LangChain Data-to-Answer Flow (RAG)

1. Load Data ■
Bring your data into LangChain as Document objects.
Common Loaders: TextLoader, CSVLoader, PyPDFLoader, WebBaseLoader
Example: loader = CSVLoader("data.csv")

2. Split Data ✂■
Break large documents into smaller, overlapping chunks for better LLM processing.
Common Splitter: RecursiveCharacterTextSplitter
chunk_size → max text per chunk
chunk_overlap → repeated text between chunks for context

3. Create Embeddings ■
Convert chunks into numeric vector representations of meaning.
Popular: OpenAIEmbeddings, HuggingFaceEmbeddings

4. Store in Vector DB ■
Save embeddings for semantic search.
Popular: FAISS, Chroma, Pinecone, Weaviate

5. Retrieve Relevant Chunks ■
Find the most semantically similar documents to the user's query.

6. Pass to LLM ■
Give retrieved chunks to the LLM to answer with context.

Full Flow: Loader → Splitter → Embeddings → Vector Store → Retrieval → LLM Answer

Loader	→	Splitter	→	Embeddings	→	Vector Store	→	Retrieval	→	LLM Answer