Stock Market Time Series Analysis with Pandas

Notebook 07: Volume and Liquidity Analysis

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

Learning Objectives:

- Analyze trading volume distribution and patterns
- Calculate Volume-Weighted Average Price (VWAP)
- Identify high-volume anomalies and market events
- Understand price-volume relationships
- Master volume-based liquidity analysis techniques

- Price tells you *where* the market is going, but **Volume** tells you *how many participants* are driving that move.
- High volume suggests strong conviction, while low volume suggests weakness or uncertainty.

This notebook focuses on:

1. **Volume Distribution:** Analyzing the spread of daily trade volume.
2. **Volume-Weighted Average Price (VWAP):** Calculating this crucial institutional trading benchmark.
3. **High Volume Anomaly:** Identifying days with unusually high trading activity.
4. **Price vs. Volume:** Visualizing their relationship on a dual-axis plot.

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

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

print("Initial DataFrame head:")
print(df[['Close', 'Volume']].head())
```

```
[**********100%**********]  1 of 1 completed
Initial DataFrame head:
Price           Close     Volume
Date
2019-01-02  37.575207  148158800
2019-01-03  33.832439  365248800
2019-01-04  35.276722  234428400
2019-01-07  35.198200  219111200
2019-01-08  35.869190  164101200
```

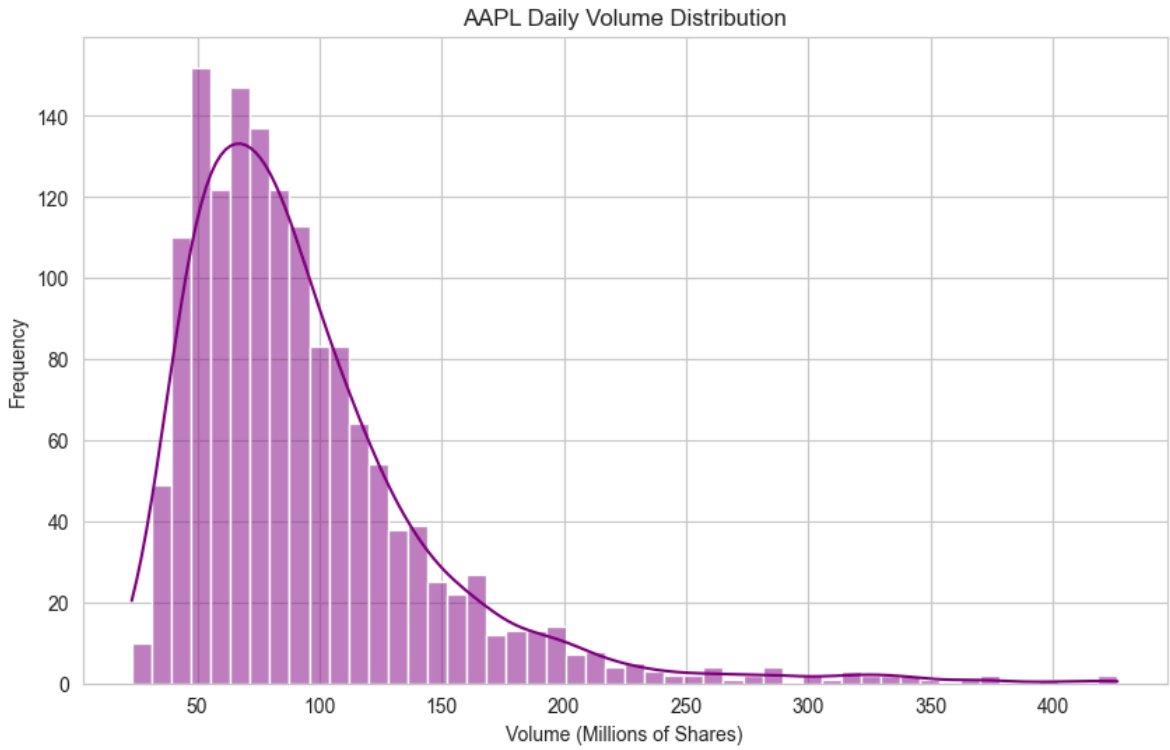
1. Analyzing the Volume Distribution

- Understanding the distribution of volume helps us determine what constitutes 'normal' activity and what qualifies as a high-volume anomaly.

In [4]:
```
# --- Code: Volume Descriptive Stats ---
print("Volume Statistics (in millions of shares):")
print(df['Volume'].describe() / 1_000_000)

# --- Code: Volume Distribution Plot ---
plt.figure(figsize=(10, 6))
sns.histplot(df['Volume'] / 1_000_000, bins=50, kde=True, color='purple')
plt.title(f'{TICKER} Daily Volume Distribution')
plt.xlabel('Volume (Millions of Shares)')
plt.ylabel('Frequency')
plt.show()
```

```
Volume Statistics (in millions of shares):
count      0.001510
mean      94.167751
std       52.325543
min       23.234700
25%       59.144200
50%       81.521050
75%      112.392050
max      426.510000
Name: Volume, dtype: float64
```



Insights on Volume Distribution

1. **Skewness:** Volume data is almost always **highly skewed to the right**. Most days have moderate volume (around the mean/median), but there are a few days with extremely high volume (the long tail).

2. **Liquidity Baseline:** The Median (50th percentile) provides a better measure of the typical daily trading volume than the Mean, as the mean is heavily pulled up by extreme spikes.

2. Calculating Volume-Weighted Average Price (VWAP)

- **VWAP** is a crucial benchmark used by institutional traders.
- It represents the average price an investor paid for the stock, weighted by the volume traded at each price.
- Traders often compare the closing price to the VWAP to evaluate trade execution.

In [5]:
```
# --- Concept: Calculating Daily VWAP ---
# The daily VWAP is calculated using the Typical Price (TP) and Volume (V):
# TP = (High + Low + Close) / 3
# VWAP = Sum(TP * V) / Sum(V)  -> For a single day, this simplifies to TP.
# For historical data, we typically use a simple proxy for the price, which is TP.

# --- Code: Calculate Typical Price ---
df['Typical_Price'] = (df['High'] + df['Low'] + df['Close']) / 3

# VWAP Calculation (for a rolling window, which makes more sense for a daily chart)
# The rolling function needs the product of (TP * Volume) and the sum of Volume
window = 30
df['TP_Volume'] = df['Typical_Price'] * df['Volume']
```

```
# Calculate Rolling VWAP (using a 30-day window for long-term trend)
rolling_tp_vol_sum = df['TP_Volume'].rolling(window=window).sum()
rolling_vol_sum = df['Volume'].rolling(window=window).sum()
df['VWAP_30d'] = rolling_tp_vol_sum / rolling_vol_sum

print(f"Rolling {window}-day VWAP (last 5 rows):")
print(df[['Close', 'VWAP_30d']].tail())
```

```
Rolling 30-day VWAP (last 5 rows):
Price            Close    VWAP_30d
Date
2024-12-24  257.286652  239.589797
2024-12-26  258.103729  240.411541
2024-12-27  254.685867  241.424010
2024-12-30  251.307861  242.160893
2024-12-31  249.534180  243.004160
```

Insights on VWAP

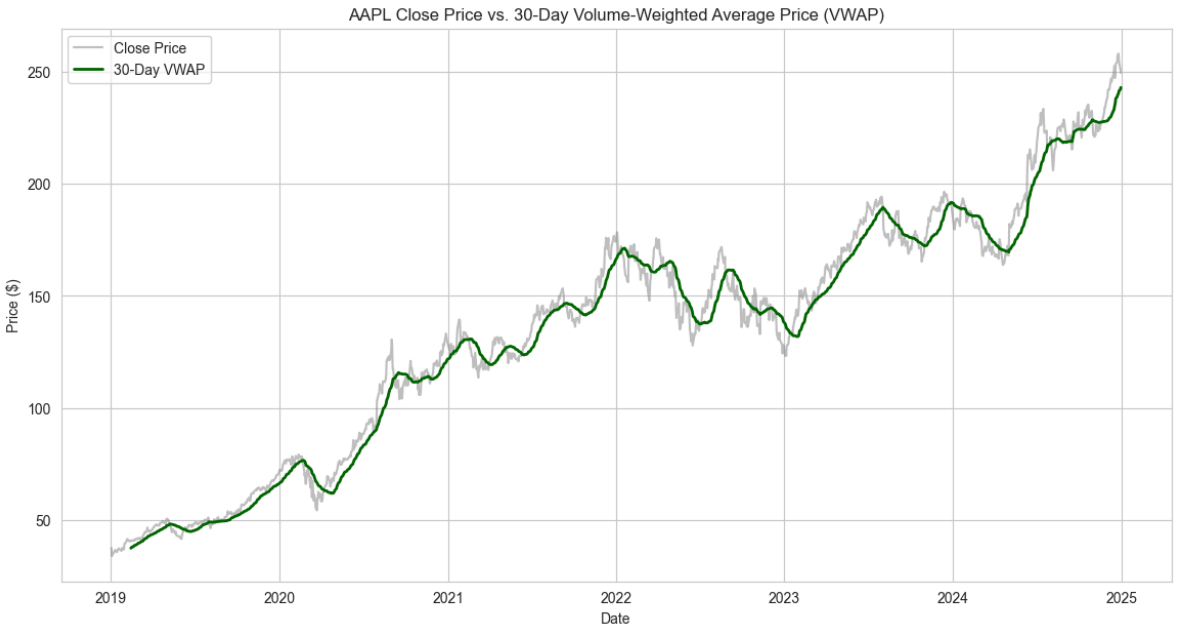
- VWAP is a more **representative average** than the Simple Moving Average (SMA) because it gives higher importance to days when more shares were traded.
- If the price closes *above* the VWAP, it suggests bullish strength, and vice-versa.

3. Visualization: VWAP vs. Price

- Visualizing the VWAP against the price helps us understand the price's position relative to its volume-weighted average over the period.

In [6]:
```
# --- Code: Plot VWAP and Close Price ---
plt.figure(figsize=(14, 7))
plt.plot(df['Close'], label='Close Price', color='grey', alpha=0.5)
plt.plot(df['VWAP_30d'].dropna(), label='30-Day VWAP', color='darkgreen', linewidth=2

plt.title(f'{TICKER} Close Price vs. 30-Day Volume-Weighted Average Price (VWAP)')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.legend()
plt.show()
```



Visualization 1 Insights

1. **Trend Confirmation:** The VWAP line acts similarly to an SMA but can react slightly differently due to volume weighting. When the Close Price is consistently *above* the VWAP, it confirms a strong uptrend backed by trading volume.

2. **Crossovers:** Crossovers between the price and VWAP signal a shift in the short-term trend's conviction, making them potential entry/exit points for traders.

4. High Volume Anomaly Detection

- We can use the statistical properties found in Section 1 (like the 90th percentile) to flag days with abnormally high trading volume.

In [7]:
```
# --- Concept: Using Quantiles ---
# Find the volume level that is exceeded on only 10% of trading days.

# Calculate the 90th percentile volume threshold
# quantile function gives the value below which a given percentage of observations fa
volume_threshold = df['Volume'].quantile(0.90)

# --- Code: Anomaly Detection and Slicing ---
high_volume_days = df[df['Volume'] > volume_threshold]

print(f"Threshold for high volume (90th percentile): {volume_threshold:,.0f} shares")
print("\nFirst 5 High Volume Days and their Close Price:")
print(high_volume_days[['Close', 'Volume']].head())
```

```
Threshold for high volume (90th percentile): 155,058,120 shares

First 5 High Volume Days and their Close Price:
Price           Close     Volume
Date
2019-01-03  33.832439  365248800
2019-01-04  35.276722  234428400
2019-01-07  35.198200  219111200
2019-01-08  35.869190  164101200
2019-01-09  36.478317  180396400
```

Anomaly Insights

- By identifying these high-volume days, we can investigate the news or events that drove such intense trading.

- Often, high volume occurs on days with major price changes (either large gains or large losses), confirming the intensity of market reaction.

5. Summary and Next Steps

Key Takeaways

- **Volume Interpretation:** We analyzed the skewed distribution of trading volume and established a baseline for 'normal' liquidity.

- **VWAP Calculation:** We successfully implemented a **Rolling VWAP** calculation, demonstrating a practical application of combined price and volume data.
- **Anomaly Detection:** We used Pandas' `.quantile()` method to quickly flag days with high trading activity for further investigation.

Next Notebook Preview

- So far, we have focused on a single stock (AAPL).
- The next step is to expand our analysis to the portfolio level.
- We will load multiple stocks, normalize their prices to compare performance, and analyze the **correlation** between their daily returns.

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
