Advanced Task: DataFrames and yfinance - Multi-Asset Portfolio Analysis

Objective:

To develop a comprehensive Python script that downloads historical price data for multiple assets using yfinance, processes and cleans the data, and performs an in-depth analysis of a diversified portfolio's performance. This task will require advanced DataFrame manipulations, including handling missing data, adjusting for corporate actions, calculating portfolio metrics, and visualizing results.

Task Description:

1. **Data Acquisition:**
    - Use the yfinance library to download historical daily price data for at least 10 different assets (e.g., stocks, ETFs, cryptocurrencies) over a 10-year period.
    - Ensure that the data includes both adjusted and unadjusted closing prices.
2. **Data Cleaning and Preparation:**
    - Handle missing data by:
        - Interpolating minor gaps in the time series.
        - Dropping or forward-filling rows where data is missing for more than a certain threshold (e.g., 5 consecutive days).
    - Adjust prices for corporate actions (splits, dividends) to ensure consistency across the time series.
3. **Portfolio Construction:**
    - Create a DataFrame that represents a diversified portfolio by assigning different weights to each asset. The weights should sum to 1.
    - Calculate the daily portfolio returns based on these weights and the adjusted closing prices of the assets.
4. **Performance Metrics Calculation:**
    - Calculate the following performance metrics for the portfolio:
        - **Cumulative Return:** The overall return of the portfolio over the entire period.
        - **Annualized Return and Volatility:** Compute these metrics to assess the portfolio's risk-return profile.
        - **Sharpe Ratio:** Adjust the portfolio's returns for risk by comparing them to a risk-free rate (e.g., 3-month Treasury yield).
        - **Drawdown Analysis:** Calculate the maximum drawdown and the duration of drawdowns to understand the portfolio's downside risk.
5. **Correlation and Risk Analysis:**

- Create a correlation matrix of the assets in the portfolio using daily returns. Use this matrix to analyze diversification benefits.
- Perform a rolling correlation analysis to observe how the relationships between assets change over time.

6. **Advanced Data Visualization:**
   - Plot the following using Matplotlib or Seaborn:
     - The cumulative return of the portfolio over time.
     - A heatmap of the correlation matrix with annotations for correlation coefficients.
     - Rolling volatility of the portfolio and individual assets to show risk dynamics over time.
     - A drawdown curve that visualizes the periods where the portfolio experienced losses relative to its peak.
7. **Optimization Challenge (Optional but Recommended):**
   - Implement a simple portfolio optimization routine using mean-variance optimization. Adjust asset weights to maximize the Sharpe ratio or minimize risk.
   - Compare the optimized portfolio's performance metrics to the original portfolio and provide insights.

Deliverables:

- A well-documented Python script or Jupyter Notebook that includes all steps of the task.
- Plots and tables summarizing the portfolio’s performance, risk analysis, and any optimizations performed.
- A brief report or markdown cells within the notebook explaining the key findings and insights derived from the analysis.