```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from faker import Faker
import random
from langchain.vectorstores import FAISS
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.docstore.document import Document
import google.generativeai as genai
import os

# Configure Gemini
GOOGLE_API_KEY = "AIzaSyDRS1nCrBdkiyv2HRIGpo_l0M4eghA4gGI"  # Replace
 with your actual key
genai.configure(api_key=GOOGLE_API_KEY)

# Streamlit setup
st.set_page_config(page_title="📊 Credit Dashboard + Gemini Bot", layout="wid
e")
st.title("📊 Generate → Analyze → Ask (Gemini + RAG Dashboard)")

tab1, tab2 = st.tabs(["🧪 Data Generator", "📈 Dashboard + 🤖 Chatbot"])

# --------- TAB 1 ---------
with tab1:
    st.subheader("🧪 Synthetic Credit Data Generator")
    fake = Faker()
    num_rows = st.number_input("Rows to generate:", min_value=10, max_value=
10000, value=100)

    if st.button("Generate Data"):
```

```
        data = []
        for _ in range(num_rows):
            data.append({
                "CustomerID": fake.unique.bothify(text='CUST-#####'),
                "PD": round(random.uniform(0.01, 0.3), 4),
                "LGD": round(random.uniform(0.2, 0.8), 4),
                "EAD": round(random.uniform(50000, 10000000), 2),
                "CreditRating": random.choice(["AAA", "AA", "A", "BBB", "BB", "B", "CCC",
"D"]),
                "Sector": random.choice(["Retail", "Manufacturing", "Technology", "Ban
king", "Insurance"]),
                "Region": random.choice(["North", "South", "East", "West"]),
                "Date": fake.date_between(start_date='-2y', end_date='today')
            })
        df = pd.DataFrame(data)
        st.session_state.generated_df = df
        st.success("✅ Data generated!")
        st.dataframe(df)
        st.download_button("📥 Download CSV", df.to_csv(index=False), "synthetic
_credit_data.csv", "text/csv")

# --------- TAB 2 ---------
with tab2:
    uploaded_file = st.file_uploader("📂 Upload CSV (optional if data generated)",
 type="csv")
    if uploaded_file:
        df = pd.read_csv(uploaded_file)
    elif "generated_df" in st.session_state:
        df = st.session_state.generated_df
    else:
        st.warning("📁 Upload or generate data to continue.")
        st.stop()
```

```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

st.header("📈 Dashboard")
st.dataframe(df)

col1, col2, col3 = st.columns(3)
col1.metric("Avg PD", f"{df['PD'].mean():.2%}")
col2.metric("Avg LGD", f"{df['LGD'].mean():.2%}")
col3.metric("Total EAD", f"{df['EAD'].sum():,.0f}")

c1, c2, c3 = st.columns(3)
with c1:
    fig1, ax1 = plt.subplots()
    sns.histplot(df['PD'], kde=True, bins=20, ax=ax1)
    st.pyplot(fig1)
with c2:
    fig2, ax2 = plt.subplots()
    sns.histplot(df['LGD'], kde=True, bins=20, ax=ax2)
    st.pyplot(fig2)
with c3:
    fig3, ax3 = plt.subplots()
    sns.histplot(df['EAD'], kde=True, bins=20, ax=ax3)
    st.pyplot(fig3)

st.markdown("### ⏳ EAD Over Time")
fig4, ax4 = plt.subplots(figsize=(10, 4))
timeline = df.groupby("Date")["EAD"].sum().reset_index()
sns.lineplot(data=timeline, x="Date", y="EAD", ax=ax4, marker="o")
st.pyplot(fig4)

# --------- Chatbot Section ---------
st.header("🤖 Ask Your Credit Analysis Assistant")
```

```
    def load_txt_docs(txt_path="regulations.txt"):
        if not os.path.exists(txt_path):
            return []
        with open(txt_path, "r") as f:
            text = f.read()
        return [Document(page_content=text)]

    @st.cache_resource
    def create_retriever(df, txt_path="regulations.txt"):
        docs = load_txt_docs(txt_path)
        rows = df.astype(str).apply(lambda x: " | ".join(x), axis=1).tolist()
        docs.extend([Document(page_content=row) for row in rows])
        splitter = CharacterTextSplitter(chunk_size=500, chunk_overlap=50)
        split_docs = splitter.split_documents(docs)
        embeddings = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
        return FAISS.from_documents(split_docs, embeddings).as_retriever()

    retriever = create_retriever(df)

    def generate_credit_prompt(query, context):
        return f"""
You are a financial risk analysis assistant. A user has shared credit data with P
D, LGD, EAD, Credit Rating, Region, Sector, and Date.

Context:
{context}

Now answer this question:
{query}

Be precise, data-driven, and easy to understand.
"""
```

```
def get_gemini_answer(query):
    docs = retriever.get_relevant_documents(query)
    context = "\n".join(doc.page_content for doc in docs[:5])
    prompt = generate_credit_prompt(query, context)
    model = genai.GenerativeModel('models/gemini-2.5-flash')
    response = model.generate_content(prompt)
    return response.text.strip()

if "chat" not in st.session_state:
    st.session_state.chat = []

query = st.chat_input("Ask about credit risk, Basel III, or trends...")
if query:
    answer = get_gemini_answer(query)
    st.session_state.chat.append(("user", query))
    st.session_state.chat.append(("bot", answer))

for role, msg in st.session_state.chat:
    st.chat_message("user" if role == "user" else "assistant").write(msg)
```