import streamlit as st

import pandas as pd

import plotly.express as px

import plotly.graph\_objects as go

from datetime import datetime

from supabase import create\_client

import openai

# ページ設定（この行はスクリプト最上部でなければならない）

st.set\_page\_config(page\_title="財務ダッシュボード", layout="wide")

# タイトル表示

st.title("📊 財務データアップロード & 可視化アプリ")

# Supabase接続設定

@st.cache\_resource

def init\_supabase():

    url = st.secrets["supabase"]["url"]

    key = st.secrets["supabase"]["key"]

    return create\_client(url, key)

supabase = init\_supabase()

openai.api\_key = st.secrets["openai"]["api\_key"]

# 定数

TABLE\_NAME = "monthly\_pl"

DATE\_COLUMN = "date"

SALES\_COLUMN = "sales"

# GPTによる売上アドバイス

def generate\_sales\_advice(df: pd.DataFrame, sales\_col: str):

    if df.empty:

        return "データが存在しないため、分析できません。"

    csv\_data = df[['year\_month', sales\_col]].to\_csv(index=False)

    prompt = f"""

あなたは財務分析の専門家です。

以下のCSVデータは、企業の3年分の月次売上データです。

売上の傾向、注意点、改善アドバイスを500文字以内で日本語で要約してください。

データ:

{csv\_data}

"""

    try:

        response = openai.ChatCompletion.create(

            model="gpt-4",

            messages=[{"role": "user", "content": prompt}],

            temperature=0.7,

            max\_tokens=500

        )

        return response.choices[0].message.content.strip()

    except Exception as e:

        return f"GPT APIエラー: {e}"

# Supabaseからデータ取得

@st.cache\_data(ttl=600)

def fetch\_supabase\_data():

    try:

        res = supabase.table(TABLE\_NAME).select("\*").order(DATE\_COLUMN, desc=False).execute()

        return pd.DataFrame(res.data)

    except Exception as e:

        st.error(f"❌ Supabase取得エラー: {e}")

        return pd.DataFrame()

# データ処理関数

def process\_data(df):

    if df.empty:

        return df

    df[DATE\_COLUMN] = pd.to\_datetime(df[DATE\_COLUMN], errors="coerce")

    df[SALES\_COLUMN] = pd.to\_numeric(df[SALES\_COLUMN], errors="coerce")

    df.dropna(subset=[DATE\_COLUMN, SALES\_COLUMN], inplace=True)

    df["year"] = df[DATE\_COLUMN].dt.year

    df["year\_month"] = df[DATE\_COLUMN].dt.strftime('%Y-%m')

    latest\_year = df["year"].max()

    return df[df["year"].isin([latest\_year, latest\_year - 1, latest\_year - 2])]

# ローカルCSVアップロードと可視化

st.title("📄 ローカルCSVでの財務分析")

uploaded\_local = st.file\_uploader("ローカルCSVをアップロード", type=["csv"], key="local\_csv")

if uploaded\_local:

    df = pd.read\_csv(uploaded\_local)

    df["date"] = pd.to\_datetime(df["date"], errors='coerce')

    df.dropna(subset=["date"], inplace=True)

    df["変動費"] = df["outsourcing\_costs"] + df["commissions\_fees"]

    固定費列 = [

        "executive\_compensation", "salaries", "bonuses", "rent\_payments",

        "welfare\_expenses", "employee\_welfare", "supplies",

        "utilities", "communication", "transportation", "communication\_expenses",

        "advertising", "entertainment", "training\_expenses", "miscellaneous\_expenses"

    ]

    df["固定費"] = df[[col for col in 固定費列 if col in df.columns]].sum(axis=1)

    if "gross\_profit" in df.columns:

        df["粗利益"] = df["gross\_profit"]

    elif "cost\_of\_sales" in df.columns:

        df["粗利益"] = df["sales"] - df["cost\_of\_sales"]

    else:

        df["粗利益"] = df["sales"] - df["変動費"] - df["固定費"]

    df["経常利益"] = df["ordinary\_profit"] if "ordinary\_profit" in df.columns else df["粗利益"]

    df\_summary = df[["date", "sales", "変動費", "固定費", "粗利益", "経常利益"]].copy()

    df\_summary = df\_summary.rename(columns={"sales": "売上高"})

    # 📊 財務指標別 三期比較グラフ（1月はじまり）

    st.subheader("📊 財務指標別 三期比較グラフ（1月はじまり）")

    selected\_metric = st.selectbox("表示する財務指標を選んでください", ["売上高", "変動費", "固定費", "粗利益", "経常利益"])

    df\_summary["月ラベル"] = pd.to\_datetime(df\_summary["date"]).dt.strftime("%m月")

    df\_summary["年度"] = pd.to\_datetime(df\_summary["date"]).dt.year

    latest\_year = df\_summary["年度"].max()

    target\_years = [latest\_year - 2, latest\_year - 1, latest\_year]

    df\_summary = df\_summary[df\_summary["年度"].isin(target\_years)]

    year\_label\_map = {

        target\_years[0]: "前々期（R4）",

        target\_years[1]: "前期（R5）",

        target\_years[2]: "当期（R6）",

    }

    df\_summary["年度ラベル"] = df\_summary["年度"].map(year\_label\_map)

    month\_order = [f"{i:02d}月" for i in range(1, 13)]

    df\_summary["月ラベル"] = pd.Categorical(df\_summary["月ラベル"], categories=month\_order, ordered=True)

    df\_plot = df\_summary[["月ラベル", "年度ラベル", selected\_metric]].dropna()

    df\_plot = df\_plot.sort\_values(["月ラベル", "年度ラベル"])

    df\_pivot = df\_plot.pivot(index="月ラベル", columns="年度ラベル", values=selected\_metric).reset\_index()

    fig = go.Figure()

    colors = {

        "前々期（R4）": "darkgreen",

        "前期（R5）": "deepskyblue",

        "当期（R6）": "lightgreen"

    }

    for col in ["前々期（R4）", "前期（R5）", "当期（R6）"]:

        if col in df\_pivot.columns:

            fig.add\_trace(go.Bar(

                x=df\_pivot["月ラベル"],

                y=df\_pivot[col],

                name=col,

                text=df\_pivot[col].apply(lambda x: f"{x:,.0f}" if pd.notnull(x) else ""),

                textposition="auto",

                marker\_color=colors[col]

            ))

    fig.update\_layout(

        title=f"{selected\_metric} 三期比較グラフ（1月〜12月）",

        xaxis\_title="月",

        yaxis\_title="金額（円）",

        barmode="group",

        height=550,

        legend\_title="年度"

    )

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