```
!pip install langchain
!pip install langchain_community
# !pip install tiktoken
# !pip install faiss-cpu
# !pip install chromadb
!pip install edgartools
!pip install langchain-text-splitters python-dotenv
!pip install langchain-google-genai
!pip install -q transformers sentence-transformers qdrant-client langchain
# !pip install pandas numpy pathlib
```

Show hidden output

```
!pip install edgartools
```

Show hidden output

```
import os
import pandas as pd
from edgar import set_identity,Company
import numpy as np
import json
from pathlib import Path

from langchain_community.document_loaders import DirectoryLoader, TextLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain_community.embeddings import SentenceTransformerEmbeddings
from langchain_community.vectorstores import Qdrant
from langchain.schema import Document
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate
```

```
os.environ["GOOGLE_API_KEY"] = "Your_API_Key"
```

```
llm = ChatGoogleGenerativeAI(model="gemini-2.0-flash")
```

Data Collection

```
set_identity("K sha kt@codes.finance")

CIK_MAP = {
    "GOOGL": "GOOGL",
    "MSFT": "MSFT",
    "NVDA": "NVDA"
}

YEARS = [2022, 2023, 2024]
```

```
SAVE_DIR = "sec_filings"
os.makedirs(SAVE_DIR, exist_ok=True)


def download_10k_for_company(ticker):
    print(f"\nProcessing: {ticker}")
    company = Company(ticker)
    filings = company.get_filings(form="10-K")

    df = filings.to_pandas()
    df['filing_date'] = pd.to_datetime(df['filing_date'])

    for year in YEARS:
        match = df[df['filing_date'].dt.year == year]
        if match.empty:
            print(f"No 10-K filing found for {ticker} in {year}")
            continue

        filing_date = match.iloc[0]['filing_date']
        filing = filings.filter(date=filing_date.strftime('%Y-%m-%d'))
        filing_obj = filing.latest().obj()

        print(f"Downloading {ticker} 10-K for {year} (filed on {filing_date.date(

        try:
            content = filing_obj.items
            filename = f"{ticker}_{year}_10K.txt"
            filepath = os.path.join(SAVE_DIR, filename)

            with open(filepath, 'a', encoding='utf-8') as f:
                for i in content:
                    x = filing_obj[i]
                    f.write(x)

            print(f"Saved to {filepath}")
        except Exception as e:
            print(f"Error saving {ticker} {year}: {e}")

if __name__ == "__main__":
    for ticker in CIK_MAP.values():
        download_10k_for_company(ticker)
```

```
Processing: GOOGL
Downloading GOOGL 10-K for 2022 (filed on 2022-02-02)...
Saved to sec_filings/GOOGL_2022_10K.txt
Downloading GOOGL 10-K for 2023 (filed on 2023-02-03)...
Saved to sec_filings/GOOGL_2023_10K.txt
Downloading GOOGL 10-K for 2024 (filed on 2024-01-31)...
Saved to sec_filings/GOOGL_2024_10K.txt

Processing: MSFT
Downloading MSFT 10-K for 2022 (filed on 2022-07-28)...
Saved to sec_filings/MSFT_2022_10K.txt
Downloading MSFT 10-K for 2023 (filed on 2023-07-27)...
Saved to sec_filings/MSFT_2023_10K.txt
```

```
Downloading MSFT 10-K for 2024 (filed on 2024-07-30)...
Saved to sec_filings/MSFT_2024_10K.txt

Processing: NVDA
Downloading NVDA 10-K for 2022 (filed on 2022-03-18)...
Saved to sec_filings/NVDA_2022_10K.txt
Downloading NVDA 10-K for 2023 (filed on 2023-02-24)...
Saved to sec_filings/NVDA_2023_10K.txt
Downloading NVDA 10-K for 2024 (filed on 2024-02-21)...
Saved to sec_filings/NVDA_2024_10K.txt
```

Chunking , Embedding and Storing

```
from typing import List, Optional


class VectorDatabaseIngestion:
  def __init__(self,
               data_directory: str = "sec_filings/",
               qdrant_url: str = ":memory:",
               collection_name: str = "sec_filings_collection",
               embedding_model: str = "sentence-transformers/all-MiniLM-L6-v2",
               chunk_size: int = 1000,
               chunk_overlap: int = 200):
    self.data_directory = Path(data_directory)
    self.qdrant_url = qdrant_url
    self.collection_name = collection_name
    self.chunk_size = chunk_size
    self.chunk_overlap = chunk_overlap

    try:
      self.embedding = SentenceTransformerEmbeddings(model_name=embedding_model)
    except Exception as e:
      print(f"Error loading embedding model: {e}")

  def load_documents(self) -> List[Document]:
    """Loads documents from the data directory."""
    try:
      loader = DirectoryLoader(str(self.data_directory),
                               glob="*.txt",
                               loader_cls=TextLoader,
                               show_progress=True)

      documents = loader.load()
      return documents
    except:
      print(f"Error loading documents from {self.data_directory}")
      return []

  def split_documents(self,documents : List[Document]) -> List[Document]:
    """Splits documents into chunks."""

    text_splitter = CharacterTextSplitter(separator="\n\n", chunk_size=self.chunk
```

```
    chunks = text_splitter.split_documents(documents)
    print(len(chunks))
    return chunks

  def ingest_documents(self):
    documents = self.load_documents()
    if not documents:
      return

    chunks = self.split_documents(documents)
    if not chunks:
      return

    try:
      qdrant = Qdrant.from_documents(
        chunks,
        self.embedding,
        location=self.qdrant_url,
        collection_name=self.collection_name
      )
      self.qdrant_db = qdrant
      return qdrant
    except Exception as e:
      print(f"Error ingesting documents: {e}")


  def search_similar_chunks(self,query: str,top_k: int = 5):
    if not self.qdrant_db:
      print("Qdrant database not initialized. Please call ingest_document first."
      return []

    try:
      results = self.qdrant_db.similarity_search(query, k=top_k)
      return results
    except Exception as e:
      print(f"Error searching similar chunks: {e}")
      return []




ingester = VectorDatabaseIngestion(
      data_directory=SAVE_DIR,
      qdrant_url=":memory:",
      collection_name="sec_filings_vector_db",
      embedding_model="sentence-transformers/all-MiniLM-L6-v2"
)
qdrant_db = ingester.ingest_documents()
```

Show hidden output

Testing Sample Query and Functions

```
query = "What was NVIDIA operating margin in 2023?"
docs = qdrant_db.similarity_search(query)
print("\nSearch Results:\n")
for doc in docs:
    print(doc.page_content[:1000] + "...")
    print('\n'+100*'~'+'\n')
```

```
Search Results:

Year Ended
                                        January 29,                January 3
                                               2023                       20
Revenue:
United States                            $8,292                     $4,349
Taiwan                                    6,986                      8,544
China (including Hong Kong)               5,785                      7,111
Other countries                           5,911                      6,910
Total revenue                           $26,974                    $26,914
No customer represented 10% or more of total revenue for fiscal years 2023,
NVIDIA CORPORATION AND SUBSIDIARIES
NOTES TO THE C...

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Year Ended
                                         Jan 28, 2024                  Jan 29,
Revenue:
Data Center                            $47,525                       $15,005
Gaming                                  10,447                         9,067
Professional Visualization               1,553                         1,544
Automotive                               1,091          903
OEM and Other                              306          455
Total revenue                          $60,922                       $26,974
NVIDIA Corporation and Subsidiaries
Notes to the Consolidated Financial Statements
(Continued...

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

January 30,                   January 31,
                                                          2022

Balance at beginning of period                        $451
Deferred revenue added during the period               821
Addition due to business combinations                    8
Revenue recognized during the period                 (778)
Balance at end of period                              $502
Revenue related to remaining performance obligations represents the contract

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Year Ended
                                           Jan 28, 2024

Revenue                                  $60,922
Gross margin                                72.7      %   56.9
Operating expenses                       $11,329
```

```
Operating income                        $32,972
Net income                              $29,760
Net income per diluted share             $11.93

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
context = "\n\n".join([doc.page_content for doc in docs])
```

```
context
```

```
'Year Ended\n                                                January 29,
January 30,                January 31,\n
2023                       2022                          2021\nRevenue:
(In millions)\nUnited States                             $8,292
$4,349                     $3,214      \nTaiwan
6,986                      8,544                         4,531      \nChina (inc
luding Hong Kong)             5,785                         7,111
3,886      \nOther countries                            5,911
```

```
Companies = {"GOOGLE":"GOOGL","MICROSOFT":"MSFT","NVIDIA":"NVDA"}
Years = [2022,2023,2024]
```

```
prompt = PromptTemplate(
    input_variable = ["context","query","companies","years"],
    template = """ You are a helpful assistant. Use ONLY the following pieces of
    The context might have some data in tabular format so parse and understand it
    For complex question Like comparsion between companies for revenue/total reve
    For Simple question You can directly answer the question based on the context
    If spending/operating margin/gross margin/profit/operating profit/total reven
    If the Question ask for revenue growth/growth also provide the percentage gro
    If you don't know the answer, just say that you don't know, don't try to make

    Data which we have :

    {context}

    {companies}
    {years}

    Question: {query}
    """
)
```

```
LMC = LLMChain(llm=llm,prompt=prompt)
final_a = LMC.run({"context":context,"query":query,"companies":Companies,"years":
print(final_a)
```

```
To determine NVIDIA's operating margin in 2023, I will use the provided data.

Operating Income in 2023: $4,224 million
Revenue in 2023: $26,974 million

Operating Margin = (Operating Income / Revenue) * 100
```

```
Operating Margin = ($4,224 / $26,974) * 100 = 15.66%

Answer: NVIDIA's operating margin in 2023 was 15.66%.
```

Checking Multi Query Reriever

```
from langchain.retrievers.multi_query import MultiQueryRetriever
```

```
multiquery_retriever = MultiQueryRetriever.from_llm(
    retriever=qdrant_db.as_retriever(search_kwargs={"k": 5}),
    llm=llm
)
```

```
multiquery_results= multiquery_retriever.invoke(query)
```

```
multiquery_results
```

Show hidden output

```
context = "\n\n".join([doc.page_content for doc in multiquery_results])
```

```
LMC = LLMChain(llm=llm,prompt=prompt)
final_a = LMC.run({"context":context,"query":query})
print(final_a)
```

```
I am sorry, but the context provided does not contain sufficient information
```

Checking by Decomposing Query

```
Companies = {"GOOGLE":"GOOGL","MICROSOFT":"MSFT","NVIDIA":"NVDA"}
Years = [2022,2023,2024]
```

```
query = "Compare cloud revenue growth rates across all three companies from 2022
```

```
decompose_prompt = PromptTemplate(
    input_variables=["companies","years","query"],
    template = """You are a Helpfull assistant. Use ONLY the following pieces of
    I want to decompose this Question/query into multiple simpler and logically o
    where each Question/query is decomposed on the basis of the question type, co
    1)Simple Direct Query : "What was Microsoft’s total revenue in 2023?" - For t
    2)Comparative Query : “How did NVIDIA’s data center revenue grow from 2022 to
    - it should be broken into Find NVIDIA data center revenue 2022,Find NVIDIA d
    3)Cross-Company Analysis : “Which company had the highest operating margin in
    - Retrieve MSFT operating margin 2023 , Retrieve GOOGL operating margin 2023,
```

```
    Provide ONLY sub-queries in the above format and place each sub-query into th
    If you don't know the answer, just say that you don't know, don't try to make

    {companies}
    {years}
    Question :{query}

    """
)
```

```
DLMC = LLMChain(llm=llm,prompt=decompose_prompt)
de_a = DLMC.run({"companies":Companies,"years":Years,"query":query})
print(de_a)
```

```
Retrieve MSFT cloud revenue 2022
Retrieve MSFT cloud revenue 2023
Calculate MSFT cloud revenue growth from 2022 to 2023
Retrieve GOOGL cloud revenue 2022
Retrieve GOOGL cloud revenue 2023
Calculate GOOGL cloud revenue growth from 2022 to 2023
Retrieve NVDA cloud revenue 2022
Retrieve NVDA cloud revenue 2023
Calculate NVDA cloud revenue growth from 2022 to 2023
Compare cloud revenue growth rates across MSFT, GOOGL, and NVDA
```

```
sub_queries = de_a.strip().split('\n')
sub_queries
```

```
['Retrieve MSFT cloud revenue 2022',
 'Retrieve MSFT cloud revenue 2023',
 'Calculate MSFT cloud revenue growth from 2022 to 2023',
 'Retrieve GOOGL cloud revenue 2022',
 'Retrieve GOOGL cloud revenue 2023',
 'Calculate GOOGL cloud revenue growth from 2022 to 2023',
 'Retrieve NVDA cloud revenue 2022',
 'Retrieve NVDA cloud revenue 2023',
 'Calculate NVDA cloud revenue growth from 2022 to 2023',
 'Compare cloud revenue growth rates across MSFT, GOOGL, and NVDA']
```

```
# Multi step retrieval
all_docs = []
for sub_query in sub_queries:
  sub_docs = multiquery_retriever.invoke(sub_query)
  all_docs.extend(sub_docs)
```

```
all_docs
```

Show hidden output

```
context = "\n\n".join([doc.page_content for doc in all_docs])
```

```
context
```

```
'(In millions)
\nYear Ended June 30,                  2023                2022
2021             \nServer products and cloud services             $79,970
$67,350            $52,589\nOffice products and cloud services
48,728             44,862              39,872\nWindows
21,507             24,732              22,488\nGaming
15,466             16,230              15,370\nLinkedIn
15,145             13,816              10,289\nSearch and news advertising
```

```
query = "Compare cloud revenue growth rates across all three companies from 2022


LMC = LLMChain(llm=llm,prompt=prompt)
final_a = LMC.run({"context":context,"query":query,"companies":Companies,"years":
print(final_a)
```

```
Okay, let's break this down to compare cloud revenue growth rates.

**1. Microsoft Cloud Revenue Growth:**

*   Microsoft Cloud revenue in 2022: Not directly available, but Microsoft Cl
*   Microsoft Cloud revenue in 2023: Microsoft Cloud revenue increased 23% to
*   So, we can calculate the cloud revenue for 2022 using the 2023 growth numl
*   Microsoft Cloud revenue growth rate from 2022 to 2023 = (111.6-91.47)/91.

**2. Google Cloud Revenue Growth:**

*   Google Cloud revenue in 2022: $26,280 million
*   Google Cloud revenue in 2023: $33,088 million
*   Google Cloud revenue growth rate from 2022 to 2023 = (33088-26280)/26280 =

**Comparison:**

*   Microsoft Cloud revenue growth rate (2022 to 2023): 21.9%
*   Google Cloud revenue growth rate (2022 to 2023): 25.9%

**Answer:** Google Cloud revenue grew faster (25.9%) than Microsoft Cloud rev
```

