Exploratory Data Analysis on Nifty Bank and Its Components

Objective : To perform an exploratory data analysis (EDA) on the Nifty Bank index and its component banks to understand their stock price distributions, correlations, and overall market behavior for making better investment decisions.

Steps involved in Exploratory Data Analysis (EDA) :

Step 1 : Importing Required Python Libraries (on 2nd slide)

Step 3 : Reading Dataset (on 2nd slide)

Step 2 : Analyzing the Data (from 2nd slide to 8th slide)

- Shape of the data
- Data Information
- Description of the Data
- Checking Columns
- Checking Missing Values
- Handling Missing Values

Step 4 : Data Visualization (from 9th slide to 14th slide)

- Converting data to datetime
- Trend Analysis (Line Chart)
- Understanding distribution and identify outliers (Box plots)
- Comparative analysis (Bar Chart)
- Correlation matrix (Heat map)

Analyzing the Data :
For better data understanding

[79]:
```
# important python libraries for data analysis and visualization

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

[80]:
```
# pandas for reading data from various file formats (like CSV, Excel), manipulating and cleaning data

df = pd.read_csv('banknifty_data.csv')
```

*[8]:
```
# command prints the first five rows of the dataFrame 'df'

print(df.head())
```

```
         Date   nifty bank   axis_bank  bandhan_bank     au_bank    sbi_bank  \
0  02-01-2019  27297.00000  623.000000        553.00  307.500000  299.100006
1  03-01-2019  27181.59961  621.400024        551.00  310.100006  295.000000
2  04-01-2019  26999.69922  612.000000        534.25  310.500000  292.100006
3  07-01-2019  27378.65039  626.000000        540.00  312.500000  301.049988
4  08-01-2019  27301.90039  636.000000        482.00  306.049988  295.799988

     bob_bank  idfc_bank  federal_bank   kotak_bank    hdfc_bank  icici_bank  \
0  123.500000  43.200001     93.199997  1247.900024  1071.400024  361.500000
1  121.400002  42.650002     95.300003  1240.000000  1062.099976  365.000000
2  120.400002  43.099998     93.250000  1238.000000  1057.625000  361.850006
3  122.500000  46.000000     95.250000  1250.949951  1063.849976  367.500000
4  122.900002  46.349998     94.849998  1243.000000  1061.000000  367.799988

    pnb_bank  indusind_bank
0  79.550003    1587.599976
1  78.099998    1573.000000
2  78.000000    1560.800049
3  82.449997    1561.000000
4  80.550003    1563.150024
```

```
# The df.round(2) function rounds all numerical values in the dataFrame df to two decimal places, and df_rounded stores the resulting dataFrame.
df_rounded = df.round(2)
```

```
# command prints the first ten rows, providing a preview of data
print(df_rounded.head(10))
```

```
         Date  nifty bank  axis_bank  bandhan_bank  au_bank  sbi_bank  \
0  02-01-2019     27297.00     623.00        553.00   307.50    299.10
1  03-01-2019     27181.60     621.40        551.00   310.10    295.00
2  04-01-2019     26999.70     612.00        534.25   310.50    292.10
3  07-01-2019     27378.65     626.00        540.00   312.50    301.05
4  08-01-2019     27301.90     636.00        482.00   306.05    295.80
5  09-01-2019     27651.25     652.35        480.00   312.00    306.00
6  10-01-2019     27713.55     668.90        452.00   316.40    305.10
7  11-01-2019     27602.80     665.00        468.80   327.42    305.40
8  14-01-2019     27389.20     663.00        453.60   326.90    301.00
9  15-01-2019     27317.55     661.30        444.15   327.70    302.00

   bob_bank  idfc_bank  federal_bank  kotak_bank  hdfc_bank  icici_bank  \
0    123.50      43.20         93.20     1247.90    1071.40      361.50
1    121.40      42.65         95.30     1240.00    1062.10      365.00
2    120.40      43.10         93.25     1238.00    1057.62      361.85
3    122.50      46.00         95.25     1250.95    1063.85      367.50
4    122.90      46.35         94.85     1243.00    1061.00      367.80
5    124.10      46.15         95.55     1237.35    1059.00      381.65
6    122.50      46.00         97.20     1239.00    1058.50      380.95
7    123.80      46.70         96.35     1225.00    1056.90      380.60
8    120.95      47.00         94.40     1217.00    1054.95      378.70
9    121.30      47.15         90.60     1217.00    1052.53      373.55

   pnb_bank  indusind_bank
0     79.55        1587.60
1     78.10        1573.00
2     78.00        1560.80
3     82.45        1561.00
4     80.55        1563.15
5     82.00        1580.00
6     80.30        1607.85
7     81.25        1570.00
8     80.35        1509.00
9     82.85        1491.00
```

```
# # command prints the last five rows of the dataFrame 'df'

print(df_rounded.tail())
```

```
            Date  nifty bank  axis_bank  bandhan_bank  au_bank  sbi_bank  \
1337  14-06-2024    49993.95    1180.95        194.40   672.65    845.00
1338  18-06-2024    50194.35    1194.00        195.00   663.45    841.55
1339  19-06-2024    50607.90    1193.00        198.30   663.00    846.80
1340  20-06-2024    51712.90    1230.10        199.95   659.90    853.00
1341  21-06-2024    51927.30    1246.00        209.09   669.95    844.90

      bob_bank  idfc_bank  federal_bank  kotak_bank  hdfc_bank  icici_bank  \
1337    283.00      77.51        173.26      1723.0    1584.00     1114.00
1338    286.25      78.28        174.90      1718.6    1596.90     1113.95
1339    287.00      82.00        175.09      1729.0    1613.40     1127.95
1340    284.00      82.92        175.95      1765.0    1669.80     1154.05
1341    285.20      84.30        179.73      1770.0    1672.85     1163.55

      pnb_bank  indusind_bank
1337    126.57         1510.5
1338    129.00         1514.0
1339    129.00         1516.2
1340    128.30         1536.0
1341    127.46         1523.0
```

```
# the first element is the number of rows and the second element is the number of columns.

df.shape
```

```
(1342, 14)
```

Summary Statistics :

```
# command generates summary statistics for the numerical columns of the dataFrame 'df', rounds the results to two decimal places.

print(df.describe().round(2))
```

```
       nifty bank  axis_bank  bandhan_bank  au_bank  sbi_bank  bob_bank  \
count     1342.00    1342.00       1342.00  1341.00   1342.00   1342.00
mean     35309.56     768.13        324.27   528.24    438.02    124.32
std       7889.64     185.38        110.53   155.31    164.38     62.34
min      16759.95     293.50        160.80   190.00    151.95     36.15
25%      30009.75     677.00        240.00   365.48    303.96     80.90
50%      35458.32     747.80        297.00   580.40    444.88    104.33
75%      42230.76     875.04        361.82   639.85    565.56    170.30
max      51927.30    1246.00        625.25   813.40    897.00    292.00

       idfc_bank  federal_bank  kotak_bank  hdfc_bank  icici_bank  pnb_bank  \
count    1342.00       1342.00     1342.00    1342.00     1342.00   1342.00
mean       52.32        101.12     1690.41    1386.04      682.44     55.88
std        18.49         32.00      222.18     212.46      238.89     26.62
min        18.50         37.00     1095.05     770.45      284.00     26.60
25%        41.00         83.12     1560.23    1221.98      434.28     36.15
50%        48.25         93.97     1750.05    1443.75      702.00     45.30
75%        59.49        129.96     1844.56    1554.57      901.54     67.28
max        99.75        179.73     2200.00    1723.45     1170.00    141.10

       indusind_bank
count        1342.00
mean         1137.58
std           331.71
min           292.00
25%           930.29
50%          1145.38
75%          1433.28
max          1816.00
```

The output table includes the following statistics for each column:

- **count**: Number of non-missing values.
- **mean**: Average value.
- **std**: Standard deviation, which measures the amount of variation.
- **min**: Minimum value.
- **25%**: 25th percentile (first quartile).
- **50%**: 50th percentile (median or second quartile).
- **75%**: 75th percentile (third quartile).
- **max**: Maximum value.

Each row of the table corresponds to one of these statistics, providing a quick overview of the central tendency, dispersion, and overall range of the data in each column.

```
print(df.dtypes) #to check datatype
```

```
Date                 object
nifty bank          float64
axis_bank           float64
bandhan_bank        float64
au_bank             float64
sbi_bank            float64
bob_bank            float64
idfc_bank           float64
federal_bank        float64
kotak_bank          float64
hdfc_bank           float64
icici_bank          float64
pnb_bank            float64
indusind_bank       float64
dtype: object
```

```
print(df.isnull().sum()) #command prints the missing value in dataframe.
```

```
Date                 0
nifty bank           0
axis_bank            0
bandhan_bank         0
au_bank              1
sbi_bank             0
bob_bank             0
idfc_bank            0
federal_bank         0
kotak_bank           0
hdfc_bank            0
icici_bank           0
pnb_bank             0
indusind_bank        0
dtype: int64
```

Handling Missing values :

```
# command fills missing values in the dataFrame

df_filled = df.fillna(method='ffill').fillna(method='bfill')
```

```
print(df_filled.isnull().sum())  # no missing value showing
```

```
Date            0
nifty bank      0
axis_bank       0
bandhan_bank    0
au_bank         0
sbi_bank        0
bob_bank        0
idfc_bank       0
federal_bank    0
kotak_bank      0
hdfc_bank       0
icici_bank      0
pnb_bank        0
indusind_bank   0
dtype: int64
```

Forward fill and Backward Fill : Forward fill and Backward fill (often abbreviated as 'ffill' & 'bfill') is practical method for handling missing values in time series data.

Note: To deal with the missing values in my data set, I had to combine both.

Forward fill alone: If the first value in the data is missing, forward fill can't fill it because there's no previous value to use.

Backward fill alone: If the last value in the data is missing, backward fill can't fill it because there's no next value to use.

Combining both: By using forward fill first and then backward fill, ensuring that all missing values are filled. Forward fill handles gaps in the middle & backward fill handles any gaps at the start.

```
df.columns # command prints the columns in dataframe
```

```
Index(['Date', 'nifty bank', 'axis_bank', 'bandhan_bank', 'au_bank',
       'sbi_bank', 'bob_bank', 'idfc_bank', 'federal_bank', 'kotak_bank',
       'hdfc_bank', 'icici_bank', 'pnb_bank', 'indusind_bank'],
      dtype='object')
```

```
# Convert 'Date' column to datetime with the original format for time series data and analysis

df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')
```

```
# Change the format to display only the year

df['Year'] = df['Date'].dt.strftime('%Y')
```

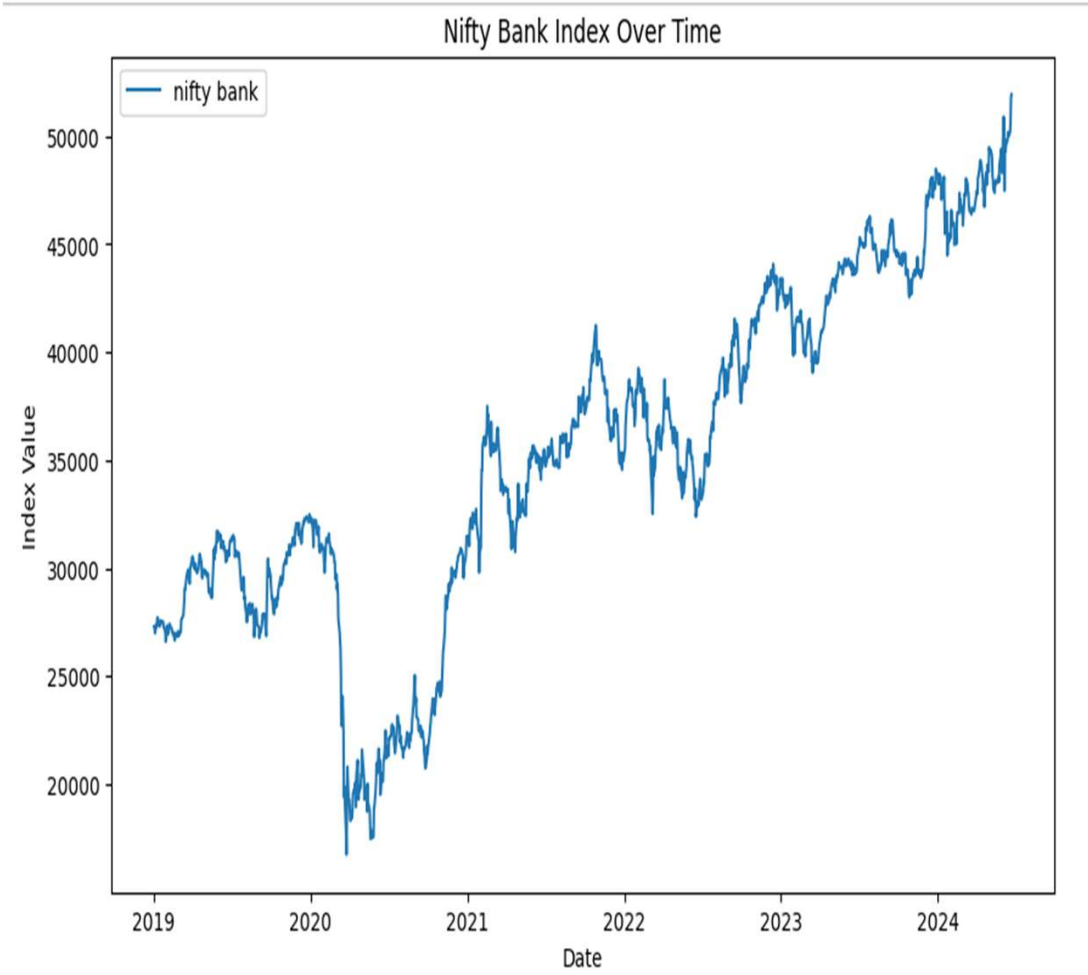
```
# command sets the 'Date' column as the index of the dataFrame df and modifies the dataFrame in place without creating a new dataframe.

df.set_index('Date', inplace=True)
```

Data Visualization:

```
# Plot the 'nifty bank' index over time to see its trend.

plt.figure(figsize=(10, 6))
plt.plot(df.index, df['nifty bank'], label='nifty bank')
plt.xlabel('Date')
plt.ylabel('Index Value')
plt.title('Nifty Bank Index Over Time')
plt.legend()
plt.show()
```

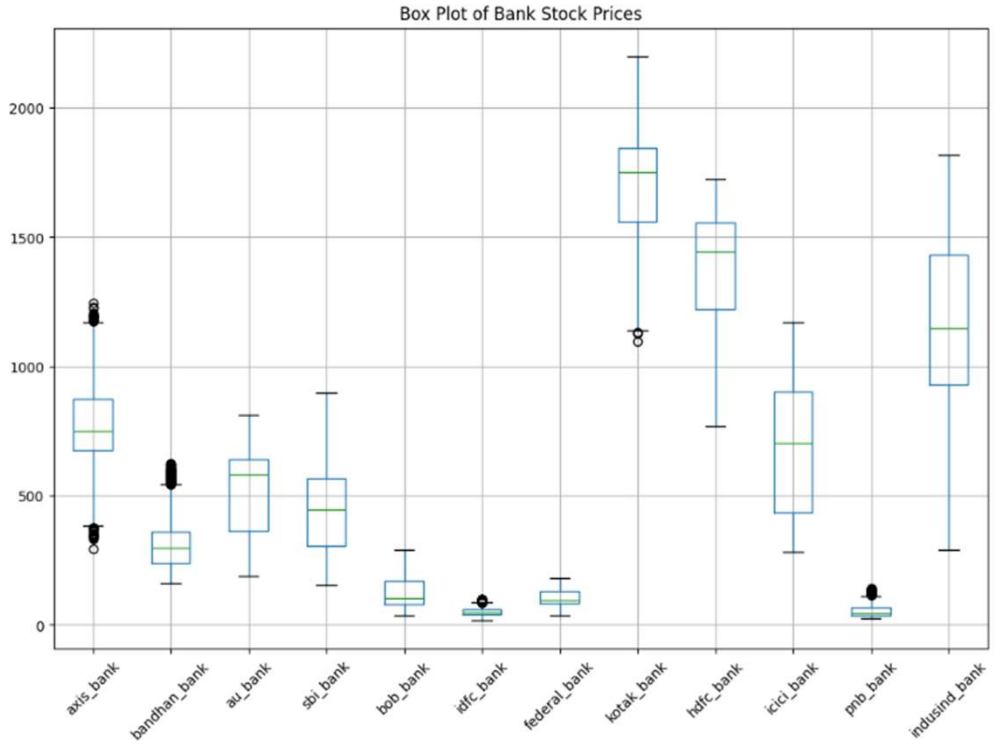


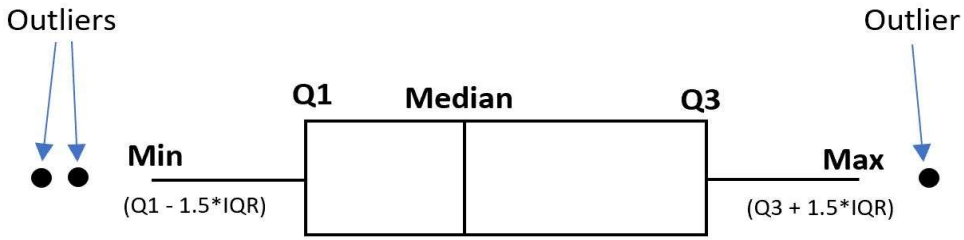
Insights from line chart:

- This plot shows that the Bank Nifty Index has been going upward overall.
- Even though there were some drops, especially in early 2020 due to the global financial impact of the COVID-19 pandemic. It shows that the banking sector has recovered well and is performing strongly now.

```
# Box plots for each bank to understand the distribution and identify outliers.

plt.figure(figsize=(12, 8))
df.boxplot(column=['axis_bank', 'bandhan_bank', 'au_bank', 'sbi_bank', 'bob_bank', 'idfc_bank', 'federal_bank', 'kotak_bank', 'hdfc_bank', 'icici_bank',
plt.title('Box Plot of Bank Stock Prices')
plt.xticks(rotation=45)
plt.show()
```





Minimum: The smallest data point excluding any outliers.
First Quartile (Q1): The median of the lower half of the dataset (25th percentile), **[25/100*(n+1)]**
Median (Q2): The middle value of the dataset (50th percentile).
Third Quartile (Q3): The median of the upper half of the dataset (75th percentile), **[75/100*(n+1)]**
Maximum: The largest data point excluding any outliers.
Interquartile Range (IQR): The range between the first quartile (Q1) and the third quartile (Q3), **(IQR=Q3-Q1)**
Outliers: Outliers in a box plot are data points that fall significantly outside the range of most of the data.

Insights from boxplot:

The box plot reveals significant differences in stock price distributions among the banks:

- Kotak Bank, HDFC Bank and ICICI Bank stand out with high median prices and wide ranges, indicating high-value stocks with significant variability.
- IDFC Bank, Federal Bank, IDFC Bank, PNB Bank have lower median prices and narrower ranges, suggesting more stable but lower-valued stocks.
- Several outliers are visible above the upper whisker (Axis Bank, Bandhan Bank, and Kotak Bank), indicating occasional extreme stock prices, which could be due to market events or investor behavior.
- Some banks like IDFC Bank and Federal Bank have fewer or no outliers, indicating more consistent stock prices within the expected range.

Conclusion:

High Risk, High Reward Stocks: Kotak Bank, HDFC Bank, ICICI Bank.

Stable and Low Risk Stocks: Bandhan Bank, Federal Bank, IDFC Bank, PNB Bank.

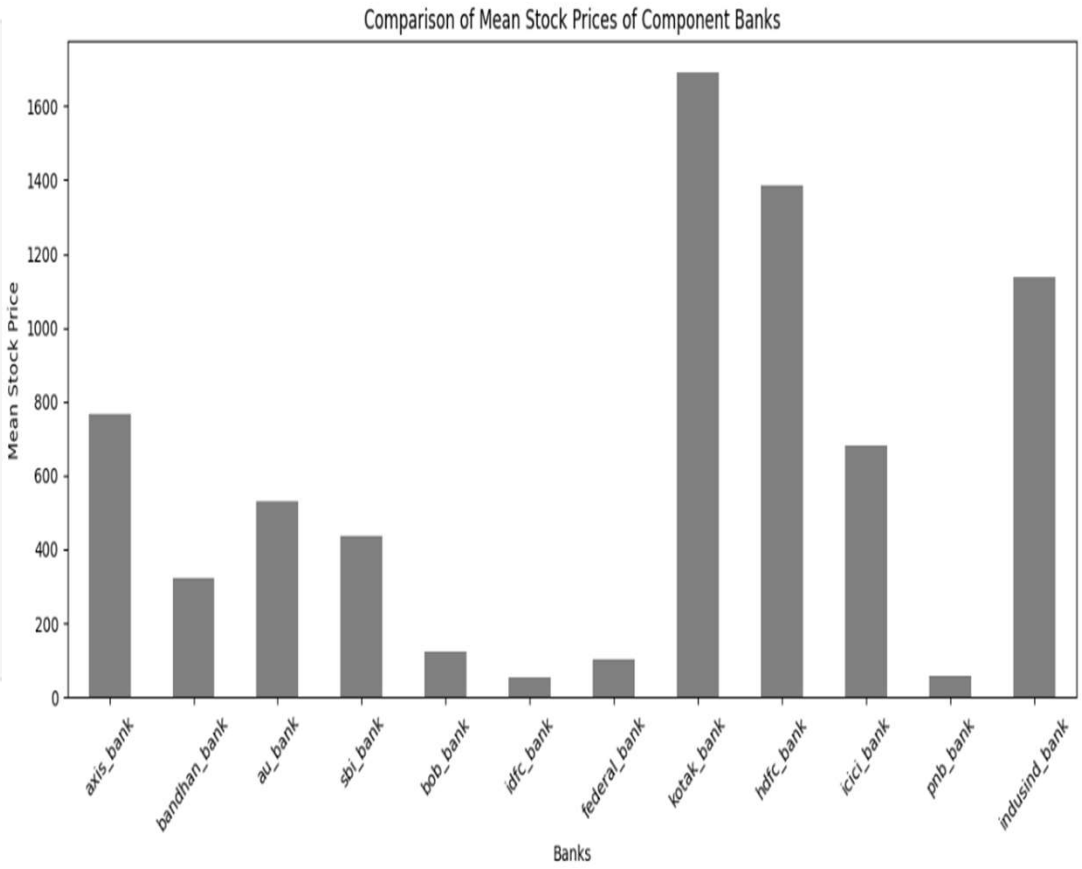
Moderate Risk and Reward Stocks: AU Bank, IndusInd Bank.

Potential for Occasional High Returns Stocks: Axis Bank.

```
# creates a bar plot to visualize the comparison of these mean stock prices BETWEEN different banks.

mean_prices = df.drop(columns=['nifty bank']).mean()


plt.figure(figsize=(12, 6))
mean_prices.plot(kind='bar', color='grey')
plt.xlabel('Banks')
plt.ylabel('Mean Stock Price')
plt.title('Comparison of Mean Stock Prices of Component Banks')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



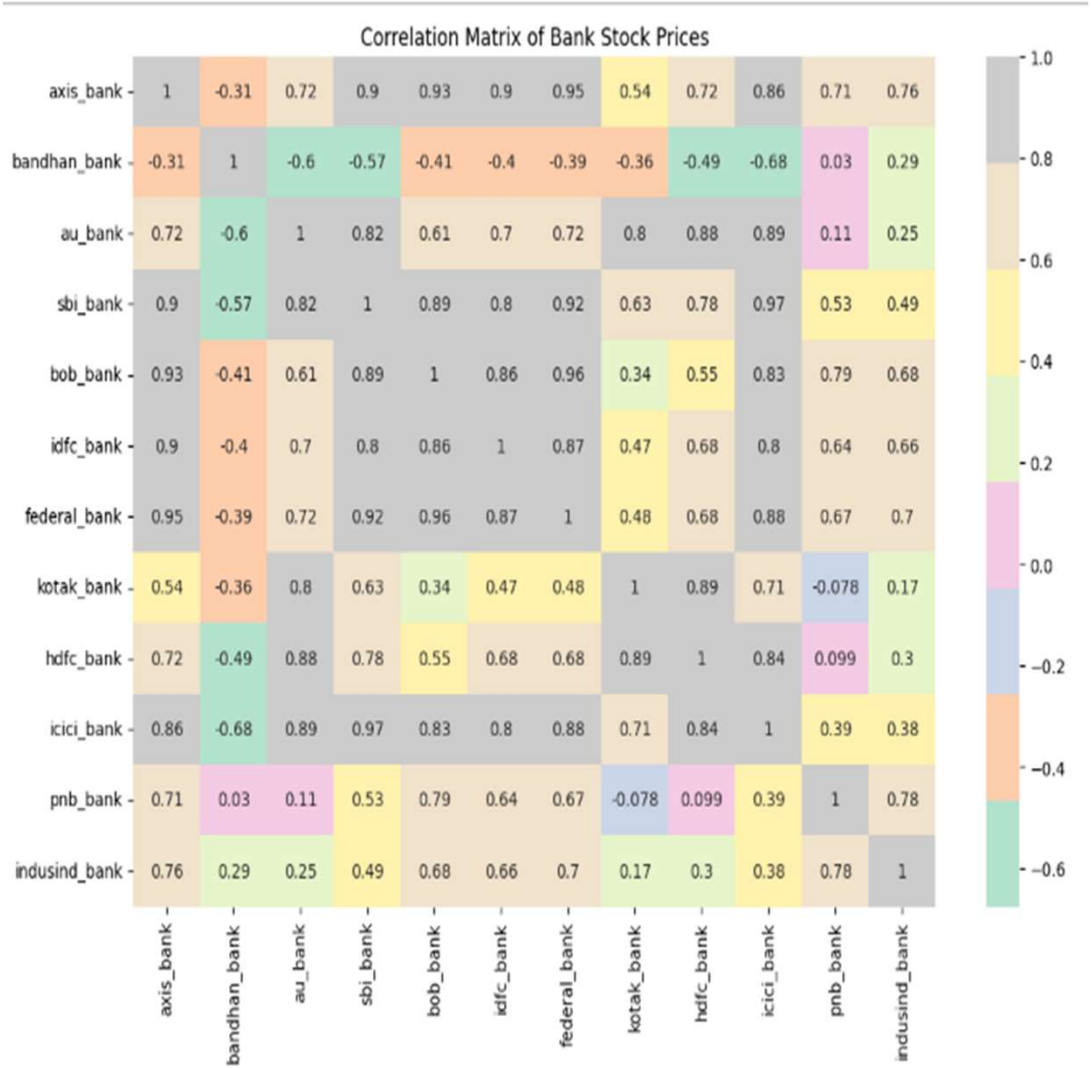
Insights from bar chart:

- Kotak Bank, HDFC Bank and Indusind Bank stand out with the highest mean stock prices, indicating strong market positions and high investor confidence.
- IDFC Bank, BOB Bank, and Federal Bank have the lowest mean stock prices, suggesting lower market valuations.
- The other banks fall in between, with varying degrees of market presence and investor appeal.

Mean Stock Price = Sum of Stock Prices / Number of Observations

```
# Correlation matrix
corr_matrix = df.drop(columns=['nifty bank']).corr()

# Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='Pastel2')
plt.title('Correlation Matrix of Bank Stock Prices')
plt.show()
```



Interpretation of Correlation matrix:

•Values close to +1 indicates strong positive correlation, -1 indicates a strong negative correlation and 0 indicates suggests no linear correlation.

•Darker colors signify strong correlation, while light colors represents weaker correlations.

•Positive correlation variable move in same directions. As one increases, the other also increases.

•Negative correlation variable move in opposite directions. An increase in one variable is associated with a decrease in the other.

Insights from heat map:

- The correlation matrix reveals that certain banks, such as Axis Bank, SBI Bank, BOB Bank, Federal Bank, and ICICI Bank, have high positive correlations with each other, indicating that their stock prices tend to move together.
- Bandhan Bank, IDFC Bank, and PNB Bank show lower or negative correlations with other banks, suggesting more independent or inverse movements in their stock prices.
- This analysis can help investors understand the relationships between different bank stocks and make more informed investment decisions.

THANK YOU !