In [ ]:
```
import pandas as pd
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
data = pd.read_csv('HDFCBANK.csv', parse_dates=['Date'], date_parser=lambda x: p

# Calculate the difference between the closing price of the first and last tradi
data['Year'] = data['Date'].dt.year
yearly_changes = data.groupby('Year').agg({'Close': lambda x: x.iloc[-1] - x.ilo
```

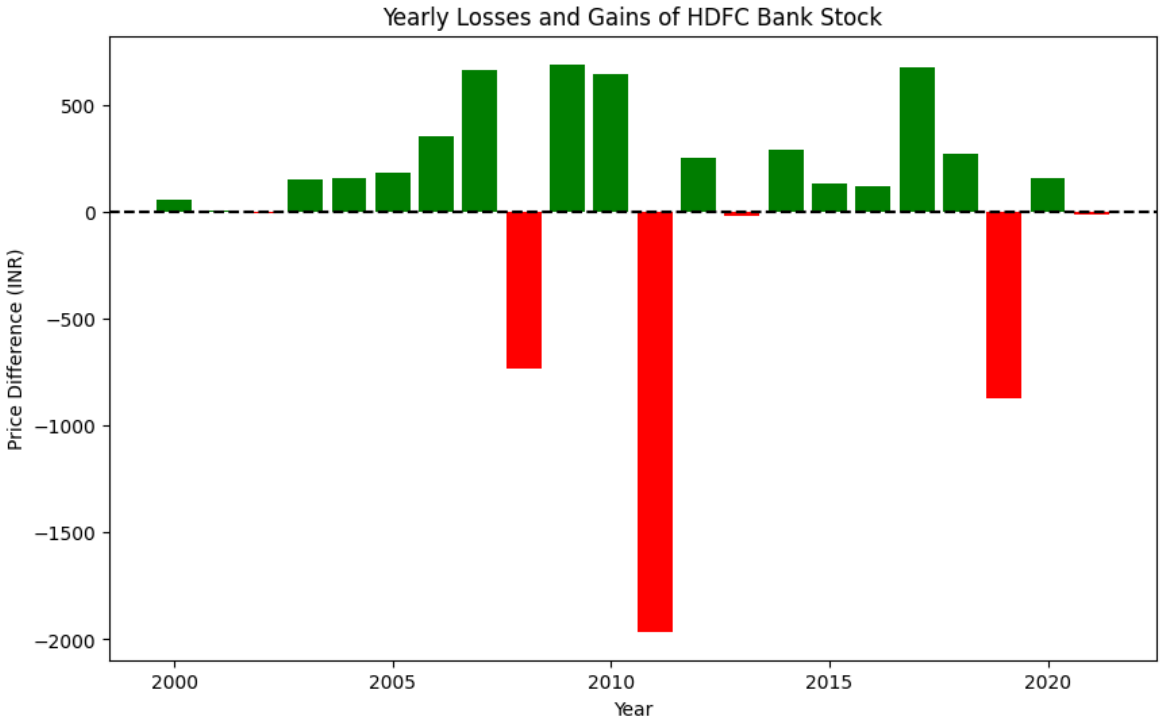
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_19804\661564874.py:1: FutureWarning:
The argument 'date_parser' is deprecated and will be removed in a future version.
Please use 'date_format' instead, or read your data in as 'object' dtype and then
call 'to_datetime'.
  data = pd.read_csv('HDFCBANK.csv', parse_dates=['Date'], date_parser=lambda x:
pd.to_datetime(x, format='%Y-%m-%d'))
```

In [ ]:
```
data.describe()
```

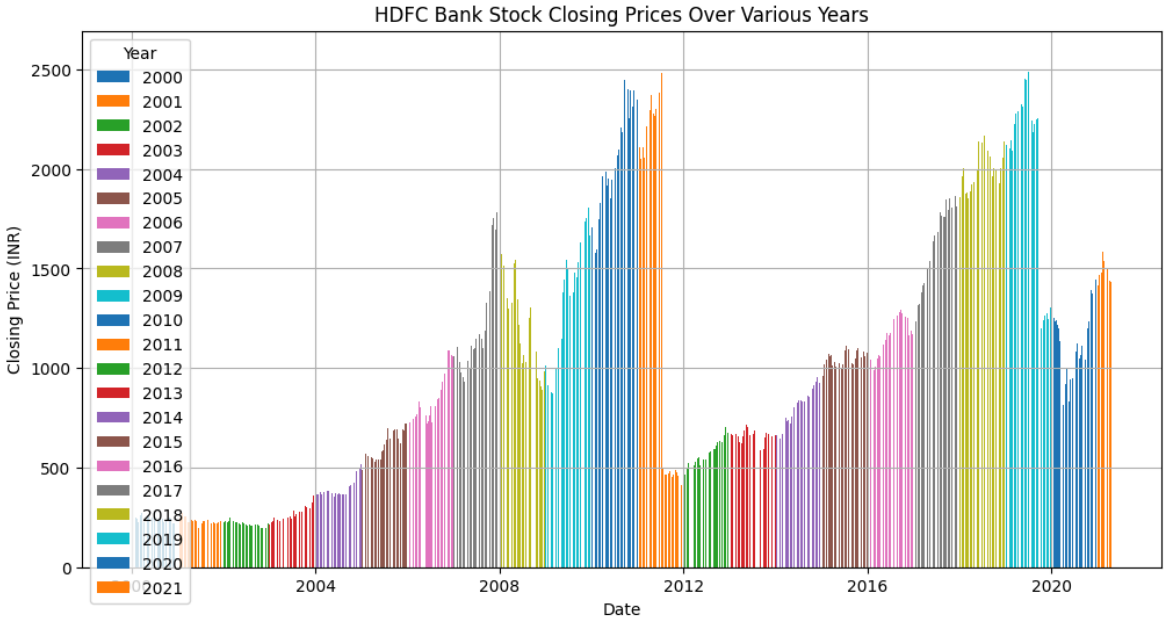
Out[ ]:

	Date	Prev Close	Open	High	Low	I
count	5306	5306.000000	5306.000000	5306.000000	5306.000000	5306.000
