```
# Import libraries
dbutils.library.installPyPI("pandas_datareader")
dbutils.library.installPyPI("avro")

dbutils.library.restartPython()
from pandas_datareader import data
from pandas_datareader._utils import RemoteDataError
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
from pyspark.sql import *
import avro
from avro.datafile import DataFileWriter, DataFileReader
from avro.io import DatumWriter, DatumReader

# Stock tickers
NASDAQ_INDEX = '^IXIC'
TMUS_STOCK = 'TMUS'

# Some common statistics
def get_stats(stock_data):
  return {
    'last': np.mean(stock_data.tail(1)),

    # using only mean values:
    'short_mean': np.mean(stock_data.tail(30)),
    'long_mean': np.mean(stock_data.tail(300)),

    # using all values:
    'short_rolling': stock_data.rolling(window=30).mean(),
    'long_rolling': stock_data.rolling(window=300).mean()
  }

def get_data(ticker, START_DATE, END_DATE):
  try:
    # Get stock close data
    stock_data = data.DataReader(ticker, 'yahoo', START_DATE, END_DATE) # Gets:
open, close, high, low, volume, adj close

    # Filter records
    business_days = pd.date_range(start=START_DATE, end=END_DATE, freq='B') #
Stores filtered days in business_days
    stock_data = stock_data.reindex(business_days) # Reindex weekdays &
propogate NaN for missing values
    stock_data = stock_data.dropna() # Remove NaN values
    # Optionally, replace NaN values with latest available price via
'stock_data = stock_data.fillna(method='ffill')'

    # Change date from index to first column
    stock_data.reset_index(level=0, inplace=True)
    stock_data = stock_data.rename(columns={'index': 'Date'})

    return stock_data

  except RemoteDataError:
    print('No data found for {t}'.format(t=ticker))

# Get percent change for any stat
def get_pctChange(ticker, stat, START_DATE, END_DATE, interval_days):

  # Backtrack start date by #intervaldays to get pct change for first
#intervaldays
  # Example: if interval = 7 days, extend start date by backtracking 7 days
  start = START_DATE
  current_start = datetime.fromisoformat(start).date()
  ADJUSTED_START = current_start - timedelta(days=interval_days)

  # Get stock data
  stock_data = get_data(ticker, ADJUSTED_START, END_DATE)

  # Rename columns
  stat_column_name = ticker+stat
  stock_data = stock_data.rename(columns={'index': 'Date', stat:
stat_column_name})

  # Filter columns to just date and stat
  stock_data = stock_data[['Date', stat_column_name]]

  # List of stats for iteration
  statList = stock_data[stat_column_name]
  # New column to append
  pct_data = []

  # Adjust interval to exclude holidays
  business_days = np.busday_count(ADJUSTED_START, START_DATE)
  adj_interval = interval_days-(interval_days-business_days)

  # Calculate % change
  for n in range(len(statList)):
    price_index = n
    prev_index = n-adj_interval
```

```
      if prev_index < 0:
        pct_data.append('NA')
      elif price_index > (len(statList)-1):
        pct_data.append('NA')
      else:
        pct = ((statList[price_index] - statList[prev_index]) /
(statList[prev_index])) * 100

        if pct >= 5 and pct <= 10:
          pct_data.append(1)
        elif pct <= -5 and pct >= -10:
          pct_data.append(0)
        else:
          pct_data.append(2)

  # Readjust columns to exclude extra days from ADJUSTED_START calculation
  pct_data = pct_data[adj_interval:]
  stock_data = stock_data[adj_interval:]

  # Append new column and return new dataframe
  stock_data['PctChange'] = pct_data
  return stock_data

# Plot statistics
def create_plot(stock_data, ticker, stat, stat_type, title):
  stats = get_stats(stock_data) # Get stats for a specific stat
  plt.subplots(figsize=(12,8))
  plt.plot(stock_data, label=ticker)
  plt.plot(stats[stat_type], label=title)
  plt.xlabel('Date')
  plt.ylabel(stat)
  plt.legend()
  plt.title('Stock Price over Time')
  plt.show()

# Get stock data
tmus_data_df = get_data(TMUS_STOCK, '2020-06-04', '2021-06-04')
display(tmus_data_df)

nasdaq_data_df = get_data(NASDAQ_INDEX, '2020-06-04', '2021-06-04')
display(nasdaq_data_df)

# Get close data and weekly percent change
tmus_data_pct = get_pctChange(TMUS_STOCK, 'Close', '2020-06-04', '2021-06-04',
7)
display(tmus_data_pct)
```

```
nasdaq_data_pct = get_pctChange(NASDAQ_INDEX, 'Close', '2020-06-04', '2021-06-
04', 7)
display(nasdaq_data_pct)

# Plot data
tmus_plot = tmus_data_df.set_index('Date') # Reindex date for plot x axis
create_plot(tmus_plot['Close'], TMUS_STOCK, 'Close', 'short_rolling', '30 Day
Moving Avg')

nasdaq_plot = nasdaq_data_df.set_index('Date') # Reindex date for plot x axis
create_plot(nasdaq_plot['Close'], NASDAQ_INDEX, 'Close', 'short_rolling', '30
Day Moving Avg')

# -----------------------------------------------------------------------------
-------------
# METHOD 1: JOIN TABLES BY 'DATE' COLUMN (AS PER INSTRUCTIONS) ----------------
-------------
# -----------------------------------------------------------------------------
-------------

# Merge nasdaq and tmus dataframes
nasdaq_tmus = pd.merge(left = nasdaq_data_pct, right = tmus_data_pct,
left_on='Date', right_on='Date')
nasdaq_tmus = nasdaq_tmus.rename(columns={'PctChange_x': '^IXIC_PctChange',
'PctChange_y': 'TMUS_PctChange'})
display(nasdaq_tmus)

# Create dataframe for spark operations
df = spark.createDataFrame(nasdaq_tmus)

# Partition data - parquet format
dbutils.fs.rm('/DBFS/',True) # Remove written data
df.write.partitionBy('^IXIC_PctChange').parquet('DBFS/nasdaq_partitions_PQ')
df.write.partitionBy('TMUS_PctChange').parquet('DBFS/tmus_partitions_PQ')

# Bucket data - parquet format
dbutils.fs.rm('/user/',True) # Remove written data
df.write.bucketBy(10, '^IXIC_PctChange').saveAsTable('nasdaq_buckets2',
format='parquet')
df.write.bucketBy(10, 'TMUS_PctChange').saveAsTable('tmus_buckets2',
format='parquet')

# Avro format -------------------------

# Rename dataframe columns for avro format
renamed_df = nasdaq_tmus.rename(columns={'^IXICClose': 'NASDAQClose',
'^IXIC_PctChange': 'NASDAQ_PctChange'})
```

```
# Reindex dataframe as string indices for avro format
renamed_df.set_index([['val']*len(renamed_df)], inplace=True)

# Convert dataframe to dictionary
dict_df = renamed_df.to_dict()

# Convert dictionary to json file
json_df = (pd.DataFrame(dict_df)).to_json()

# Convert json file to spark data frame
spark_json_df = spark.read.json(sc.parallelize([json_df]))
spark_json_df.write.json('DBFS/test')

# Convert spark data frame to avro
dbutils.fs.rm('spark_json_df.avro',True) # Remove written data
spark_json_df.write.format("avro").save("spark_json_df.avro")

# ------------------------------------------------------------------------------
-------------
# METHOD 2: JOIN TABLES BY 'CLOSE' COLUMN (OPTIMIZED/SCALABLE) ----------------
-------------
# ------------------------------------------------------------------------------
-------------

def insert_stockID(stock_data, id_name):
  new_df = stock_data.insert(0, 'stockID', id_name)
  return new_df

# Insert stockID as new column
insert_stockID(nasdaq_data_pct, 'nasdaq')
insert_stockID(tmus_data_pct, 'tmus')

# Rename Close columns for merging
nasdaq_data_pct1 = nasdaq_data_pct.rename(columns={'^IXICClose': 'Close'})
display(nasdaq_data_pct1)

tmus_data_pct1 = tmus_data_pct.rename(columns={'TMUSClose': 'Close'})
display(tmus_data_pct1)

# Merge nasdaq and tmus dataframes - combine 'Close' column and sort by date
nasdaq_tmus1 = pd.concat([nasdaq_data_pct1, tmus_data_pct1], axis=0)
nasdaq_tmus1 = nasdaq_tmus1.sort_values(by=['Date'])
display(nasdaq_tmus1)

# Create dataframe for spark operations
df = spark.createDataFrame(nasdaq_tmus1)
```

```
# Partition data - parquet format
#dbutils.fs.rm('/DBFS/',True) # Remove written data
df.write.partitionBy('stockID').parquet('DBFS/version2_PQ')

# Bucket data - parquet format
#dbutils.fs.rm('/user/',True) # Remove written data
df.write.bucketBy(10, 'PctChange').saveAsTable('vers2_buckets',
format='parquet')

# Avro format -------------------------
# Convert dataframe to dictionary
reindex_df = nasdaq_tmus1.set_index('stockID') # reindex stockID for avro
format
dict_df = reindex_df.to_dict()

# Convert dictionary to json file
json_df = (pd.DataFrame(dict_df)).to_json()

# Convert json file to spark data frame
spark_json_df = spark.read.json(sc.parallelize([json_df]))
spark_json_df.write.json('DBFS/versn2')

# Convert spark data frame to avro
dbutils.fs.rm('version2.avro',True) # Remove written data
spark_json_df.write.format("avro").save("version2.avro")
```

```
dbutils.library APIs are deprecated and will be removed in a future DBR releas
e. You can use %pip and %conda commands to install notebook scoped python libr
aries. For more information see https://docs.databricks.com/libraries/notebook
s-python-libraries.html.
PyPI package pandas_datareader has been installed already. The previously inst
alled package is `pandas_datareader`. To resolve this issue, detach and re-att
ach the notebook to create a new environment or rename the package.
dbutils.library APIs are deprecated and will be removed in a future DBR releas
e. You can use %pip and %conda commands to install notebook scoped python libr
aries. For more information see https://docs.databricks.com/libraries/notebook
s-python-libraries.html.
PyPI package avro has been installed already. The previously installed package
is `avro`. To resolve this issue, detach and re-attach the notebook to create
a new environment or rename the package.
```

	Date	High	Low	Open
1	2020-06-04T00:00:00.000+0000	101.69000244140625	99.79000091552734	101.5
2	2020-06-05T00:00:00.000+0000	102.97000122070312	100.5	101.3
3	2020-06-08T00:00:00.000+0000	105.11000061035156	100.44000244140625	100.7
4				

4	2020-06-09T00:00:00.000+0000	103.54000091552734	101.7699966430664	102.1
5	2020-06-10T00:00:00.000+0000	106.72000122070312	102.80999755859375	102.8
6	2020-06-11T00:00:00.000+0000	105.30999755859375	100.73999786376953	104.5
7	2020-06-12T00:00:00.000+0000	105.05000305175781	100.26000213623047	104.2
8	2020-06-15T00:00:00.000+0000	105.19000244140625	100.7300033569336	101.3
9	2020-06-16T00:00:00.000+0000	104	100.11000061035156	104
10	2020-06-17T00:00:00.000+0000	104.80000305175781	102.33999633789062	103
11	2020-06-18T00:00:00.000+0000	106.83999633789062	103.25	103.8
12	2020-06-19T00:00:00.000+0000	109	106.38999938964844	107.3
13	2020-06-22T00:00:00.000+0000	108.69999694824219	106.02999877929688	106.8
14	2020-06-23T00:00:00.000+0000	110.45999908447266	103.5	103.6
15	2020-06-24T00:00:00.000+0000	109.13999938964844	104.56999969482422	105.4
16	2020-06-25T00:00:00.000+0000	111.58000183105469	108.51000213623047	109.7
17	2020-06-26T00:00:00.000+0000	109.8499984741211	104.7300033569336	109.1
18	2020-06-29T00:00:00.000+0000	106.26000213623047	103.94999694824219	105.7
19	2020-06-30T00:00:00.000+0000	106.19999694824219	104	105.9

Showing all 253 rows.

