Sahith Rajesh and Krish Punyarthi

Macroeconomic Impact on Chipotle's Financial Performance

PROJECT OVERVIEW

How have specific macroeconomic factors affected Chipotle’s success in the past decade?

Purpose and Interest

Topic

- ➔ Interest in Chipotle (Personal Sentiment, Stock Trends)
- ➔ Macroeconomics

Datasets

- ➔ Global Macroeconomic Factors
- ➔ Chipotle Revenue/Income
- ➔ Chipotle Stock Prices

Findings

- ➔ Specific impacts on growth
- ➔ Correlations between factors

Data Collection

World Economic Outlook (CSV)

- ➔ International Monetary Fund (data.imf.org)
- ➔ Global Macroeconomic Factors

Chipotle Income Statement (HTML)

- ➔ Eulerpool (eulerpool.com)
- ➔ Revenue, Net Income

Chipotle Stock Price (API)

- ➔ AlphaVantage (alphavantage.co)
- ➔ Historical Stock Prices (Open, Close, High, Low, Volume)

Data Cleaning: CSV

```python
def data_parser():
    # Loading the CSV data to DataFrame
    df_downloaded = pd.read_csv("raw_world_economic_outlook.csv")

    # Countries that Chipotle operates in
    countries_to_keep = ["United States", "Canada", "United Kingdom", "France", "Germany"]
    df_countries = df_downloaded[df_downloaded["COUNTRY"].isin(countries_to_keep)].copy()

    # Filtering for relevant indicators
    indicators_list = ['Unemployment rate', 'Gross domestic product (GDP), Current prices, US dollar', 'All Items, Consumer price index (CPI), Period average',
                       'Gross capital formation, Percent of GDP', 'Net lending (+) / net borrowing (−), General government, Percent of GDP']
    df_countries_filtered = df_countries[df_countries['INDICATOR'].isin(indicators_list)].copy()

    col_names = {
    'Unemployment rate': 'Unemployment Rate (%)',
    'Gross domestic product (GDP), Current prices, US dollar': 'GDP (US dollars in billions)',
    'All Items, Consumer price index (CPI), Period average': 'Consumer price index (CPI)',
    'Gross capital formation, Percent of GDP': 'Gross capital formation (% of GDP)',
    'Net lending (+) / net borrowing (−), General government, Percent of GDP': 'Net lending/borrowing (% of GDP)'
    }
    df_countries_filtered['INDICATOR'] = df_countries_filtered['INDICATOR'].replace(col_names)

    # Keeping only relevant years and necessary columns
    years_to_keep = [str(year) for year in range(2015, 2026)]
    available_years = [year for year in years_to_keep if year in df_countries_filtered.columns]
    final_columns = ['COUNTRY', 'INDICATOR'] + available_years

    df_cleaned = df_countries_filtered[final_columns].copy()

    for year in available_years:
        df_cleaned[year] = df_cleaned[year].astype(float)

    df_cleaned[available_years] = df_cleaned[available_years].round(2)

    df_cleaned = df_cleaned.reset_index(drop=True)
```

```python
    # Resizing GDP values to be in billions
    df_cleaned.loc[df_cleaned['INDICATOR'] == 'GDP (US dollars in billions)', available_years] /= 1000000000


    # Reshaping the DataFrame to have years as a single column
    df_melted = df_cleaned.melt(id_vars=['COUNTRY', 'INDICATOR'], var_name='YEAR', value_name='VALUE')

    # Pivoting the DataFrame to have indicators as columns
    df_final = df_melted.pivot_table(index=['COUNTRY', 'YEAR'], columns='INDICATOR', values='VALUE').reset_index()

    # Writing cleaned DataFrame to CSV
    df_final.to_csv("cleaned_world_economic_outlook.csv", index=False)

    print(df_final)
```

```
INDICATOR COUNTRY  YEAR  Consumer price index (CPI)  \
0          Canada  2015                      126.57
1          Canada  2016                      128.38
2          Canada  2017                      130.43
3          Canada  2018                      133.38
4          Canada  2019                      135.98

INDICATOR  GDP (US dollars in billions)  Gross capital formation (% of GDP)  \
0                               1556.508                               23.82
1                               1527.996                               22.76
2                               1649.266                               23.55
3                               1725.300                               23.38
4                               1743.725                               23.04

INDICATOR  Net lending/borrowing (% of GDP)  Unemployment Rate (%)
0                                      -0.06                  7.00
1                                      -0.45                  7.05
2                                      -0.11                  6.42
3                                       0.36                  5.83
4                                      -0.02                  5.68
```

- ➔ Filtered countries Chipotle operates in
- ➔ Filtered specific macroeconomic factors
- ➔ Resized GDP values to be in billions

