Stock Market Time Series Analysis with Pandas

Notebook 03: Powerful Time Series Plots for Stock Analysis

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

Learning Objectives:

- Master dual-axis plotting for price and volume analysis
- Create professional candlestick charts using Plotly
- Learn event annotation techniques for storytelling
- Understand advanced financial visualization patterns
- Configure interactive charts for Jupyter notebooks

- Visualizing financial time series data effectively is crucial for identifying trends, patterns, and anomalies.
- While simple line plots give a basic overview, financial analysis demands more specialized chart types and techniques.

In this notebook, we'll dive into **Visualization Mastery**:

1. **Line Plots for Multiple Series:** Overlaying price and volume on a dual-axis chart.
2. **Candlestick Charts (Plotly):** Creating industry-standard candlestick charts to show Open, High, Low, and Close prices for each period.
3. **Annotating Major Events:** Adding text annotations to pinpoint specific events on our charts.
4. **Customizing Plots:** Enhancing readability and aesthetic appeal.

In [1]:
```
# Import necessary libraries
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go # For interactive candlestick charts

# Set a specific theme for cleaner matplotlib/seaborn visualizations
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 (using the same logic as before)
TICKER = 'AAPL'
START_DATE = '2019-01-01'
```

```
END_DATE = '2025-01-01'

# Download historical stock data from Yahoo Finance
df = yf.download(TICKER, start=START_DATE, end=END_DATE)

# Clean up MultiIndex columns (as established in Notebook 01)
df.columns = df.columns.get_level_values(0)

print("\nInitial DataFrame head:")
df.head()
```

```
[**********100%**********]  1 of 1 completed
Initial DataFrame head:
```

Out[3]:

Price	Close	High	Low	Open	Volume
Date					
2019-01-02	37.575203	37.796487	36.697210	36.854250	148158800
2019-01-03	33.832447	34.672369	33.787238	34.258355	365248800
2019-01-04	35.276733	35.345738	34.215531	34.389224	234428400
2019-01-07	35.198204	35.412351	34.715190	35.381418	219111200
2019-01-08	35.869202	36.123797	35.338600	35.586054	164101200

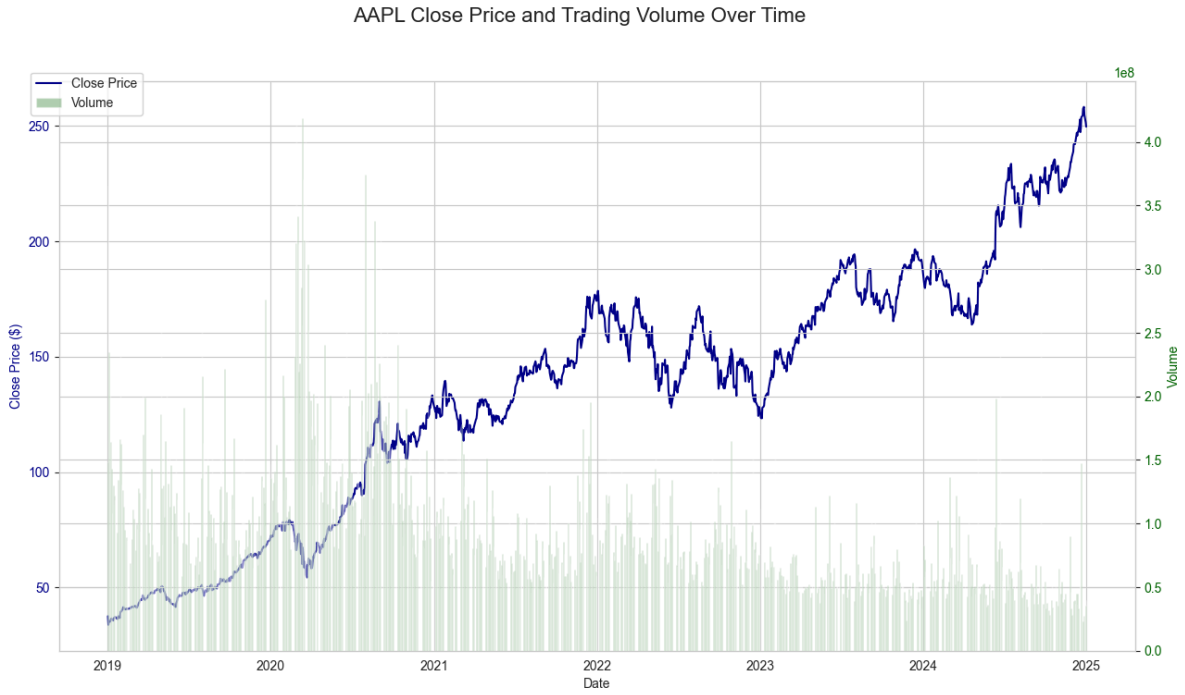
1. Overlaying Price and Volume (Dual-Axis Plot)

- Volume is a crucial indicator of conviction behind price movements.
- A dual-axis plot allows us to visualize both on the same timeline, observing their relationship.

In [4]:

```
# --- Concept: Dual-Axis Plotting ---
# Matplotlib allows creating a second y-axis that shares the same x-axis,
# perfect for comparing series with vastly different scales (like price and volume).

# --- Code: Create Dual-Axis Plot ---
# Create a figure and a set of subplots
# Set figure size for better visibility
# Axis 1: Close Price
# Axis 2: Volume

fig, ax1 = plt.subplots(figsize=(15, 8))

# Plotting Close Price on the first y-axis
color = 'darkblue'
ax1.set_xlabel('Date')
ax1.set_ylabel('Close Price ($)', color=color)
ax1.plot(df.index, df['Close'], color=color, linewidth=1.5, label='Close Price')
ax1.tick_params(axis='y', labelcolor=color)

# Create a second y-axis that shares the same x-axis
ax2 = ax1.twinx() # instantiate a second axes that shares the same x-axis
color = 'darkgreen'
ax2.set_ylabel('Volume', color=color)
ax2.bar(df.index, df['Volume'], color=color, alpha=0.3, label='Volume')
ax2.tick_params(axis='y', labelcolor=color)

# Add title and legend
```

```
# suptitle for the entire figure
fig.suptitle(f'{TICKER} Close Price and Trading Volume Over Time', fontsize=16)
fig.legend(loc='upper left', bbox_to_anchor=(0.1, 0.9))

plt.show()
```



Visualization 1 Insights

1. **Price-Volume Relationship:** We can often observe that significant price movements (sharp drops or rallies) are accompanied by higher trading volume.
    - This suggests stronger conviction or panic in the market.
2. **Volume Spikes:** Peaks in volume often coincide with major news events, earnings reports, or market turning points.
    - Sustained high volume can confirm a trend, while decreasing volume on a rising price might indicate weakening momentum.

2. Candlestick Charts with Plotly (Interactive)

- Candlestick charts are the industry standard for financial analysis.
- They concisely display four key prices (Open, High, Low, Close) for each period, offering a rich visual summary of market sentiment.

In [9]:
```
# --- Additional Setup for Plotly Rendering ---
# Configure Plotly to work better in Jupyter notebooks
import plotly.io as pio

# Set the default renderer - try different options if one doesn't work
# pio.renderers.default = "notebook"  # Try this first
# pio.renderers.default = "colab"     # Alternative option
# pio.renderers.default = "browser"   # Opens in browser
pio.renderers.default = "plotly_mimetype+notebook"  # Best for VS Code

print("Plotly renderer configured for VS Code")
```

```
Plotly renderer configured for VS Code
```

In [ ]:
```python
# --- Concept: Candlestick Structure ---

# - **Body:** The range between Open and Close.
# Green/white if Close > Open (bullish), Red/black if Close < Open (bearish).

# - **Wicks/Shadows:** The lines extending from the body to the High and Low,
# showing the full price range for the period.

# --- Code: Create Candlestick Chart ---
# For better readability, let's plot a smaller, more recent period (e.g., last 6 months
# First, let's check what dates we actually have in our data
print("Data date range:")
print(f"Start: {df.index.min()}")
print(f"End: {df.index.max()}")

# Get the last 6 months of data using a more reliable method
df_recent = df.tail(125)  # Approximately 6 months of trading days (125 trading days ≈
print(f"\nRecent data shape: {df_recent.shape}")
print(f"Recent data date range: {df_recent.index.min()} to {df_recent.index.max()}")

# Create the candlestick chart using Plotly
fig = go.Figure(data=[
    go.Candlestick(
        x=df_recent.index,
        open=df_recent['Open'],
        high=df_recent['High'],
        low=df_recent['Low'],
        close=df_recent['Close'],
        name='AAPL'
    )
])

# Update layout for better aesthetics
fig.update_layout(
    title=f'{TICKER} Candlestick Chart (Last 6 Months)',
    yaxis_title='Price ($)',
    xaxis_title='Date',
    xaxis_rangeslider_visible=False, # Hide range slider for cleaner view
    height=600
)

# Display the chart - this will work with the Jupyter Renderers extension
fig.show()
```

```
Data date range:
Start: 2019-01-02 00:00:00
End: 2024-12-31 00:00:00

Recent data shape: (125, 5)
Recent data date range: 2024-07-05 00:00:00 to 2024-12-31 00:00:00
```

In [16]:
```
# Alternative: If you still have rendering issues, uncomment the lines below
# This will save a static image that can be displayed
fig.write_image("candlestick_chart.png", width=1200, height=600)
from IPython.display import Image
Image("candlestick_chart.png")
```

Out[16]:

Visualization 2 Insights

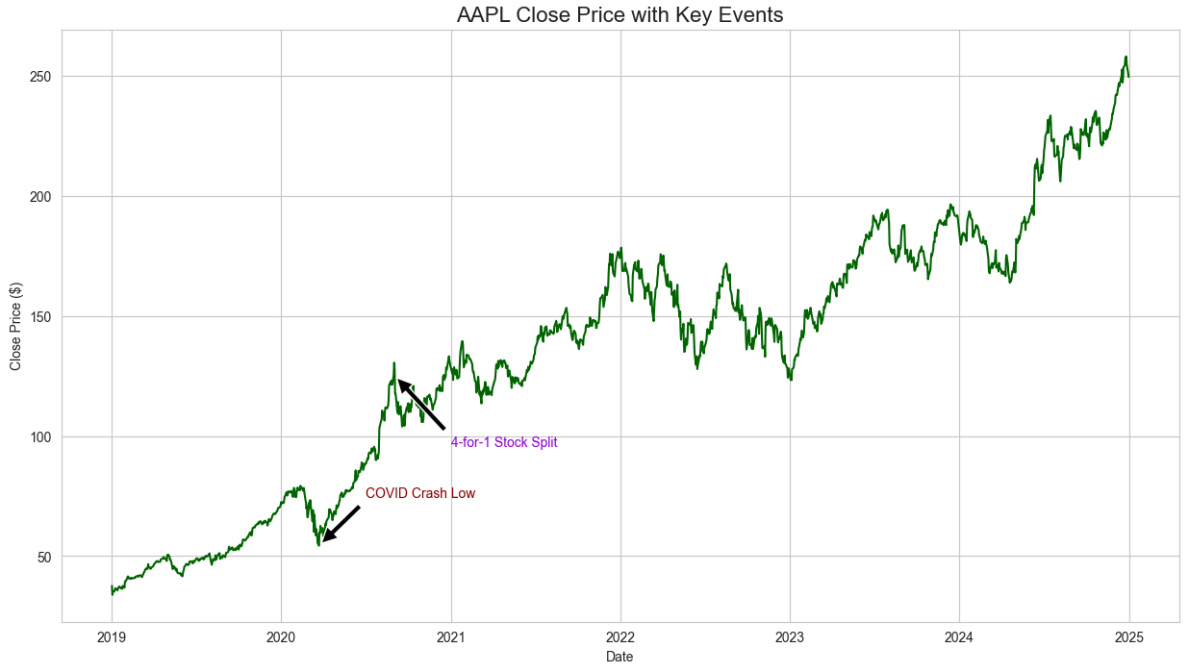
1. **Intraday Dynamics:** Candlesticks reveal the battle between buyers and sellers within a single day. Long green bodies show strong buying pressure, while long red bodies indicate selling pressure.
2. **Wick Interpretation:** Long upper wicks indicate that buyers pushed the price high, but sellers drove it back down. Long lower wicks show initial selling, but buyers pushed it back up.
3. **Patterns:** Traders look for specific candlestick patterns (e.g., Doji, Hammer, Engulfing patterns) to predict future price movements. This chart provides the raw visual data for such analysis.

3. Annotating Major Events

- Adding annotations to charts helps tell a story and highlight significant market catalysts or historical events.

In [13]:
```
# --- Concept: Adding Text Annotations ---
# Matplotlib's plt.annotate() allows placing text with an arrow,
# while plt.text() places text at a coordinate.

# --- Code: Plot with Annotations ---
# Create a basic line plot of Close Price
plt.figure(figsize=(15, 8))
plt.plot(df.index, df['Close'], color='darkgreen', linewidth=1.5)

# Add title and labels
plt.title(f'{TICKER} Close Price with Key Events', fontsize=16)
plt.xlabel('Date')
plt.ylabel('Close Price ($)')

# Example 1: Annotating the COVID crash
covid_crash_date = '2020-03-23'
covid_crash_price = df.loc[covid_crash_date]['Close']
plt.annotate(
    'COVID Crash Low',
    xy=(pd.Timestamp(covid_crash_date), covid_crash_price), # xy coordinates of the poi
    xytext=(pd.Timestamp('2020-07-01'), covid_crash_price + 20), # text position
    arrowprops=dict(facecolor='black', shrink=0.05), # arrow properties using a diction
    fontsize=10,
    color='darkred'
)

# Example 2: Annotating a stock split (Apple had a 4-for-1 split in Aug 2020)
# Note: The historical data from yfinance is *already adjusted* for splits,
# so the graph won't show a sudden drop.
# This annotation is purely for historical context.
stock_split_date = '2020-08-31'
stock_split_price = df.loc[stock_split_date]['Close']
plt.annotate(
    '4-for-1 Stock Split',
    xy=(pd.Timestamp(stock_split_date), stock_split_price),
    xytext=(pd.Timestamp('2021-01-01'), stock_split_price - 30),
    arrowprops=dict(facecolor='black', shrink=0.05),
    fontsize=10,
    color='darkviolet'
)
```

```
plt.show()
```



Visualization 3 Insights

1. **Narrative Enhancement:** Annotations turn a raw plot into a story. They help viewers immediately grasp the significance of specific data points or periods without needing external context.
2. **Historical Context:** For financial data, marking events like market crashes, stock splits, or major product launches makes the chart a powerful historical record.
3. **Clarifying Adjusted Prices:** The stock split annotation is a good example of clarifying that historical data is often *adjusted* (prices before the split are divided by the split ratio), so you don't see a literal jump or drop *on the adjusted chart*.

4. Summary and Next Steps

Key Takeaways

- **Enhanced Visualization:** We moved beyond basic line plots to create more informative charts crucial for financial analysis.
- **Dual-Axis Power:** Successfully overlaid price and volume, gaining insights into their correlated movements.
- **Candlestick Mastery:** Utilized Plotly to generate interactive candlestick charts, revealing intraday price dynamics.
- **Storytelling with Annotations:** Added contextual annotations to highlight significant market events, making charts more communicative.

Next Notebook Preview

- With our data loaded, cleaned, and visualized, it's time to quantify performance and risk.
- The next notebook will focus on **calculating and analyzing various types of returns** (percent, log, cumulative), which are fundamental metrics in financial time series analysis.

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