

```
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"X 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("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[1] / 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")

```