Project 1-Final Report

-Darshit Doshi and Jared Kohler

We downloaded the Data from SafeGraph https://shop.safegraph.com/ with a coupon code aimed at data science students.

About Safegraph: "We obtain a variety of information (collectively the "Information") from trusted third-party data partners such as mobile application developers. We collect this Information primarily through APIs, which are interfaces through which these app developers can provide us with information about their users. We sometimes collect the Information through other delivery methods, such as software development kits ("SDKs") that are embedded directly into mobile apps."

The dataset we downloaded using is composed of several related tables with information about Cafes and Coffee shops in San Francisco, CA.
The first of these tables is the Core table, containing site information about each of the businesses contained in the dataset.

In [1]:

```
import pandas as pd
import fiona, os, shapely,json, copy
from shapely.geometry import shape, mapping
import geopandas as gpd
from geopandas import GeoSeries, GeoDataFrame
from IPython.display import Image
```

In [2]:

```
core = pd.read_csv('Data/CA-722515-CORE_POI-2019-09-27.csv')
```

In [3]:

```
core.T
```

Out[3]:

	0	1	
safegraph_place_id	sg:10c639a044e74f8bb976ace7d13429ed	sg:eb7b5998f21447ba89dd8ab35a582dfe	sg:118c95bf3c4e4370920d7a026fd
parent_safegraph_place_id	NaN	NaN	
location_name	La Boulangerie De San Francisco Hayes Valley	The Cafe	Royal Ground (
safegraph_brand_ids	NaN	NaN	
brands	NaN	NaN	
top_category	Restaurants and Other Eating Places	Restaurants and Other Eating Places	Restaurants and Other Eating I
sub_category	Snack and Nonalcoholic Beverage Bars	Snack and Nonalcoholic Beverage Bars	Snack and Nonalcoholic Beverag
naics_code	722515	722515	7
street_address	500 hayes street	2369 market street	5301 geary bou
city	san francisco	san francisco	san fra
state	ca	ca	
zip_code	94102	94114	
phone_number	1.4154e+10	1.41578e+10	
open_hours	NaN	{ "Mon": [["0:00", "2:00"]], "Tue": [["18:00",...	{ "Mon": [["6:30", "22:00"]], [["6:

14 rows × 659 columns

The main information from this table that is useful to our analysis is the `open_hours` column.

In [4]:

```
core.open_hours.loc[2]
```

Out[4]:

```
'{ "Mon": [["6:30", "22:00"]], "Tue": [["6:30", "22:00"]], "Wed": [["6:30", "22:00"]], "Thu": [["6
:30", "22:00"]], "Fri": [["6:30", "22:00"]], "Sat": [["6:30", "22:00"]], "Sun": [["6:30",
"22:00"]] }'
```

The second table is called the Patterns table, and it contains most of the useful information. Using the same primary key for each business as in the last table, this table contains many additional dimensions about traffic to the business.

The table loaded below includes foot traffic data for a one month period between August and September of this year (2019).

In [5]:

```
patterns = pd.read_csv('Data/CA-722515-PATTERNS-2019_08-2019-09-27.csv')
```

In [6]:

```
patterns.head()
```

Out[6]:

	safegraph_place_id	location_name	street_address	city	state	zip_code	brands	date_range_start	date_rang
0	sg:eb7b5998f21447ba89dd8ab35a582dfe	The Cafe	2369 market street	san francisco	ca	94114	NaN	1564617600	15672
1	sg:118c95bf3c4e4370920d7a026fd3ebbe	Royal Ground Coffee	5301 geary boulevard	san francisco	ca	94121	NaN	1564617600	15672
2	sg:d9f425fc18a24414b05a053442d1e408	Chestnut Street Coffee Roastery	2331 chestnut street	san francisco	ca	94123	NaN	1564617600	15672
3	sg:de408f31b2cd4eb783cc5de4ad6ed651	Wholesome Bakery	299 divisadero street	san francisco	ca	94117	NaN	1564617600	15672
4	sg:fcc6a71553c04140863220456ff2f4e6	Project Juice	2234 chestnut street	san francisco	ca	94123	Project Juice	1564617600	15672

5 rows × 23 columns

In [7]:

```
patterns.T
```

Out[7]:

	0	1	
safegraph_place_id	sg:eb7b5998f21447ba89dd8ab35a582dfe	sg:118c95bf3c4e4370920d7a026fd3ebbe	sg
location_name	The Cafe	Royal Ground Coffee	
street_address	2369 market street	5301 geary boulevard	
city	san francisco	san francisco	
state	ca	ca	
zip_code	94114	94121	
brands	NaN	NaN	
date_range_start	1564617600	1564617600	
date_range_end	1567296000	1567296000	
raw_visit_counts	163	50	
raw_visitor_counts	146	23	
visits_by_day	[4,6,12,9,1,1,1,5,11,8,7,0,1,4,7,10,13,10,4,3,...	[1,2,1,3,1,0,2,1,3,0,1,0,5,2,1,0,0,1,2,1,0,1,1...	[2,0,3

visitor_home_cbgs	{}	{}	
visitor_work_cbgs	{}	{}	
visitor_country_of_origin	{"US":138}	{"US":22}	
distance_from_home	5986	2082	
median_dwell	39	683.5	
bucketed_dwell_times	{"<5":4,"5-20":59,"21-60":35,"61-240":56,">240...	{"<5":0,"5-20":8,"21-60":8,"61-240":2,">240":32}	{"<5":6,"5
related_same_day_brand	{"Subway":18,"In-N-Out Burger":17,"Shell Oil":...	{}	
related_same_month_brand	{"Starbucks":40,"Subway":32,"Chevron":24,"Targ...	{"Target":42,"Starbucks":37,"Walgreens":25,"Ch...	{"Starbucl
popularity_by_hour	[62,51,13,3,4,4,4,4,5,6,5,9,10,14,21,20,19,14,...	[33,33,32,32,33,32,32,30,18,7,6,4,3,1,0,0,0,7,...	[4,4,4,4,4
popularity_by_day	{"Monday":7,"Tuesday":7,"Wednesday":9,"Thursda...	{"Monday":8,"Tuesday":9,"Wednesday":8,"Thursda...	{"Monday":5,'
device_type	{"android":45,"ios":101}	{"android":8,"ios":15}	

23 rows × 535 columns

Initially cleaning removes data columns that are either not relevant to current analysis are are too sparsely recorded to be of use.

In [8]:

```
clean_patterns = patterns.drop(columns=['city', 'state', 'date_range_start', 'date_range_end', 'vis
itor_home_cbgs', 'visitor_work_cbgs', 'visitor_country_of_origin', 'related_same_day_brand',
'related_same_month_brand', 'device_type'])
```

In [9]:

```
clean_patterns_core = pd.merge(clean_patterns, core[['safegraph_place_id', 'open_hours']], left_on
='safegraph_place_id', right_on='safegraph_place_id', how = 'left')
```

Dropping incorrectly included night club 'The cafe' which appears to have been incorrectly included in the data set of cafes.

In [10]:

```
clean_patterns_core = clean_patterns_core.drop(0)
```

Storing this data to a csv file for tableau analysis

In [11]:

```
clean_patterns_core.to_csv('clean_patterns_core.csv')
```

Main dataframe

In [12]:

```
clean_patterns_core.T
```

Out[12]:

	1	2	
safegraph_place_id	sg:118c95bf3c4e4370920d7a026fd3ebbe	sg:d9f425fc18a24414b05a053442d1e408	sg:de408
location_name	Royal Ground Coffee	Chestnut Street Coffee Roastery	
street_address	5301 geary boulevard	2331 chestnut street	
zip_code	94121	94123	
brands	NaN	NaN	
raw_visit_counts	50	129	
raw_visitor_counts	23	110	
visits_by_day	[1,2,1,3,1,0,2,1,3,0,1,0,5,2,1,0,0,1,2,1,0,1,1...	[2,0,3,5,3,5,5,7,6,7,4,1,3,4,4,3,9,1,1,3,6,7,2...	[7,5,5,13,4,3
distance_from_home	2082	3687	

median_dwell	683.5	22	
bucketed_dwell_times	{"<5":0,"5-20":8,"21-60":8,"61-240":2,">240":32}	{"<5":6,"5-20":52,"21-60":38,"61-240":27,">240...	{"<5":2,"5-20":6
popularity_by_hour	[33,33,32,32,33,32,32,30,18,7,6,4,3,1,0,0,0,7,...	[4,4,4,4,4,7,8,11,11,18,16,14,16,19,23,22,11...	[36,35,35,37,37,
popularity_by_day	{"Monday":8,"Tuesday":9,"Wednesday":8,"Thursda...	{"Monday":5,"Tuesday":18,"Wednesday":20,"Thurs...	{"Monday":19,"Tues
open_hours	{ "Mon": [["6:30", "22:00"]], "Tue": [["6:30",...	NaN	{ "Mon": [['

