AirBnb in NYC: Understanding the market

Why do these data matter?

These data were collected to understand host/guest transactions on the AirBnb website to ensure that AirBnb is successful. AirBnb succeeds when the users succeed and consequentially continue to partake in business on the site. Ultimately, this means the key focus of the users, or hosts, is to make money. How do these data inform how hosts can make the most money?

The most important lever to profit is the price of the listing, meaning it is an effective measure of success when extraneous factors are taken into account.

These factors may include:

- Demand for listings in certain locations
- Customer satisfaction
- Accommodations

These data provide insight into a listings location, guest satisfaction at that listing, as well as attributes of the property. Through the 60 variables that we will consider for analysis, there is plenty of information to understand contributing factors to hosts' successes to determine appropriate pricing.

Data description

accomodates: The maximum number of people the rental can accomodate *(numeric)*
availability_60: Number of days the space is available within the next 60 days *(numeric)*
bathrooms: The number of bathrooms in the rental *(numeric)*
bed_type: The type of bed available *(categorical)*
bedrooms: The number of separate bedrooms in the rental *(numeric)*
beds: The number of beds in the rental *(numeric)*
calculated_host_listings_count_entire_homes: Number of listings a host has where an entire home is offered *(numeric)*
calculated_host_listings_count_private_rooms: Number of listings a host has with private rooms offered *(numeric)*
calendar_last_scraped: Date the calendar information was last pulled *(date)*
calendar_updated: Last time the host updated their calendar information *(date)*
cancellation_policy: Ranking of how strict the host is on their cancellation policy *(categorical)*
city: The city/borough the listing is located in *(categorical)*
cleaning_fee: The surcharge for cleaning the listing *(numeric)*
extra_people: The surcharge for each additional person staying in the rental *(numeric)*
first_review: The date of the first review of the listing *(date)*
guests_included: The number of guests included in the price *(numeric)*
host_has_profile_pic: Indicator of presence of profile picture on site *(Boolean)*
host_id: Host identifier *(numeric)*
host_identity_verified: Indicator of host's identity verification using offline source *(Boolean)*
host_is_superhost: Indicator of super host status (exemplary host behavior) *(Boolean)*
host_listings_count: The number of listings the host has *(numeric)*
host_location: The city the host is located in *(categorical)*
host_name: Name of host *(categorical)*
host_neighborhood: The neighborhood the host is located in *(categorical)*
host_response_rate: The rate which a host responds to contact from lessor *(numeric)*
host_response_time: Bucket of how long the host takes to respond *(categorical)*

host_since: The date of the first hosting *(date)*
host_verifications: Ways in which the host was verified *(categorical)*
instant_bookable: Indicator on ability to book instantly *(Boolean)*
is_location_exact: Indicator on if location is exact *(Boolean)*
last_review: Date the last review of the listing occured *(date)*
latitude: Latitude of the listing *(numeric)*
longitude: Longitude of the listing *(numeric)*
market: The market the listing is in (i.e., Adirondacks, NYC) *(categorical)*
maximum_nights: The maximum number of consecutive days the listing is available *(numeric)*
minimum_nights: The minimum number of days the place can be rented *(numeric)*
name: Given name of the listing *(categorical)*
neighborhood: The neighborhood the listing is located in *(categorical)*
number_of_reviews: The number of reviews on a listing *(numeric)*
number_of_reviews_ltm: The number of reviews on a long term listing *(numeric)*
price: The nightly price of a listing *(numeric)*
property_type: The residence type of the listing *(numeric)*
require_guest_phone_verification: Indicator on whether phone verification of guest is requied *(Boolean)*
require_guest_profile_picture: Indicator on whether host requires guest to post profile picture *(Boolean)*
review_scores_accuracy: The average rating of all guests on the accuracy of the listing's description *(numeric)*
review_scores_checkin: The average rating of guests' review of the check in process *(numeric)*
review_scores_cleanliness: The average rating of guests' review of cleanliness *(numeric)*
review_scores_communication: The average rating of guests' review on host communication *(numeric)*
review_scores_location: The average rating of guests' review of listing location *(numeric)*
review_scores_rating: The average rating of guests' review of overall host rating *(numeric)*
review_scores_value: The average rating of guests' review of listing value *(numeric)*
reviews_per_month: The number of reviews a listing gets per month *(numeric)*
room_type: The type of space that is available for rent *(categorical)*
security_deposit: The value of the security deposit required before rent *(numeric)*
smart_location: The city of the listing - cleaned *(categorical)*
state: The state the listing is in *(categorical)*
street: Another variable for city *(categorical)*
zipcode: The zipcode the listing is in *(numeric)*

Duplicate data:

- Several minimum/maximum night variables were removed since they convey the same information
  - minimum_minimum_nights,maximum_minimum_nights,minimum_maximum_nights, maximum_maximum_nights, minimum_nights_avg_ntm, maximum_nights_avg_ntm
- Availability variables were highly correlated and removed from analysis consideration
  - availability_30, availability_365, availability_90, has_availability
- Calculated host listing variables contained the same information, and therefore only one pair is needed
  - calculated_host_listings_count, calculated_host_listings_count_shared_rooms
- Weekly and monthly price were highly correlated with price
- Country and country code contained same info (all constant values)
- Requires_license is exlusively true except for 1 value and therefore not informative

Missing values:

Many attributes have thousands of missing values. This is due to the nature of AirBnb's site - reviewers are not required to provide feedback on every category, and hosts do not have to provide information in every single field. Many of the fields available on the site do not apply to every single listing, meaning that often these fields are left blank and it is not a result of error. Certain fields had a high frequency of missing values (e.g. square feet had over 90% missing values), in which we

determined not to consider those variables as eligible for analysis. Out of the variables worth considering, the highest percentage of missing values is less than 40%, where we chose to impute the values by using the median of the available values to reduce changes to the shape of the distribution. For categorical variables, the values were not imputed, as "NaN" can serve as it's own factor level.

A key target variable for prediction is "price" which has 10 zero values which will be considered missing and included in imputation, since price is required for listings.

Host neighborhood was not considered for imputation, since there are many variables that contain similar information that can be used for analysis.

First review and last review were imputed with the date 08/01/2019 with the assumption that dates farther in the past provide information about host experience, which is information we did not want to provide for missing values.

Cleaning the data

In [8]:
```
import pandas as pd
import numpy as np
import pandas_profiling as prof
from decimal import Decimal
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt

data = pd.read_csv("C:/Users/jazis/OneDrive/Documents/SMU/Fall \'19/Machine Learning I/NY Dataset/listings.csv")
```

In [9]:
```
#drop redundant info and fields not useful for analysis
sub=data.drop(['id','listing_url','scrape_id','last_scraped','summary','space','desc
ription','experiences_offered'
               , 'neighborhood_overview', 'notes', 'transit', 'access', 'interaction'
, 'house_rules',
               'thumbnail_url', 'medium_url', 'picture_url', 'xl_picture_url', 'host_
url', 'host_thumbnail_url',
               'host_picture_url', 'country_code', 'country','amenities', 'minimum_mi
nimum_nights',
               'maximum_minimum_nights','minimum_maximum_nights', 'maximum_maximum_ni
ghts','minimum_nights_avg_ntm',
               'maximum_nights_avg_ntm', 'availability_30', 'availability_365','avail
ability_90','has_availability',
                'calculated_host_listings_count','calculated_host_listings_count_shar
ed_rooms',
                'is_business_travel_ready','host_about', 'host_acceptance_rate', 'hos
t_total_listings_count',
               'jurisdiction_names','license','monthly_price','square_feet','weekly_p
rice', 'requires_license'], axis=1)
print(list(sub.columns))
```

```
['name', 'host_id', 'host_name', 'host_since', 'host_location', 'host_response_tim
e', 'host_response_rate', 'host_is_superhost', 'host_neighbourhood', 'host_listings_
count', 'host_verifications', 'host_has_profile_pic', 'host_identity_verified', 'str
eet', 'neighbourhood', 'neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'ci
ty', 'state', 'zipcode', 'market', 'smart_location', 'latitude', 'longitude', 'is_lo
cation_exact', 'property_type', 'room_type', 'accommodates', 'bathrooms', 'bedroom
s', 'beds', 'bed_type', 'price', 'security_deposit', 'cleaning_fee', 'guests_include
d', 'extra_people', 'minimum_nights', 'maximum_nights', 'calendar_updated', 'availab
ility_60', 'calendar_last_scraped', 'number_of_reviews', 'number_of_reviews_ltm', 'f
irst_review', 'last_review', 'review_scores_rating', 'review_scores_accuracy', 'revi
ew_scores_cleanliness', 'review_scores_checkin', 'review_scores_communication', 'rev
iew_scores_location', 'review_scores_value', 'instant_bookable', 'cancellation_polic
y', 'require_guest_profile_picture', 'require_guest_phone_verification', 'calculated
_host_listings_count_entire_homes', 'calculated_host_listings_count_private_rooms',
'reviews_per_month']
```

In [10]:
```
def money_to_decimal(x):
    x = x.replace("$", "").replace(",", "").replace(" ", "")
    return float(x)
def rem_percent(x):
    x=x.replace("%","")
    return float(x)/100
def truncate(n):
    return int(n * 1000) / 1000
```

In [11]:
```
#converts objects with money values into decimal values to become continous attribut
e
sub.cleaning_fee = sub.cleaning_fee.astype(str)
sub.extra_people = sub.extra_people.astype(str)
sub.security_deposit = sub.security_deposit.astype(str)
sub.price = sub.price.astype(str)
sub.loc[:,'price'] = sub.loc[:,'price'].apply(money_to_decimal)
sub.loc[:,'cleaning_fee'] = sub.loc[:,'cleaning_fee'].apply(money_to_decimal)
sub.loc[:,'extra_people'] = sub.loc[:,'extra_people'].apply(money_to_decimal)
sub.loc[:,'security_deposit'] = sub.loc[:,'security_deposit'].apply(money_to_decimal
)

#imputations
sub['price']=sub.price.mask(sub.price == 0,sub.price.median())
sub.cleaning_fee=sub.cleaning_fee.fillna(sub.cleaning_fee.median())
sub.first_review=sub.first_review.fillna('2019-08-01')
sub['first_review'] =  pd.to_datetime(sub['first_review'],
                                format='%Y-%m-%d')
sub.host_response_rate = sub.host_response_rate.astype(str)
sub.loc[:,'host_response_rate'] = sub.loc[:, 'host_response_rate'].apply(rem_percent
)
sub.host_response_rate=sub.host_response_rate.fillna(sub.host_response_rate.median
())
sub['host_since'] =  pd.to_datetime(sub['host_since'],
                                format='%Y-%m-%d')
sub.last_review=sub.last_review.fillna('2019-08-01')
sub['last_review'] =  pd.to_datetime(sub['last_review'],
                                format='%Y-%m-%d')
sub.review_scores_accuracy=sub.review_scores_accuracy.fillna(truncate(sub.review_sco
res_accuracy.median()))
sub.review_scores_checkin=sub.review_scores_checkin.fillna(truncate(sub.review_score
s_checkin.median()))
sub.review_scores_cleanliness=sub.review_scores_cleanliness.fillna(truncate(sub.revi
ew_scores_cleanliness.median()))
sub.review_scores_communication=sub.review_scores_communication.fillna(truncate(sub.
review_scores_communication.median()))
sub.review_scores_location=sub.review_scores_location.fillna(truncate(sub.review_sco
res_location.median()))
sub.review_scores_rating=sub.review_scores_rating.fillna(truncate(sub.review_scores_
rating.median()))
sub.review_scores_value=sub.review_scores_value.fillna(truncate(sub.review_scores_va
lue.median()))
sub.reviews_per_month=sub.reviews_per_month.fillna(sub.reviews_per_month.median())
sub.security_deposit=sub.security_deposit.fillna(sub.security_deposit.median())
```

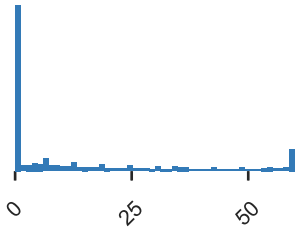
Examining the data

In [176]:
```
sub.profile_report(title="AirBnb Profiling", correlation_overrides=["recclass"])
#profile.to_file(output_file=("/Users/ksomes/Downloads/airbnb_profile.html"))
```

availability_60

Numeric

Distinct count	61
Unique (%)	0.1%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	15.47194253
Minimum	0
Maximum	60
Zeros (%)	42.4%



Toggle details

bathrooms

Numeric

Distinct count	17
Unique (%)	< 0.1%
Missing (%)	0.1%
Missing (n)	56
Infinite (%)	0.0%
Infinite (n)	0
Mean	1.144187428
Minimum	0
Maximum	15.5

Out[176]:

There are potential outliers with the following variables:

- Maximum nights: the hightest values for this variable seem to be placeholders, indicating that there is no specific maximum number of days for a guest's stay (ex: 999999 and 20000000) and the host put in meaningless values (an assumption we are making).
- Minimum nights: it appears some of these values might be user error (for example, 5 instances say the minimum number of nights is 500, and another says the minimum number is 1250).
- Reviews per month: 95% of the observations have less than 5 reviews per month. The instances where there are 20+ reviews per month are very rare, and require careful consideration when including in analysis, although there is nothing to indicate that there has been an error. For some analysis, focusing on 95% of the data and leaving out the extreme values would be beneficial.