	Date	High	Low	Open
1	2020-06-04T00:00:00.000+0000	9716.1396484375	9560.41015625	9649.6503
2	2020-06-05T00:00:00.000+0000	9845.6904296875	9685.349609375	9703.5400
3	2020-06-08T00:00:00.000+0000	9927.1298828125	9780.6103515625	9823.4404
4	2020-06-09T00:00:00.000+0000	10002.5	9863.26953125	9867.1904
5	2020-06-10T00:00:00.000+0000	10086.8896484375	9962.580078125	10012.320
6	2020-06-11T00:00:00.000+0000	9868.080078125	9491.3095703125	9791.2402
7	2020-06-12T00:00:00.000+0000	9768.6396484375	9413.6201171875	9715.8701
8	2020-06-15T00:00:00.000+0000	9756.0703125	9403	9426.9003
9	2020-06-16T00:00:00.000+0000	9963.6298828125	9748.3798828125	9949.7802
10	2020-06-17T00:00:00.000+0000	9991.2099609375	9891.8095703125	9943.3095
11	2020-06-18T00:00:00.000+0000	9959.2001953125	9885.66015625	9892.4804
12	2020-06-19T00:00:00.000+0000	10053.91015625	9872.9404296875	10042.129
13	2020-06-22T00:00:00.000+0000	10059.6103515625	9916.599609375	9945.4902
14	2020-06-23T00:00:00.000+0000	10221.849609375	10112.4404296875	10130.830
15	2020-06-24T00:00:00.000+0000	10137.5	9842.2197265625	10092.919
16	2020-06-25T00:00:00.000+0000	10023.2802734375	9810.4697265625	9899.3603
17	2020-06-26T00:00:00.000+0000	10000.669921875	9749.0703125	9995.1201
18	2020-06-29T00:00:00.000+0000	9877.33984375	9663.6103515625	9771.7197

19	2020-06-30T00:00:00.000+0000	10085.58984375	9863.669921875	9875.2900

Showing all 253 rows.

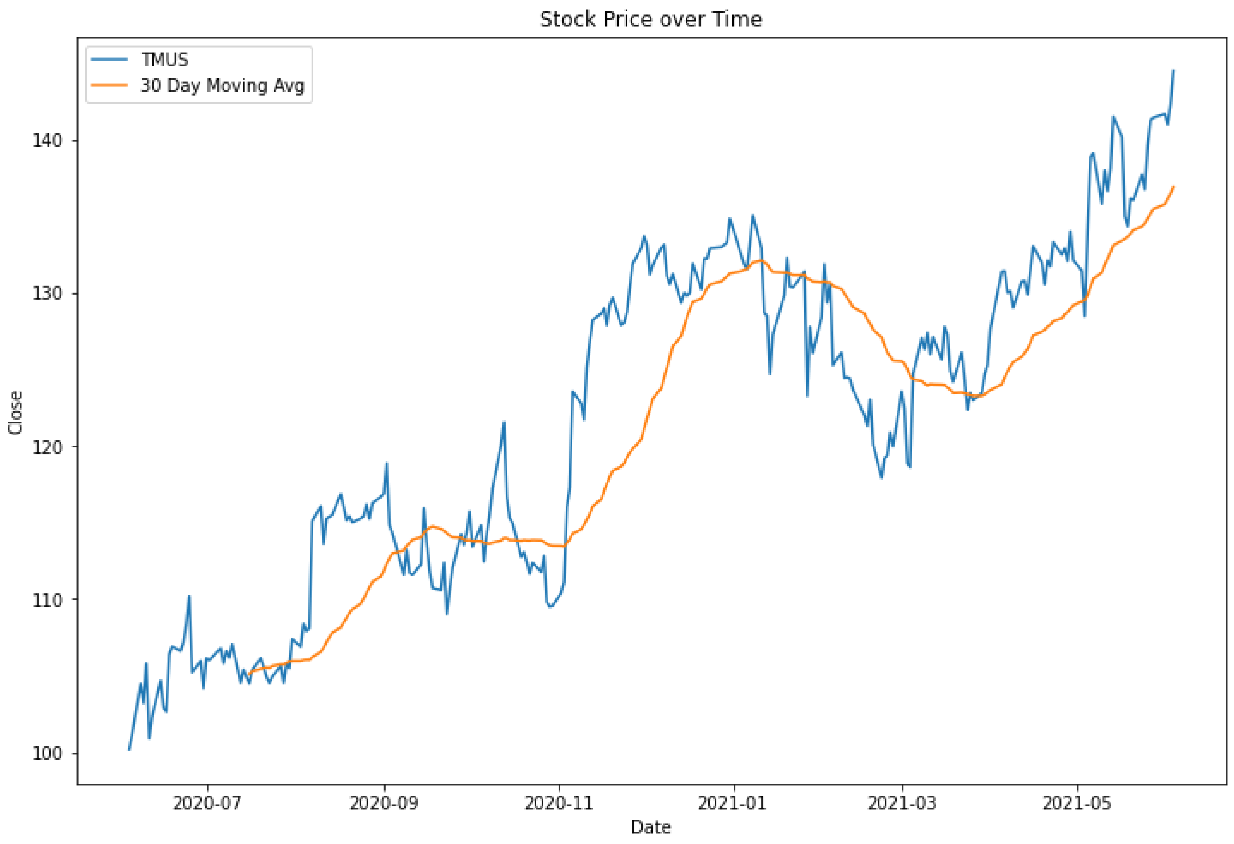
	Date	TMUSClose	PctChange
1	2020-06-04T00:00:00.000+0000	100.16999816894531	2
2	2020-06-05T00:00:00.000+0000	101.12999725341797	2
3	2020-06-08T00:00:00.000+0000	104.45999908447266	2
4	2020-06-09T00:00:00.000+0000	103.19000244140625	2
5	2020-06-10T00:00:00.000+0000	105.79000091552734	2
6	2020-06-11T00:00:00.000+0000	100.88999938964844	2
7	2020-06-12T00:00:00.000+0000	102.30999755859375	2
8	2020-06-15T00:00:00.000+0000	104.68000030517578	2
9	2020-06-16T00:00:00.000+0000	102.88999938964844	2
10	2020-06-17T00:00:00.000+0000	102.62999725341797	2
11	2020-06-18T00:00:00.000+0000	106.38999938964844	1
12	2020-06-19T00:00:00.000+0000	106.9000015258789	2
13	2020-06-22T00:00:00.000+0000	106.5999984741211	2
14	2020-06-23T00:00:00.000+0000	107.16000366210938	2
15	2020-06-24T00:00:00.000+0000	108.43000030517578	1
16	2020-06-25T00:00:00.000+0000	110.19000244140625	2
17	2020-06-26T00:00:00.000+0000	105.19999694824219	2
18	2020-06-29T00:00:00.000+0000	105.95999908447266	2
19	2020-06-30T00:00:00.000+0000	104.1500015258789	2

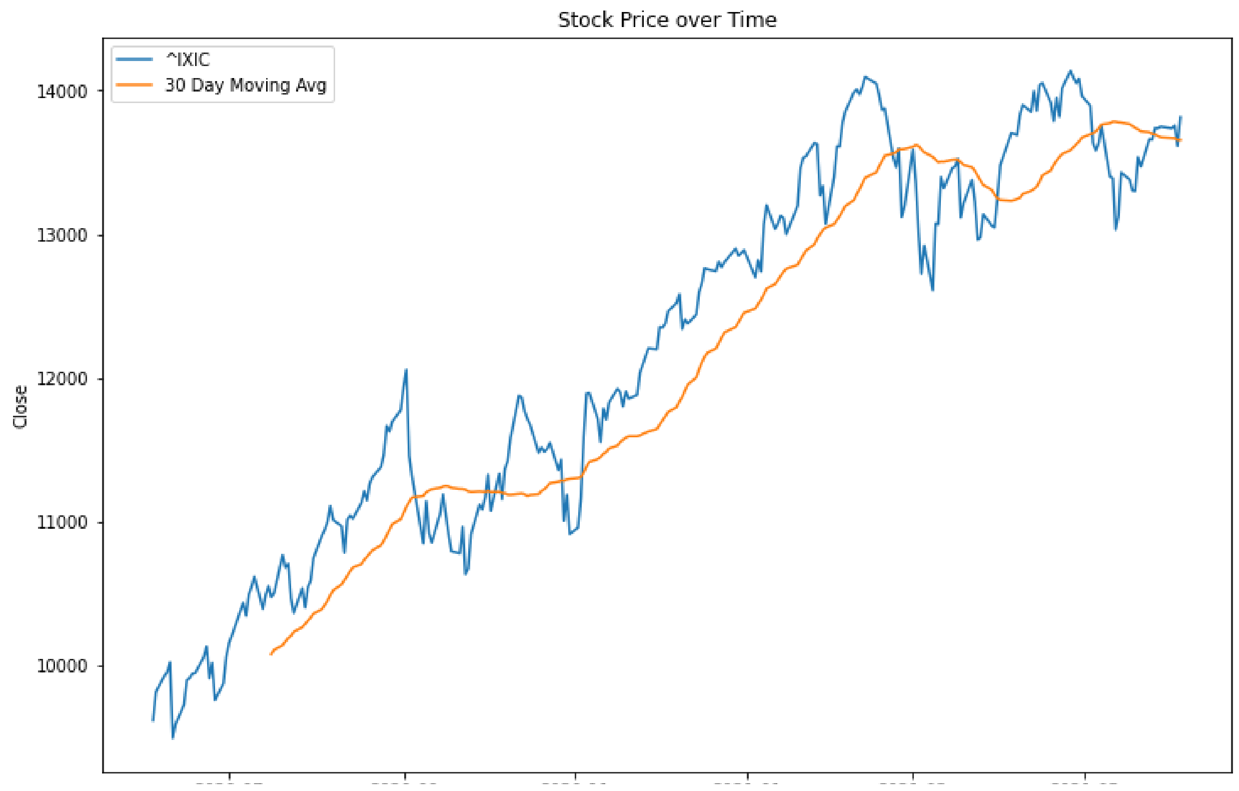
Showing all 253 rows.

	Date	^IXICClose	PctChange
1	2020-06-04T00:00:00.000+0000	9615.8095703125	2
2	2020-06-05T00:00:00.000+0000	9814.080078125	2
3	2020-06-08T00:00:00.000+0000	9924.75	2
4	2020-06-09T00:00:00.000+0000	9953.75	2
5	2020-06-10T00:00:00.000+0000	10020.349609375	2
6	2020-06-11T00:00:00.000+0000	9492.73046875	2
7	2020-06-12T00:00:00.000+0000	9588.8095703125	2
8	2020-06-15T00:00:00.000+0000	9726.01953125	2
9	2020-06-16T00:00:00.000+0000	9895.8701171875	2
10	2020-06-17T00:00:00.000+0000	9910.5302734375	2

11	2020-06-18T00:00:00.000+0000	9943.0498046875	2
12	2020-06-19T00:00:00.000+0000	9946.1201171875	2
13	2020-06-22T00:00:00.000+0000	10056.48046875	2
14	2020-06-23T00:00:00.000+0000	10131.3701171875	2
15	2020-06-24T00:00:00.000+0000	9909.169921875	2
16	2020-06-25T00:00:00.000+0000	10017	2
17	2020-06-26T00:00:00.000+0000	9757.2197265625	2
18	2020-06-29T00:00:00.000+0000	9874.150390625	2
19	2020-06-30T00:00:00.000+0000	10058.76953125	2

Showing all 253 rows.





	Date	^IXICClose	^IXIC_PctChange	TMUSCl
1	2020-06-04T00:00:00.000+0000	9615.8095703125	2	100.169
2	2020-06-05T00:00:00.000+0000	9814.080078125	2	101.129
3	2020-06-08T00:00:00.000+0000	9924.75	2	104.459
4	2020-06-09T00:00:00.000+0000	9953.75	2	103.190
5	2020-06-10T00:00:00.000+0000	10020.349609375	2	105.790
6	2020-06-11T00:00:00.000+0000	9492.73046875	2	100.889
7	2020-06-12T00:00:00.000+0000	9588.8095703125	2	102.309
8	2020-06-15T00:00:00.000+0000	9726.01953125	2	104.680
9	2020-06-16T00:00:00.000+0000	9895.8701171875	2	102.889
10	2020-06-17T00:00:00.000+0000	9910.5302734375	2	102.629
11	2020-06-18T00:00:00.000+0000	9943.0498046875	2	106.389
12	2020-06-19T00:00:00.000+0000	9946.1201171875	2	106.900
13	2020-06-22T00:00:00.000+0000	10056.48046875	2	106.599
14	2020-06-23T00:00:00.000+0000	10131.3701171875	2	107.160
15	2020-06-24T00:00:00.000+0000	9909.169921875	2	108.430
16	2020-06-25T00:00:00.000+0000	10017	2	110.190
17	2020-06-26T00:00:00.000+0000	9757.2197265625	2	105.199
18	2020-06-29T00:00:00.000+0000	9874.150390625	2	105.959
19	2020-06-30T00:00:00.000+0000	10058.76953125	2	104.150

Showing all 253 rows.

	stockID	Date	Close	PctChange
1	nasdaq	2020-06-04T00:00:00.000+0000	9615.8095703125	2
2	nasdaq	2020-06-05T00:00:00.000+0000	9814.080078125	2
3	nasdaq	2020-06-08T00:00:00.000+0000	9924.75	2
4	nasdaq	2020-06-09T00:00:00.000+0000	9953.75	2
5	nasdaq	2020-06-10T00:00:00.000+0000	10020.349609375	2
6	nasdaq	2020-06-11T00:00:00.000+0000	9492.73046875	2
7	nasdaq	2020-06-12T00:00:00.000+0000	9588.8095703125	2
8	nasdaq	2020-06-15T00:00:00.000+0000	9726.01953125	2
9	nasdaq	2020-06-16T00:00:00.000+0000	9895.8701171875	2
10	nasdaq	2020-06-17T00:00:00.000+0000	9910.5302734375	2
11	nasdaq	2020-06-18T00:00:00.000+0000	9943.0498046875	2
12	nasdaq	2020-06-19T00:00:00.000+0000	9946.1201171875	2
13	nasdaq	2020-06-22T00:00:00.000+0000	10056.48046875	2
14	nasdaq	2020-06-23T00:00:00.000+0000	10131.3701171875	2
15	nasdaq	2020-06-24T00:00:00.000+0000	9909.169921875	2
16	nasdaq	2020-06-25T00:00:00.000+0000	10017	2
17	nasdaq	2020-06-26T00:00:00.000+0000	9757.2197265625	2
18	nasdaq	2020-06-29T00:00:00.000+0000	9874.150390625	2
19	nasdaq	2020-06-30T00:00:00.000+0000	10058.76953125	2

Showing all 253 rows.

	stockID	Date	Close	PctChange
1	tmus	2020-06-04T00:00:00.000+0000	100.16999816894531	2
2	tmus	2020-06-05T00:00:00.000+0000	101.12999725341797	2
3	tmus	2020-06-08T00:00:00.000+0000	104.45999908447266	2
4	tmus	2020-06-09T00:00:00.000+0000	103.19000244140625	2
5	tmus	2020-06-10T00:00:00.000+0000	105.79000091552734	2
6	tmus	2020-06-11T00:00:00.000+0000	100.88999938964844	2
7	tmus	2020-06-12T00:00:00.000+0000	102.30999755859375	2
8	tmus	2020-06-15T00:00:00.000+0000	104.68000030517578	2
9	tmus	2020-06-16T00:00:00.000+0000	102.88999938964844	2
10	tmus	2020-06-17T00:00:00.000+0000	102.62999725341797	2
11	tmus	2020-06-18T00:00:00.000+0000	106.38999938964844	1

12	tmus	2020-06-19T00:00:00.000+0000	106.9000015258789	2
13	tmus	2020-06-22T00:00:00.000+0000	106.5999984741211	2
14	tmus	2020-06-23T00:00:00.000+0000	107.16000366210938	2
15	tmus	2020-06-24T00:00:00.000+0000	108.43000030517578	1
16	tmus	2020-06-25T00:00:00.000+0000	110.19000244140625	2
17	tmus	2020-06-26T00:00:00.000+0000	105.19999694824219	2
18	tmus	2020-06-29T00:00:00.000+0000	105.95999908447266	2
19	tmus	2020-06-30T00:00:00.000+0000	104.1500015258789	2

Showing all 253 rows.

	stockID	Date	Close	PctChange
1	nasdaq	2020-06-04T00:00:00.000+0000	9615.8095703125	2
2	tmus	2020-06-04T00:00:00.000+0000	100.16999816894531	2
3	nasdaq	2020-06-05T00:00:00.000+0000	9814.080078125	2
4	tmus	2020-06-05T00:00:00.000+0000	101.12999725341797	2
5	nasdaq	2020-06-08T00:00:00.000+0000	9924.75	2
6	tmus	2020-06-08T00:00:00.000+0000	104.45999908447266	2
7	nasdaq	2020-06-09T00:00:00.000+0000	9953.75	2
8	tmus	2020-06-09T00:00:00.000+0000	103.19000244140625	2
9	nasdaq	2020-06-10T00:00:00.000+0000	10020.349609375	2
10	tmus	2020-06-10T00:00:00.000+0000	105.79000091552734	2
11	nasdaq	2020-06-11T00:00:00.000+0000	9492.73046875	2
12	tmus	2020-06-11T00:00:00.000+0000	100.88999938964844	2
13	nasdaq	2020-06-12T00:00:00.000+0000	9588.8095703125	2
14	tmus	2020-06-12T00:00:00.000+0000	102.30999755859375	2
15	tmus	2020-06-15T00:00:00.000+0000	104.68000030517578	2
16	nasdaq	2020-06-15T00:00:00.000+0000	9726.01953125	2
17	tmus	2020-06-16T00:00:00.000+0000	102.88999938964844	2
18	nasdaq	2020-06-16T00:00:00.000+0000	9895.8701171875	2
19	tmus	2020-06-17T00:00:00.000+0000	102.62999725341797	2

Showing all 506 rows.

Data Anlaysis (see plots)

An exploration of the data has revelaed the following meaningful insights:

Seasonal effects on the stock market: Stock markets tend to perform well at the beginning of the year as this is when many investors have fresh capital to place into the market. They are therefore more likely to buy shares and push up prices. Subsequently, January is often a volatile month for share prices with large, erratic price moves as trader activity surges. The month is also closely watched because many traders believe that how stock markets perform in January will foretell their performance for the rest of the year. As expected, the graphs here show a jump in the closing price for TMUS in January 2021.

Effects on TMUS business decisions on TMUS stock: Share prices tend to fall over the summer months as fund managers and big institutional traders go on holiday. They often sell some of their shares and other assets before they go away, so that their investments are at less risk of taking a big hit if markets fall suddenly while they are not at their trading screens to respond quickly. This usually results in a dip in stock prices in the summer. However, the opposite happened to TMUS. The merger of Spring and TMUS finalized in April 2020, combining in an all shares deal at $26 billion. This led to an increase in TMUS's stock in April 2020, which continued to increase throughout June and subsequent months, as seen in the trend - TMUS's stock in April was in the $90s and rose to the $100s in June.

Correlations between NASDAQ and TMUS: NASDAQ saw a sharp increase in value from November 2020 to March 2023, and this pattern is reflected in an upward trend that TMUS witnessed as its stock also increased from November 2020, but it only rose until January of 2021. In general, however, TMUS tends to follow the overall trend that NASDAQ does. As NASDAQ rises, so does TMUS, and vice versa. This can also be noted in the gradual increase in NASDAQ from September to November 2020, which is also reflected in TMUS.

```
#Read Json file

#testJsonData = spark.read.json("...")

#display(testJsonData)
```

```
# Read Avro file

#testAvroData = spark.read.format('avro').load('...')

#display(testAvroData)
```

```
Out[1]: '\nfile_location = "/..."\n\nfile_type = "json"\n\n# CSV options\ninfe
r_schema = "false"\nfirst_row_is_header = "true"\ndelimiter = ","\n\n# The app
lied options are for CSV files. For other file types, these will be ignored.\n
df = spark.read.format(file_type)   .option("inferSchema", infer_schema)   .op
tion("header", first_row_is_header)   .option("sep", delimiter)   .load(file_l
ocation)\n\ndisplay(df)\n'
```