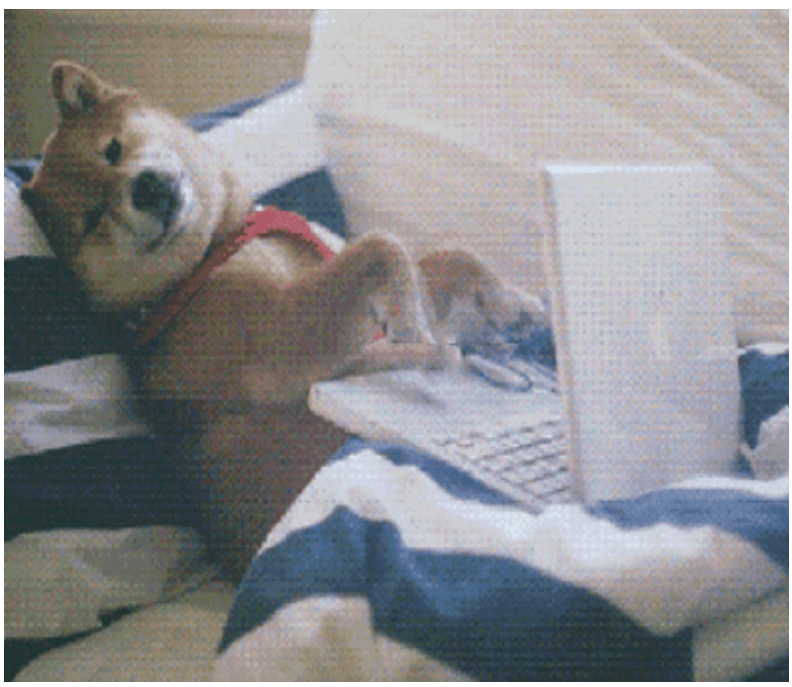
💬🗄️ 🤖ASKING QUE
YOUR DATABASE WITH

Nahuel Defossé ✉ nahuel.defosse
IBM Research Africa

ABOUT MYSELF



- 🐍Pythonista with 18
  - Co-organized SciPy
  - PyCon Argentina a
- 🚜Worked as CTO in H
- 🧪Software Engineer a
- 🛰️Worked in Foundat
Geospatial application
- 💬Currently working c
core features to differe

TOMORROW

Learn about Multimodal Geospatial Found
Terratorch by Wanjiru, Beldine and Reggie

FOLLOW ALONG (OR AT 🏡)

INTRO

In this talk we're gonna show how to use P

- 🗄 Connect to a database and execute t
- SQL Convert natural language questions
- 🔀 Create a workflow
- 🐞 More advanced nodes
- 🔖 Lessons learned

TODO: Put some drawing & rephrase

CONNECT TO A DATABASE A
THE QUERIES

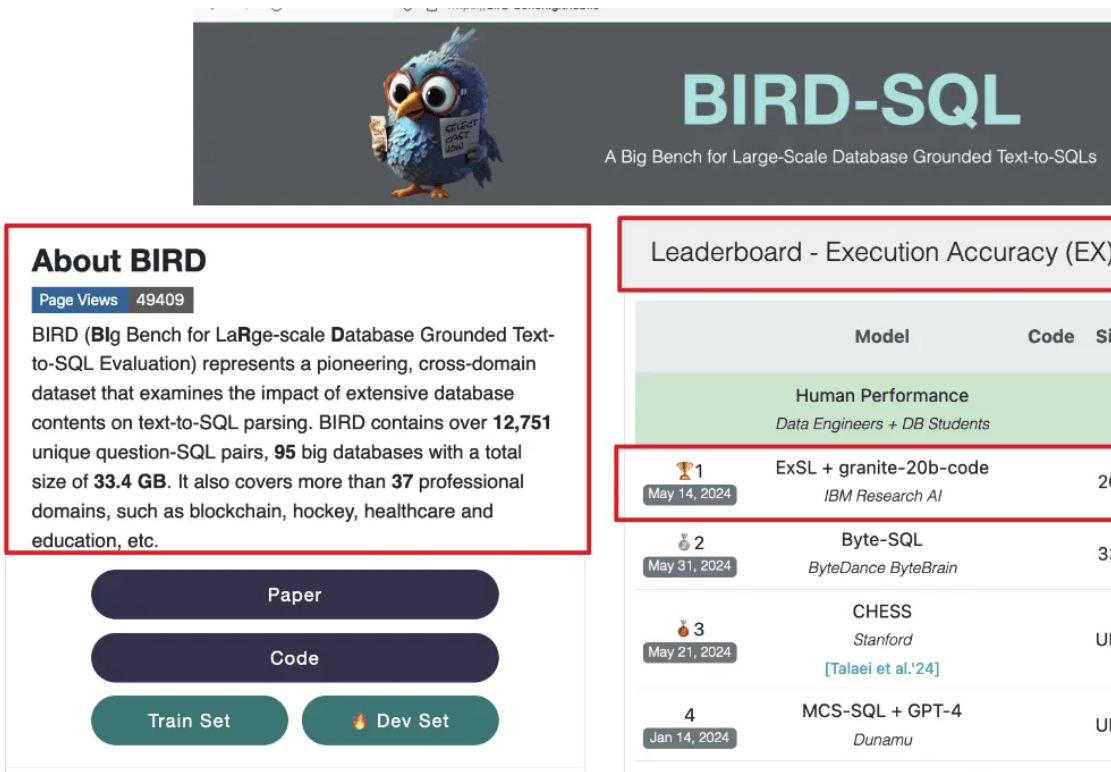
PUBLIC DATASETS USED IN TEXT TO SQL

These datasets define:

- ❓ Natural language questions
- 👌 Expected SQL
- 🏗️ Database schema & content
- 🔎 Evaluation metrics
- 🥇 Leaderboard

- 🕷️ Spide
- 🕷️🕷️ Spi
- 🦚 BIRD
- 🏹 Arche

BIRD



IBM Research in the lederboard 2024-06-02

BIRD MINI-DEV

- It consist of 500 queries classified as **sim**
  and **challenging**

```
from datasets import load_dataset, DownloadConfig

# Load the dataset
dataset = load_dataset(
  "birdsql/bird_mini_dev",
  download_config=DownloadConfig(disable_tqdm=True
)

# Contents
print("Database types: ", *dataset.keys())
sqlite_df = (dataset["mini_dev_sqlite"]
  .to_pandas()
  .pipe(lambda df: df.drop(columns=['question_id']
display(sqlite_df.head(5))
```

```
Database types:  mini_dev_mysql mini_dev_pg
mini_dev_sqlite
```

	db_id	question	evidence	
0	debit_card_specializing	What is the ratio of customers who pay in EUR ...	ratio of customers who pay in EUR against cust...	SELE CAST 'EUR
1	debit_card_specializing	In 2012, who had the least consumption in LAM?	Year 2012 can be presented as Between 201201 A...	SELE FRON INNE
2	debit_card_specializing	What was the average monthly consumption of cu...	Average Monthly consumption = AVG(Consumption)...	SELE AVG(FRON
3	debit_card_specializing	What was the difference in gas consumption bet...	Year 2012 can be presented as Between 201201 A...	SELE SUM 'CZK
4	debit_card_specializing	Which year recorded the most consumption of ga...	The first 4 strings of the Date values in the ...	SELE 4) FR

DOWNLOADING BIRD DATABASES

```
uvx gdown https://drive.google.com/file/d/13VLWIwp
```

```
→ uvx gdown https://drive.google.com/file/d/13VLWIwpw5E3d5DUkMvzw7hvHE67a4
Downloading...
From (original): https://drive.google.com/uc?id=13VLWIwpw5E3d5DUkMvzw7hvHE
From (redirected): https://drive.google.com/uc?id=13VLWIwpw5E3d5DUkMvzw7hv
id=29863c54-8181-4f9e-b437-e75026526da9
To: /Users/nahueldefosse/workspace/slides/pycon-2025-insights-from-sql-llm
100%|                                                                    | 800M/800M [
```

Extracting the archive (3.3GiB)

```
unzip minidev_703.zip
```

PICKING THE EXAMPLE DATABASE `californi`

In `minidev/MINIDEV/dev_databases/california_schools/` we find the 🛢 📄

```
california_schools
├── california_schools.sqlite
└── database_description
    ├── frpm.csv
    ├── satscores.csv
    └── schools.csv
```

```
from pathlib import Path
base = Path('./minidev/MINIDEV/dev_databases/calif
```

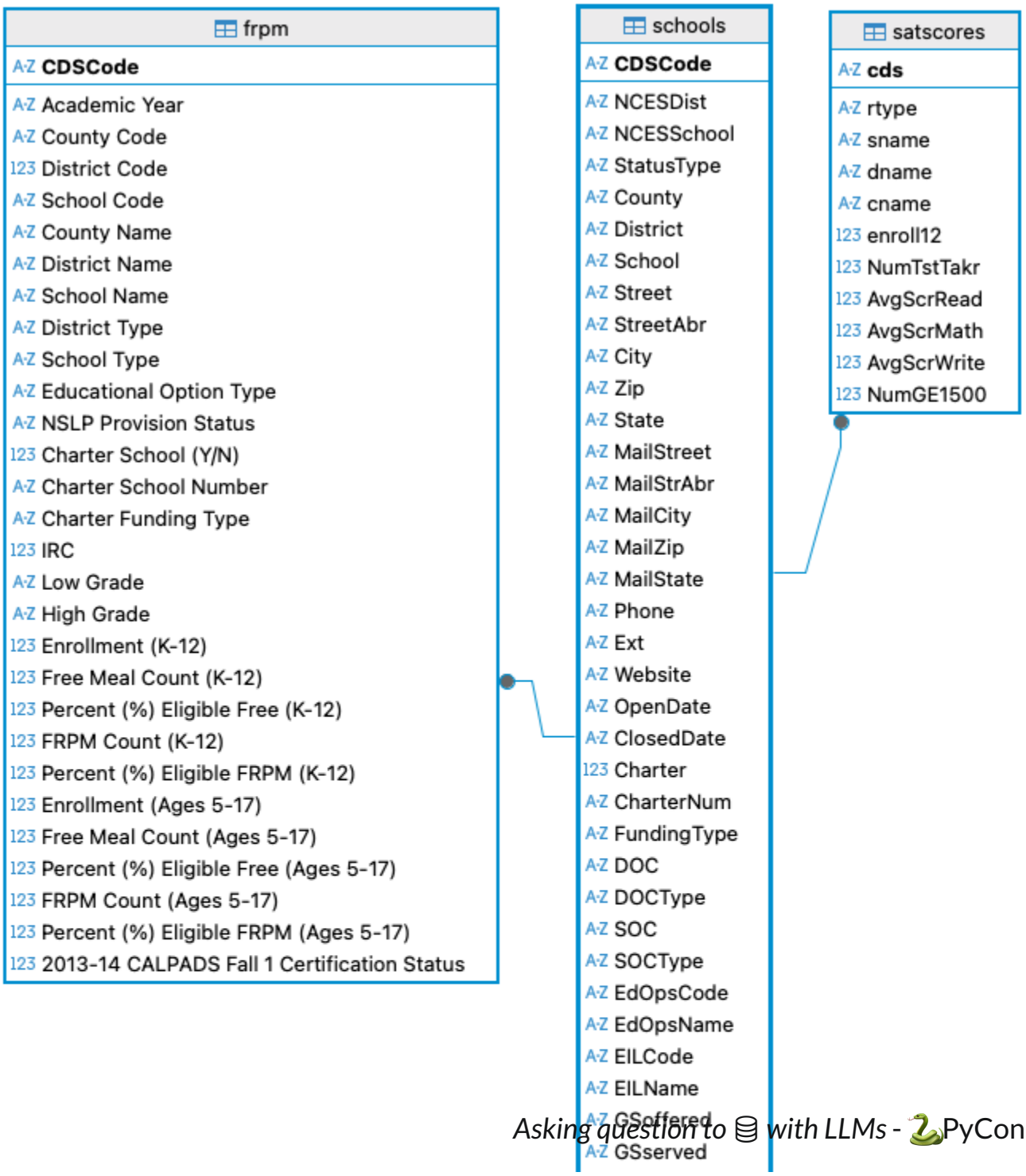
CREATING THE `Engine`

Creating an `Engine` instance connected to database.

```
from sqlalchemy import create_engine
db_path = base / 'california_schools.sqlite'
db_url = f"sqlite:///{db_path}"
engine = create_engine(db_url)
engine
```

```
Engine(sqlite:///minidev/MINIDEV/dev_databases/califor
california_schools.sqlite)
```

ER


frpm
CDSCode
Academic Year
County Code
District Code
School Code
County Name
District Name
School Name
District Type
School Type
Educational Option Type
NSLP Provision Status
Charter School (Y/N)
Charter School Number
Charter Funding Type
IRC
Low Grade
High Grade
Enrollment (K-12)
Free Meal Count (K-12)
Percent (%) Eligible Free (K-12)
FRPM Count (K-12)
Percent (%) Eligible FRPM (K-12)
Enrollment (Ages 5-17)
Free Meal Count (Ages 5-17)
Percent (%) Eligible Free (Ages 5-17)
FRPM Count (Ages 5-17)
Percent (%) Eligible FRPM (Ages 5-17)
2013-14 CALPADS Fall 1 Certification Status
schools
CDSCode
NCESDist
NCESSchool
StatusType
County
District
School
Street
StreetAbr
City
Zip
State
MailStreet
MailStrAbr
MailCity
MailZip
MailState
Phone
Ext
Website
OpenDate
ClosedDate
Charter
CharterNum
FundingType
DOC
DOCType
SOC
SOCType
EdOpsCode
EdOpsName
EILCode
EILName
GSoffered
GSserved
satscores
cds
rtype
sname
dname
cname
enroll12
NumTstTakr
AvgScrRead
AvgScrMath
AvgScrWrite
NumGE1500

GETTING ONE SIMPLE QUESTION

```
simple_queries_df = (sqlite_df
  .pipe(lambda df: df[df.db_id == 'california_scho
  .pipe(lambda df: df[df.difficulty == 'simple']))

simple_queries_df.head(2).set_index("db_id")
```

	question	evidence	
db_id			
california_schools	How many schools with an average score in Math...	Exclusively virtual refers to Virtual = 'F'	SELECT T2.Scho
california_schools	Please list the codes of the schools with a to...	Total enrollment can be represented by `Enroll...	SELECT schools

GETTING ONE SIMPLE QUESTION

```
question_sql_df = (simple_queries_df
  .pipe(lambda df: df[["question", "SQL"]])
  .reset_index())
```

Let's take a look at the `question` and `SQL`

```
question, query = question_sql_df.loc[
  0,
  ["question", "SQL"]]
display(question, query)
```

```
'How many schools with an average score in Math greate
test are exclusively virtual?'
```

```
"SELECT COUNT(DISTINCT T2.School) FROM satscores AS T1
T2 ON T1.cds = T2.CDSCode WHERE T2.Virtual = 'F' AND T
```

EXECUTE THE QUERIES

Now we run the `SQL` column captured in the variable `query` through S
as a DataFrame.

```
from sqlalchemy import text
with engine.connect() as conn:
    result = conn.execute(text(query))
    res_df = pd.DataFrame(result.fetchall()) # 🐼
    display(res_df)
```

	COUNT(DISTINCT T2.School)
0	4

Question: How many schools with an average score in Math greater than **400** in the SAT test are exclusively virtual?
SQL: SELECT **COUNT**(DISTINCT T2.School) FROM satscores AS T1 INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode WHERE T2.Virtual = 'F' AND T1.AvgScrMath > **400**

DATABASE SCHEMA WITH 🦜🔗

LangChain (🦜🔗) community 🐍📦provid
that can retrieve some schema informatio

```
1 !uv pip install langchain-community
```

```
Audited 1 package in 5ms
```

```
1 from langchain_community.utilities import SQLDatab
2 db = SQLDatabase(engine=engine)
3
4 display(db.get_usable_table_names())
```

```
['frpm', 'satscores', 'schools']
```

As we can see, the table names may not be
understandable 🤔

1. SQL Question Answering

```
print(db.get_table_info())
```

```
CREATE TABLE frpm (
        "CDSCode" TEXT NOT NULL,
        "Academic Year" TEXT,
        "County Code" TEXT,
        "District Code" INTEGER,
        "School Code" TEXT,
        "County Name" TEXT,
        "District Name" TEXT,
        "School Name" TEXT,
        "District Type" TEXT,
        "School Type" TEXT,
        "Educational Option Type" TEXT,
        "NSLP Provision Status" TEXT,
        "Charter School (Y/N)" INTEGER,
        "Charter School Number" TEXT,
        "Charter Funding Type" TEXT,
        "IRC" INTEGER,
        "Low Grade" TEXT,
        "High Grade" TEXT,
        "Enrollment (K-12)" REAL,
        "Free Meal Count (K-12)" REAL,
        "Percent (%) Eligible Free (K-12)"
REAL,
        "FRPM Count (K-12)" REAL,
        "Percent (%) Eligible FRPM (K-12)"
REAL,
```

```
        "Enrollment (Ages 5-17)" REAL,
        "Free Meal Count (Ages 5-17)" REAL,
        "Percent (%) Eligible Free (Ages
5-17)" REAL,
        "FRPM Count (Ages 5-17)" REAL,
        "Percent (%) Eligible FRPM (Ages
5-17)" REAL,
        "2013-14 CALPADS Fall 1 Certification
Status" INTEGER,
        PRIMARY KEY ("CDSCode"),
        FOREIGN KEY("CDSCode") REFERENCES
schools ("CDSCode")
)

/*
3 rows from frpm table:
CDSCode Academic Year   County Code
District Code   School Code     County Name
District Name   School Name     District Type
School Type     Educational Option Type NSLP
Provision Status   Charter School (Y/N)
Charter School Number   Charter Funding Type
IRC     Low Grade       High Grade
Enrollment (K-12)       Free Meal Count
(K-12)  Percent (%) Eligible Free (K-12)
FRPM Count (K-12)       Percent (%) Eligible
FRPM (K-12)        Enrollment (Ages 5-17)
Free Meal Count (Ages 5-17)     Percent (%)
Eligible Free (Ages 5-17)  FRPM Count (Ages
```

```
5-17)  Percent (%) Eligible FRPM (Ages 5-17)
2013-14 CALPADS Fall 1 Certification Status
01100170109835  2014-2015       01      10017
0109835 Alameda Alameda County Office of
Education      FAME Public Charter     County
Office of Education (COE)        K-12 Schools
(Public)   Traditional     None    1
0728    Directly funded 1       K       12
1087.0  565.0   0.519779208831647       715.0
0.657773689052438       1070.0  553.0
0.516822429906542       702.0
0.65607476635514        1
01100170112607  2014-2015       01      10017
0112607 Alameda Alameda County Office of
Education      Envision Academy for Arts &
Technology  County Office of Education (COE)
High Schools (Public)   Traditional     None
1       0811    Directly funded 1       9
12      395.0   186.0   0.470886075949367
186.0   0.470886075949367       376.0   182.0
0.484042553191489       182.0
0.484042553191489       1
01100170118489  2014-2015       01      10017
0118489 Alameda Alameda County Office of
Education      Aspire California College
Preparatory Academy   County Office of
Education (COE)        High Schools (Public)
Traditional     None    1       1049
Directly funded 1       9       12      244.0
```

```
134.0   0.549180327868853       175.0
0.717213114754098       230.0   128.0
0.556521739130435       168.0
0.730434782608696       1
*/


CREATE TABLE satscores (
        cds TEXT NOT NULL,
        rtype TEXT NOT NULL,
        sname TEXT,
        dname TEXT,
        cname TEXT,
        enroll12 INTEGER NOT NULL,
        "NumTstTakr" INTEGER NOT NULL,
        "AvgScrRead" INTEGER,
        "AvgScrMath" INTEGER,
        "AvgScrWrite" INTEGER,
        "NumGE1500" INTEGER,
        PRIMARY KEY (cds),
        FOREIGN KEY(cds) REFERENCES schools
("CDSCode")
)

/*
3 rows from satscores table:
cds     rtype   sname   dname   cname
enroll12        NumTstTakr      AvgScrRead
AvgScrMath      AvgScrWrite     NumGE1500
```

```
01100170000000  D       None    Alameda
County Office of Education      Alameda 398
88      418     418     417     14
01100170109835  S       FAME Public Charter
Alameda County Office of Education
Alameda 62      17      503     546     505
9
01100170112607  S       Envision Academy for
Arts & Technology  Alameda County Office of
Education      Alameda 75      71      397
387     395     5
*/


CREATE TABLE schools (
        "CDSCode" TEXT NOT NULL,
        "NCESDist" TEXT,
        "NCESSchool" TEXT,
        "StatusType" TEXT NOT NULL,
        "County" TEXT NOT NULL,
        "District" TEXT NOT NULL,
        "School" TEXT,
        "Street" TEXT,
        "StreetAbr" TEXT,
        "City" TEXT,
        "Zip" TEXT,
        "State" TEXT,
        "MailStreet" TEXT,
        "MailStrAbr" TEXT,
```

```
"MailCity" TEXT,
"MailZip" TEXT,
"MailState" TEXT,
"Phone" TEXT,
"Ext" TEXT,
"Website" TEXT,
"OpenDate" DATE,
"ClosedDate" DATE,
"Charter" INTEGER,
"CharterNum" TEXT,
"FundingType" TEXT,
"DOC" TEXT NOT NULL,
"DOCType" TEXT NOT NULL,
"SOC" TEXT,
"SOCType" TEXT,
"EdOpsCode" TEXT,
"EdOpsName" TEXT,
"EILCode" TEXT,
"EILName" TEXT,
"GSoffered" TEXT,
"GSserved" TEXT,
"Virtual" TEXT,
"Magnet" INTEGER,
"Latitude" REAL,
"Longitude" REAL,
"AdmFName1" TEXT,
"AdmLName1" TEXT,
"AdmEmail1" TEXT,
"AdmFName2" TEXT,
```

```
        "AdmLName2" TEXT,
        "AdmEmail2" TEXT,
        "AdmFName3" TEXT,
        "AdmLName3" TEXT,
        "AdmEmail3" TEXT,
        "LastUpdate" DATE NOT NULL,
        PRIMARY KEY ("CDSCode")
)

/*
3 rows from schools table:
CDSCode NCESDist        NCESSchool
StatusType      County  District
School  Street  StreetAbr       City    Zip
State   MailStreet      MailStrAbr
MailCity        MailZip MailState       Phone
Ext     Website OpenDate        ClosedDate
Charter CharterNum      FundingType     DOC
DOCType SOC     SOCType EdOpsCode
EdOpsName       EILCode EILName GSoffered
GSserved        Virtual Magnet  Latitude
Longitude       AdmFName1       AdmLName1
AdmEmail1       AdmFName2       AdmLName2
AdmEmail2       AdmFName3       AdmLName3
AdmEmail3       LastUpdate
01100170000000  0691051 None    Active
Alameda Alameda County Office of Education
None    313 West Winton Avenue  313 West
Winton Ave.    Hayward 94544-1136      CA
```

```
313 West Winton Avenue  313 West Winton Ave.
Hayward 94544-1136      CA      (510)
887-0152  None    www.acoe.org    None
None    None    None    None    00
County Office of Education (COE)        None
None    None    None    None    None    None
None    None    None    37.658212
-122.09713      L Karen Monroe
lkmonroe@acoe.org       None    None    None
None    None    None    2015-06-23
01100170109835  0691051 10546   Closed
Alameda Alameda County Office of Education
FAME Public Charter     39899 Balentine
Drive, Suite 335        39899 Balentine Dr.,
Ste. 335   Newark  94560-5359      CA
39899 Balentine Drive, Suite 335        39899
Balentine Dr., Ste. 335   Newark  94560-5359
CA      None    None    None    2005-08-29
2015-07-31      1       0728    Directly
funded 00      County Office of Education
(COE)        65      K-12 Schools (Public)
TRAD    Traditional     ELEMHIGH
Elementary-High Combination     K-12    K-12
P       0       37.521436       -121.99391
None    None    None    None    None    None
None    None    None    2015-09-01
01100170112607  0691051 10947   Active
Alameda Alameda County Office of Education
Envision Academy for Arts & Technology  15152
```

```
Webster Street     1515 Webster St.
Oakland 94612-3355      CA      1515 Webster
Street     1515 Webster St.        Oakland
94612   CA      (510) 596-8901  None
www.envisionacademy.org/        2006-08-28
None    1       0811    Directly funded 00
County Office of Education (COE)        66
High Schools (Public)   TRAD    Traditional
HS      High School     9-12    9-12    N
0       37.80452        -122.26815      Laura
Robell  laura@envisionacademy.org       None
None    None    None    None    None
2015-06-18
*/
```

CONVERT NATURAL LANGUA
QUESTIONS INTO SQL

LLMs are quite capable of writing functional SQL queries, from the `code` or `coder` ones, to specific ones for SQL generation.

For example, so
include:

- granite-20b-
- granite-34b-
- granite-20b-
- granite-20b-

More info on th
🤗

PROMPTS FOR SQL GENERATION

LLMs don't know the 🏗️structure of our d
hallucinate about it, or create some flat ou

We have to provide **extra** information abo
the instructions.

For this we will use a **prompt** string with s
holders { }.

Some research papers from our team from

📄 Weakly Supervised Detection of Hallu

📄 Localizing Persona Representations In

PROMPTS

```
system_message = """
Given an input question, create a syntactically co
run to help find the answer. Unless the user speci
specific number of examples they wish to obtain, a
at most {top_k} results. You can order the results
return the most interesting examples in the databa

Never query for all the columns from a specific ta
few relevant columns given the question.

Pay attention to use only the column names that yo
description. Be careful to not query for columns t
pay attention to which column is in which table.

Only use the following tables:
{table_info}
"""
```

source

CREATING A PROMPT

Now we construct a list of messages. These are `dicts` which
`system`, and a `content`.

```
def generate_messages(question, dialect="SQL", top_k=5, table_info=""):
    # Create a ChatPromptTemplate
    messages = [
      {
        "role": "system",
        "content": system_message.format(
          dialect=dialect,
          top_k=top_k,
          table_info=table_info
        )},
      {
        "role": "user",
        "content": question
      }
    ]

    return messages

messages = generate_messages(
  question=question,
  dialect=db.dialect, top_k=10,
  table_info=db.get_table_info())
messages
```

CREATING A PROMPT

```
[{'role': 'system',
  'content': '\nGiven an input question, create a syntactically correct sqlit
answer. Unless the user specifies in his question a\nspecific number of examp
limit your query to\nat most 10 results. You can order the results by a relev
interesting examples in the database.\n\nNever query for all the columns from
the\nfew relevant columns given the question.\n\nPay attention to use only th
the schema\ndescription. Be careful to not query for columns that do not exis
column is in which table.\n\nOnly use the following tables:\n\nCREATE TABLE f
\n\t"Academic Year" TEXT, \n\t"County Code" TEXT, \n\t"District Code" INTEGER
\n\t"County Name" TEXT, \n\t"District Name" TEXT, \n\t"School Name" TEXT, \n\
Type" TEXT, \n\t"Educational Option Type" TEXT, \n\t"NSLP Provision Status" T
INTEGER, \n\t"Charter School Number" TEXT, \n\t"Charter Funding Type" TEXT, \
TEXT, \n\t"High Grade" TEXT, \n\t"Enrollment (K-12)" REAL, \n\t"Free Meal Cou
Eligible Free (K-12)" REAL, \n\t"FRPM Count (K-12)" REAL, \n\t"Percent (%) El
\n\t"Enrollment (Ages 5-17)" REAL, \n\t"Free Meal Count (Ages 5-17)" REAL, \n
5-17)" REAL, \n\t"FRPM Count (Ages 5-17)" REAL, \n\t"Percent (%) Eligible FRP
CALPADS Fall 1 Certification Status" INTEGER, \n\tPRIMARY KEY ("CDSCode"), \n
schools ("CDSCode")\n)\n\n/*\n3 rows from frpm table:\nCDSCode\tAcademic Year
Code\tSchool Code\tCounty Name\tDistrict Name\tSchool Name\tDistrict Type\tSc
Type\tNSLP Provision Status\tCharter School (Y/N)\tCharter School Number\tCha
```

🔎THE `system` MESSAGE

```
Given an input question, create a
syntactically correct sqlite query to
run to help find the answer. Unless the user
specifies in his question a
specific number of examples they wish to
obtain, always limit your query to
at most 10 results. You can order the results
by a relevant column to
return the most interesting examples in the
database.

Never query for all the columns from a
specific table, only ask for a the
few relevant columns given the question.

Pay attention to use only the column names
that you can see in the schema
description. Be careful to not query for
columns that do not exist. Also,
pay attention to which column is in which
table.

Only use the following tables:

CREATE TABLE frpm (
        "CDSCode" TEXT NOT NULL,
```

CALLING THE LLM WITH THE PROMPT

`litellm` is a client for multiple LLM provi

```
1 !uv add litellm --quiet
```

To run inference, we just call the `complet` function:

```
1 import litellm
2
3 # model = "ollama_chat/granite-code:20b"
4 model = "watsonx/ibm/granite-3-2-8b-instruct"
5
6 response = litellm.completion(
7   model=model,
8   messages=messages,
9 )
```

```
print(response)
```

```
ModelResponse(
    id='chatcmpl-f79a3df0-2506-4e7e-aa2b-b65d
21f5d680---5976a0e7cf57f032504389f213b02431--
-e4f953c6-f9cd-4161-bccf-1f1fd819ea7b',
    created=1758486227,
    model='watsonx/ibm/granite-3-2-8b-instruc
t',
    object='chat.completion',
    system_fingerprint=None,
    choices=[
        Choices(
            finish_reason='stop',
            index=0,
            message=Message(
                content='SELECT COUNT(*) FROM
schools\nINNER JOIN frpm ON schools.CDSCode =
frpm.CDSCode\nINNER JOIN satscores ON
schools.CDSCode = satscores.cds\nWHERE
satscores."AvgScrMath" > 400\nAND
schools.Virtual = \'Y\'\nAND schools.Magnet =
0\nAND frpm.CharterSchool(Y/N) = 0\nLIMIT
10;',
                role='assistant',
                tool_calls=None,
                function_call=None,
```

STRUCTURED OUTPUT

Now that we get the SQL, we're going to ask the LLM to retu
using a Pydantic model.

```
from pydantic import BaseModel, Field
class SQLOutput(BaseModel):
    sql: str = Field(description="The SQL query")
    explanation: str = Field(
      description="The reasoning for the query construction")

# Optional
# litellm.enable_json_schema_validation = True

response = litellm.completion(
  model=model,
  messages=messages,
  response_format=SQLOutput,
)
output = SQLOutput.model_validate_json(
  response.choices[0].message.content)
output
```

STRUCTURED OUTPUT

```
SQLOutput(sql="SELECT COUNT(*) FROM schools INNER JOIN satscores ON schools.C
schools.Virtual = 'Y' AND satscores.AvgScrMath > 400;", explanation="The quer
schools from the 'schools' table that have a 'Virtual' column value of 'Y' an
Math) in the 'satscores' table greater than 400. It uses a JOIN to combine da
common 'CDSCode' column.")
```

Now we can use the LLM like we were get

```
output.sql
```

```
"SELECT COUNT(*) FROM schools INNER JOIN satscores ON
satscores.cds WHERE schools.Virtual = 'Y' AND satscore
```

```
output.explanation
```

```
"The query selects the count of virtual schools from t
have a 'Virtual' column value of 'Y' and an 'AvgScrMat
Math) in the 'satscores' table greater than 400. It us
data from both tables based on the common 'CDSCode' co
```

```
Result.fetchall() takes 1 positional argument
but 2 were given
```

CREATE A WORKFLOW

CHAINING GENERATION AN

When our code starts to become large
handle 🏋️‍♀️, `LangGraph` is a great tool to pro
reusable organization. We can add it with
`langgraph`.

We will build a small pipeline where to run
and execution.

Each node in the pipeline will be a function
state object and returns another state obj

STATE FOR SQL WORKFLOW

LangGraph uses state that is propagated t

This state can be defined with a TypedDic
BaseModel.

```
from typing import Optional, Dict, List
from dataclasses import dataclass, field
from sqlalchemy import Engine

@dataclass
class State:
    question: str = ""
    engine: Optional[Engine] = None
    database: Optional[SQLDatabase] = None
    messages: List[Dict[str, str]] = field(default
    sql: Optional[str] = None
    results: Optional[str] = None
```

CREATING NODES

The initial node will receive the user input
state.

```
def init(user_input: Dict[str, str]) -> State:
  engine = create_engine(user_input["database"])
  database = SQLDatabase(engine=engine)
  return State(
    question=user_input["question"],
    database=database,
  )
```

GENERATING THE PROMPT

```
1  def make_prompt(state: State) -> State:
2      messages = [
3        {
4          "role": "system",
5          "content": system_message.format(
6            dialect=state.database.dialect,
7            top_k=top_k,
8            table_info=state.database.get_table_info
9          )},
10       {
11         "role": "user",
12         "content": state.question,
13       }
14     ]
15     return state
```

CALLING THE LLM

The initial node will receive the user input
state.

```
def generate(state: State) -> State:

  response = litellm.completion(
    model=model,
    messages=messages,
    response_format=SQLOutput,
  )
  output = SQLOutput.model_validate_json(response.
  print(output)
  engine = create_engine(user_input["database"])
  database = SQLDatabase(engine=engine)
  return State(
    question=user_input["question"],
    database=database,
  )
```

EXECUTING THE SQL

The initial node will receive the user input
state.

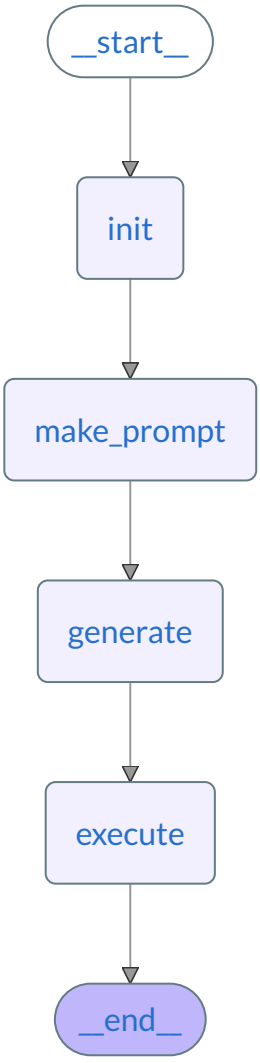
```
1 def execute(state: State) -> State:
2   with state.engine.connect() as con:
3     state.result = conn.execute(text(state.sql))
4
5   return State(
6     question=user_input["question"],
7     database=database,
8   )
```

GRAPH CREATION

```
from langgraph.graph import StateGraph, START, END

graph_builder = StateGraph(State)
graph_builder.add_node("init", init)
graph_builder.add_node("make_prompt", make_prompt)
graph_builder.add_node("generate", generate)
graph_builder.add_node("execute", execute)
graph_builder.add_edge(START, "init")
graph_builder.add_edge("init", "make_prompt")
graph_builder.add_edge("make_prompt", "generate")
graph_builder.add_edge("generate", "execute")
graph = graph_builder.compile()
```

VISUALIZING THE GRAPH 🖧



LESSONS LEARNED

MOVE FROM FUNCTION TO CLASSES

- The workflow state can be a `dict` or `TypedDict`, b
  `BaseModel` is more convenient.

```
class State(TypedDict):
  connection: ...
  messages: ...
  schema: ...
```

- A simple function that works with the state.

```
def create_sql(state: State) -> State:
  # Do something
  return State(...)
```

- This can a
  to pass m

```
class Creat
  def __ini
    self.co

  def __cal
    self, s
  ) -> Stat
    # do so
    return
```

SQL VALIDATION

- We can pass the SQL to SQLAlchemy er
the syntax is correct
- SQLGlot `parse_one()` provides asn AST
  - Finding dangerous operations (DML)
  - For any updates to the query, workin
better than string substitution.

ENHANCING THE CONTEXT

- Tables with a high number of columns c
context larger
  - RAG at the column level is a common
improve efficiency and accuracy
- Retrieving the values from the DB can h
produce better queries
- Providing some examples in the context
`question` and `SQL` pairs also improves e

DYNAMIC CONTEXT

- Other types of examples of well unders
  require some mapping.
  - For example, the term Q1 can mean,
    should be supplied dynamically.
- Function nodes, when we want to modi
  another approach is to create
  - WASM?

EXTENSIBILITY

- Offload node responsibilities into exter
(e.g. DB execution)
- Pipelines defined as YAML.
- Each node is registered in a collection o
node in the pipeline accepts the `__init`
- Use Python's built in `entry-point` syste
This is a good approach to separate pro

THANK YOU