Task

just upload these

Here is all the data you need: "BRNT Historical Data.csv" "DAX Historical Data.csv" "ITA Historical Data.csv" "MOEX Russia Index Historical Data.csv" "PPA Historical Data.csv"

Data loading

```
!pip install dtw
```

```
Requirement already satisfied: dtw in /usr/local/lib/python3.11/dist-packages (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from dtw) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from dtw) (1.15.3)
```

```
# Importing libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from dtw import dtw
import numpy as np

from scipy.spatial.distance import squareform
from scipy.cluster.hierarchy import linkage
from matplotlib.patches import Patch

# Load the CSV files into pandas DataFrames and parse the 'Date' column
df_moex = pd.read_csv('MOEX Russia Index Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_dax = pd.read_csv('DAX Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_spy = pd.read_csv('SPY Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_brnt = pd.read_csv('BRNT Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_usd_rub = pd.read_csv('USD_RUB Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_usd_uah = pd.read_csv('USD_UAH Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_ita = pd.read_csv('ITA Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_ppa = pd.read_csv('PPA Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_jets = pd.read_csv('JETS Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_rhmg = pd.read_csv('RHMG Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_mtxgn = pd.read_csv('MTXGn Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_air = pd.read_csv('AIR Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_tcfp = pd.read_csv('TCFP Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_baes = pd.read_csv('BAES Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_hiae = pd.read_csv('HIAE Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_praf = pd.read_csv('PRAF Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_bara = pd.read_csv('BARA Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_ieur = pd.read_csv('IEUR Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_sxepex = pd.read_csv('SXEPEX Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_fez = pd.read_csv('FEZ Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_xle = pd.read_csv('XLE Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_bp = pd.read_csv('BP Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_eni = pd.read_csv('ENI Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_eqnr = pd.read_csv('EQNR Historical Data.csv', parse_dates=['Date'], dayfirst=True)
df_shell = pd.read_csv('SHELl Historical Data.csv', parse_dates=['Date'], dayfirst=True)
```

```
df_moex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      25 non-null     datetime64[ns]
 1   Price     25 non-null     object
 2   Open      25 non-null     object
 3   High      25 non-null     object
 4   Low       25 non-null     object
 5   Vol.      0 non-null      float64
 6   Change %  25 non-null     object
dtypes: datetime64[ns](1), float64(1), object(5)
memory usage: 1.5+ KB
```

```
df_moex.head()
```

	Date	Price	Open	High	Low	Vol.	Change %
0	2022-03-25	2,484.13	2,620.62	2,664.74	2,470.69	NaN	-3.66%
1	2022-03-24	2,578.51	2,467.09	2,761.17	2,447.68	NaN	4.37%
2	2022-02-25	2,470.48	2,261.33	2,552.46	2,256.09	NaN	20.04%
3	2022-02-24	2,058.12	2,735.88	2,740.31	1,681.55	NaN	-33.28%
4	2022-02-22	3,084.74	2,909.40	3,113.40	2,756.46	NaN	1.58%

Next steps: Generate code with df_moex | View recommended plots | New interactive sheet

```
# Drop open, low, high columns from all the dataframes

dataframes = [df_moex, df_dax, df_spy, df_brnt, df_usd_rub, df_usd_uah, df_ita, df_ppa, df_jets, df_rhmg, df_mtxgn, df_air, df_t
columns_to_drop = ['Open', 'Low', 'High']

for df in dataframes:
  for col in columns_to_drop:
    if col in df.columns:
      df.drop(col, axis=1, inplace=True)
```

```
df_moex.head()
```

	Date	Price	Vol.	Change %
0	2022-03-25	2,484.13	NaN	-3.66%
1	2022-03-24	2,578.51	NaN	4.37%
2	2022-02-25	2,470.48	NaN	20.04%
3	2022-02-24	2,058.12	NaN	-33.28%
4	2022-02-22	3,084.74	NaN	1.58%

Next steps: Generate code with df_moex | View recommended plots | New interactive sheet

```
df_moex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      25 non-null     datetime64[ns]
 1   Price     25 non-null     object
 2   Vol.      0 non-null      float64
 3   Change %  25 non-null     object
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 932.0+ bytes
```

```
# Rename the price and volume columns in all dataframes to the price and volume of the index/equity they represent, such as pric

df_moex.rename(columns={'Price': 'Price_moex', 'Vol.': 'Vol_moex', 'Change %': 'Change_moex'}, inplace=True)
df_dax.rename(columns={'Price': 'Price_dax', 'Vol.': 'Vol_dax', 'Change %': 'Change_dax'}, inplace=True)
df_spy.rename(columns={'Price': 'Price_spy', 'Vol.': 'Vol_spy', 'Change %': 'Change_spy'}, inplace=True)
df_brnt.rename(columns={'Price': 'Price_brnt', 'Vol.': 'Vol_brnt', 'Change %': 'Change_brnt'}, inplace=True)
df_usd_rub.rename(columns={'Price': 'Price_usd_rub', 'Vol.': 'Vol_usd_rub', 'Change %': 'Change_usd_rub'}, inplace=True)
df_usd_uah.rename(columns={'Price': 'Price_usd_uah', 'Vol.': 'Vol_usd_uah', 'Change %': 'Change_usd_uah'}, inplace=True)
df_ita.rename(columns={'Price': 'Price_ita', 'Vol.': 'Vol_ita', 'Change %': 'Change_ita'}, inplace=True)
df_ppa.rename(columns={'Price': 'Price_ppa', 'Vol.': 'Vol_ppa', 'Change %': 'Change_ppa'}, inplace=True)
df_jets.rename(columns={'Price': 'Price_jets', 'Vol.': 'Vol_jets', 'Change %': 'Change_jets'}, inplace=True)
df_rhmg.rename(columns={'Price': 'Price_rhmg', 'Vol.': 'Vol_rhmg', 'Change %': 'Change_rhmg'}, inplace=True)
df_mtxgn.rename(columns={'Price': 'Price_mtxgn', 'Vol.': 'Vol_mtxgn', 'Change %': 'Change_mtxgn'}, inplace=True)
df_air.rename(columns={'Price': 'Price_air', 'Vol.': 'Vol_air', 'Change %': 'Change_air'}, inplace=True)
df_tcfp.rename(columns={'Price': 'Price_tcfp', 'Vol.': 'Vol_tcfp', 'Change %': 'Change_tcfp'}, inplace=True)
df_baes.rename(columns={'Price': 'Price_baes', 'Vol.': 'Vol_baes', 'Change %': 'Change_baes'}, inplace=True)
df_hiae.rename(columns={'Price': 'Price_hiae', 'Vol.': 'Vol_hiae', 'Change %': 'Change_hiae'}, inplace=True)
df_praf.rename(columns={'Price': 'Price_praf', 'Vol.': 'Vol_praf', 'Change %': 'Change_praf'}, inplace=True)
```

```
df_bara.rename(columns={'Price': 'Price_bara', 'Vol.': 'Vol_bara', 'Change %': 'Change_bara'}, inplace=True)
df_ieur.rename(columns={'Price': 'Price_ieur', 'Vol.': 'Vol_ieur', 'Change %': 'Change_ieur'}, inplace=True)
df_sxepex.rename(columns={'Price': 'Price_sxepex', 'Vol.': 'Vol_sxepex', 'Change %': 'Change_sxepex'}, inplace=True)
df_fez.rename(columns={'Price': 'Price_fez', 'Vol.': 'Vol_fez', 'Change %': 'Change_fez'}, inplace=True)
df_xle.rename(columns={'Price': 'Price_xle', 'Vol.': 'Vol_xle', 'Change %': 'Change_xle'}, inplace=True)
df_bp.rename(columns={'Price': 'Price_bp', 'Vol.': 'Vol_bp', 'Change %': 'Change_bp'}, inplace=True)
df_eni.rename(columns={'Price': 'Price_eni', 'Vol.': 'Vol_eni', 'Change %': 'Change_eni'}, inplace=True)
df_eqnr.rename(columns={'Price': 'Price_eqnr', 'Vol.': 'Vol_eqnr', 'Change %': 'Change_eqnr'}, inplace=True)
df_shell.rename(columns={'Price': 'Price_shell', 'Vol.': 'Vol_shell', 'Change %': 'Change_shell'}, inplace=True)
```

```
df_moex.head()
```

	Date	Price_moex	Vol_moex	Change_moex
0	2022-03-25	2,484.13	NaN	-3.66%
1	2022-03-24	2,578.51	NaN	4.37%
2	2022-02-25	2,470.48	NaN	20.04%
3	2022-02-24	2,058.12	NaN	-33.28%
4	2022-02-22	3,084.74	NaN	1.58%

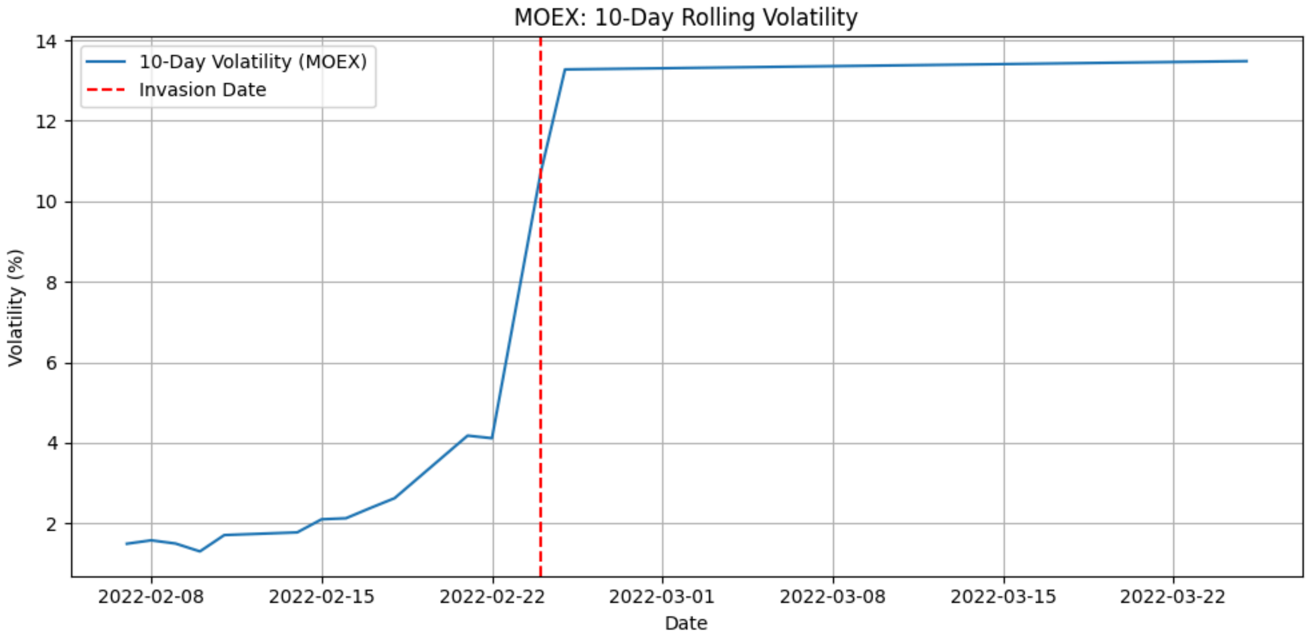
Next steps: Generate code with df_moex | View recommended plots | New interactive sheet

Process and Plot 10-Day Rolling Volatility for All Assets

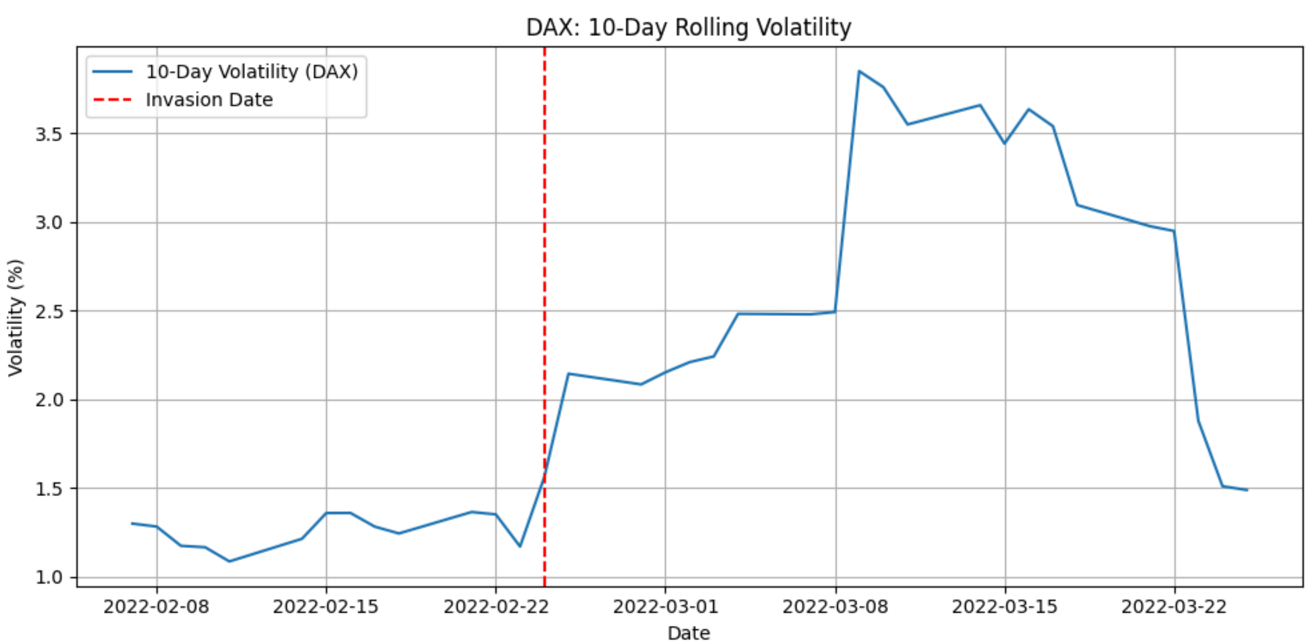
```
def process_volatility(df, asset_name):
    """
    Cleans the 'Change_<asset>' column, computes 10-day rolling volatility,
    and plots the result with invasion date marked.

    Parameters:
        df (pd.DataFrame): DataFrame with columns ['Date', f'Change_{asset_name}']
        asset_name (str): Asset identifier used in column names (e.g., 'brnt', 'dax')

    Returns:
        pd.DataFrame: Updated DataFrame with new 'Volatility_<asset>' column
    """
    change_col = f'Change_{asset_name}'
    vol_col = f'Volatility_{asset_name}'

    # Step 1: Clean and format date
    df['Date'] = pd.to_datetime(df['Date'])
    df = df.sort_values(by='Date')

    # Step 2: Clean Change column
    df[change_col] = (
        df[change_col]
        .astype(str)
        .str.replace('%', '', regex=False)
        .str.replace(',', '', regex=False)
        .str.strip()
    )

    # Step 3: Convert to float
    df[change_col] = pd.to_numeric(df[change_col], errors='coerce')

    # Step 4: Compute 10-day rolling volatility
    df[vol_col] = df[change_col].rolling(window=10).std()

    # Step 5: Display last few rows
    print(f"\n{asset_name.upper()} - Last few rows of volatility:")
    print(df[['Date', change_col, vol_col]].tail(10))

    # Step 6: Plot volatility
    plt.figure(figsize=(10, 5))
    plt.plot(df['Date'], df[vol_col], label=f'10-Day Volatility ({asset_name.upper()})')
    plt.axvline(pd.to_datetime("2022-02-24"), color='red', linestyle='--', label='Invasion Date')
    plt.title(f"{asset_name.upper()}: 10-Day Rolling Volatility")
    plt.xlabel("Date")
    plt.ylabel("Volatility (%)")
    plt.legend()
    plt.grid(True)
```

```
    plt.tight_layout()
    plt.show()

    return df

# 📊 Process each DataFrame
df_moex = process_volatility(df_moex, 'moex')
df_dax = process_volatility(df_dax, 'dax')
df_spy = process_volatility(df_spy, 'spy')
df_brnt = process_volatility(df_brnt, 'brnt')
df_usd_rub = process_volatility(df_usd_rub, 'usd_rub')
df_usd_uah = process_volatility(df_usd_uah, 'usd_uah')
df_ita = process_volatility(df_ita, 'ita')
df_ppa = process_volatility(df_ppa, 'ppa')
df_jets = process_volatility(df_jets, 'jets')
df_rhmg = process_volatility(df_rhmg, 'rhmg')
df_mtxgn = process_volatility(df_mtxgn, 'mtxgn')
df_air = process_volatility(df_air, 'air')
df_tcfp = process_volatility(df_tcfp, 'tcfp')
df_baes = process_volatility(df_baes, 'baes')
df_hiae = process_volatility(df_hiae, 'hiae')
df_praf = process_volatility(df_praf, 'praf')
df_bara = process_volatility(df_bara, 'bara')
df_ieur = process_volatility(df_ieur, 'ieur')
df_sxepex = process_volatility(df_sxepex, 'sxepex')
df_fez = process_volatility(df_fez, 'fez')
df_xle = process_volatility(df_xle, 'xle')
df_bp = process_volatility(df_bp, 'bp')
df_eni = process_volatility(df_eni, 'eni')
df_eqnr = process_volatility(df_eqnr, 'eqnr')
df_shell = process_volatility(df_shell, 'shell')
```

```
MOEX - Last few rows of volatility:
        Date  Change_moex  Volatility_moex
9 2022-02-15         3.42         2.099091
8 2022-02-16         1.28         2.124430
7 2022-02-17        -3.71         2.378627
6 2022-02-18        -3.35         2.623802
5 2022-02-21       -10.50         4.178169
4 2022-02-22         1.58         4.113110
3 2022-02-24       -33.28        10.706950
2 2022-02-25        20.04        13.276593
1 2022-03-24         4.37        13.475716
0 2022-03-25        -3.66        13.482617
```

