In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('c:/Users/saiku/Downloads/NSE_BANKING_SECTOR.csv')
```

In [3]:
```
df
```

Out[3]:

	DATE	SYMBOL	SERIES	PREV CLOSE	OPEN	HIGH	LOW	LAST	CLOSE	VWAP
0	2016-01-01	HDFC	EQ	1263.75	1261.00	1266.90	1250.65	1257.80	1258.45	1258.39
1	2016-01-04	HDFC	EQ	1258.45	1250.00	1253.90	1212.05	1217.15	1216.70	1227.55
2	2016-01-05	HDFC	EQ	1216.70	1229.90	1233.45	1206.50	1208.15	1209.40	1219.50
3	2016-01-06	HDFC	EQ	1209.40	1209.60	1220.75	1202.40	1207.55	1209.30	1210.81
4	2016-01-07	HDFC	EQ	1209.30	1198.85	1203.55	1175.00	1176.35	1179.45	1186.35
...	...	...	...	...	...	...	...	...	...	...
26	2021-05-24	DHANBANK	EQ	14.30	14.40	14.70	14.35	14.55	14.55	14.52
27	2021-05-25	DHANBANK	EQ	14.55	14.60	17.45	14.40	16.55	16.60	16.67
28	2021-05-26	DHANBANK	EQ	16.60	16.75	16.75	15.80	15.95	15.95	16.06
29	2021-05-27	DHANBANK	EQ	15.95	15.95	16.10	15.35	15.75	15.60	15.74
30	2021-05-28	DHANBANK	EQ	15.60	15.60	15.80	14.90	15.20	15.00	15.25

31 rows × 15 columns

Data Exploration

In [4]:
```
df.head()
```

Out[4]:

	DATE	SYMBOL	SERIES	PREV CLOSE	OPEN	HIGH	LOW	LAST	CLOSE	VWAP
0	2016-01-01	HDFC	EQ	1263.75	1261.00	1266.90	1250.65	1257.80	1258.45	1258.39
1	2016-01-04	HDFC	EQ	1258.45	1250.00	1253.90	1212.05	1217.15	1216.70	1227.55
2	2016-01-05	HDFC	EQ	1216.70	1229.90	1233.45	1206.50	1208.15	1209.40	1219.50
3	2016-01-06	HDFC	EQ	1209.40	1209.60	1220.75	1202.40	1207.55	1209.30	1210.81
4	2016-01-07	HDFC	EQ	1209.30	1198.85	1203.55	1175.00	1176.35	1179.45	1186.35

In [5]:
```
df.tail()
```

Out[5]:

	DATE	SYMBOL	SERIES	PREV CLOSE	OPEN	HIGH	LOW	LAST	CLOSE	VWAP	VC
41226	2021-05-24	DHANBANK	EQ	14.30	14.40	14.70	14.35	14.55	14.55	14.52	1(
41227	2021-05-25	DHANBANK	EQ	14.55	14.60	17.45	14.40	16.55	16.60	16.67	164
41228	2021-05-26	DHANBANK	EQ	16.60	16.75	16.75	15.80	15.95	15.95	16.06	22
41229	2021-05-27	DHANBANK	EQ	15.95	15.95	16.10	15.35	15.75	15.60	15.74	14
41230	2021-05-28	DHANBANK	EQ	15.60	15.60	15.80	14.90	15.20	15.00	15.25	16

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41231 entries, 0 to 41230
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   DATE                41231 non-null  object 
 1   SYMBOL              41231 non-null  object 
 2   SERIES              41231 non-null  object 
 3   PREV CLOSE          41231 non-null  float64
 4   OPEN                41231 non-null  float64
 5   HIGH                41231 non-null  float64
 6   LOW                 41231 non-null  float64
 7   LAST                41231 non-null  float64
 8   CLOSE               41231 non-null  float64
 9   VWAP                41231 non-null  float64
 10  VOLUME              41231 non-null  int64  
 11  TURNOVER            41231 non-null  float64
 12  TRADES              41231 non-null  int64  
 13  DELIVERABLE VOLUME  41231 non-null  int64  
 14  %DELIVERBLE         41231 non-null  float64
dtypes: float64(9), int64(3), object(3)
memory usage: 4.7+ MB
```

In [7]:
```
df.describe()
```

Out[7]:

	PREV CLOSE	OPEN	HIGH	LOW	LAST	CLOSE	
ount	41231.000000	41231.000000	41231.000000	41231.000000	41231.000000	41231.000000	4
nean	291.962753	292.350947	296.518484	287.723448	291.993606	292.013088	
std	452.541028	452.967892	458.224757	447.069432	452.717343	452.732064	
min	4.900000	4.950000	4.950000	4.800000	4.900000	4.900000	
25%	37.150000	37.300000	37.975000	36.450000	37.100000	37.100000	
50%	101.900000	102.000000	103.800000	99.800000	101.750000	101.850000	
75%	305.675000	306.125000	311.400000	301.050000	305.775000	305.675000	
max	2860.450000	2871.000000	2896.000000	2838.000000	2861.550000	2860.450000	

In [8]:
```
df.describe().T
```

Out[8]:

	count	mean	std	min	25%	5
PREV CLOSE	41231.0	2.919628e+02	4.525410e+02	4.900000e+00	3.715000e+01	1.019000e
OPEN	41231.0	2.923509e+02	4.529679e+02	4.950000e+00	3.730000e+01	1.020000e
HIGH	41231.0	2.965185e+02	4.582248e+02	4.950000e+00	3.797500e+01	1.038000e
LOW	41231.0	2.877234e+02	4.470694e+02	4.800000e+00	3.645000e+01	9.980000e
LAST	41231.0	2.919936e+02	4.527173e+02	4.900000e+00	3.710000e+01	1.017500e
CLOSE	41231.0	2.920131e+02	4.527321e+02	4.900000e+00	3.710000e+01	1.018500e
VWAP	41231.0	2.921607e+02	4.526553e+02	4.910000e+00	3.723000e+01	1.020200e
VOLUME	41231.0	1.042650e+07	2.953972e+07	9.194000e+03	8.216770e+05	2.777826e
TURNOVER	41231.0	1.953615e+14	4.038675e+14	1.681628e+10	5.730684e+12	4.025961e
TRADES	41231.0	5.221812e+04	8.851021e+04	9.400000e+01	5.398000e+03	1.928000e
DELIVERABLE VOLUME	41231.0	3.026935e+06	9.387528e+06	7.392000e+03	3.457530e+05	9.584380e
%DELIVERBLE	41231.0	4.154165e-01	1.961222e-01	2.010000e-02	2.527000e-01	4.147000e

In [9]:
```
df.shape
```

Out[9]:
```
(41231, 15)
```

In [10]:
```
df.dtypes
```

Out[10]:
```
DATE                   object
SYMBOL                 object
SERIES                 object
PREV CLOSE            float64
OPEN                  float64
HIGH                  float64
LOW                   float64
LAST                  float64
CLOSE                 float64
VWAP                  float64
VOLUME                  int64
TURNOVER              float64
TRADES                  int64
DELIVERABLE VOLUME      int64
%DELIVERBLE           float64
dtype: object
```

In [11]:
```
df.columns
```

Out[11]:
```
Index(['DATE', 'SYMBOL', 'SERIES', 'PREV CLOSE', 'OPEN', 'HIGH', 'LOW',
'LAST',
       'CLOSE', 'VWAP', 'VOLUME', 'TURNOVER', 'TRADES', 'DELIVERABLE VOLU
ME',
       '%DELIVERBLE'],
      dtype='object')
```

Data Cleaning

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
DATE                   0
SYMBOL                 0
SERIES                 0
PREV CLOSE             0
OPEN                   0
HIGH                   0
LOW                    0
LAST                   0
CLOSE                  0
VWAP                   0
VOLUME                 0
TURNOVER               0
TRADES                 0
DELIVERABLE VOLUME     0
%DELIVERBLE            0
dtype: int64
```

In [13]:
```
df.isna().sum()
```

Out[13]:
```
DATE                   0
SYMBOL                 0
SERIES                 0
PREV CLOSE             0
OPEN                   0
HIGH                   0
LOW                    0
LAST                   0
CLOSE                  0
VWAP                   0
VOLUME                 0
TURNOVER               0
TRADES                 0
DELIVERABLE VOLUME     0
%DELIVERBLE            0
dtype: int64
```

In [14]:
```
# Renaming the column which has all the BAnks

df.rename(columns={'SYMBOL':'BANK_NAME'},inplace=True)
```

In [15]:
```
df.head() #successfully we renamed the column
```

Out[15]:

	DATE	BANK_NAME	SERIES	PREV CLOSE	OPEN	HIGH	LOW	LAST	CLOSE	VWAP
0	2016-01-01	HDFC	EQ	1263.75	1261.00	1266.90	1250.65	1257.80	1258.45	1258.39
1	2016-01-04	HDFC	EQ	1258.45	1250.00	1253.90	1212.05	1217.15	1216.70	1227.55
2	2016-01-05	HDFC	EQ	1216.70	1229.90	1233.45	1206.50	1208.15	1209.40	1219.50
3	2016-01-06	HDFC	EQ	1209.40	1209.60	1220.75	1202.40	1207.55	1209.30	1210.81
4	2016-01-07	HDFC	EQ	1209.30	1198.85	1203.55	1175.00	1176.35	1179.45	1186.35

In [16]:
```
#Checking for duplicates
df.nunique()
```

Out[16]:
```
DATE                    1337
BANK_NAME                 36
SERIES                     1
PREV CLOSE             13706
OPEN                   11301
HIGH                   12440
LOW                    12590
LAST                   12482
CLOSE                  13707
VWAP                   25564
VOLUME                 41127
TURNOVER               41231
TRADES                 30465
DELIVERABLE VOLUME     40938
%DELIVERBLE             8019
dtype: int64
```

In [17]:
```
print(df.BANK_NAME.unique())
print(df.BANK_NAME.nunique())

# There are no Duplicates
```

```
['HDFC' 'ICICIBANK' 'SBIN' 'KOTAKBANK' 'AXISBANK' 'INDUSINDBK'
 'BANDHANBNK' 'PNB' 'BANKBARODA' 'IDBI' 'IDFCBANK' 'IDFCFIRSTB' 'YESBANK'
 'AUBANK' 'IOB' 'CANBK' 'BANKINDIA' 'UNIONBANK' 'FEDERALBNK' 'MAHABANK'
 'INDIANB' 'UCOBANK' 'CUB' 'RBLBANK' 'CENTRALBK' 'PSB' 'EQUITASBNK'
 'CSBBANK' 'UJJIVANSFB' 'KARURVYSYA' 'DCBBANK' 'SURYODAY' 'SOUTHBANK'
 'J&KBANK' 'KTKBANK' 'DHANBANK']
36
```

In [18]:

```
df
```

Out[18]:

	DATE	BANK_NAME	SERIES	PREV CLOSE	OPEN	HIGH	LOW	LAST	CLOSE
0	2016-01-01	HDFC	EQ	1263.75	1261.00	1266.90	1250.65	1257.80	1258.45
1	2016-01-04	HDFC	EQ	1258.45	1250.00	1253.90	1212.05	1217.15	1216.70
2	2016-01-05	HDFC	EQ	1216.70	1229.90	1233.45	1206.50	1208.15	1209.40
3	2016-01-06	HDFC	EQ	1209.40	1209.60	1220.75	1202.40	1207.55	1209.30
4	2016-01-07	HDFC	EQ	1209.30	1198.85	1203.55	1175.00	1176.35	1179.45
...	...	...	...	...	...	...	...	...	...
41226	2021-05-24	DHANBANK	EQ	14.30	14.40	14.70	14.35	14.55	14.55
41227	2021-05-25	DHANBANK	EQ	14.55	14.60	17.45	14.40	16.55	16.60
41228	2021-05-26	DHANBANK	EQ	16.60	16.75	16.75	15.80	15.95	15.95
41229	2021-05-27	DHANBANK	EQ	15.95	15.95	16.10	15.35	15.75	15.60
41230	2021-05-28	DHANBANK	EQ	15.60	15.60	15.80	14.90	15.20	15.00

41231 rows × 15 columns

In [19]:

```
df['DATE'] = pd.to_datetime(df['DATE'])
```

In [20]:

```
# We are Extracting only Year from Date column so we can do analysis based
df['YEAR'] = df['DATE'].dt.year
```

In [21]:
```
df.head()
```

Out[21]:

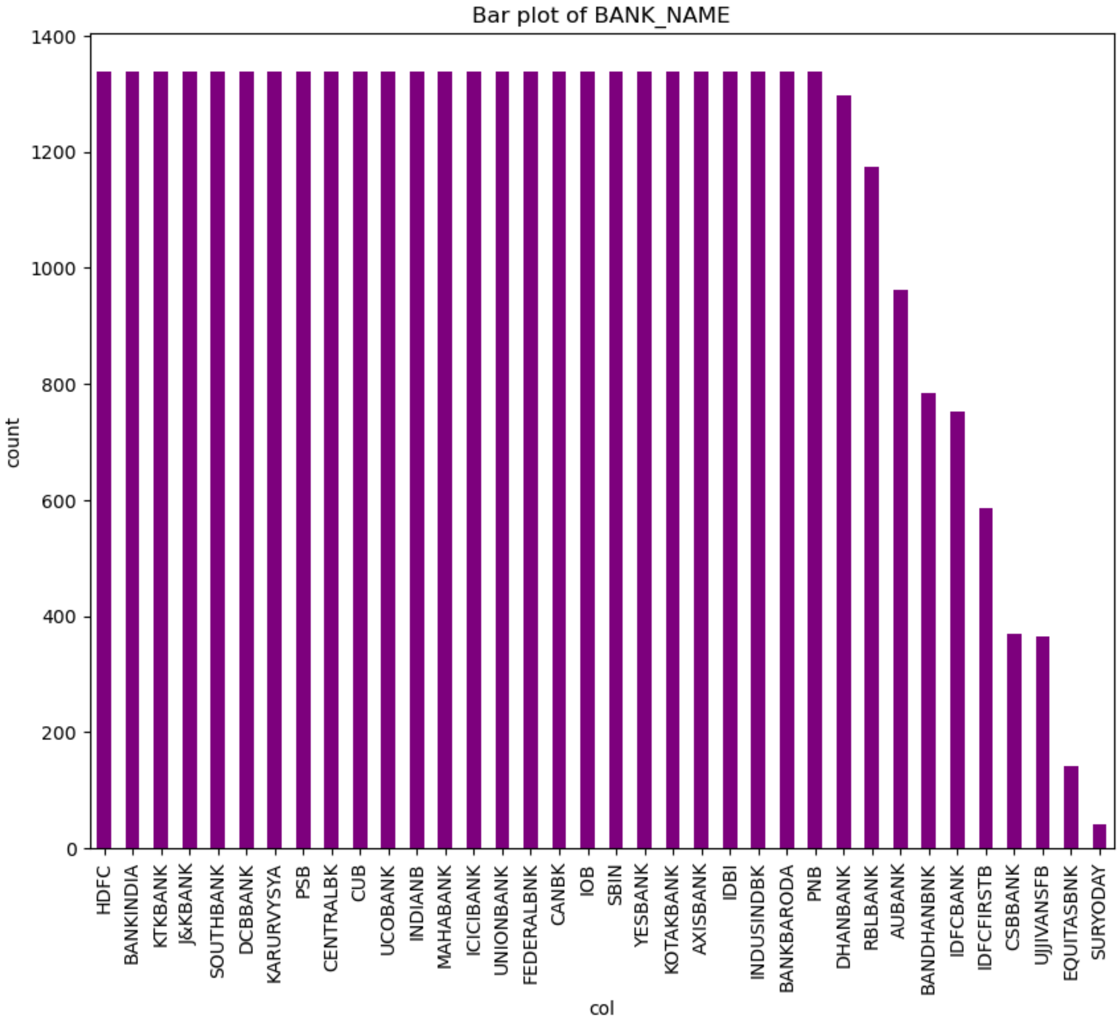
	DATE	BANK_NAME	SERIES	PREV CLOSE	OPEN	HIGH	LOW	LAST	CLOSE	VW.
0	2016-01-01	HDFC	EQ	1263.75	1261.00	1266.90	1250.65	1257.80	1258.45	1258.
1	2016-01-04	HDFC	EQ	1258.45	1250.00	1253.90	1212.05	1217.15	1216.70	1227.
2	2016-01-05	HDFC	EQ	1216.70	1229.90	1233.45	1206.50	1208.15	1209.40	1219.
3	2016-01-06	HDFC	EQ	1209.40	1209.60	1220.75	1202.40	1207.55	1209.30	1210.
4	2016-01-07	HDFC	EQ	1209.30	1198.85	1203.55	1175.00	1176.35	1179.45	1186.

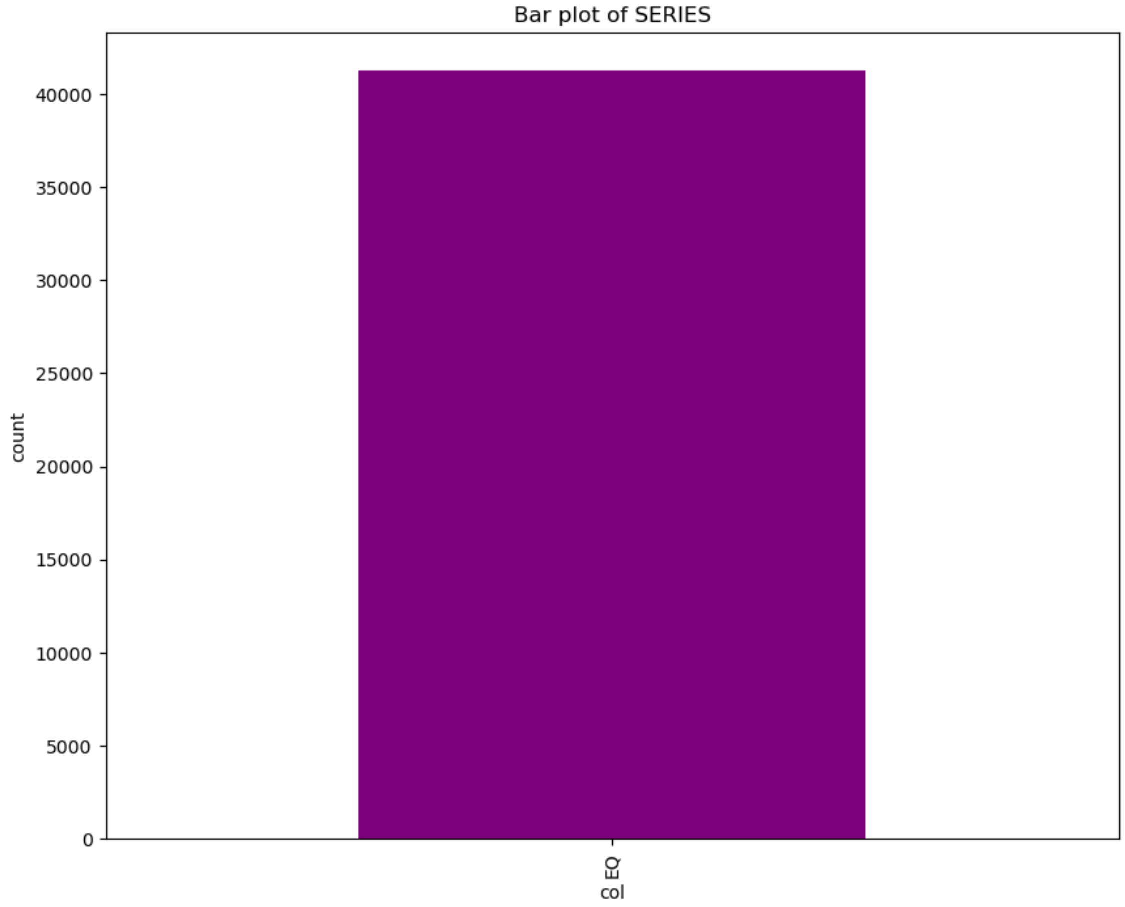
Sepearting Numerical and categorical columns

In [22]:
```
categorical_columns = df.select_dtypes(include=['object']).columns
numerical_columns = df.select_dtypes(include=np.number).columns.tolist()