Agent

```
Companies = {"GOOGLE":"GOOGL","MICROSOFT":"MSFT","NVIDIA":"NVDA"}
Years = [2022,2023,2024]


from langchain.retrievers.multi_query import MultiQueryRetriever


class Agent:
  def __init__(self,qdrant_db):
    self.qdrant_db = qdrant_db

  def decompose_query(self,query):
    decompose_prompt = PromptTemplate(
```

```
    input_variables=["companies","years","query"],
    template = """You are a Helpfull assistant. Use ONLY the following pieces of
    I want to decompose this Question/query into multiple simpler and logically o
    where each Question/query is decomposed on the basis of the question type, co
    1)Simple Direct Query : "What was Microsoft’s total revenue in 2023?" - For t
    2)Comparative Query : “How did NVIDIA’s data center revenue grow from 2022 to
    - it should be broken into Find NVIDIA data center revenue 2022,Find NVIDIA d
    3)Cross-Company Analysis : “Which company had the highest operating margin in
    - Retrieve MSFT operating margin 2023 , Retrieve GOOGL operating margin 2023,

    Provide ONLY sub-queries in the above format and place each sub-query into th
    If you don't know the answer, just say that you don't know, don't try to make

    {companies}
    {years}
    Question :{query}

    """)

    DLMC = LLMChain(llm=llm,prompt=decompose_prompt)
    de_a = DLMC.run({"companies":Companies,"years":Years,"query":query})

    sub_queries = de_a.strip().split('\n')
    if not sub_queries:
      sub_queries = [query]
    print(sub_queries)
    return sub_queries

  def multistep_retrieval(self,sub_queries):
    multiquery_retriever = MultiQueryRetriever.from_llm(
    retriever=qdrant_db.as_retriever(search_kwargs={"k": 5}),
    llm=llm)

    all_docs = []
    for sub_query in sub_queries:
      sub_docs = multiquery_retriever.invoke(sub_query)
      all_docs.extend(sub_docs)

    context = "\n\n".join([doc.page_content for doc in all_docs])
    return context

  def synth_result(self,context,query):
    prompt = PromptTemplate(
    input_variable = ["context","query","companies","years"],
    template = """ You are a helpful assistant. Use ONLY the following pieces of
    The context might have some data in tabular format so parse and understand it
    For complex question Like comparsion between companies for revenue/total reve
    For Simple question You can directly answer the question based on the context
    If spending/operating margin/gross margin/profit/operating profit/total reven
    If the Question ask for revenue growth/growth also provide the percentage gro
    If you don't know the answer, just say that you don't know, don't try to make

    Data which we have :

    {context}
```

```
    {companies}
    {years}

    Question: {query}
    """)

    LMC = LLMChain(llm=llm,prompt=prompt)
    final_a = LMC.run({"context":context,"query":query,"companies":Companies,"yea
    return final_a

  def pipeline(self,query):
    sub_queries = self.decompose_query(query)
    context = self.multistep_retrieval(sub_queries)
    result = self.synth_result(context,query)
    return result
```

```
agent = Agent(qdrant_db=qdrant_db)
```

Testing On Sample Queries

```
query = "How did NVIDIA’s data center revenue grow from 2022 to 2023?"
result = agent.pipeline(query)

print(str(result))
```

```
['Find NVIDIA data center revenue 2022', 'Find NVIDIA data center revenue 202
NVIDIA's data center revenue increased from $10,613 million in 2022 to $15,00
```

```
query = "What was NVIDIA's total revenue in fiscal year 2024?"
result = agent.pipeline(query)

print(str(result))
```

```
['What was NVIDIA’s total revenue in 2024?']
NVIDIA's total revenue in fiscal year 2024 was $60,922 million.
```

```
query = "What percentage of Google's 2023 revenue came from advertising?"
result = agent.pipeline(query)

print(str(result))
```

```
['Find Google total revenue 2023', 'Find Google advertising revenue 2023', 'Ca
Google's 2023 advertising revenue was $237,855 million, and Google's total rev
To find the percentage of Google's 2023 revenue that came from advertising, we
Percentage = (Advertising Revenue / Total Revenue) * 100
Percentage = (237,855 / 307,394) * 100 = 77.38%
So, approximately 77.38% of Google's 2023 revenue came from advertising.
```

```
query = "How much did Microsoft's cloud revenue grow from 2022 to 2023?"
result = agent.pipeline(query)

print(str(result))
```

```
['Find Microsoft cloud revenue 2022', 'Find Microsoft cloud revenue 2023', 'Ca
Microsoft's cloud revenue grew from $91.4 billion in fiscal year 2022 to $111
```

```
query = "Which of the three companies had the highest gross margin in 2023?"
result = agent.pipeline(query)

print(str(result))
```

```
['Retrieve MSFT gross margin 2023', 'Retrieve GOOGL gross margin 2023', 'Retr
I will calculate the gross margin for each of the three companies for the yea

    *   **Microsoft:** Gross margin for 2023 is $171,008 million.
    *   **Alphabet (Google):**
        To determine Google's gross margin, we need to calculate it using the prov
        Gross Margin = Total Revenue - Cost of Revenue
        For 2023:
        Total Revenue = $307,394 million
        Cost of Revenue = $133,332 million
        Gross Margin = $307,394 - $133,332 = $174,062 million
    *   **NVIDIA:**
        To determine NVIDIA's gross margin, we need to calculate it using the prov
        Gross Margin = Revenue \* Gross Margin Percentage
        For 2023:
        Total Revenue = $26,974 million
        Gross Margin Percentage = 56.9%
        Gross Margin = $26,974 \* 0.569 = $15,349.21 million

    Comparing the gross margins:

    *   Microsoft: $171,008 million
    *   Alphabet (Google): $174,062 million
    *   NVIDIA: $15,349.21 million

    Therefore, **Alphabet (Google)** had the highest gross margin in 2023.
```

```
query = "Which company had the highest operating margin in 2023?"
result = agent.pipeline(query)

print(str(result))
```

```
['Retrieve MSFT operating margin 2023', 'Retrieve GOOGL operating margin 2023
First. let's find the operating margin for each company in 2023.

    For Microsoft (MSFT):
    Revenue in 2023: $211,915 million
    Operating income in 2023: $88,523 million
    Operating margin in 2023: ($88,523 / $211,915) * 100 = 41.77%

    For NVIDIA (NVDA):
```

```
Revenue in 2023: $26,974 million
Operating income in 2023: $4,224 million
Operating margin in 2023: ($4,224 / $26,974) * 100 = 15.66%

For Alphabet (GOOGLE):
Total revenues in 2023: $307,394 million
Total income from operations in 2023: $84,293 million
Operating margin in 2023: ($84,293 / $307,394) * 100 = 27.42%

Therefore, Microsoft had the highest operating margin in 2023.
```

```
query = "Compare the R&D spending as a percentage of revenue across all three com
result = agent.pipeline(query)

print(str(result))
```

```
['Retrieve MSFT R&D spending as a percentage of revenue 2023', 'Retrieve GOOGI
Okay, I will compare the R&D spending as a percentage of revenue for Google, M

*   **Google:** In 2023, Research and development expenses as a percentage of
*   **Microsoft:** In 2023, Research and development expenses as a percentage
*   **NVIDIA:** R&D spending as a percentage of revenue in 2023 was 27.2%.
```

```
query = "What are the main AI risks mentioned by each company and how do they dif
result = agent.pipeline(query)

print(str(result))
```

```
Okay, I will analyze the provided texts and extract the main AI risks mentione

**Alphabet Inc.:**

Alphabet is very concerned about AI risks and calls out the following:
*   Harmful content
*   Inaccuracies
*   Discrimination
*   Intellectual property infringement or misappropriation
*   Defamation
*   Data privacy
*   Cybersecurity
*   Ethical issues
*   Broad effects on society
*   Unintended consequences, uses, or customization of AI tools and systems
*   Negatively affecting human rights, privacy, employment, or other social co

**Microsoft:**

Microsoft focuses on:
*   AI systems being used in ways that are unintended or inappropriate.
*   Fraudulent or abusive activities through cloud-based services.
*   Unauthorized account access
*   Payment fraud
*   Terms of service violations including cryptocurrency mining or launching o

**Differences:**
```

```
*   **Breadth of Concerns:** Alphabet's risk list is more extensive and covers
*   **Societal Impact:** Alphabet explicitly mentions the broad effects of AI
*   **Specific Risks:** Alphabet details risks like discrimination, defamatio
*   **Risk Management:** Alphabet emphasizes its investment in developing, te

In summary, both companies acknowledge AI-related risks, but Alphabet present
```

```
# Formating and Meta Data can be done based on the requirement
```