

## ✓ Introduction

This is an 'end-of-phase' project that I completed during my time at Flatiron School. In this phase, we dove deep into advance machine learning and modeling to project and predict outcomes. In combination with what I have learned in previous phases, I created this project

## Notebook 1

This notebook is one of two used for this particular project. In this notebook the goal is to perform a descriptive and inferential analysis. I am required to answer 5-10 descriptive analysis question and 2-4 inferential analysis questions

## ✓ Table of Contents

- [Introduction](#)
  - [Notebook 1](#)
  - [Table of Contents](#)
- [Imports and Data Collection](#)
  - [Creating My Dataset](#)
- [Descriptive Analysis](#)
  - [What sectors account for the most amount of companies on the S&P 500?](#)
  - [What is the total Market Capitalization for each sector?](#)
  - [What is the relation between Market Capitalization and the number of companies for each sector?](#)
  - [What companies in the Technology sector account the most Market Capitalization?](#)
  - [What companies in the Financial Services sector account the most Market Capitalization?](#)
  - [What companies in the Healthcare sector account the most Market Capitalization?](#)
  - [What companies in the Industrials sector account the most Market Capitalization?](#)
  - [What companies in the Consumer Cyclical sector account the most Market Capitalization?](#)
  - [What companies in the Communication Services sector account the most Market Capitalization?](#)

- [What companies in the Consumer Defensive sector account the most Market Capitalization?](#)
- [What companies in the Energy sector account the most Market Capitalization?](#)
- [What companies in the Real Estate sector account the most Market Capitalization?](#)
- [What companies in the Utilities sector account the most Market Capitalization?](#)
- [What companies in the Basic Materials sector account the most Market Capitalization?](#)
- [Inferential Analysis](#)
  - [Hypothesis 1: Sector ANOVA](#)
  - [Hypothesis 2.1: High Trailing P/E Ratio vs Low Trailing P/E Ratio](#)
  - [Hypothesis 2.2: High Forward P/E Ratio vs Low Forward P/E Ratio](#)
  - [Hypothesis 3.1: High PEG Ratio vs Low PEG Ratio](#)
  - [Hypothesis 3.2: High Trailing PEG Ratio vs Low Trailing PEG Ratio](#)
  - [Hypothesis 4: High Price-To-Book Ratio vs Low Price-To-Book Ratio](#)
  - [Hypothesis 5: High Debt-To-Equity Ratio vs Low Debt-To-Equity Ratio](#)
  - [Hypothesis 6.1: High Trailing EPS Ratio vs Low Trailing EPS Ratio](#)
  - [Hypothesis 6.2: High Forward EPS Ratio vs Low Forward EPS Ratio](#)
  - [Hypothesis 7: High ROE vs Low ROE](#)
  - [Hypothesis 8.1: High Gross Margin vs Low Gross Margin](#)
  - [Hypothesis 8.2: High Operating Margin vs Low Operating Margin](#)
  - [Hypothesis 8.3: High Profit Margin vs Low Profit Margin](#)
  - [Hypothesis 9.1: High Quick Ratio vs Low Quick Ratio](#)
  - [Hypothesis 9.2: High Current Ratio vs Low Current Ratio](#)
  - [Hypothesis 10.1: High Enterprise-To-EBITDA Ratio vs Low Enterprise-To-EBITDA Ratio](#)
  - [Hypothesis 10.2: High Enterprise-To-Revenue Ratio vs Low Enterprise-To-Revenue Ratio](#)
  - [Hypothesis 11.1: High Free Cash Flows vs Low Free Cash Flows](#)
  - [Hypothesis 11.2: High Total Cash Per Share vs Low Total Cash Per Share](#)
- [Summary](#)

```
import nbformat
```

```
def generate_toc(notebook_path):
    with open(notebook_path) as f:
        nb = nbformat.read(f, as_version=4)

    toc = []
    for cell in nb.cells:
        if cell.cell_type == 'markdown':
            lines = cell.source.split('\n')
            for line in lines:
                if line.startswith('#'):
                    header_level = line.count('#')
```

```

        header_text = line.replace('#', '').strip()
        toc.append((header_level, header_text))

    toc_md = ['## Table of Contents']
    for level, text in toc:
        toc_md.append(f"{' ' * (level - 1)}- [{text}](#{text.replace(' ', '-')})")

    return '\n'.join(toc_md)

notebook_path = 'stock analysis.ipynb'
toc_md = generate_toc(notebook_path)

# Print the generated TOC
# print(toc_md)

```

## ✓ Imports and Data Collection

```

# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings('ignore')

```

## ✓ Creating My Dataset

This project takes a look at stocks that are included in the S&P 500, to get those stocks in my dataset, I found a CSV file with information on every S&P500 companies. The small issue with this dataset is that it doesn't update automatically, unlike the Yahoo Finance API `yfinance` using a combination of the two, I created my dataset which takes info from `yfinance`

First thing to do is create a list of the tickers that I want to use.

```

# uploading the df
df_stocks = pd.read_csv('data/sp500_stocks.csv')

# getting a list of the unique tickers in my dataset
sp500_tickers = df_stocks['Symbol'].unique()
tickers = []

```

```

for x in sp500_tickers:
    tickers.append(x)

# importing yahoo's API
import yfinance as yf

# Various metrics from yfinance that i want to use in my analysis
keys = [
    'symbol', 'shortName', 'country', 'sector', 'previousClose', 'overallRisk', 'beta', 'tra
    'enterpriseValue', 'profitMargins', 'sharesOutstanding', 'bookValue',
    'priceToBook', 'trailingEps', 'forwardEps', 'pegRatio', 'enterpriseToRevenue',
    'enterpriseToEbitda', 'totalCash', 'totalCashPerShare', 'ebitda', 'totalDebt',
    'quickRatio', 'currentRatio', 'totalRevenue', 'debtToEquity', 'revenuePerShare',
    'returnOnAssets', 'returnOnEquity', 'freeCashflow', 'operatingCashflow',
    'earningsGrowth', 'revenueGrowth', 'grossMargins', 'ebitdaMargins',
    'operatingMargins', 'trailingPegRatio', 'priceToSalesTrailing12Months'
]

# Empty list to store results
data = []

# Loop through tickers and get the specific info
for ticker in tickers:
    stock = yf.Ticker(ticker)
    info = stock.info
    # Extract the values of the keys you're interested in
    row = {key: info.get(key, None) for key in keys}
    data.append(row)

# Convert the list of dictionaries into a pandas DataFrame
df = pd.DataFrame(data)

df['priceToSales'] = df['priceToSalesTrailing12Months']

sns.set_style('whitegrid')

```

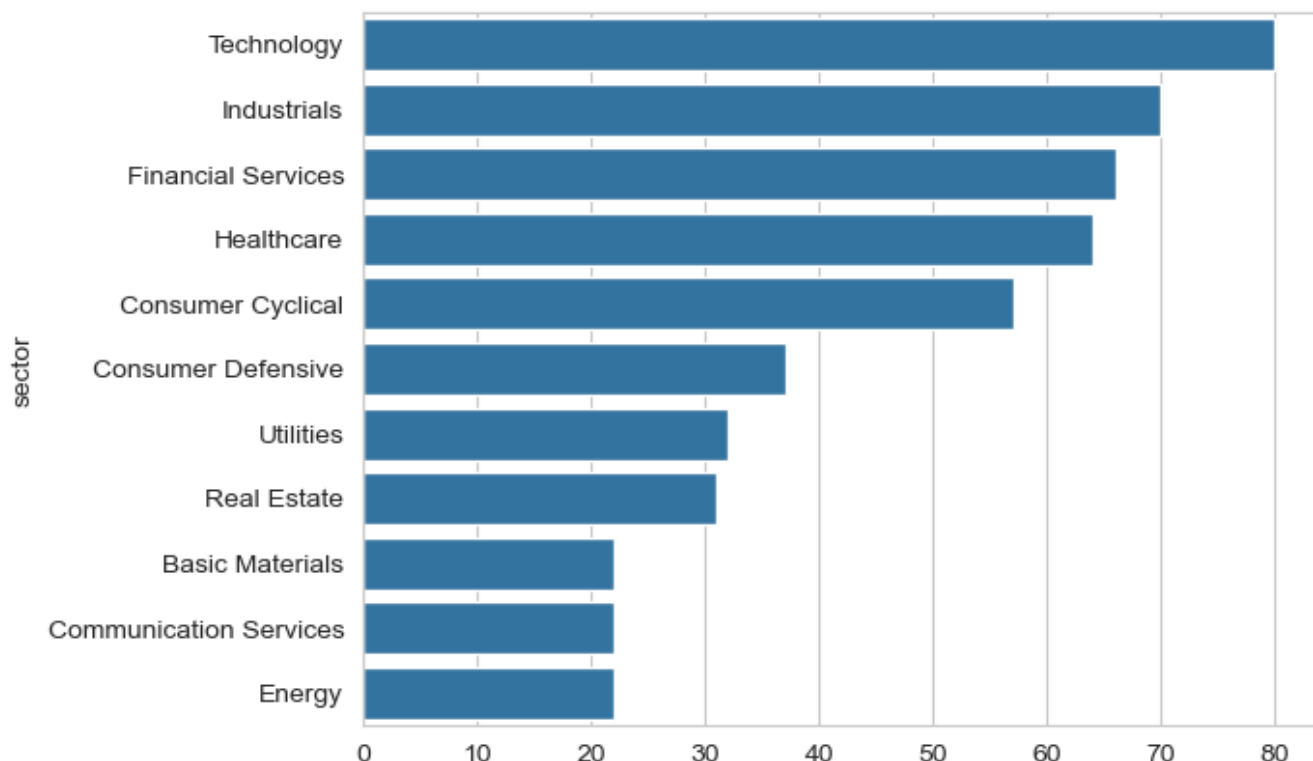
## ✓ Descriptive Analysis

- ✓ What sectors account for the most amount of companies on the S&P 500?

```
Q1 = df['sector'].value_counts()
```

```
sns.barplot(x=Q1.values, y=Q1.index)
```

```
<Axes: ylabel='sector'>
```



## ✓ What is the total Market Capitalization for each sector?

```
# Group by 'sector' and sum the 'marketCap'
```

```
Q2 = df.groupby('sector')['marketCap'].sum()
```

```
# Initialize the scaler
```

```
scaler = MinMaxScaler(feature_range=(1, 10))
```

```
# Reshape Q2 to a 2D array or convert it to a DataFrame for scaling
```

```
Q2_scaled = scaler.fit_transform(Q2.values.reshape(-1, 1))
```

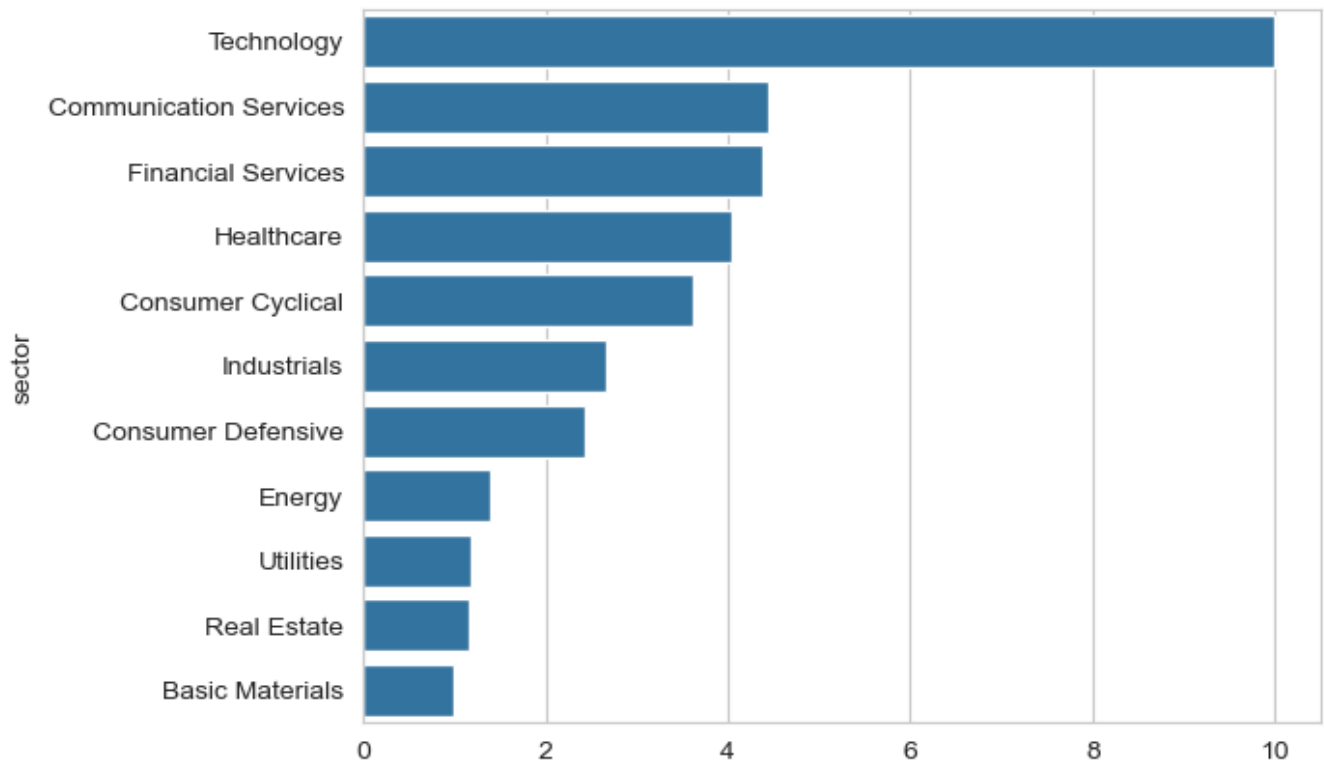
```
# Optionally, convert back to a pandas Series or DataFrame
```

```
Q2 = pd.Series(Q2_scaled.flatten(), index=Q2.index)
```

```
Q2 = Q2.sort_values(ascending=False)
```

```
sns.barplot(x=Q2.values, y=Q2.index)
```

↗ <Axes: ylabel='sector'>



✓ What is the relation between Market Capitalization and the number of companies for each sector?

```
# Value counts of sectors (Q1)
Q1 = df['sector'].value_counts()

# Group by 'sector' and sum the 'marketCap' (Q2)
Q2 = df.groupby('sector')['marketCap'].sum()

# Combine Q1 and Q2 into a single DataFrame
Q3 = pd.DataFrame({'sector_count': Q1, 'market_cap_sum': Q2})

# Initialize MinMaxScaler to scale between -1 and 1
scaler = MinMaxScaler(feature_range=(-1, 1))

# Apply scaling to both 'sector_count' and 'market_cap_sum'
Q3[['sector_count', 'market_cap_sum']] = scaler.fit_transform(Q3[['sector_count', 'market_cap_sum']])

# Scatter plot the scaled data
plt.figure(figsize=(10, 8))
ax = sns.scatterplot(x=Q3['sector_count'], y=Q3['market_cap_sum'], s=100, color='blue', edgecolor='black')

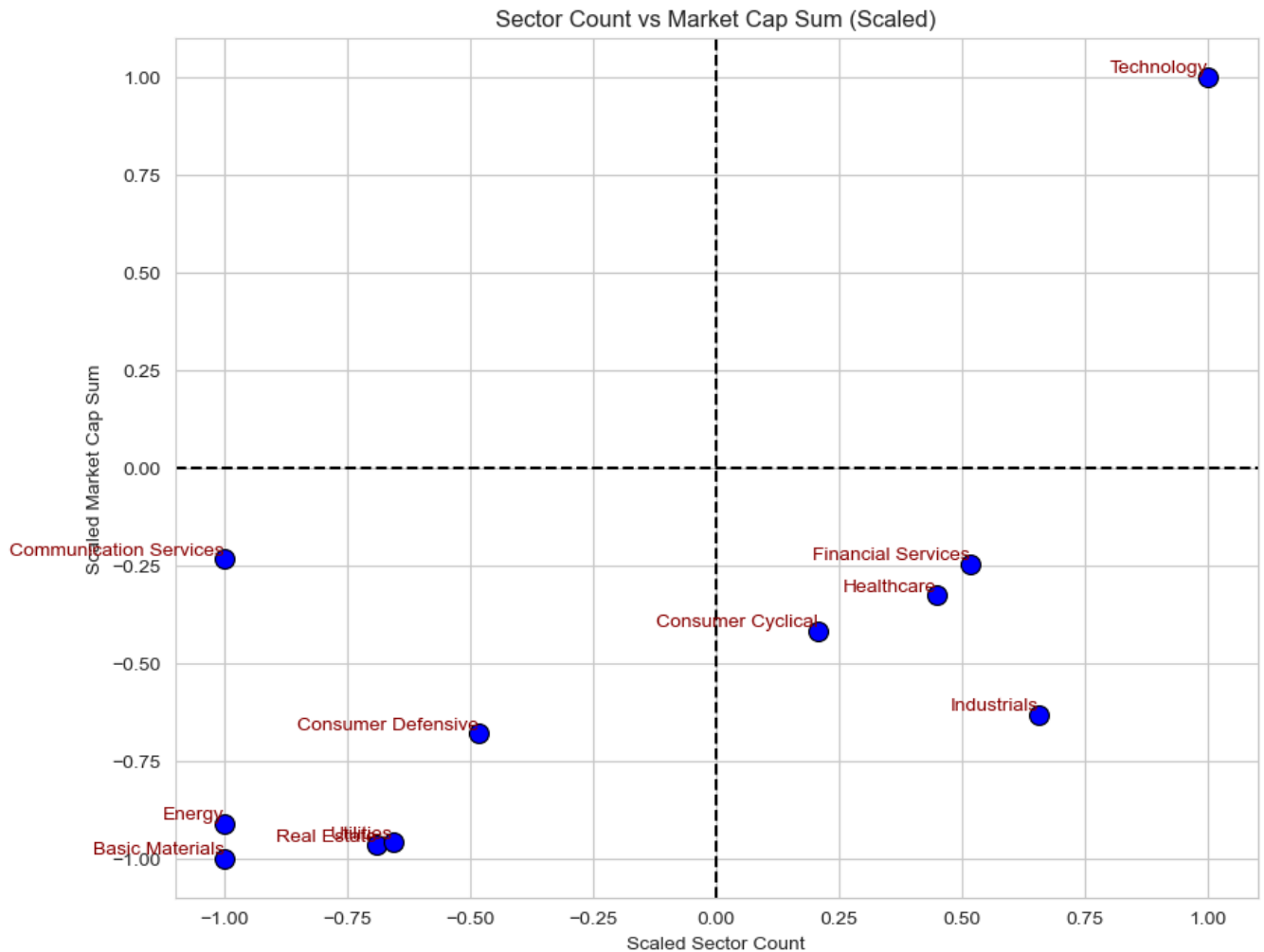
# Add labels to each data point
```

```
for i in range(Q3.shape[0]):
    plt.text(Q3['sector_count'][i], Q3['market_cap_sum'][i], Q3.index[i],
             fontsize=10, ha='right', va='bottom', color='darkred')

# Add grid lines at 0 for both axes
plt.axhline(0, color='black', linewidth=1.5, linestyle='--') # Horizontal grid line at y=0
plt.axvline(0, color='black', linewidth=1.5, linestyle='--') # Vertical grid line at x=0

# Set plot titles and labels
plt.title('Sector Count vs Market Cap Sum (Scaled)')
plt.xlabel('Scaled Sector Count')
plt.ylabel('Scaled Market Cap Sum')

plt.show()
```



## ✓ What Companies account for the most Market Capitalization

```
# Sort by market cap and take the top 50
QI = df.sort_values('marketCap', ascending=False).head(50)

# Create a figure with the desired size
plt.figure(figsize=(12, 8))

# Plot the barplot with financial colors and larger fonts
sns.barplot(data=QI, x='marketCap', y='symbol', hue='sector', palette='tab20')
```



```
# Add labels and title with bold fonts
plt.title('Top 50 Companies by Market Cap', fontsize=18, weight='bold', pad=20)
plt.xlabel('Market Cap', fontsize=14, weight='bold')
plt.ylabel('Company Symbol', fontsize=14, weight='bold')

# Add grid lines for better readability
plt.grid(color='grey', linestyle='--', linewidth=0.5, alpha=0.7)

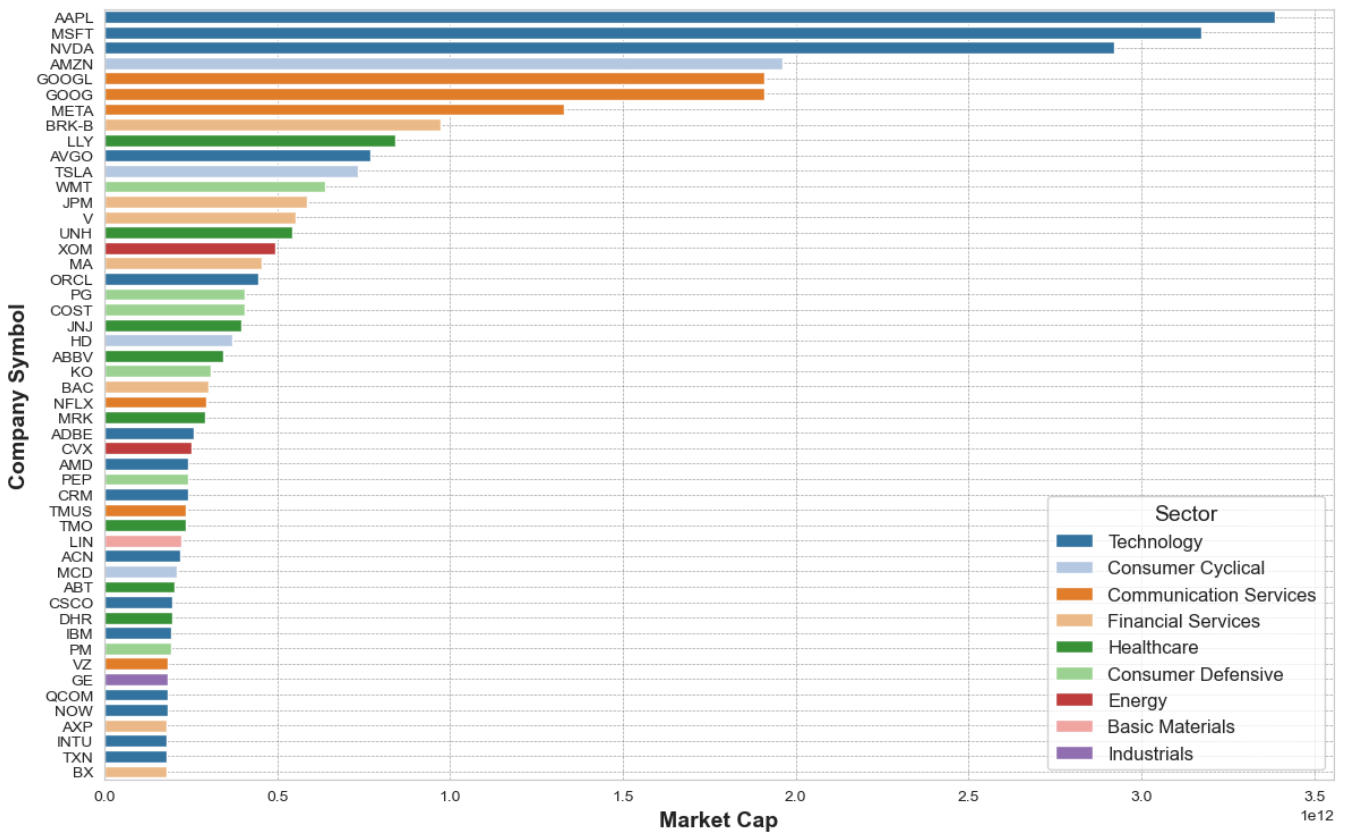
# Adjust legend placement and style
plt.legend(loc='lower right', title='Sector', fontsize=12, title_fontsize=14)

# Improve layout for PowerPoint
plt.tight_layout()

# Show or save the figure (save for PowerPoint if needed)
plt.savefig('top_50_market_cap.png', transparent=True) # If you want to save
plt.show()
```



## Top 50 Companies by Market Cap



# List of 35 stocks with the highest market cap in respective sector

```
tickers = ['AAPL', 'MSFT', 'NVDA', 'AVGO', 'ORCL', 'ADBE', 'CRM', 'AMD', 'ACN', # 9 Stocks from the Technology Sector
           'BRK-B', 'JPM', 'V', 'MA', 'BAC', 'WFC', # 6 Stocks from the Financial Services Sector
           'LLY', 'UNH', 'JNJ', 'ABBV', # 4 Stocks from the Healthcare Sector
           'GE', 'CAT', 'RTX', 'UNP', # 4 Stocks from the Industrials Sector
           'AMZN', 'TSLA', 'HD', # 3 Stocks from the Consumer Cyclical Sector
           'GOOGL', 'META', 'NFLX', # 3 Stocks from the Communication Services Sector
           'WMT', 'PG', # 2 Stocks from the Consumer Defensive Sector
```

```
'XOM', # 1 Stock from the Energy Sector
'PLD', # 1 Stock from the Real Estate Sector
'NEE', # 1 Stock from the Utilities Sector
'LIN' # 1 Stock from the Basic Materials Sector
]

QI_filtered = df[df['symbol'].isin(tickers)]
QI_filtered = QI_filtered.sort_values('marketCap', ascending=False)

# Create a figure with the desired size
plt.figure(figsize=(12, 8))

# Plot the barplot with financial colors and larger fonts
sns.barplot(data=QI_filtered, x='marketCap', y='symbol', hue='sector', palette='tab20')

# Add title with bold fonts
plt.title('My 35 Companies For a Diversified Portfolio', fontsize=18, weight='bold', pad=20)

# Remove x tick values, and x labels and y labels
plt.xticks([]) # Remove x tick values
plt.xlabel('') # Remove x label
plt.ylabel('') # Remove y label

# Add grid lines for better readability
plt.grid(color='grey', linestyle='--', linewidth=0.5, alpha=0.7)

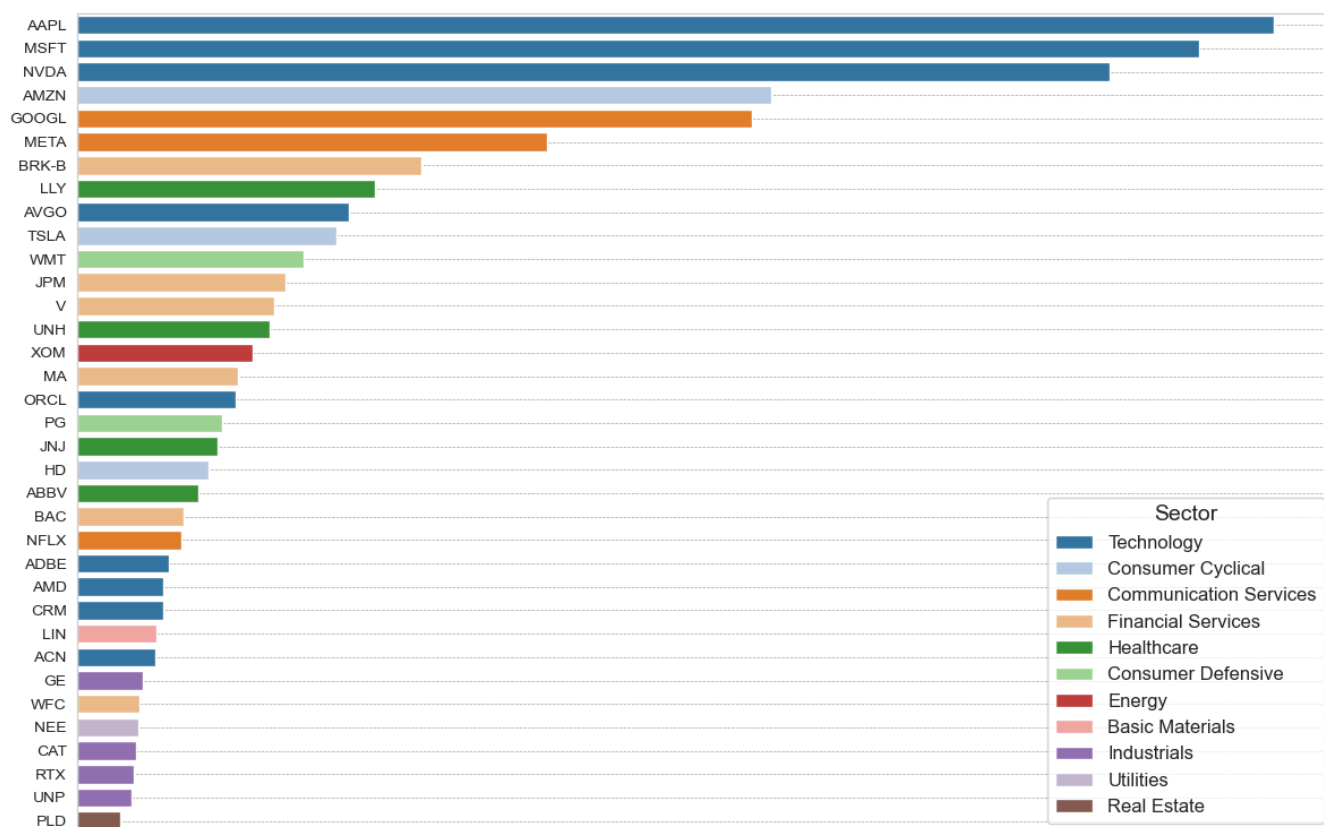
# Adjust legend placement and style
plt.legend(loc='lower right', title='Sector', fontsize=12, title_fontsize=14)

# Improve layout for PowerPoint
plt.tight_layout()

# Show or save the figure (save for PowerPoint if needed)
plt.savefig('my_35_market_cap.png', transparent=True) # If you want to save
plt.show()
```



### My 35 Companies For a Diversified Portfolio



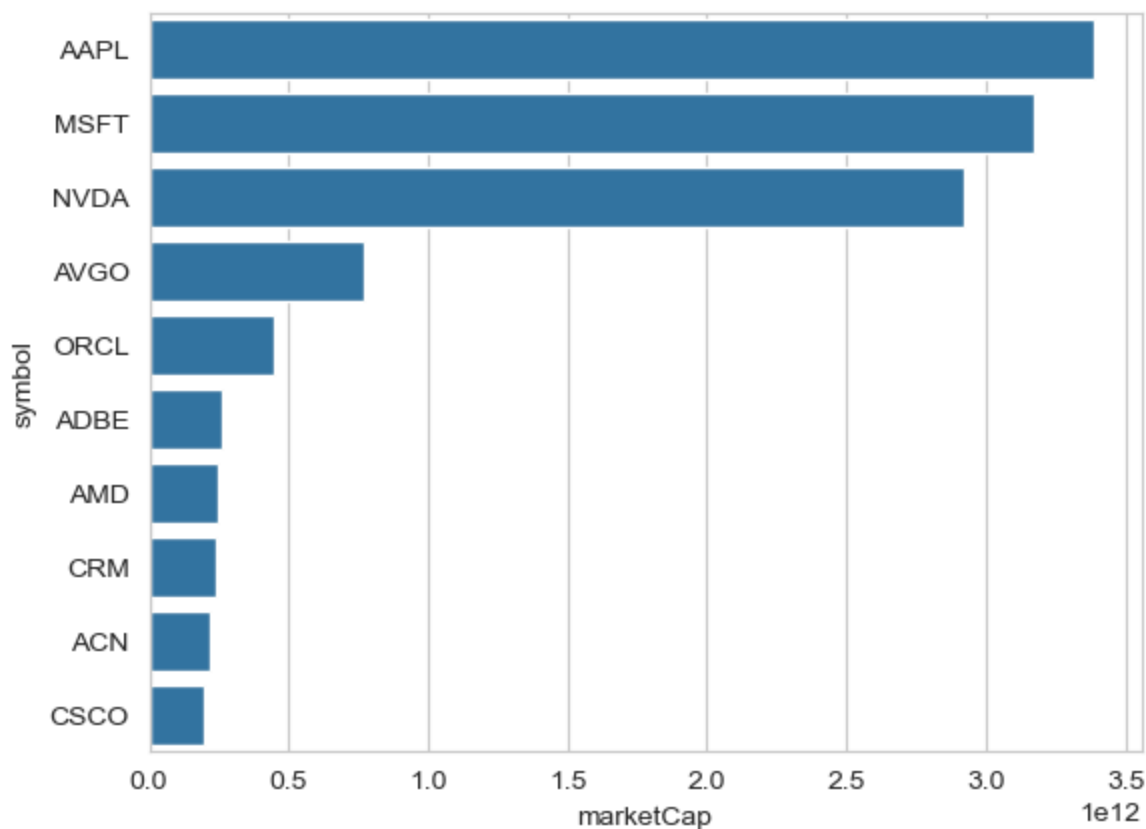
✓ What companies in the Technology sector account the most Market Capitalization?

```
Q4_MASK = df['sector'] == 'Technology'
```

```
Q4 = df[Q4_MASK][['symbol', 'marketCap']] # Select both 'symbol' and 'marketCap' columns
Q4 = Q4.sort_values('marketCap', ascending=False).head(10)
```

```
sns.barplot(x='marketCap', y='symbol', data=Q4)
```

↩ <Axes: xlabel='marketCap', ylabel='symbol'>



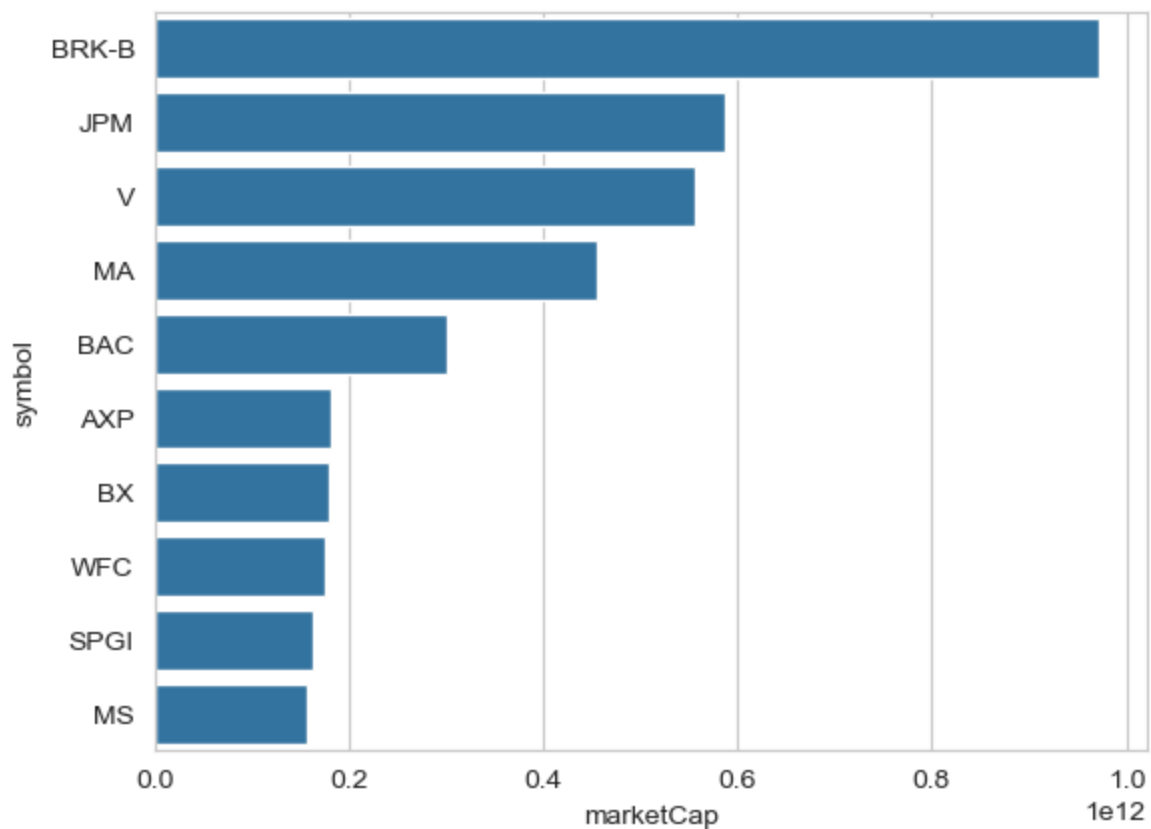
✓ What companies in the Financial Services sector account the most Market Capitalization?

```
Q5_MASK = df['sector'] == 'Financial Services'
```

```
Q5 = df[Q5_MASK][['symbol', 'marketCap']]
Q5 = Q5.sort_values('marketCap', ascending=False).head(10)
```

```
sns.barplot(x='marketCap', y='symbol', data=Q5)
```

<Axes: xlabel='marketCap', ylabel='symbol'>



✓ What companies in the Healthcare sector account the most Market Capitalization?

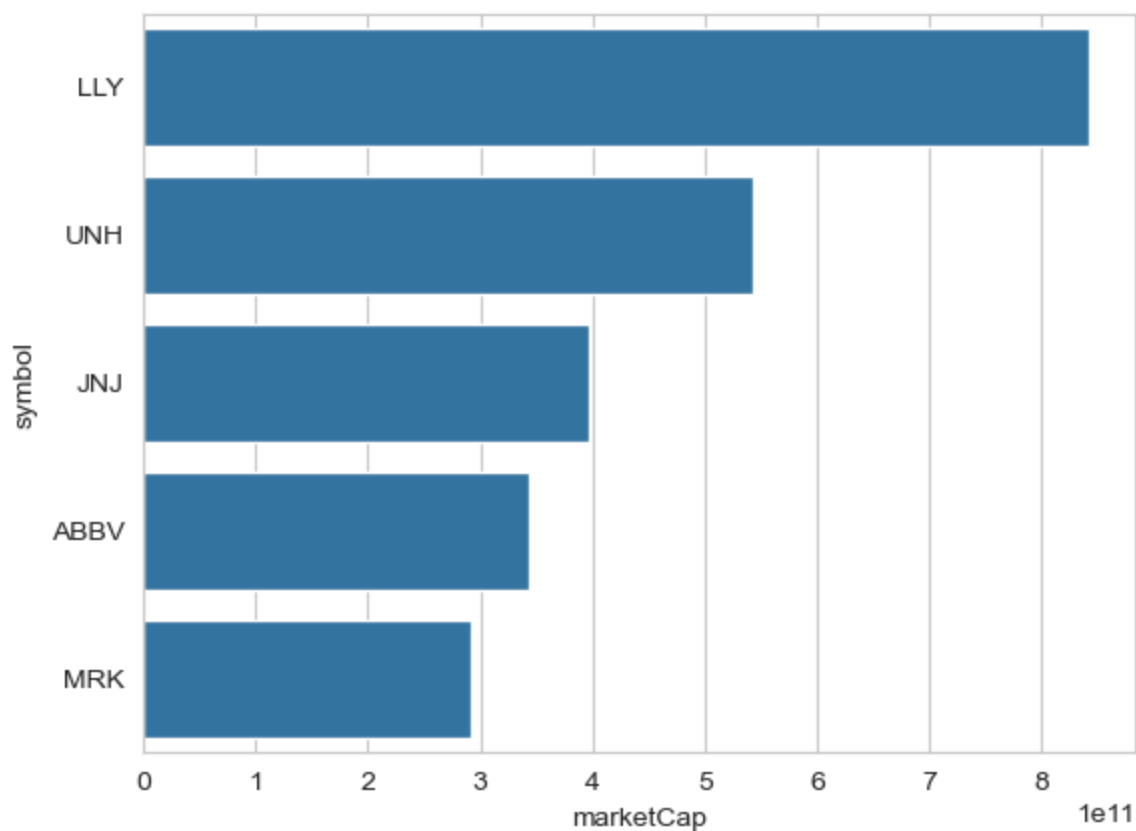
```
Q6_MASK = df['sector'] == 'Healthcare'
```

```
Q6 = df[Q6_MASK][['symbol', 'marketCap']]
```

```
Q6 = Q6.sort_values('marketCap', ascending=False).head()
```

```
sns.barplot(x='marketCap', y='symbol', data=Q6)
```

↔ <Axes: xlabel='marketCap', ylabel='symbol'>

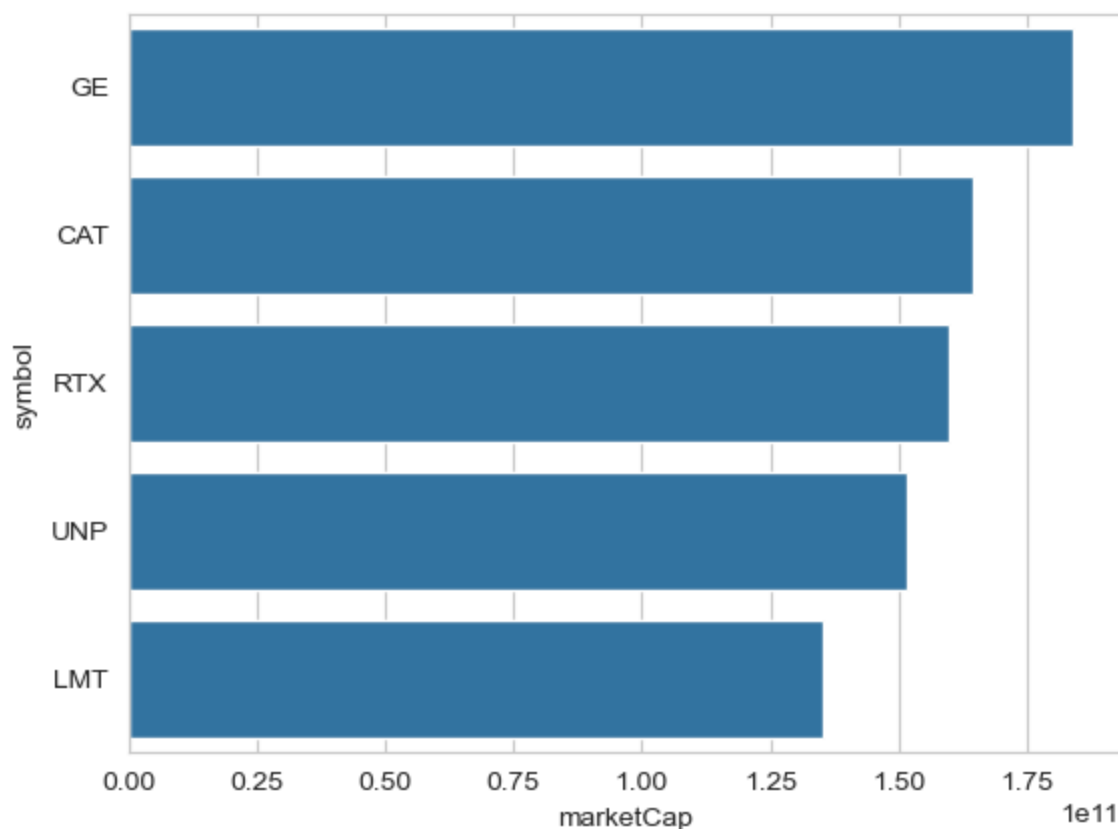


✓ What companies in the Industrials sector account the most Market Capitalization?

```
# Filter for Industrials sector
Q7_MASK = df['sector'] == 'Industrials'
Q7 = df[Q7_MASK][['symbol', 'marketCap']]
Q7 = Q7.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q7)
```

↔ <Axes: xlabel='marketCap', ylabel='symbol'>



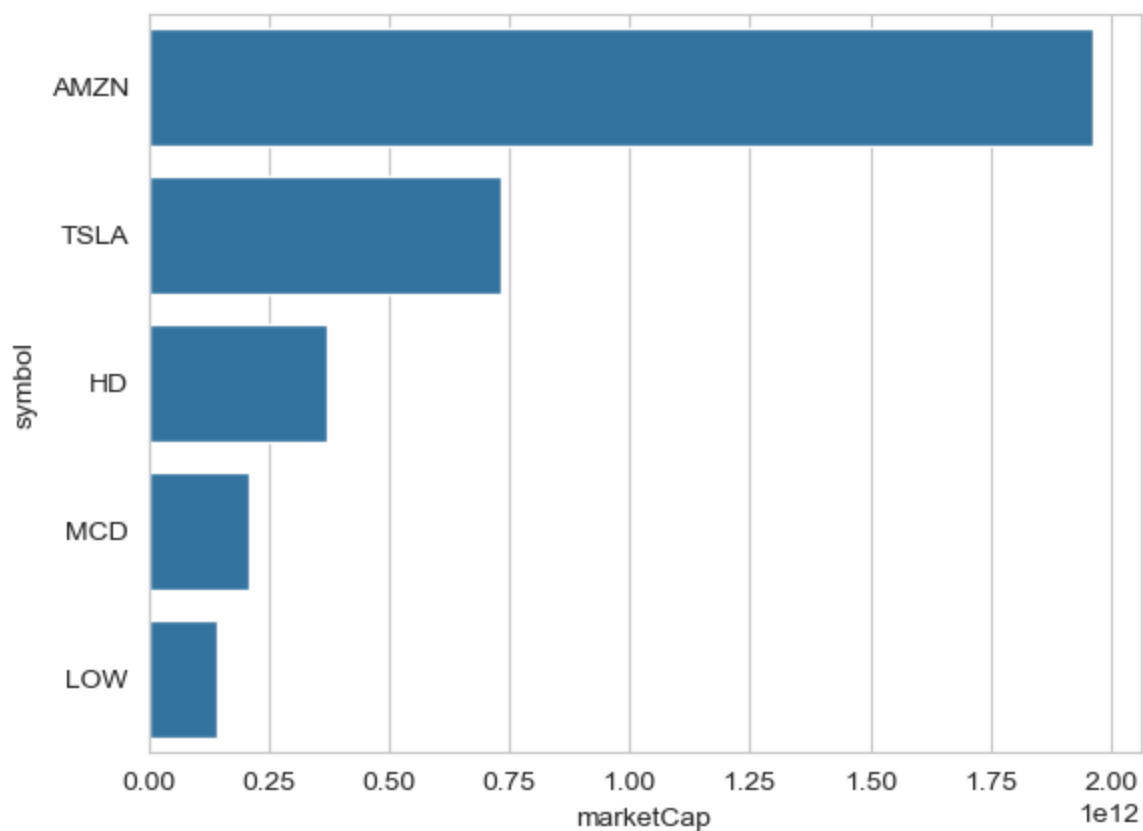
✓ What companies in the Consumer Cyclical sector account the most Market Capitalization?

```
# Filter for Consumer Cyclical sector
Q8_MASK = df['sector'] == 'Consumer Cyclical'
Q8 = df[Q8_MASK][['symbol', 'marketCap']]
Q8 = Q8.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q8)
```



<Axes: xlabel='marketCap', ylabel='symbol'>

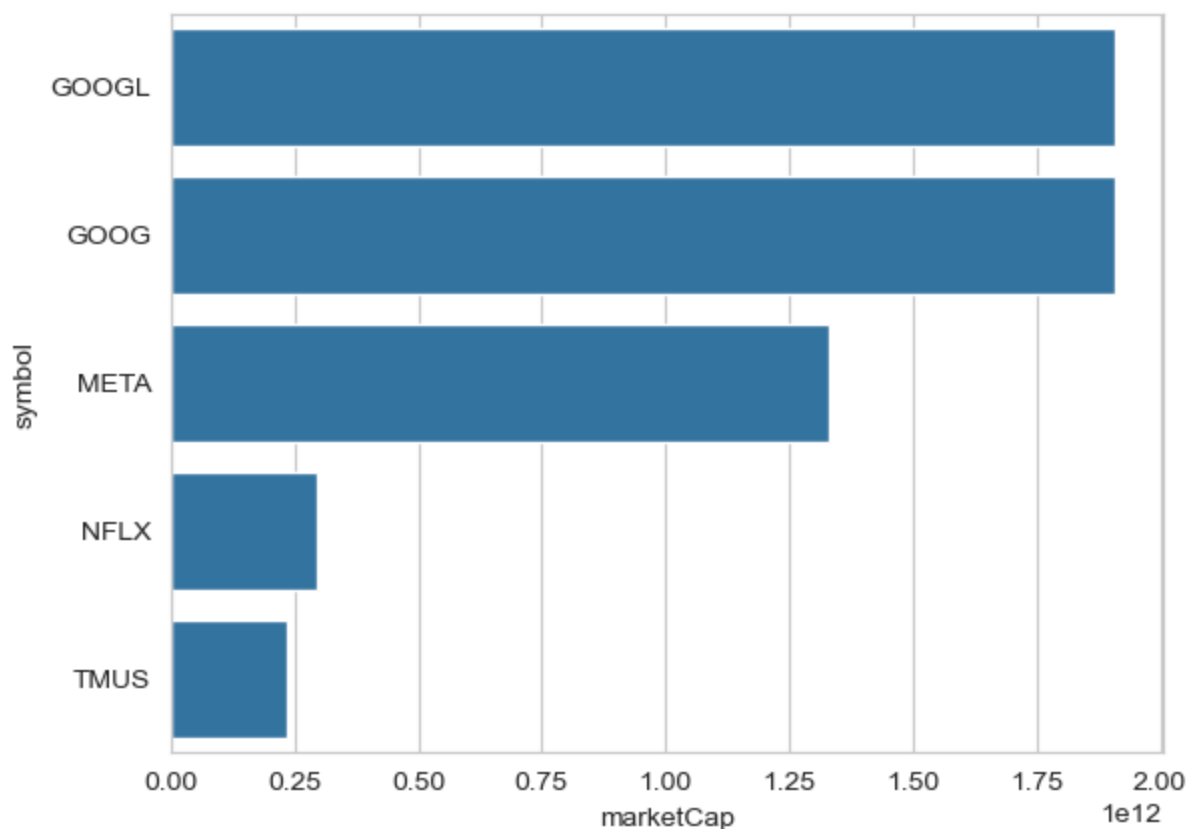


✓ What companies in the Communication Services sector account the most Market Capitalization?

```
# Filter for Communication Services sector
Q9_MASK = df['sector'] == 'Communication Services'
Q9 = df[Q9_MASK][['symbol', 'marketCap']]
Q9 = Q9.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q9)
```

 <Axes: xlabel='marketCap', ylabel='symbol'>

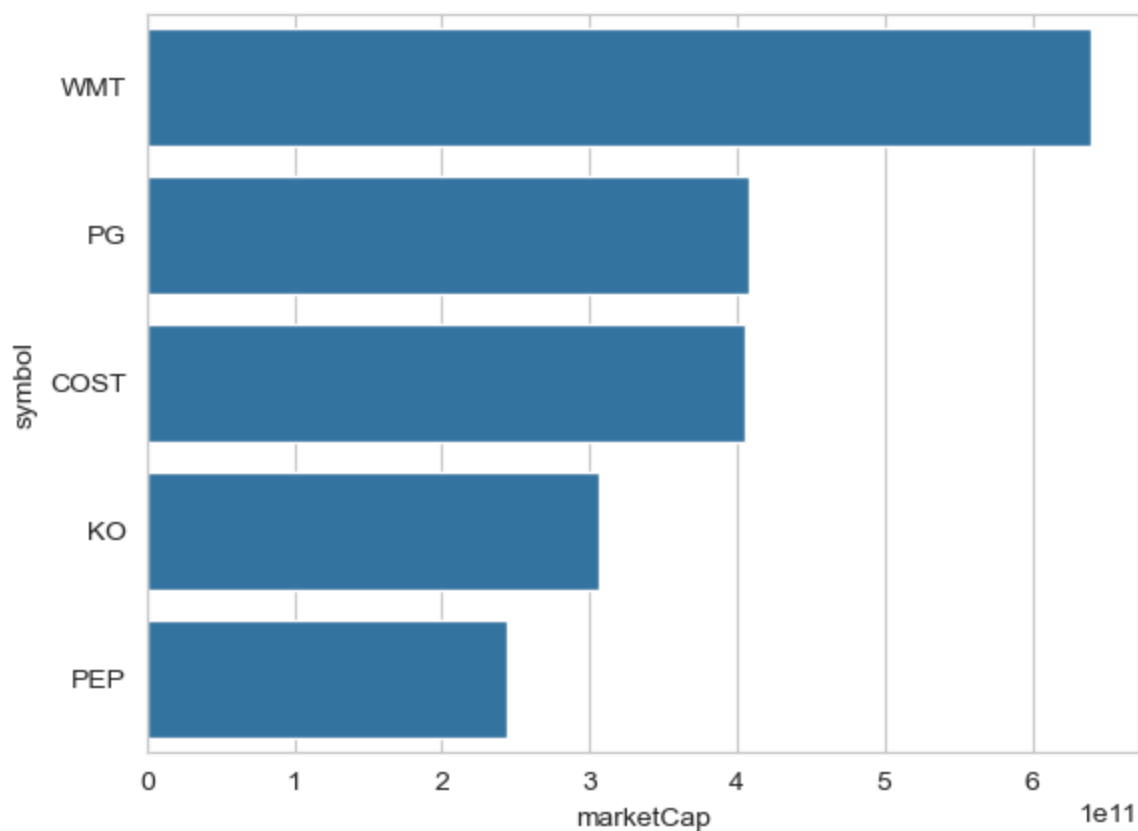


✓ What companies in the Consumer Defensive sector account the most Market Capitalization?

```
# Filter for Consumer Defensive sector
Q10_MASK = df['sector'] == 'Consumer Defensive'
Q10 = df[Q10_MASK][['symbol', 'marketCap']]
Q10 = Q10.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q10)
```

