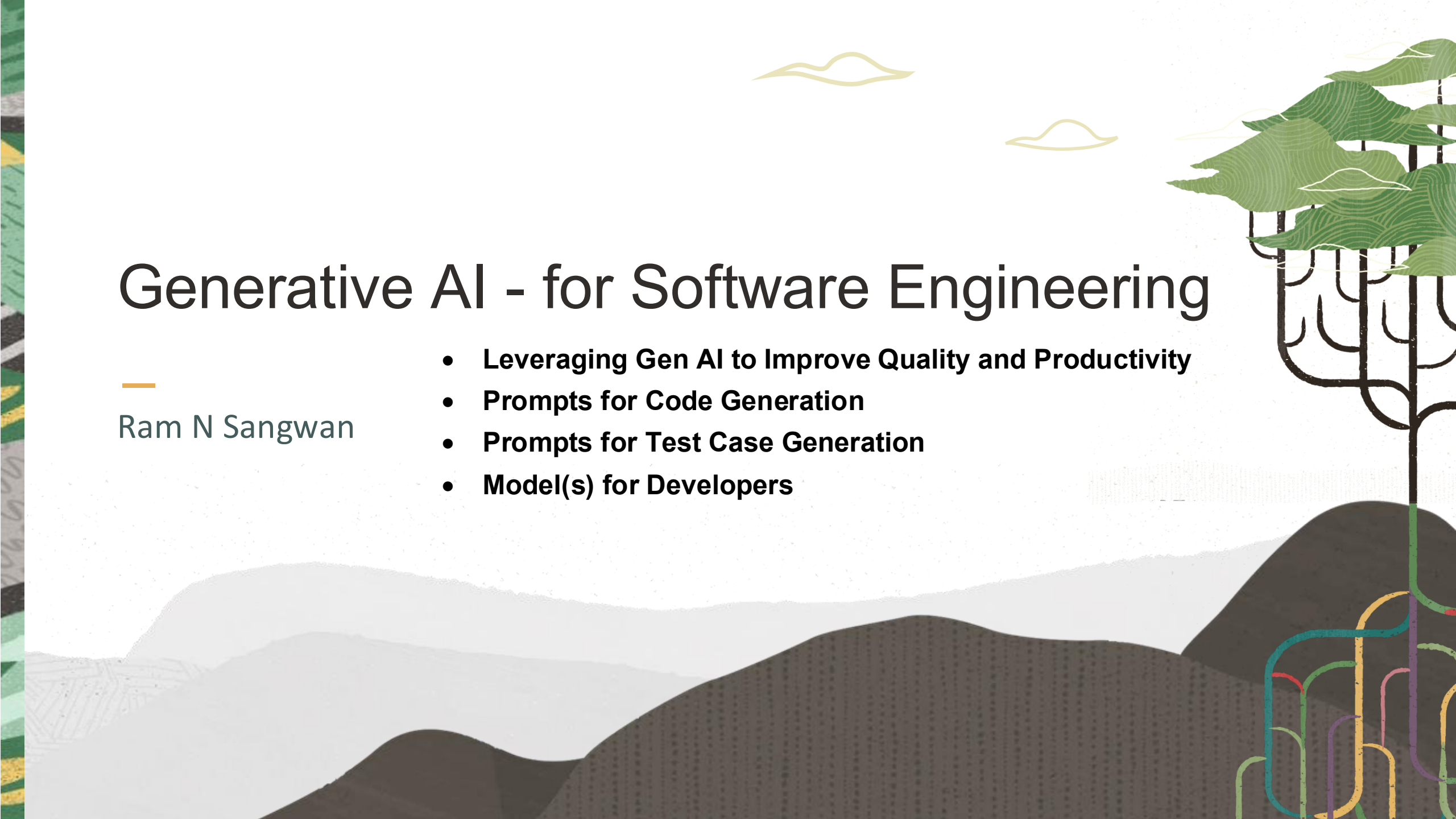
Generative AI - for Software Engineering

Ram N Sangwan

- **Leveraging Gen AI to Improve Quality and Productivity**
- **Prompts for Code Generation**
- **Prompts for Test Case Generation**
- **Model(s) for Developers**



How to Leverage Generative AI to Improve Quality, Productivity in software engineering

Main Areas of Impact

Research and Design

AI facilitates faster prototyping and iteration in the research and design phases, aiding in the development of innovative software solutions.

Code generation

Generative AI accelerates the coding process by automating repetitive tasks, leading to increased productivity and faster development cycles.

Security review and bug detection

By automating security checks and bug detection, AI helps fortify software against vulnerabilities and enhances overall quality.

Generative AI Can Assist With

- Automated Code Optimization.
- Code Review.
- Automated Refactoring.
- Style Consistency.
- Vulnerability Detection.

Leverage Gen AI to Improve Quality and Productivity

Specifically, generative AI can help software developers with

- **Automated Code Analysis**: GenAI can automatically analyze code to identify potential errors, bugs, or vulnerabilities.
- **Predictive Analytics**: By analyzing past data and patterns, generative AI can predict where bugs are most likely to occur and help developers proactively address them.
- **Real-time Monitoring**: Generative AI can monitor software applications in real-time, identifying and alerting developers to potential bugs or issues as they occur.

Considerations for Software Developers

Software development teams must consider:

- **Where generative AI best fits into the SDLC –** a clear roadmap for implementing AI is key.
- **How to train software developers –** It's important to educate your team on best practices for generating the desired results.
- **IP protection and security –** Sensitive data and proprietary information must be regulated internally.

LLMs for Developers

Model	Provider	Deployment	Core Strengths	Best Use Case
GPT-4.1 / GPT-4o	OpenAI	API (Hosted)	Best-in-class reasoning, multi-file refactors, tool/function calling	Autonomous coding agents, complex refactors
Claude 3.5 Sonnet	Anthropic	API (Hosted)	Large codebase comprehension, clean & maintainable code	Repo analysis, migrations, code reviews
DeepSeek-Coder v2	DeepSeek	Self-hosted	Code-trained at scale, strong algorithms, open weights	Private coding copilots, on-prem AI
Code LLaMA (34B)	Meta	Self-hosted	Stable, strong PHP/JS/Python support, fine-tunable	Enterprise self-hosted coding systems

Code Generation Prompts

Generating Code in Playground

- The Cohere Playground (Chat Mode) and the command model are used in these examples
- As with all command models from Cohere, you can use a Preamble to define the behaviour and format of the responses.
- We will use the following **Preamble**:

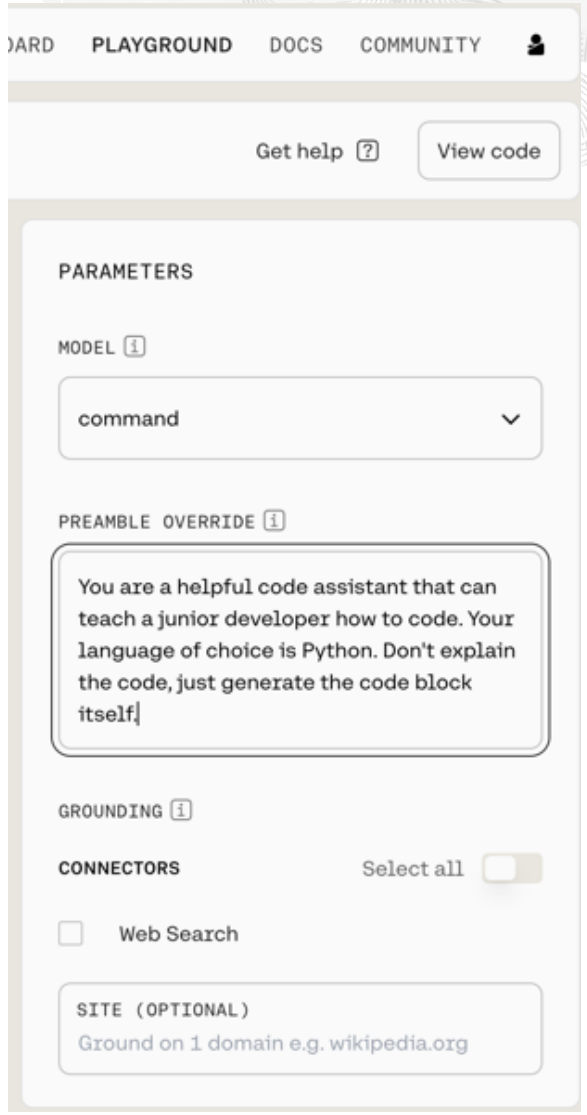
```
You are a helpful code assistant that can teach a junior developer
how to code. Your language of choice is Python. Don't explain the
code, just generate the code block itself.
```

- The User Message will then be the prompt itself with the instruction of the specific code you want the model to generate.

Basic Example

Here is the prompt if you want to try it yourself:

```
Write code that asks the user for their name and say "Hello"
```



Turn Comments Into Code

- You can also use the code generation capabilities of these LLMs to generate code from comments alone.

  Prompt:

  ```
  """1. Create a list of movies
  2. Create a list of ratings for these movies
  3. Combine them to make a json object of 10 movies with their ratings."""
  ```

Generate a Docker script

- The following prompt generates a Docker script to create a Linux machine with specific libraries installed:

  Generate a Docker script to create a simple linux machine that has python 3.10 installed with following libraries: pandas, tensorflow, numpy

```
Dockerfile
# Use an official Python runtime as a parent image
FROM python:3.10-slim-buster

# Set the working directory in the container to /app
WORKDIR /app

# Add the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir pandas tensorflow numpy

# Make port 80 available to the world outside this container
EXPOSE 80

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Code Generation Prompts

Generate a function

- The following prompt generates a function that inputs a year and determines if it's a leap year or not:

Write a function in c language that checks if a year is a leap year.

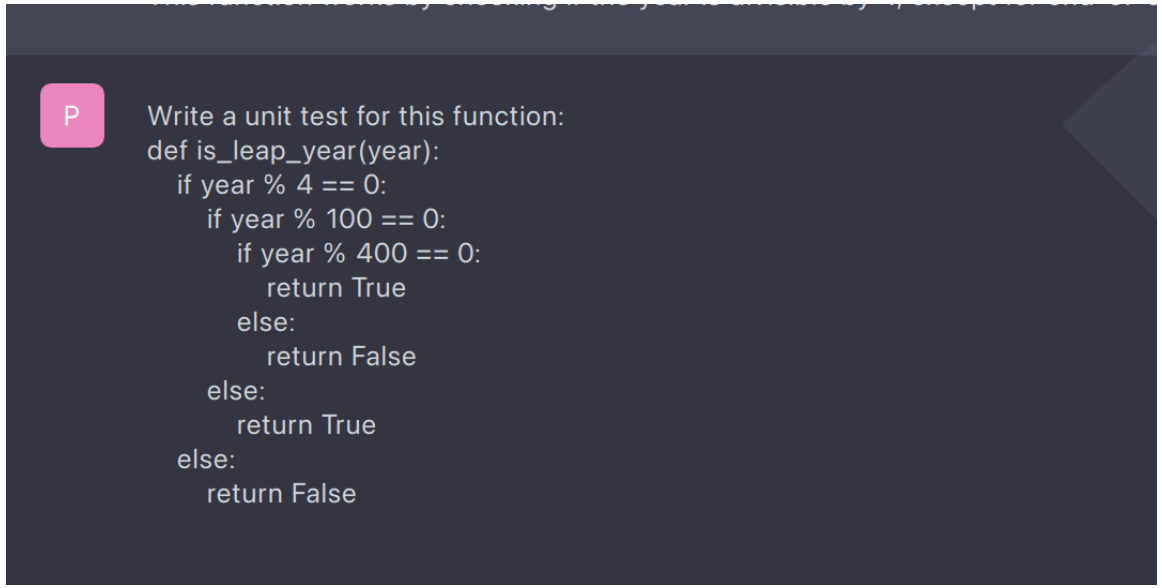
```c
#include <stdio.h>

// Function to check leap year
int isLeapYear(int year) {
    if (year % 4 == 0) {
        if (year % 100 == 0) {
            // year is divisible by 400, hence the year is a leap year
            if (year % 400 == 0)
                return 1;
            else
                return 0;
        }
        else
            return 1;
    }
    else
        return 0;
}

int main() {
    int year;
    printf("Enter a year: ");
    scanf("%d", &year);

    // Check if year is leap year
    if (isLeapYear(year)) {
        printf("%d is a leap year.", year);
    }
    else {
        printf("%d is not a leap year.", year);
```

Generate a unit test

You can use the code generation model to generate a unit test.

- The following prompt generates a unit test for a function that determines if a year is a leap year or not.

```
Write a unit test for this function:
def is_leap_year(year):
  if year % 4 == 0:
    if year % 100 == 0:
      if year % 400 == 0:
         return True
      else:
         return False
    else:
      return True
  else:
     return False
```



Generate a unit test - Response

python

```python
import unittest

def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False

class TestLeapYear(unittest.TestCase):

    def test_leap_year(self):
        self.assertEqual(is_leap_year(2000), True)
        self.assertEqual(is_leap_year(1900), False)
        self.assertEqual(is_leap_year(2004), True)
        self.assertEqual(is_leap_year(2001), False)
        self.assertEqual(is_leap_year(2100), False)

if __name__ == '__main__':
    unittest.main()
```

MySQL Query Generation

- Let's say you have a dataset with some information which you can include as part of the prompt and then instruct it to generate a specific query.

  For example:

```
"""Table departments,columns = [DepartmentId, DepartmentName]
Table students, columns = [DepartmentId, StudentId, StudentName]
Create a MySQL query for all students in the Computer Science department""”
```

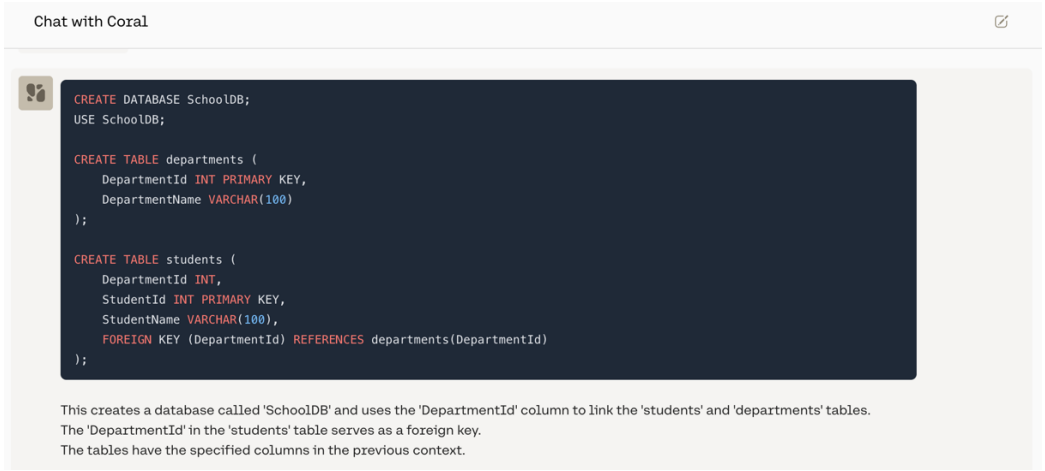
```sql
SELECT s.StudentId, s.StudentName
FROM students s
JOIN departments d ON s.DepartmentId = d.DepartmentId
WHERE d.DepartmentName = 'Computer Science'
```

Example 2: Enum

Let's do some more parsing with it

- If you are testing this example, it could be useful to have this database created.
- One approach is using a modified version of the prompt above with slightly different instructions to generate a database schema:

```
Table departments, columns = [DepartmentId, DepartmentName]
Table students, columns = [DepartmentId, StudentId, StudentName]
Create a valid database schema with the above tables and columns
```

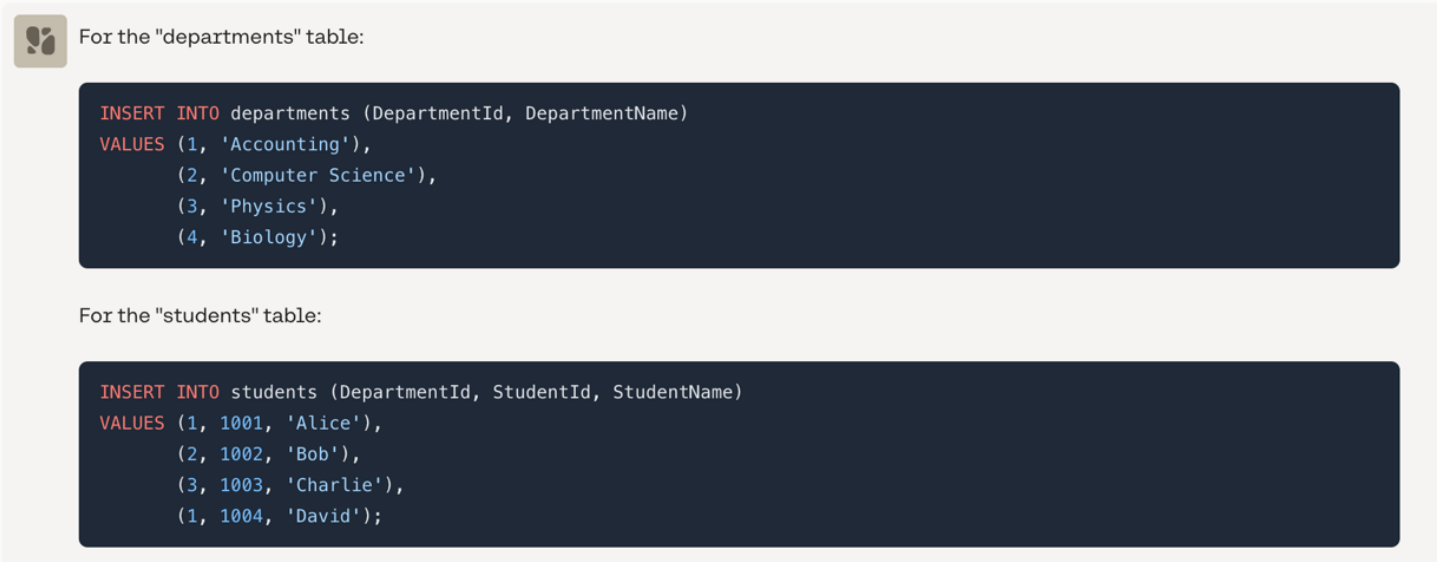


Add Dummy Data..

You will also need dummy data to test if the original generated query was valid. :

```
CREATE TABLE departments (  DepartmentId INT PRIMARY KEY,  DepartmentName VARCHAR(50));

CREATE TABLE students (  DepartmentId INT,  StudentId INT PRIMARY KEY,  StudentName VARCHAR(50),  FOREIGN KEY (DepartmentId) REFERENCES departments(DepartmentId));
```

Given the database schema above, generate valid insert statements include 4 rows for each table.



For the "departments" table:

```
INSERT INTO departments (DepartmentId, DepartmentName)
VALUES (1, 'Accounting'),
       (2, 'Computer Science'),
       (3, 'Physics'),
       (4, 'Biology');
```

For the "students" table:

```
INSERT INTO students (DepartmentId, StudentId, StudentName)
VALUES (1, 1001, 'Alice'),
       (2, 1002, 'Bob'),
       (3, 1003, 'Charlie'),
       (1, 1004, 'David');
```

Explain Code

- If you are learning to program in a certain language, it might be useful to prompt the model to explain certain bits of code.
- Let's reuse the query generated above and ask the model to explain it.
- If you are using the same PREAMBLE from before, be careful of how you are instructing the model.
- E.g. if we use the following example, the model **may** complain that it is a code assistant in Python but it still provides an explanation.

```
SELECT students.StudentId, students.StudentName
FROM students
INNER JOIN departments
ON students.DepartmentId = departments.DepartmentId
WHERE departments.DepartmentName = 'Computer Science';
Explain the above SQL statement.
```

More Tools for Developers

- **Understanding existing code**: AI2SQL can automatically write complex SQL queries, and SEEK can search and pull code from numerous data sources.
- **Code Generation**: Writing boilerplate code, repetitive functions, or tests will be massively accelerated by contextual tools.
  - Github Copilot is popular with >1M paying users, improving developer productivity by 10–50%.
- **UI Design**: Galileo AI translates natural language design prompts into design suggestions.
- **Documentation**: Mintlify can automatically document code within seconds in a standardized way.
- **Bug detection & vulnerability management**: Metabob lets you find and fix code faster than human software developers are able to do today.

Thank You