Como maximizar os lucros no Airbnb?

Nome: Diego Wenceslau

Telefone: 31 988712037 - Whatsapp

E-mail: diegowenceslau16@hotmail.com

Objetivo do projeto:

Airbnb é uma comunidade online, onde as pessoas possam anunciar e reservarem acomodações e meios de hospedagem.

A base de dados airbnb.csv (dados abertos da plataforma airbnb) contém dados de casas e apartamentos que estão disponíveis para aluguel e as características que influenciam na composição do preço da locação.

Abaixo, segue o modelo criado, realizando, preparação, análise exploratória e modelagem dos dados, para que possamos chegar no objetivo de entender as cacterísticas dos imóveis e precificar(definir valor) do aluguel de imóveis.

In [161]:
```
# Importando as bibliotecas

import os
path = os.getcwd()

import numpy as np # Utilizado para operações matemáticas
import pandas as pd # Utilizado para visualização, análise e manipulação dos d
ados
import matplotlib as mpl # Utilizado para criação de gráficos
import matplotlib.pyplot as plt # Utilizado para criação de gráficos
import seaborn as sns # Utilizado para criação de gráficos
import statsmodels.api as sm # Mostrar resultado do modelo de previsão

%matplotlib inline

# Carregando o algoritmo para utilizar o modelo de regressão e realização de m
étricas

from sklearn import linear_model
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.metrics import mean_absolute_error

# Utilizado para não mostrar avisos na execução do código

import warnings
warnings.filterwarnings("ignore")
```

Para carregar o conjunto de dados do dataset, foi utilizado o meu diretório de trabalho.

In [3]:
```
# Carregando o dataset do Airbnb

airbnb = pd.read_csv("airbnb.csv", sep = ',', encoding = "utf8")
```

In [4]:
```
# Visualizando as primeiras linhas do dataset

airbnb.head()
```

Out[4]:

	host_is_superhost	cancellation_policy	instant_bookable	host_total_listings_count	neigl
0	t	moderate	t	1.0	
1	f	strict_14_with_grace_period	f	2.0	
2	f	strict_14_with_grace_period	f	10.0	
3	f	strict_14_with_grace_period	f	10.0	
4	f	strict_14_with_grace_period	f	2.0	

5 rows × 34 columns

In [5]:
```
# Visualizando o tamanho do dataset

print('\nLinhas e Colunas: ', airbnb.shape)
```

```
Linhas e Colunas:  (7146, 34)
```

In [6]:
```
# Verificando o tipo dos dados do dataset

airbnb.dtypes
```

Out[6]:
```
host_is_superhost                object
cancellation_policy              object
instant_bookable                 object
host_total_listings_count       float64
neighbourhood_cleansed           object
latitude                        float64
longitude                       float64
property_type                    object
room_type                        object
accommodates                    float64
bathrooms                       float64
bedrooms                        float64
beds                            float64
bed_type                         object
minimum_nights                  float64
number_of_reviews               float64
review_scores_rating            float64
review_scores_accuracy          float64
review_scores_cleanliness       float64
review_scores_checkin           float64
review_scores_communication     float64
review_scores_location          float64
review_scores_value             float64
price                           float64
bedrooms_na                     float64
bathrooms_na                    float64
beds_na                         float64
review_scores_rating_na         float64
review_scores_accuracy_na       float64
review_scores_cleanliness_na    float64
review_scores_checkin_na        float64
review_scores_communication_na  float64
review_scores_location_na       float64
review_scores_value_na          float64
dtype: object
```

Logo abaixo, conseguimos realizar um resumo/sumário estatístico de algumas colunas do dataset.

Podemos perceber que termos propriedades sem banheiros(bathrooms), sem quartos(bedrooms) e sem cama(beds).

Termos acomodações que vão de 1 dia, até 365 dias.

Média de acomodações são de 3 pessoas.

Termos propriedades sem comentários, com score de 20 até 100.

In [6]:
```
# Sumário estatístico do dataset

airbnb.describe()
```

Out[6]:

	host_total_listings_count	latitude	longitude	accommodates	bathrooms	bedro
count	7146.000000	7146.000000	7146.000000	7146.000000	7146.000000	7146.000
mean	52.604954	37.765812	-122.430534	3.201092	1.328086	1.342
std	177.428653	0.022531	0.026799	1.914916	0.793787	0.932
min	0.000000	37.707430	-122.513060	1.000000	0.000000	0.000
25%	1.000000	37.751140	-122.442972	2.000000	1.000000	1.000
50%	2.000000	37.767570	-122.425490	2.000000	1.000000	1.000
75%	8.000000	37.784618	-122.411070	4.000000	1.500000	2.000
max	1199.000000	37.810310	-122.369790	16.000000	14.000000	14.000

8 rows × 27 columns

Aqui, conseguimos ver resumo/sumário estatístico de algumas variáveis de forma individual.

Podemos ver a coluna Preço(Price), que termos valores de 10 a 10.000, com média de 213.

In [7]:
```
# Sumário estatístico de algumas variáveis de forma individual

print('\nBanheiros:\n\n', airbnb.bathrooms.describe())
print('\n')
print('\nQuartos de dormir:\n\n', airbnb.bedrooms.describe())
print('\n')
print('\nPreço:\n\n', airbnb.price.describe())
```

```
Banheiros:

 count    7146.000000
mean        1.328086
std         0.793787
min         0.000000
25%         1.000000
50%         1.000000
75%         1.500000
max        14.000000
Name: bathrooms, dtype: float64



Quartos de dormir:

 count    7146.000000
mean        1.342709
std         0.932855
min         0.000000
25%         1.000000
50%         1.000000
75%         2.000000
max        14.000000
Name: bedrooms, dtype: float64



Preço:

 count     7146.000000
mean       213.309824
std        311.375499
min         10.000000
25%        100.000000
50%        150.000000
75%        235.000000
max      10000.000000
Name: price, dtype: float64
```

Agora vamos algumas analises exploradas realizando comparação das colunas do dataset.

In [9]:
```
# Quantidade por tipo de propriedade

qtd_tipo_propriedade = pd.DataFrame(airbnb['property_type'].value_counts())
qtd_tipo_propriedade.columns = ['Quantidade_tipo_propriedade']
qtd_tipo_propriedade
```

Out[9]:

	Quantidade_tipo_propriedade
Apartment	3010
House	1990
Condominium	760
Guest suite	496
Boutique hotel	183
Townhouse	140
Serviced apartment	116
Hotel	100
Loft	93
Hostel	87
Guesthouse	44
Bed and breakfast	29
Other	22
Aparthotel	20
Bungalow	17
Villa	10
Cottage	8
Resort	8
Tiny house	3
Cabin	3
Boat	2
Earth house	1
In-law	1
Timeshare	1
Castle	1
Treehouse	1

In [9]:
```
# Quantidade total de quartos por número de quartos

qtd_quartos= pd.DataFrame(airbnb['bedrooms'].value_counts())
qtd_quartos.columns = ['Quantidade_de_quartos']
qtd_quartos
```

Out[9]:

	Quantidade_de_quartos
1.0	4199
2.0	1304
0.0	804
3.0	627
4.0	175
5.0	25
6.0	9
7.0	2
14.0	1

In [10]:
```
# Valor total do preço por quantidade de quarto

qtd_quartos = airbnb.filter(items = ['bedrooms', 'price']).groupby('bedrooms')
.sum().sort_values(['price'], ascending=[False])
qtd_quartos
```

Out[10]:

	price
bedrooms	
1.0	625402.0
2.0	369160.0
3.0	270527.0
0.0	120083.0
4.0	98455.0
5.0	21244.0
6.0	16477.0
7.0	2895.0
14.0	69.0

In [11]:
```
# Quantidade total de tipo de cama

qtd_cama= pd.DataFrame(airbnb['bed_type'].value_counts())
qtd_cama.columns = ['Quantidade_de_cama']
qtd_cama
```

Out[11]:

	Quantidade_de_cama
Real Bed	7073
Futon	32
Pull-out Sofa	23
Airbed	11
Couch	7

In [12]:
```
# Valor total do preço por tipo de cama

total_cama = airbnb.filter(items = ['bed_type', 'price']).groupby('bed_type').
sum().sort_values(['price'], ascending=[False])
total_cama
```

Out[12]:

	price
bed_type	
Real Bed	1513777.0
Futon	4259.0
Pull-out Sofa	3273.0
Couch	1954.0
Airbed	1049.0

Vamos realizar o tratamento dos dados, verificando se tem linhas faltantes e valores NA(Nulos), no dataset.

Verificamos que todas as colunas contém todas as linhas e não termos valores Nulos.

In [15]:
```
# Verificando se tem linhas vazias nas colunas

airbnb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7146 entries, 0 to 7145
Data columns (total 34 columns):
host_is_superhost                 7146 non-null object
cancellation_policy               7146 non-null object
instant_bookable                  7146 non-null object
host_total_listings_count         7146 non-null float64
neighbourhood_cleansed            7146 non-null object
latitude                          7146 non-null float64
longitude                         7146 non-null float64
property_type                     7146 non-null object
room_type                         7146 non-null object
accommodates                      7146 non-null float64
bathrooms                         7146 non-null float64
bedrooms                          7146 non-null float64
beds                              7146 non-null float64
bed_type                          7146 non-null object
minimum_nights                    7146 non-null float64
number_of_reviews                 7146 non-null float64
review_scores_rating              7146 non-null float64
review_scores_accuracy            7146 non-null float64
review_scores_cleanliness         7146 non-null float64
review_scores_checkin             7146 non-null float64
review_scores_communication       7146 non-null float64
review_scores_location            7146 non-null float64
review_scores_value               7146 non-null float64
price                             7146 non-null float64
bedrooms_na                       7146 non-null float64
bathrooms_na                      7146 non-null float64
beds_na                           7146 non-null float64
review_scores_rating_na           7146 non-null float64
review_scores_accuracy_na         7146 non-null float64
review_scores_cleanliness_na      7146 non-null float64
review_scores_checkin_na          7146 non-null float64
review_scores_communication_na    7146 non-null float64
review_scores_location_na         7146 non-null float64
review_scores_value_na            7146 non-null float64
dtypes: float64(27), object(7)
memory usage: 1.9+ MB
```

In [16]:
```
# Confirmando se tem valores NA(nulos)

airbnb.isna().sum()
```

Out[16]:
```
host_is_superhost                0
cancellation_policy              0
instant_bookable                 0
host_total_listings_count        0
neighbourhood_cleansed           0
latitude                         0
longitude                        0
property_type                    0
room_type                        0
accommodates                     0
bathrooms                        0
bedrooms                         0
beds                             0
bed_type                         0
minimum_nights                   0
number_of_reviews                0
review_scores_rating             0
review_scores_accuracy           0
review_scores_cleanliness        0
review_scores_checkin            0
review_scores_communication      0
review_scores_location           0
review_scores_value              0
price                            0
bedrooms_na                      0
bathrooms_na                     0
beds_na                          0
review_scores_rating_na          0
review_scores_accuracy_na        0
review_scores_cleanliness_na     0
review_scores_checkin_na         0
review_scores_communication_na   0
review_scores_location_na        0
review_scores_value_na           0
dtype: int64
```

Agora vamos criar um novo dataset contendo apenas as colunas númericas.

In [12]:
```
# Salvando as colunas númericas em um uma nova variável e verificando o seu no
vo tamanho

airbnb_col_numericas = airbnb.select_dtypes(include='number')
airbnb_col_numericas.shape
```

Out[12]:
```
(7146, 27)
```

In [13]:
```
# Visualizando a variável que contém apenas as colunas númericas

airbnb_col_numericas.head()
```

Out[13]:

	host_total_listings_count	latitude	longitude	accommodates	bathrooms	bedrooms	beds
0	1.0	37.76931	-122.43386	3.0	1.0	1.0	2.0
1	2.0	37.74511	-122.42102	5.0	1.0	2.0	3.0
2	10.0	37.76669	-122.45250	2.0	4.0	1.0	1.0
3	10.0	37.76487	-122.45183	2.0	4.0	1.0	1.0
4	2.0	37.77525	-122.43637	5.0	1.5	2.0	2.0

5 rows × 27 columns

Chegou a hora de utilizar os gráficos para análise exploratoria.
****

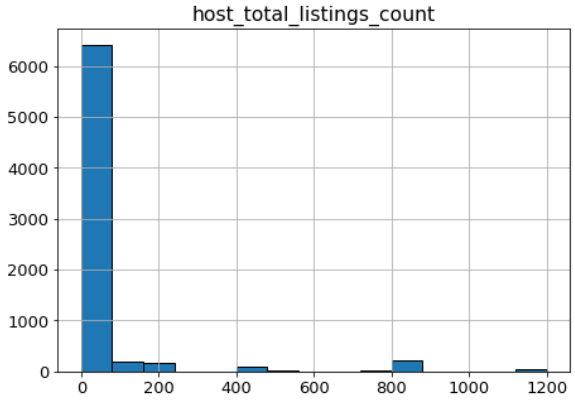
Em um dos gráficos abaixo, podemos ver:

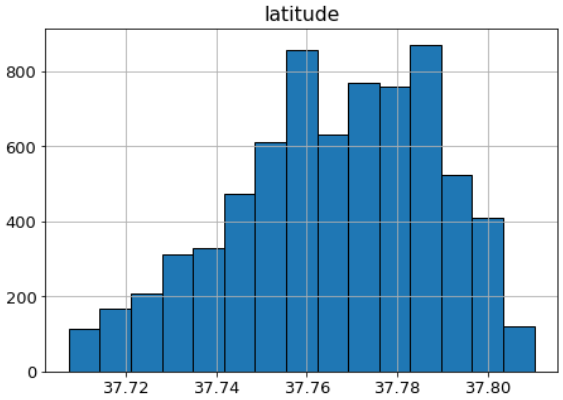
Quantidade maior de até dois banheiros.

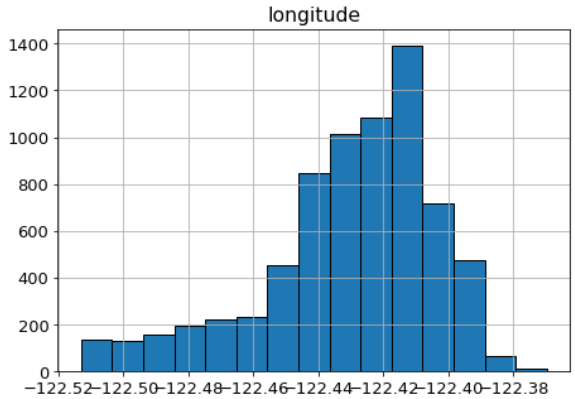
Termos uma quantidade maior de dois a quatro de quartos para dormi.

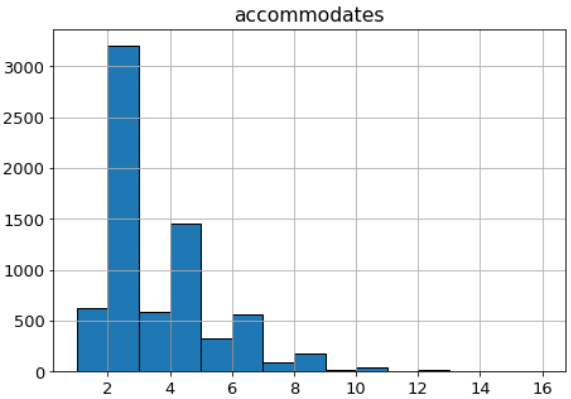
Quantidade maior de até duas camas.

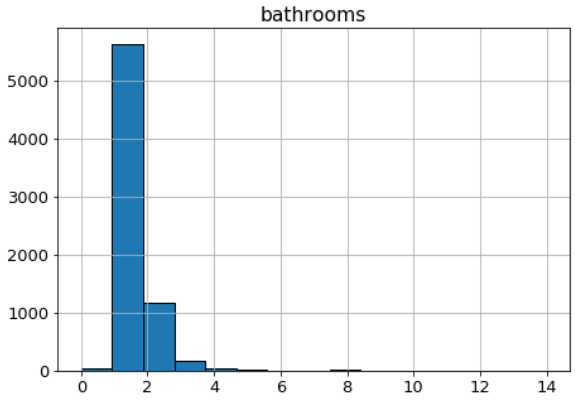
In [19]:
```
# Vamos criar gráficos para todas as variáveis númericas

plt.rcParams.update({'font.size': 13})

columns = airbnb_col_numericas.columns[:]
plt.subplots(figsize=(16,200))
length = len(columns)

for i,j in zip(columns,range(length)):
    plt.subplot((length/1),2,j+1)
    plt.subplots_adjust(wspace=0.2,hspace=0.5)
    airbnb[i].hist(bins=15,edgecolor='black')
    plt.title(i)

plt.show()
```


host_total_listings_count
0
1000
2000
3000
4000
5000
6000
0
200
400
600
800
1000
1200

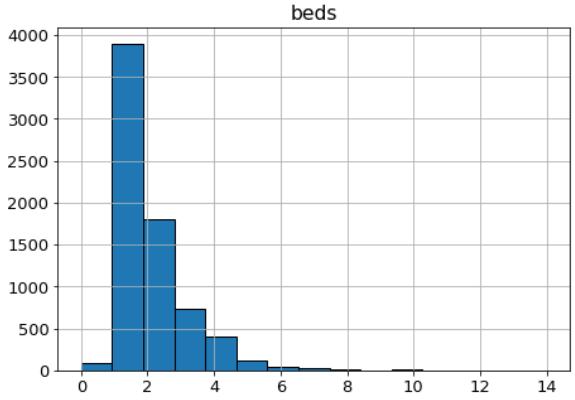

latitude
0
200
400
600
800
37.72
37.74
37.76
37.78
37.80

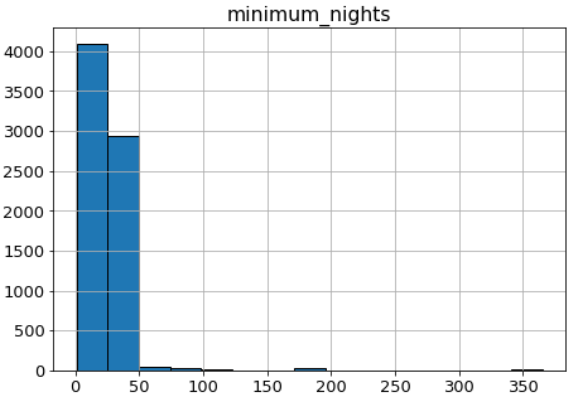

longitude
0
200
400
600
800
1000
1200
1400
−122.52
−122.50
−122.48
−122.46
−122.44
−122.42
−122.40
−122.38

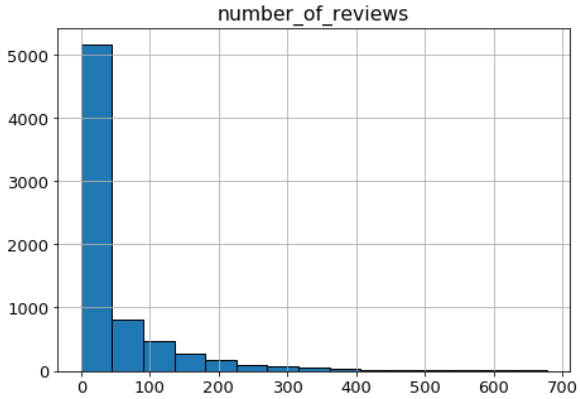

accommodates
0
500
1000
1500
2000
2500
3000
2
4
6
8
10
12
14
16

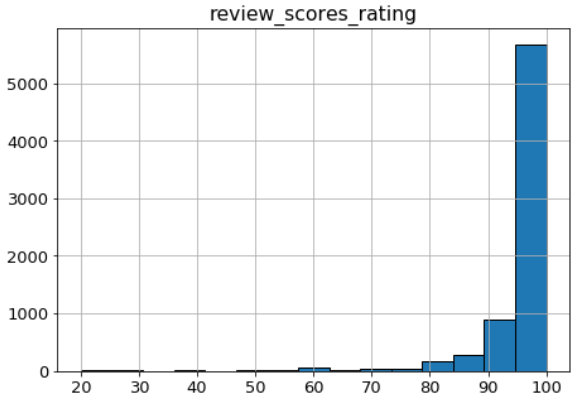

bathrooms
0
1000
2000
3000
4000
5000
0
2
4
6
8
10
12
14

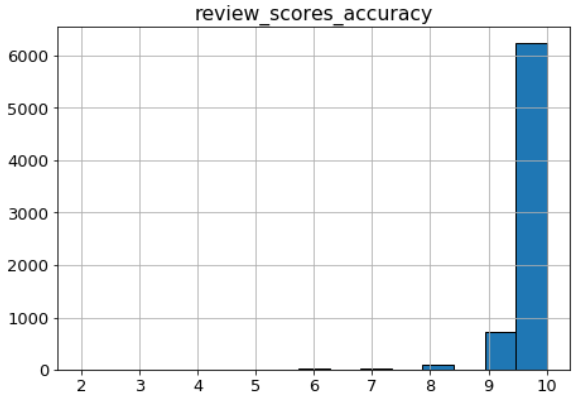

bedrooms
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
8
10
12
14

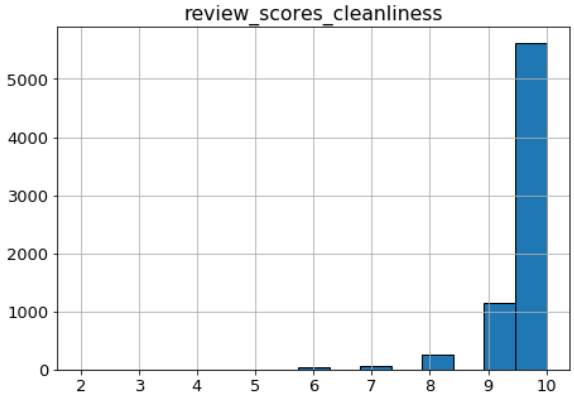

beds
0
500
1000
1500
2000
2500
3000
3500
4000
0
2
4
6
8
10
12
14

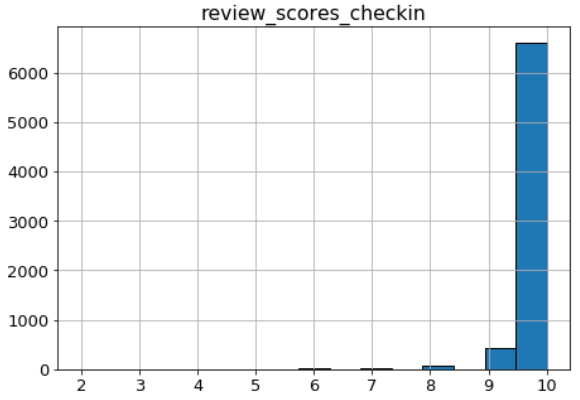

minimum_nights
0
500
1000
1500
2000
2500
3000
3500
4000
0
50
100
150
200
250
300
350

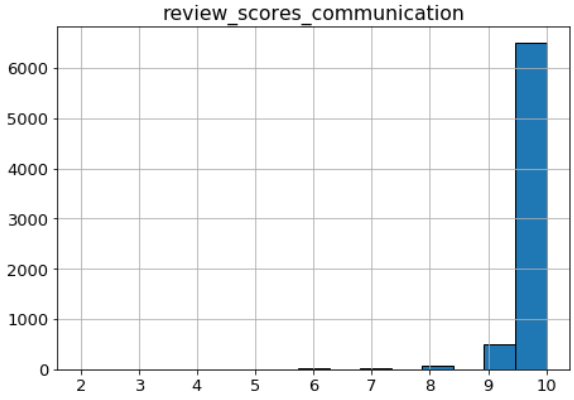

number_of_reviews
0
1000
2000
3000
4000
5000
0
100
200
300
400
500
600
700

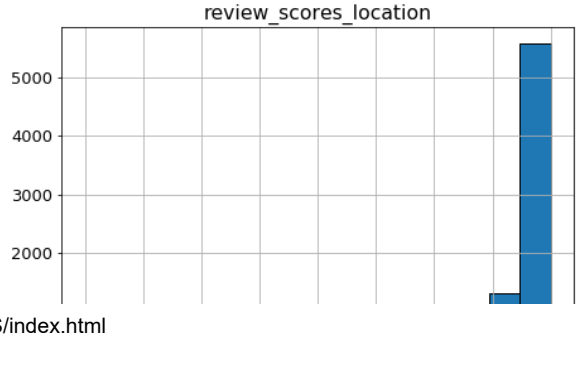

review_scores_rating
0
1000
2000
3000
4000
5000
20
30
40
50
60
70
80
90
100

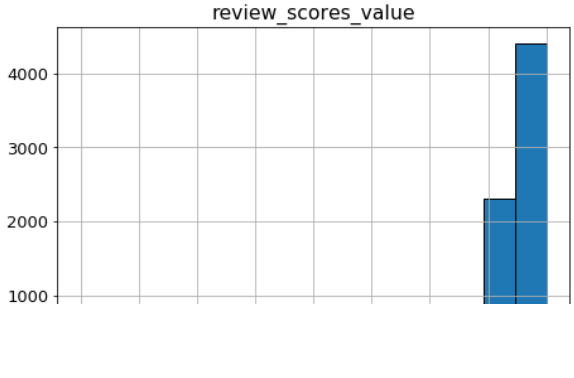

review_scores_accuracy
0
1000
2000
3000
4000
5000
6000
2
3
4
5
6
7
8
9
10

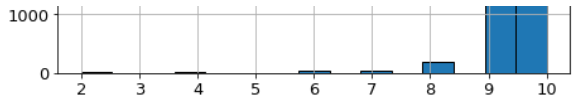

review_scores_cleanliness
0
1000
2000
3000
4000
5000
2
3
4
5
6
7
8
9
10

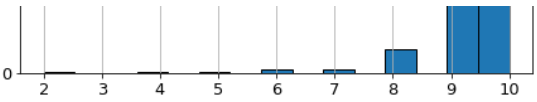

review_scores_checkin
0
1000
2000
3000
4000
5000
6000
2
3
4
5
6
7
8
9
10

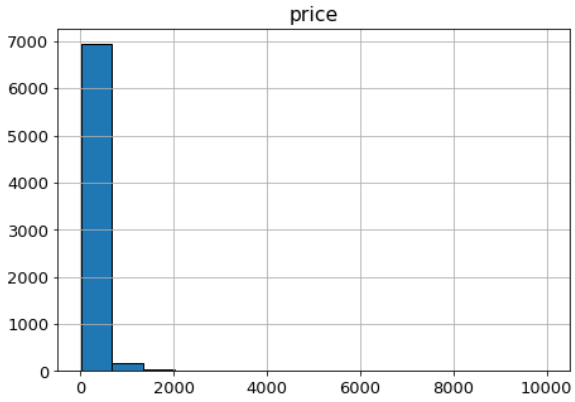

review_scores_communication
0
1000
2000
3000
4000
5000
6000
2
3
4
5
6
7
8
9
10

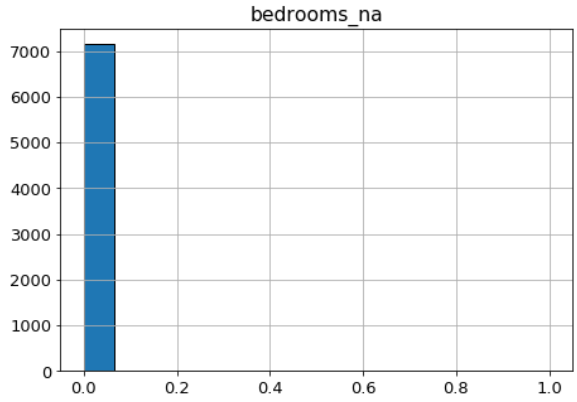

review_scores_location
2000
3000
4000
5000

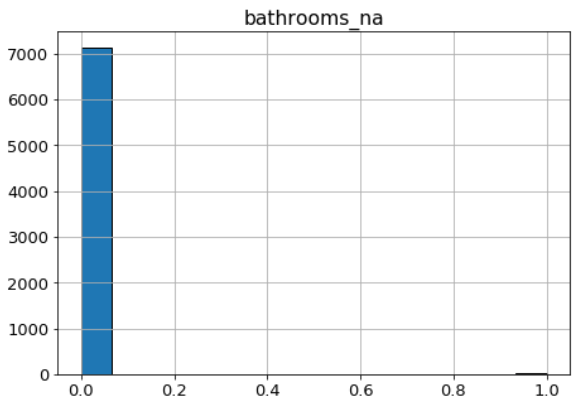

review_scores_value
1000
2000
3000
4000

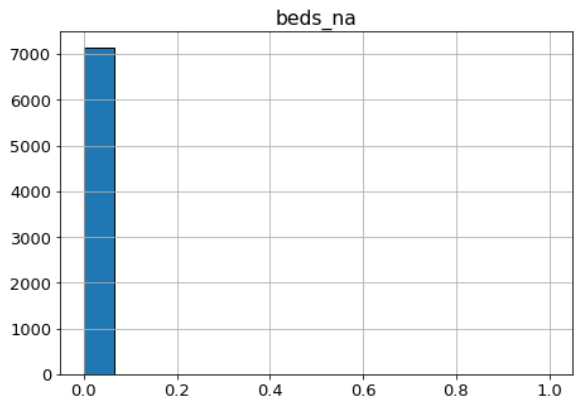

1000
0
2 3 4 5 6 7 8 9 10

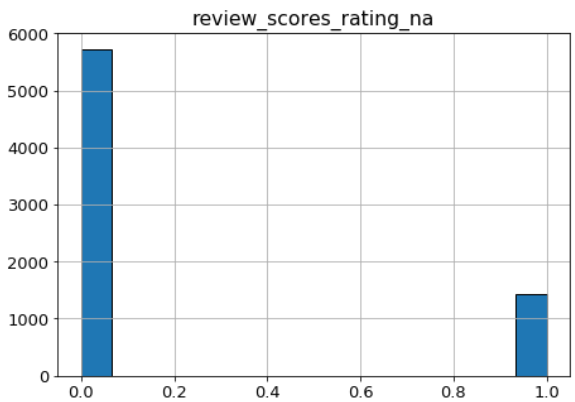

0
2 3 4 5 6 7 8 9 10

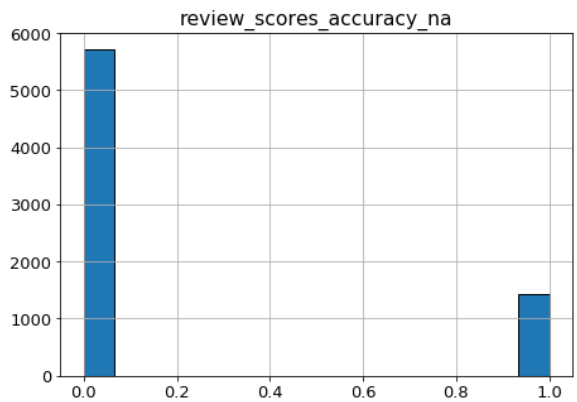

price
7000
6000
5000
4000
3000
2000
1000
0
0 2000 4000 6000 8000 10000

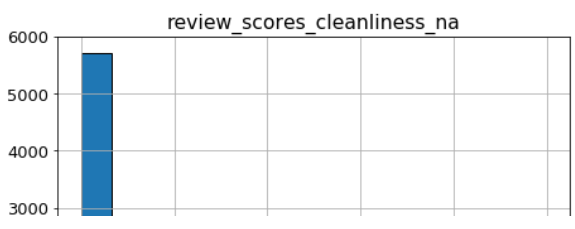

bedrooms_na
7000
6000
5000
4000
3000
2000
1000
0
0.0 0.2 0.4 0.6 0.8 1.0

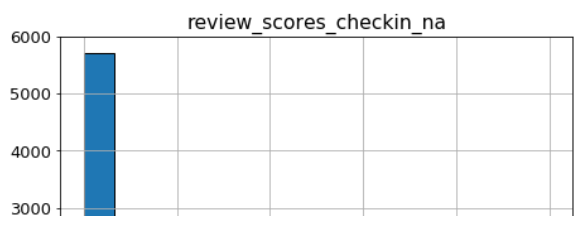

bathrooms_na
7000
6000
5000
4000
3000
2000
1000
0
0.0 0.2 0.4 0.6 0.8 1.0

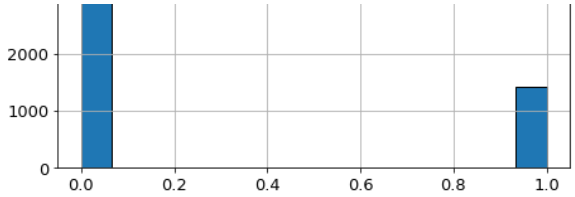

beds_na
7000
6000
5000
4000
3000
2000
1000
0
0.0 0.2 0.4 0.6 0.8 1.0

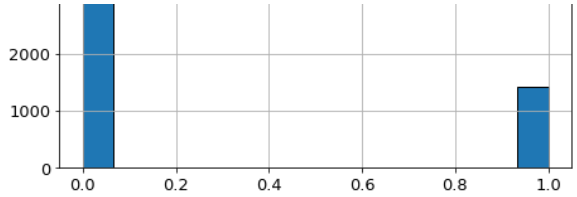

review_scores_rating_na
6000
5000
4000
3000
2000
1000
0
0.0 0.2 0.4 0.6 0.8 1.0

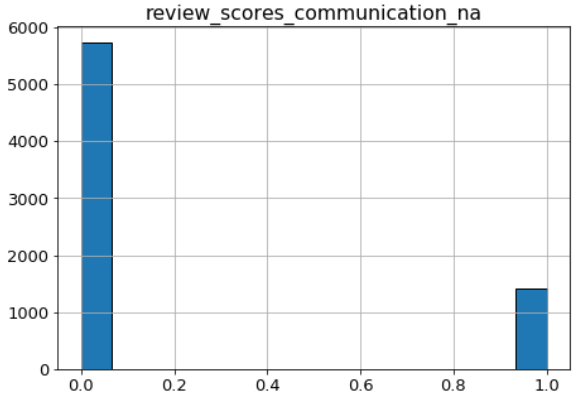

review_scores_accuracy_na
6000
5000
4000
3000
2000
1000
0
0.0 0.2 0.4 0.6 0.8 1.0

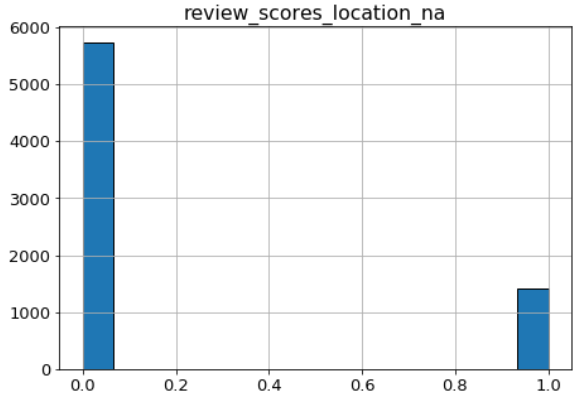

review_scores_cleanliness_na
6000
5000
4000
3000

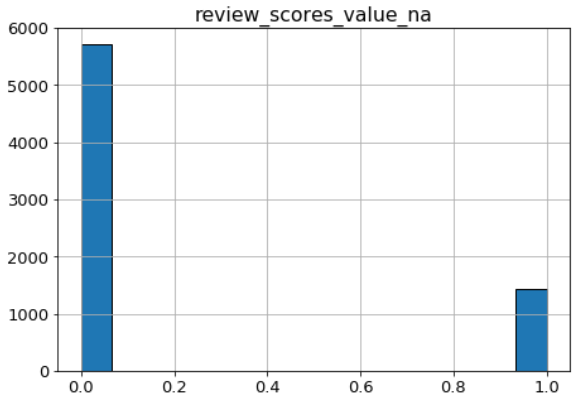

review_scores_checkin_na
6000
5000
4000
3000





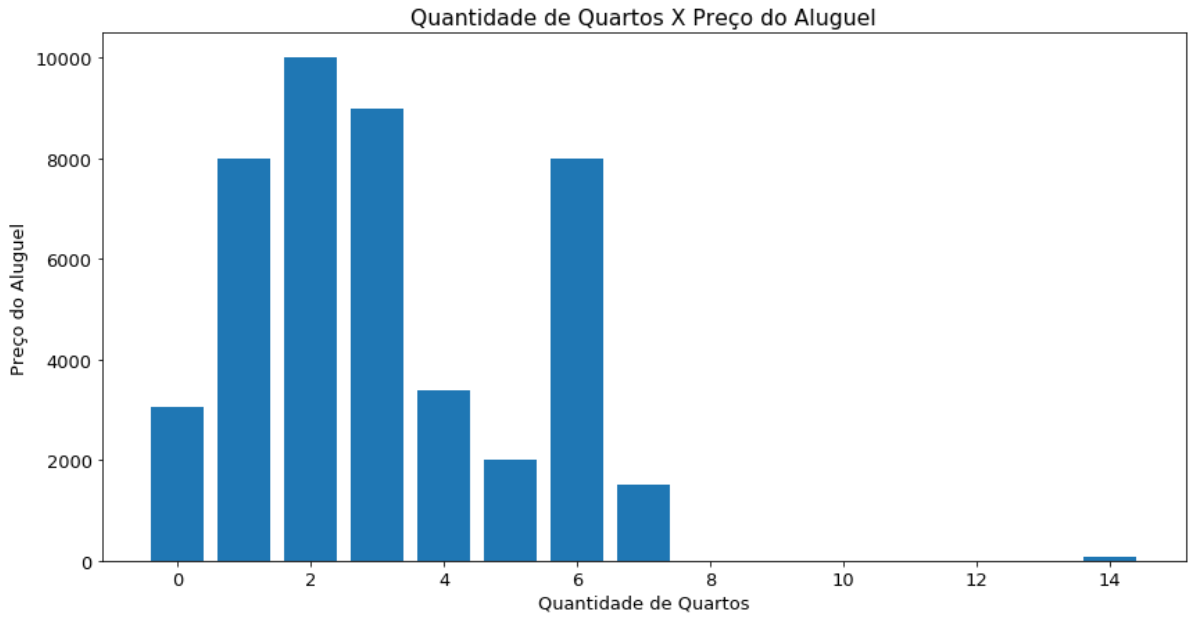






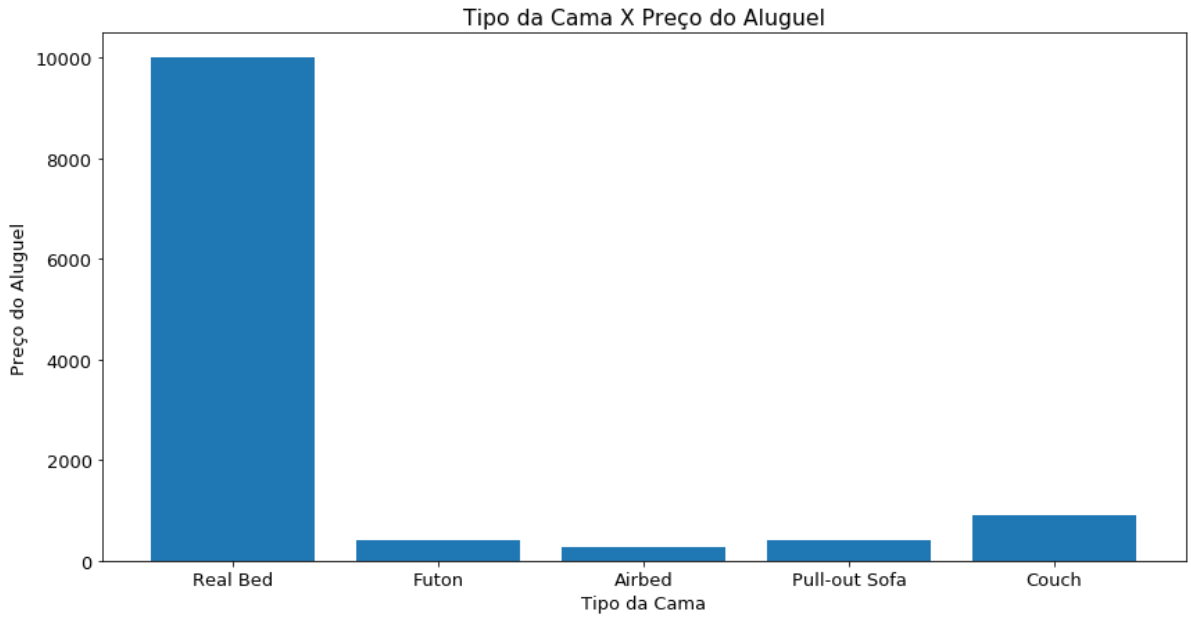
Na fígura abaixo, podemos ver que termos aluguel que chegam a ser mais altos com a quantidade maior com dois quartos.

In [20]:
```
# Comparando a relação de número de quartos com o preço do aluguel

print('\n')

plt.rcParams['figure.figsize'] = (14,7)
plt.bar(airbnb.bedrooms, airbnb.price)
plt.xlabel("Quantidade de Quartos")
plt.ylabel("Preço do Aluguel")
plt.title("Quantidade de Quartos X Preço do Aluguel")
plt.show()
```

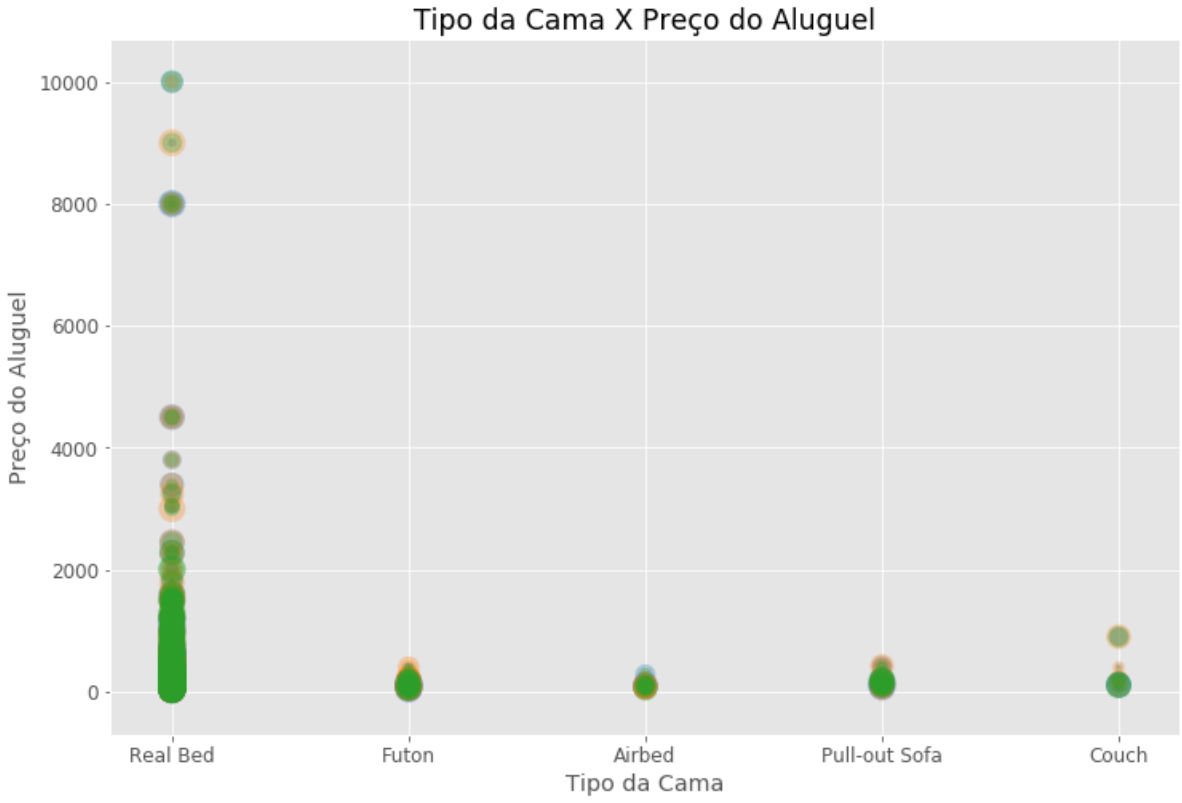


Termos muito mais uma quantidade maior para camas do tipo Real Bed, creio que seja camas mais tradicionais, com preços chegam a ser mais altos.

In [21]:
```
# Comparando a relação do tipo da cama com o preço do aluguel

print('\n')

plt.bar(airbnb.bed_type, airbnb.price)
plt.xlabel("Tipo da Cama")
plt.ylabel("Preço do Aluguel")
plt.title("Tipo da Cama X Preço do Aluguel")
plt.show()
```



In [30]:
```
# Mudando a visualização do gráfico de cima

plt.rcParams.update({'font.size': 12})
fig, ax = plt.subplots(figsize=(12,8))


for color in ['tab:blue', 'tab:orange', 'tab:green']:
    x = airbnb.bed_type
    y = airbnb.price
    plt.xlabel("Tipo da Cama")
    plt.ylabel("Preço do Aluguel")
    plt.title("Tipo da Cama X Preço do Aluguel")
    scale = 300 * np.random.rand(100)
    ax.scatter(x, y, c=color, s=scale, label=color,
               alpha=0.3, edgecolors='none')

ax.grid(True)

plt.show()
```



Quantidade de até cem comentarios são os que tem com maior frequência.

In [23]:
```
# Visualizando a frequência de quantidade de comentários

plt.style.use("ggplot")
plt.figure(figsize = (14, 8))
plt.rcParams.update({'font.size': 12})
airbnb["number_of_reviews"].hist(bins = 6, ec = "k", alpha = .6, color = "royalblue")
plt.xlabel("Número de comentários")
plt.ylabel("Frequencia")
plt.title("Quantidade de número de comentários")
plt.show()
```

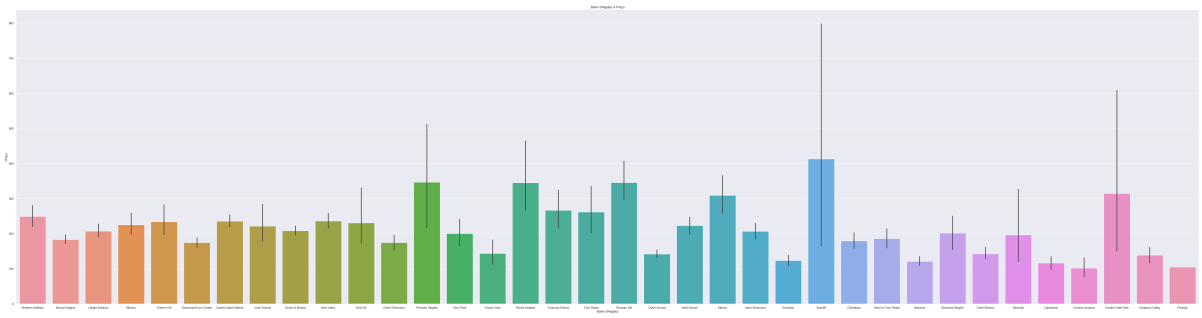
Aqui conseguimos ver que termos pontuação que varia de 20 até 100, sendo que a maiorias das pessoas realização mais as pesquisas onde se teve uma boa pontuação.

In [89]:
```
# Visualizar a frequência de quantidade de avaliação de pontuação

plt.style.use("ggplot")
plt.figure(figsize = (14, 8))
plt.rcParams.update({'font.size': 12})
airbnb["review_scores_rating"].hist(bins = 6, ec = "k", alpha = .6, color = "r
oyalblue")
plt.xlabel("Avaliação de pontuação")
plt.ylabel("Frequencia")
plt.title("Quantidade de avaliação de pontuação")
plt.show()
```

Podemos perceber que a região da propriedade, influência no preço do aluguel.

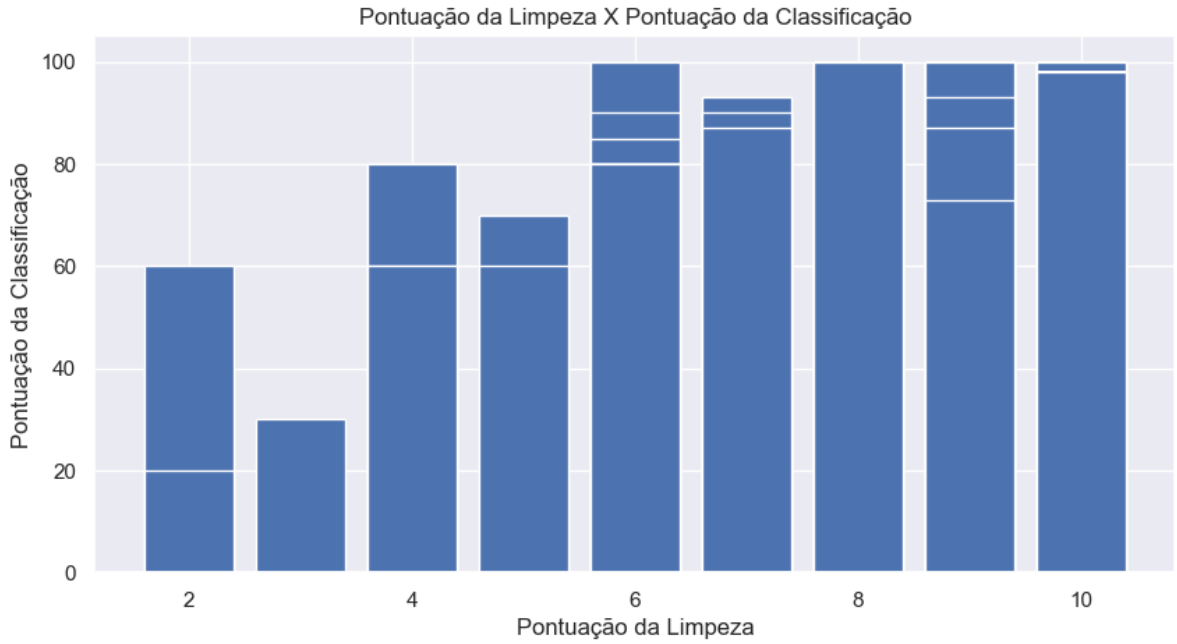
In [70]:
```
# Comparando os bairros (Regiões) com o preço.
# Obersavação: Abra o gráfico em uma nova Guia, clicando com botão direito ->
 Abrir imagem em uma nova guia

sns.set(rc={'figure.figsize':(80,20)})
sns.barplot(x= 'neighbourhood_cleansed', y = 'price', data=airbnb) \
.set(title='Bairro (Região) X Preço', xlabel='Bairro (Região)', ylabel='Preço'
)
```

Out[70]:
```
[Text(0,0.5,'Preço'),
 Text(0.5,0,'Bairro (Região)'),
 Text(0.5,1,'Bairro (Região) X Preço')]
```



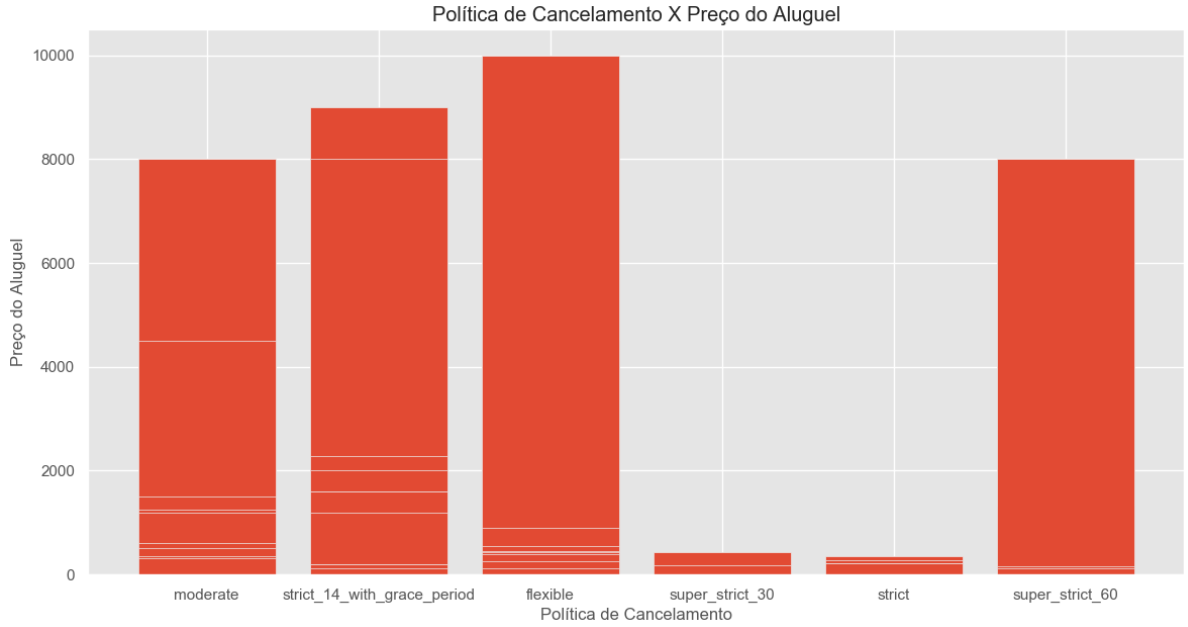
Aqui também conseguimos ver que pontuação menores que seis, possuem uma menor pontuação da classificação do aluguel.

In [83]:
```
# Comparando a relação da pontuação da limpeza com o pontuação da classificaçã
o

plt.rcParams['figure.figsize'] = (10,5)
plt.rcParams.update({'font.size': 12})
plt.bar(airbnb.review_scores_cleanliness, airbnb.review_scores_rating)
plt.xlabel("Pontuação da Limpeza")
plt.ylabel("Pontuação da Classificação")
plt.title("Pontuação da Limpeza X Pontuação da Classificação")
plt.show()
```



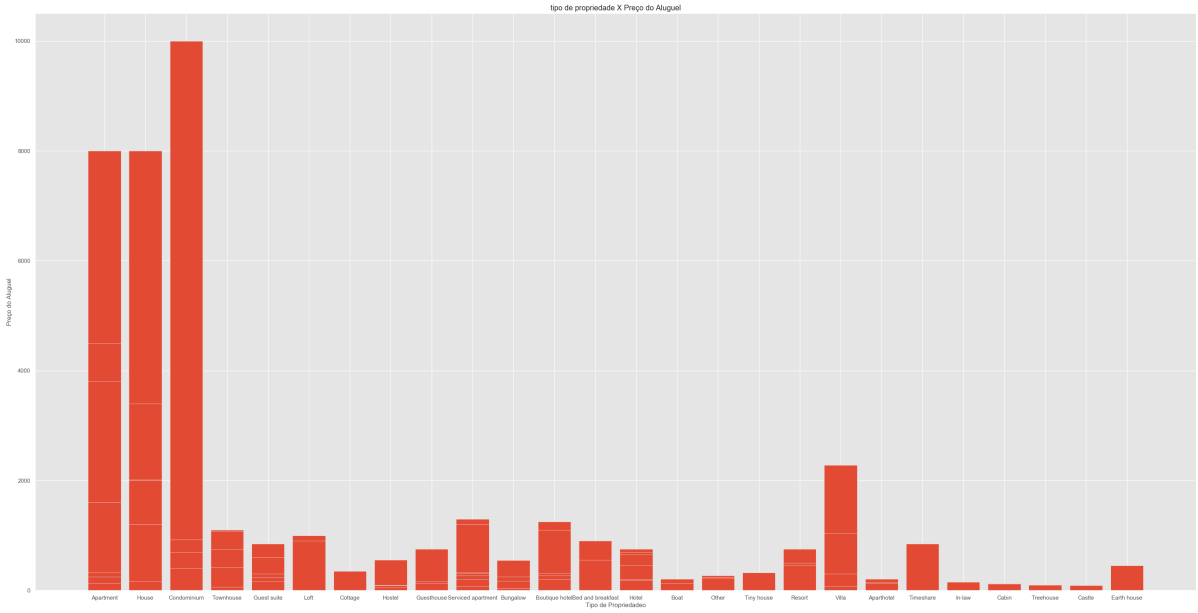
Propriedades que oferecem cancelamento flexível, tem quantidade maior com preços que chegam a ser mais altos.

In [94]:
```
# Comparando a relação política de cancelamento com o preço do aluguel

plt.rcParams['figure.figsize'] = (14,7)
plt.rcParams.update({'font.size': 10})
plt.bar(airbnb.cancellation_policy, airbnb.price)
plt.xlabel("Política de Cancelamento")
plt.ylabel("Preço do Aluguel")
plt.title("Política de Cancelamento X Preço do Aluguel")
plt.show()
```



A quantidade é muito maior com preços que chegam a ser mais altos, com propriedades do tipo, condomio, casa e apartamento.

In [98]:
```
# Comparando a relação tipo de propriedade com o preço do aluguel
# Obersavação: Abra o gráfico em uma nova Guia, clicando com botão direito ->
 Abrir imagem em uma nova guia

plt.rcParams['figure.figsize'] = (40,20)
plt.rcParams.update({'font.size': 10})
plt.bar(airbnb.property_type, airbnb.price)
plt.xlabel("Tipo de propriedade")
plt.ylabel("Preço do Aluguel")
plt.title("tipo de propriedade X Preço do Aluguel")
plt.show()
```



Agora vamos realizar a correlação entre as variáveis.
****

Podemos ver que a correlação é muito baixa da variável Price com as demais variáveis númericas do dataset.

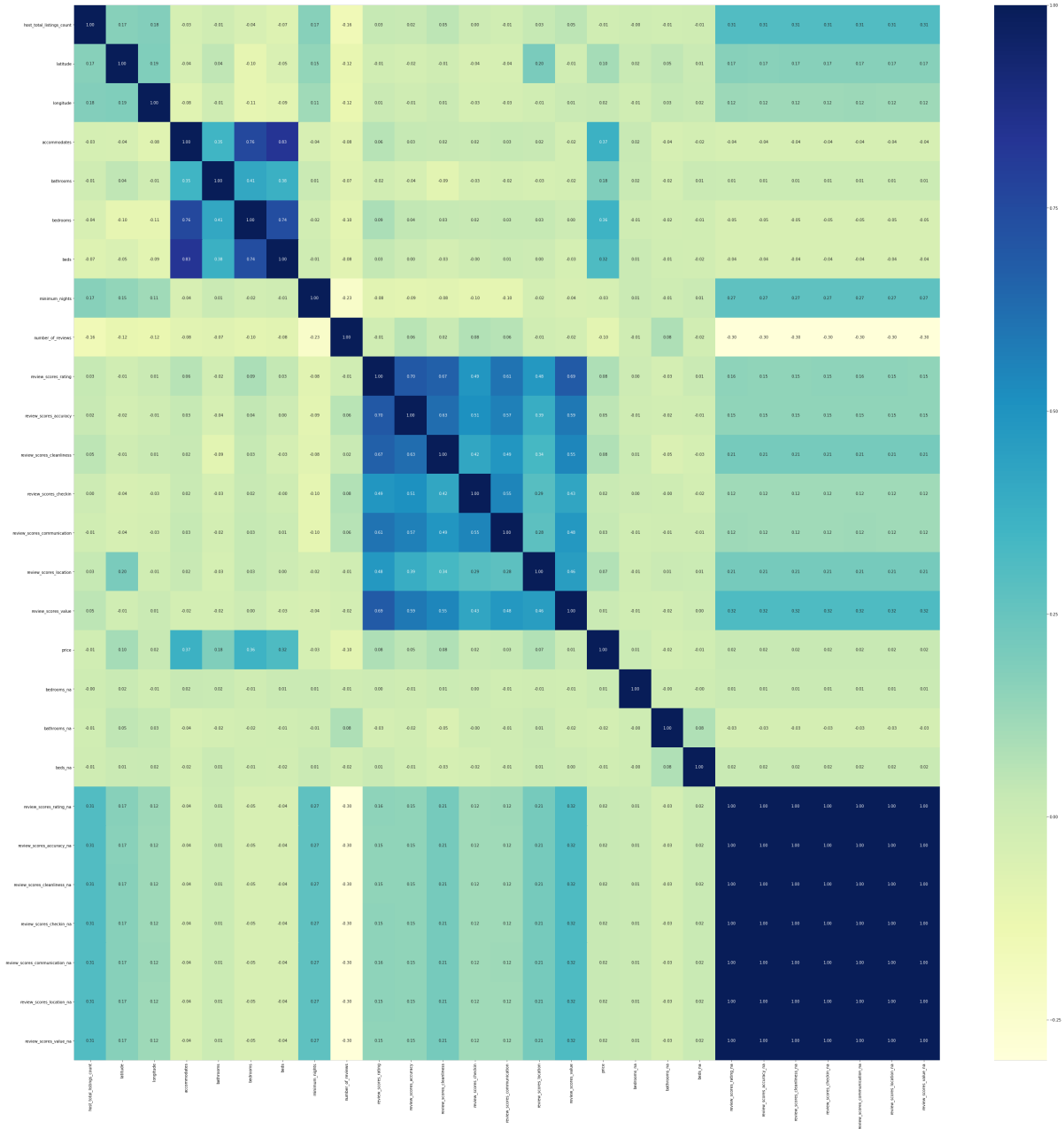
In [14]:
```
# Distribuição das classe

airbnb_col_numericas.corr(method = 'pearson')
```

Out[14]:

	host_total_listings_count	latitude	longitude	accommodates
host_total_listings_count	1.000000	0.168420	0.179394	-0.031795
latitude	0.168420	1.000000	0.186439	-0.040640
longitude	0.179394	0.186439	1.000000	-0.079097
accommodates	-0.031795	-0.040640	-0.079097	1.000000
bathrooms	-0.009684	0.043451	-0.009345	0.351962
bedrooms	-0.044223	-0.101949	-0.105547	0.758461
beds	-0.070620	-0.048615	-0.088636	0.832837
minimum_nights	0.168062	0.146478	0.111562	-0.044867
number_of_reviews	-0.162465	-0.116828	-0.123448	-0.081727
review_scores_rating	0.026785	-0.013030	0.012461	0.064387
review_scores_accuracy	0.021991	-0.021110	-0.006865	0.028920
review_scores_cleanliness	0.050223	-0.005867	0.010879	0.022402
review_scores_checkin	0.002453	-0.041226	-0.031385	0.018343
review_scores_communication	-0.014081	-0.039130	-0.026124	0.027383
review_scores_location	0.025204	0.201675	-0.010708	0.022239
review_scores_value	0.047182	-0.014218	0.012213	-0.022997
price	-0.012935	0.104869	0.021325	0.372839
bedrooms_na	-0.002981	0.015008	-0.014056	0.024458
bathrooms_na	-0.012950	0.045186	0.025316	-0.038105
beds_na	-0.005956	0.014314	0.018139	-0.021978
review_scores_rating_na	0.308821	0.168140	0.122144	-0.038223
review_scores_accuracy_na	0.308090	0.167269	0.121281	-0.037964
review_scores_cleanliness_na	0.308273	0.167420	0.121356	-0.038120
review_scores_checkin_na	0.307728	0.167448	0.120575	-0.038383
review_scores_communication_na	0.308456	0.167605	0.122189	-0.037910
review_scores_location_na	0.307728	0.167448	0.120575	-0.038383
review_scores_value_na	0.307547	0.167780	0.120375	-0.038593

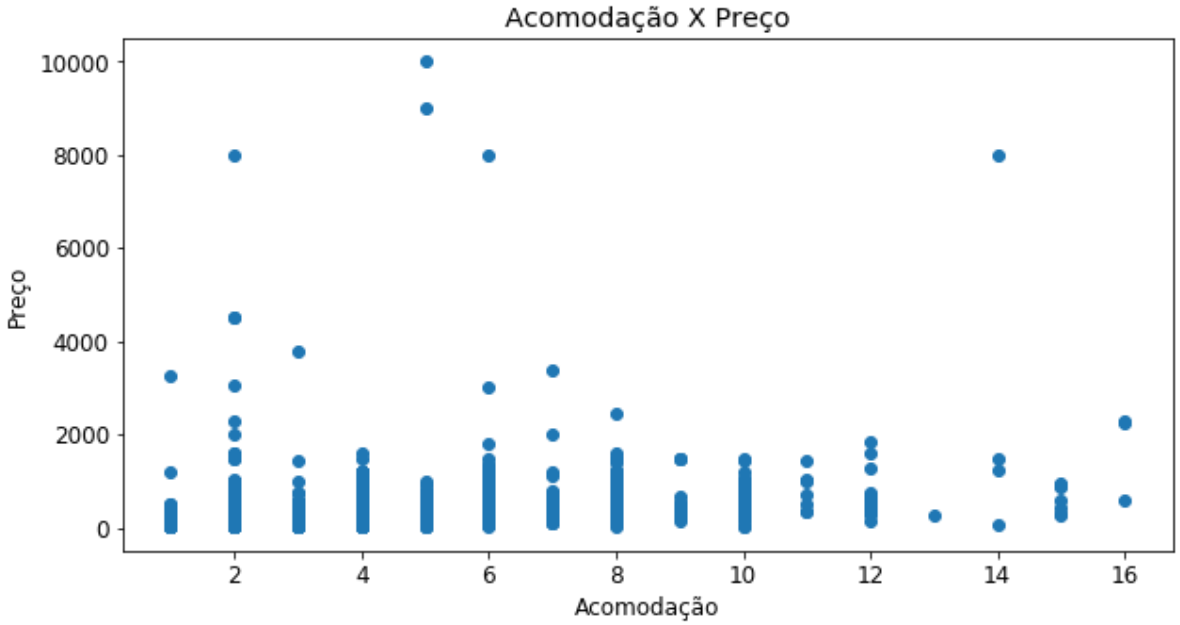
27 rows × 27 columns

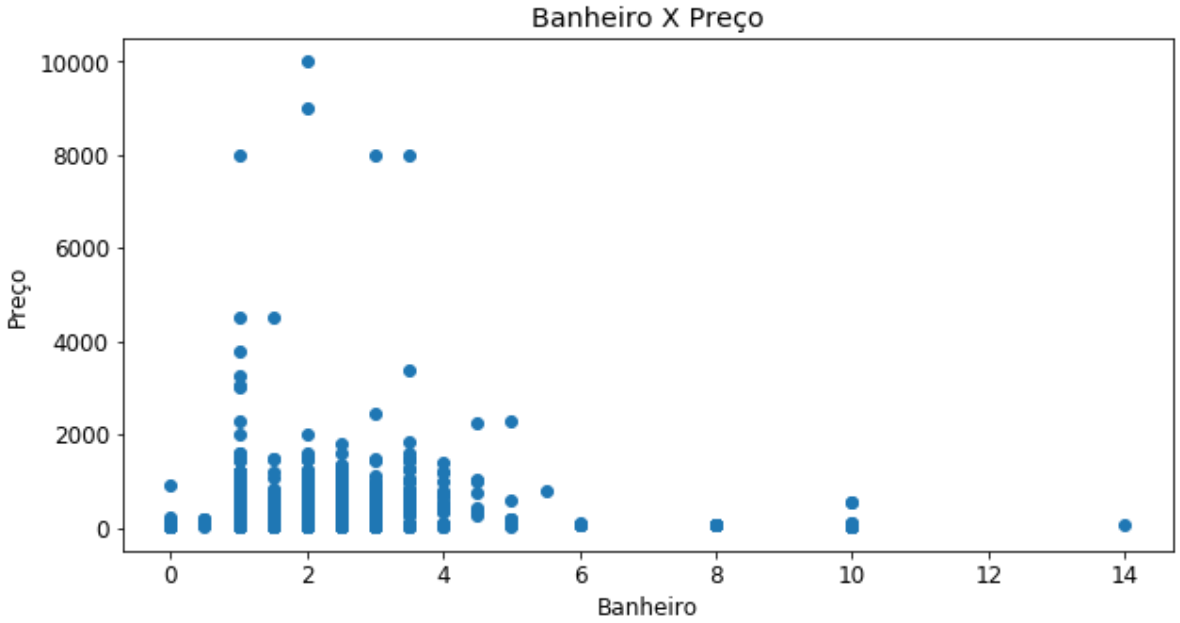
In [15]:
```
# Visualizando correlação entre as varíaveis através do gráfico

plt.figure(figsize=(50,50))
cor = airbnb_col_numericas.corr()
sns.heatmap(cor, annot=True, cmap="YlGnBu", fmt=".2f")
plt.show()
```

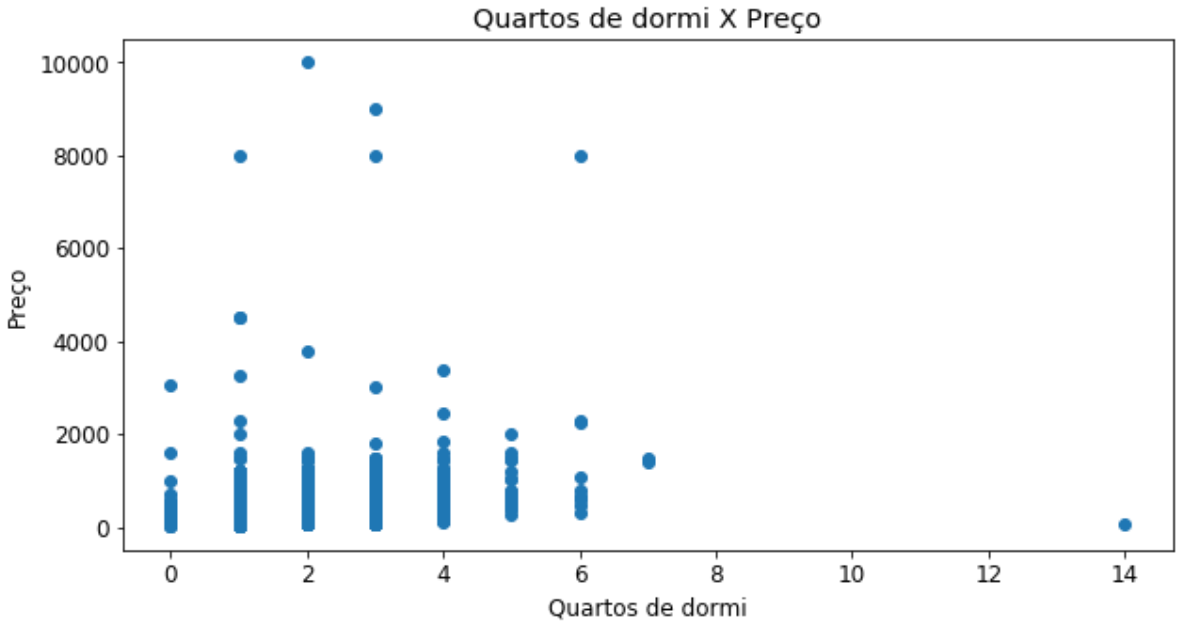


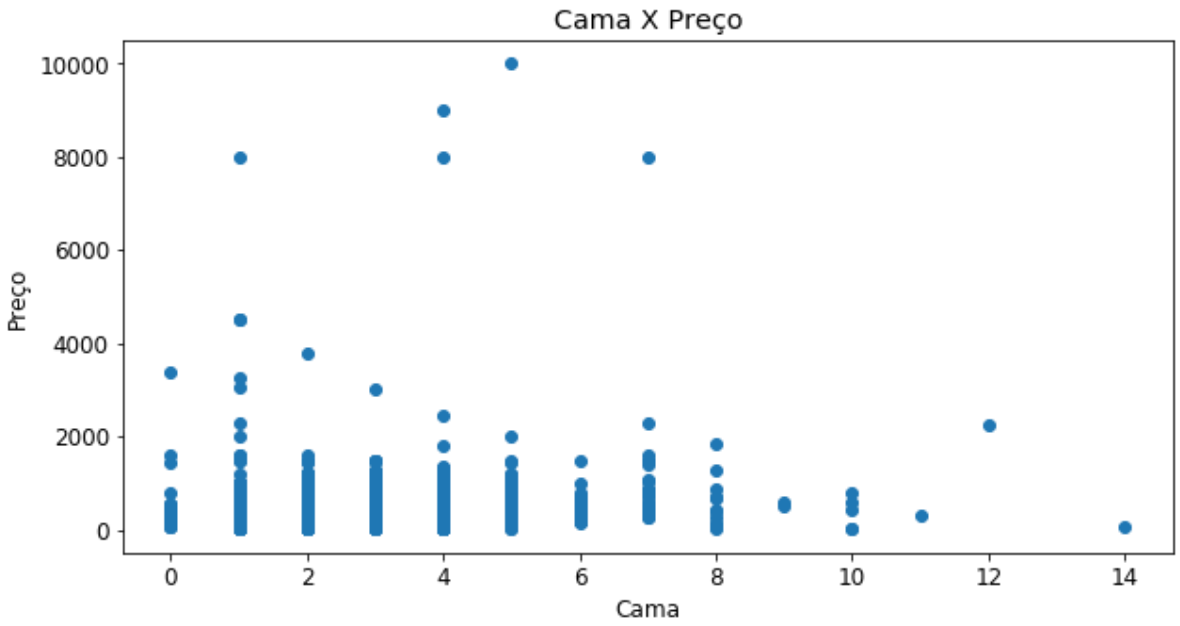
In [249]:
```
# Função para criar o gráfico de dispersão.
# Compara as varíaveis independente(Price) que mais tem correlação linear com
 a varíavel dependente.

def corr_linear(coluna_independente, coluna_dependente, titulo_x, titulo_y):
    plt.figure(figsize=(10,5))
    plt.rcParams.update({'font.size': 12})
    plt.scatter(coluna_independente, coluna_dependente)
    print('\n')
    plt.title(titulo_x+' X '+titulo_y)
    plt.xlabel(titulo_x)
    plt.ylabel(titulo_y)
    plt.show()
```

In [250]:
```
# Chamando a função para exibir os gráficos

corr_linear(airbnb.accommodates, airbnb.price, "Acomodação", "Preço")

print('\n')

corr_linear(airbnb.bathrooms, airbnb.price, "Banheiro", "Preço")

print('\n')

corr_linear(airbnb.bedrooms, airbnb.price, "Quartos de dormi", "Preço")

print('\n')

corr_linear(airbnb.beds, airbnb.price, "Cama", "Preço")

print('\n')
```


Acomodação X Preço
Preço
Acomodação
0
2000
4000
6000
8000
10000
2
4
6
8
10
12
14
16


Banheiro X Preço
Preço
Banheiro
0
2000
4000
6000
8000
10000
0
2
4
6
8
10
12
14


Quartos de dormi X Preço
Preço
Quartos de dormi
0
2000
4000
6000
8000
10000
0
2
4
6
8
10
12
14


Cama X Preço
Preço
Cama
0
2000
4000
6000
8000
10000
0
2
4
6
8
10
12
14

Preparando o modelo para machine learning

Definimos que a variável Price, será a variável target(preditora), que queremos realizar a previsão.

Para o modelo de machine learning, iremos utilizar o método de Regressão.

Modelo 01

In [119]:
```
# Passando variáveis preditoras para X. Vamos utilizar as variáveis númericas, retirando a variável Price que queremos prever.
# Passando variável target para Y.

X = airbnb_col_numericas.drop('price', axis=1)

Y = airbnb['price']

test_size = 0.30
seed = 50
```

In [120]:
```
# Vamos usar a Padronização (StandardScaler).
# Está técnica ajusta os coeficientes e torna a superfície de erros mais "tratável".

scaler = StandardScaler()
X = StandardScaler().fit_transform(X)
scaler
```

Out[120]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [121]:
```
# Utilização train_test_split, para dividir os dados em 70% para treino e 30% para teste

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size, random_state=seed)
```

In [122]:
```
# Modelo regressão linear

reg_log = linear_model.LinearRegression()
reg_log.fit(X_train,Y_train)

Y_pred = reg_log.predict(X_test)
```

In [123]:
```
# Erro médio quadrado e erro médio absoluto

from sklearn.metrics import mean_squared_error

mean_squared_error = mean_squared_error(Y_test, Y_pred)

erro_medio_absoluto = mean_absolute_error(Y_test, Y_pred)
```

In [124]:
```
# Exibindo os resultados

print ("\nCoeficiente de determinação  r² com dados de treino:", round(reg_log
.score(X_train,Y_train), 2) * 100,'%')

print ("\nCoeficiente de determinação r² com os dados de teste:", round(reg_lo
g.score(X_test,Y_test), 2) * 100,'%')

print("\nErro médio quadrado: ", round(mean_squared_error))

print("\nErro médio absoluto:", round(erro_medio_absoluto))
```

```
Coeficiente de determinação  r² com dados de treino: 25.0 %

Coeficiente de determinação r² com os dados de teste: 10.0 %

Erro médio quadrado:  134500.0

Erro médio absoluto: 99.0
```

In [125]:
```
# Exibindo os resultados de forma detalhada

resultado = sm.add_constant(X_train)
LR_model = sm.OLS(Y_train, resultado).fit()
print(LR_model.summary())
```

OLS Regression Results

Dep. Variable:	price	R-squared:	0.252
Model:	OLS	Adj. R-squared:	0.248
Method:	Least Squares	F-statistic:	67.08
Date:	Sun, 21 Feb 2021	Prob (F-statistic):	1.63e-290
Time:	18:39:08	Log-Likelihood:	-34423.
No. Observations:	5002	AIC:	6.890e+04
Df Residuals:	4976	BIC:	6.907e+04
Df Model:	25		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	209.7793	3.344	62.739	0.000	203.224	216.334
x1	-13.3985	3.626	-3.695	0.000	-20.507	-6.290
x2	41.0576	3.622	11.336	0.000	33.957	48.158
x3	11.5507	3.478	3.321	0.001	4.733	18.369
x4	75.7138	6.816	11.108	0.000	62.352	89.076
x5	7.7679	3.807	2.041	0.041	0.305	15.231
x6	70.5804	5.643	12.508	0.000	59.518	81.642
x7	-22.8280	6.448	-3.540	0.000	-35.469	-10.187
x8	-7.2659	3.565	-2.038	0.042	-14.256	-0.276
x9	-16.7015	3.646	-4.581	0.000	-23.848	-9.555
x10	23.6357	6.476	3.650	0.000	10.940	36.331
x11	-2.2742	5.200	-0.437	0.662	-12.469	7.921
x12	20.1766	4.952	4.075	0.000	10.469	29.884
x13	-2.3347	4.474	-0.522	0.602	-11.105	6.436
x14	2.7636	4.806	0.575	0.565	-6.658	12.185
x15	5.1515	3.987	1.292	0.196	-2.666	12.969

```
x16          -25.9065      5.077     -5.103      0.000     -35.860     -15.95
3
x17           -1.9155      3.961     -0.484      0.629      -9.681       5.85
0
x18           -2.3118      3.347     -0.691      0.490      -8.873       4.25
0
x19           -1.8963      2.821     -0.672      0.502      -7.427       3.63
5
x20           31.6637     66.817      0.474      0.636     -99.326     162.65
4
x21           43.8167    116.238      0.377      0.706    -184.061     271.69
4
x22          129.0646    140.717      0.917      0.359    -146.803     404.93
2
x23          -17.2403     58.058     -0.297      0.767    -131.059      96.57
9
x24         -163.0978    122.239     -1.334      0.182    -402.739      76.54
4
x25          -17.2403     58.058     -0.297      0.767    -131.059      96.57
9
x26           -6.6863     95.010     -0.070      0.944    -192.949     179.57
6
==============================================================================
=
Omnibus:                    10673.834   Durbin-Watson:                   1.99
3
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         61821649.99
4
Skew:                          18.508   Prob(JB):                         0.0
0
Kurtosis:                     546.374   Cond. No.                     2.25e+1
5
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
[2] The smallest eigenvalue is 7.63e-27. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Modelo 02

In [210]:
```
# Deixando apenas as colunas que contém alguma correlação com a coluna Price

colunas = ['accommodates', 'bathrooms', 'bedrooms', 'beds_na', 'beds']
x2 = airbnb_col_numericas.filter(items = colunas)
```

In [211]:
```
# Visualizando as primeiras linhas das colunas escolhidas

x2.head()
```

Out[211]:

	accommodates	bathrooms	bedrooms	beds_na	beds
0	3.0	1.0	1.0	0.0	2.0
1	5.0	1.0	2.0	0.0	3.0
2	2.0	4.0	1.0	0.0	1.0
3	2.0	4.0	1.0	0.0	1.0
4	5.0	1.5	2.0	0.0	2.0

In [212]:
```
# Passando variáveis preditoras para X
# Passando variável target para Y

X2 = x2

Y2 = airbnb['price']

test_size2 = 0.30
random_state = 42
```

In [213]:
```
# Vamos usar a Padronização (StandardScaler).
# Está técnica ajusta os coeficientes e torna a superfície de erros mais "trat
ável".

scaler2 = StandardScaler()
X2 = StandardScaler().fit_transform(X)
scaler2
```

Out[213]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [214]:
```
# Utilização train_test_split, para dividir os dados em 70% para treino e 30%
 para teste

X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X2, Y2, test_size=test
_size2, random_state=random_state)
```

In [215]:
```
# Modelo regressão linear

reg_log2 = linear_model.LinearRegression()
reg_log.fit(X_train2,Y_train2)

Y_pred2 = reg_log.predict(X_test2)
```

In [216]:
```
# Erro médio quadrado e erro médio absoluto

from sklearn.metrics import mean_squared_error

mean_squared_error2 = mean_squared_error(Y_test2, Y_pred2)

erro_medio_absoluto2 = mean_absolute_error(Y_test2, Y_pred2)
```

In [217]:
```
# Exibindo os resultados

print ("\nCoeficiente de determinação  r² com dados de treino:", round(reg_log
.score(X_train2,Y_train2), 2) * 100,'%')

print ("\nCoeficiente de determinação r² com os dados de teste:", round(reg_lo
g.score(X_test2,Y_test2), 2) * 100,'%')

print("\nErro médio quadrado: ", round(mean_squared_error2))

print("\nErro médio absoluto:", round(erro_medio_absoluto2))
```

```
Coeficiente de determinação  r² com dados de treino: 13.0 %

Coeficiente de determinação r² com os dados de teste: 31.0 %

Erro médio quadrado:  29896.0

Erro médio absoluto: 91.0
```

In [218]:
```
# Exibindo os resultados de forma detalhada

resultado2 = sm.add_constant(X_train2)
LR_model2 = sm.OLS(Y_train, resultado2).fit()
print(LR_model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     1.041
Date:                Sun, 21 Feb 2021   Prob (F-statistic):              0.392
Time:                        19:21:49   Log-Likelihood:                -35146.
No. Observations:                5002   AIC:                         7.030e+04
Df Residuals:                    4996   BIC:                         7.034e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        210.7862      3.855     54.679      0.000     203.229     218.344
x1            10.6458      7.527      1.414      0.157      -4.110      25.402
x2             4.4881      4.276      1.050      0.294      -3.895      12.871
x3             1.4790      6.259      0.236      0.813     -10.792      13.750
x4             1.8520      3.818      0.485      0.628      -5.633       9.336
x5            -8.2418      7.318     -1.126      0.260     -22.588       6.104
==============================================================================
Omnibus:                     9604.828   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         30250771.601
Skew:                          14.636   Prob(JB):                         0.00
Kurtosis:                     382.854   Cond. No.                         4.11
==============================================================================

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

In [233]:
```
# Tabela comparando valor do preço real com a previsão

print('Tabela da comparação dos valores real e previstos:')

pd.DataFrame(list(zip(Y_test2[10:20], Y_pred2[10:20])), columns=['Preço','Previsao'])
```

```
Tabela da comparação dos valores real e previstos:
```

Out[233]:

	Preço	Previsao
0	410.0	724.395043
1	235.0	418.682396
2	148.0	147.788437
3	100.0	147.788437
4	100.0	147.788437
5	69.0	147.788437
6	299.0	297.489272
7	290.0	283.235417
8	77.0	215.354300
9	133.0	147.788437

Modelo 03

In [234]:
```
# Deixando apenas as colunas que contém alguma correlação com a coluna Price

colunas2 = ['accommodates', 'bathrooms', 'bedrooms', 'beds_na', 'beds']
x3 = airbnb_col_numericas.filter(items = colunas2)
```

In [235]:
```
# Passando variáveis preditoras para X
# Passando variável target para Y

X3 = x3

Y3 = airbnb['price']
```

In [236]:
```
# Utilização train_test_split, para dividir os dados em 70% para treino e 30%
 para teste

X_train3, X_test3, Y_train3, Y_test3 = train_test_split(X3,
                                                        Y3,
                                                        test_size=0.30,
                                                        random_state=5)
```

In [237]:
```
# Modelo regressão linear

reg_log3 = linear_model.LinearRegression()
reg_log.fit(X_train3,Y_train3)

Y_pred3 = reg_log.predict(X_test3)
```

In [238]:
```
# Erro médio quadrado e erro médio absoluto

from sklearn.metrics import mean_squared_error

mean_squared_error3 = mean_squared_error(Y_test3, Y_pred3)

erro_medio_absoluto3 = mean_absolute_error(Y_test3, Y_pred3)
```

In [239]:
```
# Exibindo os resultados

print ("\nCoeficiente de determinação  r² com dados de treino:", round(reg_log
.score(X_train3,Y_train3), 2) * 100,'%')

print ("\nCoeficiente de determinação r² com os dados de teste:", round(reg_lo
g.score(X_test3,Y_test3), 2) * 100,'%')

print("\nErro médio quadrado: ", round(mean_squared_error3))

print("\nErro médio absoluto:", round(erro_medio_absoluto3))
```

```
Coeficiente de determinação  r² com dados de treino: 15.0 %

Coeficiente de determinação r² com os dados de teste: 16.0 %

Erro médio quadrado:  59830.0

Erro médio absoluto: 93.0
```

In [240]:
```
# Exibindo os resultados de forma detalhada

resultado3 = sm.add_constant(X_train3)
LR_model3 = sm.OLS(Y_train3, resultado3).fit()
print(LR_model3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.151
Model:                            OLS   Adj. R-squared:                  0.151
Method:                 Least Squares   F-statistic:                     178.2
Date:                Sun, 21 Feb 2021   Prob (F-statistic):          5.03e-175
Time:                        19:26:15   Log-Likelihood:                -35668.
No. Observations:                5002   AIC:                         7.135e+04
Df Residuals:                    4996   BIC:                         7.139e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -1.5563      9.595     -0.162      0.871     -20.366      17.254
accommodates   44.5968      4.341     10.274      0.000      36.087      53.106
bathrooms      12.7308      5.932      2.146      0.032       1.101      24.360
bedrooms       63.7952      7.438      8.577      0.000      49.214      78.376
beds_na       -23.4149    151.380     -0.155      0.877    -320.185     273.356
beds          -15.4216      6.849     -2.252      0.024     -28.848      -1.995
==============================================================================
Omnibus:                    10573.760   Durbin-Watson:                   2.020
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         45424133.797
Skew:                          18.255   Prob(JB):                         0.00
Kurtosis:                     468.420   Cond. No.                         171.
==============================================================================

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

In [241]:
```
# Tabela comparando valor do preço real com a previsão

print('Tabela da comparação dos valores real e previstos:')

pd.DataFrame(list(zip(Y_test3[10:20], Y_pred3[10:20])), columns=['Preço','Previsao'])
```

```
Tabela da comparação dos valores real e previstos:
```

Out[241]:

	Preço	Previsao
0	86.0	148.741629
1	400.0	703.605571
2	250.0	148.741629
3	94.0	148.741629
4	200.0	158.718348
5	80.0	116.875625
6	129.0	222.513592
7	100.0	148.741629
8	250.0	222.513592
9	315.0	197.115312

Metodologia utilizada e Conclusão

A primeira parte a ser realizada, foi definir o problema de negócio, realizando perguntas e explorando o cenário atual.

Por exemplo: Onde quero chegar ? Qual é o objetivo ?

Foi realizado pesquisas sobre o assunto e do dataset, para buscar entender o que os dados representam.

Através da analise exploratória, conseguimos realizar pesquisas que foram realizadas no dataset e verificar as características dos imóveis e precificar o aluguel dos imóveis, tendo como alguns motivos que pode definir o valor do aluguel ser mais alto ou mais baixo, e com essas informações, também pode ser utilizada para realizar processos de melhoria.

Podemos ver que a localidade da propriedade, tipo de cama, quantidade de camas e quartos, pontuação de limpeza, ser mais flexível no cancelamento, pode influenciar para mais ou para menos o preço do aluguel.

Ao verificar a correlação das variáveis com a variável Price, verificamos que termos muita variança entre os dados, não sendo linear, sendo assim tendo uma baixa correlação. Muito provavelmente influenciou no baixo valor do coeficiente de determinação na da previsão realizada do modelo utilizado por Regressão.