```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        from scipy.stats import ttest_ind
        import statsmodels.api as sm
        from statsmodels.formula.api import ols

        %matplotlib inline
```

```
In [259]: df = pd.read_csv('Desktop/sbnb.csv')
```

```
In [260]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7907 entries, 0 to 7906
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              7895 non-null   object 
 1   name                            7863 non-null   object 
 2   host_id                         7907 non-null   object 
 3   host_name                       7907 non-null   object 
 4   neighbourhood_group             7841 non-null   object 
 5   neighbourhood                   7870 non-null   object 
 6   latitude                        7907 non-null   float64
 7   longitude                       7907 non-null   float64
 8   room_type                       7890 non-null   object 
 9   price                           7907 non-null   int64  
 10  minimum_nights                  7891 non-null   object 
 11  number_of_reviews               7907 non-null   int64  
 12  last_review                     5149 non-null   object 
 13  reviews_per_month               5149 non-null   float64
 14  calculated_host_listings_count  7907 non-null   int64  
 15  availability_365                7521 non-null   float64
dtypes: float64(4), int64(3), object(9)
memory usage: 988.5+ KB
```

```
In [261]: df.isna().sum()
```

```
Out[261]: id                                  12
          name                                44
          host_id                              0
          host_name                            0
          neighbourhood_group                 66
          neighbourhood                       37
          latitude                             0
          longitude                            0
          room_type                           17
          price                                0
          minimum_nights                      16
          number_of_reviews                    0
          last_review                       2758
          reviews_per_month                 2758
          calculated_host_listings_count       0
          availability_365                   386
          dtype: int64
```

1 - Name

Replacing null values in name with NoName

```
In [262]: df.loc[df['name'].isna()]
          df['name'] = df['name'].replace(np.nan, "NoName")
```

2 - ID

Extracting all string like values from column Then Change type to int

```
In [263]: df[pd.to_numeric(df.id, errors='coerce').isnull()]
          df['id']= df['id'].replace(["do it yourself","hello","hotel is full",
                                      "don't know why but food was cool","hello ", np.nan], 0)
          df.id = df['id'].astype(int)
```

3 - Host_ID

Extracts string values and replace with 0 Change type

```
In [265]: #extracts string values
          df[pd.to_numeric(df['host_id'], errors='coerce').isnull()]
          df['host_id'] = df['host_id'].replace(['20th September 2019','31st December 2018', np.nan], 0)
          df['host_id'] = df['host_id'].astype(float)
```

4 - Host_Name

Finding all numeric values in this column and then replaced them with NoName

Find Replace all values which are less than 3 characters with NoName

```
In [266]: df[pd.to_numeric(df.host_name, errors='coerce').notnull()]
          df['host_name'] = df['host_name'].replace(['234', np.nan], 'NoName')
```

```
In [267]: incomplete = df[df['host_name'].str.len()<3]
          df['host_name'] = df['host_name'].replace(['F','Su','Rc','S','Bc','A','Ho','Y','M','El','Lu','Cj',
                                                      'Li','An','Wu','Jj','Yb','Jv','J','Ty','T','Xi','L','Mi',
                                                      'Na','Wl','Yi','Qq','V/','Sk','Xu','Js','Ng','Gy','Bm',
                                                      'Kh','Gc','Mk','W3','W','Bk','Dj','Xx','Yc','Th','Wa','Ke',
                                                      'Ay','Go','Co','Bi','D','Kc','Ts','Jg','Za','Jo','Nt',
                                                      'Cp','Ml','Z','Ow','Le','Vu','Ko','Kk','Dt','Cc','Hz','Ai',
                                                      'Fa','Bo','J','Cd','K.','Ky','Cl','Rh','J.','Ni'], 'NoName')
```

5 - Neighbourhood_Group

There are more than 2k Central Region values hence will replace NaN with this.

I tried to map the values using their neighbourhood using group_by approach but couldn't succeed.

```
In [268]: df['neighbourhood_group'].mode()
          df['neighbourhood_group'] = df['neighbourhood_group'].replace(np.nan, 'Central Region')
```

6 - Neighourhood

```
In [269]: df['neighbourhood'].value_counts()
          df['neighbourhood'] = df['neighbourhood'].replace(np.nan, 'Kallang')
```

7 - Room_Type

Replaced numbers i.e 1,2,3,4 and missing values with mode

```
In [298]: df['room_type'].value_counts()
          df['room_type'] = df['room_type'].replace([np.nan,'1','2','3','4'], 'Entire home/apt')
```

```
Out[298]: Entire home/apt    4147
          Private room       3368
          Shared room         392
          Name: room_type, dtype: int64
```

8 - Price

Finding the normal distribution range of values for price and removing outliers including but not limited to negative numbers

Range taken for price : (35 - 287)

```
In [271]: df['price'].describe()
          df.price.quantile([0.05,0.88])
```

```
Out[271]: 0.05     35.0
          0.88    287.0
          Name: price, dtype: float64
```

```
In [272]: #Removing outliers

          L_bound, U_bound = 0.05,0.88
          Range_Price = df['price'].quantile([L_bound,U_bound])

          Valid  = ((Range_Price.loc[L_bound] < df['price'].values) &(df['price'].values < Range_Price.loc[U_bound]))
          Invalid = ~Valid

          mean = np.mean(df.price[Valid])

          df.price.loc[Invalid] = mean
```

```
/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py:670: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  self._setitem_with_indexer(indexer, value)
```

9 - Minimum Nights

Removed all NaN and string values and changeed Type to int

Removing outliers : I've chosen to set the upper and lower bounds such that i get a range of values b/w 1-180 for minimum_nights. Rest of the values are considered as outliers

```
In [273]: df['minimum_nights'] = pd.to_numeric(df['minimum_nights'], errors='coerce')
          df['minimum_nights'] = df['minimum_nights'].replace(np.nan, 1)
```

```
In [274]: lower_b = 0.05
          upper_b = 0.99

          df['minimum_nights'].quantile([lower_b,upper_b])
```

```
Out[274]: 0.05      1.0
          0.99    180.0
          Name: minimum_nights, dtype: float64
```

```
In [275]: boundary = df['minimum_nights'].quantile([lower_b,upper_b])

          valid_values = ((boundary.loc[lower_b]<df['minimum_nights']) & (df['minimum_nights']<boundary.loc[upper_b]))
          invalid_values = ~valid_values

          Mean_valid = np.mean(df.minimum_nights[valid_values])
          df['minimum_nights'].loc[invalid_values] = Mean_valid
```

10 - Last Review

There are more than 2.5K Null values. I looked for guides to solve this problem. What I found was that if we have some

dates missing in a range of dates e.g

1-1-2010

2-1-2010

3-1-2010

5-1-2010

7-1-2010

As you can see, dates for 4th, 6th Jan are missing. In such cases, the date is in a range and we can fill in the NaT/Missing dates.

However, in this case It would be unwise to simply replace with any value. Hence I will drop this column

```
In [276]: df['last_review'].isna().sum()
          df.drop(columns='last_review', inplace  = True)
```

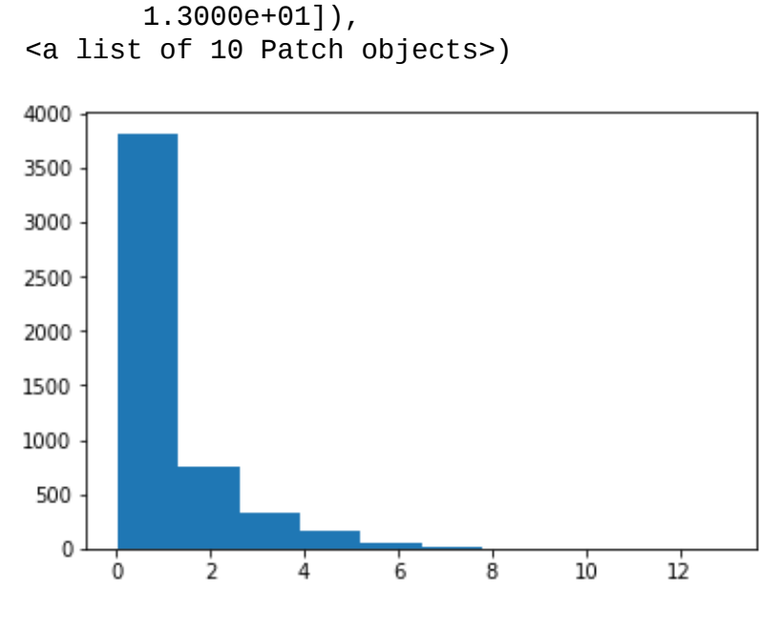
11 - Reviews per month

Based on the distribution, replacing with 1

```
In [277]: plt.hist(df['reviews_per_month'])
```

```
/opt/anaconda3/lib/python3.7/site-packages/numpy/lib/histograms.py:839: RuntimeWarning: invalid value encountered in
greater_equal
  keep = (tmp_a >= first_edge)
/opt/anaconda3/lib/python3.7/site-packages/numpy/lib/histograms.py:840: RuntimeWarning: invalid value encountered in
less_equal
  keep &= (tmp_a <= last_edge)
```

```
Out[277]: (array([3.817e+03, 7.580e+02, 3.350e+02, 1.590e+02, 5.700e+01, 1.400e+01,
                  5.000e+00, 0.000e+00, 1.000e+00, 3.000e+00]),
           array([1.0000e-02, 1.3090e+00, 2.6080e+00, 3.9070e+00, 5.2060e+00,
                  6.5050e+00, 7.8040e+00, 9.1030e+00, 1.0402e+01, 1.1701e+01,
                  1.3000e+01]),
           <a list of 10 Patch objects>)
```



```
In [278]: df['reviews_per_month'] = df['reviews_per_month'].replace(np.nan, 1)
```

12 - Calculated Host Listings

Replacing negative values

```
In [279]: df['calculated_host_listings_count'].value_counts()
          df['calculated_host_listings_count'] = df['calculated_host_listings_count'].replace([-222, -664, -333], 1)
```

13 - Availability_365

Based on the histogram there are alot of 0 values in this column,

I think it's not right to list a rental place which has 0 availability for the entire year.

```
In [280]: #plt.hist(df['availability_365'])

          df['availability_365'].describe()
          df['availability_365'] = df['availability_365'].replace(np.nan, 210)
```

14 - Number of Reviews

Extracting all string values

```
In [281]: #plt.hist(df['number_of_reviews'])
          df[pd.to_numeric(df.number_of_reviews, errors='coerce').isnull()]

          #although there is high number of 0 values, otherwise no strange values found whatsoever
```

Out[281]:

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_pe

New Column for Amount_Paid

```
In [282]: df['amount_paid'] = df['price']*df['minimum_nights']
```

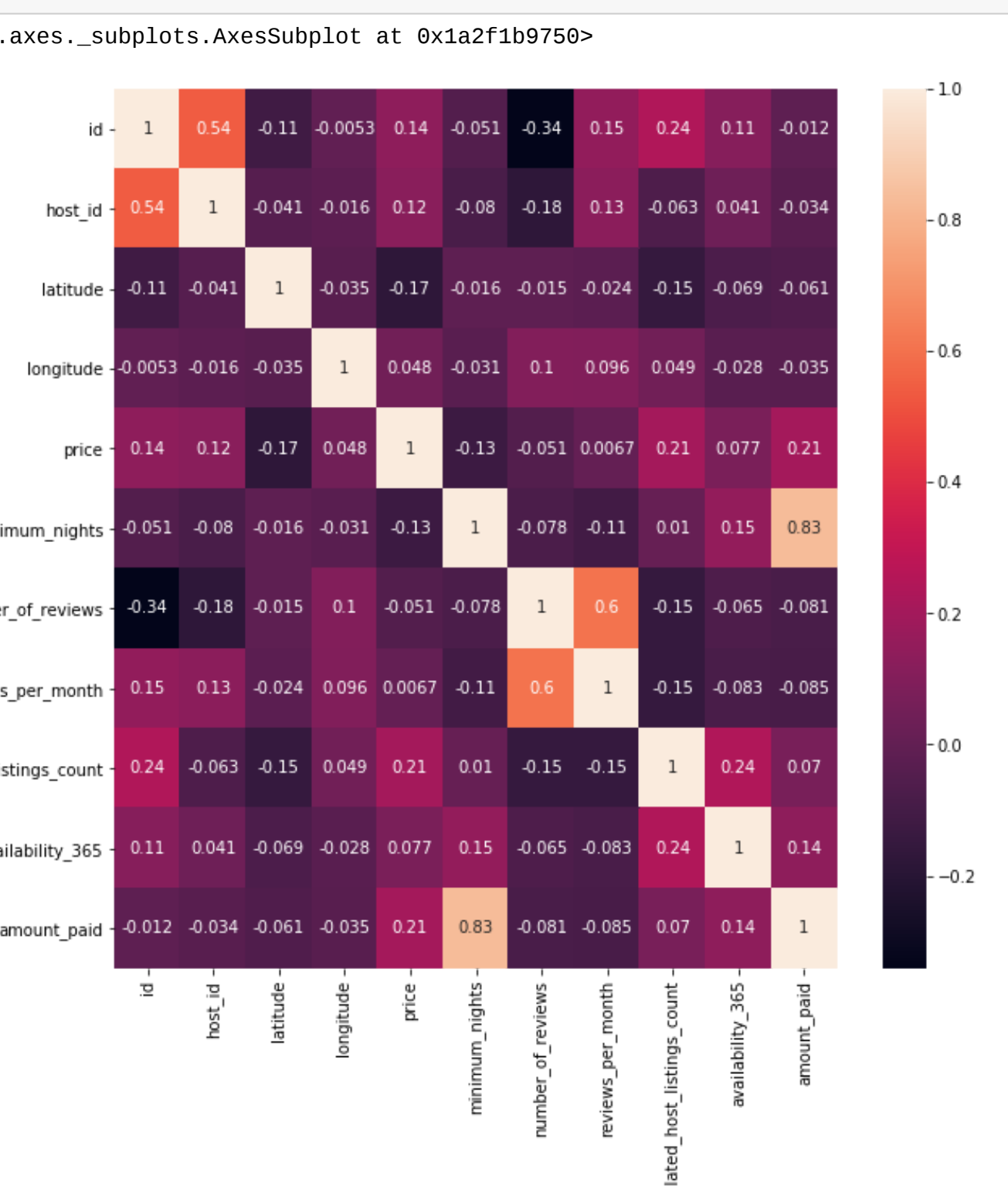
Post Cleaning Results

```
In [283]: df.isna().sum()
```

```
Out[283]: id                                0
          name                              0
          host_id                           0
          host_name                         0
          neighbourhood_group               0
          neighbourhood                     0
          latitude                          0
          longitude                         0
          room_type                         0
          price                             0
          minimum_nights                    0
          number_of_reviews                 0
          reviews_per_month                 0
          calculated_host_listings_count    0
          availability_365                  0
          amount_paid                       0
          dtype: int64
```

```
In [284]: heat = df.corr()
          fig, ax = plt.subplots(figsize = (10,10))
          sns.heatmap(heat,annot=True)
```

```
Out[284]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2f1b9750>
```



```
In [199]: # Calculated host listings count has a significant affect on price and availability 365
```

T-test

```
In [285]: #Grouping neighboourhood w.r.t to price

          groups = df.groupby('neighbourhood_group')['price'].apply(list)
```

```
In [286]: #Finding means of two sub_groups

          East_Reg = pd.Series(groups.loc['East Region'])
          WestReg = pd.Series(groups.loc['West Region'])

          print("The mean of East Region is ",East_Reg.mean())
          print("The mean of West Region is ",WestReg.mean())
```

```
The mean of East Region is  106.90269111872189
The mean of West Region is  100.05281998622186
```

```
In [288]: # N0 Hypothesis : Avergae price in East and West Region is same

          def ttest(dist_1, dist_2, alpha):
              stat, p = ttest_ind(dist_1, dist_2)
              print('Statistics = %.3f, p = %.3f' % (stat, p))
              print('.')
              if p > alpha:
                  print('Null hypothesis True')
              else:
                  print('Null hypothesis False')



          ttest(East_Reg,WestReg, 0.05)
```

```
Statistics = 2.022, p = 0.043
.
Null hypothesis False
```

2nd T-test

This is to chech whether the averge availability_365 of private and Entire home/apt is same or not.

N0_hypothesis : The avg availability_365 of private and Entire home/apt is the same

```
In [289]: Group2 = df.groupby('room_type')['availability_365'].apply(list)
```

```
In [290]: Entire_Homes = pd.Series(Group2.loc['Entire home/apt'])
          Private_room = pd.Series(Group2.loc['Private room'])

          print('Avg availability_365 of Entire homes :',Entire_Homes.mean())
          print("Avg availabiity Private room :", Private_room.mean())
```

```
Avg availability_365 of Entire homes : 213.1041716903786
Avg availabiity Private room : 199.37410926365797
```

```
In [291]: def ttest(dist_1, dist_2, alpha):
              stat, p = ttest_ind(dist_1, dist_2)
              print('Statistics = %.3f, p = %.3f' % (stat, p))
              print('.')
              if p > alpha:
                  print('Null hypothesis accepted')
              else:
                  print('Null hypothesis rejected')


          sample_size = 0
          ttest(Entire_Homes,Private_room, 0.05)
```

```
Statistics = 4.163, p = 0.000
.
Null hypothesis rejected
```

One-Way Anova

Comparison of average minimum nights stay by customers among different types of neighbourhood groups.

N0_Hypothesis: Avg minimum stay in all neighbourhood_groups is the same

```
In [292]: #Grouping with respect to neighbourhood__group and minimum night stay

          group_anova = df.groupby('neighbourhood_group')['minimum_nights']

          Central = pd.Series(group_anova.get_group('Central Region'))
          West = pd.Series(group_anova.get_group('West Region'))
          East = pd.Series(group_anova.get_group('East Region'))
```

```
In [293]: # Finding Averge for each group
          print('Average Means\n')
          print("Central :",Central.mean())
          print('West :',West.mean())
          print('East :',East.mean())
```

```
Average Means

Central : 17.000685639003954
West : 16.810822701567332
East : 20.961183977312984
```

```
In [295]: mod = ols('minimum_nights ~ neighbourhood_group', data=df).fit()
          anon=sm.stats.anova_lm(mod, typ=2)
          print(anon)
          print('\n')
          print('The P value is less than 0.05, hence we reject the null hypothesis')
```

```
                           sum_sq      df         F    PR(>F)
neighbourhood_group  1.275952e+04     4.0  5.986038  0.000083
Residual             4.210871e+06  7902.0       NaN       NaN


The P value is less than 0.05, hence we reject the null hypothesis
```

2nd One-Way Anova

Comparison of average calculated_host_listings_count among different types of room_types.

N0_Hypothesis: Avg calculated_host_listings_count among different types of room_types is the same

```
In [301]: Group4 = df.groupby('room_type')['calculated_host_listings_count'].apply(list)
```

```
In [302]: Homes = pd.Series(Group4.loc['Entire home/apt'])
          Priv_room = pd.Series(Group4.loc['Private room'])
          Shared_room = pd.Series(Group4.loc['Shared room'])

          print('Avg availability_365 of Entire homes :',Homes.mean())
          print("Avg availabiity Private room :", Priv_room.mean())
          print("Avg availabiity Shared room :", Shared_room.mean())
```

```
Avg availability_365 of Entire homes : 62.62334217506631
Avg availabiity Private room : 15.975059382422803
Avg availabiity Shared room : 19.275510204081634
```

```
In [303]: mod = ols('calculated_host_listings_count ~ room_type', data=df).fit()
          anon=sm.stats.anova_lm(mod, typ=2)
          print(anon)
          print('\n')
          print('The P value is less than 0.05, hence we reject the null hypothesis')
```

```
                 sum_sq      df           F         PR(>F)
room_type  4.231979e+06     2.0  570.586646  3.400302e-232
Residual   2.931155e+07  7904.0         NaN            NaN


The P value is less than 0.05, hence we reject the null hypothesis
```

```
In [ ]: 
```