Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Cloud Developing (01CT0720)	**Aim: Create, edit, and update the database using AWS DynamoDB along with global index using python SDK.**	
Experiment No: 04	**Date:**	**Enrolment No: 92200133030**

__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.

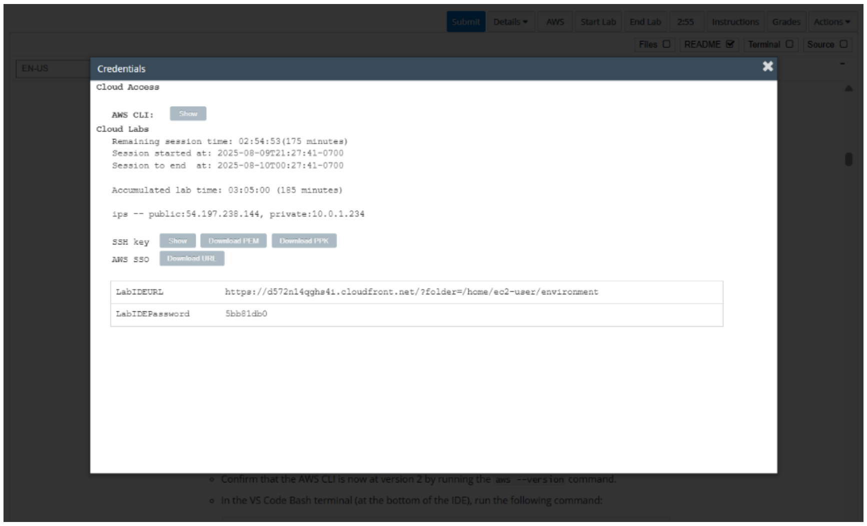
Café staff must be able to retrieve information from the table. Sofía decides to create one script that retrieves all inventory items from the table and another script (as a proof of concept) that uses a product name to retrieve a single record.

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.
    a. **LabIDEURL**
    b. **LabIDEPassword**

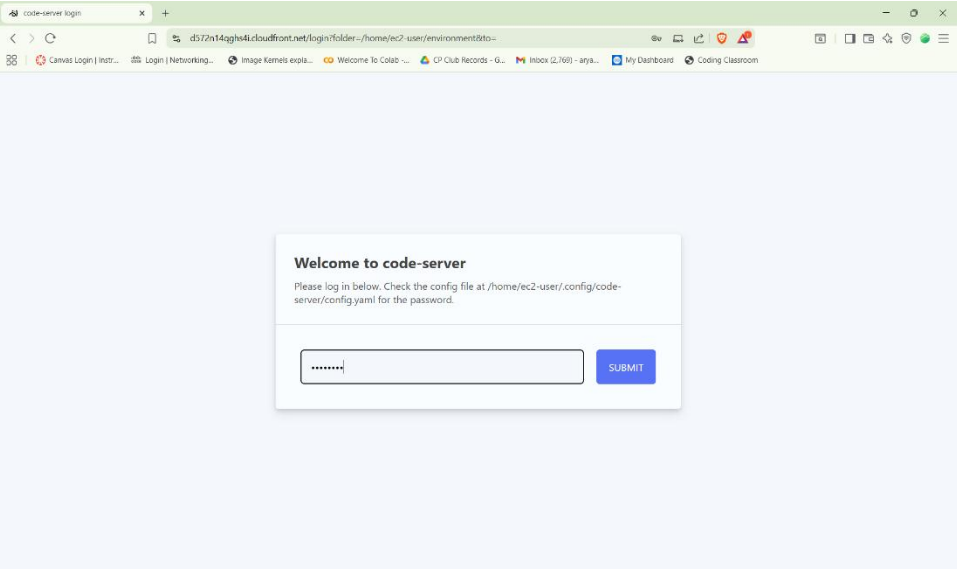


```
Cloud Access

  AWS CLI:
Cloud Labs
  Remaining session time: 02:54:53(175 minutes)
  Session started at: 2025-08-09T21:27:41-0700
  Session to end  at: 2025-08-10T00:27:41-0700

  Accumulated lab time: 03:05:00 (185 minutes)

  ips -- public:54.197.238.144, private:10.0.1.234

  SSH key
  AWS SSO
```

LabIDEURL	https://d572n14qghs4i.cloudfront.net/?folder=/home/ec2-user/environment
LabIDEPassword	5bb81db0

- 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:

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.



Welcome to code-server

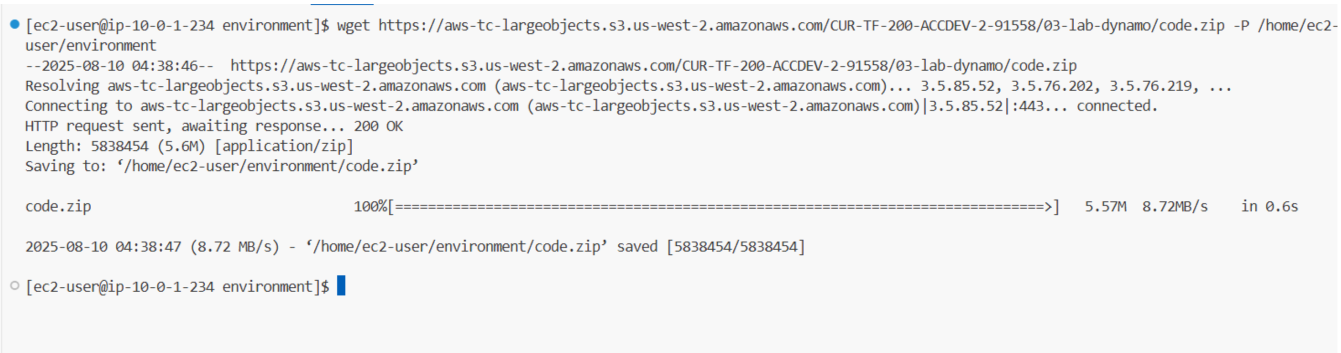
Please log in below. Check the config file at /home/ec2-user/.config/code-server/config.yaml for the password.

•••••••••

SUBMIT

5. 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:

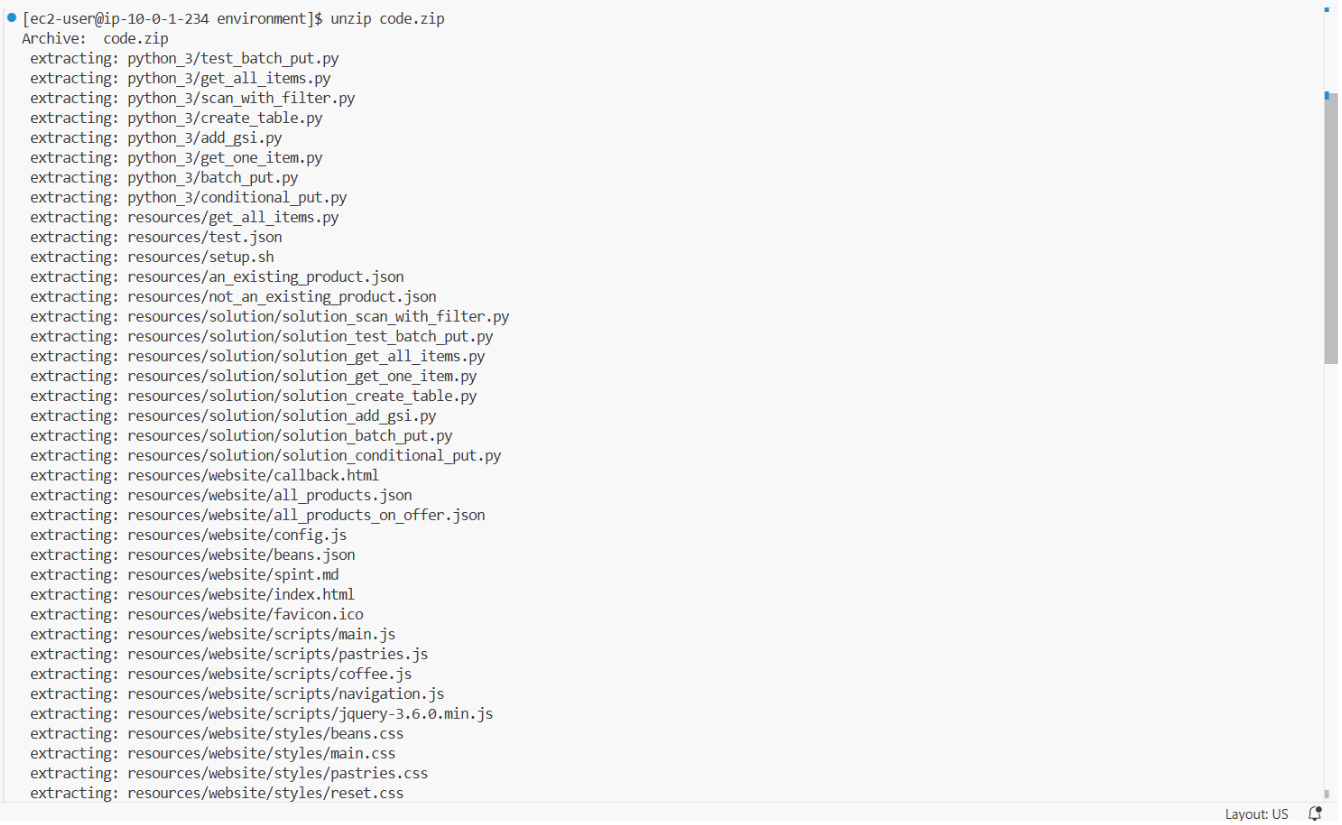
```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/03-lab-dynamo/code.zip -P /home/ec2-user/environment
```



```
● [ec2-user@ip-10-0-1-234 environment]$ 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
  --2025-08-10 04:38:46--  https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/03-lab-dynamo/code.zip
  Resolving aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)... 3.5.85.52, 3.5.76.202, 3.5.76.219, ...
  Connecting to aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)|3.5.85.52|:443... connected.
  HTTP request sent, awaiting response... 200 OK
  Length: 5838454 (5.6M) [application/zip]
  Saving to: ‘/home/ec2-user/environment/code.zip’

  code.zip                        100%[==============================================================================>]   5.57M  8.72MB/s    in 0.6s

  2025-08-10 04:38:47 (8.72 MB/s) - ‘/home/ec2-user/environment/code.zip’ saved [5838454/5838454]

○ [ec2-user@ip-10-0-1-234 environment]$ 
```

6. 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:

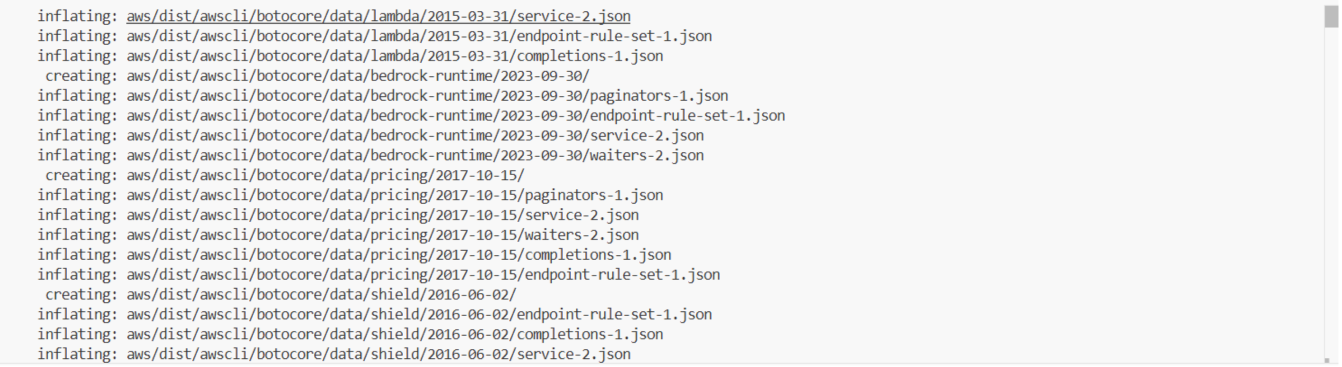
```
unzip code.zip
```



```
[ec2-user@ip-10-0-1-234 environment]$ unzip code.zip
Archive:  code.zip
  extracting: python_3/test_batch_put.py
  extracting: python_3/get_all_items.py
  extracting: python_3/scan_with_filter.py
  extracting: python_3/create_table.py
  extracting: python_3/add_gsi.py
  extracting: python_3/get_one_item.py
  extracting: python_3/batch_put.py
  extracting: python_3/conditional_put.py
  extracting: resources/get_all_items.py
  extracting: resources/test.json
  extracting: resources/setup.sh
  extracting: resources/an_existing_product.json
  extracting: resources/not_an_existing_product.json
  extracting: resources/solution/solution_scan_with_filter.py
  extracting: resources/solution/solution_test_batch_put.py
  extracting: resources/solution/solution_get_all_items.py
  extracting: resources/solution/solution_get_one_item.py
  extracting: resources/solution/solution_create_table.py
  extracting: resources/solution/solution_add_gsi.py
  extracting: resources/solution/solution_batch_put.py
  extracting: resources/solution/solution_conditional_put.py
  extracting: resources/website/callback.html
  extracting: resources/website/all_products.json
  extracting: resources/website/all_products_on_offer.json
  extracting: resources/website/config.js
  extracting: resources/website/beans.json
  extracting: resources/website/spint.md
  extracting: resources/website/index.html
  extracting: resources/website/favicon.ico
  extracting: resources/website/scripts/main.js
  extracting: resources/website/scripts/pastries.js
  extracting: resources/website/scripts/coffee.js
  extracting: resources/website/scripts/navigation.js
  extracting: resources/website/scripts/jquery-3.6.0.min.js
  extracting: resources/website/styles/beans.css
  extracting: resources/website/styles/main.css
  extracting: resources/website/styles/pastries.css
  extracting: resources/website/styles/reset.css
```

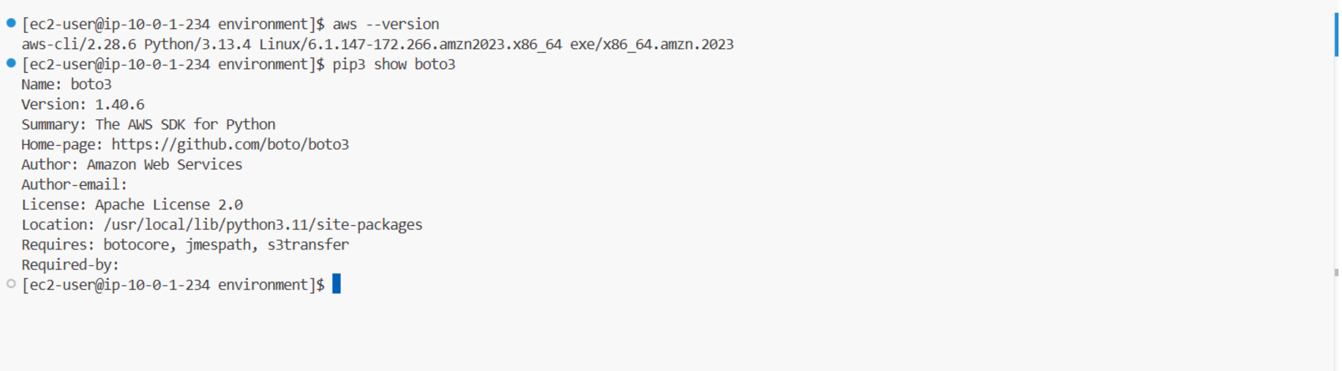
7. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.
   - To set permissions on the script and then run it, run the following commands in the Bash terminal:

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



```
  inflating: aws/dist/awscli/botocore/data/lambda/2015-03-31/service-2.json
  inflating: aws/dist/awscli/botocore/data/lambda/2015-03-31/endpoint-rule-set-1.json
  inflating: aws/dist/awscli/botocore/data/lambda/2015-03-31/completions-1.json
   creating: aws/dist/awscli/botocore/data/bedrock-runtime/2023-09-30/
  inflating: aws/dist/awscli/botocore/data/bedrock-runtime/2023-09-30/paginators-1.json
  inflating: aws/dist/awscli/botocore/data/bedrock-runtime/2023-09-30/endpoint-rule-set-1.json
  inflating: aws/dist/awscli/botocore/data/bedrock-runtime/2023-09-30/service-2.json
  inflating: aws/dist/awscli/botocore/data/bedrock-runtime/2023-09-30/waiters-2.json
   creating: aws/dist/awscli/botocore/data/pricing/2017-10-15/
  inflating: aws/dist/awscli/botocore/data/pricing/2017-10-15/paginators-1.json
  inflating: aws/dist/awscli/botocore/data/pricing/2017-10-15/service-2.json
  inflating: aws/dist/awscli/botocore/data/pricing/2017-10-15/waiters-2.json
  inflating: aws/dist/awscli/botocore/data/pricing/2017-10-15/completions-1.json
  inflating: aws/dist/awscli/botocore/data/pricing/2017-10-15/endpoint-rule-set-1.json
   creating: aws/dist/awscli/botocore/data/shield/2016-06-02/
  inflating: aws/dist/awscli/botocore/data/shield/2016-06-02/endpoint-rule-set-1.json
  inflating: aws/dist/awscli/botocore/data/shield/2016-06-02/completions-1.json
  inflating: aws/dist/awscli/botocore/data/shield/2016-06-02/service-2.json
```

8. 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`**



```
● [ec2-user@ip-10-0-1-234 environment]$ aws --version
  aws-cli/2.28.6 Python/3.13.4 Linux/6.1.147-172.266.amzn2023.x86_64 exe/x86_64.amzn.2023
● [ec2-user@ip-10-0-1-234 environment]$ pip3 show boto3
  Name: boto3
  Version: 1.40.6
  Summary: The AWS SDK for Python
  Home-page: https://github.com/boto/boto3
  Author: Amazon Web Services
  Author-email:
  License: Apache License 2.0
  Location: /usr/local/lib/python3.11/site-packages
  Requires: botocore, jmespath, s3transfer
  Required-by:
○ [ec2-user@ip-10-0-1-234 environment]$ 
```

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.

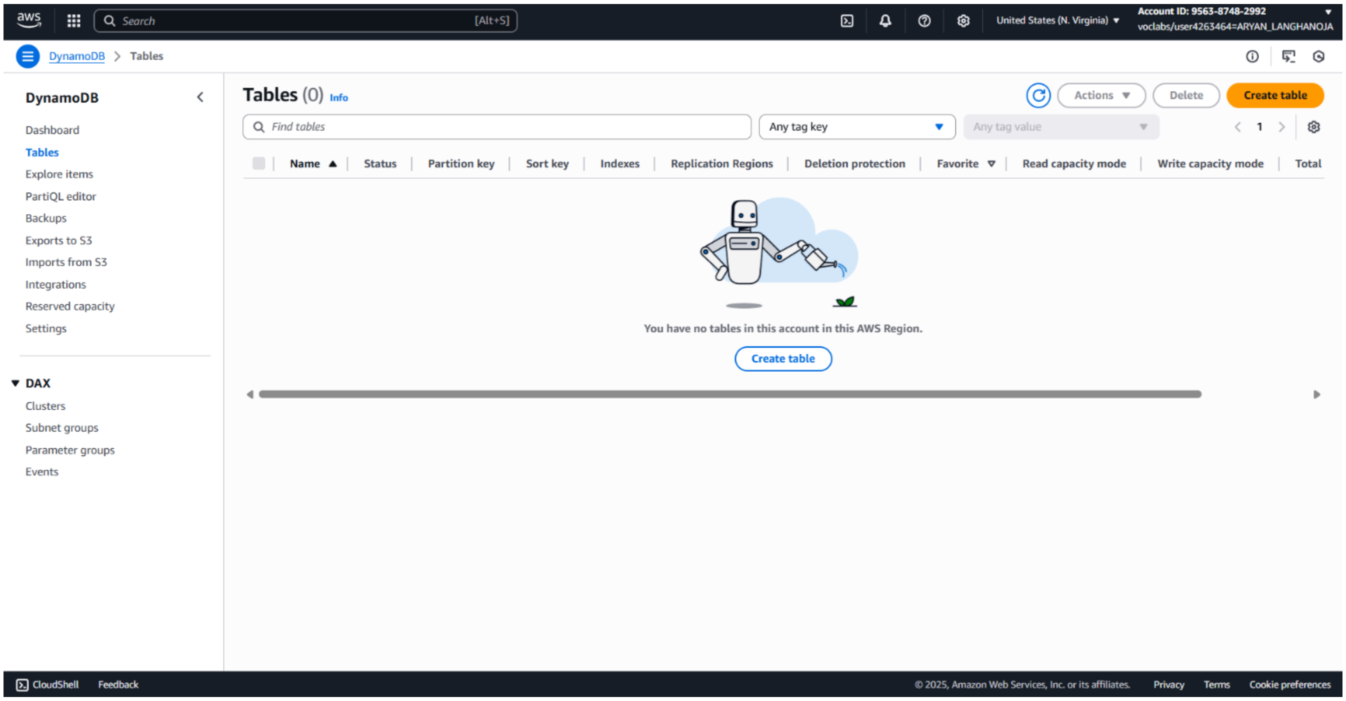
Initially, you create this table with only one *attribute*. Because every DynamoDB table requires a *primary key*, this attribute becomes the primary key for the table. Each value used as a primary key must be unique.

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 :

9. 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.



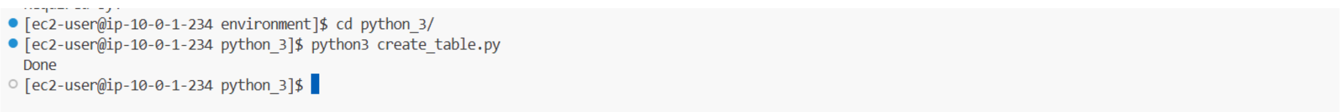
You have no tables in this account in this AWS Region.

10. Edit the script that will create the table:
    a. Return to the VS Code IDE browser tab.
    b. In the navigation pane of the VS Code IDE, expand the **python_3** directory.
    c. Open the **create_table.py** script by double-clicking it.
    d. Replace the *<FMI_1>* placeholder with the table name, which is:

       `FoodProducts`

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

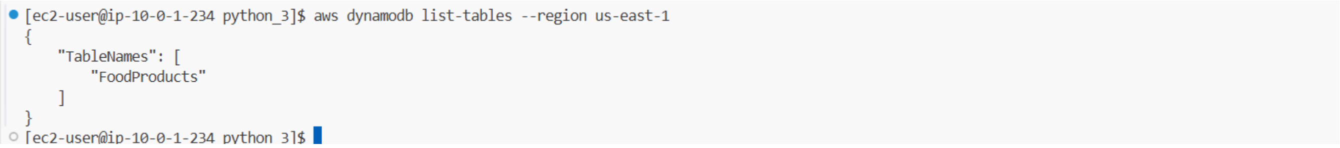
```
cd python_3
python3 create_table.py
```



```
● [ec2-user@ip-10-0-1-234 environment]$ cd python_3/
● [ec2-user@ip-10-0-1-234 python_3]$ python3 create_table.py
  Done
○ [ec2-user@ip-10-0-1-234 python_3]$ 
```

- 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
```



```
● [ec2-user@ip-10-0-1-234 python_3]$ aws dynamodb list-tables --region us-east-1
  {
      "TableNames": [
          "FoodProducts"
      ]
  }
○ [ec2-user@ip-10-0-1-234 python_3]$ 
```

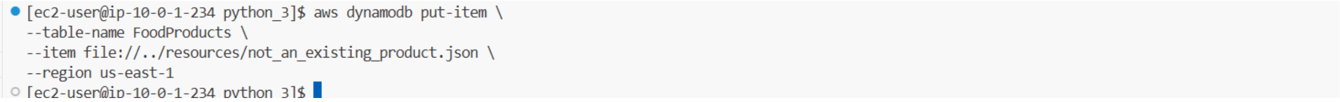
Task 3: Working with DynamoDB data – Understanding DynamoDB condition expressions

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.

12. 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.

13. 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
```



```
● [ec2-user@ip-10-0-1-234 python_3]$ aws dynamodb put-item \
  --table-name FoodProducts \
  --item file://../resources/not_an_existing_product.json \
  --region us-east-1
○ [ec2-user@ip-10-0-1-234 python_3]$ 
```

14. 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.

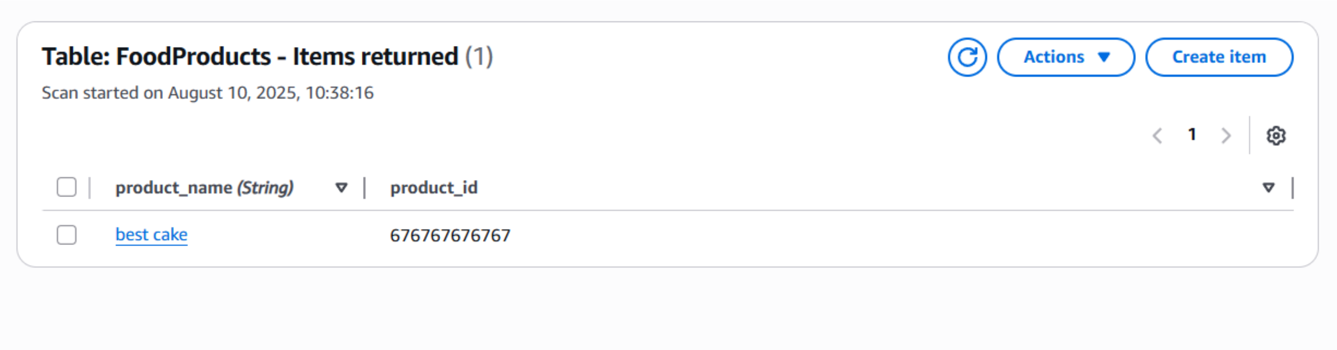


Table: FoodProducts - Items returned (1)

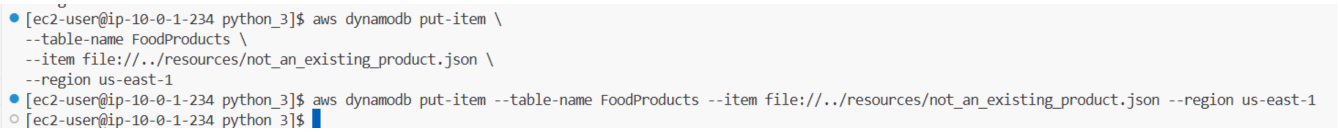
Scan started on August 10, 2025, 10:38:16

☐	product_name (String)	product_id
☐	best cake	676767676767

15. 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)

16. 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
```



```
● [ec2-user@ip-10-0-1-234 python_3]$ aws dynamodb put-item \
  --table-name FoodProducts \
  --item file://../resources/not_an_existing_product.json \
  --region us-east-1
● [ec2-user@ip-10-0-1-234 python_3]$ aws dynamodb put-item --table-name FoodProducts --item file://../resources/not_an_existing_product.json --region us-east-1
○ [ec2-user@ip-10-0-1-234 python_3]$ 
```

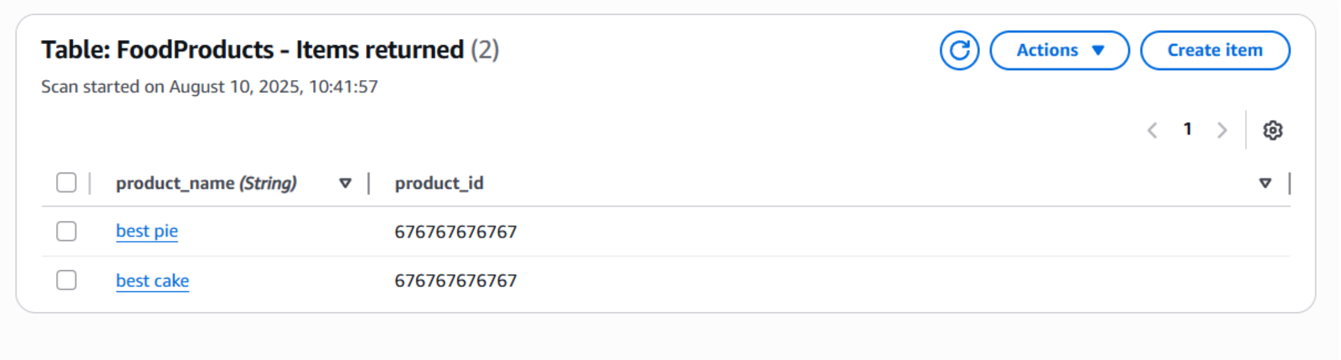


Table: FoodProducts - Items returned (2)

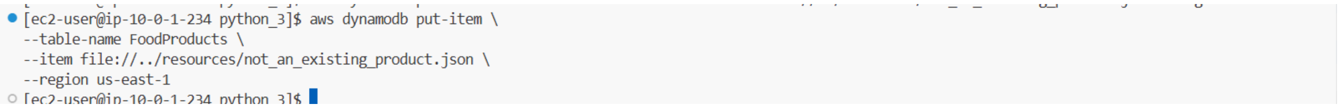
Scan started on August 10, 2025, 10:41:57

□	product_name (String)	product_id
□	best pie	676767676767
□	best cake	676767676767

17. 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)

18. 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
```



```
● [ec2-user@ip-10-0-1-234 python_3]$ aws dynamodb put-item \
  --table-name FoodProducts \
  --item file://../resources/not_an_existing_product.json \
  --region us-east-1
○ [ec2-user@ip-10-0-1-234 python_3]$ 
```

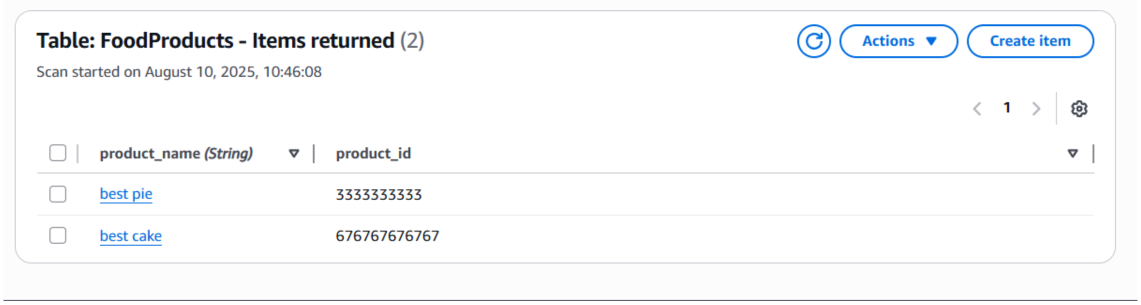


Table: FoodProducts - Items returned (2)

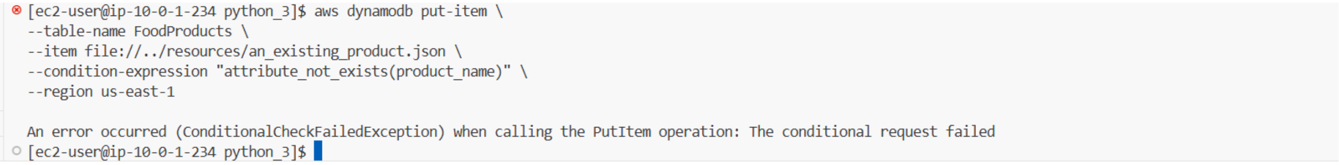
Scan started on August 10, 2025, 10:46:08

□	product_name (String)	product_id
□	best pie	3333333333
□	best cake	676767676767

19. 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)

20. 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
```



```
⊗ [ec2-user@ip-10-0-1-234 python_3]$ 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

  An error occurred (ConditionalCheckFailedException) when calling the PutItem operation: The conditional request failed
○ [ec2-user@ip-10-0-1-234 python_3]$ 
```

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

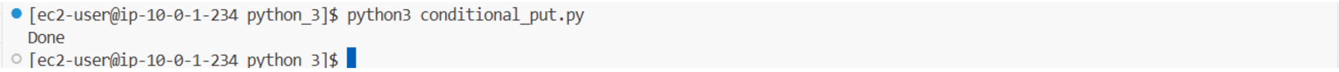
Sofía now has a good understanding of how to use the AWS CLI to control the data that is inserted into the table. She knows that the behavior for inserting data is similar with the SDK. She decides to write to the table by using Python code.

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

21. 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)

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

```
python3 conditional_put.py
```



```
● [ec2-user@ip-10-0-1-234 python_3]$ python3 conditional_put.py
  Done
○ [ec2-user@ip-10-0-1-234 python_3]$ 
```

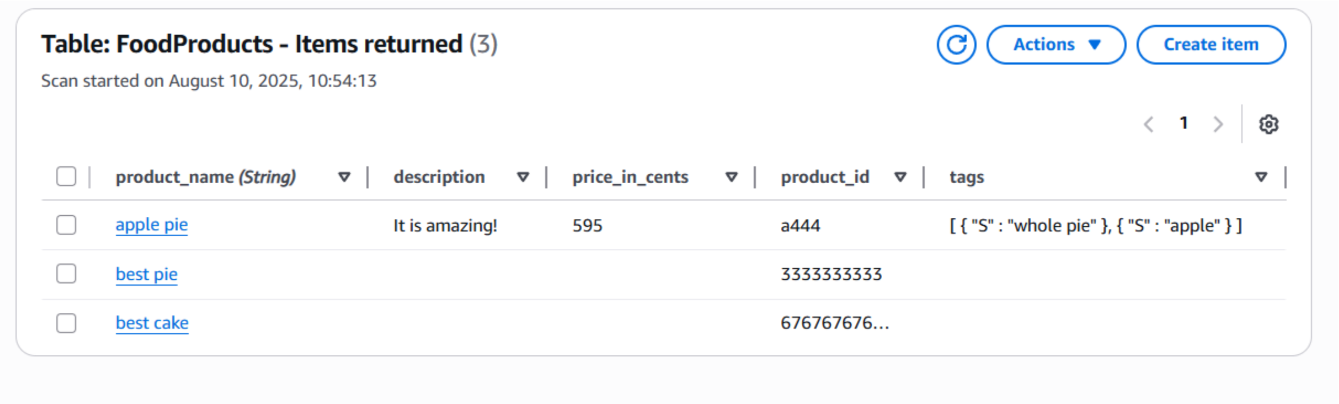


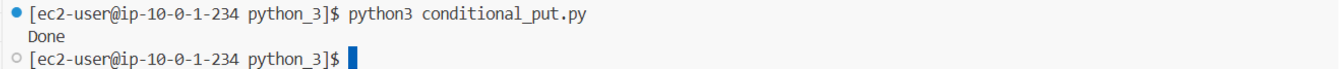
Table: FoodProducts - Items returned (3)

Scan started on August 10, 2025, 10:54:13

□	product_name *(String)*	description	price_in_cents	product_id	tags
□	apple pie	It is amazing!	595	a444	[{ "S" : "whole pie" }, { "S" : "apple" }]
□	best pie			3333333333	
□	best cake			676767676...	

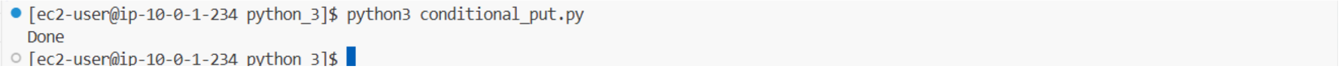
23. In the VS Code IDE, update the **conditional_put.py** script again. This time, replace the **product_id** value of *<a444>* to a555 and close the file.

24. Run the script again:
    `python3 conditional_put.py`



```
● [ec2-user@ip-10-0-1-234 python_3]$ python3 conditional_put.py
  Done
○ [ec2-user@ip-10-0-1-234 python_3]$ █
```

25. In the VS Code IDE, update the **conditional_put.py** script by replacing the **product_name** value of *<apple pie>* to `cherry pie` .
26. Run the `python3 conditional_put.py` again.



```
● [ec2-user@ip-10-0-1-234 python_3]$ python3 conditional_put.py
  Done
○ [ec2-user@ip-10-0-1-234 python_3]$ █
```

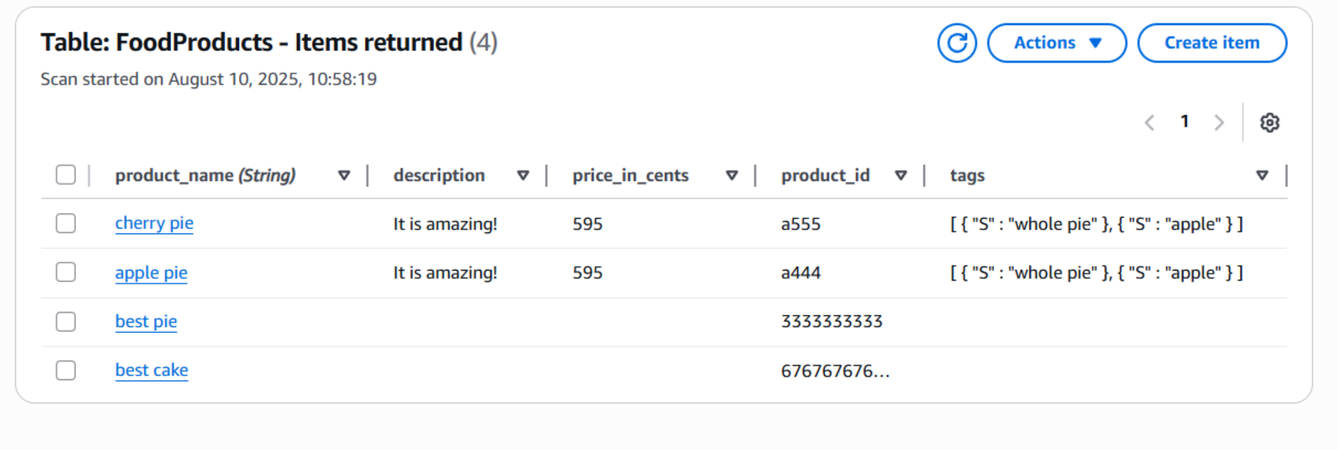


Table: FoodProducts - Items returned (4)

Scan started on August 10, 2025, 10:58:19

□	product_name *(String)*	description	price_in_cents	product_id	tags
□	cherry pie	It is amazing!	595	a555	[{ "S" : "whole pie" }, { "S" : "apple" }]
□	apple pie	It is amazing!	595	a444	[{ "S" : "whole pie" }, { "S" : "apple" }]
□	best pie			3333333333	
□	best cake			676767676...	

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

27. 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**

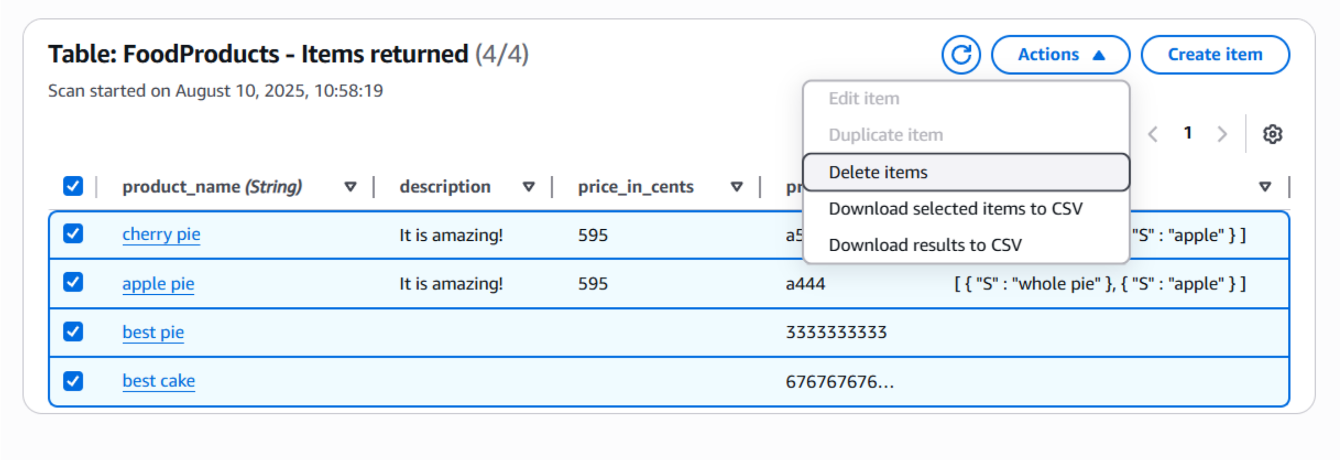


Table: FoodProducts - Items returned (4/4)

Scan started on August 10, 2025, 10:58:19

☑	product_name *(String)*	description	price_in_cents	pr	
☑	cherry pie	It is amazing!	595	a5	"S" : "apple" }]
☑	apple pie	It is amazing!	595	a444	[{ "S" : "whole pie" }, { "S" : "apple" }]
☑	best pie			3333333333	
☑	best cake			676767676...	

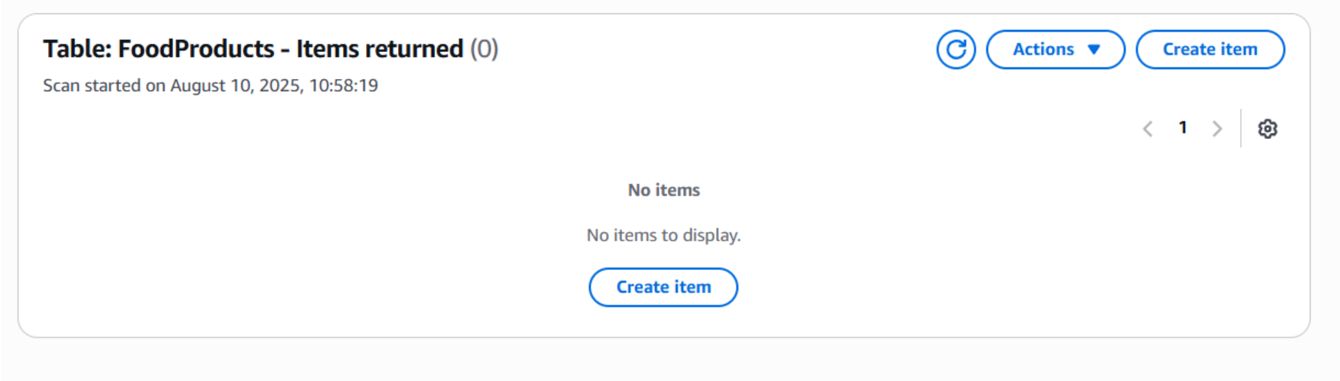


Table: FoodProducts - Items returned (0)

Scan started on August 10, 2025, 10:58:19

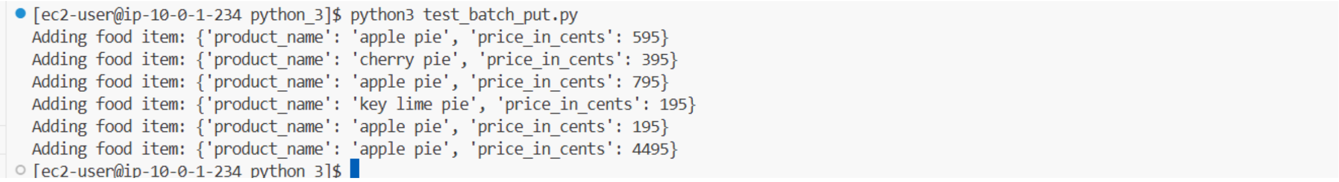
No items

No items to display.

28. 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)

29. In the VS Code terminal, run the file:
    `python3 test_batch_put.py`



```
[ec2-user@ip-10-0-1-234 python_3]$ python3 test_batch_put.py
Adding food item: {'product_name': 'apple pie', 'price_in_cents': 595}
Adding food item: {'product_name': 'cherry pie', 'price_in_cents': 395}
Adding food item: {'product_name': 'apple pie', 'price_in_cents': 795}
Adding food item: {'product_name': 'key lime pie', 'price_in_cents': 195}
Adding food item: {'product_name': 'apple pie', 'price_in_cents': 195}
Adding food item: {'product_name': 'apple pie', 'price_in_cents': 4495}
[ec2-user@ip-10-0-1-234 python_3]$ 
```

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

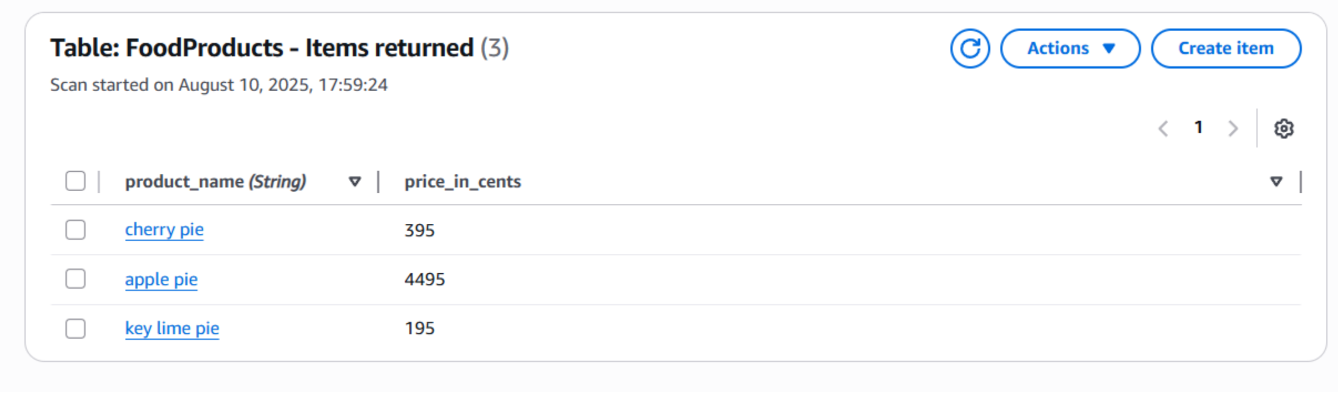


Table: FoodProducts - Items returned (3)

Scan started on August 10, 2025, 17:59:24

□	product_name *(String)*	price_in_cents
□	cherry pie	395
□	apple pie	4495
□	key lime pie	195

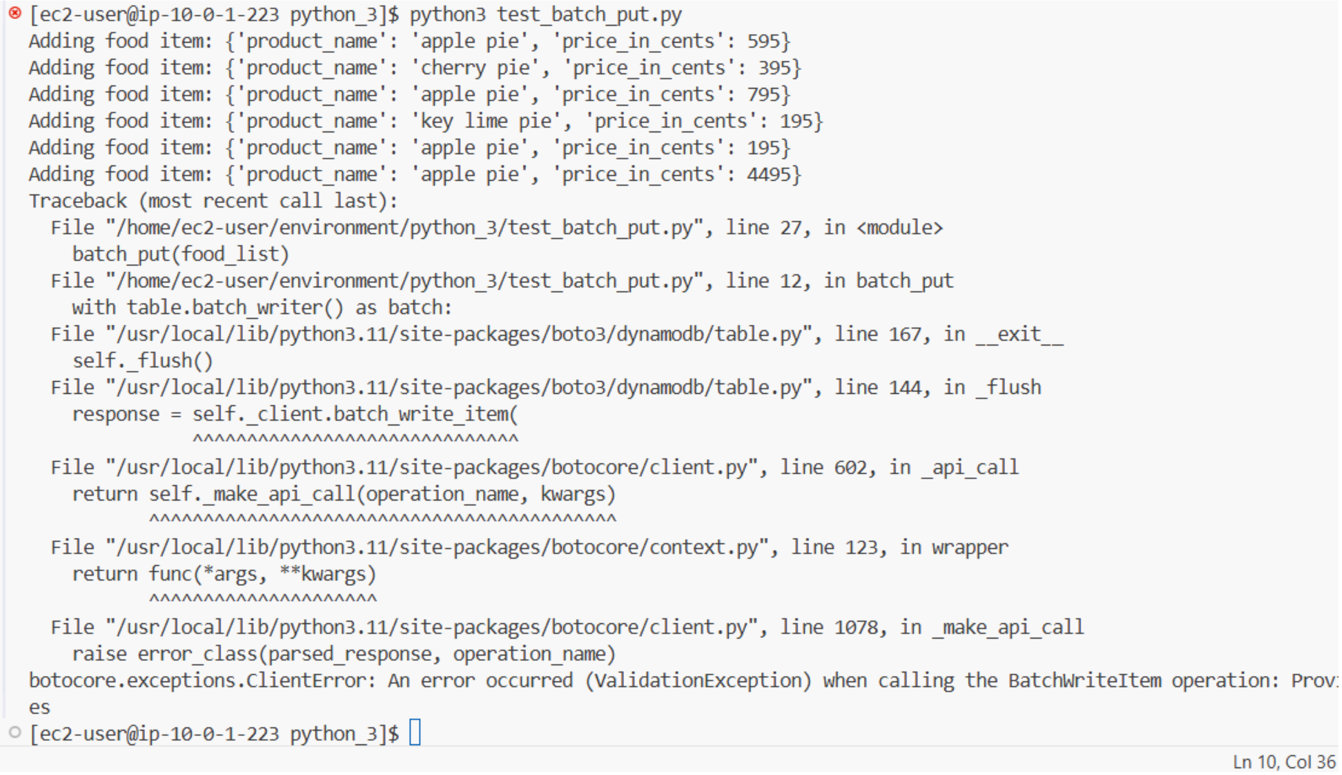
- However, you know that the café doesn't want the database to add incorrect values. For this dataset, it's better for the load to fail when duplicate **product_name** values are found instead of allowing the update to add incorrect values.
- You must change the script so that it fails when duplicates 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.

31. 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.
32. 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:
```

33. Now run the script again:
    `python3 test_batch_put.py`



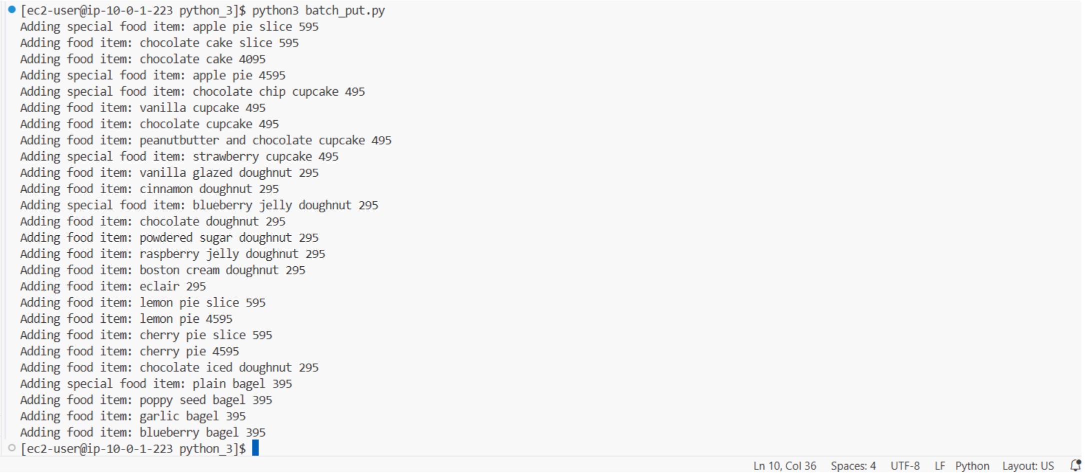
```
⊗ [ec2-user@ip-10-0-1-223 python_3]$ python3 test_batch_put.py
  Adding food item: {'product_name': 'apple pie', 'price_in_cents': 595}
  Adding food item: {'product_name': 'cherry pie', 'price_in_cents': 395}
  Adding food item: {'product_name': 'apple pie', 'price_in_cents': 795}
  Adding food item: {'product_name': 'key lime pie', 'price_in_cents': 195}
  Adding food item: {'product_name': 'apple pie', 'price_in_cents': 195}
  Adding food item: {'product_name': 'apple pie', 'price_in_cents': 4495}
  Traceback (most recent call last):
    File "/home/ec2-user/environment/python_3/test_batch_put.py", line 27, in <module>
      batch_put(food_list)
    File "/home/ec2-user/environment/python_3/test_batch_put.py", line 12, in batch_put
      with table.batch_writer() as batch:
    File "/usr/local/lib/python3.11/site-packages/boto3/dynamodb/table.py", line 167, in __exit__
      self._flush()
    File "/usr/local/lib/python3.11/site-packages/boto3/dynamodb/table.py", line 144, in _flush
      response = self._client.batch_write_item(
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "/usr/local/lib/python3.11/site-packages/botocore/client.py", line 602, in _api_call
      return self._make_api_call(operation_name, kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    File "/usr/local/lib/python3.11/site-packages/botocore/context.py", line 123, in wrapper
      return func(*args, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^^
    File "/usr/local/lib/python3.11/site-packages/botocore/client.py", line 1078, in _make_api_call
      raise error_class(parsed_response, operation_name)
  botocore.exceptions.ClientError: An error occurred (ValidationException) when calling the BatchWriteItem operation: Prov
  es
○ [ec2-user@ip-10-0-1-223 python_3]$ 
```

34. 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.

35. 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)



```
● [ec2-user@ip-10-0-1-223 python_3]$ python3 batch_put.py
  Adding special food item: apple pie slice 595
  Adding food item: chocolate cake slice 595
  Adding food item: chocolate cake 4095
  Adding special food item: apple pie 4595
  Adding special food item: chocolate chip cupcake 495
  Adding food item: vanilla cupcake 495
  Adding food item: chocolate cupcake 495
  Adding food item: peanutbutter and chocolate cupcake 495
  Adding special food item: strawberry cupcake 495
  Adding food item: vanilla glazed doughnut 295
  Adding food item: cinnamon doughnut 295
  Adding special food item: blueberry jelly doughnut 295
  Adding food item: chocolate doughnut 295
  Adding food item: powdered sugar doughnut 295
  Adding food item: raspberry jelly doughnut 295
  Adding food item: boston cream doughnut 295
  Adding food item: eclair 295
  Adding food item: lemon pie slice 595
  Adding food item: lemon pie 4595
  Adding food item: cherry pie slice 595
  Adding food item: cherry pie 4595
  Adding food item: chocolate iced doughnut 295
  Adding special food item: plain bagel 395
  Adding food item: poppy seed bagel 395
  Adding food item: garlic bagel 395
  Adding food item: blueberry bagel 395
○ [ec2-user@ip-10-0-1-223 python_3]$ 
```

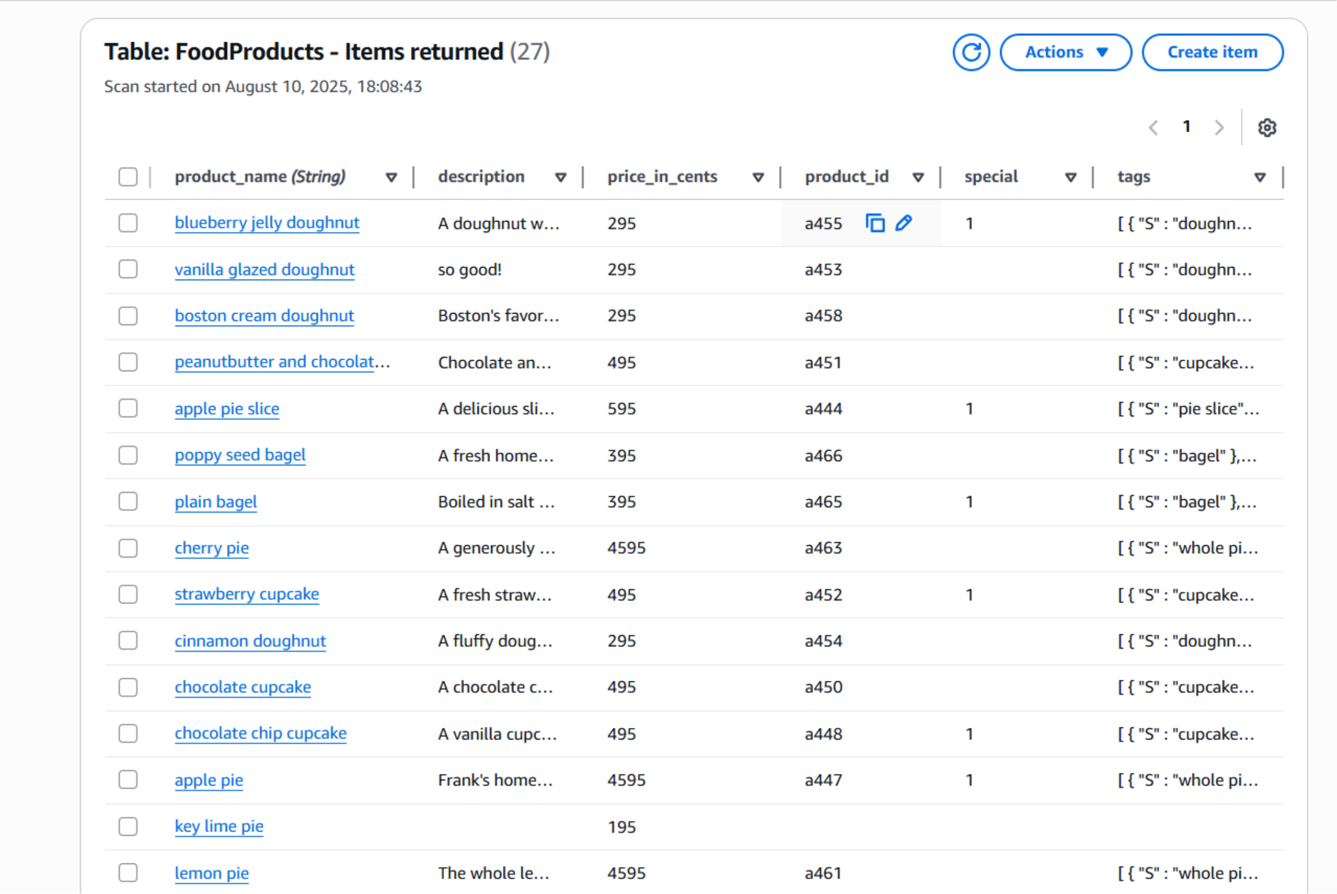


Table: FoodProducts - Items returned (27)

Scan started on August 10, 2025, 18:08:43

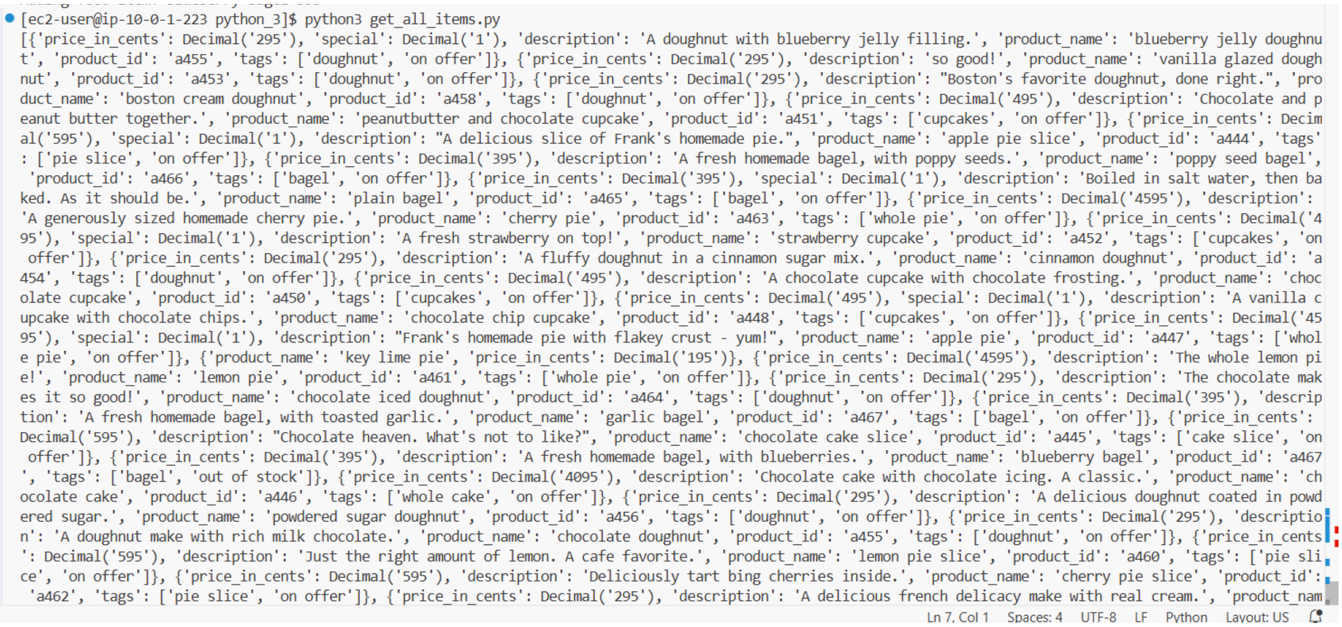
□	product_name (String)	description	price_in_cents	product_id	special	tags
□	blueberry jelly doughnut	A doughnut w…	295	a455	1	[{ "S" : "doughn…
□	vanilla glazed doughnut	so good!	295	a453		[{ "S" : "doughn…
□	boston cream doughnut	Boston's favor…	295	a458		[{ "S" : "doughn…
□	peanutbutter and chocolat…	Chocolate an…	495	a451		[{ "S" : "cupcake…
□	apple pie slice	A delicious sli…	595	a444	1	[{ "S" : "pie slice"…
□	poppy seed bagel	A fresh home…	395	a466		[{ "S" : "bagel" },…
□	plain bagel	Boiled in salt …	395	a465	1	[{ "S" : "bagel" },…
□	cherry pie	A generously …	4595	a463		[{ "S" : "whole pi…
□	strawberry cupcake	A fresh straw…	495	a452	1	[{ "S" : "cupcake…
□	cinnamon doughnut	A fluffy doug…	295	a454		[{ "S" : "doughn…
□	chocolate cupcake	A chocolate c…	495	a450		[{ "S" : "cupcake…
□	chocolate chip cupcake	A vanilla cupc…	495	a448	1	[{ "S" : "cupcake…
□	apple pie	Frank's home…	4595	a447	1	[{ "S" : "whole pi…
□	key lime pie		195			
□	lemon pie	The whole le…	4595	a461		[{ "S" : "whole pi…

36. 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)

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

```
python3 get_all_items.py
```



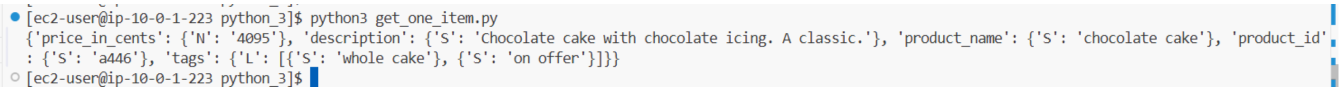
```
[ec2-user@ip-10-0-1-223 python_3]$ python3 get_all_items.py
[{'price_in_cents': Decimal('295'), 'special': Decimal('1'), 'description': 'A doughnut with blueberry jelly filling.', 'product_name': 'blueberry jelly doughnu
t', 'product_id': 'a455', 'tags': ['doughnut', 'on offer']}, {'price_in_cents': Decimal('295'), 'description': 'so good!', 'product_name': 'vanilla glazed dough
nut', 'product_id': 'a453', 'tags': ['doughnut', 'on offer']}, {'price_in_cents': Decimal('295'), 'description': "Boston's favorite doughnut, done right.", 'pro
duct_name': 'boston cream doughnut', 'product_id': 'a458', 'tags': ['doughnut', 'on offer']}, {'price_in_cents': Decimal('495'), 'description': 'Chocolate and p
eanut butter together.', 'product_name': 'peanutbutter and chocolate cupcake', 'product_id': 'a451', 'tags': ['cupcakes', 'on offer']}, {'price_in_cents': Decim
al('595'), 'special': Decimal('1'), 'description': "A delicious slice of Frank's homemade pie.", 'product_name': 'apple pie slice', 'product_id': 'a444', 'tags'
: ['pie slice', 'on offer']}, {'price_in_cents': Decimal('395'), 'description': 'A fresh homemade bagel, with poppy seeds.', 'product_name': 'poppy seed bagel',
 'product_id': 'a466', 'tags': ['bagel', 'on offer']}, {'price_in_cents': Decimal('395'), 'special': Decimal('1'), 'description': 'Boiled in salt water, then ba
ked. As it should be.', 'product_name': 'plain bagel', 'product_id': 'a465', 'tags': ['bagel', 'on offer']}, {'price_in_cents': Decimal('4595'), 'description':
'A generously sized homemade cherry pie.', 'product_name': 'cherry pie', 'product_id': 'a463', 'tags': ['whole pie', 'on offer']}, {'price_in_cents': Decimal('4
95'), 'special': Decimal('1'), 'description': 'A fresh strawberry on top!', 'product_name': 'strawberry cupcake', 'product_id': 'a452', 'tags': ['cupcakes', 'on
 offer']}, {'price_in_cents': Decimal('295'), 'description': 'A fluffy doughnut in a cinnamon sugar mix.', 'product_name': 'cinnamon doughnut', 'product_id': 'a
454', 'tags': ['doughnut', 'on offer']}, {'price_in_cents': Decimal('495'), 'description': 'A chocolate cupcake with chocolate frosting.', 'product_name': 'choc
olate cupcake', 'product_id': 'a450', 'tags': ['cupcakes', 'on offer']}, {'price_in_cents': Decimal('495'), 'special': Decimal('1'), 'description': 'A vanilla c
upcake with chocolate chips.', 'product_name': 'chocolate chip cupcake', 'product_id': 'a448', 'tags': ['cupcakes', 'on offer']}, {'price_in_cents': Decimal('45
95'), 'special': Decimal('1'), 'description': "Frank's homemade pie with flakey crust - yum!", 'product_name': 'apple pie', 'product_id': 'a447', 'tags': ['whol
e pie', 'on offer']}, {'product_name': 'key lime pie', 'price_in_cents': Decimal('195')}, {'price_in_cents': Decimal('4595'), 'description': 'The whole lemon pi
e!', 'product_name': 'lemon pie', 'product_id': 'a461', 'tags': ['whole pie', 'on offer']}, {'price_in_cents': Decimal('295'), 'description': 'The chocolate mak
es it so good!', 'product_name': 'chocolate iced doughnut', 'product_id': 'a464', 'tags': ['doughnut', 'on offer']}, {'price_in_cents': Decimal('395'), 'descrip
tion': 'A fresh homemade bagel, with toasted garlic.', 'product_name': 'garlic bagel', 'product_id': 'a467', 'tags': ['bagel', 'on offer']}, {'price_in_cents':
Decimal('595'), 'description': "Chocolate heaven. What's not to like?", 'product_name': 'chocolate cake slice', 'product_id': 'a445', 'tags': ['cake slice', 'on
 offer']}, {'price_in_cents': Decimal('395'), 'description': 'A fresh homemade bagel, with blueberries.', 'product_name': 'blueberry bagel', 'product_id': 'a467
', 'tags': ['bagel', 'out of stock']}, {'price_in_cents': Decimal('4095'), 'description': 'Chocolate cake with chocolate icing. A classic.', 'product_name': 'ch
ocolate cake', 'product_id': 'a446', 'tags': ['whole cake', 'on offer']}, {'price_in_cents': Decimal('295'), 'description': 'A delicious doughnut coated in powd
ered sugar.', 'product_name': 'powdered sugar doughnut', 'product_id': 'a456', 'tags': ['doughnut', 'on offer']}, {'price_in_cents': Decimal('295'), 'descriptio
n': 'A doughnut make with rich milk chocolate.', 'product_name': 'chocolate doughnut', 'product_id': 'a455', 'tags': ['doughnut', 'on offer']}, {'price_in_cents
': Decimal('595'), 'description': 'Just the right amount of lemon. A cafe favorite.', 'product_name': 'lemon pie slice', 'product_id': 'a460', 'tags': ['pie sli
ce', 'on offer']}, {'price_in_cents': Decimal('595'), 'description': 'Deliciously tart bing cherries inside.', 'product_name': 'cherry pie slice', 'product_id':
 'a462', 'tags': ['pie slice', 'on offer']}, {'price_in_cents': Decimal('295'), 'description': 'A delicious french delicacy make with real cream.', 'product_nam
```

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
```

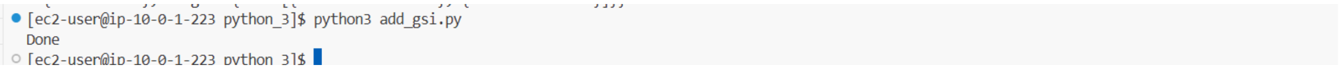


```
[ec2-user@ip-10-0-1-223 python_3]$ python3 get_one_item.py
{'price_in_cents': {'N': '4095'}, 'description': {'S': 'Chocolate cake with chocolate icing. A classic.'}, 'product_name': {'S': 'chocolate cake'}, 'product_id'
: {'S': 'a446'}, 'tags': {'L': [{'S': 'whole cake'}, {'S': 'on offer'}]}}
[ec2-user@ip-10-0-1-223 python_3]$
```

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).



```
● [ec2-user@ip-10-0-1-223 python_3]$ python3 add_gsi.py
  Done
○ [ec2-user@ip-10-0-1-223 python_3]$ 
```

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:

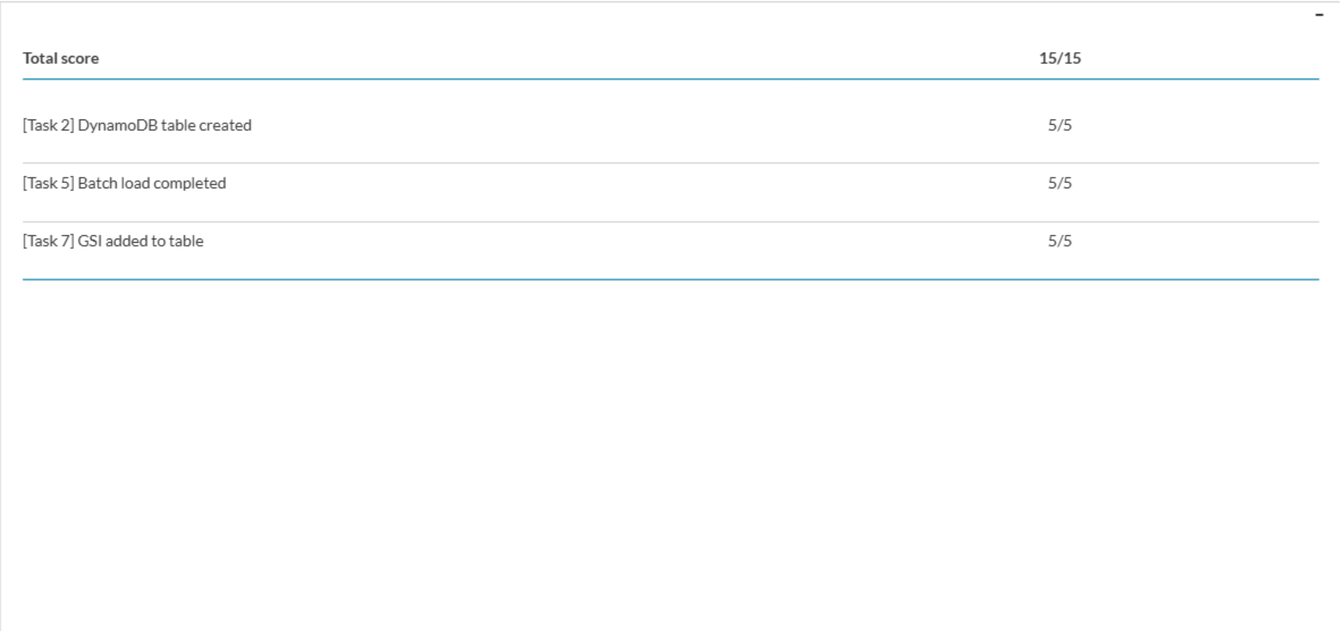
```
python3 scan_with_filter.py
```

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.

	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	**Aim: Create, edit, and update the database using AWS DynamoDB along with global index using python SDK.**	
Experiment No: 04	**Date:**	**Enrolment No: 92200133030**

Result :-



Total score	15/15
[Task 2] DynamoDB table created	5/5
[Task 5] Batch load completed	5/5
[Task 7] GSI added to table	5/5