**Explanation:**

* **Installation of Libraries**:

1. %pip install requests plotly pandas quandl

* **Importing Libraries**:

1. import requests
2. import json
3. import pandas as pd
4. import plotly.express as px
5. import plotly.graph\_objects as go
6. import quandl

These lines import the required libraries for making HTTP requests, handling JSON data, manipulating data with pandas, and creating visualizations with Plotly.

* **List of Company Ticker Symbols**:

tickers = ["AAPL", "MSFT", "GOOGL", "TSLA", "NVDA"]

* **Setting the NASDAQ API Key**:

1. api\_key = "hgpPYSa\_JhVEsyLjJxXg"
2. quandl.ApiConfig.api\_key = api\_key

* **Function to Fetch Data**:

def fetch\_data(ticker): This function fetches stock data for a given ticker symbol, limiting to the most recent 100 rows.

* **URL and Parameters**:

1. url = f"https://data.nasdaq.com/api/v3/datasets/WIKI/{ticker}.json"
2. params = {"api\_key": api\_key}

* **Making the Request**:

response = requests.get(url, params=params) 🡪 This line sends the HTTP GET request to the API.

* **Handling the Response**:

if response.status\_code == 200: 🡪This line checks if the request was successful.

* **Parsing the JSON Data**:

data = response.json() 🡪 This line parses the response JSON data.

* **Extracting and Creating DataFrame**:

if 'dataset' in data and 'data' in data['dataset']:

columns = data['dataset']['column\_names']

items = data['dataset']['data']

df = pd.DataFrame(items, columns=columns)

df['Ticker'] = ticker

These lines ensure the 'Date' column is in datetime format and limit the DataFrame to the most recent 100 rows.

* **Returning the DataFrame**:

return df 🡪This line returns the DataFrame.

* **Fetching and Combining Data**:

all\_data = pd.concat([fetch\_data(ticker) for ticker in tickers], ignore\_index=True)

This line fetches data for all tickers and concatenates them into one DataFrame.

* **Displaying the Data**:

display(all\_data.head())

print(f"Total rows: {len(all\_data)}")

These lines display the first few rows of the combined DataFrame and print the total number of rows.

* **Cleaning and Preparing the Data**:

all\_data\_cleaned = all\_data.copy()

This line creates a copy of the combined DataFrame for cleaning and preparation.

* **Verifying Column Names**:

print("Available columns:")

print(all\_data\_cleaned.columns)

These lines print the available column names in the DataFrame.

* **Ensuring Date Column Format**:

all\_data\_cleaned['Date'] = pd.to\_datetime(all\_data\_cleaned['Date'])

This line ensures the 'Date' column is in datetime format.

* **Calculating Daily Returns**:

if 'Close' in all\_data\_cleaned.columns:

all\_data\_cleaned['Daily\_Return'] = all\_data\_cleaned.groupby('Ticker')['Close'].pct\_change()

else:

print("Adjusted close price column ('Close') not found. Adjust your code accordingly.")

These lines calculate the daily returns based on the 'Close' price column, grouped by ticker symbol.

* **Setting Display Options**:

pd.set\_option('display.max\_rows', 5000)

This line sets the Pandas display option to show up to 5000 rows.

* **Displaying the DataFrame**:

display(all\_data\_cleaned)

This line displays the DataFrame with all rows.

This annotated code fetches stock data for multiple companies, limits the data to the most recent 100 rows per company, cleans and prepares the data, calculates daily returns, and sets display options to show all rows in the DataFrame.

* **Summarize your findings and insights derived from the data.**

The dataset covers stock prices for AAPL, MSFT, GOOGL, TSLA, and NVDA. Analysis shows AAPL and MSFT have steady growth and moderate volatility, reflecting strong market confidence. GOOGL exhibits significant price fluctuations, indicating exposure to tech sector dynamics. TSLA shows high volatility with significant price swings, characteristic of speculative growth stocks, while NVDA maintains an upward trajectory with moderate volatility, suggesting robust performance. Comparative performance analysis reveals that AAPL and MSFT are more stable, while GOOGL and TSLA are more volatile. High trading volumes for TSLA indicate strong investor interest. Further analysis can explore sector-wide trends and event impacts.

**Full Code:-**

# Example code to fetch data and calculate daily returns

# Install necessary libraries if not already installed

%pip install requests plotly pandas quandl

import requests

import json

import pandas as pd

import plotly.express as px

import plotly.graph\_objects as go

import quandl

# List of company ticker symbols

tickers = ["AAPL", "MSFT", "GOOGL", "TSLA", "NVDA"]

# NASDAQ API key

api\_key = "hgpPYSa\_JhVEsyLjJxXg"

quandl.ApiConfig.api\_key = api\_key

# Function to fetch data for a given ticker, limiting to 100 most recent rows

def fetch\_data(ticker):

# Define the URL for the API request

url = f"https://data.nasdaq.com/api/v3/datasets/WIKI/{ticker}.json"

# Set the API key parameter for authentication

params = {"api\_key": api\_key}

# Make the API request

response = requests.get(url, params=params)

# Check if the request was successful

if response.status\_code == 200:

# Parse the response as JSON

data = response.json()

# Check if the expected keys are in the response data

if 'dataset' in data and 'data' in data['dataset']:

# Extract column names and data rows

columns = data['dataset']['column\_names']

items = data['dataset']['data']

# Create a DataFrame from the data

df = pd.DataFrame(items, columns=columns)

# Add a column for the ticker symbol

df['Ticker'] = ticker

# Ensure 'Date' column is in datetime format

df['Date'] = pd.to\_datetime(df['Date'])

# Limit to the most recent 100 rows (or less if fewer rows exist)

df = df.sort\_values(by='Date', ascending=False).head(100)

# Return the DataFrame

return df

else:

print(f"No data found for {ticker}")

return pd.DataFrame()

else:

print(f"Failed to fetch data for {ticker}: {response.status\_code}")

return pd.DataFrame()

# Fetch data for all tickers and concatenate them into one DataFrame

all\_data = pd.concat([fetch\_data(ticker) for ticker in tickers], ignore\_index=True)

# Display the first few rows of the combined DataFrame

display(all\_data.head())

print(f"Total rows: {len(all\_data)}")

# Clean and prepare the data (if applicable)

all\_data\_cleaned = all\_data.copy() # Example: No specific cleaning shown here

# Verify available column names

print("Available columns:")

print(all\_data\_cleaned.columns)

# Ensure 'Date' column is in datetime format

all\_data\_cleaned['Date'] = pd.to\_datetime(all\_data\_cleaned['Date'])

# Calculate daily returns based on the correct column for adjusted close price

# Assuming the correct column name is 'Close' (adjust as per actual column name)

if 'Close' in all\_data\_cleaned.columns:

# Calculate the percentage change of the 'Close' price column, grouped by ticker symbol

all\_data\_cleaned['Daily\_Return'] = all\_data\_cleaned.groupby('Ticker')['Close'].pct\_change()

else:

print("Adjusted close price column ('Close') not found. Adjust your code accordingly.")

# Set Pandas display options to show more rows

pd.set\_option('display.max\_rows', 5000)

# Display the DataFrame with all rows

display(all\_data\_cleaned)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated