National College of Ireland

HS Code classification using Agentic AI for goods import

MSc Research Project

Data Analytics

Pathmanathan Gawtham
Student ID: 23386410

School of Computing

National College of Ireland

Supervisor: Athanasios Staikopoulos

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Pathmanathan Gawtham
Student ID:	23386410
Programme:	MSc Data Analytics **Year:** 2025
Module:	Research Practicum
Lecturer:	Athanasios Staikopoulos
Submission Due Date:	11/12/2025
Project Title:	HS Code classification using Agentic AI for goods import
Word Count:	871 **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: P.Gawtham

Date: 11/12/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Pathmanathan Gawtham
Student ID: 23386410

1 Introduction

This document contains instructions and information to setup and run the submitted artifact. The project uses OpenAI model to implement an Agentic AI workflow to assign the HS codes based on the description of the product.

2 System Requirements

2.1 Hardware

- Operating System - Windows 10
- Processor – Higer than 8th gen i3 processor.
- RAM – at least 16GB
- Stable Internet Connection with at least 1 Mbps speed.

2.2 Software

- VS Code
- Jupyter Notebook
- GitHub (Link)

3 Environment Requirements

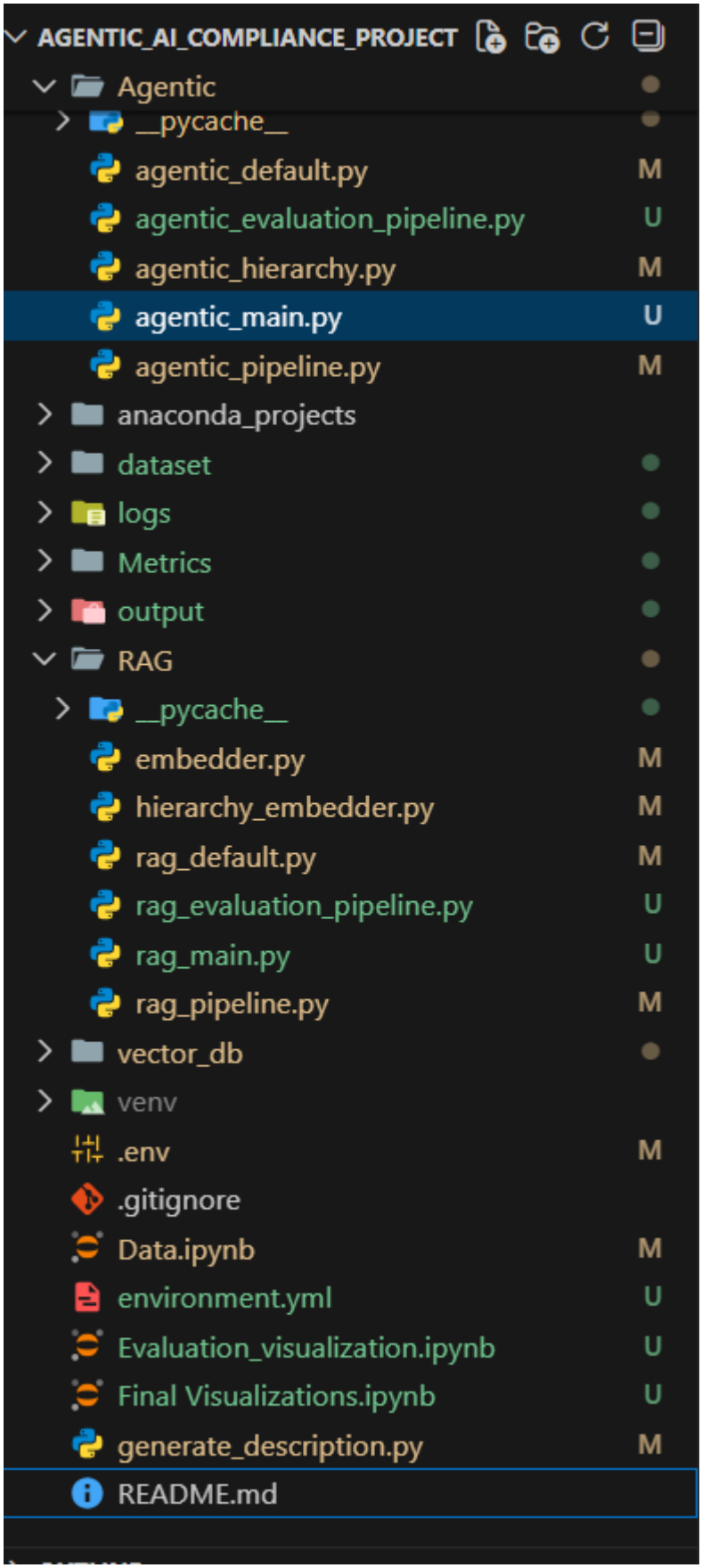
The project uses several packages in Conda environment, hence, to run the implementation the following packages are required. To ease the package installation with any environment, we have created a YAML file called **environment.yaml** which can be found in the root directory of the project.

