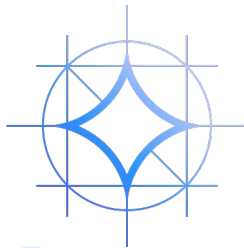




# Text-to-SQL

## Code gen vs Text gen models

CS 410/510: Large Language Models,  
Winter 2024



# Meet Our Team



Shreya Choure



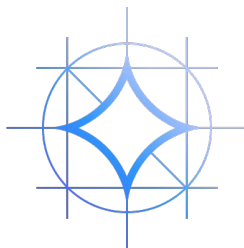
Upasana Chaudhari



Saurav Kumar Singh

# Agenda

- ❖ Problem Statement
- ❖ Motivation
- ❖ Dataset
- ❖ Models
- ❖ Evaluation
- ❖ Demo ( LLM Class dataset )
- ❖ Results and Analysis








# Problem Statement

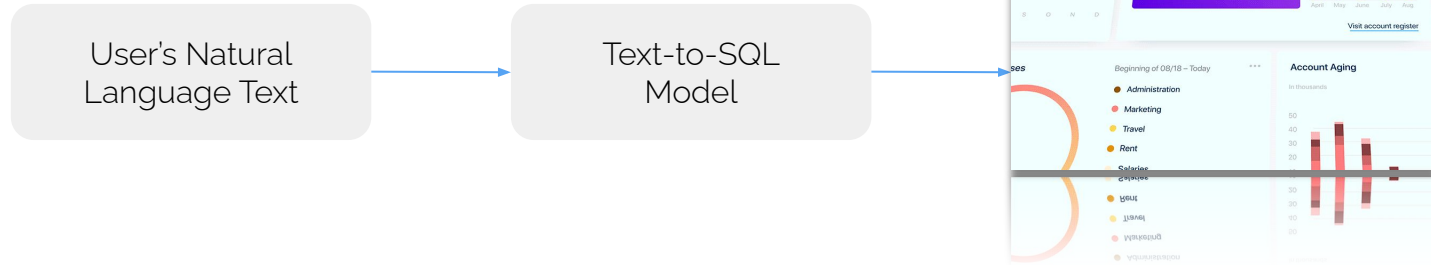


Investigate the performance of latest code and text generation models following parameter efficient supervised fine tuning (PEFT) on task of translating natural language questions into SQL queries, considering the provision of a given schema or context of the table.



# Motivation




- Improving model accuracy for better user experience
- Performance comparison of closed vs open LLM's.
- Analysis of generated query by code generation vs text generation models



Text to visualization

# Dataset Overview

- **sql-create-context**: This dataset builds from WikiSQL and Spider.
- 78.5K examples of natural language queries, SQL Query answering the question and CREATE statement as context (Schema of the database).
- Keeping text-to-sql LLMs in mind, intending to prevent hallucination of column and table names.
- **Train Dataset**: 55K
- **Test Dataset**: 1K

answer string · lengths	question string · lengths	context string · lengths
 342-396 <0.1%	 12 244	 27 489
<code>SELECT COUNT(*) FROM head WHERE age &gt; 56</code>	How many heads of the departments are older than 56 ?	<code>CREATE TABLE head (age INTEGER)</code>
<code>SELECT name, born_state, age FROM head ORDER BY age</code>	List the name, born state and age of the heads of departments ordered by age.	<code>CREATE TABLE head (name VARCHAR, born_state VARCHAR, age VARCHAR)</code>
<code>SELECT creation, name, budget_in_billions FROM department</code>	List the creation year, name and budget of each department.	<code>CREATE TABLE department (creation VARCHAR, name VARCHAR, budget_in_billions VARCHAR)</code>
<code>SELECT MAX(budget_in_billions), MIN(budget_in_billions) FROM department</code>	What are the maximum and minimum budget of the departments?	<code>CREATE TABLE department (budget_in_billions INTEGER)</code>

**src:** <https://huggingface.co/datasets/b-mc2/sql-create-context>

# Models Tested through

	Code Bison	Gemma	GPT 4 / 3.5
About	Code Bison is a model developed by Google aimed at enhancing code completion tasks.	Gemma is a series of open-source language models from Google, known for efficiency and high performance.	ChatGPT is a large language model chatbot developed by OpenAI, trained on massive amounts of text data.
Model Type	Code Generation	Text Generation	Text Generation
Availability	Closed	Open Source	Closed
Why Selected	Best code generation model by google and unexplored (since closed)	Latest model behind the flagship Gemini models	To validate earlier research results for the text-2-SQL task and compare the GPT model with latest models.



# Fine tuning proposed approach for Open Source



## Parameter Efficient Fine Tuning


- **Resource Efficiency:** enables fine-tuning with less computational resources and storage.
- **Performance Improvement:** efficacy natural language inference, question answering.
- **Adaptability and Generalization:** shown to be better than full fine-tuning.

## LoRA (Low-Rank Adaptation)

- Fine-tuning LLMs efficiently, focusing on text generation tasks.
- Offers cost savings, reduced latency.
- Addresses storage issues without compromising model quality.

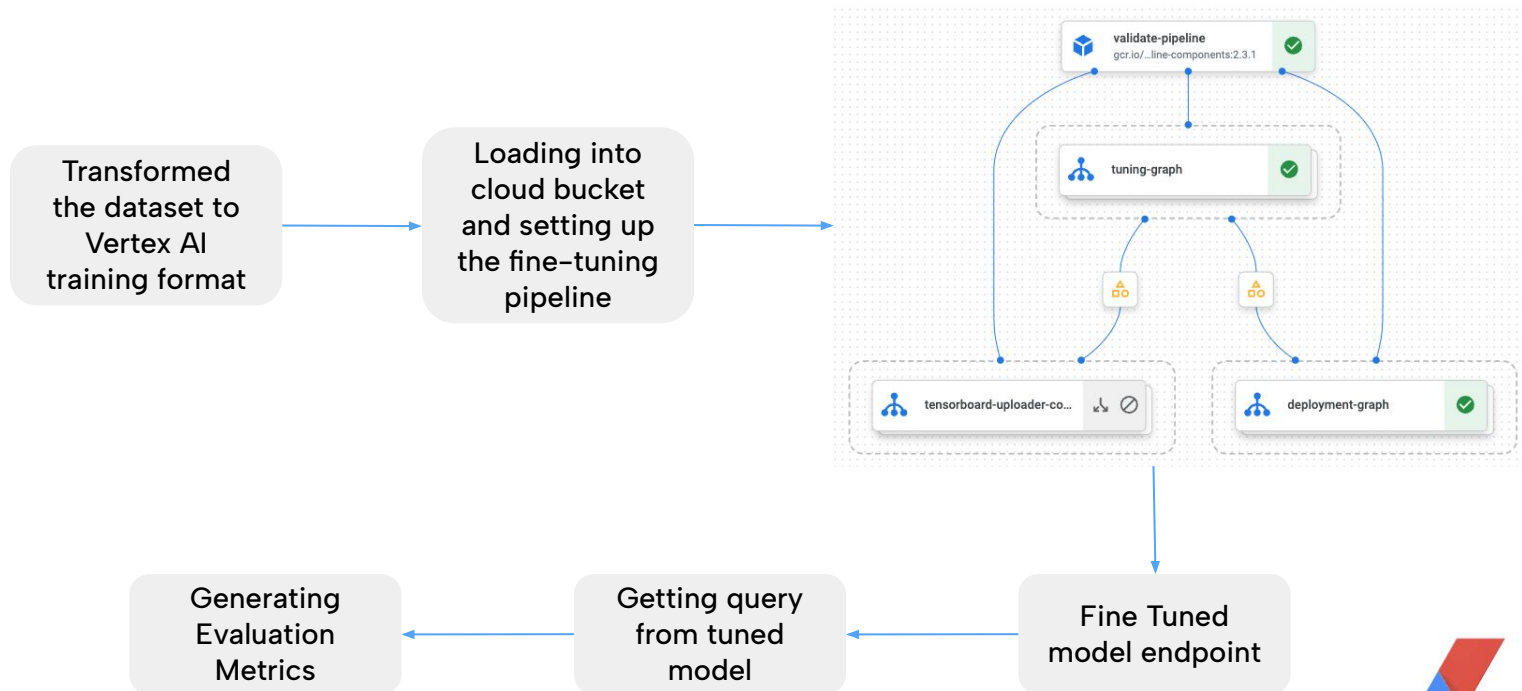


## QLoRA (Quantized Low-Rank Adaptation)

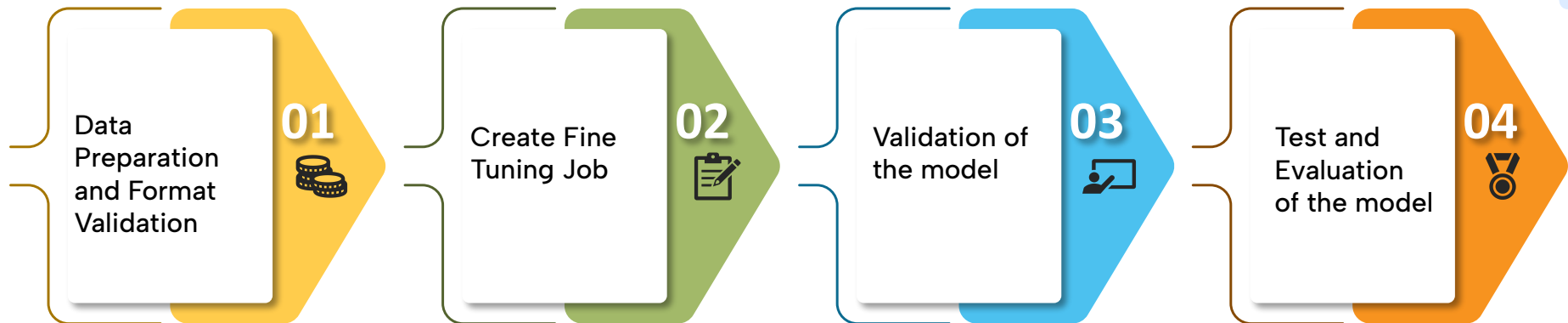
- Extends LoRA by adding quantization to reduce resources further while maintaining or improving task performance.
- 



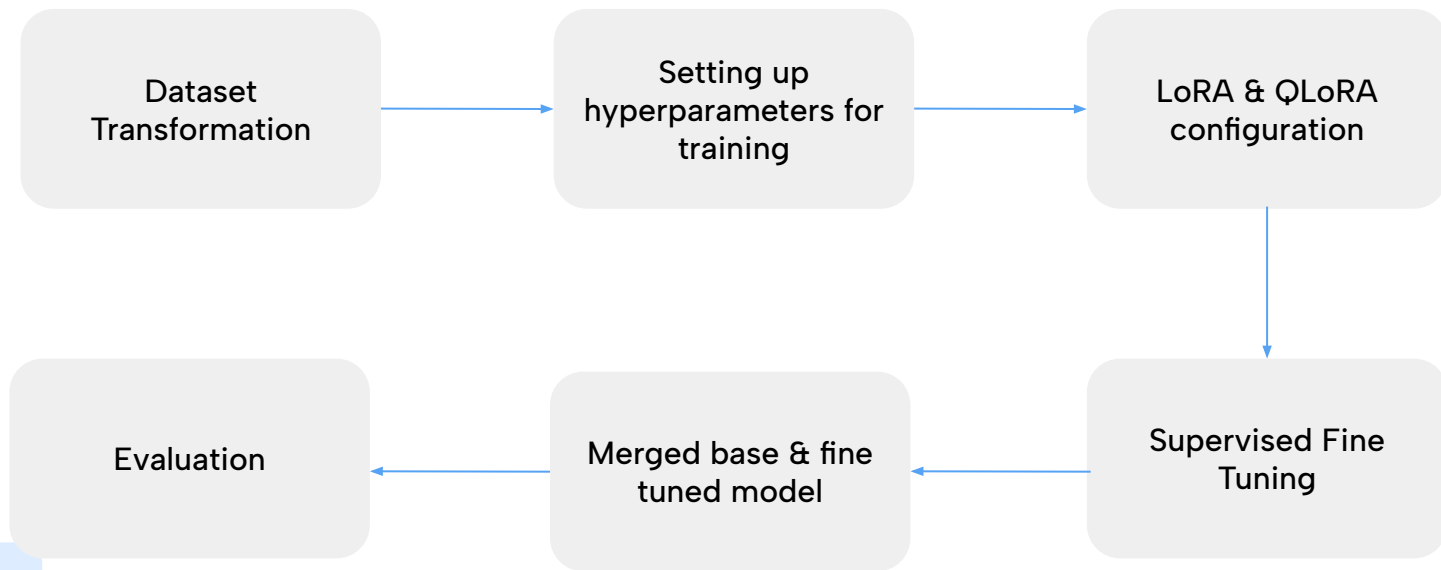
# Code Bison Fine tuning workflow



# GPT 3.5 turbo Fine tuning workflow



# Gemma Fine tuning workflow



**Transformed Data:** {'text': 'Question:\nwhat is the country with the album best of and weeks on chart is less than 5?\n\nContext:\nCREATE TABLE table\_name\_53 (country VARCHAR, album VARCHAR, weeks\_on\_chart VARCHAR)\n\nAnswer:\nSELECT country FROM table\_name\_53 WHERE album = "best of" AND weeks\_on\_chart < 5'}

# Prompt Engineering

**Code Bison**: prefix = "Give only the SQL query no extra text " +  
json\_data["input\_text"]

**Gemma**: You are a SQL expert. Generate SQL query considering the question(in natural language) and context(schema of the database). Strictly stop after generating the query.\nQuestion: {q}\nContext: {c}\nSQL Query:

**GPT 3.5**: Given the following natural language text, your job is to write a query based on the user's question:{context}  
\* \*Question: \*\* {question}

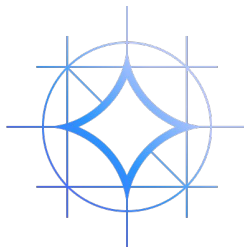
# Evaluation Metrics

**Rouge Score:** Evaluate text generation by comparing generated text against reference text.  
*Less common* in context of text to sql.

**Exact Match:** Evaluates if generated SQL query exactly matches the reference query.  
*Can be overly strict* in context of text to sql.

**Code Bleu:** Considers several aspects of code generation, such as syntax, semantics, and data flow.

*Designed specifically for code generation tasks.*



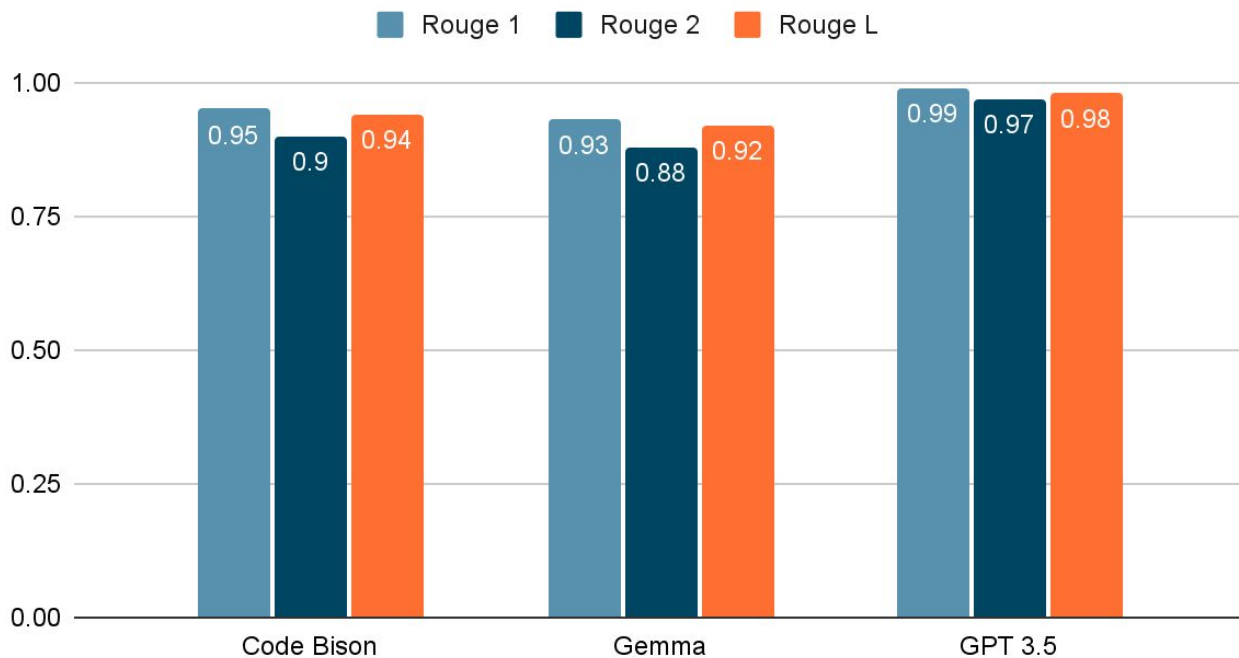
# Performance - Prompt Engineering

The below mentioned numbers indicate average scores evaluated on 1000 records from test dataset.

Model	Rouge Score	Exact Match	CodeBleu
Code Bison	Rouge - 1: 0.57 Rouge - 2: 0.30 Rouge - L: 0.56	0.001	0.88
Gemma - 2B	Rouge - 1: 0.67 Rouge - 2: 0.47 Rouge - L: 0.63	0.003	0.51
GPT- 3.5 turbo	Rouge - 1: 0.92 Rouge - 2: 0.86 Rouge - L: 0.90	0.00	0.39

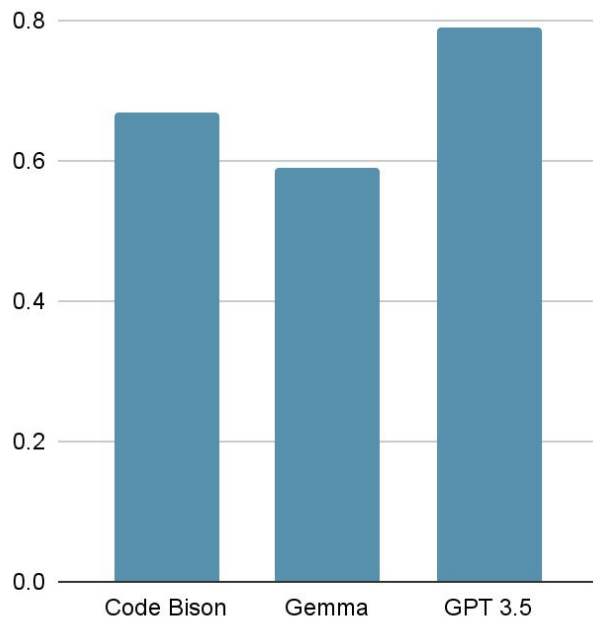
## Performance - Supervised Fine Tuning ROUGE Score

Fine Tuned Model Evaluation

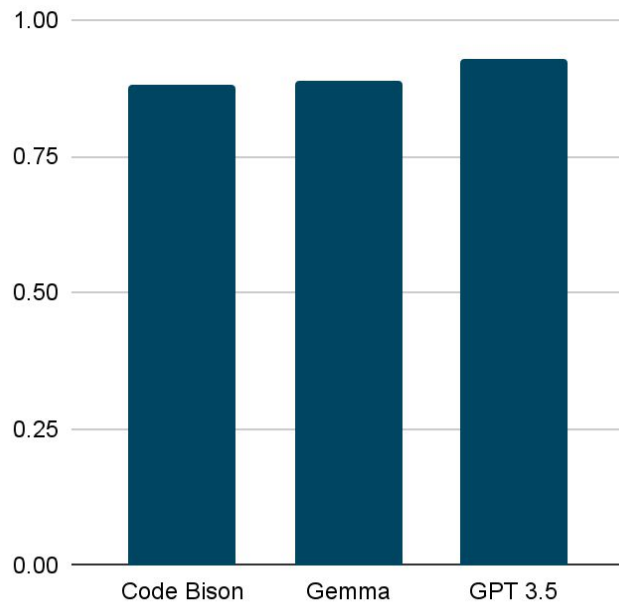


# Exact Match & Code BLEU

Exact Match



Code BLEU





# Demonstration

For demonstration purpose we created a table for how many **LLM classes** were online and how many of them were offline.

Query Query History

```
1 select * from course_schedule;
```

Data Output Messages Notifications

	day character varying	date character varying	format character varying
1	Tue	1/9	in-person
2	Thu	1/11	in-person
3	Tue	1/16	cancelled
4	Thu	1/18	online
5	Tue	1/23	online
6	Thu	1/25	in-person
7	Tue	1/30	online
8	Thu	2/1	in-person
9	Tue	2/6	online
10	Thu	2/8	in-person
11	Tue	2/13	online
12	Thu	2/15	online
13	Tue	2/20	online
14	Thu	2/22	in-person
15	Tue	2/27	online
16	Thu	2/29	in-person
17	Tue	3/5	online
18	Thu	3/7	in-person
19	Tue	3/12	online
20	Thu	3/14	online

# Demonstration

**Text:** Count number of online classes.

**Schema:** CREATE TABLE course\_schedule  
(day VARCHAR, format VARCHAR)

**Generated Queries:**

**Code Bison & Gemma:**

```
SELECT COUNT(*) FROM course_schedule WHERE format = "Online"
```

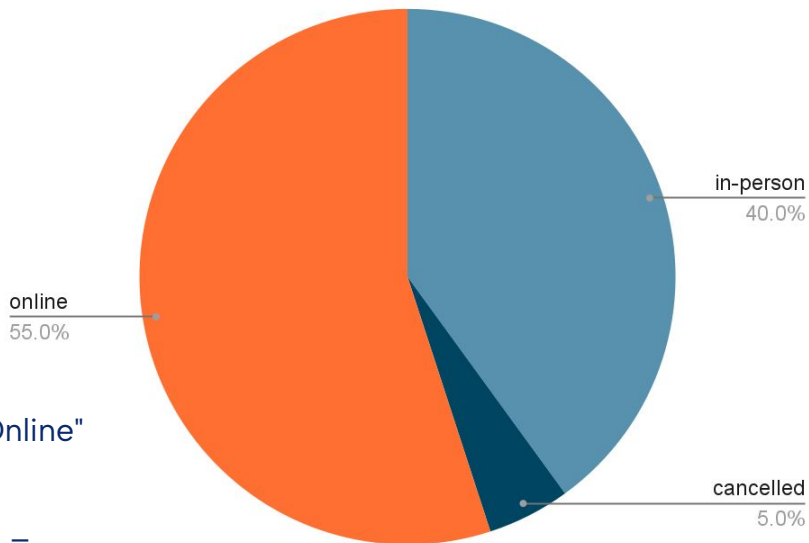
**GPT - 3.5- turbo:**

```
SELECT COUNT(*), format FROM course_schedule WHERE day =  
"online" GROUP BY format
```

**Correct Answer: 11**

**Our Answer: 11**

date



Query History

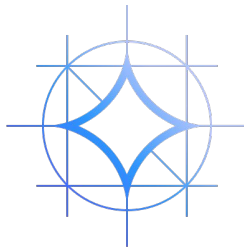
```
SELECT COUNT(*) FROM course_schedule WHERE format = 'online';
```

Data Output Messages Notifications

	count bigint
1	11

# Interesting Insights / Key Learnings




- Closed models outperform open-source models.
- There is a huge improvement in accuracy after fine-tuning.
- The exact match score is low because different SQL queries yield the same result.
- Models are almost 95% accurate for simple queries, but accuracy drops when JOINS, AGGREGATIONS, and SUBQUERIES are introduced.
- Generates context if not provided, not perfect but meaningful.





# Challenges



- Difficult to run Gemma – 2B on T4 GPU, requires more. A100.
  - Disk space ran out of memory if save\_steps parameters is less. (Can be used of analysing the model training)
  - Significant time to fine tune the model even with the most efficient PEFT (Parameter Efficient Fine Tuning) approach.
  - Since GPT & code bison is a closed model we do not have as much control over the specific way it generates SQL queries compared to building a model from scratch.
  - GPT and Code Bison models requires a paid subscription and are expensive.
- 
- 
- 

# Future Extensions

- Fine-tune for specific domains (finance, healthcare, legal) for better SQL accuracy.
- Ensure compatibility with various databases, including NoSQL.
- Automate understanding of database schemas to minimize manual setup.
- Improve NLU for complex queries with advanced features (nested conditions, aggregates, joins, subqueries).
- Introduce result visualization for easier data interpretation and analysis.





# Acknowledgement



**Special Thanks to Professor Ameeta  
and TA Rhitabrat.  
Also All Team Members.**





**Thank you !**

**Any Questions?**