<Axes: xlabel='marketCap', ylabel='symbol'>

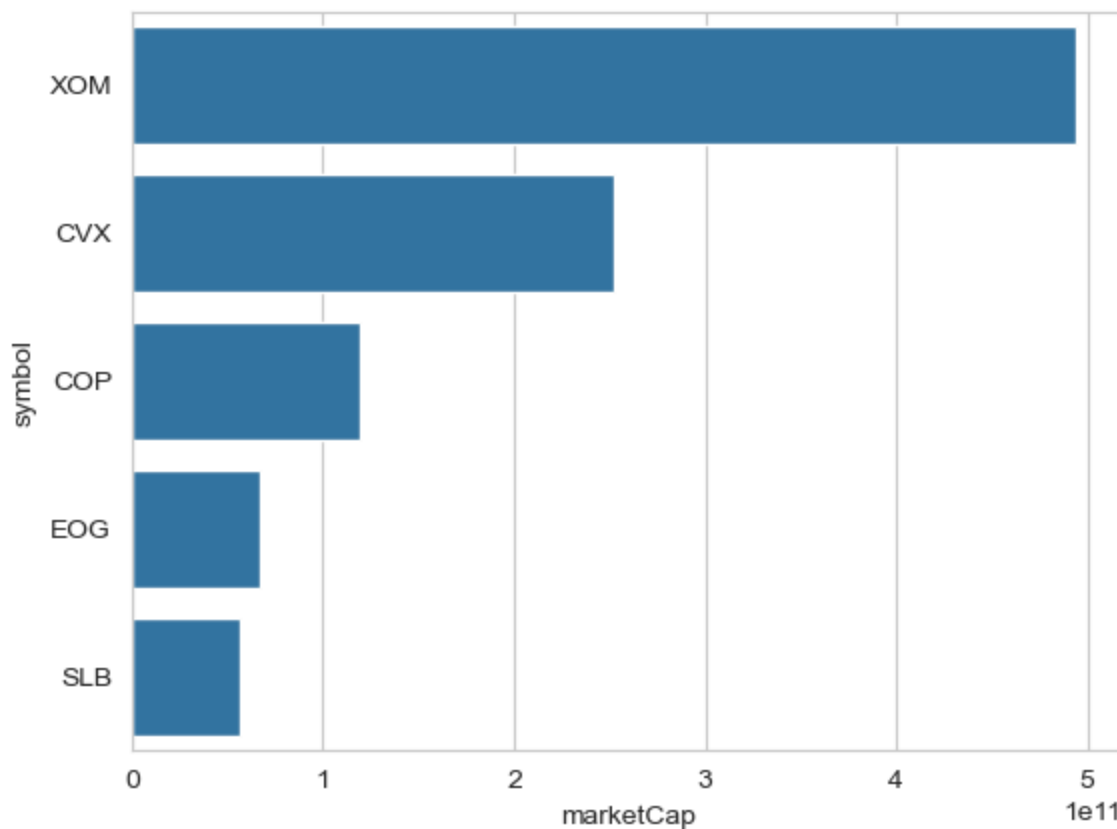


✓ What companies in the Energy sector account the most Market Capitalization?

```
# Filter for Energy sector
Q11_MASK = df['sector'] == 'Energy'
Q11 = df[Q11_MASK][['symbol', 'marketCap']]
Q11 = Q11.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q11)
```

 <Axes: xlabel='marketCap', ylabel='symbol'>

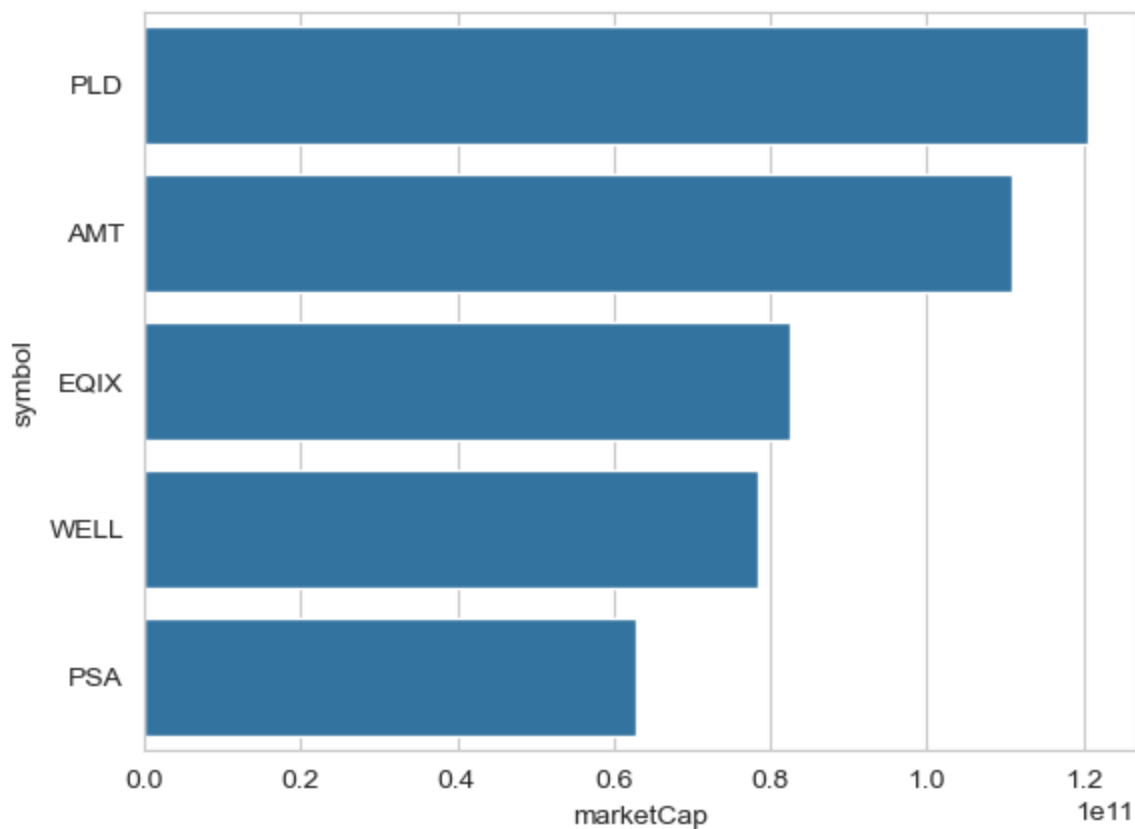


✓ What companies in the Real Estate sector account the most Market Capitalization?

```
# Filter for Real Estate sector
Q12_MASK = df['sector'] == 'Real Estate'
Q12 = df[Q12_MASK][['symbol', 'marketCap']]
Q12 = Q12.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q12)
```

<Axes: xlabel='marketCap', ylabel='symbol'>

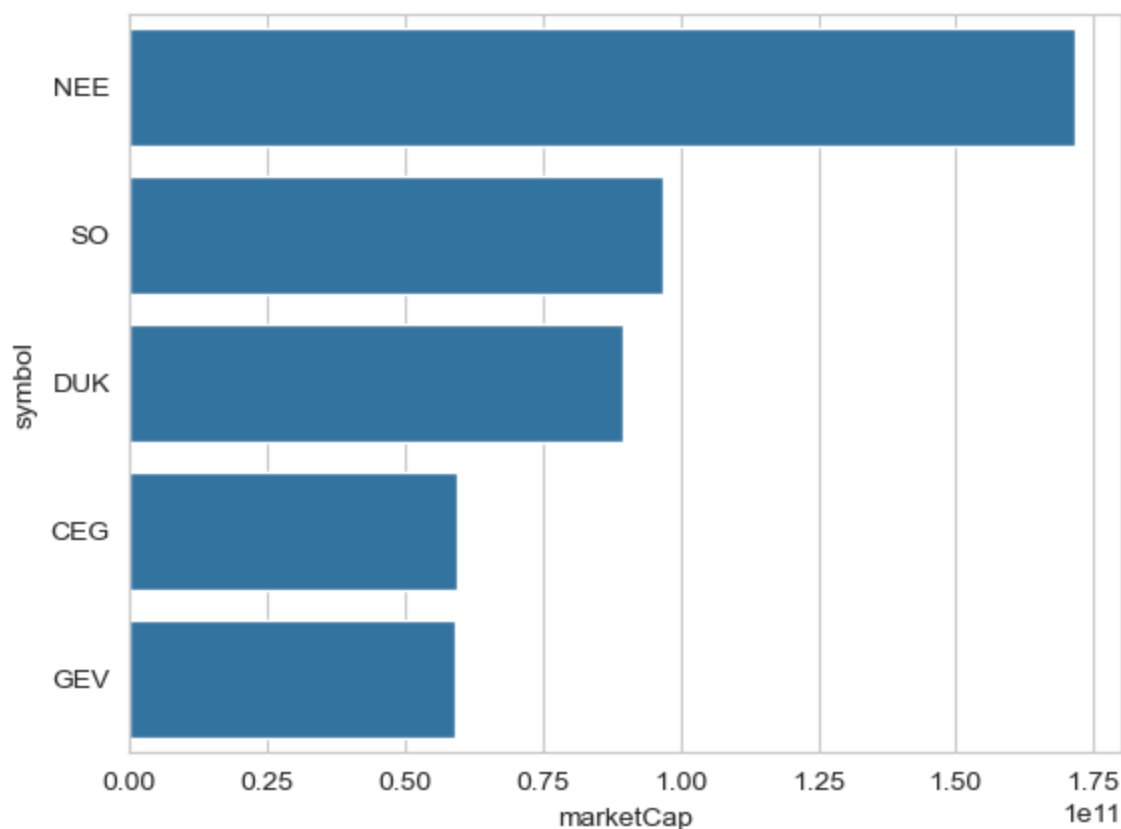


✓ What companies in the Utilities sector account the most Market Capitalization?

```
# Filter for Utilities sector
Q13_MASK = df['sector'] == 'Utilities'
Q13 = df[Q13_MASK][['symbol', 'marketCap']]
Q13 = Q13.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q13)
```

↔ <Axes: xlabel='marketCap', ylabel='symbol'>

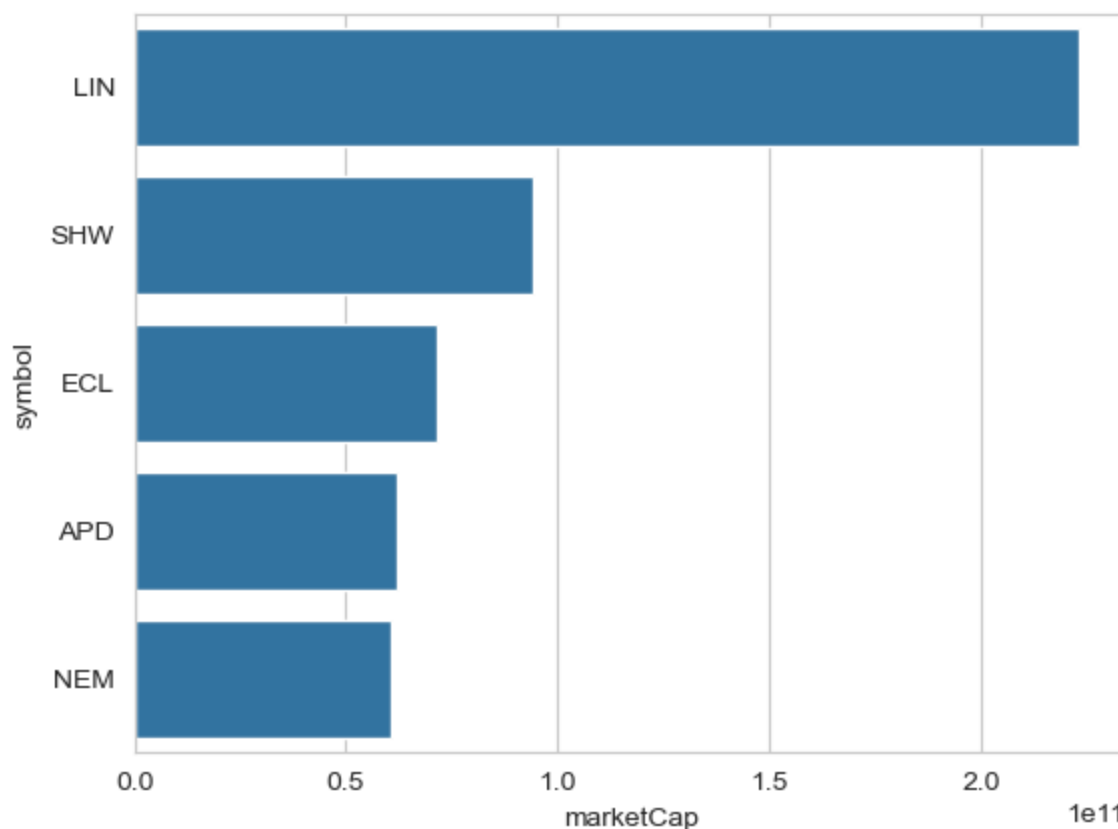


✓ What companies in the Basic Materials sector account the most Market Capitalization?

```
# Filter for Basic Materials sector
Q14_MASK = df['sector'] == 'Basic Materials'
Q14 = df[Q14_MASK][['symbol', 'marketCap']]
Q14 = Q14.sort_values('marketCap', ascending=False).head()
```

```
# Plot
sns.barplot(x='marketCap', y='symbol', data=Q14)
```


 <Axes: xlabel='marketCap', ylabel='symbol'>



## ✓ Inferential Analysis

For my Inferential Analysis, I went through yfinance and found some metrics that I would consider to be important when it comes to analyzing a stock and its price. So I collected all those metrics in my dataset earlier and now im going to see if they have a significant difference when it comes to predicting the closing price of a stock.

```
# looking at the distribution for closing prices
df['previousClose'].describe()
```

 count 503.000000  
 mean 216.408178  
 std 492.300016  
 min 6.940000  
 25% 67.945000  
 50% 120.580000  
 75% 230.625000  
 max 9088.120000  
 Name: previousClose, dtype: float64

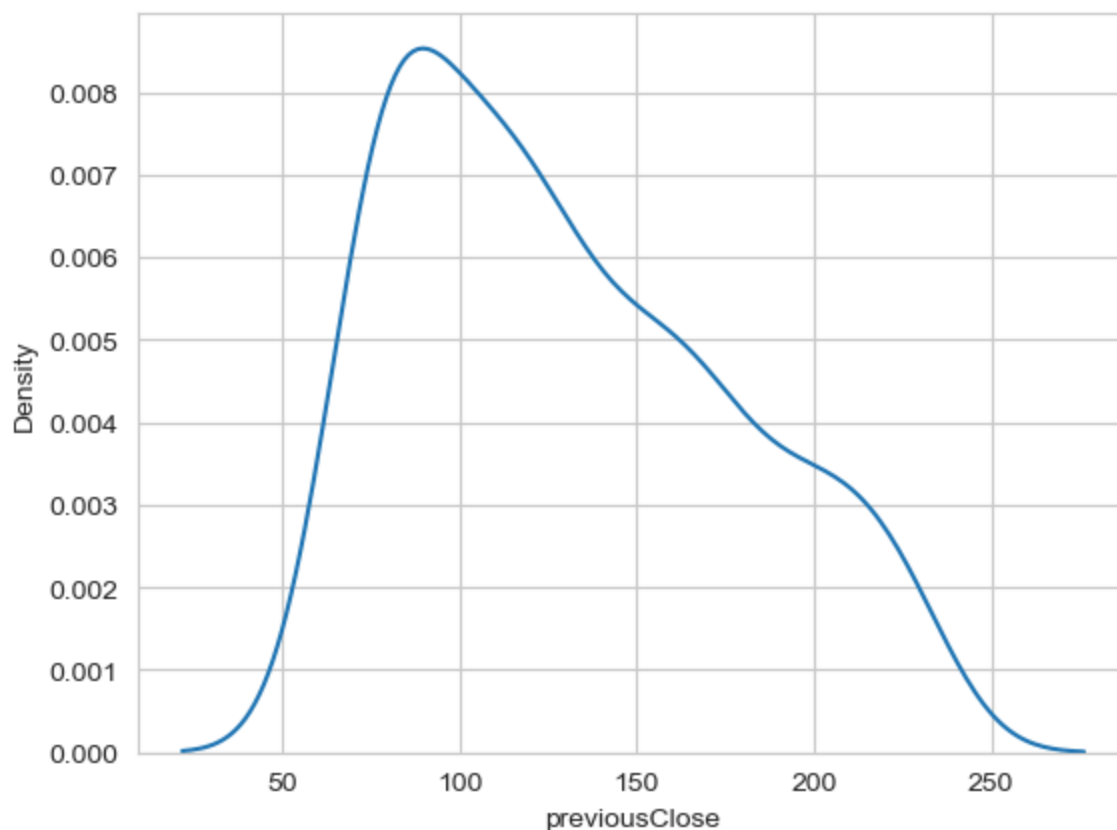
the data has some ridiculous outliers so I decided to just take the closing prices between the 25th and 75th percentile

```
# filtering the data
percentile_25 = df['previousClose'].quantile(0.25)
percentile_75 = df['previousClose'].quantile(0.75)

# Filter the DataFrame to keep only rows with values between 25th and 75th percentiles
filtered_df = df[(df['previousClose'] >= percentile_25) & (df['previousClose'] <= percentile_75)]

# a visual display of the distribution
sns.kdeplot(data=filtered_df['previousClose'])
```

↩ <Axes: xlabel='previousClose', ylabel='Density'>



testing the data for normality and to determine what test i should use

```
# Perform Shapiro-Wilk test
stat, p_value = stats.shapiro(filtered_df['previousClose'])

print("Shapiro-Wilk Test Statistic:", stat)
print("Shapiro-Wilk Test p-value:", p_value)
```

```
# Interpretation
```



```
alpha = 0.05
if p_value > alpha:
    print("Sample looks Gaussian (fail to reject H0)")
else:
    print("Sample does not look Gaussian (reject H0)")
```

```
→ Shapiro-Wilk Test Statistic: 0.9317305424456671
Shapiro-Wilk Test p-value: 2.2652754559288738e-09
Sample does not look Gaussian (reject H0)
```

```
# Perform Kolmogorov-Smirnov test against a normal distribution
stat, p_value = stats.kstest(filtered_df['previousClose'], 'norm')
```

```
print("Kolmogorov-Smirnov Test Statistic:", stat)
print("Kolmogorov-Smirnov Test p-value:", p_value)
```

```
# Interpretation
alpha = 0.05
if p_value > alpha:
    print("Sample looks Gaussian (fail to reject H0)")
else:
    print("Sample does not look Gaussian (reject H0)")
```

```
→ Kolmogorov-Smirnov Test Statistic: 1.0
Kolmogorov-Smirnov Test p-value: 0.0
Sample does not look Gaussian (reject H0)
```

neither of the normality tests i tried worked, lets see if i can apply CLT

```
# Parameters
sample_size = 30
num_samples = 1000

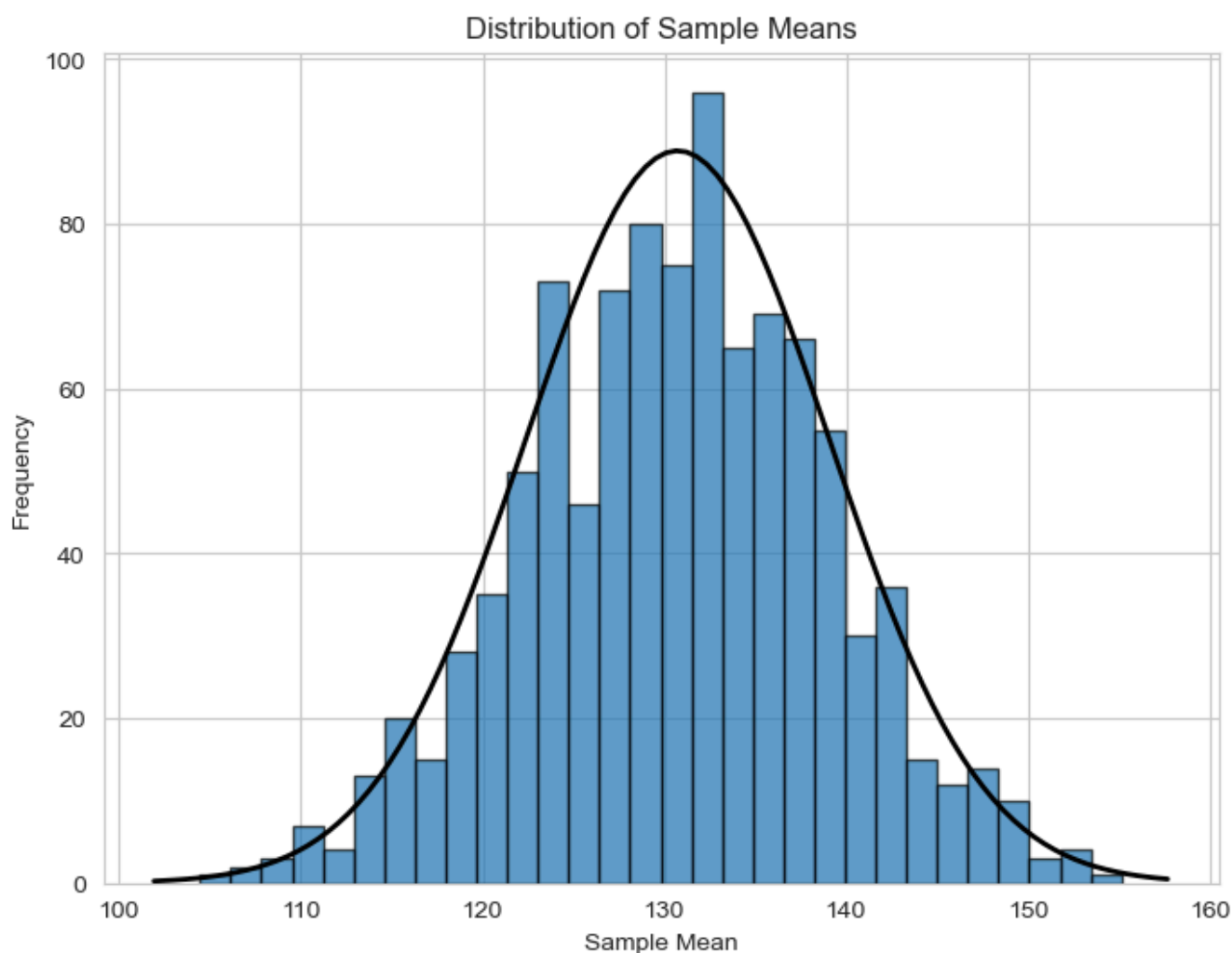
# Generate sample means
sample_means = []
for _ in range(num_samples):
    sample = filtered_df['previousClose'].sample(sample_size, replace=True)
    sample_means.append(sample.mean())

# Convert list to a pandas Series
sample_means = pd.Series(sample_means)

# Plot the distribution of sample means
plt.figure(figsize=(8, 6))
plt.hist(sample_means, bins=30, edgecolor='k', alpha=0.7)
plt.title('Distribution of Sample Means')
plt.xlabel('Sample Mean')
plt.ylabel('Frequency')
```

```
# Add a normal distribution curve for comparison
mu, std = sample_means.mean(), sample_means.std()
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, mu, std)
plt.plot(x, p * len(sample_means) * (xmax - xmin) / 30, 'k', linewidth=2)

plt.show()
```



after applying central limit theorem, I can start moving on to the hypothesis tests

I decided to scale all my metrics as it would make it easier for me to read them on a graph.

```
keys = ['trailingPE', 'forwardPE', 'volume', 'marketCap',
        'enterpriseValue', 'profitMargins', 'sharesOutstanding', 'bookValue',
        'priceToBook', 'trailingEps', 'forwardEps', 'pegRatio', 'enterpriseToRevenue',
        'enterpriseToEbitda', 'totalCash', 'totalCashPerShare', 'ebitda', 'totalDebt',
        'quickRatio', 'currentRatio', 'totalRevenue', 'debtToEquity', 'revenuePerShare',
```

```

    'returnOnAssets', 'returnOnEquity', 'freeCashflow', 'operatingCashflow',
    'earningsGrowth', 'revenueGrowth', 'grossMargins', 'ebitdaMargins',
    'operatingMargins', 'trailingPegRatio', 'priceToSales'
]

```

```
scaler = MinMaxScaler(feature_range=(-10, 10))
```

creating and storing distribution plots for each of the metrics

```

import io

# Dictionary to store the plots
plots = {}

def generate_plots():
    global plots
    plots = {}
    for key in keys:
        if key in filtered_df.columns:
            # Standardize the values
            filtered_df[key] = scaler.fit_transform(filtered_df[[key]])

            # Create and store the KDE plot
            plt.figure(figsize=(6, 4))
            sns.kdeplot(filtered_df[key], shade=True)
            plt.title(f"KDE Plot for {key} (Standardized)")
            plt.xlabel(f"{key} (Standardized)")
            plt.ylabel("Density")

            # Save plot to a BytesIO object
            buf = io.BytesIO()
            plt.savefig(buf, format='png')
            buf.seek(0)
            plots[key] = buf
            plt.close()

def show_plot(key):
    if key in plots:
        buf = plots[key]
        buf.seek(0)
        img = plt.imread(buf)
        plt.figure(figsize=(6, 4))
        plt.imshow(img)
        plt.axis('off') # Turn off the axis
        plt.show()
    else:
        print(f"No plot found for key: {key}")

