**1. Integrating Your Notebook or Local Model (LM) with ChatGPT**

If you're using a local notebook (e.g., Jupyter) with a custom language model (LM), you can integrate it with ChatGPT in several ways:

**Approach Options:**

* **API Gateway**: Wrap your LM in a REST API (using Flask or FastAPI), then use ChatGPT to send requests and receive responses.
* **LangChain or LlamaIndex**: These frameworks allow you to orchestrate multiple models and tools, including ChatGPT and your LM, for hybrid workflows.
* **Notebook Plugin**: If you're using tools like VS Code or JupyterLab, you can use extensions to connect ChatGPT to your notebook environment.

**Use Case Example:**

You can use ChatGPT to:

* Ask questions about your notebook outputs
* Generate code snippets
* Debug logic
* Summarize results
* Automate report generation

**📊 2. Streamlit Dashboard Integration for Gen AI**

Streamlit is a powerful tool for building interactive dashboards with minimal code. Here's how it helps in Gen AI contexts:

**Benefits:**

* **Real-time Interaction**: Users can input prompts, tweak parameters, and view model outputs instantly.
* **Visualization**: Display charts, tables, and even embeddings or attention maps.
* **Model Control**: Add sliders, dropdowns, and buttons to control model behavior (e.g., temperature, max tokens).
* **Deployment**: Easily deploy your Gen AI app to the web via Streamlit Cloud or other platforms.

**Use Case Example:**

Imagine a dashboard where:

* You upload sales data
* Choose a model (ChatGPT or your LM)
* Ask questions like “What are the top 3 trends?”
* View charts and summaries generated live

**🤖 3. Integrating with AI (End-to-End)**

To fully integrate your analysis with AI, you can build a pipeline like this:

**Architecture Overview:**

1. **Data Ingestion**: Upload CSVs or connect to databases
2. **Preprocessing**: Clean and transform data
3. **Analysis Engine**: Use AI models (ChatGPT, local LM) to generate insights
4. **Visualization Layer**: Streamlit dashboard for interaction
5. **Feedback Loop**: Allow users to refine queries and regenerate insights

**Tools You Can Use:**

* **ChatGPT API** or **OpenAI SDK**
* **LangChain** for orchestration
* **Streamlit** for UI
* **FastAPI** for backend services
* **Docker** for deployment

Notebook Plugin: If you're using tools like VS Code or JupyterLab, you can use extensions to connect ChatGPT to your notebook environment.?

**Copilot said:**

Copilot

Yes, you can integrate ChatGPT directly into your JupyterLab environment using a powerful extension called [**Chapyter**](https://github.com/chapyter/chapyter).

**🧠 What is Chapyter?**

**Chapyter** is a JupyterLab extension that connects GPT-4 (via OpenAI API) to your notebook. It acts as a **code interpreter** that translates natural language into Python code and executes it automatically.

**⚙️ Key Features**

* **Natural Language Programming**: Use %%chat magic command to describe tasks in plain English.
* **Auto Code Generation & Execution**: Chapyter generates and runs Python code instantly.
* **Context-Aware**: It can use previous cell outputs and history for smarter suggestions.
* **Debugging Support**: You can inspect and fix generated code directly in the notebook.
* **Customizable Prompts**: Modify how the model behaves by editing prompt templates.
* **Privacy-Focused**: Uses OpenAI API with no data stored for training.

**🚀 Quick Start**

**Installation**

Shell

pip install chapyter

Show more lines

⚠️ Note: This upgrades JupyterLab to version ≥ 4.0, which may affect existing environments.

**Usage**

1. Set your OpenAI credentials:

Shell

export OPENAI\_API\_KEY=your\_key

export OPENAI\_ORGANIZATION=your\_org

Show more lines

1. In a notebook cell:

Python

%%chat -m gpt-4-0613

List all files in the current folder

Show more lines

1. Chapyter will generate and execute the code automatically.

**📁 Example Use Cases**

* Data cleaning and transformation
* Exploratory data analysis
* Visualization generation
* Machine learning model setup
* Debugging and optimization

Here’s a step-by-step guide to help you set up **Chapyter** in your JupyterLab environment and connect it to the OpenAI API:

**🧠 What is Chapyter?**

Chapyter is a JupyterLab extension that lets you use GPT-4 (or GPT-3.5) as a **code interpreter**. You write natural language instructions in a cell using %%chat, and it generates and executes Python code automatically.

**⚙️ Step-by-Step Setup**

**✅ 1. Install Chapyter**

Shell

pip install chapyter

Show more lines

⚠️ This will upgrade JupyterLab to version ≥ 4.0, which may affect existing environments.

**✅ 2. Set Your OpenAI Credentials**

You need to set your API key and organization ID as environment variables:

**On Windows (CMD):**

BAT

setx OPENAI\_API\_KEY "your-api-key"

setx OPENAI\_ORGANIZATION "your-org-id"

Show more lines

**✅ 3. Use Chapyter in a Notebook**

In a JupyterLab notebook cell, type:

Python

%%chat -m gpt-4-0613

List all files in the current folder

Show more lines

This will:

* Send the prompt to GPT-4
* Generate Python code
* Execute it automatically
* Display the result in the notebook

**✅ 4. Advanced Features**

* Use --history to include previous cell outputs in the prompt
* Customize prompts and model settings via chapyter/programs.py
* Supports Azure OpenAI API too

**📁 Example Notebooks**

* examples/01-quick-start.ipynb: Basic usage
* examples/02-configure-chapyter.ipynb: Customization

You can find these in the [Chapyter GitHub repo](https://github.com/chapyter/chapyter).

Would you like help creating a sample notebook using Chapyter for your sales data analysis? I can also show how to combine this with Streamlit for a full Gen AI dashboard experience.

Sources

This response has no direct citations, the sources below were considered

More

Favicon type

Chapyter: ChatGPT Code Interpreter in Jupyter Notebooks - GitHub

github.com

Favicon type

Getting OpenAI api to work in jupyter. (Using provided example code )

stackoverflow.com

Favicon type

Shannon Zejiang Shen | Introducing Chapyter

szj.io

Favicon type

chapyter · PyPI

pypi.org

Favicon type

Intergrate OpenAI API for Python Programming - CyberGISX

cybergisxhub.cigi.illinois.edu

Favicon type

[P] Chapyter: ChatGPT Code Interpreter in Jupyter Notebooks : r ...

reddit.com

Favicon type

Meet Chapyter: A New Jupyter Extension That Lets ChatGPT ... - MarkTechPost

marktechpost.com

Favicon type

AI Integration in Jupyter Notebooks - OpenAI API Community Forum

community.openai.com

**Provide your feedback on BizChat**

**You said:**

Would you like a sample architecture diagram or code snippet for this integration? yes

**Copilot said:**

Here’s a **sample architecture diagram** showing how you can integrate your local notebook (with a language model) and ChatGPT into a unified Gen AI workflow:

!Integration Architecture

**Architecture Components Explained**

1. **User Interface**
   * **Jupyter Notebook / VS Code**: For local development and experimentation
   * **Streamlit Dashboard**: For interactive visualization and user input
2. **Local Language Model (LM)**
   * Hosted in your environment (e.g., Hugging Face model, LLaMA, etc.)
   * Exposed via **FastAPI** or **Flask** as a RESTful service
3. **ChatGPT Integration**
   * Accessed via **OpenAI API**
   * Used for advanced reasoning, summarization, and code generation
4. **Orchestration Layer**
   * **LangChain / LlamaIndex**: Routes queries between ChatGPT and your LM
   * Handles context, memory, and tool usage
5. **Data Layer**
   * CSVs, databases, or cloud storage
   * Preprocessed and analyzed using Pandas, NumPy, etc.
6. **Visualization Layer**
   * **Streamlit**: Displays charts, tables, and model outputs
   * Allows user interaction with models and data

Here’s a **working code example** that integrates:

* Your **Jupyter Notebook**
* A **local language model (LM)** exposed via FastAPI
* **ChatGPT** via OpenAI API
* A **Streamlit dashboard** for interaction

**🧩 1. FastAPI Wrapper for Your Local Model**

Python

# local\_model\_api.py

from fastapi import FastAPI, Request

import uvicorn

app = FastAPI()

@app.post("/generate")

async def generate(request: Request):

data = await request.json()

prompt = data.get("prompt")

# Replace with your local model inference logic

response = f"Local model response to: {prompt}"

return {"response": response}

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

uvicorn.run(app, host="0.0.0.0", port=8000)

Show more lines

**🧠 2. ChatGPT Integration via OpenAI API**

Python

# chatgpt\_api.py

import openai

openai.api\_key = "your-openai-api-key"

def ask\_chatgpt(prompt):

response = openai.ChatCompletion.create(

model="gpt-4",

messages=[{"role": "user", "content": prompt}]

)

return response.choices[0].message["content"]

**📊 3. Streamlit Dashboard**

Python

# app.py

import streamlit as st

import requests

from chatgpt\_api import ask\_chatgpt

st.title("Gen AI Sales Analysis Assistant")

prompt = st.text\_area("Enter your question or task")

model\_choice = st.selectbox("Choose Model", ["ChatGPT", "Local Model"])

if st.button("Generate"):

if model\_choice == "ChatGPT":

result = ask\_chatgpt(prompt)

else:

response = requests.post("http://localhost:8000/generate", json={"prompt": prompt})

result = response.json()["response"]

st.markdown("### Response")

st.write(result)

Show more lines

**🚀 How to Run This**

1. Start your local model API:

Shell

python local\_model\_api.py

Show more lines

1. Run the Streamlit app:

Shell

streamlit run app.py

Your **Streamlit Gen AI Sales Analysis Dashboard** is now fully integrated with:

✅ **File Upload** for dynamic CSV analysis  
✅ **Interactive Charts** for:

* Total Revenue
* Top Categories
* Regional Trends
* Monthly Seasonality

✅ **AI Insight Generation** using:

* **ChatGPT (GPT-4)** via OpenAI API
* **Local Language Model** via FastAPI

**Customizing the Dashboard Further**

Here are a few ways you can enhance the dashboard:

**1. Add Filters**

Let users filter by:

* Region
* Category
* Date range

Python

selected\_region = st.selectbox("Select Region", df["Region"].unique())

filtered\_df = df[df["Region"] == selected\_region]

Show more lines

**2. Add KPIs**

Show metrics like:

* Average sale per category
* Highest performing month
* Growth trends

Python

st.metric("Average Sale", f"${df['Sales'].mean():,.2f}")

Show more lines

**3. Add Export Options**

Allow users to download filtered data or insights:

Python

st.download\_button("Download Filtered Data", filtered\_df.to\_csv(), "filtered\_data.csv")

Show more lines

**4. Add Chat History**

Keep a log of AI responses for reference:

Python

if "history" not in st.session\_state:

st.session\_state.history = []

st.session\_state.history.append(result)

**You said:**

Implement filters for the dashboard.

**Copilot said:**

Copilot

Analysis

✅ Filters have been successfully implemented in your Streamlit dashboard! Here's what's new:

**🧭 Dashboard Filters Added**

You can now filter the sales data by:

* **Region**: Select one or multiple regions
* **Category**: Choose specific product categories
* **Date Range**: Define a custom time window

**📊 What You See**

* A filtered table of sales records
* Summary metrics:
  + **Total Sales**
  + **Average Sales**

Would you like to add:

* Dynamic charts based on filtered data?
* Export options for filtered results?
* AI-generated insights based on the filtered view?

Press Enter to see more copilot prompts.

**Provide your feedback on BizChat**

**You said:**

what would be the problem statement, idea title, idea description idea contributors

**Copilot said:**

Copilot

Here’s a structured breakdown for your Gen AI-powered sales analysis dashboard project:

**💡 Idea Title**

**Gen AI Sales Insight Assistant**

**❓ Problem Statement**

Sales analysts spend significant time manually cleaning data, generating reports, and identifying trends. This process is repetitive, error-prone, and lacks real-time interactivity. There's a need for an intelligent assistant that can automate data analysis, generate insights, and provide interactive visualizations with minimal effort.

**🌟 Idea Description**

The **Gen AI Sales Insight Assistant** is a Streamlit-based dashboard that integrates:

* **ChatGPT** for natural language insights
* A **local language model** for offline or private inference
* **Interactive filters** for region, category, and date range
* **Dynamic charts and metrics** for sales trends
* **CSV upload support** for flexible data input

Users can explore sales data, ask questions in plain English, and receive AI-generated insights instantly. This reduces manual effort, speeds up decision-making, and democratizes data access across teams.

**👥 Idea Contributors**

* **Raghunandan Malpani** – Concept, data analysis, dashboard design
* **Copilot (AI Assistant)** – Code generation, logic optimization, visualization, integration support

Would you like to format this into a pitch deck slide or a proposal document? I can also help you submit it as an innovation idea within your organization.

**Provide your feedback on BizChat**

**You said:**

add using python libraries

**Copilot said:**

Copilot

Here’s an enhanced version of your project documentation with a focus on **Python libraries used**:

**💡 Idea Title**

**Gen AI Sales Insight Assistant**

**❓ Problem Statement**

Sales analysts often spend excessive time manually cleaning data, generating reports, and identifying trends. This process is repetitive, error-prone, and lacks real-time interactivity. There's a need for an intelligent assistant that automates data analysis, generates insights, and provides interactive visualizations with minimal effort.

**🌟 Idea Description**

The **Gen AI Sales Insight Assistant** is a Python-powered Streamlit dashboard that integrates:

* 📊 **Data Analysis** using pandas
* 📈 **Visualizations** using matplotlib, seaborn, and Streamlit charts
* 🧠 **AI Insight Generation** using:
  + openai for ChatGPT integration
  + FastAPI for local model serving
* 📂 **File Upload & Filtering** using Streamlit widgets
* 🧮 **Metrics & Summaries** using Streamlit KPIs

Users can upload sales data, apply filters (region, category, date range), explore trends, and ask natural language questions to receive AI-generated insights instantly.

**🧰 Python Libraries Used**

| **Library** | **Purpose** |
| --- | --- |
| pandas | Data cleaning, transformation, aggregation |
| matplotlib | Static visualizations |
| seaborn | Enhanced statistical plots |
| streamlit | Interactive dashboard UI |
| openai | ChatGPT API integration |
| requests | Communicating with local model API |
| fastapi | Serving local model as a REST API |
| uvicorn | Running FastAPI server |

**👥 Idea Contributors**

* **Raghunandan Malpani** – Concept, data analysis, dashboard design
* **Copilot (AI Assistant)** – Code generation, logic optimization, visualization, integration support