STA 141B Final Project

Group Members:

Isabelle Berkowitz, Vinay Singal, Keon Sadeghi, Carly Schwartzberg, Aryan Punjani

Project Outline

The following serves as a guide to make our final project easy to follow along, including an outline of how we structed our project

1. **Introduction**
2. **Graphs and Visualizations**
3. **NLP Analysis for Stock Market news from CNBC**
4. **API key for Unemployment rates throughout Covid-19 Pandemic**
5. **Conclusion**

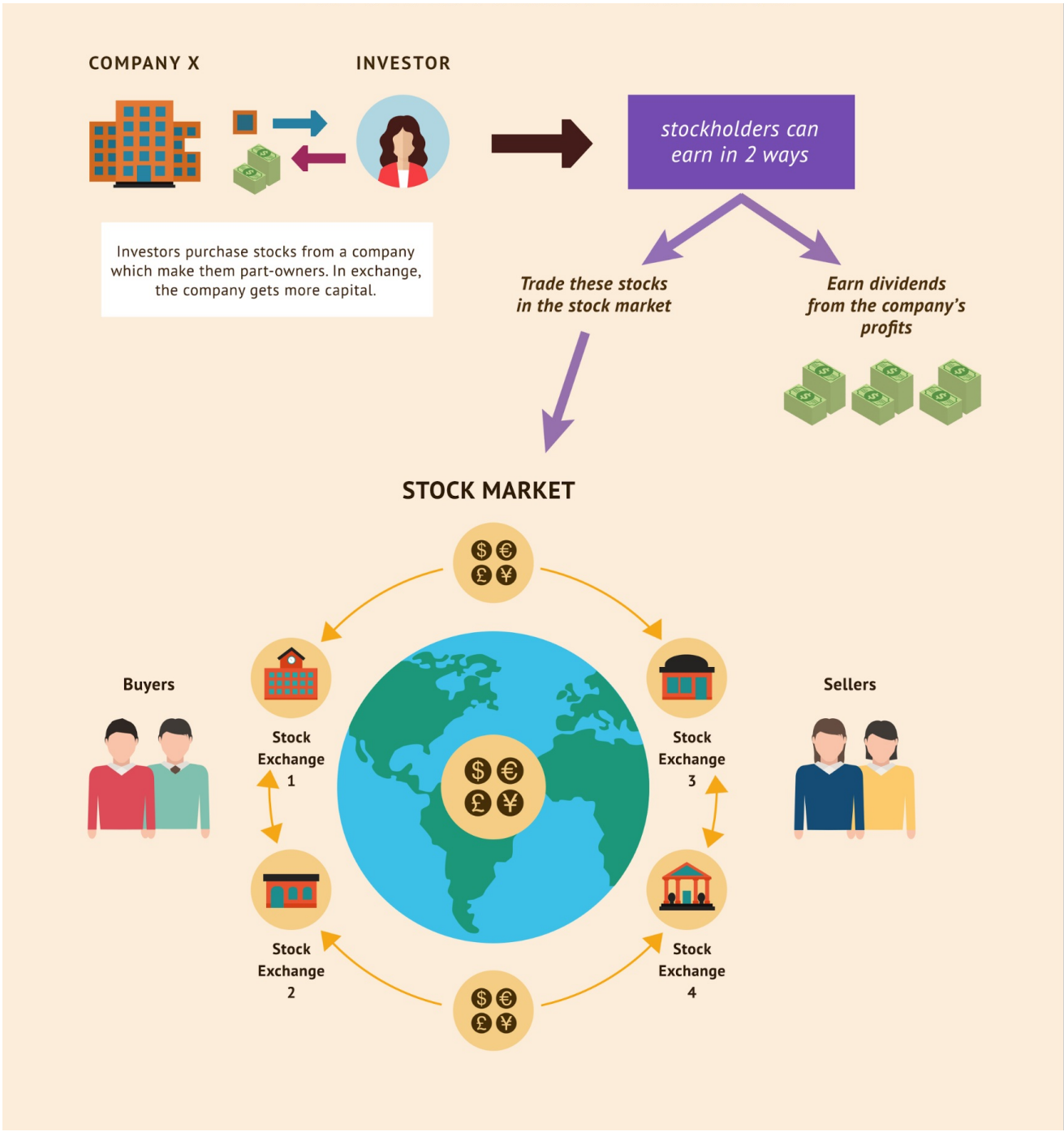
In [1]:
```
#setting up our workign enviornment
!pip install yfinance
!pip install pandas
!pip install matplotlib.pyplot


import yfinance as yf
import pandas as pd
import plotly.graph_objs as go
import matplotlib.pyplot as plt
import numpy as np
import requests
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from bs4 import BeautifulSoup
import string
```

```
Requirement already satisfied: yfinance in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site
-packages (0.2.12)
Requirement already satisfied: multitasking>=0.0.7 in /Library/Frameworks/Python.framework/Versions/3.9/lib/pyt
hon3.9/site-packages (from yfinance) (0.0.11)
Requirement already satisfied: frozendict>=2.3.4 in /Library/Frameworks/Python.framework/Versions/3.9/lib/pytho
n3.9/site-packages (from yfinance) (2.3.5)
Requirement already satisfied: numpy>=1.16.5 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9
/site-packages (from yfinance) (1.24.2)
Requirement already satisfied: appdirs>=1.4.4 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.
9/site-packages (from yfinance) (1.4.4)
Requirement already satisfied: html5lib>=1.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9
/site-packages (from yfinance) (1.1)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/
python3.9/site-packages (from yfinance) (4.11.1)
Requirement already satisfied: pytz>=2022.5 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/
site-packages (from yfinance) (2022.7)
Requirement already satisfied: pandas>=1.3.0 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9
/site-packages (from yfinance) (1.5.3)
Requirement already satisfied: requests>=2.26 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.
9/site-packages (from yfinance) (2.28.1)
Requirement already satisfied: lxml>=4.9.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/s
ite-packages (from yfinance) (4.9.2)
Requirement already satisfied: cryptography>=3.3.2 in /Library/Frameworks/Python.framework/Versions/3.9/lib/pyt
hon3.9/site-packages (from yfinance) (39.0.2)
Requirement already satisfied: soupsieve>1.2 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9
/site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.3.2.post1)
Requirement already satisfied: cffi>=1.12 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/si
te-packages (from cryptography>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: webencodings in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/
site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: six>=1.9 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site
-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/
python3.9/site-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/p
ython3.9/site-packages (from requests>=2.26->yfinance) (1.26.13)
Requirement already satisfied: idna<4,>=2.5 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/
site-packages (from requests>=2.26->yfinance) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in /Library/Frameworks/Python.framework/Versions/3.9/li
b/python3.9/site-packages (from requests>=2.26->yfinance) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in /Library/Frameworks/Python.framework/Versions/3.9/lib/pyth
on3.9/site-packages (from requests>=2.26->yfinance) (2022.12.7)
Requirement already satisfied: pycparser in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/sit
e-packages (from cffi>=1.12->cryptography>=3.3.2->yfinance) (2.21)
Requirement already satisfied: pandas in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-p
ackages (1.5.3)
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/
site-packages (from pandas) (2022.7)
Requirement already satisfied: numpy>=1.20.3 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9
/site-packages (from pandas) (1.24.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/
python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site
-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
ERROR: Could not find a version that satisfies the requirement matplotlib.pyplot (from versions: none)
ERROR: No matching distribution found for matplotlib.pyplot
```

Part 1: Introdcution

1A. Purpose

The stock market is a dynamic system for several different exchanges within our economy in which shares of companies are bought and sold. It allows investors and businesses to earn more profit and grow their companies. The figure below explains how the stock market operates in the cycle in two ways 1) through trading their stocks into the market and 2) through company profits which are dependent on a value that is determined on how well the company is performing and the market demand for their goods. There are multiple factors that influence the value of the shares such as how our economy is fluctuating, the blow-up of company products, and trends within the industry. The Dow Jones Industrial Average(DJIA) is a well-known stock market index which tracks 30 large 'publicly-owned blue-chip companies trading on the New York Stock Exchange (NYSE) and Nasdaq.' Our main goal is to help mitigate stock market volatility and fluctuation during shocks in the country. We hypothesize that the stock market bounced back strong from the Covid-19 pandemic so with that premise, when shocks occur in the country there should not be as much volatility in the market since we know that after one of the biggest economic turmoils of the 2000s the market bounced back strong.Our key question we are aiming to answer is how the stock market, specifically the trend and values of Dow Jones Industrial in pre covid (2018), peak-covid(2020), and during the 2022 recession was affected.



1B. Overview of Data Manipulation

The data set we used from Yahoo Finance contained the Dow Jones stock data from 1962 to the present. However, for this project, we wanted to focus on the effects of covid-19 on the stock market, which means we wanted to look at specific years. In order to isolate these years, we extracted a specific time period from the Yahoo Finance data. We set the start date to January 1st, 2018, and the end date to December 31st, 2022. In order to examine the effects of covid-19, we chose 2018 as our pre-covid data, 2020 as our data for the peak of the pandemic (this was the year the stock market was most affected), and 2022 as our post-covid data. While covid-19 was still present in 2022, it had begun to subside, and many businesses had recovered by this time. Since we extracted the time period 2018-2022, we wanted to narrow our focus even more, therefore we removed 2019 and 2021 from our dataset.

The years 2019 and 2021 marked significant turning points. In late 2019 covid-19 had just begun, therefore the overall data of that year would not be a suitable representation of a mid-covid year which is what we were looking for in our analysis. The covid-19 pandemic still impacted the U.S. in early 2021, making this year infeasible for post-covid representation. Therefore in order to accurately capture the effects of the pandemic during mid-covid and post-covid we chose 2020 and 2022 rather than 2019 and 2021. 2018 provided us a good baseline for pre-covid, 2020 had the best representation of the peak of the pandemic, and 2022 offered the most accurate representation of post-covid recovery. By choosing these specific years, we were able to attain the most accurate view of the effects covid-19 had on the stock market. We then calculated averages by using the highs and lows columns of the data set, and added an "averages" column to our dataset. This displayed the average stock price for each company for every day throughout the dataset. This additional data allowed us to gain a better understanding of the effects that covid-19 had on the stock market during 2018, 2020, and 2022.

By manipulating and refining our data set in this way, we were able to acquire all the information necessary for our project while removing all irrelevant data. The resulting dataset is an accurate representation of the stock market during pre-covid, mid-covid, and post-covid, and provided us with useful insights for our analysis.

1C. Summary and Explanation of Data Collection

The datasets we used for this project were extracted from yahoo finance using the yfinance library in python.We collected stock market data, specifically on the Dow Jones Industrial Average from 1962 and 2022. We took the data from 2018, 2020, and 2022 removing 2019 and 2021 in order to isolate the years we wanted to focus on. We then combined them into one dataset giving us 1259 rows and 210 columns, giving us that many rows as we used the whole twelve months. In addition, we mainly focused on two key columns from this data set which are the high and low, representing each company's high and low values per day.

In [2]:

```python
#Defined the start date and end date of the data.
startDate = '2018-01-01'
endDate = '2022-12-31'

#Defines the list of Dow Jones Industrial Average (DJIA) tickers.
names = ['AAPL', 'AMGN', 'AXP', 'BA', 'CAT', 'CRM', 'CSCO', 'CVX', 'DIS', 'DOW', 'GS', 'HD', 'HON', 'IBM', 'INT

#Gets the data for the DJIA stocks using the start and end dates we selected
data = {t: yf.download(t, start=startDate, end=endDate) for t in names}

#Combines the data in DJIA into a single dataset.
data2 = yf.download(names, start=startDate, end=endDate, group_by='t')

#Removes the data from 2019 and 2021 using != operator
data2 = data2[data2.index.year != 2019]
data2 = data2[data2.index.year != 2021]

#Uses a for loop to calculate the high/low average for every company in DJIA.
for company in data2.columns.levels[0]:
    data2[company, 'Average'] = (data2[company, 'High'] + data2[company, 'Low']) / 2

#Sorts the columns alphabetically, sets axis=1 to perform a column-wise operation..
data2 = data2.sort_index(axis=1)

#Prints the DataFrame.
data2
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  30 of 30 completed
```

Out[2]:

	AAPL							AMGN			...	
	Adj Close	Average	Close	High	Low	Open	Volume	Adj Close	Average	Close	...	Low
Date												
2018-01-02	40.888062	42.695000	43.064999	43.075001	42.314999	42.540001	102223600	151.353928	176.120003	177.000000	...	72.970001
2018-01-03	40.880951	43.313751	43.057499	43.637501	42.990002	43.132500	118071600	154.209991	178.599998	180.339996	...	74.309998
2018-01-04	41.070839	43.193750	43.257500	43.367500	43.020000	43.134998	89738400	153.560104	179.805000	179.580002	...	70.730003
2018-01-05	41.538441	43.552500	43.750000	43.842499	43.262501	43.360001	94640000	154.475052	179.955002	180.649994	...	71.820000
2018-01-08	41.384155	43.692499	43.587502	43.902500	43.482498	43.587502	82271200	154.432327	179.839996	180.600006	...	72.750000
...	...	...	...	...	...	...	...	...	...	...	...	..
2022-12-23	131.658981	131.029999	131.860001	132.419998	129.639999	130.919998	63814900	261.612915	264.095001	263.920013	...	38.209999
2022-12-27	129.831772	130.065002	130.029999	131.410004	128.720001	131.380005	69007800	261.087555	264.744995	263.390015	...	38.110001
2022-12-28	125.847855	128.450001	126.040001	131.029999	125.870003	129.669998	85438400	259.134766	263.380005	261.420013	...	37.560001
2022-12-29	129.412415	129.105000	129.610001	130.479996	127.730003	127.989998	75703700	260.859558	263.510010	263.160004	...	37.360001
2022-12-30	129.731918	128.689999	129.929993	129.949997	127.430000	128.410004	77034200	260.344086	261.370010	262.640015	...	36.970001

755 rows × 210 columns

In [3]:
```
#Since the data originally started in 1962 we wanted to isolate a certain section of time to analyze in order t
#Here we are isolating 2018 specifically to look at a pre covid year, we are setting the start date as January

startOne = '2018-01-01'
endOne = '2018-12-31'

#Now we download the yahoo finance library in order to extract the data
#We use startOne and endOne in order to create a table for 2018 data

start_data = yf.download('^DJI', start=startOne, end=endOne)

#Here we are isolating 2020 specifically to look at a year in the middle of peak covid, we are setting the star

startTwo = '2020-01-01'
endTwo = '2020-12-31'

#We use startTwo and endTwo in order to create a table for 2020 data

mid_data = yf.download('^DJI', start=startTwo, end=endTwo)

#Here we are isolating 2022 specifically to look at a post covid year, we are setting the start date as January

startThree = '2022-01-01'
endThree = '2022-12-31'

#We use startThree and endThree in order to create a table for 2022 data

end_data = yf.download('^DJI', start=startThree, end=endThree)
```

```
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
```

In [4]:
```
# Displaying the Dow Jones market prices every working day from the year 2018

start_data
```

