STOCK PRICE ANALYSIS AND PREDICTION

The stock market is a complex and dynamic environment where investors aim to make informed decisions about buying, selling, or holding stocks. Stock price analysis and prediction play a crucial role in helping investors understand market trends, identify potential opportunities, and manage risk effectively.
In this project, I have develop a stock price analysis and prediction system. The project will involve fetching historical stock price data from reliable sources like Yahoo Finance, performing exploratory data analysis (EDA) to gain insights into the data, and applying predictive models to forecast future stock prices. We will focus on the ARIMA (AutoRegressive Integrated Moving Average) model, a popular choice for time series forecasting in stock price prediction.

Content:

1. Importing data from Yahoo Finance.
2. Exploratory Data Analysis
3. Data Visualization.
    - Adjacent Closing Price of Stock
    - Volumne of BRK-B Stock
    - Closing and Opening Price
    - Moving Average of BRK-B Stock price.
    - Candlestick Graph of BRK-B Price.
    - Daily Change in BRK-B Price.
    - Correlation Matrix
4. Predicting Closing Price of BRK-B Price.

In [1]:
```
pip install yfinance
```

```
Requirement already satisfied: yfinance in c:\users\hp\appdata\local\programs
\python\python311\lib\site-packages (0.2.18)
Requirement already satisfied: pandas>=1.3.0 in c:\users\hp\appdata\local\pro
grams\python\python311\lib\site-packages (from yfinance) (1.5.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\hp\appdata\local\pro
grams\python\python311\lib\site-packages (from yfinance) (1.23.5)
Requirement already satisfied: requests>=2.26 in c:\users\hp\appdata\local\pr
ograms\python\python311\lib\site-packages (from yfinance) (2.28.2)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\hp\appdata\loc
al\programs\python\python311\lib\site-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in c:\users\hp\appdata\local\progr
ams\python\python311\lib\site-packages (from yfinance) (4.9.2)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\hp\appdata\local\pr
ograms\python\python311\lib\site-packages (from yfinance) (1.4.4)
Requirement already satisfied: pytz>=2022.5 in c:\users\hp\appdata\local\prog
rams\python\python311\lib\site-packages (from yfinance) (2022.7.1)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\hp\appdata\local
\programs\python\python311\lib\site-packages (from yfinance) (2.3.7)
Requirement already satisfied: cryptography>=3.3.2 in c:\users\hp\appdata\loc
al\programs\python\python311\lib\site-packages (from yfinance) (40.0.2)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\hp\appdata
\local\programs\python\python311\lib\site-packages (from yfinance) (4.11.1)
Requirement already satisfied: html5lib>=1.1 in c:\users\hp\appdata\local\pro
grams\python\python311\lib\site-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\hp\appdata\local\pro
grams\python\python311\lib\site-packages (from beautifulsoup4>=4.11.1->yfinan
ce) (2.3.2.post1)
Requirement already satisfied: cffi>=1.12 in c:\users\hp\appdata\local\progra
ms\python\python311\lib\site-packages (from cryptography>=3.3.2->yfinance)
(1.15.1)
Requirement already satisfied: six>=1.9 in c:\users\hp\appdata\local\programs
\python\python311\lib\site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in c:\users\hp\appdata\local\prog
rams\python\python311\lib\site-packages (from html5lib>=1.1->yfinance) (0.5.
1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\hp\appdata
\local\programs\python\python311\lib\site-packages (from pandas>=1.3.0->yfina
nce) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\hp\appdat
a\local\programs\python\python311\lib\site-packages (from requests>=2.26->yfi
nance) (3.0.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\appdata\local\prog
rams\python\python311\lib\site-packages (from requests>=2.26->yfinance) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\hp\appdata\l
ocal\programs\python\python311\lib\site-packages (from requests>=2.26->yfinan
ce) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\appdata\loca
l\programs\python\python311\lib\site-packages (from requests>=2.26->yfinance)
(2022.12.7)
Requirement already satisfied: pycparser in c:\users\hp\appdata\local\program
s\python\python311\lib\site-packages (from cffi>=1.12->cryptography>=3.3.2->y
finance) (2.21)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\hp\appdata\local\programs\p
ython\python311\lib\site-packages (5.14.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\hp\appdata\local\p
rograms\python\python311\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: packaging in c:\users\hp\appdata\local\program
s\python\python311\lib\site-packages (from plotly) (23.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
#Importing basic libraries
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from pmdarima.arima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error, mean_absolute_error
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

Data Collection ¶

In [4]:
```
data=yf.download('BRK-B',start='2022-05-01')
```

```
[**********100%**********]  1 of 1 completed
```

In [5]:
```
data
```

Out[5]:

Date	Open	High	Low	Close	Adj Close	Volume
2022-05-02	324.109985	324.369995	311.739990	318.190002	318.190002	6797200
2022-05-03	319.420013	323.579987	317.670013	318.989990	318.989990	3874700
2022-05-04	319.100006	327.279999	318.000000	326.799988	326.799988	4269600
2022-05-05	325.850006	325.850006	315.160004	318.679993	318.679993	5257600
2022-05-06	317.989990	320.369995	314.190002	318.880005	318.880005	4198800
...	...	...	...	...	...	...
2023-05-22	330.750000	331.489990	328.350006	329.130005	329.130005	2762500
2023-05-23	328.190002	329.269989	322.970001	323.109985	323.109985	4029300
2023-05-24	322.709991	323.000000	319.559998	320.200012	320.200012	3071500
2023-05-25	320.559998	320.559998	317.709991	319.019989	319.019989	4245400
2023-05-26	320.440002	322.630005	319.670013	320.600006	320.600006	3229400

270 rows × 6 columns

Exploratory Data Analysis

In [6]:
```
#checking number of rows and columns of data
data.shape
```

Out[6]:
```
(270, 6)
```

In [7]:
```
#Columns of dataset
data.columns
```

Out[7]:
```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='objec
t')
```

In [8]:
```
#Datatype of the columns
data.dtypes
```

Out[8]:
```
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

In [9]:
```
#Checking null values in data
data.isnull().sum()
```

Out[9]:
```
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [10]:
```
#Data Description
data.describe()
```

Out[10]:

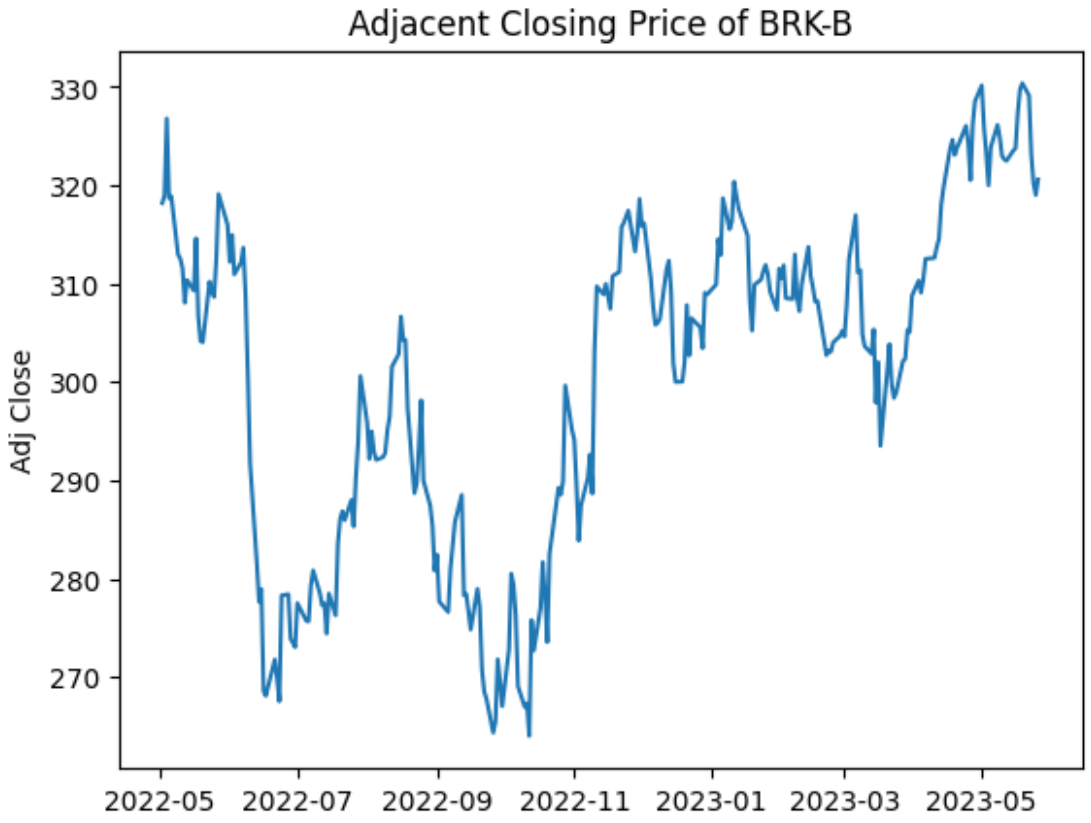
	Open	High	Low	Close	Adj Close	Volume
count	270.000000	270.000000	270.000000	270.000000	270.000000	2.700000e+02
mean	300.859482	303.199148	298.000518	300.671444	300.671444	3.887631e+06
std	17.037531	16.667084	17.075118	16.983084	16.983084	1.403945e+06
min	260.579987	267.519989	259.850006	264.000000	264.000000	1.844900e+06
25%	287.355003	289.259995	284.679993	287.455009	287.455009	3.047375e+06
50%	305.270004	307.459991	302.564987	305.129990	305.129990	3.613900e+06
75%	312.977493	315.402504	309.904999	312.507507	312.507507	4.360075e+06
max	331.000000	333.940002	329.119995	330.390015	330.390015	1.560940e+07

Data Visualization

Adjacent Closing Proce of Stock

In [11]:
```
plt.plot(data['Adj Close'])
plt.ylabel('Adj Close')
plt.title('Adjacent Closing Price of BRK-B')
```

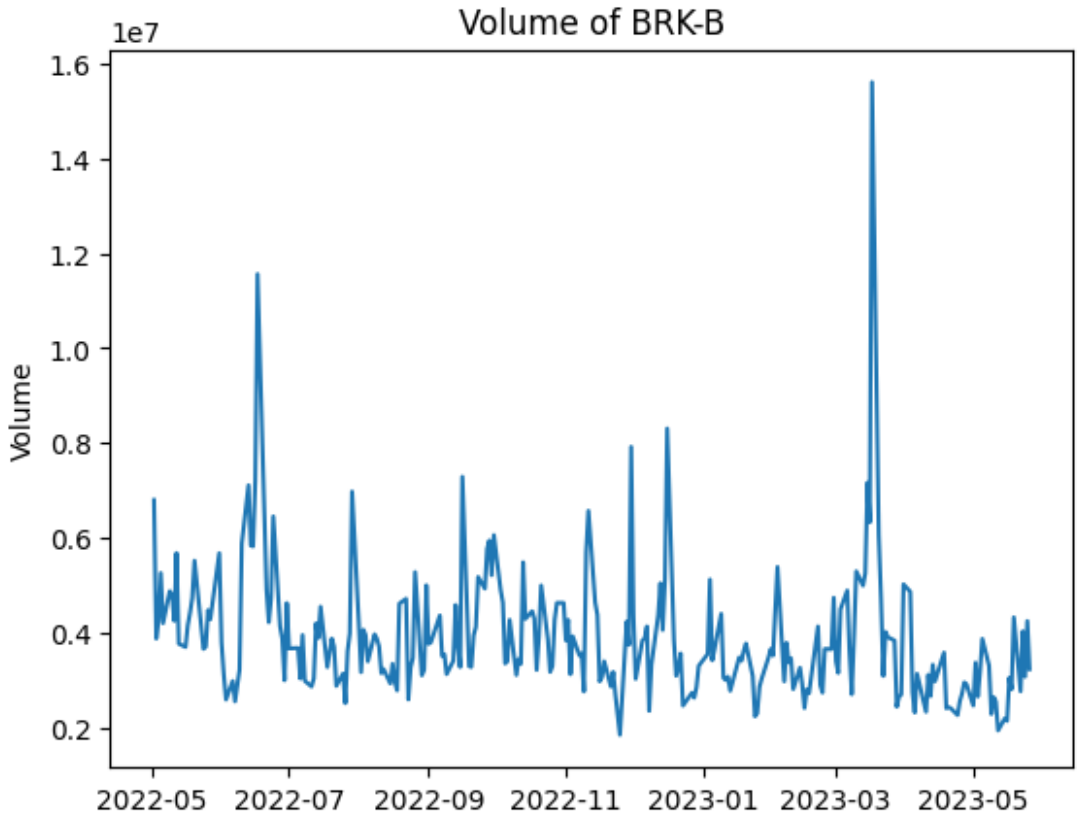
Out[11]: Text(0.5, 1.0, 'Adjacent Closing Price of BRK-B')



Volume of BRK-B Stock

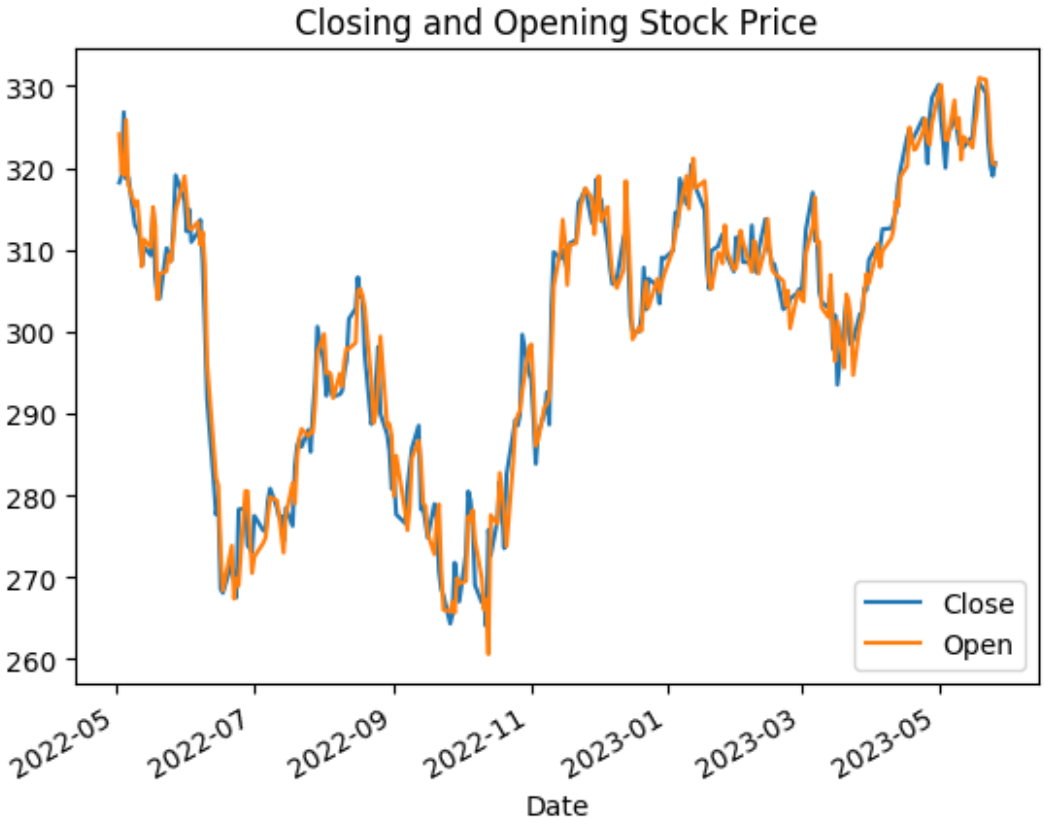
In [12]:
```
1 plt.plot(data['Volume'])
2 plt.ylabel('Volume')
3 plt.title('Volume of BRK-B')
```

Out[12]: Text(0.5, 1.0, 'Volume of BRK-B')



Closing and Opening Stock Price

In [13]:
```
data.plot(y=['Close','Open'])
plt.title('Closing and Opening Stock Price')
plt.legend()
plt.show()
```



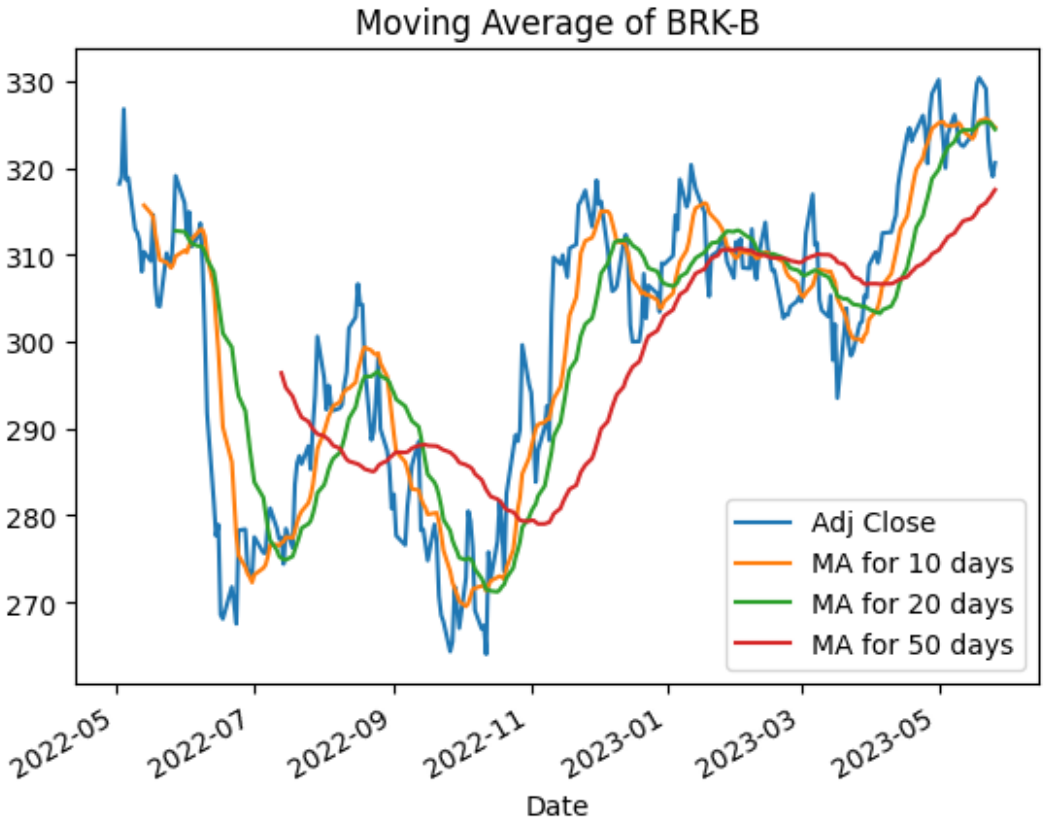
Moving Average of BRK-B Stock Price

In [14]:
```
#Calculating the Moving Average for 10 days, 20 and 50 days
ma_day = [10, 20, 50]

for ma in ma_day:
    column_name = f"MA for {ma} days"
    data[column_name] = data['Adj Close'].rolling(ma).mean()
```

In [15]:
```
1 data.plot(y=['Adj Close','MA for 10 days','MA for 20 days','MA for 50 days
2 plt.title('Moving Average of BRK-B')
```

Out[15]: Text(0.5, 1.0, 'Moving Average of BRK-B')



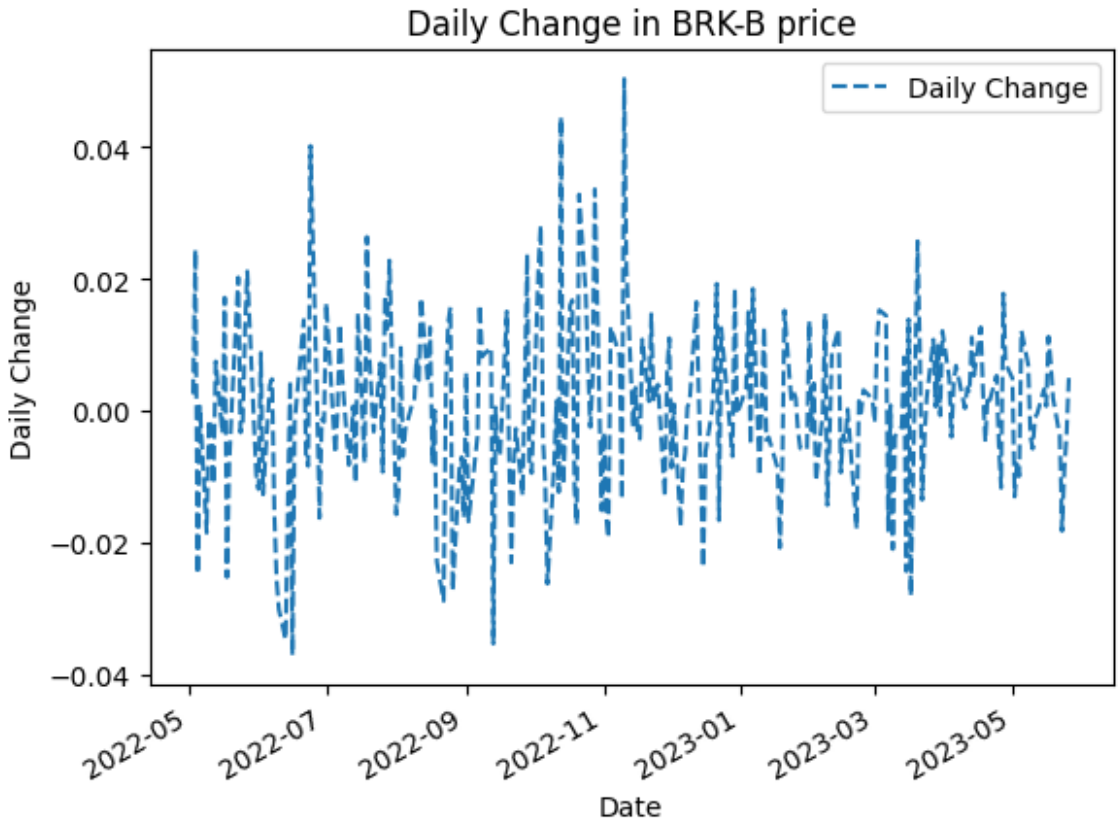
Candlestick Graph of BRK-B Stock

In [16]:
```
fig = go.Figure(data=[go.Candlestick(x=data.index,
                open=data['Open'],
                high=data['High'],
                low=data['Low'],
                close=data['Close'])])

fig.show()
```

Daily Change in BRK-B Price

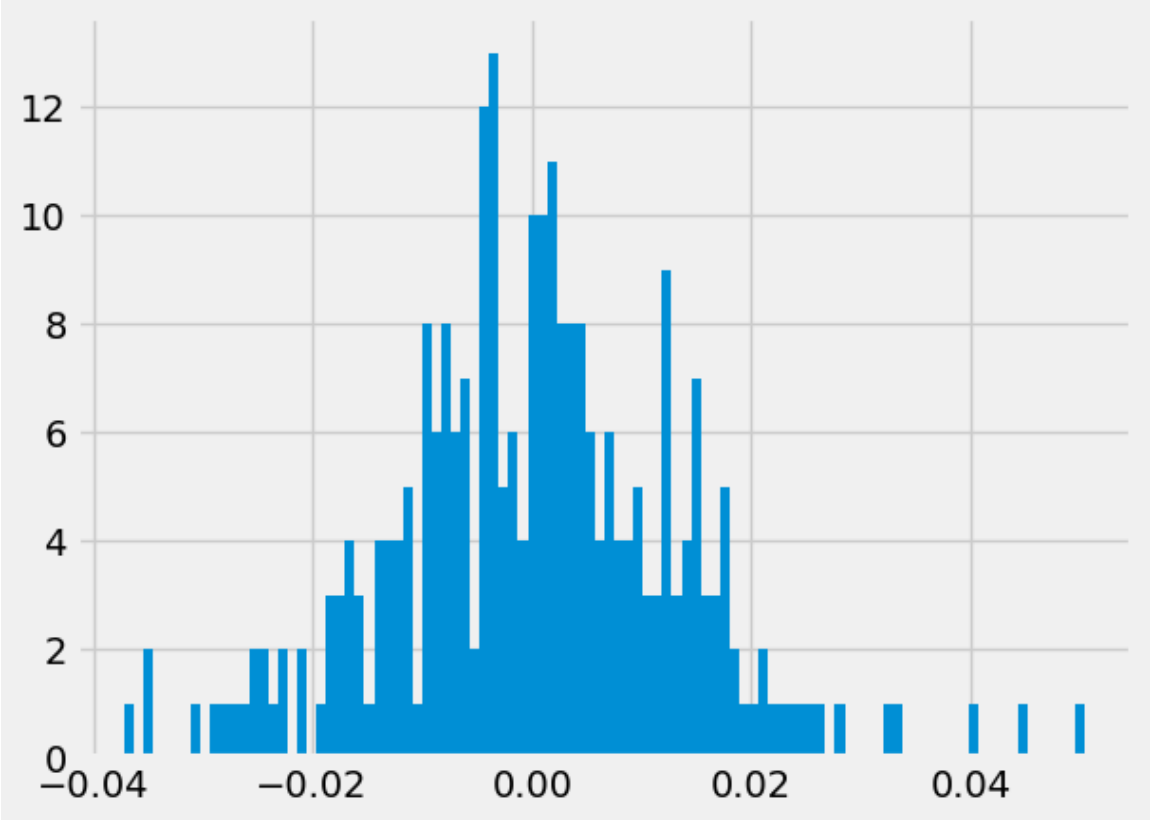
In [17]:
```
data['Daily Change']=data['Adj Close'].pct_change()
data.plot(y='Daily Change',linestyle='--')
plt.ylabel('Daily Change')
plt.title('Daily Change in BRK-B price')
plt.style.use('fivethirtyeight')
```



Histogram of Daily Change in BRK-B Stock Price

In [18]:
```
1 data['Daily Change'].hist(bins=100)
```

Out[18]: <AxesSubplot: >



Correlation Matrix

In [19]:
```
1 data.corr()
```

Out[19]:

	Open	High	Low	Close	Adj Close	Volume	MA for 10 days	MA for 20 days	
Open	1.000000	0.992975	0.991543	0.980928	0.980928	-0.268497	0.935599	0.841925	0
High	0.992975	1.000000	0.992836	0.992382	0.992382	-0.261039	0.933350	0.840243	0
Low	0.991543	0.992836	1.000000	0.991948	0.991948	-0.310639	0.928619	0.834601	0
Close	0.980928	0.992382	0.991948	1.000000	1.000000	-0.287593	0.922485	0.829800	0
Adj Close	0.980928	0.992382	0.991948	1.000000	1.000000	-0.287593	0.922485	0.829800	0
Volume	-0.268497	-0.261039	-0.310639	-0.287593	-0.287593	1.000000	-0.210175	-0.160112	-0
MA for 10 days	0.935599	0.933350	0.928619	0.922485	0.922485	-0.210175	1.000000	0.950565	0
MA for 20 days	0.841925	0.840243	0.834601	0.829800	0.829800	-0.160112	0.950565	1.000000	0
MA for 50 days	0.688598	0.690761	0.694988	0.692413	0.692413	-0.166794	0.767724	0.837302	1
Daily Change	-0.048474	0.029781	0.029626	0.115565	0.115565	-0.072702	-0.086950	-0.107042	-0

Predicting the Closing Price of BRK-B

Data Preprocessing for ARIMA Model

In [20]:
```
#split into training and testing data
close=data['Close']
train_data=close[:int(len(close)*0.90)]
test_data=close[int(len(close)*0.90):]
```

In [21]:
```
#plotting training and testing data
plt.plot(close,label='training data')
plt.plot(test_data,color='red',label='test data')
plt.legend()
plt.show()
```

Choosing the value of p, d and q

In [22]:

```
model_autoARIMA = auto_arima(train_data, start_p=0, start_q=0,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=5, max_q=5, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)
print(model_autoARIMA.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=1366.992, Time=0.08 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=1368.989, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=1368.989, Time=0.03 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=1364.998, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=1370.992, Time=0.04 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 0.230 seconds
                               SARIMAX Results
==============================================================================
=
Dep. Variable:                      y   No. Observations:                  24
3
Model:               SARIMAX(0, 1, 0)   Log Likelihood                -681.49
9
Date:                Fri, 26 May 2023   AIC                           1364.99
8
Time:                        23:16:37   BIC                           1368.48
7
Sample:                             0   HQIC                          1366.40
4
                                - 243
Covariance Type:                  opg
==============================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
sigma2        16.3519      1.312     12.459      0.000      13.780      18.92
4
==============================================================================
======
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
3.58
Prob(Q):                              0.95   Prob(JB):
0.17
Heteroskedasticity (H):               0.57   Skew:
0.10
Prob(H) (two-sided):                  0.01   Kurtosis:
3.56
==============================================================================
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
```

Hence,

p=0

d=1

q=0

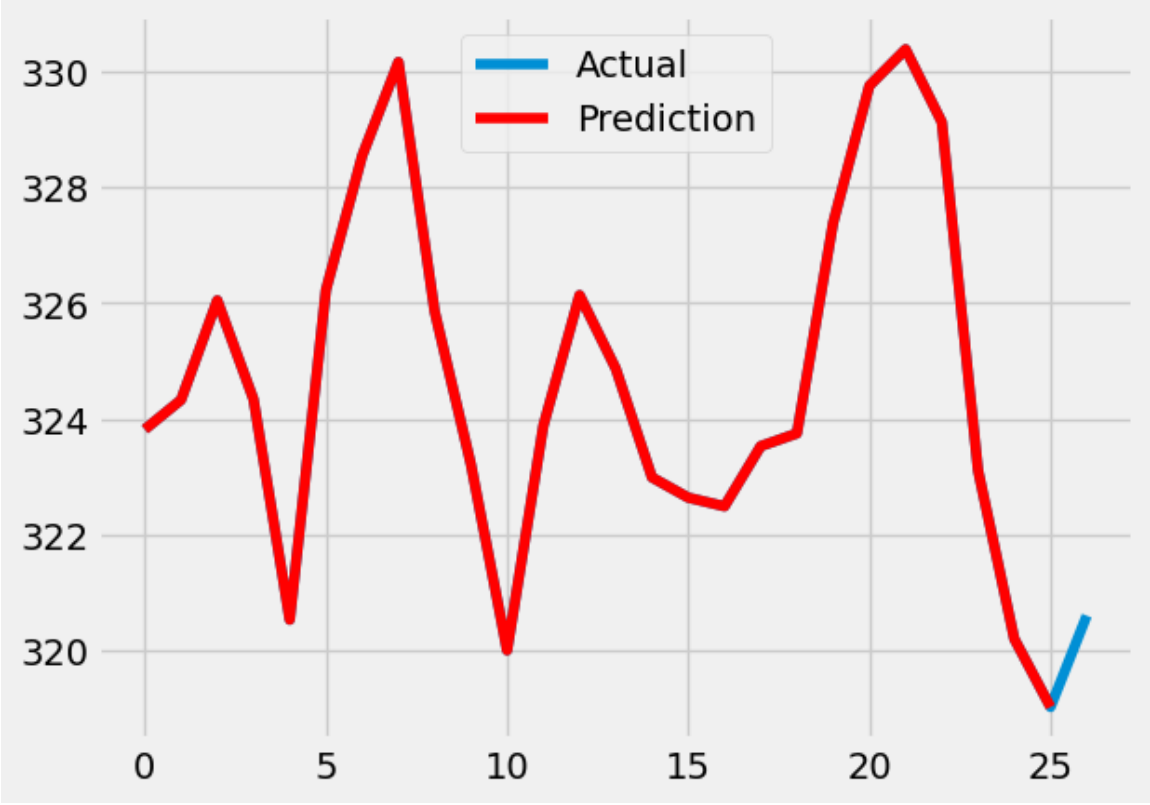
Fitting and Forecasting

In [23]:
```
train_data=list(train_data)
test_data=list(test_data)
predict=[]
number=len(test_data)
for i in range(number):

    model = ARIMA(train_data, order=(0, 1, 0))
    model_fit = model.fit()
    output = model_fit.forecast(alpha=0.05)

    predict.append(output)
    actualtestvalue=test_data[i]
    train_data.append(actualtestvalue)
```

In [24]:
```
plt.plot(test_data,label='Actual')
plt.plot(predict[1:],color='red', label='Prediction')
plt.legend()
plt.show()
```



Checking accuracy of Model

In [25]:
```
mape = np.mean(np.abs(np.array(predict) - np.array(test_data))/np.abs(np.a
```

In [26]:
```
mape
```

Out[26]:
```
0.010759226345524136
```

Mean absolute Percentage Error is close to 0.01 which means that the model is 99% accurate.