A router query engine is the decision-making component of the agent that decides which data source or tool is the best fit for a given query.

- Ensures that the query is directed to the right source
- Minimizing query time by avoiding unnecessary routing.

```
!pip install -q llama_index llama-index-readers-web llama-index-tools-google llama-index-embeddings-huggingface llama-index-llms-anthropic
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.5/56.5 kB 2.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
  Preparing metadata (setup.py) ... done
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.4/7.4 MB 17.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
  Preparing metadata (setup.py) ... done
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.9/72.9 kB 2.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 862.7/862.7 kB 18.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 37.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 25.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.1/211.1 kB 8.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 21.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 37.9/37.9 MB 18.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.6/9.6 MB 36.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 249.1/249.1 kB 11.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.3/81.3 kB 2.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.4/76.4 kB 2.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 3.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 318.9/318.9 kB 13.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 187.4/187.4 kB 5.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 861.9/861.9 kB 19.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 373.5/373.5 kB 9.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 295.8/295.8 kB 13.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 29.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.6/97.6 kB 5.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 476.0/476.0 kB 15.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.3/49.3 kB 2.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 2.6 MB/s eta 0:00:00
  Building wheel for html2text (setup.py) ... done
  Building wheel for tinysegmenter (setup.py) ... done
  Building wheel for spider-client (setup.py) ... done
  Building wheel for feedfinder2 (setup.py) ... done
  Building wheel for jieba3k (setup.py) ... done
  Building wheel for sgmllib3k (setup.py) ... done
```

```
from urllib.request import urlretrieve

urlretrieve("https://arxiv.org/pdf/2312.10997.pdf", "2312.10997.pdf")
```

```
('2312.10997.pdf', <http.client.HTTPMessage at 0x7d1909151e70>)
```

```
from llama_index.core import SimpleDirectoryReader

paper_documents = SimpleDirectoryReader(input_files=["2312.10997.pdf"]).load_data()
```

```
paper_documents[0].text[:300]
```

```
'1\nRetrieval-Augmented Generation for Large\nLanguage Models: A Survey\nYunfan Gaoa, Yun Xiongb, Xinyu Gaob, Kangxiang Jiab, Jinliu Pa
```

```
from llama_index.readers.web import SimpleWebPageReader

recipe_documents = SimpleWebPageReader(html_to_text=True).load_data(["https://tasty.co/recipe/chicken-gyros"])
```

```
recipe_documents[0].text[10000:12000]
```

```
'_00001.jpg?output-\nformat=auto&output-quality=auto&resize=600:*)\n\n##### Total Time\n\n3 hr 30 min\n\n3 hr 30 min\n\n##### Prep Time
\n\n20 minutes\n\n20 min\n\n##### Cook Time\n\n1 hr 30 min\n\n1 hr 30 min\n\n## Ingredients\n\nfor 8 servings\n\nMarinade\n\n  * 2 cups
plain full-fat greek yogurt (570 g)\n  * ¾ cup lemon juice (60 mL)\n  * ¾ cup olive oil (180 mL)\n  * 1 tablespoon kosher salt\n  * 1 t
ablespoon minced garlic\n  * 1 tablespoon ground coriander\n  * 1 tablespoon paprika\n  * 1 tablespoon ground cumin\n  * ½ teaspoon cay
enne pepper\n  * 1 teaspoon cinnamon\n  * 1 teaspoon freshly ground black pepper\n  * 2 lb boneless, skinless chicken thighs (910 g), p
```

```
from llama_index.core import Settings

Settings.chunk_size = 500
```

```
paper_nodes = Settings.node_parser.get_nodes_from_documents(paper_documents)
recipe_nodes = Settings.node_parser.get_nodes_from_documents(recipe_documents)
```

```
from llama_index.core import VectorStoreIndex
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.llms.anthropic import Anthropic
from google.colab import userdata

anthropic_api_key = userdata.get('ANTHROPIC_API_KEY')

embed_model = HuggingFaceEmbedding(
    model_name="BAAI/bge-small-en-v1.5"
)
llm = Anthropic(model="claude-3-5-sonnet-20240620", api_key=anthropic_api_key)

paper_vector_index = VectorStoreIndex(paper_nodes, embed_model=embed_model)
recipe_vector_index = VectorStoreIndex(recipe_nodes, embed_model=embed_model)

paper_query_engine = paper_vector_index.as_query_engine(llm=llm)
recipe_query_engine = recipe_vector_index.as_query_engine(llm=llm)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%  349/349 [00:00<00:00, 19.0kB/s]
config_sentence_transformers.json: 100%  124/124 [00:00<00:00, 7.55kB/s]
README.md: 100%  94.8k/94.8k [00:00<00:00, 3.79MB/s]
sentence_bert_config.json: 100%  52.0/52.0 [00:00<00:00, 2.76kB/s]
config.json: 100%  743/743 [00:00<00:00, 42.7kB/s]
model.safetensors: 100%  133M/133M [00:01<00:00, 97.5MB/s]
tokenizer_config.json: 100%  366/366 [00:00<00:00, 7.56kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 2.26MB/s]
tokenizer.json: 100%  711k/711k [00:00<00:00, 5.93MB/s]
special_tokens_map.json: 100%  125/125 [00:00<00:00, 3.78kB/s]
1_Pooling/config.json: 100%  190/190 [00:00<00:00, 5.85kB/s]
```

```
from llama_index.tools.google import GoogleSearchToolSpec
import json

google_search_api_key = userdata.get('GOOGLE_SEARCH_API_KEY')
google_search_engine = userdata.get('GOOGLE_SEARCH_ENGINE')
google_search_tool = GoogleSearchToolSpec(key=google_search_api_key, engine=google_search_engine)

test_results = google_search_tool.google_search("potato")
print(json.loads(test_results[0].text)["queries"]["request"][0]["totalResults"])
```

```
1040000000
```

```
from llama_index.core.query_engine import CustomQueryEngine
from llama_index.core.retrievers import BaseRetriever
from llama_index.core.response_synthesizers import BaseSynthesizer
from llama_index.core import PromptTemplate

qa_prompt = PromptTemplate(
    "Context information is below.\n"
    "---------------------\n"
    "{context_str}\n"
    "---------------------\n"
    "Given the context information and not prior knowledge, "
    "answer the query.\n"
    "Query: {query_str}\n"
    "Answer: "
)
```

```
class GoogleSearchQueryEngine(CustomQueryEngine):
    """Google Search Query Engine."""

    llm: Anthropic
    tool: GoogleSearchToolSpec

    def custom_query(self, query_str: str):
        response = self.tool.google_search(query_str)
        response_obj = json.loads(response[0].text)
        context_str = "\n\n".join([n["snippet"] for n in response_obj["items"][0:5]])
        output = self.llm.complete(
            qa_prompt.format(context_str=context_str, query_str=query_str)
        )
        return str(output)


from llama_index.core.query_engine import RouterQueryEngine
from llama_index.core.selectors import LLMSingleSelector
from llama_index.core.tools import QueryEngineTool


paper_vector_tool = QueryEngineTool.from_defaults(
    query_engine=paper_query_engine,
    description="Useful for retrieving information about Retrieval Augmented Generation or RAG techniques",
)
recipe_vector_tool = QueryEngineTool.from_defaults(
    query_engine=recipe_query_engine,
    description="Useful for retrieving information about cooking recipes",
)
google_query_engine = GoogleSearchQueryEngine(llm=llm, tool=google_search_tool)
google_tool = QueryEngineTool.from_defaults(
    query_engine=google_query_engine,
    description="Useful for retrieving information from the internet",
