Jupyter Notebook Execution Report

Name: Your Name

Date: February 02, 2026

Cell 1: ■ Markdown

Module 2: Hedge Validation

We test whether bonds failed as a hedge (correlation spiked in 2022) and whether gold stayed low/negative correlation vs equities.

We plot Rolling 12-Month Correlation:

SPY vs BND

SPY vs IAU

Cell 2: ■ Code

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

```
# read daily returns
df = pd.read_csv("../processed/etf_returns.csv", parse_dates=["Date"])
df = df[["Date", "Ticker", "Daily_Return"]].dropna()
```

```
# Convert to wide: Date index, tickers as columns
returns_wide = df.pivot(index="Date", columns="Ticker",
values="Daily_Return").sort_index()
```

```
print("Date range:", returns_wide.index.min(), "to", returns_wide.index.max())
print("Columns available:", len(returns_wide.columns))
returns_wide.head()
```

Output:

```
Date range: 2014-01-03 00:00:00 to 2026-01-30 00:00:00
Columns available: 12
Ticker           BND       GLD       IAU  ...       VTI    XIC.TO    ZAG.TO
```

```
Date                                       ...
2014-01-03  0.000124  0.010932  0.010943  ... -0.000210 -0.002797  0.000661
2014-01-06  0.000873  0.001760  0.000000  ... -0.002630 -0.004205  0.000661
2014-01-07  0.001247 -0.005690 -0.004996  ...  0.006434  0.008447  0.002642
2014-01-08 -0.003238 -0.005891 -0.005021  ...  0.000734  0.000930  0.000659
2014-01-09  0.001875  0.002878  0.002523  ...  0.000838  0.000929 -0.000658

[5 rows x 12 columns]
```

Cell 3: ■ Code

```
# sanity check
needed = ["SPY", "BND", "IAU"]
missing = [t for t in needed if t not in returns_wide.columns]
print("Missing ticker:", missing)
```

```
# keep only needed column that exist
cols = [t for t in needed if t in returns_wide.columns]
data = returns_wide[cols].dropna()
print("Data shape after dropna:", data.shape)
data.head()
```

Output:

```
Missing ticker: []
Data shape after dropna: (3037, 3)
Ticker           SPY       BND       IAU
Date
2014-01-03 -0.000164  0.000124  0.010943
2014-01-06 -0.002897  0.000873  0.000000
2014-01-07  0.006141  0.001247 -0.004996
2014-01-08  0.000218 -0.003238 -0.005021
2014-01-09  0.000654  0.001875  0.002523
```

Cell 4: ■ Code

```
WINDOW = 252 # ~12 months trading days
```

```
spy = data["SPY"]
```

```
# Rolling correlations
corr_spy_bnd = spy.rolling(WINDOW).corr(data["BND"]) if "BND" in data.columns else None
corr_spy_iau = spy.rolling(WINDOW).corr(data["IAU"]) if "IAU" in data.columns else None
corr_spy_gld = spy.rolling(WINDOW).corr(data["GLD"]) if "GLD" in data.columns else None
```

```
# Combine for easy plotting
corr_df = pd.DataFrame({
"SPY vs BND": corr_spy_bnd,
"SPY vs IAU": corr_spy_iau,
"SPY vs GLD": corr_spy_gld
}).dropna(how="all")
```

```
corr_df.head()
```

Output:

Date	SPY vs BND	SPY vs IAU	SPY vs GLD
2015-01-02	-0.359128	-0.154894	None
2015-01-05	-0.368907	-0.169275	None
2015-01-06	-0.373877	-0.173892	None
2015-01-07	-0.372977	-0.175055	None
2015-01-08	-0.381652	-0.176631	None

Cell 5: ■ Code

```
plt.figure()
```

```
if "SPY vs BND" in corr_df.columns and corr_df["SPY vs BND"].notna().any():
plt.plot(corr_df.index, corr_df["SPY vs BND"], label="SPY vs BND (Bonds)")
```

```
if "SPY vs IAU" in corr_df.columns and corr_df["SPY vs IAU"].notna().any():
plt.plot(corr_df.index, corr_df["SPY vs IAU"], label="SPY vs IAU (Gold)")
```

```
if "SPY vs GLD" in corr_df.columns and corr_df["SPY vs GLD"].notna().any():
plt.plot(corr_df.index, corr_df["SPY vs GLD"], label="SPY vs GLD (Gold)")
```

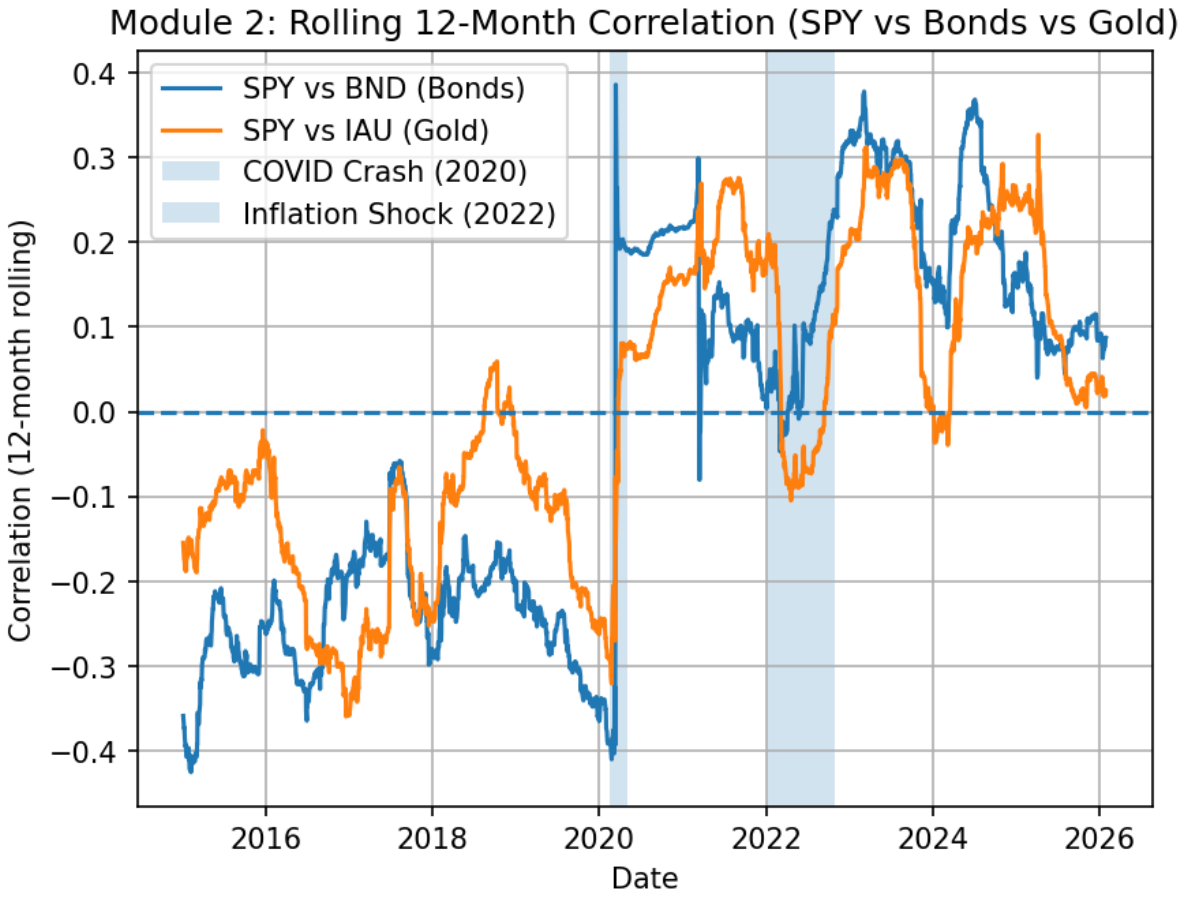
```
# Zero line
plt.axhline(0, linestyle="--")
```

```
# Crisis shading (simple + readable)
plt.axvspan(pd.to_datetime("2020-02-15"), pd.to_datetime("2020-04-30"), alpha=0.2,
label="COVID Crash (2020)")
plt.axvspan(pd.to_datetime("2022-01-01"), pd.to_datetime("2022-10-31"), alpha=0.2,
label="Inflation Shock (2022)")
```

```
plt.title("Module 2: Rolling 12-Month Correlation (SPY vs Bonds vs Gold)")
plt.xlabel("Date")
plt.ylabel("Correlation (12-month rolling)")
plt.legend()
plt.grid(True)
plt.show()
```

Output:

```
[STDERR]
&lt;string&gt;:1: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
```



Cell 6: ■ Code

```
start_2022 = "2022-01-01"
end_2022 = "2022-12-31"
```

```
corr_2022 = corr_df.loc[start_2022:end_2022].agg(["mean", "median", "min", "max"])
display(corr_2022)
```

Output:

Index	SPY vs BND	SPY vs IAU	SPY vs GLD
mean	0.11238173237618859	0.03401574895082725	nan
median	0.09104329937082835	-0.020985240735509347	nan
min	-0.047111096424244124	-0.1049464150381229	nan
max	0.3209722019725131	0.2096452021918039	nan

Cell 7: ■ Markdown

- Bonds “fail” if SPY vs BND correlation becomes positive and high during stress.
- Gold “works” if SPY vs IAU stays closer to 0 or negative during stress.

Cell 8: ■ Markdown