# app.py - V20 Dashboard Pro complet, stylé, pédagogique et sectoriel

import streamlit as st

import yfinance as yf

import pandas as pd

import plotly.express as px

from math import isnan

# ---- Page config ----

st.set\_page\_config(page\_title="Analyse Actions - Dashboard V20", layout="wide")

# ---- CSS ----

st.markdown("""

<style>

.stApp { background-color:#0f1720; color:#e6eef6; }

.big-title { font-size:32px; font-weight:700; color:#f8fafc; margin-bottom:0.25rem; }

.muted { color:#9aa6b2; }

.progress-bar { border-radius:8px; height:24px; margin-bottom:6px; }

.tooltip { color:#cfe7ff; font-size:13px; }

.tab-header { font-weight:700; margin-bottom:0.25rem; font-size:18px; }

</style>

""", unsafe\_allow\_html=True)

# ---- Sidebar ----

with st.sidebar:

st.header("Recherche")

ticker\_input = st.text\_input("Ticker (ex: AAPL, MSFT, EXA.PA)", value="AAPL")

trading\_mode = st.selectbox("Profil trading", ["Équilibré", "Long terme / Value", "Court terme / Spéculatif"])

aggressive = st.checkbox("Mode agressif", value=False)

st.write("---")

st.markdown("⚙️ Entrez le ticker exact, avec suffixe marché si nécessaire")

# ---- Helper functions ----

def safe(val, default=None):

if val is None or (isinstance(val,float) and (pd.isna(val) or val!=val)):

return default

return val

def clip\_progress(val, max\_abs=1):

try:

val\_float = float(val)

except:

val\_float = 0

percent = int(abs(val\_float)/max\_abs\*100)

return max(0, min(percent, 100))

SECTOR\_SCALES = {

"Biotechnology": {"PE":(0,40,80),"P/S":(0,6,12),"Profit Margin":(0.0,0.05,0.10)},

"Healthcare": {"PE":(0,25,45),"P/S":(0,4,8),"Profit Margin":(0.05,0.10,0.15)},

"Industrial": {"PE":(0,15,25),"P/S":(0,2,4),"Profit Margin":(0.05,0.10,0.15)},

"Technology": {"PE":(0,30,60),"P/S":(0,5,12),"Profit Margin":(0.05,0.10,0.20)},

"Financial Services": {"PE":(0,12,20),"P/S":(0,3,6),"Profit Margin":(0.05,0.12,0.18)}

}

DEFAULT\_SCALES = {"PE":(0,20,40),"P/S":(0,3,6),"Profit Margin":(0.05,0.10,0.15)}

tooltips = {

"P/E": "Price / Earnings : <15 attractif, >30 spéculatif",

"P/S": "Price / Sales : faible = bon, élevé = spéculatif",

"P/B": "Price / Book : <1 = sous-évalué, >3 = cher",

"Profit Margin": "Marge nette : >10% solide, <5% faible",

}

@st.cache\_data(ttl=300)

def fetch\_info(ticker):

t = yf.Ticker(ticker)

try: info\_raw = t.info

except: info\_raw={}

info = {}

info['ticker']=ticker

info['longName']=info\_raw.get('longName') or ticker

info['sector']=info\_raw.get('sector') or "Unknown"

info['price']=safe(info\_raw.get('currentPrice'))

info['marketCap']=safe(info\_raw.get('marketCap'))

info['trailingPE']=safe(info\_raw.get('trailingPE'))

info['priceToSales']=safe(info\_raw.get('priceToSalesTrailing12Months'))

info['priceToBook']=safe(info\_raw.get('priceToBook'))

info['profitMargins']=safe(info\_raw.get('profitMargins'))

info['operatingMargins']=safe(info\_raw.get('operatingMargins'))

info['revenueGrowth']=safe(info\_raw.get('revenueGrowth'))

info['trailingEPS']=safe(info\_raw.get('trailingEps'))

info['totalCash']=safe(info\_raw.get('totalCash'))

info['totalDebt']=safe(info\_raw.get('totalDebt'))

info['currentRatio']=safe(info\_raw.get('currentRatio'))

info['bookValue']=safe(info\_raw.get('bookValue'))

info['sharesOutstanding']=safe(info\_raw.get('sharesOutstanding'))

try: hist = t.history(period="1y", actions=False)

except: hist=None

info['history']=hist

try:

cal = t.calendar

info['calendar'] = cal.to\_dict() if cal is not None else {}

except:

info['calendar'] = {}

return info

def interpret(val, key, sector=None):

if val is None: return ("N/A","gray","Donnée indisponible")

sector\_scales = SECTOR\_SCALES.get(sector, DEFAULT\_SCALES)

if key=="PE": low,med,high = sector\_scales['PE']

elif key=="P/S": low,med,high = sector\_scales['P/S']

elif key=="Profit Margin": low,med,high = sector\_scales['Profit Margin']

else: return (str(val),"gray","")

if key=="Profit Margin": val\_display=f"{val\*100:.1f}%"

else: val\_display=f"{val:.2f}"

if key=="PE" or key=="P/S":

if val < med: return (val\_display,"green","Attractif")

if val > high: return (val\_display,"red","Élevé / spéculatif")

return (val\_display,"yellow","Dans la norme")

if key=="Profit Margin":

if val>0.10: return (val\_display,"green","Marge solide")

if val<0: return (val\_display,"red","Perte / marge négative")

return (val\_display,"yellow","Marge modeste")

return (val\_display,"gray","")

def compute\_score(info, aggressive=False, mode="Équilibré"):

score=0

if mode=="Long terme / Value":

weight={'valuation':4,'growth':2,'margins':3,'balance':3}

elif mode=="Court terme / Spéculatif":

weight={'valuation':2,'growth':4,'margins':1,'balance':1}

else:

weight={'valuation':3,'growth':3,'margins':2,'balance':2}

pe=info.get('trailingPE'); ps=info.get('priceToSales'); eps=info.get('trailingEPS'); sector=info.get('sector')

if eps and eps>0 and pe: \_,color,\_=interpret(pe,"PE",sector)

elif ps: \_,color,\_=interpret(ps,"P/S",sector)

else: color="yellow"

score += weight['valuation']\*(1 if color=="green" else 0.6 if color=="yellow" else 0.2)

growth\_val=info.get('revenueGrowth');\_,color\_g,\_=interpret(growth\_val,"Profit Margin")

score += weight['growth']\*(1 if color\_g=="green" else 0.6 if color\_g=="yellow" else 0.2)

pm\_val=info.get('profitMargins');\_,color\_m,\_=interpret(pm\_val,"Profit Margin")

score += weight['margins']\*(1 if color\_m=="green" else 0.6 if color\_m=="yellow" else 0.2)

cr\_val=info.get('currentRatio');\_,color\_cr,\_=interpret(cr\_val,"PE")

score += weight['balance']\*(1 if color\_cr=="green" else 0.6 if color\_cr=="yellow" else 0.2)

maxscore=sum(weight.values())

final\_score=(score/maxscore)\*10

if aggressive: final\_score+=0.3

final\_score=max(0,min(10,final\_score))

if final\_score>=7: verdict=("BUY","green","Acheter")

elif final\_score>=4.5: verdict=("WATCH","yellow","Surveiller")

else: verdict=("AVOID","red","Éviter")

return round(final\_score,1), verdict

# ---- Main ----

if not ticker\_input:

st.info("Entrez un ticker dans la sidebar")

st.stop()

with st.spinner("Récupération des données..."):

info = fetch\_info(ticker\_input)

# ---- Tabs ----

tab1, tab2, tab3, tab4 = st.tabs(["Overview", "Analyse détaillée", "Graphiques", "Conseil trading"])

with tab1:

st.markdown(f"### {info['longName']} ({info['ticker']})")

st.markdown(f"\*\*Prix actuel : {info['price']} €\*\* | Secteur : {info['sector']}")

with tab2:

st.subheader("🟢 Étape 1 : Valorisation")

for label, val, key in [("P/E", info['trailingPE'], "PE"),

("P/S", info['priceToSales'], "P/S"),

("P/B", info['priceToBook'], "P/S")]:

disp,color,exp = interpret(val,key,info['sector'])

display\_val = f"{val:.2f}" if val is not None else "N/A"

st.markdown(f"{label} : {display\_val} ({exp})")

st.progress(clip\_progress(val or 0, max\_abs=val if val else 1))

st.subheader("📈 Étape 2 : Croissance")

rg\_val, rg\_color, rg\_exp = interpret(info['revenueGrowth'],"Profit Margin")

st.markdown(f"Croissance revenus : {rg\_val} ({rg\_exp})")

st.progress(clip\_progress(info['revenueGrowth'], max\_abs=0.5))

st.subheader("💹 Étape 3 : Marges")

pm\_val, pm\_color, pm\_exp = interpret(info['profitMargins'],"Profit Margin")

om\_val, om\_color, om\_exp = interpret(info['operatingMargins'],"Profit Margin")

st.markdown(f"Marge nette : {pm\_val} ({pm\_exp})")

st.progress(clip\_progress(info['profitMargins'], max\_abs=0.5))

st.markdown(f"Marge opérationnelle : {om\_val} ({om\_exp})")

st.progress(clip\_progress(info['operatingMargins'], max\_abs=0.5))

st.subheader("🏦 Étape 4 : Bilan / Levier")

cr\_val, cr\_color,\_ = interpret(info['currentRatio'],"PE")

debt\_ratio = info['totalDebt']/info['totalCash'] if info['totalCash'] else None

dr\_color = "green" if debt\_ratio and debt\_ratio<1 else "yellow" if debt\_ratio and debt\_ratio<2 else "red"

st.markdown(f"Current Ratio : {cr\_val}")

st.progress(clip\_progress(info['currentRatio'], max\_abs=3))

st.markdown(f"Dette / Cash : {debt\_ratio:.2f}" if debt\_ratio else "Dette / Cash : N/A")

# ---- Volatilité ----

if info['history'] is not None:

hist = info['history']['Close']

vol\_1y = hist.pct\_change().std()

vol\_1m = hist.tail(22).pct\_change().std()

st.subheader("📉 Volatilité")

st.markdown(f"Volatilité 1 an : {vol\_1y:.2%}")

st.markdown(f"Volatilité 1 mois : {vol\_1m:.2%}")

if vol\_1m > vol\_1y: st.markdown("⚠️ Volatilité récente supérieure à la moyenne 1 an : prudence")

else: st.markdown("✅ Volatilité modérée récente")

with tab3:

if info['history'] is not None:

st.subheader("📊 Historique prix")

fig = px.line(info['history'], x=info['history'].index, y='Close',

labels={'Close':'Prix (€)'}, title='Historique 1 an')

st.plotly\_chart(fig, use\_container\_width=True)

with tab4:

score, verdict = compute\_score(info, aggressive, trading\_mode)

st.markdown(f'<h2 style="color:{verdict[1]}; text-align:center;">Verdict : {verdict[0]} ({score}/10) - {verdict[2]}</h2>', unsafe\_allow\_html=True)

st.markdown("### Conseils personnalisés :")

if info['history'] is not None:

recent\_vol = info['history']['Close'].pct\_change().tail(10).std()

if recent\_vol>0.03: st.markdown("- Volatilité récente élevée : prudence")

else: st.markdown("- Volatilité récente modérée")

if info['calendar']:

st.markdown("- Événements à venir :")

for k,v in info['calendar'].items():

st.markdown(f" - {k}: {v}")