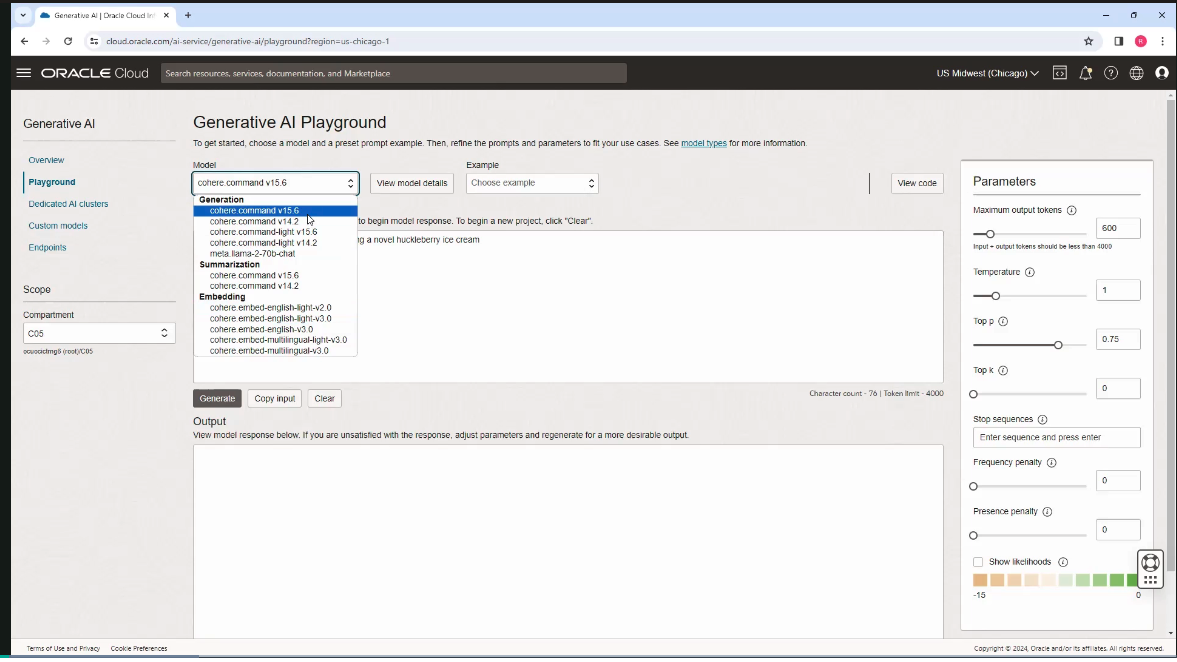
Demo : Generation Models - There are 3 different senarios

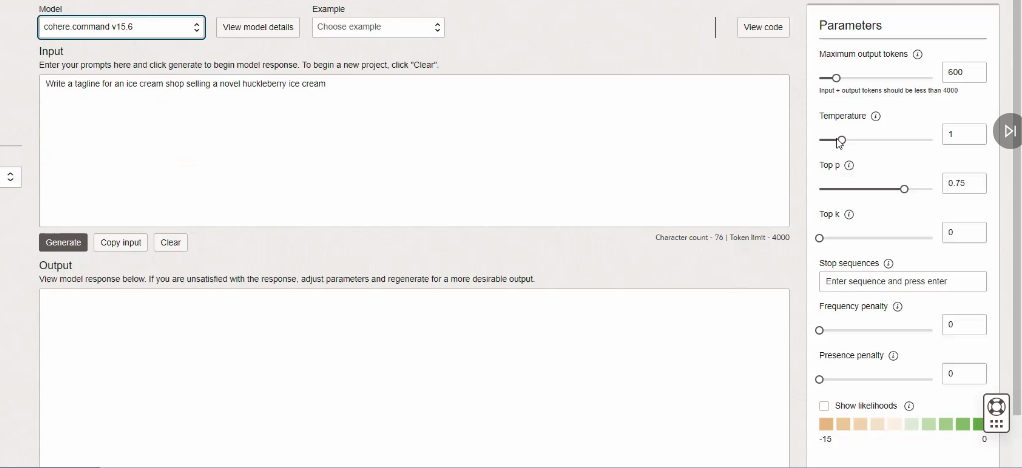
1. Text generation

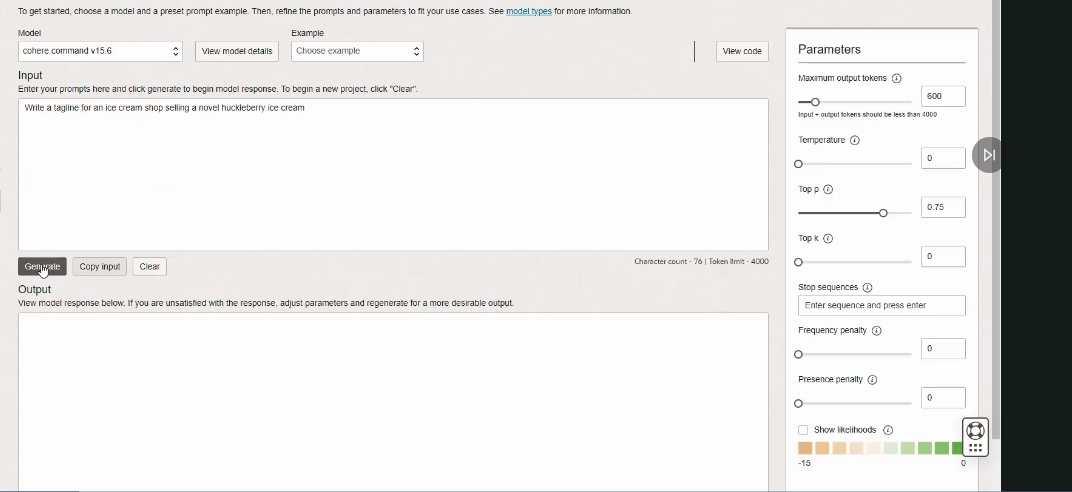
2. Data extraction

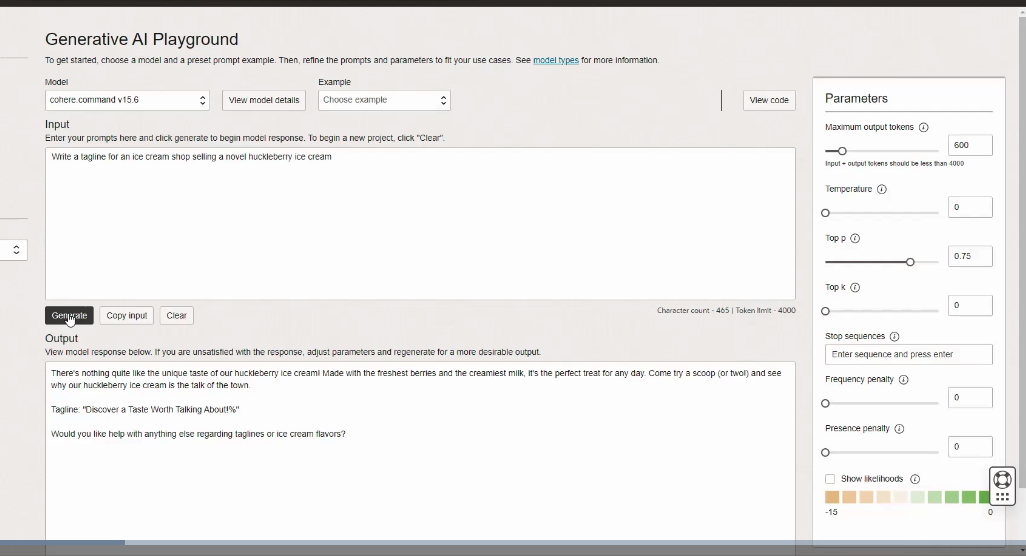
3. Text classification

1. First go to the palyground.







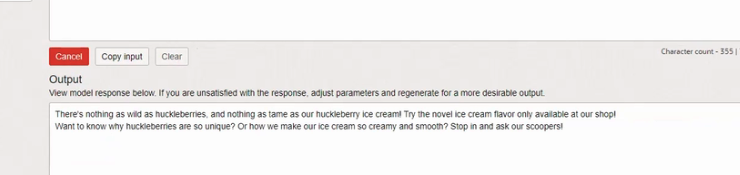


- Temperature is a parameter that controls the creativity or randomness of the text generated. A higher temperature results in more diverse and creative output, while a lower temperature makes it more deterministic.

- Temperature is affecting the probability distribution over the possible tokens at each step of the generation process.

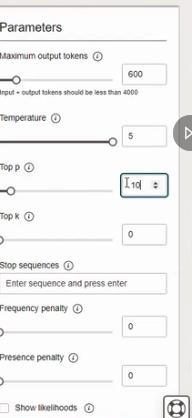
If temperature changed to 5. The output will be different as compared to zero temperature.



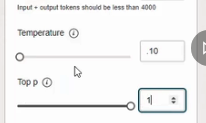


Another Important parameter is Top p which is an alternative to temperature.

For example, if we set top p as 0.10, then the model considers only the token that make up the top 10% of the probability of the next token.



In new tab change the temperature and top p values and keep the same prompt.

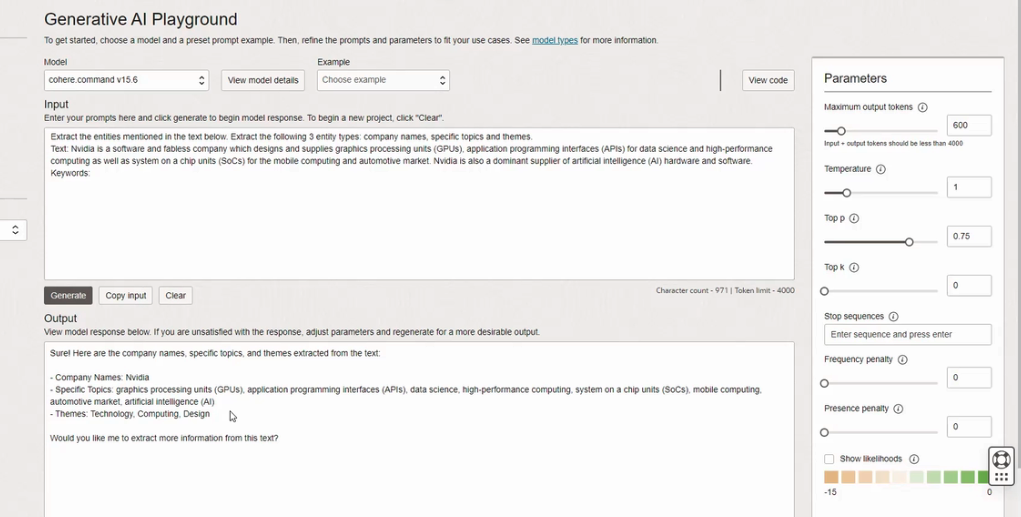


It will generate different responses for both.

**Both temperature and top p are powerful tools for controlling the behavior of large language models. You can use them independently or together when making these API calls.**

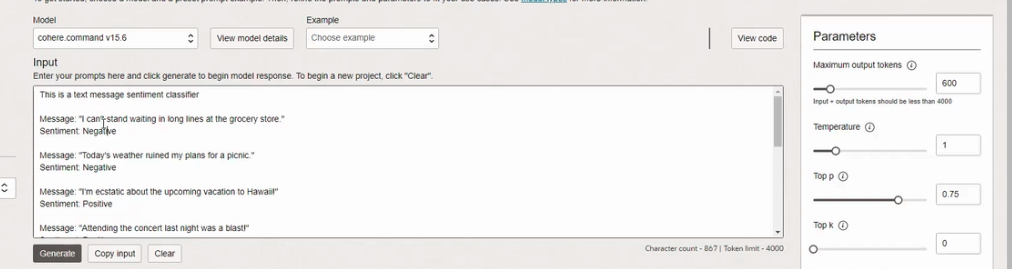
2. Data extraction: action. So in this case, asking the model-- providing the model a brief text. And asking it to extract the entities mentioned in the text below.

And specifically, asking for three entity types, company name, specific topics, and themes.

****

**You can use these generation models not only to generate text, but also extract data from text.**

3. Text Classification: Here, we are creating text classifier with the generation models. And we are providing a description of the task and providing a few examples.



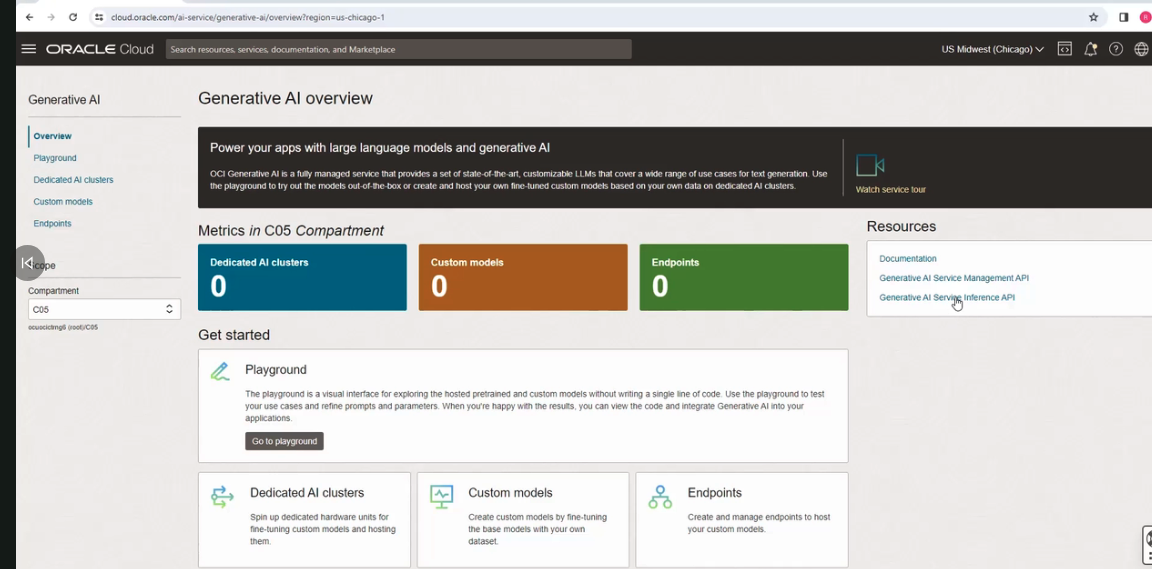
- Here showing the API how to classify the sentiment of text messages. The sentiment expresses the overall feeling or expression in the text.

- For the 10th one, we just have the text message. And we are asking the model to classify this text and provide us with the sentiment, whether it's positive or negative or neutral.

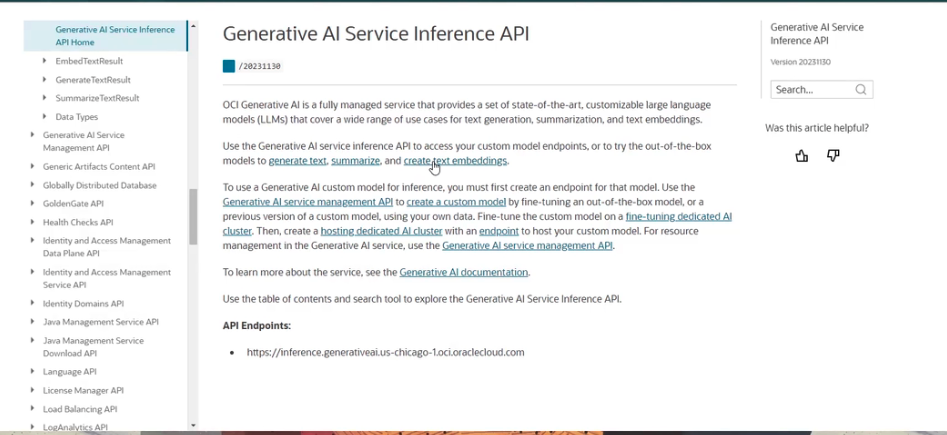




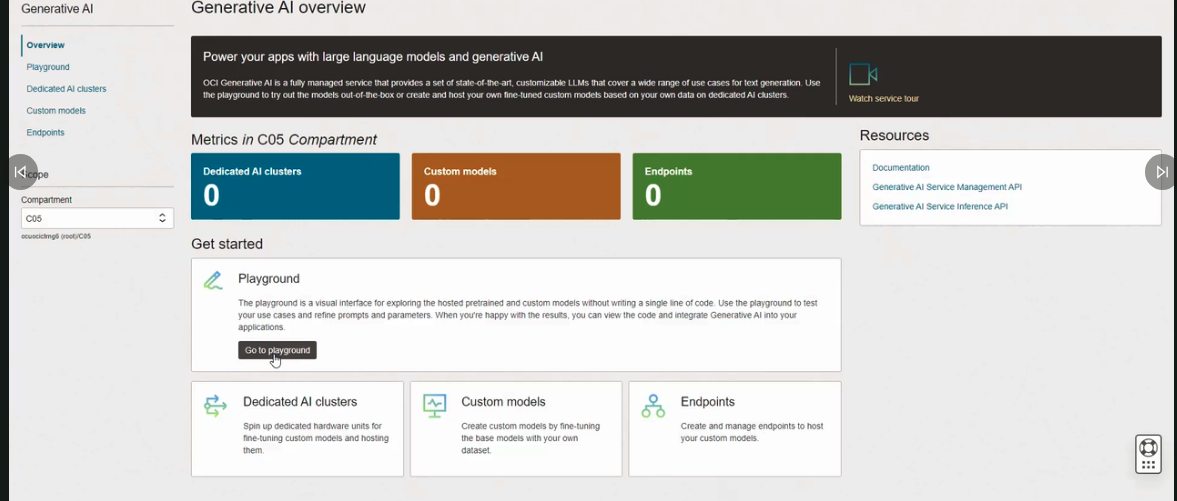




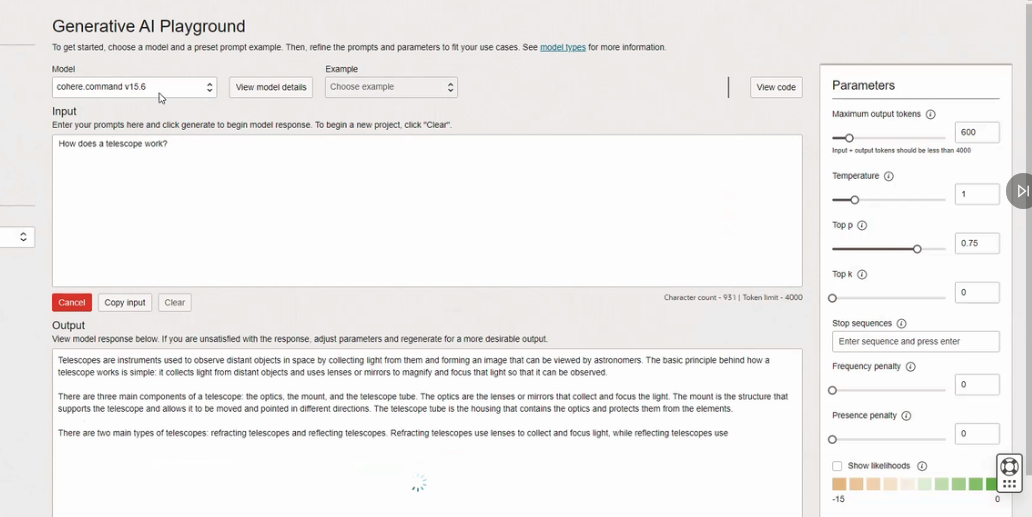
click it, you will get the documentation:



Go to playgroung.

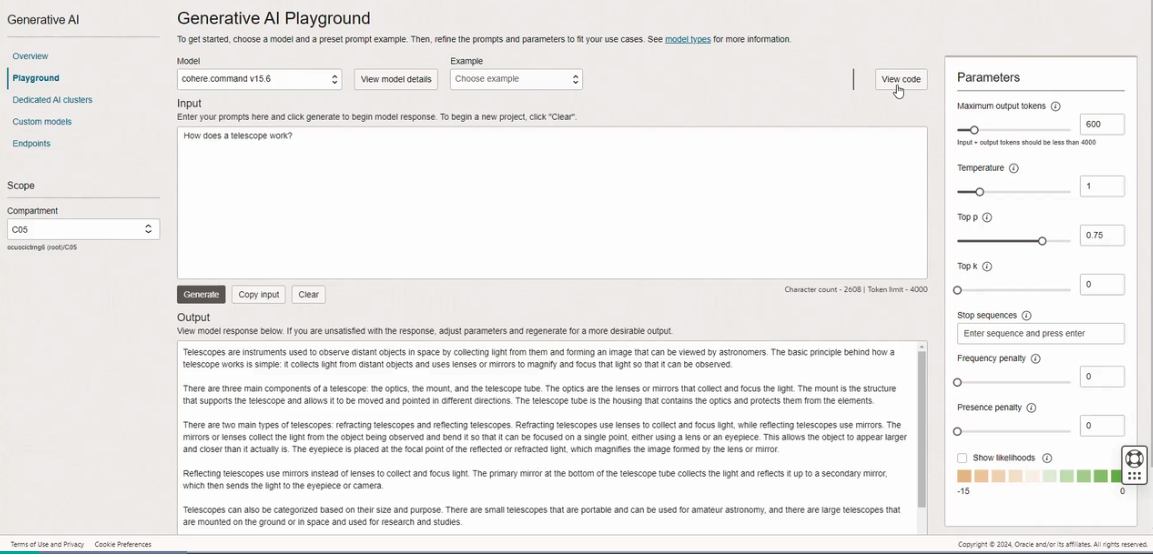


give simple prompt

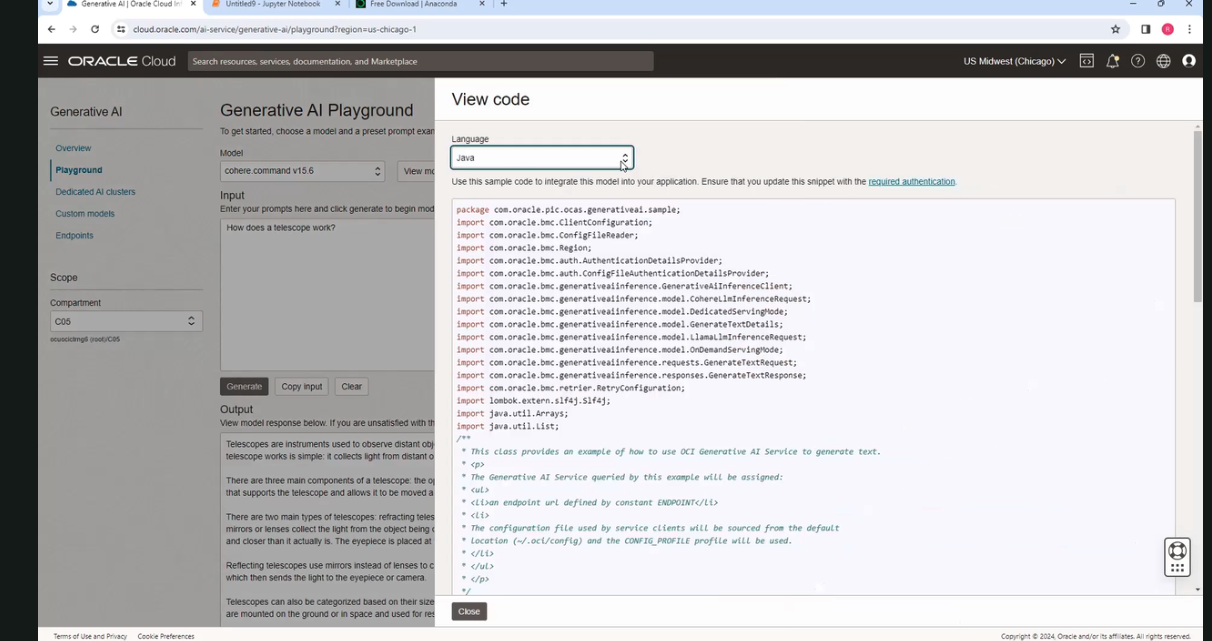


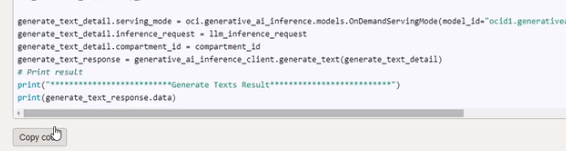
How does the API work? How can I write code to do the same activity?

Click on this button called View Code.

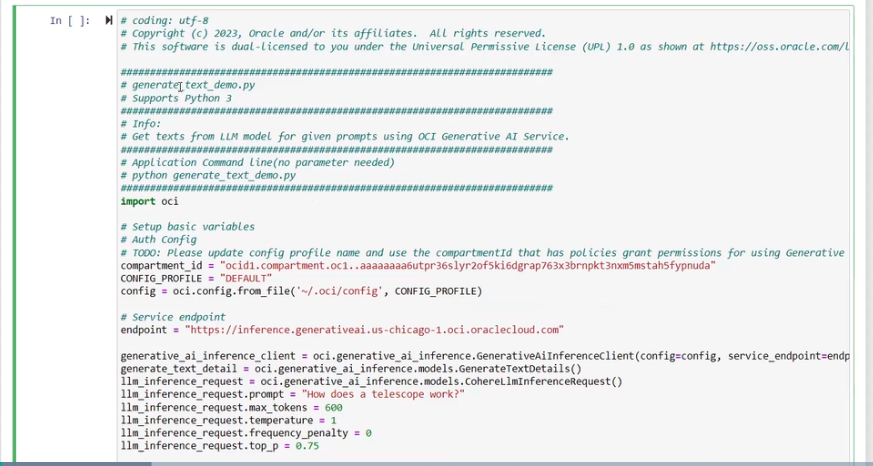


I can see the code for making this input output work in Java, as well as Python.



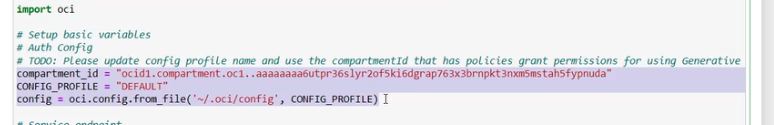


Copy and paste it in Jupyter Notebook - It lets you run the code in the browser, it sends it to a Notebook server that is running in the background. And it interfaces with whatever kernel you are running on your machine. So in this case, I'm running Python 3, which is the default. Now, the kernel will evaluate the code, and send the results back, which get displayed in the browser.)



How to execute this above code from OCI console to Jupyter

Click Shift-Enter and the code gets executed.



If you dont set the below mentioned 3 parameters the code wont work properly.

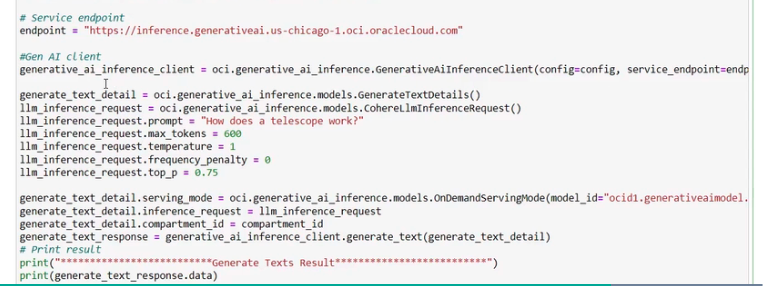
1. **Compartment\_id** - where you want this code to be executed. You're setting your config profile and config.

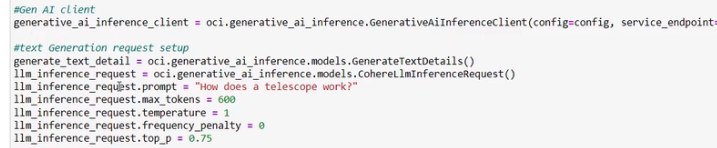
2. **CONFIG\_PROFILE** - Config profile is the name of the profile in your OCI configuration file that the script will use to authenticate and communicate with OCI services. you can use other profiles, if you have multiple OCI accounts or configuration. We are using default.

3. **config** - It is loading your OCI credentials and configuration settings from the specified profile, this default profile, in the configuration file, enabling the script to authenticate your request.

**Service endpoint** - This is the URL. This URL is the endpoint for the Generative AI service.

it's running in the US Chicago region. This is where the script is sending request to generate text. If you're using another region, you should replace US Chicago with that region name.



Below 3 lines are basically setting up the model and the compartment ID.



This sends the request to OCI generative AI service to generate text based on the provided details, and capture the response.

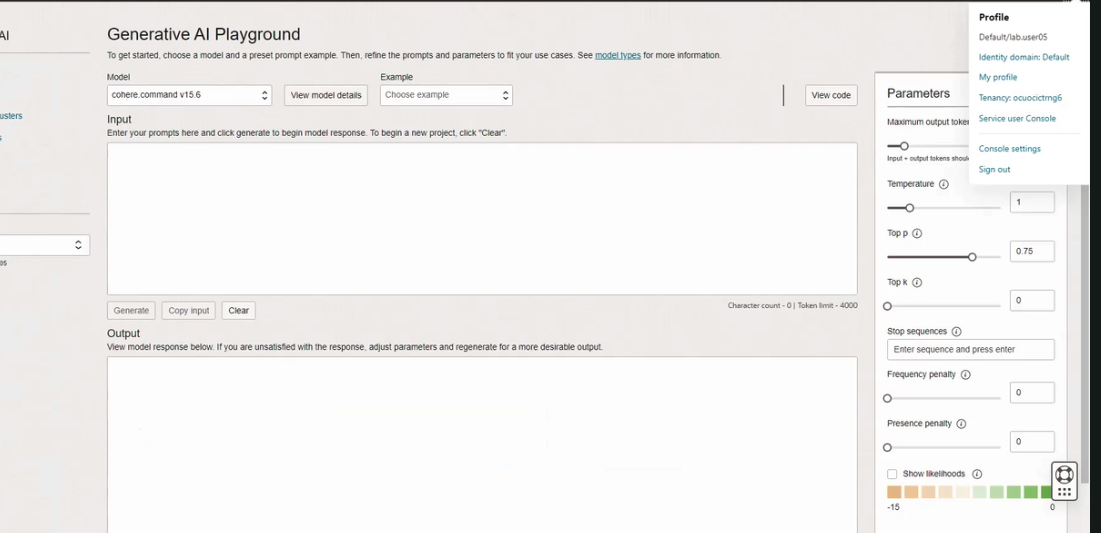


And this print statement here is basically printing the generated text to the console wrapped in a simple header footer for clarity.

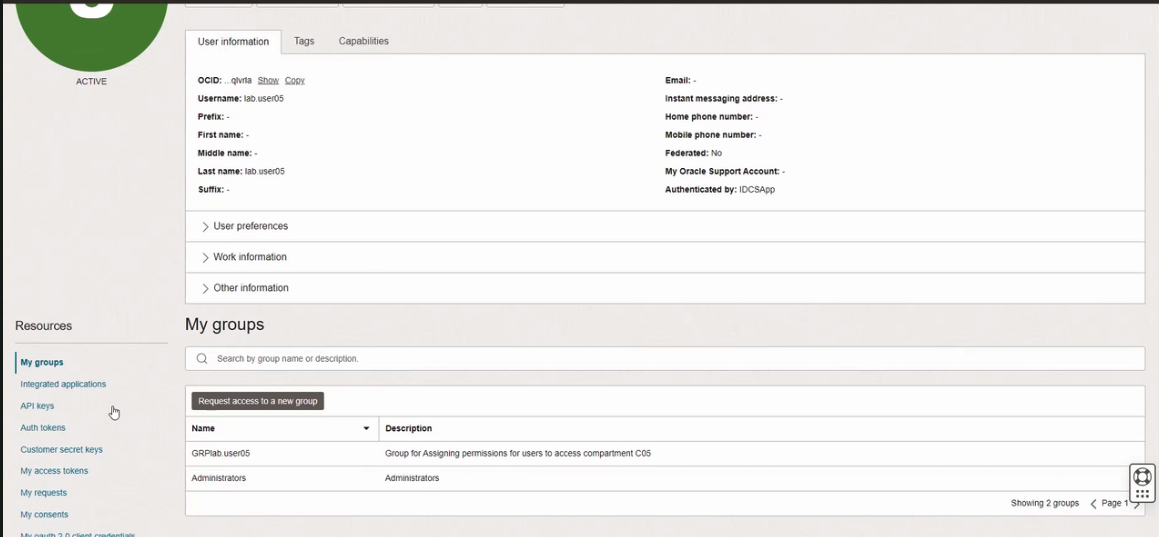




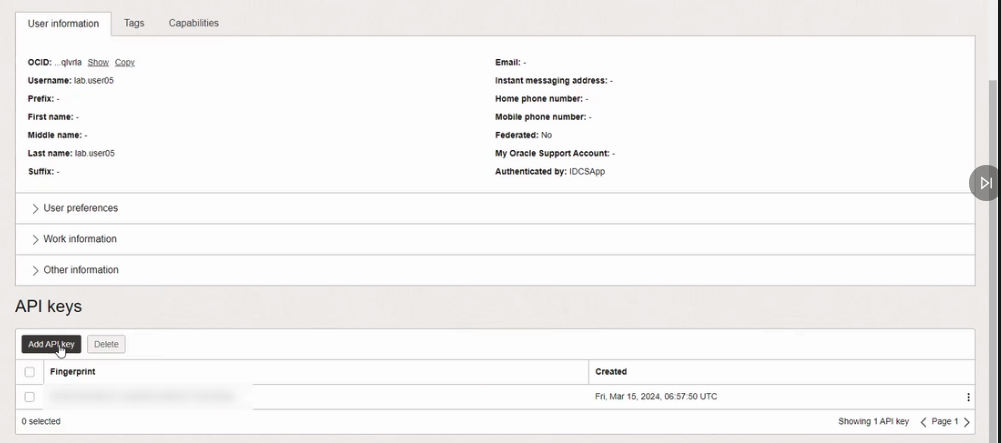
Go to playground,



Clickmy profile, it takes you to your profile.



if click on API keys,



generate new API key pair here. and click download private key. 

then click add.

