Project Team: Walking in Paris

Objective:

Identify the attributes that make Airbnb “Successful” Find the best place to stay in Hawaii

In [1]:

```
# Dependencies and Setup
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import requests
import time

# Import statistic library
import scipy.stats as stats
from scipy.stats import linregress

# Import seaborn library to create figure
import seaborn as sns

import pprint
import json
from time import sleep
from datetime import date
import zipfile

# Incorporated citipy to determine city based on latitude and longitude
# pip install citipy
from citipy import citipy

# Output File location
output_folder = "output_data"
```

```
Open the dataset from the zip file and ensure memory resources
Cleaning raw data (airbnb) to delete the column with not values, columns
values duplicated, columns with data not required.
```

In [2]:

```
# Open the zip file
zf = zipfile.ZipFile('./source_data/detail_listings.zip')
df = pd.read_csv(zf.open('detail_listings.csv'), low_memory=False)

rows_in_df = len(df)
print(f'Data set has {rows_in_df} rows')

# Drop unneeded columns:
df = df.drop(columns = [
    "listing_url",
    "thumbnail_url",
    "picture_url",
    "medium_url",
    "xl_picture_url",
    "host_url",
    "host_thumbnail_url",
    "host_picture_url",
    "scrape_id",
    "host_listings_count", # There is a host_total_listings_count that is a c
    "host_acceptance_rate", # NaN
    "calendar_last_scraped",
    "bed_type",
    "last_scraped",
    "calendar_updated",
    "has_availability",
    "availability_30",
    "availability_60",
    "availability_90",
    "availability_365",
    "license"
])

df.head(1)
```

```
Data set has 23745 rows
```

Out[2]:

	id	name	summary	space	description	experiences_offered	neighborhood_overview	not
0	5065	MAUKA BB	Perfect for your vacation, Staycation or just ...	Mauka B & B is 400 square feet studio with a p...	Perfect for your vacation, Staycation or just ...	none	Neighbors here are friendly but are not really...	Locat near Kalo Sta Pa w 100

1 rows × 85 columns

In [3]:

```
# Group the columns and show the list of the columns

print('')
print('----------------------')
print('Property-specific columns:')
print('----------------------')
# These are wildcard search terms:
cols = ["experiences_offered", "street", "city", "state", "zipcode", "market"
    , "location", "accomodate", "room", "guest", "is_business_travel_ready",
for col in df.columns:
    if col in cols:
        print(col)

print('')
print('----------------------')
print('All columns:')
print('----------------------')
for col in df.columns:
    print(col)
```

```
----------------------
Property-specific columns:
----------------------
experiences_offered
street
city
state
zipcode
market
country
instant_bookable
is_business_travel_ready

----------------------
All columns:
----------------------
id
name
summary
space
description
experiences_offered
neighborhood_overview
notes
transit
access
interaction
house_rules
host_id
host_name
host_since
host_location
host_about
host_response_time
host_response_rate
```

```
host_is_superhost
host_neighbourhood
host_total_listings_count
host_verifications
host_has_profile_pic
host_identity_verified
street
neighbourhood
neighbourhood_cleansed
neighbourhood_group_cleansed
city
state
zipcode
market
smart_location
country_code
country
latitude
longitude
is_location_exact
property_type
room_type
accommodates
bathrooms
bedrooms
beds
amenities
square_feet
price
weekly_price
monthly_price
security_deposit
cleaning_fee
guests_included
extra_people
minimum_nights
maximum_nights
minimum_minimum_nights
maximum_minimum_nights
minimum_maximum_nights
maximum_maximum_nights
minimum_nights_avg_ntm
maximum_nights_avg_ntm
number_of_reviews
number_of_reviews_ltm
first_review
last_review
review_scores_rating
review_scores_accuracy
review_scores_cleanliness
review_scores_checkin
review_scores_communication
review_scores_location
review_scores_value
requires_license
jurisdiction_names
instant_bookable
```

```
is_business_travel_ready
cancellation_policy
require_guest_profile_picture
require_guest_phone_verification
calculated_host_listings_count
calculated_host_listings_count_entire_homes
calculated_host_listings_count_private_rooms
calculated_host_listings_count_shared_rooms
reviews_per_month
```

In [4]:

```
# Create a dataframe with 1 row for each column in the source:
data_dict = pd.DataFrame(columns = [
    "ColumNumber"
    , "Column"
    , "Group"
    , "Subgroup"
    , "Notes"
])

# Loop through
for column in df:
    data_dict = data_dict.append({
            "Column": column
            , "Group": ""
            , "Subgroup": ""
            , "Notes": ""
        }
        , ignore_index=True
    )

# Make the index the column numebr
data_dict['ColumNumber'] = data_dict.index

# Add context for each column
for index, row in data_dict.iterrows():
    val = str(row["Column"])

    if val == "id":
        data_dict.loc[index, "Notes"] = "Primary Key"
    # Val.startswith("host_")
    if val.find("host") > -1:
        data_dict.loc[index, "Group"] = "Host-related"

    if val.find("review") > -1:
        data_dict.loc[index, "Group"] = "Review-related"

    if (val.find("rule") > -1) | (val.find("require") > -1):
        data_dict.loc[index, "Group"] = "Requirements-related"

    if (val.find("neighbo") > -1):
        data_dict.loc[index, "Group"] = "Neighborhood-related"

    if (val.find("nights") > -1):
        data_dict.loc[index, "Group"] = "Booking-related"

    # Property specific:
    cols = [
        "property", "room", "guest", "is_business_travel_ready", "instant_boo
        , "space", "name", "summary", "description", "notes", "transit", "acc
        , "property_type", "room_type", "amenities"
    ]
    if any(word in val for word in cols):
        data_dict.loc[index, "Group"] = "Property-related"

    # Property - "Location" specific:
    cols = [
```

```
        "location", "street", "city", "state", "zipcode", "market", "country"
        , "latitude", "longitude", "is_location_exact"
    ]
    if any(word in val for word in cols):
        data_dict.loc[index, "Group"] = "Property-related"
        data_dict.loc[index, "Subgroup"] = "Location"

    # Property - "Interior" specific:
    cols = [
        "bathrooms", "bedrooms", "beds", "square_feet"
    ]
    if any(word in val for word in cols):
        data_dict.loc[index, "Group"] = "Property-related"
        data_dict.loc[index, "Subgroup"] = "Interior"

    # Property - "Pricing" specific:
    # Things about price, or that we might associate with price
    cols = [
        "price", "weekly_price", "monthly_price"
        , "security_deposit", "cleaning_fee"
        , "accomodates", "guests_included", "extra_people"
        , "experiences_offered"
    ]
    if any(word in val for word in cols):
        data_dict.loc[index, "Group"] = "Property-related"
        data_dict.loc[index, "Subgroup"] = "Pricing"

    # Add additional code to accomodate a group and subgroup of properties:
    if val == "accommodates":
        data_dict.loc[index, "Group"] = "Property-related"
        data_dict.loc[index, "Subgroup"] = "Pricing"

print(data_dict.groupby(["Group", "Subgroup"])["Column"].count())
print('')
print('Columns without a group:')
data_dict[data_dict["Group"] == ""]

```

```
Group                  Subgroup
                                    1
Booking-related                     8
Host-related                       12
Neighborhood-related                5
Property-related                   20
                       Interior     4
                       Location    13
                       Pricing      9
Requirements-related                2
Review-related                     11
Name: Column, dtype: int64

Columns without a group:
```

Out[4]:

	ColumNumber	Column	Group	Subgroup	Notes
0	0	id			Primary Key

Making additional cleaning of the raw data in the rows

- Drop rows with number_of_reviews = 0
- Get names of indexes for which column number_of_reviews has value 0
- Strip out the dollar symboil ($) from the pricing based columns: price","weekly_price","monthly_price","security_deposit","cleaning_fee", "extra_people"

In [5]:

```
# Drop by index:
indexNames = df[df['number_of_reviews'] == 0 ].index
df.drop(indexNames , inplace=True)

# Drop rows with number_of_reviews = NaN
df.dropna(axis=0, subset=('number_of_reviews', ))

# Drop rows with NaN review data
df.dropna(axis=0, subset=(
    'review_scores_rating',
    'review_scores_accuracy',
    'review_scores_cleanliness',
    'review_scores_checkin',
    'review_scores_communication',
    'review_scores_location',
    'review_scores_value'
))

print(f'Remaining rows after cleanup: {len(df)}')
```

```
Remaining rows after cleanup: 18291
```

In [6]:

```
# Strip the $ out of the pricing-based columns
data_dict[(data_dict["Group"] == "Property-related") & (data_dict["Subgroup"]

if df["price"].dtype != "float64":
    df["price"] = df["price"].str.replace("$", "")
    df["price"] = df["price"].str.replace(",", "")
    df["price"] = df["price"].astype("float64")

if df["weekly_price"].dtype != "float64":
    df["weekly_price"] = df["weekly_price"].str.replace("$", "")
    df["weekly_price"] = df["weekly_price"].str.replace(",", "")
    df["weekly_price"] = df["weekly_price"].astype("float64")

if df["monthly_price"].dtype != "float64":
    df["monthly_price"] = df["monthly_price"].str.replace("$", "")
    df["monthly_price"] = df["monthly_price"].str.replace(",", "")
    df["monthly_price"] = df["monthly_price"].astype("float64")

if df["security_deposit"].dtype != "float64":
    df["security_deposit"] = df["security_deposit"].str.replace("$", "")
    df["security_deposit"] = df["security_deposit"].str.replace(",", "")
    df["security_deposit"] = df["security_deposit"].astype("float64")

if df["cleaning_fee"].dtype != "float64":
    df["cleaning_fee"] = df["cleaning_fee"].str.replace("$", "")
    df["cleaning_fee"] = df["cleaning_fee"].str.replace(",", "")
    df["cleaning_fee"] = df["cleaning_fee"].astype("float64")

if df["extra_people"].dtype != "float64":
    df["extra_people"] = df["extra_people"].str.replace("$", "")
    df["extra_people"] = df["extra_people"].str.replace(",", "")
    df["extra_people"] = df["extra_people"].astype("float64")

data_dict
```

Out[6]:

	ColumNumber	Column	Group	Subgroup	Notes
0	0	id			Primary Key
1	1	name	Property-related		
2	2	summary	Property-related		
3	3	space	Property-related		
4	4	description	Property-related		
...	...	...	...	...	...
80	80	calculated_host_listings_count	Host-related		
81	81	calculated_host_listings_count_entire_homes	Host-related		
82	82	calculated_host_listings_count_private_rooms	Property-related		

	ColumNumber	Column	Group	Subgroup	Notes
83	83	calculated_host_listings_count_shared_rooms	Property-related		
84	84	reviews_per_month	Review-related		

85 rows × 5 columns

```
# Create the bins to separate the number of properties by host

bins = [1, 60, 119, 179, 239, 299]

# labels - always one less than the range
group_names = ["0-59 properties", "60-119 properties", "120-179 properties",
"180-239 properties", "240+ properties"]

df["Bin_NumProperties"] = pd.cut(df["host_total_listings_count"], bins,
labels=group_names)

df_analyze = df[["host_id", "Bin_NumProperties", "host_total_listings_count"]]
df_analyze = df_analyze.rename(columns={"host_total_listings_count":
"listings"})

# Loop through and get aggs:
for index, row in df_analyze.iterrows():
    df_analyze.loc[index, 'reviews'] = df_analyze["number_of_reviews"].sum()

#df_analyze["reviews"] =
df_analyze["rating_mean"] = df_analyze["review_scores_rating"].mean()
df_analyze["rating_value_mean"] = df_analyze["review_scores_value"].mean()
df_analyze.head()
```

In [7]:

```
1
```

In [8]:

```
# Create the bins to separate the number of properties by host
bins = [0, 50, 100, 150, 200, 250, 299]

# labels - always one less than the range
group_names = ["0-50 properties", "51-100 properties", "101-150 properties",
               "201-250 properties", "251+ properties"]

df["Bin_NumProperties"] = pd.cut(df["calculated_host_listings_count"], bins,


# Loop through and get aggs:
total_reviews_per_host = df.groupby("host_id").sum()["number_of_reviews"]
average_review_per_host = df.groupby("host_id").mean()["review_scores_rating"
rating_mean_by_bin = df.groupby("Bin_NumProperties").mean()["review_scores_ra

#df_analyze["rating_value_mean"] = df_analyze["review_scores_value"].mean()
df.head(1)
```

Out[8]:

	id	name	summary	space	description	experiences_offered	neighborhood_overview	not
0	5065	MAUKA BB	Perfect for your vacation, Staycation or just ...	Mauka B & B is 400 square feet studio with a p...	Perfect for your vacation, Staycation or just ...	none	Neighbors here are friendly but are not really...	Locat near Kalo Sta Pa w 100

1 rows × 86 columns

```
1
```

In [10]:

```
# data_dict[data_dict["Group"] == "Host-related"]
host_group = df.groupby("neighbourhood").agg({
    "number_of_reviews": [
        np.count_nonzero,
        np.mean,
        np.median,
        np.var,
        np.std
    ]
})

host_group.rename(columns={'count_nonzero':'Reviews'})
host_group.head()
# bin on reviews_per_month
```

Out[10]:

	number_of_reviews				
	count_nonzero	mean	median	var	std
neighbourhood					
Aiea	14	40.428571	8.0	2872.417582	53.594940
Airport	3	4.000000	4.0	9.000000	3.000000
Ala Moana/Kakaako	28	23.214286	8.5	1138.767196	33.745625
Aliamanu/Salt Lake/Foster Village	8	74.875000	9.5	17739.267857	133.188843
Central Oahu	43	29.302326	17.0	1214.073090	34.843552

In [11]:

```
bins_num_reviews = [10, 20, 50, 100, 200, 300, 350, 400, 450, 500, 550, 600,

# labels - always one less than the range
group_names_num_reviews = [
    "1-10"
    , "11-20"
    , "21-50"
    , "100-200"
    , "201-250"
    , "251-300"
    , "301-350"
    , "351-400"
    , "401-550"
    , "451-500"
    , "501-550"
    , "551-600"
]

df["Bin_NumReviews"] = pd.cut(
    df["number_of_reviews"],
    bins = bins_num_reviews,
    labels = group_names_num_reviews
)

df["Bin_NumReviews"].value_counts()

```

Out[11]:

```
11-20      3790
1-10       2526
21-50      2368
100-200    1235
201-250     250
251-300      32
301-350      23
351-400      10
401-550       6
551-600       3
451-500       3
501-550       1
Name: Bin_NumReviews, dtype: int64
```

In [12]:

```
print(f'Sum: {df["number_of_reviews"].sum()}')
print(f'Count: {df["number_of_reviews"].count()}')

df.groupby("Bin_NumReviews")["number_of_reviews"].sum()
```

```
Sum: 623494
Count: 18291
```

Out[12]:

```
Bin_NumReviews
1-10        37991
11-20      126389
21-50      169320
100-200    169224
201-250     59598
251-300     10221
301-350      8452
351-400      4217
401-550      2864
451-500      1561
501-550       569
551-600      1983
Name: number_of_reviews, dtype: int64
```