Out[4]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-01-02	24809.349609	24864.189453	24741.699219	24824.009766	24824.009766	341130000
2018-01-03	24850.449219	24941.919922	24825.550781	24922.679688	24922.679688	456790000
2018-01-04	24964.859375	25105.960938	24963.269531	25075.130859	25075.130859	403280000
2018-01-05	25114.919922	25299.789062	25112.009766	25295.869141	25295.869141	358020000
2018-01-08	25308.400391	25311.990234	25235.410156	25283.000000	25283.000000	341390000
...	...	...	...	...	...	...
2018-12-21	22871.740234	23254.589844	22396.339844	22445.369141	22445.369141	900510000
2018-12-24	22317.279297	22339.869141	21792.199219	21792.199219	21792.199219	308420000
2018-12-26	21857.730469	22878.919922	21712.529297	22878.449219	22878.449219	433080000
2018-12-27	22629.060547	23138.890625	22267.419922	23138.820312	23138.820312	407940000
2018-12-28	23213.609375	23381.880859	22981.330078	23062.400391	23062.400391	336510000

250 rows × 6 columns

In [5]:
```
# Displaying the Dow Jones market prices every working day from the year 2020

mid_data
```

Out[5]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	28638.970703	28872.800781	28627.769531	28868.800781	28868.800781	251820000
2020-01-03	28553.330078	28716.310547	28500.359375	28634.880859	28634.880859	239590000
2020-01-06	28465.500000	28708.019531	28418.630859	28703.380859	28703.380859	252760000
2020-01-07	28639.179688	28685.500000	28565.279297	28583.679688	28583.679688	258900000
2020-01-08	28556.140625	28866.179688	28522.509766	28745.089844	28745.089844	291750000
...	...	...	...	...	...	...
2020-12-23	30046.730469	30292.529297	30046.730469	30129.830078	30129.830078	274050000
2020-12-24	30155.919922	30209.669922	30099.300781	30199.869141	30199.869141	145570000
2020-12-28	30283.230469	30525.560547	30283.230469	30403.970703	30403.970703	302490000
2020-12-29	30492.070312	30588.789062	30274.240234	30335.669922	30335.669922	357610000
2020-12-30	30415.089844	30525.349609	30393.039062	30409.560547	30409.560547	291890000

252 rows × 6 columns

In [6]:
```
# Displaying the Dow Jones market prices every working day from the year 2022.

end_data
```

Out[6]:

Date	Open	High	Low	Close	Adj Close	Volume
2022-01-03	36321.589844	36595.820312	36246.449219	36585.058594	36585.058594	347930000
2022-01-04	36636.000000	36934.839844	36636.000000	36799.648438	36799.648438	435080000
2022-01-05	36722.601562	36952.648438	36400.390625	36407.109375	36407.109375	462040000
2022-01-06	36409.050781	36464.191406	36200.679688	36236.468750	36236.468750	385890000
2022-01-07	36249.589844	36382.839844	36111.531250	36231.660156	36231.660156	356110000
...	...	...	...	...	...	...
2022-12-23	32961.058594	33226.140625	32814.019531	33203.929688	33203.929688	221050000
2022-12-27	33224.230469	33387.718750	33069.578125	33241.558594	33241.558594	246010000
2022-12-28	33264.761719	33379.550781	32869.148438	32875.710938	32875.710938	252260000
2022-12-29	33021.429688	33293.421875	33020.351562	33220.800781	33220.800781	243060000
2022-12-30	33121.609375	33152.550781	32847.820312	33147.250000	33147.250000	295500000

251 rows × 6 columns

In [7]:
```
# We now are displaying the highs, lows, and average stock price for each company in the Dow Jones Industrial p
```

In [8]:
```
highs = data2.loc[:, (slice(None), 'High')]
lows = data2.loc[:, (slice(None), 'Low')]
averages = data2.loc[:, (slice(None), 'Average')]
```

```
result = pd.concat([highs, lows, averages], axis = 1)
result
```

Out[8]:

	AAPL	AMGN	AXP	BA	CAT	CRM	CSCO	CVX	DIS	DOW	...	MRK
	High	High	High	High	High	High	High	High	High	High	...	Average
Date												
2018-01-02	43.075001	177.820007	99.730003	296.989990	159.389999	104.699997	38.950001	127.739998	111.809998	NaN	...	53.816793
2018-01-03	43.637501	181.440002	99.760002	298.500000	157.490005	106.139999	39.279999	128.940002	113.190002	NaN	...	53.540075
2018-01-04	43.367500	180.860001	101.650002	298.420013	159.580002	107.660004	39.540001	128.350006	113.000000	NaN	...	54.293894
2018-01-05	43.842499	180.880005	101.080002	308.890015	162.050003	108.300003	39.880001	128.100006	112.680000	NaN	...	54.379772
2018-01-08	43.902500	181.250000	101.199997	310.859985	166.429993	109.139999	39.959999	128.630005	111.279999	NaN	...	54.270039
...	...	...	...	...	...	...	...	...	...	...	...	...
2022-12-23	132.419998	265.290009	147.139999	189.429993	240.539993	129.860001	47.490002	177.580002	88.070000	50.910999	...	111.500000
2022-12-27	131.410004	266.609985	147.860001	192.440002	245.050003	131.750000	47.709999	180.229996	87.940002	51.220001	...	112.059998
2022-12-28	131.029999	265.619995	146.820007	191.320007	243.820007	131.139999	47.770000	179.300003	86.690002	51.349998	...	111.974998
2022-12-29	130.479996	264.880005	147.619995	190.250000	241.610001	132.949997	47.740002	179.199997	88.239998	50.801998	...	111.224998
2022-12-30	129.949997	263.230011	147.929993	190.649994	240.160004	132.639999	47.669998	179.949997	87.120003	50.530998	...	110.770000

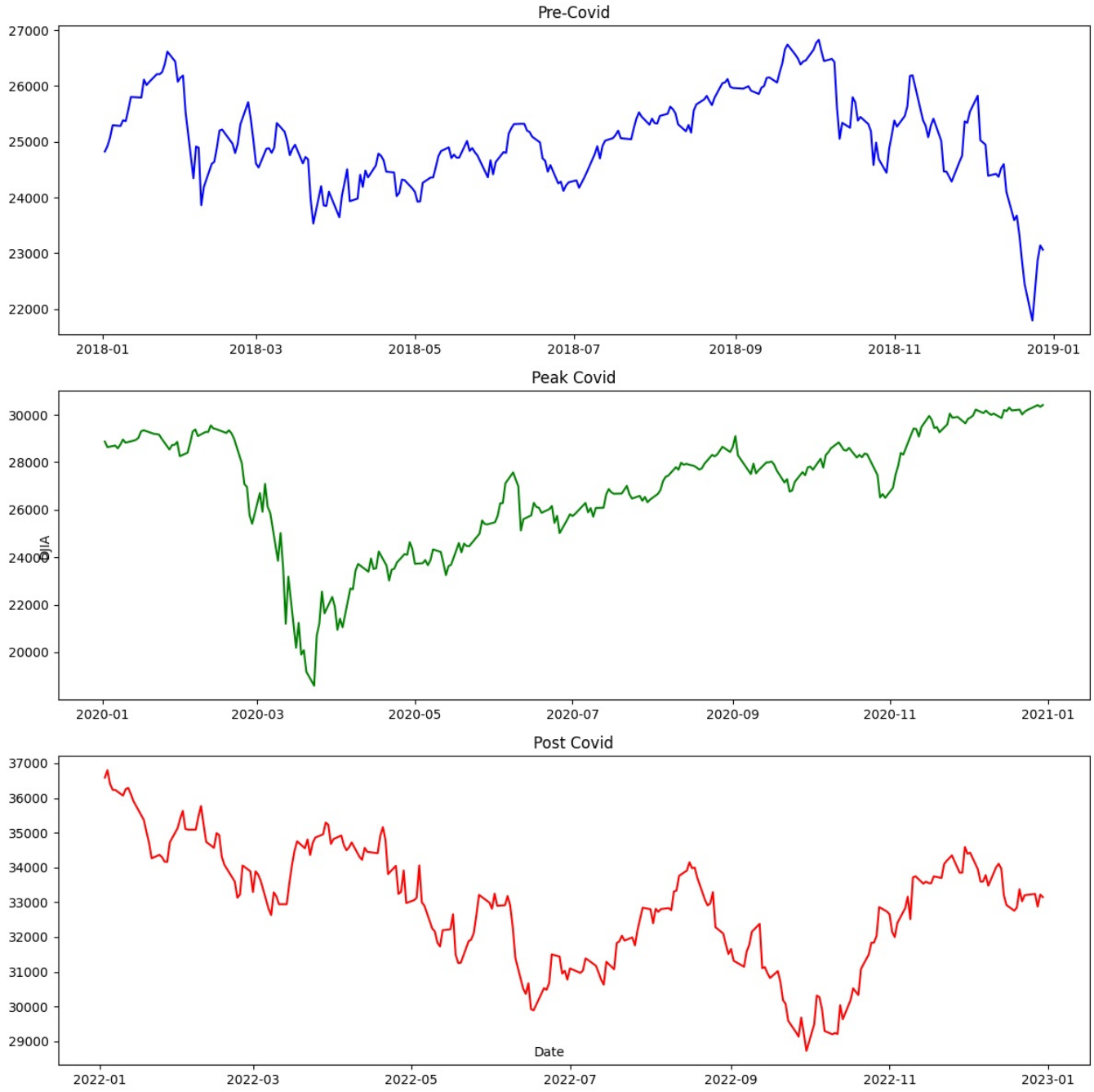
755 rows × 90 columns

Part 2: Graphs and Visualizations

2A. Line plots for the closing prices for Pre-Covid , Peak-Covid, and Post-Covid

By plotting the closing prices of the Dow Jones Industrial Average of every month of 2018, 2020, and 2022 we are able to see the stark differences between pre/post peak covid pandemic and peak pandemic. As we can see, in 2018, the highest price for the Dow Jones was almost 27,000 with its lowest point being at the end of the year at about 22,000. During 2020 (which contains the height of the pandemic), we can see that the peak of the market was just over 30,000 with the low being in March which was nearly 18,000. To put how poor the stock market was doing into perspective, the Dow Jones had its worst day since 1987 on March 15, 2020 dropping nearly 3,000 points in a single day. However, we are able to see the Dow Jones recover from that point as it steadily increased throughout the rest of the year. Finally, we are able to see interesting changes in the year 2022 which we deemed 'post-covid' as the peak fear of covid had mostly been eradicated in the eyes of businesses. While we see a decline throughout the year, the market was clearly not as volatile with the high being about 37,000 early in the year and the low being quite late in the year at about 28,000. While the DJIA did drop to 28,000 it was able to recover and stabilize around 34,000 by the end of the year.

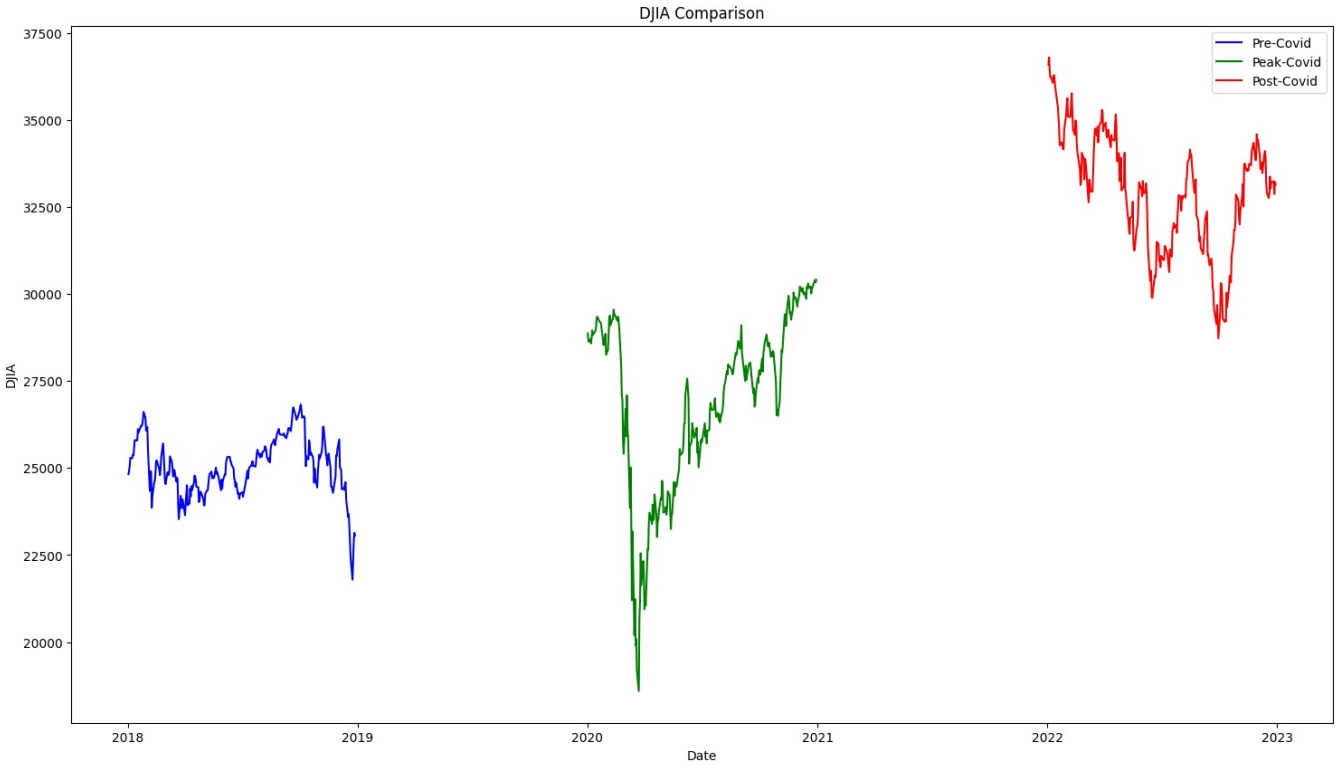
In [9]:
```
fig, axes = plt.subplots(nrows = 3, ncols = 1, figsize = (12, 12))
# Plot pre-COVID data
axes[0].plot(start_data['Close'], color = 'blue')
axes[0].set_title('Pre-Covid')

# Plot mid-COVID data
axes[1].plot(mid_data['Close'], color = 'green')
axes[1].set_title('Peak Covid')

# Plot current data
axes[2].plot(end_data['Close'], color = 'red')
axes[2].set_title('Post Covid')


fig.text(0.5, 0.04, 'Date', ha = 'center')
fig.text(0.04, 0.5, 'DJIA', va = 'center', rotation = 'vertical')

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



2B. Combined line plots for the closing prices for Pre-Covid , Peak-Covid, and Post-Covid

To gain a more thorough understanding of each individual year, we presented the graphs individually. However, to provide a meaningful comparison between all three years, we placed them all onto a single graph. By doing so, we were able to fully capture the extent of the stock market's decline in March 2020 and the surge that followed in 2022, ultimately surpassing the baseline market of 2018.

In [10]:
```
plt.figure(figsize = (18,10))

plt.plot(start_data["Close"], label = 'Pre-Covid', color='blue')

