Financial Data Analysis Project

(using yfinance, matplotlib, technical indicators)

# Problem statement

Many analysts need to combine a company's published financial statements into a single, tidy table for easy inspection and downstream analysis while keeping visual work focused on price action with technical overlays.

This project should produce a repeatable pipeline to:  
- collect fundamentals (income, balance sheet, cash flow, earnings)  
- concatenate statements into one normalized table  
- save the combined table  
- plot price action with technical indicators.

# 2. Project summary & objectives

**Title:** *Financial Analysis and Visualization of Indian Equities using yfinance*

**Primary objectives**

* Extract historical price data using yfinance for analysis.
* A runnable Jupyter notebook (.ipynb) implementing the pipeline.
* A consolidated CSV table with stacked financial statements (annual + quarterly).
* Compute common technical indicators (SMA, EMA, RSI, MACD, Bollinger Bands, Volume MA).
* Plot price and indicator overlays with matplotlib (publication-quality charts).
* A short analysis interpretation of fundamentals vs price.

# 3. Tools & dependencies

Python (3.8+)  
Libraries: yfinance, pandas, numpy, matplotlib  
  
Install:  
pip install yfinance pandas numpy matplotlib ta

# 4. Dataset

- Tickers: any ticker supported by yfinance (e.g., AAPL, MSFT, RELIANCE.NS)  
- Periods: choose period='5y' (or start/end for fixed ranges)  
- Fields: financial statements returned by yfinance.Ticker; price fields Open/High/Low/Close/Adj Close/Volume  
Notes: yfinance uses different column names across tickers/markets — inspect raw tables and adapt if needed.

# 5. High-level process flow

1. Environment setup (install libs + notebook)  
2. Download price time series for chosen ticker  
3. Fetch fundamentals using yfinance.Ticker\yfinance.download  
4. Normalize each statement (transpose periods → rows)  
5. Concatenate all statements into a single table  
6. Save the consolidated table (CSV)  
7. Compute technical indicators on price (SMA/EMA/RSI/MACD)  
8. Plot price + indicators using matplotlib  
9. Interpret results & export deliverables

# 6. Insights of Analysis

- Are revenues and net income trending up or down?  
- Does cash from operations support net income?  
- Any large swings in liabilities or negative equity?  
- Does price action reflect fundamentals?  
- Use RSI and MACD panels to note momentum vs long-term trend.

# Examples

Here are some example tickers:

* **Indian / NSE**
  + Reliance Industries → RELIANCE.NS
  + Tata Motors → TATAMOTORS.NS
  + Infosys → INFY.NS
* **Indian / BSE**
  + Same company may have a BSE variant: e.g. RELIANCE.BO for the BSE listing of Reliance.
* **Global / US**
  + Apple → AAPL
  + Microsoft → MSFT
  + Tesla → TSLA
* **Indexes**
  + Nifty 50 → ^NSEI

# Step-by-step guide

**Step 0 — Imports**

import yfinance as yf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.dates import DateFormatter

plt.rcParams['figure.figsize'] = (14,8)

**Step 1 — Price downloader (function)**

def download\_price(‘ticker’, period='5y', interval='1d', start=’YYYY-MM-DD’, end=’YYYY-MM-DD’):

df = yf.download(ticker, period=period, start=start, end=end, interval=interval, progress=False)

if df is None or df.empty:

raise ValueError(f"No price data returned for {ticker}")

df.index = pd.to\_datetime(df.index)

return df.sort\_index()

**Step 2 — Fetch fundamentals (function)**

df=yf.Ticker(‘GOOG’)

info=df.info

info

df.get\_financials()

pnl=df.financials

bs=df.balance\_sheet

cf=df.cash\_flow

fs=pd.concat([pnl,bs,cf])

fs.T

**Step 3 — Save concatenated table**

def save\_fundamentals\_table(df, filename='fundamentals\_concatenated.csv'):

if df is None or df.empty:

print("No fundamentals to save.")

return

df.to\_csv(filename, index=False)

print(f"Saved fundamentals table to: {filename}")

**Step 4 — Technical indicators**

def SMA(series, window):

return series.rolling(window=window, min\_periods=1).mean()

def EMA(series, window):

return series.ewm(span=window, adjust=False).mean()

def RSI(series, window=14):

delta = series.diff()

up = delta.clip(lower=0)

down = -1 \* delta.clip(upper=0)

ma\_up = up.ewm(alpha=1/window, adjust=False).mean()

ma\_down = down.ewm(alpha=1/window, adjust=False).mean()

rs = ma\_up / ma\_down

return 100 - (100 / (1 + rs))

def MACD(series, fast=12, slow=26, signal=9):

ema\_fast = EMA(series, fast)

ema\_slow = EMA(series, slow)

macd\_line = ema\_fast - ema\_slow

signal\_line = EMA(macd\_line, signal)

hist = macd\_line - signal\_line

return macd\_line, signal\_line, hist

**Step 5 — Plot price + indicators**

def plot\_price\_with\_indicators(price\_df, title=None):

df = price\_df.copy()

if 'Adj Close' in df.columns:

price = df['Adj Close']

else:

price = df['Close']

df['SMA50'] = SMA(price, 50)

df['SMA200'] = SMA(price, 200)

df['EMA20'] = EMA(price, 20)

df['RSI14'] = RSI(price, 14)

df['MACD'], df['MACD\_signal'], df['MACD\_hist'] = MACD(price)

fig, axs = plt.subplots(4,1,sharex=True, gridspec\_kw={'height\_ratios':[3,1,1,1]}, figsize=(14,10))

ax\_price, ax\_vol, ax\_macd, ax\_rsi = axs

ax\_price.plot(df.index, price, label='Adj Close')

ax\_price.plot(df.index, df['SMA50'], label='SMA50')

ax\_price.plot(df.index, df['SMA200'], label='SMA200')

ax\_price.plot(df.index, df['EMA20'], label='EMA20')

ax\_price.legend(loc='upper left')

if 'Volume' in df.columns:

ax\_vol.bar(df.index, df['Volume'], width=1)

ax\_macd.plot(df.index, df['MACD'], label='MACD')

ax\_macd.plot(df.index, df['MACD\_signal'], label='Signal')

ax\_macd.bar(df.index, df['MACD\_hist'])

ax\_rsi.plot(df.index, df['RSI14'], label='RSI14')

ax\_rsi.axhline(70, linestyle='--')

ax\_rsi.axhline(30, linestyle='--')

plt.suptitle(title or 'Price & Indicators')

plt.tight\_layout(); plt.show()