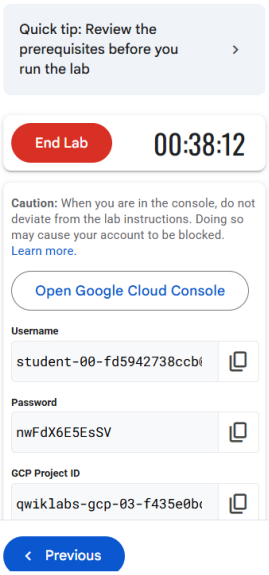


Use function calling and grounding with Google Search to create a research tool

Lab · 1 hour · No cost · Intermediate

This lab may incorporate AI tools to support your learning.

100/100

Previous Next

GENAI081

Overview

When a company "goes public" (meaning its stock can be sold publicly on the U.S. stock market) it is required to file annual and quarterly reports with the U.S. Securities and Exchange Commission (the SEC). In order for everyone to have access to the same information to use for making trading decisions, the SEC then makes these documents available publicly through a service called EDGAR, which allows for manual search via the SEC's website or automated access via an API.

conditions compared to a competitor, or how a company's managers describe changes in their strategy.

Assisting with rapid analysis has been identified as a generative AI market use case worth partners' attention.

Note: When running this lab, you may encounter Gemini 2.0 Flash quota errors (which will display as 429 Resource exhausted errors) due to quotas imposed on Qwiklabs project. If you do, stop sending queries for a few minutes, and then continue.

Objective

In this lab, you will investigate how to do the following, which is applicable to research

- Download filings from the SEC's EDGAR API.
- Consider when you might want to take a non-RAG approach to document chunk retrieval for structured documents.
- Use grounding with Google Search to retrieve up-to-date information or information requiring the search understanding offered by Google Search.
- Build a system that understands versions of a document to compare how sections have changed across multiple versions.
- Build a system that can compare certain sections of documents written on different subjects (different companies, in this case).
- Handle sequential or parallel function calls from Gemini.

Setup and requirements

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

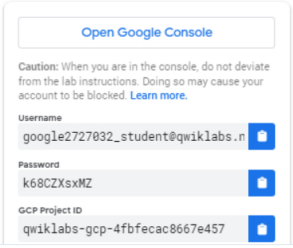
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not

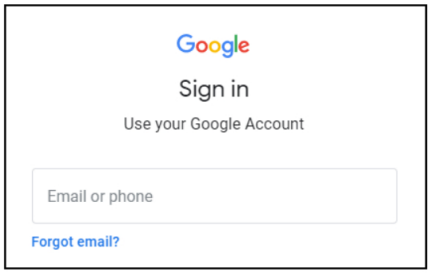
Note: If you are using a Pixelbook, open an Incognito window to run this lab.

How to start your lab and sign in to the Google Cloud Console

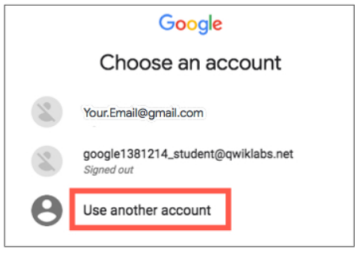
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.



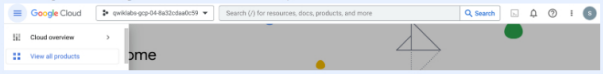
3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



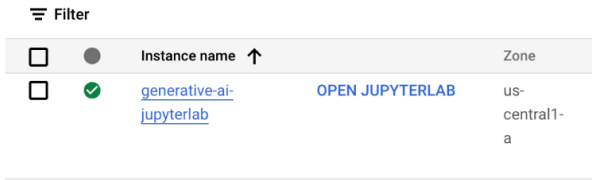
Task 1. Prepare the environment in Vertex AI Workbench

In this section, you will be setting up the environment for the **SEC filings analysis**. Additionally, you will be importing essential libraries and initializing the Vertex AI client within the notebook.

1. In your Google Cloud console, navigate to **Vertex AI**. In the top search bar of the Google Cloud console, enter **Vertex AI**, and click on the first result.

2. Navigate to the **Vertex AI > Dashboard** page and click on **Enable All Recommended APIs**.

The JupyterLab will run in a new tab.



