

Building an AI-powered ChatGPT App for Non-Technical Managers to Query Azure SQL with Natural Language in Microsoft Teams



Azure SQL



Azure Function - Python



Power Automate



Teams



- Praveen

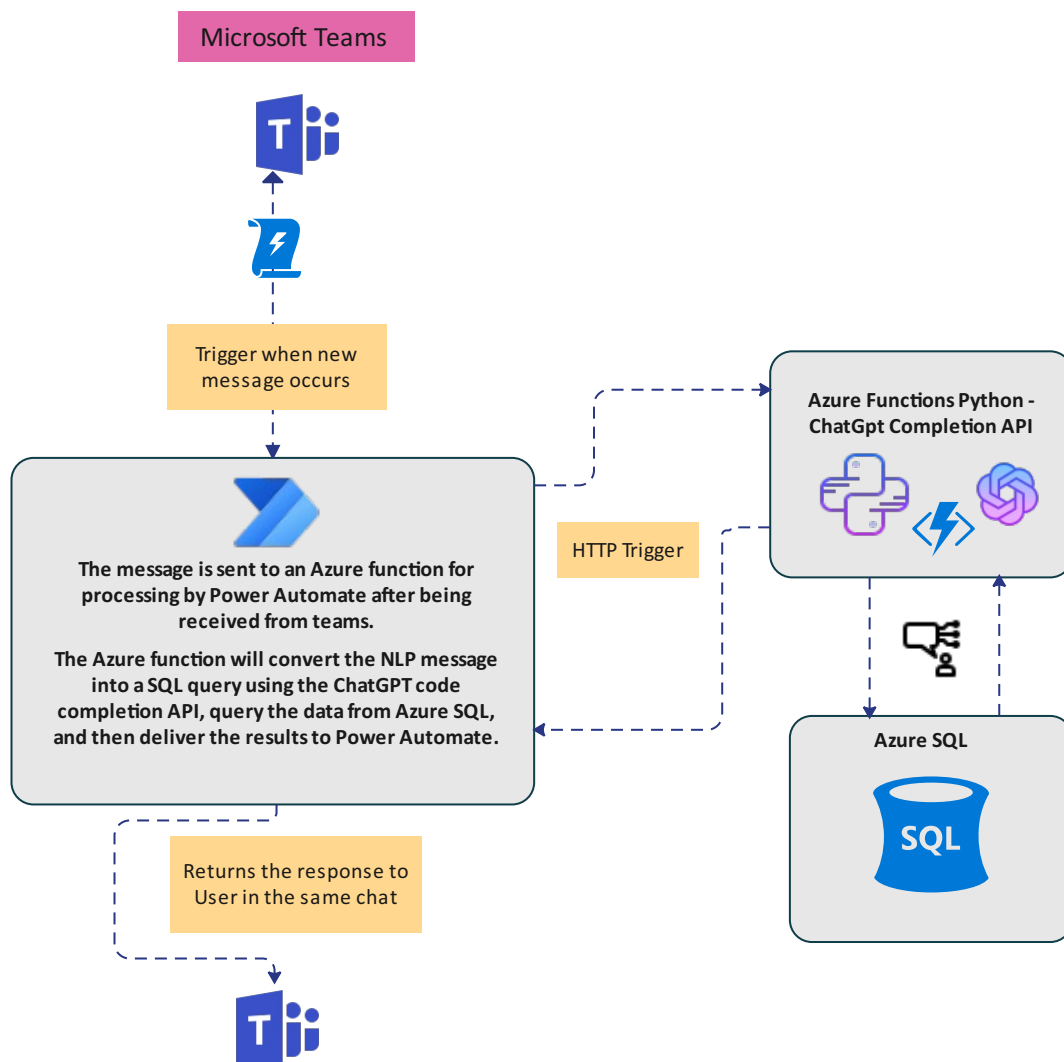
In today's data-driven world, quick and accurate data analysis is vital to the success of any business. However, not all managers have the technical skills to navigate databases and perform complex data queries. To bridge this gap, an AI-powered ChatGPT app can be developed that allows non-technical managers to query data in Azure SQL using natural language.

In this document, we will explore the process of building an AI-powered ChatGPT app that can be integrated with Microsoft Teams Channel to provide a user-friendly interface for querying data.

The app uses natural language processing (NLP) techniques via Open AI Chat GPT codex completion API (Code Davinci 002 model) to convert the user's question into a SQL query, and then sends the query to Azure SQL for execution. The app then retrieves the queried data and displays it back to the user within Microsoft Teams.

We will go into the technical specifics of the app's construction, including the use of Python Azure Functions and the Power Automate & OpenAI Code Completion APIs for generating Azure SQL code.

Sequence of High-Level Architecture Processes



Sequence of High-Level Architecture Processes

- User inputs natural language query into Microsoft Teams chat.
- Power Automate Flow will receive the new message from teams chat and sends the message to Azure Functions via HTTP Trigger Action.
- Azure Function (Python) will query the message and process it using ChatGPT Code Completion API (Code-davinci-002).

ChatGPT Code completion API determines the intent and entities in the query using NLU (Natural Language Understanding) algorithms and generate a SQL query based on the intent and entities.

We use Python to transmit the ChatGPT-generated SQL Query to Azure SQL and retrieve the response data.

Send the response to Power Automate

- Once the response is received, Power Automate formats the results and reply them back to the user in Microsoft Teams chat using Reply to the message action.

Services Used:

OpenAI:

sign up <https://platform.openai.com/> and create the account

Create the api key in <https://platform.openai.com/account/api-keys> (api keys will be used in azure functions for calling the code completion API)

Azure SQL :

Create a DB and a table. Insert records into Table. Copy connection strings from azure portal.

Microsoft Teams :

Create a Teams channel.

Power Automate flow with teams trigger: (When a New message arrives in a channel)

Create a flow trigger when a new message arrives in a teams channel.

Configure the HTTP trigger request for sending user message to Azure Function.

Receive the response & Format the data.

Send the formatted message to the Teams channel by using the action Reply to the teams message.

Azure Function written in python:

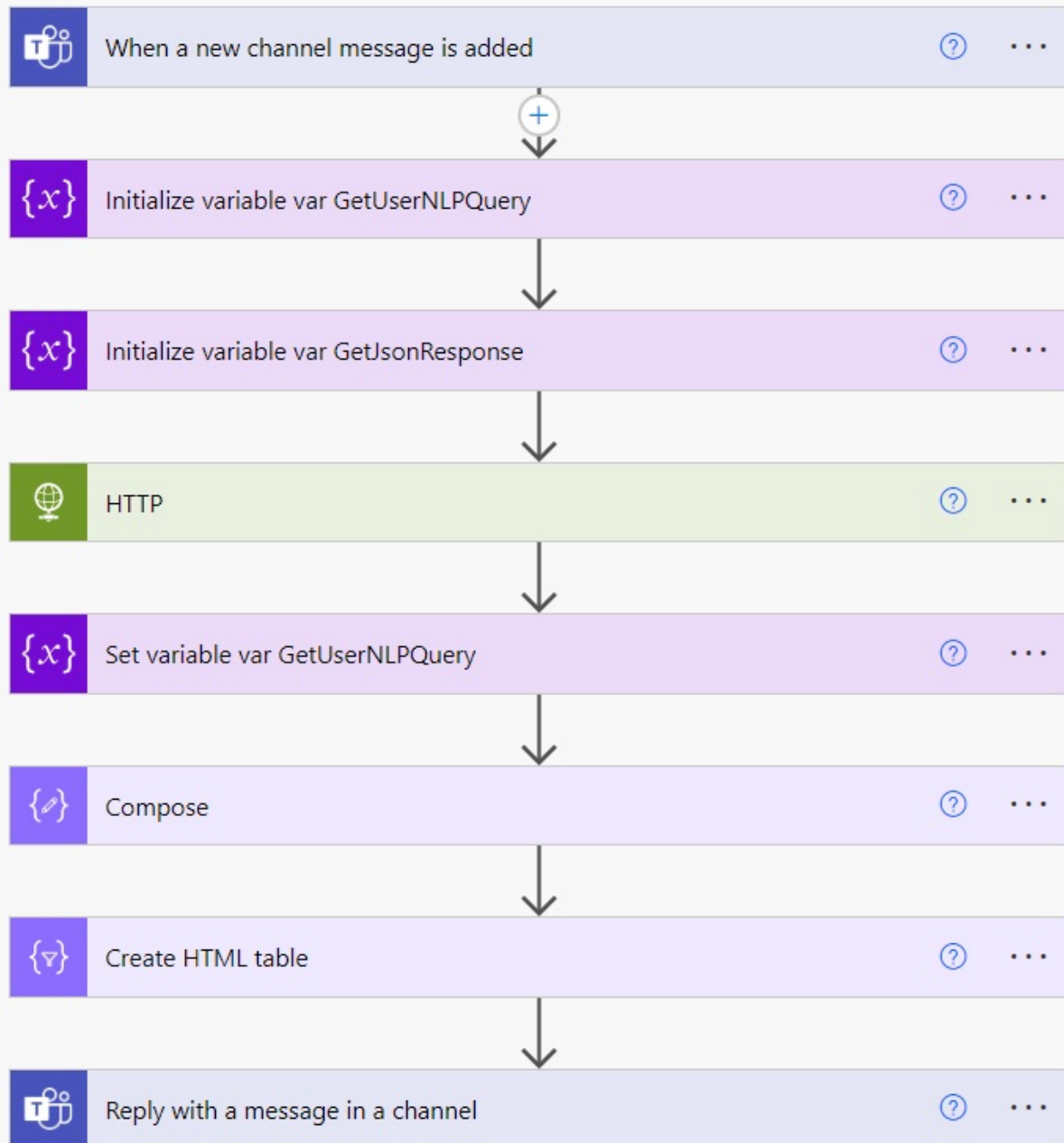
Receives the user message via HTTP triggerred action from power automate.

Process the message using ChatGPT Code Completion API (Code-davinci-002) to form the SQL Query.

Using the AI Generated SQL Query query the data and send the response back to the Power

Automate.

Power Automate Flow Reference



Azure Function - Python Code Completion API Reference

```
response = openai.Completion.create(  
    model="code-davinci-002",  
    prompt=combine_prompts(df, nlp_text),  
    temperature=0,  
    max_tokens=150,  
    top_p=1.0,  
    frequency_penalty=0.0,  
    presence_penalty=0.0,  
    stop=["#", ";"]  
)
```

Detailed explanation of code completion api parameters

Parameter: prompt - Prompts are how you get GPT-3 to do what you want. It's like programming, but with plain English. we need to pass Azure SQL Table DataFrame and user message to provides context that helps the model figure out the best possible completion. Also, quality matters— for example, spelling, unclear text, and the number of examples provided will have an effect on the quality of the completion. Another key consideration is the prompt size.

Parameter: token - When a prompt is sent to GPT-3, it's broken down into tokens. Tokens are numeric representations of words or—more often—parts of words. Numbers are used for tokens rather than words or sentences because they can be processed more efficiently. This enables GPT-3 to work with relatively large amounts of text. That said, as you've learned, there is still a limit of 2,048 tokens (approximately ~1,500 words) for the combined prompt and the resulting generated completion.

You can stay under the token limit by estimating the number of tokens that will be used in your prompt and resulting completion. On average, for English words, every four characters represent one token. So, just add the number of characters in your prompt to the response length and divide the sum by four. This will give you a general idea of the tokens required. This is helpful if you're trying to get an idea of how many tokens are required for a number of tasks.

Parameter: temperature - With a temperature between 0 and 1, we can control the randomness and creativity of the model's predictions. Incase of clear correct answer, lean towards the value of 0 and try a value of 0.9 for more creative answers.

The top p parameter specifies a sampling threshold during inference time. Top p sampling (sometimes called nucleus sampling) is a technique used to sample possible outcomes of the model.

Top p and temperature parameters both control the randomness of the model. OpenAI documentation recommends using either one parameter or the other and setting the unused parameter to the neutral case, i.e. 1.0.

Parameter: frequency_penalty - The frequency penalty parameter controls the model's tendency to repeat predictions. The frequency penalty reduces the probability of words that have already been generated. The penalty depends on how many times a word has already occurred in the prediction.

Parameter: presence_penalty - The presence penalty parameter encourages the model to make novel predictions. The presence penalty lowers the probability of a word if it already appeared in the predicted text. Unlike the frequency penalty, the presence penalty does not depend on the frequency at which words appear in past predictions.


Parameter: stop - The stop sequence is the optional settings that tells the API when to stop generating tokens

Teams Channel User message and response from chatgpt

A

admin 00:34
get sum of sales data group by year


▼ Collapse all

 admin via Power Automate 00:42
Please find the requested PO details below:

YEAR_IDSUM_SALES
2005 5195.6
2023 29963.06
[See less](#)

A

admin 00:45
get all sales data

 admin via Power Automate 00:46
Please find the requested PO details below:

rowindex	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBERS	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	Y
0	10107	30	95.7	2	2871	2/24/2023 0:00	Received1	2	2	
1	10121	34	81.35	5	2765.9	5/7/2023 0:00	Received2	5	2	
2	10134	41	94.74	2	3884.34	7/1/2023 0:00	Received3	7	2	
3	10145	45	83.26	6	3746.7	8/25/2023 0:00	Received3	8	2	
4	10159	49	100	14	5205.27	10/10/2023 0:00	Received4	10	2	

[See less](#)

Potential Use Cases

Use cases:

An AI-powered ChatGPT App for Non-Technical Managers to Query Azure SQL with Natural Language in Microsoft Teams has a wide range of potential use cases across various industries.

Here are some potential use cases for an AI-powered ChatGPT App for Non-Technical Managers to Query Azure SQL with Natural Language in Microsoft Teams:

Sales Analytics:

Non-technical sales managers can use the app to easily access and analyze sales data from Azure SQL. They can ask natural language questions such as "What is the total revenue for the current quarter?" and get instant insights without having to rely on a technical team.

Marketing Campaign Analysis:

Marketers can use the app to quickly query data from Azure SQL related to marketing campaigns. They can ask questions such as "What is the conversion rate for the latest email campaign?" and get real-time data analysis.

Financial Analysis:

Non-technical financial managers can use the app to easily access financial data from Azure SQL. They can ask questions such as "What is the total cost of goods sold for the last fiscal year?" and get instant insights into their company's financial performance.

Human Resources Analytics:

HR managers can use the app to quickly query data from Azure SQL related to employee performance and engagement. They can ask questions such as "What is the employee turnover rate for the past six months?" and get real-time data analysis.

Operations Management:

Non-technical operations managers can use the app to access and analyze operational data from Azure SQL. They can ask questions such as "What is the average production rate for the last week?" and get real-time insights to optimize their operations.