14 rows × 534 columns

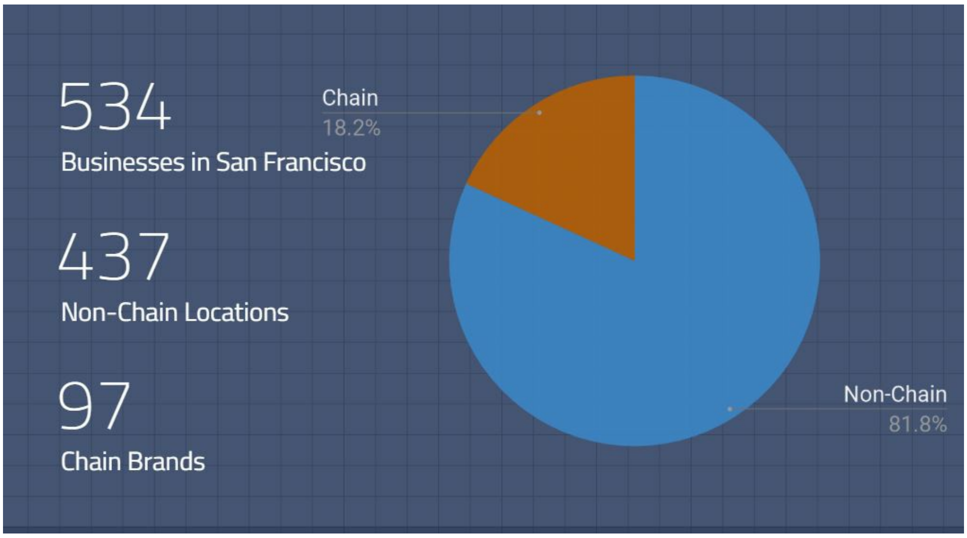
Some basic analysis of out data

There are total 534 cafes in San Francisco out of which 18.2% are chain cafes such as Starbucks, Jamba, etc and rest of them are non-chain cafes.

In [13]:

```
Image(filename='Basic_analysis.JPG', width=600)
```

Out[13]:

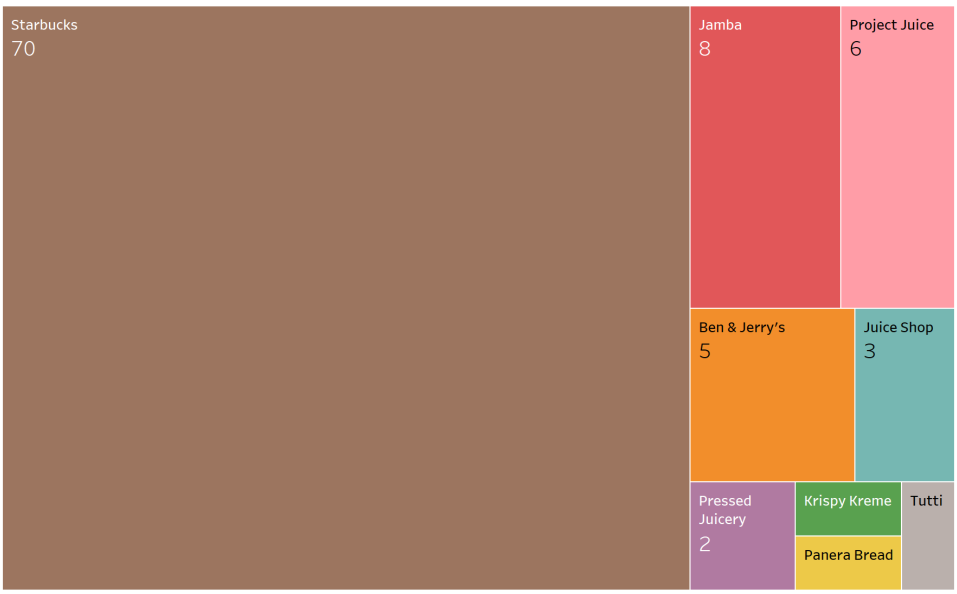


Further analyzing the chain cafes, we see that approx. 70% of them are Starbucks.

In [14]:

```
Image(filename='Brand_Distribution.png', width=600)
```

Out[14]:

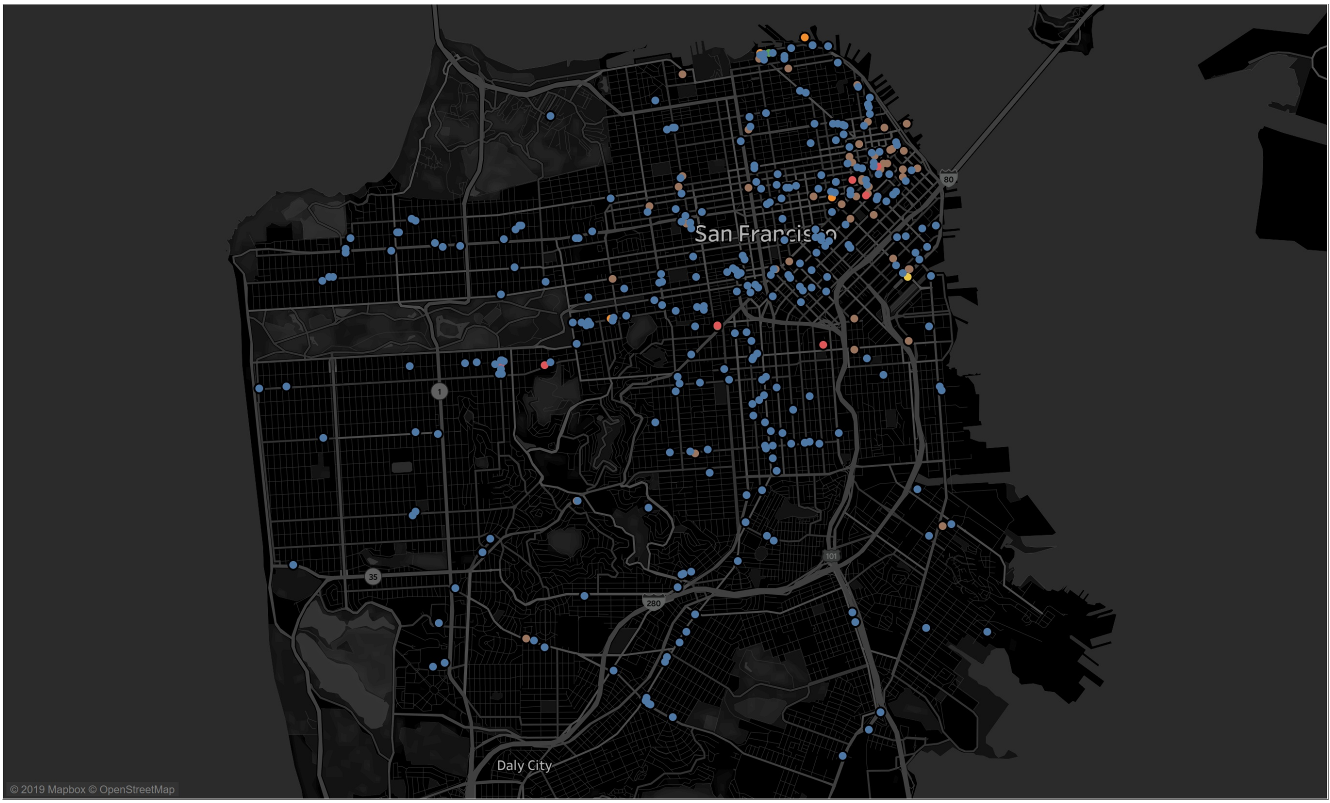


Plotting the geographic extent of our data (All cafes) on the map using Tableau and the shapefiles included with the dataset and seen later in the analysis.