In [13]:
```
df.groupby("Bin_NumReviews").agg({
    "number_of_reviews": [
        np.count_nonzero,
        np.mean,
        np.median,
        np.var,
        np.std
    ]
})
```

Out[13]:

	number_of_reviews				
	count_nonzero	mean	median	var	std
Bin_NumReviews					
1-10	2526	15.039984	15.0	8.161173	2.856777
11-20	3790	33.348021	32.0	74.548947	8.634173
21-50	2368	71.503378	70.0	197.305439	14.046545
100-200	1235	137.023482	131.0	741.095883	27.223076
201-250	250	238.392000	232.0	802.456161	28.327657
251-300	32	319.406250	318.0	176.958669	13.302581
301-350	23	367.478261	366.0	157.988142	12.569333
351-400	10	421.700000	421.5	252.900000	15.902830
401-550	6	477.333333	485.5	246.666667	15.705625
451-500	3	520.333333	518.0	24.333333	4.932883
501-550	1	569.000000	569.0	NaN	NaN
551-600	3	661.000000	669.0	577.000000	24.020824

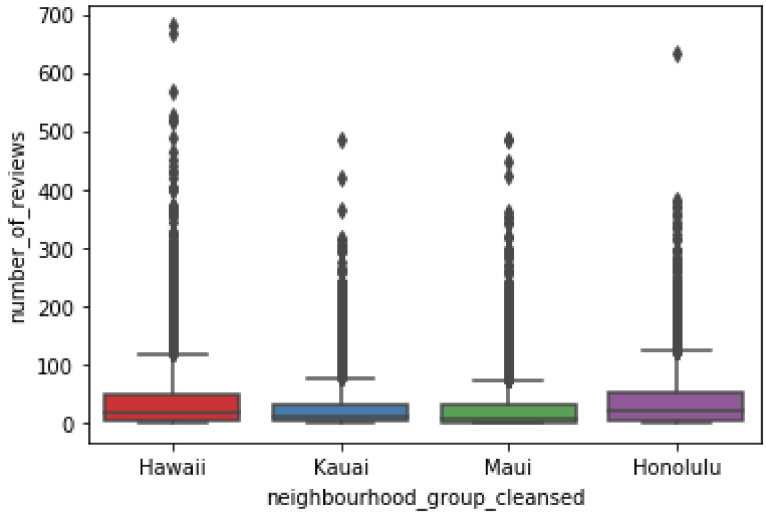
```
Hypothesis candidates:
 Neighborhood
     - Is rating related to price within a neighborhood?
     - The highest price properties in a neighboorhood will have lower ratings than the lowest price properties?
     - Best neighborhoods by rating
     - Best neighborhoods by price
     - Listing types by neighborhood
     - Listing types by price
     - Listing types by rating
 By island:
     - Is one island more expensive than the others?
```

In [ ]:
```
START QUESTIONS CODE PER TEAM MEMBER
```

```
Scott:

```

In [14]:
```
# Use seaborn library to create a boxplot
# sns.boxplot(x="day", y="total_bill", hue="smoker", data=df, palette="Set1")
sns.boxplot(
    x = "neighbourhood_group_cleansed",
    y = "number_of_reviews",
    #hue = "price",
    data = df,
    palette = "Set1"
)

# Bin by island
# data_dict[(data_dict["Group"] == "Property-related") & (data_dict["Subgroup
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x2519f4722c8>



In [15]:
```
from pandas.api.types import is_string_dtype
from pandas.api.types import is_numeric_dtype

is_string_dtype(df['price'])
is_numeric_dtype(df['price'])
df["price"]
```

Out[15]:
```
0         85.0
1        110.0
2         92.0
3        299.0
4         92.0
         ...
23569    111.0
23576     79.0
23582     70.0
23656    146.0
23741    499.0
Name: price, Length: 18291, dtype: float64
```

In [16]:
```
# Bin by island
isle = df[['latitude', 'longitude']].head(50).copy()
isle["Island"] = ""

curr_rows = len(isle)
print(f'There are {curr_rows} coordinates in the dataframe')

df["neighbourhood_group_cleansed"]
```

```
There are 50 coordinates in the dataframe
```

Out[16]:
```
0          Hawaii
1          Hawaii
2          Hawaii
3          Hawaii
4           Kauai
           ...
23569        Maui
23576      Hawaii
23582    Honolulu
23656    Honolulu
23741       Kauai
Name: neighbourhood_group_cleansed, Length: 18291, dtype: object
```

Below we are testing the hypothesis 1: If a host has a hugh number of properties then their ratings are not higher than hosts with fewer properties

```
Meg Section
Evaluate the relationship between "age of property" and review_scores_rating
Evaluate the relationship between "price" and review_scores_rating
Evaluate the relationship between "how many properties someone has" and review_scores_rating
Evaluate the relationship between "number of bedrooms" and review_scores_rating

Observations:
```

In [17]:

```
# Define the bins by number of properties by host
bins = [0, 50, 100, 150, 200, 250, 299]

# labels - always one less than the range
group_names = ["0-50 properties", "51-100 properties", "101-150 properties",
               "201-250 properties", "251+ properties"]