- openai
- chromadb
- pandas
- numpy
- scikit-learn
- tqdm
- matplotlib
- seaborn

The YAML file can be executed with the following command

- conda env create -f environment.yaml
- conda activate llms

4 Project Structure



The RAG and Agentic AI implementation are stored in separate folder called “RAG” and “Agentic” respectively. The “rag_main.py” and “agentic_main.py” will contain the respective experiments

5 Dataset

Below are the sources of the dataset,

- Dataset 01 - https://github.com/datasets/harmonized-system/blob/main/data/harmonized-system.csv
- Dataset 02 - https://oec.world/en/product-landing/hs
- Synthetic Dataset - ./dataset/synthetic_product_descriptions.csv (in the project artifact dataset folder)
- Datasest 01 (root) - ./dataset/harmonized-system.csv
- Dataset 02 (root) - ./dataset/hs_product_classification.csv

The dataset used in the project can also be found under the “dataset” folder. If a different dataset needs to be tested with this implementation, the dataset should contain the two columns “hscode”, “description”.

5.1 Generating Synthetic Dataset

Following implementation was used to generate the synthetic data.

generate_description.py > ...

You, 21 hours ago | 1 author (You)

```python
from openai import OpenAI
import pandas as pd
import time
import os
from dotenv import load_dotenv

load_dotenv()

client = OpenAI(api_key=os.environ["OPENAI_API_KEY"])

hs_df = pd.read_csv("./dataset/harmonized-system.csv")
hs_df["hscode"] = hs_df["hscode"].astype(str)
hs_df = hs_df[hs_df["hscode"].str.len() >= 5]
hs_df["hscode"] = hs_df["hscode"].astype("int64")
hs_df = hs_df.sample(n=100, random_state=42)

records = []

for _, row in hs_df.iterrows():
    prompt = f"""
    You are a data generation assistant creating realistic product descriptions for international trade datasets.

    Given the following HS code and description:
    HS Code: {row['hscode']}
    HS Description: {row['description']}

    Generate 3 different product descriptions that might appear in real sales or import/export datasets.
    Each description must:
    - Be written in natural commercial language.
    - Avoid repeating the HS description verbatim.
```

Programatiically we selected 100 HS code randomly and generated sales like description for them. Each HS code description were used to create 3 different product descriptions to introduce variability within the dataset. The screenshot below shows the prompt that was used to generate the sales like descsription dataset using HS code description

```python
for _, row in hs_df.iterrows():
    prompt = f"""
    You are a data generation assistant creating realistic product descriptions for international trade datasets.

    Given the following HS code and description:
    HS Code: {row['hscode']}
    HS Description: {row['description']}

    Generate 3 different product descriptions that might appear in real sales or import/export datasets.
    Each description must:
    - Be written in natural commercial language.
    - Avoid repeating the HS description verbatim.
    - Reflect real-world variation (brand-style phrasing, product types, short/long formats).
    - Contain between 8 and 25 words.
    - Optionally include adjectives, quantity, or intended use.
    - Avoid fictional brands or unrealistic product names.

    Output format:
    1. <description>
    2. <description>
    3. <description>
    """

    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[{"role": "user", "content": prompt}],
        temperature=0.6,
    )       You, last month • RAG Pipeline and Evaluation …

    content = response.choices[0].message.content.strip()
```

6 Running the project

Once the environment is set up, you will need to activate the environment. You will have to first execute the embedding process to create the HS collection in ChormaDB first. The final artifact that was submitted contains the setup for the final stage of the experiment.

