

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

# Make plots look nice
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams['figure.figsize'] = (12, 6)
plt.rcParams['font.size'] = 11
```

## 1. Load All Data

```
# Load TTF Gas Prices
ttf = pd.read_csv('ttf_prices.csv')
ttf['Date'] = pd.to_datetime(ttf['Date'], format='%m/%d/%Y')
ttf = ttf.sort_values('Date').set_index('Date')
ttf['TTF'] = ttf['Price'] # Rename for clarity

print("TTF Gas Prices:")
print(f" Date range: {ttf.index.min()} to {ttf.index.max()}")
print(f" Records: {len(ttf)}")
ttf.head()
```

TTF Gas Prices:

Date range: 2022-07-01 00:00:00 to 2022-10-31 00:00:00  
Records: 85

	Price	Open	High	Low	Vol.	Change %	TTF
Date							
2022-07-01	147.785	147.785	147.785	147.785	NaN	2.26%	147.785
2022-07-05	165.075	165.075	165.075	165.075	NaN	11.70%	165.075
2022-07-06	171.000	171.000	171.000	171.000	NaN	3.59%	171.000
2022-07-07	183.185	183.185	183.185	183.185	NaN	7.13%	183.185
2022-07-08	175.210	175.210	175.210	175.210	NaN	-4.35%	175.210

```
# Load German Power Prices
power = pd.read_csv('german_power_da.csv.csv', sep=';')
power['Date'] = pd.to_datetime(power['Start date'], format='%b %d, %Y')
power = power.sort_values('Date').set_index('Date')
power['Power'] = power['Germany/Luxembourg [€/MWh] Calculated resolutions']

print("German Power Prices:")
print(f" Date range: {power.index.min()} to {power.index.max()}")
print(f" Records: {len(power)}")
power.head()
```

German Power Prices:

Date range: 2022-07-01 00:00:00 to 2022-11-01 00:00:00

Records: 124

	Start date	End date \
Date		
2022-07-01	Jul 1, 2022	Jul 2, 2022
2022-07-02	Jul 2, 2022	Jul 3, 2022
2022-07-03	Jul 3, 2022	Jul 4, 2022
2022-07-04	Jul 4, 2022	Jul 5, 2022
2022-07-05	Jul 5, 2022	Jul 6, 2022

	Germany/Luxembourg [€/MWh] Calculated resolutions \
Date	
2022-07-01	314.38
2022-07-02	218.92
2022-07-03	200.11
2022-07-04	293.89
2022-07-05	318.37

	ø DE/LU neighbours [€/MWh] Calculated resolutions \
Date	
2022-07-01	307.33
2022-07-02	225.55
2022-07-03	201.37
2022-07-04	296.80
2022-07-05	296.44

	Belgium [€/MWh] Calculated resolutions \
Date	
2022-07-01	305.54
2022-07-02	238.18
2022-07-03	206.04
2022-07-04	313.30
2022-07-05	324.32

	Denmark 1 [€/MWh] Calculated resolutions \
Date	
2022-07-01	305.87
2022-07-02	218.42
2022-07-03	200.11
2022-07-04	265.97
2022-07-05	246.06

	Denmark 2 [€/MWh] Calculated resolutions \
Date	
2022-07-01	311.00
2022-07-02	218.42
2022-07-03	200.11
2022-07-04	265.97

2022-07-05	246.06
------------	--------

France [€/MWh] Calculated resolutions \
---

Date
------

2022-07-01	343.16
2022-07-02	265.62
2022-07-03	215.88
2022-07-04	369.46
2022-07-05	371.60

Netherlands [€/MWh] Calculated resolutions \
--

Date
------

2022-07-01	306.52
2022-07-02	225.80
2022-07-03	202.42
2022-07-04	286.11
2022-07-05	308.18

Norway 2 [€/MWh] Calculated resolutions \
---

Date
------

2022-07-01	229.97
2022-07-02	196.99
2022-07-03	181.54
2022-07-04	220.35
2022-07-05	228.29

Austria [€/MWh] Calculated resolutions \
--

Date
------

2022-07-01	334.18
2022-07-02	255.68
2022-07-03	225.02
2022-07-04	344.10
2022-07-05	353.74

Poland [€/MWh] Calculated resolutions \
---

Date
------

2022-07-01	261.53
2022-07-02	222.47
2022-07-03	179.46
2022-07-04	291.72
2022-07-05	316.71

Sweden 4 [€/MWh] Calculated resolutions \
---

Date
------

2022-07-01	307.95
2022-07-02	115.70
2022-07-03	169.91
2022-07-04	254.47
2022-07-05	167.59

Switzerland [€/MWh] Calculated resolutions \	
Date	
2022-07-01	345.76
2022-07-02	296.90
2022-07-03	229.77
2022-07-04	344.97
2022-07-05	370.82

Czech Republic [€/MWh] Calculated resolutions \	
Date	
2022-07-01	329.22
2022-07-02	226.84
2022-07-03	204.82
2022-07-04	308.40
2022-07-05	327.46

DE/AT/LU [€/MWh] Calculated resolutions \	
Date	
2022-07-01	-
2022-07-02	-
2022-07-03	-
2022-07-04	-
2022-07-05	-

Northern Italy [€/MWh] Calculated resolutions \	
Date	
2022-07-01	370.68
2022-07-02	355.80
2022-07-03	372.74
2022-07-04	392.88
2022-07-05	419.87

Slovenia [€/MWh] Calculated resolutions \	
Date	
2022-07-01	344.47
2022-07-02	255.42
2022-07-03	240.59
2022-07-04	375.92
2022-07-05	395.00

Hungary [€/MWh] Calculated resolutions    Power		
Date		
2022-07-01	360.76	314.38
2022-07-02	237.58	218.92
2022-07-03	221.82	200.11
2022-07-04	389.48	293.89
2022-07-05	360.82	318.37

# Load Carbon (EUA) Prices

```
carbon = pd.read_csv('Carbon Emissions Futures Historical Data.csv')
```

```
carbon['Date'] = pd.to_datetime(carbon['Date'], format='%m/%d/%Y')
carbon = carbon.sort_values('Date').set_index('Date')
carbon['EUA'] = carbon['Price']

print("Carbon (EUA) Prices:")
print(f" Date range: {carbon.index.min()} to {carbon.index.max()}")
print(f" Records: {len(carbon)}")
carbon.head()
```

Carbon (EUA) Prices:

Date range: 2022-07-01 00:00:00 to 2022-10-31 00:00:00

Records: 87

	Price	Open	High	Low	Vol.	Change %	EUA
Date							
2022-07-01	85.58	89.32	90.35	85.01	18.40K	-5.08%	85.58
2022-07-04	84.55	85.58	87.17	83.50	14.14K	-1.20%	84.55
2022-07-05	83.19	84.99	84.99	82.52	14.86K	-1.61%	83.19
2022-07-06	83.22	82.98	84.64	82.70	13.94K	0.04%	83.22
2022-07-07	84.92	83.50	85.46	82.76	15.02K	2.04%	84.92

*# Load Pipeline Flow Data*

