Seattle Airbnb Open Data

Analysis by:

- Stergios Sozos 8170129
- Aliki Ntouzgou 8170091

A sneak peek into the Airbnb activity in Seattle, WA, USA for the year 2016

Data Info

LISTINGS

Full descriptions and average score
From 2009 to 2015

REVIEWS

Reviewer ID and detailed comments
From 2009 to 2015

CALENDAR

Listing's price and availability each date
From 2016 to 2017

Data metrics

Rows per file

Listings: 3.818

Reviews: 84.849

Calendar: 1.393.570

Columns per file

Listings: 92

Reviews: 6

Calendar: 4

Main categories - Dimensions

LOCATION

PRICE

OPTIONS

FACILITIES

BED

Data Cleaning

Selected usable columns

Fixed NaNs and empty cells

Turned float into int

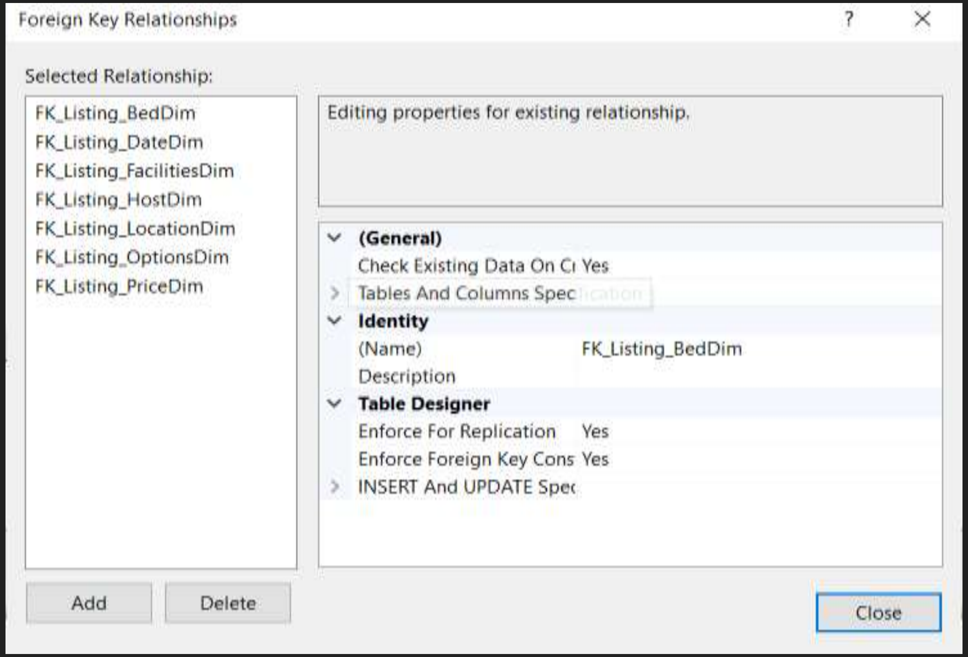
Renamed columns, for better understanding and easier use

Turned “date” column into datetime type

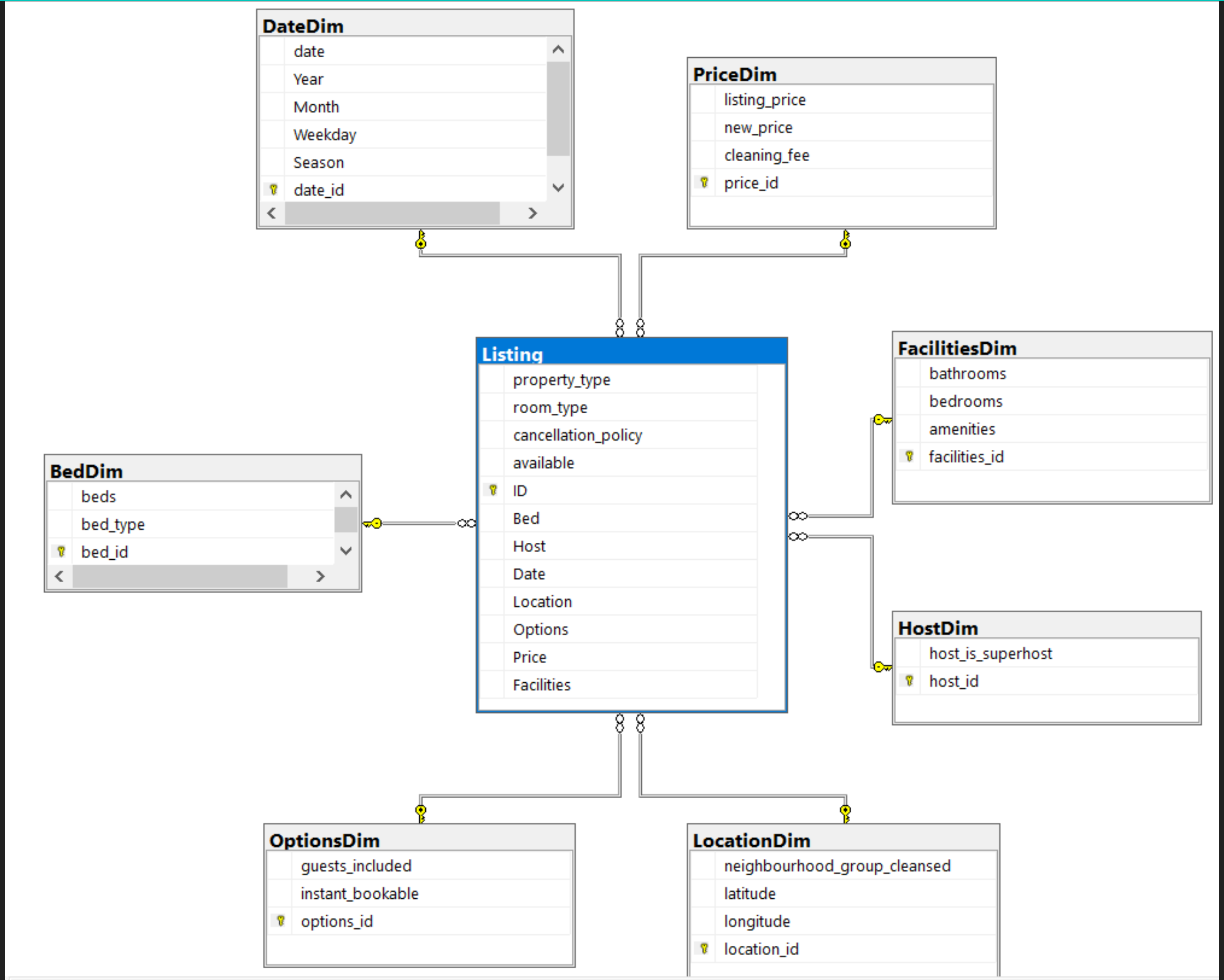
Removed Dollar sign from Prices

Star Scheme

Column Name	Data Type	Allow Nulls
neighbourhood_group_cleansed	nvarchar(50)	☐
latitude	float	☐
longitude	float	☐
location_id	int	☐
		☐



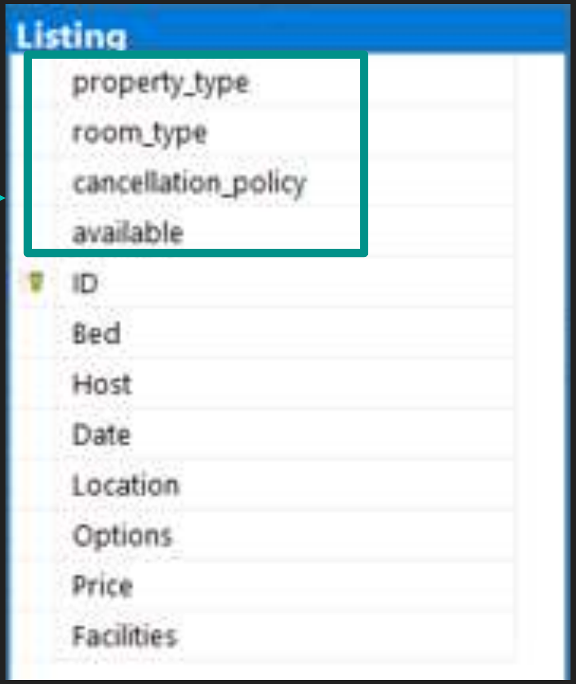
Star Scheme



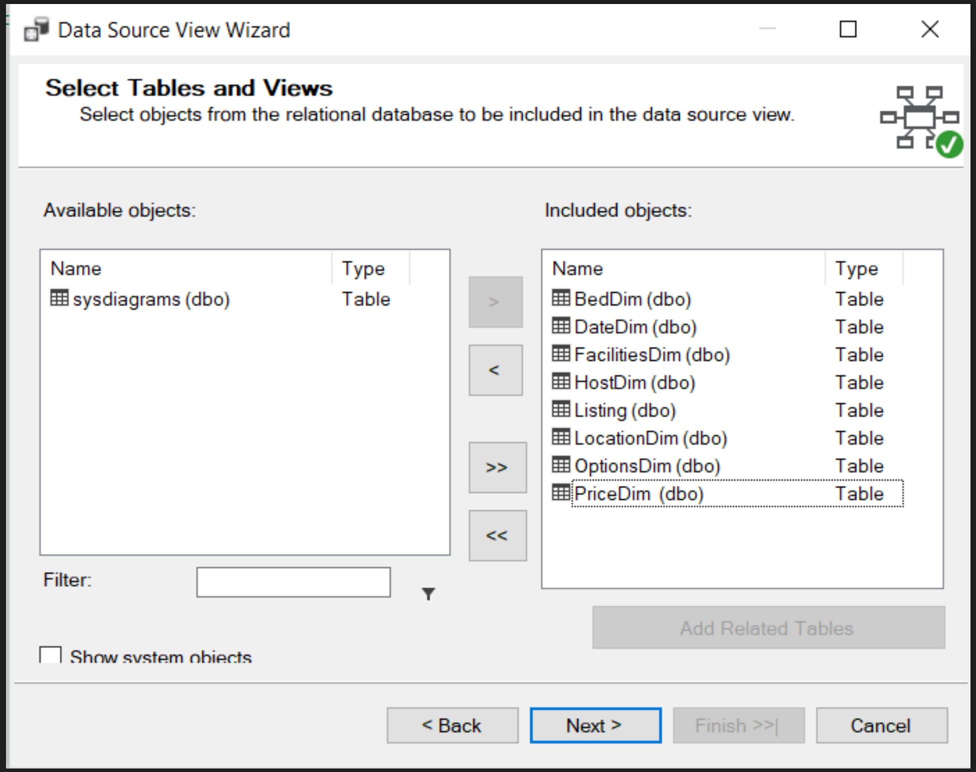
DateDim
date
Year
Month
Weekday
Season
date_id
PriceDim
listing_price
new_price
cleaning_fee
price_id
Listing
property_type
room_type
cancellation_policy
available
ID
Bed
Host
Date
Location
Options
Price
Facilities
FacilitiesDim
bathrooms
bedrooms
amenities
facilities_id
BedDim
beds
bed_type
bed_id
HostDim
host_is_superhost
host_id
OptionsDim
guests_included
instant_bookable
options_id
LocationDim
neighbourhood_group_cleansed
latitude
longitude
location_id

Fact Table – Metrics

Metrics

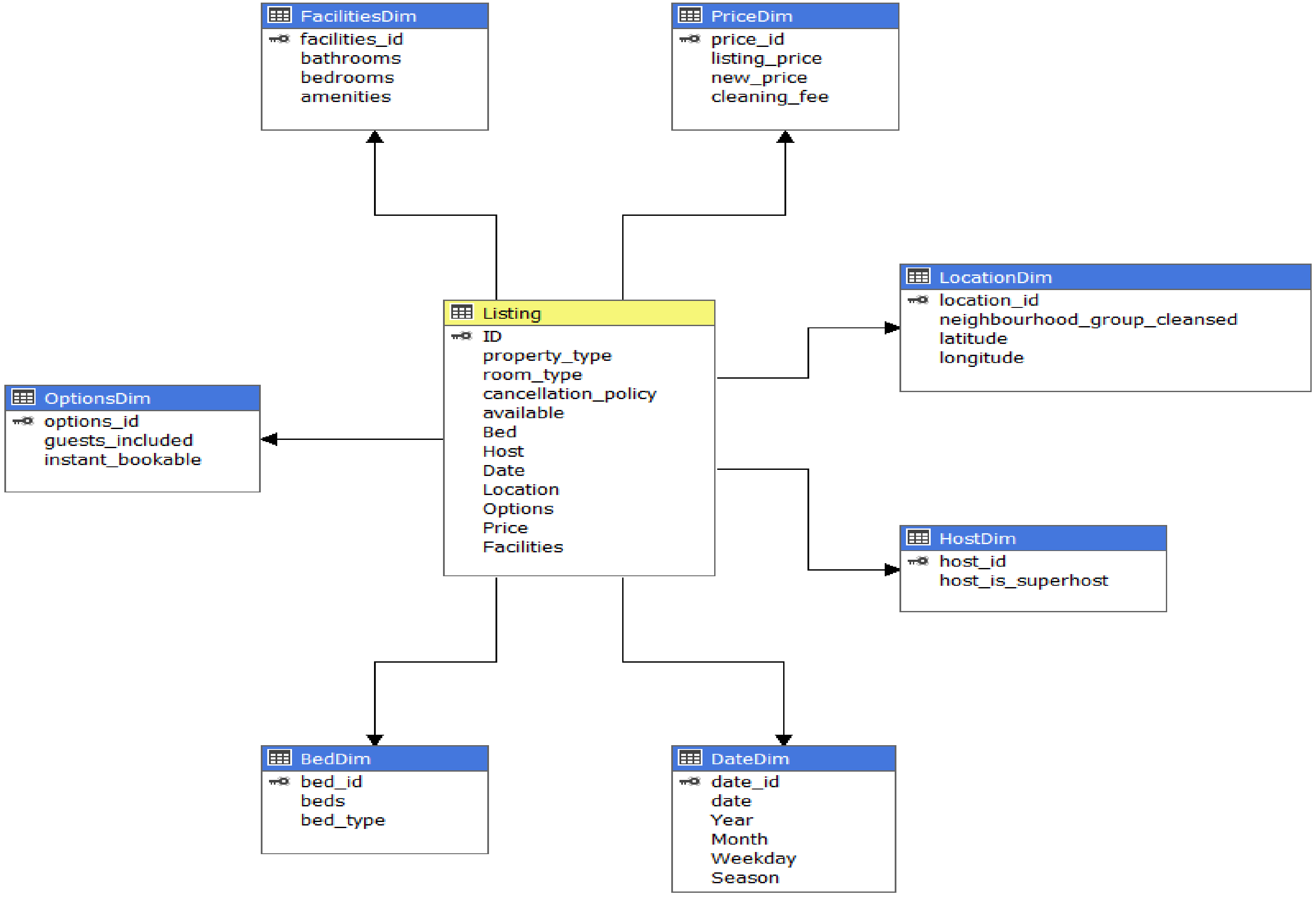


Data Cube



Data Source View Wizard
Select Tables and Views
Select objects from the relational database to be included in the data source view.
Available objects:
Name
Type
sysdiagrams (dbo)
Table
Included objects:
Name
Type
BedDim (dbo)
Table
DateDim (dbo)
Table
FacilitiesDim (dbo)
Table
HostDim (dbo)
Table
Listing (dbo)
Table
LocationDim (dbo)
Table
OptionsDim (dbo)
Table
PriceDim (dbo)
Table
Filter:
Add Related Tables
Show system objects
< Back
Next >
Finish >>|
Cancel

Data Cube


FacilitiesDim
facilities_id
bathrooms
bedrooms
amenities
PriceDim
price_id
listing_price
new_price
cleaning_fee
LocationDim
location_id
neighbourhood_group_cleansed
latitude
longitude
Listing
ID
property_type
room_type
cancellation_policy
available
Bed
Host
Date
Location
Options
Price
Facilities
OptionsDim
options_id
guests_included
instant_bookable
HostDim
host_id
host_is_superhost
BedDim
bed_id
beds
bed_type
DateDim
date_id
date
Year
Month
Weekday
Season

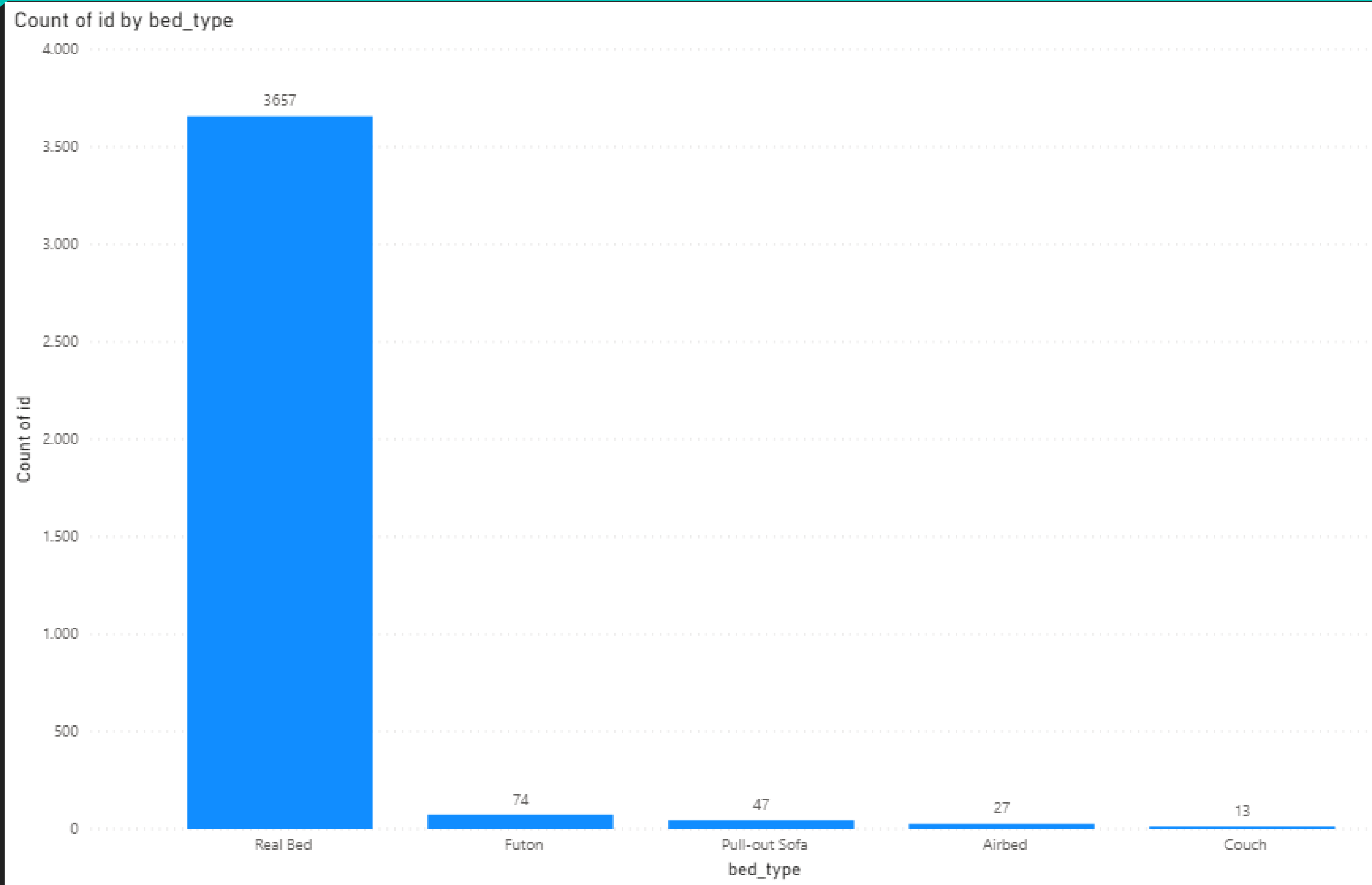
Calculated Measures

- 2699 Total Listings
- 17 Seattle Neighborhoods referred
- 32% of the Cancellation Policies is Strict
- The Average (mean) Regular Price of the Listings is 137$
- The Average (mean) Price of the Listing the day the day they were searched for is 150$
- The Average Price Addition for the Cleaning Fee is 62$
- More than 2/3 of the Searches for Listings were conducted in Winter

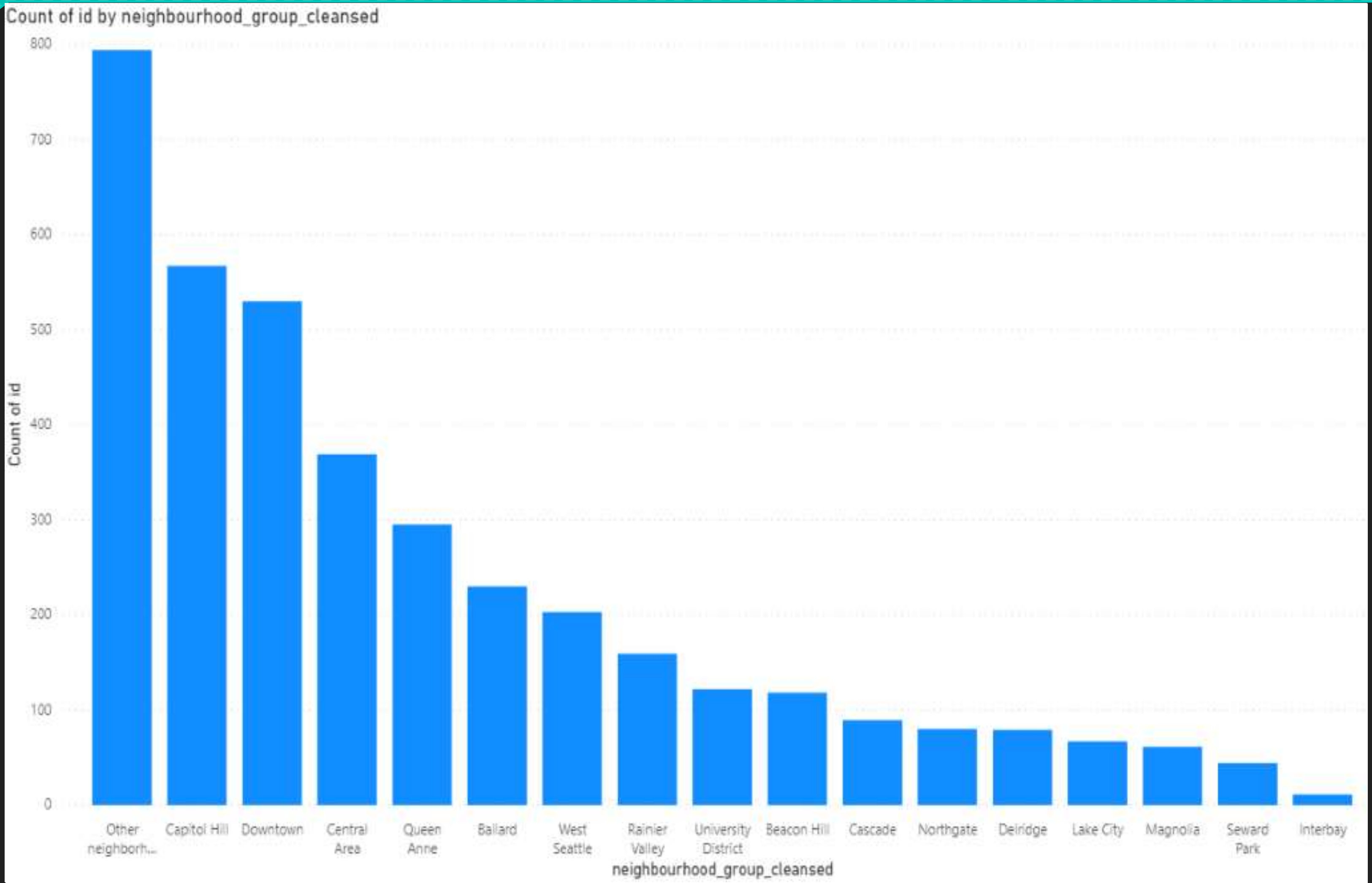
Data Visualization

Power BI Desktop

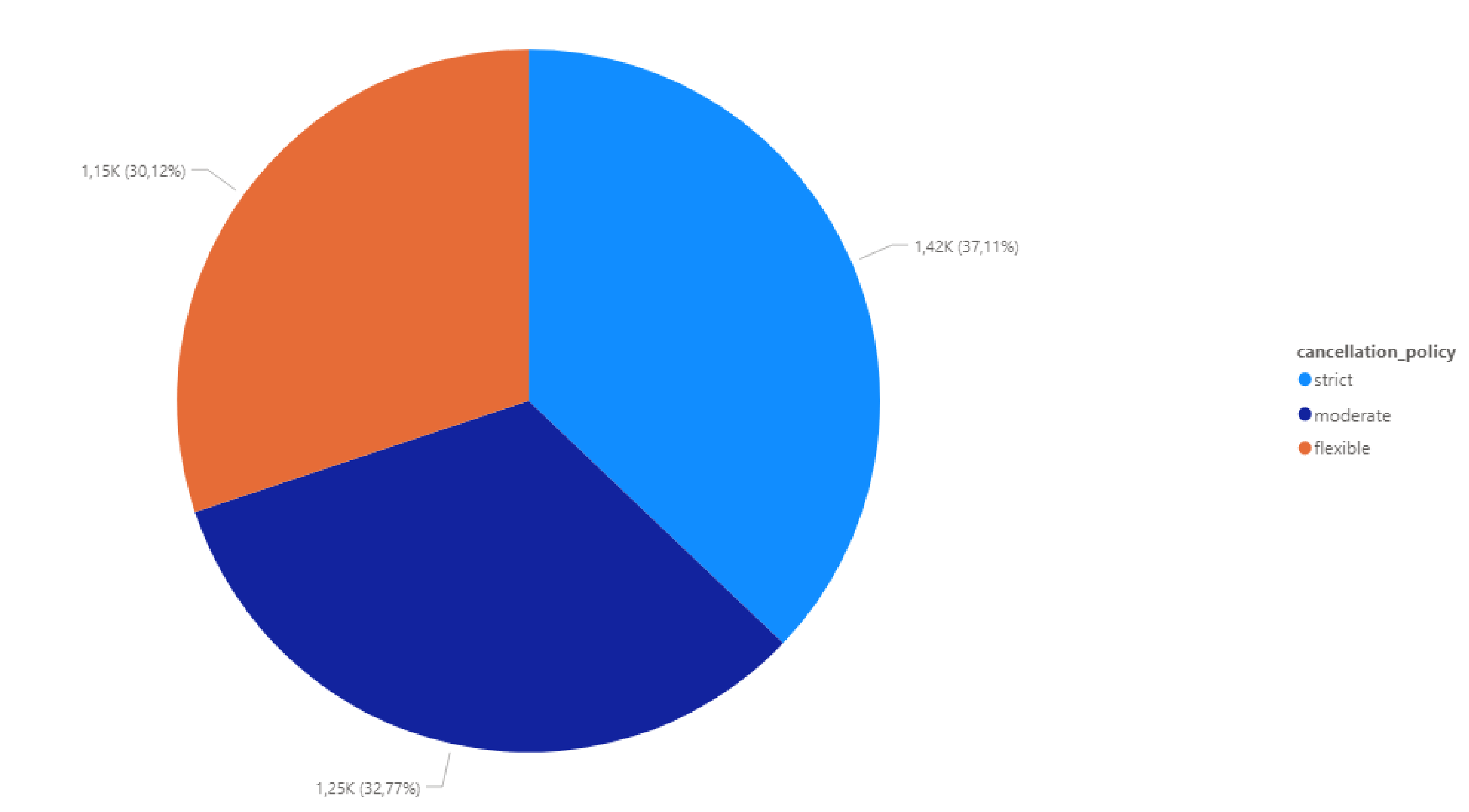
Which is the most popular bed type offered?



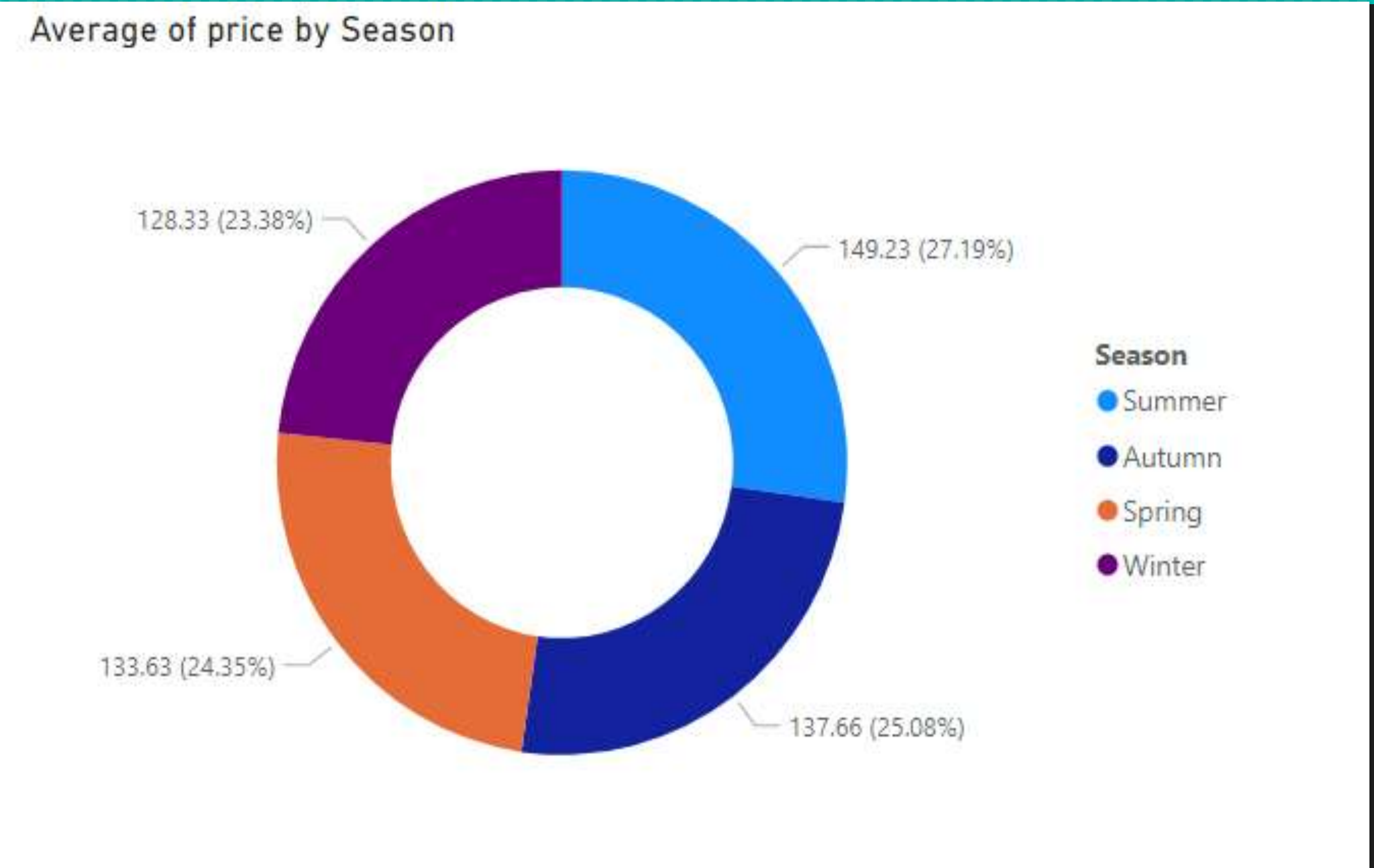
Which is the area with the most listings offered?

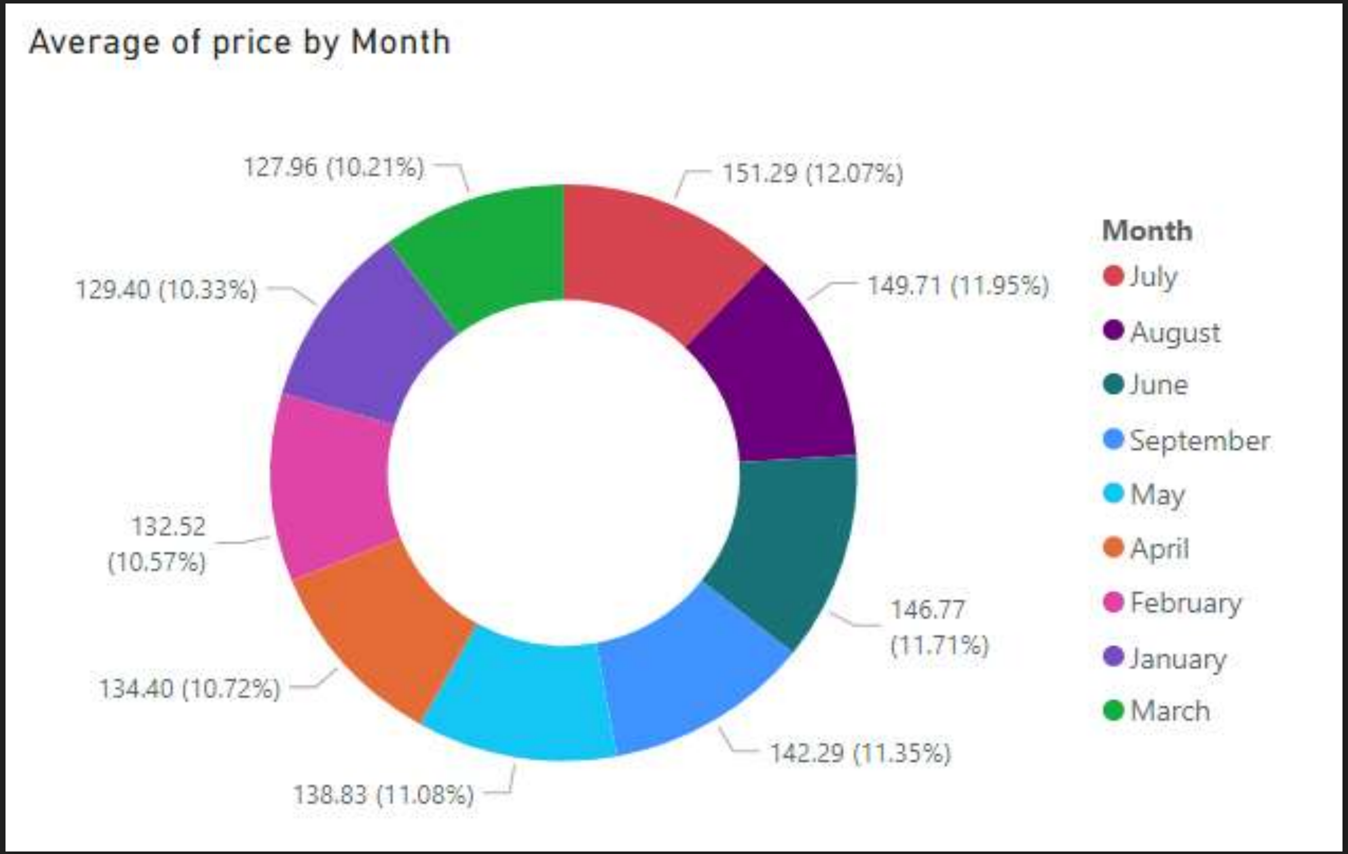


Which is the cancellation policy for the majority of the listings?

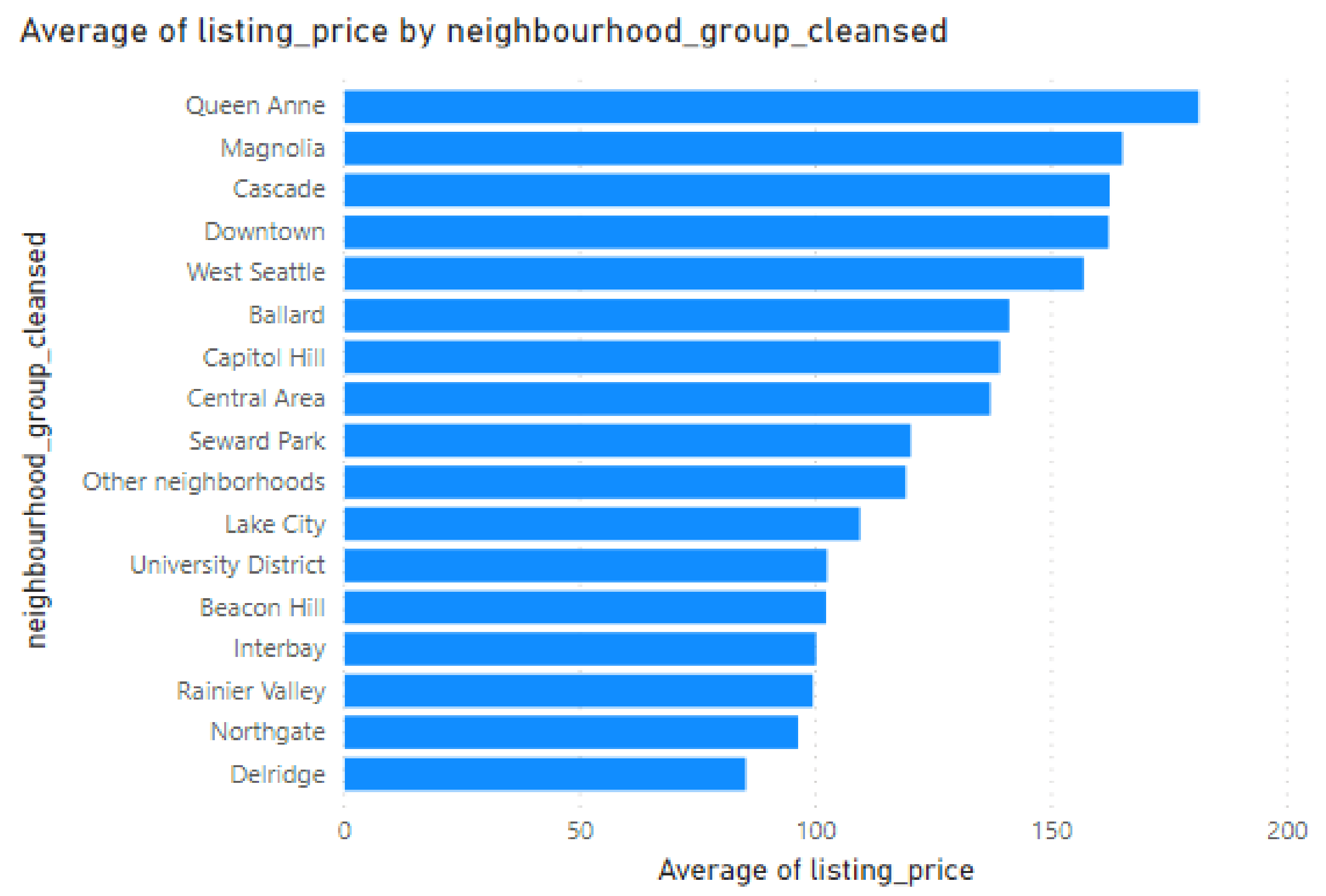


How does the owner of each listing determine on average their price according to season?

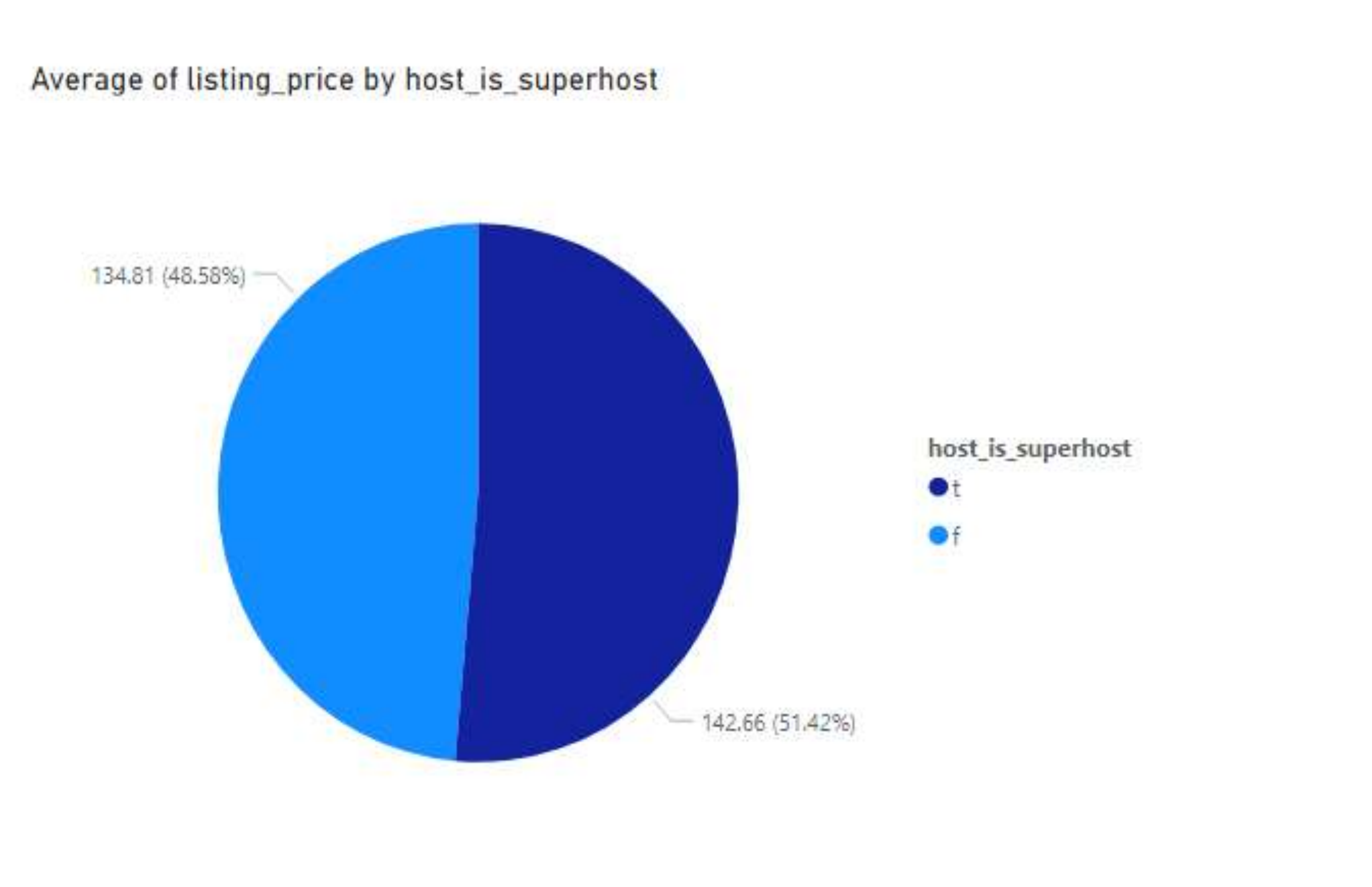



Average of price by Month
127.96 (10.21%)
151.29 (12.07%)
129.40 (10.33%)
149.71 (11.95%)
132.52 (10.57%)
146.77 (11.71%)
134.40 (10.72%)
142.29 (11.35%)
138.83 (11.08%)
Month
July
August
June
September
May
April
February
January
March

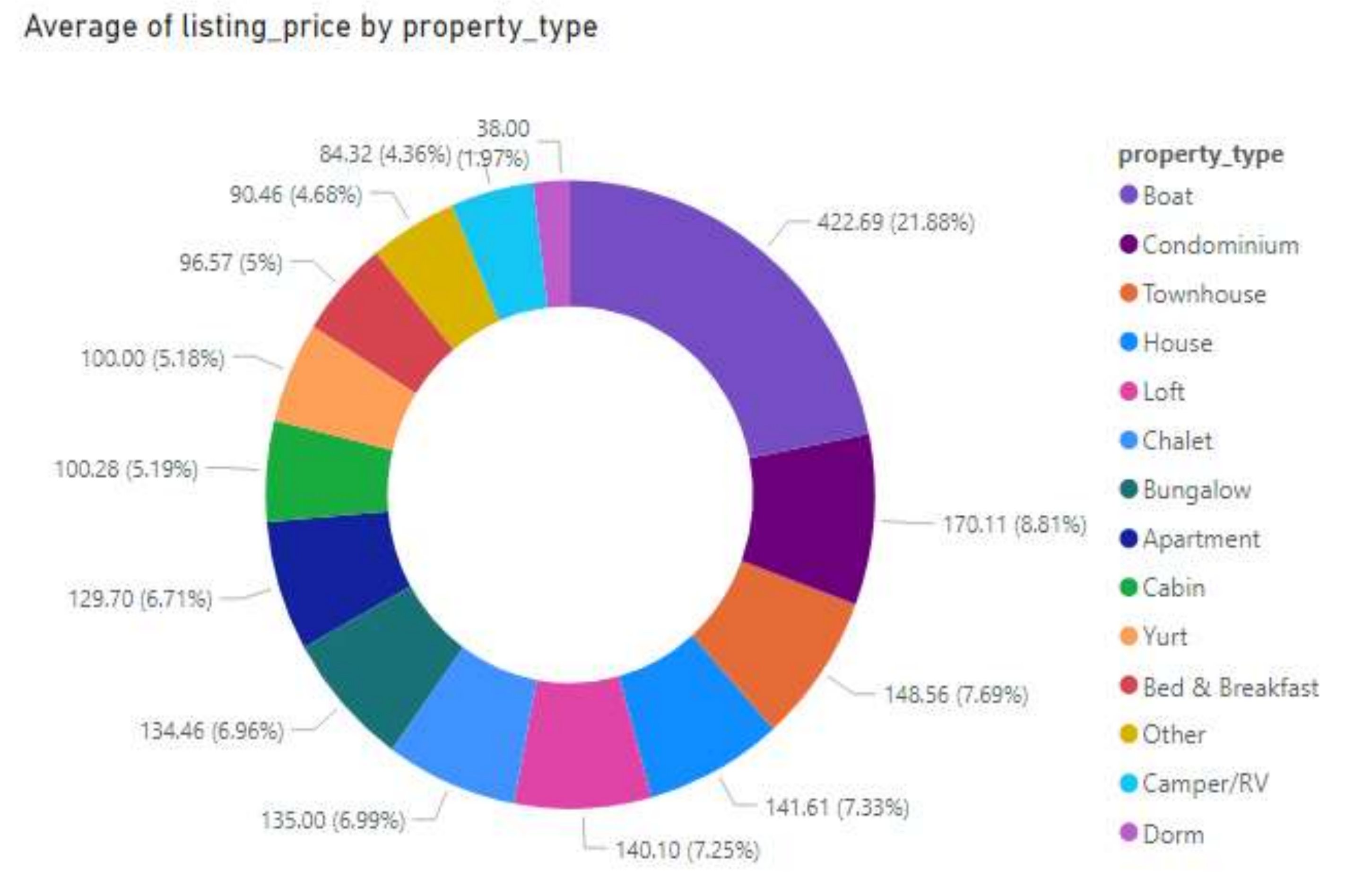
How does the owner of each listing determine on average its price according to region?



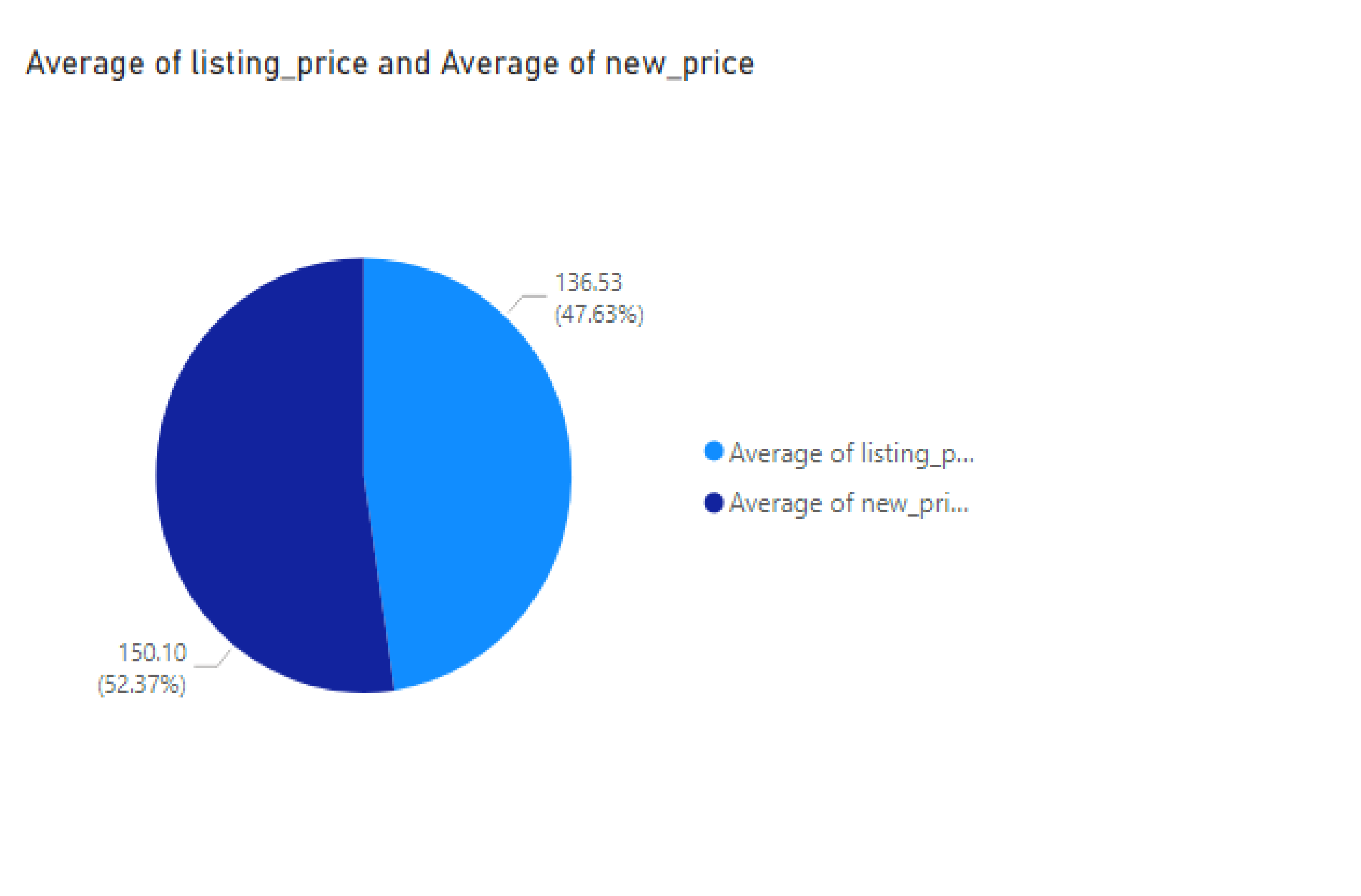
Do superhosts exploit their title to charge more?



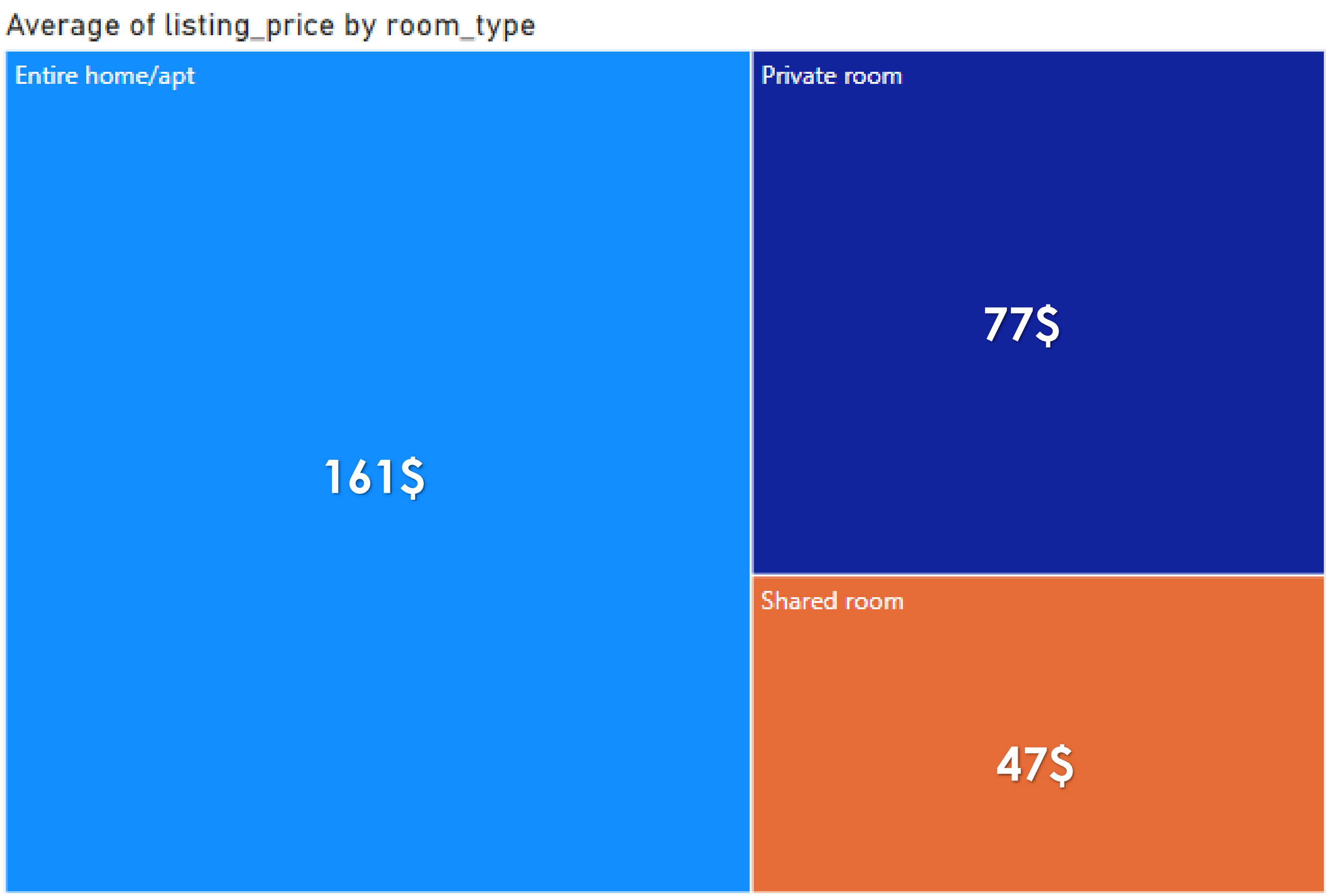
How does the host of each listing determine on average its price according to the type of listing he offers?



How different does the average listing price set by the host differ from the average price when adjusted on the basis of season and demand?

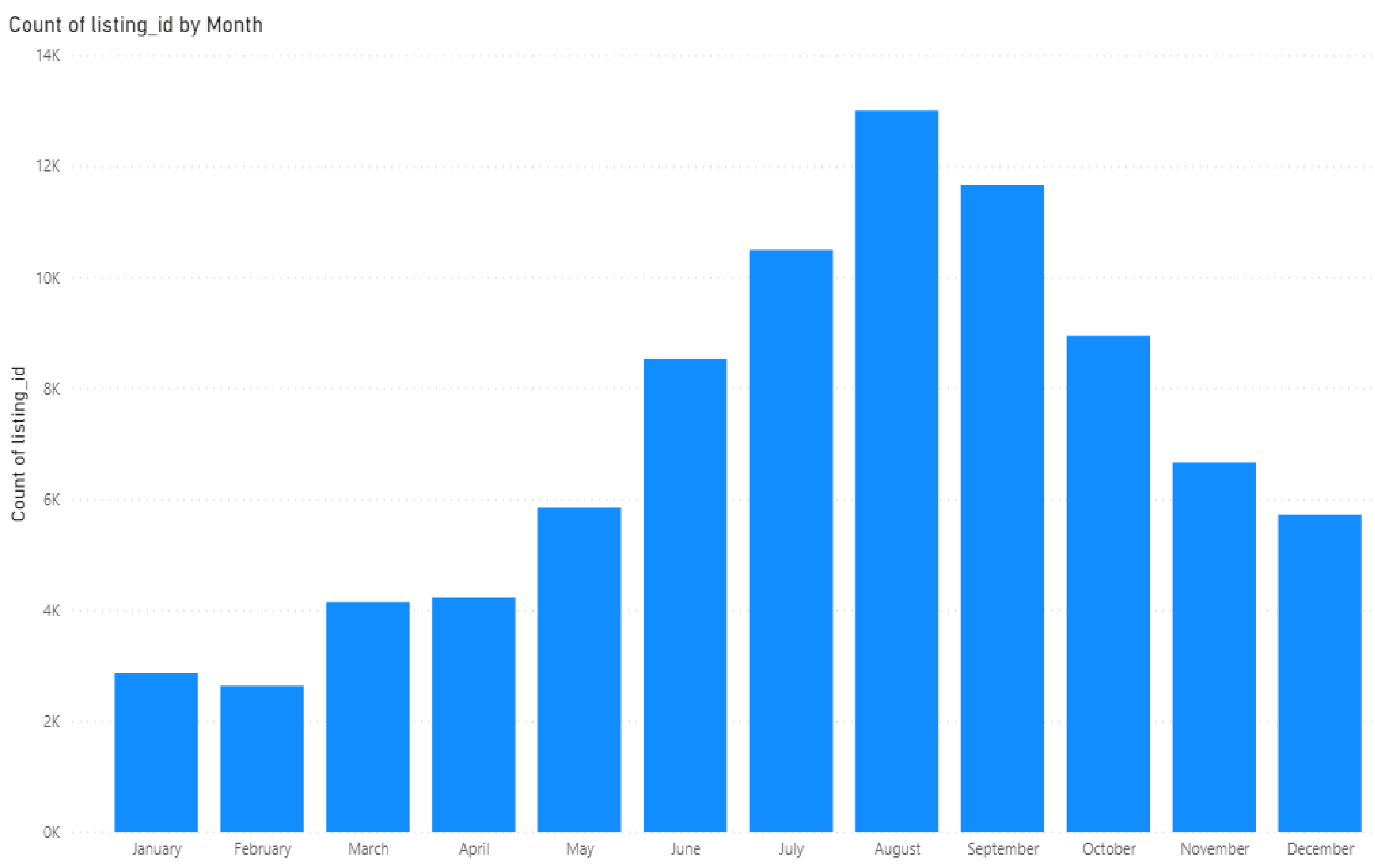


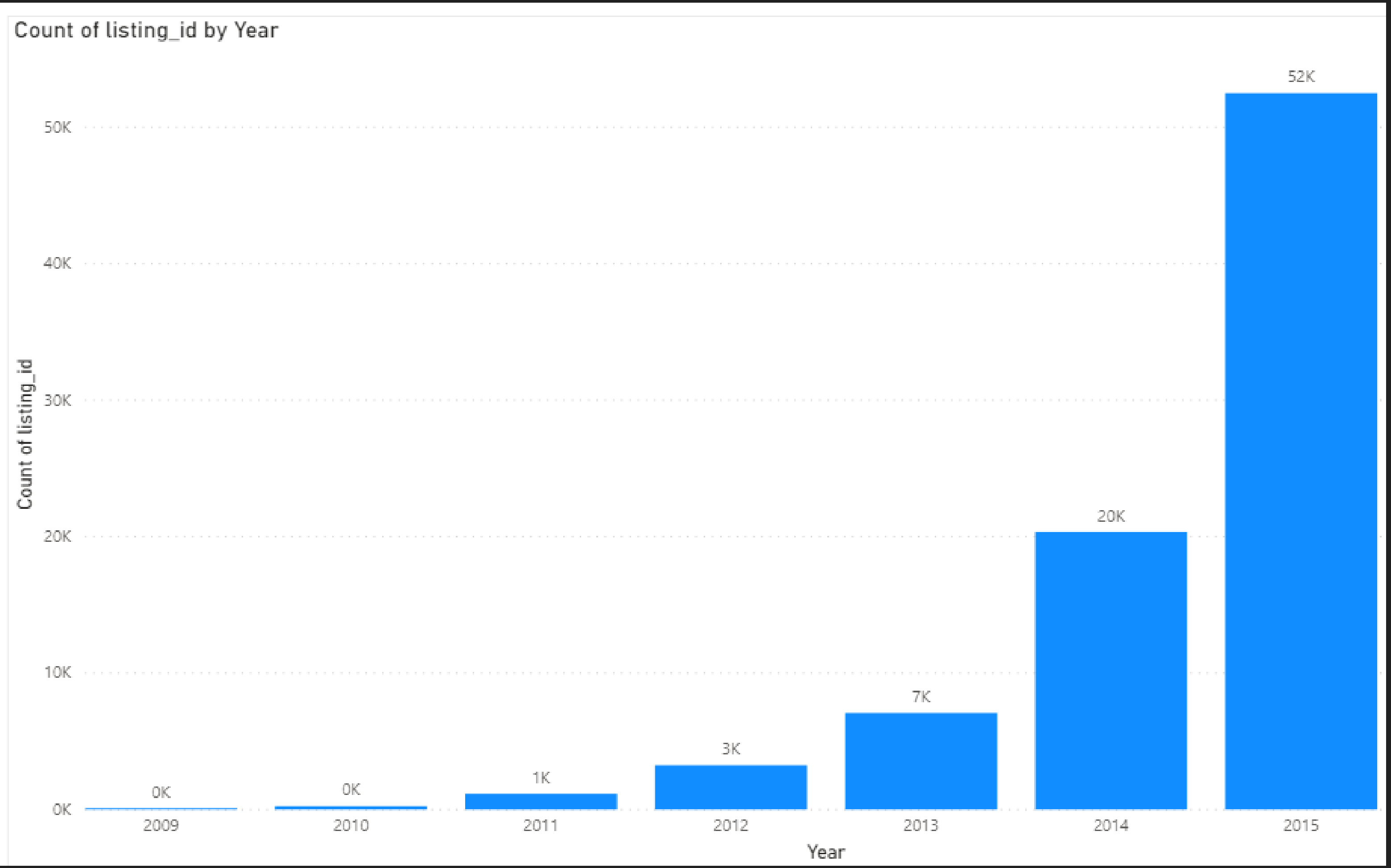
How does each host charge their listing according to the kind of room they offer?



How does each host adapt its listing according to the demand season?

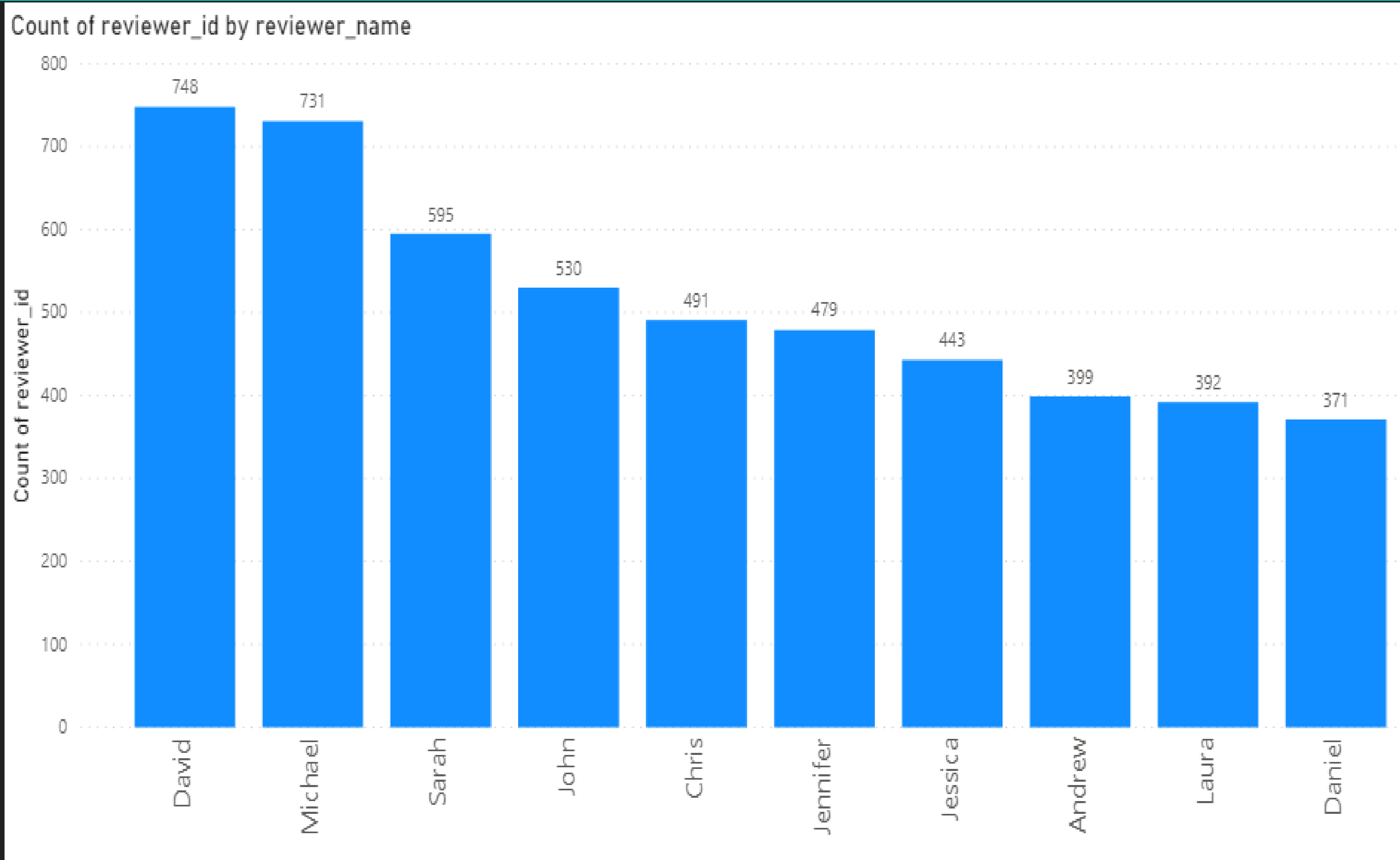
What is the variance of reviews per month?





Count of listing_id by Year
Count of listing_id
50K
40K
30K
20K
10K
0K
0K
0K
1K
3K
7K
20K
52K
2009
2010
2011
2012
2013
2014
2015
Year

What's the most common name among tourists?



Classification

Based on the sentiment of the (English reviews)

TOOLS USED: NLTK LIBRARY FROM PYTHON

NUMBER OF CLASSES: THREE

Steps before classification

- Remove all the reviews that are in any other language other besides English

```
from langdetect import detect

def detect_lang(sente):
    sente=str(sente)
    try:
        return detect(sente)
    except:
        return "None"

for index,row in reviewsDF.iterrows():
    lang=detect_lang(row['comments'])
    reviewsDF.at[index,'language'] = lang
```

Run the classification

- Run the actual classification with the NLTK library

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()

reviews_new['polarity_value']="Default"
reviews_new['neg']=0.0
reviews_new['pos']=0.0
reviews_new['neu']=0.0
reviews_new['compound']=0.0
for index,row in reviews_new.iterrows():
    ss = sid.polarity_scores(row['comments'])
    reviews_new.at[index,'polarity_value'] = ss
    reviews_new.at[index,'neg'] = ss['neg']
    reviews_new.at[index,'pos'] = ss['pos']
    reviews_new.at[index,'neu']= ss['neu']
    reviews_new.at[index,'compound'] = ss['compound']
reviews_new.head()
```

Extract the polarity scores

- Create a dataframe with every information about the reviews and append the columns with the polarity score for each row

	listing_id	id	date	reviewer_id	reviewer_name	comments	polarity_value	neg	pos	neu	com
0	7202016	38917982	2015-07-19	28943674	Bianca	Cute and cozy place. Perfect location to every...	{'neg': 0.0, 'neu': 0.462, 'pos': 0.538, 'comp...	0.000	0.538	0.462	0.7
1	7202016	39087409	2015-07-20	32440555	Frank	Kelly has a great room in a very central locat...	{'neg': 0.0, 'neu': 0.609, 'pos': 0.391, 'comp...	0.000	0.391	0.609	0.9
2	7202016	39820030	2015-07-26	37722850	Ian	Very spacious apartment, and in a great neighb...	{'neg': 0.043, 'neu': 0.772, 'pos': 0.185, 'co...	0.043	0.185	0.772	0.8
3	7202016	40813543	2015-08-02	33671805	George	Close to Seattle Center and all it has to offe...	{'neg': 0.035, 'neu': 0.765, 'pos': 0.2, 'comp...	0.035	0.200	0.765	0.8
4	7202016	41986501	2015-08-10	34959538	Ming	Kelly was a great host and very accommodating ...	{'neg': 0.0, 'neu': 0.655, 'pos': 0.345, 'comp...	0.000	0.345	0.655	0.9

Visualize the classes

- Create 3 different arrays to visualize the polarity scores

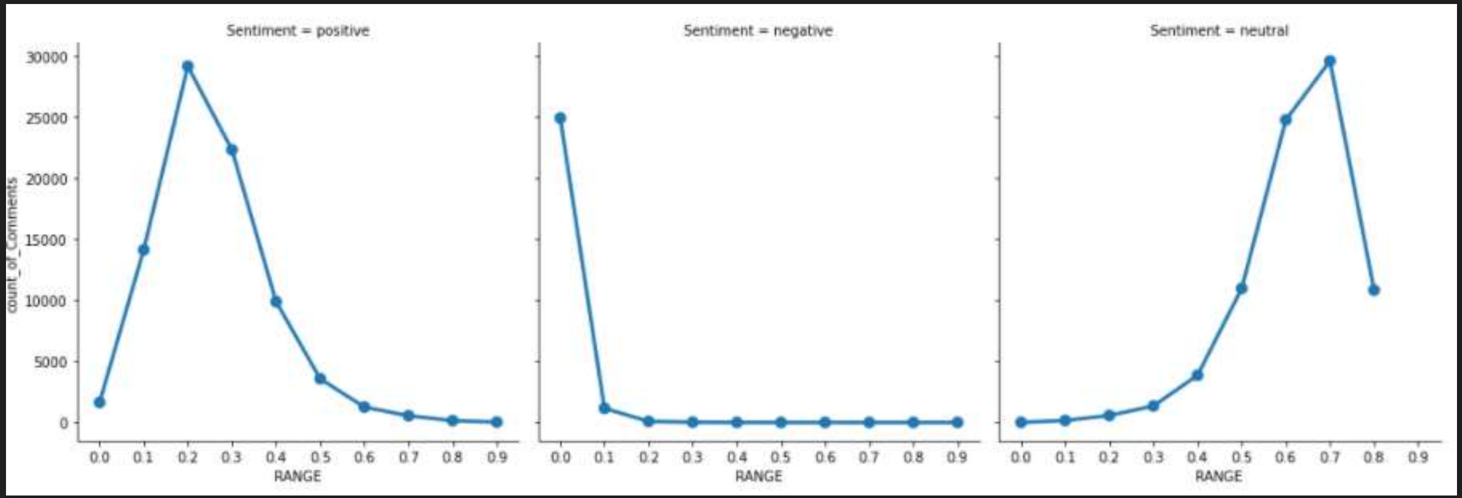
	count_of_Comments	RANGE	Sentiment
0	1640	0.0	positive
1	14126	0.1	positive
2	29151	0.2	positive
3	22306	0.3	positive
4	9881	0.4	positive

	count_of_Comments	RANGE	Sentiment
0	24916	0.0	negative
1	1156	0.1	negative
2	86	0.2	negative
3	21	0.3	negative
4	3	0.4	negative

	count_of_Comments	RANGE	Sentiment
0	8	0.0	neutral
1	157	0.1	neutral
2	569	0.2	neutral
3	1347	0.3	neutral
4	3864	0.4	neutral

Understand the distributions

- Visualize in a better way the polarity scores



Clustering

Based on text descriptions:

- id
- name
- space
- description
- neighborhood_overview
- neighbourhood_cleansed

Clustering

ALGORITHM USED:

K-MEANS

NUMBER OF CLUSTERS:

6

Steps before clustering

Combine column's text into one column

```
# let's combine the name, space, description, and neighborhood_overview into a new column
Listings['combined_description'] = df.apply(lambda x: '{} {} {} {}'.format(x['name'], x['space'],
                                            x['description'], x['neighborhood_overview']), axis=1)
print(Listings.loc[0,'combined_description'])
```

```
Stylish Queen Anne Apartment Make your self at home in this charming one-bedroom apartment, centrally-located on the west side of Queen Anne hill.   T
his elegantly-decorated, completely private apartment (bottom unit of a duplex) has an open floor plan, bamboo floors, a fully equipped kitchen, a TV,
DVD player, basic cable, and a very cozy bedroom with a queen-size bed. The unit sleeps up to four (two in the bedroom and two on the very comfortable
fold out couch, linens included) and includes free WiFi and laundry. The apartment opens onto a private deck, complete with it's own BBQ, overlooking
a garden and a forest of black bamboo.    The Apartment is perfectly-located just one block from the bus lines where you can catch a bus and be downto
wn Seattle in fifteen minutes or historic Ballard in ten or a quick five-minute walk will bring you to Whole Foods and Peet's Coffee or take a fifteen
minute walk to the top of Queen Anne Hill where you will find a variety of eclectic shops, bars, and restaurants. There is no Make your self at home i
n this charming one-bedroom apartment, centrally-located on the west side of Queen Anne hill.   This elegantly-decorated, completely private apartment
(bottom unit of a duplex) has an open floor plan, bamboo floors, a fully equipped kitchen, a TV,  DVD player, basic cable, and a very cozy bedroom wit
h a queen-size bed. The unit sleeps up to four (two in the bedroom and two on the very comfortable fold out couch, linens included) and includes free
WiFi and laundry. The apartment opens onto a private deck, complete with it's own BBQ, overlooking a garden and a forest of black bamboo.    The Apart
ment is perfectly-located just one block from the bus lines where you can catch a bus and be downtown Seattle in fifteen minutes or historic Ballard i
n ten or a quick five-minute walk will bring you to Whole Foods and Peet's Coffee or take a fifteen minute walk to the top of Queen Anne Hill where yo
u will find a variety of eclectic shops, bars, and restaurants. There is no nan
```

Steps before clustering and Clustering

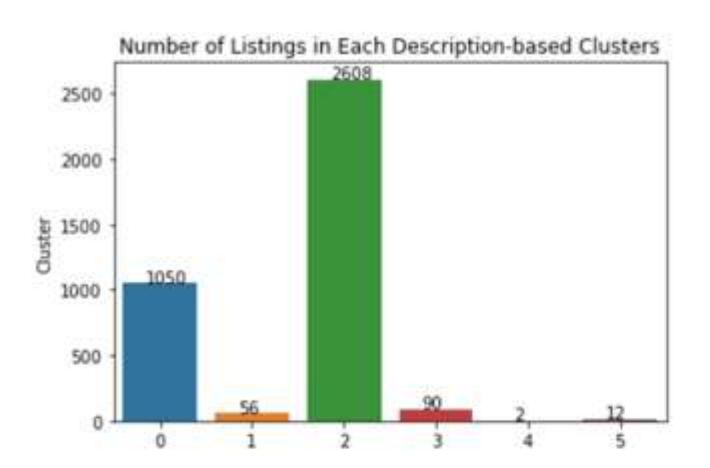
- Set weight for words

```
# Transform combined_description into tfidf format
tfidf = TfidfVectorizer(ngram_range=(1,2),stop_words='english',tokenizer=LemmaTokenizer())
tfidf.fit(df['combined_description'])
DescTfidf = tfidf.transform(df['combined_description'])
```

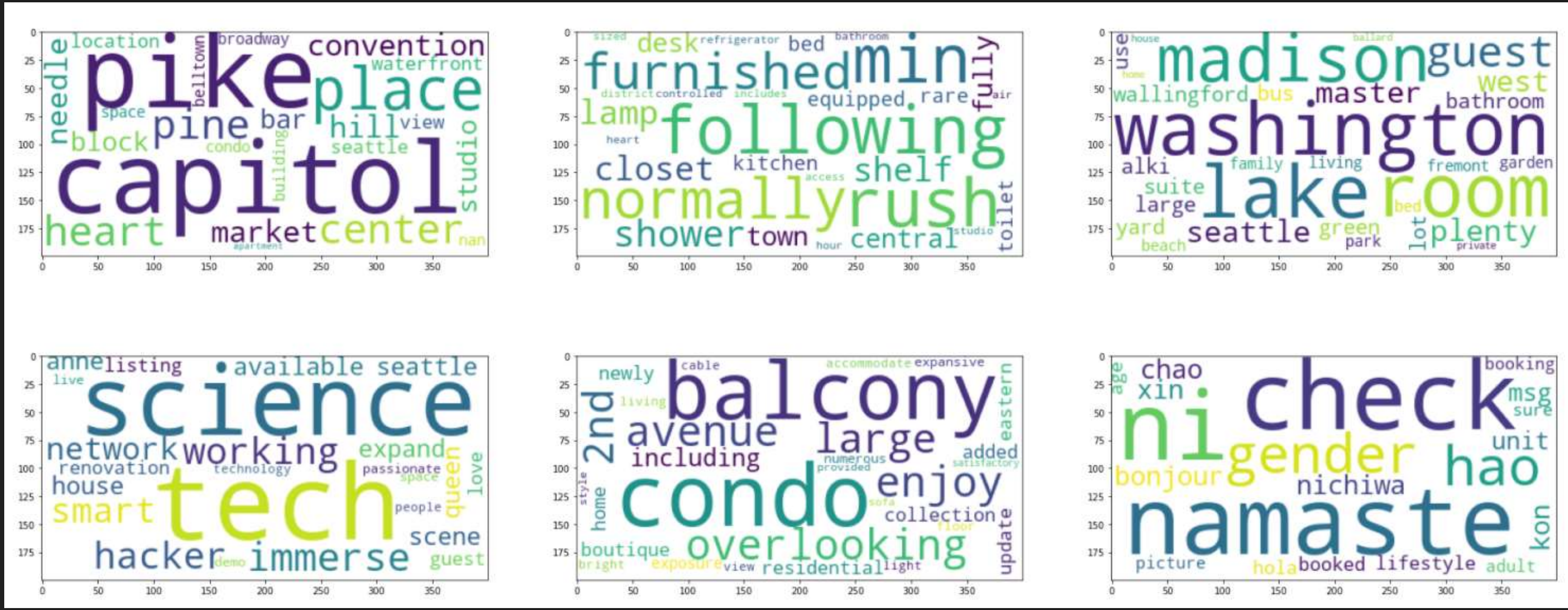
- Keep top 30 words

```
#Top 30 words that describe each cluster
# Pipeline to identify top 30 words that are "best predictor" of a cluster
pipeline = Pipeline([('tfidf', TfidfVectorizer(ngram_range=(1,2), stop_words='english', tokenizer=LemmaTokenizer())),
                     ('clf', SGDClassifier(loss='hinge', penalty='l2',
                                           alpha=1e-3, max_iter=5, random_state=42)),
])
```

- Number of Listings per Cluster



Clusters Overview with wordcloud



Clusters

Cluster 1: «Next to Capitol»

Cluster 2: «Furnished»

Clusters

Cluster 3: «Next to a Lake»

Cluster 4: «Tech House»,

Clusters

Cluster 5: «With a balcony»

Cluster 6: «Chinese»

Prediction

- Predict prices based on

```
df = listings[["host_response_rate", "host_acceptance_rate", "host_is_superhost",
               "host_listings_count", "zipcode", "property_type","room_type", "accommodate
s", "bathrooms", "bedrooms",
               "beds", "price", "number_of_reviews", "review_scores_rating", "cancellation
_policy",
               "reviews_per_month"]]
```

Steps before predicting

Create dummies

```
# select non-numeric variables and create dummies
non_num_vars = df2.select_dtypes(include=['object']).columns
df2[non_num_vars].head()
```

```
# split into test and training data
np.random.seed(1)
indices = np.random.permutation(len(df3))
train_size = int(round(0.8*len(df3)))
test_size = len(df3)-train_size

y = df3['price']
x = df3.drop('price', axis =1)

x.train = x.iloc[indices[0:train_size]]
y.train = y.iloc[indices[0:train_size]]
x.test = x.iloc[indices[train_size+1:]]
y.test = y.iloc[indices[train_size+1:]]

x2 = x.train.as_matrix()
y2 = y.train.as_matrix()
```

Random Forest

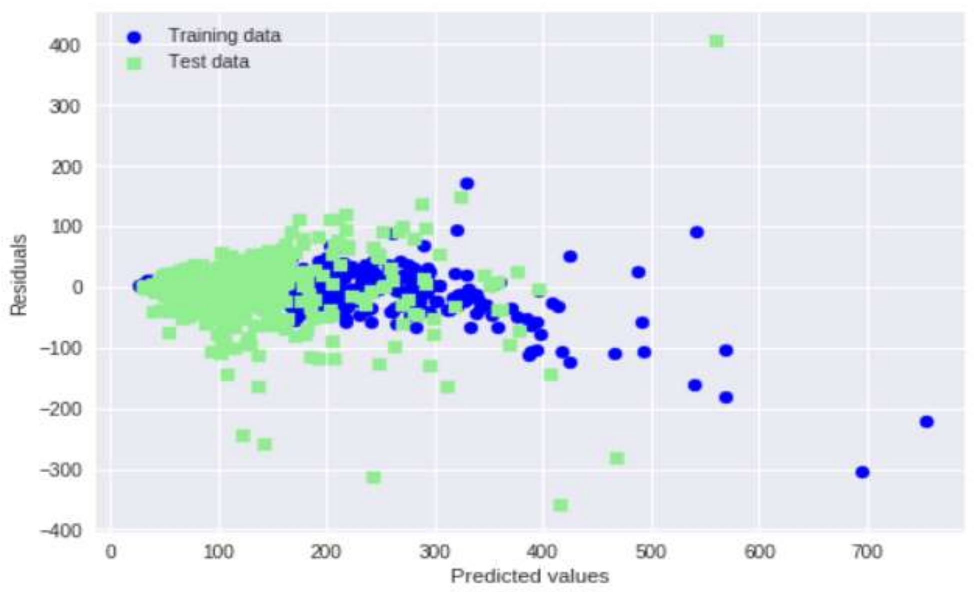
```
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor(n_estimators=500,
                               criterion='mse',
                               random_state=3,
                               n_jobs=-1)
forest.fit(X_train, y_train)
y_train_pred = forest.predict(X_train)
y_test_pred = forest.predict(X_test)

print('MSE train: %.3f, test: %.3f' % (
        mean_squared_error(y_train, y_train_pred),
        mean_squared_error(y_test, y_test_pred)))
print('R^2 train: %.3f, test: %.3f' % (
        r2_score(y_train, y_train_pred),
        r2_score(y_test, y_test_pred)))
```

```
MSE train: 360.229, test: 2275.514
R^2 train: 0.945, test: 0.670
```

Plot results

```
plt.scatter(y_train_pred,  y_train_pred - y_train,
            c='blue', marker='o', label='Training data')
plt.scatter(y_test_pred,  y_test_pred - y_test,
            c='lightgreen', marker='s', label='Test data')
plt.xlabel('Predicted values')
plt.ylabel('Residuals')
plt.legend(loc='upper left')
plt.show()
```



Any Questions?

Thank you