```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import io
import seaborn as sns
```

```
df = pd.read_csv('drive/MyDrive/gold.csv')
```

```
df.head()
```

	Date	Open	High	Low	Close	Volume	Currency
0	2000-01-04	289.5	289.5	280.0	283.7	21621	USD
1	2000-01-05	283.7	285.0	281.0	282.1	25448	USD
2	2000-01-06	281.6	282.8	280.2	282.4	19055	USD
3	2000-01-07	282.5	284.5	282.0	282.9	11266	USD
4	2000-01-10	282.4	283.9	281.8	282.7	30603	USD

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

```python
# 1. Basic Data Inspection
print("="*50)
print("DATA SHAPE:")
print(f"Rows: {df.shape[0]}, Columns: {df.shape[1]}")
print("\nCOLUMN NAMES:")
print(df.columns.tolist())
print("\nDATA TYPES:")
print(df.dtypes)
```

```
==================================================
DATA SHAPE:
Rows: 5703, Columns: 7

COLUMN NAMES:
['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Currency']

DATA TYPES:
Date         object
Open        float64
High        float64
Low         float64
Close       float64
Volume        int64
Currency     object
dtype: object
```

```
# 2. Sample Data Inspection
print("\n" + "="*50)
print("FIRST 5 RECORDS:")
print(df.head())
print("\nRANDOM 5 RECORDS:")
print(df.sample(5))
print("\nBASIC STATISTICS:")
print(df.describe())
```

```
==================================================
FIRST 5 RECORDS:
         Date   Open   High    Low  Close  Volume Currency
0  2000-01-04  289.5  289.5  280.0  283.7   21621      USD
1  2000-01-05  283.7  285.0  281.0  282.1   25448      USD
2  2000-01-06  281.6  282.8  280.2  282.4   19055      USD
3  2000-01-07  282.5  284.5  282.0  282.9   11266      USD
4  2000-01-10  282.4  283.9  281.8  282.7   30603      USD

RANDOM 5 RECORDS:
            Date    Open    High     Low   Close  Volume Currency
100   2000-05-26   270.7   273.3   270.1   272.2   30192      USD
798   2003-03-18   337.5   340.5   334.3   337.7   29388      USD
491   2001-12-20   276.0   277.0   275.8   276.5   11083      USD
3229  2012-11-06  1685.4  1720.9  1683.5  1715.0  187222      USD
4220  2016-10-13  1257.1  1263.9  1254.7  1257.6  150097      USD

BASIC STATISTICS:
              Open         High          Low        Close         Volume
count  5703.000000  5703.000000  5703.000000  5703.000000    5703.000000
mean   1040.382816  1048.339181  1031.863169  1040.298282  139141.669297
std     518.733377   522.353946   514.455903   518.524020  102537.449058
min     256.600000   259.400000   255.100000   256.600000       0.000000
25%     459.850000   463.900000   457.450000   460.500000   52938.500000
50%    1188.800000  1198.000000  1179.700000  1188.700000  126006.000000
75%    1381.400000  1392.750000  1368.100000  1383.050000  193109.000000
max    2076.400000  2089.200000  2049.000000  2069.400000  816531.000000
```

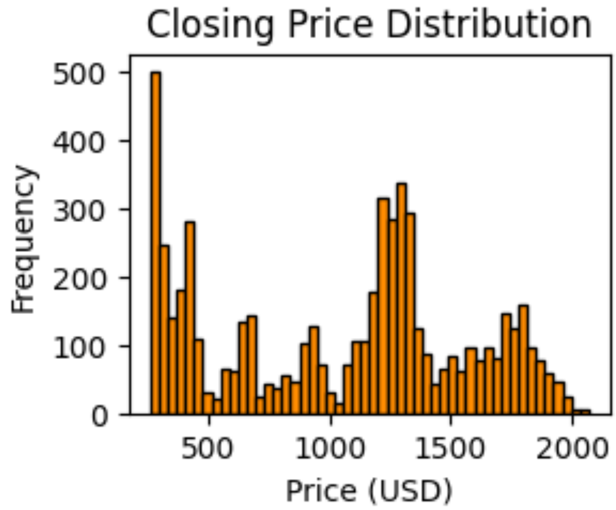
```
# 3. Data Quality Checks
print("\n" + "="*50)
print("MISSING VALUES:")
print(df.isnull().sum())
print("\nDUPLICATED ROWS:", df.duplicated().sum())
```

```
==================================================
MISSING VALUES:
Date        0
Open        0
High        0
Low         0
Close       0
Volume      0
Currency    0
dtype: int64

DUPLICATED ROWS: 0
```

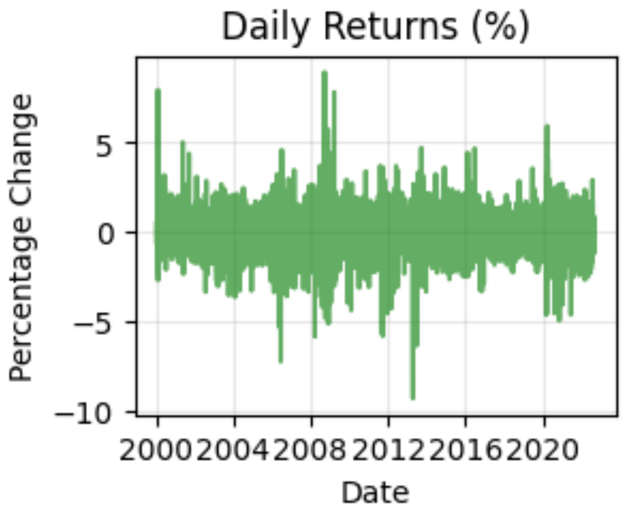
```
# 4. Data Visualization
plt.figure(figsize=(15, 10))

# Price Trends
plt.subplot(2, 2, 1)
plt.plot(pd.to_datetime(df['Date']), df['Close'], color='royalblue')
plt.title('Gold Closing Price Trend (2000-2006)')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.grid(alpha=0.3)
```

```
# Daily Price Distribution
plt.subplot(2, 2, 2)
plt.hist(df['Close'], bins=50, color='darkorange', edgecolor='black')
plt.title('Closing Price Distribution')
plt.xlabel('Price (USD)')
plt.ylabel('Frequency')
```

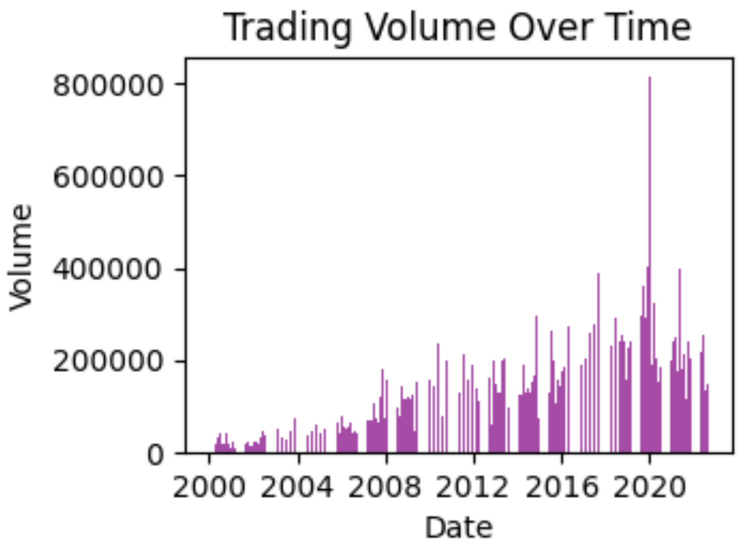
```
Text(0, 0.5, 'Frequency')
```




```
# Volatility Analysis
daily_returns = df['Close'].pct_change() * 100
plt.subplot(2, 2, 3)
plt.plot(pd.to_datetime(df['Date']), daily_returns, color='forestgreen', alpha=
plt.title('Daily Returns (%)')
plt.xlabel('Date')
plt.ylabel('Percentage Change')
plt.grid(alpha=0.3)
```



```
# Volume Analysis
plt.subplot(2, 2, 4)
plt.bar(pd.to_datetime(df['Date']), df['Volume'], color='purple', alpha=0.7)
plt.title('Trading Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Volume')

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




```
# 5. Correlation Analysis
corr_matrix = df[['Open','High','Low','Close','Volume']].corr()
print("\n" + "="*50)
print("CORRELATION MATRIX:")
print(corr_matrix)
```

```
==================================================
CORRELATION MATRIX:
            Open      High       Low     Close    Volume
Open    1.000000  0.999879  0.999825  0.999740  0.692123
High    0.999879  1.000000  0.999778  0.999861  0.693861
Low     0.999825  0.999778  1.000000  0.999893  0.688983
Close   0.999740  0.999861  0.999893  1.000000  0.690534
Volume  0.692123  0.693861  0.688983  0.690534  1.000000
```

```
# 6. Time Period Analysis
df['Year'] = pd.to_datetime(df['Date']).dt.year
annual_avg = df.groupby('Year')['Close'].mean()

plt.figure(figsize=(10, 5))
annual_avg.plot(kind='bar', color='teal')
plt.title('Average Annual Closing Price')
plt.xlabel('Year')
plt.ylabel('Average Price (USD)')
plt.xticks(rotation=0)
plt.grid(axis='y', alpha=0.3)
plt.show()
```

Data Cleaning

```
# Handle zero-volume days (market holidays)
zero_volume_days = df[df['Volume'] == 0]
print(f"\nZero-volume days found: {len(zero_volume_days)}")
df = df[df['Volume'] > 0]  # Remove zero-volume days
```

```
Zero-volume days found: 10
```

```
# Outlier detection using IQR
plt.figure(figsize=(15, 8))
```

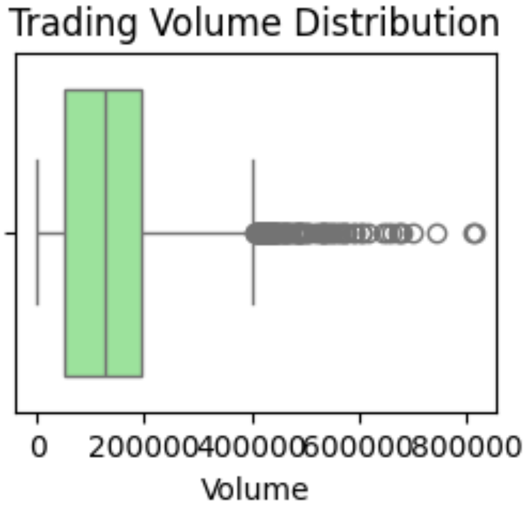
```
<Figure size 1500x800 with 0 Axes>
<Figure size 1500x800 with 0 Axes>
```

```
# Price Outliers
plt.subplot(2, 2, 1)
sns.boxplot(x=df['Close'], color='skyblue')
plt.title('Closing Price Distribution')
plt.xlabel('Price (USD)')
```

```
Text(0.5, 0, 'Price (USD)')
```

```
# Volume Outliers
plt.subplot(2, 2, 2)
sns.boxplot(x=df['Volume'], color='lightgreen')
plt.title('Trading Volume Distribution')
plt.xlabel('Volume')
```

```
Text(0.5, 0, 'Volume')
```

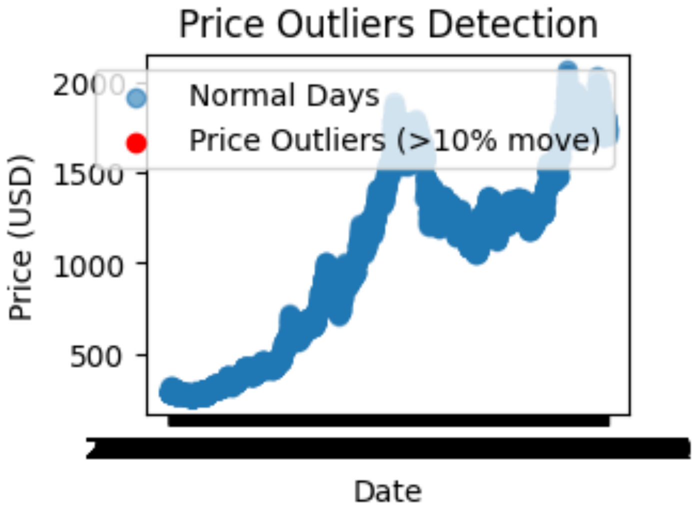



```
# IQR Calculation for Volume
Q1 = df['Volume'].quantile(0.25)
Q3 = df['Volume'].quantile(0.75)
IQR = Q3 - Q1
volume_outlier_threshold = Q3 + (1.5 * IQR)
print(f"\nVolume outlier threshold (IQR method): {volume_outlier_threshold:.0f}
```

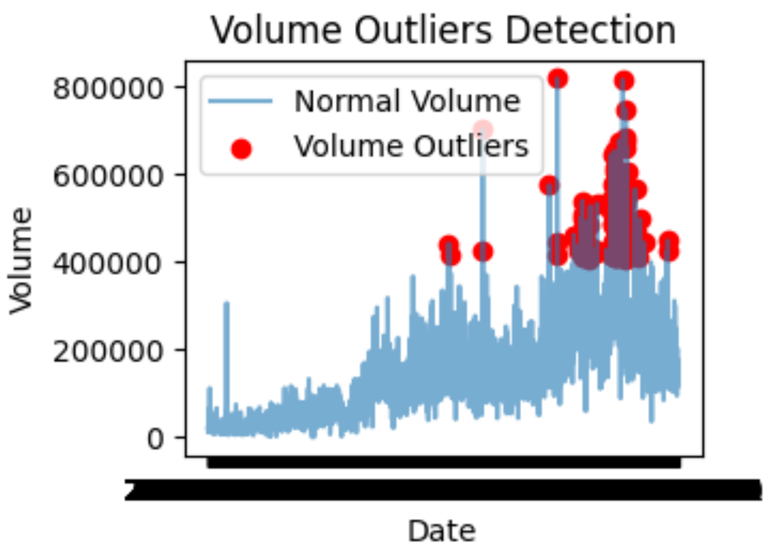
```
Volume outlier threshold (IQR method): 403300
```

```python
# Identify and visualize price outliers
df['Returns'] = df['Close'].pct_change() * 100
price_outliers = df[np.abs(df['Returns']) > 10]  # Days with >10% price moves

plt.subplot(2, 2, 3)
plt.scatter(df['Date'], df['Close'], alpha=0.6, label='Normal Days')
plt.scatter(price_outliers['Date'], price_outliers['Close'],
            color='red', label='Price Outliers (>10% move)')
plt.title('Price Outliers Detection')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7af432a48310>
```



```
# Volume outliers visualization
volume_outliers = df[df['Volume'] > volume_outlier_threshold]

plt.subplot(2, 2, 4)
plt.plot(df['Date'], df['Volume'], alpha=0.6, label='Normal Volume')
plt.scatter(volume_outliers['Date'], volume_outliers['Volume'],
            color='red', label='Volume Outliers')
plt.title('Volume Outliers Detection')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.legend()

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



```
# Data validation checks
print("\nDATA VALIDATION CHECKS:")
# Check for high>low inconsistencies
high_low_errors = df[df['High'] < df['Low']]
print(f"Rows with High < Low: {len(high_low_errors)}")
df = df[df['High'] >= df['Low']]  # Remove invalid rows

# Check for open/close outside high/low range
range_errors = df[
    (df['Open'] > df['High']) |
    (df['Open'] < df['Low']) |
    (df['Close'] > df['High']) |
    (df['Close'] < df['Low'])
]
print(f"Rows with prices outside daily range: {len(range_errors)}")
```

```
DATA VALIDATION CHECKS:
Rows with High < Low: 0
Rows with prices outside daily range: 0
```

```
# Add volatility-adjusted z-scores
df['Volatility'] = df['Close'].rolling(window=30).std()
df['Price_ZScore'] = (df['Close'] - df['Close'].rolling(window=30).mean()) / df

# Final cleaned dataset
print(f"\nFinal dataset shape: {df.shape}")
```

```
Final dataset shape: (5693, 11)
```

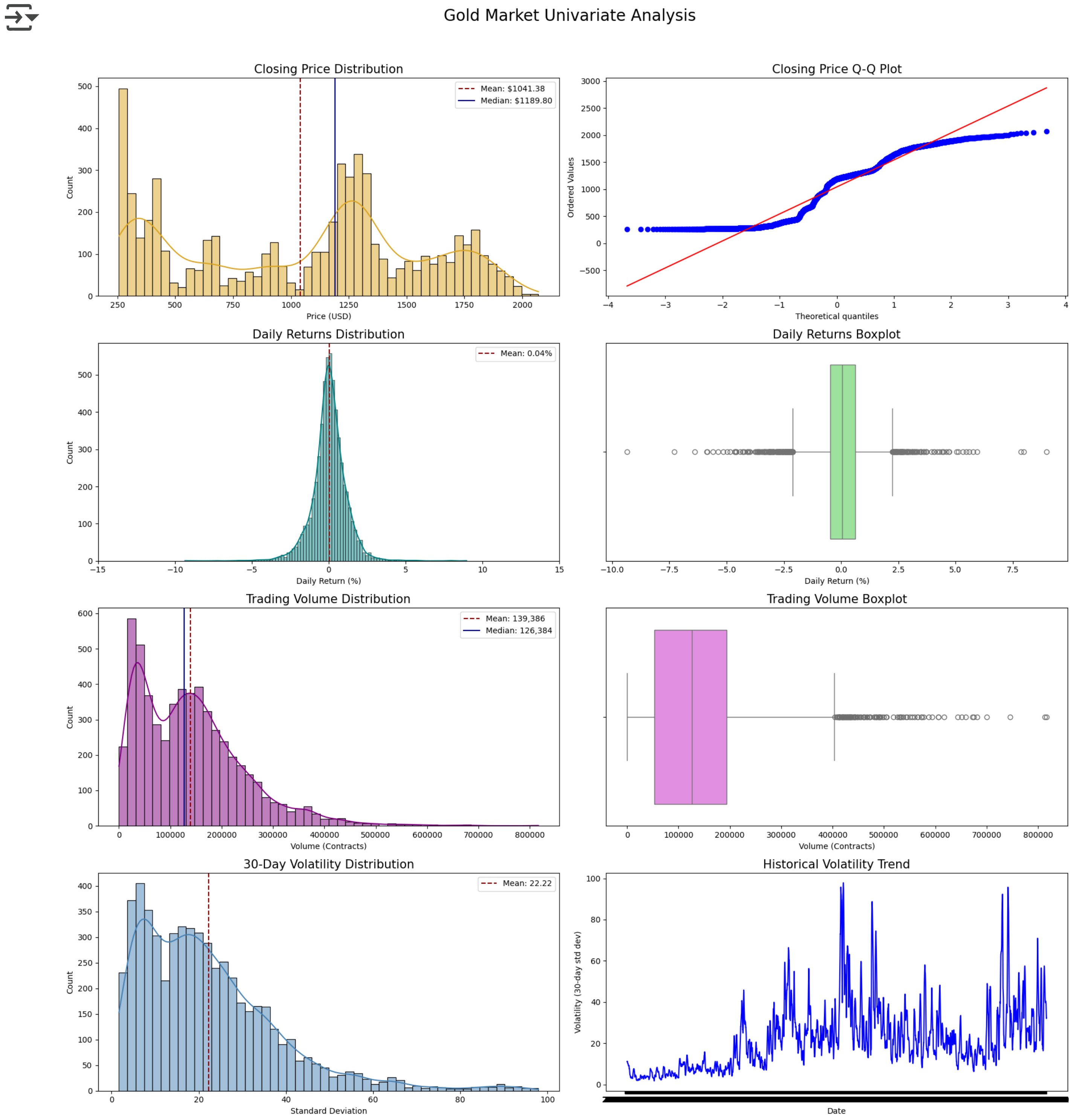
Univariate Analysis

```
import seaborn as sns
import scipy.stats as stats
```

```
# Set up subplots
fig, axes = plt.subplots(4, 2, figsize=(18, 20))
fig.suptitle('Gold Market Univariate Analysis', fontsize=20, y=0.98)

# 1. Closing Price Analysis
# -------------------------
sns.histplot(df['Close'], kde=True, ax=axes[0,0], color='goldenrod', bins=50)
axes[0,0].axvline(df['Close'].mean(), color='darkred', linestyle='--', label=f'
```

```
axes[0,0].axvline(df['Close'].median(), color='navy', linestyle='-', label=f'Me
axes[0,0].set_title('Closing Price Distribution', fontsize=15)
axes[0,0].set_xlabel('Price (USD)')
axes[0,0].legend()

stats.probplot(df['Close'], plot=axes[0,1])
axes[0,1].set_title('Closing Price Q-Q Plot', fontsize=15)

# 2. Daily Returns Analysis
# -------------------------
sns.histplot(df['Returns'].dropna(), kde=True, ax=axes[1,0], color='teal', bins
axes[1,0].axvline(df['Returns'].mean(), color='darkred', linestyle='--', label=
axes[1,0].set_title('Daily Returns Distribution', fontsize=15)
axes[1,0].set_xlabel('Daily Return (%)')
axes[1,0].set_xlim([-15, 15])
axes[1,0].legend()

sns.boxplot(x=df['Returns'], ax=axes[1,1], color='lightgreen')
axes[1,1].set_title('Daily Returns Boxplot', fontsize=15)
axes[1,1].set_xlabel('Daily Return (%)')

# 3. Trading Volume Analysis
# -------------------------
sns.histplot(df['Volume'], kde=True, ax=axes[2,0], color='purple', bins=50)
axes[2,0].axvline(df['Volume'].mean(), color='darkred', linestyle='--', label=f
axes[2,0].axvline(df['Volume'].median(), color='navy', linestyle='-', label=f'M
axes[2,0].set_title('Trading Volume Distribution', fontsize=15)
axes[2,0].set_xlabel('Volume (Contracts)')
axes[2,0].legend()

sns.boxplot(x=df['Volume'], ax=axes[2,1], color='violet')
axes[2,1].set_title('Trading Volume Boxplot', fontsize=15)
axes[2,1].set_xlabel('Volume (Contracts)')

# 4. Volatility Analysis
# -------------------------
sns.histplot(df['Volatility'].dropna(), kde=True, ax=axes[3,0], color='steelblu
axes[3,0].axvline(df['Volatility'].mean(), color='darkred', linestyle='--', lab
axes[3,0].set_title('30-Day Volatility Distribution', fontsize=15)
axes[3,0].set_xlabel('Standard Deviation')
axes[3,0].legend()

sns.lineplot(x=df['Date'], y=df['Volatility'], ax=axes[3,1], color='blue')
axes[3,1].set_title('Historical Volatility Trend', fontsize=15)
axes[3,1].set_xlabel('Date')
axes[3,1].set_ylabel('Volatility (30-day std dev)')

plt.tight_layout(rect=[0, 0, 1, 0.96])
```

```
plt.show()
```


Gold Market Univariate Analysis
Closing Price Distribution
Mean: $1041.38
Median: $1189.80
Price (USD)
Count
Closing Price Q-Q Plot
Ordered Values
Theoretical quantiles
Daily Returns Distribution
Mean: 0.04%
Daily Return (%)
Daily Returns Boxplot
Trading Volume Distribution
Mean: 139,386
Median: 126,384
Volume (Contracts)
Trading Volume Boxplot
30-Day Volatility Distribution
Mean: 22.22
Standard Deviation
Historical Volatility Trend
Volatility (30-day std dev)
Date

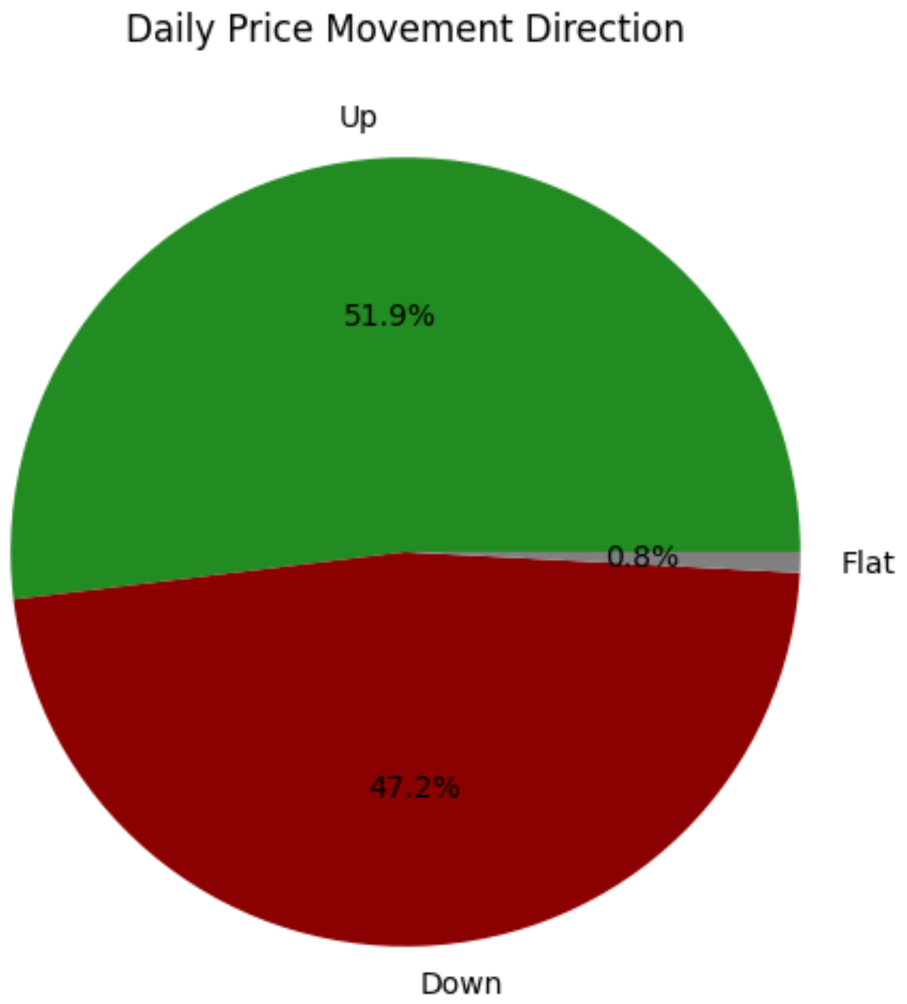
```
# Numerical Summary Statistics
num_cols = ['Close', 'Returns', 'Volume', 'Volatility']
summary = df[num_cols].describe().T
summary['skewness'] = [df[c].skew() for c in num_cols]
summary['kurtosis'] = [df[c].kurtosis() for c in num_cols]

print("="*80)
print("COMPREHENSIVE NUMERICAL SUMMARY STATISTICS")
print(summary)
```

```
================================================================================
COMPREHENSIVE NUMERICAL SUMMARY STATISTICS
             count           mean            std        min           25%
Close       5693.0    1041.375988     518.023065  256.600000    465.800000
Returns     5692.0       0.037634       1.101965   -9.344612     -0.482957
Volume      5693.0  139386.077639  102461.342795    1.000000  53268.000000
Volatility  5664.0      22.218251      16.112562    1.717085      9.725781

                      50%            75%            max  skewness  kurtosis
Close         1189.800000    1383.100000    2069.400000 -0.087728 -1.246077
Returns          0.038450       0.606358       8.968610 -0.131925  5.503014
Volume      126384.000000  193281.000000  816531.000000  1.234134  2.572365
Volatility      19.069546      29.992401      97.892747  1.459736  2.950868
```

```
# Target Variable Analysis (Closing Price)
print("\n" + "="*80)
print("TARGET VARIABLE ANALYSIS (CLOSING PRICE)")
print(f"- Long-term trend: ${df['Close'].iloc[0]:.2f} → ${df['Close'].iloc[-1]:
print(f"- Maximum single-day gain: {df['Returns'].max():.2f}%")
print(f"- Maximum single-day loss: {df['Returns'].min():.2f}%")
print(f"- Days with >5% moves: {len(df[df['Returns'].abs() > 5])} ({len(df[df['
```

```
================================================================================
TARGET VARIABLE ANALYSIS (CLOSING PRICE)
- Long-term trend: $283.70 → $1709.30
- Maximum single-day gain: 8.97%
- Maximum single-day loss: -9.34%
- Days with >5% moves: 18 (0.3% of days)
```

```
# Price Movement Analysis
plt.figure(figsize=(10, 6))
direction = df['Returns'].apply(lambda x: 'Up' if x > 0 else 'Down' if x < 0 el
direction.value_counts().plot.pie(autopct='%1.1f%%', colors=['forestgreen', 'da
plt.title('Daily Price Movement Direction')
plt.ylabel('')
plt.show()
```



Bivariate Analysis

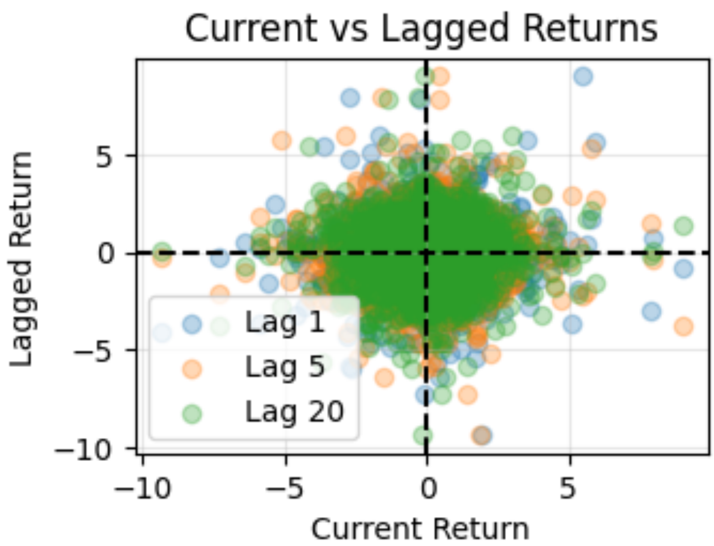
```
# 1. Correlation Analysis
plt.figure(figsize=(14, 12))
corr = df[['Close', 'Volume', 'Volatility', 'Returns']].corr()

# Pearson Correlation
plt.subplot(2, 2, 1)
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', vmin=-1, vmax=1)
plt.title('Pearson Correlation Matrix')

# Spearman Correlation
plt.subplot(2, 2, 2)
spearman_corr = df[['Close', 'Volume', 'Volatility', 'Returns']].corr(method='s
sns.heatmap(spearman_corr, annot=True, cmap='viridis', fmt='.2f', vmin=-1, vmax
plt.title('Spearman Rank Correlation Matrix')
```

```
Text(0.5, 1.0, 'Spearman Rank Correlation Matrix')
```

```
plt.subplot(2, 2, 3)
scatter = sns.scatterplot(
    x=df['Close'],
    y=df['Volume'],
    alpha=0.6,
    hue=df['Returns'].abs(),
    palette='viridis',
    size=df['Volatility'],
    legend='full'  # Ensure legend shows both color and size info
)
plt.title('Price vs Volume (Colored by Absolute Return)')
plt.xlabel('Price (USD)')
plt.ylabel('Volume (Contracts)')
plt.yscale('log')


# Add label to the hue portion of the legend
scatter.legend_.texts[0].set_text('Absolute Return (%)')
plt.show()
```

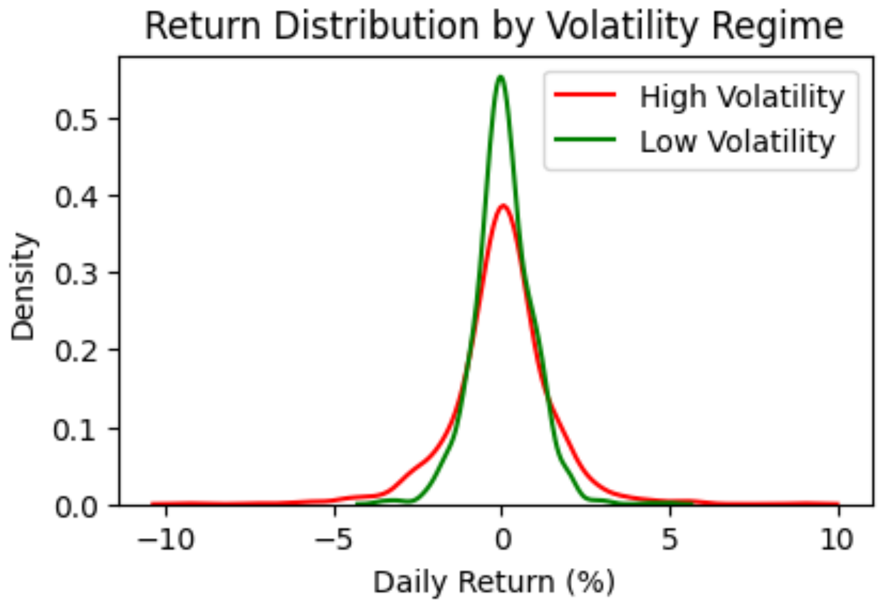
```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: Use
  fig.canvas.print_figure(bytes_io, **kw)
```

```
# 3. Lagged Return Analysis
plt.subplot(2, 2, 4)
for lag in [1, 5, 20]:
    df[f'Return_lag_{lag}'] = df['Returns'].shift(lag)
    plt.scatter(df['Returns'], df[f'Return_lag_{lag}'], alpha=0.3, label=f'Lag

plt.axhline(0, color='black', linestyle='--')
plt.axvline(0, color='black', linestyle='--')
plt.xlabel('Current Return')
plt.ylabel('Lagged Return')
plt.title('Current vs Lagged Returns')
plt.legend()
plt.grid(alpha=0.3)

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



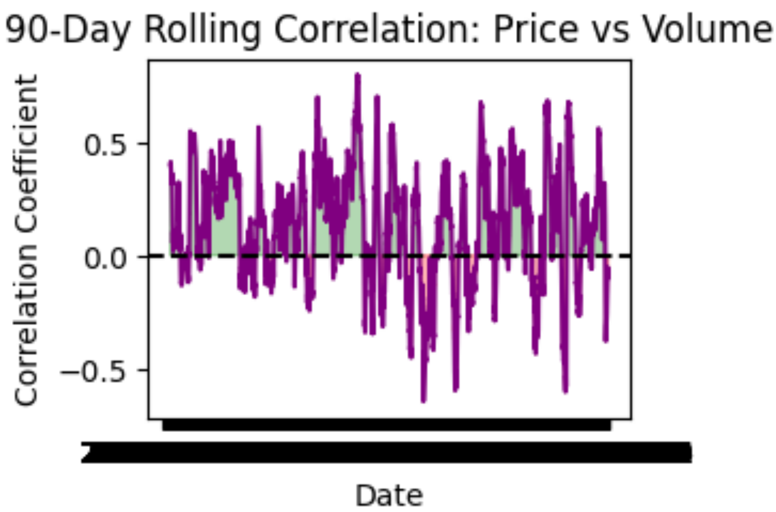
```
import matplotlib.gridspec as gridspec

fig = plt.figure(figsize=(10, 6))
gs = gridspec.GridSpec(2, 2)  # Creates a 2x2 grid layout

ax2 = plt.subplot(gs[1])  # Targets the second cell (indexing starts at 0)

# Conditional Returns
ax2 = plt.subplot(gs[1])
high_vol = df[df['Volatility'] > df['Volatility'].quantile(0.75)]
low_vol = df[df['Volatility'] < df['Volatility'].quantile(0.25)]
sns.kdeplot(high_vol['Returns'], color='red', label='High Volatility')
sns.kdeplot(low_vol['Returns'], color='green', label='Low Volatility')
plt.title('Return Distribution by Volatility Regime')
plt.xlabel('Daily Return (%)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7af42a7e9b10>
```

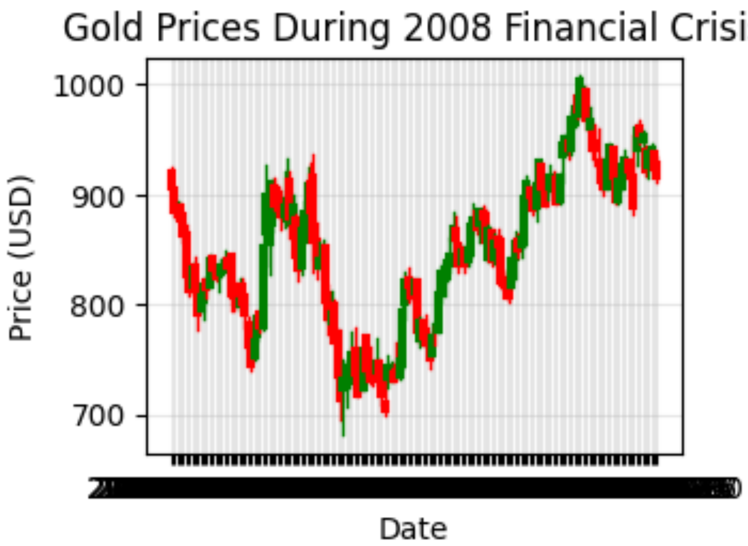


```
# Rolling Correlation: Price vs Volume
ax3 = plt.subplot(gs[2])
df['Rolling_Corr'] = df['Close'].rolling(window=90).corr(df['Volume'])
plt.plot(df['Date'], df['Rolling_Corr'], color='purple')
plt.axhline(0, color='black', linestyle='--')
plt.fill_between(df['Date'], df['Rolling_Corr'], 0, where=df['Rolling_Corr']>=0
                 color='green', alpha=0.3, interpolate=True)
plt.fill_between(df['Date'], df['Rolling_Corr'], 0, where=df['Rolling_Corr']<0,
                 color='red', alpha=0.3, interpolate=True)
plt.title('90-Day Rolling Correlation: Price vs Volume')
plt.xlabel('Date')
plt.ylabel('Correlation Coefficient')
```

```
Text(0, 0.5, 'Correlation Coefficient')
```



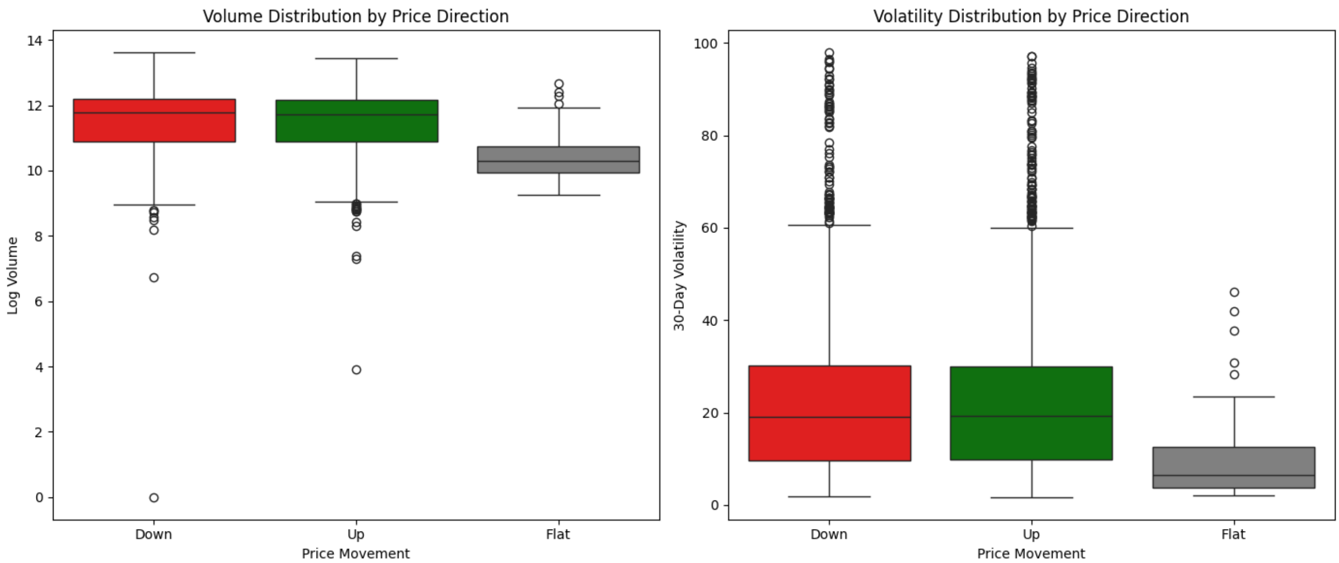
```
# Candlestick Chart for Key Period (2008 Financial Crisis)
ax4 = plt.subplot(gs[3])
crisis = df[(df['Date'] > '2008-08-01') & (df['Date'] < '2009-03-31')]
for i in range(len(crisis)):
    color = 'green' if crisis['Close'].iloc[i] >= crisis['Open'].iloc[i] else '
    plt.plot([crisis['Date'].iloc[i], crisis['Date'].iloc[i]],
             [crisis['Low'].iloc[i], crisis['High'].iloc[i]],
             color=color, linewidth=1)
    plt.plot([crisis['Date'].iloc[i], crisis['Date'].iloc[i]],
             [crisis['Open'].iloc[i], crisis['Close'].iloc[i]],
             color=color, linewidth=3)

plt.title('Gold Prices During 2008 Financial Crisis')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.grid(alpha=0.3)

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




```
# 5. Directional Movement Analysis
plt.figure(figsize=(14, 6))

# Up/Down Days vs Volume
plt.subplot(1, 2, 1)
direction = np.sign(df['Returns']).map({1: 'Up', -1: 'Down', 0: 'Flat'})
sns.boxplot(x=direction, y=np.log(df['Volume']), palette=['red', 'green', 'gray
plt.title('Volume Distribution by Price Direction')
plt.xlabel('Price Movement')
plt.ylabel('Log Volume')
```

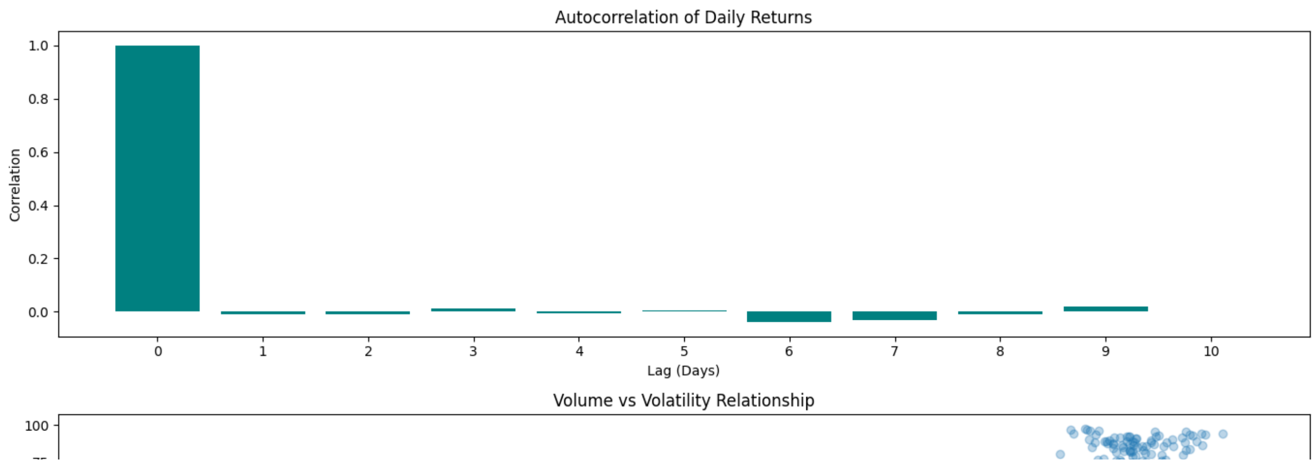
```
# Up/Down Days vs Volatility
plt.subplot(1, 2, 2)
sns.boxplot(x=direction, y=df['Volatility'], palette=['red', 'green', 'gray'])
plt.title('Volatility Distribution by Price Direction')
plt.xlabel('Price Movement')
plt.ylabel('30-Day Volatility')

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

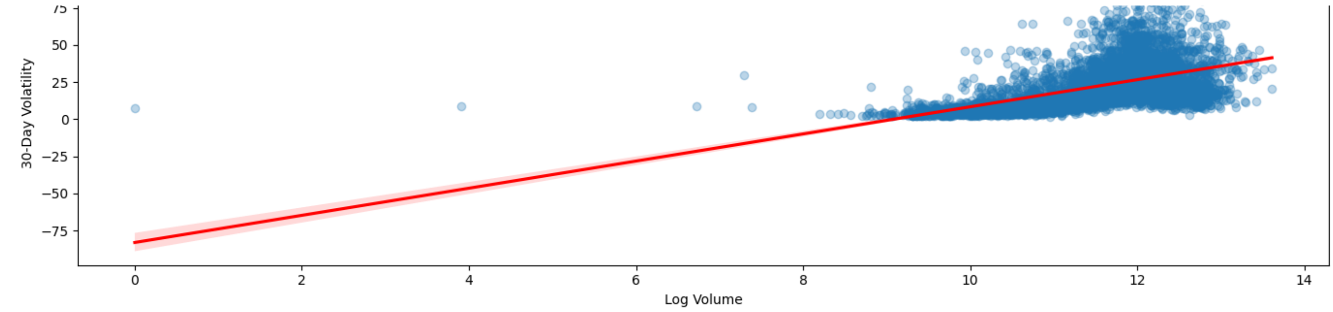
```
/tmp/ipython-input-33-4220807140.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.boxplot(x=direction, y=np.log(df['Volume']), palette=['red', 'green',
/tmp/ipython-input-33-4220807140.py:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed

  sns.boxplot(x=direction, y=df['Volatility'], palette=['red', 'green', 'gr
```

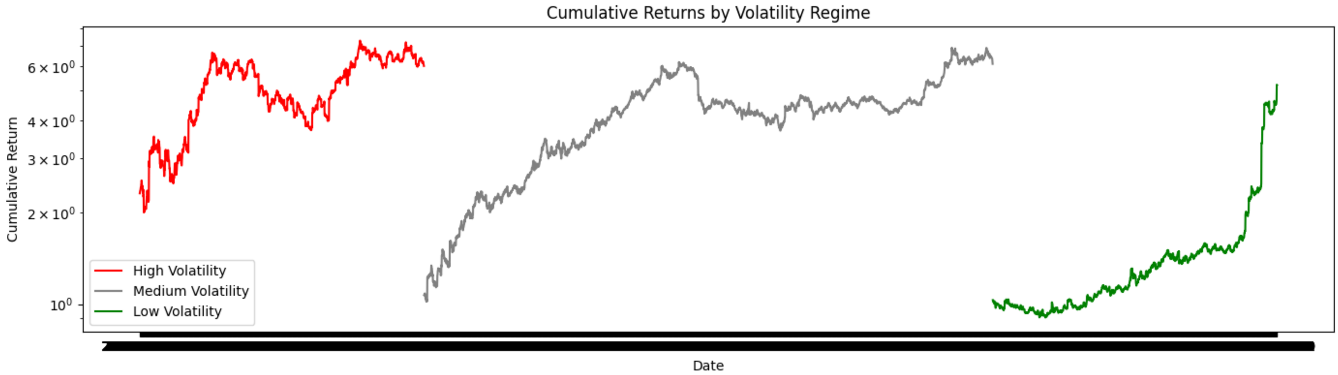



```
# 6. Advanced Analysis: Lead-Lag Relationships
fig, ax = plt.subplots(3, 1, figsize=(14, 12))
```

```
# Cross-correlation analysis
max_lag = 10
correlations = [df['Returns'].corr(df['Returns'].shift(lag)) for lag in range(m
ax[0].bar(range(max_lag+1), correlations, color='teal')
ax[0].set_title('Autocorrelation of Daily Returns')
ax[0].set_xlabel('Lag (Days)')
ax[0].set_ylabel('Correlation')
ax[0].set_xticks(range(max_lag+1))

# Volume-Volatility Relationship
sns.regplot(x=np.log(df['Volume']), y=df['Volatility'], ax=ax[1],
            scatter_kws={'alpha':0.3}, line_kws={'color':'red'})
ax[1].set_title('Volume vs Volatility Relationship')
ax[1].set_xlabel('Log Volume')
ax[1].set_ylabel('30-Day Volatility')

# Cumulative Returns by Volatility Regime
df['Cumulative_Return'] = (1 + df['Returns']/100).cumprod()
for regime, color in [('High', 'red'), ('Medium', 'gray'), ('Low', 'green')]:
    if regime == 'High':
        subset = df[df['Volatility'] > df['Volatility'].quantile(0.75)]
    elif regime == 'Low':
        subset = df[df['Volatility'] < df['Volatility'].quantile(0.25)]
    else:
        subset = df[(df['Volatility'] >= df['Volatility'].quantile(0.25)) &
                    (df['Volatility'] <= df['Volatility'].quantile(0.75))]
    ax[2].plot(subset['Date'], subset['Cumulative_Return'], color=color, label=

ax[2].set_title('Cumulative Returns by Volatility Regime')
ax[2].set_xlabel('Date')
ax[2].set_ylabel('Cumulative Return')
ax[2].set_yscale('log')
ax[2].legend()

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





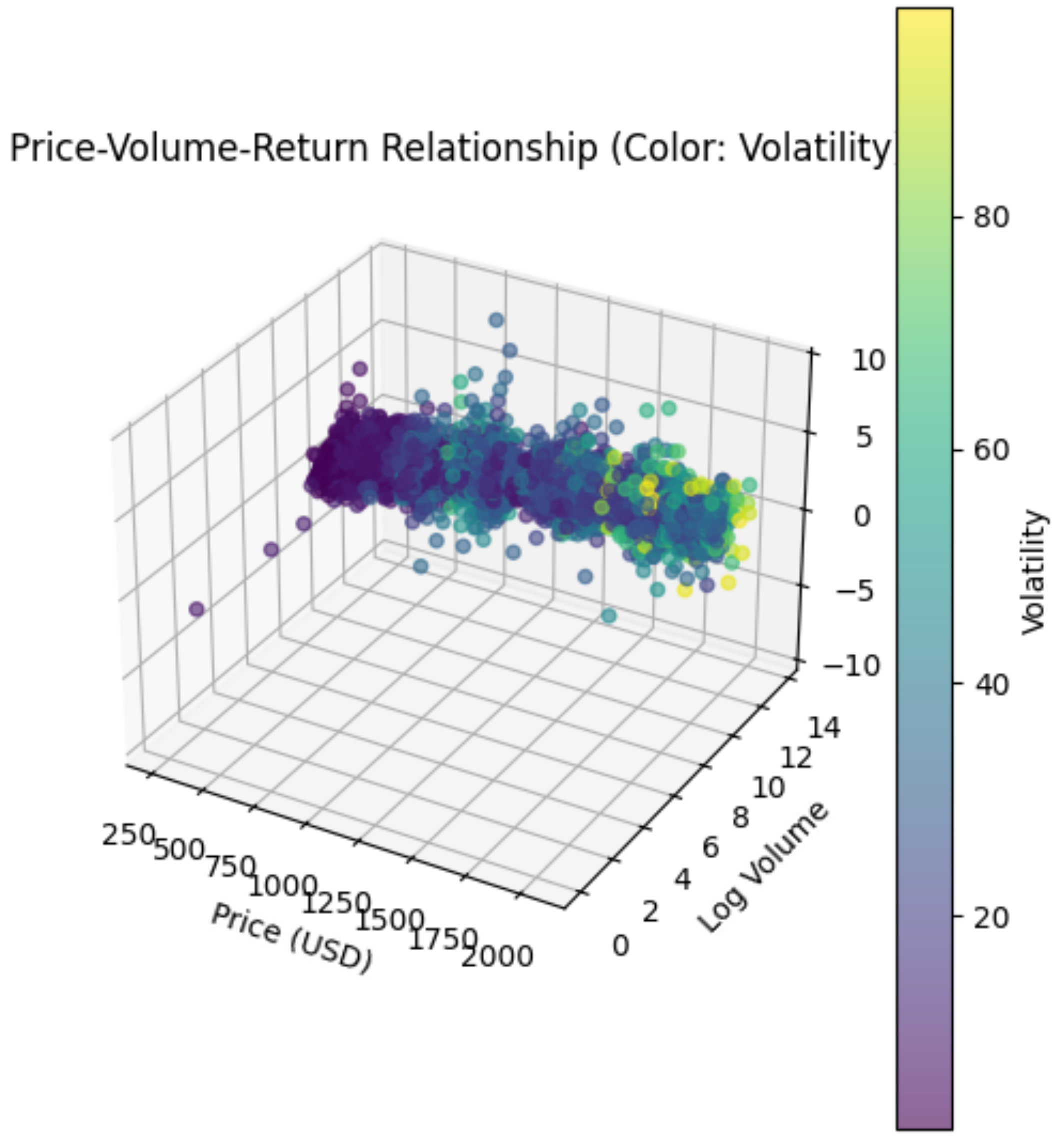


Multivariate Analysis

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from pandas.plotting import parallel_coordinates
import plotly.express as px


df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
# Create time-based features
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df['Weekday'] = df['Date'].dt.weekday
df['Quarter'] = df['Date'].dt.quarter
```

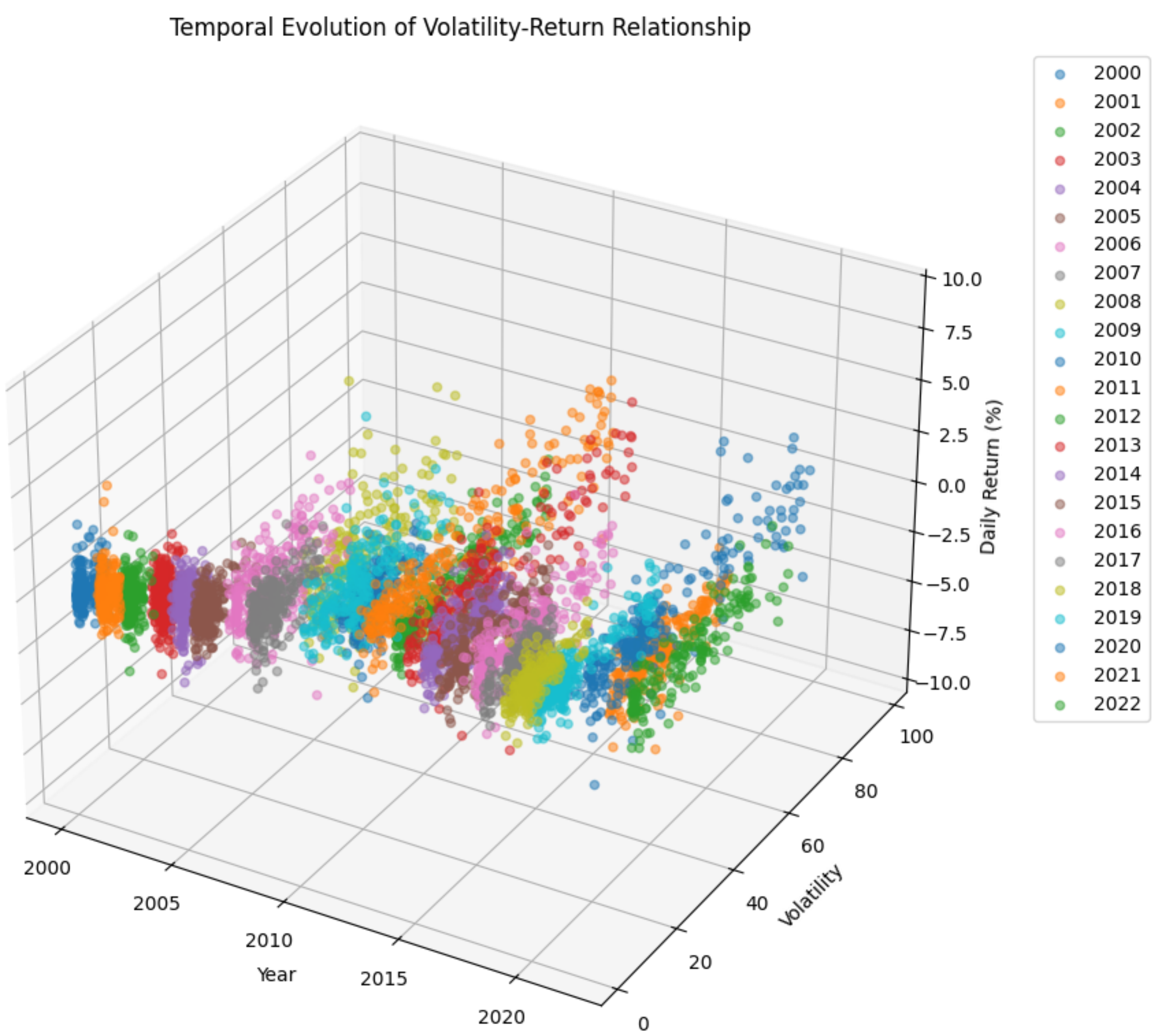
```
# 1. 3D Visualization of Key Relationships
fig = plt.figure(figsize=(20, 15))

# Price-Volume-Returns
ax1 = fig.add_subplot(231, projection='3d')
scatter1 = ax1.scatter(df['Close'], np.log(df['Volume']), df['Returns'],
                       c=df['Volatility'], cmap='viridis', alpha=0.6)
ax1.set_xlabel('Price (USD)')
ax1.set_ylabel('Log Volume')
ax1.set_zlabel('Daily Return (%)')
ax1.set_title('Price-Volume-Return Relationship (Color: Volatility)')
fig.colorbar(scatter1, ax=ax1, label='Volatility')
```

```
<matplotlib.colorbar.Colorbar at 0x7af3eda1d990>
```




```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

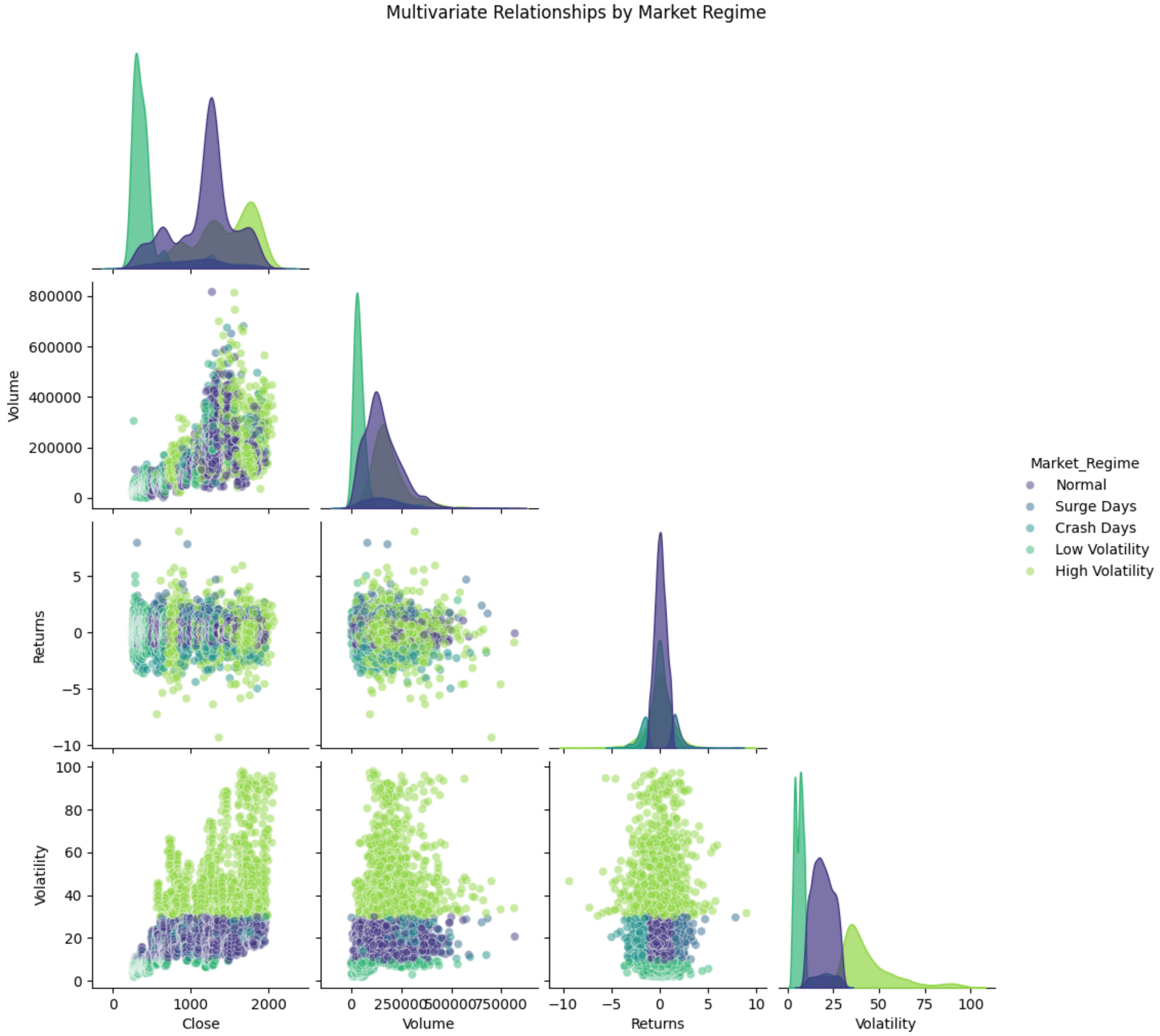
```
# Create the figure and 3D subplot
fig = plt.figure(figsize=(12, 8))
ax2 = fig.add_subplot(111, projection='3d')  # or (232) if part of a larger gri

# Ensure Year is numeric
df['Year'] = df['Year'].astype(int)

# Plot each year's data
for year in sorted(df['Year'].unique()):
    year_data = df[df['Year'] == year]
    ax2.scatter(
        year_data['Year'],
        year_data['Volatility'],
        year_data['Returns'],
        alpha=0.5,
        label=str(year)
    )

# Set axis labels and title
ax2.set_xlabel('Year')
ax2.set_ylabel('Volatility')
ax2.set_zlabel('Daily Return (%)')
ax2.set_title('Temporal Evolution of Volatility-Return Relationship')

# Optional: limit legend to fewer entries
# handles, labels = ax2.get_legend_handles_labels()
# ax2.legend(handles[:5], labels[:5], loc='best')

# Show full legend or turn off if too crowded
ax2.legend(loc='upper left', bbox_to_anchor=(1.05, 1))

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



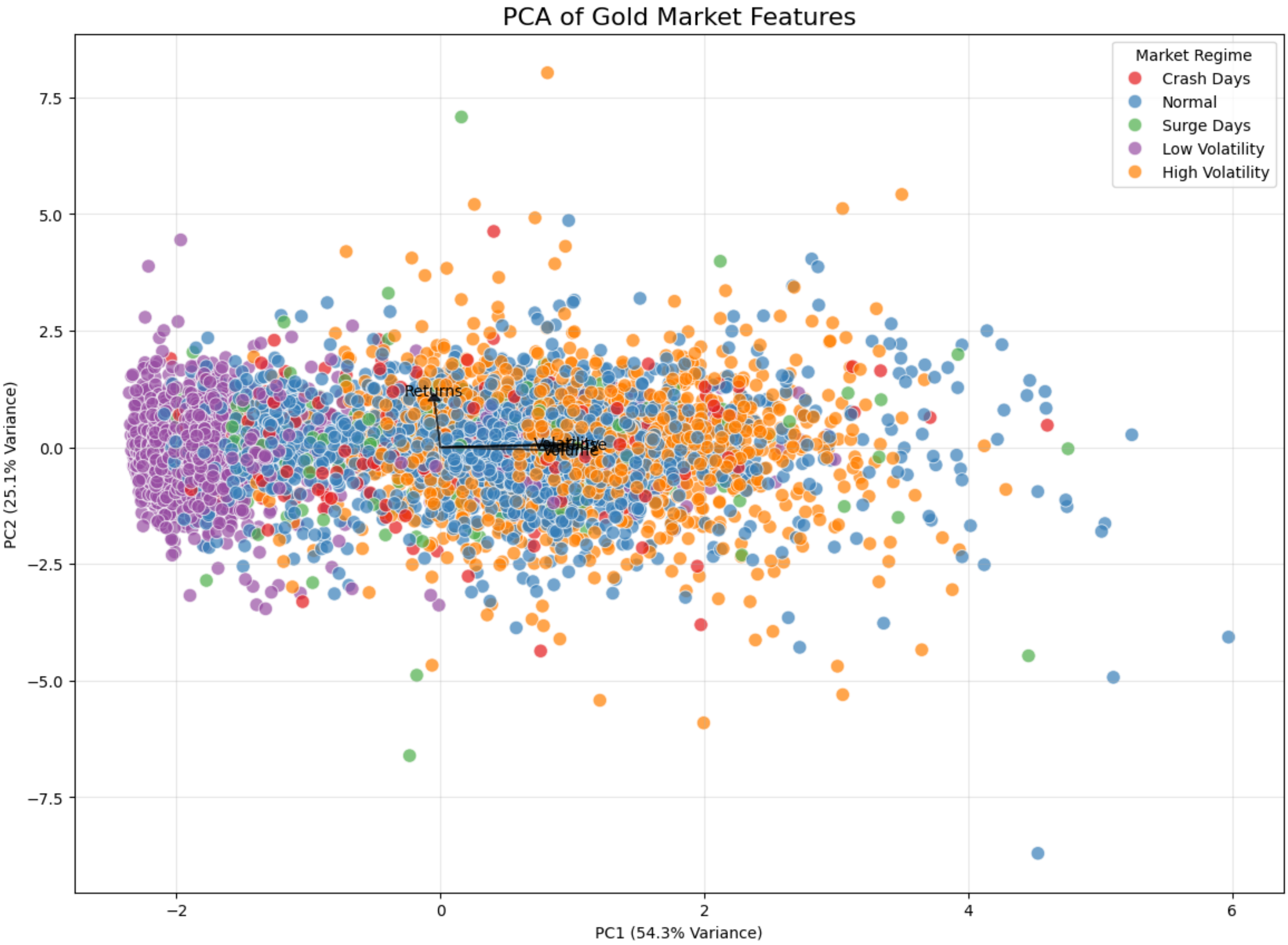

```
# 2. Pairplot with Market Regimes
df['Market_Regime'] = np.select(
    [
        df['Volatility'] < df['Volatility'].quantile(0.25),
```

```
        df['Volatility'] > df['Volatility'].quantile(0.75),
        df['Returns'] < df['Returns'].quantile(0.1),
        df['Returns'] > df['Returns'].quantile(0.9)
    ],
    ['Low Volatility', 'High Volatility', 'Crash Days', 'Surge Days'],
    default='Normal'
)

plt.figure(figsize=(12, 10))
sns.pairplot(df[['Close', 'Volume', 'Returns', 'Volatility', 'Market_Regime']],
             hue='Market_Regime', palette='viridis', corner=True, diag_kind='kc
             plot_kws={'alpha': 0.5}, diag_kws={'alpha': 0.7})
plt.suptitle('Multivariate Relationships by Market Regime', y=1.02)
plt.show()
```

```
<Figure size 1200x1000 with 0 Axes>
```



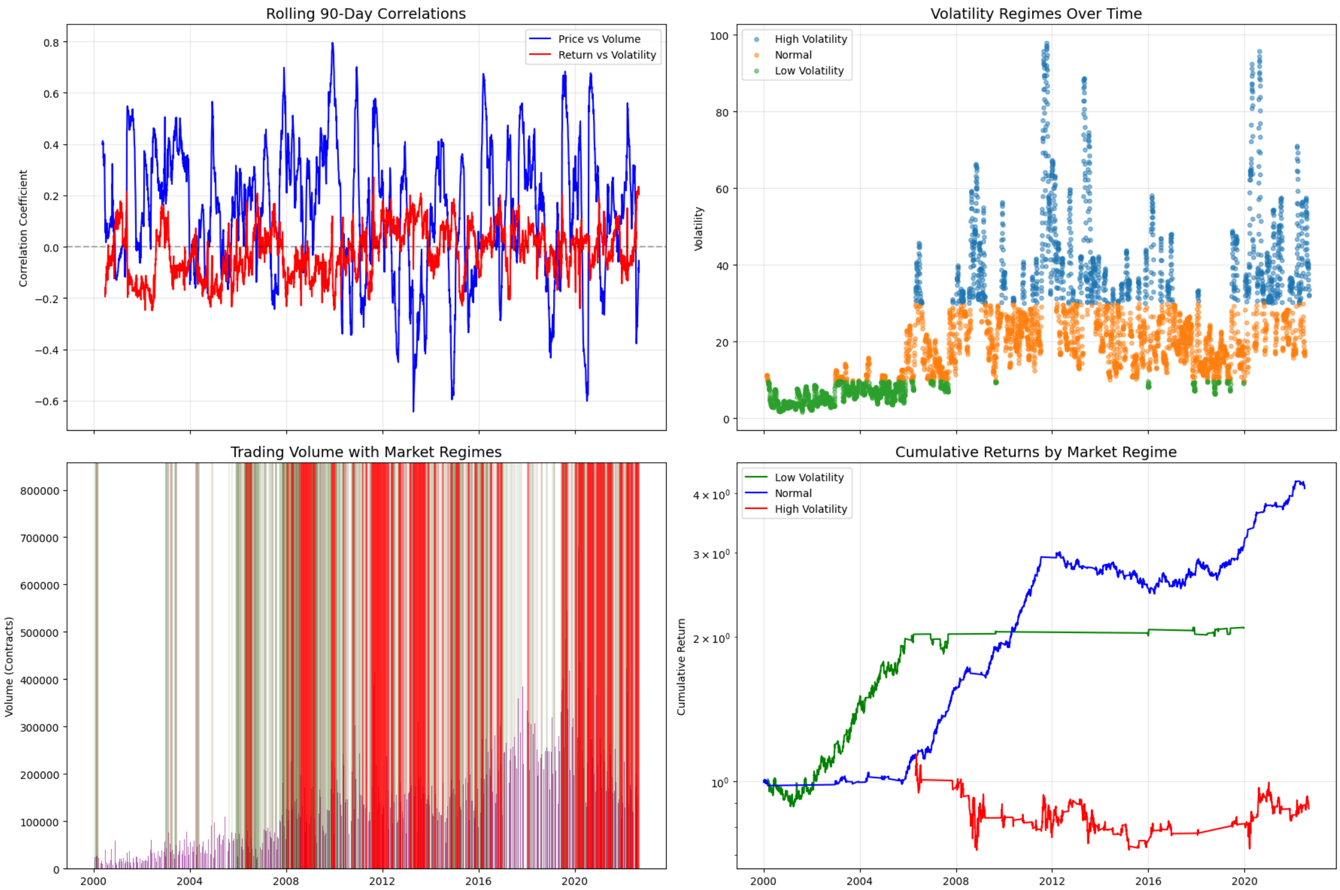
```
# 3. Principal Component Analysis (PCA)
features = ['Close', 'Volume', 'Returns', 'Volatility']
X = df[features].dropna()
X_scaled = StandardScaler().fit_transform(X)

pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)
df_pca = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])
df_pca['Market_Regime'] = df.loc[X.index, 'Market_Regime']

# PCA Visualization
plt.figure(figsize=(14, 10))
sns.scatterplot(x='PC1', y='PC2', hue='Market_Regime', data=df_pca,
                palette='Set1', alpha=0.7, s=80)
plt.title('PCA of Gold Market Features', fontsize=16)
plt.xlabel(f'PC1 ({pca.explained_variance_ratio_[0]*100:.1f}% Variance)')
plt.ylabel(f'PC2 ({pca.explained_variance_ratio_[1]*100:.1f}% Variance)')
plt.legend(title='Market Regime')
plt.grid(alpha=0.3)

# Add loadings
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)
for i, feature in enumerate(features):
    plt.arrow(0, 0, loadings[i, 0], loadings[i, 1],
              color='black', alpha=0.7, head_width=0.1)
    plt.text(loadings[i, 0]*1.2, loadings[i, 1]*1.2,
             feature, color='black', ha='center', va='center')

plt.show()
```




```
# 4. Time-Evolving Relationships
# Rolling correlations
window_size = 90  # Quarterly window
df['Rolling_Corr_Price_Volume'] = df['Close'].rolling(window_size).corr(df['Vol
df['Rolling_Corr_Return_Volatility'] = df['Returns'].rolling(window_size).corr(
```

```
# Create 2x2 grid for time evolution plots
fig, axs = plt.subplots(2, 2, figsize=(18, 12), sharex=True)

# Plot 1: Rolling Correlations
axs[0, 0].plot(df['Date'], df['Rolling_Corr_Price_Volume'], 'b-', label='Price
axs[0, 0].plot(df['Date'], df['Rolling_Corr_Return_Volatility'], 'r-', label='F
axs[0, 0].axhline(0, color='black', linestyle='--', alpha=0.3)
axs[0, 0].set_title('Rolling 90-Day Correlations', fontsize=14)
axs[0, 0].set_ylabel('Correlation Coefficient')
axs[0, 0].legend()
axs[0, 0].grid(alpha=0.3)

# Plot 2: Volatility Clusters
for regime in ['High Volatility', 'Normal', 'Low Volatility']:
    subset = df[df['Market_Regime'] == regime]
    axs[0, 1].scatter(subset['Date'], subset['Volatility'],
                     label=regime, alpha=0.5, s=15)
axs[0, 1].set_title('Volatility Regimes Over Time', fontsize=14)
axs[0, 1].set_ylabel('Volatility')
axs[0, 1].legend()
axs[0, 1].grid(alpha=0.3)

# Plot 3: Volume Anomalies
axs[1, 0].bar(df['Date'], df['Volume'], color='purple', alpha=0.6)
axs[1, 0].set_title('Trading Volume with Market Regimes', fontsize=14)
axs[1, 0].set_ylabel('Volume (Contracts)')

# Add regime overlays
for regime, color in zip(['High Volatility', 'Crash Days', 'Surge Days'],
                         ['red', 'darkred', 'green']):
    regime_dates = df[df['Market_Regime'] == regime]['Date']
    for date in regime_dates:
        axs[1, 0].axvline(date, color=color, alpha=0.1)

# Plot 4: Cumulative Returns by Regime
for regime, color in zip(['Low Volatility', 'Normal', 'High Volatility'],
                         ['green', 'blue', 'red']):
    subset = df[df['Market_Regime'] == regime]
    cumulative_return = (1 + subset['Returns']/100).cumprod()
    axs[1, 1].plot(subset['Date'], cumulative_return,
                  color=color, label=regime)
axs[1, 1].set_title('Cumulative Returns by Market Regime', fontsize=14)
axs[1, 1].set_ylabel('Cumulative Return')
axs[1, 1].set_yscale('log')
axs[1, 1].legend()
axs[1, 1].grid(alpha=0.3)
```

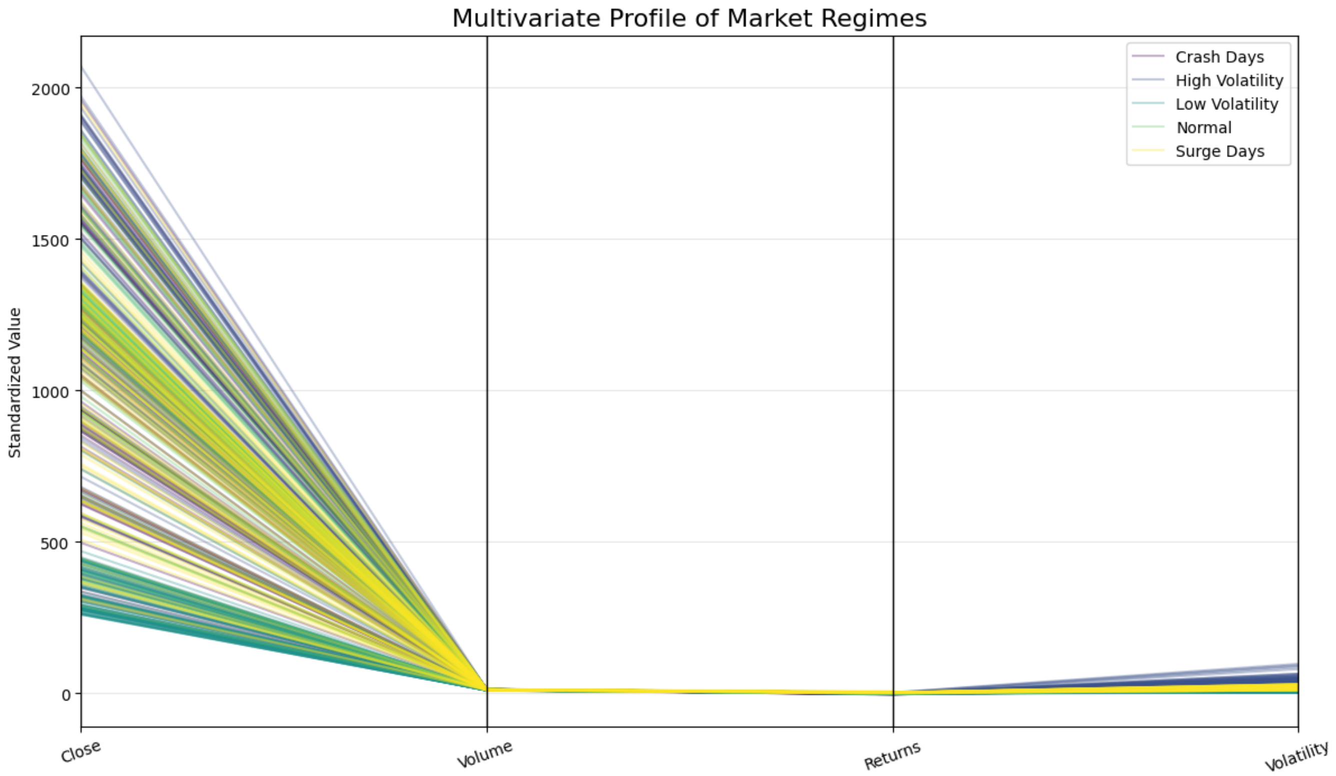
```
plt.tight_layout()
plt.show()
```




```
# 5. Parallel Coordinates for Market Regimes
plt.figure(figsize=(14, 8))
parallel_cols = ['Close', 'Volume', 'Returns', 'Volatility', 'Market_Regime']
parallel_df = df[parallel_cols].copy()
parallel_df['Volume'] = np.log(parallel_df['Volume'])  # Normalize volume
```

```
# Sample to avoid overplotting
sampled_df = parallel_df.groupby('Market_Regime').apply(
    lambda x: x.sample(min(len(x), 100))).reset_index(drop=True)

parallel_coordinates(sampled_df, 'Market_Regime',
                     colormap='viridis', alpha=0.3)
plt.title('Multivariate Profile of Market Regimes', fontsize=16)
plt.ylabel('Standardized Value')
plt.xticks(rotation=20)
plt.grid(alpha=0.3)
plt.show()
```

```
/tmp/ipython-input-48-3182872735.py:8: DeprecationWarning: DataFrameGroupBy
  sampled_df = parallel_df.groupby('Market_Regime').apply(
```



Start coding or generate with AI.