```
!pip install -qqq --upgrade pip --progress-bar off


!pip install -qqq langchain-groq==0.1.3 --progress-bar off
!pip install -qqq langchain==0.1.17 --progress-bar off
!pip install -qqq llama-parse==0.1.3 --progress-bar off
!pip install -qqq qdrant-client==1.9.1 --progress-bar off

!pip install -qqq "unstructured[md]" --progress-bar off
!pip install -qqq fastembed==0.2.7 --progress-bar off
!pip install -qqq flashrank==0.2.4 --progress-bar off
```

```
import os
os.environ["GROQ_API_KEY"] = 'my-groq-api'
```

```
!pip uninstall -y numpy
!pip install numpy==1.26.4
```

```
Found existing installation: numpy 1.26.4
Uninstalling numpy-1.26.4:
  Successfully uninstalled numpy-1.26.4
Collecting numpy==1.26.4
  Using cached numpy-1.26.4-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
Using cached numpy-1.26.4-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.0 MB)
Installing collected packages: numpy
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
xarray 2025.10.1 requires packaging>=24.1, but you have packaging 23.2 which is incompatible.
gradio 5.49.1 requires huggingface-hub<2.0,>=0.33.5, but you have huggingface-hub 0.20.3 which is incompatible.
tensorflow 2.19.0 requires protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3, but you have protob
transformers 4.57.1 requires huggingface-hub<1.0,>=0.34.0, but you have huggingface-hub 0.20.3 which is incompatible.
transformers 4.57.1 requires tokenizers<=0.23.0,>=0.22.0, but you have tokenizers 0.15.2 which is incompatible.
datasets 4.0.0 requires huggingface-hub>=0.24.0, but you have huggingface-hub 0.20.3 which is incompatible.
opencv-contrib-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 1.26.4 which is incompatib
peft 0.17.1 requires huggingface_hub>=0.25.0, but you have huggingface-hub 0.20.3 which is incompatible.
opencv-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 1.26.4 which is incompatible.
accelerate 1.11.0 requires huggingface_hub>=0.21.0, but you have huggingface-hub 0.20.3 which is incompatible.
opencv-python-headless 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 1.26.4 which is incompati
db-dtypes 1.4.3 requires packaging>=24.2.0, but you have packaging 23.2 which is incompatible.
jax 0.7.2 requires numpy>=2.0, but you have numpy 1.26.4 which is incompatible.
pytensor 2.35.1 requires numpy>=2.0, but you have numpy 1.26.4 which is incompatible.
diffusers 0.35.2 requires huggingface-hub>=0.34.0, but you have huggingface-hub 0.20.3 which is incompatible.
jaxlib 0.7.2 requires numpy>=2.0, but you have numpy 1.26.4 which is incompatible.
Successfully installed numpy-1.26.4
WARNING: The following packages were previously imported in this runtime:
  [numpy]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
import os
import textwrap
from pathlib import Path
from google.colab import userdata
from IPython.display import Markdown
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.retrievers import ContextualCompressionRetriever
from langchain.retrievers.document_compressors import FlashrankRerank
from langchain_community.document_loaders import UnstructuredMarkdownLoader
from langchain_community.embeddings.fastembed import FastEmbedEmbeddings
from langchain_community.vectorstores import Qdrant
from langchain_core.prompts import ChatPromptTemplate
from langchain_groq import ChatGroq
from llama_parse import LlamaParse


# this function formats and prints the response text to ensure it's easily readable

def print_response(response):
    """Print the response in a formatted way"""
    if isinstance(response, dict):
        if 'result' in response:
            # Use fill() instead of TextWrapper
            wrapped_text = textwrap.fill(response['result'], width=80)
            print(wrapped_text)
        if 'source_documents' in response:
            print("\n\nSources:")
            for doc in response['source_documents']:
                print(f"- {doc.metadata.get('source', 'Unknown')}")
    else:
        print(response)
```

Document Parsing

```
# Set up LlamaParse for parsing the PDF document

instruction = """The provided document is a research on India education.
It contains many graphs and tables.
Try to be precise whie answering the quistions"""
```

```
parser = LlamaParse(
    api_key = 'my-llama-api',
    result_type = "markdown",
    parsing_instruction = instruction,
    max_timeout = 5000,

)
```

```
llama_parse_documents = await parser.aload_data("/content/Fundamentals of Data Engineering.pdf")
```

```
Started parsing the file under job_id 542d0486-d4b5-449e-9024-e860db183711
```

```
parsed_doc = llama_parse_documents[0]
```

Double-click (or enter) to edit

```
Markdown(parsed_doc.text[:4096])
```

Fundamentals of Data Engineering

Plan and Build Robust Data Systems

Joe Reis & Matt Housley

Fundamentals of Data Engineering

Data engineering has grown rapidly in the past decade, leaving many software engineers, data scientists, and analysts looking for a comprehensive view of this practice. With this practical book, you'll learn how to plan and build systems to serve the needs of your organization and customers by evaluating the best technologies available through the framework of the data engineering lifecycle. Authors Joe Reis and Matt Housley walk you through the data engineering lifecycle and show you how to stitch together a variety of cloud technologies to serve the needs of downstream data consumers. You'll understand how to apply the concepts of data generation, ingestion, orchestration, transformation, storage, and governance that are critical in any data environment regardless of the underlying technology.

This book will help you:

- Get a concise overview of the entire data engineering landscape
- Assess data engineering problems using an end-to-end framework of best practices
- Cut through marketing hype when choosing data technologies, architecture, and processes
- Use the data engineering lifecycle to design and build a robust architecture
- Incorporate data governance and security across the data engineering lifecycle

Joe Reis is a "recovering data scientist," and a data engineer and architect. Matt Housley is a data engineering consultant and cloud specialist.

"The world of data has been evolving for a while now. First there were designers. Then database administrators. Then CIOs. Then data architects. This book signals the next step in the evolution and maturity of the industry. It is a must read for anyone who takes their profession and career honestly."

—Bill Inmon, creator of the data warehouse

"Fundamentals of Data Engineering is a great introduction to the business of moving, processing, and handling data. I'd highly recommend it for anyone wanting to get up to speed in data engineering or analytics, or for existing practitioners who want to fill in any gaps in their understanding."

—Jordan Tigani, founder and CEO, MotherDuck, and founding engineer and cocreator of BigQuery

DATA

Twitter: @oreillymedia

linkedin.com/company/oreilly-media

US $69.99 CAN$87.99

youtube.com/oreillymedia

ISBN: 978 1 098 10830 4

Praise for Fundamentals of Data Engineering

The world of data has been evolving for a while now. First there were designers. Then database administrators. Then CIOs. Then data architects. This book signals the next step in the evolution and maturity of the industry. It is a must read for anyone who takes their profession and career honestly.

—Bill Inmon, creator of the data warehouse

Fundamentals of Data Engineering is a great introduction to the business of moving, processing, and handling data. It explains the taxonomy of data concepts, without focusing too heavily on individual tools or vendors, so the techniques and ideas should outlast any individual trend or product. I'd highly recommend it for anyone wanting to get up to speed in data engineering or analytics, or for existing practitioners who want to fill in any gaps in their understanding.

—Jordan Tigani, founder and CEO, MotherDuck, and founding engineer and cocreator of BigQuery

If you want to lead in your industry, you must build the capabilities required to provide exceptional customer and employee experiences. This is not just a technology problem. It's a people opportunity. And it will transform your business. Data engineers are at the center of this transformation. But today the discipline is misunderstood. This book will demystify data engineering and become your ultimate guide to succeeding with data.

—Bruno Aziza, Head of Data Analytics, Google Cloud

What a book! Joe and Matt are giving you the answer to the question, "What must I understand to do data engineering?" Whether you are getting started as a data engineer or strengthening your skills, you are not looking for yet

```
document_path = Path("/content/parsed_document.md")
with document_path.open("a") as f:
    f.write(parsed_doc.text)
```

```
loader = UnstructuredMarkdownLoader(document_path)
loaded_documents = loader.load()
```

```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=2048, chunk_overlap=128)
docs = text_splitter.split_documents(loaded_documents)
len(docs)
```

```
541
```

```
print(docs[0].page_content)
```

```
Fundamentals of Data Engineering

Plan and Build Robust Data Systems
```

```
Joe Reis & Matt Housley

Fundamentals of Data Engineering

Data engineering has grown rapidly in the past decade, leaving many software engineers, data scientists, and analysts looking fo

This book will help you:

Get a concise overview of the entire data engineering landscape

Assess data engineering problems using an end-to-end framework of best practices

Cut through marketing hype when choosing data technologies, architecture, and processes

Use the data engineering lifecycle to design and build a robust architecture

Incorporate data governance and security across the data engineering lifecycle

Joe Reis is a “recovering data scientist,” and a data engineer and architect. Matt Housley is a data engineering consultant and

“The world of data has been evolving for a while now. First there were designers. Then database administrators. Then CIOs. Then

—Bill Inmon, creator of the data warehouse
```

```python
embeddings = FastEmbedEmbeddings(model_name="BAAI/bge-base-en-v1.5")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_token.py:88: 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
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(
Fetching 5 files: 100%                                  5/5 [00:01<00:00,  1.12s/it]

special_tokens_map.json: 100%                                  695/695 [00:00<00:00, 13.7kB/s]

config.json: 100%                                  740/740 [00:00<00:00, 14.9kB/s]

tokenizer.json:     711k/? [00:00<00:00, 6.01MB/s]

tokenizer_config.json:     1.24k/? [00:00<00:00, 12.5kB/s]

model_optimized.onnx: 100%                                  218M/218M [00:01<00:00, 118MB/s]
```

```python
qdrant = Qdrant.from_documents(
    docs,
    embeddings,
    # location=":memory:",
    path="./db",
    collection_name="document_embeddings",
)
```

```python
%%time
query = "What are the main stages of the data engineering lifecycle"
similar_docs = qdrant.similarity_search_with_score(query)
```

```
CPU times: user 87.1 ms, sys: 975 µs, total: 88.1 ms
Wall time: 86.7 ms
```

```python
for doc, score in similar_docs:
    print(f"text: {doc.page_content[:256]}\n")
    print(f"score: {score}")
    print("-" * 80)
    print()
```

```
text: In this chapter, you’ll learn about the data engineering lifecycle, which is the central theme of this book. The data engi

score: 0.8640435448351382
--------------------------------------------------------------------------------

text: How to use the data engineering lifecycle to design and build a robust architecture

Best practices for each stage of the data lifecycle

And you will be able to:

Incorporate data engineering principles in your current role (data scientist, analyst, softw
score: 0.8118895130962629
--------------------------------------------------------------------------------

text: The Data Lifecycle Versus the Data Engineering Lifecycle

You may be wondering about the difference between the overall data lifecycle and the data engineering lifecycle. There’s a subtl

score: 0.7974325658312166
--------------------------------------------------------------------------------

text: Several years ago, data engineering didn’t even exist as a field or job title. Now you’re reading a book called Fundamenta

score: 0.7922210029529297
--------------------------------------------------------------------------------
```

```
%%time
retriever = qdrant.as_retriever(search_kwargs={"k": 5})
retrieved_docs = retriever.invoke(query)
```

```
CPU times: user 104 ms, sys: 26 µs, total: 104 ms
Wall time: 114 ms
```

```
compressor = FlashrankRerank(model="ms-marco-MiniLM-L-12-v2")

# Initialize ContextualCompressionRetriever with the compressor and retriever

compression_retriever = ContextualCompressionRetriever(
    base_compressor=compressor, base_retriever=retriever
)
```

```
Downloading ms-marco-MiniLM-L-12-v2...
ms-marco-MiniLM-L-12-v2.zip: 100%|██████████| 21.6M/21.6M [00:00<00:00, 158MiB/s]
```

```
%%time
reranked_docs = compression_retriever.invoke(query)
len(reranked_docs)
```

```
Running pairwise ranking..
CPU times: user 1.94 s, sys: 160 ms, total: 2.1 s
Wall time: 2.02 s
3
```

```
for doc in reranked_docs:
    print(f"id: {doc.metadata['_id']}\n")
    print(f"text: {doc.page_content[:256]}\n")
    print(f"score: {doc.metadata['relevance_score']}")
    print("-" * 80)
    print()
```

```
id: d5fa724014c242ec8403b002e0f05bdc

text: In this chapter, you’ll learn about the data engineering lifecycle, which is the central theme of this book. The data engi

score: 0.9996446967124939
--------------------------------------------------------------------------------

id: 33ee819004014890b99cba475c9a242d

text: The Data Lifecycle Versus the Data Engineering Lifecycle

You may be wondering about the difference between the overall data lifecycle and the data engineering lifecycle. There’s a subtl

score: 0.9971945285797119
--------------------------------------------------------------------------------

id: 251cfa56b1784b81a0e23a3d4fd36c88

text: How to use the data engineering lifecycle to design and build a robust architecture

Best practices for each stage of the data lifecycle

And you will be able to:

Incorporate data engineering principles in your current role (data scientist, analyst, softw

score: 0.9930357933044434
--------------------------------------------------------------------------------
```

```python
llm = ChatGroq(temperature=0, model_name="llama-3.3-70b-versatile")
```

```python
from langchain.prompts import PromptTemplate
```

```python
prompt_template = """
Use the following pieces of information to answer the user's question.
If you don't know the answer, just say that you don't know, don't try to make up an answer.

Context: {context}
Question: {question}

Answer the question and provide additional helpful information,
based on the pieces of information, if applicable. Be succinct.

Responses should be properly formatted to be easily read.
"""

prompt = PromptTemplate(
    template=prompt_template, input_variables=["context", "question"]
)
```

```python
from langchain.chains import RetrievalQA
```

```python
qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=compression_retriever,
    return_source_documents=True,
    chain_type_kwargs={"prompt": prompt, "verbose": True},
)
```

```python
%%time
response = qa.invoke("What are the main stages of the data engineering lifecycle")
```

```
Running pairwise ranking..


> Entering new StuffDocumentsChain chain...


> Entering new LLMChain chain...
Prompt after formatting:

Use the following pieces of information to answer the user's question.
If you don't know the answer, just say that you don't know, don't try to make up an answer.

Context: In this chapter, you’ll learn about the data engineering lifecycle, which is the central theme of this book. The data

What Is the Data Engineering Lifecycle?

The data engineering lifecycle comprises stages that turn raw data ingredients into a useful end product, ready for consumptio

We divide the data engineering lifecycle into five stages (Figure 2-1, top):

Generation

Storage

Ingestion

Transformation

Serving data

Figure 2-1. Components and undercurrents of the data engineering lifecycle

We begin the data engineering lifecycle by getting data from source systems and storing it. Next, we transform the data and th

In general, the middle stages—storage, ingestion, transformation—can get a bit jumbled. And that’s OK. Although we split out tl

Acting as a bedrock are undercurrents (Figure 2-1, bottom) that cut across multiple stages of the data engineering lifecycle: :

The Data Lifecycle Versus the Data Engineering Lifecycle
```

```
You may be wondering about the difference between the overall data lifecycle and the data engineering lifecycle. There’s a sub

34 | Chapter 2: The Data Engineering Lifecycle

Figure 2-2. The data engineering lifecycle is a subset of the full data lifecycle

Generation: Source Systems

A source system is the origin of the data used in the data engineering lifecycle. For example, a source system could be an IoT

A major challenge in data engineering is the dizzying array of data source systems engineers must work with and understand. As

Figure 2-3. Source system example: an application database

How to use the data engineering lifecycle to design and build a robust architecture

Best practices for each stage of the data lifecycle
```

```
print_response(response)
```

```
**Main Stages of the Data Engineering Lifecycle:**  The data engineering
lifecycle is divided into five stages:  1. **Generation**: Getting data from
source systems. 2. **Storage**: Storing data, which occurs throughout the
lifecycle. 3. **Ingestion**: Ingesting data into the system. 4.
**Transformation**: Transforming the data into a useful format. 5. **Serving
data**: Serving the transformed data to analysts, data scientists, ML engineers,
and others.  **Additional Information:** These stages may not always occur in a
linear sequence and can overlap, repeat, or occur out of order. Additionally,
undercurrents such as security, data management, DataOps, data architecture,
orchestration, and software engineering support all stages of the data
engineering lifecycle.


Sources:
- /content/parsed_document.md
- /content/parsed_document.md
- /content/parsed_document.md
```

```
qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=compression_retriever,
    return_source_documents=True,
    chain_type_kwargs={"prompt": prompt, "verbose": False},
)
```

```
%%time
response = qa.invoke("What are the key differences between batch processing and stream processing in data engineering")
```

```
Running pairwise ranking..
CPU times: user 2.72 s, sys: 29.3 ms, total: 2.75 s
Wall time: 5.25 s
```

```
Markdown(response["result"])
```

Key Differences between Batch Processing and Stream Processing:

1. **Processing Frequency**: Batch processing occurs at scheduled intervals, whereas stream processing occurs in real-time or near real-time.
2. **Data Ingestion**: Batch processing typically involves ingesting data in large batches, while stream processing involves ingesting data as it is generated.
3. **Data Processing**: Batch processing involves processing data in batches, whereas stream processing involves processing data continuously as it flows in.
4. **Use Cases**: Batch processing is suitable for applications that require periodic processing, such as reporting and analytics, while stream processing is suitable for applications that require real-time processing, such as IoT sensor data and event-driven architectures.

Additional Considerations: