Analysis of Coca cola Stock

December 9, 2024

[41]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

[43]:
```
pd.read_csv("/Users/avinashhm/Desktop/Coca Cola.csv")
```

[43]:
```
          Date       Open       High        Low      Close  Adj Close  \
0     02/01/19  46.939999  47.220001  46.560001  46.930000  39.828789
1     03/01/19  46.820000  47.369999  46.529999  46.639999  39.582672
2     04/01/19  46.750000  47.570000  46.639999  47.570000  40.371952
3     07/01/19  47.570000  47.750000  46.900002  46.950001  39.845768
4     08/01/19  47.250000  47.570000  47.040001  47.480000  40.295567
…          …        …          …          …          …          …
1253  22/12/23  58.119999  58.459999  58.020000  58.320000  57.857216
1254  26/12/23  58.060001  58.709999  58.060001  58.560001  58.095314
1255  27/12/23  58.639999  58.770000  58.400002  58.709999  58.244122
1256  28/12/23  58.650002  58.869999  58.529999  58.750000  58.283806
1257  29/12/23  58.740002  58.980000  58.630001  58.930000  58.462376

        Volume  YEAR  Month  Day
0     11603700  2019      1    2
1     14714400  2019      1    3
2     13013700  2019      1    4
3     13135500  2019      1    7
4     15420700  2019      1    8
…          …     …      …    …
1253   9028500  2023     12   22
1254   6422500  2023     12   26
1255   8560100  2023     12   27
1256   8400100  2023     12   28
1257   9241600  2023     12   29

[1258 rows x 10 columns]
```

[45]:
```
data= pd.read_csv("/Users/avinashhm/Desktop/Coca Cola.csv")
```

[50]:
```
data['Date'] = pd.to_datetime(data['Date'])
data.sort_values('Date', inplace=True)
```

Daily Return

[52]:
```
data['Daily Return'] = data['Adj Close'].pct_change()
```

[54]:
```
data['Daily Return']
```

[54]:
```
21            NaN
40      -0.068172
61       0.038516
82       0.040026
124      0.070237
           …
1253     0.005691
1254     0.004115
1255     0.002561
1256     0.000681
1257     0.003064
Name: Daily Return, Length: 1258, dtype: float64
```

CAGR Calculation

[57]:
```
start_price = data['Adj Close'].iloc[0]
end_price = data['Adj Close'].iloc[-1]
num_years = (data['Date'].iloc[-1] - data['Date'].iloc[0]).days / 365.25
cagr = ((end_price / start_price) ** (1 / num_years)) - 1
print(f"CAGR: {cagr:.2%}")
```

```
CAGR: 7.20%
```

Moving Averages

[60]:
```
data['50-Day MA'] = data['Adj Close'].rolling(window=50).mean()
data['200-Day MA'] = data['Adj Close'].rolling(window=200).mean()
```

[62]:
```
data['200-Day MA']
```

[62]:
```
21              NaN
40              NaN
61              NaN
82              NaN
124             NaN
          …
1253     58.362817
1254     58.374069
1255     58.377686
1256     58.380444
```

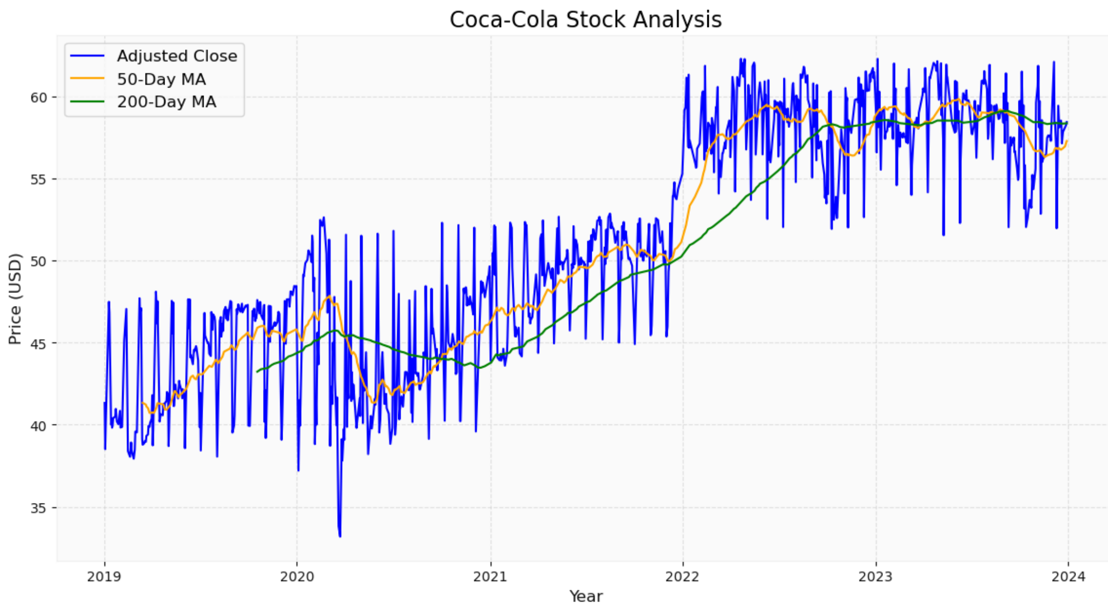
```
1257    58.382171
Name: 200-Day MA, Length: 1258, dtype: float64
```

[64]:
```
data['50-Day MA']
```

[64]:
```
21             NaN
40             NaN
61             NaN
82             NaN
124            NaN
          …
1253    56.858978
1254    56.979768
1255    57.092905
1256    57.194237
1257    57.299534
Name: 50-Day MA, Length: 1258, dtype: float64
```

Plot Trends

[67]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 7))
plt.plot(data['Date'], data['Adj Close'], label='Adjusted Close', color='blue')
plt.plot(data['Date'], data['50-Day MA'], label='50-Day MA', color='orange')
plt.plot(data['Date'], data['200-Day MA'], label='200-Day MA', color='green')
plt.title("Coca-Cola Stock Analysis", fontsize=16)
plt.xlabel("Year", fontsize=12)
plt.ylabel("Price (USD)", fontsize=12)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```



Calculate standard deviation of daily returns to measure volatility

[70]:
```
volatility = data['Daily Return'].std() * (252 ** 0.5)  # Annualized
print(f"Annualized Volatility: {volatility:.2%}")
```

```
Annualized Volatility: 73.37%
```

[ ]:

[ ]:

[ ]:

Compute Technical Indicators

[76]:
```
import pandas as pd
import numpy as np

# Load and preprocess data
data = pd.read_csv("Coca Cola.csv")
data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)

# Moving Averages
data['20-Day MA'] = data['Adj Close'].rolling(window=20).mean()
data['50-Day MA'] = data['Adj Close'].rolling(window=50).mean()
data['200-Day MA'] = data['Adj Close'].rolling(window=200).mean()
```

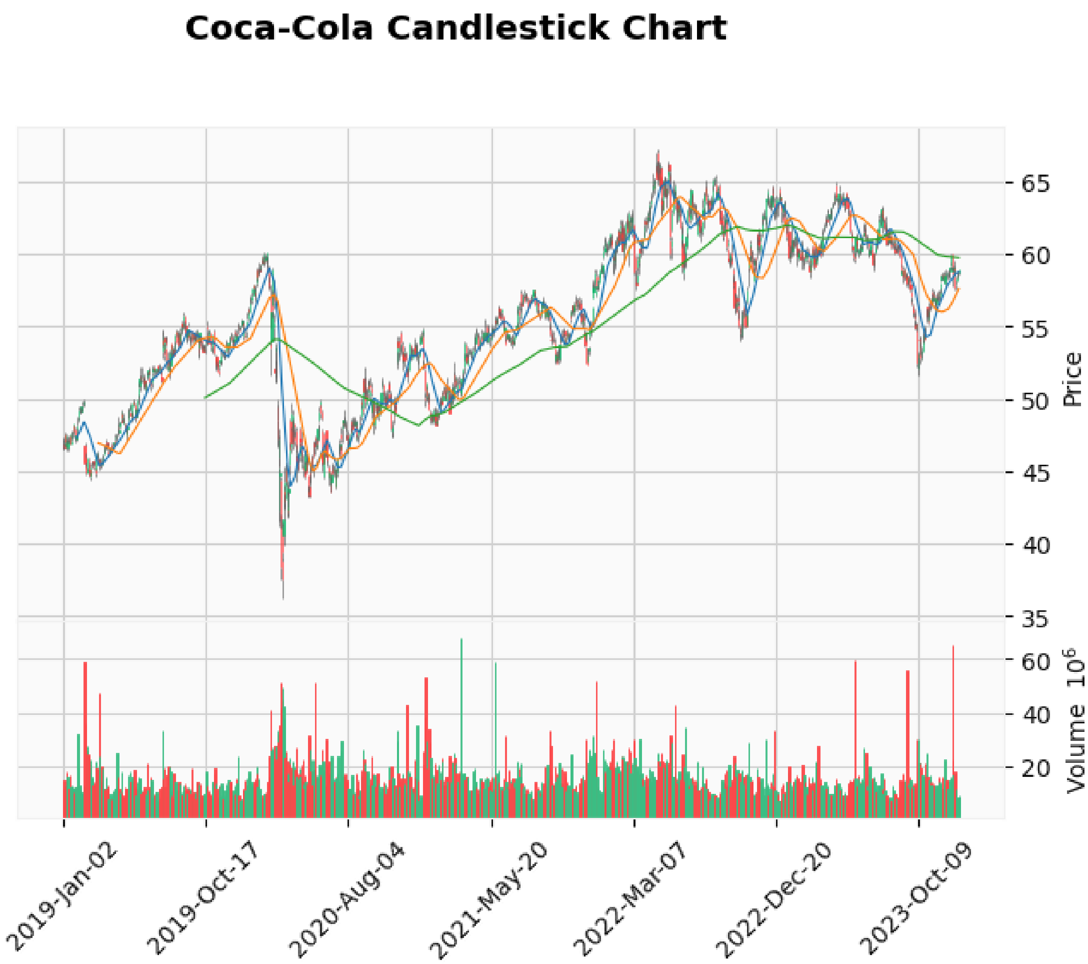
```
# RSI Calculation
delta = data['Adj Close'].diff()
gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
rs = gain / loss
data['RSI'] = 100 - (100 / (1 + rs))

# Bollinger Bands
data['Upper Band'] = data['20-Day MA'] + (data['Adj Close'].rolling(window=20).
 ↪std() * 2)
data['Lower Band'] = data['20-Day MA'] - (data['Adj Close'].rolling(window=20).
 ↪std())

# MACD
ema_12 = data['Adj Close'].ewm(span=12, adjust=False).mean()
ema_26 = data['Adj Close'].ewm(span=26, adjust=False).mean()
data['MACD'] = ema_12 - ema_26
data['Signal Line'] = data['MACD'].ewm(span=9, adjust=False).mean()
```

Candle stick Chart

[79]:
```
import mplfinance as mpf

# Plot candlestick chart with technical indicators
mpf.plot(
    data,
    type='candle',
    mav=(20, 50, 200),
    volume=True,
    title='Coca-Cola Candlestick Chart',
    style='yahoo'
)
```

```
/opt/anaconda3/lib/python3.12/site-packages/mplfinance/_arg_validators.py:84:
UserWarning:

 ================================================================

   WARNING: YOU ARE PLOTTING SO MUCH DATA THAT IT MAY NOT BE
            POSSIBLE TO SEE DETAILS (Candles, Ohlc-Bars, Etc.)
   For more information see:
   - https://github.com/matplotlib/mplfinance/wiki/Plotting-Too-Much-Data

   TO SILENCE THIS WARNING, set `type='line'` in `mpf.plot()`
   OR set kwarg `warn_too_much_data=N` where N is an integer
   LARGER than the number of data points you want to plot.

 ================================================================
```

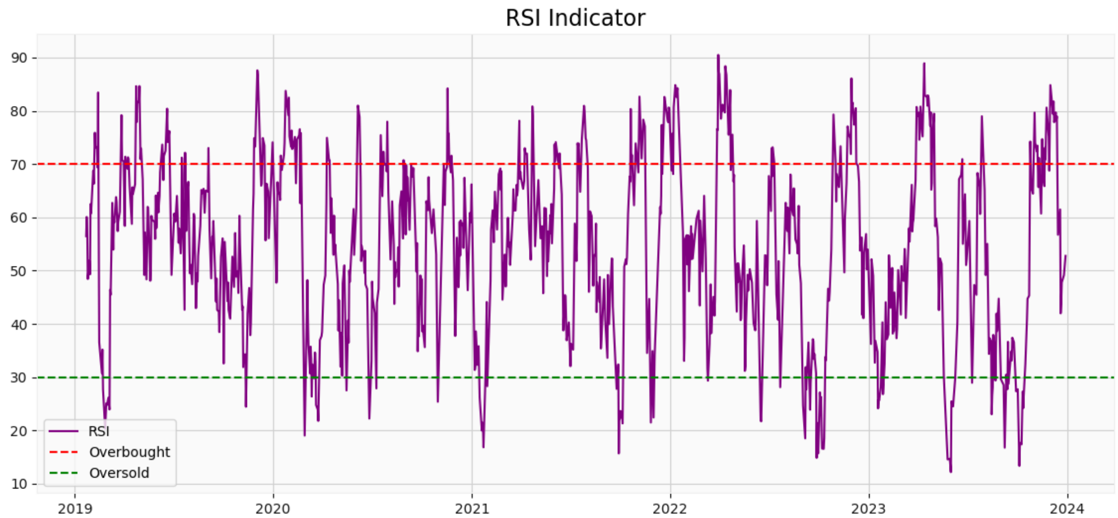
```
  warnings.warn('\n\n
================================================================ '+
```

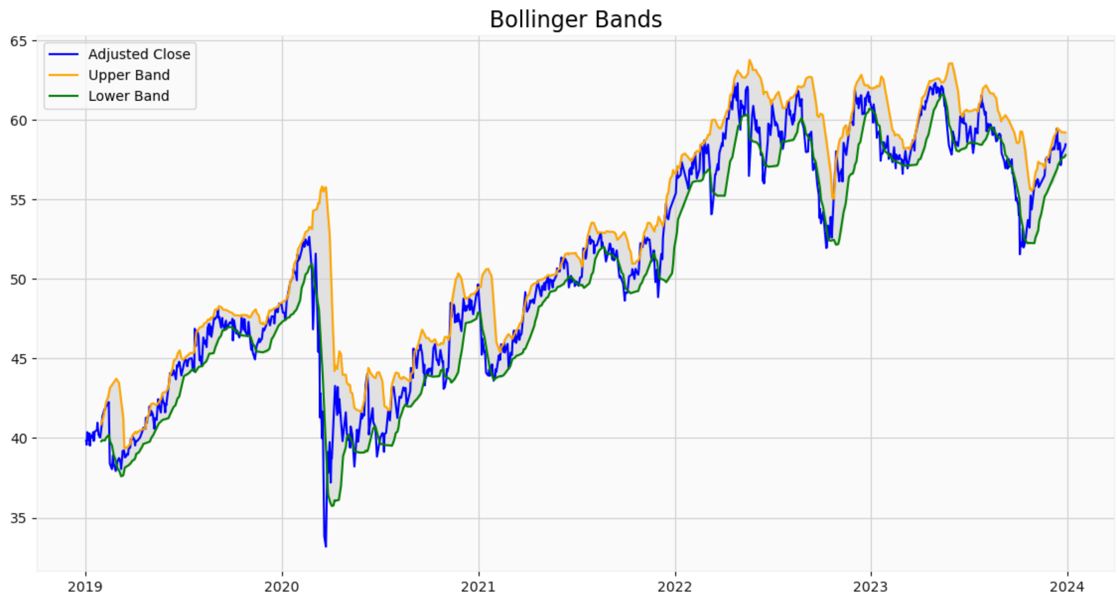


Visualize Indicators

[82]:
```python
import matplotlib.pyplot as plt

# RSI Plot
plt.figure(figsize=(14, 6))
plt.plot(data.index, data['RSI'], label='RSI', color='purple')
plt.axhline(70, color='red', linestyle='--', label='Overbought')
plt.axhline(30, color='green', linestyle='--', label='Oversold')
plt.title('RSI Indicator', fontsize=16)
plt.legend()
plt.show()

# Bollinger Bands Plot
```

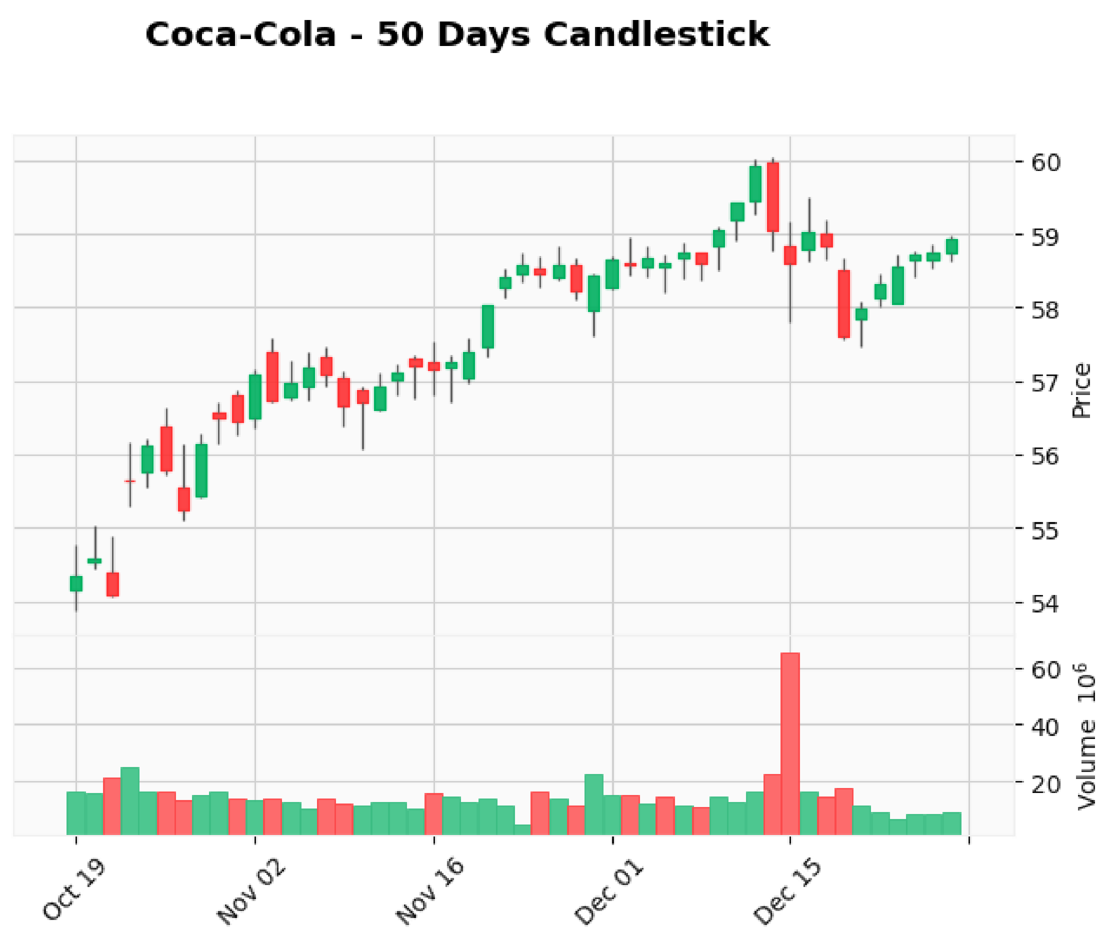
```
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Adj Close'], label='Adjusted Close', color='blue')
plt.plot(data.index, data['Upper Band'], label='Upper Band', color='orange')
plt.plot(data.index, data['Lower Band'], label='Lower Band', color='green')
plt.fill_between(data.index, data['Lower Band'], data['Upper Band'],␣
↪color='gray', alpha=0.2)
plt.title('Bollinger Bands', fontsize=16)
plt.legend()
plt.show()
```



