Stock Market Time Series Analysis with Pandas

Notebook 05: Resampling and Identifying Long-Term Trends

Part of the comprehensive learning series: Stock Market Time Series Analysis with Pandas

Learning Objectives:

- Master Pandas `.resample()` method for time series aggregation
- Convert daily data to weekly and monthly frequencies
- Apply aggregation functions like `.mean()`, `.sum()`, and `.ohlc()`
- Identify long-term trends by filtering out daily noise
- Understand downsampling techniques for trend analysis

- Daily returns contain a lot of noise that can obscure the true, underlying trends of a stock.
- **Resampling** is a powerful Pandas technique that allows us to change the frequency of our time series data (e.g., from daily to weekly or monthly) and aggregate the data accordingly.

This notebook focuses on:

1. **Downsampling (Aggregation):** Using the `.resample()` method to group daily data into lower frequencies (Weekly 'W', Monthly 'M').
2. **Aggregation Functions:** Applying different functions like `.mean()`, `.sum()`, and the finance-specific `.ohlc()`.
3. **Trend Visualization:** Plotting the resampled data to reveal long-term patterns and seasonality.

In [1]:
```
# Import necessary libraries
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('whitegrid')
```

In [2]:
```
# Suppressing future warnings for cleaner output
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [3]:
```
# Reloading the data
TICKER = 'AAPL'
START_DATE = '2019-01-01'
END_DATE = '2025-01-01'

df = yf.download(TICKER, start=START_DATE, end=END_DATE)

# Clean up MultiIndex columns
df.columns = df.columns.get_level_values(0)

print("Initial daily data shape:", df.shape)
df.head(2)
```

```
[*********************100%***********************]  1 of 1 completed
Initial daily data shape: (1510, 5)
```

Out[3]:

Price	Close	High	Low	Open	Volume
Date					
2019-01-02	37.575191	37.796476	36.697199	36.854239	148158800
2019-01-03	33.832436	34.672357	33.787227	34.258343	365248800

1. Weekly Resampling and Basic Aggregation

- We use the Pandas offset alias **'W'** to resample the data to a weekly frequency.
- The aggregation function we apply determines the resulting value (e.g., mean price, sum of volume).

A. Weekly Mean Close Price

In [8]:
```
# --- Concept: .resample('W').mean() ---
# Groups all daily data points into weekly buckets and
# calculates the average (mean) price within each week.
weekly_mean_price = df['Close'].resample('W').mean()

print("Weekly Mean Close Price (first 5 weeks):")
print(weekly_mean_price.head())
```

```
Weekly Mean Close Price (first 5 weeks):
Date
2019-01-06    35.561451
2019-01-13    36.075244
2019-01-20    36.675327
2019-01-27    36.742428
2019-02-03    38.507450
Freq: W-SUN, Name: Close, dtype: float64
```

B. Weekly Total Volume

In [9]:
```
# --- Code: Weekly Sum ---
# Volume should be summed up, as the weekly volume is the total of all daily trade
weekly_volume = df['Volume'].resample('W').sum()
```

```
print("\nWeekly Total Volume (last 5 weeks):")
print(weekly_volume.tail())
```

```
Weekly Total Volume (last 5 weeks):
Date
2024-12-08    208286500
2024-12-15    192702600
2024-12-22    368202900
2024-12-29    133685900
2025-01-05     75038200
Freq: W-SUN, Name: Volume, dtype: int64
```

2. Monthly OHLC Resampling

- For financial data, a useful resampling technique is **OHLC** (Open, High, Low, Close).
- When aggregating daily data into a month, we want:
  - **Open:** The price on the *first* trading day of the month.
  - **High:** The *highest* price reached during the entire month.
  - **Low:** The *lowest* price reached during the entire month.
  - **Close:** The price on the *last* trading day of the month.

In [10]:
```
# --- Concept: .resample('M').ohlc() ---
# The 'M' offset alias stands for monthly aggregation.
# .ohlc() applies the correct aggregation logic for financial data.

# --- Code: Monthly OHLC ---
monthly_ohlc = df['Close'].resample('M').ohlc()

print("Monthly OHLC Data (last 5 months):")
print(monthly_ohlc.tail())
```

```
Monthly OHLC Data (last 5 months):
                open        high         low       close
Date
2024-08-31  217.097168  228.725510  206.031525  227.939178
2024-09-30  221.738037  231.920654  215.317917  231.920654
2024-10-31  225.162109  235.384521  220.663025  224.863480
2024-11-30  221.877380  236.490494  220.981537  236.490494
2024-12-31  238.742477  258.103729  238.742477  249.534180
```

Insights from OHLC

- Comparing the **Open** and **Close** prices in the `monthly_ohlc` table immediately tells you whether the stock finished the month up or down.
- For example, if a month's Close price is higher than its Open price, the month was bullish, despite any volatility that occurred between the High and Low.

3. Visualization: Revealing Long-Term Trends

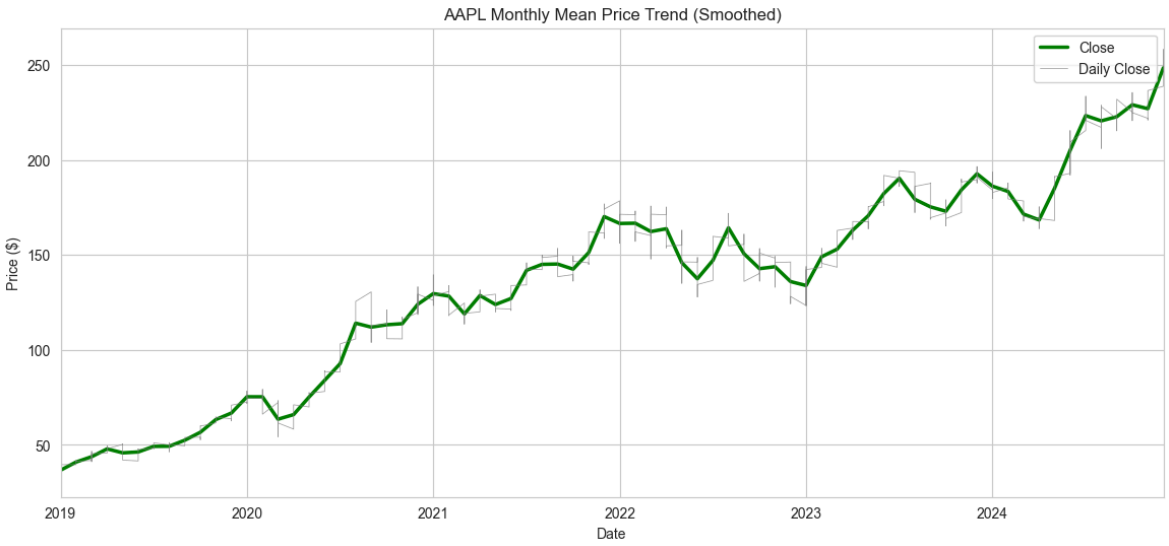
- Plotting: the resampled data smooths the noise and makes long-term trends unmistakable.

A. Monthly Mean Price Trend

In [12]:
```
# --- Code: Plot Monthly Trend ---
monthly_mean_price = df['Close'].resample('M').mean()

plt.figure(figsize=(14, 6))
monthly_mean_price.plot(title=f'{TICKER} Monthly Mean Price Trend (Smoothed)', col
df['Close'].plot(linewidth=0.5, alpha=0.8, color='grey', label='Daily Close')

plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.legend()
plt.show()
```



Visualization 1 Insights

1. **Noise Reduction:** The monthly line is significantly smoother than the daily line, making it easier to confirm the overall **secular trend** (the primary, long-term direction of the stock).
2. **Trend Confirmation:** The plot clearly shows periods where the stock entered a sustained uptrend or a prolonged correction (e.g., late 2021 into 2022).
3. **Trend Speed:** The steepness of the monthly line gives a better, less noisy visual estimate of the *rate* of price change.

B. Monthly Volume Analysis

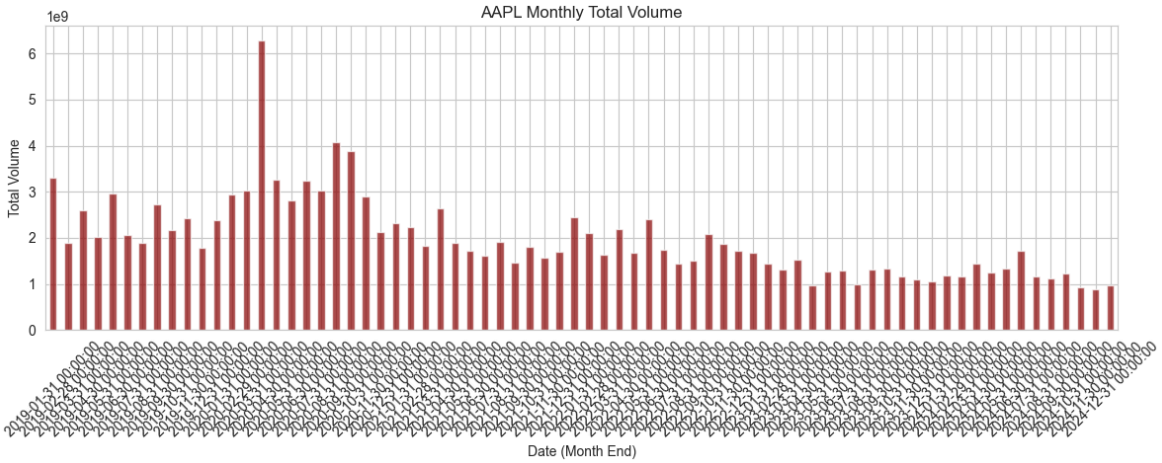
- Plotting the total monthly volume helps identify long-term changes in market interest and liquidity.

In [13]:
```
# --- Code: Plot Monthly Volume ---
monthly_volume = df['Volume'].resample('M').sum()

plt.figure(figsize=(14, 4))
monthly_volume.plot(kind='bar', title=f'{TICKER} Monthly Total Volume', color='dar

plt.xlabel('Date (Month End)')
```

```
plt.ylabel('Total Volume')
plt.xticks(rotation=45)
plt.show()
```



Visualization 2 Insights

1. **Liquidity Trend:** Observe if the total monthly volume is generally increasing or decreasing over the years. Increasing volume is often a sign of growing institutional and public interest in the stock.

2. **Seasonal Volume:** Are there certain months (e.g., January or December) that consistently show higher or lower trading activity? This could indicate tax-related trading or market cycles (which we will explore more in a later notebook).

4. Summary and Next Steps

Key Takeaways

- **Resampling Mastery:** We successfully used `.resample('W')` and `.resample('M')` to change the frequency of our data, demonstrating a core Pandas time series feature.
- **Aggregation Techniques:** Applied different aggregation methods like `.mean()`, `.sum()`, and the finance-specific `.ohlc()` to derive meaningful summary statistics for the new periods.
- **Trend Identification:** The smoothed monthly plots provided a clear view of the stock's long-term trend, filtering out the daily market noise.

Next Notebook Preview

- While resampling provides fixed-period averages, rolling statistics allow us to measure trends based on a **rolling window** of time (e.g., the last 30 days, regardless of the month-end).
- The next notebook will focus on `.rolling()` statistics, particularly **Moving Averages** and **Rolling Volatility**.

About This Project

This notebook is part of the **Stock Market Time Series Analysis with Pandas** repository - a comprehensive, beginner-to-intermediate friendly guide for mastering financial time series analysis using Python and Pandas.

Repository: `stock-time-series-analysis-with-pandas`

Author

Prakash Ukhalkar

GitHub prakash-ukhalkar

Built with ❤️ for the Python community