

[Open in app](#)



Search



Write



Build Analytics Assistant LLM App. Connected to Live Data Warehouses.



Amar Harolikar

19 min read · Aug 7, 2024



50



1



...

Flowise AI, Text-to-SQL, FastAPI. Connect & Analyze. AWS-Azure MySQL Example



AWS



Azure



MySQL



Data Warehouse

Analytics Assistant App

Build and Deploy with Flowise AI, Text-to-SQL and FastAPI Server

Implementation Guide

Codes • Configurations • GitHub Repo • Resources

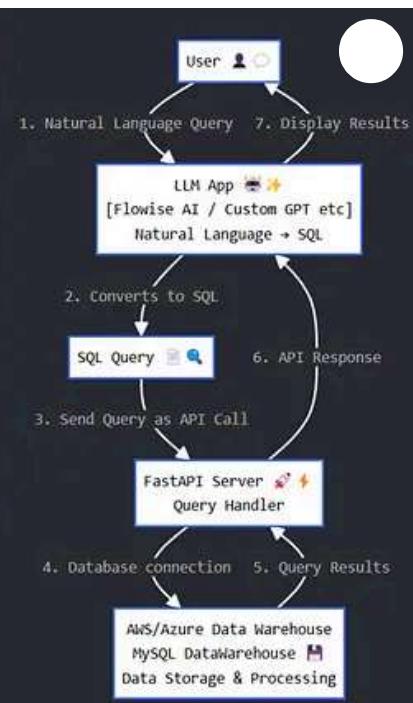


FastAPI

Microsoft Azure



Amar Harolikar | Specialist - Decision Sciences & Generative AI



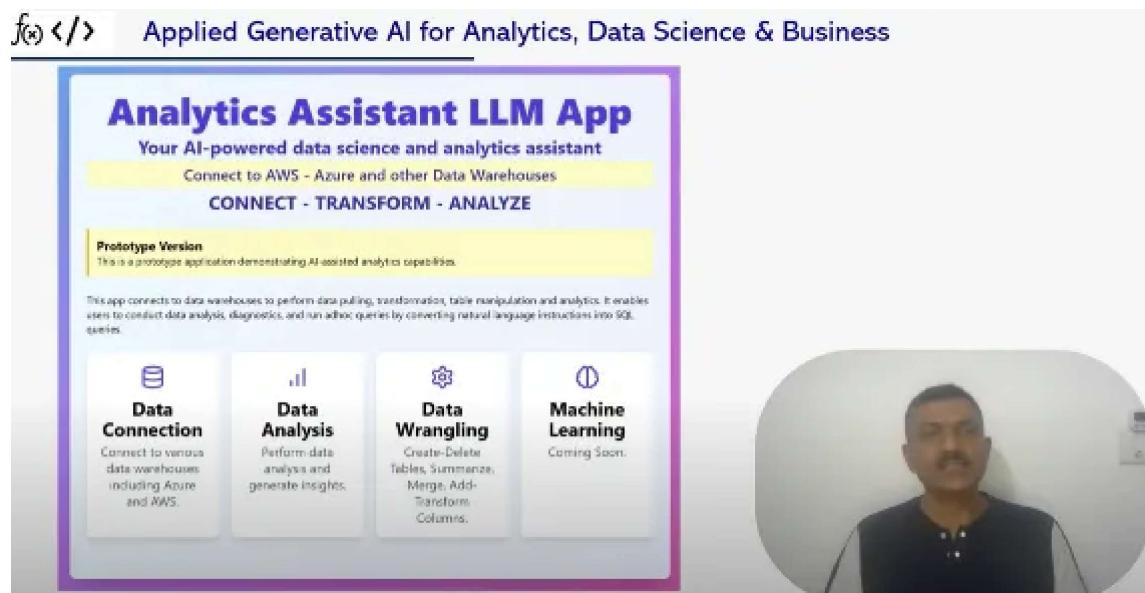
Video : Supplementary to Guide

In my earlier videos and posts, I showed how the Analytics Assistant App works... connecting to and analyzing data on AWS RDS MySQL and Azure MySQL, with table sizes ranging from 1 million to 10 million records... smoothly converting natural language text to SQL... summarizing, merging, and creating customer profiles.

In this article I am sharing a step-by-step implementation guide. Here's Friend Link to the article for free access [*Build an Analytics Assistant App*](#)
[*Friends Link*](#)

Links Below demoing how the Analytics Assistant LLM App works.

MEDIUM: *LLM Analytics Assistant: Simplifying Data Transformation & Insights. AWS & Azure MySQL.*



[Analytics Assistant LLM App : Connect to AWS — Azure. Transform & Analyze Data. — YouTube](#)

LINKEDIN : *GenAI App | LLM Analytics Assistant: Simplifying Data Transformation & Insights. AWS & Azure MySQL DW Example*

This is similar to a setup I am currently using for a live client use case. This is a barebones version for sharing publicly.

I used a FastAPI server in the middle to allow ease of connectivity to any backend database as well as the front end of our choice.

I've deployed this setup on my public website (tigzig.com) using Flowise AI, and I also have a custom GPT version. In both cases, my FastAPI server and backend databases are the same.

What we will cover in this implementation guide

1. Example of Text-to-SQL
2. Implementation: Step 1 — FastAPI Server — Code
3. Implementation: Step 2 — FastAPI Server — Deploy
4. Implementation: Step 3 — Flowise AI : Setup Custom Tool
5. Implementation: Step 4 : Flowise AI : Setup Chatflow
6. Implementation: Step 5 : Flowise AI : Deploy on your Website
7. Data Warehouse — Azure — AWS — Others
8. Additional Consideration: Security, Monitoring, Tracing, Feedback
9. Cost Considerations
10. Full Data Ingestion & Split Processing: Scenarios and Costs
11. Agentic Framework, Accuracy & Complex Workflows
12. Working with LLM / GPTs for coding: My Top 6 Favorite Techniques
13. Resources

Examples: Text-to-SQL and Analysis

In the query below, I asked the assistant to summarize the 10 million transactions table with a bunch of variables... in natural language, complete with spellings and sometimes a bit fuzzy instructions. Check out the query generated at the backend and passed to the database. It is absolutely perfect. The LLM Model used here was GPT-3.5-Turbo (it was a few weeks back before gpt-4o-mini was released)

```

the trans 10m table
can you summarize
put into tnew table temp_trans_summ
group by custid
tc code 1001 is cash sales
and tc 1002 is retail sales
need CASH_SALES_AMT
CASH_SALES_NUM
and similarly for retail sales also
and then also another field CASH_SALES_TAG
make it as 1 if CASH_SALES_NUM >= 1
similarly add retail sales tag
add total sales amount and num which is sum of cash sales and retail sales amount and nums
also want to add the average transaction value for retail cash and total
for example CASH_SALES_ATV - cash sales amount / cash sales num
|
```

sql

```

CREATE TABLE temp_trans_summ AS
SELECT
    CUSTID,
    SUM(CASE WHEN TC_CODE = 1001 THEN AMOUNT ELSE 0 END) AS CASH_SALES_AMT,
    COUNT(CASE WHEN TC_CODE = 1001 THEN 1 END) AS CASH_SALES_NUM,
    SUM(CASE WHEN TC_CODE = 1002 THEN AMOUNT ELSE 0 END) AS RETAIL_SALES_AMT,
    COUNT(CASE WHEN TC_CODE = 1002 THEN 1 END) AS RETAIL_SALES_NUM,
    CASE WHEN COUNT(CASE WHEN TC_CODE = 1001 THEN 1 END) >= 1 THEN 1 ELSE 0 END AS CASH_SA
    CASE WHEN COUNT(CASE WHEN TC_CODE = 1002 THEN 1 END) >= 1 THEN 1 ELSE 0 END AS RETAIL_
    SUM(AMOUNT) AS TOTAL_SALES_AMT,
    COUNT(1) AS TOTAL_SALES_NUM,
    SUM(CASE WHEN TC_CODE = 1001 THEN AMOUNT ELSE 0 END) / COUNT(CASE WHEN TC_CODE = 1001
    SUM(CASE WHEN TC_CODE = 1002 THEN AMOUNT ELSE 0 END) / COUNT(CASE WHEN TC_CODE = 1002
    SUM(AMOUNT) / COUNT(1) AS TOTAL_SALES_ATV
FROM TRANS_10M
GROUP BY CUSTID;
```

Text to SQL Example: Summarizing transaction table with 10 Million Records with addition of derived variables

I then merged the transaction summary with the 1 million customer table.
Now, I'm asking the assistant to create a customer sales activity profile with additional derived variables. This one is grouped by the housing variable.

```

now from this bank merged table
create a customer sales activity profile. show me the following, share in table format
group by housing variable
show in nice vertical column format, with metrics headers in rows and the group by variable in column. give proper header
total customers : count of records
Average Per Customer
total active customers : num of customers total sales amt > 0
cash active customers : num of customers with cash sales amt > 0
retail active customers : num of customers with retail sales amt > 0

Averages per active customer
average sales amount per customer : total sales amount / total active customers
average retail sales amt per retail active customer : retail sales amt / num of retail active customers per above
average cash sales amt per cash active customer : cash sales amt / num of cash active customers per above

Average Transaction values
Total ATV : sum of total sales amount / sum of total sales num
Cash ATV : cash sales amount / sum of cash sales num
Retail ATV : retail sales amount / sum of retail sales num
do it carefully. make sure query is mysql compliant
don't use average function. compute numerator and denominator separately
|
```

sql

```

SELECT
    HOUSING,
    COUNT(*) AS total_customers,
    SUM(CASE WHEN TOTAL_SALES_AMT > 0 THEN 1 ELSE 0 END) AS total_active_customers,
    SUM(CASE WHEN CASH_SALES_AMT > 0 THEN 1 ELSE 0 END) AS cash_active_customers,
    SUM(CASE WHEN RETAIL_SALES_AMT > 0 THEN 1 ELSE 0 END) AS retail_active_customers,
    SUM(TOTAL_SALES_AMT) AS total_sales_amount,
    SUM(RETAIL_SALES_AMT) AS total_retail_sales_amount,
    SUM(CASH_SALES_AMT) AS total_cash_sales_amount,
    SUM(TOTAL_SALES_NUM) AS total_sales_num,
    SUM(CASH_SALES_NUM) AS total_cash_sales_num,
    SUM(RETAIL_SALES_NUM) AS total_retail_sales_num,
    SUM(TOTAL_SALES_AMT) / SUM(CASE WHEN TOTAL_SALES_AMT > 0 THEN 1 ELSE 0 END) AS avg_sales,
    SUM(RETAIL_SALES_AMT) / SUM(CASE WHEN RETAIL_SALES_AMT > 0 THEN 1 ELSE 0 END) AS avg_retail_sales,
    SUM(CASH_SALES_AMT) / SUM(CASE WHEN CASH_SALES_AMT > 0 THEN 1 ELSE 0 END) AS avg_cash_sales,
    SUM(TOTAL_SALES_AMT) / SUM(TOTAL_SALES_NUM) AS total_atv,
    SUM(CASH_SALES_AMT) / SUM(CASH_SALES_NUM) AS cash_atv,
    SUM(RETAIL_SALES_AMT) / SUM(RETAIL_SALES_NUM) AS retail_atv
FROM
    BANK_1M_MERGED
```

Text To SQL Example: Creating a Customer Sales Activity Profile Report with additional derived variables

Finally, I ask the assistant to create a consolidated customer sales activity profile combining the individual ones generated earlier (grouped by housing/ response tag/ education variables)

can you add a total column

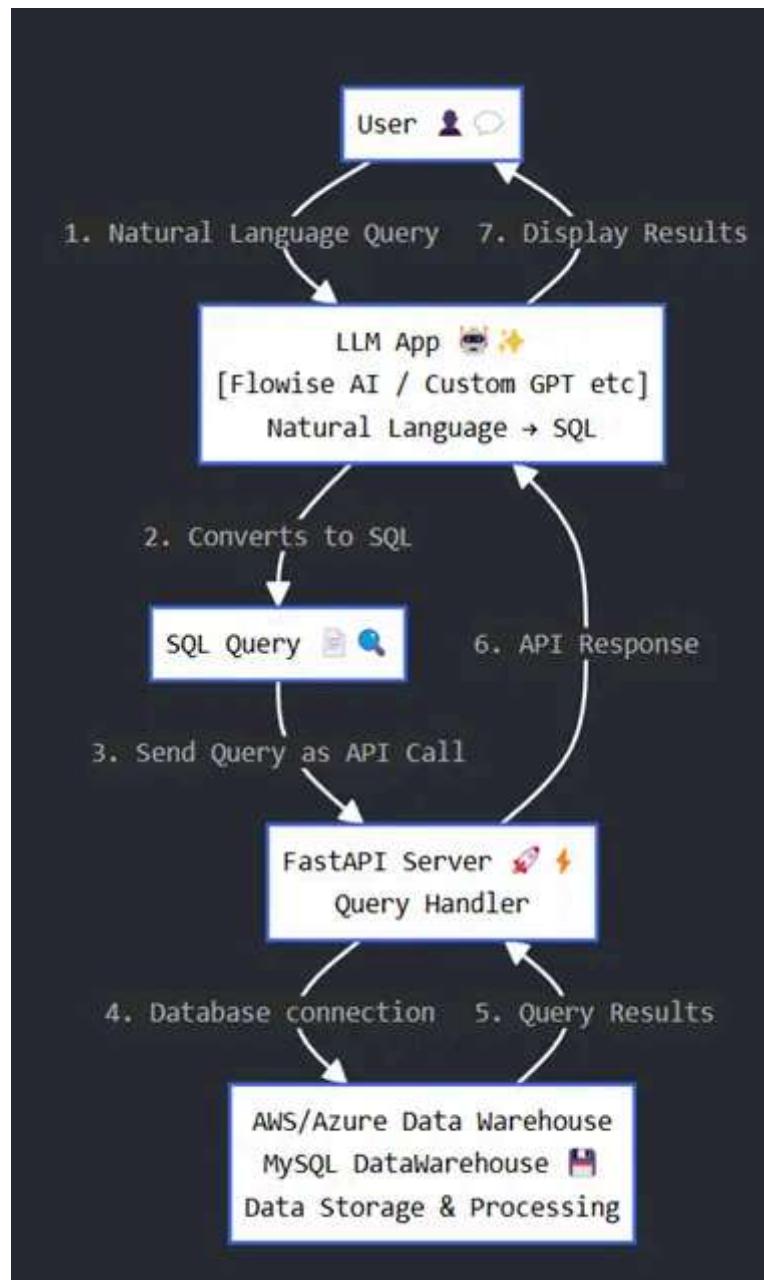
Here is the combined customer sales activity profile for the response variable, housing variable, and education variable with a total column included in a single table format:

Metric	Response No	Response Yes	Housing Yes	Housing No	Education Secondary	Education Primary	Education Unknown	Education Tertiary	Total
Total Customers	964,704	35,296	834,273	163,327	567,156	188,768	46,194	196,182	1,000,000
Total Active Customers	289,445	10,555	251,571	47,733	170,401	56,467	13,790	58,849	300,000
Cash Active Customers	53,335	1,959	47,209	7,990	31,798	9,998	2,406	11,025	55,294
Retail Active Customers	279,850	10,204	242,589	46,769	164,423	54,939	13,440	56,759	289,358
Avg Sales Amt per	32,940.54	32,904.82	32,854.70	33,367.68	32,898.98	33,133.39	32,931.14	32,860.41	32,939.28
Avg Retail Sales Amt per Retail Active Customer	33,650.45	33,756.09	33,680.30	33,700.14	33,450.48	33,700.14	33,650.14	33,650.14	33,668.28
Avg Cash Sales Amt per Cash Active Customer	2,161.89	2,093.64	2,144.88	1,752.32	1,949.51	1,949.51	1,889.48	2,163.63	2,100.85
Total ATV	964.90	965.44	8,584.35	9,756.50	963.47	970.09	971.21	962.44	964.92
Cash ATV	202.09	202.49	504.86	201.69	202.15	194.91	188.95	216.63	202.10
Retail ATV	1,010.26	1,010.96	807.95	975.50	1,010.33	1,010.63	1,010.12	1,009.81	1,010.28

If you have any more queries or need further analysis, feel free to ask!

Analytics Assistant LLM App: How it works

Shared in detail in my initial video. Summary below.



- 1. Natural Language Query:** User inputs a natural language query... this could be done through Flowise AI, Custom GPT, or other LLM platforms... or any other front end of your choice
- 2. Text-to- SQL :**The LLM App (like Flowise AI or Custom GPT) takes this natural language input and converts it to SQL... this is where the magic happens... natural language to SQL conversion.
- 3. Send Query as API Call:** The SQL query is then sent as an API call to the FastAPI server... this uses Custom Tool/ Function Calling capabilities of

an LLM Model

4. **Database Connection:** FastAPI server connects to the AWS or Azure Data Warehouse... this server is the middleman, ensuring data is retrieved securely and efficiently...and sent back to whoever is making the API call.
5. **Query Results:** The database processes the SQL query and sends back the results to the FastAPI server...
6. **API Response:** FastAPI server receives these results and sends them back to the LLM App...
7. **Display Results:** The LLM App takes this data and servers it to the user...

How to Guide | Step 1—FastAPI Server : Coding

All coding done by LLMs, in this case by ChatGPT-4o. And by the way, some of my earlier FastAPI servers were coded by the free ChatGPT 3.5. And it was also incredibly good.

I found that the easiest way is to share one of my existing FastAPI server codes and instruct the LLM to revise it with my new instructions on database server connections, endpoint setup, output sharing, and so on.

A FastAPI server allows us to create an API endpoint with Python code. [See resources section for links to articles and videos explaining the basics of FastAPI servers]

Complete code is available on my GitHub repo. You can pick it up from there and make any necessary changes.

[amararun/aws-azure-llm-assistant: LLM Analytics Assistant. Connected to AWS and Azure MySQL DataWarehouse \(github.com\)](#)

In the sections below, I am sharing explanations of various parts of the code. In case of questions or modifications, the easiest way is to ask your favorite LLM to guide you. That's how I started off... and continue to do so whenever I get a code that I need to unpack and understand.

In the first section of code below, I am importing libraries, setting up the FastAPI instance, and defining the database connection details.

For my live app on the client site, the database details are stored as environment variables. Here, I've put them in the main code for prototype purposes only.

The `get_connection` function chooses the right database connection (AWS or Azure) based on the parameters passed by the LLM Custom Tool/Function. I have also enabled slightly more detailed logging to trace the complete flow in my server logs. This is very useful to figure out where an issue might be happening.

```
from fastapi import FastAPI, HTTPException, Request, Depends
from fastapi.responses import FileResponse
import mysql.connector
import os
import tempfile
import logging

# Configure logging
logging.basicConfig(level=logging.DEBUG, format"%(asctime)s - %(name)s - %(level
logger = logging.getLogger(__name__)

app = FastAPI()

# Database connection details for AWS
aws_database_name = "example_schema"
aws_host = "db-instance-example123.us-east-1.rds.amazonaws.com"
```

```

aws_user = "admin_example"
aws_password = "awsExamplePassword!456"

# Database connection details for Azure
azure_database_name = "example_db"
azure_host = "example-server3.mysql.database.azure.com"
azure_user = "example_admin"
azure_password = "azureExamplePassword!789"

def create_aws_connection():
    connection = mysql.connector.connect(
        host=aws_host,
        user=aws_user,
        password=aws_password,
        database=aws_database_name
    )
    return connection

def create_azure_connection():
    connection = mysql.connector.connect(
        host=azure_host,
        user=azure_user,
        password=azure_password,
        database=azure_database_name,
        ssl_disabled=True
    )
    return connection

def get_connection(cloud: str):
    if cloud == "azure":
        return create_azure_connection()
    elif cloud == "aws":
        return create_aws_connection()
    else:
        raise ValueError("Invalid cloud provider specified")

```

In the code below, I define an endpoint /sqlquery/ that takes two string parameters: a full SQL query (passed by the LLM Assistant/Tool) and the cloud provider (AWS/Azure).

The code executes the SQL query. If it's a SELECT query, it fetches the results, writes them to a temporary text file, and returns the file to the user.

For non-SELECT queries, it commits the changes to the database and returns a success message.

```
@app.get("/sqlquery/")
async def sqlquery(sqlquery: str, cloud: str, request: Request):
    logger.debug(f"Received API call: {request.url} with cloud parameter: {cloud}")
    try:
        connection = get_connection(cloud)
    except ValueError as e:
        raise HTTPException(status_code=400, detail=str(e))

    try:
        cursor = connection.cursor()
        cursor.execute(sqlquery)

        # Handle queries that return results
        if cursor.description is not None:
            headers = [i[0] for i in cursor.description]
            results = cursor.fetchall()
            cursor.close()

            # Create a temporary file
            with tempfile.NamedTemporaryFile(delete=False, mode='w', suffix=".txt") as temp_file:
                temp_file.write(" | ".join(headers) + "\n")
                for row in results:
                    temp_file.write(" | ".join(str(item) for item in row) + "\n")
            temp_file_path = temp_file.name

            logger.debug(f"Query executed successfully, results written to {temp_file_path}")
            # Return the file response
            response = FileResponse(path=temp_file_path, filename="output.txt",
                                    media_type="text/plain")
            return response

        # Handle non-SELECT queries
    else:
        connection.commit()
        cursor.close()
        logger.debug("Non-SELECT query executed successfully")
        return {"status": "Query executed successfully"}

    finally:
```

```
connection.close()
logger.debug("Database connection closed")
```

In this last section of the code, I have added a middleware that processes HTTP requests... it makes sure that any temporary file created is removed after the response is sent.

The `if __name__ == "__main__"` block runs the FastAPI app with Uvicorn, listening on all network interfaces at port 8000, with a timeout of 120 seconds.

```
@app.middleware("http")
async def remove_temp_file(request, call_next):
    logger.debug(f"Processing request: {request.url}")
    response = await call_next(request)
    if isinstance(response, FileResponse) and os.path.exists(response.path):
        try:
            os.remove(response.path)
            logger.debug(f"Temporary file {response.path} removed successfully")
        except Exception as e:
            logger.error(f"Error removing temp file: {e}")
    return response

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000, timeout=120)
```

The only other piece needed is the requirements.txt file.

```
fastapi==0.70.0
uvicorn==0.15.0
```

```
mysql-connector-python  
python-multipart
```

Test the code. Run this from your terminal.

```
python3 -m uvicorn app:app --reload
```

How To Guide | Step 2 — FastAPI Server: Deploy

You can deploy on any web service platform of your choice. I find Render to be the easiest to deploy. Just takes a few minutes. I also use Heroku for some of my apps. Both are reasonably priced. Render has a great free tier.

If you need always-on, the Starter pack costs USD 7 per month, and you are charged for the time you keep the server live. If you suspend the web service, you are not charged for that part.

See videos and guides in the resources section on how to deploy a FastAPI server/web service on Render. Quick steps below:

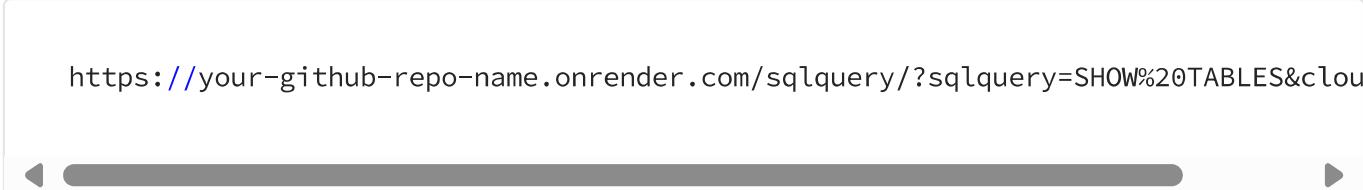
1. In the ‘New’ Button on top, choose ‘Web Service’.
2. Link and point to your GitHub Repo.
3. **Language:** Python 3
4. **Start Command:** `uvicorn app:app --host 0.0.0.0 --port $PORT`

Choose the Starter plan if you wish, and hit ‘Deploy Web Service’ and you are all set. It takes a minute or two to deploy

Once deployed, you will get a URL like this: `https://your-github-repo-name.onrender.com/`. This is the URL of your FastAPI Server, always listening for incoming requests.

Use this URL in the next step to link up Flowise AI to your FastAPI Server.

You can test it using a service like Postman (see links in the resources section on how to use Postman). Another way is to copy the full API Call URL and paste it into your web browser. An example with SHOW TABLES for AWS is shown below:



The screenshot shows a browser window with the URL `https://your-github-repo-name.onrender.com/sqlquery/?sqlquery=SHOW%20TABLES&clou` in the address bar. Below the address bar is a horizontal scrollbar with arrows at both ends. The rest of the page content is not visible.

If all is well, your browser window will display the results of the query (file download box in my case as I am outputting a file).

In case of errors, share the error back with LLM/GPTs to debug. Postman is super helpful in these situations.

How To Guide | Step 3 — Flowise AI: Setup Custom Tool

Leon Van Zyl has the absolutely marvelous step-by-step guides for Flowise AI. Leon’s complete Flowise playlist is a must watch for anybody wanting to

develop an app on Flowise.

Flowise AI Tutorial #11- Custom Tools, OpenAI Functions and Webhooks
(youtube.com)

I would recommend using Leon's videos as guide to setup your Custom Tool as well as to setup ChatFlows. Below I am sharing the key details required for the custom tool setup as well as some tips.

[Edit Tool](#) [Export](#)TOOL NAME * [?](#)

T_AZURE_AWS_MYSQL

TOOL DESCRIPTION * [?](#)

Execute API call to a based on user question . User question would be converted into a SQL Query and passed as a parameter. Another paramer is the cloud database provide ie azure or aws. See sample API endpoint URL call below.

Database server in both cases is MySQL. So make sure that queries are MySQL compliant

TOOL ICON SOURCE

<https://raw.githubusercontent.com/gilbarbara/logos/main/logos/airtable.svg>INPUT SCHEMA [?](#)[+ Add Item](#)

Property	Type	Description	Required	
sqlquery	string	mysql sql query string based on user q...	✓	
cloudVar	string	either 'aws' or 'azure' based on user que...	✓	

1 row selected Rows per page: 100 ▾ 1-2 of 2 < >

JAVASCRIPT FUNCTION [?](#)[How To Use Function](#)[See Example](#)

```
1 v /*  
2 NOTE FOR TOOL AGENT : Please wait for upto 180 seconds for query response to come in as the server  
can be slow sometimes  
3 */
```

Screenshot of my Custom Tool on Flowise AI

Tool Description that I am using shared below.

Replace with your actual endpoint URL per section above and your actual schema names

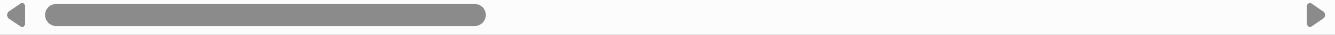
The description helps the Tool Agent to figure out when and how to use the tool.

Execute API call to a based on user question . User question would be converted Database server in both cases is MySQL. So make sure that queries are MySQL compatible
If user refers to 'schema' or 'database' it refers to the following
in Azure : give-your-schema-name-here
in AWS : give-your-schema-name-here

All the rest of the schema/databases on the MySQL server are system ones and not

Here's a API end point example for quick reference. Just sample. use all correct
<https://give-your-api-endpoint-url-here.com/sqlquery?sqlquery=SHOW%20TABLES%20FR>

Please wait for upto 180 seconds for query response to come in as the server can



Sharing below the input schema that I am using

This instructs the tool agent that the SQL query string has to stored into variable called 'sqlquery' and the data warehouse (aws or azure) specified by the user can take values either 'aws' or 'azure' and needs to be stored into variable called cloudVar. This is used while making the http fetch request to FastAPI endpoint.

INPUT SCHEMA				
Property	Type	Description	Required	
sqlquery	string	mysql sql query string based on user question	<input checked="" type="checkbox"/>	
cloudVar	string	either 'aws' or 'azure' based on user question	<input checked="" type="checkbox"/>	

JavaScript Function that I am using shared below.

Replace with your actual FastAPI endpoint URL after deploying on the web server as per the earlier section. For any modifications and changes, I tend to just share an existing JavaScript function like the one below and instruct LLM/GPT to make necessary changes.

The JavaScript executes an HTTP fetch request using the 'GET' method to the API endpoint URL that we created. As you can see in the code, the full URL is constructed using the 'sqlquery' and 'cloudVar' variables stored earlier in the input schema, in the exact format that the FastAPI server is expecting.

```
/*
NOTE FOR TOOL AGENT : Please wait for upto 180 seconds for query response to co
*/

const fetch = require('node-fetch');

// Properly encode the query parameters
const sqlQuery = encodeURIComponent($sqlquery);
const cloudVar = encodeURIComponent($cloudVar);

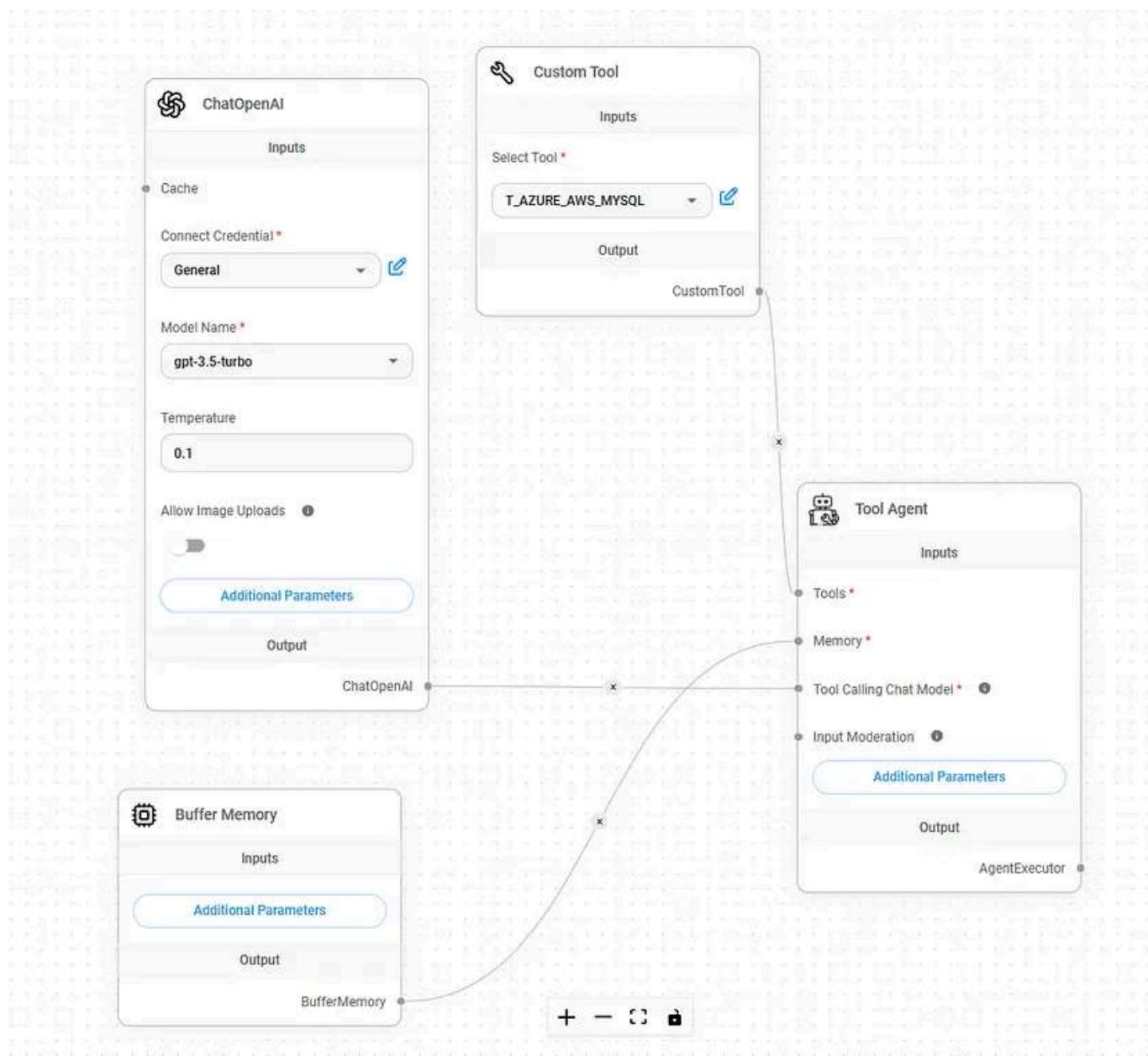
const url = `https://your-api-endpoint-url-here.com/sqlquery/?sqlquery=${sqlQuer

const options = {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json'
  }
};

try {
  const response = await fetch(url, options);
  const text = await response.text();
  return text;
} catch (error) {
  console.error(error);
  return '';
}
```

How To Guide | Step 5 — Flowise AI: Setup Chatflow

Sharing below the image of my ChatFlow. It's a pretty straightforward setup to replicate. I was using GPT-3.5-Turbo at that time, as GPT-4o-mini was yet to be released. I've changed it now. I also have a version with Claude and Google Gemini, but at that time, the OpenAI one seemed to work the smoothest.



How To Guide | Step 6 — Flowise AI: Deploy on your Website

I simply use the embed URL to embed it into my website. There are multiple options available

- Click on Embed Code Icon <> on top of ChatFlow Window
- Go to Share Chatbot -> Enable Public Sharing -> Customize Options
- Use the embed URL link to embed on public website / intranet site.

Refer to Leo's Video for a step-by-step guide and various options for customization.

[*Adding AI Chatbots to Websites — FlowiseAI Tutorial #10 \(youtube.com\)*](#)

This is how it looks on my website

f(x) </> Applied Gen AI for Analytics, Data Science & Business TIGZIG

Amar Harolikar | Specialist - Decision Sciences & Generative AI

ANALYTICS ASSISTANT : CONNECTED TO AWS AND AZURE MYSQL WAREHOUSE

ANALYTICS ASSISTANT : CONNECTED TO AWS AND AZURE MYSQL DATAWAREHOUSE LLM MODEL: gpt-3.5-turbo-0125

Retail Active Customer

Avg Cash Sales Amt per Cash Active Customer	2,161.89	2,093.64	2,144.88	1,752.32	1,949.51	1,949.51	1,889.48	2,163.63	2,100.85
Total ATV	964.90	965.44	8,594.35	9,756.50	963.47	970.09	971.21	962.44	964.92
Cash ATV	202.09	202.49	504.86	201.69	202.15	194.91	188.95	216.63	202.10
Retail ATV	1,010.26	1,010.96	807.95	975.50	1,010.33	1,010.63	1,010.12	1,009.81	1,010.28

If you have any more queries or need further analysis, feel free to ask!



hi

Hello! How can I assist you today?



Type your question



Data Warehouse — Azure — AWS — Others

Using FastAPI as a middle processing layer , you can connect to any Data Warehouse/Databases as well as different front ends and LLM Apps.

You just need the right connection parameters, typically provided by your database admin. If you want to set up one from scratch, here are some quick tips.

My existing client use case requires me to work with data warehouse schemas on AWS RDS MySQL and also Azure MySQL. For the purposes of the prototype, I have set up both instances.

For those interested in setting up AWS/Azure MySQL instances, here are some quick tips:

- Numerous videos and guides available. See the resources section below for a few.
- Request LLM/GPT to guide you step by step. Wherever you get stuck, share a screenshot of the AWS/Azure screen and let the LLM/GPT guide you.
- For the very first connection test, it's best to do it in Google Colab/Jupyter to make sure the connection works. You can use something like this:

```
import mysql.connector

# Replace placeholders with your credentials
database_name = "example_schema"
host = "db-instance-example123.us-east-1.rds.amazonaws.com"
user = "admin_example"
password = "awsExamplePassword!456"
def create_connection():
    connection = mysql.connector.connect(
        host=host,
        user=user,
        password=password,
        database=database_name
    )
    return connection

connection = create_connection()
connection
```

- For UI and admin access to the data warehouses, I prefer to use phpMyAdmin. It's best to install it via the Dockerfile route rather than installing all the dependencies one-by-one. Check the video links in the

resources section. Just ask LLM/GPT to guide you step-by-step on how to do it. Make sure to set up a user and password there.

- Later on, once your FastAPI server is working, modify the security settings in both AWS and Azure to allow inbound access only from the IP addresses of your FastAPI server. In my case, the FastAPI server was deployed on Render, and Render provides these IP addresses, as do all other web service providers.

Additional Consideration: Security , Monitoring, Tracing, Feedback

Briefly touching upon some considerations. Plan to cover those in more detail in future posts.

Tracing Message Flow: Flowise AI allows multiple options to trace the complete message flow. Especially useful in debugging complex flows. I use Langsmith. Very easy to set up and use. See Leon's tutorial ([link in resources section](#)).

Message Feedback: Very easy to enable message feedback in Flowise. Users can give a thumbs up or a thumbs down with a message box for further inputs. See Leon's tutorial ([link in resources section](#)).

Monitoring Chats: Chat histories can be monitored directly by going into Settings in the Chat Flow window -> View Messages. Chat history is by default in an SQLite database in the default database path specified during Flowise AI setup. Flowise AI also supports storing messages in MySQL or

PostgreSQL. Excellent video from business_24.ai on how to export chat histories and related aspects (link in resources section).

Database Credentials: I store them in environment variables. Just for prototype purposes, I've put them directly in the code for ease of explanation. Recommended to keep them as environment variables.

API Keys: You can also set up your own API keys to control access to the FastAPI endpoint. I have another version for this and will share it in future posts. It's fairly simple to do. The simplest implementation requires setting up a key as an environment variable and getting LLM/GPT to modify the FastAPI server code and custom tool JavaScript code accordingly. This passes an additional variable in the input schema of the custom tool. It can also be used for user access control by having the LLM Assistant request an API key before proceeding, which is then passed as a query parameter down the line.

Inbound IP Address: I have set up an inbound IP address whitelist on both Azure and AWS to only allow incoming requests from the Render server. This is to guard against any unauthorized API calls from unknown IP addresses.

Selective Access: For instance, to certain tables/schemas as well as read/write access. Can be enabled via phpMyAdmin and integrated downstream.

Prompt Injections: Whole big area. Lots of options available. Was not a high priority for my client use case, so I have not gone much into this.

Standard IT Security: These include enterprise login, SSO, other user authentications, server-side measures to guard against DDoS attacks,

unauthorized server access, control over public IP addresses, setting up a VPN if required, etc., which the IT world has been doing for ages.

Cost Considerations for LLM Analytics App

Shared in my initial video. Sharing below a quick overview of expenses associated with the LLM app platform, LLM model usage and FastAPI server. There can be a wide variance here based on the particular use case.

LLM App Platform Costs

Flowise AI:

- The LLM app builder I'm using is Flowise AI, which is open-source and free. However, you still need to host it somewhere.
- For this prototype, I've hosted it on Render, which costs me around \$7 per month. And only charged pro-rata for the time the server is turned on. You can host it pretty much on any platform of your choice including AWS / Azure / GCP / Railway / Heroku / Digital Ocean etc.

Scalability:

- This setup is less intensive, suitable for small-scale use cases.
- If you need to handle hundreds or tens of thousands of messages per minute, the \$7 server won't suffice, and you'll need a more robust solution. Either on Render or elsewhere

LLM Model Costs

GPT-3.5:

- For this prototype, on a particular day I ran approximately 470 API requests over 10 hours, costing around a total of just \$1. This involved working with a 1 million customer dataset and a 10 million transaction dataset, creating synthetic data, merging, summarizing, and generating cross tabs.
- The total token usage was about 1.9 million, with 1.8 million input tokens and around 52,000 output tokens.
- This cost was for a non-optimized setup. Proper optimization can significantly reduce the costs.

GPT-4o-mini:

- This was before GPT-4o-mini was launched. If I were to use gpt-4o-mini now the cost would be around half of that.

Optimization:

- High input token count is due to sending the entire history with every query. This can be optimized to reduce costs.
- Output tokens averaged around 110 per query, representing a reasonable size for SQL queries or cross tab results.

FastAPI Server Costs

Hosting:

- The FastAPI server, which acts as an API endpoint, is also hosted on Render for \$7 per month. Again charged pro-rata for the time the server is turned on.

- This server only charges when it is up and running, making it a cost-effective solution for small-scale use cases.

Server Load:

- For this prototype, the FastAPI server is primarily acting as a messenger, handling queries and responses without any heavy processing.
- If the server were to handle processing tasks like data cleaning, transformation, statistical procedures, or model building, a higher capacity server would be required.

Full Data Ingestion: Scenario and Costs

- This is a very different scenario than what we have discussed above. For example, building a machine learning model with OpenAI's code interpreter requires ingesting the entire dataset, leading to exponentially high API costs.
- Full data ingestion by the LLM model is not always necessary and can significantly increase costs, potentially by 1000X or more

For many use cases, processing can be done separately, and the LLM only needs to pass SQL queries or Python commands, avoiding the need for full data ingestion.

This is the use case presented in this article, where processing is happening at the database server end. In another of my use cases part of processing is happening at database server end and part of it on the FastAPI server.

Split Workflow Approach:

- For scenarios requiring full data ingestion, split the workflow into multiple modules. The LLM should ingest only the necessary and smallest amount of data directly, while the remaining data processing and model building should happen separately on the database server and/ or Python server.
- Planning to share a prototype with this approach in near future.

Agentic Framework, Accuracy & Complex Workflows

This was a simple prototype. Even with GPT-3.5, I got fairly good accuracy on SQL queries. Less than 10% of the more complex SQL queries were wrong. It was easy for me to catch, but that might not work for many scenarios. I'm sure that using GPT-4o / 4o-mini or Claude-3.5-Sonnet would significantly improve query accuracy.

For further improvements, an agentic framework is strongly recommended, with a ‘Supervisor’ or ‘Query Checker’ agent checking the query prepared by the initial ‘Query Preparer’ agent. Flowise AI supports this. Check out Leon’s excellent guide on the same.:

[How to Build AI Agent Teams \(NO-CODE!\) | Flowise Agentflows Tutorial \(youtube.com\)](#)

And Flowise AI recently launched Sequential Agents, a way more powerful framework to incorporate complex flows including : calling agents conditionally, run agents in parallel, human-in-the-loop , state management and much more. See Leon’s brilliant video guide below:

Working with LLM / GPTs for coding: My Top 6 Favorite Techniques

1. **Voice Typing:** Some of my instructions for code generation or modification go into pages. I tend to use the Voice Typing mode in Google Docs. I simply ‘talk’ whatever I need and dump the whole thing onto LLM/GPT .
2. **Video Screen Grab:** Example: In one of the videos, the contributor shared instructions and codes on screen. I took a screen grab and asked LLM/GPT to guide me through that step-by-step.
3. **Vision-Based Guidance:** Example: When filling out forms in Azure/AWS and getting confused about options or not knowing where to find something specific, I just share a screenshot with GPT and ask it to guide me. It will tell me something like, “On the bottom window, there is an icon that looks like so-and-so... click there... then do so-and-so” or “For your prototype needs, you need not choose so-and-so option but so-and-so option will work.”
4. **“I want to do the same thing”:** Example: Sometimes I take an article like this one and dump it on an LLM/GPT and tell them that I want to do the same thing. It can then explain in simple words what is happening and further guide me step-by-step.
5. **Video Transcript:** Example : For a particularly long and complex video tutorial, I pulled out the transcript and shared with LLM / GPT ... to explain in simple words .. and then guide me step by step. See my article

[How to summarize & analyze YouTube videos with AI : Two FREE and EASY options](#) | by Amar Harolikar | Medium.

6. “Use this. Change that” : Example: When I want to get a new FastAPI server coded, I don’t start from scratch. I share a previous working code and the changes required. Same with input schemas, JavaScript connector codes, etc. Otherwise, it starts coding from scratch and I need to debug the same errors that came the first time around.

Additional Resources

Numerous posts and videos available. Very grateful to all the contributors. Sharing just a few for quick reference. And anywhere you get stuck don’t hesitate to ask LLM/ GPT to guide you step by step. In fact I will very often share a screen grab from a video and request LLM to guide me for that step-by-step.

FastAPI

- [FastAPI Setup a Server in 10 Minutes | Installation, Setup & Route Creation \(youtube.com\)](#).
- [FastAPI Tutorial: An Introduction to Using FastAPI | DataCamp](#)
- [Using FastAPI to Build Python Web APIs — Real Python](#)
- [How to deploy a FastAPI API or APP on Render for free \(youtube.com\)](#).

Postman

- [What is Postman and why everyone use it ? \(youtube.com\)](#).
- [Learn Postman in 15 Minutes \(youtube.com\)](#).

Setting up AWS and Azure MySQL and phpMyadmin

- [Create AWS RDS and Connect Through PhpMyadmin | Connect RDS Through CLI \(youtube.com\)](#)
- [Create Azure Database for MySQL Flexible Server using Azure Portal | How to Create MySQL Flex Server \(youtube.com\)](#)
- [phpMyAdmin Tutorial: Introduction to phpMyAdmin | What is phpMyAdmin? \(youtube.com\)](#)

Monitoring , Tracing, Message Feedback

- [Analysing Chatflows using LangSmith – FlowiseAI Tutorial #7 \(youtube.com\)](#)
- [\(476\) 8 SECRET Flowise AI Features You DIDN'T Know Existed \(Supercharge Your Workflow!\) – YouTube \(for message feedback and other features\)](#)
- [\(476\) Flowise Database – Use MySQL and SQLite to export Flowise Messages to Excel for FREE – YouTube](#)
- [\(476\) Flowise Database – Use MySQL and SQLite to export Flowise Messages to Excel for FREE – YouTube](#)

Data Science

Artificial Intelligence

Analytics

Machine Learning

Marketing



Written by Amar Harolikar

[Edit profile](#)

38 Followers

Decision Sciences Specialist

More from Amar Harolikar

BUILD VOICE BASED AI ACTION AGENT APP
Execute Tasks, Automate Reports and Analyze Data in Live DW

Amar Harolikar

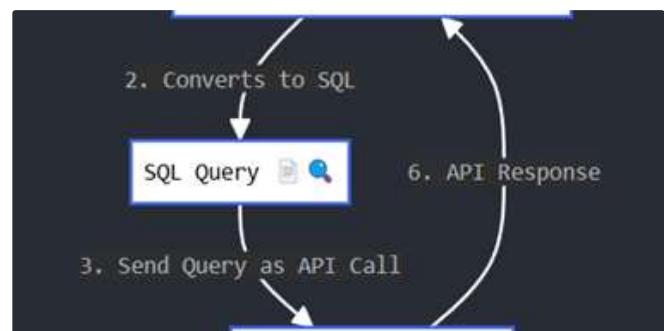
How to build Voice based AI Action Agents App to Execute Tasks,...

In this article I show how to build a voice-powered AI agentic app to automate busine...

Sep 16 50 1



...



Amar Harolikar

Build Analytics Assistant Custom GPT: Connect to live Data...

In this post and video, I demonstrate the capabilities of a Custom GPT connected to a...

Aug 18 10



...

CONNECT TO AWS - Azure and Other Data Warehouses

CONNECT - TRANSFORM - ANALYZE

Prototype Version
This is a prototype application demonstrating AI-assisted analytics capabilities.

This app connects to data warehouses to perform data pulling, transformation, table manipulation and analytics. It enables users to conduct data analysis, diagnostics, and run adhoc queries by converting natural language instructions into SQL queries.



Data Connection

Connect to various data warehouses



Data Analysis

Perform data analysis and



Data Wrangling

Create-Delete Tables, Summarize,



Machine Learning

Coming Soon.

 Amar Harolikar

LLM Analytics Assistant: Simplifying Data...

3rd part of the series on LLM Analytics Assistant Apps

Jul 12



...

Mutual Fund Analytics with ChatGPT Prompts

Merging and analysis across multiple excel files

Key input for Attribution Analysis

Excel files from HDFC site

- Print Formatted files, with images. Only data to be extracted
- Clean, merging and analysis done by ChatGPT Plus.
- Time taken : 1-2 mins

ISIN	Name Of the Instrument	Total Assets (Rs. In Lakhs)	% to NAV (Feb to Dec 2023)	Change in % (Dec 2023)
INE52A00092	Kotak Mahindra	175	2.60	0.66
INE52A00093	Instrument	175	2.60	0.66
INE52A00095	HDFC Mutual Fund Limited	5.39	-0.29	-0.54
INE52A00096	ITC Life Insurance	5.31	-0.24	-0.45
INE52A00097	State Bank of India	2.05	1.72	0.95
INE52A00098	Mahindra & Mahindra Ltd.	1.68	0.62	-0.04

New Entries in December 2023:				
ISIN	Name Of the Instrument	Quantity (Dec 2023)	Market Value (Dec 2023, Rs. In Lakhs)	% to NAV (Dec 2023)
INE22TA00028	Kotak Mahindra	4900000	9400000	-3.10
INE06AF00090	Hindustan Aeronautics Limited	199840	3500000	-1.08
INE05BS00028	MAHINDRA PHARMA LIMITED	1044172	2070684	0.66
INE42TA00024	Apollo Hospitals Enterprise Ltd.	20192	1650007	0.42
INE54BD00024	United Spirits Limited	61827	95209	0.30

Dropped Entries from February 2023: ↓

 Amar Harolikar

Mutual Fund Portfolio Analytics with ChatGPT

Merging and analyzing across multiple excel files. Key input for attribution analysis.

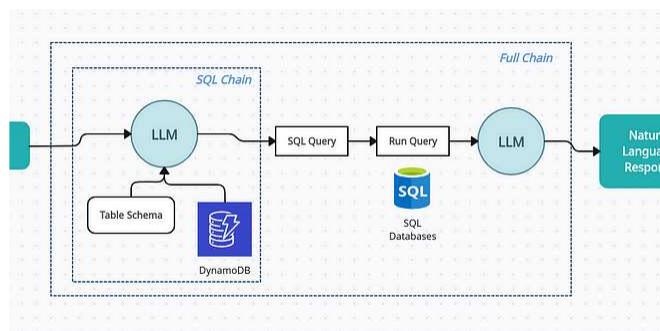
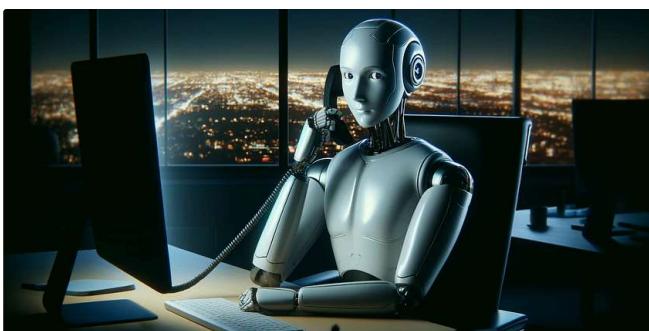
Feb 10



...

See all from Amar Harolikar

Recommended from Medium





Alan Blackmore

AI Voice Agents

AI Voice Agents for Inbound and Outbound Call Coverage

Jul 22

2

1



...



Martyna Rachanczyk

Natural Language to SQL with LangChain 🦜🔗 Amazon Bedrock...

In this tutorial, we will delve into the process of creating a Q&A system based on an SQL...

Apr 30

53



...

Lists



Predictive Modeling w/ Python

20 stories · 1599 saves



Natural Language Processing

1759 stories · 1359 saves



Practical Guides to Machine Learning

10 stories · 1945 saves



ChatGPT prompts

50 stories · 2095 saves

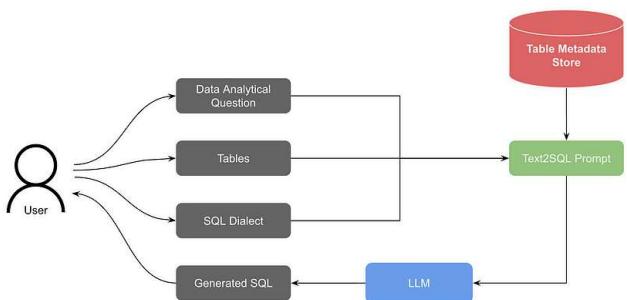


Paras Madan in GoPenAI

Advanced RAG for Database without exposing DB Data: Text to...

Data Privacy is a big concern when implementing RAG Solutions. Companies ar...

Jul 19 155 3



Pinterest Engineering in Pinterest Engineering Blog

How we built Text-to-SQL at Pinterest

Adam Obeng | Data Scientist, Data Platform Science; J.C. Zhong | Tech Lead, Analytics...

Apr 3 2.3K 22

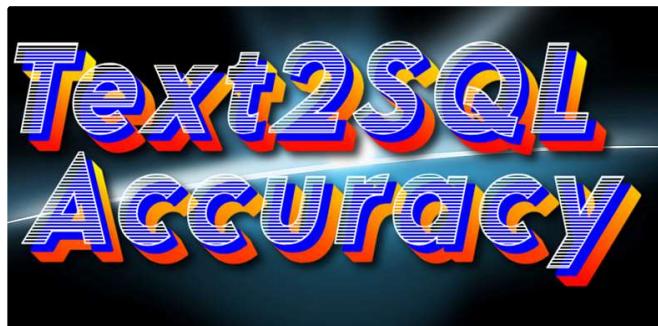


g Reflections on AI

Building SQL Validation Rules with LangGraph

In this blog post, we'll introduce a simple tool created with LangGraph, designed to...

May 30 4



M Gen. David L.

How to Improve Text2SQL Accuracy with Large Language...

Text2SQL is a highly popular application scenario in the AI field. On the BIRD-SQL tes...

Sep 5 4



See more recommendations