Homework 2

Abass Kanu

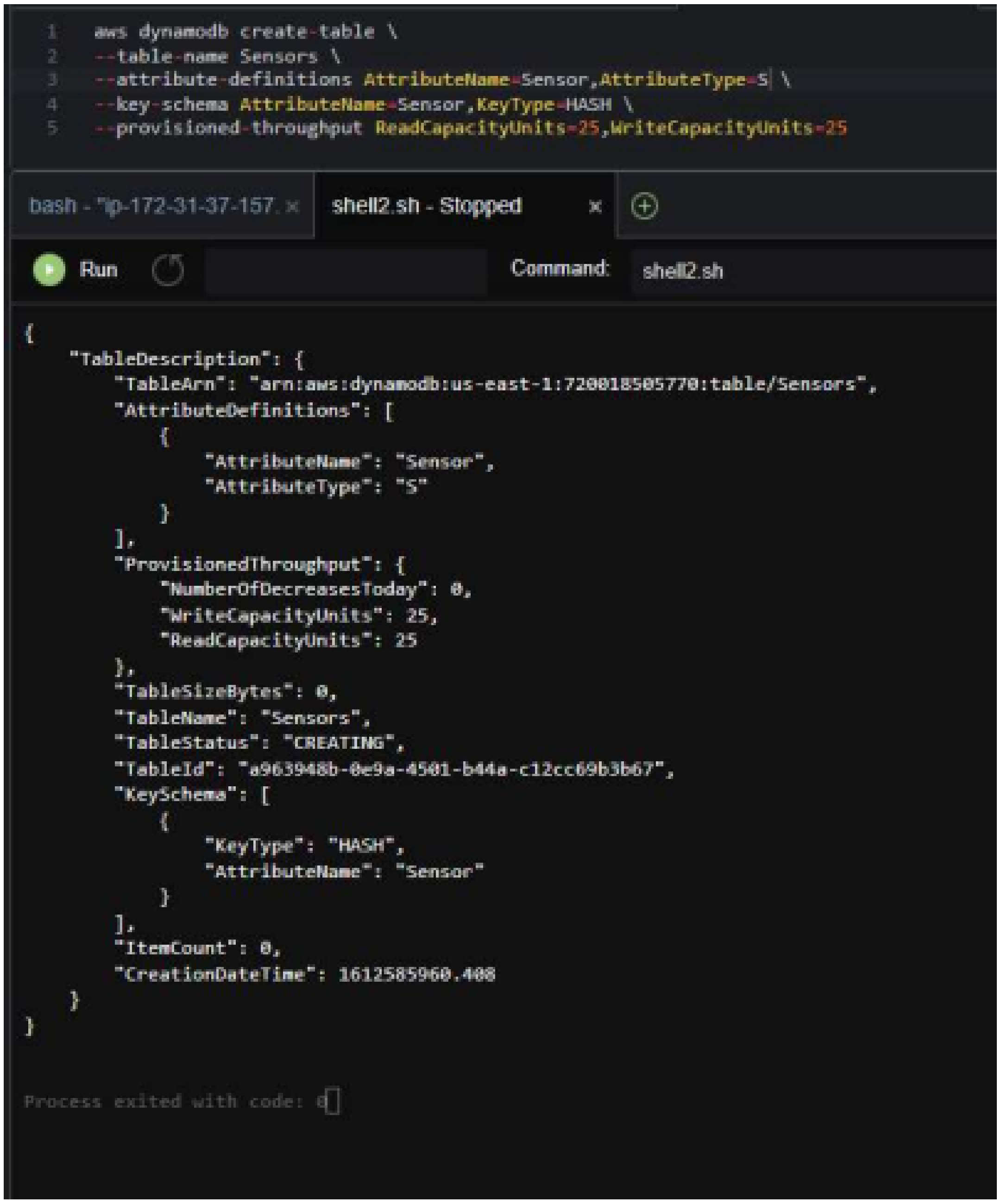
September 14th, 2021

Prof. Errol Waithe

Table of Contents for Test Cases

1. Creating the Sensor table

Figure 1 shows the successful creation of the Sensor table using the create-table, with a Hash Key named Sensor and a read/write capacity of 25 items.



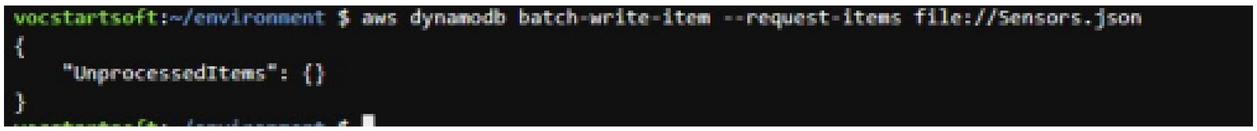
```
aws dynamodb create-table \
--table-name Sensors \
--attribute-definitions AttributeName=Sensor,AttributeType=S \
--key-schema AttributeName=Sensor,KeyType=HASH \
--provisioned-throughput ReadCapacityUnits=25,WriteCapacityUnits=25
```

```
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-east-1:720018505770:table/Sensors",
        "AttributeDefinitions": [
            {
                "AttributeName": "Sensor",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 25,
            "ReadCapacityUnits": 25
        },
        "TableSizeBytes": 0,
        "TableName": "Sensors",
        "TableStatus": "CREATING",
        "TableId": "a963948b-0e9a-4501-b44a-c12cc69b3b67",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Sensor"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": 1612585960.408
    }
}


Process exited with code: 0
```

Figure 1, Creating the Sensor table

2. JSON file loading 20 sensor items

Figure 2 shows the successful insertion of 20 different sensor items from a created JSON file.

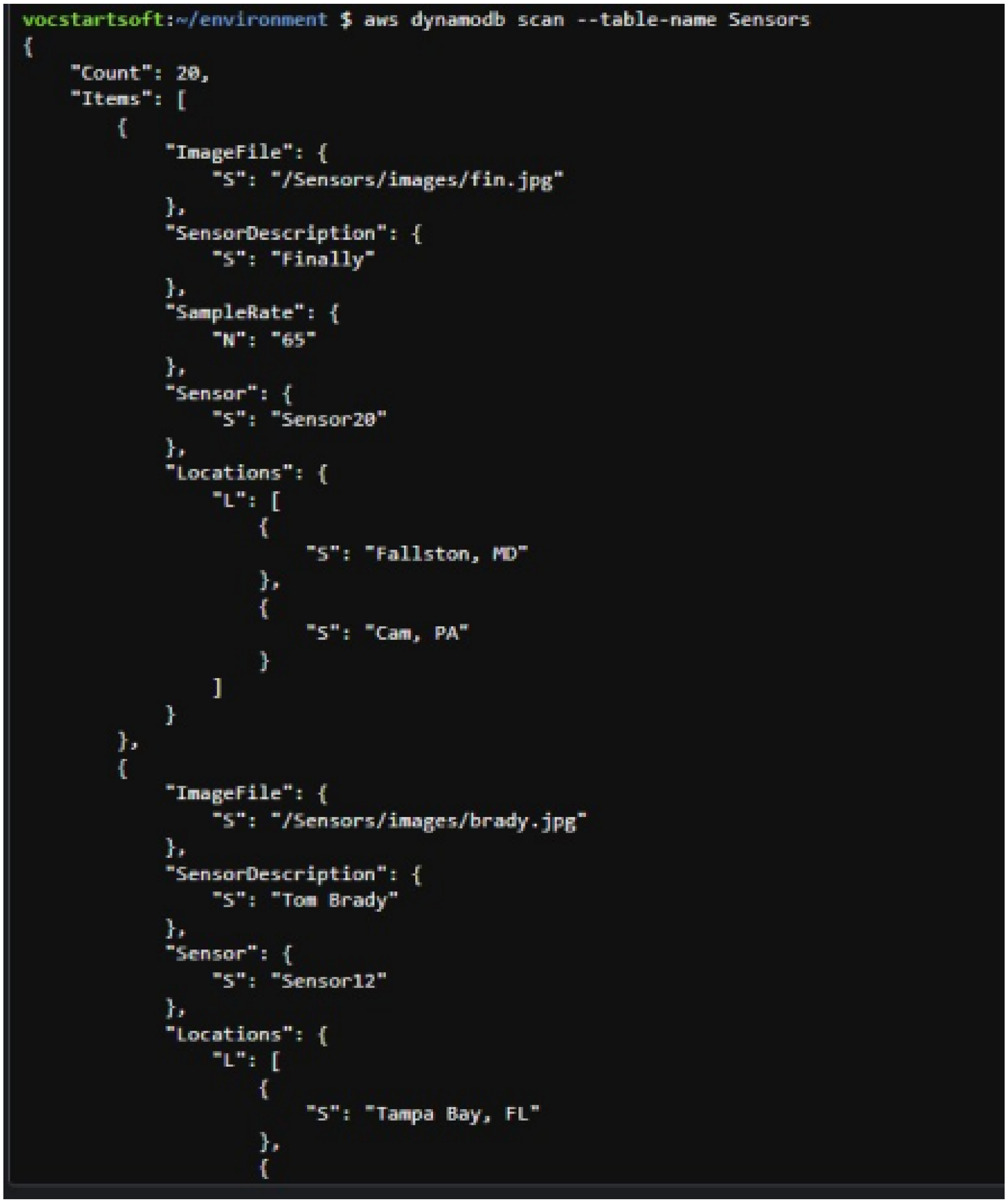


```
vocstartsoft:~/environment $ aws dynamodb batch-write-item --request-items file://Sensors.json
{
    "UnprocessedItems": {}
}
```

Figure 2, successful execution of JSON file

3. List sensors in Sensor table

Figure 3 shows the successful scanning of the Sensor table when scan –table-name Sensors was executed.

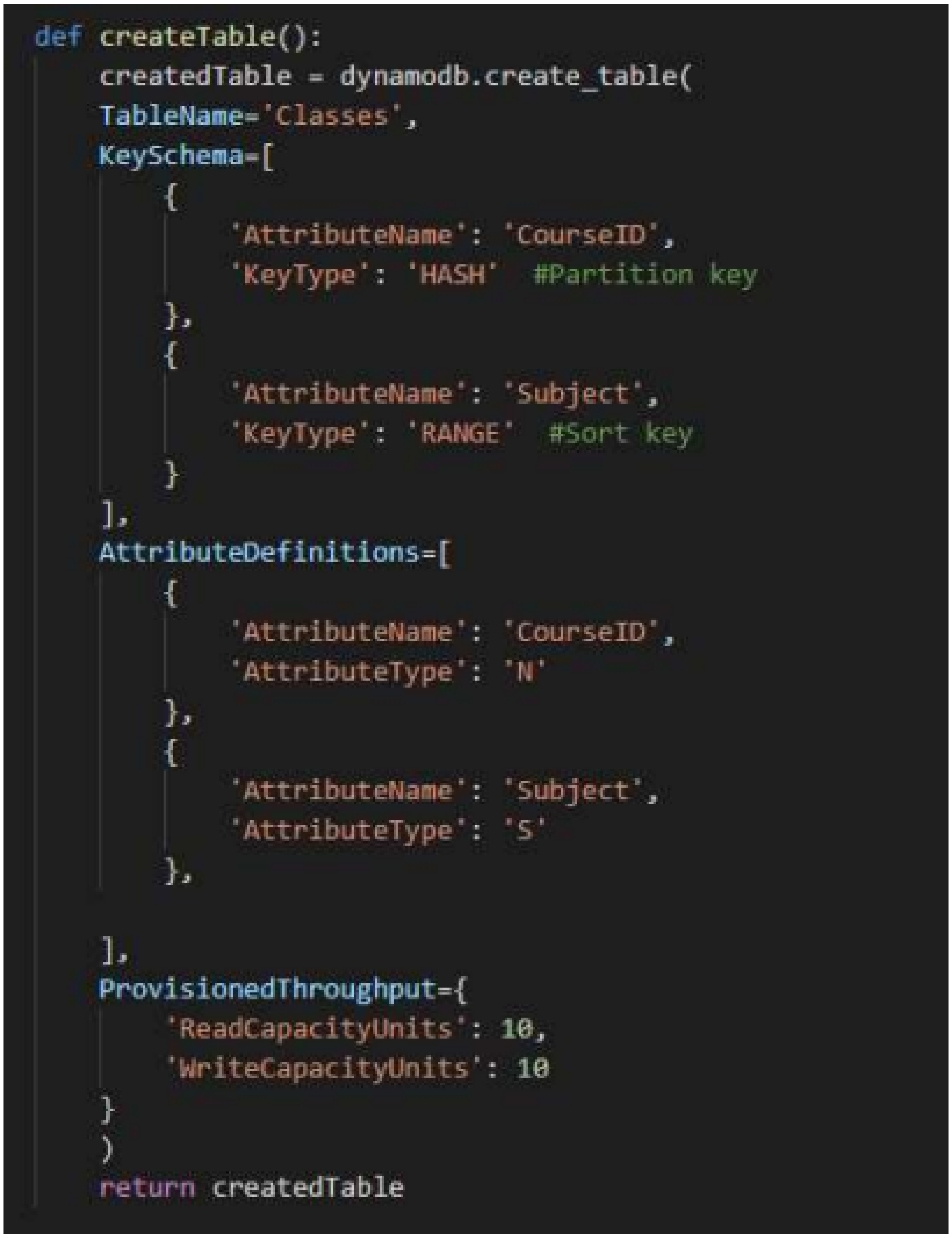


```
vocstartsoft:~/environment $ aws dynamodb scan --table-name Sensors
{
    "Count": 20,
    "Items": [
        {
            "ImageFile": {
                "S": "/Sensors/images/fin.jpg"
            },
            "SensorDescription": {
                "S": "Finally"
            },
            "SampleRate": {
                "N": "65"
            },
            "Sensor": {
                "S": "Sensor20"
            },
            "Locations": {
                "L": [
                    {
                        "S": "Fallston, MD"
                    },
                    {
                        "S": "Cam, PA"
                    }
                ]
            }
        },
        {
            "ImageFile": {
                "S": "/Sensors/images/brady.jpg"
            },
            "SensorDescription": {
                "S": "Tom Brady"
            },
            "Sensor": {
                "S": "Sensor12"
            },
            "Locations": {
                "L": [
                    {
                        "S": "Tampa Bay, FL"
                    },
                    {
```

Figure 3, Successful scanning of 20 items to sensor table

4. Create a table named Classes

Figure 4 shows the function that creates a table named Classes (MoviesCreateTable.py).

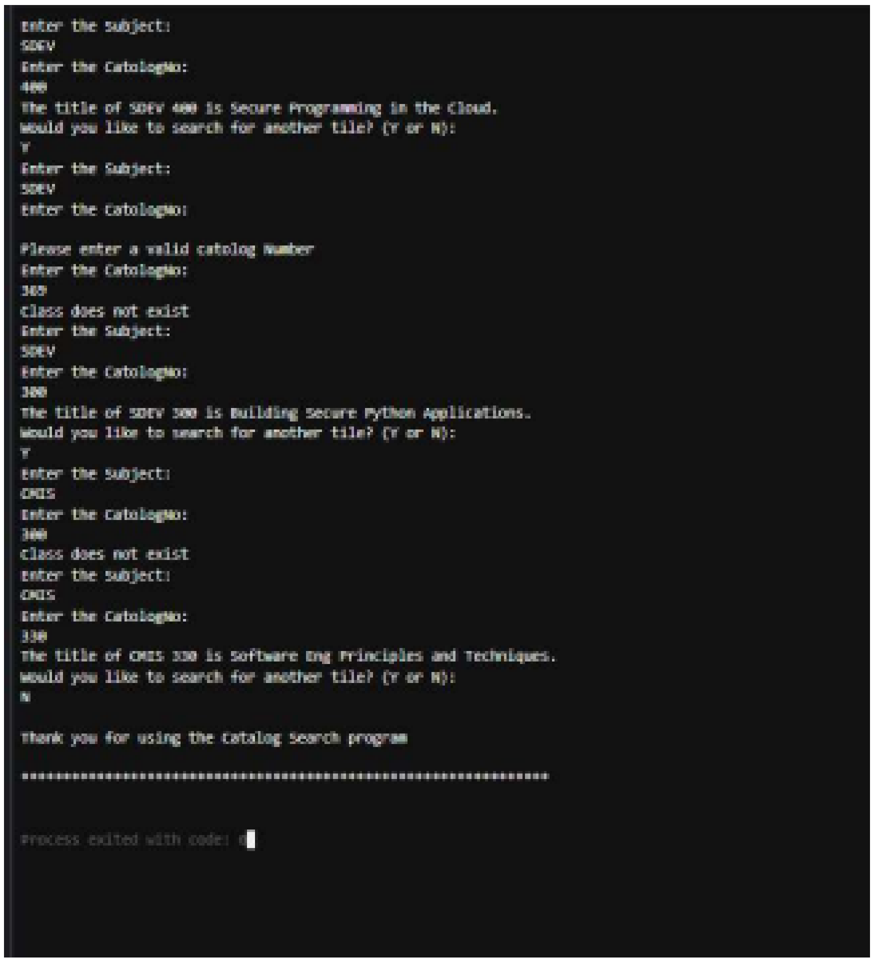


```python
def createTable():
    createdTable = dynamodb.create_table(
    TableName='Classes',
    KeySchema=[
        {
            'AttributeName': 'CourseID',
            'KeyType': 'HASH'  #Partition key
        },
        {
            'AttributeName': 'Subject',
            'KeyType': 'RANGE'  #Sort key
        }
    ],
    AttributeDefinitions=[
        {
            'AttributeName': 'CourseID',
            'AttributeType': 'N'
        },
        {
            'AttributeName': 'Subject',
            'AttributeType': 'S'
        },

    ],
    ProvisionedThroughput={
        'ReadCapacityUnits': 10,
        'WriteCapacityUnits': 10
    }
    )
    return createdTable
```

Figure 4, creating a table in python

5. Executing the program

Figure 5 shows the command line interface with various sample cases. If the user enters the information for a course in the catalog, the program will show the title of the course. If the user does not enter a subject or catalog number, the program will ask to enter one. If the user enters a class that does not exist, then the program will start back at the beginning. If the user selects that they would like to look for another course, then the program starts back at the beginning. If the user selects that they do not want to search for another title, then the program exits.

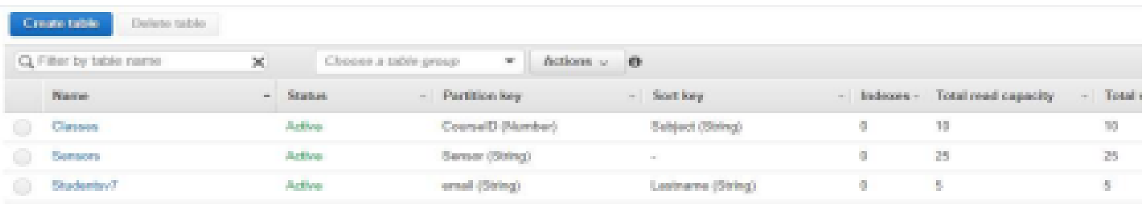


```
Enter the Subject:
SDEV
Enter the CatologNo:
400
The title of SDEV 400 is Secure Programming in the Cloud.
Would you like to search for another tile? (Y or N):
Y
Enter the Subject:
SDEV
Enter the CatologNo:

Please enter a valid catolog Number
Enter the CatologNo:
305
Class does not exist
Enter the Subject:
SDEV
Enter the CatologNo:
300
The title of SDEV 300 is Building Secure Python Applications.
Would you like to search for another tile? (Y or N):
Y
Enter the Subject:
CMIS
Enter the CatologNo:
300
Class does not exist
Enter the Subject:
CMIS
Enter the CatologNo:
330
The title of CMIS 330 is Software Eng Principles and Techniques.
Would you like to search for another tile? (Y or N):
N

Thank you for using the Catalog Search program

******************************************************************


Process exited with code: 0
```

Figure 5, command line for code

6. Delete tables from DynamoDB

Figure 6 shows the tables, Sensors and Classes, before deletion. Figure 7 shows the delete-table command being used for both tables. Figure 8 shows the table that is left in DynamoDB after deletion.

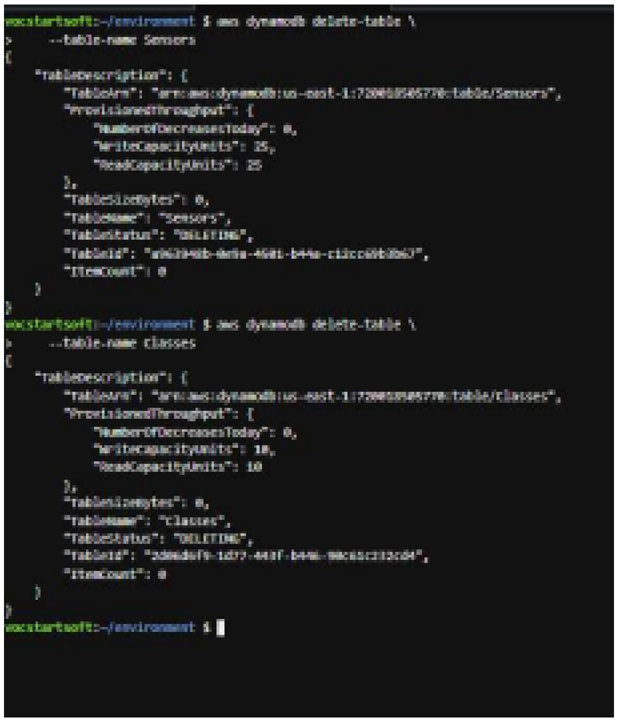


Create table · Delete table

Filter by table name · Choose a table group · Actions

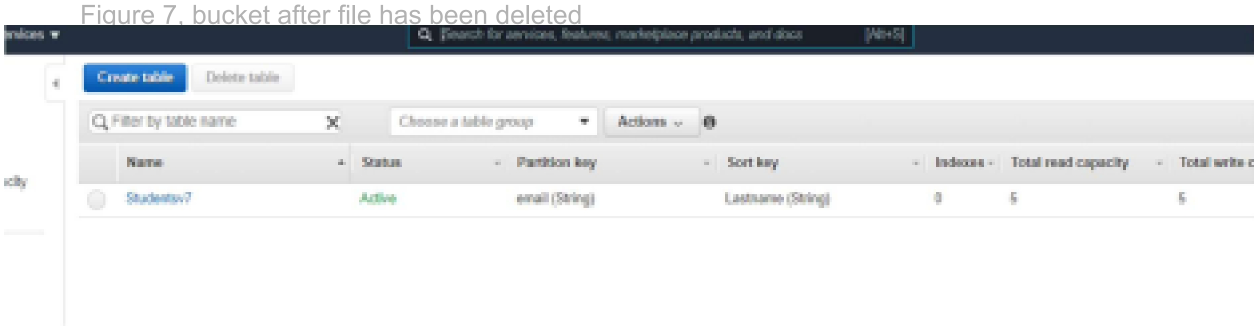
Name	Status	Partition key	Sort key	Indexes	Total read capacity	Total
Classes	Active	CourseID (Number)	Subject (String)	0	10	10
Sensors	Active	Sensor (String)	-	0	25	25
Studentsv7	Active	email (String)	Lastname (String)	0	5	5

Figure 6, Tables in DynamoDB before deletion



```
voclabs:~/environment $ aws dynamodb delete-table \
>     --table-name Sensors
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-east-1:[illegible]:table/Sensors",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 25,
            "ReadCapacityUnits": 25
        },
        "TableSizeBytes": 0,
        "TableName": "Sensors",
        "TableStatus": "DELETING",
        "TableId": "[illegible]",
        "ItemCount": 0
    }
}
voclabs:~/environment $ aws dynamodb delete-table \
>     --table-name Classes
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-east-1:[illegible]:table/Classes",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 10,
            "ReadCapacityUnits": 10
        },
        "TableSizeBytes": 0,
        "TableName": "Classes",
        "TableStatus": "DELETING",
        "TableId": "[illegible]",
        "ItemCount": 0
    }
}
voclabs:~/environment $
```

Figure 7, bucket after file has been deleted



Create table Delete table

Filter by table name Choose a table group Actions

Name	Status	Partition key	Sort key	Indexes	Total read capacity	Total write c
Studentsv7	Active	email (String)	Lastname (String)	0	5	5

Figure 8, Tables in DynamoDB after deletion

References

MoviesCreateTable.py[Source Code].http://aws.amazon.com/apache2.0/

MoviesItemsOps1.py[Source Code].http://aws.amazon.com/apache2.0/