print('Categorical Variable: ', categorical_columns)
print('Numerical Variables : ', numerical_columns)
```

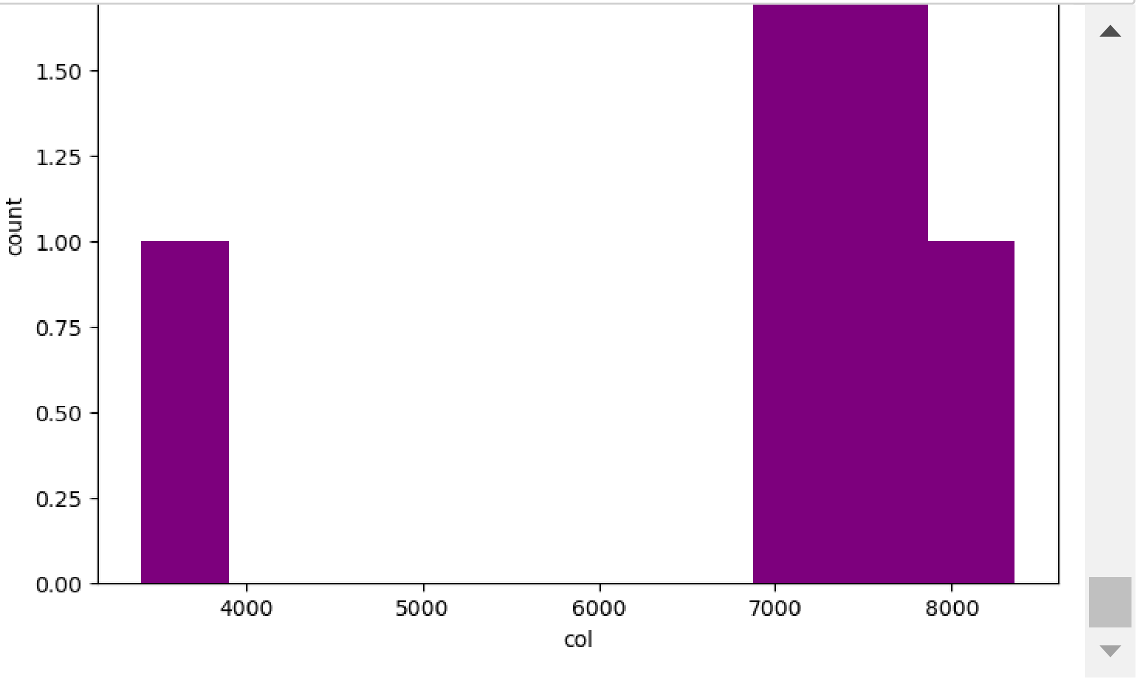
```
Categorical Variable:  Index(['BANK_NAME', 'SERIES'], dtype='object')
Numerical Variables :  ['PREV CLOSE', 'OPEN', 'HIGH', 'LOW', 'LAST', 'CLO
SE', 'VWAP', 'VOLUME', 'TURNOVER', 'TRADES', 'DELIVERABLE VOLUME', '%DELI
VERBLE', 'YEAR']
```

In [23]:
```
#univerient Analysis for Categorical Columns
for col in categorical_columns:
    plt.figure(figsize=(10,8))
    df[col].value_counts().plot(kind='bar',color='purple')
    plt.title(f'Bar plot of {col}')
    plt.xlabel('col')
    plt.ylabel('count')
    plt.show()
```





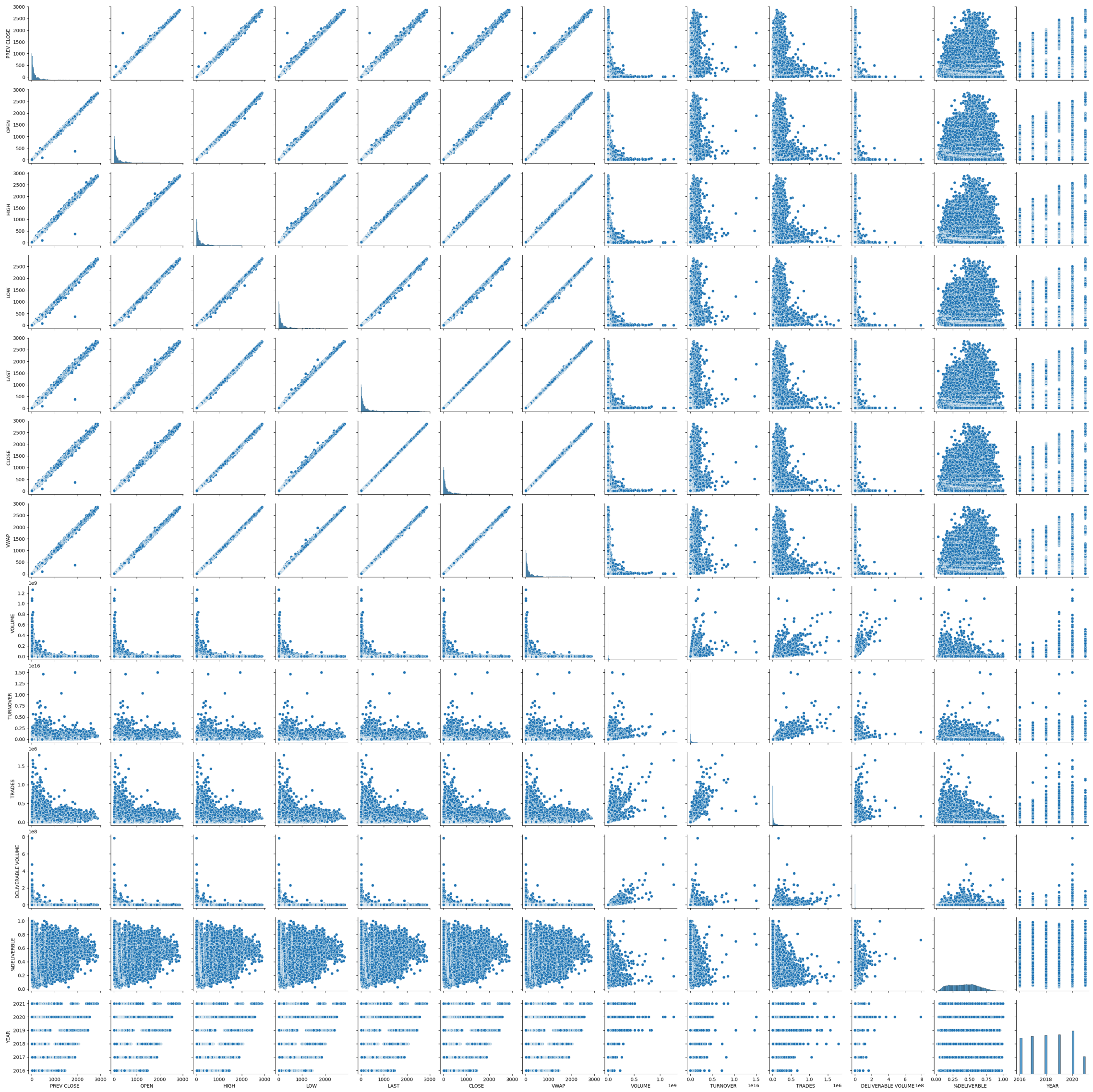
In [24]:
```
for num in numerical_columns:
    plt.figure(figsize=(8,6))
    df[num].value_counts().plot(kind='hist',color='purple')