# Plot the second DJIA dataset in green color
plt.plot(mid_data["Close"], label = 'Peak-Covid', color='green')

# Plot the third DJIA dataset in red color
plt.plot(end_data["Close"], label = 'Post-Covid', color='red')


plt.xlabel('Date')
plt.ylabel('DJIA')
plt.title('DJIA Comparison')
plt.legend()
plt.show()
```



2C. Comparing closing prices for the top 3 indicies in the stock market

We plot the bar charts for the changes in NASDAQ and S&P500 compared to DJIA from 2018 to 2022. We continue to use the finance library to get the data for the two new indexes over the time period of 2018 to 2022.

In [11]:
```
#Here we are looking at the data from yahoo finance, grabbing data from djia, sp500, and nasdaq between 2018 an
djia = yf.download('^DJI', start = startDate, end = endDate)
sp500 = yf.download('^GSPC', start = startDate, end = endDate)
nasdaq = yf.download('^IXIC', start = startDate, end = endDate)

#We are looking at the close columns for each stock in order to retrieve the closing prices
#We select the column which contains the adjusted closing prices of the Dow Jones, S&P 500, and NASDAQ indices
#Then we are grouping the data by year by using the '.resample()' method and return the values necessary with t
#Then we are simply finding the percent change of each index using '.pct_change()'
djiaNew = djia['Adj Close'].resample('Y').last().pct_change()
sp500New = sp500['Adj Close'].resample('Y').last().pct_change()
nasdaqNew = nasdaq['Adj Close'].resample('Y').last().pct_change()
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```

The Covid-19 pandemic had a profound impact on the global economy, affecting every sector and industry. In the United States, the stock market was not immune to this impact, with the Dow Jones Industrial Average (DJIA) being one of the hardest hit. Comprised of the 30 largest publicly traded companies in the US, the DJIA is a widely recognized indicator of the stock market's overall performance.

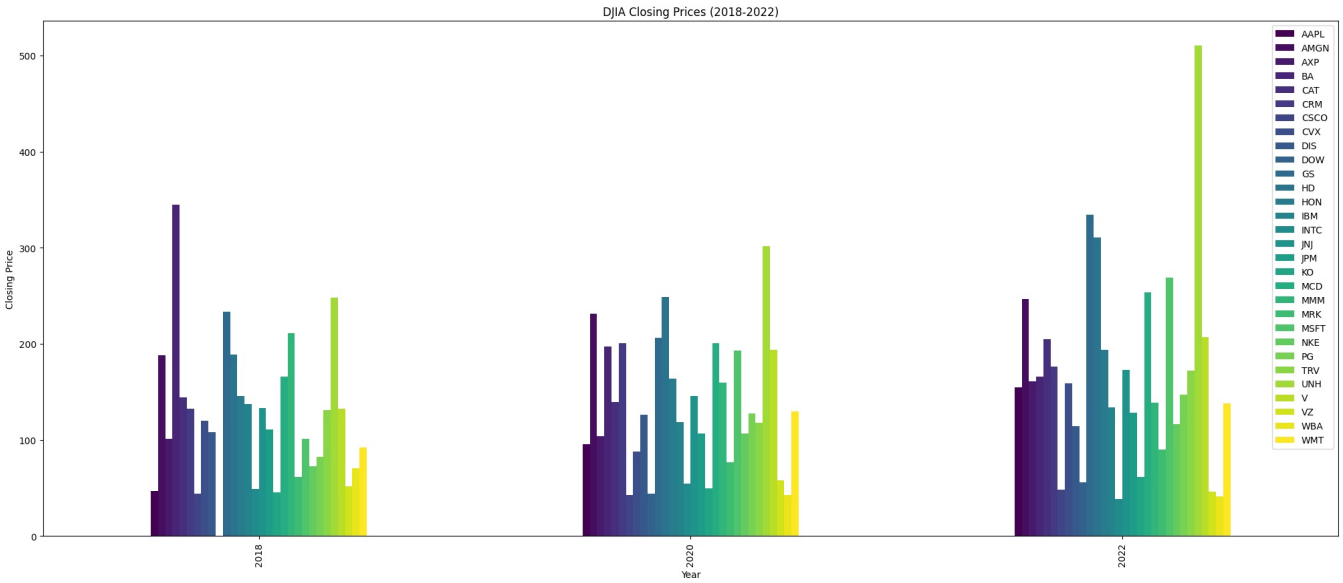
However, the DJIA is not the only index that tracks the performance of American companies. The S&P 500, which monitors the stock prices of 500 large-cap American stocks, is another widely followed index. Both the DJIA and S&P 500 offer a comprehensive view of the state of the stock market in general. On the other hand, the Nasdaq Composite Index is unique in that it only includes stocks that are traded on the Nasdaq market, and is completely electronic. Despite their differences, all three indexes saw significant drops in their annual returns in 2020 due to the Covid-19 pandemic. The Nasdaq Composite Index saw the greatest drop during this period, as it is heavily composed of technology companies that were particularly affected by the pandemic-induced economic recession.

In [12]:
```
# Here we are creating a bar chart to show the changes of each index in the given time period
#Storing data in data3 with x axis of the indices and y axis of the percent change of them
data3 = [go.Bar(x = ['DJIA', 'S&P 500', 'NASDAQ'], y = [djiaNew[-1], sp500New[-1], nasdaqNew[-1]])]

#here we are setting the title of the graph to: 'Index Returns from 2018-2022' and the title of the y-axis to:
lay = go.Layout(title = 'Index Returns from 2018-2022', yaxis = dict(title = 'Returns'))

#here we are creating a plot using data3
blueBar = go.Figure(data=data3, layout=lay)
blueBar.show()
```

2D. Closing prices for all 30 companies in the Dow Jones Industrial for each of the 3 years

We created this visualization to provide a concise summary of the fluctuation in each stock's closing price over the three-year period, allowing for easy comparison between different companies in the Dow Jones and showing the effect of individual stocks on the entire Industrial Average. This is important to notice as it displays the fluctuation of each company in the DJIA, thus explaining the fluctuation of the index as a whole. It is also interesting to see that the companies that showed an upward trend are related to the healthcare industry and describe the increase in dow jones right after the peak of covid as a lot of funds were put into vaccinations and curing people affected by the virus. The empty bar in 2018 is present as Dow Inc was formed in 2019 and added to DJIA that year, thus the closing price for the company for the year 2018 cannot be obtained as it did not exist at the time.

In [13]:
```
#Choosing all closing prices of each year from columns named 'close'
close = data2.xs('Close', level=1, axis=1)

# Grouping by average closing price per year for each company
closeMean = close.groupby(close.index.year).mean()

# Creating a color map with unique colors
colors = plt.cm.viridis(np.linspace(0, 1, len(close.columns)))

# Plotting the average closing price for each company in the form of a bar chart
closeMean.plot(kind='bar', figsize=(25, 10), color=colors)


# Adding necessary labels for plot.
plt.xlabel('Year')
plt.ylabel('Closing Price')
plt.title('DJIA Closing Prices (2018-2022)')
plt.show()

## DOW Inc became a company in 2019, thats why the 2018 section does not have a bar.
```



2E. Pie chart for the 11 sectors that make up the stock market

For the code in this part, we hardcode the values of each sector and use matplot library to show a pie chart. We have to hardcode the values for this as the percentages of the industry and grouping vary by website and since it is just 11 numbers, it is hard to find a dataset just with the values of the proportion of each sector accurately represented.

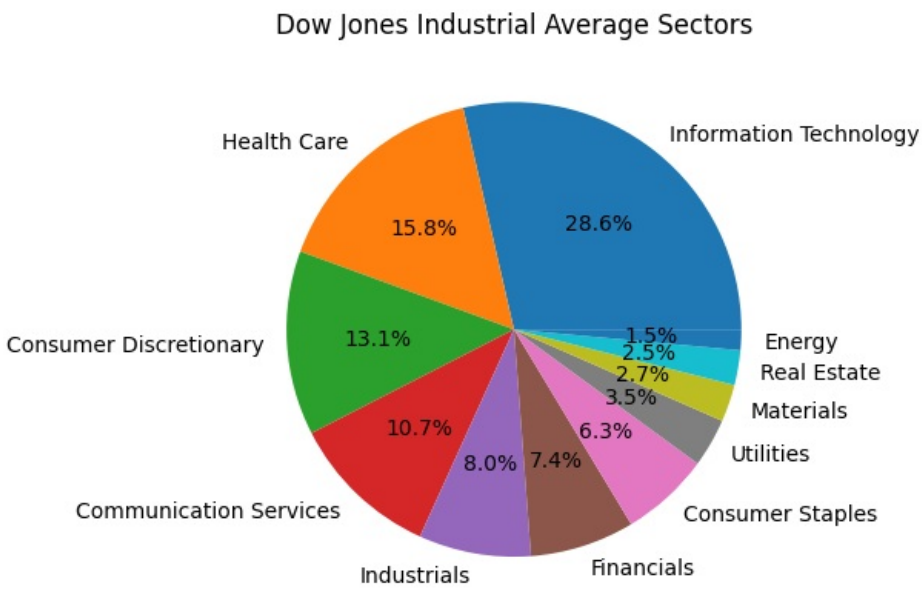
The stock market can be a complex concept for novice investors. However, it is important to understand the idea of stock sectors. Stock sectors refer to a group of publicly-traded companies that operate within the same general field of business, such as healthcare, energy, or real estate. These companies share similar characteristics, and understanding the different sectors can help investors diversify their portfolios and gain insights into how the market is categorized and can even help and guide new investors on which market/industry to invest in based on the returns for the past quarter in each sector.

Currently, the stock market includes 11 sectors, according to the widely used Global Industry Classification Standard (GICS). These sectors include information technology, healthcare, financials, consumer discretionary, consumer staples, energy, materials, industrials, real estate, utilities, and communication services. Each sector is unique, with its own set of challenges and opportunities, and investors can benefit from understanding the strengths and weaknesses of each sector.

Within the Dow Jones Industrial Average, information technology and healthcare are currently the most prominent sectors, making up 28.6% and 15.8% respectively. These sectors have seen significant growth in recent years, driven by advances in technology and an aging population that requires increasing healthcare services. Conversely, energy is currently the least prominent sector in the Dow Jones, making up just 1.5%. This is reflective of the ongoing shift away from traditional fossil fuels and towards renewable energy sources.

It is worth noting that while the stock market sectors can be useful in understanding market trends, they are not foolproof predictors of individual stock performance. Investors must also consider a range of other factors, such as company-specific financials, global economic conditions, and political events. Nonetheless, gaining a deeper understanding of stock market sectors can help investors make informed decisions and build a diversified portfolio that reflects their unique investment goals and risk tolerance.

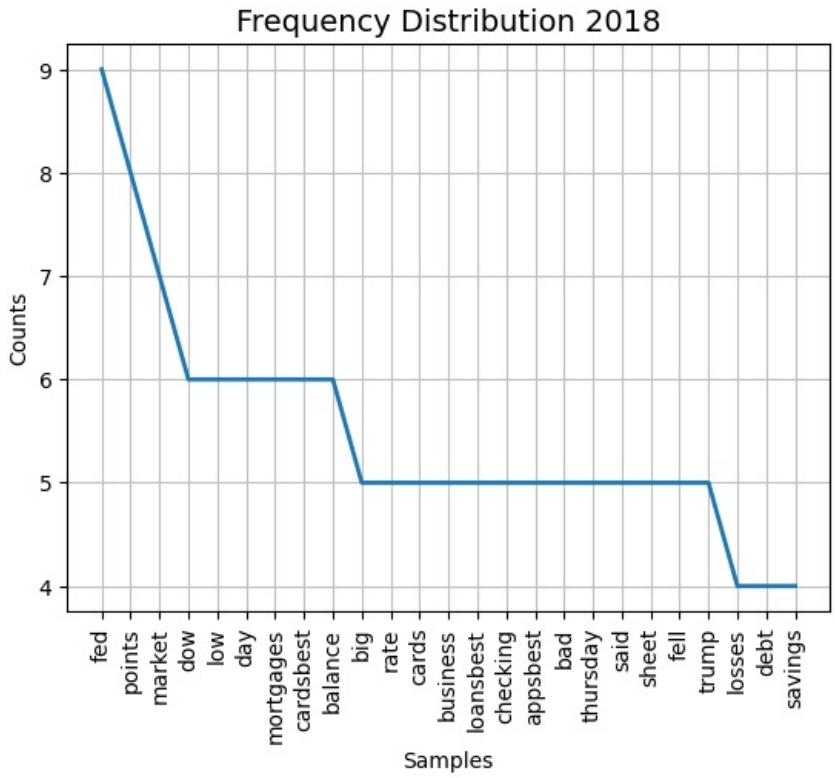
In [14]:
```
#Pie chart showing sector comp in djia


# Naming sectors and pairing them with percentages that they hold of the Dow Jones
sectors = ['Information Technology', 'Health Care', 'Consumer Discretionary', 'Communication Services', 'Indust
percentages = [27.17, 15.05, 12.46, 10.15, 7.58, 7.06, 6.02, 3.28, 2.52, 2.35, 1.39]


# Plotting a pie chart
plt.pie(percentages, labels=sectors,autopct='%1.1f%%')
# Adding a relevant title
plt.title('Dow Jones Industrial Average Sectors')
plt.show()

## As of August 31, 2020: three companies(Pfizer, Raytheon, Exxon) were replaced for companies (Salesforce.com,
```



Part 3: NLP Analysis for Stock Market news from CNBC

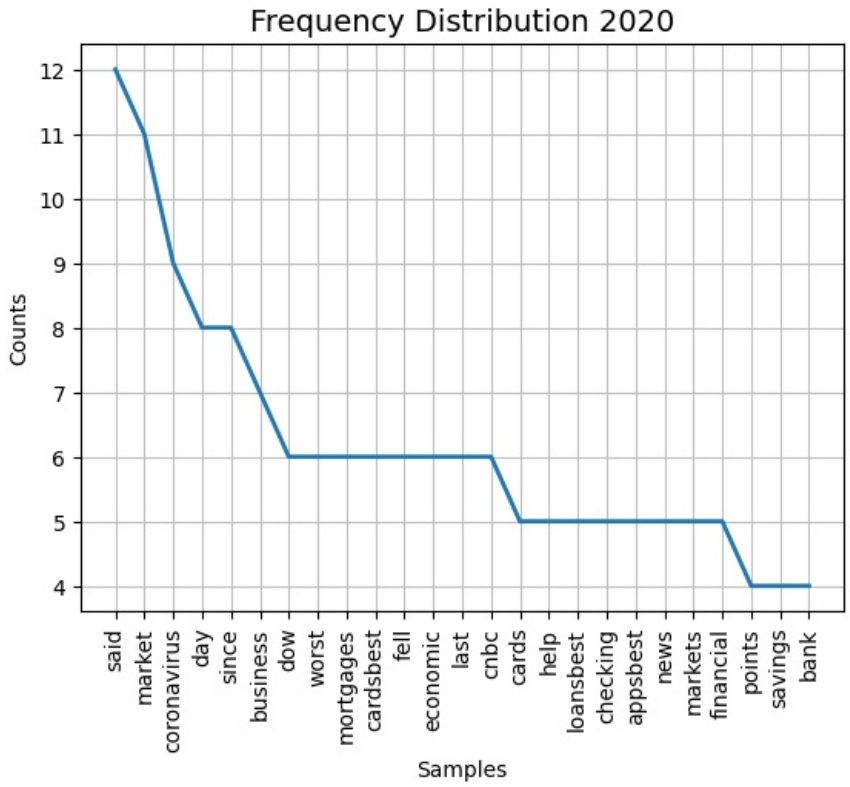
In [15]:

```
#Here we are importing a url and setting it equal to url2018
url2018 = "https://www.cnbc.com/2018/12/20/us-futures-following-wednesdays-sell-off.html"

#here we are retrieving the content of the url2018
request = requests.get(url2018)

#we are using BeautifulSoup to retrieve the text from html
beau = BeautifulSoup(request.content, 'html.parser')
text = beau.get_text()

#setting the text to lowercase
text = text.lower()
#removing all the punctuation
text = text.translate(str.maketrans('', '', string.punctuation))
#separating the text into individual words
text = nltk.word_tokenize(text)
#making stopwords
stopWords = set(stopwords.words("english"))

#here we are removing words that did not appear in the article but showed up in the data frequency distribution
removeWords = set(["accountsbest", "credit", "personal", "small", "loans"])
#creating a list of words while filtering out certain words such as the removeWords and stopWords
text = list(word for word in text if word.isalpha() and len(word) > 2 and word.casefold() not in stopWords and w
#create a frequency distribution using text
frequencyPlot = FreqDist(text)

#creating a title for the plot, and setting it to font size 14
plt.title('Frequency Distribution 2018', fontsize = 14)

#plotting 25 words
frequencyPlot.plot(25)
```



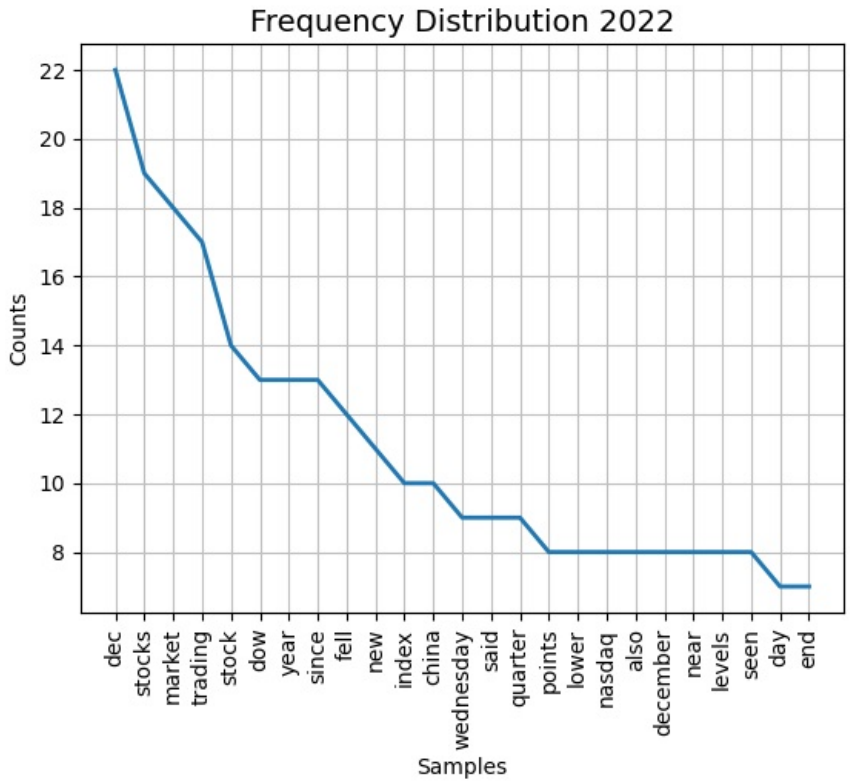
Out[15]: <Axes: title={'center': 'Frequency Distribution 2018'}, xlabel='Samples', ylabel='Counts'>

In [16]:
```
import requests
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from bs4 import BeautifulSoup
import string
import matplotlib.pyplot as plt

#Here we are importing a url and setting it equal to url2020
url2020 = "https://www.cnbc.com/2020/03/15/traders-await-futures-open-after-fed-cuts-rates-launches-easing-prog

#here we are retrieving the content of the url2020
request = requests.get(url2020)

#we are using BeautifulSoup to retrieve the text from html
beau = BeautifulSoup(request.content, 'html.parser')
text = beau.get_text()

#setting the text to lowercase
text = text.lower()
#removing all the punctuation
text = text.translate(str.maketrans('', '', string.punctuation))
#separating the text into individual words
text = nltk.word_tokenize(text)
#making stopwords
stopWords = set(stopwords.words("english"))

#here we are removing words that did not appear in the article but showed up in the data frequency distribution
removeWords = set(["accountsbest", "credit", "personal", "small", "loans"])
#creating a list of words while filtering out certain words such as the removeWords and stopWords
text = list(word for word in text if word.isalpha() and len(word) > 2 and word.casefold() not in stopWords and w
#create a frequency distribution using text
frequencyPlot = FreqDist(text)

#creating a title for the plot, and setting it to font size 14
plt.title('Frequency Distribution 2020', fontsize = 14)

#plotting 25 words
frequencyPlot.plot(25)
```



Out[16]: <Axes: title={'center': 'Frequency Distribution 2020'}, xlabel='Samples', ylabel='Counts'>

In [17]:
```
#Here we are importing a url and setting it equal to url2022
url2022 = "https://www.cnbc.com/2022/12/27/stock-market-futures-open-to-close-news.html"

#here we are retrieving the content of the url2022
request = requests.get(url2022)

#we are using BeautifulSoup to retrieve the text from html
beau = BeautifulSoup(request.content, 'html.parser')
text = beau.get_text()

#setting the text to lowercase
text = text.lower()
#removing all the punctuation
text = text.translate(str.maketrans('', '', string.punctuation))
#separating the text into individual words
text = nltk.word_tokenize(text)
#making stopwords
stopWords = set(stopwords.words("english"))

#here we are removing words that did not appear in the article but showed up in the data frequency distribution
removeWords = set(["accountsbest", "credit", "personal", "small", "loans"])
#creating a list of words while filtering out certain words such as the removeWords and stopWords
text = list(word for word in text if word.isalpha() and len(word) > 2 and word.casefold() not in stopWords and w
#create a frequency distribution using text
frequencyPlot = FreqDist(text)

#creating a title for the plot, and setting it to font size 14
plt.title('Frequency Distribution 2022', fontsize = 14)

#plotting 25 words
frequencyPlot.plot(25)
```



Out[17]: <Axes: title={'center': 'Frequency Distribution 2022'}, xlabel='Samples', ylabel='Counts'>

When looking at the frequency distributions from 2018, 2020 and 2022 there are several things to take away. We used three articles from CNBC relating to Dow Jones. We wanted to find out what differences there would be in terms of some of the words in the three frequency distributions.

In the 2018 article, the most frequent words were fed, points and market. It is expected that the Fed (Federal Reserve System) would show up frequently in the article as the article is about a fed rate hike that took place. Points is also in the title of the article as it explains how Dow Jones dropped 470 points. Lastly, market is a word that is expected to appear frequently in almost any article that covers stocks such as this one.

In the 2020 article, the most frequent words were said, market and coronavirus. The word said showed up frequently as the article included multiple quotes from people familiar with the stock market. They were mainly quotes centered around their thoughts on how covid will affect the stock market. As in the 2018 article, the word market showed up frequently as the article discussed how the rise of covid had affected the stock market. Lastly, coronavirus is a word that was expected to appear frequently in the article as it was published on March 15, 2020.

In the 2022 article, the most frequent words were stocks, trading and market. The word stock is expected to appear frequently in any article relating to the stock market including this one. The word trading appeared frequently as the article discussed the trading levels of several companies. Lastly, the word market was one of the most common words as in the 2018 and 2020 article.

When looking at some of the words in the frequency distribution, there are some words that make it more evident when the article was written. One of the words in the 2018 frequency distribution is trump, which makes it more likely that the article was written during Trump's presidency. One of the words in the 2020 frequency distribution is coronavirus, which makes it obvious that this article was written during or after the coronavirus pandemic. One of the words in the 2022 frequency distribution is december, which makes it more likely that the article was written in the month of december. Some of the words that appeared in the frequency distribution at first weren't in the article, so we used exclude_words to get rid of those words from the distribution. These words were likely a part of the HTML tags or metadata of the webpage which is why we used exclude_words to filter them out.

Part 4: API key for Unemployment rates throughout Covid-19 Pandemic

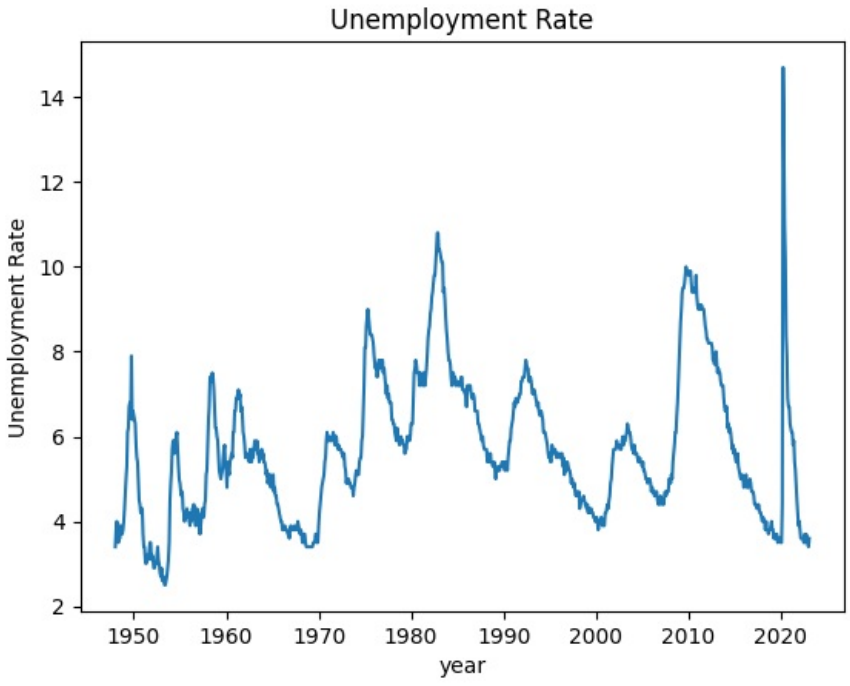
In [18]:
```
!pip install fredapi
```

```
Requirement already satisfied: fredapi in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-
packages (0.5.0)
Requirement already satisfied: pandas in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-p
ackages (from fredapi) (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/
python3.9/site-packages (from pandas->fredapi) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/
site-packages (from pandas->fredapi) (2022.7)
Requirement already satisfied: numpy>=1.20.3 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9
/site-packages (from pandas->fredapi) (1.24.2)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site
-packages (from python-dateutil>=2.8.1->pandas->fredapi) (1.16.0)
```

In [19]:

```
#the following two lines are required for macos only. They add the capacity to check #certificates and verify t
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

from fredapi import Fred

#get the api key from fred website.
fred = Fred(api_key='4df6af011a2d28b7e0e811da718ed7e0')
#get the unemployment rate from the data for all years.
unemployment = fred.get_series('UNRATE', start_date='2018-01-01', end_date='2022-01-01')
#since the unemployment rates are not given for each day, we get all the data from 1900s to #end date. We get i
print(unemployment.head)


import matplotlib.pyplot as plt
#plotting the unemployment rates for each year
plt.plot(unemployment)
plt.title('Unemployment Rate')
plt.xlabel('year')
plt.ylabel('Unemployment Rate')
#print the plot
plt.show()
#setting parameter of graph to increase graph size to analyze it better.
plt.rcParams["figure.figsize"] = (30,15)
```

```
<bound method NDFrame.head of 1948-01-01    3.4
1948-02-01    3.8
1948-03-01    4.0
1948-04-01    3.9
1948-05-01    3.5
              ...
2022-10-01    3.7
2022-11-01    3.6
2022-12-01    3.5
2023-01-01    3.4
2023-02-01    3.6
Length: 902, dtype: float64>
```



To code unemployment rates, we make use of API accessing with a key to extract the data. Fred is an online library/resource that contains datasets and resources related to the economy and stock market. We get the reference key from https://fred.stlouisfed.org/docs/api/fred/series.html And use the data retrieved from it to create a data frame with the DJIA stock price for the year. We then plot the unemployment rate over the last few years and compare it to the graphs of DJIA over the three years being compared. Since there is no precise dataset available that can show us just the unemployment rate over the three given years

accurately, we just see the general trend over the last 70 years. This also gives us a better understanding of how drastic the unemployment rate increase in 2020 really was.

The Dow Jones Industrial Average (DJIA) performance can provide valuable insights into the broader economy. The stock prices of the 30 companies included in the DJIA are closely tied to economic conditions and can reflect changes in consumer demand, corporate earnings, and overall economic activity. Significant changes in the DJIA can signal shifts in the labor market and the unemployment rate.

An example of this can be seen in the graphs. in 2020(April), when the United States experienced a sharp increase in unemployment rates due to the COVID-19 pandemic. As businesses shuttered and stay-at-home orders were put in place, the stock market saw significant declines, and the DJIA experienced its largest single-day drop in history. This was indicative of the significant impact the pandemic was having on the economy, with widespread job losses and an acute decline in economic activity. While the correlation between the DJIA and unemployment rates is not perfect, as the stock market can sometimes anticipate changes in economic conditions, the two are closely linked. As the economy slows, the DJIA tends to fall, and this can lead to increased unemployment rates. Similarly, as the economy recovers and the DJIA rebounds, job growth tends to follow.

Part 6: Conclusion

In conclusion, through data manipulation, visualizations, APIs, and NLP analysis we are clearly able to see that the Covid-19 pandemic severely affected the Dow Jones Industrial Average and in turn the stock market as a whole.

Out of this report, we would like to show that the stock market should not be affected by unforeseen incidents in the world. Yes, the market had a huge dip because of uncertainty, but we are able to see how fast it is able to recover from something as big of a crisis as the Covid-19 pandemic. We hope that in the future, people will see that the market corrects itself so that the economy does not fall drastically in cases such as the pandemic that peaked in 2020.

References

“Data to Fish.” Data to Fish, https://datatofish.com/bar-chart-python-matplotlib/.

“Data to Fish.” Data to Fish, https://datatofish.com/line-chart-python-matplotlib/.

Foimbert. “Dow Drops Nearly 3,000 Points, as Coronavirus Collapse Continues; Worst Day since '87.” CNBC, CNBC, 16 Mar. 2020, https://www.cnbc.com/2020/03/15/traders-await-futures-open-after-fed-cuts-rates-launches-easing-program.html.

Kolakowski, Mark. “A Brief History of Bear Markets.” Investopedia, Investopedia, 23 Sept. 2022, https://www.investopedia.com/a-history-of-bear-markets-4582652.

Matplotlib Pie Charts, https://www.w3schools.com/python/matplotlib_pie_charts.asp.

Tanayamac. “Dow Closes More than 350 Points Lower in Broad Selloff, Apple Tumbles 3%.” CNBC, CNBC, 28 Dec. 2022, https://www.cnbc.com/2022/12/27/stock-market-futures-open-to-close-news.html.

Tomwfranck. “Dow Drops 470 Points to 14-Month Low in Second Day of Big Losses Following Fed Rate Hike.” CNBC, CNBC, 20 Dec. 2018, https://www.cnbc.com/2018/12/20/us-futures-following-wednesdays-sell-off.html.

“Yfinance.” PyPI, https://pypi.org/project/yfinance/.