Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load and Clean Data

In [2]:
```
# List of files to process
files = [
    "C:\\Users\\asvai\\OneDrive\\Desktop\\appledata.csv",
    "C:\\Users\\asvai\\OneDrive\\Desktop\\MicrosoftData.csv",
    "C:\\Users\\asvai\\OneDrive\\Desktop\\Qualcommdata.csv",
    "C:\\Users\\asvai\\OneDrive\\Desktop\\ciscodataset.csv",
    "C:\\Users\\asvai\\OneDrive\\Desktop\\amddataset.csv"
]
chunk_size = 100000  # Set size for loading data in chunks

# Process each file
for file in files:
    company_name = file.split("\\")[-1].split(".")[0]  # Extract company na
    chunks = pd.read_csv(file, chunksize=chunk_size)  # Read the CSV in chu

    company_df = pd.DataFrame()  # Initialize an empty DataFrame for the co
    for chunk in chunks:
        # Ensure 'Date' column is in datetime format
        chunk['Date'] = pd.to_datetime(chunk['Date'], errors='coerce')

        # Rename columns except for 'Date' with the company name
        chunk = chunk.rename(columns=lambda x: f"{company_name}_{x}" if x

        # Append the chunk to the company-specific DataFrame
        company_df = pd.concat([company_df, chunk], ignore_index=True)

    # Drop duplicates and sort by date
    company_df = company_df.drop_duplicates(subset=['Date']).sort_values(by

    # Save each company's processed data to a CSV file
    company_df.to_csv(f"{company_name}_processed.csv", index=False)
```

Merge Data

In [3]:
```
# List of processed files (assuming they have been saved in the previous st
processed_files = [
    "appledata_processed.csv",
    "MicrosoftData_processed.csv",
    "Qualcommdata_processed.csv",
    "ciscodataset_processed.csv",
    "amddataset_processed.csv"
]

# Initialize the combined DataFrame
combined_df = pd.read_csv(processed_files[0])  # Start with the first file

# Merge all processed files
for file in processed_files[1:]:
    chunk = pd.read_csv(file)
    combined_df = pd.merge(combined_df, chunk, on="Date", how="outer")  # (

# Convert 'Date' column to datetime and set as index
combined_df['Date'] = pd.to_datetime(combined_df['Date'])
combined_df.set_index('Date', inplace=True)

# Remove non-numeric characters and convert columns to numeric where possib
for column in combined_df.columns:
    combined_df[column] = pd.to_numeric(combined_df[column].replace('[^\d.]
```

Summary Statistics for Volume

In [9]:
```
# Filter volume columns only (assuming they contain 'Volume' in column name
volume_columns = [col for col in combined_df.columns if 'Volume' in col]
volume_data = combined_df[volume_columns]

# Resample volume data yearly and calculate summary statistics
yearly_volume_summary = volume_data.resample('Y').agg(['mean', 'median', 'r

# Save the yearly volume summary statistics to a CSV file
yearly_volume_summary.to_csv('yearly_volume_summary_statistics.csv')
print(yearly_volume_summary.head())  # Shows a sample of the summary statis
print(yearly_volume_summary.tail())
```

	appledata_Volume			
	mean	median	min	max
Date				
2014-12-31	1.852233e+08	171824800.0	108264240	334631120
2015-12-31	2.053931e+08	194808800.0	108684720	494407600
2016-12-31	1.456817e+08	130515160.0	74594800	322969840
2017-12-31	1.153464e+08	101867580.0	57111400	447349200
2018-12-31	1.405962e+08	120950560.0	55819240	364186240

	MicrosoftData_Volume			
	mean	median	min	max
Date				
2014-12-31	2.829265e+07	27987700.0	22180480	36364840
2015-12-31	3.275326e+07	31771820.0	19158590	59984700
2016-12-31	2.880275e+07	24989380.0	15702720	61262540
2017-12-31	2.181698e+07	20705390.0	11386390	49143030
2018-12-31	2.932275e+07	26080740.0	13634510	67970920

	Qualcommdata_Volume			
	mean	median	min	max
Date				
2014-12-31	1.513258e+07	9443018.0	6578256	70897810
2015-12-31	1.293916e+07	11301900.0	6373458	65827250
2016-12-31	9.728877e+06	9117372.5	3818566	21075750
2017-12-31	1.106713e+07	8713596.5	4377755	80796380
2018-12-31	1.165289e+07	9828384.0	3618923	35871980

	ciscodataset_Volume			
	mean	median	min	max
Date				
2014-12-31	2.692045e+07	26717910.0	18330440	41762660
2015-12-31	2.639153e+07	25150880.0	14399360	117194000
2016-12-31	2.409677e+07	21779445.0	11760100	92684060
2017-12-31	1.784602e+07	17130570.0	8727910	32649190
2018-12-31	2.304268e+07	20773780.0	13252090	54458320

	amddataset_Volume			
	mean	median	min	max
Date				
2014-12-31	1.478874e+07	15538960.0	7292980	25392300
2015-12-31	1.247296e+07	9055612.0	3530645	73180050
2016-12-31	2.689677e+07	20711140.0	5591035	170721100
2017-12-31	6.983272e+07	59297850.0	28107190	267432900
2018-12-31	8.352210e+07	72507680.0	28994160	256903900

	appledata_Volume			
	mean	median	min	max
Date				
2020-12-31	1.558445e+08	137636800.0	78260420	418474000
2021-12-31	8.751158e+07	80242480.0	40999950	178155000
2022-12-31	8.475567e+07	78172990.0	53950200	182602000
2023-12-31	6.311919e+07	56743120.0	31346600	154357300
2024-12-31	6.078184e+07	52318780.0	28111340	198134300

	MicrosoftData_Volume			
	mean	median	min	max
Date				
2020-12-31	3.776181e+07	32889260.0	19925840	93226370
2021-12-31	2.462256e+07	23595075.0	13900170	44727790
2022-12-31	2.994586e+07	28899185.0	16774610	51033800
2023-12-31	2.557969e+07	24348070.0	12484240	54686050
2024-12-31	1.921704e+07	18289360.0	9932830	40709240

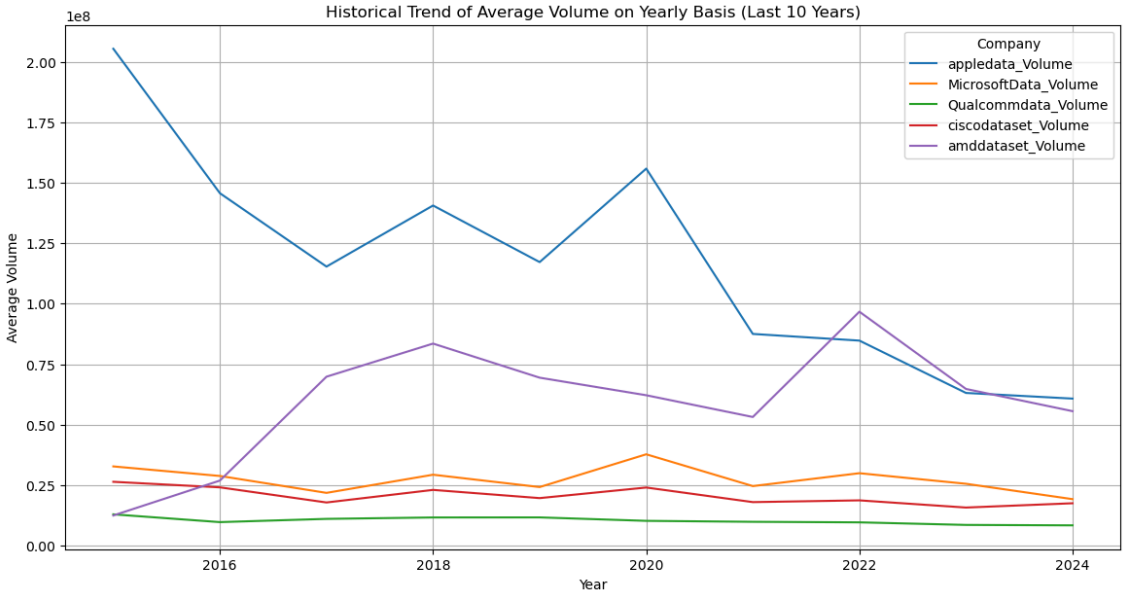
	Qualcommdata_Volume				\
	mean	median	min	max	
Date					
2020-12-31	1.025385e+07	9209197.0	4757536	27910880	
2021-12-31	9.857086e+06	7965431.5	4197116	48321480	
2022-12-31	9.631657e+06	8551812.0	4967492	36120200	
2023-12-31	8.568128e+06	7594454.0	2658635	29250380	
2024-12-31	8.392220e+06	7368388.0	3801395	25894850	

	ciscodataset_Volume				\
	mean	median	min	max	
Date					
2020-12-31	2.402713e+07	22029045.0	9814844	63608530	
2021-12-31	1.799480e+07	16720205.0	8117578	44540880	
2022-12-31	1.871777e+07	18357550.0	11816980	30317890	
2023-12-31	1.574931e+07	15497670.0	7336040	26332530	
2024-12-31	1.752339e+07	17855650.0	8118296	29669600	

	amddataset_Volume			
	mean	median	min	max
Date				
2020-12-31	6.216980e+07	54859330.0	28368150	155676100
2021-12-31	5.317958e+07	42760470.0	24310890	225368700
2022-12-31	9.669872e+07	92247750.0	47022190	225394100
2023-12-31	6.476508e+07	57061030.0	31780660	173372000
2024-12-31	5.558415e+07	52300405.0	24630430	120815200

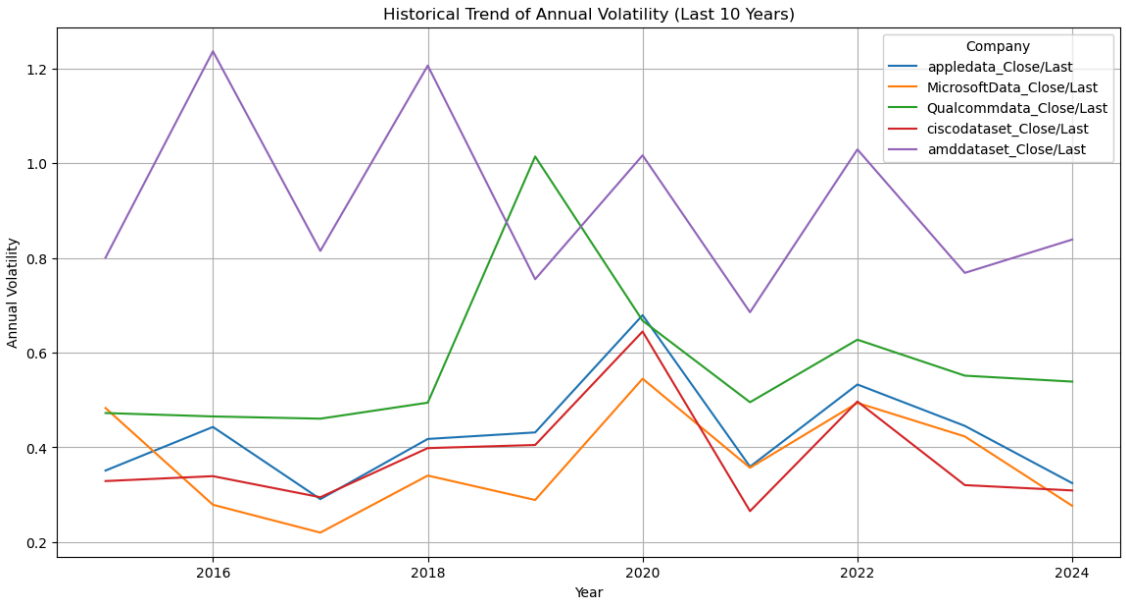
Historical Trends

In [5]:
```
# Extract the yearly average volume for each company
yearly_avg_volume = yearly_volume_summary.xs('mean', level=1, axis=1)

# Filter for the last 10 years
last_10_years = yearly_avg_volume.tail(10)

# Plot historical trend of average volume on a yearly basis for the last 1
plt.figure(figsize=(14, 7))
for column in last_10_years.columns:
    plt.plot(last_10_years.index.year, last_10_years[column], label=column

# Set plot title and labels
plt.title('Historical Trend of Average Volume on Yearly Basis (Last 10 Year
plt.xlabel('Year')
plt.ylabel('Average Volume')
plt.legend(title='Company', loc='upper right')
plt.grid(True)

# Show the plot
plt.show()
```



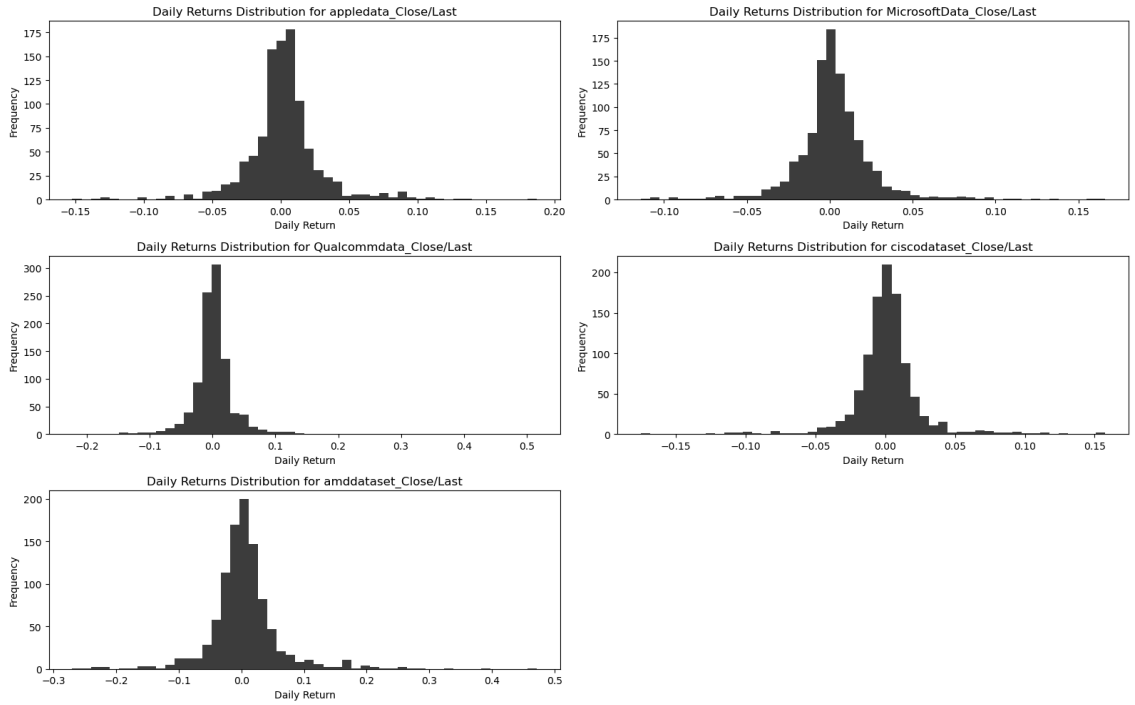
Calculate and Plot Annual Volatility(stock fluctuates over a period)

In [6]:
```
# Extract closing price columns (assuming they contain 'Close' in their nam
closing_columns = [col for col in combined_df.columns if 'Close' in col]
closing_prices = combined_df[closing_columns]

# Calculate daily returns
daily_returns = closing_prices.pct_change()

# Calculate annual volatility (standard deviation of daily returns, scaled
annual_volatility = daily_returns.resample('Y').std() * (252 ** 0.5)

# Filter for the last 10 years of volatility data
last_10_years_volatility = annual_volatility.tail(10)

# Plot historical trend of annual volatility for the last 10 years
plt.figure(figsize=(14, 7))
for column in last_10_years_volatility.columns:
    plt.plot(last_10_years_volatility.index.year, last_10_years_volatility[

# Set plot title and labels
plt.title('Historical Trend of Annual Volatility (Last 10 Years)')
plt.xlabel('Year')
plt.ylabel('Annual Volatility')
plt.legend(title='Company', loc='upper right')
plt.grid(True)

# Show the plot
plt.show()
```



Histogram

In [7]:
```
# Plot histograms for each company's daily returns
plt.figure(figsize=(16, 10))

for i, column in enumerate(daily_returns.columns, 1):
    plt.subplot(3, 2, i)  # Arrange plots in a 3x2 grid for 5 companies
    plt.hist(daily_returns[column].dropna(), bins=50, alpha=0.75, color='bl
    plt.title(f'Daily Returns Distribution for {column}')
    plt.xlabel('Daily Return')
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



Heatmap

In [8]:
```
# Filter for the last 10 years of volume data
last_10_years_volume = yearly_avg_volume.tail(10)

# Create a heatmap for the average volume over the past 10 years
plt.figure(figsize=(10, 6))
sns.heatmap(last_10_years_volume, annot=True, fmt=".2f", cmap="YlGnBu", cba

# Set plot title and labels
plt.title('Heatmap of Average Volume for Last 10 Years')
plt.xlabel('Company')
plt.ylabel('Year')

plt.show()
```