In [28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
```

In [29]:
```
dogecoin_prices = pd.read_csv('DOGE-USD.csv')
dogecoin_prices
```

Out[29]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-11-09	0.001207	0.001415	0.001181	0.001415	0.001415	6259550.0
1	2017-11-10	0.001421	0.001431	0.001125	0.001163	0.001163	4246520.0
2	2017-11-11	0.001146	0.001257	0.001141	0.001201	0.001201	2231080.0
3	2017-11-12	0.001189	0.001210	0.001002	0.001038	0.001038	3288960.0
4	2017-11-13	0.001046	0.001212	0.001019	0.001211	0.001211	2481270.0
...	...	...	...	...	...	...	...
1756	2022-08-31	0.061534	0.063333	0.061058	0.061330	0.061330	309748693.0
1757	2022-09-01	0.061336	0.062479	0.060194	0.062372	0.062372	328765413.0
1758	2022-09-02	0.062372	0.062712	0.060947	0.061635	0.061635	273453013.0
1759	2022-09-03	NaN	NaN	NaN	NaN	NaN	NaN
1760	2022-09-04	0.062682	0.062744	0.062667	0.062696	0.062696	297513408.0

1761 rows × 7 columns

In [30]:
```
dogecoin_prices['date'] = pd.to_datetime(dogecoin_prices['Date'])
```

In [31]:
```
elon_tweets = pd.read_csv('TweetsElonMusk.csv')
elon_tweets
```

Out[31]:

	id	conversation_id	created_at	date	time	timezone	
0	1381273474400800773	1381002894032347138	2021-04-11 18:50:33 EEST	2021-04-11	18:50:33	300	4
1	1381273076709478403	1372444955050971142	2021-04-11 18:48:58 EEST	2021-04-11	18:48:58	300	4
2	1381258144916008964	1381230136918433792	2021-04-11 17:49:38 EEST	2021-04-11	17:49:38	300	4
3	1381221447322935303	1381221447322935303	2021-04-11 15:23:49 EEST	2021-04-11	15:23:49	300	4
4	1381129584435818496	1381079981485252611	2021-04-11 09:18:47 EEST	2021-04-11	09:18:47	300	4
...	...	...	...	...	...	...	
12557	1382255613665701894	1382189497694121990	2021-04-14 11:53:14 EEST	2021-04-14	11:53:14	300	4
12558	1382239892445401089	1382189497694121990	2021-04-14 10:50:45 EEST	2021-04-14	10:50:45	300	4
12559	1382239304097824768	1382189497694121990	2021-04-14 10:48:25 EEST	2021-04-14	10:48:25	300	4
12560	1382131928619495429	1382046129450258434	2021-04-14 03:41:45 EEST	2021-04-14	03:41:45	300	4
12561	1382052264802721792	1382046129450258434	2021-04-13 22:25:11 EEST	2021-04-13	22:25:11	300	4

12562 rows × 36 columns

In [32]:
```
elon_tweets['date'] = pd.to_datetime(elon_tweets['date'])
```

In [33]:
```
elon_tweets['tweet']
```

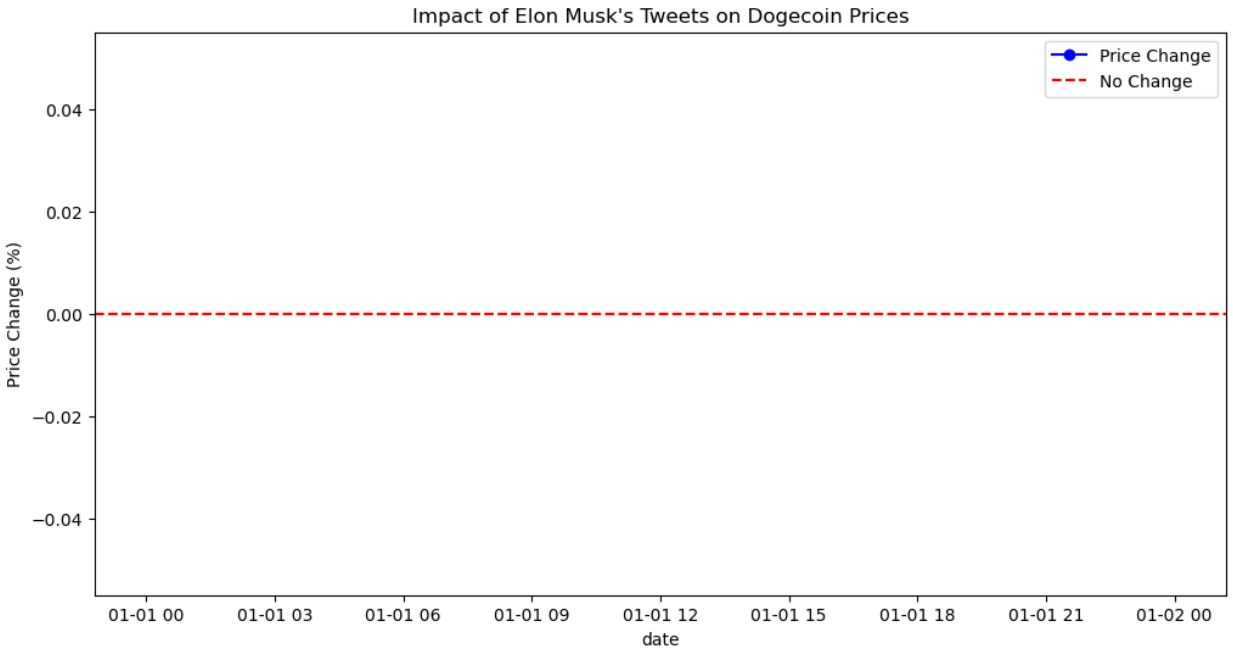
Out[33]:
```
0        @vincent13031925 For now. Costs are decreasing...
1                                Love this beautiful shot
2        @agnostoxxx @CathieDWood @ARKInvest Trust the ...
3                     The art In Cyberpunk is incredible
4                                          @itsALLrisky 🤣🤣
                              ...
12557    @eugenelee3 @PPathole @SpaceX @Tesla Yeah, not...
12558    @PPathole @SpaceX @Tesla That was my night job...
12559    @PPathole @SpaceX @Tesla True. Ancient times …...
12560                   @Erdayastronaut @Tesla Absolutely
12561    @Erdayastronaut @Tesla Tesla is building up co...
Name: tweet, Length: 12562, dtype: object
```

In [ ]:
```
```

In [34]:
```
dogecoin_tweets = elon_tweets[elon_tweets['tweet'].str.contains('Dogecoin')]
```

In [35]:
```
time_window = 6
```

In [62]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime


# Filter tweets that mention Dogecoin
dogecoin_tweets = elon_tweets[elon_tweets['tweet'].str.contains('DOGE')]

# Define the time window to analyze price changes after a tweet (in hours)
time_window = 6

# Create a new column to store the price change after each tweet
dogecoin_tweets['Price_Change'] = np.nan

# Iterate through each tweet
for index, tweet in dogecoin_tweets.iterrows():
    tweet_date = tweet['date']
    tweet_close = dogecoin_prices.loc[dogecoin_prices['date'] == tweet_date, 'C
    future_closes = dogecoin_prices.loc[(dogecoin_prices['date'] > tweet_date)
                                        (dogecoin_prices['date'] <= tweet_date
                                        'Close']
    if not future_closes.empty:
        price_change = (future_closes.iloc[-1] - tweet_close) / tweet_close * 1
        dogecoin_tweets.loc[index, 'Price_Change'] = price_change

# Plot the price change after each tweet
plt.figure(figsize=(12, 6))
plt.plot(dogecoin_tweets['date'], dogecoin_tweets['Price_Change'], 'bo-', label
plt.axhline(0, color='r', linestyle='--', label='No Change')
plt.xlabel('date')
plt.ylabel('Price Change (%)')
plt.title("Impact of Elon Musk's Tweets on Dogecoin Prices")
plt.legend()
plt.show()
```



In [72]:
```
filtered_prices = dogecoin_prices[dogecoin_prices['Date'] < pd.to_datetime('201
filtered_prices['daily_returns'] = filtered_prices['Close'].pct_change()
```

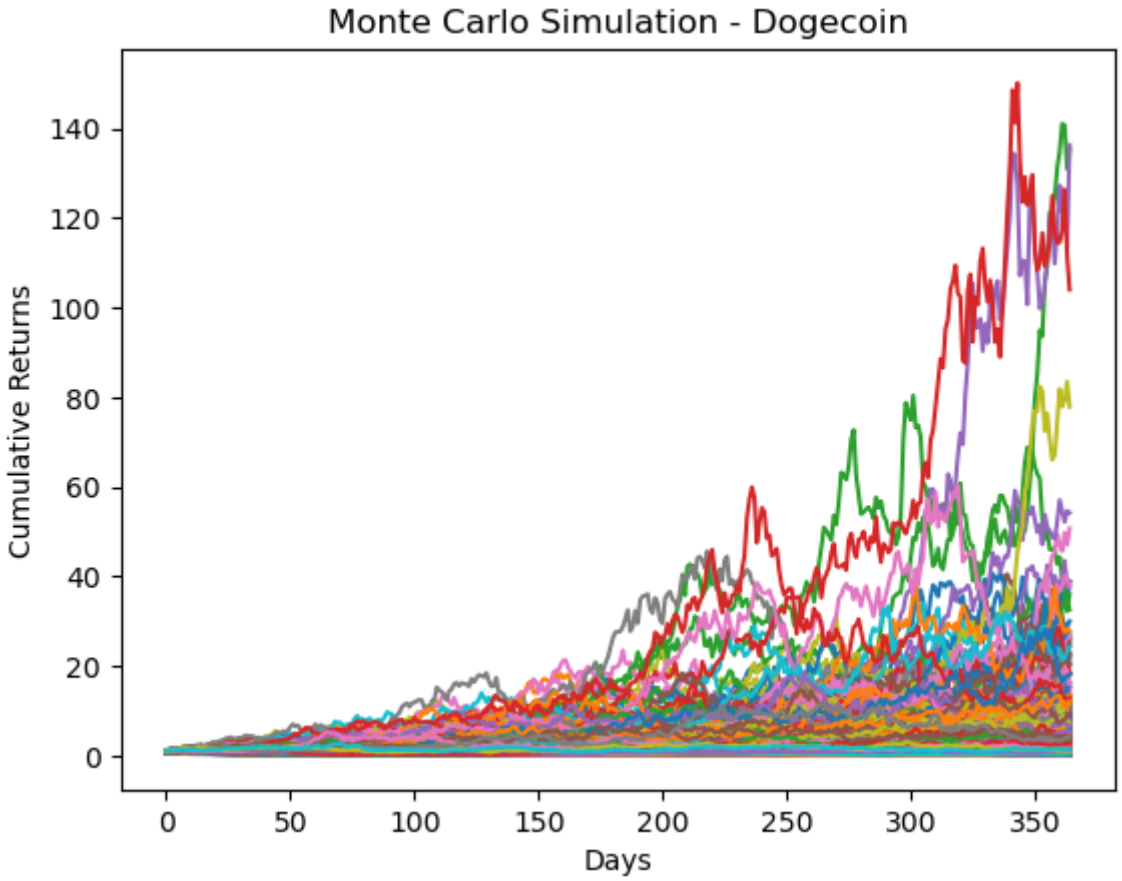
```
/var/folders/xh/qqlnwhcn7c7cv5c7s2s5d59h0000gn/T/ipykernel_21919/686570824.py:
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_prices['daily_returns'] = filtered_prices['Close'].pct_change()
```

In [73]:
```
import numpy as np

# Parameters for the simulation
num_simulations = 1000
num_days = 365  # Number of days to simulate into the future

# Calculate mean and standard deviation of historical daily returns
mean_return = filtered_prices['daily_returns'].mean()
std_return = filtered_prices['daily_returns'].std()

# Generate random samples for each simulation
simulations = np.random.normal(loc=mean_return, scale=std_return, size=(num_day

# Calculate the cumulative sum of returns for each simulation
cumulative_returns = (simulations + 1).cumprod(axis=0)
```

In [74]:
```
import numpy as np

# Parameters for the simulation
num_simulations = 1000
num_days = 365  # Number of days to simulate into the future

# Calculate mean and standard deviation of historical daily returns
mean_return = filtered_prices['daily_returns'].mean()
std_return = filtered_prices['daily_returns'].std()
```

```
# Generate random samples for each simulation
simulations = np.random.normal(loc=mean_return, scale=std_return, size=(num_day

# Calculate the cumulative sum of returns for each simulation
cumulative_returns = (simulations + 1).cumprod(axis=0)
```

In [75]:
```
import matplotlib.pyplot as plt

# Plot the cumulative returns for each simulation
plt.plot(cumulative_returns)
plt.xlabel('Days')
plt.ylabel('Cumulative Returns')
plt.title('Monte Carlo Simulation - Dogecoin')
plt.show()
```



In [41]:
```
# Calculate the average price for each simulation day
average_prices = cumulative_returns.mean(axis=1)

# Calculate the standard deviation of prices for each simulation day
std_prices = cumulative_returns.std(axis=1)
```

