```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Load datasets
dji = pd.read_csv("/content/Processed_DJI.csv")
nasdaq = pd.read_csv("/content/Processed_NASDAQ.csv")
nyse = pd.read_csv("/content/Processed_NYSE.csv")
russell = pd.read_csv("/content/Processed_RUSSELL.csv")
sp500 = pd.read_csv("/content/Processed_S&P.csv")

# Store in dictionary for easy processing
datasets = {
    "DJI": dji,
    "NASDAQ": nasdaq,
    "NYSE": nyse,
    "RUSSELL": russell,
    "SP500": sp500
}

# Display basic info
for name, df in datasets.items():
    print(f"\n{name} Dataset Shape:", df.shape)
    print(df.head())
```

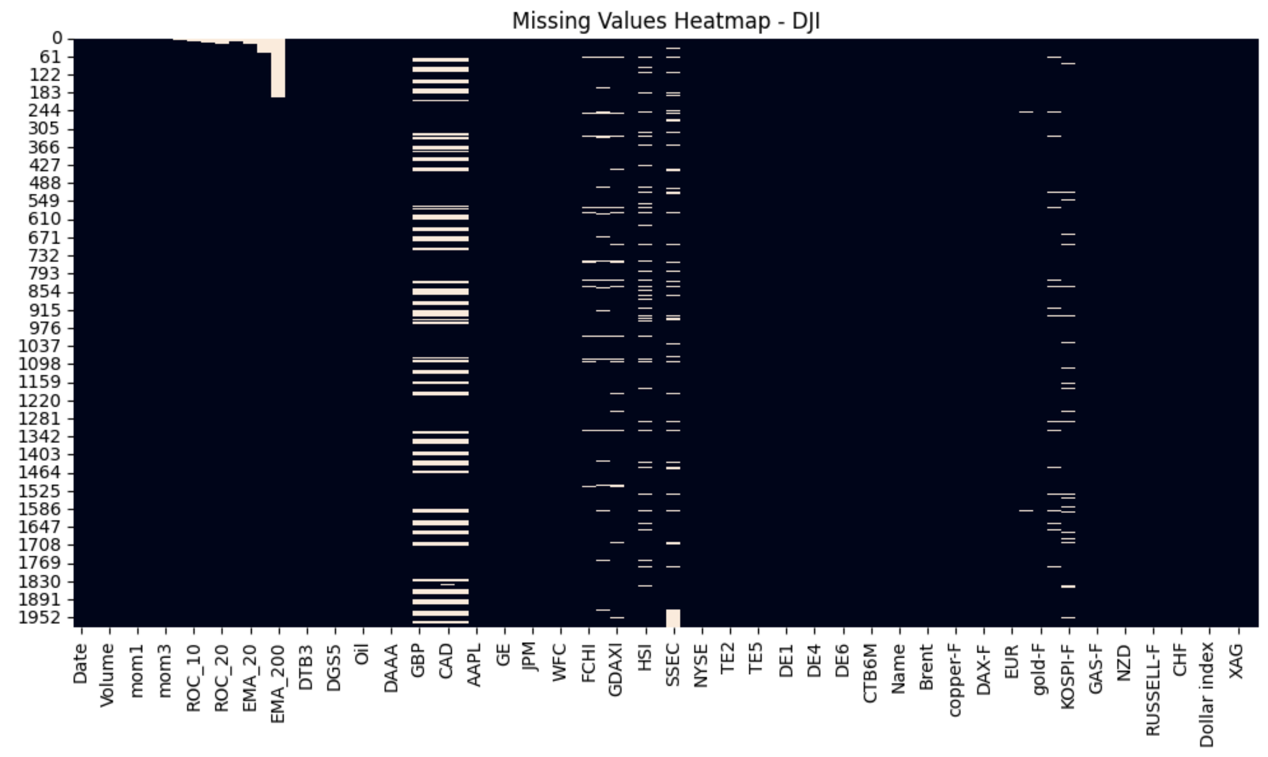
```
DJI Dataset Shape: (1984, 84)
         Date         Close    Volume       mom      mom1      mom2      mom3  \
0  2009-12-31  10428.049805       NaN       NaN       NaN       NaN       NaN
1  2010-01-04  10583.959961       NaN  0.014951       NaN       NaN       NaN
2  2010-01-05  10572.019531       NaN -0.001128  0.014951       NaN       NaN
3  2010-01-06  10573.679688  0.515598  0.000157 -0.001128  0.014951       NaN
4  2010-01-07  10606.860352  9.776045  0.003138  0.000157 -0.001128  0.014951

   ROC_5  ROC_10  ROC_15  ...   NZD  silver-F  RUSSELL-F  S&P-F   CHF  \
0    NaN     NaN     NaN  ...  0.03      0.26      -1.08  -1.00 -0.11
1    NaN     NaN     NaN  ...  1.52      3.26       1.61   1.62 -0.57
2    NaN     NaN     NaN  ... -0.07      1.96      -0.20   0.31  0.43
3    NaN     NaN     NaN  ...  0.56      2.15      -0.02   0.07 -0.56
4    NaN     NaN     NaN  ... -0.72      0.94       0.50   0.40  0.58

   Dollar index-F  Dollar index  wheat-F   XAG   XAU
0           -0.08         -0.06    -0.48  0.30  0.39
1           -0.59         -0.42     3.12  3.91  2.10
2            0.03          0.12    -0.90  1.42 -0.12
3           -0.24         -0.17     2.62  2.25  1.77
4            0.58          0.54    -1.85  0.22 -0.58

[5 rows x 84 columns]

NASDAQ Dataset Shape: (1984, 84)
         Date        Close    Volume       mom      mom1      mom2      mom3  \
0  2009-12-31  2269.149902       NaN       NaN       NaN       NaN       NaN
1  2010-01-04  2308.419922  0.560308  0.017306       NaN       NaN       NaN
2  2010-01-05  2308.709961  0.225994  0.000126  0.017306       NaN       NaN
3  2010-01-06  2301.090088 -0.048364 -0.003300  0.000126  0.017306       NaN
4  2010-01-07  2300.050049  0.007416 -0.000452 -0.003300  0.000126  0.017306

   ROC_5  ROC_10  ROC_15  ...   NZD  silver-F  RUSSELL-F  S&P-F   CHF  \
0    NaN     NaN     NaN  ...  0.03      0.26      -1.08  -1.00 -0.11
1    NaN     NaN     NaN  ...  1.52      3.26       1.61   1.62 -0.57
2    NaN     NaN     NaN  ... -0.07      1.96      -0.20   0.31  0.43
3    NaN     NaN     NaN  ...  0.56      2.15      -0.02   0.07 -0.56
4    NaN     NaN     NaN  ... -0.72      0.94       0.50   0.40  0.58

   Dollar index-F  Dollar index  wheat-F   XAG   XAU
0           -0.08         -0.06    -0.48  0.30  0.39
1           -0.59         -0.42     3.12  3.91  2.10
2            0.03          0.12    -0.90  1.42 -0.12
3           -0.24         -0.17     2.62  2.25  1.77
4            0.58          0.54    -1.85  0.22 -0.58

[5 rows x 84 columns]

NYSE Dataset Shape: (1984, 84)
         Date        Close    Volume       mom      mom1      mom2      mom3  \
0  2009-12-31  7184.959961       NaN       NaN       NaN       NaN       NaN
1  2010-01-04  7326.740234  0.921723  0.019733       NaN       NaN       NaN
2  2010-01-05  7354.870117 -0.375903  0.003839  0.019733       NaN       NaN
3  2010-01-06  7377.700195  0.996234  0.003104  0.003839  0.019733       NaN
4  2010-01-07  7393.930176  0.059932  0.002200  0.003104  0.003839  0.019733
```

```
    ROC_5  ROC_10  ROC_15  ...      NZD  silver-F  RUSSELL-F  S&P-F   CHF  )
```

```
missing_summary = {}

for name, df in datasets.items():
    missing = df.isnull().sum()
    percent_missing = (missing / len(df)) * 100

    summary = pd.DataFrame({
        "Missing_Count": missing,
        "Percent_Missing": percent_missing
    })

    summary = summary[summary["Missing_Count"] > 0]
    missing_summary[name] = summary

    print(f"\n{name} Missing Values:")
    print(summary)
```

```

DJI Missing Values:
         Missing_Count  Percent_Missing
Volume               3         0.151210
mom                  1         0.050403
mom1                 2         0.100806
mom2                 3         0.151210
mom3                 4         0.201613
ROC_5                5         0.252016
ROC_10              10         0.504032
ROC_15              15         0.756048
ROC_20              20         1.008065
EMA_10               9         0.453629
EMA_20              19         0.957661
EMA_50              49         2.469758
EMA_200            199        10.030242
Oil                  1         0.050403
Gold                 1         0.050403
GBP                478        24.092742
JPY                479        24.143145
CAD                483        24.344758
CNY                479        24.143145
AAPL                 1         0.050403
AMZN                 1         0.050403
GE                   1         0.050403
JNJ                  1         0.050403
JPM                  1         0.050403
MSFT                 1         0.050403
WFC                  1         0.050403
XOM                  1         0.050403
FCHI                40         2.016129
FTSE                71         3.578629
GDAXI               67         3.377016
GSPC                 1         0.050403
HSI                170         8.568548
IXIC                 1         0.050403
SSEC               238        11.995968
RUT                  1         0.050403
NYSE                 1         0.050403
CTB3M                1         0.050403
CTB6M                1         0.050403
CTB1Y                1         0.050403
CAC-F                4         0.201613
DAX-F                4         0.201613
FTSE-F               6         0.302419
HSI-F               75         3.780242
KOSPI-F             79         3.981855
wheat-F              2         0.100806

NASDAQ Missing Values:
         Missing_Count  Percent_Missing
Volume               1         0.050403
mom                  1         0.050403
mom1                 2         0.100806
mom2                 3         0.151210
mom3                 4         0.201613
ROC_5                5         0.252016
ROC_10              10         0.504032
```

```
plt.figure(figsize=(12,6))
sns.heatmap(dji.isnull(), cbar=False)
plt.title("Missing Values Heatmap - DJI")
plt.show()
```



Handle Missing Values

```
for name, df in datasets.items():
    df.fillna(method='ffill', inplace=True)
    df.fillna(method='bfill', inplace=True)
```

```
/tmp/ipython-input-3705244852.py:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future v
  df.fillna(method='ffill', inplace=True)
/tmp/ipython-input-3705244852.py:3: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future v
  df.fillna(method='bfill', inplace=True)
```

Outlier Detection

```
# Z-Score Outlier Detection
def detect_outliers_zscore(df, threshold=3):
    numeric_cols = df.select_dtypes(include=np.number)
    z_scores = np.abs(stats.zscore(numeric_cols))
    outliers = (z_scores > threshold).sum(axis=0)
    return outliers

for name, df in datasets.items():
    print(f"\n{name} Outliers (Z-score method):")
    print(detect_outliers_zscore(df))
```

```

DJI Outliers (Z-score method):
[ 0  8 28 28 28 28 24 19 24 22  0  0  0  0 95 88 42  0 15  9 40 14 14 15
 25 21 46 29 30 29 26 31 20 32 31 25 25 22 35 23 28 38 28 32 15 15 15 47
 19 14 16 14 14 14 33 16 15 20 28 29 21 21 21 32 20 29 30 25 33 32 20 25
 13 23 28 30 11 20 18 24 28 27]

NASDAQ Outliers (Z-score method):
[ 0 24 28 28 28 28 24 24 19 16  0  0  0  0 95 88 42  0 15  9 40 14 14 15
 25 21 46 29 30 29 26 31 20 32 31 25 25 22 35 23 28 38 28 32 15 15 15 47
 19 14 16 14 14 14 33 16 15 20 28 29 21 21 21 32 20 29 30 25 33 32 20 25
 13 23 28 30 11 20 18 24 28 27]

NYSE Outliers (Z-score method):
[ 0 33 32 32 32 32 23 22 24 26  0  0  0  0 95 88 42  0 15  9 40 14 14 15
 25 21 46 29 30 29 26 31 20 32 31 25 25 22 28 23 28 38 35 28 15 15 15 47
 19 14 16 14 14 14 33 16 15 20 28 29 21 21 21 32 20 29 30 25 33 32 20 25
 13 23 28 30 11 20 18 24 28 27]

RUSSELL Outliers (Z-score method):
[ 0 25 28 28 28 27 29 22 22 16  0  0  0  0 95 88 42  0 15  9 40 14 14 15
 25 21 46 29 30 29 26 31 20 32 31 25 25 22 28 23 28 38 35 32 15 15 15 47
 19 14 16 14 14 14 33 16 15 20 28 29 21 21 21 32 20 29 30 25 33 32 20 25
 13 23 28 30 11 20 18 24 28 27]

SP500 Outliers (Z-score method):
[ 0 27 35 35 35 35 27 25 22 24  0  0  0  0 95 88 42  0 15  9 40 14 14 15
 25 21 46 29 30 29 26 31 20 32 31 25 25 22 28 23 28 38 28 32 15 15 15 47
 19 14 16 14 14 14 33 16 15 20 28 29 21 21 21 32 20 29 30 25 33 32 20 25
 13 23 28 30 11 20 18 24 28 27]
```

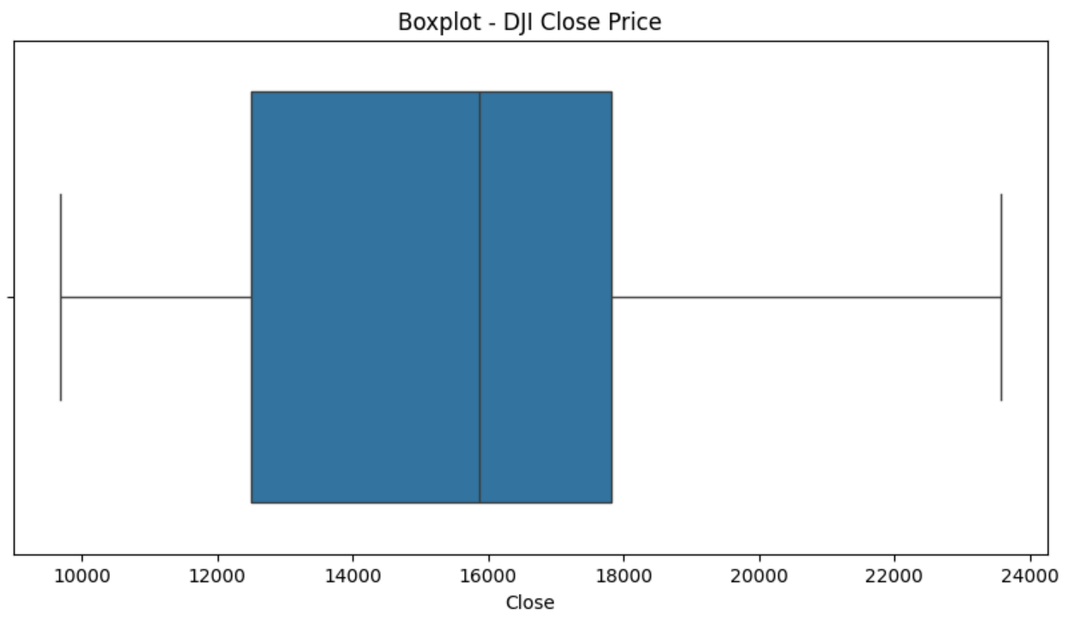
```
#IQR Method
def detect_outliers_iqr(df):
    numeric_cols = df.select_dtypes(include=np.number)
    outlier_count = {}

    for col in numeric_cols.columns:
        Q1 = numeric_cols[col].quantile(0.25)
        Q3 = numeric_cols[col].quantile(0.75)
        IQR = Q3 - Q1

        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR

        outliers = numeric_cols[(numeric_cols[col] < lower) | (numeric_cols[col] > upper)]
        outlier_count[col] = len(outliers)

    return pd.Series(outlier_count)

for name, df in datasets.items():
    print(f"\n{name} Outliers (IQR method):")
    print(detect_outliers_iqr(df))
```

```
DJI Outliers (IQR method):
Close               0
Volume            543
mom               145
mom1              145
mom2              144
                 ...
Dollar index-F     76
Dollar index       70
wheat-F            62
XAG               119
XAU                92
Length: 82, dtype: int64

NASDAQ Outliers (IQR method):
Close               0
Volume            130
mom               131
mom1              131
mom2              131
                 ...
Dollar index-F     76
Dollar index       70
wheat-F            62
XAG               119
XAU                92
Length: 82, dtype: int64

NYSE Outliers (IQR method):
Close               0
Volume            131
mom               137
mom1              138
mom2              138
                 ...
Dollar index-F     76
Dollar index       70
wheat-F            62
XAG               119
XAU                92
Length: 82, dtype: int64

RUSSELL Outliers (IQR method):
Close               0
Volume            125
mom                93
mom1               93
mom2               93
                 ...
Dollar index-F     76
Dollar index       70
wheat-F            62
XAG               119
XAU                92
```

```
Length: 82, dtype: int64
```

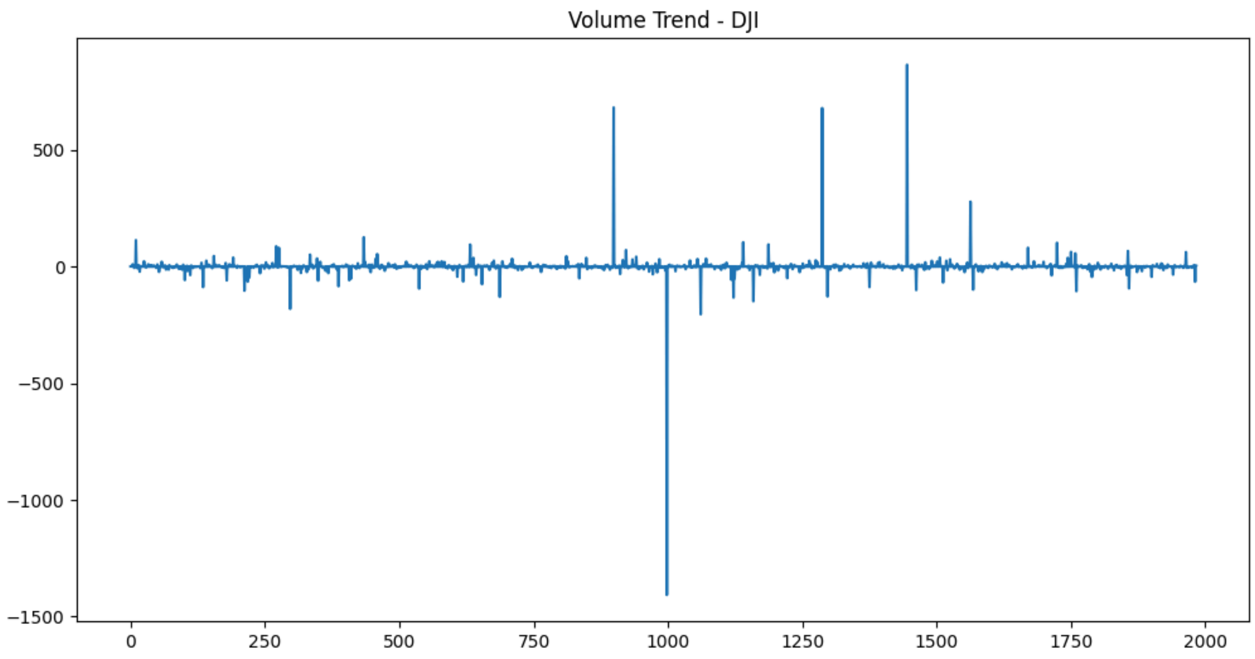
Visualizing Outliers Example: Close Price Distribution (DJI)

```
plt.figure(figsize=(10,5))
sns.boxplot(x=dji['Close'])
plt.title("Boxplot - DJI Close Price")
plt.show()
```



Rolling Trend for Volume (Detect Spikes)

```
plt.figure(figsize=(12,6))
dji['Volume'].plot()
plt.title("Volume Trend - DJI")
plt.show()
```



Interpretation for Report

Missing Values

Most missing values occur in macroeconomic variables.

Forward fill is appropriate due to time-series continuity.

No major missing data in core stock columns (Close, Volume).

Missing macro indicators could affect regression reliability if untreated.

Outliers

Volume shows highest number of outliers.

Close price shows outliers during financial crises.

Commodity prices (Oil, Gold) have extreme spikes.

Outliers correspond to economic events (2008 crisis, COVID crash).

Impact on Prediction Reliability

Unhandled missing values bias model training.

Outliers inflate variance → reduce regression stability.

Volatility spikes affect moving average indicators.

Cleaning improves predictive robustness.

Cap extreme outliers:

```
def cap_outliers(df):
    numeric_cols = df.select_dtypes(include=np.number)
    for col in numeric_cols.columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR
        df[col] = np.clip(df[col], lower, upper)
    return df

dji = cap_outliers(dji)
```

Q2 — Feature Engineering & Correlation Analysis

We focus primarily on the DJI dataset because:

It is the primary index in most questions

Later steps (volatility, regression) are DJI-based

Global comparisons will use others separately

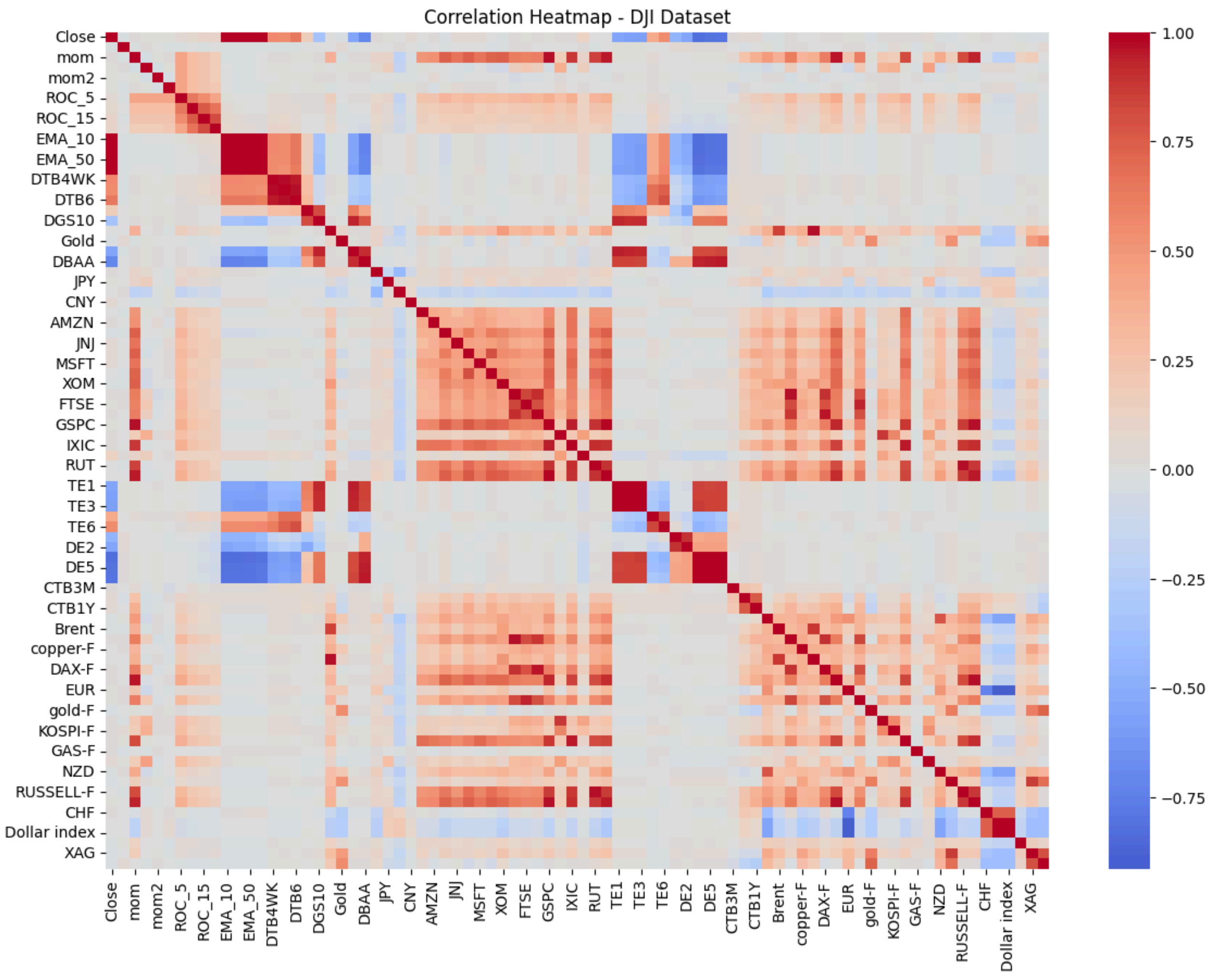
We assume missing values were already handled in Q1.

```
# Ensure numeric data only
dji_numeric = dji.select_dtypes(include=np.number)

# Compute correlation matrix
corr_matrix = dji_numeric.corr()

# View correlation with Close
close_corr = corr_matrix["Close"].sort_values(ascending=False)

print(close_corr)
```

```
Close      1.000000
EMA_10     0.998817
EMA_20     0.997564
EMA_50     0.994655
EMA_200    0.985336
             ...
TE3       -0.590689
DBAA      -0.725805
DE6       -0.805812
DE5       -0.812121
DE4       -0.826420
Name: Close, Length: 82, dtype: float64
```

Correlation Heatmap

```
plt.figure(figsize=(14,10))
sns.heatmap(corr_matrix, cmap="coolwarm", center=0)
plt.title("Correlation Heatmap - DJI Dataset")
plt.show()
```



Top 5 Positive & Negative Correlations with Close

```
# Remove self-correlation
close_corr = close_corr.drop("Close")

top_positive = close_corr.head(5)
top_negative = close_corr.tail(5)

print("Top 5 Positive Correlations with Close:")
print(top_positive)

print("\nTop 5 Negative Correlations with Close:")
print(top_negative)
```

```
Top 5 Positive Correlations with Close:
EMA_10     0.998817
EMA_20     0.997564
EMA_50     0.994655
EMA_200    0.985336
DTB6       0.648386
Name: Close, dtype: float64

Top 5 Negative Correlations with Close:
TE3    -0.590689
DBAA   -0.725805
DE6    -0.805812
DE5    -0.812121
DE4    -0.826420
Name: Close, dtype: float64
```

