

How to Turn Messy Text Into a Cool Graph Database with Ollama, LangChain, & Neo4J



Minich Analytics
1.5K subscribers

Subscribe

16



Share

Ask

Download

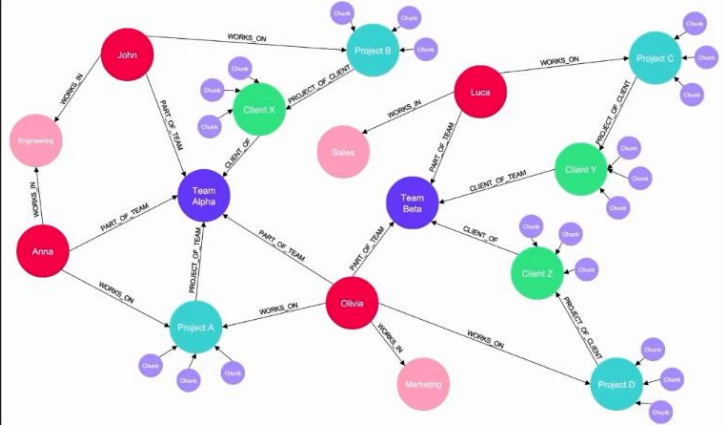


340 views Premiered Jun 4, 2025 #AI #KG #Python

In this video, we walk through how to build a Knowledge Graph (KG) from unstructured data using Python and Neo4j AuraDB. You'll learn how to extract entities and relationships, clean and merge multiple mini-KGs, and populate your graph database with meaningful connections that reflect real-world context.

We also cover common issues like entity deduplication and why some relationships might not appear in your Neo4j browser—and how to fix that.

GraphRAG with Ollama, LangChain, & Neo4j

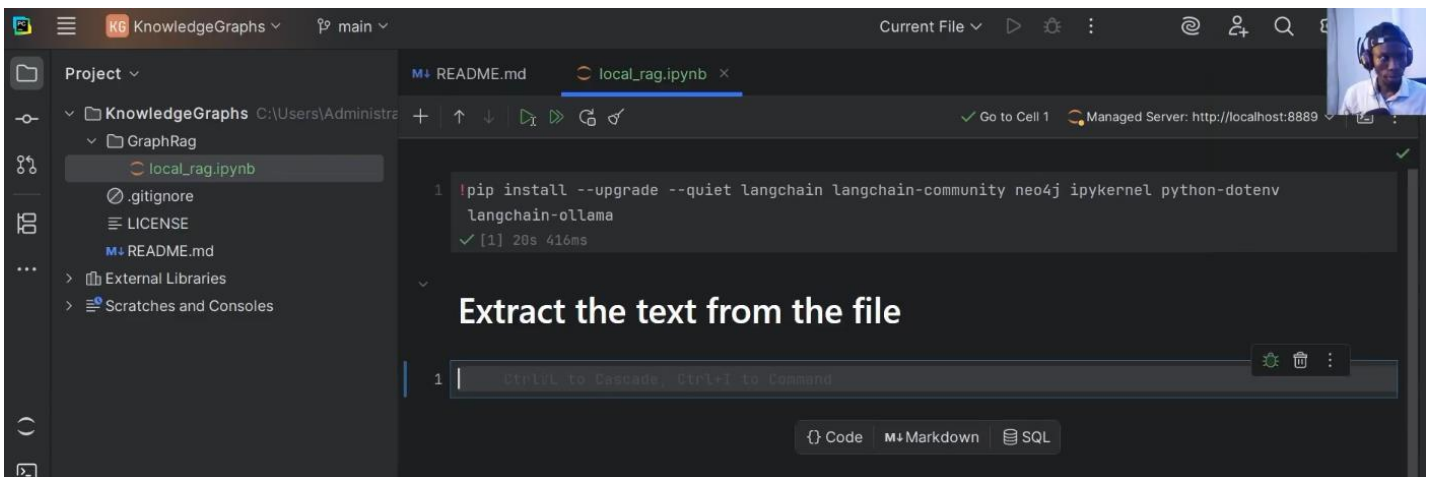


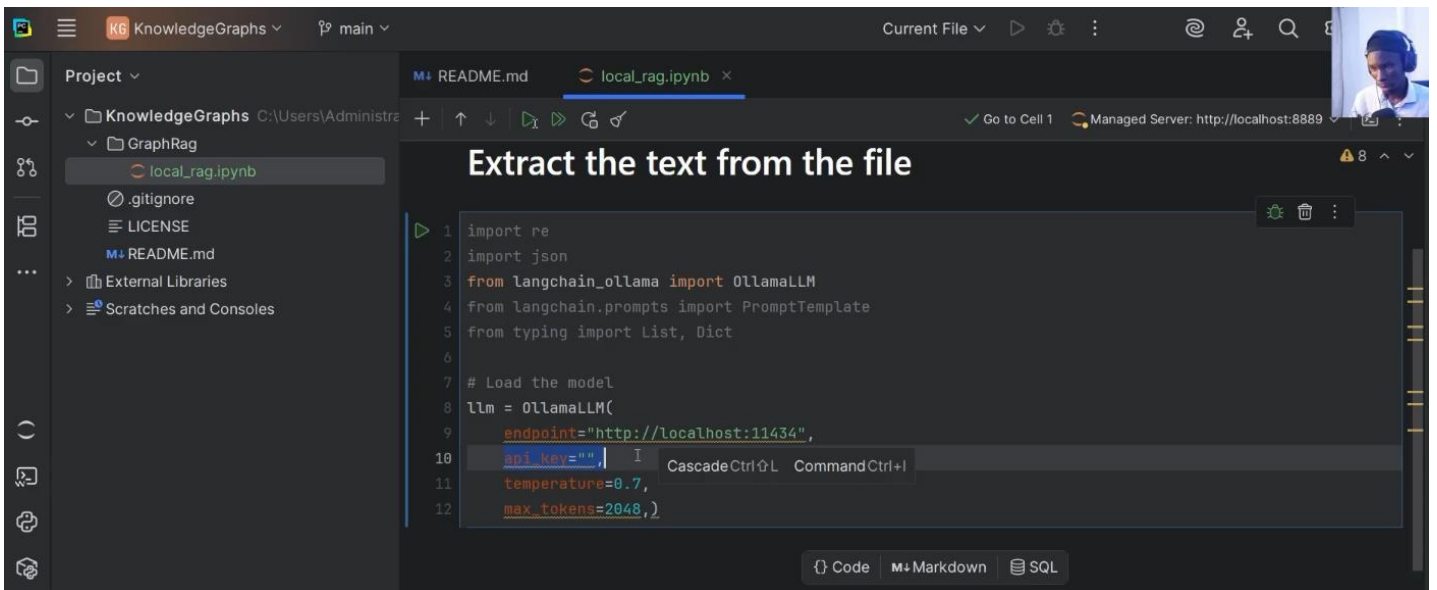
Goal:

- Build a Knowledge Graph from text and load it into Neo4j AuraDB

🌱 Steps We'll Cover:

1. Extract entities and relationships from text
2. Clean and structure the data into JSON
3. Load nodes & relationships into Neo4j
4. Visualize and query the graph