In [15]:

```
Image(filename='geographic_extent.png', width=1000)
```

Out[15]:

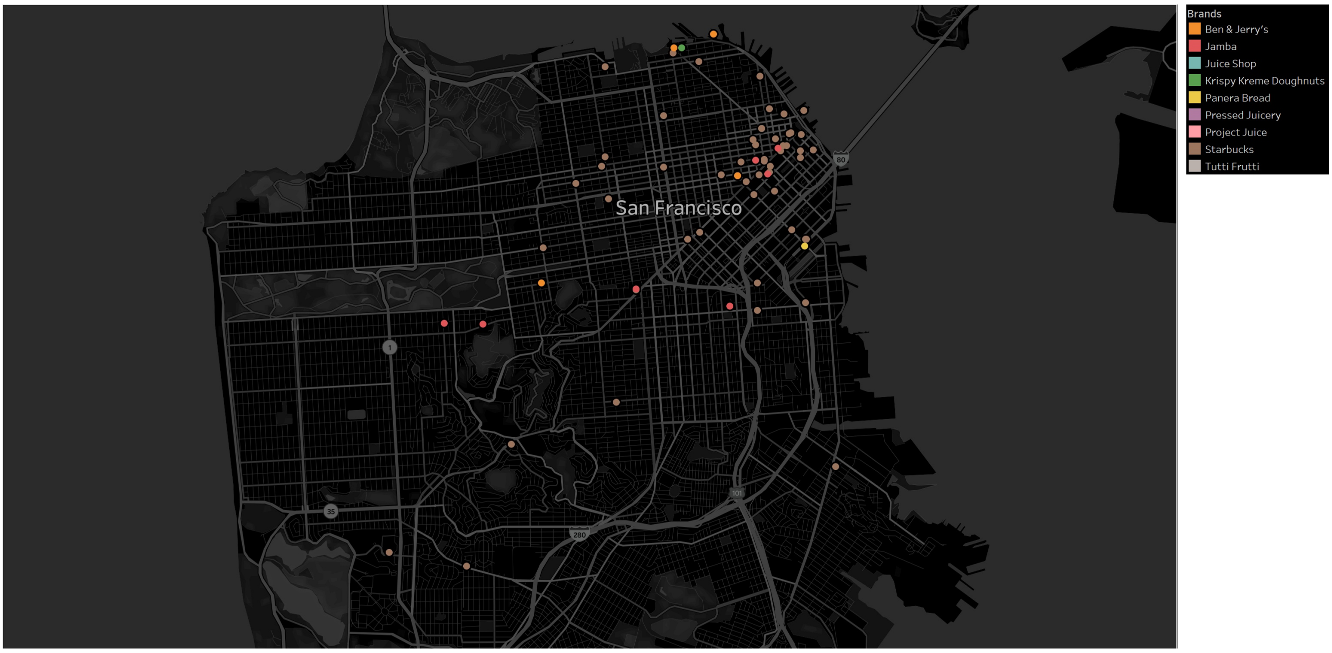


Plotting only the chain brands on the map to analyze how they're located throughtout the city. We found out that 90% of the chain stores are located in the small radius around the downtown district.

In [16]:

```
Image(filename='geographic_extent_only_chain.png', width=1000)
```

Out[16]:



Popularity by hour

The data set includes data for each business, representing the foot-traffic in that business during each hour of the day. This data is stored in list objects with 24 values - one for each hour of the day. Extracting this and cleaning it to only show information for the hours that the business shows to be open.

In [17]:

```
popularity_hour = clean_patterns_core[['safegraph_place_id', 'location_name', 'brands',
'popularity_by_hour', 'open_hours']]
```

Checking for % of null values

In [18]:

```
len(popularity_hour[popularity_hour.open_hours.isna()])/len(popularity_hour)
```

Out[18]:

```
0.2846441947565543
```

Only considering the not null values

In [19]:

```
popularity_hour = popularity_hour[pd.notna(popularity_hour['open_hours'])]
```

Creating functions for extracting the start and end time

In [20]:

```
def clean_starttime(x):
    if x:
        return int(x[0][0].split(':')[0])
    else:
        return float('NaN')
```

In [21]:

```
def clean_endtime(x):
    if x:
        return int(x[0][1].split(':')[0])
    else:
        return float('NaN')
```

For the sake of analysis, Wednesday will be used to represent a typical day. The open hours for Wednesday will be used for visualization purposes.

In [22]:

```
popularity_hour['start_time']=popularity_hour.open_hours.apply(lambda x :
clean_starttime(json.loads(x)['Wed']))
popularity_hour['end_time']=popularity_hour.open_hours.apply(lambda x : clean_endtime(json.loads(x)
['Wed']))
```

In [23]:

```
popularity_hour = popularity_hour[popularity_hour.start_time.notna()]
```

In [24]:

```
popularity_hour.head().T
```

Out[24]:

	1	3	
safegraph_place_id	sg:118c95bf3c4e4370920d7a026fd3ebbe	sg:de408f31b2cd4eb783cc5de4ad6ed651	sg:a8248377476f4f40a
location_name	Royal Ground Coffee	Wholesome Bakery	
brands	NaN	NaN	
popularity_by_hour	[33,33,32,32,33,32,32,30,18,7,6,4,3,1,0,0,0,7,...	[36,35,35,37,37,37,42,47,46,38,49,53,56,52,38,...	[29,30,32,32,32,25,25,29,27,31
open_hours	{ "Mon": [["6:30", "22:00"]], "Tue": [["6:30",...	{ "Mon": [["7:00", "19:00"]], "Tue": [["7:00",...	{ "Mon": [["5:30", "20:00"
start_time	6	7	
end_time	22	19	

A plot showing the number of business opening at each hour of the day.

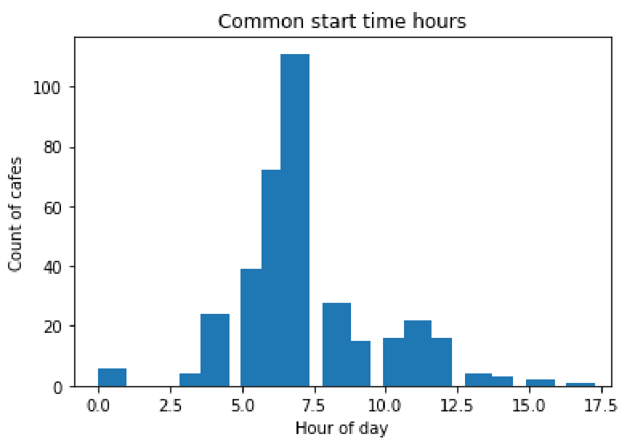
In [76]:

```
ph = popularity_hour.start_time.plot.hist(bins=24, width=1)
ph.set_xlabel('Hour of day')
ph.set_ylabel('Count of cafes')
ph.set_title('Common start time hours')
```

Out[76]:

```
Text(0.5, 1.0, 'Common start time hours')
```

Out[76]:



A similar plot done on tableau to find the number of cafes opening at each hour of the day. Most common time for cafes to open is 6:00-8:00 AM

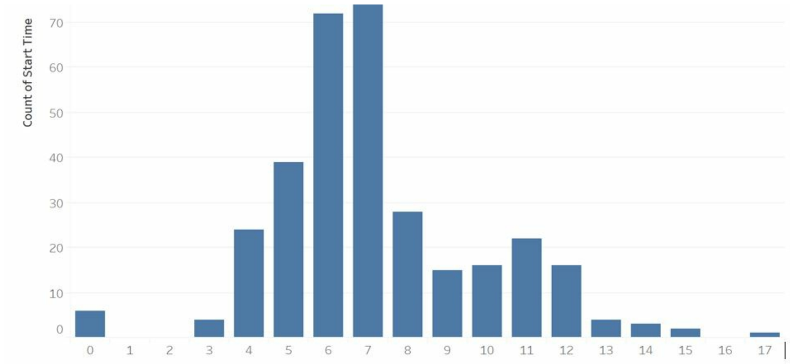
In [26]:

```
popularity_hour.to_csv('start_end_time.csv')
```

In [27]:

```
Image(filename='start_time.JPG', width=500)
```

Out[27]:

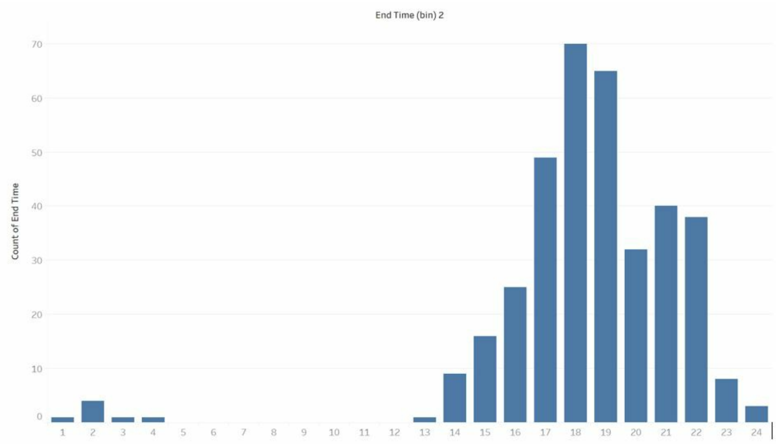


A plot done on tableau to find the number of cafes closing at each hour of the day. Most common time for cafes to close is 6:00-8:00 pm.