mean	2010-08-18 21:26:56.132679936	1007.093884	1007.472767	1019.986939	993.822211	1007.364
min	2000-01-03 00:00:00	157.400000	162.150000	167.900000	157.000000	163.000
25%	2005-04-13 12:00:00	479.912500	482.112500	486.912500	473.100000	480.700
50%	2010-08-17 12:00:00	934.750000	939.350000	953.950000	922.175000	935.600
75%	2015-12-17 18:00:00	1421.000000	1423.525000	1440.000000	1399.000000	1422.812
max	2021-04-30 00:00:00	2565.800000	2566.000000	2583.300000	2553.700000	2563.000
std	NaN	635.757762	635.461516	641.444674	629.502818	635.722

In [ ]:
```
plt.figure(figsize=(10, 6))
plt.bar(yearly_changes.index, yearly_changes['Close'], color=['red' if x < 0 els
plt.title('Yearly Losses and Gains of HDFC Bank Stock')
plt.xlabel('Year')
plt.ylabel('Price Difference (INR)')
plt.axhline(y=0, color='black', linestyle='--')  # Adding a horizontal line at y
plt.show()
```

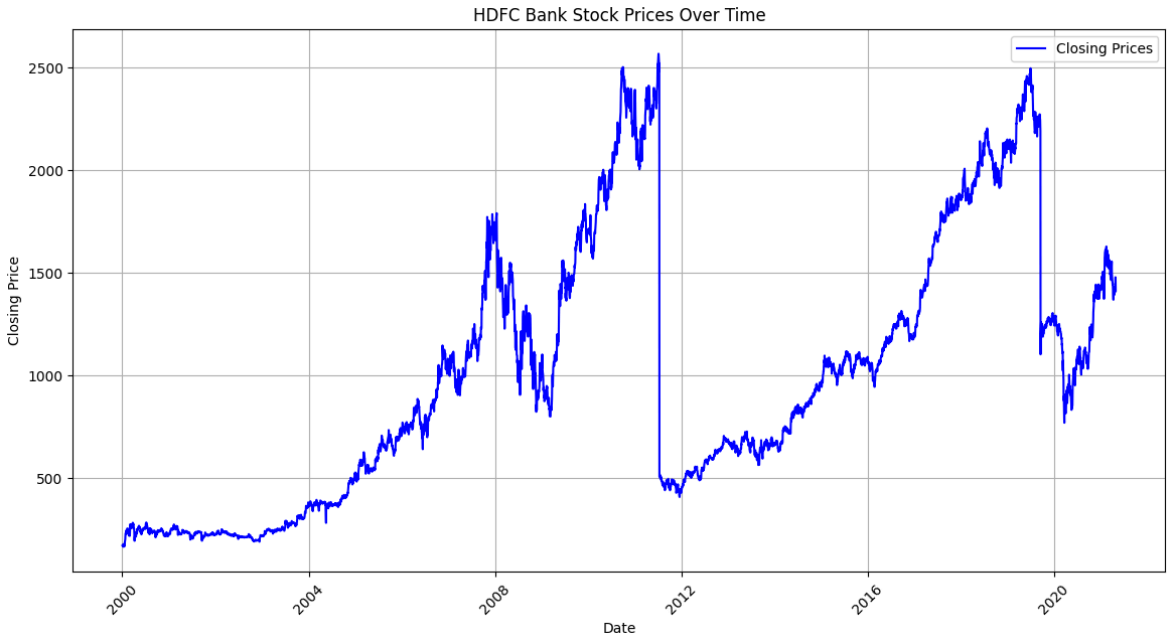


In [ ]:
```
plt.figure(figsize=(12, 6))
for year in data['Year'].unique():
    plt.bar(data[data['Year'] == year]['Date'], data[data['Year'] == year]['Clos