Skewed variables:

All of the continuous variables are strongly right skewed, with the exception of the review scores for hosts which have a left skew. Possible transformations will be needed for certain learning techniques.

Descriptive Statistics of Key Variables

Price:
As shown in the table below, the prices are skewed right (mean is greater than the median), meaning there are more listings with a price below the average than above. 75% of the prices per night are 175 dollars or less, but there are high-end options up to 10,000 dollars a night. The prices are widely spread, as shown by the large range (99,990) and standard deviation (236).

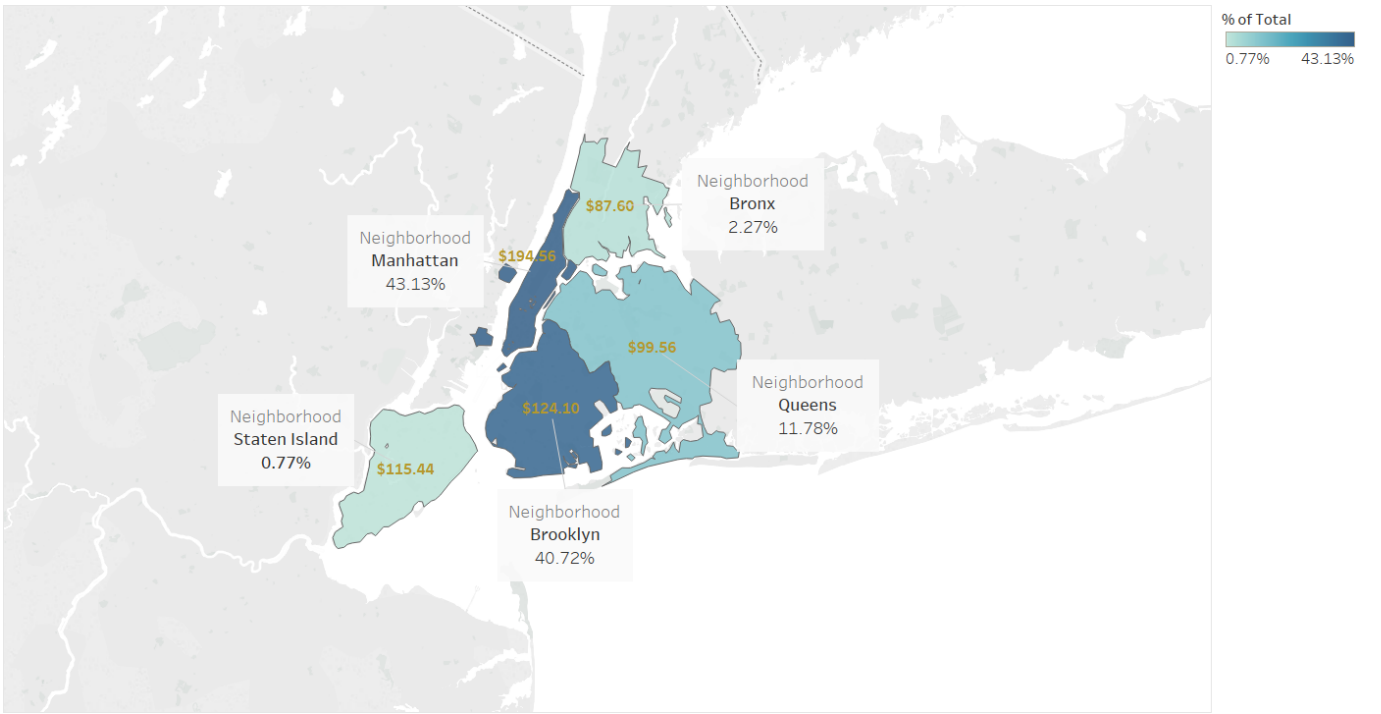
In [15]:
```
#price
sub.price.describe()
sub.price.std()

plt.style.use('ggplot')

from PIL import Image
#importing viz created in tableau
im_frame = Image.open('C:/Users/jazis/OneDrive/Documents/SMU/Fall \'19/Machine Learn
ing I/NY Dataset/Avg Price x Neighborhood (Map).png')

print('Average Airbnb Rental Price by Neighborhood'.center(110))
im_frame
```

Out[15]:
```
count    48864.000000
mean       151.474664
std        236.576536
min         10.000000
25%         69.000000
50%        105.000000
75%        175.000000
max      10000.000000
Name: price, dtype: float64
```

Out[15]:
```
236.57653562777037
```

Out[15]:

Average Airbnb Rental Price by Neighborhood



The neighbourhood_group_cleansed variable was grouped by zipcode and mapped in Tableau to create the visualization above. This view allows us to clearly see the distribution and price differential by neighboorhood. The neighborhoods are shaded based on the percentage of listings found in that area (darker shading indicates higher density). The average listing price for each neigherbhood overlays each area and the areas are labelled with the neighborhood name as well as the percentage of total listings found there. In New York, 84% of all listings are hosted in either Manhattan or Brooklyn, followed by Queens which host ~12% of the total listings. Manhattan listings, on average, are priced ~36% higher than those in Brooklyn and twice as high those in the Queens. The Bronx and Staten Island only account for ~3% of all listings.

Beds:
75% of the listings have 2 beds or less, with a greater concentration of listings having only one bed available. However, there are 5 listings with over 20 times as many beds as the median number of beds, indicating that this variable is highly skewed. The maximum number of beds a listing has is 40--which is much greater than the average number of 1.55. An early hypothesis is that these values with larger values are in specific markets; the market variable will likely have a relationship with the number of beds.

In [30]:

```
sub.beds.describe()
sub.beds.std()

#Bed Count by Listing
Bed_Counts = sub.groupby(['beds']).agg({'host_listings_count': 'count'})

BC_Percent_of_Total = Bed_Counts.groupby('beds').apply(lambda x:
    float(x.sum())/Bed_Counts.sum())

#Bed Count Price
BC_Avg_Price = sub.groupby(['beds']).agg({'price': 'mean'})
BC_Avg_Price = round(BC_Avg_Price,2)

Bed_Counts_Bar = BC_Percent_of_Total.plot(kind='bar', legend = False, title = 'Perce
ntage of Listings by Bed Count')
Bed_Counts_Bar.set_ylabel('Perenctage of Total Listing', fontsize=10)
Bed_Counts_Bar.set_xlabel('Number of Beds in Listing', fontsize=10)

BC_Avg_Price_Bar = BC_Avg_Price.plot(kind='bar', legend = False, title = 'Average Pr
ice of Listings by Bed Count')
BC_Avg_Price_Bar.set_ylabel('Average Price', fontsize=10)
BC_Avg_Price_Bar.set_xlabel('Number of Beds in Listing', fontsize=10)
```

Out[30]:
```
count    48822.000000
mean         1.548073
std          1.112344
min          0.000000
25%          1.000000
50%          1.000000
75%          2.000000
max         40.000000
Name: beds, dtype: float64
```

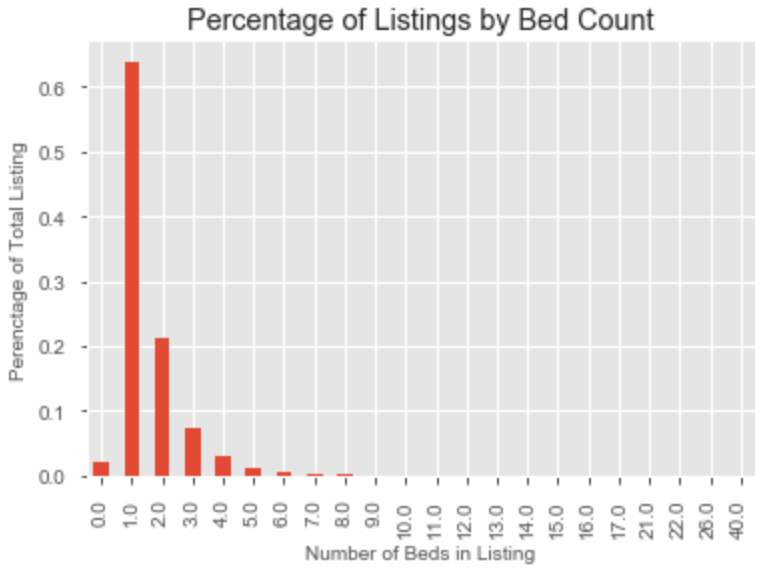
Out[30]:
```
1.1123440223779768
```

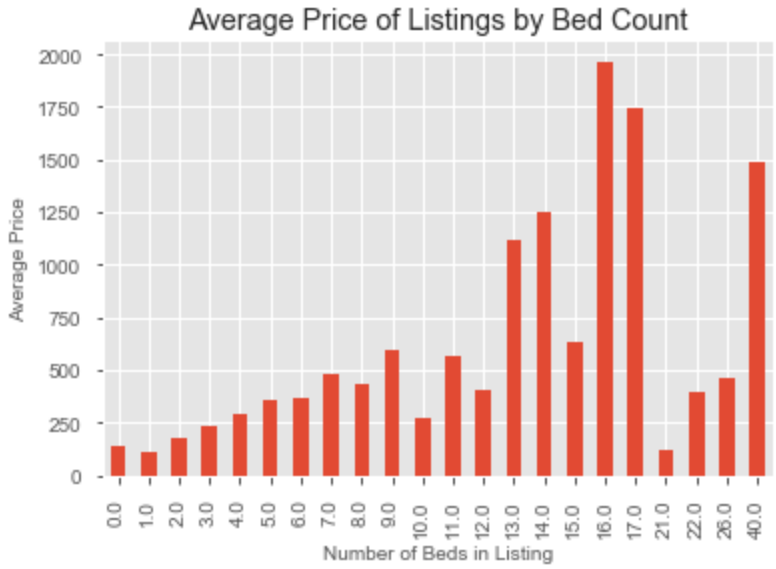
Out[30]:
```
Text(0, 0.5, 'Perenctage of Total Listing')
```

Out[30]:
```
Text(0.5, 0, 'Number of Beds in Listing')
```

Out[30]:
```
Text(0, 0.5, 'Average Price')
```

Out[30]:
```
Text(0.5, 0, 'Number of Beds in Listing')
```





The bar charts above show the distribution of listings by the number of beds available (top) as well as the average price by number of beds (bottom). These views are important because they allow us to see the relationship between price and frequency for the number of beds. We can see that the listings price tends to increase with the number of listings. The majority of listings with more than 2 rooms are priced over 250 dollars, but over 60 percent of all listings only have 1 bed and these listing on average are priced under $100.

Accomodates:
Similar to the "beds" variable, there is a much greater concentration of listings that accomodate 1-4 people, but there are listings with high accomodations that skew the distribution.

In [20]:

```
sub.accommodates.describe()
sub.accommodates.std()

#Listings by Accomodation Size
Accommodates_Freq = sub.groupby(['accommodates']).agg({'host_listings_count': 'coun
t'})
AF_Percent_of_Total = Accommodates_Freq.groupby('accommodates').apply(lambda x:
    float(x.sum())/Accommodates_Freq.sum())

#Price by Accomodation Size
Accomm_Avg_Price = sub.groupby(['accommodates']).agg({'price': 'mean'})
Accomm_Avg_Price = round(Accomm_Avg_Price,2)

#Print Bar Charts
Accommodates_Freq_Bar = AF_Percent_of_Total.plot(kind='bar', legend = False, title =
'Percentage of Listings by Accommodation Sizes')
Accommodates_Freq_Bar.set_ylabel('Percentage of Total Listings', fontsize=10)
Accommodates_Freq_Bar.set_xlabel('Number of People Listing Can Accommodate', fontsiz
e=10)

Accomm_Avg_Price_Bar = Accomm_Avg_Price.plot(kind='bar', legend = False, title = 'Av
erage Listing Price by Accommodation Capacity')
Accomm_Avg_Price_Bar.set_ylabel('Average Price', fontsize=10)
Accomm_Avg_Price_Bar.set_xlabel('Number of People Listing Can Accommodates', fontsiz
e=10)
```

Out[20]:
```
count    48864.000000
mean         2.846206
std          1.882121
min          1.000000
25%          2.000000
50%          2.000000
75%          4.000000
max         26.000000
Name: accommodates, dtype: float64
```

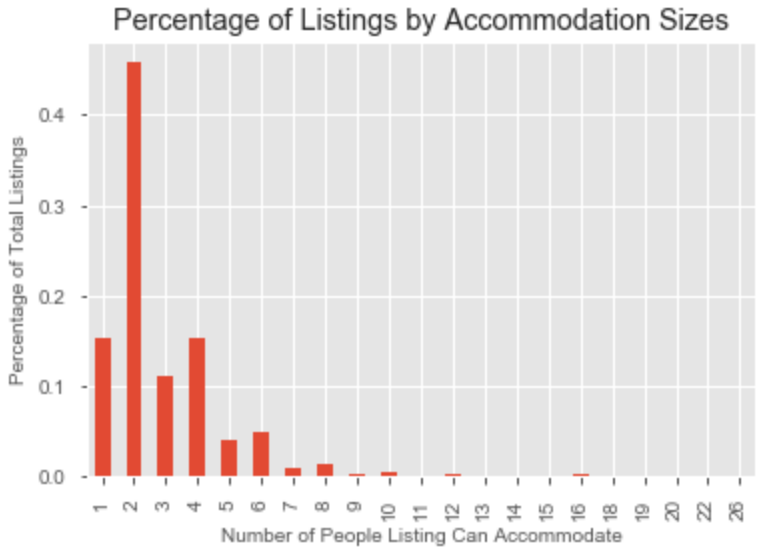
Out[20]:
```
1.8821207230880654
```

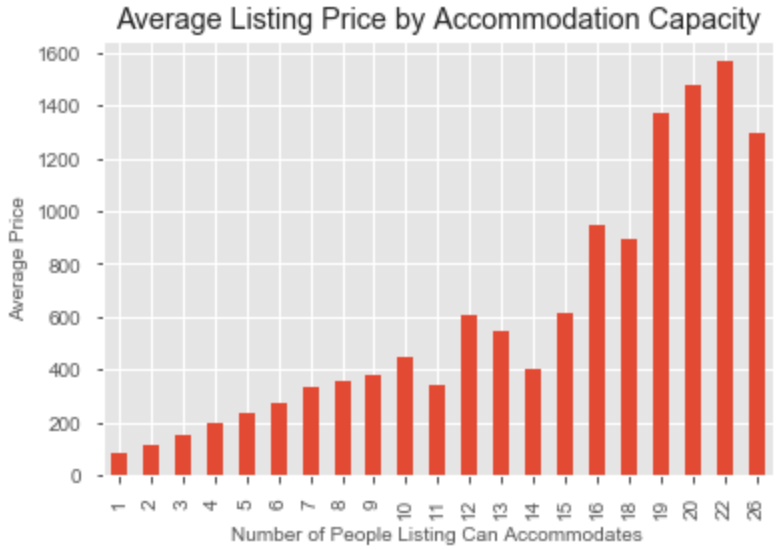
Out[20]:
```
Text(0, 0.5, 'Percentage of Total Listings')
```

Out[20]:
```
Text(0.5, 0, 'Number of People Listing Can Accommodate')
```

Out[20]:
```
Text(0, 0.5, 'Average Price')
```

Out[20]:
```
Text(0.5, 0, 'Number of People Listing Can Accommodates')
```





The bar charts above aggregate the number of people each listing can accommodate by their average price (bottom) and percentage of total (top). The average listing price increases with the number of guests a listing is able to accommodate. We are able to see that over 80% of all listings will accommodate up to 4 people -- this distribution aligns with what we found for the number of beds per listing.

Review scores rating: Most people give positive overall reviews for hosts, as shown by 75% of reviews being over a 9/10. This suggests that the rating most likely doesn't provide much information about good experiences but provides more about poor hosting experiences.

In [29]:

```
sub.review_scores_rating.describe()
sub.review_scores_rating.std()

#Score Rating Price
bins = pd.cut(sub['review_scores_rating'], [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 1
00])

Review_Score_Avg_Price = sub.groupby(bins)['review_scores_rating'].agg({'price': 'me
an'})
Review_Score_Avg_Price = round(Review_Score_Avg_Price,2)

#Score Rating Percentage
Review_Score_Count = sub.groupby(bins)['review_scores_rating'].agg(['count'])
RSC_Percent_of_Total = Review_Score_Count.groupby('review_scores_rating').apply(lamb
da x:
    float(x.sum())/Review_Score_Count.sum())

#Print Charts
Review_Score_Avg_Price_Bar = Review_Score_Avg_Price.plot(kind='bar', legend = False,
title = 'Average Price by Review Score Rating')
Review_Score_Avg_Price_Bar.set_xlabel('Review Score Rating', fontsize=10)
Review_Score_Avg_Price_Bar.set_ylabel('Average Listing Price', fontsize=10)

Review_Score_Count_Bar = RSC_Percent_of_Total.plot(kind='bar', legend = False, title
= 'Percentage of Total Listings by Review Score Rating')
Review_Score_Count_Bar.set_xlabel('Review Score Rating', fontsize=10)
Review_Score_Count_Bar.set_ylabel('Percentage of Total Listings', fontsize=10)
```

Out[29]:
```
count    48864.000000
mean        94.400786
std          7.636545
min         20.000000
25%         93.000000
50%         96.000000
75%         99.000000
max        100.000000
Name: review_scores_rating, dtype: float64
```

Out[29]:
```
7.636545344082264
```

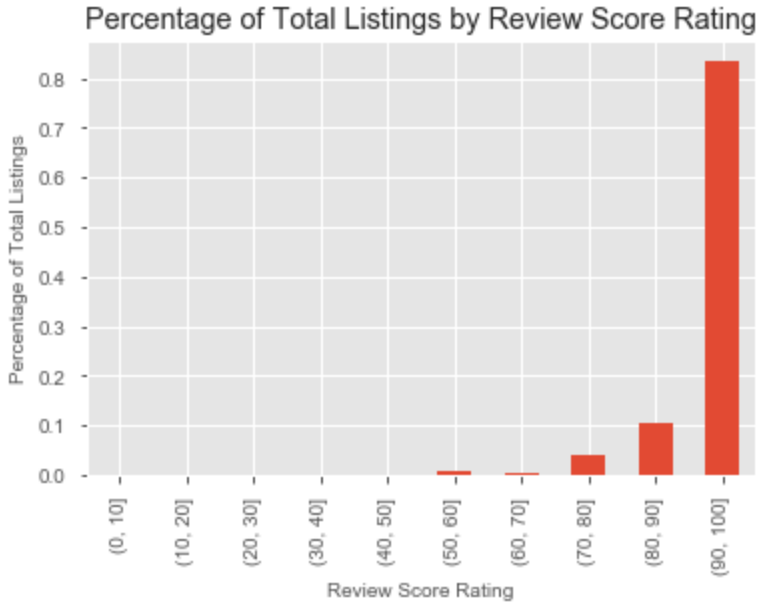
```
C:\Users\jazis\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launche
r.py:7: FutureWarning: using a dict on a Series for aggregation
is deprecated and will be removed in a future version
  import sys
```

Out[29]:
```
Text(0.5, 0, 'Review Score Rating')
```

Out[29]:
```
Text(0, 0.5, 'Average Listing Price')
```

Out[29]:
```
Text(0.5, 0, 'Review Score Rating')
```

Out[29]:
```
Text(0, 0.5, 'Percentage of Total Listings')
```



The charts above group the review score rating by 10-unit intervals. The first chart shows the average listing price by rating and we can clearly see that rating increases with price. The second chart shows the percentage of listings that fall into each rating bucket. Over 80% of all listings receive a score of at least 90 and over 90% score at least an 80. This is valuable from a host's perspective to see that most customers who leave a review had a pleasant exeperence. This also means that customer's who select a listing based on a high volume of positive reviews may also have high expectation for their stay.

Reviews per month:
Most hosts only receive around 1 review a month, which could be indicative of the typical rate that people list their property on the site. The maximum number of reviews in a month is 66, in which case they probably have many properties that they list (which is likely a less common occurrence).

In [111]:
```
sub.reviews_per_month.describe()
sub.reviews_per_month.std()
```

Out[111]:
```
count    48864.000000
mean         1.229748
std          1.530471
min          0.010000
25%          0.270000
50%          0.710000
75%          1.550000
max         66.610000
Name: reviews_per_month, dtype: float64
```

Out[111]:
```
1.5304708320260711
```

Maximum and minimum nights: As shown by the statistics for both minimum and maximum nights, there is a wide range of periods that properties can be available. Most properties are available for a minimum of 1-5 nights, and a maximum of 29-112 nights. The New York AirBnb market (from these values) appears to offer great flexibility for customers looking for extended stays.

In [113]:
```
sub.minimum_nights.describe()
sub.maximum_nights.describe()
```

Out[113]:
```
count    48864.000000
mean         7.093116
std         20.264170
min          1.000000
25%          1.000000
50%          2.000000
75%          5.000000
max       1250.000000
Name: minimum_nights, dtype: float64
```

Out[113]:
```
count    4.886400e+04
mean     4.541429e+04
std      9.715671e+06
min      1.000000e+00
25%      2.900000e+01
50%      1.124000e+03
75%      1.125000e+03
max      2.147484e+09
Name: maximum_nights, dtype: float64
```

In [115]:
```
%%html
<style>
  table {margin-left: 0 !important;}
</style>
```

Room type:
Most listings provide the entire space for the customer, followed closely by private spaces within the property. Only 2% of the listings offered shared spaces.

Entire home or apartment	Private room	Shared room
51.8%	45.8%	2.4%

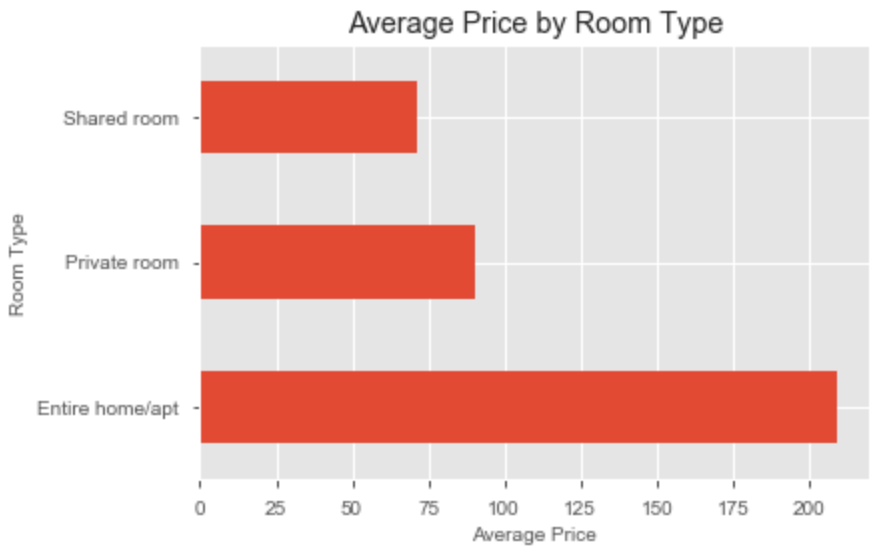
In [23]:
```
#Room Type Percentage
room_Type_Grouped = sub.groupby(['room_type']).agg({'host_listings_count': 'count'})
RT_Percent_of_Total = room_Type_Grouped.groupby('room_type').apply(lambda x:
    float(x.sum())/room_Type_Grouped.sum())

#Room Type Price
RT_Avg_Price = sub.groupby(['room_type']).agg({'price': 'mean'})
RT_Avg_Price = round(RT_Avg_Price,2)

#Print Bar Charts
RT_Price_Bar = RT_Avg_Price.plot(kind='barh', legend = False, title = 'Average Price
by Room Type')
RT_Price_Bar.set_xlabel('Average Price', fontsize=10)
RT_Price_Bar.set_ylabel('Room Type', fontsize=10)

RT_Per_Bar = RT_Percent_of_Total.plot(kind='barh', legend = False, title ='Percentag
e of Total Listings by Room Type')
RT_Per_Bar.set_xlabel('Percent of Total Listings', fontsize=10)
RT_Per_Bar.set_ylabel('Room Type', fontsize=10)
```

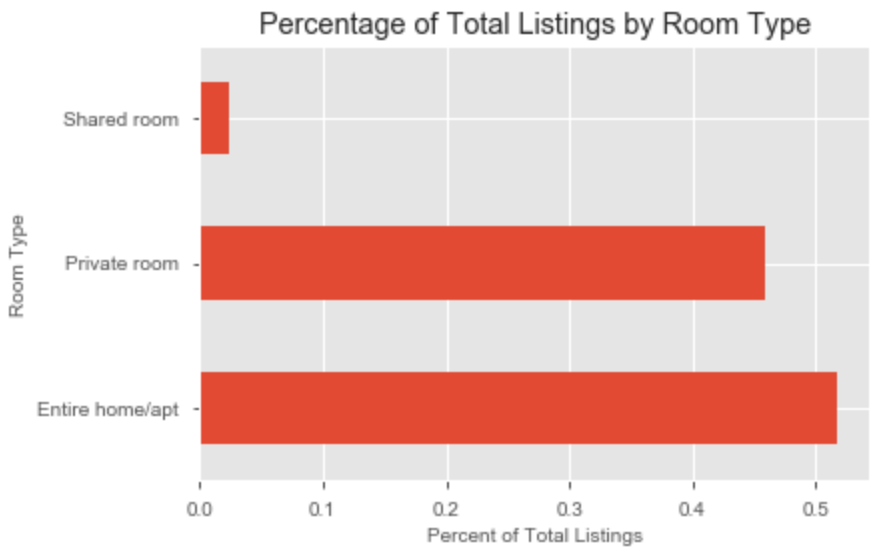
Out[23]: Text(0.5, 0, 'Average Price')

Out[23]: Text(0, 0.5, 'Room Type')

Out[23]: Text(0.5, 0, 'Percent of Total Listings')

Out[23]: Text(0, 0.5, 'Room Type')





The bar charts above show the distribution and average price of all listings by room type. We are able to see that while the majority of listings are for a private room or an entire home/apt, there is signifcant difference in their pricing. The average price of an entire home/apt is twice the price of a private room.

Property type: 98% of the properties fall into 5 types. Most of the relationships that we uncover will most likely be most powerful for apartment properties, since a great majority of the listings are apartments.

Apartment	House	Townhouse	Condominium	Loft
79.0%	7.9%	3.4%	3.1%	2.9%

Cancellation policy:
Most hosts offer some sort of cancellation flexibility, whether its a 14 day grace period or flexible to moderate policies. Less than 1% of the hosts have policies that are considered "strict"; it is most likely that it's in the host's long term interest to show some flexibility with guests.

Strict 14 with grace period	flexible	moderate	super strict 60	super strict 30	strict
44.9%	31.5%	23.3%	0.02%	0.01%	0.01%

In [31]:

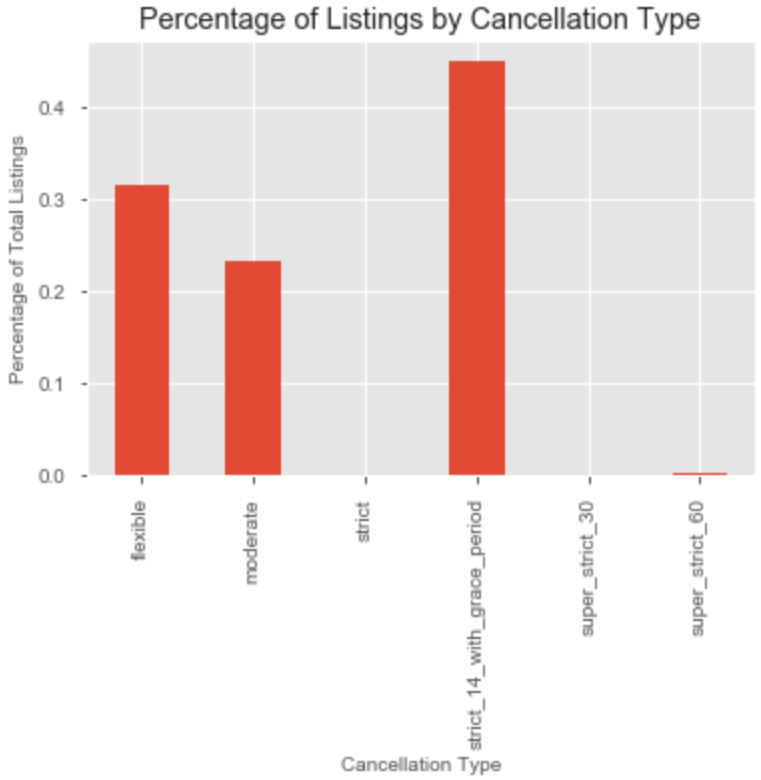
```
#Listings by Accomodation Size
Cancel_Type_Ct = sub.groupby(['cancellation_policy']).agg({'host_listings_count': 'c
ount'})
CTC_Percent_of_Total = Cancel_Type_Ct.groupby('cancellation_policy').apply(lambda x:
    float(x.sum())/Cancel_Type_Ct.sum())

#Price by Accomodation Size
CTC_Avg_Price = sub.groupby(['cancellation_policy']).agg({'price': 'mean'})
CTC_Avg_Price = round(CTC_Avg_Price,2)

#Print Bar Charts
CTC_Percent_of_Total_Bar = CTC_Percent_of_Total.plot(kind='bar', legend = False, tit
le = 'Percentage of Listings by Cancellation Type')
CTC_Percent_of_Total_Bar.set_ylabel('Percentage of Total Listings', fontsize=10)
CTC_Percent_of_Total_Bar.set_xlabel('Cancellation Type', fontsize=10)

CTC_Avg_Price_Bar = CTC_Avg_Price.plot(kind='bar', legend = False, title = 'Average
 Listing Price by Cancellation Type')
CTC_Avg_Price_Bar.set_ylabel('Average Price', fontsize=10)
CTC_Avg_Price_Bar.set_xlabel('Cancellation Type', fontsize=10)
```

Out[31]: Text(0, 0.5, 'Percentage of Total Listings')

Out[31]: Text(0.5, 0, 'Cancellation Type')

Out[31]: Text(0, 0.5, 'Average Price')

Out[31]: Text(0.5, 0, 'Cancellation Type')



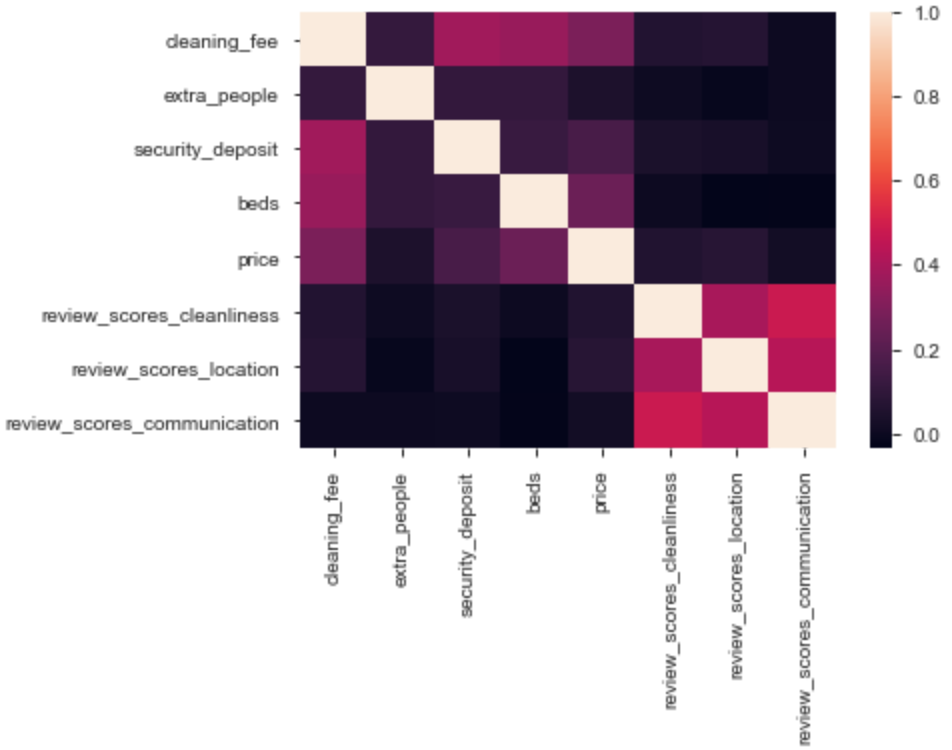
The charts above show the average price for each cancellation type (bottom) as well as the percentage of total listings with that cancellation type (top). From this view, we are able to see that the average price increases as the cancellation policy strictness increases. This information would be valuable for a host to know and consider. While having a super strict cancellation policy may offer the host more security and on average increases a listings price by 50-100 dollars, very few of their competitors would be this strict. This could indicate that customers may not be willing to select a listing that is not as flexible.

Relationships between continuous variables

Investigating correlated variables

In [122]:
```
contDF = sub.filter(['cleaning_fee','extra_people','security_deposit', 'accomodates'
, 'beds', 'price',
                     'review_scores_cleanliness', 'review_scores_location', 'review_s
cores_communication'], axis=1)
corr=contDF.corr()
sns.heatmap(corr,
        xticklabels=corr.columns,
        yticklabels=corr.columns)
```

Out[122]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3f2530b8>
```



Correlation does not appear to be a concern for the select continuous variables (price, cleaning fee, extra people, security deposit, accomodates, beds, review scores cleanliness, review scores location, and review scores communication). There appears to be somewhat high correlation with the review scores for the host, but nothing which would be of concern for analysis.

Pairwise relationships

Cleaning fee and security deposit: There is no clear pairwise relationship between the cleaning fee and the security deposit, although it does reveal that most listings have lower costs for each, as there as a greater concentration of points closer to the origin.
Extra people and security deposit: There doesn't appear to be a relationship between the security deposit required and the cost for extra people.
Accomodates and beds: There is a strong positive linear relationship between the number of beds and the nuber of people the listing accommodates; this makes intuitive sense and confirms variable accuracy.
Cleaning fee and cleanliness reviews: It seems that listings with lower cleaning fees have lower cleanliness ratings than those with higher cleaning fees. This could suggest that hosts with higher cleaning fees have higher standards of cleanliness and put more resources into maintaining cleanliness.

In [135]:

```
#cleaning_fee and extra_people and security_deposit
plt.subplot(2, 2, 1)
plt.scatter(sub.security_deposit,sub.cleaning_fee, alpha=0.5)
plt.title('Cleaning fee vs. Security deposit')
plt.xlabel('Security deposit')
plt.ylabel('Cleaning fee')
plt.xscale('log')
plt.subplot(2, 2, 2)
plt.scatter(sub.security_deposit,sub.extra_people, alpha=0.5)
plt.title('Extra people vs. Security deposit')
plt.xlabel('Security deposit')
plt.ylabel('Extra people')
plt.xscale('log')
plt.subplot(2, 2, 3)
plt.scatter(sub.accommodates,sub.beds, alpha=0.5)
plt.title('Accommodates vs. beds')
plt.xlabel('Accommodates')
plt.ylabel('Beds')
plt.subplot(2, 2, 4)
plt.scatter(sub.cleaning_fee,sub.review_scores_cleanliness, alpha=0.5)
plt.title('Cleaning fee vs. cleanliness reviews')
plt.xlabel('cleaning fee')
plt.ylabel('cleanliness score')

plt.tight_layout()
plt.show()
#accodomates and beds
#cleaning_fee and review_scores_cleanliness
#review_scores_checkin and #review_scores_communication
```

Out[135]: <matplotlib.axes._subplots.AxesSubplot at 0x1a228ef438>

Out[135]: <matplotlib.collections.PathCollection at 0x1a22968128>

Out[135]: Text(0.5, 1.0, 'Cleaning fee vs. Security deposit')

Out[135]: Text(0.5, 0, 'Security deposit')

Out[135]: Text(0, 0.5, 'Cleaning fee')

Out[135]: <matplotlib.axes._subplots.AxesSubplot at 0x1a228c9ac8>

Out[135]: <matplotlib.collections.PathCollection at 0x1a22995e80>

Out[135]: Text(0.5, 1.0, 'Extra people vs. Security deposit')

Out[135]: Text(0.5, 0, 'Security deposit')

Out[135]: Text(0, 0.5, 'Extra people')

Out[135]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2297b668>

Out[135]: <matplotlib.collections.PathCollection at 0x1a34a98c18>

Out[135]: Text(0.5, 1.0, 'Accommodates vs. beds')

Out[135]: Text(0.5, 0, 'Accommodates')

Out[135]: Text(0, 0.5, 'Beds')

Out[135]: <matplotlib.axes._subplots.AxesSubplot at 0x1a34a98f28>

Out[135]: <matplotlib.collections.PathCollection at 0x1a34acb9b0>

Out[135]: Text(0.5, 1.0, 'Extra people vs. Security deposit')

Out[135]: Text(0.5, 0, 'Security deposit')

Out[135]: Text(0, 0.5, 'Extra people')

Relationships between continuous and categorical variables

Differences between neighborhoods: There appears to be slight differences among security deposit values between NYC neighborhoods, but not so much of a difference between how many people the listings accommodate. **Differences between cancellation policies:** It appears that listings with a very strict 30 day cancellation policy tend to accommodate more people and have lower host reviews on average compared to listings with different cancellation policies.

There doesn't seem to be a difference between instant bookable listings and host review scores, nor location review scores and if the location was exact or not.

In [166]:
```
fig, ax= plt.subplots(3,2, figsize=(18, 15))
ax[0,0].set_yscale('log')
sns.boxplot(x='neighbourhood_group_cleansed', y='security_deposit', data=sub, ax=ax[
0,0])
sns.boxplot(x='neighbourhood_group_cleansed', y='accommodates', data=sub, ax=ax[0,1
])
sns.boxplot(x='cancellation_policy', y='accommodates', data=sub, ax=ax[1,0])
sns.boxplot(x='cancellation_policy', y='review_scores_rating', data=sub, ax=ax[1,1])
sns.boxplot(x='instant_bookable', y='review_scores_rating', data=sub, ax=ax[2,0])
sns.boxplot(x='is_location_exact', y='review_scores_location', data=sub, ax=ax[2,1])
```

Out[166]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3507b710>

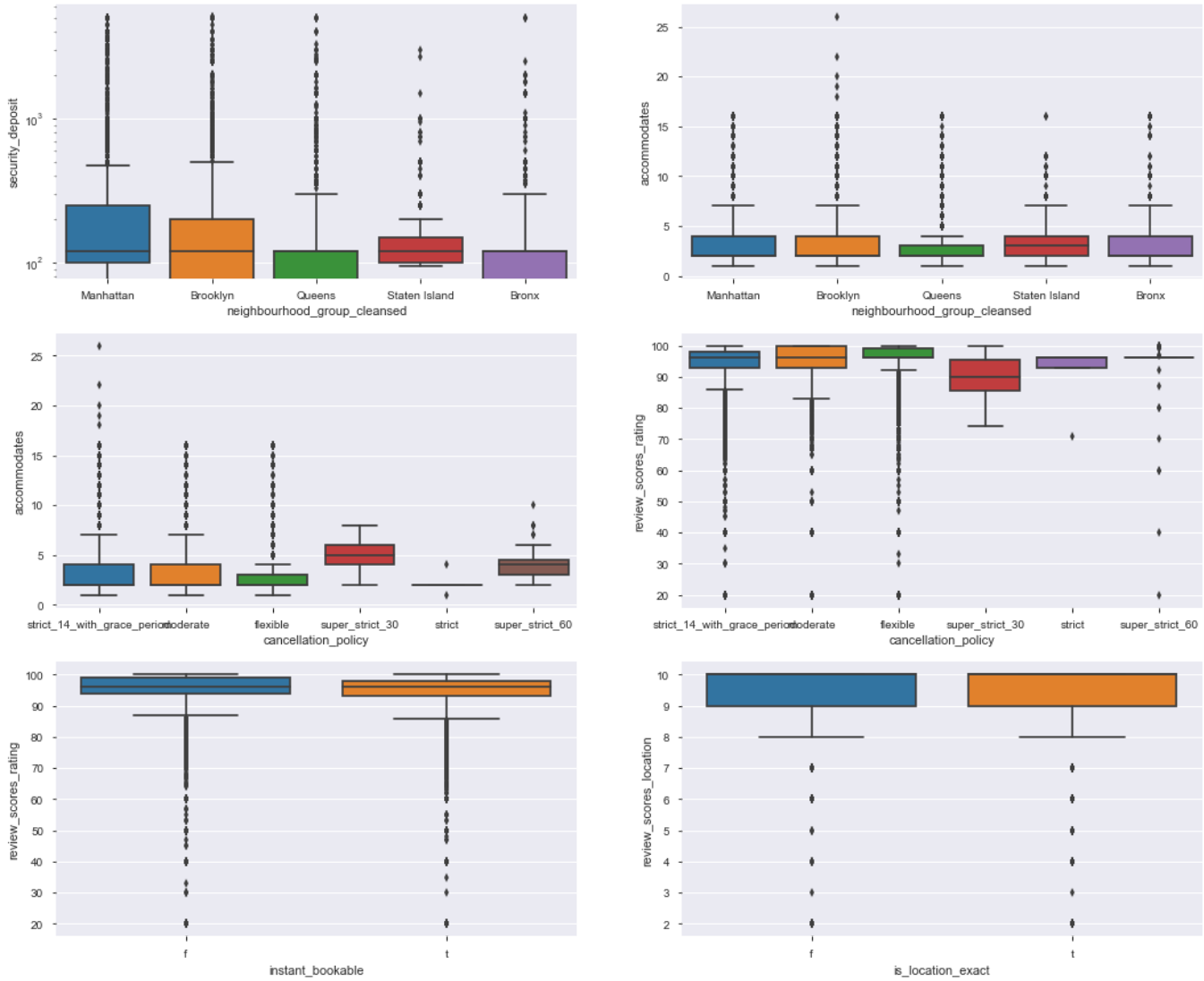
Out[166]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3507c908>

Out[166]: <matplotlib.axes._subplots.AxesSubplot at 0x1a433a9278>

Out[166]: <matplotlib.axes._subplots.AxesSubplot at 0x1a43399be0>

Out[166]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3782f588>

Out[166]: <matplotlib.axes._subplots.AxesSubplot at 0x1a46ac8ef0>

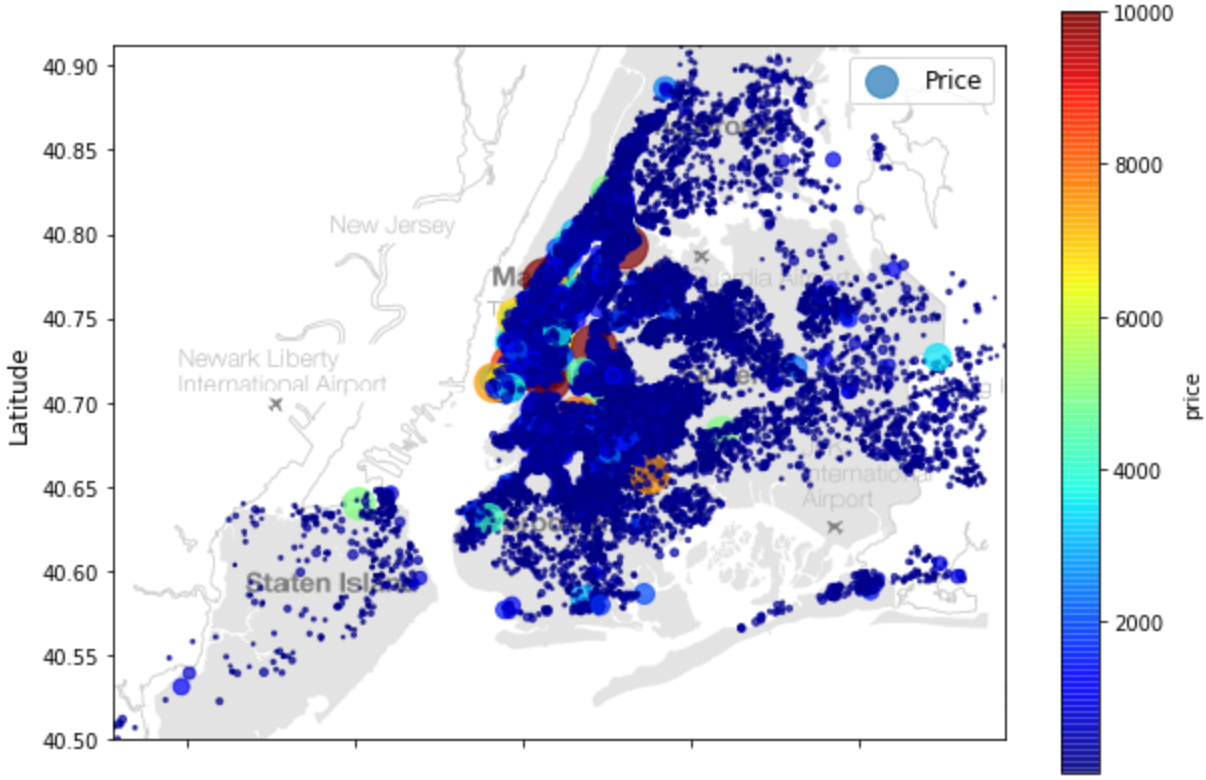


In [3]:

```
import numpy as np
import pandas as pd
import pandas_profiling as pd_prof
from decimal import Decimal

data = pd.read_csv("C:\\Users\\Yat\\Documents\\MSDS\\MSDS 7331\\ML_Lab_Data\\listings.csv")

sub=data.drop(['id','listing_url','scrape_id','last_scraped','summary','space','description','experiences_offered'
               , 'neighborhood_overview', 'notes', 'transit', 'access', 'interaction', 'house_rules',
               'thumbnail_url', 'medium_url', 'picture_url', 'xl_picture_url', 'host_url', 'host_thumbnail_url',
               'host_picture_url', 'country_code', 'country','amenities', 'minimum_minimum_nights',
               'maximum_minimum_nights','minimum_maximum_nights', 'maximum_maximum_nights','minimum_nights_avg_ntm',
               'maximum_nights_avg_ntm', 'availability_30', 'availability_365','availability_90','has_availability',
                'calculated_host_listings_count','calculated_host_listings_count_shared_rooms',
                'is_business_travel_ready','host_about', 'host_acceptance_rate', 'host_total_listings_count',
               'jurisdiction_names','license','monthly_price','square_feet','weekly_price', 'requires_license'], axis=1)

#functions courtesy of Karen
def money_to_decimal(x):
    x = x.replace("$", "").replace(",", "").replace(" ", "")
    return float(x)

def rem_percent(x):
    x=x.replace("%","")
    return float(x)/100
def truncate(n):
    return int(n * 1000) / 1000


#courtesy of Karen
#converts objects with money values into decimal values to become continous attribute
sub.cleaning_fee = sub.cleaning_fee.astype(str)
sub.extra_people = sub.extra_people.astype(str)
sub.security_deposit = sub.security_deposit.astype(str)
sub.price = sub.price.astype(str)
sub.loc[:,'price'] = sub.loc[:,'price'].apply(money_to_decimal)
sub.loc[:,'cleaning_fee'] = sub.loc[:,'cleaning_fee'].apply(money_to_decimal)
sub.loc[:,'extra_people'] = sub.loc[:,'extra_people'].apply(money_to_decimal)
sub.loc[:,'security_deposit'] = sub.loc[:,'security_deposit'].apply(money_to_decimal)

#imputations
sub['price']=sub.price.mask(sub.price == 0,sub.price.median())
sub.cleaning_fee=sub.cleaning_fee.fillna(sub.cleaning_fee.median())
sub.first_review=sub.first_review.fillna('2019-08-01')
sub['first_review'] =  pd.to_datetime(sub['first_review'],
                                format='%Y-%m-%d')
sub.host_response_rate = sub.host_response_rate.astype(str)
sub.loc[:,'host_response_rate'] = sub.loc[:, 'host_response_rate'].apply(rem_percent)
```

```
sub.host_response_rate=sub.host_response_rate.fillna(sub.host_response_rate.median
())
sub['host_since'] =  pd.to_datetime(sub['host_since'],
                                format='%Y-%m-%d')
sub.last_review=sub.last_review.fillna('2019-08-01')
sub['last_review'] =  pd.to_datetime(sub['last_review'],
                                format='%Y-%m-%d')
sub.review_scores_accuracy=sub.review_scores_accuracy.fillna(truncate(sub.review_sco
res_accuracy.median()))
sub.review_scores_checkin=sub.review_scores_checkin.fillna(truncate(sub.review_score
s_checkin.median()))
sub.review_scores_cleanliness=sub.review_scores_cleanliness.fillna(truncate(sub.revi
ew_scores_cleanliness.median()))
sub.review_scores_communication=sub.review_scores_communication.fillna(truncate(sub.
review_scores_communication.median()))
sub.review_scores_location=sub.review_scores_location.fillna(truncate(sub.review_sco
res_location.median()))
sub.review_scores_rating=sub.review_scores_rating.fillna(truncate(sub.review_scores_
rating.median()))
sub.review_scores_value=sub.review_scores_value.fillna(truncate(sub.review_scores_va
lue.median()))
sub.reviews_per_month=sub.reviews_per_month.fillna(sub.reviews_per_month.median())
sub.security_deposit=sub.security_deposit.fillna(sub.security_deposit.median())
```

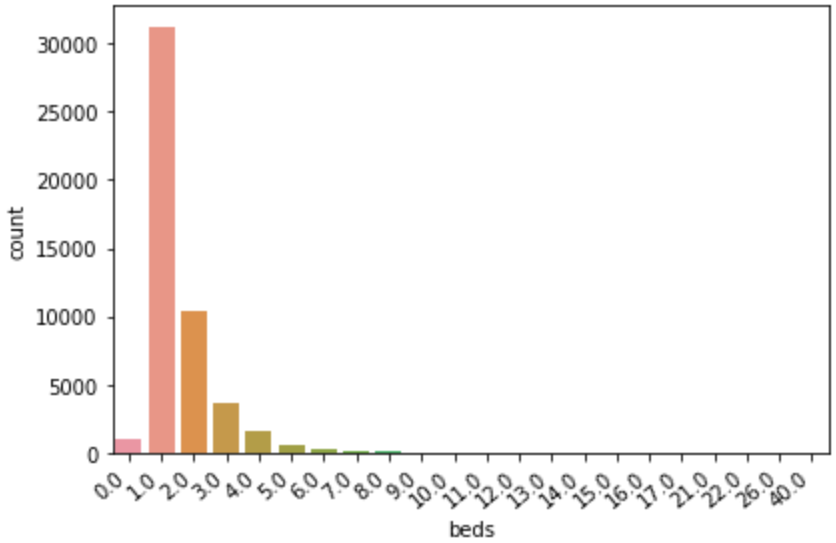
In [4]:
```
import geopandas as gpd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

%matplotlib inline


NYC_img=mpimg.imread("C:\\Users\\Yat\\Documents\\MSDS\\MSDS 7331\\ML_Lab_Data\\nyc-boros.png")
ax = sub.plot(kind="scatter", x="longitude", y="latitude",
    s=sub['price']/20, label="Price",
    c="price", cmap=plt.get_cmap("jet"),
    colorbar=True, alpha=0.7, figsize=(10,7),
)
plt.imshow(NYC_img, extent=[-74.244, -73.713, 40.5, 40.912], alpha=0.5)
plt.ylabel("Latitude", fontsize=12)
plt.xlabel("Longitude", fontsize=12)

plt.legend(fontsize=12)
plt.show()
```



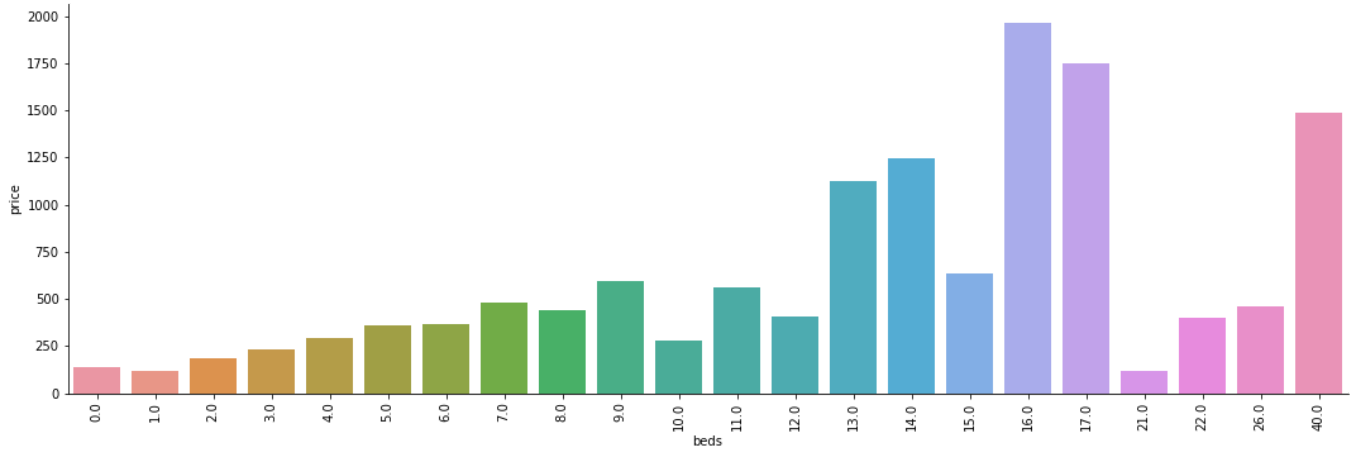
Projecting Rental Price Onto The NYC Boros Map

In the scatter plot, a high concentration of rentals were from the Manhattan and Brooklyn. Majority of the rental prices were below the 2000 price range. However there are some high rental price (8000 and above) spots within Manhattan area. Our preliminary analysis suggest that the highly concentrated rental areas correspond to certain landmarks and mass transit locations such as Time Square or La Guardia Airport

In [15]:
```
import seaborn as sns
ax=sns.countplot(x="beds", data=sub)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()

ax=sns.catplot(x="beds", y="price", kind="bar", data=sub, ci=None,height=5, aspect=3
)
ax.set_xticklabels(rotation=90)
```



Out[15]: <seaborn.axisgrid.FacetGrid at 0x1d6dac69358>

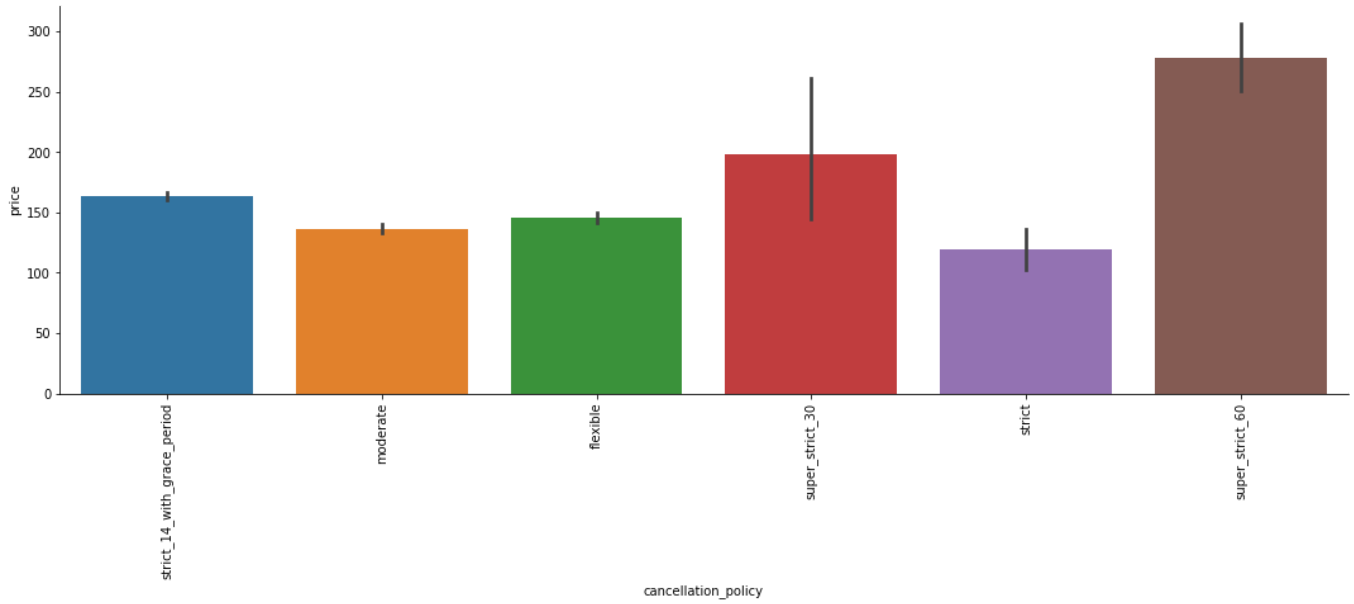


Beds

Most units have between 1 to 2 beds with price ranges up to 250. 0 bed option was shown in the plot and there could be reason such as these units may have alternative sleeping arrangement and will need to investigate further for explanations. The beds versus price plot is right skewed and there are outliers such as with 40 beds option the price was lower than the 16 or 17-bed options.

In [6]:
```
ax=sns.catplot(x="cancellation_policy", y="price", kind="bar", data=sub,height=5, aspect=3)
ax.set_xticklabels(rotation=90)
```

Out[6]: <seaborn.axisgrid.FacetGrid at 0x1d6e2f1d5c0>

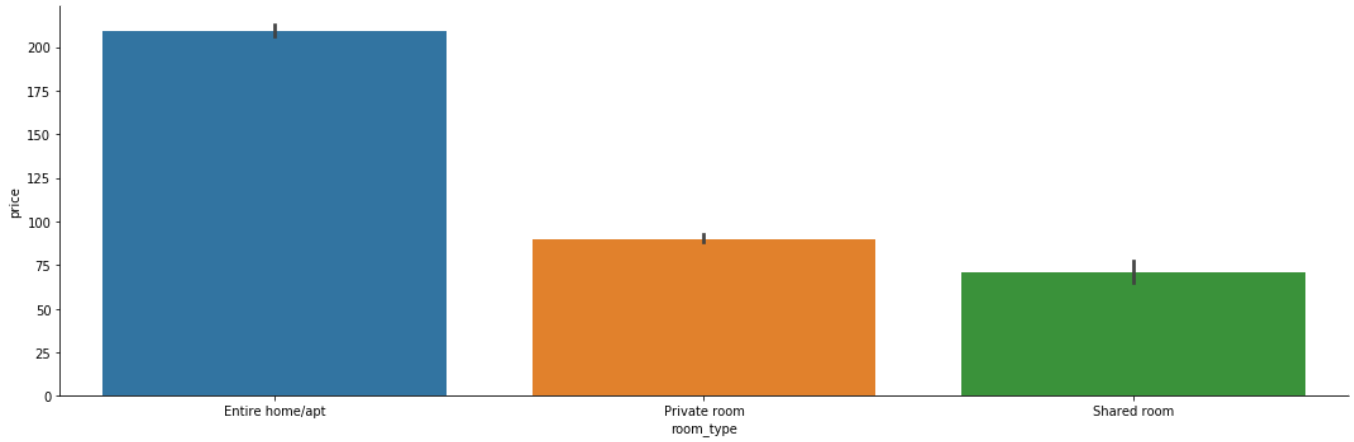


Cancellation Policy and Price

Rental Price over 200 usually has a (super strict 60) 60 days cancellation policy while the 140-150 units usually have a moderate to flixible cancallation. Units with >150 to <200 have a 14 days grace period and at 200, the cencellation policy is 30 days (super strict 20). Properties with a $< 150 price and strict cancellation policy will need to be further examine to determine if this could be grouped into either the super strict 30 or super strict 60 group.

In [7]:
```
sns.catplot(x="room_type", y="price", kind="bar", data=sub,height=5, aspect=3)
```

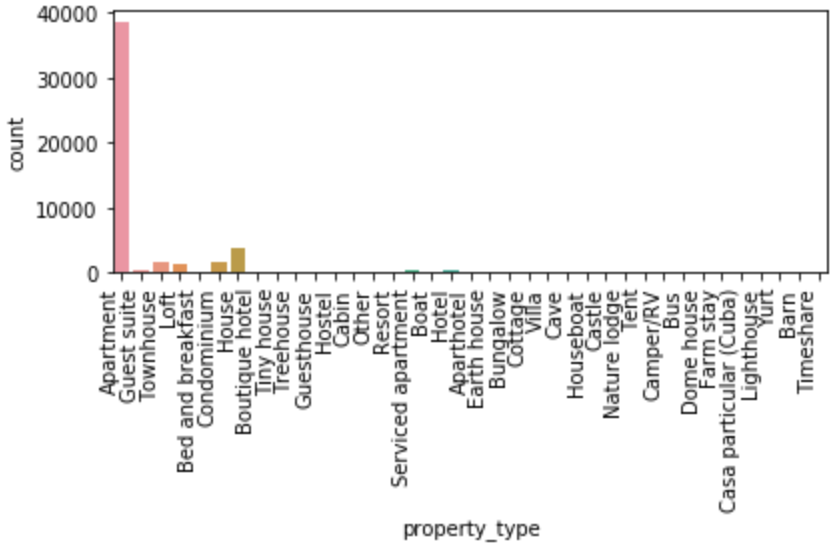
Out[7]: <seaborn.axisgrid.FacetGrid at 0x1d6ecf37a58>



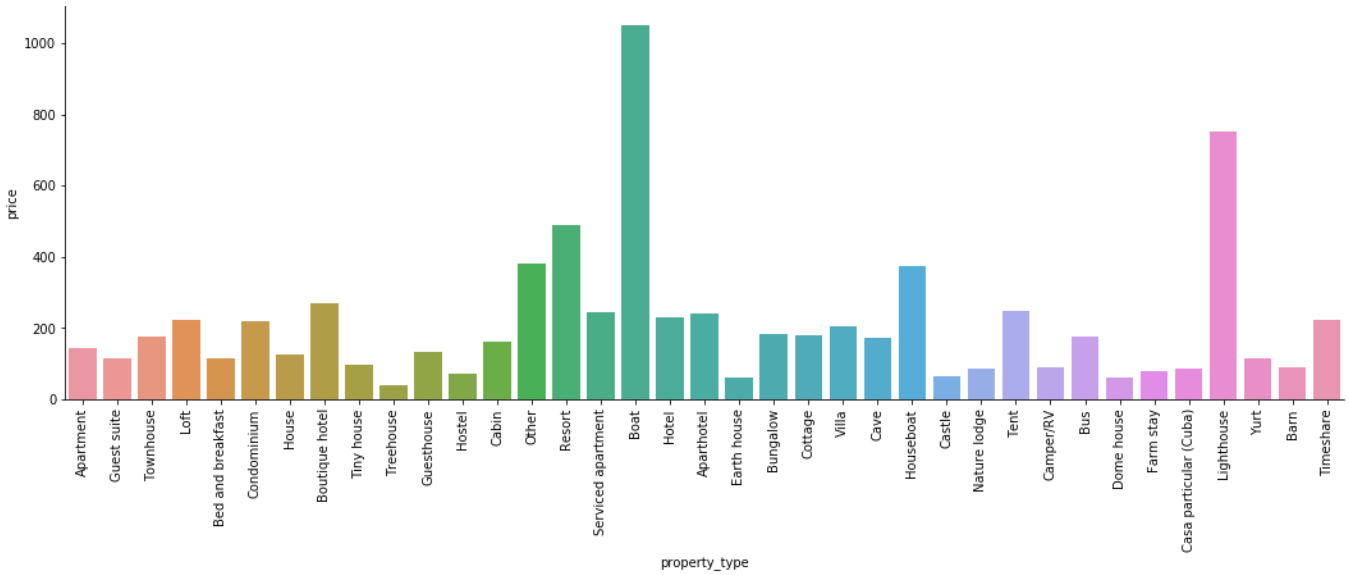
Room Type

In terms of room type, most prefer the entire home/ apartment option which costs approxmiately 200. For reference, standard deviation of price is 236.58. This is followed by private room at 90 and shared room at 70.

In [8]:
```
ax=sns.countplot(x="property_type", data=sub)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")
plt.tight_layout()
plt.show()

ax=sns.catplot(x="property_type", y="price", kind="bar", data=sub, ci=None,height=5,
aspect=3)
ax.set_xticklabels(rotation=90)
```



Out[8]: <seaborn.axisgrid.FacetGrid at 0x1d6da0e7668>

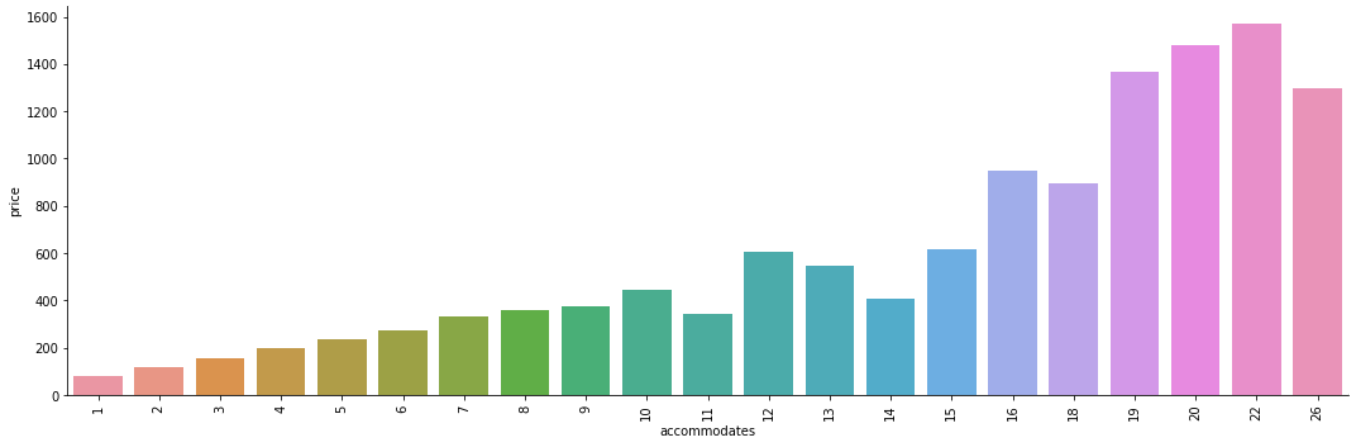


Property Type

In the property type by count plot, we see that the most popular and common type is an apartment. In the plot we can also see that boutique hotel is also a common choice for Airbnb renters after apartment. In the property type versus price, we can see that the rental price variable ranged from the $100 to over 1000$. Most of the rental properties were in the sub $500 range however we also see eunusual properties such as boat and lighthouses which could cost renters from >$ 500 to >$ 1000.

In [9]:
```
ax=sns.catplot(x="accommodates", y="price", kind="bar", data=sub, ci=None,height=5,
aspect=3)
ax.set_xticklabels(rotation=90)
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6eaff1438>
```

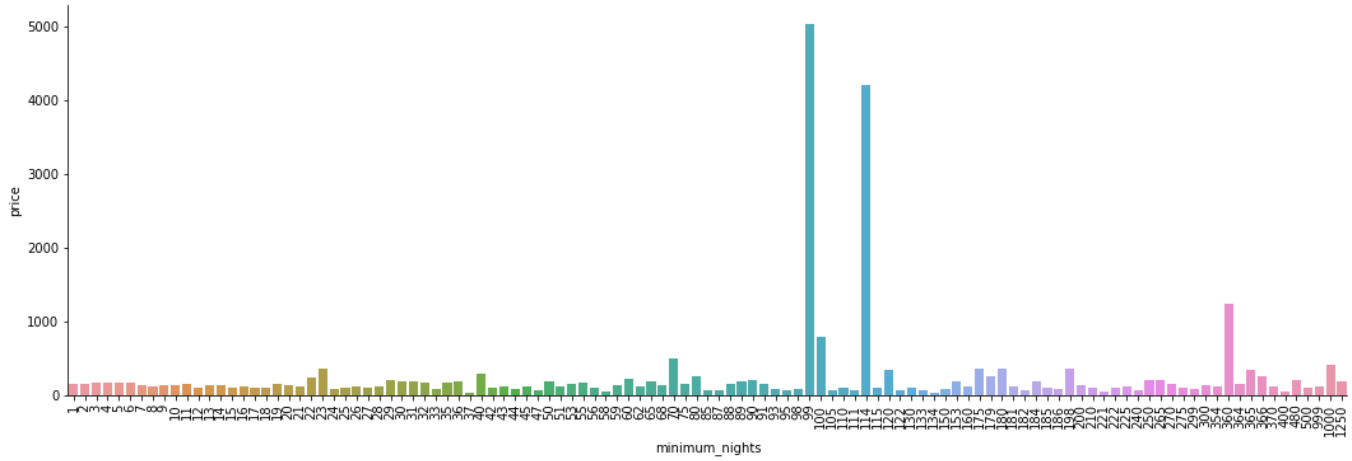


Accommodates

Properties that can accomodates between 1-4 guests were usually in the >= $200 range. For 5 to 9 guests, the cost is between $ > 200 to $400. In general, the rent price costs more if it can accommodate more guests. There were exceptions where a 11-guest, 14-guest, 18-guest and 26-guest units would cost less than a smaller units (10,13,16,22-guest variants)

In [10]:
```
plt.figure(figsize=(20,15))
ax=sns.catplot(x="minimum_nights", y="price", kind="bar", data=sub, ci=None,height=5
, aspect=3)
ax.set_xticklabels(rotation=90)
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6d7e7b358>

<Figure size 1440x1080 with 0 Axes>
```

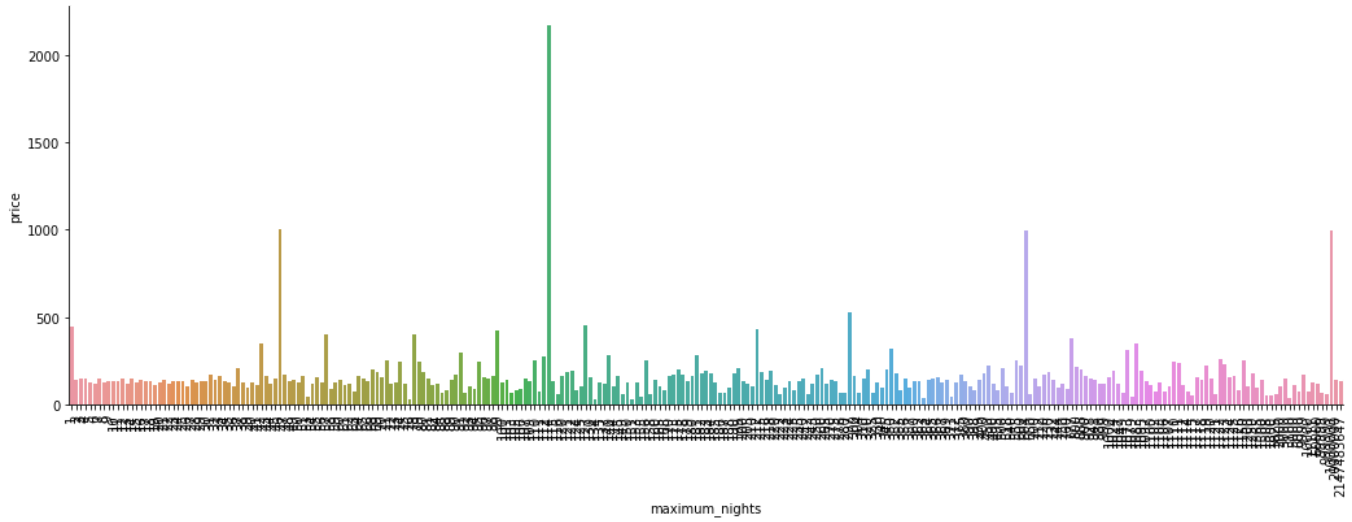


Minimum Nights

We see that two rental units with 99 or 114 minimum nights stay requirement had a price of >$4000. Assuming most renters were staying for a short period of time from 1 day to months, these two points could be the total cost of renting a certain property type.

In [11]:
```
plt.figure(figsize=(20,15))
ax=sns.catplot(x="maximum_nights", y="price", kind="bar", data=sub, ci=None,height=5
, aspect=3)
ax.set_xticklabels(rotation=90)
```

Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6e46c5eb8>

<Figure size 1440x1080 with 0 Axes>
```

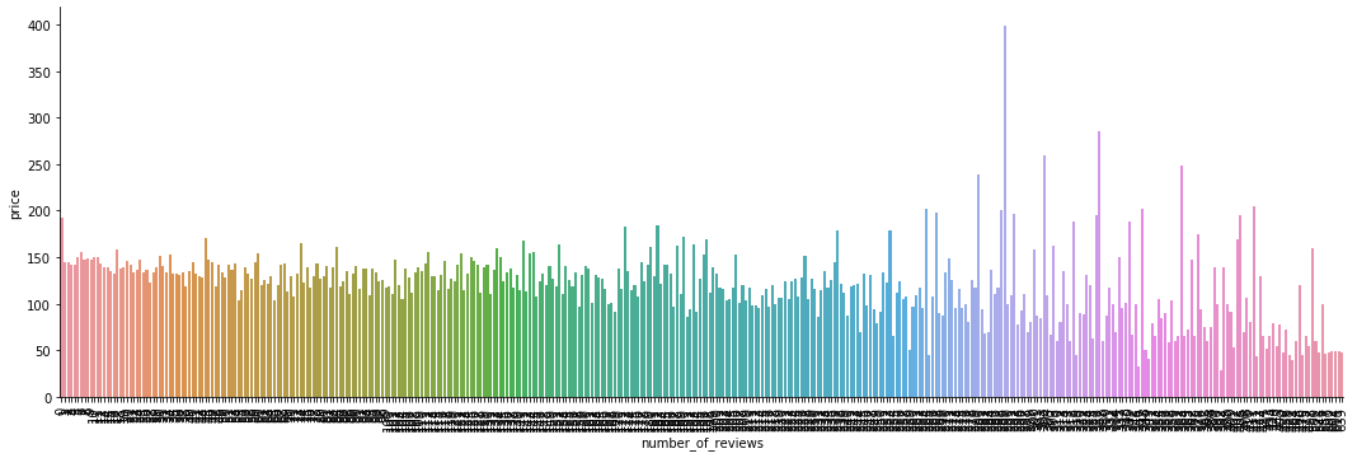


Maximum Nights

Simialr to the minimum nights stay, some of the maximum nights data points had $1000 to over 2000$ rental price that that could either be the total cost of renting the unit upto the maximum days or they were pricing for a long term stay.

In [12]:
```
plt.figure(figsize=(20,15))
ax=sns.catplot(x="number_of_reviews", y="price", kind="bar", data=sub , ci=None,heig
ht=5, aspect=3)
ax.set_xticklabels(rotation=90)
```

Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6e46c5f98>

<Figure size 1440x1080 with 0 Axes>
```

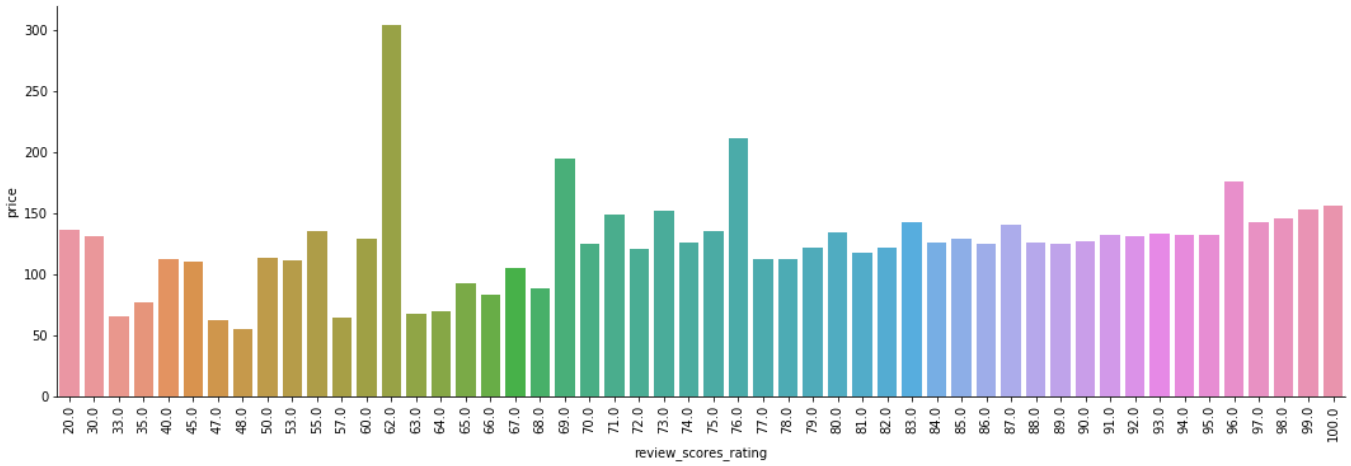


Number of Reviews vs Price

Number of reviews were uniformly distributed for rent price below $200 and there seem to be a high amount of reviews for a 400 price Airbnb unit. This may need to further investigate to find out which unit, host and location this property was located.

In [13]:
```
plt.figure(figsize=(20,15))
ax=sns.catplot(x="review_scores_rating", y="price", kind="bar", data=sub , ci=None,h
eight=5, aspect=3)
ax.set_xticklabels(rotation=90)
```

Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x1d6d8ab9f28>

<Figure size 1440x1080 with 0 Axes>
```



Review Rating Score vs Price

Rating Score between 80 to 100 were associated primarily with the rental units between >$100 to <200. There was a low score of 62 that was associated with a rental price of $300 and this needs to be examine to reference the property type and location to see if we could perhaps isolate the point.

What else do we need to know?

Guest Demand:
Guests ultimately choose their preferred listing due to a complex combination personal preferences that is challenging to understand. The current feature list doesn't contain a lot of information about why certain locations are preferrable over others. We could learn more about why demand is higher for certain properties if the following fields were collected:

- Closest airport: provide information on travel convenience
- Distance from closest airport: quantifies travel convenience
- Closest landmark: provides information on probable draw of guest to listing
- Distance to closest landmark: quantifies convenience to attraction

Customer satisfaction:
Most guests tend to rank hosts highly; usually only very poor experiences are reflected in host ratings. Therefore, there might be additional benefit to using text analytics for sentiment analysis on the guest review descriptions to understand more fully their actual experience, rather than the reviews that tend to be biased towards perfect rankings.

Additionally, it would be beneficial to understand the length of the host's preferred stay. If hosts only offer their listings for a certain period of time, it could cater to different kinds of guests with different kinds of preferences. A variable containing the length of stay could specify the market to which the listing is targeting.