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

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

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])]

# Supabase用CSVアップロード

uploaded\_file = st.file\_uploader("📤 Supabase用CSVをアップロード", type=["csv"])

if uploaded\_file:

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

    st.dataframe(df\_csv)

    if st.button("📥 Supabaseにアップロード"):

        try:

            data\_list = df\_csv.to\_dict(orient="records")

            success = 0

            for row in data\_list:

                res = supabase.table(TABLE\_NAME).insert(row).execute()

                if res.data:

                    success += 1

            fetch\_supabase\_data.clear()

            st.success(f"✅ {success} 件のデータをSupabaseに保存しました。")

        except Exception as e:

            st.error("❌ アップロードエラー:")

            st.code(str(e))

# Supabaseから取得して可視化

df\_supabase = fetch\_supabase\_data()

df\_processed = process\_data(df\_supabase)

if not df\_processed.empty:

    st.subheader("📊 月次売上の推移")

    fig = px.bar(df\_processed, x="year\_month", y=SALES\_COLUMN, color="year",

                 title="月次売上の推移", labels={"year\_month": "年月", SALES\_COLUMN: "売上高 (円)"})

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

    st.subheader("📋 売上データ（表形式）")

    st.dataframe(

        df\_processed[[DATE\_COLUMN, "year\_month", SALES\_COLUMN]]

        .rename(columns={DATE\_COLUMN: "日付", "year\_month": "年月", SALES\_COLUMN: "売上高"})

        .sort\_values("日付", ascending=False)

        .style.format({"売上高": "{:,.0f} 円"}),

        use\_container\_width=True

    )

    st.subheader("💬 GPTによる売上分析")

    with st.spinner("分析中..."):

        comment = generate\_sales\_advice(df\_processed, SALES\_COLUMN)

        st.markdown(comment)

else:

    st.info("📭 Supabaseデータが見つかりません。")

# ローカルCSVによる財務分析（Supabase非連携）

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["月"] = df["date"].dt.strftime("%Y-%m")

    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[["月", "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"})

    st.subheader("📅 月次財務データ一覧")

    st.dataframe(df\_summary)

    # PLツリーマップ

    st.subheader("🧩 利益構造（ツリーマップ）")

    selected\_month = st.selectbox("表示する月を選んでください", df\_summary["月"].unique())

    row = df\_summary[df\_summary["月"] == selected\_month].iloc[0]

    labels = ["売上高", "変動費", "固定費", "粗利益", "経常利益"]

    values = [row[l] / 1\_000\_000 for l in labels]

    parents = ["", "売上高", "粗利益", "売上高", "粗利益"]

    fig = px.treemap(

        names=labels,

        values=values,

        parents=parents,

        title=f"{selected\_month} の利益構造（百万円）"

    )

    fig.update\_traces(texttemplate="%{label}<br>%{value:.1f} 百万円")

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

    # 比較グラフ

    all\_months = df\_summary["月"].tolist()

    idx = all\_months.index(selected\_month)

    prev\_month = all\_months[idx - 1] if idx > 0 else None

    prev\_year = f"{int(selected\_month[:4]) - 1}-{selected\_month[5:]}"

    prev\_year = prev\_year if prev\_year in df\_summary["月"].values else None

    def show\_comparison(before, after, label):

        if not before or not after:

            st.info(f"{label}の比較データがありません")

            return

        b = df\_summary[df\_summary["月"] == before].iloc[0]

        a = df\_summary[df\_summary["月"] == after].iloc[0]

        delta = a["経常利益"] - b["経常利益"]

        ratio = (delta / b["経常利益"]) \* 100 if b["経常利益"] != 0 else 0

        fig = go.Figure()

        for name, data in zip([before, after], [b, a]):

            fig.add\_trace(go.Bar(

                x=["売上高", "変動費", "固定費", "粗利益", "経常利益"],

                y=[data["売上高"], data["変動費"], data["固定費"], data["粗利益"], data["経常利益"]],

                name=name

            ))

        fig.update\_layout(title=f"{label}比較: {before}→{after} 差分: {delta:,.0f}円 ({ratio:.1f}%)", barmode="group")

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

    show\_comparison(prev\_month, selected\_month, "前月")

    show\_comparison(prev\_year, selected\_month, "前年同月")