4. In the new tab, under Launcher in the Notebook section, select **Python 3 (ipykernel)**.

5. In the **Explorer** on the left, right-click on the newly created file, select **Rename**, and change the name to `sec_filings_analysis.ipynb`.

```
import os
import re
import requests
from datetime import datetime

from bs4 import BeautifulSoup, NavigableString

import vertexai
from vertexai.generative_models import (
    GenerativeModel,
    GenerationConfig,
    Tool,
    FunctionDeclaration,
    Part)
```

Either click the play button ▶ at the top or enter **SHIFT+ENTER** on your keyboard to execute the cell.

```
project_id = !gcloud config get project
project_id = project_id[0]

vertexai.init(project=project_id, location="us-central1")
```

Click **Check my progress** to verify the objective.

Create and rename the Python Notebook

Check my progress

Assessment Completed!

Task 2. Download SEC filings and parse text to analyze

In this section, you will tackle the limitations of processing massive SEC filings with Gemini. You will explore utilizing the document structure to efficiently extract specific sections and confirm their suitability for analysis within Gemini's token limit.

1. Run the following code in your notebook and update the values of `PARTNER_COMPANY` and `PARTNER_WEBSITE` to prepare a header to call **EDGAR** on behalf of your company.

```
# IMPORTANT! You must set these variables with REAL VALUES
for the correct headers to be in place to access the API
PARTNER_COMPANY = "YOUR COMPANY'S NAME"
PARTNER_WEBSITE = "https://your-companys-website"

{PARTNER_WEBSITE}"}
```

2. Use the following code to name a local directory for storing filings and defining the following two functions:

   - **download_single_filing** - It downloads a single SEC filing identified by a company's Central Index Key (CIK), the form type(an annual 10-K or quarterly 10-Q) and the date of filing.
   - **download_range_of_filings** - It queries all the filings of a company to find filings within a date range of a particular form type and download them.

```
BASE_DIR = "filings" # Directory for storing raw filings

def download_single_filing(url, cik, form, date,
file_extension):
    """
    .

    Args:
        cik (str): Central Index Key (CIK) for the company.
        form (str): The type of SEC filing (e.g., 10-K, 10-
Q).
        date (str): The filing date in YYYY-MM-DD format.
        file_extension (str): File extension for saved file
(usually 'txt' or 'zip').
    """

    # Define request headers to simulate a browser visit
    response = requests.get(url, headers=headers)

    if response.status_code == 200:

        # Make the directory accord
        dir_name = f"{BASE_DIR}/{cik}"
        os.makedirs(dir_name, exist_ok=True)
        file_path = f"{dir_name}/{form}_{date}
```

```
{file_extension}"

            file.write(response.content)

        print(f"Downloaded {form} for CIK {cik} on {date}
to {file_path}")

        return file_path

    else:
        print(f"Failed to download {form} for CIK {cik} on
{date}. Status code: {response.status_code}")

        return None

def download_range_of_filings(cik,
starting_year_and_quarter,
                              ending_year_and_quarter,
include_10q = False):
    """
    Download filings from EDGAR for a given CIK and clean
them up

        cik (str): Central Index Key (CIK)
        starting_year_and_quarter (str): Specified in the
format "2023 Q1"
        ending_year_and_quarter (str): Specified in the
format "2024 Q4"
        include_10q (bool): Whether to include 10-Qs in
addition to 10-Ks
    """

    url = f"https://data.sec.gov/submissions/CIK{cik}.json"
    headers = {'User-Agent': 'Google Partner Learning
Services Demo +https://partners.cloud.google.com/learn'}
    response = requests.get(url, headers=headers)

    forms_to_download = {"10-K", "10-Q"} if include_10q
else {"10-K"}

    start_year, start_quarter =
starting_year_and_quarter.split()
    end_year, end_quarter = ending_year_and_quarter.split()

        data = response.json()

        filing_paths = []
        for filing_date, form, accession_number in
zip(data['filings']['recent']['filingDate'],
data['filings']['recent']['form'], data['filings']
['recent']['accessionNumber']):
            if (form in forms_to_download) and
(int(start_year) <= datetime.strptime(filing_date, '%Y-%m-
%d').year <= int(end_year)):
                file_path = download_single_filing(

f"https://www.sec.gov/Archives/edgar/data/{cik}/{accession_nu
', '')}/{accession_number}.txt",
                    cik, form, filing_date, 'htm'
                )
                filing_paths.append (file_path)
        return filing_paths
    else:
        print("Error from call.")
```

3. Next, test the functions using the **CIK of Alphabet**, Google's parent company, which is `0001652044`.

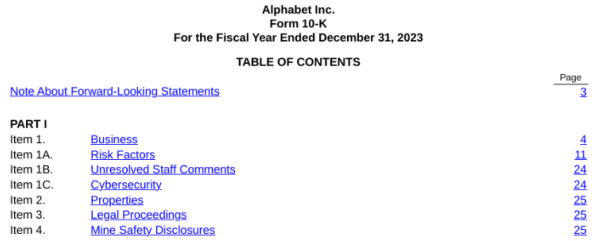
```
alphabet_cik = "0001652044"

download_range_of_filings(cik=alphabet_cik,
                starting_year_and_quarter="2024 Q1",
                ending_year_and_quarter="2024 Q4")
```

Output:

```
Downloaded 10-K for CIK 0001652044 on 2024-01-31 to
filings/0001652044/10-K_2024-01-31.htm
'filings/0001652044/10-K_2024-01-31.htm'
```

Open in New Browser Tab to preview the document. Notice that the table of contents of these documents are highly structured and all the **10-K filings** will follow the same structure.

Output:



Alphabet Inc.
Form 10-K
For the Fiscal Year Ended December 31, 2023

TABLE OF CONTENTS

		Page
Note About Forward-Looking Statements		3
PART I		
Item 1.	Business	4
Item 1A.	Risk Factors	11
Item 1B.	Unresolved Staff Comments	24
Item 1C.	Cybersecurity	24
Item 2.	Properties	25
Item 3.	Legal Proceedings	25
Item 4.	Mine Safety Disclosures	25

5. Instantiate a **GenerativeModel** using Gemini Pro version `gemini-2.0-flash`

```
model = GenerativeModel("gemini-2.0-flash",

generation_config=GenerationConfig(temperature=0),
                        )
```

6. To determine if you can send the entire document to **Gemini** to analyze at once, use your GenerativeModel's `count_tokens` method to see the number of tokens in the raw file.

```
downloaded_path = "filings/0001652044/10-K_2024-01-31.htm"

with open(downloaded_path, 'r') as f:
    filing_text = f.read()

response = model.count_tokens(filing_text)
print(response)
```

```
total_tokens: 5798629
total_billable_characters: 13029439
prompt_tokens_details {
  modality: TEXT
  token_count: 5798629
}
```

7. With a token count of **5,798,629**, you can see that even with Gemini 2.0 Flash's large token window of 1,048,576 tokens, these documents are too long to read in a single pass.

8. You could implement a **Retrieval-Augmented Generation (RAG)** framework to query small chunks of these documents, but here you are looking for a broader understanding of sections as a whole rather than smaller chunks consisting of a few facts in the document. Here, you don't mind passing a large number of tokens to **Gemini**, as this will be an internal tool used by a relatively small number of

9. SEC filings are required to adhere to a strict structure with named sections. You can use this standard structure of the documents to read the text between one section header and the next.

10. Run the following code in your notebook to define a list of the `sections(items)` and `functions` to help retrieve all of the text from one item header until the next item header.

```
items = ['Business',
        'Risk Factors',
        'Unresolved Staff Comments',
        'Properties', 'Legal Proceedings',
        'Mine Safety Disclosures',
        'Market for Registrant's Common Equity, Related
Stockholder Matters and Issuer Purchases of Equity
Securities',
        'Management's Discussion and Analysis of Financial
Condition and Results of Operations',
        'Financial Statements and Supplementary Data',
```

```
        'Changes in and Disagreements with Accountants on
Accounting and Financial Disclosure',
        'Controls and Procedures',
        'Other Information',
        'Disclosure Regarding Foreign Jurisdictions that
Prevent Inspections',
        'Directors, Executive Officers, and Corporate
Governance',
        'Executive Compensation',
        'Security Ownership of Certain Beneficial Owners
and Management and Related Stockholder Matters',
        'Certain Relationships and Related Transactions,
and Director Independence',
        'Principal Accountant Fees and Services',
        'Exhibit and Financial Statement Schedules',
        'Form 10-K Summary']

def find_div_id(soup, item_name: str):
```

```
    else:
        print(f"Couldn't find a matching tag for:
{item_name}")
        return None

def get_text_between(soup, cur_name, end_name):
    cur = soup.find('div', id=find_div_id(soup,
cur_name)).next_sibling
    end = soup.find('div', id=find_div_id(soup, end_name))
    while cur and cur != end:
        if isinstance(cur, NavigableString):
            text = cur.strip()
            if len(text):
                yield text
        cur = cur.next_element

def get_items_from_filings(item_names, filing_paths):

    print("Items of interest: " + ", ".join(item_names) +
```

```
    for path in filing_paths:
        with open(path, 'r', encoding='utf-8') as file:
            content = file.read()

        soup = BeautifulSoup(content, 'html.parser')

        for item in item_names:
            item_index = items.index(item)
            item_index = item_index if item_index <
len(items) - 1 else 0
            item_output = ' '.join(text for text in
get_text_between(soup, item, items[item_index + 1]))
            item_strings[item] += f"From
{os.path.basename(path)}" + "\n" + item_output + "\n"

    return "\n\n".join(item_strings.values())
```

11. Next, test the above functions by running the following command and save your

```
item_names = ["Management's Discussion and Analysis of
Financial Condition and Results of Operations"]

report = get_items_from_filings(item_names,
[downloaded_path])

# Print the first 2,000 characters as an example.
print(report[0:2000] + "...")
```

Output:

```
Items of interest: Management's Discussion and Analysis of
Financial Condition and Results of Operations

<management analysis="" and="" condition="" discussion=""
financial="" of="" operations="" results="">
```

```
AND ANALYSIS OF FINANCIAL CONDITION AND RESULTS OF OPERATIONS
Please read the following discussion and analysis of our financial
condition and results of operations together with "Note about
Forward-Looking Statements," Part I, Item 1 "Business," Part I,
```

```
Item 1A "Risk Factors," and our consolidated financial statements
and related notes included under Item 8 of this Annual Report on
Form 10-K. The following section generally discusses 2023 results
compared to 2022 results.
</management>
```

12. Count the number of tokens in this section to see if it is under **Gemini's token window** of 1,048,576 tokens.

```
response = model.count_tokens(report)
print(response)
```

Output:

```
total_tokens: 12688
total_billable_characters: 50620
prompt_tokens_details {
  modality: TEXT
  token_count: 12688
}
```

Click **Check my progress** to verify the objective.

Download the htm file and test the functions

Check my progress

Assessment Completed!

Task 3. Let Gemini look up companies' CIKs via Google Search

You were able to download **Alphabet's** annual report because you were provided its `Central Index Key (CIK)` number. Gemini already knows some CIKs for large public companies like Alphabet, but for some smaller companies, it may hallucinate and invent inaccurate CIKs. You can instead look up correct numbers by having Gemini use everyone's favorite well-known, public, up-to-date source of information: Google Search.

1. Run the following command to see **Gemini** provide a `CIK` from its internal knowledge. The **CIK of Summit Therapeutics** is actually `0001599298`, but without the ability to look it up, Gemini provides some alternative numbers.

```
model.generate_content("What is Summit Therapeutics' CIK
```

Output:

```
"Summit Therapeutics' CIK is **0001522020**.
```

2. Use Grounding with Google Search to provide Gemini with a tool to conduct Google searches.

```
from google import genai
from google.genai.types import Tool, GenerateContentConfig,
GoogleSearch, HttpOptions
from IPython.display import display, HTML
import os

project_id = os.environ['GOOGLE_CLOUD_PROJECT'] =
'qwiklabs-gcp-03-f435e0bdc8a5'
```

```
# Initialize the client, explicitly passing project and
location for Vertex AI
client = genai.Client(project=project_id,
location=location,
```

```
http_options=HttpOptions(api_version="v1"))
model_id = "gemini-2.0-flash"

# Configure Google Search as a tool for grounding
search_tool = Tool(
    google_search=GoogleSearch()
)

# Helper function to lookup a company's SEC CIK
def lookup_cik(company_name: str) -> str:
    prompt = f"""
Find the Securities and Exchange Commission's Central Index
Key (CIK)
for {company_name}.
Return only the 10 digit integer CIK including leading
```

```
    # Generate content with grounding via Google Search
    response = client.models.generate_content(
        model=model_id,
        contents=prompt,
        config=GenerateContentConfig(
            tools=[search_tool],
            response_modalities=["TEXT"],
        )
    )

    # Concatenate all text parts of the response
    cik = "".join(part.text for part in
response.candidates[0].content.parts).strip()
    print(f"Lookup for CIK of {company_name} resulted in:
{cik}\n")

    # Display the grounding metadata (search suggestions
and sources)
    if hasattr(response.candidates[0],
'grounding_metadata'):
```

```
                `
                response.candidates[0]

.grounding_metadata.search_entry_point.rendered_content
            )
        )

    return cik
```

Note: When you use Grounding with Google Search, you are required to display the corresponding Google Search Suggestions which help you understand what Google search was conducted and allow you to investigate the results. The **lookup_cik** function below includes a **display** block that renders the provided HTML in this notebook.

3. Test the function and confirm you see the Google Search Suggestion results.

Output:

```
Lookup for CIK of Summit Therapeutics resulted in: 0001599298
```


```
'0001599298'
```

4. Create a `FunctionDeclaration` that will help Gemini know about function to lookup a **CIK**.

```
lookup_cik_fd = FunctionDeclaration(
    name="lookup_cik",
    description="Look up a company's CIK used for its SEC
filings.",
    parameters={
        "type": "object",
```

```
                "type": "string",
                "description": "The name of the company to
look up."
            },
        },
        "required": [
            "company_name"
        ]
```

```
    },
)
```

5. Save your notebook.

Note: The **search_tool** loaded above cannot be combined with other Tools when passed to Gemini, so in order to create a Tool that combines it and other functions, you can create a dedicated model instance and invoke that via another function as you are doing here.

Create a FunctionDeclaration to lookup a CIK

Check my progress

Assessment Completed!

Task 4. Empower Gemini to retrieve document sections from relevant years

In this section, you will equip **Gemini** to analyze specific sections of public company

comprehensive analysis.

1. In your notebook, paste and run the following `FunctionDeclaration` that you can use to let Gemini know about the function you used earlier to download SEC filings and review its parameters. In particular, pay attention to how the `items_of_interest` section allows Gemini to provide an array of strings, but consisting only of strings matching the enum values you provide (the names of the various sections in the reports).

```
retrieve_filings_fd = FunctionDeclaration(
    name="retrieve_filings",
    description="Retrieve filings from the SEC EDGAR API
for a company within a date range.",
    parameters={
        "type": "object",
        "properties": {
            "cik": {
                "type": "string"
            },
            "starting_year_and_quarter": {
                "type": "string",
                "description": "The first report quarter
year and quarter in the format: 2024 Q1"
            },
            "ending_year_and_quarter": {
                "type": "string",
                "description": "The year and quarter in the
format: 2024 Q1"
            },
            "include_quarterly_reports": {
                "type": "boolean",
                "description": "Whether to include 10-Q
quarterly filings in addition to annual 10-K filings"
            },
            "items_of_interest": {
                "description": "An array of one or more
section (called items) of interest from 10-K or 10-Q
filings"
                "type": "string",
                "enum": [
                    "Business",
                    "Risk Factors",
                    "Unresolved Staff Comments",
                    "Properties",
                    "Legal Proceedings",
                    "Mine Safety Disclosures",
                    "Market for Registrant's Common Equity,
```

```
Related Stockholder Matters and Issuer Purchases of Equity
Securities",
                    "Management's Discussion and Analysis
of Financial Condition and Results of Operations",
                    "Quantitative and Qualitative
Disclosures About Market Risk",
                    "Financial Statements and Supplementary
Data",
                    "Changes in and Disagreements with
Accountants on Accounting and Financial Disclosure",
                    "Controls and Procedures"
```

```
Jurisdictions that Prevent Inspections",
                    "Directors, Executive Officers and
Corporate Governance",
                    "Executive Compensation",
                    "Security Ownership of Certain
Beneficial Owners and Management and Related Stockholder
Matters",
                    "Certain Relationships and Related
Transactions, and Director Independence",
                    "Principal Accountant Fees and
Services",
                    "Exhibit and Financial Statement
Schedules",
                    "Form 10-K Summary"
                ]
                }
        }
        },
        "required": [
            "cik"
```

```
            "items_of_interest"
        ]
    },
)
```

2. Run the code snippet below to create a **Tool** that combines both of the `FunctionDeclarations` above.

```
sec_tool = Tool(function_declarations=[retrieve_filings_fd,
                                       lookup_cik_fd])
```

3. Create a **system_instruction** and instantiate a new model that will follow the instructions to create analyses using the **SEC filings**. In your instructions, you will provide Gemini the current date so that it can calculate relevant dates for queries using relative terminology.

```
    model="gemini-2.0-flash",
    contents="Find the SEC CIK for Acme Corp. Return only
the 10-digit CIK.",
    config=GenerateContentConfig(tools=[search_tool],
response_modalities=["TEXT"])
)
cik = "".join(p.text for p in
response1.candidates[0].content.parts).strip()

from vertexai.generative_models import GenerativeModel,
GenerationConfig, Tool, FunctionDeclaration

# Define your function declarations (lookup_cik_fd,
retrieve_filings_fd)
sec_tool = Tool(function_declarations=[lookup_cik_fd,
retrieve_filings_fd])

system_instruction = """
    - You are a research assistant to a financial analyst.
    - Answer the user's question with an analysis, basing
```

```
    - Quote the SEC filing documents to support your
analysis.
    - If you are not certain about a CIK, use your tool to
look it up.
    - The current date is {current_date}.
    """.format(
        current_date=datetime.today().strftime("%Y-%m-%d")
    )

model = GenerativeModel(
    model_name="gemini-2.0-flash",
```

```
    generation_config=GenerationConfig(temperature=0),
    system_instruction=system_instruction,
    tools=[sec_tool]
)

response2 = model.generate_content(
    contents=f"Retrieve the 10-K for CIK {cik} covering
2023 Q4, and extract 'Risk Factors'."
)
```

4. When you ask Gemini a question related to a public company, it may return a response, or it may return a request for a function call. Starting with `Gemini 2.0`, it may ask for function calls in separate rounds of chat, or multiple **parallel function calls** in a single round of response. If multiple function calls are requested, your response **must include a function response for each function call** Gemini has requested (the `Part.from_function_response` section). To handle these cases, it is usually very helpful to define a `handle_response` function:

```
def handle_response(response):

    parts_for_inner_response = []
    for part in response.candidates[0].content.parts:
        # If the content part has an attribute called
'text'
        if hasattr(part, "text"):
            print("\n" + part.text)
            function_call = part.function_call
            try:
                if function_call.name == "lookup_cik":
                    cik =
lookup_cik(function_call.args["company_name"])
                    parts_for_inner_response.append(
                        Part.from_function_response(
                            name="lookup_cik",
                            response={
                                "content": cik,
                            },
                        )
                    )
                if function_call.name ==
"retrieve_filings":
                    context = ""
                    filing_paths =
download_range_of_filings(function_call.args["cik"],

function_call.args.get("include_10q", False)
                    )
                    if filing_paths:
                        # TODO: Load cleaned filing docs
                        report =
get_items_from_filings(function_call.args["items_of_interest"
filing_paths)
                        parts_for_inner_response.append(
                            Part.from_function_response(
                                name="retrieve_filings",
                                response={
                                    "content": report,
                                },
                            )
                        )
                    else:
                        print("No valid filings found or an
error was encountered in retrieving them.")
                print(part)
                print(e)
    if parts_for_inner_response:
        inner_response =
chat.send_message(parts_for_inner_response)
        handle_response(inner_response)
```

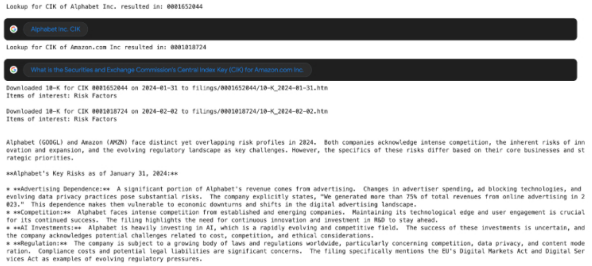
5. Start a chat session with the model.

```
chat = model.start_chat()
```

6. Run the following cell to ask a question that compares **a corresponding section across two companies reports**.

```
handle_response(response)
```

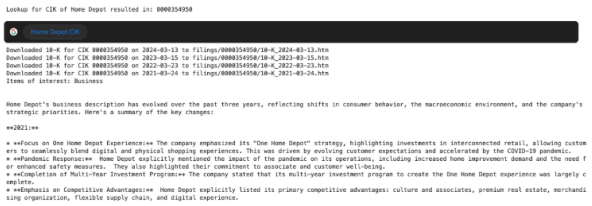
Output:



7. You can also compare sections across time. Run the following code snippet to see an example of Gemini **comparing one section across multiple 'versions' (or**

```
chat = model.start_chat()
response = chat.send_message("How has Home Depot changed
the way it describes its business over the past 3 years?")
handle_response(response)
```

Output:



8. Save your notebook.

Start a chat session and ask questions to the model

Check my progress

Assessment Completed!

Congratulations!

Congratulations! In this lab, you have explored the retrieval of structured documents from the SEC's EDGAR API. You also investigated alternative approaches for efficient and accurate document chunk retrieval. To enhance the system's capabilities, you integrated grounding with Google Search to access real-time information and leverage the search engine's understanding of complex queries. Furthermore, you developed functionalities for version comparison, enabling the analysis of changes across different document versions pertaining to distinct subjects.

Manual Last Updated May 12, 2025

Lab Last Tested May 12, 2025