Data Mining Assignment **Sayeh Sobhani**

Student no. 99422102

Assignment 1.1

In this assignment we are supposed to analyze two datasets. The first dataset is from the NY Airbnb data and the second one is about international football results.

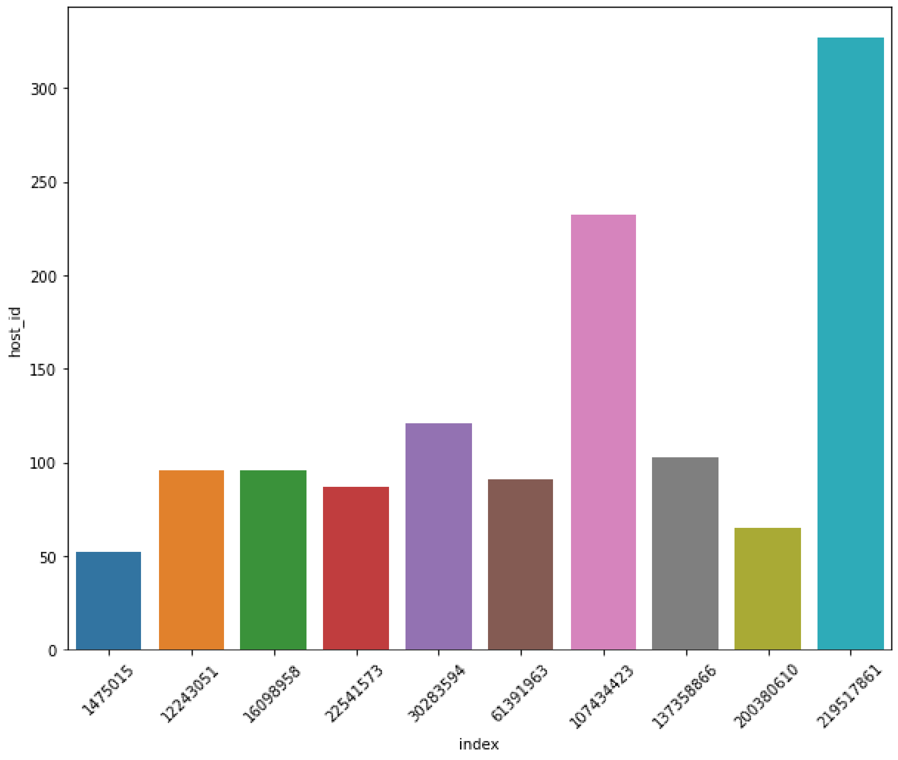
In the first dataset (NY):

We can see that the null data for 'last_review' and 'reviews_per_month' columns are common. And their corresponding 'number_of_reviews' is also zero. So, that means there are no reviews for that instance at all, hence can be filled with zero.

We can also see that there are no duplicate instances in the data.

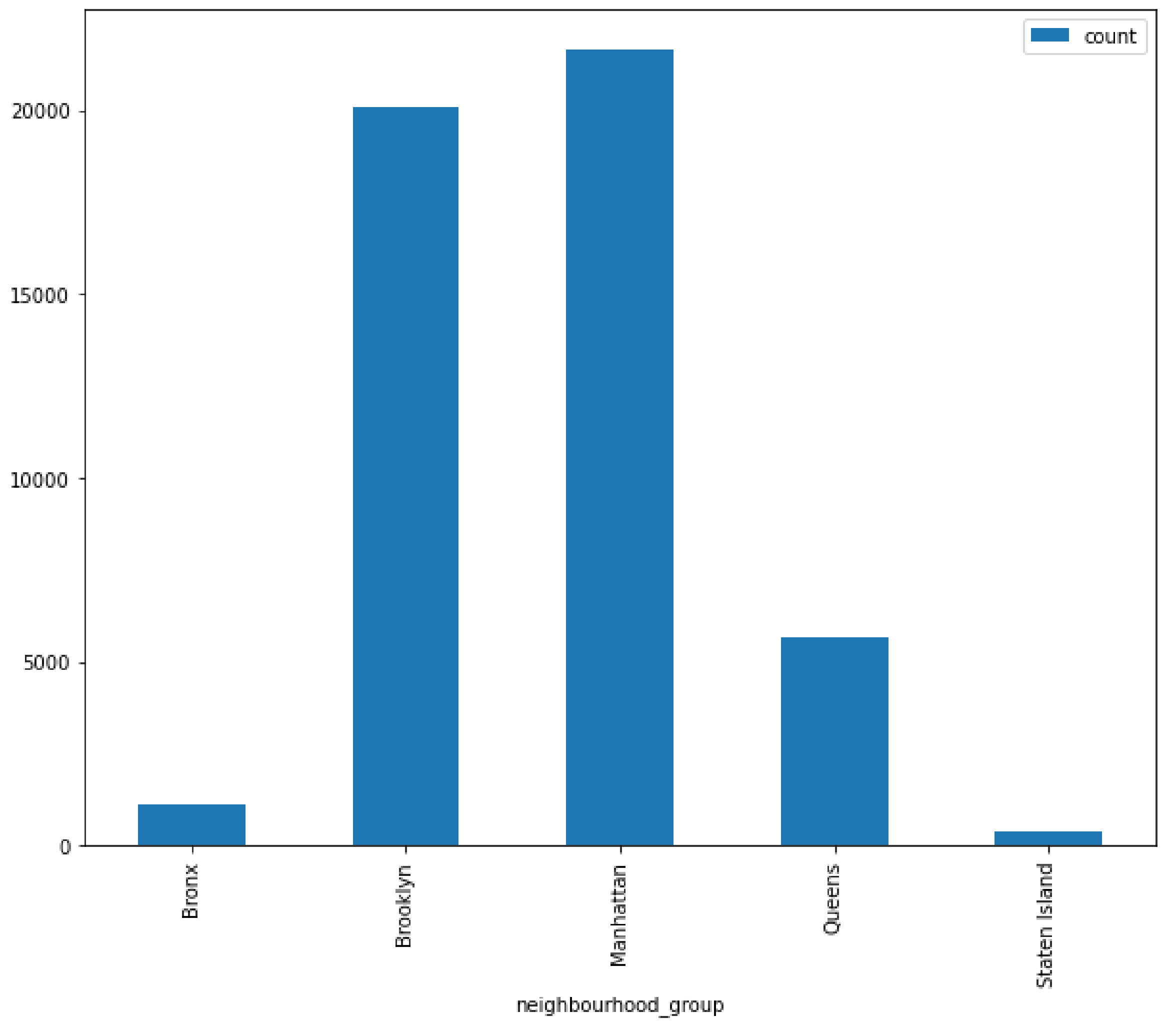
I cleaned some of the columns I didn't find useful, like the id column, and the rows with price of zero.

To get a better understanding of which hosts have the most dedicated datapoints to, I inspect this plot:

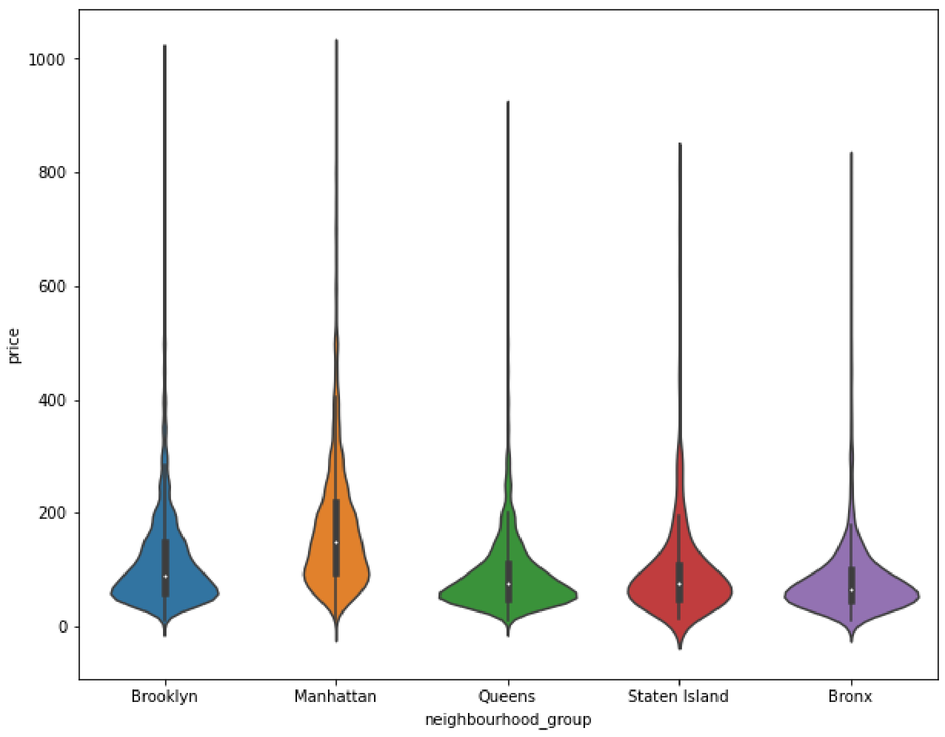


The host with the id no. 219517861 has 327 indexes dedicated to it. The properties of this host are in Manhattan which is a hot spot.

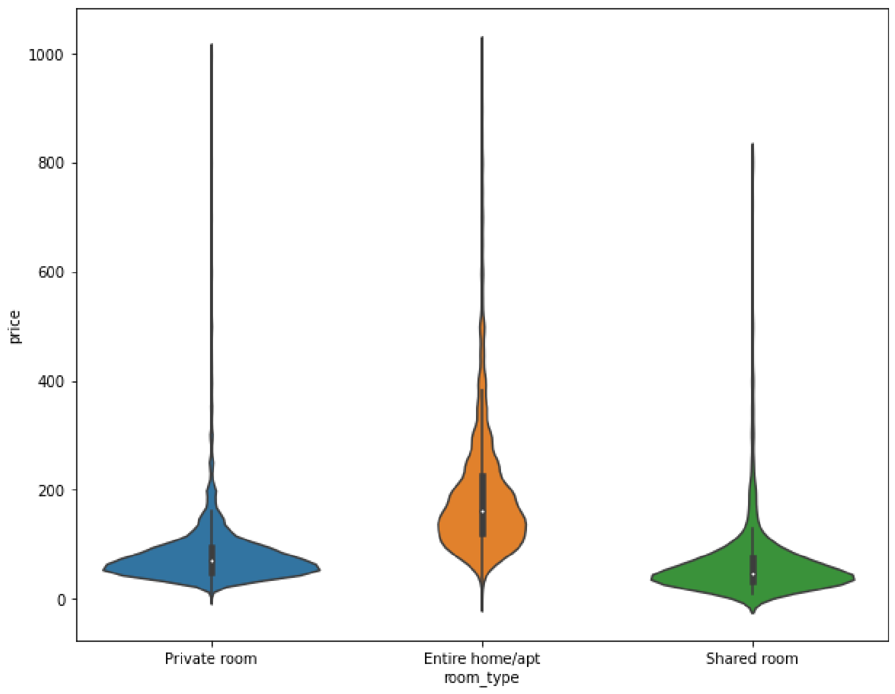
To get a better understanding of the neighborhood data count, lets take a look at this plot.



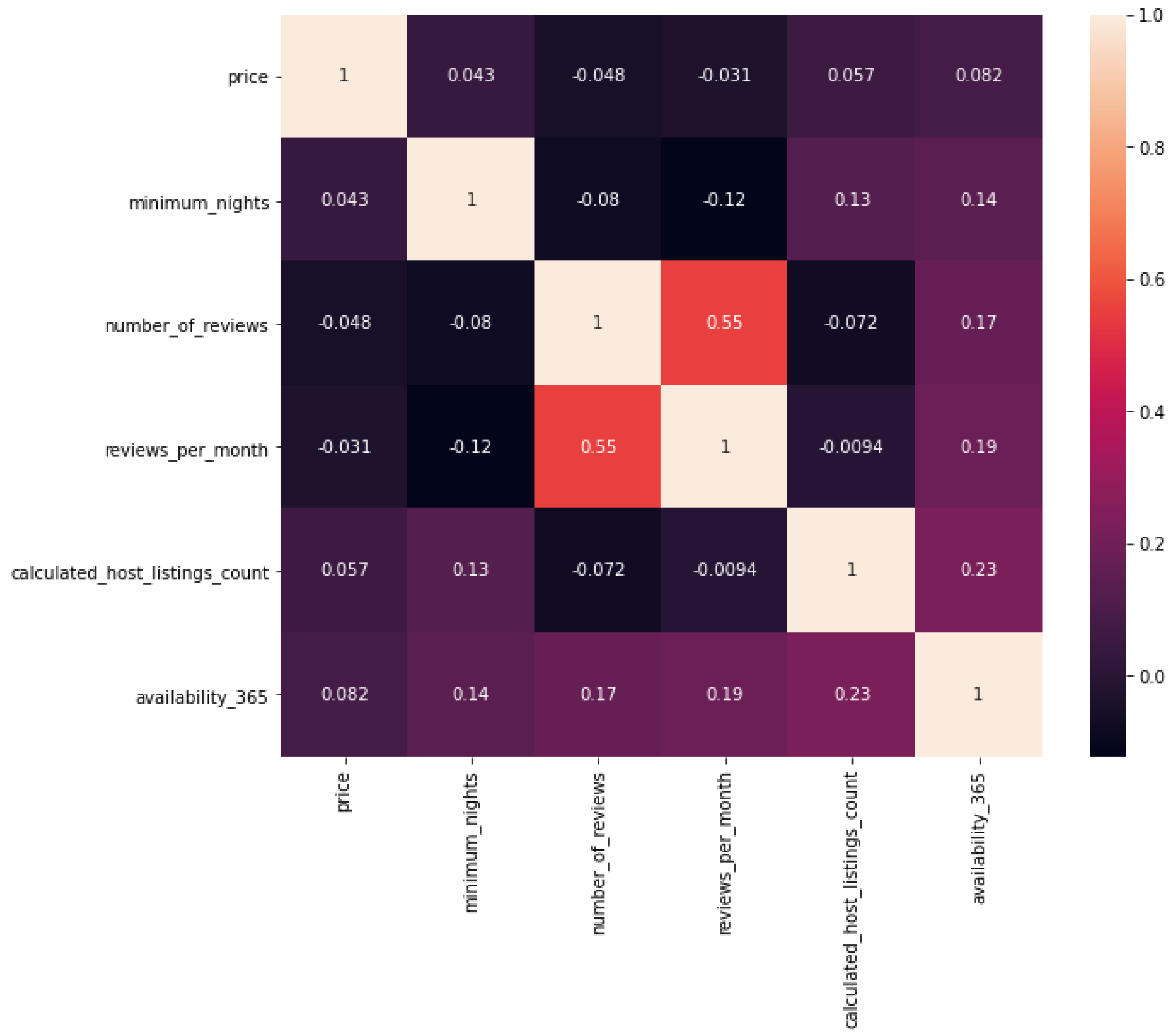
As can be seen in the next plot Clearly Manhattan has higher prices followed by Brooklyn.



