# app.py

import streamlit as st

import yfinance as yf

import pandas as pd

import pandas\_ta as ta

import numpy as np

from datetime import datetime, timedelta

# Page config and theme-like CSS to mimic Investing.com dark look

st.set\_page\_config(page\_title="RP Stock Analyzer", layout="wide", initial\_sidebar\_state="expanded")

st.markdown(

"""

<style>

.stApp { background-color: #0b1620; color: #E6E7E8; }

.header { color: #FFD84D; font-weight:700; }

.card { background: #0f2430; padding:12px; border-radius:8px; box-shadow: 0 2px 10px rgba(0,0,0,0.5); }

.small { color:#bfc7ca; font-size:13px }

.signal-buy { color:#00c853; font-weight:700 }

.signal-sell { color:#ff5252; font-weight:700 }

.signal-neutral { color:#ffd54f; font-weight:700 }

table td, table th { color: #E6E7E8; }

</style>

""",

unsafe\_allow\_html=True,

)

st.markdown('<div class="header" style="font-size:28px">RP Stock Analyzer — Investing.com style</div>', unsafe\_allow\_html=True)

st.markdown('<div class="small">Indian market (NSE/BSE) indicators: RSI, MACD, Stoch, ATR, ADX, CCI, EMAs, Pivot Points, Volume & Summary</div>', unsafe\_allow\_html=True)

st.write("")

# Sidebar inputs

with st.sidebar:

st.markdown("## Inputs")

st.info("Use NSE tickers like: TCS.NS, RELIANCE.NS, INFY.NS, HDFCBANK.NS, SBIN.NS")

ticker = st.text\_input("Ticker (yfinance)", value="TCS.NS")

period = st.selectbox("History period", ["6mo","1y","2y","90d"], index=1)

interval = st.selectbox("Interval", ["1d","1wk","1h"], index=0)

fetch = st.button("Analyze")

def fetch\_data(ticker, period, interval):

try:

t = yf.Ticker(ticker)

df = t.history(period=period, interval=interval, auto\_adjust=False)

if df is None or df.empty:

return None

df = df.dropna()

return df

except Exception as e:

return None

def compute\_all(df):

d = df.copy()

close = d["Close"]

high = d["High"]

low = d["Low"]

open\_ = d["Open"]

# Indicators

d["RSI\_14"] = ta.rsi(close, length=14)

stoch = ta.stoch(high, low, close, k=14, d=3)

d = pd.concat([d, stoch], axis=1)

d["STOCHRSI"] = ta.stochrsi(close, length=14)

macd = ta.macd(close, fast=12, slow=26, signal=9)

d = pd.concat([d, macd], axis=1)

d["ADX\_14"] = ta.adx(high, low, close, length=14)["ADX\_14"]

d["CCI\_14"] = ta.cci(high, low, close, length=14)

d["ATR\_14"] = ta.atr(high, low, close, length=14)

d["ULTOSC"] = ta.ultosc(high, low, close)

d["ROC\_12"] = ta.roc(close, length=12)

d["EMA\_8"] = ta.ema(close, length=8)

d["EMA\_20"] = ta.ema(close, length=20)

d["EMA\_50"] = ta.ema(close, length=50)

d["SMA\_200"] = ta.sma(close, length=200)

d["VolumeMA20"] = d["Volume"].rolling(20).mean()

# Bull/Bear (close - ema20)

d["BullBear\_20"] = d["Close"] - d["EMA\_20"]

# Pivot Points based on previous completed candle

if len(d) >= 2:

prev = d.iloc[-2]

else:

prev = d.iloc[-1]

H = prev["High"]; L = prev["Low"]; C = prev["Close"]; O = prev["Open"]

P = (H + L + C) / 3.0

range\_hl = H - L

pivots = {

"Classic": {

"S3": round(L - 2\*(H - P),2),"S2": round(P - (H - L),2),"S1": round(2\*P - H,2),

"Pivot": round(P,2),"R1": round(2\*P - L,2),"R2": round(P + (H - L),2),"R3": round(H + 2\*(P - L),2)

},

"Fibonacci": {

"S3": round(P - 1.000\*range\_hl,2),"S2": round(P - 0.618\*range\_hl,2),"S1": round(P - 0.382\*range\_hl,2),

"Pivot": round(P,2),"R1": round(P + 0.382\*range\_hl,2),"R2": round(P + 0.618\*range\_hl,2),"R3": round(P + 1.000\*range\_hl,2)

}

}

return d, pivots

def make\_signals(latest):

sig = {}

try:

rsi = latest["RSI\_14"]

sig["RSI"] = "Buy" if rsi < 30 else ("Sell" if rsi > 70 else ("Buy" if rsi>=50 else "Neutral"))

except: sig["RSI"]="N/A"

try:

macdh = latest.get("MACDh\_12\_26\_9", latest.get("MACDh", np.nan))

sig["MACD"] = "Buy" if macdh > 0 else "Sell"

except: sig["MACD"]="N/A"

try:

k = latest.get("STOCHk\_14\_3\_3", latest.get("STOCHk\_14\_3\_3", np.nan))

dline = latest.get("STOCHd\_14\_3\_3", latest.get("STOCHd\_14\_3\_3", np.nan))

sig["Stoch"] = "Buy" if k > dline else "Sell"

except: sig["Stoch"]="N/A"

try:

adx = latest["ADX\_14"]; sig["ADX"] = "Trend" if adx > 25 else "Weak"

except: sig["ADX"]="N/A"

try:

cci = latest["CCI\_14"]; sig["CCI"] = "Buy" if cci > 100 else ("Sell" if cci < -100 else "Neutral")

except: sig["CCI"]="N/A"

try:

bb = latest["BullBear\_20"]; sig["BullBear"] = "Buy" if bb > 0 else "Sell"

except: sig["BullBear"]="N/A"

