Stock Market Analysis

In this project, we will be seeing Data Analysis and Visualization of three Stocks namely **Tata Motors**, **Maruti Suzuki**, and **Mahindra and Mahindra**. We acquired the dataset for past one year ranging from 20-Nov-2023 to 14-Nov-2024. There are total 3 files for three different companies under analysis in csv format. The data was downloaded from the NSE website directly. Each file has 14 attributes and we will use these to get basic idea about a stock's pricing, their correlational analysis, etc. by plotting them in different forms of graphs. Plotting these attributes against each other for different selected companies, we will gain many valuable insights about the on-going trends and patterns. The first step will be data pre-processing and then we will move forward with visualization and uncovering patterns!

Required Libraries

```
import pandas as pd
import os
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
from ta.trend import MACD, ADXIndicator
from ta.momentum import RSIIndicator, StochasticOscillator
from ta.volatility import BollingerBands
from datetime import datetime
```

Data Viewing and Pre-Processing

- Observe that in which data type are all the columns' data given, so that we can change the string numeric values to the suitable data type which can be plotted by the libraries.
- Here, all the columns are in `object` data type.

```
csv_files = ['MandM.csv', 'Maruti.csv', 'Tata.csv']
for file in csv_files:
    print(f"Data types for columns in {file}:")
    df = pd.read_csv(file)
```

```python
    print(df.dtypes)
    print("\n")
```

Output

```
PREV. CLOSE     object
ltp             object
close           object
vwap            object
52W H           object
52W L           object
VOLUME          object
VALUE           object
No of trades    object
dtype: object


Data types for columns in Tata.csv:
Date             object
series           object
OPEN             object
HIGH             object
LOW              object
PREV. CLOSE      object
ltp              object
close            object
vwap             object
52W H            object
52W L           float64
VOLUME           object
VALUE            object
No of trades     object
dtype: object
```

- Now, analyze the data values for figuring out whether any cleaning of data is required or not. Here, analyzing only a single file is also possible as they all have the same format.

```
file_path = 'Tata.csv'
data = pd.read_csv(file_path)
print(data.head())
```

Output

```
          Date series    OPEN    HIGH     LOW PREV. CLOSE     ltp   close  \
0  14-Nov-2024     EQ  786.60  792.00  772.00      786.25  776.00  774.30
1  13-Nov-2024     EQ  787.00  792.65  775.55      784.85  786.85  786.25
2  12-Nov-2024     EQ  806.00  813.10  783.05      804.70  784.75  784.85
3  11-Nov-2024     EQ  801.00  831.45  792.00      805.45  805.00  804.70
4  08-Nov-2024     EQ  821.95  822.00  801.10      819.75  803.55  805.45

     vwap     52W H  52W L       VOLUME              VALUE No of trades
0  779.43  1,179.00  649.3  1,17,40,909   9,15,12,58,844.20     3,13,693
1  785.62  1,179.00  649.3  1,46,74,022  11,52,82,10,756.95     3,91,032
2  792.60  1,179.00  649.3  1,65,26,921  13,09,92,59,115.85     5,94,079
3  816.67  1,179.00  649.3  2,75,87,619  22,52,99,43,363.60     5,75,463
4  807.29  1,179.00  641.9  1,60,72,692  12,97,52,78,301.60     4,78,152
```

- On observing the numeric values, it is evident that the numeric values have commas which can create an issue while plotting, so we will remove all the commas from the numeric values and also change their data type to `float`.

```
# Comma removal
input_files = ['MandM.csv', 'Maruti.csv', 'Tata.csv']
output_directory = 'cleaned_files/'

if not os.path.exists(output_directory):
    os.makedirs(output_directory)

for file in input_files:
    df = pd.read_csv(file, dtype=str)
    df = df.map(lambda x: x.replace(',', '') if isinstance(x, str) else x)
    output_path = os.path.join(output_directory, file)
    df.to_csv(output_path, index=False)
    print(f"Processed {file} and saved to {output_path}.")
```

Output

```
Processed MandM.csv and saved to cleaned_files/MandM.csv.
Processed Maruti.csv and saved to cleaned_files/Maruti.csv.
Processed Tata.csv and saved to cleaned_files/Tata.csv.
```

```python
file_paths = {
    "Mahindra": r"cleaned_files\MandM.csv",
    "Maruti": r"cleaned_files\Maruti.csv",
    "Tata": r"cleaned_files\Tata.csv"
}
```

```python
#Data type modification

numeric_columns = ["OPEN", "HIGH", "LOW", "PREV. CLOSE", "ltp", "close", 
"vwap",
                   "52W H", "52W L", "VOLUME", "VALUE", "No of trades"]

for company, file_path in file_paths.items():
    print(f"Processing file for {company}: {file_path}")
    data = pd.read_csv(file_path)

    for column in numeric_columns:
        if column in data.columns:
            try:
                data[column] = pd.to_numeric(data[column], errors='coerce')
            except Exception as e:
                print(f"Error converting column {column} in {file_path}: {e}")

    data.to_csv(file_path, index=False)
    print(f"File for {company} updated successfully: {file_path}\n")
```

Ouput

```
Processing file for Mahindra: cleaned_files\MandM.csv
File for Mahindra updated successfully: cleaned_files\MandM.csv

Processing file for Maruti: cleaned_files\Maruti.csv
File for Maruti updated successfully: cleaned_files\Maruti.csv

Processing file for Tata: cleaned_files\Tata.csv
File for Tata updated successfully: cleaned_files\Tata.csv
```

Analysis and Visualization

- To make the colour theme uniform throughout, we can use:

```
color_map = {
    'Mahindra': '#DD2A7B',
    'Tata': '#8134AF',
    'Maruti': '#F58529'
}
```

Statistical Analysis

1. Candlestick plot

> Explanation:

- A **candlestick chart** is a type of financial chart used to represent the price movements of an asset (such as a stock) over time. It is widely used in technical analysis for visualizing and interpreting price data, especially in stock trading. Each "candlestick" on the chart represents a specific time period (e.g., a day, a week) and provides information about four key price metrics: Open , High , Low and Close.
- Each candlestick consists of two parts:
  - **Body**: The range between the open and close prices. A green body indicates a rise (bullish), and a red body indicates a fall (bearish).
  - **Wicks/Shadows**: The lines above and below the body represent the high and low prices during the period.

```
def create_candlestick_chart(data, title):
    figure = go.Figure(data=[go.Candlestick(
        x=data["Date"],
        open=data["OPEN"],
        high=data["HIGH"],
        low=data["LOW"],
        close=data["close"]
    )])

    figure.update_layout(
        title=title,
        xaxis_title="Date",
        yaxis_title="Stock Price",
        xaxis_rangeslider_visible=False,
    )
    return figure
```

```
mandm_data = pd.read_csv(file_paths["Mahindra"], parse_dates=["Date"])
tata_data = pd.read_csv(file_paths["Tata"], parse_dates=["Date"])
maruti_data = pd.read_csv(file_paths["Maruti"], parse_dates=["Date"])

mandm_chart = create_candlestick_chart(mandm_data, "Candlestick Chart for
Mahindra and Mahindra")
tata_chart = create_candlestick_chart(tata_data, "Candlestick Chart for Tata
Motors")
maruti_chart = create_candlestick_chart(maruti_data, "Candlestick Chart for
Maruti Suzuki")

mandm_chart.show()
tata_chart.show()
maruti_chart.show()
```

Output

Observation

Mahindra and Mahindra:

- Trend Analysis:

  The stock shows a strong upward trend from January to July 2024, indicating positive growth. After hitting a peak around July-August, the trend appears to stabilize with some sideways movement before a slight decline in November.
- Volatility:

  The candlestick sizes vary, indicating moderate volatility during the year. The sharp movements around July reflect a possible reaction to market news or events.
- Sentiment: Neutral to Slightly Bullish

  Strong growth in the first half of the year with a slight pullback later, reflecting cautious optimism.

  Showed the most stable growth pattern, with strong performance during the first half and minimal decline later.

Tata Motors:

- Trend Analysis:

  A strong uptrend is visible from January to June 2024, where the stock price moves from approximately 700 to over 1,200. After peaking mid-year, the stock starts a consistent downtrend, especially from September onward.

- Volatility:

  Sharp fluctuations can be observed around the mid-year, indicating high volatility likely due to significant market events or announcements.
- Sentiments: Bearish

  Started the year strong but faced significant declines in the second half, indicating growing pessimism.

  Experienced the steepest decline post-mid-year, indicating significant corrections or bearish trends after a strong rally.

Maruti Suzuki:

- Trend Analysis:

  A mild upward trend is seen from January to March 2024, with the stock stabilizing around the mid-year. However, a noticeable decline begins in September and continues through November.
- Volatility:

  The chart shows relatively smaller candlestick sizes compared to the other stocks, indicating lower volatility for most of the year. However, there are brief periods of sharp price movements, especially around July and September.
- Maruti Suzuki: Bearish

  Mild growth early in the year followed by a sharp decline after September, showing weakening investor confidence.

  Displayed moderate movement with lower volatility, but the downtrend in the latter half suggests weakening investor confidence.

---

2. Rangeslider plot + Stock performance since past one year

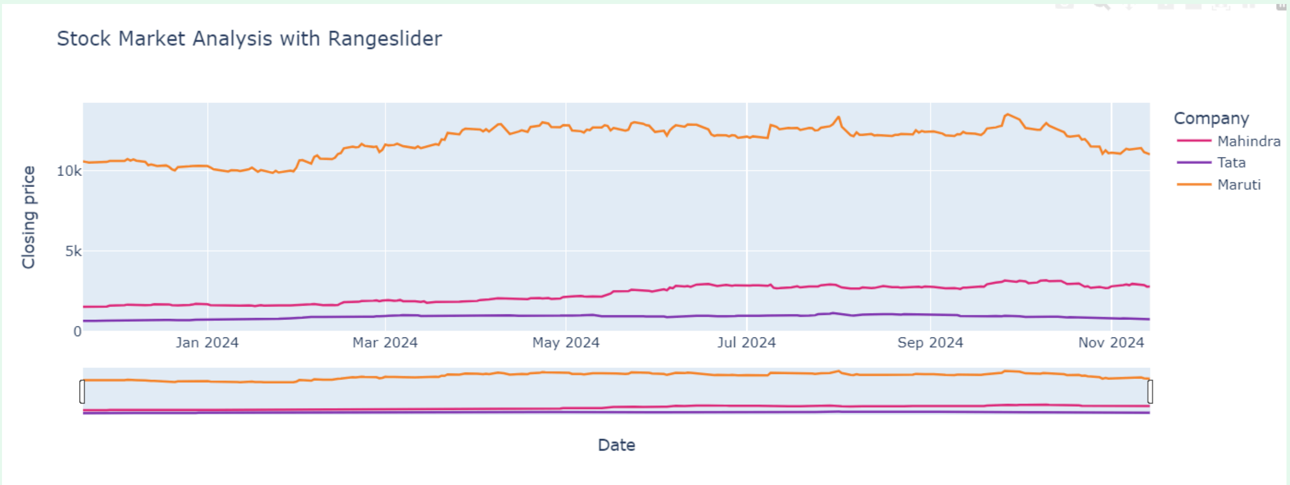
> Explanation:

- A **rangeslider graph** is an interactive chart used to visualize data over time, allowing users to explore and zoom in on specific periods. In the context of stock market analysis, it displays the **closing prices** of stocks for different companies over time, with the ability to adjust the visible time range using a slider at the bottom.

- Key features:
  - **X-axis**: Represents the dates over time.
  - **Y-axis**: Represents the closing prices of stocks.
  - **Rangeslider**: A slider below the chart allows users to zoom in and adjust the time range for closer analysis.

```
def load_and_prepare_data(filename, company_name):
    data = pd.read_csv(filename, parse_dates=['Date'])
    data.dropna(subset=["close"], inplace=True)
    data['Company'] = company_name
    return data

mandm_data = load_and_prepare_data(file_paths["Mahindra"], 'Mahindra')
tata_data = load_and_prepare_data(file_paths["Tata"], 'Tata')
maruti_data = load_and_prepare_data(file_paths["Maruti"], 'Maruti')

combined_data = pd.concat([mandm_data, tata_data, maruti_data],
ignore_index=True)

figure = px.line(combined_data, x='Date', y='close', color='Company',
                 title='Stock Market Analysis with Rangeslider',
                 color_discrete_map=color_map)

figure.update_layout(
    xaxis_rangeslider_visible=True,
    xaxis_title="Date",
    yaxis_title="Closing price",
)

figure.show()
```

Output



```
# Preparing the data for furthur plotting
df_list = []

for ticker, path in file_paths.items():
    data = pd.read_csv(path)
    data['Date'] = pd.to_datetime(data['Date'], format='%d-%b-%Y')
    data['Ticker'] = ticker  # Adding a Ticker column to distinguish companies
    df_list.append(data)

df = pd.concat(df_list)
```

---

3. Volume Plot

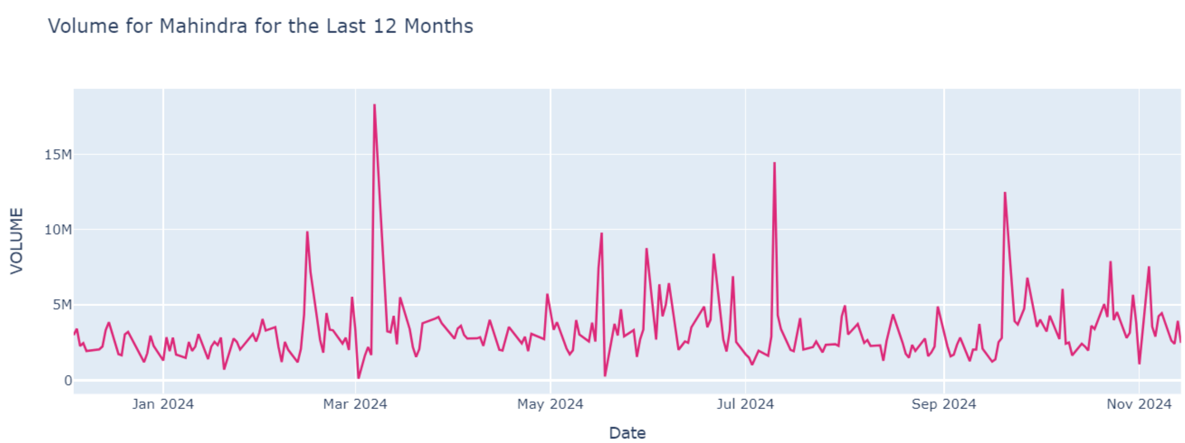
> Explanation:

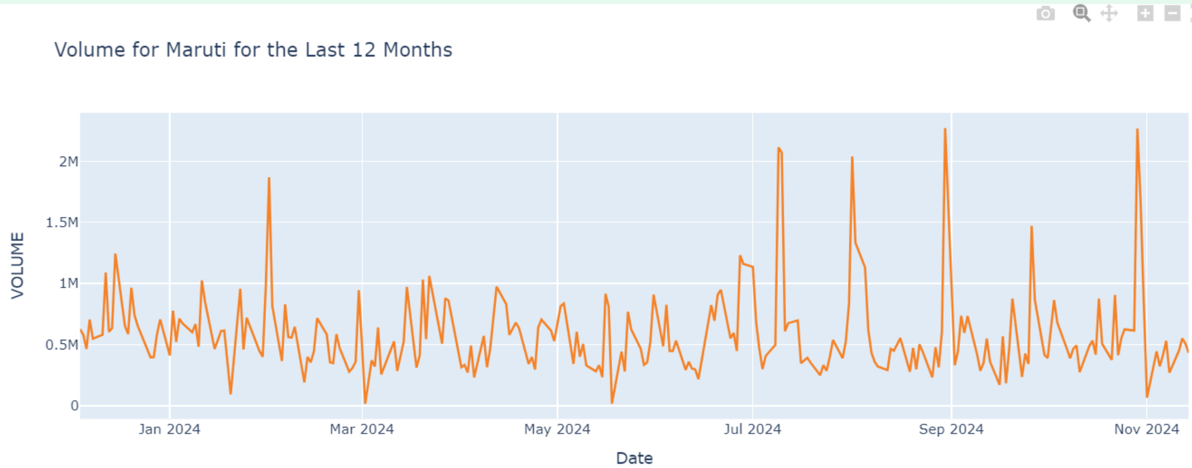
- The **volume plot** visualizes the trading volume of stocks over time, showing how much of a stock was traded during each period (e.g., daily, weekly). In this specific code:
  - **X-axis**: Represents the dates over the last 12 months.
  - **Y-axis**: Represents the trading volume, indicating the number of shares traded for each company on each day.

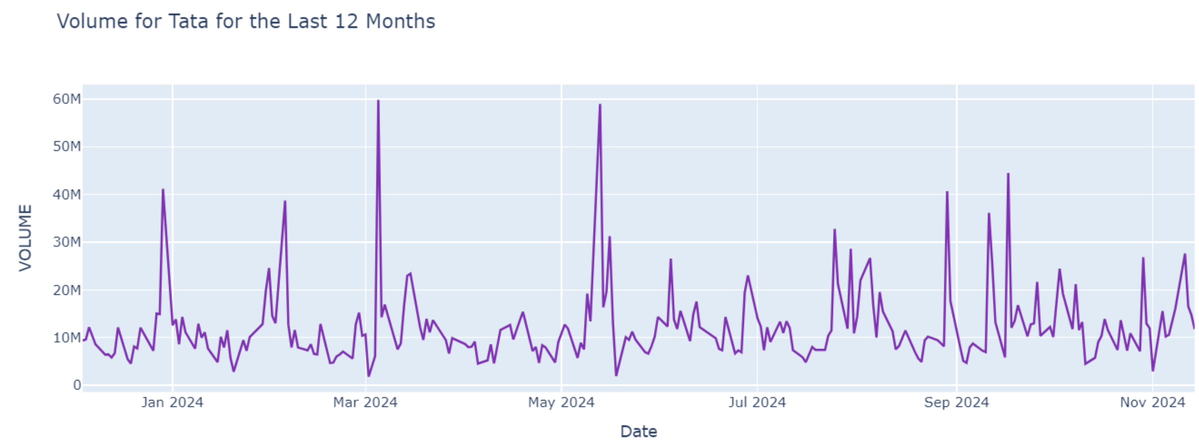
```
# Filter for the last 12 months (optional, based on your dataset)
start_date = datetime.now() - pd.DateOffset(months=12)
end_date = datetime.now()
```

```python
df = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]

# Plot volume for each company individually
for ticker in file_paths.keys():
    company_data = df[df['Ticker'] == ticker]
    fig = px.line(company_data, x='Date', y='VOLUME',
                  title=f"Volume for {ticker} for the Last 12 Months")
    fig.update_traces(line_color=color_map[ticker])
    fig.show()
```

Output







Observation

- The plots will display the changes in trading volumes for each company over the past 12 months.
- Peaks in the volume indicate periods of high investor activity, potentially due to news, announcements, or market sentiment shifts.
- Troughs represent low trading activity, often occurring during stable or uneventful periods.

Mahindra:

- Likely exhibits consistent trading volumes, with occasional spikes indicating heightened interest or reactions to key events.
- A pattern of steady volume could suggest it is less speculative and more stable in the market and consistent investor interest.

Tata:

- Higher spikes in volume might suggest that the company frequently experiences market-moving news or is a target for active trading and subject to market speculation.
- Periodic surges may align with major product launches, financial reports, or other corporate events.

Maruti:

- Volume trends might show seasonality, especially if influenced by the automotive market cycles or consumer sentiment around festivals.
- Spikes could indicate significant investor activity following industry-specific developments, such as shifts in demand or regulatory changes.
- Maruti might show mixed trends, with periods of stability interrupted by significant surges during industry events.

Seasonal Patterns:

- If the volume increases at specific times of the year (e.g., the start or end of the fiscal year), it could suggest institutional trading or portfolio rebalancing by investors.
- Mahindra likely exhibits stability, Tata shows high volatility, and Maruti has mixed behavior reflecting industry dynamics.

---

4. Button Plot

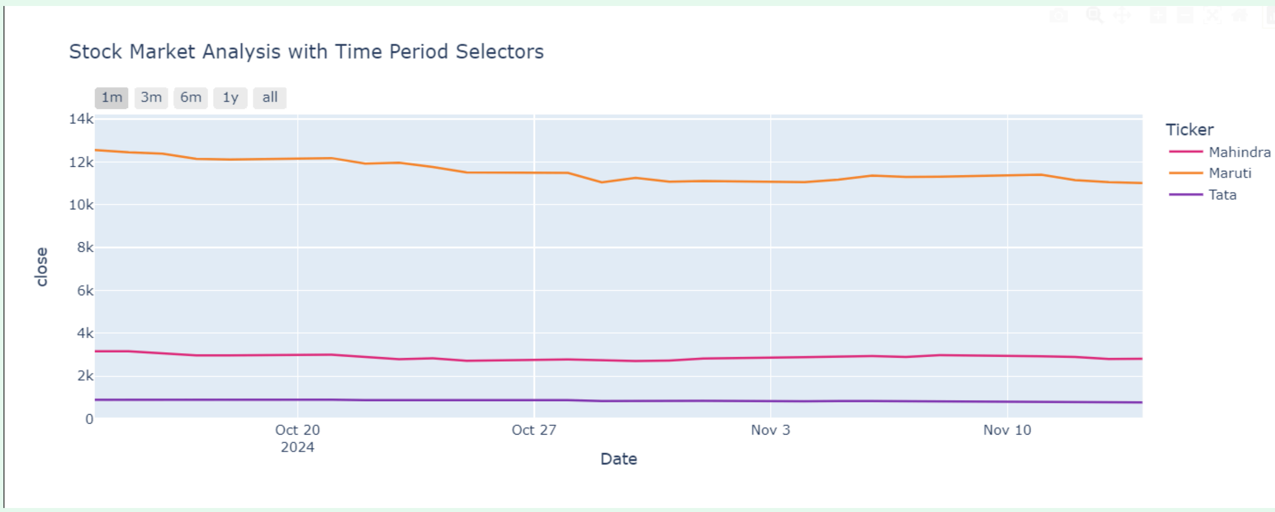
- This interactive chart lets users zoom in on specific timeframes, providing a dynamic view of stock performance. It is useful for examining stock trends over different periods, helping investors and analysts make more informed decisions based on historical data.

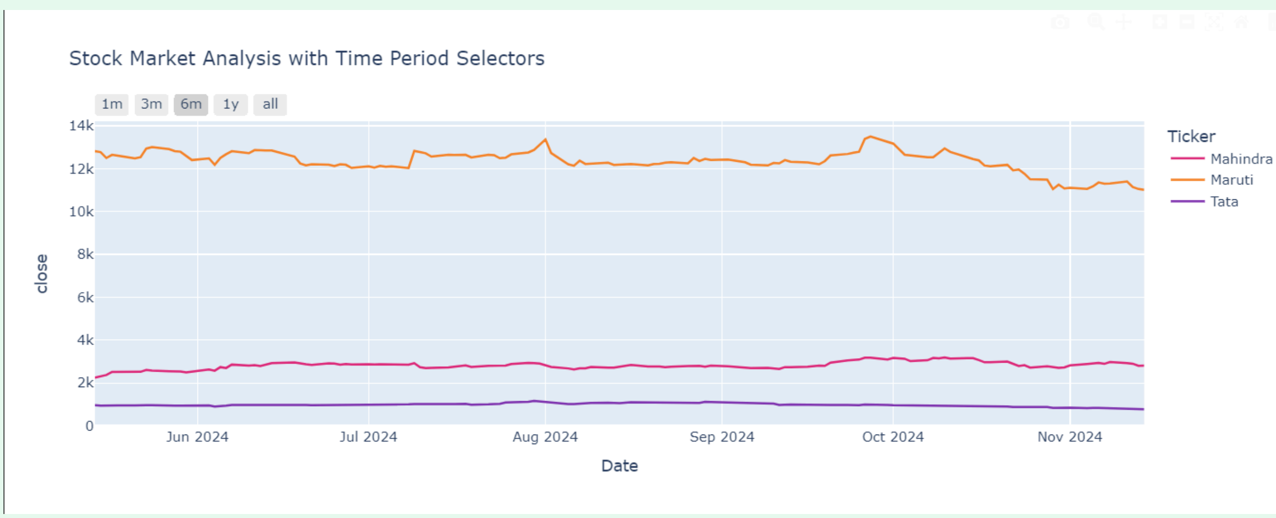
  - **X-axis**: Represents the dates of stock prices.
- **Y-axis**: Represents the closing prices of the stocks.

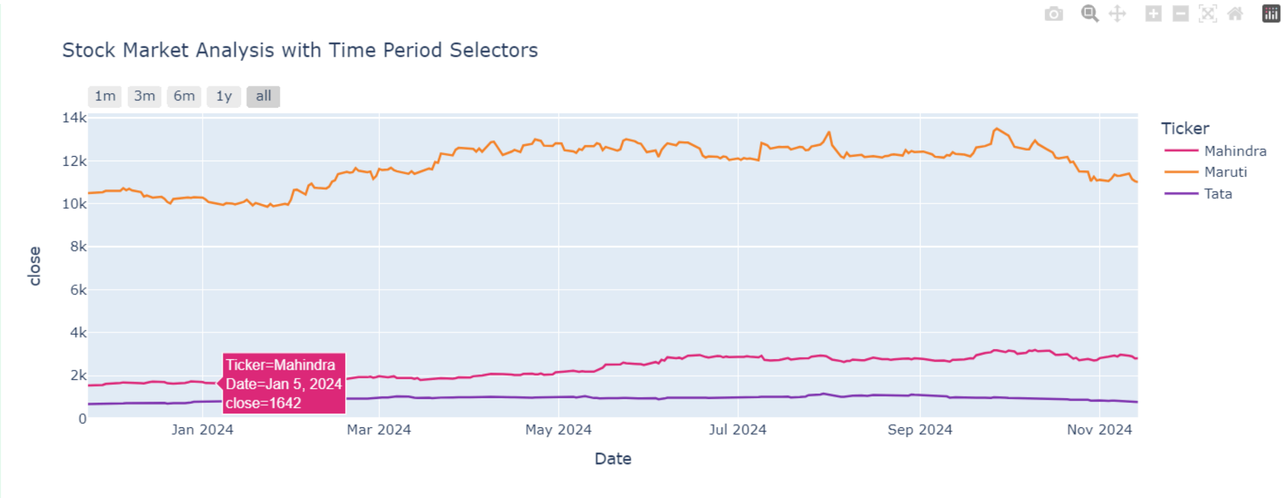
```
figure = px.line(df,
                 x='Date',
                 y='close',
                 color='Ticker',
                 title='Stock Market Analysis with Time Period Selectors',
                 color_discrete_map=color_map)

figure.update_xaxes(
    rangeselector=dict(
        buttons=list([
```

```
            dict(count=1, label="1m", step="month", stepmode="backward"),
            dict(count=3, label="3m", step="month", stepmode="backward"),
            dict(count=6, label="6m", step="month", stepmode="backward"),
            dict(count=1, label="1y", step="year", stepmode="backward"),
            dict(step="all")
        ])
    )
)

figure.show()
```

Ouptut







---

5. Scatter Matrix Plotting

> Explanation:

- The scatter matrix plot allows users to explore and analyze the correlations between multiple stock features for different companies. It provides insights into how the features interact, helping to identify trends, outliers, and potential relationships between the stock variables, such as whether higher `OPEN` prices correlate with higher `VOLUME`, or if there is any correlation between `HIGH` and `close` prices.

```
columns_to_plot = ['OPEN', 'HIGH', 'LOW', 'PREV. CLOSE', 'close', 'VOLUME']

numeric_data = df[columns_to_plot + ['Ticker']]

scatter_matrix = px.scatter_matrix(
    numeric_data,
    dimensions=columns_to_plot,
    color='Ticker',
    title="Scatter Matrix of Stock Data by Company",
    color_discrete_map=color_map,
    labels={col: col for col in columns_to_plot},
    height=800,
    width=800
)

scatter_matrix.show()
```

Output

6. Comparing Daily Returns between Stocks

> Definition:

- Daily return represents the percentage change in a stock's closing price from one trading day to the next. It is a measure of the daily performance of a stock, indicating whether its price increased or decreased and by what percentage.

> Formula:

For a given stock t on day t, the daily return is calculated as:

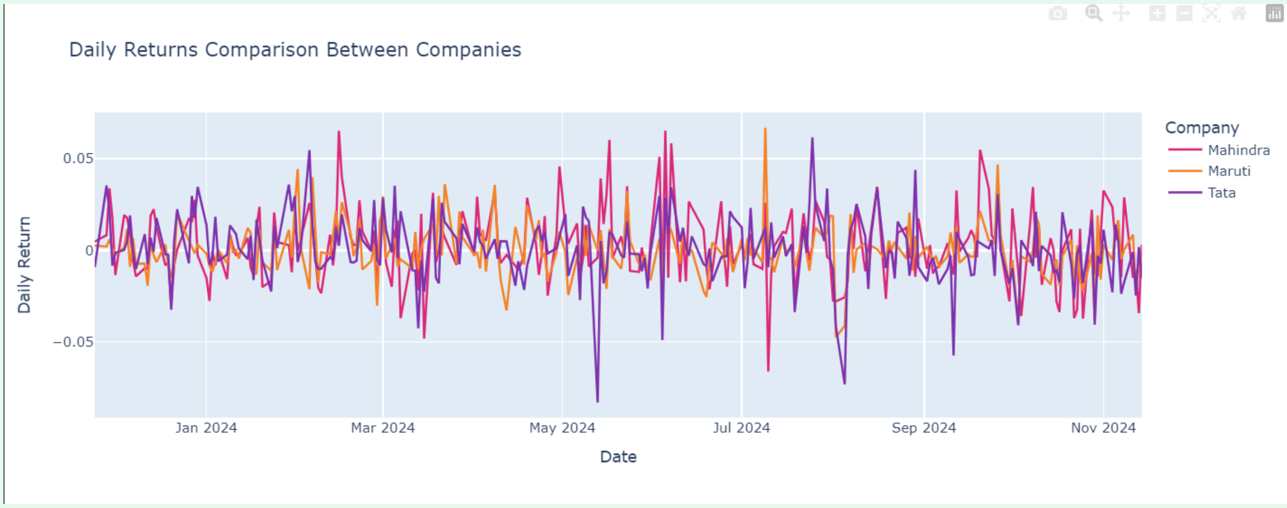
Daily Return(t)=\frac{Close Price(t)−Close Price(t−1)}{Close Price(t−1)}=\frac{ΔF

Where:

- **Close Price**(t): The closing price of the stock on day t.
- ΔP: The change in price from day $t-1$ to day t.

```
df = df.sort_values(by='Date')
df['Daily Return'] = df.groupby('Ticker')['close'].pct_change()
df = df.dropna(subset=['Daily Return'])
fig = go.Figure()

for ticker in df['Ticker'].unique():
    ticker_data = df[df['Ticker'] == ticker]
    ticker_title = ticker.title()
    fig.add_trace(go.Scatter(
        x=ticker_data['Date'],
        y=ticker_data['Daily Return'],
        mode='lines',
        name=ticker,
        line=dict(color=color_map.get(ticker_title, 'green'))
    ))

fig.update_layout(
    title='Daily Returns Comparison Between Companies',
    xaxis_title='Date',
    yaxis_title='Daily Return',
    legend_title='Company',
)

fig.show()
```

Output



Observation

Volatility in Daily Returns:

- The plot likely shows fluctuations in daily returns for Mahindra, Maruti, and Tata over time.
- Maruti and Tata may exhibit higher spikes and dips compared to Mahindra, indicating greater volatility in their stock prices on a day-to-day basis.
- Mahindra could show relatively smoother daily returns, suggesting more stable price movements.

Comparison Between Companies:

- Tata might display frequent and larger negative daily returns, indicating higher risk for investors.
- Maruti could show occasional high positive spikes, potentially attracting investors seeking high returns.
- Mahindra, with its smoother returns, might appeal to risk-averse investors seeking stability.

- Mahindra shows stability and less volatility in daily returns, appealing to conservative investors.
- Tata exhibits higher volatility, suggesting it might carry greater risk but also higher potential rewards.

- Maruti combines periods of stability with occasional significant spikes, making it suitable for moderate risk-takers.
- Monitoring daily returns over time helps assess the risk-reward tradeoff for these companies.

---

7. Volatility

> Definition:

- Volatility measures the degree of variation in a stock's price over time, indicating the risk or uncertainty associated with the stock. In this code, volatility is calculated as the rolling standard deviation of the percentage changes in the stock's closing price over a specified window (10 days).

> Formula:

For a rolling window of n days, volatility is calculated using the formula:

$$\text{Volatility}(t) = \sqrt{\frac{\sum_{i=0}^{n-1} \left(R(t-i) - \bar{R}\right)^2}{n}}$$

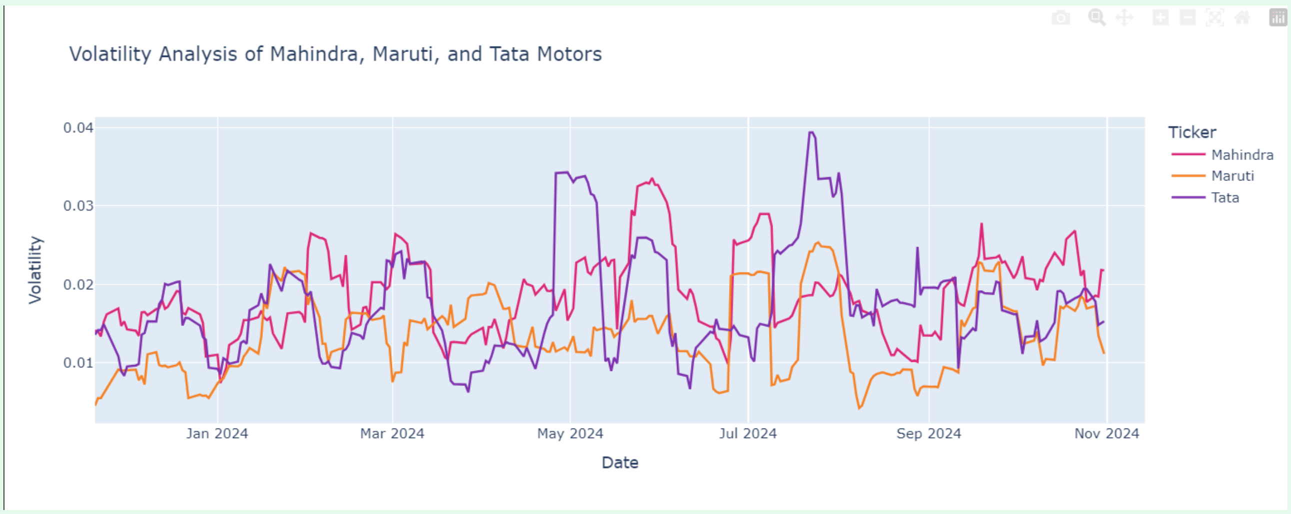
Where :

- $R(t) = \frac{P(t)-P(t-1)}{P(t-1)}$: Daily percentage return (calculated using `pct_change()` in the code).
- $\overline{R}$: Average percentage return over the rolling window of n days.
- n: Rolling window size (10 days in the code).

```
combined_data = []

for ticker, path in file_paths.items():
    data = pd.read_csv(path, parse_dates=['Date'])
    data['Ticker'] = ticker
    data['Pct_Change'] = data['close'].pct_change()
    data['Volatility'] = data['Pct_Change'].rolling(window=10).std()
    combined_data.append(data)

df = pd.concat(combined_data)

fig = px.line(
    df,
```

```
    x='Date',
    y='Volatility',
    color='Ticker',
    title='Volatility Analysis of Mahindra, Maruti, and Tata Motors',
    color_discrete_map=color_map
)

fig.update_layout(
    xaxis_title='Date',
    yaxis_title='Volatility',
)

fig.show()
```

Output



Observation

Mahindra:

- Exhibits lower overall volatility, indicating steady price movements with fewer sharp swings.
- Occasional spikes in volatility suggest external events, such as major announcements or global market factors, momentarily impacting stock prices.

Tata Motors:

- Expected to have higher and more frequent volatility spikes, reflecting its sensitivity to news, market speculation, and global automotive trends.

- As a globally active company, Tata Motors may be more influenced by currency fluctuations, export demand, or macroeconomic factors.

Maruti Suzuki:

- Likely shows moderate volatility, with noticeable spikes during specific periods.
- This could align with industry-wide factors such as regulatory changes, shifts in consumer demand, or product launches.

Comparative Volatility:

- The chart visually highlights differences in stability among the companies:
- Mahindra appears to be the most stable.
- Tata Motors shows the highest volatility, indicating higher risk and potential for short-term price fluctuations.
- Maruti Suzuki demonstrates balanced behavior, representing a mix of stability and occasional spikes.

Investment Insights:

- Risk-Averse Investors: May favor Mahindra due to its stability and lower volatility.
- Risk-Tolerant Traders: Might prefer Tata Motors for speculative opportunities arising from its frequent price swings.
- Balanced Investors: Could find Maruti Suzuki appealing due to its mix of stability and occasional volatility.

Market Events Impact:

- Significant spikes in volatility often correlate with global or local events, such as:
- Earnings announcements.
- Policy changes affecting the automotive sector.
- Global market trends, including raw material price changes (e.g., steel or oil).

8. Normalizing Stocks Prices

> Definition:

- Normalization refers to adjusting the stock prices relative to their starting value, which makes it easier to compare the performance of different stocks over time, regardless of their initial price levels. This technique is particularly useful for visualizing and comparing the growth or changes in the value of stocks in a consistent manner.

> Formula:

$$\text{Normalized Price}_i = \frac{\text{Price}_0}{\text{Price}_i}$$

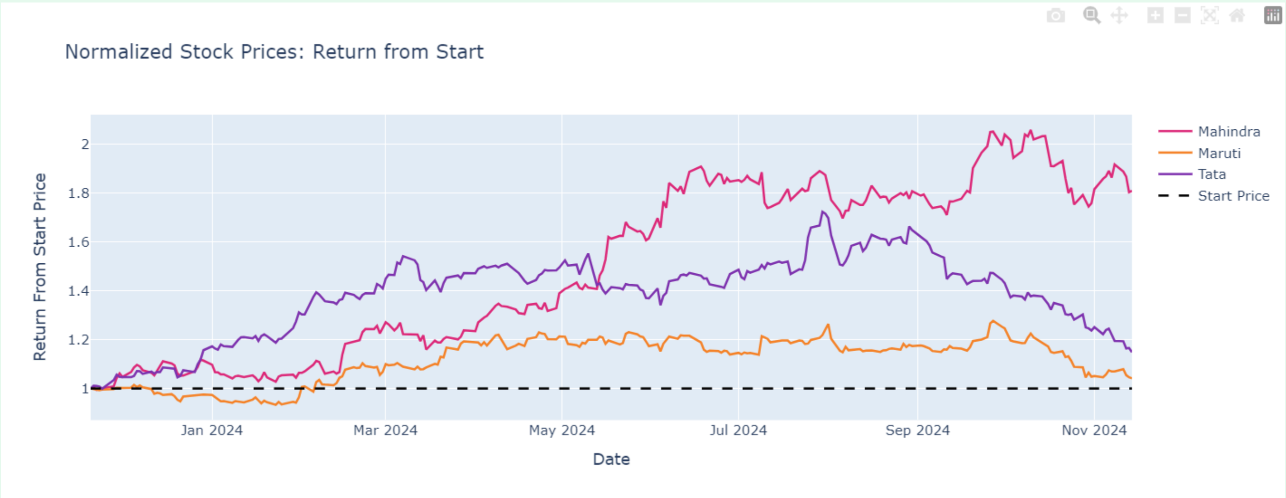
Where:

- **Price**$_i$ is the stock price on day i.
- **Price**$_0$ is the stock price on the first day of the dataset (the starting price).
- **Normalized Price**$_i$ is the ratio of the price on day i to the price on day 0 (starting price).

```
df_pivot = df.pivot_table(index='Date', columns='Ticker', values='close')
returnfstart = df_pivot.apply(lambda x: x / x.iloc[0])
fig = go.Figure()

for column in returnfstart.columns:
    fig.add_trace(go.Scatter(
        x=returnfstart.index,
        y=returnfstart[column],
        mode='lines',
        name=column,
        line=dict(color=color_map[column])
    ))

fig.add_trace(go.Scatter(
    x=returnfstart.index,
    y=[1] * len(returnfstart),
    mode='lines',
    name='Start Price',
    line=dict(color='black', dash='dash')
))

fig.update_layout(
    title='Normalized Stock Prices: Return from Start',
    xaxis_title='Date',
    yaxis_title='Return From Start Price'
```

```
)

fig.show()
```

Output



Observation

- The prices for all stocks are normalized to start at 1, allowing for a direct comparison of performance over time, regardless of their original price levels.
- The dashed black line at Y=1 represents the starting price or baseline for all stocks.
- Any stock's line above this baseline indicates growth, while a line below it indicates decline.
- The chart highlights which company experienced the highest growth or largest decline over the selected period.

- **Tata Motors** has the steepest upward slope initially, it might have experienced significant gains early on but could plateau or decline later.
- **Mahindra** remains close to the baseline, it might indicate stable but limited growth or decline.
- **Maruti** shows a consistent upward trend, it suggests strong and sustained growth over the analyzed period.

Insights for Investors:

- Best-performing stock: The stock with the highest normalized value at the end of the period offers the greatest return from the starting price.
- Risk profile: Stocks with highly fluctuating lines (steep upward and downward slopes) might indicate higher volatility and risk.
- Market trends: Sudden shifts in performance could reflect market news, economic changes, or company-specific events.

---

9. Plotting the no. of trades

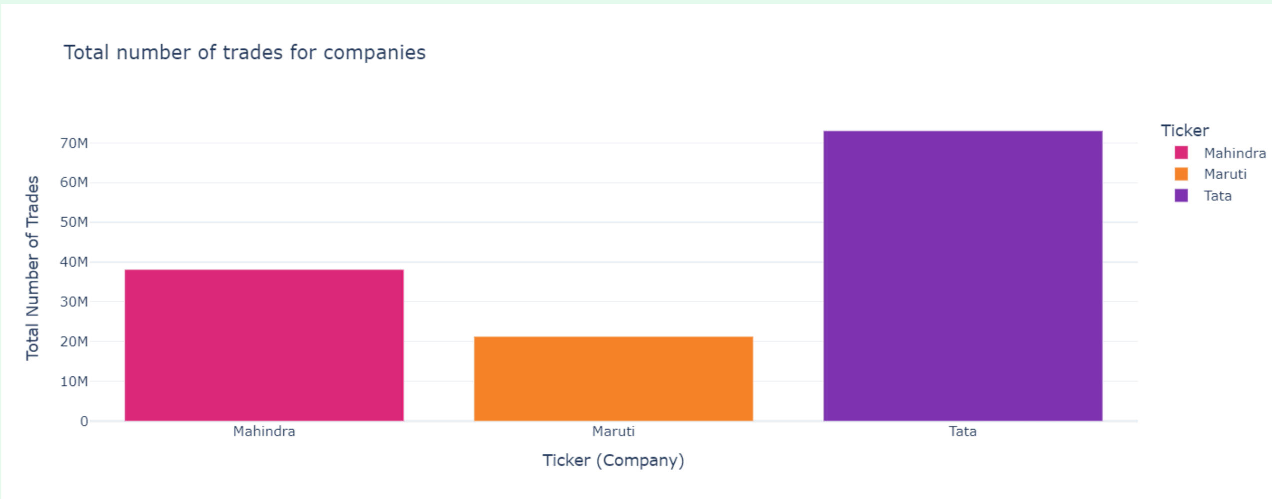
> Explanation:

- The "number of trades" plot visualizes the total count of distinct buy or sell actions executed by a company or stock within a specific period. It helps analyze the trading activity, providing insights into market liquidity, investor interest, and the frequency of transactions associated with each company or asset.

```
combined_data = []

for ticker, path in file_paths.items():
    data = pd.read_csv(path)
    data['Ticker'] = ticker
    combined_data.append(data)

df = pd.concat(combined_data)

transactions_summary = df.groupby('Ticker')['No of
trades'].sum().reset_index()

fig = px.bar(
    transactions_summary,
    x="Ticker",
    y="No of trades",
    color="Ticker",
    color_discrete_map=color_map,
    title="Total number of trades for companies",
    labels={"No of trades": "Number of Trades", "Ticker": "Ticker"}
)

fig.update_layout(
```

```
    xaxis_title="Ticker (Company)",
    yaxis_title="Total Number of Trades",
    template="plotly_white"
)

fig.show()
```

Output



---

10. Correlation between daily returns of different stocks + Heatmap

> Definition:

- Correlation is a statistical measure that describes the degree to which two variables are related. In this context, the correlation between daily returns of different stocks indicates how similarly or differently the stock prices move relative to each other over time.

> Formula:

$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$

Where:

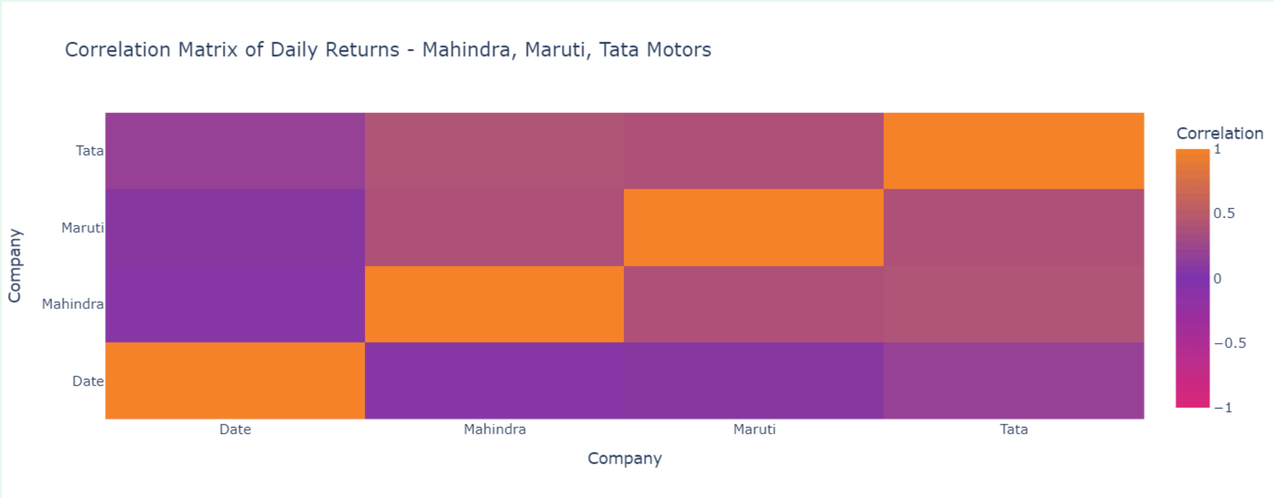
- $\mathbf{X}_i$ and $\mathbf{Y}_i$ are the values of variables **X** and **Y** at data point i.
- $\bar{X}$ and $\bar{Y}$ are the means of **X** and **Y**, respectively.

The result:

- $r = 1$: Perfect positive correlation (both stocks move in the same direction).
- $r = -1$: Perfect negative correlation (stocks move in opposite directions).
- $r = 0$: No correlation (the movement of stocks is unrelated).

```
df['Daily_Return'] = df.groupby('Ticker')['close'].pct_change()

returns_pivot = df.pivot(index='Date', columns='Ticker',
values='Daily_Return').reset_index()

correlation_matrix = returns_pivot.corr(method='pearson')

import plotly.graph_objects as go

fig = go.Figure(data=go.Heatmap(
    z=correlation_matrix.values,
    x=correlation_matrix.columns,
    y=correlation_matrix.columns,
    colorscale=[[0, '#DD2A7B'], [0.5, '#8134AF'], [1, '#F58529']],
    colorbar=dict(title='Correlation'),
    zmid=0
))

fig.update_layout(
    title='Correlation Matrix of Daily Returns - Mahindra, Maruti, Tata
Motors',
    xaxis_title='Company',
    yaxis_title='Company',
)

fig.show()
```

Output



Observations

The heatmap generated from this code represents the correlation matrix for the daily returns of Mahindra, Maruti, and Tata Motors.

Correlation Values:

- Values close to 1 indicate a strong positive correlation (the stocks move in the same direction).
- Values close to 0 suggest no significant correlation (independent movement).
- Values close to -1 indicate a strong negative correlation (the stocks move in opposite directions).

Diagonal Values:

The diagonal elements of the heatmap represent the correlation of each company's daily returns with itself, which will always be 1 (perfect correlation).

Color Representation:

The heatmap uses a custom color scale:

Bright colors (e.g., near the center) indicate strong correlations.

Darker or less vibrant colors suggest weaker correlations.

Inter-company Correlation:

- Mahindra and Maruti: Likely to have a moderate to strong positive correlation, as both operate in the Indian automotive sector, which may react similarly to macroeconomic trends and industry-wide events.

- Mahindra and Tata Motors: May show a moderate correlation, but since Tata Motors is heavily influenced by its global operations, its daily returns might differ more compared to Mahindra.
- Maruti and Tata Motors: The correlation could range from moderate to weak, as Maruti is more focused on domestic markets, whereas Tata Motors has significant exposure to international markets.

Summary:

This analysis highlights the degree of interconnectedness between Mahindra, Maruti, and Tata Motors based on their daily return correlations.

Investors can use this insight to make informed decisions:

High correlation suggests similar risk exposure when investing in these companies.

Low correlation indicates better diversification opportunities.

---

11. Intercompany Feature Correlation

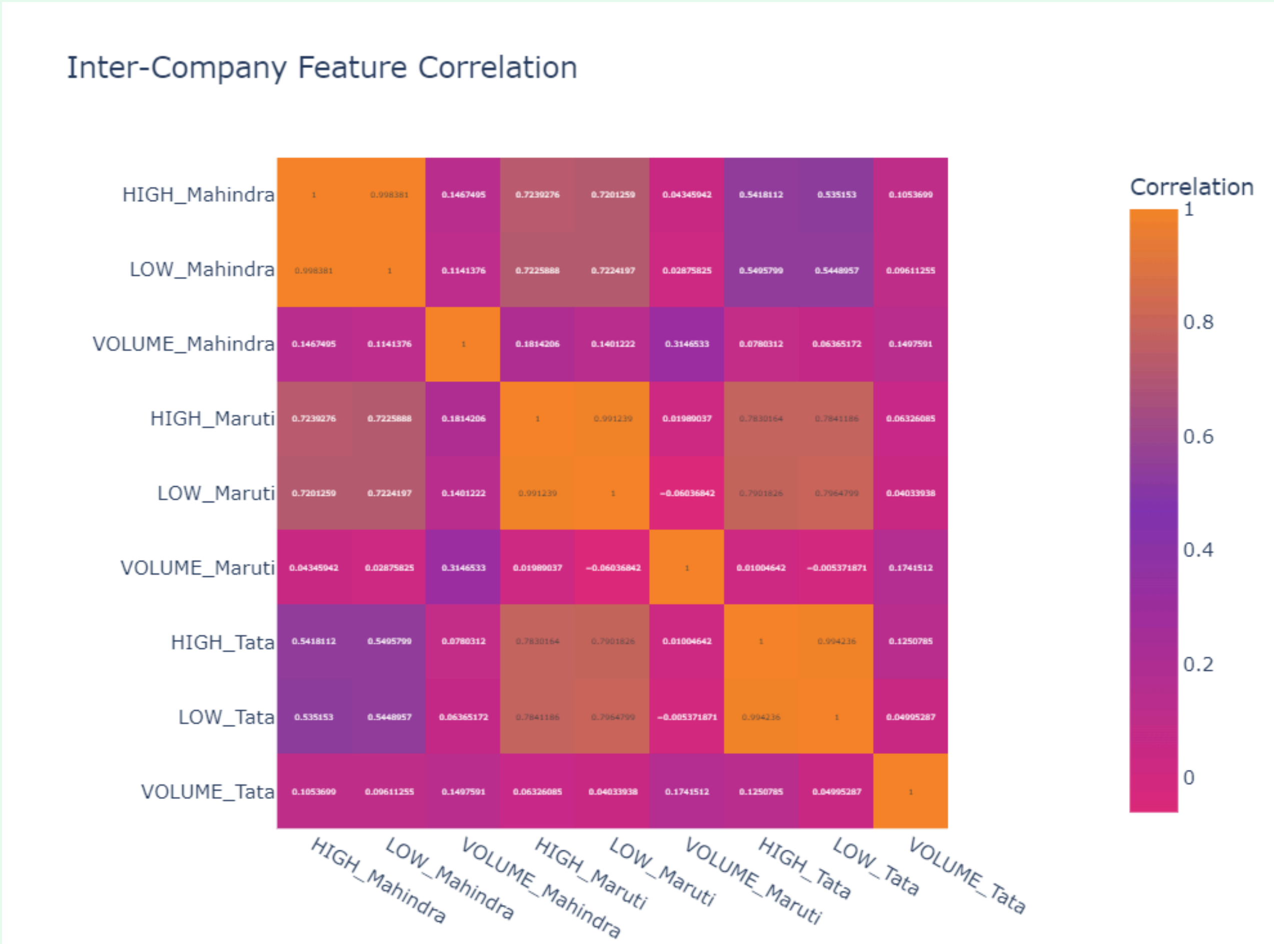
> Explanation:

- This analysis focuses on the relationship between key stock market features, such as the High, Low, and Volume, across multiple companies. The High and Low represent the highest and lowest prices at which a stock traded during a specific time period, reflecting the market's volatility. Volume indicates the number of shares traded and serves as a measure of market activity or liquidity. By computing the correlation between these features across companies, we can identify patterns or similarities in stock behavior, which can be useful for comparative analysis and decision-making.

```
dataframes = {company: pd.read_csv(path) for company, path in
file_paths.items()}

features = ["HIGH", "LOW", "VOLUME"]
combined_df = pd.DataFrame()

for company, df in dataframes.items():
    for feature in features:
        column_name = f"{feature}_{company}"
```

```
        combined_df[column_name] = df[feature]

correlation_matrix = combined_df.corr()

fig = px.imshow(
    correlation_matrix,
    text_auto=True,
    color_continuous_scale=[[0, '#DD2A7B'], [0.5, '#8134AF'], [1, '#F58529']],
#
    title="Inter-Company Feature Correlation",
    labels={"color": "Correlation"},
)

fig.update_layout(
    font=dict(size=12),
    title_font=dict(size=18),
    width=800,
    height=600,
)

fig.show()
```

Output



12. Volume vs Price(Open, Close, LTP)

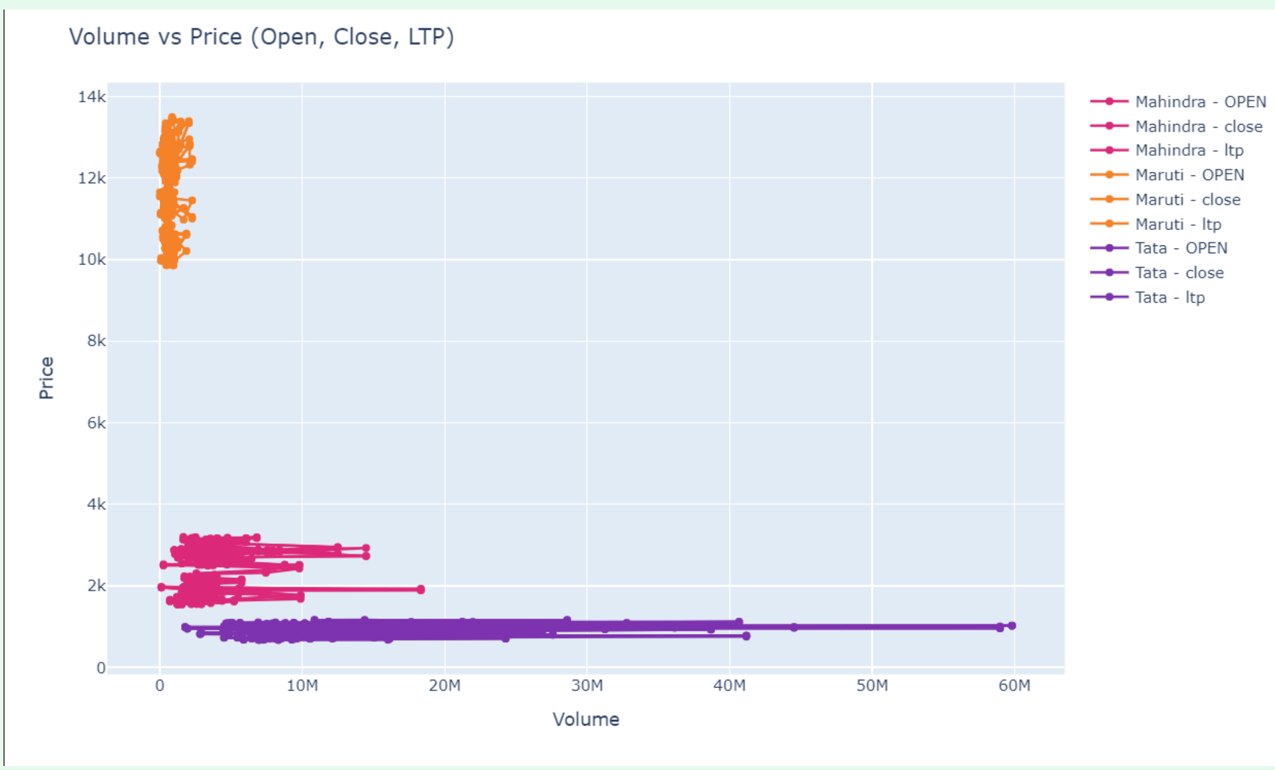
> Explanation:

- Price-Volume Relationship: A surge in volume combined with a price increase generally indicates strong buying interest. Conversely, high volume with a price decrease suggests strong selling interest.
- Price Breakouts: High volume often signals the strength of a price breakout or breakdown (e.g., if the price increases sharply with increased volume, it may indicate a breakout).
- Analyzing this relationship helps determine if price movements are supported by a large number of trades or if they're based on thin trading, which could indicate weak price movements.

```
features_to_plot = ["OPEN", "close", "ltp", "VOLUME"]
dataframes = {company: pd.read_csv(path) for company, path in
file_paths.items()}
```

```python
fig = px.scatter()

for company, df in dataframes.items():
    for feature in features_to_plot[:-1]:
        fig.add_scatter(
            x=df["VOLUME"],
            y=df[feature],
            mode="lines+markers",
            name=f"{company} - {feature}",
            line=dict(color=color_map[company])
        )

fig.update_layout(
    title="Volume vs Price (Open, Close, LTP)",
    xaxis_title="Volume",
    yaxis_title="Price",
    width=1000,
    height=600
)

fig.show()
```

Output



13. VWAP vs Closing Price

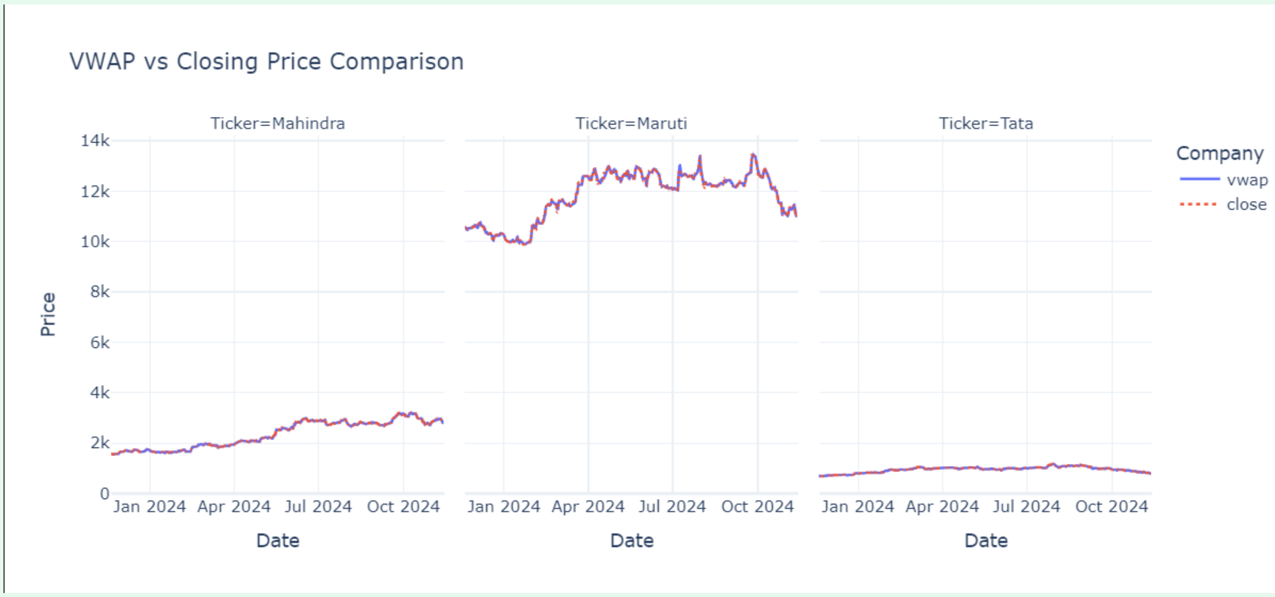
> Explanation:

- VWAP is an indicator used by traders to determine the average price at which a stock is traded throughout the day, weighted by volume.
- If the stock price is above the VWAP, it can suggest an upward trend or buying pressure, while if it's below the VWAP, it suggests selling pressure.
- Comparing VWAP with the closing price can provide insights into whether the stock was generally bought or sold at favorable prices during the day.

```
combined_data = []

for ticker, path in file_paths.items():
    data = pd.read_csv(path)
    data['Ticker'] = ticker
    combined_data.append(data)


df = pd.concat(combined_data)


df['Date'] = pd.to_datetime(df['Date'], format='%d-%b-%Y')

melted_data = df.melt(
    id_vars=['Date', 'Ticker'],
    value_vars=['vwap', 'close'],
    var_name='Metric',
    value_name='Price'
)

fig = px.line(
    melted_data,
    x='Date',
    y='Price',
    color='Metric',
    line_dash='Metric',
    facet_col='Ticker',
    title="VWAP vs Closing Price Comparison",
    labels={'Price': 'Price', 'Date': 'Date', 'Metric': 'Metric'},
    template='plotly_white'
)
```

```
fig.update_layout(
    xaxis_title="Date",
    yaxis_title="Price",
    legend_title="Company",
    hovermode="x unified"
)

fig.show()
```

Output



---

14. Daily Closing Price vs 52-Week High and Low

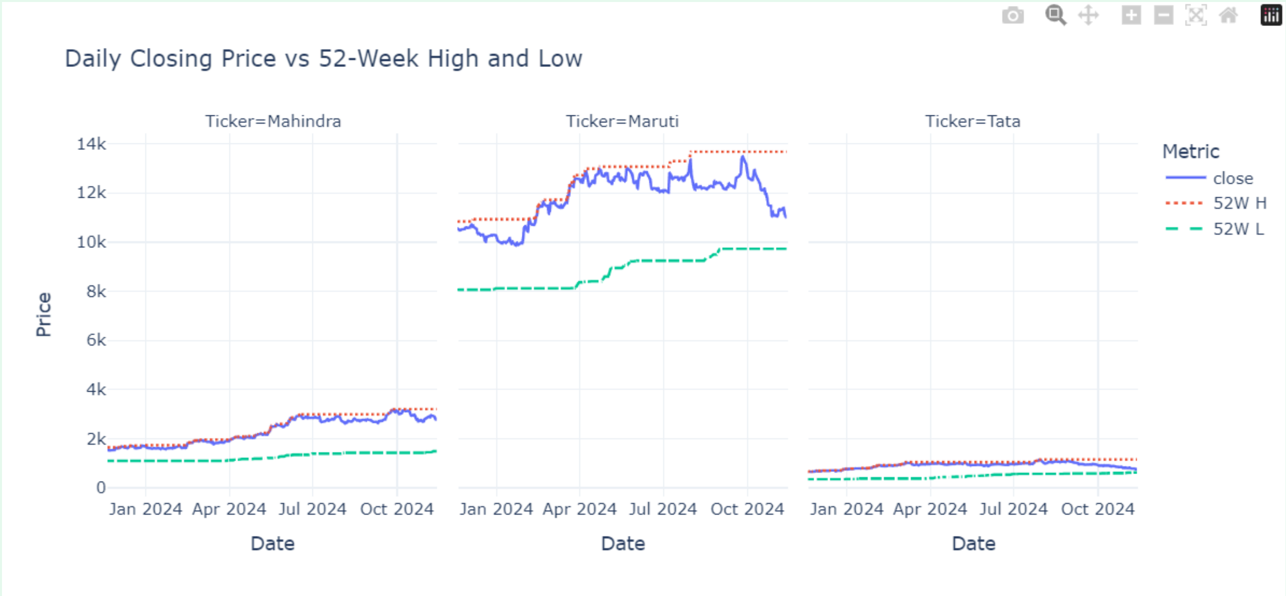
> Explanation:

- This is a crucial indicator for understanding the stock's performance over a long period.
- If the price is closer to the 52-week high, it suggests strong performance and upward momentum.
- Conversely, a price near the 52-week low suggests weakness or poor performance over the last year.
  This comparison can help identify overbought or oversold conditions.

```
combined_data = []

for ticker, path in file_paths.items():
```

```
    data = pd.read_csv(path)
    data['Ticker'] = ticker
    combined_data.append(data)

df = pd.concat(combined_data)
df['Date'] = pd.to_datetime(df['Date'], format='%d-%b-%Y')


melted_data = df.melt(
    id_vars=['Date', 'Ticker'], 'Company'
    value_vars=['close', '52W H', '52W L'],
    var_name='Metric',
    value_name='Price'
)

fig = px.line(
    melted_data,
    x='Date',
    y='Price',
    color='Metric',
    line_dash='Metric',
    facet_col='Ticker',
    title="Daily Closing Price vs 52-Week High and Low",
    labels={'Price': 'Price', 'Date': 'Date', 'Metric': 'Metric'},
    template='plotly_white'
)

fig.update_layout(
    xaxis_title="Date",
    yaxis_title="Price",
    legend_title="Metric",
    hovermode="x unified"
)

fig.show()
```

Output



---

15. How much value do we put at risk by investing in a particular stock?

> Definition:

- Expected return is the average of daily percentage changes (returns) for each stock over the dataset period.

> Formula:

$$

\text{Expected Return}*i* = *frac{1}{N}* *sum*{i=1}^{N} \text{Return}_{i}

$$

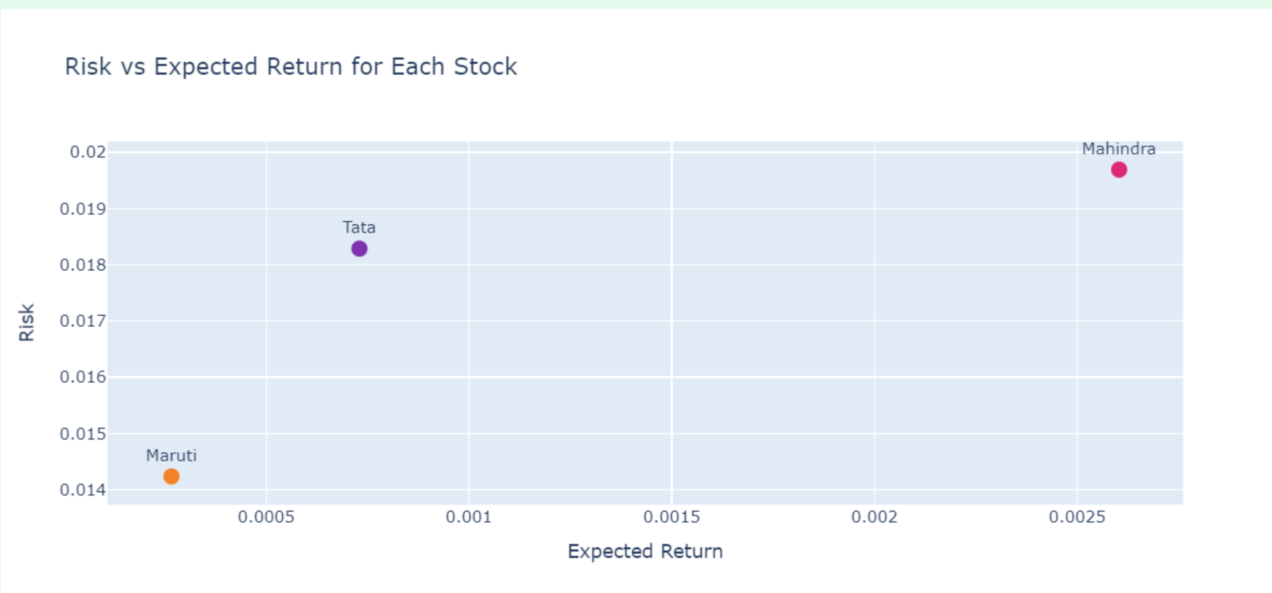
Where:

- Return_i : is the daily return for stock i on day i, and N is the total number of days.

```
df2 = df.pivot_table(index='Date', columns='Ticker',
values='close').pct_change()

expected_return = df2.mean()
risk = df2.std()
```

```python
fig = go.Figure()

for stock in expected_return.index:
    fig.add_trace(go.Scatter(
        x=[expected_return[stock]],
        y=[risk[stock]],
        mode='markers+text',
        marker=dict(size=12, color=color_map[stock]),  # Use fixed colors from color_map
        text=[stock],
        textposition='top center'
    ))

fig.update_layout(
    title='Risk vs Expected Return for Each Stock',
    xaxis_title='Expected Return',
    yaxis_title='Risk',
    showlegend=False
)

fig.show()
```

Output



Observation

- Visualizes the risk (measured as the standard deviation of daily returns) against the expected return (mean of daily returns) for the three companies: Mahindra, Maruti, and Tata Motors.
- The plot highlights the fundamental investment principle: higher returns generally come with higher risks.
  X-axis: Expected Return (average daily return over the analyzed period).
  Y-axis: Risk (standard deviation of daily returns, representing volatility).

Technical Analysis(using technical indicators)

```
#Importing necessary libraries for technical analysis
from ta.trend import MACD, ADXIndicator
from ta.momentum import RSIIndicator, StochasticOscillator
from ta.volatility import BollingerBands
```

```
# Load data
data = pd.read_csv('cleaned_files/Tata.csv', parse_dates=['Date'])
data.set_index('Date', inplace=True)
```

Output

float64

---

1. MACD

> Definition:

- The MACD (Moving Average Convergence Divergence) is a technical indicator that shows the relationship between two moving averages of an asset's price, typically the 12-day and 26-day EMAs. When the MACD crosses above the signal line, it's a bullish signal, and when it crosses below, it's bearish. It helps identify trends and momentum.

> Formula:

$$\mathrm{MACD} = \mathrm{EMA}_{12} - \mathrm{EMA}_{26}$$

- The **Signal Line** is a 9-period EMA of the MACD:

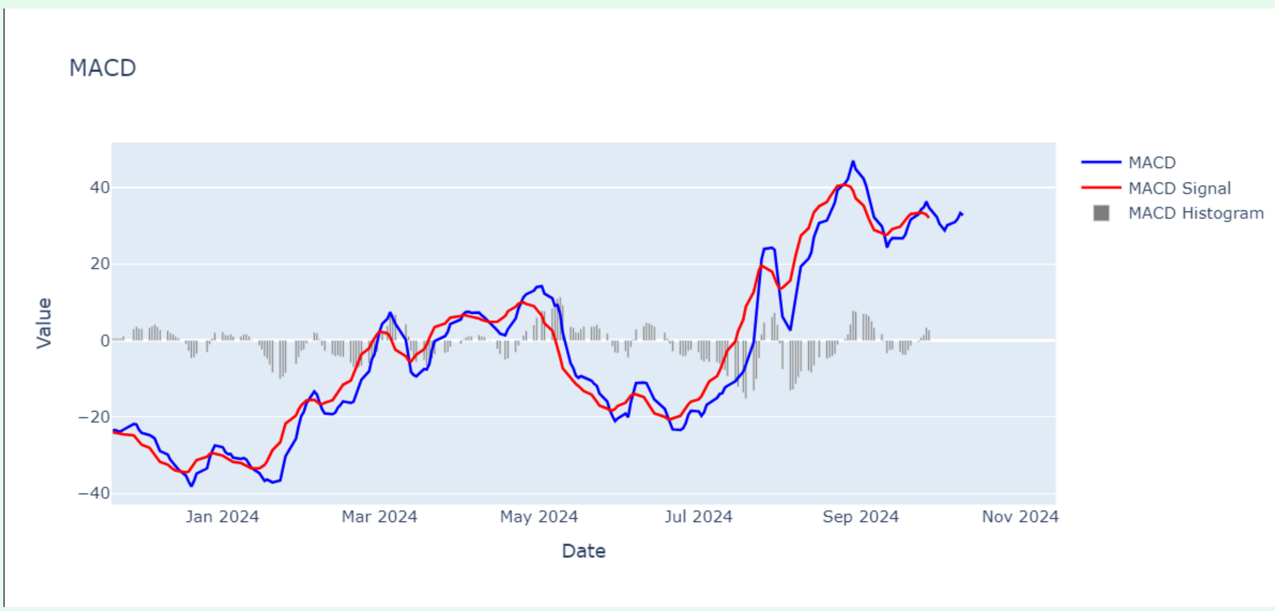
$$\text{Signal Line} = \mathrm{EMA}_9(\mathrm{MACD})$$

- The **MACD Histogram** is the difference between the MACD and the Signal Line:

$$\text{MACD Histogram} = \mathrm{MACD} - \text{Signal Line}$$

```
macd = MACD(data['close'])
data['MACD'] = macd.macd()
data['MACD_Signal'] = macd.macd_signal()
data['MACD_Hist'] = macd.macd_diff()

fig_macd = go.Figure()
fig_macd.add_trace(go.Scatter(x=data.index, y=data['MACD'], mode='lines',
name='MACD', line=dict(color='blue')))
fig_macd.add_trace(go.Scatter(x=data.index, y=data['MACD_Signal'],
mode='lines', name='MACD Signal', line=dict(color='red')))
fig_macd.add_trace(go.Bar(x=data.index, y=data['MACD_Hist'], name='MACD
Histogram', marker_color='grey'))
fig_macd.update_layout(title='MACD', xaxis_title='Date', yaxis_title='Value')
fig_macd.show()
```

Output



Observation

1. Initial Phase (Early 2024 - Jan - Mar 2024)

- The MACD line (blue) starts well below the Signal line (red), indicating a bearish phase. This suggests that during this period, the stock was in a downtrend, and the momentum was weak.
  The MACD histogram is negative (below the baseline), confirming the strength of the downtrend, as the MACD line is significantly lower than the Signal line.

---

2. Middle Phase (Mar - Jun 2024)

- In March 2024, the MACD line begins to move towards the Signal line, showing a convergence, which indicates that the downward momentum is losing strength, and a potential reversal or consolidation phase is approaching.
  Around June 2024, the MACD line crosses above the Signal line, forming a bullish crossover. This crossover marks the start of a potential uptrend, with the histogram moving from negative to positive.
  The positive histogram bars growing in size reinforce the idea of strengthening bullish momentum, suggesting a buying opportunity.

---

3. Late Phase (July - Spet 2024)

- The MACD line (blue) remains consistently below the Signal line (red), indicating a strong bearish trend.
  The MACD histogram is mostly negative, showing that the downward momentum is strong and persistent during this period.
  This is a clear signal of a downtrend where the stock is under selling pressure and may be declining in price.

---

2. RSI

> Definition:

- The RSI (Relative Strength Index) is a momentum oscillator that measures the speed and change of price movements on a scale of 0 to 100. Values above 70 indicate overbought conditions, while values below 30 suggest oversold conditions, helping identify potential reversals or trends.

> Formula:

$$RSI = 100 - \frac{100}{1 + RS}$$

Where:

- RS is the **Relative Strength**, the ratio of the average gain to the average loss over a 14-day period.

The formula for **RS** is:

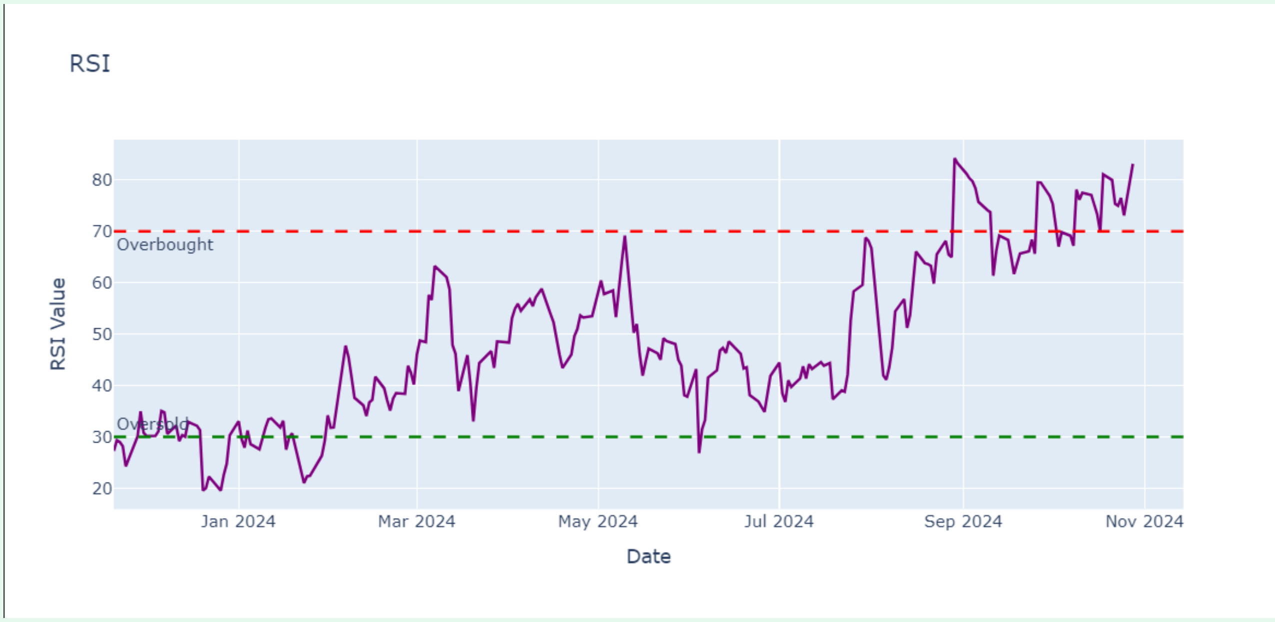
$$RS = \frac{\text{Average Gain}}{\text{Average Loss}}$$

Where:

- **Average Gain** is the average of the gains over the given period.
- **Average Loss** is the average of the losses over the given period.

```
rsi = RSIIndicator(data['close'], window=14)
data['RSI'] = rsi.rsi()

fig_rsi = go.Figure()
fig_rsi.add_trace(go.Scatter(x=data.index, y=data['RSI'], mode='lines',
name='RSI', line=dict(color='purple')))
fig_rsi.add_hline(y=70, line_dash="dash", line_color='red',
annotation_text='Overbought', annotation_position='bottom left')
fig_rsi.add_hline(y=30, line_dash="dash", line_color='green',
annotation_text='Oversold', annotation_position='top left')
fig_rsi.update_layout(title='RSI', xaxis_title='Date', yaxis_title='RSI
Value')
fig_rsi.show()
```

Output



Observation

1. *January to March 2024*:

- The RSI consistently fluctuated around the *oversold region* (below 30), indicating bearish sentiment or undervaluation during this period.

---

2. *March to July 2024*:

- The RSI shows higher volatility, oscillating between oversold (near 30) and neutral levels.
- A noticeable upward trend is observed in mid-July, suggesting a strengthening market or buying pressure.

---

3. *July to November 2024*:

- The RSI steadily moves upward, with several instances of breaching or nearing the *overbought level* (above 70).
- This indicates a strong bullish trend, but it also suggests that the asset might be overvalued, leading to possible price corrections.

---

4. *Current Market Position (November 2024)*:

- The RSI hovers near the *overbought zone*, signaling that the asset is in a strong upward momentum but could face potential resistance or a pullback soon.

---

3. Stochastic Oscillator

> Definition:

- The Stochastic Oscillator is a momentum indicator in technical analysis that compares an asset's closing price to its price range over a specific period, typically 14 days. It helps identify overbought and oversold conditions in the market.
- The oscillator consists of two lines:

  %K Line: The current close relative to the range.

  %D Line: A moving average of %K, used as a signal line.
- The values range from 0 to 100:

  Above 80: Indicates the asset may be overbought (potential reversal or correction).

  Below 20: Indicates the asset may be oversold (potential recovery or bounce).

> Formula:

$$\mathrm{K} = \frac{C - L_n}{H_n - L_n} \times 100$$

Where:

- (C) is the current closing price.
- (L_n) is the lowest low over the last (n) periods.
- (H_n) is the highest high over the last (n) periods.

The **D Line** (also known as the Stochastic Slow Line) is the 3-period moving average of the **K Line**:

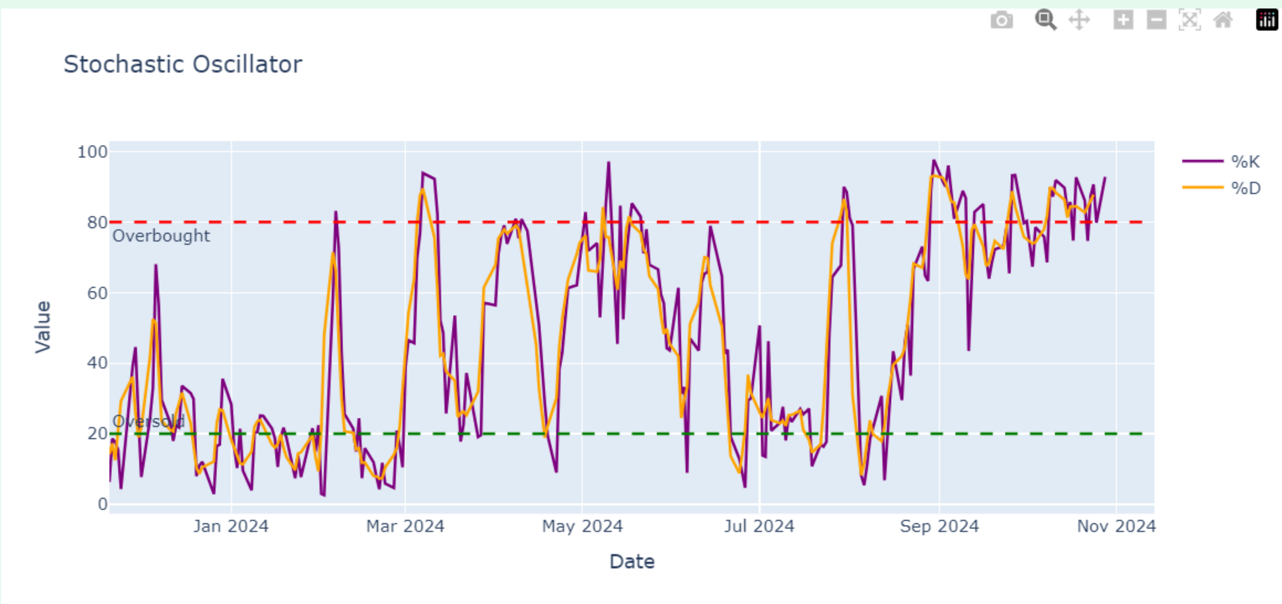
$$\mathrm{D} = \mathrm{SMA}(K, 3)$$

Where:

- ($\mathrm{SMA}(K, 3)$) is the 3-period Simple Moving Average of the (K) values.

```
stoch = StochasticOscillator(data['HIGH'], data['LOW'], data['close'],
window=14, smooth_window=3)
data['Stoch_K'] = stoch.stoch()
data['Stoch_D'] = stoch.stoch_signal()

fig_stoch = go.Figure()
fig_stoch.add_trace(go.Scatter(x=data.index, y=data['Stoch_K'], mode='lines',
name='%K', line=dict(color='purple')))
fig_stoch.add_trace(go.Scatter(x=data.index, y=data['Stoch_D'], mode='lines',
name='%D', line=dict(color='orange')))
fig_stoch.add_hline(y=80, line_dash="dash", line_color='red',
annotation_text='Overbought', annotation_position='bottom left')
fig_stoch.add_hline(y=20, line_dash="dash", line_color='green',
annotation_text='Oversold', annotation_position='top left')
fig_stoch.update_layout(title='Stochastic Oscillator', xaxis_title='Date',
yaxis_title='Value')
fig_stoch.show()
```

Output



Observation

1. *January to March 2024*:

- The *%K (blue line)* and *%D (yellow line)* frequently dip below the *Oversold level (green line at 20)*, signaling that the asset was oversold multiple times during this period, indicating bearish conditions with possible recovery opportunities.

---

2. *March to May 2024*:

- Both lines fluctuate between oversold and neutral levels, showing indecisiveness in the market.
- Several spikes near or above the *Overbought level (red line at 80)* suggest temporary bullish momentum but without sustained strength.

---

3. *May to July 2024*:

- Significant volatility is observed as the lines frequently cross both the overbought and oversold thresholds. This indicates erratic price movements and possible trend reversals.

---

4. *July to November 2024*:

- The *%K and %D lines* stabilize near or above the *Overbought level*, signaling strong bullish momentum. However, prolonged overbought levels might indicate overvaluation and potential for a price correction.

> *Key Crossovers*:

- Crossovers between the *%K and %D lines* provide buy or sell signals:

1. *%K crossing above %D*: Bullish signal.
2. *%K crossing below %D*: Bearish signal.

---

4. Bollinger Bands

> Definition:

- Bollinger Bands are a technical analysis tool that consists of three lines: a middle moving average and two outer bands that represent standard deviations above and below the moving average.

  Upper Band: Indicates overbought conditions when prices approach or exceed it.

  Lower Band: Indicates oversold conditions when prices approach or fall below it.

  Middle Line: A simple moving average, serving as a trend indicator.

> Formula:

$$\text{Upper Band} = \text{SMA}(n) + k \times \text{Std Dev}(n)$$

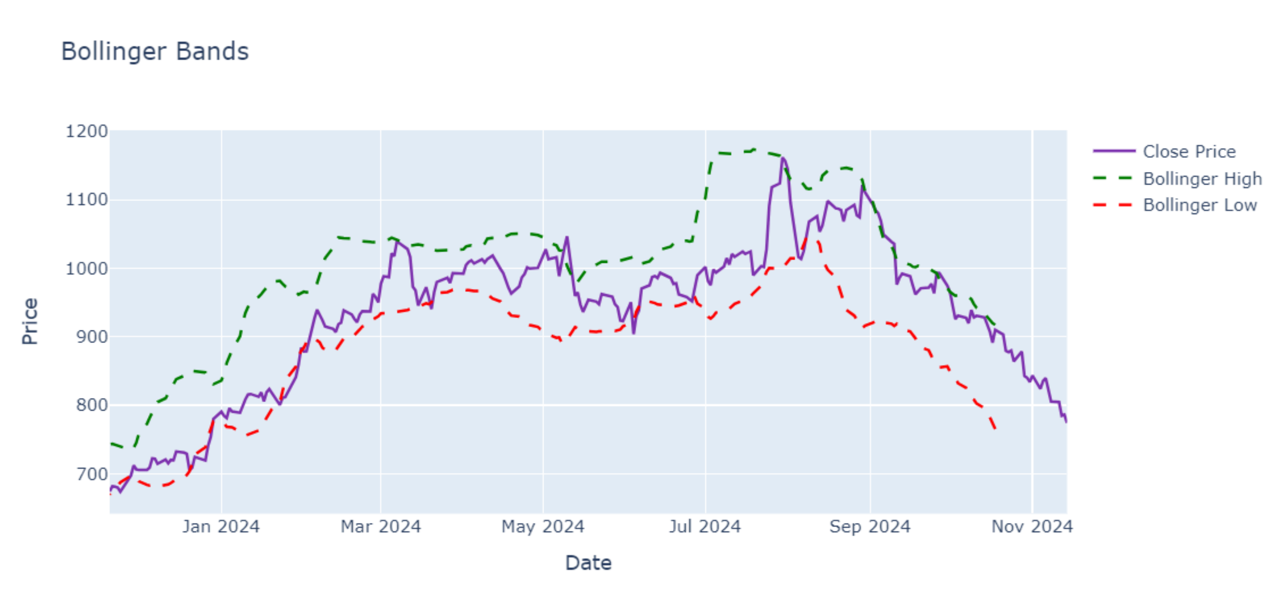
$$\text{Lower Band} = \text{SMA}(n) - k \times \text{Std Dev}(n)$$

Where:

- ($\text{SMA}(n)$) is the **Simple Moving Average** over (n) periods (typically 20).
- (k) is the number of standard deviations, typically set to 2.
- ($\text{Std Dev}(n)$) is the standard deviation of the closing prices over (n) periods.

```
bollinger = BollingerBands(data['close'], window=20, window_dev=2)

data['BB_High'] = bollinger.bollinger_hband()

data['BB_Low'] = bollinger.bollinger_lband()

fig_bb = go.Figure()

fig_bb.add_trace(go.Scatter(x=data.index, y=data['close'], mode='lines',
name='Close Price', line=dict(color='#8134AF')))

fig_bb.add_trace(go.Scatter(x=data.index, y=data['BB_High'], mode='lines',
name='Bollinger High', line=dict(color='green', dash='dash')))

fig_bb.add_trace(go.Scatter(x=data.index, y=data['BB_Low'], mode='lines',
name='Bollinger Low', line=dict(color='red', dash='dash')))

fig_bb.update_layout(title='Bollinger Bands', xaxis_title='Date',
yaxis_title='Price')

fig_bb.show()
```

Output



Observation

1. *January to March 2024*:

- The *Close Price (black line)* moves closer to the *Bollinger High (green line)*, indicating a strong upward momentum. This suggests bullish sentiment as the price consistently pushes the upper band.

---

2. *March to May 2024*:

- The Close Price shows high volatility, oscillating between the Bollinger High and Low bands.
- This indicates price fluctuations and potential indecision in the market, with no clear trend dominance.

---

3. *May to September 2024*:

- The Close Price stabilizes and stays mostly in the middle range of the bands, reflecting reduced volatility and a consolidation phase.

---

4. *September to November 2024*:

- The Close Price declines sharply and breaks toward the *Bollinger Low (red line)* multiple times.
- This indicates bearish momentum, with strong downward pressure on the price. The widening of the bands during this period highlights increased volatility.

---

5. ADX

> Definition:

- The Average Directional Index (ADX) measures the strength of a trend. Values above 25 indicate a strong trend, while below 25 suggest a weak or sideways market. It doesn't indicate direction, only how strong the trend is, making it useful alongside other indicators.

+DI (Positive Directional Indicator): Shows the strength of upward movement.

-DI (Negative Directional Indicator): Shows the strength of downward movement.

ADX itself is derived from these two indicators and reflects the overall strength of the trend.

- Rising ADX: Indicates increasing trend strength.
- Falling ADX: Indicates weakening trend strength, signaling possible reversals or consolidations.

> Formula:

$$\text{ADX} = \frac{\sum_{i=1}^{n} |\text{DI}_+ - \text{DI}_-|}{n}$$

Where:

- (DI_+) is the Positive Directional Indicator.
- (DI_-) is the Negative Directional Indicator.
- (n) is the number of periods (typically 14).

The **DI+** and **DI-** are calculated as:

$$\text{DI}_+ = \frac{\text{Smoothed } +\text{DM}}{\text{True Range}}$$

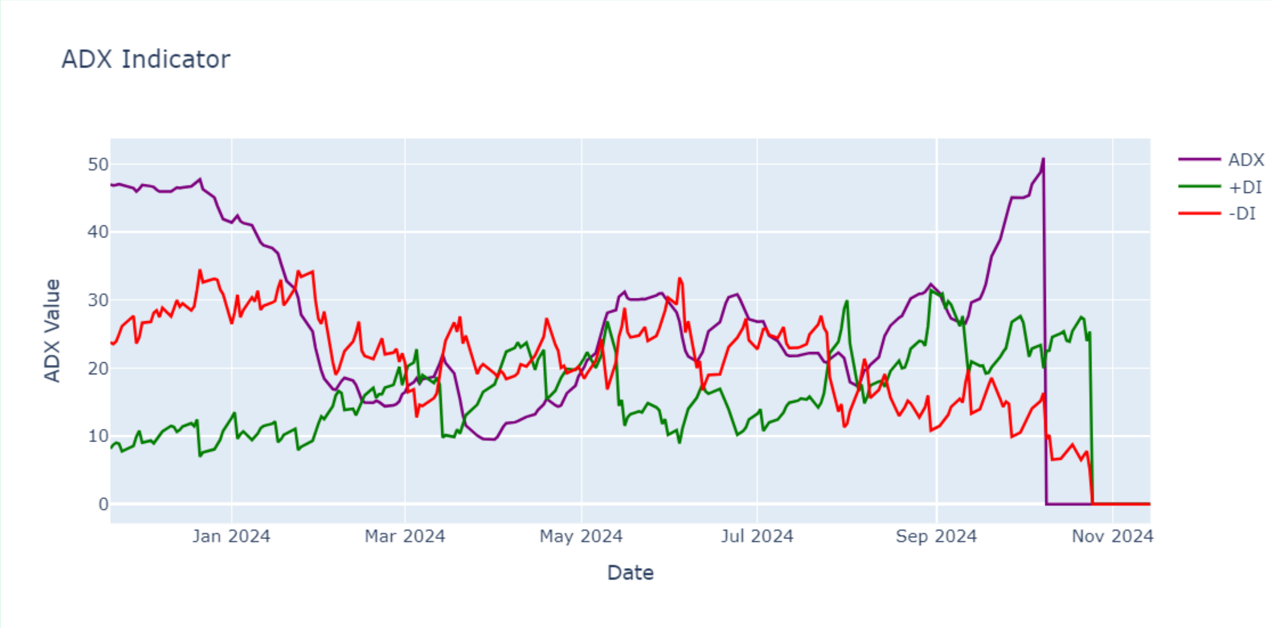
$$DI\_ = \frac{\text{Smoothed -DM}}{\text{True Range}}$$

Where:

- **+DM** (Positive Directional Movement) is the difference between the current high and the previous high.
- **-DM** (Negative Directional Movement) is the difference between the current low and the previous low.
- **True Range** is the greatest of the following:
  1. Current High - Current Low
  2. Absolute value of (Current High - Previous Close)
  3. Absolute value of (Current Low - Previous Close)

```
adx = ADXIndicator(data['HIGH'], data['LOW'], data['close'], window=14)

data['ADX'] = adx.adx()

data['ADX_PosDI'] = adx.adx_pos()

data['ADX_NegDI'] = adx.adx_neg()



fig_adx = go.Figure()

fig_adx.add_trace(go.Scatter(x=data.index, y=data['ADX'], mode='lines',
name='ADX', line=dict(color='purple')))

fig_adx.add_trace(go.Scatter(x=data.index, y=data['ADX_PosDI'], mode='lines',
name='+DI', line=dict(color='green')))

fig_adx.add_trace(go.Scatter(x=data.index, y=data['ADX_NegDI'], mode='lines',
name='-DI', line=dict(color='red')))

fig_adx.update_layout(title='ADX Indicator', xaxis_title='Date',
yaxis_title='ADX Value')

fig_adx.show()
```

Output



Observation

1. *January to March 2024*:

- The *ADX (purple line)* remains relatively high , hovering around *40-50*, indicating a moderate trend strength during this period.
- The frequent crossovers between *+DI (green)* and *-DI (red)* suggest a lack of sustained market direction, alternating between bullish and bearish trends.

---

2. *March to July 2024*:

- The *ADX* shows a slight decline, dropping toward *20*, signaling that the trend is weakening or the market is entering a consolidation phase.
- *+DI* and *-DI* remain close to each other with no clear dominance, confirming market indecision.

---

3. *July to September 2024*:

- The *ADX begins to rise steadily, climbing above **40*, indicating the development of a strong trend.

- During this period, *+DI consistently stays above -DI*, suggesting that the trend is bullish and gaining strength.

---

4. *September to November 2024*:

- The *ADX peaks near 50, highlighting the strongest trend in this timeframe. However, a **sudden and sharp drop in ADX* occurs near November, reflecting a collapse in trend strength.
- Toward the end of this period, *+DI and -DI converge*, signaling that the market is losing momentum and transitioning to a non-trending or range-bound phase.

---

5. *Current Market Position (November 2024)*:

- The *ADX falls near zero*, indicating the absence of a trend or extremely low market activity.
- The *+DI (green)* and *-DI (red)* lines are also at low levels, confirming market stagnation or very low directional momentum.

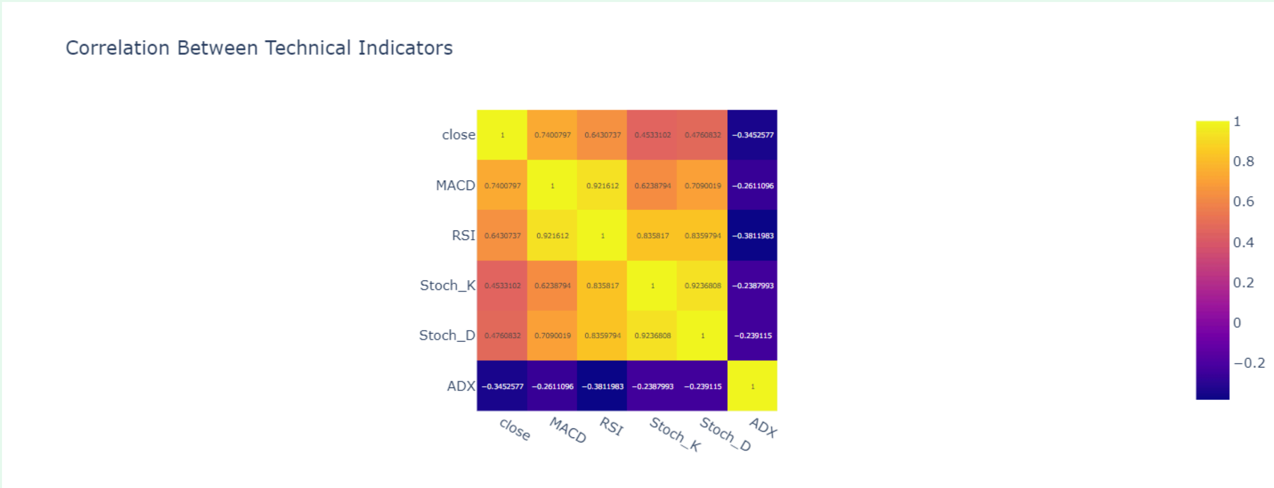
---

6. Correlation Heatmap

- The correlation heatmap between these above used indicators helps analyze how they move in relation to one another, aiding in strategy development and identifying potential signals for trading.

```
technical_indicators = data[['close', 'MACD', 'RSI', 'Stoch_K', 'Stoch_D', 'ADX']]

fig_corr = px.imshow(technical_indicators.corr(), text_auto=True, title='Correlation Between Technical Indicators')

fig_corr.show()
```

Output



💧 Observation

This heatmap represents the correlation between various technical indicators, such as *ADX,* ***RSI,*** *MACD,* ***Stoch_K,*** *Stoch_D, and **Close Price*. The color gradient shows the strength and direction of the correlation:

- *Yellow (Close to 1):* Strong positive correlation.
- *Purple (Close to -1):* Strong negative correlation.
- *Neutral Shades:* Weak or no correlation.

---

1. *Close Price vs Other Indicators:*

- *MACD (0.81):* Shows a strong positive correlation with the closing price, indicating that MACD aligns well with price movements.
- *RSI (0.72):* Also positively correlated with the closing price, suggesting that RSI reflects overbought/oversold conditions effectively in relation to price changes.
- *Stoch_K (0.62) and Stoch_D (0.65):* Moderate positive correlation, showing they are decent but less reliable in predicting price movements compared to MACD or RSI.
- *ADX (-0.42):* Displays a weak negative correlation with the closing price, indicating that ADX measures trend strength but not directly price direction.

2. *MACD vs Other Indicators:*

- *RSI (0.95):* Extremely high positive correlation, suggesting that both indicators align closely in identifying momentum and trends.

- *Stoch_K (0.71) and Stoch_D (0.79):* Moderate to strong positive correlation, indicating that MACD and stochastic indicators often complement each other in trend analysis.
- *ADX (-0.52):* Moderate negative correlation, showing that MACD and ADX measure different aspects of the market (trend direction vs trend strength).

3. *RSI vs Other Indicators:*

- *Stoch_K (0.86) and Stoch_D (0.87):* Strong positive correlation, reflecting that RSI and stochastic indicators are similarly effective in identifying overbought/oversold conditions.
- *ADX (-0.66):* Strong negative correlation, highlighting that RSI's momentum analysis often contrasts with ADX's trend strength measurements.

4. *Stoch_K vs Stoch_D:*

- *(0.92):* Very strong positive correlation, as expected, since both are derived from similar calculations and measure similar aspects of market momentum.

5. *ADX vs Other Indicators:*

- *Close (-0.42), MACD (-0.52), RSI (-0.66), Stoch_K (-0.54), Stoch_D (-0.57):* ADX has a generally negative correlation with all other indicators. This is because ADX focuses on *trend strength* rather than the direction of price or momentum, often providing contrasting signals.

Key Takeaways:

- *MACD and RSI* are highly correlated, making them reliable complementary indicators for identifying momentum and trend direction.
- *ADX* works independently of most other indicators, focusing on trend strength rather than price movements or overbought/oversold conditions.
- *Stoch_K and Stoch_D* are closely related and provide similar insights, useful for confirming RSI signals.
- The closing price correlates best with *MACD* and *RSI*, suggesting these indicators are the most effective for aligning technical analysis with price movement.