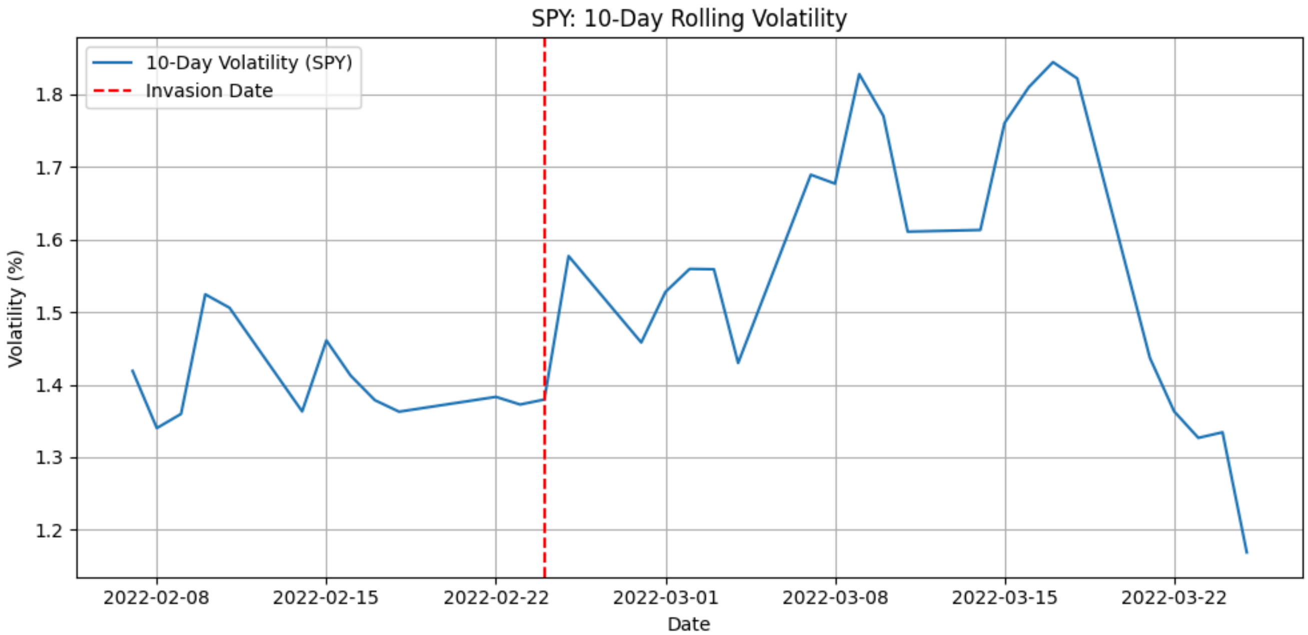



```
DAX - Last few rows of volatility:
        Date  Change_dax  Volatility_dax
9 2022-03-14        2.21        3.658182
8 2022-03-15       -0.09        3.441274
7 2022-03-16        3.76        3.635103
6 2022-03-17       -0.36        3.539318
5 2022-03-18        0.17        3.094878
4 2022-03-21       -0.60        2.975400
3 2022-03-22        1.02        2.948245
2 2022-03-23       -1.31        1.877879
1 2022-03-24       -0.07        1.508527
0 2022-03-25        0.22        1.487274
```

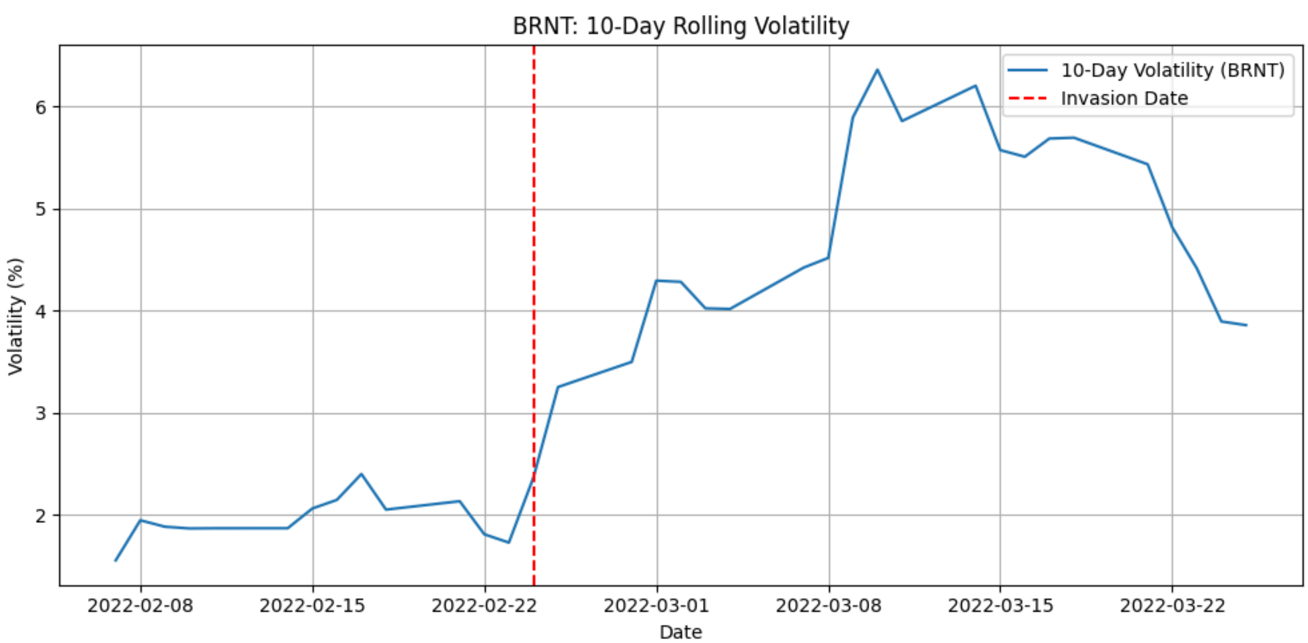



```
SPY - Last few rows of volatility:
```

```
        Date  Change_spy  Volatility_spy
9 2022-03-14       -0.73        1.612962
8 2022-03-15        2.20        1.760248
7 2022-03-16        2.22        1.809592
6 2022-03-17        1.25        1.844131
5 2022-03-18        0.78        1.821721
4 2022-03-21       -0.03        1.436790
3 2022-03-22        1.17        1.363189
2 2022-03-23       -1.29        1.326619
1 2022-03-24        1.51        1.334312
0 2022-03-25        0.49        1.168846
```

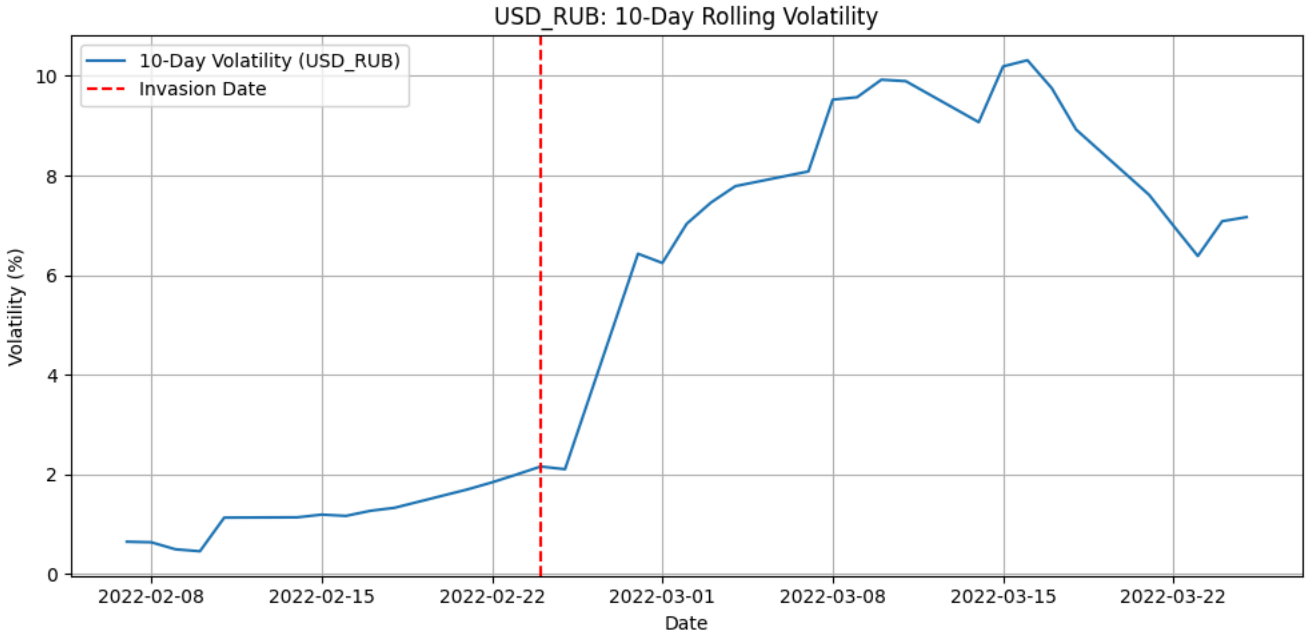



```
BRNT - Last few rows of volatility:
        Date  Change_brnt  Volatility_brnt
9 2022-03-14        -5.23         6.199866
8 2022-03-15        -2.97         5.569531
7 2022-03-16        -1.16         5.504586
6 2022-03-17         5.69         5.682848
5 2022-03-18         0.88         5.690773
4 2022-03-21         6.62         5.430490
3 2022-03-22        -0.98         4.813509
2 2022-03-23         4.91         4.411254
1 2022-03-24        -0.43         3.891604
0 2022-03-25         0.62         3.855902
```

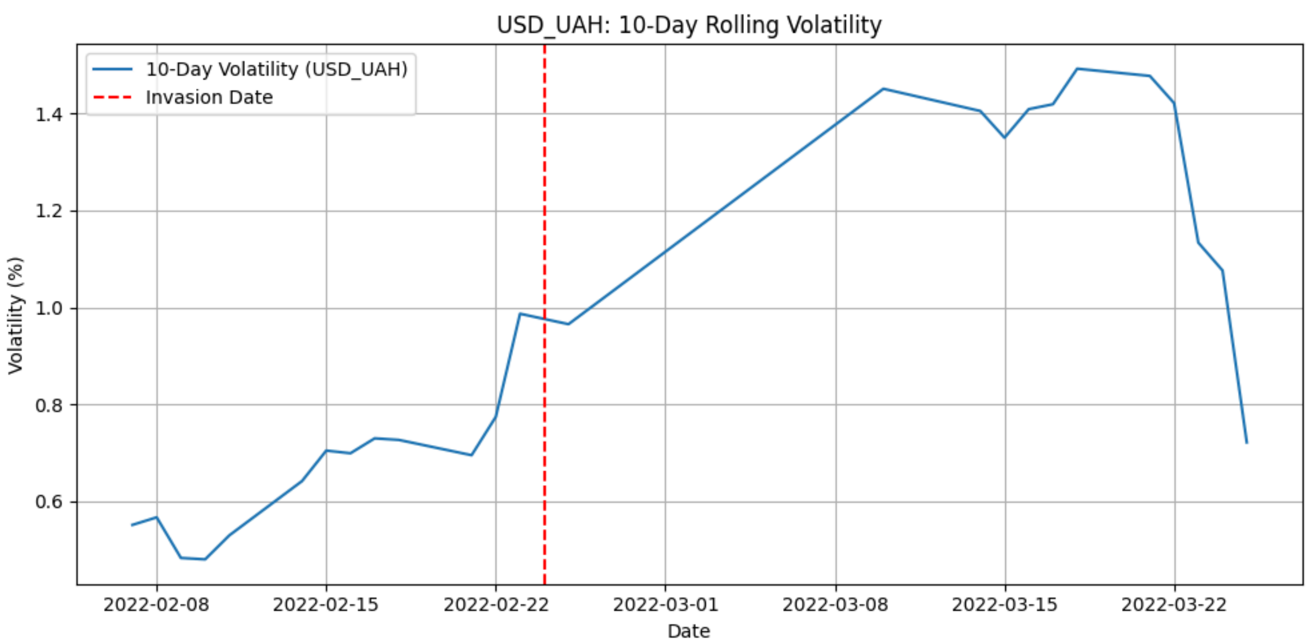



```
USD_RUB - Last few rows of volatility:
        Date  Change_usd_rub  Volatility_usd_rub
9 2022-03-14           -9.49            9.071087
```

```
8 2022-03-15          -13.54          10.189846
7 2022-03-16           -8.59          10.314066
6 2022-03-17            5.25           9.753057
5 2022-03-18            3.94           8.922543
4 2022-03-21            0.15           7.611430
3 2022-03-22           -0.61           6.991853
2 2022-03-23           -9.44           6.384447
1 2022-03-24            6.16           7.082611
0 2022-03-25            1.85           7.164697
```

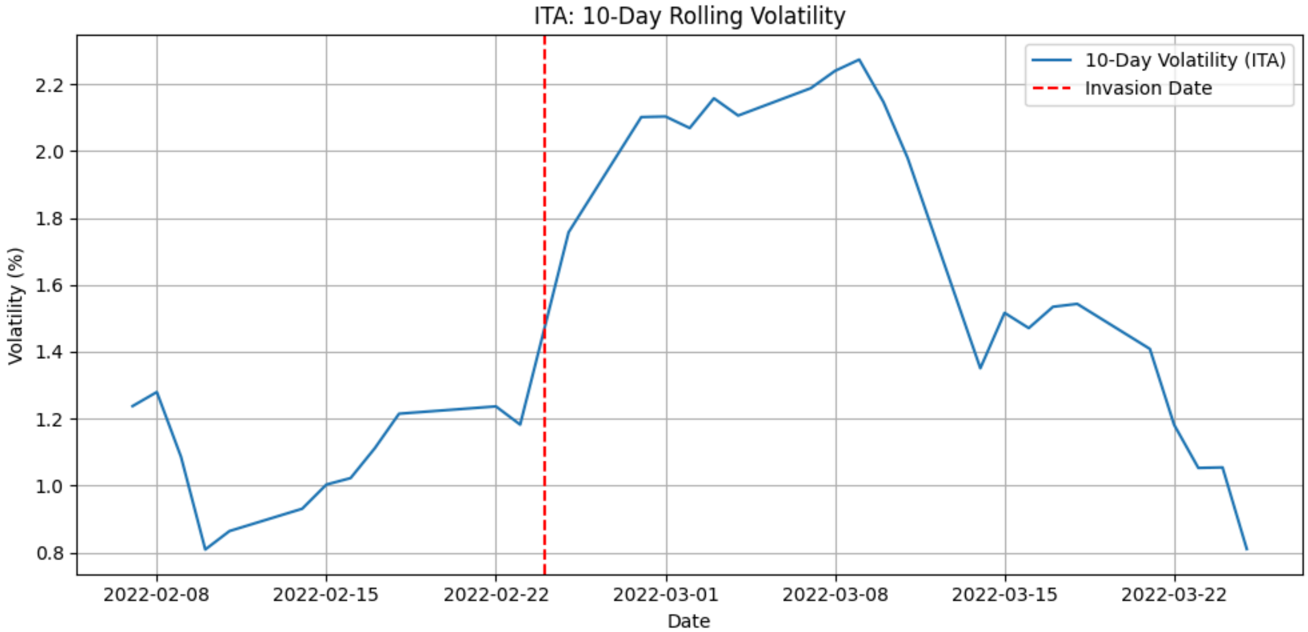



```
USD_UAH - Last few rows of volatility:
        Date  Change_usd_uah  Volatility_usd_uah
9 2022-03-14            0.00            1.404549
8 2022-03-15            1.02            1.349469
7 2022-03-16           -0.99            1.408342
6 2022-03-17            1.00            1.418467
5 2022-03-18           -1.01            1.491697
4 2022-03-21            0.00            1.476779
3 2022-03-22            0.00            1.420769
2 2022-03-23            0.00            1.133324
1 2022-03-24            0.00            1.075962
0 2022-03-25            0.85            0.721681
```

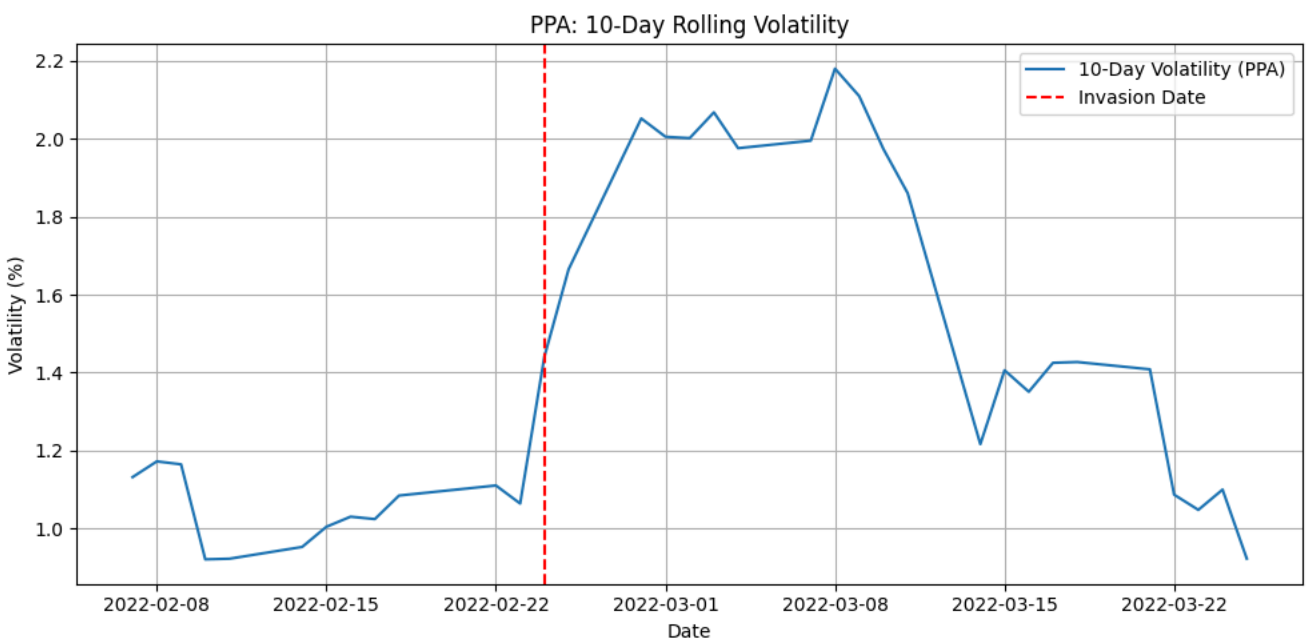



```
ITA - Last few rows of volatility:
        Date  Change_ita  Volatility_ita
9 2022-03-14       -0.46        1.350850
8 2022-03-15        1.70        1.516415
7 2022-03-16       -0.23        1.470852
6 2022-03-17        1.56        1.534617
```

```
5 2022-03-18        0.59        1.543265
4 2022-03-21        1.28        1.408580
3 2022-03-22        1.20        1.181882
2 2022-03-23       -0.39        1.052726
1 2022-03-24        0.96        1.054221
0 2022-03-25        0.35        0.810174
```

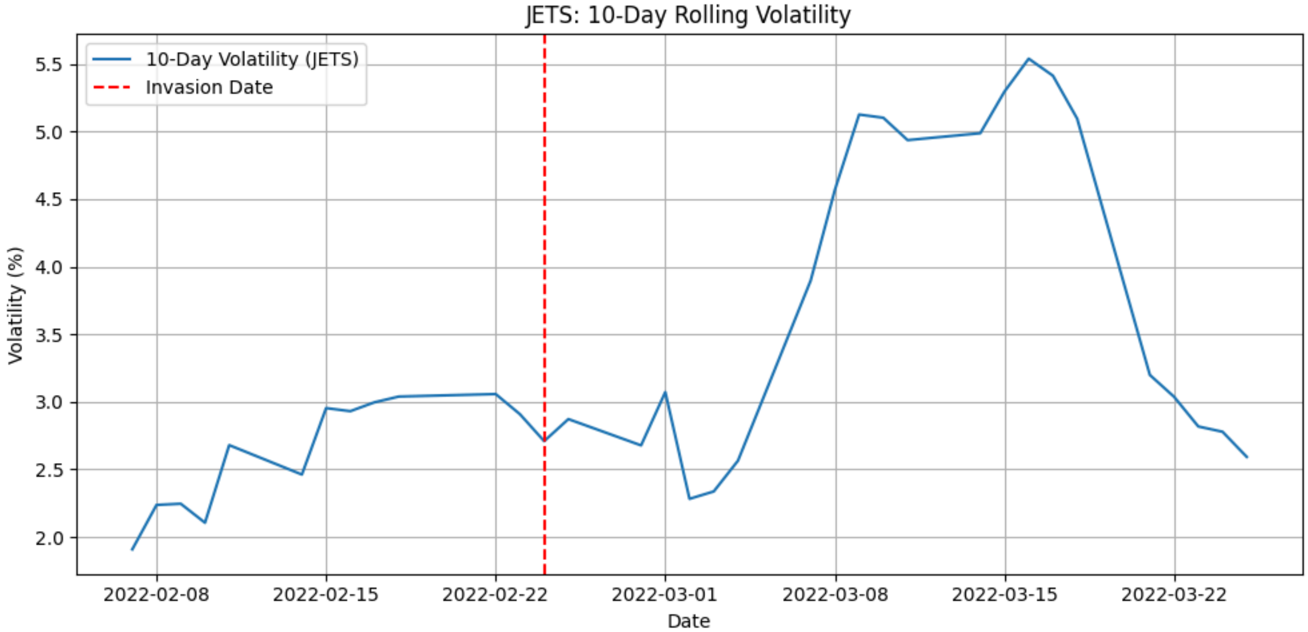



```
PPA - Last few rows of volatility:
        Date  Change_ppa  Volatility_ppa
9 2022-03-14       -0.48        1.216295
8 2022-03-15        1.90        1.405924
7 2022-03-16       -0.74        1.350827
6 2022-03-17        1.39        1.425155
5 2022-03-18        0.61        1.427165
4 2022-03-21        0.97        1.408429
3 2022-03-22        1.03        1.086716
2 2022-03-23       -0.55        1.047653
1 2022-03-24        1.43        1.099606
0 2022-03-25        0.38        0.922403
```

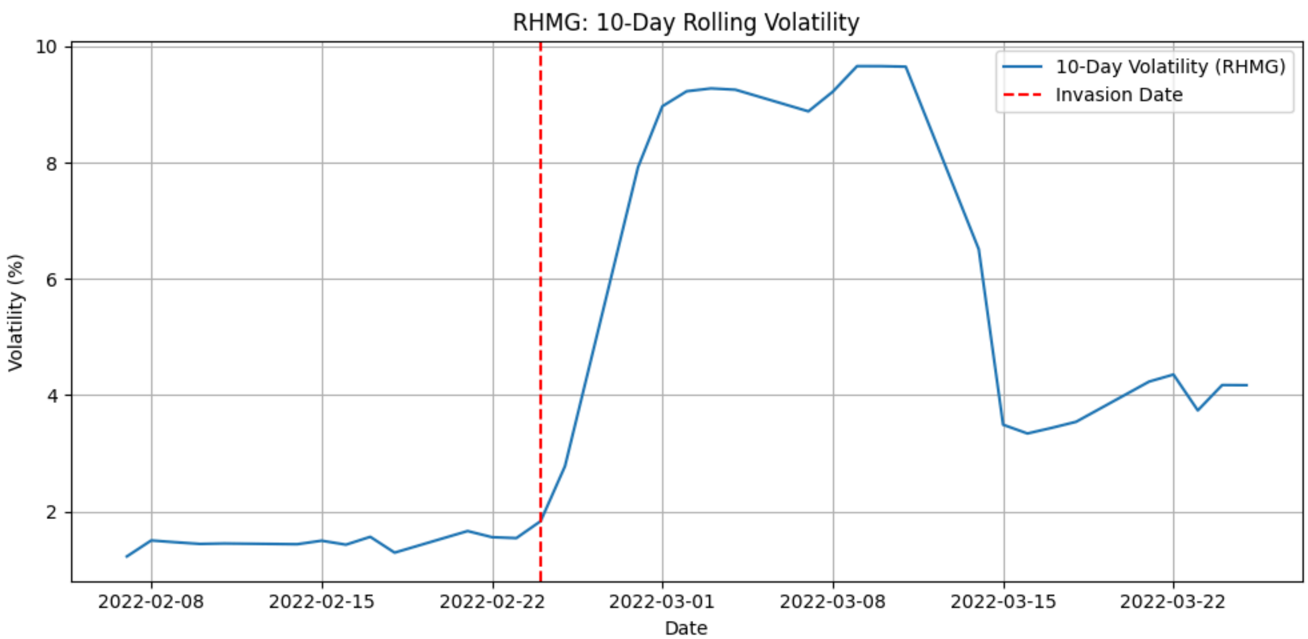



```
JETS - Last few rows of volatility:
        Date  Change_jets  Volatility_jets
9 2022-03-14         0.60         4.986802
8 2022-03-15         5.84         5.295107
7 2022-03-16         5.01         5.538325
6 2022-03-17        -0.20         5.412343
5 2022-03-18         1.48         5.094731
4 2022-03-21        -2.47         3.199019
```

```
3 2022-03-22        2.24         3.037495
2 2022-03-23       -1.41         2.817980
1 2022-03-24        2.12         2.778909
0 2022-03-25        0.72         2.592961
```

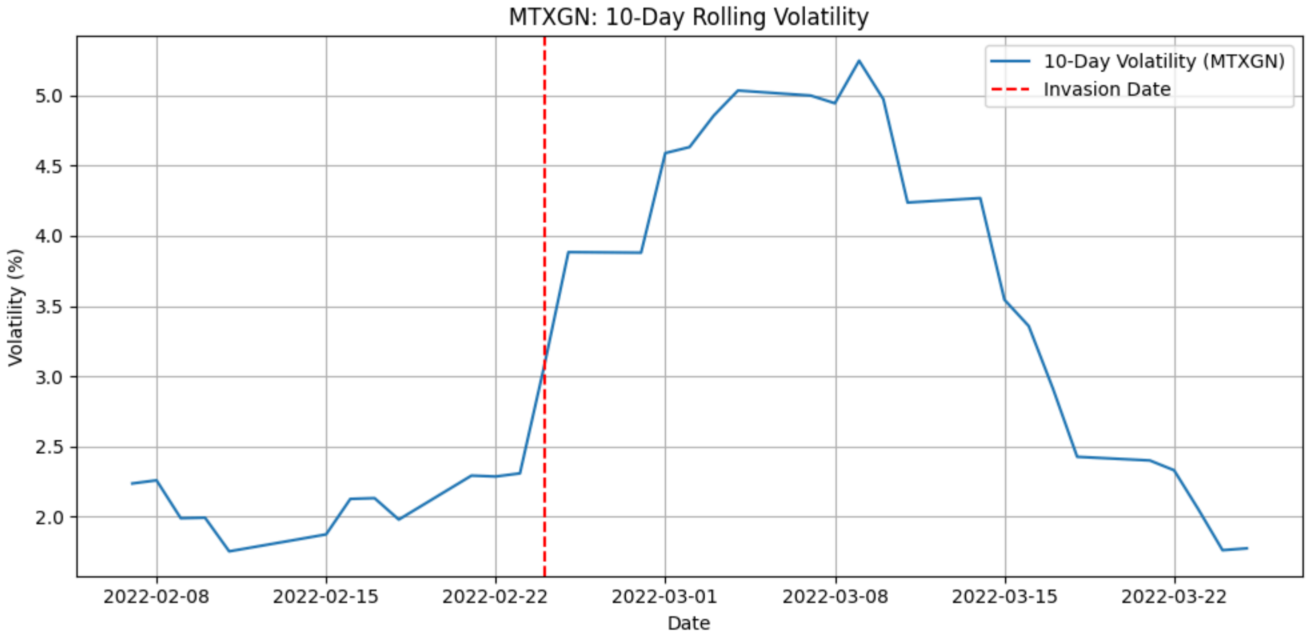



```
RHMG - Last few rows of volatility:
        Date  Change_rhmg  Volatility_rhmg
9 2022-03-14        -2.42         6.509424
8 2022-03-15         2.08         3.491443
7 2022-03-16         1.13         3.340904
6 2022-03-17         4.29         3.437710
5 2022-03-18         3.99         3.541943
4 2022-03-21         9.12         4.233703
3 2022-03-22        -3.70         4.354753
2 2022-03-23         1.04         3.737641
1 2022-03-24         8.89         4.174393
0 2022-03-25         3.09         4.171731
```

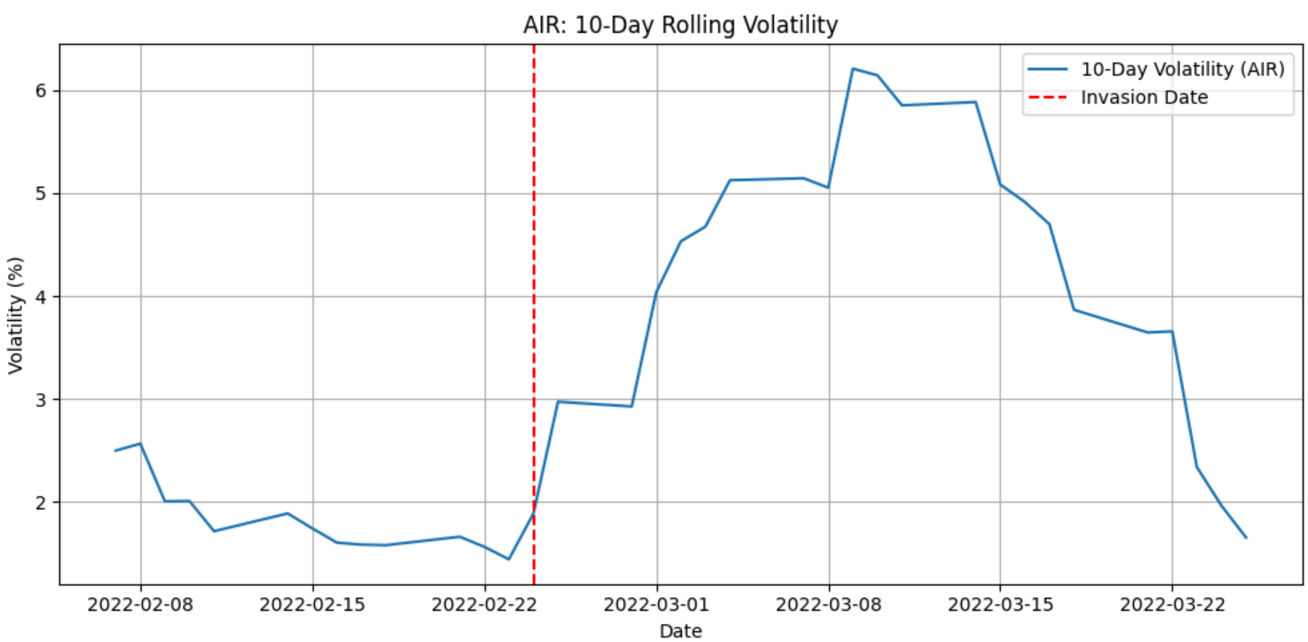



```
MTXGN - Last few rows of volatility:
        Date  Change_mtxgn  Volatility_mtxgn
9 2022-03-14          3.42          4.269038
8 2022-03-15          0.29          3.544909
7 2022-03-16          4.05          3.358267
6 2022-03-17          0.19          2.912835
5 2022-03-18         -1.27          2.426884
4 2022-03-21          1.33          2.400851
3 2022-03-22          0.75          2.331011
2 2022-03-23         -1.26          2.053735
```

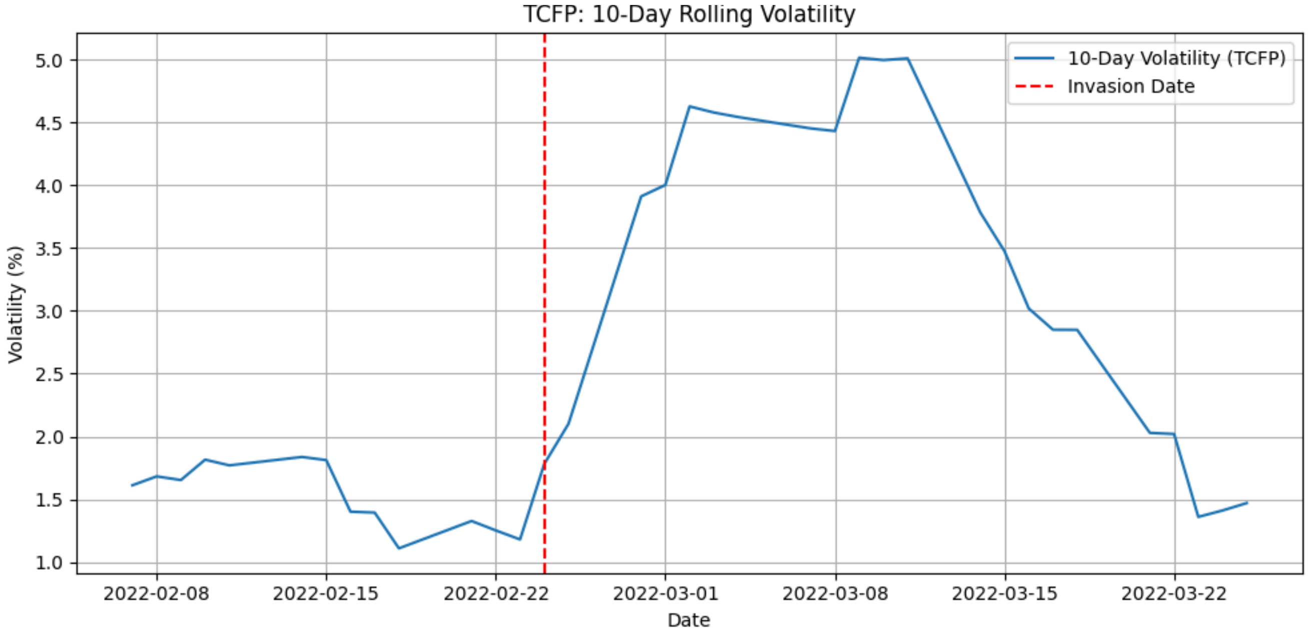
```
1 2022-03-24        -0.19          1.761964
0 2022-03-25         1.56          1.774918
```



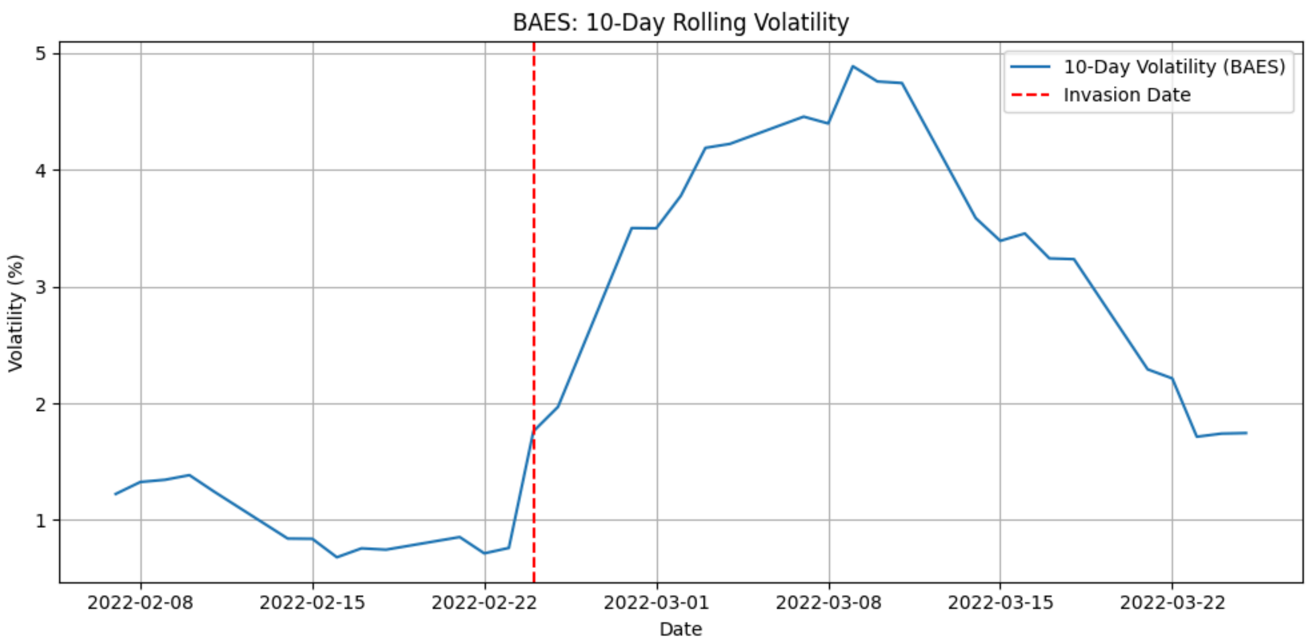

```
AIR - Last few rows of volatility:
        Date  Change_air  Volatility_air
9 2022-03-14        0.96        5.883300
8 2022-03-15       -1.38        5.082694
7 2022-03-16        3.59        4.913345
6 2022-03-17        0.39        4.698253
5 2022-03-18       -1.02        3.867194
4 2022-03-21       -0.77        3.646892
3 2022-03-22        2.79        3.657413
2 2022-03-23       -0.95        2.341012
1 2022-03-24        0.15        1.966719
0 2022-03-25        0.13        1.657049
```



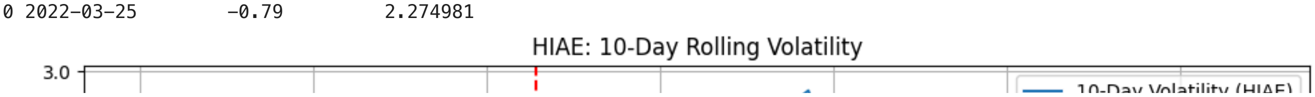

```
TCFP - Last few rows of volatility:
        Date  Change_tcfp  Volatility_tcfp
9 2022-03-14        -2.14         3.783189
8 2022-03-15         1.64         3.474811
7 2022-03-16        -1.17         3.018929
6 2022-03-17         1.68         2.849748
5 2022-03-18         0.27         2.848725
4 2022-03-21         2.45         2.028705
3 2022-03-22         0.52         2.019814
2 2022-03-23         0.30         1.359778
1 2022-03-24         1.72         1.410943
0 2022-03-25         1.95         1.469859
```

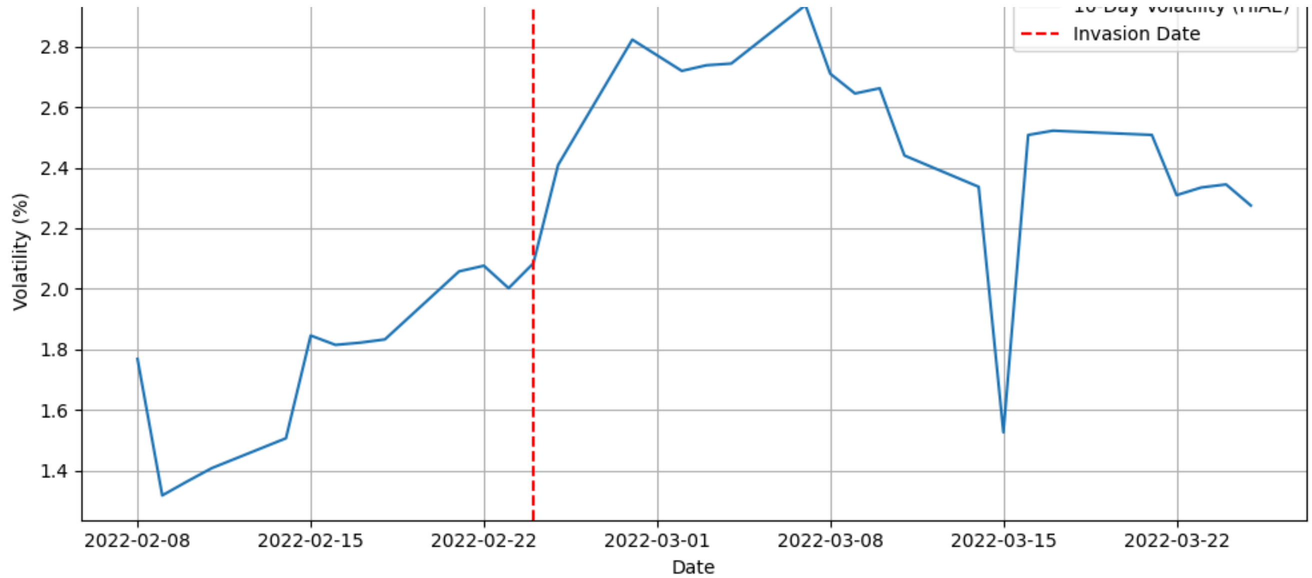



```
BAES – Last few rows of volatility:
        Date  Change_baes  Volatility_baes
9 2022-03-14        -1.28         3.585573
8 2022-03-15         1.22         3.390360
7 2022-03-16        -3.60         3.453253
6 2022-03-17         2.46         3.240045
5 2022-03-18        -0.08         3.233636
4 2022-03-21         1.16         2.289877
3 2022-03-22         0.77         2.212589
2 2022-03-23        -0.24         1.713108
1 2022-03-24         1.88         1.739962
0 2022-03-25         0.83         1.744539
```

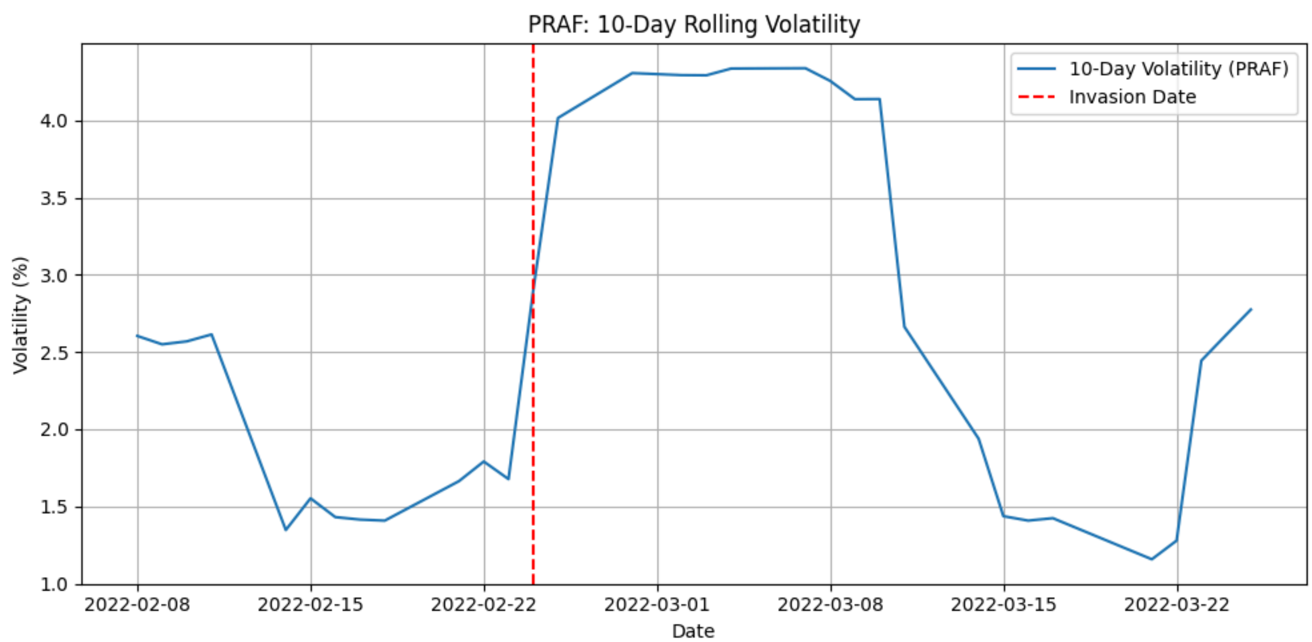



```
HIAE – Last few rows of volatility:
        Date  Change_hiae  Volatility_hiae
9 2022-03-11         1.24         2.439900
8 2022-03-14        -0.28         2.336590
7 2022-03-15        -0.32         1.526495
6 2022-03-16         6.20         2.507610
5 2022-03-17        -1.12         2.521866
4 2022-03-21         0.03         2.507811
3 2022-03-22        -2.17         2.309129
2 2022-03-23        -0.54         2.334442
1 2022-03-24        -0.09         2.344513
0 2022-03-25        -0.79         2.274981
```

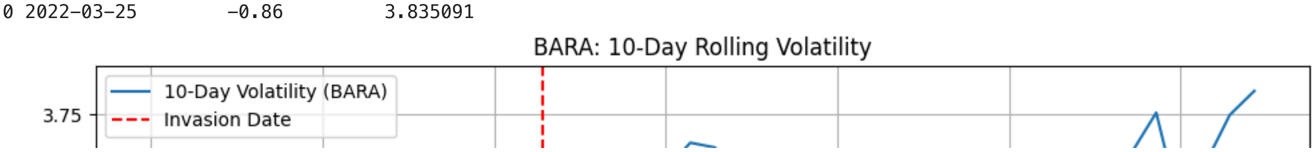


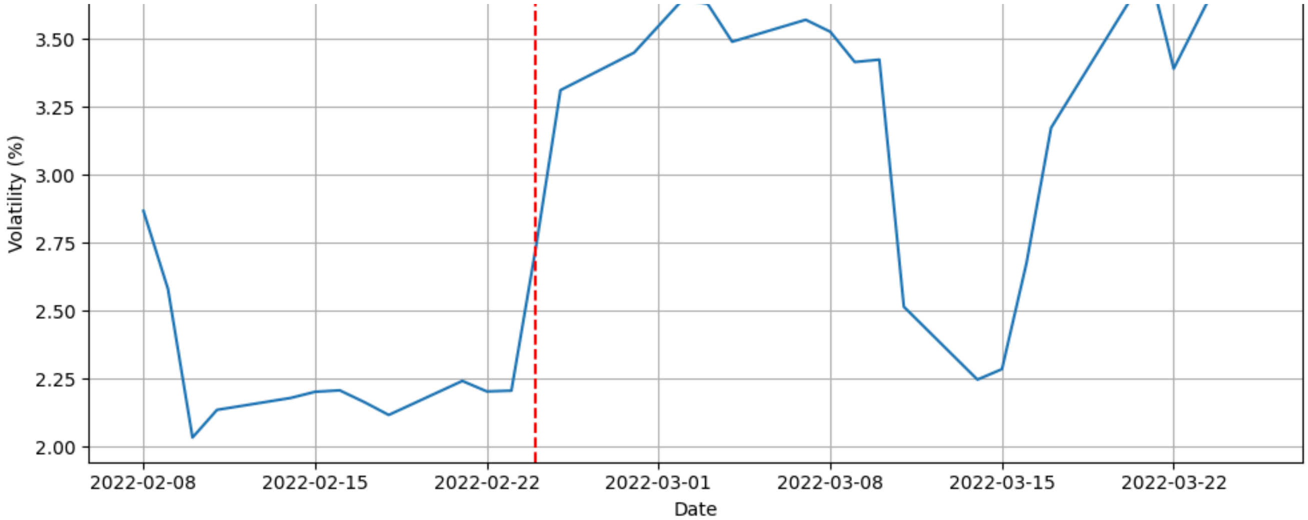



```
PRAF – Last few rows of volatility:
        Date  Change_praf  Volatility_praf
9 2022-03-11         1.42         2.663200
8 2022-03-14        -1.11         1.938449
7 2022-03-15        -1.12         1.436932
6 2022-03-16         1.10         1.408301
5 2022-03-17         0.75         1.423871
4 2022-03-21         0.39         1.157756
3 2022-03-22        -1.76         1.278517
2 2022-03-23         6.95         2.444158
1 2022-03-24        -2.32         2.610407
0 2022-03-25        -2.53         2.774840
```

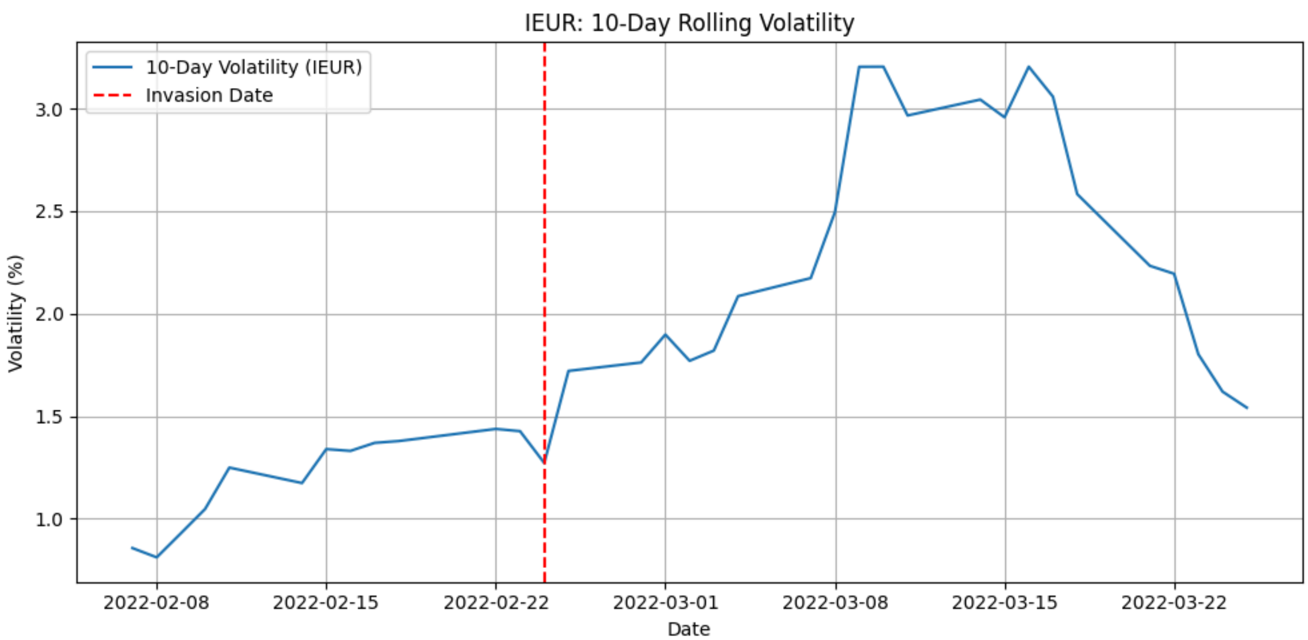



```
BARA – Last few rows of volatility:
        Date  Change_bara  Volatility_bara
9 2022-03-11         3.61         2.514656
8 2022-03-14         1.90         2.246742
7 2022-03-15         4.10         2.285293
6 2022-03-16         6.24         2.675823
5 2022-03-17         8.00         3.172412
4 2022-03-21        -4.56         3.755786
3 2022-03-22         0.56         3.390005
2 2022-03-23        -1.00         3.568433
1 2022-03-24        -1.18         3.747611
0 2022-03-25        -0.86         3.835091
```

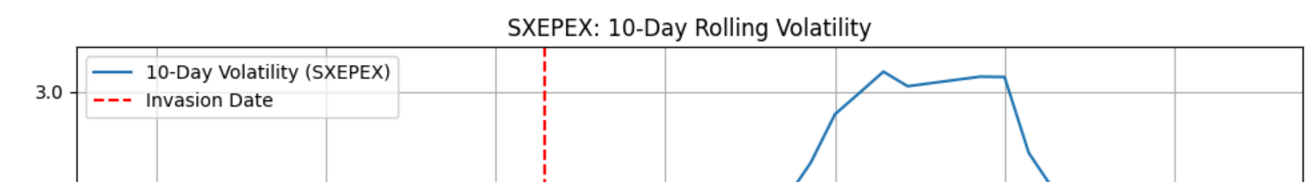


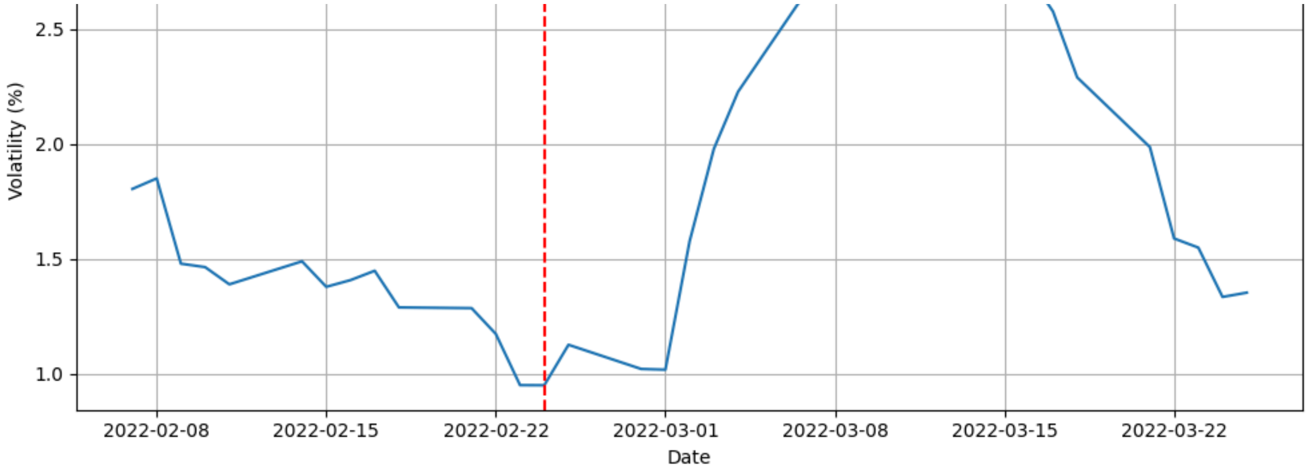



```
IEUR - Last few rows of volatility:
        Date  Change_ieur  Volatility_ieur
9 2022-03-14         1.88         3.043680
8 2022-03-15         0.65         2.957350
7 2022-03-16         3.92         3.204049
6 2022-03-17         0.63         3.057696
5 2022-03-18         0.90         2.582843
4 2022-03-21        -1.01         2.233230
3 2022-03-22         1.26         2.194484
2 2022-03-23        -1.75         1.802065
1 2022-03-24         0.57         1.619808
0 2022-03-25         0.13         1.541535
```

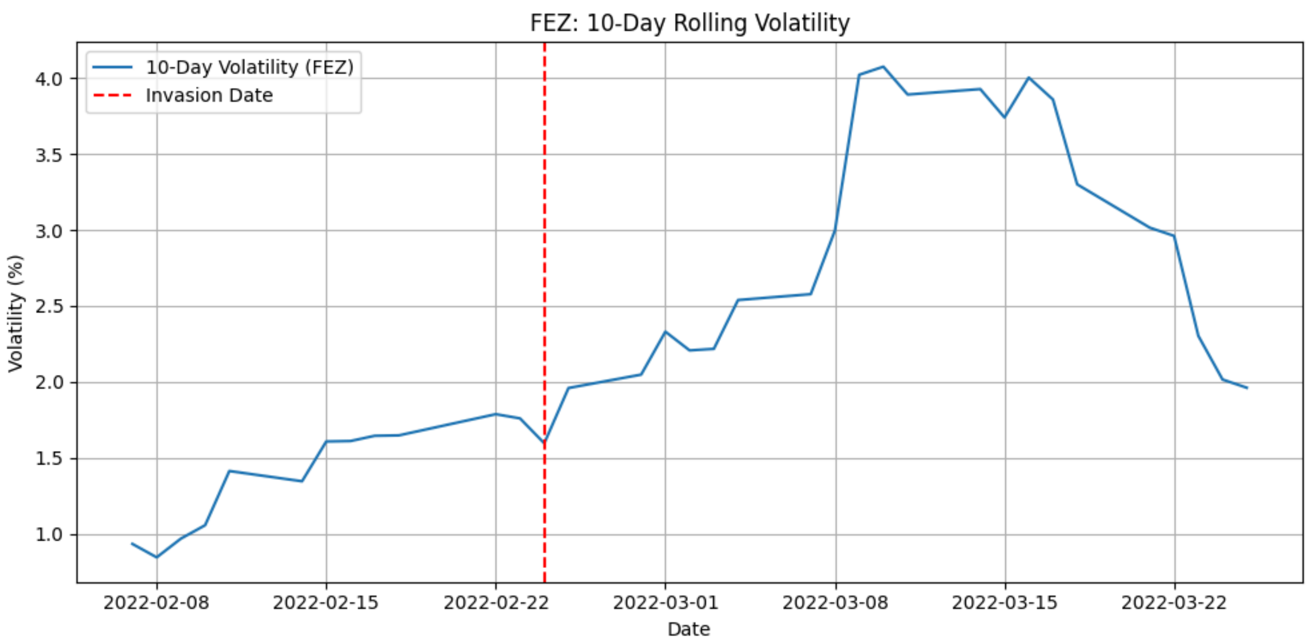



```
SXEPEX - Last few rows of volatility:
        Date  Change_sxepex  Volatility_sxepex
9 2022-03-14          -1.57           3.067273
8 2022-03-15           0.28           3.065724
7 2022-03-16          -0.51           2.735016
6 2022-03-17           2.06           2.578062
5 2022-03-18          -0.57           2.290866
4 2022-03-21           2.68           1.987722
3 2022-03-22           0.14           1.589712
2 2022-03-23           1.85           1.549926
1 2022-03-24          -0.05           1.335184
0 2022-03-25           1.27           1.353931
```

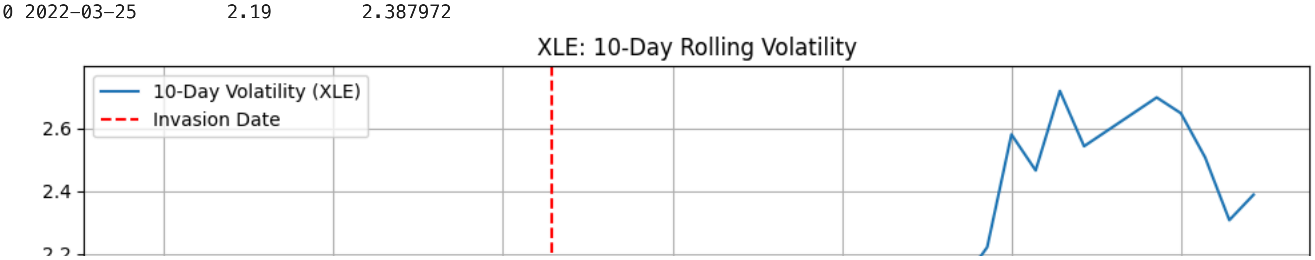


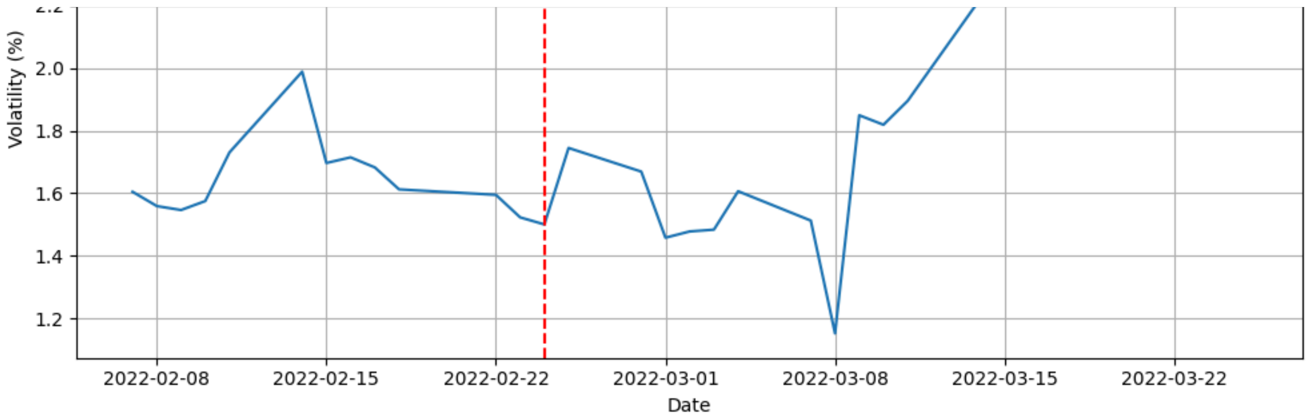



```
FEZ - Last few rows of volatility:
        Date  Change_fez  Volatility_fez
9 2022-03-14        2.14        3.927024
8 2022-03-15        0.97        3.740333
7 2022-03-16        4.78        4.003064
6 2022-03-17        0.12        3.858396
5 2022-03-18        0.34        3.300030
4 2022-03-21       -1.61        3.014635
3 2022-03-22        1.46        2.960815
2 2022-03-23       -2.31        2.303071
1 2022-03-24        0.89        2.015408
0 2022-03-25        0.10        1.961501
```

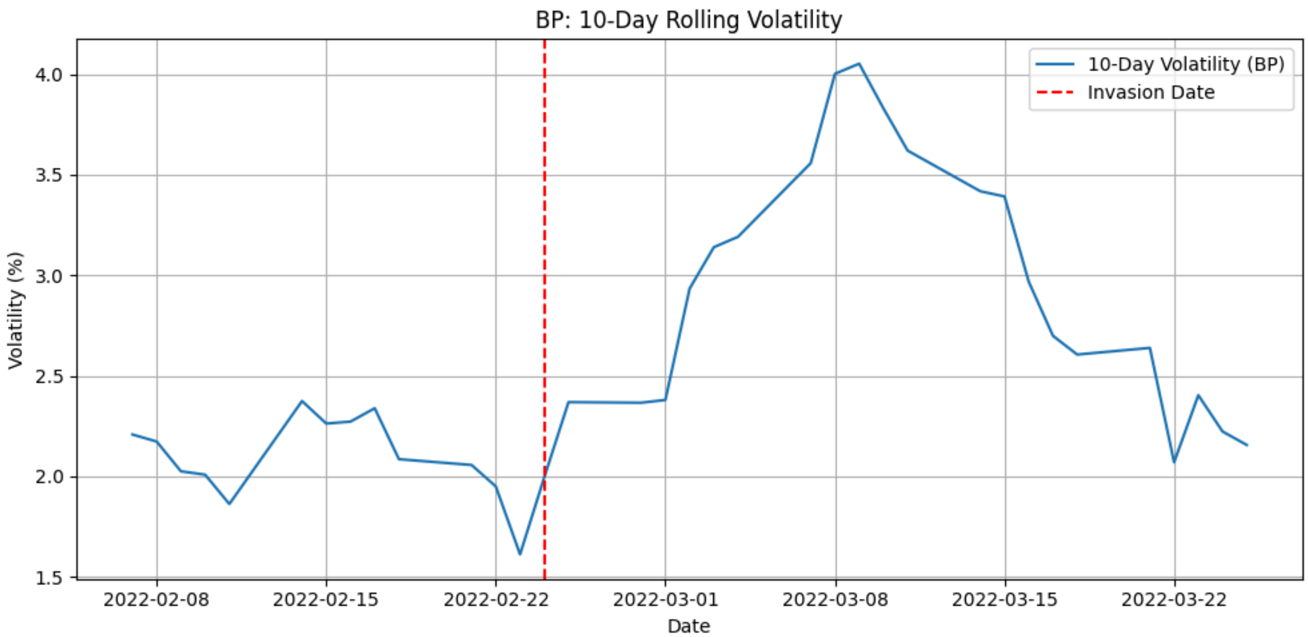



```
XLE - Last few rows of volatility:
        Date  Change_xle  Volatility_xle
9 2022-03-14       -2.99        2.220610
8 2022-03-15       -3.66        2.580203
7 2022-03-16       -0.46        2.465446
6 2022-03-17        3.44        2.719056
5 2022-03-18       -0.09        2.542603
4 2022-03-21        3.05        2.698214
3 2022-03-22       -0.74        2.647851
2 2022-03-23        1.72        2.506902
1 2022-03-24        0.25        2.306805
0 2022-03-25        2.19        2.387972
```

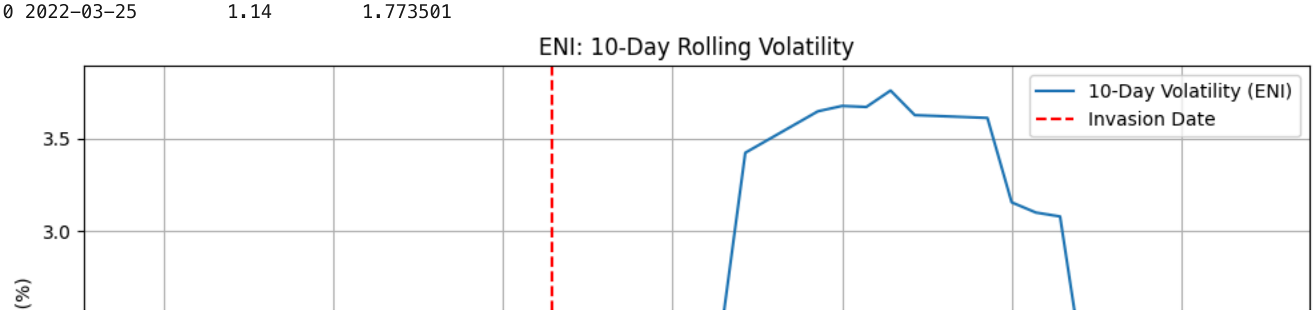


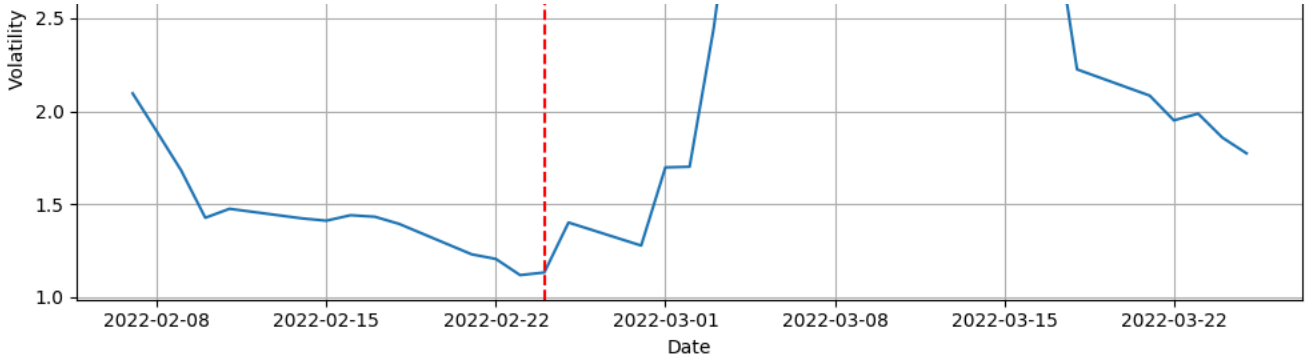



```
BP – Last few rows of volatility:
        Date  Change_bp  Volatility_bp
9 2022-03-14      -1.08       3.418057
8 2022-03-15       1.26       3.392004
7 2022-03-16      -0.10       2.967903
6 2022-03-17       2.05       2.699062
5 2022-03-18      -2.08       2.606024
4 2022-03-21       4.05       2.639140
3 2022-03-22      -0.99       2.070470
2 2022-03-23       4.47       2.403896
1 2022-03-24       0.34       2.223511
0 2022-03-25       0.64       2.156495
```

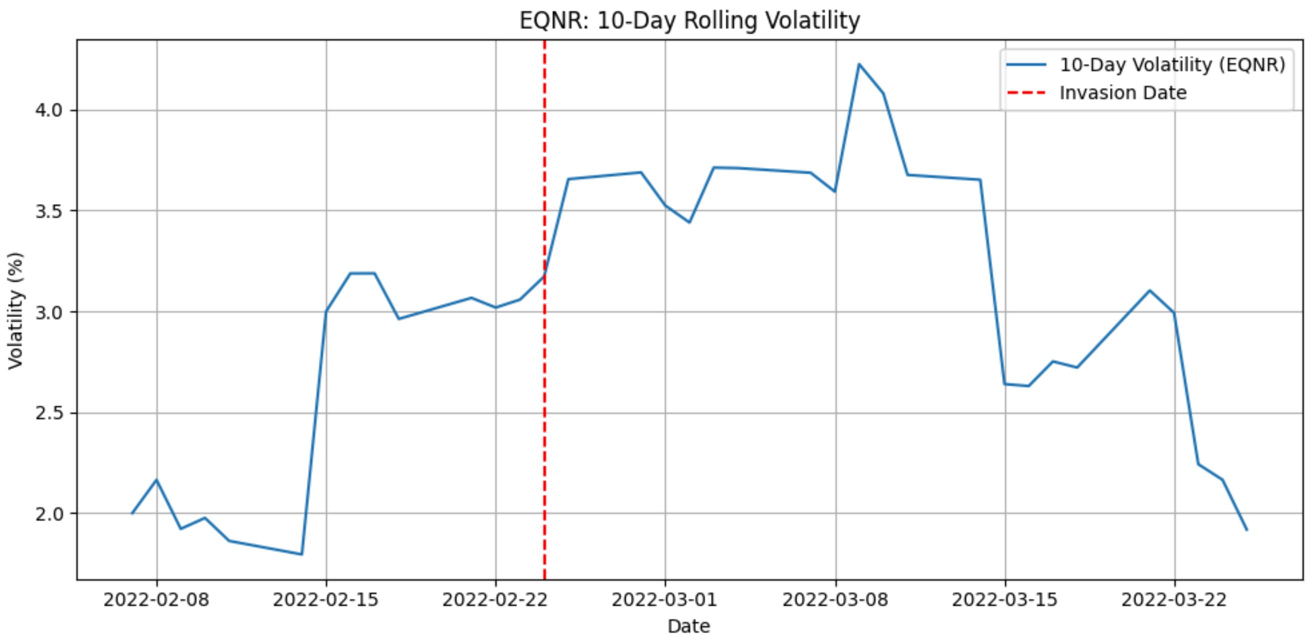



```
ENI – Last few rows of volatility:
        Date  Change_eni  Volatility_eni
9 2022-03-14       -0.54        3.609872
8 2022-03-15       -0.42        3.155320
7 2022-03-16       -1.19        3.100349
6 2022-03-17        2.73        3.079118
5 2022-03-18       -2.97        2.225070
4 2022-03-21        2.94        2.084089
3 2022-03-22        0.05        1.950972
2 2022-03-23        0.85        1.986958
1 2022-03-24        0.79        1.858373
0 2022-03-25        1.14        1.773501
```

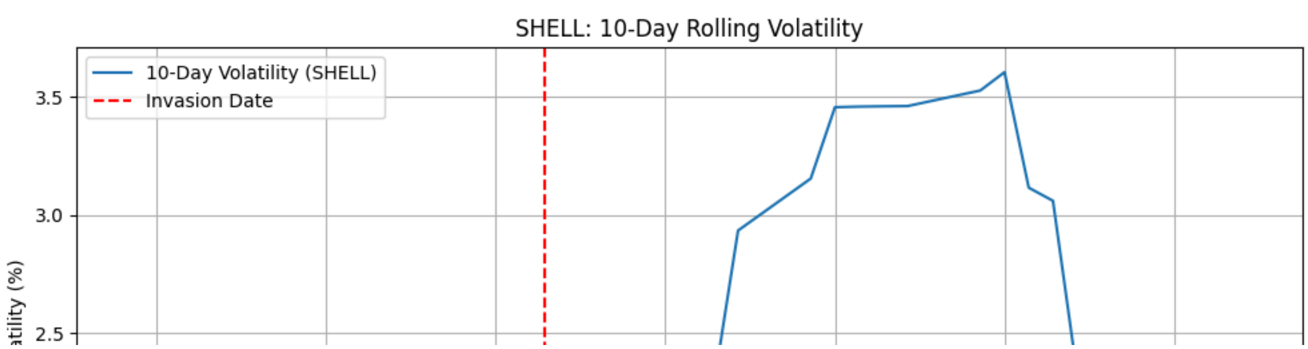


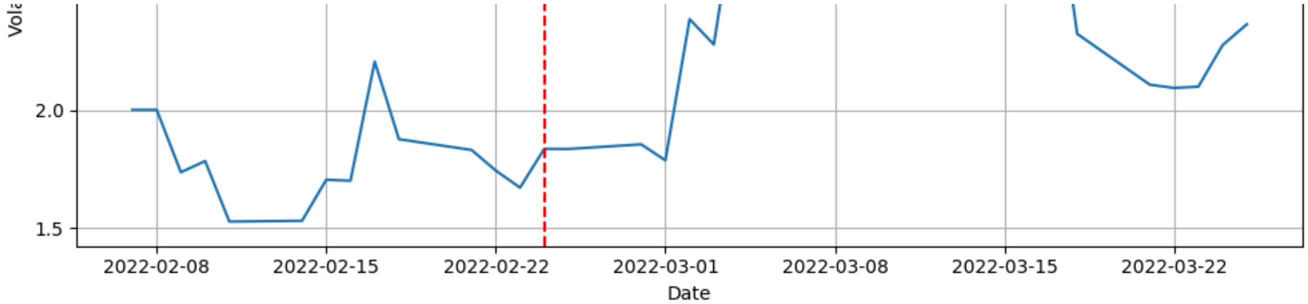



```
EQNR - Last few rows of volatility:
        Date  Change_eqnr  Volatility_eqnr
9 2022-03-14         0.26         3.652427
8 2022-03-15        -0.96         2.638880
7 2022-03-16         1.00         2.629012
6 2022-03-17         3.62         2.751195
5 2022-03-18         0.97         2.720772
4 2022-03-21         5.20         3.103089
3 2022-03-22        -0.32         2.992065
2 2022-03-23         1.87         2.241354
1 2022-03-24        -0.45         2.163871
0 2022-03-25         1.59         1.916929
```




```
SHELL - Last few rows of volatility:
        Date  Change_shell  Volatility_shell
9 2022-03-14         -2.34          3.527111
8 2022-03-15         -2.48          3.604700
7 2022-03-16          0.55          3.115819
6 2022-03-17          2.07          3.060215
5 2022-03-18          0.82          2.324265
4 2022-03-21         -0.44          2.108543
3 2022-03-22          4.37          2.094894
2 2022-03-23         -0.44          2.100279
1 2022-03-24          3.74          2.276097
0 2022-03-25         -1.40          2.364958
```




Vol
2.0
1.5
2022-02-08
2022-02-15
2022-02-22
2022-03-01
2022-03-08
2022-03-15
2022-03-22
Date

```
df_moex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 25 entries, 24 to 0
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             25 non-null     datetime64[ns]
 1   Price_moex       25 non-null     object
 2   Vol_moex         0 non-null      float64
 3   Change_moex      25 non-null     float64
 4   Volatility_moex  16 non-null     float64
dtypes: datetime64[ns](1), float64(3), object(1)
memory usage: 1.2+ KB
```

```
# prompt: create a master dataframe that has the date column, and prices and volumes of all the equities/indices...

# Merge the dataframes on the 'Date' column
master_df = df_moex.merge(df_dax, on='Date', how='outer')
master_df = master_df.merge(df_spy, on='Date', how='outer')
master_df = master_df.merge(df_brnt, on='Date', how='outer')
master_df = master_df.merge(df_usd_rub, on='Date', how='outer')
master_df = master_df.merge(df_usd_uah, on='Date', how='outer')
master_df = master_df.merge(df_ita, on='Date', how='outer')
master_df = master_df.merge(df_ppa, on='Date', how='outer')
master_df = master_df.merge(df_jets, on='Date', how='outer')
master_df = master_df.merge(df_rhmg, on='Date', how='outer')
master_df = master_df.merge(df_mtxgn, on='Date', how='outer')
master_df = master_df.merge(df_air, on='Date', how='outer')
master_df = master_df.merge(df_tcfp, on='Date', how='outer')
master_df = master_df.merge(df_baes, on='Date', how='outer')
master_df = master_df.merge(df_hiae, on='Date', how='outer')
master_df = master_df.merge(df_praf, on='Date', how='outer')
master_df = master_df.merge(df_bara, on='Date', how='outer')
master_df = master_df.merge(df_ieur, on='Date', how='outer')
master_df = master_df.merge(df_sxepex, on='Date', how='outer')
master_df = master_df.merge(df_fez, on='Date', how='outer')
master_df = master_df.merge(df_xle, on='Date', how='outer')
master_df = master_df.merge(df_bp, on='Date', how='outer')
master_df = master_df.merge(df_eni, on='Date', how='outer')
master_df = master_df.merge(df_eqnr, on='Date', how='outer')
master_df = master_df.merge(df_shell, on='Date', how='outer')

# Sort the dataframe by date
master_df = master_df.sort_values(by='Date')

# Post-invasion flag
invasion_date = pd.to_datetime("2022-02-24")
master_df['Post_Invasion'] = (master_df['Date'] >= invasion_date).astype(int)

# Display the first few rows of the master dataframe
display(master_df.head(2), master_df.tail(2))
```

	Date	Price_moex	Vol_moex	Change_moex	Volatility_moex	Price_dax	Vol_dax	Change_dax	Volatility_dax	Price_spy	...	Vo
0	2022-01-25	3,258.74	NaN	0.73	NaN	15,123.87	86.07M	0.75	NaN	434.47	...	
1	2022-01-26	3,357.66	NaN	3.04	NaN	15,459.39	82.97M	2.22	NaN	433.38	...	

2 rows × 102 columns

	Date	Price_moex	Vol_moex	Change_moex	Volatility_moex	Price_dax	Vol_dax	Change_dax	Volatility_dax	Price_spy	...	V
42	2022-03-24	2,578.51	NaN	4.37	13.475716	14,273.79	77.32M	-0.07	1.508527	450.49	...	
43	2022-03-25	2,484.13	NaN	-3.66	13.482617	14,305.76	73.04M	0.22	1.487274	452.69	...	

2 rows × 102 columns

```
master_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44 entries, 0 to 43
Columns: 102 entries, Date to Post_Invasion
```

```
dtypes: datetime64[ns](1), float64(74), int64(1), object(26)
memory usage: 35.2+ KB
```

```
# Convert all Price_* columns to float

# Clean and convert 'Price_moex' and 'Price_dax' to numeric, handling potential errors
df_moex['Price_moex'] = pd.to_numeric(df_moex['Price_moex'].astype(str).str.replace(',', '', regex=False), errors='coerce').asty
df_dax['Price_dax'] = pd.to_numeric(df_dax['Price_dax'].astype(str).str.replace(',', '', regex=False), errors='coerce').astype('

# Verify the data types
print("\nData types after conversion:")
print(df_moex[['Date', 'Price_moex']].info())
print(df_dax[['Date', 'Price_dax']].info())

# Drop non-informative volume columns

master_df.drop(columns=['Vol_moex', 'Vol_usd_rub'], inplace=True)

master_df.info()
```

```
 44  Vol_air            44 non-null     object
 45  Change_air         44 non-null     float64
 46  Volatility_air     35 non-null     float64
 47  Price_tcfp         44 non-null     float64
 48  Vol_tcfp           44 non-null     object
 49  Change_tcfp        44 non-null     float64
 50  Volatility_tcfp    35 non-null     float64
 51  Price_baes         44 non-null     float64
 52  Vol_baes           44 non-null     object
 53  Change_baes        44 non-null     float64
 54  Volatility_baes    35 non-null     float64
 55  Price_hiae         41 non-null     float64
 56  Vol_hiae           41 non-null     object
 57  Change_hiae        41 non-null     float64
 58  Volatility_hiae    32 non-null     float64
 59  Price_praf         41 non-null     float64
 60  Vol_praf           41 non-null     object
 61  Change_praf        41 non-null     float64
 62  Volatility_praf    32 non-null     float64
 63  Price_bara         41 non-null     float64
 64  Vol_bara           41 non-null     object
 65  Change_bara        41 non-null     float64
 66  Volatility_bara    32 non-null     float64
 67  Price_ieur         43 non-null     float64
 68  Vol_ieur           43 non-null     object
 69  Change_ieur        43 non-null     float64
 70  Volatility_ieur    34 non-null     float64
 71  Price_sxepex       44 non-null     float64
 72  Vol_sxepex         44 non-null     object
 73  Change_sxepex      44 non-null     float64
 74  Volatility_sxepex  35 non-null     float64
 75  Price_fez          43 non-null     float64
 76  Vol_fez            43 non-null     object
 77  Change_fez         43 non-null     float64
 78  Volatility_fez     34 non-null     float64
 79  Price_xle          43 non-null     float64
 80  Vol_xle            43 non-null     object
 81  Change_xle         43 non-null     float64
 82  Volatility_xle     34 non-null     float64
 83  Price_bp           44 non-null     float64
 84  Vol_bp             44 non-null     object
 85  Change_bp          44 non-null     float64
 86  Volatility_bp      35 non-null     float64
 87  Price_eni          44 non-null     float64
 88  Vol_eni            44 non-null     object
 89  Change_eni         44 non-null     float64
 90  Volatility_eni     35 non-null     float64
 91  Price_eqnr         44 non-null     float64
 92  Vol_eqnr           44 non-null     object
 93  Change_eqnr        44 non-null     float64
 94  Volatility_eqnr    35 non-null     float64
 95  Price_shell        44 non-null     object
 96  Vol_shell          44 non-null     object
 97  Change_shell       44 non-null     float64
 98  Volatility_shell   35 non-null     float64
 99  Post_Invasion      44 non-null     int64
dtypes: datetime64[ns](1), float64(72), int64(1), object(26)
memory usage: 34.5+ KB
```

Analytics

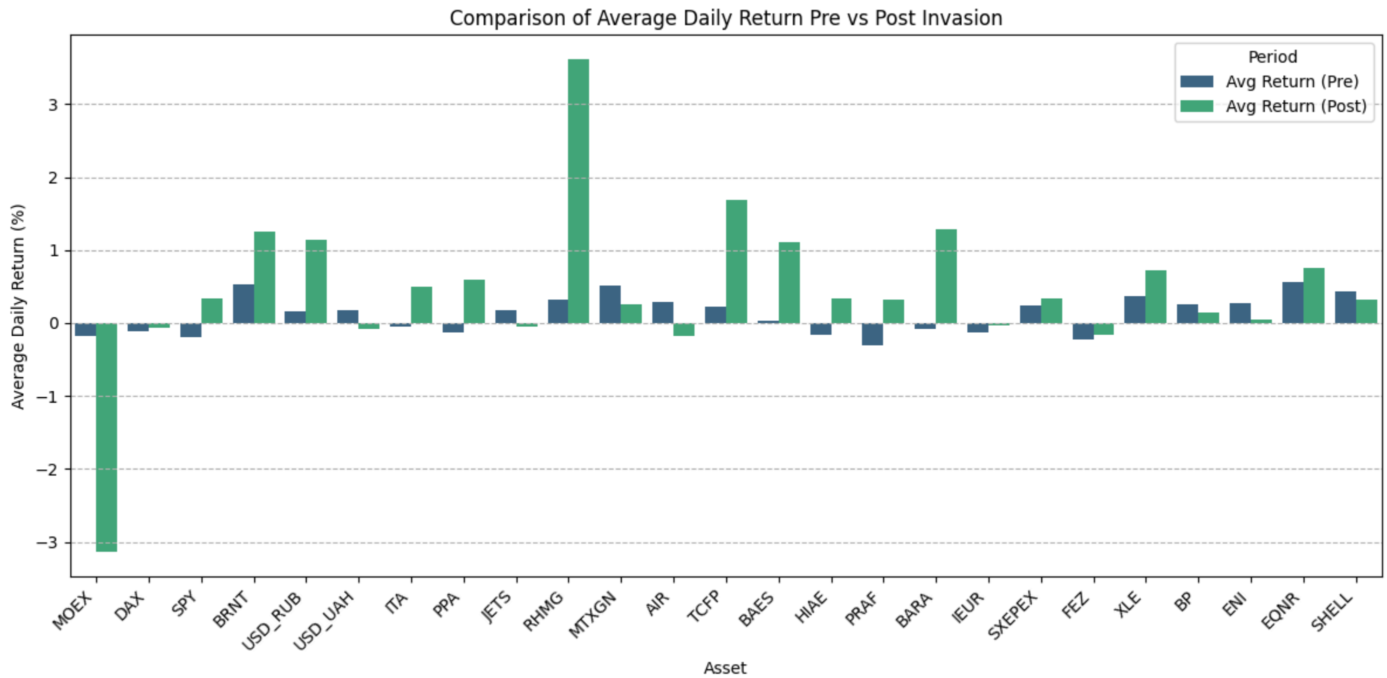
```
# Compare average return and volatility pre vs post invasion

results = []

for asset in ['moex', 'dax', 'spy', 'brnt', 'usd_rub', 'usd_uah', 'ita', 'ppa', 'jets', 'rhmg', 'mtxgn', 'air', 'tcfp', 'baes',

    pre = master_df[master_df['Post_Invasion'] == 0][f'Change_{asset}']
    post = master_df[master_df['Post_Invasion'] == 1][f'Change_{asset}']

    vol_pre = master_df[master_df['Post_Invasion'] == 0][f'Volatility_{asset}']
    vol_post = master_df[master_df['Post_Invasion'] == 1][f'Volatility_{asset}']

    results.append({
        'Asset': asset.upper(),
        'Avg Return (Pre)': pre.mean(),
        'Avg Return (Post)': post.mean(),
        'Avg Volatility (Pre)': vol_pre.mean(),
        'Avg Volatility (Post)': vol_post.mean()
    })

df_summary = pd.DataFrame(results)
df_summary
```

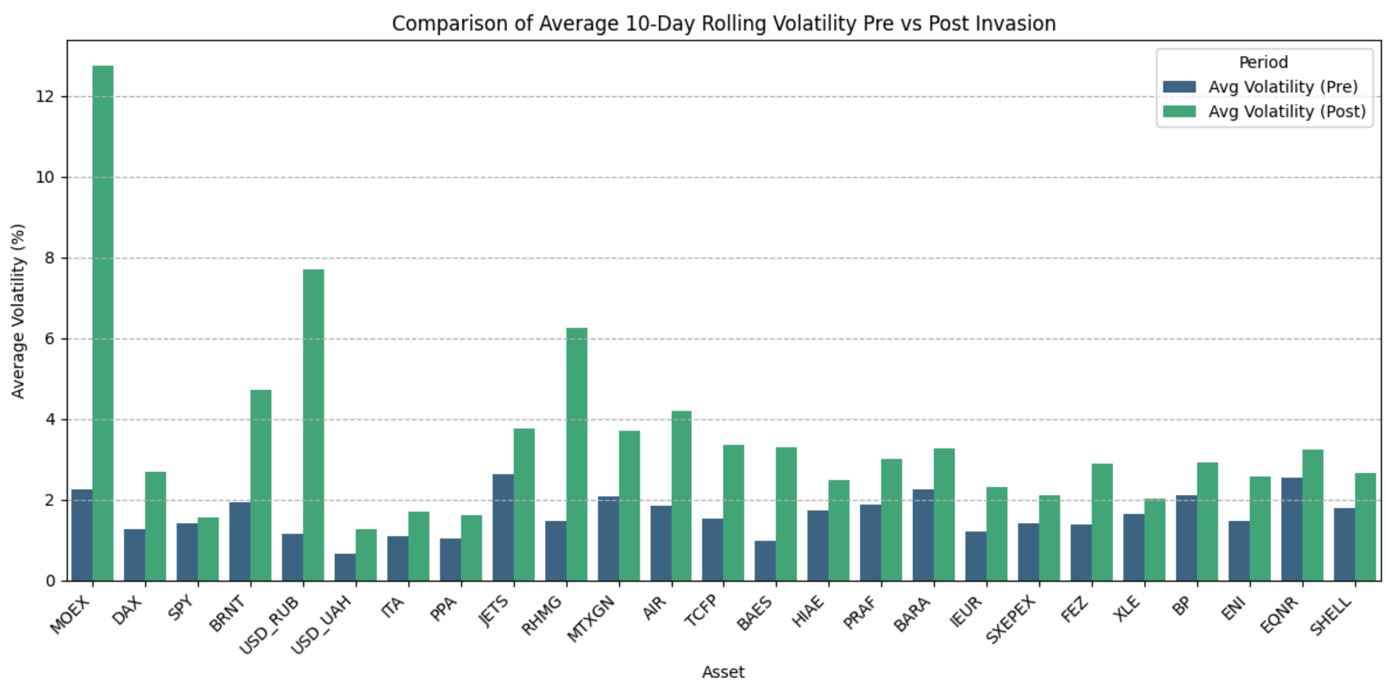
	Asset	Avg Return (Pre)	Avg Return (Post)	Avg Volatility (Pre)	Avg Volatility (Post)
0	MOEX	-0.175714	-3.132500	2.238705	12.735469
1	DAX	-0.109091	-0.065000	1.256182	2.689128
2	SPY	-0.188571	0.330909	1.406821	1.570113
3	BRNT	0.537727	1.247273	1.946249	4.718068
4	USD_RUB	0.158182	1.147727	1.154455	7.713305
5	USD_UAH	0.181364	-0.076667	0.659456	1.276386
6	ITA	-0.050476	0.492273	1.081409	1.705321
7	PPA	-0.128571	0.600000	1.048380	1.623602
8	JETS	0.180952	-0.042273	2.627433	3.768123
9	RHMG	0.327273	3.612273	1.466134	6.242258
10	MTXGN	0.516364	0.260455	2.083038	3.690300
11	AIR	0.291364	-0.170909	1.837993	4.194190
12	TCFP	0.220000	1.691364	1.526619	3.350508
13	BAES	0.026364	1.102273	0.978166	3.306556
14	HIAE	-0.162381	0.340000	1.734913	2.478544
15	PRAF	-0.310476	0.320500	1.885201	3.007313
16	BARA	-0.077143	1.290500	2.261410	3.269118
17	IEUR	-0.127143	-0.035909	1.196044	2.300292
18	SXEPEX	0.232727	0.338182	1.416924	2.098271
19	FEZ	-0.218571	-0.155000	1.384980	2.888262
20	XLE	0.361905	0.715909	1.652420	2.011196
21	BP	0.250909	0.149091	2.094969	2.909806
22	ENI	0.271364	0.052727	1.479117	2.558126
23	EQNR	0.564545	0.761818	2.553143	3.226227
24	SHELL	0.431818	0.314091	1.794149	2.658390

Next steps: Generate code with df_summary | View recommended plots | New interactive sheet

```
# Visualizing df_summary

# Melt the DataFrame for easier plotting
df_summary_melted_return = df_summary.melt(
    id_vars='Asset',
    value_vars=['Avg Return (Pre)', 'Avg Return (Post)'],
```

```
        var_name='Period',
        value_name='Average Daily Return (%)'
    )

    df_summary_melted_volatility = df_summary.melt(
        id_vars='Asset',
        value_vars=['Avg Volatility (Pre)', 'Avg Volatility (Post)'],
        var_name='Period',
        value_name='Average 10-Day Rolling Volatility (%)'
    )

    # Create bar plot for Average Daily Return
    plt.figure(figsize=(12, 6))
    sns.barplot(data=df_summary_melted_return, x='Asset', y='Average Daily Return (%)', hue='Period', palette='viridis')
    plt.title('Comparison of Average Daily Return Pre vs Post Invasion')
    plt.ylabel('Average Daily Return (%)')
    plt.xlabel('Asset')
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Period')
    plt.grid(axis='y', linestyle='--')
    plt.tight_layout()
    plt.show()

    # Create bar plot for Average 10-Day Rolling Volatility
    plt.figure(figsize=(12, 6))
    sns.barplot(data=df_summary_melted_volatility, x='Asset', y='Average 10-Day Rolling Volatility (%)', hue='Period', palette='viri
    plt.title('Comparison of Average 10-Day Rolling Volatility Pre vs Post Invasion')
    plt.ylabel('Average Volatility (%)')
    plt.xlabel('Asset')
    plt.xticks(rotation=45, ha='right')
    plt.legend(title='Period')
    plt.grid(axis='y', linestyle='--')
    plt.tight_layout()
    plt.show()
```



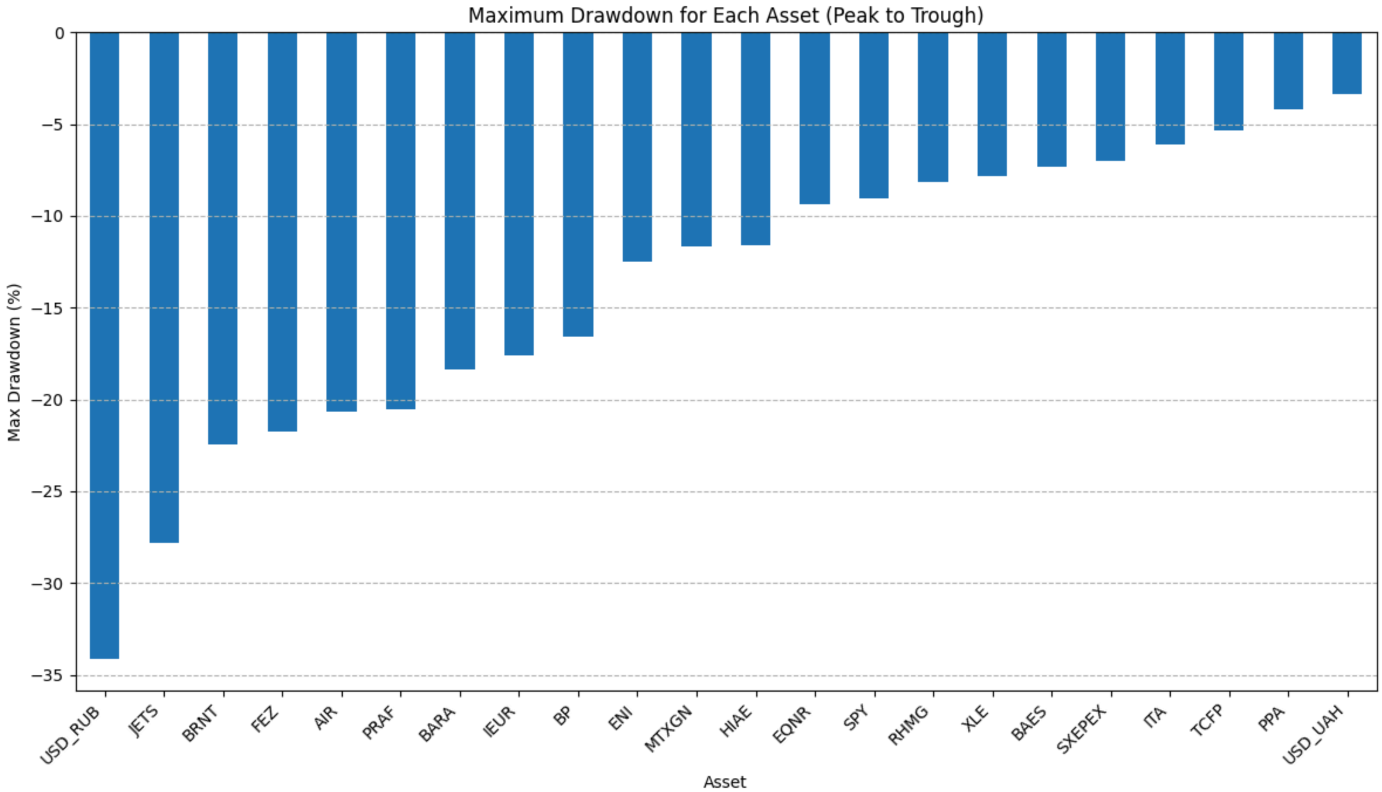



```
# Ensure all Price_* columns are numeric
for col in master_df.columns:
    if col.startswith('Price_'):
        master_df[col] = pd.to_numeric(master_df[col], errors='coerce')
```

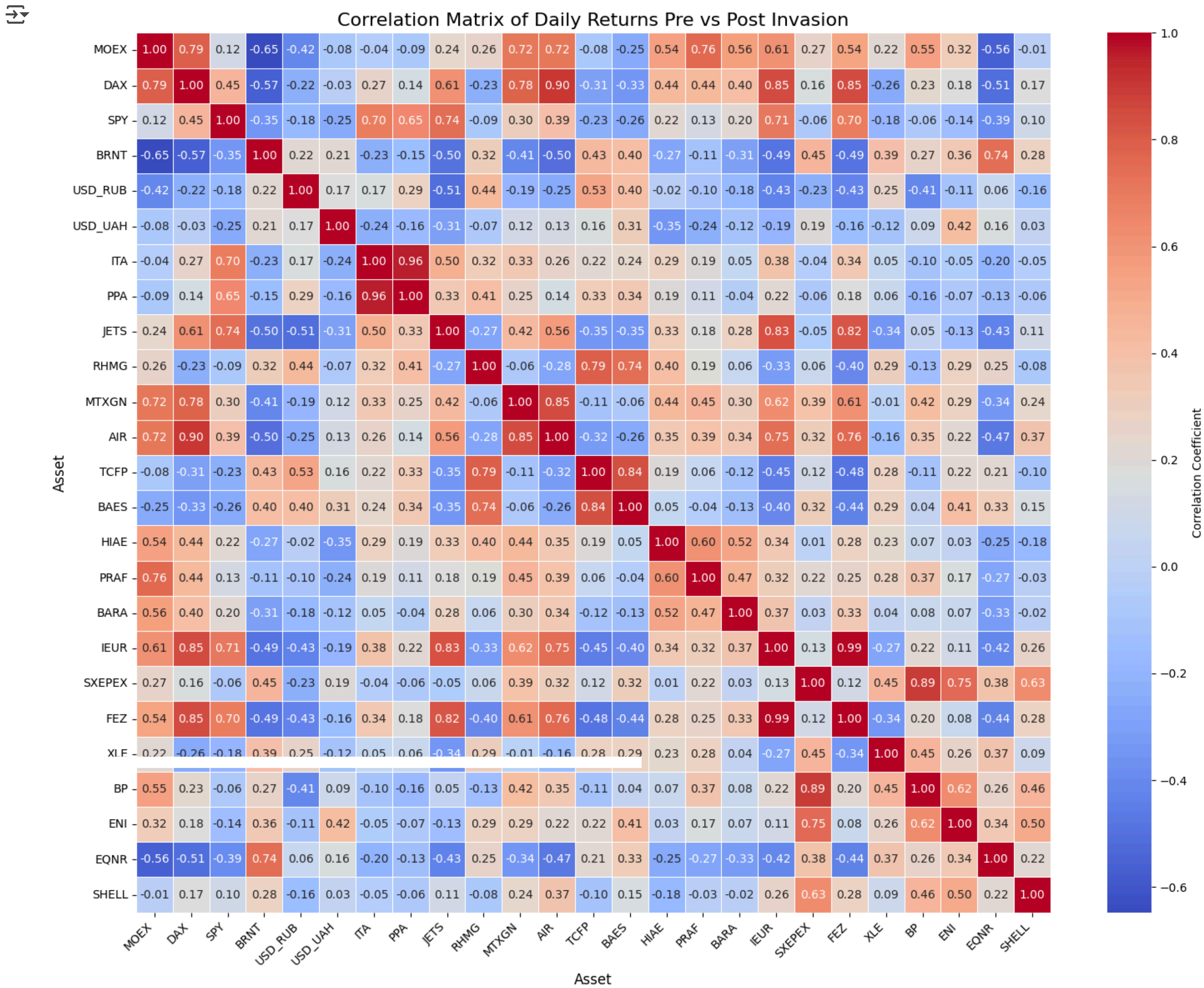
```
# Volatility Spikes & Drawdown Analysis

# Identify the biggest volatility spikes

# Calculate maximum drawdown in prices for each asset
# (how much it fell from the peak before bottoming)

# Shows market panic moments and asset fragility
```

```
# Shows market panic moments and asset fragility
# Helps visualize investor behavior under stress

def max_drawdown(prices):
    """
    Calculates the maximum drawdown of a price series.

    Parameters:
        prices (pd.Series): A pandas Series of price data.

    Returns:
        float: The maximum drawdown as a percentage (negative value).
               Returns 0 if the series is empty or only one value.
    """
    if prices.empty or len(prices) < 2:
        return 0

    # Calculate cumulative maximum
    cumulative_max = prices.cummax()

    # Calculate the drawdown
    drawdown = (prices - cumulative_max) / cumulative_max * 100

    # Find the minimum drawdown (most negative)
    max_dd = drawdown.min()

    return max_dd

max_drawdowns = {}

for asset in ['moex', 'dax', 'spy', 'brnt', 'usd_rub', 'usd_uah', 'ita', 'ppa', 'jets', 'rhmg', 'mtxgn', 'air', 'tcfp', 'baes', '
    price_col_name = f'Price_{asset}'
    if price_col_name in master_df.columns and pd.api.types.is_numeric_dtype(master_df[price_col_name]):
        # Ensure there are enough data points for drawdown calculation
        if master_df[price_col_name].dropna().shape[0] > 1:
            dd = max_drawdown(master_df[price_col_name].dropna())
            max_drawdowns[asset.upper()] = dd
        else:
             max_drawdowns[asset.upper()] = None
             print(f"Warning: Not enough data points for Max Drawdown for {asset.upper()}. Column '{price_col_name}'.")

    else:
        max_drawdowns[asset.upper()] = None
        print(f"Warning: Could not calculate Max Drawdown for {asset.upper()}. Column '{price_col_name}' not found or not numeric


# Create a DataFrame for visualization
df_max_drawdowns = pd.DataFrame.from_dict(max_drawdowns, orient='index', columns=['Max Drawdown (%)'])

# Drop assets where drawdown could not be calculated
df_max_drawdowns = df_max_drawdowns.dropna()

# Sort values for better visualization
df_max_drawdowns = df_max_drawdowns.sort_values(by='Max Drawdown (%)')

# Plotting the Max Drawdowns
plt.figure(figsize=(12, 7))
df_max_drawdowns['Max Drawdown (%)'].plot(kind='bar')
plt.title('Maximum Drawdown for Each Asset (Peak to Trough)')
plt.xlabel('Asset')
plt.ylabel('Max Drawdown (%)')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```

```
Warning: Not enough data points for Max Drawdown for MOEX. Column 'Price_moex'.
Warning: Not enough data points for Max Drawdown for DAX. Column 'Price_dax'.
Warning: Not enough data points for Max Drawdown for SHELL. Column 'Price_shell'.
```




```
# Correlation Matrix (Volatility and Returns)

# Compute pairwise correlations between assets' returns or volatility
# Understand contagion effects between markets (e.g., Brent Crude vs MOEX)

change_cols = [f'Change_{a}' for a in ['moex', 'dax', 'spy', 'brnt', 'usd_rub', 'usd_uah', 'ita', 'ppa', 'jets', 'rhmg', 'mtxgn'

# Ensure only change columns with enough non-null values are used
# Drop columns that might be all NaN or have very few data points after merging
valid_change_cols = [col for col in change_cols if col in master_df.columns and master_df[col].dropna().shape[0] > 1]

if not valid_change_cols:
    print("No valid change columns found for correlation matrix.")
else:
    correlation_matrix = master_df[valid_change_cols].corr()

    # Map column names back to original asset names for better readability on the plot
    column_labels = [col.replace('Change_', '').upper() for col in correlation_matrix.columns]

    plt.figure(figsize=(15, 12)) # Increase figure size
    sns.heatmap(
        correlation_matrix,
        annot=True,        # Add annotations (correlation values)
        fmt=".2f",         # Format annotations to 2 decimal places
        cmap='coolwarm',   # Use a diverging colormap
        linewidths=.5,     # Add lines between cells
        cbar_kws={'label': 'Correlation Coefficient'} # Label the color bar
    )
    plt.title('Correlation Matrix of Daily Returns Pre vs Post Invasion', fontsize=16)
    plt.xlabel('Asset', fontsize=12)
    plt.ylabel('Asset', fontsize=12)
    # Set custom tick labels
    plt.xticks(ticks=np.arange(len(column_labels)) + 0.5, labels=column_labels, rotation=45, ha='right', fontsize=10)
    plt.yticks(ticks=np.arange(len(column_labels)) + 0.5, labels=column_labels, rotation=0, fontsize=10)
    plt.tight_layout() # Adjust layout to prevent labels overlapping
    plt.show()
```



Based on the generated correlation matrix heatmap:

General Interpretation:

- **Color Intensity:** The color intensity and hue indicate the strength and direction of the correlation.
  - **Strong Positive Correlation (closer to 1, reddish):** Assets move in the same direction. When one asset's return goes up, the other tends to go up as well.

  - **Strong Negative Correlation (closer to -1, bluish):** Assets move in opposite directions. When one asset's return goes up, the other tends to go down.
  - **Weak or No Correlation (closer to 0, lighter colors):** Assets' movements are not strongly related.

Specific Inferences (Look at the individual cell values):

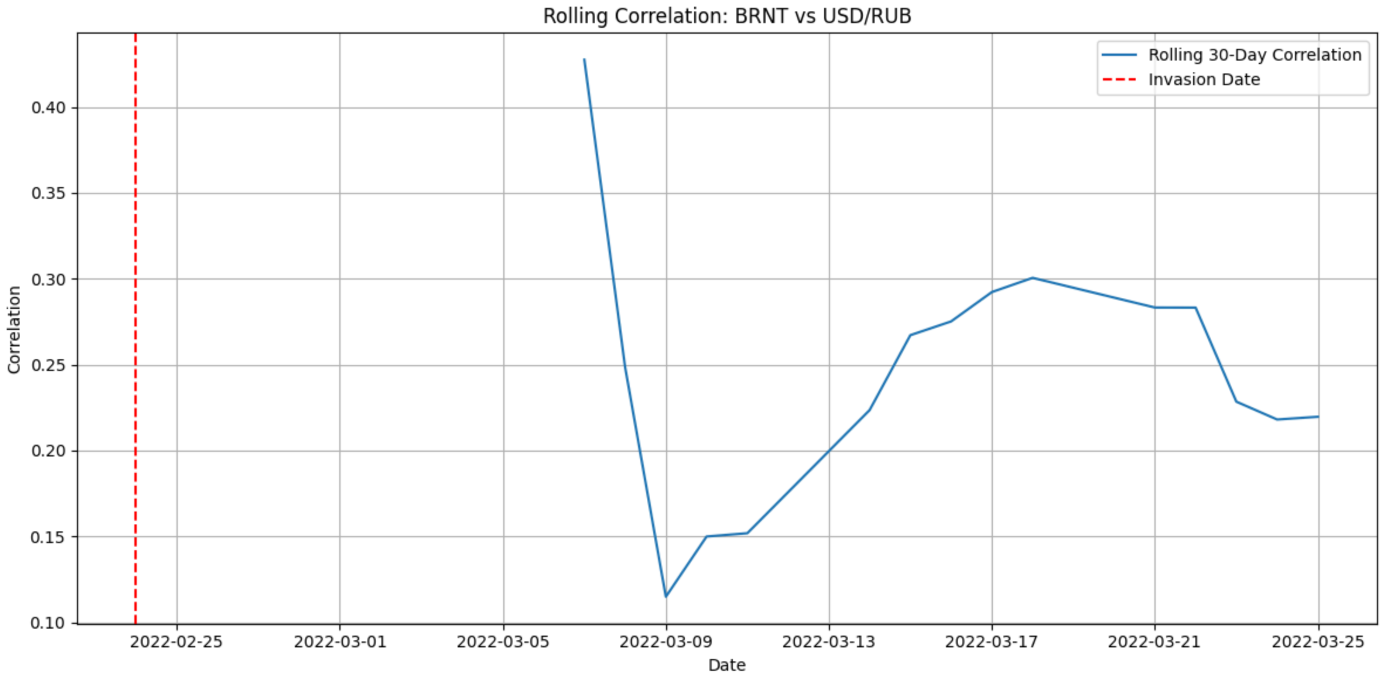
- **Diagonal:** The diagonal is always 1.00 as it represents the correlation of an asset with itself.
- **Symmetry:** The matrix is symmetrical; the correlation between Asset A and Asset B is the same as between Asset B and Asset A.
- **Pairs with High Positive Correlation:** Look for high positive values (close to 1). This indicates assets that are highly influenced by similar market factors or are in the same sector/region. For example, you would expect indices from similar markets (like DAX and FEZ or ITA and PPA) or assets within the same sector (like the various energy or aerospace stocks if included) to have high positive correlations.
- **Pairs with High Negative Correlation:** Look for high negative values (close to -1). This indicates assets that tend to move inversely. This can be interesting for diversification or hedging strategies. For example, a currency pair like USD/RUB might have a negative correlation with the MOEX index if a stronger ruble negatively impacts the predominantly export-oriented companies listed on the MOEX.
- **Low Correlation Pairs:** Look for values close to 0. These assets are less sensitive to the same drivers and could offer diversification benefits in a portfolio.
- **MOEX (MOEX) Correlations:** Pay close attention to the row/column for MOEX. Its correlations with other indices (like DAX, SPY, FEZ) and assets will be particularly informative about how the Russian market moves in relation to global markets and specific sectors during this period (which spans before and after the invasion). You might observe significant shifts in these correlations post-invasion compared to pre-invasion if the conflict fundamentally changed the relationship between markets.
- **Brent (BRNT) Correlations:** Look at the correlation of BRNT with other assets, especially energy stocks (XLE, BP, ENI, EQNR, SHELL) and potentially currencies (USD/RUB, USD/UAH). High positive correlation with energy stocks and potentially negative correlations with currencies of oil-importing nations could be expected.
- **Currency Pair (USD/RUB, USD/UAH) Correlations:** Examine how these currency pairs correlate with other assets. Changes in the value of the Ruble and Hryvnia are likely linked to the economic and geopolitical situation, and their correlation with indices and commodity prices can reveal these linkages.
- **Sector-Specific Correlations:** If you included various stocks from specific sectors (Aerospace/Defense like ITA, PPA, JETS, RHMG, MTXGN, AIR, TCFP, BAES, HIAE, PRAF, BARA, IEUR, SXEPEX, FEZ, XLE, BP, ENI, EQNR, SHELL), you can see how correlated assets within the same sector are, and how correlated assets across different sectors are. Aerospace/Defense might show different correlation patterns compared to energy assets, especially in the context of a conflict.

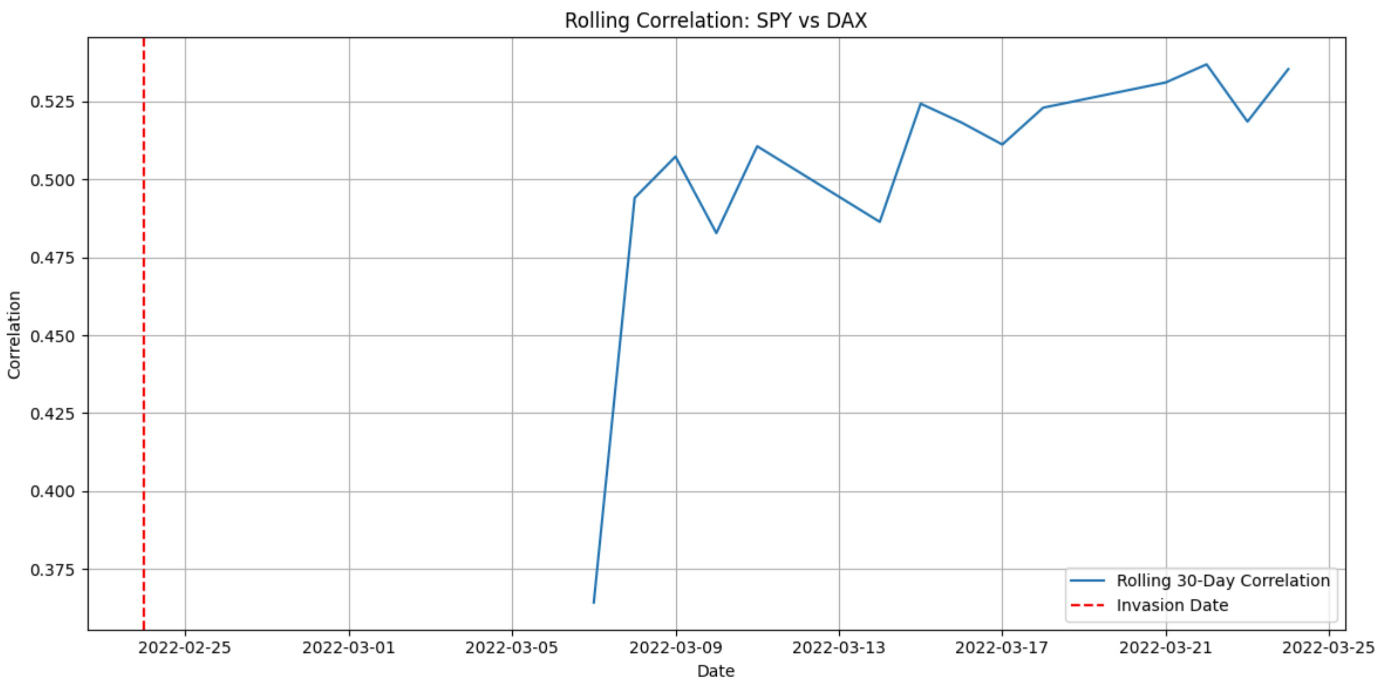
In summary, the correlation matrix provides a snapshot of the pairwise linear relationships between the daily returns of the analyzed assets during the entire period covered by the data. By examining the specific values, you can infer which assets tend to move together, which move in opposite directions, and which have less predictable relationships, giving insights into market linkages and dependencies.

To get a deeper understanding of the impact of the invasion, you would ideally compute separate correlation matrices for the *pre-invasion* and *post-invasion* periods and compare them directly. The code provided calculates a single correlation matrix for the entire dataset.

```
# Rolling Correlation Between Important Assets

# How did the correlation between S&P 500 and Brent Crude change over time?
# Shows how markets become more/less connected in stress

# Function to calculate and plot rolling correlation


def plot_rolling_correlation(df, asset1_change_col, asset2_change_col, window=30, title='Rolling Correlation'):
    """
    Calculates and plots the rolling correlation between two asset change columns.

    Parameters:
        df (pd.DataFrame): DataFrame containing the date and asset change columns.
        asset1_change_col (str): Name of the first asset's change column (e.g., 'Change_brnt').
        asset2_change_col (str): Name of the second asset's change column (e.g., 'Change_usd_rub').
        window (int): The rolling window size for correlation calculation.
        title (str): Title for the plot.
    """
    # Select the two columns and drop rows with NaN values in either column
    temp_df = df[[asset1_change_col, asset2_change_col]].dropna()

    # Calculate rolling correlation
    rolling_corr = temp_df[asset1_change_col].rolling(window=window).corr(temp_df[asset2_change_col])

    # Add the Date index back for plotting
    rolling_corr = pd.DataFrame({'Date': temp_df.index, 'Correlation': rolling_corr.values})
    # Ensure 'Date' is datetime if it's not already the index
```

```
        if not pd.api.types.is_datetime64_any_dtype(rolling_corr['Date']):
            rolling_corr['Date'] = df.loc[rolling_corr.index, 'Date']

        # Plotting the rolling correlation
        plt.figure(figsize=(12, 6))
        plt.plot(rolling_corr['Date'], rolling_corr['Correlation'], label=f'Rolling {window}-Day Correlation')
        plt.axvline(pd.to_datetime("2022-02-24"), color='red', linestyle='--', label='Invasion Date')
        plt.title(title)
        plt.xlabel("Date")
        plt.ylabel("Correlation")
        plt.legend()
        plt.grid(True)
        plt.tight_layout()
        plt.show()

    # Calculate and plot rolling correlation for BRNT-USD/RUB
    plot_rolling_correlation(master_df, 'Change_brnt', 'Change_usd_rub', window=30, title='Rolling Correlation: BRNT vs USD/RUB')

    # Calculate and plot rolling correlation for DAX-SPY
    plot_rolling_correlation(master_df, 'Change_spy', 'Change_dax', window=30, title='Rolling Correlation: SPY vs DAX')
```






```
# List of asset pairs for which to plot rolling correlation (Change columns)
correlation_pairs = [
    ('moex', 'dax'),
    ('moex', 'spy'),
    ('moex', 'brnt'),
    ('moex', 'usd_uah'),
    ('moex', 'ita'),
    ('moex', 'ppa'),
    ('dax', 'brnt'),
    ('spy', 'brnt'),
    ('brnt', 'usd_uah'),
    ('dax', 'ita'),
    ('dax', 'ppa'),
```