In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1. Estimate Beta

estimate every year using daily stock returns using 1 month, 3 month, 6 month, 12 month, 24 month

In [ ]:
```
#import that damn file
path = r"C:\Users\morganhales\Downloads\dsf_1995_2022.csv"
df = pd.read_csv(path, parse_dates=['date'])
```

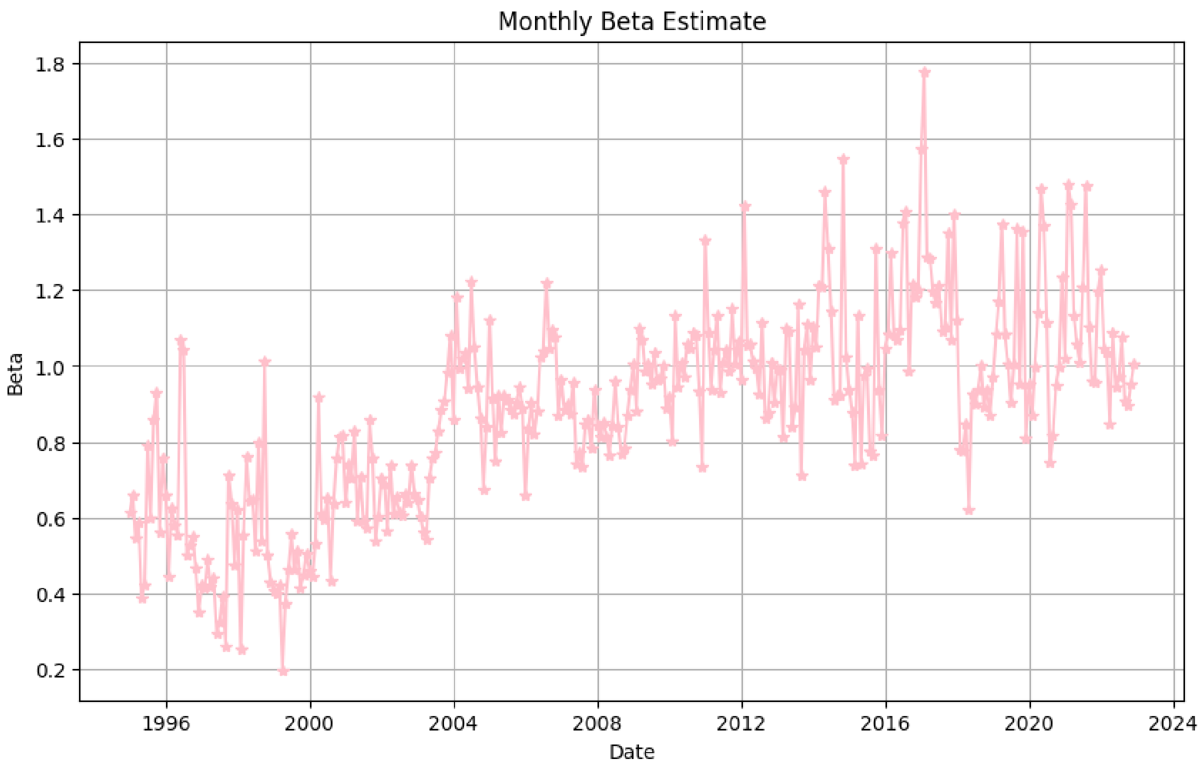
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\843179714.py:3: DtypeWarnin
g: Columns (5,6,9) have mixed types. Specify dtype option on import or set low_memor
y=False.
  df = pd.read_csv(path, parse_dates=['date'])
```

In [ ]:
```
#Format the data correctly
df['RET'] = pd.to_numeric(df['RET'], errors='coerce')

#take care of missing values
df['RET'] = df['RET'].fillna(0)
df['vwretd'] = df['vwretd'].fillna(0)

#extract year and month
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
```

In [ ]:
```
#Now we will compute betas
grouped = df.groupby(['year', 'month'])

betas = {}
for(year, month), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year,month)] = beta

#convert to dataframe
betas_monthly_df = pd.DataFrame(list(betas.items()), columns = ['Year_Month', 'Beta
#make a date column ffrom the year_month column
betas_monthly_df["Date"] = pd.to_datetime(betas_monthly_df['Year_Month'].apply(lamb
betas_monthly_df = betas_monthly_df.sort_values('Date')
```

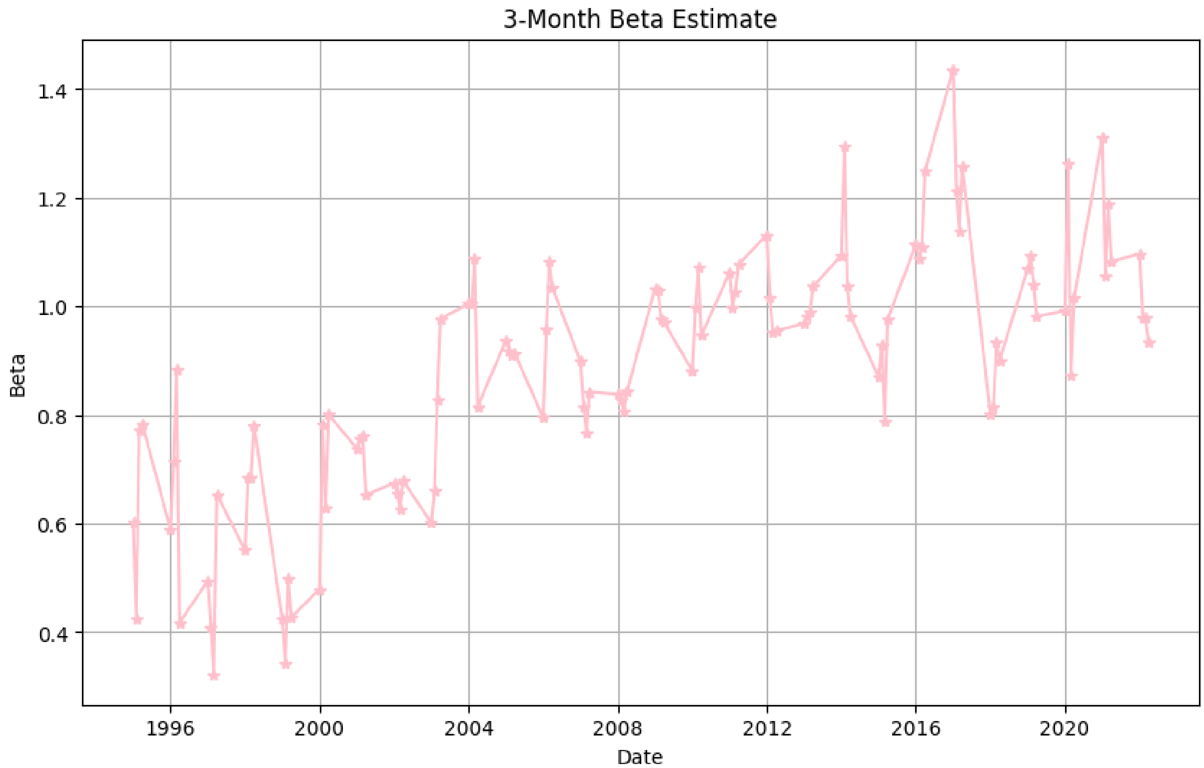
In [ ]:
```
#Monthly Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(betas_monthly_df['Date'], betas_monthly_df['Beta'], marker= '*', color= 'p
```

```
plt.title("Monthly Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```



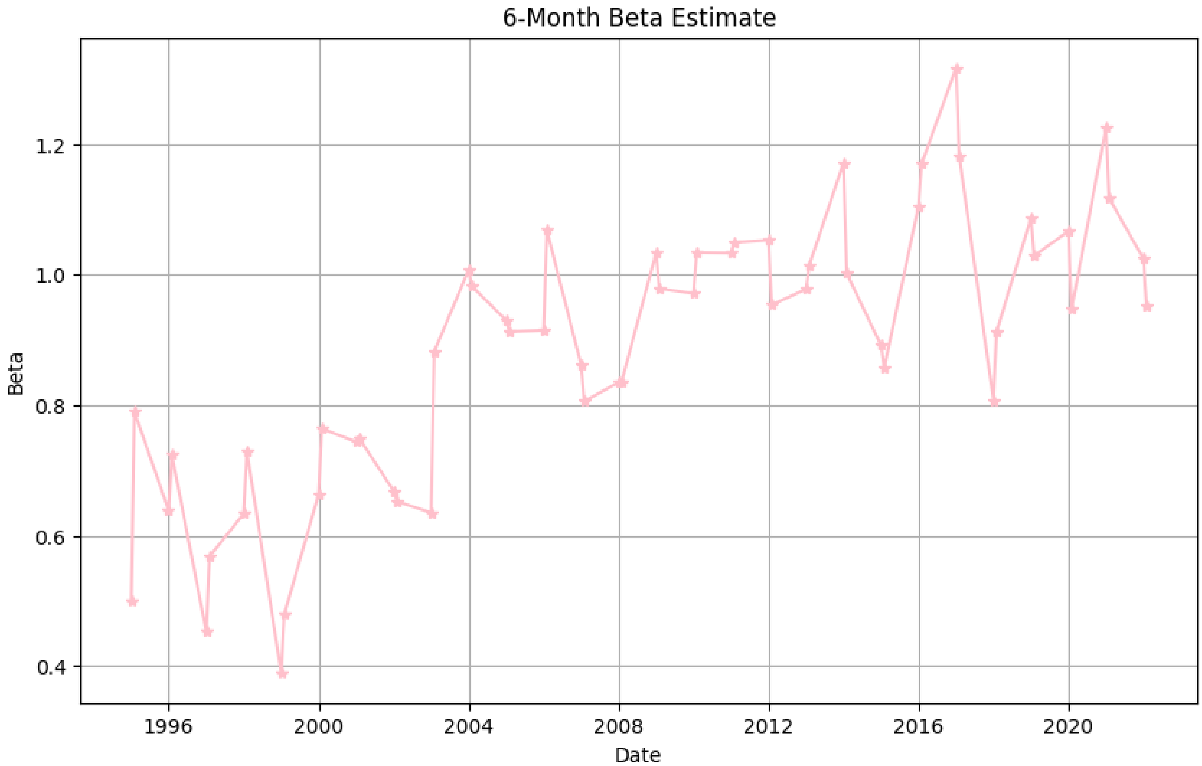
In [ ]:
```
#3 Month Beta Estimate
df['quarter'] = df['date'].dt.quarter
grouped = df.groupby(['year', 'quarter'])

betas = {}
for(year, quarter), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year,quarter)] = beta

#convert to dataframe
betas_quarterly_df = pd.DataFrame(list(betas.items()), columns = ['Year_Quarter', '
#make a date column ffrom the year_month column
betas_quarterly_df["Date"] = pd.to_datetime(betas_quarterly_df['Year_Quarter'].appl
betas_quarterly_df = betas_quarterly_df.sort_values('Date')

#3 Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(betas_quarterly_df['Date'], betas_quarterly_df['Beta'], marker= '*', color
plt.title("3-Month Beta Estimate")
```

```
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

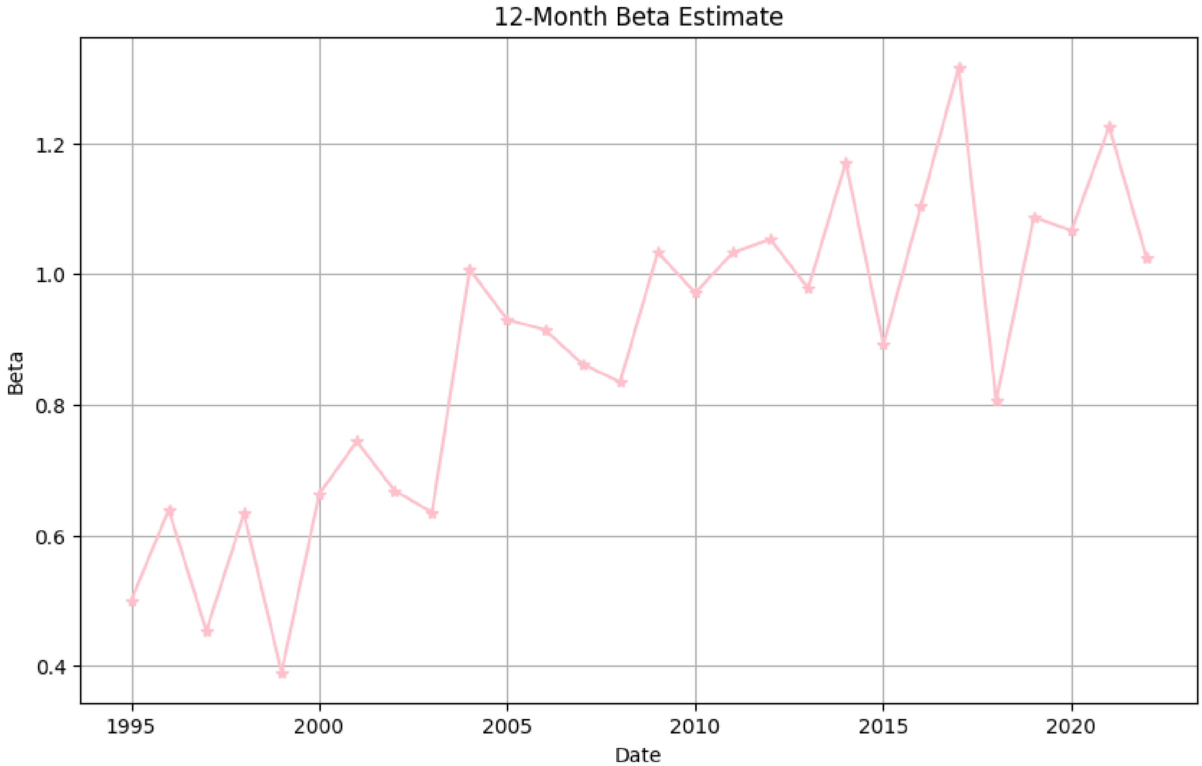


In [ ]:
```
#3 Month Beta Estimate
df['semiannual'] = (df['date'].dt.month - 1) // 6 + 1
grouped = df.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')

#6 Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(betas_semi_df['Date'], betas_semi_df['Beta'], marker= '*', color= 'pink')
plt.title("6-Month Beta Estimate")
plt.xlabel('Date')
```

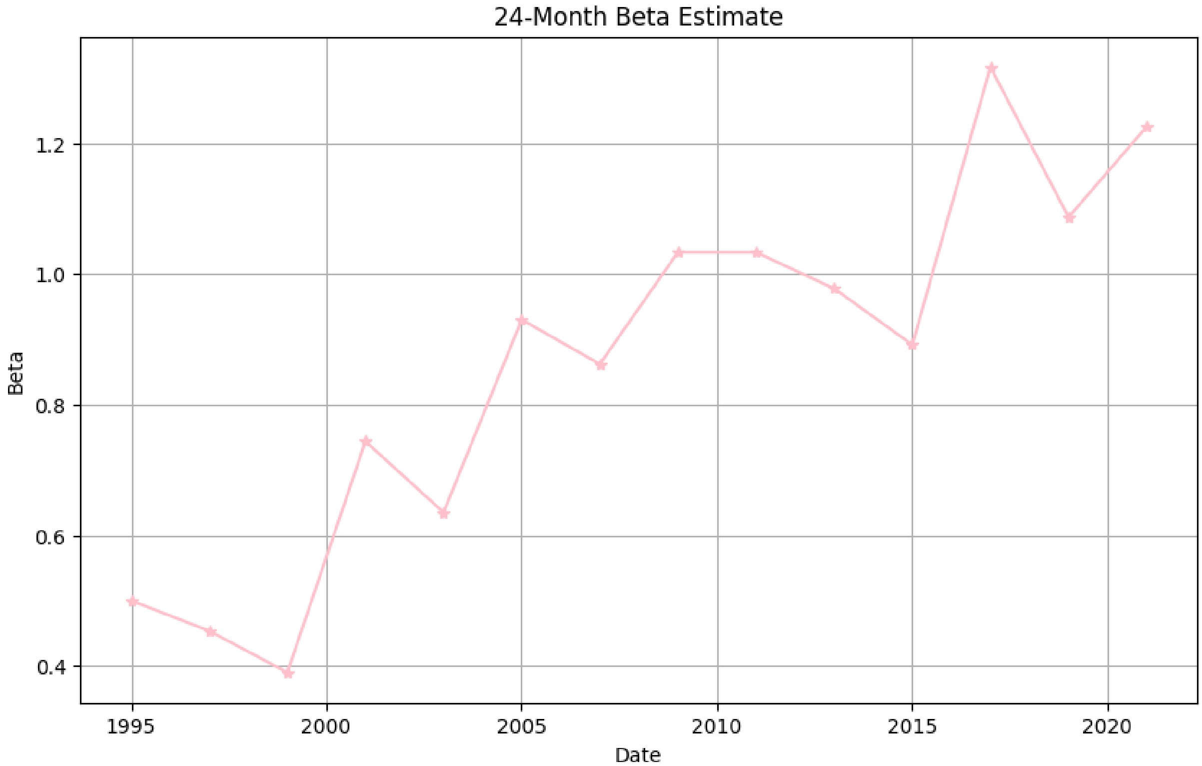
```
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```



In [ ]:
```
#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pin
plt.title("12-Month Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```



In [ ]:
```
#24-Month Beta Estimates
two_yearly = []  # Renamed to better reflect the 2-year nature
beta_24 = []  # Renamed for clarity

# Loop with a step of 4 to get every 4th data point for 24-month estimates
for i in range(0, len(betas_semi_df['Year_SemiAnnual']), 4):
    two_yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta_24.append(betas_semi_df['Beta'][i])

two_yearly_beta_df = pd.DataFrame({
    'two_yearly': two_yearly,
    'beta_24': beta_24
})


#24-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(two_yearly_beta_df['two_yearly'], two_yearly_beta_df['beta_24'], marker= '
plt.title("24-Month Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```



2. Monthly Beta Estimates

estimate the beta using previous 12, 24 and 36 months stock returns (using CRSP_MSF file)

In [ ]:
```
#import that other damn file
path = r"C:\Users\morganhales\Downloads\msf_1926_2022.csv"
df_msf = pd.read_csv(path, parse_dates=['date'])
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\2328063758.py:3: DtypeWarni
ng: Columns (9) have mixed types. Specify dtype option on import or set low_memory=F
alse.
  df_msf = pd.read_csv(path, parse_dates=['date'])
```

In [ ]:
```
#Format the data correctly
df_msf['RET'] = pd.to_numeric(df_msf['RET'], errors='coerce')

#take care of missing values
df_msf['RET'] = df_msf['RET'].fillna(0)
df_msf['vwretd'] = df_msf['vwretd'].fillna(0)

#extract year and month
df_msf['year'] = df_msf['date'].dt.year
df_msf['month'] = df_msf['date'].dt.month
```

In [ ]:
```
#msf betas
grouped = df_msf.groupby(['year', 'month'])

betas = {}
for(year, month), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
```

```
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year,month)] = beta

#convert to dataframe
betas_msf_df = pd.DataFrame(list(betas.items()), columns = ['Year_Month', 'Beta'])
#make a date column ffrom the year_month column
betas_msf_df["Date"] = pd.to_datetime(betas_msf_df['Year_Month'].apply(lambda x: f'
betas_msf_df = betas_msf_df.sort_values('Date')
```

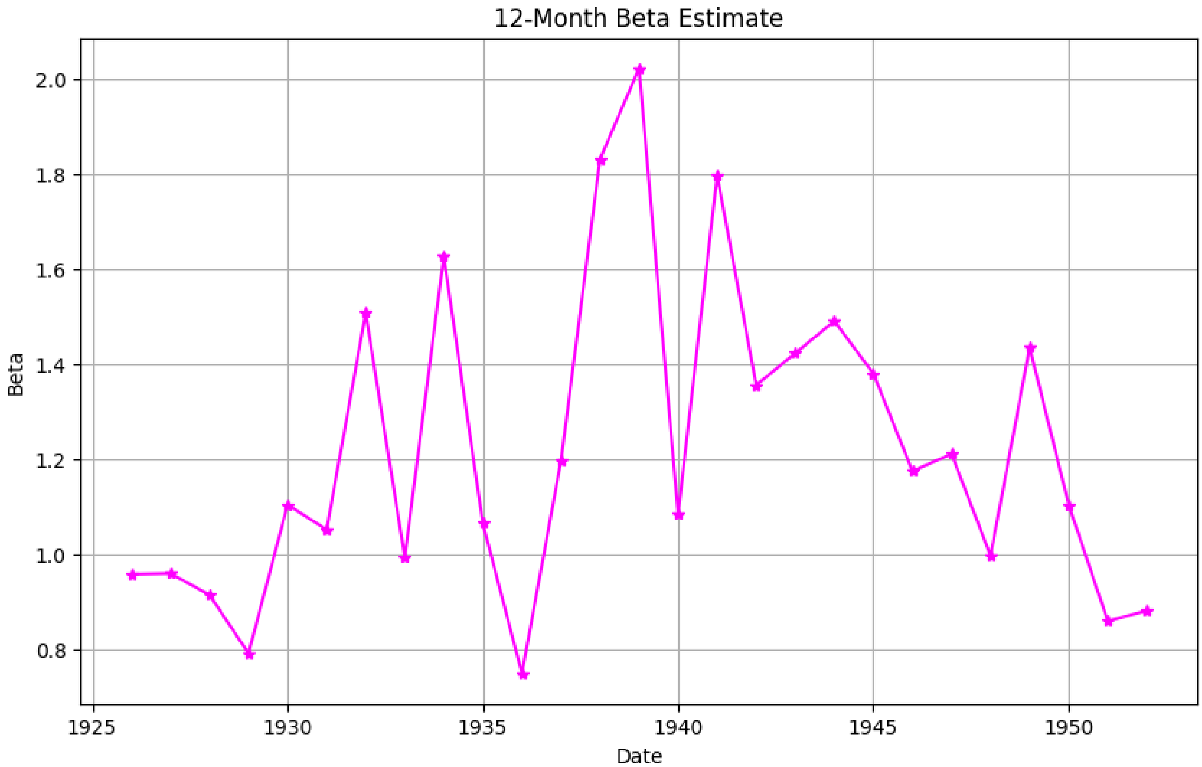
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\1930989249.py:13: RuntimeWa
rning: invalid value encountered in scalar divide
  beta = cov/var_vwretd
```

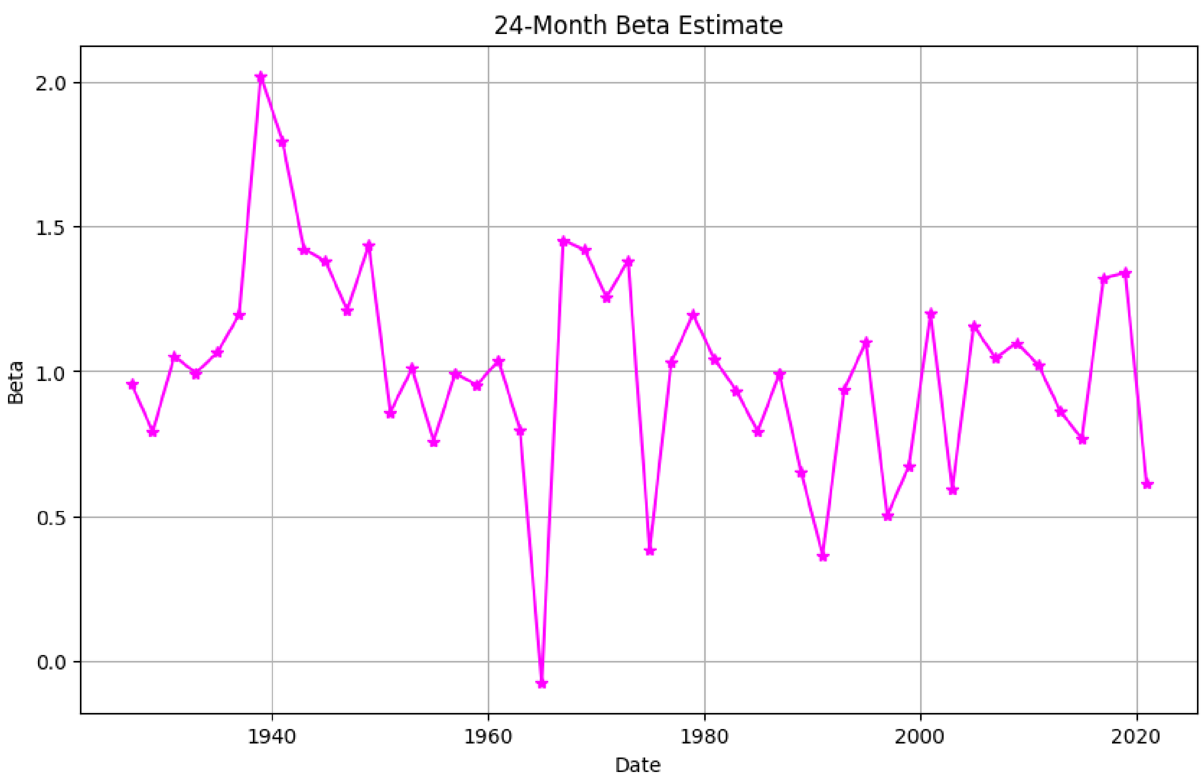
In [ ]:

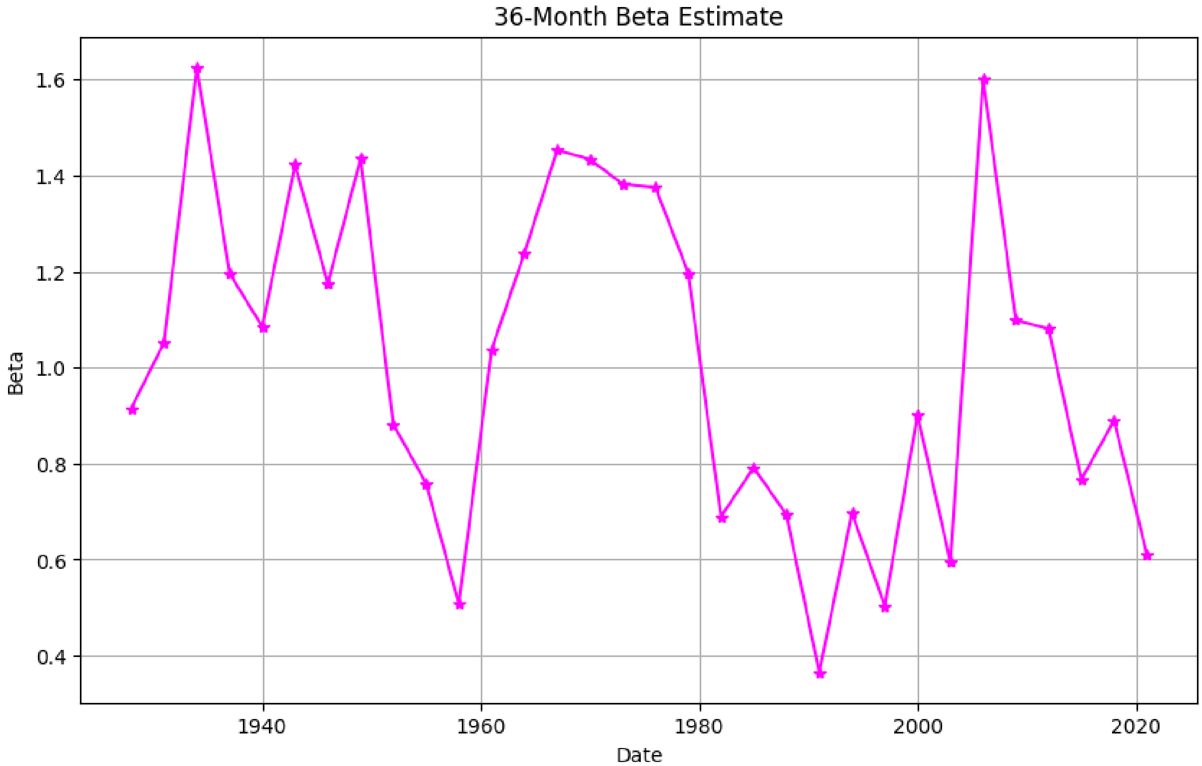
```
#6 Month Beta Estimate
df_msf['semiannual'] = (df_msf['date'].dt.month - 1) // 6 + 1
grouped = df_msf.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_msf_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual',
#make a date column ffrom the year_month column
betas_msf_semi_df["Date"] = pd.to_datetime(betas_msf_semi_df['Year_SemiAnnual'].app
betas_df = betas_msf_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_msf_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_msf_semi_df['Beta'][i])
yearly_msf_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
```

