import streamlit as st

import pandas as pd

import random

st.set\_page\_config(page\_title="Paulistão Compacto 2025 – Totalmente Interativo", layout="wide")

st.title("Paulistão Compacto 2025 – Totalmente Interativo")

# 1️⃣ Nomes dos times

st.subheader("Editar nomes dos 16 times")

default\_teams = [

"Água Santa", "Botafogo-SP", "Bragantino", "Corinthians",

"Guarani", "Inter de Limeira", "Mirassol", "Noroeste",

"Novorizontino", "Palmeiras", "Ponte Preta", "Portuguesa",

"Red Bull Bragantino", "Santos", "São Bernardo", "São Paulo"

]

teams = []

for i in range(16):

name = st.text\_input(f"Time {i+1}", default\_teams[i])

teams.append(name)

# 2️⃣ Sorteio fase de grupos

if st.button("Sortear 5 jogos por time (fase de grupos)"):

random.seed(42)

pairs = [(teams[i], teams[j]) for i in range(16) for j in range(i+1, 16)]

random.shuffle(pairs)

rounds = [[] for \_ in range(5)]

team\_opponents = {t:set() for t in teams}

available\_pairs = pairs.copy()

def can\_place(pair, rnd):

a, b = pair

teams\_in\_round = {t for match in rounds[rnd] for t in match}

if a in teams\_in\_round or b in teams\_in\_round:

return False

if b in team\_opponents[a] or a in team\_opponents[b]:

return False

return True

for rnd in range(5):

attempts = 0

while len(rounds[rnd]) < 8 and attempts < 20000:

attempts += 1

if not available\_pairs:

break

pair = None

for p in available\_pairs:

if can\_place(p, rnd):

pair = p

break

if pair is None:

random.shuffle(available\_pairs)

continue

rounds[rnd].append(pair)

a,b = pair

team\_opponents[a].add(b)

team\_opponents[b].add(a)

available\_pairs = [pp for pp in available\_pairs if set(pp) != set(pair)]

if len(rounds[rnd]) < 8:

rounds = [[] for \_ in range(5)]

team\_opponents = {t:set() for t in teams}

available\_pairs = pairs.copy()

random.shuffle(available\_pairs)

rnd = -1

continue

home\_count = {t:0 for t in teams}

assign\_rounds = []

for rnd\_matches in rounds:

rnd\_assigned = []

for a,b in rnd\_matches:

if home\_count[a] < home\_count[b]:

home, away = a,b

elif home\_count[b] < home\_count[a]:

home, away = b,a

else:

if random.random() < 0.5:

home, away = a,b

else:

home, away = b,a

home\_count[home] += 1

rnd\_assigned.append((home, away))

assign\_rounds.append(rnd\_assigned)

rows = []

for r\_idx, rnd in enumerate(assign\_rounds, start=1):

for match in rnd:

rows.append({"Rodada": r\_idx, "Mandante": match[0], "Visitante": match[1],

"Gols Mandante": 0, "Gols Visitante": 0})

df\_rounds = pd.DataFrame(rows)

st.session\_state["fase\_grupos"] = df\_rounds

st.session\_state["matches"] = {}

st.success("Sorteio concluído! Agora você pode editar os placares da fase de grupos.")

# 3️⃣ Placar fase de grupos

if "fase\_grupos" in st.session\_state:

st.subheader("Editar Placar da Fase de Grupos")

df\_rounds = st.session\_state["fase\_grupos"]

edited\_df = st.data\_editor(df\_rounds, num\_rows="dynamic")

st.session\_state["fase\_grupos"] = edited\_df

# Cartões para desempate

st.subheader("Cartões para critério disciplinar")

cards\_df = pd.DataFrame({"Time": teams, "Cartões": [0]\*16})

cards\_df = st.data\_editor(cards\_df, num\_rows="fixed")

cards\_dict = dict(zip(cards\_df["Time"], cards\_df["Cartões"]))

# 4️⃣ Classificação oficial

points = {t:0 for t in teams}

gf = {t:0 for t in teams}

ga = {t:0 for t in teams}

matches = {}

for \_, row in edited\_df.iterrows():

h, a = row["Mandante"], row["Visitante"]

gh, ga\_ = int(row["Gols Mandante"]), int(row["Gols Visitante"])

gf[h] += gh

gf[a] += ga\_

ga[h] += ga\_

ga[a] += gh

matches[(h,a)] = (gh, ga\_)

if gh > ga\_:

points[h] += 3

elif gh < ga\_:

points[a] += 3

else:

points[h] += 1

points[a] += 1

classif = []

for t in teams:

classif.append({"Time": t, "Pontos": points[t], "GP": gf[t], "GC": ga[t],

"SG": gf[t]-ga[t], "Cartões": cards\_dict[t]})

df\_classif = pd.DataFrame(classif)

def desempate\_confronto(t1,t2):

if (t1,t2) in matches:

gh, ga\_ = matches[(t1,t2)]

elif (t2,t1) in matches:

ga\_, gh = matches[(t2,t1)]

else:

return 0

if gh > ga\_:

return 1

elif gh < ga\_:

return -1

return 0

def ranking(df):

df\_sorted = df.sort\_values(["Pontos","SG","GP"], ascending=[False, False, False]).reset\_index(drop=True)

i = 0

while i < len(df\_sorted)-1:

t1, t2 = df\_sorted.loc[i,"Time"], df\_sorted.loc[i+1,"Time"]

if df\_sorted.loc[i,"Pontos"] == df\_sorted.loc[i+1,"Pontos"] and \

df\_sorted.loc[i,"SG"] == df\_sorted.loc[i+1,"SG"] and \

df\_sorted.loc[i,"GP"] == df\_sorted.loc[i+1,"GP"]:

res = desempate\_confronto(t1,t2)

if res == -1:

df\_sorted.iloc[[i,i+1]] = df\_sorted.iloc[[i+1,i]].values

elif res == 0:

if df\_sorted.loc[i,"Cartões"] > df\_sorted.loc[i+1,"Cartões"]:

df\_sorted.iloc[[i,i+1]] = df\_sorted.iloc[[i+1,i]].values

elif df\_sorted.loc[i,"Cartões"] == df\_sorted.loc[i+1,"Cartões"]:

if random.random() < 0.5:

df\_sorted.iloc[[i,i+1]] = df\_sorted.iloc[[i+1,i]].values

i += 1

return df\_sorted

df\_classif = ranking(df\_classif)

st.subheader("Classificação Atualizada")

st.dataframe(df\_classif)

# 5️⃣ Fase final totalmente interativa

st.subheader("Fase Final – Registrar resultados de ida e volta")

top8 = df\_classif.head(8)["Time"].tolist()

quartas = [(top8[0], top8[7]), (top8[1], top8[6]), (top8[2], top8[5]), (top8[3], top8[4])]

# Função para criar dataframe interativo de mata-mata

def create\_knockout\_df(round\_name, matches):

rows = []

for idx, (h,a) in enumerate(matches, start=1):

rows.append({

"Partida": f"{round\_name} {idx}",

"Mandante (ida)": h, "Visitante (ida)": a,

"Gols Mandante (ida)": 0, "Gols Visitante (ida)": 0,

"Mandante (volta)": a, "Visitante (volta)": h,

"Gols Mandante (volta)": 0, "Gols Visitante (volta)": 0

})

return pd.DataFrame(rows)

# Quartas

df\_q = create\_knockout\_df("Q", quartas)

edited\_q = st.data\_editor(df\_q, num\_rows="dynamic")

st.session\_state["quartas"] = edited\_q

# Função para calcular vencedores

def get\_winners(df\_knockout):

winners = []

for \_, row in df\_knockout.iterrows():

ida\_h, ida\_a = int(row["Gols Mandante (ida)"]), int(row["Gols Visitante (ida)"])

volta\_h, volta\_a = int(row["Gols