plt.title('HDFC Bank Stock Closing Prices Over Various Years')
plt.xlabel('Date')
plt.ylabel('Closing Price (INR)')
plt.legend(title='Year', loc='upper left')
plt.grid(True)
plt.show()
```

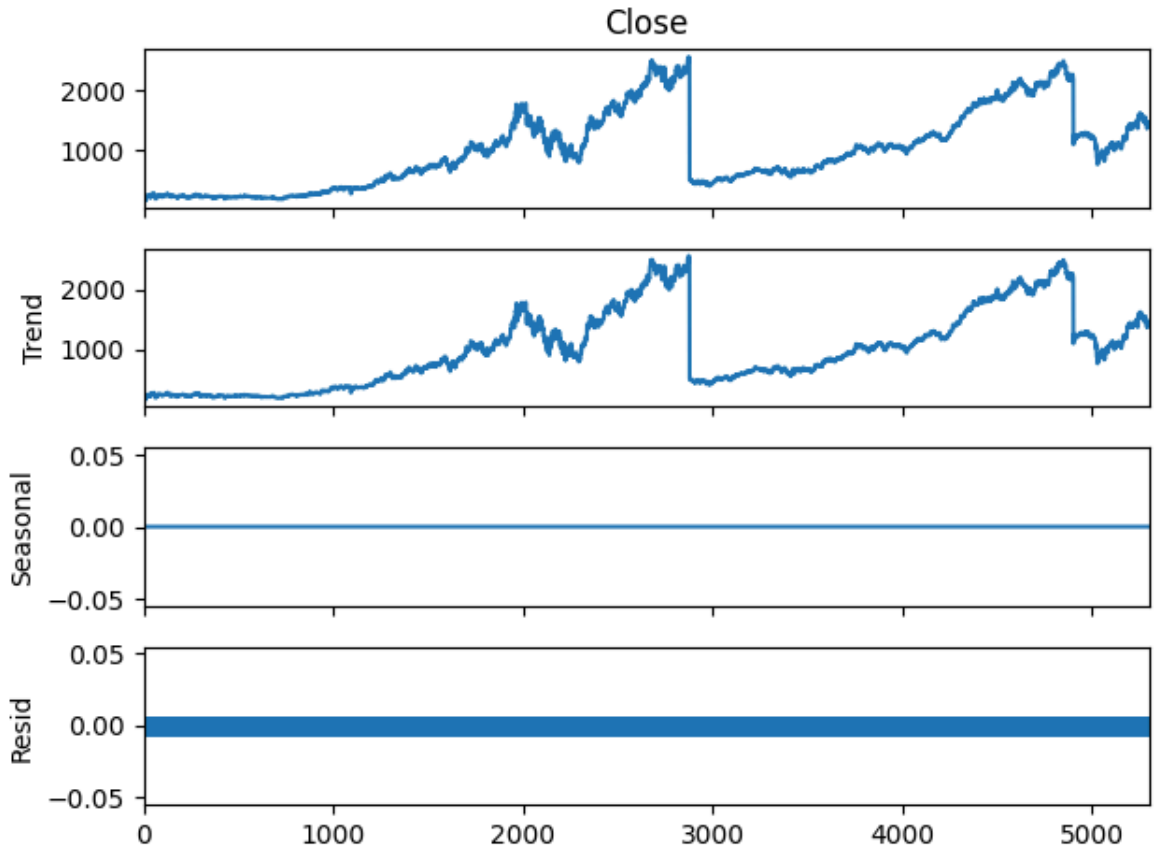


In [ ]:
```
dates = pd.to_datetime(data['Date'])
closing_prices = data['Close']

# Plotting the data
plt.figure(figsize=(14, 7))
plt.plot(dates, closing_prices, color='b', label='Closing Prices')
plt.title('HDFC Bank Stock Prices Over Time')
```

```
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
result = seasonal_decompose(data['Close'], model='additive', period=1)
result.plot()
plt.show()
```



In [ ]:
```
def adf_test(series):
    result = adfuller(series)
```

```
    print('ADF Statistic:', result[0])
    print('p-value:', result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print(f'{key}: {value}')
```

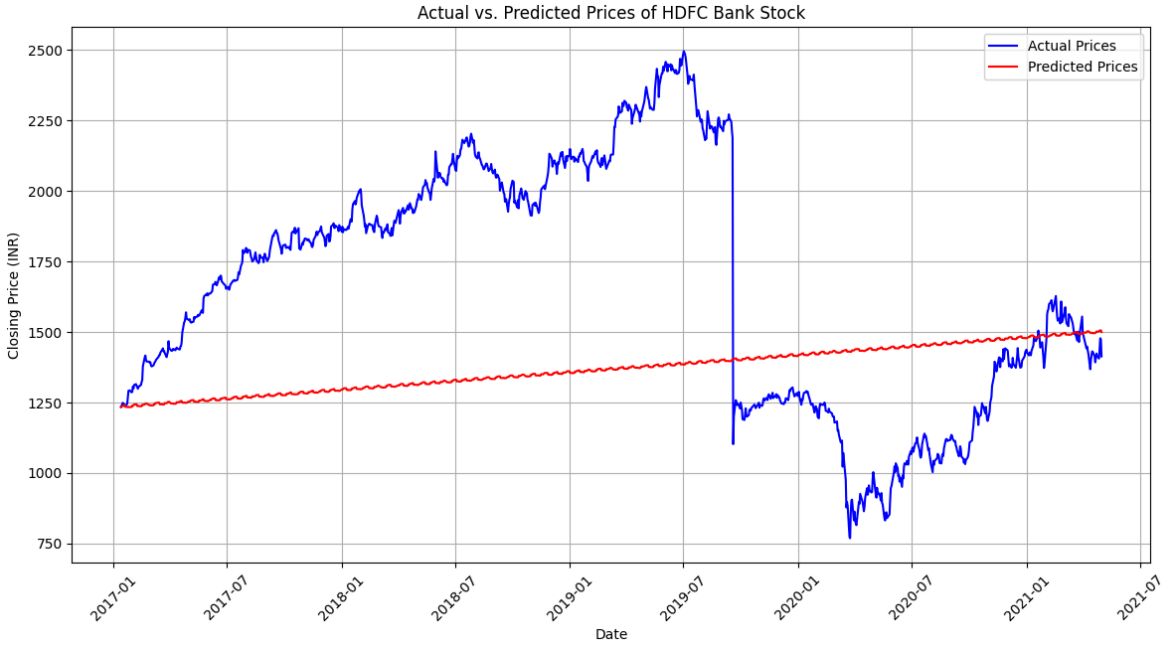
In [ ]:
```
adf_test(data['Close'])
```

```
ADF Statistic: -2.295729291137966
p-value: 0.17334905995747613
Critical Values:
1%: -3.4315832642803406
5%: -2.8620849763501828
10%: -2.5670600903865166
```

In [ ]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Split data into train and test sets
train_size = int(len(data) * 0.8)
train, test = data[:train_size], data[train_size:]

# Define and fit SARIMA model
model = SARIMAX(train['Close'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
fit_model = model.fit()

# Forecast future prices
forecast = fit_model.forecast(steps=len(test))

# Evaluate model performance
mae = mean_absolute_error(test['Close'], forecast)
mse = mean_squared_error(test['Close'], forecast)
rmse = np.sqrt(mse)

print('Mean Absolute Error:', mae)
print('Mean Squared Error:', mse)
print('Root Mean Squared Error:', rmse)
```

```
Mean Absolute Error: 479.17319105420734
Mean Squared Error: 310755.24079332955
Root Mean Squared Error: 557.4542499553928
```

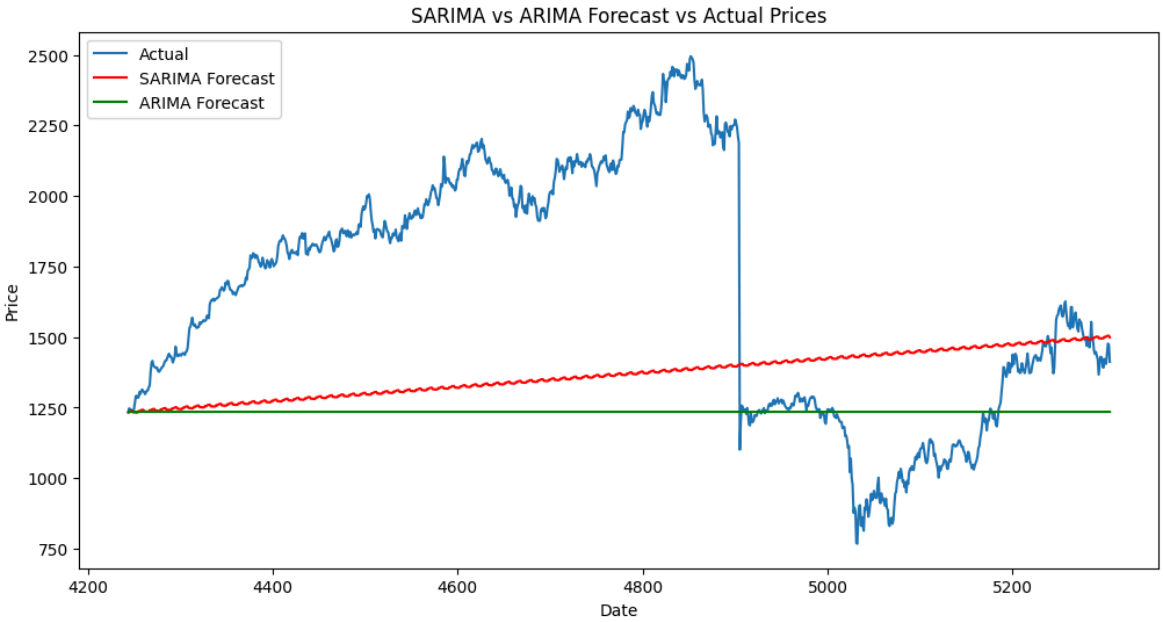
In [ ]:
```
# Visualize actual vs. predicted prices
plt.figure(figsize=(14, 7))
plt.plot(test['Date'], test['Close'], color='blue', label='Actual Prices')
plt.plot(test['Date'], forecast, color='red', label='Predicted Prices')
plt.title('Actual vs. Predicted Prices of HDFC Bank Stock')
plt.xlabel('Date')
plt.ylabel('Closing Price (INR)')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```python
from statsmodels.tsa.arima.model import ARIMA

# Define and fit ARIMA model
arima_model = ARIMA(train['Close'], order=(1, 1, 1))
arima_fit = arima_model.fit()

# Forecast future prices using ARIMA
arima_forecast = arima_fit.forecast(steps=len(test))

# Evaluate model performance
arima_mae = mean_absolute_error(test['Close'], arima_forecast)
arima_mse = mean_squared_error(test['Close'], arima_forecast)
arima_rmse = np.sqrt(arima_mse)

print('ARIMA Mean Absolute Error:', arima_mae)
print('ARIMA Mean Squared Error:', arima_mse)
print('ARIMA Root Mean Squared Error:', arima_rmse)
```

```
ARIMA Mean Absolute Error: 498.5845831710436
ARIMA Mean Squared Error: 375162.5307971079
ARIMA Root Mean Squared Error: 612.5051271598531
```

In [ ]:
```python
plt.figure(figsize=(12, 6))
plt.plot(test.index, test['Close'], label='Actual')
plt.plot(test.index, forecast, color='red', label='SARIMA Forecast')
plt.plot(test.index, arima_forecast, color='green', label='ARIMA Forecast')
plt.title('SARIMA vs ARIMA Forecast vs Actual Prices')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend(loc='upper left')
plt.show()
```


SARIMA vs ARIMA Forecast vs Actual Prices
Actual
SARIMA Forecast
ARIMA Forecast
Price
Date
2500
2250
2000
1750
1500
1250
1000
750
4200
4400
4600
4800
5000
5200