

01_data_ingestion_and_cleaning

January 29, 2026

1 Purpose

Download SPY adjusted close prices and produce clean daily return series for time-series modelling.

Clean daily returns form the basis for volatility estimation, risk forecasting, and position sizing decisions in portfolio management.

2 Assumptions and scope

SPY adjusted close prices assumed to fully account for dividends and stock splits.

Trading days follow US market calendar; non-trading days are excluded.

Missing prices imply non-trading days, not data errors.

Analysis focuses on daily frequency data from 2010 to 2024.

3 Library imports and configuration

```
[1]: import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

4 Data ingestion

```
[2]: # Download SPY daily price data from Yahoo Finance
df = yf.download(
    tickers="SPY",
    start="2010-01-01",
    end="2025-01-01",      # end is exclusive
    auto_adjust=False,    # keep 'Adj Close'
    actions=False
)
df.columns = df.columns.get_level_values(0)
```

```
[*****100%*****] 1 of 1 completed
```

Raw prices are downloaded once and persisted locally to ensure reproducibility of downstream results.

```
[3]: # Ensure datetime index
df.index = pd.to_datetime(df.index)
```

```
[4]: print(df.head(), df.tail())
```

Price Date	Adj Close	Close	High	Low	Open \
2010-01-04	85.027954	113.330002	113.389999	111.510002	112.370003
2010-01-05	85.253036	113.629997	113.680000	112.849998	113.260002
2010-01-06	85.313065	113.709999	113.989998	113.430000	113.519997
2010-01-07	85.673187	114.190002	114.330002	113.180000	113.500000
2010-01-08	85.958298	114.570000	114.620003	113.660004	113.889999

Price Date	Volume
2010-01-04	118944600
2010-01-05	111579900
2010-01-06	116074400
2010-01-07	131091100
2010-01-08	126402800

Price Low	Adj Close	Close	High
2024-12-24	594.320801	601.299988	601.340027
2024-12-26	594.360352	601.340027	602.479980
2024-12-27	588.103882	595.010010	597.780029
2024-12-30	581.392517	588.219971	591.739990
2024-12-31	579.277466	586.080017	590.640015

Price Date	Volume
2024-12-24	33160100
2024-12-26	41219100
2024-12-27	64969300
2024-12-30	56578800
2024-12-31	57052700

5 Initial data checks

```
[5]: # Basic structure check
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3774 entries, 2010-01-04 to 2024-12-31
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
#   ...
```

```

---  -----  -----  -----
0  Adj Close  3774 non-null  float64
1  Close      3774 non-null  float64
2  High       3774 non-null  float64
3  Low        3774 non-null  float64
4  Open       3774 non-null  float64
5  Volume     3774 non-null  int64
dtypes: float64(5), int64(1)
memory usage: 206.4 KB

```

```
[6]: # Check date range
df.index.min(), df.index.max()
```

```
[6]: (Timestamp('2010-01-04 00:00:00'), Timestamp('2024-12-31 00:00:00'))
```

```
[7]: # Check for missing values
df.isna().sum()
```

```
[7]: Price
Adj Close    0
Close        0
High         0
Low          0
Open         0
Volume       0
dtype: int64
```

```
[8]: # Logical Validation
def validate_data(df):
    checks = {
        "Zero Volume Days": (df['Volume'] == 0).sum(),
        "Negative Prices": (df['Adj Close'] <= 0).sum(),
        "Stale Prices (>3 days)": (df['Adj Close'].diff() == 0).rolling(3).
        ↪sum().max()
    }
    return pd.Series(checks)

print(validate_data(df))
```

```

Zero Volume Days          0.0
Negative Prices           0.0
Stale Prices (>3 days)    1.0
dtype: float64

```

6 Handling non-trading days and missing values

```
[9]: df = df.sort_index()
```

Non-trading days are not present in the dataset.

The only missing return observation arises from differencing.

7 Return construction

```
[ ]: # Simple returns
df["adj_return"] = df["Adj Close"].pct_change()

# Log returns
df["adj_log_return"] = np.log(df["Adj Close"]).diff()

df[["Adj Close", "adj_return", "adj_log_return"]].head()
```

```
[ ]: Price      Adj Close  adj_return  adj_log_return
Date
2010-01-04  85.027954      NaN          NaN
2010-01-05  85.253036    0.002647    0.002644
2010-01-06  85.313065    0.000704    0.000704
2010-01-07  85.673187    0.004221    0.004212
2010-01-08  85.958298    0.003328    0.003322
```

We use log returns because they have additive properties and align with time-series assumptions.

```
[ ]: # Drop first
returns_df = df[["Adj Close", "adj_return", "adj_log_return"]].dropna()
returns_df.head()
```

```
[ ]: Price      Adj Close  adj_return  adj_log_return
Date
2010-01-05  85.253036    0.002647    0.002644
2010-01-06  85.313065    0.000704    0.000704
2010-01-07  85.673187    0.004221    0.004212
2010-01-08  85.958298    0.003328    0.003322
2010-01-11  86.078316    0.001396    0.001395
```

8 Save cleaned outputs

```
[12]: # Save raw price data
df.to_csv("../data/raw/SPY_daily_prices.csv")

# Save cleaned returns
returns_df.to_csv("../data/processed/daily_returns.csv")
```

9 Summary

Downloaded daily adjusted SPY prices from 2010–2024

Constructed simple and log return series

Verified data completeness and trading-day structure

Saved cleaned datasets for downstream analysis