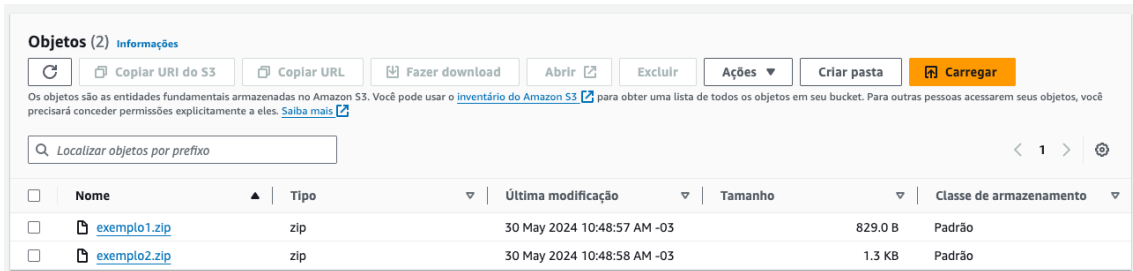


### Exercício 1 – Criar a Tabela ProductCatalog com chave de Partição Id e provisionamento

- 1) Primeiro passo foi criar um bucket no S3 e anexar os exemplos



The screenshot shows the Amazon S3 console interface. At the top, there are buttons for 'Copiar URI do S3', 'Copiar URL', 'Fazer download', 'Abrir', 'Excluir', 'Ações', 'Criar pasta', and 'Carregar'. Below these buttons, there is a search bar labeled 'Localizar objetos por prefixo'. The main area displays a table of objects:

<input type="checkbox"/>	Nome	Tipo	Última modificação	Tamanho	Classe de armazenamento
<input type="checkbox"/>	exemplo1.zip	zip	30 May 2024 10:48:57 AM -03	829.0 B	Padrão
<input type="checkbox"/>	exemplo2.zip	zip	30 May 2024 10:48:58 AM -03	1.3 KB	Padrão

- 2) Copiar o link do bucket do S3, sincronizar com o cloud9 e por fim descomprimir o arquivo

```
voclabs:~/environment $ aws s3 sync s3://labedb6diego .
download: s3://labedb6diego/exemplo2.zip to ./exemplo2.zip
download: s3://labedb6diego/exemplo1.zip to ./exemplo1.zip
voclabs:~/environment $ unzip exemplo1.zip
Archive:  exemplo1.zip
  inflating: exemplo1/ProductCatalog.json
voclabs:~/environment $ unzip exemplo2.zip
Archive:  exemplo2.zip
  inflating: exemplo2/Forum.json
  inflating: exemplo2/Resposta.json
  inflating: exemplo2/Thread.json
```

Exercício: DynamoDB (Ex 1, Ex 2 e Ex3)

Grupo: Diego Moura, Gisele Siqueira, Marcelo Barbugli,  
Matheus Higa, Ricardo Geroto, Roberto Eyama

- 3) Realizar a criação da Tabela ProductCatalog via cloud9

```
inflating: exemplo2/Thread.json
voclabs:~/environment $ aws dynamodb create-table \
> --table-name ProductCatalog \
> --attribute-definitions AttributeName=Id,AttributeType=N \
> --key-schema AttributeName=Id,KeyType=HASH \
> --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5
{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Id",
        "AttributeType": "N"
      }
    ],
    "TableName": "ProductCatalog",
    "KeySchema": [
      {
        "AttributeName": "Id",
        "KeyType": "HASH"
      }
    ],
    "TableStatus": "CREATING",
    "CreationDateTime": "2024-05-30T14:03:41.435000+00:00",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    }
  },
}
```

- 4) Agora será realizado um comando para listar as tabelas existentes

```
voclabs:~/environment $ aws dynamodb list-tables
{
  "TableNames": [
    "NomeDaTabela",
    "ProductCatalog",
    "sells"
  ]
}
```

- 5) Agora irei utilizar o comando para carregar o arquivo Json dataset – Exemplo 1 e validar a criação no DynamoDB

```
voclabs:~/environment $ aws dynamodb batch-write-item --request-items file://exemplo1/ProductCatalog.json
{
  "UnprocessedItems": {}
}
```

## Curso de Especialização em Big Data – Escola Politécnica da USP

Exercício: DynamoDB (Ex 1, Ex 2 e Ex3)

Grupo: Diego Moura, Gisele Siqueira, Marcelo Barbugli,  
Matheus Higa, Ricardo Geroto, Roberto Eyama

**Tabelas (3)** [Informações](#)

<input type="checkbox"/>	Nome	Status	Chave de partição	Chave de classificaç...	Índices	Proteção contra exclus...	Modo de capacidade de leit...	Modo de capa...
<input type="checkbox"/>	<a href="#">NomeDaTabela</a>	Ativo	ID (S)	-	0	Desativado	Provisionada (5)	Provisionada (5)
<input type="checkbox"/>	<a href="#">ProductCatalog</a>	Ativo	Id (N)	-	0	Desativado	Provisionada (10)	Provisionada (5)
<input type="checkbox"/>	<a href="#">sells</a>	Ativo	user (S)	order (S)	2	Desativado	Provisionada (1)	Provisionada (5)

- 6) Agora vou utilizar o comando scan para trazer as informações da tabela

```
voclabs:~/environment $ aws dynamodb scan --table-name ProductCatalog
{
  "Items": [
    {
      "Title": {
        "S": "18-Bike-204"
      },
      "Price": {
        "N": "500"
      },
      "Brand": {
        "S": "Brand-Company C"
      },
      "Description": {
        "S": "205 Description"
      },
      "Color": {
        "L": [
          {
            "S": "Red"
          },
          {
            "S": "Black"
          }
        ]
      },
      "ProductCategory": {
        "S": "Bicycle"
      },
      "Id": {
        "N": "205"
      },
      "BicycleType": {
        "S": "Hybrid"
      }
    },
    {
      "Title": {
        "S": "19-Bike-203"
      },
      "Price": {
        "N": "300"
      },
      "Brand": {
        "S": "Brand-Company B"
      },
      "Description": {
        "S": "203 Description"
      },
      "Color": {
        "L": [
```

## **Curso de Especialização em Big Data – Escola Politécnica da USP**

Exercício: DynamoDB (Ex 1, Ex 2 e Ex3)

Grupo: Diego Moura, Gisele Siqueira, Marcelo Barbugli,  
Matheus Higa, Ricardo Geroto, Roberto Eyama



- 7) Utilizarei o comando para pegar um item da tabela

```
voclabs:~/environment $ aws dynamodb get-item \  
> --table-name ProductCatalog \  
> --key '{"Id":{"N":"101"}}' \  
{  
  "Item": {  
    "Title": {  
      "S": "Book 101 Title"  
    },  
    "InPublication": {  
      "BOOL": true  
    },  
    "PageCount": {  
      "N": "500"  
    },  
    "Dimensions": {  
      "S": "8.5 x 11.0 x 0.5"  
    },  
    "ISBN": {  
      "S": "111-1111111111"  
    },  
    "Authors": {  
      "L": [  
        {  
          "S": "Author1"  
        }  
      ]  
    },  
    "Price": {  
      "N": "2"  
    },  
    "ProductCategory": {  
      "S": "Book"  
    },  
    "Id": {  
      "N": "101"  
    }  
  }  
}
```

- 8) Agora vou realizar o update de um item na tabela ProductCatalog e validar na console

```
voclabs:~/environment $ aws dynamodb update-item \
> --table-name ProductCatalog \
> --key '{"Id": {"N": "201"}}' \
> --update-expression "SET #Color = list_append(#Color, :values)" \
> --expression-attribute-names '{"#Color": "Color"}' \
> --expression-attribute-values '":values" : [{"S": "Blue"}, {"S": "Yellow"}]' \
> --return-consumed-capacity TOTAL
{
  "ConsumedCapacity": {
    "TableName": "ProductCatalog",
    "CapacityUnits": 1.0
  }
}
```

Concluído. Unidades de capacidade de leitura consumidas: 0.5

Itens retornados (8)

	Id (Número)	Brand	Color	Descr
<input type="checkbox"/>	<a href="#">205</a>	Brand-Com...	[{"S": "Red"}, {"S": "Black"}]	205 D
<input type="checkbox"/>	<a href="#">203</a>	Brand-Com...	[{"S": "Red"}, {"S": "Green"}, {"S": "Black"}]	203 D
<input type="checkbox"/>	<a href="#">202</a>	Brand-Com...	[{"S": "Green"}, {"S": "Black"}]	202 D
<input type="checkbox"/>	<a href="#">201</a>	Mountain A	[{"S": "Red"}, {"S": "Black"}, {"S": "Blue"}, {"S": "Yellow"}]	201 D
<input type="checkbox"/>	<a href="#">204</a>	Brand-Com...	[{"S": "Red"}]	204 D
<input type="checkbox"/>	<a href="#">102</a>			
<input type="checkbox"/>	<a href="#">103</a>			
<input type="checkbox"/>	<a href="#">101</a>			

- 9) Agora vamos remover a tabela

```
voclabs:~/environment $ aws dynamodb delete-table --table-name ProductCatalog
{
  "TableDescription": {
    "TableName": "ProductCatalog",
    "TableStatus": "DELETING",
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 10,
      "WriteCapacityUnits": 5
    },
    "TableSizeBytes": 0,
    "ItemCount": 0,
    "TableArn": "arn:aws:dynamodb:us-east-1:374741742859:table/ProductCatalog",
    "TableId": "b39550f9-b365-42f1-b4f4-ca6cdaa4b91c",
    "DeletionProtectionEnabled": false
  }
}
```

Exercício 2 – Fórum de Discussão

- 1) Aqui vou iniciar criando as 3 tabelas (Forum, Thread e Resposta) e validar no DynamoDB, com capacidades de leitura e escrita diferentes

```
voclabs:~/environment $ aws dynamodb create-table \  
> --table-name Forum \  
> --attribute-definitions \  
>   AttributeName=Nome,AttributeType=S \  
> --key-schema \  
>   AttributeName=Nome,KeyType=HASH \  
> --provisioned-throughput \  
> ReadCapacityUnits=10,WriteCapacityUnits=5  
{
```

```
voclabs:~/environment $ aws dynamodb create-table \  
> --table-name Thread \  
> --attribute-definitions \  
>   AttributeName=NomeForum,AttributeType=S \  
>   AttributeName=Tema,AttributeType=S \  
> --key-schema \  
>   AttributeName=NomeForum,KeyType=HASH \  
>   AttributeName=Tema,KeyType=RANGE \  
> --provisioned-throughput \  
> ReadCapacityUnits=10,WriteCapacityUnits=5  
{
```

```
voclabs:~/environment $ aws dynamodb create-table \  
> --table-name Resposta \  
> --attribute-definitions \  
>   AttributeName=Id,AttributeType=S \  
>   AttributeName=DataResposta,AttributeType=S \  
> --key-schema \  
>   AttributeName=Id,KeyType=HASH \  
>   AttributeName=DataResposta,KeyType=RANGE \  
> --provisioned-throughput \  
> ReadCapacityUnits=10,WriteCapacityUnits=5  
{
```



Tabelas (5) Informações

Localizar tabelas por nome de tabela Qualquer chave de etiqueta Qualquer valor de etiqueta

	Nome	Status	Chave de partição	Chave de classificaç...	Índices	Proteção contra exclus...	Modo de capacidade de leit...	Modo de capaci
<input type="checkbox"/>	<a href="#">Thread</a>	Ativo	NomeForum (S)	Tema (S)	0	Desativado	Provisionada (10)	Provisionada (5)
<input type="checkbox"/>	<a href="#">sells</a>	Ativo	user (S)	order (S)	2	Desativado	Provisionada (1)	Provisionada (1)
<input type="checkbox"/>	<a href="#">Resposta</a>	Ativo	Id (S)	DataResposta (S)	0	Desativado	Provisionada (10)	Provisionada (5)
<input type="checkbox"/>	<a href="#">NomeDaTabela</a>	Ativo	ID (S)	-	0	Desativado	Provisionada (5)	Provisionada (5)
<input type="checkbox"/>	<a href="#">Forum</a>	Ativo	Nome (S)	-	0	Desativado	Provisionada (10)	Provisionada (5)

2) Agora carregar os arquivos do exemplo 2

```
voclabs:~/environment $ aws dynamodb batch-write-item --request-items file://exemplo2/Forum.json
{
  "UnprocessedItems": {}
}
voclabs:~/environment $ aws dynamodb batch-write-item --request-items file://exemplo2/Thread.json
{
  "UnprocessedItems": {}
}
voclabs:~/environment $ awsdynamodb batch-write-item --request-items file://exemplo2/Resposta.json
bash: awsdynamodb: command not found
voclabs:~/environment $ aws dynamodb batch-write-item --request-items file://exemplo2/Resposta.json
{
  "UnprocessedItems": {}
}
voclabs:~/environment $
```

## **Curso de Especialização em Big Data – Escola Politécnica da USP**

Exercício: DynamoDB (Ex 1, Ex 2 e Ex3)

Grupo: Diego Moura, Gisele Siqueira, Marcelo Barbugli,  
Matheus Higa, Ricardo Geroto, Roberto Eyama



- 3) Vou realizar uma listagem na tabela Resposta

```
voclabs:~/environment $ aws dynamodb scan --table-name Resposta
{
  "Items": [
    {
      "PostadoPor": {
        "S": "Usuario A"
      },
      "Id": {
        "S": "Gatos#Gatos Topico 1"
      },
      "DataResposta": {
        "S": "2015-09-15T19:58:22.947Z"
      },
      "Mensagem": {
        "S": "Gatos Topico 1 Resposta 1 texto"
      }
    },
    {
      "PostadoPor": {
        "S": "Usuario B"
      },
      "Id": {
        "S": "Gatos#Gatos Topico 1"
      },
      "DataResposta": {
        "S": "2015-09-22T19:58:22.947Z"
      },
      "Mensagem": {
        "S": "Gatos Topico 1 Resposta 2 texto"
      }
    },
    {
      "PostadoPor": {
        "S": "Usuario A"
      },
      "Id": {
        "S": "Gatos#Gatos Topico 2"
      },
      "DataResposta": {
        "S": "2015-09-29T19:58:22.947Z"
      },
      "Mensagem": {
        "S": "Gatos Topico 2 Resposta 1 texto"
      }
    },
    {
      "PostadoPor": {
        "S": "Usuario A"
      },
      "Id": {
        "S": "Gatos#Gatos Topico 2"
      },
      "DataResposta": {
        "S": "2015-10-05T19:58:22.947Z"
      },
      "Mensagem": {
        "S": "Gatos Topico 2 Resposta 2 texto"
      }
    }
  ],
  "Count": 4,
  "ScannedCount": 4,
  "ConsumedCapacity": null
}
```

- 4) Agora vou explorar os dados da tabela para olhar os itens que tem mais de uma Thread e mais de 50 visualizações na tabela Forum

```
voclabs:~/environment $ aws dynamodb scan \
> --table-name Forum \
> --filter-expression 'Threads >=
> :threads AND Vistas >= :views' \
> --expression-attribute-values '{
>     ":threads" : {"N": "1"},
>     ":views" : {"N": "50"}
> }' \
> --return-consumed-capacity TOTAL
{
  "Items": [
    {
      "Nome": {
        "S": "Gatos"
      },
      "Threads": {
        "N": "2"
      },
      "Categoria": {
        "S": "Felinos"
      },
      "Vistas": {
        "N": "1000"
      },
      "Mensagem": {
        "N": "4"
      }
    }
  ],
  "Count": 1,
  "ScannedCount": 2,
  "ConsumedCapacity": {
    "TableName": "Forum",
    "CapacityUnits": 0.5
  }
}
```

- 5) Vou realizar uma consulta para retornar apenas a primeira resposta do tópico e a resposta mais recente, na tabela resposta

```
> --table-name Resposta \
> --key-condition-expression 'Id = :Id' \
> --expression-attribute-values '{
>   ":Id" : {"S": "Gatos#Gatos Topico 1"}}'\
> --max-items 1 \
> --scan-index-forward \
> --return-consumed-capacity TOTAL
{
  "Items": [
    {
      "PostadoPor": {
        "S": "Usuario A"
      },
      "Id": {
        "S": "Gatos#Gatos Topico 1"
      },
      "DataResposta": {
        "S": "2015-09-15T19:58:22.947Z"
      },
      "Mensagem": {
        "S": "Gatos Topico 1 Resposta 1 texto"
      }
    }
  ],
  "Count": 2,
  "ScannedCount": 2,
  "ConsumedCapacity": {
    "TableName": "Resposta",
    "CapacityUnits": 0.5
  },
  "NextToken": "eyJFeGNsdXNpdmVTdGFydEtleSI6IG51bGwsICJib3RvX3RydW5jYXRlX2Ftb3VudCI6IDF9"
```

- 6) Realizarei a atualização de um item na tabela fórum

## **Curso de Especialização em Big Data – Escola Politécnica da USP**

Exercício: DynamoDB (Ex 1, Ex 2 e Ex3)

Grupo: Diego Moura, Gisele Siqueira, Marcelo Barbugli,  
Matheus Higa, Ricardo Geroto, Roberto Eyama



- 7) Realizarei a remoção de um item da tabela Resposta

```
voclabs:~/environment $ aws dynamodb query \
> --table-name Resposta \
> --key-condition-expression 'Id = :Id' \
> --expression-attribute-values '{
>   ":Id" : {"S": "Gatos#Gatos Topico 1"}
> }'\
> --return-consumed-capacity TOTAL
{
  "Items": [
    {
      "PostadoPor": {
        "S": "Usuario A"
      },
      "Id": {
        "S": "Gatos#Gatos Topico 1"
      },
      "DataResposta": {
        "S": "2015-09-15T19:58:22.947Z"
      },
      "Mensagem": {
        "S": "Gatos Topico 1 Resposta 1 texto"
      }
    },
    {
      "PostadoPor": {
        "S": "Usuario B"
      },
      "Id": {
        "S": "Gatos#Gatos Topico 1"
      },
      "DataResposta": {
        "S": "2015-09-22T19:58:22.947Z"
      },
      "Mensagem": {
        "S": "Gatos Topico 1 Resposta 2 texto"
      }
    }
  ],
  "Count": 2,
  "ScannedCount": 2,
  "ConsumedCapacity": {
    "TableName": "Resposta",
    "CapacityUnits": 0.5
  }
}
```

## Exercício: DynamoDB (Ex 1, Ex 2 e Ex3)

Grupo: Diego Moura, Gisele Siqueira, Marcelo Barbugli, Matheus Higa, Ricardo Geroto, Roberto Eyama

### Exercício 3 – Criando e Deletando tabela no DynamoDB com Python

- 1) Vamos criar uma tabela usando o comando pre-existente do python

```
1 import boto3
2
3 # boto3 is the AWS SDK library for Python.
4 # We can use the low-level client to make API calls to DynamoDB.
5 client = boto3.client('dynamodb', region_name='us-east-1')
6
7 try:
8     resp = client.create_table(
9         TableName="Books",
10         # Declare your Primary Key in the KeySchema argument
11         KeySchema=[
12             {
13                 "AttributeName": "Author",
14                 "KeyType": "HASH"
15             },
16             {
17                 "AttributeName": "Title",
18                 "KeyType": "RANGE"
19             }
20         ],
21         # Any attributes used in KeySchema or Indexes must be declared in AttributeDefinitions
22         AttributeDefinitions=[
23             {
24                 "AttributeName": "Author",
25                 "AttributeType": "S"
26             },
27             {
28                 "AttributeName": "Title",
29                 "AttributeType": "S"
30             }
31         ],
32         # ProvisionedThroughput controls the amount of data you can read or write to DynamoDB per second.
33         # You can control read and write capacity independently.
34         ProvisionedThroughput={
35             "ReadCapacityUnits": 1,
36             "WriteCapacityUnits": 1
37         }
38     )
39     print("Table created successfully!")
40 except Exception as e:
41     print("Error creating table:")
42     print(e)
43
```

bash - "ip-172-31-28-125 x" create\_table.py - Stopped x

Run Command: create\_table.py

Table created successfully!



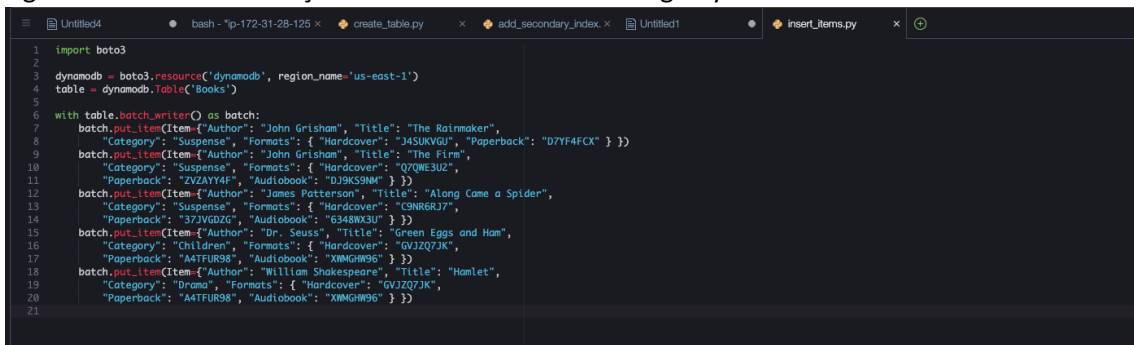
### 2) Validar se a tabela foi criada no console



The screenshot shows the AWS Management Console interface for a DynamoDB table named 'Books'. On the left, a sidebar lists tables: 'Books', 'Forum', 'NomeDaTabela', 'Resposta', 'sells', and 'Thread'. The 'Books' table is selected. The main panel shows the 'Visão geral' (Overview) tab. A warning banner at the top states: 'Proteja sua tabela do DynamoDB contra gravações e exclusões acidentais' (Protect your DynamoDB table from accidental writes and deletions). Below this, the 'Informações gerais' (General information) section displays the following details:

Chave de partição	Chave de classificação	Modo de capacidade	Status da tabela
Author (String)	Title (String)	Provisionada	Ativo
Alarmes: Nenhum alarme ativo	Recuperação em um ponto anterior no tempo (PITR): Desativado	Política baseada em recursos: Não ativo	

### 3) Agora irei realizar a inserção de itens na tabela via código Python



```
1 import boto3
2
3 dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
4 table = dynamodb.Table('Books')
5
6 with table.batch_writer() as batch:
7     batch.put_item(Item={'Author': 'John Grisham', 'Title': 'The Rainmaker',
8     'Category': 'Suspense', 'Formats': {'Hardcover': 'J4SUKVGV', 'Paperback': 'D7YF4FCX'}})
9     batch.put_item(Item={'Author': 'John Grisham', 'Title': 'The Firm',
10     'Category': 'Suspense', 'Formats': {'Hardcover': 'Q7QME3UE',
11     'Paperback': 'ZV2AYY4F', 'Audiobook': 'DJ9KS9NM'}})
12     batch.put_item(Item={'Author': 'James Patterson', 'Title': 'Along Came a Spider',
13     'Category': 'Suspense', 'Formats': {'Hardcover': 'C9NR6RJ7',
14     'Paperback': '37JVG0Z6', 'Audiobook': '6348WX3U'}})
15     batch.put_item(Item={'Author': 'Dr. Seuss', 'Title': 'Green Eggs and Ham',
16     'Category': 'Children', 'Formats': {'Hardcover': 'GVJZQ7JK',
17     'Paperback': 'A4TFUR38', 'Audiobook': 'XIMGHW96'}})
18     batch.put_item(Item={'Author': 'William Shakespeare', 'Title': 'Hamlet',
19     'Category': 'Drama', 'Formats': {'Hardcover': 'GVJZQ7JK',
20     'Paperback': 'A4TFUR38', 'Audiobook': 'XIMGHW96'}})
21
```

- 4) Validarei a existencia dos itens na console da AWS

**Books** Visualização automática Visualizar detalhes da tabela

▼ Verificar ou consultar itens

☒ Verificar ☐ Consulta

Selecionar uma tabela ou índice: Tabela - Books

Selecionar projeção de atributos: Todos os atributos

▶ Filtros

**Executar** Redefinir

✔ Concluído. Unidades de capacidade de leitura consumidas: 0.5

**Itens retornados (5)** Atualizar Ações ▼ Criar item

<input type="checkbox"/>	Author (String) ▼	Title (String) ▼	Category ▼	Formats ▼
<input type="checkbox"/>	<a href="#">William Shakespeare</a>	Hamlet	Drama	{ "Hardcover": { "S": "GVJZQ7JK" }, "Paperback": { "S": ...
<input type="checkbox"/>	<a href="#">Dr. Seuss</a>	Green Eggs and Ham	Children	{ "Hardcover": { "S": "GVJZQ7JK" }, "Paperback": { "S": ...
<input type="checkbox"/>	<a href="#">James Patterson</a>	Along Came a Spider	Suspense	{ "Hardcover": { "S": "C9NR6RJ7" }, "Paperback": { "S": ...
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Firm	Suspense	{ "Hardcover": { "S": "Q7QWE3U2" }, "Paperback": { "S": ...
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Rainmaker	Suspense	{ "Hardcover": { "S": "J4SUKVGU" }, "Paperback": { "S": ...

- 5) Agora realizarei um uma consulta no python para buscar um item

```
1 import boto3
2
3 dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
4 table = dynamodb.Table('Books')
5
6 resp = table.get_item(Key={"Author": "John Grisham", "Title": "The Rainmaker"})
7
8 print(resp['Item'])
9
```

6) Realizarei essa consulta na Console da AWS

### Books

Visualização automática

Visualizar detalhes da tabela

▼ Verificar ou consultar itens

☐ Verificar

☒ Consulta

Selecionar uma tabela ou índice

Tabela - Books ▼

Selecionar projeção de atributos

Todos os atributos ▼

Author (Chave de partição)

John Grisham

Title (Chave de classificação)

Igual a ▼

The Rainmaker

☐ Classificar de modo decrescente

► Filtros

Executar

Redefinir

✓ Concluído. Unidades de capacidade de leitura consumidas: 0.5

Itens retornados (1)

↻

Ações ▼

Criar item

< 1 >

⚙️

🗖️

<input type="checkbox"/>	Author (String) ▼	Title (String) ▼	Category ▼	Formats ▼
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Rainmaker	Suspense	{ "Hardcover" : { "S" : "J4SUKVGU" }, "Paperback" : { "S" : ...

7) Realizarei a execução do index secundário na tabela books

```

1 # Boto3 is the AWS SDK library for Python.
2 # You can use the low-level client to make API calls to DynamoDB.
3 client = boto3.client('dynamodb', region_name='us-east-1')
4
5 try:
6     resp = client.update_table(
7         TableName="Books",
8         # Any attributes used in your new global secondary index must be declared in AttributeDefinitions
9         AttributeDefinitions=[
10             {
11                 "AttributeName": "Category",
12                 "AttributeType": "S"
13             },
14         ],
15         # This is where you add, update, or delete any global secondary indexes on your table.
16         GlobalSecondaryIndexUpdates=[
17             {
18                 "Create": {
19                     # You need to name your index and specifically refer to it when using it for queries.
20                     "IndexName": "CategoryIndex",
21                     # Like the table itself, you need to specify the key schema for an index.
22                     # For a global secondary index, you can use a simple or composite key schema.
23                     "KeySchema": [
24                         {
25                             "AttributeName": "Category",
26                             "KeyType": "HASH"
27                         },
28                     ],
29                     # You can choose to copy only specific attributes from the original item into the index.
30                     # You might want to copy only a few attributes to save space.
31                     "Projection": {
32                         "ProjectionType": "ALL"
33                     },
34                     # Global secondary indexes have read and write capacity separate from the underlying table.
35                     "ProvisionedThroughput": {
36                         "ReadCapacityUnits": 1,
37                         "WriteCapacityUnits": 1,
38                     }
39                 }
40             }
41         ],
42     )
43     print("Secondary index added!")
44 except Exception as e:
45     print("Error updating table:")
46     print(e)
47

```

8) Verificarei na Console a criação do index secundario

DynamoDB > Tabelas > Books

**Tabelas (6)**

- Books
- Forum
- NomeDaTabela
- Resposta

**Books**

Índices secundários globais (1) [Informações](#)

Nome	Status	Chave de partição	Chave de classificação	Capacidade de leitura	Capacidade de escrita
CategoryIndex	Criando	Category (String)	-	1 O Auto Scaling está desativado	1 O Auto Scaling está desativado

9) Quando o index secundário estiver ativo, irei realizar uma buscando por index Category

▼ Verificar ou consultar itens

☐ Verificar

☒ Consulta

Selecionar uma tabela ou índice

Índice - CategoryIndex ▼

Selecionar projeção de atributos

Atributos projetados ▼

Category (Chave de partição)

Suspense

► Filtros

Executar

Redefinir

✓ Concluído. Unidades de capacidade de leitura consumidas: 0.5

Itens retornados (3)

↺

↻

Ações ▼

Criar Item

<

1

>

⚙

✖

<input type="checkbox"/>	Author (String) ▼	Title (String) ▼	Category ▼	Formats ▼
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Firm	Suspense	{ "Hardcover" : { "S" : "Q7QWE3U2" }, "Paperback" : { "S" : "Q7QWE3U2" } }
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Rainmaker	Suspense	{ "Hardcover" : { "S" : "J4SUKVGU" }, "Paperback" : { "S" : "J4SUKVGU" } }
<input type="checkbox"/>	<a href="#">James Patterson</a>	Along Came a Spider	Suspense	{ "Hardcover" : { "S" : "C9NR6RJ7" }, "Paperback" : { "S" : "C9NR6RJ7" } }

10) Agora realizarei a atualização de um dos itens  
ANTES

Itens retornados (3)

↺

1

↻

⚙️

🔗

Atualizar

Ações ▾

Criar Item

<input type="checkbox"/>	Author (String) ▾	Title (String) ▾	Category ▾	Formats ▾
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Firm	Suspense	{ "Hardcover": { "S": "Q7QWE3U2" }, "Paperback": { "S": "Q7QWE3U2" } }
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Rainmaker	Suspense	{ "Hardcover": { "S": "J4SUKVGU" }, "Paperback": { "S": "J4SUKVGU" } }
<input type="checkbox"/>	<a href="#">James Patterson</a>	Along Came a Spider	Suspense	{ "Hardcover": { "S": "C9NR6RJ7" }, "Paperback": { "S": "C9NR6RJ7" } }

DEPOIS

✓ Concluído. Unidades de capacidade de leitura consumidas: 0.5

✕

Itens retornados (3)

↺

1

↻

⚙️

🔗

Atualizar

Ações ▾

Criar Item

<input type="checkbox"/>	Author (String) ▾	Title (String) ▾	Category ▾	Formats ▾
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Firm	Suspense	{ "Hardcover": { "S": "Q7QWE3U2" }, "Paperback": { "S": "Q7QWE3U2" } }
<input type="checkbox"/>	<a href="#">John Grisham</a>	The Rainmaker	Suspense	{ "Audiobook": { "S": "J4SUKVGU" }, "Paperback": { "S": "J4SUKVGU" } }
<input type="checkbox"/>	<a href="#">James Patterson</a>	Along Came a Spider	Suspense	{ "Hardcover": { "S": "C9NR6RJ7" }, "Paperback": { "S": "C9NR6RJ7" } }

```

1 # The UpdateItem API allows you to update a particular item as identified by its key.
2 resp = table.update_item(
3     Key={"Author": "John Grisham", "Title": "The Rainmaker"},
4     # Expression attribute names specify placeholders for attribute names to use in your update expressions.
5     ExpressionAttributeNames={
6         "#formats": "Formats",
7         "#audiobook": "Audiobook",
8     },
9     # Expression attribute values specify placeholders for attribute values to use in your update expressions.
10    ExpressionAttributeValues={
11        ":id": "8WE3KPTP",
12    },
13    # UpdateExpression declares the updates you want to perform on your item.
14    # For more details about update expressions, see https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.UpdateExpressions.html
15    UpdateExpression="SET #formats.#audiobook = :id",
16 )

```

22

11) Por fim realizaremos a exclusão da tabela

```
1 import boto3
2
3 client = boto3.client('dynamodb', region_name='us-east-1')
4
5 try:
6     resp = client.delete_table(
7         TableName="Books",
8     )
9     print("Table deleted successfully!")
10 except Exception as e:
11     print("Error deleting table:")
12     print(e)
13
```

