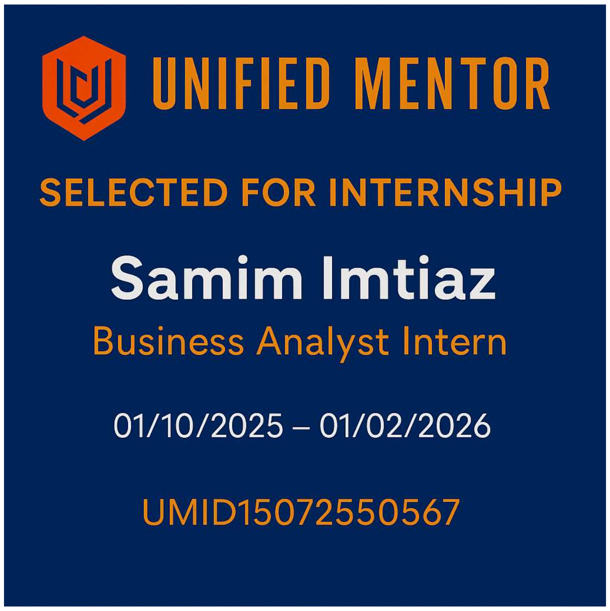
UNIFIED MENTOR
YOUR SKILL, SUCCESS & JOURNEY


UNIFIED MENTOR

SELECTED FOR INTERNSHIP

Samim Imtiaz

Business Analyst Intern

01/10/2025 – 01/02/2026

UMID15072550567

📘 Objective

Top 10 🌐 Global Stocks Analysis 📊

📕 Import Necessary Library 🗂️

In [3]:
```
import yfinance as yf
import pandas as pd

# list of 10 global tickers (mix of US + Indian)
tickers = ['AAPL', 'MSFT', 'GOOGL', 'AMZN', 'TSLA',
           'META', 'NVDA', 'RELIANCE.NS', 'HDFCBANK.NS', 'TCS.NS']

# empty list to collect data
dfs = []

for t in tickers:
    data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False
    data['Stock'] = t
    dfs.append(data)

# combine all
df = pd.concat(dfs)
df.reset_index(inplace=True)
df.to_csv("global_stocks.csv", index=False)
print("Saved ✅ global_stocks.csv →", len(df), "rows")
```

```
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)
<ipython-input-3-494b2b4ff797>:12: FutureWarning: YF.download() has changed a
rgument auto_adjust default to True
  data = yf.download(t, start="2020-01-01", end="2025-11-01", progress=False)

Saved ✅ global_stocks.csv → 14607 rows
```

🖥️ Diplay Top 5 Rows

In [24]:
```
df.head()
```

Out[24]:

Price	Date	Close	High	Low	Open	Volume	Stock	Close	High	Lo
Ticker		AAPL	AAPL	AAPL	AAPL	AAPL		MSFT	MSFT	MS
0	2020-01-02	72.538528	72.598907	71.292319	71.545905	135480400.0	AAPL	NaN	NaN	N
1	2020-01-03	71.833305	72.594071	71.608700	71.765682	146322800.0	AAPL	NaN	NaN	N
2	2020-01-06	72.405670	72.444313	70.703005	70.954181	118387200.0	AAPL	NaN	NaN	N
3	2020-01-07	72.065155	72.671348	71.845377	72.415345	108872000.0	AAPL	NaN	NaN	N
4	2020-01-08	73.224403	73.526295	71.768079	71.768079	132079200.0	AAPL	NaN	NaN	N

5 rows × 52 columns

🔧 Resetting The Index to Stock Lavel

In [25]:
```
import yfinance as yf
import pandas as pd

# 10 Global Stocks (mix of US + India)
tickers = ['AAPL', 'MSFT', 'GOOGL', 'AMZN', 'TSLA',
           'META', 'NVDA', 'RELIANCE.NS', 'HDFCBANK.NS', 'TCS.NS']

# Download all at once
data = yf.download(tickers, start="2020-01-01", end="2025-11-01", auto_adjust=F

# Fix multi-level columns
data.columns.names = ['Attribute', 'Stock']

# Convert wide → long
df = data.stack(level='Stock').reset_index()

# Remove 'Adj Close' safely if missing
cols_to_keep = [c for c in ['Date', 'Stock', 'Open', 'High', 'Low', 'Close', 'V
df = df[cols_to_keep]

# Ensure proper datetime and sorting
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df = df.sort_values(['Stock', 'Date']).reset_index(drop=True)

# Save clean file
df.to_csv("Global_Stocks_Clean.csv", index=False)
print("✅ Saved Global_Stocks_Clean.csv —", len(df), "rows")
print(df.head())
```

```
<ipython-input-25-95b906306376>:15: FutureWarning: The previous implementatio
n of stack is deprecated and will be removed in a future version of pandas. S
ee the What's New notes for pandas 2.1.0 for details. Specify future_stack=Tr
ue to adopt the new implementation and silence this warning.
  df = data.stack(level='Stock').reset_index()

✅ Saved Global_Stocks_Clean.csv — 14607 rows
Attribute       Date Stock       Open       High        Low      Close  \
0         2020-01-02  AAPL  74.059998  75.150002  73.797501  75.087502
1         2020-01-03  AAPL  74.287498  75.144997  74.125000  74.357498
2         2020-01-06  AAPL  73.447502  74.989998  73.187500  74.949997
3         2020-01-07  AAPL  74.959999  75.224998  74.370003  74.597504
4         2020-01-08  AAPL  74.290001  76.110001  74.290001  75.797501

Attribute       Volume
0          135480400.0
1          146322800.0
2          118387200.0
3          108872000.0
4          132079200.0
```

In [26]:
```
df.head()
```

Out[26]:

Attribute	Date	Stock	Open	High	Low	Close	Volume
0	2020-01-02	AAPL	74.059998	75.150002	73.797501	75.087502	135480400.0
1	2020-01-03	AAPL	74.287498	75.144997	74.125000	74.357498	146322800.0
2	2020-01-06	AAPL	73.447502	74.989998	73.187500	74.949997	118387200.0
3	2020-01-07	AAPL	74.959999	75.224998	74.370003	74.597504	108872000.0
4	2020-01-08	AAPL	74.290001	76.110001	74.290001	75.797501	132079200.0

Proper types & order

In [8]:
```

df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df = df.sort_values(['Stock','Date']).reset_index(drop=True)


g = df.groupby('Stock', group_keys=False)
```

Core OHLC enrichments

In [9]:
```
df['Prev Close'] = g['Close'].shift(1)
df['Last'] = df['Close']                                    # daily last price ≈
df['VWAP'] = (df['High'] + df['Low'] + df['Close']) / 3     # proxy (no intraday
df['Turnover'] = df['Close'] * df['Volume']                 # value traded (curre
df['Deliverable Volume'] = df['Volume']                     # proxy (Yahoo lacks
```

Daily metrics

In [10]:
```

df['Daily_Return_%'] = (df['Close'] / df['Prev Close'] - 1) * 100
df['Volatility'] = df['High'] - df['Low']
df['7D_MA_Close']  = g['Close'].transform(lambda s: s.rolling(7,  min_periods=7
df['30D_MA_Close'] = g['Close'].transform(lambda s: s.rolling(30, min_periods=3
```

Rolling-period returns (trading-day approximations)

In [12]:
```
def pct_ret(s, periods):
    return (s / s.shift(periods) - 1) * 100

df['1DAY'] = g['Close'].apply(lambda s: pct_ret(s, 1))
df['1WK']  = g['Close'].apply(lambda s: pct_ret(s, 5))
df['1M']   = g['Close'].apply(lambda s: pct_ret(s, 21))
df['1Y']   = g['Close'].apply(lambda s: pct_ret(s, 252))
df['5Y']   = g['Close'].apply(lambda s: pct_ret(s, 1260))
df['52WK'] = df['1Y']  # synonym; keep both if you want both labels
```

