**Homework 2**

***Python Dynamo Database Interface***

**Student**: Patrick Walsh

**School**: University of Maryland Global Campus

**Course**: SDEV 400 6980

**Date**: 7/10/2021

**Professor**: Dr. Craig Poma

1. (35points) Using your AWS Educate Cloud9 AWS CLI environment create a Table named Sensors with a Hash Key named Sensor. Sensor is an attribute that uniquely identifies an item in the database. Sensor will be of type String. Additional attributes that may be included in each database item include:

* SensorDescription(String) –A string that describes the Sensor
* ImageFile (String) –A filename that provides an image of the sensor. The filename is fully qualified to include to path location. (e.g. /Sensors/images/acoustic8-elementarray.jpg)
* SampleRate (Number) –The sample rate of the sensor (e.g. 2048)
* Locations (Set) –A set of locations where this Sensor has been used to collect data. (e.g. {Aberdeen MD, Warren MI, Orlando FL})

After successfully creating the table, continue to use the AWS-CLI to load at least 20 different sensor items. The sensor items should be stored in a JSON file. When creating the JSON file, be sure to not include all attributes for each item. For example, all items must have the Sensor attribute, but some items might be missing the ImageFile, Sample, Locations or SensorDescription attributes. In addition, add additional attributes (of your choosing), on the fly, to at least 5 of the items. Finally, while still using the AWS-CLI, write a statement that prints all items in the table. Provide and describe your AWS-CLI commands used for this exercise. Provide screen captures supporting the successful execution of each command.

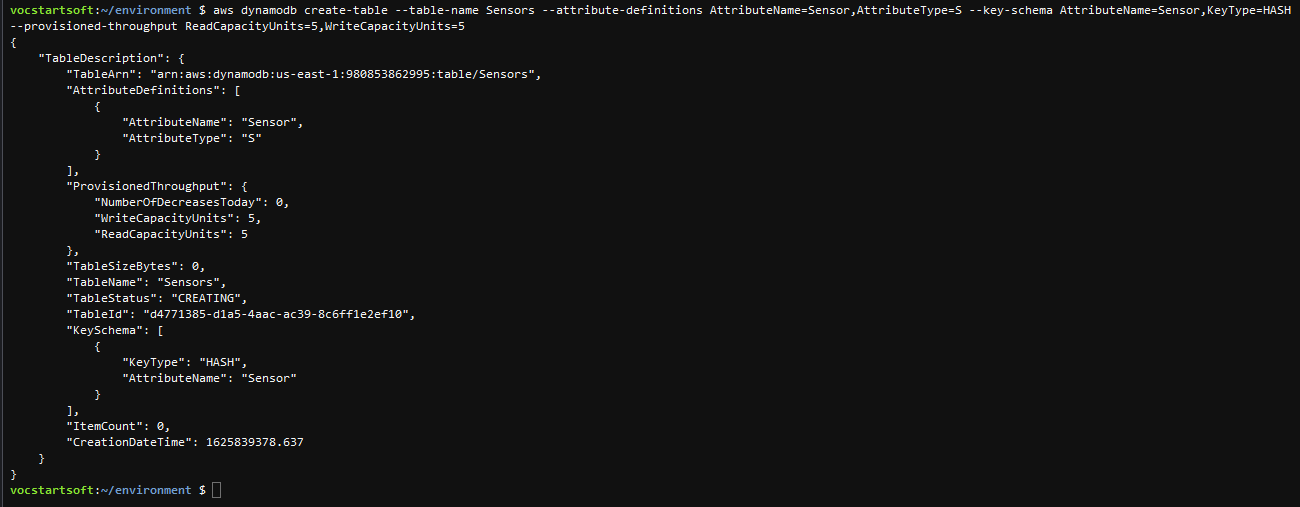
To create the DynamoDB table named Sensors from the AWS Cloud9 terminal, I ran the following command with the AWS CLI:

*aws dynamodb create-table –table-name Sensors –attribute-definitions AttributeName=Sensor,AttributeType=S –key-schema AttributeName=Sensor,KeyType=HASH –provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5*

The command creates the table with a Hash Key named Sensor of Type string. Commands were taken from the website Dynobase (Wilinski, 2020b). The output can be seen in Figure 1.

**Figure 1**

*Screen capture showing AWS CLI command.*

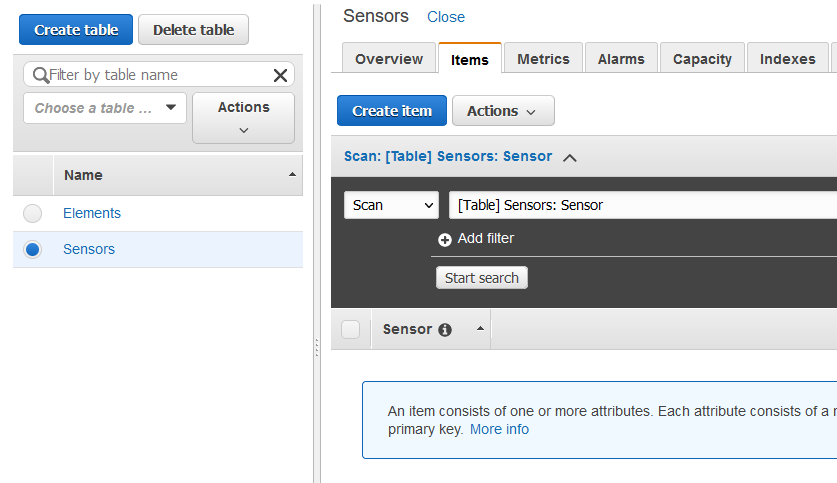


*Note*. From Cloud9 terminal.

In Figure 2, we also see that the empty table has been created in DynamoDB.

**Figure 2**

*Screen capture showing DynamoDB Sensors table created.*



*Note*. From DynamoDB.

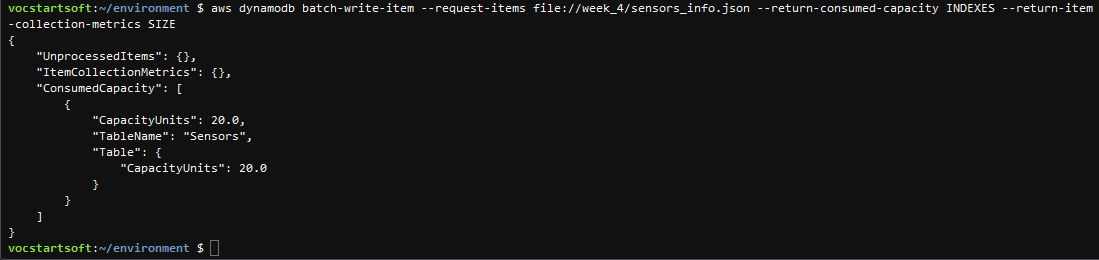
To load items into the Sensors table, I ran the following command which loads records from a .json file:

*aws dynamodb batch-write-item --request-items file://week\_4/sensors\_info.json --return-consumed-capacity INDEXES --return-item-collection-metrics SIZE*

The command loads the items as a batch so that multiple items inside the .json file can be loaded in one command. The output can be seen in Figure 3.

**Figure 3**

*Screen capture showing AWS CLI command.*

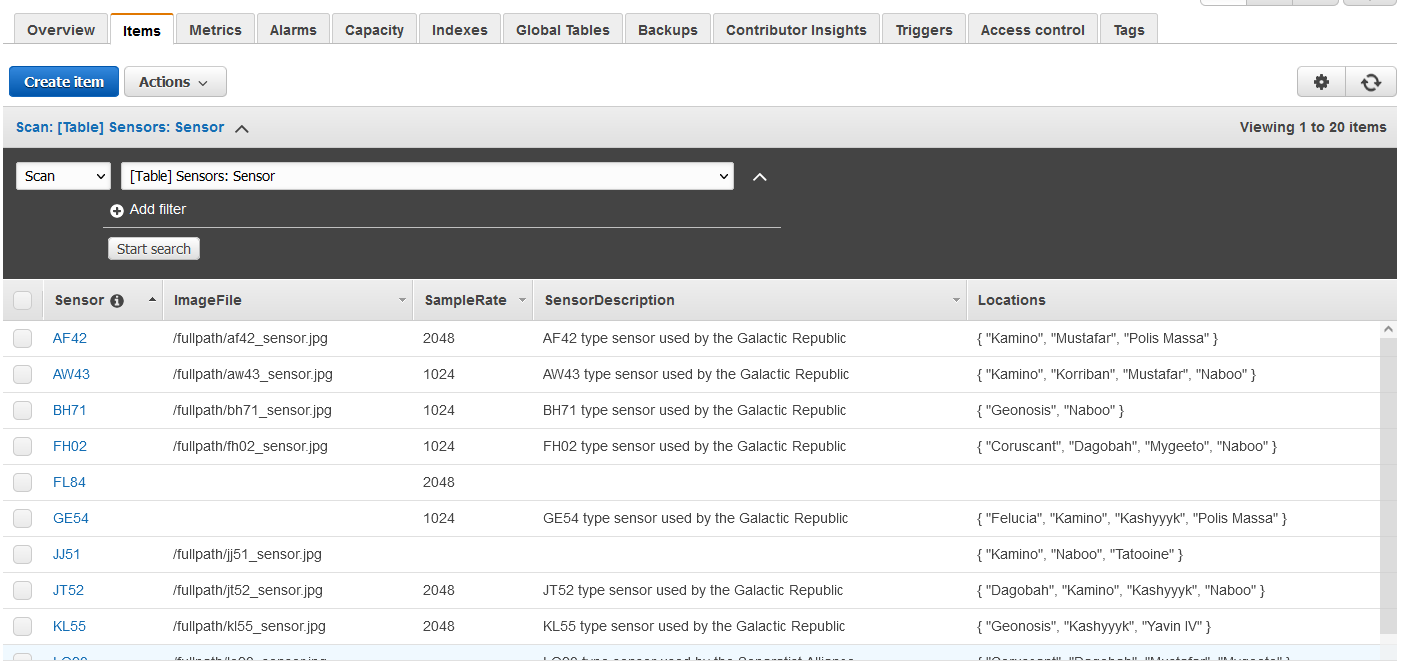


*Note*. From Cloud9 terminal.

In Figure 4, we see that the items have now been loaded into the Sensors table.

**Figure 4**

*Screen capture showing items loaded into DynamoDB Sensors table.*



*Note*. From DynamoDB.

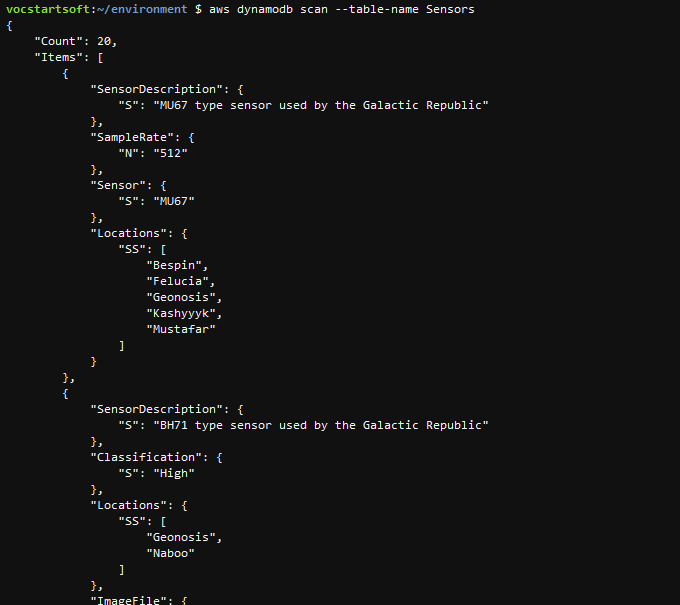
Finally, I ran the following command to return all items in the Sensors table:

*aws dynamodb scan --table-name Sensors*

The partial output is seen below in Figure 5.

**Figure 5**

*Screen capture showing AWS CLI command.*



*Note*. From Cloud9 terminal.

1. (40 points) Using the AWS SDK and Python within your AWS Educate Cloud9 environment, create a DynamoDB table and functionality as described below:

* Table named Courses to hold attribute including Subject (e.g. SDEV), CatalogNbr (e.g. 400), Title (e.g. Secure Programming in the Cloud), NumCredits (e.g. 3), and a CourseID (e.g. 001). The CourseID should represent the Hash Key for the Courses table.
* Input 10 Course items of your choice. Note that all attributes are required for each item entered in the Courses table.
* Provide a command line simple interface that allows a user to search for a title given the Subject and CatalogNbr. The search program should continue to loop until the user requests to exit. Also, if both Subject and CatalogNbr are not entered, the program should request the data be re-entered.

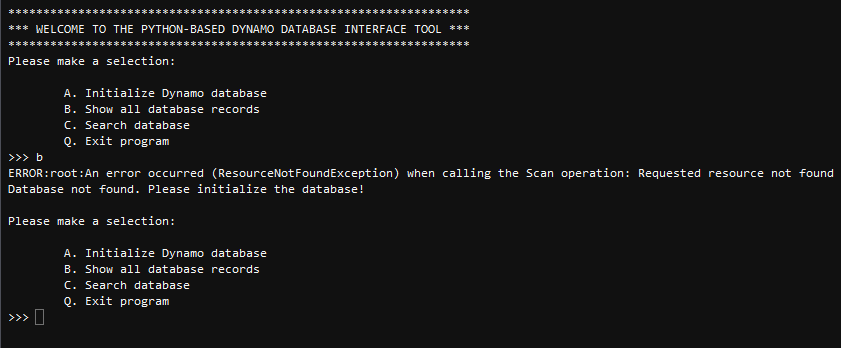
This test table shows a series of commands used to test the functionality of the Python-based DynamoDB interface program. Python code snippets were used and modified from the website Dynobase (Wilinski, 2020a).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Figure Reference** |
| 1 | b, c, sdev, 400, q | Messages saying database does not exist since it has not been initialized yet. Then a quit message when q is entered. | ERROR:root:An error occurred (ResourceNotFoundException) when calling the Scan operation: Requested resource not found  Database not found. Please initialize the database!  Exiting program.... | 6, 7, 8 |
| 2 | aws, a, b | Message saying to enter a valid response, then messages saying database is being initialized and adding records. Then a message showing all records in database. | Initializing database...  Adding records...  Database initialized!  Showing all records for Courses table | 9, 10 |
| 3 | c, <Enter>, asd, <Enter>, asd, asd, n | Messages asking to enter Subject and CatalogNbr. When a blank entry is attempted for either Subject or CatalogNbr there should be an error message. When a record then doesn’t exist is searched, the program should say so. | Enter search criteria...  Please enter a Subject  Subject is required.  Please enter a Subject  Now enter a CatalogNbr  CatalogNbr is required.  Now enter a CatalogNbr  Course not found!  Would you like to search for another record?  Invalid response!  Would you like to search for another record? | 11 |
| 4 | c, sdev, 325, y, hist, 250, n | Messages asking to enter Subject and CatalogNbr. Should show record for SDEV325 (not case sensitive), but should not show a record for HIST250. Will go back to main menu when user enters that they do not want to search for another record. | Enter search criteria...  Please enter a Subject  Now enter a CatalogNbr  [shows record]  Would you like to search for another record?  Enter search criteria...  Please enter a Subject  Now enter a CatalogNbr  Course not found!  Would you like to search for another record? | 12 |

**Test Case 1**

**Figure 6**

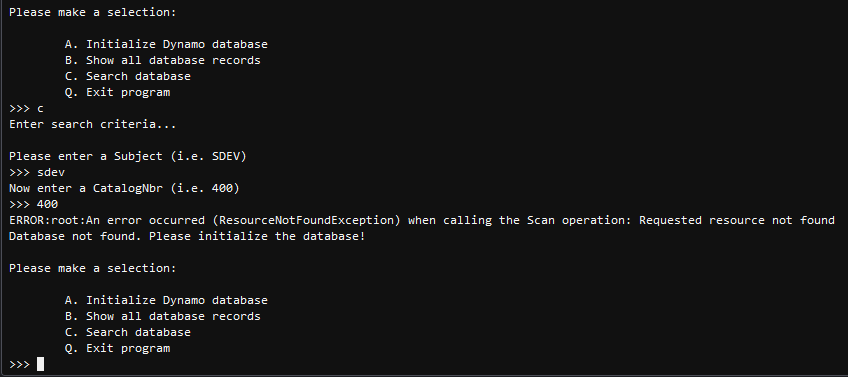
*Screen capture showing Python DynamoDB interface.*



*Note*. From Cloud9 terminal.

**Figure 7**

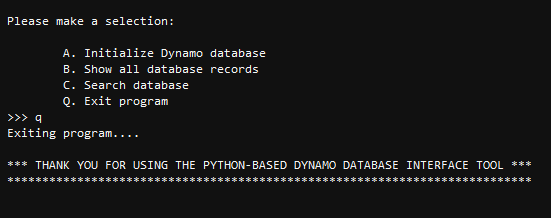
*Screen capture showing Python DynamoDB interface.*



*Note*. From Cloud9 terminal.

**Figure 8**

*Screen capture showing Python DynamoDB interface.*

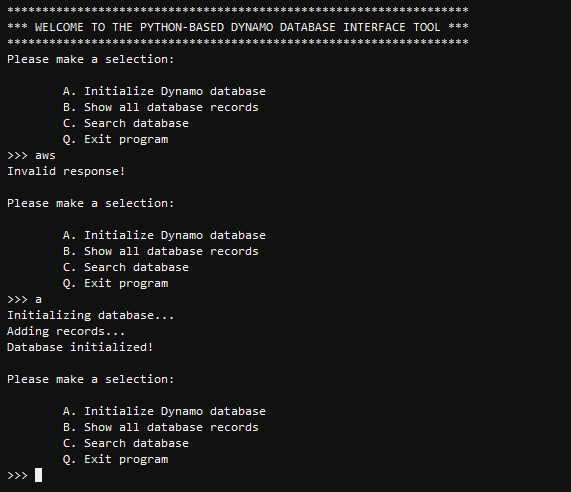


*Note*. From Cloud9 terminal.

**Test Case 2**

**Figure 9**

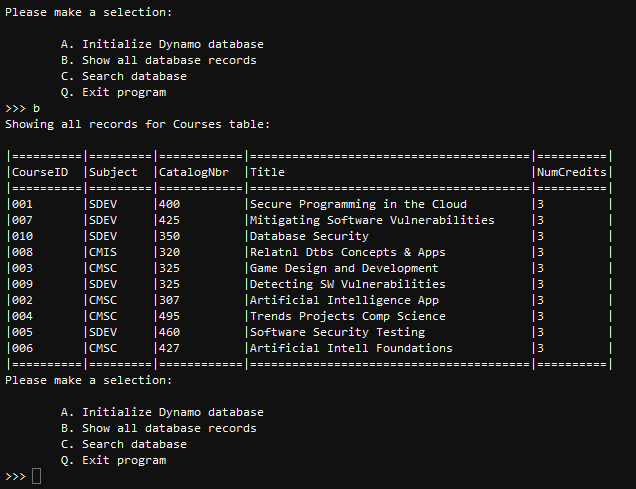
*Screen capture showing Python DynamoDB interface.*



*Note*. From Cloud9 terminal.

**Figure 10**

*Screen capture showing Python DynamoDB interface.*

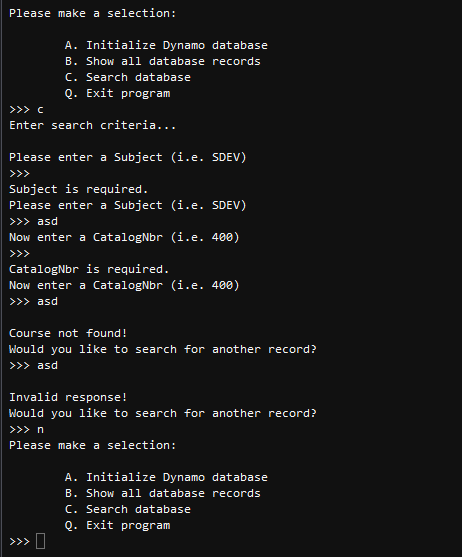


*Note*. From Cloud9 terminal.

**Test Case 3**

**Figure 11**

*Screen capture showing Python DynamoDB interface.*

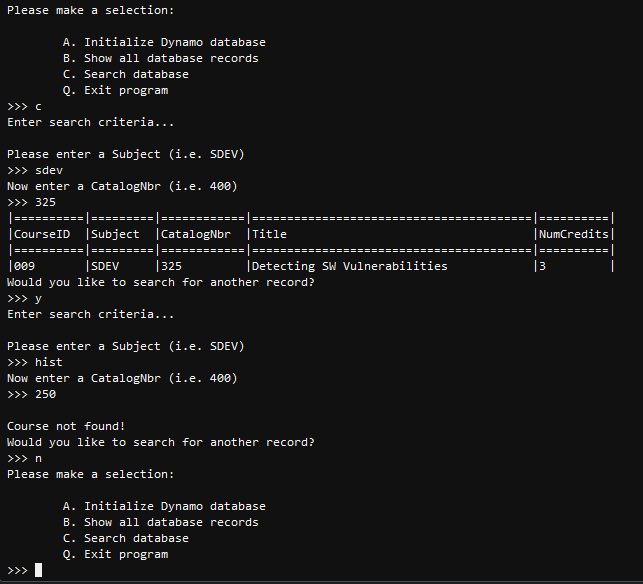


*Note*. From Cloud9 terminal.

**Test Case 4**

**Figure 12**

*Screen capture showing Python DynamoDB interface.*



*Note*. From Cloud9 terminal.

1. (15 points) Use the AWS CLI to delete all DynamoDB tables that were created for this exercise. Provide AWS CLI responses and the commands used to successfully complete this task in your report.

I used the following commands to delete both tables used in this assignment:

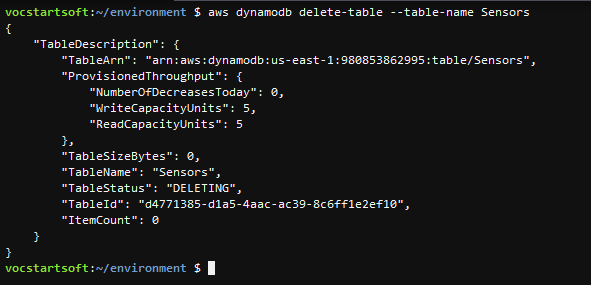
*aws dynamodb delete-table --table-name Sensors*

*aws dynamodb delete-table --table-name Courses*

The responses can be seen in Figures 13, 14, and 15.

**Figure 13**

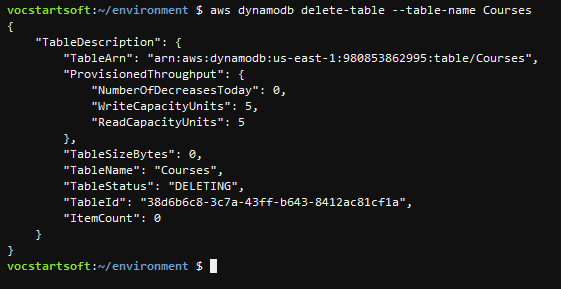
*Screen capture showing AWS CLI command.*



*Note*. From Cloud9 terminal.

**Figure 14**

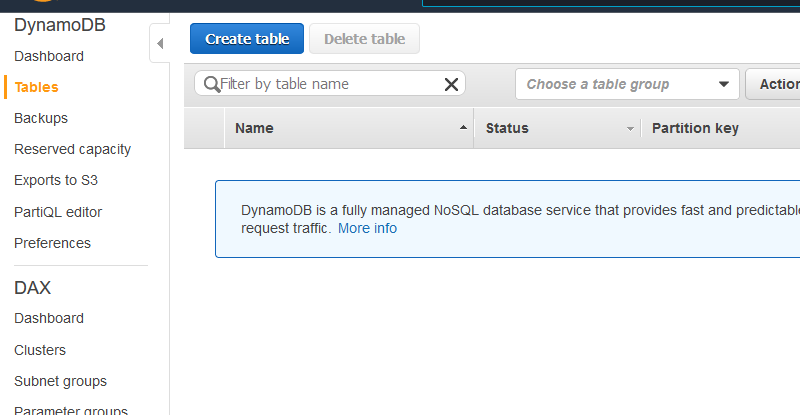
*Screen capture showing AWS CLI command.*



*Note*. From Cloud9 terminal.

**Figure 15**

*Screen capture showing all tables deleted.*



*Note*. From DynamoDB.

1. (10 points) Submit your report in Word or PDF format will all required components no later than the due date.

**References**

Wilinski, R. (2020a, February 27). *Boto3 is a Python library for AWS, which helps interacting with their services including DynamoDB*. Dynobase. https://dynobase.dev/dynamodb-python-with-boto3/

Wilinski, R. (2020b, November 29). *This cheat sheet covers the most important DynamoDB CLI query examples and table manipulation commands*. Dynobase. https://dynobase.dev/dynamodb-cli-query-examples/