```
plt.plot(yearly_msf_beta_df['yearly'], yearly_msf_beta_df['beta'], marker= '*', col
plt.title("12-Month Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()

#24-Month Beta Estimates
two_yearly = []  # Renamed to better reflect the 2-year nature
beta_24 = []  # Renamed for clarity

# Loop with a step of 4 to get every 4th data point for 24-month estimates
for i in range(0, len(betas_msf_semi_df['Year_SemiAnnual']), 4):
    two_yearly.append(betas_msf_semi_df['Year_SemiAnnual'][i][0])
    beta_24.append(betas_msf_semi_df['Beta'][i])

two_yearly_msf_beta_df = pd.DataFrame({
    'two_yearly': two_yearly,
    'beta_24': beta_24
})


#24-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(two_yearly_msf_beta_df['two_yearly'], two_yearly_msf_beta_df['beta_24'], m
plt.title("24-Month Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()


# 36-Month Beta Estimates
three_yearly = []  # Reflecting the 3-year nature
beta_36 = []  # For clarity

# Loop with a step of 6 to get every 6th data point for 36-month estimates
for i in range(0, len(betas_msf_semi_df['Year_SemiAnnual']), 6):
    three_yearly.append(betas_msf_semi_df['Year_SemiAnnual'][i][0])
    beta_36.append(betas_msf_semi_df['Beta'][i])

three_yearly_msf_beta_df = pd.DataFrame({
    'three_yearly': three_yearly,
    'beta_36': beta_36
})

# 36-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(three_yearly_msf_beta_df['three_yearly'], three_yearly_msf_beta_df['beta_3
plt.title("36-Month Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\3447317548.py:14: RuntimeWa
rning: invalid value encountered in scalar divide
  beta = cov/var_vwretd
```







3. Adjust Beta Computation

winsorize daily stock rates

compute stocks betas as before (using winsorized return series)

In [ ]:
```
# Compute the winsorized thresholds
df['lower_threshold'] = -2 * df['vwretd']
df['upper_threshold'] = 4 * df['vwretd']

# Winsorize the stock returns based on the thresholds
df['winsorized_RET'] = np.where(df['RET'] < df['lower_threshold'], df['lower_thresh
```

In [ ]:
```
grouped = df.groupby(['year', 'month'])

betas_wins_low = {}
for(year, month), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["lower_threshold"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas_wins_low[(year,month)] = beta


#convert to dataframe
betas_wins_low_df = pd.DataFrame(list(betas_wins_low.items()), columns = ['Year_Mon
#make a date column ffrom the year_month column
```

```
betas_wins_low_df["Date"] = pd.to_datetime(betas_wins_low_df['Year_Month'].apply(la
betas_wins_low_df = betas_wins_low_df.sort_values('Date')


betas_wins_up = {}
for(year, month), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["upper_threshold"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas_wins_up[(year,month)] = beta


#convert to dataframe
betas_wins_up_df = pd.DataFrame(list(betas_wins_up.items()), columns = ['Year_Month
#make a date column ffrom the year_month column
betas_wins_up_df["Date"] = pd.to_datetime(betas_wins_up_df['Year_Month'].apply(lamb
betas_wins_up_df = betas_wins_up_df.sort_values('Date')


betas_wins = {}
for(year, month), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["winsorized_RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas_wins[(year,month)] = beta


#convert to dataframe
betas_wins_df = pd.DataFrame(list(betas_wins.items()), columns = ['Year_Month', 'Be
#make a date column ffrom the year_month column
betas_wins_df["Date"] = pd.to_datetime(betas_wins_df['Year_Month'].apply(lambda x:
betas_wins_df = betas_wins_df.sort_values('Date')
```

4. Compute the descriptive Stats

In [ ]:
```
from scipy.stats import skew, kurtosis
desc_stats = {
    'Title': 'Winsorized Betas Lower',
    'N': len(betas_wins_low_df['Beta']),
    'Mean': betas_wins_low_df['Beta'].mean(),
    'Standard Deviation': betas_wins_low_df['Beta'].std(),
    'Skewness': skew(betas_wins_low_df['Beta']),
    'Kurtosis': kurtosis(betas_wins_low_df['Beta']),
    'Minimum Value': betas_wins_low_df['Beta'].min(),
    '1% Percentile': betas_wins_low_df['Beta'].quantile(0.01),
```

```
    '5% Percentile': betas_wins_low_df['Beta'].quantile(0.05),
    '25% Percentile': betas_wins_low_df['Beta'].quantile(0.25),
    '50% Percentile': betas_wins_low_df['Beta'].quantile(0.50),
    '75% Percentile': betas_wins_low_df['Beta'].quantile(0.75),
    '95% Percentile': betas_wins_low_df['Beta'].quantile(0.95),
    '99% Percentile': betas_wins_low_df['Beta'].quantile(0.99),
    'Maximum Value': betas_wins_low_df['Beta'].max(),
}


# Print descriptive statistics
for key, value in desc_stats.items():
    if isinstance(value, str):  # If the value is a string (e.g., "Title"), print i
        print(f"{key}: {value}")
    else:
        print(f"{key}: {value:.4f}")

desc_stats = {
    'Title': 'Winsorized Betas Upper',
    'N': len(betas_wins_up_df['Beta']),
    'Mean': betas_wins_up_df['Beta'].mean(),
    'Standard Deviation': betas_wins_up_df['Beta'].std(),
    'Skewness': skew(betas_wins_up_df['Beta']),
    'Kurtosis': kurtosis(betas_wins_up_df['Beta']),
    'Minimum Value': betas_wins_up_df['Beta'].min(),
    '1% Percentile': betas_wins_up_df['Beta'].quantile(0.01),
    '5% Percentile': betas_wins_up_df['Beta'].quantile(0.05),
    '25% Percentile': betas_wins_up_df['Beta'].quantile(0.25),
    '50% Percentile': betas_wins_up_df['Beta'].quantile(0.50),
    '75% Percentile': betas_wins_up_df['Beta'].quantile(0.75),
    '95% Percentile': betas_wins_up_df['Beta'].quantile(0.95),
    '99% Percentile': betas_wins_up_df['Beta'].quantile(0.99),
    'Maximum Value': betas_wins_up_df['Beta'].max(),
}

# Print descriptive statistics
for key, value in desc_stats.items():
    if isinstance(value, str):  # If the value is a string (e.g., "Title"), print i
        print(f"{key}: {value}")
    else:
        print(f"{key}: {value:.4f}")


desc_stats = {
    'Title': 'Winsorized Betas',
    'N': len(betas_wins_df['Beta']),
    'Mean': betas_wins_df['Beta'].mean(),
    'Standard Deviation': betas_wins_df['Beta'].std(),
    'Skewness': skew(betas_wins_df['Beta']),
    'Kurtosis': kurtosis(betas_wins_df['Beta']),
    'Minimum Value': betas_wins_df['Beta'].min(),
    '1% Percentile': betas_wins_df['Beta'].quantile(0.01),
    '5% Percentile': betas_wins_df['Beta'].quantile(0.05),
    '25% Percentile': betas_wins_df['Beta'].quantile(0.25),
    '50% Percentile': betas_wins_df['Beta'].quantile(0.50),
    '75% Percentile': betas_wins_df['Beta'].quantile(0.75),
```

```python
    '95% Percentile': betas_wins_df['Beta'].quantile(0.95),
    '99% Percentile': betas_wins_df['Beta'].quantile(0.99),
    'Maximum Value': betas_wins_df['Beta'].max(),
}

# Print descriptive statistics
for key, value in desc_stats.items():
    if isinstance(value, str):  # If the value is a string (e.g., "Title"), print i
        print(f"{key}: {value}")
    else:
        print(f"{key}: {value:.4f}")

desc_stats = {
    'Title': 'Betas',
    'N': len(betas_df['Beta']),
    'Mean': betas_df['Beta'].mean(),
    'Standard Deviation': betas_df['Beta'].std(),
    'Skewness': skew(betas_df['Beta']),
    'Kurtosis': kurtosis(betas_df['Beta']),
    'Minimum Value': betas_df['Beta'].min(),
    '1% Percentile': betas_df['Beta'].quantile(0.01),
    '5% Percentile': betas_df['Beta'].quantile(0.05),
    '25% Percentile': betas_df['Beta'].quantile(0.25),
    '50% Percentile': betas_df['Beta'].quantile(0.50),
    '75% Percentile': betas_df['Beta'].quantile(0.75),
    '95% Percentile': betas_df['Beta'].quantile(0.95),
    '99% Percentile': betas_df['Beta'].quantile(0.99),
    'Maximum Value': betas_df['Beta'].max(),
}

# Print descriptive statistics
for key, value in desc_stats.items():
    if isinstance(value, str):  # If the value is a string (e.g., "Title"), print i
        print(f"{key}: {value}")
    else:
        print(f"{key}: {value:.4f}")
```

```
Title: Winsorized Betas Lower
N: 336.0000
Mean: -2.0000
Standard Deviation: 0.0000
Skewness: 0.2628
Kurtosis: -1.1674
Minimum Value: -2.0000
1% Percentile: -2.0000
5% Percentile: -2.0000
25% Percentile: -2.0000
50% Percentile: -2.0000
75% Percentile: -2.0000
95% Percentile: -2.0000
99% Percentile: -2.0000
Maximum Value: -2.0000
Title: Winsorized Betas Upper
N: 336.0000
Mean: 4.0000
Standard Deviation: 0.0000
Skewness: -0.2628
Kurtosis: -1.1674
Minimum Value: 4.0000
1% Percentile: 4.0000
5% Percentile: 4.0000
25% Percentile: 4.0000
50% Percentile: 4.0000
75% Percentile: 4.0000
95% Percentile: 4.0001
99% Percentile: 4.0001
Maximum Value: 4.0001
Title: Winsorized Betas
N: 336.0000
Mean: -0.2632
Standard Deviation: 0.4135
Skewness: 0.1329
Kurtosis: -0.1963
Minimum Value: -1.5125
1% Percentile: -1.1063
5% Percentile: -0.8507
25% Percentile: -0.5682
50% Percentile: -0.2709
75% Percentile: 0.0101
95% Percentile: 0.4699
99% Percentile: 0.6542
Maximum Value: 0.8959
Title: Betas
N: 195.0000
Mean: 1.0633
Standard Deviation: 0.4022
Skewness: nan
Kurtosis: nan
Minimum Value: -0.5594
1% Percentile: 0.1123
5% Percentile: 0.5017
25% Percentile: 0.8553
50% Percentile: 1.0369
```

```
75% Percentile: 1.2553
95% Percentile: 1.7294
99% Percentile: 2.1437
Maximum Value: 2.7013
```

5. Compute the Correlation Across the Various Beta Measures for the Stocks

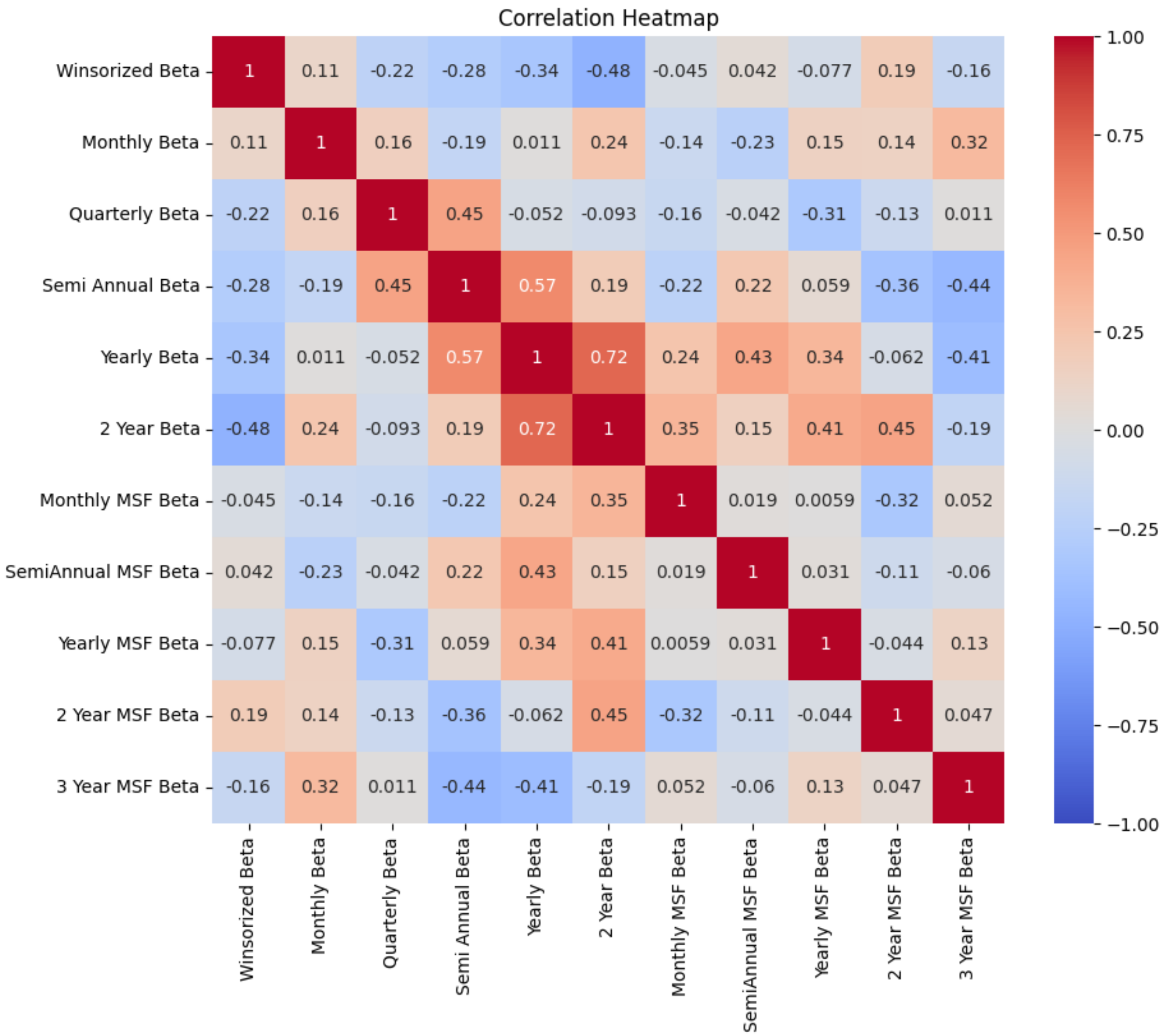
In [ ]:
```
monthly_win = betas_wins_df['Beta']
monthly = betas_monthly_df['Beta']

quarterly = betas_quarterly_df["Beta"]
semi = betas_semi_df['Beta']
yearly = yearly_beta_df['beta']
twoYear = two_yearly_beta_df['beta_24']

msf_monthly = betas_msf_df['Beta']
msf_semi = betas_msf_semi_df['Beta']
msf_yearly = yearly_msf_beta_df['beta']
msf_two_yearly = two_yearly_msf_beta_df['beta_24']
msf_three_yearly = three_yearly_msf_beta_df['beta_36']


betas_df = pd.concat([monthly_win, monthly, quarterly, semi, yearly, twoYear, msf_m
                     keys=['Winsorized Beta', 'Monthly Beta', 'Quarterly Beta', 'Se
                           'SemiAnnual MSF Beta', "Yearly MSF Beta", "2 Year MSF Be
```

In [ ]:
```
print(betas_df)
```

```
      Winsorized Beta  Monthly Beta  Quarterly Beta  Semi Annual Beta  \
0            0.270893      0.613747        0.602375          0.499890
1            0.198654      0.659474        0.424036          0.790122
2            0.294607      0.545460        0.773786          0.638925
3            0.338169      0.589159        0.784205          0.724715
4           -0.061376      0.389063        0.590401          0.453507
...               ...           ...             ...               ...
1160              NaN           NaN             NaN               NaN
1161              NaN           NaN             NaN               NaN
1162              NaN           NaN             NaN               NaN
1163              NaN           NaN             NaN               NaN
1164              NaN           NaN             NaN               NaN

      Yearly Beta  2 Year Beta  Monthly MSF Beta  SemiAnnual MSF Beta  \
0        0.499890     0.499890               NaN                  NaN
1        0.638925     0.453507               NaN             1.180105
2        0.453507     0.389882          0.359551             0.957883
3        0.633287     0.743574               NaN             1.094806
4        0.389882     0.634920         -0.829630             0.959592
...           ...          ...               ...                  ...
1160          NaN          NaN         -0.811630                  NaN
1161          NaN          NaN         -0.204515                  NaN
1162          NaN          NaN         -0.568603                  NaN
1163          NaN          NaN               NaN                  NaN
1164          NaN          NaN         -0.036537                  NaN

      Yearly MSF Beta  2 Year MSF Beta  3 Year MSF Beta
0                 NaN              NaN              NaN
1            0.957883         0.959592         0.914921
2            0.959592         0.791692         1.051788
3            0.914921         1.051788         1.625584
4            0.791692         0.994379         1.196101
...               ...              ...              ...
1160              NaN              NaN              NaN
1161              NaN              NaN              NaN
1162              NaN              NaN              NaN
1163              NaN              NaN              NaN
1164              NaN              NaN              NaN

[1165 rows x 11 columns]
```

In [ ]:

```python
import seaborn as sns
correlation_matrix = betas_df[['Winsorized Beta', 'Monthly Beta', 'Quarterly Beta',
                           'SemiAnnual MSF Beta', "Yearly MSF Beta", "2 Year MSF Be
# Plot heatmap
plt.figure(figsize=(10, 8))  # Set the figure size
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)

# Display the plot
plt.title("Correlation Heatmap")
plt.show()
```



Compute the Mean Beta for Each Industry for Each Year and Plot the Beta's over the 1995-2022 Time Period

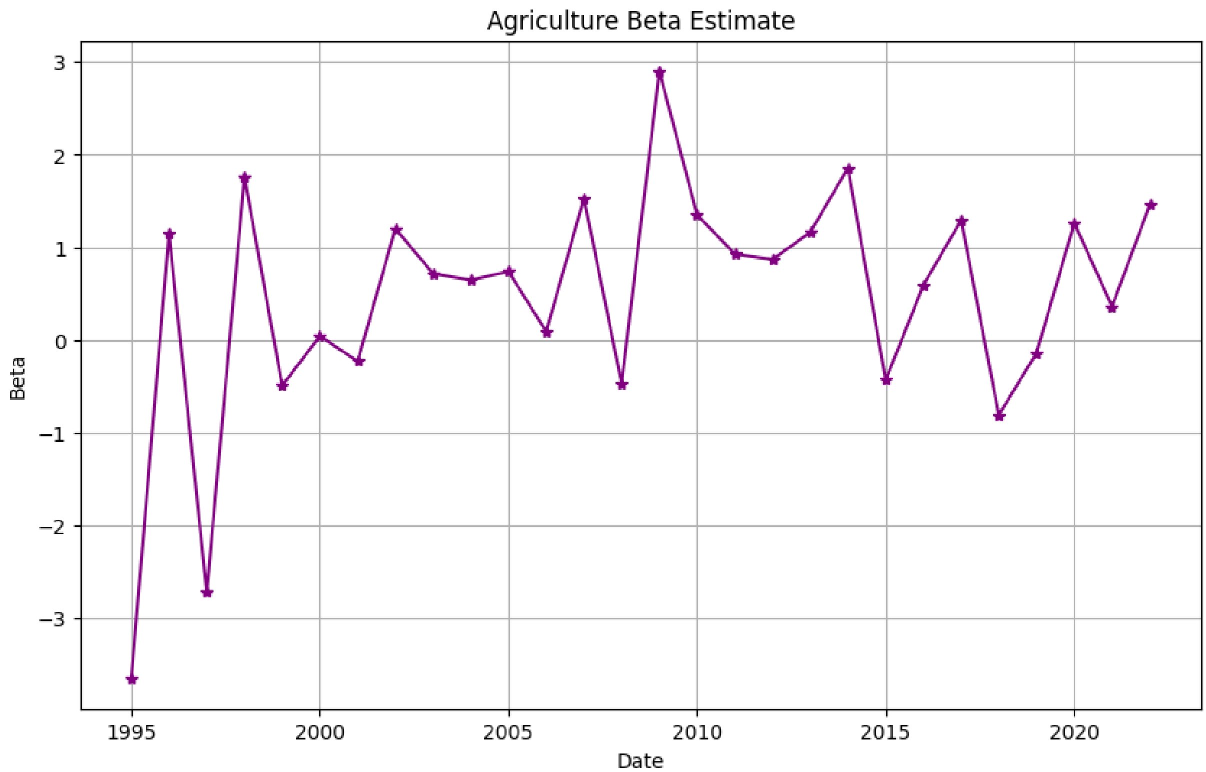
Pick one of the beta's that you have computed in the previous step and explain why you chose this beta in a couple of bullet points.

In [ ]:
```
print(df)
df['HSICCD'] = pd.to_numeric(df['HSICCD'], errors='coerce')
agriculture = df[(df['HSICCD'] >= 1) & (df["HSICCD"] <= 999)]
print(agriculture)
```

```
         PERMNO       date  SHRCD  PERMCO  HEXCD  HSICCD     CUSIP      PRC  \
0         10001 1995-02-01     11    7953      2  4925.0  36720410    7.500   
1         10002 1995-02-01     11    7954      3  6020.0  05978R10  -13.125   
2         10003 1995-02-01     11    7957      3  6020.0  39031810    2.125   
3         10009 1995-02-01     11    7965      3  6030.0  46334710   17.500   
4         10010 1995-02-01     11    7967      3  3840.0  12709510    5.250   
...         ...        ...    ...     ...    ...     ...       ...      ...   
1063459   93423 2022-12-01     11   53440      1  7389.0  83001A10   24.030   
1063460   93426 2022-12-01     11   53443      1  3676.0  92835K10   41.070   
1063461   93429 2022-12-01     11   53447      5  6211.0  12503M10  126.820   
1063462   93434 2022-12-01     11   53427      3  9999.0  78513510    1.150   
1063463   93436 2022-12-01     11   53453      3  9999.0  88160R10  194.700   

                VOL       RET     SHROUT  OPENPRC    vwretd  year  month  
0            3100.0 -0.032258     2224.0    7.500  0.001851  1995      2  
1               0.0  0.000000     2999.0      NaN  0.001851  1995      2  
2             500.0  0.000000     5038.0    2.125  0.001851  1995      2  
3            1902.0  0.037647     1164.0   17.000  0.001851  1995      2  
4            2205.0  0.000000    10359.0    5.375  0.001851  1995      2  
...             ...       ...        ...      ...       ...   ...    ...  
1063459   1382587.0 -0.002491    83157.0   24.110  0.000230  2022     12  
1063460     36940.0  0.012324    12551.0   40.720  0.000230  2022     12  
1063461    474877.0 -0.000158   106082.0  127.290  0.000230  2022     12  
1063462     28689.0  0.036036    42623.0    1.110  0.000230  2022     12  
1063463  80030010.0  0.000000  3157752.0  197.080  0.000230  2022     12  

[1063464 rows x 15 columns]
         PERMNO       date  SHRCD  PERMCO  HEXCD  HSICCD     CUSIP    PRC  \
646       11614 1995-02-01     11    9437      3   170.0  66649910  13.00   
721       11790 1995-02-01     11     153      3   170.0  01623010  16.50   
806       12006 1995-02-01     11    9922      1   115.0  24487820  28.25   
1010      16468 1995-02-01     11     686      3   920.0  05882210   6.75   
1161      20598 1995-02-01     10     737      3   250.0  12803020    NaN   
...         ...        ...    ...     ...    ...     ...       ...    ...   
1060316   18592 2022-12-01     11   56723      1   721.0  22052L10  66.25   
1060833   20598 2022-12-01     11     737      3   250.0  12803020  58.13   
1062282   77300 2022-12-01     11   11324      1   782.0  81018610  55.23   
1062729   85570 2022-12-01     11    7556      3   700.0  12753720   2.11   
1063024   89447 2022-12-01     11   43326      3   700.0  12824610  34.51   

               VOL       RET    SHROUT  OPENPRC    vwretd  year  month  
646          300.0  0.009709    4004.0  13.0000  0.001851  1995      2  
721         1900.0  0.031250    7028.0  16.0625  0.001851  1995      2  
806         4270.0  0.000000    5151.0  28.2500  0.001851  1995      2  
1010        1500.0 -0.035714    2523.0   6.7500  0.001851  1995      2  
1161           NaN  0.000000    2078.0      NaN  0.001851  1995      2  
...            ...       ...       ...      ...       ...   ...    ...  
1060316  1705948.0 -0.013550  714492.0  67.3500  0.000230  2022     12  
1060833   468836.0 -0.002574   44135.0  58.6000  0.000230  2022     12  
1062282   876728.0 -0.012516   55465.0  56.3300  0.000230  2022     12  
1062729    63458.0 -0.004717   55824.0   2.1200  0.000230  2022     12  
1063024    53136.0 -0.012872   17732.0  35.1700  0.000230  2022     12  

[2533 rows x 15 columns]
```

In [ ]:
```
df['HSICCD'] = pd.to_numeric(df['HSICCD'], errors='coerce')
agriculture = df[(df['HSICCD'] >= 1) & (df["HSICCD"] <= 999)]


agriculture['semiannual'] = (agriculture['date'].dt.month - 1) // 6 + 1
grouped = agriculture.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("Agriculture Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

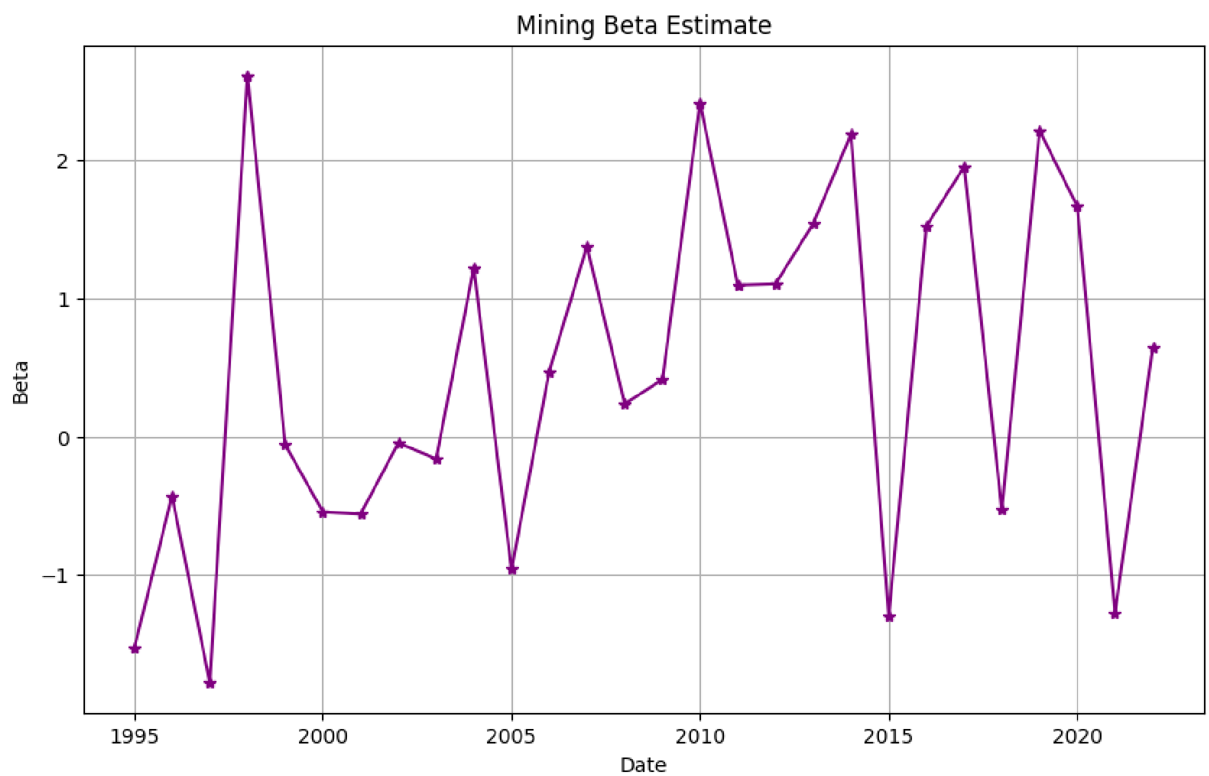
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\2491928311.py:5: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  agriculture['semiannual'] = (agriculture['date'].dt.month - 1) // 6 + 1
```



In [ ]:
```
mining = df[(df['HSICCD'] >= 1000) & (df["HSICCD"] <= 1499)]

mining['semiannual'] = (mining['date'].dt.month - 1) // 6 + 1
grouped = mining.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')
```

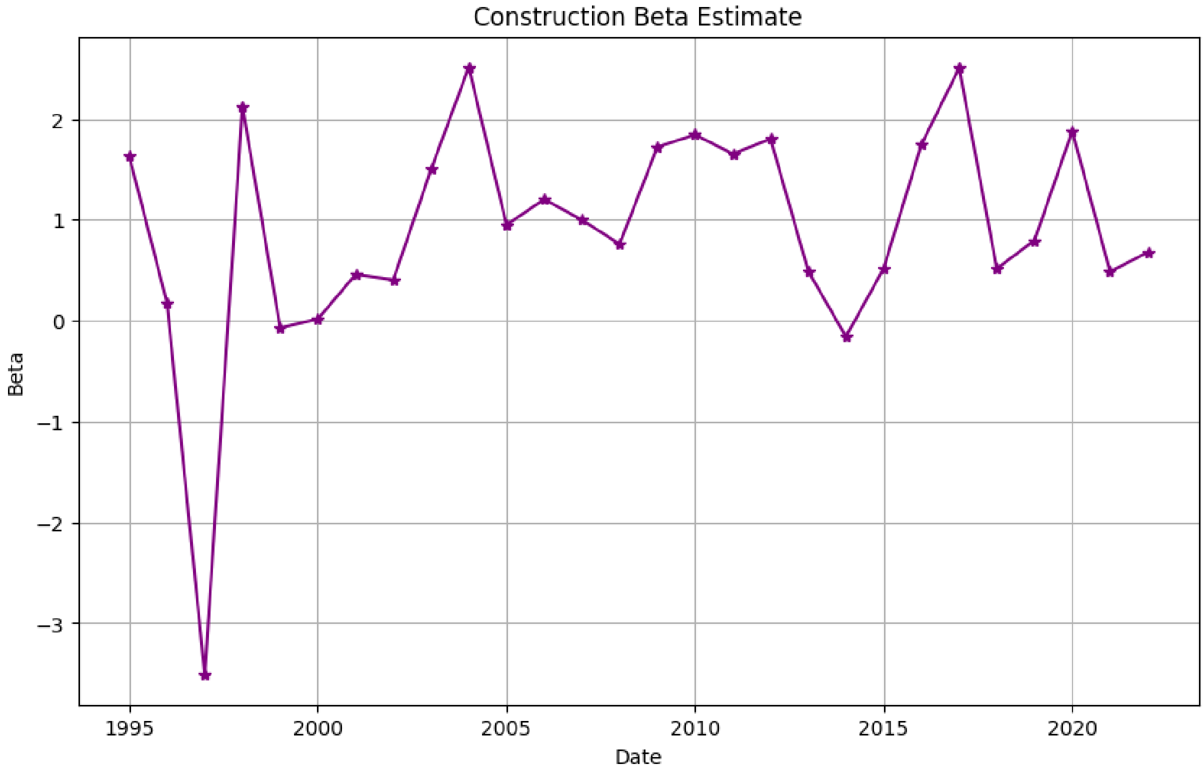
```
#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("Mining Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\2359988712.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  mining['semiannual'] = (mining['date'].dt.month - 1) // 6 + 1
```



In [ ]:
```
construction = df[(df['HSICCD'] >= 1500) & (df["HSICCD"] <= 1799)]

construction['semiannual'] = (construction['date'].dt.month - 1) // 6 + 1
grouped = construction.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("Construction Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

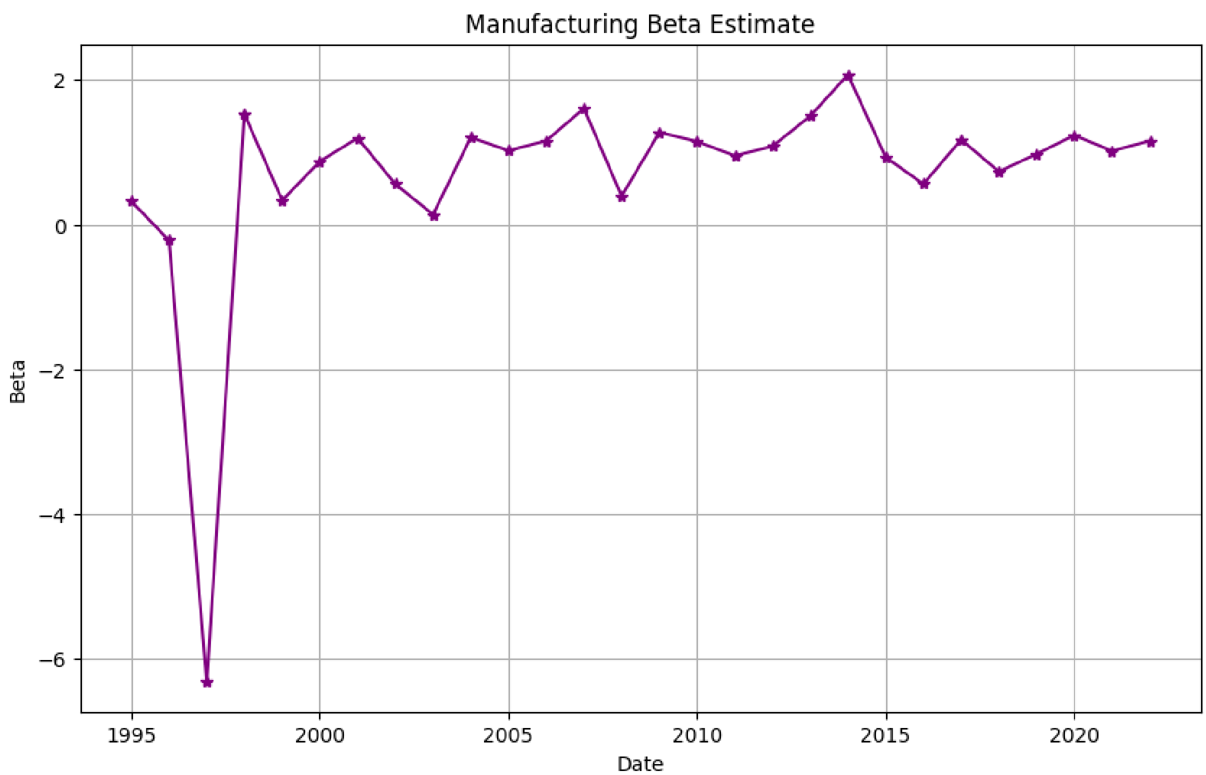
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\2007596591.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  construction['semiannual'] = (construction['date'].dt.month - 1) // 6 + 1
```



In [ ]:
```
manufacturing = df[(df['HSICCD'] >= 2000) & (df["HSICCD"] <= 3999)]

manufacturing['semiannual'] = (manufacturing['date'].dt.month - 1) // 6 + 1
grouped = manufacturing.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
```

```
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("Manufacturing Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\125270076.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  manufacturing['semiannual'] = (manufacturing['date'].dt.month - 1) // 6 + 1
```

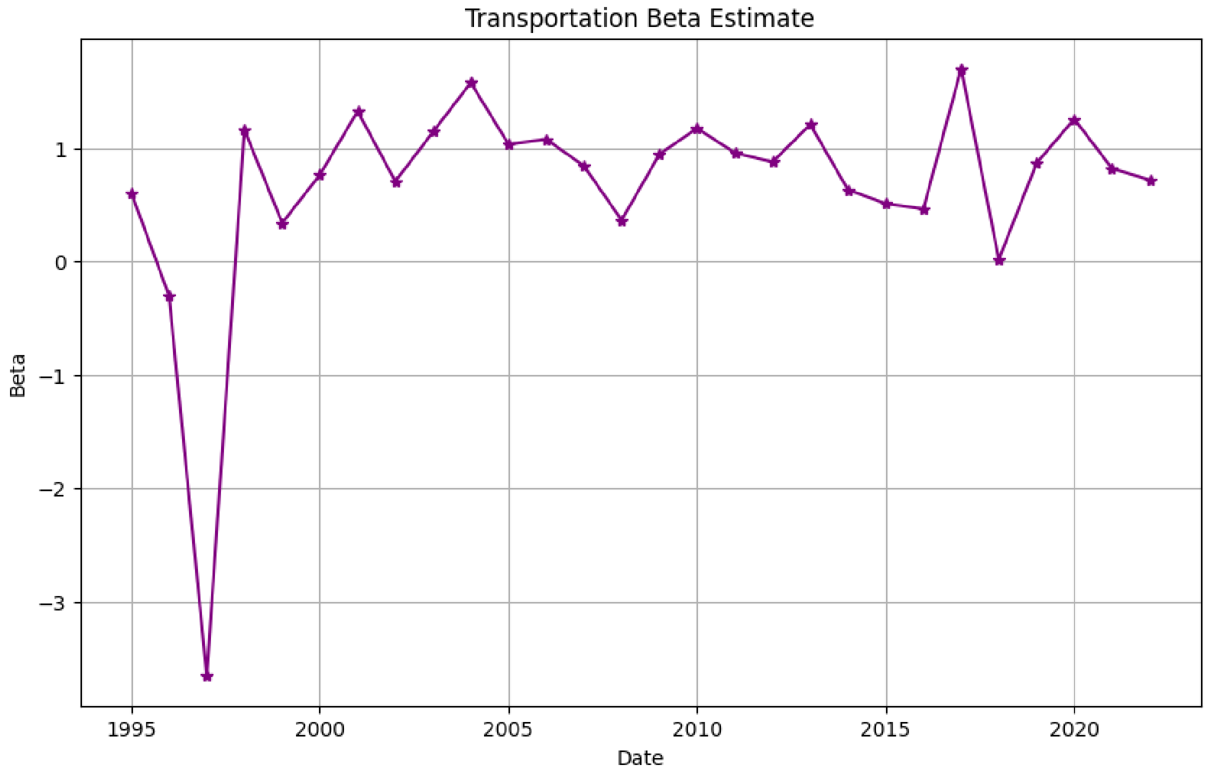


In [ ]:

```
transportation = df[(df['HSICCD'] >= 4000) & (df["HSICCD"] <= 4999)]

transportation['semiannual'] = (transportation['date'].dt.month - 1) // 6 + 1
grouped = transportation.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
```

```
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("Transportation Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

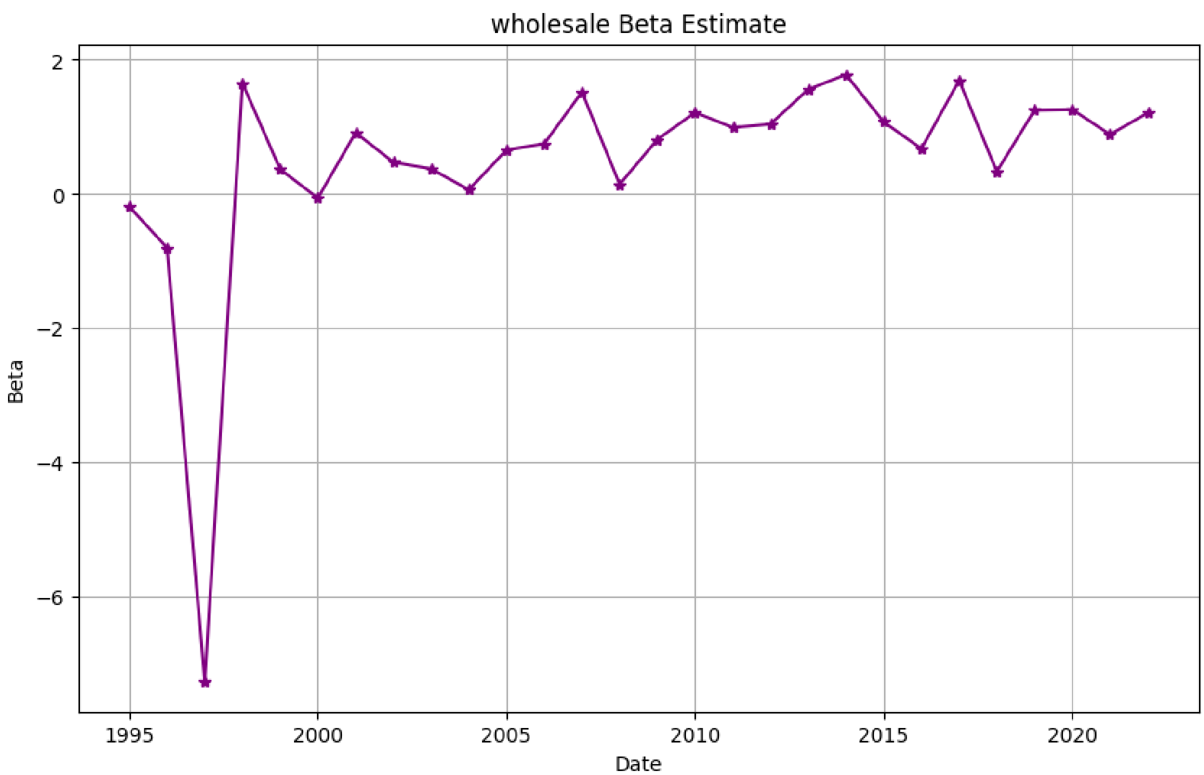
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\1924025394.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  transportation['semiannual'] = (transportation['date'].dt.month - 1) // 6 + 1
```



In [ ]:
```
wholesale = df[(df['HSICCD'] >= 5000) & (df["HSICCD"] <= 5199)]

wholesale['semiannual'] = (wholesale['date'].dt.month - 1) // 6 + 1
grouped = wholesale.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
```

```
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("wholesale Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

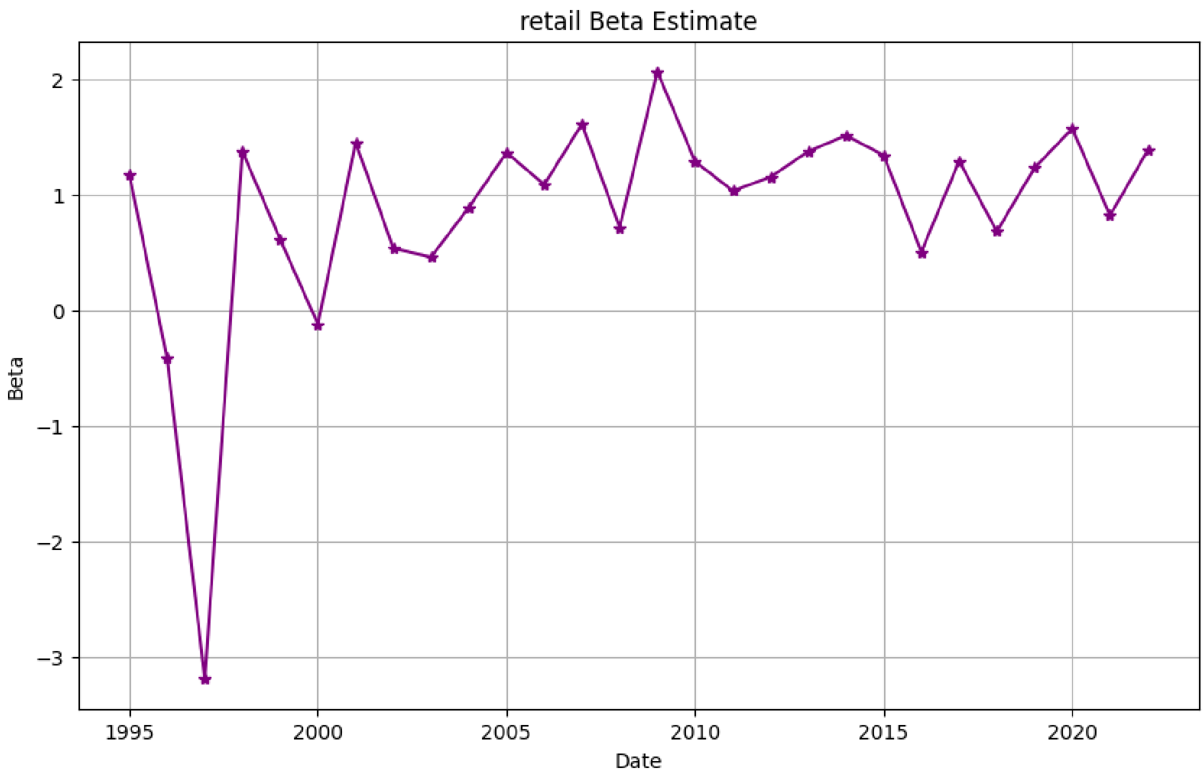
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\3264974785.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  wholesale['semiannual'] = (wholesale['date'].dt.month - 1) // 6 + 1
```



In [ ]:
```
retail = df[(df['HSICCD'] >= 5200) & (df["HSICCD"] <= 5999)]

retail['semiannual'] = (retail['date'].dt.month - 1) // 6 + 1
grouped = retail.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
```

```
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("retail Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

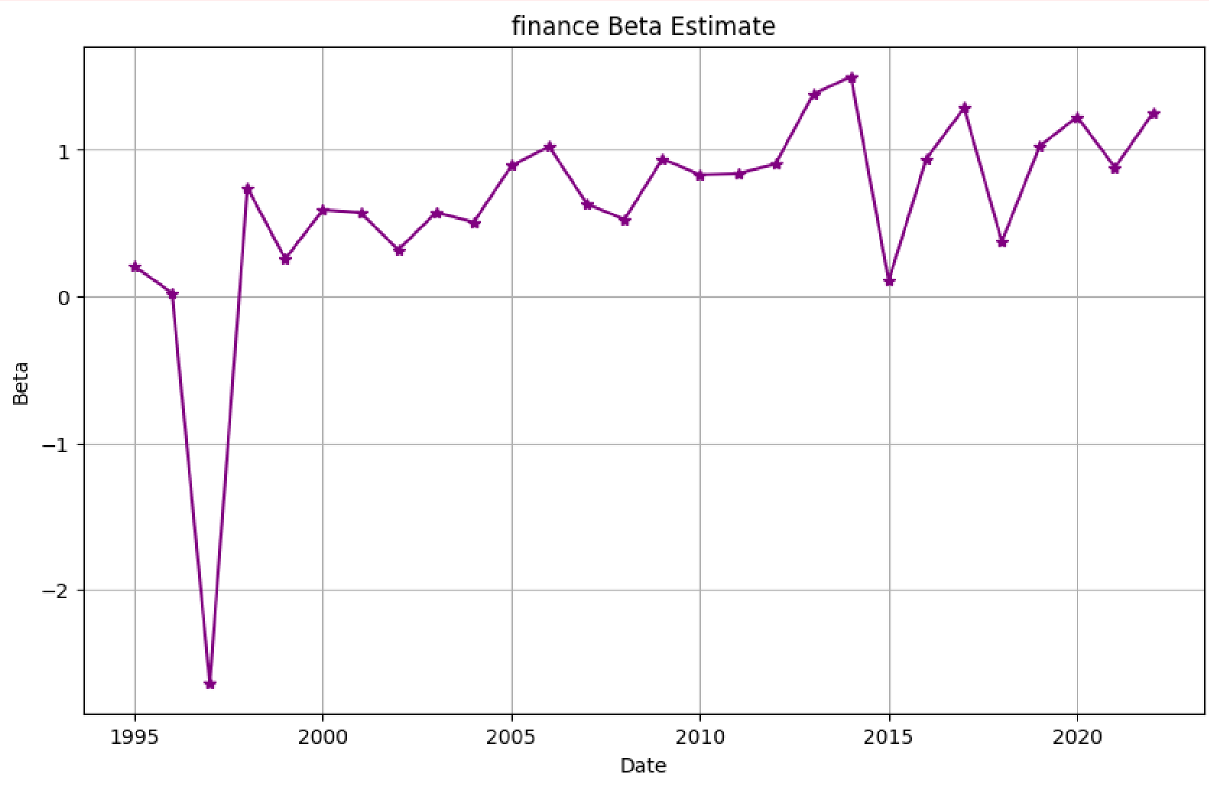
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\2023588093.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  retail['semiannual'] = (retail['date'].dt.month - 1) // 6 + 1
```



In [ ]:
```
finance = df[(df['HSICCD'] >= 6000) & (df["HSICCD"] <= 6799)]

finance['semiannual'] = (finance['date'].dt.month - 1) // 6 + 1
grouped = finance.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
```

```
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("finance Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\2347087767.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  finance['semiannual'] = (finance['date'].dt.month - 1) // 6 + 1
```

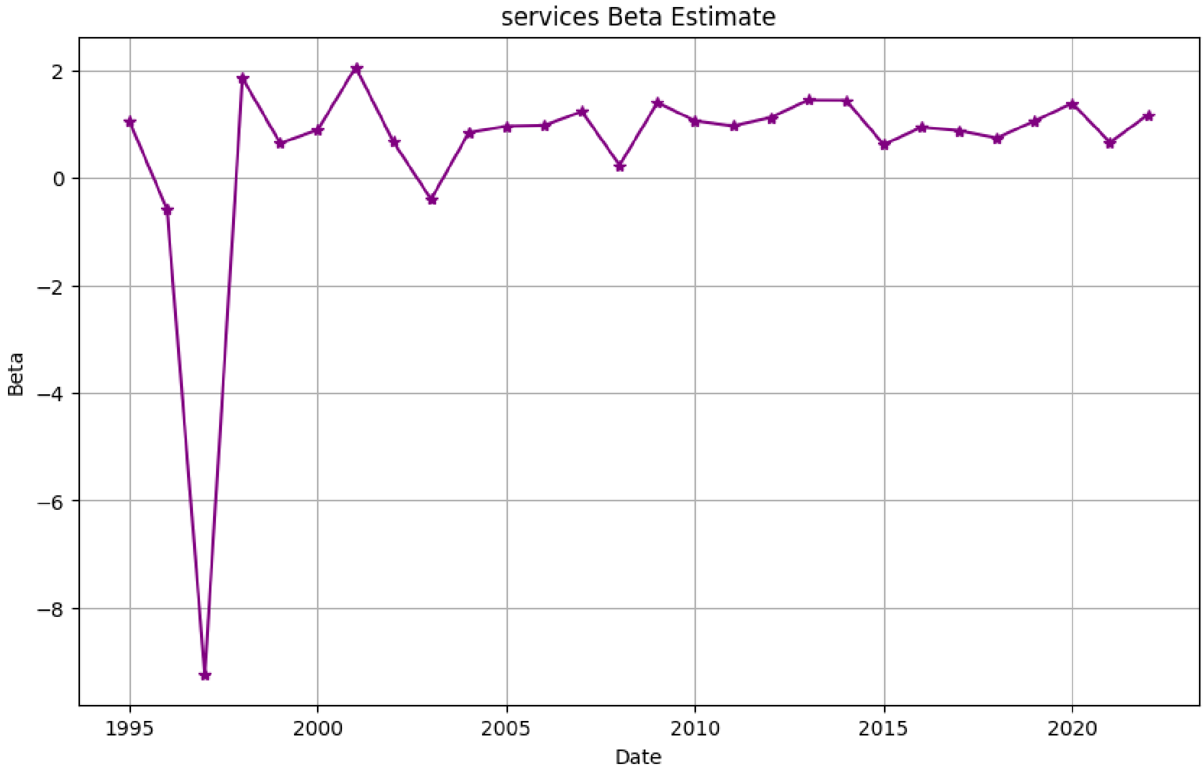


In [ ]:

```
services = df[(df['HSICCD'] >= 7000) & (df["HSICCD"] <= 8999)]

services['semiannual'] = (services['date'].dt.month - 1) // 6 + 1
grouped = services.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
```

```
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0,55,2):
    yearly.append(betas_semi_df['Year_SemiAnnual'][i][0])
    beta.append(betas_semi_df['Beta'][i])
yearly_beta_df = pd.DataFrame({
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("services Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

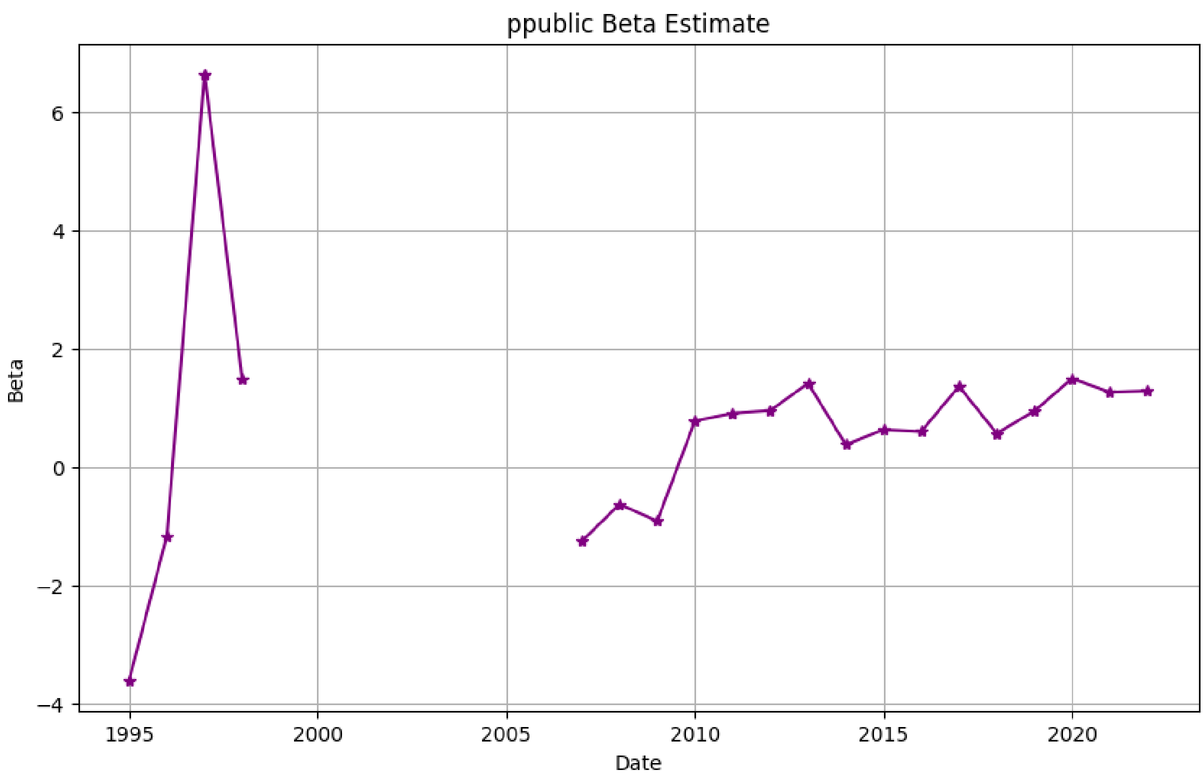
```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\2424501540.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  services['semiannual'] = (services['date'].dt.month - 1) // 6 + 1
```



