

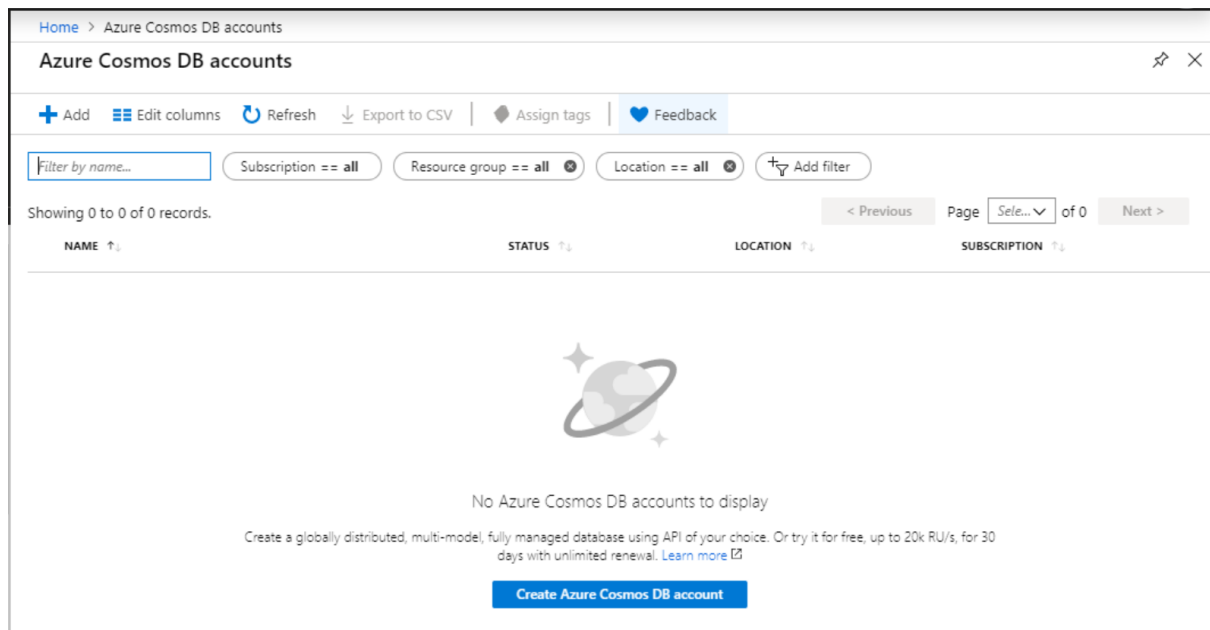
## **Create an Azure Cosmos DB using MongoDB API**

In this tutorial I'm going to show how to:

- Create an Azure Cosmos DB/MongoDB API
- Add data to the Azure Cosmos DB
- Query Azure Cosmos DB through Data Explorer
- Query Azure Cosmos DB through MongoDB Shell

## Step 01: Create an Azure Cosmos DB/MongoDB API

Click + Add to add new Cosmos DB Account.

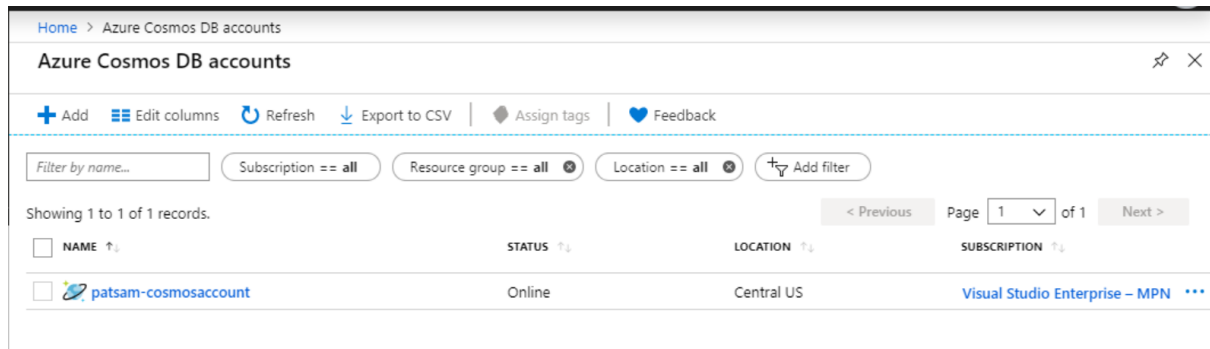



Provide relevant details and click Review + Create.

- Resource group: patsam-rg
- Account name: patsam-cosmosaccount
- API: Azure Cosmos DB for Mongo DB API
- Location: Central US

The screenshot shows the 'Create Azure Cosmos DB Account' form. At the top, there's a breadcrumb 'Home > Azure Cosmos DB > Create Azure Cosmos DB Account'. Below the title, there's a purple banner with a discount offer. The form has tabs: 'Basics', 'Network', 'Tags', and 'Review + create'. The 'Basics' tab is active. The form contains the following fields: 'Subscription' (Visual Studio Enterprise - MPN), 'Resource Group' (patsam-rg), 'Account Name' (patsam-cosmosaccount), 'API' (Azure Cosmos DB for MongoDB API), 'Location' (US: Central US), 'Geo-Redundancy' (Enable/Disable), and 'Multi-region Writes' (Enable/Disable). At the bottom, there are buttons: 'Review + create', 'Previous', and 'Next: Network'.

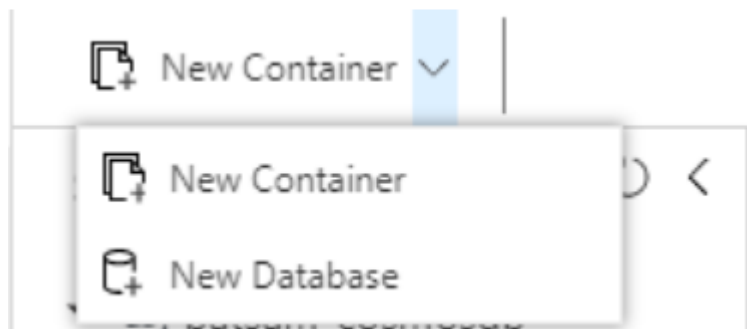
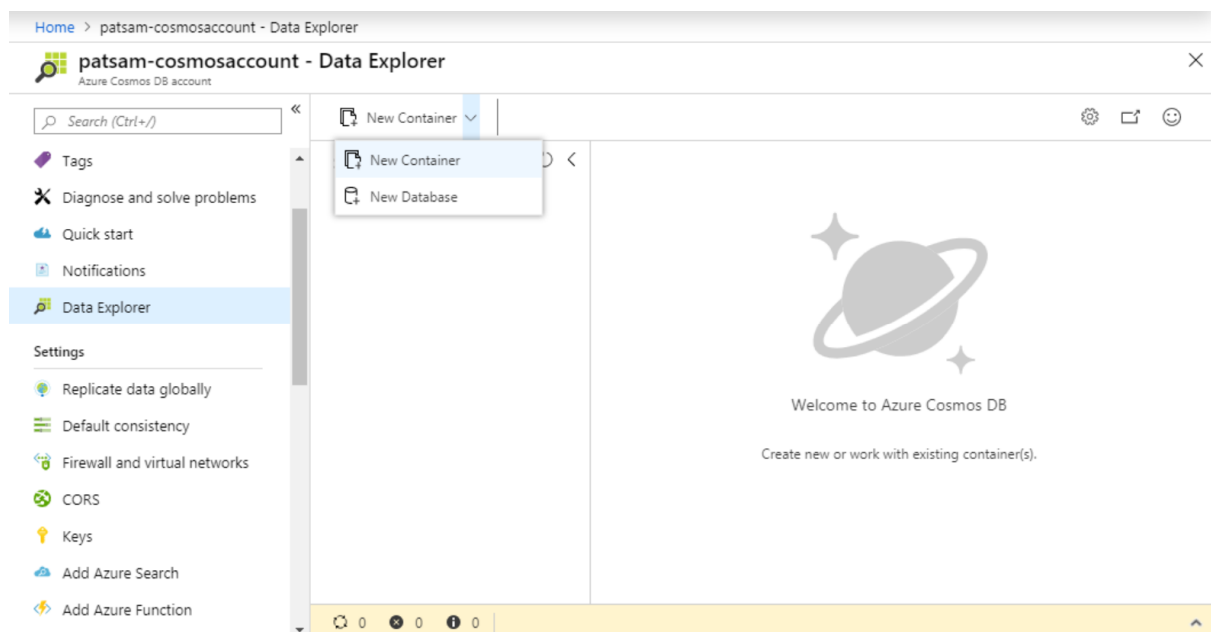
Once the deployment finishes, we can see the newly created Cosmos DB Account.



