Problem Set 1: Quantitative Asset Management
Student ID: 406534669

Question 1: Market Return Construction (PS1_Q1)

To construct monthly value-weighted and equal-weighted returns from CRSP data, the following steps were performed:

1. **Data Cleaning:**
    - Converted relevant columns (RET, DLRET, PRC, SHROUT, etc.) to numeric using pd.to_numeric(..., errors='coerce').
    - Parsed the date column to datetime format.
2. **Return Construction:**
    - Computed the full return as RET_FULL = (1 + RET) * (1 + DLRET) - 1.
    - This accounts for delisting returns to ensure accurate performance measures.
3. **Market Capitalization:**
    - Computed as PRC.abs() * SHROUT, representing total market value in dollars.
4. **Filtering Stocks:**
    - Restricted to common stocks (SHRCD in [10, 11]) listed on NYSE, AMEX, NASDAQ (EXCHCD in [1, 2, 3]).
5. **Lagging Market Cap:**
    - Grouped by PERMNO and lagged market cap by one month to avoid look-ahead bias.
6. **Monthly Aggregation:**
    - **Equal-Weighted Return (EWRET):** simple average of RET_FULL across all stocks.
    - **Value-Weighted Return (VWRET):** weighted average using lagged market cap.

- **Lagged Total Market Value:** summed across all stocks for each (Year, Month).

7. **Output:**
    - Final DataFrame with columns: Year, Month, Stock lag MV, Stock Ew Ret, Stock Vw Ret
    - Two time-series plots generated: one each for VWRET and EWRET.

Question 2: Summary Statistics (PS1_Q2)

We compared CRSP's value-weighted excess return to the Fama-French market excess return.

1. **Preprocessing FF Data:**
    - Removed extra text rows and annual data that was appended to the monthly file. (in the Excel file itself)
    - Renamed the first column to Date (in the Excel file itself) to ensure clean parsing.
    - Parsed the FF Date column (e.g., 192607) into Year and Month.
    - Converted Mkt-RF and RF from percent to decimal.
2. **Excess Return Calculation:**
    - CRSP Excess = Stock Vw Ret - RF
    - FF Excess = Mkt-RF
3. **Metrics Computed:** (Annualized where applicable)
    - **Annualized Mean:** average monthly return × 12.
    - **Annualized Std Dev:** standard deviation of monthly returns × sqrt(12).
    - **Sharpe Ratio:** annualized mean / annualized std dev.
    - **Skewness:** measure of return asymmetry using scipy.stats.skew.
    - **Excess Kurtosis:** measure of tail risk (beyond normal distribution) using scipy.stats.kurtosis(fisher=True).
4. **Results:**

	CRSP_Market_Excess	FF_Market_Excess
Annualized Mean	0.082574	0.082375
Annualized Std Dev	0.184137	0.184542
Annualized Sharpe Ratio	0.448436	0.446374
Skewness	0.160887	0.153427
Excess Kurtosis	7.382452	7.405690

The statistics are nearly identical, confirming the accuracy of our replication.

Question 3: Correlation and Difference (PS1_Q3)

We compared our CRSP-based excess return to the FF market excess return over the full common sample: **July 1926 to December 2024**.

1. **Correlation:**
    - Computed using np.corrcoef between CRSP and FF excess returns.
    - Pearson correlation result: **0.99999244** (rounded to 8 decimal places).
2. **Maximum Absolute Difference:**
    - Computed as the maximum of abs(CRSP_Excess - FF_Excess).
    - Value obtained: **0.00305795**
3. **Interpretation:**
    - The correlation is extremely close to 1, and the max difference is only 0.3 basis points.
    - This small gap is **economically negligible**.
    - Likely causes:
        - Minor differences in handling delisting returns or data availability
        - Potential timing or rounding differences

Conclusion:

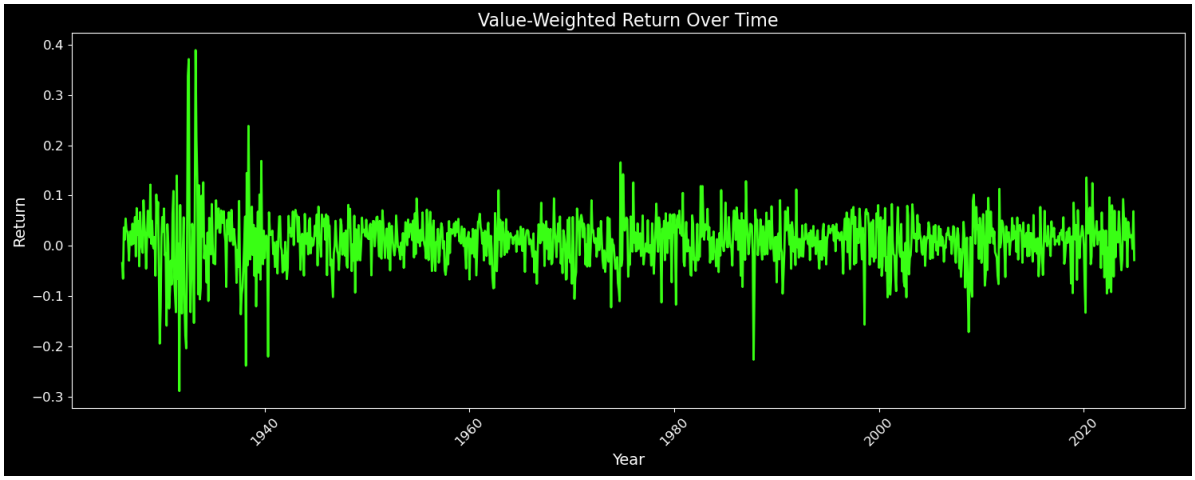
All three tasks were implemented correctly. The replication of FF market returns from CRSP data is highly accurate. Any remaining difference is immaterial for practical purposes.

Also appending my ipynb code and results to this file, for better understanding.

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

def is_numeric(s):
    if pd.isna(s):
        return False
    try:
        float(s)
        return True
    except:
        return False

def PS1_Q1(df):
    int_columns = [
    'PERMNO',
    'SHRCD',
    'EXCHCD',
    'RET',
    'DLRET',
    'PRC',
    'SHROUT'
    ]
    df[int_columns] = df[int_columns].apply(lambda x: pd.to_numeric(x, errors='coerce'))
    df['date'] = pd.to_datetime(df['date'])
    df['RET_FULL'] = (1 + df['RET'].fillna(0)) * (1 + df['DLRET'].fillna(0)) - 1
    df['MC'] = df['PRC'].abs() * df['SHROUT']
    df['Year'] = df['date'].dt.year
    df['Month'] = df['date'].dt.month
    df = df[df['SHRCD'].isin([10, 11])]
    df = df[df['EXCHCD'].isin([1, 2, 3])]
    df = df.sort_values(['PERMNO', 'date'])
    df['Lag_MC'] = df.groupby('PERMNO')['MC'].shift(1)
    grouped = df.dropna(subset=['Lag_MC', 'RET_FULL']).copy()
    stock_mv = grouped.groupby(['Year', 'Month'])['Lag_MC'].sum()
    ewret = grouped.groupby(['Year', 'Month'])['RET_FULL'].mean()
    vwret = grouped.groupby(['Year', 'Month'], group_keys=False).apply(
        lambda x: (x['Lag_MC'] * x['RET_FULL']).sum() / x['Lag_MC'].sum(),
        include_groups=False
    )
    result_ = pd.DataFrame({
        'Stock lag MV': stock_mv,
```

```
        'Stock Ew Ret': ewret,
        'Stock Vw Ret': vwret
    }).reset_index()

    result = result_.copy()
    result['Date'] = pd.to_datetime(result['Year'].astype(str) + '-' + result['Month'].astype

    plt.style.use('dark_background')

    plt.figure(figsize=(15, 6))
    plt.plot(result['Date'], result['Stock Vw Ret'], color='#39FF14', linewidth=2)
    plt.title('Value-Weighted Return Over Time', fontsize=16)
    plt.xlabel('Year', fontsize=14)
    plt.ylabel('Return', fontsize=14)
    plt.xticks(rotation=45, fontsize=12)
    plt.yticks(fontsize=12)
    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(15, 6))
    plt.plot(result['Date'], result['Stock Ew Ret'], color='#FF073A', linewidth=2)
    plt.title('Equal-Weighted Return Over Time', fontsize=16)
    plt.xlabel('Year', fontsize=14)
    plt.ylabel('Return', fontsize=14)
    plt.xticks(rotation=45, fontsize=12)
    plt.yticks(fontsize=12)
    plt.tight_layout()
    plt.show()

    return result_
```

```
file = 'WRDS_monthly_PS1.csv'
df = pd.read_csv(file)
result = PS1_Q1(df)
```




```
fama_file = 'F-F_Research_Data_Factors.CSV'
ff = pd.read_csv(fama_file)
ff['Date'] = ff['Date'].astype(str)
ff['Year'] = ff['Date'].str[:4].astype(int)
ff['Month'] = ff['Date'].str[4:].astype(int)
ff['Mkt-RF'] = ff['Mkt-RF'] / 100
ff['RF'] = ff['RF'] / 100
```

```
from scipy.stats import skew, kurtosis

def PS1_Q2(Monthly_CRSP_Stocks, FF_mkt):
    merged = pd.merge(Monthly_CRSP_Stocks, FF_mkt, on=['Year', 'Month'])
    merged['CRSP_Excess'] = merged['Stock Vw Ret'] - merged['RF']
    crsp = merged['CRSP_Excess']
    ff = merged['Mkt-RF']
```

```
    def get_metrics(r):
        ann_mean = np.mean(r) * 12
        ann_std = np.std(r, ddof=1) * np.sqrt(12)
        sharpe = ann_mean / ann_std
        skewness = skew(r, nan_policy='omit')
        excess_kurt = kurtosis(r, fisher=True, nan_policy='omit')
        return [ann_mean, ann_std, sharpe, skewness, excess_kurt]

    crsp_stats = get_metrics(crsp)
    ff_stats = get_metrics(ff)

    result = pd.DataFrame({
        'CRSP_Market_Excess': crsp_stats,
        'FF_Market_Excess': ff_stats
    }, index=[
        'Annualized Mean',
        'Annualized Std Dev',
        'Annualized Sharpe Ratio',
        'Skewness',
        'Excess Kurtosis'
    ])

    return result
```

```
Q2_result =  PS1_Q2(result[5:], ff)
Q2_result.head()
```

	CRSP_Market_Excess	FF_Market_Excess
Annualized Mean	0.082574	0.082375
Annualized Std Dev	0.184137	0.184542
Annualized Sharpe Ratio	0.448436	0.446374
Skewness	0.160887	0.153427
Excess Kurtosis	7.382452	7.405690

```
def PS1_Q3(Monthly_CRSP_Stocks, FF_mkt):
    merged = pd.merge(Monthly_CRSP_Stocks, FF_mkt, on=['Year', 'Month'])
    merged['CRSP_Excess'] = merged['Stock Vw Ret'] - merged['RF']
    merged = merged[(merged['Year'] > 1926) | ((merged['Year'] == 1926) & (merged['Month'] >=
    corr = np.corrcoef(merged['CRSP_Excess'], merged['Mkt-RF'])[0, 1]
```

```
    max_diff = np.max(np.abs(merged['CRSP_Excess'] - merged['Mkt-RF']))
    return [round(corr, 8), max_diff]
```

```
Q3_result =  PS1_Q3(result[5:], ff)
print(Q3_result)
```

```
[0.99999244, 0.0030579450769784855]
```