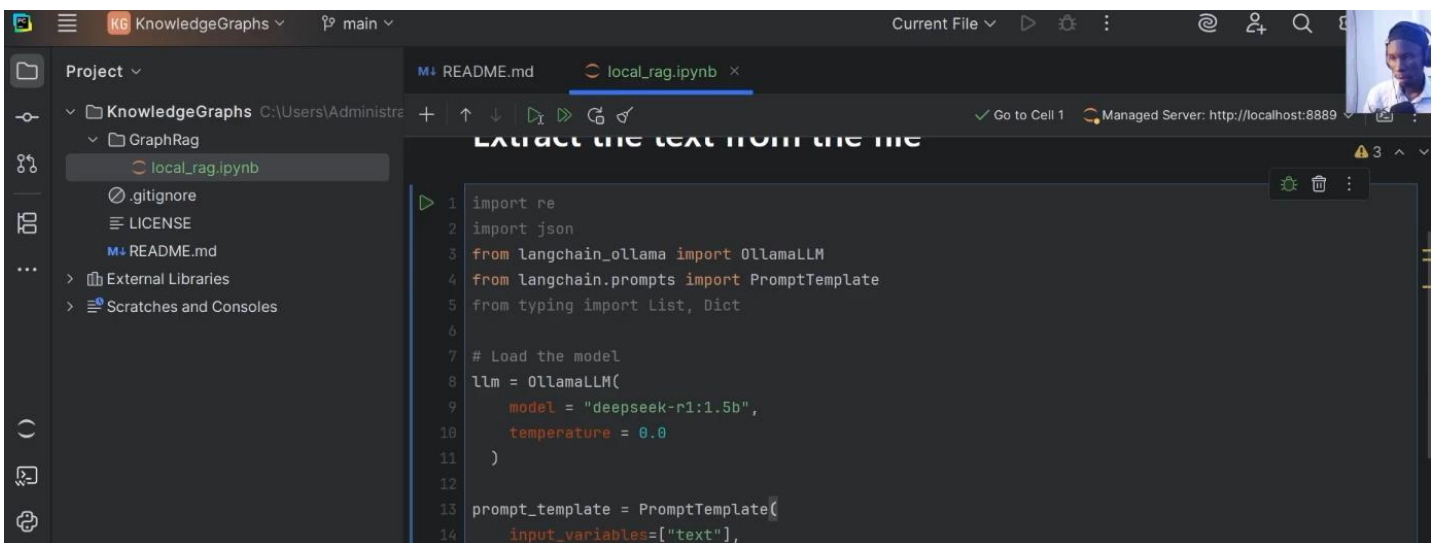
Project KnowledgeGraphs C:\Users\Administrat... main

local_rag.ipynb

Extract the text from the file

```
1 import re
2 import json
3 from langchain_ollama import OllamaLLM
4 from langchain.prompts import PromptTemplate
5 from typing import List, Dict
6
7 # Load the model
8 llm = OllamaLLM(
9     endpoint="http://localhost:11434",
10     api_key="",
11     temperature=0.7,
12     max_tokens=2048,)
```

Code Markdown SQL

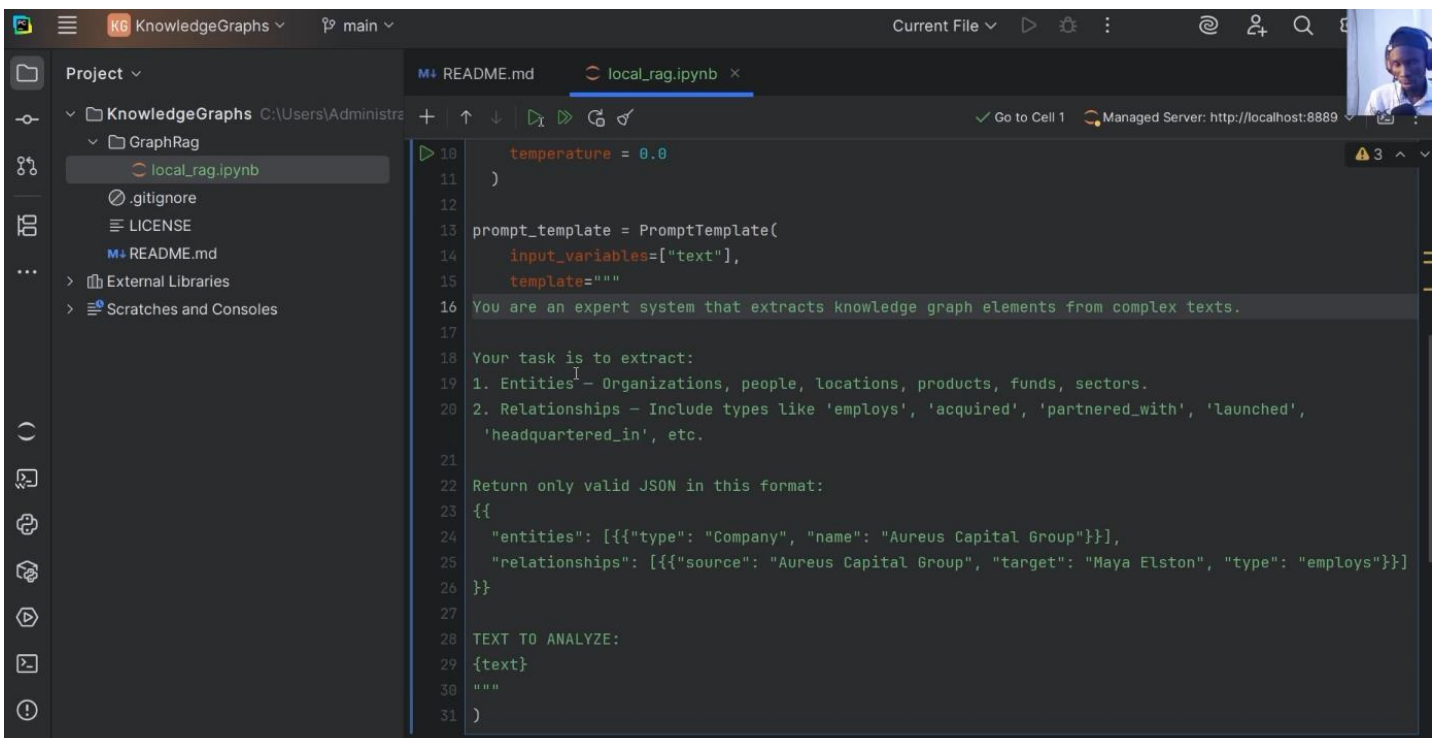


Project KnowledgeGraphs C:\Users\Administrat... main

local_rag.ipynb

Extract the text from the file

```
1 import re
2 import json
3 from langchain_ollama import OllamaLLM
4 from langchain.prompts import PromptTemplate
5 from typing import List, Dict
6
7 # Load the model
8 llm = OllamaLLM(
9     model = "deepseek-r1:1.5b",
10     temperature = 0.0
11 )
12
13 prompt_template = PromptTemplate(
14     input_variables=["text"],
```



Project KnowledgeGraphs C:\Users\Administrat... main

local_rag.ipynb

```
10     temperature = 0.0
11 )
12
13 prompt_template = PromptTemplate(
14     input_variables=["text"],
15     template=""
16 )
17
18 You are an expert system that extracts knowledge graph elements from complex texts.
19
20 Your task is to extract:
21 1. Entities - Organizations, people, locations, products, funds, sectors.
22 2. Relationships - Include types like 'employs', 'acquired', 'partnered_with', 'launched',
23    'headquartered_in', etc.
24
25 Return only valid JSON in this format:
26 {{
27     "entities": [{"type": "Company", "name": "Aureus Capital Group"}],
28     "relationships": [{"source": "Aureus Capital Group", "target": "Maya Elston", "type": "employs"}]}
29
30 TEXT TO ANALYZE:
31 {text}
32 ""
33 )
```

KnowledgeGraphs main

Project

- KnowledgeGraphs C:\Users\Administr...
- GraphRag
 - local_rag.ipynb
 - .gitignore
 - LICENSE
 - README.md
- External Libraries
- Scratches and Consoles

local_rag.ipynb

Return only valid JSON in this format:

```
{
  "entities": [{"type": "Company", "name": "Aureus Capital Group"}],
  "relationships": [{"source": "Aureus Capital Group", "target": "Maya Elston", "type": "employs"}]}

```

TEXT TO ANALYZE:

```
{text}
"""
)
```

Chunking

Chunking

```
1 import re
2 from typing import List, Dict
3
4 | Ctrl+L to Cascade, Ctrl+I to Command
5 def smart_chunk(text: str, max_words: int = 300) -> List[str]:
6     # Split by section headers and paragraphs
7     sections = re.split(r'\n\s*\n+', text)
8     chunks = []
9     current_chunk = ""
10
11     for section in sections:
12         words = section.split()
13         if len(current_chunk.split()) + len(words) <= max_words:
```

KnowledgeGraphs main

Project

- KnowledgeGraphs C:\Users\Administr...
- GraphRag
 - local_rag.ipynb
 - .gitignore
 - LICENSE
 - README.md
- External Libraries
- Scratches and Consoles

local_rag.ipynb

```
6     chunks = []
7     current_chunk = ""
8
9     for section in sections:
10         words = section.split()
11         if len(current_chunk.split()) + len(words) <= max_words:
12             current_chunk += " " + section
13         else:
14             chunks.append(current_chunk.strip())
15             current_chunk = section
16
17     if current_chunk:
18         chunks.append(current_chunk.strip())
19
20     return chunks

```

[5] 13ms

Extraction and cleaning

Double-click to edit this empty Markdown cell

KnowledgeGraphs main

Project

- KnowledgeGraphs C:\Users\Administrat...
- GraphRag
 - local_rag.ipynb
 - .gitignore
 - LICENSE
 - README.md
- External Libraries
- Scratches and Consoles

Extraction and cleaning

```
1 from typing import Dict
2 def extract_kg(text_chunk: str) -> Dict:
3     try:
4         chain = prompt_template | llm
5         result = chain.invoke({"text": text_chunk})
6
7         json_str = re.search(r'\{[s\S]*\}', result).group(0)
8         return json.loads(json_str)
9
10    except Exception as e:
11        print(f"Error extracting from chunk: {e}")
12        print("Raw output:\n", result)
13        return {"entities": [], "relationships": []}
14
```

Double-click to edit this empty Markdown cell

KnowledgeGraphs main

Project

- KnowledgeGraphs C:\Users\Administrat...
- GraphRag
 - local_rag.ipynb
 - .gitignore
 - LICENSE
 - README.md
- External Libraries
- Scratches and Consoles

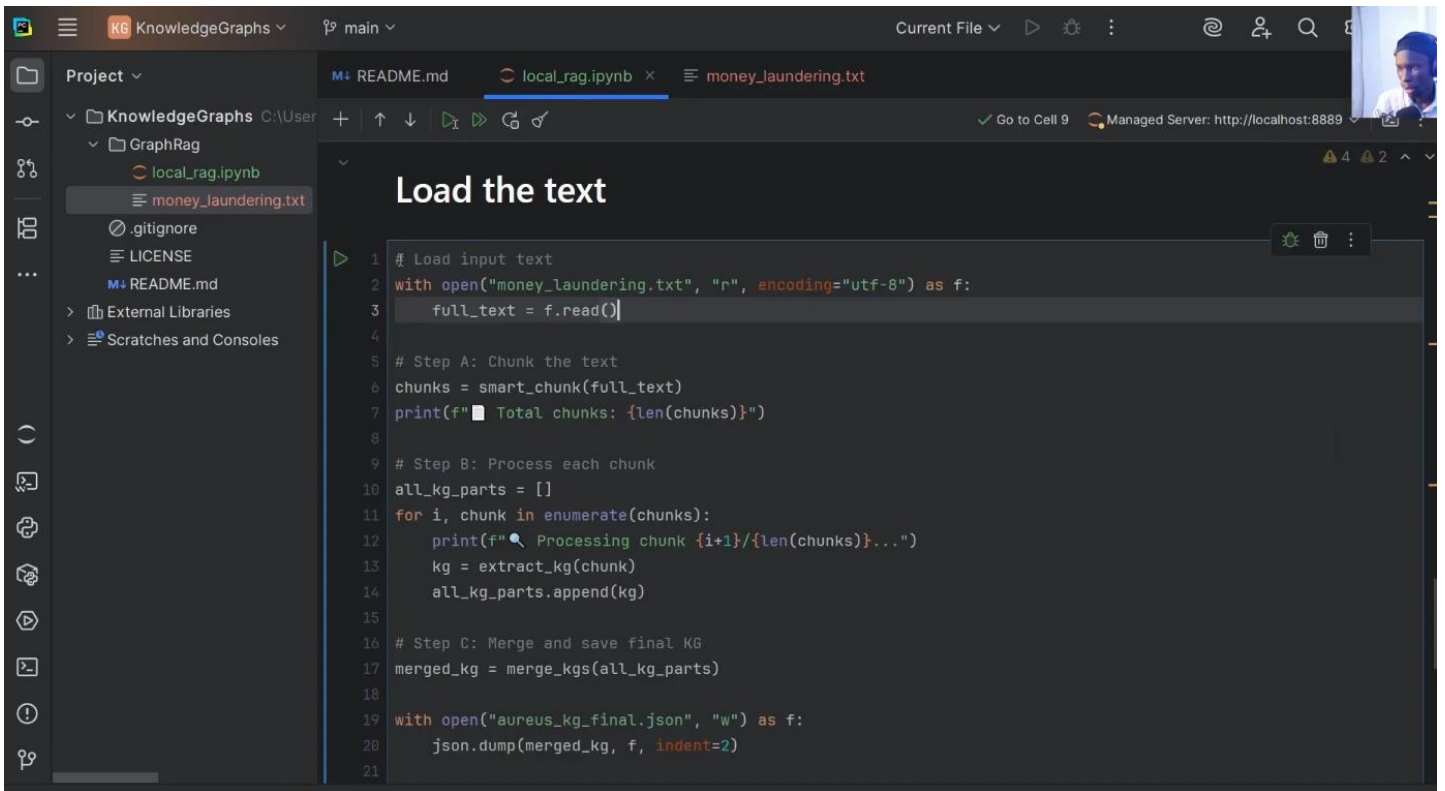
Merging - Post processing

```
1 def merge_kgs(kg_list: List[Dict]) -> Dict:
2     seen_entities = set()
3     all_entities = []
4     all_relationships = []
5
6     for kg in kg_list:
7         for entity in kg["entities"]:
8             key = (entity["type"], entity["name"])
9             if key not in seen_entities:
10                 all_entities.append(entity)
11                 seen_entities.add(key)
12
13         all_relationships.extend(kg["relationships"])
14
15     return {"entities": all_entities, "relationships": all_relationships}
16
```

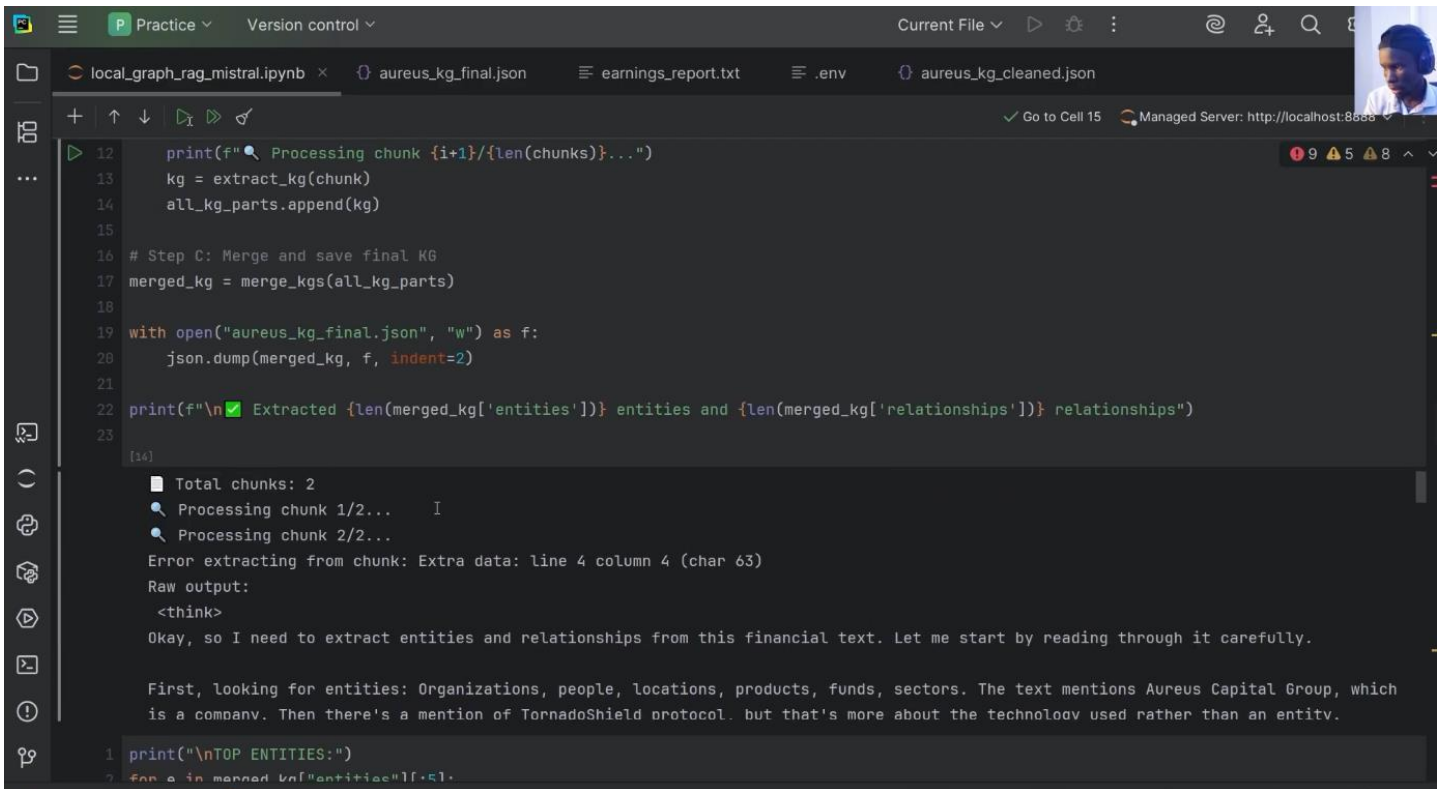
✓ [6] 488ms

Load the text

Double-click to edit this empty Markdown cell



```
1 # Load input text
2 with open("money_laundering.txt", "r", encoding="utf-8") as f:
3     full_text = f.read()
4
5 # Step A: Chunk the text
6 chunks = smart_chunk(full_text)
7 print(f"Total chunks: {len(chunks)}")
8
9 # Step B: Process each chunk
10 all_kg_parts = []
11 for i, chunk in enumerate(chunks):
12     print(f"Processing chunk {i+1}/{len(chunks)}...")
13     kg = extract_kg(chunk)
14     all_kg_parts.append(kg)
15
16 # Step C: Merge and save final KG
17 merged_kg = merge_kgs(all_kg_parts)
18
19 with open("aureus_kg_final.json", "w") as f:
20     json.dump(merged_kg, f, indent=2)
21
```



```
12 print(f"Processing chunk {i+1}/{len(chunks)}...")
13 kg = extract_kg(chunk)
14 all_kg_parts.append(kg)
15
16 # Step C: Merge and save final KG
17 merged_kg = merge_kgs(all_kg_parts)
18
19 with open("aureus_kg_final.json", "w") as f:
20     json.dump(merged_kg, f, indent=2)
21
22 print(f"\nExtracted {len(merged_kg['entities'])} entities and {len(merged_kg['relationships'])} relationships")
23
```

[14]

■ Total chunks: 2
🔍 Processing chunk 1/2... I
🔍 Processing chunk 2/2...

Error extracting from chunk: Extra data: line 4 column 4 (char 63)

Raw output:
<think>
Okay, so I need to extract entities and relationships from this financial text. Let me start by reading through it carefully.

First, looking for entities: Organizations, people, locations, products, funds, sectors. The text mentions Aureus Capital Group, which is a company. Then there's a mention of TornadoShield protocol. but that's more about the technology used rather than an entity.

```
1 print("\nTOP ENTITIES:")
2 for e in merged_kg["entities"][:5]:
```



```
Practice Version control Current File
local_graph_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg_cleaned.json
Go to Cell 15 Managed Server: http://localhost:8888

12 print(f" Processing chunk {i+1}/{len(chunks)}...")
13 kg = extract_kg(chunk)
14 all_kg_parts.append(kg)
15
16 # Step C: Merge and save final KG
17 merged_kg = merge_kgs(all_kg_parts)
18
19 with open("aureus_kg_final.json", "w") as f:
20     json.dump(merged_kg, f, indent=2)
21
22 print(f"\n✅ Extracted {len(merged_kg['entities'])} entities and {len(merged_kg['relationships'])} relationships")
23
[14]
Okay, so I need to extract entities and relationships from this financial text. Let me start by reading through it carefully.

First I looking for entities: Organizations, people, locations, products, funds, sectors. The text mentions Aureus Capital Group, which is a company. Then there's a mention of TornadoShield protocol, but that's more about the technology used rather than an entity. Blockchain analytics firms like ChainVisor and ForensicByte are mentioned as tools used in the investigation.

Next, people: Linda Meng is mentioned as the Chief Compliance Officer at TriMark Bank. She was placed on leave and under investigation for bribes. So that's a person involved in the investigation.

Locations: The text talks about Riga, Latvia, Istanbul, Turkey (from TornadoShield), Liechtenstein (TriMark Bank). Also mentions Monte

1 print("\nTOP ENTITIES:")
2 for e in merged_kg["entities"][:5]:
```

```
Practice Version control Current File
local_graph_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg_cleaned.json
Go to Cell 15 Managed Server: http://localhost:8888

15
16 # Step C: Merge and save final KG
17 merged_kg = merge_kgs(all_kg_parts)
18
19 with open("aureus_kg_final.json", "w") as f:
20     json.dump(merged_kg, f, indent=2)
21
22 print(f"\n✅ Extracted {len(merged_kg['entities'])} entities and {len(merged_kg['relationships'])} relationships")
23
[14]
Now for relationships. The text describes how funds were routed through wallets linked to IP addresses in Riga, Latvia, Istanbul, Turkey. It mentions emails from TriMark Bank exposed without KYC checks. The Chief Compliance Officer was on leave and under investigation.

So the relationships are:

I
- Auerus Capital Group employs or links with others (employs)
- Auerus Capital Group is involved in routing funds through wallets
- Auerus Capital Group is linked to other companies via wallet clusters
- Auerus Capital Group is linked to TriMark Bank, which was exposed for financial fraud

1 print("\nTOP ENTITIES:")
2 for e in merged_kg["entities"][:5]:
3     print(f"- {e['type']}: <15>: {e['name']}")
4
5 print("\nTOP RELATIONSHIPS:")
```

Practice > GraphRag > local_graph_rag_mistral.ipynb 98:1 LF UTF-8 Windsurf 4 spaces Python 3.12 (Practice)

PracticeVersion control

local_graph_rag_mistral.ipynb | aureus_kg_final.json | earnings_report.txt | .env | aureus_kg_cleaned.json

Go to Cell 15Managed Server: http://localhost:8888

```
15
16 # Step C: Merge and save final KG
17 merged_kg = merge_kgs(all_kg_parts)
18
19 with open("aureus_kg_final.json", "w") as f:
20     json.dump(merged_kg, f, indent=2)
21
22 print(f"\n✅ Extracted {len(merged_kg['entities'])} entities and {len(merged_kg['relationships'])} relationships")
23
[14]
```

So the relationships are:

- Aureus Capital Group employs or links with others (employs)
- Aureus Capital Group is involved in routing funds through wallets
- Aureus Capital Group is linked to other companies via wallet clusters
- Aureus Capital Group is linked to TriMark Bank, which was exposed for financial fraud

Wait, the text says "exposed from TriMark Bank (Liechtenstein)", so that's a relationship between Aureus and TriMark.

```
1 print("\nTOP ENTITIES:")
2 for e in merged_kg["entities"][:5]:
3     print(f"- {e['type']}: <15>: {e['name']}")
4
5 print("\nTOP RELATIONSHIPS:")
```

PracticeVersion control

local_graph_rag_mistral.ipynb | aureus_kg_final.json | earnings_report.txt | .env | aureus_kg_cleaned.json

Go to Cell 15Managed Server: http://localhost:8888

```
15
16 # Step C: Merge and save final KG
17 merged_kg = merge_kgs(all_kg_parts)
18
19 with open("aureus_kg_final.json", "w") as f:
20     json.dump(merged_kg, f, indent=2)
21
22 print(f"\n✅ Extracted {len(merged_kg['entities'])} entities and {len(merged_kg['relationships'])} relationships")
23
[14]
```

between these as well.

Putting it all together:

Entities:

```
[
  {
    "type": "Company",
    "name": "Aureus Capital Group"
  },
```

```
1 print("\nTOP ENTITIES:")
2 for e in merged_kg["entities"][:5]:
3     print(f"- {e['type']}: <15>: {e['name']}")
4
5 print("\nTOP RELATIONSHIPS:")
```

Practice

Version control

Project

Practice

.venv library root

GraphRag

.env

aureus_kg_cleaned.json

aureus_kg_final.json

earnings_report.txt

local_graph_rag_mistral.ipynb

enhancing_rag_with_graph.ipynb

main.py

External Libraries

Scratches and Consoles

local_graph_rag_mistral.ipynb

aureus_kg_final.json

earnings_report.txt

.env

aureus_kg...

Go to Cell 15

Managed Server: http://localhost:8888

15

16 # Step C: Merge and save final KG

17 merged_kg = merge_kgs(all_kg_parts)

18

19 with open("aureus_kg_final.json", "w") as f:

20 json.dump(merged_kg, f, indent=2)

21

22 print(f"\n✅ Extracted {len(merged_kg['entities'])} entities and {len(merged_kg['relationships'])} relationships")

23

[14]

Entities:

[

{

"type": "Company",

"name": "Aureus Capital Group"

},

{

"type": "Technology",

"name": "TornadoShield protocol"

}

]

1 print("\nTOP ENTITIES:")

2 for e in merged_kg["entities"][:5]:

3 print(f"- {e['type']}: <15>: {e['name']}")

4

Practice

Version control

Project

Practice

.venv library root

GraphRag

.env

aureus_kg_cleaned.json

aureus_kg_final.json

earnings_report.txt

local_graph_rag_mistral.ipynb

enhancing_rag_with_graph.ipynb

main.py

External Libraries

Scratches and Consoles

local_graph_rag_mistral.ipynb

aureus_kg_final.json

earnings_report.txt

.env

aureus_kg...

Go to Cell 15

Managed Server: http://localhost:8888

15

16 # Step C: Merge and save final KG

17 merged_kg = merge_kgs(all_kg_parts)

18

19 with open("aureus_kg_final.json", "w") as f:

20 json.dump(merged_kg, f, indent=2)

21

22 print(f"\n✅ Extracted {len(merged_kg['entities'])} entities and {len(merged_kg['relationships'])} relationships")

23

[14]

"name": "PaladinGraph AI"

},

{

"type": "NLP algorithm",

"name": "NLP algorithms"

},

{

"type": "Link prediction model",

"name": "link prediction models"

}

],

1 print("\nTOP ENTITIES:")

2 for e in merged_kg["entities"][:5]:

3 print(f"- {e['type']}: <15>: {e['name']}")

4

Practice Version control

Project

- Practice D:\Practice
 - .venv library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

local_graph_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg_

Go to Cell 15 Managed Server: http://localhost:8888

```
1 print("\nTOP ENTITIES:")
2 for e in merged_kg["entities"][:5]:
3     print(f"- {e['type']}: <15>: {e['name']}")
4
5 print("\nTOP RELATIONSHIPS:")
6 for r in merged_kg["relationships"][:5]:
7     print(f"- {r['source']} --[{r['type']}]--> {r['target']}")
8
```

[15]

- company : Aureus Capital Group
- company : Orion Strategic Holdings
- location : Zurich, Switzerland
- location : Dubai, UAE
- location : Singapore

TOP RELATIONSHIPS:

- Aureus Capital Group --[employs]--> Alexander Volkov
- Aureus Capital Group --[partnered_with]--> Andromeda Resources
- Aureus Capital Group --[partnered_with]--> Bayfront Maritime Logistics

Practice Version control

Project

- Practice D:\Practice
 - .venv library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg_cleaned.json

Go to Cell 15 Managed Server: http://localhost:8888

```
1 print("\nTOP ENTITIES:")
2 for e in merged_kg["entities"][:5]:
3     print(f"- {e['type']}: <15>: {e['name']}")
4
5 print("\nTOP RELATIONSHIPS:")
6 for r in merged_kg["relationships"][:5]:
7     print(f"- {r['source']} --[{r['type']}]--> {r['target']}")
8
```

[15]

- location : Zurich, Switzerland
- location : Dubai, UAE
- location : Singapore

TOP RELATIONSHIPS:

- Aureus Capital Group --[employs]--> Alexander Volkov
- Aureus Capital Group --[partnered_with]--> Andromeda Resources
- Aureus Capital Group --[partnered_with]--> Bayfront Maritime Logistics
- Orion Strategic Holdings --[employs]--> Alexander Volkov
- Orion Strategic Holdings --[partnered_with]--> Andromeda Resources

Build the Knowledge Graph

Practice Version control

Project

- Practice D:\Practice
 - .venv library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

local_graph_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg...

Go to Cell 15 Managed Server: http://localhost:8686

Build the Knowledge Graph

```
1
2 from neo4j import GraphDatabase
3 import json
4 from dotenv import load_dotenv
5 import os
6
7 load_dotenv()
8
9 class Neo4JGraph:
10     def __init__(self):
11         self.uri = os.getenv("NEO4J_URI")
12         self.user = os.getenv("NEO4J_USERNAME")
13         self.password = os.getenv("NEO4J_PASSWORD")
14
15     try:
16         self.driver = GraphDatabase.driver(
17             self.uri,
18             auth=(self.user, self.password),
19             max_connection_lifetime=30
20         )
21         print("Connection initialized")
22     except Exception as e:
23         print(f"X Driver creation failed: {str(e)}")
```

Practice Version control

Project

- Practice D:\Practice
 - .venv library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

local_graph_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg...

Go to Cell 15 Managed Server: http://localhost:8686

```
17         self.uri,
18         auth=(self.user, self.password),
19         max_connection_lifetime=30
20     )
21     print("Connection initialized")
22 except Exception as e:
23     print(f"X Driver creation failed: {str(e)}")
24     raise
25
26 def verify_connection(self):
27     try:
28         with self.driver.session() as session:
29             result = session.run("RETURN 1 AS test")
30             return result.single()["test"] == 1
31     except Exception as e:
32         print(f"Connection test failed: {str(e)}")
33         return False
34
35 def create_graph(self, entities, relationships):
36     if not self.verify_connection():
37         raise ConnectionError("Cannot proceed without valid connection")
38
39     with self.driver.session() as session:
40         session.run("MATCH (n) DETACH DELETE n")
41
```

Practice Version control

Project

- Practice D:\Practice
 - .env library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

local_graph_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg...

Go to Cell 15 Managed Server: http://localhost:8888

```
32 print(f"Connection test failed: {str(e)}")
33 return False
34
35 def create_graph(self, entities, relationships):
36     if not self.verify_connection():
37         raise ConnectionError("Cannot proceed without valid connection")
38
39     with self.driver.session() as session:
40         session.run("MATCH (n) DETACH DELETE n")
41
42         # Create nodes
43         for entity in entities:
44             label = entity["type"]
45             name = entity["name"]
46             session.run(
47                 f"MERGE (n:{label} {{name: $name}})",
48                 name=name
49             )
50
51         # Create relationships
52         for rel in relationships:
53             query = f"""
54             MATCH (a:{rel['source_type']} {{name: $source}})
55             MATCH (b:{rel['target_type']} {{name: $target}})
56             MERGE (a)-[r:{rel['type'].upper()}]->(b)
```

Practice Version control

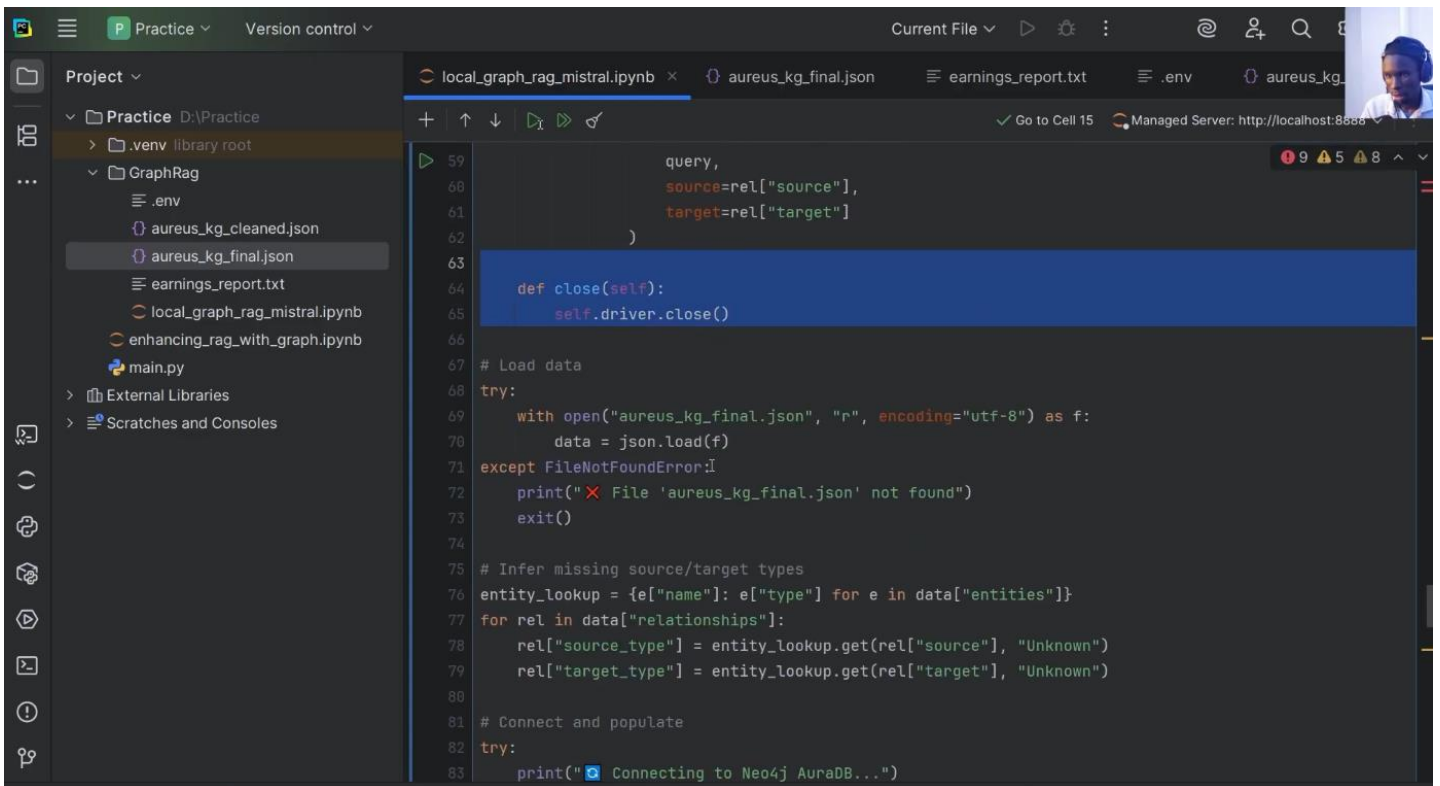
Project

- Practice D:\Practice
 - .env library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

local_graph_rag_mistral.ipynb aureus_kg_final.json earnings_report.txt .env aureus_kg...

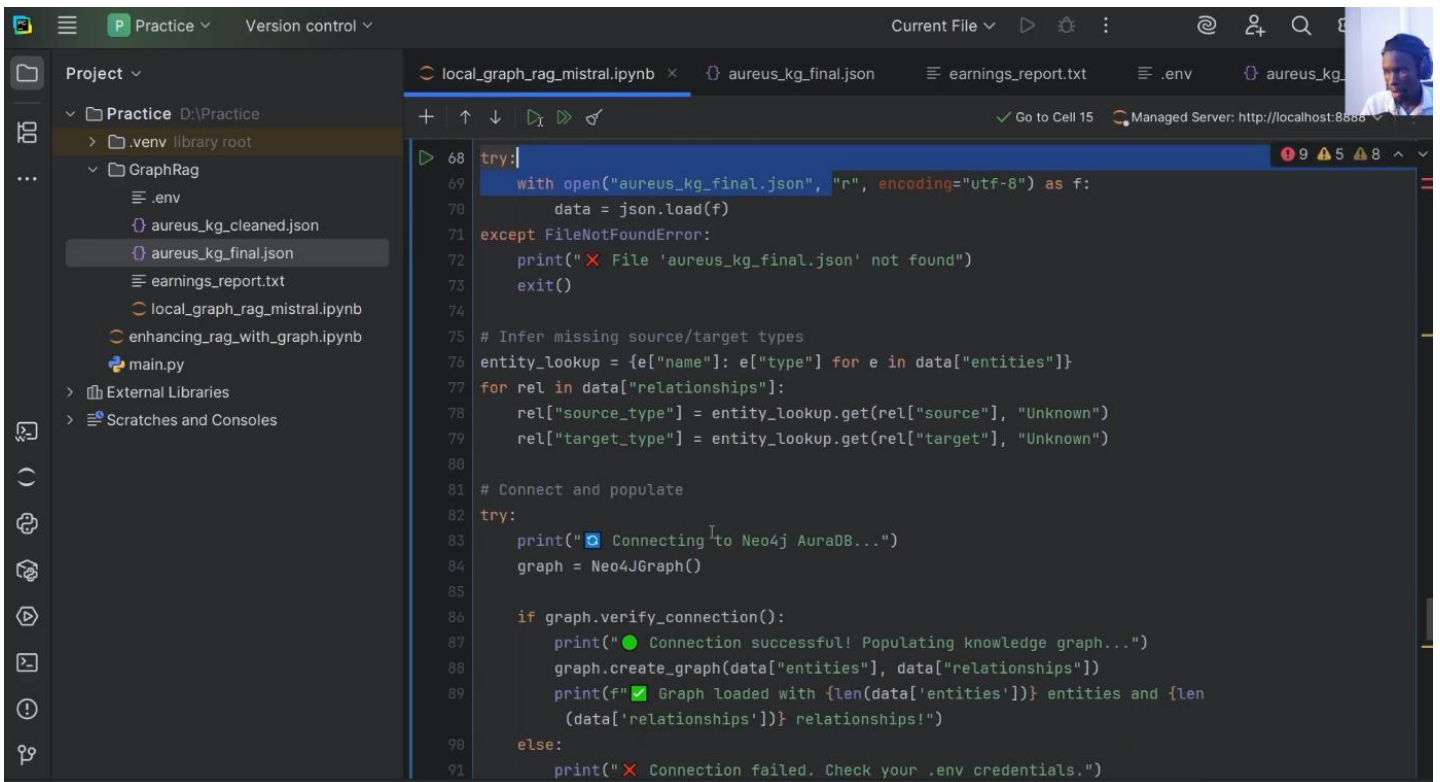
Go to Cell 15 Managed Server: http://localhost:8888

```
44 label = entity["type"]
45 name = entity["name"]
46 session.run(
47     f"MERGE (n:{label} {{name: $name}})",
48     name=name
49 )
50
51 # Create relationships
52 for rel in relationships:
53     query = f"""
54     MATCH (a:{rel['source_type']} {{name: $source}})
55     MATCH (b:{rel['target_type']} {{name: $target}})
56     MERGE (a)-[r:{rel['type'].upper()}]->(b)
57     """
58     session.run(
59         query,
60         source=rel["source"],
61         target=rel["target"]
62     )
63
64 def close(self):
65     self.driver.close()
66
67 # Load data
68 try:
```



The screenshot shows a Jupyter Notebook interface with a dark theme. The left sidebar displays a project structure for 'Practice' located at 'D:\Practice'. The project contains a '.venv' directory, a 'GraphRag' folder, and several files including '.env', 'aureus_kg_cleaned.json', 'aureus_kg_final.json', 'earnings_report.txt', 'local_graph_rag_mistral.ipynb', 'enhancing_rag_with_graph.ipynb', and 'main.py'. The main editor area shows the 'local_graph_rag_mistral.ipynb' file, which contains a Python script. The script defines a 'query' function, a 'close' method for a class, and a main execution block that loads data from 'aureus_kg_final.json', infers missing source/target types, and connects to a Neo4j AuraDB instance. The script is currently at line 83, with a status bar indicating 'Go to Cell 15' and a 'Managed Server' URL of 'http://localhost:8668'.

```
59         query,
60         source=rel["source"],
61         target=rel["target"]
62     )
63
64     def close(self):
65         self.driver.close()
66
67 # Load data
68 try:
69     with open("aureus_kg_final.json", "r", encoding="utf-8") as f:
70         data = json.load(f)
71 except FileNotFoundError:
72     print("❌ File 'aureus_kg_final.json' not found")
73     exit()
74
75 # Infer missing source/target types
76 entity_lookup = {e["name"]: e["type"] for e in data["entities"]}
77 for rel in data["relationships"]:
78     rel["source_type"] = entity_lookup.get(rel["source"], "Unknown")
79     rel["target_type"] = entity_lookup.get(rel["target"], "Unknown")
80
81 # Connect and populate
82 try:
83     print("🔌 Connecting to Neo4j AuraDB...")
```



This screenshot shows the continuation of the Python script from the previous image. The script now includes a 'try' block starting at line 68, which attempts to load the JSON data. It also includes a 'graph.verify_connection()' call to check the database connection, followed by 'graph.create_graph()' to create the knowledge graph. The script concludes with a print statement indicating the graph is loaded with a specific number of entities and relationships. The status bar at the bottom indicates 'Go to Cell 15' and the 'Managed Server' URL.

```
68 try:
69     with open("aureus_kg_final.json", "r", encoding="utf-8") as f:
70         data = json.load(f)
71 except FileNotFoundError:
72     print("❌ File 'aureus_kg_final.json' not found")
73     exit()
74
75 # Infer missing source/target types
76 entity_lookup = {e["name"]: e["type"] for e in data["entities"]}
77 for rel in data["relationships"]:
78     rel["source_type"] = entity_lookup.get(rel["source"], "Unknown")
79     rel["target_type"] = entity_lookup.get(rel["target"], "Unknown")
80
81 # Connect and populate
82 try:
83     print("🔌 Connecting to Neo4j AuraDB...")
84     graph = Neo4jGraph()
85
86     if graph.verify_connection():
87         print("🟢 Connection successful! Populating knowledge graph...")
88         graph.create_graph(data["entities"], data["relationships"])
89         print(f"✅ Graph loaded with {len(data['entities'])} entities and {len(
90             data['relationships'])} relationships!")
91     else:
92         print("❌ Connection failed. Check your .env credentials.")
```


Project ▾

- Practice D:\Practice
 - .venv library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

local_graph_rag_mistral.ipynb × aureus_kg_final.json earnings_report.txt .env aureus_kg...

Go to Cell 15 Managed Server: http://localhost:8888

```
79 rel["target_type"] = entity_lookup.get(rel["target"], "Unknown")
80
81 # Connect and populate
82 try:
83     print("🔌 Connecting to Neo4j AuraDB...")
84     graph = Neo4jGraph()
85
86     if graph.verify_connection():
87         print("🟢 Connection successful! Populating knowledge graph...")
88         graph.create_graph(data["entities"], data["relationships"])
89         print(f"✅ Graph loaded with {len(data['entities'])} entities and {len(
            (data['relationships']))} relationships!")
90     else:
91         print("❌ Connection failed. Check your .env credentials.")
92 except Exception as e:
93     print(f"❌ Critical error: {str(e)}")
94 finally:
95     graph.close()
96
97 446ms
```

```
1 from pyvis.network import Network
2
3 # Create a PyVis network
```

Practice ▾ Version control ▾

Project ▾

- Practice D:\Practice
 - .venv library root
 - GraphRag
 - .env
 - aureus_kg_cleaned.json
 - aureus_kg_final.json
 - earnings_report.txt
 - local_graph_rag_mistral.ipynb
 - enhancing_rag_with_graph.ipynb
 - main.py
 - External Libraries
 - Scratches and Consoles

local_graph_rag_mistral.ipynb × aureus_kg_final.json earnings_report.txt .env aureus_kg...

Current File ▾ Go to Cell 15 Managed Server: http://localhost:8888

```
85
86
87     if graph.verify_connection():
88         print("🟢 Connection successful! Populating knowledge graph...")
89         graph.create_graph(data["entities"], data["relationships"])
90         print(f"✅ Graph loaded with {len(data['entities'])} entities and {len(
            (data['relationships']))} relationships!")
91     else:
92         print("❌ Connection failed. Check your .env credentials.")
93 except Exception as e:
94     print(f"❌ Critical error: {str(e)}")
95 finally:
96     graph.close()
97
98 [1] 8s 388ms
```

```
🔌 Connecting to Neo4j AuraDB...
🔌 Connection initialized
🟢 Connection successful! Populating knowledge graph...
✅ Graph loaded with 8 entities and 7 relationships!
```

```
1 from pyvis.network import Network
2
3 # Create a PyVis network
4 net = Network(height="750px", width="100%", notebook=True)
5
```

Neo4j Aura console interface showing a query execution. The left sidebar contains navigation options: Developer hub, Data services, Instances, Import, Graph Analytics, Data APIs, Tools (Query, Explore), Operations (Metrics, Logs), Project (Users, Billing, Settings, Learning).

The main area displays the query editor and results. The query executed is:

```
1 MATCH (a)-[r]->(b) RETURN a, r, b LIMIT 100
2
```

The results are shown in a graph visualization mode, displaying a network of nodes and relationships. The nodes are labeled with names like "Orion Strategic", "Aureus Capital", "Pinnacle", "On Corp", "Sengora", "Terra", and "Pinnacle". The graph is visualized with green nodes and orange relationships.

The bottom section shows a table view of the results, with columns for entity and name.

Neo4j Aura console interface showing the database information and query execution. The left sidebar contains navigation options: Developer hub, Data services, Instances, Import, Graph Analytics, Data APIs, Tools (Query, Explore), Operations (Metrics, Logs), Project (Users, Billing, Settings, Learning).

The main area displays the query editor and results. The query executed is:

```
1 MATCH (a)-[r]->(b) RETURN a, r, b LIMIT 100
2
```

The results are shown in a graph visualization mode, displaying a network of nodes and relationships. The nodes are labeled with names like "Orion Strategic", "Aureus Capital", "Pinnacle", "On Corp", "Sengora", "Terra", and "Pinnacle". The graph is visualized with green nodes and orange relationships.

The bottom section shows a table view of the results, with columns for entity and name.

The database information section on the left shows the following details:

- Nodes (8): company, location
- Relationships (0)
- Property keys: data, id, name, nodes, relationships, style, type, visualisation

Last update: 1:00:31 AM

KG KnowledgeGraphs

main

Current File

⏏ ⚙ ⋮

🔍 👤

Project

KnowledgeGraphs C:\User

GraphRag

earnings_report.txt

local_rag.ipynb

.gitignore

LICENSE

README.md

External Libraries

Scratches and Consoles

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

In March 2025, the Financial Crimes Enforcement Network (FinCEN), in collaboration with Interpol, uncovered a sophisticated money laundering operation involving Orion Strategic Holdings and its subsidiaries.

At the center of the operation was Alexander Volkov, a dual citizen of Russia and Cyprus, who served as the Chief Executive Officer of Orion Strategic Holdings.

Over a 4-year period (2021-2025), Orion Strategic Holdings funneled over \$1.2 billion USD through a network of 15 entities, including:

Helios Global Ltd. (registered in the British Virgin Islands),

Tianyu Ventures (Hong Kong),

Starlake Investments (Luxembourg),

Juno Capital SA (Uruguay).

These entities used complex layers of cross-border wire transfers, real estate acquisitions, and cryptocurrency transactions to move funds.

One notable transaction involved the purchase of a \$95 million luxury hotel in Dubai, listed under Celestial Holdings.

Parallel investigations by the UK's Serious Fraud Office (SFO) and the Monetary Authority of Singapore (MAS) revealed that the laundered funds were then routed through crypto wallets associated with the TornadoShield protocol, a decentralized privacy solution.

In January 2025, whistleblower emails leaked from TriMark Bank (Liechtenstein) exposed internal memos approving the use of these wallets for Orion Strategic Holdings.

Legal and Enforcement Actions:

KG KnowledgeGraphs

main

Current File

⏏ ⚙ ⋮

🔍 👤

Project

KnowledgeGraphs C:\User

GraphRag

earnings_report.txt

local_rag.ipynb

.gitignore

LICENSE

README.md

External Libraries

Scratches and Consoles

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

In April 2025, Alexander Volkov was arrested in Monte Carlo by Interpol agents. He is facing charges of money laundering and fraud.

Orion Strategic Holdings is under a global asset freeze initiated by the EU Court of Justice.

Bayfront Maritime Logistics has lost its license from the Panama Maritime Authority.

TriMark Bank is facing a \$300 million fine from FINMA (Swiss Financial Market Supervisory Authority) for gross negligence in handling Orion Strategic Holdings' funds.

Over \$520 million in assets have been recovered so far, including cryptocurrency wallets, Swiss bank accounts, and real estate.

Technologies Used in the Investigation:

Knowledge graph systems developed by PaladinGraph AI were instrumental in mapping entity relationships.

NLP algorithms scanned over 200,000 emails and contracts, extracting connections between persons, companies, and locations.

Link prediction models surfaced previously unknown connections between TriMark Bank, Bayfront Maritime, and Tianyu Ventures.

Visual graph analytics revealed that the majority of suspicious transactions spiked shortly after geopolitical tensions between Russia and the West.