how do i call a sql hostname and database through a python parameter?

###### **ChatGPT said:**

To call a SQL hostname and database using a **Python parameter**, you typically use an external library like psycopg2 for PostgreSQL or mysql.connector for MySQL. You can pass the hostname and database name as **parameters to the connection function**.

Here's how you do it generically:

### **🔧 Example using psycopg2 (PostgreSQL):**

python

CopyEdit

import psycopg2

def connect\_to\_db(hostname, dbname, username, password):

conn = psycopg2.connect(

host=hostname,

dbname=dbname,

user=username,

password=password

)

return conn

# Call it with your parameters

hostname = "my.server.com"

dbname = "my\_database"

username = "my\_user"

password = "my\_password"

conn = connect\_to\_db(hostname, dbname, username, password)

### **💡 Same pattern using mysql.connector (MySQL):**

python

CopyEdit

import mysql.connector

def connect\_to\_db(hostname, dbname, username, password):

conn = mysql.connector.connect(

host=hostname,

database=dbname,

user=username,

password=password

)

return conn

conn = connect\_to\_db("localhost", "testdb", "root", "mypassword")

### **🔄 If you want it from command-line args or a config file:**

You can use argparse, os.environ, or configparser to load those parameters and pass them in.

Would you like an example using argparse or .env file too?

4o

##### **You said:**

How do I initialize a CLI loop to create an HTTP endpoint?

###### **ChatGPT said:**

To **initialize a CLI loop to create an HTTP endpoint**, you're likely trying to:

* Start a **command-line interface (CLI)** that waits for user input (loop),
* While also hosting an **HTTP server endpoint** (e.g., with Flask or FastAPI) concurrently.

This involves **concurrency**: running both the CLI loop and the HTTP server at the same time.

## **✅ Example Using threading + Flask**

python

CopyEdit

from flask import Flask

import threading

# --- Define the HTTP server ---

app = Flask(\_\_name\_\_)

@app.route("/status")

def status():

return {"message": "Server is running!"}

def run\_server():

app.run(port=5000, debug=False, use\_reloader=False)

# --- Define the CLI Loop ---

def cli\_loop():

while True:

cmd = input(">>> ")

if cmd == "exit":

print("Shutting down...")

break

else:

print(f"Unknown command: {cmd}")

# --- Start both in parallel ---

if \_\_name\_\_ == "\_\_main\_\_":

server\_thread = threading.Thread(target=run\_server)

server\_thread.daemon = True # Allows exiting the script even if thread is alive

server\_thread.start()

cli\_loop()

## **🔍 Explanation**

* run\_server() starts the HTTP server on port 5000.
* cli\_loop() keeps taking input until exit is typed.
* threading.Thread runs the server in the background while the main thread handles the CLI.

Would you like this in an asyncio version or with FastAPI?

4o

##### **You said:**

I am running into error AttributeError: Llama has no create(), what could be the reasons for it

###### **ChatGPT said:**

The error AttributeError: Llama has no attribute create() likely means you're trying to call a method (create()) that **doesn’t exist** on the Llama object in your environment. This usually happens when using the llama-cpp-python library incorrectly or mixing it up with other LLM APIs like OpenAI's.

### **✅ Here's how to troubleshoot and fix it:**

#### **1. 🔍 Understand the library you're using**

If you're using [llama-cpp-python](https://github.com/abetlen/llama-cpp-python), **the correct class is**:

python

CopyEdit

from llama\_cpp import Llama

llm = Llama(model\_path="path/to/model.gguf")

* There is **no** create() method.
* Instead, you call the instance directly like a function:

python

CopyEdit

response = llm("What is the capital of France?")

print(response["choices"][0]["text"])

#### **2. 🧠 Don’t confuse it with OpenAI's API**

This is often a mix-up between:

python

CopyEdit

openai.ChatCompletion.create(...) # ✅ OpenAI-style

vs.

python

CopyEdit

llm(...) # ✅ llama-cpp-python-style

If you're copying code from an OpenAI example and trying to use it with llama\_cpp, it won't work.

### **✅ Correct Minimal Example:**

python

CopyEdit

from llama\_cpp import Llama

llm = Llama(model\_path="models/llama-2-7b-chat.gguf") # adjust path

output = llm("Q: What is the capital of France? A:", max\_tokens=16)

print(output["choices"][0]["text"])

Would you like help integrating this into a Flask or CLI loop setup too?

4o

##### **You said:**

so instead of llm.create i just do llm

###### **ChatGPT said:**

Yes, exactly! ✅

With llama-cpp-python, **you don't use .create()** — instead, you just **call the llm object like a function**, because the Llama class implements the \_\_call\_\_() method.

### **🔁 Instead of this (❌ incorrect):**

python

CopyEdit

llm.create("What is the capital of France?")

### **✅ Do this:**

python

CopyEdit

output = llm("What is the capital of France?", max\_tokens=32)

print(output["choices"][0]["text"])

### **🧠 Full Example:**

python

CopyEdit

from llama\_cpp import Llama

llm = Llama(model\_path="models/llama-2-7b-chat.gguf") # replace with your model path

response = llm("Q: What is the capital of France? A:", max\_tokens=32)

print(response["choices"][0]["text"])

Let me know if you're using this inside a Flask route or another framework — I can help plug it in.

4o

##### **You said:**

the model im using is not returning any actual SQL like I want it to, it is only displaying SOLUTION: which i forced to be there

###### **ChatGPT said:**

Got it — if your model is only returning something like:

makefile

CopyEdit

SOLUTION:

…but **not actually generating any SQL**, here’s why that might be happening and how to fix it.

### **🔍 Common Causes**

#### **1. Prompt is not specific or clear enough**

LLMs (especially local ones) need very structured prompting. If you just say "SOLUTION:", it may stop there if it thinks that’s all it should return.

#### **✅ Try this example prompt:**

python

CopyEdit

prompt = """You are an expert SQL assistant.

USER: Get the total sales per customer in the 'orders' table.

SOLUTION:

SELECT customer\_id, SUM(sales) FROM orders GROUP BY customer\_id;

USER: {user\_question}

SOLUTION:""".format(user\_question="Find the average age of users by city from the users table")

The pattern of input-output helps the model understand what it’s supposed to return.

#### **2. The model isn’t fine-tuned for SQL generation**

If you’re using a generic base model (e.g. LLaMA 2 without fine-tuning), it may **not know how to write SQL well**, especially for complex queries.

✅ Consider:

* Using a model **fine-tuned for SQL tasks** like codellama-instruct, mistral-instruct, or text-to-sql models.
* Adding **lots of SQL examples** in your prompt as few-shot demonstrations.

#### **3. Your prompt ends prematurely**

Make sure you're not truncating the prompt and that max\_tokens is high enough (e.g. 64–128+) to allow for full SQL generation.

python

CopyEdit

response = llm(prompt, max\_tokens=128)

print(response["choices"][0]["text"])

### **✅ Recommended Fix Flow**

1. Build a strong, few-shot prompt with USER: → SOLUTION: format.
2. Make sure max\_tokens is high enough (e.g., 128).
3. Use a model that’s trained or fine-tuned on SQL/text generation tasks.
4. Verify your model isn't stopping at \n or other stop sequences.

Would you like me to generate a full working prompt template that you can copy/paste?

4o

##### **You said:**

is there any way to force only the SQL string to show up

###### **ChatGPT said:**

Yes — you can **force the model to only return a SQL string** by crafting your prompt and decoding options carefully. Here's how:

### **✅ 1. Use a tightly formatted prompt**

Give **only SQL as expected output**, and use a delimiter like triple backticks to anchor where the SQL should go:

python

CopyEdit

prompt = """You are an AI that only returns SQL queries — no explanations.

USER: Find the names of customers who placed an order in the last 30 days.

SQL:

```sql

"""

# Call the model

output = llm(prompt, max\_tokens=128)

sql\_text = output["choices"][0]["text"]

# Post-process to extract the SQL only (optional)

import re

match = re.search(r"```sql\n(.\*?)```", sql\_text, re.DOTALL)

clean\_sql = match.group(1).strip() if match else sql\_text.strip()

print(clean\_sql)

### **✅ 2. Use a stop sequence**

If your model supports it (e.g., llama-cpp does), set a **stop token** to cut off after the SQL ends.

python

CopyEdit

output = llm(prompt, max\_tokens=128, stop=["```", "\n\n", "USER:", "SOLUTION:"])

This helps prevent rambling or extra commentary after the SQL.

### **✅ 3. Set temperature to 0**

Make sure the model is deterministic and not chatty:

python

CopyEdit

output = llm(prompt, max\_tokens=128, temperature=0)

### **🧠 Summary: Combine these for best results**

* Prompt format: SQL:\n```sql\n then end with `\n````
* Set temperature=0 and add stop=["```"]
* Extract SQL via regex if needed

Would you like a reusable function that does all of this in one go?

4o

##### **You said:**

could you make a skeleton of an angular UI with a text box, submit button, interface, and modifiable tables

###### **ChatGPT said:**

Absolutely! Here's a **skeleton Angular UI** that includes:

* ✅ A **text box** for input (like a natural language query)
* ✅ A **submit button**
* ✅ An **interface** for your query result (assume it's tabular)
* ✅ A **table** that displays results and is easily modifiable (e.g., sortable or editable later)

### **🔧 Step-by-step Setup**

#### **1. Component HTML (app.component.html)**

html

CopyEdit

<div class="p-4">

<h2 class="text-xl font-semibold mb-4">Natural Language SQL Query</h2>

<div class="flex gap-2 mb-4">

<input type="text"

[(ngModel)]="userQuery"

placeholder="Enter your question..."

class="border p-2 w-full rounded" />

<button (click)="submitQuery()"

class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600">

Submit

</button>

</div>

<table \*ngIf="tableData.length" class="w-full border-collapse border border-gray-300">

<thead>

<tr>

<th \*ngFor="let header of tableHeaders" class="border p-2">{{ header }}</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let row of tableData">

<td \*ngFor="let header of tableHeaders" class="border p-2">{{ row[header] }}</td>

</tr>

</tbody>

</table>

</div>

#### **2. Component TypeScript (app.component.ts)**

ts

CopyEdit

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html'

})

export class AppComponent {

userQuery: string = '';

tableHeaders: string[] = [];

tableData: any[] = [];

submitQuery(): void {

// Dummy response simulation

this.tableHeaders = ['id', 'name', 'email'];

this.tableData = [

{ id: 1, name: 'Alice', email: 'alice@example.com' },

{ id: 2, name: 'Bob', email: 'bob@example.com' }

];

// In a real app: call backend API with `this.userQuery`

// and update `tableHeaders` and `tableData` based on the response.

}

}

#### **3. App Module Update (app.module.ts)**

Ensure FormsModule is imported so [(ngModel)] works:

ts

CopyEdit

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({

declarations: [AppComponent],

imports: [BrowserModule, FormsModule],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

### **⚙️ Modify Later**

You can:

* Add sorting/filtering via ngFor pipes or Angular Material
* Make the table editable with contenteditable
* Connect submitQuery() to an actual backend endpoint via HttpClient

Would you like me to extend this with backend integration or Angular Material styling?

4o

##### **You said:**

could you put a pop up when hitting submit that shows that a response is loading

###### **ChatGPT said:**

Definitely! You can use a **popup (modal or overlay)** to show a "Loading..." message when the user hits **Submit**. Here's how to add that to the Angular skeleton:

### **✅ Step-by-Step Update**

#### **1. Update app.component.ts to Track Loading State**

ts

CopyEdit

loading: boolean = false;

submitQuery(): void {

this.loading = true;

// Simulate an async call (replace with actual HTTP later)

setTimeout(() => {

this.tableHeaders = ['id', 'name', 'email'];

this.tableData = [

{ id: 1, name: 'Alice', email: 'alice@example.com' },

{ id: 2, name: 'Bob', email: 'bob@example.com' }

];

this.loading = false;

}, 2000);

}

#### **2. Update app.component.html with the Popup Overlay**

Add this **above or below your main content**, inside the top <div>:

html

CopyEdit

<!-- Loading Popup -->

<div \*ngIf="loading" class="fixed inset-0 flex items-center justify-center bg-black bg-opacity-50 z-50">

<div class="bg-white p-6 rounded shadow-xl text-center">

<h2 class="text-lg font-semibold mb-2">Processing...</h2>

<p class="text-gray-600">Please wait while we generate your SQL query and results.</p>

</div>

</div>

#### **3. ✅ Bonus: Add Spinner (Optional)**

You can add a CSS spinner inside the popup if you want:

html

CopyEdit

<div class="animate-spin rounded-full h-6 w-6 border-t-2 border-b-2 border-blue-500 mx-auto my-2"></div>

Add it inside the div with bg-white.

### **✨ Final Result**

When the **submit button** is clicked:

* The loading variable becomes true
* The popup shows up and blocks the screen
* After the simulated async task finishes, loading becomes false and the popup disappears

Would you like me to refactor this into a reusable Angular component or service for loading modals?

4o

##### **You said:**

How do I make it so that example prompts come up in the frontend? (ie, there are buttons that have these prompts and when you click on them the text space is filled with the text of the button that was clicked)

###### **ChatGPT said:**

Perfect use case! To implement **example prompt buttons** that autofill the text box when clicked, follow this simple pattern:

## **✅ Goal**

* Display buttons with example prompts
* When a button is clicked, it fills the input box (userQuery) with that prompt

### **🛠️ Step-by-Step Implementation**

#### **1. Update app.component.ts with example prompts**

ts

CopyEdit

export class AppComponent {

userQuery: string = '';

tableHeaders: string[] = [];

tableData: any[] = [];

loading: boolean = false;

examplePrompts: string[] = [

"List all employees hired after 2020.",

"Show total sales grouped by region.",

"Find all users with no orders."

];

usePrompt(prompt: string): void {