# MAs

price = latest["Close"]

ma\_signals = {}

for name in ["EMA\_8","EMA\_20","EMA\_50","SMA\_200"]:

v = latest.get(name, np.nan)

if np.isnan(v): ma\_signals[name] = "N/A"

else: ma\_signals[name] = "Buy" if price > v else "Sell"

sig["MAs"] = ma\_signals

return sig

# Main run

if fetch:

if not ticker:

st.warning("Enter a ticker (example: TCS.NS)")

else:

with st.spinner("Fetching data..."):

df = fetch\_data(ticker, period, interval)

if df is None or df.empty:

st.error("No data for that ticker/interval. Use NSE tickers like TCS.NS or try different interval/period.")

else:

df\_ind, pivots = compute\_all(df)

latest = df\_ind.iloc[-1]

st.markdown(f'<div class="card"><b>{ticker}</b> &nbsp; &nbsp; Latest: {latest.name.date() if hasattr(latest.name, "date") else latest.name} &nbsp; Close: <b>{latest["Close"]:.2f}</b></div>', unsafe\_allow\_html=True)

# Top row: summary

col1, col2, col3 = st.columns([1,1,2])

with col1:

st.markdown("### Signals")

signals = make\_signals(latest)

buy = sum(1 for v in (list(signals.values()) if "MAs" not in signals else [signals[k] for k in signals if k!="MAs"]) if v=="Buy")

sell = sum(1 for v in (list(signals.values()) if "MAs" not in signals else [signals[k] for k in signals if k!="MAs"]) if v=="Sell")

overall = "Neutral"

if buy+sell>0:

ratio = buy/(buy+sell)

overall = "Strong Buy" if ratio>=0.7 else ("Buy" if ratio>=0.55 else ("Strong Sell" if ratio<=0.3 else ("Sell" if ratio<=0.45 else "Neutral")))

st.markdown(f'<div class="card"><b style="font-size:20px">{overall}</b><div class="small">Buy {buy} — Sell {sell}</div></div>', unsafe\_allow\_html=True)

# small table

st.write("")

st.table(pd.DataFrame({

"Indicator": ["RSI(14)","MACD hist","ADX(14)","ATR(14)","Bull/Bear(20)"],

"Value": [round(latest.get("RSI\_14",np.nan),2), round(latest.get("MACDh\_12\_26\_9", latest.get("MACDh", np.nan)),3),

round(latest.get("ADX\_14",np.nan),2), round(latest.get("ATR\_14",np.nan),3), round(latest.get("BullBear\_20",np.nan),3)],

"Signal":[signals.get("RSI"), signals.get("MACD"), signals.get("ADX"), round(latest.get("ATR\_14",np.nan),3), signals.get("BullBear")]

}).set\_index("Indicator"))

with col2:

st.markdown("### Moving Averages")

ma\_df = pd.DataFrame([

["EMA 8", round(latest.get("EMA\_8",np.nan),3), signals["MAs"]["EMA\_8"]],

["EMA 20", round(latest.get("EMA\_20",np.nan),3), signals["MAs"]["EMA\_20"]],

["EMA 50", round(latest.get("EMA\_50",np.nan),3), signals["MAs"]["EMA\_50"]],

["SMA 200", round(latest.get("SMA\_200",np.nan),3), signals["MAs"]["SMA\_200"]],

], columns=["MA","Value","Signal"])

st.table(ma\_df.set\_index("MA"))

with col3:

st.markdown("### Volume")

st.metric("Latest Volume", f"{int(latest['Volume']):,}")

st.metric("20-day Avg Volume", f"{int(latest['VolumeMA20']):,}")

# Price chart (simple)

st.markdown("## Price (Close) — last 500 points")

st.line\_chart(df["Close"].tail(500))

# Pivots

st.markdown("## Pivot Points (previous candle)")

for name, table in pivots.items():

st.markdown(f"\*\*{name}\*\*")

st.table(pd.DataFrame(table, index=[0]).T.rename(columns={0:"Value"}))

# Show last rows with indicators

st.markdown("## Latest indicator values")

display\_df = df\_ind[["Close","RSI\_14","STOCHk\_14\_3\_3","STOCHd\_14\_3\_3","MACDh\_12\_26\_9","ADX\_14","CCI\_14","ATR\_14","EMA\_20","EMA\_50","SMA\_200"]].tail(10)

st.dataframe(display\_df.style.format("{:.3f}"))

st.success("Analysis ready. Use ticker like TCS.NS or RELIANCE.NS; change period/interval in sidebar.")

else:

st.info("Enter ticker (example TCS.NS) and click Analyze to run RP Stock Analyzer.")