Azure Cosmos DB accounts			
Add Edit columns Refresh Export to CSV Assign tags Feedback			
Filter by name... Subscription == all Resource group == all Location == all Add filter			
Showing 1 to 1 of 1 records. < Previous Page 1 of 1 Next >			
NAME	STATUS	LOCATION	SUBSCRIPTION
 patsam-cosmosaccount	Online	Central US	Visual Studio Enterprise - MPN

Next, we need to add a Database. Click add New Database.


Data Explorer → New Database





Provide relevant details and Click OK.


- Database id: patsam-cosmosdb

New Database

 Start at \$24/mo per database, multiple containers included  
[More details](#)

\* Database id 

☒ Provision throughput 

\* Throughput (400 - 100,000 RU/s) 

+

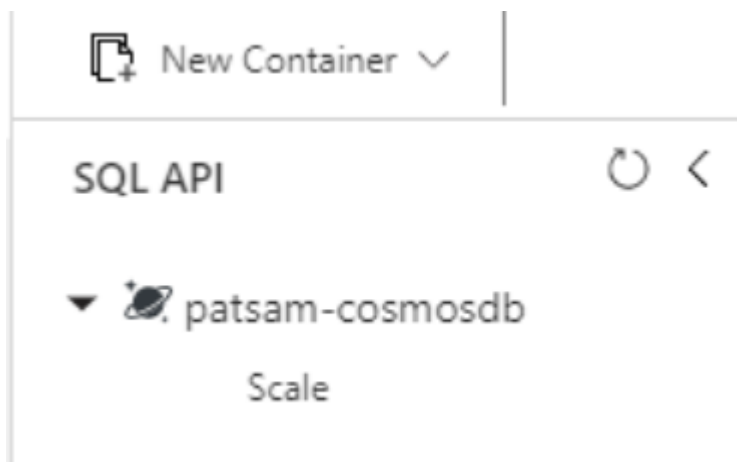
-

Estimated spend (USD): **\$0.032 hourly / \$0.77 daily** (1 region,  
1000RU/s - \$0.00008/RU/s)

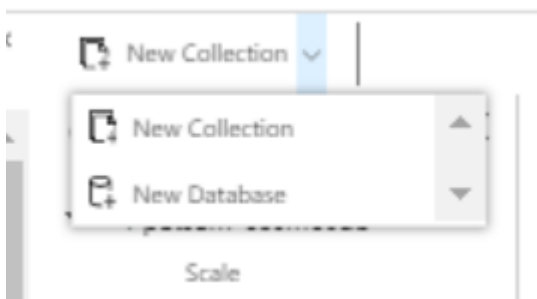
OK

Patrick Sameera Jayalath

We can see the newly created Database.



Next, we need to add a Container. Click on New Collection.



Provide relevant data and Click OK.

- Database id: patsam-cosmosdb (select the Database we created earlier)
- Collection id: Customers
- Shared key: \_id

Add Collection

Start at \$24/mo per database, multiple containers included

[More details](#)

Database id

Create new

Use existing

patsam-cosmosdb

Collection id

Customers

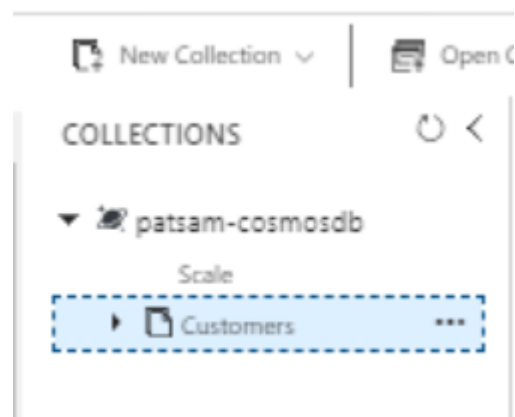
Shard key

\_id

☐ My shard key is larger than 100 bytes

☐ Provision dedicated throughput for this collection

We can see the newly created Collection.



Add another Collection – Sales.

- Collection id: Sales

Add Collection

Start at \$24/mo per database, multiple containers included  
[More details](#)

\*

Database id

Create new

Use existing

patsam-cosmosdb

\*

Collection id

Sales

\*

Shard key

\_id

OK

New Collection

Open C

COLLECTIONS

patsam-cosmosdb

Scale

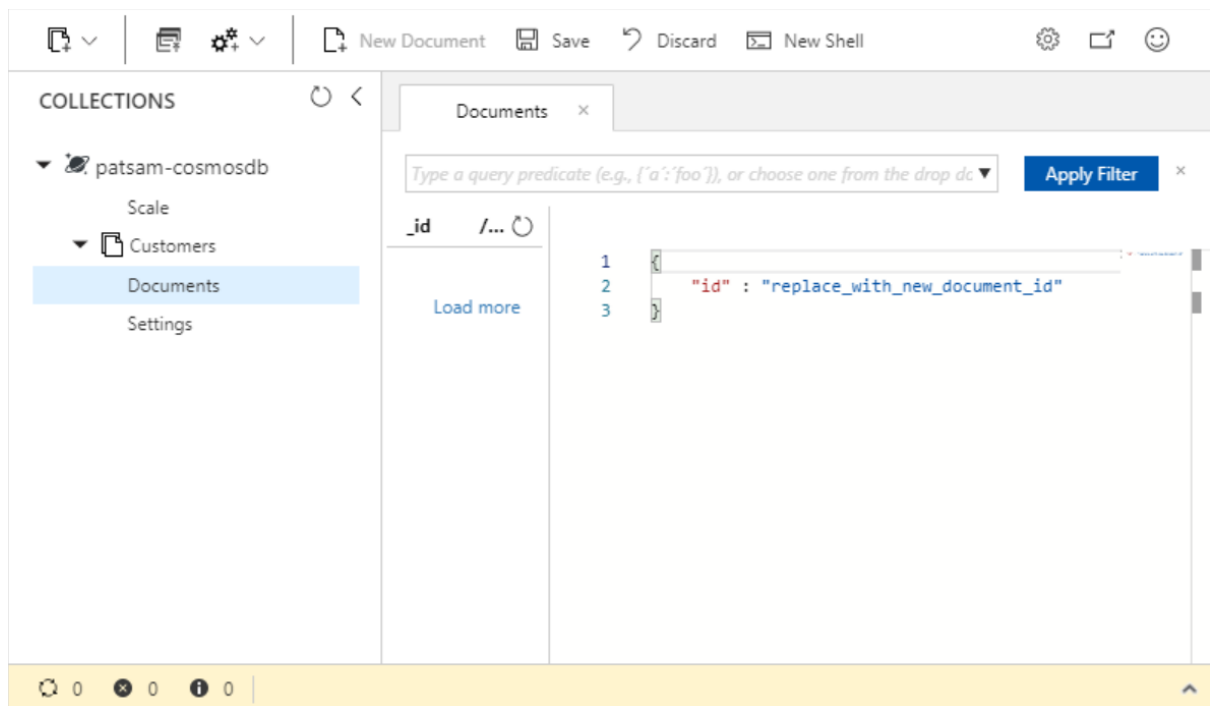
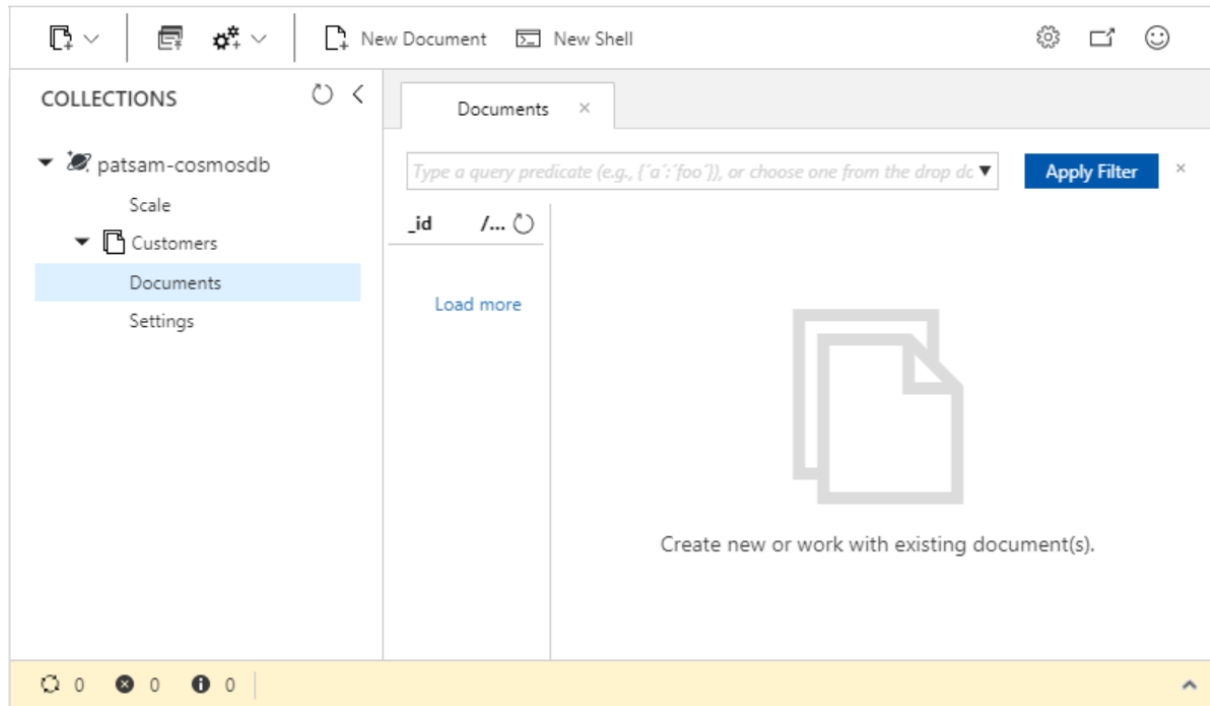
Customers

Sales

## **Step 02: Add data to the Azure Cosmos DB**

In Data Explorer, expand the patsam-cosmosdb Database, and expand the Customers Collection.

Next, select Documents, and then Click New Document.

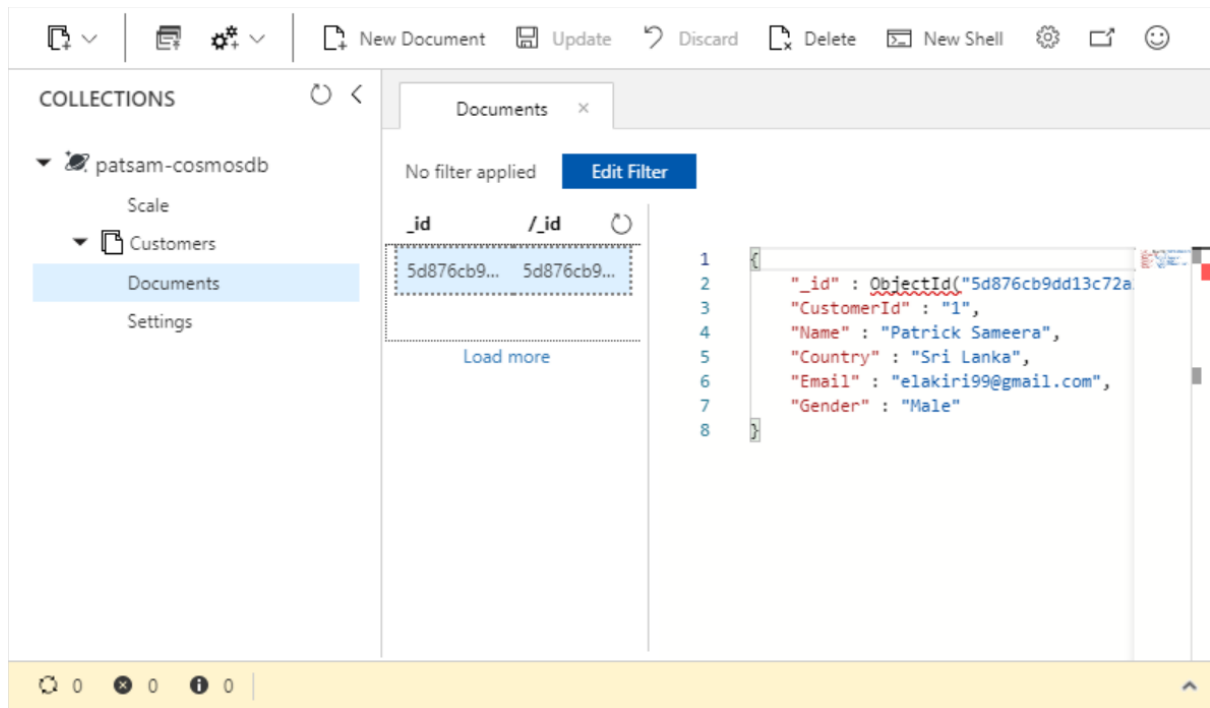




Add following document structure and Click Save.

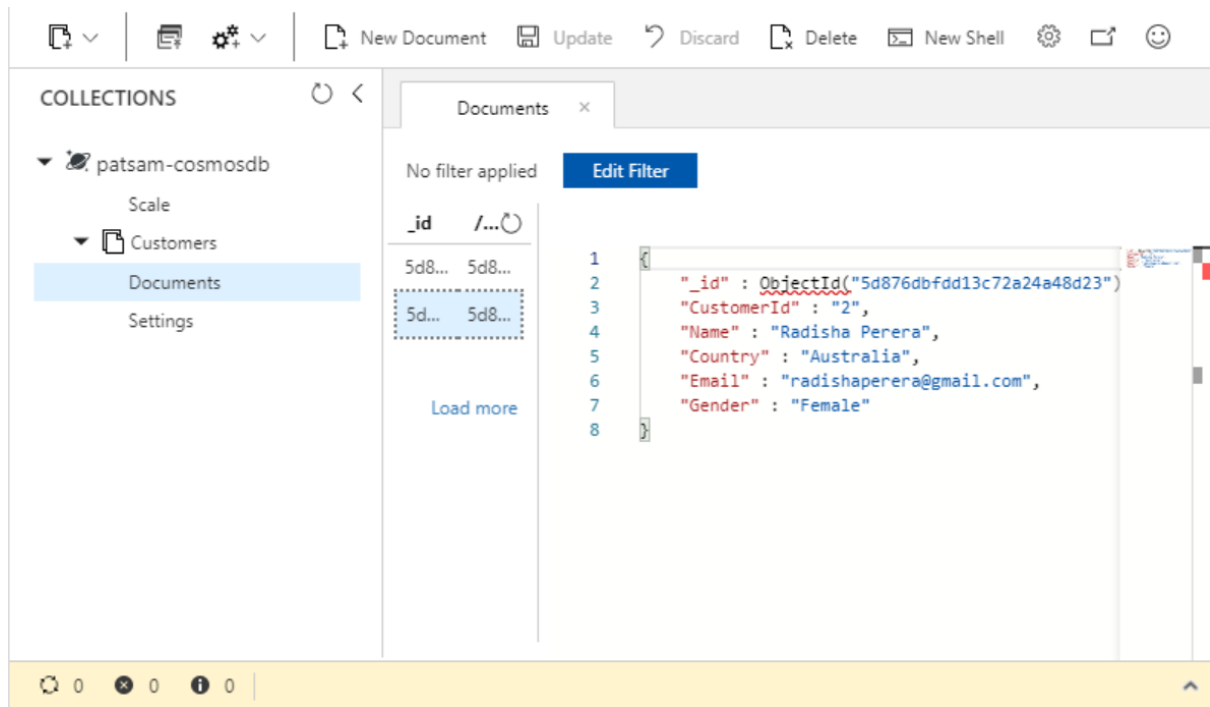
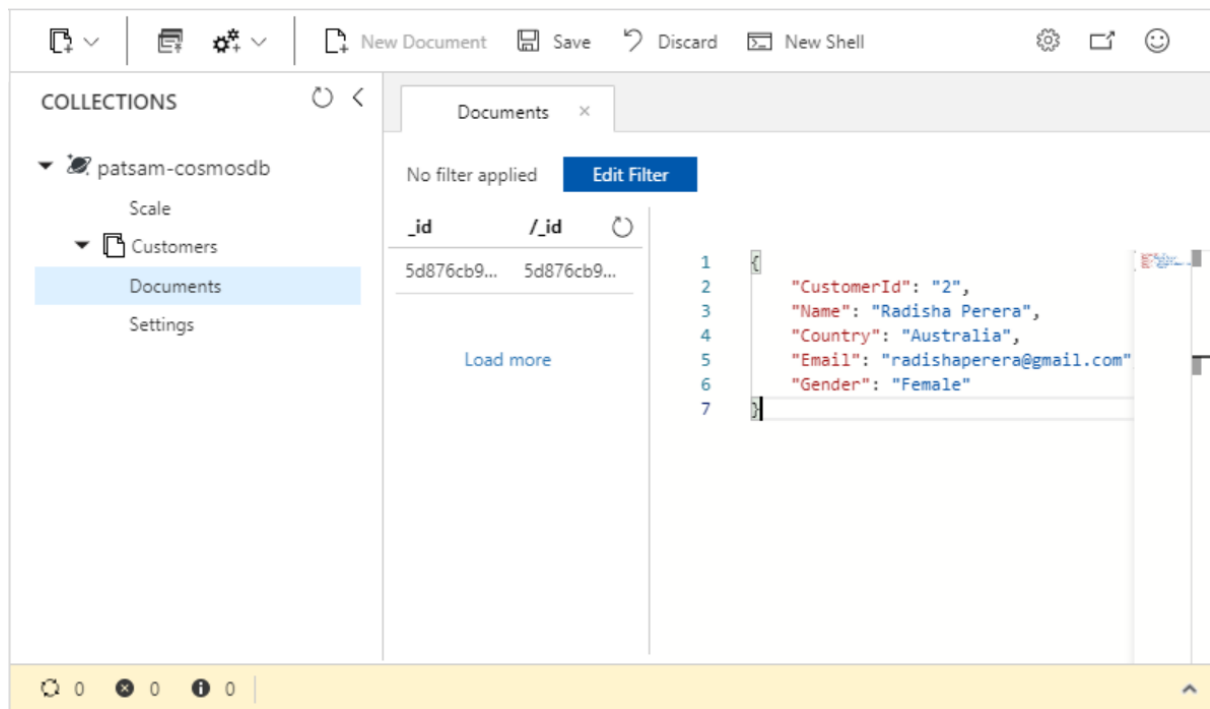
```
{
  "CustomerId": "1",
  "Name": "Patrick Sameera",
  "Country": "Sri Lanka",
  "Email": "elakiri99@gmail.com",
  "Gender": "Male"
}
```

Notice when Click Save it adds additional `_id` JSON property.



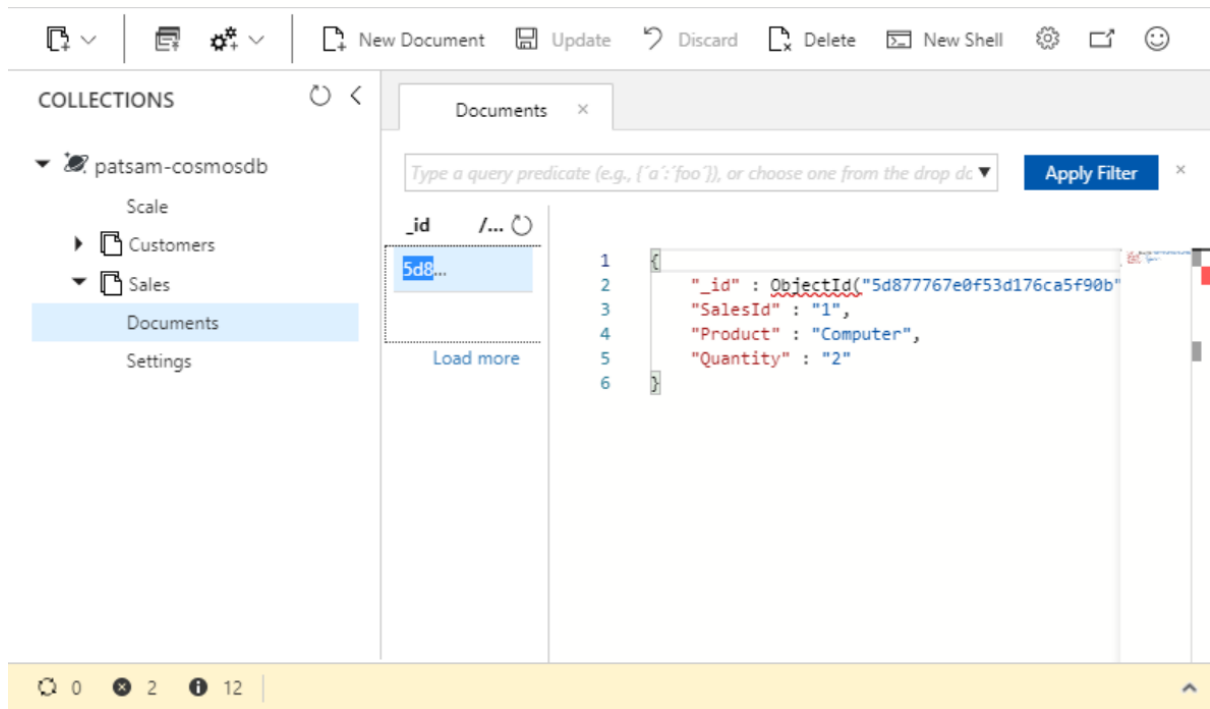
Add another document structure.

```
{  
  "CustomerId": "2",  
  "Name": "Radisha Perera",  
  "Country": "Australia",  
  "Email": "radishaperera@gmail.com",  
  "Gender": "Female"  
}
```



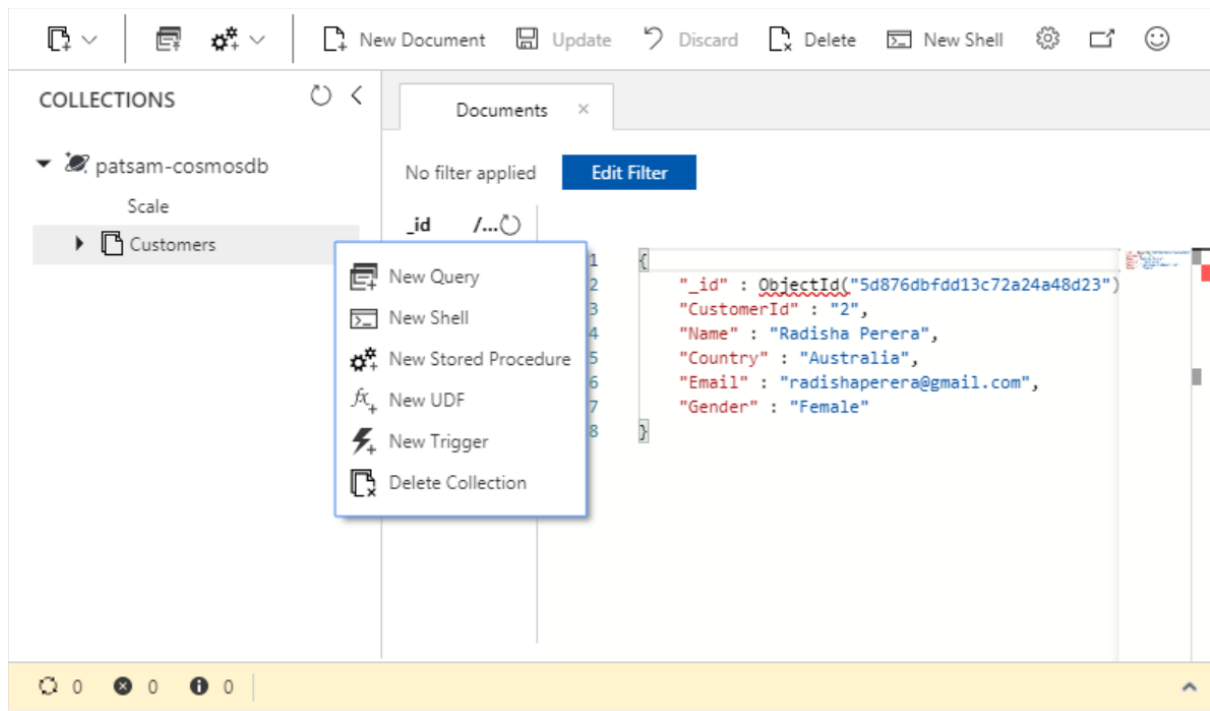
Add document structure for Sales.

```
{  
  "SalesId": "1",  
  "Product": "Computer",  
  "Quantity": "2"  
}
```



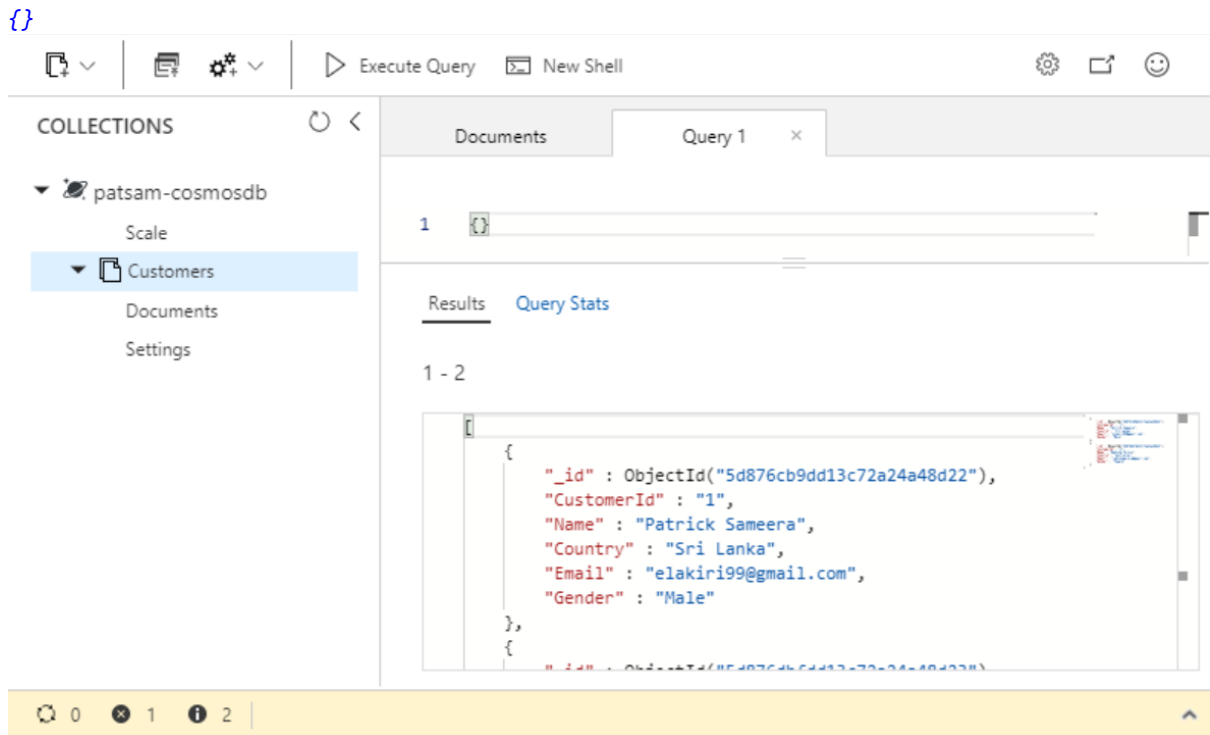
### Step 03: Query Azure Cosmos DB through Data Explorer

Click on "... " of the Customers Collection and you will get to see New Query option.



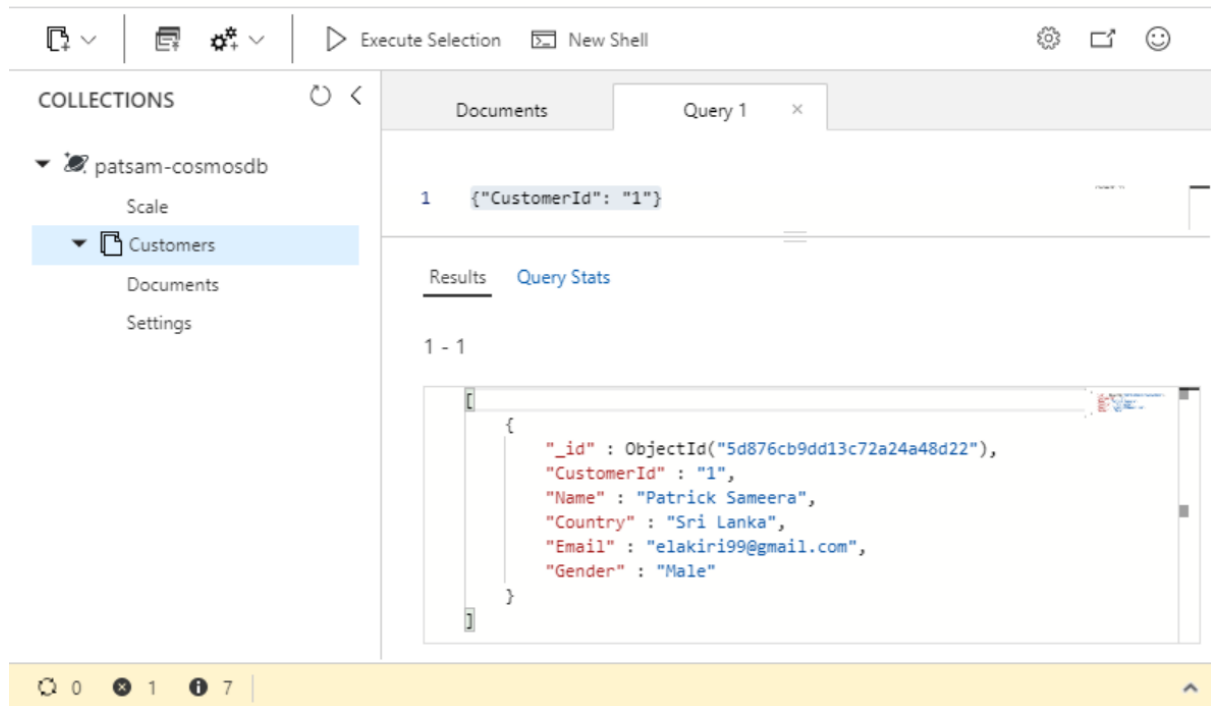
Click on New Query option.

Now we can run queries against the selected Collection.



Run bellow query.

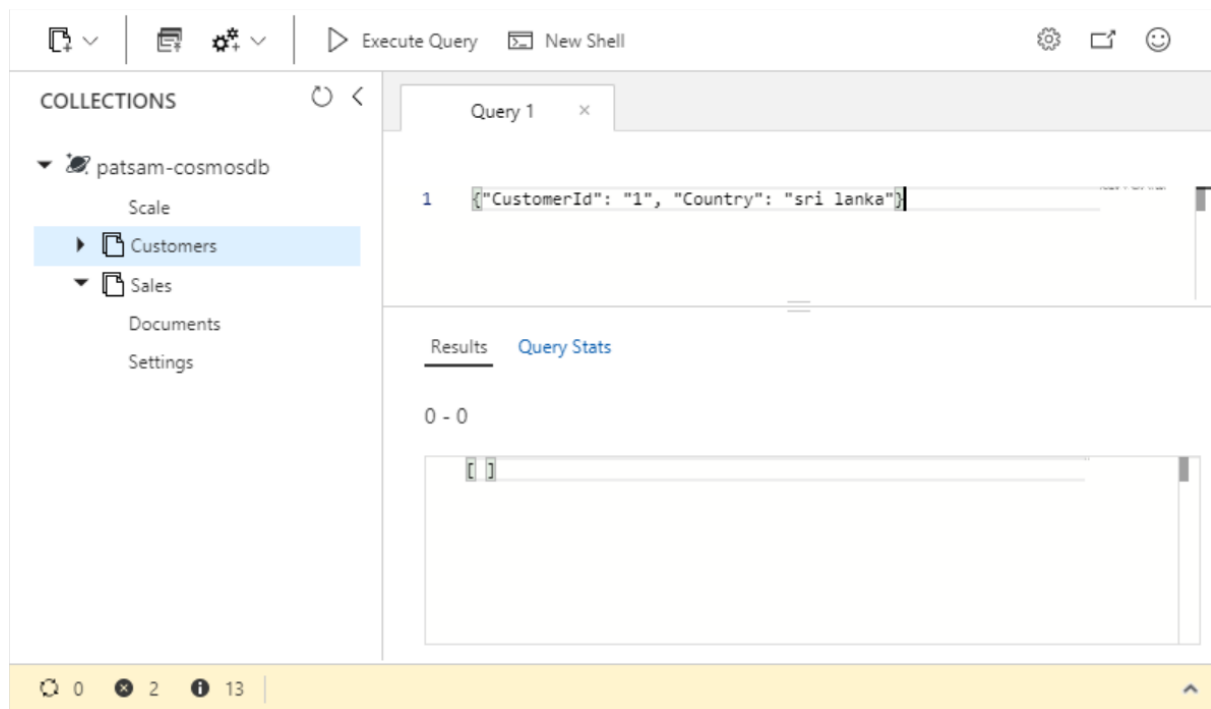
```
{"CustomerId": "1"}
```



Run bellow query.

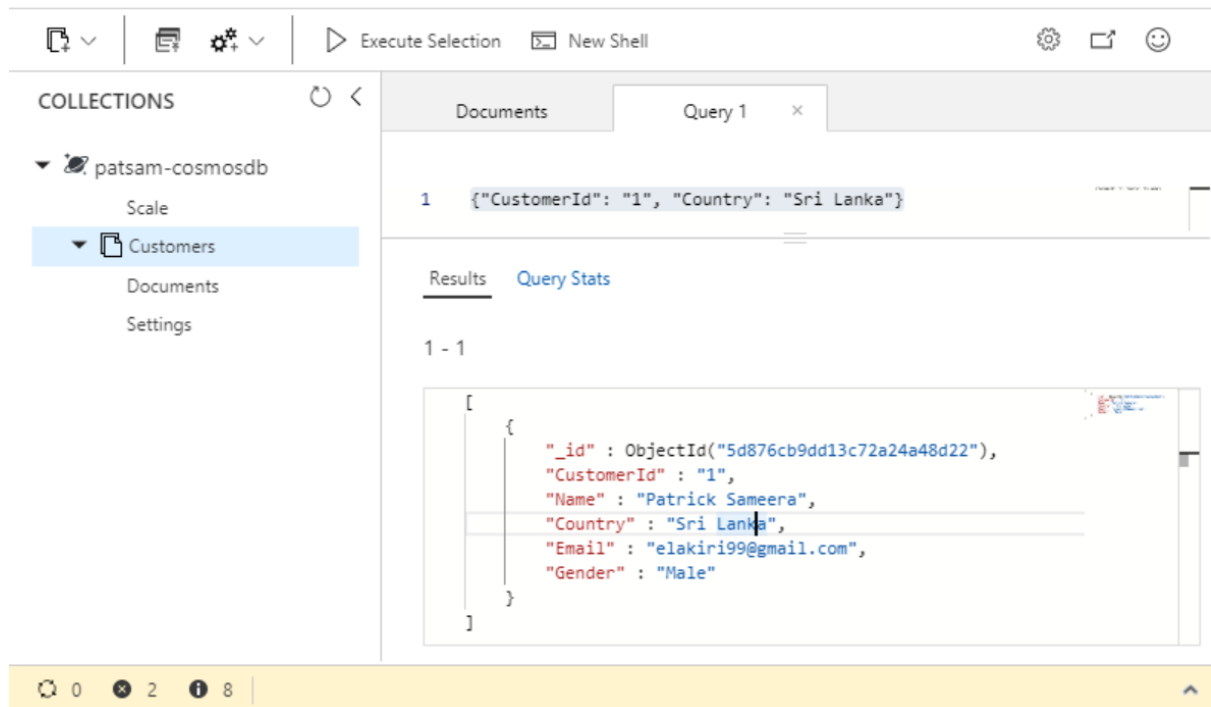
```
{"CustomerId": "1", "Country": "sri lanka"}
```

It doesn't return any data even though we have "Sri Lanka" as a Country. That is because it's case sensitive.



Run bellow query.

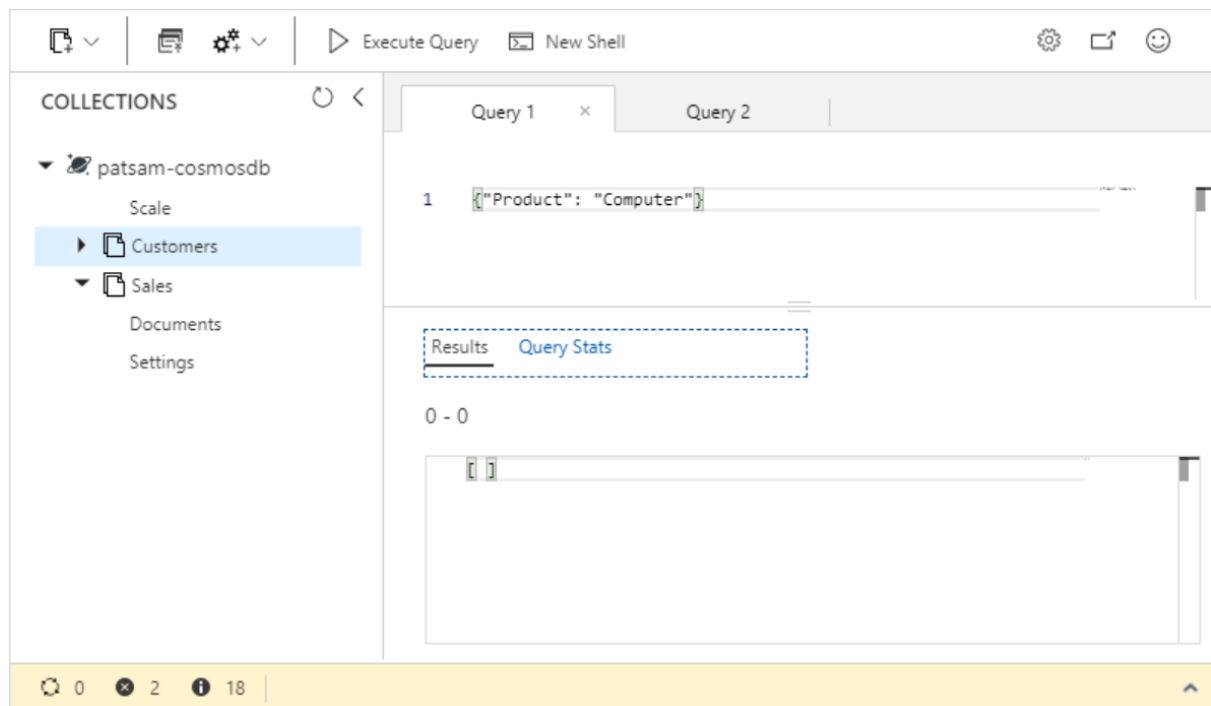
```
{"CustomerId": "1", "Country": "Sri Lanka"}
```



Run bellow query against Sales collection.

```
{"Product": "Computer"}
```

It won't return any data because we are running the query against the Customers Collection.



So, if want to run that query, we need to select Sales Collection.

`{"Product": "Computer"}`

The screenshot shows the CosmosDB Explorer interface. On the left, the 'COLLECTIONS' pane lists the database 'patsam-cosmosdb' with collections 'Scale', 'Customers', and 'Sales'. The 'Sales' collection is selected. The main area shows a query editor with the query `1 {"Product": "Computer"}`. Below the query, the 'Results' tab is active, displaying a single document: 

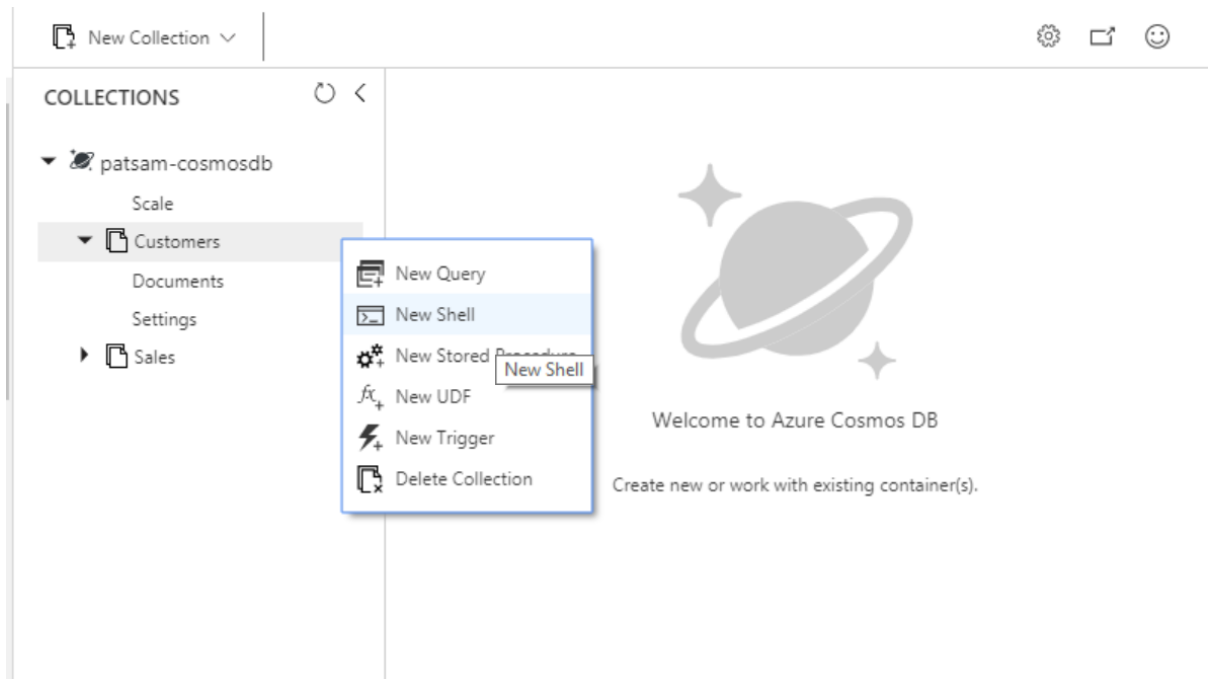
```
{
  "_id": ObjectId("5d877767e0f53d176ca5f90b"),
  "SalesId": "1",
  "Product": "Computer",
  "Quantity": "2"
}
```

. The status bar at the bottom indicates 'Successfully fetched 1 document for container Sales'.

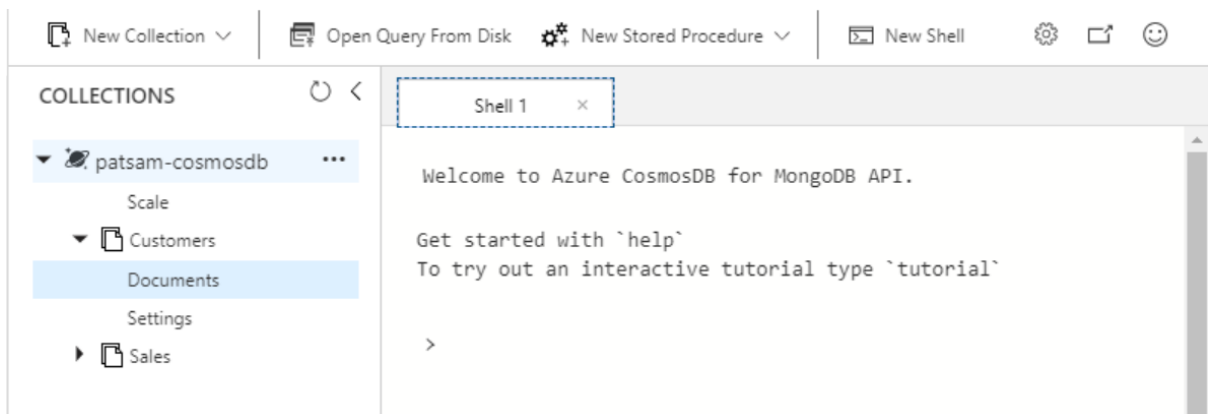
#### Step 04: Query Azure Cosmos DB through MongoDB Shell

We can use a Shell to run operations on the Database level.

Click on "..."/> of the Customers Collection and you will get to see New Shell option.



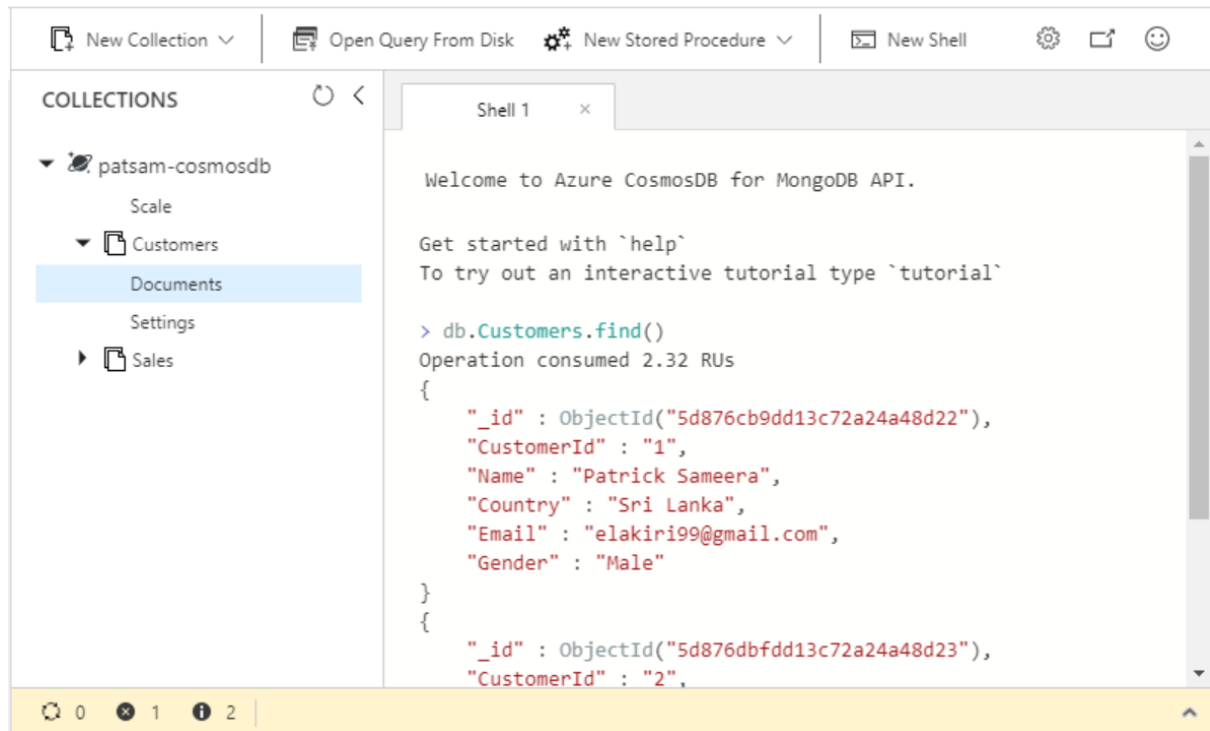
It will open a new Shell tab to run queries.





To run bellow query, paste it in the Shell and Click Enter.

`db.Customers.find()`



The screenshot shows the Azure CosmosDB Shell interface. On the left, the 'COLLECTIONS' pane shows a database named 'patsam-cosmosdb' with collections 'Scale', 'Customers', 'Documents', 'Settings', and 'Sales'. The 'Customers' collection is selected. The main shell area shows the following text:

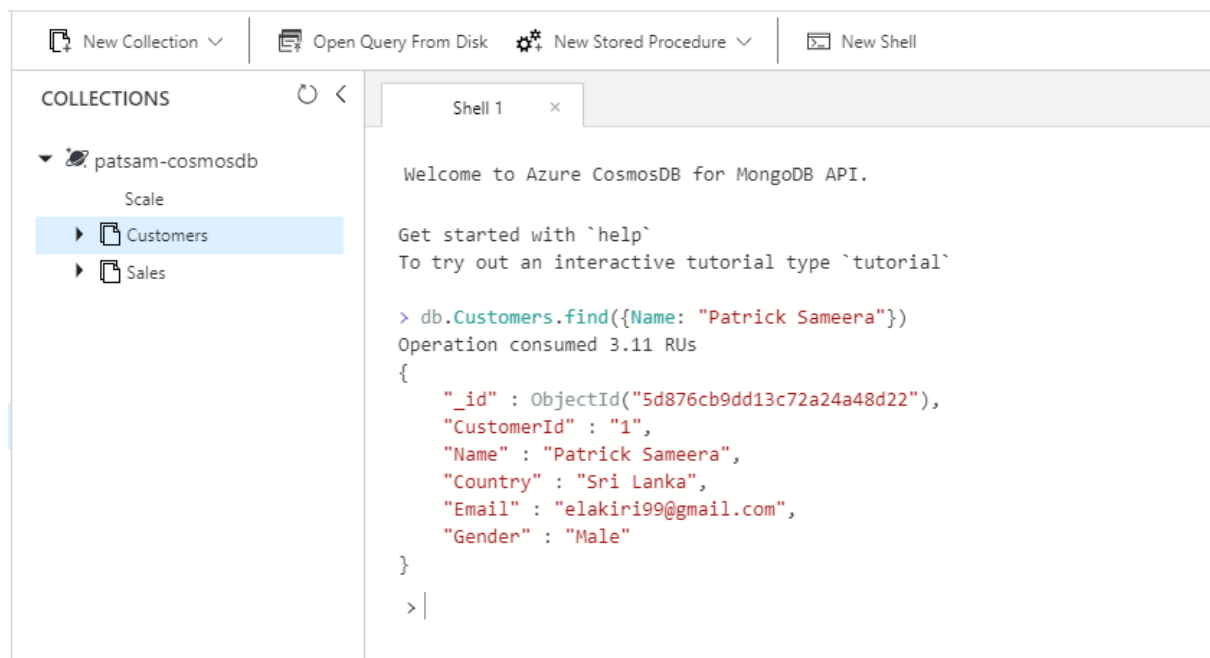
```
Welcome to Azure CosmosDB for MongoDB API.

Get started with `help`
To try out an interactive tutorial type `tutorial`

> db.Customers.find()
Operation consumed 2.32 RUs
{
  "_id" : ObjectId("5d876cb9dd13c72a24a48d22"),
  "CustomerId" : "1",
  "Name" : "Patrick Sameera",
  "Country" : "Sri Lanka",
  "Email" : "elakiri99@gmail.com",
  "Gender" : "Male"
}
{
  "_id" : ObjectId("5d876dbfdd13c72a24a48d23"),
  "CustomerId" : "2",
```

Run bellow query.

`db.Customers.find({Name: "Patrick Sameera"})`



The screenshot shows the Azure CosmosDB Shell interface. On the left, the 'COLLECTIONS' pane shows a database named 'patsam-cosmosdb' with collections 'Scale', 'Customers', and 'Sales'. The 'Customers' collection is selected. The main shell area shows the following text:

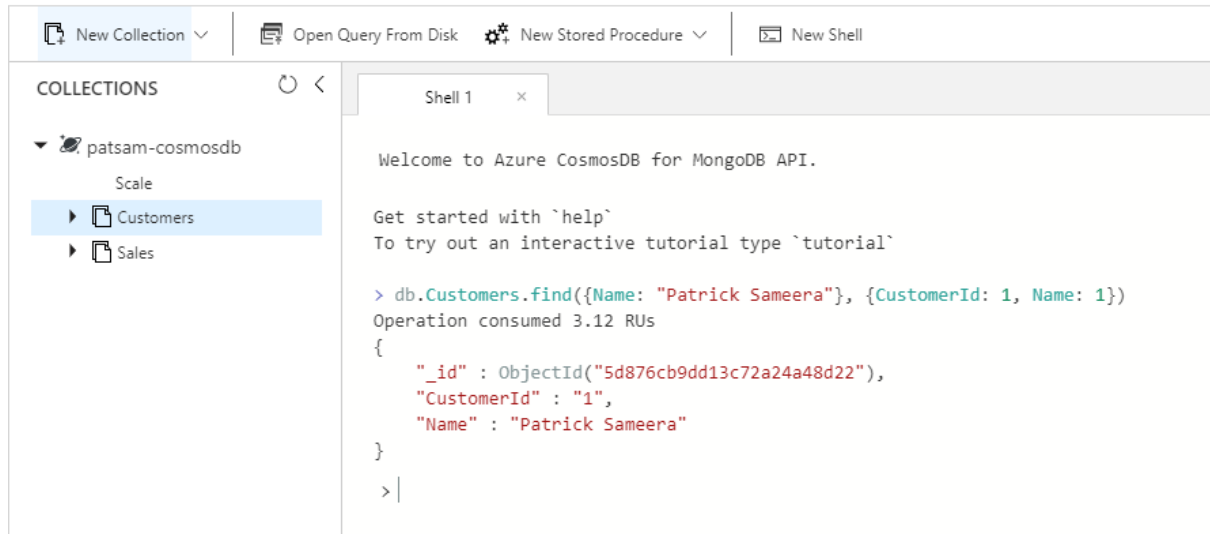
```
Welcome to Azure CosmosDB for MongoDB API.

Get started with `help`
To try out an interactive tutorial type `tutorial`

> db.Customers.find({Name: "Patrick Sameera"})
Operation consumed 3.11 RUs
{
  "_id" : ObjectId("5d876cb9dd13c72a24a48d22"),
  "CustomerId" : "1",
  "Name" : "Patrick Sameera",
  "Country" : "Sri Lanka",
  "Email" : "elakiri99@gmail.com",
  "Gender" : "Male"
}
> |
```

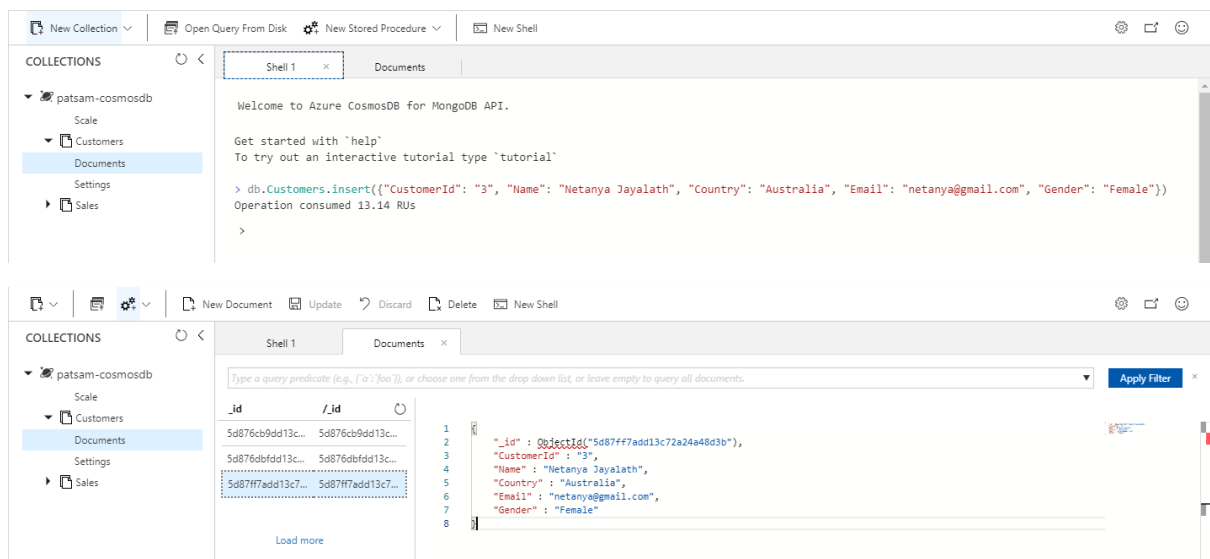
Run bellow query.

```
db.Customers.find({Name: "Patrick Sameera"}, {CustomerId: 1, Name: 1})
```



Run bellow query.

```
db.Customers.insert({"CustomerId": "3", "Name": "Netanya Jayalath", "Country": "Australia", "Email": netanya@gmail.com, "Gender": "Female"})
```



If you want to learn about more query techniques on MongoDB SQL, have a look bellow URLs:

<https://docs.mongodb.com/manual/reference/sql-comparison/>

<https://www.sqlshack.com/getting-started-with-azure-cosmos-db-and-mongodb-api/>