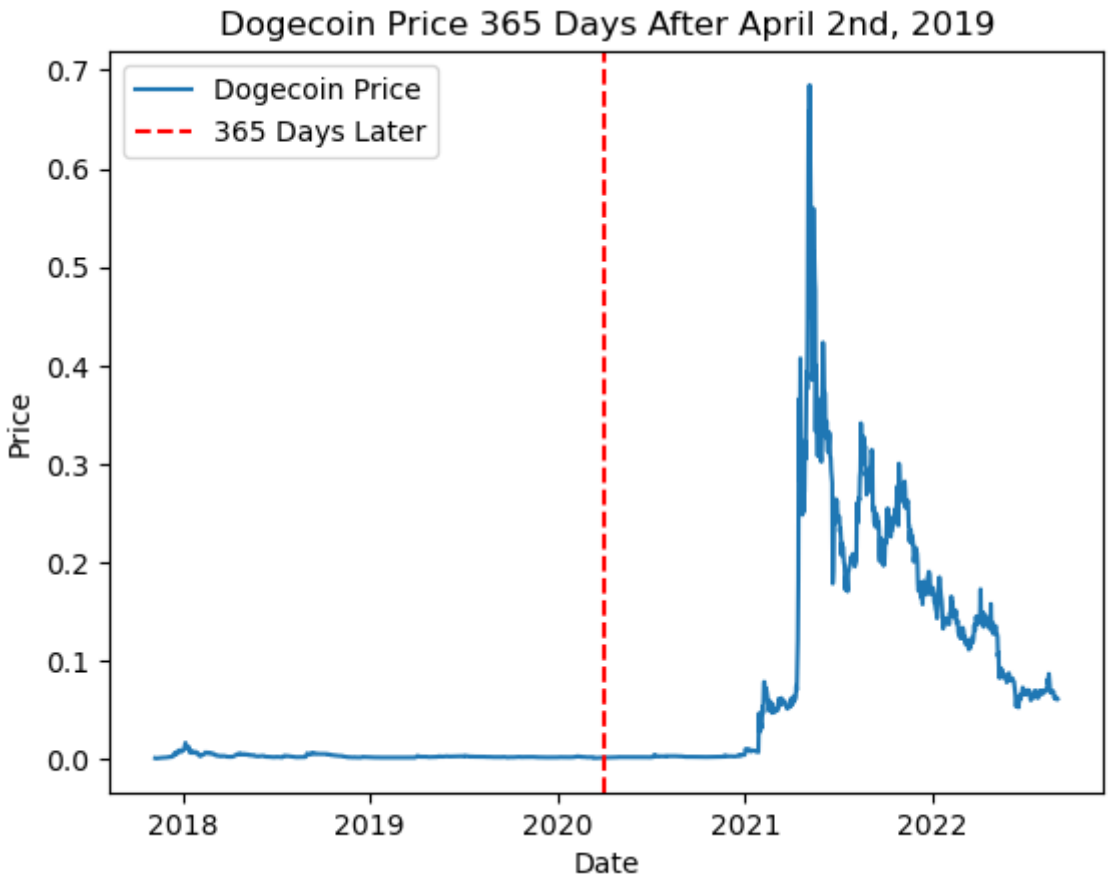
In [42]:
```
average_prices.mean()
```

Out[42]:
```
2.123764687918773
```

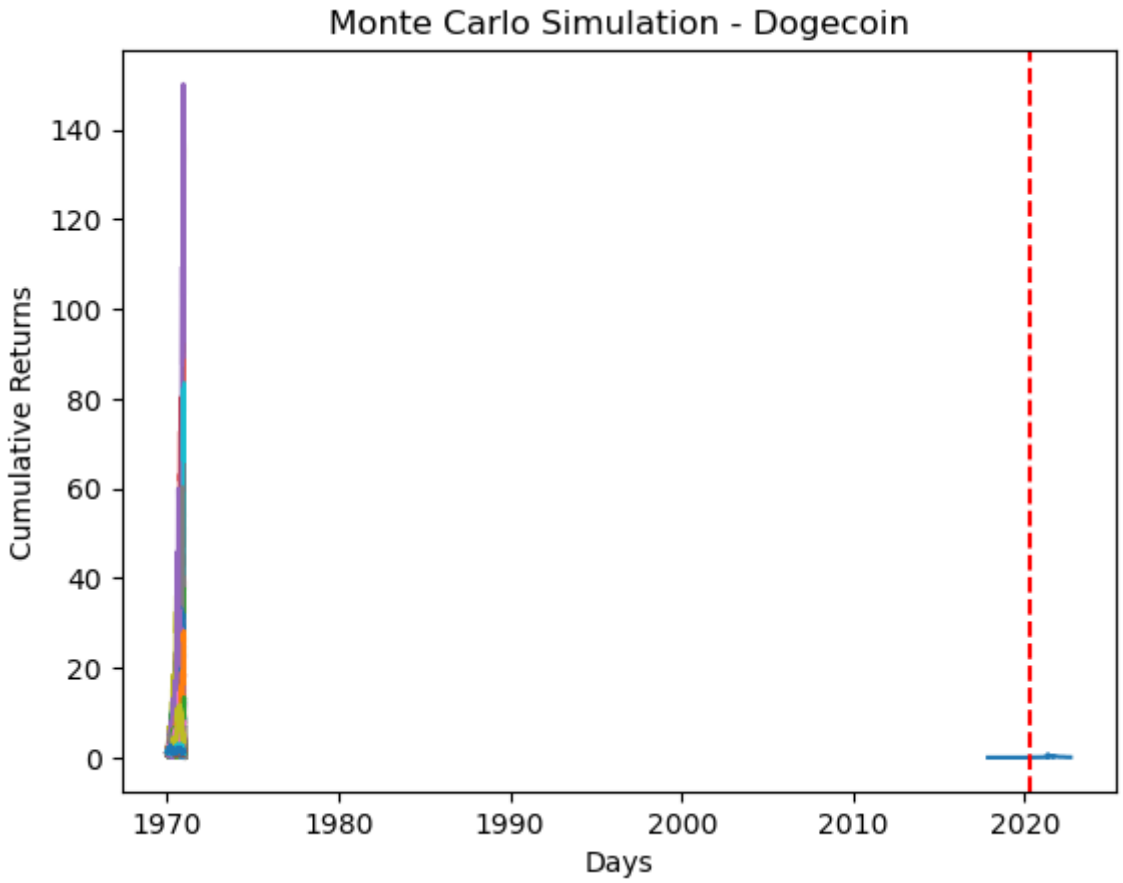
In [43]:
```
std_prices.std()
```

Out[43]:
```
2.2965862451047667
```

In [67]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the Dogecoin price data
dogecoin_prices = pd.read_csv('DOGE-USD.csv')

# Convert the date column to datetime format
dogecoin_prices['Date'] = pd.to_datetime(dogecoin_prices['Date'])

# Filter the dataset to include data from April 2nd, 2019
filtered_prices = dogecoin_prices[dogecoin_prices['Date'] == pd.to_datetime('2(

# Calculate the target date after 365 days
target_date = filtered_prices['Date'] + pd.DateOffset(days=365)

# Get the price exactly 365 days after April 2nd, 2019
target_price = dogecoin_prices[dogecoin_prices['Date'] == target_date.iloc[0]][

# Plot the price
plt.plot(dogecoin_prices['Date'], dogecoin_prices['Close'], label='Dogecoin Pri
plt.axvline(target_date.iloc[0], color='r', linestyle='--', label='365 Days Lat

# Set the plot title and labels
plt.title("Dogecoin Price 365 Days After April 2nd, 2019")
plt.xlabel("Date")
plt.ylabel("Price")

# Display the legend and show the plot
plt.legend()
plt.show()
```



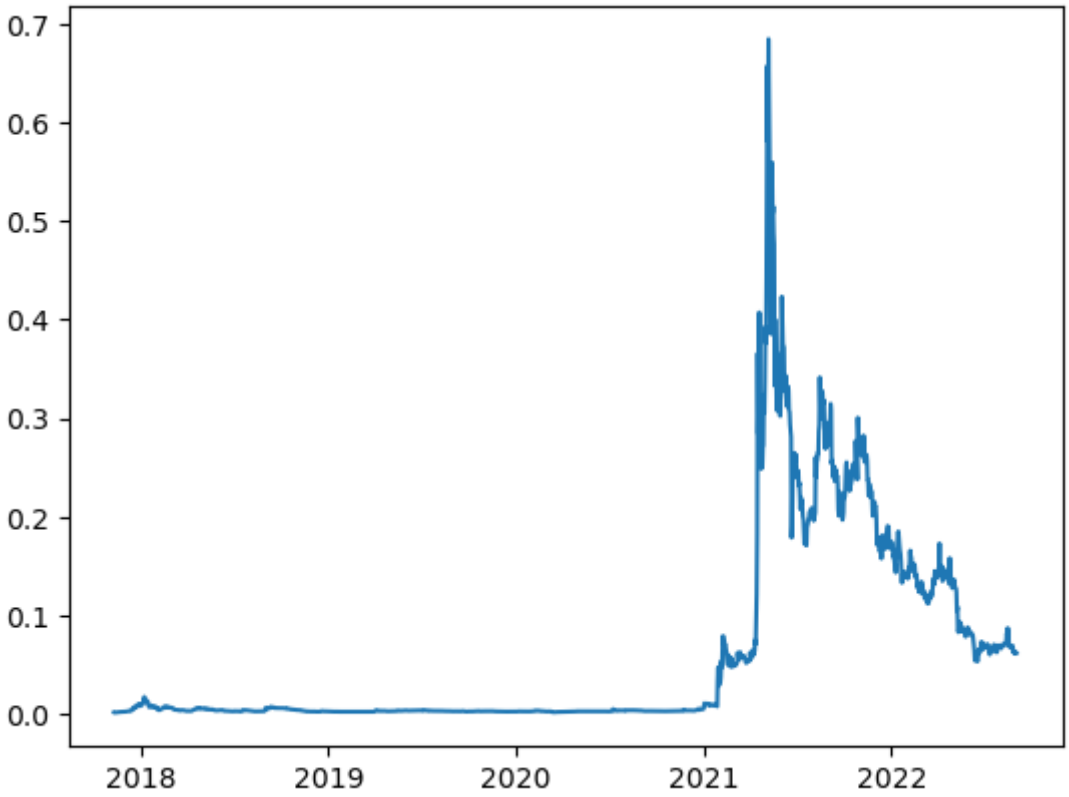
In [76]:
```
import matplotlib.pyplot as plt

# Plot the cumulative returns for each simulation
plt.plot(dogecoin_prices['Date'], dogecoin_prices['Close'], label='Dogecoin Pri
plt.axvline(target_date.iloc[0], color='r', linestyle='--', label='365 Days Lat
plt.plot(cumulative_returns)
plt.xlabel('Days')
plt.ylabel('Cumulative Returns')
plt.title('Monte Carlo Simulation - Dogecoin')
plt.show()
```



In [66]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Convert the date column to datetime format
dogecoin_prices['Date'] = pd.to_datetime(dogecoin_prices['Date'])

# Load the Elon Musk's tweets data
#elon_tweets = pd.read_csv('elon_tweets.csv')

# Convert the date column to datetime format
elon_tweets['date'] = pd.to_datetime(elon_tweets['date'])

# Filter Elon Musk's tweets mentioning Dogecoin
dogecoin_tweets = elon_tweets[elon_tweets['tweet'].str.contains('Dogecoin')]

# Plot the Dogecoin price
```

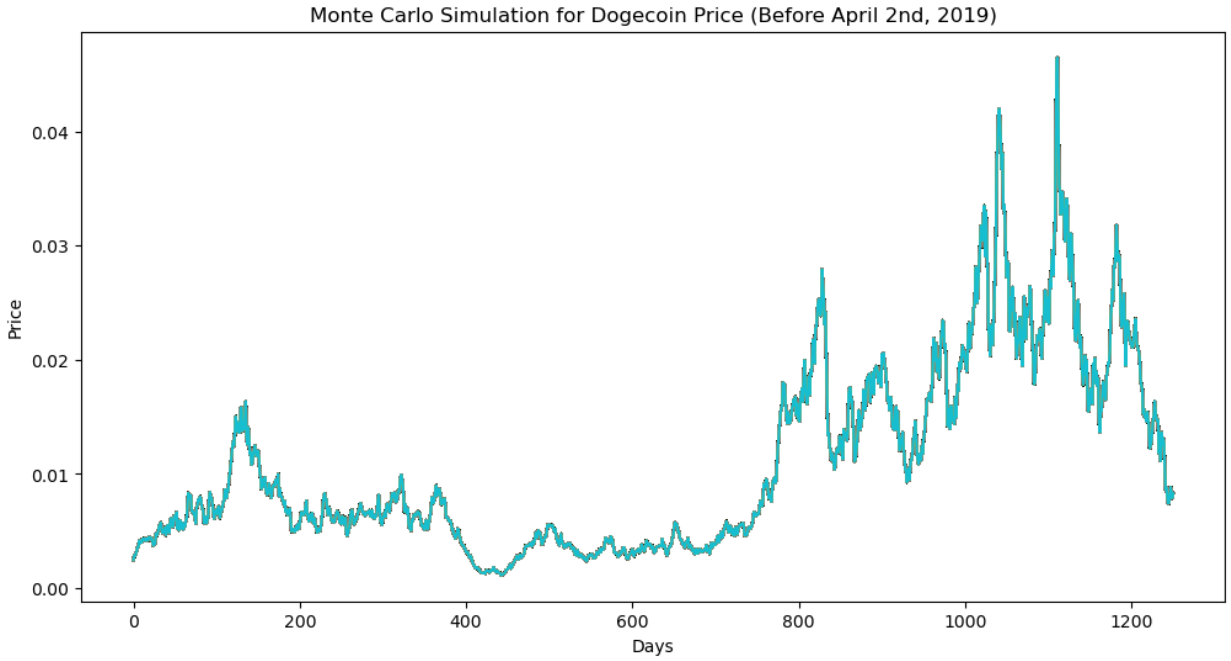
```
plt.plot(dogecoin_prices['Date'], dogecoin_prices['Close'], label='Dogecoin Pri

# Plot Elon Musk's tweets
plt.scatter(dogecoin_tweets['Date'], dogecoin_prices.loc[dogecoin_tweets['Date'
            marker='o', color='r', label="Elon Musk's Tweets")

# Set the plot title and labels
plt.title("Impact of Elon Musk's Tweets on Dogecoin Price")
plt.xlabel("Date")
plt.ylabel("Price")

# Display the legend and show the plot
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/core/indexes/
base.py:3802, in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/_libs/index.p
yx:138, in pandas._libs.index.IndexEngine.get_loc()

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/_libs/index.p
yx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.P
yObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.P
yObjectHashTable.get_item()

KeyError: 'Date'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[66], line 23
     20 plt.plot(dogecoin_prices['Date'], dogecoin_prices['Close'], label='Dog
ecoin Price')
     22 # Plot Elon Musk's tweets
---> 23 plt.scatter(dogecoin_tweets['Date'], dogecoin_prices.loc[dogecoin_twee
ts['Date']]['Close'],
     24             marker='o', color='r', label="Elon Musk's Tweets")
     26 # Set the plot title and labels
     27 plt.title("Impact of Elon Musk's Tweets on Dogecoin Price")

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/core/frame.p
y:3807, in DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/core/indexes/
base.py:3804, in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'Date'
```



In [78]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Filter the data before April 2nd, 2019
start_date = pd.to_datetime('2019-04-02')
filtered_prices = dogecoin_prices[dogecoin_prices['Date'] < start_date]

# Select the 'Close' prices
prices = filtered_prices['Close'].values

# Calculate the daily returns
returns = np.diff(prices) / prices[:-1]

# Calculate the mean and standard deviation of daily returns
mean_return = np.mean(returns)
std_return = np.std(returns)

# Set the simulation parameters
num_simulations = 5000
num_days = 1251

# Perform the Monte Carlo simulation
simulations = np.zeros((num_days, num_simulations))
simulations[0, :] = prices[-1]

for i in range(1, num_days):
    drift = 1 + mean_return
    shock = std_return * np.random.normal()
    simulations[i, :] = simulations[i-1, :] * (drift + shock)
```

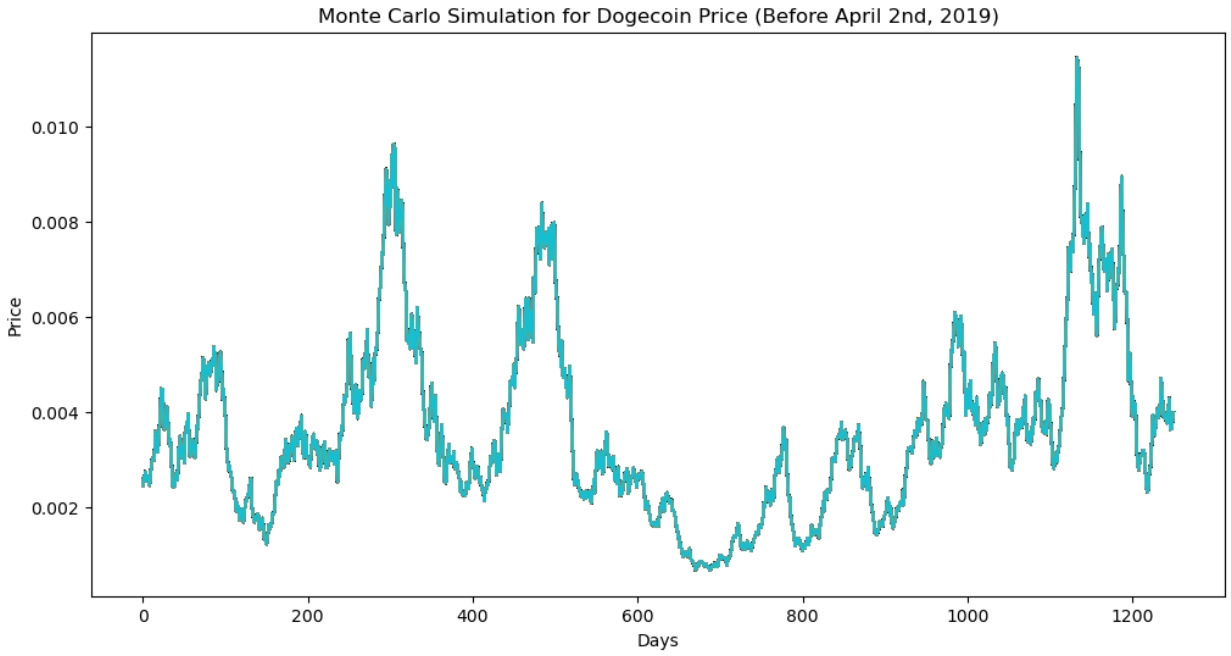
```
# Plot the Monte Carlo simulations
plt.figure(figsize=(12, 6))
plt.plot(simulations)
plt.title("Monte Carlo Simulation for Dogecoin Price (Before April 2nd, 2019)")
plt.xlabel("Days")
plt.ylabel("Price")
plt.show()
```



In [86]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Filter the data before April 2nd, 2019
start_date = pd.to_datetime('2019-04-02')
filtered_prices = dogecoin_prices[dogecoin_prices['Date'] < start_date]

# Select the 'Close' prices
prices = filtered_prices['Close'].values

# Calculate the daily returns
returns = np.diff(prices) / prices[:-1]

# Calculate the mean and standard deviation of daily returns
mean_return = np.mean(returns)
std_return = np.std(returns)

# Set the simulation parameters
num_simulations = 5000
num_days = 1251

# Perform the Monte Carlo simulation
simulations = np.zeros((num_days, num_simulations))
simulations[0, :] = prices[-1]
```

```
for i in range(1, num_days):
    drift = 1 + mean_return
    shock = std_return * np.random.normal()
    simulations[i, :] = simulations[i-1, :] * (drift + shock)

# Plot the Monte Carlo simulations
plt.figure(figsize=(12, 6))
plt.plot(simulations)
plt.title("Monte Carlo Simulation for Dogecoin Price (Before April 2nd, 2019)")
plt.xlabel("Days")
plt.ylabel("Price")
plt.show()
import numpy as np

# Perform the Monte Carlo simulation (assuming you already have this code)
simulations = np.zeros((num_days, num_simulations))
simulations[0, :] = prices[-1]
# ... Simulation code ...

# Calculate the mean and standard deviation of the simulated prices
```

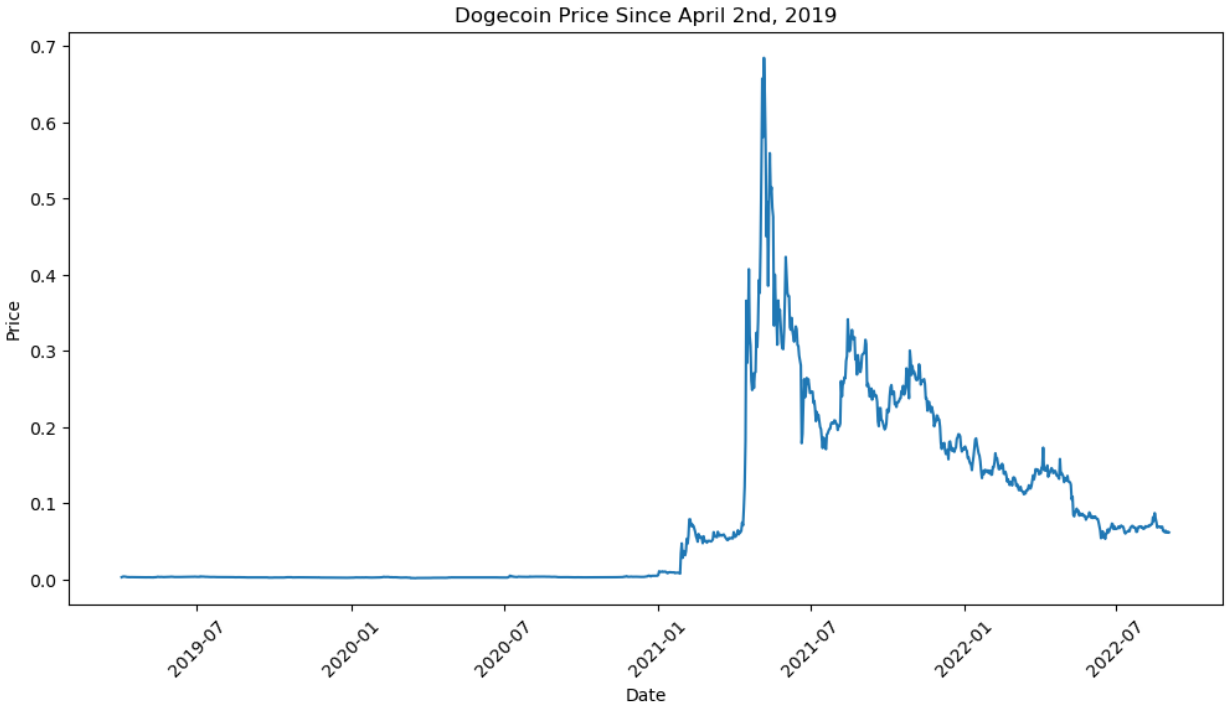


In [84]:

```
import numpy as np

# Perform the Monte Carlo simulation (assuming you already have this code)
simulations = np.zeros((num_days, num_simulations))
simulations[0, :] = prices[-1]
# ... Simulation code ...

# Calculate the mean and standard deviation of the simulated prices
mean_simulation = np.mean(simulations[-1, :])
std_simulation = np.std(simulations[-1, :])

print("Mean Simulation:", mean_simulation)
print("Standard Deviation:", std_simulation)
```

```
Mean Simulation: 0.0
Standard Deviation: 0.0
```

In [82]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Convert the date column to datetime format
dogecoin_prices['Date'] = pd.to_datetime(dogecoin_prices['Date'])

# Filter the data since April 2nd, 2019
start_date = pd.to_datetime('2019-04-02')
filtered_prices = dogecoin_prices[dogecoin_prices['Date'] >= start_date]

# Plot the Dogecoin price
plt.figure(figsize=(12, 6))
plt.plot(filtered_prices['Date'], filtered_prices['Close'])
plt.title("Dogecoin Price Since April 2nd, 2019")
plt.xlabel("Date")
plt.ylabel("Price")
plt.xticks(rotation=45)
plt.show()
```



In [83]:
```
import pandas as pd
import numpy as np

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Convert the date column to datetime format
dogecoin_prices['Date'] = pd.to_datetime(dogecoin_prices['Date'])

# Filter the data since April 2nd, 2019
start_date = pd.to_datetime('2019-04-02')
filtered_prices = dogecoin_prices[dogecoin_prices['Date'] >= start_date]

# Calculate the mean and standard deviation of Dogecoin's price
```

```
mean_price = np.mean(filtered_prices['Close'])
std_price = np.std(filtered_prices['Close'])

print("Mean Price:", mean_price)
print("Standard Deviation:", std_price)
```

```
Mean Price: 0.08229928457234212
Standard Deviation: 0.11256315574009294
```

In [93]:

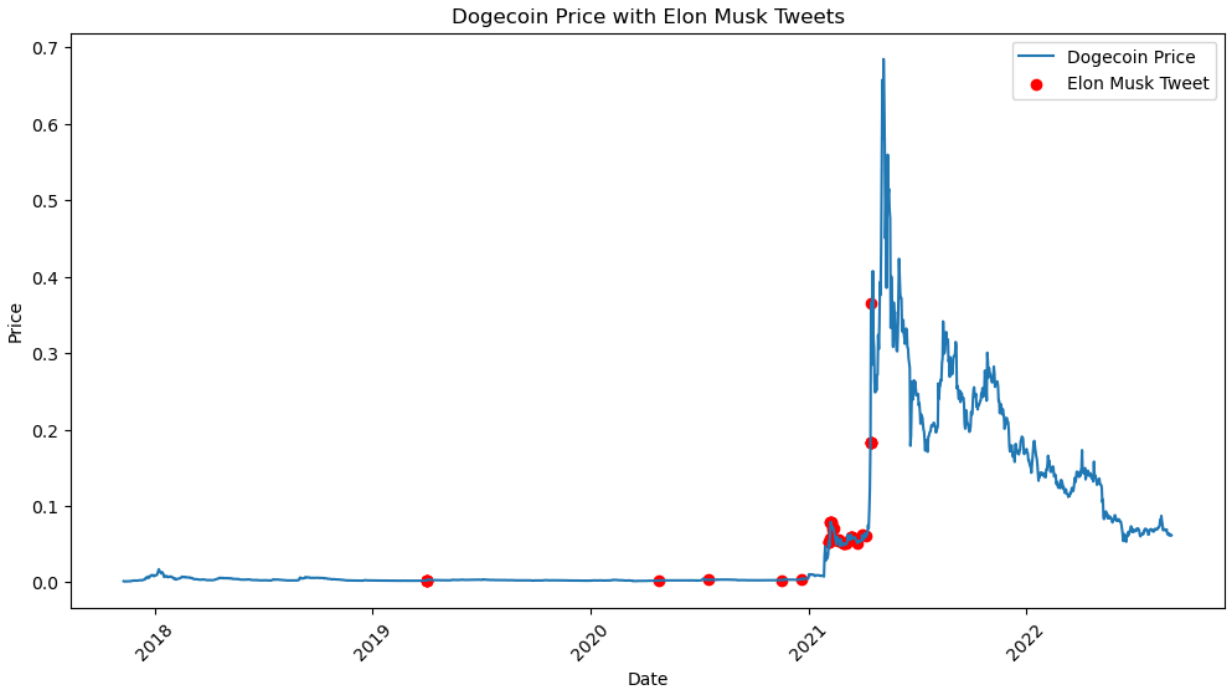
```
import pandas as pd

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Load Elon Musk's tweets data
#elon_tweets = pd.read_csv('elon_tweets.csv')

# Convert the date columns to datetime format
dogecoin_prices['date'] = pd.to_datetime(dogecoin_prices['Date'])
elon_tweets['date'] = pd.to_datetime(elon_tweets['date'])


# Filter Elon Musk's tweets mentioning Dogecoin and additional keywords
keywords = ['Dogecoin', 'crypto', 'DOGE', 'digital currency']
dogecoin_tweets = elon_tweets[elon_tweets['tweet'].str.contains('|'.join(keywor


# Merge Dogecoin price and Elon Musk's tweets based on date
merged_data = pd.merge(dogecoin_prices, dogecoin_tweets, on='date')

merged_data = merged_data.dropna()
correlation_coefficient = merged_data['Close'].corr(merged_data['retweet'])


print("Correlation Coefficient:", correlation_coefficient)
```

```
Correlation Coefficient: nan
```

In [102…

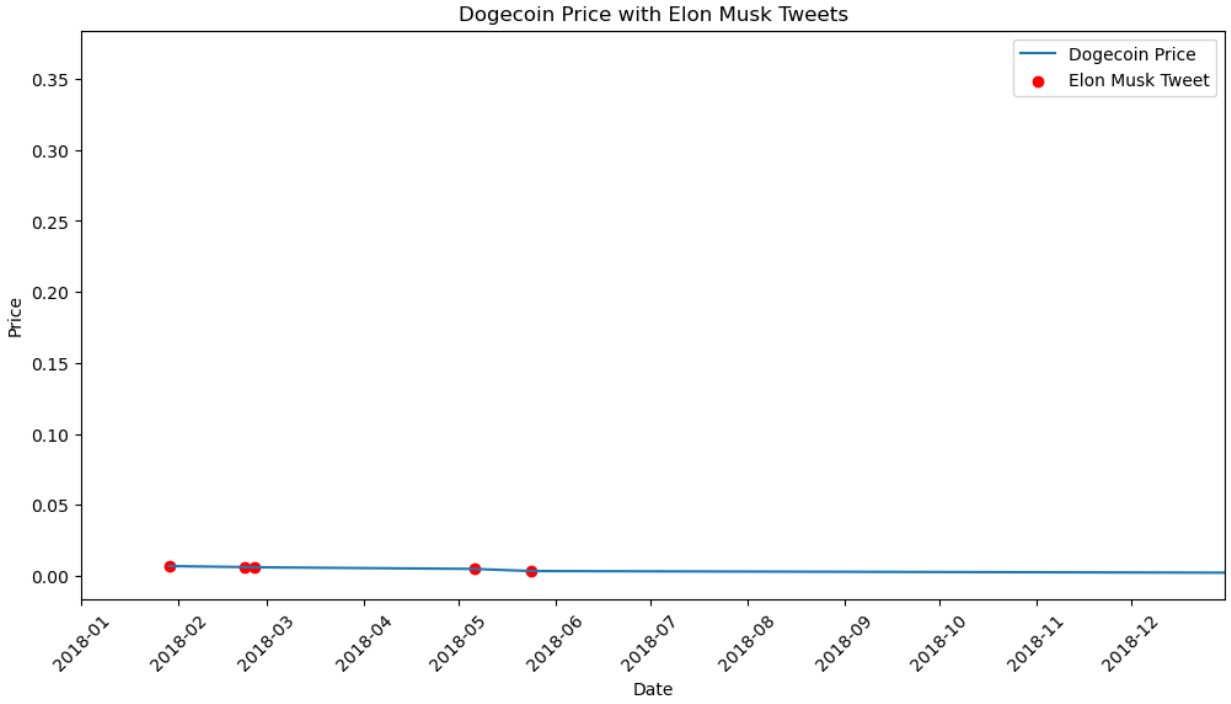
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Convert the date column to datetime format
dogecoin_prices['date'] = pd.to_datetime(dogecoin_prices['Date'])

# Load Elon Musk's tweets data
#elon_tweets = pd.read_csv('elon_tweets.csv')

# Convert the date column to datetime format
elon_tweets['date'] = pd.to_datetime(elon_tweets['date'])

keywords = ['Dogecoin', 'DOGE',]
dogecoin_tweets = elon_tweets[elon_tweets['tweet'].str.contains('|'.join(keywor


# Merge Dogecoin prices with Elon Musk tweets based on date
merged_data = pd.merge(dogecoin_prices, dogecoin_tweets, on='date', how='inner'

# Plot the Dogecoin price with markers for tweet dates
```

```
plt.figure(figsize=(12, 6))
plt.plot(dogecoin_prices['date'], dogecoin_prices['Close'], label='Dogecoin Pri
plt.scatter(merged_data['date'], merged_data['Close'], color='red', label='Elon
plt.title("Dogecoin Price with Elon Musk Tweets")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

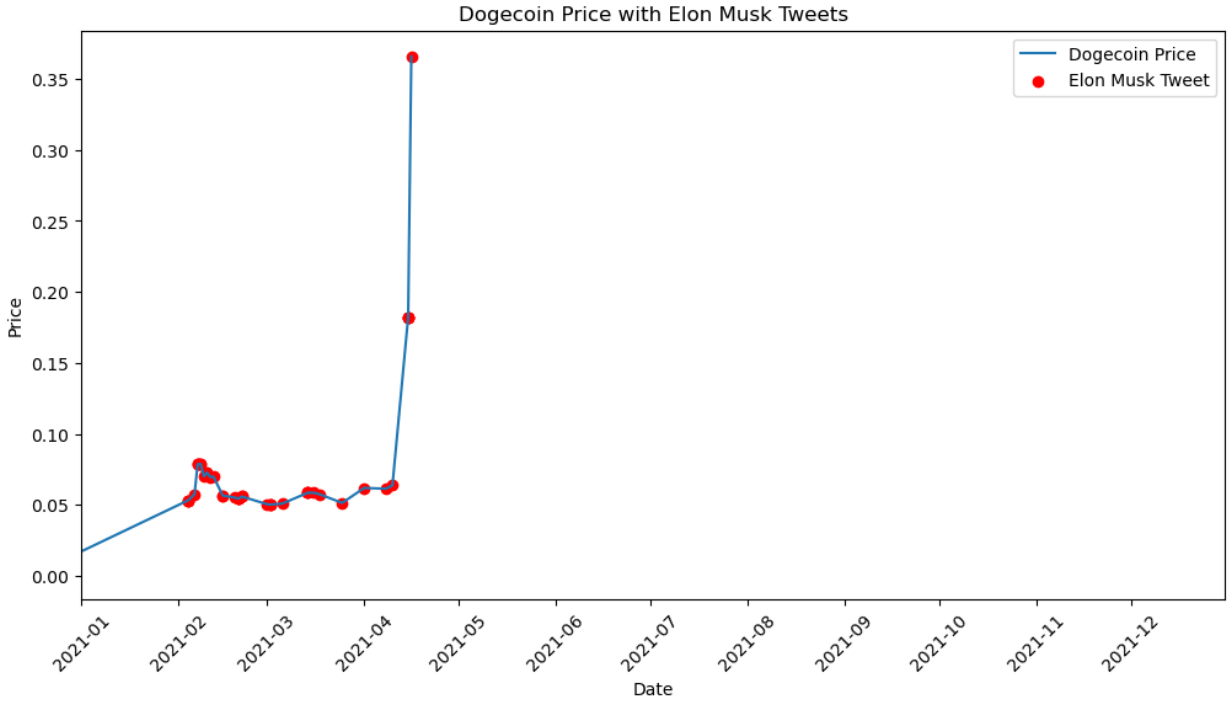


In [98]:

```
plt.figure(figsize=(12, 6))
plt.plot(merged_data['date'], merged_data['Close'], label='Dogecoin Price')
plt.scatter(merged_data['date'], merged_data['Close'], color='red', label='Elon
plt.title("Dogecoin Price with Elon Musk Tweets")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.xticks(rotation=45)

# Specify the year to zoom in on
year_to_zoom = 2018

# Set the x-axis limits to the desired time range
plt.xlim(pd.Timestamp(year=year_to_zoom, month=1, day=1), pd.Timestamp(year=yea

plt.show()
```

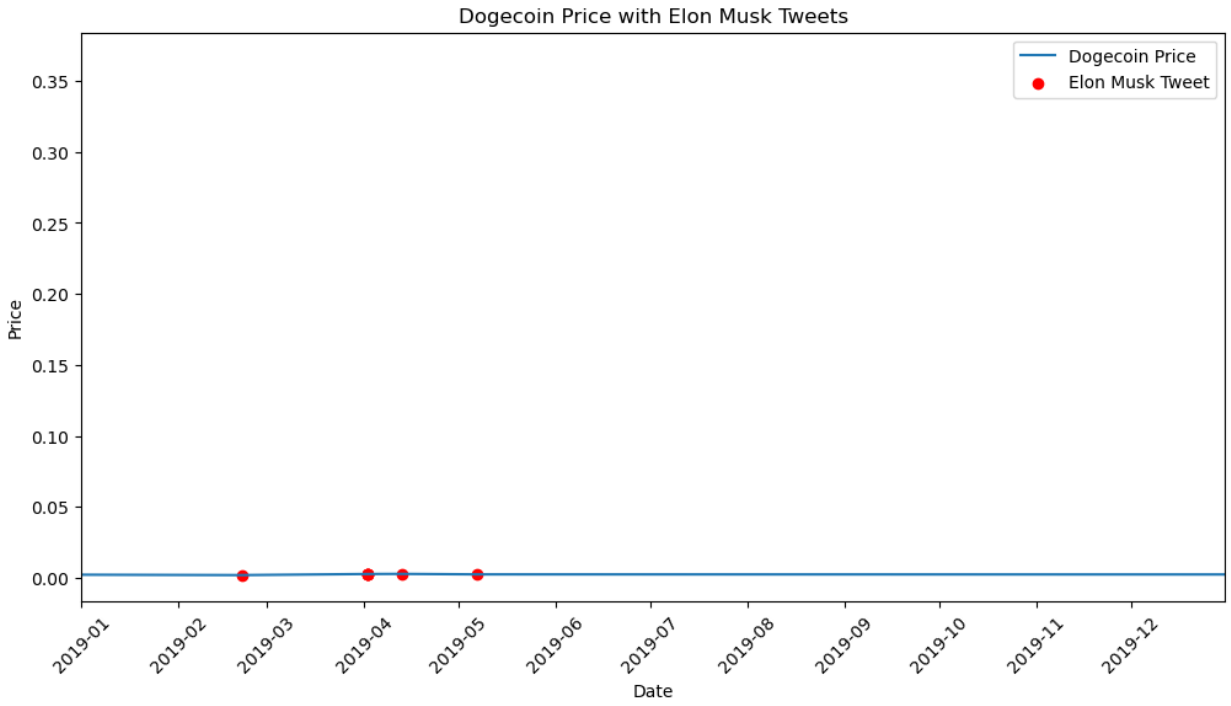


In [99]:
```
plt.figure(figsize=(12, 6))
plt.plot(merged_data['date'], merged_data['Close'], label='Dogecoin Price')
plt.scatter(merged_data['date'], merged_data['Close'], color='red', label='Elon
plt.title("Dogecoin Price with Elon Musk Tweets")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.xticks(rotation=45)

# Specify the year to zoom in on
year_to_zoom = 2021

# Set the x-axis limits to the desired time range
plt.xlim(pd.Timestamp(year=year_to_zoom, month=1, day=1), pd.Timestamp(year=yea

plt.show()
```



In [100…

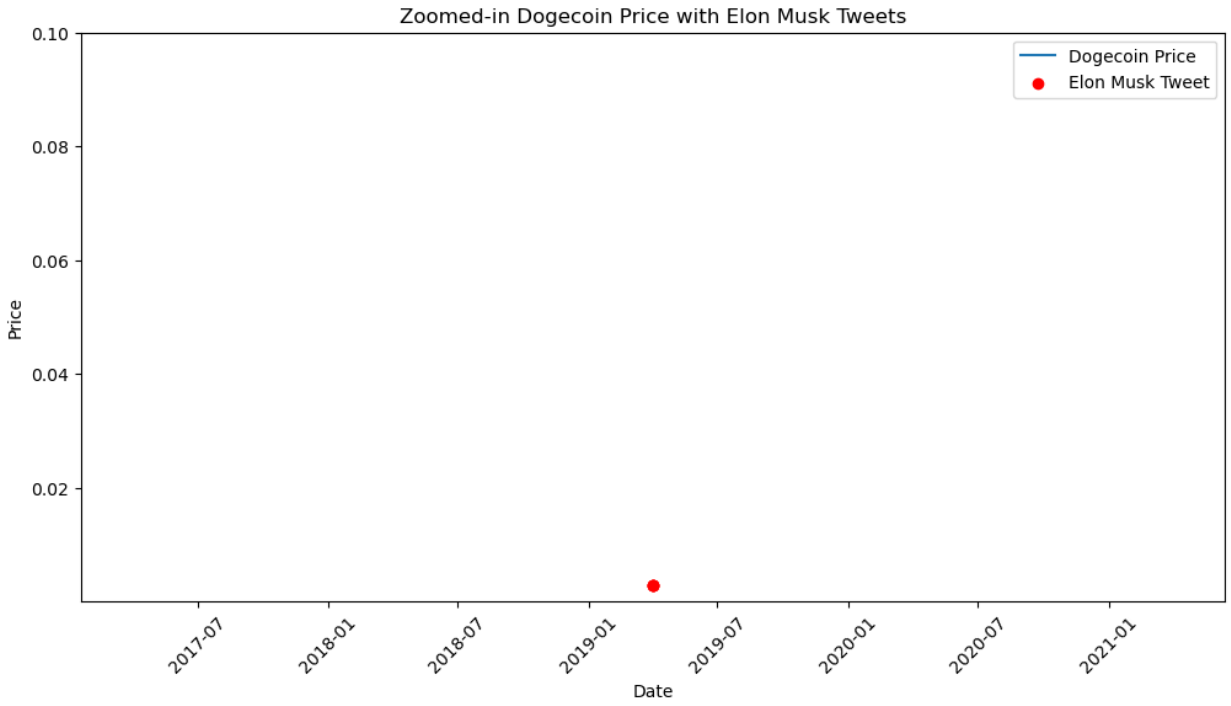
```
plt.figure(figsize=(12, 6))
plt.plot(merged_data['date'], merged_data['Close'], label='Dogecoin Price')
plt.scatter(merged_data['date'], merged_data['Close'], color='red', label='Elon
plt.title("Dogecoin Price with Elon Musk Tweets")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.xticks(rotation=45)

# Specify the year to zoom in on
year_to_zoom = 2019

# Set the x-axis limits to the desired time range
plt.xlim(pd.Timestamp(year=year_to_zoom, month=1, day=1), pd.Timestamp(year=yea

plt.show()
```



In [105…

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Convert the date column to datetime format
dogecoin_prices['date'] = pd.to_datetime(dogecoin_prices['Date'])

# Load Elon Musk's tweets data
#elon_tweets = pd.read_csv('elon_tweets.csv')

# Convert the date column to datetime format
elon_tweets['date'] = pd.to_datetime(elon_tweets['date'])

# Filter Elon Musk's tweets mentioning Dogecoin
keywords = ['Dogecoin', 'DOGE',]
dogecoin_tweets = elon_tweets[elon_tweets['tweet'].str.contains('|'.join(keywor

# Merge Dogecoin prices with Elon Musk tweets based on date
merged_data = pd.merge(dogecoin_prices, dogecoin_tweets, on='date', how='inner'

# Specify the time range to zoom in on
start_date = pd.Timestamp(year=2019, month=1, day=1)
end_date = pd.Timestamp(year=2019, month=12, day=31)

# Filter the merged data within the specified time range
zoomed_data = merged_data[(merged_data['date'] >= start_date) & (merged_data['c

# Plot the Dogecoin price with markers for tweet dates
plt.figure(figsize=(12, 6))
plt.plot(zoomed_data['date'], zoomed_data['Close'], label='Dogecoin Price')
plt.scatter(zoomed_data['date'], zoomed_data['Close'], color='red', label='Elon
plt.title("Zoomed-in Dogecoin Price with Elon Musk Tweets")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
```

```
plt.xticks(rotation=45)

# Specify the y-axis limits for price zoom
price_lower_limit = 0.0001  # Set the lower limit of the price
price_upper_limit = 0.1  # Set the upper limit of the price

# Set the y-axis limits to the desired price range
plt.ylim(price_lower_limit, price_upper_limit)

plt.show()
```