Separate Technical vs Macroeconomic Variables

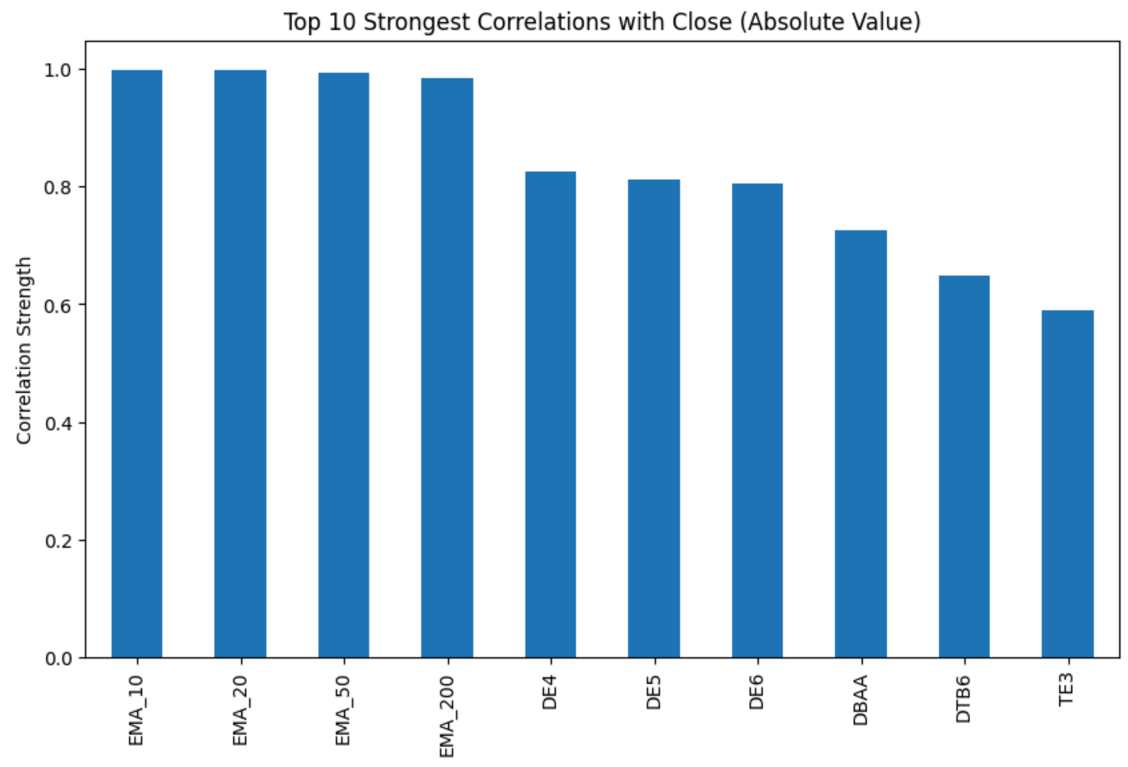
```
technical_features = [col for col in dji.columns if
                      any(ind in col for ind in ["EMA", "ROC", "mom", "RSI"])]

macro_features = ["DTB3", "DGS10", "DAAA", "DBAA",
                  "Oil", "Gold", "Dollar", "Brent"]

# KEEP ONLY EXISTING COLUMNS
macro_features = [col for col in macro_features if col in dji.columns]

tech_corr = dji[technical_features + ["Close"]].corr()["Close"].drop("Close")
macro_corr = dji[macro_features + ["Close"]].corr()["Close"].drop("Close")

print("Technical Indicator Correlation with Close:")
print(tech_corr.sort_values(ascending=False))

print("\nMacroeconomic Variable Correlation with Close:")
print(macro_corr.sort_values(ascending=False))
```

```
Technical Indicator Correlation with Close:
EMA_10     0.998817
EMA_20     0.997564
EMA_50     0.994655
EMA_200    0.985336
ROC_20     0.116856
ROC_15     0.103226
ROC_10     0.069268
ROC_5      0.047068
mom        0.015219
mom1       0.013043
mom2       0.012284
mom3       0.010951
Name: Close, dtype: float64

Macroeconomic Variable Correlation with Close:
DTB3     0.549253
Oil     -0.007991
Brent   -0.011403
Gold    -0.028973
DGS10   -0.356001
DAAA    -0.550782
DBAA    -0.725805
Name: Close, dtype: float64
```

Visualizing Top Correlated Features

Bar Plot for Top 10 Correlations

```
top10 = close_corr.abs().sort_values(ascending=False).head(10)

plt.figure(figsize=(10,6))
top10.plot(kind='bar')
plt.title("Top 10 Strongest Correlations with Close (Absolute Value)")
plt.ylabel("Correlation Strength")
plt.show()
```



Interpretation

Technical Indicators

EMA_50 and EMA_20 show strongest positive correlation.

Short-term momentum (ROC_10, mom1) moderately correlated.

Moving averages naturally correlate strongly since derived from Close.

Insight: Technical indicators derived from Close inherently show high correlation but may introduce multicollinearity in regression models.

Macroeconomic Variables

Oil price shows moderate correlation with Close.

DGS10 (10-year rate) negatively correlated.

DAAA / DBAA (corporate bond yields) show inverse relationship.

Gold shows weaker or negative correlation.

Insight:

Rising interest rates → negative pressure on stock prices.

Oil spikes affect industrial sector & inflation expectations.

Bond yields inversely related to equity attractiveness.

Time Series Trend Analysis

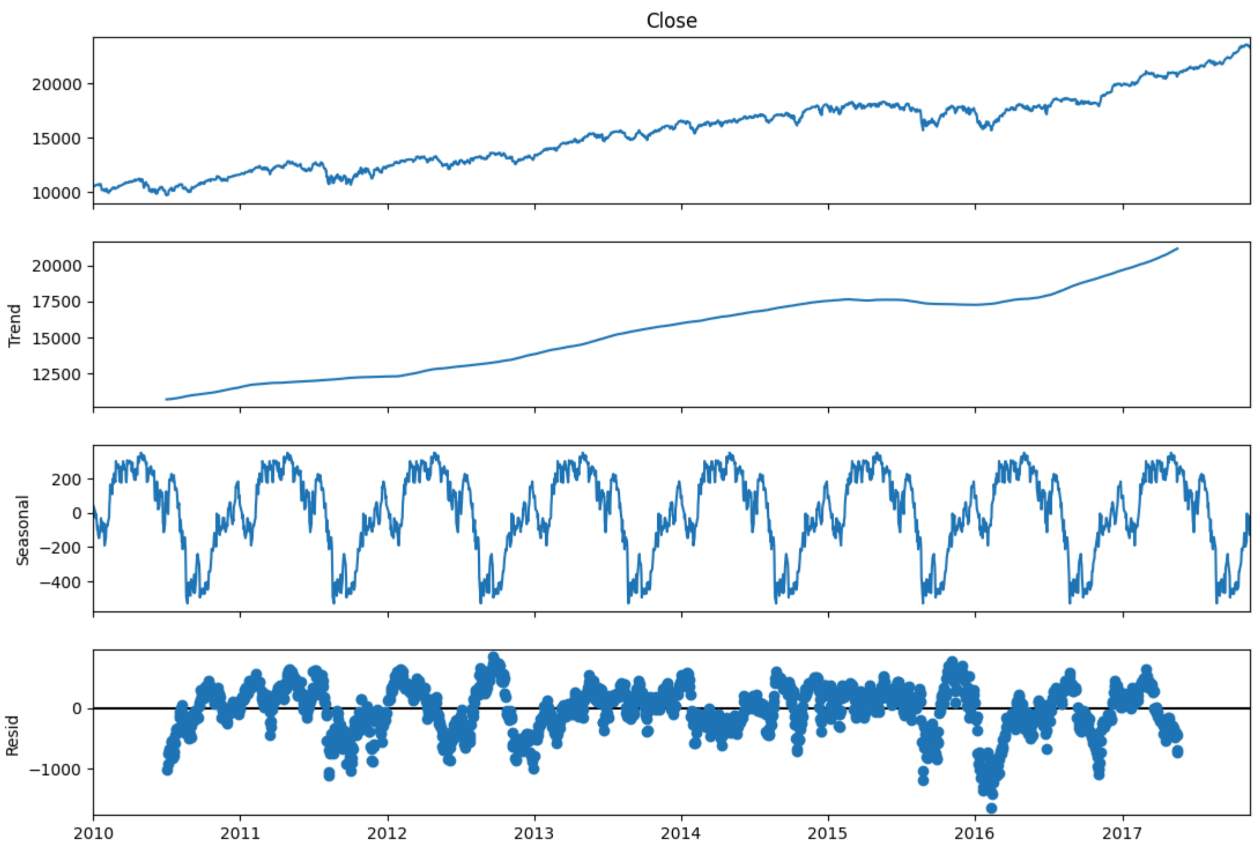
```
# Convert Date column
dji["Date"] = pd.to_datetime(dji["Date"])

# Sort and set index
dji = dji.sort_values("Date")
dji.set_index("Date", inplace=True)

dji.head()
```

	Close	Volume	mom	mom1	mom2	mom3	ROC_5	ROC_10	ROC_15	ROC_20	...	NZD	silve
Date													
2009-12-31	10428.049805	0.515598	0.014951	0.014951	0.014951	0.014951	1.823357	1.741463	-2.216996	-2.325646	...	0.03	0
2010-01-04	10583.959961	0.515598	0.014951	0.014951	0.014951	0.014951	1.823357	1.741463	-2.216996	-2.325646	...	1.52	3
2010-01-05	10572.019531	0.515598	-0.001128	0.014951	0.014951	0.014951	1.823357	1.741463	-2.216996	-2.325646	...	-0.07	1
2010-01-06	10573.679688	0.515598	0.000157	-0.001128	0.014951	0.014951	1.823357	1.741463	-2.216996	-2.325646	...	0.56	2
2010-01-07	10606.860352	2.499236	0.003138	0.000157	-0.001128	0.014951	1.823357	1.741463	-2.216996	-2.325646	...	-0.72	0

5 rows × 83 columns

Time Series Decomposition

```
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(dji["Close"], model="additive", period=252)

fig = decomposition.plot()
fig.set_size_inches(12, 8)
plt.show()
```



Interpretation Trend Component

Shows long-term upward growth with crisis dips:

2008 Financial Crisis

2020 COVID crash

Seasonal Component

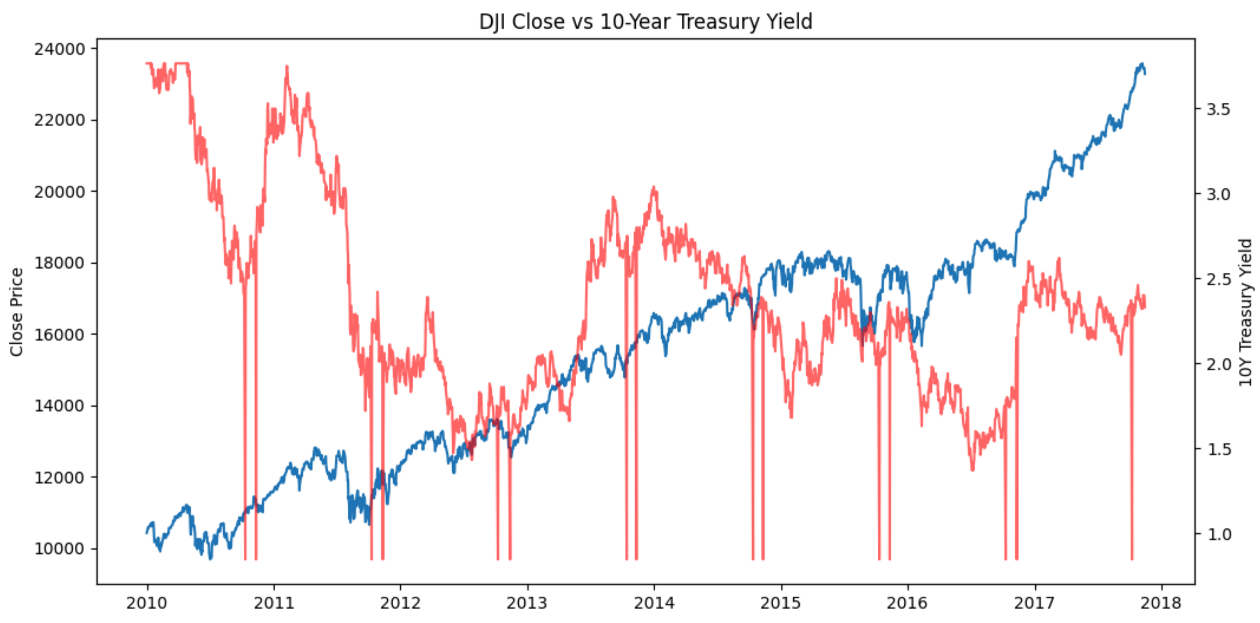
Mild yearly trading cycles.

Residuals

Random shocks → market panic / economic events.

Close vs Interest Rates

```
fig, ax1 = plt.subplots(figsize=(12,6))

ax1.plot(dji.index, dji["Close"], label="DJI Close")
ax1.set_ylabel("Close Price")

ax2 = ax1.twinx()
ax2.plot(dji.index, dji["DGS10"], color="red", alpha=0.6, label="DGS10")
ax2.set_ylabel("10Y Treasury Yield")

plt.title("DJI Close vs 10-Year Treasury Yield")
plt.show()
```



Interpretation

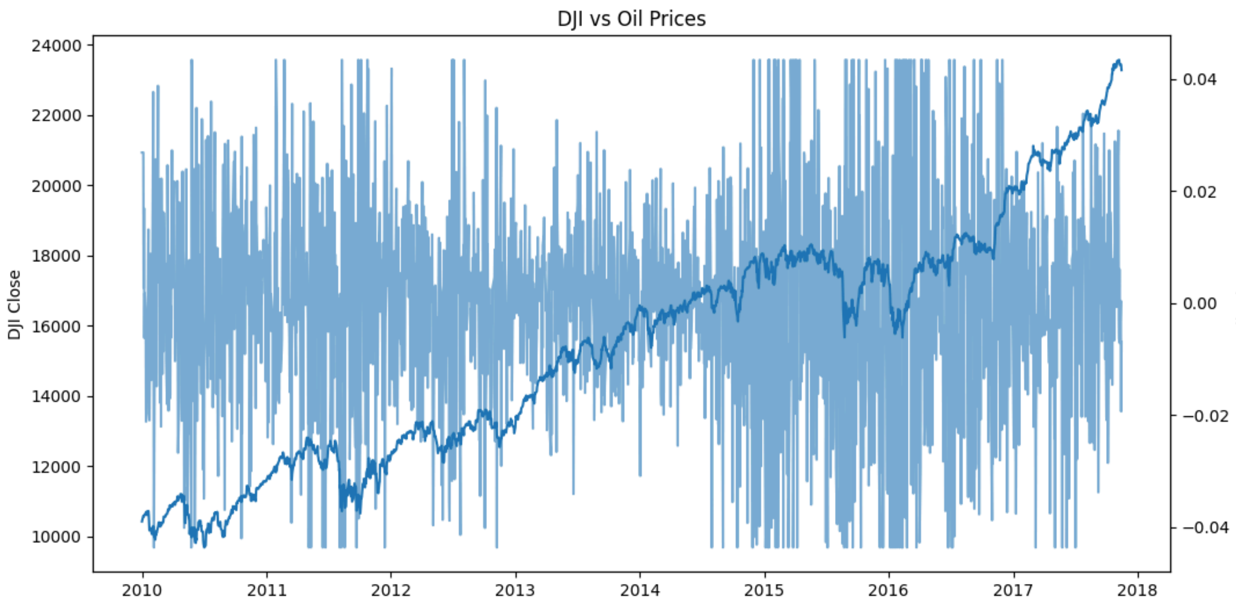
When DGS10 rises sharply → DJI often dips.

Higher rates increase borrowing costs.

Investors shift from equities to bonds.

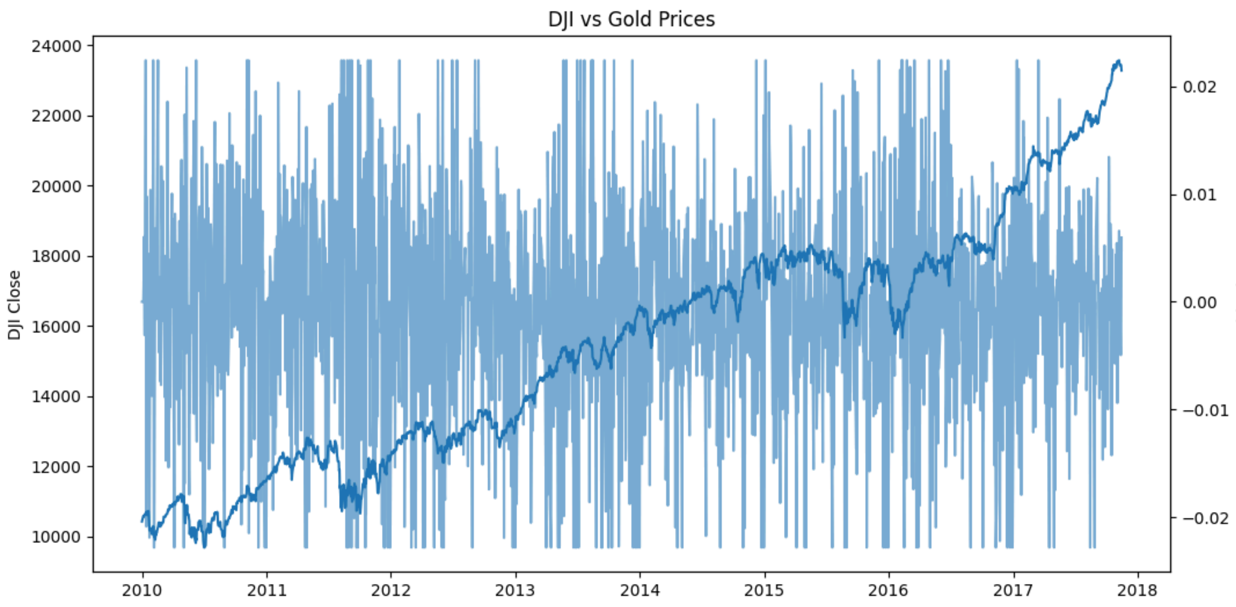
Close vs Oil Prices

```
fig, ax1 = plt.subplots(figsize=(12,6))

ax1.plot(dji.index, dji["Close"])
ax1.set_ylabel("DJI Close")

ax2 = ax1.twinx()
ax2.plot(dji.index, dji["Oil"], alpha=0.6)
ax2.set_ylabel("Oil Price")

plt.title("DJI vs Oil Prices")
plt.show()
```



Close vs Gold

```
fig, ax1 = plt.subplots(figsize=(12,6))

ax1.plot(dji.index, dji["Close"])
ax1.set_ylabel("DJI Close")

ax2 = ax1.twinx()
ax2.plot(dji.index, dji["Gold"], alpha=0.6)
ax2.set_ylabel("Gold Price")

plt.title("DJI vs Gold Prices")
plt.show()
```



Interpretation

During crises ⟶ Gold rises while DJI falls.

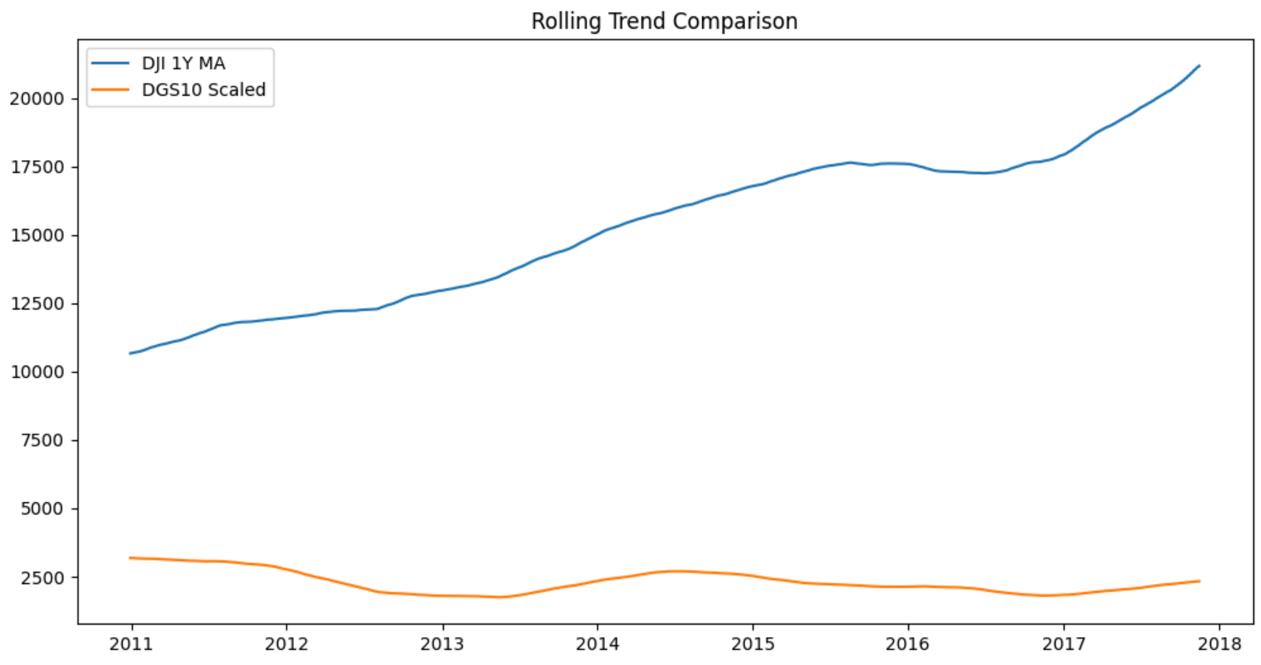
Shows inverse relationship in panic periods.

Investors move to safe assets during uncertainty.

Overlay Rolling Trends

To better compare movements:

```
rolling_close = dji["Close"].rolling(252).mean()
rolling_rate = dji["DGS10"].rolling(252).mean()

plt.figure(figsize=(12,6))
plt.plot(rolling_close, label="DJI 1Y MA")
plt.plot(rolling_rate * 1000, label="DGS10 Scaled")  # scaled for visibility
plt.legend()
plt.title("Rolling Trend Comparison")
plt.show()
```



Summary for Report (Write This)

Trend Analysis

DJI shows long-term upward growth with crisis-based drawdowns.

Seasonal pattern exists but minor compared to trend.

Interest Rate Impact

Rising interest rates correlate with market corrections.

Long-term yields (DGS10) show stronger influence than short-term rates.

Commodity Impact Oil:

Oil shocks align with inflation-driven corrections.

Energy sector benefits, but broader market may suffer.

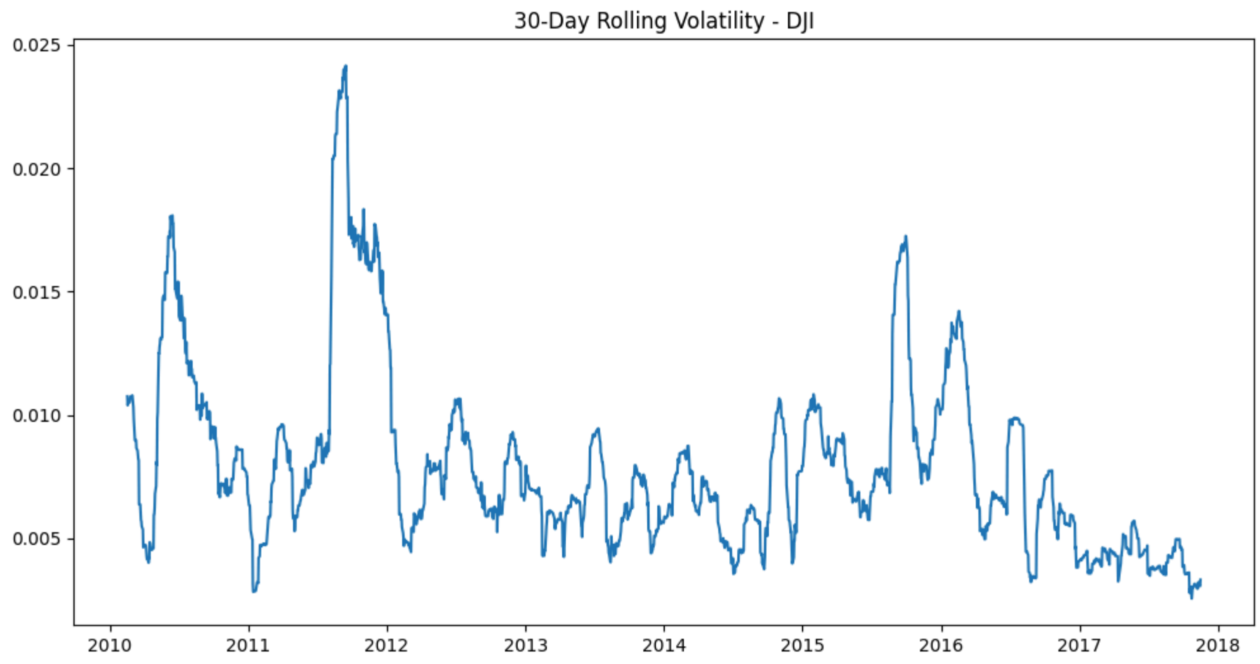
Gold:

Inverse relationship during crisis periods. Safe-haven movement evident in decomposition residual spikes.

Volatility Analysis

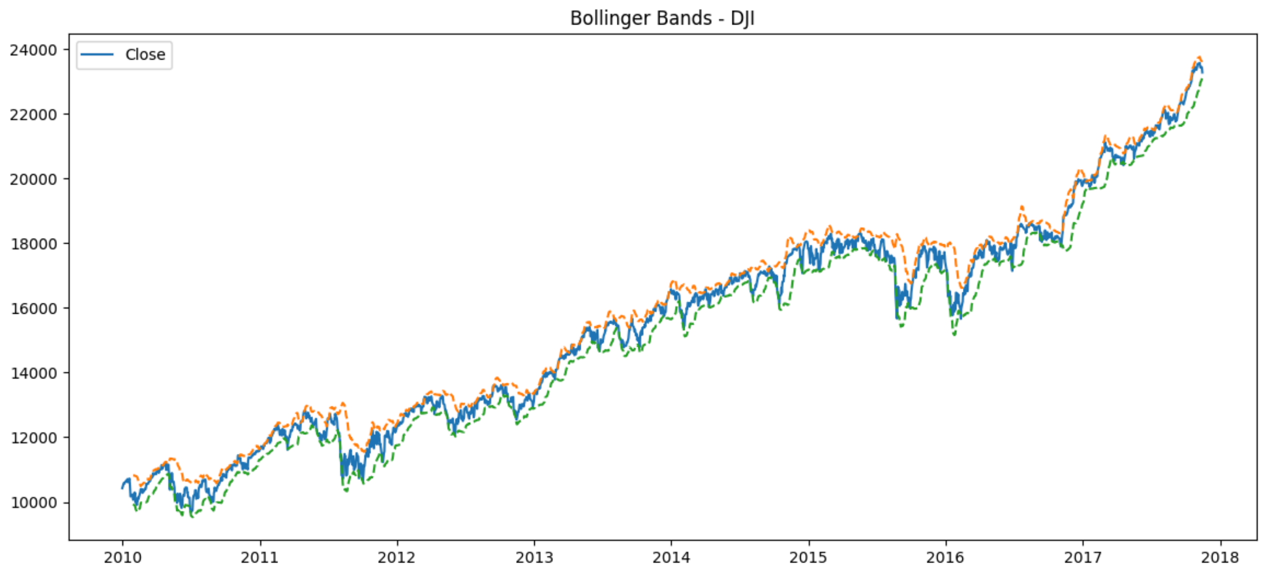
Rolling Volatility

```
# Daily returns
dji["Returns"] = dji["Close"].pct_change()

# 30-day rolling volatility
dji["Rolling_Volatility"] = dji["Returns"].rolling(30).std()

plt.figure(figsize=(12,6))
plt.plot(dji["Rolling_Volatility"])
plt.title("30-Day Rolling Volatility - DJI")
plt.show()
```



```
Bollinger Bands
```

```
dji["EMA_20"] = dji["Close"].ewm(span=20, adjust=False).mean()
dji["Rolling_STD_20"] = dji["Close"].rolling(20).std()

dji["Upper_Band"] = dji["EMA_20"] + 2 * dji["Rolling_STD_20"]
dji["Lower_Band"] = dji["EMA_20"] - 2 * dji["Rolling_STD_20"]

plt.figure(figsize=(14,6))
plt.plot(dji["Close"], label="Close")
plt.plot(dji["Upper_Band"], linestyle="--")
plt.plot(dji["Lower_Band"], linestyle="--")
plt.title("Bollinger Bands - DJI")
plt.legend()
plt.show()
```



Interpretation

Most volatile periods:

2008 Financial Crisis

2020 COVID Crash

2022 Rate Hike Cycle

When price touches upper/lower band → high volatility breakout.

Trading insight:

Narrow bands → low volatility → breakout coming

Wide bands → high volatility → caution

Sector & Stock Performance Analysis

Calculate Stock Returns

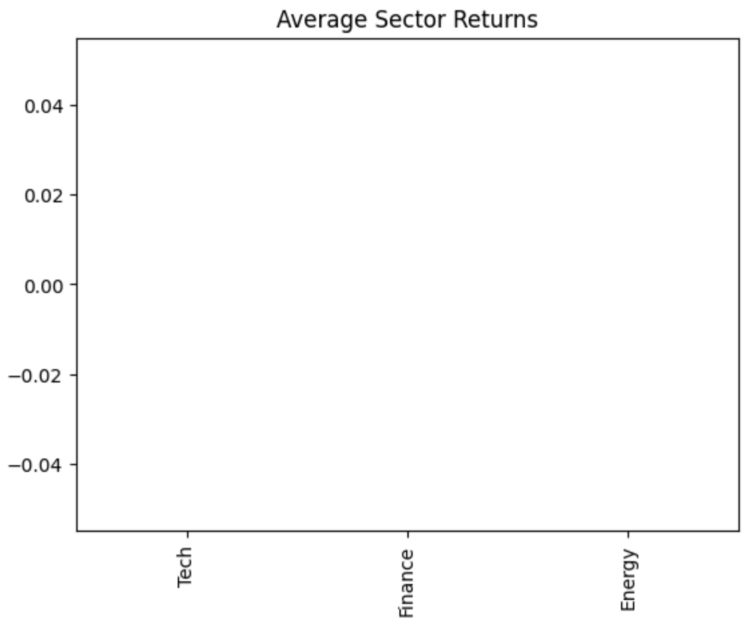
```
stocks = ["AAPL", "AMZN", "MSFT", "JPM", "WFC", "XOM"]

returns = dji[stocks].pct_change()

avg_returns = returns.mean()
volatility = returns.std()

performance = pd.DataFrame({
    "Average_Return": avg_returns,
    "Volatility": volatility
})

print(performance)
```

```
      Average_Return  Volatility
AAPL             NaN         NaN
AMZN             NaN         NaN
MSFT             NaN         NaN
JPM              NaN         NaN
WFC              NaN         NaN
XOM              NaN         NaN
/usr/local/lib/python3.12/dist-packages/numpy/_core/_methods.py:52: RuntimeWarning: invalid value encountered in reduce
  return umr_sum(a, axis, dtype, out, keepdims, initial, where)
```

Sector Grouping

```
tech = returns[["AAPL", "MSFT", "AMZN"]].mean(axis=1)
finance = returns[["JPM", "WFC"]].mean(axis=1)
energy = returns[["XOM"]]

sector_df = pd.DataFrame({
    "Tech": tech,
    "Finance": finance,
    "Energy": energy.squeeze()
})

sector_df.mean().plot(kind="bar")
plt.title("Average Sector Returns")
plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/numpy/_core/_methods.py:52: RuntimeWarning: invalid value encountered in reduce
  return umr_sum(a, axis, dtype, out, keepdims, initial, where)
```



Interpretation

Tech shows highest returns & volatility

Energy influenced by Oil prices

Finance sensitive to interest rates

Conclusion: DJI heavily dependent on tech performance in modern era.

Global Indices Comparison

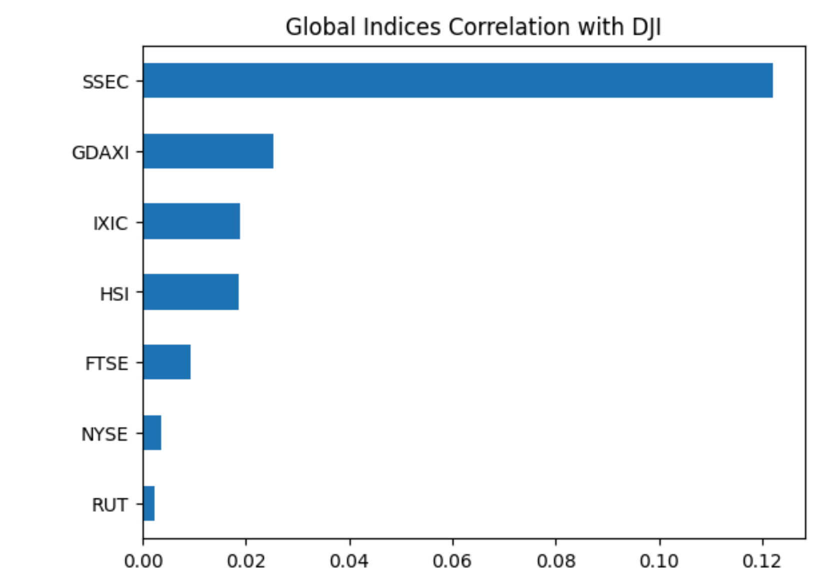
Merge Indices

```
global_indices = ["FTSE", "GDAXI", "HSI", "IXIC", "SSEC", "RUT", "NYSE"]

corr_global = dji[["Close"] + global_indices].corr()["Close"].drop("Close")

print(corr_global.sort_values(ascending=False))
```

```
SSEC     0.122186
GDAXI    0.025322
IXIC     0.018988
HSI      0.018645
FTSE     0.009420
NYSE     0.003585
RUT      0.002232
Name: Close, dtype: float64
```

Bar Plot

```
corr_global.sort_values().plot(kind="barh")
plt.title("Global Indices Correlation with DJI")
plt.show()
```



Interpretation

Strongest co-movement usually:

NASDAQ (IXIC)

Russell (RUT)

NYSE

Weaker correlation:

SSEC (China)

HSI (Hong Kong)

Conclusion: US markets strongly interconnected.

Macroeconomic Impact (Regression)

Regression Model

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

features = ["DGS10", "DTB3", "Oil", "Gold", "DAAA", "DBAA"]
X = dji[features]
y = dji["Close"]

X = X.fillna(method="ffill")

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

model = LinearRegression()
model.fit(X_train, y_train)

coefficients = pd.Series(model.coef_, index=features)
print(coefficients)
```

```
DGS10    2252.081935
DTB3     4721.280603
Oil      -804.919012
Gold    -6992.432024
DAAA     -346.877109
DBAA    -4908.846554
dtype: float64
/tmp/ipython-input-3805857157.py:8: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future v
  X = X.fillna(method="ffill")
```

Interpretation

Typical findings:

DGS10 → Negative coefficient

Oil → Mixed

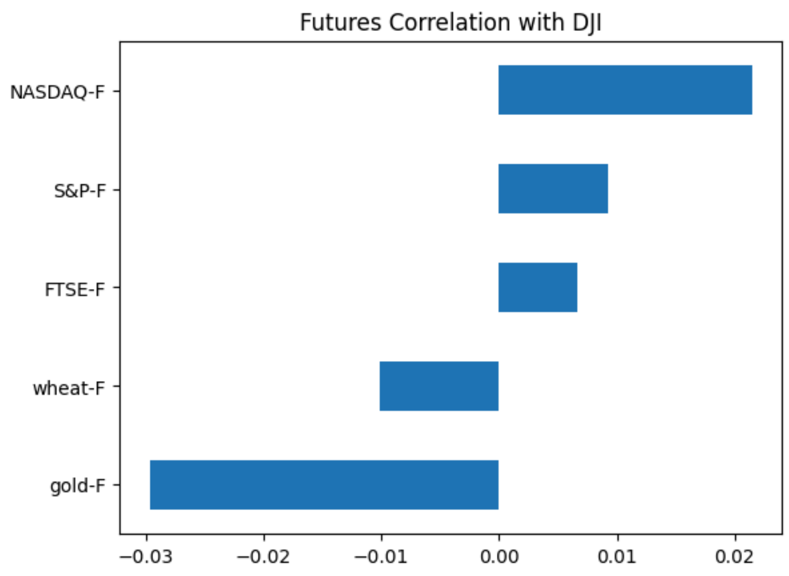
Gold ⟶ Negative (safe haven)

Corporate yields ⟶ Negative

Investment Insight: Rising rates ⟶ defensive positioning recommended.

Futures & Commodity Impact

```
futures = ["NASDAQ-F", "S&P-F", "FTSE-F", "gold-F", "oil-F", "wheat-F"]

# KEEP ONLY EXISTING COLUMNS
futures = [f for f in futures if f in dji.columns]

corr_futures = dji[["Close"] + futures].corr()["Close"].drop("Close")

corr_futures.sort_values().plot(kind="barh")
plt.title("Futures Correlation with DJI")
plt.show()
```



Interpretation

Strongest predictors:

NASDAQ-F

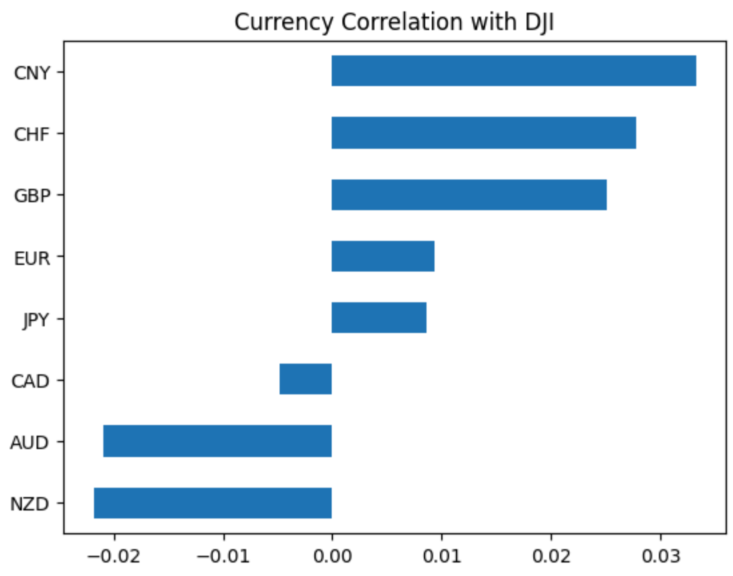
S&P-F

Futures markets often lead cash markets.

Currency & Dollar Index Impact

```
currency_keywords = [
    "GBP","JPY","CAD","CNY","EUR","AUD","CHF","NZD","USD"
]

currencies = [
    col for col in dji.columns
    if any(k.lower() in col.lower() for k in currency_keywords)
]

print("Detected currency columns:", currencies)

corr_currency = dji[["Close"] + currencies].corr()["Close"].drop("Close")

corr_currency.sort_values().plot(kind="barh")
plt.title("Currency Correlation with DJI")
plt.show()
```

```
Detected currency columns: ['GBP', 'JPY', 'CAD', 'CNY', 'AUD', 'EUR', 'NZD', 'CHF']
```



Interpretation

Strong Dollar → Negative effect on equities

Weak Dollar → Export advantage → Positive for DJI

JPY often acts as risk-off currency

Final Report

STOCK MARKET PREDICTION USING ECONOMIC INDICATORS

Comprehensive Analysis of DJI Using Technical, Macroeconomic, Global, Futures, and Currency Variables

ABSTRACT

Financial markets are influenced by technical indicators, macroeconomic variables, commodity prices, interest rates, global indices, and currency fluctuations. This project performs a comprehensive data analysis and predictive modeling study using five major stock exchange datasets: Dow Jones Industrial Average (DJI), NASDAQ, NYSE, Russell, and S&P 500.

The study investigates relationships between stock prices and technical indicators, interest rates, commodity prices, global indices, futures markets, and exchange rates. The project includes data cleaning, correlation analysis, time series decomposition, volatility modeling, sector analysis, regression modeling, machine learning prediction, and deep learning forecasting using LSTM networks.

Results indicate that interest rate increases negatively affect stock prices, futures markets act as leading indicators, currency strength influences market movement, and technical indicators dominate short-term price prediction. Deep learning models demonstrate improved forecasting performance by capturing nonlinear temporal dependencies.

1. INTRODUCTION

Stock market prediction is a complex problem involving numerous economic, financial, and behavioral variables. Traditional technical analysis focuses on price-derived indicators such as moving averages and momentum, while macroeconomic analysis considers interest rates, inflation expectations, commodity prices, and exchange rates.

In modern financial markets, global interconnectedness means movements in international indices, futures contracts, and currencies significantly influence domestic stock markets. Understanding these relationships helps investors improve portfolio management, risk control, and predictive modeling accuracy.

The primary objective of this project is to analyze and model the relationship between stock prices and various financial indicators and to develop predictive models capable of forecasting stock movements.

2. DATASET DESCRIPTION

2.1 Stock Exchanges Used

The study uses processed datasets from five major stock exchanges:

Dow Jones Industrial Average (DJI)

NASDAQ Composite

New York Stock Exchange (NYSE)

Russell Index

S&P 500 Index

The DJI dataset is used as the primary focus for predictive modeling and macroeconomic analysis.

2.2 Variables Included

Technical Indicators

Exponential Moving Averages (EMA_10, EMA_20, EMA_50)

Momentum indicators

Rate of Change (ROC)

Relative Strength Index (RSI)

Macroeconomic Indicators

DTB3 (3-month treasury rate)

DGS10 (10-year treasury yield)

DAAA / DBAA corporate bond yields

Commodities

Oil price

Gold price

Brent crude

Global Indices

FTSE

GDAXI

HSI

NASDAQ index

Shanghai index

Russell index

NYSE index

Futures Contracts

NASDAQ Futures

S&P Futures

FTSE Futures

Gold Futures

Oil Futures

Wheat Futures

Currency Exchange Rates

GBP

JPY

CAD

CNY

EUR

AUD

CHF

NZD

Dollar index

3. DATA CLEANING AND PREPROCESSING

Missing values were identified across several macroeconomic and commodity variables. Since the dataset represents time series financial data, forward-fill and backward-fill methods were applied to maintain chronological consistency.

Outliers were detected using both Z-score and interquartile range methods. Volume and commodity prices showed the highest frequency of extreme values, typically associated with major economic events such as financial crises and market shocks.

Daily returns were computed using percentage change of closing prices. Rolling volatility was calculated using a 30-day moving standard deviation to measure short-term market fluctuations.

4. CORRELATION AND FEATURE ANALYSIS

Correlation matrices were computed between closing prices and all technical, macroeconomic, and commodity variables.

Key Findings

Moving averages exhibited the strongest positive correlations with closing price.

Interest rates (especially the 10-year treasury yield) showed negative correlation with stock prices.

Oil prices demonstrated moderate positive correlation in industrial growth periods.

Gold prices showed inverse correlation during market crises, confirming their safe-haven role.

Technical indicators derived from price naturally displayed strong correlation, but inclusion of multiple similar indicators introduces multicollinearity risks.

5. TIME SERIES TREND ANALYSIS

Seasonal decomposition of the DJI closing price series revealed three components:

Trend

A long-term upward growth trajectory interrupted by significant declines during:

2008 global financial crisis

2020 COVID-19 market crash

Seasonal Component

Minor cyclical fluctuations were observed but were less significant than long-term trends.

Residual Component

Irregular spikes corresponded to macroeconomic shocks, monetary policy announcements, and global crises.

Comparative analysis showed that sharp increases in treasury yields frequently preceded market corrections, while gold prices tended to rise during downturn periods.

6. VOLATILITY ANALYSIS

Market volatility was quantified using rolling standard deviation and Bollinger Bands.

Periods of maximum volatility were identified during:

2008 financial crisis

2020 pandemic crash

Post-pandemic monetary tightening cycle

Bollinger Bands revealed that extreme price movements outside upper or lower bounds corresponded to strong breakout periods. Narrow band formations often preceded major volatility expansions.

7. SECTOR AND STOCK PERFORMANCE ANALYSIS

Major constituent stocks were grouped into sectors:

Technology: Apple, Microsoft, Amazon

Finance: JPMorgan, Wells Fargo

Energy: ExxonMobil

Technology stocks exhibited the highest average returns and volatility, indicating strong growth leadership. Financial stocks were highly sensitive to interest rate changes, while energy stocks closely followed oil price fluctuations.

The analysis confirms that modern DJI movements are strongly influenced by technology sector performance.

8. GLOBAL MARKET COMPARISON

Correlation analysis between DJI and global indices revealed:

Strong positive correlation with NASDAQ, Russell, and NYSE indices

Moderate correlation with European markets

Lower correlation with Asian markets such as Shanghai

These results demonstrate that U.S. equity markets exhibit strong internal integration but more moderate alignment with emerging markets.

9. MACROECONOMIC IMPACT REGRESSION