df["Bin_NumProperties"] = pd.cut(df["calculated_host_listings_count"], bins,

rating_mean_by_bin = df.groupby("Bin_NumProperties").mean()["review_scores_ra

df.head(2)
```

Out[17]:

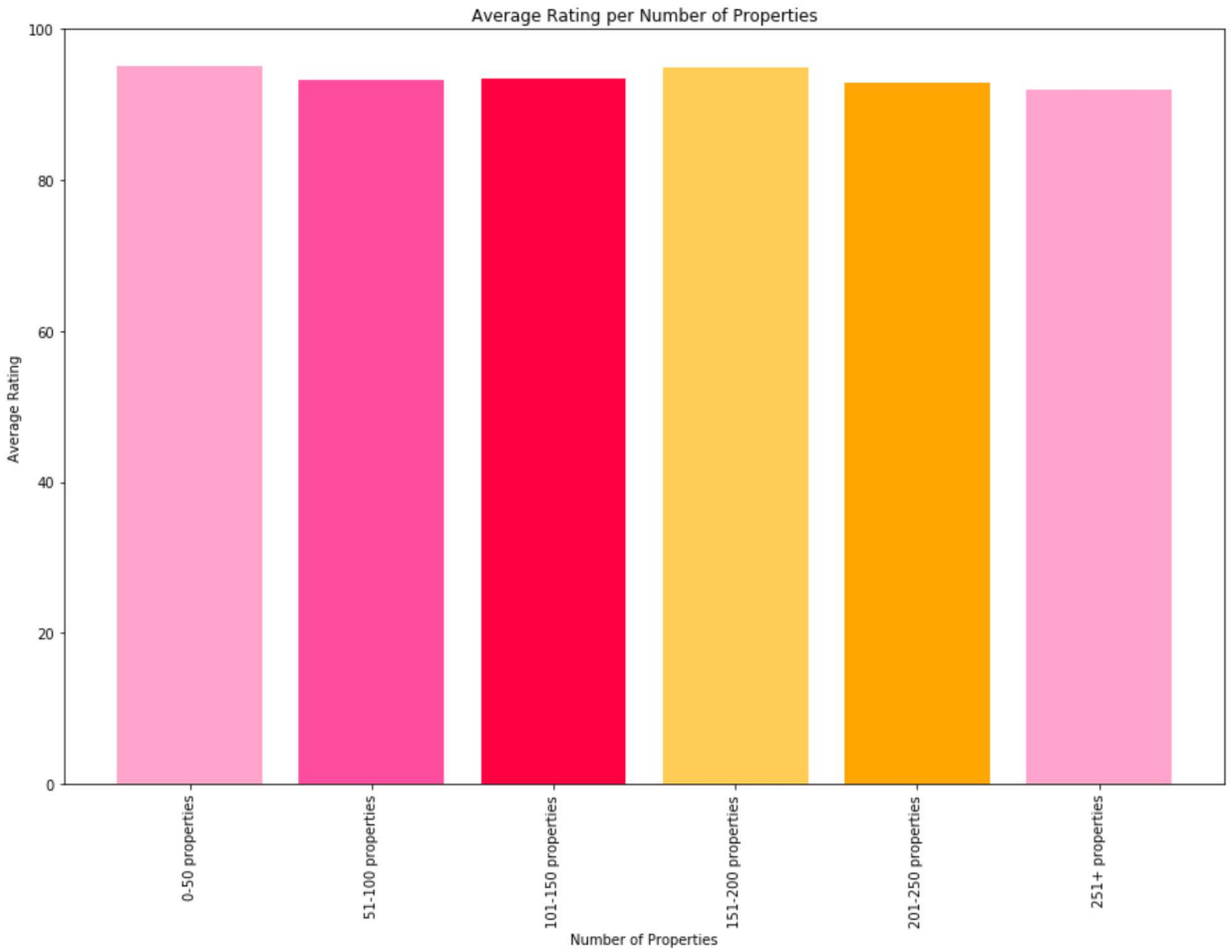
	id	name	summary	space	description	experiences_offered	neighborhood_overview
0	5065	MAUKA BB	Perfect for your vacation, Staycation or just ...	Mauka B & B is 400 square feet studio with a p...	Perfect for your vacation, Staycation or just ...	none	Neighbors here are friendly but are not really...
1	5269	Upcountry Hospitality in the 'Auwai Suite	The 'Auwai Suite is a lovely, self-contained a...	The ‘Auwai Suite is a lovely, self-contained a...	The 'Auwai Suite is a lovely, self-contained a...	none	We are located on the "sunny side" of Waimea, ...

2 rows × 87 columns

In [18]:

```
# Define the axis
x_axis = group_names
y_axis = rating_mean_by_bin

# Bring in the colors from Hawaii
hawaiian_hibiscus_hex_palette = ["#ffa4cd", "#ff4e9e", "#ff0040", "#ffcd55",

# Define the properties of the figure

plt.subplots(figsize=(15,10))
plt.title("Average Rating per Number of Properties")
plt.xlabel("Number of Properties")
plt.ylabel("Average Rating")
plt.xticks(rotation=90)
plt.ylim(0, 100)
plt.bar(x_axis, y_axis, color = hawaiian_hibiscus_hex_palette)

# Show the figure
plt.show
```

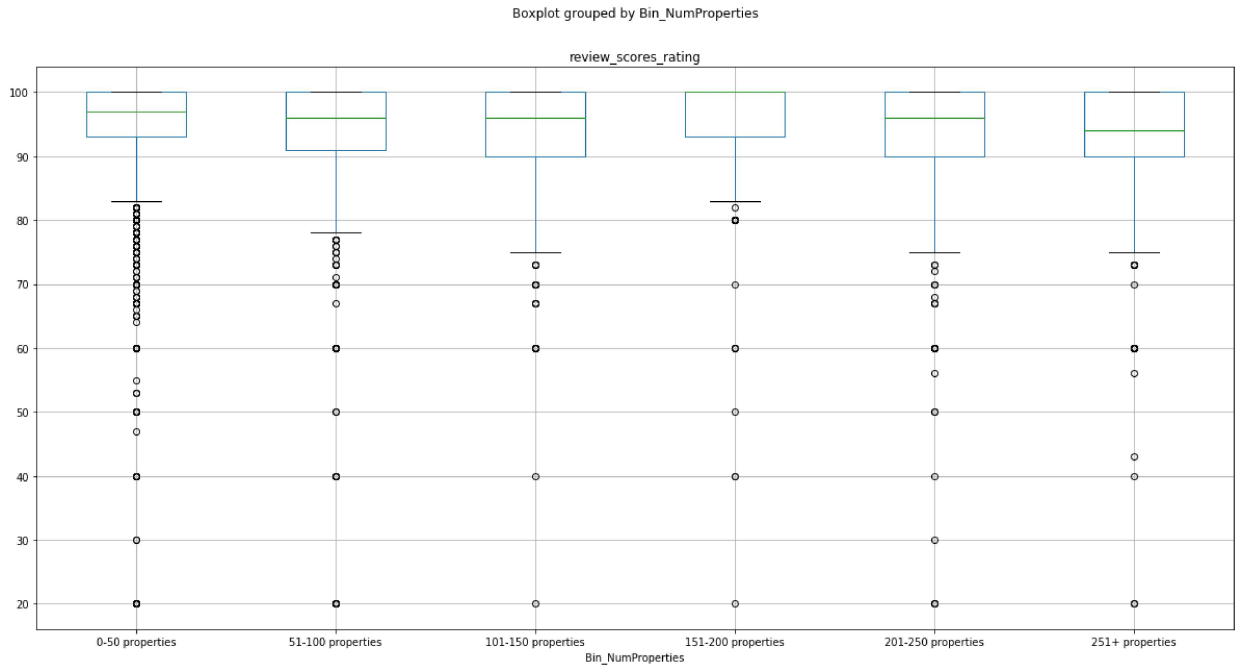
Out[18]: <function matplotlib.pyplot.show(*args, **kw)>



```
1 #ask Meg the objective of the figure below
```

In [19]:
```
1 df.boxplot("review_scores_rating", by="Bin_NumProperties", figsize=(20, 10))
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x251a2053788>



To run the next code, we have "import scipy.stats as stats". Next we are creating a set of group by Number of properties and review score rating

In [20]:

```
# Extract individual groups
group1 = df[df["Bin_NumProperties"] == "0-50 properties"]["review_scores_rati
group2 = df[df["Bin_NumProperties"] == "51-100 properties"]["review_scores_ra
group3 = df[df["Bin_NumProperties"] == "101-150 properties"]["review_scores_r
group4 = df[df["Bin_NumProperties"] == "151-200 properties"]["review_scores_r
group5 = df[df["Bin_NumProperties"] == "201-250 properties"]["review_scores_r
group6 = df[df["Bin_NumProperties"] == "251+ properties"]["review_scores_rati

stats.f_oneway(group1, group2, group3, group4, group5, group6)
```

Out[20]:

```
F_onewayResult(statistic=43.20051018106537, pvalue=1.9870577727779467e-44)
```

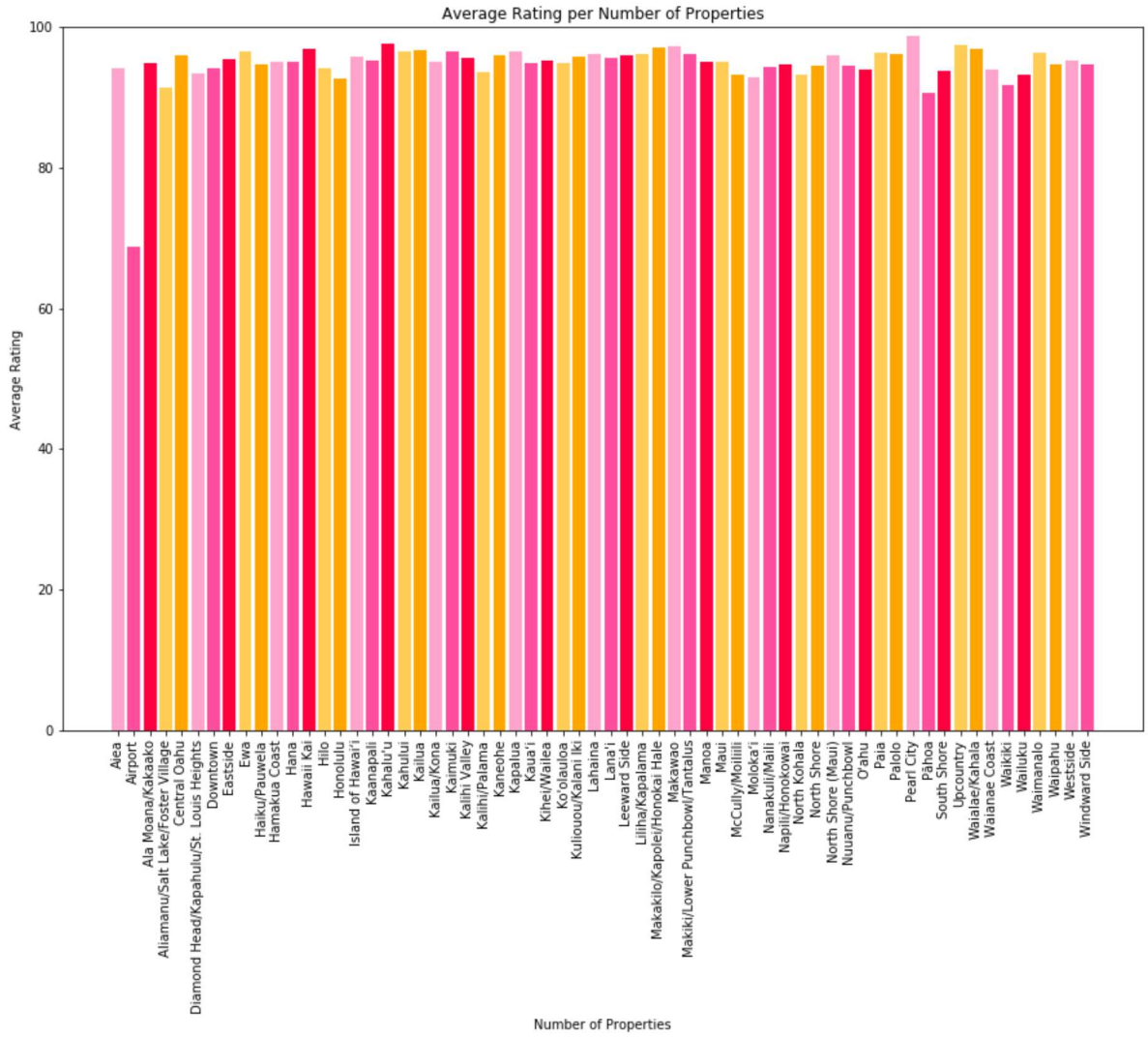
Define neighbourhood

In [21]:

```
neighborhood_list = df["neighbourhood"].unique()

average_rating_per_neighborhood = df.groupby("neighbourhood").mean()["review_
average_rating_per_neighborhood

x_axis = neighborhood_list
y_axis = average_rating_per_neighborhood

cleaned_neighborhood_list = [x for x in neighborhood_list if str(x) != 'nan']
cleaned_neighborhood_list.sort()
cleaned_neighborhood_list

x_axis = cleaned_neighborhood_list
y_axis = average_rating_per_neighborhood

hawaiian_hibiscus_hex_palette = ["#ffa4cd", "#ff4e9e", "#ff0040", "#ffcd55",

plt.subplots(figsize=(15,10))
plt.xticks(rotation=90)
plt.title("Average Rating per Number of Properties")
plt.xlabel("Number of Properties")
plt.ylabel("Average Rating")
plt.ylim(0, 100)
plt.bar(x_axis, y_axis, color = hawaiian_hibiscus_hex_palette, align='center'
plt.show
```

Out[21]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```

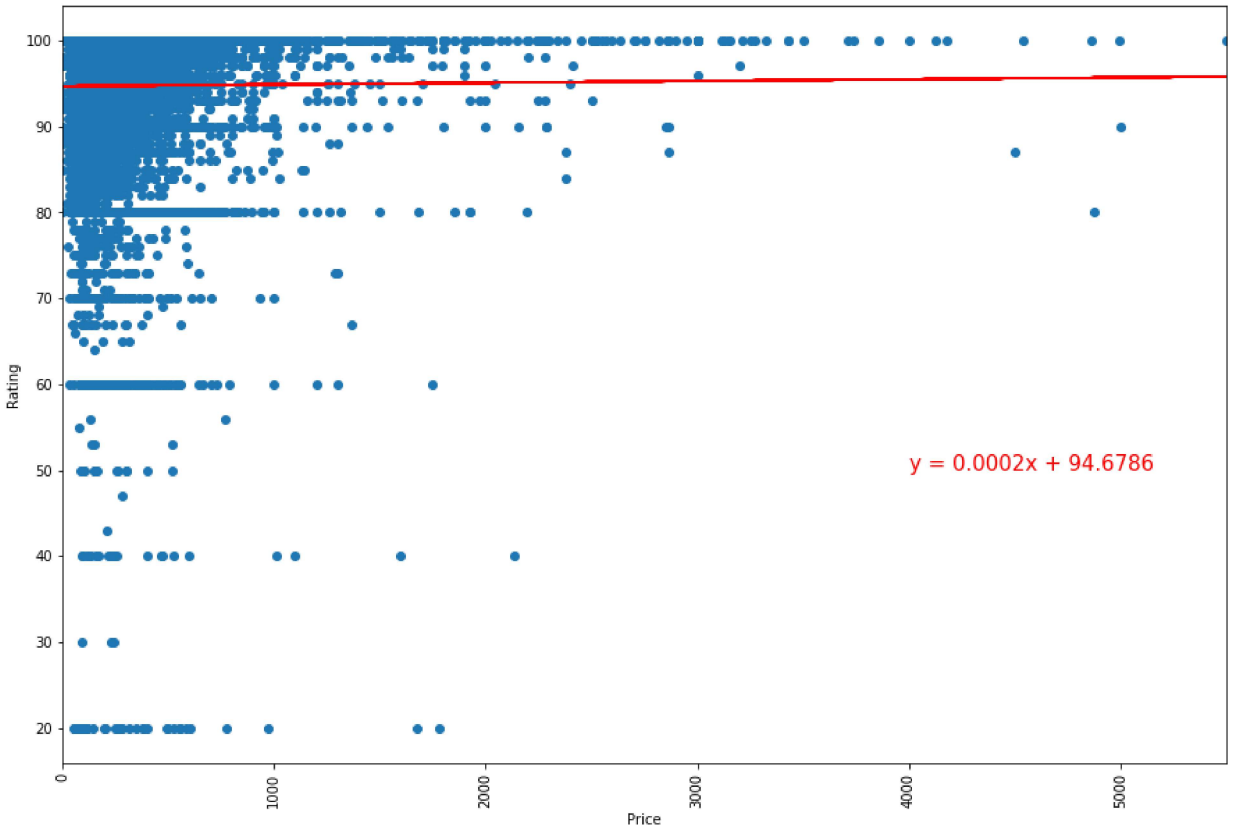



```
1 write notes here for the code below
```

In [22]:

```
# Define axis
x_axis = df["price"]
y_axis = df["review_scores_rating"]


mask = ~np.isnan(x_axis) & ~np.isnan(y_axis)
slope, intercept, r_value, p_value, std_err = stats.linregress(x_axis[mask],
regress_values = x_axis * slope + intercept
line_eq = "y = " + str(round(slope,4)) + "x + " + str(round(intercept,4))

plt.subplots(figsize=(15,10))
plt.xticks(rotation=90)
plt.xlim(0, 5500)
plt.plot(x_axis,regress_values,"r-")
plt.scatter(x_axis,y_axis)
plt.annotate(line_eq,(4000,50),fontsize=15,color="red")
plt.xlabel('Price')
plt.ylabel('Rating')
print(f"The r-squared is: {r_value}")
plt.show()
```

```
The r-squared is: 0.018787686050668887
```



Armon Code Below: Evaluate the relationship between "number of bedrooms" and review_scores_rating.

Observations:

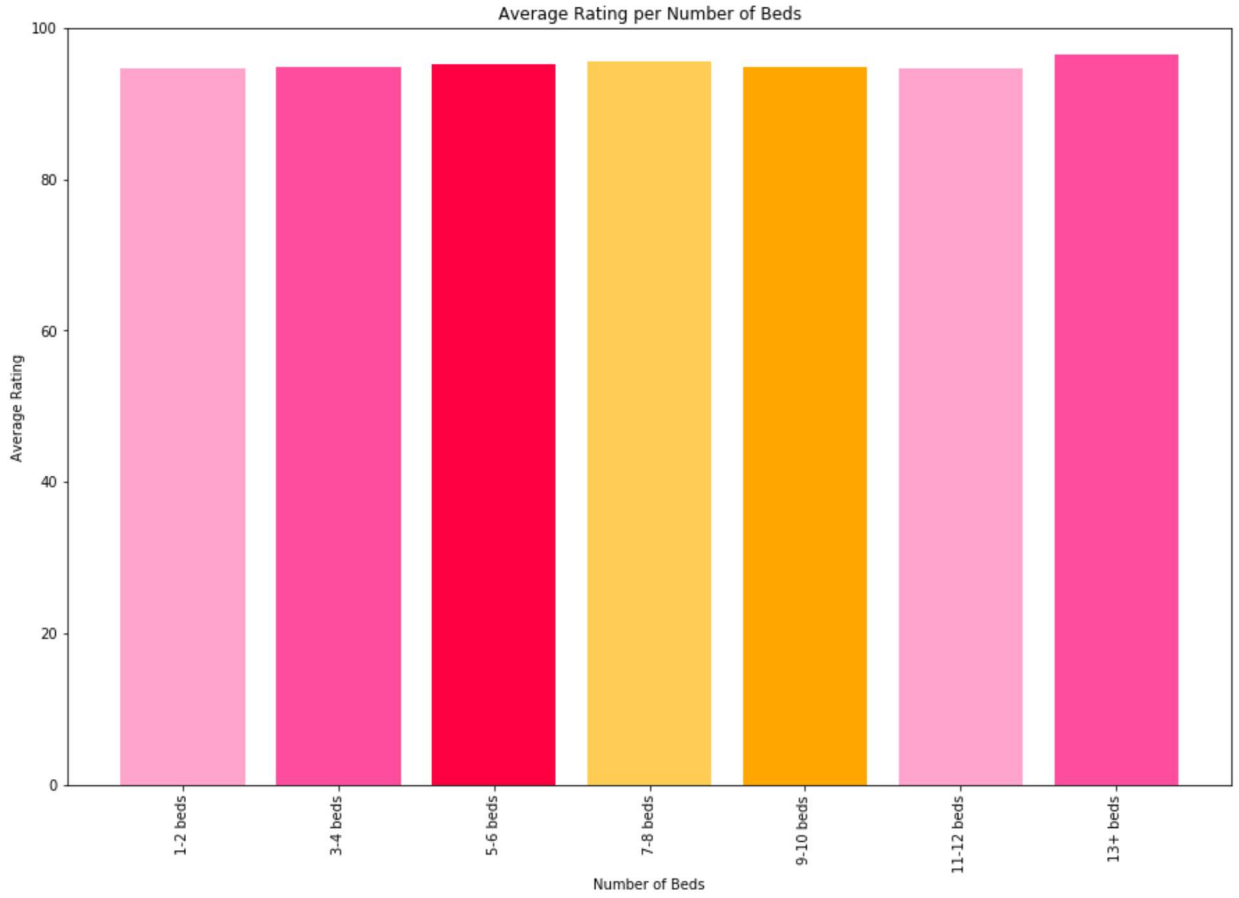
In [24]:
```
# create a dataframe with no 0's for 'number of beds' becuase that would be i
df2 = df.copy()
df2 = df2[(df2['beds'] != 0)]
```

In [25]:
```
# zero beds is actually an option. A 'sofa bed' is not a real bed and is not
bins = [0, 2, 4, 6, 8, 10, 12, 20]

group_names = ["1-2 beds", "3-4 beds", "5-6 beds", "7-8 beds", "9-10 beds", '

df2["bin_bed_count"] = pd.cut(df2["beds"], bins, labels=group_names)

rating_mean_by_bed_count = df2.groupby("bin_bed_count").mean()["review_scores
```

In [26]:

```
x_axis = group_names
y_axis = rating_mean_by_bed_count

hawaiian_hex_palette = ["#00739f", "#b23948", "#ff000e", "#ff407c", "#ffb2e1"

plt.subplots(figsize=(15,10))
plt.title("Average Rating per Number of Beds")
plt.xlabel("Number of Beds")
plt.ylabel("Average Rating")
plt.xticks(rotation=90)
plt.ylim(0, 100)
plt.bar(x_axis, y_axis, color = hawaiian_hibiscus_hex_palette)
plt.show
```

Out[26]: <function matplotlib.pyplot.show(*args, **kw)>



Lance Code Below: Evaluate the relationship between "Type of place" and review_scores_rating
Evaluate the relationship between "Island" and review_scores_rating

In [27]:

```
# Define a group by every rating properties
average_rating_prop = df.groupby("property_type").mean()["review_scores_ratir
#property_type = df.groupby("property_type").count()["id"]

# dropped NaN values for property types that do not have ratings in hawaii su
average_rating_prop = average_rating_prop.dropna()
average_rating_prop
```

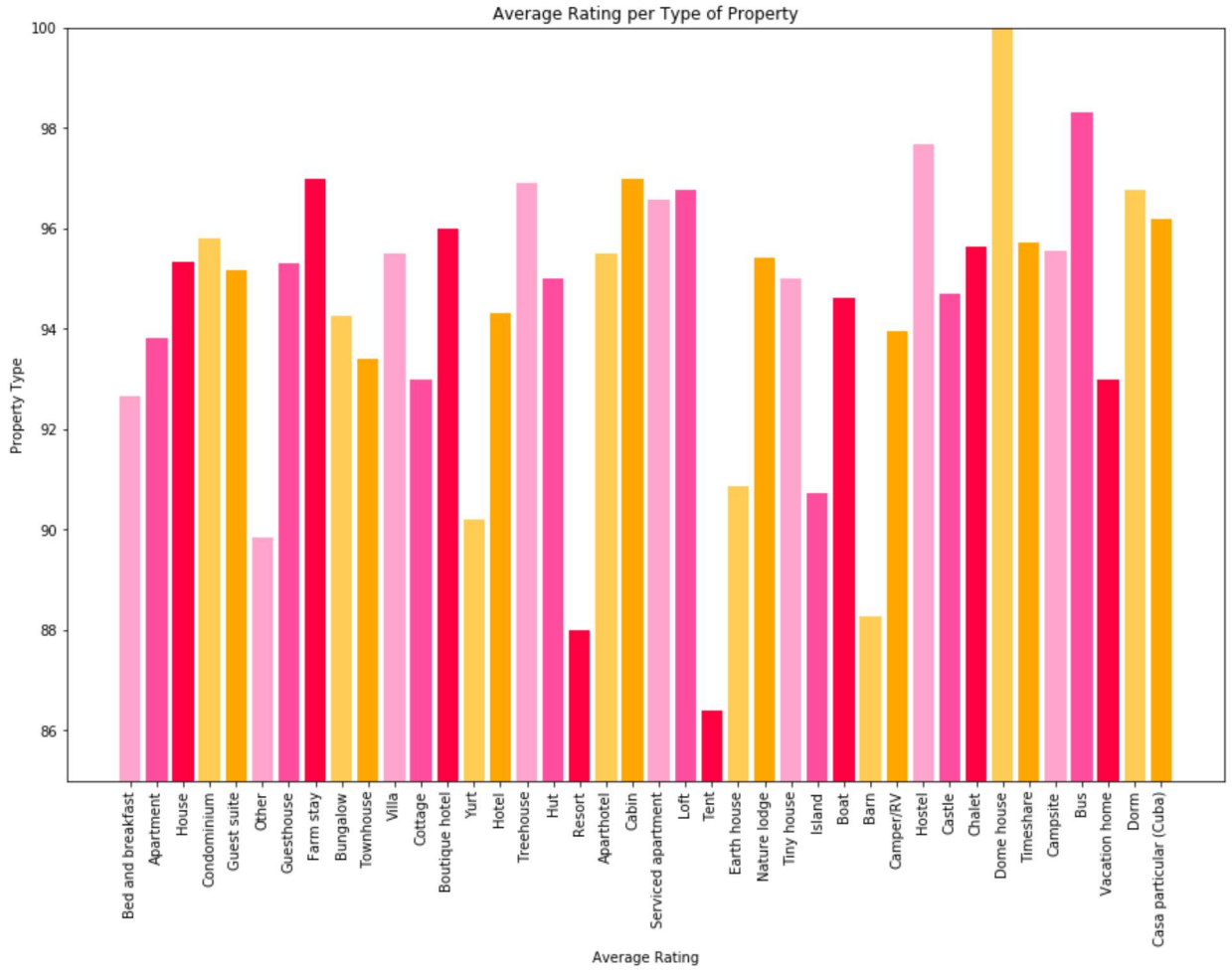
Out[27]:

```
property_type
Aparthotel                  92.649123
Apartment                   93.820818
Barn                        95.333333
Bed and breakfast           95.806061
Boat                        95.166667
Boutique hotel              89.837209
Bungalow                    95.305732
Bus                         97.000000
Cabin                       94.255814
Camper/RV                   93.411765
Campsite                    95.500000
Casa particular (Cuba)      93.000000
Castle                      96.000000
Chalet                      90.200000
Condominium                 94.314025
Cottage                     96.919094
Dome house                  95.000000
Dorm                        88.000000
Earth house                 95.500000
Farm stay                   96.988636
Guest suite                 96.588076
Guesthouse                  96.773270
Hostel                      86.384615
Hotel                       90.876923
House                       95.415111
Hut                         95.000000
Island                      90.714286
Loft                        94.625000
Nature lodge                88.260870
Other                       93.945205
Resort                      97.673684
Serviced apartment          94.702564
Tent                        95.631579
Timeshare                  100.000000
Tiny house                  95.714286
Townhouse                   95.563474
Treehouse                   98.307692
Vacation home               93.000000
Villa                       96.764085
Yurt                        96.200000
Name: review_scores_rating, dtype: float64
```

In [29]:

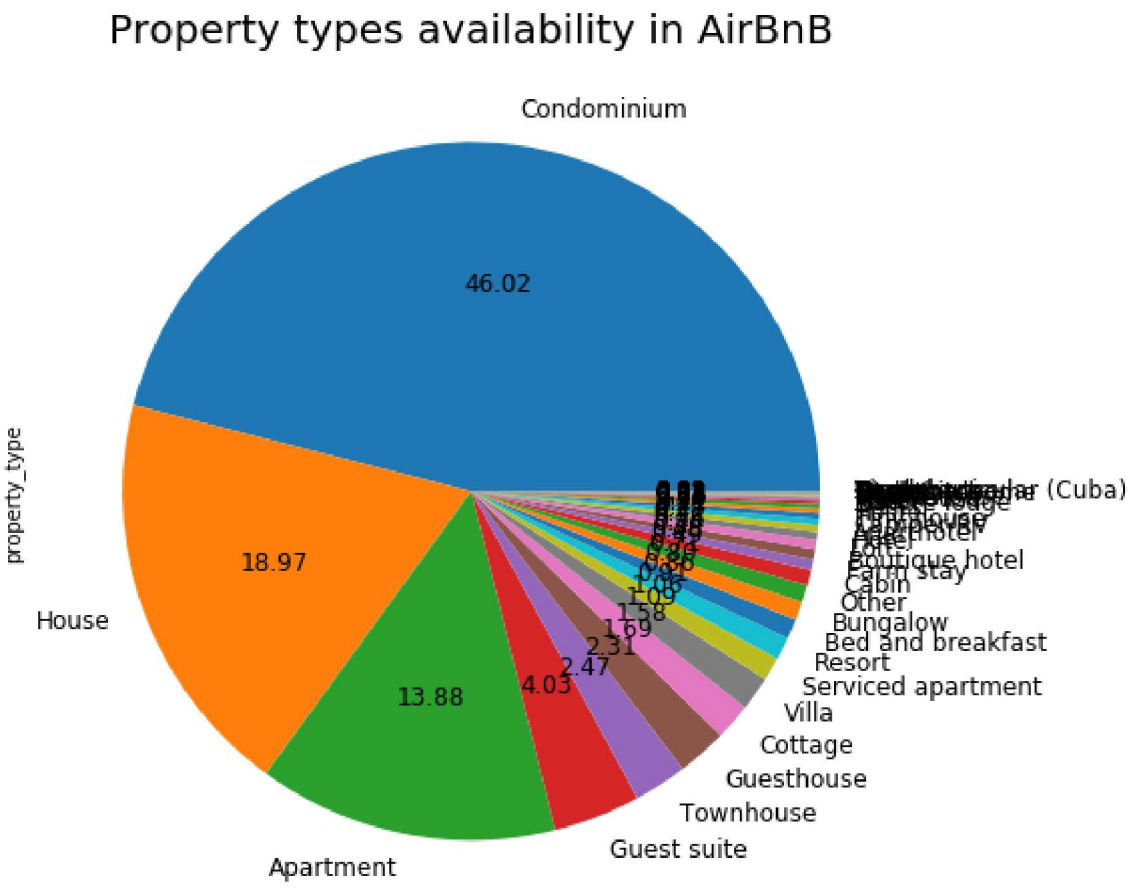
```
# set x and y axis for bar graph
x_axis = df["property_type"].unique()
y_axis = average_rating_prop

# print lables/title and x/y limits to improve visual
plt.subplots(figsize=(15,10))
plt.title("Average Rating per Type of Property")
plt.xlabel("Average Rating")
plt.ylabel("Property Type")
plt.xticks(rotation=90)
plt.ylim(85, 100)

# display graph
plt.bar(x_axis, y_axis, color = hawaiian_hibiscus_hex_palette)
plt.show
```

Out[29]: <function matplotlib.pyplot.show(*args, **kw)>



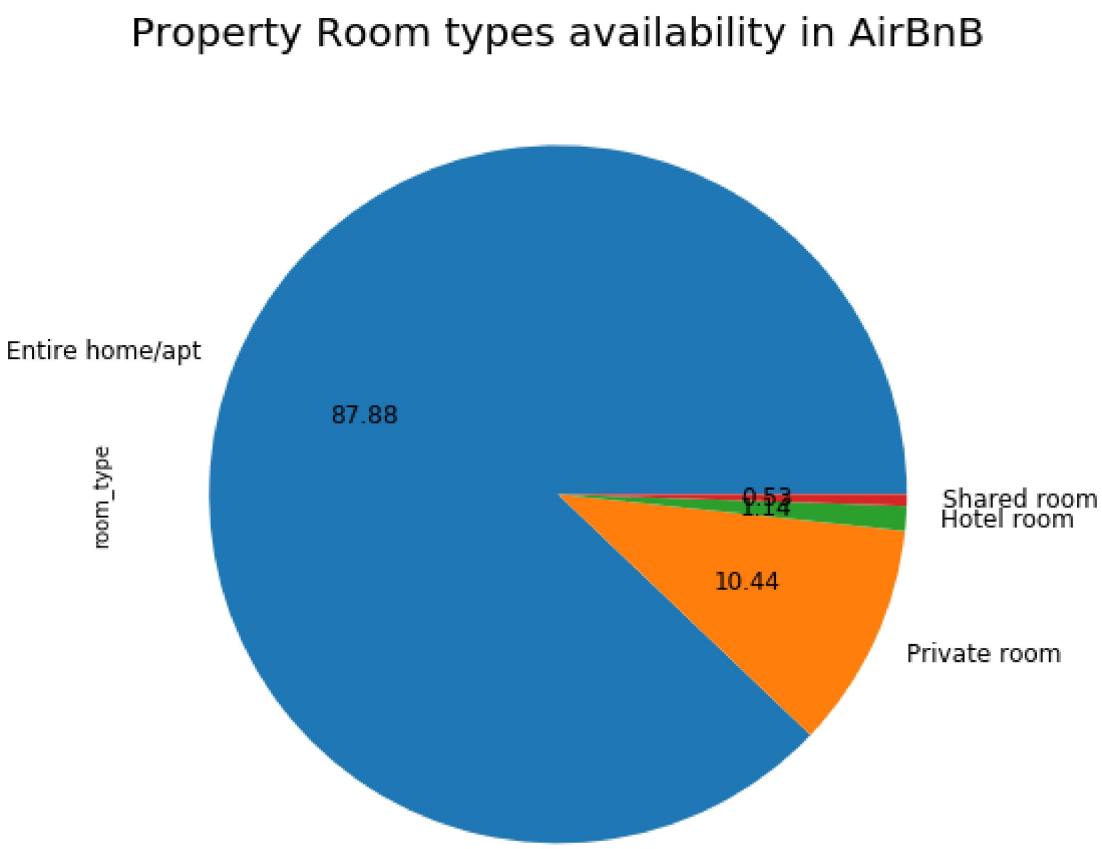
In [34]:
```
# Build a pie to show the availability by property type in Airbnb Hawaii.
prop_type_pie=df.property_type.value_counts()/len(df.property_type)
prop_type_pie.plot.pie(autopct='%.2f',fontsize=12,figsize=(8,8))
plt.title('Property types availability in AirBnB',fontsize=20)
```

Out[34]: Text(0.5, 1.0, 'Property types availability in AirBnB')



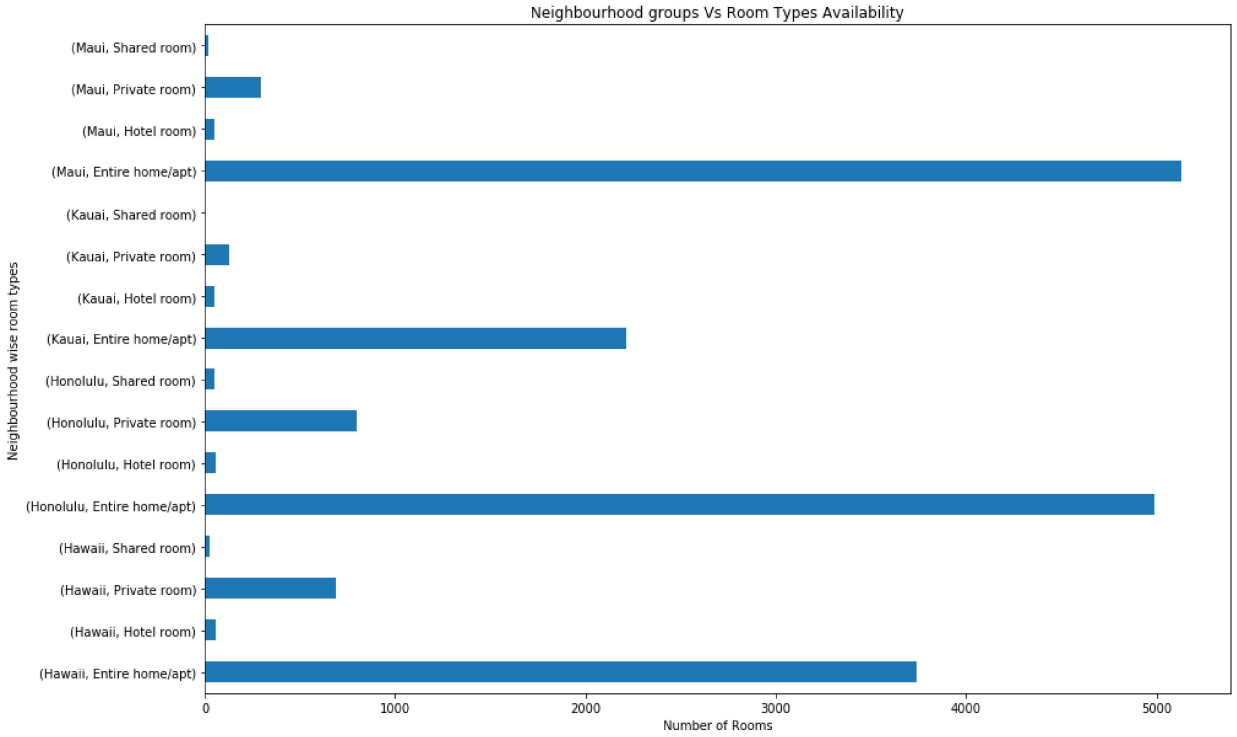
In [40]:
```
# Build a pie to show the availability by property room type in Airbnb Hawaii
room_type_pie=df.room_type.value_counts()/len(df.room_type)
room_type_pie.plot.pie(autopct='%.2f',fontsize=12,figsize=(8,8))
plt.title('Property Room types availability in AirBnB',fontsize=20)
```

Out[40]: Text(0.5, 1.0, 'Property Room types availability in AirBnB')



In [42]:
```
plt.subplots(figsize=(15,10))
df.groupby(['neighbourhood_group_cleansed','room_type']).room_type.count().pl
plt.ylabel('Neighbourhood wise room types')
plt.xlabel('Number of Rooms')
plt.title('Neighbourhood groups Vs Room Types Availability')
```

Out[42]: Text(0.5, 1.0, 'Neighbourhood groups Vs Room Types Availability')



In [ ]:
```
plt.figure(figsize=(10,6))
plt.scatter(df.longitude, df.latitude, c=df.availability_365, cmap='spring',\
            edgecolor='black', linewidth=1, alpha=0.75)

cbar = plt.colorbar()
cbar.set_label('availability_365')
```

```
Linear Regression

```

In [ ]:
```

```