In [107…

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Load the Dogecoin price data
#dogecoin_prices = pd.read_csv('dogecoin_prices.csv')

# Convert the date column to datetime format
dogecoin_prices['date'] = pd.to_datetime(dogecoin_prices['Date'])

# Load Elon Musk's tweets data
#elon_tweets = pd.read_csv('elon_tweets.csv')

# Convert the date column to datetime format
elon_tweets['date'] = pd.to_datetime(elon_tweets['date'])

keywords = ['Dogecoin', 'DOGE',]
dogecoin_tweets = elon_tweets[elon_tweets['tweet'].str.contains('|'.join(keywor



# Merge Dogecoin prices with Elon Musk tweets based on date
merged_data = pd.merge(dogecoin_prices, dogecoin_tweets, on='date', how='inner'

# Create a feature matrix X with the tweet data
X = merged_data['sentiment_score'].values.reshape(-1, 1)  # Use the sentiment c
```

```
# Create a target variable y with the Dogecoin prices
y = merged_data['Close'].values

# Create a linear regression model and fit it to the data
regression_model = LinearRegression()
regression_model.fit(X, y)

# Print the coefficients of the linear regression model
print("Intercept:", regression_model.intercept_)
print("Coefficient:", regression_model.coef_[0])

# Predict Dogecoin prices based on the tweet data
predicted_prices = regression_model.predict(X)

# Plot the actual Dogecoin prices and the predicted prices
plt.figure(figsize=(12, 6))
plt.plot(merged_data['date'], y, label='Actual Prices')
plt.plot(merged_data['date'], predicted_prices, label='Predicted Prices')
plt.title("Dogecoin Prices: Actual vs. Predicted")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/core/indexes/
base.py:3802, in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/_libs/index.p
yx:138, in pandas._libs.index.IndexEngine.get_loc()

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/_libs/index.p
yx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.P
yObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.P
yObjectHashTable.get_item()

KeyError: 'sentiment_score'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[107], line 26
     23 merged_data = pd.merge(dogecoin_prices, dogecoin_tweets, on='date', ho
w='inner')
     25 # Create a feature matrix X with the tweet data
---> 26 X = merged_data['sentiment_score'].values.reshape(-1, 1)  # Use the se
ntiment compound score as the feature
     28 # Create a target variable y with the Dogecoin prices
     29 y = merged_data['Close'].values

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/core/frame.p
y:3807, in DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~/Desktop/UCI/anaconda3/lib/python3.10/site-packages/pandas/core/indexes/
base.py:3804, in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'sentiment_score'
```

In [108…

```
merged_data
```

Out[108]:

	Date	Open	High	Low	Close	Adj Close	Volume	date	
0	2019-04-02	0.002459	0.002863	0.002394	0.002795	0.002795	6.029836e+07	2019-04-02	111317
1	2019-04-02	0.002459	0.002863	0.002394	0.002795	0.002795	6.029836e+07	2019-04-02	111317
2	2019-04-02	0.002459	0.002863	0.002394	0.002795	0.002795	6.029836e+07	2019-04-02	111316
3	2019-04-02	0.002459	0.002863	0.002394	0.002795	0.002795	6.029836e+07	2019-04-02	111300
4	2020-04-25	0.002102	0.002146	0.002087	0.002142	0.002142	2.298104e+08	2020-04-25	125403
5	2020-07-18	0.003060	0.003629	0.003031	0.003473	0.003473	2.040809e+08	2020-07-18	128429
6	2020-11-17	0.002908	0.002976	0.002886	0.002936	0.002936	4.663146e+07	2020-11-17	132877
7	2020-12-20	0.003926	0.004678	0.003827	0.004625	0.004625	5.080660e+08	2020-12-20	134059
8	2021-02-04	0.037226	0.057869	0.035945	0.053289	0.053289	1.304084e+10	2021-02-04	135724
9	2021-02-04	0.037226	0.057869	0.035945	0.053289	0.053289	1.304084e+10	2021-02-04	135724
10	2021-02-04	0.037226	0.057869	0.035945	0.053289	0.053289	1.304084e+10	2021-02-04	135723
11	2021-02-06	0.046931	0.058308	0.044904	0.057595	0.057595	5.946101e+09	2021-02-06	135797
12	2021-02-07	0.057502	0.084357	0.054239	0.078782	0.078782	1.426102e+10	2021-02-07	135824
13	2021-02-07	0.057502	0.084357	0.054239	0.078782	0.078782	1.426102e+10	2021-02-07	135824
14	2021-02-08	0.078352	0.084945	0.064702	0.078825	0.078825	1.284438e+10	2021-02-08	135864

	Date	Open	High	Low	Close	Adj Close	Volume	date	
15	2021-02-08	0.078352	0.084945	0.064702	0.078825	0.078825	1.284438e+10	2021-02-08	135854
16	2021-02-10	0.070111	0.081091	0.068525	0.072896	0.072896	6.785088e+09	2021-02-10	135951
17	2021-02-11	0.072844	0.074301	0.068290	0.069676	0.069676	3.818557e+09	2021-02-11	135979
18	2021-02-12	0.069650	0.072610	0.061445	0.070069	0.070069	4.190844e+09	2021-02-12	136000
19	2021-02-15	0.062568	0.063924	0.048547	0.056591	0.056591	4.944805e+09	2021-02-15	136109
20	2021-02-15	0.062568	0.063924	0.048547	0.056591	0.056591	4.944805e+09	2021-02-15	136109
21	2021-02-20	0.055132	0.060286	0.051628	0.054384	0.054384	3.175469e+09	2021-02-20	136306
22	2021-02-20	0.055132	0.060286	0.051628	0.054384	0.054384	3.175469e+09	2021-02-20	136304
23	2021-02-21	0.054369	0.058428	0.053556	0.055980	0.055980	2.450293e+09	2021-02-21	136360
24	2021-03-01	0.048070	0.051479	0.048029	0.050599	0.050599	1.494427e+09	2021-03-01	136647
25	2021-03-02	0.050596	0.052382	0.049299	0.050262	0.050262	1.346282e+09	2021-03-02	136685
26	2021-03-02	0.050596	0.052382	0.049299	0.050262	0.050262	1.346282e+09	2021-03-02	136666
27	2021-03-06	0.049601	0.052397	0.049383	0.050984	0.050984	1.480482e+09	2021-03-06	136805
28	2021-03-14	0.062384	0.063052	0.058592	0.058592	0.058592	2.752133e+09	2021-03-14	137088
29	2021-03-14	0.062384	0.063052	0.058592	0.058592	0.058592	2.752133e+09	2021-03-14	137088
30	2021-03-14	0.062384	0.063052	0.058592	0.058592	0.058592	2.752133e+09	2021-03-14	137088
31	2021-03-16	0.057086	0.058921	0.055493	0.058607	0.058607	1.393772e+09	2021-03-16	137160

	Date	Open	High	Low	Close	Adj Close	Volume	date	
32	2021-03-18	0.057640	0.058828	0.054143	0.057383	0.057383	9.097774e+08	2021-03-18	137265
33	2021-03-25	0.051699	0.052407	0.049697	0.051448	0.051448	1.084214e+09	2021-03-25	137498
34	2021-04-01	0.053655	0.070111	0.053644	0.061986	0.061986	5.816047e+09	2021-04-01	137756
35	2021-04-08	0.059036	0.061745	0.058817	0.061464	0.061464	1.055258e+09	2021-04-08	138026
36	2021-04-15	0.121167	0.187326	0.120736	0.182207	0.182207	1.791662e+10	2021-04-15	138255
37	2021-04-15	0.121167	0.187326	0.120736	0.182207	0.182207	1.791662e+10	2021-04-15	13825
38	2021-04-15	0.121167	0.187326	0.120736	0.182207	0.182207	1.791662e+10	2021-04-15	138246
39	2021-04-15	0.121167	0.187326	0.120736	0.182207	0.182207	1.791662e+10	2021-04-15	138245
40	2021-04-16	0.181587	0.437700	0.180488	0.365870	0.365870	6.941068e+10	2021-04-16	13828(

41 rows × 43 columns

In [ ]: