Air BnB Texas Project by Wyatt Salinas

This Air BnB data was downloaded from kaggle: https://www.kaggle.com/PromptCloudHQ/airbnb-property-data-from-texas. I have in interest in real estate data and data mining, so this will code will be tailored towards my interests. Feel free to ask questions if you have any, I am more than happy to talk about it or explain any of this code. This is 100% the code of Wyatt Salinas and no one else.

In [1]:

```
# Will be working from the pandas library. There will be some sections where numpy and statistics will be used.
import pandas as pd
import numpy as np
import statistics
```

In [2]:

```
data = pd.read_csv('Airbnb_Texas_Rentals.csv')
```

Below is the head (first 5 data points) from the data frame that was created from the kaggle data. I will want to change some of the columns of data to analyze the data better.

In [3]:

```
data.head()
```

Out[3]:

	Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title	
0	1	$27	2	Humble	May 2016	Welcome to stay in private room with queen bed...	30.020138	-95.293996	2 Private rooms/bathroom 10min from IAH airport	https://www
1	2	$149	4	San Antonio	November 2010	Stylish, fully remodeled home in upscale NW – ...	29.503068	-98.447688	Unique Location! Alamo Heights - Designer Insp...	https://www
2	3	$59	1	Houston	January 2017	'River house on island close to the city' \nA ...	29.829352	-95.081549	River house near the city	https://www
3	4	$60	1	Bryan	February 2016	Private bedroom in a cute little home situated...	30.637304	-96.337846	Private Room Close to Campus	https://www
4	5	$75	2	Fort Worth	February 2017	Welcome to our original 1920's home. We recent...	32.747097	-97.286434	The Porch	https://www

Prior to starting any analysis, we will want to look into our data and see if there are any big issues. The first issue I will look for is NaN values. Below, I will use some call-outs to find where there are NaN values in the data set.

In [4]:

```
nan_df = data[data.isna().any(axis=1)]
display(nan_df.head())
```

	Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title	
25	26	NaN	2	San Antonio	July 2014	2 bedroom 4 bed chalet/town-home sleeps 6-8 wi...	NaN	NaN	ChaletAzul 2RM/4BD 30day rent Avail	https://
103	104	NaN	1	Mexia	March 2017	Cozy cabin at Lake Mexia has nice lake views, ...	NaN	NaN	Cozy cabin with great lake views!	https://w
104	105	NaN	1	Fort Worth	September 2015	We are located in a quiet neighborhood. Cute h...	NaN	NaN	Room And A Pool 2	https://w
									down to	

105	106	NaN	1	Galveston	September 2016	My place is close to the beach, family-friendl...	NaN	NaN	earth private room (share common areas)	https://w
167	168	NaN	1	Fredericksburg	February 2016	Casita on Creek is a little cottage located on...	NaN	NaN	Casita on Creek	https://w

There aren't many (4) data points that contain NaN values. These will still need to be dropped to ensure that we are able to use accurate data when working with our analysis.

In [5]:
```
data = data.dropna()
```

In [6]:
```
nan_df = data[data.isna().any(axis=1)]
display(nan_df.head())
```

	Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title	url

Above, there are no more data points in our data set that contain NaN as a value.

The next potential issue is the way our data is stored. We will want to make sure that we are using integers or float values for any mathematical analysis that will take place. Let's see what type of data we are working with...

In [7]:
```
data.dtypes
```

Out[7]:
```
Unnamed: 0                  int64
average_rate_per_night     object
bedrooms_count             object
city                       object
date_of_listing            object
description                object
latitude                  float64
longitude                 float64
title                      object
url                        object
dtype: object
```

Average rate per night is an object and I will want this to be an integer without the $ symbol, so that this can be analyzed as a numerical value.

In [8]:
```
data['average_rate_per_night'] = data['average_rate_per_night'].apply(lambda x: x.split('$')[1])
```

In [9]:
```
data['average_rate_per_night'] = data['average_rate_per_night'].astype(int)
```

The average rate per night should now be an int32 instead of the object and we will want to make sure that the dollar sign is now eliminated.