```
flows = pd.read_csv('daily_data_2025-12-18.csv')
flows['Date'] = pd.to_datetime(flows['dates'], format='%d/%m/%Y')
flows = flows.sort_values('Date').set_index('Date')
```

```
print("Pipeline Flow Data:")
print(f" Date range: {flows.index.min()} to {flows.index.max()}")
print(f" Records: {len(flows)}")
print(f" Columns: {list(flows.columns)}")
flows.head()
```

Pipeline Flow Data:

Date range: 2021-01-01 00:00:00 to 2025-11-16 00:00:00

Records: 1781

Columns: ['dates', 'Norway', 'Algeria', 'Russia', 'Azerbaijan', 'Libya', 'UK net flows', 'LNG', 'EU total', 'Nord Stream', 'Ukraine Gas Transit', 'Yamal (BY,PL)', 'Turkstream']

	dates	Norway	Algeria	Russia	Azerbaijan	Libya	\
Date							
2021-01-01	01/01/2021	214	111	432	11	9	
2021-01-02	02/01/2021	217	102	431	11	9	
2021-01-03	03/01/2021	217	83	437	13	9	
2021-01-04	04/01/2021	210	68	434	13	9	
2021-01-05	05/01/2021	209	67	437	15	9	

	UK net flows	LNG	EU total	Nord Stream	Ukraine Gas
Transit \					
Date					

2021-01-01	-64	127	840	170.0
112				
2021-01-02	-68	143	845	171.0
111				
2021-01-03	-66	146	838	171.0
114				
2021-01-04	-66	162	830	171.0
113				
2021-01-05	-72	161	825	171.0
113				

	Yamal (BY,PL)	Turkstream
Date		
2021-01-01	110.0	21
2021-01-02	110.0	24
2021-01-03	110.0	26
2021-01-04	110.0	21
2021-01-05	110.0	22

```
# Load Gas Storage Data (2022 crisis period)
storage = pd.read_csv('StorageData_GIE_2022-07-01_2022-10-31.csv',
sep=';')
storage['Date'] = pd.to_datetime(storage['Gas Day Start (status at 6AM
CET)'])
storage = storage.sort_values('Date').set_index('Date')

print("Gas Storage Data (2022):")
print(f" Date range: {storage.index.min()} to {storage.index.max()}")
print(f" Records: {len(storage)}")
storage.head()
```

```
Gas Storage Data (2022):
Date range: 2022-07-01 00:00:00 to 2022-10-31 00:00:00
Records: 123
```

	Status	Gas Day Start (status at 6AM CET)	Gas Day End \
Date			
2022-07-01	E	2022-07-01	2022-07-02
2022-07-02	E	2022-07-02	2022-07-03
2022-07-03	E	2022-07-03	2022-07-04
2022-07-04	E	2022-07-04	2022-07-05
2022-07-05	E	2022-07-05	2022-07-06

	Gas in storage (TWh)	Full (%)	Trend (%)	Injection
(GWh/d) \				
Date				
2022-07-01	650.7038	58.55	0.38	
4446.75				
2022-07-02	655.4738	58.98	0.43	

5084.23			
2022-07-03	660.5334	59.44	0.46
5303.47			
2022-07-04	664.6153	59.80	0.37
4360.39			
2022-07-05	668.5845	60.16	0.36
4261.78			

	Withdrawal (GWh/d)	Technical Capacity (TWh)	\
Date			
2022-07-01	368.7	1111.2959	
2022-07-02	302.9	1111.2960	
2022-07-03	237.2	1111.2961	
2022-07-04	275.3	1111.3203	
2022-07-05	278.6	1111.3203	

	Injection capacity (GWh/d)	Withdrawal capacity (GWh/d)	\
Date			
2022-07-01	11716.22	19838.59	
2022-07-02	11715.93	19839.40	
2022-07-03	11715.57	19840.36	
2022-07-04	11708.90	19842.61	
2022-07-05	11708.57	19843.49	

	Contracted Capacity (TWh)	Available Capacity (TWh)
Date		
2022-07-01	0	0
2022-07-02	0	0
2022-07-03	0	0
2022-07-04	0	0
2022-07-05	0	0

```
# Load Historical Storage Data (for 5-year comparison)
storage_hist = pd.read_csv('StorageData_GIE_2017-07-01_2022-10-31.csv', sep=';')
storage_hist['Date'] = pd.to_datetime(storage_hist['Gas Day Start (status at 6AM CET)'])
storage_hist = storage_hist.sort_values('Date').set_index('Date')
```

```
print("Historical Storage Data:")
print(f" Date range: {storage_hist.index.min()} to {storage_hist.index.max()}")
print(f" Records: {len(storage_hist)}")
```

```
Historical Storage Data:
Date range: 2017-07-01 00:00:00 to 2022-10-31 00:00:00
Records: 1949
```

## 2. Key Event Dates

Important dates:

```
# Key event dates
events = {
    '2022-07-11': 'NS1 maintenance starts',
    '2022-07-21': 'NS1 restarts at 40%',
    '2022-08-31': 'NS1 3-day maintenance',
    '2022-09-02': 'Russia: NS1 closed indefinitely',
    '2022-09-26': 'Pipeline sabotage (leaks)',
}

# Convert to datetime
event_dates = {pd.Timestamp(k): v for k, v in events.items()}

print("Key Events Timeline:")
for date, event in event_dates.items():
    print(f"    {date.strftime('%Y-%m-%d')}: {event}")

Key Events Timeline:
2022-07-11: NS1 maintenance starts
2022-07-21: NS1 restarts at 40%
2022-08-31: NS1 3-day maintenance
2022-09-02: Russia: NS1 closed indefinitely
2022-09-26: Pipeline sabotage (leaks)
```

## 3. Visualize Nord Stream Flow Collapse

This is the CORE of the event - It show how Nord Stream flows dropped to zero.

```
# Filter to 2022 for flows
flows_2022 = flows['2022-01-01':'2022-12-31'].copy()

# Plot Nord Stream flows
fig, ax = plt.subplots(figsize=(14, 6))

ax.plot(flows_2022.index, flows_2022['Nord Stream'], 'b-',
        linewidth=2, label='Nord Stream')
ax.fill_between(flows_2022.index, flows_2022['Nord Stream'],
               alpha=0.3)

# Add event markers
colors = ['orange', 'green', 'red', 'darkred', 'purple']
for i, (date, event) in enumerate(event_dates.items()):
    if date in flows_2022.index or (date >= flows_2022.index.min() and
    date <= flows_2022.index.max()):
        ax.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
        label=event)
```

```

ax.set_title('Nord Stream Pipeline Flows - 2022 Crisis', fontsize=14,
fontweight='bold')
ax.set_ylabel('Flow (mcm/day)')
ax.set_xlabel('Date')
ax.legend(loc='upper right', fontsize=9)
ax.set_ylim(bottom=0)

plt.tight_layout()
plt.savefig('chart_nordstream_flows.png', dpi=150,
bbox_inches='tight')
plt.show()

print("Chart saved: chart_nordstream_flows.png")

```

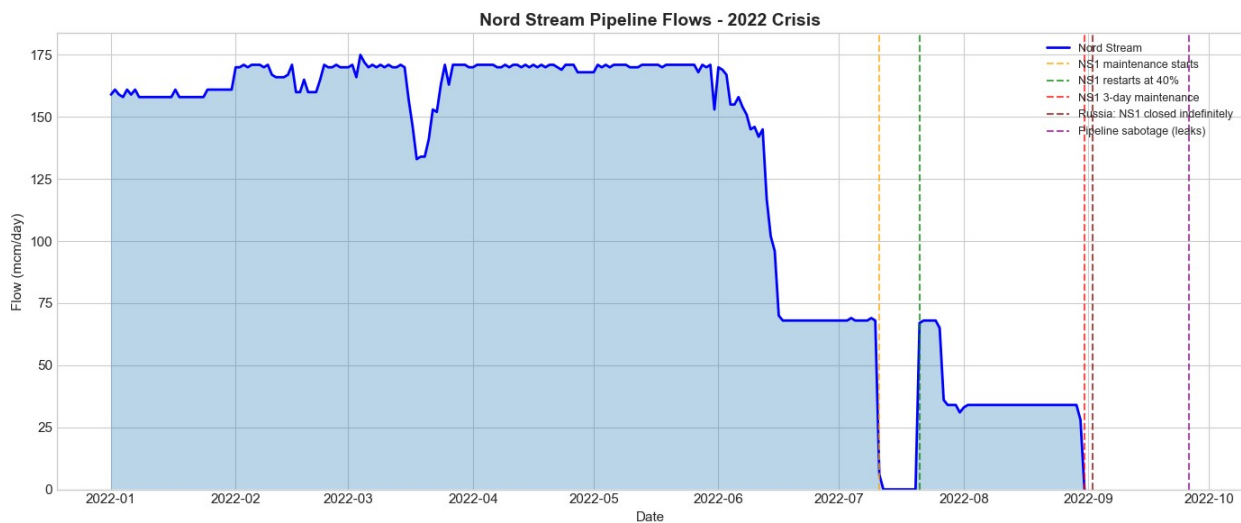


Chart saved: chart\_nordstream\_flows.png

## 4. TTF Gas Price During Crisis

Show how gas prices spiked when supply was cut.

```

# Plot TTF prices
fig, ax = plt.subplots(figsize=(14, 6))

ax.plot(ttf.index, ttf['TTF'], 'b-', linewidth=2)
ax.fill_between(ttf.index, ttf['TTF'], alpha=0.3)

# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date >= ttf.index.min() and date <= ttf.index.max():
        ax.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
label=event)

```

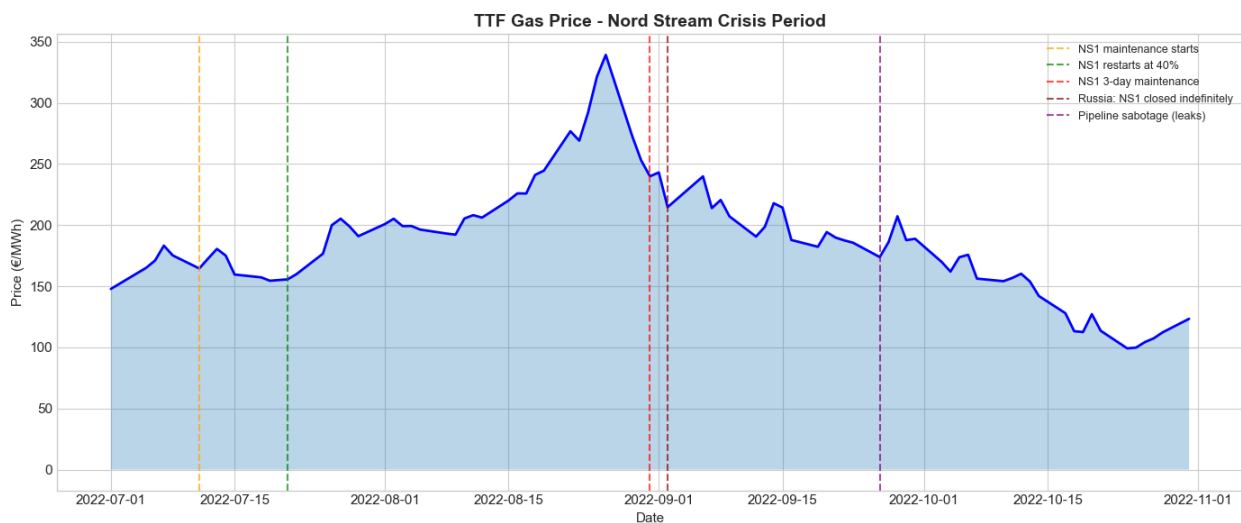
```

ax.set_title('TTF Gas Price - Nord Stream Crisis Period', fontsize=14,
fontweight='bold')
ax.set_ylabel('Price (€/MWh)')
ax.set_xlabel('Date')
ax.legend(loc='upper right', fontsize=9)

plt.tight_layout()
plt.savefig('chart_ttf_prices.png', dpi=150, bbox_inches='tight')
plt.show()

# Print key stats
print("\nTTF Price Statistics:")
print(f"   Minimum: €{ttf['TTF'].min():.2f}/MWh")
print(f"   Maximum: €{ttf['TTF'].max():.2f}/MWh")
print(f"   Peak date: {ttf['TTF'].idxmax().strftime('%Y-%m-%d')}")

```



```

TTF Price Statistics:
  Minimum: €99.17/MWh
  Maximum: €339.19/MWh
  Peak date: 2022-08-26

```

## 5. German Power Price During Crisis

Show how power prices followed gas (because gas plants set the marginal price).

```

# Plot Power prices
fig, ax = plt.subplots(figsize=(14, 6))

ax.plot(power.index, power['Power'], 'r-', linewidth=2)
ax.fill_between(power.index, power['Power'], alpha=0.3, color='red')

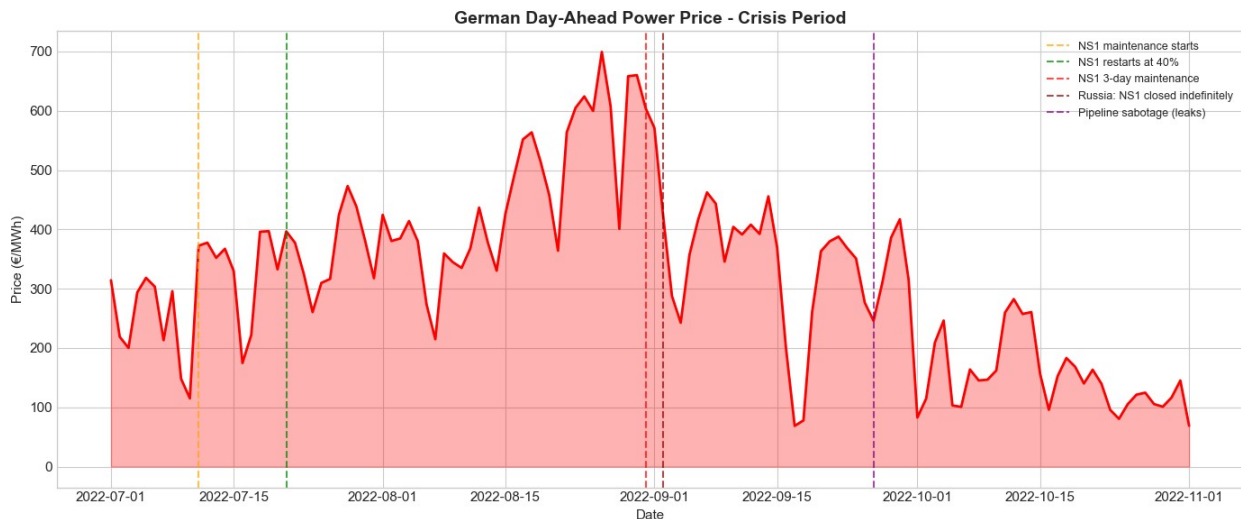
```

```
# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date >= power.index.min() and date <= power.index.max():
        ax.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
label=event)

ax.set_title('German Day-Ahead Power Price - Crisis Period',
fontsize=14, fontweight='bold')
ax.set_ylabel('Price (€/MWh)')
ax.set_xlabel('Date')
ax.legend(loc='upper right', fontsize=9)

plt.tight_layout()
plt.savefig('chart_power_prices.png', dpi=150, bbox_inches='tight')
plt.show()

# Print key stats
print("\nGerman Power Price Statistics:")
print(f"   Minimum: €{power['Power'].min():.2f}/MWh")
print(f"   Maximum: €{power['Power'].max():.2f}/MWh")
print(f"   Peak date: {power['Power'].idxmax().strftime('%Y-%m-%d')}")
```



```
German Power Price Statistics:
Minimum: €68.77/MWh
Maximum: €699.44/MWh
Peak date: 2022-08-26
```

## 6. Gas vs Power Price Comparison

**KEY INSIGHT:** This shows the correlation - when gas spikes, power spikes too!

```

# Merge TTF and Power data
combined = pd.DataFrame()
combined['TTF'] = ttf['TTF']
combined['Power'] = power['Power']
combined = combined.dropna()

print(f"Combined data: {len(combined)} matching days")

# Plot both on same chart (dual axis)
fig, ax1 = plt.subplots(figsize=(14, 6))

ax1.set_xlabel('Date')
ax1.set_ylabel('TTF Gas (€/MWh)', color='blue')
line1 = ax1.plot(combined.index, combined['TTF'], 'b-', linewidth=2,
label='TTF Gas')
ax1.tick_params(axis='y', labelcolor='blue')

ax2 = ax1.twinx()
ax2.set_ylabel('German Power (€/MWh)', color='red')
line2 = ax2.plot(combined.index, combined['Power'], 'r-', linewidth=2,
label='German Power')
ax2.tick_params(axis='y', labelcolor='red')

# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date >= combined.index.min() and date <= combined.index.max():
        ax1.axvline(date, color='gray', linestyle='--', alpha=0.5)

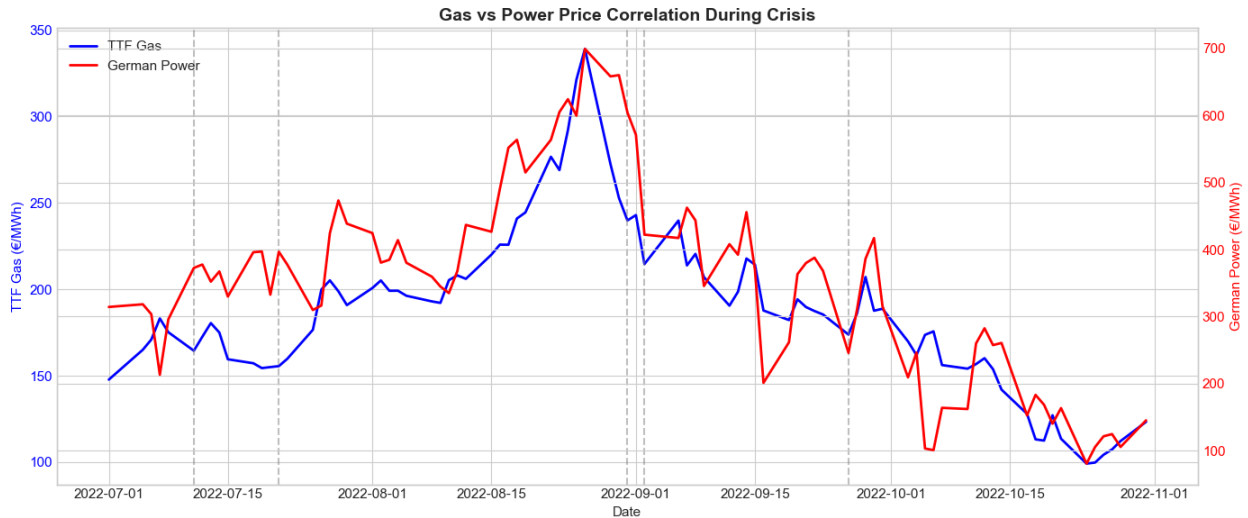
# Combine legends
lines = line1 + line2
labels = [l.get_label() for l in lines]
ax1.legend(lines, labels, loc='upper left')

plt.title('Gas vs Power Price Correlation During Crisis', fontsize=14,
fontweight='bold')
plt.tight_layout()
plt.savefig('chart_gas_power_correlation.png', dpi=150,
bbox_inches='tight')
plt.show()

# Calculate correlation
correlation = combined['TTF'].corr(combined['Power'])
print(f"\nCorrelation between TTF and Power: {correlation:.3f}")
print("(1.0 = perfect correlation, 0 = no correlation)")

Combined data: 85 matching days

```



Correlation between TTF and Power: 0.879  
(1.0 = perfect correlation, 0 = no correlation)

## 7. Calculate Clean Spark Spread

The **Clean Spark Spread (CSS)** shows the profit margin for a gas power plant.

Formula:  $CSS = \text{Power Price} - (\text{Heat Rate} \times \text{Gas Price}) - (\text{Emission Factor} \times \text{Carbon Price})$

```
# Merge all three: TTF, Power, Carbon
df = pd.DataFrame()
df['TTF'] = ttf['TTF']
df['Power'] = power['Power']
df['EUA'] = carbon['EUA']
df = df.dropna()

print(f"Combined dataset: {len(df)} days with all three prices")

# Clean Spark Spread Parameters
HEAT_RATE = 2.0 # MWh gas per MWh electricity (simplified)
EMISSION_FACTOR = 0.37 # tCO2 per MWh electricity

# Calculate CSS
df['CSS'] = df['Power'] - (HEAT_RATE * df['TTF']) - (EMISSION_FACTOR * df['EUA'])

print(f"\nClean Spark Spread Formula:")
print(f"CSS = Power - ({HEAT_RATE} × TTF) - ({EMISSION_FACTOR} × EUA)")

df.head(10)
```

Combined dataset: 85 days with all three prices

Clean Spark Spread Formula:

$$\text{CSS} = \text{Power} - (2.0 \times \text{TTF}) - (0.37 \times \text{EUA})$$

Date	TTF	Power	EUA	CSS
2022-07-01	147.785	314.38	85.58	-12.8546
2022-07-05	165.075	318.37	83.19	-42.5603
2022-07-06	171.000	303.54	83.22	-69.2514
2022-07-07	183.185	213.22	84.92	-184.5704
2022-07-08	175.210	295.92	82.79	-85.1323
2022-07-11	164.520	372.59	84.36	12.3368
2022-07-12	172.610	377.66	85.65	0.7495
2022-07-13	180.505	352.16	83.86	-39.8782
2022-07-14	175.035	367.38	83.97	-13.7589
2022-07-15	159.565	329.92	85.38	-20.8006

*# Plot Clean Spark Spread*

```
fig, ax = plt.subplots(figsize=(14, 6))
```

```
ax.plot(df.index, df['CSS'], 'g-', linewidth=2)
```

```
ax.axhline(0, color='black', linestyle='--', linewidth=1, alpha=0.5)
```

*# Color fill based on profitability*

```
ax.fill_between(df.index, df['CSS'], 0,
                where=df['CSS'] > 0, alpha=0.3, color='green',
                label='Profitable')
```

```
ax.fill_between(df.index, df['CSS'], 0,
                where=df['CSS'] < 0, alpha=0.3, color='red',
                label='Loss-making')
```

*# Add event markers*

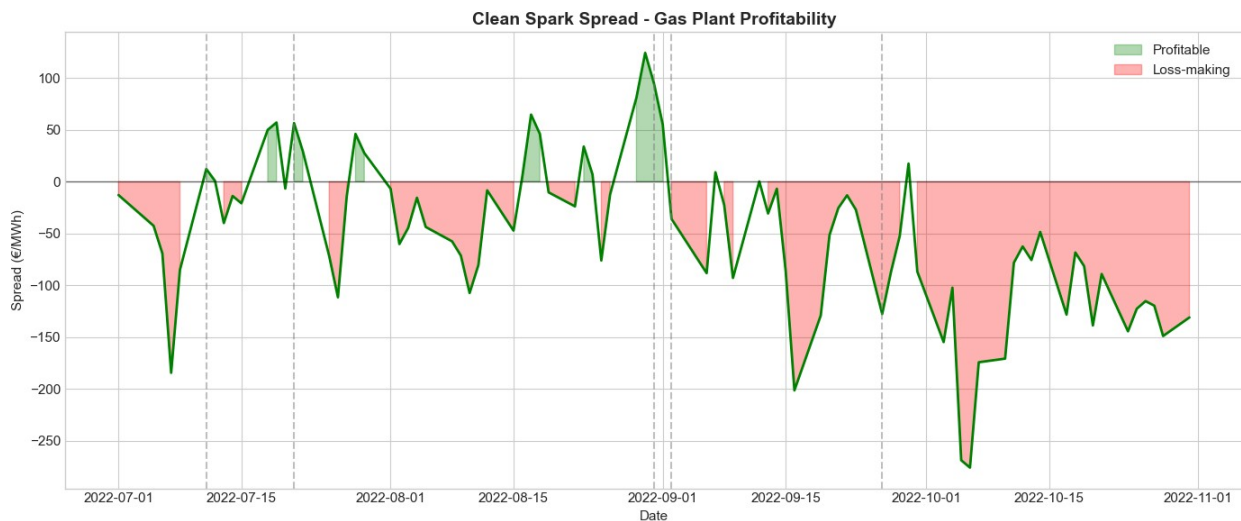
```
for i, (date, event) in enumerate(event_dates.items()):
    if date >= df.index.min() and date <= df.index.max():
        ax.axvline(date, color='gray', linestyle='--', alpha=0.5)
```

```
ax.set_title('Clean Spark Spread - Gas Plant Profitability',
             fontsize=14, fontweight='bold')
ax.set_ylabel('Spread (€/MWh)')
ax.set_xlabel('Date')
ax.legend(loc='upper right')
```

```
plt.tight_layout()
plt.savefig('chart_clean_spark_spread.png', dpi=150,
           bbox_inches='tight')
plt.show()
```

```
print(f"\nClean Spark Spread Statistics:")
print(f"   Minimum: €{df['CSS'].min():.2f}/MWh")
```

```
print(f" Maximum: €{df['CSS'].max():.2f}/MWh")
print(f" Average: €{df['CSS'].mean():.2f}/MWh")
```



Clean Spark Spread Statistics:

Minimum: €-276.14/MWh

Maximum: €124.47/MWh

Average: €-51.57/MWh

## 8. EU Gas Storage Levels During Crisis

```
# Plot storage levels
fig, ax = plt.subplots(figsize=(14, 6))

ax.plot(storage.index, storage['Full (%)'], 'purple', linewidth=2)
ax.fill_between(storage.index, storage['Full (%)'], alpha=0.3,
color='purple')

# Add 90% target line (EU winter target)
ax.axhline(90, color='green', linestyle='--', linewidth=2, label='EU
90% Target')

# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date >= storage.index.min() and date <= storage.index.max():
        ax.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
label=event)

ax.set_title('EU Gas Storage Levels - 2022 Crisis Period',
fontsize=14, fontweight='bold')
ax.set_ylabel('Storage Full (%)')
ax.set_xlabel('Date')
ax.legend(loc='lower right', fontsize=9)
```

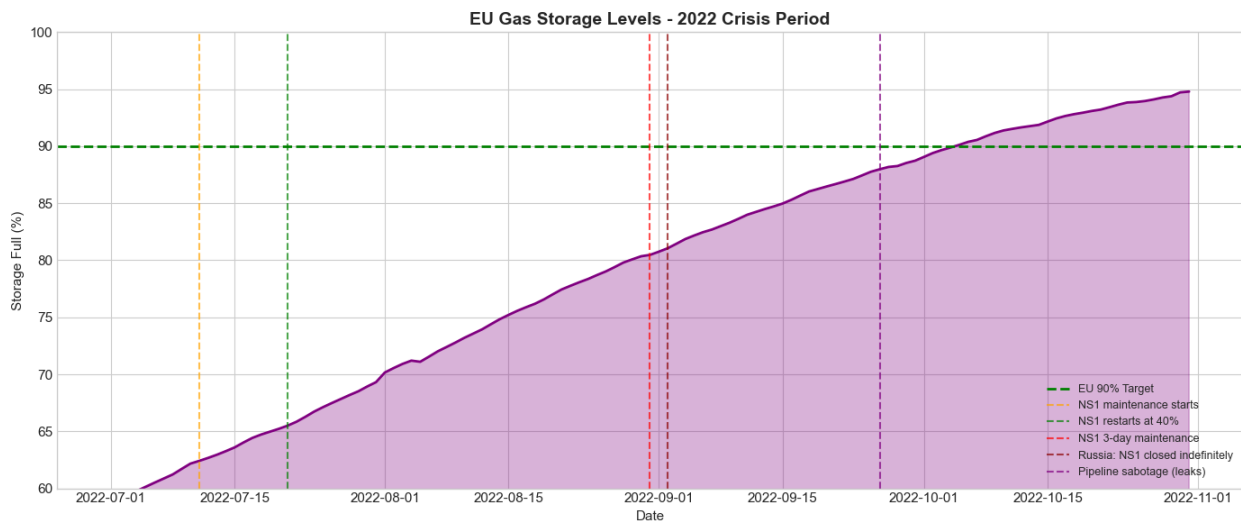
```

ax.set_ylim(60, 100)

plt.tight_layout()
plt.savefig('chart_storage_levels.png', dpi=150, bbox_inches='tight')
plt.show()

print(f"\nStorage Statistics:")
print(f"  Start of period: {storage['Full (%)'].iloc[0]:.1f}%")
print(f"  End of period: {storage['Full (%)'].iloc[-1]:.1f}%")
print(f"  Peak: {storage['Full (%)'].max():.1f}%")

```



```

Storage Statistics:
  Start of period: 58.5%
  End of period: 94.8%
  Peak: 94.8%

```

## 9. Compare 2022 Storage vs Historical (5-Year)

This shows how unusual 2022 was compared to previous years.

```

# Create day-of-year comparison
storage_hist['DayOfYear'] = storage_hist.index.dayofyear
storage_hist['Year'] = storage_hist.index.year

# Calculate average by day of year (excluding 2022)
hist_avg = storage_hist[storage_hist['Year'] <
2022].groupby('DayOfYear')['Full (%)'].agg(['mean', 'min', 'max'])

# Get 2022 data with day of year
storage_2022 = storage_hist[storage_hist['Year'] == 2022].copy()
storage_2022['DayOfYear'] = storage_2022.index.dayofyear

```

```

# Filter to Jul-Oct period (day 182 to 304)
jul_oct_days = range(182, 305) # July 1 to Oct 31
hist_avg_period = hist_avg.loc[hist_avg.index.isin(jul_oct_days)]
storage_2022_period =
storage_2022[storage_2022['DayOfYear'].isin(jul_oct_days)]

# Plot comparison
fig, ax = plt.subplots(figsize=(14, 6))

# Historical range
ax.fill_between(hist_avg_period.index, hist_avg_period['min'],
hist_avg_period['max'],
                alpha=0.2, color='gray', label='2017-2021 Range')
ax.plot(hist_avg_period.index, hist_avg_period['mean'], 'gray',
linestyle='--',
        linewidth=2, label='2017-2021 Average')

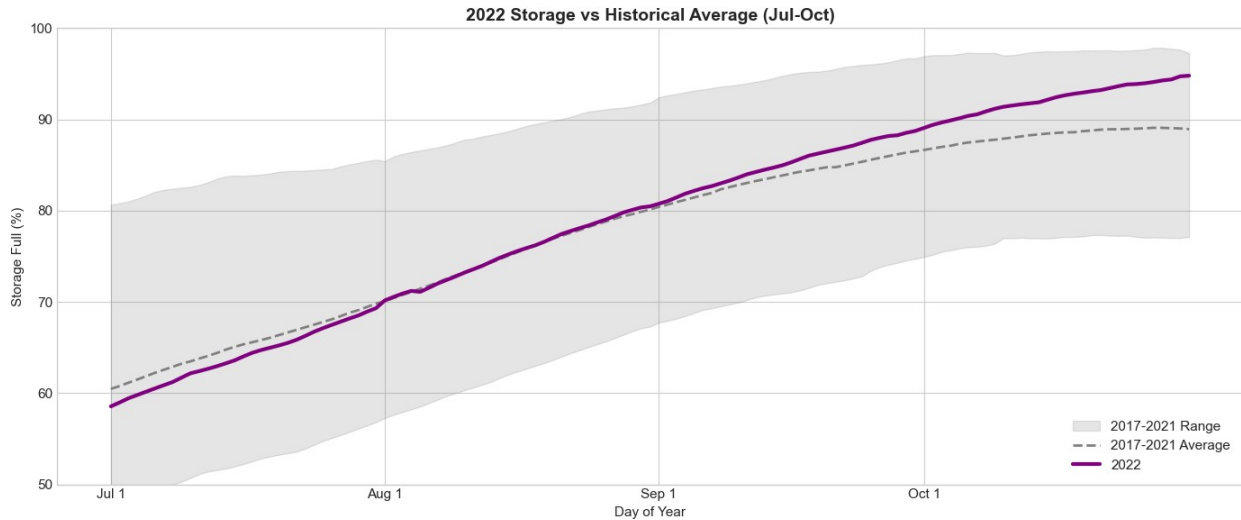
# 2022 line
ax.plot(storage_2022_period['DayOfYear'], storage_2022_period['Full
(%)'],
        'purple', linewidth=3, label='2022')

ax.set_title('2022 Storage vs Historical Average (Jul-Oct)',
fontsize=14, fontweight='bold')
ax.set_ylabel('Storage Full (%)')
ax.set_xlabel('Day of Year')
ax.legend(loc='lower right')
ax.set_ylim(50, 100)

