Airbnb Pricing Analysis Case Study

Created by: Malcomb Brown
Updated: 2023-01-11

Import libraries

In [1]:
```
import os                                       # for interacting with the operating syste
import pandas as pd                             # for manipulating data
from sqlalchemy import create_engine            # for creating the connection engine to th
from database import mysql_cnxn                 # database credentials
import plotly.express as px                     # for interactive plotting
from plotly.offline import init_notebook_mode # Plotly notebook mode

init_notebook_mode(connected = True)            # Plotly graphs will persist
pd.set_option("min_rows", 20)                   # Sets the minimum rows returned from a qu
```

RUN SETUP SCRIPT FIRST!!!!

Setups project subfolder, establishes connection with the database, loads SQL Magic, and imports the dataset.

In [2]:
```
class ProjectSetup():

    """
        Class to setup my Data Analysis Projects.  Automates setting up the project
    directory subfolders and connect to the MySQL database the dataset will be loaded
    to.  Loads the SQL Magic Ipython-sql extension to allow database queries to be done
    in SQL.  Instead of loading the entire table into a dataframe with Pandas and then
    filter, I can filter in the database, saving storage resources. \n

    Database must already exist on the RDBMS Server and the name updated in the
    'database.py' file.

    Parameters
    ============
        database: str
                  Name of the database that will be queried
                  Default: "airbnb"

          create: bool
                  Boolean value that when 'True' will create project subdirectories.
                  Default: False

            conn: str
                  Database connection string.  Currently
                  Default: MySQL
    """

    # Class Variables
    paths = {"raw": "\\Original\\", "prepared": "\\Prepared\\",
            "uploaded": "\\Uploaded\\", "errors": "\\Errors\\",
            "archive": "\\Archive\\"}

    def __init__(self, database: str, create: bool = False):
        self.database = database
        self.conn = mysql_cnxn + database
        self.create = create
```

```
        if self.create:
            self.create_paths()
        else:
            self.mount_paths()

        self.db_connection()

    def __repr__(self):
        return f"{self.database.capitalize()} Data Analysis Case Study Setup Script."

    def mount_paths(self):
        """ Sets up file paths to the project's subfolders."""

        print("="*75)
        print("Getting project directories.....")
        # Get the current path
        self.base_path = os.getcwd()

        # Create a path to the directory for the original csv files.
        self.raw_data_path = f"{self.base_path}{self.paths['raw']}"

        # Create a path to the directory for cleaned datasets
        self.prepared_data_path = f"{self.base_path}{self.paths['prepared']}"

        # Create a path to the directory for files to be loaded in the database
        self.uploaded_data_path = f"{self.base_path}{self.paths['uploaded']}"

        # Create path to the directory to save removed records
        self.errors_data_path = f"{self.base_path}{self.paths['errors']}"

        # Create path to the archive directory
        self.archive_path = f"{self.base_path}{self.paths['archive']}"
        print("All directory paths saved.")
        print("="*75)

    def create_paths(self):
        """ Creates project subdirectories and mounts the paths."""

        self.mount_paths()
        print("="*75)
        print("Creating project folders......")
        dirs = [self.raw_data_path, self.prepared_data_path, self.uploaded_data_path,
                self.errors_data_path, self.archive_path]

        for d in dirs:
            try:
                os.mkdir(d)
            except OSError as error:
                print(error)
                print(f"Project directory not created: {d}")
            print(f"Project directory created: {d}")

        print("Project subfolders setup.")
        print("="*75)

    def db_connection(self):
        """
            Establishes a connection, via SQLAlchemy's 'create_engine' method to the
            database.  Setup notebook to run SQL inline with the '%'.  Currently will
            only work for MySQL and SQLite.

            Adding PosrgreSQL and MS SQL Server....
        """

        print("="*75)
```

```
        print("Loading Ipython-sql.....")

        # Using SQL Magic to interact with the MySQL database
        %load_ext sql

        print("Connecting and configuring to the MySQL database.....")

        # Establish the connection to the MySQL database
        %sql $self.conn

        # Configure output to be returned as a Pandas dataframe.
        %config SqlMagic.autopandas = True

        self.eng = create_engine(self.conn)     # Create the engone to connect to the My
        print("Connection complete!")
        print("="*75)

    def extract_dataset(self, nfile: str, out_file: str = "raw_listings"):
        """
            Extracts data from a single csv file.  If the file is not in the same direct
            <class_name>, the file path needs to be included.  File can also be extracte
            Prints the metadata of the dataframe and saves to project subfolder.

        Parameters
        ============
               nfile: str
                      Name, path, or URL of the csv file to extract

            out_file: str
                      Name of the file that the extracted data will be stored in the 'Or
                      project folder
        """

        print("="*75)
        print("Extracting csv file.....")
        self.raw = pd.read_csv(nfile)
        print("Saving original dataset.....")
        self.raw.to_csv(f"{self.raw_data_path}{out_file}.csv")
        print("Printing metadata......\n")
        print("-"*65)
        print(self.raw.info())
        print("="*75)
        return self.raw
```

Initialize ProjectSetup object

In [3]:
```
project =  ProjectSetup(database="airbnb")
```

```
============================================================================
Getting project directories.....
All directory paths saved.
============================================================================
============================================================================
Loading Ipython-sql.....
Connecting and configuring to the MySQL database.....
Connection complete!
============================================================================
```

In [4]:
```
print(project)
```

```
Airbnb Data Analysis Case Study Setup Script.
```

Save the url of the dataset

In [5]:
```
url = "http://data.insideairbnb.com/united-states/ny/new-york-city/2022-09-07/visualisat
```

Table Schema

Column name	Description
id	Listing id
name	Name of listing
host_id	Host id
host_name	Name of host
neighbourhood_group	Neighbourhood group the listing is in
neighbourhood	Neighbourhood the listing is in
latitude	Latitude coordinate of listing location
longitude	Longitude coordinate of listing location
room_type	Room type of the listing
price	Price of the listing
minimum_nights	Minimum number of nights stay for listing
number_of_reviews	Number of reviews for listing
last_review	Date of the latest review
reviews_per_month	Number of reviews per month of listing
calculated_host_listings_count	Number of listings the host has
availability_365	The availability of the listing in the next 365 days
number_of_reviews_ltm	Number of reviews of listing in last 12 months
license	If host is licensed

Extract CSV file

In [6]:
```
df = project.extract_dataset(url)
```

```
================================================================================
Extracting csv file.....
Saving original dataset.....
Printing metadata......

----------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39881 entries, 0 to 39880
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              39881 non-null  int64
 1   name                            39868 non-null  object
 2   host_id                         39881 non-null  int64
 3   host_name                       39831 non-null  object
 4   neighbourhood_group             39881 non-null  object
 5   neighbourhood                   39881 non-null  object
 6   latitude                        39881 non-null  float64
 7   longitude                       39881 non-null  float64
 8   room_type                       39881 non-null  object
```

```
 9   price                           39881 non-null  int64
 10  minimum_nights                  39881 non-null  int64
 11  number_of_reviews               39881 non-null  int64
 12  last_review                     31519 non-null  object
 13  reviews_per_month               31519 non-null  float64
 14  calculated_host_listings_count  39881 non-null  int64
 15  availability_365                39881 non-null  int64
 16  number_of_reviews_ltm           39881 non-null  int64
 17  license                         5 non-null      object
dtypes: float64(3), int64(8), object(7)
memory usage: 5.5+ MB
None
================================================================================
```

Null values for 'name' and 'host_name' are unnecessary columns because of the 'host_id'.
Remove the 'last_review', 'reviews_per_month', and 'license' columns
Optimize the data types before uploading.

In [7]:
```
# Save the original data file.
df.to_csv(f"{project.raw_data_path}raw_listings.csv", index=False)
```

Inspect the dataframe

In [8]:
```
df
```

Out[8]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitu
0	77765	Superior @ Box House	417504	The Box House Hotel	Brooklyn	Greenpoint	40.737
1	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.645
2	45910	Beautiful Queens Brownstone! - 5BR	204539	Mark	Queens	Ridgewood	40.703
3	45935	Room in Beautiful Townhouse.	204586	L	Bronx	Mott Haven	40.806
4	45936	Couldn't Be Closer To Columbia Uni	867225	Rahul	Manhattan	Morningside Heights	40.806
5	80493	Cozy room in East Village with AC	434987	Jennifer	Manhattan	East Village	40.723
6	46911	Large Room in private Brownstone in Park Slope	210746	Kathleen R.	Brooklyn	Prospect Heights	40.680
7	49048	B and B Style Rooms for Rent w bath	35935	Angela	Brooklyn	Bedford-Stuyvesant	40.682
8	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.753
9	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-	40.685

						Stuyvesant	
...	...	...	...	...	...	...	
39871	50547940	Lovely 2 bedroom apartment in Brooklyn	408408008	Bernadette	Brooklyn	Green-Wood Cemetery	40.655
39872	628769808856889664	Two bedroom apartment in Hoboken New Jersey.	14468718	Burak	Manhattan	West Village	40.740
39873	629813073409916623	Lovely/ 1 bedroom/ cool- 15 mins to NYC	459789709	Lima	Manhattan	Upper West Side	40.785
39874	637667152834352362	Big 1 Bedroom in Jersey City	121909760	Brian	Manhattan	Ellis Island	40.718
39875	670851597591736404	Modern Luxury Home	242323176	Stephanie	Queens	Queens Village	40.706
39876	27577588	Luxury Studio ON Grove Street E0C - B1CA	37412692	Kim	Manhattan	Ellis Island	40.718
39877	654151117629853651	Lovely 3-bedroom apartment	117540494	Miriam	Queens	Rosedale	40.647
39878	553754115911961053	Trendy 3-bedroom apartment near Manhattan	15048320	India	Manhattan	Upper West Side	40.787
39879	698195550745703156	Luxurious private waterfront terrace, 2BR 2BA Apt	151487807	Asser	Brooklyn	Williamsburg	40.709
39880	48971505	Just Blocks to Grove PATH and JC Med Ctr	46201	J	Manhattan	Ellis Island	40.718

39881 rows × 18 columns

Transform the data

In [9]:

```
# Drop unnecessary columns
df.drop(columns=["name", "host_name", "last_review", "reviews_per_month", "license"], in

# Change the 'neighbourhood_group' and 'neighbourhood' column names
df.rename(columns={"neighbourhood_group": "neighborhood_group", "neighbourhood": "neighb
                   "calculated_host_listings_count": "total_host_listings",
                   "number_of_reviews_ltm": "reviews_in_last_yr"}, inplace=True)

# Convert remaining data types
df = df.convert_dtypes()
```

In [10]:
```
# Save dataframe
df.to_csv(f"{project.prepared_data_path}listings_prepped.csv", index=False)
```

Reinspect the data

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39881 entries, 0 to 39880
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    39881 non-null  Int64
 1   host_id               39881 non-null  Int64
 2   neighborhood_group    39881 non-null  string
 3   neighborhood          39881 non-null  string
 4   latitude              39881 non-null  Float64
 5   longitude             39881 non-null  Float64
 6   room_type             39881 non-null  string
 7   price                 39881 non-null  Int64
 8   minimum_nights        39881 non-null  Int64
 9   number_of_reviews     39881 non-null  Int64
 10  total_host_listings   39881 non-null  Int64
 11  availability_365      39881 non-null  Int64
 12  reviews_in_last_yr    39881 non-null  Int64
dtypes: Float64(2), Int64(8), string(3)
memory usage: 4.3 MB
```

No Null values remain

In [12]:
```
# Check summary statistics before uploading
df.describe(include="all")
```

Out[12]:

	id	host_id	neighborhood_group	neighborhood	latitude	longitude	room_type
count	3.988100e+04	3.988100e+04	39881	39881	39881.000000	39881.000000	39881
unique	NaN	NaN	5	244	NaN	NaN	4
top	NaN	NaN	Manhattan	Bedford-Stuyvesant	NaN	NaN	Entire home/apt
freq	NaN	NaN	16847	2779	NaN	NaN	22761
mean	1.315489e+14	1.313420e+08	NaN	NaN	40.728870	-73.945665	NaN
std	2.465197e+17	1.455674e+08	NaN	NaN	0.058623	0.057870	NaN
min	2.539000e+03	2.438000e+03	NaN	NaN	40.500314	-74.269520	NaN
25%	1.633197e+07	1.363938e+07	NaN	NaN	40.687760	-73.983340	NaN
50%	3.823683e+07	5.974663e+07	NaN	NaN	40.724545	-73.953710	NaN
75%	5.255780e+07	2.233746e+08	NaN	NaN	40.763200	-73.925600	NaN
max	7.098549e+17	4.782606e+08	NaN	NaN	40.928810	-73.690060	NaN

Top Categorical Variables

Frequency in parenthesis.

Neighborhood Group: Manhattan (16847)

Neighborhood: Bedford-Stuyvesant (2779)

Room Type: Entire home/apt (22761)

There appears to be record(s) that have no price (0) value.

In [13]:
```
# Get a dataframe of all the records where the price equals 0
zero_prices = df[df["price"] == 0]
zero_prices
```

Out[13]:

	id	host_id	neighborhood_group	neighborhood	latitude	longitude	room_type	price	minir
20862	40560656	273324213	Brooklyn	Williamsburg	40.72096	-73.9586	Hotel room	0	
21428	41740615	268417148	Manhattan	Midtown	40.74459	-73.98574	Hotel room	0	
21429	41740622	269311462	Manhattan	Upper East Side	40.76442	-73.96303	Hotel room	0	
21626	42065543	307634016	Manhattan	Midtown	40.74444	-73.9892	Hotel room	0	
21628	42065545	310429455	Manhattan	Midtown	40.75917	-73.96926	Hotel room	0	
21629	42065547	308721299	Manhattan	Hell's Kitchen	40.76404	-73.99478	Hotel room	0	
21630	42065555	309714886	Brooklyn	Williamsburg	40.71523	-73.95908	Hotel room	0	
21641	42065562	307633956	Manhattan	Financial District	40.70958	-74.00874	Hotel room	0	
21642	42065563	309772430	Bronx	Mott Haven	40.81513	-73.91602	Hotel room	0	
21643	42065564	314151200	Manhattan	Financial District	40.70462	-74.01027	Hotel room	0	
21651	42279171	265458818	Manhattan	Chinatown	40.7161	-73.99518	Hotel room	0	
21664	42228997	314197504	Manhattan	Lower East Side	40.72186	-73.99278	Hotel room	0	
21783	42384501	262458398	Manhattan	Chelsea	40.74793	-73.99117	Hotel room	0	
22160	43078550	334334264	Manhattan	Kips Bay	40.74097	-73.98339	Hotel room	0	
22164	43078552	342053968	Manhattan	Lower East Side	40.72214	-73.98857	Hotel room	0	
22168	43035720	318559292	Manhattan	Midtown	40.75028	-73.98547	Hotel room	0	
22181	43205598	335389657	Manhattan	Midtown	40.76448	-73.98055	Hotel room	0	
22326	43247386	335072254	Manhattan	Hell's Kitchen	40.76756	-73.98312	Hotel room	0	
22328	43247472	324955773	Manhattan	Midtown	40.76085	-73.96938	Hotel room	0	
22334	43247631	318788301	Manhattan	Hell's Kitchen	40.76175	-73.9882	Hotel room	0	
23017	44567521	360662584	Manhattan	East Village	40.72743	-73.99136	Hotel room	0	
23826	46059074	373324108	Manhattan	Theater District	40.762368	-73.985676	Hotel room	0	
23838	46251446	374516933	Manhattan	Lower East Side	40.719732	-73.993996	Hotel room	0	
23902	46087899	373522899	Brooklyn	Williamsburg	40.72121	-73.957209	Hotel room	0	
23984	46336133	375044940	Manhattan	Chelsea	40.755322	-74.001772	Hotel room	0	
24115	46723973	376877842	Manhattan	Hell's Kitchen	40.765708	-73.995575	Hotel room	0	
25050	48089897	261016212	Brooklyn	Bedford-	40.696787	-73.958005	Hotel room	0	

				Stuyvesant				
25189	48325676	390077597	Manhattan	Upper West Side	40.781629	-73.982004	Hotel room	0
25269	48417136	390810530	Manhattan	Midtown	40.746836	-73.982699	Hotel room	0
38826	45861040	371797355	Queens	Rockaway Park	40.5811	-73.83028	Hotel room	0

In [14]:
```
# Save deleted records to the Error folder
zero_prices.to_csv(f"{project.errors_data_path}zero_prices.csv", index=False)
```

In [15]:
```
# Filter out the 'zero_prices' and overwrite the 'listings_df'
df = df[df.price != 0]
```

In [16]:
```
# Save dataframe
df.to_csv(f"{project.uploaded_data_path}listings.csv", index=False)
```

Upload the cleaned verified dataset to a MySQL database for analysis

In [17]:
```
df.to_sql(name="listings", con=project.eng, if_exists="replace", index=False)
```

Out[17]:
```
39851
```

Initial Assumptions

- Listings in Manhattan will have higher average prices.
- Entire home/apt listings will have higher average prices.
- Listings with more reviews will have average higher prices

Questions

- Does the neighborhood group geographic location impact price?
- Does room_type have an affect on price?
- Does the number and/or quality of reviews reflect the price?
- Does availability have any affect on price?
- Where do the neighborhoods lie geographically?
- Does proximity to downtown affect the price? Waterfront?
- Could there be other factors? Traffic? Crime? Shopping?
- Linear Regression? Decision Tree? Random Forest? Or SVM?

In [18]:
```
%%sql
SELECT
    COUNT(id)                           AS listings,
    COUNT(DISTINCT host_id)             AS hosts,
    COUNT(DISTINCT neighborhood_group)  AS neighborhood_groups,
    COUNT(DISTINCT neighborhood)        AS neighborhoods,
    COUNT(DISTINCT room_type)           AS room_types,
    ROUND(AVG(price), 2)                AS avg_price,
    MIN(price)                          AS min_price,
    MAX(price)                          AS max_price
FROM
    airbnb.listings;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
1 rows affected.
```

Out[18]:

	listings	hosts	neighborhood_groups	neighborhoods	room_types	avg_price	min_price	max_price
0	39851	26263	5	244	4	197.70	10	16500

Find the median

In [19]:
```
# Get the median listing price
df.price.median()
```

Out[19]:
```
130.0
```

In [20]:
```
# Check the skew
df.price.skew()
```

Out[20]:
```
18.615593182925227
```

There are outliers, in price, too the higher, right, side of the dataset

Dataset has a positive or right skew.

Visualize the distribution with histogram

In [21]:
```
fig = px.histogram(df, x="price",
                   nbins=100, title="Price Distribution",
                   labels={"price": "Price (USD)"})
fig.show()
```

What is the most common price point?

In [22]:
```
df.price.value_counts().to_frame().head(1)
```

Out[22]:

	price
150	1164

In [23]:
```
del df
```

What percent of listings are priced below $200.00?

In [24]:
```
%%sql
SELECT
    *,
    ROUND(((listings_below_200 / total_listings) * 100), 2) AS percent_below_200
FROM
    (SELECT
        COUNT(id) AS total_listings,
        (SELECT COUNT(id) FROM airbnb.listings WHERE price <= 199) AS listings_below_200
    FROM
        airbnb.listings) AS listings_dist;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
1 rows affected.
```

Out[24]:

	total_listings	listings_below_200	percent_below_200
0	39851	28225	70.83

70.83% of all listings are priced between 0 and 199.00 dollars

Quick Feature Summary

```
            Listings: 39,851
               Hosts: 26,263
Neighborhood Groups: 5
      Neighborhoods: 244
         Room Types: 4
Average Price (USD): 197.70
Minimum Price (USD): 10.00
 Median Price (USD): 130.00
Maximum Price (USD): 16,500.00
```

The price of 150.00 is the most common price point, appearing 1164 times.

Location

Top 10 Neighbohoods by average price

In [25]:
```
%%sql top_10_neighborhoods <<
SELECT
    neighborhood,
    COUNT(id) AS total_listings,
    ROUND(AVG(price), 2) AS avg_price
FROM
```

```
    airbnb.listings
GROUP BY
    neighborhood
ORDER BY
    avg_price DESC
LIMIT 10;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
10 rows affected.
Returning data to local variable top_10_neighborhoods
```

Visualize the Top 10 Neighborhoods

In [26]:

```
fig = px.bar(data_frame=top_10_neighborhoods,
             x="neighborhood",
             y="avg_price",
             color="neighborhood",
             title="Top 10 Neighborhoods by Average Prices",
             labels={"neighborhood": "Neighborhood", "avg_price": "Avg. Price (USD)", "t
             text="avg_price",
             hover_data=["total_listings"])
fig.show()
```

In [27]:

```
del top_10_neighborhoods
```

Prospect Park and Fort Wadsworth have the highest average price for listings in their neighborhood. The values are skewed do to the low number of listings for each.

Prospect Park has 7 listings and Fort Wadsworth only has 1.

What are the bottom 10 Neighborhoods by average listing price?

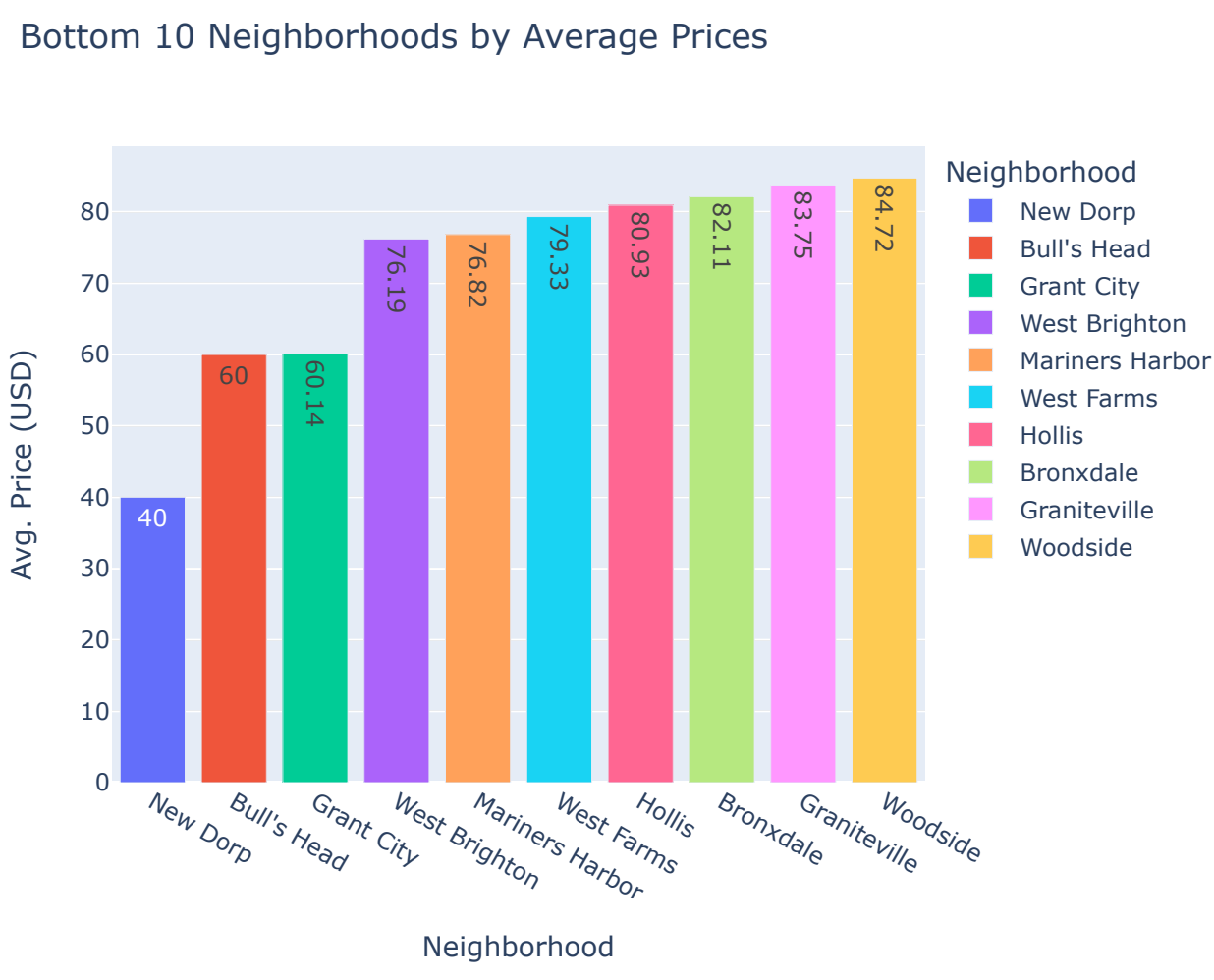
In [28]:

```
%%sql bottom_10_neighborhoods <<
SELECT
    neighborhood,
    COUNT(id) AS total_listings,
    ROUND(AVG(price), 2) AS avg_price
FROM
    airbnb.listings
GROUP BY
    neighborhood
ORDER BY
    avg_price
LIMIT 10;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
10 rows affected.
Returning data to local variable bottom_10_neighborhoods
```

In [29]:

```
fig = px.bar(data_frame=bottom_10_neighborhoods,
             x="neighborhood",
             y="avg_price",
             color="neighborhood",
             title="Bottom 10 Neighborhoods by Average Prices",
             labels={"neighborhood": "Neighborhood", "avg_price": "Avg. Price (USD)", "t
             text="avg_price",
             hover_data=["total_listings"])
fig.show()
```



With some neighborhoods having only 1 listing, comparing average prices by this is highly susceptible to outliers.

In [30]:
```
del bottom_10_neighborhoods
```

Average price by 'neighborhood_group'

In [31]:
```
%%sql neighborhood_groups <<
SELECT
    neighborhood_group,
    COUNT(DISTINCT neighborhood) AS neighborhoods,
    COUNT(id) AS listings,
    ROUND(AVG(price), 2) AS avg_price,
    MAX(price) AS max_price,
    MIN(price) AS min_price
FROM
    airbnb.listings
GROUP BY
    neighborhood_group
ORDER BY avg_price DESC;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
5 rows affected.
Returning data to local variable neighborhood_groups
```

In [32]:
```
neighborhood_groups
```

Out[32]:

	neighborhood_group	neighborhoods	listings	avg_price	max_price	min_price
0	Manhattan	34	16823	265.31	16500	10
1	Brooklyn	51	14841	157.97	10000	10
2	Staten Island	46	446	143.16	2500	33
3	Queens	58	6174	131.39	10000	10
4	Bronx	55	1567	124.82	9994	10

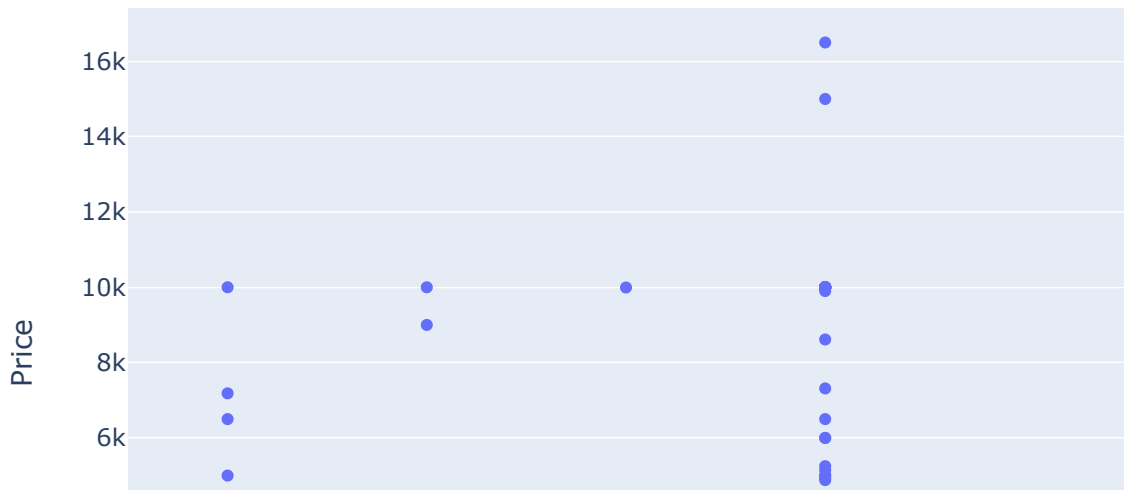
In [33]:
```
fig = px.bar(data_frame=neighborhood_groups,
             x="neighborhood_group",
             y="avg_price",
             color="neighborhood_group",
             title="Average Price by Neighborhood Group",
             labels={"neighborhood_group": "Neighborhood Group",
                     "avg_price": "Avg. Price (USD)", "listings": "Total Listings",
                    "neighborhoods": "Neighborhoods", "max_price": "Max Price (USD)",
                    "min_price": "Min Price (USD)"},
             text="avg_price",
             hover_data=["listings", "neighborhoods", "max_price", "min_price"])
fig.show()
```

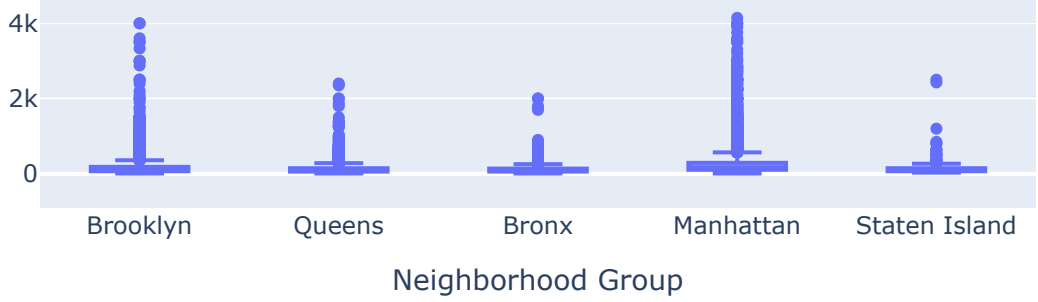
As expected, Manhattan listings have the highest average price.
Manhattan also has the most listings
Unexpectedly, Staten Island, has, by far, the least number of listings and the lowest max price but has the third highest average price.

In [34]:
```
ngbox_df = %sql SELECT neighborhood_group, price FROM airbnb.listings;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
39851 rows affected.
```

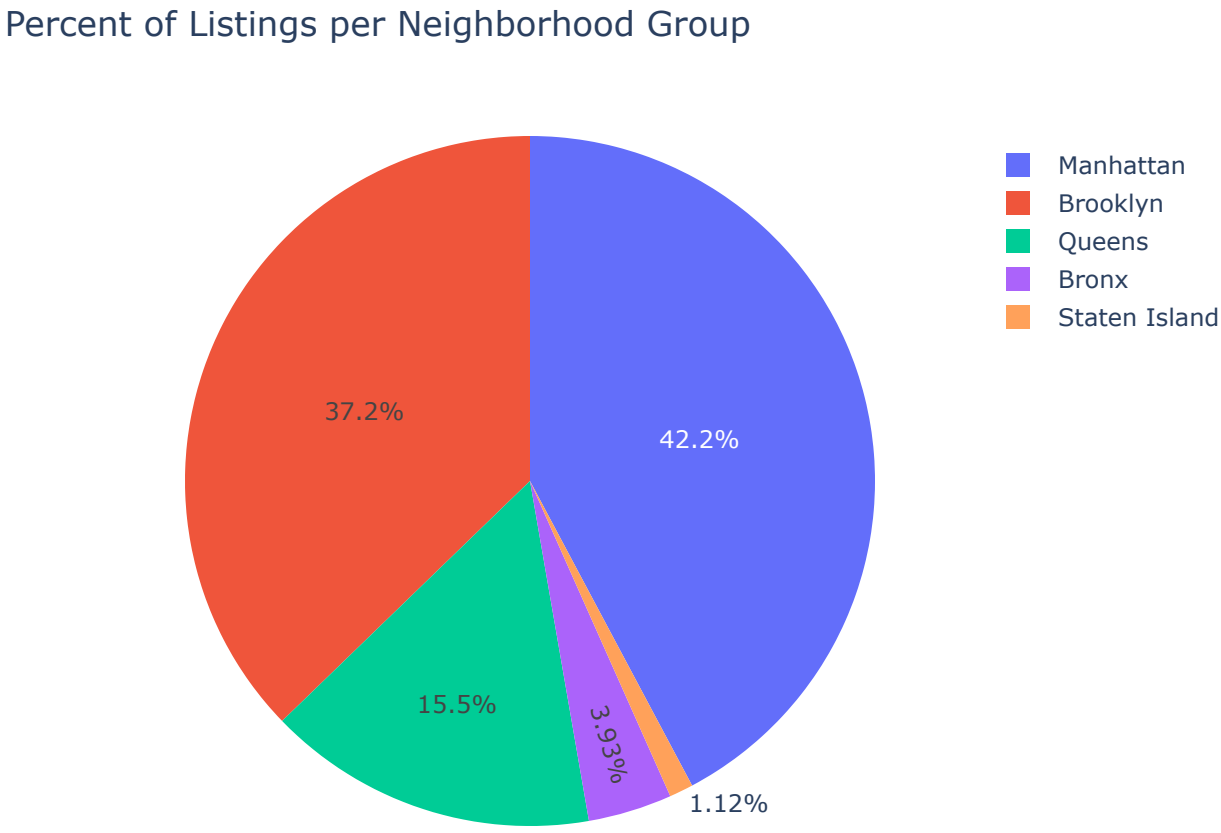
In [35]:
```
fig = px.box(data_frame=ngbox_df,
             x="neighborhood_group",
             y="price",
             title="Neighborhood Group Outliers",
             labels={"neighborhood_group": "Neighborhood Group", "price": "Price"})
fig.show()
```





What percentage of listings does each neighborhood group have?

In [36]:
```
fig = px.pie(data_frame=neighborhood_groups,
             names="neighborhood_group",
             values="listings",
             labels={"neighborhood_group": "Neighborhood Group", "listings": "Listings"}
             title="Percent of Listings per Neighborhood Group")
fig.show()
```



79.4% of all listings are located in either Manhattan or Brooklyn.

In [37]:
```
del neighborhood_groups
del ngbox_df
```

Which neighborhood groups have the most availabilities?

In [38]:
```
%%sql avail_df <<
SELECT
    ur.neighborhood_group,
    ar.rentals_available,
    ar.avg_price AS avg_price_avail,
    ur.rentals_unavailable,
    ur.avg_price AS avg_price_unavail
FROM
    (SELECT
        neighborhood_group,
        COUNT(availability_365) AS rentals_unavailable,
        ROUND(AVG(price), 2) AS avg_price
    FROM
        airbnb.listings
    WHERE availability_365 <= 0
    GROUP BY neighborhood_group
    ORDER BY rentals_unavailable DESC) AS ur
        JOIN
    (SELECT
        neighborhood_group,
        COUNT(availability_365) AS rentals_available,
        ROUND(AVG(price), 2) AS avg_price
    FROM
        airbnb.listings
    WHERE availability_365 > 0
    GROUP BY neighborhood_group
    ORDER BY rentals_available DESC) AS ar
    ON ur.neighborhood_group = ar.neighborhood_group;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
5 rows affected.
Returning data to local variable avail_df
```

In [39]:
```
avail_df
```

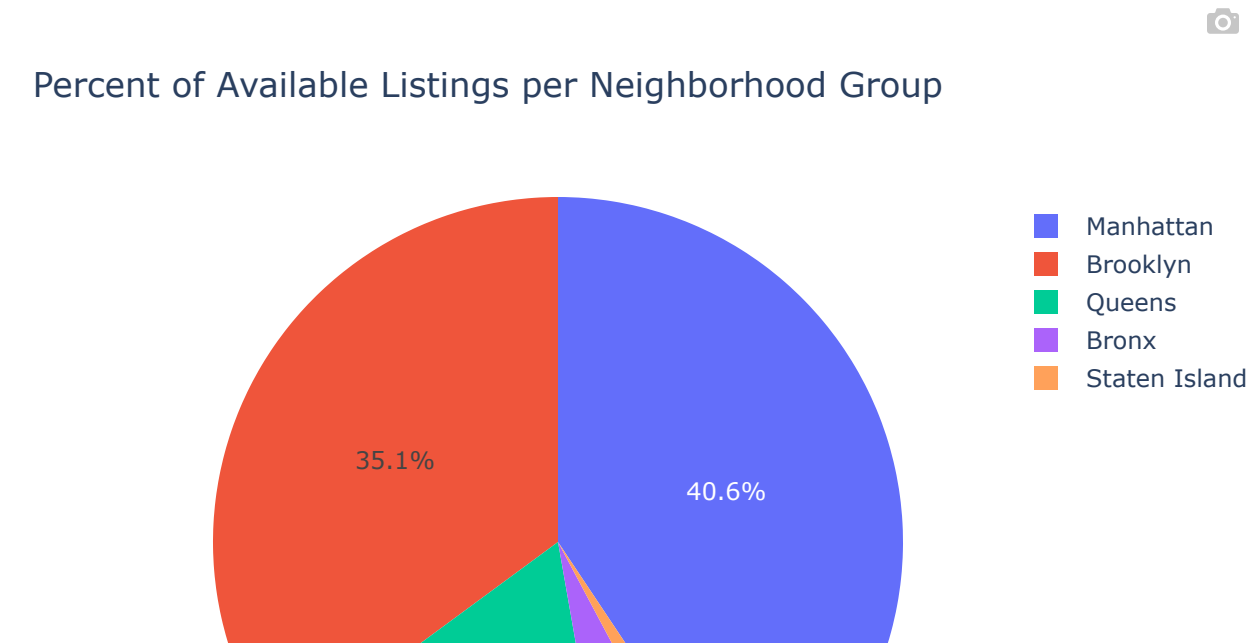
Out[39]:

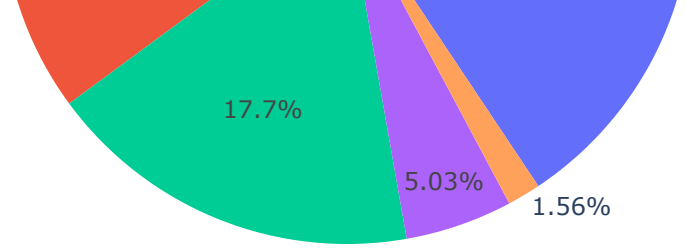
	neighborhood_group	rentals_available	avg_price_avail	rentals_unavailable	avg_price_unavail
0	Manhattan	10372	305.28	6451	201.04
1	Brooklyn	8958	178.21	5883	127.15
2	Queens	4509	140.85	1665	105.77
3	Bronx	1283	131.15	284	96.21
4	Staten Island	398	145.25	48	125.88

In [40]:
```
fig = px.bar(data_frame=avail_df,
             x="neighborhood_group",
             y=["rentals_available", "rentals_unavailable"],
             title="Listing Availability by Neighborhood Group",
             labels={"neighborhood_group": "Neighborhood Group",
                     "avg_price_avail": "Avilable Avg. Price",
                     "avg_price_unavail": "Unavailable Avg. Price",
                     "rentals_available": "Available",
                     "rentals_unavailable": "Unavailable",
                     "value": "Number of Listings",
                     "variable": "Rental Availability"
                    },
             hover_data=["avg_price_avail", "avg_price_unavail"],
             orientation = "v",
             barmode="group",
             text_auto=True)
fig.show()
```

In [41]:

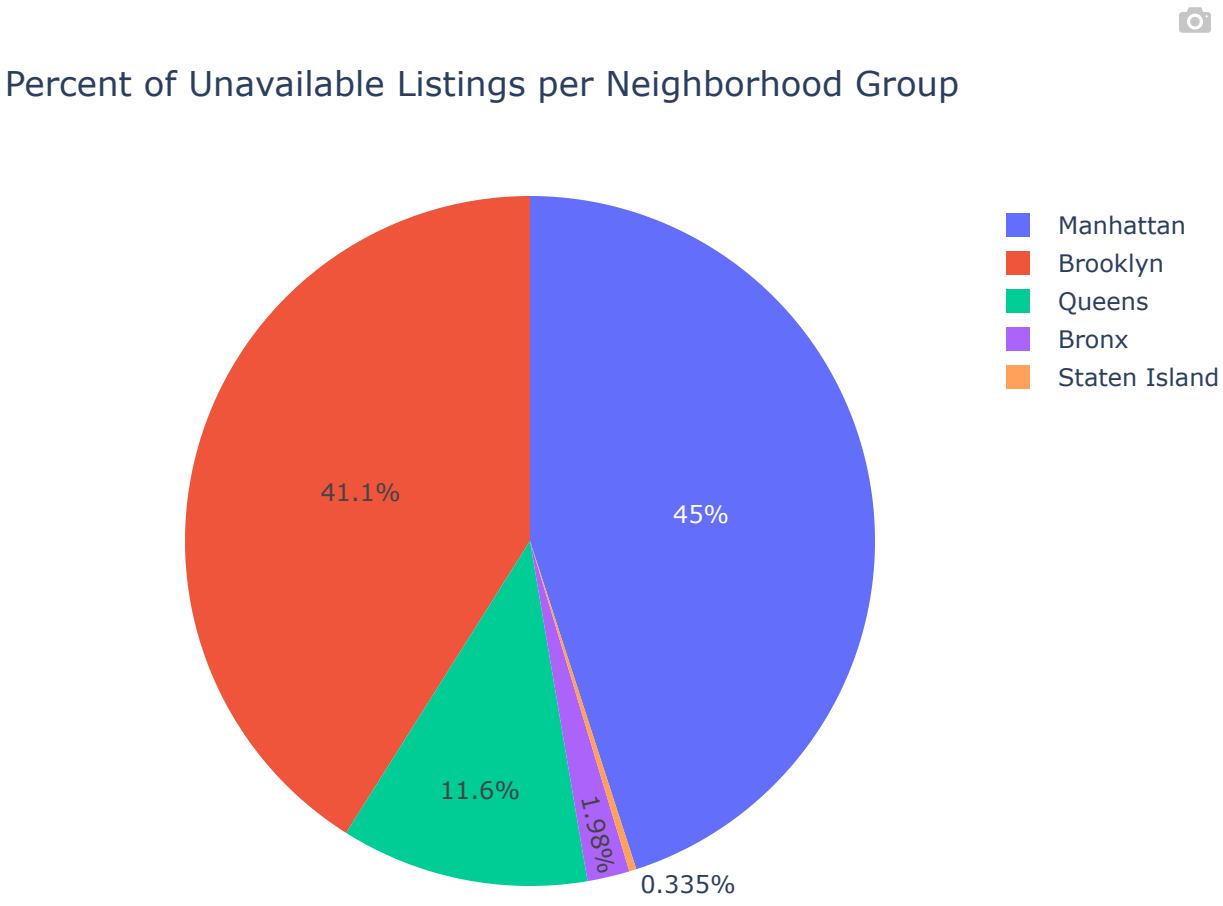
```
fig = px.pie(data_frame=avail_df,
             names="neighborhood_group",
             values="rentals_available",
             labels={"neighborhood_group": "Neighborhood Group",
                     "rentals_available": "Rentals Available",
                     "avg_price_avail": "Avg. Price (USD)"},
             title="Percent of Available Listings per Neighborhood Group",
             hover_data=["avg_price_avail"])
fig.show()
```





In [42]:

```
fig = px.pie(data_frame=avail_df,
             names="neighborhood_group",
             values="rentals_unavailable",
             labels={"neighborhood_group": "Neighborhood Group",
                     "rentals_unavailable": "No. of Unavailabilities",
                     "avg_price_unavail": "Avg. Price (USD)"},
             title="Percent of Unavailable Listings per Neighborhood Group",
             hover_data=["avg_price_unavail"])
fig.show()
```



In [43]:

```
del avail_df
```

75.7% of all listings with any availability over the next year are located in either Manhattan or Brooklyn.

86.1% of all listings with no availabilities over the next year are in Manhattan and Brooklyn.

Visualize price by location

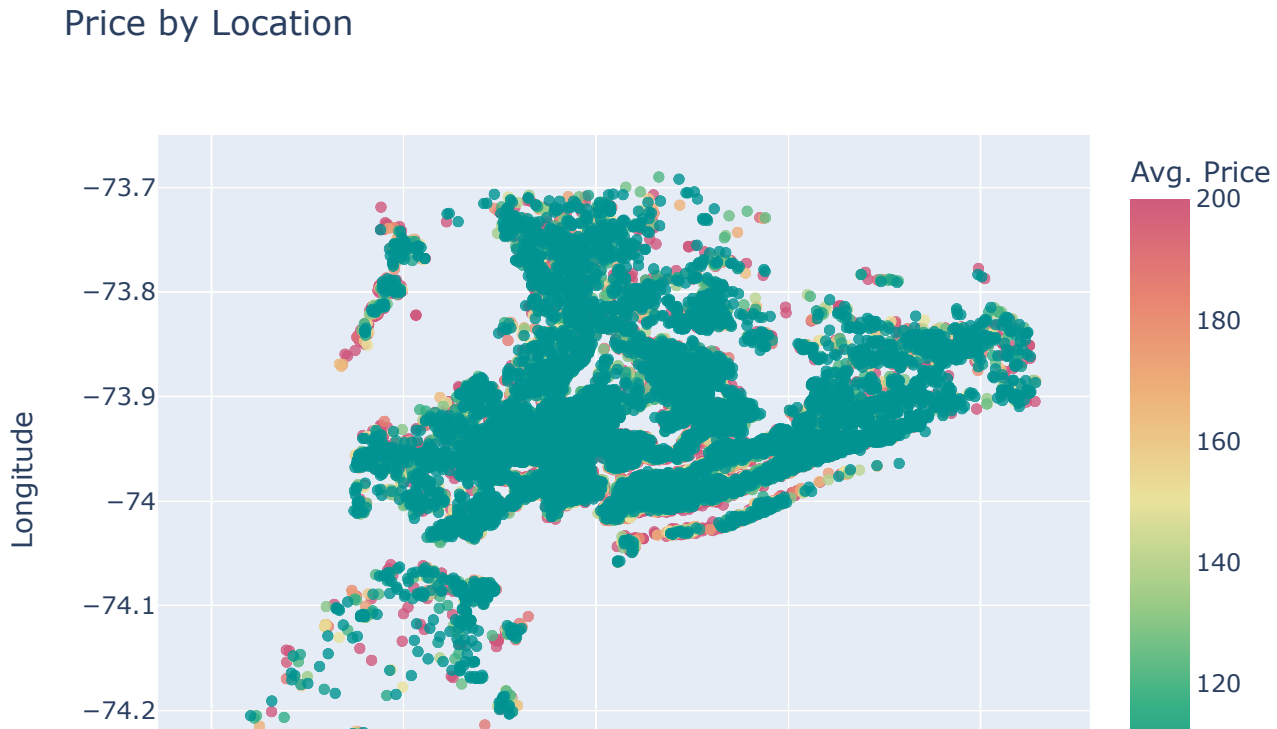
In [44]:
```
%%sql map_df <<
SELECT
    neighborhood,
    neighborhood_group,
    COUNT(id) AS total_listings,
    ROUND(AVG(price), 2) AS avg_price,
    latitude,
    longitude
FROM
    airbnb.listings
GROUP BY
    neighborhood, neighborhood_group, latitude, longitude
ORDER BY
    avg_price DESC;
```

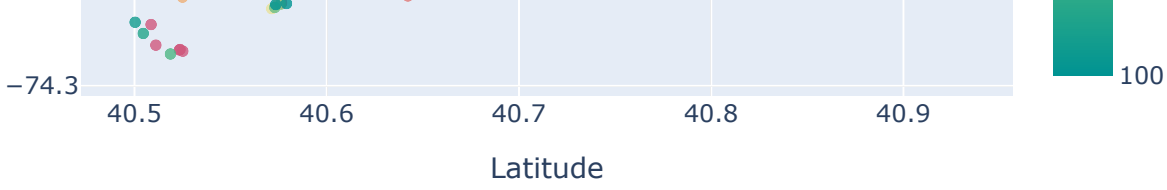
```
 * mysql+pymysql://root:***@localhost/airbnb
38751 rows affected.
Returning data to local variable map_df
```

In [45]:
```
# Need a list of average prices for the scatter plot
prices = [float(price) for price in map_df.avg_price.to_list()]
```

In [46]:
```
fig = px.scatter(data_frame=map_df, x="latitude", y="longitude",
                 color=prices,
                 hover_data=["neighborhood", "neighborhood_group"],
                 labels={"neighborhood_group": "Neighborhood Group", "neighborhood": "Ne
                         "size": "Avg. Price", "latitude": "Latitude", "longitude": "Lor
                         "color": "Avg. Price"},
                 range_color=[100, 200],
                 title="Price by Location",
                 color_continuous_scale='temps',
                 opacity=0.8)

fig.show()
```





In [47]:
```
del map_df
```

Room Type

In [48]:
```
%%sql room_types <<
SELECT
    room_type,
    COUNT(id) AS listings,
    ROUND(AVG(price), 2) AS avg_price,
    MIN(price) AS min_price,
    MAX(price) AS max_price
FROM
    airbnb.listings
GROUP BY
    room_type
ORDER BY
    avg_price DESC;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
4 rows affected.
Returning data to local variable room_types
```

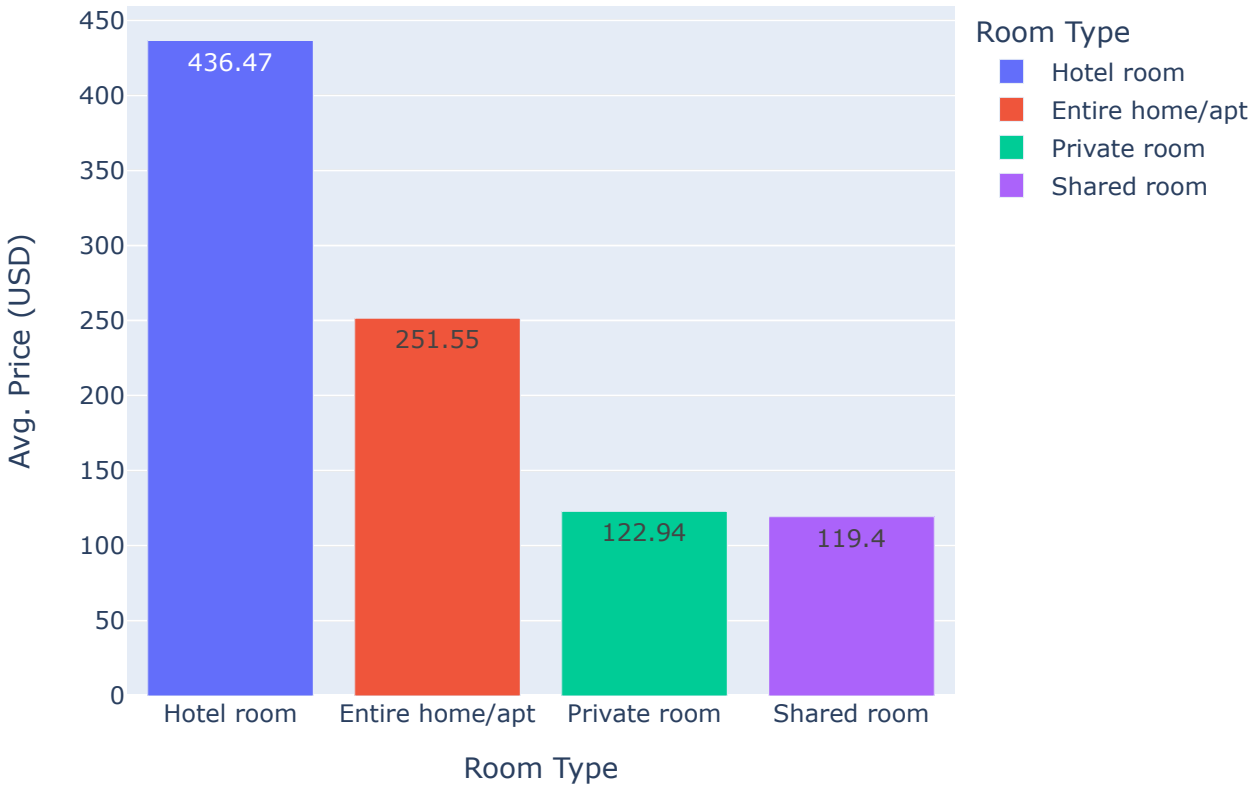
In [49]:
```
room_types
```

Out[49]:

	room_type	listings	avg_price	min_price	max_price
0	Hotel room	172	436.47	100	1998
1	Entire home/apt	22761	251.55	10	15000
2	Private room	16361	122.94	10	16500
3	Shared room	557	119.40	10	10000

In [50]:
```
fig = px.bar(data_frame=room_types,
             x="room_type",
             y="avg_price",
             color="room_type",
             title="Average Price by Room Type",
             labels={"room_type": "Room Type",
                     "avg_price": "Avg. Price (USD)", "listings": "Total Listings",
                     "max_price": "Max Price (USD)", "min_price": "Min Price (USD)"},
             text="avg_price",
             hover_data=["listings", "max_price", "min_price"])
fig.show()
```

Average Price by Room Type



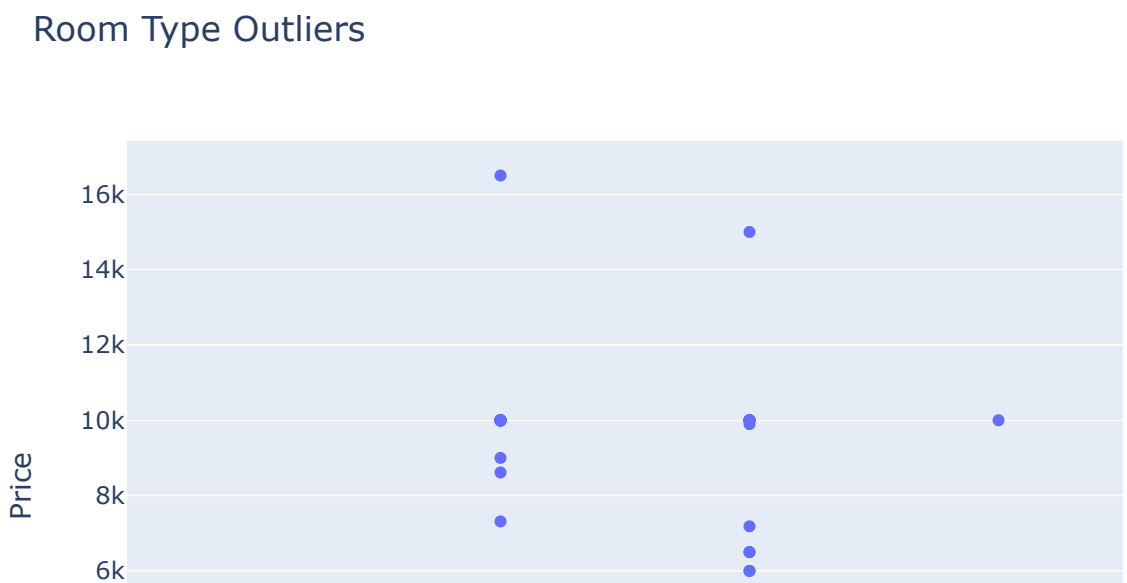
Hotel rooms have the highest average prices.
Entire Home/Apts have the second highest average price, more than Private and Shared rooms combined.

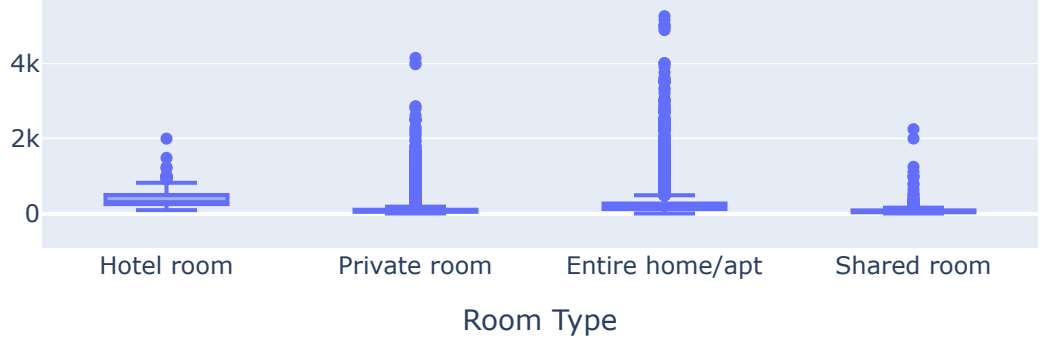
Room Type Outliers

In [51]:

```
rtbox_df = %sql SELECT room_type, price FROM airbnb.listings;
fig = px.box(data_frame=rtbox_df,
             x="room_type",
             y="price",
             title="Room Type Outliers",
             labels={"room_type": "Room Type", "price": "Price"})
fig.show()
```

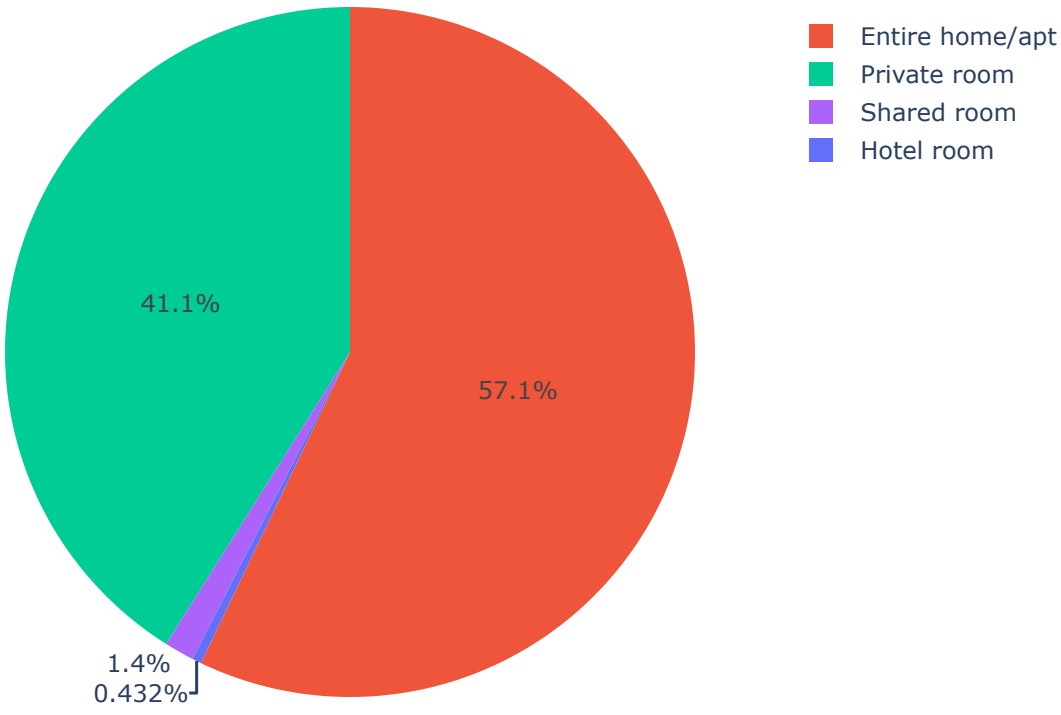
```
 * mysql+pymysql://root:***@localhost/airbnb
39851 rows affected.
```





Hotel rooms have more consistent pricing, havint the fewest outliers

In [52]:
```
fig = px.pie(data_frame=room_types,
             names="room_type",
             values="listings",
             color="room_type",
             title="Number of Listings by Room Type",
             labels={"room_type": "Room Type", "listings": "Listings" , "avg_price": "Av
            )
fig.show()
```



Entire home/apt and Private room account for 98.2% of all listings

In [53]:
```
del room_types
del rtbox_df
```

Room type and Location

In [54]:
```
%%sql borough_df <<
SELECT
    neighborhood_group,
    room_type,
    COUNT(id) AS listings,
    ROUND(AVG(price), 2) AS avg_price,
    MIN(price) AS min_price,
    MAX(price) AS max_price
FROM
    airbnb.listings
GROUP BY
    neighborhood_group, room_type
ORDER BY
    avg_price DESC;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
18 rows affected.
Returning data to local variable borough_df
```

In [55]:
```
borough_df
```

Out[55]:

	neighborhood_group	room_type	listings	avg_price	min_price	max_price
0	Manhattan	Hotel room	159	451.36	143	1998
1	Brooklyn	Hotel room	5	319.60	145	529
2	Manhattan	Entire home/apt	10862	300.65	29	15000
3	Brooklyn	Entire home/apt	8154	216.45	30	7184
4	Queens	Hotel room	8	213.63	100	282
5	Manhattan	Private room	5552	194.92	10	16500
6	Queens	Entire home/apt	2736	191.69	10	10000
7	Staten Island	Entire home/apt	273	180.49	39	2500
8	Manhattan	Shared room	250	175.12	29	10000
9	Bronx	Entire home/apt	736	164.57	28	2000
10	Bronx	Private room	793	90.53	11	9994
11	Brooklyn	Private room	6510	86.91	10	10000
12	Staten Island	Private room	172	84.41	33	500
13	Queens	Private room	3334	83.12	19	9000
14	Queens	Shared room	96	82.09	16	1250
15	Bronx	Shared room	38	70.45	10	775
16	Brooklyn	Shared room	172	70.39	15	1000
17	Staten Island	Shared room	1	59.00	59	59

In [56]:
```
fig = px.bar(data_frame=borough_df,
             x="neighborhood_group",
             y="avg_price",
             color="room_type",
             title="Average Price by Neighborhood Group and Room Type",
```

```
            labels={"room_type": "Room Type", "avg_price": "Avg. Price (USD)",
                    "neighborhood_group": "Neighborhood Group",
                    "listings": "Total Listings", "uid": "Location & Room Type"},
            hover_data=["listings"],
            text_auto=True,
            barmode="group"
           )
fig.show()
```

In [57]:
```
del borough_df
```

Availability by Room Type

In [58]:
```
%%sql avail_df <<
SELECT
    ar.room_type,
    ar.rentals_available,
    ar.avg_price AS avg_price_avail,
    ur.rentals_unavailable,
    ur.avg_price AS avg_price_unavail
FROM
    (SELECT
        room_type,
        COUNT(availability_365) AS rentals_unavailable,
        ROUND(AVG(price), 2) AS avg_price
    FROM
        airbnb.listings
    WHERE availability_365 <= 0
    GROUP BY room_type
```

```
    ORDER BY rentals_unavailable DESC) AS ur
        JOIN
    (SELECT
        room_type,
        COUNT(availability_365) AS rentals_available,
        ROUND(AVG(price), 2) AS avg_price
    FROM
        airbnb.listings
    WHERE availability_365 > 0
    GROUP BY room_type
    ORDER BY rentals_available DESC) AS ar
    ON ur.room_type = ar.room_type;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
4 rows affected.
Returning data to local variable avail_df
```

In [59]:

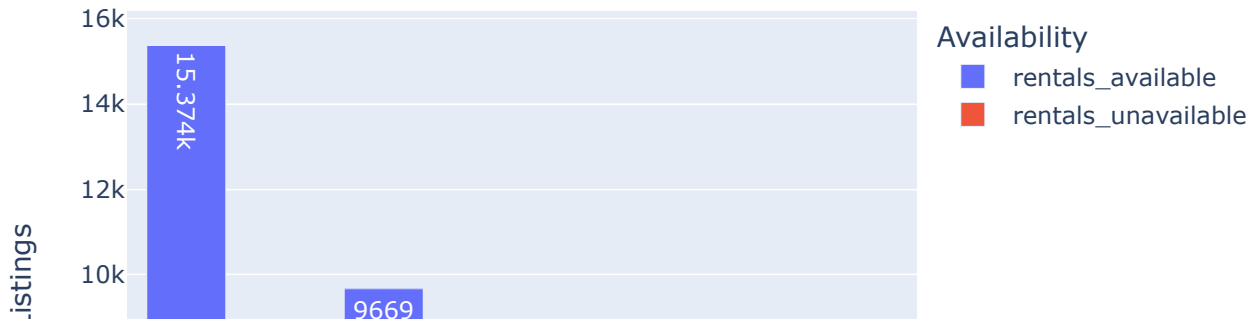
```
avail_df
```

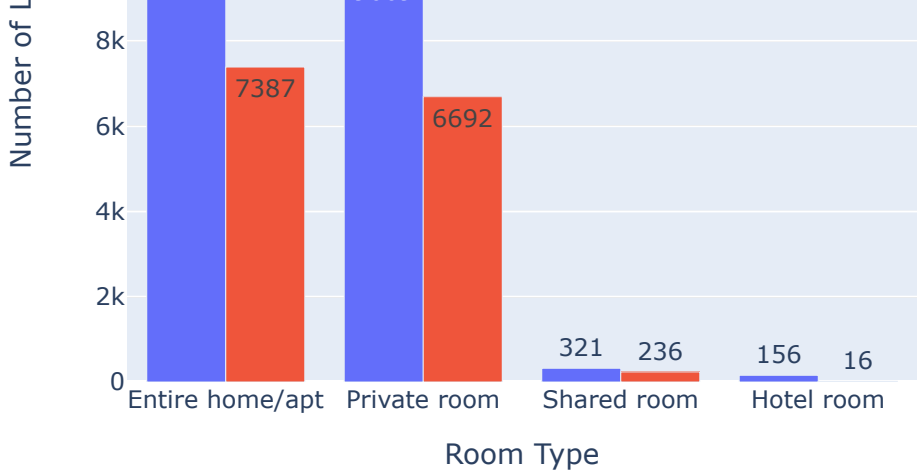
Out[59]:

	room_type	rentals_available	avg_price_avail	rentals_unavailable	avg_price_unavail
0	Entire home/apt	15374	270.41	7387	212.29
1	Private room	9669	141.55	6692	96.05
2	Shared room	321	91.11	236	157.88
3	Hotel room	156	441.46	16	387.81

In [60]:

```
fig = px.bar(data_frame=avail_df,
             x="room_type",
             y=["rentals_available", "rentals_unavailable"],
             title="Listing Availability by Room Type",
             labels={"rentals_available": "No. of Availabilities",
                     "room_type": "Room Type",
                     "avg_price_avail": "Available Avg. Price (USD)",
                     "avg_price_unavail": "Unavailable Avg. Price (USD)",
                     "rentals_available": "Available",
                     "rentals_unavailable": "Unavailable",
                     "variable": "Availability",
                     "value": "Number of Listings"
                    },
             hover_data=["avg_price_avail", "avg_price_unavail"],
             orientation = "v",
             barmode="group",
             text_auto=True)
fig.show()
```





In [61]:
```
del avail_df
```

Reviews

In [62]:
```
%sql SELECT * FROM airbnb.listings LIMIT 5;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
5 rows affected.
```

Out[62]:

	id	host_id	neighborhood_group	neighborhood	latitude	longitude	room_type	price	minimum_nights
0	77765	417504	Brooklyn	Greenpoint	40.73777	-73.95366	Hotel room	308	2
1	2539	2787	Brooklyn	Kensington	40.64529	-73.97238	Private room	299	30
2	45910	204539	Queens	Ridgewood	40.70309	-73.89963	Entire home/apt	425	30
3	45935	204586	Bronx	Mott Haven	40.80635	-73.92201	Private room	60	30
4	45936	867225	Manhattan	Morningside Heights	40.80630	-73.95985	Private room	75	31

In [63]:
```
%%sql features_df <<
SELECT
    minimum_nights,
    number_of_reviews,
    total_host_listings,
    availability_365,
    reviews_in_last_yr,
    price
FROM
    airbnb.listings;
```

```
 * mysql+pymysql://root:***@localhost/airbnb
39851 rows affected.
Returning data to local variable features_df
```

In [64]:
```
features_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 39851 entries, 0 to 39850
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   minimum_nights       39851 non-null  int64
 1   number_of_reviews    39851 non-null  int64
 2   total_host_listings  39851 non-null  int64
 3   availability_365     39851 non-null  int64
 4   reviews_in_last_yr   39851 non-null  int64
 5   price                39851 non-null  int64
dtypes: int64(6)
memory usage: 1.8 MB
```

In [65]:
```
features_df.describe()
```

Out[65]:

	minimum_nights	number_of_reviews	total_host_listings	availability_365	reviews_in_last_yr	price
count	39851.000000	39851.000000	39851.000000	39851.000000	39851.000000	39851.000000
mean	19.138842	26.696218	16.943188	131.634689	7.754360	197.695942
std	31.441428	56.268948	59.618002	138.544503	18.786123	353.423923
min	1.000000	0.000000	1.000000	0.000000	0.000000	10.000000
25%	2.000000	1.000000	1.000000	0.000000	0.000000	80.000000
50%	14.000000	5.000000	1.000000	76.000000	1.000000	130.000000
75%	30.000000	25.000000	4.000000	277.000000	7.000000	219.000000
max	1250.000000	1480.000000	453.000000	365.000000	949.000000	16500.000000

In [66]:
```
features_df.corr()["price"].to_frame().sort_values(by="price", ascending=False)[1:]
```

Out[66]:

	price
availability_365	0.095126
total_host_listings	0.042657
reviews_in_last_yr	-0.002460
number_of_reviews	-0.032753
minimum_nights	-0.035438

In [67]:
```
del features_df
```

Price not significantly correlated to:

- Reviews
  - Number of reviews
  - Reviews in the last 12 months
- Minimum number of nights
- Availability over the next 12 months

Takeaways

Overall

- 70.83% of all listings are priced between 0 and 199.00 dollars
- Listings have an average price of 197.70 and a median price of 130.00
- Prices are skewed to the high side

Location

- Manhattan & Brooklyn are #1 and #2, respectively, in average listings price
- 79.4% of all listings are located in Manhattan and Brooklyn, respectively.
- 75.7% of all listings with any availability over the next year are located in Manhattan and Brooklyn, respectively.
- 86.1% of all listings with no availabilities over the next year are in Manhattan and Brooklyn.

Room Type

- Entire home/apt and Private room account for 98.2% of all listings
- Hotel rooms have the highest average prices.
- Entire Home/Apts have the second highest average price, more than Private and Shared rooms combined.

Reviews

- No significant correlation with price

Recommendations

1. Further Analyze Location

We should gather more data related to 'neighborhood_group' locations such as crime data and proximity to cultural or sporting venues to better understand what drives location's influence on price. The data required is public so it can be easily collected and at minimal cost. The collection and analysis can be completed in two weeks.

2. Feature Selection

A machine learning model will be required to supply our clients with a rental price suggestion. To this end, feature selection and normalization will have to be planned and executed. The features will include neighborhood, neighborhood group, and room type. Selection and standardization can be accomplished with in a week.

3. ML Model Development

Suggestion #2 is a prerequisite. Model selection, testing, and deployment will take approximately four weeks not to include regular testing and refactoring as new data becomes available.