)

query_engine = RouterQueryEngine(
    selector=LLMSingleSelector.from_defaults(llm=llm),
    query_engine_tools=[
        paper_vector_tool,
        recipe_vector_tool,
        google_tool
    ],
    llm=llm
)


result = query_engine.query("Explain Modular RAG in one paragraph")
result
```

```
Response(response='Modular RAG represents an evolution in the Retrieval-Augmented Generation (RAG) approach, offering greater
flexibility and adaptability compared to its predecessors. This architecture incorporates various strategies to enhance its
components, such as introducing a search module for similarity searches and refining the retriever through fine-tuning. It supports
both sequential processing and integrated end-to-end training across its components. While building upon the principles of Advanced
and Naive RAG, Modular RAG introduces specialized modules like the Search module and RAG-Fusion to improve retrieval and processing
capabilities. These innovations allow for more efficient handling of diverse data sources and complex query scenarios. The overall
structure of Modular RAG is not limited to sequential retrieval and generation but includes methods such as iterative and adaptive
retrieval, making it a more versatile and powerful tool for information retrieval and generation tasks.', source_nodes=
[NodeWithScore(node=TextNode(id_='1fef4c5f-26e0-43bc-87fb-36847581a1ff', embedding=None, metadata={'page_label': '4', 'file_name':
'2312.10997.pdf', 'file_path': '2312.10997.pdf', 'file_type': 'application/pdf', 'file_size': 1662567, 'creation_date': '2024-09-
11', 'last_modified_date': '2024-09-11'}, excluded_embed_metadata_keys=['file_name', 'file_type', 'file_size', 'creation_date',
'last_modified_date', 'last_accessed_date'], excluded_llm_metadata_keys=['file_name', 'file_type', 'file_size', 'creation_date',
'last_modified_date', 'last_accessed_date'], relationships={<NodeRelationship.SOURCE: '1'>: RelatedNodeInfo(node_id='fe3e21d3-3e1f-
41ac-84d0-8b7198bc777c', node_type=<ObjectType.DOCUMENT: '4'>, metadata={'page_label': '4', 'file_name': '2312.10997.pdf',
'file_path': '2312.10997.pdf', 'file_type': 'application/pdf', 'file_size': 1662567, 'creation_date': '2024-09-11',
'last_modified_date': '2024-09-11'}, hash='8c0de5ef5721890c81ea91db9da60c58b65496c6e40726abceb30d56ae0deee3'),
<NodeRelationship.PREVIOUS: '2'>: RelatedNodeInfo(node_id='bbb9ca7c-615a-4589-a1a1-00867600d124', node_type=<ObjectType.TEXT: '1'>,
metadata={'page_label': '4', 'file_name': '2312.10997.pdf', 'file_path': '2312.10997.pdf', 'file_type': 'application/pdf',
'file_size': 1662567, 'creation_date': '2024-09-11', 'last_modified_date': '2024-09-11'},
hash='20a72f85802029222a9d5ebe9887249c168b9325c5a7bf48babb91b984882220')}, text='The main\nmethods in post-retrieval process include
rerank chunks and\ncontext compressing. Re-ranking the retrieved information to\nrelocate the most relevant content to the edges of
the prompt is\na key strategy. This concept has been implemented in frame-\nworks such as LlamaIndex2, LangChain3, and HayStack
[12].\nFeeding all relevant documents directly into LLMs can lead\nto information overload, diluting the focus on key details
with\nirrelevant content.To mitigate this, post-retrieval efforts con-\ncentrate on selecting the essential information,
emphasizing\ncritical sections, and shortening the context to be
processed.\n2https://www.llamaindex.ai\n3https://www.langchain.com/C. Modular RAG\nThe modular RAG architecture advances beyond the
for-\nmer two RAG paradigms, offering enhanced adaptability and\nversatility. It incorporates diverse strategies for improving
its\ncomponents, such as adding a search module for similarity\nsearches and refining the retriever through fine-tuning. Inno-
\nvations like restructured RAG modules [13] and rearranged\nRAG pipelines [14] have been introduced to tackle specific\nchallenges.
The shift towards a modular RAG approach is\nbecoming prevalent, supporting both sequential processing and\nintegrated end-to-end
training across its components. Despite\nits distinctiveness, Modular RAG builds upon the foundational\nprinciples of Advanced and
```

```
Naive RAG, illustrating a progres-\nsion and refinement within the RAG family.\n1) New Modules: The Modular RAG framework
introduces\nadditional specialized components to enhance retrieval and\nprocessing capabilities. The Search module adapts to spe-
\ncific scenarios, enabling direct searches across various data\nsources like search engines, databases, and knowledge
graphs,\nusing LLM-generated code and query languages [15]. RAG-\nFusion addresses traditional search limitations by employing\na
multi-query strategy that expands user queries into diverse\nperspectives, utilizing parallel vector searches and intelligent\nre-
ranking to uncover both explicit and transformative knowl-\nedge [16]. The Memory module leverages the LLM's memory\nto guide
retrieval, creating an unbounded memory pool that', mimetype='text/plain', start_char_idx=1404, end_char_idx=3575,
text_template='{metadata_str}\n\n{content}', metadata_template='{key}: {value}', metadata_seperator='\n'),
score=0.7536143632338909), NodeWithScore(node=TextNode(id_='bbb9ca7c-615a-4589-a1a1-00867600d124', embedding=None, metadata=
{'page_label': '4', 'file_name': '2312.10997.pdf', 'file_path': '2312.10997.pdf', 'file_type': 'application/pdf', 'file_size':
1662567, 'creation_date': '2024-09-11', 'last_modified_date': '2024-09-11'}, excluded_embed_metadata_keys=['file_name', 'file_type',
'file_size', 'creation_date', 'last_modified_date', 'last_accessed_date'], excluded_llm_metadata_keys=['file_name', 'file_type',
'file_size', 'creation_date', 'last_modified_date', 'last_accessed_date'], relationships={<NodeRelationship.SOURCE: '1'>:
RelatedNodeInfo(node_id='fe3e21d3-3e1f-41ac-84d0-8b7198bc777c', node_type=<ObjectType.DOCUMENT: '4'>, metadata={'page_label': '4',
'file_name': '2312.10997.pdf', 'file_path': '2312.10997.pdf', 'file_type': 'application/pdf', 'file_size': 1662567, 'creation_date':
'2024-09-11', 'last_modified_date': '2024-09-11'}, hash='8c0de5ef5721890c81ea91db9da60c58b65496c6e40726abceb30d56ae0deee3'),
<NodeRelationship.NEXT: '3'>: RelatedNodeInfo(node_id='1fef4c5f-26e0-43bc-87fb-36847581a1ff', node_type=<ObjectType.TEXT: '1'>,
metadata={}, hash='04e873eb2809becbe3c19043fca20d791961d216ca0683a1c22ed8ffd5d95842')}, text='4\nFig. 3. Comparison between the
three paradigms of RAG. (Left) Naive RAG mainly consists of three parts: indexing, retrieval and generation. (Middle)\nAdvanced RAG
proposes multiple optimization strategies around pre-retrieval and post-retrieval, with a process similar to the Naive RAG, still
following a\nchain-like structure. (Right) Modular RAG inherits and develops from the previous paradigm, showcasing greater
flexibility overall. This is evident in the\nintroduction of multiple specific functional modules and the replacement of existing
modules. The overall process is not limited to sequential retrieval and\ngeneration; it includes methods such as iterative and
adaptive retrieval.\nPre-retrieval process . In this stage, the primary focus is\non optimizing the indexing structure and the
original query.\nThe goal of optimizing indexing is to enhance the quality of\nthe content being indexed. This involves strategies:
enhancing\ndata granularity, optimizing index structures, adding metadata,\nalignment optimization, and mixed retrieval. While the
```

```
result = query_engine.query("What ingredients do I need to make chicken gyros?")
