

Data Automation with LLM

ODSC East 2024

Rami Krispin, April 25, 2024

About Me

- Senior Manager
- Forecasting
- MLOps
- Open Source
- Author

About Me

- Senior Manager
- Forecasting
- **MLOps**
- Open Source
- Author

<https://github.com/RamiKrispin/lang2sql>

Agenda

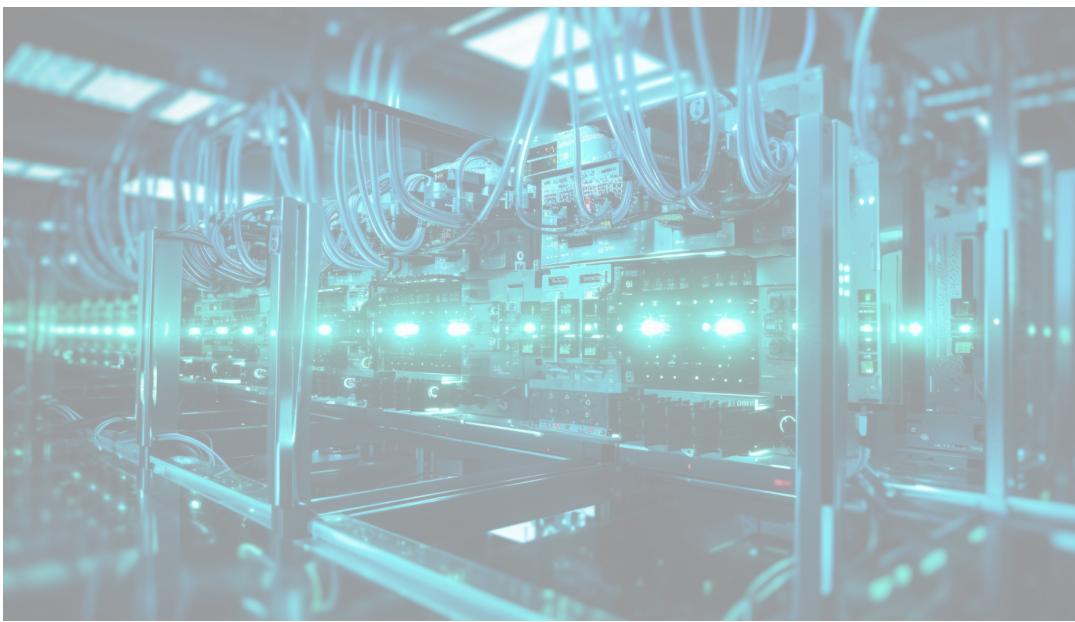
- Motivation
- Architecture
- OpenAI POC
- Ollama POC
- Next Steps

LLM in a Nutshell

LLM in a Nutshell



Data



Model Training



Inference



Fine Tuning

LLM in a Nutshell



The busy person's
intro to LLMs

Andrey Karpathy

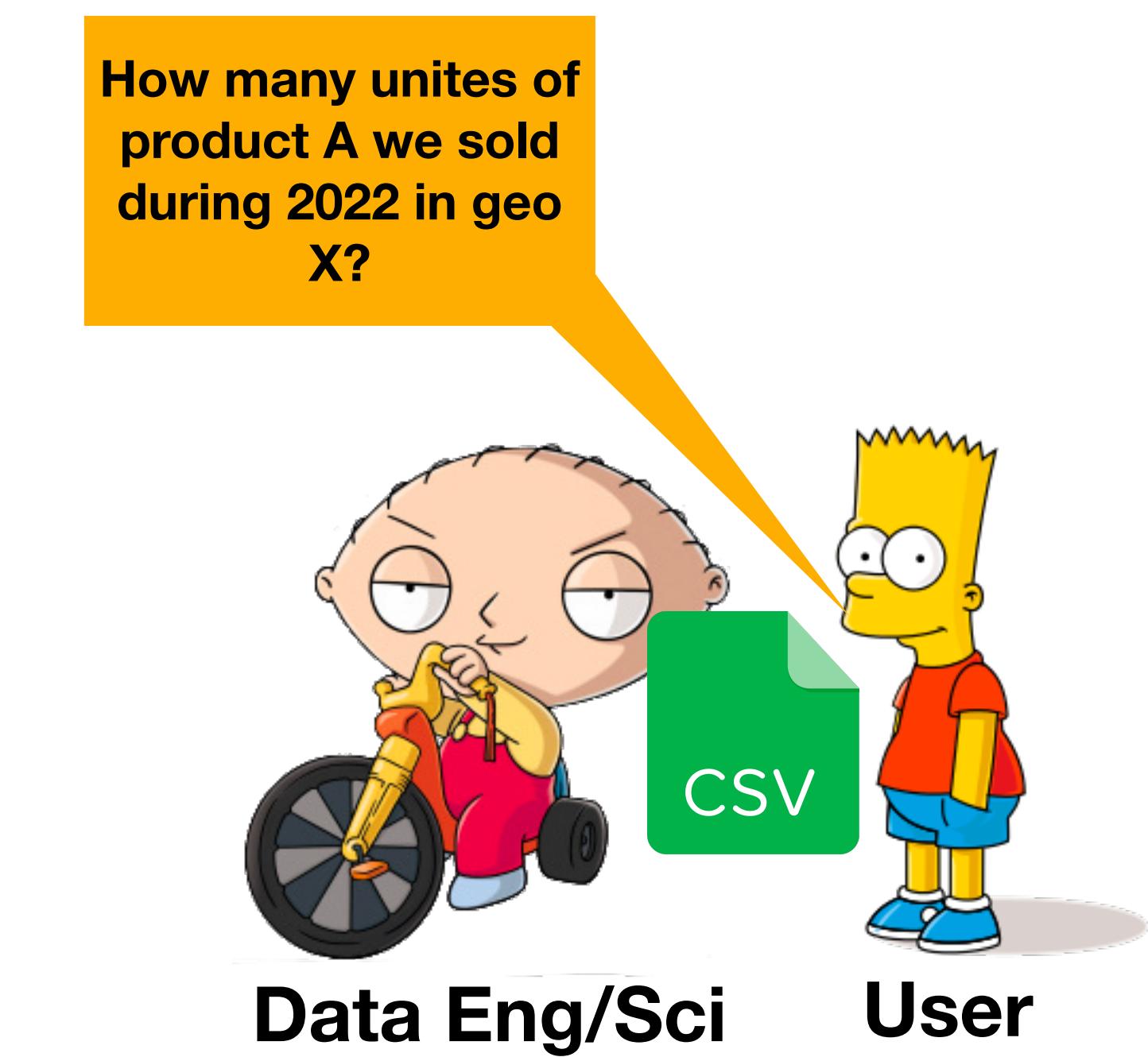


Motivation

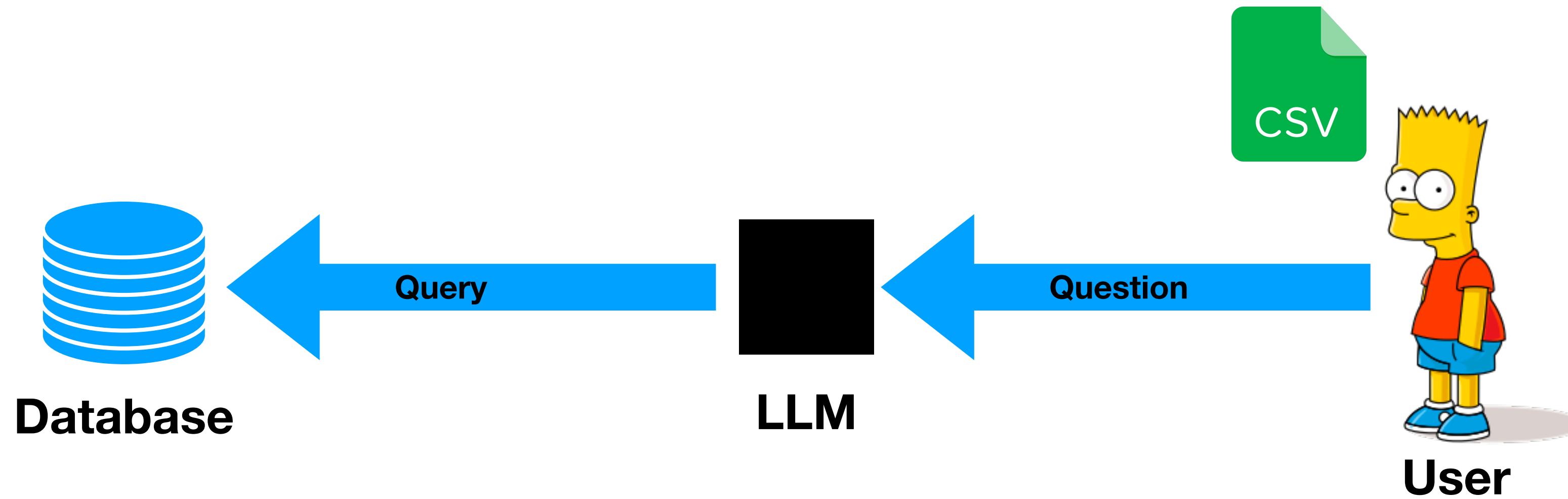
Motivation



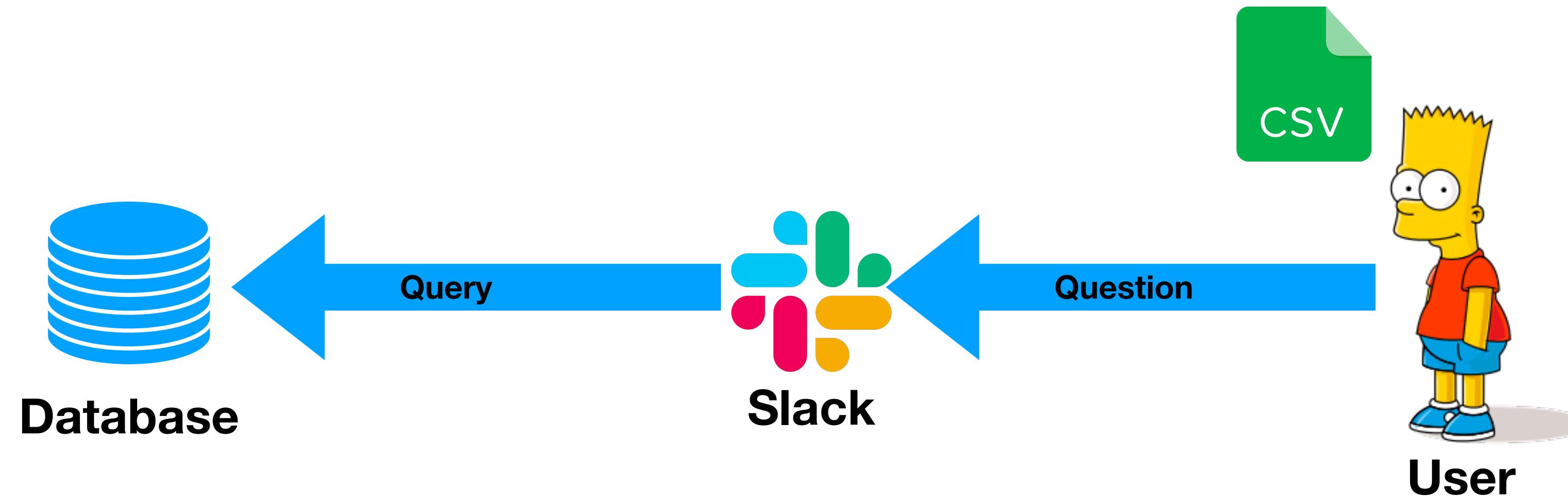
Database



Motivation



Motivation



Motivation



A Hacker's Guide to Language Models

Jeremy Howard, fast.ai

Motivation

```
import torch
from peft import PeftModel
from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig

ax_model = '/home/jhoward/git/ext/axolotl/glora-out'

tokr = AutoTokenizer.from_pretrained('meta-llama/Llama-2-7b-hf')

model = AutoModelForCausalLM.from_pretrained('meta-llama/Llama-2-7b-hf',
                                              torch_dtype=torch.bfloat16, device_map=0)
model = PeftModel.from_pretrained(model, ax_model)
model = model.merge_and_unload()
model.save_pretrained('sql-model')

toks = tokr(sql_prompt(tst), return_tensors="pt")

res = model.generate(**toks.to("cuda"), max_new_tokens=250).to('cpu')

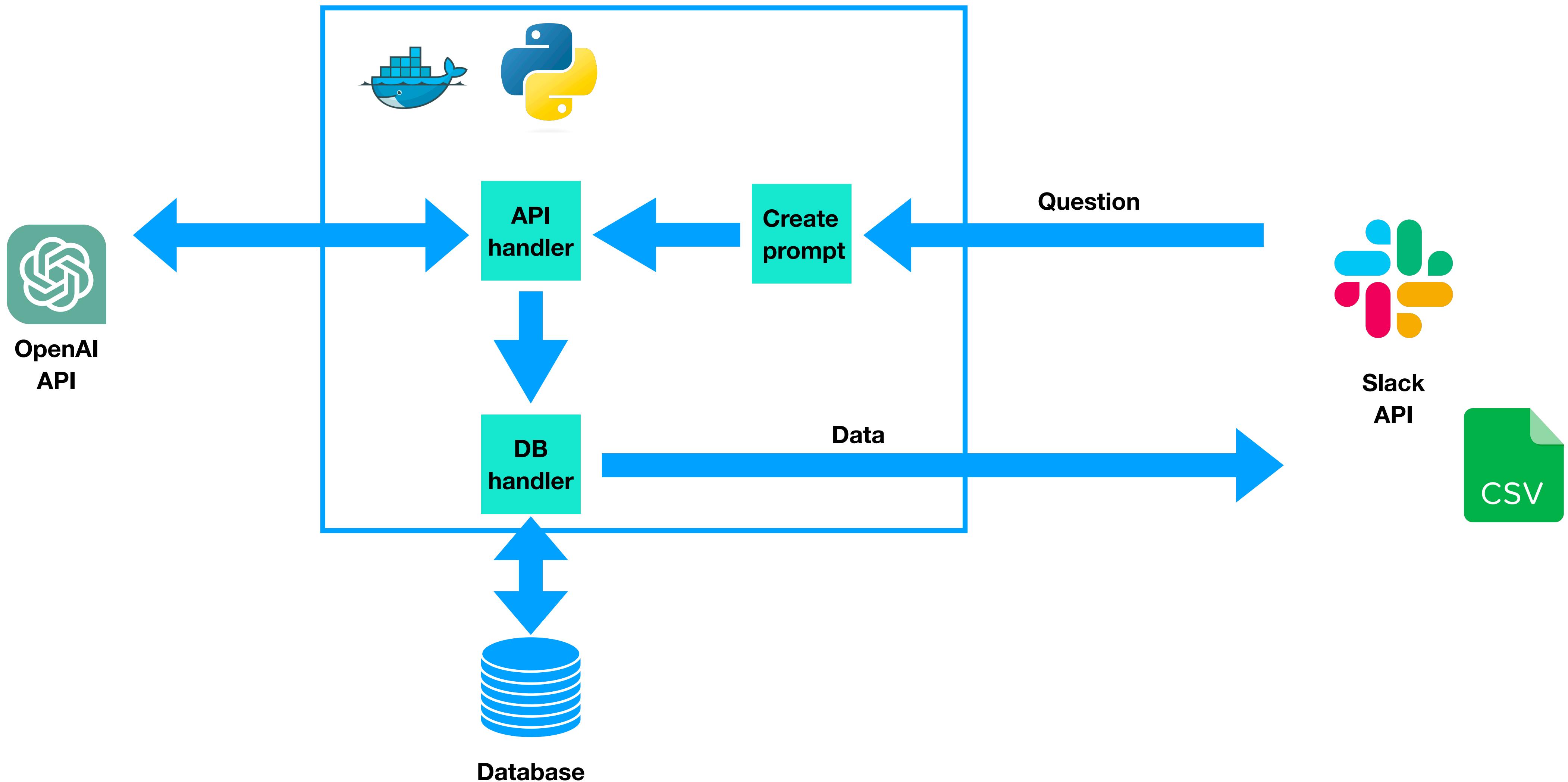
print(tokr.batch_decode(res)[0])
```

<s> SYSTEM: Use the following contextual information to concisely answer the question.

USER: CREATE TABLE farm_competition (Hosts VARCHAR, Theme VARCHAR)
====
Get the count of competition hosts by theme.
ASSISTANT: SELECT COUNT(Hosts), Theme FROM farm_competition GROUP BY Theme</s>

Architecture

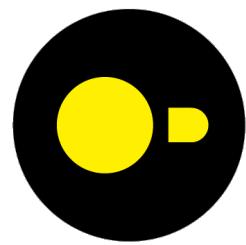
Architecture



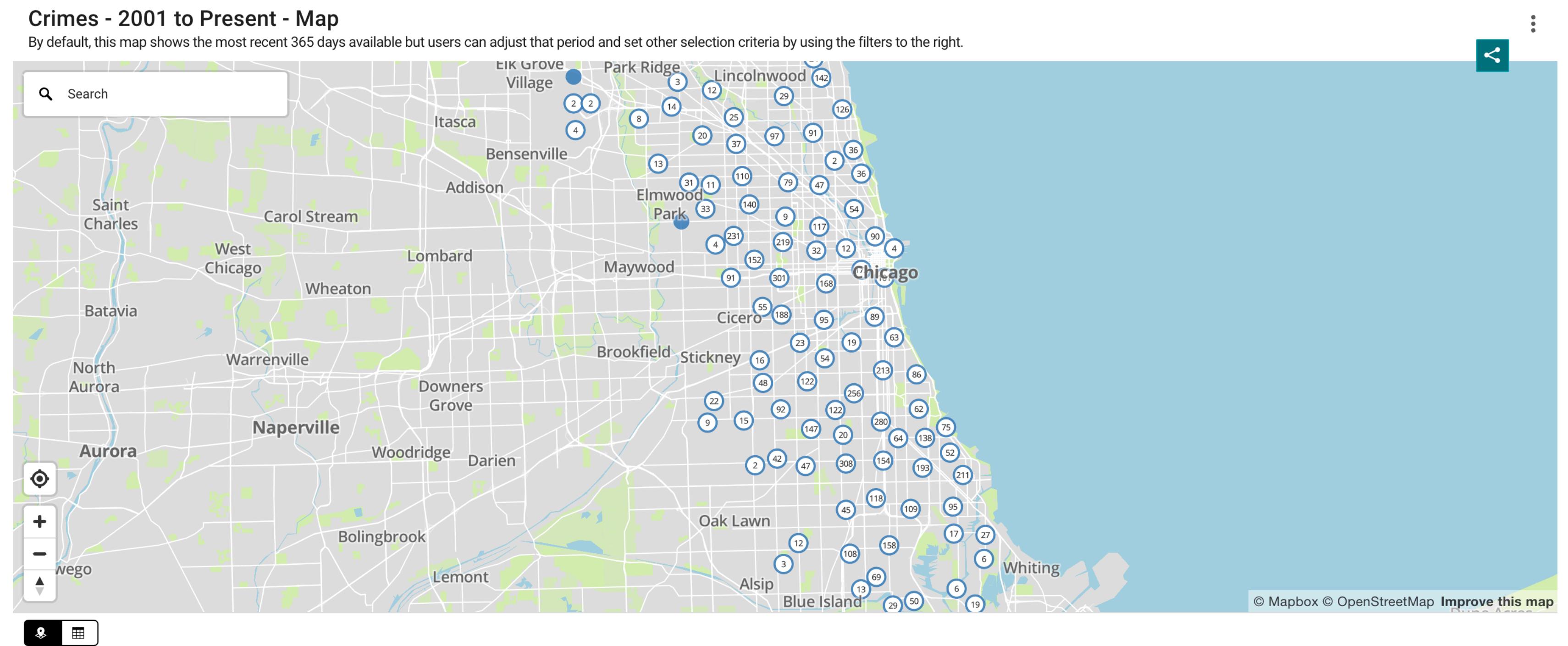
OpenAI POC

OpenAI POC

Tools and Data

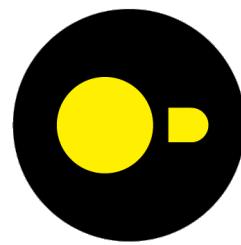


DuckDB



OpenAI POC

Tools and Data



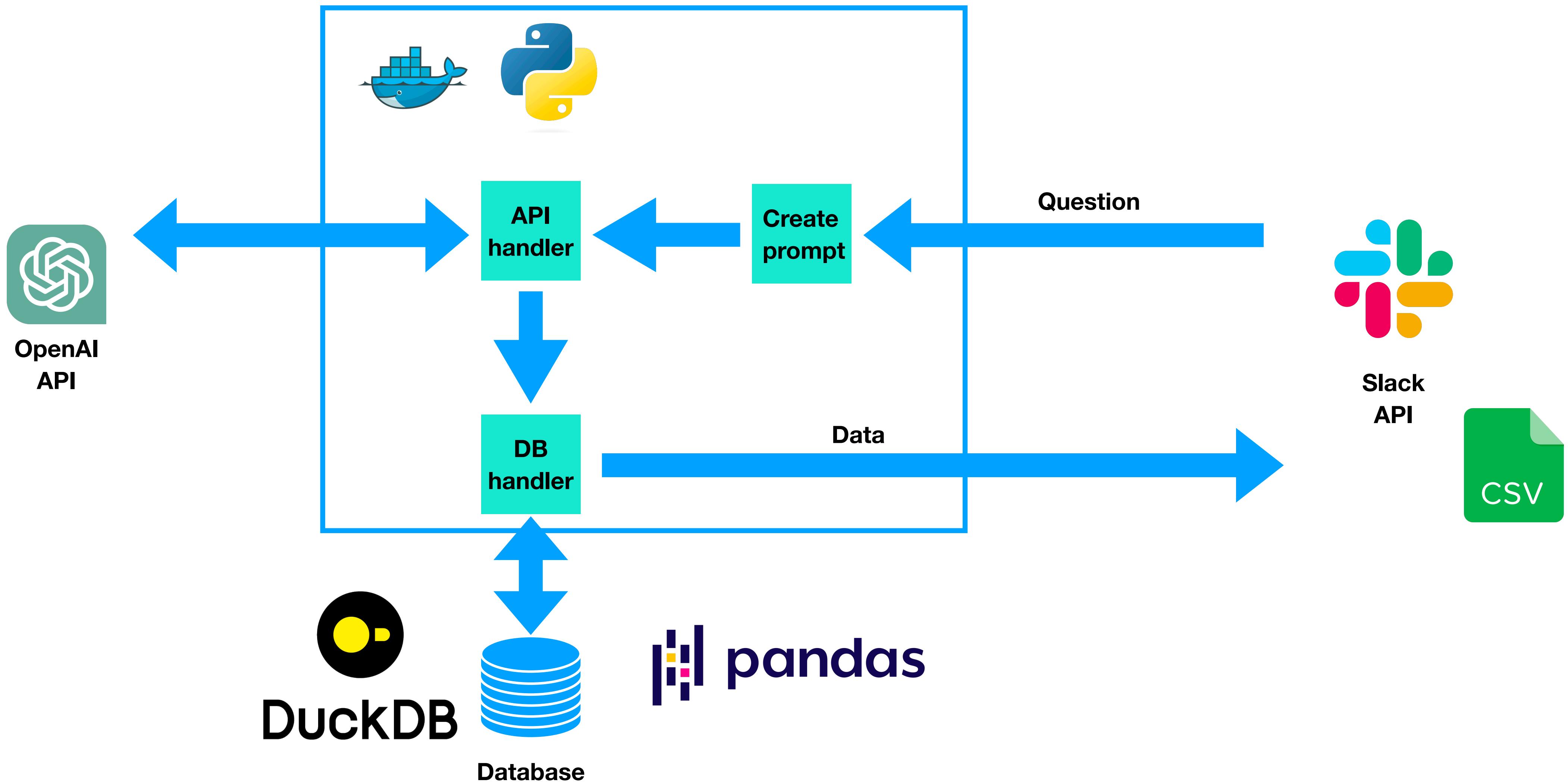
DuckDB



```
duckdb.sql("DESCRIBE SELECT * FROM chicago_crime;")
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar	extra varchar
id	BIGINT	YES	NULL	NULL	NULL
case_number	VARCHAR	YES	NULL	NULL	NULL
datetime	VARCHAR	YES	NULL	NULL	NULL
block	VARCHAR	YES	NULL	NULL	NULL
iucr	VARCHAR	YES	NULL	NULL	NULL
primary_type	VARCHAR	YES	NULL	NULL	NULL
description	VARCHAR	YES	NULL	NULL	NULL
location_description	VARCHAR	YES	NULL	NULL	NULL
arrest	BOOLEAN	YES	NULL	NULL	NULL
domestic	BOOLEAN	YES	NULL	NULL	NULL
beat	BIGINT	YES	NULL	NULL	NULL
district	BIGINT	YES	NULL	NULL	NULL
ward	DOUBLE	YES	NULL	NULL	NULL
community_area	BIGINT	YES	NULL	NULL	NULL
fbi_code	VARCHAR	YES	NULL	NULL	NULL
x_coordinate	DOUBLE	YES	NULL	NULL	NULL
y_coordinate	DOUBLE	YES	NULL	NULL	NULL
year	BIGINT	YES	NULL	NULL	NULL
updated_on	VARCHAR	YES	NULL	NULL	NULL
latitude	DOUBLE	YES	NULL	NULL	NULL
longitude	DOUBLE	YES	NULL	NULL	NULL
21 rows					6 columns

Architecture



Prompt Engineering 101

It is All About the Context

**How many cases ended up with
arrest?**

OpenAI POC

Prompt

 You
Create an SQL query that counts the number of records that ended up with an arrest

 ChatGPT
To create an SQL query that counts the number of records where an arrest occurred, you need to have a column in your database that indicates whether an arrest happened or not. For the sake of this example, let's assume you have a table called `crime_records` with a column named `arrested` that is a boolean column indicating whether an arrest occurred (1 for true, 0 for false).
Here's a sample SQL query:

```
sql
SELECT COUNT(*) AS arrest_count
FROM crime_records
WHERE arrested = 1;
```

In this query:

- `SELECT COUNT(*)`: This selects the count of records.
- `AS arrest_count`: This aliases the result column as `arrest_count` for easier reference.
- `FROM crime_records`: Specifies the table from which to retrieve the data.
- `WHERE arrested = 1`: Filters the records to include only those where the `arrested` column is equal to 1 (indicating an arrest).

Please adjust the table and column names based on your actual database schema. If your schema is different, provide more details, and I can help you tailor the query accordingly.

OpenAI POC

Prompt

The screenshot shows a conversation between a user and ChatGPT. The user, identified by a green circular icon with 'RA' and the title 'You', asks for an SQL query to count records where an arrest occurred. ChatGPT, represented by a blue circular icon with a swirl, responds by explaining that it needs a boolean column 'arrested' in a 'crime_records' table. It then provides a sample SQL query:

```
sql
SELECT COUNT(*) AS arrest_count
FROM crime_records
WHERE arrested = 1;
```

A blue rectangular box highlights this code block. Below the code, ChatGPT explains the components of the query. At the bottom, there are icons for copy, download, and other actions.

In this query:

- `SELECT COUNT(*)`: This selects the count of records.
- `AS arrest_count`: This aliases the result column as `arrest_count` for easier reference.
- `FROM crime_records`: Specifies the table from which to retrieve the data.
- `WHERE arrested = 1`: Filters the records to include only those where the `arrested` column is equal to 1 (indicating an arrest).

Please adjust the table and column names based on your actual database schema. If your schema is different, provide more details, and I can help you tailor the query accordingly.

Copy Download ⌂ ⌂ ⌂ ⌂

OpenAI POC

Prompt

The screenshot shows a dark-themed chat interface. A user message from 'You' is at the top, followed by a response from 'ChatGPT'. The user message includes a small profile icon with 'RA' and the text: 'I have a table named chicago_crime with the crime records in Chicago City since 2021. The Arrest field defines if the case ended up with arrest or not, and it is a boolean (true or false). I want to create an SQL query that counts the number of records that ended up with an arrest.' The AI response from 'ChatGPT' includes a small profile icon with a green swirl and the text: 'Certainly! You can use the `COUNT` function along with a `WHERE` clause to filter records where the Arrest field is true. Here's an example SQL query:' Below this, there is a code block labeled 'sql' containing the following SQL query:

```
sql
SELECT COUNT(*) AS ArrestedCases
FROM chicago_crime
WHERE Arrest = true;
```

A 'Copy code' button is located to the right of the code block. Below the code block, a note explains the query: 'This query selects the count of records from the `chicago_crime` table where the `Arrest` field is true and aliases the result as `ArrestedCases`.'

At the bottom of the AI response, there are four small icons: a copy icon, a share icon, a link icon, and a refresh/circular arrow icon.

OpenAI POC

Prompt

A screenshot of a ChatGPT interface. The user (RA) asks about a table named 'chicago_crime' containing crime records from Chicago City since 2021. The 'Arrest' field is a boolean indicating if the case ended with an arrest. The user wants to create an SQL query to count the number of records where 'Arrest' is true. ChatGPT responds by providing an example SQL query:

```
sql
SELECT COUNT(*) AS ArrestedCases
FROM chicago_crime
WHERE Arrest = true;
```

The AI's response is highlighted with a blue border. Below the code, a note explains its purpose: "This query selects the count of records from the `chicago_crime` table where the `Arrest` field is true and aliases the result as `ArrestedCases`."

OpenAI POC

Prompt

SYSTEM Given the following SQL tables, your job is to write queries given a user's request.

```
CREATE TABLE Orders (
    OrderID int,
    CustomerID int,
    OrderDate datetime,
    OrderTime varchar(8),
    PRIMARY KEY (OrderID)
);
```

```
CREATE TABLE OrderDetails (
    OrderDetailID int,
    OrderID int,
    ProductID int,
    Quantity int,
    PRIMARY KEY (OrderDetailID)
);
```

```
CREATE TABLE Products (
    ProductID int,
    ProductName varchar(50),
    Category varchar(50),
    UnitPrice decimal(10, 2),
    Stock int,
    PRIMARY KEY (ProductID)
);
```

```
CREATE TABLE Customers (
    CustomerID int,
    FirstName varchar(50),
    LastName varchar(50),
    Email varchar(100),
    Phone varchar(20),
    PRIMARY KEY (CustomerID)
);
```

USER Write a SQL query which computes the average total order value for all orders on 2023-04-01.

OpenAI POC

Prompt

SYSTEM

Given the following SQL tables, your job is to write queries given a user's request.

```
CREATE TABLE Orders (
    OrderID int,
    CustomerID int,
    OrderDate datetime,
    OrderTime varchar(8),
    PRIMARY KEY (OrderID)
);
```

```
CREATE TABLE OrderDetails (
    OrderDetailID int,
    OrderID int,
    ProductID int,
    Quantity int,
    PRIMARY KEY (OrderDetailID)
);
```

```
CREATE TABLE Products (
    ProductID int,
    ProductName varchar(50),
    Category varchar(50),
    UnitPrice decimal(10, 2),
    Stock int,
    PRIMARY KEY (ProductID)
);
```

```
CREATE TABLE Customers (
    CustomerID int,
    FirstName varchar(50),
    LastName varchar(50),
    Email varchar(100),
    Phone varchar(20),
    PRIMARY KEY (CustomerID)
);
```

USER

Write a SQL query which computes the average total order value for all orders on 2023-04-01.

OpenAI POC

Prompt

SYSTEM Given the following SQL tables, your job is to write queries given a user's request.

```
CREATE TABLE Orders (
    OrderID int,
    CustomerID int,
    OrderDate datetime,
    OrderTime varchar(8),
    PRIMARY KEY (OrderID)
);
```

```
CREATE TABLE OrderDetails (
    OrderDetailID int,
    OrderID int,
    ProductID int,
    Quantity int,
    PRIMARY KEY (OrderDetailID)
);
```

```
CREATE TABLE Products (
    ProductID int,
    ProductName varchar(50),
    Category varchar(50),
    UnitPrice decimal(10, 2),
    Stock int,
    PRIMARY KEY (ProductID)
);
```

```
CREATE TABLE Customers (
    CustomerID int,
    FirstName varchar(50),
    LastName varchar(50),
    Email varchar(100),
    Phone varchar(20),
    PRIMARY KEY (CustomerID)
);
```

USER Write a SQL query which computes the average total order value for all orders on 2023-04-01.

OpenAI POC

Prompt

```
import duckdb
import pandas as pd
import openai
import ollama

class SqlPrompt:

    """
    Setting Up the Prompt Features
    """

    def __init__(self, table):
        self.table = table
        self.question = None
        self.message = None

    def get_table_schema(self):
        """
        The function uses DuckDB and SQL to extract table schema from Pandas DataFrame object
        """

        tbl_describe = duckdb.sql("DESCRIBE SELECT * FROM " + self.table + ";")
        col_attr = tbl_describe.df()[["column_name", "column_type"]]
        col_attr["column_joint"] = col_attr["column_name"] + " " + col_attr["column_type"]
        self.schema = str(list(col_attr["column_joint"].values)).replace('[', '').replace(']', '').replace('\'', '')
        self.column_names = col_attr["column_name"]
        self.column_type = col_attr["column_type"]
```

OpenAI POC

Prompt

```
def set_prompt(self, question):

    if "schema" not in self.__dict__.keys():
        self.get_table_schema()

    system_template = """

Given the following SQL table, your job is to write queries given a user's request. \n

CREATE TABLE {} ({})

"""

    user_template = "Write a SQL query that returns - {}"

    self.system = system_template.format(self.table, self.schema)
    self.user = user_template.format(question)
    self.message = [
        {
            "role": "system",
            "content": self.system
        },
        {
            "role": "user",
            "content": self.user
        }
    ]
```

OpenAI POC

Prompt

```
def set_prompt(self, question):

    if "schema" not in self.__dict__.keys():
        self.get_table_schema()

    system_template = """

        Given the following SQL table, your job is to write queries given a user's request. \n\n

        CREATE TABLE {} ({})

        """

    user_template = "Write a SQL query that returns - {}"

    self.system = system_template.format(self.table, self.schema)
    self.user = user_template.format(question)
    self.message = [
        {
            "role": "system",
            "content": self.system
        },
        {
            "role": "user",
            "content": self.user
        }
    ]
```

OpenAI POC

Prompt

```
def openai_request(self,
                    openai_api_key,
                    model = "gpt-3.5-turbo",
                    temperature = 0,
                    max_tokens = 256,
                    frequency_penalty = 0,
                    presence_penalty= 0):
    openai.api_key = openai_api_key
    self.openai_response = openai.ChatCompletion.create(
        model = model,
        messages = self.message,
        temperature = temperature,
        max_tokens = max_tokens,
        frequency_penalty = frequency_penalty,
        presence_penalty = presence_penalty)

    self.query = add_quotes(query = self.openai_response["choices"][0]["message"]["content"],
                           col_names = self.column_names)

    self.query = remove_code_chunk(self.query)

def get_data(self):
    if self.message is None:
        print("The prompt is not defined")
        return
    else:
        print(duckdb.sql(self.query))
        self.data = duckdb.sql(self.query)
```

OpenAI POC

Prompt

```
def openai_request(self,
                   openai_api_key,
                   model = "gpt-3.5-turbo",
                   temperature = 0,
                   max_tokens = 256,
                   frequency_penalty = 0,
                   presence_penalty= 0):
    openai.api_key = openai_api_key
    self.openai_response = openai.ChatCompletion.create(
        model = model,
        messages = self.message,
        temperature = temperature,
        max_tokens = max_tokens,
        frequency_penalty = frequency_penalty,
        presence_penalty = presence_penalty)

    self.query = add_quotes(query = self.openai_response["choices"][0]["message"]["content"],
                           col_names = self.column_names)

    self.query = remove_code_chunk(self.query)

def get_data(self):
    if self.message is None:
        print("The prompt is not defined")
        return
    else:
        print(duckdb.sql(self.query))
        self.data = duckdb.sql(self.query)
```

OpenAI POC

Prompt

```
def ask_question(self,
                 question,
                 openai_api_key,
                 model = "gpt-3.5-turbo",
                 temperature = 0,
                 max_tokens = 256,
                 frequency_penalty = 0,
                 presence_penalty= 0):
    self.set_prompt(question)
    self.openai_request(openai_api_key = openai_api_key,
                        model = "gpt-3.5-turbo",
                        temperature = 0,
                        max_tokens = 256,
                        frequency_penalty = 0,
                        presence_penalty= 0)
```

OpenAI POC

Examples

```
question = "How many cases ended up with arrest?"  
  
my_prompt.ask_question(question = question, openai_api_key = os.getenv('OPENAI_KEY'))  
  
print(my_prompt.query)
```

```
SELECT COUNT(*) AS arrest_count  
FROM chicago_crime  
WHERE "arrest" = TRUE;
```

```
my_prompt.get_data()
```

arrest_count
int64
40483

OpenAI POC

Examples

```
question = "How many cases ended up with arrest?"  
  
my_prompt.ask_question(question = question, openai_api_key = os.getenv('OPENAI_KEY'))
```

```
print(my_prompt.query)
```

```
SELECT COUNT(*) AS arrest_count  
FROM chicago_crime  
WHERE "arrest" = TRUE;
```

```
my_prompt.get_data()
```

arrest_count
int64
40483

OpenAI POC

Examples

```
question = "How many cases ended up with arrest?"  
  
my_prompt.ask_question(question = question, openai_api_key = os.getenv('OPENAI_KEY'))  
  
print(my_prompt.query)
```

```
SELECT COUNT(*) AS arrest_count  
FROM chicago_crime  
WHERE "arrest" = TRUE;
```

```
my_prompt.get_data()
```

arrest_count
int64
40483

OpenAI POC

Examples

```
question = "How many cases ended up with arrest?"  
  
my_prompt.ask_question(question = question, openai_api_key = os.getenv('OPENAI_KEY'))  
  
print(my_prompt.query)  
  
SELECT COUNT(*) AS arrest_count  
FROM chicago_crime  
WHERE "arrest" = TRUE;  
  
my_prompt.get_data()
```

arrest_count
int64
40483

OpenAI POC

Examples

```
question = "How many cases ended up with arrest during 2024?"  
my_prompt.ask_question(question = question, openai_api_key = os.getenv('OPENAI_KEY'))  
print(my_prompt.query)
```

```
SELECT COUNT(*) AS arrest_count  
FROM chicago_crime  
WHERE "arrest" = TRUE AND "year" = 2024;
```

```
my_prompt.get_data()
```

arrest_count
int64
9383

OpenAI POC

Examples

```
question = "How many cases ended up with arrest during 2024?"  
my_prompt.ask_question(question = question, openai_api_key = os.getenv('OPENAI_KEY'))  
print(my_prompt.query)
```

```
SELECT COUNT(*) AS arrest_count  
FROM chicago_crime  
WHERE "arrest" = TRUE AND "year" = 2024;
```

```
my_prompt.get_data()
```

arrest_count
int64

9383

Ollama POC

Ollama POC

Code

```
def ask_ollama(self,
               question,
               model):
    self.set_prompt(question)
    response = ollama.chat(model=model, messages = self.message)
    self.ollama_response = ollama.chat(model=model, messages = self.message)
    self.markdown = add_quotes(query = self.ollama_response['message'][‘content’],
                               col_names = self.column_names)
    self.markdown = remove_text(query = self.markdown)
    self.query = remove_code_chunk(query=self.markdown)
```

Ollama POC

Example

```
my_prompt.ask_ollama(question= question, model = "phi3")  
  
my_prompt.ollama_response  
  
{'model': 'phi3',  
 'created_at': '2024-04-25T12:12:39.878662228Z',  
 'message': {'role': 'assistant',  
  'content': '```sql\nSELECT COUNT(*) AS arrested_cases FROM chicago_crime WHERE arrest = TRUE;\n```\nThis SQL statement counts the number of rows in the `chicago_crime` table where the `arrest` column is set to `TRUE`, indicating that an arrest was made. The result will be labeled as `arrested_cases`.}',  
 'done': True,  
 'total_duration': 31718092141,  
 'load_duration': 2485667,  
 'prompt_eval_duration': 411394000,  
 'eval_count': 76,  
 'eval_duration': 31166075000}  
  
print(my_prompt.markdown)  
  
```sql  
SELECT COUNT(*) AS arrested_cases FROM chicago_crime WHERE "arrest" = TRUE;
```  
  
duckdb.query(my_prompt.query)  
  
arrested_cases  
int64  
40483
```

OpenAI POC

Conclusion

- Works well for simple queries
- Reproducibility
- Cost
- Performance

Next Steps

Next Steps

- Ollama
- Normalized to other frameworks
- Fine tuning
- Other applications - visualization, forecasting, etc

Resources

- Code - <https://github.com/RamiKrispin/lang2sql>
- Jeremy Howard, [A Hackers' Guide to Language Models](#)
- Notebook: <https://github.com/fastai/lm-hackers/blob/main/lm-hackers.ipynb>
- Andrej Karpathy, [Intro to LLM](#)
- OpenAI example - <https://platform.openai.com/examples/default-sql-translate>
- Ollama - <https://ollama.com/>

Questions?

My Data Science Channel



Thank You!