In [10]:
```
data.dtypes
```

Out[10]:
```
Unnamed: 0                  int64
average_rate_per_night      int32
bedrooms_count             object
city                       object
date_of_listing            object
description                object
latitude                  float64
longitude                 float64
title                      object
url                        object
dtype: object
```

In [11]:
```
data.head()
```

Out[11]:

	Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title	
0	1	27	2	Humble	May 2016	Welcome to stay in private room with queen bed...	30.020138	-95.293996	2 Private rooms/bathroom 10min from IAH airport	https://ww
1	2	149	4	San Antonio	November 2010	Stylish, fully remodeled home in upscale NW – ...	29.503068	-98.447688	Unique Location! Alamo Heights - Designer Insp...	https://ww

2	3	59	1	Houston	January 2017	'River house on island close to the city' \nA ...	29.829352	-95.081549	River house near the city	https://ww
3	4	60	1	Bryan	February 2016	Private bedroom in a cute little home situated...	30.637304	-96.337846	Private Room Close to Campus	https://ww
4	5	75	2	Fort Worth	February 2017	Welcome to our original 1920's home. We recent...	32.747097	-97.286434	The Porch	https://ww

The dollar sign has been eliminated from the average rate per night and it is now an integer. We can begin working mathematically on this when needed.

Let's find the mean, min, and max prices of the average rate per night on our data. We can see if anything looks odd and if we need to look into it further.

In [12]:

```
mean_price = data['average_rate_per_night'].mean()
max_price = data['average_rate_per_night'].max()
min_price = data['average_rate_per_night'].min()

print(f'The mean price of the data is {mean_price}')
print(f'The max price of the data is {max_price}')
print(f'The min price of the data is {min_price}')
```

```
The mean price of the data is 211.55975188011197
The max price of the data is 10000
The min price of the data is 10
```

10000 dollars a night and 10 dollars a night is probably unlikely, we can look further into this data. It is possible if it is a mansion or the other could likely be a shared studio...

In [13]:

```
data.loc[data['average_rate_per_night'] == 10000]
```

Out[13]:

	Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title	
2820	2821	10000	6	Conroe	January 2017	11,000 SF gated estate just north of The Woodl...	30.242734	-95.536784	Houston Super Bowl Resort - 5 BR / 8 Bath	https://ww
6733	6734	10000	5	Houston	October 2016	My place is close to Coveniently located near ...	29.703628	-95.307482	Superbowl LI rental w/private pool!	https://ww
15923	15924	10000	4	Rosharon	August 2016	My place is close to restaurants and dining. Y...	29.407731	-95.396449	Secluded getaway w/resort ambience & close to ...	https://ww
16144	16145	10000	5	Houston	April 2015	Huge Mansion Perfect for super bowl. Close to ...	29.786687	-95.398257	HUGE Mansion for Super Bowl, close to everything	https://ww

There are (4) different data points that have the average rate per night as 10000 dollars. They actually seem accurate as I see that each house has a good amount of bedrooms (>4), I see the keyword of mansion a couple of times, I see 11000 square feet, the word huge,etc...

I may look into this further in the future, but as of now I will keep them as is.

In [14]:

```
data.loc[data['average_rate_per_night'] == 10]
```

Out[14]:

	Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title
									Pasture

4588	4589	10	Studio	Devine	October 2013	This listing is for HORSES or ranch friendly l...	29.183682	-98.958212	Boarding West Field - Everfree Shire R...	https://v
8218	8219	10	1	Devine	October 2013	RV / trailer parking is available on either si...	29.184473	-98.956849	RV / Trailer parking stay or large vehicle sto...	https://v
8664	8665	10	Studio	Devine	October 2013	This listing is for HORSES or ranch friendly l...	29.184316	-98.959619	Pasture Boarding East Field - Everfree Shire R...	https://v
8809	8810	10	Studio	Devine	October 2013	This listing is for HORSES or ranch friendly l...	29.184170	-98.959875	Pasture Boarding 6 Acres - Everfree Shire Ranch	https://v
11086	11087	10	1	Terlingua	November 2016	We have 40 acres, my daughter and I live on th...	29.320489	-103.617154	Camping under the stars.	https://v
11611	11612	10	1	Houston	June 2014	Daybed in common area (den area). Quiet area a...	29.724620	-95.477325	Thrifty traveler Galleria Houston	https://v
12145	12146	10	1	Leander	February 2016	My home office is tidy and has good light. You...	30.550058	-97.835372	Air Mattress in My Home Office	https://v
12232	12233	10	Studio	Plainview	December 2014	CITY CODE ENFORCEMENT OFFICER states that airb...	34.184068	-101.745630	SHUT DOWN BY CITY	https:/

These seem quite odd to me as I see that there are multiple descriptions saying these are for horses or City Code Enforcement Officer states etc...

For the sake of not being confident in this data at this present time, I will delete these values. These could be credible, but don't seem to fit the particular data structure that I am looking for.

In [15]:

```
data = data[data['average_rate_per_night'] != 10]
```

In [16]:

```
data.loc[data['average_rate_per_night'] == 10]
```

Out[16]:

Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title	url

In [17]:

```
data.head()
```

Out[17]:

	Unnamed: 0	average_rate_per_night	bedrooms_count	city	date_of_listing	description	latitude	longitude	title	
0	1	27	2	Humble	May 2016	Welcome to stay in private room with queen bed...	30.020138	-95.293996	2 Private rooms/bathroom 10min from IAH airport	https://ww
1	2	149	4	San Antonio	November 2010	Stylish, fully remodeled home in upscale NW – ...	29.503068	-98.447688	Unique Location! Alamo Heights - Designer Insp...	https://ww
2	3	59	1	Houston	January 2017	'River house on island close to the city' \nA ...	29.829352	-95.081549	River house near the city	https://ww

3	4	60	1	Bryan	February 2016	Private bedroom in a cute little home situated...	30.637304	-96.337846	Private Room Close to Campus	https://ww
4	5	75	2	Fort Worth	February 2017	Welcome to our original 1920's home. We recent...	32.747097	-97.286434	The Porch	https://ww

I would like to change date_of_listing from it being merged together as an 'object' to it being split up and then extra columns will be added. There will be a 'Month' column as well as a 'year' column. We could potentially look into sorting by year and price and see if inflation or just cost of living is increasing.

In [18]:

```python
data['Month'] = data['date_of_listing'].apply(lambda x: x.split(' ')[0])
data['Year'] = data['date_of_listing'].apply(lambda x: x.split(' ')[1])
del data['date_of_listing']
del data['Unnamed: 0']
data.head()
```

Out[18]:

	average_rate_per_night	bedrooms_count	city	description	latitude	longitude	title	ur
0	27	2	Humble	Welcome to stay in private room with queen bed...	30.020138	-95.293996	2 Private rooms/bathroom 10min from IAH airport	https://www.airbnb.com/rooms/18520444 location.
1	149	4	San Antonio	Stylish, fully remodeled home in upscale NW – ...	29.503068	-98.447688	Unique Location! Alamo Heights - Designer Insp...	https://www.airbnb.com/rooms/17481455 location.
2	59	1	Houston	'River house on island close to the city' \nA ...	29.829352	-95.081549	River house near the city	https://www.airbnb.com/rooms/16926307 location.
3	60	1	Bryan	Private bedroom in a cute little home situated...	30.637304	-96.337846	Private Room Close to Campus	https://www.airbnb.com/rooms/11839729 location.
4	75	2	Fort Worth	Welcome to our original 1920's home. We recent...	32.747097	-97.286434	The Porch	https://www.airbnb.com/rooms/17325114 location.

Let's now find the average_rate_per_night for each city. We will first need to find the amount of cities that are in this data. If there are many cities, then I might end up filtering data where the city has to have at least a count of 5.

In [19]:

```python
data['city'].unique()
```

Out[19]:

```
array(['Humble', 'San Antonio', 'Houston', 'Bryan', 'Fort Worth',
       'Conroe', 'Cedar Creek', 'Rockport', 'Irving', 'Euless',
       'Round Mountain', 'Kerrville', 'New Braunfels', 'Austin',
       'Port Aransas', 'Frisco', 'Katy', 'College Station', 'Denton',
       'Richmond', 'Plano', 'Arlington', 'Breckenridge', 'Spicewood',
       'Dallas', 'Killeen', 'Abilene', 'Marble Falls', 'Pasadena',
       'Allen', 'Bandera', 'Manor', 'League City', 'Concan',
       'Canyon Lake', 'Spring', 'McKinney', 'La Porte', 'Buchanan Dam',
       'Baytown', 'Gary City', 'Eustace', 'Colleyville', 'Mabank',
       'Cibolo', 'Pipe Creek', 'Haltom City', 'Navasota',
       'South Padre Island', 'Corpus Christi', 'Granbury', 'Round Rock',
       'Sweeny', 'Sealy', 'Shenandoah', 'Brenham', 'Schertz',
       'Carrollton', 'New Ulm', 'Hempstead', 'Fayetteville', 'Montgomery',
       'Cleburne', 'Kemah', 'Bridge City', 'Lago Vista', 'Tow',
       'Bridgeport', 'Kingsville', 'Kyle', 'Converse', 'Lakeway',
       'Galveston', 'Terlingua', 'Bastrop', 'River Oaks', 'Saginaw',
       'Belton', 'Canyon', 'Temple', 'Little Elm', 'Tool', 'Sugar Land',
       'Seabrook', 'Rogers', 'Harker Heights', 'Hickory Creek',
       'Grapevine', 'Buda', 'Canton', 'Celina', 'White Settlement',
```

```
       'Roanoke', 'Horseshoe Bay', 'Edom', 'Paige', 'Llano', 'Hutto',
       'Eddy', 'Nacogdoches', 'Rosenberg', 'Brazoria', 'Lumberton',
       'Crosby', 'Texas City', 'Corinth', 'Leander', 'Amarillo',
       'Dripping Springs', 'Port Isabel', 'Laguna Vista', 'Harlingen',
       'Salado', 'Aubrey', 'Fredericksburg', 'Grand Prairie', 'DeSoto',
       'Elgin', 'Comfort', 'Burnet', 'Argyle', 'Red Oak', 'Del Valle',
       'Tomball', 'Shady Shores', 'Clear Lake Shores', 'El Lago',
       'Clifton', 'Boerne', 'Cedar Park', 'Blanco', 'Cat Spring', 'Waco',
       'Ingram', 'Lakehills', 'Crystal Beach', 'Gordon', 'Brookshire',
       'Bastrop County', 'Fulshear', 'Graham', 'Copper Canyon', 'Rosanky',
       'Bacliff', 'Mineola', 'Alvin', 'Bolivar Peninsula', 'Aransas Pass',
       'Alpine', 'Jasper', 'Benbrook', 'Lewisville', 'Pflugerville',
       'Bleiblerville', 'Nederland', 'Mason', 'Balch Springs',
       'Smithville', 'Bellaire', 'The Woodlands', 'Manchaca', 'Highlands',
       'Watauga', 'Brownwood', 'Bay City', 'Flower Mound', 'Georgetown',
       'Stephenville', 'Kingsland', 'Columbus', 'Pearland', 'Crawford',
       'Travis County', 'Lantana', 'San Saba', 'Wimberley', 'Early',
       'Point Venture', 'Universal City', 'Quitman', 'Pontotoc', 'Aledo',
       'Brownsville', 'Krum', 'Shoreacres', 'Justin', 'Weatherford',
       'Sargent', 'San Leon', 'Haslet', 'Lindale', 'Highland Village',
       'Van', 'Copperas Cove', 'Deer Park', 'Palestine', 'Port Arthur',
       'Wharton', 'Riviera Beach', 'Sanger', 'Taft', 'McDade',
       'City-by-the Sea', 'Willis', 'Fort Bend County', 'Damon',
       'Star Harbor', 'Azle', 'City by the Sea', 'Nolanville', 'Kemp',
       'Matagorda', 'Groves', 'Hurst', 'Enchanted Oaks', 'Fredonia',
       'Flint', 'Santa Fe', 'Murchison', 'Anahuac', 'Bedford', 'Fulton',
       'Webster', 'Bee Cave', 'Jamaica Beach', 'Streetman', 'Bayside',
       'Richland Hills', 'North Richland Hills', 'Cottonwood Shores',
       'China', 'Rio Frio', 'Dale', 'Lockhart', 'Dayton',
       'Fair Oaks Ranch', 'Palacios', 'Garfield', 'Malakoff', 'Beaumont',
       'Ennis', 'Cedar Hill', 'Gun Barrel City', 'Taylor', 'Kerens',
       'Coppell', 'Teague', 'Waxahachie', 'Fairfield', 'Cross Roads',
       'Midlothian', 'Corsicana', 'Lancaster', 'Forney',
       'North Padre Island Corpus Christi', 'Mexia', 'Lampasas',
       'Kaufman', 'DESOTO', 'Groesbeck', 'Trinidad', 'Crandall',
       'Live Oak', 'houston', 'Forreston', 'Lake Dallas', 'Athens',
       'Hillsboro', 'Porter', 'Mertens', 'Glenn Heights', 'Selma', 'West',
       'New Waverly', 'Marion', 'Kirby', 'Ingleside', 'Dawson',
       'Wills Point', 'Livingston', 'Duncanville', 'Melissa', 'Round Top',
       'Hockley', 'Princeton', 'Glen Rose', 'Coleman', 'Hunt',
       'Seagoville', 'Center', 'Comanche', 'Burleson', 'Crowley',
       'Carmine', 'Somerville', 'Seguin', 'Rockdale', 'Burton',
       'Montgomery County', 'Hondo', 'Cleveland', 'Whitney',
       'Ben Wheeler', 'Gatesville', 'Sabinal', 'Florence', 'McGregor',
       'Atascocita', 'Ellinger', 'Woodway', 'Caddo', 'Meridian', 'Baird',
       'Weimar', 'Huffman', 'Kingwood', 'Oak Point', 'Coldspring',
       'Eastland', 'Caldwell', 'Garden Ridge', 'Alamo Heights',
       'Grand Saline', 'Liberty Hill', 'McQueeney', 'Morgan', 'Bellville',
       'Broaddus', 'Natalia', 'Milam', 'Blue Mound', 'Clyde',
       'Hallettsville', 'Mansfield', 'Washington', 'Fairview',
       'Eagle Lake', 'Palo Pinto', 'Hearne', 'Leakey', 'Center Point',
       'Goodrich', 'Cisco', 'Graford', 'Spring Branch', 'Keene', 'Joshua',
       'Nemo', 'Cresson', 'Martindale', 'Ovalo', 'Ledbetter', 'Medina',
       'Scurry', 'Kennedale', 'keller', 'Lucas', 'The Colony', 'Waller',
       'Industry', 'Granite Shoals', 'Channelview', 'Helotes',
       'Van Alstyne', 'Chappell Hill', 'New Caney', 'Magnolia',
       'Carthage', 'Dublin', 'Alvarado', 'Castroville',
       'Possum Kingdom Lake', 'Santo', 'Prosper', 'San Augustine',
       'Godley', 'China Spring', 'New ULM', 'South Houston', 'San Marcos',
       'Mountain Home', 'Eliasville', 'Venus', 'Grandview', 'Tarpley',
       'West Lake Hills', 'Plainview', 'Decatur', 'Port Neches',
       'Sunrise Beach Village', 'Millican', 'Anna', 'Savannah',
       'Llano County', 'Bryan-College Station', 'Pilot Point', 'Bertram',
       'Mico', 'Webberville', 'Devine', 'Jonestown', 'Junction',
       'Los Fresnos', 'Menard', 'Hereford', 'Briarcliff', 'Northlake',
       'Bendera', 'Rio Medina', 'Bluffton', 'Johnson City', 'Brady',
       'Eden', 'Jacksboro', 'Rancho Viejo', 'Runaway Bay', 'fort worth',
       'South padre island', 'Kendall County', 'Cherokee', 'Lohn',
       'Blanco County', 'Horseshoe bay', 'San Benito', 'Bulverde',
       'Bowie', 'Happy', 'Terlingua Ranch', 'The Hills', 'Chico',
       'Del Valley', 'Keller', 'Art', 'Springtown', 'New Fairview',
       'Volente', 'Zephyr', 'KYLE', 'Westworth Village', 'Sunset', '诺斯莱克',
       'Addison', 'Shore Acres', 'Frankston', 'Buffalo Gap',
       'Surfside Beach', 'Farmersville', 'Winnsboro', 'Willow Park',
       'Freeport', 'Hudson Oaks', 'East Bernard', 'Red Rock',
       'Trophy Club', 'LaRue', 'Garden Valley', 'Sugarland', 'Gilchrist',
       'Dickinson', 'Bolivar Pennisula', 'Angleton', 'Friendswood',
       'Winnie', 'La Marque', 'City By The Sea', 'Bolivar', 'Orange',
       'Yantis', 'Richardson', 'CRYSTAL BEACH', 'Boliver peninsula',
       'Missouri City', 'Surfside', 'Sunset Valley', '阿纳瓦克',
       'Tiki Island', 'Anson', 'University Park', 'Berryville', 'Emory',
       'Chandler', 'Waelder', 'Silsbee', 'Leonard', 'Southlake',
       'Lake Jackson', 'Alba', 'Whitewright', 'Bullard', 'El Campo',
       'Beach City', 'Manvel', 'Rosharon', 'CIty by the Sea', 'Goliad',
       'Needville', 'Riviera', 'Bells', 'Hawkins', 'Galveston County',
       'Merkel', 'Montalba', 'Port Bolivar', 'Tuscola'], dtype=object)
```

Yes, there are very many cities. I will now use the groupby function to find the count of each city and take the top 10 based on the amount of listings for that given city...

In [20]:

```
data_by_city = data.groupby('city').count()
data_by_city.rename(columns={"url":"City_Count"}, inplace=True)
data_by_city = data_by_city.drop(['average_rate_per_night', 'bedrooms_count','description','latitude', 'longitu
data_by_city.head()
```

Out[20]:

	City_Count
city	
Abilene	106
Addison	22
Alamo Heights	2
Alba	2
Aledo	5

In [21]:

```
data_by_city = data_by_city.sort_values(['City_Count'], ascending = False).reset_index()
data_by_city.head(30)
```

Out[21]:

	city	City_Count
0	Houston	1501
1	Austin	1477
2	San Antonio	1223
3	Dallas	1144
4	Fort Worth	998
5	Corpus Christi	641
6	College Station	480
7	Irving	403
8	Bryan	358
9	New Braunfels	306
10	Galveston	304
11	McKinney	286
12	Arlington	272
13	Denton	252
14	Katy	216
15	South Padre Island	195
16	Port Aransas	187
17	Rockport	184
18	Horseshoe Bay	168
19	Richmond	157
20	Marble Falls	154
21	Plano	152
22	Conroe	151
23	League City	138
24	Canyon Lake	134
25	Wimberley	132
26	Lago Vista	131
27	Spicewood	126
28	Bandera	124
29	Spring	121

We could take the top 10 cities by using data_by_city[0:9], but I want to use as many unique functions as possible. So, I will use a more unique fuction below and will only take cities that have over 304 (not including 304) listings so that the last city will be New Braunfels.

In [22]:

```
data_by_city = data_by_city[data_by_city['City_Count'] > 304]
data_by_city
```

Out[22]:

	city	City_Count

0	Houston	1501
1	Austin	1477
2	San Antonio	1223
3	Dallas	1144
4	Fort Worth	998
5	Corpus Christi	641
6	College Station	480
7	Irving	403
8	Bryan	358
9	New Braunfels	306

I will now use these cities on the orginal data to keep these cities and then get rid of the rest of the data. We are still looking for the average air bnb price of each city that is in the top 10 cities by the listing count.

```
In [23]: city_list = data_by_city['city'].tolist()
         city_list
```

```
Out[23]: ['Houston',
          'Austin',
          'San Antonio',
          'Dallas',
          'Fort Worth',
          'Corpus Christi',
          'College Station',
          'Irving',
          'Bryan',
          'New Braunfels']
```

```
In [24]: top_cities_data = data[data['city'].isin(city_list)]
         top_cities_data
```

Out[24]:

	average_rate_per_night	bedrooms_count	city	description	latitude	longitude	title	
1	149	4	San Antonio	Stylish, fully remodeled home in upscale NW – ...	29.503068	-98.447688	Unique Location! Alamo Heights - Designer Insp...	https://www.airbnb.com/rooms/17481 locat
2	59	1	Houston	'River house on island close to the city' \nA ...	29.829352	-95.081549	River house near the city	https://www.airbnb.com/rooms/16926 locat
3	60	1	Bryan	Private bedroom in a cute little home situated...	30.637304	-96.337846	Private Room Close to Campus	https://www.airbnb.com/rooms/11839 locat
4	75	2	Fort Worth	Welcome to our original 1920's home. We recent...	32.747097	-97.286434	The Porch	https://www.airbnb.com/rooms/17325 locat
7	25	1	Fort Worth	This is a beautiful bedroom with a queen size ...	32.689611	-97.298161	Friendly Private Room in Quiet Neighborhood	https://www.airbnb.com/rooms/18977 locat
...	...	...	...	...	...	...	...	
18254	60	1	Dallas	An entire 1 bedroom 700+sqft condo in the hear...	32.892303	-96.772049	Quiet comfort living in Dallas	https://www.airbnb.com/rooms/1011 locatio
18255	99	2	San Antonio	An inviting 1920's cottage home in a popular u...	29.452893	-98.486756	Midtown Cottage Near Riverwalk	https://www.airbnb.com/rooms/18766 locat
				Amazing 3BHK				

18256	13	1	Dallas	Apartment in a picturesque Commun...	33.001955	-96.777615	Room in Dallas!	https://www.airbnb.com/rooms/18719 locat
18257	65	2	San Antonio	My quaint and cozy home is conveniently locate...	29.450142	-98.505333	Vibrant Spacious Loft!	https://www.airbnb.com/rooms/18179 locat
18258	75	1	San Antonio	Cute brand new, modern, guest house (we call i...	29.425508	-98.466833	Brand New Modern Pod in Historic Dignowity Hill	https://www.airbnb.com/rooms/16318 locat

8531 rows × 10 columns

In [25]:
```
top_cities_data['city'].unique()
```

Out[25]:
```
array(['San Antonio', 'Houston', 'Bryan', 'Fort Worth', 'Irving',
       'New Braunfels', 'Austin', 'College Station', 'Dallas',
       'Corpus Christi'], dtype=object)
```

Wanted to confirm that these are the only 10 cities that are in the new data set. It is confirmed that this was all done correctly. Next, narrow down to the important categories at this moment... average price and city.

In [26]:
```
price_city_data = top_cities_data[['average_rate_per_night', 'city']]
price_city_data
```

Out[26]:

	average_rate_per_night	city
1	149	San Antonio
2	59	Houston
3	60	Bryan
4	75	Fort Worth
7	25	Fort Worth
...	...	...
18254	60	Dallas
18255	99	San Antonio
18256	13	Dallas
18257	65	San Antonio
18258	75	San Antonio

8531 rows × 2 columns

In [27]:
```
average_prices_by_city = price_city_data.groupby(['city'],sort=False)['average_rate_per_night'].mean().reset_i
average_prices_by_city
```

Out[27]:

	city	average_rate_per_night
0	San Antonio	93.530662
1	Houston	232.759494
2	Bryan	281.684358
3	Fort Worth	132.332665
4	Irving	85.702233
5	New Braunfels	299.120915
6	Austin	151.981043
7	College Station	249.664583
8	Dallas	112.897727
9	Corpus Christi	216.062402

In [28]:
```
average_prices_by_city = average_prices_by_city.sort_values(['average_rate_per_night'], ascending=False).reset_
del average_prices_by_city['index']
average_prices_by_city
```

Out[28]:

	city	average_rate_per_night
0	New Braunfels	299.120915
1	Bryan	281.684358
2	College Station	249.664583
3	Houston	232.759494
4	Corpus Christi	216.062402
5	Austin	151.981043
6	Fort Worth	132.332665
7	Dallas	112.897727
8	San Antonio	93.530662
9	Irving	85.702233

It appears that New Braunfels has the highest average_rate_per_night and Irving has the lowest. Remember that each of these cities has over 300 listings, so there is a good amount of data to back up this claim of the average_rate_per_night in Texas.

Let's now look at the average price per night based on the year it was listed. My knowledge of economics would tell me that there should be an increase every year. The housing market tends to increase around 3 percent per year, so my hypothesis would be that the renting market will also increase by 3 percent per year. There are likely many factors that will prove me incorrect, but let's see first.

In [29]:

```
price_year_data = data[['average_rate_per_night', 'Year']]
price_year_data['Year'].astype(int)
price_year_data
```

Out[29]:

	average_rate_per_night	Year
0	27	2016
1	149	2010
2	59	2017
3	60	2016
4	75	2017
...	...	...
18254	60	2013
18255	99	2015
18256	13	2016
18257	65	2016
18258	75	2015

18209 rows × 2 columns

In [30]:

```
price_year_data = price_year_data.groupby(['Year'],sort=False)['average_rate_per_night'].mean().reset_index()
price_year_data
```

Out[30]:

	Year	average_rate_per_night
0	2016	210.976955
1	2010	140.336134
2	2017	287.917314
3	2013	171.864542
4	2015	194.472766
5	2014	186.859238
6	2012	190.262829
7	2011	185.595339
8	2009	133.888889
9	2008	129.000000

In [31]:

```
price_year_data = price_year_data.sort_values(['Year'], ascending=True).reset_index()
del price_year_data['index']
price_year_data
```

Out[31]:

	Year	average_rate_per_night
0	2008	129.000000
1	2009	133.888889
2	2010	140.336134
3	2011	185.595339
4	2012	190.262829
5	2013	171.864542
6	2014	186.859238
7	2015	194.472766
8	2016	210.976955
9	2017	287.917314

In [32]:

```
prices_list = price_year_data['average_rate_per_night'].tolist()

dollar_change = [0]

percent_change = [0]

for i in range(len(prices_list)-1):
    percent_change.append(((prices_list[i+1]/prices_list[i])-1)*100)
    dollar_change.append(prices_list[i+1] - prices_list[i])

print(dollar_change)
print(percent_change)
```

```
[0, 4.888888888888886, 6.447245564892626, 45.25920452926934, 4.667489552618747, -18.39828670300028, 14.99469570
398756, 7.613528320452303, 16.504189567617487, 76.94035810885418]
[0, 3.7898363479758723, 4.815370131455077, 32.25057089211407, 2.514874338005324, -9.669932295551387, 8.72471746
8823018, 4.074472539233565, 8.486632817133955, 36.46860765146709]
```

In [33]:

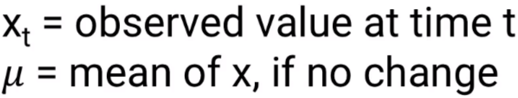
```
price_year_data['Dollar_Change'] = dollar_change
price_year_data['Percent_Change'] = percent_change

price_year_data
```

Out[33]:

	Year	average_rate_per_night	Dollar_Change	Percent_Change
0	2008	129.000000	0.000000	0.000000
1	2009	133.888889	4.888889	3.789836
2	2010	140.336134	6.447246	4.815370
3	2011	185.595339	45.259205	32.250571
4	2012	190.262829	4.667490	2.514874
5	2013	171.864542	-18.398287	-9.669932
6	2014	186.859238	14.994696	8.724717
7	2015	194.472766	7.613528	4.074473
8	2016	210.976955	16.504190	8.486633
9	2017	287.917314	76.940358	36.468608

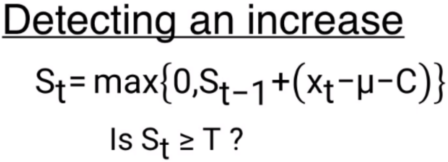
So, it appears that our finidings don't find an exact 3 percent increase per year. There are obviously many factors that are in play where it would be expected to not find a perfect 3 percent increase. Some factors could be: how many houses are available in the market, how many houses are listed at that time on air bnb, all houses for each year would stay exactly the same, all houses are availbe to be rented year-round and the pricing doesn't change (likely air bnb values fluctuate based on location and based on the time of year).

Price does appear to trend upward, we can use CUSUM to see if we can detect change... We can use our Dollar_Change column from the price_year_data dataframe from above to run this experiment.



x_t = observed value at time t

μ = mean of x, if no change



Detecting an increase

$$S_t = \max\{0, S_{t-1} + (x_t - \mu - C)\}$$

Is $S_t \geq T$?

St will be our value that we are tracking. This will change as we go through the data. C will be a number that we will come up with where we can control how quickly or slow change is detected. A large C will result in a slower detection, a smaller C will result in a quicker detection. T is our threshold. When the threshold is detected, we know that change is detected.

In [37]:

```
dollar_change
```

Out[37]:

```
[0,
 4.888888888888886,
 6.447245564892626,
 45.25920452926934,
 4.667489552618747,
 -18.39828670300028,
 14.99469570398756,
 7.613528320452303,
 16.504189567617487,
 76.94035810885418]
```

Because there is not that much data, it won't be the easiest to find a good C value or a good threshold value. But we can eye the data and see that change could be detected around 2011 as there was a 32% increase (45 dollar increase). We can first look at the data with no C to see how it looks.

In [48]:

```
C = 0
St = [0] * len(dollar_change)

dollar_change_mean = statistics.mean(dollar_change)
```

```
for i in range(1, len(dollar_change)):
    St[i] = max(0, St[i-1] + (dollar_change_mean - dollar_change[i] - C))

St.pop(-1)
St
```

Out[48]:

```
[0,
 11.0028424644692,
 20.44732825293466,
 0,
 11.224241800739339,
 45.5142598570977,
 46.41129550646822,
 54.689498539374,
 54.077040325114595]
```

The St increases quickly to 45 and then gradually ends up at 54. Let's try it now with a C value and a threshold value and see where the change is detected. We will use a c value of 12.9 (10% of the smallest value) and a threshold value of 10.

In [59]:

```
C = 12.9
t = 10

years = price_year_data['Year'].tolist()

St = [0] * len(dollar_change)

dollar_change_mean = statistics.mean(dollar_change)


for i in range(1, len(dollar_change)):
    St[i] = max(0, St[i-1] + (dollar_change_mean - dollar_change[i] - C))
    if St[i] > t:
        print(f'The year where change is detected is {years[i]}')
        print(f'The St value is {St[i]} while the detection threshold was {t}')
        break
```

```
The year where change is detected is 2013
The St value is 21.390018056358365 while the detection threshold was 10
```

let's try it with a larger C value and keep the same t value. There is such a large increase, that likely this won't matter. We will use a C value of 25 and the same t value of 20.

In [64]:

```
C = 20
t = 10

years = price_year_data['Year'].tolist()

St = [0] * len(dollar_change)

dollar_change_mean = statistics.mean(dollar_change)


for i in range(1, len(dollar_change)):
    St[i] = max(0, St[i-1] + (dollar_change_mean - dollar_change[i] - C))
    if St[i] > t:
        print(f'The year where change is detected is {years[i]}')
        print(f'The St value is {St[i]} while the detection threshold was {t}')
        break
```

```
The year where change is detected is 2013
The St value is 14.290018056358363 while the detection threshold was 10
```

We can see that the higher the c value the closer it was to that same threshold. This shows that as c gets larger, a larger the change will need to be to be detected. Here is the C value of 0 with the same threshold of 10.

In [66]:

```
C = 0
t = 10

years = price_year_data['Year'].tolist()

St = [0] * len(dollar_change)

dollar_change_mean = statistics.mean(dollar_change)


for i in range(1, len(dollar_change)):
    St[i] = max(0, St[i-1] + (dollar_change_mean - dollar_change[i] - C))
    if St[i] > t:
        print(f'The year where change is detected is {years[i]}')
        print(f'The St value is {St[i]} while the detection threshold was {t}')
        break
```

```
The year where change is detected is 2009
The St value is 11.0028424644692 while the detection threshold was 10
```

With no c value, change is detected immidiately. With a smaller c value, the more sensitive the detection will be.

All in all, we can likely say that there is noticeable change in 2013. With a c value of 20 and a threshold value of 10, it is detected. This is qutie a large c and a small threshold value relative to the data. We can also see that there is a 32 percent increase in price between 2012 and 2013, so it makes sense that we will detect a large change at that point.

This is the end of my analysis. As stated before, this is 100% my work and am open to talking further about this information.