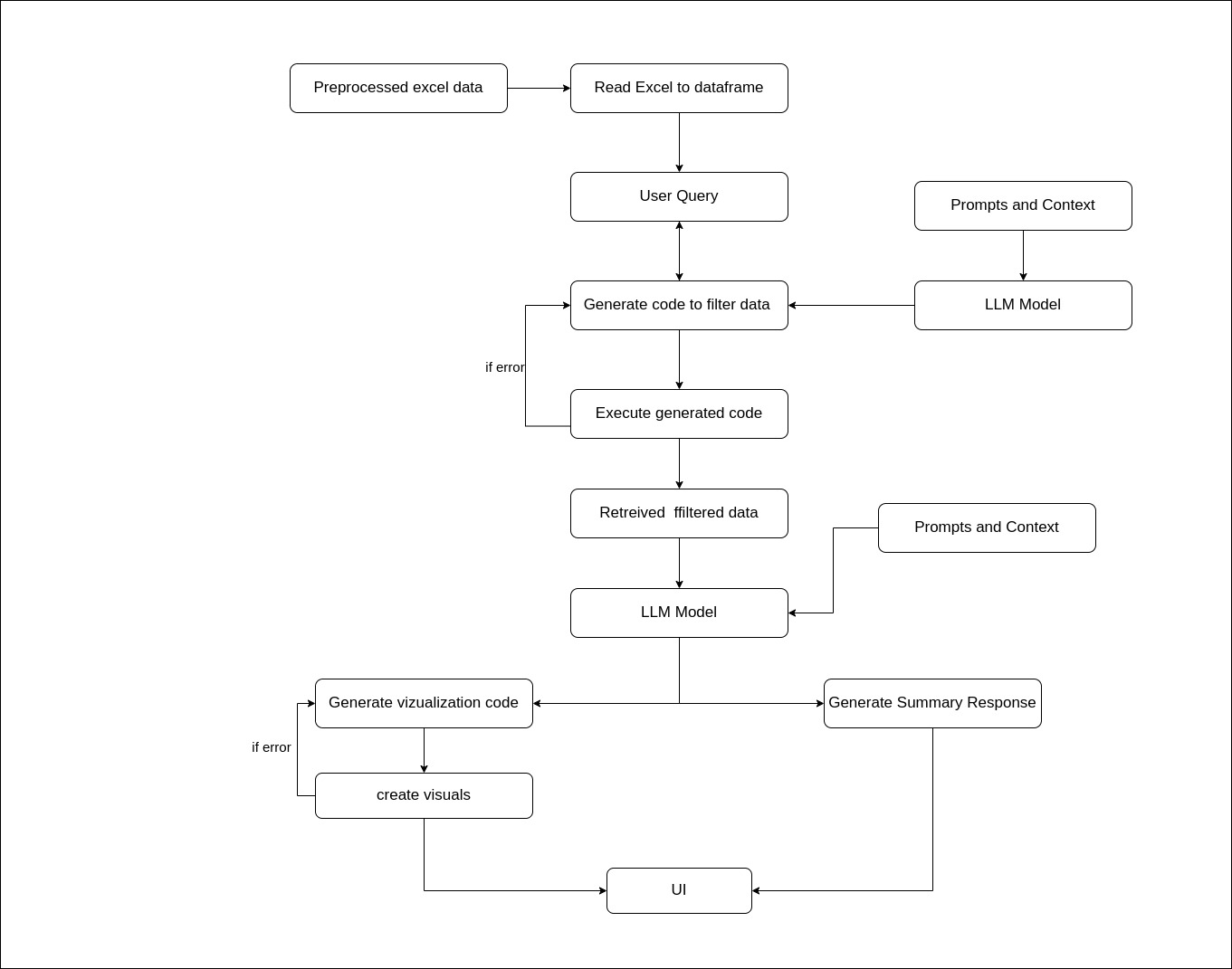
### **Findings (So Far)**

We began by working with a cleaned Excel file and created prompts intended to generate visualizations. These prompts were passed to a Language Model (LLM) along with instructions to extract the relevant data from the Excel file based on the prompt's intent. The LLM successfully retrieved the corresponding data, which was then used for generating visualizations and summaries based on the prompt using particular tech stack and UI tool.



Example Prompt Used:

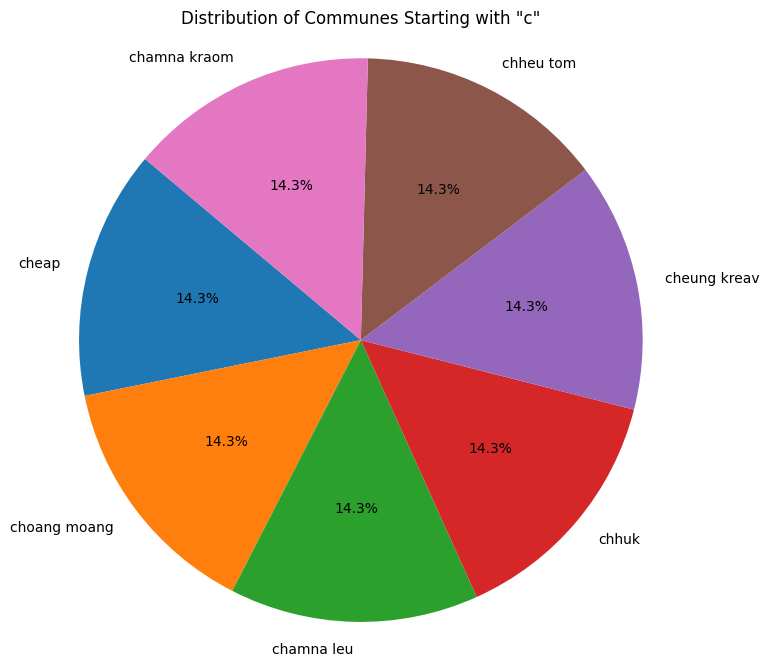
**>>>> “Plot a pie chart that shows the distribution of the ‘commune’ column where values start with the letter ‘c’.**”

1.....First it retreives the relevant data from excel

Relevant data we get ..

[{""Index"":""cheap"",""Value"":28},{""Index"":""choang moang"",""Value"":28},{""Index"":""chamna leu"",""Value"":28},{""Index"":""chhuk"",""Value"":28},{""Index"":""cheung kreav"",""Value"":14},{""Index"":""chheu tom"",""Value"":14},{""Index"":""chamna kraom"",""Value"":14}]"

2......Plot that will be generated--



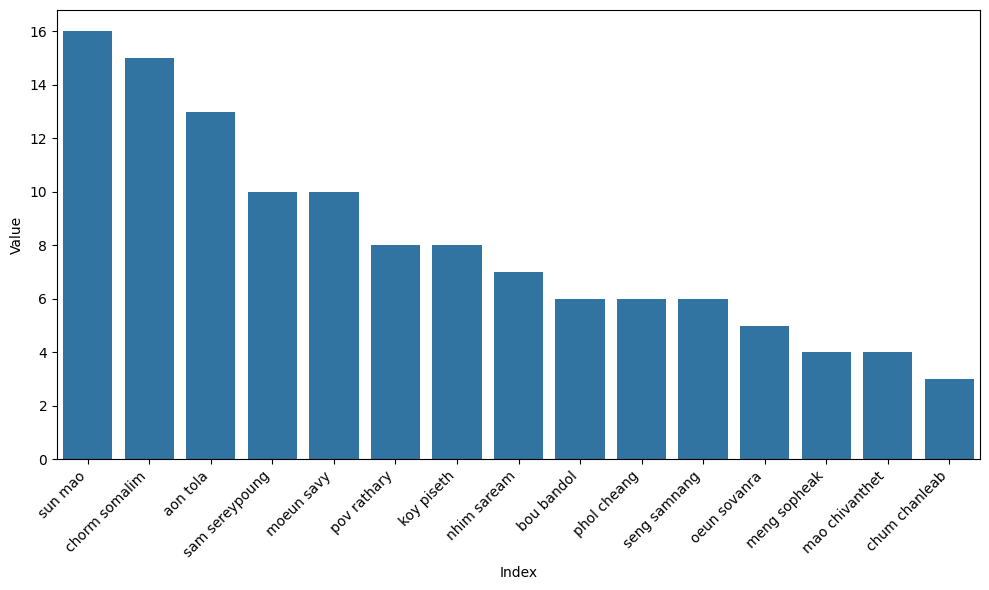
**>>>> How many people in the dataset in Respondent Edu column that completed grade\_2 also show their corresponding Name of enumerator**.

1.....First it retreives the relevant data from excel

Relevant data we get --

|  |  |  |
| --- | --- | --- |
| **Index** | **Value** |  |
| sun mao | 16 |  |
| chorm somalim | 15 |  |
| aon tola | 13 |  |
| sam sereypoung | 10 |  |
| moeun savy | 10 |  |
| pov rathary | 8 |  |
| koy piseth | 8 |  |
| nhim saream | 7 |  |
| bou bandol | 6 |  |
| phol cheang | 6 |  |
| seng samnang | 6 |  |
| oeun sovanra | 5 |  |
| meng sopheak | 4 |  |
| mao chivanthet | 4 |  |
| chum chanleab | 3 |  |

Plot that will be generated--



**Pricing for  LLm model deployment on EC2 Compute**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Instance** | **GPU** | **On Demand Price/hr\*** | **1-yr ISP Effective Hourly (Linux)** | **3-yr ISP Effective Hourly (Linux)** |
| Single GPU VMs | g5.12xlarge | 4 | $5.672 | $3.403 | $2.269 |
| g5.16xlarge | 1 | $4.096 | $2.458 | $1.638 |
| g6.8xlarge | 1 | $2.014 | $1.249 | $0.856 |

**Note:**

If we wanna use locally deployed llm then we have to use compute with GPU accelerator. Above there are some ec2 computes on which we can deploy our model. To use the LLM model we need to use GPU accelerated computes.

We can’t use AWS fargate(serverless) as it does not support GPU accelerators.

We may also require other aws services like load balancer , autoscaling for scalability

**Alternatives using AWS bedrock services**

|  |  |  |
| --- | --- | --- |
| AWS BedRock Service Model on demand pricing | | |
| Model | Input | Output |
| GPT-4.1 | $2.00 / 1M tokens | $8.00 / 1M tokens |
| o4-mini | $1.100 / 1M tokens | $4.400 / 1M tokens |
| Claude 3.7 Sonnet | $0.003 /1k token | $0.015/1k token |
| Claude 3.5 Haiku | $0.0008/1k token | $0.004/1k token |
| Llama 4 Maverick 17B | $0.00024/1k token | $0.00097 /1k token |

### **Conclusion:**

**AWS Bedrock services** offer a more cost-effective and manageable solution compared to EC2 GPU instances. With token-based pricing, Bedrock is more affordable for scalable workloads, costing approximately **$300/month** for 30 million tokens, while EC2 instances incur higher costs and require complex management of infrastructure, scaling, and load balancing. EC2 also lacks the flexibility and scalability of Bedrock for variable workloads. Therefore, **AWS Bedrock** is the optimal choice for cost efficiency, ease of management, and seamless scalability in deploying LLM models.

Summary :

Deploying large language models (LLMs) using **AWS Bedrock** is significantly more cost-effective and operationally efficient than hosting them directly on **EC2 GPU instances**. While EC2 offers control and customization, the high infrastructure cost and management overhead make it less practical.

### **Cost Comparison:**

Let’s compare the monthly cost of running an LLM for a medium-scale workload: **30 million tokens/month** (20M input, 10M output):

**Option A: AWS Bedrock or GPT-4.1**

|  |  |  |
| --- | --- | --- |
| **Token Usage** | **Rate** | **Monthly Cost** |
| 20M Input Tokens | $2.00 / 1M tokens | $40 |
| 10M Output Tokens | $8.00 / 1M tokens | $80 |
| **Total** |  | **$120/month** |

#### **Option B: EC2 Hosting (g5.12xlarge with 4 GPUs)**

|  |  |
| --- | --- |
| **Usage Assumption** | **Value** |
| On-Demand Rate | $5.672/hr |
| Runtime (24/7) | 720 hrs/month |
| **Total Monthly Cost** | $5.672 × 720 = **$4,088/month** |

Even with a **3-year reserved instance plan**, the cost reduces to:  
> $2.269/hr × 720 = **$1,633/month\**

**2. Efficiency & Management Overhead**

|  |  |  |
| --- | --- | --- |
| **Feature** | **EC2 Hosted LLM** | **AWS Bedrock (e.g., GPT-4.1)** |
| **Setup Time** | High (manual configuration) | Zero (fully managed) |
| **Scalability** | Manual (Auto Scaling Groups, ELB) | Seamless auto-scaling |
| **Model Updates** | Manual | Automatic (new versions added) |
| **GPU Cost Optimization** | Requires active monitoring | Not needed |
| **Maintenance** | OS, drivers, model mgmt | None |

### **Security Considerations**

* **AWS Bedrock** and **OpenAI GPT APIs** (via Azure or OpenAI direct) offer **enterprise-grade security**:
  + **Data encryption at rest and in transit**
  + **VPC integration** (for AWS)
  + **No model training on your inputs by default**
  + **Fine-grained access control** via IAM or API keys
* **EC2-based deployments** require you to manage security controls yourself: firewalls, IAM roles, data handling, patching, etc.

### **Conclusion--**

### **Final Recommendation**

**AWS Bedrock or OpenAI’s GPT API** is the most **cost-efficient, secure, and scalable solution** for deploying and using large language models (LLMs). These managed services eliminate the complexity of infrastructure management, provide automatic scalability, and come with built-in enterprise-grade security.

In contrast, **EC2-based hosting is significantly more expensive and operationally intensive**. Beyond the high hourly cost of GPU instances, teams must also manage:

* **Load balancers** to distribute traffic
* **Query handlers or inference APIs** to serve the model
* **Auto-scaling groups** for elasticity
* **Monitoring, logging, and security** configurations manually

These components not only increase the **cost of ownership** but also introduce potential **operational risks and maintenance overhead**, making EC2-hosted deployments less practical for most use cases.

**Questions we have....**

**Question....**Could you please provide some sample prompts that can be performed over data.

**Question** ...Can you specify the types of data validation expected during the data cleaning process?

**Question**.....What frontend technologies should we leverage for displaying the visualizations? Should we go with React.js or consider other frameworks/tools?

**Question**.... What needs to be displayed on dashboards as metrics( Any specific visuals ).What are the codes and subcodes mean ?