



# 大模型 – Text2SQL

作者: Calvin

QQ: 179209347

Mail: 179209347@qq.com

# 介绍

## 笔记简介:

- 面向对象: 深度学习初学者
- 依赖课程: **线性代数, 统计概率**, 优化理论, 图论, 离散数学, 微积分, 信息论

## 知乎专栏:

<https://zhuanlan.zhihu.com/p/693738275>

## Github & Gitee 地址:

[https://github.com/mymagicpower/AIAS/tree/main/deep\\_learning](https://github.com/mymagicpower/AIAS/tree/main/deep_learning)

[https://gitee.com/mymagicpower/AIAS/tree/main/deep\\_learning](https://gitee.com/mymagicpower/AIAS/tree/main/deep_learning)

## \* 版权声明:

- 仅限用于个人学习
- 禁止用于任何商业用途

## 问题背景

编写 SQL 语句是很常见的应用开发场景。然而，不同的数据库在 SQL 增删改查操作语法、类型字段和引号使用等方面存在一些细微差异，导致实际开发过程中容易犯错。就算是有经验的工程师，在面对复杂的联表查询、分组、排序等组合实现时，也往往需要花费大量心思调试和排错。

通过使用大模型，可以快速生成 SQL 语句，无需具备 SQL 编程的经验，节省了编写 SQL 语句的时间和精力。此外，与手动编写 SQL 语句相比，使用大模型生成的 SQL 语句也相对更准确和稳定，可以帮助减少错误和漏洞的出现。

### SQL 语法

例如，不同数据库获取当前系统时间的语法不同：

- MySQL 使用 now() 和 sysdate()
- SQL Server 使用 getdate()
- 又比如，不同数据库拼接字符串的语法不同：
- MySQL 使用 concat() 函数
- SQL Server 使用 “+” 号
- Oracle 则使用 “||” 号

### 数据类型

例如，不同数据库支持的特殊数据类型不同：

- MySQL 支持 BOOLEAN 和 SET 类型
- SQL Server 则支持 SMALL DATETIME 和 MONEY 等类型
- 又比如，同样是 FLOAT 类型，也存在细微的差异：
- MySQL 中表示单精度浮点数
- SQL Server 中表示双精度浮点数



# SQL领域细分任务

在SQL领域，大模型可以用于各种细分任务。以下是一些常见的任务：

- **SQL查询生成 ( Text2SQL )**：大模型可以接收自然语言查询，并生成对应的SQL查询语句。这是最常见的任务，涉及将自然语言转换为SQL查询的语法和结构。
- **SQL查询解析**：大模型可以接收SQL查询语句，并理解其语义和结构。这可以用于解析和理解复杂的SQL查询，从中提取信息或执行其他操作。
- **SQL查询优化**：大模型可以分析给定的SQL查询，并提供优化建议。这可以包括重写查询以改进性能、选择索引或重新组织查询计划等。
- **数据库查询结果解释**：大模型可以解释SQL查询的结果，并提供相关的上下文信息。这可以帮助用户理解查询结果、推断数据关系或提供解释性的分析。
- **数据库查询验证**：大模型可以验证给定的SQL查询是否符合数据库模式和约束。这可以用于检查查询的语法正确性、表连接的有效性等。
- **数据库查询安全性分析**：大模型可以分析SQL查询，并检测潜在的安全漏洞或风险。这可以帮助保护数据库免受SQL注入等攻击。
- **数据库查询语义理解**：大模型可以理解SQL查询的语义含义，并提供更高级别的查询理解和处理。这可以包括理解查询的目的、意图和上下文等。

这些任务不是相互独立的，它们可能会相互关联并在实际应用中结合使用。此外，这只是SQL领域中的一些细分任务示例，还有其他可能的任务和应用。

## SQL查询生成 – Text2SQL

- Text-to-SQL (或者Text2SQL), 顾名思义就是把文本转化为SQL语言, 更学术一点的定义是: 把数据库领域下的自然语言 (Natural Language, NL) 问题, 转化为在关系型数据库中可以执行的结构化查询语言 (Structured Query Language, SQL), 因此Text-to-SQL也可以被简写为NL2SQL。

输入自然语言问题, 比如:

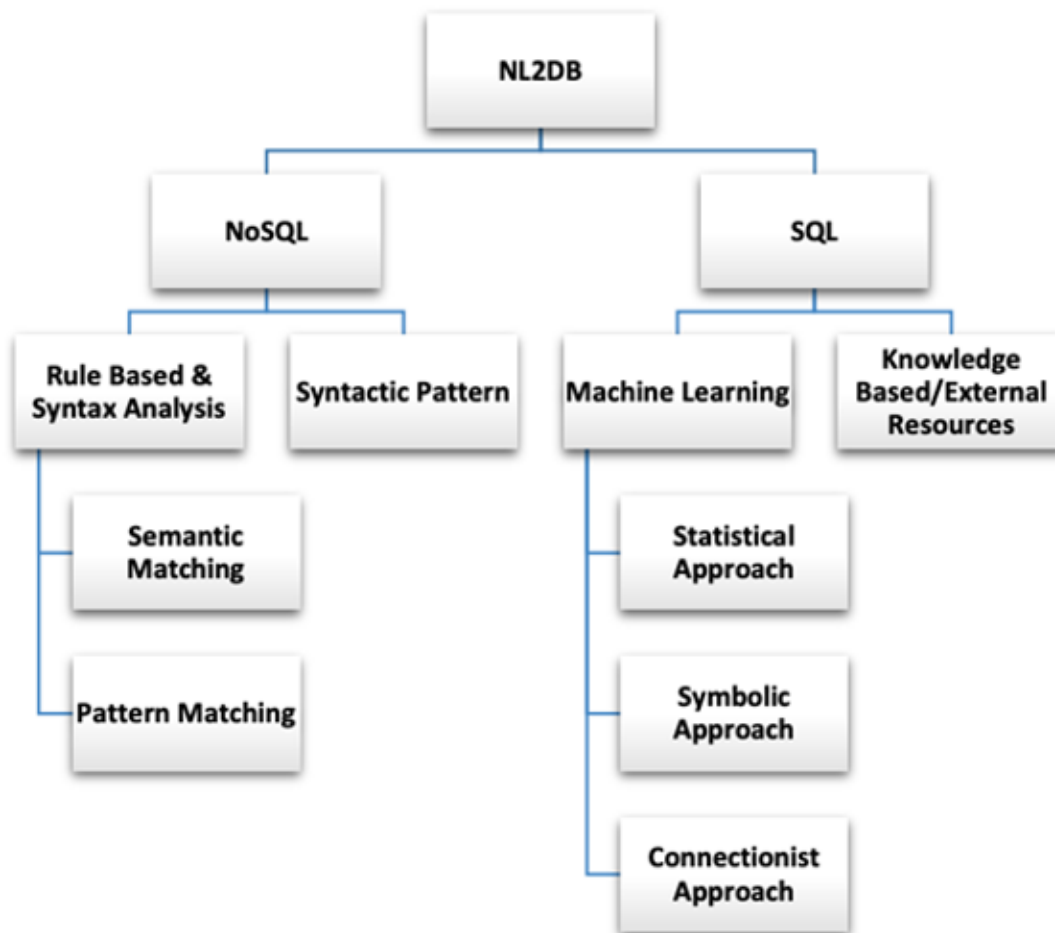
查询表t\_user的相关信息, 结果按id降序排序, 只保留前10个数据

输出: SQL, 比如:

```
SELECT * FROM t_user ORDER BY id DESC LIMIT 10
```


# SQL查询生成 – Text2SQL

- SQL (结构化查询语言) 是用于查询数据库和检索数据的最常用的语言之一。借助机器学习和基于知识的资源, 可以实现文本语言到SQL的转换。在NoSQL类别中, 还包括基于规则的语法分析、语义匹配和模式匹配等各种其他方法。然而, 在本章节中, 我们仅限于将自然语言转换为SQL的研究。



# Text2SQL – 模型选型

- SQLCoder 是 Defog 团队推出的一款前沿的大语言模型，专门用于将**自然语言问题转化为 SQL 查询**。在自然语言到 SQL 生成任务上，其性能超过了 gpt-3.5-turbo，并且显著地超越了所有流行的开源模型。
- SQLCoder-34B是在基于CodeLlama模型的基础上进行了微调。

任务类型	大模型	参数	授权	说明
SQL生成 Text2SQL	 SQLCoder	<ul style="list-style-type: none"> <li>defog-sqlcoder-34b</li> <li>defog-sqlcoder2-15b</li> <li>defog-sqlcoder-7b</li> </ul>	开源	Defog的SQLCoder是一种LLM（语言模型）用于将自然语言问题转换为SQL查询。SQLCoder-34B是在基于CodeLlama模型的基础上进行了微调。
多任务	GPT	<ul style="list-style-type: none"> <li>gpt4-turbo</li> <li>gpt4</li> <li>gpt-3.5</li> </ul>	商用	GPT可以用于生成SQL查询语句。GPT能够理解问题的语义，并生成与之相匹配的SQL语句等多种任务。
需微调	Code LLaMa	<ul style="list-style-type: none"> <li>7B、13B 和 34B</li> </ul>	开源	基于LLaMa2，主要以Python等为主，不太擅长写SQL
需微调	WizardCoder	<ul style="list-style-type: none"> <li>15B、34B</li> </ul>	开源	基于LLama2，不擅长写SQL
需微调	StarCoder	<ul style="list-style-type: none"> <li>15B</li> </ul>	开源	小型的通用代码生成模型，不擅长写SQL

# Text2SQL – 数据集

数据集	问题和SQL对	数据库数量	领域	表格数/数据库	单/多轮	语言
ATIS	5280	1	单领域	32	单轮	英文
GeoQuery	877	1	单领域	6	单轮	英文
Scholar	817	1	单领域	7	单轮	英文
Academic	196	1	单领域	15	单轮	英文
IMDB	131	1	单领域	16	单轮	英文
Yelp	128	1	单领域	7	单轮	英文
Advising	3898	1	单领域	10	单轮	英文
Restaurants	378	1	单领域	3	单轮	英文
WikiSQL	80657	26521	多领域	1	单轮	英文
NL2SQL	49974	5291	多领域	1	单轮	中文
Spider	10181	200	多领域	5.1	单轮	英文
CSpider	9691	166	多领域	5.3	单轮	中文
SParc	4298	200	多领域	5.1	多轮	英文
CoSQL	3007	200	多领域	5.1	多轮	英文



