Intro to Retrieval-Augmented Generation (RAG)

A Beginner Lab for Enterprise Data Platforms

What this PPT pack covers:

- How RAG works end-to-end (documents ⟶ vector store ⟶ answers)
- Why enterprises separate indexing and querying
- How retrieval enables grounded, explainable AI

Retrieval-Augmented Generation (RAG)

- Course: Introduction to RAG for Beginners
- Goal: Understand how modern AI analyzes enterprise documents and answers questions
- Prerequisites: No coding required – focus on concepts and system behavior

What is RAG?

- RAG = Retrieval + Generation
- Retrieve relevant information from documents first
- Generate augmented answers using retrieved context only

Why RAG Exists

- LLMs memory does not include your private documents
- LLMs may hallucinate in confidence but give incorrect answers
- RAG grounds AI answers in real, retrievable data

Enterprise Use Case 1

- Regulated Knowledge Assistant (Risk, Compliance, legal, finance & other corporate functions)
- What RAG enables
  - AI answers **only from approved internal documents**
  - Every answer includes **citations** (page, policy, clause)
  - Governance controls who can ask what
- Business value
  - Faster regulatory responses
  - Reduced legal/compliance risk
  - Lower dependency on SMEs for routine queries
  - Executive confidence in AI-assisted decisions

Enterprise Use Case 2

- Enterprise Reporting & Explainability Layer (finance, strategy & Operations)
- What RAG enables
  - Natural-language explanations grounded in:
    - management commentary
    - board papers
    - analyst notes
  - Answers tied directly to **source documents**
  - Consistent interpretation across teams
- Business value
  - Faster executive insight
  - Better strategic decisions
  - Reduced misinterpretation of numbers
  - AI as an *explainability layer* over enterprise data

Enterprise Use Case 3

- Institutional Memory & Decision Intelligence (Management, Transformation, M&A)
- What RAG enables
  - Ask:
    “Why did we choose option A in 2021?”
    “What risks were identified?”
  - Answers grounded in **original decision artifacts**
  - AI becomes a **corporate memory system**
- Business value
  - Better long-term decision quality
  - Faster executive onboarding
  - Reduced strategic drift
  - Preservation of organizational knowledge

High-Level RAG Architecture

- Documents → Indexing → Vector Database
- User Question → Retrieval → LLM Answer
- Two distinct phases: Indexing and Querying

Indexing Phase (index.py)

- Runs offline or when documents change
- Reads documents and extracts text
- Splits text into chunks
- Creates embeddings and stores them in a vector database

Why Chunking Matters

- LLMs have input size limits
- Smaller chunks improve retrieval accuracy
- Industry systems retrieve chunks, not full documents

Embeddings Explained

- Embeddings convert text into numbers
- Similar meanings produce similar vectors
- Enables semantic (meaning-based) search

Vector Databases

- Store embeddings and metadata
- Enable fast similarity search
- Examples: Chroma, Pinecone, Weaviate, pgvector

Querying Phase (query.py)

- Runs every time a user asks a question
- Embeds the question
- Retrieves relevant chunks
- Uses an LLM to generate a grounded answer

Why Retrieval Comes First

- The model can only answer using retrieved data
- Better retrieval = better answers
- Retrieval quality matters more than model size

Metadata and Citations

- Metadata tracks source and page number
- Enables citations and trust
- Required for enterprise and compliance use cases

Industry Best Practices in RAG

- Separate indexing and querying
- Idempotent indexing (safe re-runs)
- Persistent vector databases
- Grounded answer generation

What You Built Conceptually

- A real-world RAG architecture
- A system that avoids hallucinations
- A foundation used in enterprise AI systems

Next Steps

- Visualization embeddings (optional function)
- Build use cases for governed RAG capabilities on private data platform
- Reranking and evaluation
- Deploy as an API service