Data Cleaning: HTML

	Year	Revenue (Billions)	Revenue Growth (%)	Net Income (Billions)	\
0	2015	4.50	9.57	0.48	
1	2016	3.90	-13.26	0.02	
2	2017	4.48	14.65	0.18	
3	2018	4.86	8.67	0.18	
4	2019	5.59	14.84	0.35	
5	2020	5.98	7.12	0.36	
6	2021	7.55	26.12	0.65	
7	2022	8.63	14.40	0.90	
8	2023	9.87	14.33	1.23	
9	2024	11.31	14.61	1.53	
10	2025	12.08	6.76	1.62	

	Net Income Growth (%)
0	6.74
1	-95.37
2	700.00
3	0.00
4	98.86
5	1.43
6	83.66
7	37.88
8	36.60
9	24.92
10	5.48

- Sent a get request
- Parsed HTML tags
- Inconsistencies:
  - Removed non-digit characters in year
  - Replaced NaN value with 0.0

```python
def web_parser():
    # Retrieving HTML content
    url = "https://eulerpool.com/en/stock/Chipotle-Mexican-Grill-Stock-US1696561059"
    resp = requests.get(url)

    if resp.status_code != 200:
        print("Error: Unable to fetch data")
        return

    soup = BeautifulSoup(resp.text, "html.parser")

    # Parsing HTML content
    table = soup.find("div", {"class":"flex flex-nowrap md:mr-8 lg:mr-0 z-50"})
    years = table.find_all("span", {"class": "w-14 pr-5 whitespace-nowrap"})

    # Processing Header
    years_list = [year.text.strip() for year in years]
    years_list = years_list[15:26]

    # INCONSISTENCY: Remove non-digit characters from header ('e' in '2025e')
    years_list = [re.sub(r'\D', '', year) for year in years_list]
    years_list = [int(year) for year in years_list]

    # Processing Data Rows
    data_table = table.find_all("table")[1]
    data_rows = data_table.find("tbody").find_all("tr")

    revenue_row_cells = data_rows[0].find_all("td")
    all_revenue_values = [float(cell.text.strip()) for cell in revenue_row_cells]
    final_revenue_values = all_revenue_values[15:26]

    revenue_growth_row_cells = data_rows[1].find_all("td")
    all_revenue_growth_values = [float(cell.text.strip()) if cell.text.strip() not in ['-'] else 0.0 for cell in revenue_growth_row_cells]
    final_revenue_growth_values = all_revenue_growth_values[15:26]

    net_income_row_cells = data_rows[4].find_all("td")
    all_net_income_values = [float(cell.text.strip()) for cell in net_income_row_cells]
    final_net_income_values = all_net_income_values[15:26]
```

```python
    net_income_growth_row_cells = data_rows[5].find_all("td")

    # INCONSISTENCY: Replacing with 0.0 fixes NaN value for Net Income Growth, since there was no growth
    all_net_income_growth_values = [float(cell.text.strip()) if cell.text.strip() not in ['-'] else 0.0 for cell in net_income_growth_row_cells]
    final_net_income_growth_values = all_net_income_growth_values[15:26]

    # Creating DataFrame with years as rows and metrics as columns
    df = pd.DataFrame({
        'Year': years_list,
        'Revenue (Billions)': final_revenue_values,
        'Revenue Growth (%)': final_revenue_growth_values,
        'Net Income (Billions)': final_net_income_values,
        'Net Income Growth (%)': final_net_income_growth_values
    })


    # Writing cleaned DataFrame to CSV
    df.to_csv("cleaned_html.csv", index=False)

    print(df)
```

Data Cleaning: API

```python
def api_parser():
  # Retrieving data from Alpha Vantage API
  url = "https://www.alphavantage.co/query?function=TIME_SERIES_MONTHLY&symbol=CMG&apikey=YOUR_APLKEY"
  resp = requests.get(url)
  if resp.status_code != 200:
    print("Error: Unable to fetch data")
    return

  # Parsing JSON data
  data = resp.json()
  years = []
  open_prices = []
  high_prices = []
  low_prices = []
  close_prices = []
  for date, metrics in data["Monthly Time Series"].items():
    if date[:4] < "2015" or date[5:7] != "09": # Only take September data for each year from 2015 to 2025
      continue
    years.append(int(date[:4]))

    # INCONSISTENCY: Remove last two characters because cents only go to two decimal places
    open = metrics['1. open']
    open_prices.append(open[:-2])

    high = metrics['2. high']
    high_prices.append(high[:-2])

    low = metrics['3. low']
    low_prices.append(low[:-2])

    close = metrics['4. close']
    close_prices.append(close[:-2])

  # Creating DataFrame
  df = pd.DataFrame({'Year': years,'Open Price': open_prices,'High Price': high_prices,'Low Price': low_prices,'Close Price': close_prices})
  df.sort_values(by='Year',inplace=True, ignore_index=True)

  # Fixing Share Price to account for the 50-for-1 stock split in 2024
  df['Open Price'] = df['Open Price'].astype(float) * [1 if year < 2024 else 50 for year in df['Year']]
  df['High Price'] = df['High Price'].astype(float) * [1 if year < 2024 else 50 for year in df['Year']]
  df['Low Price'] = df['Low Price'].astype(float) * [1 if year < 2024 else 50 for year in df['Year']]
  df['Close Price'] = df['Close Price'].astype(float) * [1 if year < 2024 else 50 for year in df['Year']]


  # Writing cleaned DataFrame to CSV
  df.to_csv("cleaned_json_api.csv", index=False)

  print(df)
```

```python
### EXTRA DATA NEEDED FOR MACHINE LEARNING PURPOSES ###
months = []
open_values = []
high_values = []
low_values = []
close_values = []
volume_values = []
for date, metrics in data["Monthly Time Series"].items():
  if date[:4] < "2015":
    continue
  months.append(date[:-3])

  open = metrics['1. open']
  open_values.append(open[:-2])

  high = metrics['2. high']
  high_values.append(high[:-2])

  low = metrics['3. low']
  low_values.append(low[:-2])

  close = metrics['4. close']
  close_values.append(close[:-2])

  volume = metrics['5. volume']
  volume_values.append(int(volume))

df_monthly = pd.DataFrame({'Month': months,'Open Price': open_values,'High Price': high_values,'Low Price': low_values,'Close Price': close_values,'Volume': volume_values})
df_monthly.sort_values(by="Month",inplace=True,ignore_index=True)

df_monthly.to_csv("cleaned_json_api_monthly.csv", index=False)

print('\n\n')
print(df_monthly)

########### Function Call ###########
api_parser()
```

	Year	Open Price	High Price	Low Price	Close Price
0	2015	697.73	741.89	696.10	720.25
1	2016	415.81	441.20	395.11	423.50
2	2017	316.94	326.45	295.11	307.83
3	2018	476.19	497.87	453.67	454.52
4	2019	833.00	857.90	779.78	840.47
5	2020	1325.00	1384.46	1180.00	1243.71
6	2021	1908.00	1958.54	1816.31	1817.52
7	2022	1578.84	1754.56	1501.07	1502.76
8	2023	1928.16	1976.03	1795.01	1831.83
9	2024	2799.50	2950.00	2632.50	2881.00
10	2025	2090.50	2113.00	1915.00	1959.50

- ➔ Sent a get request
- ➔ Filtered for 2015-2025
- ➔ Inconsistency:
  - ◆ Removed extra digits from monetary values

Insight 1: GDP vs Revenue Correlation

- ➔ Identified strength of correlation between each country's GDP and Chipotle's revenue
  - ◆ Lowest: France
  - ◆ Highest: USA
- ➔ Shows how closely each country's economic health relates to Chipotle's performance
- ➔ Valuable to predict international markets
- ➔ 95% of Chipotle stores located in the USA

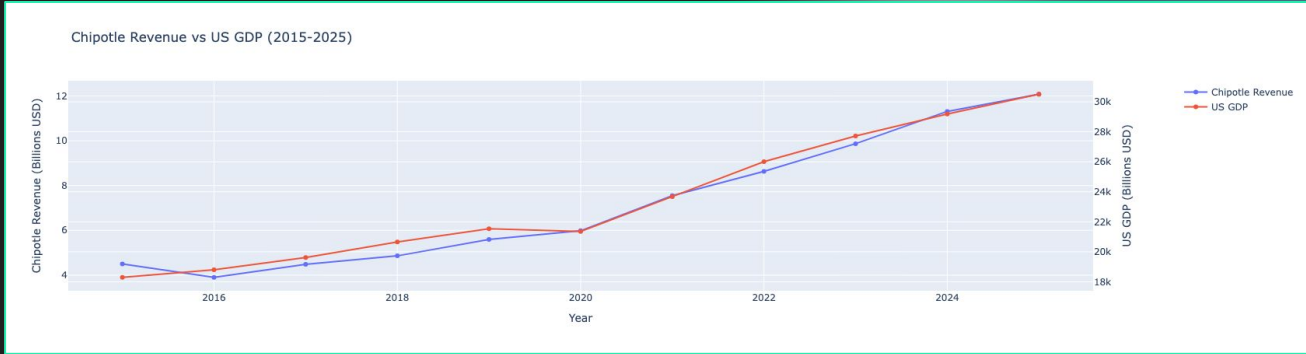
```python
def insight1():
    df_economic = pd.read_csv("cleaned_world_economic_outlook.csv")
    df_chipotle = pd.read_csv("cleaned_html.csv")

    countries = df_economic['COUNTRY'].unique()
    results = []

    for country in countries:
        country_data = df_economic[df_economic['COUNTRY'] == country].copy()

        # Merge with Chipotle data
        merged = pd.merge(df_chipotle,country_data,left_on='Year',right_on='YEAR',how='inner')

        # Calculate correlation
        correlation = merged['Revenue (Billions)'].corr(merged['GDP (US dollars in billions)'])

        results.append({'Country': country,'Correlation': correlation})

    # Create results dataframe and sort by correlation
    results_df = pd.DataFrame(results)
    results_df = results_df.sort_values('Correlation', ascending=False, ignore_index=True)

    print(results_df)
```

	Country	Correlation
0	United States	0.992707
1	United Kingdom	0.952713
2	Canada	0.949468
3	Germany	0.932536
4	France	0.931466

Visualization 1: Dual-Axis Line Graph

- ➔ Helps explain Insight 1
- ➔ Chipotle Revenue vs US GDP
- ➔ Shows correlation throughout the decade
- ➔ Very close correlation

```python
def visual1():
    df_economic = pd.read_csv("cleaned_world_economic_outlook.csv")
    df_chipotle = pd.read_csv("cleaned_html.csv")

    # Filter for US
    us_gdp = df_economic[df_economic['COUNTRY'] == 'United States'].copy()

    # Extract years and GDP values
    years = us_gdp['YEAR'].tolist()
    us_values = us_gdp['GDP (US dollars in billions)'].tolist()

    # Get Chipotle revenue data
    chipotle_years = df_chipotle['Year'].tolist()
    chipotle_revenue = df_chipotle['Revenue (Billions)'].tolist()


    # Create figure with secondary y-axis
    fig = make_subplots(specs=[[{"secondary_y": True}]])


    fig.add_trace(
        go.Scatter(x=chipotle_years,y=chipotle_revenue,name="Chipotle Revenue"),secondary_y=False)

    fig.add_trace(go.Scatter(x=years,y=us_values,name="US GDP"),secondary_y=True)

    # Figure title
    fig.update_layout(title_text="Chipotle Revenue vs US GDP (2015-2025)")

    # x-axis title
    fig.update_xaxes(title_text="Year")

    # y-axes titles
    fig.update_yaxes(title_text="Chipotle Revenue (Billions USD)", secondary_y=False)
    fig.update_yaxes(title_text="US GDP (Billions USD)", secondary_y=True)

    fig.show()
```



Insight 2: Nominal vs Real Income

- ➔ Shows growth without inflation
- ➔ Current income in 2015 USD
- ➔ “True” financial performance
- ➔ Purchasing power of earnings

```
def insight2():
    df_economic = pd.read_csv("cleaned_world_economic_outlook.csv")
    df_chipotle = pd.read_csv("cleaned_html.csv")

    # Since Chipotle is primarily US-based
    us_data = df_economic[df_economic['COUNTRY'] == 'United States'].copy()

    merged = pd.merge(df_chipotle,us_data,left_on='Year',right_on='YEAR',how='inner')
    # print(merged)

    base_cpi = merged[merged["Year"] == 2015]['Consumer price index (CPI)'].values[0]

    df = pd.DataFrame()
    df['Year'] = merged['Year']
    df['CPI'] = merged['Consumer price index (CPI)']
    df['Nominal Income (Billions)'] = merged['Net Income (Billions)']
    df['Real Income (Billions)'] = (merged['Net Income (Billions)'] * (base_cpi / merged['Consumer price index (CPI)'])).round(2)
    df['Difference (Billions)'] = df['Nominal Income (Billions)'] - df['Real Income (Billions)']
    df['Percent Difference (%)'] = (df['Difference (Billions)'] / df['Nominal Income (Billions)'] * 100).round(2)
    print(df)

    total_nominal = df['Nominal Income (Billions)'].sum()
    total_real = df['Real Income (Billions)'].sum()
    total_diff = total_nominal - total_real
    percent_diff = total_diff / total_nominal * 100

    print(f'\nTotal Nominal Income (Billions): {total_nominal}')
    print(f'Total Real Income (Billions): {total_real}')
    print(f'Total Difference (Billions): {total_diff.round(1)}')
    print(f'Total % Difference: {percent_diff.round(2)}%')


    total_inflation = df.iloc[-1]['CPI'] / df.iloc[0]['CPI'] * 100
    dollar_value = df.iloc[0]['CPI'] / df.iloc[-1]['CPI']

    print(f'\nTotal Inflation: {total_inflation.round(2)}%')
    print(f'$1 in 2025 was worth ${dollar_value.round(2)} in 2015')
```

	Year	CPI	Nominal Income (Billions)	Real Income (Billions)	\
0	2015	237.00	0.48	0.48	
1	2016	240.00	0.02	0.02	
2	2017	245.12	0.18	0.17	
3	2018	251.10	0.18	0.17	
4	2019	255.65	0.35	0.32	
5	2020	258.86	0.36	0.33	
6	2021	270.97	0.65	0.57	
7	2022	292.62	0.90	0.73	
8	2023	304.70	1.23	0.96	
9	2024	313.70	1.53	1.16	
10	2025	323.07	1.62	1.19	

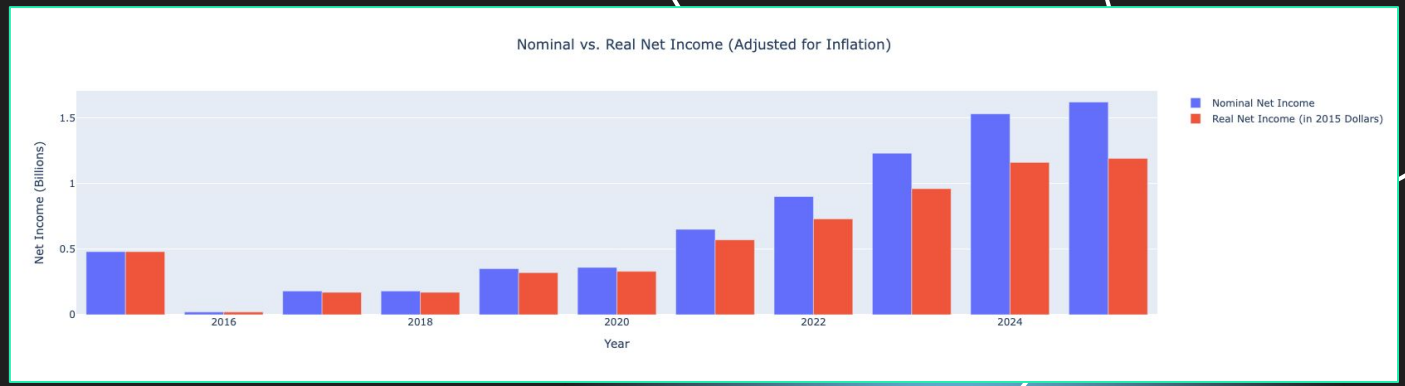
	Difference (Billions)	Percent Difference (%)
0	0.00	0.00
1	0.00	0.00
2	0.01	5.56
3	0.01	5.56
4	0.03	8.57
5	0.03	8.33
6	0.08	12.31
7	0.17	18.89
8	0.27	21.95
9	0.37	24.18
10	0.43	26.54

```
Total Nominal Income (Billions): 7.5
Total Real Income (Billions): 6.1
Total Difference (Billions): 1.4
Total % Difference: 18.67%

Total Inflation: 136.32%
$1 in 2025 was worth $0.73 in 2015
```

Visualization 2:

- ➔ Compares Nominal (reported) profits vs. Real (inflation-adjusted) profits.
- ➔ Supports Insight 2 on the Inflation Illusion.
- ➔ Shows the widening gap between reported profits and true value.
- ➔ Illustrates the company's true financial health and purchasing power.

```python
def visual3():

    df_economic = pd.read_csv("cleaned_world_economic_outlook.csv")
    df_chipotle = pd.read_csv("cleaned_html.csv")

    us_data = df_economic[df_economic['COUNTRY'] == 'United States'].copy()
    merged = pd.merge(df_chipotle,us_data,left_on='Year',right_on='YEAR',how='inner')

    base_cpi = merged[merged["Year"] == 2015]['Consumer price index (CPI)'].values[0]

    df = pd.DataFrame()
    df['Year'] = merged['Year']
    df['CPI'] = merged['Consumer price index (CPI)']
    df['Nominal Income (Billions)'] = merged['Net Income (Billions)']
    df['Real Income (Billions)'] = (merged['Net Income (Billions)'] * (base_cpi / merged['Consumer price index (CPI)'])).round(2)
    df['Difference (Billions)'] = df['Nominal Income (Billions)'] - df['Real Income (Billions)']
    df['Percent Difference (%)'] = (df['Difference (Billions)'] / df['Nominal Income (Billions)'] * 100).round(2)


    fig = go.Figure()

    fig.add_trace(go.Bar(
        x=df['Year'],
        y=df['Nominal Income (Billions)'],
        name='Nominal Net Income',
    ))

    fig.add_trace(go.Bar(
        x=df['Year'],
        y=df['Real Income (Billions)'],
        name='Real Net Income (in 2015 Dollars)',
    ))

    fig.update_layout(
        title='Nominal vs. Real Net Income (Adjusted for Inflation)',
        title_x=0.5,
        xaxis_title='Year',
        yaxis_title='Net Income (Billions)',
    )

    fig.show()
```



```python
def insight3():
    df_stock = pd.read_csv("cleaned_json_api.csv")
    df_html = pd.read_csv("cleaned_html.csv")

    df_merged = pd.merge(df_stock, df_html, on='Year', how='inner')

    df_merged['Stock Price Growth (%)'] = (df_merged['Close Price'].pct_change() * 100).round(2)
    results_table = df_merged[['Year', 'Stock Price Growth (%)', 'Revenue Growth (%)', 'Net Income Growth (%)']]

    print("Year-over-Year Growth Rate Comparison:")

    print(results_table.to_string(index = False))

    df_valid = results_table.dropna()

    max_stock = df_valid["Stock Price Growth (%)"].max()
    max_rev = df_valid["Revenue Growth (%)"].max()
    max_income = df_valid["Net Income Growth (%)"].max()

    best_stock_year = df_valid[df_valid["Stock Price Growth (%)"] == max_stock].iloc[0]
    best_rev_year = df_valid[df_valid["Revenue Growth (%)"] == max_rev].iloc[0]
    best_income_year = df_valid[df_valid["Net Income Growth (%)"] == max_income].iloc[0]

    print("\nKey Takeaways:")
    print(f"Highest stock price growth: {best_stock_year['Year']} ({max_stock}%)")
    print(f"Strongest revenue growth: {best_rev_year['Year']} ({max_rev}%)")
    print(f"Best net income growth: {best_income_year['Year']} ({max_income}%)")
```

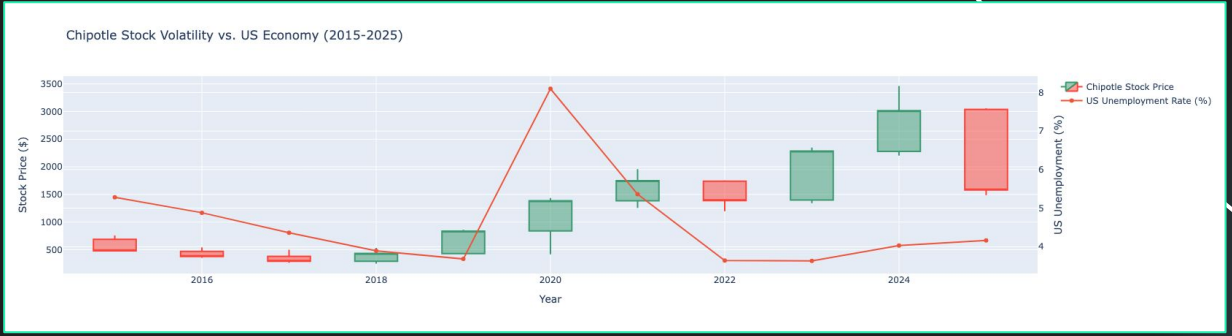
```
Year-over-Year Growth Rate Comparison:
 Year  Stock Price Growth (%)  Revenue Growth (%)  Net Income Growth (%)
 2015                     NaN                9.57                   6.74
 2016                  -41.20              -13.26                 -95.37
 2017                  -27.31               14.65                 700.00
 2018                   47.65                8.67                   0.00
 2019                   84.91               14.84                  98.86
 2020                   47.98                7.12                   1.43
 2021                   46.14               26.12                  83.66
 2022                  -17.32               14.40                  37.88
 2023                   21.90               14.33                  36.60
 2024                   57.27               14.61                  24.92
 2025                  -31.99                6.76                   5.48

Key Takeaways:
Highest stock price growth: 2019 (84.91%)
Strongest revenue growth: 2021 (26.12%)
Best net income growth: 2017 (700.0%)
```

Insight 3: Stock Price vs Revenue/Income

- ➔ Compares stock price growth ("hype") against profit growth ("fundamentals")
- ➔ Stock price is often disconnected from sales growth.
- ➔ In 2017, revenue grew, but stock fell 27.31%.
- ➔ In 2020, revenue was slow, but stock jumped 47.98%
- ➔ Investor sentiment drives the stock's valuation

Visualization 3: Candlestick Plot

- ➔ Shows stock price volatility vs. U.S. Unemployment Rate.
- ➔ Visually confirms the disconnect found in Insight 3.
- ➔ Stock price continued rising during high unemployment (2020).
- ➔ Investor perception drives stock price, not just the economy.




```python
def visual2():
    df_monthly = pd.read_csv("cleaned_json_api_monthly.csv", parse_dates=['Month'])

    july_2024 = pd.to_datetime('2024-07-01')
    post_split_mask = (df_monthly['Month'] >= july_2024)
    price_cols = ['Open Price', 'High Price', 'Low Price', 'Close Price']

    df_monthly.loc[post_split_mask, price_cols] = df_monthly.loc[post_split_mask, price_cols] * 50
    df_monthly.loc[df_monthly['Month'] == '2024-06-01', 'Low Price'] *= 50

    df_monthly_indexed = df_monthly.set_index('Month')

    agg_rules = {
        'Open Price': 'first',
        'High Price': 'max',
        'Low Price': 'min',
        'Close Price': 'last'
    }

    df_yearly = df_monthly_indexed.resample('Y').agg(agg_rules)
    df_yearly['Year'] = df_yearly.index.year

    df_world = pd.read_csv("cleaned_world_economic_outlook.csv")
    df_us_data = df_world[df_world['COUNTRY'] == 'United States']

    df_final = pd.merge(df_yearly, df_us_data, left_on='Year', right_on='YEAR', how='inner')
    df_final = df_final.sort_values(by='Year')

    fig = go.Figure()

    fig.add_trace(
        go.Candlestick(
            x=df_final['Year'],
            open=df_final['Open Price'],
            high=df_final['High Price'],
            low=df_final['Low Price'],
            close=df_final['Close Price'],
            name="Chipotle Stock Price"
        )
    )
```

```python
    fig.add_trace(
        go.Scatter(
            x=df_final['Year'],
            y=df_final['Unemployment Rate (%)'],
            name="US Unemployment Rate (%)",
            yaxis="y2"
        )
    )

    fig.update_layout(
        title_text="Chipotle Stock Volatility vs. US Economy (2015-2025)",
        xaxis_title="Year",
        xaxis_rangeslider_visible=False,
        yaxis=dict(title="Stock Price ($)", side="left"),
        yaxis2=dict(title="US Unemployment (%)", overlaying="y", side="right")
    )

    fig.show()
```

Insight 4: Linear Regression

- ➔ Performs economic sensitivity analysis on revenue growth
- ➔ Used supervised Linear Regression with three macro factors
- ➔ Model explains 42.1% of Revenue Growth
- ➔ GDP growth has the strongest impact (Coefficient: 2.61)
- ➔ Unemployment has a surprising positive impact (Coefficient: 1.98)

```python
def insight4():
    df_economic = pd.read_csv("cleaned_world_economic_outlook.csv")
    df_chipotle = pd.read_csv("cleaned_html.csv")

    us_data = df_economic[df_economic['COUNTRY'] == 'United States']

    df_merged = pd.merge(df_chipotle, us_data, left_on='Year', right_on='YEAR', how='inner')

    # Inflation Rate (annual % change in CPI)
    df_merged['Inflation Rate (%)'] = df_merged['Consumer price index (CPI)'].pct_change() * 100

    # GDP Growth Rate
    df_merged['GDP growth (percent)'] = df_merged['GDP (US dollars in billions)'].pct_change() * 100

    df_model_data = df_merged[[
        'Year',
        'Revenue Growth (%)',
        'GDP growth (percent)',
        'Unemployment Rate (%)',
        'Inflation Rate (%)'
    ]].copy()

    df_model_data = df_model_data.iloc[1:].reset_index(drop=True)

    y = df_model_data['Revenue Growth (%)']
    X = df_model_data[['GDP growth (percent)', 'Unemployment Rate (%)', 'Inflation Rate (%)']]

    model = LinearRegression()
    model.fit(X, y)

    r2 = model.score(X, y)
    coefficients = pd.DataFrame(model.coef_,X.columns,columns=['Coefficient'])

    print(f"\nPrediction Accuracy (R-squared): {round(r2, 3)}")
    print(f"{round(r2*100, 2)}% of the variance in Chipotle's Revenue Growth is explained by these 3 macro factors.")
    print("\n")
    print(coefficients)
```

```
Prediction Accuracy (R-squared): 0.421
42.1% of the variance in Chipotle's Revenue Growth is explained by these 3 macro factors.


                        Coefficient
GDP growth (percent)       2.612691
Unemployment Rate (%)      1.982135
Inflation Rate (%)        -0.667328
```

```python
def insight5():
    df = pd.read_csv("cleaned_json_api_monthly.csv")
    features = df[['Close Price', 'Volume']]

    # Normalize the data (since price and volume are on diff. scales)
    scaler = StandardScaler()
    features = scaler.fit_transform(features)

    # Applying K-means
    kmeans = KMeans(n_clusters = 4, random_state=0, n_init=10)
    df['Cluster'] = kmeans.fit_predict(features)

    stats = df.groupby('Cluster').agg({'Close Price': ['mean', 'min', 'max'], 'Volume': ['mean', 'min', 'max'], 'Month': 'count'}).round(2)
    stats.columns = ['Avg Price', 'Min Price', 'Max Price', 'Avg Volume', 'Min Volume', 'Max Volume', 'Count']
    # stats.sort_values(by='Avg Price', inplace=True)
    print(stats)

    cluster_labels = {
    0: "Steady Growth Period",
    1: "Pre-Split Period",
    2: "Post-Split Period",
    3: "High Growth Period"
    }

    print("\n\nClusters:")
    for num, label in cluster_labels.items():
        print(f"{num}: {label}")


    print("\n\nCluster Timeline Distribution:")
    for cluster_id in [0,1,2,3]:
        cluster_months = df[df['Cluster'] == cluster_id]['Month'].tolist()
        earliest = cluster_months[0]
        latest = cluster_months[-1]
        print(f"Cluster {cluster_id}: {earliest} to {latest}")
```

Cluster	Avg Price	Min Price	Max Price	Avg Volume	Min Volume	Max Volume	Count
0	535.95	62.65	1003.91	1.963568e+07	6555620	84327316	66
1	2506.57	2067.62	3159.60	5.143283e+06	3371680	6811628	10
2	50.14	31.63	61.52	3.157414e+08	165561762	493788343	17
3	1541.92	1052.36	1962.28	6.008137e+06	3767928	9080785	38

```
Clusters:
0: Steady Growth Period
1: Pre-Split Period
2: Post-Split Period
3: High Growth Period


Cluster Timeline Distribution:
Cluster 0: 2015-01 to 2024-06
Cluster 1: 2023-04 to 2024-05
Cluster 2: 2024-07 to 2025-11
Cluster 3: 2020-06 to 2023-10
```

Insight 5: K-Means Grouping

- ➔ Unsupervised learning model
- ➔ 4 groups (k = 4)
- ➔ Explains trends over time
  - ◆ 50-for-1 stock split
- ➔ Identifies specific market phases and shifts in trading behavior

Conclusion/Findings

Measure Chipotle's Reliance on the US Economy

Strongest Macro Indicator

A Linear Regression model R^2 = 42% confirmed that GDP growth has the strongest positive impact on revenue +$2.61 per $1 increase in GDP growth.

Brand Effect:

The model also showed a counterintuitive positive relationship with unemployment (+$1.98), suggesting Chipotle benefits as consumers trade down during economic slowdowns.

Calculate Chipotle's True Profits After Inflation

Real Profits:

Analysis of Nominal vs. Real Net Income using 2015 as the base year proved that a large portion of reported profit growth is an inflation illusion. The widening gap between the two values shows the true purchasing power of the company's profits grew much slower than reported.

Explain What Drives the Chipotle Stock Price

Valuation Driven by Sentiment:
The stock is often disconnected from economic fundamentals, as visually proven by the price surging in 2020 despite high unemployment, confirming it's heavily influenced by market expectations and corporate narrative.

ML Liquidity Analysis:
K-Means clustering found the 50-for-1 stock split created a new market phase. Trading volume for this post-split phase surged to over 312 million shares, confirming the split successfully increased market liquidity.

Conclusion Cont.

Machine Learning Skills & Data Science

K-Means Clustering :

- Used to detect distinct trading phases in CMG's stock and clearly identify the new high-liquidity period created by the stock split.

Linear Regression:

- Used to measure how strongly GDP, unemployment, and inflation influence Chipotle's revenue growth.

Advanced Plotly Visualizations:

- Used plotly.graph_objects to build Candlestick charts, multi-axis visuals, and more customizable graphics.

Pandas Data Manipulation:

- .corr() for correlation between economic variables and revenue.
- .merge() to combine IMF, Eulerpool, and Alpha Vantage data by Year.

Finance & Economics Concepts

- **Real vs. Nominal Profitability:** Learned how inflation distorts reported net income and how to evaluate true purchasing-power growth.
- **Income Statement Interpretation:** Analyzed revenue and net income to assess Chipotle's fundamentals.
- **Market Behavior & Drivers:** Found that CMG's stock is often driven by investor sentiment and trading regimes, while revenue is more tied to macroeconomic fundamentals like US GDP.

Cited Sources

If you used any additional sources to complete your Data Analysis section, list them here:

- Chipotle 50-to-1 stock split (Web Collection Requirement 2): https://ir.chipotle.com/2024-03-19-CHIPOTLE-BOARD-OF-DIRECTORS-APPROVES-50-FOR-1-STOCK-SPLIT
- Pandas DataFrame .corr() function (Insight 1): https://www.w3schools.com/python/pandas/pandas_correlations.asp
- K-Means scikit-learn Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
- Linear Regression scikit-learn Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- Plotly Graph Objects Documentation (Visualizations 1, 2, 3): https://plotly.com/python-api-reference/generated/plotly.graph_objects.Figure.html
- Plotly Multiple Axes Documentation (Visualization 1): https://plotly.com/python/multiple-axes/
- Plotly Candlestick Documentation (Visualization 2): https://plotly.com/python/candlestick-charts/

Thank you!