6.1 Embedder.py

RAG > embedder.py > df_hs

You, 12 seconds ago | 1 author (You)

```python
import chromadb
import os
from openai import OpenAI
from chromadb.utils import embedding_functions
from dotenv import load_dotenv
from tqdm import tqdm
import pandas as pd

load_dotenv()
client  = OpenAI()

def create_embeddings(collection_name):

    client = chromadb.PersistentClient(path=f"./vector_db/chroma_{collection_name}")

    openai_ef = embedding_functions.OpenAIEmbeddingFunction(
        model_name="text-embedding-3-large",
        api_key=os.environ["OPENAI_API_KEY"]
    )

    collection = client.get_or_create_collection(
        name=collection_name,
        embedding_function = openai_ef
    )
```

Run this python file with the command “python .\RAG\embedder.py”

and once the collection have been created, you can start testing or experimenting with the implmentation by running the following files.

- **Python .\RAG \ rag_main.py**
- **Python .\Agentic\ agentic_main.py**

6.2 rag_main.py

RAG > rag_main.py

```python
from rag_evaluation_pipeline import main_RAG_Evaluation, main_RAG_LLM_Evaluation

#------------------ DATATSET 02 ------------------

# #RAG
main_RAG_Evaluation("./dataset/test_ext_hs_codes.csv",14)
# #RAG + LLM
main_RAG_LLM_Evaluation("./dataset/test_ext_hs_codes.csv",14)


#------------------ Synthetic DATASET ------------------

#RAG
main_RAG_Evaluation("./dataset/synthetic_product_descriptions.csv",14)
# #RAG + LLM
main_RAG_LLM_Evaluation("./dataset/synthetic_product_descriptions.csv",14)

```

This file will contain the RAG and RAG with LLM experiments that are ready to run. **I would advise to run each function one by one to avoid heavy memory consumption and latency issues.**

The numeric value passed in the evaluation function is the number of retrievals the RAG will fetch from vector Database. By changing the value, you can experiment with different retrieval depth and measure the performance of different methodology.

6.3 Agentic_main.py

Agentic > agentic_main.py

```python
from agentic_evaluation_pipeline import evaluate_agentic_pipeline

#------------------ DATATSET 02 ------------------

# #Flat RAG + Agentic AI
evaluate_agentic_pipeline("./dataset/test_ext_hs_codes.csv",top_k=20)


#------------------ Synthetic DATASET ------------------

# #Flat RAG + Agentic AI
evaluate_agentic_pipeline("./dataset/synthetic_product_descriptions.csv",top_k=20)
```

This file will contain the Agentic experiments that are ready to run. **I would advise to run each function one by one to avoid heavy memory consumption and latency issues.**

7 LLM Prompts

7.1 agentic_default.py

In file agentic_default.py you will find the prompts we used for agentic AI. Below is a screenshot of Reasoning agent.

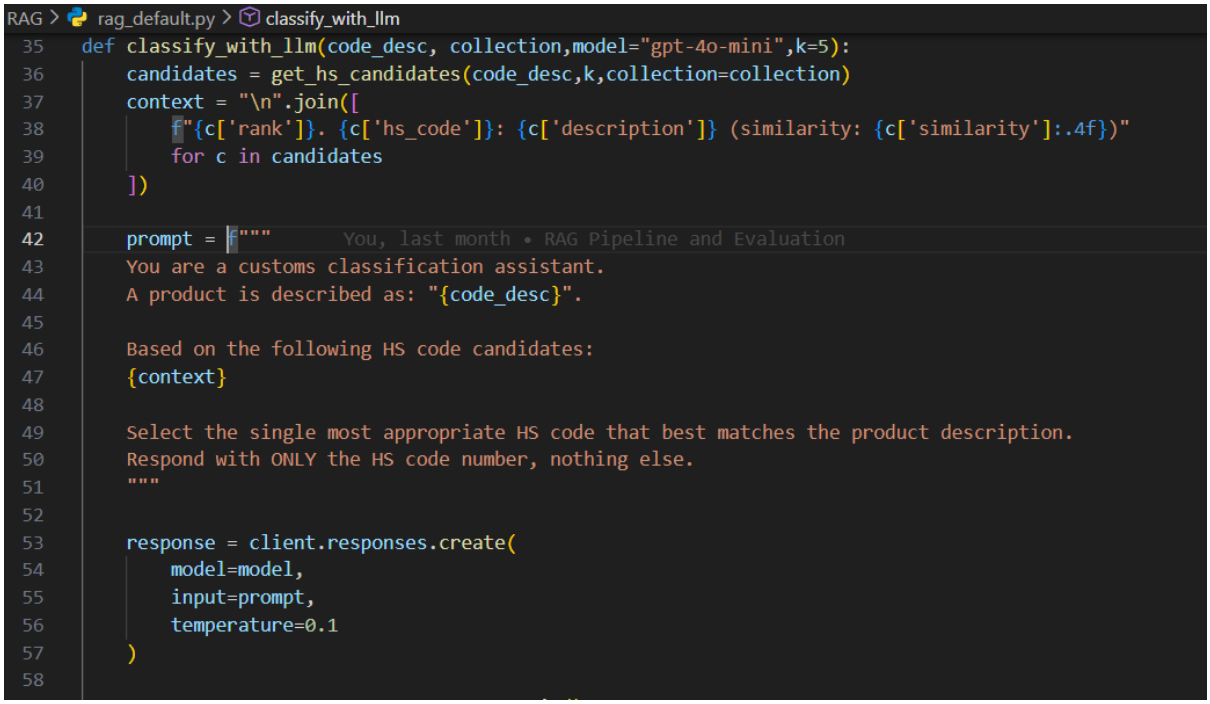
```
def reasoning_module(product_desc, candidates):
    '''
    This agentic flow is created to assess and select the suitable HS code from the
    bundle of retrieval that was queried from ChromaDB
    '''
    context = "\n".join([f"{c['rank']}. {c['hs_code']}: {c['description']} (similarity: {c['similarity']:.4f}" for i, c i
    prompt = f"""
    You are a customs classification expert.
    Product description: "{product_desc}"
    Candidate HS codes:
    {context}

    Also, use your own semantic understandings and knowledge of Harmonized System Codes to
    assign the correct code.

    The HS code should only have 5 or 6 digit codes.
    It cannot be of 4-digit or 2-digit.
    If there are any punctuations in the HS Code, remove the punctuation and persist the entire code.

    Select the single most appropriate HS code and briefly explain why.
    Respond with plain JSON only, no code blocks or additional text:
    {{ "selected_code": "xxxxx", "justification": "..." }}
    """
    resp = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[{"role": "user", "content": prompt}],
        temperature=0.0
    )
```

You can modify the prompt to experiment the behaviour of the agent. The temperature of the model can also be modified. The reflection agent also can be found in the same file.

```
def reflective_correction(product_desc, candidates, prev_result, error_msg):
    '''
    This agentic flow is created to reflect and check the suitable HS code in case an invalid
    HS code was selected by the reasoning agent.
    '''

    reflection_prompt = f"""
    You are an expert in HS Code classification.

    The previous classification attempt failed validation:
    Error: {error_msg}
    Product description: "{product_desc}"
    Candidates:
    {json.dumps(candidates, indent=2)}
    Previous justification: "{prev_result['justification']}"

    Also, use your own semantic understandings and knowledge of Harmonized System Codes to
    assign the correct code.

    The HS code should only have 5 or 6 digit codes.
    It cannot be of 4-digit or 2-digit.
    If there are any punctuations in the HS Code, remove the punctuation and persist the entire code.

    Reflect on your reasoning and provide a corrected HS code and justification.
    Respond with plain JSON only, no code blocks or additional text:
    {{ "selected_code": "xxxxx", "justification": "..." }}
    """
```

7.2 rag_default.py

Like the Agentic component, for RAG we have implemented the LLM for which we have created prompt to select the HS code from the retrieval bundle.



```
RAG > rag_default.py > classify_with_llm
35  def classify_with_llm(code_desc, collection,model="gpt-4o-mini",k=5):
36      candidates = get_hs_candidates(code_desc,k,collection=collection)
37      context = "\n".join([
38          f"{c['rank']}. {c['hs_code']}: {c['description']} (similarity: {c['similarity']:.4f})"
39          for c in candidates
40      ])
41
42      prompt = f"""      You, last month • RAG Pipeline and Evaluation
43      You are a customs classification assistant.
44      A product is described as: "{code_desc}".
45
46      Based on the following HS code candidates:
47      {context}
48
49      Select the single most appropriate HS code that best matches the product description.
50      Respond with ONLY the HS code number, nothing else.
51      """
52
53      response = client.responses.create(
54          model=model,
55          input=prompt,
56          temperature=0.1
57      )
58
```

7.3 .env

In the environmental file of the project, we have stored the OPENAI Api key with the key value "OPENAI_API_KEY". Don't share the API key with anyone who is not involved in the project. The API has 10000 call per day limitaion.

8 Output

The output of the process can be seen in the terminal where you execute the python files

8.1 Agentic flow output

```
========== Agentic Evaluation Complete ==========
Samples: 300
Top-1 Accuracy: 0.7233
Top-20 Accuracy: 0.7233
Precision: 0.5298
Recall: 0.4306
F1 Score: 0.4631
Total Evaluation Time: 856.99 sec
Avg Latency: 2.8566 sec/sample
CSV Log: ./logs/agentic_logs\agentic_eval_20251210_100624.csv
JSONL Log: ./logs/agentic_logs\agentic_eval_20251210_100624.jsonl
```

The logs can be found in the folder indicated in the screenshot. The CSV file will contain the evaluation metric values and the JSONL will contain the prediction resutls.

8.2 RAG flow output

```
(llms) PS C:\Users\Asus\Desktop\Study\Msc DA\Research\Arti

========== Evaluation Results ==========
Total samples: 300
Top-1 Accuracy: 0.5767
Top-3 Accuracy: 0.7867
Precision (macro): 0.3947
Recall (macro): 0.3035
F1 Score (macro): 0.3311
```

9 Hierarchical Retrieval Flow (optional)

You will need to first embed the HS code from dataset 01 with a hierarchical order of 2-dgit codes, 4-digit codes and then 6-digit codes. The "**hierarchy_embedder.py"** file contains the logic for it.

9.1 hierarchy_embedder.py

```
You, 21 hours ago | 1 author (You)
from openai import OpenAI      You, 3 weeks ago • Complete Agentic pipeline
import chromadb
from chromadb.utils import embedding_functions
import pandas as pd
from tqdm import tqdm
import os
from dotenv import load_dotenv

load_dotenv()
# Load HS dataset
df = pd.read_csv("./dataset/harmonized-system.csv")

client = OpenAI()
chroma_client = chromadb.PersistentClient(path="./vector_db/")

# Set up embedding function
openai_ef = embedding_functions.OpenAIEmbeddingFunction(
        model_name="text-embedding-3-large",
        api_key = os.environ["OPENAI_API_KEY"]
)

# Create collections for each level
collections = {
    2: chroma_client.get_or_create_collection(name="hs_codes_2d", embedding_function=openai_ef),
    4: chroma_client.get_or_create_collection(name="hs_codes_4d", embedding_function=openai_ef),
    6: chroma_client.get_or_create_collection(name="hs_codes_6d", embedding_function=openai_ef)
}
```

Once the embedder successfully create the Hierarchically structured HS code in vector space, we can excute the experiment to get the results.

9.2 rag_main.py

```
You, 6 minutes ago | 1 author (You)
from agentic_evaluation_pipeline import evaluate_agentic_pipeline,evaluate_agentic_hierarchy

#------------------ DATATSET 02 ------------------
print("DATATSET 02 results")
# #Flat RAG + Agentic AI
evaluate_agentic_pipeline("./dataset/test_ext_hs_codes.csv",top_k=20)

evaluate_agentic_hierarchy("./dataset/test_ext_hs_codes.csv",top_k=20)      You, 6 minutes ago • U

#------------------ Synthetic DATASET ------------------
print("Synthetic Dataset results")
# #Flat RAG + Agentic AI
evaluate_agentic_pipeline("./dataset/synthetic_product_descriptions.csv",top_k=20)
```