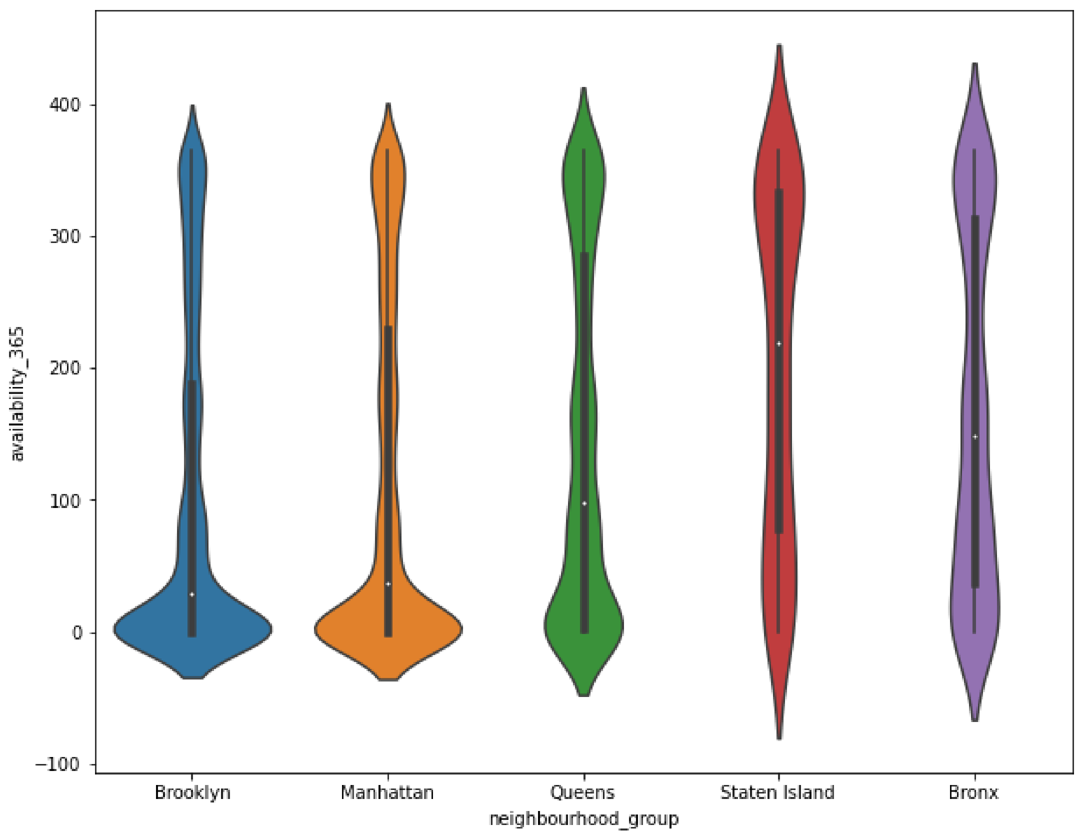
To inspect the room type and its correlation with the price, if we inspect this plot we can get a better understanding that prices of Entire home/apt is more and shared rooms have less cost, which is expected.



I investigate the correlation between 'price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availability_365 with the correlation heatmap plot. reviews_per_month and number_of_reviews have high correlation, which was expected

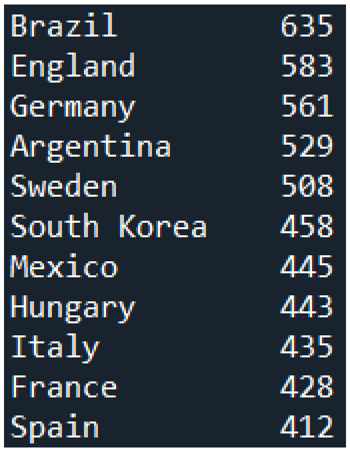


The availability 365 days were inspected for different neighborhoods and as a result, Staten Island has more availability followed by Bronx in a year. This is expected as Manhattan and Brooklyn have more population and has tourist spots.



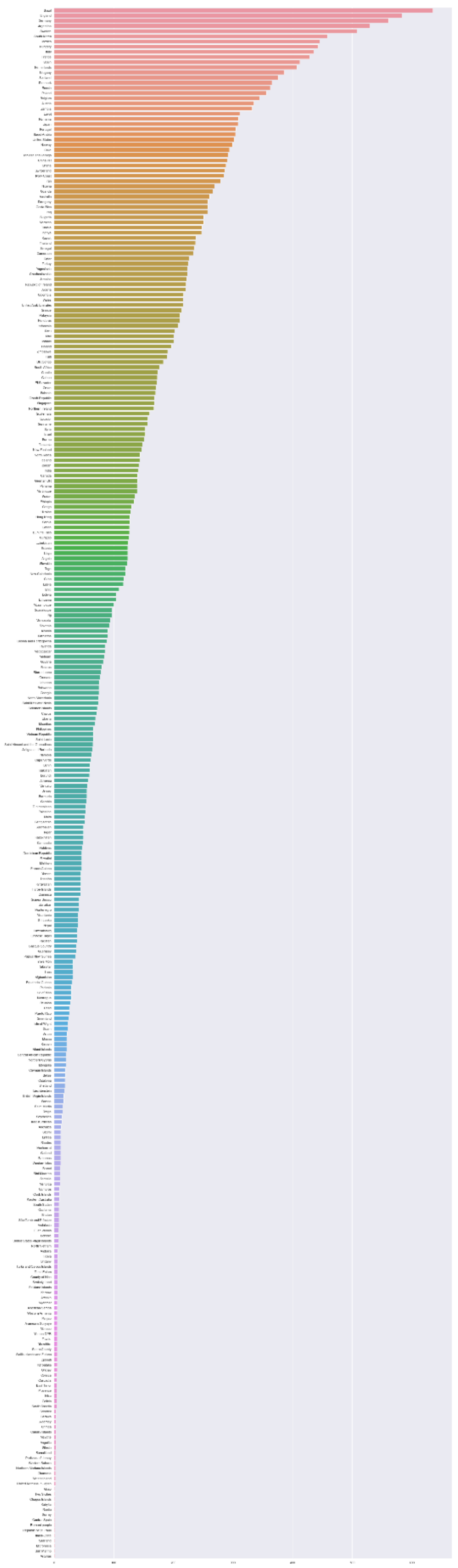
To analyze the second dataset, I started by adding two columns of winner and loser to the dataset to initialize the results of each match.

The number of most won matches are by these 11 first teams between 300 teams in this dataset:



```
Brazil         635
England        583
Germany        561
Argentina      529
Sweden         508
South Korea    458
Mexico         445
Hungary        443
Italy          435
France         428
Spain          412
```

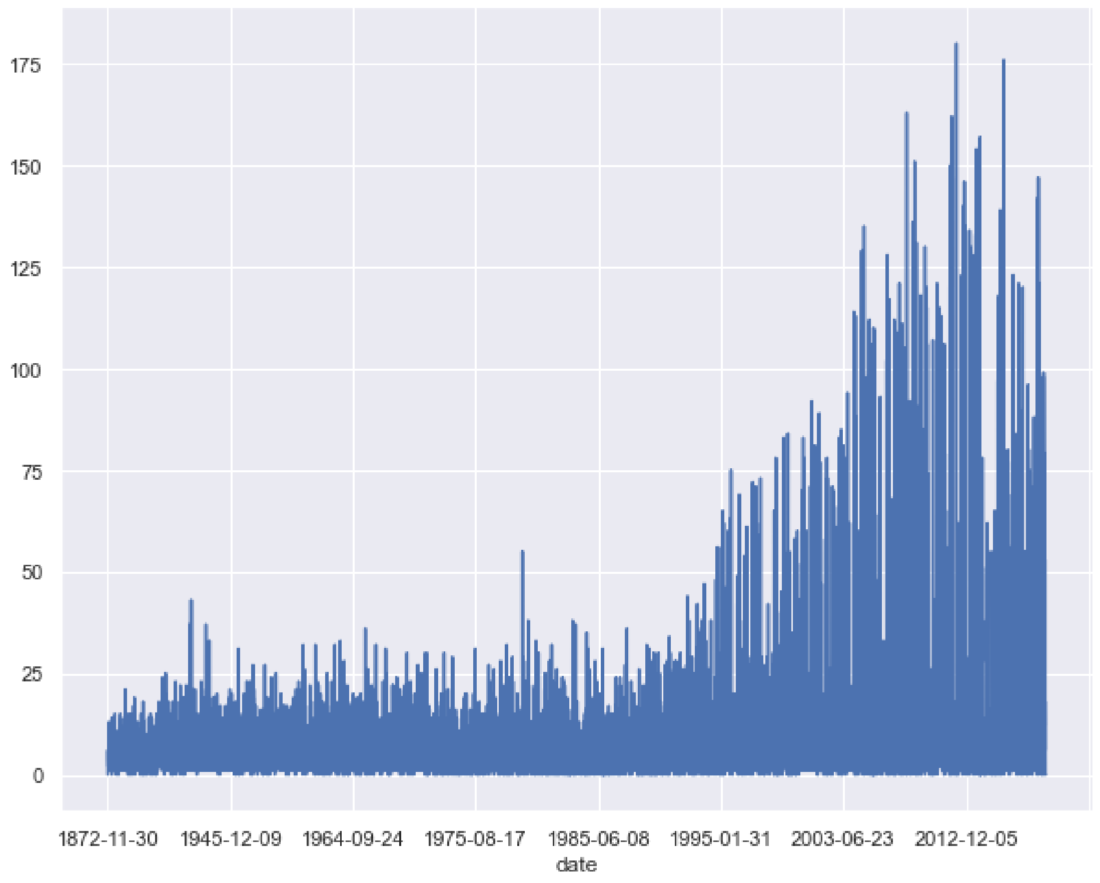
To get a better view of which teams won how many matches, let's take a look at the winners plot:



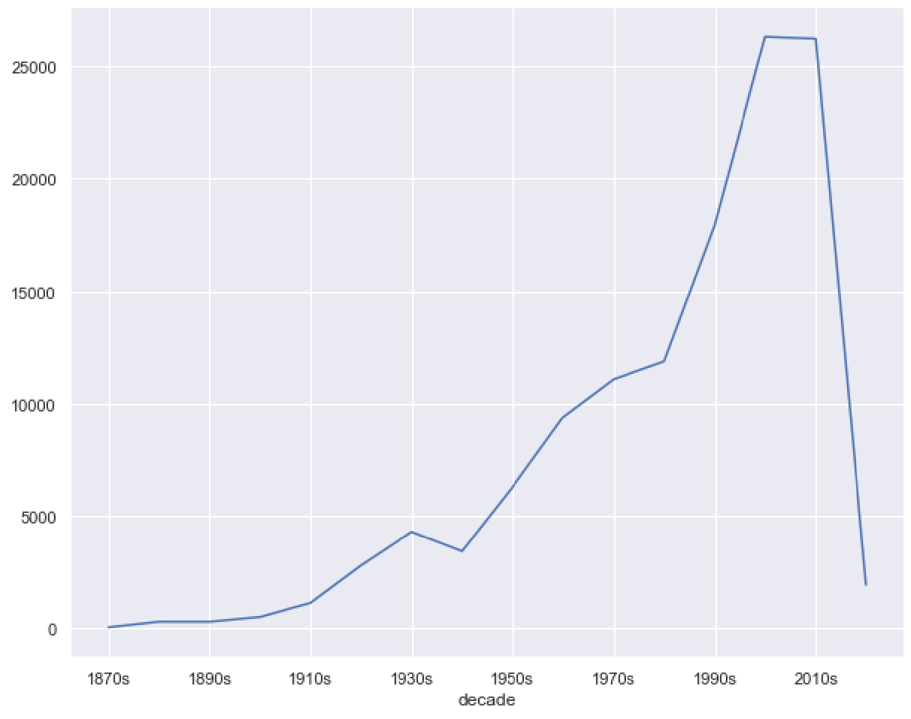
To make the interpretability of dates easier, I added a year scale timeline that divides dates of games into old, middle, modern matches.

And also I converted the date column for easier use

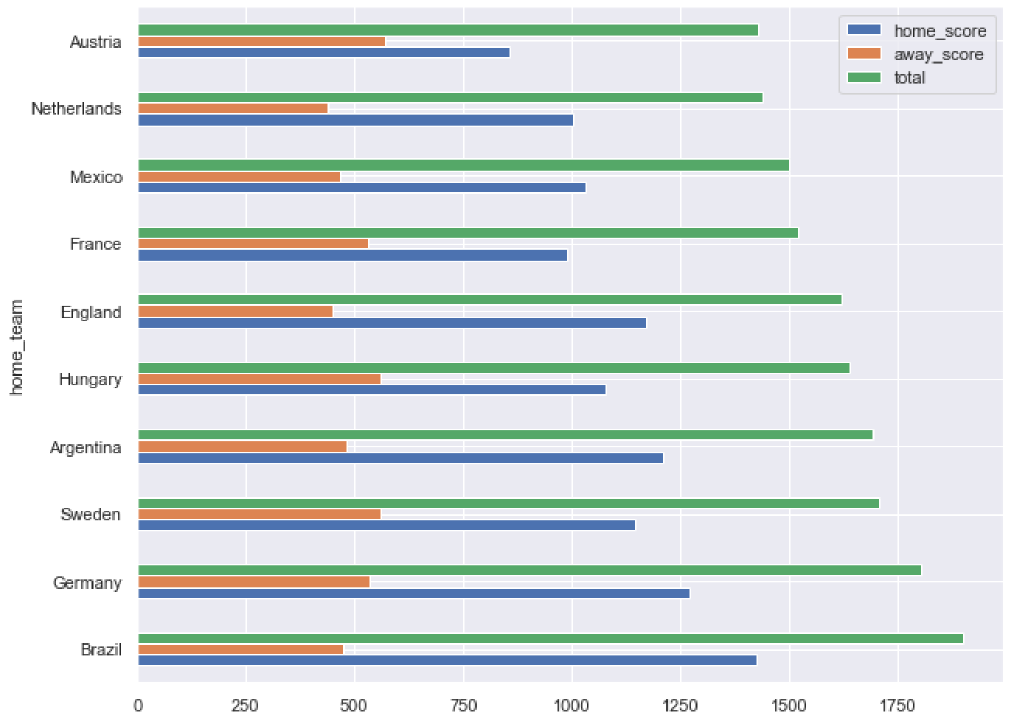
I added a Goal column and to understand the number of goals overtime, we can see this plot. This could also be interpreted into the increasing number of footbal matches over time



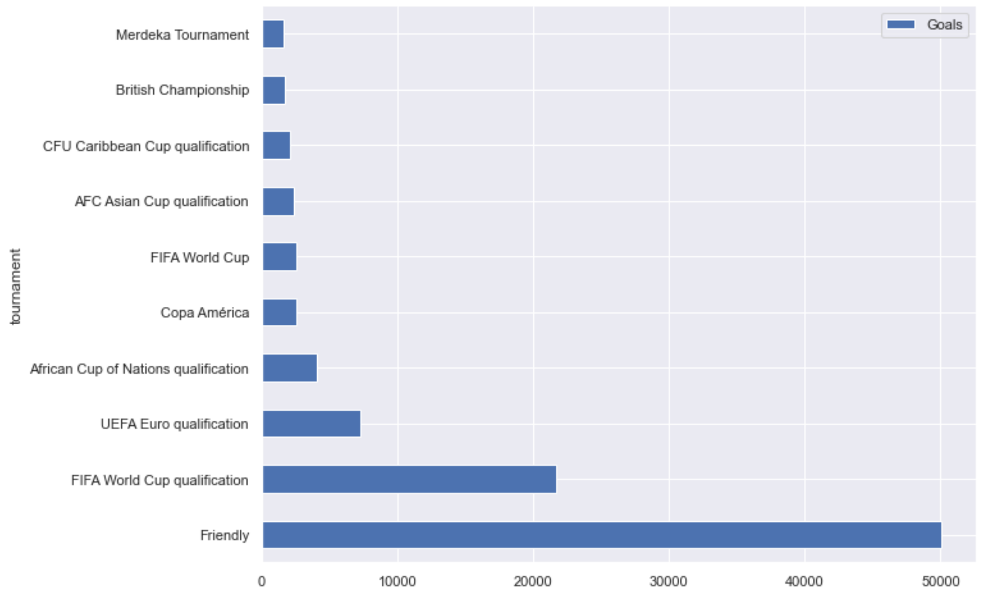
For better undertanding, lets look at a better plot of goals in decades



Another way to look at the data we have is to inspect home team scores:



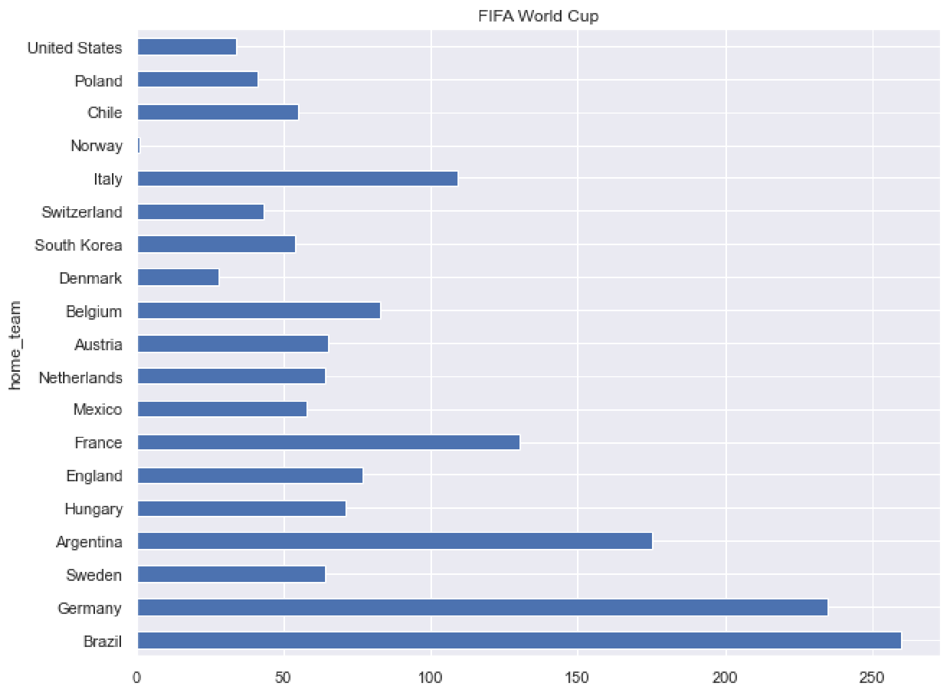
To see the number of goals in each tonament, we can take a look at this plot:



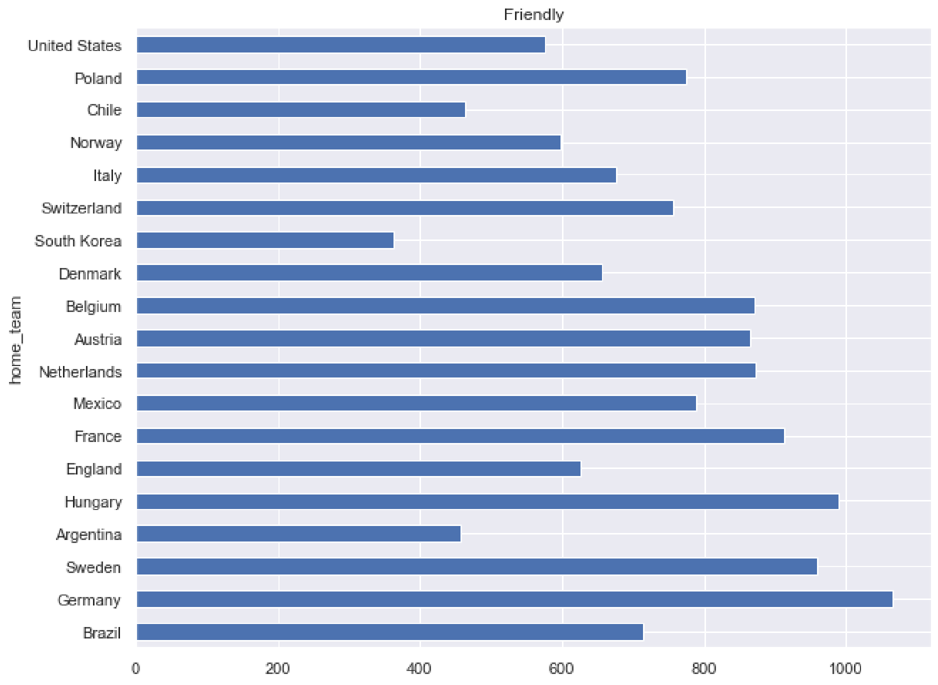
We can see that the number of goals in friendly tournamets are the highest overal.

To get into more detail about how each team did in some of these torunaments:

Based on this plot, Brazil had the most number of goals overal In the FIFA.



Meanwile in friendly tournamets



Germany has the most number of goals in friendly tournamets while brazil does poorly. Which could be interpreted, since Brazil was on top of the list in other plots, that FIFA games results might play a more significant role for teams.

References

1)Videos and codes that Mr. Sharifi shared

2)Kaggle.com

3)Stackoverflow.com

4)Medium.com

5)Pandas’ documents

6)Matplotlib documents

Appendix

```
# -*- coding: utf-8 -*-
"""
@author: Sayeh
Spyder editor
"""

#%%dataset and package import
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import normaltest
import plotly.express as px

df1 = pd.read_csv(r'C:\Users\Asus\Desktop\Data Mining\01_assignment\1.1\NY
airbnb\AB_NYC_2019.csv')
df2 = pd.read_csv(r'C:\Users\Asus\Desktop\Data Mining\01_assignment\1.1\International
footbal results\results.csv')

#%% understanding and cleaning and processing the dataset 1

#to get a better understanding of how our data looks we look at the first 10 rows
df1.shape
df1.head(10)

#lets see some info about this dataset and its features
df1.info()

df1.describe()

df1 = df1.dropna(subset=['name'])
df1 = df1.drop(columns=['host_name'])
#df1 = df1.drop(columns=['id'])
#numeric data info
df1._get_numeric_data().describe()

#null managing
df1.isna().sum()

#no data duplications
df1.duplicated().sum()

#the review null data
null_data = df1[df1['reviews_per_month'].isnull()]
null_data.head()
null_data.shape
null_data['number_of_reviews'].value_counts()
df1 = df1.drop(columns=['last_review'])

#cleaning prices = zero
```

```python
df1 = df1[df1['price']>0]
df1.head()
df1['price'].describe()

#to get a better understanding of data in each column
for cols in df1.columns:
    #if df1[cols].dtype == 'object' or df1[cols].dtype == 'bool':
        print('column : ',cols)
        print(df1[cols].value_counts().head())

# to better understand the host ids data
hosts = df1['host_id'].value_counts().reset_index()[:10]
plt.rcParams['figure.figsize'] = (10,8)
ax = sns.barplot(x='index',y='host_id',data=hosts)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45)
plt.show()

#neighbourhood counts
df1.groupby('neighbourhood_group')['id'].agg(['count']).plot(kind="bar")

#neighbourhood vs the prices
sample_data = df1[df1['price']<1000]
sns.violinplot(x='neighbourhood_group', y='price', data=sample_data)
plt.show()

#Room type inspection vs price
df1['room_type'].value_counts()

sns.violinplot(x='room_type', y='price', data=sample_data)
plt.show()

room = df1.groupby('room_type')['price']
private = room.get_group('Private room')
entire = room.get_group('Entire home/apt')
shared = room.get_group('Shared room')

#correlation of price and
sns.heatmap(df1[['price','minimum_nights','number_of_reviews','reviews_per_month','calculated_host_listings_count','availability_365']].corr(), annot=True)
plt.show()
for i in [private,entire,shared]:
    print(normaltest(i))

#availability365 of neighborhoods
sns.violinplot(x='neighbourhood_group',y='availability_365', data=df1)

#End of inspections and investigatingg codes of dataframe 1

#%%lets beggin analyzing the second dataset

df2.shape
df2.head(10)

#lets see some info about this dataset and its features
df2.info()

df2.describe()

#introducing a winner column to dataser
def winner(row):
    if row['home_score'] > row['away_score']: return row['home_team']
    elif row['home_score'] < row['away_score']: return row['away_team']
    else: return 'DRAW'

df2['winner'] = df2.apply(lambda row: winner(row), axis=1)
```

```python
df2.head()

#introducing a loser column to our dataset
def loser(row):
    if row['home_score'] < row['away_score']: return row['home_team']
    elif row['home_score'] > row['away_score']: return row['away_team']
    else: return 'DRAW'

df2['loser'] = df2.apply(lambda row: loser(row), axis=1)
df2.head()

#lets see what the winners look like
winners = pd.value_counts(df2.winner)
winners = winners.drop('DRAW')
winners.head(11)

#a plot for the winners
fig, ax = plt.subplots(figsize=(25, 100))
sns.set(font_scale=1)
sns.barplot(y = winners.index.tolist(), x = winners.tolist())

#making a yearscale of matches from old to new
df2 = df2.dropna(axis=0)
for i in range(len(df2)):
    if df2.iat[i, 0].find('-') != -1:
        df2.iat[i, 0] = int(df2.iat[i, 0][:str(df2.iat[i,0]).find('-')])
df2['Year Scale'] = df2['date'].apply(lambda x:(('Old Match(<1976)' , 'Middling Match(1976<.<2000)')[x > 1976],'Modern Match(>2000)')[x > 2000])

#changing dates a bit, adding years and decades column
def give_date(x):
    t = pd.to_datetime(x['date'])
    x['year'] = t.year
    x['month'] = t.month
    x['day_of_week'] = t.dayofweek
    return x

df2 = df2.apply(give_date, axis = 1)

decades = 10 * (df2['year'] // 10)
decades = decades.astype('str') + 's'
df2['decade'] = decades
df2.head()
#
fig = px.scatter(df2, x='home_team', y='away_team',color='tournament',height=1700)
fig.show()

#
x = df2.groupby('home_team')[['home_score','away_score']].sum()
x['total'] = x['home_score'] + x['away_score']
x = x.sort_values('total', ascending = False)
x = x[:10]
x.plot(kind = 'barh')

#goals in years
df2['Goals'] = df2['home_score']+df2['away_score']
x = df2.groupby('decade')['Goals'].sum()
print(x)
x.plot()

#tournamets and goals
x = df2.pivot_table('Goals', index = 'tournament', aggfunc = 'sum')
x = x.sort_values('Goals', ascending  = False)
x = x[:10]
print(x)
```

```
x.plot(kind = 'barh')

#FIFA cup and team performance
x = df2.pivot_table('Goals', index = 'home_team', columns = 'tournament', aggfunc = 'sum', fill_value = 0, margins = True, margins_name = 'Total')
x = x.sort_values('Total', ascending = False)
x = x[:20]
x['FIFA World Cup'][1:].plot(kind = 'barh', title = 'FIFA World Cup')

#Friendly matches and teams
x['Friendly'][1:].plot(kind = 'barh', title = 'Friendly')

```