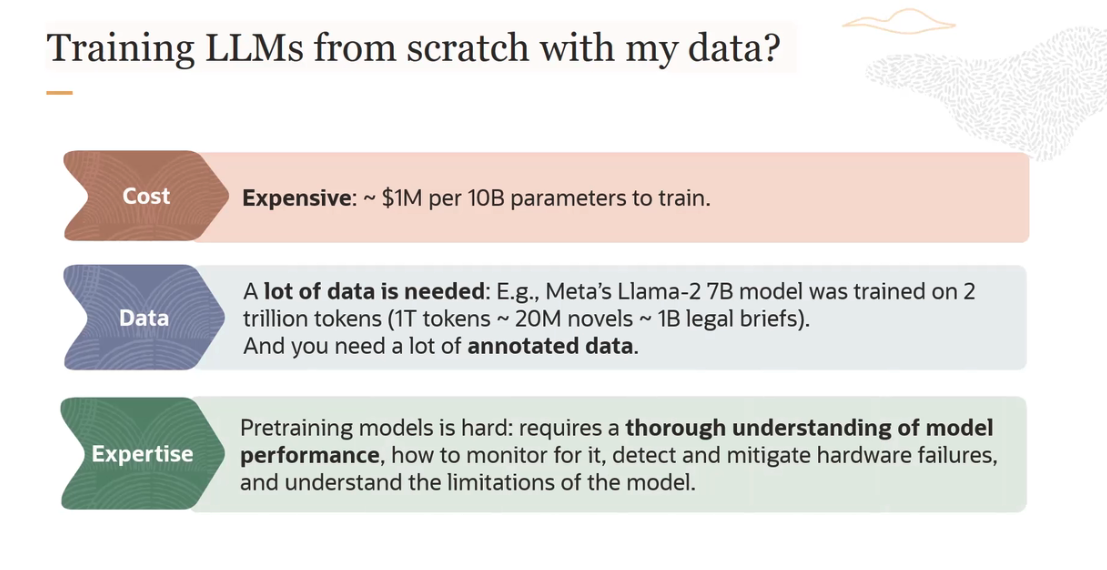
Training LLMs from scratch with my data?

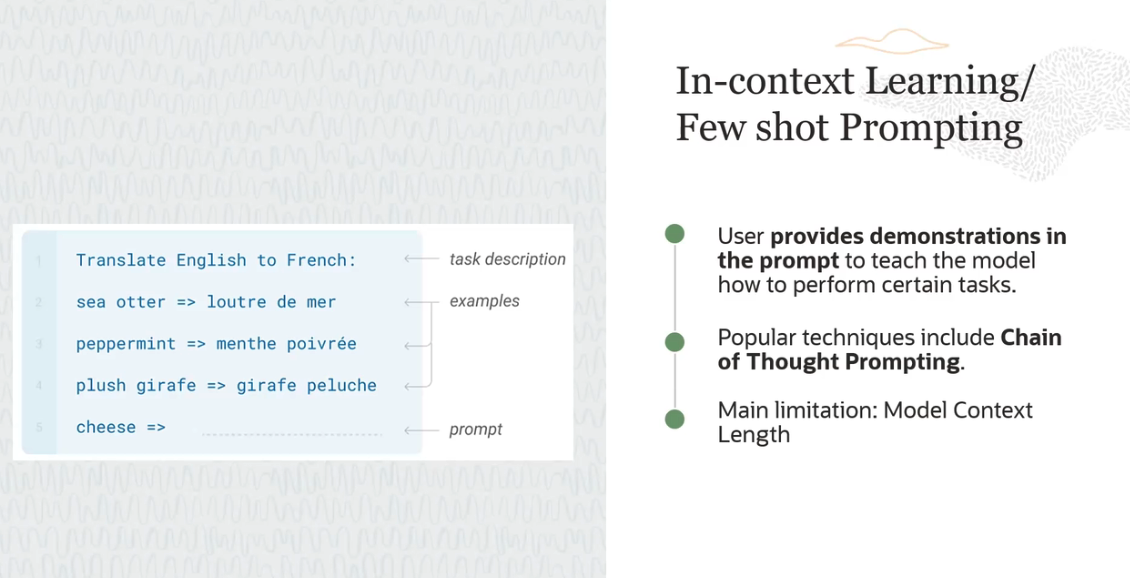
It's not a great idea to do so. Why? There are three main reasons.



What are the options?

There are 3 options to customize your LLMs.

**1. In-context/ Few shot prompting:**

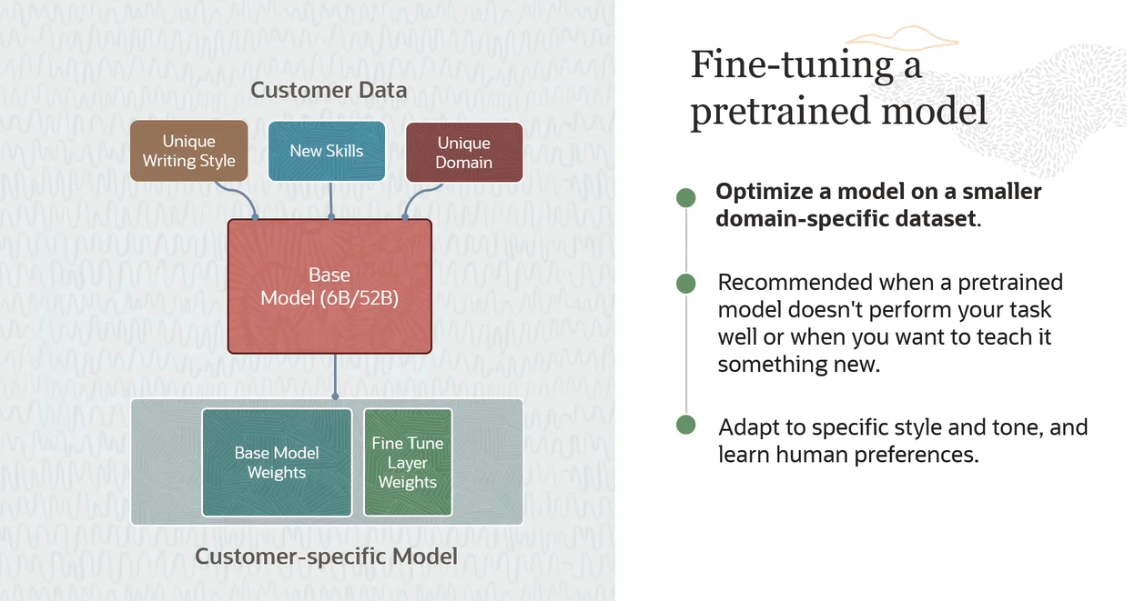


- **In-context/ Few shot prompting** - The basic idea is that user provides demonstrations in the prompt to teach the model how to perform certain tasks.

- Another technique which is used is called **chain of thought prompting**, where you are asking the model to break a problem into smaller chunks and solve each of these intermediate steps or each of these intermediate chunks.

- The main **limitation** here is the model context window and the length. Many of the models have length around 4,096 tokens or even smaller. And that is all the number of tokens a model can process at any given time. So this is the main limitation why you would not use few-shot prompting.

**2. Fine-tuning a pretrained model**

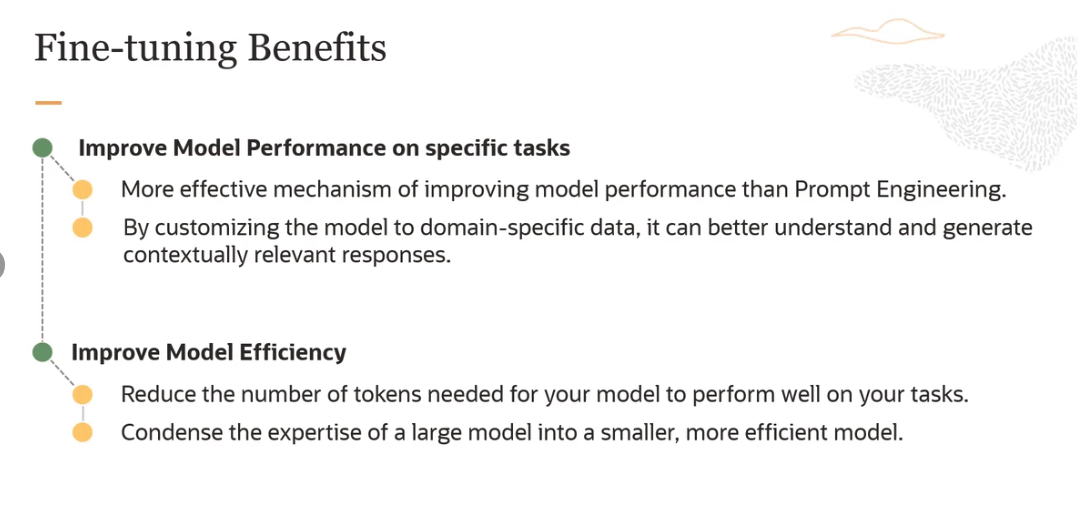


This is recommended when a pre-trained model doesn't perform your task well or when you want to teach it something new. And using fine-tuning, your model can adapt to specific style and tone and learn human preferences.

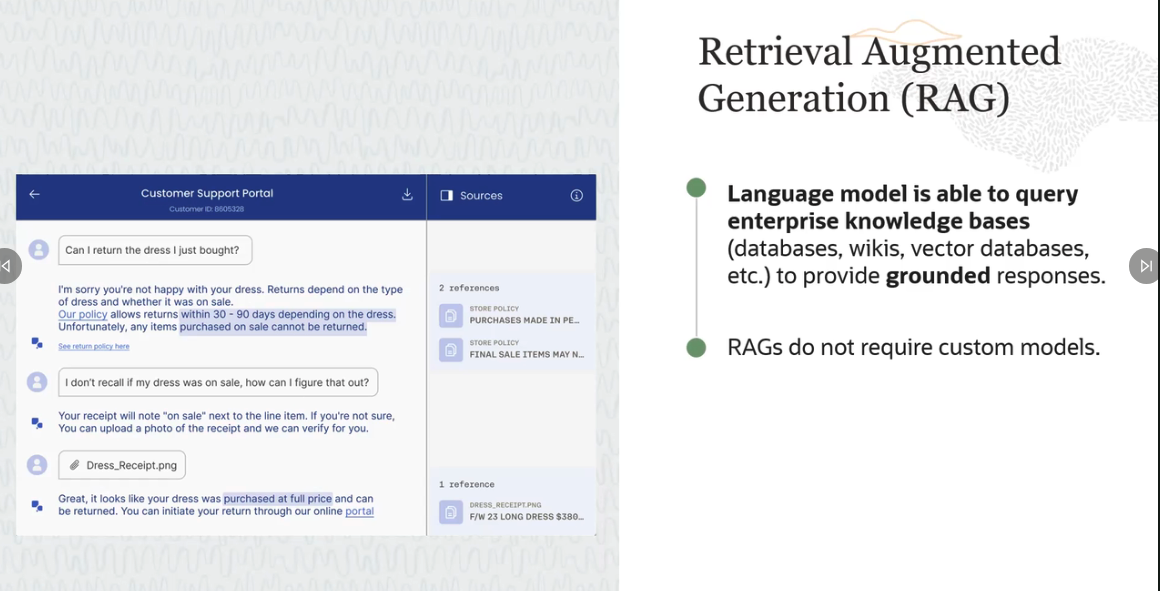
\*There are **two main advantages** of doing fine-tuning.

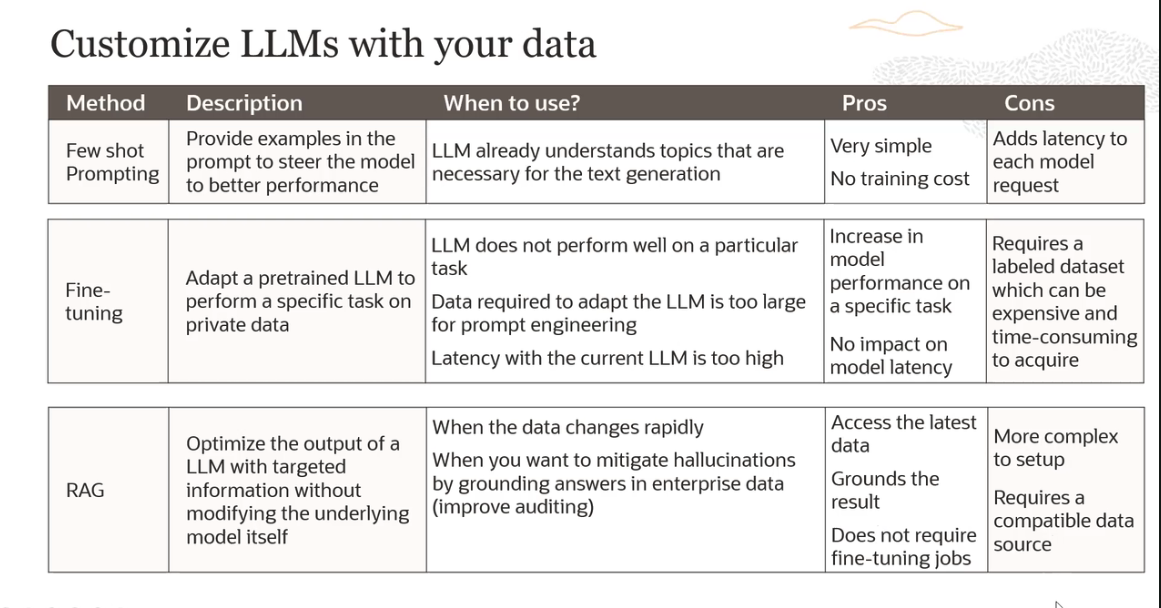
- The first one is you are improving the model performance on specific tasks.

- The second advantage is you are improving the model efficiency.



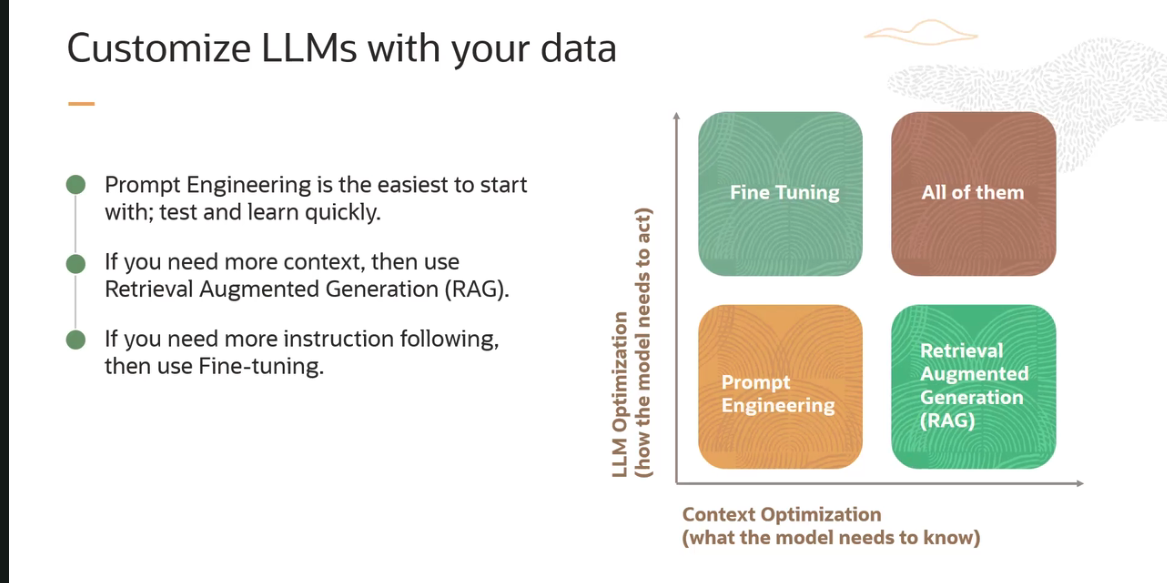
**3. Retrieval Augmented Generation (RAG)**

****

****

**HOW TO FIGURE OUT WHICH ONE TO USE?**

Two things to see Context optimization and LLM optimization



Let's look at what a typical journey would look like. You would start at the bottom corner here. You have a prompt, and then you create an evaluation framework, and you figure out what your baseline is. So that's where you start. Start with a simple prompt.

Then you can give some few shot examples of input/output pairs you want your model to follow. So still in prompt engineering, but you're giving some-- you're adding a few shot examples. Then you can add a simple retriever using RAG.

Now let us say you have these few shot examples increase your model performance. Now you hook your model to an enterprise knowledge base and create a RAG system. Now let us say that you are satisfied with your model, but the output is not coming out in the format or style that you really want.

So now you can take this model and you can fine-tune this model, which is built on RAG. And then it-- probably the output is in your style, but then maybe you figure out that once you have done that, the retrieval results are not that good and so you want to optimize the RAG system further. So you can actually go back and optimize your retrieval and so on and so forth.

You can see a pattern here where you are literally using all the different techniques, and it depends on your optimization journey when and which technique to use.

