30706	Private room	0.36	2018-09-19T	90707760	Isaac
18	Hollywood	10000342	68.4	34.09983	-118.33797	Private room	0.09	2015-12-30T	40673663	Jim
19	Hollywood	10000342	68.4	34.09983	-118.33797	Private room	0.09	2016-01-01T	52285153	David
20	Hollywood	10000342	68.4	34.09983	-118.33797	Private room	0.09	2016-01-03T	51939342	Hongyi
21	Hollywood	10000342	68.4	34.09983	-118.33797	Private room	0.09	2015-12-23T	27998704	Meriem
22	Hollywood	10001629	85.5	34.09937	-118.34302	Private room	0.04	2015-12-26T	16181498	Kevin
23	Hollywood	10001629	85.5	34.09937	-118.34302	Private room	0.04	2015-12-24T	398671	Ilias
24	Pasadena	10001793	144.78	34.14133	-118.14408	Entire home/	0.83	2017-08-26T	142268543	Wei
25	Pasadena	10001793	144.78	34.14133	-118.14408	Entire home/	0.83	2017-07-16T	138447415	Richard
26	Pasadena	10001793	144.78	34.14133	-118.14408	Entire home/	0.83	2017-06-13T	133455870	Lisa
27	Pasadena	10001793	144.78	34.14133	-118.14408	Entire home/	0.83	2017-06-05T	36296083	Karen
28	Pasadena	10001793	144.78	34.14133	-118.14408	Entire home/	0.83	2017-05-15T	16547741	Michael
29	Pasadena	10001793	144.78	34.14133	-118.14408	Entire home/	0.83	2016-10-26T	85488655	Aynur

*Do the same for *sfjoined.xslx*

Create a 3D Map using the data provided

1. Under “Insert” tab click on 3D Map



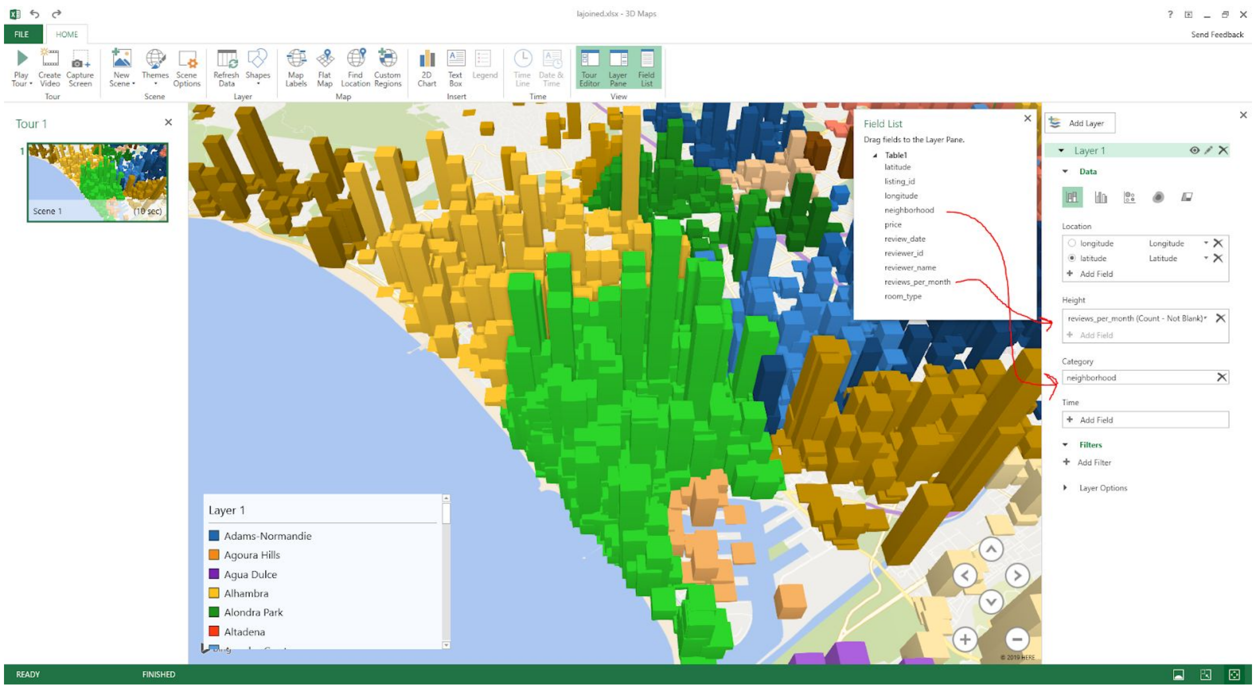
A2 Alhambra

	neighborhood	listing_id	price	latitude	longitude	room_type	reviews_per_month	review_date	reviewer_id	reviewer_name
2	Alhambra	19696235	11.4	34.09051	-118.12378	Private room	0.07	7/6/2017	56961102	Nicholas
3	Alhambra	19696235	11.4	34.09051	-118.12378	Private room	0.07	7/6/17	138816339	Nicholas
4	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	8/12/17	129776126	Davinder
5	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	7/23/17	121446853	Dmitry
6	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	7/14/17	35007654	ChloÃ©
7	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	6/23/17	129902460	Gabriel
8	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	5/24/17	16396760	Dennis
9	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	4/28/17	93770476	Khristina
10	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	4/22/17	82543267	Luxury Host LA
11	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	8/6/17	24158832	Rosario
12	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	7/6/19	57418178	Ben
13	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	12/29/17	34897966	Sean
14	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	11/15/17	32386355	Michelle
15	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	9/19/17	128359346	Geane
16	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	9/16/17	138132444	Robert
17	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	9/2/17	62728812	Shannon
18	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	8/24/17	138273473	Virginie
19	Beverly Hills	7132616	11.4	34.06275	-118.38831	Entire home/a	0.55	8/19/17	144758765	Yolanda
20	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	10/28/18	217968801	Joshua
21	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	10/27/18	47623822	Carrie
22	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	9/30/18	118034497	Christina
23	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	9/9/18	214136524	Isaiah
24	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	5/12/18	182822188	Sid
25	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	5/4/18	64990116	Tim
26	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	5/1/18	92563970	Abhinav
27	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	4/21/18	175341617	Essence
28	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	1/2/18	129358776	Emil
29	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	11/20/17	154499857	Melissa
30	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	8/18/19	167715759	Nawaf
31	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	7/17/19	96061708	Jon
32	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	7/9/19	108373856	ä¸'å‡¤
33	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	2/17/19	150521061	Sung K
34	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	2/11/19	58600349	Anthony
35	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	12/24/18	118227121	Aysia
36	Glendale	19967977	11.4	34.17599	-118.28501	Entire home/a	0.77	11/11/18	211435317	Calvin
37	Koreatown	17547478	11.4	34.05757	-118.30121	Entire home/a	2.41	7/14/17	132036841	Christopher
38	Koreatown	17547478	11.4	34.05757	-118.30121	Entire home/a	2.41	7/17/17	41394636	Peter

3D Maps Tours
This workbook has 3D Maps tours available.
Open 3D Maps to edit or play the tours.

lajoined

2. Click the “**+**” to create a new tour.
3. To create a 3D Map by reviews per month, drag “**neighborhood**” from the **Field List** to **Category** and “**reviews_per_month**” to **Height**.



