```
import numpy as np
import pandas as pd
from faker import Faker
import random

fake = Faker()
np.random.seed(42)
random.seed(42)

num_users = 10

def generate_financial_data(num_users):
    data = {
        'User_ID': [i for i in range(1, num_users + 1)],
        'Income': np.random.uniform(30000, 150000, num_users).round
        'Expenses': np.random.uniform(5000, 50000, num_users).round
        'HealthInsurance': np.random.uniform(0, 5000, num_users).ro
        'HomeLoan': np.random.uniform(0, 10000, num_users).round(2)
        'ELSS': np.random.uniform(0, 5000, num_users).round(2),
        'NPS': np.random.uniform(0, 5000, num_users).round(2),
        'PPF': np.random.uniform(0, 5000, num_users).round(2),
        'HouseRent': np.random.uniform(0, 12000, num_users).round(2
        'Previous_Tax_Amount': np.random.uniform(2000, 20000, num_u
        'State': [fake.state_abbr() for _ in range(num_users)],
        'Filing_Status': [random.choice(['Single', 'Married', 'Head
        'Tax_Credits': np.random.uniform(0, 5000, num_users).round(
    }

    for column in ['HealthInsurance', 'HomeLoan', 'ELSS', 'NPS', 'P
        data[column] = [value if random.random() > 0.5 else 0 for v

    df = pd.DataFrame(data)
    return df

financial_data = generate_financial_data(num_users)
financial_data
```

Release notes quantizer_bnb_4b •••

You are not subscribed. Learn more

You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.

At your current usage level, this runtime may last up to 3 hours 30 minutes.

Manage sessions

Want more memory and disk space?

Upgrade to Colab Pro

Python 3 Google Compute Engine backend (GPU)

Showing resources from 10:09 PM to 10:33 PM

System RAM
3.5 / 12.7 GB

GPU RAM
4.9 / 15.0 GB

Disk
51.2 / 112.6 GB

	User_ID	Income	Expenses	HealthInsurance	HomeLoan	
0	1	74944.81	5926.30	3059.26	6075.45	
1	2	144085.72	48645.94	697.47	0.00	
2	3	117839.27	42459.92	0.00	650.52	
3	4	101839.02	14555.26	0.00	9488.86	45
4	5	48722.24	13182.12	0.00	0.00	12
5	6	48719.34	13253.20	0.00	8083.97	33
6	7	36970.03	18690.90	998.37	3046.14	15
7	8	133941.14	28614.04	0.00	0.00	26
8	9	102133.80	24437.53	0.00	0.00	27
9	10	114968.71	18105.31	232.25	4401.52	

Next steps: code financial_data recommended interactive

```
import pandas as pd

def generate_tax_regulations():
    tax_brackets = ['10% - $0 to $10,000', '12% - $10,001 to $40,0
                    '24% - $85,001 to $160,000', '32% - $160,001 t
    standard_deductions = [12000] * len(tax_brackets)
    tax_credits = [500, 1000, 1500, 2500, 3000, 4500]

    regulations = {
        'Tax_Bracket': tax_brackets,
        'Standard_Deductions': standard_deductions,
        'Tax_Credits': tax_credits
    }
    df = pd.DataFrame(regulations)
    return df

tax_regulations = generate_tax_regulations()
tax_regulations
```

	Tax_Bracket	Standard_Deductions	Tax_Credits
0	10% - $0 to $10,000	12000	500
1	12% - $10,001 to $40,000	12000	1000
2	22% - $40,001 to $85,000	12000	1500
3	24% - $85,001 to $160,000	12000	2500

Next steps: code tax_regulations recommended interactive

```
# Apply tax regulations to the financial data
def apply_tax_regulations(financial_df, regulations_df):
    # Simplified model for applying tax brackets and deductions
    def calculate_tax(user_income, deductions, standard_deductions
        # Determine tax rate based on income
        if user_income <= 10000:
            tax_rate = 0.10
        elif user_income <= 40000:
            tax_rate = 0.12
        elif user_income <= 85000:
            tax_rate = 0.22
        elif user_income <= 160000:
            tax_rate = 0.24
        elif user_income <= 200000:
            tax_rate = 0.32
        else:
            tax_rate = 0.35

        # Assuming standard deduction applies regardless of filing
        standard_deduction = standard_deductions
        taxable_income = max(user_income - deductions - standard_d
        return taxable_income * tax_rate

    # Assuming we use the first row of the regulations_df for simp
    standard_deductions = regulations_df['Standard_Deductions'].il

    # Calculate estimated tax for each user
    financial_df['Estimated_Tax'] = financial_df.apply(
        lambda row: calculate_tax(row['Income'], row[['HealthInsur
        axis=1
    )
    return financial_df

# Generate fake financial data
num_users = 1000
financial_data = generate_financial_data(num_users)

# Apply tax regulations to the financial data
financial_data_with_taxes = apply_tax_regulations(financial_data,
financial_data_with_taxes.head()
```

	User_ID	Income	Expenses	HealthInsurance	HomeLoan	
0	1	33771.50	7602.92	4575.45	926.25	31
1	2	106369.25	48609.62	0.00	0.00	
2	3	67722.72	44770.36	789.77	9145.49	27
3	4	91028.48	46748.85	3479.50	0.00	
4	5	138907.98	49770.85	0.00	2587.12	

Next steps: code financial_data_with_taxes recommended

```
pip install langchain-community
```

```
Requirement already satisfied: langchain-community in /usr/loc
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/p
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/loca
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/l
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 ir
Requirement already satisfied: langchain<0.4.0,>=0.3.1 in /usr
Requirement already satisfied: langchain-core<0.4.0,>=0.3.6 ir
Requirement already satisfied: langsmith<0.2.0,>=0.1.125 in /u
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/p
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0
Requirement already satisfied: requests<3,>=2 in /usr/local/li
Requirement already satisfied: tenacity!=8.4.0,<9.0.0,>=8.1.0
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/loc
Requirement already satisfied: yarl<2.0,>=1.12.0 in /usr/local
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in /
Requirement already satisfied: typing-inspect<1,>=0.4.0 in /us
Requirement already satisfied: langchain-text-splitters<0.4.0,
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/lo
Requirement already satisfied: packaging<25,>=23.2 in /usr/loc
Requirement already satisfied: typing-extensions>=4.7 in /usr/
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/l
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/l
Requirement already satisfied: charset-normalizer<4,>=2 in /us
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/loca
Requirement already satisfied: certifi>=2017.4.17 in /usr/loca
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/
Requirement already satisfied: anyio in /usr/local/lib/python3
Requirement already satisfied: httpcore==1.* in /usr/local/lib
Requirement already satisfied: sniffio in /usr/local/lib/pytho
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/l
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/
Requirement already satisfied: annotated-types>=0.6.0 in /usr/
Requirement already satisfied: pydantic-core==2.23.4 in /usr/l
Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/
Requirement already satisfied: exceptiongroup in /usr/local/li
```

```
import pandas as pd
from langchain.docstore.document import Document
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import Chroma

# Prepare documents for LangChain
documents = []
for _, row in financial_data_with_taxes.iterrows():
    content = (f"User_ID: {row['User_ID']}, Income: {row['Income']
               f"HealthInsurance: {row['HealthInsurance']}, HomeLo
               f"ELSS: {row['ELSS']}, NPS: {row['NPS']}, PPF: {row
               f"Previous_Tax_Amount: {row['Previous_Tax_Amount']}
               f"Filing_Status: {row['Filing_Status']}, Tax_Credit
```

```
                f"Estimated_Tax: {row['Estimated_Tax']}")

    documents.append(Document(page_content=content))


documents=documents[:100]


hg_embeddings = HuggingFaceEmbeddings()
persist_directory = '/content/'

langchain_chroma = Chroma.from_documents(
    documents=documents,
    collection_name="financial_data",
    embedding=hg_embeddings,
    persist_directory=persist_directory
)
```

```
6ada9d77>:1: LangChainDeprecationWarning: The class `HuggingFa
gingFaceEmbeddings()
6ada9d77>:1: LangChainDeprecationWarning: Default values for H
gingFaceEmbeddings()
3.10/dist-packages/sentence_transformers/cross_encoder/CrossEn
ook import tqdm, trange
3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWar
 does not exist in your Colab secrets.
 the Hugging Face Hub, create a token in your settings tab (htt
euse this secret in all of your notebooks.
entication is recommended but still optional to access public
```

```
from torch import cuda, bfloat16, float16
import transformers
from transformers import AutoTokenizer
from transformers import AutoModelForCausalLM
from langchain.llms import HuggingFacePipeline
from time import time

model_id = 'HuggingFaceH4/zephyr-7b-beta'
device = f'cuda:{cuda.current_device()}' if cuda.is_available() el

bnb_config = transformers.BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type='nf4',
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=bfloat16
)

model = transformers.AutoModelForCausalLM.from_pretrained(
    model_id,
    trust_remote_code=True,
    quantization_config=bnb_config,
    device_map='auto',
)
tokenizer = AutoTokenizer.from_pretrained(model_id)
```

```
model.safetensors.index.json: 100%        23.9k/23.9k [00:00<00:00, 487kB/s]
Downloading shards: 100%                  8/8 [01:34<00:00, 9.63s/it]
model-00001-of-00008.safetensors: 100%    1.89G/1.89G [00:13<00:00, 134MB/s]
model-00002-of-00008.safetensors: 100%    1.95G/1.95G [00:11<00:00, 175MB/s]
model-00003-of-00008.safetensors: 100%    1.98G/1.98G [00:15<00:00, 248MB/s]
model-00004-of-00008.safetensors: 100%    1.95G/1.95G [00:09<00:00, 288MB/s]
model-00005-of-00008.safetensors: 100%    1.98G/1.98G [00:12<00:00, 165MB/s]
model-00006-of-00008.safetensors: 100%    1.95G/1.95G [00:16<00:00, 262MB/s]
model-00007-of-00008.safetensors: 100%    1.98G/1.98G [00:09<00:00, 250MB/s]
model-00008-of-00008.safetensors: 100%    816M/816M [00:02<00:00, 275MB/s]
```

```
query_pipeline = transformers.pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    torch_dtype=float16,
    max_new_tokens=500,
    device_map="auto",
)
llm = HuggingFacePipeline(pipeline=query_pipeline)
```

```
<ipython-input-17-a8dc69e43169>:9: LangChainDeprecationWarning
  llm = HuggingFacePipeline(pipeline=query_pipeline)
```

```
from langchain.chains import RetrievalQA
from langchain.prompts import PromptTemplate

template = """
Based on the following financial data and tax regulations, analyze
Financial Data: {question}
Context: {context}
Answer:
"""
PROMPT = PromptTemplate(input_variables=["context", "query"], temp

# Set up retriever
retriever = langchain_chroma.as_retriever(search_kwargs={"k": 5})
```

```
# Function to remove duplicates from retrieved documents
def remove_duplicates(documents):
    seen = set()
    unique_docs = []
    for doc in documents:
        if doc.page_content not in seen:
            unique_docs.append(doc)
            seen.add(doc.page_content)
    return unique_docs

# Set up the QA chain
qa_chain = RetrievalQA.from_chain_type(
    llm, retriever=retriever, chain_type_kwargs={"prompt": PROMPT}
)

def get_tax_optimization_recommendations(query):
    # Retrieve documents
    raw_docs = retriever.get_relevant_documents(query)

    # Remove duplicates
    unique_docs = remove_duplicates(raw_docs)

    # Prepare the context for the prompt
    context = " ".join([doc.page_content for doc in unique_docs])

    # Use the QA chain to get the response
    result = qa_chain({"context": context, "query": query})
    return result

# Example query
query = "Analyze - User_ID: 317, Income: 65185.29, Expenses: 6770.
response = get_tax_optimization_recommendations(query)
```

```
<ipython-input-18-40f8880f770a>:32: LangChainDeprecationWarnir
  raw_docs = retriever.get_relevant_documents(query)
<ipython-input-18-40f8880f770a>:41: LangChainDeprecationWarnir
  result = qa_chain({"context": context, "query": query})
Starting from v4.46, the `logits` model output will have the s
```

```
print(response['result'])
```

```

Based on the following financial data and tax regulations, ana
Financial Data: Analyze - User_ID: 317, Income: 65185.29, Expe
Context: User_ID: 40, Income: 146613.85, Expenses: 31658.35, H

User_ID: 54, Income: 88734.33, Expenses: 35606.33, HealthInsur

User_ID: 13, Income: 141563.72, Expenses: 15071.79, HealthInsu

User_ID: 14, Income: 126974.45, Expenses: 29163.85, HealthInsu

User_ID: 36, Income: 68784.35, Expenses: 8425.89, HealthInsura
Answer:

Based on the financial data and tax regulations provided, here

1. User_ID: 317
   - Maximize health insurance deductions by increasing the pr
   - Consider investing in ELSS or NPS to avail tax benefits u
```

```
   - Claim tax credits for professional tax, tuition fees, anc
   - Adjust the taxable income by claiming deductions for medi
   - Calculate the tax liability accurately to avoid underpaym

2. User_ID: 40
   - Claim tax credits for professional tax, tuition fees, anc
   - Adjust the taxable income by claiming deductions for medi
   - Consider investing in ELSS or NPS to avail tax benefits u
   - Calculate the tax liability accurately to avoid underpaym

3. User_ID: 54
   - Claim tax credits for professional tax, tuition fees, anc
   - Adjust the taxable income by claiming deductions for medi
   - Consider investing in ELSS or NPS to avail tax benefits u
   - Calculate the tax liability accurately to avoid underpaym

4. User_ID: 13
   - Claim tax credits for professional tax, tuition fees, anc
   - Adjust the taxable income by claiming deductions for medi
   - Consider investing in ELSS or NPS to avail tax benefits u
   - Calculate the tax liability accurately to avoid underpaym
```

Change runtime type