CSE3086 – NOSQL DATABASES

DIGITAL ASSIGNMENT – 2

Done By: 19MIA1074 – S.Vamsidhar

Topic Chosen: Armoury Management System

Key – Value Datastore Chosen: Amazon Dynamo DB

Programming Language Chosen: Python

IDE Chosen: Jupyter Notebook

About Dynamo DB:

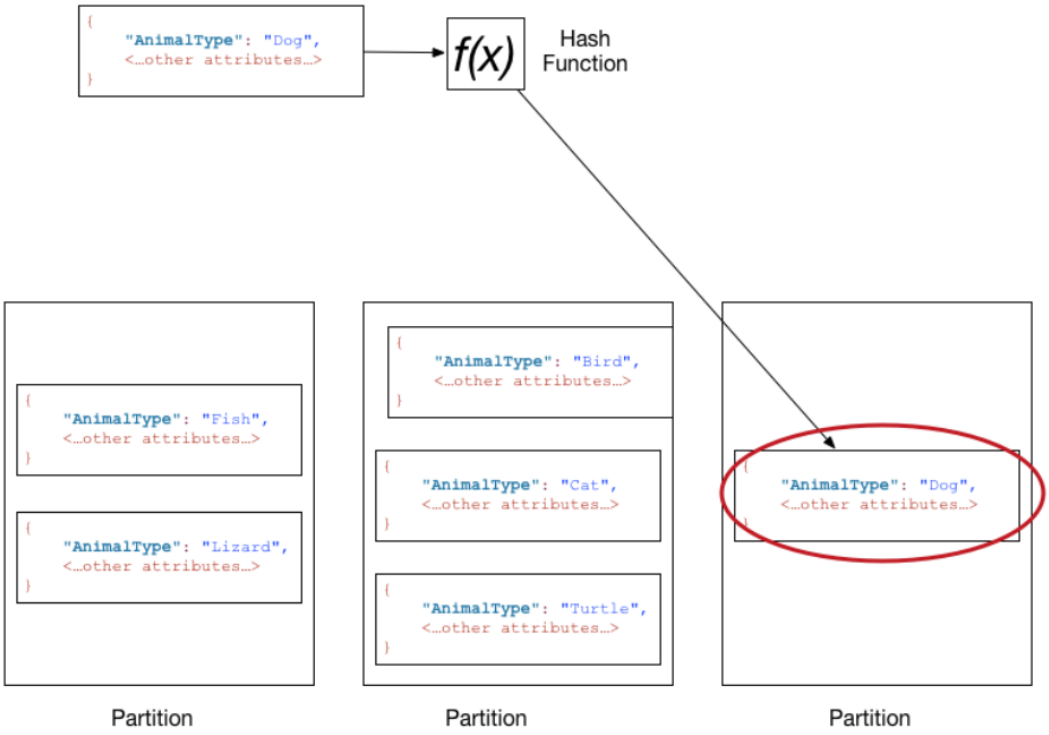
Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale.

DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.

The key system is similar to that of Cassandra. The difference is the change in the name of the clustering key which is sort key in this case.

Partition Key:

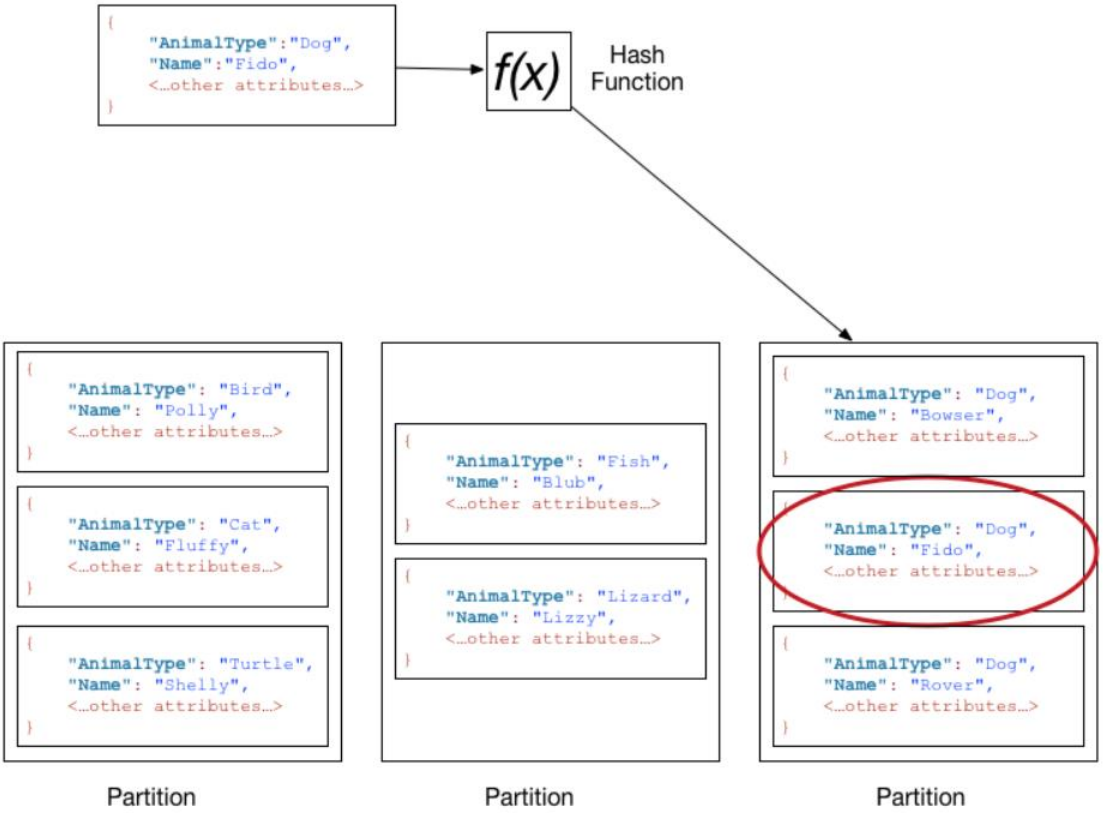
If the table has a simple primary key (partition key only), DynamoDB stores and retrieves each item based on its partition key value.



Sort Key:

If the table has a composite primary key (partition key and sort key), DynamoDB calculates the hash value of the partition key in the same way as described in the Partition key section.

However, it stores all the items with the same partition key value physically close together, ordered by sort key value.



Packages required to link Dynamo DB with Python:

1) Boto3
2) Awscli

After installing both these packages in the local system, we have to type the command “aws configure” in the local command prompt or anaconda prompt.

After typing that, the system asks for security credentials such as Access Key, Security Key, Region and Output Format. We can skip filling the Output Format field.

Once done, we are ready to connect Dynamo DB based on our AWS (Amazon Web Services) account with Python.

References:

I had referred mostly the official AWS documentations to grasp a basic idea about Dynamo DB and websites such as stack overflow to learn the method to link with Python and solve the encountered errors during the process.

Here's a sample reference document which I had referred to learn about the data types:

https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Query.html

Code:

```
import boto3

from botocore.exceptions import ClientError

from pprint import pprint

from decimal import Decimal

import time


client = boto3.client('dynamodb')


def create():

    table = client.create_table(

        TableName = 'Armoury',

        KeySchema = [

            {
```

```
        'AttributeName': '_id',
        'KeyType': 'HASH'  # Partition key
    },
    {
        'AttributeName': 'gun_type',
        'KeyType': 'RANGE'  # Sort key
    }
],
AttributeDefinitions = [
    {
        'AttributeName': '_id',
        'AttributeType': 'N'
    },
    {
        'AttributeName': 'gun_type',
        'AttributeType': 'S'
    },

],
ProvisionedThroughput = {
    'ReadCapacityUnits': 10,
    'WriteCapacityUnits': 10
```

```
        }
    )
    return table

def show_menu():
    print("**********MENU**********")
    print("1. Insert gun details")
    print("2. Find a particular gun details with the help of primary keys")
    print("3. Update gun details")
    print("4. Delete a particular gun details")

    option = input("Enter option: ")
    return option

def insert():

    print("**********ADD OPERATION**********")

    _id = input("Enter the ID of the gun: ")
    name1 = input("Enter the name of the gun: ")
    gun_type = input("Enter the type of the gun: ")
    country_of_origin = input("Enter the country of origin of the gun: ")
```

```
    ammo = input("Enter the details of the ammunition used for the gun: ")
    organizations_using = input("Enter the name of the organization using the gun: ")
    effective_firing_range = input("Enter the effective firing range of the gun: ")


    _id = int(_id)
    name1 = name1.lower()
    gun_type = gun_type.lower()
    country_of_origin = country_of_origin.lower()
    ammo = ammo.lower()
    organizations_using = organizations_using.lower()
    effective_firing_range = int(effective_firing_range)


    response = client.put_item(
      TableName = 'Armoury',
      Item = {
          '_id': {
              'N': "{}".format(_id),
          },
          'name1': {
              'S': "{}".format(name1),
```

```
        },
        'gun_type': {
            'S': "{}".format(gun_type),
        },
        'country_of_origin': {
            "S": "{}".format(country_of_origin),
        },
        'ammo': {
            "S": "{}".format(ammo),
        },
        'organizations_using': {
            "S": "{}".format(organizations_using),
        },
        'effective_firing_range': {
            "N": "{}".format(effective_firing_range), # In terms of meters
        }
    }
)

print('Record Added!!!')
return response
```

```
def get():

    print("**********RETRIEVAL OPERATION**********")

    _id = input("Enter the ID of the gun: ")
    gun_type = input("Enter the type of the gun: ")

    _id = int(_id)
    gun_type = gun_type.lower()

    try:
        response = client.get_item(
            TableName = 'Armoury',
            Key = {
                '_id': {
                    'N': "{}".format(_id),
                },
                'gun_type': {
                    'S': "{}".format(gun_type),
                }
            }
            )
```

```
    except ClientError as e:
        print(e.response['Error']['Message'])
    else:
        pprint(response['Item'])
        return response['Item']


def update():


    # AWS's official documentation says that we cannot update the
primary key attributes.
    # Instead, we can delete the item and create a new item with new
attributes.


    # If the values are meant to remain the same, we just retype the
existing values as it is.
    # Else the values will be treated as NULL if we skip the input stage of a
particular field


    # So, first, we enter the id and type of the gun and then update its
values.


    print("**********UPDATE OPERATION**********")
```

```
    _id = input("Enter the ID of the gun: ")
    name1 = input("Enter the name of the gun: ")
    gun_type = input("Enter the type of the gun: ")
    country_of_origin = input("Enter the country of origin of the gun: ")
    ammo = input("Enter the details of the ammunition used for the gun: ")
    organizations_using = input("Enter the name of the organization using the gun: ")
    effective_firing_range = input("Enter the effective firing range of the gun: ")

    _id = int(_id)
    name1 = name1.lower()
    gun_type = gun_type.lower()
    country_of_origin = country_of_origin.lower()
    ammo = ammo.lower()
    organizations_using = organizations_using.lower()
    effective_firing_range = int(effective_firing_range)

    response = client.update_item(
        TableName = 'Armoury',
        Key = {
            '_id': {
```

```
            'N': "{}".format(_id),
        },
        'gun_type': {
            'S': "{}".format(gun_type),
        }
    },
    ExpressionAttributeNames = {
        '#N': 'name1',
        '#C': 'country_of_origin',
        '#A': 'ammo',
        '#O': 'organizations_using',
        '#E': 'effective_firing_range'
    },
    ExpressionAttributeValues = {
        ':n': {
            'S': "{}".format(name1),
        },
        ':c': {
            'S': "{}".format(country_of_origin),
        },
        ':a': {
            'S': "{}".format(ammo),
```

```
            },
            ':o': {
                'S': "{}".format(organizations_using),
            },
            ':e': {
                'N': "{}".format(effective_firing_range),
            }
        },
        UpdateExpression = 'SET #N = :n, #C = :c, #A = :a, #O = :o, #E = :e',
        ReturnValues = "UPDATED_NEW"
    )

    print('Record Updated!!!')
    return response

def delete():

    print("**********DELETE OPERATION**********")

    _id = input("Enter the ID of the gun: ")
    gun_type = input("Enter the type of the gun: ")
    name1 = input("Enter the name of the gun: ")
```

```
_id = int(_id)
gun_type = gun_type.lower()
name1 = name1.lower()

try:
    response = client.delete_item(
        TableName = 'Armoury',
        Key = {
            '_id': {
                'N': "{}".format(_id),
            },
            'gun_type': {
                'S': "{}".format(gun_type),
            }
        },
        ConditionExpression = "name1 = :n",
        ExpressionAttributeValues = {
            ':n': {
                'S': "{}".format(name1),
            }
        }
```

```
    )

except ClientError as e:
    if e.response['Error']['Code'] == "ConditionalCheckFailedException":
        print(e.response['Error']['Message'])
    else:
        raise
else:
    print('Record Deleted!!!')
    return response def delete():

print("**********DELETE OPERATION**********")

_id = input("Enter the ID of the gun: ")
gun_type = input("Enter the type of the gun: ")
name1 = input("Enter the name of the gun: ")

_id = int(_id)
gun_type = gun_type.lower()
name1 = name1.lower()

try:
```

```
    response = client.delete_item(
        TableName = 'Armoury',
        Key = {
            '_id': {
                'N': "{}".format(_id),
            },
            'gun_type': {
                'S': "{}".format(gun_type),
            }
        },
        ConditionExpression = "name1 = :n",
        ExpressionAttributeValues = {
            ':n': {
                'S': "{}".format(name1),
            }
        }
    )

except ClientError as e:
    if e.response['Error']['Code'] == "ConditionalCheckFailedException":
        print(e.response['Error']['Message'])
    else:
```

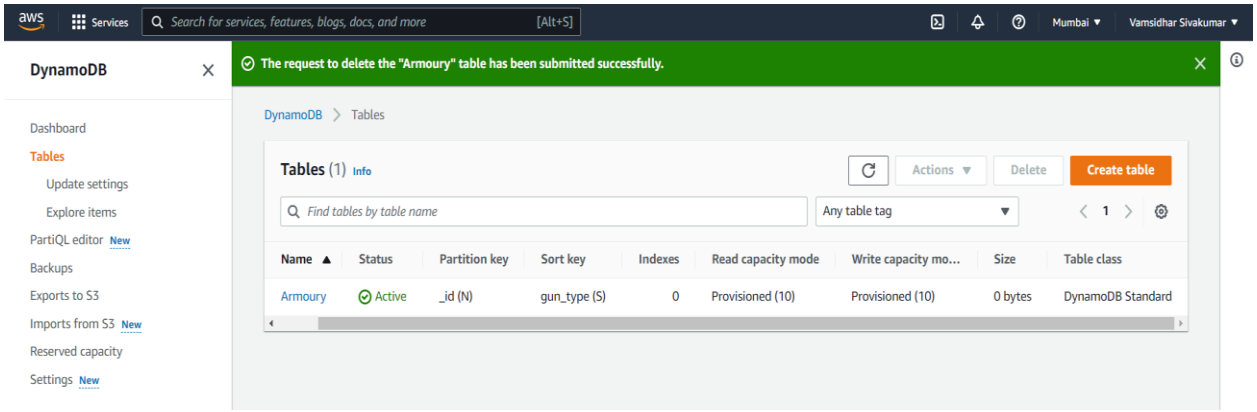
```
            raise
    else:
        print('Record Deleted!!!')
        return response

movie_table = create()
print("Create DynamoDB succeeded.....")
print("Table status:{}".format(movie_table))

time.sleep(5)

def main_loop():
    while True:
        option = show_menu()
        if option == "1":
            insert()
        elif option == "2":
            get()
        elif option == "3":
            update()
        elif option == "4":
            delete()
```

```
        elif option == "5":
            client.close()
            break
        else:
            print("Invalid option")
        print("")


main_loop()
```

Output Screenshots:

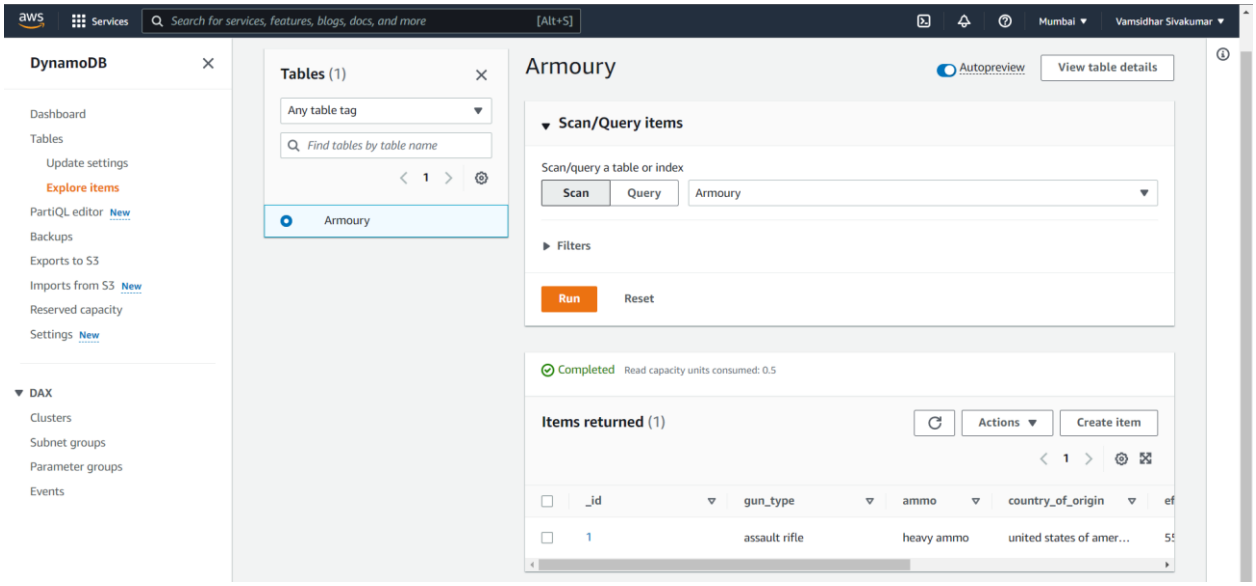
I had first created the table and then cross checked with AWS whether the creation process was completed or not.

```
In [9]: movie_table = create()
        print("Create DynamoDB succeeded...........")
        print("Table status:{}".format(movie_table))

        time.sleep(5)

Create DynamoDB succeeded...........
Table status:{'TableDescription': {'AttributeDefinitions': [{'AttributeName': '_id', 'AttributeType': 'N'}, {'AttributeName':
'gun_type', 'AttributeType': 'S'}], 'TableName': 'Armoury', 'KeySchema': [{'AttributeName': '_id', 'KeyType': 'HASH'}, {'Attrib
uteName': 'gun_type', 'KeyType': 'RANGE'}], 'TableStatus': 'CREATING', 'CreationDateTime': datetime.datetime(2022, 10, 26, 20,
7, 32, 736000, tzinfo=tzlocal()), 'ProvisionedThroughput': {'NumberOfDecreasesToday': 0, 'ReadCapacityUnits': 10, 'WriteCapacit
yUnits': 10}, 'TableSizeBytes': 0, 'ItemCount': 0, 'TableArn': 'arn:aws:dynamodb:ap-south-1:075388929226:table/Armoury', 'Table
Id': '1772dd3c-d995-4d3a-bf51-340c386572fb'}, 'ResponseMetadata': {'RequestId': '001H5D1873HQGGPT5DNC6RRFU3VV4KQNSO5AEMVJF66Q9A
SUAAJG', 'HTTPStatusCode': 200, 'HTTPHeaders': {'server': 'Server', 'date': 'Wed, 26 Oct 2022 14:37:32 GMT', 'content-type': 'a
pplication/x-amz-json-1.0', 'content-length': '575', 'connection': 'keep-alive', 'x-amzn-requestid': '001H5D1873HQGGPT5DNC6RRFU
3VV4KQNSO5AEMVJF66Q9ASUAAJG', 'x-amz-crc32': '1764212370'}, 'RetryAttempts': 0}}
```



I had then added the records of various guns and cross checked with AWS.

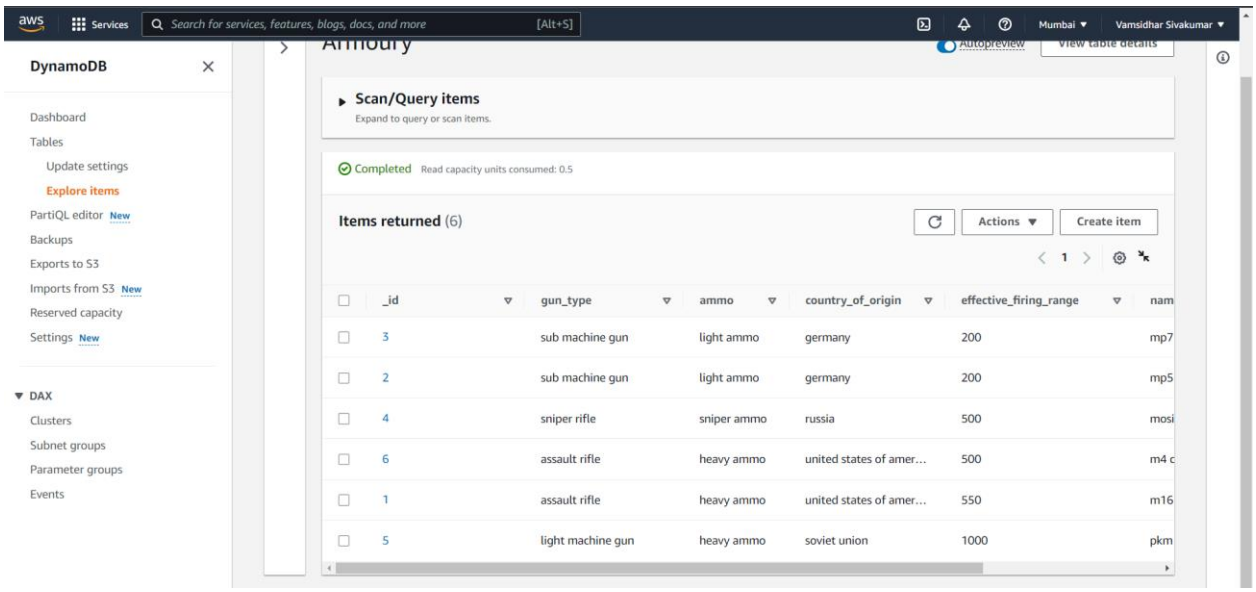
```
In [11]: main_loop()

**********MENU*********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 1
**********ADD OPERATION*********
Enter the ID of the gun: 1
Enter the name of the gun: M16
Enter the type of the gun: Assault Rifle
Enter the country of origin of the gun: United States of America
Enter the details of the ammunition used for the gun: Heavy Ammo
Enter the name of the organization using the gun: NATO
Enter the effective firing range of the gun: 550
Record Added!!!
```



In [14]:
```
main_loop()
```

```
**********MENU**********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 1
**********ADD OPERATION**********
Enter the ID of the gun: 2
Enter the name of the gun: MP5
Enter the type of the gun: Sub Machine Gun
Enter the country of origin of the gun: Germany
Enter the details of the ammunition used for the gun: Light Ammo
Enter the name of the organization using the gun: NATO
Enter the effective firing range of the gun: 200
Record Added!!!

**********MENU**********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 1
**********ADD OPERATION**********
Enter the ID of the gun: 3
Enter the name of the gun: MP7
Enter the type of the gun: Sub Machine Gun
Enter the country of origin of the gun: Germany
Enter the details of the ammunition used for the gun: Light Ammo
Enter the name of the organization using the gun: NATO
Enter the effective firing range of the gun: 200
Record Added!!!


**********MENU**********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 1
**********ADD OPERATION**********
Enter the ID of the gun: 4
Enter the name of the gun: Mosin-Nagant
Enter the type of the gun: Sniper Rifle
Enter the country of origin of the gun: Russia
Enter the details of the ammunition used for the gun: Sniper Ammo
Enter the name of the organization using the gun: Islamic State
Enter the effective firing range of the gun: 500
Record Added!!!
```

```
**********MENU**********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 1
**********ADD OPERATION**********
Enter the ID of the gun: 5
Enter the name of the gun: PKM
Enter the type of the gun: Light Machine Gun
Enter the country of origin of the gun: Soviet Union
Enter the details of the ammunition used for the gun: Heavy Ammo
Enter the name of the organization using the gun: Islamic State
Enter the effective firing range of the gun: 1000
Record Added!!!
```



I had then tested the retrieval operation.

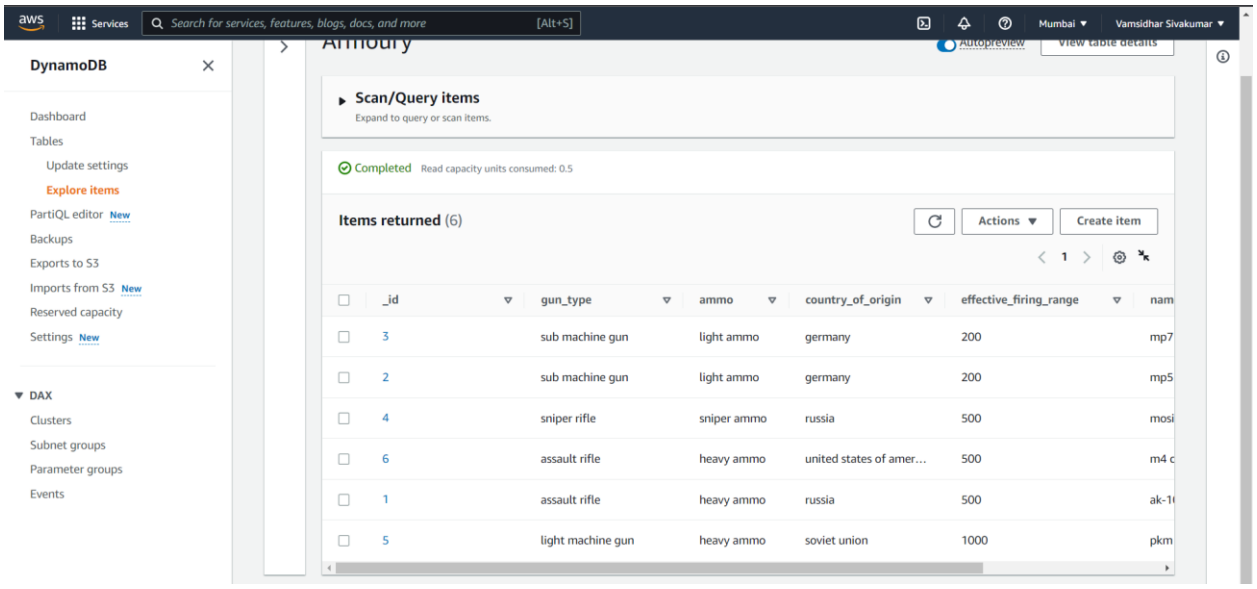
```
In [18]: main_loop()
```

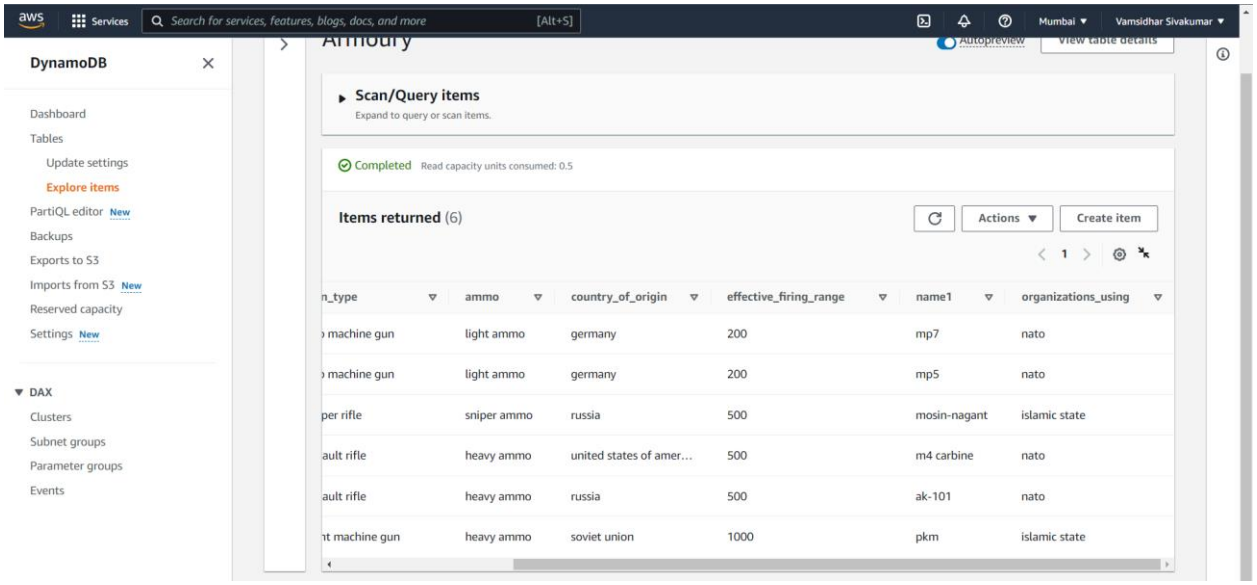
```
**********MENU**********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 2
**********RETRIEVAL OPERATION**********
Enter the ID of the gun: 1
Enter the type of the gun: Assault Rifle
{'_id': {'N': '1'},
 'ammo': {'S': 'heavy ammo'},
 'country_of_origin': {'S': 'united states of america'},
 'effective_firing_range': {'N': '550'},
 'gun_type': {'S': 'assault rifle'},
 'name1': {'S': 'm16'},
 'organizations_using': {'S': 'nato'}}
```

I had then tested the update operation and cross checked with AWS.

```
In [19]: main_loop()
```

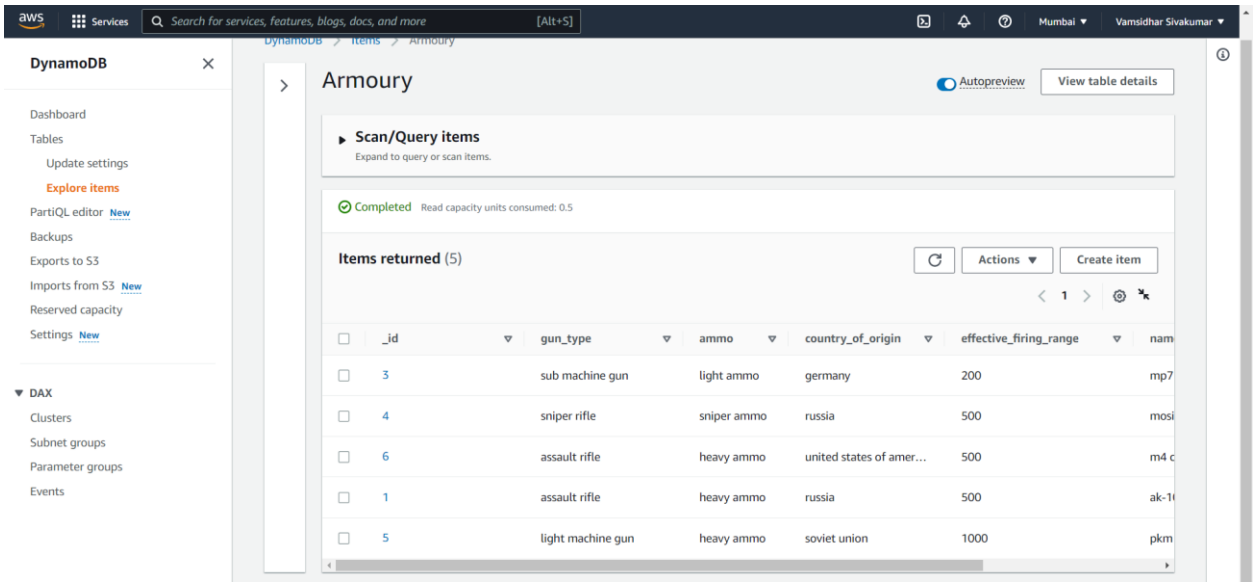
```
**********MENU**********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 3
**********UPDATE OPERATION**********
Enter the ID of the gun: 1
Enter the name of the gun: AK-101
Enter the type of the gun: Assault Rifle
Enter the country of origin of the gun: Russia
Enter the details of the ammunition used for the gun: Heavy Ammo
Enter the name of the organization using the gun: NATO
Enter the effective firing range of the gun: 500
Record Updated!!!
```





I had then tested the delete operation and cross checked with AWS.

```
In [23]: main_loop()

*********MENU*********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 4
*********DELETE OPERATION*********
Enter the ID of the gun: 2
Enter the type of the gun: Sub Machine Gun
Enter the name of the gun: MP5
Record Deleted!!!

*********MENU*********
1. Insert gun details
2. Find a particular gun details with the help of primary keys
3. Update gun details
4. Delete a particular gun details
Enter option: 5
```



Explanation:

In this code, there are 5 phases namely creation of the table, insertion of records into the table, retrieving the records, updating the records and deleting the records.

Now, let's have a look at the phases one by one.

1) *Creation Phase:*

Here, I had created a table named “Armoury” with the fields, “_id” as partition key and “gun_type” as sort key. I had also set few parameters such as “ReadCapacityUnits” and “WriteCapacityUnits” to 10.

2) *Insertion Phase:*

Here, the table is made to contain the following fields:

- _id – The ID of a particular gun with respect to the table
- name1 – The name of a particular gun
- gun_type – The type of a particular gun
- country_of_origin – The country from which a particular gun was first produced
- ammo – The ammunition (bullets) a particular gun uses
- organizations_using – Organizations using a particular gun
- effective_firing_range – The effective firing range of a particular gun in order to make it more effective in neutralizing a target.

For simplicity purposes, all phases of the code convert the input data into lower case and typecast the values of numeric fields into integer values.

All fields are treated as String data types except “_id” and “effective_firing_range”, which are treated as Numeric data types.

3) *Retrieval Phase:*

Here, the primary key (“_id” and “gun_type”) is taken as input and the details of the corresponding gun are displayed based on the given input.

4) *Updation Phase:*

Here, the primary key (“_id” and “gun_type”) as well as other fields are taken as input and the details of the corresponding gun can be updated

with other information. More details are given in the form of comments in the Updation code phase.

5) *Deletion Phase:*

Here, the primary key ("_id" and "gun_type") and the field "name1" are taken as input and the details of the corresponding gun can be deleted.