# Text2SQL – 典型数据集介绍

数据集	说明	地址
WikiSQL	<b>入门数据集:</b> WikiSQL是一个大型的语义解析数据集，由80,654个自然语句表述和24,241张表格的sql标注构成。WikiSQL中每一个问句的查询范围仅限于同一张表，不包含排序、分组、子查询等复杂操作。虽然数据规模大，SQL语法却非常简单；适合做NL2SQL任务入门。	<a href="https://huggingface.co/datasets/wikisql">https://huggingface.co/datasets/wikisql</a>
Spider	<b>难度最大数据集:</b> 耶鲁大学在2018年新提出的一个大规模的NL2SQL (Text-to-SQL) 数据集。该数据集包含了10,181条自然语言问句、分布在200个独立数据库中的5,693条SQL，内容覆盖了138个不同的领域。涉及的SQL语法最全面，是目前难度最大的NL2SQL数据集。	<a href="https://yale-lily.github.io/spider">https://yale-lily.github.io/spider</a>
Cspider 中文	CSpider是 <b>Spider的中文版</b> ，西湖大学出品。美中不足的是，数据集只是翻译了Spider的question部分，表格列名等仍是英文，需要额外处理对齐。	<a href="https://taolusi.github.io/CSpider-explorer/">https://taolusi.github.io/CSpider-explorer/</a>
NL2SQL 中文	<b>中文NL2SQL数据集:</b> 2020年之前公开的Text2SQL数据集中唯一一份高质量的中文数据集。数据集使用金融以及通用领域的表格数据作为数据源，提供在此基础上人工标注的自然语言和SQL语句的匹配对。	<a href="https://tianchi.aliyun.com/competition/entrance/231716/information">https://tianchi.aliyun.com/competition/entrance/231716/information</a>
SparC	<b>多轮交互Text2SQL:</b> 耶鲁大学在2019年提出的基于对话的Text-to-SQL数据集。SParC是一个跨域上下文语义分析的数据集，是Spider任务的上下文交互版本。SParC由4298个对话（12k+个单独的问题，每个对话平均4-5个子问题，由14个耶鲁学生标注）组成，这些问题通过用户与138个领域的200个复杂数据库进行交互获得。	<a href="https://yale-lily.github.io/sparc">https://yale-lily.github.io/sparc</a>

# Text2SQL – 典型数据集介绍

数据集	说明	地址
CoSQL	<b>多轮交互Text2SQL</b> : 耶鲁大学在2019年提出的基于对话的Text-to-SQL数据集。内容和Sparc相似，但是标注风格略有不同，例如数据集中SQL各关键字的分布差异较大。	<a href="https://yale-lily.github.io/cosql">https://yale-lily.github.io/cosql</a>
CHASE 中文	<b>多轮交互中文Text2SQL</b> : 2021年，微软亚研院和北航、西安交大联合提出的首个大规模上下文依赖的Text-to-SQL中文数据集。内容分为CHASE-C和CHASE-T两部分，CHASE-C从头标注实现，CHASE-T将Sparc从英文翻译为中；相比以往数据集，CHASE大幅增加了hard类型的数据规模，减少了上下文独立样本的数据量，弥补了Text2SQL多轮交互任务中文数据集的空白。	<a href="https://xjtu-intsoft.github.io/chase/">https://xjtu-intsoft.github.io/chase/</a>
TableQA 中文	2020年6月，追一科技公司提出了一个大规模跨领域Text-to-SQL数据集TableQA，其中包含64891个问题和6000多个表的20311个唯一SQL查询。	<a href="https://www.luge.ai/#/luge/dataDetail?id=12">https://www.luge.ai/#/luge/dataDetail?id=12</a>
DuSQL 中文	2020年11月，百度针对跨域文本到SQL任务提出了一个大规模、实用的中文数据集DuSQL，它包含200个数据库、813个表和23797个Question-SQL对。	<a href="https://www.luge.ai/#/luge/dataDetail?id=13">https://www.luge.ai/#/luge/dataDetail?id=13</a>
BIRD-SQL 中文	2023年5月，香港大学和阿里巴巴提出了一个大规模跨域数据集BIRD，其中包含超过12751个独特的问题 SQL、95个大数据库，总大小为33.4GB。它还涵盖区块链、曲棍球、医疗保健和教育等超过37个专业领域。	<a href="https://bird-bench.github.io/">https://bird-bench.github.io/</a>

## Text2SQL – 评测指标 (1/3)

名称	说明	论文
<b>Execution Accuracy (EX)</b>	<ul style="list-style-type: none"><li>计算SQL执行结果正确的数量在数据集中的比例，结果存在高估的可能。</li></ul>	<a href="https://arxiv.org/pdf/2208.13629.pdf">https://arxiv.org/pdf/2208.13629.pdf</a>
<b>Exact Match (EM)</b>	<ul style="list-style-type: none"><li>计算模型生成的SQL和标注SQL的匹配程度，结果存在低估的可能。</li></ul>	
<b>Question Match Accuracy (QM)</b>	<ul style="list-style-type: none"><li>问题匹配准确率是通过所有问题的EM分数来计算的。只有当所有预测的SQL子句都正确时，每个问题的值才为1。</li></ul>	
<b>Interaction Match Accuracy (IM)</b>	<ul style="list-style-type: none"><li>交互匹配准确度是通过对所有交互（问题序列）进行EM得分计算的。每个交互的得分只有在交互内的所有问题都正确时才为1。</li></ul>	

## Text2SQL – 评测指标 (2/3)

阿里达摩院联合HKU推出Text-to-SQL新基准 BIRD 重点关注:

- 海量且真实的数据库内容
- 自然语言问题与数据库内容之间的外部知识推理
- 在处理大型数据库时 SQL 的效率

评价指标	说明	网址
执行准确率	<ul style="list-style-type: none"><li>对比模型预测的 SQL 执行结果与真实标注 SQL 执行结果的差异;</li></ul>	论文: <a href="https://arxiv.org/abs/2305.03111">https://arxiv.org/abs/2305.03111</a>
有效效率分数	同时考虑 SQL 的准确性与高效性, 对比模型预测的 SQL 执行速度与真实标注 SQL 执行速度的相对差异, 将运行时间视为效率的主要指标。	主页: <a href="https://bird-bench.github.io">https://bird-bench.github.io</a>  代码: <a href="https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/bird">https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/bird</a>

# Text2SQL – 评测指标 (3/3)

测评的两类问题:

- 基础问题类型 (Fundamental Type), 包括传统 Text-to-SQL 数据集中涵盖的问题类型
- 推理问题类型 (Reasoning Type), 推理问题类型则包括需要外部知识来理解值的问题

Question Type	Sub Type	Question / SQL	Percentage	Reasoning Type	Domain Knowledge		
Fundamental Type	Match-based	How many gas stations in <b>CZE</b> has <b>Premium</b> gas?  SELECT COUNT(GasStationID) FROM gasstations WHERE Country = ' <b>CZE</b> ' AND Segment = ' <b>Premium</b> '	83.9 %		Name the ID and age of patient with two or more laboratory examinations which show their <b>hematoclit level exceeded the normal range</b> .  SELECT T1.ID, STRFTIME('%Y', CURRENT_TIMESTAMP) - STRFTIME('%Y', T1.Birthday) FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T1.ID IN ( SELECT ID FROM Laboratory WHERE <b>HCT &gt; 52</b> GROUP BY ID HAVING COUNT(ID) >= 2 )	23.6 %	
	Ranking	What are the titles of the <b>top 5</b> posts with the highest popularity?  SELECT Title FROM posts <b>ORDER BY</b> ViewCount DESC <b>LIMIT 5</b>	20.3 %				
	Comparison	How many color cards with no borders have been ranked <b>higher than 12000</b> on EDHRec?  SELECT COUNT(id) FROM cards WHERE edhrecRank > <b>12000</b> AND borderColor = 'borderless'	16.7 %		Numeric Computation	Among the posts with a score of over 20, what is the <b>percentage</b> of them being owned by an elder user?  SELECT <b>CAST(SUM(IIF(T2.Age &gt; 65, 1, 0)) AS REAL) * 100 / count(T1.Id)</b> FROM posts AS T1 INNER JOIN users AS T2 ON T1.OwnerUserId = T2.Id WHERE T1.Score > 20	24.5 %
	Counting	<b>How many</b> of the members' hometowns are from Maryland state?  SELECT <b>COUNT(T2.member_id)</b> FROM zip_code AS T1 INNER JOIN member AS T2 ON T1.zip_code = T2.zip WHERE T1.state = 'Maryland'	30.4 %		Synonym	How many clients opened their accounts in Jesenik branch were <b>women ? (female)</b> .  SELECT COUNT(T1.client_id) FROM client AS T1 INNER JOIN district AS T2 ON T1.district_id = T2.district_id WHERE <b>T1.gender = 'F'</b> AND T2.A2 = 'Jesenik'	7.2 %
	Aggregation	What is the <b>average height</b> of the superheroes from Marvel Comics?  SELECT <b>AVG(T1.height_cm)</b> FROM superhero AS T1 INNER JOIN publisher AS T2 ON T1.publisher_id = T2.id WHERE T2.publisher_name = 'Marvel Comics'	15.7 %		Value Illustration	Among the <b>weekly issuance</b> accounts, how many have a loan of under 200000?  SELECT COUNT(T1.account_id) FROM loan AS T1 INNER JOIN account AS T2 ON T1.account_id = T2.account_id WHERE <b>T2.frequency = 'POPLATEK TYDNE'</b> AND T1.amount < 200000	70.1 %



## Text2SQL – 模型训练/微调项目

- 因为大模型的参数量非常大，从头训练一个自己的大模型训练成本非常高；
- 提示词工程的效果达不到要求，通过自有数据微调，更好的提升大模型在特定领域的能力；
- 训练一个轻量级的微调模型，提升特定业务场景个性化服务能力；
- 数据安全的问题；

项目名称	说明	网址
<b>DB-GPT-Hub</b>	eosphoros组织提出的专注于大模型Text-to-SQL微调的开源项目，包含了大模型下载、数据集预处理、LoRA和QLoRA等微调技术、模型预测、模型评估等步骤。	<a href="https://github.com/eosphoros-ai/DB-GPT-Hub">https://github.com/eosphoros-ai/DB-GPT-Hub</a>
<b>sqlcoder</b>	Defog组织提出的先进的Text-to-SQL的大模型，表现亮眼，效果优于GPT3.5、wizardcoder和starcoder等，仅次于GPT4。	<a href="https://github.com/defog-ai/sqlcoder">https://github.com/defog-ai/sqlcoder</a>
<b>modal_finetune_sql</b>	项目基于LLaMa 2 7b模型进行Text-to-SQL微调，有完整的训练、微调、评估流程。	<a href="https://github.com/run-llama/modal_finetune_sql">https://github.com/run-llama/modal_finetune_sql</a>
<b>LLaMA-Efficient-Tuning</b>	这是一个易于使用的LLM微调框架，支持LLaMA-2、BLOOM、Falcon、Baichuan、Qwen、ChatGLM2等。	<a href="https://github.com/hiyouga/LLaMA-Factory">https://github.com/hiyouga/LLaMA-Factory</a>

# 基础模型 (1/2)

模型名称	说明	代码地址	模型地址
☆ <b>Llama</b>	☆ 2023年2月, Meta AI提出开源大模型Llama, 有7b、13b、33b、65b共4种规模。	☆ <a href="https://github.com/facebookresearch/llama">https://github.com/facebookresearch/llama</a>	➤ <a href="https://huggingface.co/meta-llama">https://huggingface.co/meta-llama</a>
☆ <b>ChatGLM</b>	☆ 2023年3月, 清华大学提出了开源的双语大模型ChatGLM, 基于GLM框架, 参数规格为6b。	☆ <a href="https://github.com/THUDM/ChatGLM3">https://github.com/THUDM/ChatGLM3</a>	➤ <a href="https://huggingface.co/THUDM/chatglm3-6b">https://huggingface.co/THUDM/chatglm3-6b</a>
☆ <b>Alpaca</b>	☆ 2023年3月, 斯坦福大学提出基于Llama 7b模型微调的开源大模型Alpaca, 有7b共1种规格, 训练更简单和便宜。	☆ <a href="https://github.com/tatsu-lab/stanford_alpaca">https://github.com/tatsu-lab/stanford_alpaca</a>	➤ <a href="https://huggingface.co/tatsu-lab/alpaca-7b-wdiff">https://huggingface.co/tatsu-lab/alpaca-7b-wdiff</a>
☆ <b>Vicuna</b>	☆ 2023年3月, UC伯克利大学联合CMU、斯坦福大学提出的开源大模型Vicuna, 有7b、13b共2种规格。	☆ <a href="https://github.com/lm-sys/FastChat">https://github.com/lm-sys/FastChat</a>	➤ <a href="https://huggingface.co/lmsys">https://huggingface.co/lmsys</a>
☆ <b>WizardLM</b>	☆ 2023年4月, 北京大学和微软提出进化指令大模型WizardLM, 有7b、13b、30b共3种规格, 2023年6月, 提出了数学领域的大模型WizardMath, 2023年8月提出了代码领域的大模型WizardCoder。	☆ <a href="https://github.com/nlpxucan/WizardLM">https://github.com/nlpxucan/WizardLM</a>	➤ <a href="https://huggingface.co/WizardLM">https://huggingface.co/WizardLM</a>
☆ <b>Falcon</b>	☆ 2023年6月, 阿联酋提出了大模型Falcon, 这是一种仅在网络数据集上训练的开源大模型, 具有1b、7b、40b和180b四个参数规范。值得注意的是, 其中Falcon 40B的性能超过了LLaMA 65B。	☆ <a href="https://huggingface.co/tiiuae/falcon-180B">https://huggingface.co/tiiuae/falcon-180B</a>	➤ <a href="https://huggingface.co/tiiuae">https://huggingface.co/tiiuae</a>

## 基础模型 (2/2)

模型名称	说明	代码地址	模型地址
☆ InternLM	☆ 2023年7月, 上海人工智能实验室和商汤科技等提出了InternLM, 开源了针对实际场景量身定制的7b和20b参数模型和聊天模型以及训练系统。	☆ <a href="https://github.com/InternLM/InternLM/">https://github.com/InternLM/InternLM/</a>	➤ <a href="https://huggingface.co/internlm">https://huggingface.co/internlm</a>
☆ Llama 2	☆ 2023年7月, Meta AI提出第二代Llama系列开源大模型Llama 2, 和Llama 1相比, 训练数据多40%, 上下文长度翻倍, 模型有7b、13b、34b、70b共4种规格, 但是34b没有开源。	☆ <a href="https://github.com/facebookresearch/llama">https://github.com/facebookresearch/llama</a>	➤ <a href="https://huggingface.co/meta-llama">https://huggingface.co/meta-llama</a>
☆ Code Llama	☆ 2023年8月, Meta AI 在 Llama 2 的基础上提出 Code Llama。Code Llama 在多个代码基准测试中达到了开放模型中最先进的性能。有基础模型 (Code Llama)、Python 专业化 (Code Llama - Python) 和指令跟踪模型 (instruction-following models), 每个模型都有 7B、13B 和 34B 参数。	☆ <a href="https://github.com/facebookresearch/code-llama">https://github.com/facebookresearch/code-llama</a>	➤ <a href="https://huggingface.co/codellama">https://huggingface.co/codellama</a>
☆ Qwen	☆ 2023年8月, 阿里云提出大语言模型系列Qwen-7B (简称通义千问), 在海量数据上进行预训练, 包括网页文本、书籍、代码等, 开源了两个版本Qwen-7B和Qwen-7B-Chat。2023年9月, 阿里云更新了Qwen-7B和Qwen-7B-Chat, 并开源了Qwen-14B和Qwen-14B-Chat。	☆ <a href="https://github.com/QwenLM/Qwen">https://github.com/QwenLM/Qwen</a>	➤ <a href="https://huggingface.co/Qwen">https://huggingface.co/Qwen</a>
☆ Baichuan 2	☆ 2023年9月, 百川智能提出新一代开源大语言模型Baichuan 2, 在2.6万亿个tokens的高质量语料上训练, 有7B和13B的基础版和聊天版, 以及4bits量化版聊天模型。	☆ <a href="https://github.com/baichuan-inc/Baichuan2">https://github.com/baichuan-inc/Baichuan2</a>	➤ <a href="https://huggingface.co/baichuan-inc">https://huggingface.co/baichuan-inc</a>
☆ Phi-1.5	☆ 2023年9月, 微软研究院提出开源语言模型phi-1.5, 一个拥有1.3b个参数的Transformer, 使用与phi-1相同的数据源进行训练, 增加了由各种NLP合成文本组成的新数据源。当根据测试常识、语言理解和逻辑推理的基准进行评估时, phi-1.5在参数少于10b的模型中表现出近乎最先进的性能。		➤ <a href="https://huggingface.co/microsoft/phi-1_5">https://huggingface.co/microsoft/phi-1_5</a>

# 大模型微调方法总结 (1/2)

模型名称	说明	代码	论文
☆ <b>P-Tuning</b>	2021年3月, 清华大学等提出了针对大模型微调方法P-Tuning, 采用可训练的连续提示词嵌入, 降低了微调成本。	☆ <a href="https://github.com/THUDM/P-tuning">https://github.com/THUDM/P-tuning</a>	➤ <a href="https://arxiv.org/pdf/2103.10385.pdf">https://arxiv.org/pdf/2103.10385.pdf</a>
☆ <b>LoRA</b>	2021年6月, 微软提出的针对大模型微调的Low-Rank Adaptation (LoRA) 方法, 冻结预训练权重。	☆ <a href="https://github.com/microsoft/LoRA">https://github.com/microsoft/LoRA</a>	➤ <a href="https://arxiv.org/pdf/2106.09685.pdf">https://arxiv.org/pdf/2106.09685.pdf</a>
☆ <b>P-Tuning V2</b>	2021年10月, 清华大学提出了P-Tuning的改进版本P-Tuning V2, 性能更优。	☆ <a href="https://github.com/THUDM/P-tuning-v2">https://github.com/THUDM/P-tuning-v2</a>	➤ <a href="https://arxiv.org/pdf/2110.07602.pdf">https://arxiv.org/pdf/2110.07602.pdf</a>
☆ <b>RLHF</b>	2022年12月, OpenAI使用RLHF方法训练ChatGPT, 利用人类反馈信号直接优化语言模型, 表现优异。	☆ <a href="https://github.com/huggingface/blog/blob/main/zh/rlhf.md">https://github.com/huggingface/blog/blob/main/zh/rlhf.md</a>	➤ <a href="https://huggingface.co/blog/rlhf">https://huggingface.co/blog/rlhf</a>
☆ <b>RRHF</b>	2023年4月, 阿里巴巴提出了一种新的学习范式称为RRHF (Rank Responses to Align Language Models with Human Feedback without tears), 可以像微调一样轻松调整并实现PPO算法在HH数据集集中的性能。	☆ <a href="https://github.com/GanjinZero/RRHF">https://github.com/GanjinZero/RRHF</a>	➤ <a href="https://arxiv.org/pdf/2304.05302.pdf">https://arxiv.org/pdf/2304.05302.pdf</a>

## 大模型微调方法总结 (2/2)

模型名称	说明	代码	论文
☆ QLoRA	2023年5月, 华盛顿大学基于冻结的4bit量化模型, 结合LoRA方法训练, 进一步降低了微调门槛。	☆ <a href="https://github.com/artidoro/qlora">https://github.com/artidoro/qlora</a>	➤ <a href="https://arxiv.org/pdf/2305.14314.pdf">https://arxiv.org/pdf/2305.14314.pdf</a>
☆ RLTF	2023年7月, 腾讯提出了RLTF (Reinforcement Learning from Unit Test Feedback), 这是一种新颖的online强化学习框架, 具有多粒度的单元测试反馈, 用于细化code LLMs。	☆ <a href="https://github.com/Zyq-scut/RLTF">https://github.com/Zyq-scut/RLTF</a>	➤ <a href="https://arxiv.org/pdf/2307.04349.pdf">https://arxiv.org/pdf/2307.04349.pdf</a>
☆ RRTF	2023年7月, 华为提出RRTF (Rank Responses to align Test&Teacher Feedback)。与 RLHF 相比, RRHF可以有效地将语言模型的输出概率与人类偏好对齐, 调优期间只需要1-2个模型, 并且在实现、超参数调优和训练方面比PPO更简单。		➤ <a href="https://arxiv.org/pdf/2307.14936v1.pdf">https://arxiv.org/pdf/2307.14936v1.pdf</a>
☆ RLAIF	2023年9月, 谷歌提出了RLAIF (来自AI反馈的强化学习RL), 这是一种由现成的LLM代替人类来标记偏好的技术。他们发现RLHF和 RLAIF方法在摘要任务上取得了相似的结果。		➤ <a href="https://arxiv.org/pdf/2309.00267.pdf">https://arxiv.org/pdf/2309.00267.pdf</a>



## Text2SQL - SQLCoder

SQLCoder 是 Defog 团队推出的一款前沿的大语言模型，专门用于将**自然语言问题转化为 SQL 查询**。在自然语言到 SQL 生成任务上，其性能超过了 gpt-3.5-turbo，并且显著地超越了所有流行的开源模型。

SQLCoder-34B是一个拥有34B个参数的模型，在自身的 sql-eval测试框架上对于自然语言转SQL生成任务表现优于 gpt-4和gpt-4-turbo，并且显著优于所有流行的开源模型。

SQLCoder-34B是在基于CodeLlama模型的基础上进行了微调。

大模型	正确率
defog-sqlcoder-34b	84.0
gpt4-turbo-2023-11-09	82.5
gpt4-2023-11-09	82.5
defog-sqlcoder2-15b	77.5
gpt4-2023-08-28	74.0
defog-sqlcoder-7b	71.0
gpt-3.5-2023-10-04	66.0
claude-2	64.5
gpt-3.5-2023-08-28	61.0
claude_instant_1	61.0
text-davinci-003	52.5

# Text2SQL - SQLCoder - 数据库Schema

```
CREATE TABLE products (  
  product_id INTEGER PRIMARY KEY, -- Unique ID for each product  
  name VARCHAR(50), -- Name of the product  
  price DECIMAL(10,2), -- Price of each unit of the product  
  quantity INTEGER -- Current quantity in stock  
);  
  
CREATE TABLE customers (  
  customer_id INTEGER PRIMARY KEY, -- Unique ID for each customer  
  name VARCHAR(50), -- Name of the customer  
  address VARCHAR(100) -- Mailing address of the customer  
);  
  
CREATE TABLE salespeople (  
  salesperson_id INTEGER PRIMARY KEY, -- Unique ID for each salesperson  
  name VARCHAR(50), -- Name of the salesperson  
  region VARCHAR(50) -- Geographic sales region  
);
```

```
CREATE TABLE sales (  
  sale_id INTEGER PRIMARY KEY, -- Unique ID for each sale  
  product_id INTEGER, -- ID of product sold  
  customer_id INTEGER, -- ID of customer who made purchase  
  salesperson_id INTEGER, -- ID of salesperson who made the sale  
  sale_date DATE, -- Date the sale occurred  
  quantity INTEGER -- Quantity of product sold  
);  
  
CREATE TABLE product_suppliers (  
  supplier_id INTEGER PRIMARY KEY, -- Unique ID for each supplier  
  product_id INTEGER, -- Product ID supplied  
  supply_price DECIMAL(10,2) -- Unit price charged by supplier  
);  
  
-- sales.product_id can be joined with products.product_id  
-- sales.customer_id can be joined with customers.customer_id  
-- sales.salesperson_id can be joined with salespeople.salesperson_id  
-- product_suppliers.product_id can be joined with products.product_id
```

<https://defog.ai/sqlcoder-demo/>

# Text2SQL - SQLCoder - 查询语句

## 查询语句:


**中文:** 我们在纽约地区的收入前三名产品是什么?

**英文:** What are our top 3 products by revenue in the New York region?

Ask a question

我们在纽约地区的收入前三名产品是什么?

Verifying...

  
Privacy Terms

Get SQL

Enter special instructions (optional)

- If the question cannot be answered given the database schema, return "I do not know"

Enter your database schema

CREATE TABLE products (  
product\_id INTEGER PRIMARY KEY, -- Unique ID for each product  
name VARCHAR(50), -- Name of the product  
price DECIMAL(10,2), -- Price of each unit of the product  
quantity INTEGER -- Current quantity in stock  
);

SQLCoder's Generated Query

This public facing demo runs a limited version of SQLCoder with greedy search, and will have worse performance than our production. To use an optimized version, please run this on your own servers with num\_beams=5, or [contact us](#) for a demo.

```
SELECT p.name,  
       SUM(s.quantity * s.price) AS total_sales  
FROM sales s  
JOIN products p ON s.product_id = p.product_id  
JOIN salespeople sp ON s.salesperson_id = sp.salesperson_id  
WHERE sp.region = 'New York'  
GROUP BY p.name  
ORDER BY total_sales DESC NULLS LAST  
LIMIT 3;
```

```
SELECT p.name,  
       SUM(s.quantity * s.price) AS total_sales  
FROM sales s  
JOIN products p ON s.product_id = p.product_id  
JOIN salespeople sp ON s.salesperson_id = sp.salesperson_id  
WHERE sp.region = 'New York'  
GROUP BY p.name  
ORDER BY total_sales DESC NULLS LAST  
LIMIT 3;
```

<https://defog.ai/sqlcoder-demo/>

# SQL 查询优化

## 问题背景:

- SQL 查询优化是开发者接触到最多的也是最常用的数据库性能优化手段。
- SQL 查询优化的目标是通过各种手段减少查询的响应时间,降低数据库负载,提高查询效率。
- 在同一数据库环境中,查询执行效率取决于SQL查询语句的质量。
- 许多SQL程序员难以编写高质量的SQL语句,优化复杂查询需要大量时间和精力。

## SQL优化:

- 大语言模型能够分析SQL查询语句,提供查询重写和优化建议。
- 它可以根据查询语句的结构和语义,推断出更高效的查询方式,并快速提供相应的优化建议。
- 极大地减轻了开发人员和维护人员的负担。

SELECT rDelete FROM table\_test WHERE rTopicId = 975312468 AND replyId = (SELECT T rTargetReplyid FROM table\_test WHERE replyId = 131415161718 ) 帮我优化这条SQL 查询语句

为了优化这条SQL查询语句,你可以考虑以下几点:

1. 确保表 `table\_test` 上有适当的索引,以加快查询速度。在 `rTopicId` 和 `replyId` 列上创建索引可能会有帮助。
2. 考虑使用连接 (JOIN) 来替代子查询。这样可以减少查询的复杂性和执行时间。例如,你可以尝试以下查询:

```
sql
SELECT t1.rDelete
FROM table_test t1
JOIN table_test t2 ON t1.replyId = t2.rTargetReplyid
WHERE t1.rTopicId = 975312468 AND t2.replyId = 131415161718
```

这个查询使用了两个表的连接,避免了子查询。

3. 如果可能的话,考虑对表进行分区或分片,以提高查询性能。这取决于你的数据库系统和数据量。

# SQL 提示词工程

## prompt的组成包四个元素:

- Instruction (指令, 必须)
- Context (上下文信息, 可选)
- Input Data (输入, 需要处理的数据, 可选)
- Output Indicator (输出指引, 规定输出的类型或格式, 可选)

面向大语言模型的开发应用过程就是如下公式:

$$\text{LMM}(\text{Instruction} + \text{Context} + \text{Input Data} + \text{Output Indicator}) = \text{Output}$$



# SQL 提示词工程

技巧 1: 增加 Role角色, 然后去下达你的指令。

技巧 2: 描述清楚需要输出的格式, 结构化输出。

技巧 3: few-shot prompt, 给模型一些示例。

技巧 4: 不要用较短的模糊的prompt试图阐明意图, 用较长的更具体的prompt, 可以得到更相关和更详细的输出。

技巧 5: 迭代prompt, 直到达到你的需求

1) 分析为什么结果不满意。

2) 按照前面4条重新阐明。

3) 用小批数据实验prompt。

技巧6: 碰到推理问题时, 让模型多思考一会。

```
> prompt = ""
>     现在你是一个数据分析师,SQL专家,请根据用户提供的表的信息, 以及用户的需求, 写出效率最高的SQL,
>     表信息如下:
>         表名: students;
>         字段: id,name,age,location
>         用户需求:统计一下姓名年龄大于23,姓名包含andy且在beijing,的的学生个数。
>         并且要求输出的SQL以#开头,以#结尾, 样例如下:
>             #SELECT * FROM table#
>             #SELECT COUNT(*) FROM table#
>         注意不需要分析过程, 直接给出SQL语句
>     ""
> inputtext = ""<human>:
>     {}
> <aibot>:
>     "".format(prompt)
```

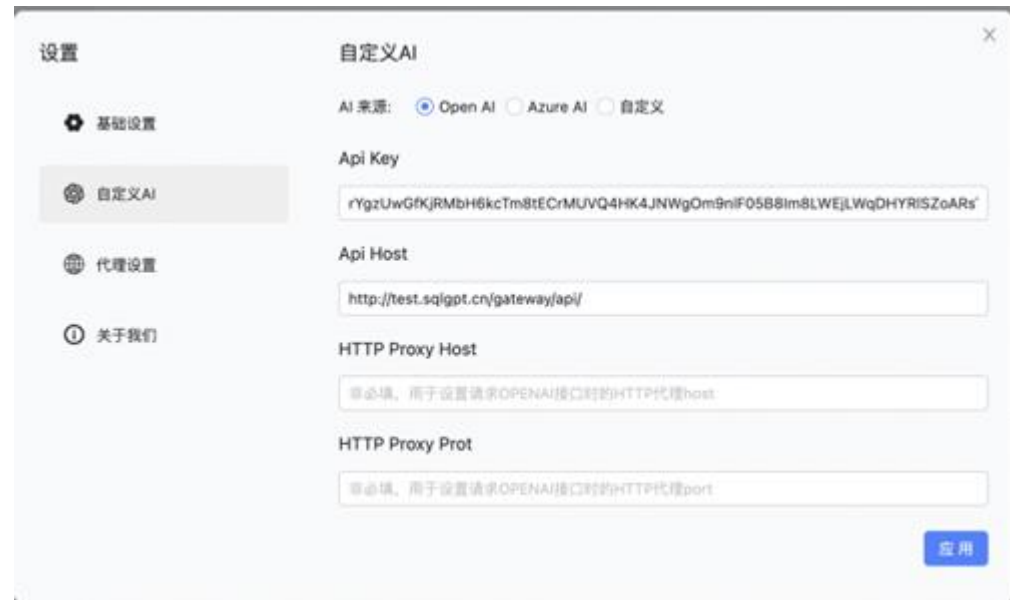
## Text2SQL - 智能SQL客户端选型

项目名称	说明	授权	网址
Chat2DB	Chat2DB 是一款有开源免费的多数据库客户端工具，支持windows、mac本地安装，也支持服务器端部署，web网页访问。和传统的数据库客户端软件Navicat、DBeaver 相比Chat2DB集成了AIGC的能力，能够将自然语言转换为SQL，也可以将SQL转换为自然语言，可以给出研发人员SQL的优化建议，极大的提升人员的效率，是AI时代数据库研发人员的利器，未来即使不懂SQL的运营业务也可以使用快速查询业务数据、生成报表能力。	开源 非商用	<a href="https://github.com/chat2db/Chat2DB">https://github.com/chat2db/Chat2DB</a> <a href="https://doc.sqlgpt.cn/zh/">https://doc.sqlgpt.cn/zh/</a>

# Chat2DB – 样例

<https://sqlgpt.cn/zh>

- AI智能助手，支持自然语言转SQL、SQL转自然语言、SQL优化建议
- SQL查询、AI查询和数据报表完美集成的一体化解决方案设计与实现
- 支持团队协作，研发无需知道线上数据库密码，解决企业数据库账号安全问题
- 强大的数据管理能力，支持数据表、视图、存储过程、函数、触发器、索引、序列、用户、角色、授权等管理
- 强大的扩展能力，目前已经支持MySQL、PostgreSQL、Oracle、SQLServer、ClickHouse、OceanBase、H2、SQLite等等，未来会支持更多的数据库
- 前端使用 Electron 开发，提供 Windows、Mac、Linux 客户端、网页版本一体化的解决方案
- 支持环境隔离、线上、日常数据权限分离



设置

自定义AI

AI 来源: ☒ Open AI ☐ Azure AI ☐ 自定义

Api Key

Api Host

HTTP Proxy Host

HTTP Proxy Port

应用



@db.sqlgpt.cn/ch...

保存记录

表

- order
- order\_item
- product
- t3
- t5
- t\_orderdetail

chat2db-con... X

在DEMO数据库中创建一张学生表，包含学生姓名、性别，建表语句要包含注释、自增主键，请给出完整

1

2 CREATE TABLE student (

3 id INT AUTO\_INCREMENT COMMENT '学生ID',

4 name VARCHAR(50) COMMENT '学生姓名',

5 gender VARCHAR(10) COMMENT '学生性别',

6 PRIMARY KEY (id)

7 ) COMMENT '学生表' ;

8

9

10

> 执行 保存

暂无数据

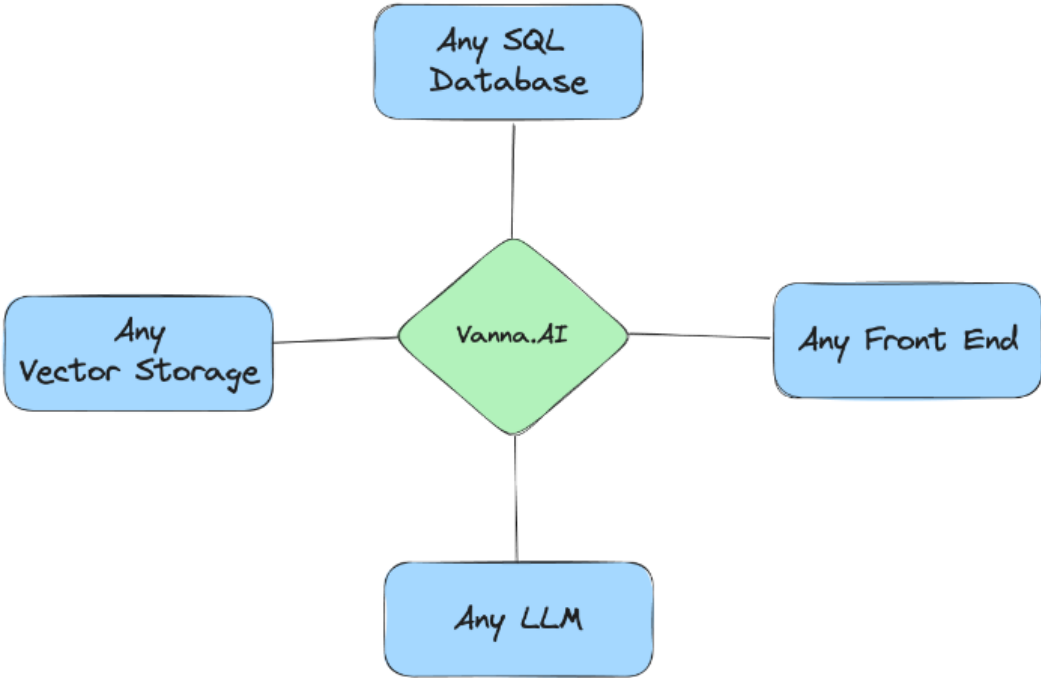
# Text2SQL - 智能SQL框架选型

项目名称	说明	授权	网址
Vanna	Vanna是一个为用户提供更便捷地与SQL数据库交互的工具。它通过允许用户使用自然语言来查询数据库，然后将这些语言转换成SQL语句，从而实现数据库信息的查询和获取。	开源	<a href="https://github.com/vanna-ai/vanna">https://github.com/vanna-ai/vanna</a>

**使用场景：**  
适用于数据分析、数据库开发等场景。

**使用者：**  
Vanna AI为**数据分析师**和**数据库开发人员**提供了强大的工具，简化了SQL的创建流程。无论是初学者还是经验丰富的专业人员，Vanna都能满足需求。

- 特色功能：**  
提供高效的SQL查询语句生成服务，提升数据处理效率和准确性。
- 快速生成SQL查询语句
  - 支持多种数据库
  - 高精度和安全性



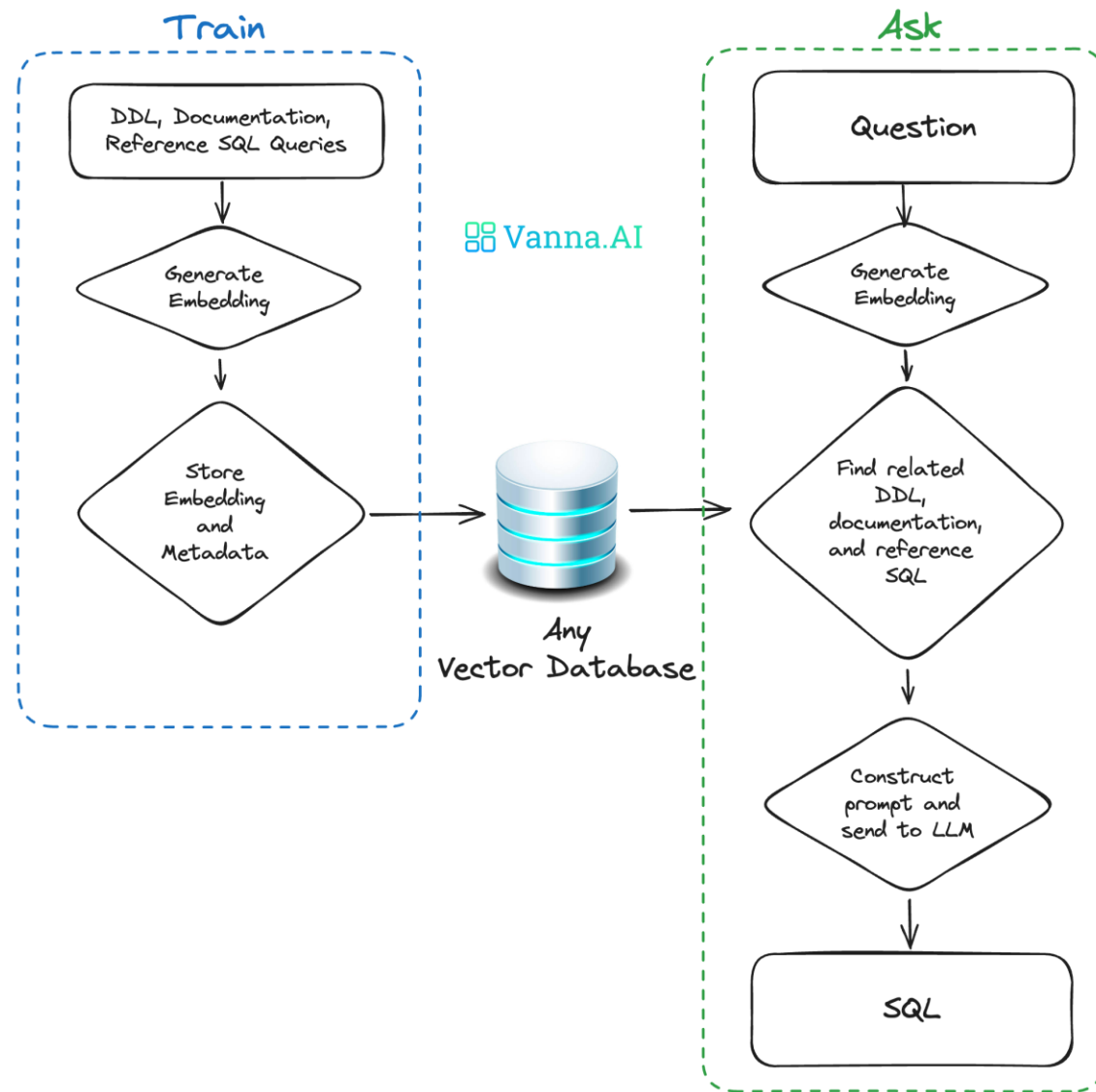
# Vanna 简介

Vanna 的使用分为两个简单步骤 - 在你的数据上“训练”（这里的训练指的是提取特征向量）一个 RAG “模型”，然后提出问题，该问题将返回可设置为自动在你的数据库上运行的 SQL 查询。

- 在你的数据上训练一个 RAG “模型”
- 提问

## 主要功能:

- **自然语言转SQL查询:** 用户可以用平时说话的方式提出问题，Vanna会自动将其转换成SQL语句，并查询数据库。
- **支持各种数据库:** Vanna可以连接到任何使用SQL的数据库，不论数据库的类型。
- **训练模型:** 用户可以根据自己的数据和需求训练Vanna，使得它在转换问题为SQL语句时更准确。
- **灵活的用户界面:** Vanna提供了多种界面选择，如Jupyter Notebook、Web应用、Slack机器人等，以满足不同用户的需求。同时，它支持自定义前端开发。
- **数据安全和隐私保护:** 在处理查询时，数据库内容不会发送到LLM或向量数据库，SQL执行在本地环境中进行。
- **自我学习和优化:** Vanna能根据用户的历史查询和反馈学习和改进，使得未来的查询更加准确。
- **适用于非技术用户:** 对于不懂SQL的用户，Vanna提供了一种简单直观的方式来与数据库交互。它基于开源Python框架，使用检索增强生成（RAG）技术进行SQL生成和相关功能。其主要特点是通过与SQL数据库的交互，准确地通过大型语言模型(LLM)生成SQL查询。





# Vanna 工作原理

Vanna 的使用分为两个步骤：

## 1. 创建知识库：

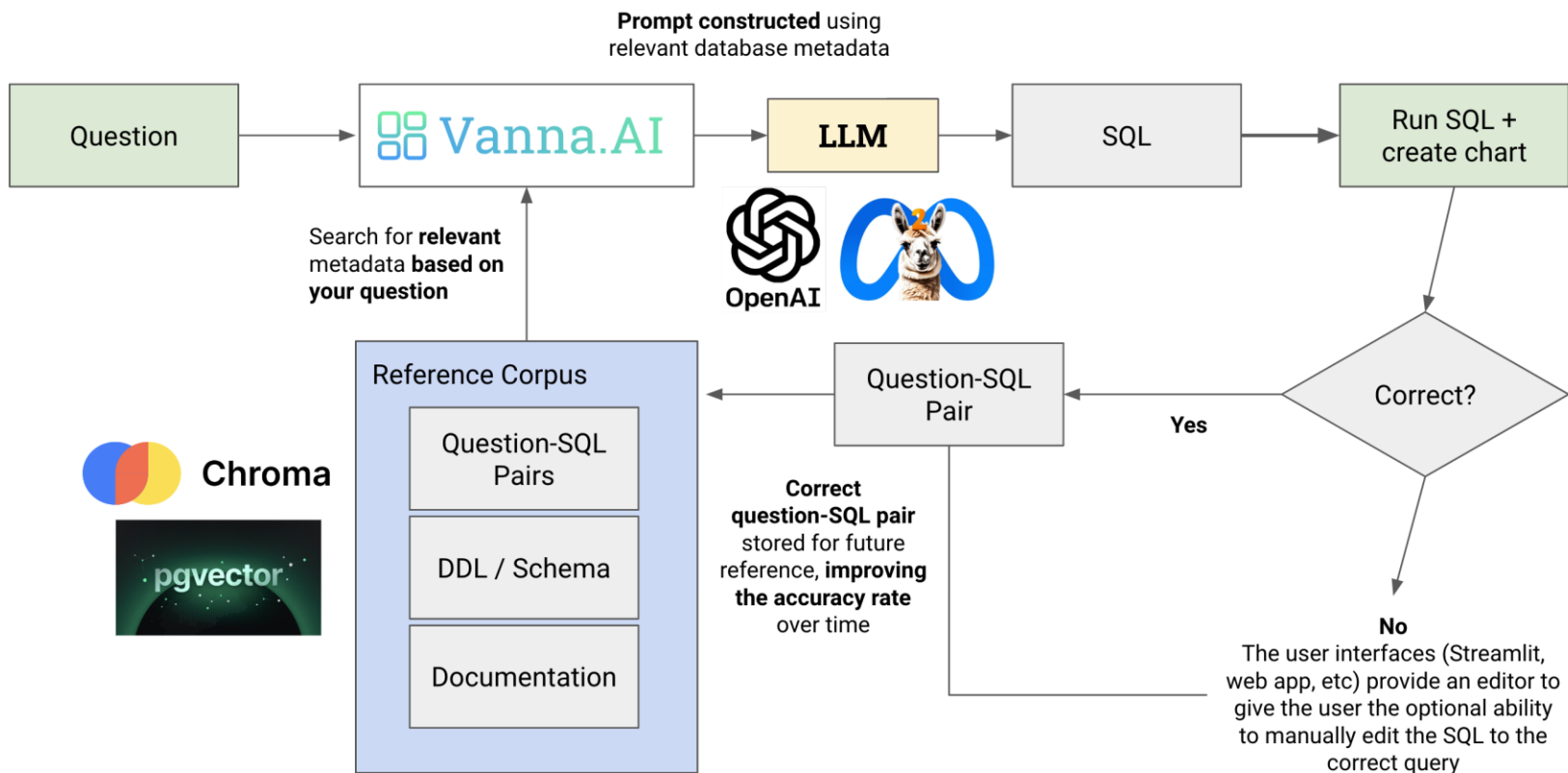
在你的数据上“训练”（这里的训练指的是提取特征向量）一个 RAG “模型”

- 使用DDL语句训练
- 使用文档训练
- 使用SQL训练

## 2. 提出问题：

该问题将返回可设置为自动在你的数据库上运行的 SQL 查询。

- 使用自然语言提出问题
- 搜索向量引擎，在知识库中找到相似度最高的资料
- 将问题与搜索的结果合并成提示词
- 将提示词提交给大语言模型
- LLM返回生成的SQL
- 使用生成的SQL查询数据库
- 获取查询的数据，并生成图表
- 如果结果不符合预期，则返回第2步。或者直接修改生成的SQL。



# 环境搭建

## 用户交互方式:

授权	网址
Jupyter Notebook	<a href="https://github.com/vanna-ai/vanna/blob/main/notebooks/getting-started.ipynb">https://github.com/vanna-ai/vanna/blob/main/notebooks/getting-started.ipynb</a>
vanna-ai/vanna-streamli	<a href="https://github.com/vanna-ai/vanna-streamlit">https://github.com/vanna-ai/vanna-streamlit</a>
vanna-ai/vanna-flask	<a href="https://github.com/vanna-ai/vanna-flask">https://github.com/vanna-ai/vanna-flask</a>
vanna-ai/vanna-slack	<a href="https://github.com/vanna-ai/vanna-slack">https://github.com/vanna-ai/vanna-slack</a>

## 安装配置:

```
# 安装  
pip install vanna
```

```
# 导入  
# 如果您要自定义LLM或向量数据库, 请参阅文档。  
import vanna as vn
```

## 使用样例:

```
1 | !pip install vanna  
2 | import vanna  
3 | from vanna.remote import VannaDefault  
4 | vn = VannaDefault(model='chinook', api_key=vanna.get_api_key('my-email@example.com'))  
5 | vn.connect_to_sqlite('https://vanna.ai/Chinook.sqlite')  
6 | vn.ask('What are the top 10 artists by sales?')  
7 |  
8 | from vanna.flask import VannaFlaskApp  
9 | VannaFlaskApp(vn).run()
```

# 训练（创建知识库）

## 1、使用DDL语句训练

DDL语句包含有关数据库中表名、列、数据类型和关系的信息。

```
vn.train(ddl="""
CREATE TABLE IF NOT EXISTS my-table (
  id INT PRIMARY KEY,
  name VARCHAR(100),
  age INT
)
""")
```

## 2、使用文档训练

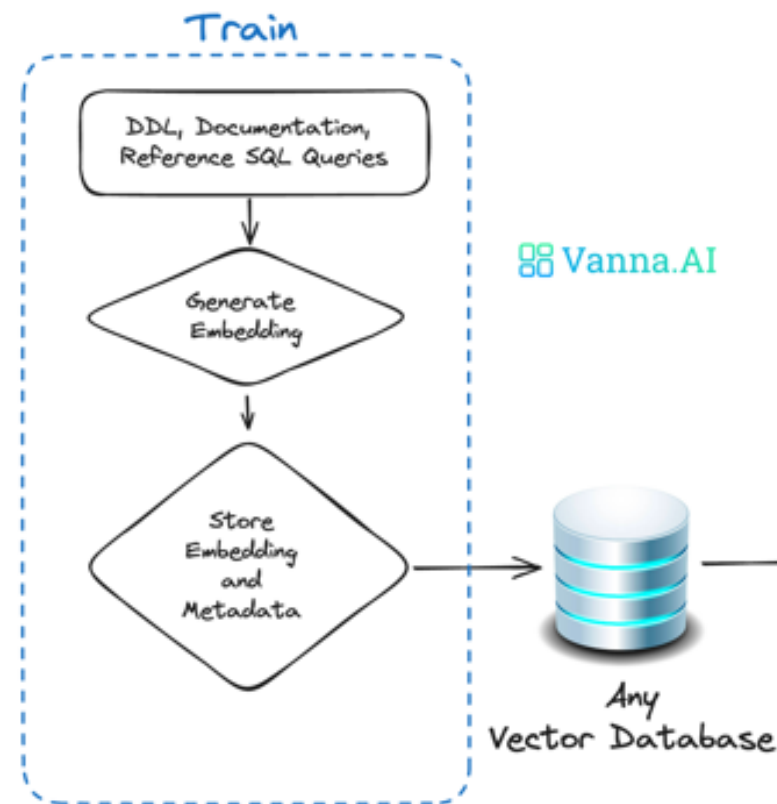
如果需要于业务术语或定义的文档。

```
vn.train(documentation="Our business defines XYZ as ...")
```

## 3、使用SQL训练

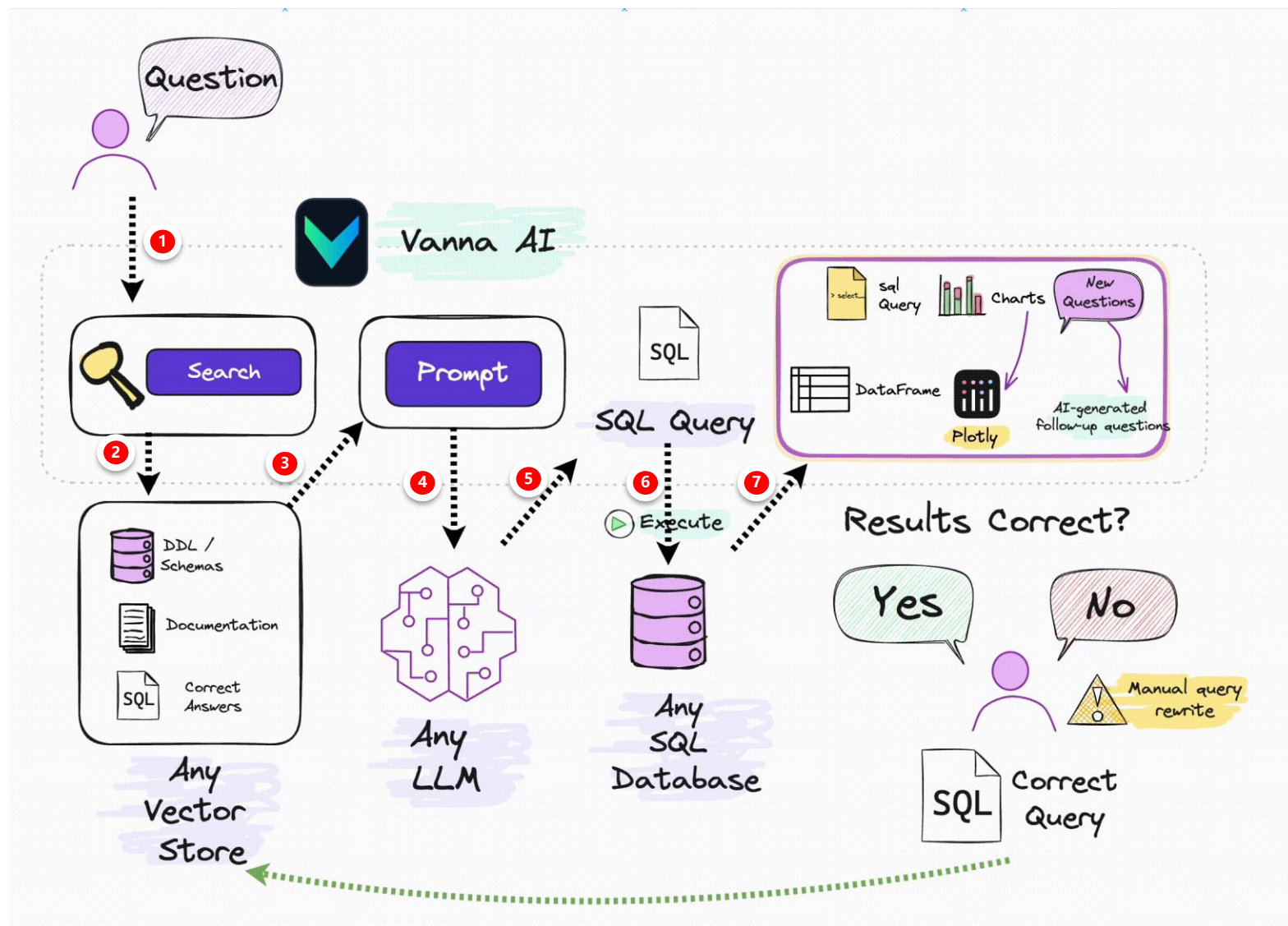
您还可以将SQL查询添加到训练数据中。如果您已经有一些查询可用，只需从编辑器中复制并粘贴它们即可开始生成新的SQL。

```
vn.train(sql="SELECT name, age FROM my-table WHERE name = 'John Doe'")
```



# 提问的流程

1. 使用自然语言提出问题
2. 搜索向量引擎，在知识库中找到相似度最高的资料
3. 将问题与搜索的结果合并成提示词
4. 将提示词提交给大语言模型
5. LLM返回生成的SQL
6. 使用生成的SQL查询数据库
7. 获取查询的数据，并生成图表
8. 如果结果不符合预期，则返回第2步。或者直接修改生成的SQL。



# 提问例子

通过提问可以得到查询结果，以及连接到数据库时的表格和自动生成的Plotly图表。

```
vn.ask("What are the top 10 customers by sales?")
```

得到SQL:

```
SELECT c.c_name as customer_name,
       sum(l.l_extendedprice * (1 - l.l_discount)) as total_sales
FROM   snowflake_sample_data.tpch_sf1.lineitem l join
snowflake_sample_data.tpch_sf1.orders o
       ON l.l_orderkey = o.o_orderkey join
snowflake_sample_data.tpch_sf1.customer c
       ON o.o_custkey = c.c_custkey
GROUP BY customer_name
ORDER BY total_sales desc limit 10;
```

Top 10 Customers by Sales



No.	CUSTOMER_NAME	TOTAL_SALES
0	Customer#000143500	6757566.0218
1	Customer#000095257	6294115.3340
2	Customer#000087115	6184649.5176
3	Customer#000131113	6080943.8305
4	Customer#000134380	6075141.9635
5	Customer#000103834	6059770.3232
6	Customer#000069682	6057779.0348
7	Customer#000102022	6039653.6335
8	Customer#000098587	6027021.5855
9	Customer#000064660	5905659.6159



# RAG vs. Fine-Tuning

**检索增强生成: RAG - Retrieval-Augmented Generation**

**模型微调: Fine-Tuning**

RAG	模型微调
<ul style="list-style-type: none"><li>• 可在LLMs之间移植</li><li>• 如果数据变得过时, 可以轻松删除训练数据</li><li>• 比微调运行成本更低</li><li>• 更具未来性 - 如果有更好的LLM出现, 可以轻松替换</li></ul>	<ul style="list-style-type: none"><li>• 如果需要最小化提示中的标记, 可以选择微调</li><li>• 启动较慢</li><li>• 训练和运行成本较高 (通常如此)</li></ul>

## 扩展Vanna

Vanna旨在连接到任何数据库、LLM和向量数据库。有一个VannaBase抽象基类定义了一些基本功能。该软件包提供了与OpenAI和ChromaDB一起使用的实现。可以轻松扩展Vanna以使用自己的LLM或向量数据库。



# 实践演示

## 1. 安装导入Vanna:

```
!pip install vanna
import vanna
from vanna.remote import VannaDefault
```

## 2. 登录 Vanna:

Vanna提供了一个获取API密钥的功能。你将收到一个发送到你的电子邮件的代码。你可以保存你的API密钥以供将来使用，这样你就不必每次都登录。

```
api_key = vanna.get_api_key('my-email@example.com')
```

## 3. 设置模型:

Chinook是一个公共模型，它指的是Chinook示例数据库。

```
vanna_model_name = 'chinook'
# This is the name of the RAG model. This is typically associated with a specific dataset.
vn = VannaDefault(model=vanna_model_name, api_key=api_key)
```

# 实践演示

## 4. 连接数据库:

这里我们正在连接到一个SQLite数据库, 但你可以连接到任何SQL数据库。

```
vn.connect_to_sqlite('https://vanna.ai/Chinook.sqlite')
```

## 5. 提问:

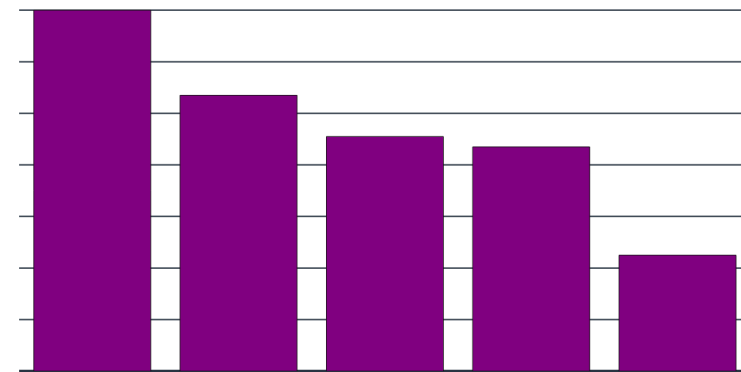
现在我们将使用vn.ask来提问, 它将生成SQL语句, 运行SQL语句, 显示表格, 并生成图表。

```
vn.ask("What are the top 5 artists by sales?")
```

```
SELECT a.name,
       sum(il.quantity) as totalsales
FROM   artist a
       INNER JOIN album al
         ON a.artistid = al.artistid
       INNER JOIN track t
         ON al.albumid = t.albumid
       INNER JOIN invoiceline il
         ON t.trackid = il.trackid
GROUP BY a.name
ORDER BY totalsales desc limit 5;
```



	Name	totalsales
0	Iron Maiden	140
1	U2	107
2	Metallica	91
3	Led Zeppelin	87
4	Os Paralamas Do Sucesso	45

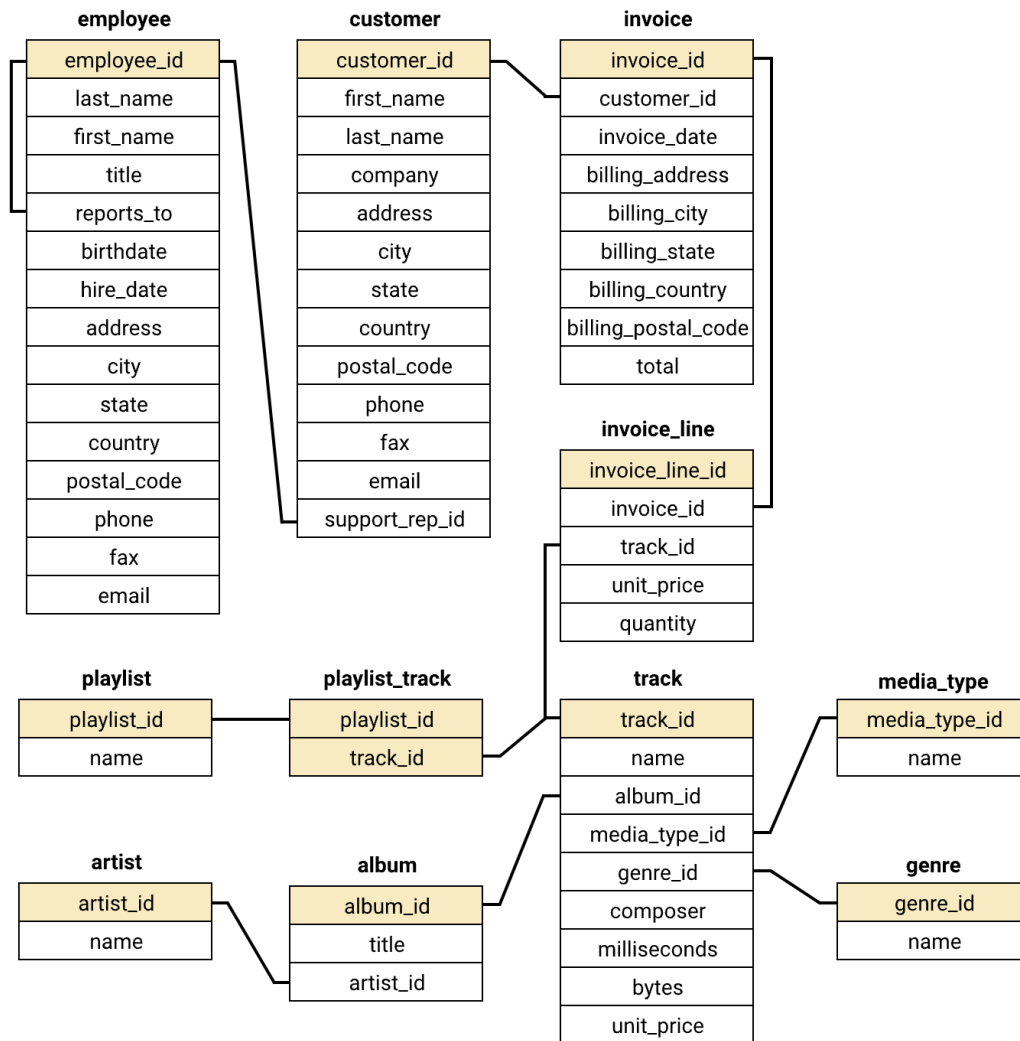


# 实践演示

## 提问参考样例：

- What is the total sales for each artist?
- What are the top-selling albums for each artist?
- Which genre has the highest sales?
- What is the total sales for each genre?
- What are the sales trends over the years?
- How many albums are there in the database?
- Who are the top-selling artists in each genre?
- What are the top-selling tracks?
- What is the average sales per artist?
- How does the sales distribution vary across different genres?

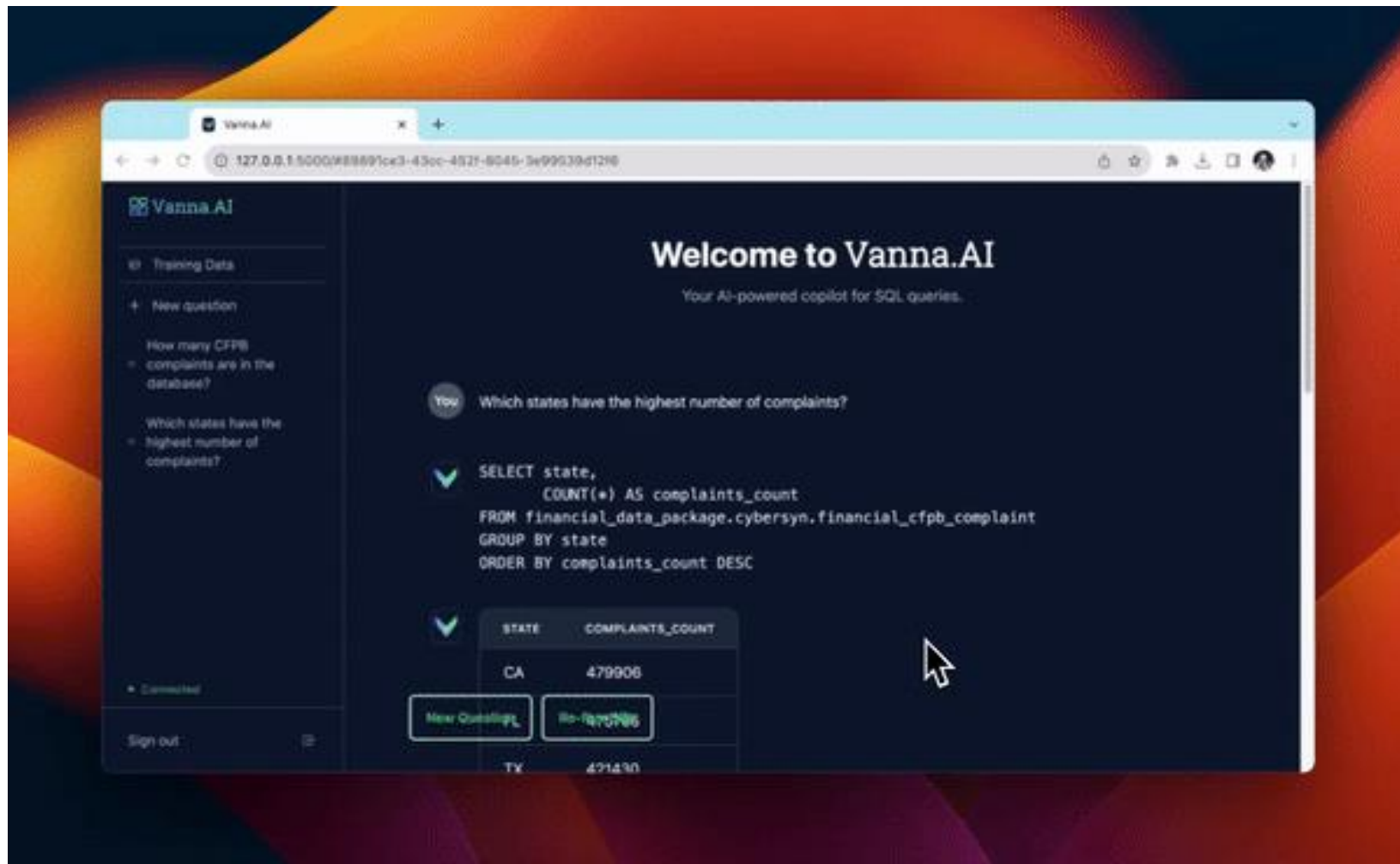
## 参考：这些是数据库中的表



# 实践演示

## 启动用户界面:

```
from vanna.flask import VannaFlaskApp
app = VannaFlaskApp(vn)
app.run()
```



# 其它用户交互界面 - Vanna.AI Streamlit App

## 安装配置:

这个项目使用Poetry, 所以首先需要安装它。

```
pip install poetry  
poetry install --with dev
```

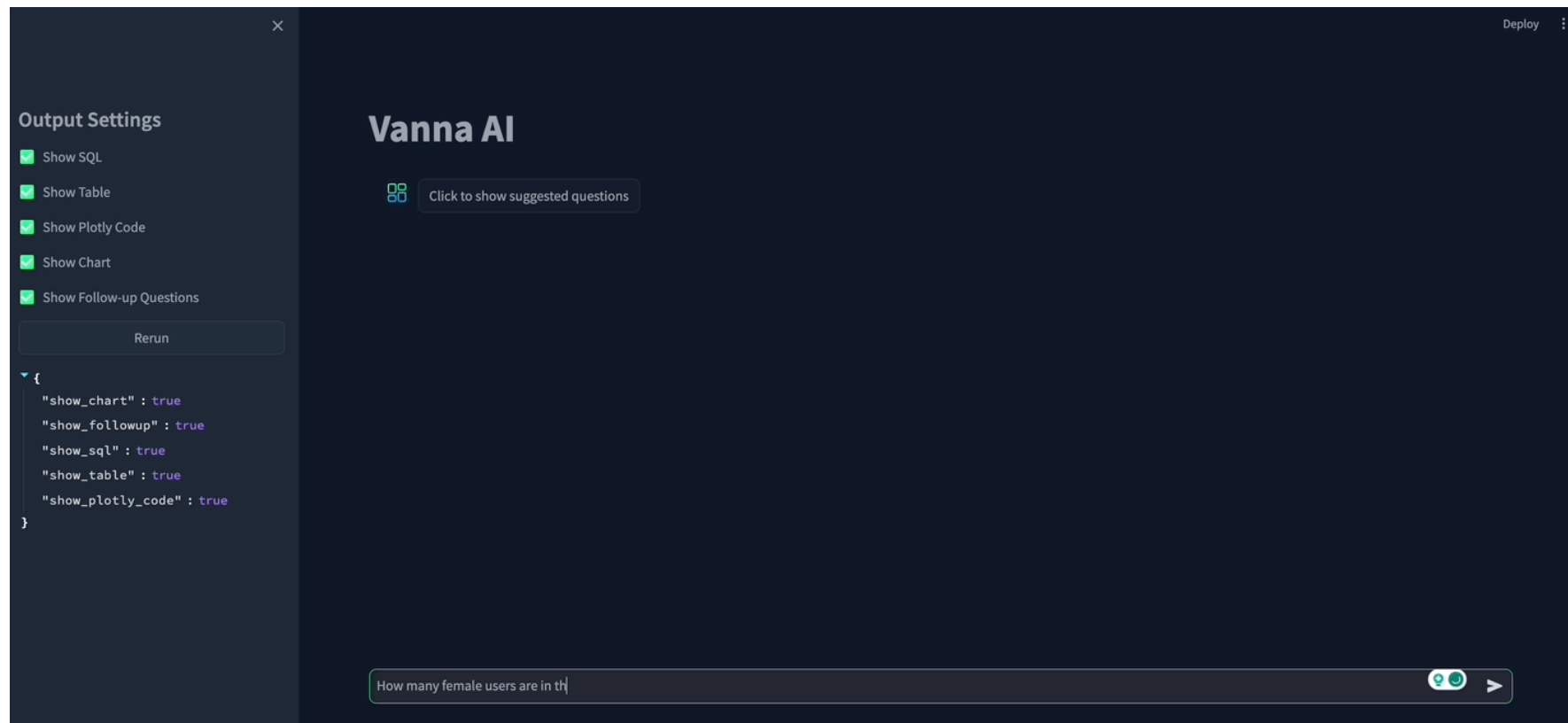
## Poetry可以用于:

- 安装和管理项目的依赖项
- 创建一个干净的虚拟环境, 该环境与您当前的Python环境完全隔离开来

## 运行:

这个项目使用Poetry, 所以首先需要安装它。

```
poetry run streamlit run app.py
```







Thank

You