**Aim** :- Create, edit, and update the database using AWS DynamoDB along with global index using python SDK.

**Lab overview and objectives**

In this lab, you use Amazon DynamoDB to store and manage menu information. Using databases, such as DynamoDB, simplifies data management because you can easily query, sort, edit, and index data. You will use both the AWS Command Line Interface (AWS CLI) and the AWS SDK for Python (Boto3) to work with DynamoDB.

In upcoming labs, you will use application programming interface (API) calls from the café website to dynamically retrieve and update data that's stored in a DynamoDB table.

After completing this lab, you should be able to:

* Create a new DynamoDB table
* Add data to the table
* Modify table items based on conditions
* Query the table
* Add a global secondary index to the table

When you start the lab, the following resources are already created for you in the AWS account:

* VS Code Integrated Development Environment running on as EC2 instance

**AWS service restrictions**

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond those that this lab describes.

**Scenario**

The café website is up and running, and the café staff noticed a significant increase in new customer visits. Multiple customers also mentioned that it would be helpful if the website had an up-to-date menu. They could then use the menu to check the availability of food items before going to the café.

Frank and Martha ask Sofía to explore whether she can implement this feature for customers. Sofía is feeling more confident in her coding skills and has also been learning about different ways to store information in AWS. She knows that before they can dynamically update data on the website, she must first choose a data storage service to hold the data. She also needs to learn how to manage table data, load the product records, and create scripts to retrieve information from the data platform.

**A business request from the café: Store menu information in the cloud**

Frank and Martha mentioned to Sofía that they want the website to dynamically update its menu information. To prepare for this new functionality, Sofía decides to store this information in DynamoDB.

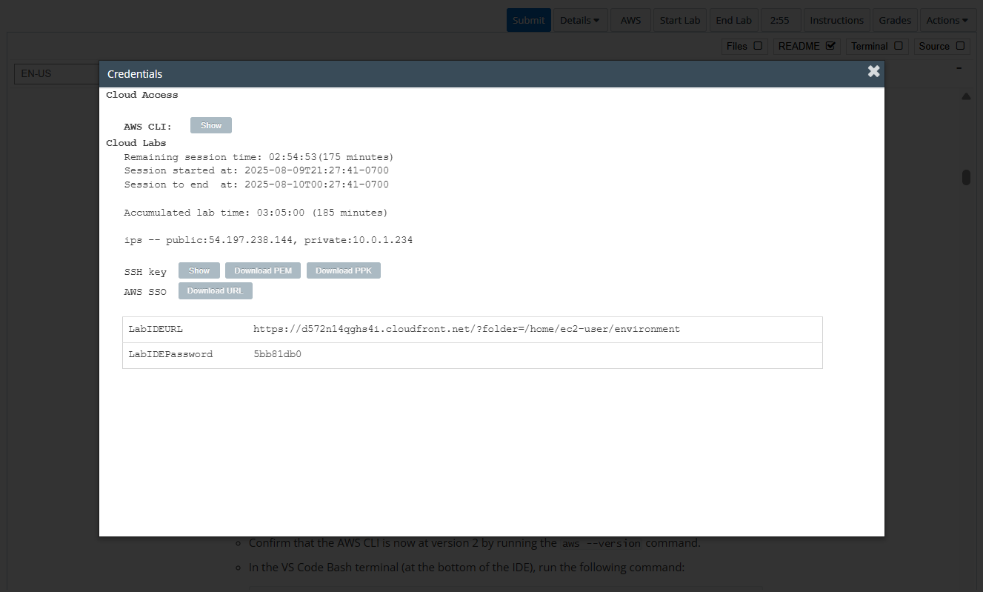
Café staff must be able to retrieve information from the table. Sofía decides to create one script that retrieves all inventory items from the table and another script (as a proof of concept) that uses a product name to retrieve a single record.

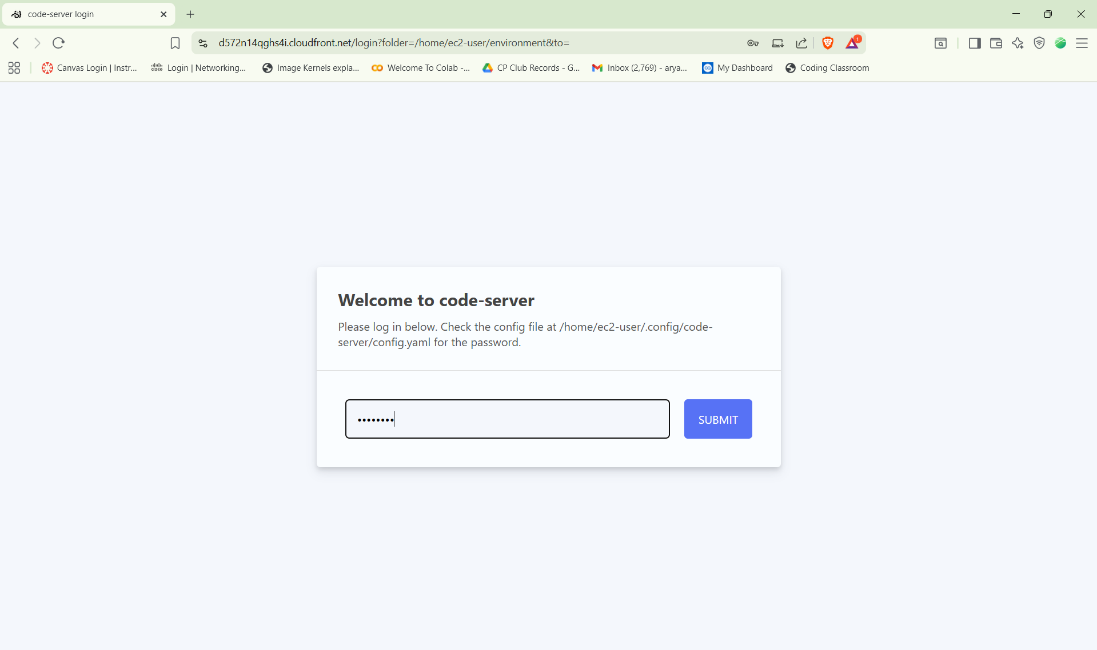
For this first challenge, you take on the role of Sofía. You use the AWS CLI and the SDK for Python to configure and create a DynamoDB table, load records into the table, and extract data from the table.

You will also connect to Visual Studio Code Integrated Development Environment (VS Code IDE) for running the python scripts and CLI commands.

**Task 1: Preparing the lab**

Connect to the VS Code IDE.

1. At the top of these instructions, choose Details followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
   1. **LabIDEURL**
   2. **LabIDEPassword**
3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPassword** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.

****

1. Download and extract the files that you need for this lab.

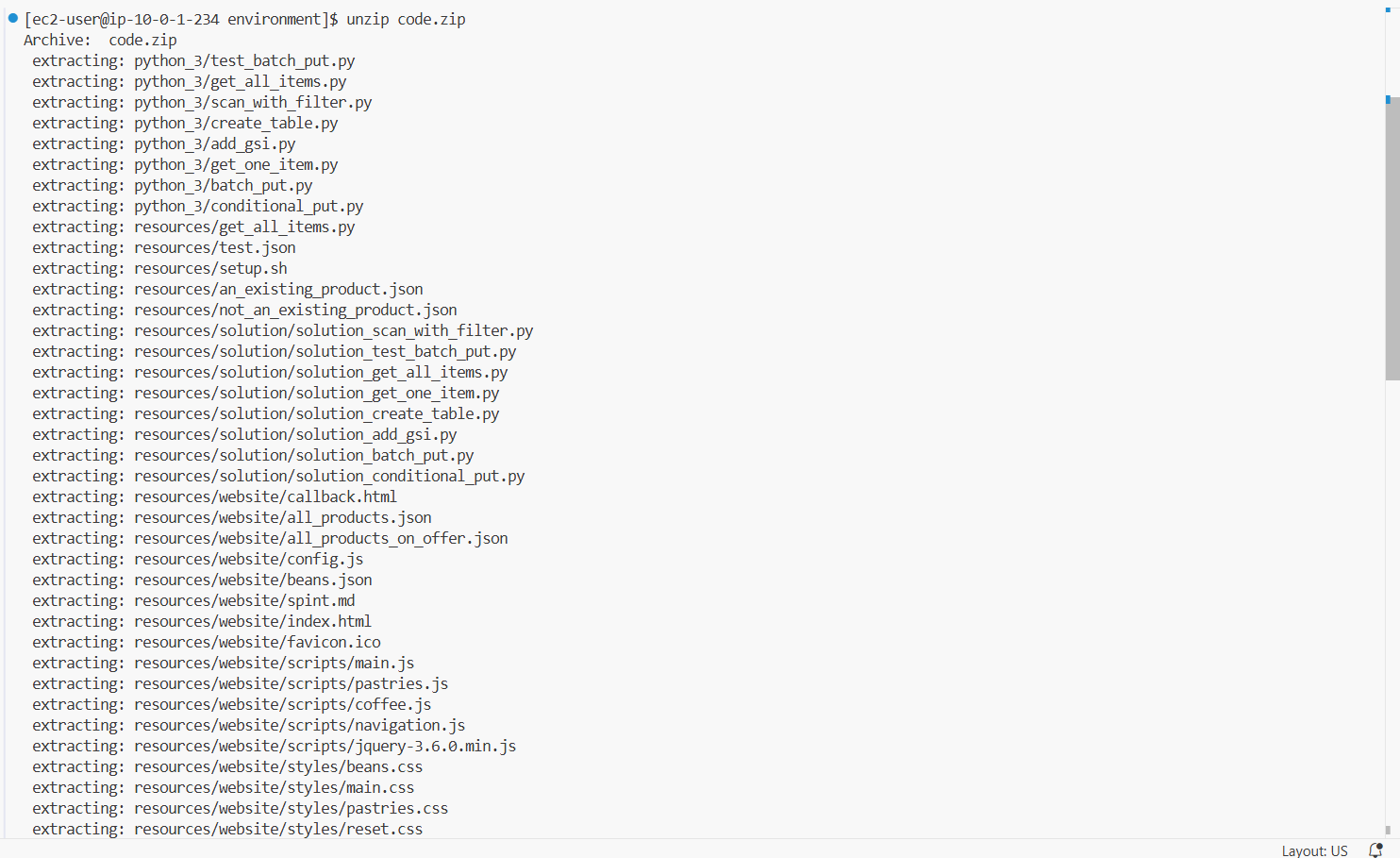
* In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/03-lab-dynamo/code.zip -P /home/ec2-user/environment

1. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.

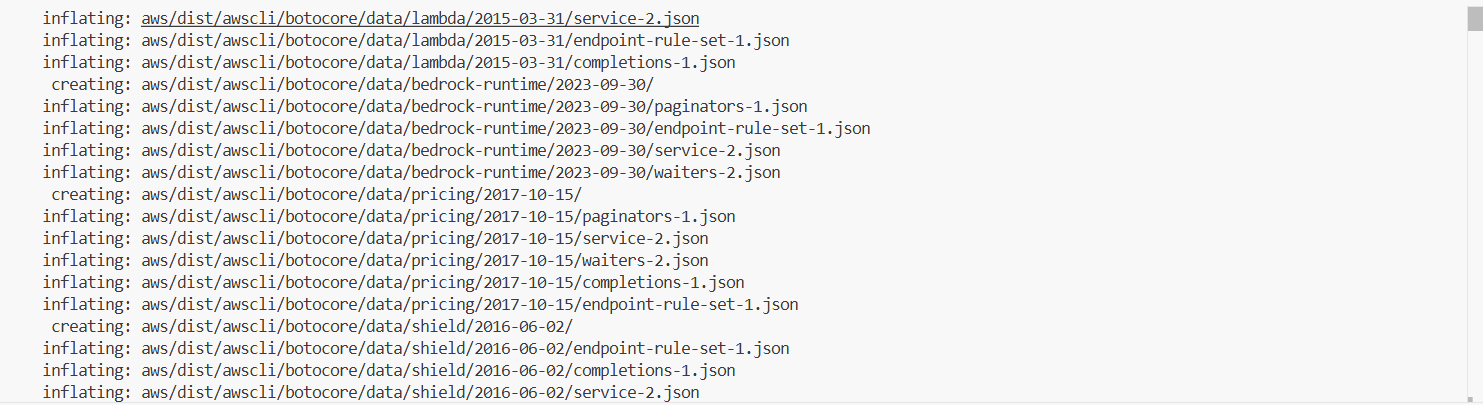
* Extract the file by running the following command:

unzip code.zip



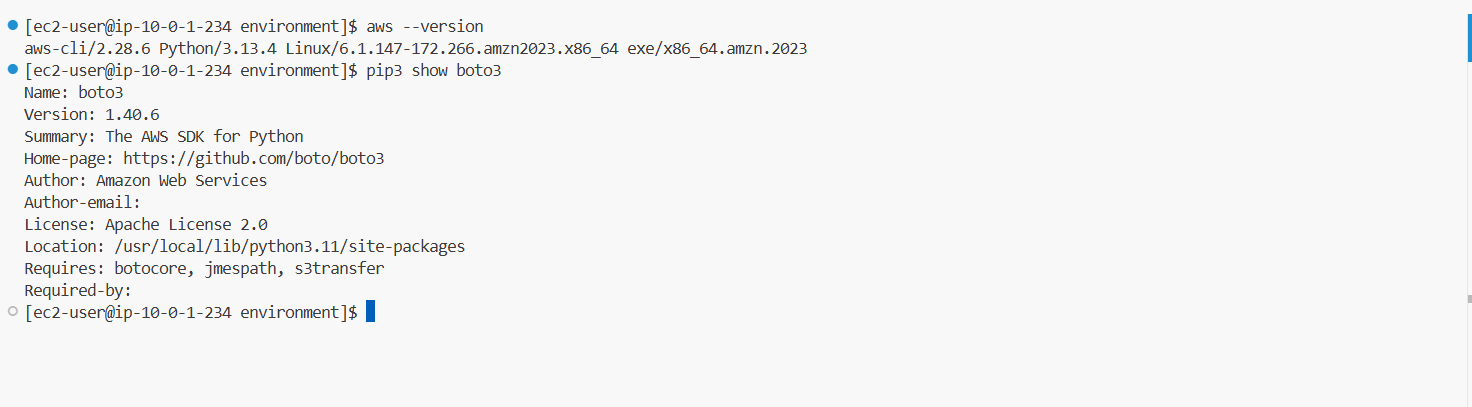
1. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.

* To set permissions on the script and then run it, run the following commands in the Bash terminal:

chmod +x ./resources/setup.sh && ./resources/setup.sh

1. Verify the AWS CLI version and also verify that the SDK for Python is installed.

* Confirm that the AWS CLI is now at version 2 by running the **aws --version** command.
* In the VS Code Bash terminal (at the bottom of the IDE), run the following command:

**pip3 show boto3**

**Task 2: Creating a DynamoDB table by using the SDK for Python**

To store and dynamically manage the café's menu items, Sofía decides to create a new DynamoDB table.

In this task, you take on the role of Sofía to create and define the new DynamoDB table.

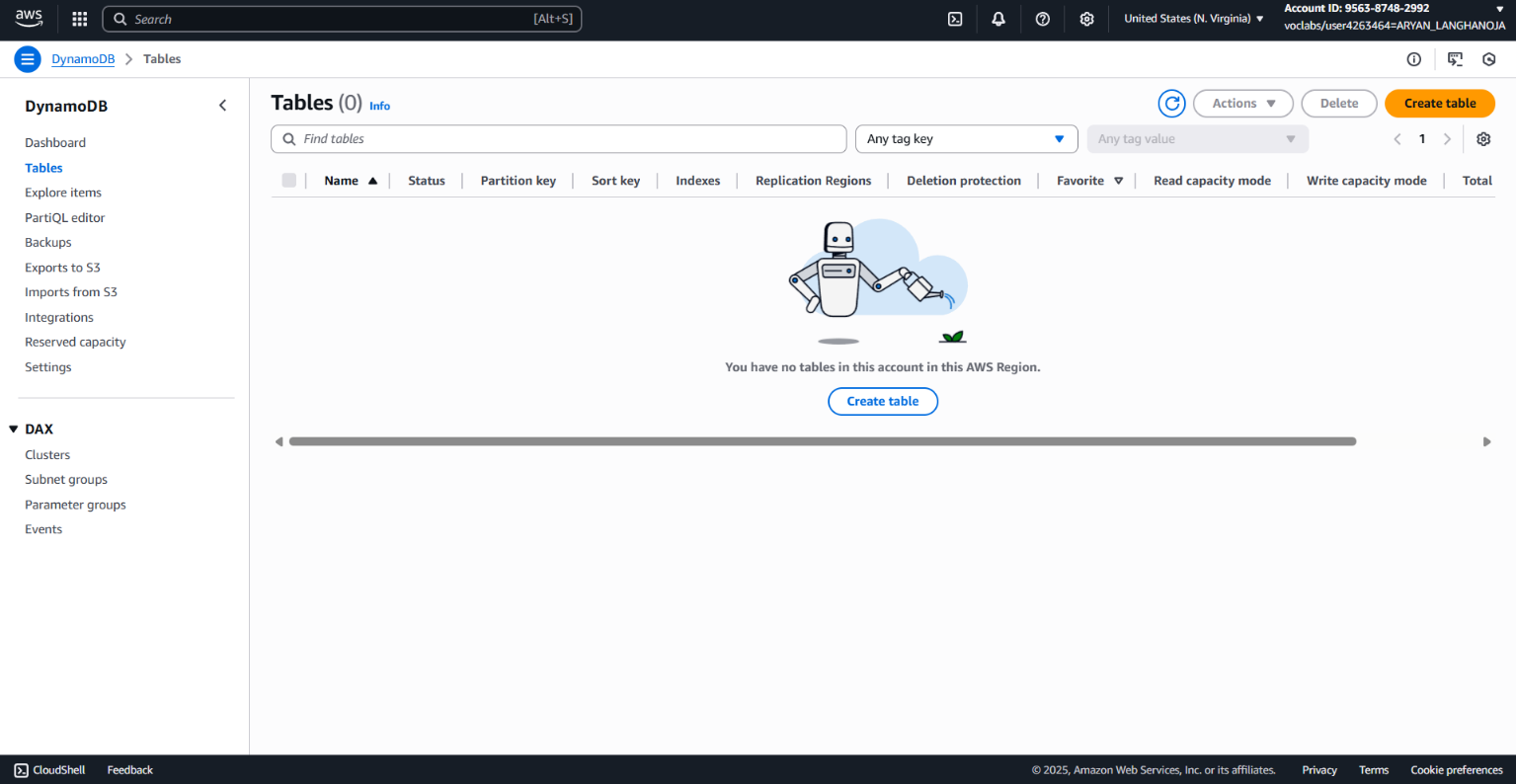
Initially, you create this table with only one *attribute*. Because every DynamoDB table requires a *primary key*, this attribute becomes the primary key for the table. Each value used as a primary key must be unique.

The **product\_name** is the first attribute that you define in the table. The **product\_name** attribute works well because the café's product names should not be duplicated. Also, the café wants to use the product names to query details about each record.

First, verify that no tables exist in the environment :

1. On the AWS Management Console, in the search box type and select DynamoDB .

* On the **Amazon DynamoDB** Management Console, choose **Tables** from the navigation pane.
* Review the **Tables** pane and confirm that no tables exist.



1. Edit the script that will create the table:
   1. Return to the VS Code IDE browser tab.
   2. In the navigation pane of the VS Code IDE, expand the **python\_3** directory.
   3. Open the **create\_table.py** script by double-clicking it.
   4. Replace the *<FMI\_1>* placeholder with the table name, which is:

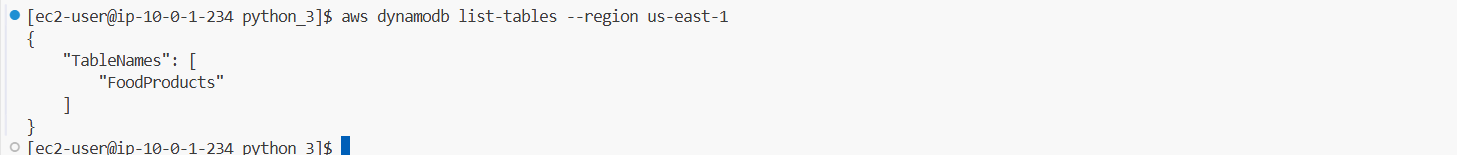
**FoodProducts**

1. In the VS Code Bash terminal, run the following code:

**cd python\_3**

**python3 create\_table.py**

* Once done (and not before), run the following command to make sure the table was successfully created:

** aws dynamodb list-tables --region us-east-1**

**Task 3: Working with DynamoDB data – Understanding DynamoDB condition expressions**

Now that Sofía created the table, she wants to understand what happens when records are written to it.

In this task, you continue as Sofía to insert the first record into the table.

1. Review the JavaScript Object Notation (JSON) data that defines the new record.
   * In the VS Code IDE, expand the **resources** folder.
   * Open the **not\_an\_existing\_product.json** file by double-clicking it.
2. To insert the new record, run the following command. Ensure that you are still in the python\_3 folder.

**aws dynamodb put-item \**

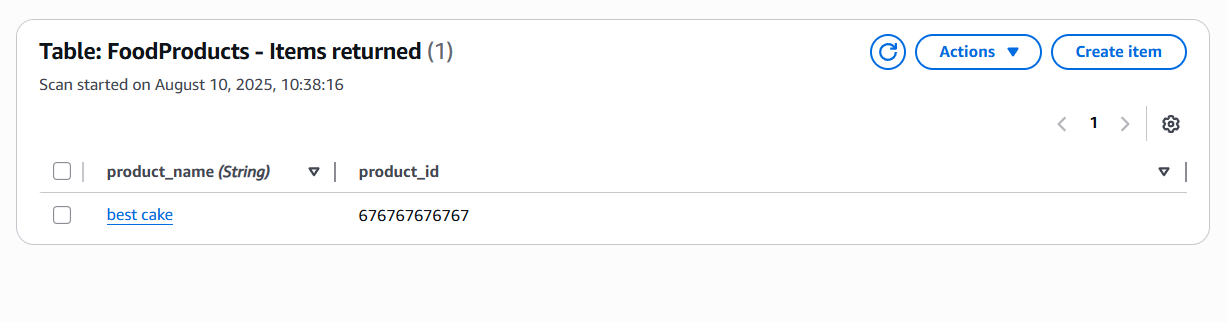
**--table-name FoodProducts \**

**--item file://../resources/not\_an\_existing\_product.json \**

**--region us-east-1**

1. Verify that the new record was added to the table by using the DynamoDB console to complete the following tasks:

* Return to the DynamoDB console and choose the **FoodProducts** link.
* Choose **Explore table items**.
* Under **Items returned**, review the information.



1. Update the JSON data to create a new record:

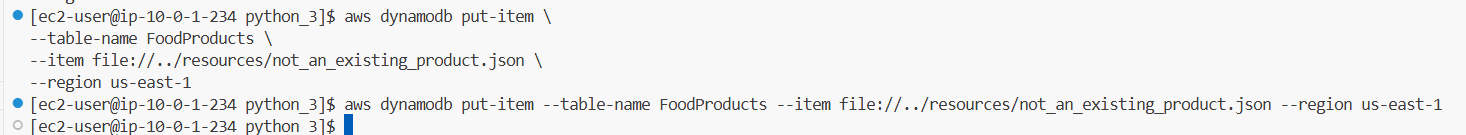
* Return to the VS Code IDE and load the **not\_an\_existing\_product.json** file in the text editor.
* Replace the **product\_name** value of *<best cake>* with best pie
* Do not change the **product\_id** value.
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

1. To add the new record, run the following command. Notice that this command is the same AWS CLI command that you used to add the first record.

**aws dynamodb put-item \**

**--table-name FoodProducts \**

**--item file://../resources/not\_an\_existing\_product.json \**

**--region us-east-1**



1. Update the JSON record:

* Return to the VS Code IDE and the **not\_an\_existing\_product.json** file.
* Don't change the value of **product\_name**.
* Replace the **product\_id** value of *<676767676767>* with 3333333333
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

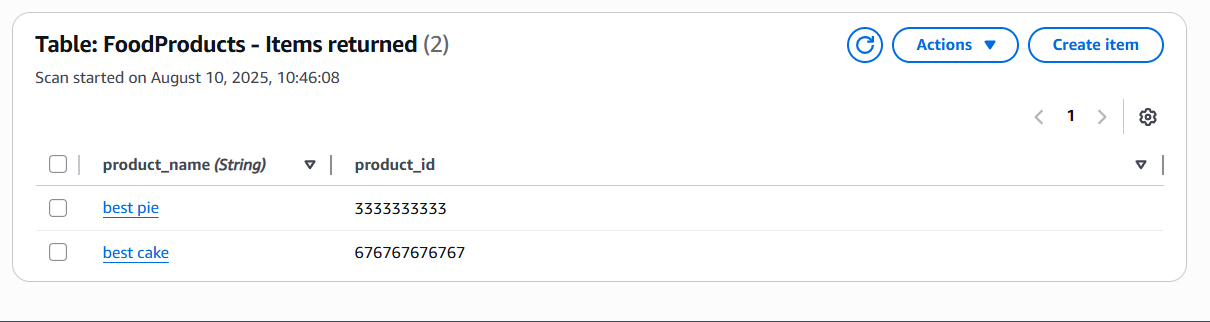
1. Run the previous AWS CLI **put-item** command again:

**aws dynamodb put-item \**

**--table-name FoodProducts \**

**--item file://../resources/not\_an\_existing\_product.json \**

**--region us-east-1**



1. Update the JSON record:

* Return to the VS Code IDE and the **not\_an\_existing\_product.json** file.
* Don't change the value of **product\_name**.
* Replace the **product\_id** value of *<3333333333>* with 2222222222
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

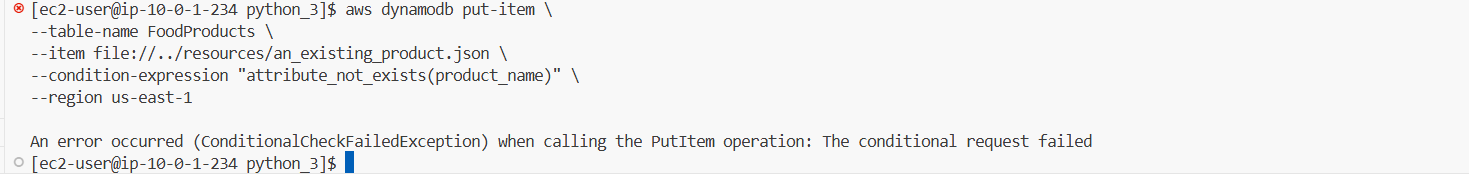
1. In the VS Code terminal, run the following AWS CLI **put-item** command:

**aws dynamodb put-item \**

**--table-name FoodProducts \**

**--item file://../resources/an\_existing\_product.json \**

**--condition-expression "attribute\_not\_exists(product\_name)" \**

**--region us-east-1**

**Task 4: Adding and modifying a single item by using the SDK**

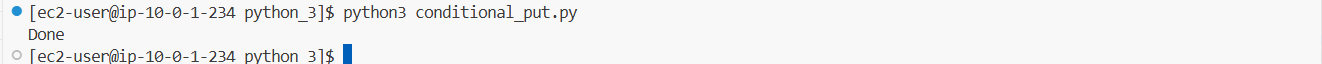
Sofía now has a good understanding of how to use the AWS CLI to control the data that is inserted into the table. She knows that the behavior for inserting data is similar with the SDK. She decides to write to the table by using Python code.

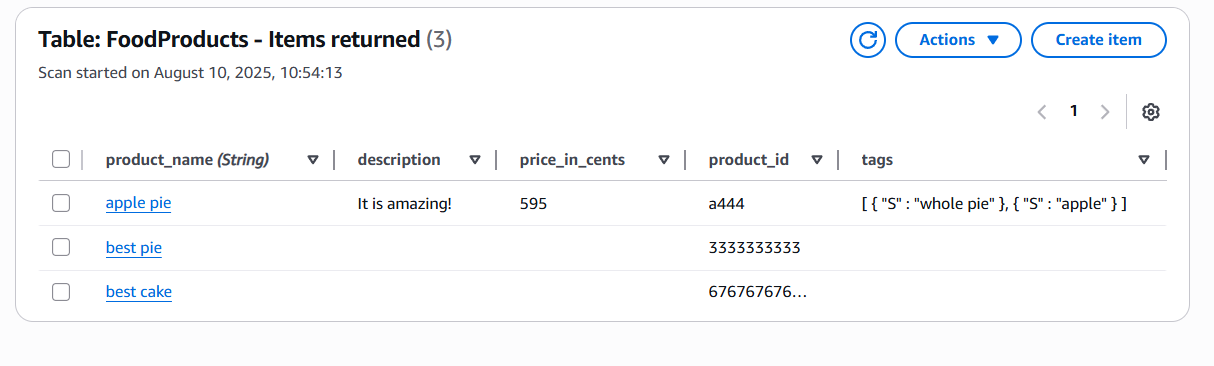
In this task, you continue as Sofía to add and modify a single item by using the SDK.

1. Update the conditional\_put.py script.

* In the VS Code IDE, go to the **python\_3** directory.
* Open the **conditional\_put.py** script.
* Replace the *<FMI>* placeholders as directed in the script. You can also refer to the code analysis in the following step.
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

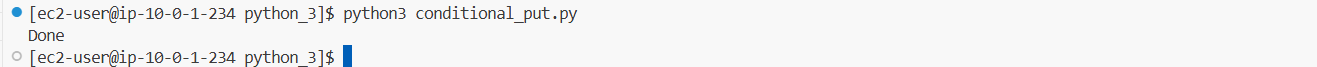
1. In the VS Code terminal, run the file.

**python3 conditional\_put.py**

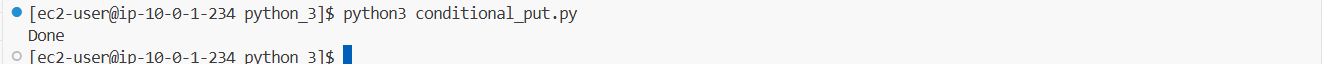
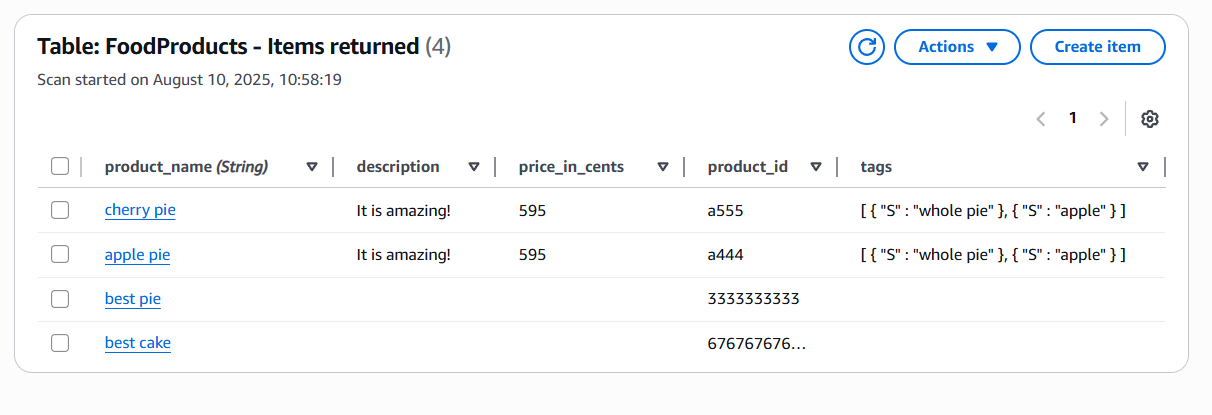


1. In the VS Code IDE, update the **conditional\_put.py** script again. This time, replace the **product\_id** value of *<a444>* to a555 and close the file.

1. Run the script again:

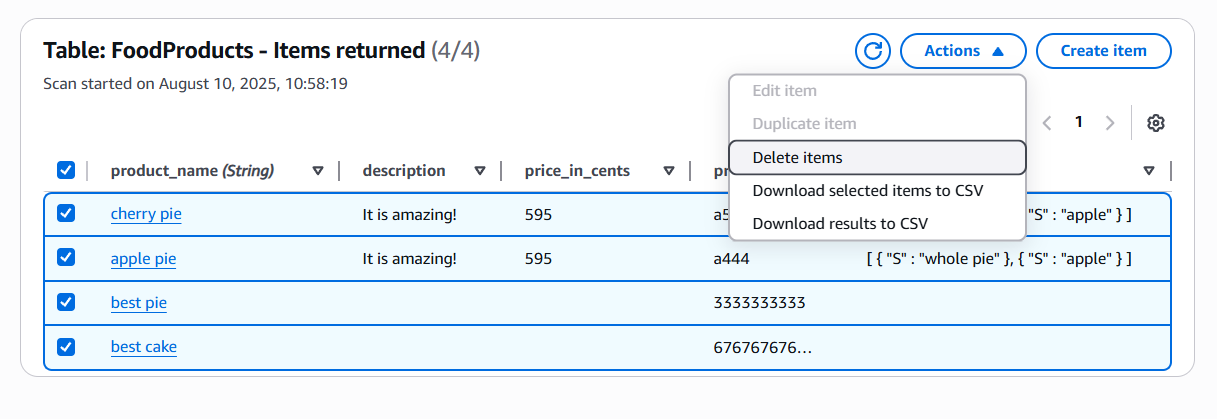
**python3 conditional\_put.py**

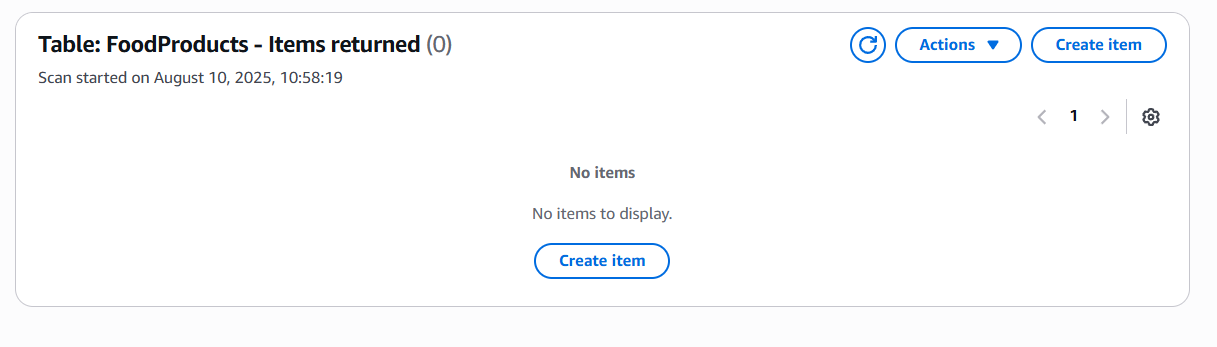
1. In the VS Code IDE, update the **conditional\_put.py** script by replacing the **product\_name** value of *<apple pie>* to cherry pie .
2. Run the python3 conditional\_put.py again.



**Task 5: Adding multiple items by using the SDK and batch processing**

1. In the DynamoDB **Item explorer**, refresh the view of the data by choosing **Run**

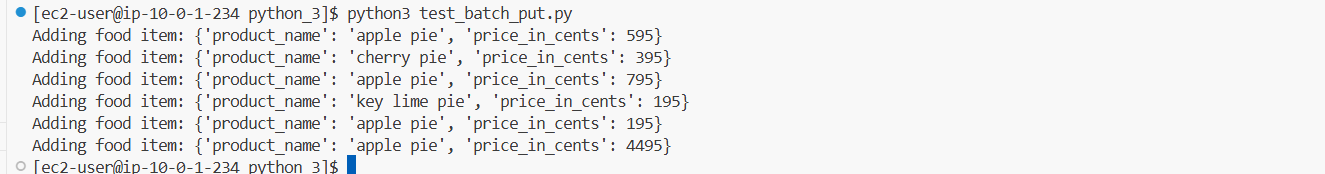
* Delete all records:
* Select the check boxes for all the table records.
* From the **Actions** menu, choose **Delete item(s)**.
* In the pop-up window confirmation box, enter Delete and choose **Delete items**



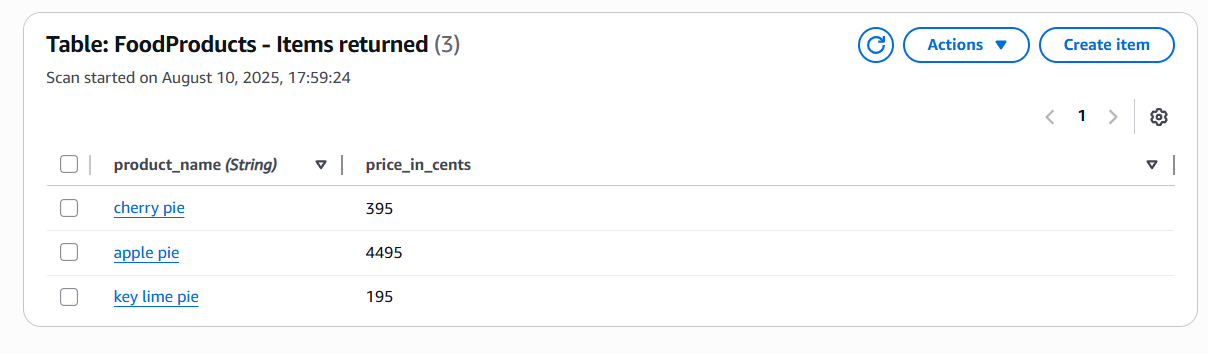
1. Update the **test\_batch\_put.py** script:

* In the VS Code IDE, open the **python\_3** > **test\_batch\_put.py** script.
* Update the *<FMI\_1>* placeholder with the FoodProducts table name.
* Replace the *<FMI\_2>* with the product\_name primary key name.
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

1. In the VS Code terminal, run the file:

**python3 test\_batch\_put.py**

1. In the DynamoDB **Item explorer**, select the **FoodProducts** table, and run the scan again.



* However, you know that the café doesn't want the database to add incorrect values. For this dataset, it's better for the load to fail when duplicate **product\_name** values are found instead of allowing the update to add incorrect values.
* You must change the script so that it fails when duplicates are included in the batch. You can then review and clean up the data. To implement this feature, you remove the **overwrite\_by\_pkeys** parameter from the **batch\_writer** method.

1. To prepare for the production data load, go to the browser tab with the DynamoDB console, and delete all records from the table as you did in the previous steps.
2. You can fix the overwrite behavior by updating the **test\_batch\_put.py** script and preparing to load the production data.

* In the VS Code IDE, open **python\_3** > **test\_batch\_put.py**.
* Update line 12 by changing *<with table.batch\_writer(overwrite\_by\_pkeys=['product\_name']) as batch>* to the following and closing the file:

**with table.batch\_writer() as batch:**

1. Now run the script again:

**python3 test\_batch\_put.py**

****

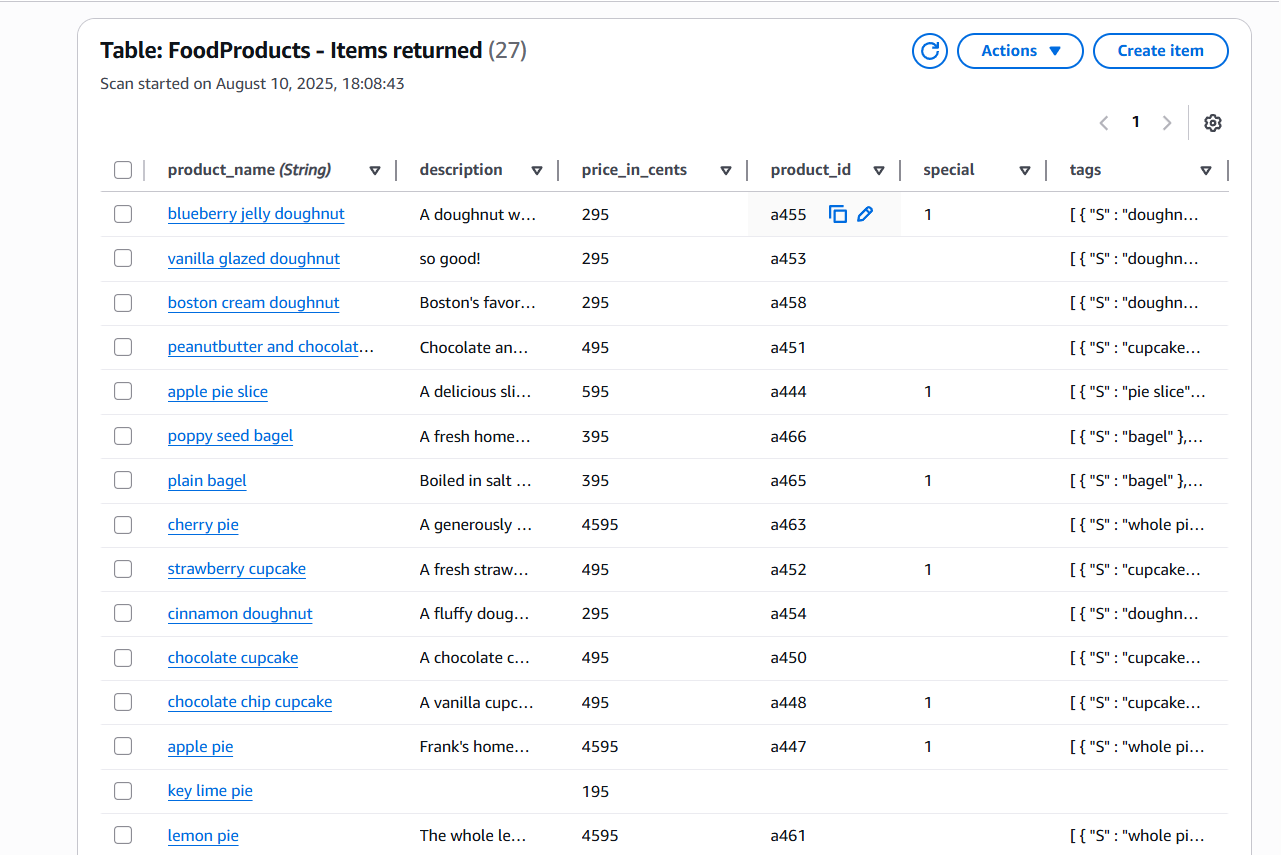
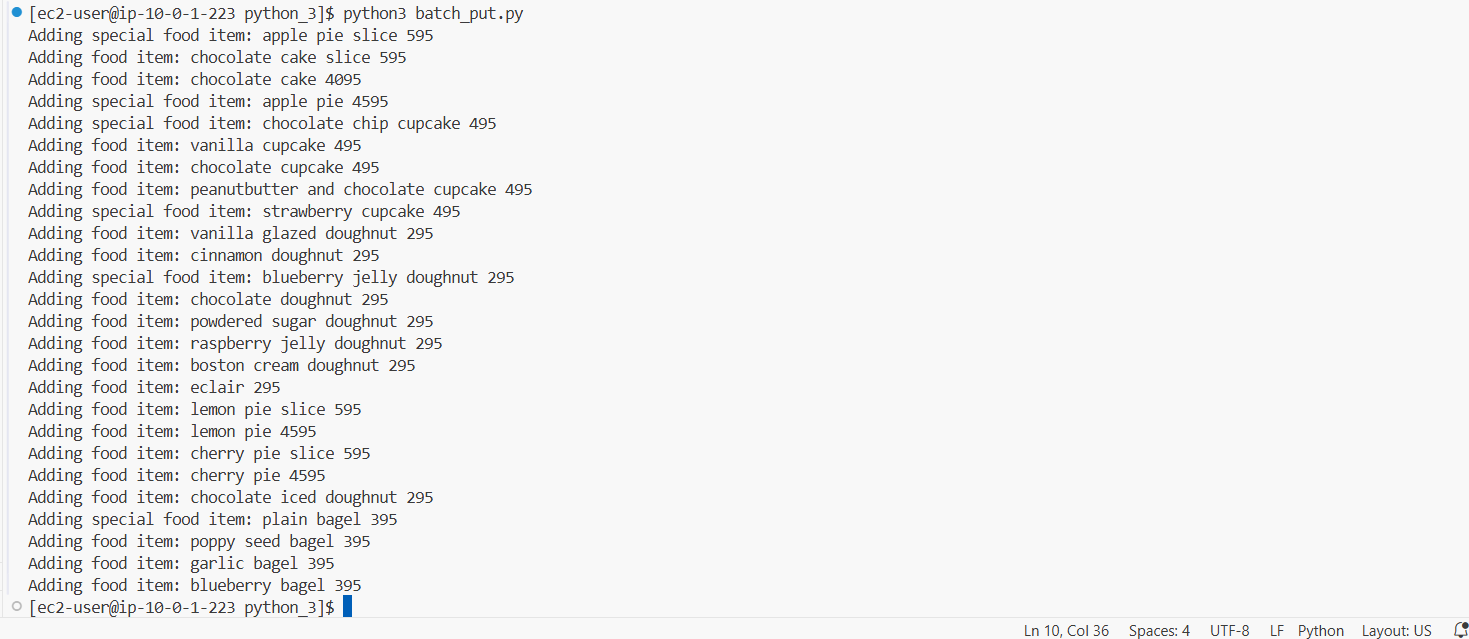
1. In VS Code, review the contents of the **resources/website/all\_products.json** file. You will find many items. These items have several attributes, and some include an optional integer attribute called **specials**.

In order to load the raw JSON used in the website, you use a new script called **batch\_put.py**.

It is very similar to the **test\_batch\_put.py** script. This script allows for the optional integer **special** attribute and also maps the names of more fields to the correct DynamoDB attribute types.

1. Modify the **python\_3/batch\_put.py** script.

* Replace *<FMI>* with FoodProducts
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

****

1. Edit the script that selects all records from the table:

* In VS Code IDE, open **python\_3** > **get\_all\_items.py**.
* **Note**: Do not use the **get\_all\_items\_py** file in the **resources** folder.
* Update the *<FMI\_1>* placeholder with the FoodProducts table name.
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

1. In the VS Code terminal, run the script:

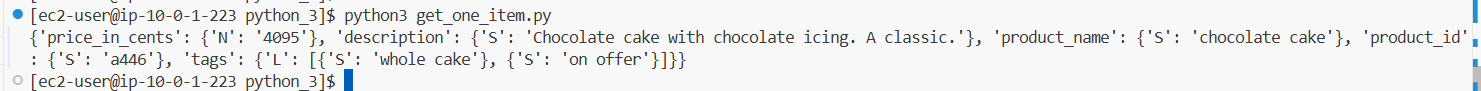
**python3 get\_all\_items.py**



38. Update the **get\_one\_item.py** script.

* Replace the *<FMI\_1>* with the name of the table's primary key.
* Close the file by choosing **X** from the top. (Your changes are saved automatically).

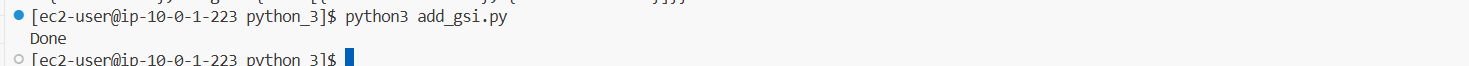
39. In the VS Code terminal, run the following command:

**python3 get\_one\_item.py**

**Task 7: Adding a global secondary index to the table**

40. Update the **add\_gsi.py** script.

* Replace the *<FMI\_1>* with the **KeyType** of HASH
* Close the file by choosing **X** from the top. (Your changes are saved automatically).



41. Update the **scan\_with\_filter.py** script.

* Change *<FMI\_1>* to special\_GSI
* Change *<FMI\_2>* to tags
* Close the file by choosing **X** from the top. (Your changes are saved automatically)

42. Run the following command:

**python3 scan\_with\_filter.py**

**Conclusion:-**

* In this Lab I Learned the AWS DynamoDB Service.
* I Had done the CRUD operations of FoodProduct Table.
* I had Learned About Primary key.
* I Had Learned about Overwrite by PK.
* I had learned how to add the index in the table.

**Result :-**