this.userQuery = prompt;

}

submitQuery(): void {

this.loading = true;

// Simulated backend call

setTimeout(() => {

this.tableHeaders = ['id', 'name', 'email'];

this.tableData = [

{ id: 1, name: 'Alice', email: 'alice@example.com' },

{ id: 2, name: 'Bob', email: 'bob@example.com' }

];

this.loading = false;

}, 2000);

}

}

#### **2. Update app.component.html to show example prompt buttons**

Place this **above or below the input box**, wherever fits best:

html

CopyEdit

<div class="mb-4">

<p class="mb-2 font-medium text-gray-700">Try an example:</p>

<div class="flex flex-wrap gap-2">

<button \*ngFor="let prompt of examplePrompts"

(click)="usePrompt(prompt)"

class="bg-gray-200 px-3 py-1 rounded hover:bg-gray-300">

{{ prompt }}

</button>

</div>

</div>

### **✅ How it works**

* Each example prompt becomes a clickable button.
* Clicking one sets userQuery = prompt, autofilling the text input.
* The input remains editable after filling.

Would you like to also remember the last selected prompt across page reloads using localStorage?

4o

##### **You said:**

how do i display an example schema

###### **ChatGPT said:**

To display an **example schema** in your Angular frontend (e.g., to show the user what tables and columns exist), you can present it as:

* ✅ A formatted text block (easiest)
* ✅ A styled table (if you want structure)
* ✅ A collapsible panel (for longer schemas)

## **✅ Option 1: Code Block (Formatted Text)**

### **Step 1: Add this to your HTML (app.component.html):**

html

CopyEdit

<div class="mb-4">

<h3 class="text-lg font-semibold mb-2">Example Schema</h3>

<pre class="bg-gray-100 p-4 rounded font-mono overflow-x-auto text-sm">

TABLE users (

id INT,

name TEXT,

email TEXT,

city TEXT

)

TABLE orders (

id INT,

user\_id INT,

order\_date DATE,

total DECIMAL

)

</pre>

</div>

This uses a <pre> block with monospace formatting, suitable for most schema displays.

## **✅ Option 2: Collapsible Panel**

If you want a cleaner UI, make it collapsible:

html

CopyEdit

<button (click)="showSchema = !showSchema"

class="text-blue-600 underline mb-2">

{{ showSchema ? 'Hide' : 'Show' }} Example Schema

</button>

<div \*ngIf="showSchema">

<pre class="bg-gray-100 p-4 rounded font-mono overflow-x-auto text-sm">

TABLE users (

id INT,

name TEXT,

email TEXT,

city TEXT

)

TABLE orders (

id INT,

user\_id INT,

order\_date DATE,

total DECIMAL

)

</pre>

</div>

And in app.component.ts:

ts

CopyEdit

showSchema: boolean = false;

## **✅ Option 3: JSON-Style Schema (Optional)**

If you want to show the schema **dynamically** (maybe from the backend), you can store it as an array of objects and use \*ngFor.

Let me know if you'd prefer that version with live rendering from an API or JSON object.

4o

##### **You said:**

ok great, it works, but how do i make it so that its not at the very left side of the page

###### **ChatGPT said:**

Awesome! To move your schema display (and the rest of your content) away from the **left edge** of the page, you can:

## **✅ Option 1: Use Padding or Margin on the Outer Container**

Wrap your content in a div with horizontal padding or margin:

html

CopyEdit

<div class="max-w-4xl mx-auto px-4 py-6">

<!-- Your content here (schema, inputs, buttons, etc.) -->

</div>

### **Explanation:**

* mx-auto centers the container horizontally
* px-4 adds horizontal padding
* max-w-4xl keeps content readable on large screens
* py-6 adds top/bottom spacing

## **✅ Option 2: Apply to Body or App Root (Global Layout Fix)**

If everything is too left-aligned by default, you can fix it in your global app.component.html or layout component like this:

html

CopyEdit

<body class="bg-white text-gray-800">

<main class="max-w-5xl mx-auto px-6 py-8">

<app-root></app-root>

</main>

</body>

This ensures all pages are padded and centered.

## **✅ Option 3: Add a Container in app.component.html**

Wrap your existing Angular UI like so:

html

CopyEdit

<div class="container mx-auto px-4 py-6">

<h1 class="text-2xl font-bold mb-4">Your App Title</h1>

<!-- example prompts, input, buttons, schema display, table -->

</div>

Let me know if you'd like to add a sidebar or grid layout next!

4o

##### **You said:**

.query-card { max-width: 800px; /\* Optional: limit how wide it grows \*/ margin: 2rem auto; /\* centers it horizontally \*/ /\* or, if you want just a left‐indent without centering: \*/ /\* margin: 2rem 1rem 0 2rem; \*/ } would this work

###### **ChatGPT said:**

Yes, that CSS absolutely works — and here's how to interpret it:

### **✅ What this .query-card style does:**

css

CopyEdit

.query-card {

max-width: 800px; /\* Limits width to keep it readable \*/

margin: 2rem auto; /\* Top margin: 2rem; Bottom: 0 (default); Left/Right: auto (centers horizontally) \*/

}

This centers the element **horizontally** and adds vertical spacing (2rem from the top). It's a great way to keep your query box or card layout clean and centered on the page.

### **✅ If you want a left-indent instead of centering, uncomment this:**

css

CopyEdit

/\* margin: 2rem 1rem 0 2rem; \*/

That gives you:

* Top: 2rem
* Right: 1rem
* Bottom: 0
* Left: 2rem → creates a left indent

### **🔧 How to apply it**

In your HTML:

html

CopyEdit

<div class="query-card">

<!-- your textbox, button, schema, table, etc. -->

</div>

Just make sure .query-card is defined in your component CSS or in styles.css depending on scope.

Want help applying consistent padding/margins across your whole app layout too?

4o

##### **You said:**

how do i display an example schema IN CSS

###### **ChatGPT said:**

If you want to **display an example schema in CSS** (i.e., style a block of schema text to look nice), you can't literally *write* the schema in CSS, but you can **style it** using CSS applied to an HTML element like <pre>, <code>, or a <div>.

### **✅ HTML + CSS Example for Schema Display**

#### **🔹 HTML (app.component.html)**

html

CopyEdit

<div class="schema-box">

<pre>

TABLE users (

id INT,

name TEXT,

email TEXT,

city TEXT

)

TABLE orders (

id INT,

user\_id INT,

order\_date DATE,

total DECIMAL

)

</pre>

</div>

#### **🔹 CSS (app.component.css or styles.css)**

css

CopyEdit

.schema-box {

background-color: #f5f5f5; /\* light gray background \*/

border: 1px solid #ccc; /\* light border \*/

border-radius: 6px;

padding: 1rem;

font-family: 'Courier New', monospace;

white-space: pre-wrap; /\* allows wrapping if needed \*/

max-width: 800px;

margin: 2rem auto; /\* centers on the page \*/

box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1); /\* optional shadow \*/

}

### **🧠 Notes:**

* Use <pre> for preserving indentation and line breaks
* Style .schema-box to make it look like a terminal or code block
* You can also use <code> inside <pre> if you want syntax highlighting (manually or via library)

Would you like a fancier look using a theme (like a dark terminal-style schema block)?

4o

##### **You said:**

ok great, it works, but how do i make it so that its not at the very left side of the page but in css this time please

###### **ChatGPT said:**

Great question — to move your schema (or **any element**) away from the **left edge of the page using CSS**, you have a few clean options:

### **✅ Option 1: Center It Horizontally (Preferred for Readability)**

css

CopyEdit

.schema-box {

max-width: 800px;

margin: 2rem auto; /\* Top/Bottom = 2rem, Left/Right = auto (centers the box) \*/

padding: 1rem;

background-color: #f5f5f5;

border-radius: 6px;

font-family: monospace;

}

* ✅ margin: auto horizontally **centers** the block.
* ✅ max-width keeps it readable.

### **✅ Option 2: Left Indent Without Centering**

css

CopyEdit

.schema-box {

margin-left: 2rem; /\* Indent from the left \*/

margin-top: 2rem;

max-width: 800px;

padding: 1rem;

background-color: #f5f5f5;

border-radius: 6px;

font-family: monospace;

}

* Use this if you want it **left-aligned but indented** instead of centered.

### **✅ Option 3: Full-Width but Padded**

css

CopyEdit

.schema-box {

width: 100%;

padding-left: 2rem;

padding-right: 2rem;

padding-top: 1rem;

padding-bottom: 1rem;

background-color: #f5f5f5;

font-family: monospace;

}

* Use this when you want it to stretch across the screen but not touch the sides.

Let me know if you want to add a sidebar, flex layout, or responsive behavior too!

4o

##### **You said:**

the LLM output table is displaying nothing for some reason

###### **ChatGPT said:**

Got it — if your **LLM output table is displaying nothing**, here’s a quick checklist to debug it:

### **✅ 1. Is tableData populated?**

Check in your app.component.ts if tableData actually has content:

ts

CopyEdit

console.log(this.tableData); // Add this inside submitQuery()

You should see an array like:

ts

CopyEdit

[

{ id: 1, name: 'Alice', email: 'alice@example.com' },

{ id: 2, name: 'Bob', email: 'bob@example.com' }

]

If it’s empty, the table won’t show anything.

### **✅ 2. Is tableHeaders correct and matching the keys?**

If your headers are wrong or don't match the keys in the data, the cells will render blank.

Check:

ts

CopyEdit

console.log(this.tableHeaders); // Should be: ['id', 'name', 'email']

Then make sure they match the keys in your data objects.

### **✅ 3. \*\*Check the *ngIf and ngFor conditions***

Here’s the working version of the table (confirm yours looks like this):

html

CopyEdit

<table \*ngIf="tableData.length" class="w-full border-collapse border border-gray-300">

<thead>

<tr>

<th \*ngFor="let header of tableHeaders" class="border p-2">{{ header }}</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let row of tableData">

<td \*ngFor="let header of tableHeaders" class="border p-2">

{{ row[header] }}

</td>

</tr>

</tbody>

</table>

* ✅ \*ngIf="tableData.length" makes sure table only appears if there's data.
* ✅ row[header] must match the keys in each object.

### **✅ 4. Was the LLM response parsed correctly?**

If your LLM returns a SQL result as a string or JSON, make sure it's parsed into a usable object:

ts

CopyEdit

const rawResult = '[{"id": 1, "name": "Alice"}, {"id": 2, "name": "Bob"}]';

this.tableData = JSON.parse(rawResult);

this.tableHeaders = Object.keys(this.tableData[0]);

### **✅ 5. Angular isn’t updating? Trigger change detection**

If you update data outside Angular’s zone (e.g., from a promise or custom event), use:

ts

CopyEdit

import { ChangeDetectorRef } from '@angular/core';

constructor(private cd: ChangeDetectorRef) {}

someAsyncCallback() {

this.tableData = [...];

this.cd.detectChanges(); // forces Angular to update view

}

Would you like to paste your current submitQuery() and table HTML so I can spot the exact issue?

4o

##### **You said:**

please try and help in python

###### **ChatGPT said:**

Perfect — if the **LLM (Python backend)** is returning a result, but **your Angular table shows nothing**, it likely means the **Python output isn’t structured correctly** for the frontend to render.

Let’s fix it from the Python side.

### **✅ Goal: Return a well-structured JSON response like this:**

json

CopyEdit

{

"headers": ["id", "name", "email"],

"rows": [

{"id": 1, "name": "Alice", "email": "alice@example.com"},

{"id": 2, "name": "Bob", "email": "bob@example.com"}

]

}

### **✅ Backend Example (Flask)**

python

CopyEdit

from flask import Flask, request, jsonify

from llama\_cpp import Llama

import sqlite3 # or whatever DB you're using

app = Flask(\_\_name\_\_)

llm = Llama(model\_path="path/to/model.gguf")

@app.route('/query', methods=['POST'])

def run\_query():

user\_prompt = request.json.get("prompt")

# 1. Generate SQL from the LLM

response = llm(f"SOLUTION:\n{user\_prompt}", max\_tokens=128)

sql = response["choices"][0]["text"].strip()

# 2. Run SQL against your database

conn = sqlite3.connect("your.db")

cursor = conn.cursor()

try:

cursor.execute(sql)

rows = cursor.fetchall()

headers = [desc[0] for desc in cursor.description]

except Exception as e:

return jsonify({"error": str(e)}), 400

finally:

conn.close()

# 3. Return JSON to frontend

row\_dicts = [dict(zip(headers, row)) for row in rows]

return jsonify({"headers": headers, "rows": row\_dicts})

### **✅ Angular should consume this like:**

ts

CopyEdit

this.http.post('/query', { prompt: this.userQuery }).subscribe((res: any) => {

this.tableHeaders = res.headers;

this.tableData = res.rows;

});

### **🔍 What Could Be Wrong**

| **Problem** | **Fix** |
| --- | --- |
| LLM outputs only text, not SQL | Extract SQL with a regex or clean formatting |
| SQL is invalid or throws error | Catch it and return {"error": "..."} |
| SQL result is a list of tuples | Convert to list of dicts with headers |
| JSON response not structured | Always return { headers: [...], rows: [...] } |

Want me to generate an example that runs on a fake SQLite table for testing?

4o

##### **You said:**

File "/common/home/jfs199/Desktop/Project02/database\_llm.py", line 11, in <module> conn.executescript(f.read()) sqlite3.OperationalError: near ")": syntax error

###### **ChatGPT said:**

The error you're getting:

go

CopyEdit

sqlite3.OperationalError: near ")": syntax error

means there's a **syntax issue in your SQL script**, likely in a CREATE TABLE or INSERT statement — and specifically it's happening **near a closing parenthesis**.

### **🔍 How to fix it**

#### **1. Find the broken SQL line**

Open the SQL file you're reading with f.read(). Look for something like:

sql

CopyEdit

CREATE TABLE users (

id INTEGER,

name TEXT,

email TEXT,

); -- ← Trailing comma here causes the error

or:

sql

CopyEdit

CREATE TABLE orders (

id INTEGER

); -- ← maybe missing a comma if you meant to have more columns

Check all CREATE TABLE, INSERT INTO, etc., for:

* ❌ **Trailing commas before closing )**
* ❌ **Empty column lists**
* ❌ **Missing commas between columns**

#### **2. ✅ Fix the SQL Script**

**Correct version:**

sql

CopyEdit

CREATE TABLE users (

id INTEGER,

name TEXT,

email TEXT

);

### **🔧 Example Fix in Python**

If you’re executing this in Python:

python

CopyEdit

with open("init.sql", "r") as f:

conn.executescript(f.read())

Make sure the contents of init.sql are valid SQL.

Would you like to paste your SQL schema file here so I can tell you exactly where the problem is?

4o

##### **You said:**

this is happening in the python backend, not sql

###### **ChatGPT said:**

Got it — you're absolutely right that this is happening **inside your Python backend**, but the error:

go

CopyEdit

sqlite3.OperationalError: near ")": syntax error

still refers to a **malformed SQL statement being passed into SQLite** from Python — specifically via:

python

CopyEdit

conn.executescript(f.read())

This line reads a SQL file (e.g., a schema or seed data file) and runs its contents using sqlite3.

### **✅ What This Means**

The **problem is in the SQL text** you're feeding into executescript(). The **Python part is fine** — it's just forwarding broken SQL.

So:

1. Open the SQL file you're reading (e.g., init.sql)
2. Find any CREATE TABLE or INSERT INTO statements
3. Check for:

#### **🔴 Common Syntax Errors:**

| **Error** | **Fix** |
| --- | --- |
| email TEXT, ) (trailing comma) | remove the comma before ) |
| CREATE TABLE foo () | add at least one column definition |
| INSERT INTO users VALUES (); | ensure column values match schema |

### **✅ Example**

#### **❌ Broken SQL (causes your error)**

sql

CopyEdit

CREATE TABLE users (

id INTEGER,

name TEXT,

);

#### **✅ Correct SQL**

sql

CopyEdit

CREATE TABLE users (

id INTEGER,

name TEXT

);

### **✅ How to Debug in Python**

You can print the SQL script before executing it to see where it's going wrong:

python

CopyEdit

with open("init.sql", "r") as f:

sql\_script = f.read()

print("DEBUG: Executing SQL script:\n", sql\_script)

conn.executescript(sql\_script)

If you send me the contents of the SQL file you’re using, I can spot the exact issue for you. Want to paste it here?

4o

##### **You said:**

jfs199@kill:~/Desktop/Project02/db-llm-frontend$ npm start > db-llm-frontend@0.0.1 start > ng serve --proxy-config proxy.conf.json ✔ Browser application bundle generation complete. Initial Chunk Files | Names | Raw Size vendor.js | vendor | 4.15 MB | polyfills.js | polyfills | 234.27 kB | styles.css, styles.js | styles | 129.49 kB | main.js | main | 43.86 kB | runtime.js | runtime | 6.51 kB | | Initial Total | 4.55 MB Build at: 2025-05-09T06:35:23.490Z - Hash: 6fff432531f2e54f - Time: 8844ms Error: src/app/components/query/query.component.ts:61:41 - error TS2554: Expected 1 arguments, but got 3. 61 this.qs.sendQuery(this.userQuery, netid, password) ~~~~~~~~~~~~~~~ \*\* Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ \*\* ✖ Failed to compile. This is what returns when I try to run

###### **ChatGPT said:**

Thanks — this error:

go

CopyEdit

error TS2554: Expected 1 arguments, but got 3.

means you're calling a function with **too many arguments**, specifically this line:

ts

CopyEdit

this.qs.sendQuery(this.userQuery, netid, password)

…but the sendQuery() method (probably in your QueryService) **only accepts one argument** — likely just this.userQuery.

### **✅ Fix Options**

#### **🛠️ Option 1: Modify the Function Call (Frontend Fix)**

If you **don’t need netid and password**, just change the call to:

ts

CopyEdit

this.qs.sendQuery(this.userQuery);

#### **🛠️ Option 2: Modify the Function Definition (Backend/Service Fix)**

If your backend is expecting all three (e.g., for authentication), change the function signature in query.service.ts to:

ts

CopyEdit

sendQuery(query: string, netid: string, password: string) {

return this.http.post('/query', {

prompt: query,

netid: netid,

password: password

});

}

### **🔍 Where to Check**

Go to:

bash

CopyEdit

src/app/services/query.service.ts

And look for:

ts

CopyEdit

sendQuery(...)

Update its parameters to match how you're calling it from the component.

Would you like help writing the backend Flask endpoint to handle netid and password securely too?

4o

##### **You said:**

how do i make the text appear ABOVE the text boxes instead of next to them?

###### **ChatGPT said:**

Great question — to make **labels or text appear above text boxes** instead of next to them in Angular (or plain HTML), you just need to:

### **✅ Wrap each input and label in a vertical layout (block-level)**

Here’s the **simple HTML structure**:

html

CopyEdit

<div class="mb-4">

<label for="userQuery" class="block mb-1 font-medium text-gray-700">Your Query</label>

<input id="userQuery"

[(ngModel)]="userQuery"

type="text"

class="w-full border rounded px-3 py-2" />

</div>

<div class="mb-4">

<label for="netid" class="block mb-1 font-medium text-gray-700">NetID</label>

<input id="netid"

[(ngModel)]="netid"

type="text"

class="w-full border rounded px-3 py-2" />

</div>

<div class="mb-4">

<label for="password" class="block mb-1 font-medium text-gray-700">Password</label>

<input id="password"

[(ngModel)]="password"

type="password"

class="w-full border rounded px-3 py-2" />

</div>

### **✅ Explanation of Key Classes**

* class="block" → makes the <label> a block element (so it appears on its own line)
* mb-1 → small spacing between label and input
* mb-4 on the wrapper → spacing between field blocks
* w-full → makes input full width

Would you like a styled version with floating labels or inline validation too?

4o

##### **You said:**

when you click submit and have to enter credentials, how do you disable the submit button so you cant submit while entering your netid and password

###### **ChatGPT said:**

Great — to **disable the submit button** while the user is entering their NetID and password (i.e., after clicking Submit but before they confirm their credentials), you just need to:

### **✅ Use a Boolean Flag to Track the State**

#### **🔹 In app.component.ts:**

ts

CopyEdit

isAwaitingCredentials: boolean = false;

submitQuery(): void {

// Set to true when user needs to enter credentials

this.isAwaitingCredentials = true;

// Show credential input UI (modal, section, etc.)

}

### **✅ Disable the Button in app.component.html**

html

CopyEdit

<button

(click)="submitQuery()"

[disabled]="isAwaitingCredentials"

class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600 disabled:bg-gray-400">

Submit

</button>

* [disabled]="isAwaitingCredentials" disables the button
* disabled:bg-gray-400 (if using Tailwind) changes button appearance when disabled

### **✅ Re-enable When Ready**

When credentials are entered and you're ready to send the request:

ts

CopyEdit

confirmCredentials(): void {

this.isAwaitingCredentials = false;

// Now send the query

this.qs.sendQuery(this.userQuery, this.netid, this.password).subscribe(...);

}

Would you like to show the credential inputs in a popup/modal when Submit is clicked?

4o

##### **You said:**

how do you use fastAPI to send a query to ILAB, run and then return the results to the frontend?

###### **ChatGPT said:**

To use **FastAPI to send a SQL query to Rutgers iLab**, run it on the database, and return results to the frontend, you’ll need to:

## **✅ Setup Overview**

1. **Use FastAPI as your backend**
2. **Connect to iLab’s PostgreSQL or MySQL database via SSH tunnel**
3. **Execute the query**
4. **Return results as JSON to your Angular frontend**

### **🔧 Step 1: Install Dependencies**

bash

CopyEdit

pip install fastapi uvicorn asyncpg paramiko sshtunnel

(Use psycopg2 or mysql-connector-python depending on your DB engine.)

### **🔧 Step 2: Use SSH Tunnel to Access iLab (if needed)**

If iLab requires SSH tunneling (likely), use sshtunnel:

python

CopyEdit

from sshtunnel import SSHTunnelForwarder

server = SSHTunnelForwarder(

('ilab-login.rutgers.edu', 22), # SSH hostname and port

ssh\_username='your\_netid',

ssh\_password='your\_password',

remote\_bind\_address=('127.0.0.1', 5432), # DB host/port on the iLab machine

local\_bind\_address=('127.0.0.1', 6543) # Local port you'll connect to

)

server.start()

Now your local Python app can connect to the iLab DB like it’s local.

### **🔧 Step 3: FastAPI Backend Code**

python

CopyEdit

from fastapi import FastAPI, Request

from pydantic import BaseModel

import psycopg2

from sshtunnel import SSHTunnelForwarder

app = FastAPI()

class QueryRequest(BaseModel):

prompt: str

netid: str

password: str

@app.post("/query")

def run\_query(req: QueryRequest):

# Create tunnel to iLab

with SSHTunnelForwarder(

('ilab-login.rutgers.edu', 22),

ssh\_username=req.netid,

ssh\_password=req.password,

remote\_bind\_address=('127.0.0.1', 5432),

local\_bind\_address=('127.0.0.1', 6543)

) as tunnel:

# Connect to the DB through the tunnel

conn = psycopg2.connect(

host='127.0.0.1',

port=6543,

user='your\_db\_user', # likely same as netid

password=req.password,

database='your\_db\_name'

)

cursor = conn.cursor()

try:

cursor.execute(req.prompt)

rows = cursor.fetchall()

columns = [desc[0] for desc in cursor.description]

results = [dict(zip(columns, row)) for row in rows]

except Exception as e:

return {"error": str(e)}

finally:

conn.close()

return {"headers": columns, "rows": results}

### **🔧 Step 4: Connect from Angular**

ts

CopyEdit

this.http.post('/query', {

prompt: this.userQuery,

netid: this.netid,

password: this.password

}).subscribe((res: any) => {

this.tableHeaders = res.headers;

this.tableData = res.rows;

});

### **🔐 Security Note**

* Don’t log passwords
* Use HTTPS if deployed
* Ideally use key-based auth instead of password in SSH

Would you like help converting this to use key-based SSH authentication instead of password?

4o

##### **You said:**

the pop-up (with netid/password) needs to always pop up when a query is submitted

###### **ChatGPT said:**

Perfect — here’s how to make sure the **NetID/password popup always appears when a query is submitted**, and only sends the query **after the user confirms credentials**.

## **✅ Goal:**

* User clicks **Submit**
* Popup opens asking for NetID + Password
* When user confirms:  
  + Query is sent to FastAPI
  + Table is updated with results
  + Popup closes

## **🧩 1. Component Logic (app.component.ts or query.component.ts)**

ts

CopyEdit

userQuery: string = '';

netid: string = '';

password: string = '';

showCredentialPopup: boolean = false;

isLoading: boolean = false;

tableHeaders: string[] = [];

tableData: any[] = [];

// Step 1: Show popup on Submit

onSubmit(): void {

this.showCredentialPopup = true;

}

// Step 2: Send query after confirming credentials

confirmAndSend(): void {

this.isLoading = true;

this.showCredentialPopup = false;

this.qs.sendQuery(this.userQuery, this.netid, this.password).subscribe(

(res: any) => {

this.tableHeaders = res.headers;

this.tableData = res.rows;

this.isLoading = false;

},

(err) => {

console.error(err);

this.isLoading = false;

}

);

}

## **🧩 2. HTML Template (app.component.html)**

html

CopyEdit

<!-- Main query input -->

<div class="mb-4">

<label class="block mb-1 font-semibold">Query</label>

<input [(ngModel)]="userQuery" class="w-full border p-2 rounded" />

</div>

<!-- Submit Button -->

<button (click)="onSubmit()"

class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600">

Submit

</button>

<!-- Credential Popup -->

<div \*ngIf="showCredentialPopup" class="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50">

<div class="bg-white p-6 rounded shadow-lg max-w-sm w-full">

<h2 class="text-lg font-bold mb-4">Enter Credentials</h2>

<label class="block mb-1">NetID</label>

<input [(ngModel)]="netid" class="w-full border p-2 mb-3 rounded" />

<label class="block mb-1">Password</label>

<input [(ngModel)]="password" type="password" class="w-full border p-2 mb-4 rounded" />

<div class="flex justify-end gap-2">

<button (click)="showCredentialPopup = false" class="px-3 py-1 border rounded">Cancel</button>

<button (click)="confirmAndSend()" class="bg-blue-500 text-white px-4 py-1 rounded hover:bg-blue-600">

Confirm

</button>

</div>

</div>

</div>

<!-- Optional Loading Indicator -->

<div \*ngIf="isLoading" class="mt-4 text-blue-600">Running query...</div>

### **✅ Summary of What This Does:**

* **Always shows the NetID/password popup when a query is submitted**
* **Only sends the query to the backend after Confirm is clicked**
* **Cleanly disables Submit from running directly**

Would you like to turn the credential popup into a reusable component or modal library (like Angular Material)?

4o

##### **You said:**

in Angular please

###### **ChatGPT said:**

Absolutely! Here’s how to do the **always-on NetID/password popup on submit** **entirely in Angular**, step-by-step:

## **✅ Desired Behavior in Angular**

1. User types a query and clicks **Submit**
2. A popup/modal appears asking for **NetID + Password**
3. When the user confirms:  
   * The query is sent to your FastAPI backend
   * The table updates with the response
   * The popup closes

## **🧩 1. app.component.ts (or query.component.ts)**

ts

CopyEdit

import { Component } from '@angular/core';

import { HttpClient } from '@angular/common/http';

@Component({

selector: 'app-root',

templateUrl: './app.component.html'

})

export class AppComponent {

userQuery = '';

netid = '';

password = '';

showCredentialPopup = false;

isLoading = false;

tableHeaders: string[] = [];

tableData: any[] = [];

constructor(private http: HttpClient) {}

onSubmit(): void {

this.showCredentialPopup = true;

}

confirmAndSend(): void {

this.isLoading = true;

this.showCredentialPopup = false;

const payload = {

prompt: this.userQuery,

netid: this.netid,

password: this.password

};

this.http.post<any>('/query', payload).subscribe(

(res) => {

this.tableHeaders = res.headers;

this.tableData = res.rows;

this.isLoading = false;

},

(error) => {

console.error('Query error:', error);

this.isLoading = false;

}

);

}

}

## **🧩 2. app.component.html**

html

CopyEdit

<!-- Query input -->

<div class="mb-4">

<label class="block mb-1 font-semibold">Query</label>

<input [(ngModel)]="userQuery" class="w-full border p-2 rounded" />

</div>

<!-- Submit Button -->

<button (click)="onSubmit()"

class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600">

Submit

</button>

<!-- Credentials Popup -->

<div \*ngIf="showCredentialPopup" class="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50">

<div class="bg-white p-6 rounded shadow-lg max-w-sm w-full">

<h2 class="text-lg font-bold mb-4">Enter Credentials</h2>

<label class="block mb-1">NetID</label>

<input [(ngModel)]="netid" class="w-full border p-2 mb-3 rounded" />

<label class="block mb-1">Password</label>

<input [(ngModel)]="password" type="password" class="w-full border p-2 mb-4 rounded" />

<div class="flex justify-end gap-2">

<button (click)="showCredentialPopup = false" class="px-3 py-1 border rounded">Cancel</button>

<button (click)="confirmAndSend()" class="bg-blue-500 text-white px-4 py-1 rounded hover:bg-blue-600">

Confirm

</button>

</div>

</div>

</div>

<!-- Loading Spinner -->

<div \*ngIf="isLoading" class="mt-4 text-blue-600">

Loading...

</div>

<!-- Results Table -->

<table \*ngIf="tableData.length" class="mt-6 w-full border border-collapse">

<thead>

<tr>

<th \*ngFor="let header of tableHeaders" class="border p-2 text-left">{{ header }}</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let row of tableData">

<td \*ngFor="let header of tableHeaders" class="border p-2">{{ row[header] }}</td>

</tr>

</tbody>

</table>

## **✅ Recap**

* 🔁 onSubmit() always triggers the popup
* ✅ User must **confirm credentials** before any query runs
* 📦 Uses HttpClient.post() to send to your FastAPI /query endpoint

Would you like to move this popup into a reusable modal component or Angular Material dialog?

4o

##### **You said:**

and then once that is done and they get a correct net id and password combination, how do i make it so that the login dialog is skipped when another query is submitted on the same instance?

###### **ChatGPT said:**

Great follow-up! To **skip the login popup after the user has already entered valid NetID and password once**, just add a flag that tracks whether the user is already "authenticated" for this session.

## **✅ Step-by-Step Plan**

1. ✅ Always show the credential popup on the **first** query
2. ✅ After a successful login, **store the credentials**
3. ✅ On future queries, **reuse stored credentials** and **skip the popup**

## **🧩 1. Update app.component.ts (or query.component.ts)**

ts

CopyEdit

userQuery = '';

netid = '';

password = '';

credentialsConfirmed = false;

showCredentialPopup = false;

isLoading = false;

tableHeaders: string[] = [];

tableData: any[] = [];

constructor(private http: HttpClient) {}

onSubmit(): void {

// If credentials haven't been entered yet, show popup

if (!this.credentialsConfirmed) {

this.showCredentialPopup = true;

} else {

this.sendQuery(); // Skip popup

}

}

confirmAndSend(): void {

this.credentialsConfirmed = true;

this.showCredentialPopup = false;

this.sendQuery(); // Now send the query

}

sendQuery(): void {

this.isLoading = true;

const payload = {

prompt: this.userQuery,

netid: this.netid,

password: this.password

};

this.http.post<any>('/query', payload).subscribe(

(res) => {

this.tableHeaders = res.headers;

this.tableData = res.rows;

this.isLoading = false;

},

(err) => {

console.error('Query error:', err);

this.isLoading = false;

}

);

}

## **🧩 2. Your HTML stays the same (app.component.html)**

No change needed — the \*ngIf="showCredentialPopup" logic already handles the popup display.

## **✅ How This Works:**

* On the **first submit**, it shows the popup and collects credentials
* Once confirmed, it sets credentialsConfirmed = true
* On all **future submits**, it skips the popup and reuses the saved NetID/password

Would you like to store the credentials in sessionStorage so it persists across full-page reloads but not longer?

4o

##### **You said:**

how do i parse my SSH output into Angular table columns

###### **ChatGPT said:**

Great — if your backend (e.g., FastAPI + Python + SSH to iLab) returns **SSH command output** (like a SQL result), and you want to **parse it into an Angular table**, the key is to:

## **✅ Step-by-Step Plan**

### **🔹 1. Return structured data from the backend**

Avoid sending raw SSH string output. Instead, **parse it in Python** into:

json

CopyEdit

{

"headers": ["column1", "column2", "column3"],

"rows": [

{ "column1": "value1", "column2": "value2", "column3": "value3" },

...

]

}

### **🔧 Python Example (parsing SSH output)**

Assume you're using paramiko to run a command like psql -c "SELECT \* FROM table" over SSH.

python

CopyEdit

import paramiko

import json

def run\_ssh\_query(netid, password, sql):

ssh = paramiko.SSHClient()

ssh.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

ssh.connect(

hostname='ilab-login.rutgers.edu',

username=netid,

password=password

)

stdin, stdout, stderr = ssh.exec\_command(f'psql -d your\_db -c "{sql}" -A -F "," --no-align')

output = stdout.read().decode()

ssh.close()

# Example raw output:

# id,name,email

# 1,Alice,alice@example.com

# 2,Bob,bob@example.com

lines = output.strip().split('\n')

headers = lines[0].split(',')

rows = [dict(zip(headers, line.split(','))) for line in lines[1:]]

return {

"headers": headers,

"rows": rows

}

### **🟩 Angular Side: app.component.ts**

ts

CopyEdit

this.http.post('/query', {

prompt: this.userQuery,

netid: this.netid,

password: this.password

}).subscribe((res: any) => {

this.tableHeaders = res.headers;

this.tableData = res.rows;

});

### **🟩 Angular Table HTML**

html

CopyEdit

<table \*ngIf="tableData.length" class="mt-6 w-full border border-collapse">

<thead>

<tr>

<th \*ngFor="let header of tableHeaders" class="border p-2 text-left">{{ header }}</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let row of tableData">

<td \*ngFor="let header of tableHeaders" class="border p-2">{{ row[header] }}</td>

</tr>

</tbody>

</table>

### **🔍 Summary**

* SSH output must be parsed to structured JSON **in Python**
* Angular expects: { headers: [...], rows: [ {...}, {...} ] }
* Use psql -A -F "," --no-align to return **CSV-like output** from PostgreSQL over SSH

Would you like help adapting this if your SSH command returns tab-separated or messy multiline output instead?

4o

##### **You said:**

i am having trouble with the page size, how do I implement the columns per page thing?

###### **ChatGPT said:**

To implement **pagination for table rows** in Angular — often called "rows per page" or "page size" — you can do it manually or use a library like Angular Material.

I'll show you both options:

## **✅ Option 1: Manual Pagination (Vanilla Angular)**

### **🧩 1. Component (app.component.ts)**

ts

CopyEdit

tableData: any[] = []; // all rows

tableHeaders: string[] = []; // columns

pageSize: number = 10; // rows per page

currentPage: number = 1;

get pagedData(): any[] {

const start = (this.currentPage - 1) \* this.pageSize;

return this.tableData.slice(start, start + this.pageSize);

}

get totalPages(): number {

return Math.ceil(this.tableData.length / this.pageSize);

}

goToPage(page: number): void {

this.currentPage = page;

}

### **🧩 2. HTML (app.component.html)**

#### **📋 Your Table (unchanged except it uses pagedData instead of tableData)**

html

CopyEdit

<table \*ngIf="tableData.length" class="w-full border border-collapse mt-4">

<thead>

<tr>

<th \*ngFor="let header of tableHeaders" class="border p-2">{{ header }}</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let row of pagedData">

<td \*ngFor="let header of tableHeaders" class="border p-2">{{ row[header] }}</td>

</tr>

</tbody>

</table>

#### **📚 Pagination Controls**

html

CopyEdit

<div \*ngIf="tableData.length > pageSize" class="mt-2 flex items-center gap-2">

<button (click)="goToPage(currentPage - 1)" [disabled]="currentPage === 1">Prev</button>

<ng-container \*ngFor="let page of [].constructor(totalPages); let i = index">

<button (click)="goToPage(i + 1)" [class.font-bold]="currentPage === (i + 1)">

{{ i + 1 }}

</button>

</ng-container>

<button (click)="goToPage(currentPage + 1)" [disabled]="currentPage === totalPages">Next</button>

</div>

## **✅ Option 2: Use Angular Material Table (Quick Setup)**

Install Angular Material (if you want a built-in paginator):

bash

CopyEdit

ng add @angular/material

Use <mat-paginator> with <mat-table>. If you’d like this version, I can walk you through the exact setup.

### **✅ Recap**

| **You Want...** | **Use This** |
| --- | --- |
| Fully manual control | Option 1 (above) ✅ |
| Out-of-the-box UI | Angular Material |

Would you like page size options (e.g., "Show 10, 25, 50 rows") added too?

4o

##### **You said:**

how do i split results into separate columns, because right now its just a combination and a mess of text

###### **ChatGPT said:**

Reasoned for a couple of seconds

If your results are coming back as a single string or an unstructured blob, you'll first need to ensure that your backend is returning structured data (e.g. a JSON array of objects where each object corresponds to a row with key/value pairs for each column). For example, instead of sending a string like:

perl

CopyEdit

"1, Alice, alice@example.com\n2, Bob, bob@example.com"

your backend should return something like:

json

CopyEdit

{

"headers": ["id", "name", "email"],

"rows": [

{ "id": 1, "name": "Alice", "email": "alice@example.com" },

{ "id": 2, "name": "Bob", "email": "bob@example.com" }

]

}

Once your data is structured, you can use Angular Material’s table components (which include built-in sorting using MatSort) to display your data across separate columns.

Below is an example of how you can set up an Angular Material table that sorts by columns.

## **1. Backend Data Structure (Ensure Proper Formatting)**

Make sure that your query endpoint (e.g. using FastAPI or Flask) returns JSON with a "headers" array and a "rows" array of objects. For example:

json

CopyEdit

{

"headers": ["id", "name", "email"],

"rows": [

{ "id": 1, "name": "Alice", "email": "alice@example.com" },

{ "id": 2, "name": "Bob", "email": "bob@example.com" }

]

}

This structure lets you iterate over headers for the table columns and rows for the data.

## **2. Angular Material Table Setup**

### **Step A: Install Angular Material (if not already installed)**

bash

CopyEdit

ng add @angular/material

### **Step B: Set Up the Component**

#### **app.component.ts**

ts

CopyEdit

import { Component, OnInit, ViewChild } from '@angular/core';

import { MatTableDataSource } from '@angular/material/table';

import { MatSort } from '@angular/material/sort';

import { HttpClient } from '@angular/common/http';

export interface RowData {

[key: string]: any;

}

@Component({

selector: 'app-root',

templateUrl: './app.component.html'

})

export class AppComponent implements OnInit {

displayedColumns: string[] = [];

dataSource = new MatTableDataSource<RowData>();

// Reference to the MatSort directive in the template

@ViewChild(MatSort) sort!: MatSort;

constructor(private http: HttpClient) {}

ngOnInit() {

this.sendQuery();

}

sendQuery(): void {

// Replace '/query' with your actual endpoint URL

this.http.post<any>('/query', { prompt: 'some query', netid: 'yourNetid', password: 'yourPass' })

.subscribe((res) => {

// Assume res.headers is an array of strings and res.rows is an array of objects.

this.displayedColumns = res.headers;

this.dataSource.data = res.rows;

// Set sort after data is assigned

this.dataSource.sort = this.sort;

}, error => {

console.error(error);

});

}

}

#### **app.component.html**

html

CopyEdit

<div class="mat-elevation-z8">

<table mat-table [dataSource]="dataSource" matSort class="full-width-table">

<!-- Define column definitions dynamically -->

<ng-container \*ngFor="let column of displayedColumns" [matColumnDef]="column">

<th mat-header-cell \*matHeaderCellDef mat-sort-header> {{ column | titlecase }} </th>

<td mat-cell \*matCellDef="let element"> {{ element[column] }} </td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

<tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

</table>

</div>

#### **app.component.css (optional)**

css

CopyEdit

.full-width-table {

width: 100%;

margin-top: 20px;

}

## **3. Explanation**

* **Data Parsing:** Your backend should parse the raw SSH output or query result into a JSON object with headers and rows. Angular then treats each row as an object, and each key in that object is mapped to a column.
* **Dynamic Columns:** The <ng-container> block iterates over displayedColumns to create column definitions dynamically. Each column header is given the mat-sort-header attribute to enable sorting.
* **Sorting with MatSort:** The MatTableDataSource is assigned to use this.sort (which is a reference to the <table>'s sort directive). This provides built-in sorting functionality.
* **MatSort Usage:** When the user clicks on a column header, MatSort automatically sorts the table data by that column.

## **4. Troubleshooting Tips**

* **Messy Text Issue:** If your results are still coming as a “combination and a mess of text,” double-check your backend logic. Ensure that before sending the data to the frontend, you parse (for example, using string splitting, regex, or CSV parsing libraries) and format the output into a structured object.

**MatSort Not Working:** Verify that you've imported the necessary Angular Material modules in your app module:  
  
 ts  
CopyEdit  
import { MatTableModule } from '@angular/material/table';

import { MatSortModule } from '@angular/material/sort';

@NgModule({

imports: [

// ... other imports

MatTableModule,

MatSortModule

],

})

export class AppModule { }

By following these steps, your Angular frontend will display your query result in separate columns (with sorting functionality via MatSort) instead of a messy text blob. Let me know if you need further assistance or clarifications!