In [ ]:
```
public = df[(df['HSICCD'] >= 9000) & (df["HSICCD"] <= 9999)]

public['semiannual'] = (public['date'].dt.month - 1) // 6 + 1
grouped = public.groupby(['year', 'semiannual'])

betas = {}
for(year, semiannual), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year, semiannual)] = beta

#convert to dataframe
betas_semi_df = pd.DataFrame(list(betas.items()), columns = ['Year_SemiAnnual', 'Be
#make a date column ffrom the year_month column
betas_semi_df["Date"] = pd.to_datetime(betas_semi_df['Year_SemiAnnual'].apply(lambd
betas_semi_df = betas_semi_df.sort_values('Date')


#12 Month Beta Estimates
yearly = []
beta = []

for i in range(0, len(betas_semi_df), 2):
    yearly.append(betas_semi_df['Year_SemiAnnual'].iloc[i][0])
    beta.append(betas_semi_df['Beta'].iloc[i])
yearly_beta_df = pd.DataFrame({
```

```
    'yearly' : yearly,
    'beta' : beta
})


#12-Month Beta Estimate Plot
plt.figure(figsize=(10,6))
plt.plot(yearly_beta_df['yearly'], yearly_beta_df['beta'], marker= '*', color= 'pur
plt.title("ppublic Beta Estimate")
plt.xlabel('Date')
plt.ylabel("Beta")
plt.grid(True)
plt.show()
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\1451630569.py:3: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  public['semiannual'] = (public['date'].dt.month - 1) // 6 + 1
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\1451630569.py:10: RuntimeWa
rning: Degrees of freedom <= 0 for slice
  cov_matrix = np.cov(group["RET"], group['vwretd'])
c:\Users\morganhales\AppData\Local\Programs\Python\Python311\Lib\site-packages\numpy
\lib\function_base.py:2748: RuntimeWarning: divide by zero encountered in divide
  c *= np.true_divide(1, fact)
c:\Users\morganhales\AppData\Local\Programs\Python\Python311\Lib\site-packages\numpy
\lib\function_base.py:2748: RuntimeWarning: invalid value encountered in multiply
  c *= np.true_divide(1, fact)
```



7. Briefly Describe the Findings from the Beta Computation and from the Graphs

- I used the oringial calculation for beta as it is easier to understand.
- On average, beta doesn't necessarily follow the same path accross industries. Some industries are less susceptible to historical economic events.
- Overall, beta has increased from the 90's to today. This means that on average, assets are becoming more volatile than the market. Of course, this is not true year over year, or for every industry.

CAPM, Beta and Stock Returns

Form Deciles of the betas - chose only one from previous section

In [ ]:

```
#msf betas
grouped = df_msf.groupby(['year', 'month'])

betas = {}
for(year, month), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year,month)] = beta

#convert to dataframe
betas_msf_df = pd.DataFrame(list(betas.items()), columns = ['Year_Month', 'Beta'])
#make a date column ffrom the year_month column
betas_msf_df["Date"] = pd.to_datetime(betas_msf_df['Year_Month'].apply(lambda x: f'
betas_msf_df = betas_msf_df.sort_values('Date')
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\1930989249.py:13: RuntimeWa
rning: invalid value encountered in scalar divide
  beta = cov/var_vwretd
```

In [ ]:

```
df_msf['Year_Month'] = df_msf[['year', 'month']].apply(lambda row: (row['year'], ro
```

```
         PERMNO        date  SHRCD  EXCHCD TICKER                      COMNAM  \
0         10000  1985-12-31    NaN     NaN    NaN                         NaN   
1         10000  1986-01-31   10.0     3.0  OMFGA  OPTIMUM MANUFACTURING INC   
2         10000  1986-02-28   10.0     3.0  OMFGA  OPTIMUM MANUFACTURING INC   
3         10000  1986-03-31   10.0     3.0  OMFGA  OPTIMUM MANUFACTURING INC   
4         10000  1986-04-30   10.0     3.0  OMFGA  OPTIMUM MANUFACTURING INC   
...         ...         ...    ...     ...    ...                         ...   
4927547   93436  2022-08-31   11.0     3.0   TSLA                   TESLA INC   
4927548   93436  2022-09-30   11.0     3.0   TSLA                   TESLA INC   
4927549   93436  2022-10-31   11.0     3.0   TSLA                   TESLA INC   
4927550   93436  2022-11-30   11.0     3.0   TSLA                   TESLA INC   
4927551   93436  2022-12-30   11.0     3.0   TSLA                   TESLA INC   

         PERMCO  ISSUNO  HEXCD HSICCD  ... CFACPR  CFACSHR     ALTPRC  SPREAD  \
0          7952   10396      3   3990  ...    NaN      NaN   -2.56250     NaN   
1          7952   10396      3   3990  ...    1.0      1.0   -4.37500   0.250   
2          7952   10396      3   3990  ...    1.0      1.0   -3.25000   0.250   
3          7952   10396      3   3990  ...    1.0      1.0   -4.43750   0.125   
4          7952   10396      3   3990  ...    1.0      1.0   -4.00000   0.250   
...         ...     ...    ...    ...  ...    ...      ...        ...     ...   
4927547   53453   66252      3   9999  ...    1.0      1.0  275.60999     NaN   
4927548   53453   66252      3   9999  ...    1.0      1.0  265.25000     NaN   
4927549   53453   66252      3   9999  ...    1.0      1.0  227.53999     NaN   
4927550   53453   66252      3   9999  ...    1.0      1.0  194.70000     NaN   
4927551   53453   66252      3   9999  ...    1.0      1.0  123.18000     NaN   

           ALTPRCDT       RETX    vwretd  year  month  semiannual  
0        1986-01-07        NaN  0.043061  1985     12           2  
1        1986-01-31          C  0.009830  1986      1           1  
2        1986-02-28  -0.257143  0.072501  1986      2           1  
3        1986-03-31   0.365385  0.053887  1986      3           1  
4        1986-04-30  -0.098592 -0.007903  1986      4           1  
...             ...        ...       ...   ...    ...         ...  
4927547  2022-08-31  -0.072489 -0.036240  2022      8           2  
4927548  2022-09-30  -0.037589 -0.091324  2022      9           2  
4927549  2022-10-31  -0.142168  0.077403  2022     10           2  
4927550  2022-11-30  -0.144326  0.052365  2022     11           2  
4927551  2022-12-30  -0.367334 -0.057116  2022     12           2  

[4927552 rows x 29 columns]
```

In [ ]:

```python
grouped = df_msf.groupby(['year', 'month'])

betas = {}
for(year, month), group in grouped:
    #Covariance between return on security and return on market portfolio
    #RET is return on security, vwretd is market porfolio return
    cov_matrix = np.cov(group["RET"], group['vwretd'])
    cov = cov_matrix[0,1]

    #calulate variance of vwretd
    var_vwretd = np.var(group['vwretd'])
    beta = cov/var_vwretd
    betas[(year,month)] = beta

#convert to dataframe
```

```
betas_msf_df = pd.DataFrame(list(betas.items()), columns = ['Year_Month', 'Beta'])
#make a date column ffrom the year_month column
betas_msf_df["Date"] = pd.to_datetime(betas_msf_df['Year_Month'].apply(lambda x: f'
betas_msf_df = betas_msf_df.sort_values('Date')
```

```
C:\Users\morganhales\AppData\Local\Temp\2\ipykernel_4240\1326234759.py:12: RuntimeWa
rning: invalid value encountered in scalar divide
  beta = cov/var_vwretd
```

In [ ]:
```
merged_df = df_msf.merge(betas_msf_df[['Year_Month', 'Beta']], on='Year_Month', how
merged_df['Beta_Decile'] = pd.qcut(merged_df['Beta'], 10, labels=False) + 1
```

Form equal weighted portfolios and compute the average beta and the equal weighted (1/N) portfolio excess return for each decile and the difference between portfolio 10 (high beta) and protfolio 1 (low beta)

In [ ]:
```
import statsmodels.api as sm
from statsmodels.stats.sandwich_covariance import cov_hac

# Convert Beta and RET columns to float type
merged_df['Beta'] = pd.to_numeric(merged_df['Beta'], errors='coerce')
merged_df['RET'] = pd.to_numeric(merged_df['RET'], errors='coerce')

# Handle NaNs - You can either drop them or fill them. Here's how to do both:
# df1.dropna(subset=['Beta', 'RET'], inplace=True)  # To drop NaNs
merged_df['Beta'].fillna(0, inplace=True)  # To fill NaNs with 0 for Beta column
merged_df['RET'].fillna(0, inplace=True)   # To fill NaNs with 0 for RET column


# Group by Beta_Decile
grouped = merged_df.groupby('Beta_Decile')

# Calculate average Beta and equal-weighted return for each decile
decile_stats = grouped.agg(Avg_Beta=('Beta', 'mean'), Equal_Weighted_Return=('RET',


# Compute the difference between high beta and low beta portfolios
diff_return = decile_stats.loc[10, 'Equal_Weighted_Return'] - decile_stats.loc[1, '

# Assuming market excess return is 'vwretd'
market_excess = merged_df['vwretd']
nw_t_stats = []

for decile, data in grouped:
    y = data['RET']  # Excess return of the portfolio
    X = sm.add_constant(market_excess.loc[data.index])  # Market excess return
    model = sm.OLS(y, X).fit(cov_type='HAC', cov_kwds={'maxlags': 5})

    # Get Newey-West adjusted t-statistic for the alpha (constant)
    alpha_t_stat = model.tvalues['const']
    nw_t_stats.append(alpha_t_stat)

decile_stats['Newey_West_t_stat'] = nw_t_stats
```

In [ ]:
```
print(decile_stats)
```

```
             Avg_Beta  Equal_Weighted_Return  Newey_West_t_stat
Beta_Decile
1.0         -5.543145               0.004262          -0.687600
2.0         -0.928770               0.004577          -2.230944
3.0         -0.550281               0.009645          -6.166544
4.0         -0.299792              -0.002577         -13.224949
5.0         -0.114841               0.007324          -9.242075
6.0          0.092232               0.023823           2.127568
7.0          0.306277               0.001687           6.653131
8.0          0.577654               0.004657         -21.529849
9.0          1.100289               0.023199          15.467931
10.0         8.251615               0.003224          -0.469079
```

Repeat the previous steps for the value-weighted portfolio (weighted by the market capitalization) returns

In [ ]:
```
# Calculate Market Cap for each observation
merged_df['MarketCap'] = merged_df['PRC'] * merged_df['SHROUT']

# Calculate Total Market Cap for each Beta_Decile
total_market_cap = merged_df.groupby('Beta_Decile')['MarketCap'].transform('sum')

# Calculate Value-Weighted Return for each observation
merged_df['Value_Weighted_RET'] = (merged_df['MarketCap'] / total_market_cap) * mer
```

In [ ]:
```
# Group by Beta Deciles
grouped = merged_df.groupby('Beta_Decile')

# Calculate average Beta and Value-Weighted Excess Return for each decile
decile_stats_vw = grouped.agg(Avg_Beta=('Beta', 'mean'), Value_Weighted_Return=('vw

import statsmodels.api as sm
from statsmodels.stats.sandwich_covariance import cov_hac

market_excess = merged_df['vwretd']
nw_t_stats_vw = []

for decile, data in grouped:
    y = data['vwretd']
    X = sm.add_constant(market_excess.loc[data.index])
    model = sm.OLS(y, X).fit(cov_type='HAC', cov_kwds={'maxlags': 5})

    # Get Newey-West adjusted t-statistic for the alpha (constant)
    alpha_t_stat = model.tvalues['const']
    nw_t_stats_vw.append(alpha_t_stat)

decile_stats_vw['Newey_West_t_stat'] = nw_t_stats_vw
```

In [ ]:
```
print(decile_stats_vw)
```

Beta_Decile	Avg_Beta	Value_Weighted_Return	Newey_West_t_stat
1.0	-5.543145	0.003626	89.206825
2.0	-0.928770	0.004754	-41.141344
3.0	-0.550281	0.011812	110.304983
4.0	-0.299792	0.000982	-10.602951
5.0	-0.114841	0.010021	84.134234
6.0	0.092232	0.023932	126.198929
7.0	0.306277	-0.000131	-1.757040
8.0	0.577654	0.009538	-69.378341
9.0	1.100289	0.016231	122.631662
10.0	8.251615	0.001864	-50.370785

- It is not always true that an increase in beta results in an increase in the return.
- There are other factors that result in the expected return, explained further in the pdf.