YTD return (vs first close of the year for each stock)

In [13]:
```
df['Year'] = df['Date'].dt.year
first_close_y = df.groupby(['Stock','Year'])['Close'].transform('first')
df['YTD'] = (df['Close'] / first_close_y - 1) * 100
```

Round presentation columns

In [14]:
```
round_cols = ['Daily_Return_%','1DAY','1WK','1M','1Y','5Y','52WK','YTD','VWAP',
df[round_cols] = df[round_cols].round(4)
```

Reorder columns

In [15]:
```
cols = ['Date','Stock','Open','High','Low','Close','Prev Close','Last','VWAP',
        'Volume','Turnover','Deliverable Volume','Volatility',
        'Daily_Return_%','7D_MA_Close','30D_MA_Close',
        '1DAY','1WK','1M','YTD','52WK','1Y','5Y']
df = df[cols]
```

Save for Power BI

In [16]:
```
out_path = r"C:\Users\SAMIM IMTIAZ\Desktop\Global_Stocks_Enriched.csv"
df.to_csv(out_path, index=False)
print("✅ Saved:", out_path, "| Rows:", len(df), "| Stocks:", df['Stock'].nuni
```

```
✅ Saved: C:\Users\SAMIM IMTIAZ\Desktop\Global_Stocks_Enriched.csv | Rows: 14
607 | Stocks: 10
```

First 5 rows

In [17]:
```
df.head()
```

Out[17]:

Attribute	Date	Stock	Open	High	Low	Close	Prev Close	Last	VWAP
0	2020-01-02	AAPL	74.059998	75.150002	73.797501	75.087502	NaN	75.087502	74.678
1	2020-01-03	AAPL	74.287498	75.144997	74.125000	74.357498	75.087502	74.357498	74.542
2	2020-01-06	AAPL	73.447502	74.989998	73.187500	74.949997	74.357498	74.949997	74.375
3	2020-01-07	AAPL	74.959999	75.224998	74.370003	74.597504	74.949997	74.597504	74.730
4	2020-01-08	AAPL	74.290001	76.110001	74.290001	75.797501	74.597504	75.797501	75.399

5 rows × 23 columns

📕 Import Necessary Library 🗂️ for Debt Equity Reco

In [2]:
```
import yfinance as yf
import pandas as pd
```

10 tickers (mix US + India)

In [3]:

```
tickers = ['AAPL','AMZN','GOOGL','META','MSFT','NVDA','TSLA',
           'RELIANCE.NS','HDFCBANK.NS','TCS.NS']

rows_fin = []
rows_reco = []

for t in tickers:
    tk = yf.Ticker(t)

    # Annual financials (Income Statement) ↓
    fin_a = tk.financials    # rows index like 'Total Revenue','Net Income'
    # Quarterly financials ↓
    fin_q = tk.quarterly_financials

    # Balance sheet (annual) for Debt/Equity ↓
    bs_a = tk.balance_sheet   # rows: 'Total Liab','Total Stockholder Equity'
    # Quarterly balance sheet ↓
    bs_q = tk.quarterly_balance_sheet

    # Shares outstanding (for EPS if needed)
    info = tk.info or {}
    shares = info.get('sharesOutstanding', None)

    # Helper: tidy function
    def extract(fin_df, bs_df, period):
        if fin_df is None or fin_df.empty:
            return
        for dt, col in fin_df.items():    # dt is pd.Timestamp
            year = dt.year
            qtr  = (dt.month-1)//3 + 1
            revenue   = fin_df.at['Total Revenue', dt]   if 'Total Revenue' in
            netincome = fin_df.at['Net Income', dt]      if 'Net Income' in fir

            # EPS: if fin has 'Diluted EPS' use it; else compute from shares if
            if 'Diluted EPS' in fin_df.index:
                eps = fin_df.at['Diluted EPS', dt]
            elif shares and shares > 0 and netincome is not None:
                eps = netincome / shares
            else:
                eps = None

            totaldebt = None
            equity    = None
            if bs_df is not None and not bs_df.empty and dt in bs_df.columns:
                totaldebt = bs_df.at['Total Liab', dt] if 'Total Liab' in bs_df
                equity    = bs_df.at['Total Stockholder Equity', dt] if 'Total

            rows_fin.append({
                'Stock': t, 'Period': period,
                'Year': year, 'Quarter': f'Q{qtr}',
                'Revenue': revenue, 'NetProfit': netincome, 'EPS': eps,
                'TotalDebt': totaldebt, 'Equity': equity
            })

    # Build annual & quarterly rows
    extract(fin_a, bs_a, 'Annual')
    extract(fin_q, bs_q, 'Quarterly')

    # Analyst “recommendation mean/key” (Yahoo)
    # Mean: 1=Strong Buy … 5=Sell
    rec_mean = info.get('recommendationMean', None)
    rec_key  = info.get('recommendationKey', None)
    rows_reco.append({'Stock': t, 'Date': pd.Timestamp.today().normalize(),
                      'RecommendationMean': rec_mean, 'RecommendationKey': rec_
```

Create DataFrames

In [4]:
```
df_fin  = pd.DataFrame(rows_fin).sort_values(['Stock','Period','Year','Quarter'
df_reco = pd.DataFrame(rows_reco)
```

Saveing CSVs to local Desktop

In [5]:
```
df_fin.to_csv('financials.csv', index=False)
df_reco.to_csv('analyst_reco.csv', index=False)

print('Saved financials.csv:', len(df_fin), 'rows')
print('Saved analyst_reco.csv:', len(df_reco), 'rows')
```

```
Saved financials.csv: 100 rows
Saved analyst_reco.csv: 10 rows
```

Finance Save for Power BI

In [8]:
```
out_path = r"C:\Users\SAMIM IMTIAZ\Desktop\Global_Financial.csv"
df_fin.to_csv(out_path, index=False)
print("✅ Saved:", out_path, "| Rows:", len(df_fin), "| Stocks:", df_fin['Stoc
```

```
✅ Saved: C:\Users\SAMIM IMTIAZ\Desktop\Global_Financial.csv | Rows: 100 | St
ocks: 10
```

Reco Dataset Save for Power BI

In [6]:
```
# Save for Power BI
out_path = r"C:\Users\SAMIM IMTIAZ\Desktop\Global_Reco.csv"
df_reco.to_csv(out_path, index=False)
print("✅ Saved:", out_path, "| Rows:", len(df_reco), "| Stocks:", df_reco['St
```

```
✅ Saved: C:\Users\SAMIM IMTIAZ\Desktop\Global_Reco.csv | Rows: 10 | Stocks:
10
```

first 5 Rows

In [7]:
```
df_fin.head()
```

Out[7]:

	Stock	Period	Year	Quarter	Revenue	NetProfit	EPS	TotalDebt	Equity
3	AAPL	Annual	2021	Q3	3.658170e+11	9.468000e+10	5.61	None	None
2	AAPL	Annual	2022	Q3	3.943280e+11	9.980300e+10	6.11	None	None
1	AAPL	Annual	2023	Q3	3.832850e+11	9.699500e+10	6.13	None	None
0	AAPL	Annual	2024	Q3	3.910350e+11	9.373600e+10	6.08	None	None
8	AAPL	Quarterly	2024	Q2	8.577700e+10	2.144800e+10	1.40	None	None

first 5 Rows

In [8]:
```
df_reco.head()
```

Out[8]:

	Stock	Date	RecommendationMean	RecommendationKey
0	AAPL	2025-11-04	2.00000	buy
1	AMZN	2025-11-04	1.29851	strong_buy
2	GOOGL	2025-11-04	1.48485	strong_buy
3	META	2025-11-04	1.44118	strong_buy
4	MSFT	2025-11-04	1.21053	strong_buy