

## 用DeepSeek AI帮你SQL开发与调优

1 最近在SQL等编程中使用AI的三种方式：AI对话，AI Copilot，AI Agent。这三种模式对应不同的场景，在工作和个人开发的实践中都用到了，有其代表性。

2 第一种方式是直接和AI对话。这种方式在我工作中用得比较多。作为数据工程师，我主要使用SQL语言来处理数据。一方面因为主要的工作环境是浏览器中的云端IDE，代码不能导入Cursor这样的AI编程工具，而云端IDE自带的AI编程功能较弱，因此有和更强AI对话的需要；另一方面，一个大的数据开发工程，其单个SQL文件(基本上对应着一个表的加工逻辑)之间的逻辑是解耦得非常松散，只通过数据表来传递依赖关系。因此只需要将SQL文件依赖的表结构告诉AI，并讲清楚要实现的目标，以及实现的逻辑(如果有的话)，AI很容易生成高质量的SQL代码。

一个典型的Prompt如下：

```
1 CREATE TABLE tbl_1(  
2   fa STRING COMMENT '字段a',  
3   fb STRING COMMENT '字段b'  
4 )COMMENT '表1'  
5 PARTITION BY(  
6   ds STRING '日期分区, yyyyymmdd'  
7 )  
8 ;  
9  
10 - -  
  
12 基于上面的tbl_1, 用HIVE SQL实现下面的逻辑:  
13 1. 去重: xxx逻辑  
14 2. 过滤: xxx逻辑  
15 3. ...  
16  
17 输出表结构如下:  
18 CREATE TABLE tbl_2(  
19   f1 STRING COMMENT '字段1',  
20   f2 STRING COMMENT '字段2'
```

```
21 ) COMMENT '表2'
22 PARTITION BY (
23     ds STRING '日期分区, YYYYMMdd'
24 )
25 ;
26
27 代码风格参考下面的SQL :
28 . . .
29
```

3 自从2023年chatGPT 流行后，我就一直使用这种方式写代码以及重构SQL，基本上实现的都是逻辑非常复杂的SQL。一个体感是即便是GPT-3.5这样古早时期的LLM，在SQL代码生成上，都是能胜任大部分任务的。有了O1和R1这样的推理模型后，其代码质量更是精良，绝大多数时候都能实现一遍过。

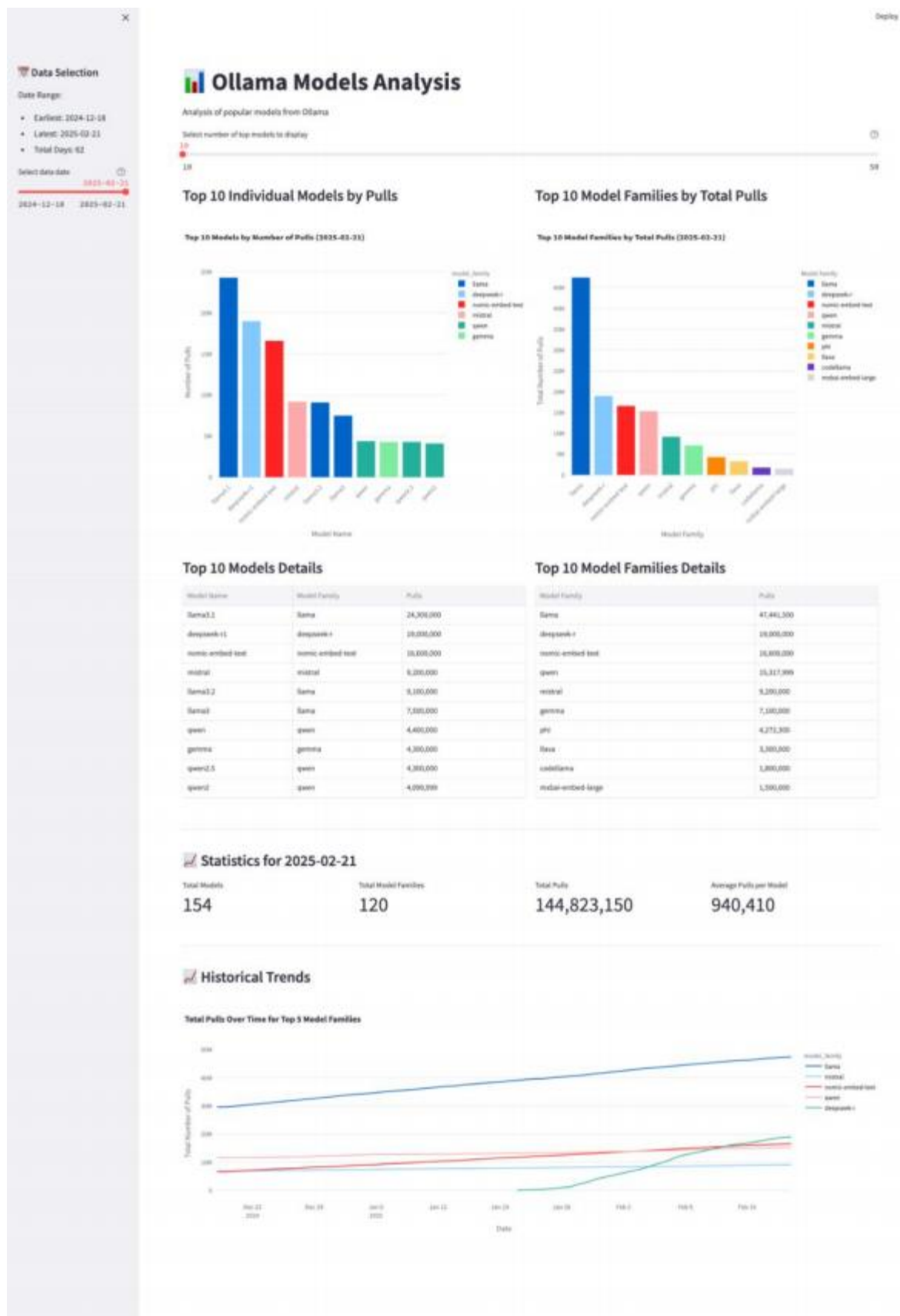
4 另一个实践心得是：为增加代码可靠性，建议使用两个LLM同时生成代码，并互相review。一方面可以让不同LLM的给出多个输出，任君挑选，通过比较更容易得到代码质量高的实现；另一方面，互相review可以多一层代码质量的保证。下图是我让chatGPT O3-mini对Deepseek R1生成的代码做review的截图：



## LLM之间互相Review代码

5 第二种方式是使用AI Copilot，一开始是Github copilot，现在是cursor。在工作中，主要用来写数据平台的python代码；在业余项目中，最近一个是抓取ollama上开源LLM的拉

取数量、做数据可视化，从而观察业界趋势(从最后的趋势图上可以清晰看到Deepseek R1 的崛起)。

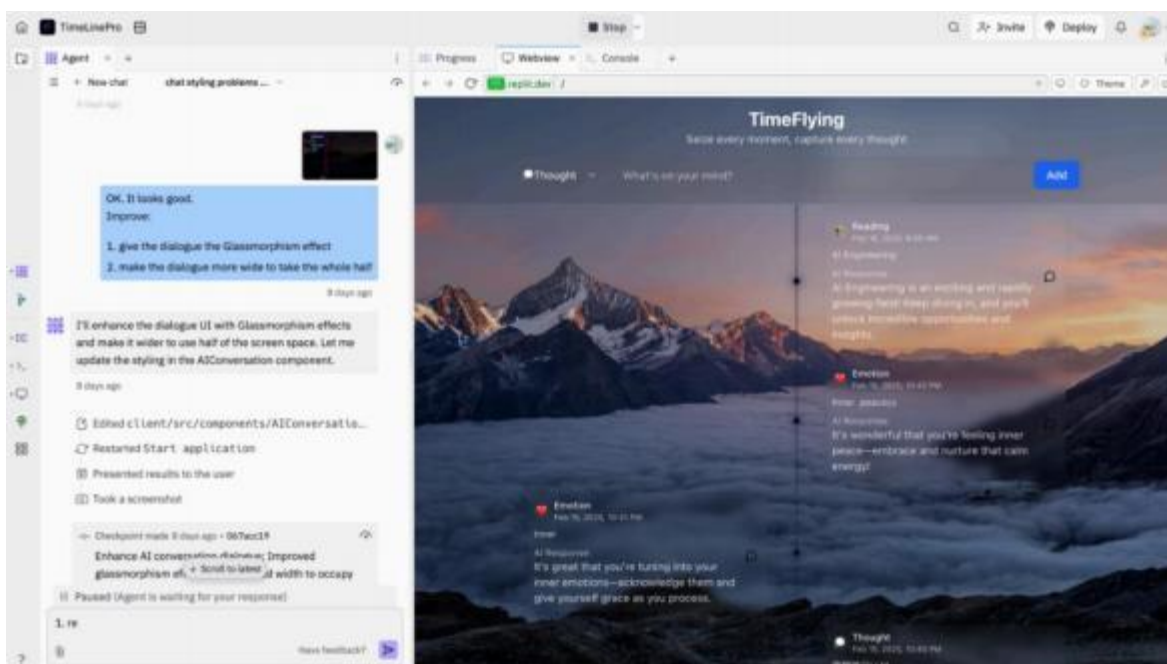


ollama Models Analysis

这类项目有一个特点，就是我自己对技术细节非常清楚，AI仅仅作为助手，帮我加快开发

的效率。以上面的ollama Models Analysis任务为例，我知道它的架构:抓取网页、解析拉取数量、存储数据、使用streamlit做数据可视化、使用bash+crontab做每日调度等等。我也明确它接下来可能的技术迭代方向:部署到公网、数据云端存储、数据分层计算等等。我主导整个技术栈；出现错误我能很快找到技术解决方案。在这种任务中，人主导的是技术实现。

6 第三种方式是使用AI Agent。我试用过的产品是Bolt.new、V0和Repl.it，主要用在我原本不擅长的技术领域，比如前端或者全栈开发。于是一起开发工作都交给AI，我只管提需求，评估结构，给出反馈。我是个老板；AI是我的程序员。在我试用的三个产品中，前面两个都因为它们总会在某个错误上卡住出不来(同一个错误反复出现，不能修复)，而因为我自己不熟悉相应的技术，也无法提供解决方案，从而无奈抛弃。就像老板招了一个不靠谱的程序员，项目搞到一半就黄了一样，只能换一个。下图就是我使用Repl.it实现的【时光悠悠】产品初版：



Repl.it的编程Agent

可以看到，我只是不断地提需求、做评估、给反馈，AI帮我搞定剩下的事情:写代码、改代码、运行代码、修bug等等。唯一的缺点是：如果AI表现不好，卡住了，我也被卡住了。这是典型的Type 2项目，用户不能主导实现，在AI能力不足以全自动化时，会遇到麻烦。

7 在给AI提需求的过程中，我参考的经验是提供一个PRD (产品设计文档) markdown文件，包含产品描述(实现标准、验证清单)、设计元素、技术栈(个人觉得不必要，反正我也不懂；不过选一个主流的技术栈可以让AI更容易debug)、用户故事(用户在什么情况下要使用什么功能实现什么目标)、其他事项。一个例子如下：

```

1  ---
2  name: "Mood Tracker"
3  about: "Modern mood tracking web app with data visualization"
4  date_created: "2025-01-26"
5  project_name: "MoodTracker"
6  tech_stack: [ "NextJS 15", "Typescript", "shadcn", "Tailwind CSS", "chart.js":
7  version: "1.3"
8  ---
9
10 # 🎨 Mood Tracker PRD
11
12 A modern web application for logging daily moods and visualizing emotional tr
13
14 ---
15
16 ## 1. **success criteria**
17
18 1. **core Functionality**
19   - [ ] **clickable calendar**: users can select a date to log or edit a mood
20   - [ ] **Emoji & Note Input**: A modal or dialog with an emoji picker and 1
21   - [ ] **Local Data storage**: persist mood entries between sessions.
22   - [ ] **Data visualization**: At least two chart.js Charts to display wee
23   - **Mobile-Responsive**: Layout should adjust for smaller Screens wit
24
25 2. **validation checklist**
26   - [ ] **Build & Run**: Fresh `npm install` & `npm run dev` works without er
27   - [ ] **calendar Interaction**: clicking a calendar date opens the mood log
28   - [ ] **color coding**: Each date cell or icon changes based on mood score
29   - **chart page**: A Separate page or section to visualize stats (e.g. 8 days)
30   - **Data persistence**: Entries remain available if the user navigates
31
32 ---
33
34 ## 2. **Tech stack**
35
36 - **NextJS 15** (APP Router) for site structure -
37 - **Typescript** for type safety
38 - **shadcn** UI Components (dialogs, buttons, forms) -
39 - **Tailwind CSS** for styling
40 - **chart.js** for data visualization
41 - **date-fns** for date operations
42 - **localStorage** (or equivalent) for local data storage -
43 - **@emoji-mart/react** for an emoji picker
44
45 ### **why These choices?**
46 - *NextJS + Typescript*: Great for server/client flexibility and type safety

```

```

47 - **shadcn + Tailwind**: Rapid UI development with consistent design
48 - **chart.js**: straightforward library for rendering charts
49 - **date-fns**: Lightweight date utilities
50
51 ---
52
53 ## 3. **Design & Mood scores**
54
55 | Moodscore | Mood          | Tailwind color | Emoji          |
56 |-----|-----|-----|-----|
57 | 1          | Angry         | `red-500`      | 🙄             |
58 | 2          | Sad           | `orange-400`   | 😞             |
59 | 3          | Neutral       | `yellow-300`   | 😐             |
60 | 4          | Happy         | `lime-400`     | 😊             |
61 | 5          | Ecstatic      | `emerald-500`  | 😄             |
62
63 > You can style each date cell background or display an icon to indicate the
64
65 ---
66
67 ## 4. **user stories**
68
69 1. **Daily Mood Logging**
70   - **As a user**, I Want to quickly log how I feel each day So I Can track
71     - **Given** I Click on a specific date
72     - **when** I Choose an emoji and type a note
73     - **Then** the date on the calendar updates visually to reflect my mood
74
75 2. **Mood Analysis**
76   - **As a user**, I Want to see a higher-level overview of my moods So I Can
77     - **Given** I navigate to a "stats" page
78     - **when** I Select a timeframe (weekly, monthly, etc.)
79     - **Then** I See at least two types of charts illustrating changes or d
80

```

8 在给AI反馈时，有两点心得：一是用“任务类型”开头，比如feature(新功能)、debug（出错了）、ui(界面调整)等，这样AI会更清楚本轮对话的目的；二是在必要时，附上一些界面截图，特别是要做UI调整，或者debug时，让AI有更丰富的输入，会更有效。

希望对你使用AI构建自己心中的代码项目产品等有帮助。