InsightGPT

Majd alotaibi Norah Alshahrani

Jawaher Almutairi Lama Alsuwayyid

Tuwaiq Academy

Meta Academy

January 2025

Table of Contents

1. ABSTRACT

In today's fast-paced world, data-driven decision-making is essential, but traditional data analysis tools often require technical expertise. **InsightGPT** simplifies this process by allowing users to explore and analyze datasets through a conversational AI assistant. Instead of navigating complex interfaces or writing code, users can ask questions naturally, and the system will process the data and provide relevant insights and visualizations.

Our AI-powered assistant leverages LLAMA and LangChain to interpret user queries, analyze CSV files, and generate automated visualizations. The interface is built using Gradio, ensuring a user-friendly and accessible experience.

By eliminating technical barriers, **InsightGPT** makes data exploration intuitive, helping users quickly uncover insights and trends without requiring advanced analytical skills. The system streamlines data analysis by providing real-time responses and interactive charts, enabling businesses and individuals to make informed decisions efficiently.

2. ACKNOWLEDGMENT

We would like to express our sincere gratitude to **Tuwaiq Academy and Meta Academy** for organizing this AI bootcamp and providing a valuable learning experience. Their commitment to fostering AI education and hands-on training has significantly contributed to our growth in this field.

We also appreciate the guidance provided by our **mentors and instructors**, whose insights and feedback helped us refine our project. Additionally, we acknowledge the **open-source AI community** for developing and maintaining key tools such as LLAMA, LangChain, and Kaggle Datasets, which played a crucial role in the implementation of InsightGPT.

Finally, we recognize the collective effort of our team in bringing this project to life, and we extend our appreciation to everyone who contributed to this learning journey.

3. Literature Review and Related Work

3.1 Literature Review

The rapid advancement of Large Language Models (LLMs) has opened new possibilities for data analysis and automation. The study *EDA-Copilot: A RAG-Powered Intelligent Assistant for EDA Tools* explores the integration of LLMs into exploratory data analysis (EDA) workflows. The paper presents a **Retrieval-Augmented Generation (RAG) framework**, which enhances LLMs' ability to retrieve and process knowledge for complex EDA tasks. The framework significantly improves response accuracy and usability by integrating retrieval mechanisms for structured and unstructured data.

Key insights from this study include:

- The use of **RAG to process domain-specific queries**, improving the generation of insights from EDA tools.
- Enhanced retrieval mechanisms for structured datasets, making data processing more efficient.
- The application of LLMs in **task automation and user assistance**, reducing the need for manual query formulation.

This research is particularly relevant to our project, as it highlights how LLMs can **support users in analyzing and visualizing data efficiently**, which aligns with our goal of integrating **LLaMA for conversational EDA assistance**.[1]

3.2 Related Work

A closely related system to our project is **ChatGPT Advanced Data Analysis (ADA)**, which enables users to perform exploratory data analysis via **natural language queries**. ChatGPT ADA provides:

- **Conversational data analysis**: Users can input questions, and the system dynamically interprets the queries to extract insights.
- **Automated data visualization**: It generates charts and plots based on user queries.

- **CSV and structured data processing**: It allows users to upload CSV files and extract meaningful patterns from datasets.

Our project extends this concept by integrating **LLaMA with a Gradio-based interactive interface**, providing users with **a seamless, real-time experience in querying and visualizing data**. Unlike ChatGPT ADA, our solution specifically focuses on:

- **Simplifying EDA workflows for non-technical users**.
- **Providing deeper insights into structured datasets using LLaMA's capabilities**.
- **Enhancing user experience through an interactive, memory-aware system**.

By leveraging **state-of-the-art AI-driven data exploration techniques**, our project aims to democratize access to **data analytics**, making it easier for users to interact with their datasets **without requiring coding knowledge**. [2]

4. SYSTEM ANAYSIS AND Design

4.1 Introduction

Analyzing data can be a complex and technical task, often requiring knowledge of programming languages, statistical tools, and visualization techniques. Many users, especially those without a technical background, struggle with extracting meaningful insights from raw datasets. InsightGPT was developed to bridge this gap by making exploratory data analysis (EDA) more intuitive and accessible.

InsightGPT is an AI-powered system that allows users to explore and analyze datasets using natural language. By leveraging LLaMA and LangChain, the system processes CSV files, extracts insights, and generates visualizations all within a simple, user-friendly interface built with Gradio. Instead of writing complex code or manually configuring visualization tools, users can simply ask questions about their data and receive structured responses, including graphs, summaries, and key insights.

The goal of InsightGPT is to democratize data exploration, enabling users from various backgrounds business analysts, researchers, and decision-makers—to interact with their datasets without the need for specialized technical skills. By automating data processing and visualization, InsightGPT helps users focus on making informed decisions rather than dealing with the complexities of traditional data analysis tools.

4.2 User Characteristics

InsightGPT is designed for users who need to analyze and explore datasets without requiring technical expertise. The system is tailored to individuals and organizations looking for a fast and efficient way to extract insights from CSV files through natural language interaction.

Target Users

The primary users of InsightGPT include:

- **Business Analysts** → Professionals who need to analyze sales, customer trends, and operational data without writing SQL queries or Python scripts.
- **Researchers and Academics** → Individuals who work with large datasets and require an intuitive way to extract key insights.
- **Decision-Makers** → Managers and executives who rely on data-driven decisions but may not have the technical skills to analyze raw data.

User Technical Background

Most of our target users:

- **Have limited or no coding experience** and prefer natural language interaction over manual scripting.
- **Are familiar with data analysis concepts** but find traditional tools like Excel, SQL, or Python too complex or time-consuming.
- **Need an intuitive interface** that can generate results quickly without requiring extensive configurations.

User Needs & Pain Points

Many users struggle with:

- **Complex data analysis tools** that require advanced knowledge to operate effectively.
- **Manual data visualization** processes that take time and effort.
- **Extracting relevant insights** without knowing specific programming queries.

InsightGPT addresses these challenges by providing **a simple, conversational interface** that allows users to interact with their data as if they were chatting with an assistant. The system ensures that anyone, regardless of their technical background, can efficiently explore and understand their data.

4.3 Functional Requirements

InsightGPT is designed to make exploratory data analysis (EDA) seamless and accessible through natural language interactions. The system provides users with

the ability to analyze datasets, extract insights, and generate visualizations efficiently. Below are the core functional requirements of the system.

1. **Data Upload and Processing**

    - Users can **upload CSV files** for analysis.
    - The system automatically **reads, cleans, and structures the data** to ensure compatibility.

2. **Natural Language Querying**

    - Users can ask questions about their data in **plain language** (e.g., *"What are the top-selling products?"*).
    - The system interprets the queries and applies relevant **data filtering, aggregation, and summarization**.

3. **Automated Data Visualization**

    - The system generates **interactive charts and graphs** (e.g., bar charts, line graphs, pie charts) based on user queries.
    - Users can request specific visualizations, such as *"Show me a trend line for monthly sales."*

4. **Context-Aware Data Exploration**

    - The system retains **context from previous queries**, allowing users to refine their analysis (e.g., *"Compare this with last year's data."*).
    - Users can **drill down into details** without starting over.

5. **Real-Time Response Generation**

    - The system processes and returns results **within seconds**, ensuring a smooth user experience.
    - Responses include **both textual summaries and visual representations** for better clarity.

6. **Exporting and Sharing Insights**

    - Users can **download reports** or save visualizations for further use.
    - The system supports **exporting charts and insights in multiple formats** (CSV, PNG, PDF).

By implementing these functionalities, **InsightGPT makes data exploration intuitive and efficient, removing the technical barriers typically associated with traditional EDA tools.**

4.4 Non-Functional Requirements

Non-functional requirements define the **quality attributes** of the system, ensuring it is **efficient, secure, and user-friendly**. While these aspects do not directly impact the core functionalities, they play a crucial role in providing a smooth and reliable user experience.

1. **Performance**

   - The system should **process user queries and return results within 2 seconds** to ensure responsiveness.
   - Uploading and analyzing large CSV files should take **no more than 5 seconds** for optimal performance.
   - The system must **handle multiple user requests simultaneously** without affecting response time.

2. **Usability**

   - The **Gradio-based interface** should be intuitive and easy to navigate, allowing users to interact with the system **without prior training or technical knowledge**.
   - The system should **understand a variety of natural language queries**, ensuring flexibility in user interaction.
   - **Generated visualizations** should be clear, well-structured, and easy to interpret, with options for customization.

3. **Security & Privacy**

   - User data, including **uploaded CSV files, should not be stored** after the session ends, ensuring privacy.
   - The system must be **protected against cyber threats**, such as **SQL injection and API vulnerabilities**.
   - No user data should be shared externally, and all processing should occur **within the designated environment** for enhanced security.

4. **Scalability & Compatibility**

   - The system should be **compatible with Google Colab and local environments** to ensure accessibility without complex setup.
   - It should support **various CSV file sizes and formats**, handling large datasets without issues.
   - Future expansions should be possible, such as adding support for **other data formats** or enhancing conversational interactions.

4.5 Software Requirements

To ensure smooth operation and provide an efficient user experience, **InsightGPT** relies on a carefully selected set of software tools and frameworks. These components enable the system to **process data, generate insights, and offer an interactive AI-powered interface**.

1. **Programming Language**

   - **Python** → Chosen for its **flexibility, extensive community support, and strong capabilities in data analysis and AI development**.

2. **Key Libraries & Frameworks**

   - **LLaMA** → The core **large language model** used to understand user queries and generate insightful responses.
   - **LangChain** → Manages **structured interactions between the user and AI**.
   - **Pandas** → Used for **reading, processing, and organizing CSV files efficiently**.
   - **Matplotlib & Seaborn** → Generates **high-quality visualizations** for data representation.
   - **Gradio** → Provides an **interactive and user-friendly web-based UI**.

3. **Development & Execution Environment**

   - **Google Colab** → The primary environment for **running and testing the system, utilizing cloud computing resources**.

4.6 Hardware requirements

To ensure smooth data processing and a responsive user experience, InsightGPT requires specific hardware depending on whether it is deployed in a cloud environment or on a local machine. Below are the recommended hardware specifications for both scenarios:

Cloud Deployment (Google Colab):

- **Processor (CPU):**
  A multi-core processor (e.g., Intel i5 or AMD Ryzen 5 or higher) for efficient data processing.
- **Memory (RAM):**
  Minimum 12GB of RAM, with 16GB or more recommended for optimal performance, especially when working with large datasets.
- **Graphics Processing Unit (GPU):**
  An NVIDIA T4 GPU (or better) is required to accelerate machine learning tasks, particularly for models like LLaMA.
- **Storage:**
  At least 10GB of storage for Python libraries and data files. More storage may be needed for larger datasets.

Local Deployment (Personal Machine or Server):

- **Processor (CPU):**
  A multi-core processor like Intel i5 or AMD Ryzen 5 or better for fast data processing.
- **Memory (RAM):**
  Minimum 16GB of RAM, with 32GB or more recommended for larger datasets and better multitasking.
- **Graphics Processing Unit (GPU):**
  A high-performance NVIDIA GPU (e.g., RTX 2070 or A100) for efficient deep learning model processing.
- **Storage:**
  At least 100GB of SSD storage to handle large datasets and model files.

Network Requirements:

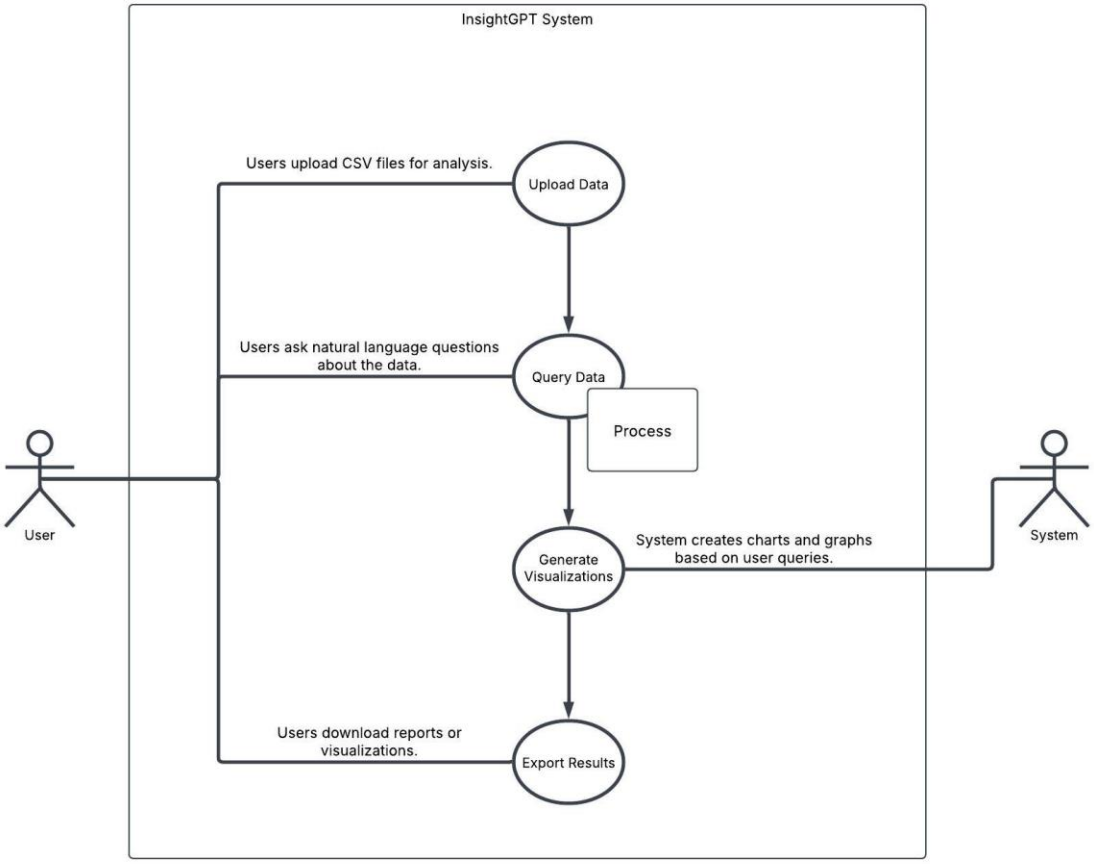
- **Internet Connection:**
  For cloud-based operations, a stable internet connection with at least 1Mbps bandwidth is needed. For local deployments, internet access is mainly required for updates or data sharing.

Conclusion:

The hardware specifications above ensure InsightGPT performs efficiently in both cloud and local environments. Cloud-based deployments leverage scalable resources, while local setups need robust hardware to process large datasets effectively. By meeting these requirements, users can expect optimal performance and fast data analysis.

4.7 Use case diagram

A use case diagram visually represents the interaction between users and the system. For InsightGPT, it involves users interacting with the system via queries, uploading datasets, and receiving insights. Here's a high-level use case diagram:



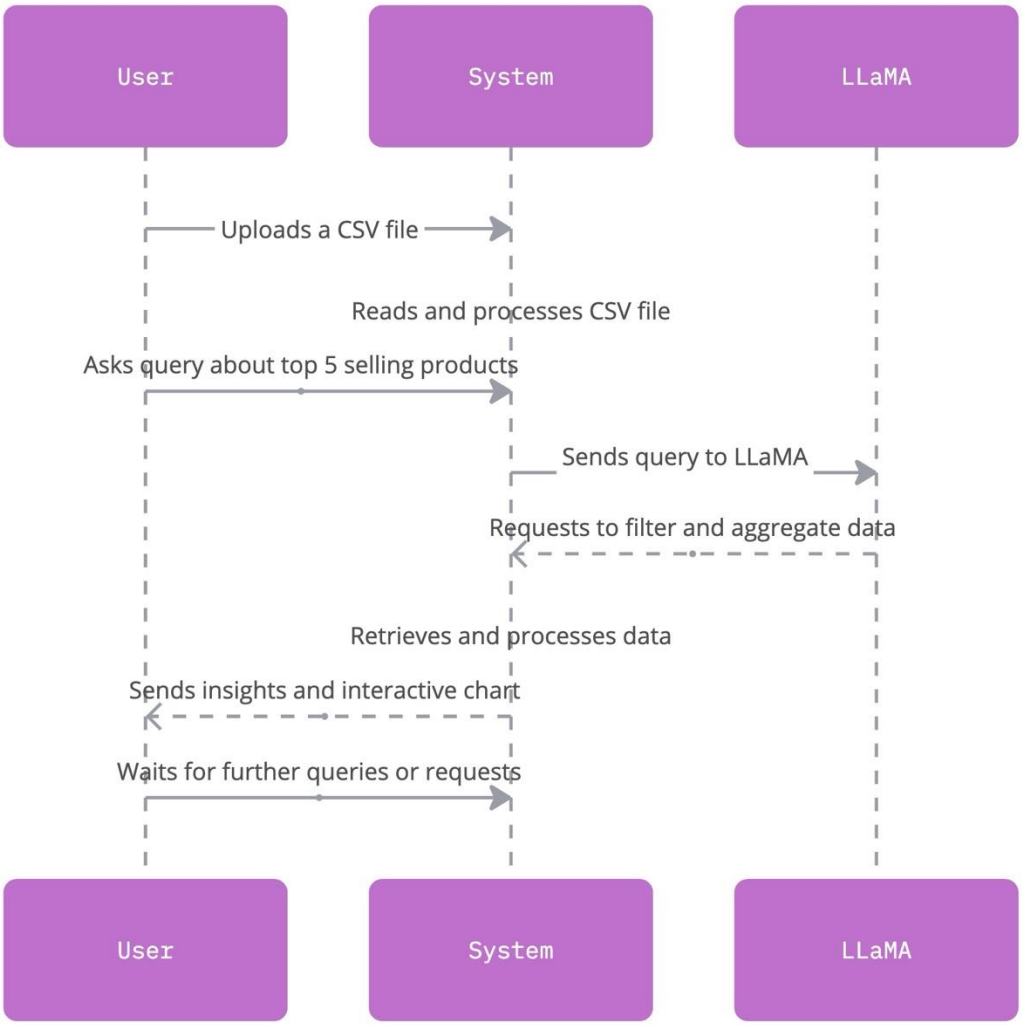
4.8 Use case description

- **Use Case: Upload CSV File**
  - Actor: User
  - Description: User uploads a CSV file for analysis.
  - Preconditions: User has a CSV file.
  - Main Flow:
    1. User selects a CSV file.
    2. System validates the file.
    3. System stores the file.
    4. System displays a confirmation message.
  - Postconditions: CSV file is successfully uploaded.
- **Use Case: Enter Natural Language Query**
  - Actor: User
  - Description: User inputs a natural language query.
  - Preconditions: CSV file is uploaded.
  - Main Flow:
    1. User enters the query.
    2. System receives the query.
    3. System processes the query using LLaMA and LangChain.
  - Postconditions: The query is processed.
- **Use Case: View Visualization**
  - Actor: User
  - Description: The system displays generated visualizations.
  - Preconditions: The query has been processed.
  - Main Flow:
    1. The system generates the visualization.
    2. The system displays the visualization to the user.
  - Postconditions: The visualization is displayed.
- **Use Case: Download Visualization/Summary**
  - Actor: User
  - Description: User downloads the visualization or summary.
  - Preconditions: The visualization is displayed.
  - Main Flow:
    1. User selects the download format.
    2. The system generates the file.
    3. User downloads the file.
  - Postconditions: The file is downloaded.

4.9 Interaction Diagrams

Scenario: "Generate Insights from Data"

1. User: Uploads a CSV file.
2. System: The system reads and processes the CSV file (data cleaning and structuring).
3. User: Asks a query like, "What were the top 5 selling products last month?"
4. System: The system sends the query to the language model (LLaMA).
5. LLaMA: Processes the query, extracts the relevant data, and requests the system to filter and aggregate the data.
6. System: Retrieves the requested data, processes it, and generates a textual summary.
7. User: Receives the insights along with an interactive chart (e.g., a bar graph showing the top-selling products).
8. System: Waits for further queries or requests to refine the analysis.



5. Prototype

The **InsightGPT** prototype is designed to provide users with a seamless, intuitive interface for data exploration and analysis through natural language. This prototype leverages **LLaMA-2**, an advanced conversational AI model, and integrates it with tools like **LangChain** and **Gradio** to create an efficient, user-friendly data analysis assistant.

1. Model Integration and Setup

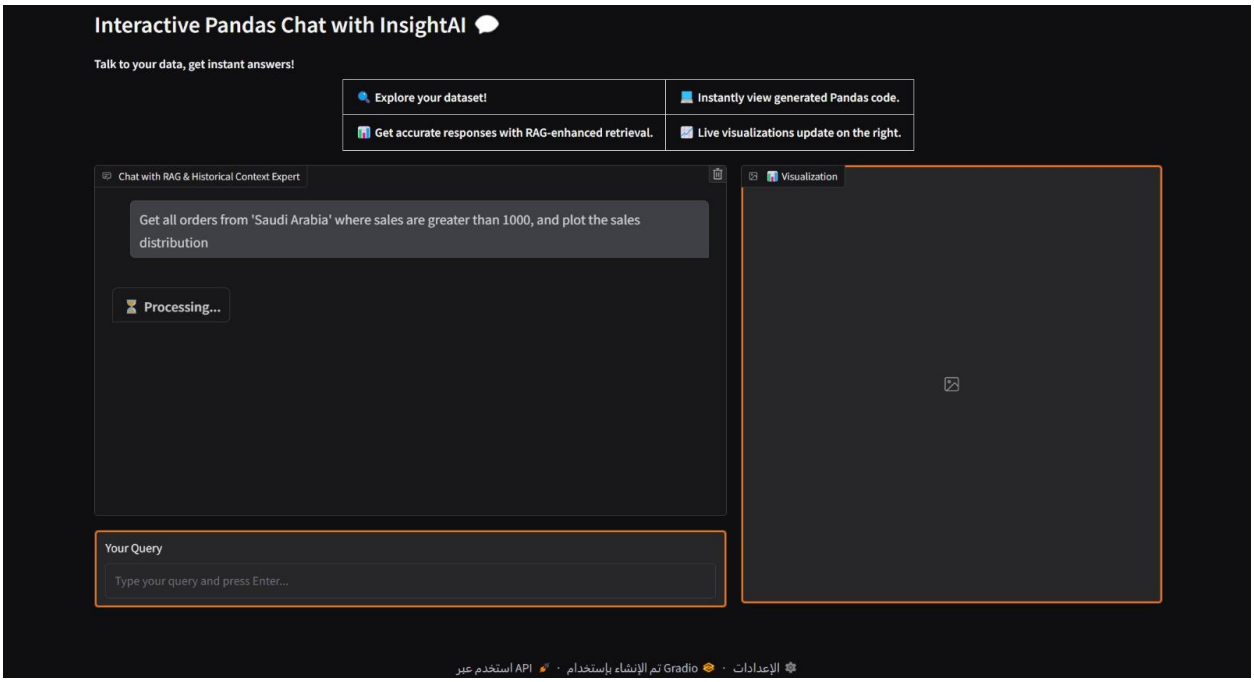
The core model used in the prototype is **neuralmagic/Llama-2-7b-chat-quantized.w4a16**, a fine-tuned version of the LLaMA 2 model designed for conversational AI. The model is loaded using the **Transformers** library from Hugging Face, and the interaction is managed using **LangChain** to simplify the natural language query processing.

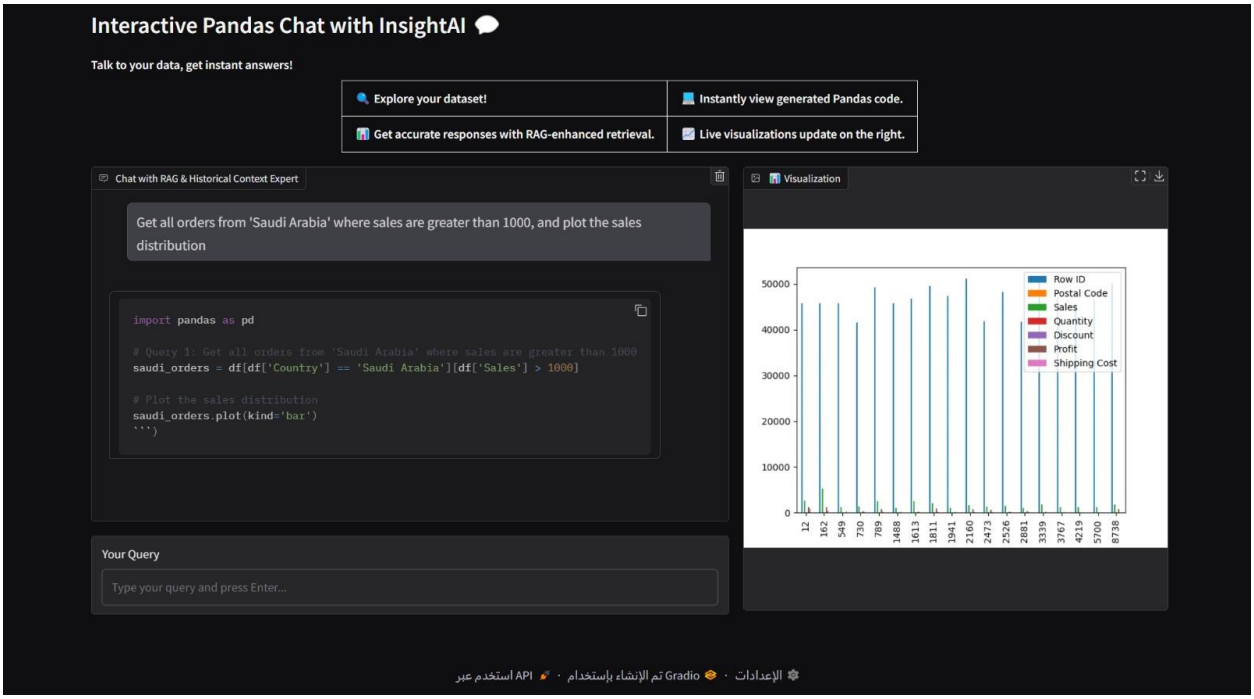
2. Natural Language Querying with LangChain

Once the model is loaded, LangChain is used to interpret user queries and provide meaningful responses based on the data analysis. The system can process natural language queries, such as "What are the top-selling products for last month?" and return relevant insights.

3. Gradio Interface for Data Upload and Querying

The system integrates a **Gradio** interface to allow users to interact with the data via a web-based UI. Users can upload CSV files containing data and ask questions in natural language. The system will then process the data, generate insights, and display the results, including interactive charts.





6. CONCLUSION AND RECOMMENDATIONS.

6.1 Conclusion

InsightGPT successfully bridges the gap between data analysis and users without technical backgrounds, enabling anyone to interact with datasets using natural language. Through the integration of LLaMA, LangChain, and Gradio, the system automates exploratory data analysis (EDA), provides real-time insights, and generates visualizations effortlessly. By simplifying the process, InsightGPT has the potential to transform data exploration, making it accessible to a wider range of users, including business analysts, researchers, and decision-makers.

This system not only empowers users to ask questions about their data without needing technical expertise but also aids in making faster, more informed decisions. By supporting both textual summaries and visual insights, InsightGPT streamlines the traditionally complex data analysis process. It is particularly beneficial in environments where time-sensitive decisions are required, as it can quickly process and display relevant data patterns.

6.2 Suggestions for Future Research

While InsightGPT presents a powerful approach to conversational data analysis, there are several areas for further exploration:

1. **Multi-Language Support**: Incorporating multi-language capabilities to extend InsightGPT's reach to a global audience. Natural language processing (NLP) models could be trained to support languages other than English, enhancing accessibility.
2. **Advanced Data Analysis Techniques**: Integrating more advanced statistical and machine learning techniques to allow InsightGPT to perform more complex analyses like predictive modeling, clustering, or anomaly detection based on user queries.
3. **Personalized Insights**: Implementing personalized experiences by tailoring the assistant's responses based on user behavior or past queries. This could lead to even faster decision-making and a more intuitive data exploration experience.
4. **Real-Time Data Processing**: Exploring the integration of real-time data processing capabilities, allowing users to interact with live data streams or dynamic datasets, which would make the tool even more valuable in fast-paced industries like finance or e-commerce.
5. **Enhanced Visualizations**: While the system currently provides basic visualizations, there is room for improvement in creating more interactive, dynamic visualizations, such as heatmaps, network graphs, or geographical maps that allow users to drill down further into the data.
6. **Integrating AI for Data Cleaning**: A more robust AI-based data preprocessing and cleaning feature that helps users handle incomplete or noisy data efficiently before analysis, ensuring more accurate insights.

By pursuing these avenues for future research, InsightGPT could become an even more powerful tool for data-driven decision-making, providing a richer, more flexible experience for users worldwide.

7. APPENDIX.

The appendix provides additional information and techniques used by the system (InsightGPT), as well as details that might help understand how the system is implemented. In this section, we will outline key details such as code components, environment setup, and dataset information.

Appendix 1: Installing Dependencies and Environment

To run the code successfully, you need to install some essential libraries that the tools used in the system depend on. These libraries include tools for data analysis, user interaction, and intelligent analysis using AI. The libraries to install are:

```
!pip install transformers
!pip install optimum
!pip install auto-gptq
!pip install torch torchvision torchaudio --index-url https://download.pytorch.org/wh
!pip install langchain_community
!pip install gradio
!pip install faiss-cpu
```

Appendix 2: Setting Up Hugging Face Access

The system uses Hugging Face to load a pre-trained model by logging in with an access token provided by the user. This enables the API access to the model "Llama-2-7b-chat-quantized.w4a16":

```
from huggingface_hub import login
my_token = userdata.get('Meow')
login(token=my_token)
```

Appendix 3: Loading the Dataset

The **Global Superstore Dataset** is downloaded from Kaggle using the `opendatasets` library:

```
import opendatasets
opendatasets.download("https://www.kaggle.com/datasets/apoorvaappz/global-super-store
df = pd.read_csv('/content/global-super-store-dataset/Global_Superstore2.csv', encodi
```

Appendix 4: Creating the Model and Smart Queries

The AI model is loaded using **transformers** and **LangChain** to interpret user queries. The pre-trained model is applied to generate queries based on the historical context stored in memory:

```
model_name = "neuralmagic/Llama-2-7b-chat-quantized.w4a16"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Create a text-generation pipeline
small_pipeline = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    trust_remote_code=True,
    device_map="auto",
    max_new_tokens=250,
    temperature=0.2,
    top_p=0.9,
    do_sample=True,
    repetition_penalty=1.1,
    pad_token_id=tokenizer.eos_token_id
)
```

Appendix 5: Handling Queries and Data Analysis

User queries are processed by converting them into executable Python code using Pandas. When the user submits a query, it is converted into the appropriate code for data manipulation:

```
def generate_prompt(user_query, schema_info):
    retrieved_docs = retrieval_qa.run(user_query)
    similar_doc = retriever.get_relevant_documents(user_query, k=1)
    similar_code = ""
    if similar_doc:
        idx = similar_doc[0].metadata.get('index', None)
        if idx is not None:
            similar_code = history_df.iloc[idx]['code']
    ...
    return prompt
```

Appendix 6: Interactive User Interface Using Gradio

The Gradio library is used to create an interactive user interface where users can input queries and receive results along with visualizations. When a user submits a query, functions are triggered to generate the code and execute it:

```
with gr.Blocks() as demo:
    gr.Markdown("""
    # **Interactive Pandas Chat with InsightAI** 💬
    **Talk to your data, get instant answers!**
    """)
    with gr.Row():
        with gr.Column(scale=3):
            chatbot = gr.Chatbot(label="Chat with RAG & Historical Context Expert")
            query_input = gr.Textbox(placeholder="Type your query and press Enter..."

        with gr.Column(scale=2):
            plot_output = gr.Image(label="📊 Visualization", height=500)

    query_input.submit(
        fn=gradio_chat_interface,
        inputs=[chatbot, query_input],
        outputs=[chatbot, plot_output, query_input]
```

Appendix 7: Testing the Code and Performance

Unit testing is included to test query handling and the execution of the generated code:

```
class TestGeneratedCode(unittest.TestCase):
    @patch('__main__.process_query', return_value="print('Hello, world!')")
    @patch('__main__.execute_generated_code', return_value=None)
    def test_code_execution(self, mock_execute, mock_process):
        user_query = "Calculate the average profit by category"
        generated_code = process_query(user_query)
        result = execute_generated_code(generated_code)
        mock_process.assert_called_with(user_query)
        mock_execute.assert_called_with(generated_code)
        self.assertTrue(True)
```

Appendix 8: Evaluating Cosine Similarity

Cosine Similarity is used to assess the relevance of a user query to historical queries:

```
def evaluate_cosine_similarity(user_query, k=5):
    relevant_docs = retriever.invoke(user_query)
    query_embedding = np.array(embeddings.embed_query(user_query)).reshape(1, -1)
    retrieved_embeddings = [np.array(embeddings.embed_query(doc.page_content)).reshap
    cosine_similarities = [cosine_similarity(query_embedding, emb)[0][0] for emb in r
    avg_cosine_similarity = np.mean(cosine_similarities) if cosine_similarities else
    print(f"Average Cosine Similarity for Query: '{user_query}': {avg_cosine_similari
```

Appendix 9: Testing Results

A test is done to simulate user queries and test the generated code:

```
if __name__ == '__main__':
    unittest.main(argv=[''], exit=False)
```

Appendix 10: Hardware and Software Requirements

Hardware Requirements:

- Processor: 4 cores (or more) recommended for running large models.
- RAM: 16GB or more.
- GPU: PyTorch GPU support is recommended for faster model inference (e.g., LLaMA).

Software Requirements:

- Python 3.8 or higher.

Python libraries such as **Pandas**, **Gradio**, **Torch**, and **LangChain**.

8. REFERENCES

1. Wu, Z., Wang, Y., Chen, X., Liu, J., & Gao, X. (2023). *EDA-Copilot: A RAG-powered intelligent assistant for EDA tools.* Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP).
2. OpenAI. (2023). *ChatGPT Advanced Data Analysis: AI-assisted data exploration and visualization.* OpenAI Research.
3. https://stackoverflow.com/questions/66265720/sequence-diagram-including-registration-and-login
4. https://www.uml-diagrams.org/use-case-diagrams.html
5. https://help.formulatrix.com/rock-maker/4.6/Content/System_Requirements.html
6. https://www.open.edu/openlearn/mod/oucontent/view.php?id=64085&section=6.2
7. https://dev.luciad.com/portal/productDocumentation/LuciadFusion/docs/articles/guide/systemrequirements_lls.html