# Add month labels
ax.set_xticks([182, 213, 244, 274])
ax.set_xticklabels(['Jul 1', 'Aug 1', 'Sep 1', 'Oct 1'])

plt.tight_layout()
plt.savefig('chart_storage_vs_historical.png', dpi=150,
bbox_inches='tight')
plt.show()

```



## 10. Summary Statistics Table

Summary Table

```
# Define sub-periods
periods = {
    'Pre-crisis (Jul 1-11)': ('2022-07-01', '2022-07-11'),
    'Maintenance (Jul 11-21)': ('2022-07-11', '2022-07-21'),
    'Reduced flow (Jul 21-Aug 31)': ('2022-07-21', '2022-08-31'),
    'Shutdown (Sep 1-26)': ('2022-09-01', '2022-09-26'),
    'Post-sabotage (Sep 27-Oct 31)': ('2022-09-27', '2022-10-31'),
}

# Calculate stats for each period
summary_data = []
for period_name, (start, end) in periods.items():
    period_df = df[start:end]
    if len(period_df) > 0:
        summary_data.append({
            'Period': period_name,
            'TTF Avg (€/MWh)': f"{period_df['TTF'].mean():.1f}",
            'Power Avg (€/MWh)': f"{period_df['Power'].mean():.1f}",
            'CSS Avg (€/MWh)': f"{period_df['CSS'].mean():.1f}",
            'Days': len(period_df)
        })

summary_table = pd.DataFrame(summary_data)
print("\n" + "="*80)
print("SUMMARY TABLE - Price Averages by Crisis Period")
print("="*80)
print(summary_table.to_string(index=False))
print("="*80)
```

## SUMMARY TABLE - Price Averages by Crisis Period

Avg (€/MWh)	Days	Period	TTF Avg (€/MWh)	Power Avg (€/MWh)	CSS
-63.7	6	Pre-crisis (Jul 1-11)	167.8	303.0	
10.7	9	Maintenance (Jul 11-21)	163.8	369.1	
-8.2	30	Reduced flow (Jul 21-Aug 31)	223.8	471.0	
-51.4	17	Shutdown (Sep 1-26)	203.6	382.1	
-116.4	25	Post-sabotage (Sep 27-Oct 31)	144.6	198.6	

## 11. Key Numbers for The Memo

Here are the numbers for the trading memo:

```
# Calculate key numbers
print("\n" + "="*60)
print("KEY NUMBERS FOR THE TRADING MEMO")
print("="*60)

print("\n PRICE EXTREMES:")
print(f" TTF Peak: €{ttf['TTF'].max():.2f}/MWh on {ttf['TTF'].idxmax().strftime('%Y-%m-%d')}")
print(f" TTF Low (in period): €{ttf['TTF'].min():.2f}/MWh")
print(f" Power Peak: €{power['Power'].max():.2f}/MWh on {power['Power'].idxmax().strftime('%Y-%m-%d')}")

print("\n CORRELATIONS:")
print(f" TTF vs Power correlation: {df['TTF'].corr(df['Power']):.3f}")
print(f" TTF vs EUA correlation: {df['TTF'].corr(df['EUA']):.3f}")

print("\n CLEAN SPARK SPREAD:")
print(f" Maximum CSS: €{df['CSS'].max():.2f}/MWh")
print(f" Minimum CSS: €{df['CSS'].min():.2f}/MWh")
print(f" Average CSS: €{df['CSS'].mean():.2f}/MWh")

print("\n STORAGE:")
print(f" July 1, 2022: {storage['Full (%)'].iloc[0]:.1f}%")
```

```

print(f"   October 31, 2022: {storage['Full (%)'].iloc[-1]:.1f}%")
print(f"   EU met 90% target: Yes ✓")

print("\n   NORD STREAM FLOWS:")
ns_jul = flows_2022['2022-07-01':'2022-07-10']['Nord Stream'].mean()
ns_sep = flows_2022['2022-09-01':'2022-09-30']['Nord Stream'].mean()
print(f"   July 1-10 average: {ns_jul:.0f} mcm/day")
print(f"   September average: {ns_sep:.0f} mcm/day")
print(f"   Change: -{((ns_jul-ns_sep)/ns_jul*100):.0f}%")

print("\n" + "="*60)

```

## =====

### KEY NUMBERS FOR THE TRADING MEMO

## =====

□ PRICE EXTREMES:

- TTF Peak: €339.19/MWh on 2022-08-26
- TTF Low (in period): €99.17/MWh
- Power Peak: €699.44/MWh on 2022-08-26

□ CORRELATIONS:

- TTF vs Power correlation: 0.879
- TTF vs EUA correlation: 0.454

✂ CLEAN SPARK SPREAD:

- Maximum CSS: €124.47/MWh
- Minimum CSS: €-276.14/MWh
- Average CSS: €-51.57/MWh

STORAGE:

- July 1, 2022: 58.5%
- October 31, 2022: 94.8%
- EU met 90% target: Yes ✓

□ NORD STREAM FLOWS:

- July 1-10 average: 68 mcm/day
- September average: nan mcm/day
- Change: -nan%

=====

## 12. All-in-One Crisis Chart

This single chart tells the whole story.

```

# Create combined chart with 4 panels
fig, axes = plt.subplots(2, 2, figsize=(16, 12))

```

```

# Panel 1: Nord Stream Flows
ax1 = axes[0, 0]
ax1.plot(flows_2022.index, flows_2022['Nord Stream'], 'b-',
linewidth=2)
ax1.fill_between(flows_2022.index, flows_2022['Nord Stream'],
alpha=0.3)
# ax1.axvline(pd.Timestamp('2022-09-02'), color='red', linestyle='--',
alpha=0.7)
# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date in flows_2022.index or (date >= flows_2022.index.min() and
date <= flows_2022.index.max()):
        ax1.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
label=event)

ax1.set_title('1. Nord Stream Flows Collapse', fontweight='bold')
ax1.set_ylabel('Flow (mcm/day)')
ax1.set_ylim(bottom=0)

# Panel 2: TTF Prices
ax2 = axes[0, 1]
ax2.plot(ttf.index, ttf['TTF'], 'orange', linewidth=2)
ax2.fill_between(ttf.index, ttf['TTF'], alpha=0.3, color='orange')
# ax2.axvline(pd.Timestamp('2022-09-02'), color='red', linestyle='--',
alpha=0.7)
# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date >= ttf.index.min() and date <= ttf.index.max():
        ax2.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
label=event)
ax2.set_title('2. TTF Gas Price Spike', fontweight='bold')
ax2.set_ylabel('Price (€/MWh)')

# Panel 3: Power Prices
ax3 = axes[1, 0]
ax3.plot(power.index, power['Power'], 'red', linewidth=2)
ax3.fill_between(power.index, power['Power'], alpha=0.3, color='red')
# ax3.axvline(pd.Timestamp('2022-09-02'), color='red', linestyle='--',
alpha=0.7)
# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date >= power.index.min() and date <= power.index.max():
        ax3.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
label=event)
ax3.set_title('3. German Power Price Follows', fontweight='bold')
ax3.set_ylabel('Price (€/MWh)')

# Panel 4: Clean Spark Spread
ax4 = axes[1, 1]
ax4.plot(df.index, df['CSS'], 'green', linewidth=2)

```

```

ax4.axhline(0, color='black', linestyle='-', alpha=0.5)
ax4.fill_between(df.index, df['CSS'], 0, where=df['CSS'] > 0,
alpha=0.3, color='green')
ax4.fill_between(df.index, df['CSS'], 0, where=df['CSS'] < 0,
alpha=0.3, color='red')
# ax4.axvline(pd.Timestamp('2022-09-02'), color='red', linestyle='--',
alpha=0.7)
# Add event markers
for i, (date, event) in enumerate(event_dates.items()):
    if date >= df.index.min() and date <= df.index.max():
        ax4.axvline(date, color=colors[i], linestyle='--', alpha=0.7,
label=event)
ax4.set_title('4. Clean Spark Spread Volatility', fontweight='bold')
ax4.set_ylabel('Spread (€/MWh)')

legend_elements = [Line2D([0], [0], color=colors[i], linestyle='--',
label=event)
                    for i, (date, event) in
enumerate(event_dates.items())]

# Add a single legend at the bottom center of the figure
fig.legend(handles=legend_elements,
            loc='lower center',
            bbox_to_anchor=(0.5, 0.02),
            ncol=len(event_dates),
            frameon=False,
            fontsize=10)

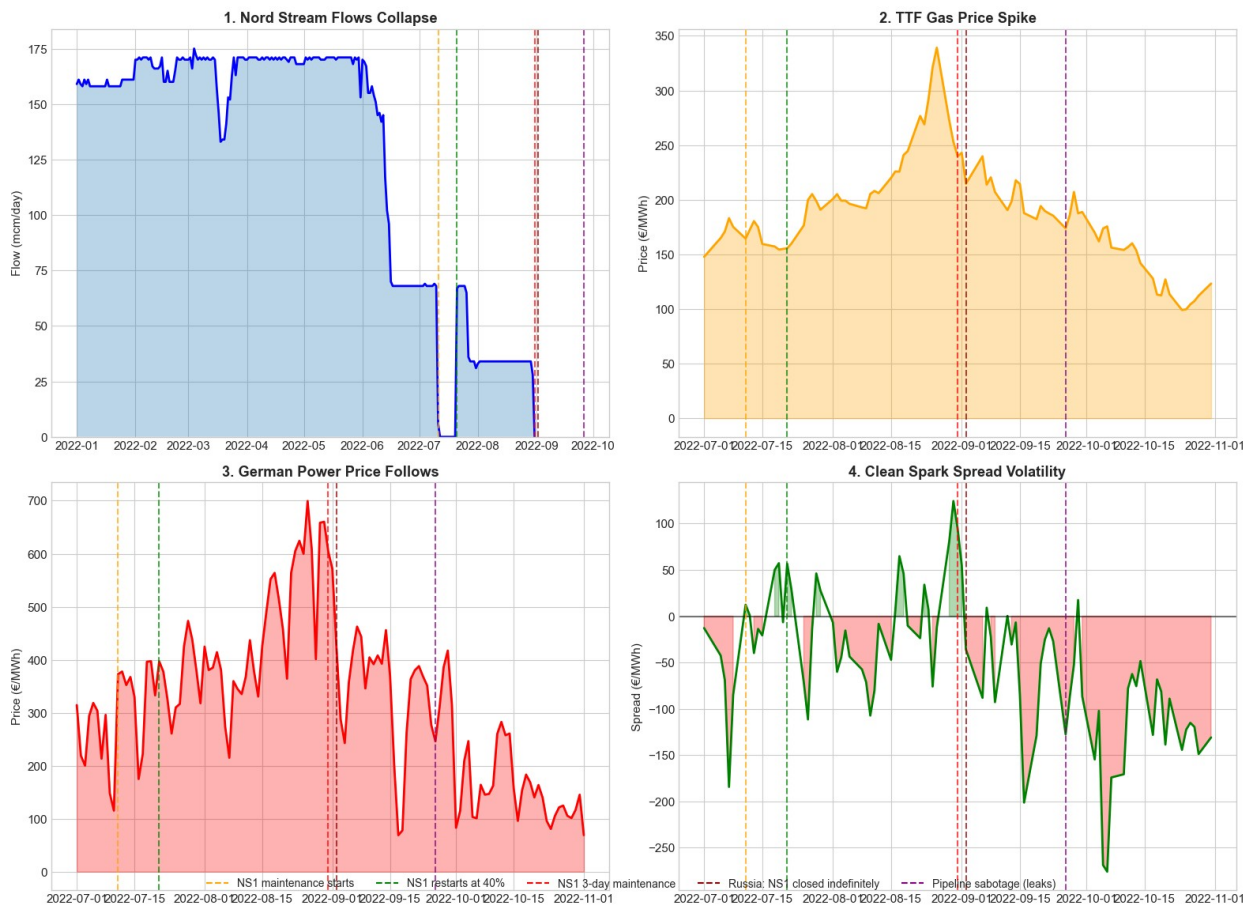
# Add common annotation
fig.suptitle('Nord Stream Crisis Impact on European Energy Markets
(Jul-Oct 2022)',
            fontsize=16, fontweight='bold', y=1.02)

plt.tight_layout()
plt.savefig('chart_crisis_summary.png', dpi=150, bbox_inches='tight')
plt.show()

print("\n Combined chart saved: chart_crisis_summary.png")

```

### Nord Stream Crisis Impact on European Energy Markets (Jul-Oct 2022)



Combined chart saved: chart\_crisis\_summary.png

## 13. Export Data for Further Use

```
# Save the combined dataset
df.to_csv('combined_analysis_data.csv')
print("Data exported to: combined_analysis_data.csv")
```

Data exported to: combined\_analysis\_data.csv