**Lab 3**

**Galen J. Yanofsky**

**SDEV 400 7981**

**Dr. James Robertson**

**02 December 2017**

1. **Describe and differentiate with syntax and commands. How you would create in DynamoDB, the equivalent of the following SQL Schema?**

Both SQL and NoSQL databases offer different solutions depending on the business requirement that needs to be met. SQL stands for Structured Query Language and is used in Relational Databases (RDBMS). RDBMS are table based databases where data is represented in the form of rows and columns. RDBMS databases have a predefined structured, also known as a schema. RDBMS databases that employ SQL are vertically scalable and require more computing power as they grow in size.

NoSQL is a type of database that does not employ the use Structured Query Language. These types of databases use UnQL. UnQL use a variety of different languages to accomplish queries and database tasks. Where SQL databases are vertically scalable, NoSQL databases are horizontally scalable. As a NoSQL database increases in size, the number of servers and resources available for the data must also increase. NoSQL databases use a dynamic schema where data is unstructured. They are document based, with key-value pairs, graph databases or wide-column stores.

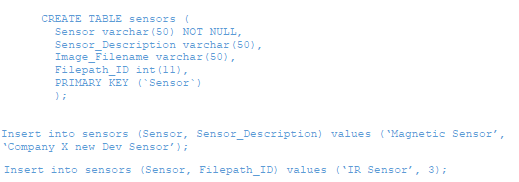
****

Figure 1: MySQL Script

The above screen capture shows the creation of a table in Structured Query Language. Below the creation command, it shows the user attempting to populate the table with corresponding unique data. The data ‘inserted’ is structured similarly to the schema that is defined above. This way the table populated properly. To configure DynamoDB to be able to perform the same operation in the CLI, I must first configure the user access to specify which account the database will be created under. The screen capture below shows this.

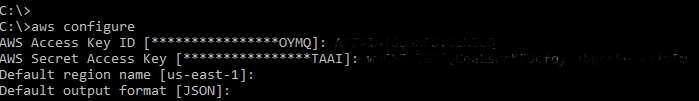


Figure 2: AWS Configure

Within AWS CLI the create-table command can be used to make a similar table as shown in the above SQL entry. The details of the command are below.

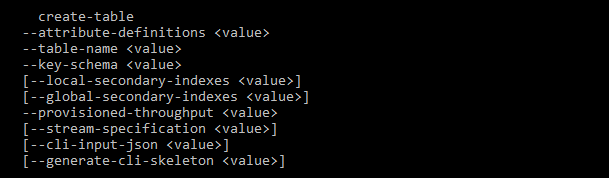


Figure 3: Create Table

I structure my table creation syntax using this structure. The AWS help is very comprehensive. One can learn whatever they need to by consulting the help feature. In order to create a duplicate to the table above the following command is structured within the CLI.



Figure 4: Create Table Command

As discussed above, the DynamoDB functions differently from standard SQL. Tables are only created with the attributes that are intended to be queried. Other attributes using JSON/shorthand population scripts to increase data variability. When we run the above shorthand command, the CLI returns the following:

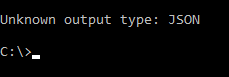


Figure 5: Confirmation

I then confirm that the table was created via the AWS console. Within AWS DynamoDB, my list of tables now shows that Sensors has been added. Shown below.

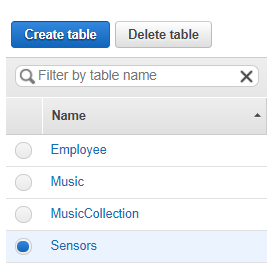


Figure 6: Create Table Shown In DynamoDB

Next, I populate the newly created database.

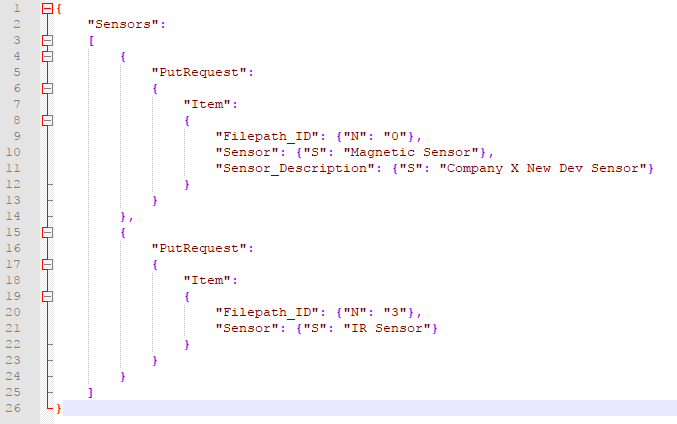


Figure 7: JSON Script

For this table, I made the mistake of making my numeric “Filepath\_ID” value my HASH keytype. This value ultimately determines how data is stored in my database. Because of this I had to add an arbitrary value of ‘0’ for Filepath\_ID to the first INSERT statement. The CLI returns the following to confirm that I successfully populated the database.



Figure 8:Write Item to DynamoDB

I then do an integrity check to make sure that the table was properly populated in DynamoDB console as well. The console also shows that the table has been populated.

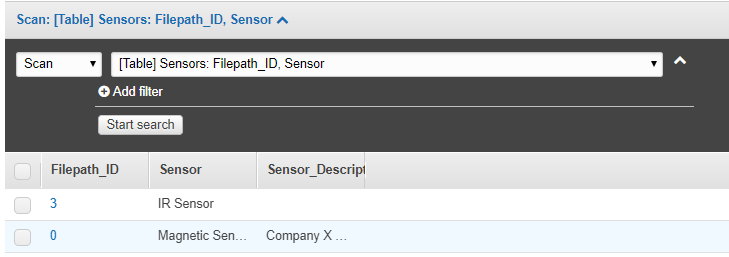


Figure 9: Sensor Records within DynamoDB

1. **Use AWS CLI commands to create an RDS instance running MySQL.**

The RDS MySQL database should have the following properties.

* db-name = SDEV400RDSTest
* DB instance identifier = “user-supplied”
* Master Username = “user-supplied”
* Master user Password = “user-supplied”
* instance-class = db.t2.micro
* Storage = 20 GB
* Engine = MySQL

I structure the following database create command in order to create a database with the above specifications. The CLI returns the following which confirms to me that the Db instance was successfully launched.

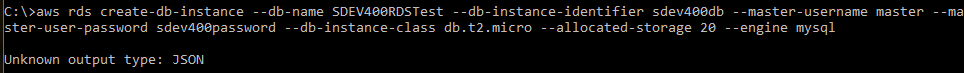


Figure 10: Create DB Instance

I then do an integrity check on the AWS console within the RDS service to ensure that I now have an additional database instance that is running.

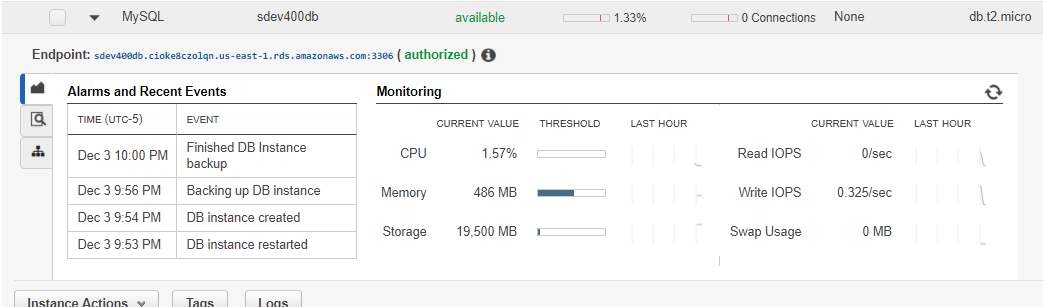


Figure 11: DB Overview

The RDS console shows that the database has been created and is running successfully. This means the CLI command was successful.

Next, I list the specifications of the database that was created. I type the following command into the CLI and it returns information about the database that was created.



Figure 12: Describe DB

  
The information is accurate in terms of the original database specifications.

Figure 13: JSON Script

To create the schema for the database and tables specified in the Lab 3 instructions, I choose to use MySQL Workbench. It is a handy GUI utility that can be used for comprehensive database administration, and configuration tasks. I begin by establishing the connection between the application and the RDS database that was created.

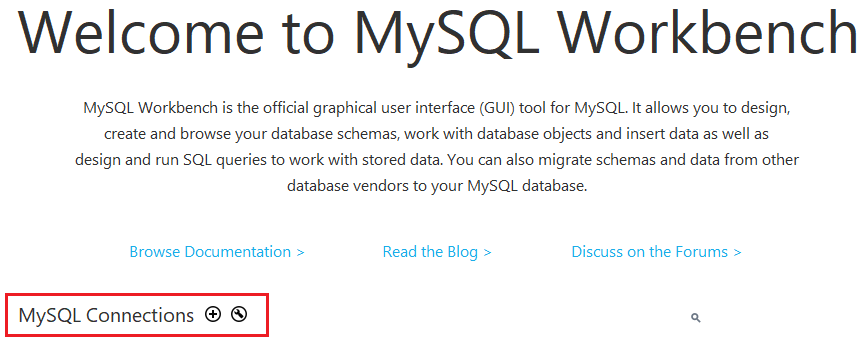


Figure 14: MySQL Workbench

I click the ‘+’ sign to add a database connection. An input dialog pane is displayed and I enter the details associated with the database. These details can either be copied from the CLI or the AWS management console. Below shows the filled out input dialog.

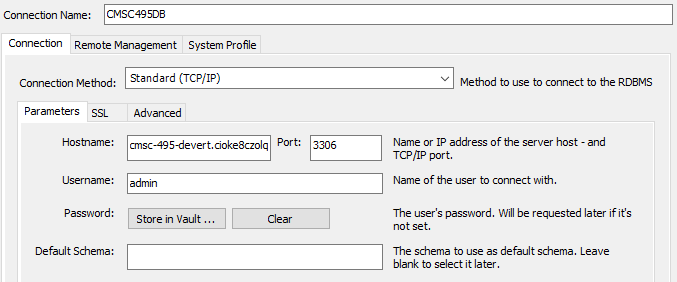


Figure 15

The proper details have been inputted as well as the proper credentials to be able to successfully login to the MySQL database. I then test the database connectivity. I am successfully able to connect to the MySQL database.

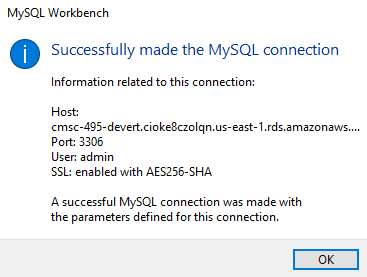


Figure 16

I create the Courses Table to hold the specified information.

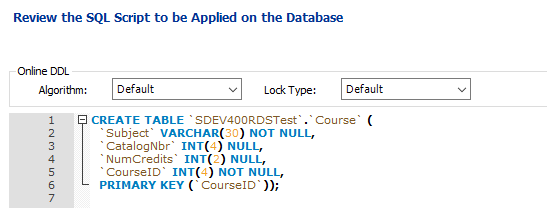


Figure 17

I create the Students Table to hold the specified information

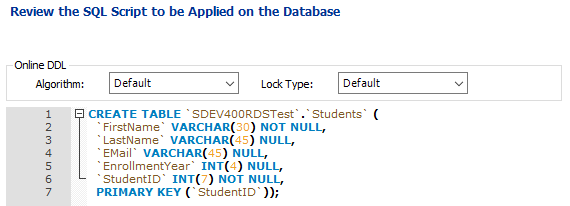


Figure 18

I create the StudentCourses Table to hold the specified information

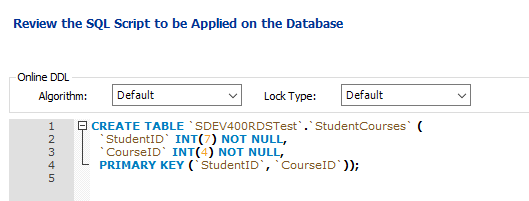


Figure 19

I then insert the data into the tables. The following data is inserted into the database and tables that were just created.

This inserts the following 5 records into the Courses Table

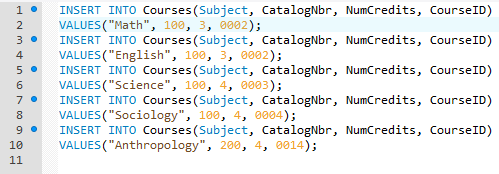


Figure 20

This inserts the following 10 records into the StudentCourses Table

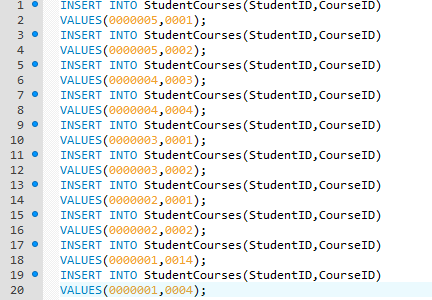


Figure 21

This inserts the following 5 records in Students Table

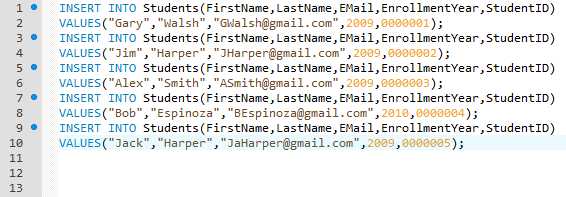


Figure 22

I connect to my database via the CLI using the following command



Figure 23

I then switch to the SDEV400RDSTest database by utilizing the ‘use’ command.

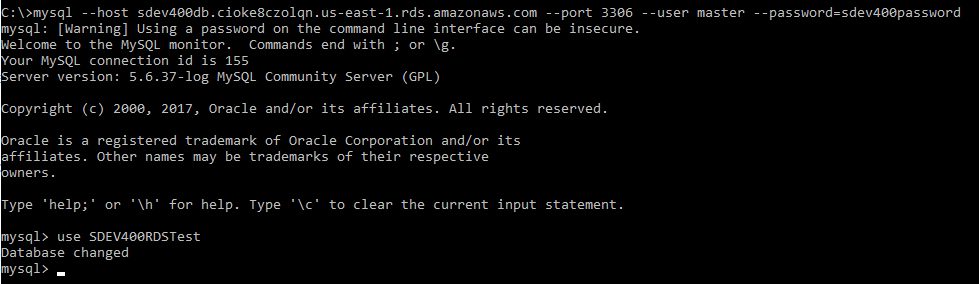


Figure 24

I query the entire database using a ‘SELECT \* FROM’ statement’.



Figure 25

The following results are returned in MySQL table format.

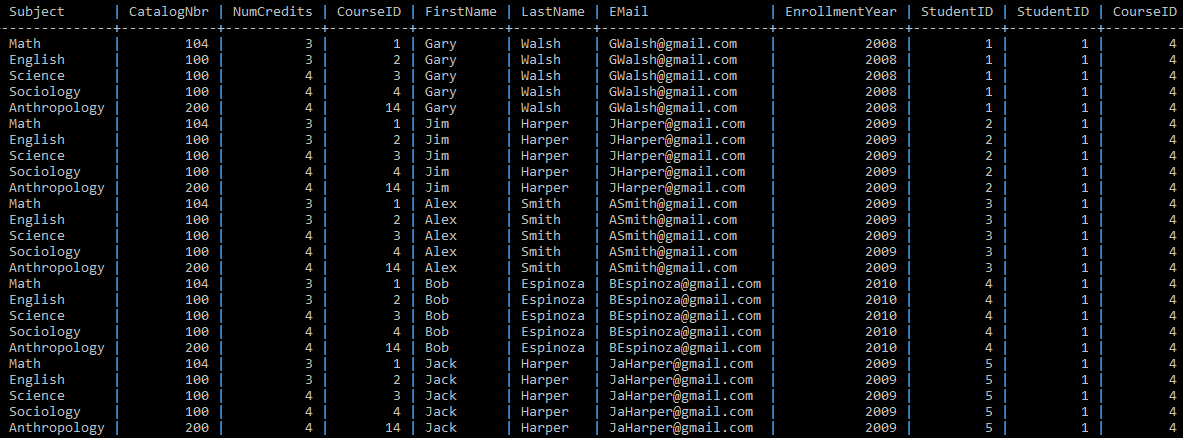


Figure 26

1. **Use DynamoDB to create identical tables to those in Part 2**

I structure each command using the ‘aws dynamodb create-table’ command. The following screen captures show each unique creation command and their respective outputs.

Create Courses Table:



Figure 27

Output Returned:



Figure 28

Create Students Table:



Figure 29

Output Returned:



Figure 30

Create StudentCourses Table:



Figure 31

Output Returned:



Figure 32

I do an integrity check using the DynamoDB CLI to ensure that the tables have been created.

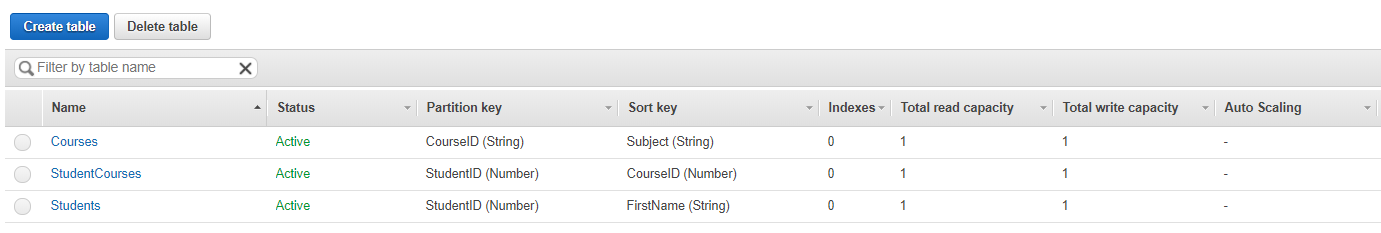


Figure 33

Each of the tables has been created with the appropriate HASH and RANGE values. Now I populate each table with the same data as in the MySQL database using a custom JSON Script.

I populate the Courses Table with 5 records. The JSON scripts are below.

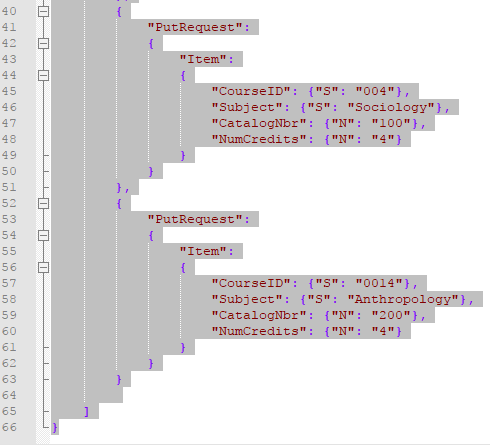


Figure 34

The following write confirmation is received within the CLI.

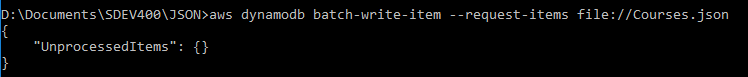


Figure 35

I check the AWS DynamoDB Management console to ensure that the records have been inserted. The screen capture of the results are below. Each record has been inserted into the database.

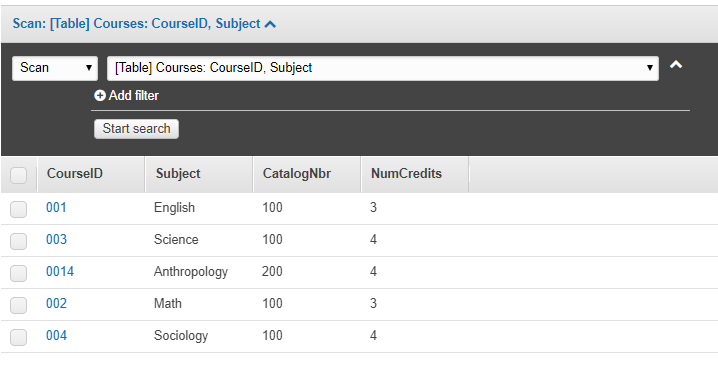


Figure 36

I populate the Students table with 5 records. The JSON Script is below.

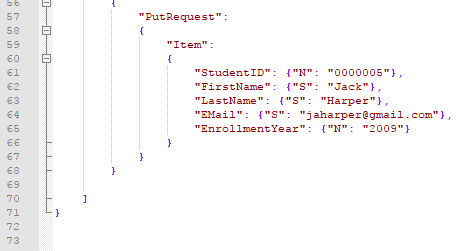
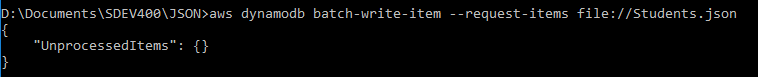
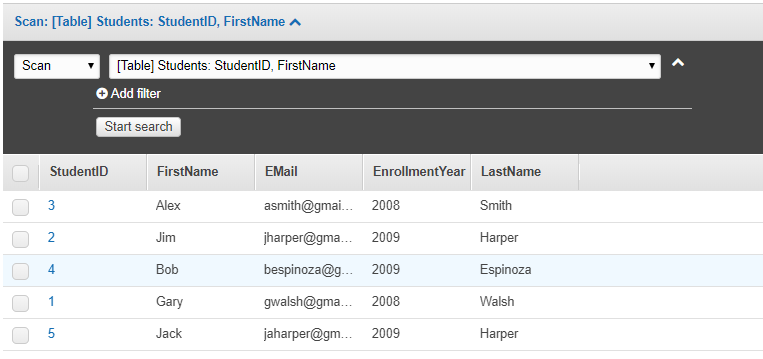


Figure 37

The following write confirmation is received within the CLI.



I check the AWS DynamoDB Management console to ensure that the records have been inserted. The screen capture of the results are below. Each record has been inserted into the database.



Finally, I populate the StudentCourses table. This table requires 10 records with 2 attributes per record. The JSON script is below:



Figure 38

The following write confirmation is received within the CLI.

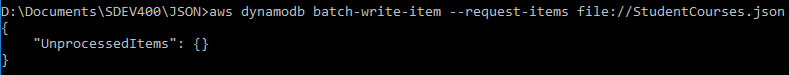


Figure 39

For this last table population, I check the AWS DynamoDB Management console to ensure that the records have been inserted. The screen capture of the results are below. Each record has been inserted into the database.

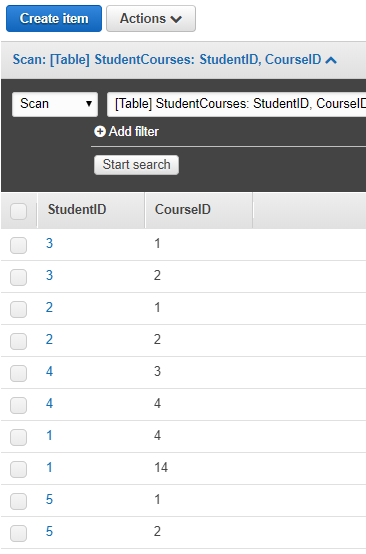


Figure 40

I then add two new records to the Students table. The records have additional attributes of ExpectedGradYear and GPA. The additional records are added using the command below.

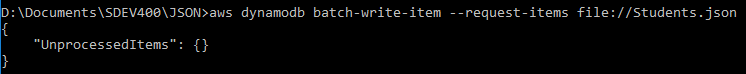


Figure 41

The JSON script use for this populating the tables with the additional records is also shown below.

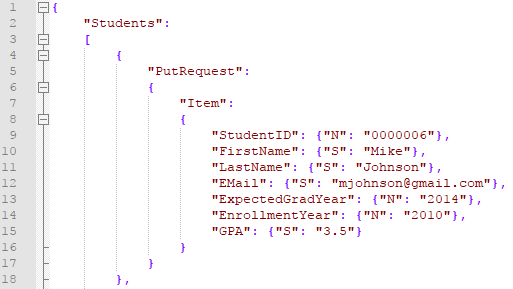


Figure 42

I then perform an integrity check to make sure that the additional records were added to the tables.



Figure 43

The additional records are contained in the red squares. I chose to use a “String” type for the GPA datatype because I wanted to make sure that DynamoDB would process the decimal appropriately. Next I update one of the additional records and change their. For this part of the Lab, I will choose to update the ‘LastName’ and ‘Email’ of the Tom Phillips entry. To complete this task, I use the ‘update-item’ command via the CLI.

I constructed the following command to update the record I specified above.



Figure 44

For some unknown reason, the CLI is returning this error. I will continue to troubleshoot this issue, until update any of the records above. The error that I am receiving is shown below. I have tried to follow the recommendations, but I am not making any positive progress.



Figure 45

Finally I delete all the tables that have been added to the test database for this Lab. The CLI and corresponding outputs are posted below.

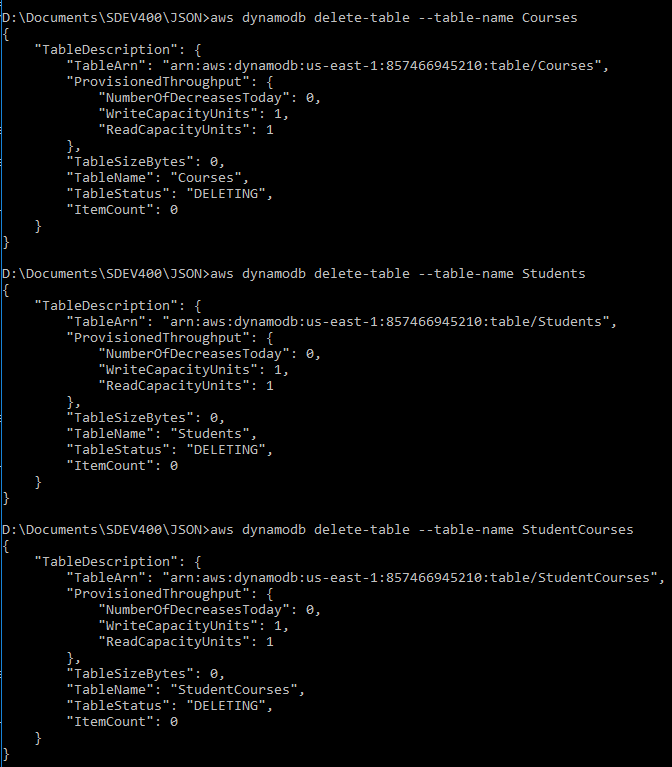


Figure 46

Finally I confirm these delete actions by visiting the DynamoDb service within the AWS management console. As is shown below, all tables within DynamoDB have been deleted.

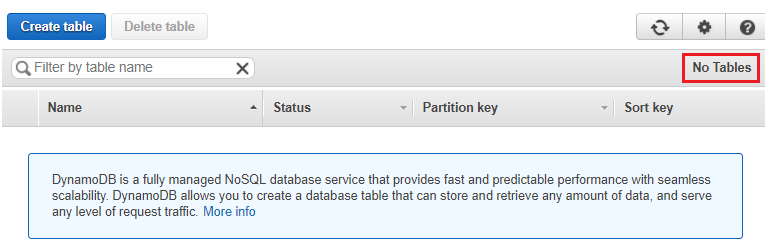


Figure 47

# References

Amazon. (2017, December 04). *Documentation*. Retrieved from AWS: https://aws.amazon.com/documentation/

Amazon. (n.d.). *AWS Training*. Retrieved from aws.amazon.com: https://aws.amazon.com/training/intro\_series/

Amazon. (n.d.). *Getting Started with Amazon Web Services (AWS)*. Retrieved from aws.amazon.com: https://aws.amazon.com/getting-started/