DEVMiniProjectPhase- 1

May 3, 2024

Colab Link: MiniProject

1 DEV Mini Project

1.0.1 Tracking Cryptocurrency prices over time

- Dhruv Ojha - 60009220179
- Aanshuvi Shah - 60009220168
- Akash Upadhyay - 60009220182

[54]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[55]:
```
df=pd.read_csv("/content/btc.csv")
df.head()
```

[55]:

	date	txVolume(USD)	adjustedTxVolume(USD)	txCount	marketcap(USD)
0	2009-01-03	NaN	NaN	NaN	NaN
1	2009-01-04	NaN	NaN	NaN	NaN
2	2009-01-05	NaN	NaN	NaN	NaN
3	2009-01-06	NaN	NaN	NaN	NaN
4	2009-01-07	NaN	NaN	NaN	NaN

	price(USD)	exchangeVolume(USD)	realizedCap(USD)	generatedCoins	fees
0	NaN	NaN	0.0	NaN	NaN
1	NaN	NaN	0.0	NaN	NaN
2	NaN	NaN	0.0	NaN	NaN
3	NaN	NaN	0.0	NaN	NaN
4	NaN	NaN	0.0	NaN	NaN

	activeAddresses	averageDifficulty	paymentCount	medianTxValue(USD)
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN

	medianFee	blockSize	blockCount

0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	NaN

[56]:
```
df.isnull().sum()/len(df)
#dropping none because the no ratio is >0.5
```

[56]:
```
date                    0.000000
txVolume(USD)           0.409990
adjustedTxVolume(USD)   0.409990
txCount                 0.001561
marketcap(USD)          0.409990
price(USD)              0.409990
exchangeVolume(USD)     0.409990
realizedCap(USD)        0.000000
generatedCoins          0.001561
fees                    0.001561
activeAddresses         0.001561
averageDifficulty       0.001561
paymentCount            0.001561
medianTxValue(USD)      0.409990
medianFee               0.067638
blockSize               0.001561
blockCount              0.001561
dtype: float64
```

[57]:
```
df[['txVolume(USD)','adjustedTxVolume(USD)','marketcap(USD)','price(USD)','exchangeVolume(USD)
  ↪describe(include='all')
```

[57]:

	txVolume(USD)	adjustedTxVolume(USD)	marketcap(USD)	price(USD)	\
count	2.268000e+03	2.268000e+03	2.268000e+03	2268.000000	
mean	2.949397e+09	1.256045e+09	4.392693e+10	2616.874400	
std	5.068949e+09	1.947619e+09	6.027239e+10	3506.035835	
min	3.287234e+07	2.053871e+07	7.794864e+08	68.500000	
25%	2.709126e+08	1.111571e+08	4.883377e+09	357.387500	
50%	9.655342e+08	3.406209e+08	9.632702e+09	644.385000	
75%	3.770078e+09	1.920227e+09	7.018244e+10	4079.200000	
max	4.835307e+10	1.624809e+10	3.261873e+11	19475.800000	

	exchangeVolume(USD)	medianTxValue(USD)
count	2.268000e+03	2268.000000
mean	2.639195e+09	169.082232
std	5.322340e+09	325.665238
min	0.000000e+00	0.024145
25%	2.065660e+07	52.141489

50%	7.277100e+07	77.244383
75%	3.716645e+09	172.271704
max	4.510573e+10	4176.310572

[58]:
```
df.fillna({'txVolume(USD)':df['txVolume(USD)'].median()},inplace=True)
df.fillna({'adjustedTxVolume(USD)':df['adjustedTxVolume(USD)'].
  ↪median()},inplace=True)
df.fillna({'marketcap(USD)':df['marketcap(USD)'].median()},inplace=True)
df.fillna({'exchangeVolume(USD)':df['exchangeVolume(USD)'].
  ↪median()},inplace=True)
df.fillna({'medianTxValue(USD)':df['medianTxValue(USD)'].median()},inplace=True)
df.fillna({'price(USD)':df['price(USD)'].median()},inplace=True)
```

[59]:
```
df.isnull().sum()
```

[59]:
```
date                    0
txVolume(USD)           0
adjustedTxVolume(USD)   0
txCount                 6
marketcap(USD)          0
price(USD)              0
exchangeVolume(USD)     0
realizedCap(USD)        0
generatedCoins          6
fees                    6
activeAddresses         6
averageDifficulty       6
paymentCount            6
medianTxValue(USD)      0
medianFee             260
blockSize               6
blockCount              6
dtype: int64
```

[60]:
```
df[['txCount','generatedCoins','fees','activeAddresses','averageDifficulty','paymentCount','bl
  ↪describe(include='all')
```

[60]:

	txCount	generatedCoins	fees	activeAddresses	\
count	3838.000000	3838.000000	3838.000000	3.838000e+03	
mean	113123.449713	4641.993498	52.314836	2.790826e+05	
std	115376.139653	2823.200515	105.312467	2.986859e+05	
min	0.000000	200.000000	0.000000	4.000000e+00	
25%	5812.250000	2087.500000	4.690489	1.355775e+04	
50%	65248.500000	3925.000000	19.145662	1.670925e+05	
75%	214696.750000	6650.000000	45.197141	5.393028e+05	
max	490459.000000	34600.000000	1495.946477	1.283929e+06	

	averageDifficulty	paymentCount	blockSize	blockCount	medianFee
count	3.838000e+03	3.838000e+03	3.838000e+03	3838.000000	3584.000000
mean	9.085305e+11	1.830944e+05	5.995056e+07	152.498958	0.000464
std	2.016322e+12	1.860499e+05	6.013257e+07	36.636937	0.006262
min	1.000000e+00	0.000000e+00	8.600000e+02	4.000000	0.000000
25%	1.203462e+06	7.707750e+03	2.619996e+06	138.000000	0.000000
50%	6.119726e+09	1.298930e+05	3.662134e+07	152.000000	0.000100
75%	2.818009e+11	3.421838e+05	1.172087e+08	168.000000	0.000310
max	9.064160e+12	1.883742e+06	2.180726e+08	692.000000	0.220000

[61]:
```
df.fillna({'txCount':df['txCount'].median()},inplace=True)
df.fillna({'generatedCoins':df['generatedCoins'].median()},inplace=True)
df.fillna({'fees':df['fees'].median()},inplace=True)
df.fillna({'activeAddresses':df['activeAddresses'].median()},inplace=True)
df.fillna({'averageDifficulty':df['averageDifficulty'].median()},inplace=True)
df.fillna({'paymentCount':df['paymentCount'].median()},inplace=True)
df.fillna({'blockSize':df['blockSize'].median()},inplace=True)
df.fillna({'blockCount':df['blockCount'].median()},inplace=True)
df.fillna({'medianFee':df['medianFee'].median()},inplace=True)
```

[62]:
```
df.isnull().sum()
```

[62] :
```
date                    0
txVolume(USD)           0
adjustedTxVolume(USD)   0
txCount                 0
marketcap(USD)          0
price(USD)              0
exchangeVolume(USD)     0
realizedCap(USD)        0
generatedCoins          0
fees                    0
activeAddresses         0
averageDifficulty       0
paymentCount            0
medianTxValue(USD)      0
medianFee               0
blockSize               0
blockCount              0
dtype: int64
```

[63]:
```
df[df.duplicated()]
```

[63] :
```
Empty DataFrame
Columns: [date, txVolume(USD), adjustedTxVolume(USD), txCount, marketcap(USD),
price(USD), exchangeVolume(USD), realizedCap(USD), generatedCoins, fees,
activeAddresses, averageDifficulty, paymentCount, medianTxValue(USD), medianFee,
```

blockSize, blockCount]
Index: []

[64]:
```
df.head()
```

[64]:

	date	txVolume(USD)	adjustedTxVolume(USD)	txCount	marketcap(USD)	\
0	2009-01-03	9.655342e+08	3.406209e+08	65248.5	9.632702e+09	
1	2009-01-04	9.655342e+08	3.406209e+08	65248.5	9.632702e+09	
2	2009-01-05	9.655342e+08	3.406209e+08	65248.5	9.632702e+09	
3	2009-01-06	9.655342e+08	3.406209e+08	65248.5	9.632702e+09	
4	2009-01-07	9.655342e+08	3.406209e+08	65248.5	9.632702e+09	

	price(USD)	exchangeVolume(USD)	realizedCap(USD)	generatedCoins	\
0	644.385	72771000.0	0.0	3925.0	
1	644.385	72771000.0	0.0	3925.0	
2	644.385	72771000.0	0.0	3925.0	
3	644.385	72771000.0	0.0	3925.0	
4	644.385	72771000.0	0.0	3925.0	

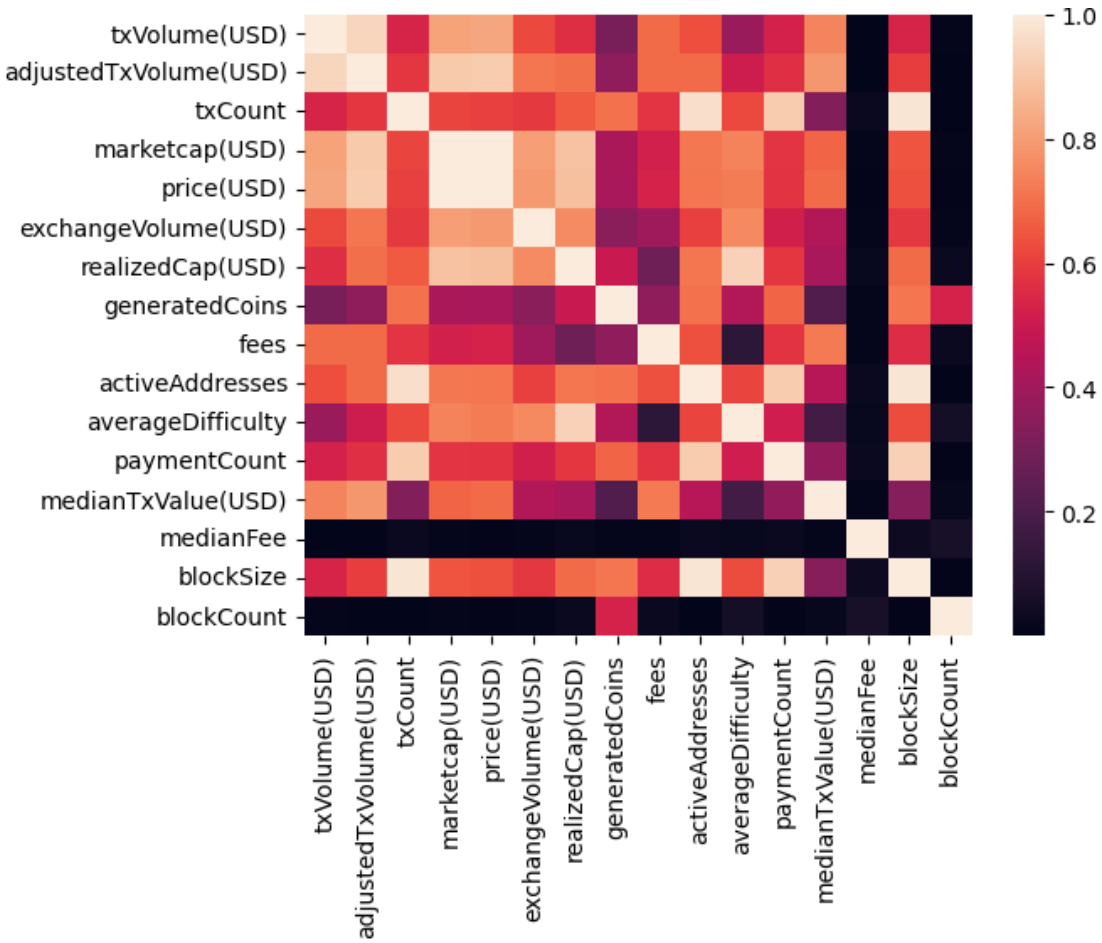
	fees	activeAddresses	averageDifficulty	paymentCount	\
0	19.145662	167092.5	6.119726e+09	129893.0	
1	19.145662	167092.5	6.119726e+09	129893.0	
2	19.145662	167092.5	6.119726e+09	129893.0	
3	19.145662	167092.5	6.119726e+09	129893.0	
4	19.145662	167092.5	6.119726e+09	129893.0	

	medianTxValue(USD)	medianFee	blockSize	blockCount
0	77.244383	0.0001	36621338.5	152.0
1	77.244383	0.0001	36621338.5	152.0
2	77.244383	0.0001	36621338.5	152.0
3	77.244383	0.0001	36621338.5	152.0
4	77.244383	0.0001	36621338.5	152.0

[65]:
```
data = df
data.drop(["date"],axis=1,inplace=True)
```

[66]:
```
sns.heatmap(data.corr().abs())
```

[66]: <Axes: >



[67]:
```
data.corr()
```

[67]:

	txVolume(USD)	adjustedTxVolume(USD)	txCount	\
txVolume(USD)	1.000000	0.941850	0.532546	
adjustedTxVolume(USD)	0.941850	1.000000	0.578802	
txCount	0.532546	0.578802	1.000000	
marketcap(USD)	0.815616	0.911132	0.611544	
price(USD)	0.823033	0.917022	0.603941	
exchangeVolume(USD)	0.618199	0.712651	0.587923	
realizedCap(USD)	0.561160	0.697362	0.655617	
generatedCoins	-0.304866	-0.360624	-0.702100	
fees	0.688654	0.689921	0.577731	
activeAddresses	0.631888	0.691369	0.967694	
averageDifficulty	0.385031	0.506383	0.620025	
paymentCount	0.525403	0.564015	0.916753	
medianTxValue(USD)	0.744569	0.786733	0.331241	
medianFee	0.003870	0.001044	-0.030469	

blockSize	0.531686	0.594699	0.981672
blockCount	0.011682	0.000101	0.001730

	marketcap(USD)	price(USD)	exchangeVolume(USD)	\
txVolume(USD)	0.815616	0.823033	0.618199	
adjustedTxVolume(USD)	0.911132	0.917022	0.712651	
txCount	0.611544	0.603941	0.587923	
marketcap(USD)	1.000000	0.999569	0.803498	
price(USD)	0.999569	1.000000	0.791774	
exchangeVolume(USD)	0.803498	0.791774	1.000000	
realizedCap(USD)	0.892218	0.883963	0.761627	
generatedCoins	-0.419226	-0.415134	-0.347123	
fees	0.522126	0.531273	0.390779	
activeAddresses	0.715823	0.712188	0.601547	
averageDifficulty	0.740684	0.724600	0.754500	
paymentCount	0.575254	0.570573	0.517328	
medianTxValue(USD)	0.673901	0.687735	0.440514	
medianFee	-0.007997	-0.007494	-0.008901	
blockSize	0.643242	0.636415	0.584482	
blockCount	-0.009234	-0.007197	-0.011486	

	realizedCap(USD)	generatedCoins	fees	\
txVolume(USD)	0.561160	-0.304866	0.688654	
adjustedTxVolume(USD)	0.697362	-0.360624	0.689921	
txCount	0.655617	-0.702100	0.577731	
marketcap(USD)	0.892218	-0.419226	0.522126	
price(USD)	0.883963	-0.415134	0.531273	
exchangeVolume(USD)	0.761627	-0.347123	0.390779	
realizedCap(USD)	1.000000	-0.495194	0.277778	
generatedCoins	-0.495194	1.000000	-0.362395	
fees	0.277778	-0.362395	1.000000	
activeAddresses	0.711647	-0.699319	0.636286	
averageDifficulty	0.930530	-0.439743	0.119391	
paymentCount	0.581996	-0.674523	0.570932	
medianTxValue(USD)	0.417979	-0.208726	0.720259	
medianFee	-0.021608	-0.008049	0.008925	
blockSize	0.690035	-0.714249	0.555303	
blockCount	-0.030735	0.528861	0.028565	

	activeAddresses	averageDifficulty	paymentCount	\
txVolume(USD)	0.631888	0.385031	0.525403	
adjustedTxVolume(USD)	0.691369	0.506383	0.564015	
txCount	0.967694	0.620025	0.916753	
marketcap(USD)	0.715823	0.740684	0.575254	
price(USD)	0.712188	0.724600	0.570573	
exchangeVolume(USD)	0.601547	0.754500	0.517328	
realizedCap(USD)	0.711647	0.930530	0.581996	

generatedCoins	-0.699319	-0.439743	-0.674523
fees	0.636286	0.119391	0.570932
activeAddresses	1.000000	0.612464	0.916471
averageDifficulty	0.612464	1.000000	0.508415
paymentCount	0.916471	0.508415	1.000000
medianTxValue(USD)	0.449693	0.178636	0.366733
medianFee	-0.027685	-0.022524	-0.030667
blockSize	0.981958	0.626813	0.927958
blockCount	0.001644	-0.051041	0.005118

	medianTxValue(USD)	medianFee	blockSize	blockCount
txVolume(USD)	0.744569	0.003870	0.531686	0.011682
adjustedTxVolume(USD)	0.786733	0.001044	0.594699	0.000101
txCount	0.331241	-0.030469	0.981672	0.001730
marketcap(USD)	0.673901	-0.007997	0.643242	-0.009234
price(USD)	0.687735	-0.007494	0.636415	-0.007197
exchangeVolume(USD)	0.440514	-0.008901	0.584482	-0.011486
realizedCap(USD)	0.417979	-0.021608	0.690035	-0.030735
generatedCoins	-0.208726	-0.008049	-0.714249	0.528861
fees	0.720259	0.008925	0.555303	0.028565
activeAddresses	0.449693	-0.027685	0.981958	0.001644
averageDifficulty	0.178636	-0.022524	0.626813	-0.051041
paymentCount	0.366733	-0.030667	0.927958	0.005118
medianTxValue(USD)	1.000000	0.009728	0.338314	0.018740
medianFee	0.009728	1.000000	-0.032065	-0.065002
blockSize	0.338314	-0.032065	1.000000	0.007350
blockCount	0.018740	-0.065002	0.007350	1.000000

[68]:
```
#Since marketcap(USD) and price (USD) had a correlation of approximately 1, we drop marketplace(USD)
df.drop(["marketcap(USD)"],axis=1,inplace=True)
```

[69]:
```
print(df["paymentCount"].max())
print(df["paymentCount"].min())
```

```
1883742.0
0.0
```

[70]:
```
print(df["price(USD)"].max())
print(df["price(USD)"].min())
```

```
19475.8
68.5
```

[71]:
```
#paymentCount /price
df["paymentCount Bins"] = pd.cut(x=df["paymentCount"], bins=[0,500000,1000000,1500000,2000000])
```

```
df['price(USD) Bins'] = pd.cut(x=df['price(USD)'],
 ↪bins=[0,5000,10000,15000,20000])
```

[72]:
```
df.head()
```

[72]:

	txVolume(USD)	adjustedTxVolume(USD)	txCount	price(USD)	\
0	9.655342e+08	3.406209e+08	65248.5	644.385	
1	9.655342e+08	3.406209e+08	65248.5	644.385	
2	9.655342e+08	3.406209e+08	65248.5	644.385	
3	9.655342e+08	3.406209e+08	65248.5	644.385	
4	9.655342e+08	3.406209e+08	65248.5	644.385	

	exchangeVolume(USD)	realizedCap(USD)	generatedCoins	fees	\
0	72771000.0	0.0	3925.0	19.145662	
1	72771000.0	0.0	3925.0	19.145662	
2	72771000.0	0.0	3925.0	19.145662	
3	72771000.0	0.0	3925.0	19.145662	
4	72771000.0	0.0	3925.0	19.145662	

	activeAddresses	averageDifficulty	paymentCount	medianTxValue(USD)	\
0	167092.5	6.119726e+09	129893.0	77.244383	
1	167092.5	6.119726e+09	129893.0	77.244383	
2	167092.5	6.119726e+09	129893.0	77.244383	
3	167092.5	6.119726e+09	129893.0	77.244383	
4	167092.5	6.119726e+09	129893.0	77.244383	

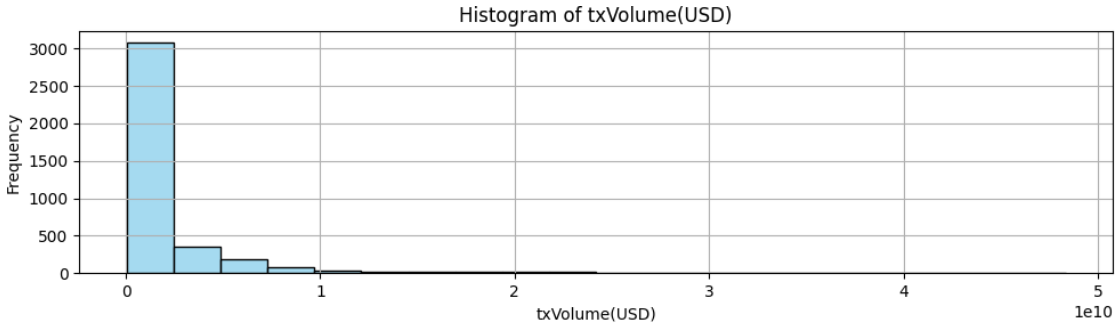
	medianFee	blockSize	blockCount	paymentCount Bins	price(USD) Bins
0	0.0001	36621338.5	152.0	(0, 500000]	(0, 5000]
1	0.0001	36621338.5	152.0	(0, 500000]	(0, 5000]
2	0.0001	36621338.5	152.0	(0, 500000]	(0, 5000]
3	0.0001	36621338.5	152.0	(0, 500000]	(0, 5000]
4	0.0001	36621338.5	152.0	(0, 500000]	(0, 5000]

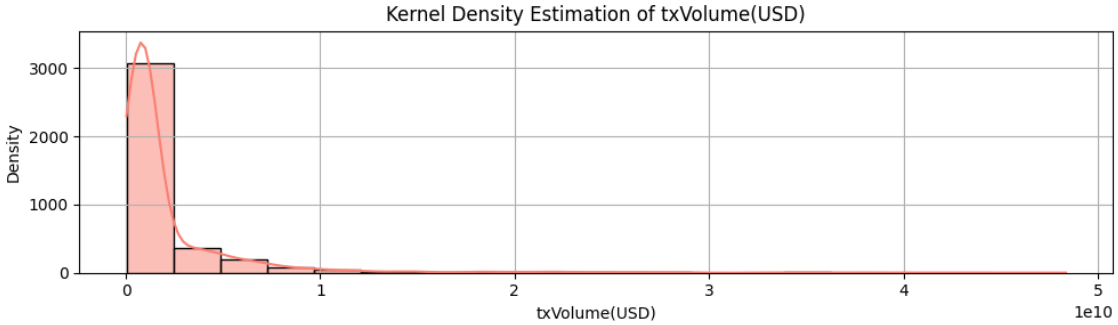
[73]:
```
df.info()
```

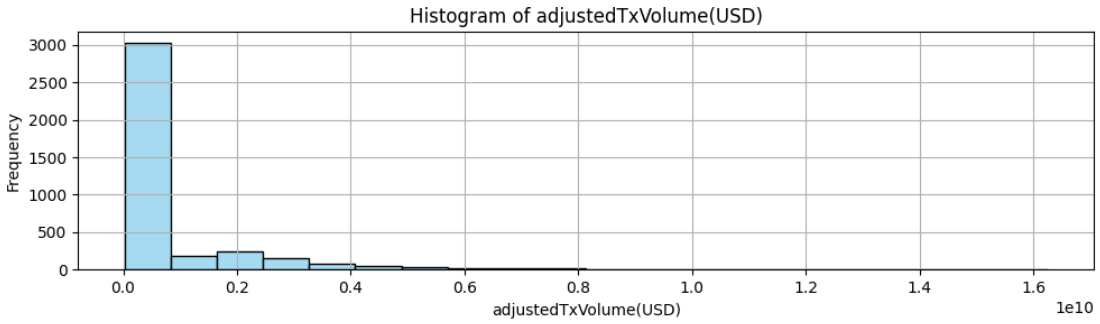
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3844 entries, 0 to 3843
Data columns (total 17 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   txVolume(USD)          3844 non-null   float64
 1   adjustedTxVolume(USD)  3844 non-null   float64
 2   txCount                3844 non-null   float64
 3   price(USD)             3844 non-null   float64
 4   exchangeVolume(USD)    3844 non-null   float64
 5   realizedCap(USD)       3844 non-null   float64
 6   generatedCoins         3844 non-null   float64
```

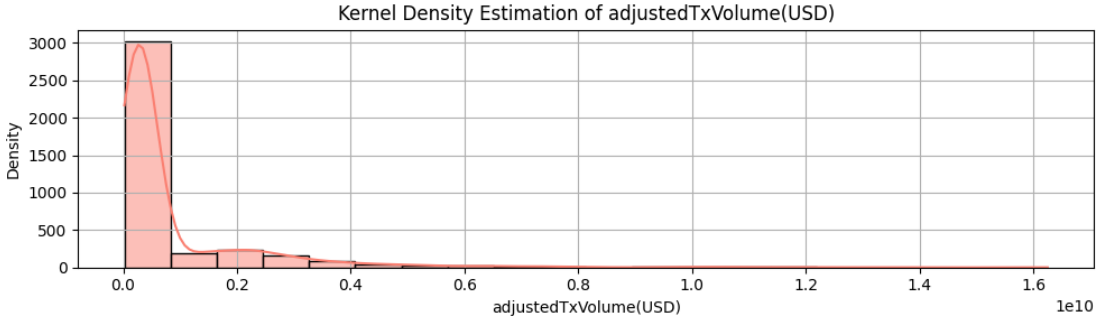
```
 7   fees                    3844 non-null   float64
 8   activeAddresses         3844 non-null   float64
 9   averageDifficulty       3844 non-null   float64
 10  paymentCount            3844 non-null   float64
 11  medianTxValue(USD)      3844 non-null   float64
 12  medianFee               3844 non-null   float64
 13  blockSize               3844 non-null   float64
 14  blockCount              3844 non-null   float64
 15  paymentCount  Bins      3469 non-null   category
 16  price(USD)  Bins        3844 non-null   category
dtypes: category(2), float64(15)
memory usage: 458.6 KB
```

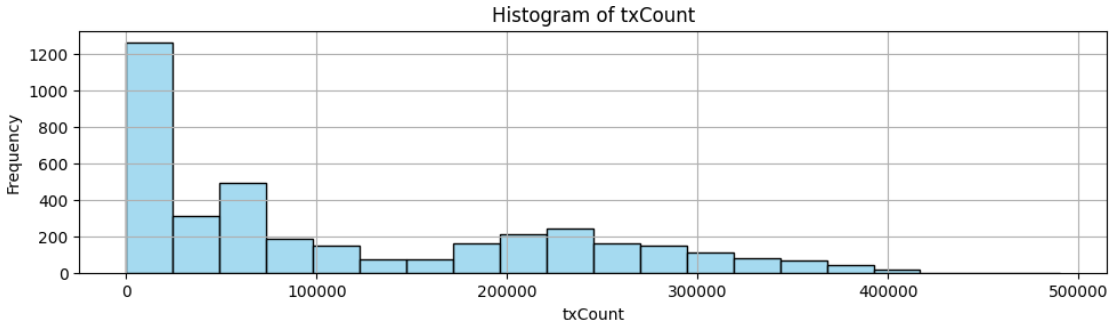
[74]:
```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

numeric_columns = df.select_dtypes(include=['float64']).columns

for column in numeric_columns:
    plt.figure(figsize=(10, 6))

    # Histogram
    plt.subplot(2, 1, 1)
    sns.histplot(df[column], kde=False, color='skyblue', bins=20)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.grid(True)

    # KDE plot
    plt.subplot(2, 1, 2)
    sns.histplot(df[column], kde=True, color='salmon', bins=20)
    plt.title(f'Kernel Density Estimation of {column}')
    plt.xlabel(column)
    plt.ylabel('Density')
    plt.grid(True)

    plt.tight_layout()

    plt.show()
```

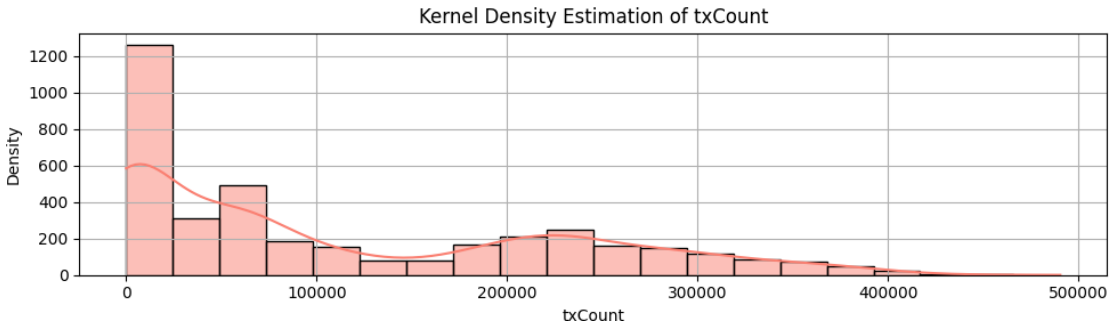

Histogram of txVolume(USD)
Frequency
txVolume(USD)
1e10

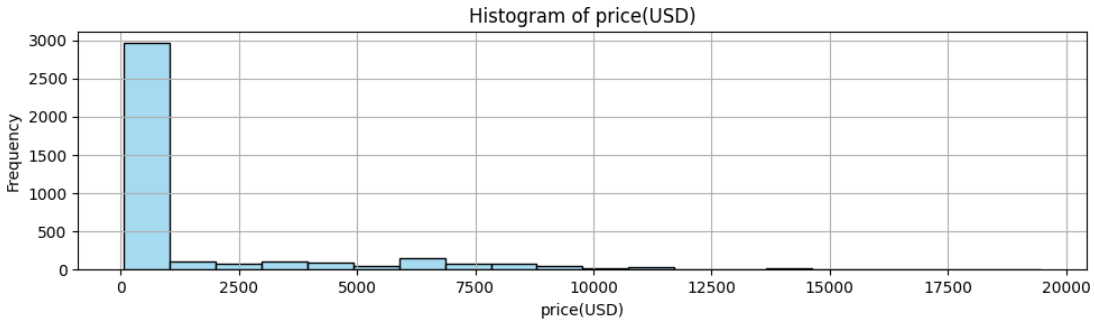

Kernel Density Estimation of txVolume(USD)
Density
txVolume(USD)
1e10

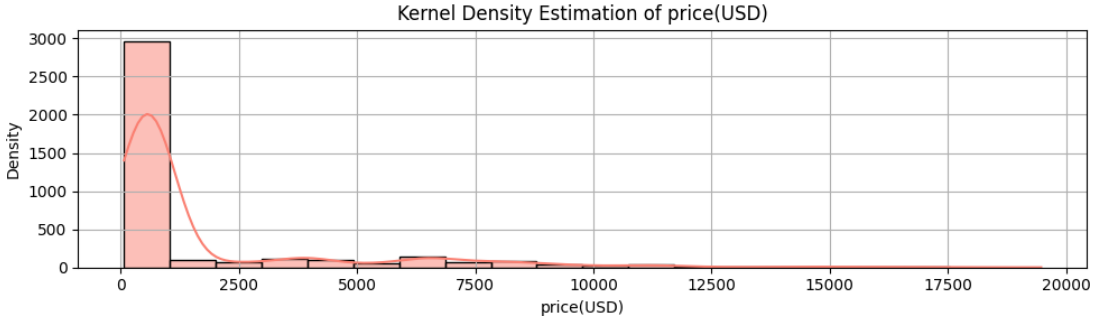

Histogram of adjustedTxVolume(USD)
Frequency
adjustedTxVolume(USD)
1e10

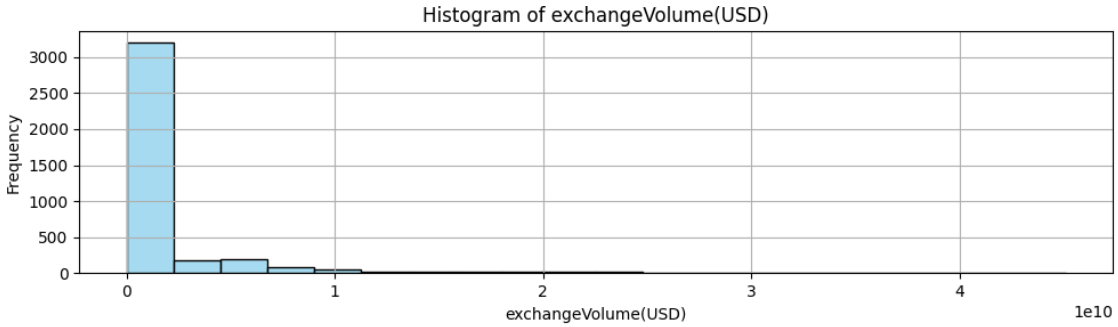

Kernel Density Estimation of adjustedTxVolume(USD)
Density
adjustedTxVolume(USD)
1e10

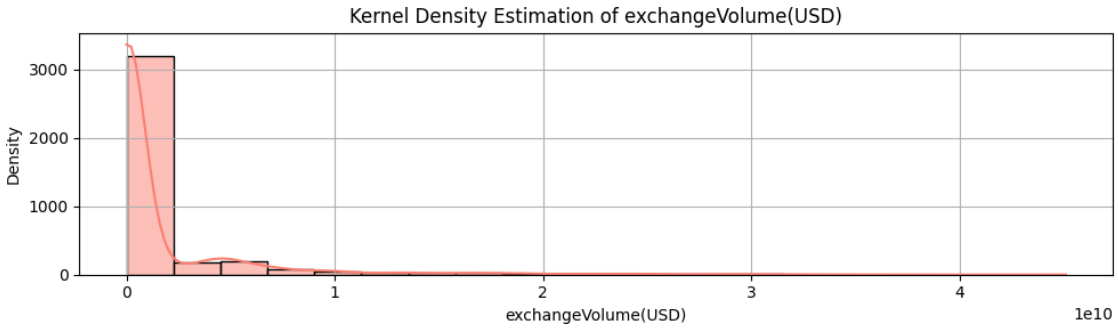

Histogram of txCount
Frequency
txCount
0
100000
200000
300000
400000
500000

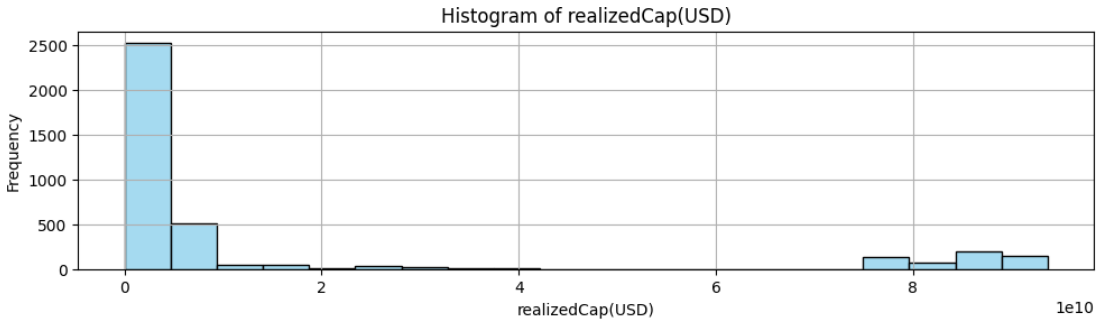

Kernel Density Estimation of txCount
Density
txCount
0
100000
200000
300000
400000
500000

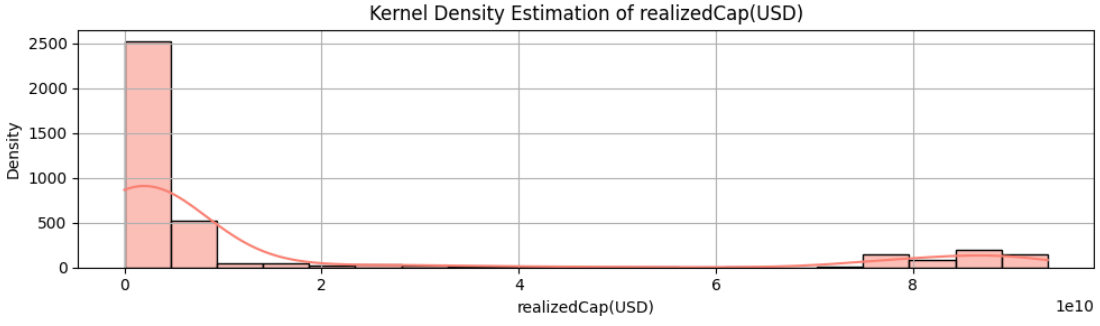

Histogram of price(USD)
Frequency
price(USD)
0
2500
5000
7500
10000
12500
15000
17500
20000

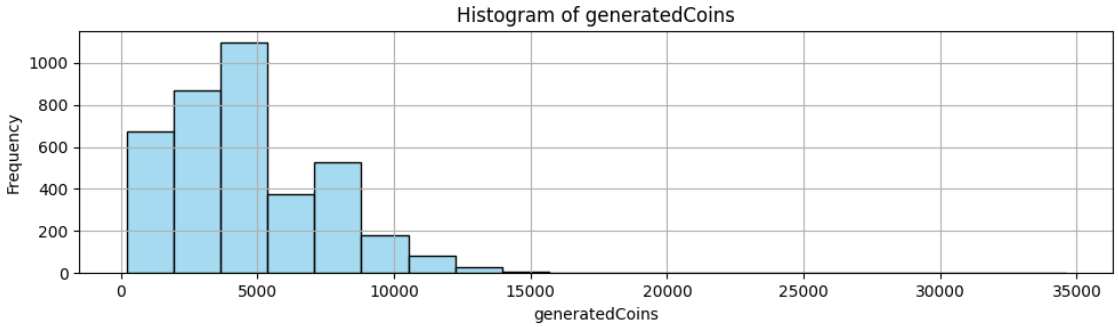

Kernel Density Estimation of price(USD)
Density
price(USD)
0
2500
5000
7500
10000
12500
15000
17500
20000

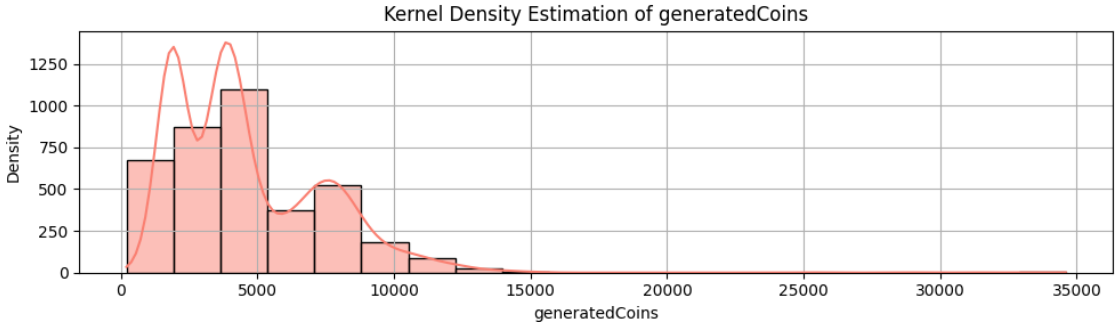

Histogram of exchangeVolume(USD)
Frequency
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
exchangeVolume(USD)
1e10

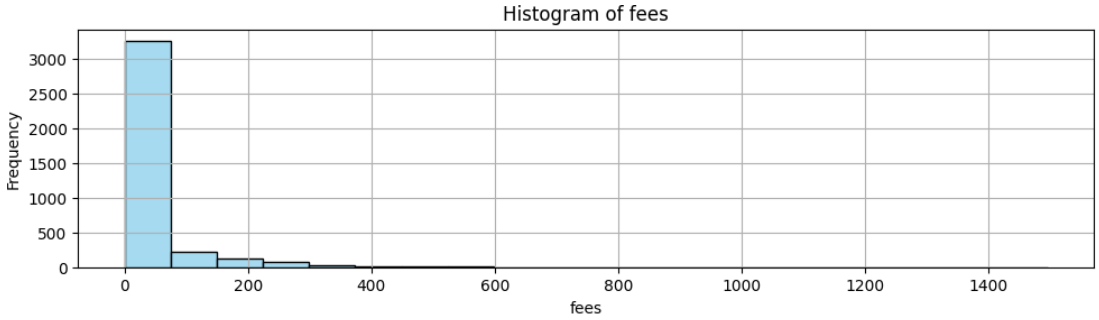

Kernel Density Estimation of exchangeVolume(USD)
Density
3000
2000
1000
0
0
1
2
3
4
exchangeVolume(USD)
1e10

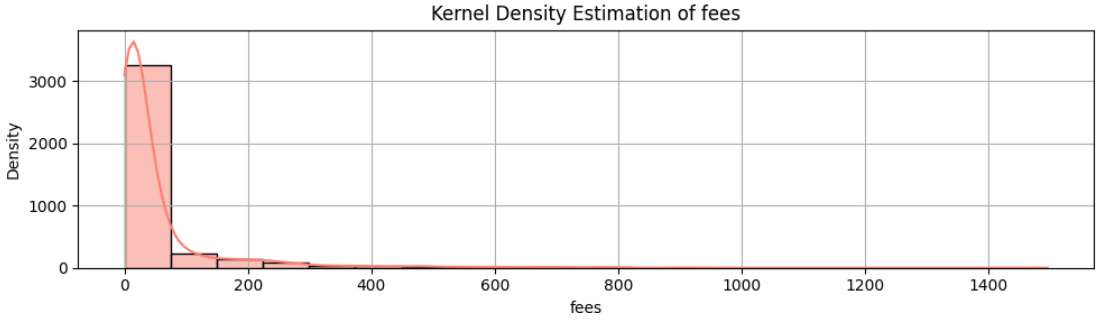

Histogram of realizedCap(USD)
Frequency
2500
2000
1500
1000
500
0
0
2
4
6
8
realizedCap(USD)
1e10

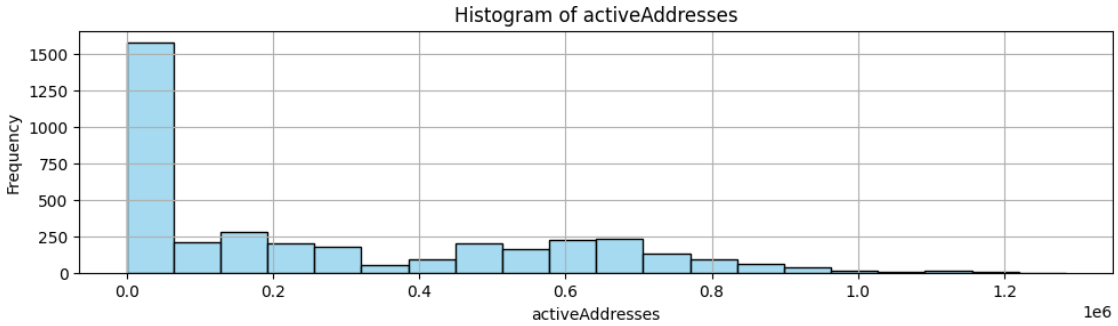

Kernel Density Estimation of realizedCap(USD)
Density
2500
2000
1500
1000
500
0
0
2
4
6
8
realizedCap(USD)
1e10

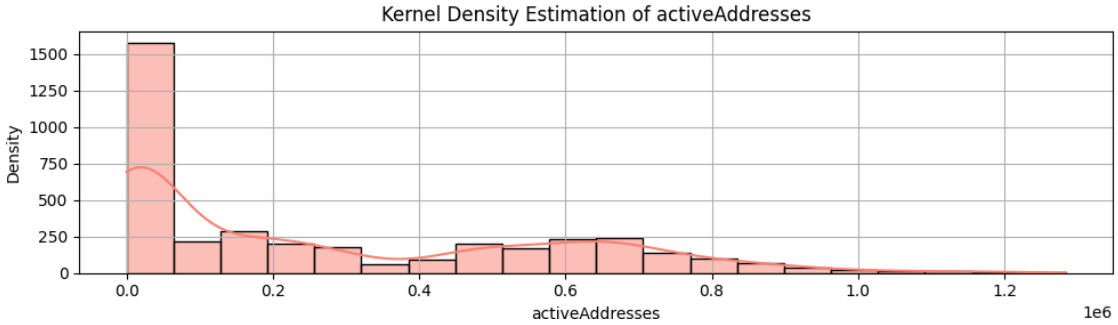

Histogram of generatedCoins
Frequency
generatedCoins
0
200
400
600
800
1000
0
5000
10000
15000
20000
25000
30000
35000

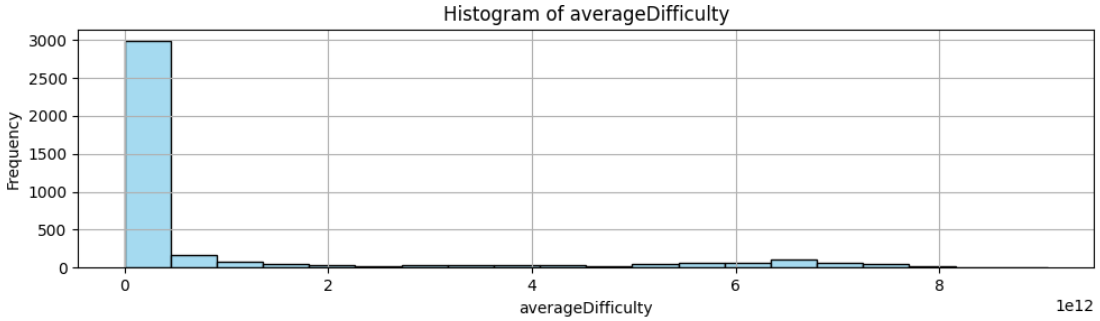

Kernel Density Estimation of generatedCoins
Density
generatedCoins
0
250
500
750
1000
1250
0
5000
10000
15000
20000
25000
30000
35000

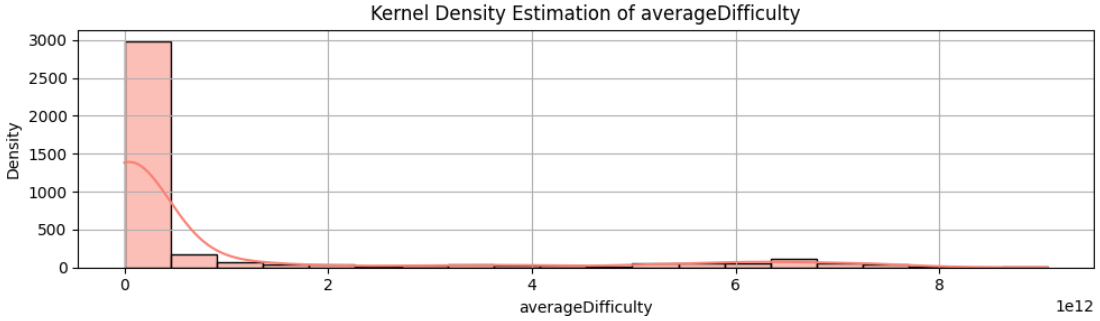

Histogram of fees
Frequency
fees
0
500
1000
1500
2000
2500
3000
0
200
400
600
800
1000
1200
1400

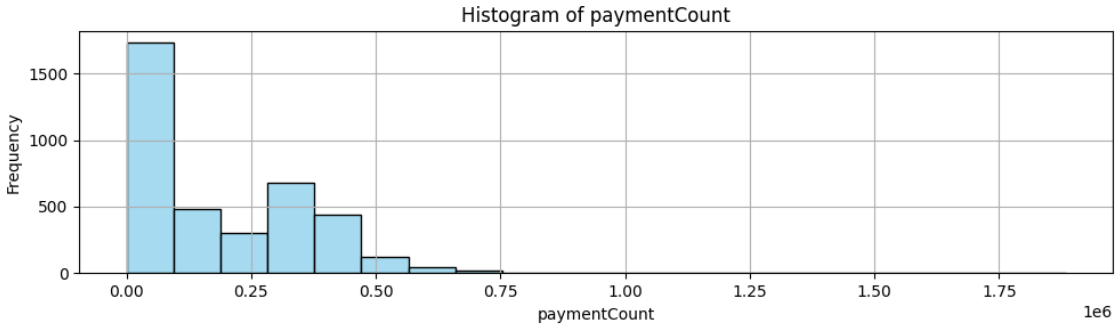

Kernel Density Estimation of fees
Density
fees
0
1000
2000
3000
0
200
400
600
800
1000
1200
1400

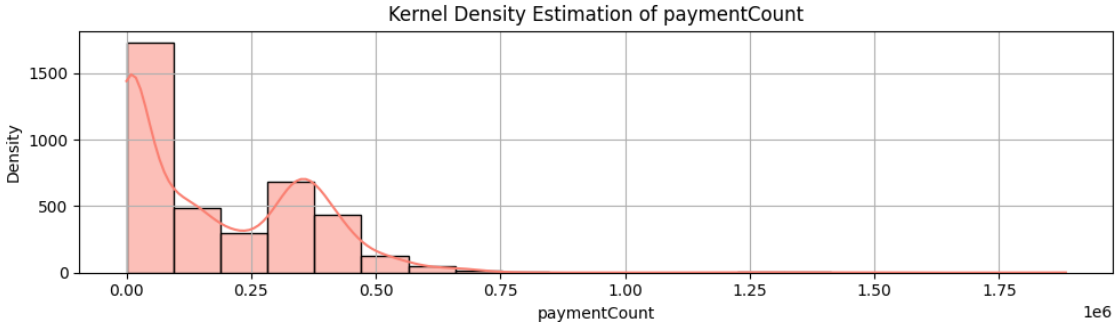

Histogram of activeAddresses
Frequency
activeAddresses
1e6

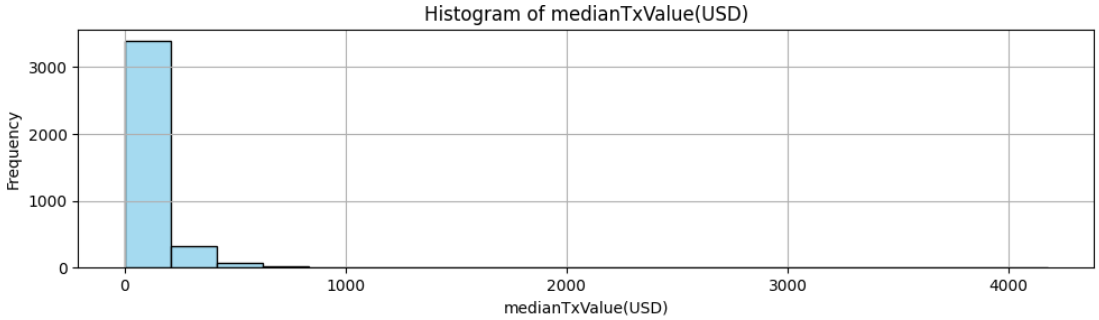

Kernel Density Estimation of activeAddresses
Density
activeAddresses
1e6

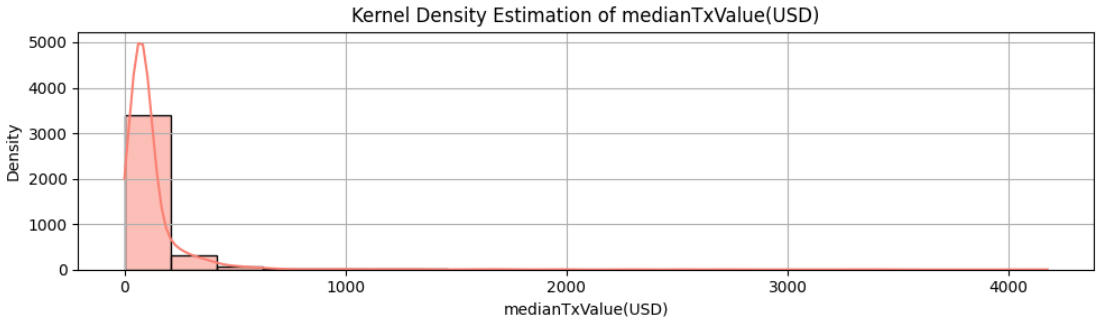

Histogram of averageDifficulty
Frequency
averageDifficulty
1e12

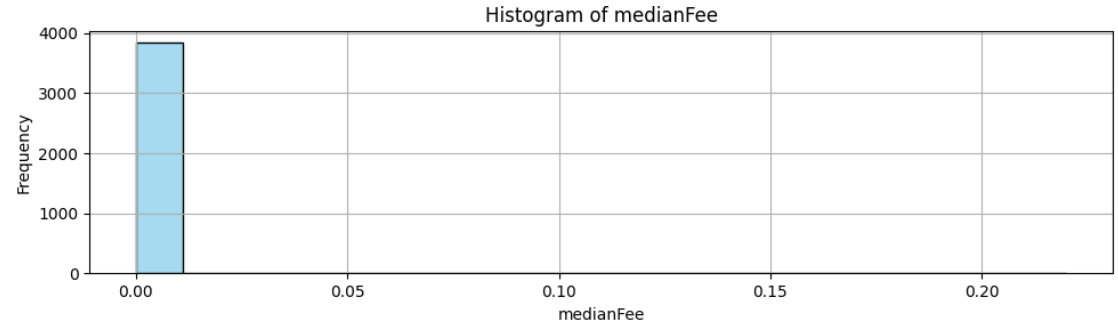

Kernel Density Estimation of averageDifficulty
Density
averageDifficulty
1e12

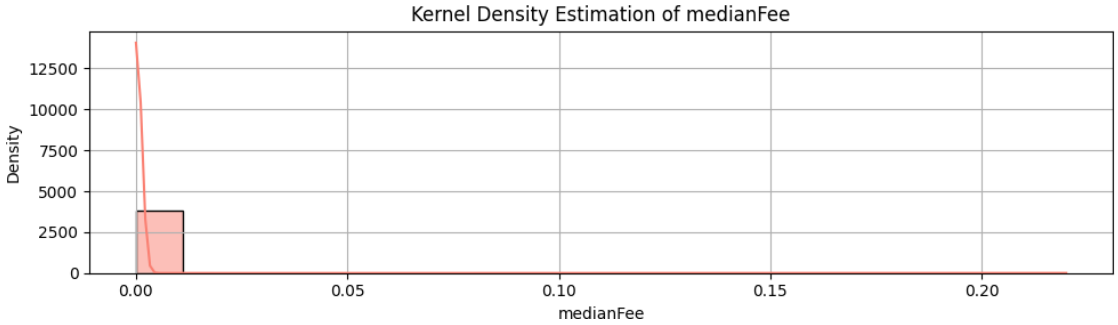

Histogram of paymentCount
Frequency
paymentCount
1e6

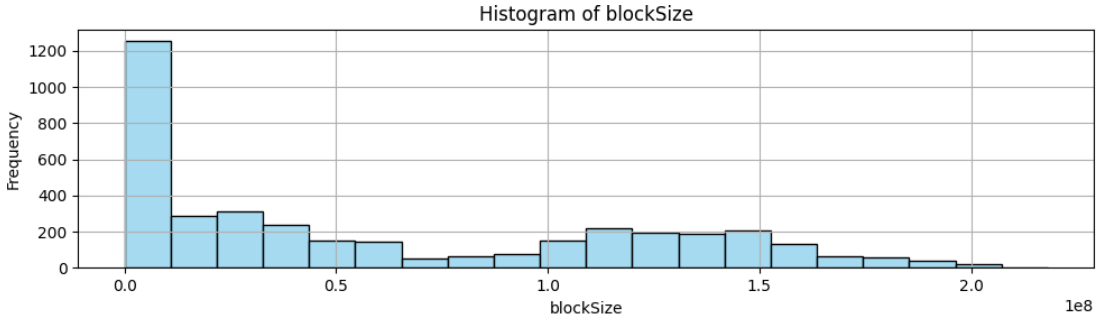

Kernel Density Estimation of paymentCount
Density
paymentCount
1e6

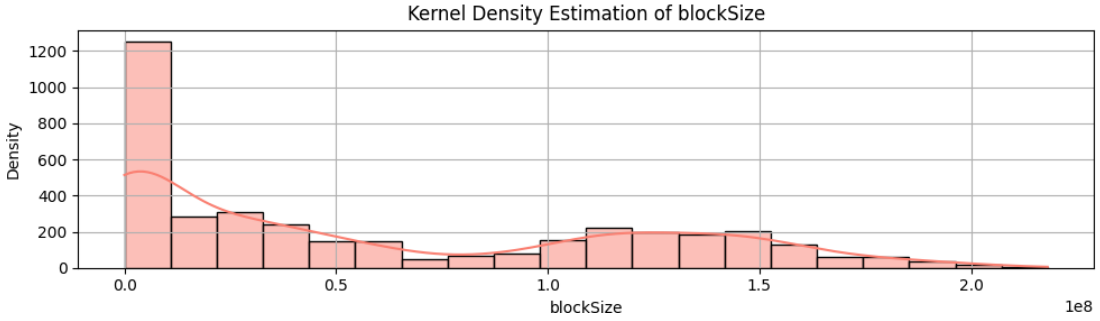

Histogram of medianTxValue(USD)
Frequency
medianTxValue(USD)

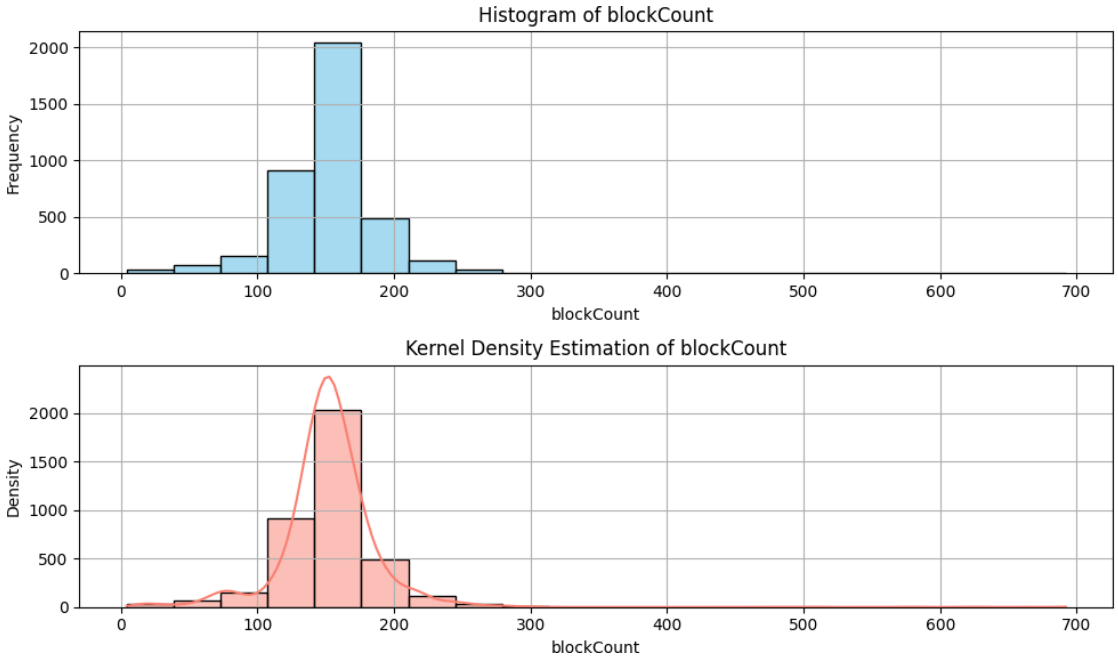

Kernel Density Estimation of medianTxValue(USD)
Density
medianTxValue(USD)


Histogram of medianFee
Frequency
4000
3000
2000
1000
0
0.00
0.05
0.10
0.15
0.20
medianFee


Kernel Density Estimation of medianFee
Density
12500
10000
7500
5000
2500
0
0.00
0.05
0.10
0.15
0.20
medianFee


Histogram of blockSize
Frequency
1200
1000
800
600
400
200
0
0.0
0.5
1.0
1.5
2.0
blockSize
1e8


Kernel Density Estimation of blockSize
Density
1200
1000
800
600
400
200
0
0.0
0.5
1.0
1.5
2.0
blockSize
1e8

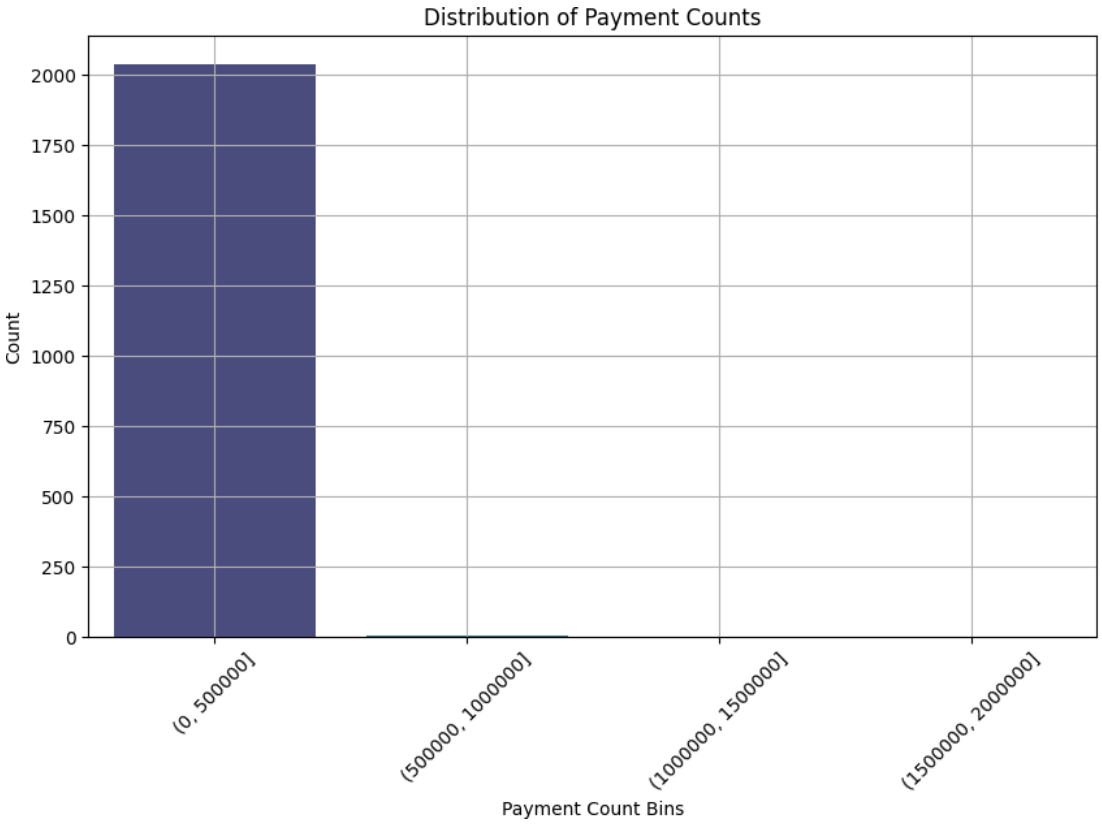


[89]:
```
import matplotlib.pyplot as plt
import seaborn as sns

payment_count_bins_counts = df['paymentCount Bins'].value_counts()

plt.figure(figsize=(10, 6))
sns.barplot(x=payment_count_bins_counts.index, y=payment_count_bins_counts.
↪values, palette='viridis')
plt.title('Distribution of Payment Counts')
plt.xlabel('Payment Count Bins')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

<ipython-input-89-989cc329841c>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

sns.barplot(x=payment_count_bins_counts.index, y=payment_count_bins_counts.values, palette='viridis')



[90]:
```
df=pd.read_csv("/content/doge.csv")
df.head()
```

[90]:

	date	txVolume(USD)	adjustedTxVolume(USD)	txCount	marketcap(USD)	price(USD)	exchangeVolume(USD)	realizedCap(USD)	generatedCoins	fees	activeAddresses	averageDifficulty	paymentCount
0	2013-12-08	NaN	NaN	4665	NaN	NaN	NaN	0.0	1.240744e+08	4245.695900	6990	0.494418	4655
1	2013-12-09	NaN	NaN	9339	NaN	NaN	NaN	0.0	7.619429e+08	7641.853601	10863	2.376904	9367
2	2013-12-10	NaN	NaN	9450	NaN	NaN	NaN	0.0	7.109302e+08	5478.650505	12132	3.361733	9561
3	2013-12-11	NaN	NaN	28951	NaN	NaN	NaN	0.0	7.709661e+08				
4	2013-12-12	NaN	NaN	44036	NaN	NaN	NaN	0.0	6.988183e+08				

```
3  2863.190170          43044          5.092981         28074
4   954.049113          61432          7.277436         40245

   medianTxValue(USD)  medianFee  blockSize  blockCount
0                 NaN     1.0000    1707027        2486
1                 NaN     1.0000    2759110        1526
2                 NaN     1.0000    2971715        1417
3                 NaN     0.0001    8431778        1575
4                 NaN     0.0001   13549784        1460
```

[91]:
```
df.isnull().sum()/len(df)
```

[91]:
```
date                    0.000000
txVolume(USD)           0.003425
adjustedTxVolume(USD)   0.003425
txCount                 0.000000
marketcap(USD)          0.003425
price(USD)              0.003425
exchangeVolume(USD)     0.003425
realizedCap(USD)        0.000000
generatedCoins          0.000000
fees                    0.000000
activeAddresses         0.000000
averageDifficulty       0.000000
paymentCount            0.000000
medianTxValue(USD)      0.003425
medianFee               0.000000
blockSize               0.000000
blockCount              0.000000
dtype: float64
```

[92]:
```
df[['txVolume(USD)','adjustedTxVolume(USD)','marketcap(USD)','price(USD)','exchangeVolume(USD)
  ↪describe(include='all')
```

[92]:
```
        txVolume(USD)  adjustedTxVolume(USD)  marketcap(USD)    price(USD)    \
count    2.037000e+03           2.037000e+03    2.037000e+03  2037.000000
mean     8.478678e+07           1.115406e+07    1.628500e+08     0.001465
std      2.292751e+08           2.217908e+07    2.255299e+08     0.001954
min      2.725822e+05           1.260354e+05    1.669611e+06     0.000087
25%      2.539105e+06           5.729270e+05    2.076736e+07     0.000210
50%      1.117022e+07           1.703473e+06    3.007730e+07     0.000344
75%      6.071726e+07           1.159882e+07    2.756158e+08     0.002370
max      3.506450e+09           2.249827e+08    1.928700e+09     0.017115

        exchangeVolume(USD)  medianTxValue(USD)
count          2.037000e+03         2037.000000
mean           9.704686e+06           11.264403
```

std	2.197459e+07	40.912553
min	0.000000e+00	0.001225
25%	1.563060e+05	0.240000
50%	7.082490e+05	1.994694
75%	1.070230e+07	5.493398
max	2.881840e+08	430.917201

[93]:
```
df.fillna({"txVolume(USD)": df["txVolume(USD)"].median()}, inplace=True)
df.fillna({"adjustedTxVolume(USD)":df["adjustedTxVolume(USD)"].
 ↪median()},inplace=True)
df.fillna({"marketcap(USD)": df["marketcap(USD)"].median()}, inplace=True)
df.fillna({"price(USD)": df["price(USD)"].median()}, inplace=True)
df.fillna({"exchangeVolume(USD)": df["exchangeVolume(USD)"].median()},␣
 ↪inplace=True)
df.fillna({"medianTxValue(USD)": df["medianTxValue(USD)"].median()},␣
 ↪inplace=True)
```

[94]:
```
df.isnull().sum()
```

[94] :
date	0
txVolume(USD)	0
adjustedTxVolume(USD)	0
txCount	0
marketcap(USD)	0
price(USD)	0
exchangeVolume(USD)	0
realizedCap(USD)	0
generatedCoins	0
fees	0
activeAddresses	0
averageDifficulty	0
paymentCount	0
medianTxValue(USD)	0
medianFee	0
blockSize	0
blockCount	0

dtype: int64

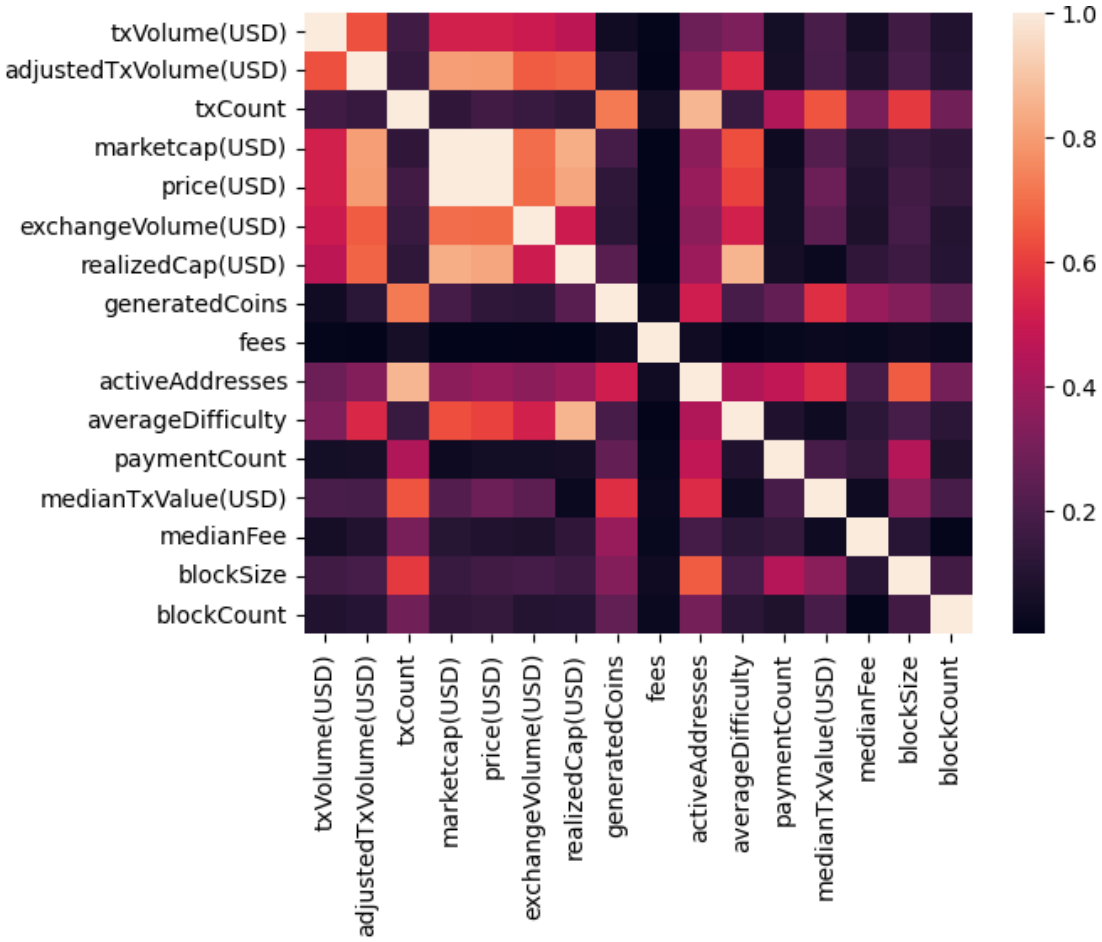
[95]:
```
df[df.duplicated()]
```

[95] : Empty DataFrame
Columns: [date, txVolume(USD), adjustedTxVolume(USD), txCount, marketcap(USD), price(USD), exchangeVolume(USD), realizedCap(USD), generatedCoins, fees, activeAddresses, averageDifficulty, paymentCount, medianTxValue(USD), medianFee, blockSize, blockCount]
Index: []

[96]:
```
data = df
data.drop(["date"],axis=1,inplace=True)
```

[97]:
```
sns.heatmap(data.corr().abs())
```

[97] : <Axes: >



[98]:
```
data.corr()
```

[98] :

	txVolume(USD)	adjustedTxVolume(USD)	txCount	\
txVolume(USD)	1.000000	0.634288	0.171616	
adjustedTxVolume(USD)	0.634288	1.000000	0.148330	
txCount	0.171616	0.148330	1.000000	
marketcap(USD)	0.523121	0.802099	0.129250	
price(USD)	0.523375	0.799480	0.173619	
exchangeVolume(USD)	0.498121	0.658956	0.150908	
realizedCap(USD)	0.465127	0.673976	0.123639	

generatedCoins	-0.046381	-0.110736	0.719195
fees	0.009659	0.004933	0.059436
activeAddresses	0.274730	0.333463	0.861133
averageDifficulty	0.318602	0.543317	0.151865
paymentCount	0.052895	0.059566	0.433595
medianTxValue(USD)	0.191895	0.187732	0.643583
medianFee	0.056255	0.087101	-0.305269
blockSize	0.170311	0.185285	0.589810
blockCount	0.089448	0.101293	0.287335

	marketcap(USD)	price(USD)	exchangeVolume(USD)	\
txVolume(USD)	0.523121	0.523375	0.498121	
adjustedTxVolume(USD)	0.802099	0.799480	0.658956	
txCount	0.129250	0.173619	0.150908	
marketcap(USD)	1.000000	0.997181	0.694360	
price(USD)	0.997181	1.000000	0.689602	
exchangeVolume(USD)	0.694360	0.689602	1.000000	
realizedCap(USD)	0.841361	0.824289	0.500588	
generatedCoins	-0.181327	-0.125181	-0.117236	
fees	0.000518	0.002322	0.007699	
activeAddresses	0.351510	0.382904	0.352746	
averageDifficulty	0.632415	0.606783	0.521069	
paymentCount	0.033859	0.049145	0.050842	
medianTxValue(USD)	0.216886	0.275112	0.241510	
medianFee	0.103833	0.089961	0.076101	
blockSize	0.149816	0.172738	0.182082	
blockCount	0.129196	0.140282	0.094268	

	realizedCap(USD)	generatedCoins	fees	\
txVolume(USD)	0.465127	-0.046381	0.009659	
adjustedTxVolume(USD)	0.673976	-0.110736	0.004933	
txCount	0.123639	0.719195	0.059436	
marketcap(USD)	0.841361	-0.181327	0.000518	
price(USD)	0.824289	-0.125181	0.002322	
exchangeVolume(USD)	0.500588	-0.117236	0.007699	
realizedCap(USD)	1.000000	-0.230168	0.000786	
generatedCoins	-0.230168	1.000000	0.039521	
fees	0.000786	0.039521	1.000000	
activeAddresses	0.389283	0.508994	0.045745	
averageDifficulty	0.856373	-0.190410	0.005369	
paymentCount	0.057000	0.257803	0.019792	
medianTxValue(USD)	0.028228	0.558876	0.025167	
medianFee	0.127860	-0.381350	-0.022982	
blockSize	0.159047	0.333334	0.041737	
blockCount	0.099054	0.251775	0.030186	

	activeAddresses	averageDifficulty	paymentCount	\

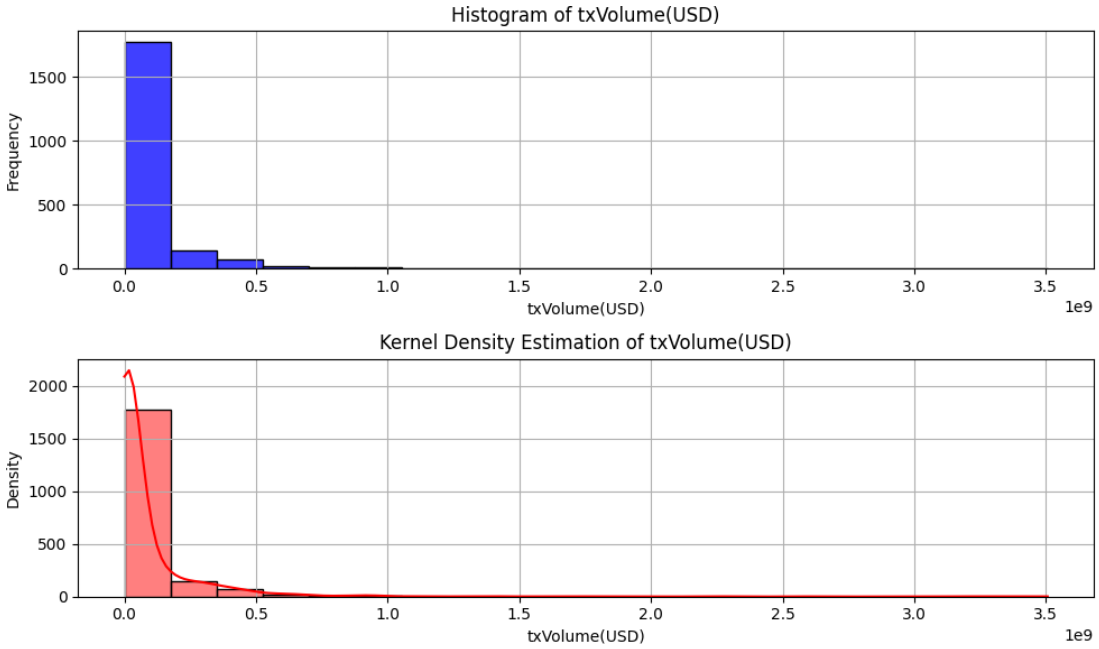
txVolume(USD)	0.274730	0.318602	0.052895
adjustedTxVolume(USD)	0.333463	0.543317	0.059566
txCount	0.861133	0.151865	0.433595
marketcap(USD)	0.351510	0.632415	0.033859
price(USD)	0.382904	0.606783	0.049145
exchangeVolume(USD)	0.352746	0.521069	0.050842
realizedCap(USD)	0.389283	0.856373	0.057000
generatedCoins	0.508994	-0.190410	0.257803
fees	0.045745	0.005369	0.019792
activeAddresses	1.000000	0.434147	0.475946
averageDifficulty	0.434147	1.000000	0.085170
paymentCount	0.475946	0.085170	1.000000
medianTxValue(USD)	0.552771	-0.036752	0.190353
medianFee	-0.180213	0.118737	-0.139998
blockSize	0.660071	0.187336	0.447274
blockCount	0.300292	0.115868	0.081228

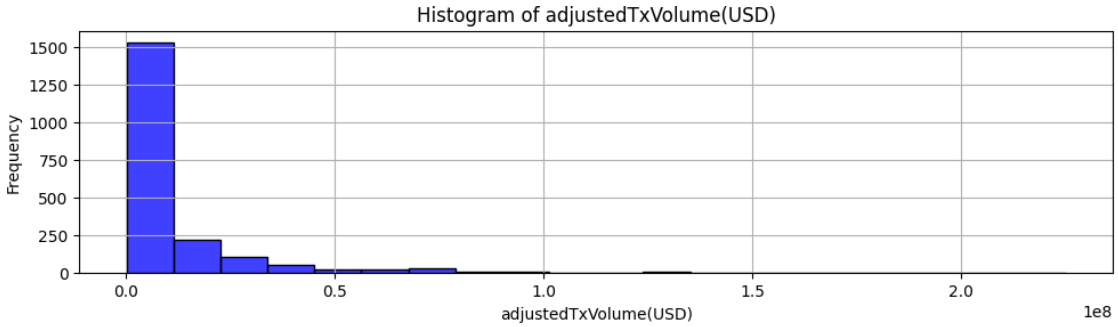
	medianTxValue(USD)	medianFee	blockSize	blockCount
txVolume(USD)	0.191895	0.056255	0.170311	0.089448
adjustedTxVolume(USD)	0.187732	0.087101	0.185285	0.101293
txCount	0.643583	-0.305269	0.589810	0.287335
marketcap(USD)	0.216886	0.103833	0.149816	0.129196
price(USD)	0.275112	0.089961	0.172738	0.140282
exchangeVolume(USD)	0.241510	0.076101	0.182082	0.094268
realizedCap(USD)	0.028228	0.127860	0.159047	0.099054
generatedCoins	0.558876	-0.381350	0.333334	0.251775
fees	0.025167	-0.022982	0.041737	0.030186
activeAddresses	0.552771	-0.180213	0.660071	0.300292
averageDifficulty	-0.036752	0.118737	0.187336	0.115868
paymentCount	0.190353	-0.139998	0.447274	0.081228
medianTxValue(USD)	1.000000	-0.037343	0.345331	0.188358
medianFee	-0.037343	1.000000	-0.106344	-0.011698
blockSize	0.345331	-0.106344	1.000000	0.172780
blockCount	0.188358	-0.011698	0.172780	1.000000

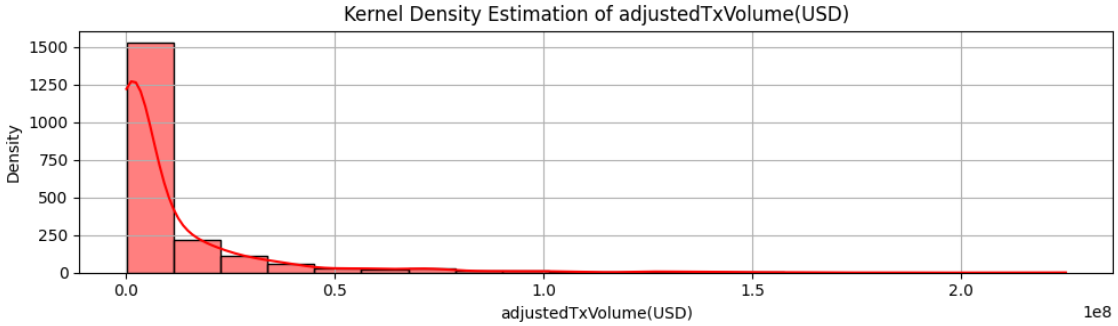
[99]:
```
df.drop(['marketcap(USD)'],axis=1,inplace=True)
```

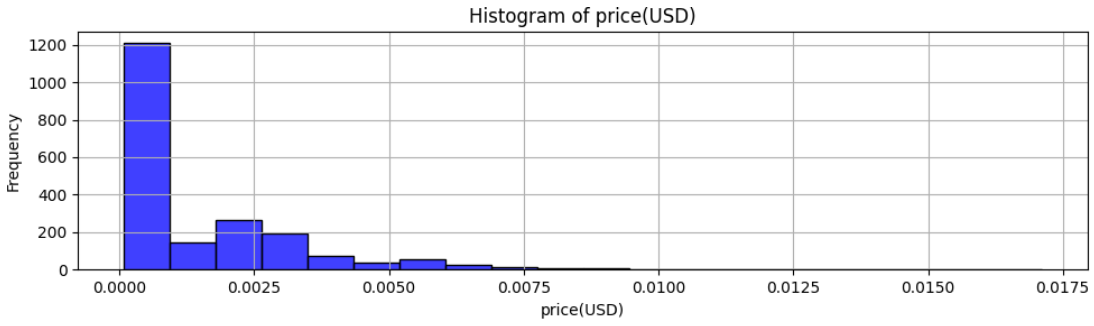
[100]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

numeric_columns = df.select_dtypes(include=['float64']).columns

for column in numeric_columns:
    plt.figure(figsize=(10, 6))

    # Histogram
```

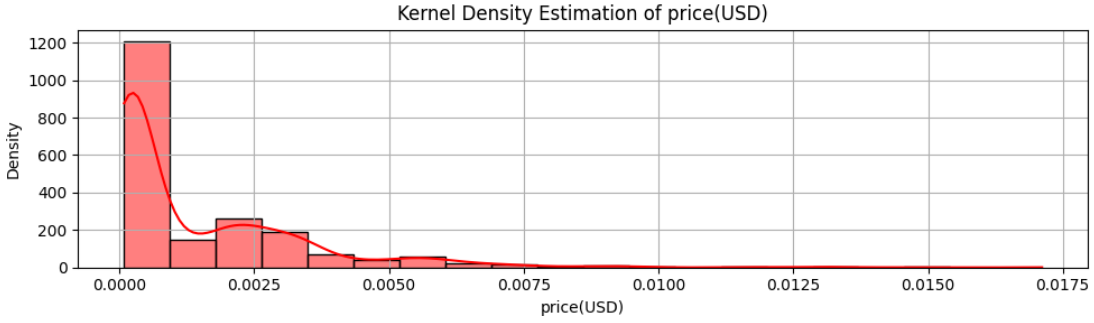
```
    plt.subplot(2, 1, 1)
    sns.histplot(df[column], kde=False, color='blue', bins=20)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.grid(True)

    # KDE plot
    plt.subplot(2, 1, 2)
    sns.histplot(df[column], kde=True, color='red', bins=20)
    plt.title(f'Kernel Density Estimation of {column}')
    plt.xlabel(column)
    plt.ylabel('Density')
    plt.grid(True)

    plt.tight_layout()

    plt.show()
```

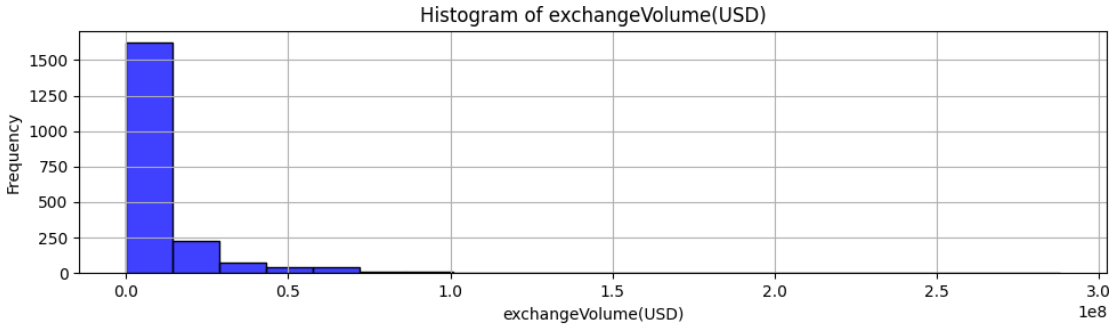


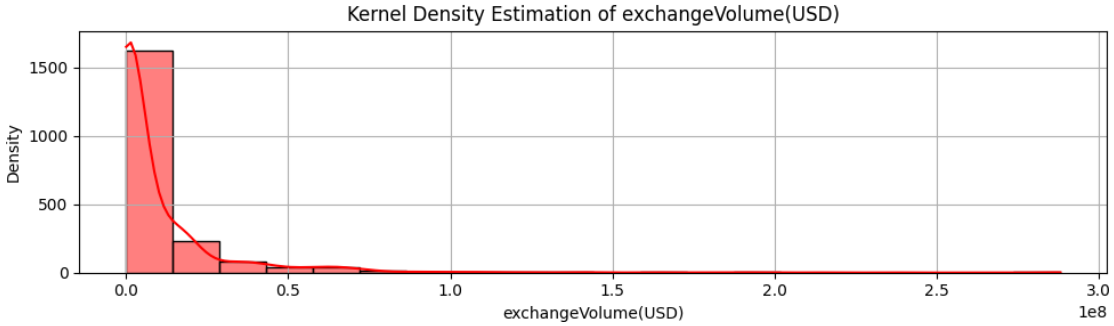

Histogram of adjustedTxVolume(USD)
Frequency
1500
1250
1000
750
500
250
0
0.0
0.5
1.0
1.5
2.0
adjustedTxVolume(USD)
1e8

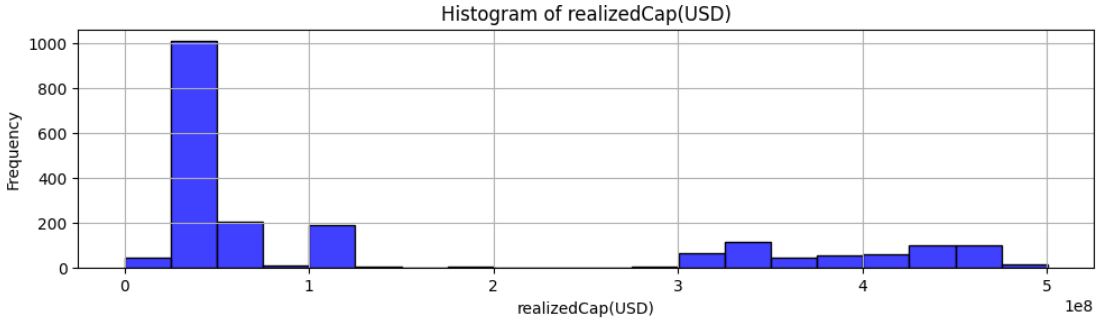

Kernel Density Estimation of adjustedTxVolume(USD)
Density
1500
1250
1000
750
500
250
0
0.0
0.5
1.0
1.5
2.0
adjustedTxVolume(USD)
1e8

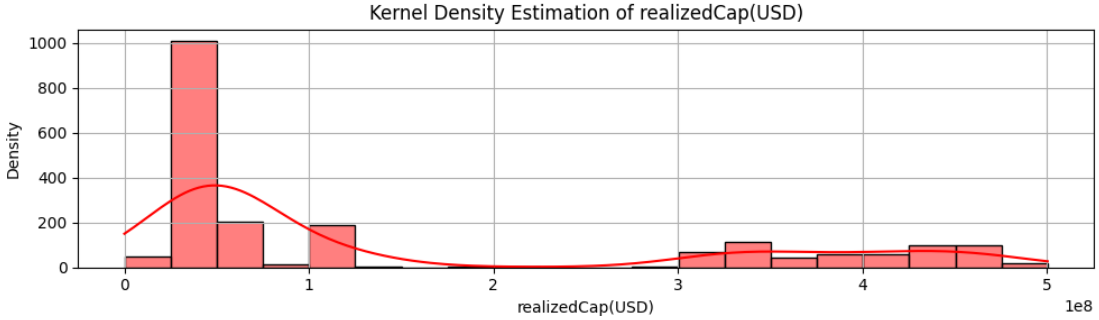

Histogram of price(USD)
Frequency
1200
1000
800
600
400
200
0
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
price(USD)

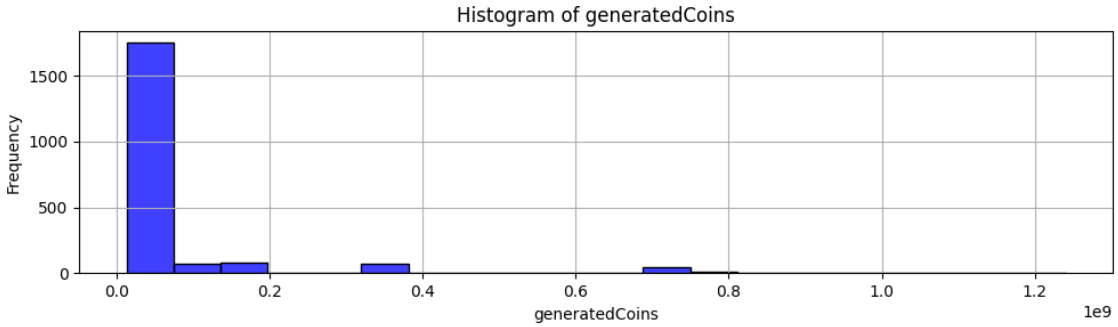

Kernel Density Estimation of price(USD)
Density
1200
1000
800
600
400
200
0
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
price(USD)

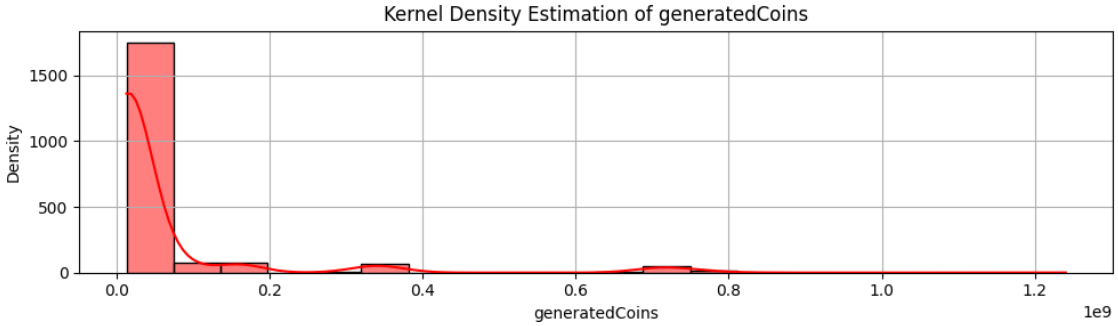

Histogram of exchangeVolume(USD)
Frequency
1500
1250
1000
750
500
250
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
exchangeVolume(USD)
1e8

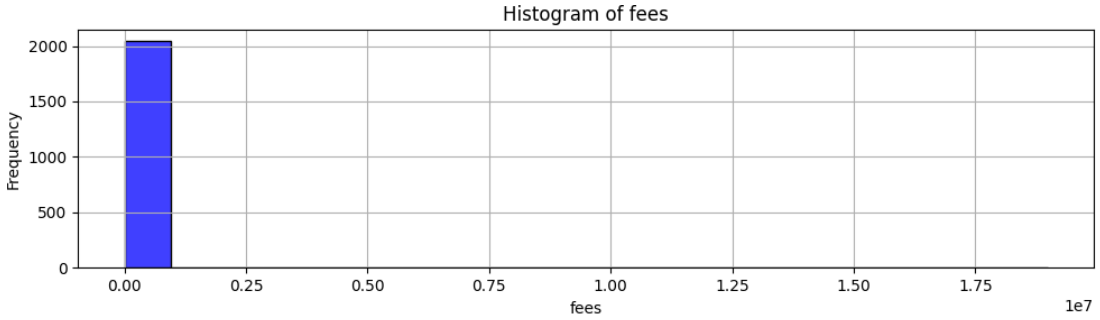

Kernel Density Estimation of exchangeVolume(USD)
Density
1500
1000
500
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
exchangeVolume(USD)
1e8

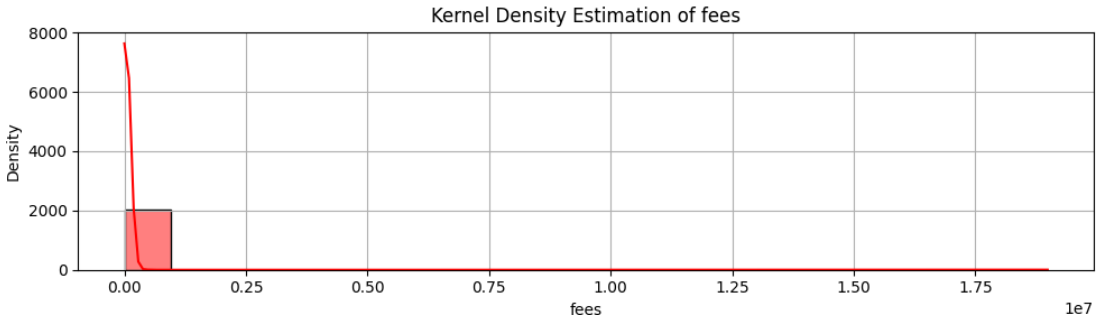

Histogram of realizedCap(USD)
Frequency
1000
800
600
400
200
0
0
1
2
3
4
5
realizedCap(USD)
1e8

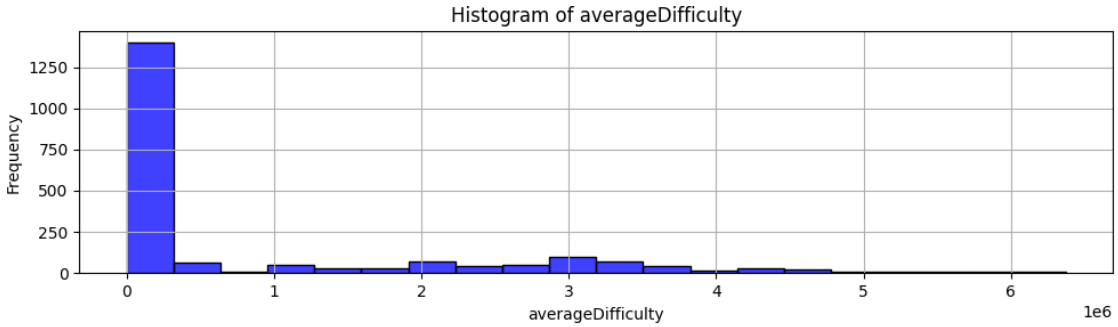

Kernel Density Estimation of realizedCap(USD)
Density
1000
800
600
400
200
0
0
1
2
3
4
5
realizedCap(USD)
1e8

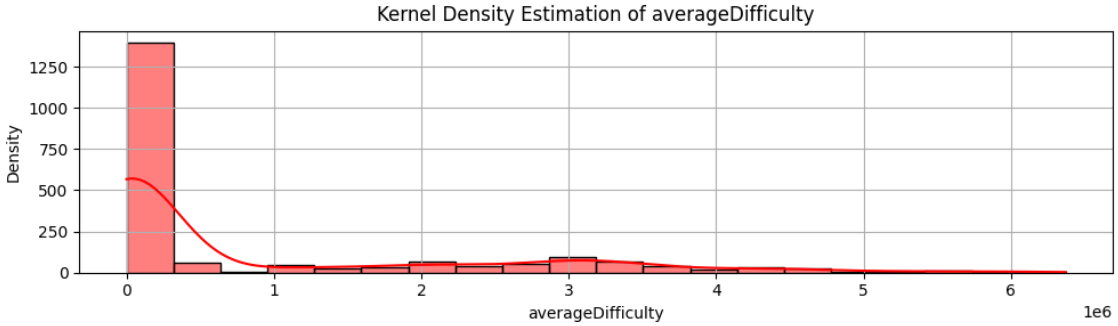

Histogram of generatedCoins
Frequency
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
generatedCoins
1e9

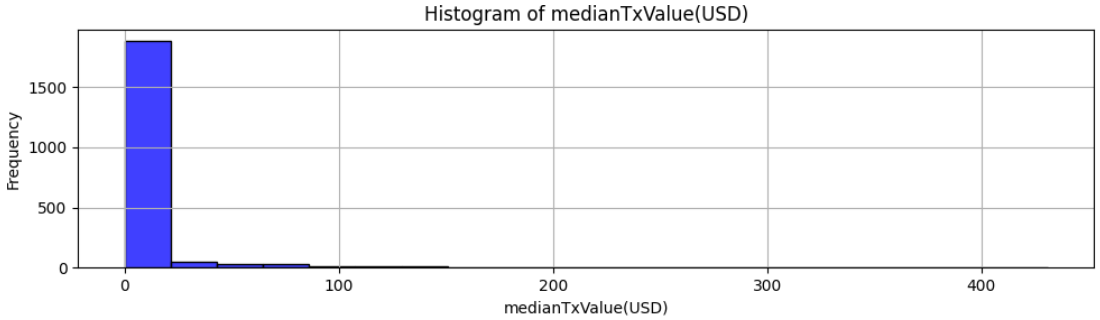

Kernel Density Estimation of generatedCoins
Density
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
generatedCoins
1e9

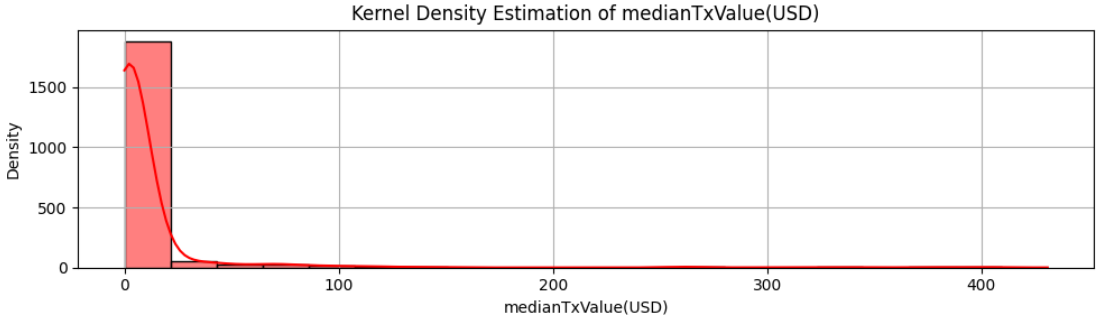

Histogram of fees
Frequency
2000
1500
1000
500
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
fees
1e7

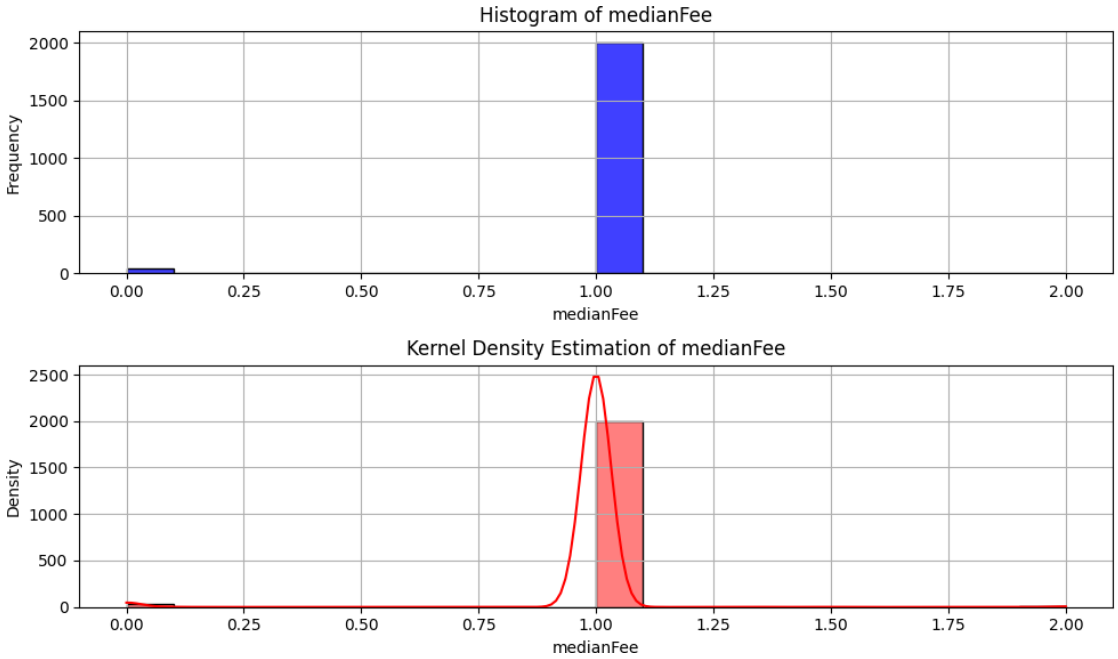

Kernel Density Estimation of fees
Density
8000
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
fees
1e7


Histogram of averageDifficulty
Frequency
averageDifficulty
1e6


Kernel Density Estimation of averageDifficulty
Density
averageDifficulty
1e6


Histogram of medianTxValue(USD)
Frequency
medianTxValue(USD)


Kernel Density Estimation of medianTxValue(USD)
Density
medianTxValue(USD)



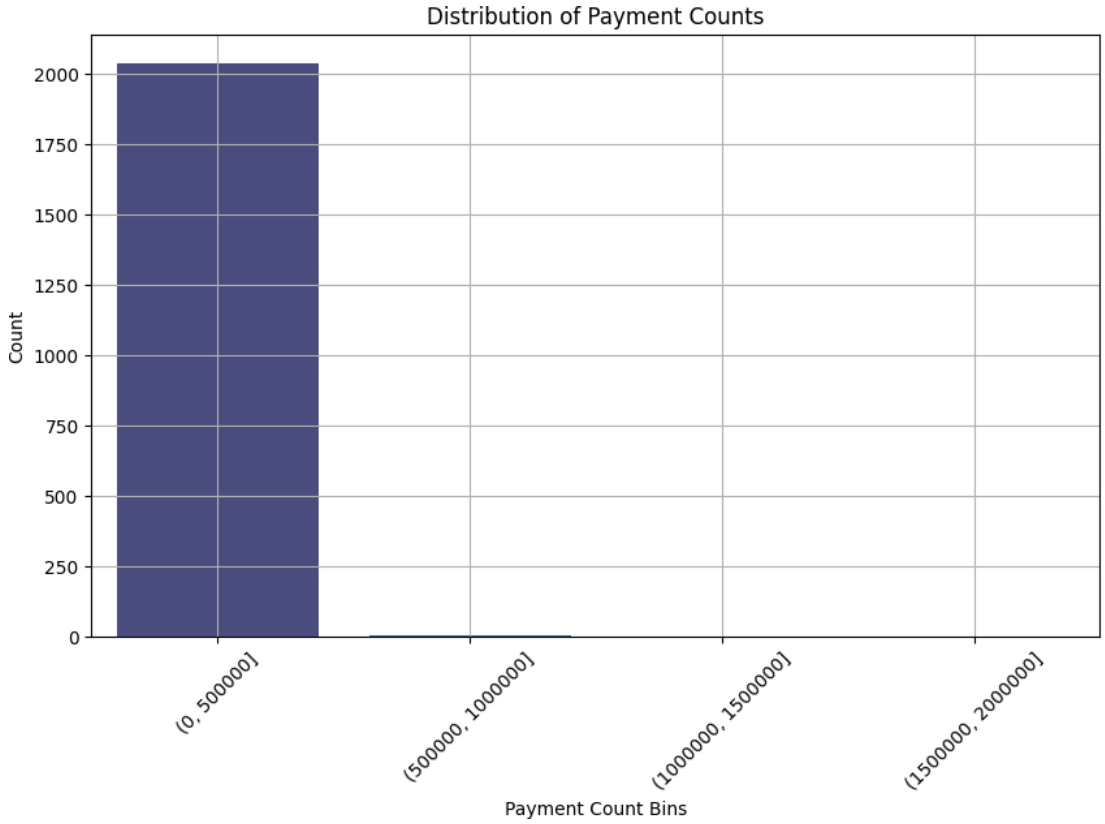
[102]:
```
df['paymentCount Bins'] = pd.cut(x=df['paymentCount'],␣
↪bins=[0,500000,1000000,1500000,2000000])
df['price(USD) Bins'] = pd.cut(x=df['price(USD)'],␣
↪bins=[0,5000,10000,15000,20000])
```

[103]:
```
import matplotlib.pyplot as plt
import seaborn as sns

payment_count_bins_counts = df['paymentCount Bins'].value_counts()

plt.figure(figsize=(10, 6))
sns.barplot(x=payment_count_bins_counts.index, y=payment_count_bins_counts.
↪values, palette='viridis')
plt.title('Distribution of Payment Counts')
plt.xlabel('Payment Count Bins')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

<ipython-input-103-989cc329841c>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=payment_count_bins_counts.index,
y=payment_count_bins_counts.values, palette='viridis')
```



[104]:
```
df=pd.read_csv("/content/eth.csv")
df.head()
```

[104]:

	date	txVolume(USD)	adjustedTxVolume(USD)	txCount	marketcap(USD)	\
0	2015-07-30	NaN	NaN	NaN	NaN	
1	2015-07-31	NaN	NaN	NaN	NaN	
2	2015-08-01	NaN	NaN	NaN	NaN	
3	2015-08-02	NaN	NaN	NaN	NaN	
4	2015-08-03	NaN	NaN	NaN	NaN	

	price(USD)	exchangeVolume(USD)	generatedCoins	fees	activeAddresses	\
0	NaN	NaN	39311.09375	NaN	NaN	
1	NaN	NaN	36191.71875	NaN	NaN	
2	NaN	NaN	27702.18750	NaN	NaN	
3	NaN	NaN	28227.18750	NaN	NaN	
4	NaN	NaN	27976.71875	NaN	NaN	

	medianTxValue(USD)	medianFee	averageDifficulty	paymentCount	blockSize	\

```
0                NaN         NaN        1.213355e+11          NaN    4449897
1                NaN         NaN        6.029419e+11          NaN    3994458
2                NaN         NaN        8.872542e+11          NaN    3044344
3                NaN         NaN        1.020254e+12          NaN    3112348
4                NaN         NaN        1.125837e+12          NaN    3099953

   blockCount
0        6911
1        6863
2        5293
3        5358
4        5280
```

[105]:
```
df.isnull().sum()/len(df)
```

[105]:
```
date                    0.00000
txVolume(USD)           0.00554
adjustedTxVolume(USD)   0.00554
txCount                 0.00554
marketcap(USD)          0.00554
price(USD)              0.00554
exchangeVolume(USD)     0.00554
generatedCoins          0.00000
fees                    0.00554
activeAddresses         0.00554
medianTxValue(USD)      0.00554
medianFee               0.00554
averageDifficulty       0.00000
paymentCount            0.00554
blockSize               0.00000
blockCount              0.00000
dtype: float64
```

[106]:
```
df[['txVolume(USD)','adjustedTxVolume(USD)','marketcap(USD)','price(USD)','exchangeVolume(USD)
  ↪describe(include='all')
```

[106]:
```
         txVolume(USD)  adjustedTxVolume(USD)  marketcap(USD)    price(USD)   \
count     1.436000e+03           1.436000e+03    1.436000e+03   1436.000000
mean      1.438503e+09           5.260592e+08    2.037973e+10    206.884796
std       2.896731e+09           7.770806e+08    2.520651e+10    257.223116
min       5.393942e+04           5.393942e+04    3.198436e+07      0.431589
25%       1.507173e+07           1.488566e+07    9.107410e+08     10.842500
50%       3.711814e+08           3.059312e+08    1.304593e+10    126.165000
75%       1.332677e+09           7.187061e+08    2.900864e+10    298.022500
max       3.513900e+10           6.332180e+09    1.355333e+11   1397.480000

         exchangeVolume(USD)   medianTxValue(USD)
```

count	1.436000e+03	1436.000000
mean	1.627336e+09	16.831795
std	2.502592e+09	42.811020
min	1.021280e+05	0.000000
25%	1.289878e+07	0.024766
50%	6.121870e+08	4.369963
75%	2.133069e+09	13.184015
max	1.866147e+10	592.522447

[107]:
```
df.fillna({'txVolume(USD)':df['txVolume(USD)'].median()},inplace=True)
df.fillna({'adjustedTxVolume(USD)':df['adjustedTxVolume(USD)'].
  ↪median()},inplace=True)
df.fillna({'marketcap(USD)':df['marketcap(USD)'].median()},inplace=True)
df.fillna({'exchangeVolume(USD)':df['exchangeVolume(USD)'].
  ↪median()},inplace=True)
df.fillna({'medianTxValue(USD)':df['medianTxValue(USD)'].median()},inplace=True)
df.fillna({'price(USD)':df['price(USD)'].median()},inplace=True)
```

[108]:
```
df.isnull().sum()
```

[108]:
```
date                    0
txVolume(USD)           0
adjustedTxVolume(USD)   0
txCount                 8
marketcap(USD)          0
price(USD)              0
exchangeVolume(USD)     0
generatedCoins          0
fees                    8
activeAddresses         8
medianTxValue(USD)      0
medianFee               8
averageDifficulty       0
paymentCount            8
blockSize               0
blockCount              0
dtype: int64
```

[109]:
```
df[['txCount','generatedCoins','fees','activeAddresses','averageDifficulty','paymentCount','bl
  ↪describe(include='all')
```

[109]:

	txCount	generatedCoins	fees	activeAddresses	\
count	1.436000e+03	1444.000000	1436.000000	1436.000000	
mean	3.455590e+05	24144.079056	345.359400	153193.362117	
std	3.236584e+05	6237.419918	504.012330	150497.567509	
min	1.329000e+03	12869.625000	5.507452	732.000000	
25%	4.155725e+04	20310.046875	33.591396	16555.000000	

50%	2.715380e+05	23808.750000	278.329601	128994.500000
75%	6.089498e+05	31087.617188	444.867459	265542.000000
max	1.349890e+06	39311.093750	5862.526226	719093.000000

	averageDifficulty	paymentCount	blockSize	blockCount	medianFee
count	1.444000e+03	1.436000e+03	1.444000e+03	1444.000000	1436.000000
mean	1.347804e+15	1.681079e+05	6.495785e+07	5636.655125	0.000548
std	1.312557e+15	1.651185e+05	5.869168e+07	717.927119	0.000431
min	1.213355e+11	1.213000e+03	2.555486e+06	2829.000000	0.000063
25%	5.773685e+13	2.867900e+04	9.058450e+06	5188.250000	0.000319
50%	1.274839e+15	1.546085e+05	4.621581e+07	5930.000000	0.000432
75%	2.625292e+15	2.638225e+05	1.219571e+08	6067.000000	0.000525
max	3.606036e+15	1.081291e+06	1.988823e+08	6911.000000	0.005674

[110]:
```
df.fillna({'txCount':df['txCount'].median()},inplace=True)
df.fillna({'generatedCoins':df['generatedCoins'].median()},inplace=True)
df.fillna({'fees':df['fees'].median()},inplace=True)
df.fillna({'activeAddresses':df['activeAddresses'].median()},inplace=True)
df.fillna({'averageDifficulty':df['averageDifficulty'].median()},inplace=True)
df.fillna({'paymentCount':df['paymentCount'].median()},inplace=True)
df.fillna({'blockSize':df['blockSize'].median()},inplace=True)
df.fillna({'blockCount':df['blockCount'].median()},inplace=True)
df.fillna({'medianFee':df['medianFee'].median()},inplace=True)
```

[111]:
```
df.isnull().sum()
```

[111] :
```
date                    0
txVolume(USD)           0
adjustedTxVolume(USD)   0
txCount                 0
marketcap(USD)          0
price(USD)              0
exchangeVolume(USD)     0
generatedCoins          0
fees                    0
activeAddresses         0
medianTxValue(USD)      0
medianFee               0
averageDifficulty       0
paymentCount            0
blockSize               0
blockCount              0
dtype: int64
```

[112]:
```
df[df.duplicated()]
```

[112] : Empty DataFrame
Columns: [date, txVolume(USD), adjustedTxVolume(USD), txCount, marketcap(USD), price(USD), exchangeVolume(USD), generatedCoins, fees, activeAddresses, medianTxValue(USD), medianFee, averageDifficulty, paymentCount, blockSize, blockCount]
Index: []

[113]:
```
df.head()
```

[113]:

	date	txVolume(USD)	adjustedTxVolume(USD)	txCount	marketcap(USD)	\
0	2015-07-30	3.711814e+08	3.059312e+08	271538.0	1.304593e+10	
1	2015-07-31	3.711814e+08	3.059312e+08	271538.0	1.304593e+10	
2	2015-08-01	3.711814e+08	3.059312e+08	271538.0	1.304593e+10	
3	2015-08-02	3.711814e+08	3.059312e+08	271538.0	1.304593e+10	
4	2015-08-03	3.711814e+08	3.059312e+08	271538.0	1.304593e+10	

	price(USD)	exchangeVolume(USD)	generatedCoins	fees	\
0	126.165	612186976.0	39311.09375	278.329601	
1	126.165	612186976.0	36191.71875	278.329601	
2	126.165	612186976.0	27702.18750	278.329601	
3	126.165	612186976.0	28227.18750	278.329601	
4	126.165	612186976.0	27976.71875	278.329601	

	activeAddresses	medianTxValue(USD)	medianFee	averageDifficulty	\
0	128994.5	4.369963	0.000432	1.213355e+11	
1	128994.5	4.369963	0.000432	6.029419e+11	
2	128994.5	4.369963	0.000432	8.872542e+11	
3	128994.5	4.369963	0.000432	1.020254e+12	
4	128994.5	4.369963	0.000432	1.125837e+12	

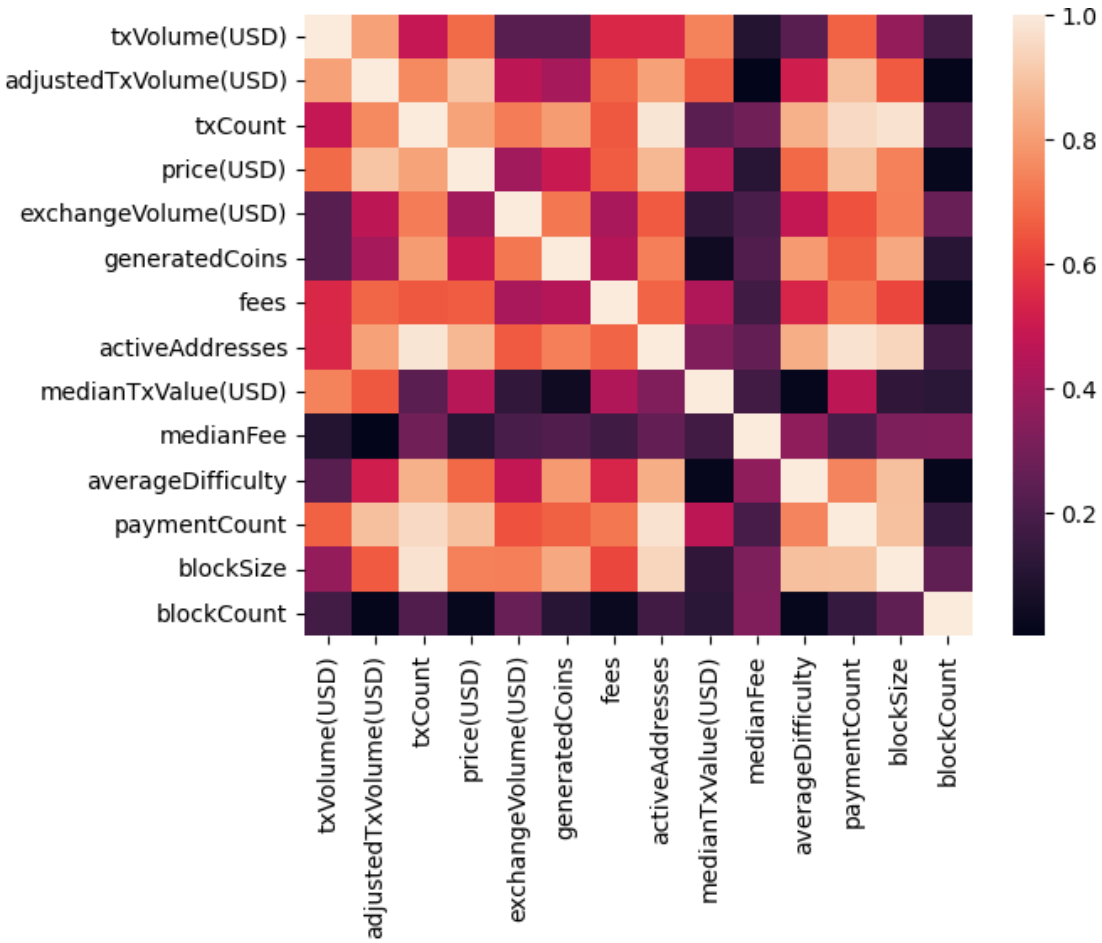
	paymentCount	blockSize	blockCount
0	154608.5	4449897	6911
1	154608.5	3994458	6863
2	154608.5	3044344	5293
3	154608.5	3112348	5358
4	154608.5	3099953	5280

[114]:
```
data = df
data.drop(["date"],axis=1,inplace=True)
```

[115]:
```
df.drop(["marketcap(USD)"],axis=1,inplace=True)
```

[116]:
```
sns.heatmap(data.corr().abs())
```

[116]: <Axes: >



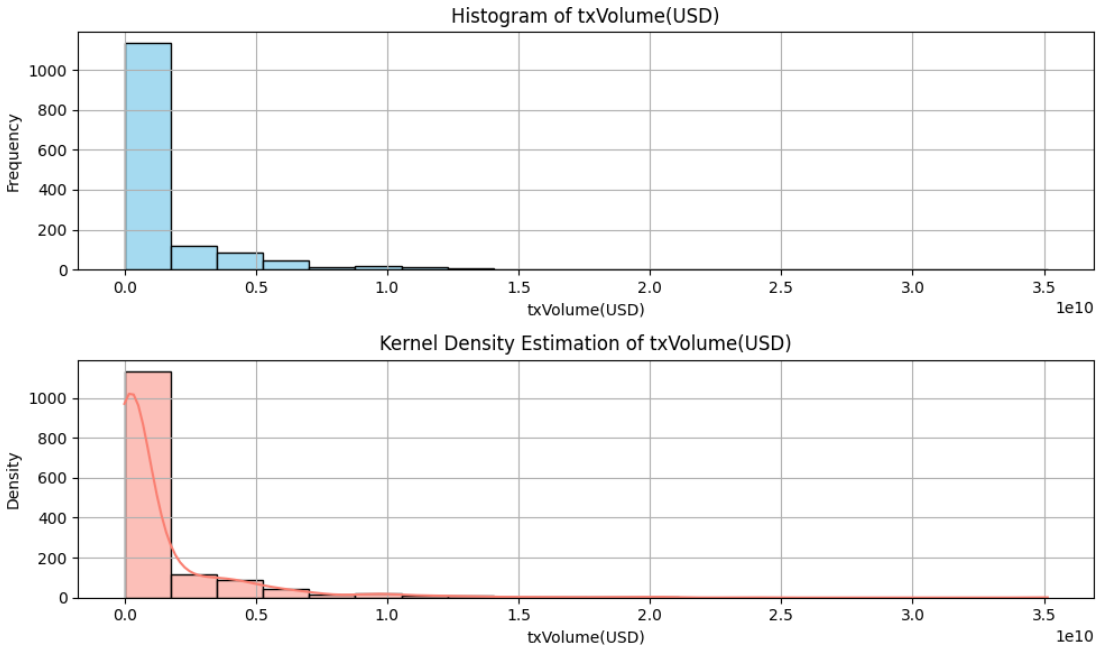
[117]:
```
print(df["paymentCount"].max())
print(df["paymentCount"].min())
```

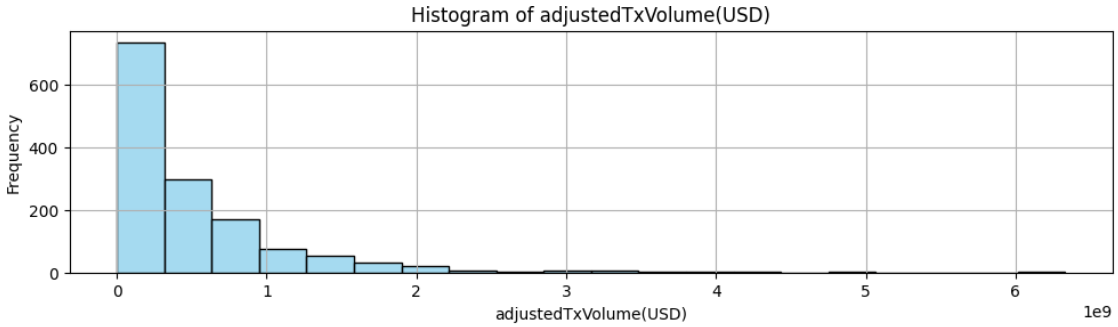
```
1081291.0
1213.0
```

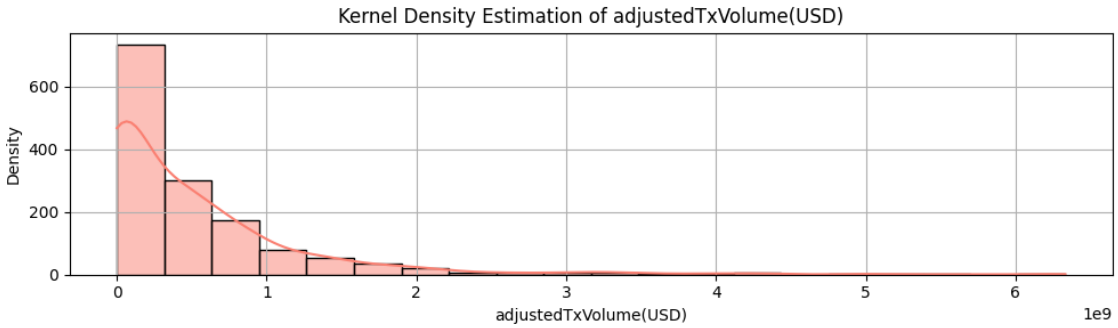
[118]:
```
df["paymentCount Bins"] = pd.cut(x=df["paymentCount"],␣
↪bins=[0,500000,1000000,1500000,2000000])
df["price(USD) Bins"] = pd.cut(x=df["price(USD)"],␣
↪bins=[0,5000,10000,15000,20000])
```

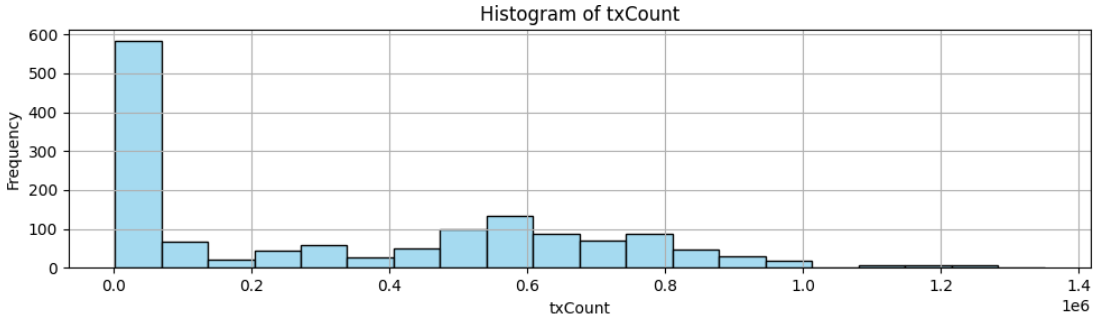
[119]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

numeric_columns = df.select_dtypes(include=["float64"]).columns
```

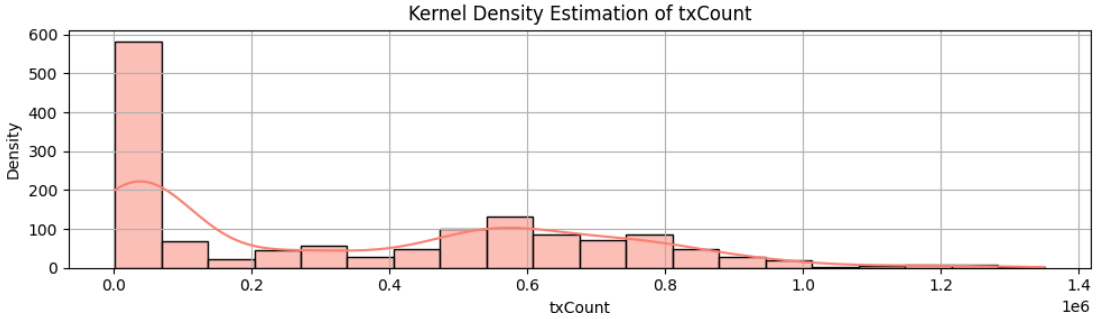
```
for column in numeric_columns:
    plt.figure(figsize=(10, 6))

    # Histogram
    plt.subplot(2, 1, 1)
    sns.histplot(df[column], kde=False, color='skyblue', bins=20)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.grid(True)

    # KDE plot
    plt.subplot(2, 1, 2)
    sns.histplot(df[column], kde=True, color='salmon', bins=20)
    plt.title(f'Kernel Density Estimation of {column}')
    plt.xlabel(column)
    plt.ylabel('Density')
    plt.grid(True)

    plt.tight_layout()

    plt.show()
```

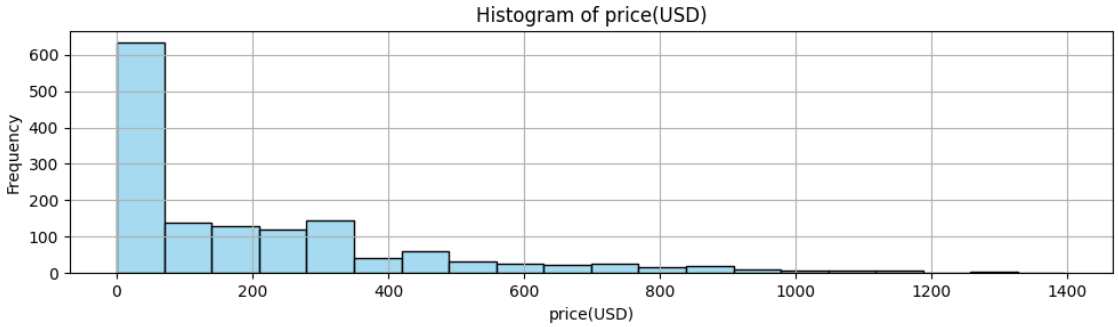


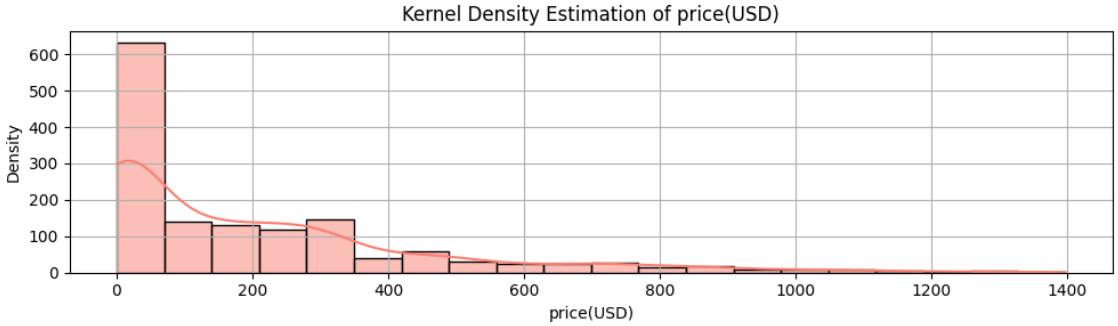

Histogram of adjustedTxVolume(USD)
Frequency
0
200
400
600
0
1
2
3
4
5
6
adjustedTxVolume(USD)
1e9

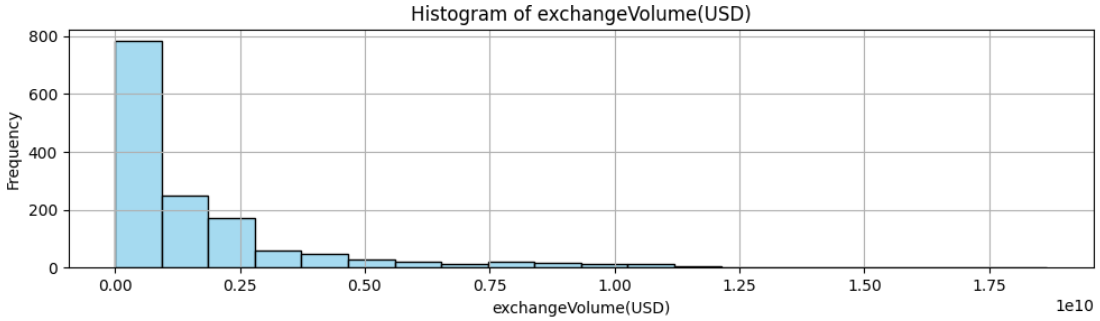

Kernel Density Estimation of adjustedTxVolume(USD)
Density
0
200
400
600
0
1
2
3
4
5
6
adjustedTxVolume(USD)
1e9

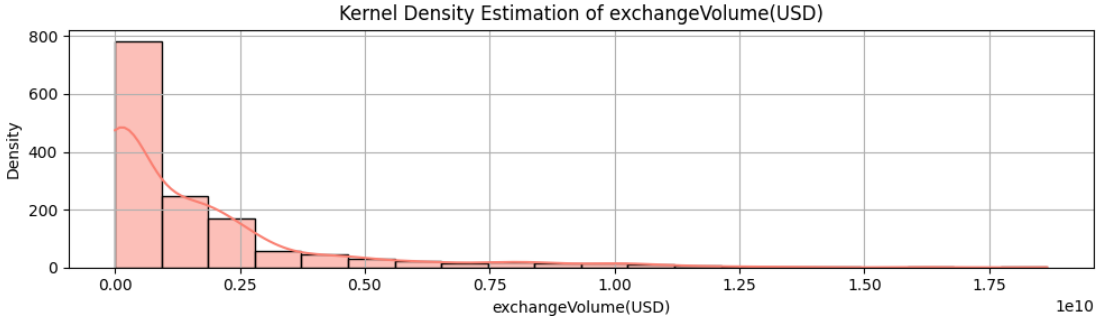

Histogram of txCount
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
txCount
1e6

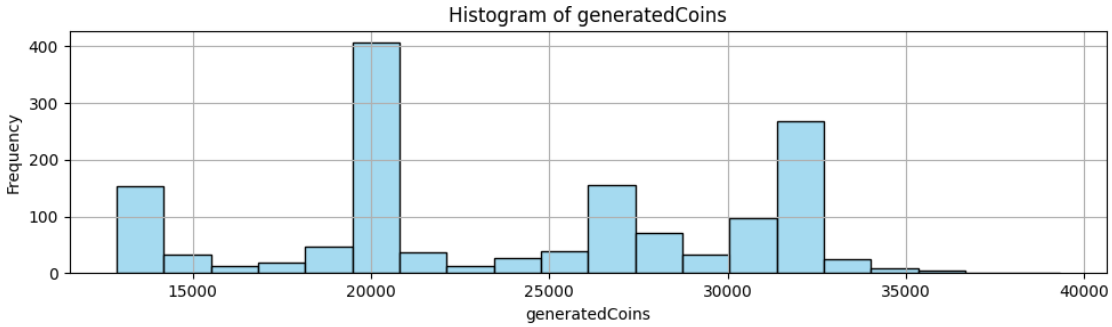

Kernel Density Estimation of txCount
Density
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
txCount
1e6

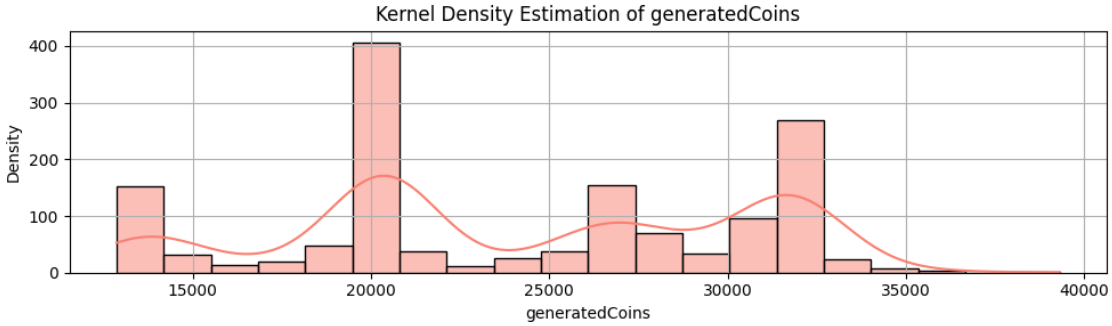

Histogram of price(USD)
Frequency
price(USD)

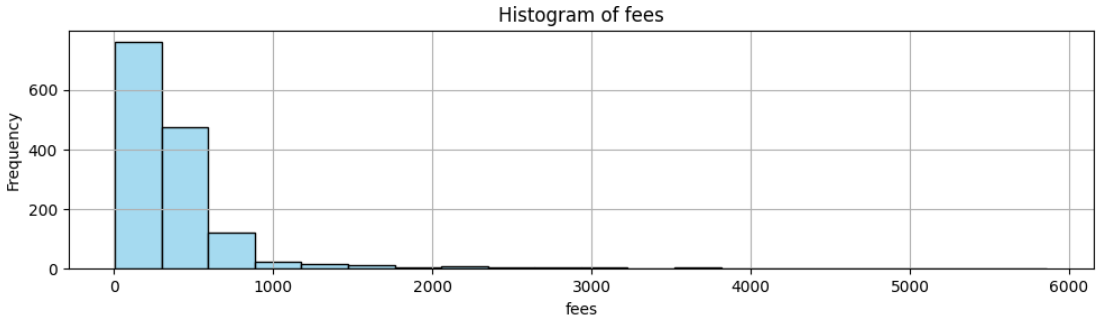

Kernel Density Estimation of price(USD)
Density
price(USD)

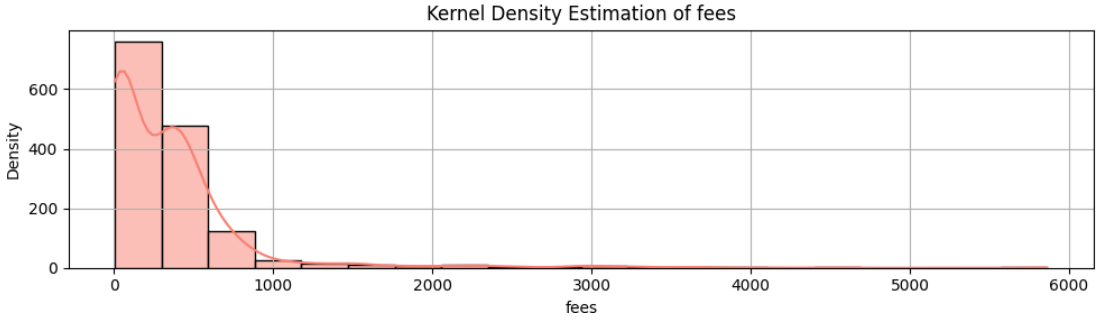

Histogram of exchangeVolume(USD)
Frequency
exchangeVolume(USD)
1e10

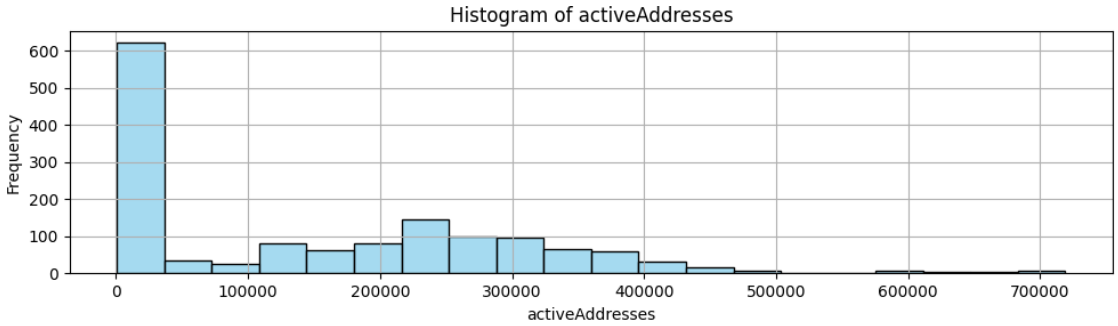

Kernel Density Estimation of exchangeVolume(USD)
Density
exchangeVolume(USD)
1e10

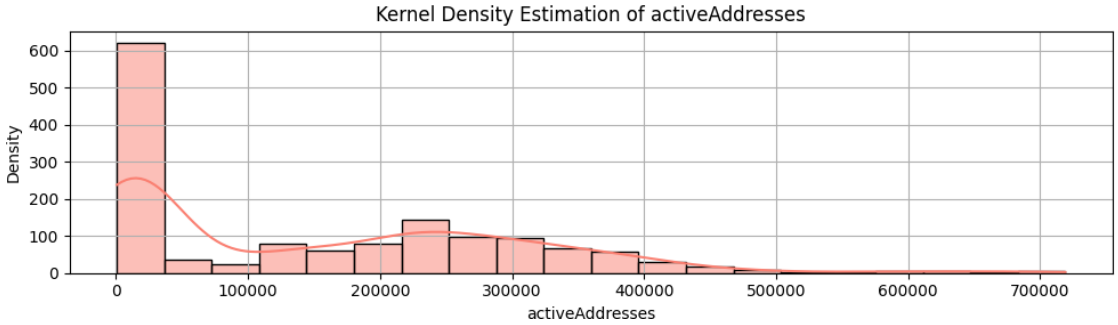

Histogram of generatedCoins
Frequency
0
100
200
300
400
15000
20000
25000
30000
35000
40000
generatedCoins

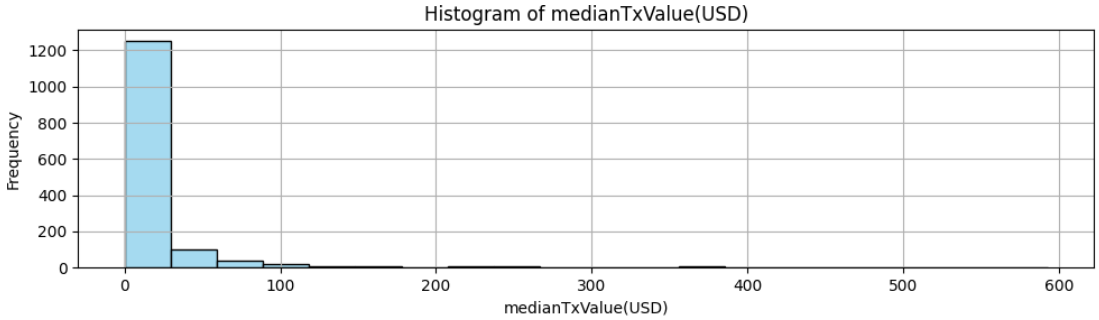

Kernel Density Estimation of generatedCoins
Density
0
100
200
300
400
15000
20000
25000
30000
35000
40000
generatedCoins

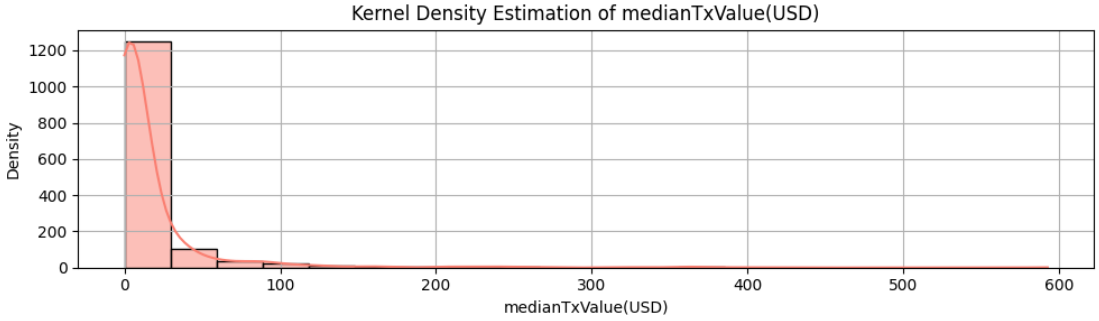

Histogram of fees
Frequency
0
200
400
600
0
1000
2000
3000
4000
5000
6000
fees

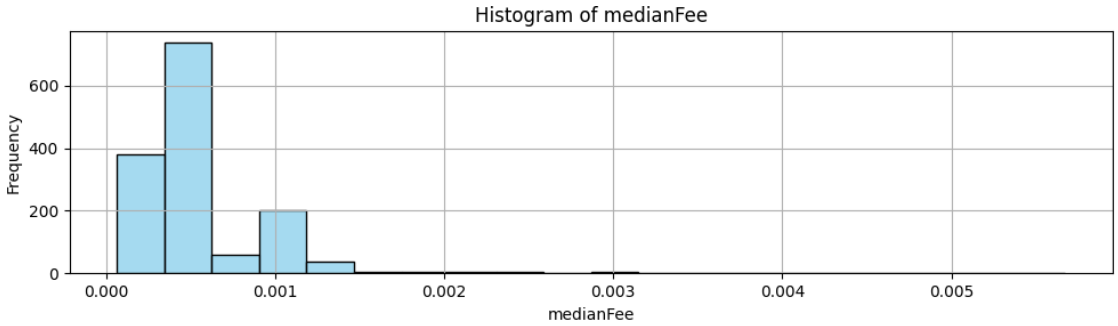

Kernel Density Estimation of fees
Density
0
200
400
600
0
1000
2000
3000
4000
5000
6000
fees

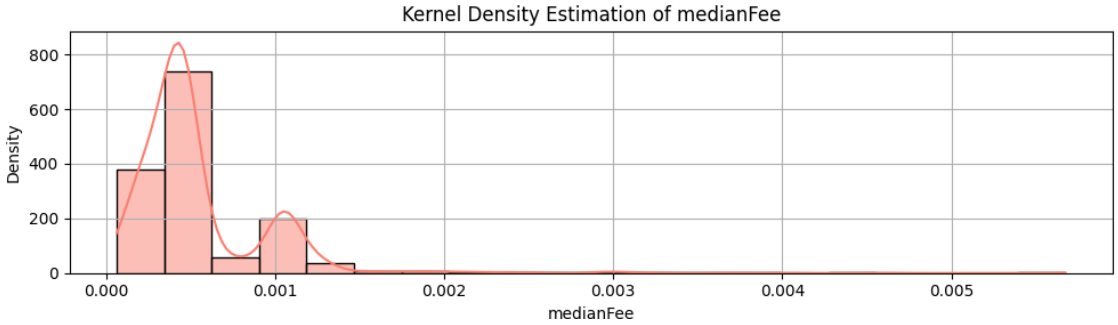

Histogram of activeAddresses
Frequency
activeAddresses
0
100000
200000
300000
400000
500000
600000
700000

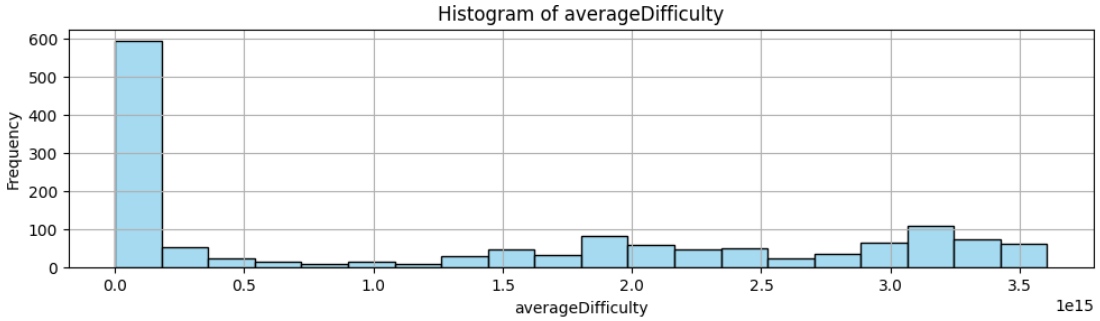

Kernel Density Estimation of activeAddresses
Density
activeAddresses
0
100000
200000
300000
400000
500000
600000
700000

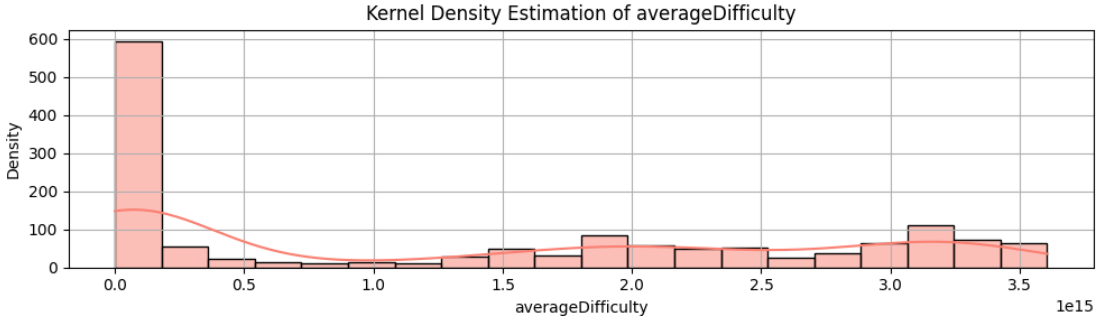

Histogram of medianTxValue(USD)
Frequency
medianTxValue(USD)
0
100
200
300
400
500
600

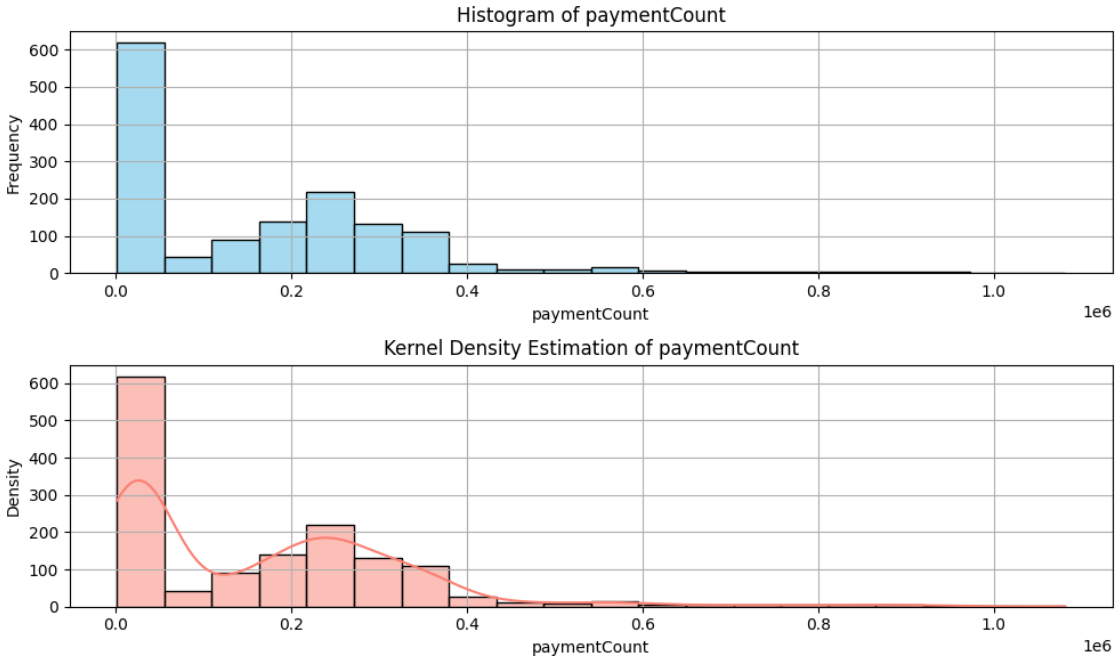

Kernel Density Estimation of medianTxValue(USD)
Density
medianTxValue(USD)
0
100
200
300
400
500
600


Histogram of medianFee
Frequency
0
200
400
600
0.000
0.001
0.002
0.003
0.004
0.005
medianFee


Kernel Density Estimation of medianFee
Density
0
200
400
600
800
0.000
0.001
0.002
0.003
0.004
0.005
medianFee


Histogram of averageDifficulty
Frequency
0
100
200
300
400
500
600
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e15
averageDifficulty


Kernel Density Estimation of averageDifficulty
Density
0
100
200
300
400
500
600
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e15
averageDifficulty

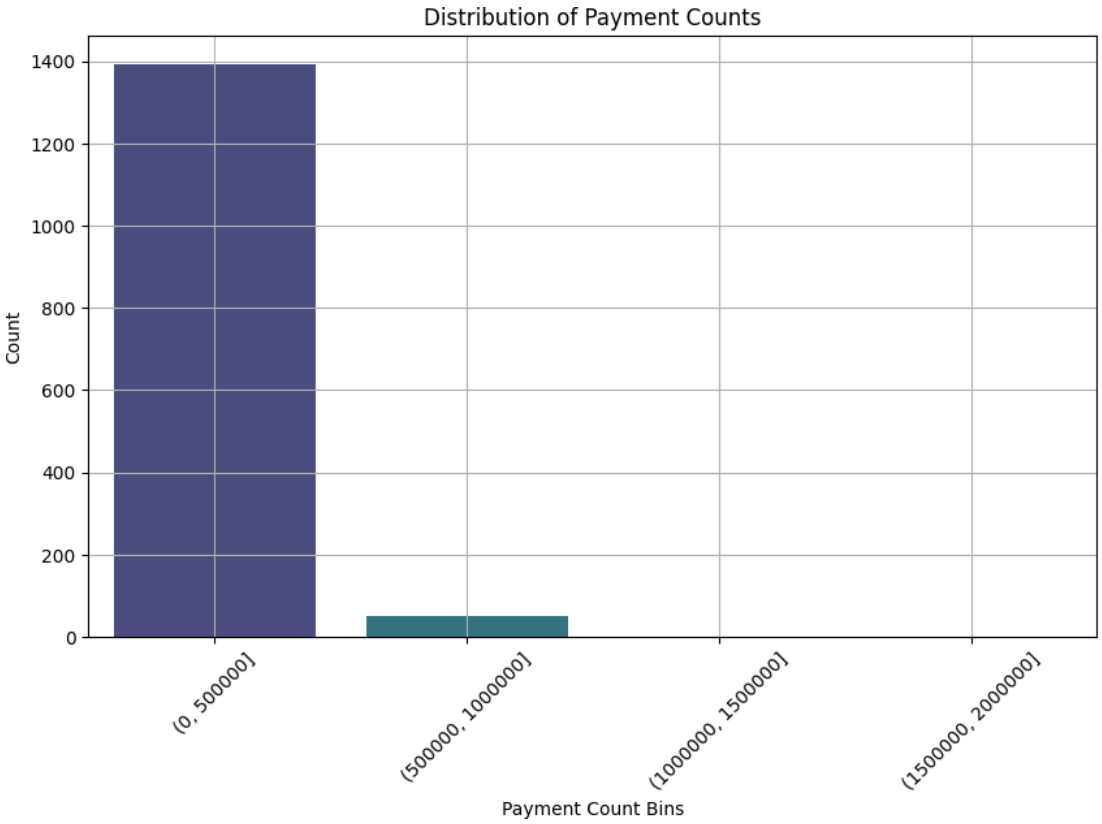


[120]:
```
import matplotlib.pyplot as plt
import seaborn as sns

payment_count_bins_counts = df['paymentCount Bins'].value_counts()

plt.figure(figsize=(10, 6))
sns.barplot(x=payment_count_bins_counts.index, y=payment_count_bins_counts.
↪values, palette='viridis')
plt.title('Distribution of Payment Counts')
plt.xlabel('Payment Count Bins')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

<ipython-input-120-989cc329841c>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
  sns.barplot(x=payment_count_bins_counts.index,
y=payment_count_bins_counts.values, palette='viridis')
```


Distribution of Payment Counts
Count
1400
1200
1000
800
600
400
200
0
(0, 500000]
(500000, 1000000]
(1000000, 1500000]
(1500000, 2000000]
Payment Count Bins

[120]: