Leveraging U.S. Census Data

I will be doing the following:

1. Decide upon which state to invest in.
2. Decide upon which city after picking state.
3. After choosing city, obtain custom dataset from U.S. Census Website with 2018 Data for population per zip code
4. Sort through the U.S. Census Data to find the top zip codes to analyze

Qualitative Decisions and Assumptions:

- Focus on where people are moving to. Which state are people leaving and to which state are the most people going to?
- The investment firm is a smaller firm looking to expand into a new area.
- The firm will want to have clusters of zip codes nearby for ease of management.
- Firm will not be outsourcing work to other property managers. Work will be done in-house.
- We will not be buying apartment buildings but are open to do so in the future.
- We will look for areas where laws are favorable to landlords as a bonus.
- Since dataset given has data until April 2018, we will use data on or before that date to simulate real time

1. Deciding on which state to invest in:

We first look at U.S. Census Data insights from 2017. This article was published on December 20, 2017:

https://www.census.gov/newsroom/press-releases/2017/estimates-idaho.html#:~:text=DEC.,state%20population%20estimates%20released%20today (https://www.census.gov/newsroom/press-releases/2017/estimates-idaho.html#:~:text=DEC.,state%20population%20estimates%20released%20today)

Where are the people moving to?

Per the article, we have the following:

Top 10 States in Numeric Growth: 2016 to 2017

Rank	Name	2010	2016	2017	Numeric growth
1	Texas	25,146,100	27,904,862	28,304,596	399,734
2	Florida	18,804,594	20,656,589	20,984,400	327,811
3	California	37,254,518	39,296,476	39,536,653	240,177
4	Washington	6,724,545	7,280,934	7,405,743	124,809
5	North Carolina	9,535,721	10,156,689	10,273,419	116,730
6	Georgia	9,688,690	10,313,620	10,429,379	115,759
7	Arizona	6,392,309	6,908,642	7,016,270	107,628
8	Colorado	5,029,325	5,530,105	5,607,154	77,049
9	Tennessee	6,346,295	6,649,404	6,715,984	66,580
10	South Carolina	4,625,381	4,959,822	5,024,369	64,547

Top 10 Most Populous States: 2017

Rank	Name	2010	2016	2017
1	California	37,254,518	39,296,476	39,536,653
2	Texas	25,146,100	27,904,862	28,304,596
3	Florida	18,804,594	20,656,589	20,984,400
4	New York	19,378,110	19,836,286	19,849,399
5	Pennsylvania	12,702,857	12,787,085	12,805,537
6	Illinois	12,831,565	12,835,726	12,802,023
7	Ohio	11,536,730	11,622,554	11,658,609
8	Georgia	9,688,690	10,313,620	10,429,379
9	North Carolina	9,535,721	10,156,689	10,273,419
10	Michigan	9,884,129	9,933,445	9,962,311

Top 10 States in Percentage Growth: 2016 to 2017

Rank	Name	2010	2016	2017	Percent growth
1	Idaho	1,567,650	1,680,026	1,716,943	2.2
2	Nevada	2,700,691	2,939,254	2,998,039	2.0
3	Utah	2,763,889	3,044,321	3,101,833	1.9
4	Washington	6,724,545	7,280,934	7,405,743	1.7
5	Florida	18,804,594	20,656,589	20,984,400	1.6
6	Arizona	6,392,309	6,908,642	7,016,270	1.6
7	Texas	25,146,100	27,904,862	28,304,596	1.4
8	District of Columbia	601,766	684,336	693,972	1.4
9	Colorado	5,029,325	5,530,105	5,607,154	1.4
10	Oregon	3,831,072	4,085,989	4,142,776	1.4

I took this data and color coded which ones have overlap on all 3 lists:

- From this overview, I placed priority on "Most Populous" and "Numerical Growth"
- We see that Texas was present in all 3 columns
- There are close runner-ups for which, if I had more time, I can investigate more into.

Most Populous	Numeric Growth	Percentage Growth
California	Texas	Idaho
Texas	Florida	Nevada
Florida	California	Utah
New York	Washington	Washington
Pennsylvania	North Carolina	Florida
Illinois	Georgia	Arizona
Ohio	Arizona	Texas
Georgia	Colorado	District of Columbia
North Carolina	Tennessee	Colorado
Michigan	South Carolina	Oregon

2. Deciding which city to invest in

Going back to the Zillow dataset, I will filter out with pandas all the cities that exist in Texas and rank them based on the number of zip codes present in each city region.

- Use value_counts() method to rank the top 5 cities
- We will choose the city with the most zip codes to invest.

In [21]:

```
## Import the data

import pandas as pd
df_zillow = pd.read_csv('Data Files/zillow_data.csv')
df_zillow.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14723 entries, 0 to 14722
Columns: 272 entries, RegionID to 2018-04
dtypes: float64(219), int64(49), object(4)
memory usage: 30.6+ MB
```

In [22]:
```python
#Narrow down to Texas only dataframe and do value_count() for cities

df_zillow_1_bool = df_zillow['State'].isin(['TX'])
df_zillow_1 = df_zillow[df_zillow_1_bool]
df_zillow_1
```

Out[22]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996-05	1996-
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235700.0	236900.0	23670
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	212200.0	21220
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77300.0	77300.0	7730
5	91733	77084	Houston	TX	Houston	Harris	6	95000.0	95200.0	9540
8	91940	77449	Katy	TX	Houston	Harris	9	95400.0	95600.0	9580
...	...	...	...	...	...	...	...	...	...	
14372	91640	76941	Mertzon	TX	San Angelo	Irion	14373	NaN	NaN	Na
14472	92897	79313	Anton	TX	Levelland	Hockley	14473	NaN	NaN	Na
14492	92921	79355	Plains	TX	NaN	Yoakum	14493	NaN	NaN	Na
14599	92929	79366	Ransom Canyon	TX	Lubbock	Lubbock	14600	134500.0	134500.0	13440
14695	91948	77457	Matagorda	TX	Bay City	Matagorda	14696	90700.0	91000.0	9120

989 rows × 272 columns

In [23]:
```python
df_zillow_1['City'].value_counts()
```

Out[23]:
```
Houston           86
San Antonio       47
Austin            38
Dallas            33
Fort Worth        26
                  ..
Aldine             1
Weatherford        1
Mount Pleasant     1
Canton             1
Mart               1
Name: City, Length: 540, dtype: int64
```

We see from value_counts() of City that Houston has the most number of Cities. We will move forward with Houston as our choice.

Now that we have decided that Houston is a good fit, we can grab the corresponding population data from the U.S. Census database for all the zip codes in Houston. The steps to do so are as follows:

1. I will first separate only the Houston zip codes from the 2018 dataset that I was given.
2. Make a list of all the zip codes that exist only for Houston
3. Census API can be called from the browser. All we have to do is input in the proper site address
   - Input in the list of zip codes into the Census API
   - This is a workaround to manually choosing all the zip codes.

In [24]:
```python
# Making list of the zip codes in Houston:

list_Houston_zips_bool = df_zillow_1['Metro'] == 'Houston'
list_Houston_zips = df_zillow_1[list_Houston_zips_bool]
list_Houston_zips_1 = list_Houston_zips['RegionName'].tolist()

print(list_Houston_zips_1)
```

```
[77494, 77084, 77449, 77573, 77584, 77429, 77479, 77036, 77433, 77077, 77379, 77459, 7709
5, 77450, 77082, 77057, 77007, 77521, 77083, 77346, 77070, 77375, 77373, 77081, 77063, 77
386, 77042, 77546, 77407, 77072, 77015, 77396, 77008, 77040, 77511, 77089, 77406, 77339,
77088, 77581, 77469, 77539, 77099, 77090, 77388, 77024, 77064, 77004, 77080, 77055, 7749
8, 77044, 77060, 77338, 77096, 77065, 77035, 77471, 77054, 77356, 77092, 77074, 77006, 77
056, 77381, 77520, 77354, 77377, 77571, 77073, 77079, 77365, 77025, 77034, 77093, 77380,
77566, 77382, 77477, 77304, 77550, 77389, 77019, 77075, 77598, 77536, 77489, 77502, 7759
0, 77018, 77532, 77041, 77014, 77027, 77506, 77493, 77087, 77021, 77515, 77301, 77049, 77
586, 77047, 77091, 77017, 77067, 77066, 77005, 77530, 77071, 77016, 77551, 77062, 77045,
77058, 77583, 77357, 77355, 77316, 77098, 77043, 77033, 77345, 77384, 77061, 77504, 7747
8, 77053, 77385, 77020, 77503, 77038, 77505, 77069, 77523, 77086, 77545, 77578, 77039, 77
568, 77531, 77303, 77318, 77302, 77541, 77378, 77030, 77051, 77031, 77028, 77401, 77591,
77048, 77013, 77510, 77554, 77029, 77059, 77002, 77441, 77447, 77474, 77422, 77587, 7703
7, 77085, 77003, 77372, 77563, 77078, 77418, 77518, 77068, 77562, 77032, 77094, 77565, 77
486, 77547, 77665, 77517, 77362, 77514, 77050, 77650, 77534, 77577]
```

In [25]:
```python
#### URL Template used to call the Census API:

####       https://data.census.gov/cedsci/table?q=DP05&t=Age%20and%20Sex&g=8600000US +
#          [ADD ZIP CODES HERE]
#          + &tid=ACSST5Y2019.S0101&hidePreview=false
```

In [26]:
```python
#I added in the zip codes to the template
# https://data.census.gov/cedsci/table?q=DP05&t=Age%20and%20Sex&g=8600000US77002,77003,7700
```

New URL used to call the Census API with zip codes:

https://data.census.gov/cedsci/table?
q=DP05&t=Age%20and%20Sex&g=8600000US77002,77003,77004,77005,77006,77007,77008,77013,77014,77
(https://data.census.gov/cedsci/table?
q=DP05&t=Age%20and%20Sex&g=8600000US77002,77003,77004,77005,77006,77007,77008,77013,77014,77

I have saved the Census data file as "2017_Census_data_86_zipcodes_houston.csv"

Now we sort and do a .head() to see the top 5 zip codes with the highest population

In [27]:
```
df_texas_census = pd.read_csv('Data Files/2017_Census_data_86_zipcodes_houston.csv')
df_texas_census
```

Out[27]:

	GEO_ID	NAME	S0101_C01_001E	S0101_C01_001M	S0101_C01_002E	S0101_C01_002M
0	id	Geographic Area Name	Estimate!!Total!!Total population	Margin of Error!!Total MOE!!Total population	Estimate!!Total!!Total population!!AGE!!Under ...	Margin of Error!!Total MOE!!Total population!!...
1	8600000US77002	ZCTA5 77002	12370	1216	23	30
2	8600000US77003	ZCTA5 77003	9646	717	532	138
3	8600000US77004	ZCTA5 77004	37642	1454	1805	354
4	8600000US77005	ZCTA5 77005	28233	624	2007	273
...	...	...	...	...	...	...
82	8600000US77098	ZCTA5 77098	13444	795	540	179
83	8600000US77099	ZCTA5 77099	51905	2633	4250	539
84	8600000US77339	ZCTA5 77339	41403	1253	1930	421
85	8600000US77345	ZCTA5 77345	29090	834	1430	342
86	8600000US77598	ZCTA5 77598	24689	1188	2139	443

87 rows × 458 columns

Let's clean it up a little. Steps done here:

```
1. Removing all columns I don't need. I only want zip code column and population c
olumn
2. Fixing the data type to int in order to sort properly
3. Sorting the dataframe by population highest to lowest
4. .head() to find the top 5
```

In [28]:
```
df_texas_census.drop(df_texas_census.columns[3:,], axis = 1, inplace = True)
df_texas_census.drop(df_texas_census.columns[0], axis = 1, inplace = True)
# df_texas_census
```

In [29]:
```
df_texas_census = df_texas_census.drop(labels=[0], axis =0)
df_texas_census
```

Out[29]:

	NAME	S0101_C01_001E
1	ZCTA5 77002	12370
2	ZCTA5 77003	9646
3	ZCTA5 77004	37642
4	ZCTA5 77005	28233
5	ZCTA5 77006	21945
...	...	...
82	ZCTA5 77098	13444
83	ZCTA5 77099	51905
84	ZCTA5 77339	41403
85	ZCTA5 77345	29090
86	ZCTA5 77598	24689

86 rows × 2 columns

In [30]:
```
df_texas_census['S0101_C01_001E'] = df_texas_census['S0101_C01_001E'].astype(int)
df_texas_census.sort_values(by=['S0101_C01_001E'], ascending=False, axis=0, inplace=True, i
```

In [31]:
```
df_texas_census.head()
```

Out[31]:

	NAME	S0101_C01_001E
69	ZCTA5 77084	104582
28	ZCTA5 77036	76605
80	ZCTA5 77095	72081
59	ZCTA5 77072	62162
63	ZCTA5 77077	57757

We see the dataset has no null values. Good to go.

In [32]:
```
df_texas_census.isnull().values.any()
```

Out[32]:
```
False
```

There we go. Our top 5 zip codes to analyze are:

1) 77084

2) 77036

3) 77095

4) 77072

5) 77077

** I'll save each zip code into a separate CSV to prep it for time series analysis**

In [33]:
```
# This will make 5 separate .csv files with corresponding zip code names and data pulled fr


zip_list=[77084,77036,77077,77095,77072]
# type(zip_list[3])

for zip in zip_list:
    df = 'df_zillow_' + str(zip) + '.csv'
    df1 = df_zillow_1[df_zillow_1['RegionName'] == zip]
    df1.to_csv(df)
```

Final Steps:

1. Use pd.melt() method to keep only the columns that we want and turn price data from row of values to column of values
2. I create new .csv files which only contain the zip code and the associated value of homes
3. Change column names to "ds" and "y" in order for the dataset to play nice with Facebook Prophet time series analysis
4. Run quick check for any nulls/Nans for my sanity

In [34]:
```
# Prep each zip code by MELTING it using the pd.melt method
# Create two columns "ds" and "y" to make sure the dataframe will work well with the Facebo

list_of_zip_excels = ['df_zillow_77036', 'df_zillow_77077', 'df_zillow_77072', 'df_zillow_7

for excel in list_of_zip_excels:

    excel_df = pd.read_csv(excel + '.csv')
    excel_df = excel_df.drop(labels='Unnamed: 0', axis=1)
    excel_df = pd.melt(excel_df, id_vars=['RegionID', 'RegionName', 'City', 'State', 'Metro

    df_prepped = pd.DataFrame()
    df_prepped['ds'] = excel_df['variable']  # multi-column assignment works for existing c
    df_prepped['y'] = excel_df['value']
    df_prepped.to_csv((str(excel) + '_prepped_fbprophet' + '.csv'),index=False)
```

In [35]:
```
df_zillow_77036_prepped = pd.read_csv('df_zillow_77036_prepped_fbprophet.csv')
df_zillow_77036_prepped.isnull().values.any()
```

Out[35]: False

In [36]:
```
df_zillow_77036_prepped
```

Out[36]:

	ds	y
0	1996-04	120400.0
1	1996-05	118700.0
2	1996-06	117300.0
3	1996-07	116100.0
4	1996-08	115300.0
...	...	...
260	2017-12	177700.0
261	2018-01	177700.0
262	2018-02	179800.0
263	2018-03	185100.0
264	2018-04	189800.0

265 rows × 2 columns

In [37]:
```
df_zillow_77077_prepped = pd.read_csv('df_zillow_77077_prepped_fbprophet.csv')
df_zillow_77077_prepped.isnull().values.any()
```

Out[37]: False

In [38]:
```
df_zillow_77072_prepped = pd.read_csv('df_zillow_77072_prepped_fbprophet.csv')
df_zillow_77072_prepped.isnull().values.any()
```

Out[38]: False

In [39]:
```
df_zillow_77084_prepped = pd.read_csv('df_zillow_77084_prepped_fbprophet.csv')
df_zillow_77084_prepped.isnull().values.any()
```

Out[39]: False

In [40]:
```
df_zillow_77095_prepped = pd.read_csv('df_zillow_77095_prepped_fbprophet.csv')
df_zillow_77095_prepped.isnull().values.any()
```

Out[40]: False