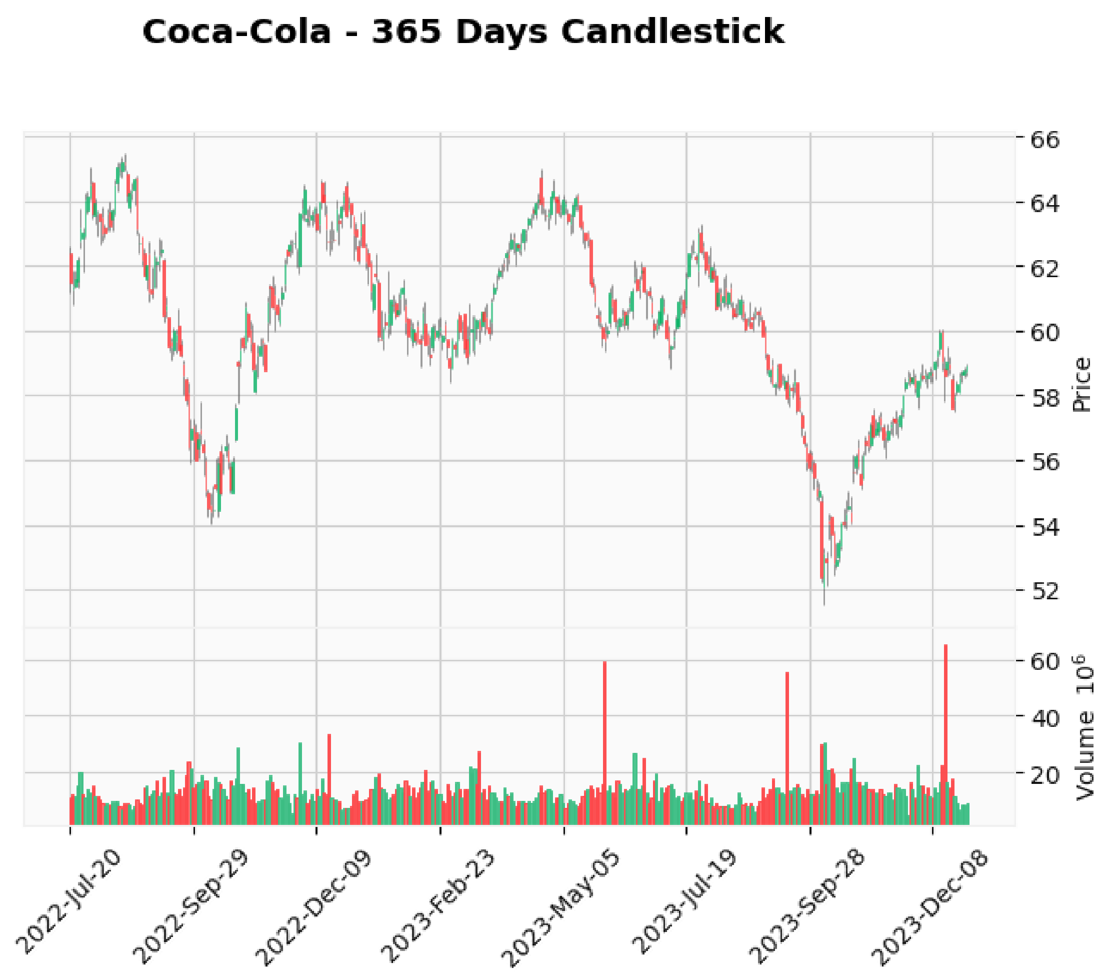


Summary of Analysis Candlestick Chart: Offers a clear visual representation of price movements (open, high, low, close). RSI: Identifies overbought (>70) and oversold (<30) conditions. Bollinger Bands: Highlights periods of high/low volatility. MACD: Provides buy/sell signals based on line crossovers.

[84]:
```
# Candlestick chart for the last 50 days
data_50_days = data.tail(50)
mpf.plot(data_50_days, type='candle', style='yahoo', title="Coca-Cola - 50 Days␣
↪Candlestick", volume=True)

# Candlestick chart for the last 200 days
data_200_days = data.tail(200)
mpf.plot(data_200_days, type='candle', style='yahoo', title="Coca-Cola - 200␣
↪Days Candlestick", volume=True)

#Candlestick chart for the last 365 days
data_365_days = data.tail(365)
mpf.plot(data_365_days, type='candle', style='yahoo', title="Coca-Cola - 365␣
↪Days Candlestick", volume=True)
```



Coca-Cola - 200 Days Candlestick



[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

Findings

[91]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the stock data (Make sure to update the path if necessary)
data = pd.read_csv('Coca Cola.csv')
data['Date'] = pd.to_datetime(data['Date'])
```

```
data.set_index('Date', inplace=True)

# Calculate moving averages
data['50-Day MA'] = data['Adj Close'].rolling(window=50).mean()
data['200-Day MA'] = data['Adj Close'].rolling(window=200).mean()

# Calculate RSI (Relative Strength Index)
delta = data['Adj Close'].diff()
gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
rs = gain / loss
data['RSI'] = 100 - (100 / (1 + rs))

# Calculate Bollinger Bands
data['Upper Band'] = data['50-Day MA'] + (data['Adj Close'].rolling(window=50).
 ↪std() * 2)
data['Lower Band'] = data['50-Day MA'] - (data['Adj Close'].rolling(window=50).
 ↪std() * 2)

# Calculate MACD (Moving Average Convergence Divergence)
ema_12 = data['Adj Close'].ewm(span=12, adjust=False).mean()
ema_26 = data['Adj Close'].ewm(span=26, adjust=False).mean()
data['MACD'] = ema_12 - ema_26
data['Signal Line'] = data['MACD'].ewm(span=9, adjust=False).mean()

# Identify key findings
findings = {}

# 1. Moving Averages
findings['50-Day MA'] = data['50-Day MA'].iloc[-1]
findings['200-Day MA'] = data['200-Day MA'].iloc[-1]
if findings['50-Day MA'] > findings['200-Day MA']:
    findings['MA Trend'] = "Uptrend (50-Day MA is above 200-Day MA)"
else:
    findings['MA Trend'] = "Downtrend (50-Day MA is below 200-Day MA)"

# 2. RSI Analysis
findings['RSI'] = data['RSI'].iloc[-1]
if findings['RSI'] > 70:
    findings['RSI Trend'] = "Overbought (RSI > 70)"
elif findings['RSI'] < 30:
    findings['RSI Trend'] = "Oversold (RSI < 30)"
else:
    findings['RSI Trend'] = "Neutral (RSI between 30 and 70)"

# 3. MACD Analysis
findings['MACD'] = data['MACD'].iloc[-1]
```

```python
findings['Signal Line'] = data['Signal Line'].iloc[-1]
if findings['MACD'] > findings['Signal Line']:
    findings['MACD Trend'] = "Bullish (MACD > Signal Line)"
else:
    findings['MACD Trend'] = "Bearish (MACD < Signal Line)"

# 4. Bollinger Bands
findings['Upper Band'] = data['Upper Band'].iloc[-1]
findings['Lower Band'] = data['Lower Band'].iloc[-1]
if data['Adj Close'].iloc[-1] > findings['Upper Band']:
    findings['Bollinger Bands Trend'] = "Price is above Upper Band (Potential Overbought)"
elif data['Adj Close'].iloc[-1] < findings['Lower Band']:
    findings['Bollinger Bands Trend'] = "Price is below Lower Band (Potential Oversold)"
else:
    findings['Bollinger Bands Trend'] = "Price is within Bands (Normal)"

# 5. General Candlestick Analysis (50, 200, 365 days)
findings['Candlestick Insights'] = {
    '50-Day Period': data.tail(50).iloc[-1][['Open', 'Close', 'High', 'Low']].to_dict(),
    '200-Day Period': data.tail(200).iloc[-1][['Open', 'Close', 'High', 'Low']].to_dict(),
    '365-Day Period': data.tail(365).iloc[-1][['Open', 'Close', 'High', 'Low']].to_dict()
}

# Print Findings
print("Stock Analysis Report:")
for key, value in findings.items():
    print(f"\n{key}: {value}")
```

```
Stock Analysis Report:

50-Day MA: 56.87808845520019

200-Day MA: 58.3882851600647

MA Trend: Downtrend (50-Day MA is below 200-Day MA)

RSI: 52.75860697309015

RSI Trend: Neutral (RSI between 30 and 70)

MACD: 0.32211584437807517
```

```
Signal Line: 0.4185841630880368

MACD Trend: Bearish (MACD < Signal Line)

Upper Band: 59.90132278440249

Lower Band: 53.854854125997896

Bollinger Bands Trend: Price is within Bands (Normal)

Candlestick Insights: {'50-Day Period': {'Open': 58.7400016784668, 'Close':
58.93000030517578, 'High': 58.97999954223633, 'Low': 58.630001068115234},
'200-Day Period': {'Open': 58.7400016784668, 'Close': 58.93000030517578, 'High':
58.97999954223633, 'Low': 58.630001068115234}, '365-Day Period': {'Open':
58.7400016784668, 'Close': 58.93000030517578, 'High': 58.97999954223633, 'Low':
58.630001068115234}}
```

[ ]:

[ ]:

Other Analysis

Average Volume

[97]:
```
# Calculate average volume
average_volume = data['Volume'].rolling(window=50).mean().iloc[-1]

# Compare current volume with average
current_volume = data['Volume'].iloc[-1]
if current_volume > average_volume:
    volume_trend = "Higher volume than average"
else:
    volume_trend = "Lower volume than average"
```

[99]:
```
average_volume
```

[99]: 14901360.0

[101]:
```
volume_trend
```

[101]: 'Lower volume than average'

[ ]:

ATR (Volatility)

[105]:
```
# Calculate ATR (Average True Range)
data['High-Low'] = data['High'] - data['Low']
```

```
data['High-Close'] = abs(data['High'] - data['Adj Close'].shift())
data['Low-Close'] = abs(data['Low'] - data['Adj Close'].shift())
data['True Range'] = data[['High-Low', 'High-Close', 'Low-Close']].max(axis=1)
atr = data['True Range'].rolling(window=14).mean().iloc[-1]
```

[107]:
```
atr
```

[107]: 0.9503773280552456

[ ]:
```

```

Beta Calculation

[175]:
```
# Assuming you have the data for the S&P 500 or another market index
market_data = pd.read_csv('/Users/avinashhm/Desktop/Coca Cola.csv')
market_data['Date']= pd.to_datetime(market_data['Date'])
market_data.set_index('Date', inplace=True)

# Calculate daily returns for both Coca-Cola and the market
coca_cola_returns = data['Adj Close'].pct_change()
market_returns = market_data['Adj Close'].pct_change()

# Calculate Beta using covariance and variance
covariance = coca_cola_returns.cov(market_returns)
variance = market_returns.var()
beta = covariance / variance
```

```
/var/folders/t1/qt3r9zkn2rn4cyzr0txf6fc00000gn/T/ipykernel_9005/4102761424.py:3:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  market_data['Date']= pd.to_datetime(market_data['Date'])
```

[119]:
```
market_returns
```

[119]:
```
Date
2019-02-01         NaN
2019-03-01   -0.006179
2019-04-01    0.019940
2019-07-01   -0.013033
2019-08-01    0.011288
                 …
2023-12-22    0.005691
2023-12-26    0.004115
2023-12-27    0.002561
2023-12-28    0.000681
2023-12-29    0.003064
Name: Adj Close, Length: 1258, dtype: float64
```

[121]:
```
beta
```

[121]: 0.7115925725017358

[123]:
```
variance
```

[123]: 0.00018162925747906605

[125]:
```
coca_cola_returns
```

[125]:
```
Date
2019-01-02         NaN
2019-01-03   -0.006179
2019-01-04    0.019940
2019-01-07   -0.013033
2019-01-08    0.011288
                ...
2023-12-22    0.005691
2023-12-26    0.004115
2023-12-27    0.002561
2023-12-28    0.000681
2023-12-29    0.003064
Name: Adj Close, Length: 1258, dtype: float64
```

[127]:
```
market_data.set_index
```

[127]:
```
<bound method DataFrame.set_index of                  Open       High        Low
Close  Adj Close    Volume  \
Date
2019-02-01  46.939999  47.220001  46.560001  46.930000  39.828789  11603700
2019-03-01  46.820000  47.369999  46.529999  46.639999  39.582672  14714400
2019-04-01  46.750000  47.570000  46.639999  47.570000  40.371952  13013700
2019-07-01  47.570000  47.750000  46.900002  46.950001  39.845768  13135500
2019-08-01  47.250000  47.570000  47.040001  47.480000  40.295567  15420700
…                 …          …          …          …          …         …
2023-12-22  58.119999  58.459999  58.020000  58.320000  57.857216   9028500
2023-12-26  58.060001  58.709999  58.060001  58.560001  58.095314   6422500
2023-12-27  58.639999  58.770000  58.400002  58.709999  58.244122   8560100
2023-12-28  58.650002  58.869999  58.529999  58.750000  58.283806   8400100
2023-12-29  58.740002  58.980000  58.630001  58.930000  58.462376   9241600

            YEAR  Month  Day
Date
2019-02-01  2019      1    2
2019-03-01  2019      1    3
2019-04-01  2019      1    4
2019-07-01  2019      1    7
2019-08-01  2019      1    8
```

```
…           …     …   …
2023-12-22  2023    12   22
2023-12-26  2023    12   26
2023-12-27  2023    12   27
2023-12-28  2023    12   28
2023-12-29  2023    12   29

[1258 rows x 9 columns]>
```

[ ]:

[ ]:

Analysis

1. Volume Analysis

[133]:
```
# Calculate the 50-day moving average of volume
data['50-Day Volume MA'] = data['Volume'].rolling(window=50).mean()

# Compare current volume with average volume
current_volume = data['Volume'].iloc[-1]
average_volume = data['50-Day Volume MA'].iloc[-1]
if current_volume > average_volume:
    volume_trend = "Higher volume than average"
else:
    volume_trend = "Lower volume than average"

print(f"Volume Trend: {volume_trend}")
```

```
Volume Trend: Lower volume than average
```

2. Average True Range (ATR) for Volatility

[136]:
```
# Calculate True Range
data['High-Low'] = data['High'] - data['Low']
data['High-Close'] = abs(data['High'] - data['Adj Close'].shift())
data['Low-Close'] = abs(data['Low'] - data['Adj Close'].shift())
data['True Range'] = data[['High-Low', 'High-Close', 'Low-Close']].max(axis=1)

# Calculate 14-day ATR (average of True Range)
atr = data['True Range'].rolling(window=14).mean().iloc[-1]
print(f"ATR (Volatility): {atr:.2f}")
```

```
ATR (Volatility): 0.95
```

3. Fibonacci Retracement Levels

[139]:
```
# Define the high and low price over a period
high_price = data['High'].max()
```

```
low_price = data['Low'].min()

# Fibonacci levels
diff = high_price - low_price
level_23_6 = high_price - 0.236 * diff
level_38_2 = high_price - 0.382 * diff
level_50 = high_price - 0.5 * diff
level_61_8 = high_price - 0.618 * diff

print(f"Fibonacci Levels:")
print(f"23.6% Level: {level_23_6}")
print(f"38.2% Level: {level_38_2}")
print(f"50% Level: {level_50}")
print(f"61.8% Level: {level_61_8}")
```

```
Fibonacci Levels:
23.6% Level: 59.90051777648926
38.2% Level: 55.38473828887939
50% Level: 51.73499870300293
61.8% Level: 48.08525911712647
```

4. On-Balance Volume (OBV)

[142]:
```
# Calculate OBV
data['Daily Return'] = data['Adj Close'].pct_change()
data['Direction'] = np.where(data['Daily Return'] >= 0, 1, -1)
data['OBV'] = data['Direction'] * data['Volume']
data['OBV'] = data['OBV'].cumsum()

# Last OBV value
obv = data['OBV'].iloc[-1]
print(f"OBV (On-Balance Volume): {obv}")
```

```
OBV (On-Balance Volume): 999023100
```

5. Price Action and Candlestick Patterns

[145]:
```
# Bullish Engulfing Pattern (Current candle engulfs previous)
data['Bullish Engulfing'] = np.where(
    (data['Adj Close'].shift(1) < data['Open'].shift(1)) &
    (data['Adj Close'] > data['Open']) &
    (data['Open'] < data['Close'].shift(1)), 1, 0
)

# Count bullish engulfing patterns
bullish_engulfing_count = data['Bullish Engulfing'].sum()
print(f"Bullish Engulfing Patterns Count: {bullish_engulfing_count}")
```

```
Bullish Engulfing Patterns Count: 3
```

6. Earnings Per Share (EPS) and P/E Ratio

[148]:
```
# Let's assume we have EPS and stock price (P/E ratio) from a CSV or database
eps = 2.10  # Example EPS for Coca-Cola
price = data['Adj Close'].iloc[-1]
pe_ratio = price / eps

print(f"P/E Ratio: {pe_ratio:.2f}")
```

```
P/E Ratio: 27.84
```

7. Dividend Yield and Payout Ratio

[151]:
```
# Example Dividend Yield and Payout Ratio (You would get this from company data)
dividend_per_share = 1.68  # Example dividend per share
dividend_yield = (dividend_per_share / price) * 100

# Assuming a payout ratio from company earnings (Earnings = EPS * Shares␣
 ↪Outstanding)
payout_ratio = (dividend_per_share / eps) * 100

print(f"Dividend Yield: {dividend_yield:.2f}%")
print(f"Payout Ratio: {payout_ratio:.2f}%")
```

```
Dividend Yield: 2.87%
Payout Ratio: 80.00%
```

8. Beta (Stock Volatility Relative to Market)

[154]:
```
# Assuming you have market data for S&P 500 or another benchmark index
market_data = pd.read_csv('/Users/avinashhm/Desktop/Coca Cola.csv')
market_data['Date'] = pd.to_datetime(market_data['Date'])
market_data.set_index('Date', inplace=True)

# Calculate daily returns for Coca-Cola and the market
coca_cola_returns = data['Adj Close'].pct_change()
market_returns = market_data['Adj Close'].pct_change()

# Calculate Beta using covariance and variance
covariance = coca_cola_returns.cov(market_returns)
variance = market_returns.var()
beta = covariance / variance

print(f"Beta: {beta:.2f}")
```

```
Beta: 0.71
```

```
/var/folders/t1/qt3r9zkn2rn4cyzr0txf6fc00000gn/T/ipykernel_9005/2847744484.py:3:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
```

```
as-expected, please specify a format.
  market_data['Date'] = pd.to_datetime(market_data['Date'])
```

9. Institutional Ownership

[157]:
```python
# Example institutional ownership data (percentage of shares owned by␣
↪institutions)
institutional_ownership = 70  # Example: 70% of shares are held by␣
↪institutional investors

print(f"Institutional Ownership: {institutional_ownership}%")
```

```
Institutional Ownership: 70%
```

[ ]:
```python

```

Integrating These Findings into Your Report

[161]:
```python
findings = {}

# Add findings to the report
findings['Volume Trend'] = volume_trend
findings['ATR (Volatility)'] = atr
findings['Fibonacci Levels'] = {
    "23.6%": level_23_6,
    "38.2%": level_38_2,
    "50%": level_50,
    "61.8%": level_61_8
}
findings['OBV'] = obv
findings['Bullish Engulfing Patterns'] = bullish_engulfing_count
findings['P/E Ratio'] = pe_ratio
findings['Dividend Yield'] = dividend_yield
findings['Payout Ratio'] = payout_ratio
findings['Beta'] = beta
findings['Institutional Ownership'] = institutional_ownership

# Print the entire findings
for key, value in findings.items():
    print(f"{key}: {value}")
```

```
Volume Trend: Lower volume than average
ATR (Volatility): 0.9503773280552456
Fibonacci Levels: {'23.6%': 59.90051777648926, '38.2%': 55.38473828887939,
'50%': 51.73499870300293, '61.8%': 48.08525911712647}
OBV: 999023100
Bullish Engulfing Patterns: 3
P/E Ratio: 27.83922649565197
Dividend Yield: 2.873643059461249
Payout Ratio: 80.0
```

```
Beta: 0.7115925725017358
Institutional Ownership: 70
```