**Assignment 3 – Documentation**

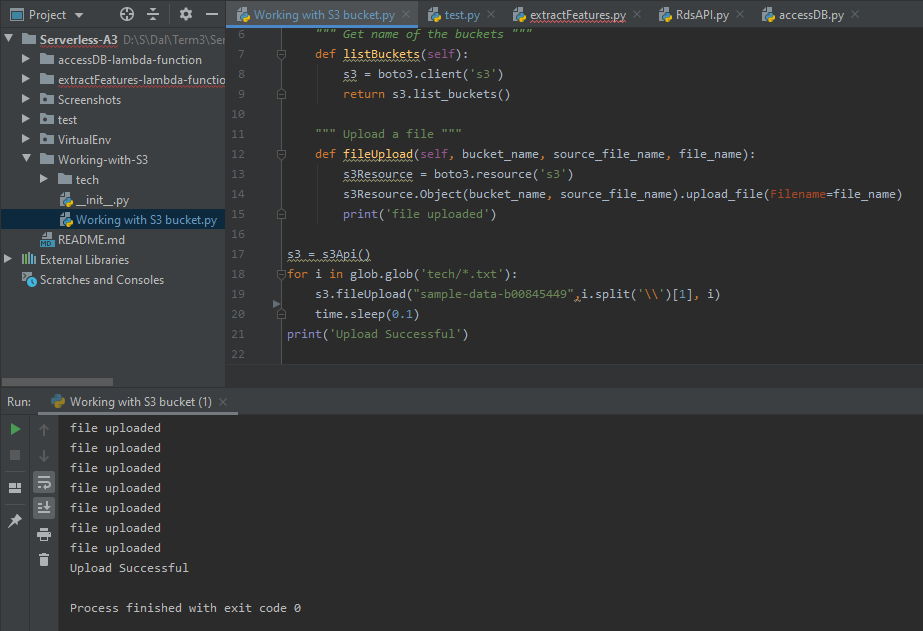
1. **Part A. Build an event-driven serverless application using AWS Lambda.**
2. *a. Create your 1st S3 bucket SampleDataB00xxxxxx and upload the files given in the Tech folder one at a time with a delay of 100 milliseconds. You need to write a script or use the SDK to upload the files one at a time to the S3 bucket.*
3. 

Figure a S3 bucket script

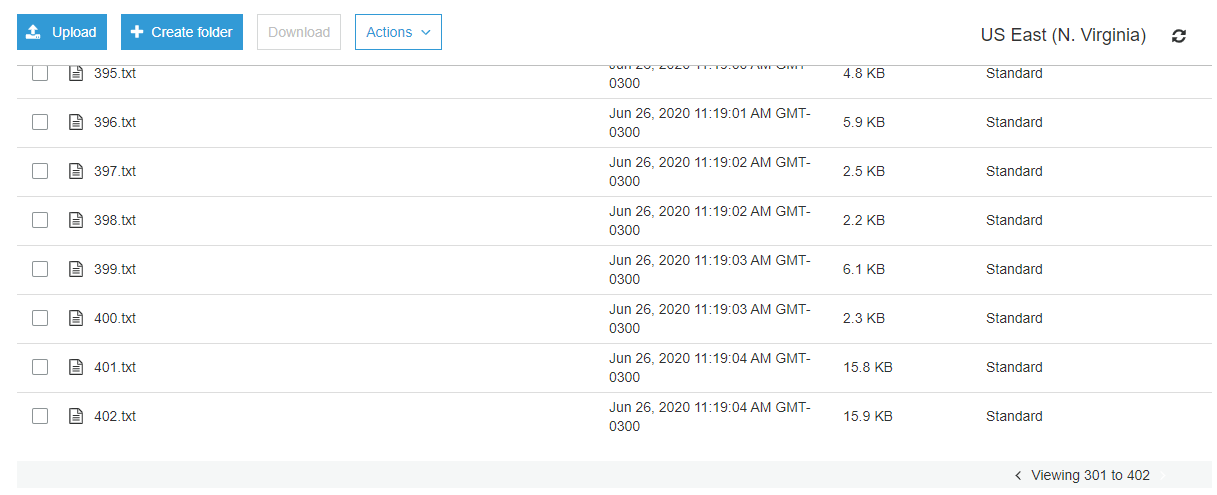
1. figure a shows the script created to upload all the tech folder files to the S3 bucket named ‘sample-data-b00845449’. All the files are uploaded with a delay of 0.1 seconds as specified. I have used ‘glob’ module to get list of all files.
2. 

Figure b S3 bucket files

1. After all the files are uploaded, the files can be seen in the bucket as shown in figure b.
2. *b. If a file is available on the 1st bucket, then it triggers extractFeatures Lambda function, which is the 1st lambda function.*

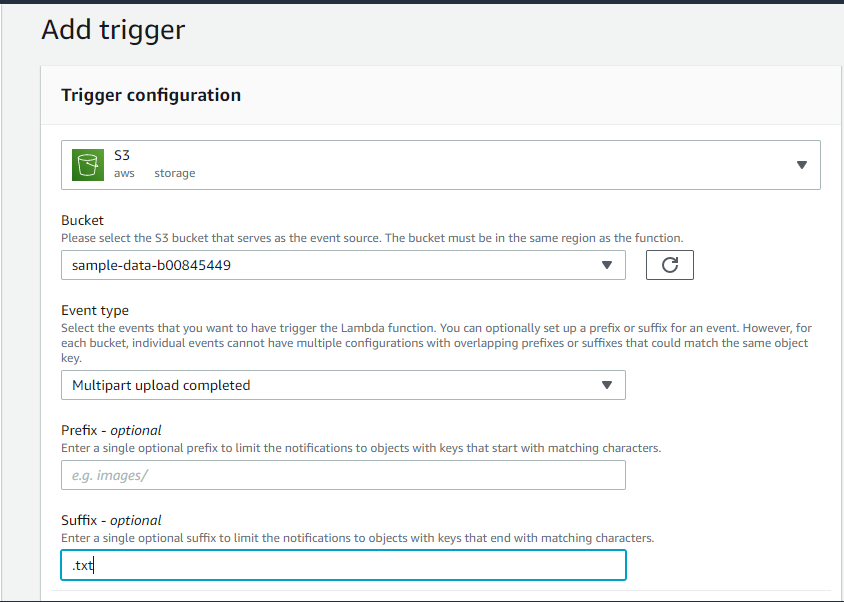


Figure c Add trigger

After creation of a lambda function ‘ExtractFeatures’, A trigger is added for event ‘Multipart upload’ for the files with suffix ‘.txt’ in bucket ‘sample-data-b00845449’ where we added files earlier. The event ‘put’ can also be used.

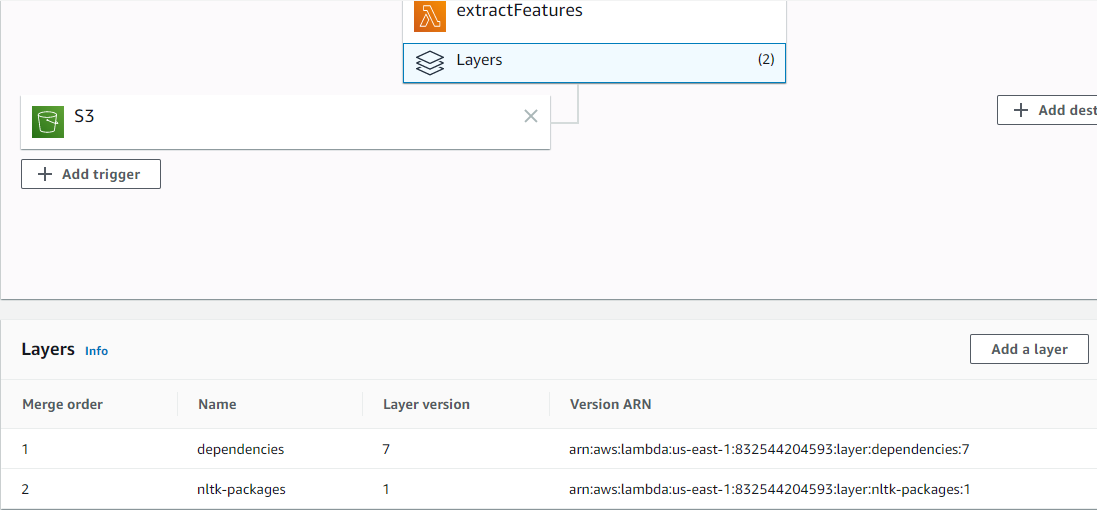


Figure d Add layer

As nltk module is used in the function, two layers are added; one contains the nltk module and another layer contains the nltk packages which are required to download for the program (stopwords, punkt corpora, etc.). Hence 2 layers are added to the ExtractFeatures function.



Figure e ExtractFunction Lambda handler script

Figure e shows the lambda function for ExtractFunction.

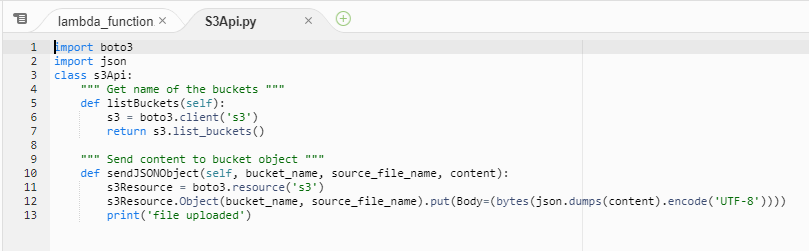


Figure f S3Api function

Figure f shows the S3Api function used in ExtractFunction Lambda Handler to work with S3.

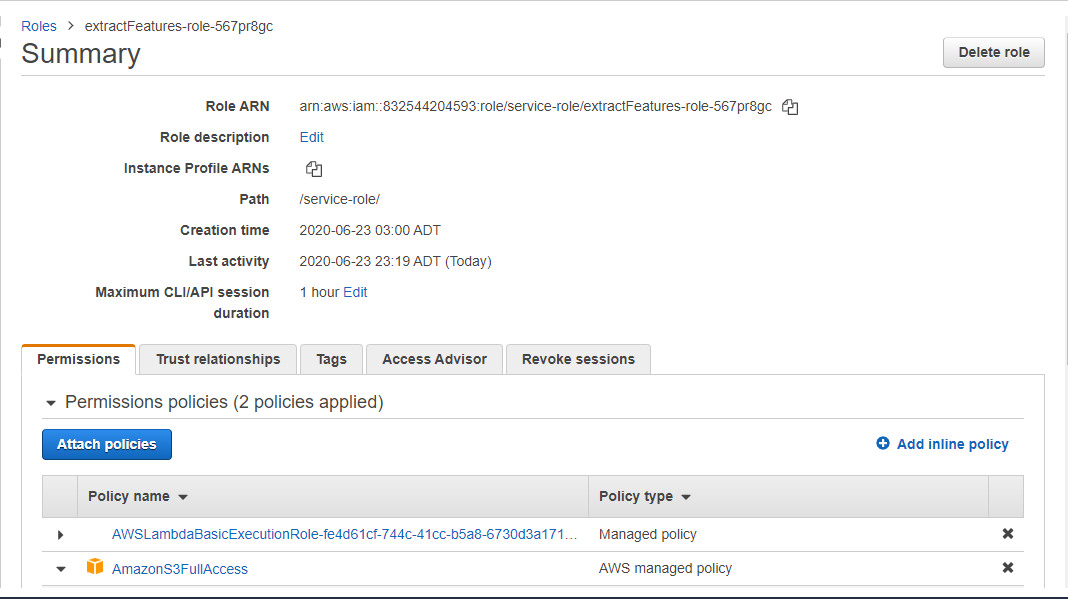


Figure g Policy for ExtractFeatures

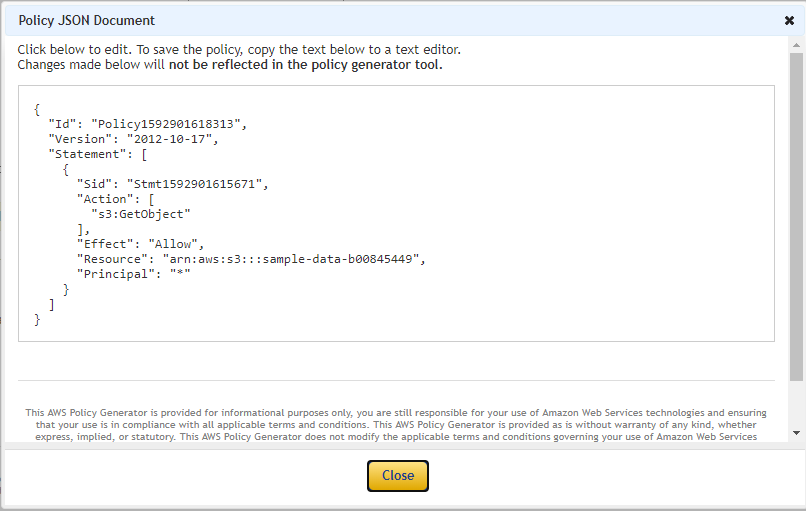
1. Figure g shows the role for the extractFeatures lambda function. One is the default lambda policy which gives access to cloudwatch logs and another is to give full access to S3.
2. 

Figure h getObject policy

1. A policy for specifically getObject() can also be created and put here which will give specific access rights. The policy is provided in figure h.
2. *c. This lambda function extracts the Named entities from the file and creates a JSON array of named entities\* for that file.*

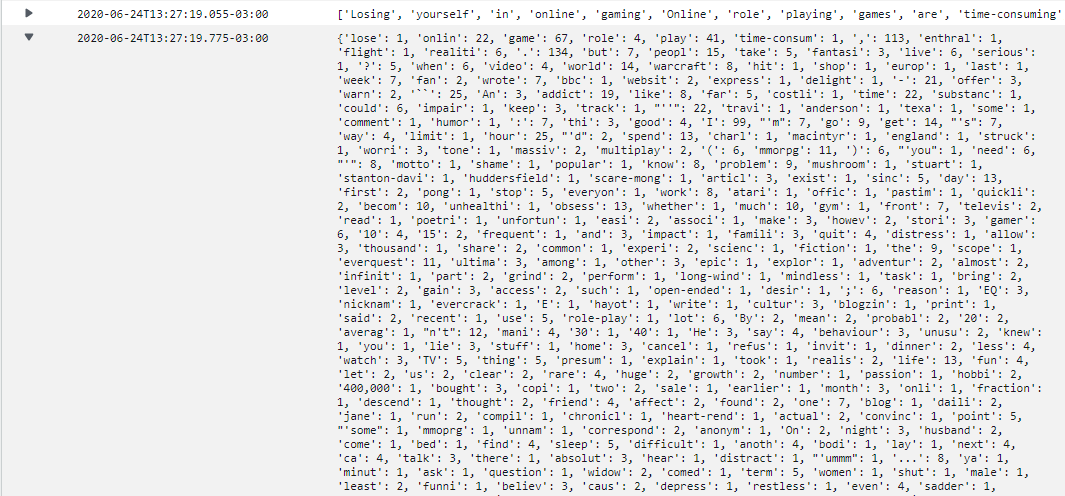


Figure i NamedEntities Extracted in Cloudwatch Logs

Cloudwatch logs shows the NamedEntities Extracted using ExtractFeatures lambda function from sample-data-b00845449 bucket keys. The ExtractFeatures lambda function will save this extracted entities in another bucket ‘tagsb00845449’.

1. *d. E.g. 001.txt contains Asia, Soviet, Serbia etc., then the JSON array created by the function should be “001ne”: {“Asia”:1, “Soviet”:1…..etc.}.*
2. 

Figure j 001ne.txt

Figure J shows the 001ne.txt generated in the bucket ‘tagsb00845449’. A dictionary of named entities with their frequencies in 001ne key.

1. *e. This file will be saved as 001ne.txt in a new bucket - TagsB00xxxxxx.*

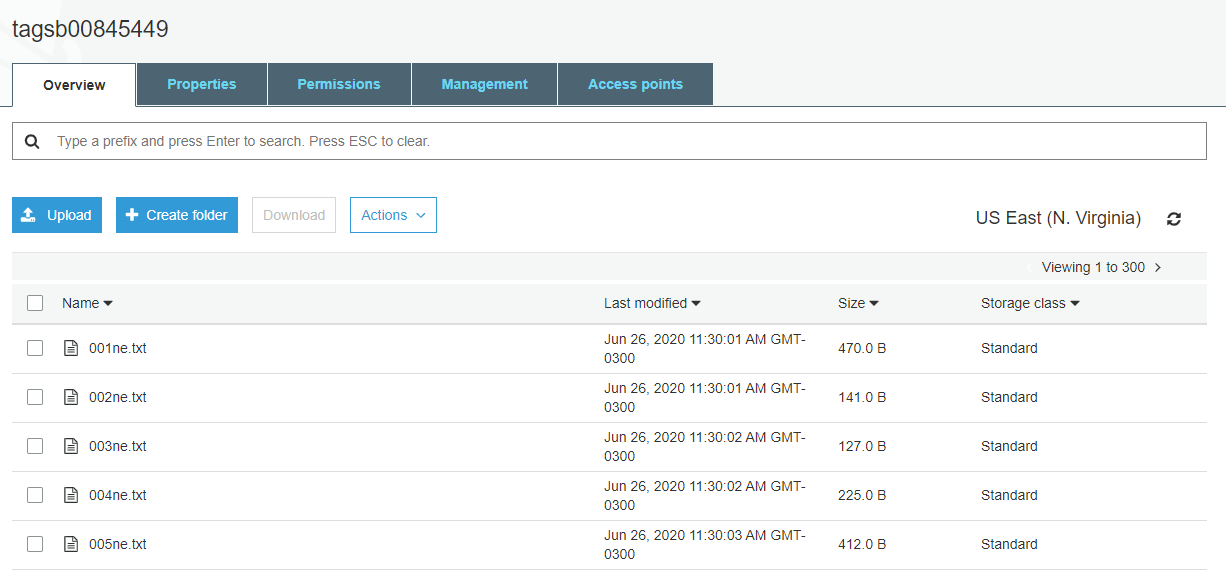


Figure k tagsb00845449 bucket files

Figure K shows all the files generated using extractFeature lambda handler where each file containing the named entity of each key in bucket sample-data-b00845449.

1. *f. Once the file is available on this 2nd bucket, then accessDB Lambda function will automatically be triggered.*

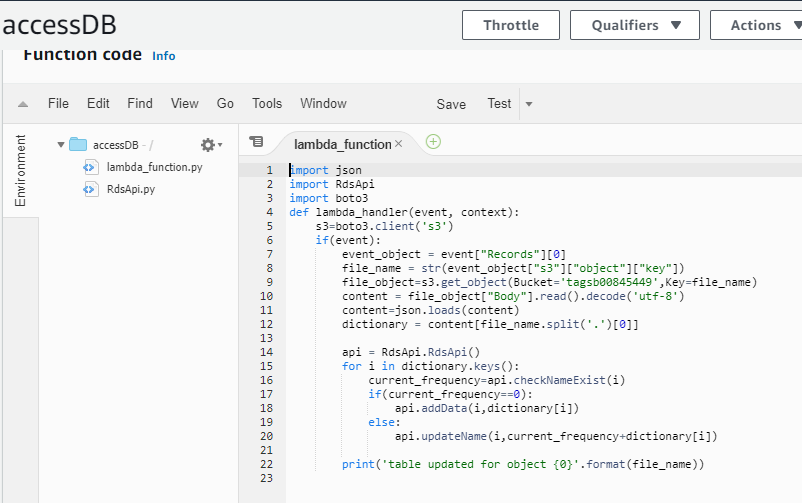


Figure l accessDB lambda handler

Figure l shows accessDB lambda function script which fetches every object from tagsb00845449 bucket and saves it to AWS RDS SQL database.

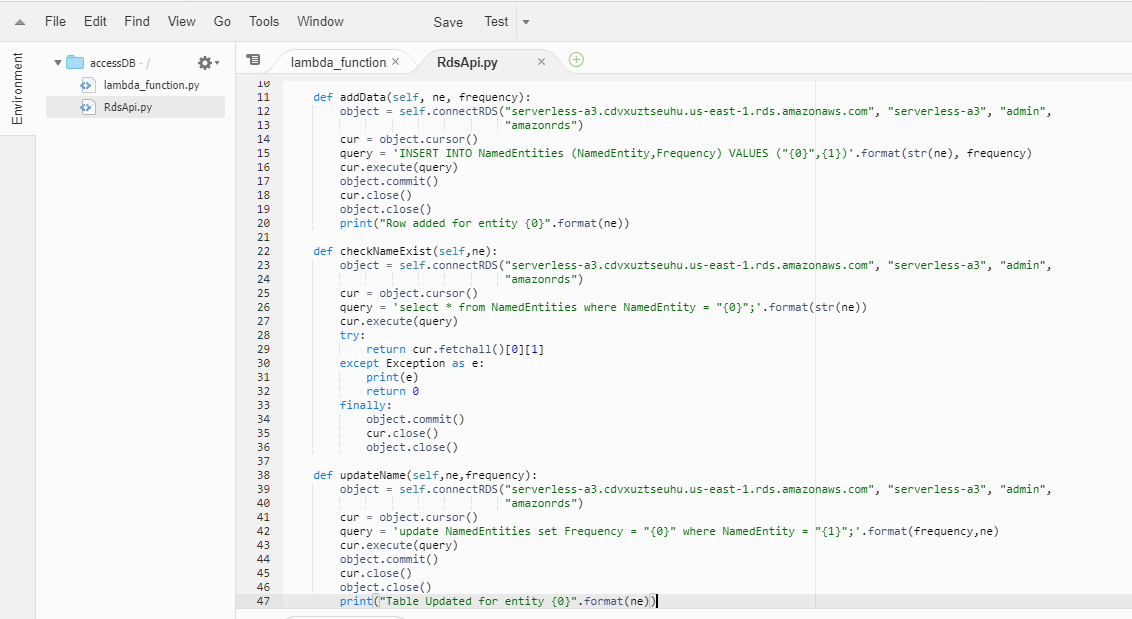


Figure m RDSApi script

1. Figure m shows the RDSApi script used for the AccessDB lambda handler to work with RDS.
2. *g. accessDB is your 2nd Lambda function. This Lambda function reads each named entity JSON file and updates the MySQL database table (two fields - key, value).*

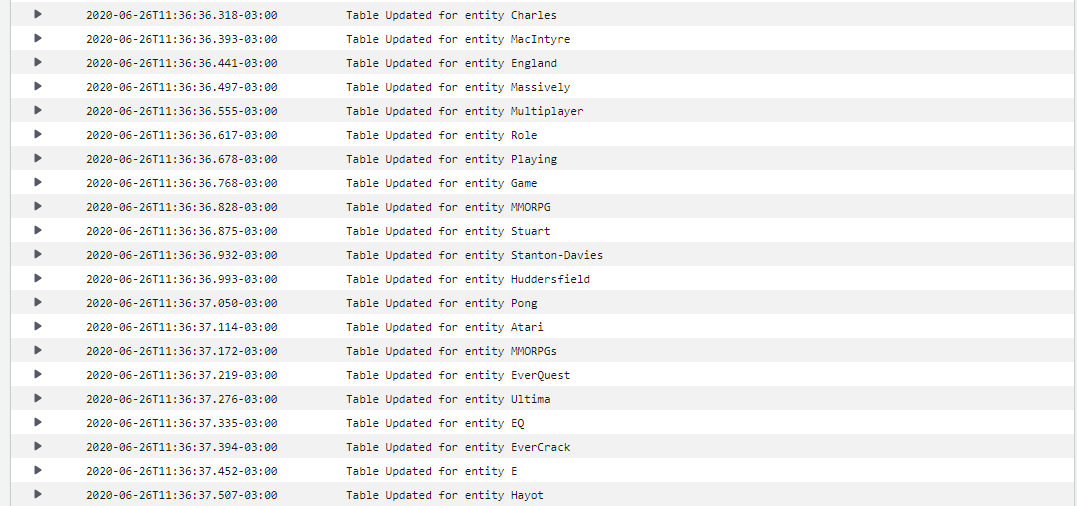


Figure n cloudwatch logs for accessDB

Figure n shows logs which conveys table updated for every entity present in the files of tagsb00845449 bucket. accessDB is fetching every entity from files of tagsb00845449 and adding it to SQL database with their frequency. If the entity is already present, then the frequency is updated.

1. h. E.g. 001ne.txt contains “001ne”: {“Asia”:1, “Soviet”:1…..etc.}. Then this Lambda function will update a MySQL database table where “Asia” will be a value for field “NamedEntity”, and “1” will be the value for field “Frequency”. The field “NamedEntity” can be considered as the primary key.

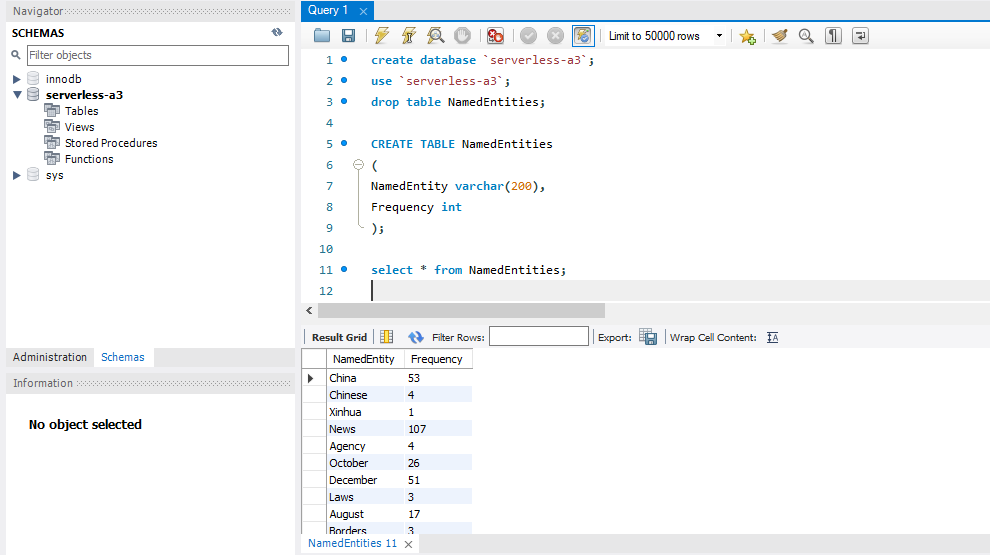


Figure o serverless-a3 database queries and entries

Figure 0 shows the database created from AWS RDS. The database contains a table with 2 columns ‘NamedEntity’ and ‘Frequency’. After the accessDB lambda handler is implemented, all the entities are stored in the database as shown in figure 0.

**References**

[1]. 2020. [online] Available at: <https://www.youtube.com/watch?v=EsqjHDpLpB4> [Accessed 26 June 2020].

[2]. Docs.aws.amazon.com. 2020. *What Is AWS Lambda? - AWS Lambda*. [online] Available at: <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html> [Accessed 26 June 2020].

[3]. 2020. [online] Available at: <https://www.youtube.com/watch?v=vXiZO1c5Sk0> [Accessed 26 June 2020].

[4]. Programcreek.com. 2020. *Nltk.Ne\_Chunk Python Example*. [online] Available at: <https://www.programcreek.com/python/example/91258/nltk.ne\_chunk> [Accessed 26 June 2020].