Project 2 - Bit Coin Prices - Part 4 - Perform Data Analysis

Dataset Link: https://www.kaggle.com/datasets/chakradharmattapalli/bitcoin-prices

Importing Necessary Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
Matplotlib is building the font cache; this may take a moment.
```

Import and read dataset

```
#Write Your Code Here
df=pd.read_csv('bitcoin_prices2.csv')
df.head()
```

	open	high	low	close	tick_volume	year	month	day
0	5.26	5.47	4.80	5.21	69150	2012	1	2
1	5.22	5.29	4.65	4.88	125170	2012	1	3
2	4.88	5.70	4.75	5.57	131170	2012	1	4
3	5.57	7.22	5.57	6.94	182328	2012	1	5
4	6.95	7.21	6.13	6.70	218077	2012	1	6

Data Cleaning

a. Missing Value

```
#Write Your Code Here
df.isnull().sum()
```

```
open           0
high           0
low            0
close          0
tick_volume    0
year           0
month          0
day            0
dtype: int64
```

b. Duplicate data

```
#Write Your Code Here
df.duplicated().sum()
```

```
0
```

Data Analysis

1. What was the average closing price of the stock for each month in the year?

```
#write your code here
df['date'] = pd.to_datetime(df[['year', 'month', 'day']])

# group data by month and calculate average closing price for each
month
avg_close_price = df.groupby(df['date'].dt.strftime('%Y-%m'))
['close'].mean()

# print resulting series
avg_close_price
```

```
date
2012-01        6.051818
```

```
2012-02        5.100952
2012-03        4.916364
2012-04        5.002857
2012-05        5.078261
               ...
2020-08    11631.657143
2020-09    10684.284545
2020-10    11901.793636
2020-11    16650.635238
2020-12    21788.898182
Name: close, Length: 108, dtype: float64
```

1. What was the average daily price range (high - low) for a given month and year?

```
#write your code here
year = 2020
month = 1

# filter data by year and month
data_filtered = df[(df['year'] == year) & (df['month'] == month)]

# calculate daily price range and store it in new 'price_range' column
data_filtered['price_range'] = data_filtered['high'] -
data_filtered['low']

# calculate average daily price range for selected month and year
avg_price_range = data_filtered['price_range'].mean()

# print resulting average daily price range
print(f'The average daily price range for {month}/{year} was:
{avg_price_range}')
```

```
The average daily price range for 1/2020 was: 406.90636363636366

C:\Users\HP\AppData\Local\Temp\ipykernel_11544\1296620740.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data_filtered['price_range'] = data_filtered['high'] -
data_filtered['low']
```

1. What was the total tick volume for each year in the dataset?

```
#Write your code here
yearly_tick_volume = df.groupby('year')['tick_volume'].sum()
yearly_tick_volume
```

```
year
2012     16069420
2013    344894281
2014      2400003
2015      2384980
2016      4923692
2017    339002469
2018    558906751
2019    415228459
2020    478490502
Name: tick_volume, dtype: int64
```

Data Visualization

Q1. groupby dataframe on year and find the mean value,

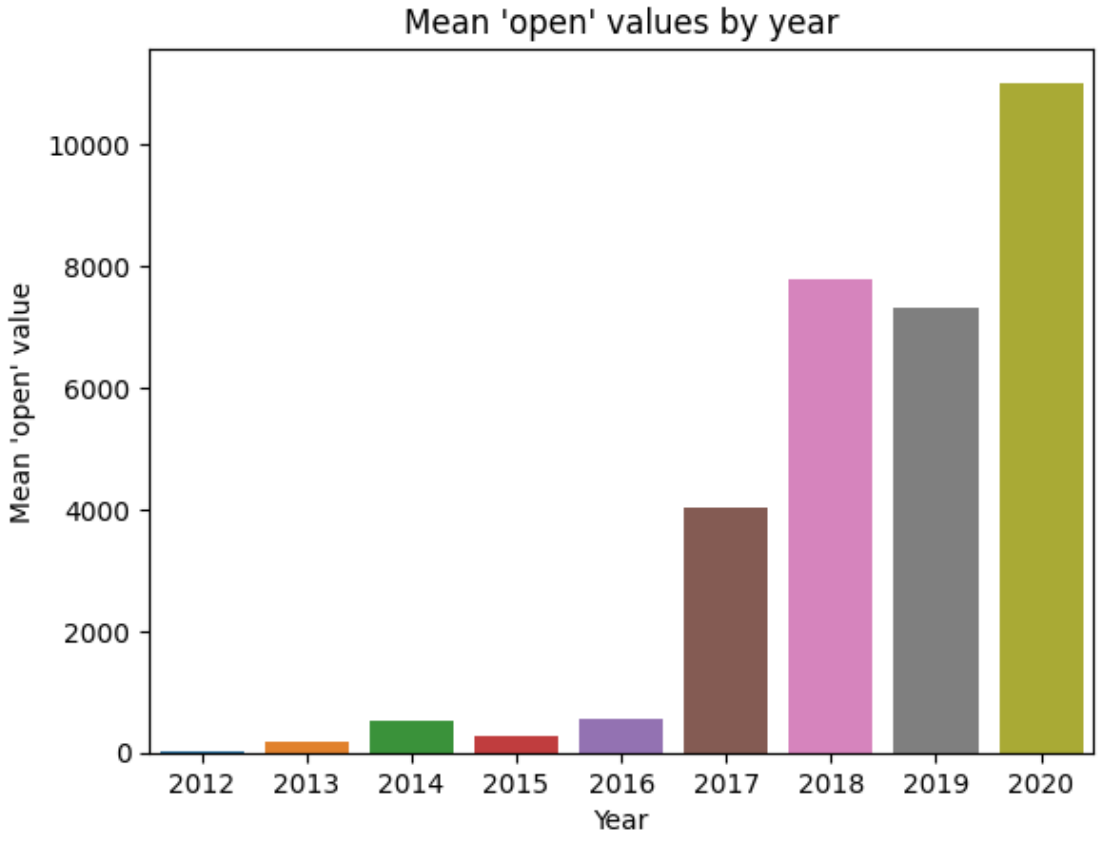
```
# Write your code here
# Group the data by the 'year' column and calculate the mean value for
each column
df_mean = df.groupby('year', as_index=False).mean()

# Print the resulting DataFrame
print(df_mean)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_11544\2155066857.py:3:
FutureWarning: The default value of numeric_only in
DataFrameGroupBy.mean is deprecated. In a future version, numeric_only
will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df_mean = df.groupby('year', as_index=False).mean()

   year          open          high           low         close
tick_volume  \
0  2012      8.249119      8.482605      8.034674      8.303602
6.156866e+04
1  2013    182.396475    195.594023    170.112989    185.867663
1.321434e+06
2  2014    520.448123    534.720881    502.673027    518.500077
9.195414e+03
3  2015    268.885862    275.916628    263.178314    269.679579
9.137854e+03
4  2016    560.208467    568.469004    552.945556    561.547548
1.886472e+04
5  2017   4030.433980   4208.879728   3878.672007   4072.584116
1.153070e+06
6  2018   7784.464930   8041.540000   7478.756503   7755.060839
1.954219e+06
7  2019   7328.262819   7556.984942   7115.426564   7352.219653
1.603199e+06
8  2020  11020.477846  11335.288615  10744.427808  11104.194962
1.840348e+06
```

```
      month        day
0  6.501916  15.819923
1  6.513410  15.704981
2  6.524904  15.750958
3  6.540230  15.685824
4  6.524904  15.708812
5  6.731293  15.792517
6  6.094406  15.667832
7  6.513514  15.725869
8  6.519231  15.811538
```

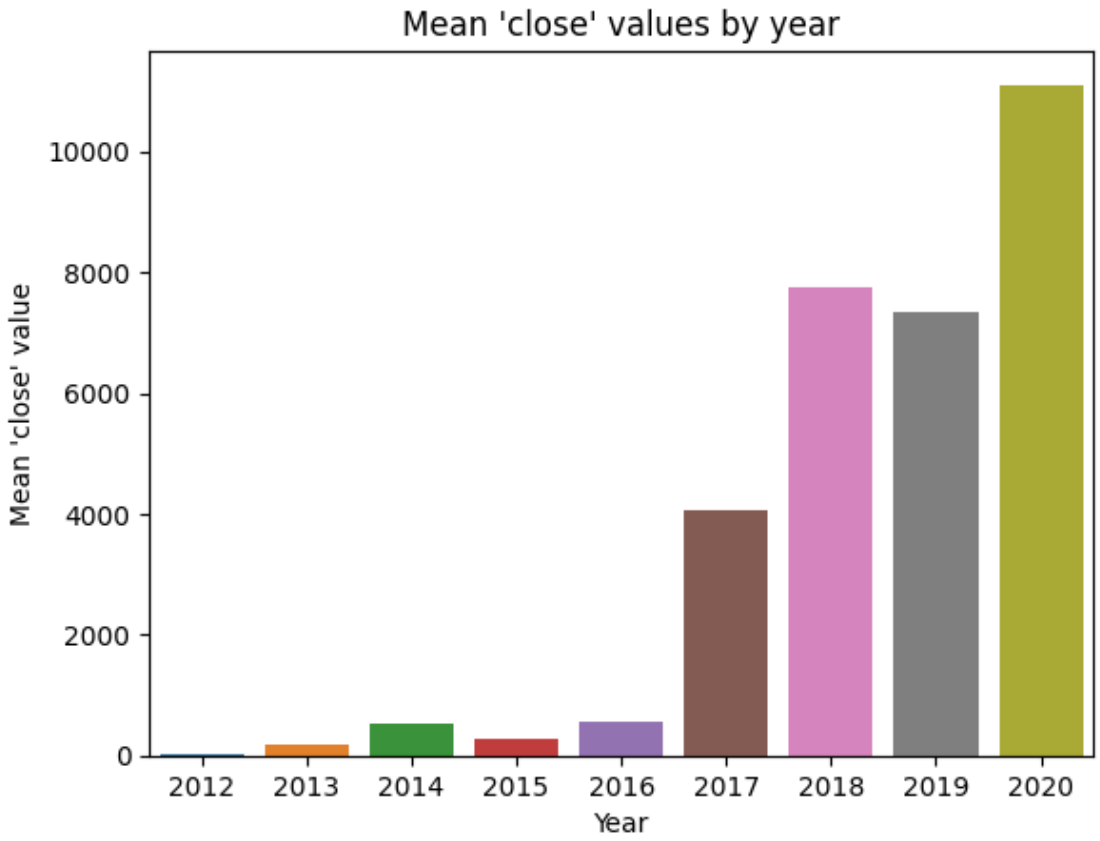
Q2. Visualize mean value of open yearly.

```
sns.barplot(x='year',y='open',data=df_mean)

plt.title("Mean 'open' values by year")
plt.xlabel("Year")
plt.ylabel("Mean 'open' value")

# Display the plot
plt.show()
```



Q3. Visualize mean value of close yearly.

```
# Write your code here

sns.barplot(x='year',y='close',data=df_mean)

plt.title("Mean 'close' values by year")
plt.xlabel("Year")
plt.ylabel("Mean 'close' value")

# Display the plot
plt.show()
```



Q4. groupby dataframe on year and find the median value.

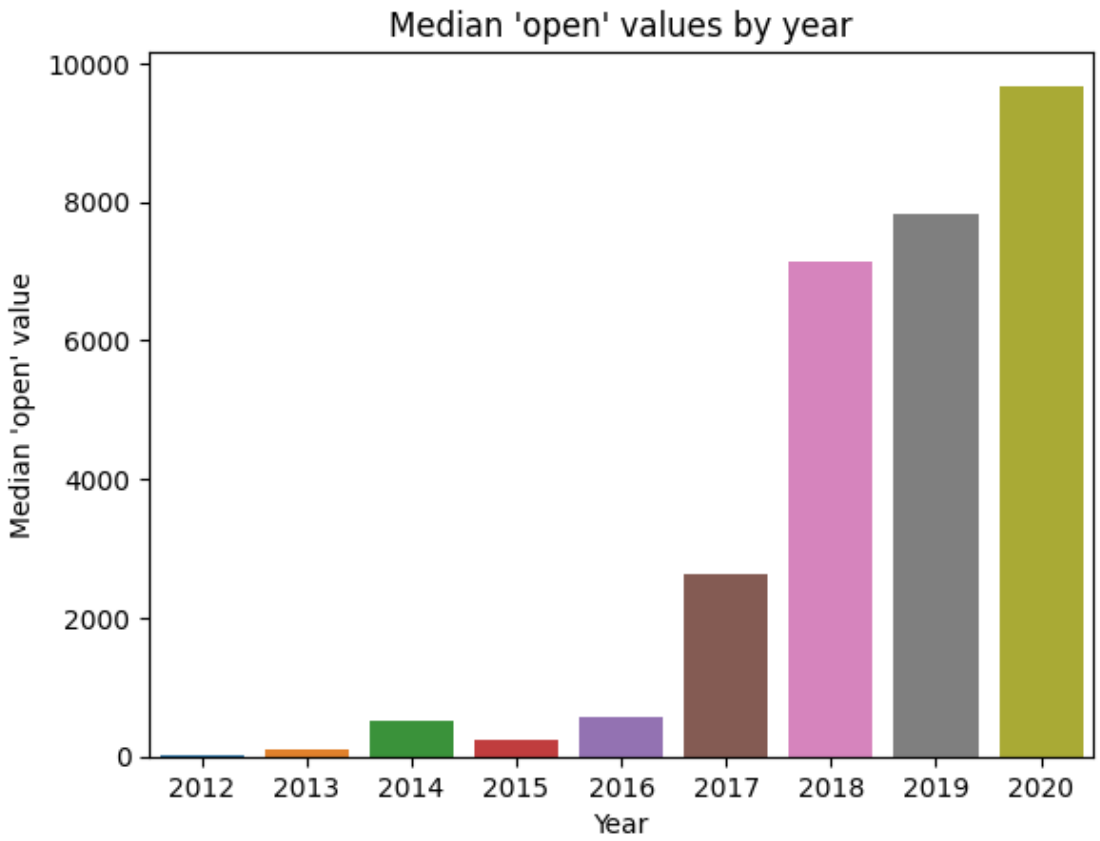
```
# Write your code here
df_median = df.groupby('year', as_index=False).median()

# Print the resulting DataFrame
print(df_median)
```

```
   year      open     high       low     close  tick_volume  month
day
0  2012     6.800     7.10     6.510     6.700      46830.0    7.0
16.0
1  2013   111.300   116.44   106.000   112.250      17966.0    7.0
```

```
16.0
2  2014   501.210   517.00   477.300   501.220       8073.0    7.0
16.0
3  2015   246.000   251.99   241.090   245.970       7811.0    7.0
16.0
4  2016   577.960   582.05   573.040   576.850      17009.0    7.0
16.0
5  2017  2622.985  2743.55  2547.640  2640.920     445335.0    7.0
16.0
6  2018  7138.750  7421.45  6868.370  7148.500    1278851.5    6.0
16.0
7  2019  7815.950  8115.60  7483.790  7831.450    1302461.0    7.0
16.0
8  2020  9672.340  9887.64  9467.515  9683.635    1752751.0    7.0
16.0

C:\Users\HP\AppData\Local\Temp\ipykernel_11544\3941005774.py:2:
FutureWarning: The default value of numeric_only in
DataFrameGroupBy.median is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or
select only columns which should be valid for the function.
  df_median = df.groupby('year', as_index=False).median()
```

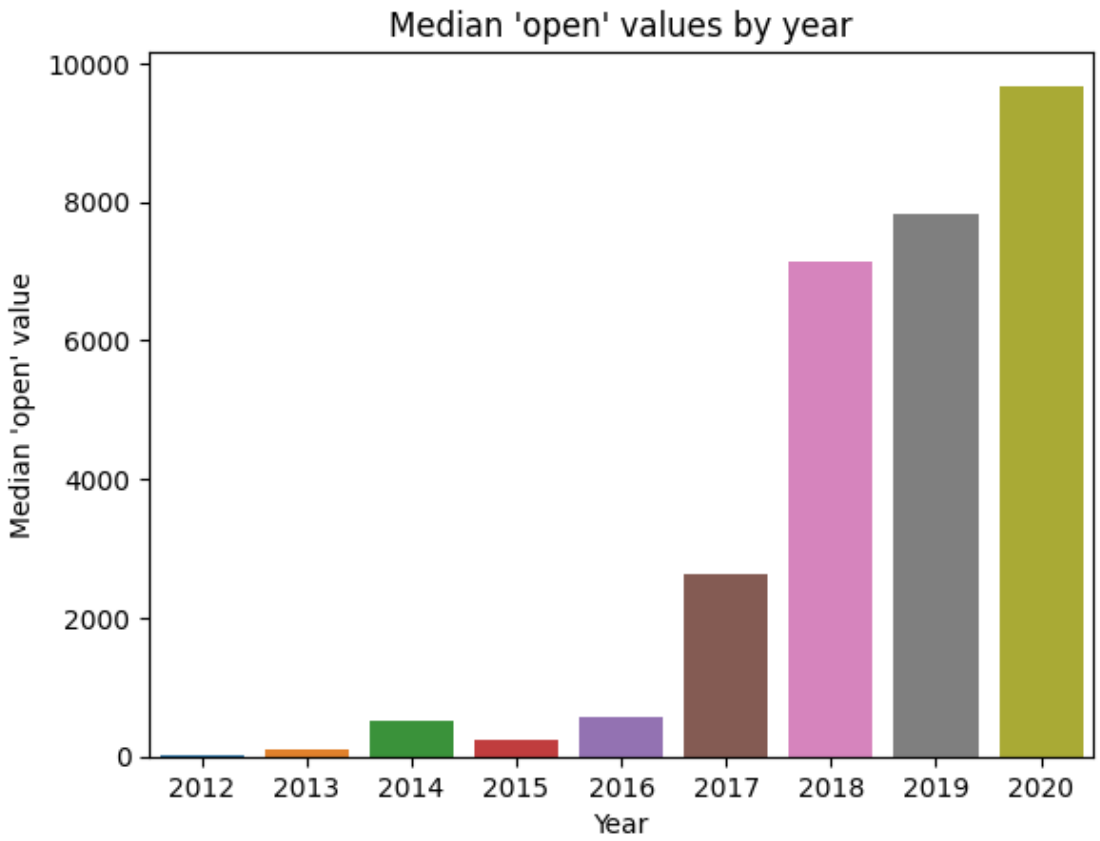
Q5. Visualize median value of open yearly.

```
# Write your code here
sns.barplot(x='year',y='open',data=df_median)

plt.title("Median 'open' values by year")
plt.xlabel("Year")
plt.ylabel("Median 'open' value")

# Display the plot
plt.show()
```



Q6. Visualize median value of close yearly.

```
# Write your code here
sns.barplot(x='year',y='open',data=df_median)

plt.title("Median 'open' values by year")
plt.xlabel("Year")
plt.ylabel("Median 'open' value")

# Display the plot
plt.show()
```


Median 'open' values by year
Median 'open' value
10000
8000
6000
4000
2000
0
2012
2013
2014
2015
2016
2017
2018
2019
2020
Year