# Groq API: Conversation Management & Information Extraction 🚀

This project demonstrates two core NLP tasks using the Groq API with the OpenAI-compatible Python SDK.

It is implemented in a Google Colab notebook and designed to be completely framework-free, relying only on standard Python libraries.

---

## 🎯 Objectives

1. Efficiently manage and summarize a conversation history.

2. Extract structured information (like user details) from unstructured text using function calling.

---

## ✨ Features

- Conversation History Management: Maintains a running history of user–assistant interactions.

- Periodic Summarization: Automatically summarizes the conversation after every `k` turns to keep context concise.

- Flexible Truncation: Retrieve conversation history with limits on turns or character length.

- Structured JSON Extraction: Uses Groq’s function-calling feature to parse chats and extract details (name, email, phone, etc.).

- Framework-Free: Built with only the `groq` and `openai` Python clients (no LangChain or other frameworks).

## 🔧 Setup and Installation

#Prerequisites

- Google Account (for Colab)

- GitHub Account

- Groq API Key → Get one here: [Groq Console](https://console.groq.com/keys)

### Installation Steps

1. Clone the Repository

```bash

git clone <https://github.com/rahulj004/Groq-Assignment-Repo.git>  
  
2. **Open in Google Colab**  
Upload the provided .ipynb notebook into Colab.

**3. Install Dependencies**  
Run the first cell in the notebook:

!pip install groq openai -q

**4. Configure API Key**  
Update the setup cell with your key:

5.GROQ\_API\_KEY = "YOUR\_GROQ\_API\_KEY"

**⚙️ How It Works**

The project is divided into two main tasks.

**Task 1: Conversation Management**

Handled by the ConversationManager class.

**Key Methods:**

* **\_\_init\_\_(self, client, model, summarize\_after\_k\_turns)**  
  Initializes with the Groq client and sets summarization frequency.
* **add\_message(self, role, content)**  
  Adds a new message to history. After every k assistant responses, it auto-summarizes and replaces old history with a compact summary.
* **get\_truncated\_history(self, max\_turns, max\_chars)**  
  Returns the conversation history truncated by number of turns or character length.

**Example:**

# Initialize to summarize after every 3 assistant turns

manager = ConversationManager(client, MODEL, summarize\_after\_k\_turns=3)

# Add messages

manager.add\_message("user", "Hi, what's the weather like in Pune?")

manager.add\_message("assistant", "It's currently 28°C and partly cloudy in Pune.") # Turn 1

# ... after 3 turns, history will be summarized automatically

**Task 2: Information Extraction**

This task uses **function calling** to extract structured details from free-form text.

**JSON Schema:**

A schema defines the structure of extracted details (e.g., name, email, age).

**Extraction Function:**

**extract\_user\_details(chat\_content)**

1. Takes a string input (chat message).
2. Calls the Groq API (chat.completions.create) with the schema-defined tool.
3. If relevant details are present, the model calls the tool → returning data as JSON.
4. If no details are present, it returns None.

**Example:**

chat = "Hi, my name is Rahul and my email is rahul@example.com."

details = extract\_user\_details(chat)

# Output

# { "name": "Rahul", "email": "rahul@example.com" }

**📂 Project Structure**

Groq-Assignment-Repo/

│── README.md # Documentation

│── groq\_conversation.ipynb # Google Colab Notebook (main implementation)

**🚀 Future Improvements**

* Extend schema to extract addresses, company names, or social handles.
* Add persistence layer (e.g., SQLite/JSON file) for saving conversation history.
* Provide visualization of conversation summaries.