In [78]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

In [2]:
```
#_____________Data load_________#
calendar_boston = pd.read_csv("calendar_boston.csv")
listings_boston = pd.read_csv("listings_boston.csv")
reviews_boston = pd.read_csv("reviews_boston.csv")
calendar_seattle = pd.read_csv("calendar_seattle.csv")
listings_seattle = pd.read_csv("listings_seattle.csv")
reviews_seattle = pd.read_csv("reviews_seattle.csv")
```

In [27]:
```
#_________Preview Datafreame_________
#calendar_boston.head()
#listings_boston.head()
reviews_boston.head()
#calendar_seatle.head()
#listings_seatle.head()
#reviews_seatle.head()
```

Out[27]:

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	1178162	4724140	2013-05-21	4298113	Olivier	My stay at islam's place was really cool! Good...
1	1178162	4869189	2013-05-29	6452964	Charlotte	Great location for both airport and city - gre...
2	1178162	5003196	2013-06-06	6449554	Sebastian	We really enjoyed our stay at Islams house. Fr...
3	1178162	5150351	2013-06-15	2215611	Marine	The room was nice and clean and so were the co...
4	1178162	5171140	2013-06-16	6848427	Andrew	Great location. Just 5 mins walk from the Airp...

In [3]:
```
#merge Boston listing and calendar df
linstings_calendar_boston = pd.merge(listings_boston, calendar_boston, left_on
='id', right_on='listing_id')
```

In [12]:
```
linstings_calendar_boston.head()
```

Out[12]:

	id	listing_url	scrape_id	last_scraped	name	sum
0	12147973	https://www.airbnb.com/rooms/12147973	20160906204935	2016-09-07	Sunny Bungalow in the City	(s f h M bed h
1	12147973	https://www.airbnb.com/rooms/12147973	20160906204935	2016-09-07	Sunny Bungalow in the City	(s f h M bed h
2	12147973	https://www.airbnb.com/rooms/12147973	20160906204935	2016-09-07	Sunny Bungalow in the City	(s f h M bed h
3	12147973	https://www.airbnb.com/rooms/12147973	20160906204935	2016-09-07	Sunny Bungalow in the City	(s f h M bed h
4	12147973	https://www.airbnb.com/rooms/12147973	20160906204935	2016-09-07	Sunny Bungalow in the City	(s f h M bed h

5 rows × 99 columns

In [4]:
```
# Deleting Duplicate column from boston df
del linstings_calendar_boston['listing_id']
```

In [5]:
```
#merge Seattle listing and calendar df
linstings_calendar_seattle = pd.merge(listings_seattle, calendar_seattle, left
_on ='id', right_on='listing_id')
```

In [6]:
```
# Deleting Duplicate column from boston df
del linstings_calendar_seattle['listing_id']
```

In [7]:
```
#merge Boston listing_calendar and reviews df
linstings_calendar_reviews_boston = pd.merge(listings_boston,reviews_boston, l
eft_on ='id', right_on='listing_id')
```

In [91]:
```
#review dataframe
linstings_calendar_reviews_boston.head()
```

Out[91]:

	id_x	listing_url	scrape_id	last_scraped	name	summa
0	3075044	https://www.airbnb.com/rooms/3075044	20160906204935	2016-09-07	Charming room in pet friendly apt	Charmi and qu room ir seco flo 191(
1	3075044	https://www.airbnb.com/rooms/3075044	20160906204935	2016-09-07	Charming room in pet friendly apt	Charmi and qu room ir seco flo 191(
2	3075044	https://www.airbnb.com/rooms/3075044	20160906204935	2016-09-07	Charming room in pet friendly apt	Charmi and qu room ir seco flo 191(
3	3075044	https://www.airbnb.com/rooms/3075044	20160906204935	2016-09-07	Charming room in pet friendly apt	Charmi and qu room ir seco flo 191(
4	3075044	https://www.airbnb.com/rooms/3075044	20160906204935	2016-09-07	Charming room in pet friendly apt	Charmi and qu room ir seco flo 191(

5 rows × 98 columns

In [8]:
```
#merge Seattle listing_calendar and reviews df
linstings_calendar_reviews_seattle = pd.merge(listings_seattle,reviews_seattle
, left_on ='id', right_on='listing_id')
```

In [17]:
```
#linstings_calendar_reviews_boston.describe()
linstings_calendar_reviews_seattle.describe()
```

Out[17]:

	id_x	scrape_id	host_id	host_listings_count	host_total_listings_count
count	8.484900e+04	8.484900e+04	8.484900e+04	84849.000000	84849.000000
mean	3.005067e+06	2.016010e+13	9.304441e+06	4.135417	4.135417
std	2.472877e+06	4.172681e+01	1.019022e+07	14.177846	14.177846
min	4.291000e+03	2.016010e+13	4.193000e+03	1.000000	1.000000
25%	7.946330e+05	2.016010e+13	1.393266e+06	1.000000	1.000000
50%	2.488228e+06	2.016010e+13	5.486995e+06	1.000000	1.000000
75%	4.694479e+06	2.016010e+13	1.395977e+07	3.000000	3.000000
max	1.024814e+07	2.016010e+13	5.076344e+07	502.000000	502.000000

8 rows × 33 columns

In [19]:
```
#Column comparison between two Dataframe for union
columns_BSTN= linstings_calendar_reviews_boston.columns
columns_STLE = linstings_calendar_reviews_seattle

aux = []

for column in columns_BSTN:
    if column not in columns_STLE:
        aux.append(column)

print("We don't have this information in Seatle data:",aux)

#We drop this variables.

linstings_calendar_reviews_boston.drop(aux, axis=1,inplace=True)
```

```
We don't have this information in Seatle data: ['access', 'interaction', 'hou
se_rules']
```

In [20]:
```
#Appended Seattle dataframe under Boston data
Boston_Seattle = linstings_calendar_reviews_boston.append(linstings_calendar_r
eviews_seattle, ignore_index=True)
```

In [21]:
```
Boston_Seattle.count()
```

Out[21]:
```
id_x              153124
listing_url       153124
scrape_id         153124
last_scraped      153124
name              153124
                   ...
id_y              153124
date              153124
reviewer_id       153124
reviewer_name     153124
comments          153053
Length: 98, dtype: int64
```

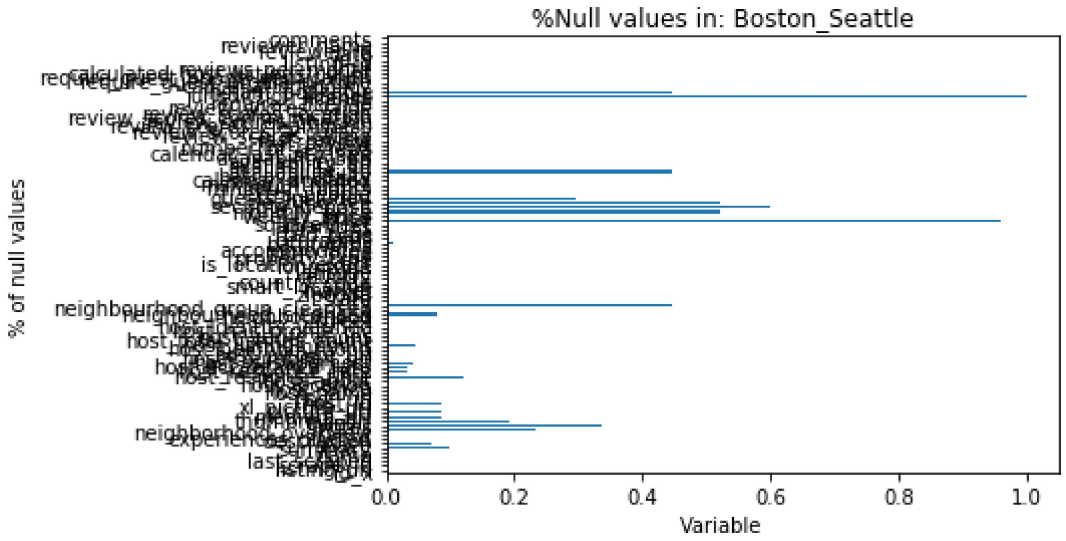
In [22]:
```
###Dimension of dataset

print("Dimension df for Boston_Seattle:", Boston_Seattle.shape)
```

```
Dimension df for Boston_Seattle: (153124, 98)
```

In [27]:
```
#Review null values
#Queries for many columns in dataset how to identify the columns with null val
ues
Boston_Seattle.isnull().sum()
```

Out[27]:
```
id_x               0
listing_url        0
scrape_id          0
last_scraped       0
name               0
                  ..
id_y               0
date               0
reviewer_id        0
reviewer_name      0
comments          71
Length: 98, dtype: int64
```

In [26]:
```
def plot_null_values(df, ss):
    '''
    Input: Dataframe
    Output: Plot for the percentage of null values in dataset
    '''
    df_aux = Boston_Seattle.isnull().sum()/df.shape[0]
    df_aux.plot(kind='barh')6
    plt.title("%Null values in: "+ ss)
    plt.ylabel("Variable")
    plt.xlabel("% of null values")
    #plt.savefig("/images/"+ss+'.png')
    plt.show()

Boston_Seattle_df = { "Boston_Seattle": Boston_Seattle }


for name, df in Boston_Seattle_df.items():
     plot_null_values(df, name)
```



In [50]:
```
#Identifying null values and updating with value No comments
#Boston_Seattle.isnull().sum()

Boston_Seattle["comments"].fillna("No Comments", inplace = True)
```

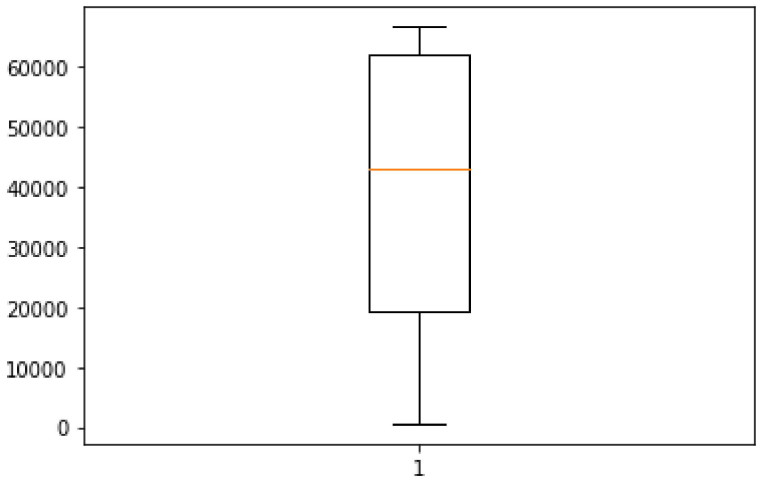
In [134]:
```
Boston_Seattle.describe()
```

Out[134]:

	id_x	scrape_id	host_id	host_listings_count	host_total_listings_count
count	1.531240e+05	1.531240e+05	1.531240e+05	153124.000000	153124.000000
mean	3.787517e+06	2.016046e+13	1.159702e+07	8.521727	8.521727
std	3.248161e+06	3.987460e+08	1.354244e+07	46.101018	46.101018
min	3.353000e+03	2.016010e+13	4.193000e+03	1.000000	1.000000
25%	1.071843e+06	2.016010e+13	1.697505e+06	1.000000	1.000000
50%	3.139972e+06	2.016010e+13	6.592975e+06	2.000000	2.000000
75%	5.958674e+06	2.016091e+13	1.764404e+07	4.000000	4.000000
max	1.484378e+07	2.016091e+13	9.287818e+07	749.000000	749.000000

8 rows × 39 columns

In [52]:
```
y = list(Boston_Seattle['cancellation_policy'].value_counts())
plt.boxplot(y)
plt.show()
```



In [54]:
```
Boston_Seattle.groupby(['name']).count()
```

Out[54]:

name	id_x	listing_url	scrape_id	last_scraped	summary	space	descri
BY Downtown*Universities*Hospitals	81	81	81	81	81	81	
Ballard Garden Apartment	25	25	25	25	25	25	
Capitol Hill Gem in Great Location	12	12	12	12	12	12	
Cherry Hill Cozy, Budget Loft-Bed	68	68	68	68	68	68	
Comfortable Green Lake Home Base	34	34	34	34	0	34	
...	...	...	...	...	...	...	
【Boston】中心近【Chinatown】	1	1	1	1	1	1	
交通便利 经济实惠的公寓单间。	2	2	2	2	2	0	
位于比肯山的1卧單位房源	62	62	62	62	0	62	
温馨双人房，一分钟到地铁。	158	158	158	158	158	0	
红线地铁JFK/UMASS 站边上的一个房间	14	14	14	14	14	14	

5949 rows × 97 columns

In [55]:
```
#cleaning price variable.

Boston_Seattle.price = Boston_Seattle.price.apply(lambda x: x.split('.')[0]).r
eplace('[^0-9]', '', regex=True).apply(lambda x: int(x))

Boston_Seattle.extra_people = Boston_Seattle.extra_people.apply(lambda x: x.sp
lit('.')[0]).replace('[^0-9]', '', regex=True).apply(lambda x: int(x))

Boston_Seattle.cleaning_fee = Boston_Seattle.cleaning_fee.fillna("$0")

Boston_Seattle.cleaning_fee = Boston_Seattle.cleaning_fee.apply(lambda x: x.sp
lit('.')[0]).replace('[^0-9]', '', regex=True).apply(lambda x: int(x))

Boston_Seattle.weekly_price   = Boston_Seattle.weekly_price.fillna("$0")

Boston_Seattle.weekly_price   = Boston_Seattle.weekly_price.apply(lambda x: x.
split('.')[0]).replace('[^0-9]', '', regex=True).apply(lambda x: int(x))

Boston_Seattle.monthly_price    = Boston_Seattle.monthly_price.fillna("$0")

Boston_Seattle.monthly_price  = Boston_Seattle.monthly_price.apply(lambda x: x
.split('.')[0]).replace('[^0-9]', '', regex=True).apply(lambda x: int(x))

Boston_Seattle.security_deposit    = Boston_Seattle.security_deposit.fillna(
"$0")

Boston_Seattle.security_deposit   = Boston_Seattle.security_deposit.apply(lamb
da x: x.split('.')[0]).replace('[^0-9]', '', regex=True).apply(lambda x: int(x
))


dates = ["calendar_last_scraped", "last_scraped", "host_since"]
for col_date in dates:

    Boston_Seattle[col_date] = pd.to_datetime(Boston_Seattle[col_date]) #good
 format for date variable

Boston_Seattle.reset_index(drop=True, inplace=True)
```

In [135]:
```
Boston_Seattle.describe().iloc[:,20:30]
```

Out[135]:

	maximum_nights	availability_30	availability_60	availability_90	availability_365	number_c
count	1.531240e+05	153124.000000	153124.000000	153124.000000	153124.000000	1531
mean	1.123016e+05	13.393178	31.724498	53.361165	249.596517	
std	3.339892e+06	10.776851	20.794700	29.355944	118.014479	
min	1.000000e+00	0.000000	0.000000	0.000000	0.000000	
25%	3.000000e+01	3.000000	12.000000	32.000000	153.000000	
50%	1.125000e+03	12.000000	33.000000	59.000000	305.000000	
75%	1.125000e+03	23.000000	51.000000	80.000000	346.000000	1
max	1.000000e+08	30.000000	60.000000	90.000000	365.000000	4

In [60]:
```
duplicate_df=Boston_Seattle[Boston_Seattle.duplicated()]
print("Number of duplicated rows : ",duplicate_df.shape)
```

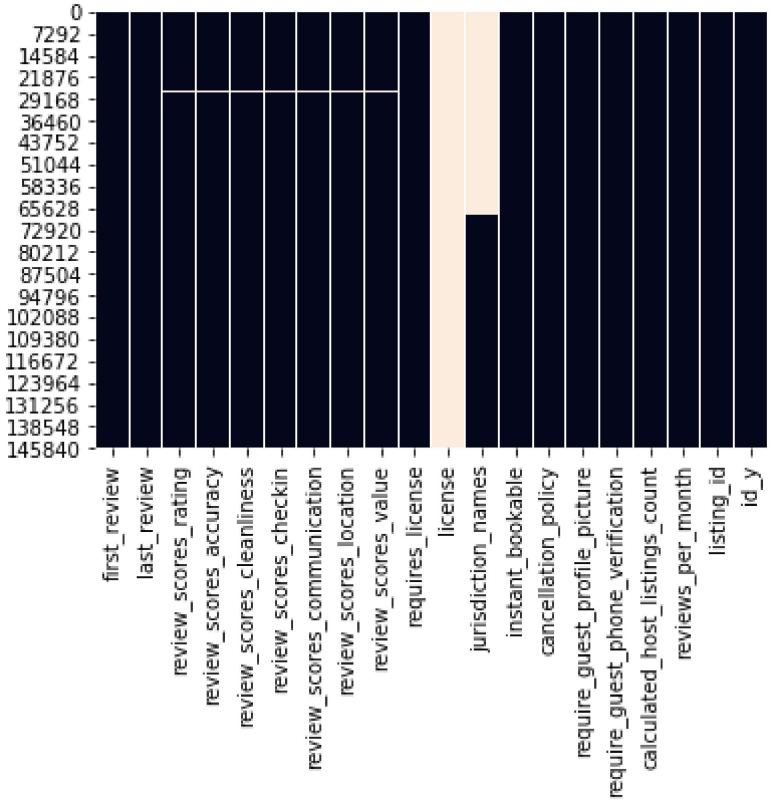
```
Number of duplicated rows :  (0, 98)
```

In [62]:
```
print(Boston_Seattle.isnull().sum())
```

```
id_x              0
listing_url       0
scrape_id         0
last_scraped      0
name              0
                 ..
id_y              0
date              0
reviewer_id       0
reviewer_name     0
comments          0
Length: 98, dtype: int64
```

In [133]:
```
sns.heatmap(Boston_Seattle.iloc[:, 0:25].isnull(), cbar=False )
sns.heatmap(Boston_Seattle.iloc[:, 25:50].isnull(), cbar=False)
sns.heatmap(Boston_Seattle.iloc[:, 50:75].isnull(), cbar=False )
sns.heatmap(Boston_Seattle.iloc[:, 75:95].isnull(), cbar=False)
```

Out[133]: <matplotlib.axes._subplots.AxesSubplot at 0x23c732c01c8>



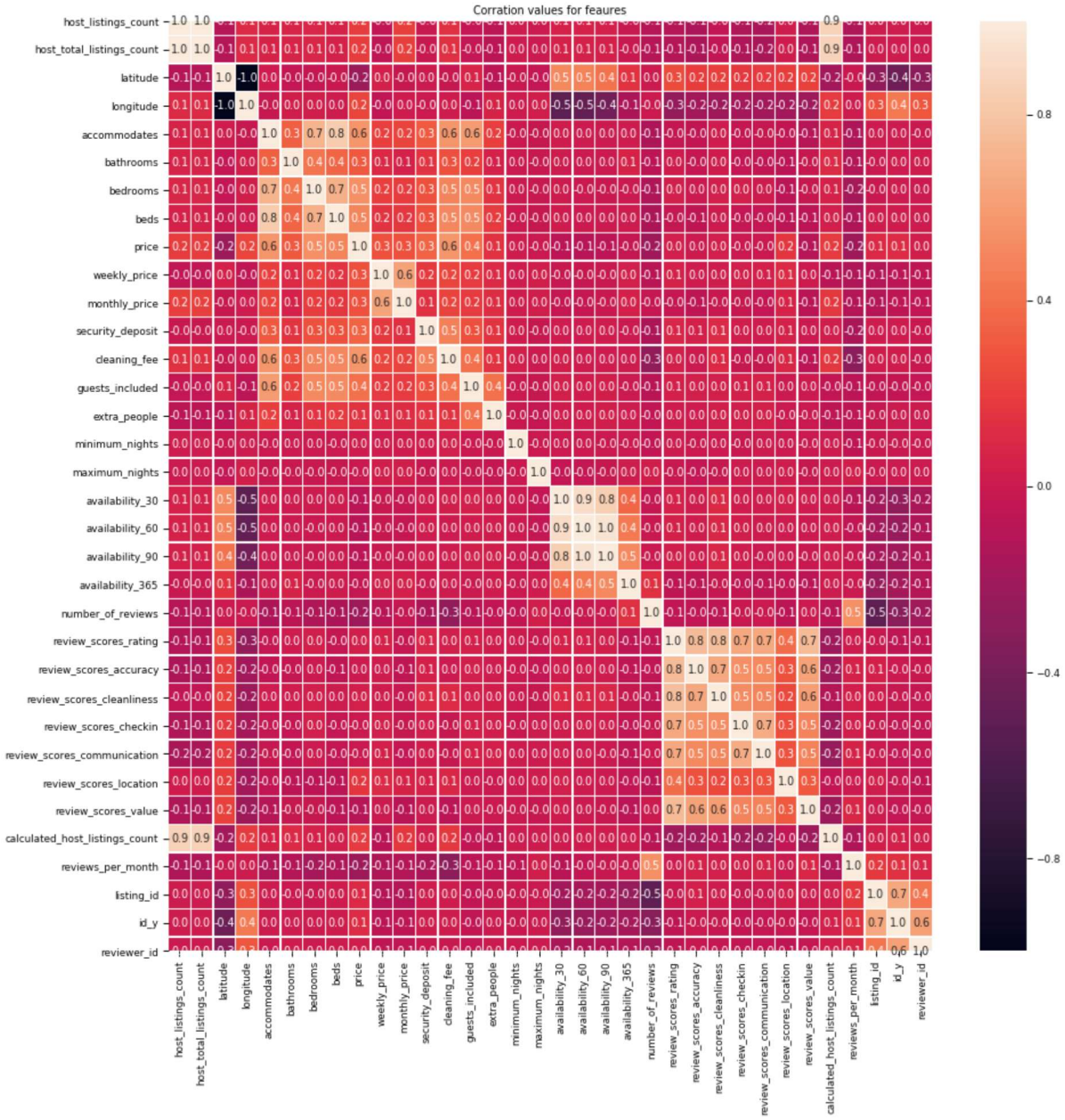
In [145]:
```
#delete unuseful features, with a high percentage of null values, with a lot o
f same values and features with high cardinality.
dic_null_values = {}
for col in Boston_Seattle.columns:
    pcte = Boston_Seattle[col].isnull().sum()/Boston_Seattle.shape[0]
    dic_null_values[col] = pcte

cols_drop= []

for col, pcte in dic_null_values.items():
    if pcte > 0.7:
        cols_drop.append(col)

Boston_Seattle.drop(cols_drop, axis=1, inplace=True)

# Detecting cardinality and features with a lot of same values

#We have that the following features have a big variance: id, scrape_id, hot_i
d. So we will drop it.

Boston_Seattle.drop(["id_x", "scrape_id", "host_id"],axis=1, inplace=True)


print("New dimension of our dataset:", Boston_Seattle.shape)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-145-6ded018a93b0> in <module>
      1 #delete unuseful features, with a high percentage of null values, wit
h a lot of same values and features with high cardinality.
----> 2 import plotly.graph_objs as go
      3 dic_null_values = {}
      4 for col in Boston_Seattle.columns:
      5     pcte = Boston_Seattle[col].isnull().sum()/Boston_Seattle.shape[0]

ModuleNotFoundError: No module named 'plotly'
```

In [146]:
```
import plotly.offline as py
import plotly.graph_objs as go
py.init_notebook_mode()

price = Boston_Seattle['price'].values.tolist()
boston_price = Boston_Seattle['price'].loc[Boston_Seattle['city'] == 'Boston']
.values.tolist()
seatle_price = Boston_Seattle['price'].loc[Boston_Seattle['city'] == 'Seatle']
.values.tolist()

trace0 = go.Histogram(
    x=boston_price,
    histnorm='probability',
    name="Boston Prices",
    marker = dict(
        color = 'rgba(100, 149, 237, 0.6)',
    )
)
trace1 = go.Histogram(
    x= seatle_price,
    histnorm='probability',
    name="Seatle prices",
    marker = dict(
        color = 'rgba(255, 182, 193, 0.6)',
    )
)
trace2 = go.Histogram(
    x=price,
    histnorm='probability',
    name="Prices distribution",
     marker = dict(
        color = 'rgba(169, 169, 169, 0.6)',
    )
)
fig = tools.make_subplots(rows=2, cols=2, specs=[[{}, {}], [{'colspan': 2}, No
ne]],
                          subplot_titles=('Boston','Seatle', 'Two cities'))

fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 2)
fig.append_trace(trace2, 2, 1)

fig['layout'].update(showlegend=True, title='Distribution of Prices of Airbnb
s', bargap=0.05)
iplot(fig, filename='custom-sized-subplot-with-subplot-titles')

print("Null values:", Boston_Seattle.price.isnull().sum())
```

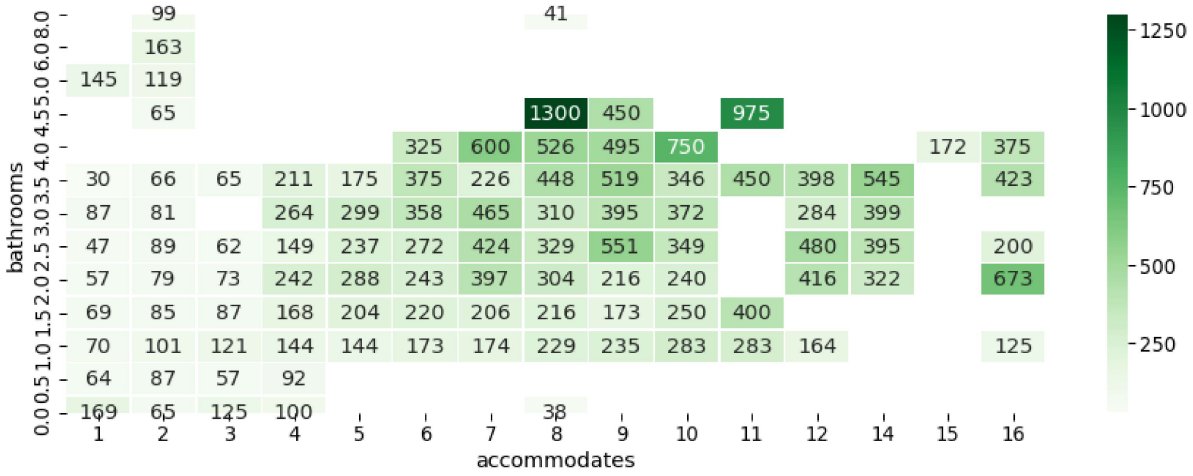
```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-146-070cb44e25d7> in <module>
----> 1 import plotly.offline as py
      2 import plotly.graph_objs as go
      3
      4 py.init_notebook_mode()
      5 price = Boston_Seattle['price'].values.tolist()

ModuleNotFoundError: No module named 'plotly'
```

In [153]:
```
#correlation values

f = plt.figure(figsize= (15,15))
sns.set_context("paper", font_scale=1.)
sns.heatmap(Boston_Seattle.corr(),  annot=True,  linewidths=.3, fmt= '.1f')
plt.title("Corration values for feaures");
```



In [156]:
```
#property type
f = plt.figure(figsize= (15,5))
sns.set_context("paper", font_scale=1.5)
sns.heatmap(Boston_Seattle.query('price <= 4000')\
                .groupby(['bathrooms', 'accommodates'])\
                .mean()['price']\
                .reset_index()\
                .pivot('bathrooms', 'accommodates', 'price')\
                .sort_index(ascending=False),
            cmap="Greens", fmt='.0f', annot=True, linewidths=0.5,)
```

Out[156]: <matplotlib.axes._subplots.AxesSubplot at 0x23c75fd4488>

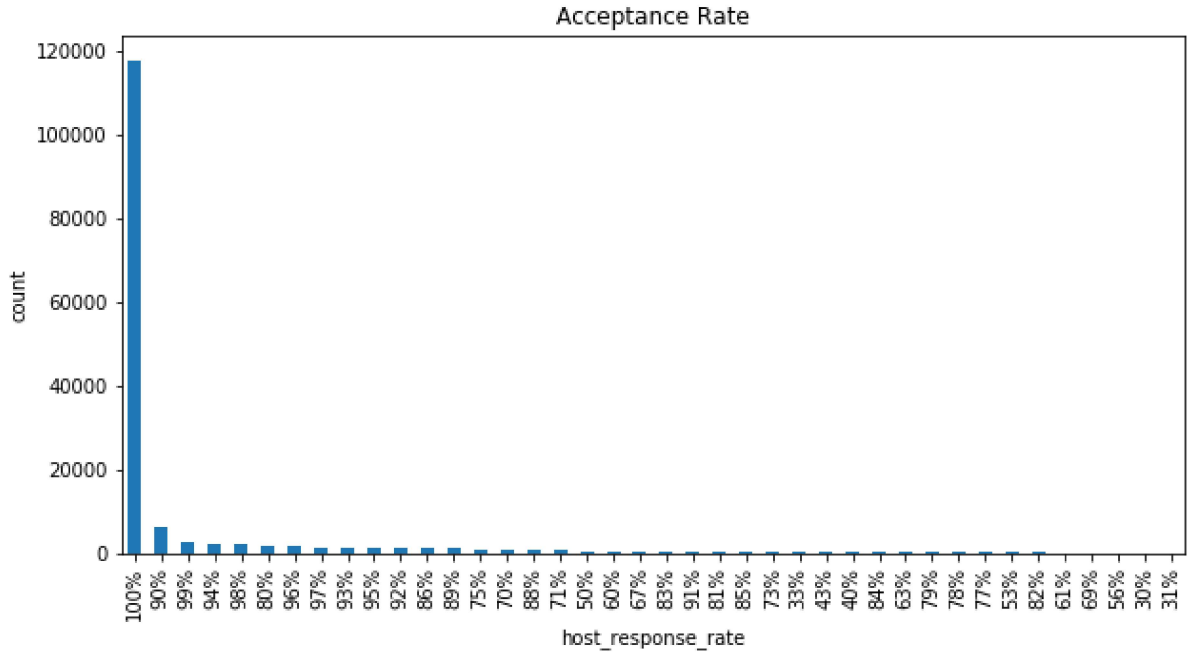


In [163]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-163-7d40d4766841> in <module>
      3                {'x': 0.39, 'y': 300, 'text': 'Top Performer Had Score
 above 90 Percentile','color': 'mediumaquamarine'}]
      4
----> 5 generate_barplot(title_text, annotations)

NameError: name 'generate_barplot' is not defined
```

In [99]:
```
Boston_Seattle.host_response_rate.value_counts().nlargest(40).plot(kind='bar',
figsize=(10,5))
plt.title("Acceptance Rate")
plt.ylabel("count")
plt.xlabel("host_response_rate")
```

Out[99]:
```
Text(0.5, 0, 'host_response_rate')
```



In [103]:
```
#Inserted new column to combine multiple score rating range columns into one
Boston_Seattle.insert(1,"rating_score_range","null")
```

In [130]:
```
#To filter records----------dataFrame = dataFrame[dataFrame['Car'].str.contai
ns('Lamborghini')]
#replace rating values from multiple columns
Boston_Seattle['score_ranges_10-20'].mask(Boston_Seattle['score_ranges_10-20']
== 1, '10-20', inplace=True)

#Boston_Seattle["score_ranges_10-20"] = np.where(Boston_Seattle["score_ranges_
10-20”] == 1, range_10_20, 0)
#Boston_Seattle.loc[Boston_Seattle["score_ranges_10-20"] == 1, "score_ranges_1
0-20"] = "10-20"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_20-30"] == 1, "score_ranges_2
0-30"] = "20-30"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_30-40"] == 1, "score_ranges_3
0-40"] = "30-40"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_40-50"] == 1, "score_ranges_4
0-50"] = "40-50"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_50-60"] == 1, "score_ranges_5
0-60"] = "50-60"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_60-70"] == 1, "score_ranges_6
0-70"] = "60-70"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_70-80"] == 1, "score_ranges_7
0-80"] = "70-80"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_80-90"] == 1, "score_ranges_8
0-90"] = "80-90"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_90-100"] == 1, "score_ranges_
90-100"] = "90-100"
#Boston_Seattle.loc[Boston_Seattle["score_ranges_100+"] == 1, "score_ranges_10
0+"] = "100+"

#score_range=Boston_Seattle.["score_ranges_10-20"]+Boston_Seattle.["score_rang
es_20-30"]+Boston_Seattle.["score_ranges_30-40"]+Boston_Seattle.["score_ranges
_40-50"]+Boston_Seattle.["score_ranges_40-50"]+Boston_Seattle.["score_ranges_5
0-60"]+Boston_Seattle.["score_ranges_60-70"]+Boston_Seattle.["score_ranges_70-
80"]+Boston_Seattle.["score_ranges_80-90"]+Boston_Seattle.["score_ranges_90-10
0"]+Boston_Seattle.["score_ranges_100+"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get
_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

KeyError: 'score_ranges_10-20'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-130-4cba6c9d38d0> in <module>
      1 #To filter records-----------dataFrame = dataFrame[dataFrame['Car'].s
tr.contains('Lamborghini')]
      2 #replace rating values from multiple columns
----> 3 Boston_Seattle['score_ranges_10-20'].mask(Boston_Seattle['score_range
s_10-20'] == 1, '10-20', inplace=True)
      4 #Boston_Seattle["score_ranges_10-20"] = np.where(Boston_Seattle["scor
e_ranges_10-20”] == 1, range_10_20, 0)
      5

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem_
_(self, key)
   2978             if self.columns.nlevels > 1:
   2979                 return self._getitem_multilevel(key)
-> 2980             indexer = self.columns.get_loc(key)
   2981             if is_integer(indexer):
   2982                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get
_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(
key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=to
lerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()
```

```
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

KeyError: 'score_ranges_10-20'
```