Field List
Drag fields to the Layer Pane.
Table1
- latitude
- listing_id
- longitude
- neighborhood
- price
- review_date
- reviewer_id
- reviewer_name
- reviews_per_month
- room_type

Add Layer
Layer 1
Data
Location
- longitude: Longitude
- latitude: Latitude
- Add Field

Height
- reviews_per_month (Count - Not Blank)
- Add Field

Category
- neighborhood

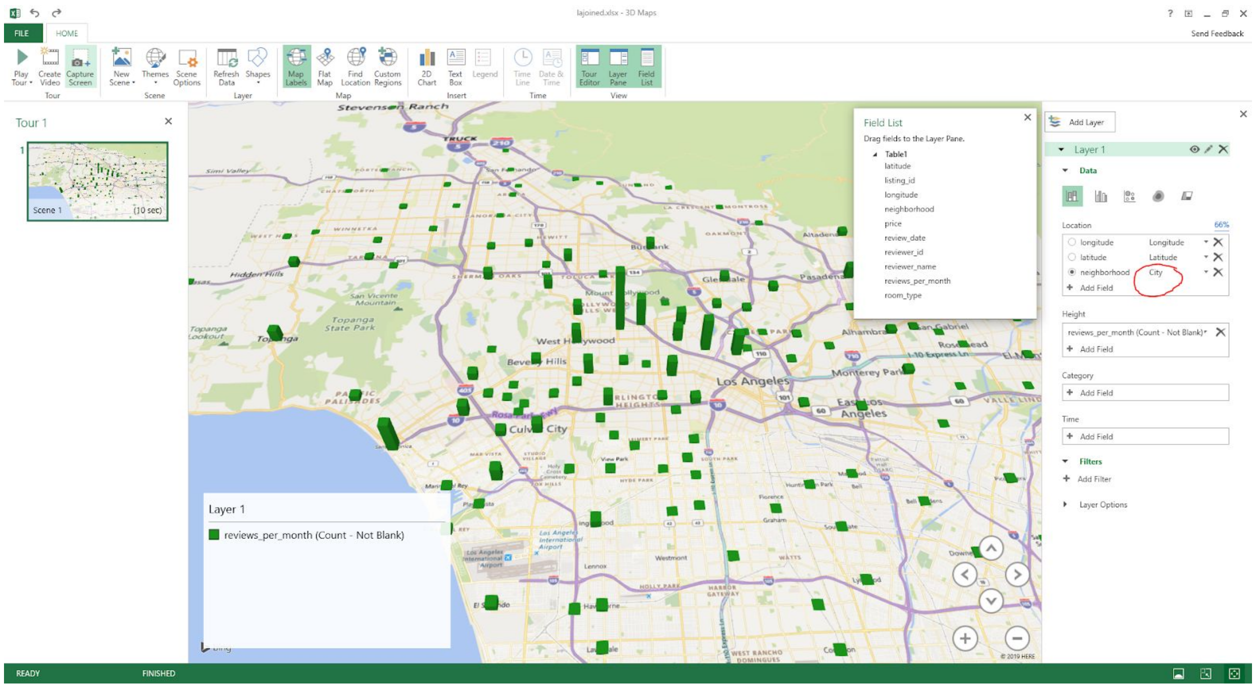
Time
- Add Field

Filters
- Add Filter

Layer Options

Layer 1
- 🟦 Adams-Normandie
- 🟧 Agoura Hills
- 🟪 Agua Dulce
- 🟨 Alhambra
- 🟩 Alondra Park
- 🟥 Altadena

4. To view the amount of **reviews_per_month** for each **neighborhood**, you want to drag **neighborhood** to location then select the type as “City” and keep **reviews_per_month** in Height.



Field List

Drag fields to the Layer Pane.

- Table1
  - latitude
  - listing_id
  - longitude
  - neighborhood
  - price
  - review_date
  - reviewer_id
  - reviewer_name
  - reviews_per_month
  - room_type

Layer 1

Data

Location 66%

longitude	Longitude
latitude	Latitude
neighborhood	City

Add Field

Height

reviews_per_month (Count - Not Blank)

Add Field

Category

Add Field

Time

Add Field

Filters

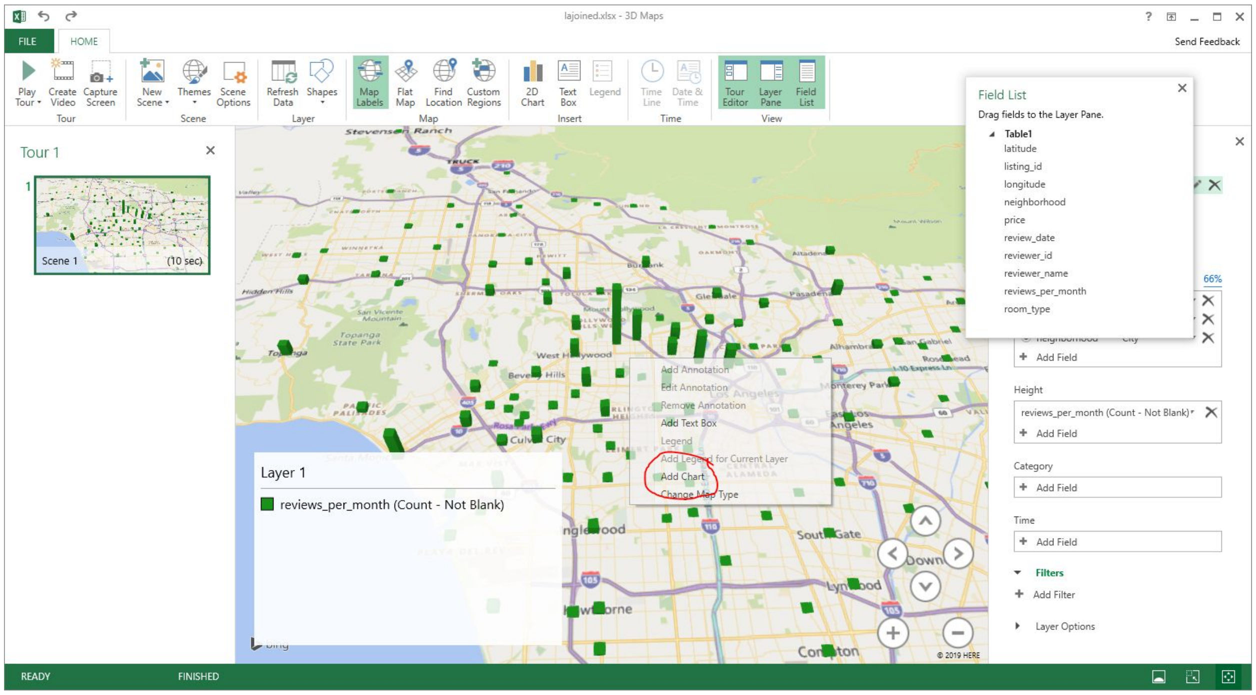
Add Filter

Layer Options

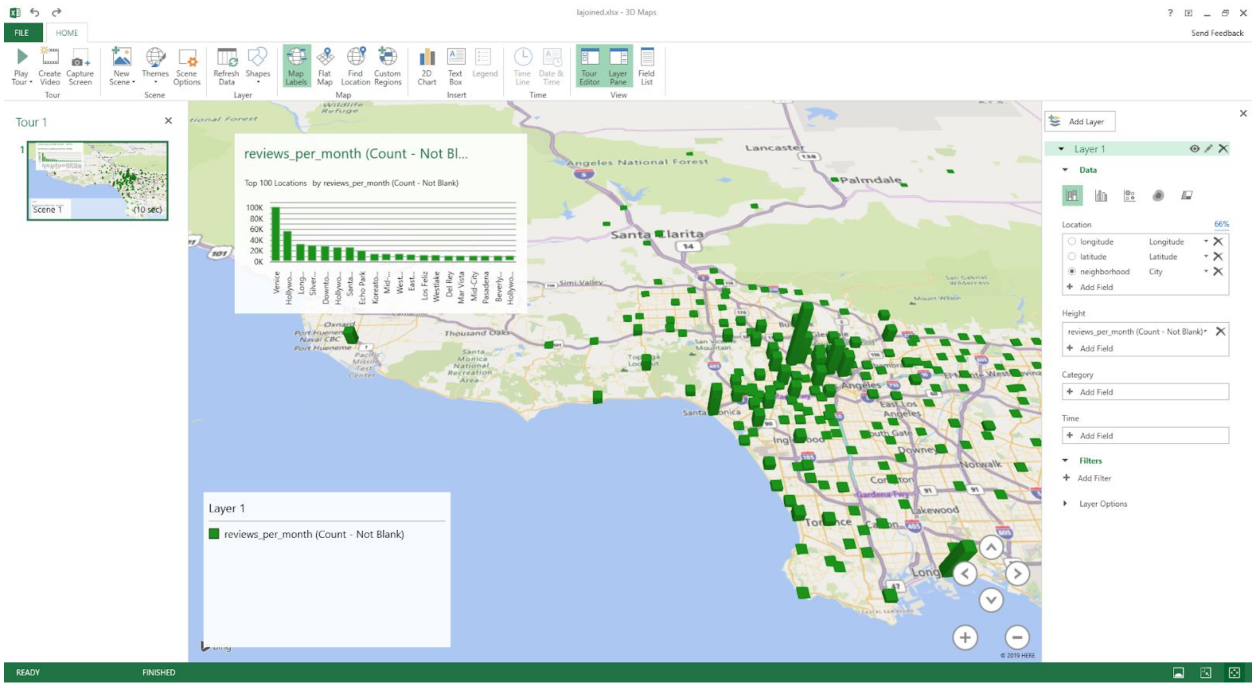
Layer 1

🟩 reviews_per_month (Count - Not Blank)

5. You can also view a chart by right clicking the map and selecting “Add Chart”.

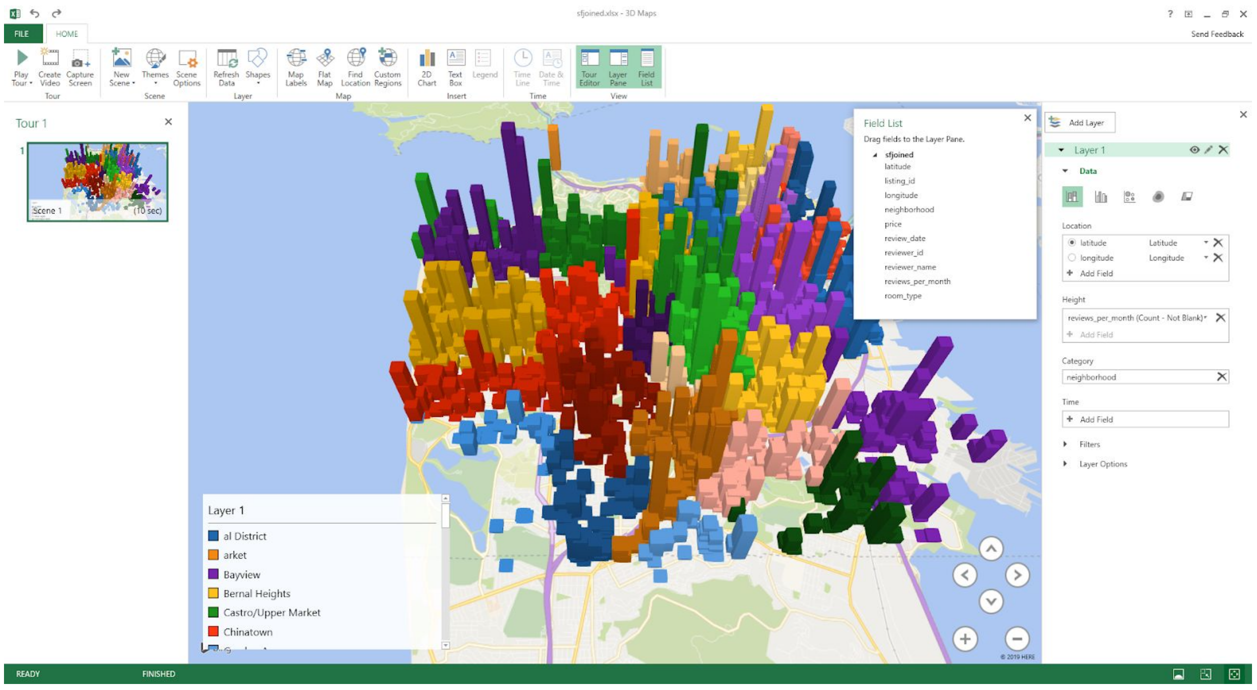


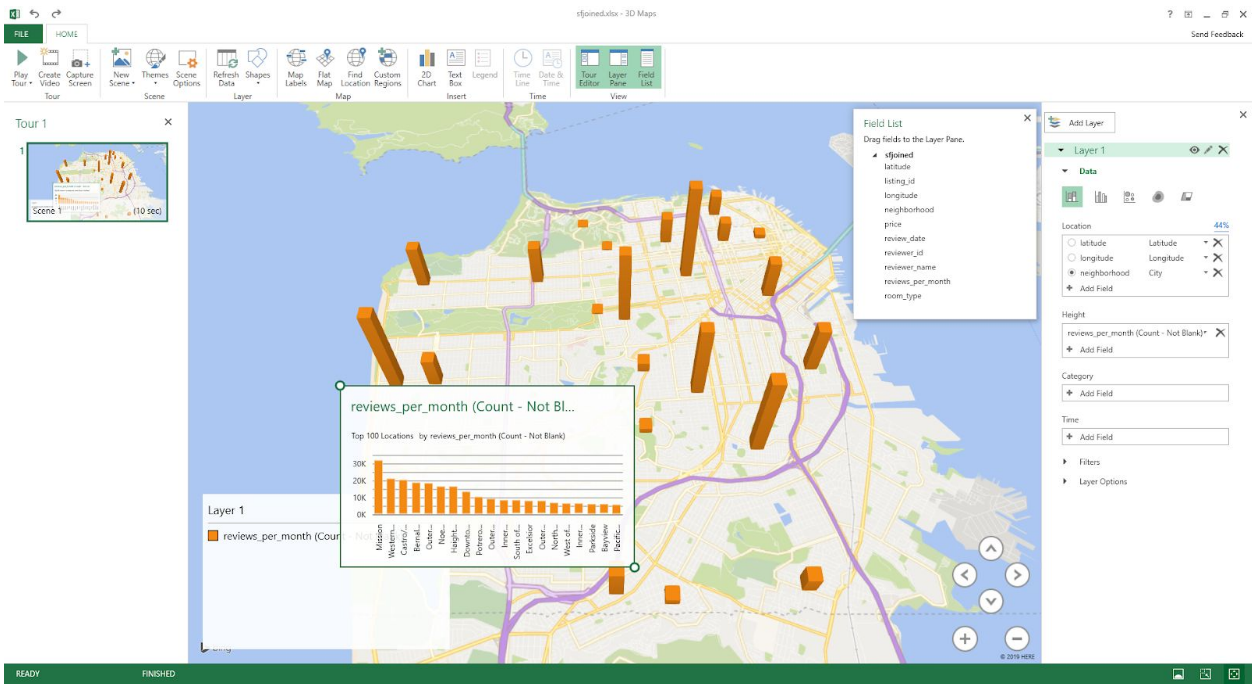
6. Your results will look like this:



