We did a report of the visualization and regression analysis of the distribution of Airbnbs in four signal cities(Shanghai, Istanbul, Singapore and Tokyo) in Asia. We interpret the data about the basic attributes of Airbnb distribution from the website inside Airbnb.

We mainly focused on the prices, room types, locations and popularity. After the visualization of the prices and popularity with room types and neighborhood in the four cities, We run the regressions to see which is the most significant factor to affect the price, then we decide which method is more accurate to predict.

We also did a sentimental analysis to show the key words from the positive reviews for those four cities.

We hope to combine the data analysis, data visualization, and business insights, such as which factors contributed to the price more, and what do customers care about in terms of the sentimental analysis.

In [117]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.image as img
from PIL import Image
import requests
from io import BytesIO
import seaborn as sns
from scipy.stats import norm
from scipy import stats
%matplotlib inline

import statsmodels.formula.api as smf
```

Shanghai

1.1 Data Visualization

In [118]:
```
sh = pd.read_csv('https://raw.githubusercontent.com/AmysnL/data-bootcamp-final-project/main/shanghai.csv
sh.drop(['name','host_name','last_review'],axis = 1,inplace=True)
sh = sh.dropna(subset = ['reviews_per_month'])
sh = sh.loc[sh['price']>0,:]
sh.head(100)
```

Out[118]:

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews
0	24963	98203	NaN	徐汇区 / Xuhui District	31.20918	121.45150	Entire home/apt	468	3	85
1	24991	98203	NaN	徐汇区 / Xuhui District	31.21095	121.45105	Entire home/apt	535	3	1
2	139828	681552	NaN	普陀区 / Putuo District	31.24399	121.44296	Entire home/apt	355	3	26
3	139846	681552	NaN	静安区 / Jing'an District	31.24400	121.44433	Entire home/apt	584	1	57
5	185736	891951	NaN	徐汇区 / Xuhui District	31.21968	121.44930	Private room	500	1	8
...	...	...	...	...	...	...	...	...	...	...
110	3904190	4471201	NaN	虹口区 / Hongkou District	31.25960	121.50027	Private room	227	10	5
111	3904262	4471201	NaN	虹口区 / Hongkou District	31.26088	121.49711	Private room	187	10	5
112	3907673	4471201	NaN	虹口区 / Hongkou District	31.26069	121.49792	Entire home/apt	789	3	2
113	3907888	4471201	NaN	虹口区 / Hongkou District	31.26029	121.49665	Private room	201	10	4
114	3907912	4471201	NaN	虹口区 / Hongkou District	31.25996	121.49591	Private room	234	5	1

100 rows × 13 columns

In [119]:
```
new_col = []
for i in sh['neighbourhood']:
    x = ''.join(i.split()[-2:])
    new_col.append(x)
sh['neighbourhood_en'] =new_col
#= sh['neighbourhood_en']
```

In [ ]:
```
```

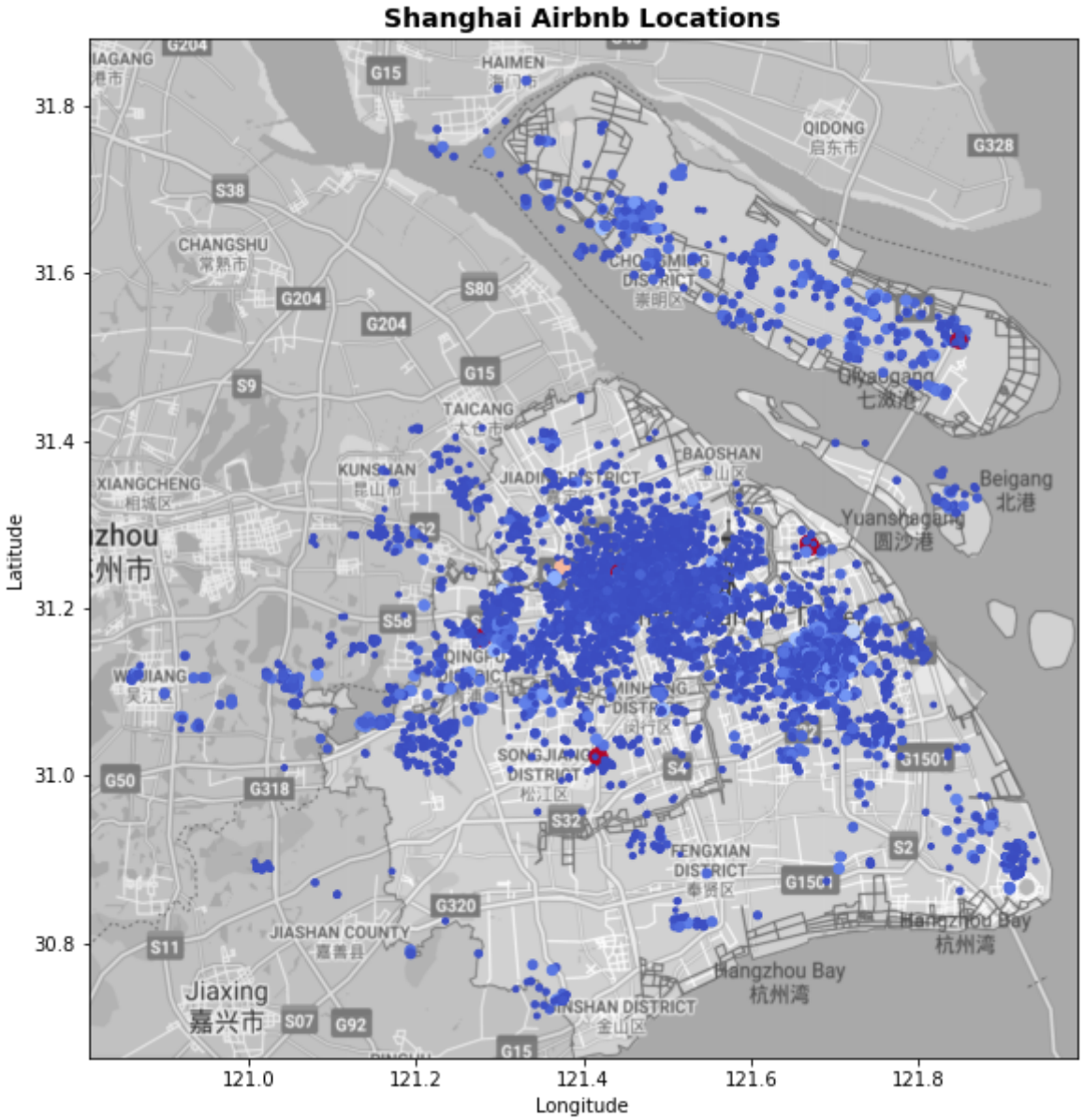
In [120]:
```
def import_img(web):
    response = requests.get(web)
    img = Image.open(BytesIO(response.content))
    return img
```

In [121]:
```
def scatter_a(city,df,sqrt,colorsize,figsize1,figsize2):#to create scatter map for the city
    mpl.rcParams.update(mpl.rcParamsDefault)
    %matplotlib inline

    cmap = plt.cm.coolwarm
    n = mpl.colors.Normalize()
    fig,ax = plt.subplots(figsize=(figsize1,figsize2))
    df.plot.scatter(ax=ax,x='longitude',y='latitude',s=df['price']**sqrt,
                color=cmap(n(df['availability_365'].values)*colorsize))
    ax.set_xlim(df['longitude'].min()-0.05,df['longitude'].max()+0.05)
    ax.set_ylim(df['latitude'].min()-0.05,df['latitude'].max()+0.05)
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    ax.set_title(city +' Airbnb Locations',size=14,fontweight='bold')
    return fig,ax
def map_range(df):#THis is to get the range of map of a certain city
    xl=df['longitude'].min()
    xh=df['longitude'].max()
    yl=df['latitude'].min()
    yh=df['latitude'].max()
    return xl,xh,yl,yh

def location(city,df,sqrt,colorsize,figsize1,figsize2):#to create scatter map for the city
    mpl.rcParams.update(mpl.rcParamsDefault)
    %matplotlib inline

    cmap = plt.cm.coolwarm
    n = mpl.colors.Normalize()
    fig,ax = plt.subplots(figsize=(figsize1,figsize2))
    df.plot.scatter(ax=ax,x='longitude',y='latitude',s=df['price']**sqrt,
                color=cmap(n(df['price'].values)*colorsize))
    ax.set_xlim(df['longitude'].min()-0.05,df['longitude'].max()+0.05)
    ax.set_ylim(df['latitude'].min()-0.05,df['latitude'].max()+0.05)
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    ax.set_title(city+' Airbnb Locations',size=14,fontweight='bold')
    return fig,ax
```

1.1 Location and availabiity Overview

In [122]:
```
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
fig,ax = location('Shanghai', sh,0.4,1.8,10,10)
xl,xh,yl,yh = map_range(sh)
sh_map = import_img('https://github.com/570558305/dmafinal/blob/main/%E6%88%AA%E5%B1%8F2020-12-16%20%E4%

bw_img = sh_map.convert('L')

ax.imshow(bw_img,extent=[xl-0.05 ,xh+0.23 ,yl-0.25 ,yh+0.13 ], cmap='gray')
```

Out[122]:
```
<matplotlib.image.AxesImage at 0x129f45550>
```



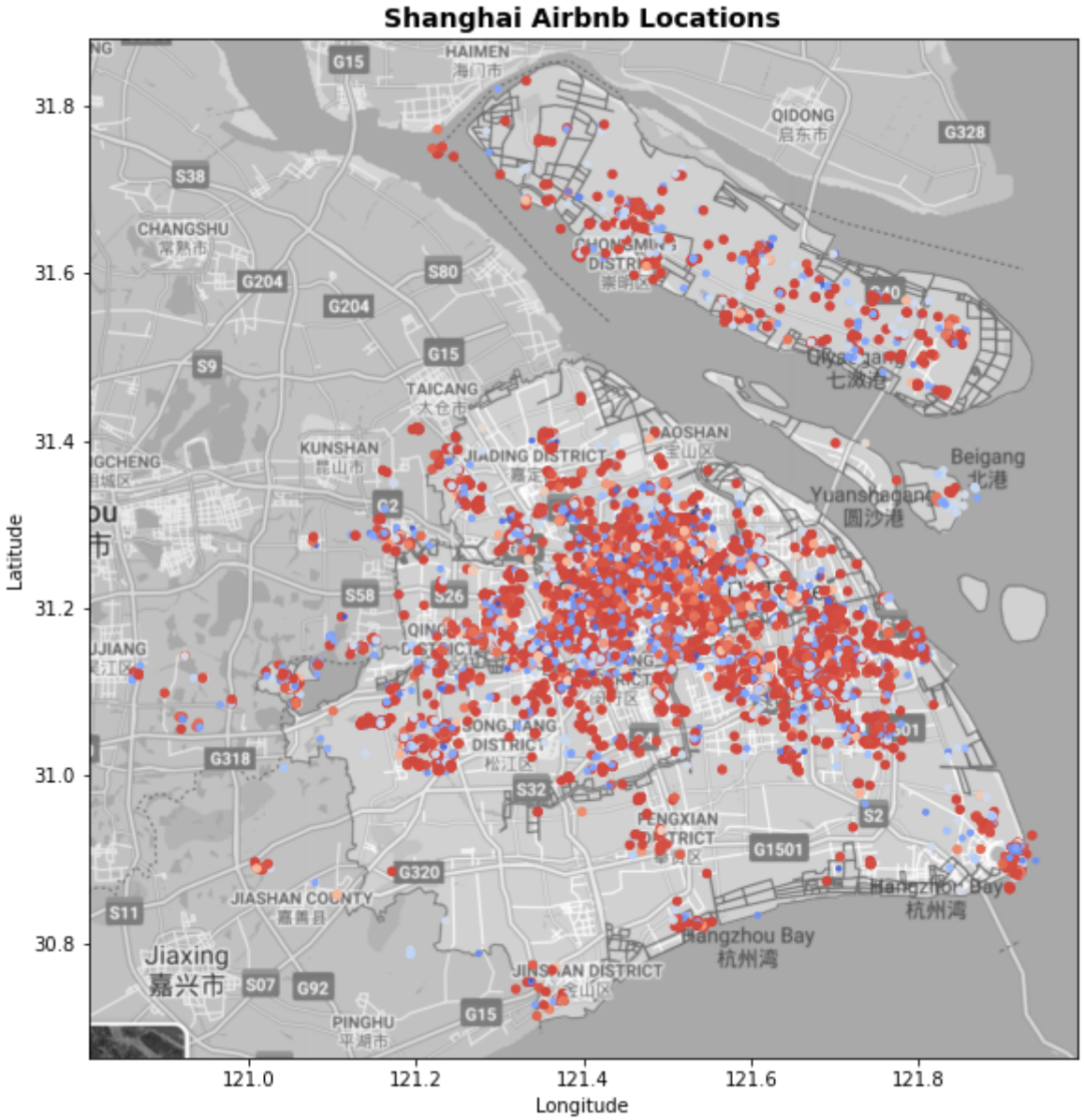
Note that the colder the color, the cheaper the price, the red spot shows that the prices are really high.

The scatter plot here shows that the distribution of Airbnbs located in Shanghai. They mainly located in two parts in Shanghai, the one hub is located at the west of Oriental Tower, the other is located at the east of the tower.

In [123]:
```
def scatter_(city,df,sqrt,colorsize,figsize1,figsize2):#to create scatter map for the city
    mpl.rcParams.update(mpl.rcParamsDefault)
    %matplotlib inline

    cmap = plt.cm.coolwarm
    n = mpl.colors.Normalize()
    fig,ax = plt.subplots(figsize=(figsize1,figsize2))
    df.plot.scatter(ax=ax,x='longitude',y='latitude',s=df['availability_365']**sqrt,
                color=cmap(n(df['availability_365'].values)*colorsize))
    ax.set_xlim(df['longitude'].min()-0.05,df['longitude'].max()+0.05)
    ax.set_ylim(df['latitude'].min()-0.05,df['latitude'].max()+0.05)
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    ax.set_title(city+' Airbnb Locations',size=14,fontweight='bold')
    return fig,ax
```

In [124]:
```
fig,ax = scatter_('Shanghai', sh,0.5,0.92,10,10)
xl,xh,yl,yh = map_range(sh)
sh_map = import_img('https://github.com/570558305/dmafinal/blob/main/%E6%88%AA%E5%B1%8F2020-12-16%20%E4%
bw_img = sh_map.convert('L')
#plt.imshow(bw_img, cmap = 'gray')

ax.imshow(bw_img,extent=[xl-0.1 ,xh+0.2 ,yl-0.2 ,yh+0.14], cmap = 'gray')
```

Out[124]:
```
<matplotlib.image.AxesImage at 0x12a4810a0>
```



The scatter plot shows the availability of airbnbs in Shanghai, the warmer the color, there are more available rooms at the airbnb represented by the spot, vice versa.

1.2 Neighborhood

1.21 The distribution of Airbnbs in each district

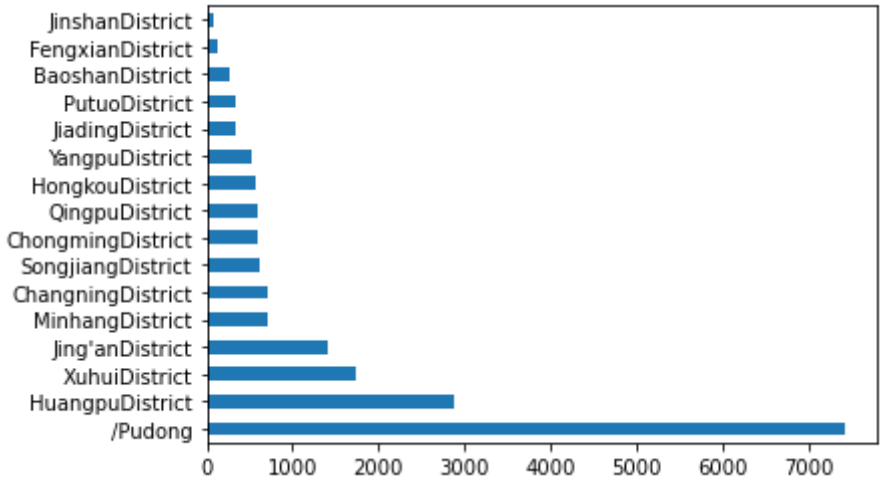
In [125]:
```
pd.DataFrame(sh['neighbourhood'].unique())
```

Out[125]:

	0
0	徐汇区 / Xuhui District
1	普陀区 / Putuo District
2	静安区 / Jing'an District
3	长宁区 / Changning District
4	杨浦区 / Yangpu District
5	虹口区 / Hongkou District
6	浦东新区 / Pudong
7	黄浦区 / Huangpu District
8	嘉定区 / Jiading District
9	宝山区 / Baoshan District
10	松江区 / Songjiang District
11	青浦区 / Qingpu District
12	奉贤区 / Fengxian District
13	崇明区 / Chongming District
14	闵行区 / Minhang District
15	金山区 / Jinshan District

In [126]:
```
CountStatus = pd.value_counts(sh['neighbourhood_en'].values, sort=True)
CountStatus.plot.barh()
```

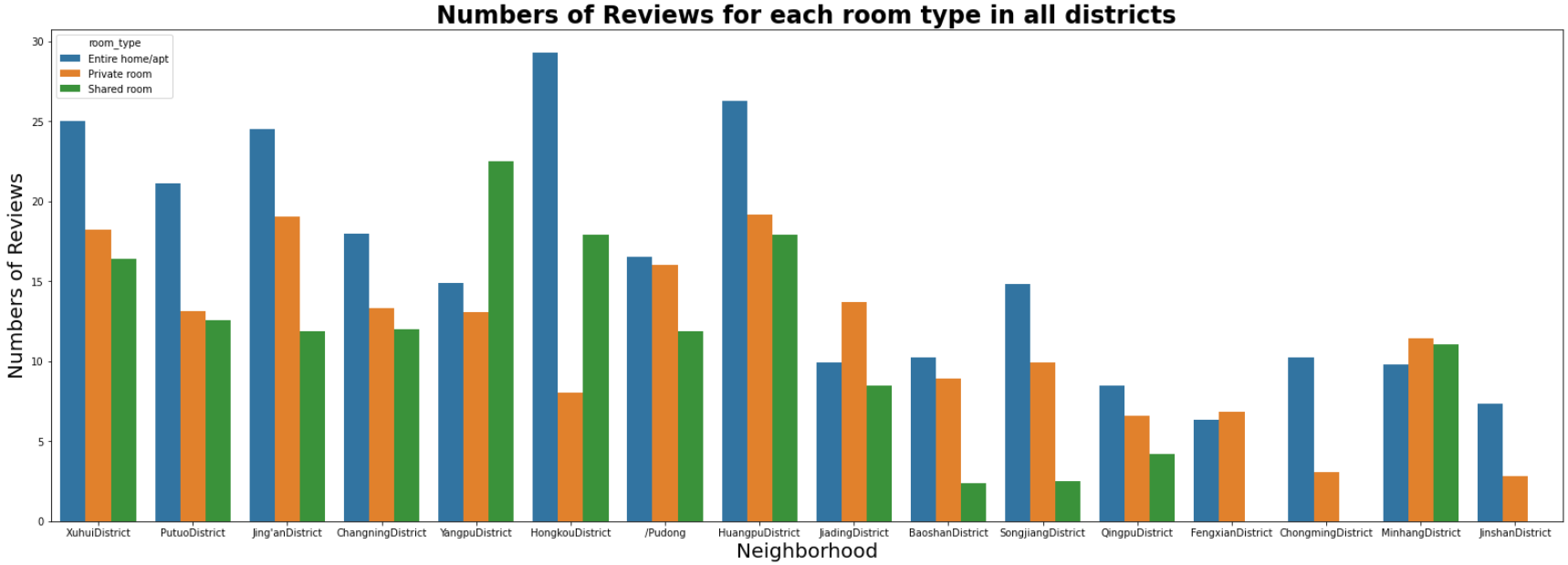
Out[126]: <AxesSubplot:>



Pudong has the most airbnbs located, Huangpu, Xuhui and Jingan followed. Jinshan and Fengxian has the least airbnbs.

In [127]:
```
fig, ax = plt.subplots(figsize=(27,9))
ax =sns.barplot(data=sh, x="neighbourhood_en", y ='number_of_reviews', hue="room_type",ci=None )

ax.set_xlabel('Neighborhood',fontsize=20)
ax.set_ylabel('Numbers of Reviews',fontsize=20)
ax.set_title('Numbers of Reviews for each room type in all districts',fontweight='bold',fontsize=24)
```

Out[127]: Text(0.5, 1.0, 'Numbers of Reviews for each room type in all districts')



The number of Reviews indicates the popularity of the rooms and neighborhood. We can see that the blue lines, which represents the entire home/apartment, are generally higher than others, especially in Hongkou, Huangpu and Jingan. Shared rooms in Yangpu and Xuhui,

Private rooms in Huangpu, Xuhui and Jing'an are also very popular.

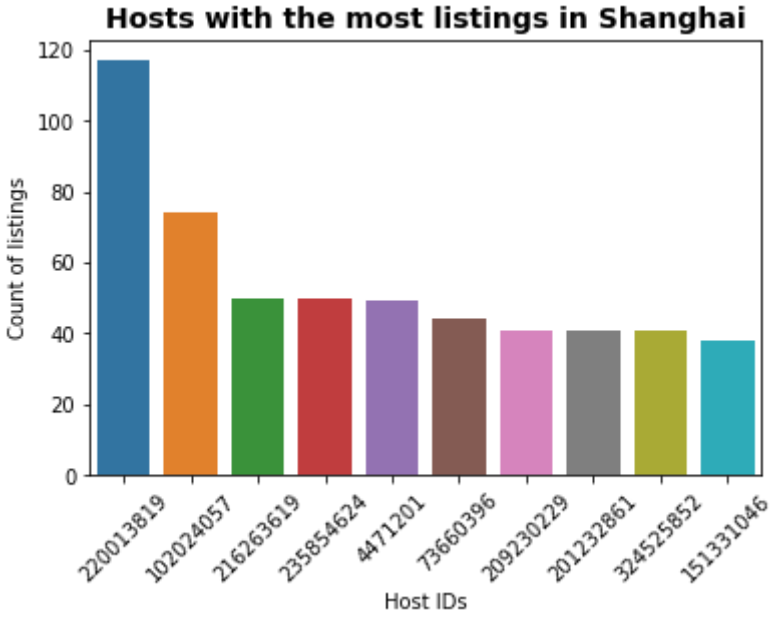
1.3 Host

In [128]:
```
top_host = sh['host_id'].value_counts()[:10]
```

In [129]:
```
ax = sns.barplot(top_host.index, top_host.values,order=top_host.index)
ax.set_xticklabels(ax.get_xticklabels (),rotation=45)
ax.set_title('Hosts with the most listings in Shanghai',size=14,fontweight='bold')
ax.set_ylabel('Count of listings')
ax.set_xlabel('Host IDs')
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

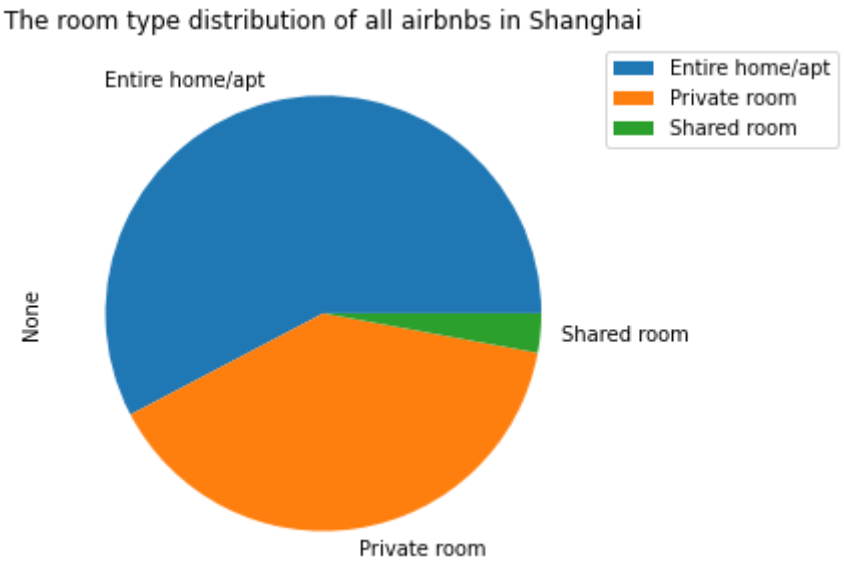
Out[129]:
```
Text(0.5, 0, 'Host IDs')
```

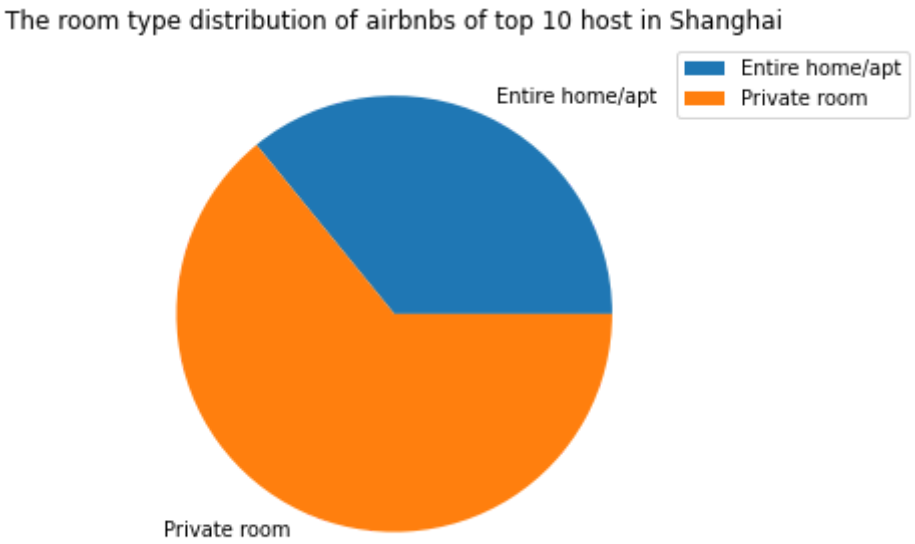


Here is the top 10 hosts who has the most airnbnbs listed. Among them, 22013819 has the most listing which is over 100.

In [130]:
```
CountStatus = pd.value_counts(sh['room_type'].values, sort=True)
CountStatus.plot.pie(figsize = (5,5))
plt.legend(bbox_to_anchor=(1, 1))
plt.title('The room type distribution of all airbnbs in Shanghai')
x =sh["host_id"].value_counts().nlargest(n=10).index.values
host_220 = sh.loc[sh['host_id'].isin(x) ,:]
host_220 = sh.loc[sh['host_id']==220013819,:]

host_rt = host_220.groupby(['room_type'])[['room_type']].count()
fix,ax = plt.subplots(figsize = (5,5))
host_rt.plot(kind='pie',x='room_type',ax=ax,subplots=True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.legend(bbox_to_anchor=(1, 1))
plt.title('The room type distribution of airbnbs of top 10 host in Shanghai')
```

Out[130]:
```
Text(0.5, 1.0, 'The room type distribution of airbnbs of top 10 host in Shanghai')
```

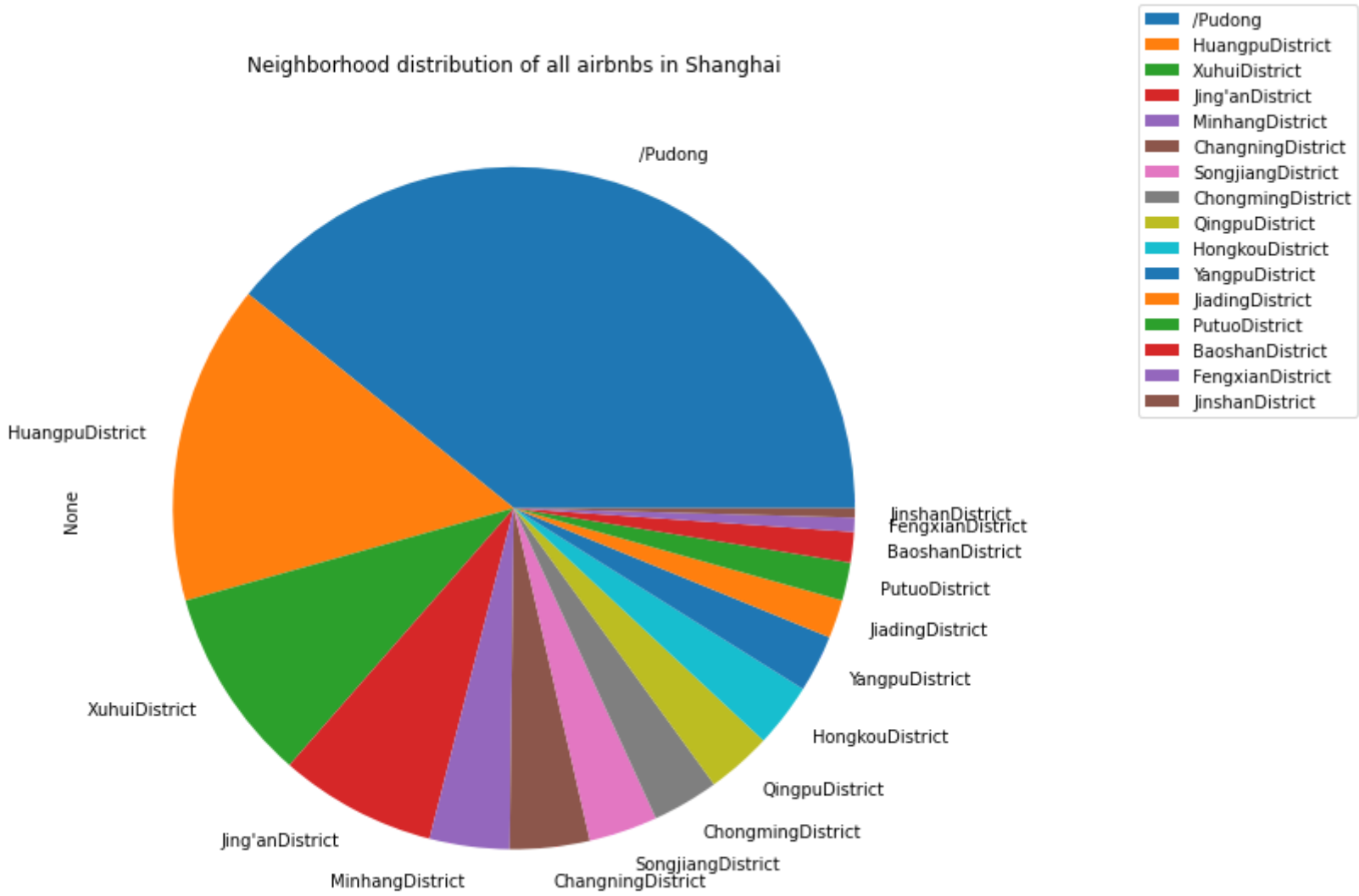


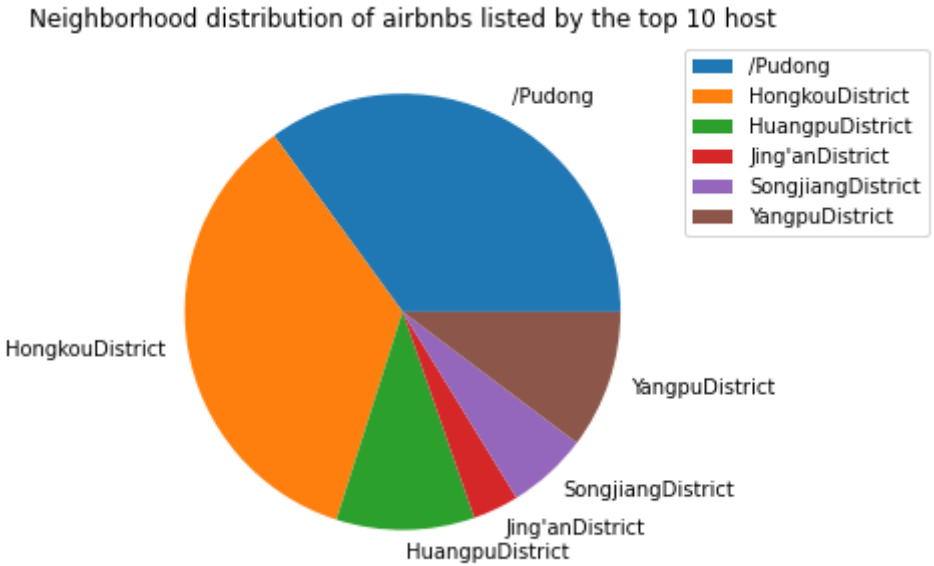


The first pie chart shows that the room types of all the airbnbs located in Shanghai while the second shows that the room types of the airbnbs listed by the top 10 hosts. We can see those hosts who has more listings, the larger scale of business, prefer to provide private rooms rather than the entire rooms.

In [131]:
```
CountStatus = pd.value_counts(sh['neighbourhood_en'].values, sort=True)
CountStatus.plot.pie(figsize = (9,10))
plt.title('Neighborhood distribution of all airbnbs in Shanghai')
plt.legend(bbox_to_anchor=(1.5, 1.1))

host_n = host_220.groupby(['neighbourhood_en'])[['neighbourhood_en']].count()
fix,ax = plt.subplots(figsize = (7,5))
host_n.plot(kind='pie',x='neighbourhood_en',ax=ax,subplots=True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.title('Neighborhood distribution of airbnbs listed by the top 10 host')
plt.legend(bbox_to_anchor=(1, 1))
```

Out[131]: <matplotlib.legend.Legend at 0x126b0ca90>


Neighborhood distribution of all airbnbs in Shanghai
/Pudong
HuangpuDistrict
XuhuiDistrict
Jing'anDistrict
MinhangDistrict
ChangningDistrict
SongjiangDistrict
ChongmingDistrict
QingpuDistrict
HongkouDistrict
YangpuDistrict
JiadingDistrict
PutuoDistrict
BaoshanDistrict
FengxianDistrict
JinshanDistrict
None

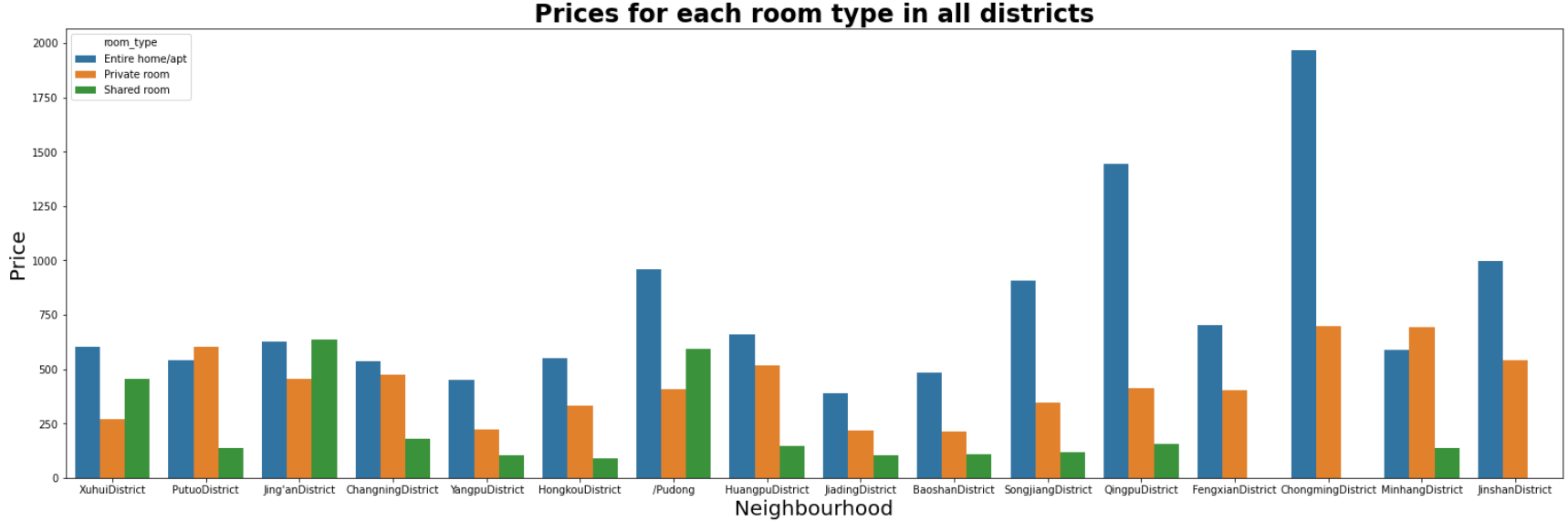


The first pie chart shows that the neighborhood of all the airbnbs located in Shanghai while the second shows that the neighbourhood locations of the airbnbs listed by the top 10 hosts. We can see those hosts who has more listings, the larger scale of business, has more rooms at Hongkou and Yangpu than average. Their mainly distributions are in Pudong, Hongkou, Huangpu and Yangpu.

1.4 Prices

In [132]:
```
fig, ax = plt.subplots(figsize=(26,8))
ax =sns.barplot(data=sh, x="neighbourhood_en", y ='price', hue="room_type",ci=None)
ax.set_xlabel('Neighbourhood',fontsize=20)
ax.set_ylabel('Price',fontsize=20)
ax.set_title('Prices for each room type in all districts',fontweight='bold',fontsize=24)
```

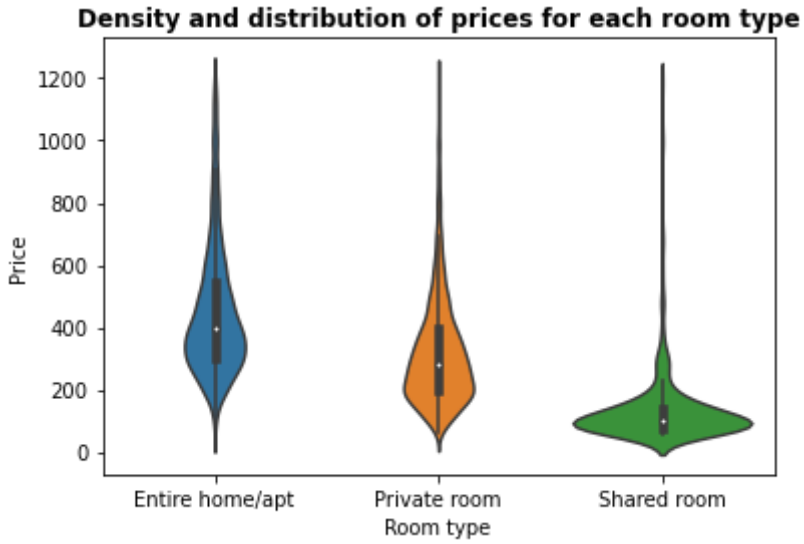
Out[132]:
```
Text(0.5, 1.0, 'Prices for each room type in all districts')
```



The bar chart shows the prices distribution in all districts. It is clear that generally, the prices of entire room are higher, especially in Chongming, Qingpu, Jinshan, Songjiang and Pudong. The commonality of them is that they are all more far away from the center of Shanghai than the others. Also, the variance of the private room prices in all neighborhood are relatively small.

In [133]:
```
ax = sns.violinplot(data=sh[sh.price < 1200] ,x='room_type',y='price')

ax.set_xlabel('Room type')
ax.set_ylabel('Price')
ax.set_title('Density and distribution of prices for each room type',fontweight='bold')
```

Out[133]: Text(0.5, 1.0, 'Density and distribution of prices for each room type')



2. Descriptive Data Analysis

2.1Data Overview

In [134]:
```
sh.describe()
```

Out[134]:

	id	host_id	neighbourhood_group	latitude	longitude	price	minimum_nights	number_of_reviews
count	1.897100e+04	1.897100e+04	0.0	18971.000000	18971.000000	18971.000000	18971.000000	18971.000000
mean	3.333042e+07	1.759078e+08	NaN	31.194727	121.515302	625.776554	3.687101	17.615044
std	9.492727e+06	9.776061e+07	NaN	0.107171	0.150654	1619.554824	19.602914	32.296250
min	2.496300e+04	9.820300e+04	NaN	30.712780	120.859010	55.000000	1.000000	1.000000
25%	2.724361e+07	9.460691e+07	NaN	31.141185	121.443750	252.000000	1.000000	2.000000
50%	3.517468e+07	1.720595e+08	NaN	31.203550	121.484770	360.000000	1.000000	6.000000
75%	4.082778e+07	2.544090e+08	NaN	31.233035	121.661200	537.000000	1.000000	18.000000
max	4.601800e+07	3.722724e+08	NaN	31.829920	121.940020	70780.000000	500.000000	426.000000

In [135]:
```
sh.isnull().sum()
```

Out[135]:
```
id                                  0
host_id                             0
neighbourhood_group             18971
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
neighbourhood_en                    0
dtype: int64
```

In [136]:
```
sh.drop(columns=['neighbourhood_group'],axis=1,inplace=True)
sh
```

Out[136]:

	id	host_id	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_mo
0	24963	98203	徐汇区 / Xuhui District	31.20918	121.45150	Entire home/apt	468	3	85	0
1	24991	98203	徐汇区 / Xuhui District	31.21095	121.45105	Entire home/apt	535	3	1	0
2	139828	681552	普陀区 / Putuo District	31.24399	121.44296	Entire home/apt	355	3	26	0
3	139846	681552	静安区 / Jing'an District	31.24400	121.44433	Entire home/apt	584	1	57	0
5	185736	891951	徐汇区 / Xuhui District	31.21968	121.44930	Private room	500	1	8	0
...	...	...	...	...	...	...	...	...	...	
35370	45980053	344682139	长宁区 / Changning District	31.22631	121.35636	Entire home/apt	479	1	1	1
35502	46001320	89259546	黄浦区 / Huangpu District	31.22222	121.45827	Entire home/apt	326	1	1	1
35506	46003354	152976705	徐汇区 / Xuhui District	31.18105	121.44647	Entire home/apt	248	1	1	1
35510	46005230	246151722	浦东新区 / Pudong	30.91364	121.91692	Private room	133	1	1	1
35565	46018001	323466730	静安区 / Jing'an District	31.25939	121.46204	Entire home/apt	218	1	1	1

18971 rows × 13 columns

In [ ]:
```
```

The Correlations here is about all the columns, however, not all of them are valid information. We set it as a reference that can be looked up if needed.

2.2 Descriptive Data Analysis¶

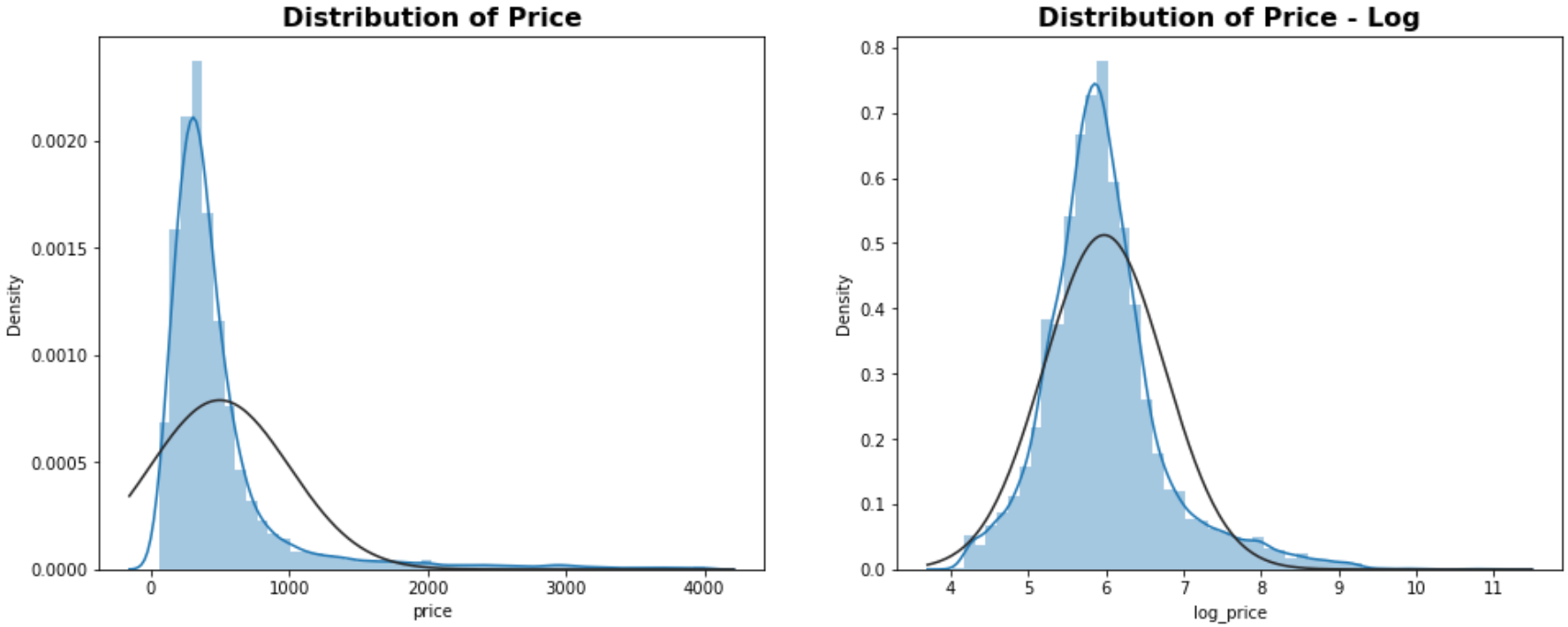
In [137]:
```
# encode str
en_sh = sh.copy()
en_sh['neighbourhood'] = en_sh['neighbourhood'].astype('category').cat.codes
en_sh['room_type'] = en_sh['room_type'].astype("category").cat.codes
mean = en_sh['reviews_per_month'].mean()
en_sh['reviews_per_month'].fillna(mean, inplace=True) ####
en_sh['log_price'] = np.log(en_sh.price+1)  #####
en_sh = en_sh.drop(columns=['id' ,'host_id','price']) # delete price column
en_sh.isnull().sum()
```

Out[137]:
```
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
minimum_nights                    0
number_of_reviews                 0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
neighbourhood_en                  0
log_price                         0
dtype: int64
```

In [138]:
```
fig,ax = plt.subplots(1,2,figsize = (16,6))
sns.distplot(sh['price'].loc[sh['price'] <= 4000], fit = norm, ax = ax[0])
ax[0].set_title("Distribution of Price", size = 16,weight = 'bold')
sns.distplot(en_sh['log_price'],fit=norm , ax = ax[1])
ax[1].set_title("Distribution of Price - Log", size = 16,weight = 'bold')
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```

Out[138]: Text(0.5, 1.0, 'Distribution of Price - Log')



We can see that the log-price distribution is approximately normal distribution of the mean of 6.

2.3 Correlation

In [139]:
```
listings_df = sh
```

In [140]:
```
# Change data type
listings_price_df = listings_df[['price','minimum_nights','number_of_reviews','reviews_per_month','calcu
listings_price_df[:5]
```

Out[140]:

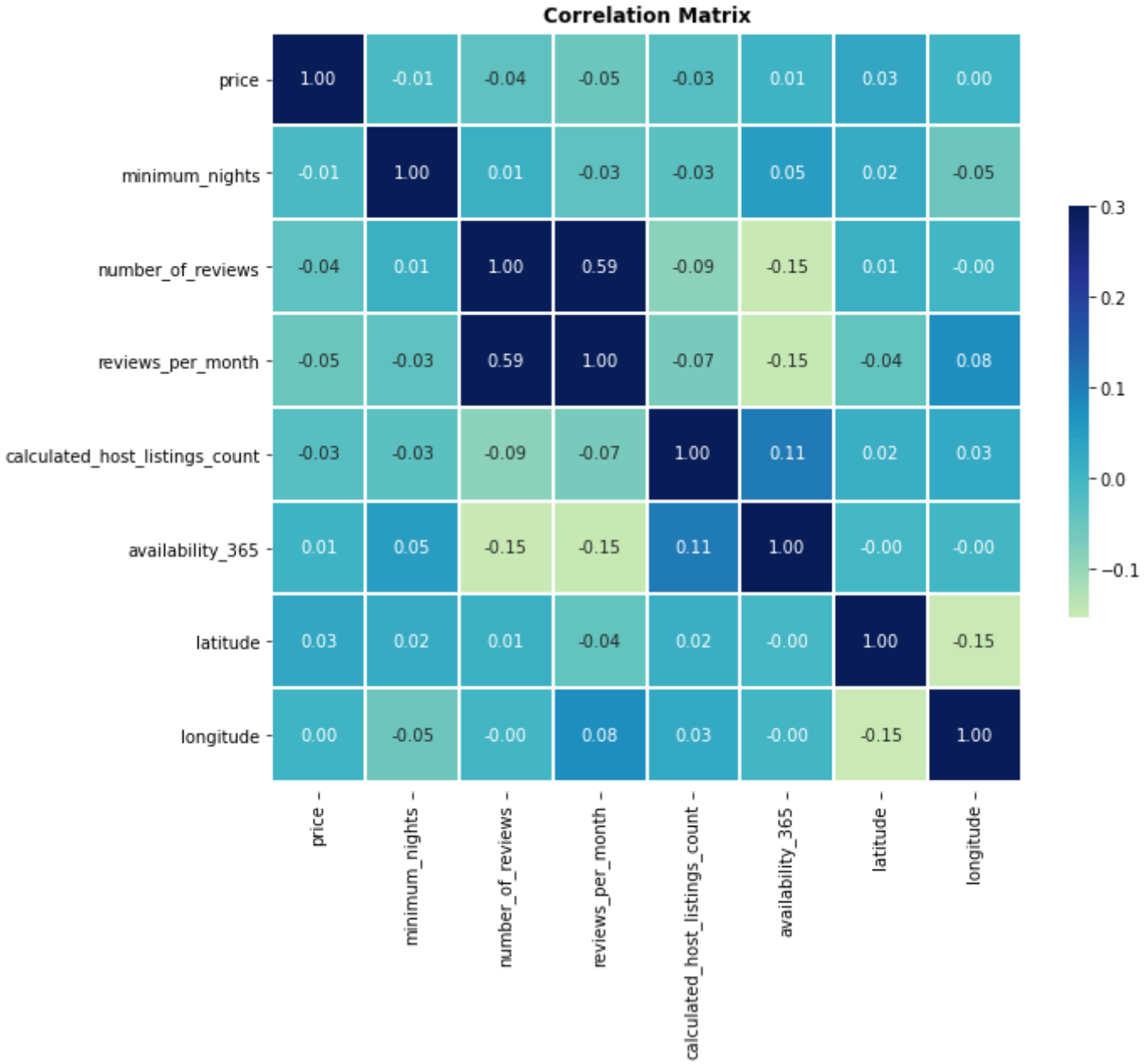
	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365	latitude	longitude
0	468	3	85	0.69	2	329	31.20918	121.45150
1	535	3	1	0.01	2	68	31.21095	121.45105
2	355	3	26	0.25	16	244	31.24399	121.44296
3	584	1	57	0.51	16	227	31.24400	121.44433
5	500	1	8	0.17	1	364	31.21968	121.44930

This is the correlation matrix showing relationships between different variables.

This will help the host to improve decisions.

In [141]:
```
plt.figure(figsize=(10,10))
palette = sns.diverging_palette(20, 220, n=256)
corr=listings_price_df.corr(method='pearson')
sns.heatmap(corr, annot=True, fmt=".2f", cmap="YlGnBu", vmax=.3, center=0,square=True, linewidths=1, cba
plt.title("Correlation Matrix",size=12, weight='bold')
```

Out[141]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```



3. Regression

3.1 Linear Regression

In [142]:
```
reg_price = smf.ols('price ~ neighbourhood +room_type + availability_365 + number_of_reviews + calculate
                    ,data=sh).fit()
print(reg_price.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.029
Model:                            OLS   Adj. R-squared:                  0.028
Method:                 Least Squares   F-statistic:                     27.31
Date:                Fri, 18 Dec 2020   Prob (F-statistic):          3.28e-106
Time:                        10:26:47   Log-Likelihood:            -1.6683e+05
No. Observations:               18971   AIC:                         3.337e+05
Df Residuals:                   18949   BIC:                         3.339e+05
Df Model:                          21
Covariance Type:            nonrobust
================================================================================================================
                                                   coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------------------------------------------------
Intercept                                     507.0284     90.167      5.623      0.000     330.293
683.763
neighbourhood[T.奉贤区 / Fengxian District]    267.7108    165.726      1.615      0.106     -57.126
592.548
neighbourhood[T.宝山区 / Baoshan District]      92.4224    129.302      0.715      0.475    -161.022
345.867
neighbourhood[T.崇明区 / Chongming District]  1114.5657    108.098     10.311      0.000     902.684
1326.447
neighbourhood[T.徐汇区 / Xuhui District]       217.3812     94.406      2.303      0.021      32.337
402.426
neighbourhood[T.普陀区 / Putuo District]       311.0914    121.979      2.550      0.011      72.001
550.182
neighbourhood[T.杨浦区 / Yangpu District]      109.1023    111.483      0.979      0.328    -109.413
327.618
neighbourhood[T.松江区 / Songjiang District]   418.5277    107.321      3.900      0.000     208.170
628.885
neighbourhood[T.浦东新区 / Pudong]              467.0881     88.497      5.278      0.000     293.626
640.551
neighbourhood[T.虹口区 / Hongkou District]     215.7792    109.488      1.971      0.049       1.173
430.385
neighbourhood[T.金山区 / Jinshan District]     505.3903    194.516      2.598      0.009     124.122
886.658
```

```
neighbourhood[T.长宁区 / Changning District]   226.7623    104.767      2.164      0.030      21.409
432.115
neighbourhood[T.闵行区 / Minhang District]     315.2999    104.605      3.014      0.003     110.265
520.335
neighbourhood[T.青浦区 / Qingpu District]      712.3114    108.489      6.566      0.000     499.663
924.960
neighbourhood[T.静安区 / Jing'an District]     298.8031     96.194      3.106      0.002     110.253
487.353
neighbourhood[T.黄浦区 / Huangpu District]     293.6454     91.505      3.209      0.001     114.288
473.003
room_type[T.Private room]                   -434.8098     25.786    -16.863      0.000    -485.352    -
384.268
room_type[T.Shared room]                    -396.5248     70.455     -5.628      0.000    -534.623    -
258.427
availability_365                               0.0230      0.093      0.249      0.803      -0.158
0.204
number_of_reviews                             -0.7698      0.452     -1.704      0.088      -1.655
0.115
calculated_host_listings_count                -1.2220      0.380     -3.215      0.001      -1.967
-0.477
reviews_per_month                            -46.2366      9.997     -4.625      0.000     -65.832
-26.641
==============================================================================
Omnibus:                    45439.067   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        624873210.723
Skew:                          24.608   Prob(JB):                         0.00
Kurtosis:                     890.749   Cond. No.                     8.29e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.29e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Here we run an ordinary least square regression for price on factors that may affect the price through the exploration and observation. We have H0: beta = 0 against H1: beta != 0 for every beta in this regression model.

From the regression result we can see that neighbourhood coontributed positively to Price. The neighbourhood that contributed the most is Chongming District.

3.2 Log

In [143]:
```
reg_logprice = smf.ols('log_price ~ neighbourhood +room_type + availability_365 + number_of_reviews + ca
                       ,data= en_sh).fit()
print(reg_logprice.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_price   R-squared:                       0.173
Model:                            OLS   Adj. R-squared:                  0.173
Method:                 Least Squares   F-statistic:                     663.5
Date:                Fri, 18 Dec 2020   Prob (F-statistic):               0.00
Time:                        10:26:47   Log-Likelihood:                -20360.
No. Observations:               18971   AIC:                         4.073e+04
Df Residuals:                   18964   BIC:                         4.079e+04
Df Model:                           6
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                        6.3144      0.018    351.978      0.000       6.279       6.350
neighbourhood                   -0.0004      0.001     -0.270      0.787      -0.003       0.002
room_type                       -0.5863      0.009    -62.623      0.000      -0.605      -0.568
availability_365             -8.852e-05   4.09e-05     -2.166      0.030      -0.000   -8.42e-06
number_of_reviews               -0.0011      0.000     -5.558      0.000      -0.001      -0.001
calculated_host_listings_count  -0.0007      0.000     -3.967      0.000      -0.001      -0.000
reviews_per_month               -0.0157      0.004     -3.583      0.000      -0.024      -0.007
==============================================================================
Omnibus:                     5615.257   Durbin-Watson:                   1.864
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            20693.488
Skew:                           1.456   Prob(JB):                         0.00
Kurtosis:                       7.208   Cond. No.                         965.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

From the regressin result above, we can see that neighbourhood, room_type, availability_365, number_of_reviews, calculated_host_listings_count, and reviews_per_month, all contribute negatively to Price.

Since the P value of neighbourhood is greater than 5%, its impact on price is not significant.

4. Machine Learning

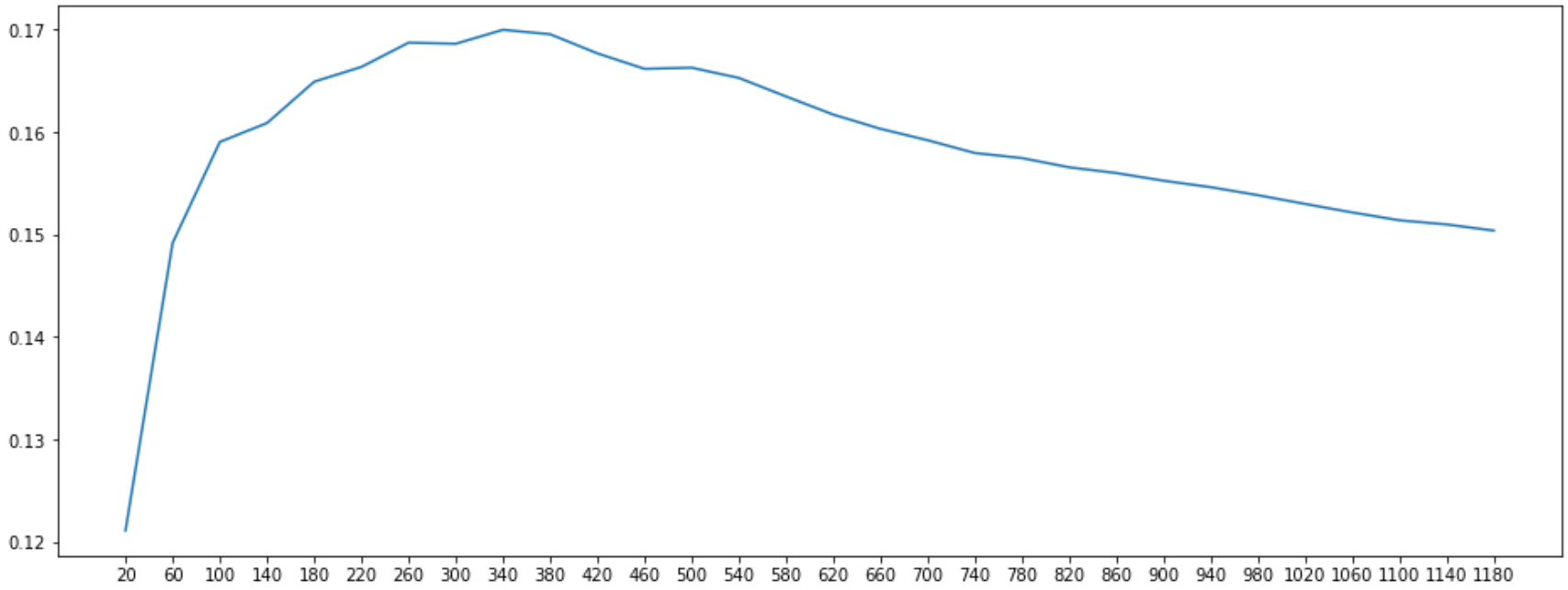
In [144]:
```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(en_sh[['room_type']].values, en_sh[['log_price']
                                                        test_size=0.25, random_state=0)
```

(1) KNN

In [*]:
```
from sklearn.neighbors import KNeighborsRegressor as knn
from sklearn.model_selection import cross_val_score
scores = pd.Series(dtype='float')
for i in range(20,1200,40):
    scores[str(i)] = cross_val_score(knn(n_neighbors=i),X_train1, y_train1,cv=5).mean()
indx = scores.idxmax()
```

In [146]:
```
fig,ax = plt.subplots(figsize=(16,6))
plt.plot(scores.index,scores.values)
```

Out[146]:
```
[<matplotlib.lines.Line2D at 0x1290b5790>]
```



In [147]:
```
skl_knn = knn(n_neighbors = int(indx)).fit(X_train1, y_train1)  #####
knn_score1 = skl_knn.score(X_test1,y_test1)
print(knn_score1)
```

```
0.170708579377991
```

(2) Random Forest

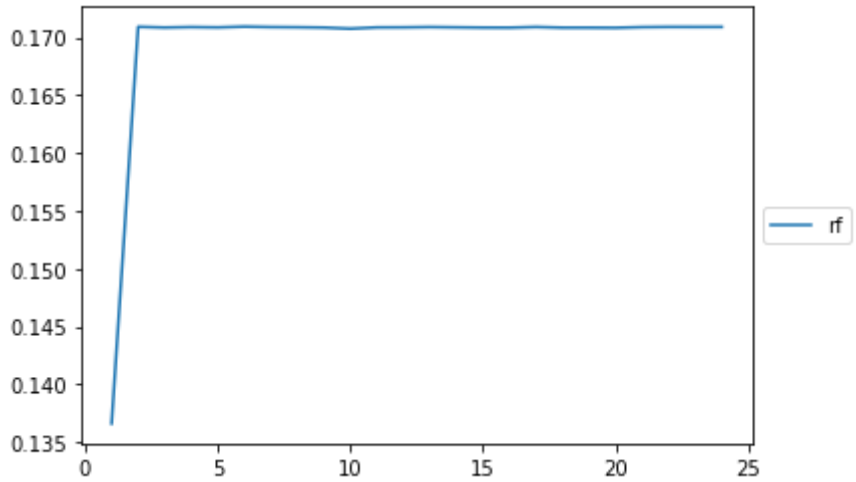
In [148]:
```
from sklearn.ensemble import RandomForestRegressor as rf
cross_val_score(rf(n_estimators = 100, max_depth = 3), X_train1, y_train1.ravel(), cv=5).mean()
```

Out[148]:
```
0.1708614249914519
```

In [149]:
```
cv_scores = pd.DataFrame()
for i in range (1,25):
    cv_scores.loc[i,'rf'] = cross_val_score(rf(n_estimators = 100, max_depth = i), X_train1, y_train1.ra
```

In [150]:
```
ax = cv_scores.plot()
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
```

Out[150]: <matplotlib.legend.Legend at 0x129137c40>



In [151]:
```
indx = scores.idxmax()
skl_rf = rf(n_estimators=100,max_depth=int(indx)).fit(X_train1, y_train1.ravel())
rf_score1 = skl_rf.score(X_test1,y_test1)
print(rf_score1)
```

```
0.1779720637527994
```

In [152]:
```
score_table = pd.DataFrame({'K Nearest Neighbors':[knn_score1],'Random Forest':[rf_score1]})
score_table = score_table.T
score_table.columns = ['Score']
score_table
```

Out[152]:

	Score
K Nearest Neighbors	0.170709
Random Forest	0.177972

According to the table above, we can use Random Forest as the score for this model for log_price, which is higher than KNN.

However, the score of KNN and Random Forest are relatively low. It could be due to some intangible features such as

the room quality, service friendliness, and the environment, etc

Istanbul

In [153]:
```
istb = pd.read_csv('https://raw.githubusercontent.com/AmysnL/data-bootcamp-final-project/main/Istanbul.c
```

In [154]:
```
istb.head()
```

Out[154]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights
0	4826	The Place	6603	Kaan	NaN	Uskudar	41.05650	29.05367	Entire home/apt	720	1
1	20815	The Bosphorus from The Comfy Hill	78838	Gülder	NaN	Besiktas	41.06984	29.04545	Entire home/apt	816	365
2	27271	LOVELY APT. IN PERFECT LOCATION	117026	Mutlu	NaN	Beyoglu	41.03254	28.98153	Entire home/apt	233	30
3	28277	Duplex Apartment with Terrace	121607	Alen	NaN	Sisli	41.04471	28.98567	Hotel room	761	3
4	28318	Cosy home overlooking Bosphorus	121721	Aydin	NaN	Sariyer	41.09048	29.05559	Entire home/apt	823	3

In [155]:
```
istb.drop(['name','host_name','last_review'],axis = 1,inplace=True)
istb = istb.dropna(subset = ['reviews_per_month'])
istb = istb.loc[istb['price']>0,:]
istb.head(100)
```

Out[155]:

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews
0	4826	6603	NaN	Uskudar	41.05650	29.05367	Entire home/apt	720	1	1
1	20815	78838	NaN	Besiktas	41.06984	29.04545	Entire home/apt	816	365	41
2	27271	117026	NaN	Beyoglu	41.03254	28.98153	Entire home/apt	233	30	13
6	30697	132137	NaN	Beyoglu	41.03350	28.97626	Private room	768	1	1
7	33368	135136	NaN	Sisli	41.05382	28.99739	Private room	384	2	1
...	...	...	...	...	...	...	...	...	...	...
143	374837	466302	NaN	Beyoglu	41.02752	28.97705	Entire home/apt	240	1	126
144	378120	1899923	NaN	Beyoglu	41.03499	28.97238	Entire home/apt	733	7	11
145	378383	1900682	NaN	Adalar	40.87361	29.13289	Entire home/apt	734	2	25
147	385296	734040	NaN	Sariyer	41.08679	29.05244	Private room	144	7	10
148	391645	1958346	NaN	Kadikoy	40.99609	29.02780	Entire home/apt	500	3	72

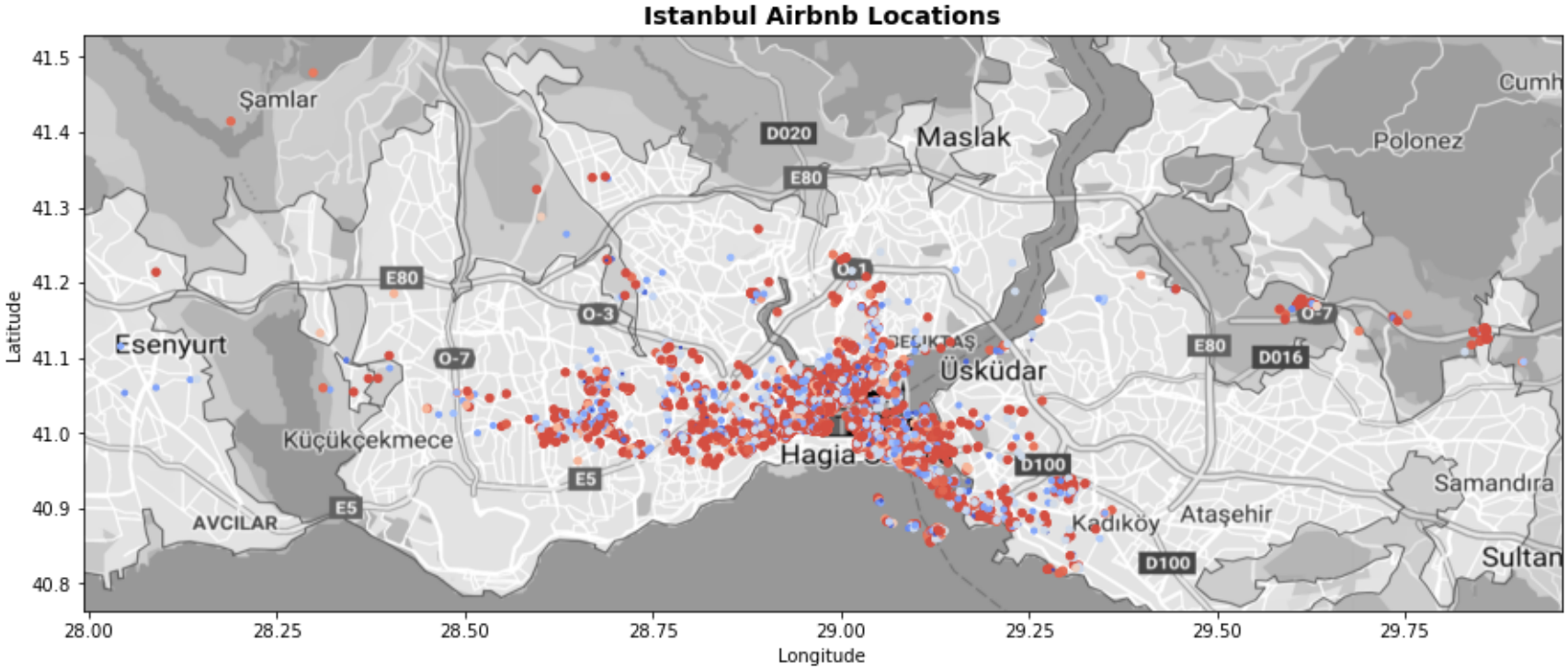
100 rows × 13 columns

1. Data Visualization

1.1 Location and availabiitv Overview

In [156]:
```
fig,ax = scatter_('Istanbul', istb,0.5,0.91,15,12)
xl,xh,yl,yh = map_range(istb)
istb_map = import_img('https://github.com/570558305/dmafinal/blob/main/istb.png?raw=true')#po

bw_img = istb_map.convert('L')
#plt.imshow(bw_img, cmap = 'gray')

ax.imshow(bw_img,extent=[xl-0.45 ,xh+0.7 ,yl-0.65  ,yh+0.3 ], cmap = 'gray')
```

Out[156]: <matplotlib.image.AxesImage at 0x12ac48f40>



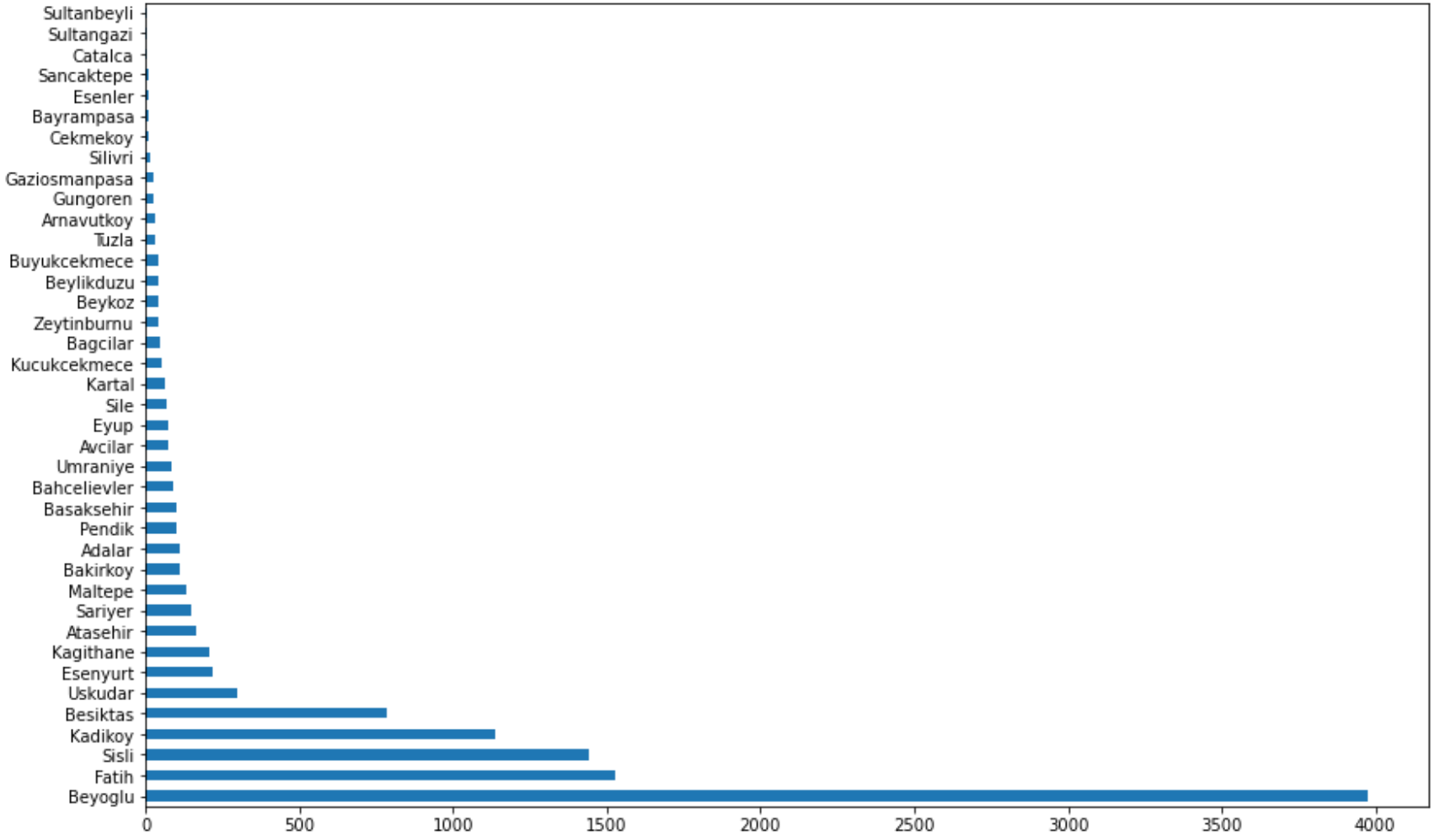
Note that the colder the color, the cheaper the price, so the warmer the color, the higher the price. the red spot shows that the prices are higher.

1.2 Neighborhood

1.21 The distribution of Airbnbs in each district

In [157]:
```
CountStatus = pd.value_counts(istb['neighbourhood'].values, sort=True)
CountStatus.plot.barh(figsize=(14,9))
```

Out[157]: <AxesSubplot:>



Beyoglu, Fatih, and Sisli have the most airbnbs located.

Sultanbeyli, Sultangazi, and Catalca have the least airbnbs.

In [158]:
```
CountStatus.head(10)
```

Out[158]:
```
Beyoglu      3976
Fatih        1525
Sisli        1444
Kadikoy      1137
Besiktas      783
Uskudar       296
Esenyurt      216
Kagithane     210
Atasehir      163
Sariyer       146
dtype: int64
```

In [159]:
```
istb['room_type'].unique()
```

Out[159]:
```
array(['Entire home/apt', 'Private room', 'Shared room', 'Hotel room'],
      dtype=object)
```

In [160]:
```
col = ['city']+list(istb['room_type'].unique())
review = pd.DataFrame(columns = col)
g = ['Atasehir', 'Besiktas', 'Beyoglu', 'Esenyurt', 'Fatih', 'Kadikoy',
       'Kagithane', 'Sariyer', 'Sisli', 'Uskudar'] #The top ten cities with the most number of airbnbs
review['city'] = g
```

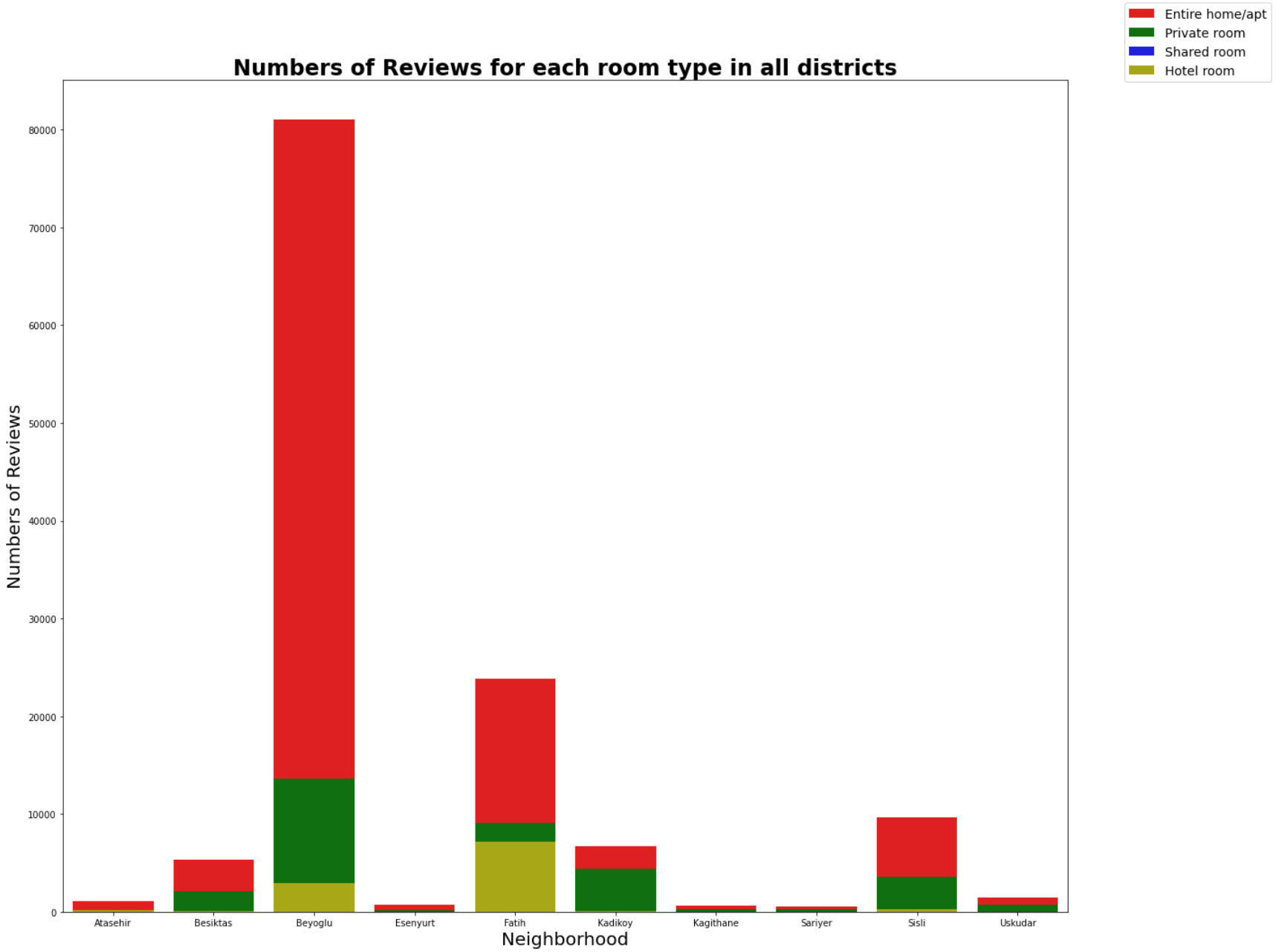
In [161]:
```
d1 = istb.loc[istb['neighbourhood'].isin(g),:].groupby(['neighbourhood','room_type'])[['number_of_reviev
d2 = d1.reset_index()
for c in review.columns.values[1:]:
    lst = d2.loc[d2['room_type']==c,['number_of_reviews']].values
    review[c] = lst
review
```

Out[161]:

	city	Entire home/apt	Private room	Shared room	Hotel room
0	Atasehir	1113	80	2	196
1	Besiktas	5301	2115	6	97
2	Beyoglu	81066	13573	442	2902
3	Esenyurt	683	155	1	2
4	Fatih	23795	9062	188	7164
5	Kadikoy	6711	4382	58	47
6	Kagithane	599	261	1	14
7	Sariyer	578	242	11	6
8	Sisli	9644	3555	142	239
9	Uskudar	1434	726	3	2

In [162]:
```
fig, ax = plt.subplots(figsize=(20,17))

ax =sns.barplot(data=review, x="city", y='Entire home/apt',color='r',label = 'Entire home/apt')
ax =sns.barplot(data=review, x="city", y='Private room',color='g',label ='Private room' )
ax =sns.barplot(data=review, x="city", y='Shared room',color='b',label = 'Shared room')
ax =sns.barplot(data=review, x="city", y='Hotel room',color='y', label = 'Hotel room')
ax.set_xlabel('Neighborhood',fontsize=20)
ax.set_ylabel('Numbers of Reviews',fontsize=20)
ax.set_title('Numbers of Reviews for each room type in all districts',fontweight='bold',fontsize=24,)
ax.legend(bbox_to_anchor=(1.05, 1.1), loc=2, fontsize='14')
```

Out[162]:
```
<matplotlib.legend.Legend at 0x12a46da90>
```



Among the top ten cities with the most number of Airbnbs:

The number of Reviews indicates the popularity of the roomtype and neighborhood.

Roomtype: shared room is the least popular, entire home/apt is the most popular.

Neighborhood: Sisli is the neighborhood with the most reviews, which means it is really popular. The least popular neighborhood: Kadikoy, Esenyurt, Kagithane

1.3 Host

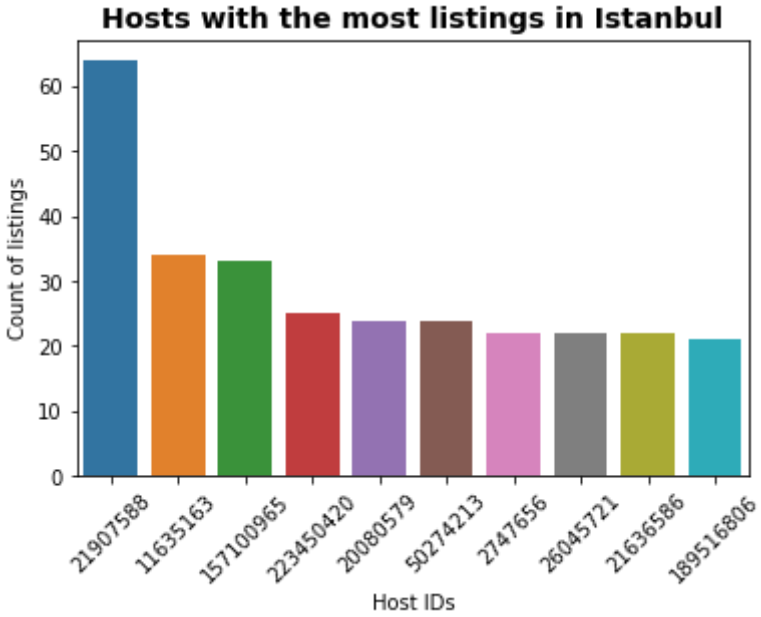
In [ ]:

In [163]:
```
istb_top_host = istb['host_id'].value_counts()[:10]
```

In [164]:
```
ax = sns.barplot(istb_top_host.index, istb_top_host.values,order=istb_top_host.index)
ax.set_xticklabels(ax.get_xticklabels (),rotation=45)
ax.set_title('Hosts with the most listings in Istanbul',size=14,fontweight='bold')
ax.set_ylabel('Count of listings')
ax.set_xlabel('Host IDs')
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

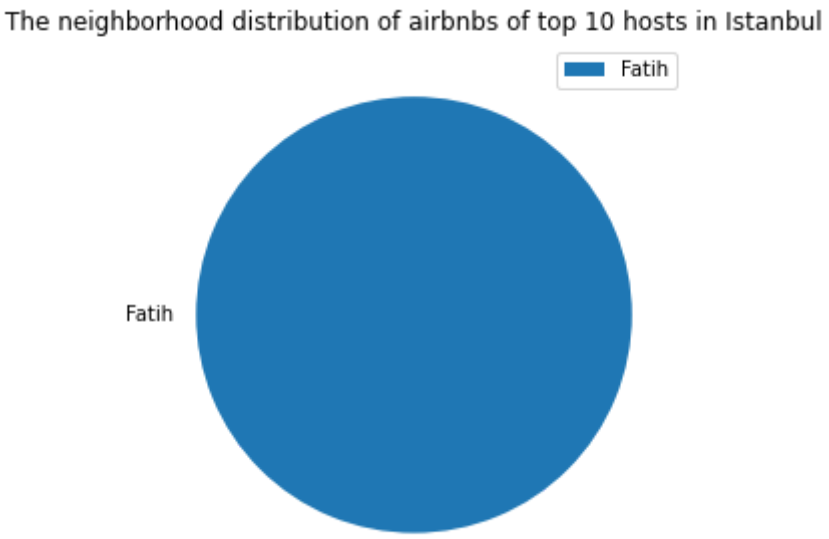
Out[164]:
```
Text(0.5, 0, 'Host IDs')
```



In [165]:
```
x = istb['host_id'].value_counts().nlargest(n=10).index.values
host_219 = istb.loc[istb['host_id'].isin(x),:]
host_219 = istb.loc[istb['host_id'] == 21907588,:]
```

In [166]:
```
host_n = host_219.groupby(['neighbourhood'])[['neighbourhood']].count()
fix,ax= plt.subplots(figsize = (7,5))
host_n.plot(kind= 'pie', x = 'neighbourhood', ax=ax, subplots = True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.legend(bbox_to_anchor=(1, 1))
plt.title('The neighborhood distribution of airbnbs of top 10 hosts in Istanbul')
```

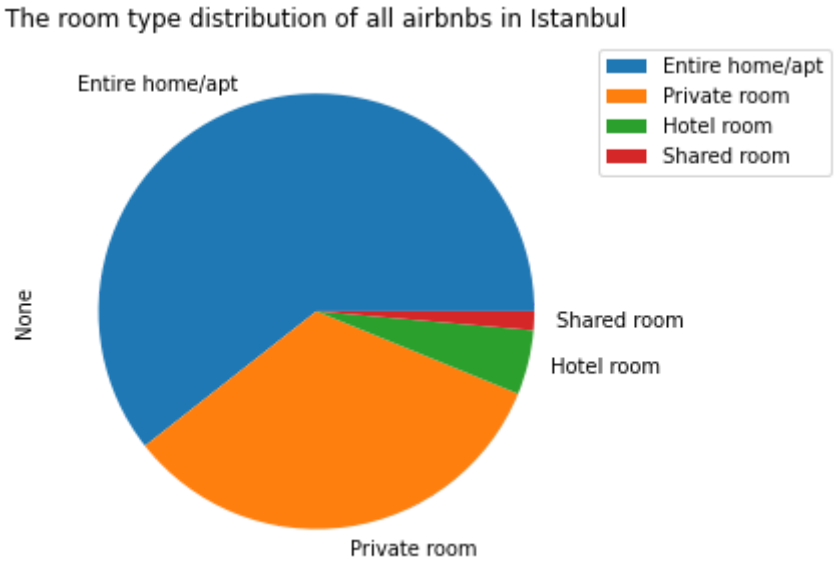
Out[166]:
```
Text(0.5, 1.0, 'The neighborhood distribution of airbnbs of top 10 hosts in Istanbul')
```

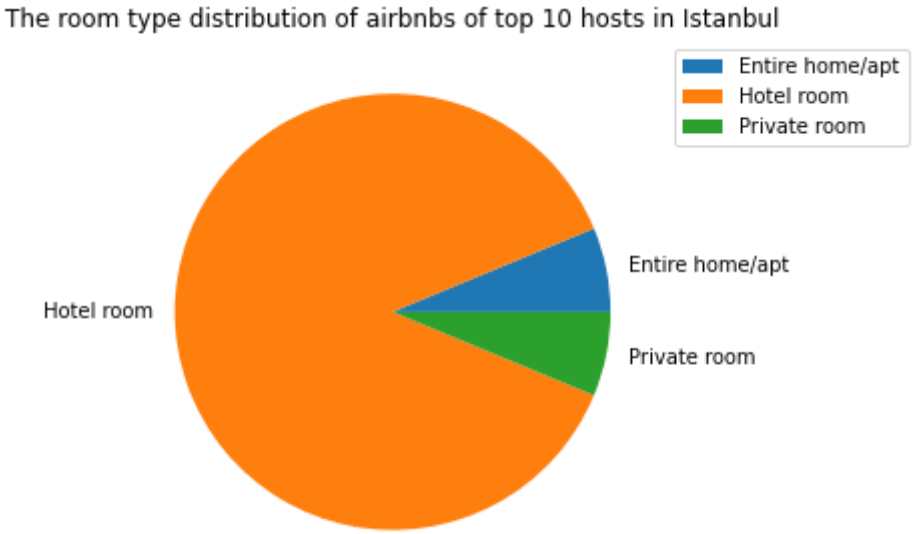


All the top hosts' airbnb rooms are in Fatih

In [167]:
```
CountStatus = pd.value_counts(istb['room_type'].values, sort=True)
CountStatus.plot.pie(figsize = (7,5))
host_is = host_219.groupby(['room_type'])[['room_type']].count()
plt.legend(bbox_to_anchor = (1,1))
plt.title('The room type distribution of all airbnbs in Istanbul')
fix,ax = plt.subplots(figsize = (7,5))
host_is.plot(kind = 'pie',x = "room_type", ax=ax, subplots = True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.legend(bbox_to_anchor = (1,1))
plt.title('The room type distribution of airbnbs of top 10 hosts in Istanbul')
```

Out[167]: Text(0.5, 1.0, 'The room type distribution of airbnbs of top 10 hosts in Istanbul')



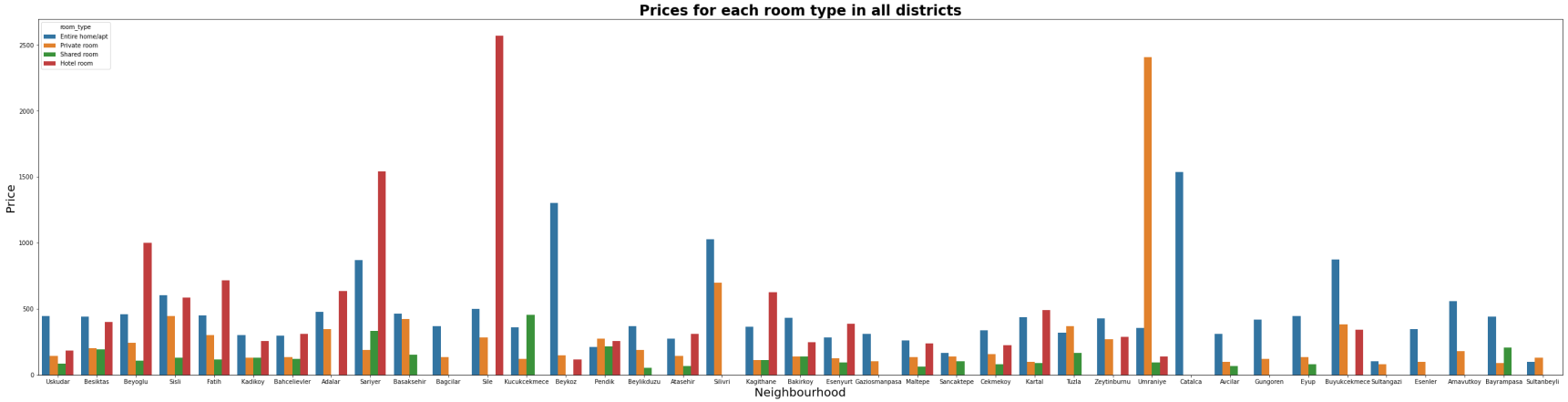


The top hosts tends to have more hotel rooms.

1.4 Prices

In [168]:
```
fig, ax = plt.subplots(figsize=(46,11))
ax =sns.barplot(data=istb, x="neighbourhood", y ='price', hue="room_type",ci= None)
ax.set_xlabel('Neighbourhood',fontsize=20)
ax.set_ylabel('Price',fontsize=20)
ax.set_title('Prices for each room type in all districts',fontweight='bold',fontsize=24)
```

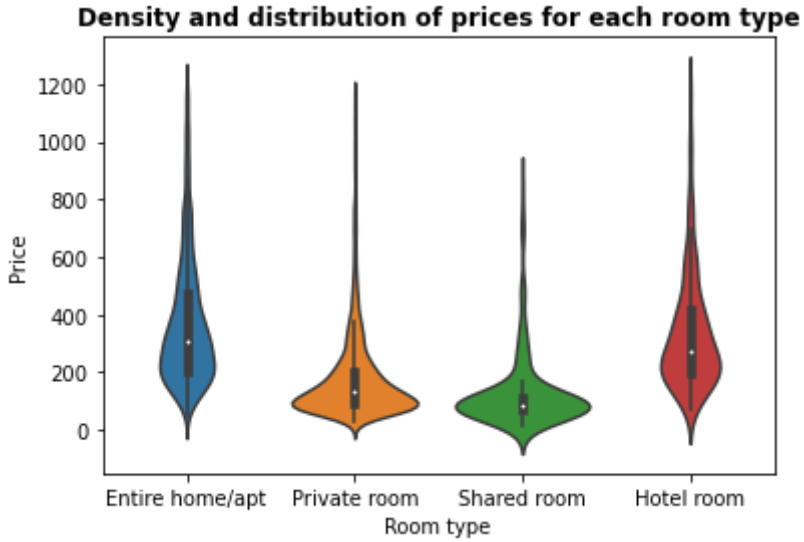
Out[168]:
```
Text(0.5, 1.0, 'Prices for each room type in all districts')
```



The prices of Hotel room in Uskudar is largely higher than other rooms

In [169]:
```
ax = sns.violinplot(data=istb[istb.price < 1200] ,x='room_type',y='price')

ax.set_xlabel('Room type')
ax.set_ylabel('Price')
ax.set_title('Density and distribution of prices for each room type',fontweight='bold')
```

Out[169]: Text(0.5, 1.0, 'Density and distribution of prices for each room type')



2. Descriptive Data Analysis

2.1Data Overview

In [170]:
```
istb = pd.read_csv('https://raw.githubusercontent.com/AmysnL/data-bootcamp-final-project/main/Istanbul.c
```

In [171]:
```
istb.describe()
```

Out[171]:

	id	host_id	neighbourhood_group	latitude	longitude	price	minimum_nights	number_of_reviews
count	2.372800e+04	2.372800e+04	0.0	23728.000000	23728.000000	23728.000000	23728.000000	23728.000000
mean	2.913711e+07	1.493973e+08	NaN	41.028416	28.982111	484.643248	4.525202	7.870828
std	1.305964e+07	1.155452e+08	NaN	0.045713	0.127503	1973.884093	27.614191	23.229127
min	4.826000e+03	6.603000e+03	NaN	40.813960	28.019010	0.000000	1.000000	0.000000
25%	2.101860e+07	3.285440e+07	NaN	41.005120	28.973210	137.000000	1.000000	0.000000
50%	3.398637e+07	1.477727e+08	NaN	41.031850	28.983485	247.000000	1.000000	0.000000
75%	3.965902e+07	2.588145e+08	NaN	41.048530	29.020050	446.000000	3.000000	4.000000
max	4.397093e+07	3.522041e+08	NaN	41.479030	29.907780	76922.000000	1125.000000	345.000000

In [172]:
```
istb.isnull().sum()
```

Out[172]:
```
id                                  0
name                               54
host_id                             0
host_name                           1
neighbourhood_group             23728
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     12375
reviews_per_month               12375
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

In [173]:
```
istb.drop(columns= ['neighbourhood_group'], axis = 1, inplace = True)
istb
```

Out[173]:

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_c
0	4826	The Place	6603	Kaan	Uskudar	41.05650	29.05367	Entire home/apt	720	1	
1	20815	The Bosphorus from The Comfy Hill	78838	Gülder	Besiktas	41.06984	29.04545	Entire home/apt	816	365	
2	27271	LOVELY APT. IN PERFECT LOCATION	117026	Mutlu	Beyoglu	41.03254	28.98153	Entire home/apt	233	30	
3	28277	Duplex Apartment with Terrace	121607	Alen	Sisli	41.04471	28.98567	Hotel room	761	3	
4	28318	Cosy home overlooking Bosphorus	121721	Aydin	Sariyer	41.09048	29.05559	Entire home/apt	823	3	
...	...	...	...	...	...	...	...	...	...	...	...
23723	43963636	Avcılarda özel oda	297895734	Aykhan	Avcilar	40.97870	28.72668	Private room	171	1	
23724	43966333	1+1 LUXURY RESIDENCEIN MASLAK HYGIENE CERTIFIC...	69089629	Metin	Sisli	41.11798	29.00886	Entire home/apt	597	1	
23725	43966442	The Rooms In Beyoğlu	286090194	Seyfulla	Beyoglu	41.03839	28.98831	Private room	144	1	
23726	43967082	Istanbul dream apartement	288314755	Adil	Esenyurt	41.01065	28.67427	Entire home/apt	603	2	
23727	43970934	Şile sahilinin eşşiz manzarası sizleri bekliyor	352204054	Engin	Sile	41.17426	29.60997	Private room	103	1	

23728 rows × 15 columns

In [ ]:

2.2 Descriptive Data Analysis

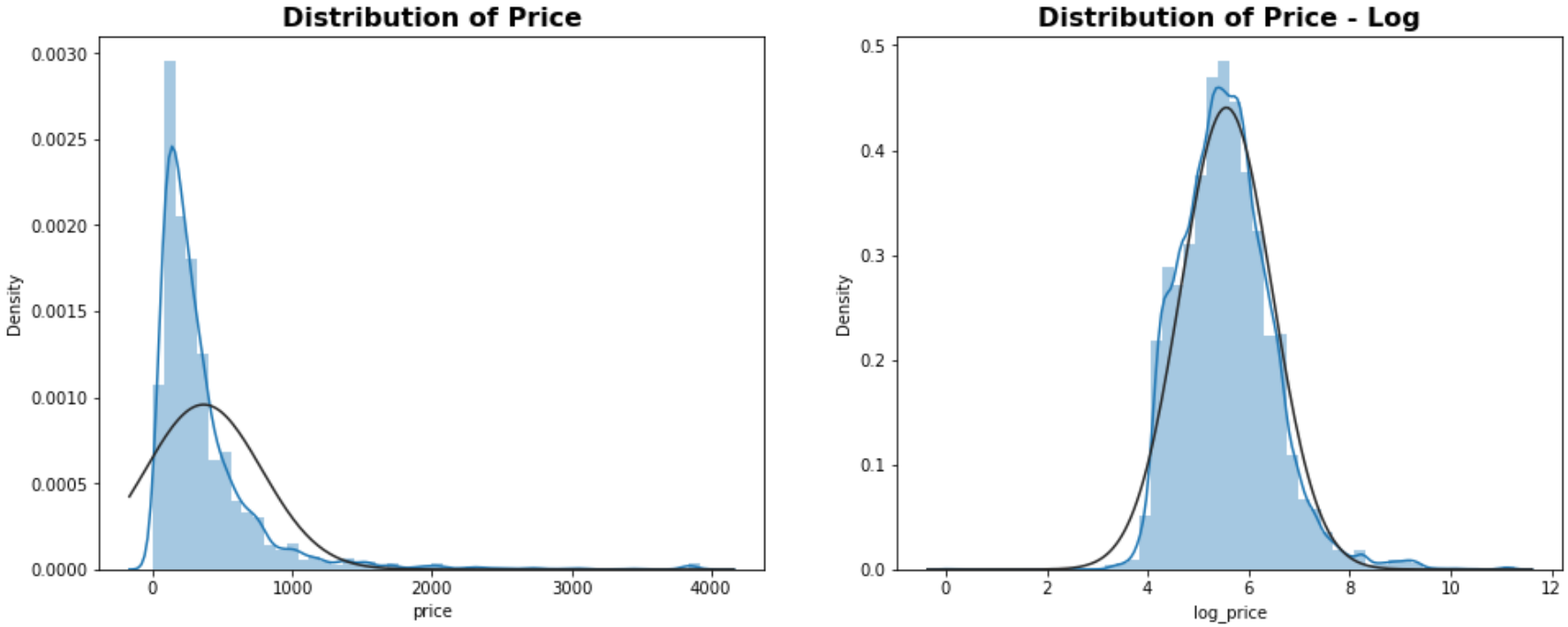
In [174]:
```
# encode str
en_istb = istb.copy()
en_istb['neighbourhood'] = en_istb['neighbourhood'].astype('category').cat.codes
en_istb['room_type'] = en_istb['room_type'].astype("category").cat.codes
mean = en_istb['reviews_per_month'].mean()
en_istb['reviews_per_month'].fillna(mean, inplace=True) ####
en_istb['log_price'] = np.log(en_istb.price+1)  #####
en_istb = en_istb.drop(columns=['id' ,'host_id','price']) # delete price column
en_istb.isnull().sum()
```

Out[174]:
```
name                                54
host_name                            1
neighbourhood                        0
latitude                             0
longitude                            0
room_type                            0
minimum_nights                       0
number_of_reviews                    0
last_review                      12375
reviews_per_month                    0
calculated_host_listings_count       0
availability_365                     0
log_price                            0
dtype: int64
```

In [175]:
```
fig,ax = plt.subplots(1,2,figsize = (16,6))
sns.distplot(istb['price'].loc[istb['price'] <= 4000], fit = norm, ax = ax[0])
ax[0].set_title("Distribution of Price", size = 16,weight = 'bold')
sns.distplot(en_istb['log_price'],fit=norm , ax = ax[1])
ax[1].set_title("Distribution of Price - Log", size = 16,weight = 'bold')
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```

Out[175]: Text(0.5, 1.0, 'Distribution of Price - Log')



The log price distribution is approximately a normal distribution of a mean of approximately 5.5

2.3 Correlation

In [176]:
```
listings_istb = istb
```

In [177]:
```
# Change data type
listings_price_istb = listings_istb[['price','minimum_nights','number_of_reviews','reviews_per_month','c
listings_price_istb[:5]
```

Out[177]:

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365	latitude	longitude
0	720	1	1	0.01	1	365	41.05650	29.05367
1	816	365	41	0.33	2	279	41.06984	29.04545
2	233	30	13	0.19	1	289	41.03254	28.98153
3	761	3	0	NaN	19	365	41.04471	28.98567
4	823	3	0	NaN	1	88	41.09048	29.05559

This is the correlation matrix showing relationships between different variables.

This will help the host to improve decisions.

In [178]:
```
ure(figsize=(10,10))
 = sns.diverging_palette(20, 220, n=256)
stings_price_istb.corr(method='pearson')
tmap(corr, annot=True, fmt=".2f", cmap="PRGn", vmax=.3, center=0,square=True, linewidths=1, cbar_kws={"sh
le("Correlation Matrix",size=12, weight='bold')
```

Out[178]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```

3. Regression

3.1 Linear Regression

In [179]:
```
reg_price = smf.ols('price ~ room_type + availability_365 + number_of_reviews + calculated_host_listings
                    ,data=istb).fit()
print(reg_price.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.006
Model:                            OLS   Adj. R-squared:                  0.006
Method:                 Least Squares   F-statistic:                     10.16
Date:                Fri, 18 Dec 2020   Prob (F-statistic):           9.72e-13
Time:                        10:29:12   Log-Likelihood:            -1.0027e+05
No. Observations:               11353   AIC:                         2.006e+05
Df Residuals:                   11345   BIC:                         2.006e+05
Df Model:                           7
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                      451.5023     36.982     12.209      0.000     379.011     523.994
room_type[T.Hotel room]        254.9871     76.677      3.325      0.001     104.687     405.287
room_type[T.Private room]     -209.6897     34.671     -6.048      0.000    -277.650    -141.729
room_type[T.Shared room]      -346.6551    133.898     -2.589      0.010    -609.119     -84.191
availability_365                 0.0821      0.111      0.737      0.461      -0.136       0.301
number_of_reviews               -0.5454      0.683     -0.799      0.424      -1.883       0.793
calculated_host_listings_count   2.5545      2.080      1.228      0.219      -1.523       6.632
reviews_per_month              -21.1929     23.874     -0.888      0.375     -67.991      25.605
==============================================================================
Omnibus:                    31959.448   Durbin-Watson:                   1.972
Prob(Omnibus):                  0.000   Jarque-Bera (JB):       1091095646.808
Skew:                          37.102   Prob(JB):                         0.00
Kurtosis:                    1519.920   Cond. No.                     2.37e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.37e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From the regressino result above, we can see that room_type[T.Hotel room] , availability_365, calculated_host_listings_count all contributed positively to the price.

room_type[T.Private room], room_type[T.Shared room], number_of_reviews , reviews_per_month have negative contribution.

3.2 Log

In [180]:
```
reg_logprice = smf.ols('log_price ~ neighbourhood +room_type + availability_365 + number_of_reviews + ca
                       ,data= en_sh).fit()
print(reg_logprice.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_price   R-squared:                       0.173
Model:                            OLS   Adj. R-squared:                  0.173
Method:                 Least Squares   F-statistic:                     663.5
Date:                Fri, 18 Dec 2020   Prob (F-statistic):               0.00
Time:                        10:29:12   Log-Likelihood:                -20360.
No. Observations:               18971   AIC:                         4.073e+04
Df Residuals:                   18964   BIC:                         4.079e+04
Df Model:                           6
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                        6.3144      0.018    351.978      0.000       6.279       6.350
neighbourhood                   -0.0004      0.001     -0.270      0.787      -0.003       0.002
room_type                       -0.5863      0.009    -62.623      0.000      -0.605      -0.568
availability_365             -8.852e-05   4.09e-05     -2.166      0.030      -0.000   -8.42e-06
number_of_reviews               -0.0011      0.000     -5.558      0.000      -0.001      -0.001
calculated_host_listings_count  -0.0007      0.000     -3.967      0.000      -0.001      -0.000
reviews_per_month               -0.0157      0.004     -3.583      0.000      -0.024      -0.007
==============================================================================
Omnibus:                     5615.257   Durbin-Watson:                   1.864
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            20693.488
Skew:                           1.456   Prob(JB):                         0.00
Kurtosis:                       7.208   Cond. No.                         965.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

From the regression result above, we can see that neighbourhood, room_type, availability_365,

From the regression result above, we can see that neighbourhood, room_type, availability_365, number_of_reviews, calculated_host_listings_count, and reviews_per_month, all negatively contribute to Price.

Since the P value of neighbourhood is greater than 5%, its impact on price is not significant.

4. Machine Learning

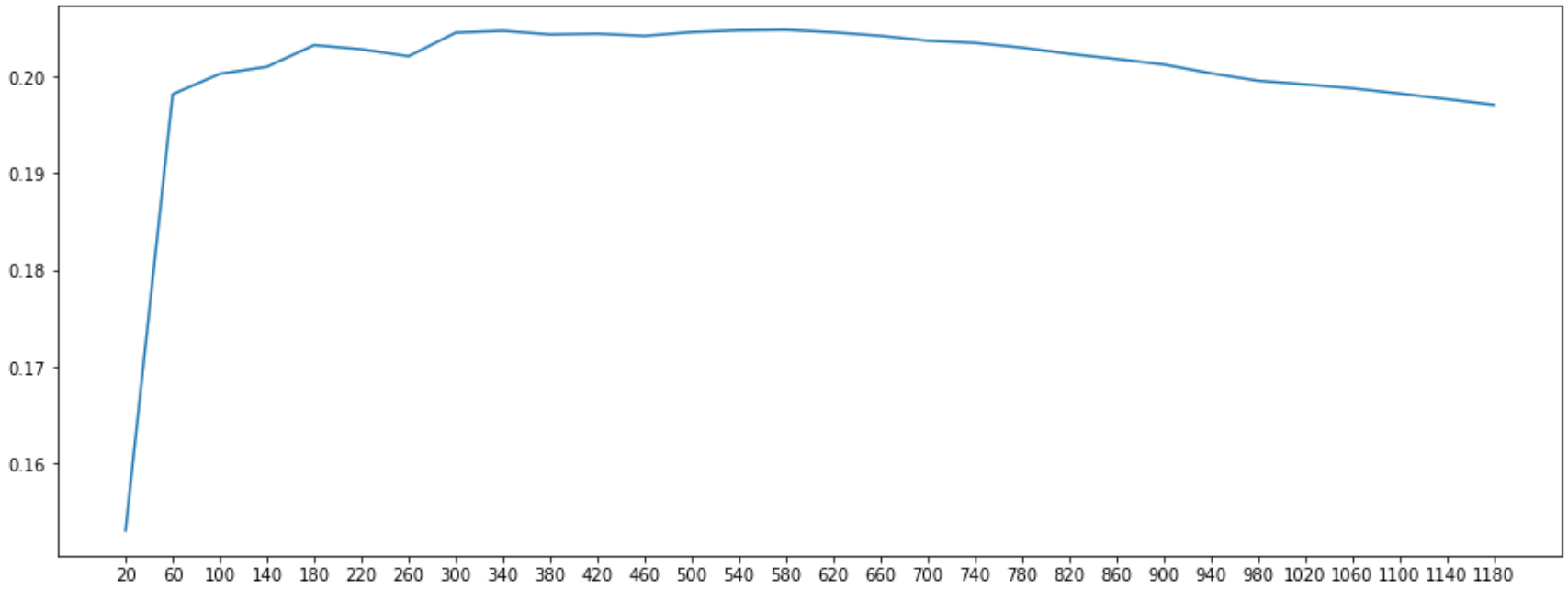
In [181]:
```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(en_istb[['room_type']].values, en_istb[['log_pri
                                                        test_size=0.25, random_state=0)
```

(1) KNN

In [*]:
```
from sklearn.neighbors import KNeighborsRegressor as knn
from sklearn.model_selection import cross_val_score
scores = pd.Series(dtype='float')
for i in range(20,1200,40):
    scores[str(i)] = cross_val_score(knn(n_neighbors=i),X_train1, y_train1,cv=5).mean()
indx = scores.idxmax()
```

In [183]:
```
fig,ax = plt.subplots(figsize=(16,6))
plt.plot(scores.index,scores.values)
```

Out[183]:
```
[<matplotlib.lines.Line2D at 0x129152c40>]
```



In [184]:
```
skl_knn = knn(n_neighbors = int(indx)).fit(X_train1, y_train1)  #####
knn_score1 = skl_knn.score(X_test1,y_test1)
print(knn_score1)
```

```
0.20994304368246441
```

(2) Random Forest

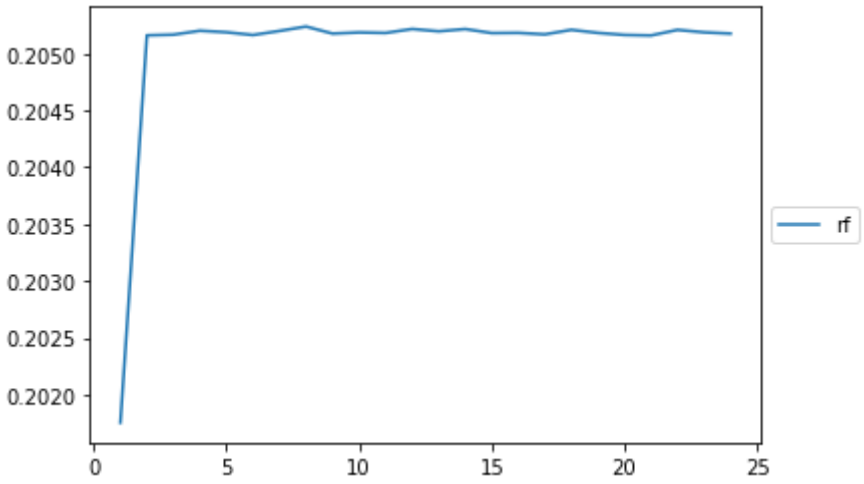
In [185]:
```
from sklearn.ensemble import RandomForestRegressor as rf
cross_val_score(rf(n_estimators = 100, max_depth = 3), X_train1, y_train1.ravel(), cv=5).mean()
```

Out[185]:
```
0.20514967801414014
```

In [186]:
```
scores = pd.DataFrame()
 i in range (1,25):
 cv_scores.loc[i,'rf'] = cross_val_score(rf(n_estimators = 100, max_depth = i), X_train1, y_train1.ravel
```

In [187]:
```
ax = cv_scores.plot()
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
```

Out[187]:
```
<matplotlib.legend.Legend at 0x1290a6640>
```



In [188]:
```
indx = scores.idxmax()
skl_rf = rf(n_estimators=100,max_depth=int(indx)).fit(X_train1, y_train1.ravel())
rf_score1 = skl_rf.score(X_test1,y_test1)
print(rf_score1)
```

```
0.21040392141680886
```

In [189]:
```
score_table = pd.DataFrame({'K Nearest Neighbors':[knn_score1],'Random Forest':[rf_score1]})
score_table = score_table.T
score_table.columns = ['Score']
score_table
```

Out[189]:

	Score
K Nearest Neighbors	0.209943
Random Forest	0.210404

According to the table above, we can use Random Forest as the score for this model for log_price, which is higher than KNN.

However, the score of KNN and Random Forest are relatively low. It could be due to some intangible features such as

the room quality, service friendliness, and the environment, etc

In [ ]:

In [ ]:

Singapore

In [190]:
```
sp = pd.read_csv('https://raw.githubusercontent.com/AmysnL/data-bootcamp-final-project/main/Singapore.cs
sp.head()
```

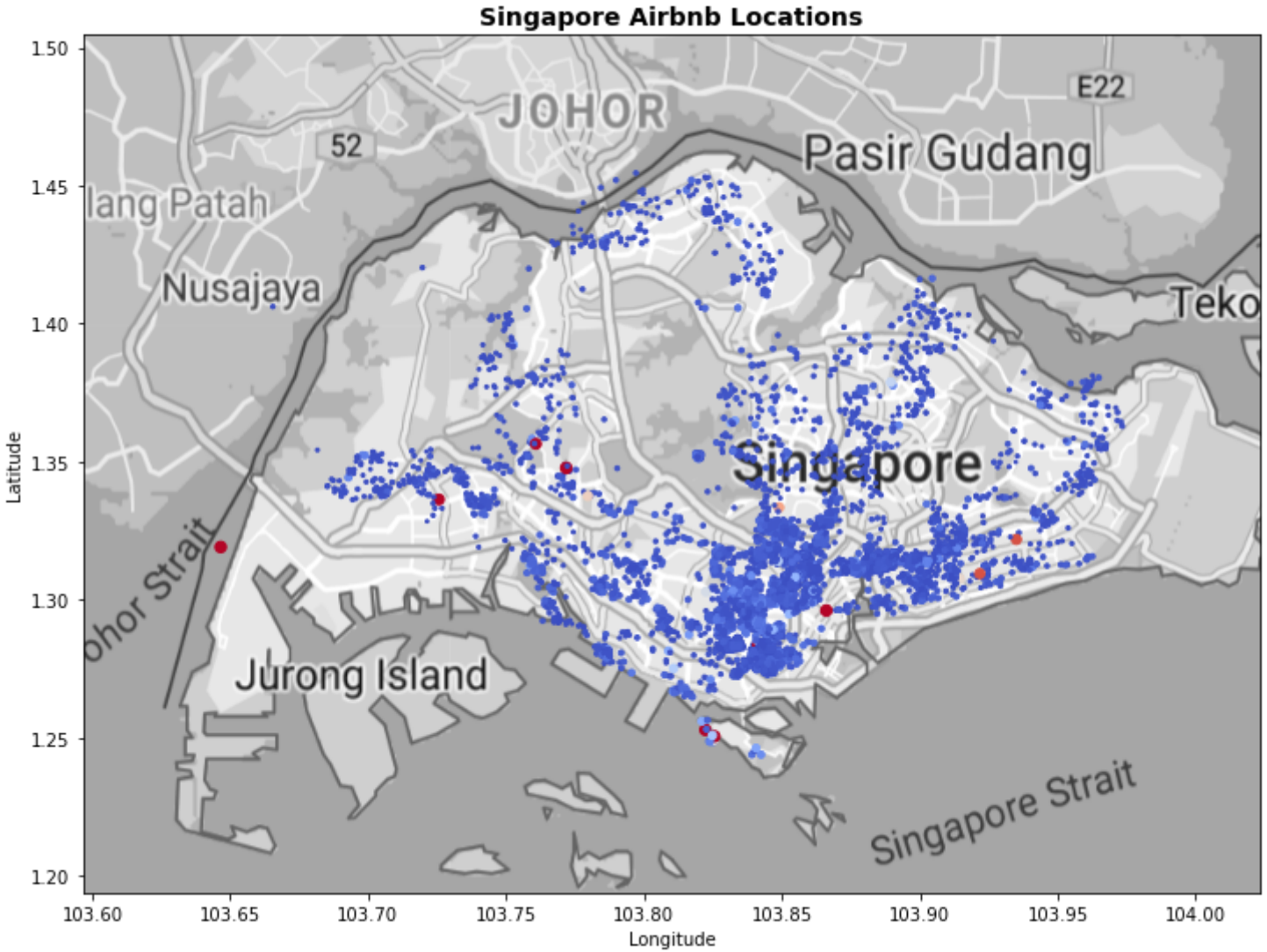
Out[190]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nigh
0	49091	COZICOMFORT LONG TERM STAY ROOM 2	266763	Francesca	North Region	Woodlands	1.44255	103.79580	Private room	83	18
1	50646	Pleasant Room along Bukit Timah	227796	Sujatha	Central Region	Bukit Timah	1.33235	103.78521	Private room	81	9
2	56334	COZICOMFORT	266763	Francesca	North Region	Woodlands	1.44246	103.79667	Private room	69	
3	71609	Ensuite Room (Room 1 & 2) near EXPO	367042	Belinda	East Region	Tampines	1.34541	103.95712	Private room	206	
4	71896	B&B Room 1 near Airport & EXPO	367042	Belinda	East Region	Tampines	1.34567	103.95963	Private room	94	

1.1 Location and availabiity Overview

In [191]:
```
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
fig,ax = location('Singapore', sp,0.4,1.8,12,12)
xl,xh,yl,yh = map_range(sp)
sp_map = import_img('https://github.com/570558305/dmafinal/blob/main/sp.png?raw=true')#po

bw_img = sp_map.convert('L')

ax.imshow(bw_img,extent=[xl-0.19 ,xh+0.27  ,yl-0.24 ,yh +0.17 ], cmap='gray')
```

Out[191]:
```
<matplotlib.image.AxesImage at 0x12e1999d0>
```

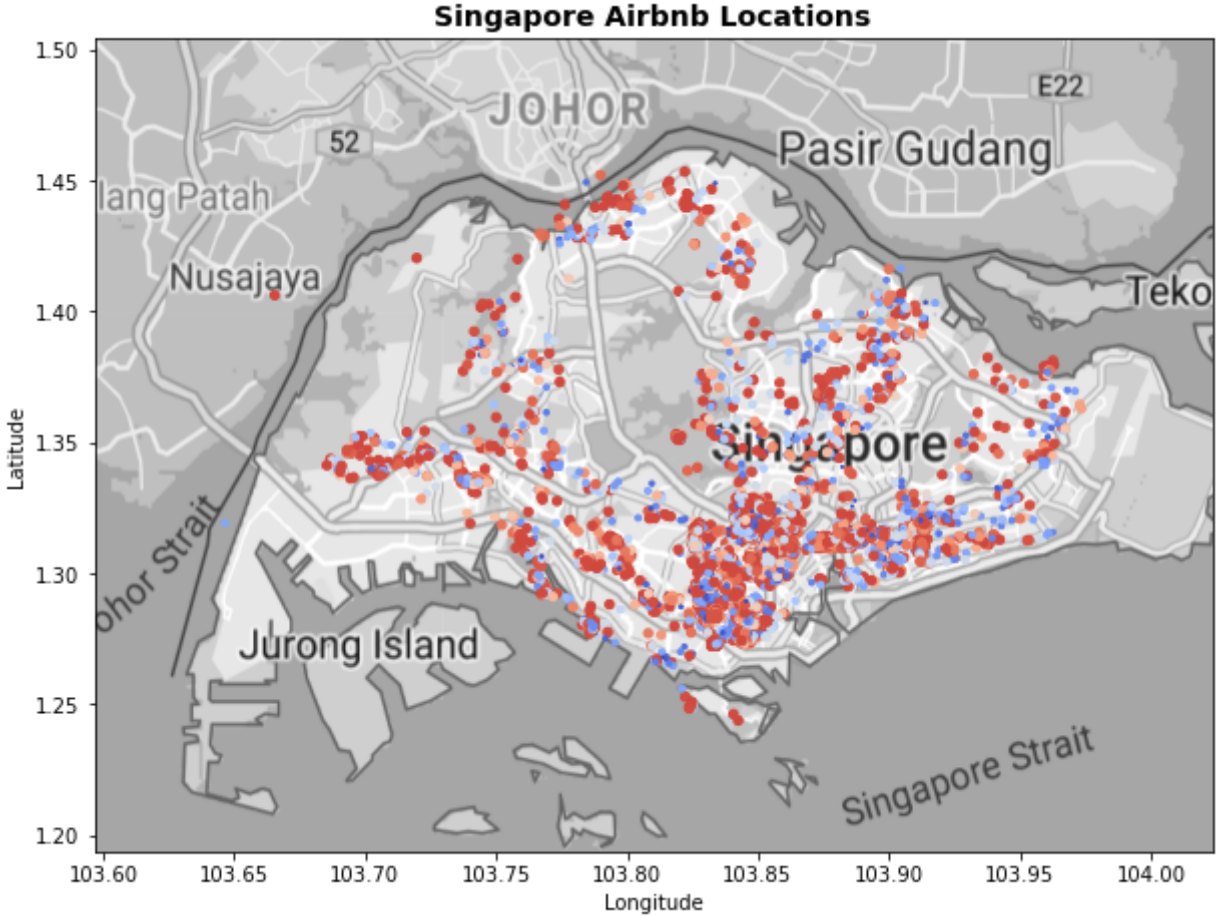


Note that the colder the color, the cheaper the price, the red spot shows that the prices are really high.

The scatter plot here shows that the distribution of Airbnbs located in Singapore.

In [192]:
```
fig,ax = scatter_('Singapore', sp,0.5,0.92,10,10)
xl,xh,yl,yh = map_range(sp)
sp_map = import_img('https://github.com/570558305/dmafinal/blob/main/sp.png?raw=true')
bw_img = sp_map.convert('L')
#plt.imshow(bw_img, cmap = 'gray')

ax.imshow(bw_img,extent=[xl-0.19 ,xh+0.27  ,yl-0.24 ,yh +0.17 ], cmap='gray')
```

Out[192]: <matplotlib.image.AxesImage at 0x131b98eb0>



The scatter plot shows the availability of airbnbs in Singapore, the warmer the color, there are more available rooms at the airbnb represented by the spot, vice versa.

1.2 Neighborhood

1.21 The distribution of Airbnbs in each district

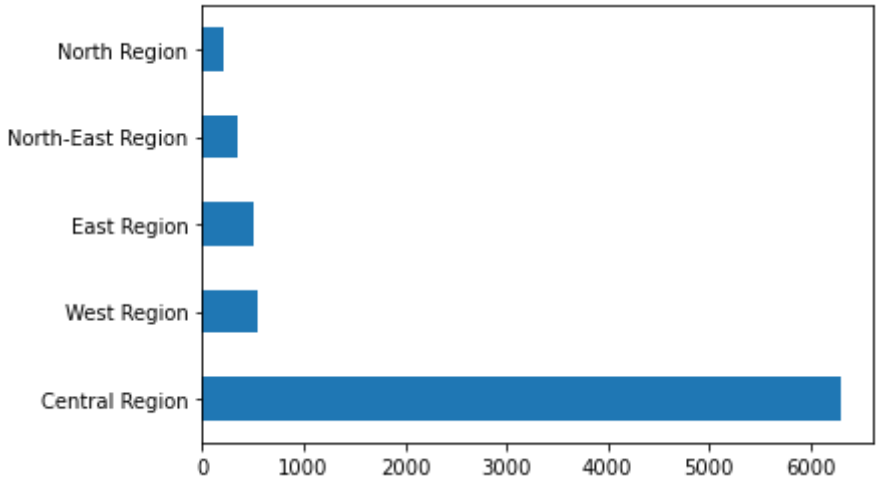
In [193]:
```
pd.DataFrame(sp['neighbourhood_group'].unique())
```

Out[193]:

	0
0	North Region
1	Central Region
2	East Region
3	West Region
4	North-East Region

In [194]:
```
CountStatus = pd.value_counts(sp['neighbourhood_group'].values, sort=True)
CountStatus.plot.barh()
```

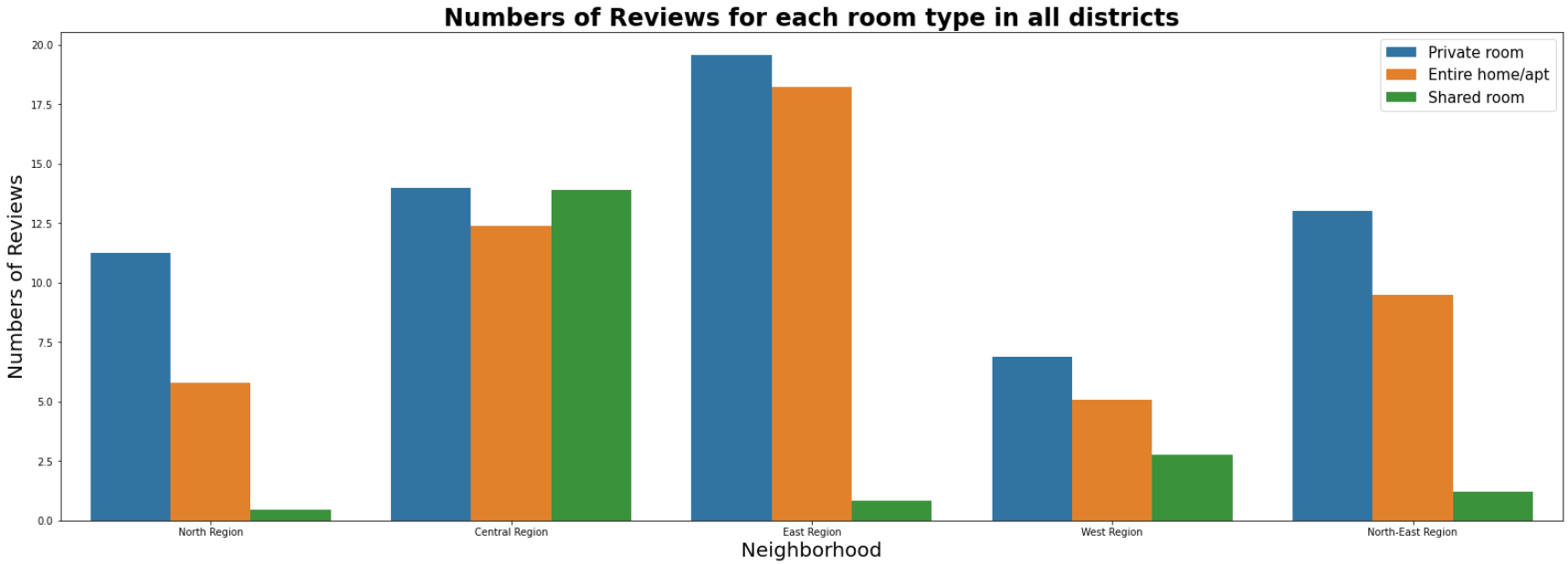
Out[194]: <AxesSubplot:>



Central Region has the most airbnbs located. North Region has the least airbnbs.

In [195]:
```
fig, ax = plt.subplots(figsize=(27,9))
ax =sns.barplot(data=sp, x="neighbourhood_group", y ='number_of_reviews', hue="room_type",ci=None )

ax.set_xlabel('Neighborhood',fontsize=20)
ax.set_ylabel('Numbers of Reviews',fontsize=20)
ax.set_title('Numbers of Reviews for each room type in all districts',fontweight='bold',fontsize=24)
ax.legend(fontsize = 15)
```

Out[195]: <matplotlib.legend.Legend at 0x13351fa90>



The number of Reviews indicates the popularity of the rooms and neighborhood.

Private room is the most popular in all of the neighborhood groups.

For most of the group, the popularity is: Private > Entire > Shared.

Central region is very special. The review of shared room is almost the same with that of private room.

1.3 Host

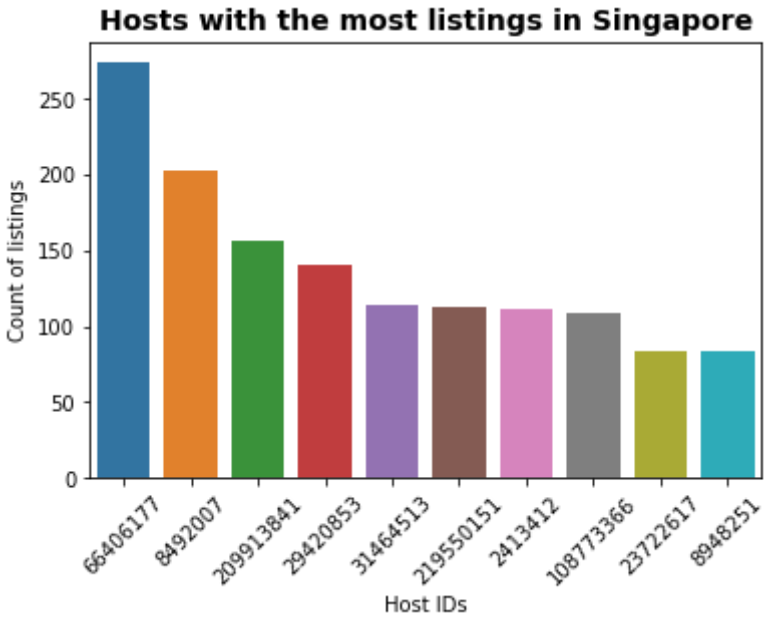
In [196]:
```
top_host_sp = sp['host_id'].value_counts()[:10]
top_host_sp
```

Out[196]:
```
66406177     274
8492007      203
209913841    157
29420853     141
31464513     114
219550151    113
2413412      112
108773366    109
23722617      84
8948251       83
Name: host_id, dtype: int64
```

In [197]:
```
ax = sns.barplot(top_host_sp.index, top_host_sp.values,order=top_host_sp.index)
ax.set_xticklabels(ax.get_xticklabels (),rotation=45)
ax.set_title('Hosts with the most listings in Singapore',size=14,fontweight='bold')
ax.set_ylabel('Count of listings')
ax.set_xlabel('Host IDs')
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

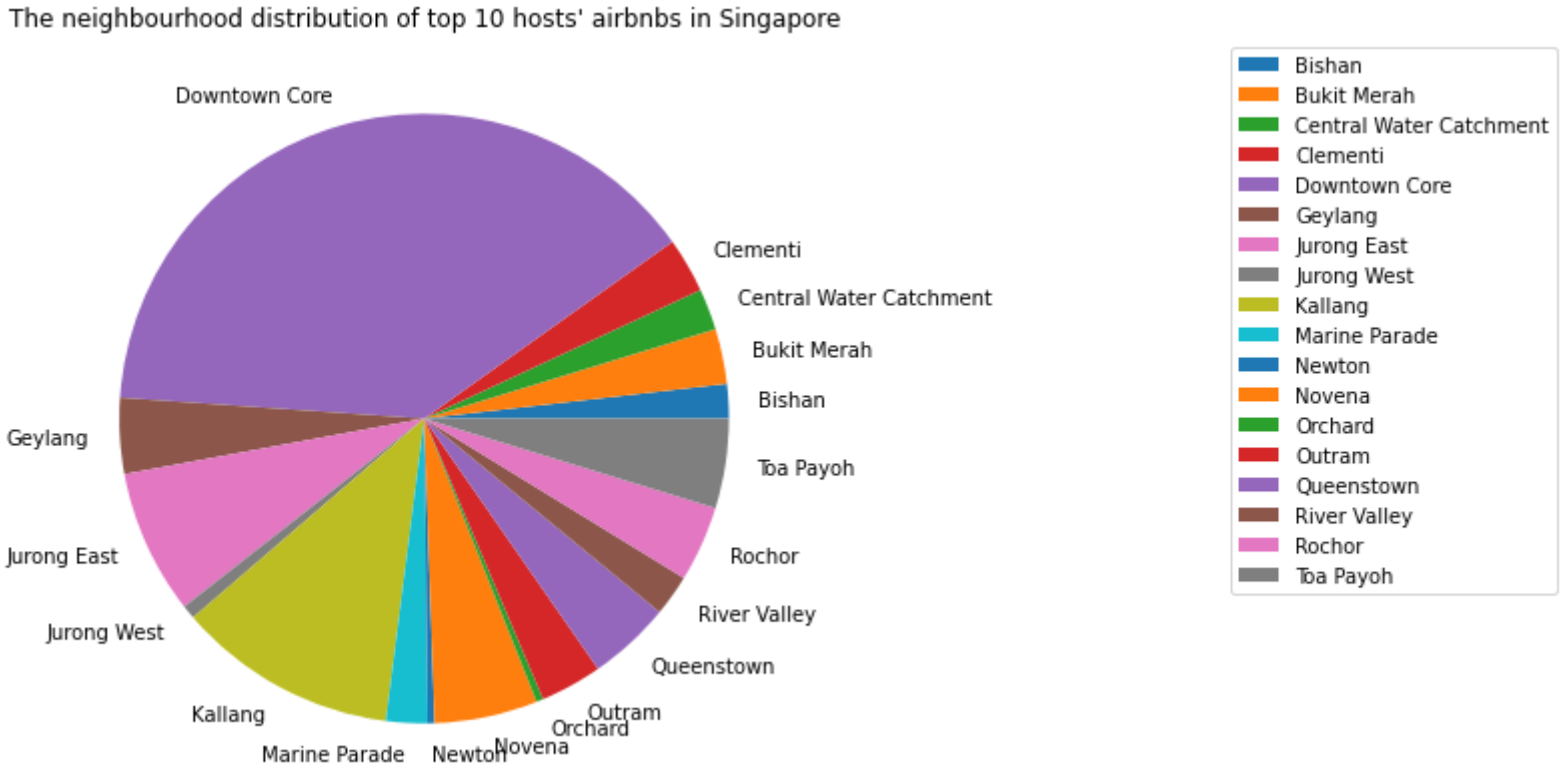
Out[197]:
```
Text(0.5, 0, 'Host IDs')
```



In [198]:
```
x = istb['host_id'].value_counts().nlargest(n=10).index.values
host_664 = sp.loc[sp['host_id'].isin(x),:]
host_664 = sp.loc[sp['host_id'] == 66406177,:]
```

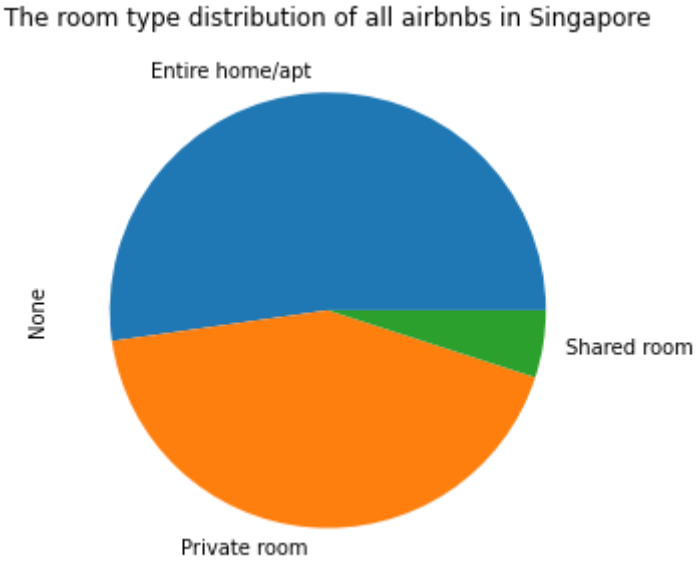
In [199]:
```
host_n = host_664.groupby(['neighbourhood'])[['neighbourhood']].count()
fix,ax = plt.subplots(figsize = (7,7))
host_n.plot(kind='pie',x='neighbourhood',ax=ax,subplots=True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.legend(bbox_to_anchor=(2, 1))
plt.title("The neighbourhood distribution of top 10 hosts' airbnbs in Singapore")
```

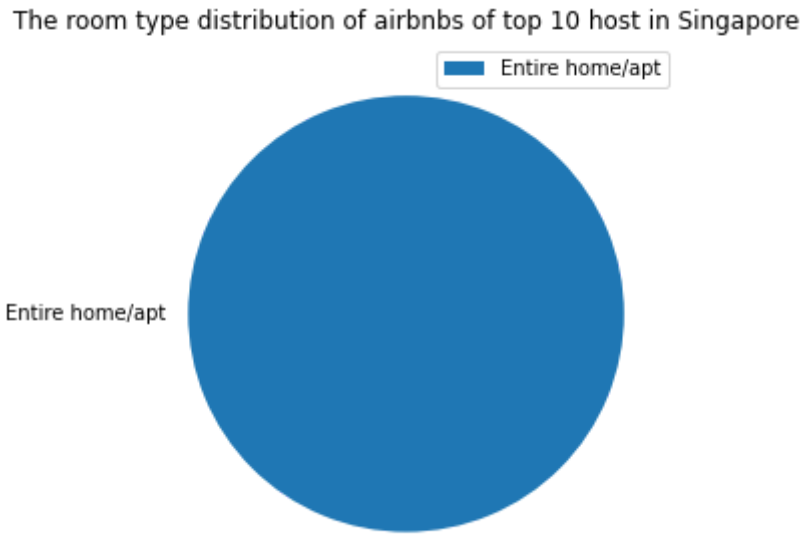
Out[199]:
```
Text(0.5, 1.0, "The neighbourhood distribution of top 10 hosts' airbnbs in Singapore")
```



In [200]:
```
CountStatus = pd.value_counts(sp['room_type'].values, sort=True)
CountStatus.plot.pie(figsize = (7,5))
plt.title('The room type distribution of all airbnbs in Singapore')
x =sp["host_id"].value_counts().nlargest(n=10).index.values
host_664 = sp.loc[sp['host_id'].isin(x) ,:]
host_664 = sp.loc[sp['host_id']==66406177,:]

host_rt = host_664.groupby(['room_type'])[['room_type']].count()
fix,ax = plt.subplots(figsize = (7,5))
host_rt.plot(kind='pie',x='room_type',ax=ax,subplots=True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.legend(bbox_to_anchor=(1, 1))
plt.title('The room type distribution of airbnbs of top 10 host in Singapore')
```

Out[200]:
```
Text(0.5, 1.0, 'The room type distribution of airbnbs of top 10 host in Singapore')
```



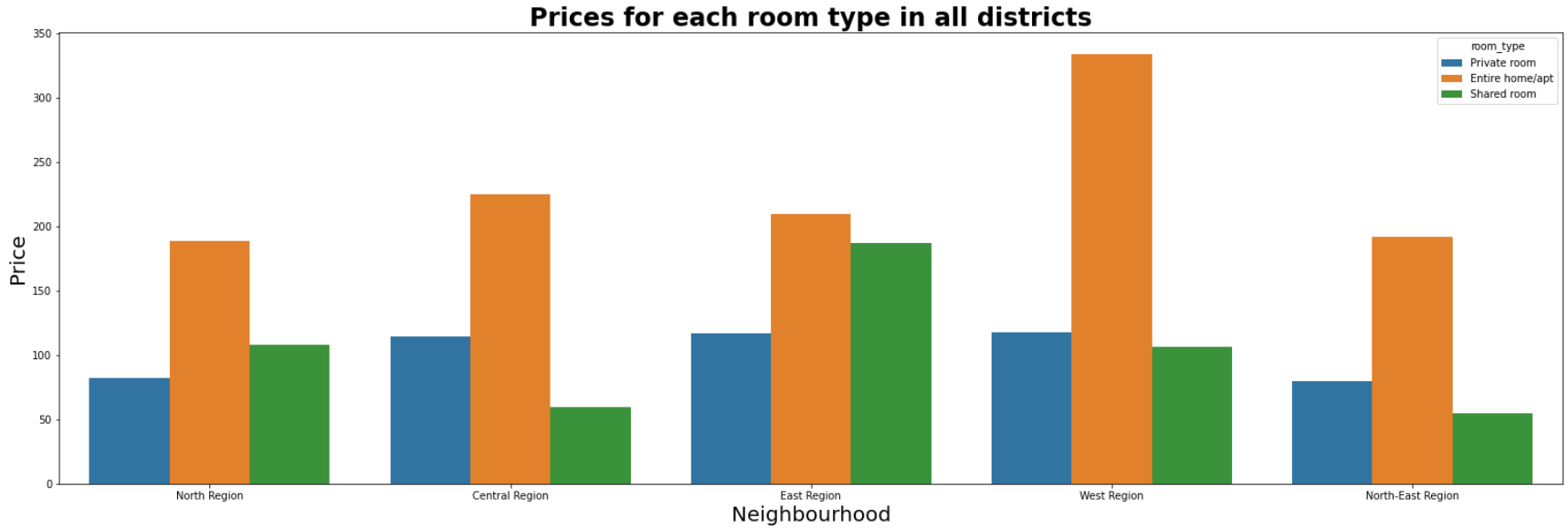


All the top 10 hosts listed the room type: entire home/apartment

1.4 Prices

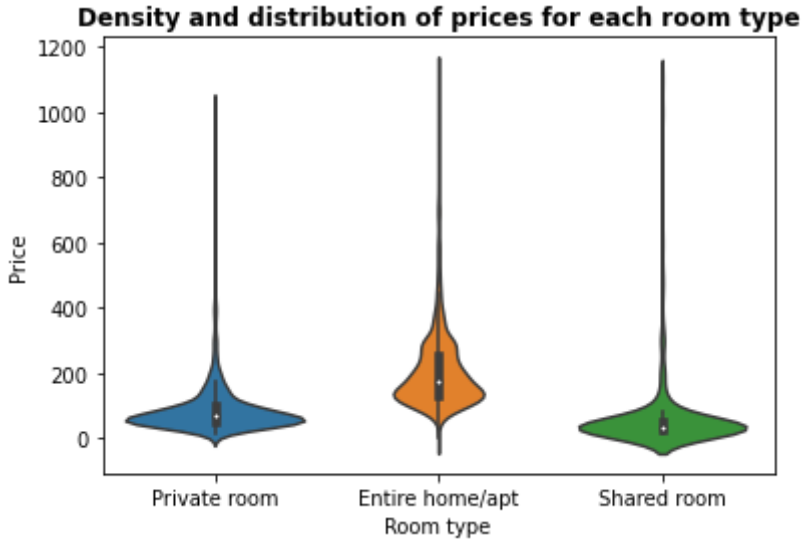
In [201]:
```
fig, ax = plt.subplots(figsize=(26,8))
ax =sns.barplot(data=sp, x="neighbourhood_group", y ='price', hue="room_type",ci=None)
ax.set_xlabel('Neighbourhood',fontsize=20)
ax.set_ylabel('Price',fontsize=20)
ax.set_title('Prices for each room type in all districts',fontweight='bold',fontsize=24)
```

Out[201]:
```
Text(0.5, 1.0, 'Prices for each room type in all districts')
```



In [202]:
```
ax = sns.violinplot(data=sp[sp.price < 1200] ,x='room_type',y='price')

ax.set_xlabel('Room type')
ax.set_ylabel('Price')
ax.set_title('Density and distribution of prices for each room type',fontweight='bold')
```

Out[202]:
```
Text(0.5, 1.0, 'Density and distribution of prices for each room type')
```



2. Descriptive Data Analysis

2.1Data Overview

In [203]:
```
sp.describe()
```

Out[203]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calc
count	7.907000e+03	7.907000e+03	7907.000000	7907.000000	7907.000000	7907.000000	7907.000000	5149.000000	
mean	2.338862e+07	9.114481e+07	1.314192	103.848787	169.332996	17.510054	12.807386	1.043669	
std	1.016416e+07	8.190910e+07	0.030577	0.043675	340.187599	42.094616	29.707746	1.285851	
min	4.909100e+04	2.366600e+04	1.243870	103.646560	0.000000	1.000000	0.000000	0.010000	
25%	1.582180e+07	2.305808e+07	1.295795	103.835825	65.000000	1.000000	0.000000	0.180000	
50%	2.470627e+07	6.344891e+07	1.311030	103.849410	124.000000	3.000000	2.000000	0.550000	
75%	3.234850e+07	1.553811e+08	1.322110	103.872535	199.000000	10.000000	10.000000	1.370000	
max	3.811276e+07	2.885676e+08	1.454590	103.973420	10000.000000	1000.000000	323.000000	13.000000	

In [204]:
```
sp.isnull().sum()
```

Out[204]:
```
id                                 0
name                               2
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
last_review                     2758
reviews_per_month               2758
calculated_host_listings_count     0
availability_365                   0
dtype: int64
```

In [205]:
```
sp.drop(columns=['last_review','reviews_per_month'],axis=1,inplace=True)
sp
```

Out[205]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mini
0	49091	COZICOMFORT LONG TERM STAY ROOM 2	266763	Francesca	North Region	Woodlands	1.44255	103.79580	Private room	83	
1	50646	Pleasant Room along Bukit Timah	227796	Sujatha	Central Region	Bukit Timah	1.33235	103.78521	Private room	81	
2	56334	COZICOMFORT	266763	Francesca	North Region	Woodlands	1.44246	103.79667	Private room	69	
3	71609	Ensuite Room (Room 1 & 2) near EXPO	367042	Belinda	East Region	Tampines	1.34541	103.95712	Private room	206	
4	71896	B&B Room 1 near Airport & EXPO	367042	Belinda	East Region	Tampines	1.34567	103.95963	Private room	94	
...	...	...	...	...	...	...	...	...	...	...	...
7902	38105126	Loft 2 pax near Haw Par / Pasir Panjang. Free ...	278109833	Belle	Central Region	Queenstown	1.27973	103.78751	Entire home/apt	100	
7903	38108273	3bedroom luxury at Orchard	238891646	Neha	Central Region	Tanglin	1.29269	103.82623	Entire home/apt	550	
7904	38109336	[Farrer Park] New City Fringe CBD Mins to MRT	281448565	Mindy	Central Region	Kallang	1.31286	103.85996	Private room	58	
7905	38110493	Cheap Master Room in Central of Singapore	243835202	Huang	Central Region	River Valley	1.29543	103.83801	Private room	56	
7906	38112762	Amazing room with private bathroom walk to Orc...	28788520	Terence	Central Region	River Valley	1.29672	103.83325	Private room	65	

7907 rows × 14 columns

In [ ]:

2.2 Descriptive Data Analysis¶

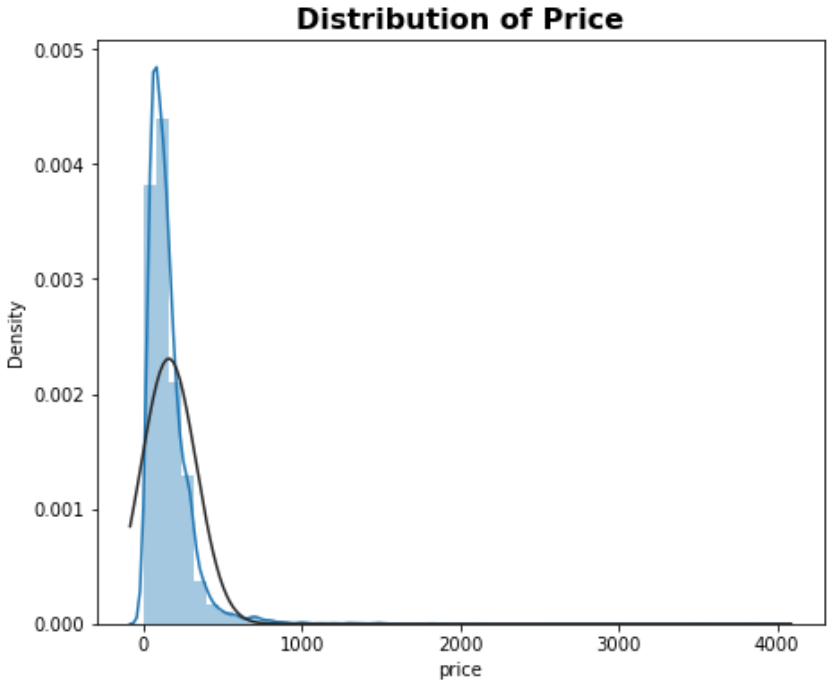
In [206]:
```
# encode str
en_sp = sp.copy()
en_sp['neighbourhood_group'] = en_sp['neighbourhood_group'].astype('category').cat.codes
en_sp['room_type'] = en_sp['room_type'].astype("category").cat.codes
#mean = en_sp['reviews_per_month'].mean()
#en_sp['reviews_per_month'].fillna(mean, inplace=True) ####
en_sp['log_price'] = np.log(en_sp.price+1)  #####
en_sp = en_sp.drop(columns=['id' ,'host_id','price']) # delete price column
en_sp.isnull().sum()
```

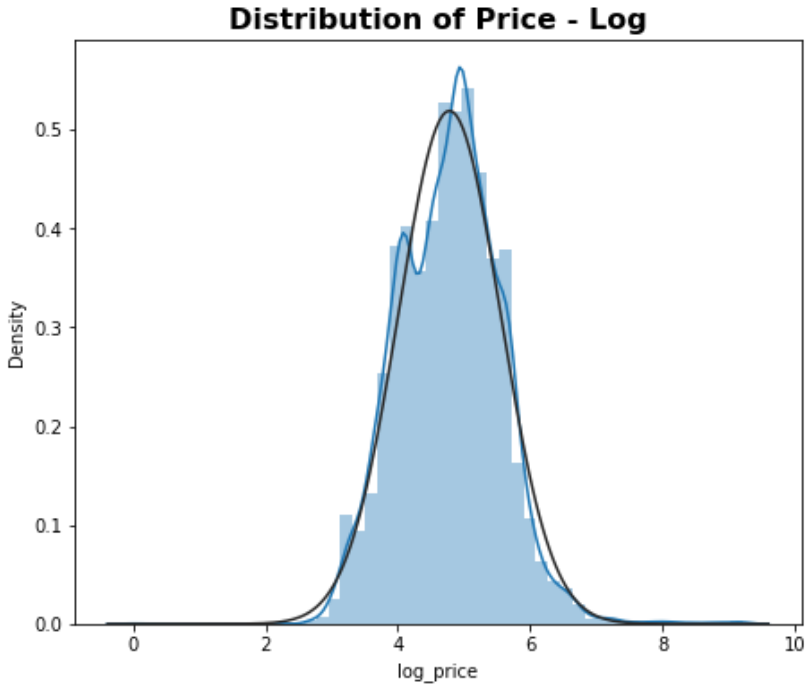
Out[206]:
```
name                              2
host_name                         0
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
minimum_nights                    0
number_of_reviews                 0
calculated_host_listings_count    0
availability_365                  0
log_price                         0
dtype: int64
```

In [207]:
```
fig,ax = plt.subplots(1,2,figsize = (16,6))
sns.distplot(sp['price'].loc[sp['price'] <= 4000], fit = norm, ax = ax[0])
ax[0].set_title("Distribution of Price", size = 16,weight = 'bold')
sns.distplot(en_sp['log_price'],fit=norm , ax = ax[1])
ax[1].set_title("Distribution of Price - Log", size = 16,weight = 'bold')
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```

Out[207]: Text(0.5, 1.0, 'Distribution of Price - Log')





Type *Markdown* and LaTeX: α^2

2.3 Correlation

In [208]:
```
listings_sp = sp
```

In [209]:
```
# Change data type
listings_price_sp = listings_sp[['price','minimum_nights','number_of_reviews','calculated_host_listings_
listings_price_sp[:5]
```

Out[209]:

	price	minimum_nights	number_of_reviews	calculated_host_listings_count	availability_365	latitude	longitude
0	83	180	1	2	365	1.44255	103.79580
1	81	90	18	1	365	1.33235	103.78521
2	69	6	20	2	365	1.44246	103.79667
3	206	1	14	9	353	1.34541	103.95712
4	94	1	22	9	355	1.34567	103.95963

This is the correlation matrix showing relationships between different variables.

This will help the host to improve decisions.

In [210]:
```
plt.figure(figsize=(10,10))
palette = sns.diverging_palette(20, 220, n=256)
corr=listings_price_df.corr(method='pearson')
sns.heatmap(corr, annot=True, fmt=".2f", cmap="BuPu", vmax=.3, center=0,square=True, linewidths=1, cbar_
plt.title("Correlation Matrix",size=12, weight='bold')
```

Out[210]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```

3. Regression

3.1 Linear Regression

In [211]:
```
reg_price = smf.ols('price ~ neighbourhood_group +room_type + availability_365 + number_of_reviews + cal
                    ,data=sp).fit()
print(reg_price.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.039
Model:                            OLS   Adj. R-squared:                  0.038
Method:                 Least Squares   F-statistic:                     35.74
Date:                Fri, 18 Dec 2020   Prob (F-statistic):           1.39e-62
Time:                        10:32:02   Log-Likelihood:                -57155.
No. Observations:                7907   AIC:                         1.143e+05
Df Residuals:                    7897   BIC:                         1.144e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================================
                                           coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------------------------
Intercept                                241.1372      8.452     28.529      0.000     224.568     2
57.706
neighbourhood_group[T.East Region]         0.9008     15.739      0.057      0.954     -29.952
31.753
neighbourhood_group[T.North Region]      -36.8141     23.935     -1.538      0.124     -83.733
10.105
neighbourhood_group[T.North-East Region] -38.3723     18.842     -2.037      0.042     -75.308
-1.437
neighbourhood_group[T.West Region]        29.5014     15.274      1.931      0.053      -0.441
59.443
room_type[T.Private room]               -130.4967      8.541    -15.279      0.000    -147.240    -1
13.754
room_type[T.Shared room]                -181.3890     17.961    -10.099      0.000    -216.598    -1
46.180
availability_365                           0.0735      0.027      2.743      0.006       0.021
0.126
number_of_reviews                         -0.5222      0.128     -4.069      0.000      -0.774
-0.271
calculated_host_listings_count            -0.3711      0.065     -5.722      0.000      -0.498
-0.244
==============================================================================
Omnibus:                    17331.314   Durbin-Watson:                   1.961
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         82019741.679
Skew:                          20.062   Prob(JB):                         0.00
Kurtosis:                     500.336   Cond. No.                     1.67e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.67e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From the regression result above, we can see that

neighbourhood_group[T.East Region], neighbourhood_group[T.West Region], availability_365 contributed positively to price.

neighbourhood_group[T.North Region], neighbourhood_group[T.North-East Region], room_type[T.Private room], room_type[T.Shared room], number_of_reviews, calculated_host_listings_count have negative contribution to price.

3.2 Log

In [212]:
```
reg_logprice = smf.ols('log_price ~ neighbourhood_group +room_type + availability_365 + number_of_reviev
                       ,data= en_sp).fit()
print(reg_logprice.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_price   R-squared:                       0.435
Model:                            OLS   Adj. R-squared:                  0.435
Method:                 Least Squares   F-statistic:                     1216.
Date:                Fri, 18 Dec 2020   Prob (F-statistic):               0.00
Time:                        10:32:02   Log-Likelihood:                -6884.7
No. Observations:                7907   AIC:                         1.378e+04
Df Residuals:                    7901   BIC:                         1.382e+04
Df Model:                           5
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                        5.1818      0.014    364.813      0.000       5.154       5.210
neighbourhood_group             -0.0550      0.006     -9.661      0.000      -0.066      -0.044
room_type                       -0.8493      0.012    -71.523      0.000      -0.873      -0.826
availability_365                 0.0006   4.63e-05     13.140      0.000       0.001       0.001
number_of_reviews               -0.0013      0.000     -5.964      0.000      -0.002      -0.001
calculated_host_listings_count  -0.0009      0.000     -8.037      0.000      -0.001      -0.001
==============================================================================
Omnibus:                     2793.724   Durbin-Watson:                   1.732
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            24390.848
Skew:                           1.445   Prob(JB):                         0.00
Kurtosis:                      11.104   Cond. No.                         631.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

From the regression result above, we can see that neighbourhood_group, room_type, availability_365, number of revies, and calculated host listings count, all negatively contribute to the price.

4. Machine Learning

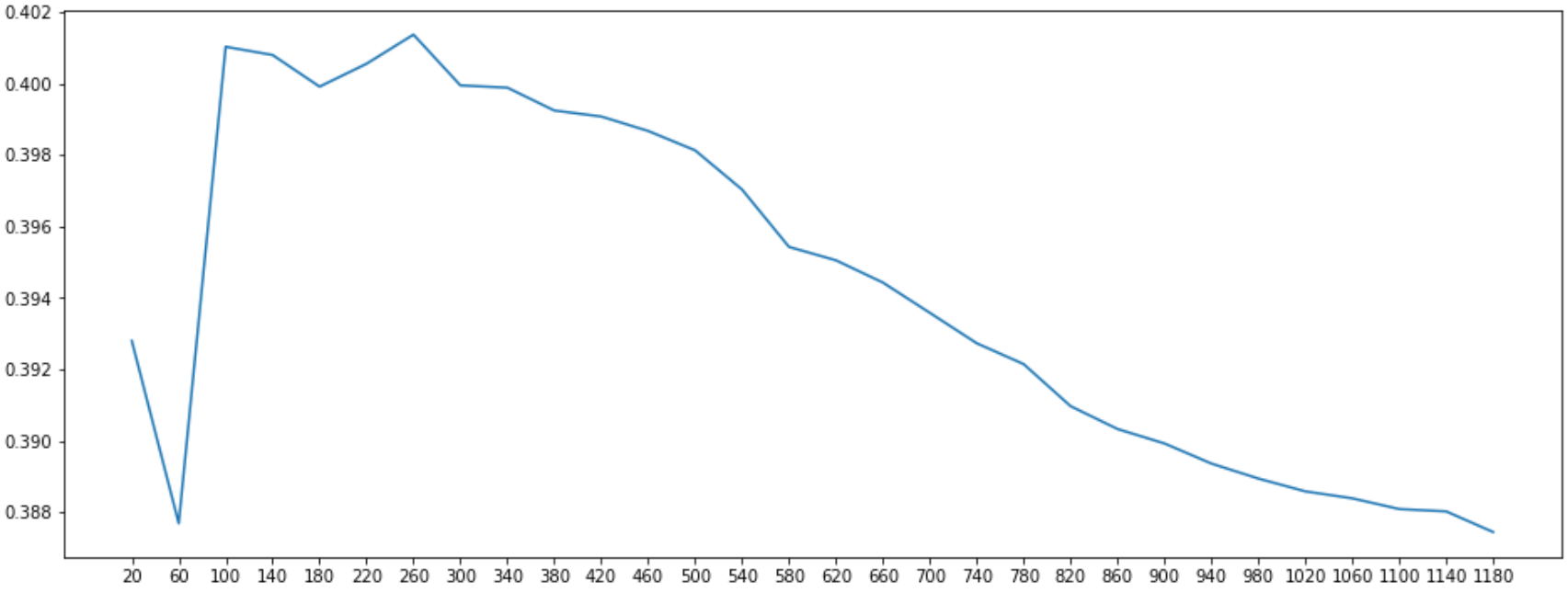
In [213]:
```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(en_sp[['room_type']].values, en_sp[['log_price']
                                                        test_size=0.25, random_state=0)
```

(1) KNN

In [*]:
```
from sklearn.neighbors import KNeighborsRegressor as knn
from sklearn.model_selection import cross_val_score
scores = pd.Series(dtype='float')
for i in range(20,1200,40):
    scores[str(i)] = cross_val_score(knn(n_neighbors=i),X_train1, y_train1,cv=5).mean()
indx = scores.idxmax()
```

In [215]:
```
fig,ax = plt.subplots(figsize=(16,6))
plt.plot(scores.index,scores.values)
```

Out[215]:
```
[<matplotlib.lines.Line2D at 0x12a0ab820>]
```



In [216]:
```
skl_knn = knn(n_neighbors = int(indx)).fit(X_train1, y_train1)  #####
knn_score1 = skl_knn.score(X_test1,y_test1)
print(knn_score1)
```

```
0.44201485921723627
```

(2) Random Forest

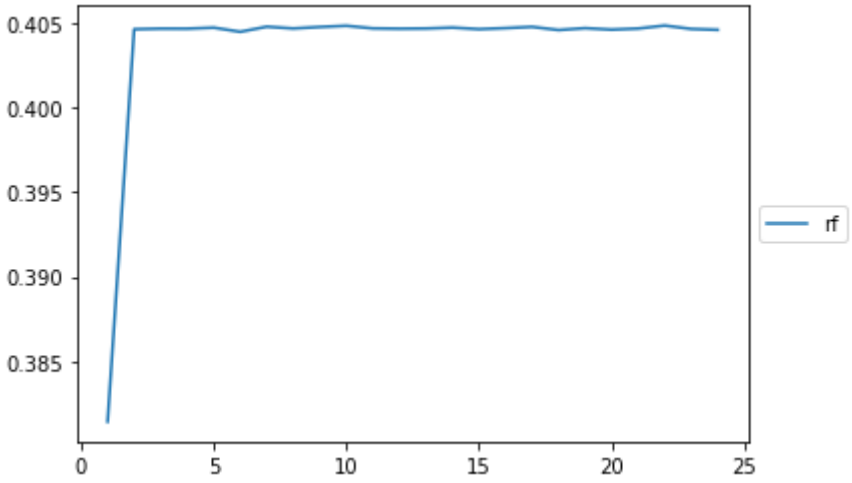
In [217]:
```
from sklearn.ensemble import RandomForestRegressor as rf
cross_val_score(rf(n_estimators = 100, max_depth = 3), X_train1, y_train1.ravel(), cv=5).mean()
```

Out[217]:
```
0.40466273552100684
```

In [218]:
```
cv_scores = pd.DataFrame()
for i in range (1,25):
    cv_scores.loc[i,'rf'] = cross_val_score(rf(n_estimators = 100, max_depth = i), X_train1, y_train1.ra
```

In [219]:
```
ax = cv_scores.plot()
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
```

Out[219]:
```
<matplotlib.legend.Legend at 0x12916b0a0>
```



In [220]:
```
indx = scores.idxmax()
skl_rf = rf(n_estimators=100,max_depth=int(indx)).fit(X_train1, y_train1.ravel())
rf_score1 = skl_rf.score(X_test1,y_test1)
print(rf_score1)
```

```
0.44717965377871616
```

In [221]:
```
score_table = pd.DataFrame({'K Nearest Neighbors':[knn_score1],'Random Forest':[rf_score1]})
score_table = score_table.T
score_table.columns = ['Score']
score_table
```

Out[221]:

	Score
K Nearest Neighbors	0.442015
Random Forest	0.447180

According to the table above, we can use Random Forest as the score for this model for log_price, which is higher than KNN.

However, the score of KNN and Random Forest are relatively low. It could be due to some intangible features such as

the room quality, service friendliness, and the environment, etc

Tokyo

1.1 Data Visualization

1.1 Location and availabiity Overview

In [222]:
```
tky = pd.read_csv('https://raw.githubusercontent.com/AmysnL/data-bootcamp-final-project/main/Tokyo.csv')
tky.head()
```

Out[222]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimu
0	35303	La Casa Gaienmae C Harajuku, Omotesando is nearby	151977	Miyuki	NaN	Shibuya Ku	35.67152	139.71203	Private room	4196	
1	197677	Oshiage Holiday Apartment	964081	Yoshimi & Marek	NaN	Sumida Ku	35.71721	139.82596	Entire home/apt	10975	
2	289597	Private apt in central Tokyo #203	341577	Hide&Kei	NaN	Nerima Ku	35.74267	139.65810	Entire home/apt	4196	
3	370759	Cozy flat #203, local area YET 10 mins to shib...	1573631	Gilles,Mayumi,Taiki	NaN	Setagaya Ku	35.66344	139.65593	Entire home/apt	6994	
4	700253	Private apt in central Tokyo #201	341577	Hide&Kei	NaN	Nerima Ku	35.74264	139.65832	Entire home/apt	3981	

In [223]:

```
# drop the extreme for visualization:
tky_v = tky.loc[tky['latitude']>=35,:]
tky_v
```

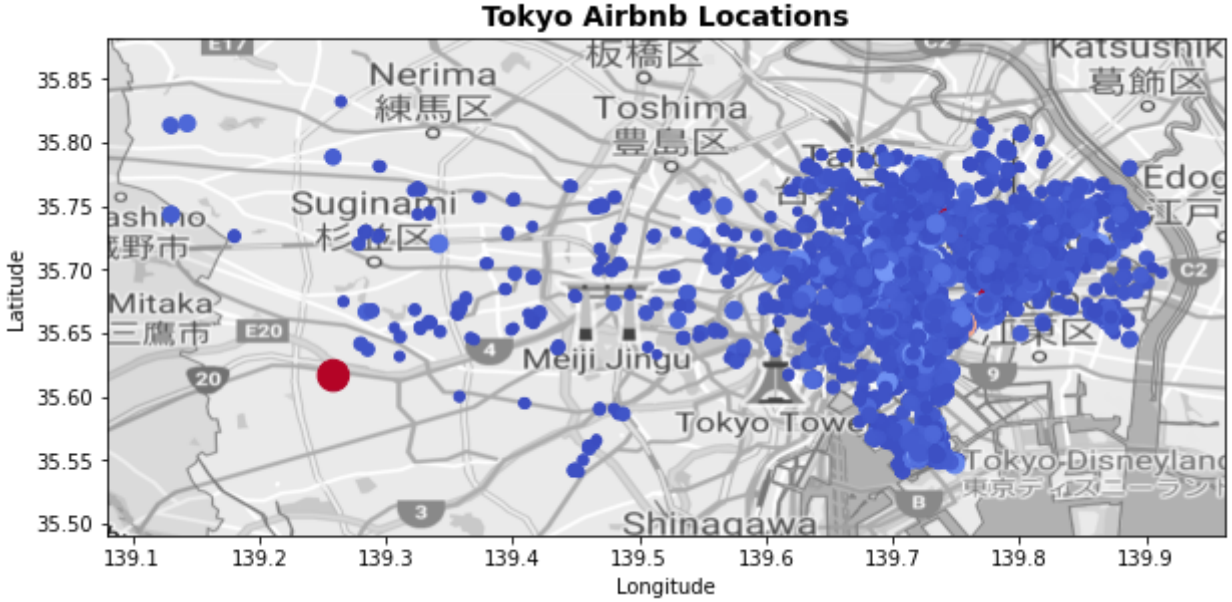
Out[223]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type
0	35303	La Casa Gaienmae C Harajuku, Omotesando is nearby	151977	Miyuki	NaN	Shibuya Ku	35.67152	139.71203	Private room
1	197677	Oshiage Holiday Apartment	964081	Yoshimi & Marek	NaN	Sumida Ku	35.71721	139.82596	Entire home/apt
2	289597	Private apt in central Tokyo #203	341577	Hide&Kei	NaN	Nerima Ku	35.74267	139.65810	Entire home/apt
3	370759	Cozy flat #203, local area YET 10 mins to shib...	1573631	Gilles,Mayumi,Taiki	NaN	Setagaya Ku	35.66344	139.65593	Entire home/apt
4	700253	Private apt in central Tokyo #201	341577	Hide&Kei	NaN	Nerima Ku	35.74264	139.65832	Entire home/apt
...	...	...	...	...	...	...	...	...	...
11461	36083287	1min to station · Skytree/Tatami hotel/new design	235406925	Sayoko佐代子	NaN	Sumida Ku	35.70862	139.81393	Entire home/apt
11462	36083512	Asakusa Wired house 5pax near by UENO/AKIHABARA	267481408	Hyongsu	NaN	Taito Ku	35.72464	139.78078	Entire home/apt
11463	36084566	东京精品民宿	161702799	Yang	NaN	Toshima Ku	35.73587	139.73397	Private room
11464	36085357	#4 DIRECT ACCESS TO DISNEY IN BUS! 3MIN TO STA...	208189463	Shotaro	NaN	Edogawa Ku	35.66255	139.87236	Entire home/apt
11465	36086604	103号室:立地最高!駅近･買い物便利!改装したばかりの綺麗な和室です。	254667806	新東明	NaN	Shinjuku Ku	35.69889	139.70544	Private room

11425 rows × 14 columns

In [224]:
```
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
fig,ax = location('Tokyo', tky_v,0.4,1.8,10,10)
xl,xh,yl,yh = map_range(tky_v)
tky_map = import_img('https://github.com/570558305/dmafinal/blob/main/tkyv.png?raw=true')
bw_img = tky_map.convert('L')

ax.imshow(bw_img,extent=[xl-0.1  ,xh+0.28  ,yl-0.33  ,yh+0.25 ], cmap='gray')
```

Out[224]: <matplotlib.image.AxesImage at 0x12e1a9d90>

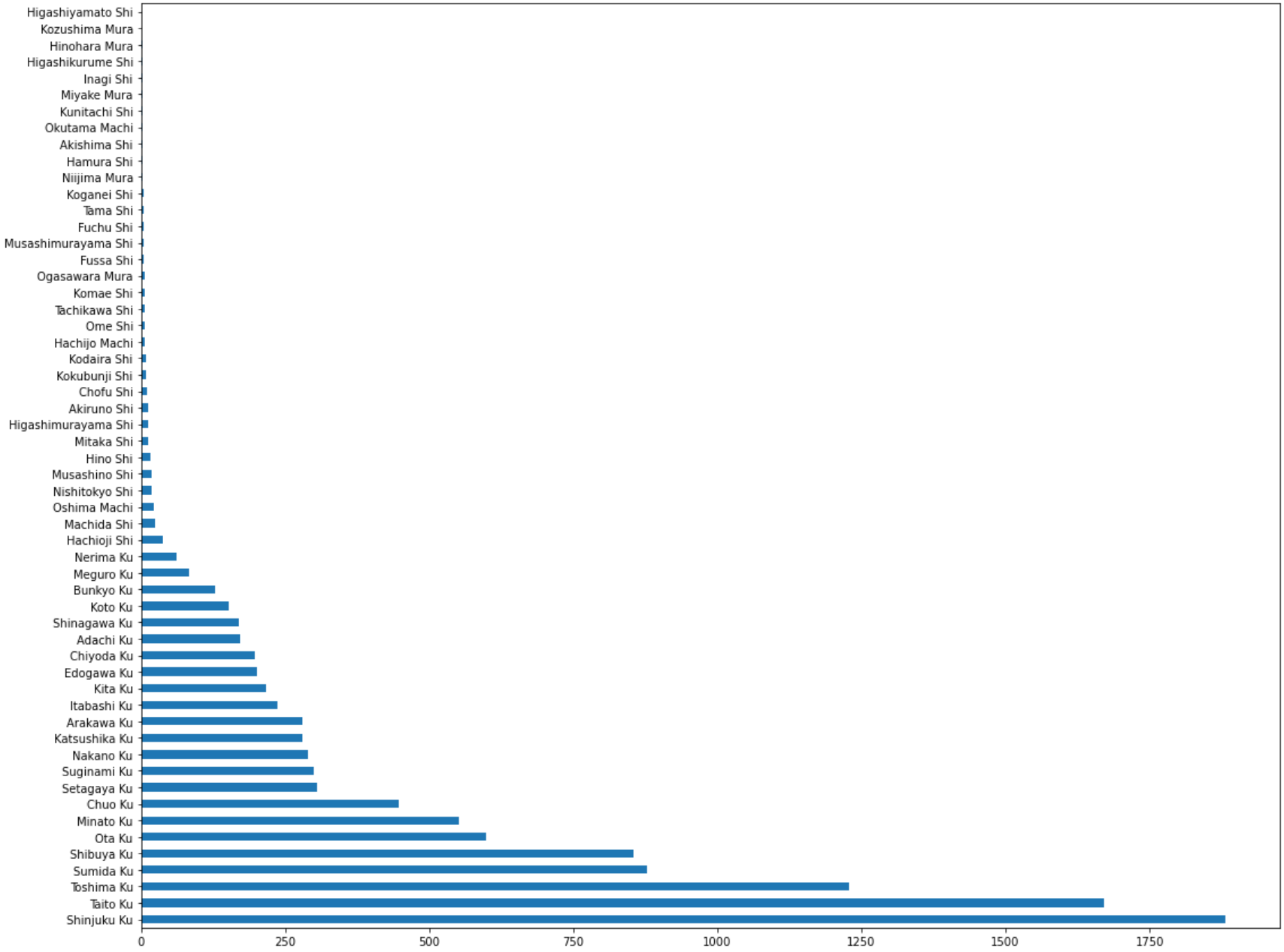


1.2 Neighborhood

1.21 The distribution of Airbnbs in each district

In [225]:
```
CountStatus = pd.value_counts(tky['neighbourhood'].values, sort=True)
CountStatus.plot.barh(figsize=(18,15))
```

Out[225]: <AxesSubplot:>



In [226]:
```
CountStatus.head(10)
```

Out[226]:
```
Shinjuku Ku    1882
Taito Ku       1670
Toshima Ku     1228
Sumida Ku       879
Shibuya Ku      854
Ota Ku          598
Minato Ku       551
Chuo Ku         448
Setagaya Ku     305
Suginami Ku     299
dtype: int64
```

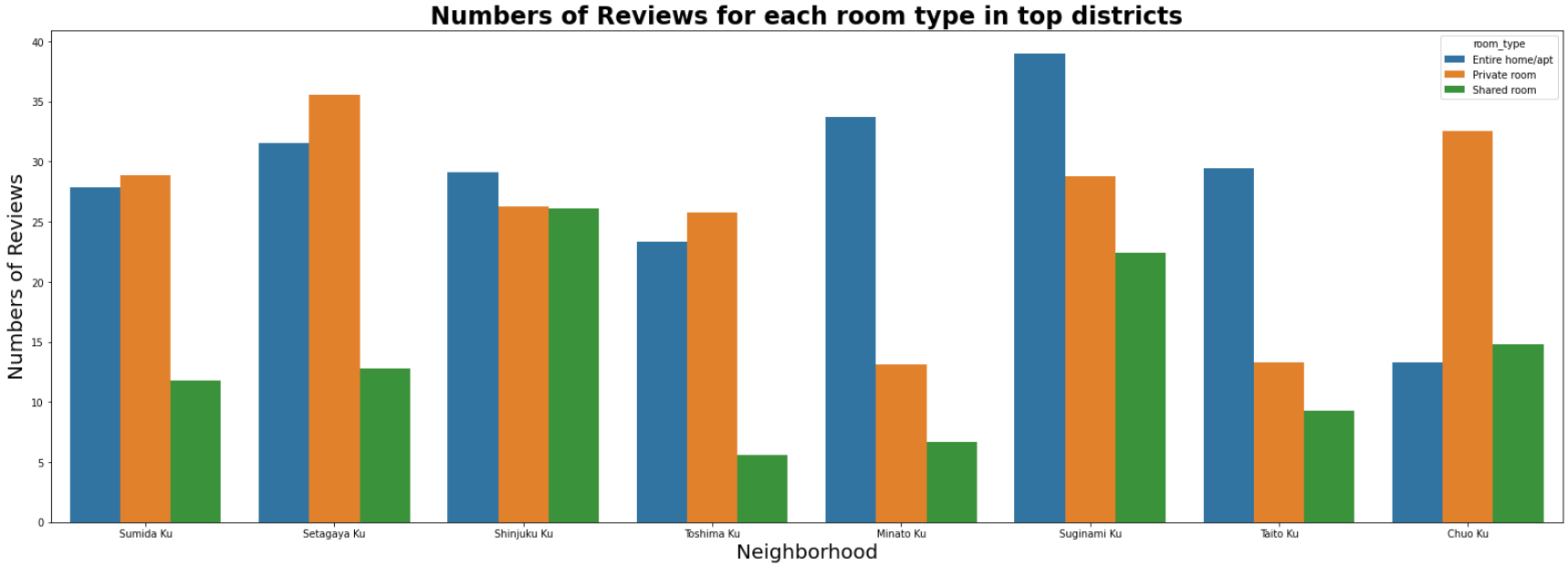
In [227]:
```
tky['room_type'].unique()
```

Out[227]:
```
array(['Private room', 'Entire home/apt', 'Shared room'], dtype=object)
```

In [228]:
```
g = ['Shinjuku Ku','Taito Ku','Toshima Ku','Sumida Ku','Shibuya Ku'
    'Ota Ku','Minato Ku','Chuo Ku','Setagaya Ku','Suginami Ku'] #The top ten cities with the most numbe
tky_10 = tky.loc[tky['neighbourhood'].isin(g)]
```

In [229]:
```
fig, ax = plt.subplots(figsize=(27,9))
ax =sns.barplot(data=tky_10, x="neighbourhood", y ='number_of_reviews', hue="room_type",ci=None )

ax.set_xlabel('Neighborhood',fontsize=20)
ax.set_ylabel('Numbers of Reviews',fontsize=20)
ax.set_title('Numbers of Reviews for each room type in top districts',fontweight='bold',fontsize=24)
```

Out[229]: Text(0.5, 1.0, 'Numbers of Reviews for each room type in top districts')



The number of Reviews indicates the popularity of the rooms and neighborhood. We can see that the green bar, representing the Shared room, is the least popular in most of the neighbourhoods. Entire home/apt is more popular in Shinjuku Ku, Minato Ku, Suginami Ku, and Taito Ku. Private room is more popular in Sumida Ku, Setagaya Ku, Toshima Ku, and Chuo Ku.

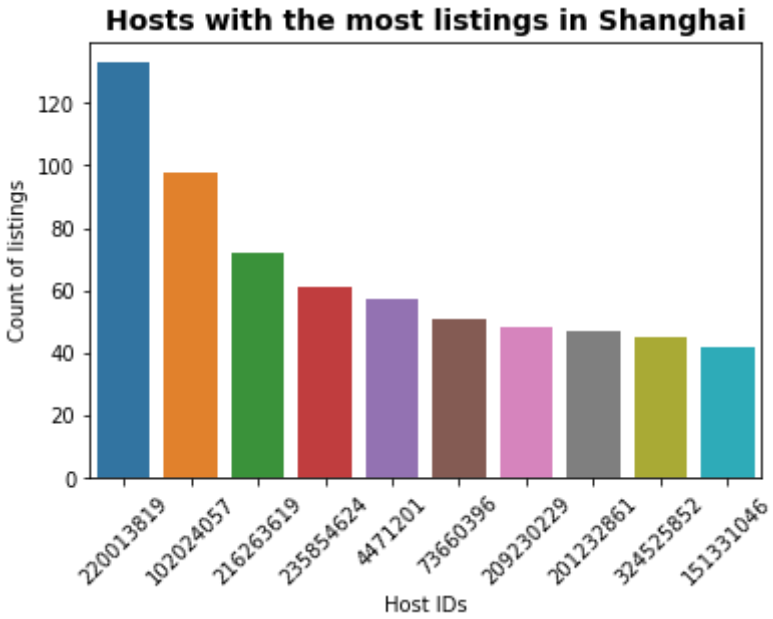
1.3 Host

In [230]:
```
top_host_t = tky['host_id'].value_counts()[:10]
```

In [231]:
```
ax = sns.barplot(top_host.index, top_host_t.values,order=top_host.index)
ax.set_xticklabels(ax.get_xticklabels (),rotation=45)
ax.set_title('Hosts with the most listings in Shanghai',size=14,fontweight='bold')
ax.set_ylabel('Count of listings')
ax.set_xlabel('Host IDs')
```

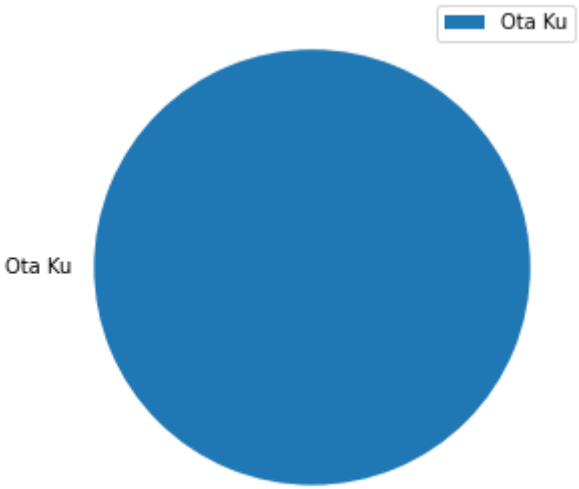
```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the followin
g variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
```

Out[231]:
```
Text(0.5, 0, 'Host IDs')
```



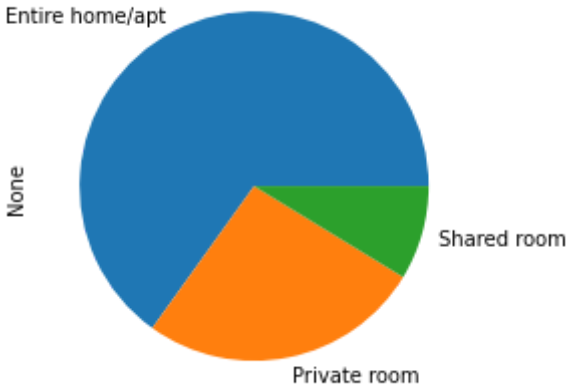
In [232]:
```
x =tky["host_id"].value_counts().nlargest(n=10).index.values
host_111 = tky.loc[tky['host_id'].isin(x) ,:]
host_111 = tky.loc[tky['host_id']==111043861,:]


host_n = host_111.groupby(['neighbourhood'])[['neighbourhood']].count()
fix,ax = plt.subplots(figsize = (7,5))
host_n.plot(kind='pie',x='neighbourhood',ax=ax,subplots=True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.legend(bbox_to_anchor=(1, 1))
```

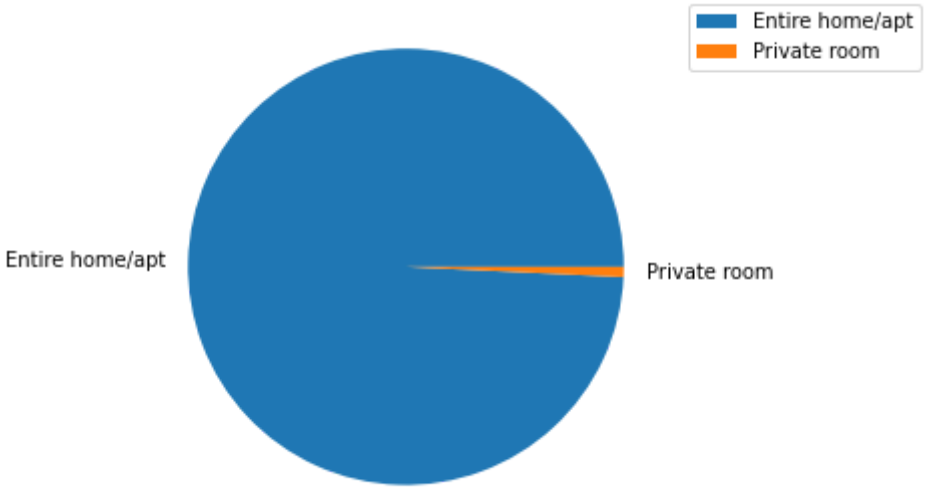
Out[232]: <matplotlib.legend.Legend at 0x13532cc10>



In [233]:
```
CountStatus = pd.value_counts(tky['room_type'].values, sort=True)
CountStatus.plot.pie()
host_rt = host_111.groupby(['room_type'])[['room_type']].count()
fix,ax = plt.subplots(figsize = (7,5))
host_rt.plot(kind='pie',x='room_type',ax=ax,subplots=True)
ax.set_ylabel('  ')
ax.set_xlabel('')
plt.legend(bbox_to_anchor=(1, 1))
```

Out[233]: <matplotlib.legend.Legend at 0x135cbe7c0>

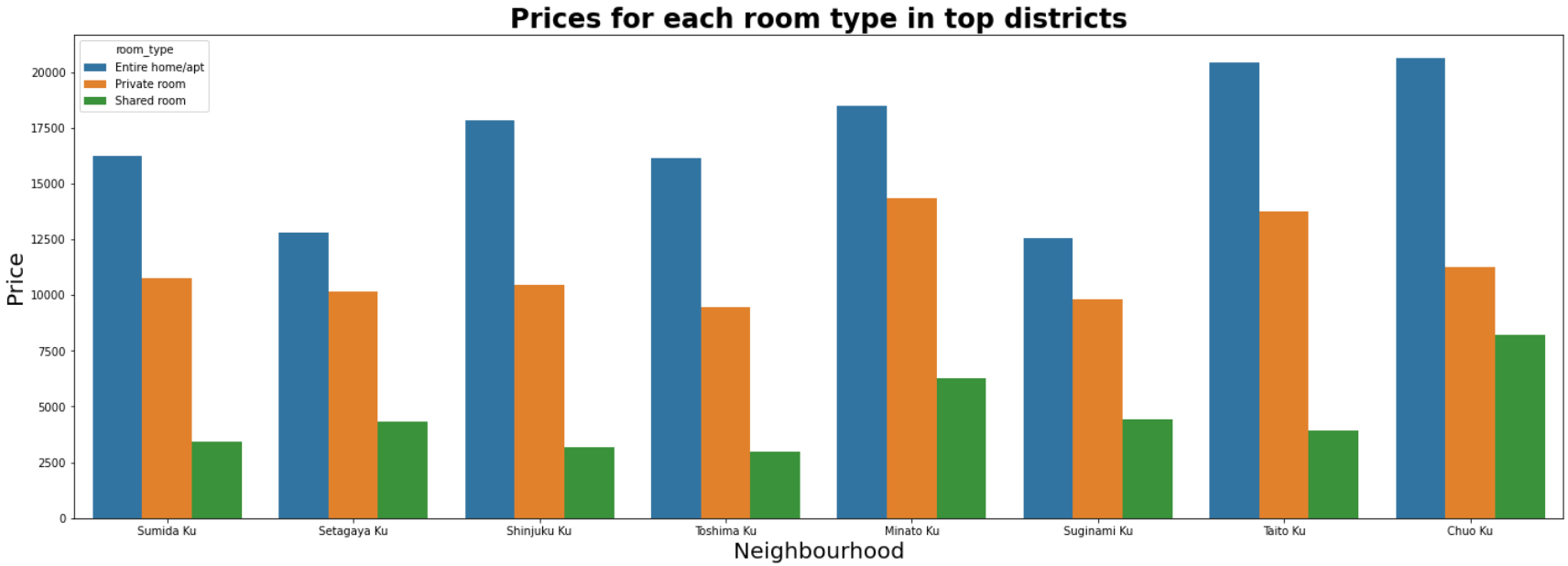




1.4 Prices

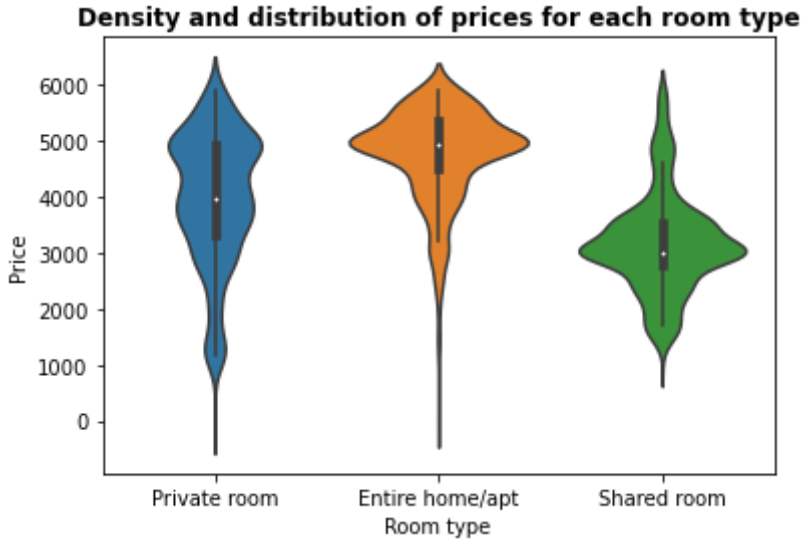
In [234]:
```
fig, ax = plt.subplots(figsize=(24,8))
ax =sns.barplot(data=tky_10, x="neighbourhood", y ='price', hue="room_type",ci=None)
ax.set_xlabel('Neighbourhood',fontsize=20)
ax.set_ylabel('Price',fontsize=20)
ax.set_title('Prices for each room type in top districts',fontweight='bold',fontsize=24)
```

Out[234]:
```
Text(0.5, 1.0, 'Prices for each room type in top districts')
```



In [235]:
```
ax = sns.violinplot(data=tky[tky.price < 6000] ,x='room_type',y='price')

ax.set_xlabel('Room type')
ax.set_ylabel('Price')
ax.set_title('Density and distribution of prices for each room type',fontweight='bold')
```

Out[235]:
```
Text(0.5, 1.0, 'Density and distribution of prices for each room type')
```



In [ ]:

2. Descriptive Data Analysis

2.1Data Overview

In [236]:
```
tky = pd.read_csv('https://raw.githubusercontent.com/AmysnL/data-bootcamp-final-project/main/Tokyo.csv')
tky.head()
```

Out[236]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimu
0	35303	La Casa Gaienmae C Harajuku, Omotesando is nearby	151977	Miyuki	NaN	Shibuya Ku	35.67152	139.71203	Private room	4196	
1	197677	Oshiage Holiday Apartment	964081	Yoshimi & Marek	NaN	Sumida Ku	35.71721	139.82596	Entire home/apt	10975	
2	289597	Private apt in central Tokyo #203	341577	Hide&Kei	NaN	Nerima Ku	35.74267	139.65810	Entire home/apt	4196	
3	370759	Cozy flat #203, local area YET 10 mins to shib...	1573631	Gilles,Mayumi,Taiki	NaN	Setagaya Ku	35.66344	139.65593	Entire home/apt	6994	
4	700253	Private apt in central Tokyo #201	341577	Hide&Kei	NaN	Nerima Ku	35.74264	139.65832	Entire home/apt	3981	

In [237]:
```
tky.describe()
```

Out[237]:

	id	host_id	neighbourhood_group	latitude	longitude	price	minimum_nights	number_of_reviews
count	1.146600e+04	1.146600e+04	0.0	11466.000000	11466.000000	1.146600e+04	11466.000000	11466.000000
mean	2.640545e+07	1.399776e+08	NaN	35.687159	139.734537	1.498470e+04	3.316239	26.028868
std	7.565212e+06	8.064289e+07	NaN	0.217853	0.092771	3.327345e+04	7.766766	38.455449
min	3.530300e+04	1.519770e+05	NaN	27.072330	139.130020	0.000000e+00	1.000000	0.000000
25%	2.215251e+07	5.783171e+07	NaN	35.679620	139.698975	6.026000e+03	1.000000	2.000000
50%	2.841873e+07	1.502326e+08	NaN	35.701910	139.729165	1.000700e+04	1.000000	12.000000
75%	3.222416e+07	2.135896e+08	NaN	35.724215	139.784270	1.721600e+04	2.000000	32.000000
max	3.608660e+07	2.714130e+08	NaN	35.832430	142.202880	1.000046e+06	180.000000	453.000000

In [238]:
```
tky.isnull().sum()
```

Out[238]:
```
id                          0
name                        0
host_id                     0
host_name                  16
neighbourhood_group     11466
neighbourhood               0
latitude                    0
longitude                   0
room_type                   0
price                       0
minimum_nights              0
number_of_reviews           0
last_review              1677
reviews_per_month        1677
dtype: int64
```

In [239]:
```
tky.drop(columns=['neighbourhood_group','last_review','reviews_per_month'],axis=1,inplace=True)
tky
```

Out[239]:

	id	name	host_id	host_name	neighbourhood	latitude	longitude	room_type	price	minimum_night
0	35303	La Casa Gaienmae C Harajuku, Omotesando is nearby	151977	Miyuki	Shibuya Ku	35.67152	139.71203	Private room	4196	2
1	197677	Oshiage Holiday Apartment	964081	Yoshimi & Marek	Sumida Ku	35.71721	139.82596	Entire home/apt	10975	
2	289597	Private apt in central Tokyo #203	341577	Hide&Kei	Nerima Ku	35.74267	139.65810	Entire home/apt	4196	3
3	370759	Cozy flat #203, local area YET 10 mins to shib...	1573631	Gilles,Mayumi,Taiki	Setagaya Ku	35.66344	139.65593	Entire home/apt	6994	2
4	700253	Private apt in central Tokyo #201	341577	Hide&Kei	Nerima Ku	35.74264	139.65832	Entire home/apt	3981	3
...	...	...	...	...	...	...	...	...	...	.
11461	36083287	1min to station · Skytree/Tatami hotel/new design	235406925	Sayoko佐代子	Sumida Ku	35.70862	139.81393	Entire home/apt	21951	
11462	36083512	Asakusa Wired house 5pax near by UENO/AKIHABARA	267481408	Hyongsu	Taito Ku	35.72464	139.78078	Entire home/apt	7747	
11463	36084566	东京精品民宿	161702799	Yang	Toshima Ku	35.73587	139.73397	Private room	56275	
11464	36085357	#4 DIRECT ACCESS TO DISNEY IN BUS! 3MIN TO STA...	208189463	Shotaro	Edogawa Ku	35.66255	139.87236	Entire home/apt	6456	
11465	36086604	103号室:立地最高!駅近･買い物便利!改装したばかりの綺麗な和室です。	254667806	新東明	Shinjuku Ku	35.69889	139.70544	Private room	7962	

11466 rows × 11 columns

In [ ]:

2.2 Descriptive Data Analysis¶

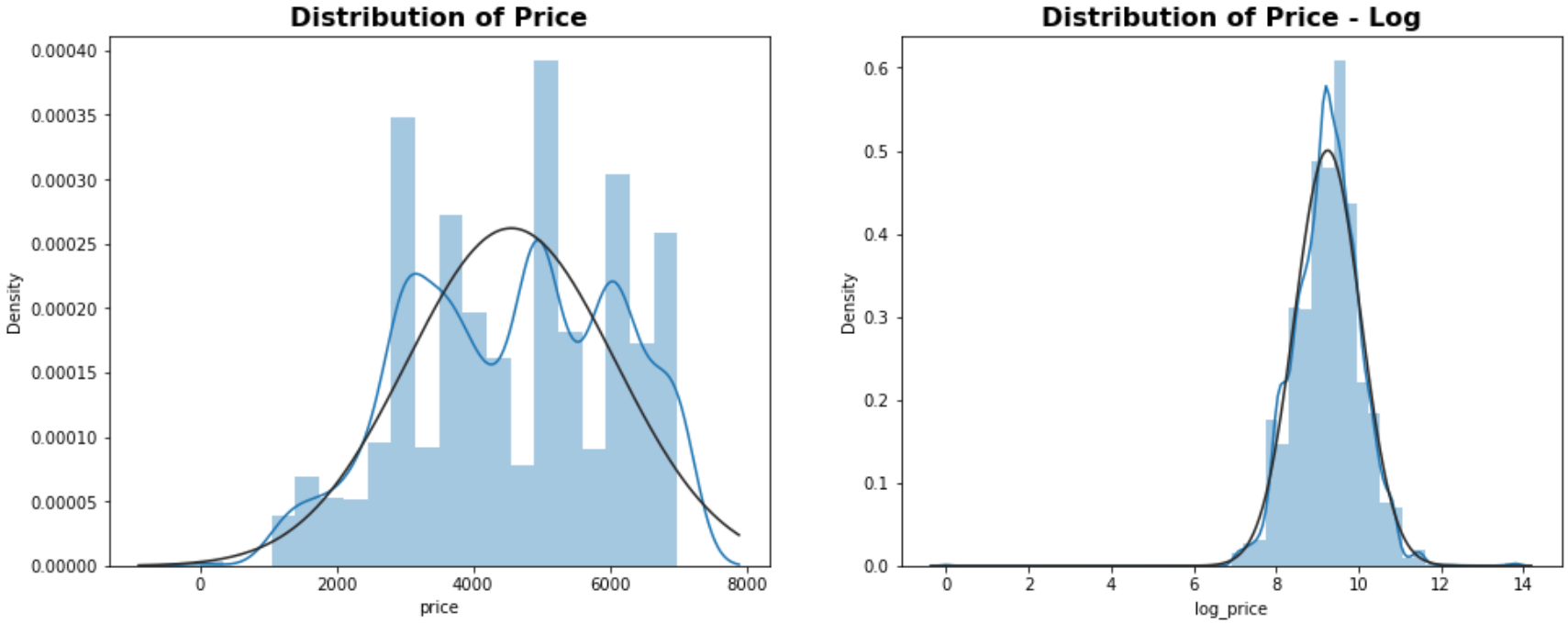
In [240]:
```
en_t = tky.copy()
en_t['neighbourhood'] = en_t['neighbourhood'].astype('category').cat.codes
en_t['room_type'] = en_t['room_type'].astype("category").cat.codes
#mean = en_t['reviews_per_month'].mean()
#en_t['reviews_per_month'].fillna(mean, inplace=True) ####
en_t['log_price'] = np.log(en_t.price+1)  #####
en_t = en_t.drop(columns=['id' ,'host_id','price']) # delete price column
en_t.isnull().sum()
```

Out[240]:
```
name                 0
host_name           16
neighbourhood        0
latitude             0
longitude            0
room_type            0
minimum_nights       0
number_of_reviews    0
log_price            0
dtype: int64
```

In [241]:
```
fig,ax = plt.subplots(1,2,figsize = (16,6))
sns.distplot(tky['price'].loc[tky['price'] <= 7000], fit = norm, ax = ax[0])
ax[0].set_title("Distribution of Price", size = 16,weight = 'bold')
sns.distplot(en_t['log_price'],fit=norm , ax = ax[1])
ax[1].set_title("Distribution of Price - Log", size = 16,weight = 'bold')
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `d
isplot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```

Out[241]: Text(0.5, 1.0, 'Distribution of Price - Log')



the log price distribution is approaximately a normal distribution of mean of 9

2.3 Correlation

In [242]:
```
listings_t = tky
```

In [243]:
```
# Change data type
listings_price_t = listings_t[['price','minimum_nights','number_of_reviews','latitude','longitude']]
listings_price_t[:5]
```

Out[243]:

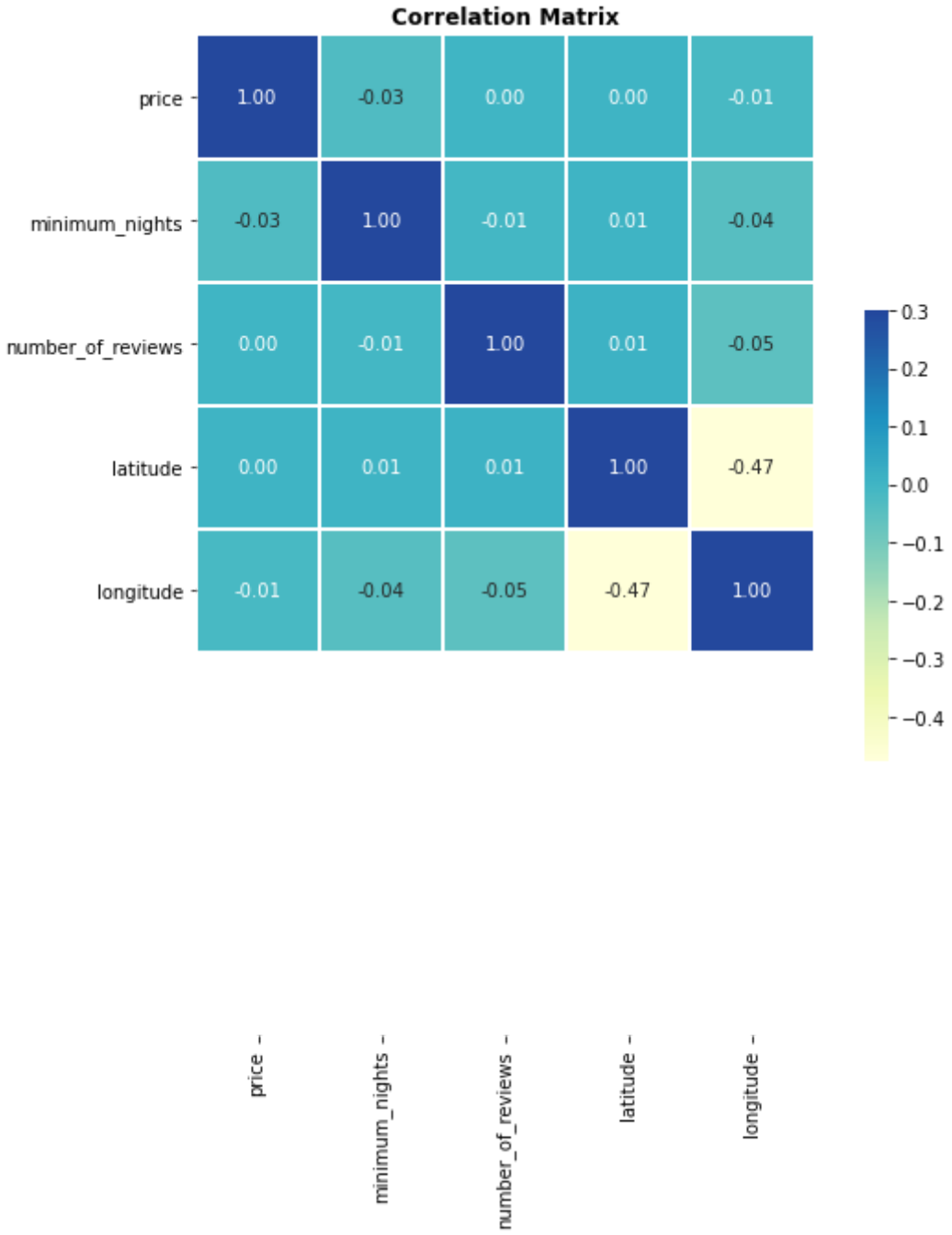
	price	minimum_nights	number_of_reviews	latitude	longitude
0	4196	28	18	35.67152	139.71203
1	10975	3	156	35.71721	139.82596
2	4196	30	107	35.74267	139.65810
3	6994	29	99	35.66344	139.65593
4	3981	30	101	35.74264	139.65832

This is the correlation matrix showing relationships between different variables.

This will help the host to improve decisions.

In [244]:
```
t.figure(figsize=(10,10))
lette = sns.diverging_palette(20, 220, n=256)
rr=listings_price_t.corr(method='pearson')
s.heatmap(corr, annot=True, fmt=".2f", cmap="YlGnBu", vmax=.3, center=0,square=True, linewidths=1, cbar_
t.title("Correlation Matrix",size=12, weight='bold')
```

Out[244]:
```
Text(0.5, 1.0, 'Correlation Matrix')
```



3. Regression

3.1 Linear Regression

In [245]:
```
reg_price = smf.ols('price ~ room_type  + number_of_reviews '
                    ,data=tky).fit()
print(reg_price.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.019
Model:                            OLS   Adj. R-squared:                  0.019
Method:                 Least Squares   F-statistic:                     75.92
Date:                Fri, 18 Dec 2020   Prob (F-statistic):           1.28e-48
Time:                        10:33:09   Log-Likelihood:            -1.3555e+05
No. Observations:               11466   AIC:                         2.711e+05
Df Residuals:                   11462   BIC:                         2.711e+05
Df Model:                           3
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                     1.861e+04    448.580     41.490      0.000    1.77e+04    1.95e+04
room_type[T.Private room] -7793.2988    714.250    -10.911      0.000   -9193.350   -6393.247
room_type[T.Shared room]  -1.377e+04   1117.570    -12.321      0.000    -1.6e+04   -1.16e+04
number_of_reviews            -14.8056      8.070     -1.835      0.067     -30.624       1.013
==============================================================================
Omnibus:                    27368.655   Durbin-Watson:                   1.646
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        235885671.323
Skew:                          24.590   Prob(JB):                         0.00
Kurtosis:                     703.945   Cond. No.                         173.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

From the regression result above, we can see that,

room_type[T.Private room], room_type[T.Shared room], number_of_reviews all negatively contribute to Price.

3.2 Log

3.2 Log

In [246]:
```
reg_logprice = smf.ols('log_price ~ neighbourhood +room_type +  number_of_reviews '
                       ,data= en_t).fit()
print(reg_logprice.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_price   R-squared:                       0.264
Model:                            OLS   Adj. R-squared:                  0.264
Method:                 Least Squares   F-statistic:                     1371.
Date:                Fri, 18 Dec 2020   Prob (F-statistic):               0.00
Time:                        10:33:09   Log-Likelihood:                -11906.
No. Observations:               11466   AIC:                         2.382e+04
Df Residuals:                   11462   BIC:                         2.385e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      9.5186      0.019    513.889      0.000       9.482       9.555
neighbourhood  0.0012      0.000      3.057      0.002       0.000       0.002
room_type     -0.6326      0.010    -63.279      0.000      -0.652      -0.613
number_of_reviews    -0.0015      0.000     -9.008      0.000      -0.002      -0.001
==============================================================================
Omnibus:                     2325.920   Durbin-Watson:                   1.455
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            76221.606
Skew:                          -0.176   Prob(JB):                         0.00
Kurtosis:                      15.626   Cond. No.                         166.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

From the regression result above, we can see that room_type and number_of_reviews all negatively contribute to the price.

4. Machine Learning

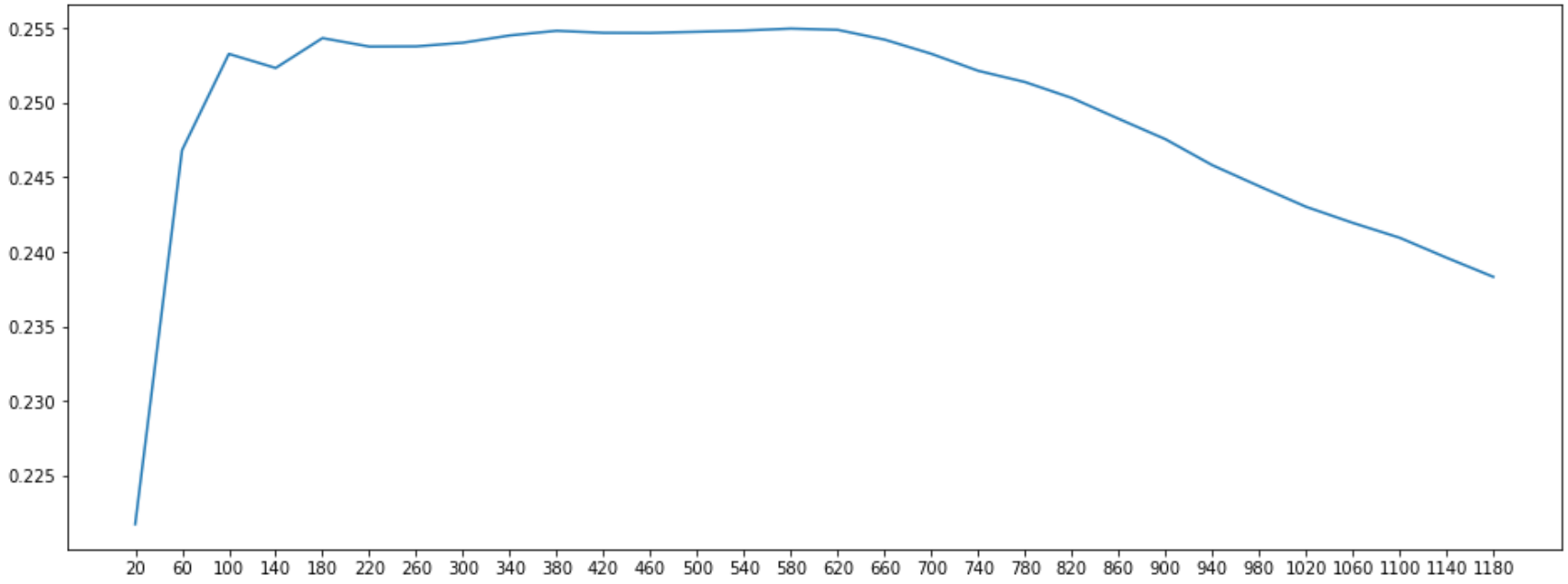
In [247]:
```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(en_t[['room_type']].values, en_t[['log_price']].
                                                        test_size=0.25, random_state=0)
```

(1) KNN

In [*]:
```
from sklearn.neighbors import KNeighborsRegressor as knn
from sklearn.model_selection import cross_val_score
scores = pd.Series(dtype='float')
for i in range(20,1200,40):
    scores[str(i)] = cross_val_score(knn(n_neighbors=i),X_train1, y_train1,cv=5).mean()
indx = scores.idxmax()
```

In [249]:
```
fig,ax = plt.subplots(figsize=(16,6))
plt.plot(scores.index,scores.values)
```

Out[249]:
```
[<matplotlib.lines.Line2D at 0x131cb11f0>]
```



In [250]:
```
skl_knn = knn(n_neighbors = int(indx)).fit(X_train1, y_train1)  #####
knn_score1 = skl_knn.score(X_test1,y_test1)
print(knn_score1)
```

```
0.27378530213152075
```

(2) Random Forest

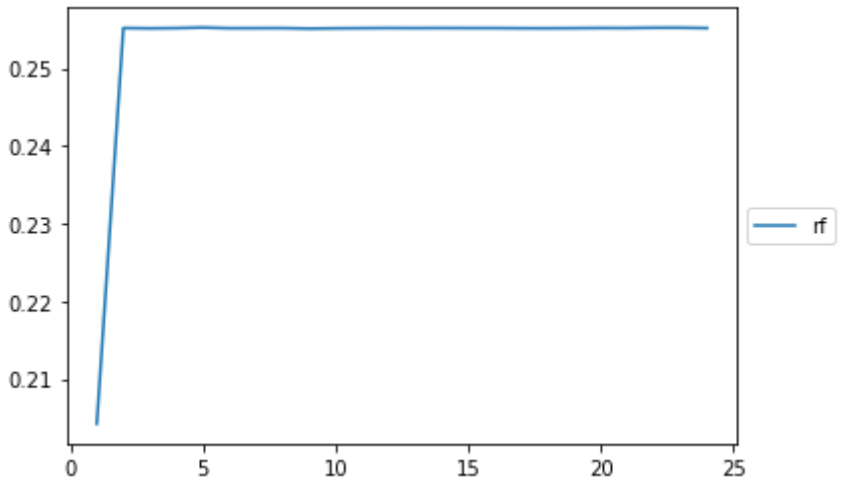
In [251]:
```
from sklearn.ensemble import RandomForestRegressor as rf
cross_val_score(rf(n_estimators = 100, max_depth = 3), X_train1, y_train1.ravel(), cv=5).mean()
```

Out[251]:
```
0.25515309288403865
```

In [252]:
```
cv_scores = pd.DataFrame()
for i in range (1,25):
    cv_scores.loc[i,'rf'] = cross_val_score(rf(n_estimators = 100, max_depth = i), X_train1, y_train1.ra
```

In [253]:
```
ax = cv_scores.plot()
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
```

Out[253]:
```
<matplotlib.legend.Legend at 0x131cea2b0>
```



In [254]:
```
indx = scores.idxmax()
skl_rf = rf(n_estimators=100,max_depth=int(indx)).fit(X_train1, y_train1.ravel())
rf_score1 = skl_rf.score(X_test1,y_test1)
print(rf_score1)
```

```
0.2739916217016166
```

In [255]:
```
score_table = pd.DataFrame({'K Nearest Neighbors':[knn_score1],'Random Forest':[rf_score1]})
score_table = score_table.T
score_table.columns = ['Score']
score_table
```

Out[255]:

	Score
K Nearest Neighbors	0.273785
Random Forest	0.273992

According to the table above, we can use Random Forest as the score for this model for log_price, which is higher than KNN.

However, the score of KNN and Random Forest are relatively low. It could be due to some intangible features such as

the room quality, service friendliness, and the environment, etc

In [256]:
```
sh2 = sh.loc[:,['id', 'neighbourhood', 'room_type', 'price', 'minimum_nights',
       'number_of_reviews' ]]
sh2['city'] = np.array(['shanghai']*len(sh))
lst = []
for i in sh2['price']:
    i1 = i*0.15
    lst.append(i1)
sh2['price'] = lst
```

In [257]:
```
ist2 = istb.loc[:,['id', 'neighbourhood', 'room_type', 'price', 'minimum_nights',
       'number_of_reviews' ]]
ist2['city'] = np.array(['istanbul']*len(istb))
lst2=[]
for i in ist2['price']:
    i1 = i*0.13
    lst2.append(i1)

ist2['price'] = lst2
```

In [258]:
```
tky2 = tky.loc[:,['id', 'neighbourhood', 'room_type', 'price', 'minimum_nights',
       'number_of_reviews' ]]
tky2['city'] = np.array(['tokyo']*len(tky))
lst3=[]
for i in tky2['price']:
    i1 = i*0.0097
    lst3.append(i1)

tky2['price'] = lst3
```

In [259]:
```
sg2 = sp.loc[:,['id', 'neighbourhood', 'room_type', 'price', 'minimum_nights',
       'number_of_reviews' ]]
sg2['city'] = np.array(['Singapore']*len(sp))
lst4=[]
for i in sg2['price']:
    i1 = i*0.75
    lst4.append(i1)

sg2['price'] = lst4
```

In [260]:
```
total = pd.concat([sh2,ist2,tky2,sg2],sort=False)
```

In [261]:
```
total
```

Out[261]:

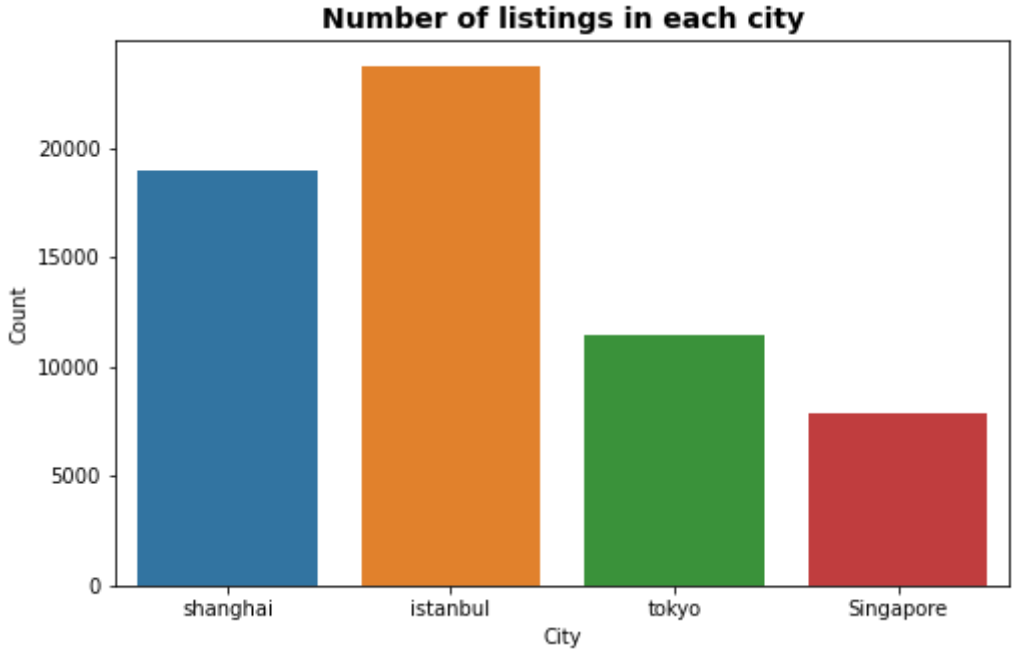
	id	neighbourhood	room_type	price	minimum_nights	number_of_reviews	city
0	24963	徐汇区 / Xuhui District	Entire home/apt	70.20	3	85	shanghai
1	24991	徐汇区 / Xuhui District	Entire home/apt	80.25	3	1	shanghai
2	139828	普陀区 / Putuo District	Entire home/apt	53.25	3	26	shanghai
3	139846	静安区 / Jing'an District	Entire home/apt	87.60	1	57	shanghai
5	185736	徐汇区 / Xuhui District	Private room	75.00	1	8	shanghai
...	...	...	...	...	...	...	...
7902	38105126	Queenstown	Entire home/apt	75.00	3	0	Singapore
7903	38108273	Tanglin	Entire home/apt	412.50	6	0	Singapore
7904	38109336	Kallang	Private room	43.50	30	0	Singapore
7905	38110493	River Valley	Private room	42.00	14	0	Singapore
7906	38112762	River Valley	Private room	48.75	90	0	Singapore

62072 rows × 7 columns

The number of listing comparison

In [262]:
```
figure,ax = plt.subplots(figsize=(8,5))
sns.countplot(x='city',data=total,ax=ax)
ax.set_xlabel('City')
ax.set_ylabel('Count')
ax.set_title('Number of listings in each city',size=14,fontweight='bold')
```

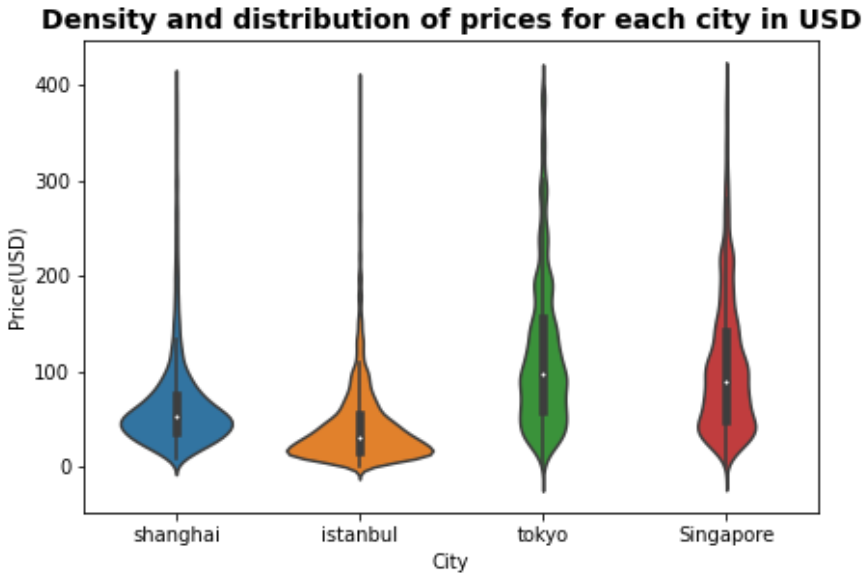
Out[262]:
```
Text(0.5, 1.0, 'Number of listings in each city')
```

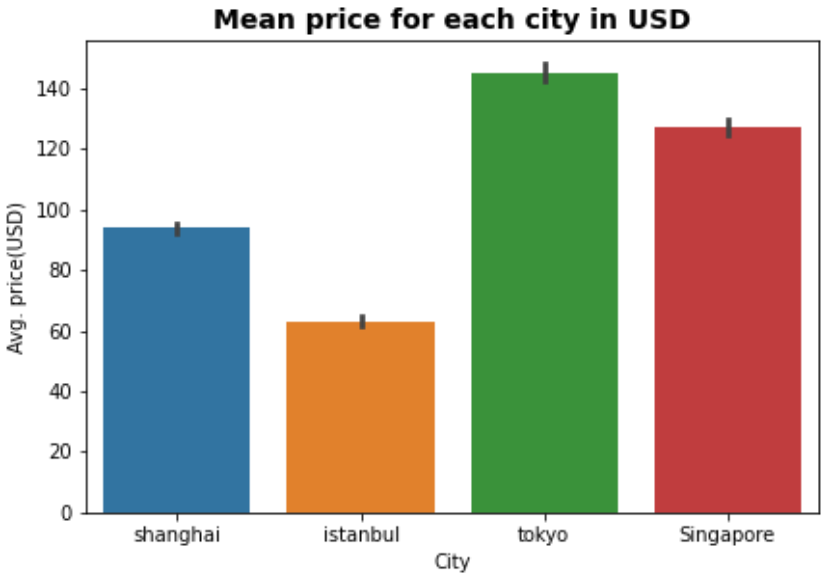


The price comparison

In [263]:
```
total_dealt = total[total.price < 400]
figure,ax = plt.subplots(1,2,figsize=(15,4.5))
sns.barplot(x='city', y='price', data=total, ci=68,ax=ax[1])
ax[1].set_xlabel('City')
ax[1].set_ylabel('Avg. price(USD)')
ax[1].set_title('Mean price for each city in USD',size=14, fontweight='bold')
sns.violinplot(data=total_dealt,x='city',y='price',ax=ax[0])
ax[0].set_xlabel('City')
ax[0].set_ylabel('Price(USD)')
ax[0].set_title('Density and distribution of prices for each city in USD',size=14, fontweight='bold')
```

Out[263]:
```
Text(0.5, 1.0, 'Density and distribution of prices for each city in USD')
```

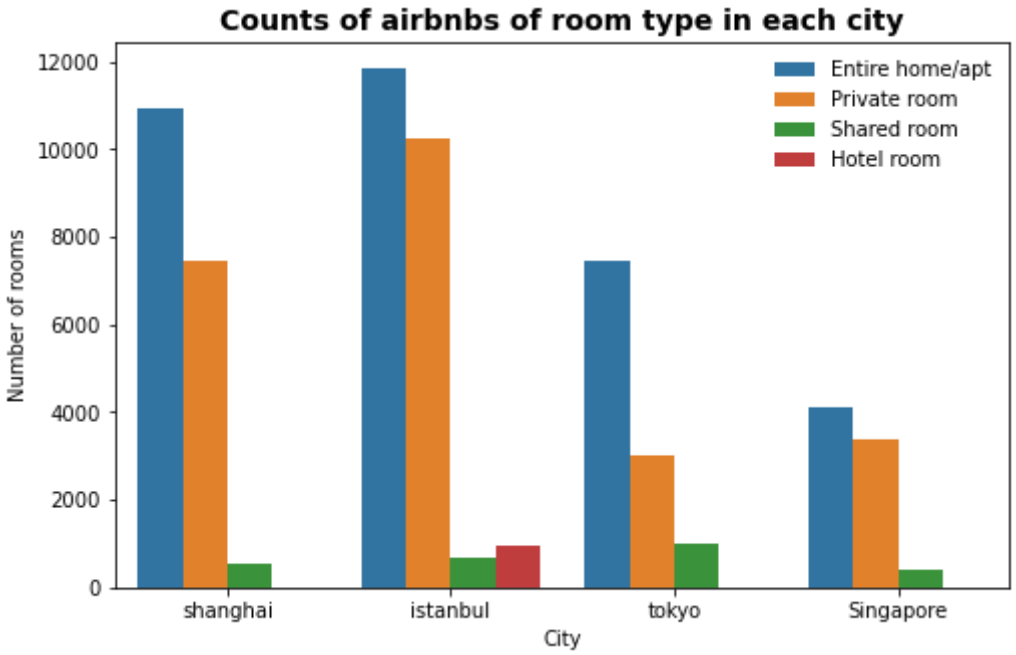




The room type comparison

In [264]:
```
figure,ax = plt.subplots(figsize=(8,5))
sns.countplot(data=total,x='city',hue='room_type',ax=ax)
ax.set_title('Counts of airbnbs of room type in each city',fontsize=14,fontweight ='bold')
ax.set_xlabel('City')
ax.set_ylabel('Number of rooms')
ax.legend(frameon=False)
```

Out[264]:
```
<matplotlib.legend.Legend at 0x136ab0c40>
```



Generally, there are the most entire rooms listing, followed by private rooms and shared rooms. In Singapore, the difference of the amount of listed entire room and private room is small.

Machine Learning Comparison

In [265]:
```
scoreTable = {'K Nearest Neighbors': [0.170709,0.209943,0.442015,0.273785],
'Random Forest': [0.177920,0.210339,0.447361,0.273956]}
scoreTable = pd.DataFrame(scoreTable)
scoreTable["Prediction"] = list(["Shanghai","Istanbul","Singapore","Tokyo"])
scoreTable = scoreTable.set_index("Prediction")
scoreTable = scoreTable.T
scoreTable
```

Out[265]:

Prediction	Shanghai	Istanbul	Singapore	Tokyo
K Nearest Neighbors	0.170709	0.209943	0.442015	0.273785
Random Forest	0.177920	0.210339	0.447361	0.273956

So we use Random Forest for the four cities for predictions since their scores are higher than K-nn

Sentimental Analysis

In [266]:
```
conda install -c conda-forge wordcloud #import wordcloud for sentimental analysis visualization
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [267]:
```
sh_r = pd.read_csv('https://raw.githubusercontent.com/570558305/airbnb_review/master/sh_reviews.csv', he
                   encoding= 'unicode_escape')
sh_r.head(5)
```

Out[267]:

	comments
0	Even though the actual apt listed was not read...
1	Jia's place is in a vibrant and (in my opinion...
2	The host canceled my reservation the day befor...
3	_°ç_ä½_½®__å¥½ç_,_¿å__´ä_ç¬¦å_é¢__
4	ä½_½®__„å¥,_»å_____¹ä¾¿,_¿é_å¾_¹²_,å®__,__å¥!

In [268]:
```
import pandas as pd
istanbul = pd.read_csv(r'/Users/amyshuning/Desktop/istb_reviews.csv',header=0, encoding = 'unicode_escap
istanbul
```

Out[268]:

	comments
0	My daughter and her friend and I all stayed at...
1	Amazing location, super friendly hosts and rea...
2	I had an amazing stay! The room was spacious w...
3	I wasn't sure about the location at first, thi...
4	TÃ¼rkische Gastfreundschaft par excellence!
...	...
28394	.
28395	It won't be long till this is on Airbnb Luxe. ...
28396	Harika bir ev, inanÄ±lmaz gÃ¼zel bir manzara, ...
28397	___Ð°Ð½Ð°Ð²Ð¸Ð³Ð°Ñ‚Ð¾Ñ€__ Ð½Ð° Ð´Ð²Ð° Ð´Ð½_. Ð_...
28398	Mehmet__ place is even more beautiful than dep...

28399 rows × 1 columns

In [269]:
```
sp_r = pd.read_csv('https://raw.githubusercontent.com/570558305/airbnb_review/master/sp_reviews.csv', he
                   encoding= 'unicode_escape')
sp_r.head()
```

Out[269]:

	comments
0	Fran was absolutely gracious and welcoming. Ma...
1	A comfortable room in a smart condo developmen...
2	esta ubicado frente al barrio chino, se encuen...
3	The location is very convenient
4	The apartment is simple yet has everything you...

In [270]:
```
tky_r = pd.read_csv('https://raw.githubusercontent.com/570558305/airbnb_review/master/tky_reviews.csv',
                    encoding= 'unicode_escape')
tky_r
```

Out[270]:

	comments
0	Couldn__ get any better!\r\n\r\nThe apartment ...
1	Nice place to stay. Really liked ishikawatai.
2	_±æ¥æ_ä¸___³å__°é____»ã_è¿__ï½¤å____ï½ºï_ï¾__ï...
3	Great room for a great value. would stay again.
4	nice environment and the host is really kind. ...
...	...
25251	ï½µï½¼ï½_ï½¤è²·___ä¾¿å_
25252	___å±__ä½__å±_____ï½5äººã_æ³_____½¤_____...
25253	Shintaro is a very great landlord. He prepared...
25254	__æ»___§ã__·æ_æ»___§ã_ä¸___±ã_____½¤å¿____...
25255	é§_____è¿__ï½¤ã__¨å___¶ºé°__ï½±ï_ï¾__ï½¨ã_____...

25256 rows × 1 columns

AFINN is a list of English words rated for valence with an integer between minus five (negative) and plus five (positive). The words have been manually labeled by Finn Årup Nielsen in 2009-2011. The file is tab-separated.

In [271]:
```
n_data = {"abandon":"-2","abandoned":"-2","abandons":"-2","abducted":"-2","abduction":"-2",
          "abductions":"-2","abhor":"-3","abhorred":"-3","abhorrent":"-3","abhors":"-3",
          "abilities":"2","ability":"2","aboard":"1","absentee":"-1","absentees":"-1","absolve":"2",
          "absolved":"2","absolves":"2","absolving":"2","absorbed":"1","abuse":"-3","abused":"-3",
          "abuses":"-3","abusive":"-3","accept":"1","accepted":"1","accepting":"1","accepts":"1",
          "accident":"-2","accidental":"-2","accidentally":"-2","accidents":"-2","accomplish":"2",
          "accomplished":"2","accomplishes":"2","accusation":"-2","accusations":"-2","accuse":"-2",
          "accused":"-2","accuses":"-2","accusing":"-2","ache":"-2","achievable":"1","aching":"-2",
          "acquit":"2","acquits":"2","acquitted":"2","acquitting":"2","acrimonious":"-3","active":"1",
          "adequate":"1","admire":"3","admired":"3","admires":"3","admiring":"3","admit":"-1",
          "admits":"-1","admitted":"-1","admonish":"-2","admonished":"-2","adopt":"1","adopts":"1",
          "adorable":"3","adore":"3","adored":"3","adores":"3","advanced":"1","advantage":"2",
          "advantages":"2","adventure":"2","adventures":"2","adventurous":"2","affected":"-1",
          "affection":"3","affectionate":"3","afflicted":"-1","affronted":"-1","afraid":"-2","aggravate"
          "aggravates":"-2","aggravating":"-2","aggression":"-2","aggressions":"-2","aggressive":"-2","a
          "agonise":"-3","agonised":"-3","agonises":"-3","agonising":"-3","agonize":"-3","agonized":"-3"
```

In [272]:
```
score_word_dict = dict(afinn_data)
def string_score(df, score_word_dict):
    score = 0
    words = df.split(" ")
    for word in words:
        if(word in score_word_dict):
            score += int(score_word_dict[word])
    return score
```

Shanghai

In [273]:
```
#summary
from sklearn.feature_extraction.text import CountVectorizer
select = pd.DataFrame()
select = sh_r[['comments']]
vect = CountVectorizer()
scores = []
for i in select['comments']:
    score = string_score(i,score_word_dict)
    scores.append(score)
select['scores'] = scores
select.sort_values(ascending = False,by = 'scores',inplace = True)
select.tail()
```

Out[273]:

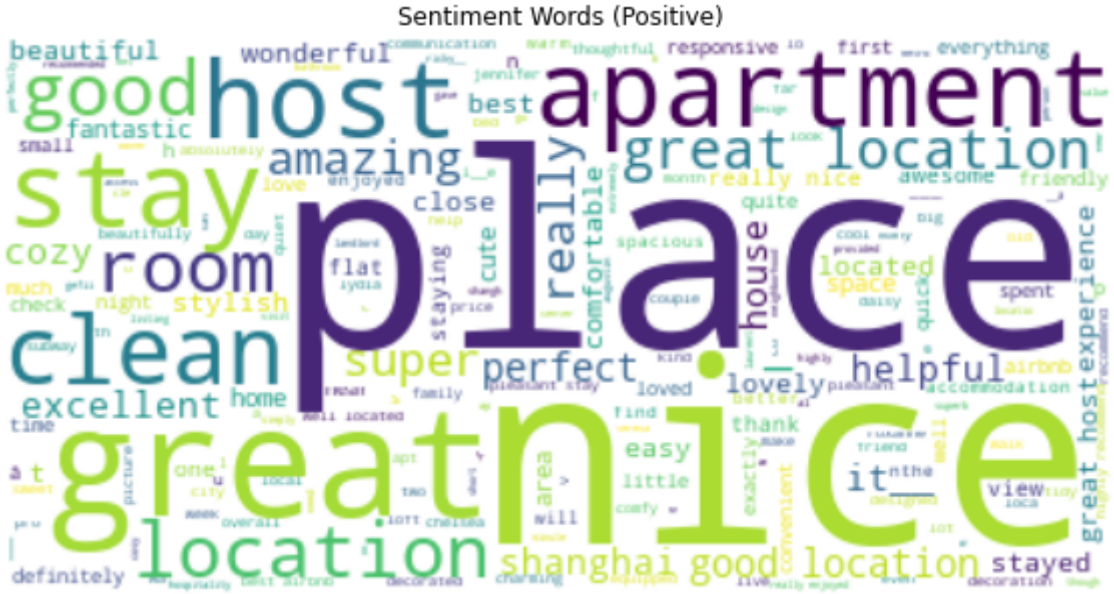
	comments	scores
16930	This apartment can accommodate up to 8 adults ...	-6
27707	Location is good, 20 min walk to the bund and ...	-7
16870	It was a poor experience. When we entered the...	-7
13863	The worst flat that I__e ever booked.\nDust, o...	-8
26962	Selfish, cheap, childish, terrible, dishonest ...	-9

In [274]:
```
positive = select[select['scores'] >= 5]
#negative = select[select['scores'] < -10]
print(f"ada {len(positive)} summary positif")
#print(f"ada {len(negative)} summary negativ")
```

```
ada 1128 summary positif
```

In [275]:
```
sitive
m wordcloud import WordCloud
.figure( figsize=(10,10) )
dcloud = WordCloud(background_color='white', relative_scaling=0, normalize_plurals = True).generate(posi
.title("Sentiment Words (Positive)")

.imshow(wordcloud, interpolation='bilinear')
.axis("off")
```

Out[275]: (-0.5, 399.5, 199.5, -0.5)



Tokyo

In [ ]:

In [276]:
```
#summary
select = pd.DataFrame()
select = tky_r[['comments']]
vect = CountVectorizer()
scores = []
for i in select['comments']:
    score = string_score(str(i),score_word_dict)
    scores.append(score)
select['scores'] = scores
select.sort_values(ascending = False,by = 'scores',inplace = True)
select.tail()
```

Out[276]:

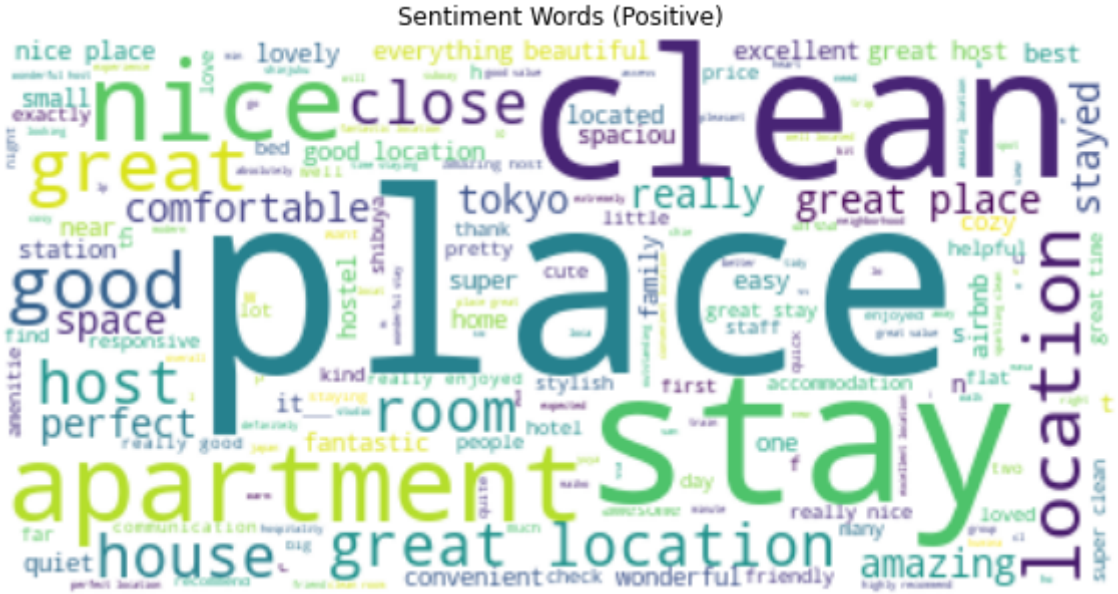
	comments	scores
17124	Es ist eine schÃ¶ne Unterkunft. Sie war grÃ¶_e...	-23
16215	Die Wohnung von Emily ist sehr gut gelegen. Ma...	-23
20606	Extremely terrible experience! The owner is a...	-25
16301	Wir haben zweimal in Emily__ Unterkunft verbra...	-29
21988	Wer typisches japanisches Wohnen in der Gro_st...	-34

In [277]:
```
positive = select[select['scores'] >= 5]
#negative = select[select['scores'] < -10]
print(f"ada {len(positive)} summary positif")
#print(f"ada {len(negative)} summary negativ")
```

```
ada 7001 summary positif
```

In [278]:
```
#Positive
from wordcloud import WordCloud
plt.figure( figsize=(10,10) )
wordcloud = WordCloud(background_color='white', relative_scaling=0, normalize_plurals = True).generate(p
plt.title("Sentiment Words (Positive)")

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

Out[278]: (-0.5, 399.5, 199.5, -0.5)



Singapore

In [279]:
```
#summary
from sklearn.feature_extraction.text import CountVectorizer
select = pd.DataFrame()
select = sp_r[['comments']]
vect = CountVectorizer()
scores = []
for i in select['comments']:
    score = string_score(str(i),score_word_dict)
    scores.append(score)
select['scores'] = scores
select.sort_values(ascending = False,by = 'scores',inplace = True)
select.tail()
```

Out[279]:

	comments	scores
369	Personally i wouldnt recommend staying here. W...	-16
2209	Leider hat die Unterkunft nicht ganz unseren V...	-16
2525	The experience in Rajan__ house had NOT A THIN...	-18
6915	Die Unterkunft an sich ist in Ordnung. Im Prin...	-19
5103	We stayed at this location between 5-12 Nov 20...	-30

In [280]:
```
positive = select[select['scores'] >= 5]
#negative = select[select['scores'] < -10]
print(f"ada {len(positive)} summary positif")
#print(f"ada {len(negative)} summary negativ")
```

```
ada 3304 summary positif
```

In [281]:
```
#Positive
from wordcloud import WordCloud
plt.figure( figsize=(10,10) )
wordcloud = WordCloud(background_color='white', relative_scaling=0, normalize_plurals = True).generate(p
plt.title("Sentiment Words (Positive)")

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

Out[281]: (-0.5, 399.5, 199.5, -0.5)

Istanbul

In [282]:
```
#summary
from sklearn.feature_extraction.text import CountVectorizer
select = pd.DataFrame()
select = istanbul[['comments']]
vect = CountVectorizer()
scores = []
for i in select['comments']:
    score = string_score(str(i),score_word_dict)
    scores.append(score)
select['scores'] = scores
select.sort_values(ascending = False,by = 'scores',inplace = True)
select.tail()
```

Out[282]:

	comments	scores
8401	Ich habe die Buchung aufgrund der positiven Er...	-20
13295	Die Wohnung ist sehr zentral, 1 Minute vom Tak...	-20
20222	Die Wohnung war genau nach meinen Erwartungen....	-26
20551	Wir waren unter den ersten GÃ¤sten in Ahmets W...	-29
2279	Ich habe stolze 5 Monate in Istanbul verbracht...	-41

In [283]:
```
positive = select[select['scores'] >= 5]
#negative = select[select['scores'] < -10]
print(f"ada {len(positive)} summary positif")
#print(f"ada {len(negative)} summary negativ")
```

```
ada 11517 summary positif
```

In [284]:
```
#Positive
from wordcloud import WordCloud
plt.figure( figsize=(10,10) )
wordcloud = WordCloud(background_color='white', relative_scaling=0, normalize_plurals = True).generate(p
plt.title("Sentiment Words (Positive)")

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

Out[284]:
```
(-0.5, 399.5, 199.5, -0.5)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: