



# EVEN CHILDREN CAN WORK WITH POSTGRES

TATIANA KRUPENYA

October 5, 2023

# NICE TO MEET YOU

- I am the CEO of DBeaver Corporation
- DBeaver is a UI tool for database management and administration
- DBeaver was born as an open-source project in 2011
- The number of DBeaver users has grown to more than 8 million
- DBeaver is used in every country in the world





# ChatGPT FOR QUERIES

chat.openai.com

“write a SQL statement to show all invoices from Postgres”



write a SQL statement to show all invoices from Postgres



Certainly! To retrieve all invoices from a PostgreSQL database, you can use the following SQL statement:   

sql

 Copy code

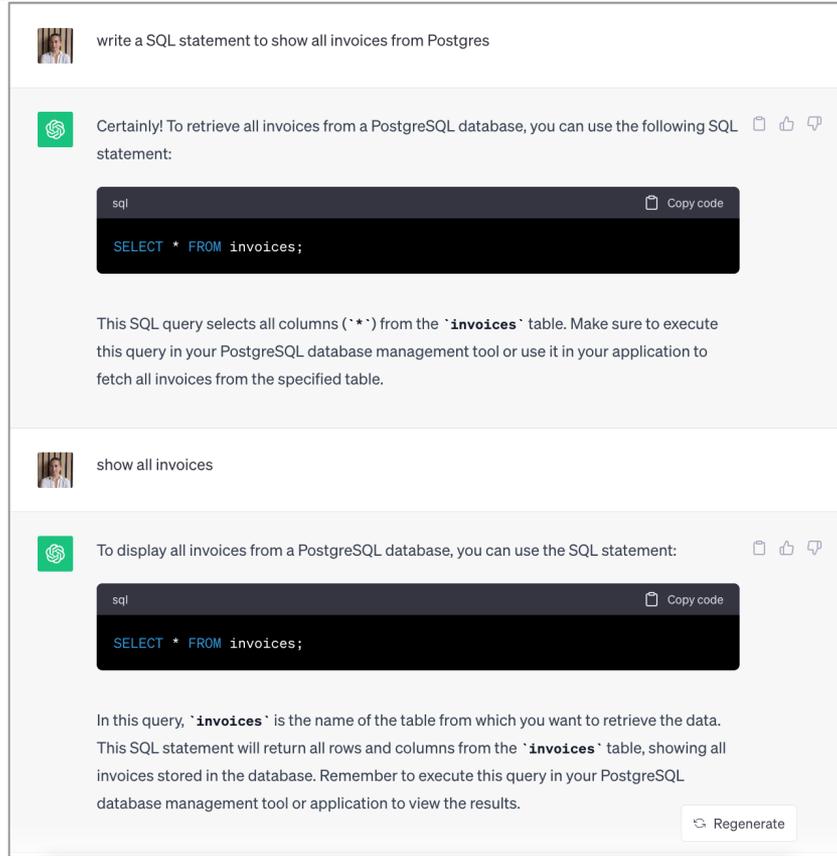
```
SELECT * FROM invoices;
```

This SQL query selects all columns (``\*) from the `invoices` table. Make sure to execute this query in your PostgreSQL database management tool or use it in your application to fetch all invoices from the specified table.

# ChatGPT FOR QUERIES

chat.openai.com

“show all invoices”



write a SQL statement to show all invoices from Postgres

Certainly! To retrieve all invoices from a PostgreSQL database, you can use the following SQL statement:

```
sql Copy code  
SELECT * FROM invoices;
```

This SQL query selects all columns (`*`) from the `invoices` table. Make sure to execute this query in your PostgreSQL database management tool or use it in your application to fetch all invoices from the specified table.

show all invoices

To display all invoices from a PostgreSQL database, you can use the SQL statement:

```
sql Copy code  
SELECT * FROM invoices;
```

In this query, `invoices` is the name of the table from which you want to retrieve the data. This SQL statement will return all rows and columns from the `invoices` table, showing all invoices stored in the database. Remember to execute this query in your PostgreSQL database management tool or application to view the results.

Regenerate

# ChatGPT FOR QUERIES

chat.openai.com

“show all invoices”



show all invoices



I'm sorry for any confusion, but I don't have access to specific databases or files, so I can't show you specific invoices. However, I can certainly help guide you on how to create an invoice or provide information about what to include in an invoice if that's what you're looking for. Let me know how I can assist you further!



# CONTEXT IS THE KEY

chat.openai.com

“Show all invoices”

SQL

Copy code

```
SELECT * FROM invoices;
```

1

“write a **SQL statement** to show all invoices from **Postgres**”

2

## ChatGPT



### Examples

"Explain quantum computing in simple terms" →

"Got any creative ideas for a 10 year old's birthday?" →

"How do I make an HTTP request in Javascript?" →



### Capabilities

Remembers what user said earlier in the conversation

Allows user to provide follow-up corrections

Trained to decline inappropriate requests



### Limitations

May occasionally generate incorrect information

May occasionally produce harmful instructions or biased content

Limited knowledge of world and events after 2021

# WHAT IS YOUR QUERY CONTEXT?

The screenshot shows the DBeaver Database Navigator interface. The left sidebar displays a tree view of the database structure for 'pagila', with 'public' > 'film' selected. The main window shows the 'Properties' tab for the 'film' table. The table name is 'film', Object ID is 525409, and it is located in the 'pg\_default' tablespace. The 'Columns' tab is active, displaying a table with 13 columns.

|              | Column Name              | #  | Data type     | Identity | Collation | Not Null | Default                          |
|--------------|--------------------------|----|---------------|----------|-----------|----------|----------------------------------|
| Columns      | 123 film_id              | 1  | serial4       |          |           | [v]      | nextval('film_film_id_seq'::regc |
| Constraints  | ABC title                | 2  | varchar(255)  | default  |           | [v]      |                                  |
| Foreign Keys | ABC descript             | 3  | text          | default  |           | [ ]      |                                  |
| Indexes      | 123 release_year         | 4  | year          |          |           | [ ]      |                                  |
| Dependencies | 123 language_id          | 5  | int2          |          |           | [v]      |                                  |
| References   | 123 original_language_id | 6  | int2          |          |           | [ ]      |                                  |
| Partitions   | 123 rental_duration      | 7  | int2          |          |           | [v]      | 3                                |
| Triggers     | 123 rental_rate          | 8  | numeric(4, 2) |          |           | [v]      | 4.99                             |
| Rules        | 123 length               | 9  | int2          |          |           | [ ]      |                                  |
| Policies     | 123 replacement_cost     | 10 | numeric(5, 2) |          |           | [v]      | 19.99                            |
| Statistics   | ABC rating               | 11 | mpaa_rating   |          |           | [ ]      | 'G'::mpaa_rating                 |
| Permissions  | last_update              | 12 | timestamp     |          |           | [v]      | now()                            |
| DDL          | special_features         | 13 | _text         | default  |           | [ ]      |                                  |
| Virtual      |                          |    |               |          |           |          |                                  |

- Database type
- Database name
- Schema name
- Table name
- Column name
- Primary keys
- Foreign keys

# IT'S TOO MUCH

The screenshot shows the DBeaver Database Navigator interface. On the left, a tree view shows the database structure: pagila > Schemas > public > Tables > film. The main pane shows the 'Properties' tab for the 'film' table. The 'Columns' tab is active, displaying a table with the following data:

| Column Name            | Base Type        | Ident | Collation | Not Null | Default                          |
|------------------------|------------------|-------|-----------|----------|----------------------------------|
| 1 film_id              | 1 serial         |       |           | [v]      | nextval('film_film_id_seq'::regc |
| 2 title                | 2 varchar(255)   |       |           | [v]      |                                  |
| 3 description          | 3 text           |       | default   | [ ]      |                                  |
| 4 release_year         | 4 integer        |       |           | [ ]      |                                  |
| 5 language_id          | 5 integer        |       |           | [v]      |                                  |
| 6 original_language_id | 6 integer        |       |           | [ ]      |                                  |
| 7 rental_rate          | 7 integer        |       |           | [v]      | 3                                |
| 8 rental_rate          | 8 numeric(4, 2)  |       |           | [v]      | 4.99                             |
| 9 length               | 9 integer        |       |           | [ ]      |                                  |
| 10 replacement_cost    | 10 numeric(5, 2) |       |           | [v]      | 19.99                            |
| 11 rating              | 11 mpaa_rating   |       |           | [ ]      | 'G'::mpaa_rating                 |
| 12 last_update         | 12 timestamp     |       |           | [v]      | now()                            |
| 13 special_features    | 13 text          |       | default   | [ ]      |                                  |

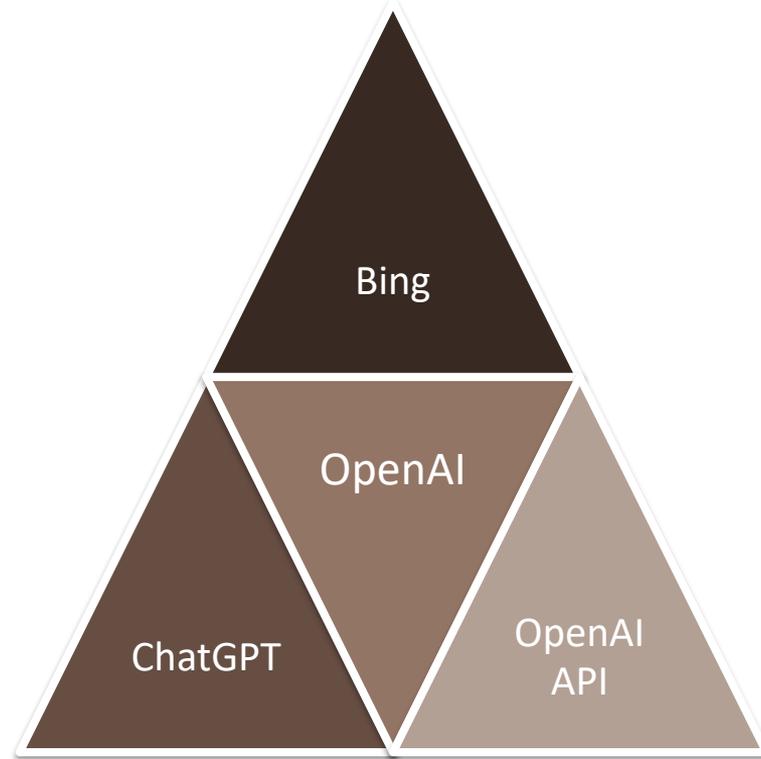
- Database type
- Database name
- Schema name
- Column name
- Primary keys
- Foreign keys

# OPEN AI INTERFACES

[platform.openai.com](https://platform.openai.com)

**OpenAI API** is a good option if you want to automate your work.

Remember, it's not free.



# HOW IT WORKS

Determine context

```
graph TD; A[Determine context] --> B[Write request]; B --> C[Form prompt]; C --> D[Get response]; D --> E[Show results];
```

Write request

Form prompt

Get response

Show results

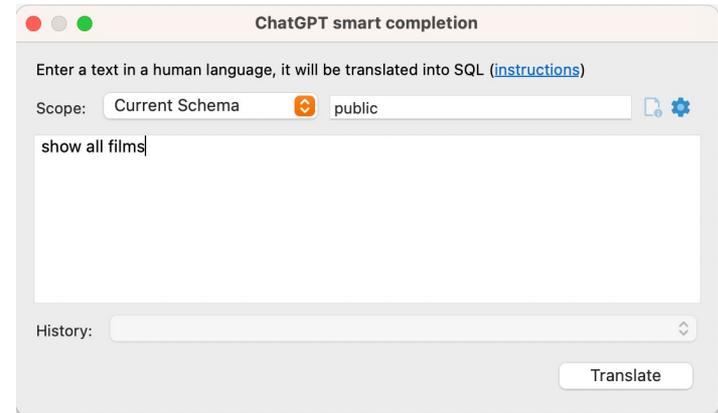
# DETERMINE CONTEXT



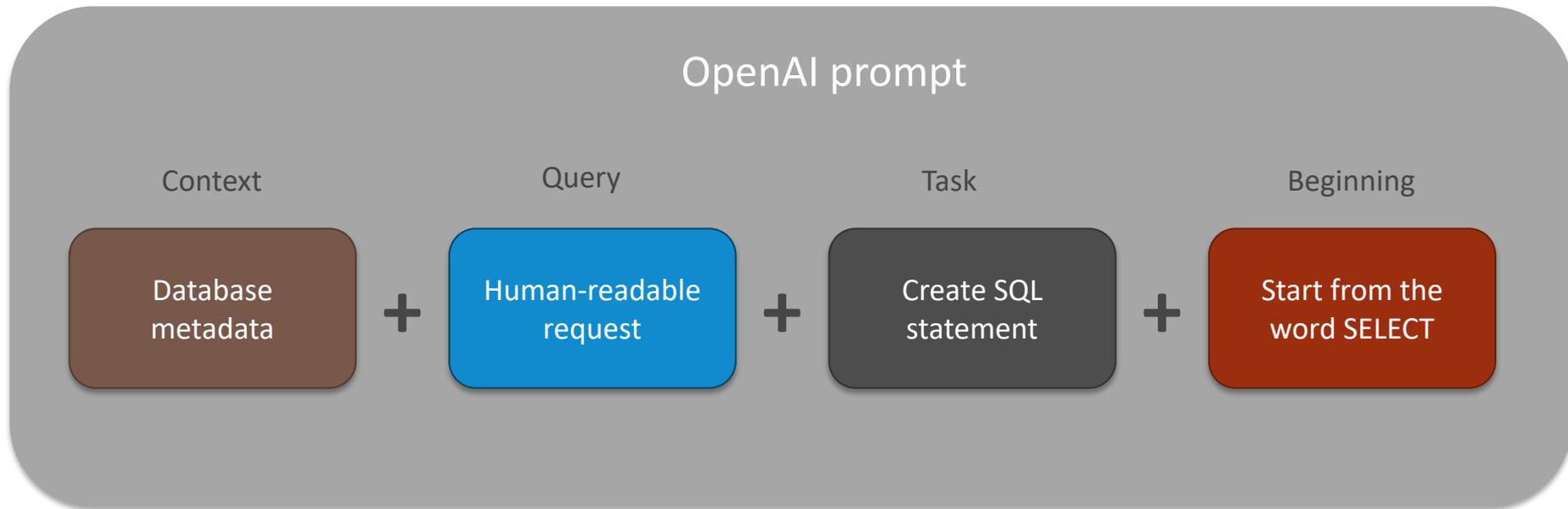
- More information is better than less
- The number of tokens is the main limitation of your context:
  - 1000 words are around 750 tokens
  - GPT 3.0 – 8,000 tokens
  - GPT 3.5 – 16,000 tokens
  - GPT 4.0 – 32,000 tokens
- Don't trust OpenAI API memory

# WRITE REQUEST – IT'S SIMPLE

- ✓ Any human language
- ✓ You can formulate your request in any words
- ✓ Not necessary to be grammatically correct
- ✓ You can request data from different tables
- ✓ You can provide hints for OpenAI, like “use joins”

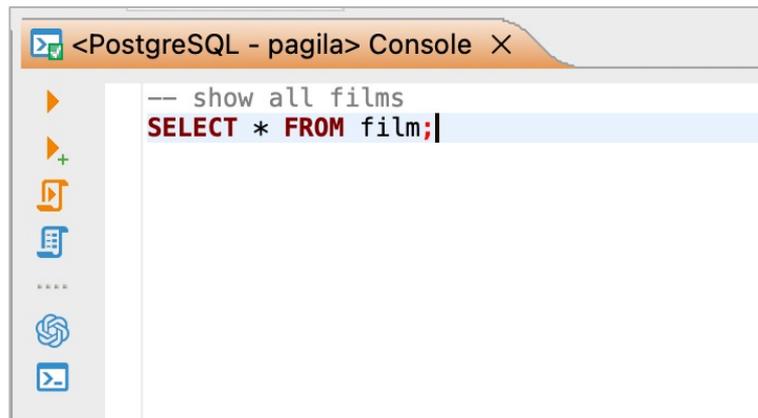


# FORM PROMPT



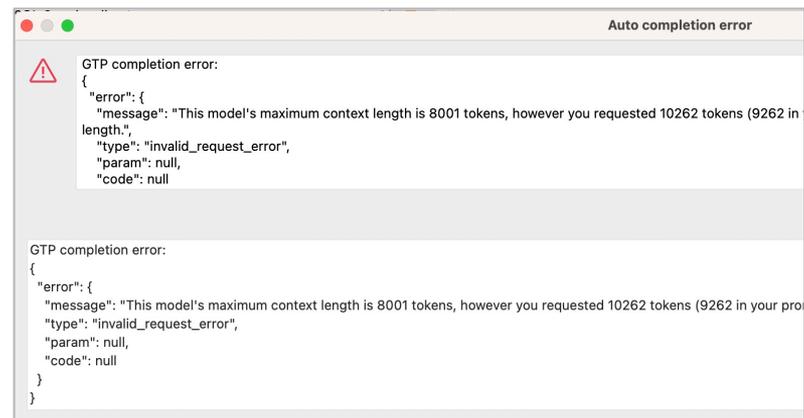
# GET RESPONSE

POSITIVE



A screenshot of a PostgreSQL console window titled "<PostgreSQL - pagila> Console". The window contains the following text: "-- show all films" followed by "SELECT \* FROM film;". The text is highlighted in blue. On the left side of the window, there is a vertical toolbar with several icons: a yellow arrow, a plus sign, a document icon, a speech bubble icon, a gear icon, and a terminal icon.

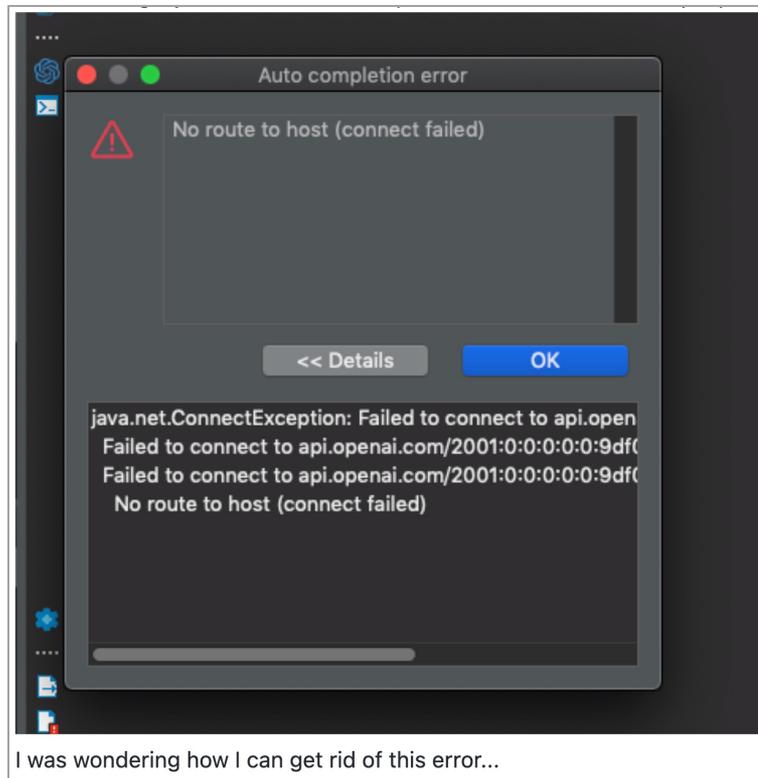
NEGATIVE



A screenshot of an "Auto completion error" dialog box. The dialog box has a title bar with three colored buttons (red, yellow, green) and the text "Auto completion error". Below the title bar, there is a warning icon (a triangle with an exclamation mark) and the text "GTP completion error:". Below this, there is a JSON object representing the error: {"error": {"message": "This model's maximum context length is 8001 tokens, however you requested 10262 tokens (9262 in length.", "type": "invalid\_request\_error", "param": null, "code": null}}. Below this, there is another "GTP completion error:" label followed by another identical JSON object: {"error": {"message": "This model's maximum context length is 8001 tokens, however you requested 10262 tokens (9262 in your pro", "type": "invalid\_request\_error", "param": null, "code": null}}.

# WHY DID SOMETHING GO WRONG?

- X You do not have internet access
- X OpenAI is under a high load
- X Your API token is invalid
- X You sent too many tokens
- X You chose the wrong model
- X You forgot to pay your OpenAI bills



# NO GUARANTEES

The screenshot shows the DBeaver Ultimate 23.2.0 interface. The main window is titled "<PostgreSQL - pagila> Console". The left sidebar shows the database structure for "pagila", including schemas like "information\_schem", "pg\_catalog", and "public", and tables like "actor", "address", "category", "city", "country", "film", "film\_actor", "film\_category", "inventory", "language", "payment", "rental", "staff", "store", "actor\_info", and "film\_list".

The console window contains the following SQL query:

```
-- show all films rented on May 25 2005
select
  film.title,
  rental.rental_date
from
  film
join inventory on
  film.film_id = inventory.film_id
join rental on
  inventory.inventory_id = rental.inventory_id
where
  rental.rental_date = '2005-05-25';
```

The results pane shows a table with two columns: "title" and "rental\_date". The table is currently empty, with the message "No data- 145ms, on 2023-10-01 at 13:17:27" displayed at the bottom. The status bar at the bottom of the window shows "pagila", "CST", and "en\_US".

Asking the Right Question Is Half the Answer

# YOUR SUCCESS

The screenshot shows the DBeaver Ultimate 23.2.0 interface. The main window is titled "<PostgreSQL - pagila> Console". The left sidebar shows the database structure for "pagila", including schemas, tables, and views. The central console area contains a SQL query:

```
-- show all films rented between midnight May 25 2005 and midnight May 26 2005
select
  film.title,
  rental.rental_date
from
  film
join inventory on
  film.film_id = inventory.film_id
join rental on
  inventory.inventory_id = rental.inventory_id
where
  rental.rental_date >= '2005-05-25 00:00:00'
  and rental.rental_date < '2005-05-26 00:00:00';
```

Below the query, the execution plan is shown as "film(+)" 1. The results are displayed in a table with columns "title" and "rental\_date".

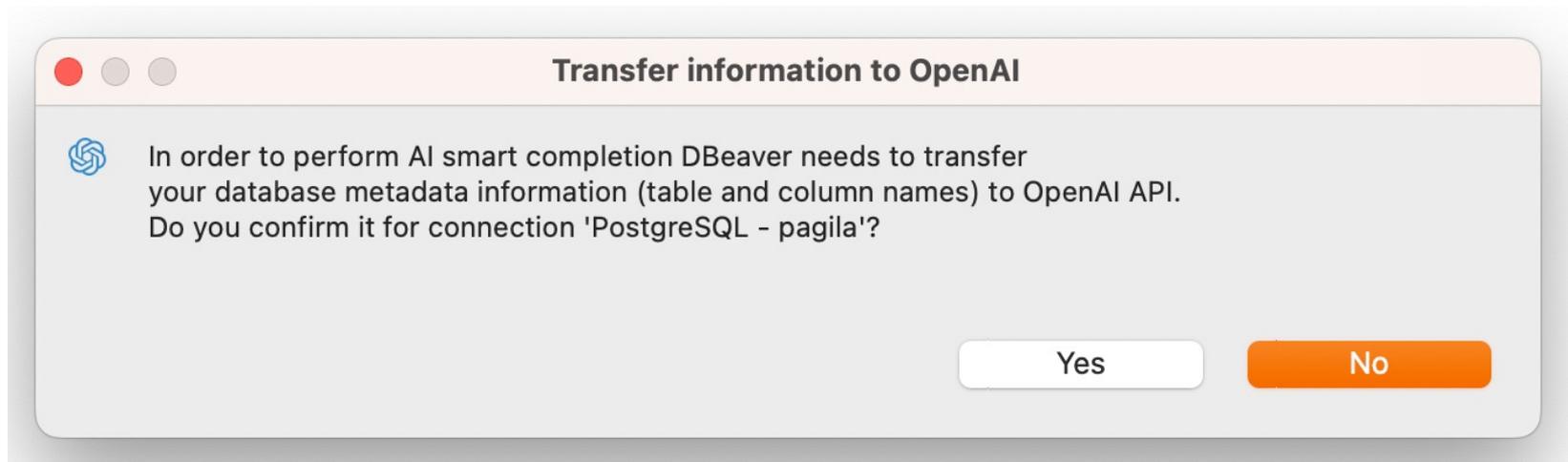
|   | title               | rental_date             |
|---|---------------------|-------------------------|
| 1 | AIRPORT POLLOCK     | 2005-05-25 19:37:47.000 |
| 2 | ALLEY EVOLUTION     | 2005-05-25 09:21:29.000 |
| 3 | AMERICAN CIRCUS     | 2005-05-25 21:58:58.000 |
| 4 | ANTHEM LUKE         | 2005-05-25 14:57:22.000 |
| 5 | APACHE DIVINE       | 2005-05-25 01:59:46.000 |
| 6 | ARMAGEDDON LOST     | 2005-05-25 16:12:52.000 |
| 7 | BACKLASH UNDEFEATED | 2005-05-25 20:46:11.000 |
| 8 | BANGER PINOCCHIO    | 2005-05-25 09:41:01.000 |

The bottom status bar indicates "137 row(s) fetched - 334ms, on 2023-10-01 at 13:21:28". The interface also shows various toolbars and panels like "SQL Editor", "Query Builder", and "Panels".

# UNOBVIOUS LIMITATIONS

- Huge schemas with hundreds and thousands of tables
- Analytical databases
- Long names for tables and columns
- Brand new databases with a lack of documentation
- NoSQL databases

# IS IT SAFE?



## Enterprise privacy at OpenAI

Trust and privacy are at the core of our mission at OpenAI. We're committed to privacy and security for ChatGPT Enterprise and our API Platform.

[Read commitments ↓](#)

---

### Our commitments

#### Ownership: You own and control your data

- ✓ We do *not* train on your business data, and our models don't learn from your usage
- ✓ You own your inputs and outputs (where allowed by law)
- ✓ You control how long your data is retained (ChatGPT Enterprise)

#### Control: You decide who has access

- ✓ Enterprise-level authentication through SAML SSO
- ✓ Fine-grained control over access and available features
- ✓ Custom models are yours alone to use, they are not shared with anyone else

#### Security: Comprehensive compliance

- ✓ We've been audited for SOC 2 compliance
- ✓ Data encryption at rest (AES-256) and in transit (TLS 1.2+)
- ✓ Visit our [Trust Portal](#) to understand more about our security measures

<https://openai.com/policies/api-data-usage-policies>

# ARE THERE ALTERNATIVES?



GitHub Co-pilot



Azure OpenAI



Meta Llano

# HOW GOOD IS AI?

Table 2: The Execution Accuracy (EX) of SOTA text-to-SQL models in BIRD. The human performance is also provided.

| Models            | Development Data |                       | Testing Data  |                       |
|-------------------|------------------|-----------------------|---------------|-----------------------|
|                   | w/o knowledge    | w/ knowledge          | w/o knowledge | w/ knowledge          |
| <i>FT-based</i>   |                  |                       |               |                       |
| T5-Base           | 6.32             | 11.54 (+5.22)         | 7.06          | 12.89 (+5.83)         |
| T5-Large          | 9.71             | 19.75 (+10.04)        | 10.38         | 20.94 (+10.56)        |
| T5-3B             | 10.37            | 23.34 (+12.97)        | 11.17         | 24.05 (+12.88)        |
| <i>ICL-based</i>  |                  |                       |               |                       |
| Codex             | 25.42            | 34.35 (+8.93)         | 24.86         | 36.47 (+11.61)        |
| ChatGPT           | 24.05            | <b>37.22</b> (+13.17) | 26.77         | 39.30 (+12.53)        |
| ChatGPT + COT     | <b>25.88</b>     | 36.64 (+10.76)        | 28.95         | 40.08 (+11.24)        |
| Human Performance | -                | -                     | <b>72.37</b>  | <b>92.96</b> (+20.59) |

Can LLM Already Serve as A Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs,  
Jinyang Li, Binyuan Hui, Ge Qu  
arXiv:2305.03111v1 [cs.CL] 4 May 2023

## WHAT ARE THE BEST USE CASES

- Good for users who are not familiar with SQL but need to work with databases
- Good for simple queries, even if they include a few joins
- Good for well-known databases with detailed documentation, like PostgreSQL
- Good point to start if you have forgotten the SQL language

## USEFUL LINKS

|                    |   |
|--------------------|---|
| EMAIL:             | <a href="mailto:tati@dbeaver.com">tati@dbeaver.com</a>      |
| DBEAVER GITHUB:    | <a href="https://github.com/dbeaver">github.com/dbeaver</a> |
| DBEAVER COMMUNITY: | <a href="https://dbeaver.io">dbeaver.io</a>                 |
| DBEAVER PRO:       | <a href="https://dbeaver.com">dbeaver.com</a>               |

# THANK YOU