    plt.title(f'Histogram of {num}')
    plt.xlabel('col')
    plt.ylabel('count')
```



Visualisation

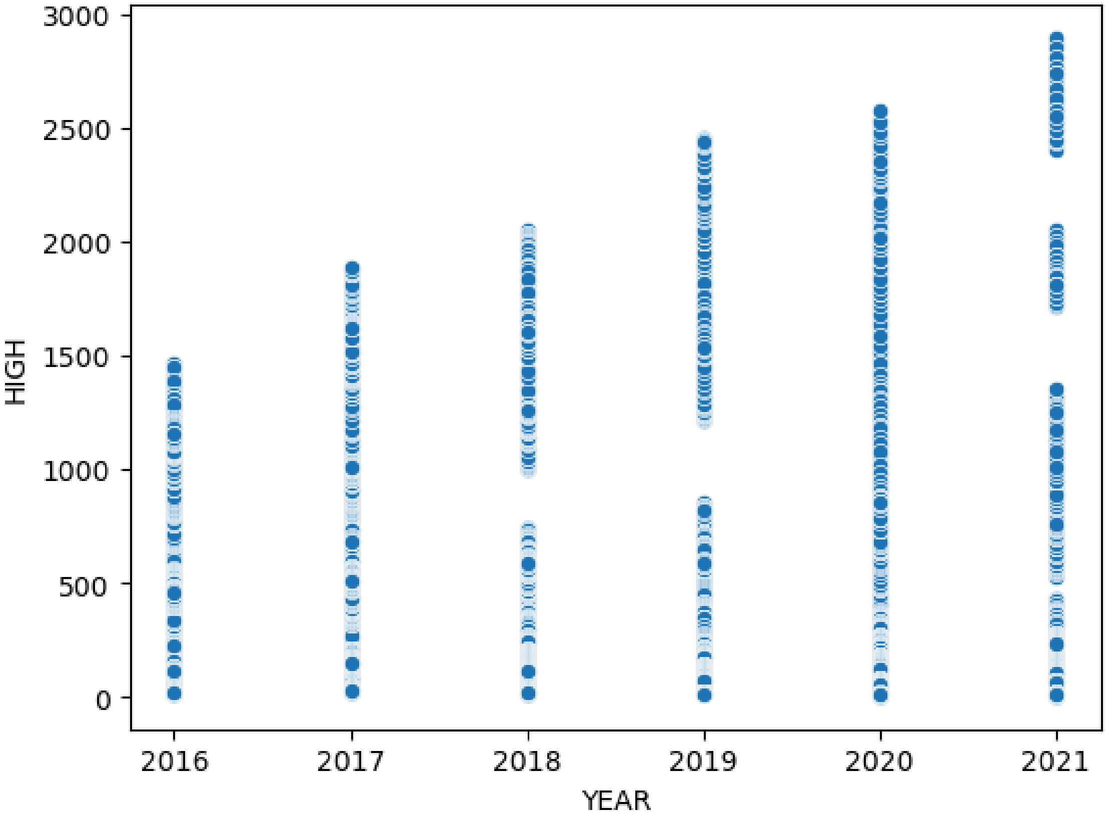
In [25]:
```
sns.pairplot(df)
```

Out[25]:
```
<seaborn.axisgrid.PairGrid at 0x1c89be57750>
```



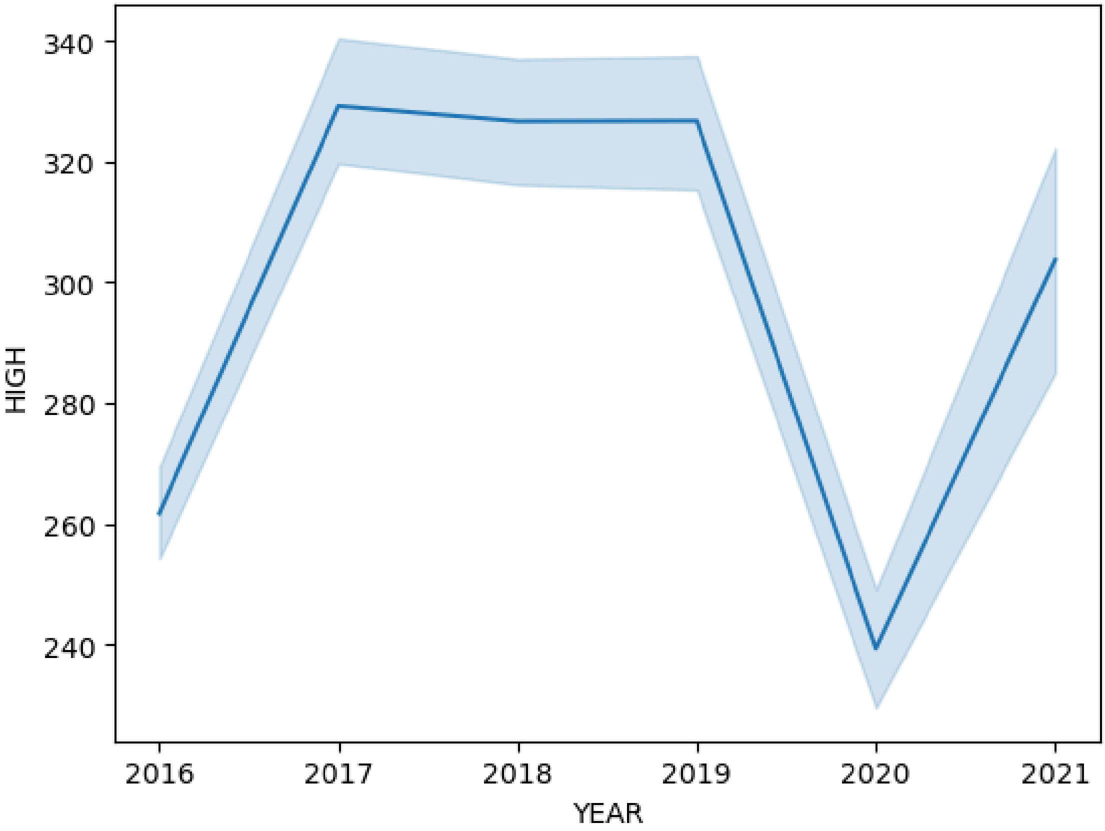
In [26]:
```
sns.scatterplot(x='YEAR',y='HIGH',data=df)
```

Out[26]:
```
<Axes: xlabel='YEAR', ylabel='HIGH'>
```



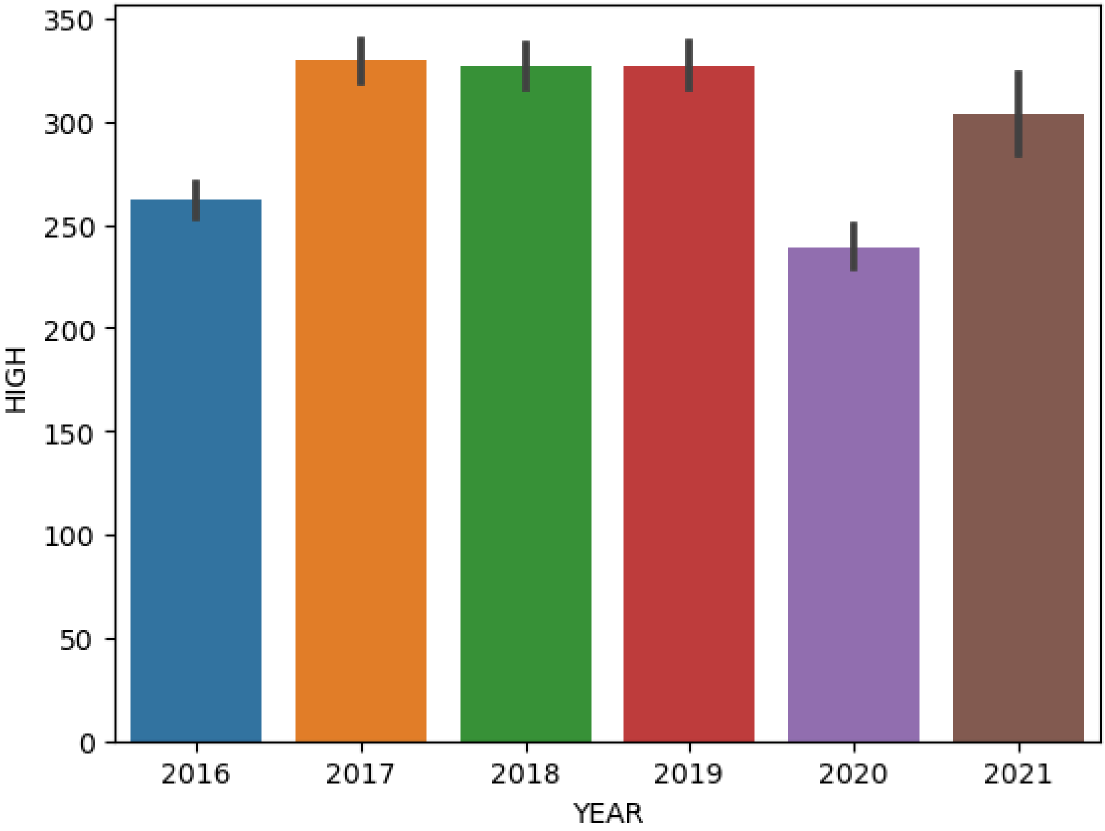
In [27]:
```
sns.lineplot(x='YEAR',y='HIGH',data=df)
```

Out[27]:
```
<Axes: xlabel='YEAR', ylabel='HIGH'>
```



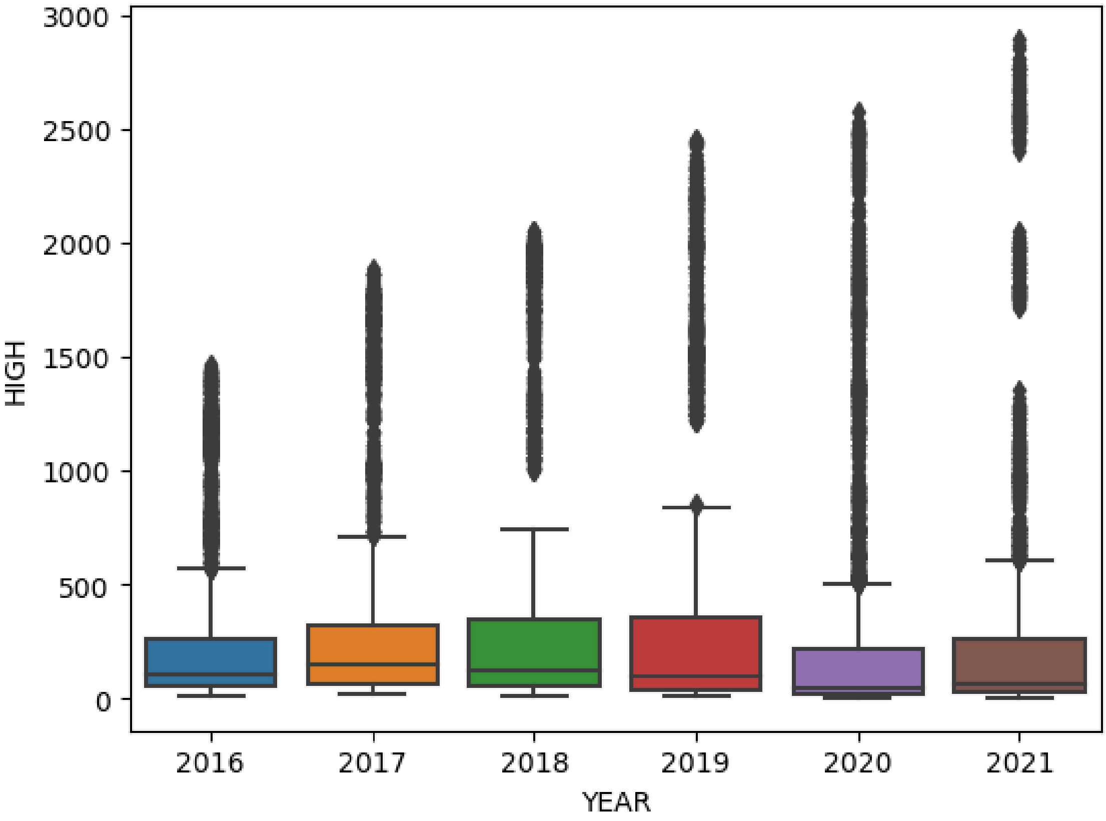
In [28]:
```
sns.barplot(x='YEAR',y='HIGH',data=df)
```

Out[28]:
```
<Axes: xlabel='YEAR', ylabel='HIGH'>
```



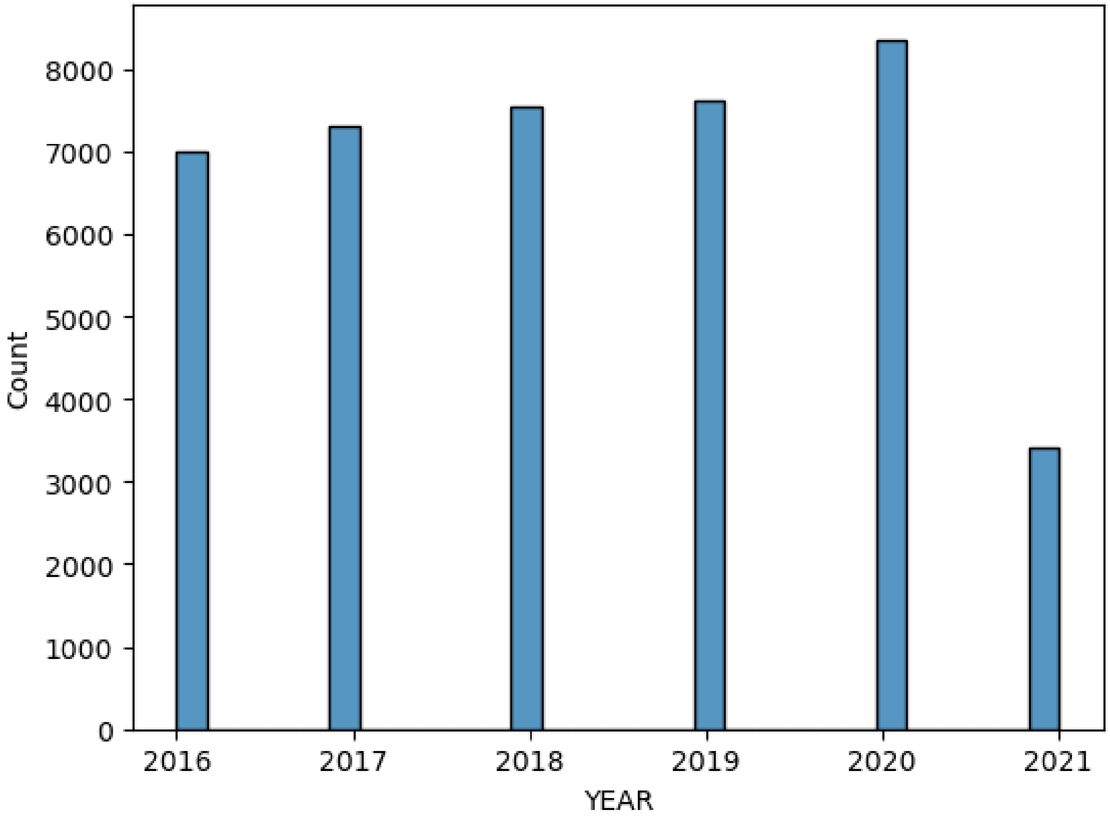
In [29]:
```
sns.boxplot(x='YEAR',y='HIGH',data=df)
```

Out[29]:
```
<Axes: xlabel='YEAR', ylabel='HIGH'>
```



```
In [30]: sns.histplot(df['YEAR'])
```

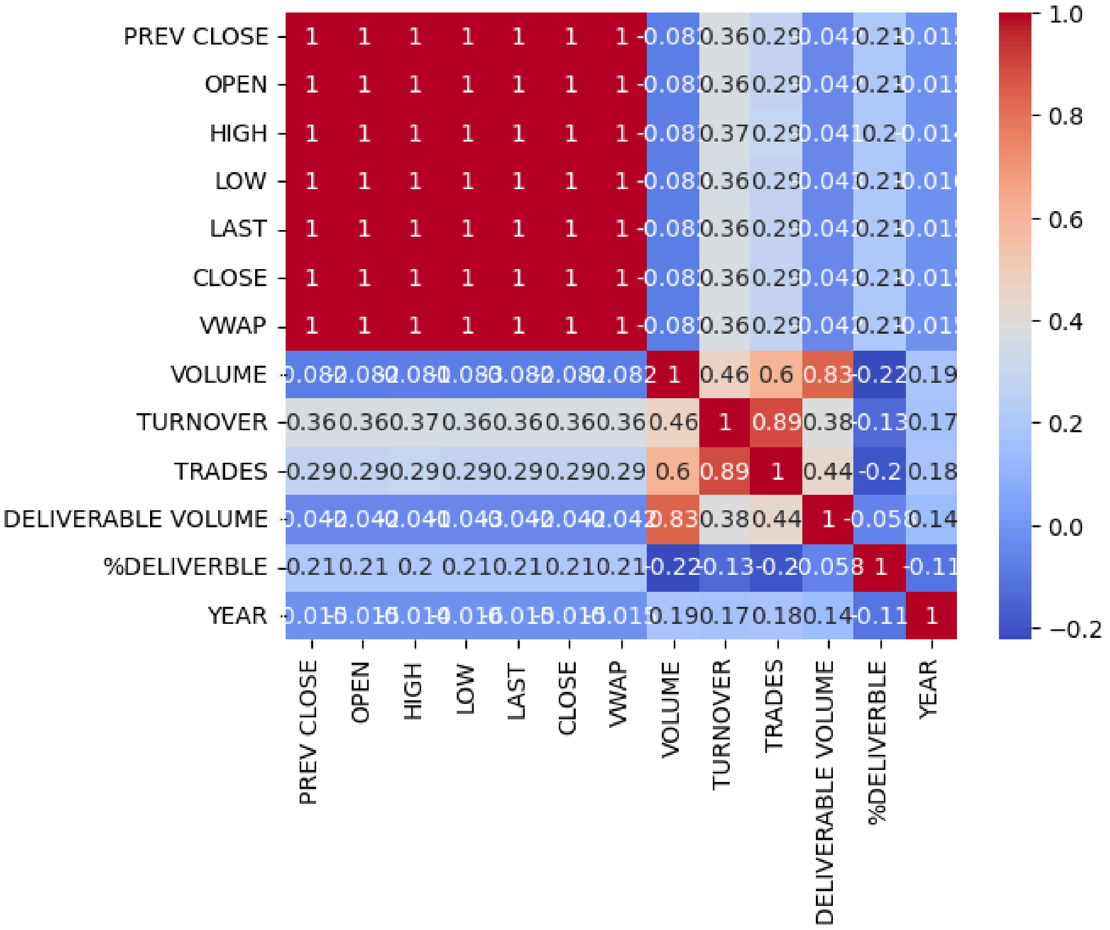
```
Out[30]: <Axes: xlabel='YEAR', ylabel='Count'>
```



```
In [31]: # Checking correlation
         sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
```

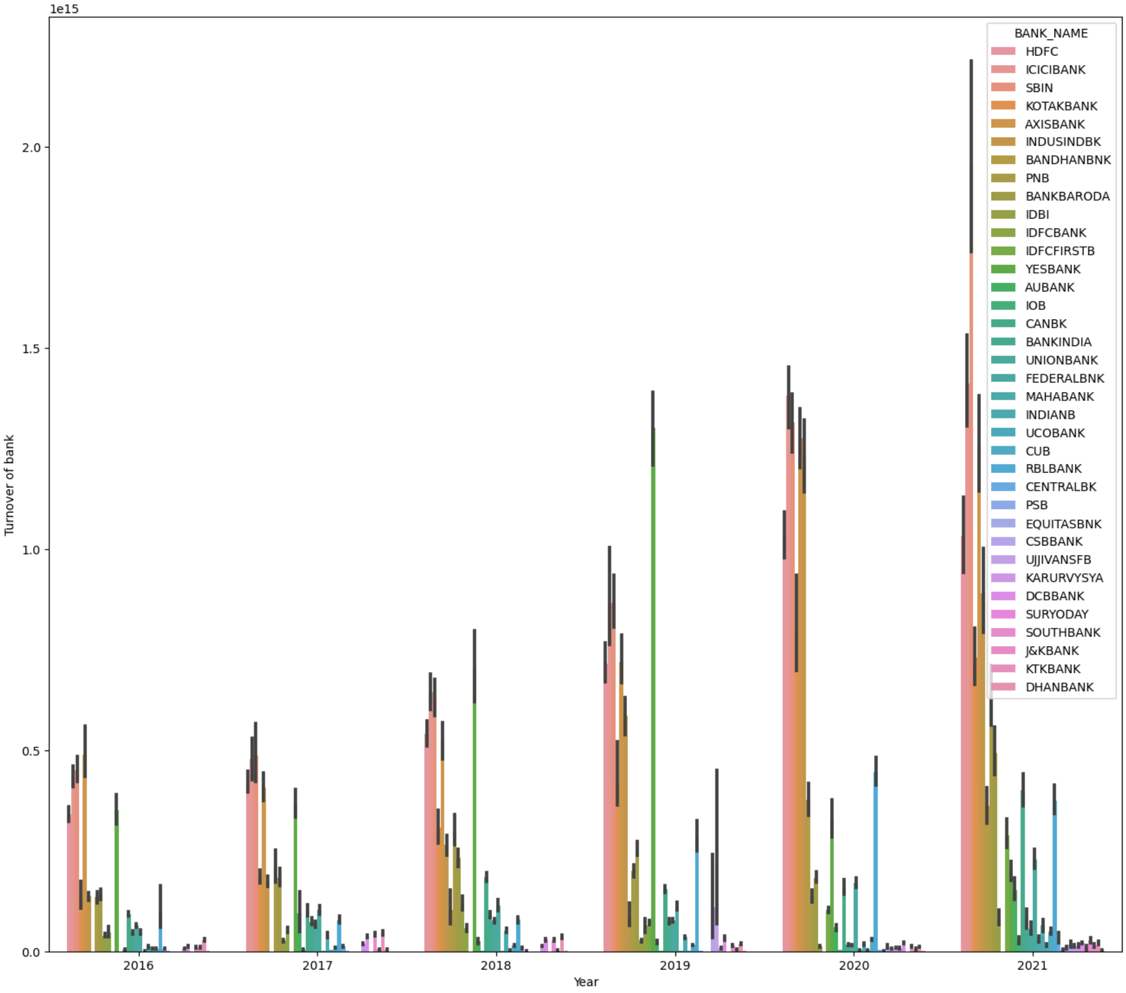
```
C:\Users\saiku\AppData\Local\Temp\ipykernel_9428\808907183.py:2: FutureWa
rning: The default value of numeric_only in DataFrame.corr is deprecated.
In a future version, it will default to False. Select only valid columns
or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
```

```
Out[31]: <Axes: >
```



In [32]:

```
#Checking for banks which produced Highest Turnover
plt.figure(figsize=(16,14))
sns.barplot(x='YEAR',y='TURNOVER',data=df,hue='BANK_NAME')
plt.xlabel('Year')
plt.ylabel('Turnover of bank')
plt.show()
```



In [33]:

```
# Now we are going to Extratct the Banks with Higest Turnover
Banks_turnover = df.groupby(['BANK_NAME']).agg({'TURNOVER':'sum'})
```

In [34]:

```
Banks_turnover
```

Out[34]:

```
                 TURNOVER
 BANK_NAME
    AUBANK   5.295731e+16
  AXISBANK   9.729472e+17
BANDHANBNK   1.706873e+17
BANKBARODA   2.941803e+17
 BANKINDIA   8.274627e+16
     CANBK   2.107119e+17
 CENTRALBK   1.110650e+16
   CSBBANK   5.339832e+15
       CUB   2.335741e+16
   DCBBANK   3.442576e+16
  DHANBANK   2.446045e+15
EQUITASBNK   1.322798e+15
FEDERALBNK   1.578363e+17
      HDFC   8.601080e+17
 ICICIBANK   1.082915e+18
      IDBI   6.305864e+16
  IDFCBANK   3.906347e+16
IDFCFIRSTB   7.106673e+16
   INDIANB   4.473944e+16
INDUSINDBK   6.842308e+17
       IOB   6.265144e+15
   J&KBANK   8.380664e+15
KARURVYSYA   1.448634e+16
 KOTAKBANK   5.304417e+17
   KTKBANK   3.663533e+16
  MAHABANK   4.741024e+15
       PNB   3.045492e+17
       PSB   1.099163e+15
   RBLBANK   2.658997e+17
      SBIN   1.123804e+18
 SOUTHBANK   2.899253e+16
  SURYODAY   3.075047e+14
   UCOBANK   5.877181e+15
 UJJIVANSFB   6.304566e+15
 UNIONBANK   7.968693e+16
   YESBANK   7.722318e+17
```

In [35]:

```
# Now we are going to extract the Bank which has the Higest turnover among

Bank_with_higest_turnover = Banks_turnover['TURNOVER'].idxmax()
print(f'The Bank which has Higest Turnover is : {Bank_with_higest_turnover
```

```
The Bank which has Higest Turnover is : SBIN
```

In [36]:

```
# Now we are going to extract the Bank which has the Lowest turnover among
Bank_with_lowest_turnover = Banks_turnover['TURNOVER'].idxmin()
print(f'The Bank which has Lowest Turnover is : {Bank_with_lowest_turnover
```

```
The Bank which has Lowest Turnover is : SURYODAY
```

In [37]:

```
Year_turnover = df.groupby(['YEAR']).agg({'TURNOVER':'sum'})
```

In [38]:

```
Year_turnover
```

Out[38]:

	TURNOVER
YEAR	
2016	7.614890e+17
2017	9.118193e+17
2018	1.266562e+18
2019	1.714089e+18
2020	2.313062e+18
2021	1.087930e+18

In [39]:

```
# Now we are going to extract the Year which has the Higest turnover among

year_with_higest_turnover = Year_turnover['TURNOVER'].idxmax()
print(f'The Bank which has Higest Turnover is : {year_with_higest_turnover
```

```
The Bank which has Higest Turnover is : 2020
```

In [40]:

```
# Now we are going to extract the Year which has the Lowest turnover among
year_with_lowest_turnover = Year_turnover['TURNOVER'].idxmin()
print(f'The Bank which has Higest Turnover is : {year_with_lowest_turnover
```

```
The Bank which has Higest Turnover is : 2016
```

In [ ]: