Setting up Python

Steps

Install Python

- https://www.python.org/downloads/

Install Jupyter Notebook

- pip install notebook
- pip install jupyterlab

Open Jupyter notebook

- Understanding default directory
- Running a line of code

Text Models

Writing email

Prompt – Write an email requesting a meeting with a potential client about a new product.

Prompt – I have received the following email from one of our suppliers –

“””

Dear [Your Name], We haven’t received the payment for Invoice #12345. Could you please update us on the payment status? Regards, [Supplier Name].

“””

Draft the response email informing the supplier that the payment will be made by the end of the week.

Text Models

Summarize/ query text

Prompt – Summarize the following text

“””

The first lens scans use cases for generative AI that organizations could adopt. We define a “use case” as a targeted application of generative AI to a specific business challenge, resulting in one or more measurable outcomes. For example, a use case in marketing is the application of generative AI to generate creative content such as personalized emails, the measurable outcomes of which potentially include reductions in the cost of generating such content and increases in revenue from the enhanced effectiveness of higher-quality content at scale. We identified 63 generative AI use cases spanning 16 business functions that could deliver total value in the range of $2.6 trillion to $4.4 trillion in economic benefits annually when applied across industries.

That would add 15 to 40 percent to the $11 trillion to $17.7 trillion of economic value that we now estimate nongenerative artificial intelligence and analytics could unlock. (Our previous estimate from 2017 was that AI could deliver $9.5 trillion to $15.4 trillion in economic value.)

Our second lens complements the first by analyzing generative AI’s potential impact on the work activities required in some 850 occupations. We modeled scenarios to estimate when generative AI could perform each of more than 2,100 “detailed work activities”—such as “communicating with others about operational plans or activities”—that make up those occupations across the world economy. This enables us to estimate how the current capabilities of generative AI could affect labor productivity across all work currently done by the global workforce.

Some of this impact will overlap with cost reductions in the use case analysis described above, which we assume are the result of improved labor productivity. Netting out this overlap, the total economic benefits of generative AI—including the major use cases we explored and the myriad increases in productivity that are likely to materialize when the technology is applied across knowledge workers’ activities—amounts to $6.1 trillion to $7.9 trillion annually (Exhibit 2).

“””

Text Models

Translation

Prompt – Translate the following to Hindi - "Actions speak louder than words"

Prompt – Translate this to English "Naach na jaane aangan tedha"

Image Models

Experiments

Credit Card Bill

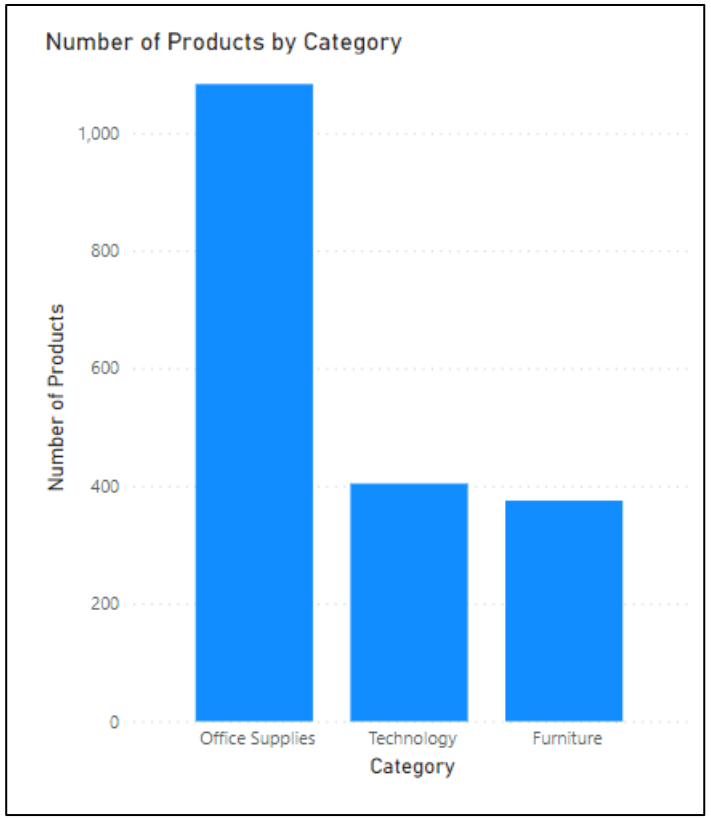
This is a bill in which you have to pay. If you do not pay within one (1) month, a $250.00 fine is assessed.

Name: John Phillips
Address 123 Main Street
San Francisco, CA 12345

Phone: (123) 456-7890
CC Number: XXXXXXXXXXX1234
Bill Received: 01/16/1968

Your Transactions

Item	Price
The ABC Store - Cookies	$2.81
Orville's Bakery - Donuts	$5.95
Stan's Gas Station - 10 Gallons of Gas	$40.00
	Total: $48.76



Code Generating Models

Problem 1: Retrieve the names of all students who are in grade 10.

Table name is “Students” and below is a sample of data in this table

ID	Name	Age	Grade
1	Alice	15	10
2	Bob	16	11
3	Carol	15	10
…	…	…	…

Have: Table named Students containing columns ID, Name, Age and Grade.

Want: Retrieve names of students in grade 10.

Prompt – I have a table named Students containing columns ID, Name, Age and Grade. Give me PostgreSQL code to retrieve names of students in grade 10.

Code Generating Models

Problem 2: Identify salespeople who have made more than one sale in any 7-day window.

Table name is Sales and below is a sample of data in this table

SaleID	Date	Product	Amount	SalespersonID
1	2023-01-01	Laptop	1000	201
2	2023-01-03	Phone	500	202
3	2023-01-05	Laptop	1000	201
4	2023-01-07	Tablet	300	203
5	2023-01-09	Phone	500	202

Have: Table named Sales containing columns SaleID, Date and SalespersonID.

Want: Identify salespeople who have made more than one sale in any 7-day window.

Prompt – I have a table named “Sales” containing columns SaleID, Date and SalespersonID. Give me PostgreSQL code to identify salespeople who have made more than one sale in any 7-day window.

Code Generating Models

Problem 3: Debug SQL code

Code

SELECT SUM(Bonus) as TotalBonus FROM Employees;

Error

ERROR: function sum(character varying) does not exist

LINE 1: SELECT SUM(Bonus) as TotalBonus

Prompt – I ran the following code in PostgreSQL -

SELECT SUM(Bonus) as TotalBonus

FROM Employees;

And got this error -

ERROR: function sum(character varying) does not exist

LINE 1: SELECT SUM(Bonus) as TotalBonus

How can I resolve it.

Code Generating Models

Problem 4: Explain SQL code

```
WITH MonthlySales AS (
   SELECT EXTRACT(YEAR FROM o.OrderDate) AS Year, EXTRACT(MONTH FROM o.OrderDate) AS Month, p.ProductCategory,
      SUM(od.Quantity * od.Price) OVER (PARTITION BY p.ProductCategory ORDER BY o.OrderDate ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT
ROW) AS CumulativeSales
   FROM Orders o
   JOIN OrderDetails od ON o.OrderID = od.OrderID
   JOIN Products p ON od.ProductID = p.ProductID
   GROUP BY EXTRACT(YEAR FROM o.OrderDate), EXTRACT(MONTH FROM o.OrderDate), p.ProductCategory
),
YearlyAverages AS (
   SELECT ProductCategory, AVG(CumulativeSales) AS AvgSales
   FROM MonthlySales
   WHERE Year = 2022
   GROUP BY ProductCategory
)
SELECT ms.Year, ms.Month, ms.ProductCategory, ms.CumulativeSales, ya.AvgSales
FROM MonthlySales ms
JOIN YearlyAverages ya ON ms.ProductCategory = ya.ProductCategory
WHERE ms.CumulativeSales > ya.AvgSales
ORDER BY ms.ProductCategory, ms.Year, ms.Month;
```

Code Generating Models

Python code

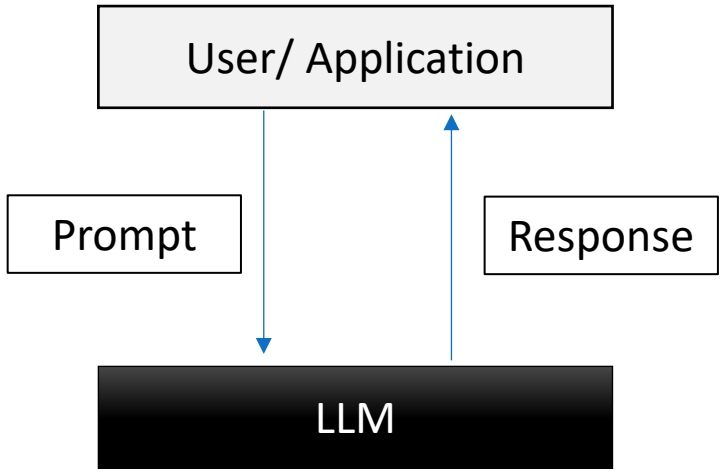
Problem: Create a bar chart of the 'Age' column against the 'Name' column from the given DataFrame.
data = { 'Name': ['Alice', 'Bob', 'Carol', 'David', 'Emily'], 'Age': [25, 30, 22, 28, 24], }

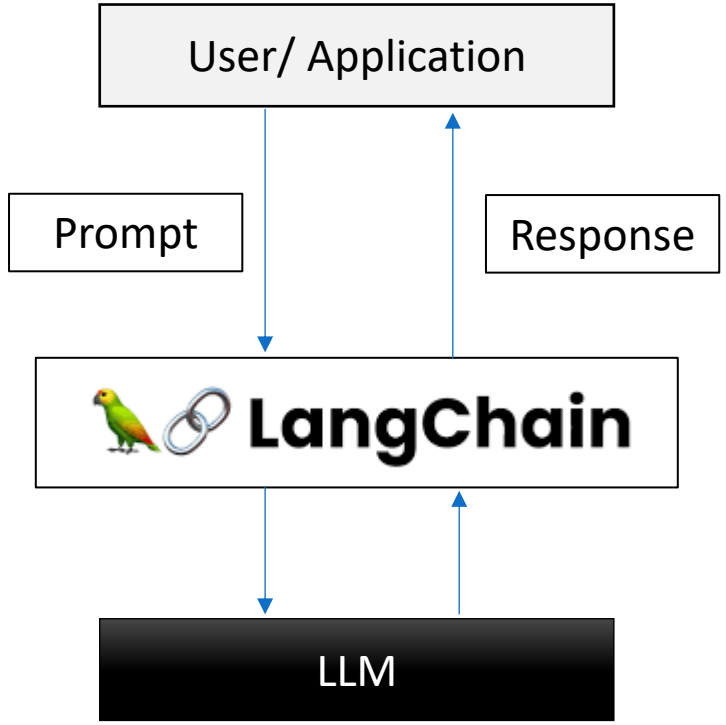
Prompt – Give Python code for creating a bar chart of the 'Age' column against the 'Name' column from the given DataFrame.
data = { 'Name': ['Alice', 'Bob', 'Carol', 'David', 'Emily'], 'Age': [25, 30, 22, 28, 24], }

LangChain

For a simple prompt-response setup, langchain is not much useful

Why?

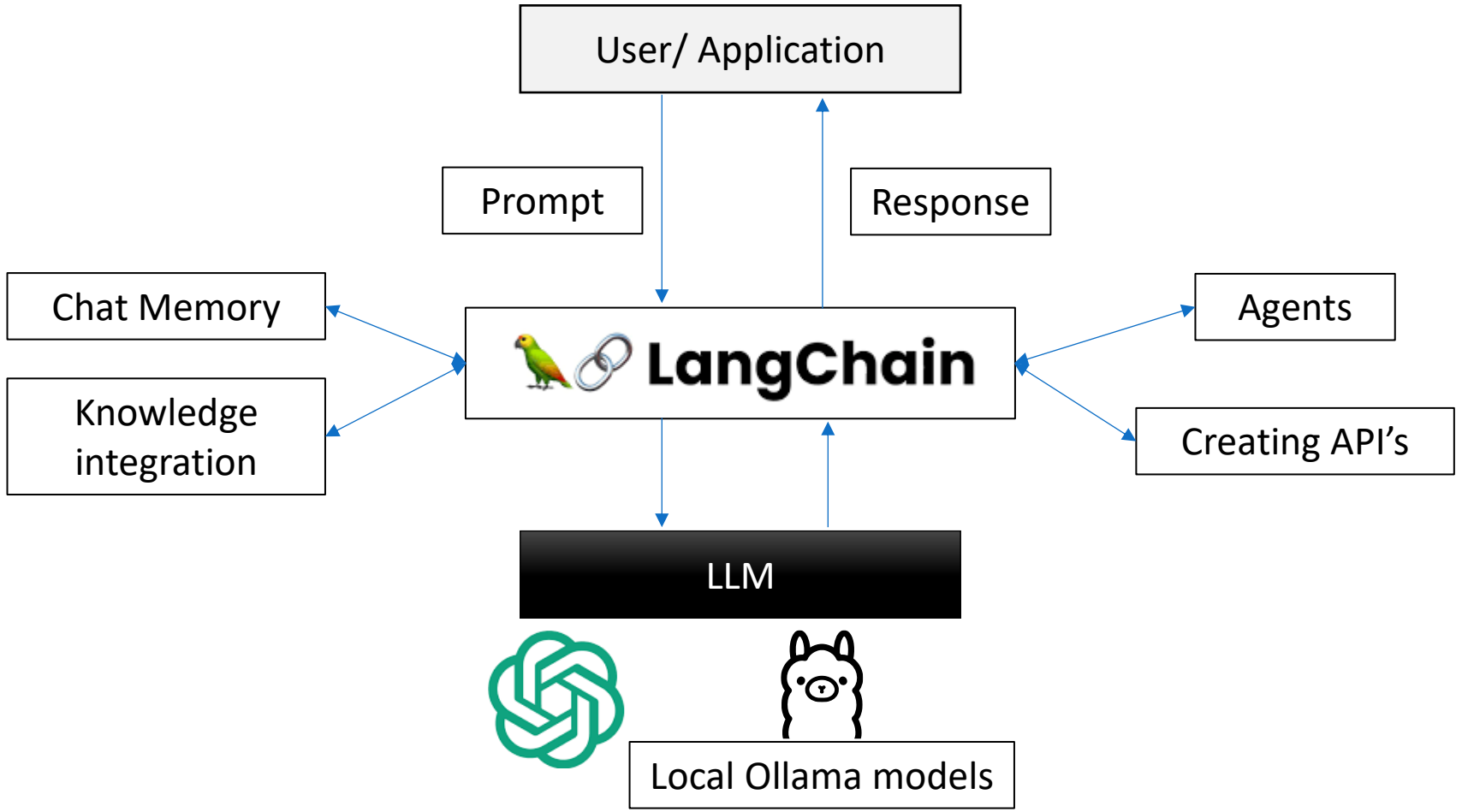




LangChain

For advanced applications, langchain simplifies the coding

Why?



LangChain

Why & How?

What is LangChain?
LangChain is a framework designed to simplify the creation of applications using LLMs.

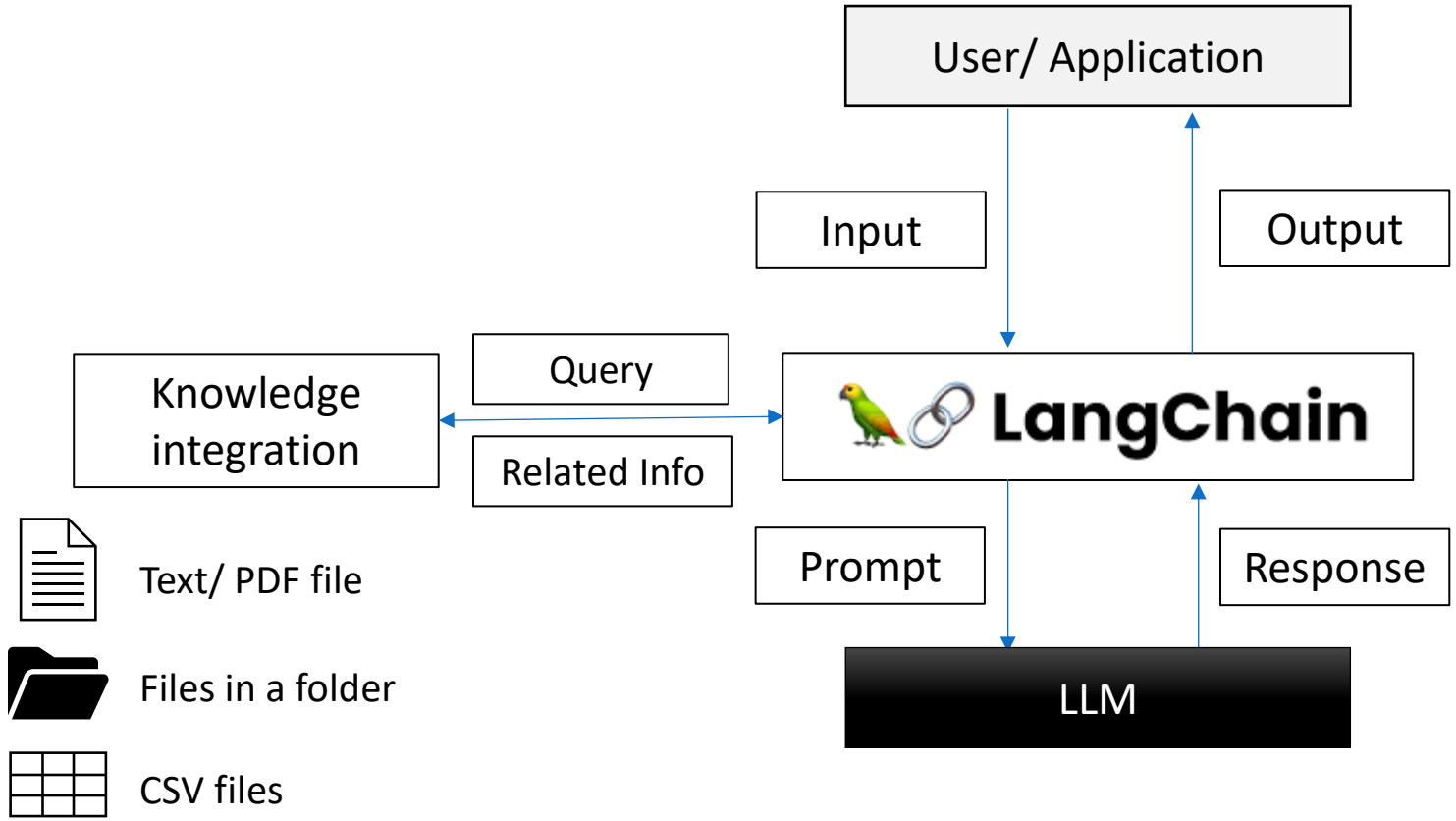
Why use LangChain?
- It makes it easy to add and replace components that bring advanced capabilities like agency, knowledge integration, memory etc.
- Easily switch between local Ollama models or other models like OpenAI/ Google's Generative AI

Install Langchain libraries
- pip install langchain
- pip install langchain-core
- pip install langchain-Ollama
- pip install langchain_community

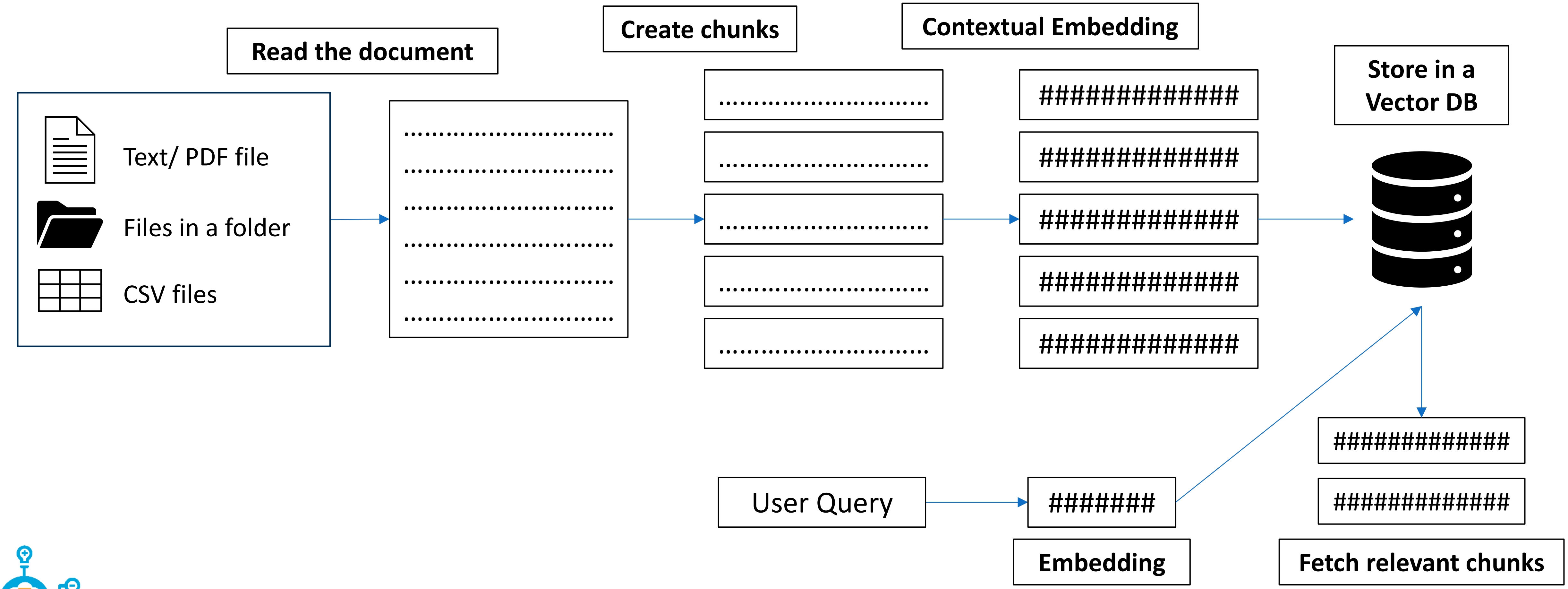
Knowledge Integration

What?

Retrieval Augmented Generation (RAG)



RAG Process


Read the document
Text/ PDF file
Files in a folder
CSV files
Create chunks
Contextual Embedding
Store in a
Vector DB
User Query
Embedding
Fetch relevant chunks

Tools and Agents

What