```

# Call this function to generate and store all plots

```
generate_plots()
```

creating the functions necessary to quickly perform hypothesis tests

```
def eleven_way_ANOVA(dataset1, dataset2, dataset3, dataset4, dataset5, dataset6, dataset7, c
    label1='', label2='', label3='', label4='', label5='', label6='', label
    save=False):

    # Calculate the means of all datasets
    means = [np.mean(ds) for ds in [dataset1, dataset2, dataset3, dataset4, dataset5, dataset6, dataset7]]

    # Perform ANOVA test
    _, p_value = stats.f_oneway(dataset1, dataset2, dataset3, dataset4, dataset5, dataset6, dataset7)
    if p_value < 0.05:
        print('Reject the Null Hypothesis')
    else:
        print('Fail to reject the Null Hypothesis')

    # Use a style template
    sns.set_style('whitegrid')

    plt.figure(figsize=(12, 8))

    # Define the datasets and labels for easy looping
    datasets = [dataset1, dataset2, dataset3, dataset4, dataset5, dataset6, dataset7, dataset8, dataset9, dataset10, dataset11]
    labels = [label1, label2, label3, label4, label5, label6, label7, label8, label9, label10, label11]
    colors = sns.color_palette('tab10', 11) # Using 11 different colors from Seaborn's palette

    # Plot all datasets using kdeplot
    for ds, label, color in zip(datasets, labels, colors):
        sns.kdeplot(ds, label=label, color=color, alpha=1)

    # Generate KDE values for each dataset
    kdes = [stats.gaussian_kde(ds) for ds in datasets]

    # Generate x values for KDE
    x_vals = np.linspace(min(min(ds) for ds in datasets), max(max(ds) for ds in datasets), 100)

    # Fill the entire area under the curves for each dataset
    # for kde, color in zip(kdes, colors):
    #     plt.fill_between(x_vals, kde(x_vals), color=color, alpha=0.3)

    # Plot vertical lines at the means for each dataset, stopping at the KDE curves
    for mean, kde, label, color in zip(means, kdes, labels, colors):
        plt.plot([mean, mean], [0, kde(mean)[0]], '--', color=color, label=f'{label} Mean: {mean}')

    # Add labels and title
    plt.title('Comparison of 11 Groups', fontsize=20, fontweight='bold')
    plt.xlabel('Value', fontsize=16)
```

```

plt.ylabel('')
plt.yticks(visible=False)

# Add annotations for each dataset
for i, (label, mean, color) in enumerate(zip(labels, means, colors)):
    plt.annotate(f'{label} | Mean: {mean:.2f}', xy=(0.75, 0.85 - i*0.05), xycoords='axes',
                bbox=dict(facecolor='white', edgecolor=color, boxstyle='round,pad=0.5'))

if save:
    plt.savefig(f'ANOVA.png', transparent=True)

plt.show()

def two_sample_kdeplot(dataset=filtered_df, metric='', label1='Greater than Average', label2='Less than Average'):
    # Remove missing values in the metric column to prevent issues
    dataset = dataset.dropna(subset=[metric, 'previousClose'])

    # Calculate the mean of the metric
    mean_value = dataset[metric].mean()

    # Create masks for values greater and less than the mean
    MASK1 = dataset[metric] >= mean_value
    MASK2 = dataset[metric] < mean_value

    # Filter datasets based on the masks
    dataset1 = dataset[MASK1]['previousClose']
    dataset2 = dataset[MASK2]['previousClose']

    # Check if either dataset is too small for KDE
    if len(dataset1) < 2 or len(dataset2) < 2:
        print(f"One of the datasets for {metric} has fewer than 2 elements. Unable to compute KDE")
        return

    # Calculate the means of both datasets
    mean1 = np.mean(dataset1)
    mean2 = np.mean(dataset2)

    # adjusting the labels
    label1 = (f'{label1} {metric}')
    label2 = (f'{label2} {metric}')

    # Perform two sample t-test
    _, p_value = stats.ttest_ind(dataset1, dataset2)
    if p_value < 0.05:
        print('Reject the Null Hypothesis')
    else:
        print('Fail to reject the null hypothesis')

    # Use a style template
    sns.set_style('whitegrid')

```

```

plt.figure(figsize=(12, 8))

# Plot both datasets
sns.kdeplot(dataset1, label=label1, color='red', alpha=1)
sns.kdeplot(dataset2, label=label2, color='blue', alpha=1)

# Compute KDE values
kde_dataset1 = stats.gaussian_kde(dataset1)
kde_dataset2 = stats.gaussian_kde(dataset2)

# Generate x values for KDE
x_vals = np.linspace(min(dataset1.min(), dataset2.min()), max(dataset1.max(), dataset2.max()), 100)

# Fill the entire area under the curves
plt.fill_between(x_vals, kde_dataset1(x_vals), color='red', alpha=0.3)
plt.fill_between(x_vals, kde_dataset2(x_vals), color='blue', alpha=0.3)

# Get KDE values at means
kde_dataset1_at_mean1 = kde_dataset1(mean1)[0]
kde_dataset2_at_mean2 = kde_dataset2(mean2)[0]

# Add vertical lines at the means, stopping at the KDE curves
plt.plot([mean1, mean1], [0, kde_dataset1_at_mean1], 'r--', label=f'{label1} Mean: {mean1:.2f}')
plt.plot([mean2, mean2], [0, kde_dataset2_at_mean2], 'b--', label=f'{label2} Mean: {mean2:.2f}')

# Add labels and title
plt.title(f'Test Between {label1} vs {label2}', fontsize=20, fontweight='bold')
plt.xlabel('', fontsize=16)
plt.ylabel('')
plt.yticks(visible=False)

# Add annotations
plt.annotate(f'{label1} | Mean: {mean1:.2f}', xy=(0.75, 0.85), xycoords='axes fraction',
            bbox=dict(facecolor='white', edgecolor='red', boxstyle='round,pad=0.5'))
plt.annotate(f'{label2} | Mean: {mean2:.2f}', xy=(0.75, 0.75), xycoords='axes fraction',
            bbox=dict(facecolor='white', edgecolor='blue', boxstyle='round,pad=0.5'))

if save == True:
    plt.savefig(f'{label1} v {label2} TTest.png', transparent=True)
plt.show()

def descriptive_visual(dataset=filtered_df, metric='', save=False):
    # Drop rows with NaN values in 'metric' and 'previousClose' columns
    dataset = dataset.dropna(subset=[metric, 'previousClose'])

    # Calculate the mean of the specified metric
    mean_value = dataset[metric].mean()

```

```

# Create masks for values greater and less than the mean
mask_above_mean = dataset[metric] >= mean_value
mask_below_mean = dataset[metric] < mean_value

# Filter datasets based on the masks and calculate the mean 'previousClose' for each group
mean_previous_close_above = dataset[mask_above_mean]['previousClose'].mean()
mean_previous_close_below = dataset[mask_below_mean]['previousClose'].mean()

# Set theme colors to match financial analysis (green for positive, red for negative)
labels = ['Above Mean', 'Below Mean']
mean_values = [mean_previous_close_above, mean_previous_close_below]
colors = ['#4CAF50', '#F44336'] # Green for "above", red for "below"

# Create a styled bar graph
plt.figure(figsize=(8, 6))
plt.bar(labels, mean_values, color=colors, edgecolor='black', linewidth=1.5)

# Set y-axis limit to 150
plt.ylim(0, 150)

# Add titles and labels
plt.ylabel('Mean Close ($)', fontsize=18, fontweight='bold')
plt.title(f'Mean Close for {metric} Above and Below Average', fontsize=20, fontweight='bold')

# Customize ticks and grid for presentation polish
plt.xticks(fontsize=16, fontweight='bold')
plt.yticks(fontsize=16, fontweight='bold')
plt.grid(True, axis='y', linestyle='--', alpha=0.7)

# Add annotations to show mean values on the bars
for i, value in enumerate(mean_values):
    plt.text(i, value + 1.5, f'${value:.2f}', ha='center', fontsize=14, fontweight='bold')

# Remove the top and right spines to clean up the plot
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Show the plot
plt.tight_layout()
if save == True:
    plt.savefig(f'{metric}.png', transparent=True)
plt.show()
plt.show()

```

## ✓ Hypothesis 1: Sector ANOVA

Null Hypothesis ( $H_0$ ):  $\mu_T = \mu_{FS} = \mu_{HC} = \mu_I = \mu_{CC} = \mu_{CS} = \mu_{CD} = \mu_E = \mu_{RE} = \mu_U = \mu_{BM}$

Alternate Hypothesis ( $H_A$ ):  $\mu_T \neq \mu_{FS} \neq \mu_{HC} \neq \mu_I \neq \mu_{CC} \neq \mu_{CS} \neq \mu_{CD} \neq \mu_E \neq \mu_{RE} \neq \mu_U \neq \mu_{BM}$

The null hypothesis states that the mean closing price for companies in each sector are equal

The alternate hypothesis states that the null hypothesis is not true

```
MASK_T = filtered_df['sector'] == 'Technology'
MASK_FS = filtered_df['sector'] == 'Financial Services'
MASK_HC = filtered_df['sector'] == 'Healthcare'
MASK_I = filtered_df['sector'] == 'Industrials'
MASK_CC = filtered_df['sector'] == 'Consumer Cyclical'
MASK_CS = filtered_df['sector'] == 'Communication Services'
MASK_CD = filtered_df['sector'] == 'Consumer Defensive'
MASK_E = filtered_df['sector'] == 'Energy'
MASK_RE = filtered_df['sector'] == 'Real Estate'
MASK_U = filtered_df['sector'] == 'Utilities'
MASK_BM = filtered_df['sector'] == 'Basic Materials'

H1_T = filtered_df[MASK_T]['previousClose']
H1_FS = filtered_df[MASK_FS]['previousClose']
H1_HC = filtered_df[MASK_HC]['previousClose']
H1_I = filtered_df[MASK_I]['previousClose']
H1_CC = filtered_df[MASK_CC]['previousClose']
H1_CS = filtered_df[MASK_CS]['previousClose']
H1_CD = filtered_df[MASK_CD]['previousClose']
H1_E = filtered_df[MASK_E]['previousClose']
H1_RE = filtered_df[MASK_RE]['previousClose']
H1_U = filtered_df[MASK_U]['previousClose']
H1_BM = filtered_df[MASK_BM]['previousClose']

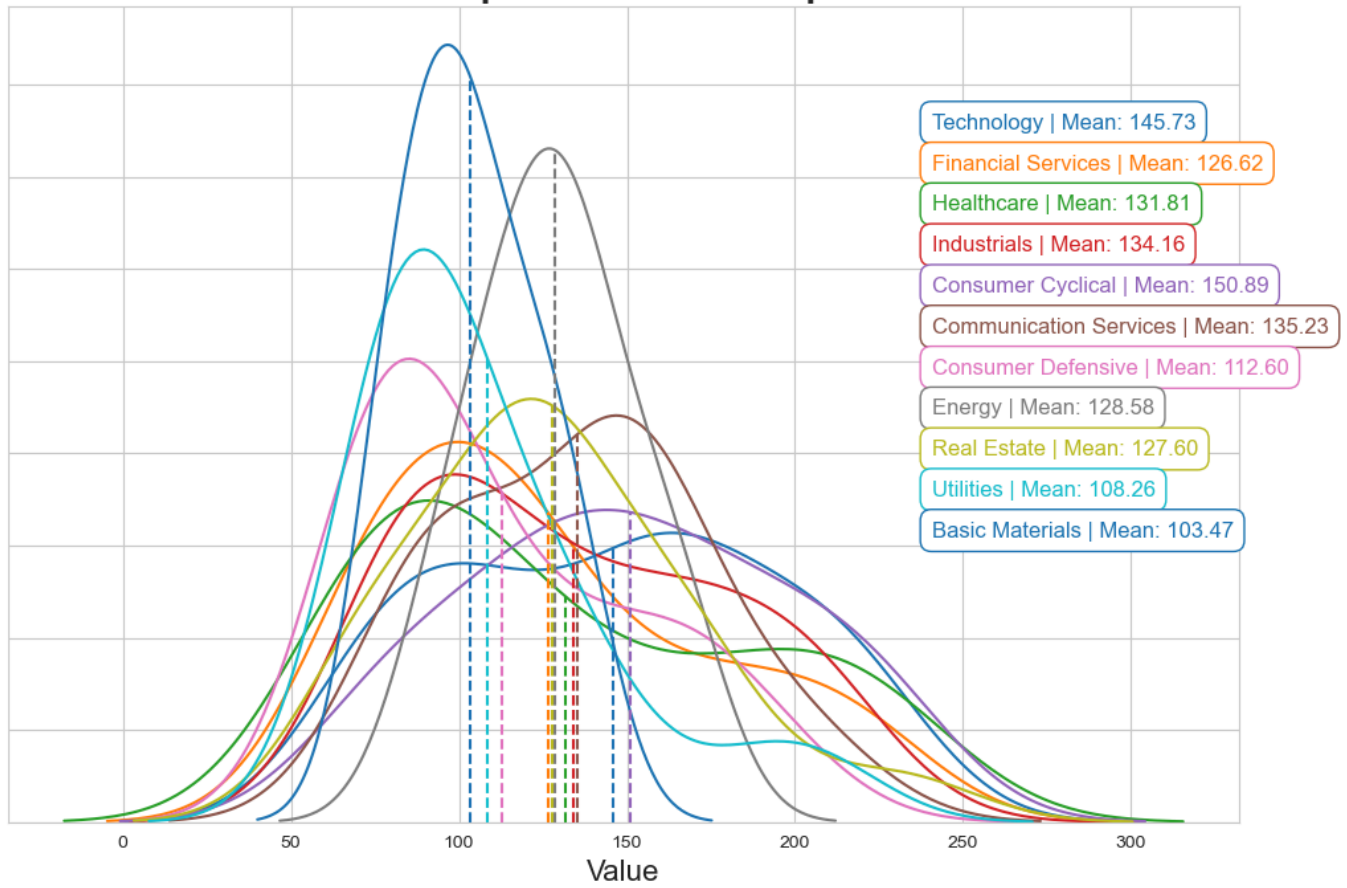
eleven_way_ANOVA(
    H1_T, H1_FS, H1_HC, H1_I, H1_CC, H1_CS, H1_CD, H1_E, H1_RE, H1_U, H1_BM,
    label1='Technology', label2='Financial Services', label3='Healthcare',
    label4='Industrials', label5='Consumer Cyclical', label6='Communication Services',
    label7='Consumer Defensive', label8='Energy', label9='Real Estate',
    label10='Utilities', label11='Basic Materials',
    save=False
)
```





Reject the Null Hypothesis

### Comparison of 11 Groups



### ✓ Hypothesis 2.1: High Trailing P/E Ratio vs Low Trailing P/E Ratio

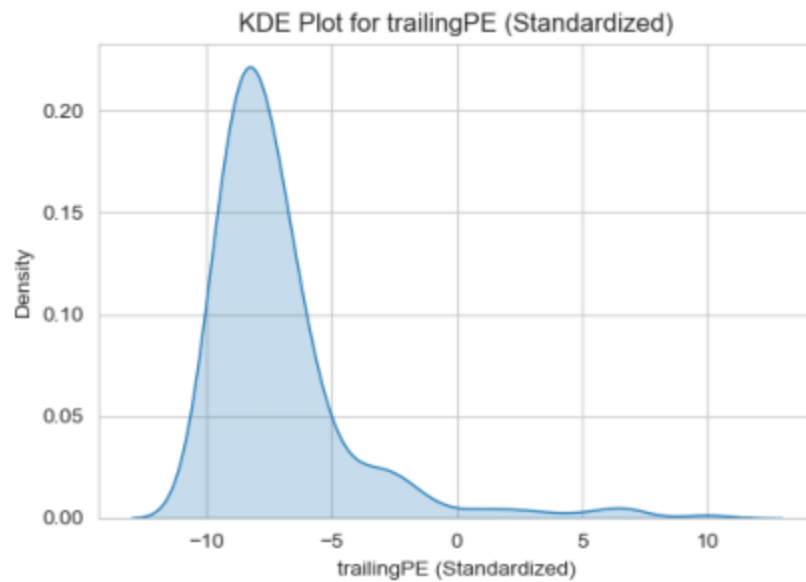
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Trailing P/E Ratio is equal to companies with a Low Trailing P/E Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Trailing P/E Ratio is NOT equal to companies with a Low Trailing P/E Ratio.

```
show_plot('trailingPE')
```

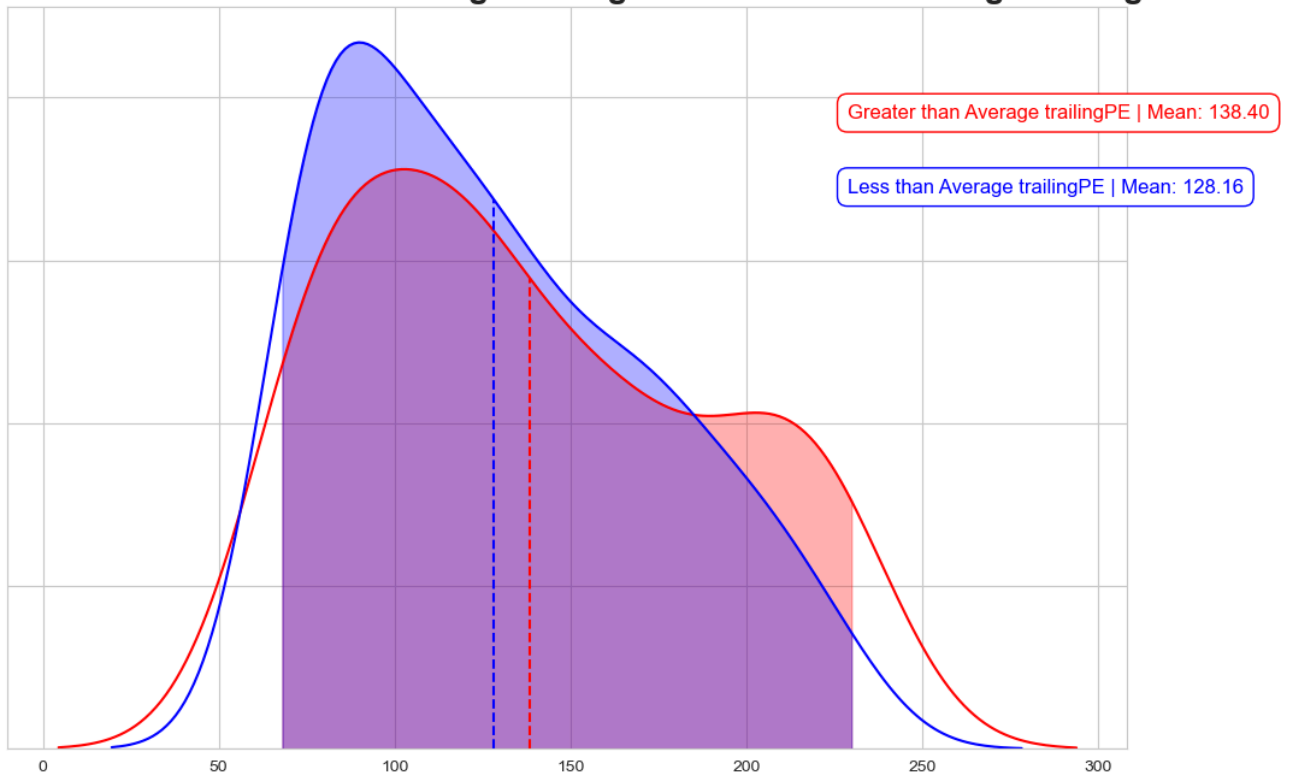


```
two_sample_kdeplot(metric='trailingPE')
```

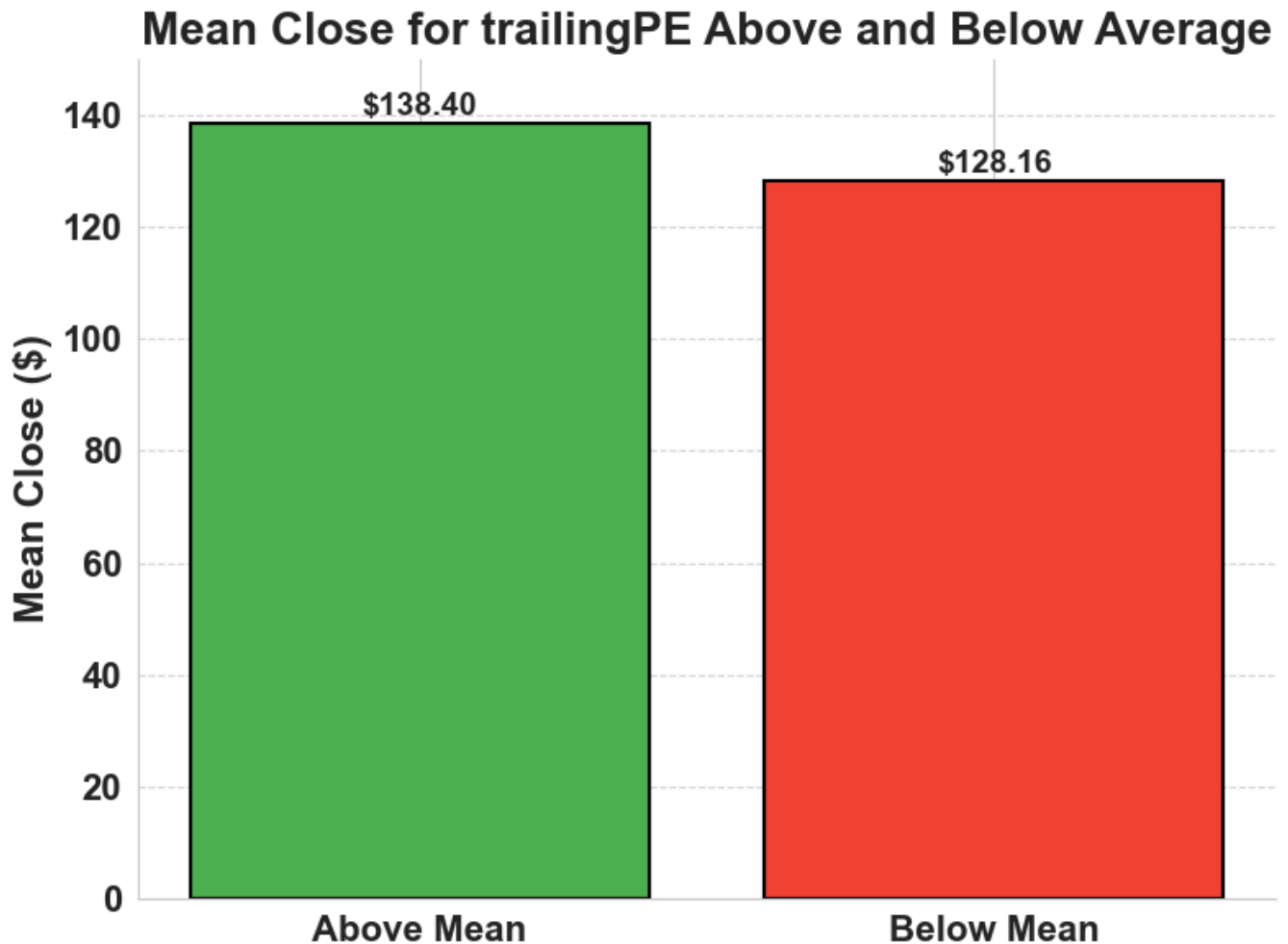


Fail to reject the null hypothesis

### Test Between Greater than Average trailingPE vs Less than Average trailingPE



```
descriptive_visual(metric='trailingPE')
```



## ✓ Hypothesis 2.2: High Forward P/E Ratio vs Low Forward P/E Ratio

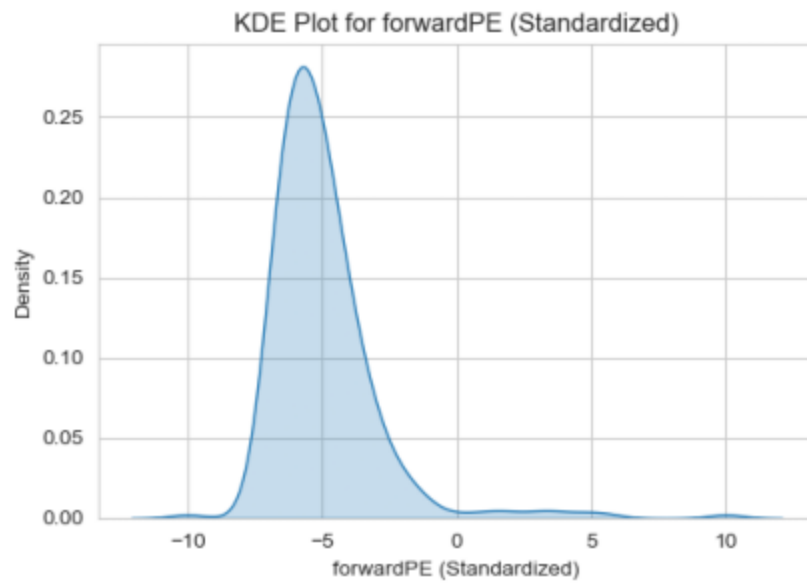
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Forward P/E Ratio is equal to companies with a Low Forward P/E Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Forward P/E Ratio is NOT equal to companies with a Low Forward P/E Ratio.

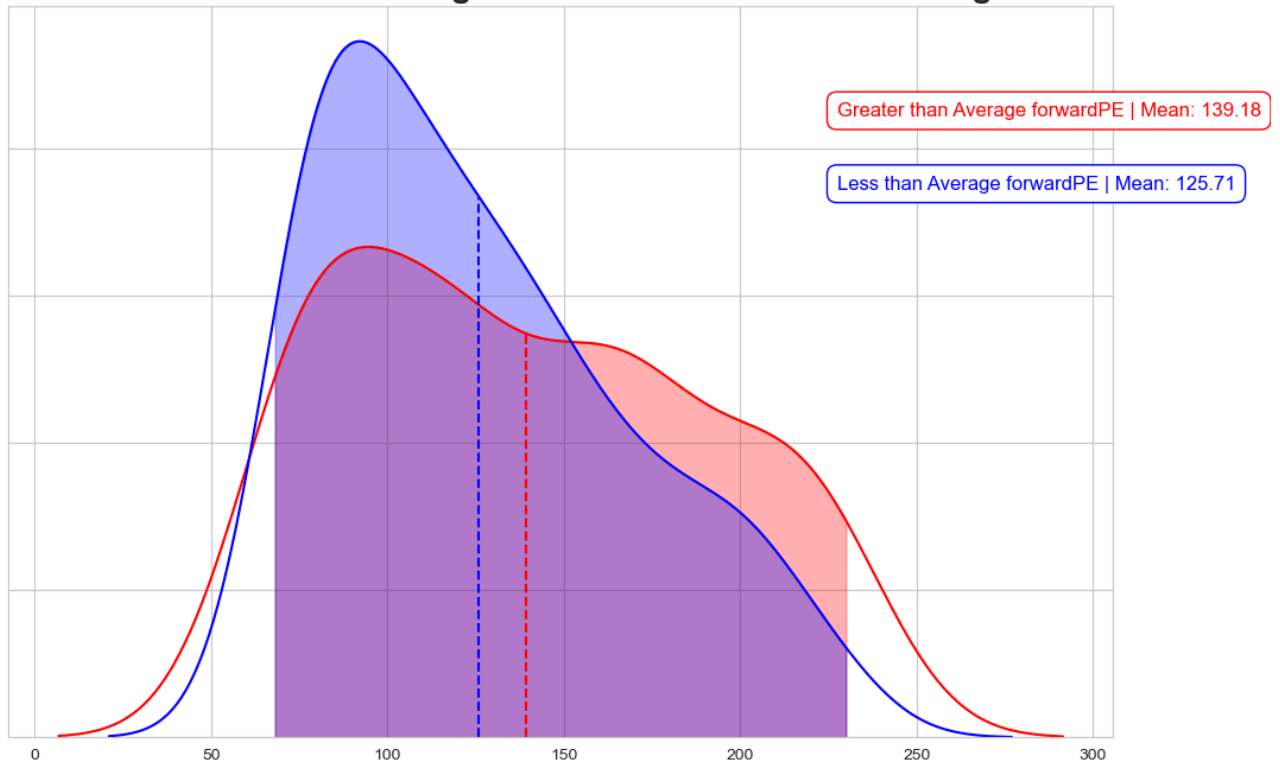
```
show_plot('forwardPE')
```



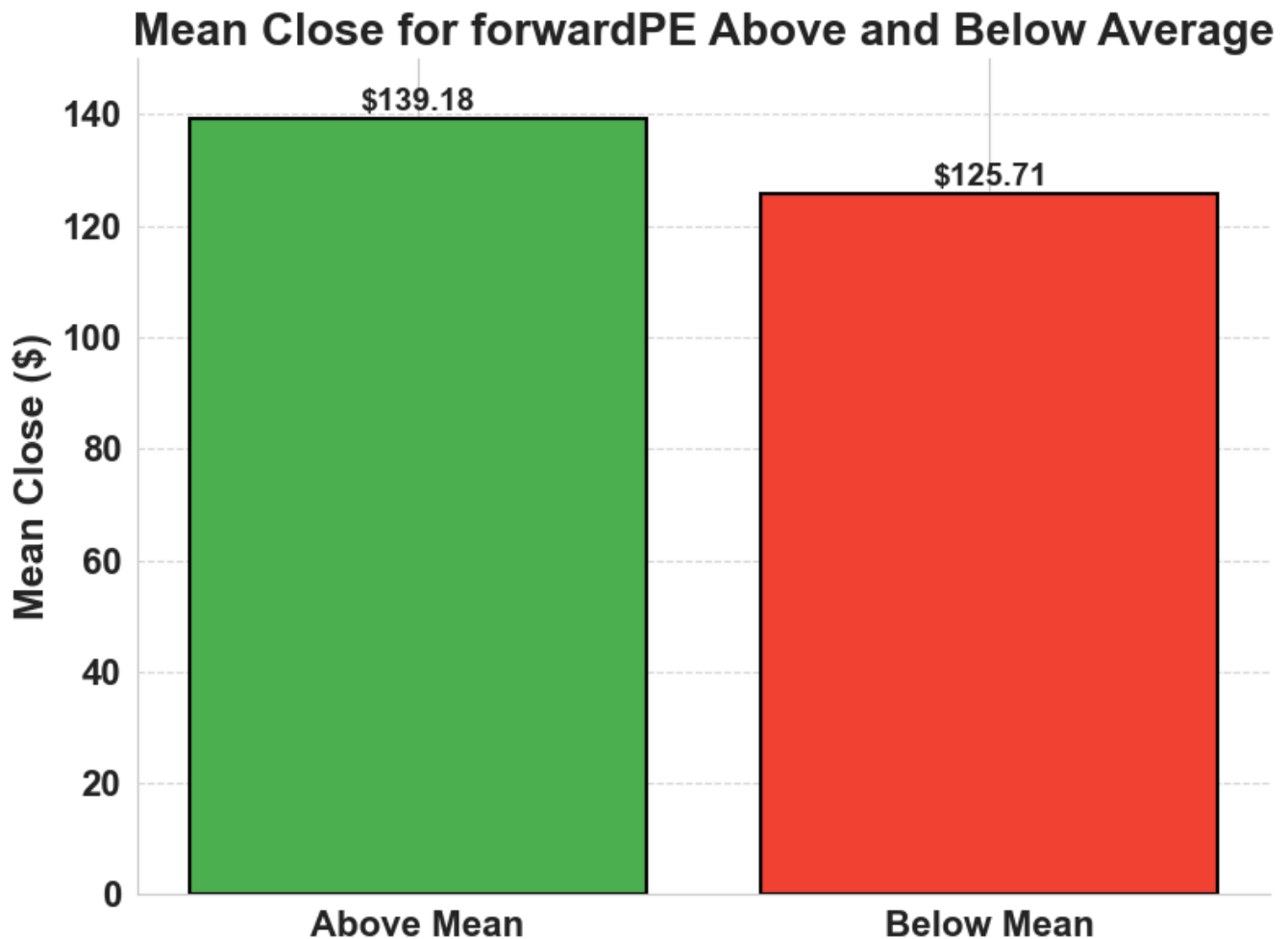
```
two_sample_kdeplot(metric='forwardPE')
```



Reject the Null Hypothesis

**Test Between Greater than Average forwardPE vs Less than Average forwardPE**

```
descriptive_visual(metric='forwardPE', save=True)
```



### ✓ Hypothesis 3.1: High PEG Ratio vs Low PEG Ratio

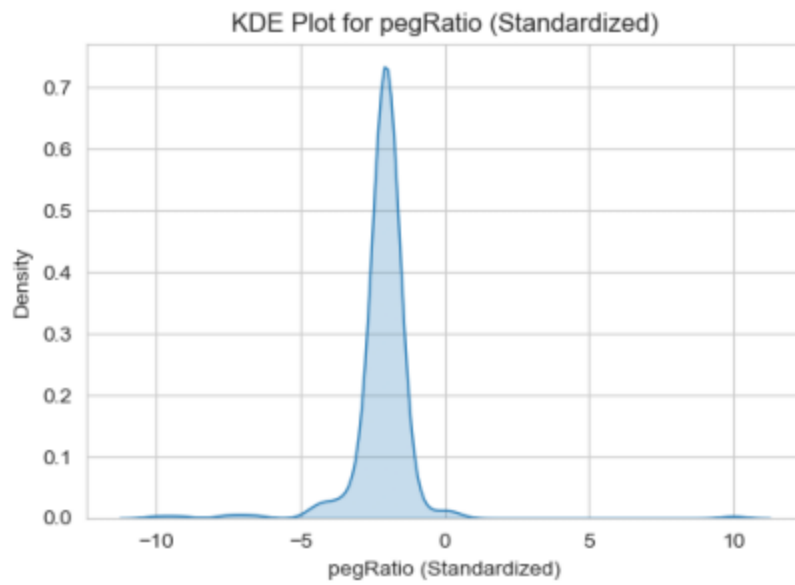
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High PEG Ratio is equal to companies with a Low PEG Ratio.

The alternate hypothesis states that the mean closing price for companies with a High PEG Ratio is NOT equal to companies with a Low PEG Ratio.

```
show_plot('pegRatio')
```



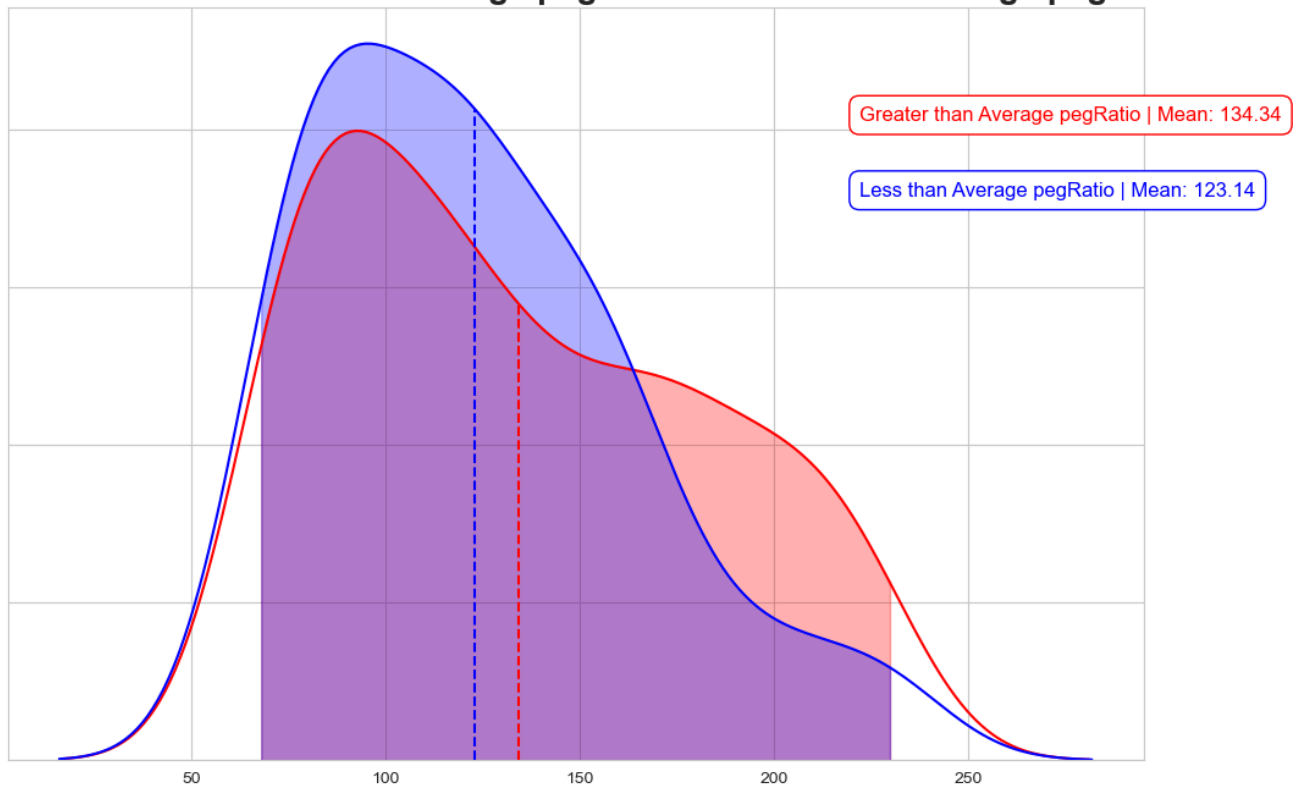
```
two_sample_kdeplot(metric='pegRatio')
```



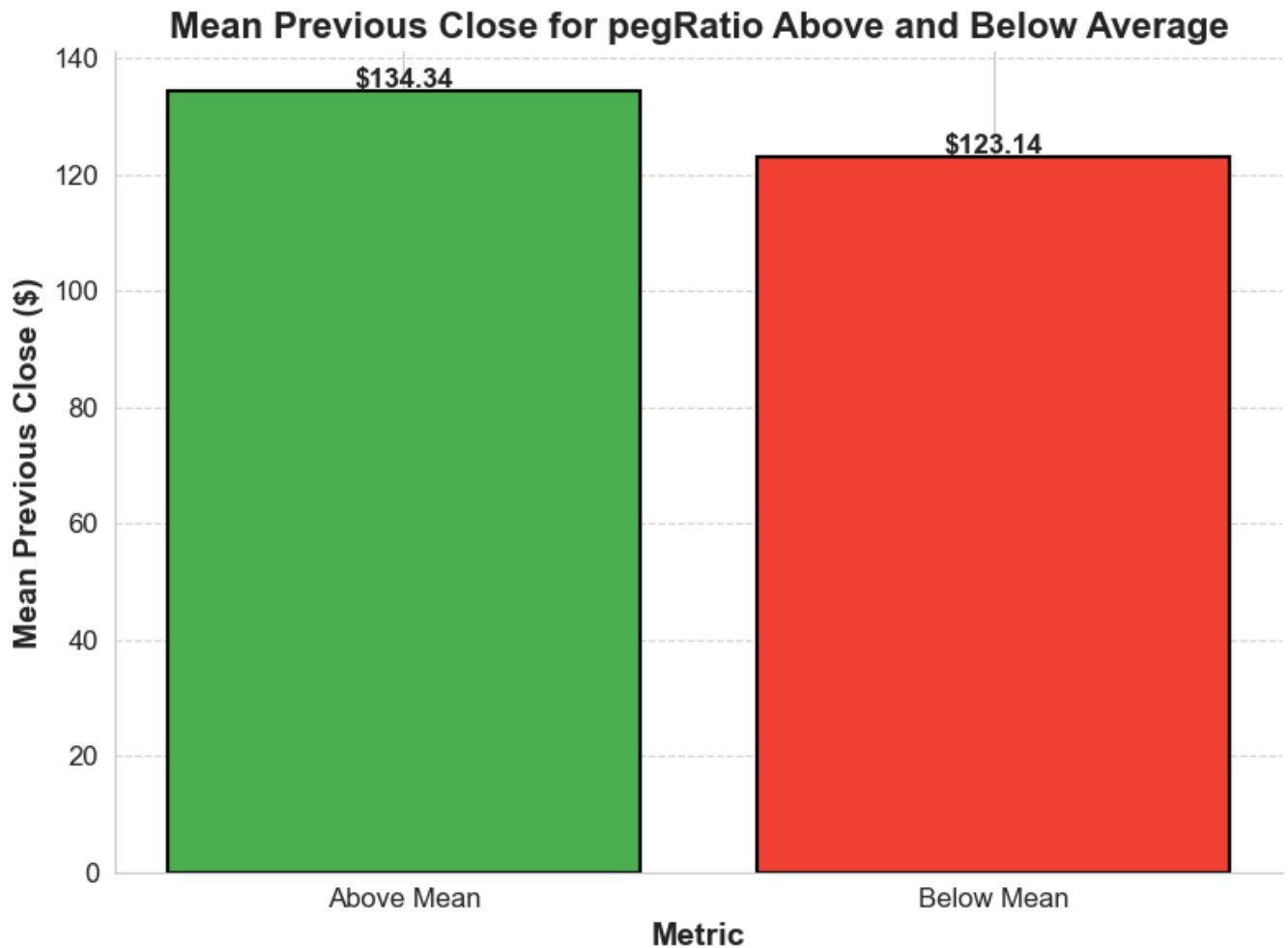


Fail to reject the null hypothesis

### Test Between Greater than Average pegRatio vs Less than Average pegRatio



```
descriptive_visual(metric='pegRatio')
```



### ✓ Hypothesis 3.2: High Trailing PEG Ratio vs Low Trailing PEG Ratio

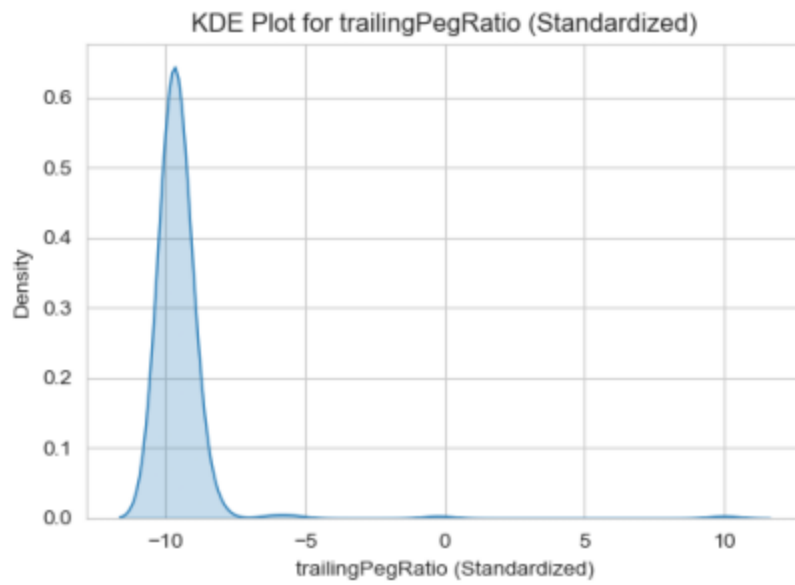
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Trailing PEG Ratio is equal to companies with a Low Trailing PEG Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Trailing PEG Ratio is NOT equal to companies with a Low Trailing PEG Ratio.

```
show_plot('trailingPegRatio')
```

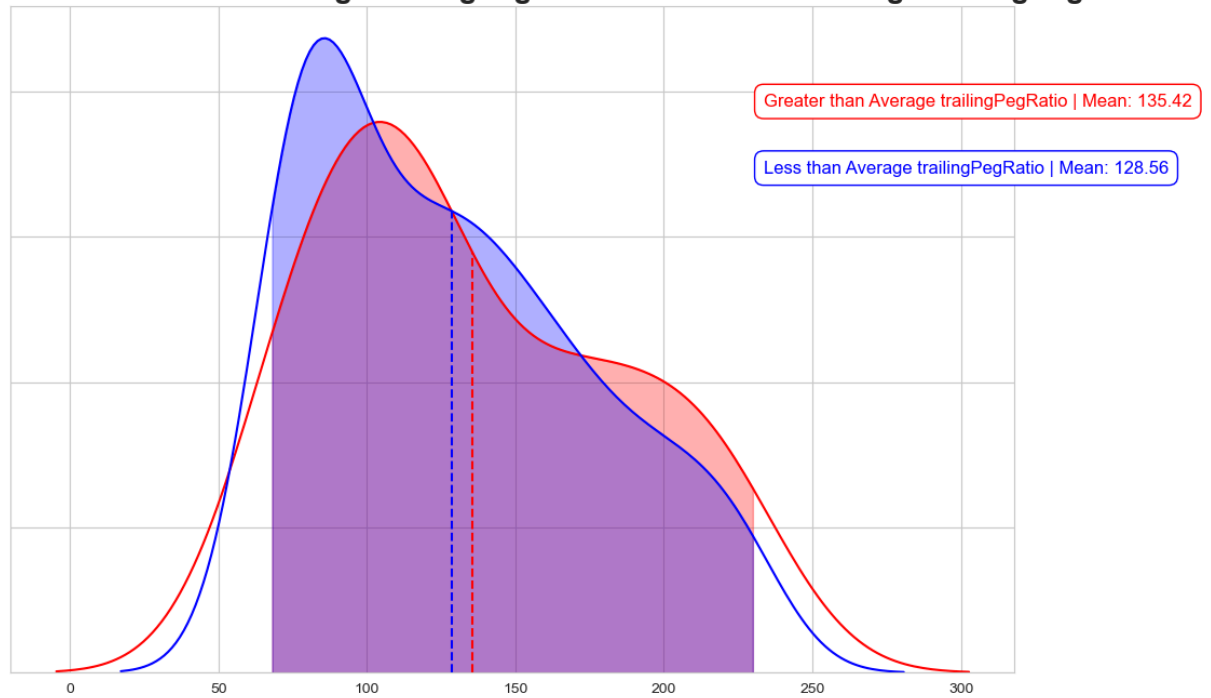


```
two_sample_kdeplot(metric='trailingPegRatio')
```

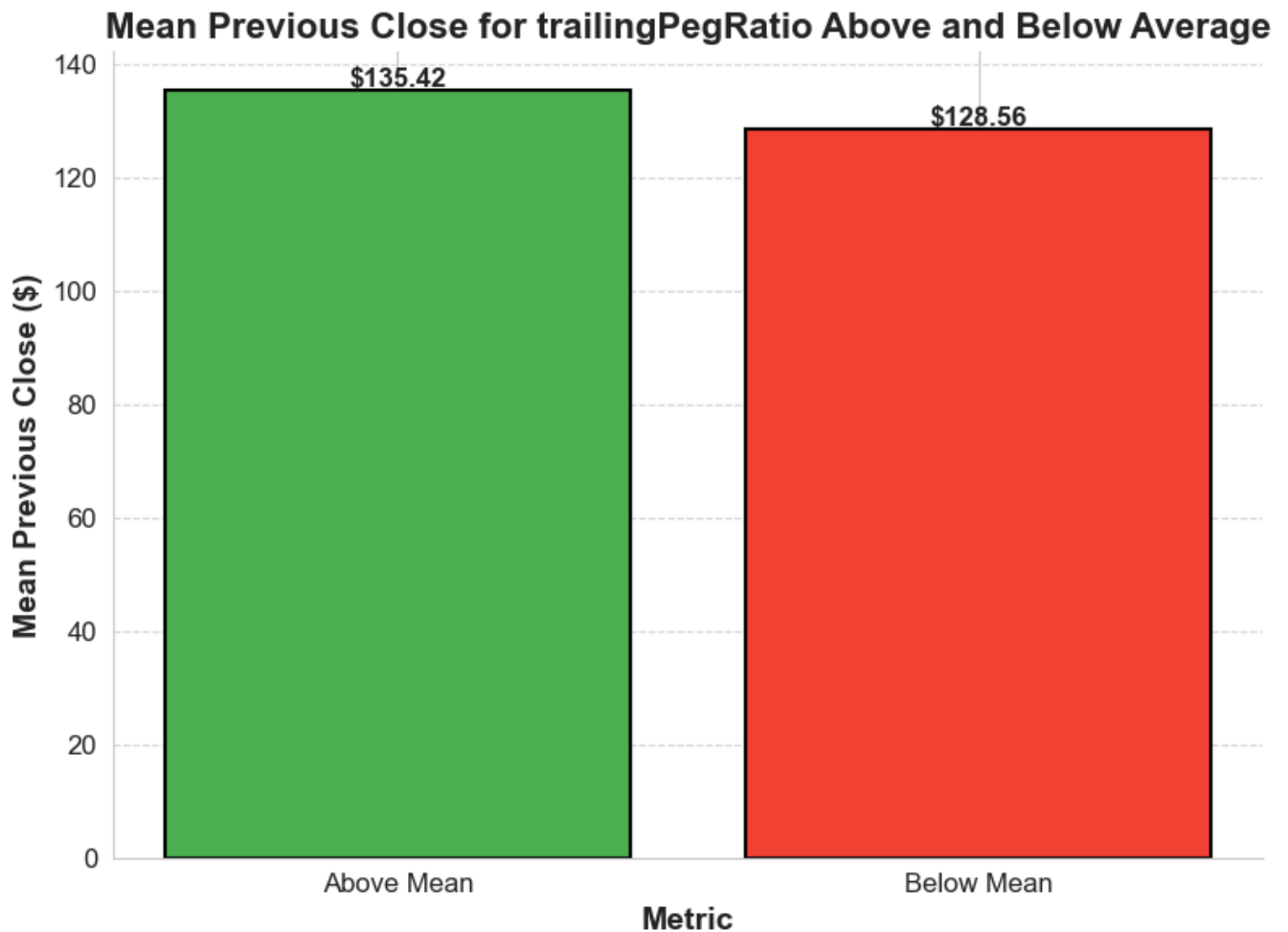


Fail to reject the null hypothesis

### Test Between Greater than Average trailingPegRatio vs Less than Average trailingPegRatio



```
descriptive_visual(metric='trailingPegRatio')
```



## ✓ Hypothesis 4.1: High Price-To-Book Ratio vs Low Price-To-Book Ratio

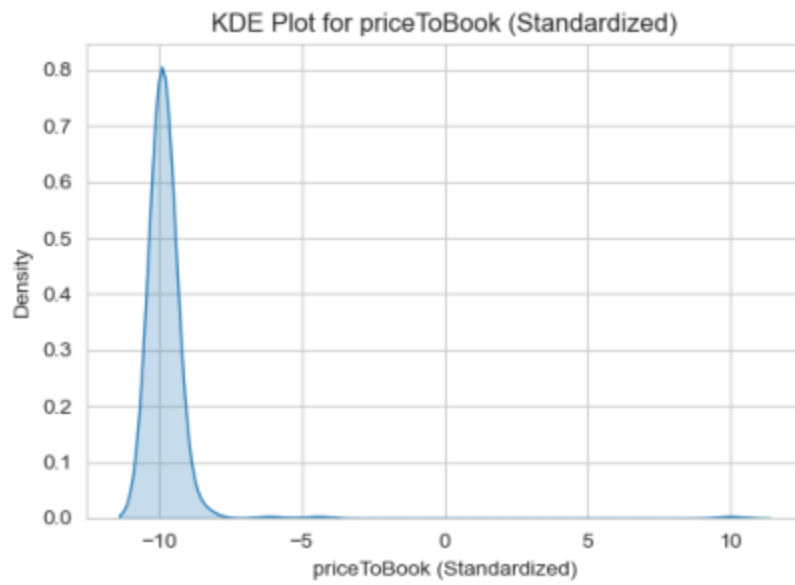
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Price-To-Book Ratio is equal to companies with a Low Price-To-Book Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Price-To-Book Ratio is NOT equal to companies with a Low Price-To-Book Ratio.

```
show_plot('priceToBook')
```

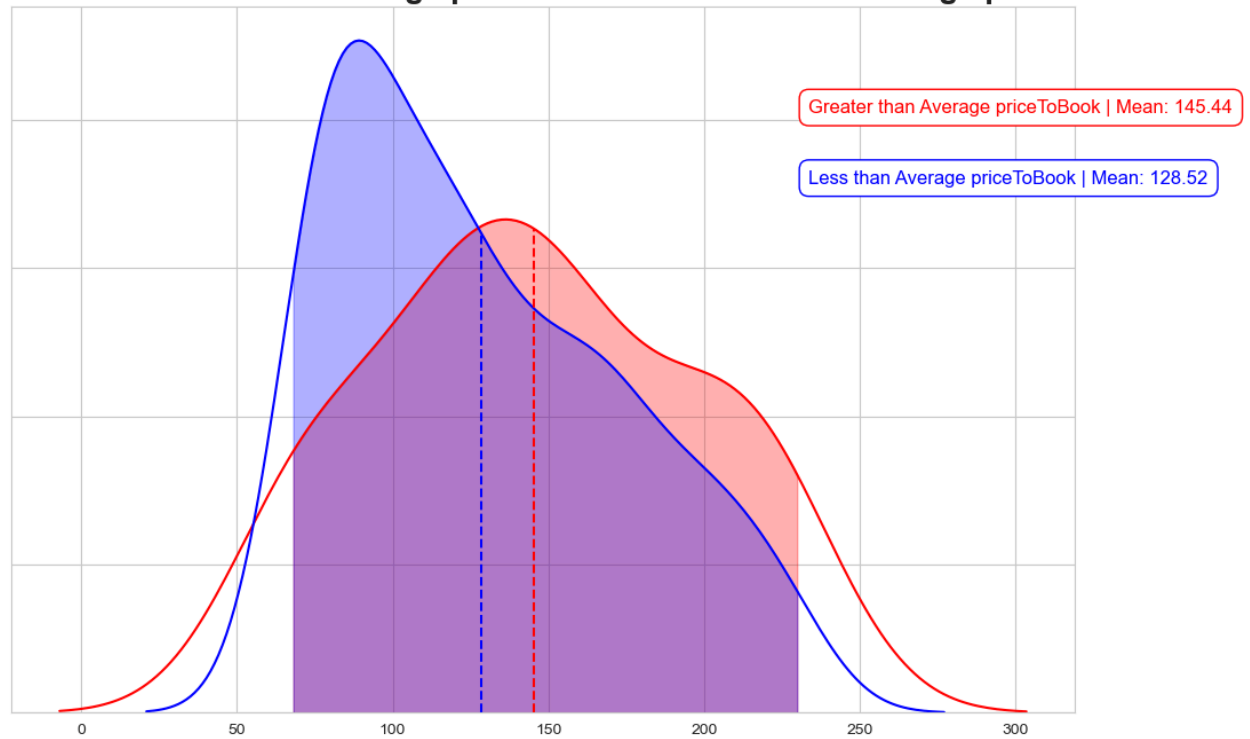


```
two_sample_kdeplot(metric='priceToBook')
```



Fail to reject the null hypothesis

### Test Between Greater than Average priceToBook vs Less than Average priceToBook



```
descriptive_visual(metric='priceToBook')
```



## ✓ Hypothesis 4.2: High Price-To-Sales Ratio vs Low Price-To-Sales Ratio

Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

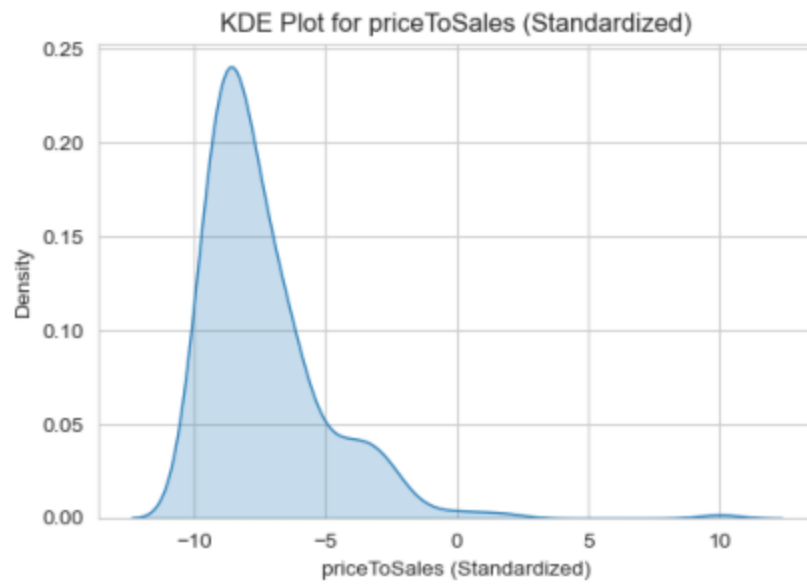
Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Price-To-Sales Ratio is equal to companies with a Low Price-To-Sales Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Price-To-Sales Ratio is NOT equal to companies with a Low Price-To-Sales Ratio.

```
show_plot('priceToSales')
```

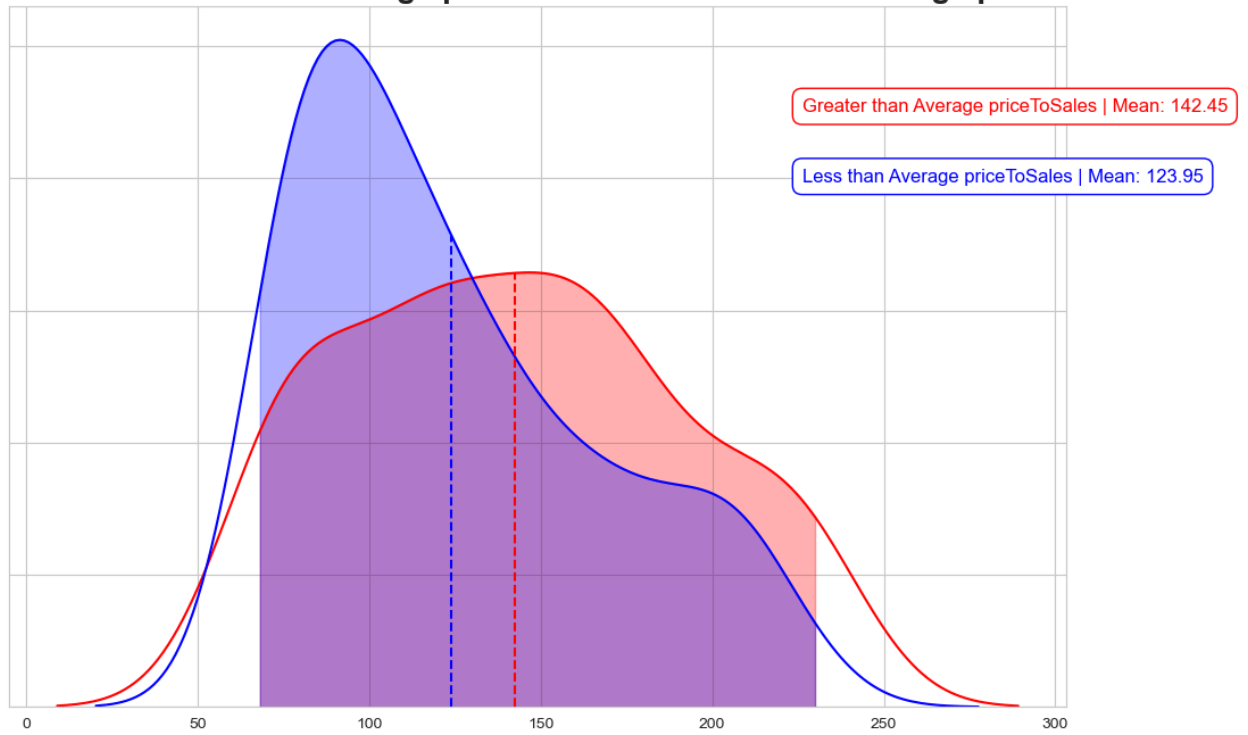




```
two_sample_kdeplot(metric='priceToSales')
```



Reject the Null Hypothesis

**Test Between Greater than Average priceToSales vs Less than Average priceToSales**

```
descriptive_visual(metric='priceToSales', save=True)
```



## ✓ Hypothesis 5: High Debt-To-Equity Ratio vs Low Debt-To-Equity Ratio

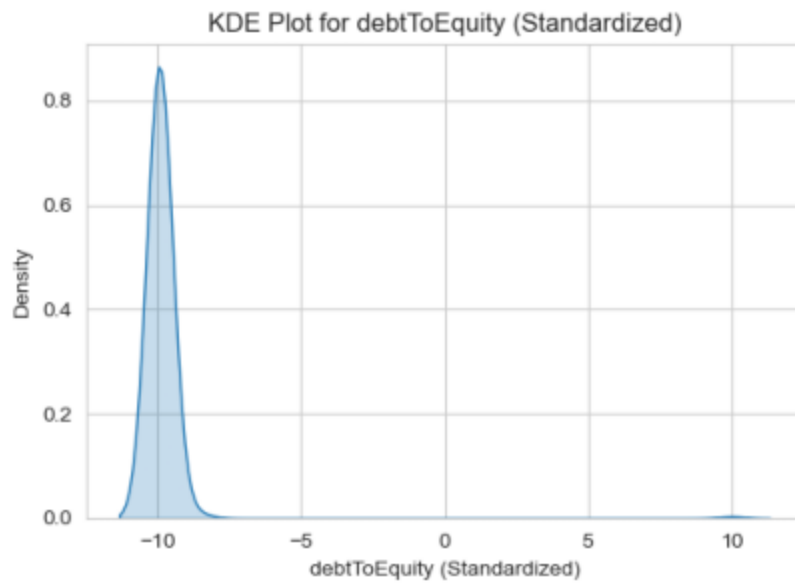
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Debt-To-Equity Ratio is equal to companies with a Low Debt-To-Equity Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Debt-To-Equity Ratio is NOT equal to companies with a Low Debt-To-Equity Ratio.

```
show_plot('debtToEquity')
```

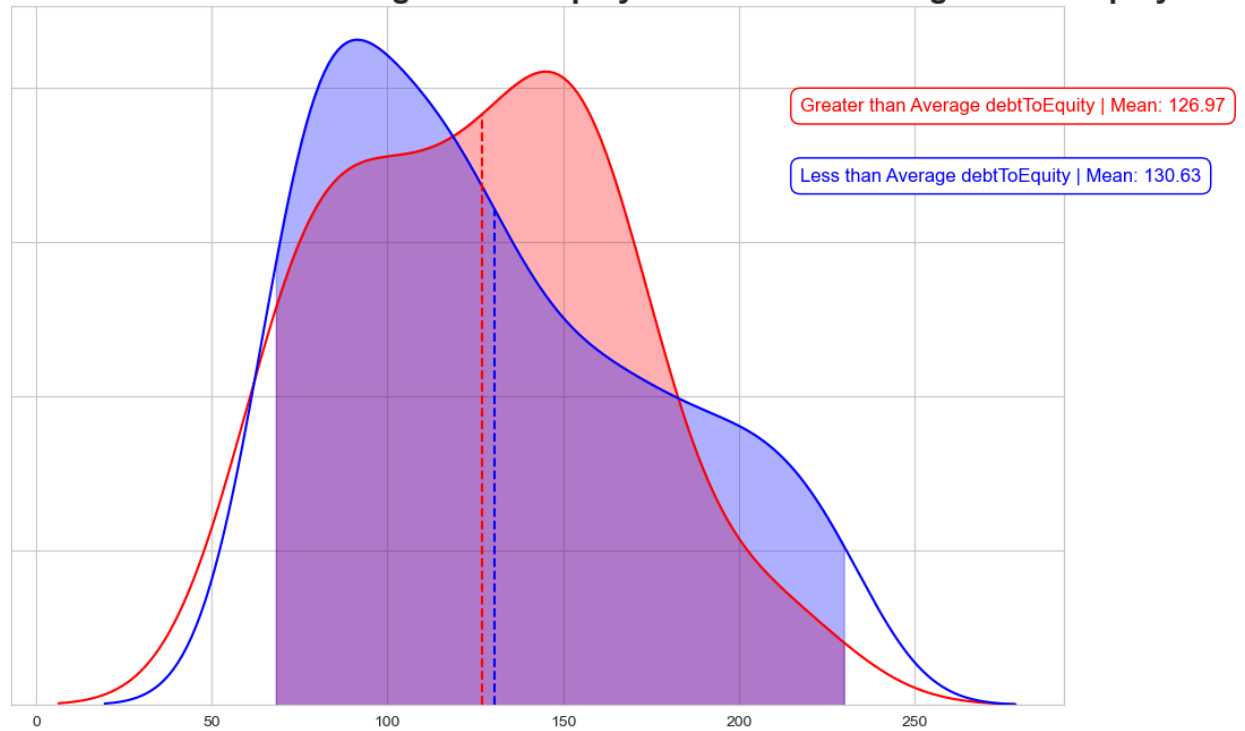


```
two_sample_kdeplot(metric='debtToEquity')
```



Fail to reject the null hypothesis

### Test Between Greater than Average debtToEquity vs Less than Average debtToEquity



```
descriptive_visual(metric='debtToEquity')
```



### ✓ Hypothesis 6.1: High Trailing EPS Ratio vs Low Trailing EPS Ratio

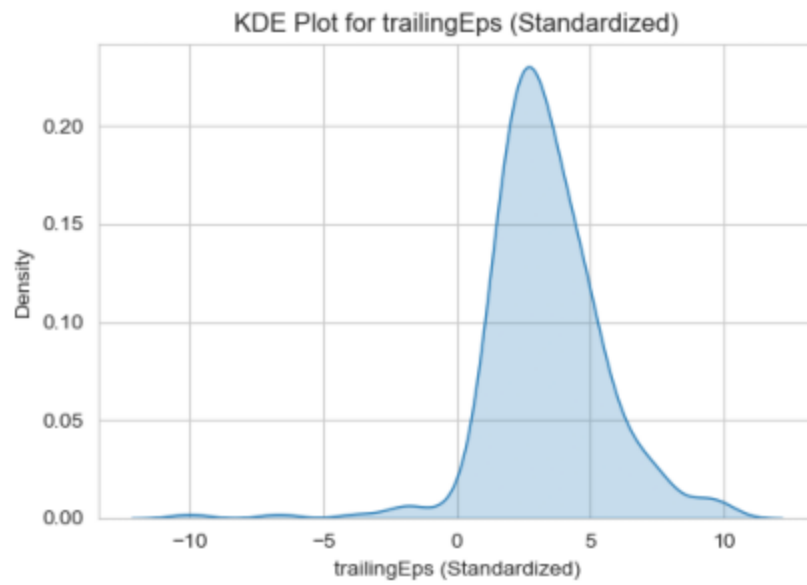
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Trailing EPS Ratio is equal to companies with a Low Trailing EPS Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Trailing EPS Ratio is NOT equal to companies with a Low Trailing EPS Ratio.

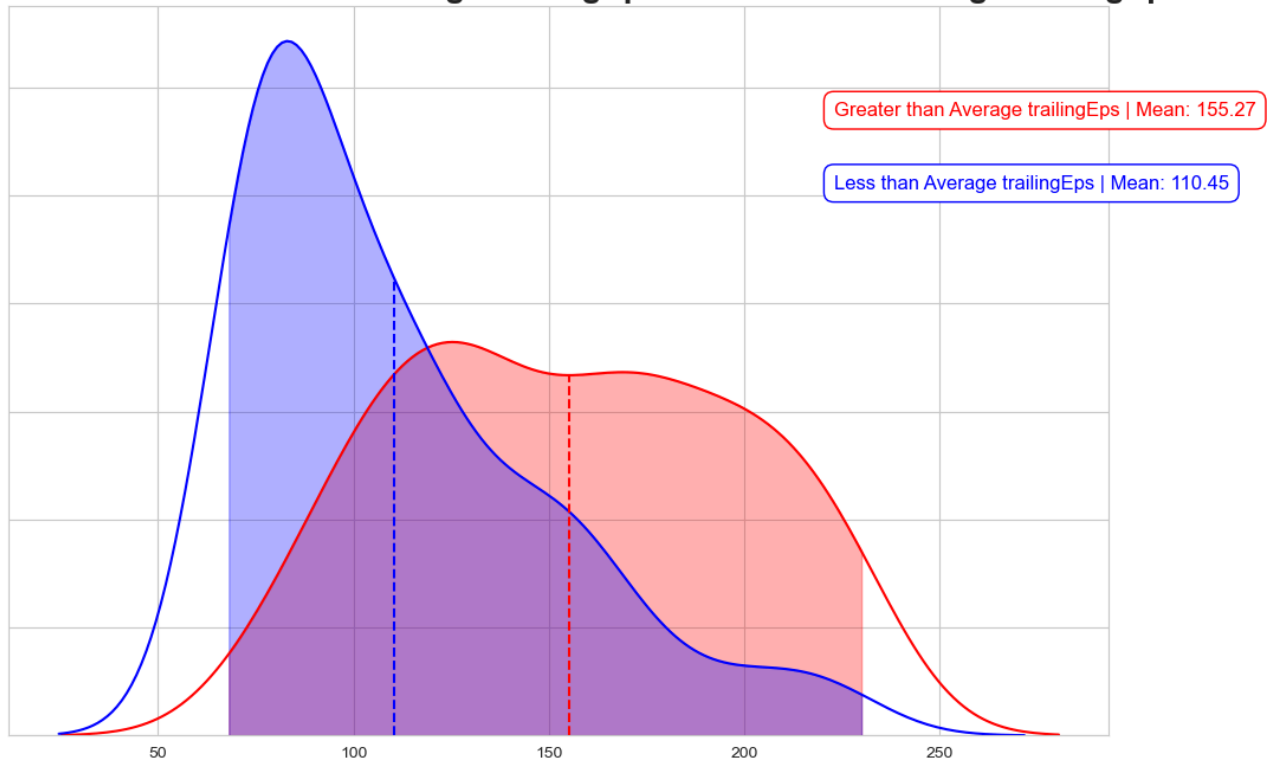
```
show_plot('trailingEps')
```



```
two_sample_kdeplot(metric='trailingEps')
```

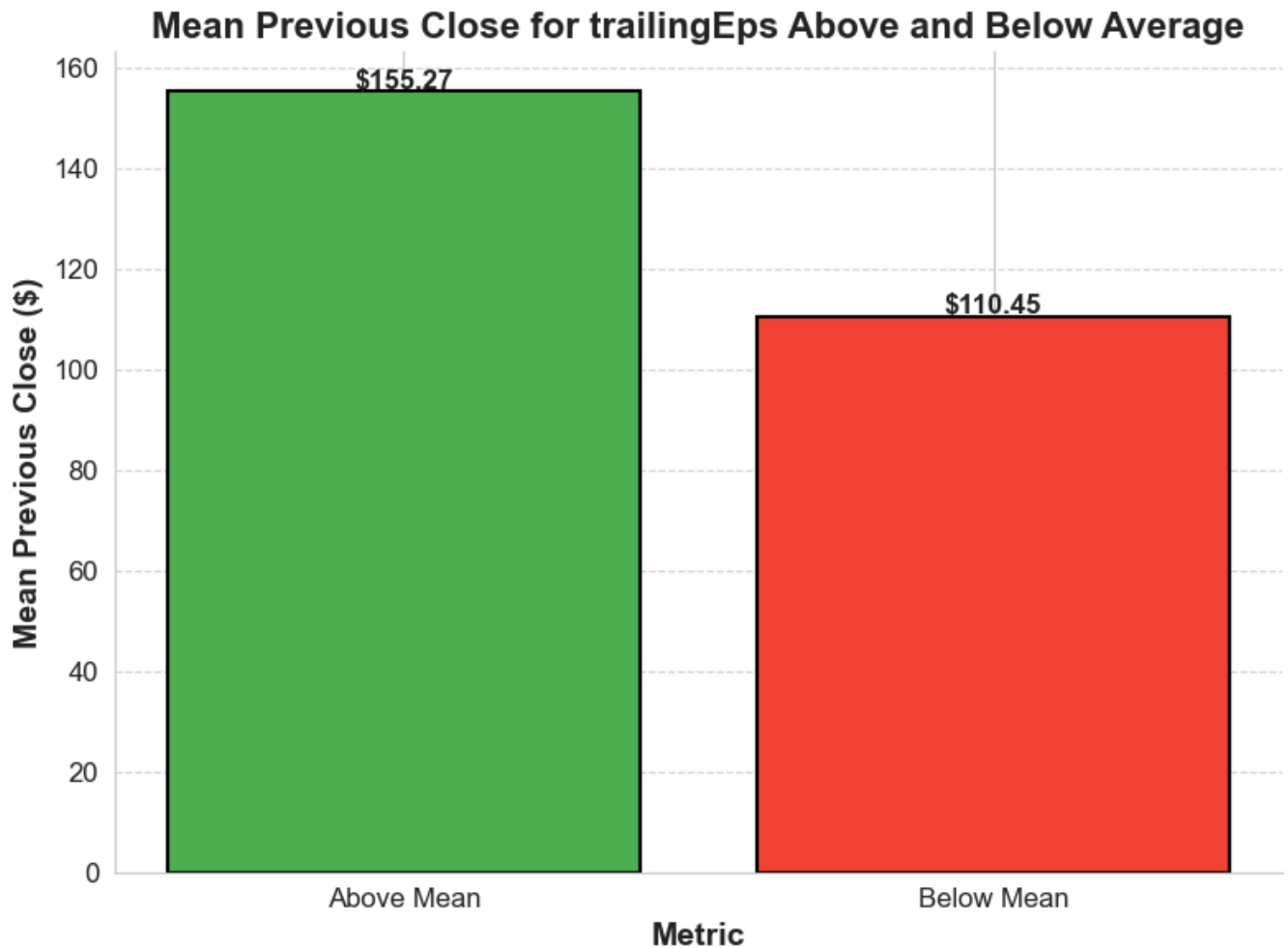


Reject the Null Hypothesis

**Test Between Greater than Average trailingEps vs Less than Average trailingEps**

```
descriptive_visual(metric='trailingEps')
```





## ✓ Hypothesis 6.2: High Forward EPS Ratio vs Low Forward EPS Ratio

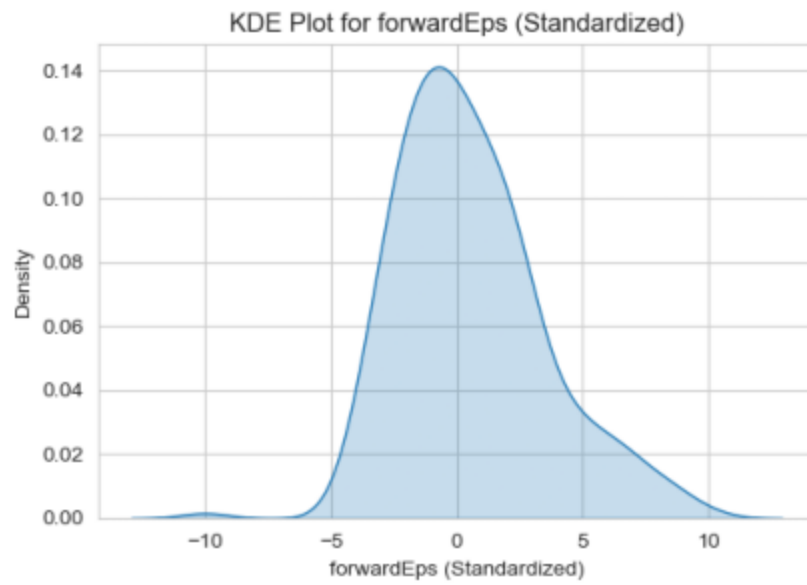
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Forward EPS Ratio is equal to companies with a Low Forward EPS Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Forward EPS Ratio is NOT equal to companies with a Low Forward EPS Ratio.

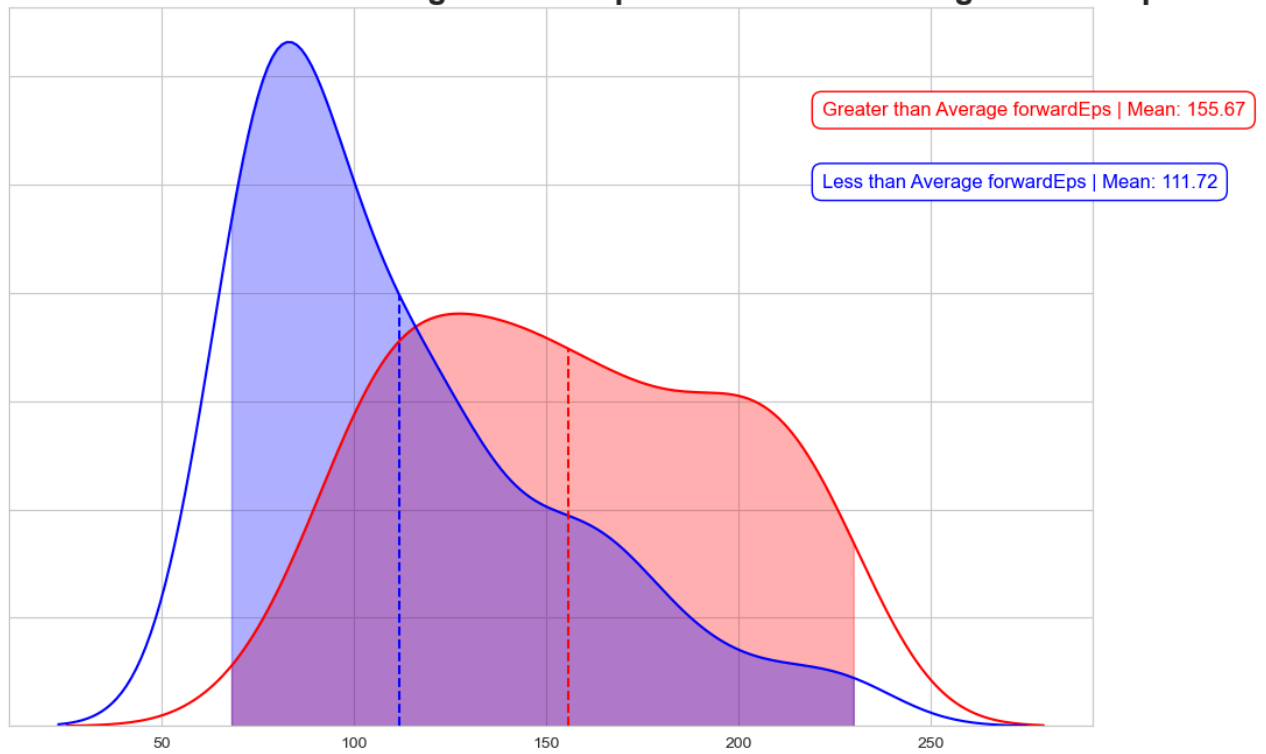
```
show_plot('forwardEps')
```



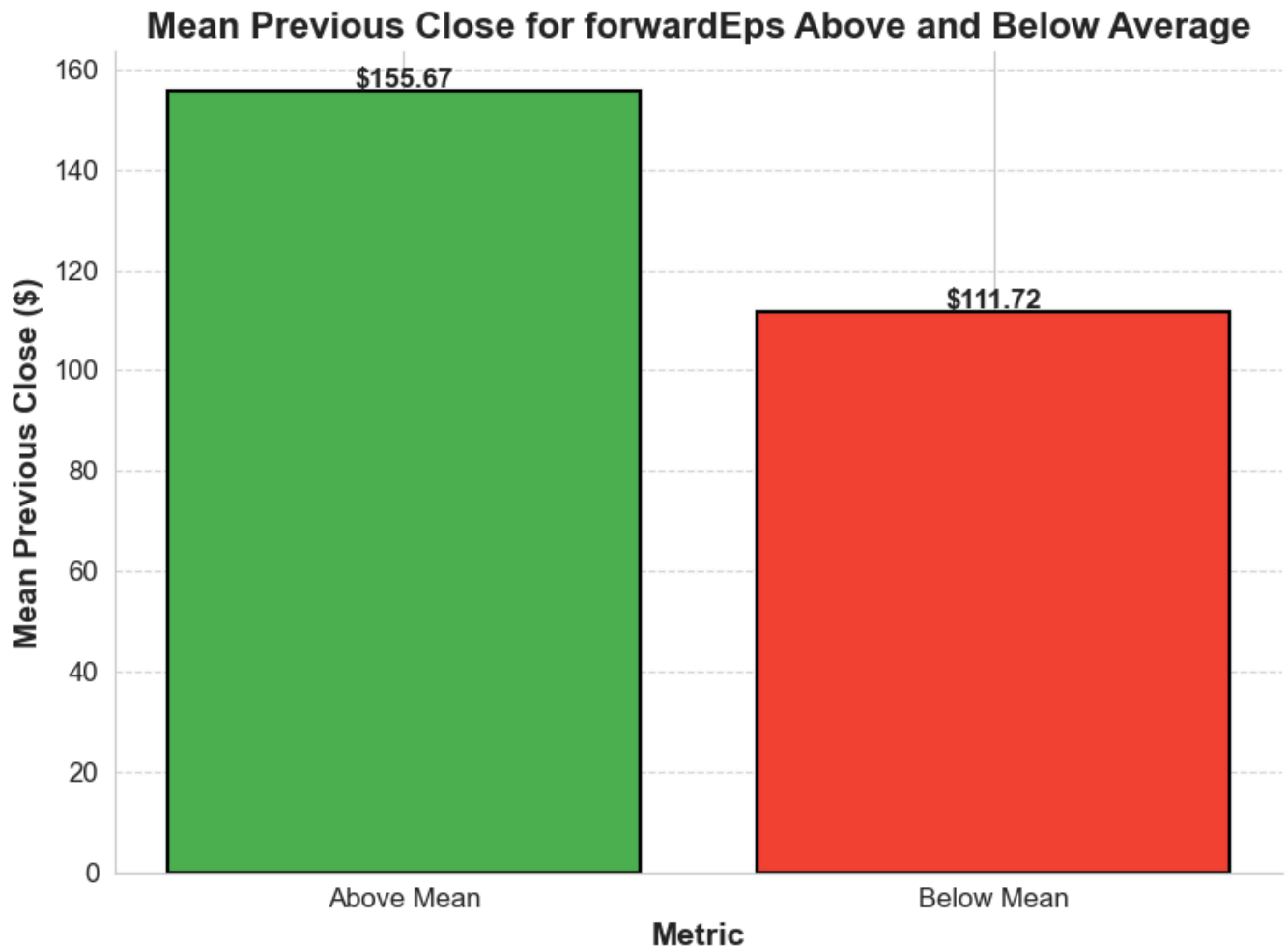
```
two_sample_kdeplot(metric='forwardEps')
```



Reject the Null Hypothesis

**Test Between Greater than Average forwardEps vs Less than Average forwardEps**

```
descriptive_visual(metric='forwardEps')
```



## ✓ Hypothesis 7: High ROE vs Low ROE

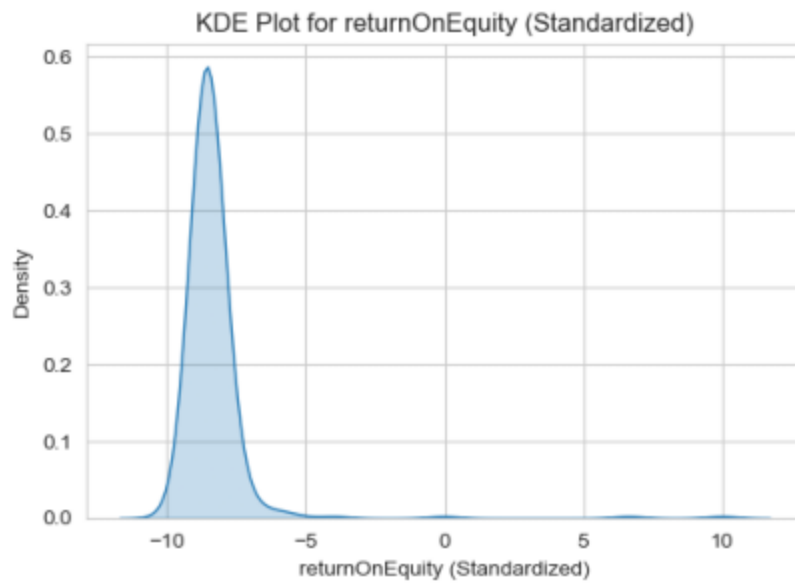
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High ROE is equal to companies with a Low ROE.

The alternate hypothesis states that the mean closing price for companies with a High ROE is NOT equal to companies with a Low ROE.

```
show_plot('returnOnEquity')
```

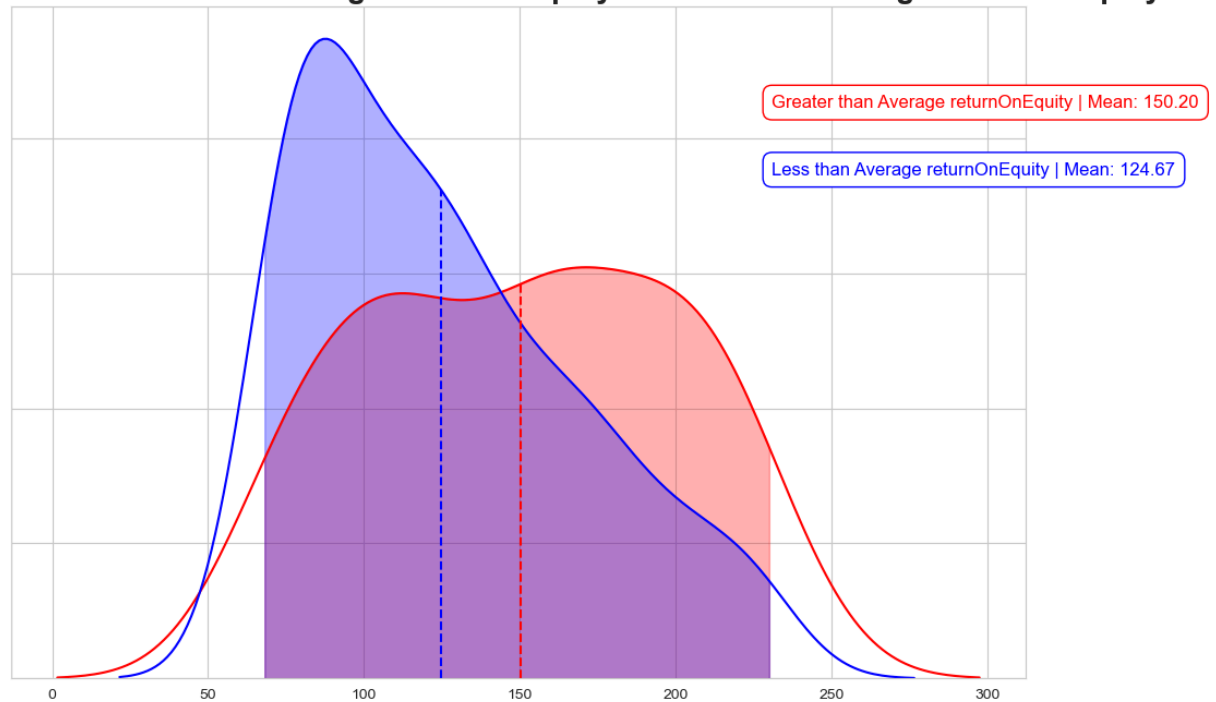


```
two_sample_kdeplot(metric='returnOnEquity')
```

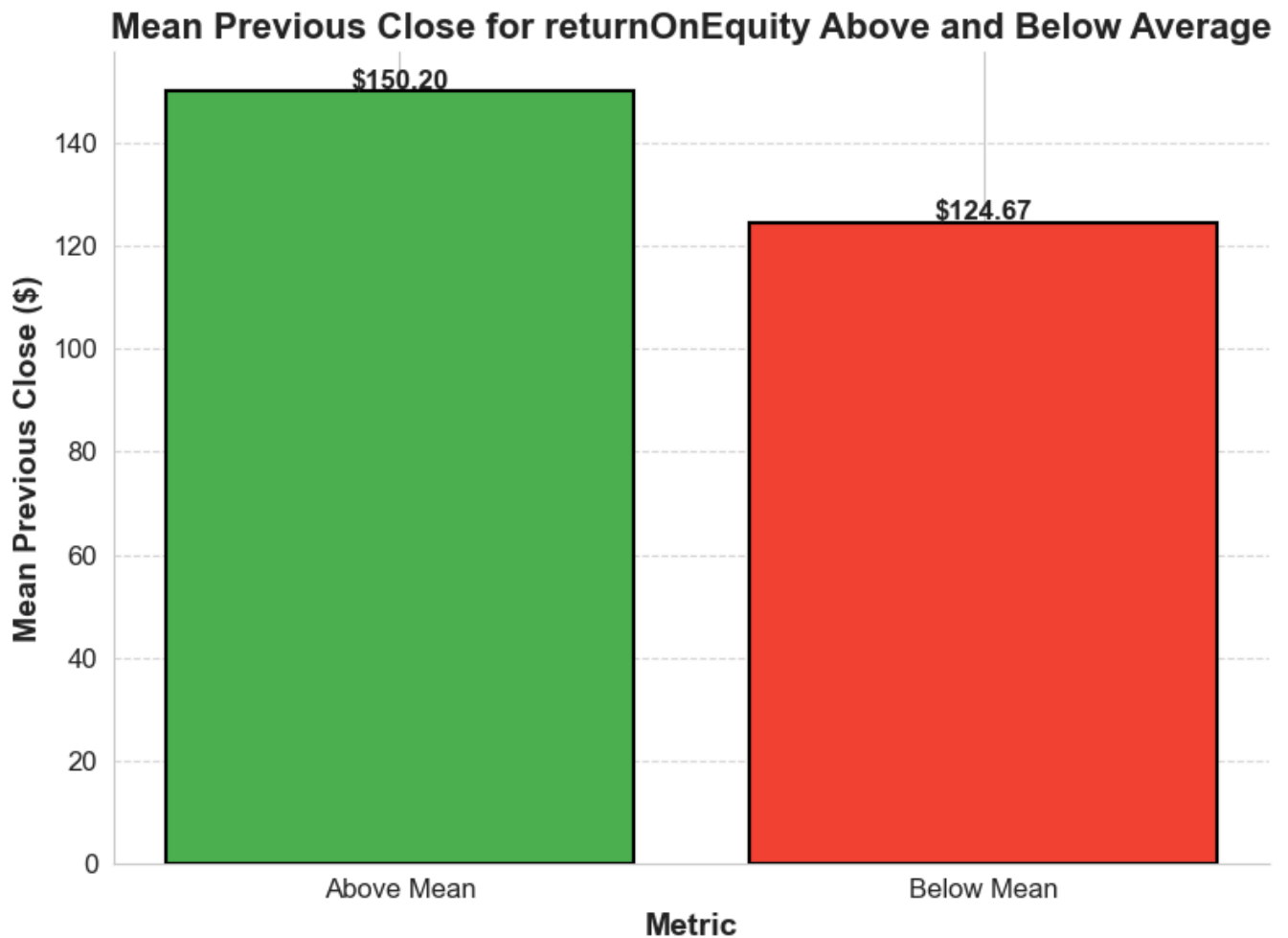


Reject the Null Hypothesis

### Test Between Greater than Average returnOnEquity vs Less than Average returnOnEquity



```
descriptive_visual(metric='returnOnEquity')
```



## ✓ Hypothesis 8.1: High Gross Margin vs Low Gross Margin

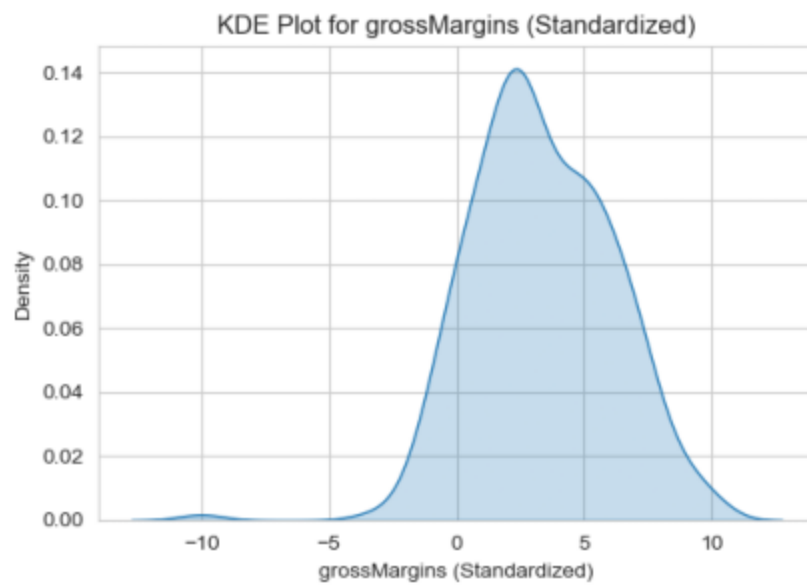
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Gross Margin is equal to companies with a Low Gross Margin.

The alternate hypothesis states that the mean closing price for companies with a High Gross Margin is NOT equal to companies with a Low Gross Margin.

```
show_plot('grossMargins')
```



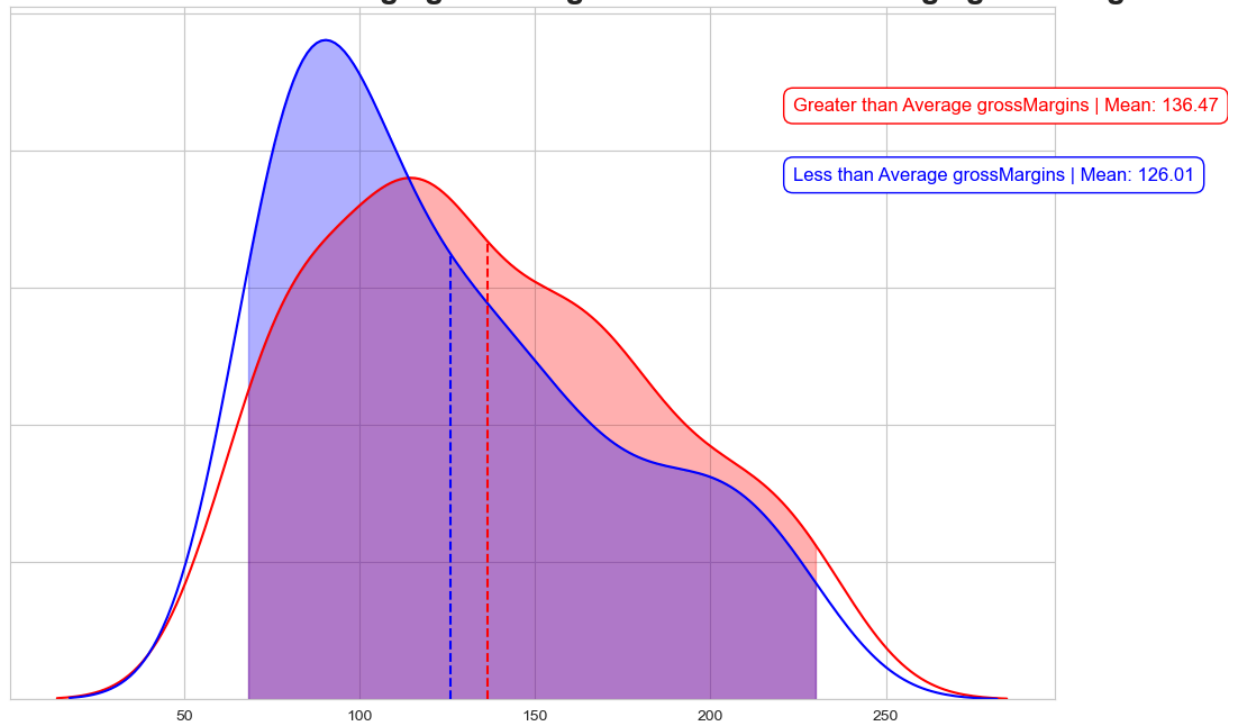
```
two_sample_kdeplot(metric='grossMargins')
```



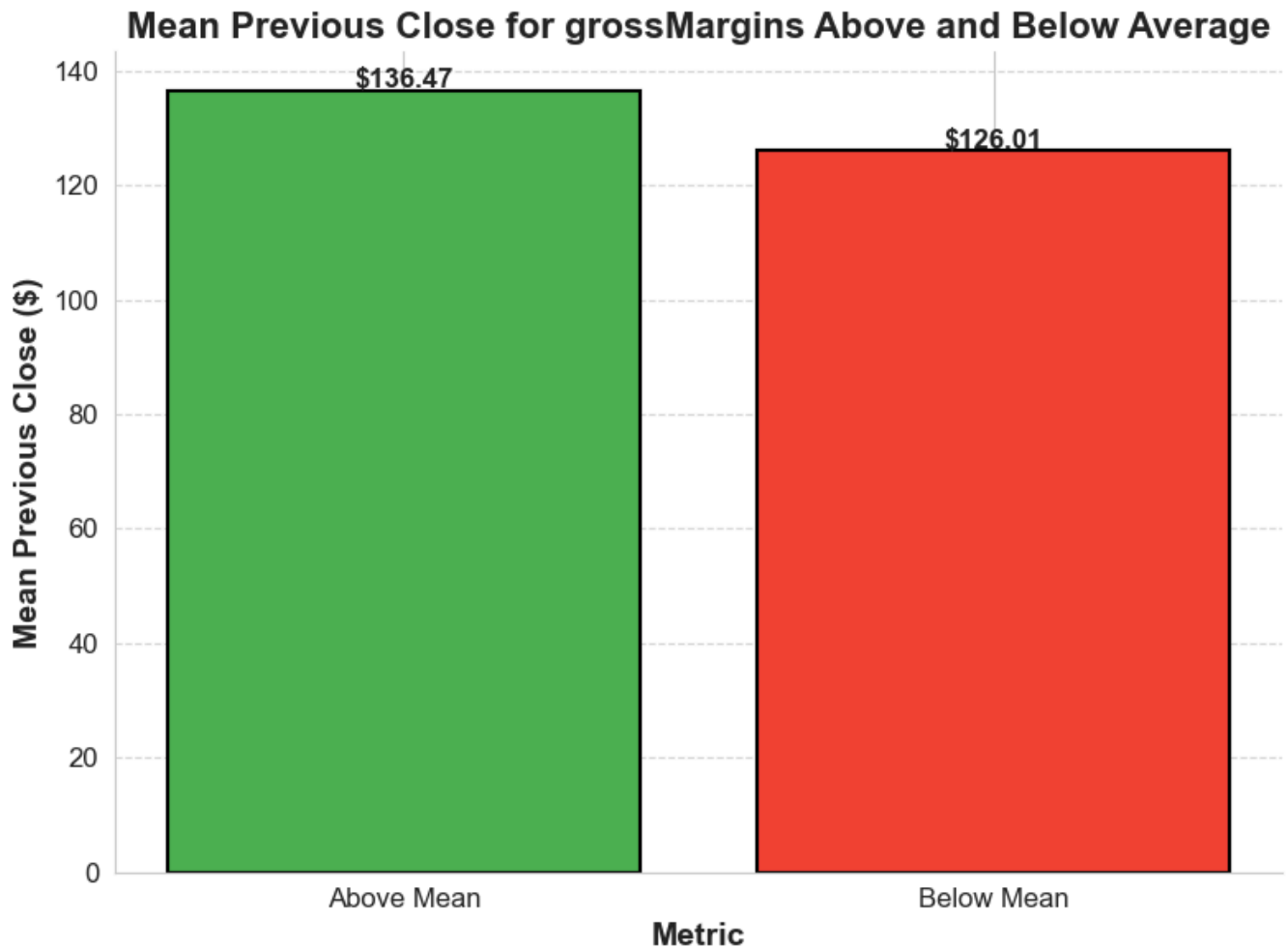


Fail to reject the null hypothesis

### Test Between Greater than Average grossMargins vs Less than Average grossMargins



```
descriptive_visual(metric='grossMargins')
```



## ✓ Hypothesis 8.2: High Operating Margin vs Low Operating Margin

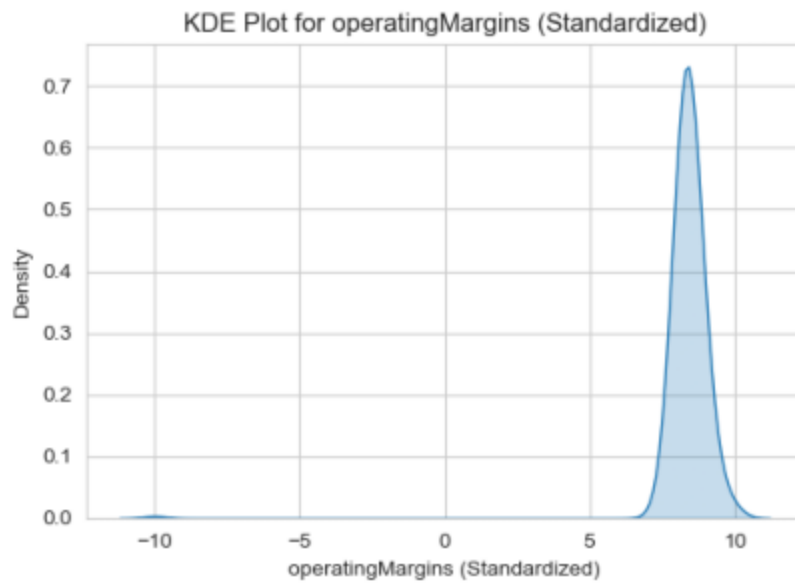
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Operating Margin is equal to companies with a Low Operating Margin.

The alternate hypothesis states that the mean closing price for companies with a High Operating Margin is NOT equal to companies with a Low Operating Margin.

```
show_plot('operatingMargins')
```

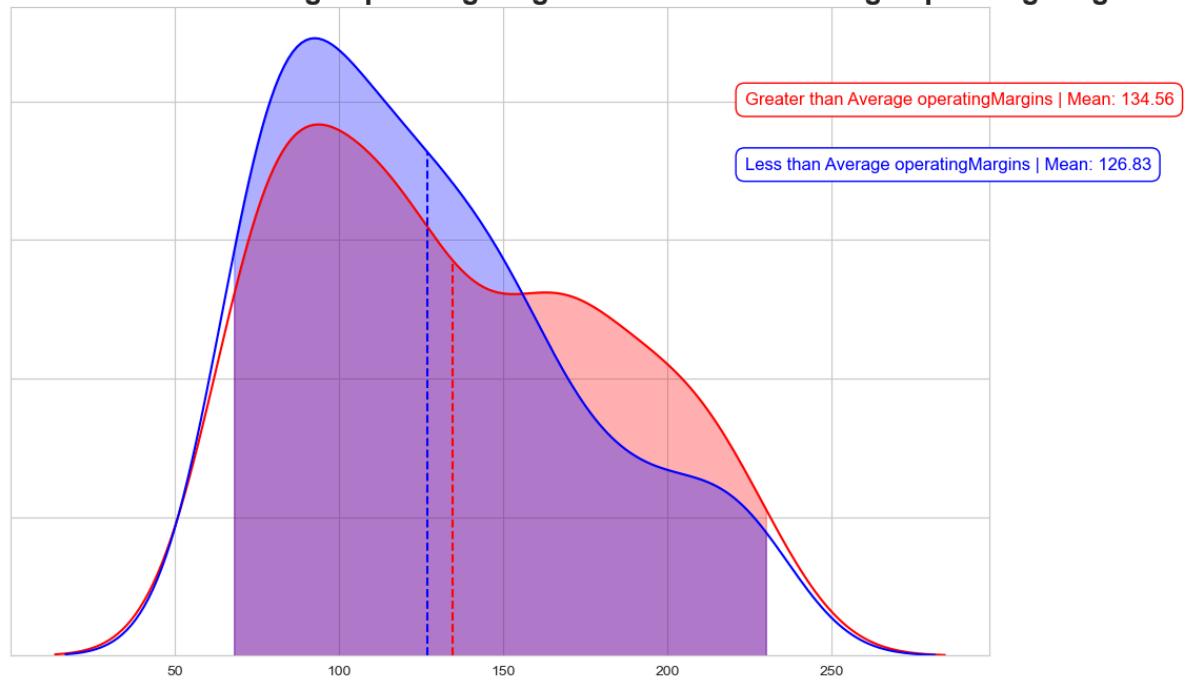


```
two_sample_kdeplot(metric='operatingMargins')
```

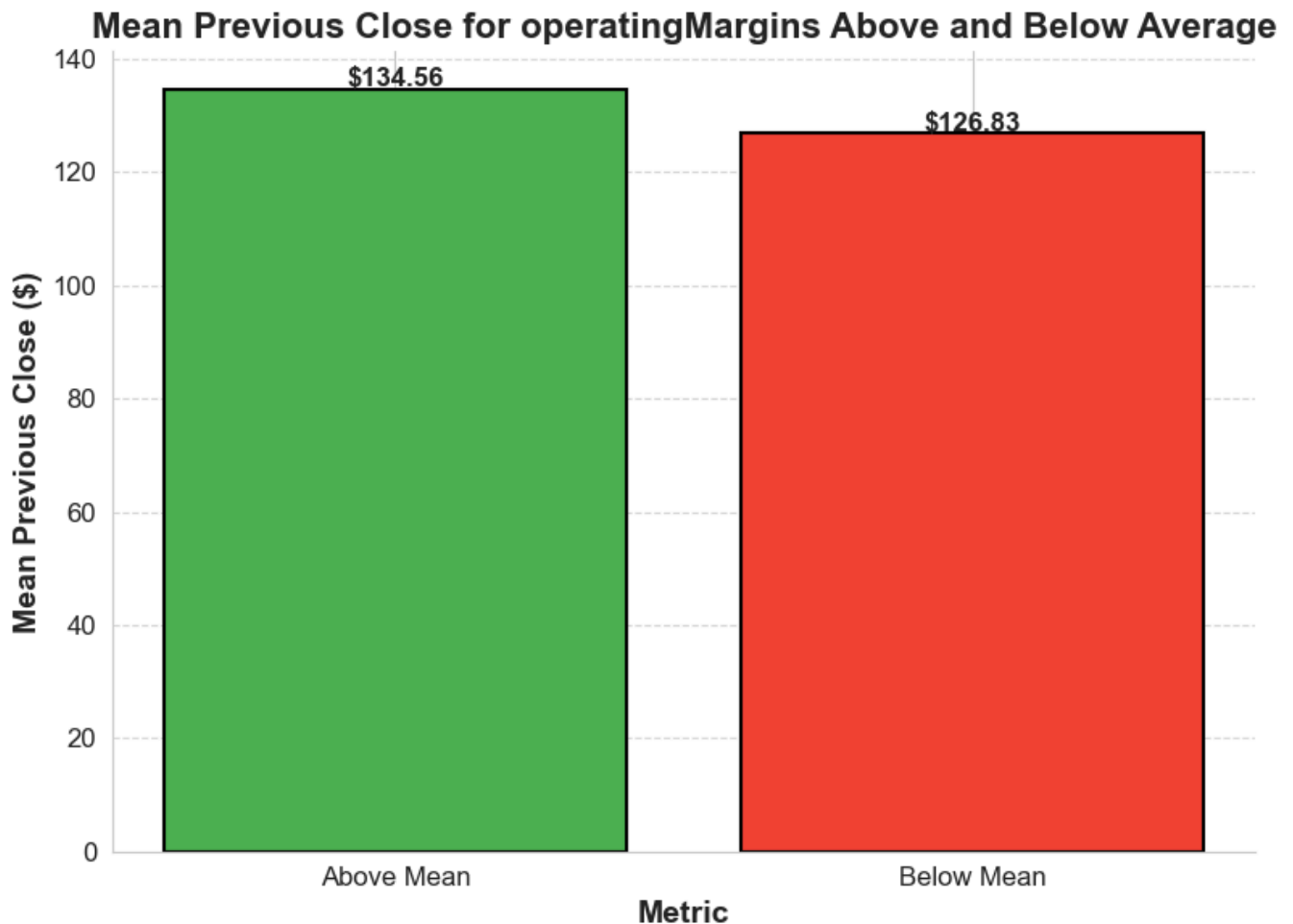


Fail to reject the null hypothesis

### Test Between Greater than Average operatingMargins vs Less than Average operatingMargins



```
descriptive_visual(metric='operatingMargins')
```



### ✓ Hypothesis 8.3: High Profit Margin vs Low Profit Margin

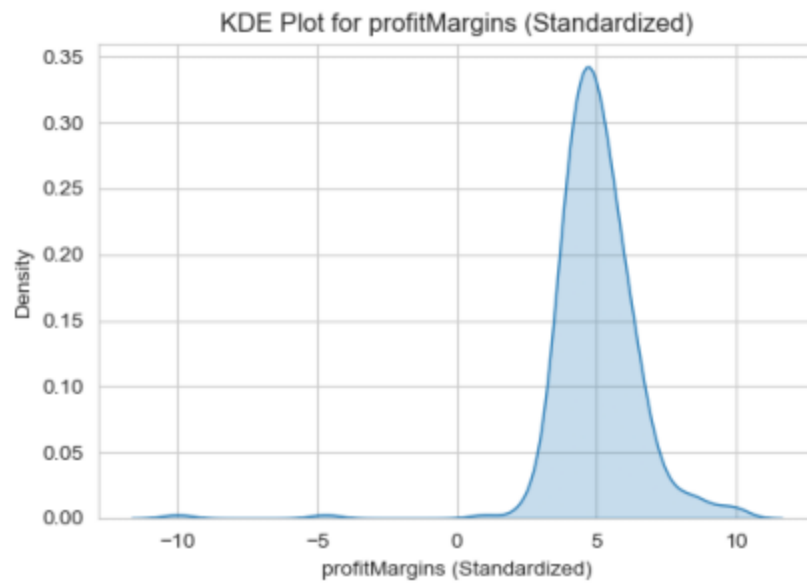
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Profit Margin is equal to companies with a Low Profit Margin.

The alternate hypothesis states that the mean closing price for companies with a High Profit Margin is NOT equal to companies with a Low Profit Margin.

```
show_plot('profitMargins')
```

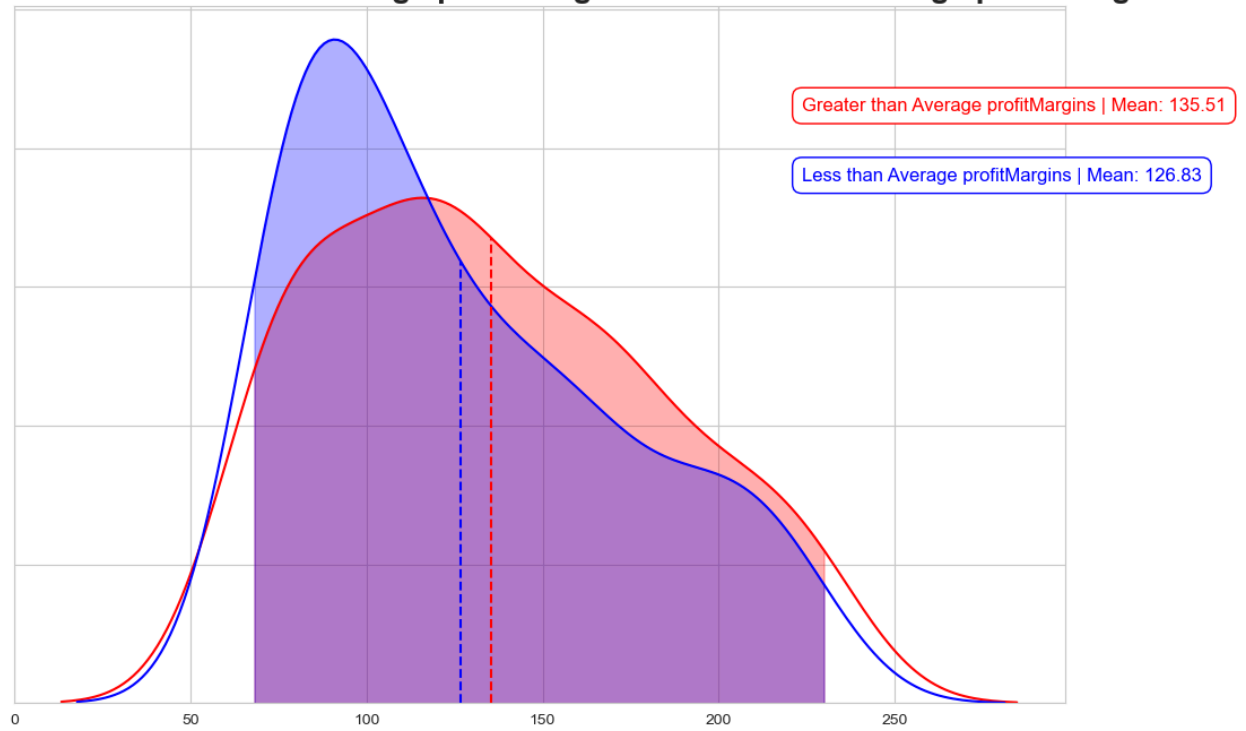


```
two_sample_kdeplot(metric='profitMargins')
```

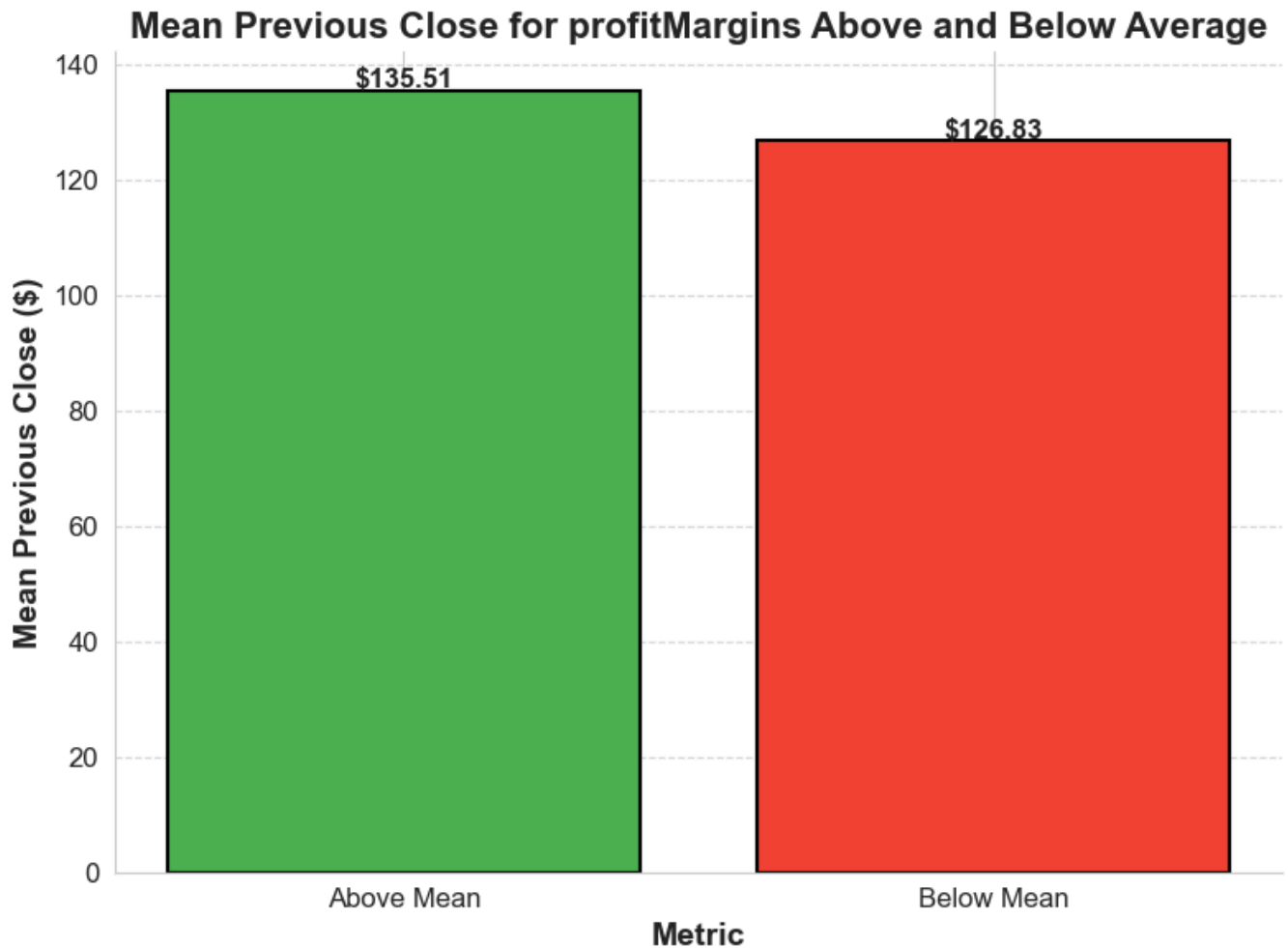


Fail to reject the null hypothesis

### Test Between Greater than Average profitMargins vs Less than Average profitMargins



```
descriptive_visual(metric='profitMargins')
```



## ✓ Hypothesis 9.1: High Quick Ratio vs Low Quick Ratio

Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

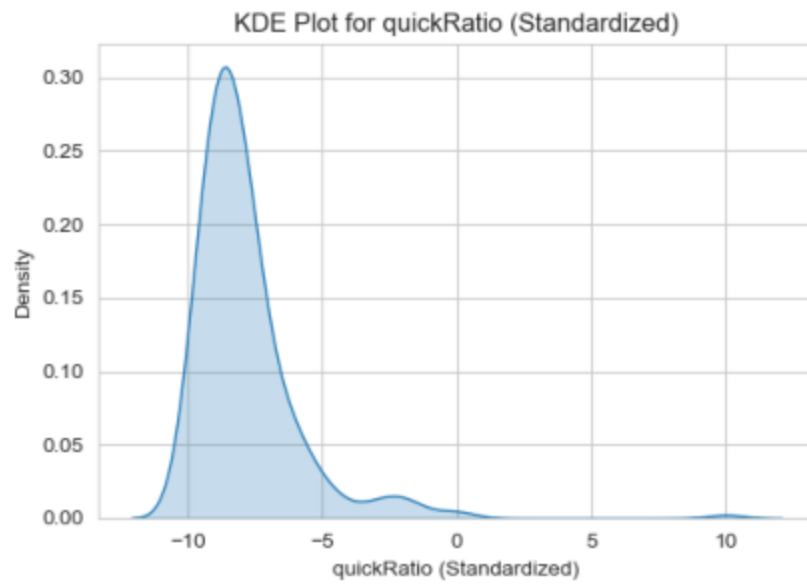
Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Quick Ratio is equal to companies with a Low Quick Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Quick Ratio is NOT equal to companies with a Low Quick Ratio.

```
show_plot('quickRatio')
```



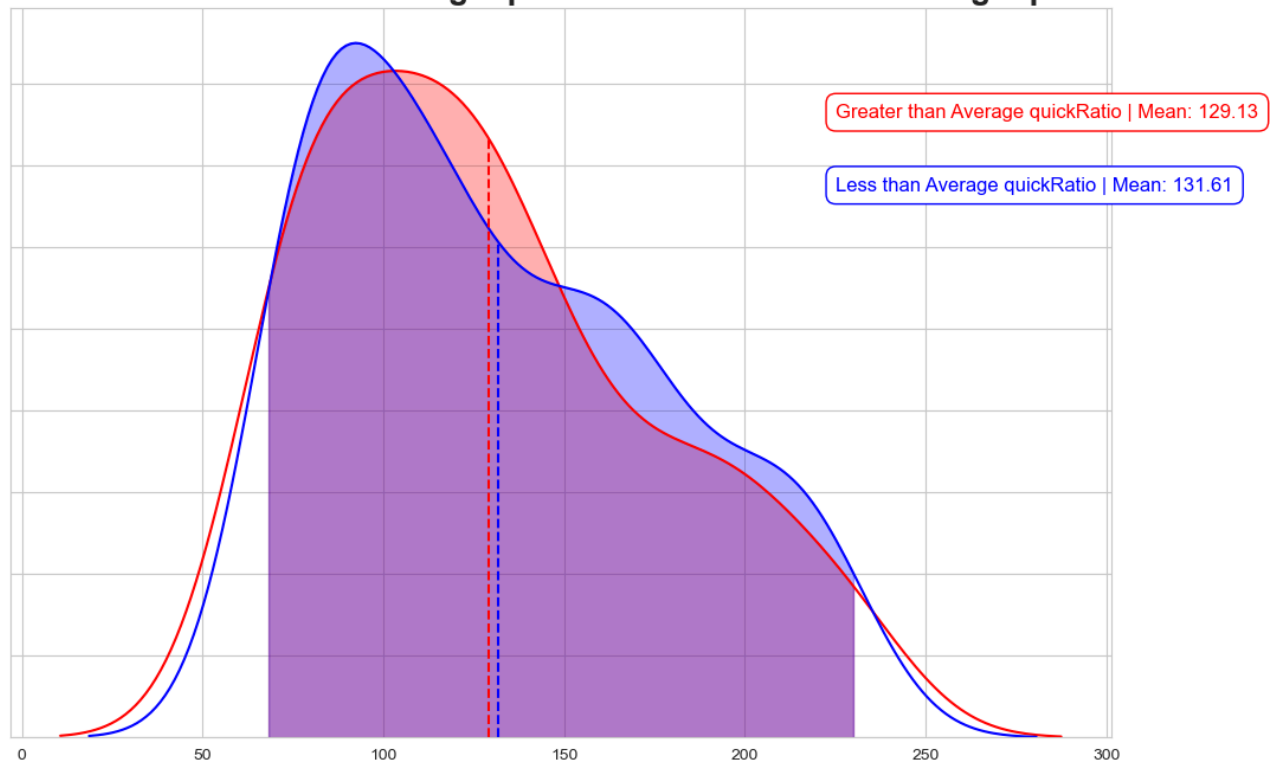


```
two_sample_kdeplot(metric='quickRatio')
```

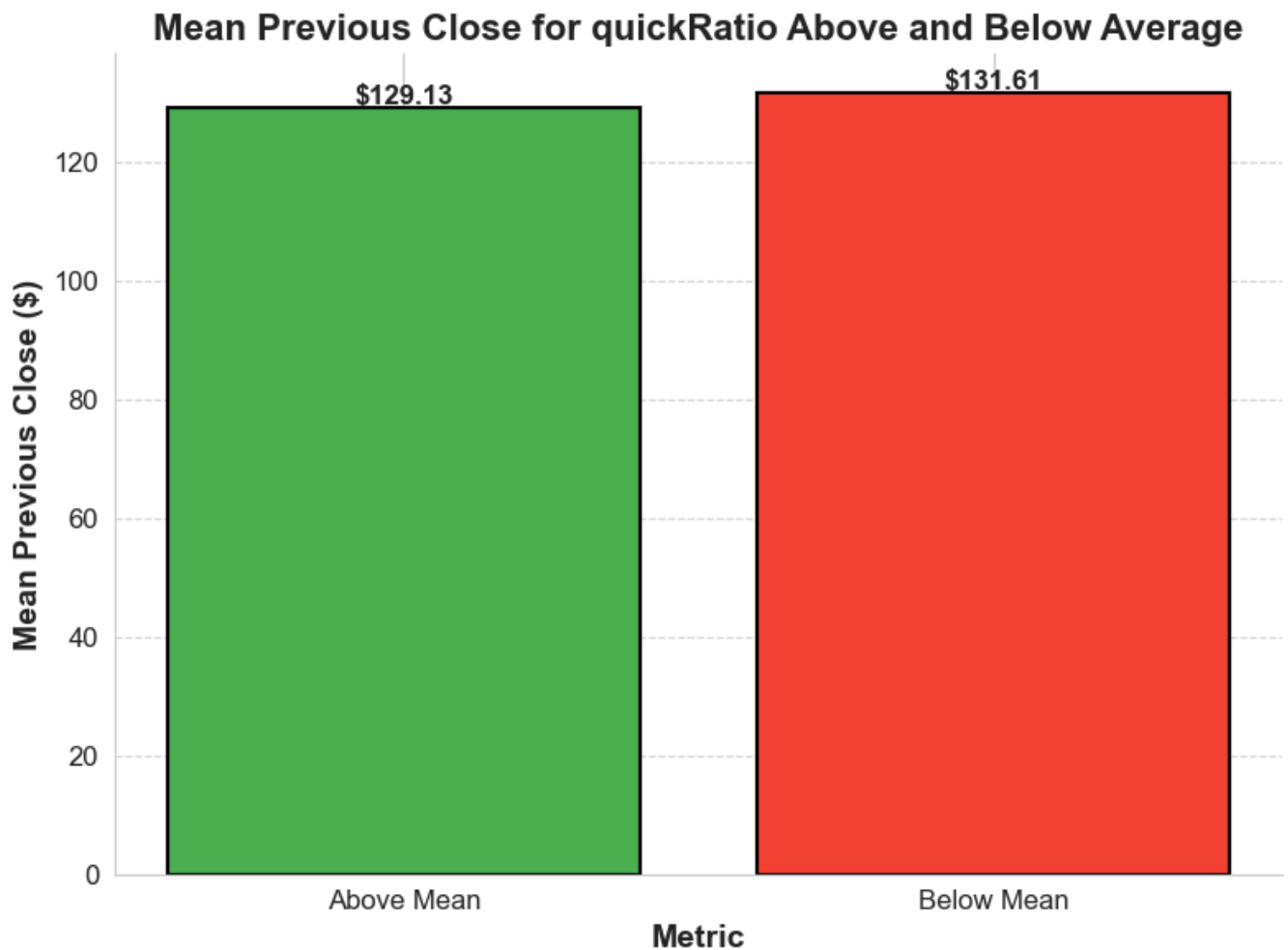


Fail to reject the null hypothesis

### Test Between Greater than Average quickRatio vs Less than Average quickRatio



```
descriptive_visual(metric='quickRatio')
```



## ✓ Hypothesis 9.2: High Current Ratio vs Low Current Ratio

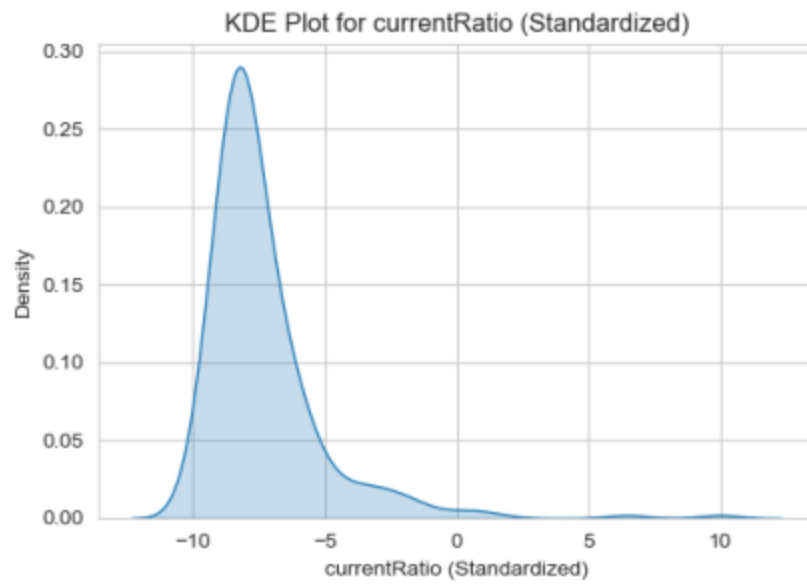
Null Hypothesis ( $H_0$ ):  $\mu_H = \mu_L$

Alternate Hypothesis ( $H_A$ ):  $\mu_H \neq \mu_L$

The null hypothesis states that the mean closing price for companies with a High Current Ratio is equal to companies with a Low Current Ratio.

The alternate hypothesis states that the mean closing price for companies with a High Current Ratio is NOT equal to companies with a Low Current Ratio.

```
show_plot('currentRatio')
```

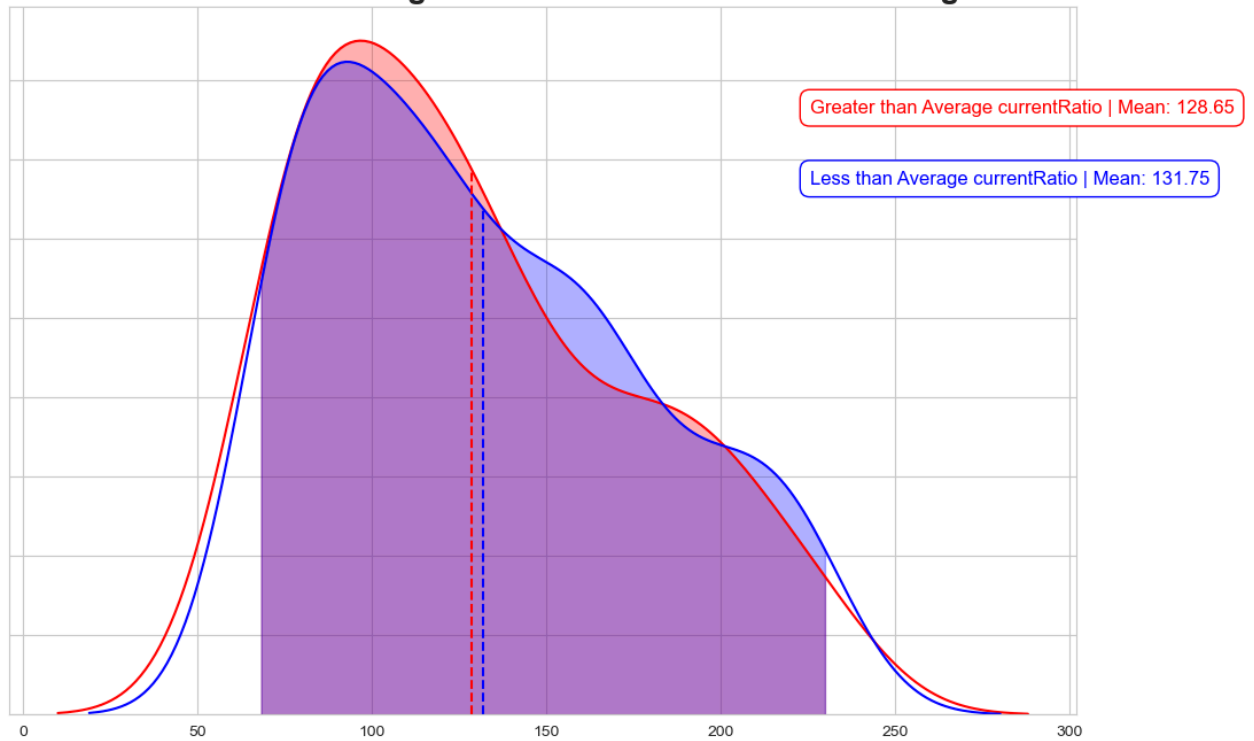


```
two_sample_kdeplot(metric='currentRatio')
```

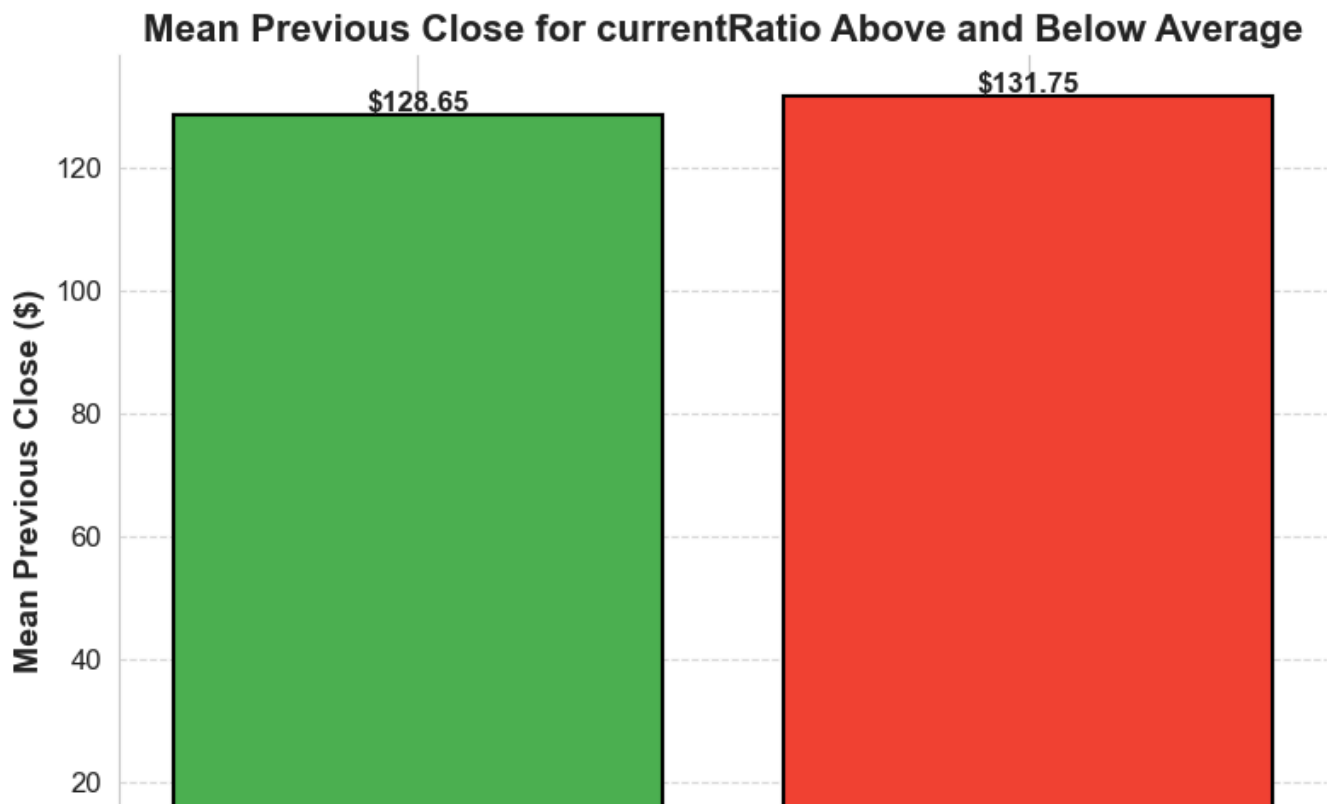


Fail to reject the null hypothesis

### Test Between Greater than Average currentRatio vs Less than Average currentRatio



```
descriptive_visual(metric='currentRatio')
```



✓ Hypothesis 10.1: High Enterprise-To-EBITDA Ratio vs Low Enterprise-To-EBITDA Ratio