*Do the same for *sfjoined.xslx*

Your results will look similar to lajoined.xslx except we can see that it is in San Francisco area.

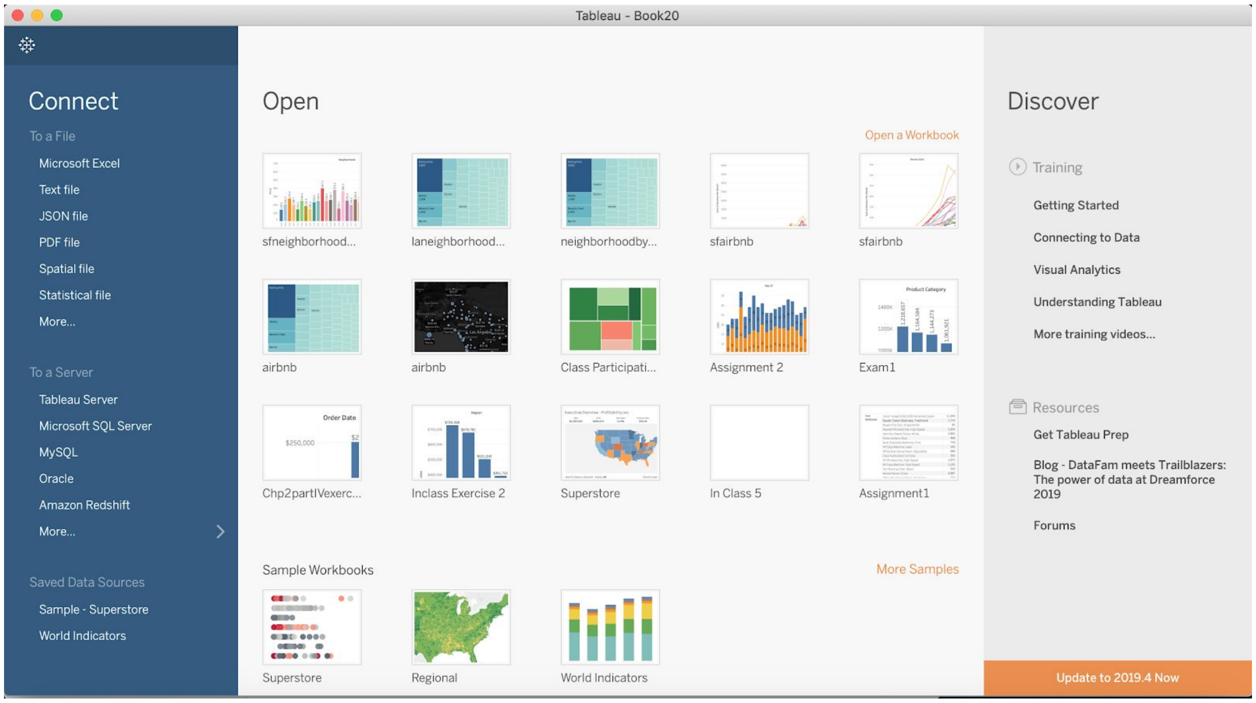




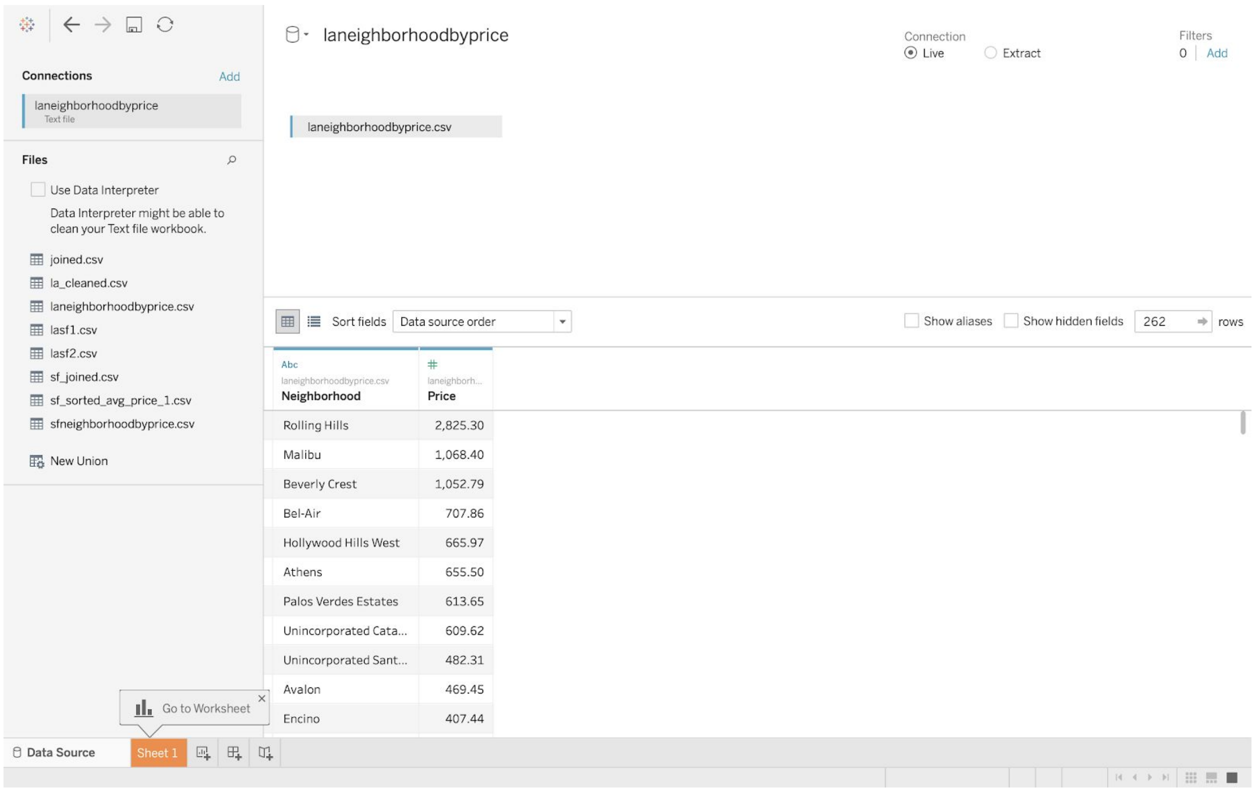
Step 8: Analyzing Neighborhood by Price in Tableau

In this step, we are going to analyze `laneighborhoodbyprice.csv` and `sfneighborhoodbyprice.csv` from Step 4.

1. Open Tableau and click on More to upload your `laneighborhoodbyprice.csv`



2. After loading, click on Sheet 1 at the bottom



laneighborhoodbyprice

Connection: (● Live) ○ Extract

Filters: 0 | Add

Connections — Add

laneighborhoodbyprice
Text file

Files

☐ Use Data Interpreter

Data Interpreter might be able to clean your Text file workbook.

- joined.csv
- la_cleaned.csv
- laneighborhoodbyprice.csv
- lasf1.csv
- lasf2.csv
- sf_joined.csv
- sf_sorted_avg_price_1.csv
- sfneighborhoodbyprice.csv
- New Union

laneighborhoodbyprice.csv

Sort fields: Data source order

☐ Show aliases ☐ Show hidden fields 262 rows

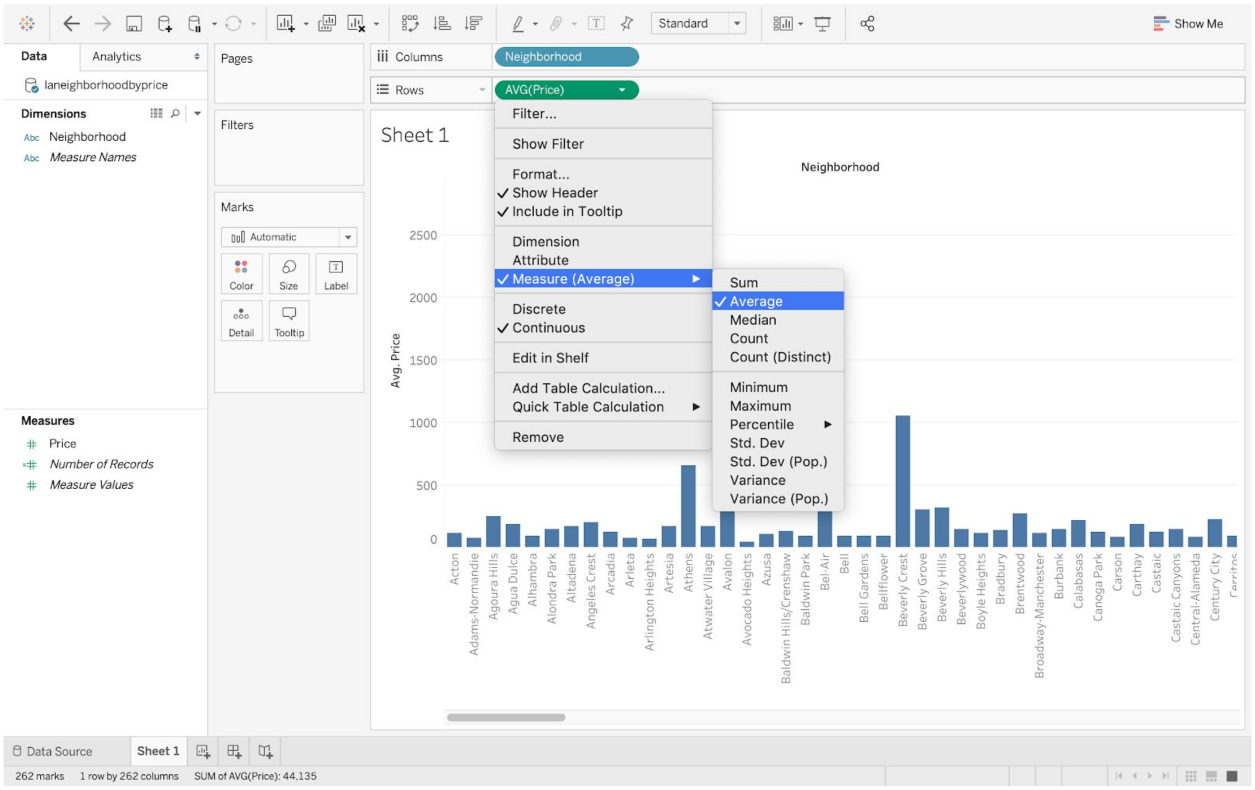
Neighborhood	Price
Rolling Hills	2,825.30
Malibu	1,068.40
Beverly Crest	1,052.79
Bel-Air	707.86
Hollywood Hills West	665.97
Athens	655.50
Palos Verdes Estates	613.65
Unincorporated Cata...	609.62
Unincorporated Sant...	482.31
Avalon	469.45
Encino	407.44

Go to Worksheet

Data Source | Sheet 1

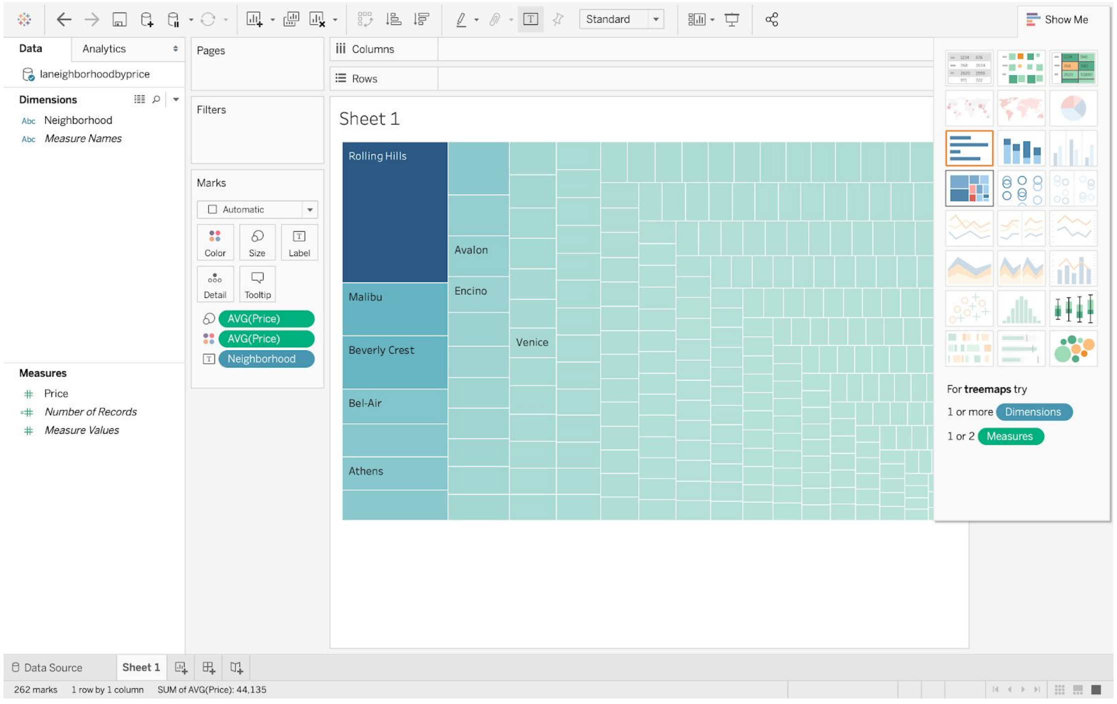
3. In Sheet 1, drag the Dimension Neighborhood to Columns and the Measure Price to Rows. By default, Tableau will turn your data into a Bar Chart. Since there are too many fields in Neighborhood, we will use **Treemap** instead. Make

sure you change the measurement of Price to AVG instead of SUM.



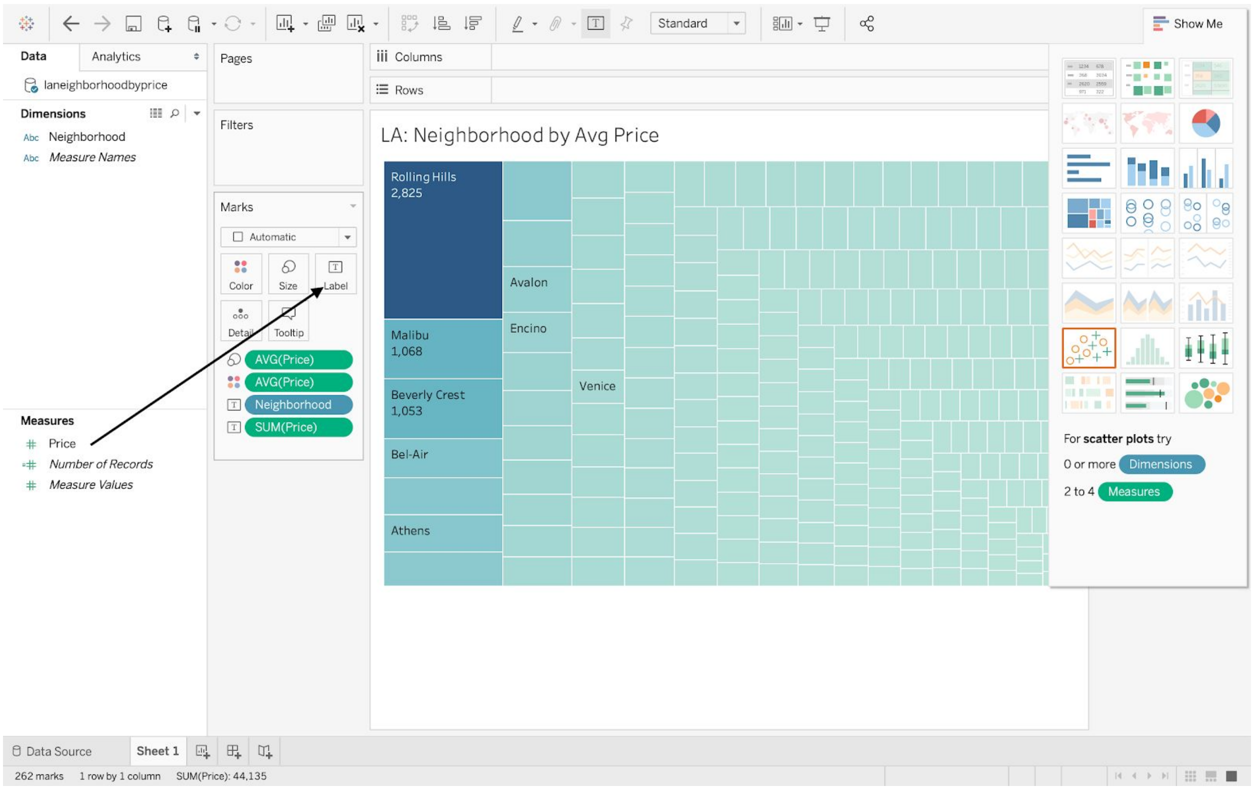
262 marks 1 row by 262 columns SUM of AVG(Price): 44,135

4. Go to your top right and click on Treemap.

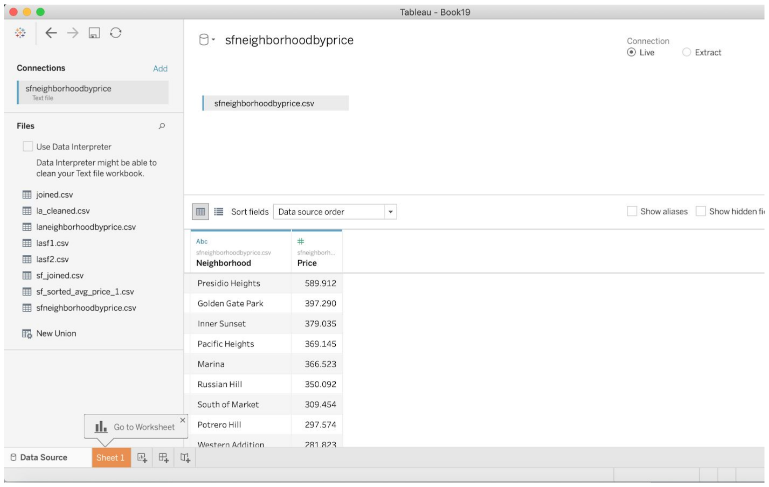


262 marks 1 row by 1 column SUM of AVG(Price): 44,135

5. Drag the Measure Price to the Label Box to show Price in your Treemap.



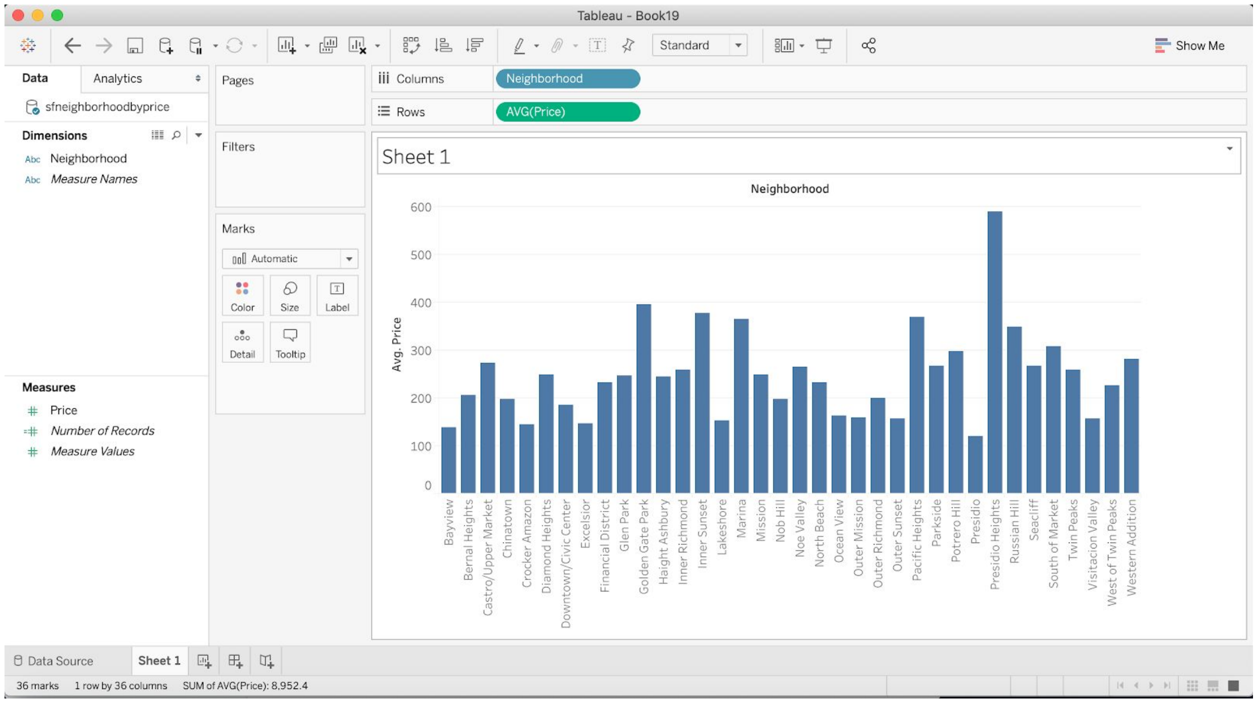
6. Go to File → New, to start a new Tableau.
7. Click on More to upload your `sfneighborhoodbyprice.csv`



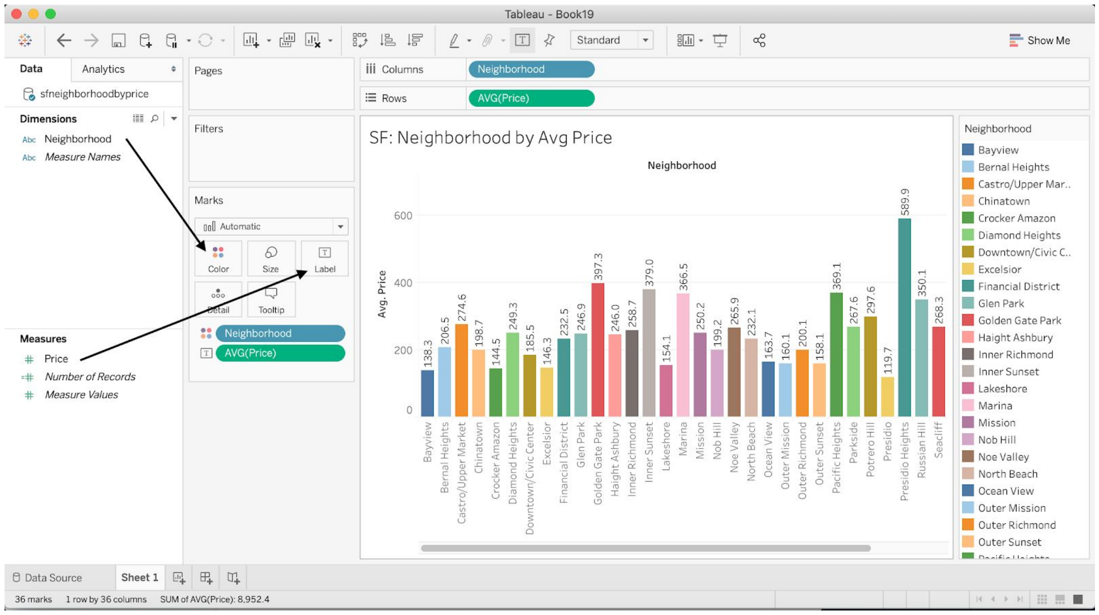
Neighborhood	Price
Presidio Heights	589.912
Golden Gate Park	397.290
Inner Sunset	379.035
Pacific Heights	369.145
Marina	366.523
Russian Hill	350.092
South of Market	309.454
Potrero Hill	297.574
Western Addition	281.823

8. After loading, click on **Sheet 1** at the bottom

9. In Sheet 1, drag the Dimension Neighborhood to Columns and the Measure Price to Rows. This time, we want to use **Bar Graph** as we have a good amount of fields in the Neighborhood Dimension. *Don't forget to change the measurement of price from SUM to AVG*



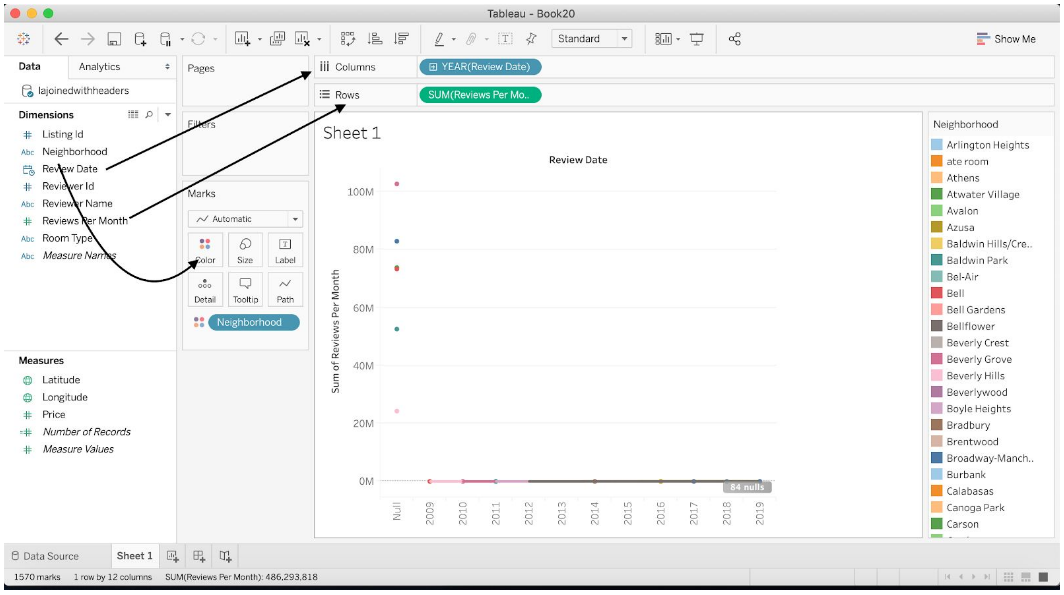
10. Drag and Drop Dimension Neighborhood to Color, Price to Label to show the Price in your Bar Chart.



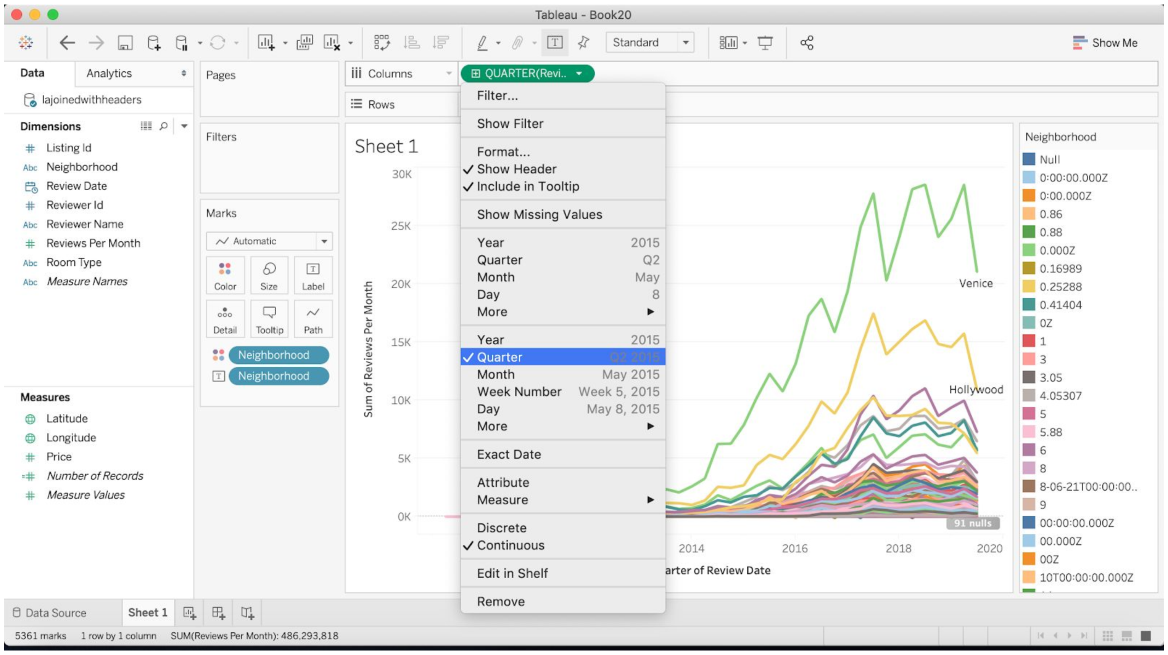
Step 9: Using Tableau to see Trend

In this step, we are going to analyze `lajoined.csv` from Step 6, using Tableau to see the Trend in Los Angeles from 2009 to 2019.

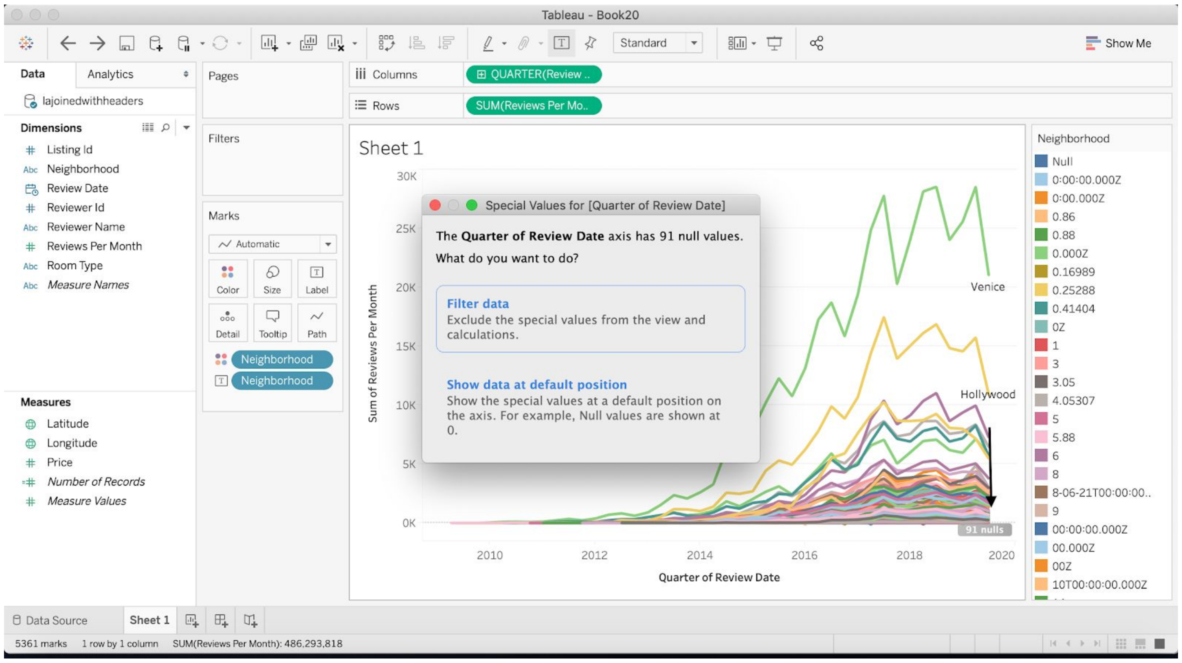
1. Open Tableau and click on More to upload your `lajoined.csv`
    a. Drag Review Date to Column, Reviews Per Month to Rows. *Make sure to change the measurement to SUM for Reviews Per Month.



    b. Filter the Review date to Quarter, Year



c. Click on the nulls and filter data.



Special Values for [Quarter of Review Date]

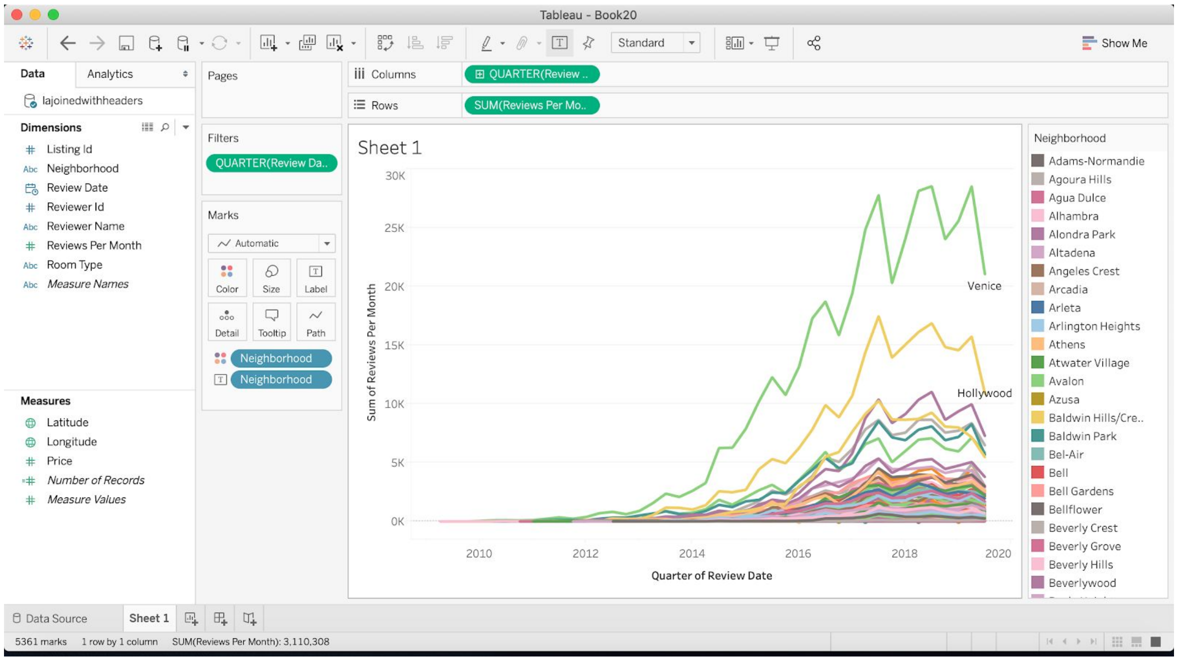
The **Quarter of Review Date** axis has 91 null values.
What do you want to do?

Filter data
Exclude the special values from the view and calculations.

Show data at default position
Show the special values at a default position on the axis. For example, Null values are shown at 0.

5361 marks 1 row by 1 column SUM(Reviews Per Month): 486,293,818

d. Result should look like this



5361 marks 1 row by 1 column SUM(Reviews Per Month): 3,110,308