result
```

```
Response(response="To make chicken gyros, you'll need several ingredients for the marinade, tzatziki sauce, and serving.\n\nFor the
marinade, gather Greek yogurt, lemon juice, olive oil, kosher salt, minced garlic, ground coriander, paprika, ground cumin, cayenne
pepper, cinnamon, black pepper, and boneless, skinless chicken thighs.\n\nThe tzatziki sauce requires cucumber, Greek yogurt, minced
garlic, lemon juice, fresh dill, fresh parsley, kosher salt, black pepper, and yellow onion.\n\nFor serving, you'll need pita breads,
sliced onion, and sliced tomato.\n\nAdditionally, you'll want to have a sturdy 10-inch wooden skewer on hand as part of the special
equipment needed for this recipe.", source_nodes=[NodeWithScore(node=TextNode(id_='d5191c57-c5c1-4f45-8738-35119bacfce7',
embedding=None, metadata={}, excluded_embed_metadata_keys=[], excluded_llm_metadata_keys=[], relationships={<NodeRelationship.SOURCE:
'1'>: RelatedNodeInfo(node_id='https://tasty.co/recipe/chicken-gyros', node_type=<ObjectType.DOCUMENT: '4'>, metadata={},
hash='a8906e2a9458bd4707f8f31387dcade31e57c0f1be2807adfb450cf24081737d'), <NodeRelationship.PREVIOUS: '2'>:
RelatedNodeInfo(node_id='9332ae40-f93f-4a55-a34a-1764a63d8a31', node_type=<ObjectType.TEXT: '1'>, metadata={},
hash='c7024b3318ac0868307d36eb2641053e846368349ef11baa0fb823f70b3b358f'), <NodeRelationship.NEXT: '3'>:
RelatedNodeInfo(node_id='13366ddb-3f4c-4124-8f8c-2e4e0e5e0e4c', node_type=<ObjectType.TEXT: '1'>, metadata={},
hash='db4b5e33a489f07c4da029f0116d6ff6681d06dea42ea8fc5eaf2f831e522adc')}, text='Succulent\nchicken thighs are bathed in a lively
marinade bursting with the flavors of\nlemon, garlic, and fragrant oregano, then grilled to juicy perfection. Nestled\nin warm pita
bread and adorned with crisp veggies and a dollop of creamy\ntzatziki sauce, each bite is a mouthful of Mediterranean flavors. This
simple\nrecipe lets you bypass a trip to a Greek restaurant and instead whip up some\ndelicious gyros right in the comfort of your own
kitchen. This recipe is sure\nto be a hit with your family. Make it a weekly favorite!\n\nMike Price\n\nTasty Team\n\nUpdated on May
13, 2024\n\n96% would make again\n\n  * [Share via facebook](https://www.facebook.com/dialog/share?
href=https%3A%2F%2Ftasty.co%2Frecipe%2Fchicken-gyros&app_id=1060852764021442)\n  * [Share via pinterest]
(https://pinterest.com/pin/create/button/?url=https%3A%2F%2Ftasty.co%2Frecipe%2Fchicken-
gyros&description=Chicken%20Gyros&media=https%3A%2F%2Fimg.buzzfeed.com%2Fvideo-api-
prod%2Fassets%2Fa360484107084401add1475402287908%2F1504739525_00001.jpg)\n  * [Share via email](mailto:?subject=Chicken
Gyros&body=https://tasty.co/recipe/chicken-gyros)\n  * [Share via sms](sms:?&body=Chicken Gyros - https://tasty.co/recipe/chicken-
gyros)\n  * Print\n\n##### Total Time\n\n3 hr 30 min\n\n3 hr 30 min\n\n##### Prep Time\n\n20 minutes\n\n20 min\n\n##### Cook Time\n\n1
hr 30 min\n\n1 hr 30 min\n\n!', mimetype='text/plain', start_char_idx=8565, end_char_idx=9893,
text_template='{metadata_str}\n\n{content}', metadata_template='{key}: {value}', metadata_seperator='\n'), score=0.7725959922826776),
NodeWithScore(node=TextNode(id_='13366ddb-3f4c-4124-8f8c-2e4e0e5e0e4c', embedding=None, metadata={}, excluded_embed_metadata_keys=[],
excluded_llm_metadata_keys=[], relationships={<NodeRelationship.SOURCE: '1'>: RelatedNodeInfo(node_id='https://tasty.co/recipe/chicken-
gyros', node_type=<ObjectType.DOCUMENT: '4'>, metadata={}, hash='a8906e2a9458bd4707f8f31387dcade31e57c0f1be2807adfb450cf24081737d'),
<NodeRelationship.PREVIOUS: '2'>: RelatedNodeInfo(node_id='d5191c57-c5c1-4f45-8738-35119bacfce7', node_type=<ObjectType.TEXT: '1'>,
metadata={}, hash='e09c1bad26f512e7c5cf5a9f4e7ab3b918c42a43793a3f52b9a8e977a7429235'), <NodeRelationship.NEXT: '3'>:
RelatedNodeInfo(node_id='68123d4a-3b8c-40f9-bd0e-e16daf50d611', node_type=<ObjectType.TEXT: '1'>, metadata={},
hash='647f6faab881db4669c32f728c9a999e2bce7ba449844687f67e71e19e64c978')}, text='[Chicken Gyros](https://img.buzzfeed.com/video-api-
\nprod/assets/a360484107084401add1475402287908/1504739525_00001.jpg?output-\nformat=auto&output-quality=auto&resize=600:*)\n\n#####
Total Time\n\n3 hr 30 min\n\n3 hr 30 min\n\n##### Prep Time\n\n20 minutes\n\n20 min\n\n##### Cook Time\n\n1 hr 30 min\n\n1 hr 30
min\n\n## Ingredients\n\nfor 8 servings\n\nMarinade\n\n  * 2 cups plain full-fat greek yogurt (570 g)\n  * ¼ cup lemon juice (60 mL)\n
* ¾ cup olive oil (180 mL)\n  * 1 tablespoon kosher salt\n  * 1 tablespoon minced garlic\n  * 1 tablespoon ground coriander\n  * 1
tablespoon paprika\n  * 1 tablespoon ground cumin\n  * ½ teaspoon cayenne pepper\n  * 1 teaspoon cinnamon\n  * 1 teaspoon freshly
ground black pepper\n  * 2 lb boneless, skinless chicken thighs (910 g), pounded flat\n\nTzatziki Sauce\n\n  * 1 large cucumber,
shredded\n  * 2 cups plain full-fat greek yogurt (570 g)\n  * 1 tablespoon minced garlic\n  * ¼ cup lemon juice (60 mL)\n  * 2
tablespoons finely chopped fresh dill\n  * 2 tablespoons finely chopped fresh parsley\n  * kosher salt, to taste\n  * freshly ground
black pepper, to taste\n  * ½ large yellow onion, peeled and root end removed\n\nFor Serving\n\n  * 8 pita breads\n  * sliced onion\n
* sliced tomato\n\nSpecial Equipment\n\n  * 1 sturdy 10-inch wooden skewer\n\n## Nutrition Info\n\nView Info\n\n  * Calories 510\n  *
Fat 15g\n  * Carbs 50g\n  * Fiber 3g\n  * Sugar 17g\n  * Protein 43g\n\nEstimated values based on one serving size.\n\n##
Preparation\n\n  1. Make the marinade: In a large bowl, combine the yogurt, lemon juice, olive oil, salt, garlic, coriander, paprika,
cumin, cayenne, cinnamon, and black pepper and stir well.\n  2.', mimetype='text/plain', start_char_idx=9893, end_char_idx=11500,
text_template='{metadata_str}\n\n{content}', metadata_template='{key}: {value}', metadata_seperator='\n'), score=0.76051880140087)],
metadata={'d5191c57-c5c1-4f45-8738-35119bacfce7': {}, '13366ddb-3f4c-4124-8f8c-2e4e0e5e0e4c': {}, 'selector_result':
MultiSelection(selections=[SingleSelection(index=1, reason="The question 'What ingredients do I need to make chicken gyros?' is
directly related to cooking recipes. Choice 2 specifically mentions being useful for retrieving information about cooking recipes,
making it the most relevant option for answering this question.")])})
```

As we can see, the chicken gyros recipe vector store was correctly chosen to answer that question.

Finally, let's ask it a question that can be answered with a Google Search.

```
result = query_engine.query("How tall is the Eiffel Tower?")
result
```

```
Response(response="According to the context information provided, the Eiffel Tower is 330 metres (1,083 ft) tall. This is equivalent to the height of an 81-storey building, and it is described as the tallest structure in Paris. \n\nIt's worth noting that there is a slight discrepancy in the information provided, as one source mentions a height of 984 feet. However, the more specific measurement of 330 metres (1,083 ft) is likely the more accurate and up-to-date figure.\n\nAdditionally, the context mentions that 6 meters were recently added to the tower's height due to the installation of a new antenna for digital terrestrial radio. This suggests that the current height might be slightly greater than 330 metres, but an exact updated measurement is not provided in the given information.", source_nodes=[], metadata={'selector_result': MultiSelection(selections=[SingleSelection(index=2, reason="The question 'How tall is the Eiffel Tower?' requires retrieving factual information from a general knowledge source, which is most likely to be found on the internet. Option 3 specifically mentions retrieving information from the internet, making it the most relevant choice for answering this question.")])})
```