In [28]:

```
Image(filename='end_time.JPG', width=500)
```

Out[28]:



Now we find the average popularity of each cafe throughout the day and then plot it using mapbox and javascript

In [29]:

```
popularity_hour_list = popularity_hour.values.tolist()
```

In [31]:

```
for x1 in popularity_hour_list:
    x1[3] = [int(x1[3].strip('][').split(",")[i]) if (i+1) > x1[5] and (i+1) < x1[6] else 0 for i in
range(len(x1[3].split(",")))]
```

In [32]:

```
popularity_hour_dataframe = pd.DataFrame(popularity_hour_list)
```

In [34]:

```
popularity_hour_dataframe = popularity_hour_dataframe.drop(columns=[4,5,6])
```

In [35]:

```
popularity_hour_dataframe.head()
```

Out[35]:

	0	1	2	3
0	sg:118c95bf3c4e4370920d7a026fd3ebbe	Royal Ground Coffee	NaN	[0, 0, 0, 0, 0, 0, 32, 30, 18, 7, 6, 4, 3, 1, ...
1	sg:de408f31b2cd4eb783cc5de4ad6ed651	Wholesome Bakery	NaN	[0, 0, 0, 0, 0, 0, 0, 47, 46, 38, 49, 53, 56, ...
2	sg:a8248377476f4f40ac9eeaef65279570	Happy Donuts	NaN	[0, 0, 0, 0, 0, 25, 25, 29, 27, 31, 34, 34, 41...
3	sg:58be0fc40bc54869aad901c0a58458f7	Sightglass Coffee	NaN	[0, 0, 0, 0, 0, 0, 0, 9, 15, 17, 14, 17, 12, 1...
4	sg:521f82bd4a1c40d4b672f0fb9330d052	Muddy Waters Coffee House	NaN	[0, 0, 0, 0, 0, 0, 0, 0, 6, 5, 11, 19, 23, 27,...

In [36]:

```
popularity_hour_dataframe = popularity_hour_dataframe.rename(columns={0: "safegraph_", 1: "Name", 2
:"Brands", 3: "Hours"})
```

In [37]:

```
for i in range(24):
    popularity_hour_dataframe[str(i+1)] = popularity_hour_dataframe['Hours'].apply(lambda x:x[i])
```

In [38]:

```
popularity_hour_dataframe = popularity_hour_dataframe.drop(columns=['Hours'])
```

A simplified version of the traffic data. Now there is a column for each hour of the day with a value for each business.

In [39]:

```
popularity_hour_dataframe.head()
```

Out[39]:

	safegraph_	Name	Brands	1	2	3	4	5	6	7	...	15	16	17	18	19	20	21	22	23	24
0	sg:118c95bf3c4e4370920d7a026fd3ebbe	Royal Ground Coffee	NaN	0	0	0	0	0	0	32	...	0	0	0	7	19	28	26	0	0	0
1	sg:de408f31b2cd4eb783cc5de4ad6ed651	Wholesome Bakery	NaN	0	0	0	0	0	0	0	...	38	31	21	21	0	0	0	0	0	0
2	sg:a8248377476f4f40ac9eeaef65279570	Happy Donuts	NaN	0	0	0	0	0	25	25	...	36	37	30	19	30	0	0	0	0	0
3	sg:58be0fc40bc54869aad901c0a58458f7	Sightglass Coffee	NaN	0	0	0	0	0	0	0	...	15	26	22	15	0	0	0	0	0	0
4	sg:521f82bd4a1c40d4b672f0fb9330d052	Muddy Waters Coffee House	NaN	0	0	0	0	0	0	0	...	25	18	18	13	24	30	24	11	0	0

5 rows × 27 columns

The third and final component of the dataset is a set of shape files, showing the outline of each of the business. Instead of the outlines, a centroid will be more useful, allowing for a bubble map of locations.

In [40]:

```
with fiona.open('Data/CA-722515-GEOMETRY-2019-09-27.shp') as src:
    meta = src.meta
    meta['schema']['geometry'] = 'Point'
    with fiona.open('Data/centroids.shp', 'w', **meta) as dst:
        for f in src:
            centroid = shape(f['geometry']).centroid
            f['geometry'] = mapping(centroid)
            dst.write(f)
```

In [41]:

```
centroids= gpd.read_file('Data/centroids.shp')
```

The centroid values are merged with the hourly traffic data, producing a product nearly ready to map.

In [43]:

```
centroid_hours = centroids.merge(popularity_hour_dataframe, on='safegraph_')
```

A `geojson` file will serve well for mapping purposes

In [45]:

```
centroid_hours.to_file("centroid_hours3.geojson", driver='GeoJSON')
```

A live version of an animated visualization can be seen at the below link.

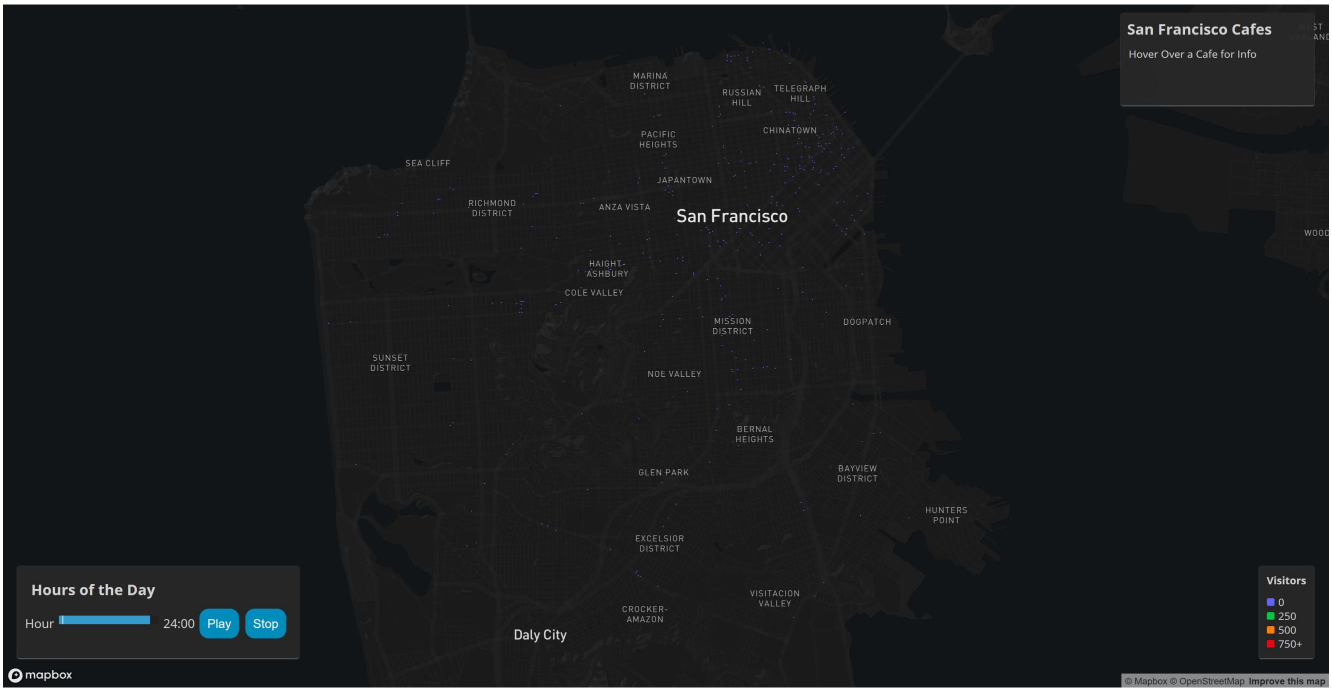
http://htmlpreview.github.io/?https://github.com/darshitpd/San-Francisco-Cafe-Data-Analysis/blob/master/Web-animations/san_fran_cafe_hourly1.html

This screen capture shows the location of the cafes with the small blue dots. The hour represented is midnight, so almost all cafes are closed showing no business.

In [46]:

```
Image(filename='Midnight.png', width=1000)
```

Out[46]:

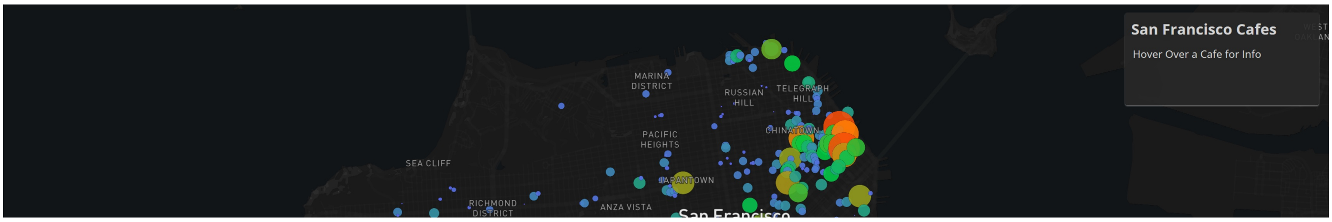


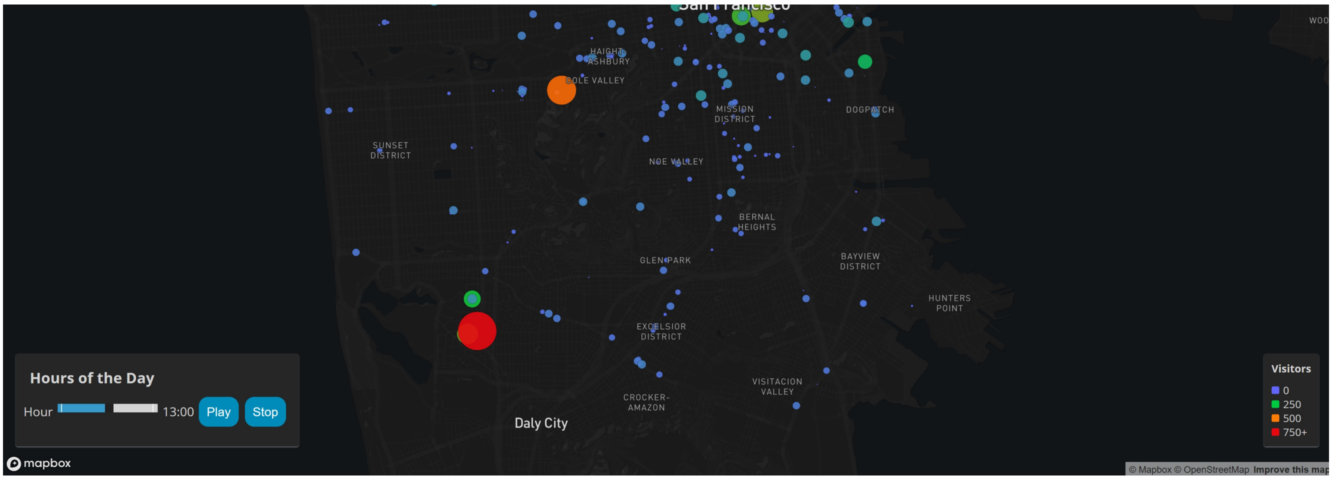
In contrast, this screen capture represents mid-day. Now nearly all cafes are open and are colored and sized relative to their number of visitors.

In [47]:

```
Image(filename='Midday.png', width=1000)
```

Out[47]:





Analyzing the traffic near the cafes when they're closed

In [48]:

```
popularity_hour.head()
```

Out[48]:

	safegraph_place_id	location_name	brands	popularity_by_hour	open_hours	start_time
1	sg:118c95bf3c4e4370920d7a026fd3ebbe	Royal Ground Coffee	NaN	[33,33,32,32,33,32,32,30,18,7,6,4,3,1,0,0,0,7,...	{ "Mon": [["6:30", "22:00"]], "Tue": [["6:30",...	6.0
3	sg:de408f31b2cd4eb783cc5de4ad6ed651	Wholesome Bakery	NaN	[36,35,35,37,37,37,42,47,46,38,49,53,56,52,38,...	{ "Mon": [["7:00", "19:00"]], "Tue": [["7:00",...	7.0
5	sg:a8248377476f4f40ac9eeaef65279570	Happy Donuts	NaN	[29,30,32,32,32,25,25,29,27,31,34,34,41,42,36,...	{ "Mon": [["5:30", "20:00"]], "Tue": [["5:30",...	5.0
6	sg:58be0fc40bc54869aad901c0a58458f7	Sightglass Coffee	NaN	[16,15,9,7,7,7,10,9,15,17,14,17,12,13,15,26,22...	{ "Mon": [["7:00", "19:00"]], "Tue": [["7:00",...	7.0
7	sg:521f82bd4a1c40d4b672f0fb9330d052	Muddy Waters Coffee House	NaN	[3,6,2,2,0,0,0,0,6,5,11,19,23,27,25,18,18,13,2...	{ "Mon": [["6:00", "23:00"]], "Tue": [["6:00",...	6.0

In [49]:

```
popularity_hour_closed_list = popularity_hour.values.tolist()
```

In [51]:

```
for x1 in popularity_hour_closed_list:
    x1[3] = [int(x1[3].strip('][').split(",")[i]) if (i+1) < x1[5] or (i+1) > x1[6] else 0 for i in
range(len(x1[3].split(",")))]
```

In [52]:

```
popularity_hour_closed_dataframe = pd.DataFrame(popularity_hour_closed_list)
```

In [53]:

```
popularity_hour_closed_dataframe = popularity_hour_closed_dataframe.drop(columns=[4,5,6])
```

In [54]:

```
popularity_hour_closed_dataframe.head()
```

Out[54]:

	0	1	2	3
0	sg:118c95bf3c4e4370920d7a026fd3ebbe	Royal Ground Coffee	NaN	[33, 33, 32, 32, 33, 0, 0, 0, 0, 0, 0, 0, 0, 0...
1	sg:de408f31b2cd4eb783cc5de4ad6ed651	Wholesome Bakery	NaN	[36, 35, 35, 37, 37, 37, 0, 0, 0, 0, 0, 0, 0, ...
2	sg:a8248377476f4f40ac9eeaef65279570	Happy Donuts	NaN	[29, 30, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
3	sg:58be0fc40bc54869aad901c0a58458f7	Sightglass Coffee	NaN	[16, 15, 9, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0...
4	sg:521f82bd4a1c40d4b672f0fb9330d052	Muddy Waters Coffee House	NaN	[3, 6, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

In [55]:

```
popularity_hour_closed_dataframe = popularity_hour_closed_dataframe.rename(columns={0:
"safegraph_", 1: "Name", 2: "Brands", 3: "Hours"})
```

In [56]:

```
for i in range(24):
    popularity_hour_closed_dataframe[str(i+1)] = popularity_hour_closed_dataframe['Hours'].apply(la
mbda x:x[i])
```

In [57]:

```
popularity_hour_closed_dataframe = popularity_hour_closed_dataframe.drop(columns=['Hours'])
```

In [58]:

```
popularity_hour_closed_dataframe.head()
```

Out[58]:

	safegraph_	Name	Brands	1	2	3	4	5	6	7	...	15	16	17	18	19	20	21	22	23	24
0	sg:118c95bf3c4e4370920d7a026fd3ebbe	Royal Ground Coffee	NaN	33	33	32	32	33	0	0	...	0	0	0	0	0	0	0	0	37	35
1	sg:de408f31b2cd4eb783cc5de4ad6ed651	Wholesome Bakery	NaN	36	35	35	37	37	37	0	...	0	0	0	0	0	39	39	37	37	39
2	sg:a8248377476f4f40ac9eeaef65279570	Happy Donuts	NaN	29	30	32	32	0	0	0	...	0	0	0	0	0	0	29	27	32	31
3	sg:58be0fc40bc54869aad901c0a58458f7	Sightglass Coffee	NaN	16	15	9	7	7	7	0	...	0	0	0	0	0	43	39	33	32	19
4	sg:521f82bd4a1c40d4b672f0fb9330d052	Muddy Waters Coffee House	NaN	3	6	2	2	0	0	0	...	0	0	0	0	0	0	0	0	0	6

5 rows × 27 columns

In [59]:

```
centroid_closed_hours = centroids.merge(popularity_hour_closed_dataframe, on='safegraph_')
```

In [60]:

```
centroid_closed_hours.to_file("centroid_closed_hours3.geojson", driver='GeoJSON')
```

As we have the open and closed time, and traffic around the cafes both when they're open and

As we have the open and closed time, and traffic around the cafes both when they're open and closed, we can use that to find places which might be possible locations to find a new cafe location.

The data collection method is picking up traffic (represented in red) at most of the locations, even when they are shown to be closed. This suggests that in some fashion there is still foot traffic near these locations, potentially traffic that could be taken advantage of for business purposes.

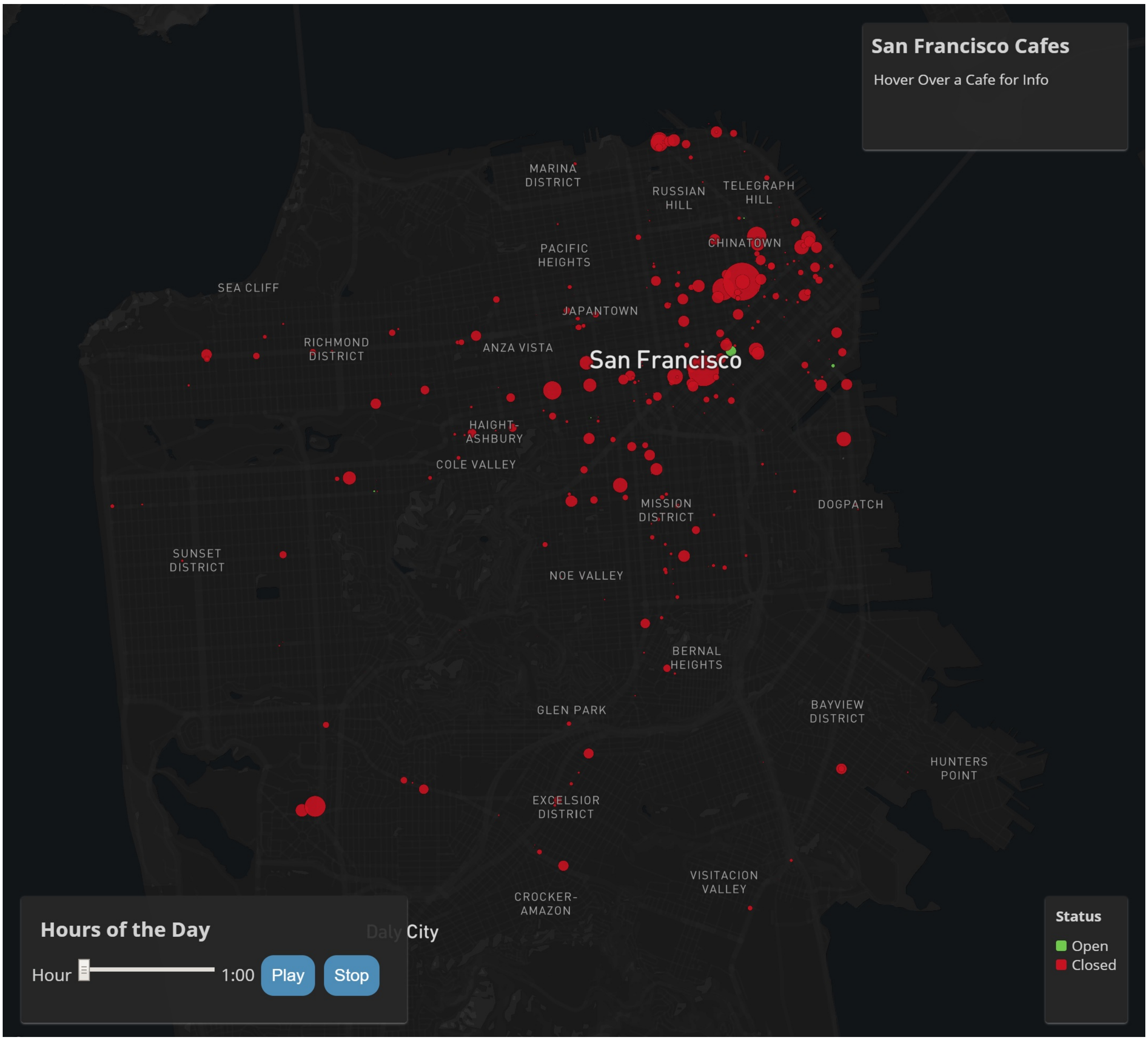
A live version of an animated visualization can be seen at the below link. Detailed exploration of this tool could reveal numerous potential sites.

https://htmlpreview.github.io/?https://github.com/darshitpd/San-Francisco-Cafe-Data-Analysis/blob/master/Web-animations/san_fran_cafe_all.html

In [68]:

```
Image(filename='1am.png', width=800)
```

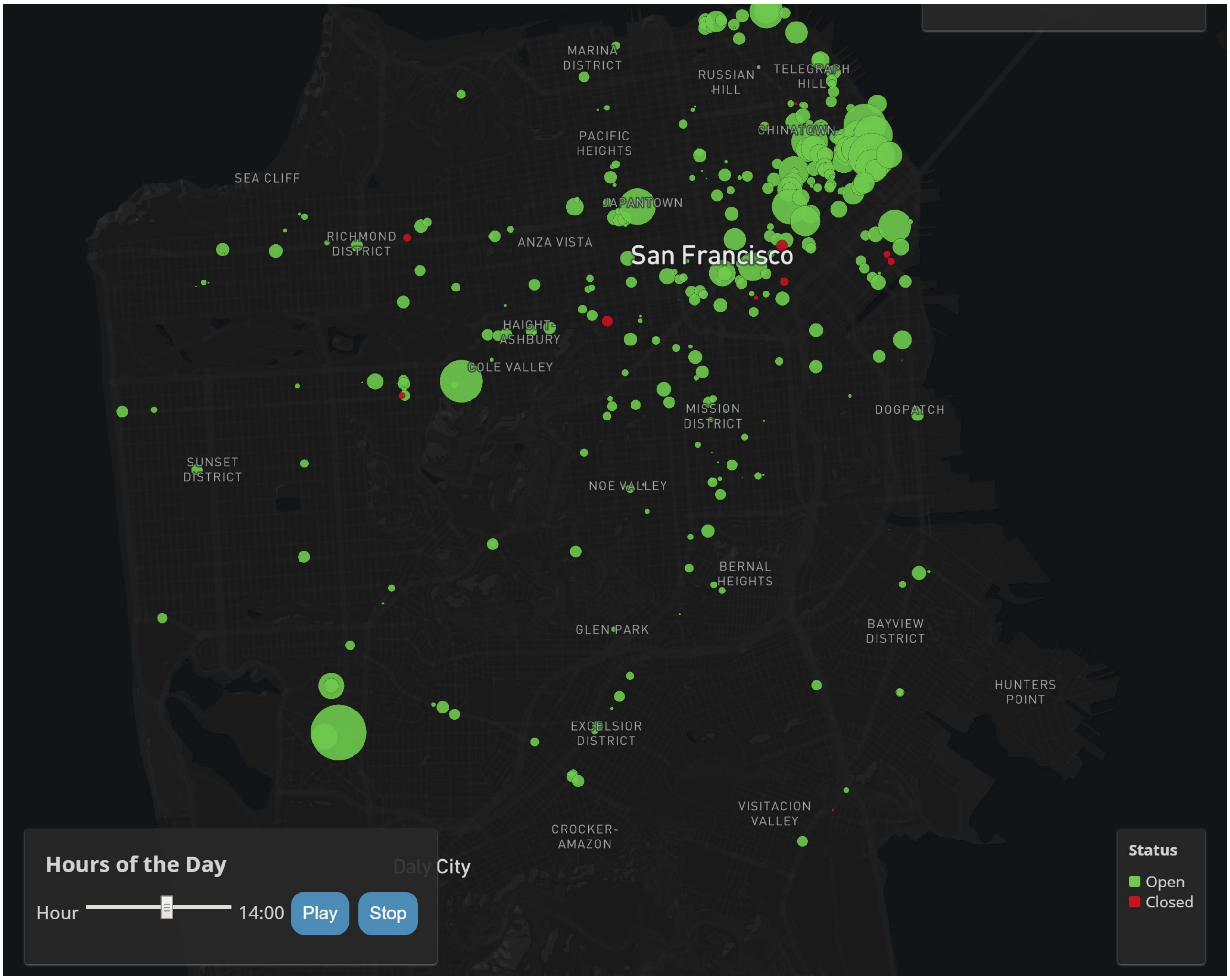
Out[68]:



In [69]:

```
Image(filename='2pm.png', width=800)
```

Out[69]:



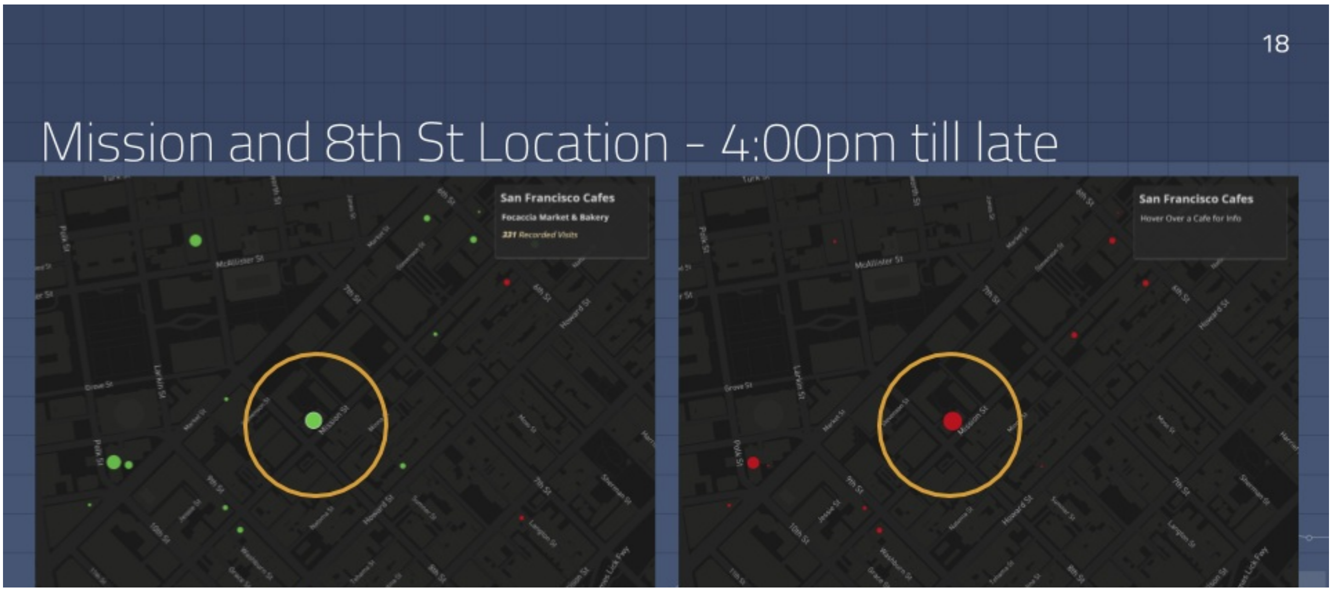
Mission and 8th St Intersection

By viewing the data in this way, it is possible to identify potential locations where traffic is high, even though shops are closed. One particular location that demonstrated this idea is near the Mission and 8th St Intersection. There is a bakery in the area that shows high traffic during all of its open hours, but it closes particularly early - approximately 4pm. Even after it is closed, the location and other surrounding locations show steady rates of human traffic, suggesting the possibility that there is un-tapped potential in this area for further business development, particularly in the later afternoon and evening hours.

In [70]:

```
Image(filename='Copy of Project 1 Final Presentation.png', width=1000)
```

Out[70]:



Further analysis

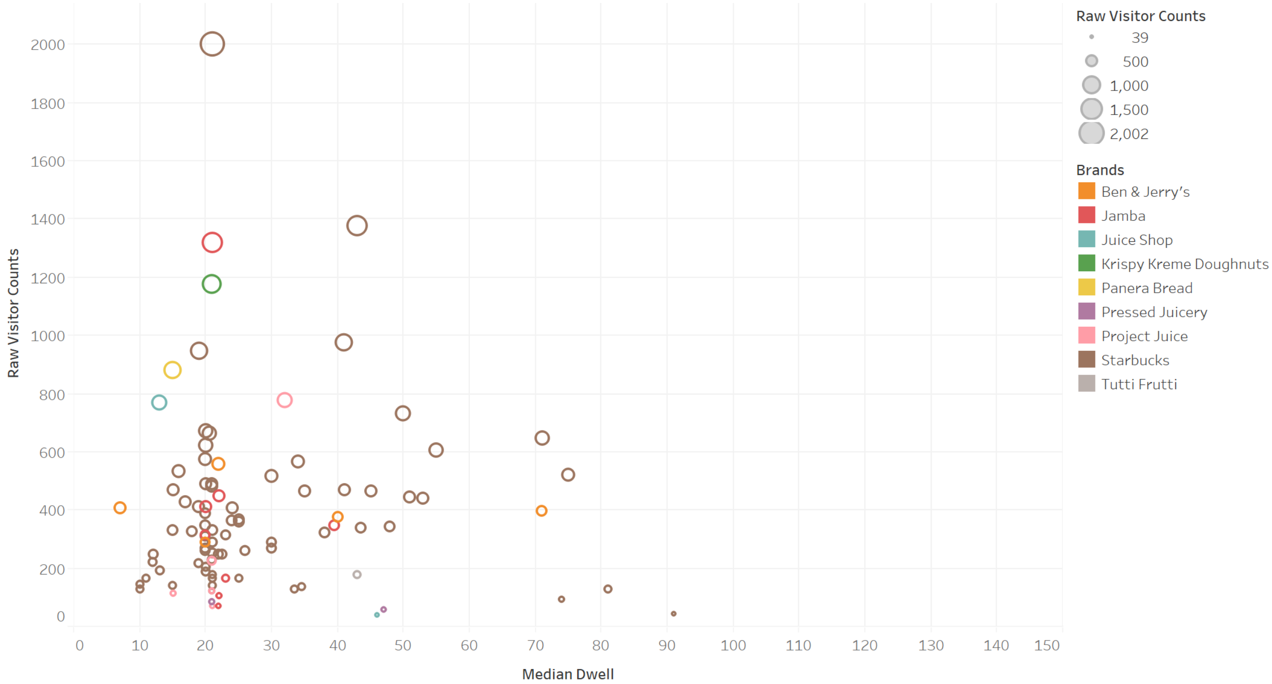
20 Minutes is the most common customer dwell time for chain stores

With all median customer dwell times in chains topping out at 90 minutes. Assuming that chain stores have done extensive market research for profitability, the 20-40 minute time frame appears to be the ideal time for customer dwell.

In [71]:

```
Image(filename='Dwell_Chain.png', width=800)
```

Out[71]:



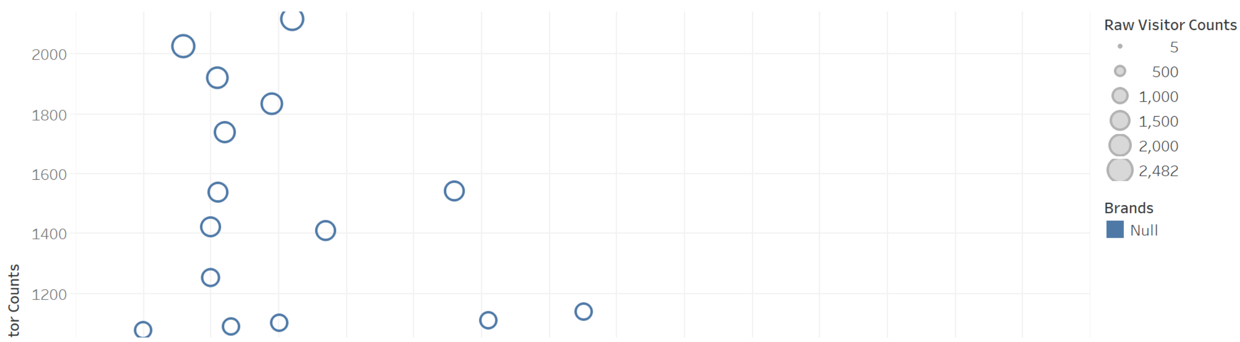
All dwell times longer than 90 minutes are found at non-chain stores

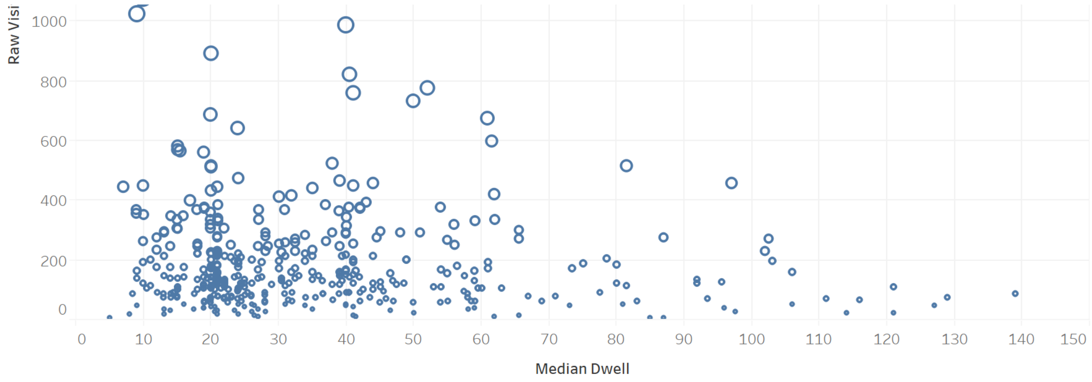
Non-chain stores have clustered dwells at 20 and 40, but also have many instances of longer median dwell times. These long dwell stores also seem to have very small numbers of customers, suggesting that these shops are not thriving to the extent of the stores with faster customer turn-over.

In [77]:

```
Image(filename='Dwell2.png', width=800)
```

Out[77]:





Stategies

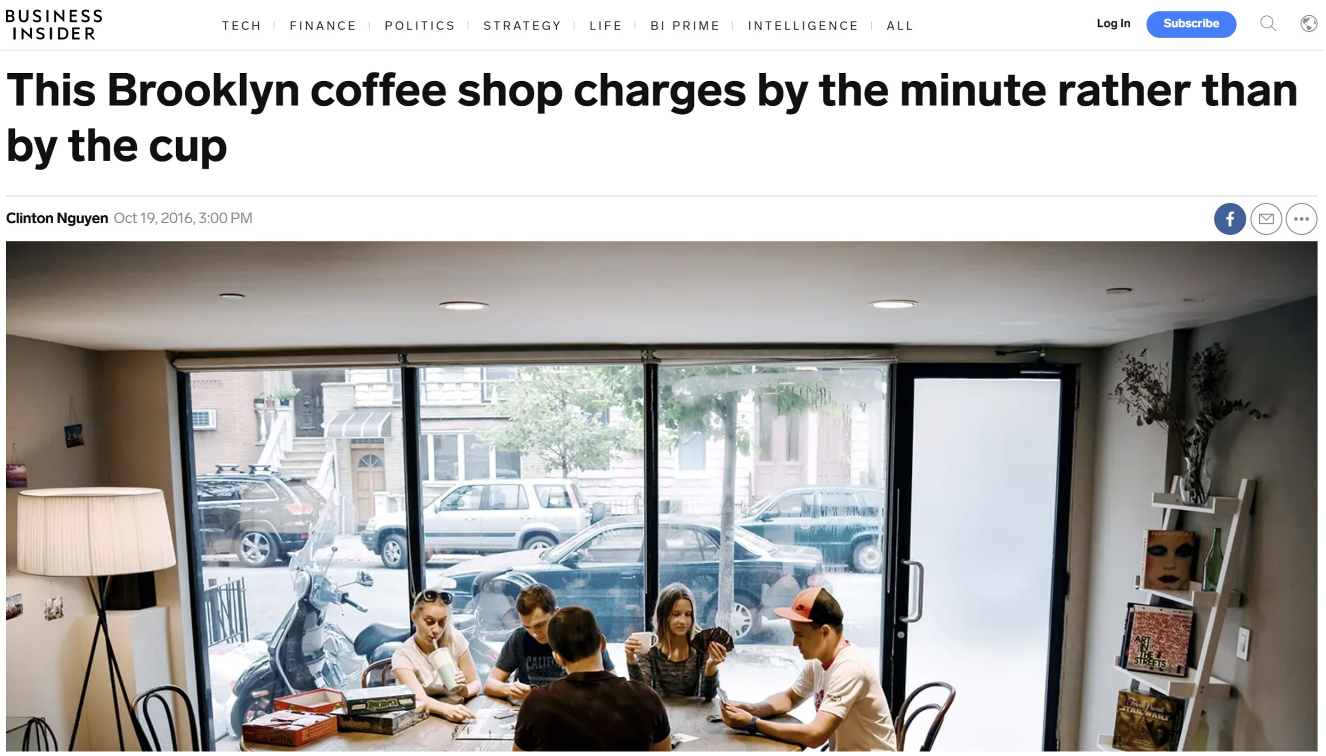
If a shop is committed to the idea of facilitating long customer experiences, one potential approach would be to follow the example of this Brooklyn shop which charges customers not by the item, but by the minute above 1 hour (after paying a $6 base fee.)

https://www.businessinsider.com/coffee-shop-charges-for-time-spent-2016-10

In [73]:

```
Image(filename='Annotation 2019-10-16 143234.png', width=1000)
```

Out[73]:



BUSINESS INSIDER

TECH | FINANCE | POLITICS | STRATEGY | LIFE | BI PRIME | INTELLIGENCE | ALL

Log In Subscribe

This Brooklyn coffee shop charges by the minute rather than by the cup

Clinton Nguyen Oct 19, 2016, 3:00 PM

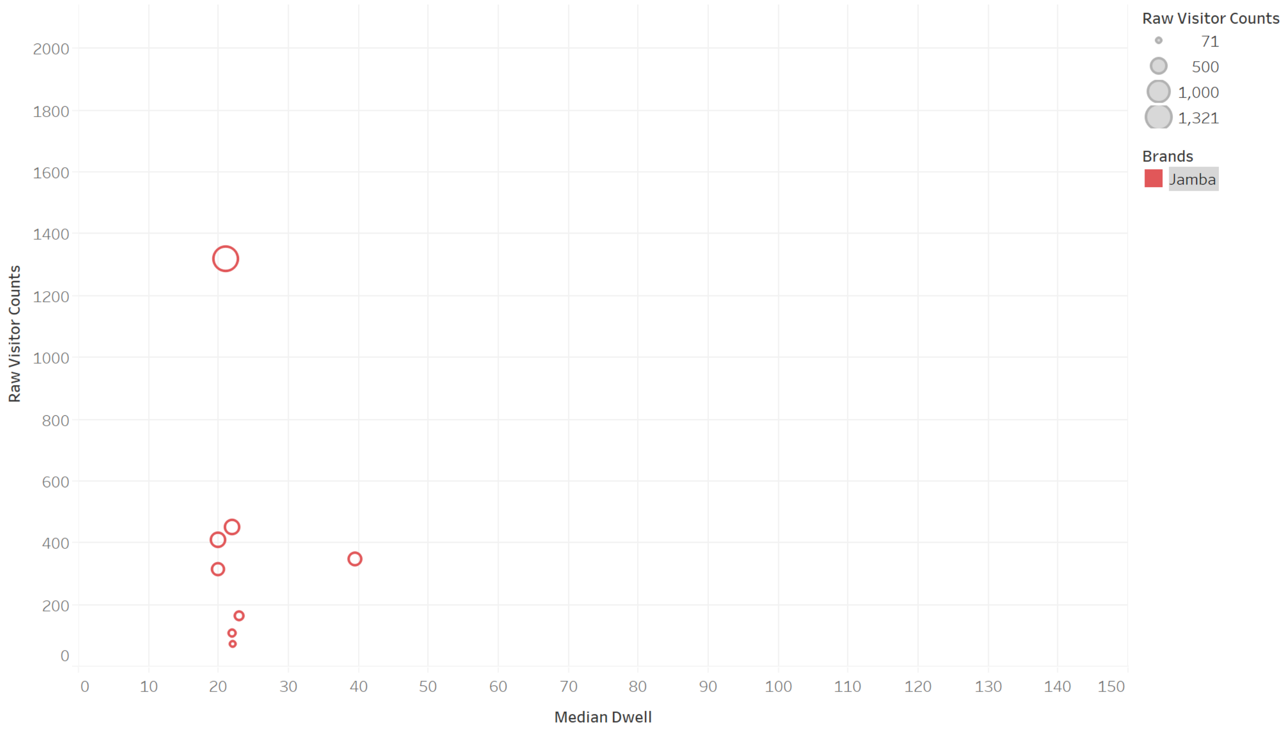
Data Concerns

Although the data is interesting and may provide some business insight, it is necessary to also provide some caution about its use and realiability.

In [74]:

```
Image(filename='Jamba.png', width=1000)
```

Out[74]:



The above chart shows the median customer dwell time for all Jamba Juice stores. There is a high level of consistency clustered at the 20 min mark. Oddly, there is one store which shows double the time. Looking up this store's location reveals something interesting.

In [75]:

```
Image(filename='Annotation 2019-10-15 170550.png', width=1000)
```

Out[75]:

The store location is directly next to an escape room location that advertises 60min customer experiences. If indeed this store is actually giving 20min turn-around for customers, it seems likely that the 40 min times are being corrupted by data from the store next door, mixing with the Jamba location. Perhaps some of the customers shown by GPS to be at the Jamba location are actually at the adjacent business and staying longer. This is just one particularly clear example of many such potential issues in the data.

A SafeGraph employee admitted that there are flaws and reliabiltiy issues with the data

“There is always inherent noise in our visit attribution algorithm, due to jumpiness/errors in GPS data (such as GPS drift, GPS ping scattering in city environments)”

“Our visit data is collected from a panel of mobile GPS devices data. The likely reasons you would see visits outside of business hours are (1) we are picking up visits in the building which are not to the actual POI (for instance a condo over a Starbucks) or (2) we are picking up night workers at a location.”

This leads to questions about all of the data. When the data shows that there are many individuals at a location after regular business hours, are these actually potential customers, or is the collection method merely picking of the presence of people in a nearby appartment building, bus stop, or theater?

Recommended Data Use Case

1. Use current data to identify positive potential store locations/times in high-traffic, low-coverage areas

1. Acquire SafeGraph data for other (non cafe) businesses in identified target areas to better model consumer flow through these neighborhoods

1. Observe best identified sites, on-the-ground, to confirm accuracy/inaccuracy of patterns shown in the data and rule out false positives

Additionally

-Use the data to fine-tune the hours and operations of current businesses to follow best practices of competitors or take advantage of missed opportunities

-Always double-check before making a financial decision based on the data due to the bad signal to noise ratio in SafeGraph’s collection methods

End of Current Analysis

Youtube Link:

https://www.youtube.com/watch?v=hN3JWIhM7eQ&feature=youtu.be

Follow this link for a narrated, visual review of this analysis.