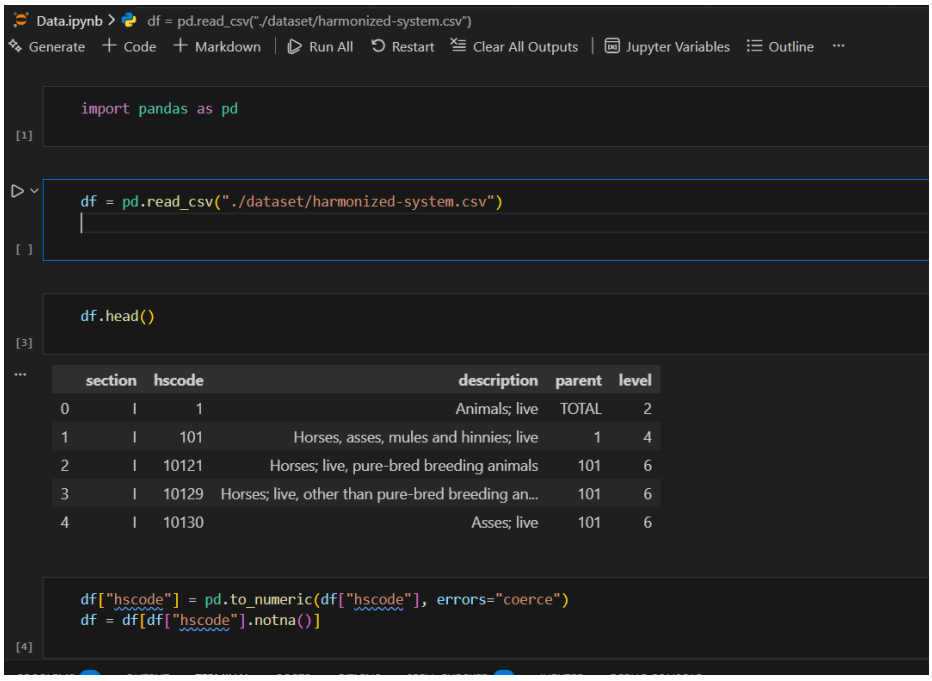
In rag_main.py file you will have to import the function "**evaluate_agentic_hierarchy**" and pass the parameter as shown in the screenshot below.

You can comment the other experiments if needed. Then you can run the rag_main.py file and wait for the results to show. (The hierarchical retrival is heavy in latency and computaion, it took around 2 hours to produce results with 1000 HS Code test in my local machine)

The log path for the outcomes is in the following folder from the root directory ./logs/agentic_hier_logs/

10 Data Cleaning

The data cleaning process and test dataset generation can be found in Data.ipynb file



11 Data Visualization

Final Visualizations.ipynb > plt.figure(figsize=(8,6))

Generate | Code | Markdown | Run All | Restart | Clear All Outputs | Jupyter Variab

```
[1]
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
[79]
df = pd.read_excel("./Metrics/Final Metrics.xlsx",sheet_name=["RAG","LLM"])
rag_df = pd.read_csv("./Metrics/RAG.csv")
rag_df
```

...

	Case	K	Sample	Top-1	Top-K	Precision	Recall	F1 Score
0	Case 01	3	1000	0.9770	1.0000	0.9550	0.9550	0.9550
1	Case 03	3	1000	0.7580	0.9330	0.6147	0.6239	0.6177
2	Case 05	3	300	0.5667	0.7800	0.3886	0.2936	0.3218
3	Case 01	7	1000	0.9760	1.0000	0.9531	0.9531	0.9531
4	Case 03	7	1000	0.7580	0.9690	0.6147	0.6239	0.6177
5	Case 05	7	300	0.5667	0.8767	0.3886	0.2936	0.3218
6	Case 01	14	1000	0.9760	1.0000	0.9531	0.9531	0.9531
7	Case 03	14	1000	0.7580	0.9840	0.6147	0.6239	0.6177
8	Case 05	14	300	0.5700	0.9333	0.3906	0.2969	0.3250

The visualization of the results are done in “Final Visualizations.ipynb” file