Lab: Dynamo DB

1. Overview

Amazon DynamoDB (https://aws.amazon.com/dynamodb) is a NoSQL key-value and document database .

Amazon DynamoDB
Fast and flexible NoSQL database service for any scale
Get started with Amazon DynamoDB on the AWS Free Tier

Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multiregion, multimaster, durable database with built-in security, backup and restore, and in-memory caching for internet-scale applications. DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second.

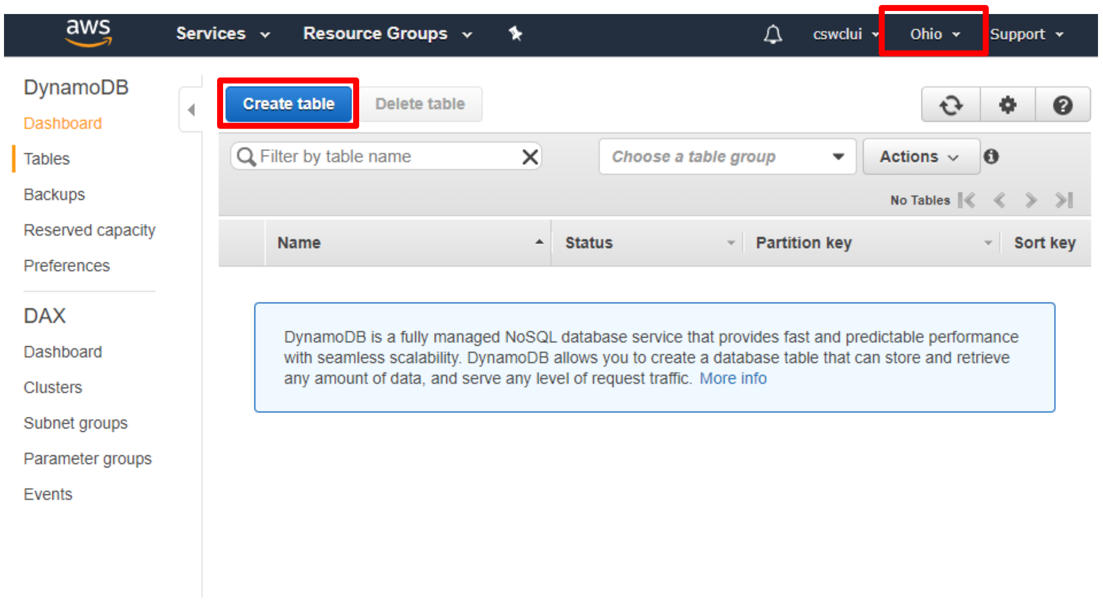
Many of the world's fastest growing businesses such as Lyft, Airbnb, and Redfin as well as enterprises such as Samsung, Toyota, and Capital One depend on the scale and performance of DynamoDB to support their mission-critical workloads.

Hundreds of thousands of AWS customers have chosen DynamoDB as their key-value and document database for mobile, web, gaming, ad tech, IoT, and other applications that need low-latency data access at any scale. Create a new table for your application and let DynamoDB handle the rest.

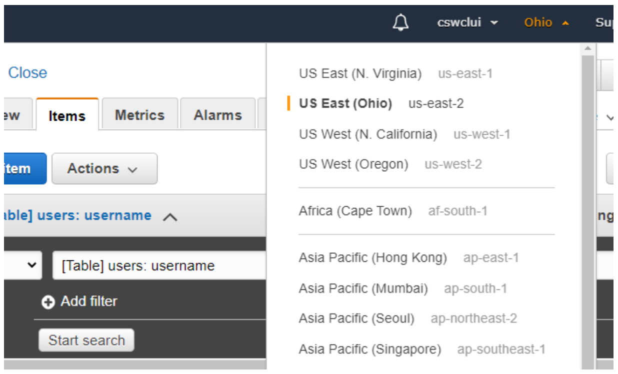
Introduction to Amazon DynamoDB (1:01)

Complete the following steps to create a table in DynamoDB.
First, navigate to your DynamoDB console at console.aws.amazon.com/dynamodb. Select the region **US East (Ohio)** with code **us-east-2** .

Remark: You cannot use ***us-east-2*** *region if you are using the AWS Starter Account. Use* ***us-east-1*** *instead.*

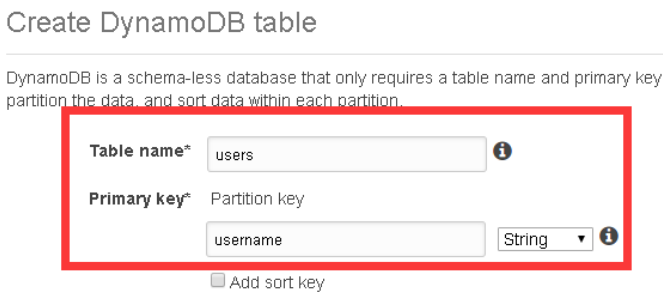


DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB allows you to create a database table that can store and retrieve any amount of data, and serve any level of request traffic. More info


cswclui
Ohio
Close
Items
Metrics
Alarms
Actions
[Table] users: username
Add filter
Start search
US East (N. Virginia) us-east-1
US East (Ohio) us-east-2
US West (N. California) us-west-1
US West (Oregon) us-west-2
Africa (Cape Town) af-south-1
Asia Pacific (Hong Kong) ap-east-1
Asia Pacific (Mumbai) ap-south-1
Asia Pacific (Seoul) ap-northeast-2
Asia Pacific (Singapore) ap-southeast-1

Click “Create table” and create a table "users".

Specify **username** as the partition key (select **String** as data type).



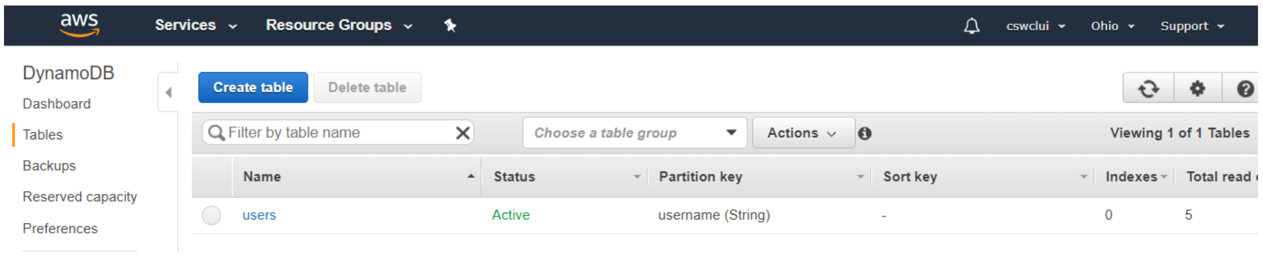
Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key partition the data, and sort data within each partition.

Table name*	users
Primary key*	Partition key
	username — String
	Add sort key

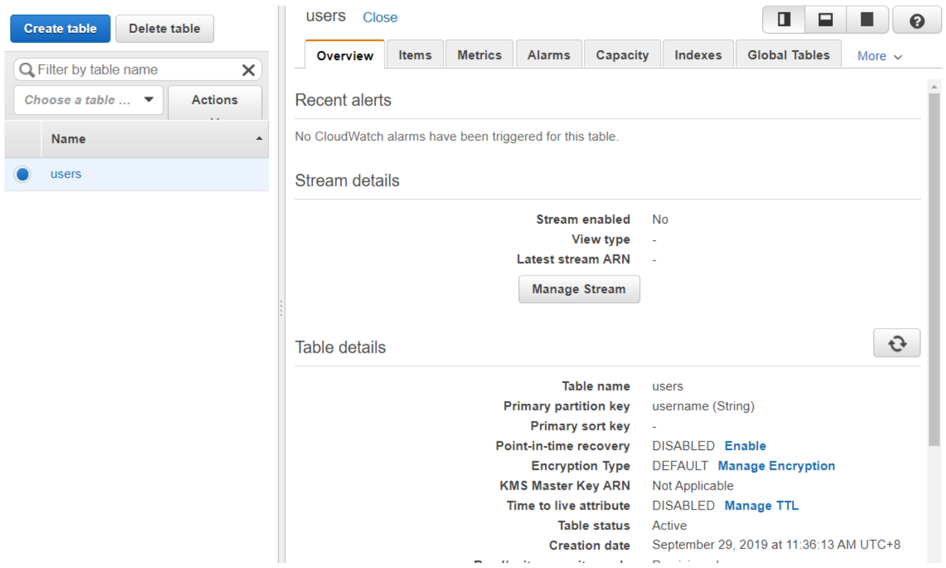
Click on blue “Create” button on the bottom.

This will kick off the table creation process and will take about a minute to complete



Name	Status	Partition key	Sort key	Indexes	Total read
users	Active	username (String)	-	0	5

Click the **users** database and check the **Overview** page.



Recent alerts

No CloudWatch alarms have been triggered for this table.

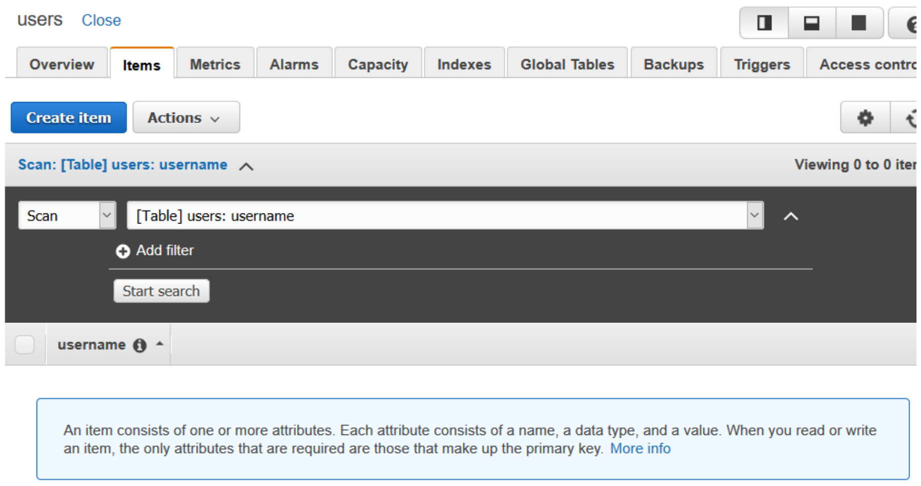
Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

Table details

Table name	users
Primary partition key	username (String)
Primary sort key	-
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT Manage Encryption
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	September 29, 2019 at 11:36:13 AM UTC+8

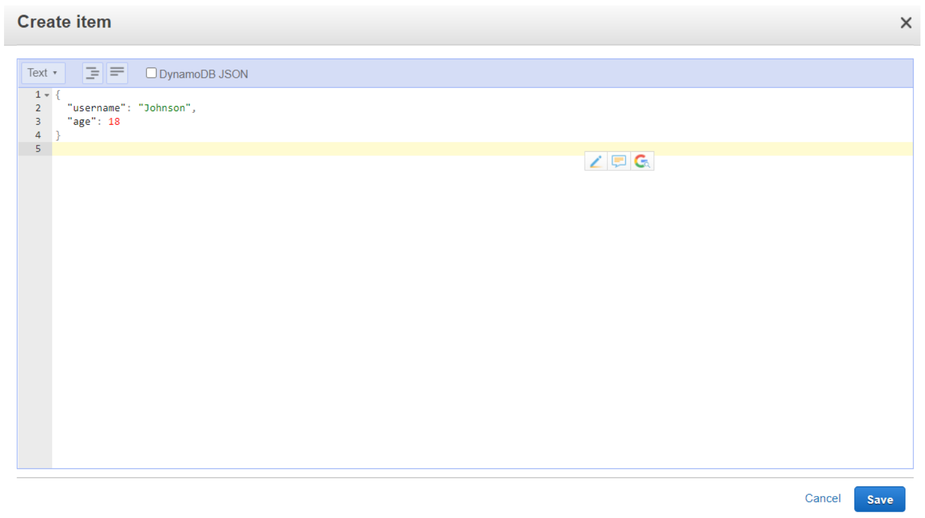
Also, click the **Items** tab to check that the table is empty.



An item consists of one or more attributes. Each attribute consists of a name, a data type, and a value. When you read or write an item, the only attributes that are required are those that make up the primary key. More info

Click “Create Item” and select “Text” to add the following item to the table.

```
{
  "age": 18,
  "username": "Johnson"
}
```

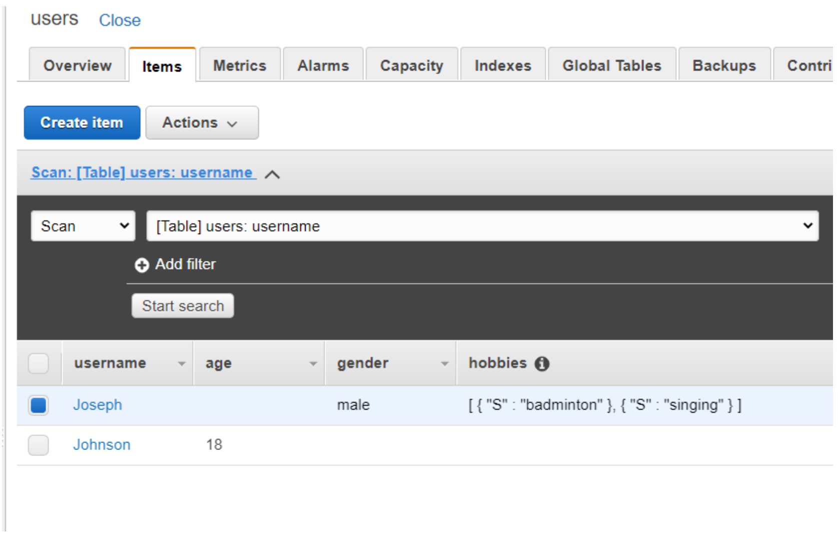


```
{
  "username": "Johnson",
  "age": 18
}
```

Similarly, add the second item

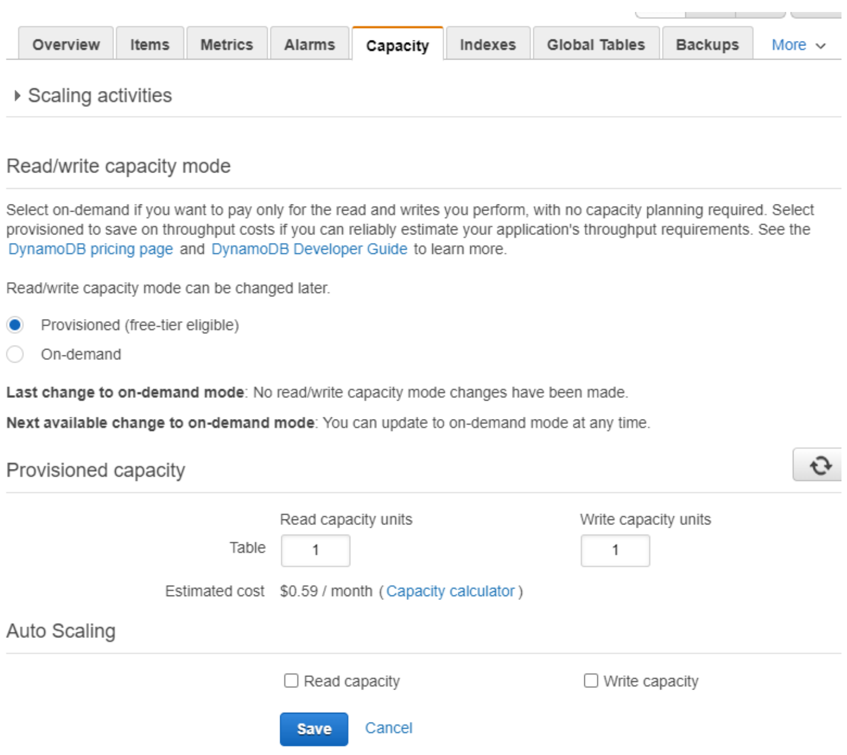
```
{
  "username": "Joseph",
  "gender": "male",
  "hobbies": ["badminton", "singing"]
}
```

Result:



	username	age	gender	hobbies
☑	Joseph		male	[{ "S" : "badminton" }, { "S" : "singing" }]
☐	Johnson	18		

Under the **Capacity** tab of your table, set **1** for the read and write for the provisioned capacity.



Scaling activities

Read/write capacity mode

Select on-demand if you want to pay only for the read and writes you perform, with no capacity planning required. Select provisioned to save on throughput costs if you can reliably estimate your application's throughput requirements. See the DynamoDB pricing page and DynamoDB Developer Guide to learn more.

Read/write capacity mode can be changed later.

[✓ Provisioned (free-tier eligible)] On-demand

Last change to on-demand mode: No read/write capacity mode changes have been made.

Next available change to on-demand mode: You can update to on-demand mode at any time.

Provisioned capacity

	Read capacity units	Write capacity units
Table	1	1
Estimated cost	$0.59 / month (Capacity calculator)	

Auto Scaling

☐ Read capacity ☐ Write capacity

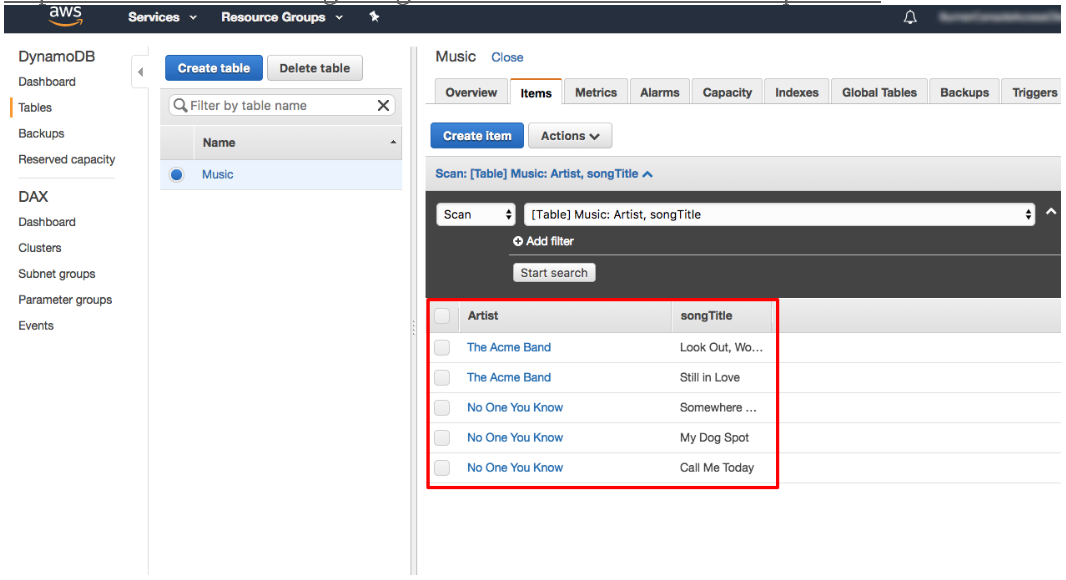
Save Cancel

2. Exercise: Create Music Table

Complete the following tutorial at to create Music table with Artist as the Partition key and sort key with the following data.

- Artist: No One You Know, songTitle: Call Me Today.
- Artist: No One You Know; songTitle: My Dog Spot
- Artist: No One You Know; songTitle: Somewhere Down The Road
- Artist: The Acme Band; songTitle: Still in Love
- Artist: The Acme Band; songTitle: Look Out, World

https://aws.amazon.com/getting-started/hands-on/create-nosql-table/



Scan: [Table] Music: Artist, songTitle

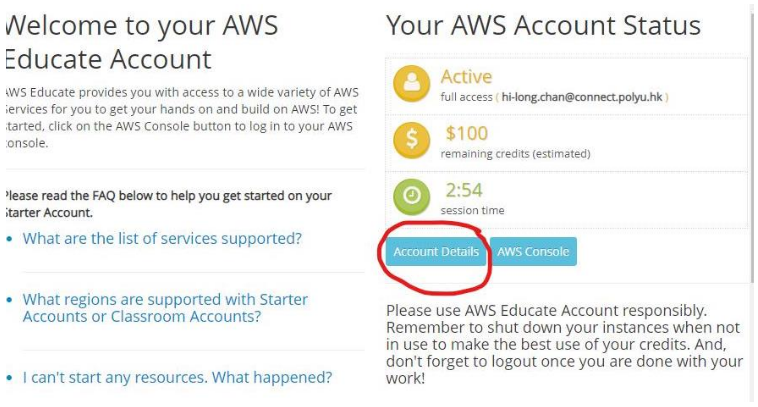
Scan | [Table] Music: Artist, songTitle

Artist	songTitle
The Acme Band	Look Out, Wo...
The Acme Band	Still in Love
No One You Know	Somewhere ...
No One You Know	My Dog Spot
No One You Know	Call Me Today

3. Create IAM user for programmatic access

AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.

Remark: You can only create IAM user if you make use of regular AWS accounts. If you are using the AWS Starter Account, use the access key ID and private key provided in ***Account Details****.*



Welcome to your AWS Educate Account

AWS Educate provides you with access to a wide variety of AWS Services for you to get your hands on and build on AWS! To get started, click on the AWS Console button to log in to your AWS console.

Please read the FAQ below to help you get started on your Starter Account.

- What are the list of services supported?
- What regions are supported with Starter Accounts or Classroom Accounts?
- I can't start any resources. What happened?

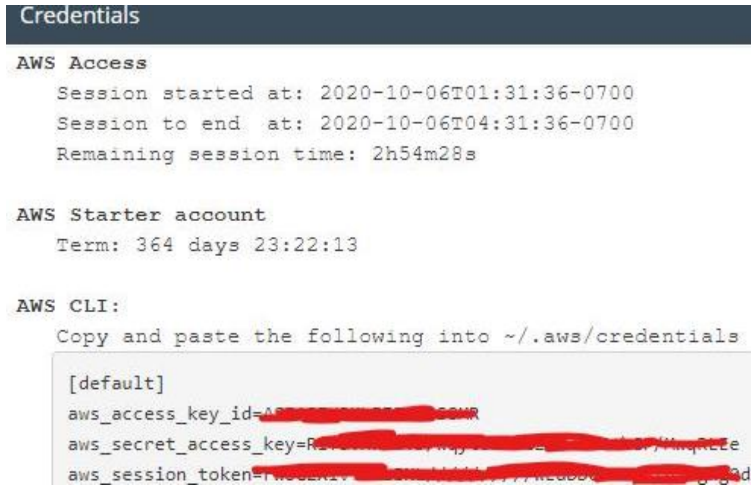
Your AWS Account Status

Active
full access (hi-long.chan@connect.polyu.hk)

$100
remaining credits (estimated)

2:54
session time

Please use AWS Educate Account responsibly. Remember to shut down your instances when not in use to make the best use of your credits. And, don't forget to logout once you are done with your work!

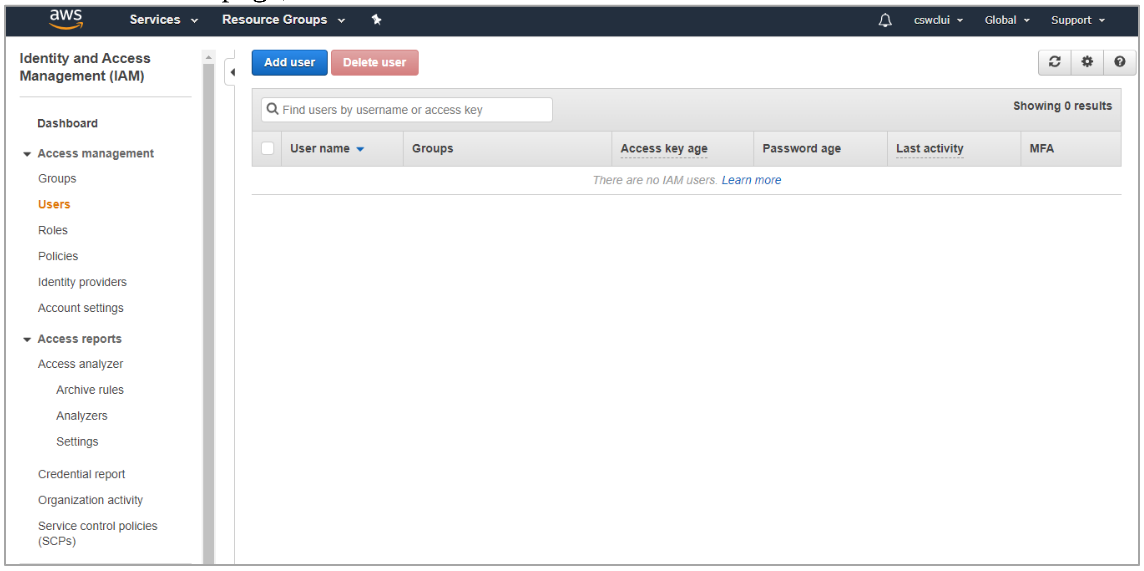


Credentials

```
AWS Access
    Session started at: 2020-10-06T01:31:36-0700
    Session to end  at: 2020-10-06T04:31:36-0700
    Remaining session time: 2h54m28s

AWS Starter account
    Term: 364 days 23:22:13

AWS CLI:
    Copy and paste the following into ~/.aws/credentials
     [default]
     aws_access_key_id=[redacted]
     aws_secret_access_key=[redacted]
     aws_session_token=[redacted]
```

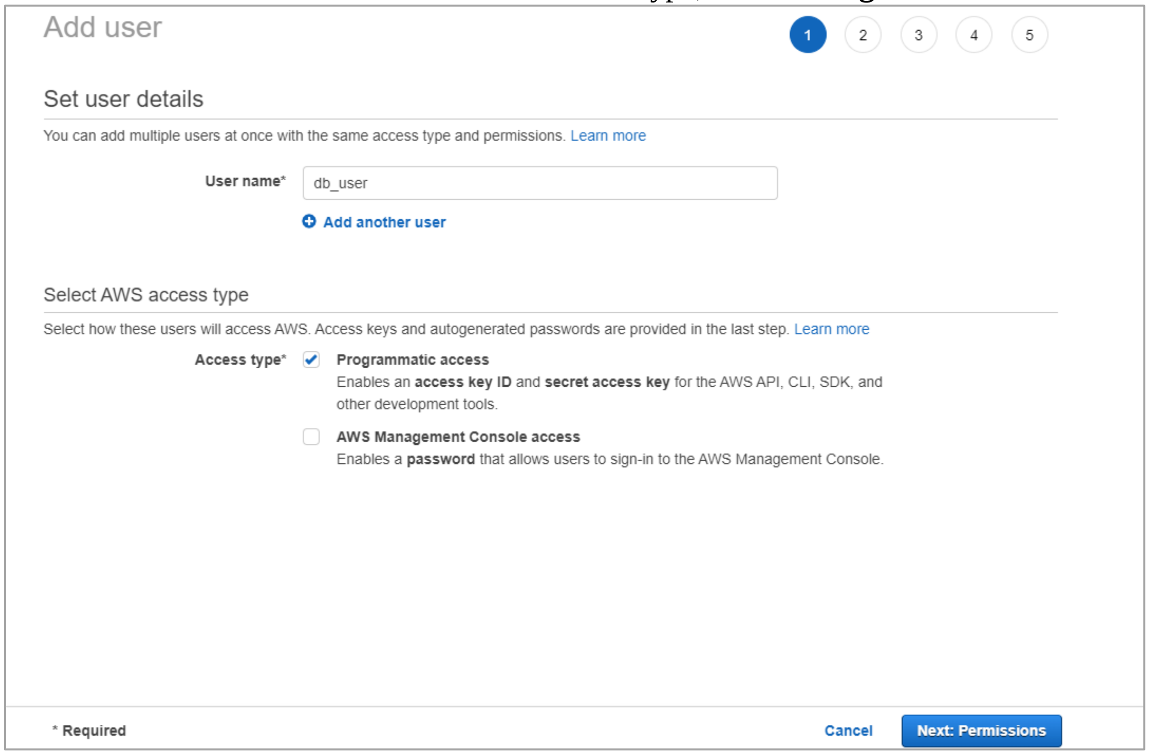
Visit https://console.aws.amazon.com/iam/

Under the **Users** page, click **Add user**.



There are no IAM users. Learn more

Create a user with username **db_user**. For access type, choose **Programmatic access**.



You can add multiple users at once with the same access type and permissions. Learn more

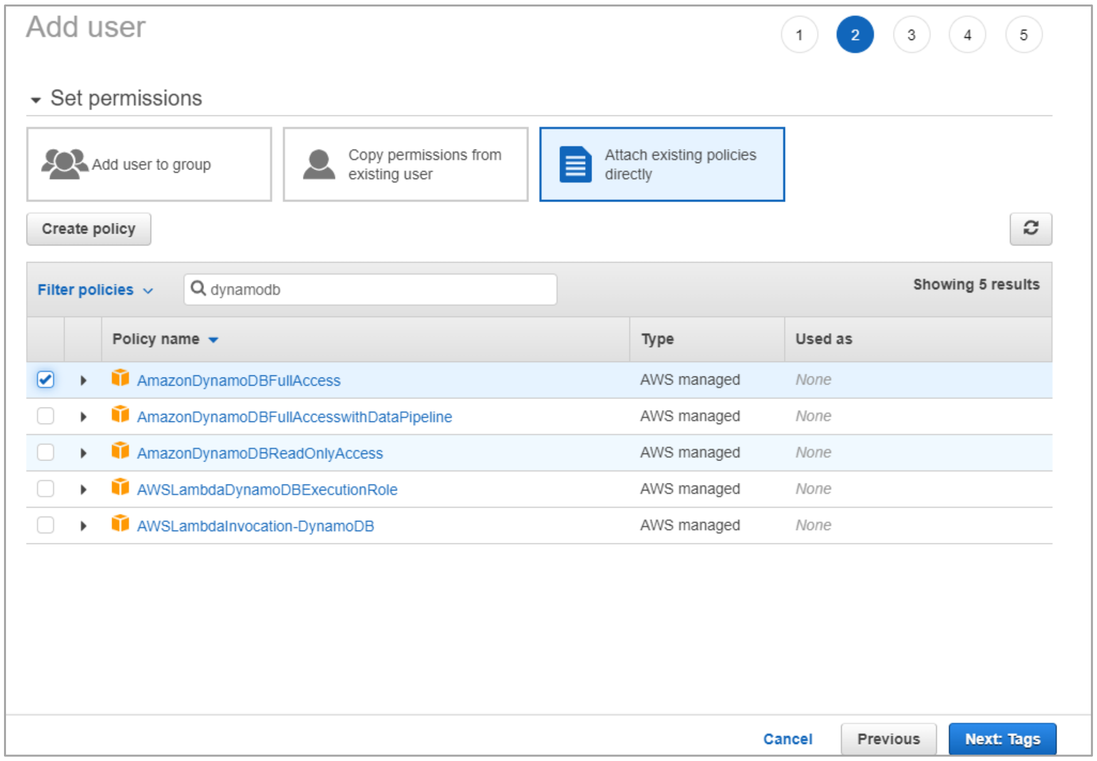
User name* db_user

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* [✓] Programmatic access
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

[ ] AWS Management Console access
Enables a **password** that allows users to sign-in to the AWS Management Console.

Click **Next: Permissions**. Choose Attach existing policies directly.
Select the policy **AmazonDynamoDBFullAccess**.



Add user — 1 2 3 4 5

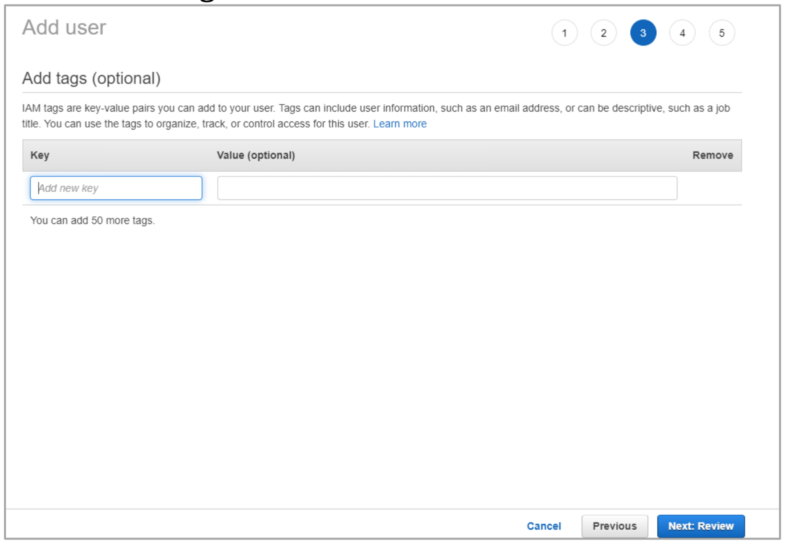
Set permissions

Add user to group
Copy permissions from existing user
Attach existing policies directly
Create policy
Filter policies

dynamodb — Showing 5 results

		Policy name	Type	Used as
✓	▸	AmazonDynamoDBFullAccess	AWS managed	None
	▸	AmazonDynamoDBFullAccesswithDataPipeline	AWS managed	None
	▸	AmazonDynamoDBReadOnlyAccess	AWS managed	None
	▸	AWSLambdaDynamoDBExecutionRole	AWS managed	None
	▸	AWSLambdaInvocation-DynamoDB	AWS managed	None

Click **Next: Tags**.



Add user — 1 2 3 4 5

Add tags (optional)

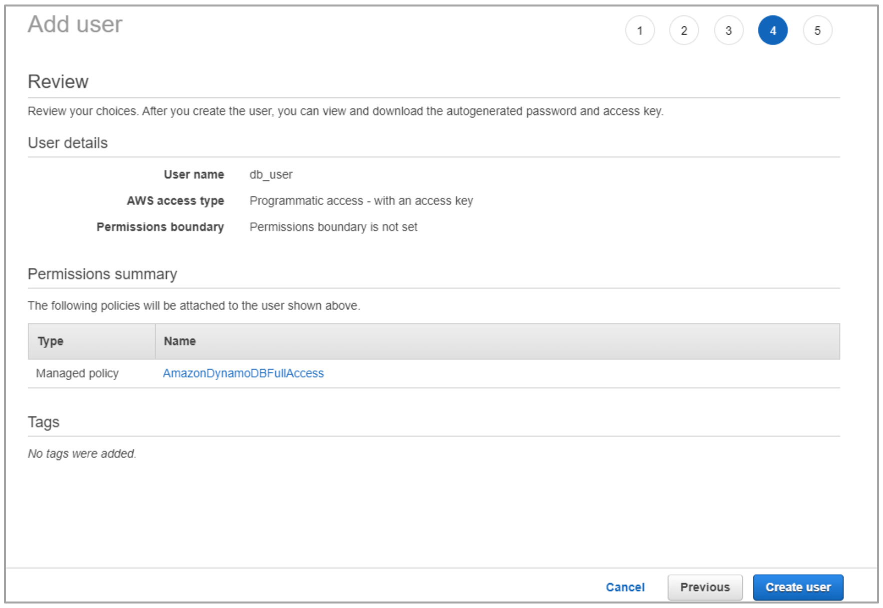
IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. Learn more

Key	Value (optional)	Remove
Add new key		

You can add 50 more tags.

Click **Next:Reviews**.

Review your choice. Click **Create users**.



Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	db_user
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

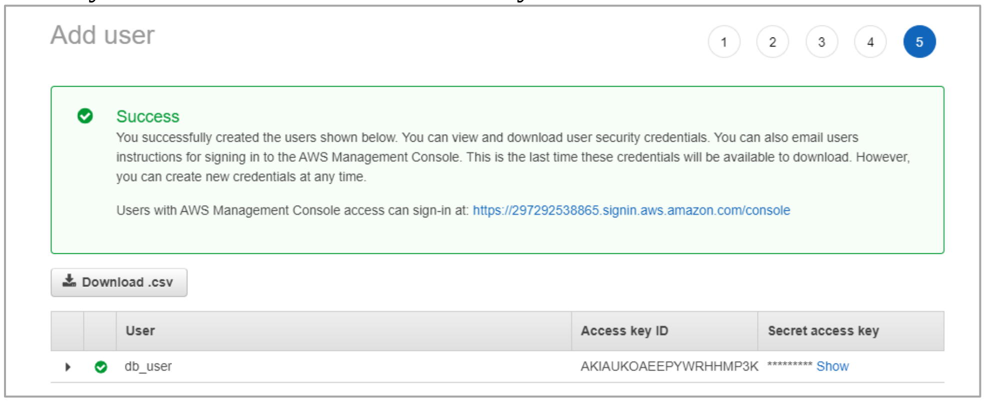
The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonDynamoDBFullAccess

Tags

No tags were added.

Record your access key ID and secret access key. You should store your secret access keys securely. If you do not write down the key or download the key file to your computer before you press "Close" or "Cancel" you will not be able to retrieve the secret key in future. Then you'll have to delete the keys which you created start to create new keys.



Add user

1 2 3 4 5

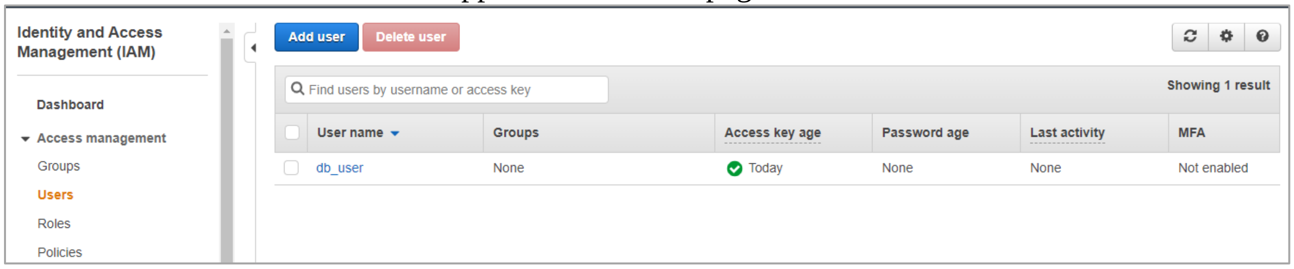
Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://297292538865.signin.aws.amazon.com/console

User	Access key ID	Secret access key
db_user	AKIAUKOAEEPYWRHHMP3K	********* Show

Click **Close**. The created user will appear under **Users** page.



User name	Groups	Access key age	Password age	Last activity	MFA
db_user	None	Today	None	None	Not enabled

4. Accessing DynamoDB with the Command Line Interface

4.1. Configure AWS CLI

Download and install AWS CLI (If you have not done so).

https://aws.amazon.com/cli

Inside the command prompt, enter:
aws configure --profile dbaccess
Provide your access Key ID and secret access key (with access rights to the DynamoDB).
Choose **us-east-2** as the default region and json as the default output.

4.2. AWS CLI Commands

Refer to the following page for command reference for DynamoDB:
https://docs.aws.amazon.com/cli/latest/reference/dynamodb/index.html

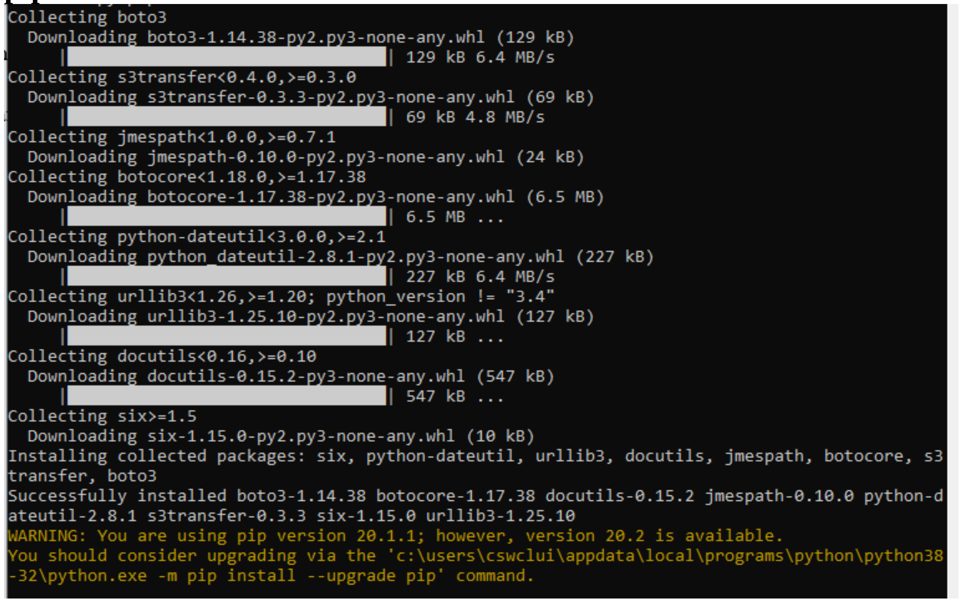
Try the following command to list the tables in DynamoDB.
aws dynamodb list-tables --profile dbaccess

Try the following command to list the tables in DynamoDB.
aws dynamodb describe-table --table-name users --profile dbaccess

5. Using python to access DynamoDB.

Install the boto3 library using pip (if you have not done so).

pip install boto3



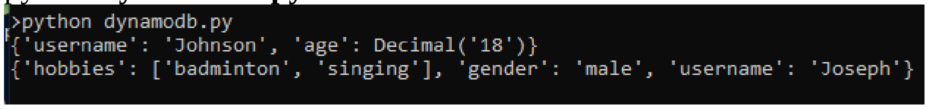
```
Collecting boto3
  Downloading boto3-1.14.38-py2.py3-none-any.whl (129 kB)
     |████████████████████████████████| 129 kB 6.4 MB/s
Collecting s3transfer<0.4.0,>=0.3.0
  Downloading s3transfer-0.3.3-py2.py3-none-any.whl (69 kB)
     |████████████████████████████████| 69 kB 4.8 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting botocore<1.18.0,>=1.17.38
  Downloading botocore-1.17.38-py2.py3-none-any.whl (6.5 MB)
     |████████████████████████████████| 6.5 MB ...
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
     |████████████████████████████████| 227 kB 6.4 MB/s
Collecting urllib3<1.26,>=1.20; python_version != "3.4"
  Downloading urllib3-1.25.10-py2.py3-none-any.whl (127 kB)
     |████████████████████████████████| 127 kB ...
Collecting docutils<0.16,>=0.10
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)
     |████████████████████████████████| 547 kB ...
Collecting six>=1.5
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: six, python-dateutil, urllib3, docutils, jmespath, botocore, s3
transfer, boto3
Successfully installed boto3-1.14.38 botocore-1.17.38 docutils-0.15.2 jmespath-0.10.0 python-d
ateutil-2.8.1 s3transfer-0.3.3 six-1.15.0 urllib3-1.25.10
WARNING: You are using pip version 20.1.1; however, version 20.2 is available.
You should consider upgrading via the 'c:\users\cswclui\appdata\local\programs\python\python38
-32\python.exe -m pip install --upgrade pip' command.
```

Create a python script **dynamodb1.py.**

```python
import boto3
import pprint

session = boto3.Session(
    aws_access_key_id = '[Your AWS Access Key ID] ',
    aws_secret_access_key='[Your AWS Secret Access Key]'
)

dynamodb = session.resource('dynamodb', region_name='us-east-2' )
table = dynamodb.Table('users')

def table_scan():
    result = table.scan()
    for i in result['Items']:
        print(i)

table_scan()
```

Execute your python script.

python **dynamodb1.py**



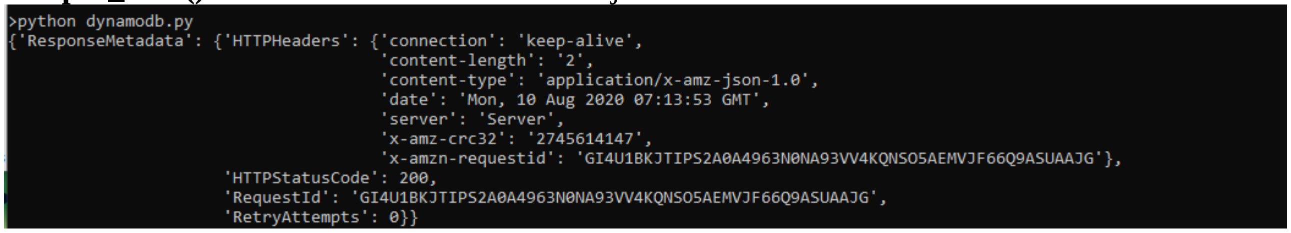
```
>python dynamodb.py
{'username': 'Johnson', 'age': Decimal('18')}
{'hobbies': ['badminton', 'singing'], 'gender': 'male', 'username': 'Joseph'}
```

Example

Modify **dynamodb1.py** to comment out the call to `table_scan()`.
Add the following code to add an item to **users** table.

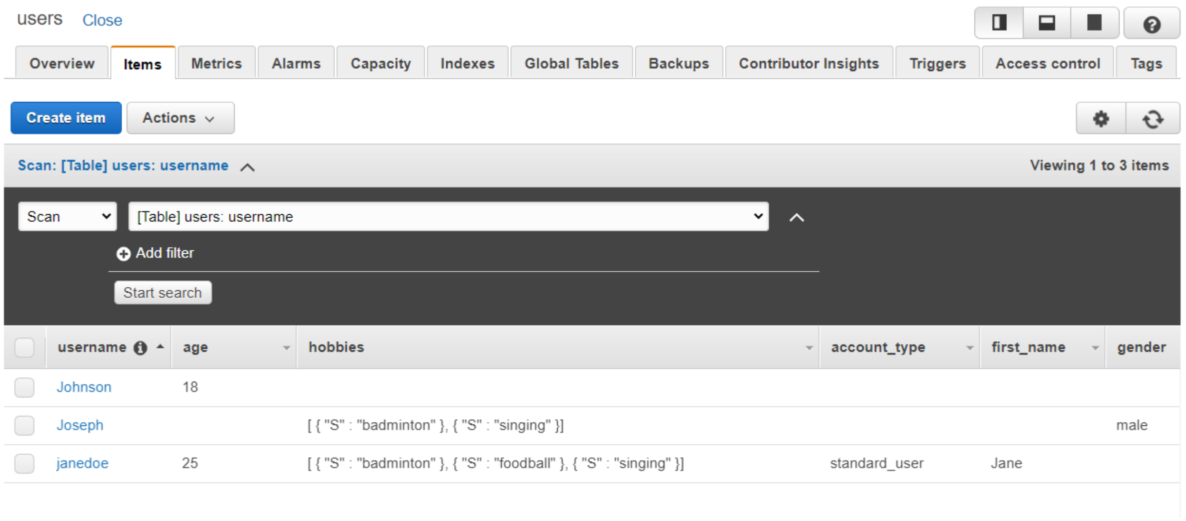
```
#table_scan()

def insert_item_db():
    response = table.put_item(
        Item={
        'username': 'janedoe',
        'first_name': 'Jane',
        'last_name': 'Doe',
        'age': 25,
        'hobbies':['badminton', 'foodball','singing'],
        'account_type': 'standard_user'
        }
    )
    pprint (response)

insert_item_db()
```

The **put_item()** will return the follow JSON object.



```
>python dynamodb.py
{'ResponseMetadata': {'HTTPHeaders': {'connection': 'keep-alive',
                                      'content-length': '2',
                                      'content-type': 'application/x-amz-json-1.0',
                                      'date': 'Mon, 10 Aug 2020 07:13:53 GMT',
                                      'server': 'Server',
                                      'x-amz-crc32': '2745614147',
                                      'x-amzn-requestid': 'GI4U1BKJTIPS2A0A4963N0NA93VV4KQNSO5AEMVJF66Q9ASUAAJG'},
                      'HTTPStatusCode': 200,
                      'RequestId': 'GI4U1BKJTIPS2A0A4963N0NA93VV4KQNSO5AEMVJF66Q9ASUAAJG',
                      'RetryAttempts': 0}}
```

Verify in DynamoDB website that the item is created.



Scan: [Table] users: username   Viewing 1 to 3 items

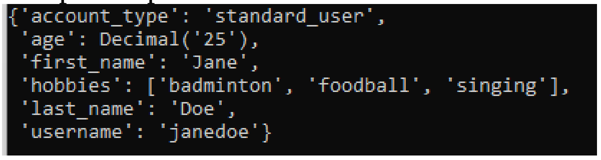
Scan   [Table] users: username

username	age	hobbies	account_type	first_name	gender
Johnson	18				
Joseph		[{ "S" : "badminton" }, { "S" : "singing" }]			male
janedoe	25	[{ "S" : "badminton" }, { "S" : "foodball" }, { "S" : "singing" }]	standard_user	Jane	

Comment out the call to `insert_item_db().`
Add the following code to retrieve an item using the primary key.

```
#insert_item_db()

def get_db_item(): #retrieve an item using primary key
    response = table.get_item(
        Key={
            'username': 'janedoe'
        }
    )
    item = response['Item']

    print(item)


get_db_item()
```

Sample output:



```
{'account_type': 'standard_user',
 'age': Decimal('25'),
 'first_name': 'Jane',
 'hobbies': ['badminton', 'foodball', 'singing'],
 'last_name': 'Doe',
 'username': 'janedoe'}
```

6. Example: Movie Table

6.1. Create Table

Create a table named Movies. The primary key for the table is composed of the following attributes:

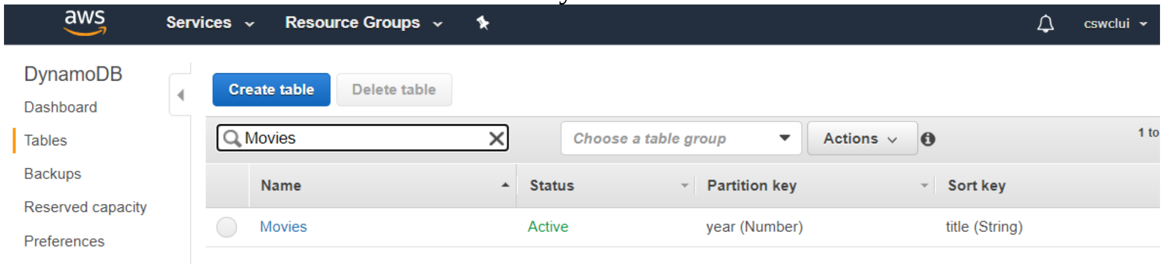
- year – The partition key. The AttributeType is N for number.
- title – The sort key. The AttributeType is S for string.

Copy the following program and paste it into a file named **MoviesCreateTable.py** and provide your AWS Access Key ID and Secret Access key.

```
import boto3

session = boto3.Session(
    aws_access_key_id = '[Your AWS Access Key ID]',
    aws_secret_access_key='[Your AWS Secret Access Key]'
)

dynamodb = session.resource('dynamodb', region_name='us-east-2' )

def create_movie_table():
    table = dynamodb.create_table(
        TableName='Movies',
        KeySchema=[
            {
                'AttributeName': 'year',
                'KeyType': 'HASH'  # Partition key
            },
            {
                'AttributeName': 'title',
                'KeyType': 'RANGE'  # Sort key
            }
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'year',
                'AttributeType': 'N'
            },
            {
                'AttributeName': 'title',
                'AttributeType': 'S'
            },

        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 5,
            'WriteCapacityUnits': 5
        }
    )
    return table


movie_table = create_movie_table()
print("Table status:", movie_table.table_status)
```

Execute the python script.
python MoviesCreateTable.py

Check that the **Movies** table is created in DynamoDB.



Name	Status	Partition key	Sort key
Movies	Active	year (Number)	title (String)

6.2. Load Sample Data

Download and unzip the movie data is in JSON format in the current folder.
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/moviedata.zip

Create python script **MoviesLoadData.py**

```
import boto3
import json
from decimal import Decimal

session = boto3.Session(
    aws_access_key_id = '[Your AWS Access Key ID]',
    aws_secret_access_key='[Your AWS Secret Access Key]'
)

dynamodb = session.resource('dynamodb', region_name='us-east-2' )

def load_movies(movies):
    table = dynamodb.Table('Movies')
    for movie in movies:
        year = int(movie['year'])
        title = movie['title']
        print("Adding movie:", year, title)
        table.put_item(Item=movie)


with open("moviedata.json") as json_file:
    movie_list = json.load(json_file, parse_float=Decimal)
load_movies(movie_list)
```

Execute the script:
python MoviesLoadData.py

6.3. Reading data from Dynamo DB tables

Options for reading data from DynamoDB tables

- GetItem
- Query
- Scan

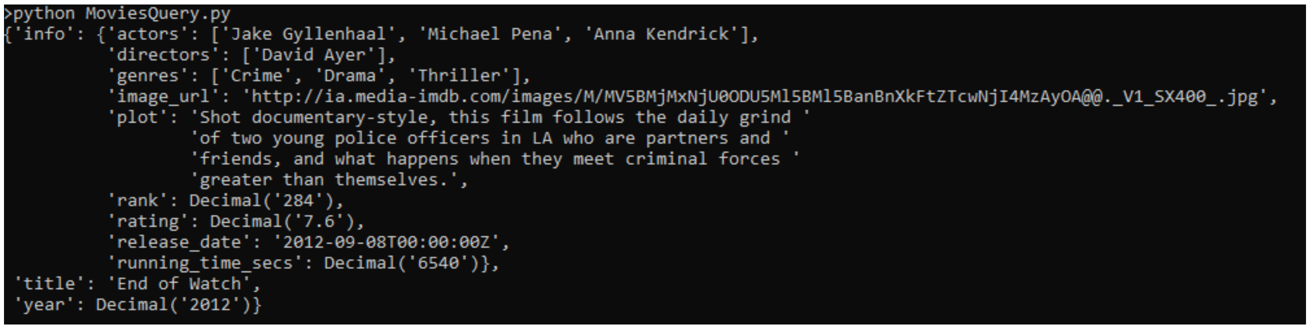
Create a python script **MoviesQuery.py**.

```
import boto3
from pprint import pprint
from boto3.dynamodb.conditions import Key, Attr

session = boto3.Session(
    aws_access_key_id = '[Your AWS Access Key ID]',
    aws_secret_access_key='[Your AWS Secret Access Key]'
)

dynamodb = session.resource('dynamodb', region_name='us-east-2' )
```

Example 1: Read an item

Use the **get_item** method to read a single item. You must specify the whole primary key (i.e. year and title).

```
def read_item(year, title):
    table = dynamodb.Table('Movies')
    response = table.get_item(Key={'year': year, 'title': title})
    movie = response['Item']
    pprint(movie)

read_item(2012, 'End of Watch')
```

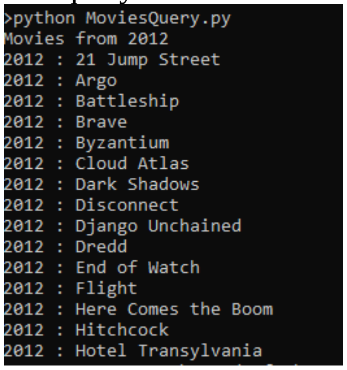


```
>python MoviesQuery.py
{'info': {'actors': ['Jake Gyllenhaal', 'Michael Pena', 'Anna Kendrick'],
          'directors': ['David Ayer'],
          'genres': ['Crime', 'Drama', 'Thriller'],
          'image_url': 'http://ia.media-imdb.com/images/M/MV5BMjMxNjU0ODU5Ml5BMl5BanBnXkFtZTcwNjI4MzAyOA@@._V1_SX400_.jpg',
          'plot': 'Shot documentary-style, this film follows the daily grind '
                  'of two young police officers in LA who are partners and '
                  'friends, and what happens when they meet criminal forces '
                  'greater than themselves.',
          'rank': Decimal('284'),
          'rating': Decimal('7.6'),
          'release_date': '2012-09-08T00:00:00Z',
          'running_time_secs': Decimal('6540')},
 'title': 'End of Watch',
 'year': Decimal('2012')}
```

Example 2: Query All Movies Released in a Year (e.g. 2012)

```
def query_movies(year):

    table = dynamodb.Table('Movies')

    response = table.query(

        KeyConditionExpression=Key('year').eq(year)

    )

    movies = response['Items']



    print(f"Movies from {year}")



    for movie in movies:

        print(movie['year'], ":", movie['title'])



query_movies(2012)
```

The query results will be ordered by the sort key (Title).



```
>python MoviesQuery.py
Movies from 2012
2012 : 21 Jump Street
2012 : Argo
2012 : Battleship
2012 : Brave
2012 : Byzantium
2012 : Cloud Atlas
2012 : Dark Shadows
2012 : Disconnect
2012 : Django Unchained
2012 : Dredd
2012 : End of Watch
2012 : Flight
2012 : Here Comes the Boom
2012 : Hitchcock
2012 : Hotel Transylvania
```

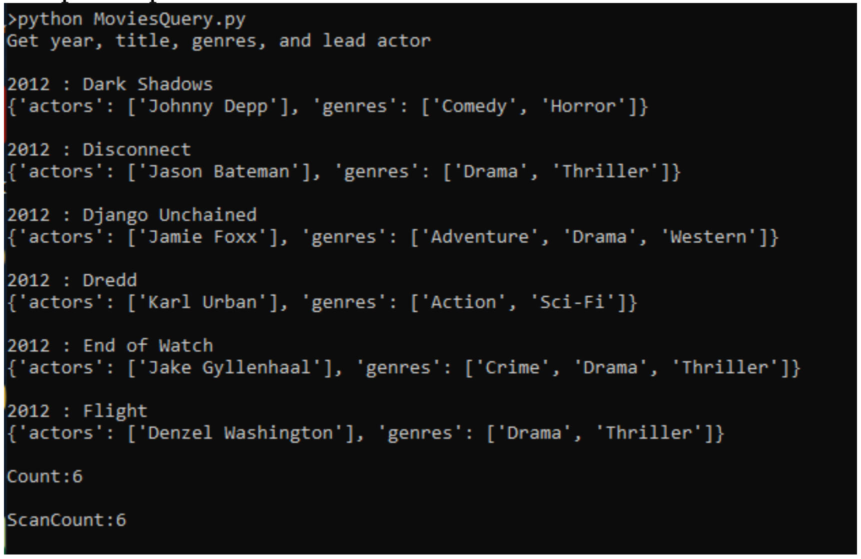
Example 3: Query All Movies Released in a Year (e.g. 2012) with Certain Titles (e.g. beginning with the letter "A" through the letter "L").

- Amazon DynamoDB returns all the item attributes by default. To get only some, rather than all of the attributes, use a projection expression
- If you need to write an expression containing an attribute name that conflicts with a DynamoDB reserved word (e.g. year), we can define an expression attribute name to use in the place of the reserved word.
- Use the KeyConditionExpression parameter to provide a specific value for the partition key.
  - The Query operation will return all of the items from the table or index with that partition key value.
  - We scan optionally narrow the scope of the Query operation by specifying a sort key value and a comparison operator in KeyConditionExpression.

```
def query_and_project_movies(year):

    table = dynamodb.Table('Movies')



    # Expression attribute names can only reference items in the projection expression.

    response = table.query(ProjectionExpression="#yr, title, info.genres,
info.actors[0]",

        ExpressionAttributeNames={"#yr": "year"},

        KeyConditionExpression= Key('year').eq(year) & Key('title').between('D', 'H')

        )



    print(f"Get year, title, genres, and lead actor")



    movies = response['Items']



    for movie in movies:

        print(f"\n{movie['year']} : {movie['title']}")

        pprint(movie['info'])



    print(f"\nCount:{response['Count']}")

    print(f"\nScanCount:{response['ScannedCount']}")
```

```
query_and_project_movies(2012)
```

Sample output:



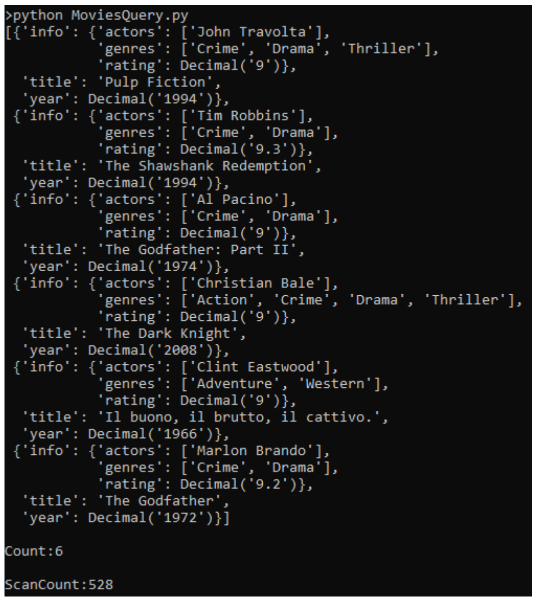
```
>python MoviesQuery.py
Get year, title, genres, and lead actor

2012 : Dark Shadows
{'actors': ['Johnny Depp'], 'genres': ['Comedy', 'Horror']}

2012 : Disconnect
{'actors': ['Jason Bateman'], 'genres': ['Drama', 'Thriller']}

2012 : Django Unchained
{'actors': ['Jamie Foxx'], 'genres': ['Adventure', 'Drama', 'Western']}

2012 : Dredd
{'actors': ['Karl Urban'], 'genres': ['Action', 'Sci-Fi']}

2012 : End of Watch
{'actors': ['Jake Gyllenhaal'], 'genres': ['Crime', 'Drama', 'Thriller']}

2012 : Flight
{'actors': ['Denzel Washington'], 'genres': ['Drama', 'Thriller']}

Count:6

ScanCount:6
```

Example 4: Use table scan to find all movies with title that begins with 'K'.

```
def table_scan1():

    table = dynamodb.Table('Movies')

    response = table.scan(

        ProjectionExpression="#yr, title, info.genres, info.actors[0]",

        ExpressionAttributeNames={"#yr": "year"},

        FilterExpression=Key('title').begins_with('K')

        )

    pprint(response['Items'])

    print(f"\nCount:{response['Count']}")

    print(f"\nScanCount:{response['ScannedCount']}")


table_scan1()
```

Example 5: Use table scan to find all movies with info.rating>=9

```
from decimal import *

def table_scan2():

    table = dynamodb.Table('Movies')
    response = table.scan(

        ProjectionExpression="#yr, title, info.genres, info.actors[0], info.rating",

        ExpressionAttributeNames={"#yr": "year"},

        FilterExpression=Attr('info.rating').gte(Decimal(9))



        )

    pprint(response['Items'])

    print(f"\nCount:{response['Count']}")

    print(f"\nScanCount:{response['ScannedCount']}")


table_scan2()
```



```
>python MoviesQuery.py
[{'info': {'actors': ['John Travolta'],
           'genres': ['Crime', 'Drama', 'Thriller'],
           'rating': Decimal('9')},
  'title': 'Pulp Fiction',
  'year': Decimal('1994')},
 {'info': {'actors': ['Tim Robbins'],
           'genres': ['Crime', 'Drama'],
           'rating': Decimal('9.3')},
  'title': 'The Shawshank Redemption',
  'year': Decimal('1994')},
 {'info': {'actors': ['Al Pacino'],
           'genres': ['Crime', 'Drama'],
           'rating': Decimal('9')},
  'title': 'The Godfather: Part II',
  'year': Decimal('1974')},
 {'info': {'actors': ['Christian Bale'],
           'genres': ['Action', 'Crime', 'Drama', 'Thriller'],
           'rating': Decimal('9')},
  'title': 'The Dark Knight',
  'year': Decimal('2008')},
 {'info': {'actors': ['Clint Eastwood'],
           'genres': ['Adventure', 'Western'],
           'rating': Decimal('9')},
  'title': 'Il buono, il brutto, il cattivo.',
  'year': Decimal('1966')},
 {'info': {'actors': ['Marlon Brando'],
           'genres': ['Crime', 'Drama'],
           'rating': Decimal('9.2')},
  'title': 'The Godfather',
  'year': Decimal('1972')}]

Count:6

ScanCount:528
```

6.4. CRUD Operations

Refer to the following tutorial:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.03.html

7. References

- DynamoDB Developer Guide
  - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
  - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html
- BOTO 3
  - https://boto3.amazonaws.com/v1/documentation/api/latest/guide/dynamodb.html
- Tutorial: Python and DynamoDB
  - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.html
- Tutorial: Create and Query a NoSQL Table
  - https://aws.amazon.com/getting-started/hands-on/create-nosql-table/