Name: Aditi Das **IT497**
Student ID: 202001259 **Date:** 31st March 2023

---

1. Complete NoSQL design for following problems:

1a. A gift shop wants to start a portal which allows users to view various items present in its store. The items details like name, price, available quantity etc. are shown. Along with that, pictures of item from different angles are also available. There's a high-res image also available for every item. How will you design a data solution for this?

Ans. I will undertake the following steps to design a data solution in NoSQL for this:

i. I will store the high-resolution images in an Azure Storage Blob and obtain its URL endpoint.

ii. The first table (called "**All-Items**") that will be created will have **"Category Type"** as the **Partition Key**. This would make the Partition Key both a **group key** and a **balanced key**. The **Row Key** for the same will contain the unique product id for the given Partition Key. For e.g. a valid Partition Key would be "Clothes" and a valid Row Key for the it would be "Shirt-1". Other valid keys sets might be {"Clothes","Shirt-2"} and {"Toys","Doll-1"}.

iii. "All-Items" would store properties in a **Key-Value** form including the URL endpoint for the high-resolution image. It would also contain properties such as name, price, available quantity (SKU) etc. This would prevent looping in case the user wanted to view the image in higher resolution.

iv. The other tables in this data solution would hugely depend upon what the client actually wants. For e.g., if a client (gift shop owner/manager) wants to be notified whenever a product is running out of stock (let us assume that for said client, this holds true when the quantity of the product in question is less than 4), we could design a table called "**Restock-Urgently**".

v. "Restock-Urgently" would have the **SKU** of an item as its **Partition Key** thus preventing any kind of looping. The **Row Key** for this table would be the "**Partition Key + Row Key**" of "**All-Items**". Data will automatically get added to this when SKU for a product goes below 4. Similarly, data will automatically get removed too if SKU for an item goes above 3.

vi. Having such a table prevents us from running into an $O(n^2)$ type of complexity wherein one has to iterate through all the Partition Keys and then within them, the Row Key for "All-Items".

1b. A company has training period for employees. It has around 500 training modules that an employee needs to cover over the period of 6 months. A module could be a video, document, e-book or other form or content. The requirement is to track the status of modules covered by employee over the training period.
How will you design a data base solution to for this?

Ans. My database solution for the same:

i. The first thing to notice here is that our main aim here is to track the progress of the individual employees.

ii. We could use Azure Blob Storage to store the URL endpoints of these training modules.

iii. Let these modules be identified as M-1, M-2,…,M-500 and the employees be identified as E-1, E-2,…, E-N.

iv. The first table that would be designed (called "**Module-Information**") would contain the **ids** of these modules (M-1,M-2,..M-500) as the **Partition Key** and we could place any random value as the **Row Key** (say **1**) since there are only 500 modules (hence, we are letting go the condition that Partition Key needs to be group key and our Partition Key here is perfectly balanced) and not much variation can be expected from this kind of request. The URL endpoints will be stored alongside these Partition Keys.

v. Not necessary for the problem statement mentioned here, but we can design another table called "**Employee-Details**" where the **Partition Key** could be the **Division** the employee works in and the **Row Key** could be the **Employee-ID**.

vi. Now, coming to the actual request of the client, which is tracking an Employee's progress, we will design a table called "**Employee-Training-Status**".

vii. We will have the **Employee-ID** as the **Partition Key** and the **Module-ID** as the **Row Key**. We could have had it the other way around too, but then tracking a single employee's progress would have resulted in an $O(n^2)$ type of complexity.

viii. The progress of each module would be stored in a property called "**Progress**" and an internal API would convert the duration of the video watched/number of pages read to a percent after obtaining the data from the Row Key's URL endpoint by obtaining the value from Module-Information (as it is the Partition Key there).

2. Go through the following tutorials:

2a. Intro to Azure Cosmos DB:

Ans. Here are a few things I found noteworthy about Azure Cosmos DB based on the tutorial and other topics covered during the lectures:

i. **Global distribution**: Azure Cosmos DB is a globally distributed database service that allows you to easily store and manage data across multiple regions around the world. This feature ensures low latency access to data and high availability of your application, regardless of the geographic location of your users.

ii. **Multi-model database**: Azure Cosmos DB supports multiple data models such as document, key-value, graph, and column-family, which provides a flexible data model that can accommodate different types of data.

iii. **Scalability**: Azure Cosmos DB is designed for horizontal scaling, allowing you to increase your database capacity and throughput as your application grows, without any downtime or performance degradation.

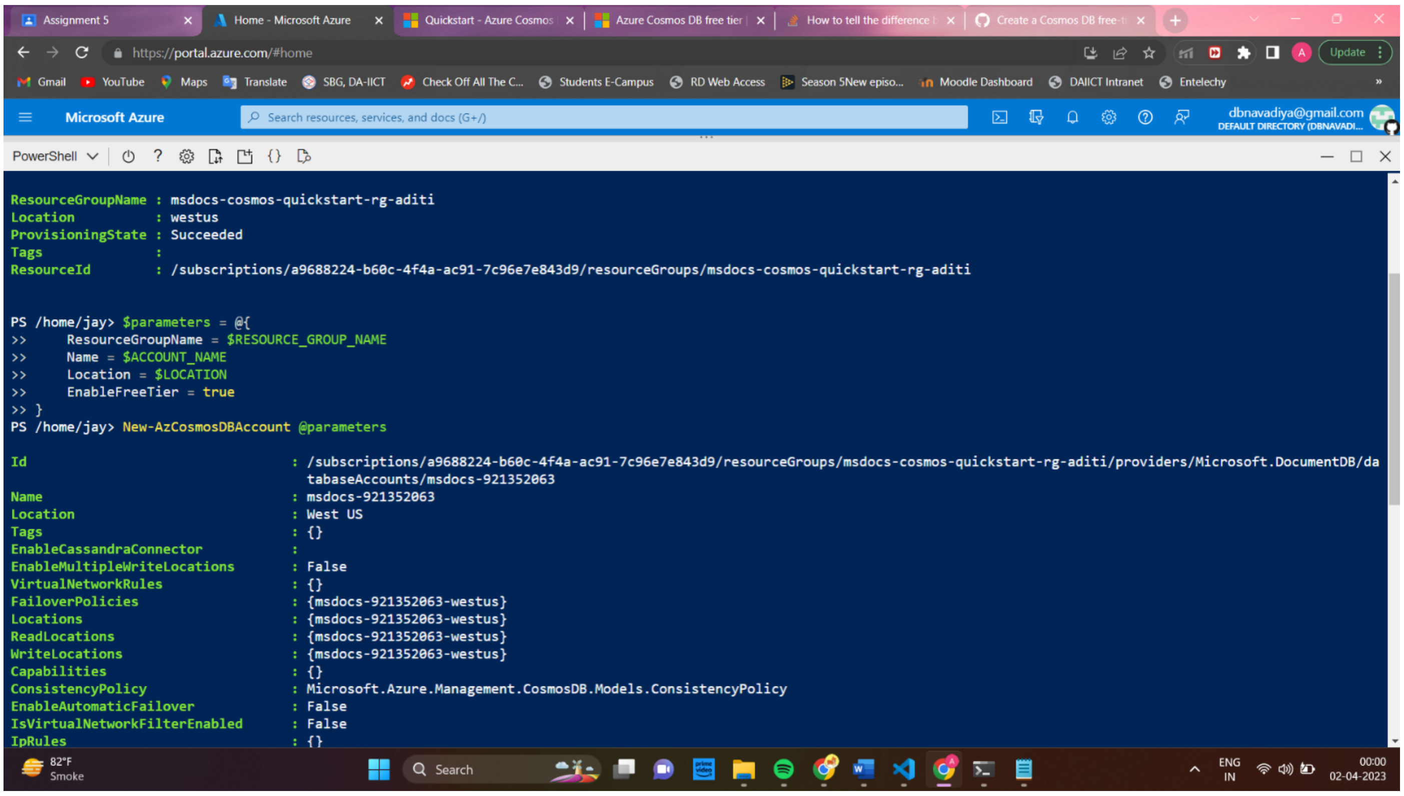
iv. **Consistency**: Azure Cosmos DB offers a choice of five well-defined consistency levels to provide fine-grained control over the trade-offs between consistency, availability, and latency.

v. **Security**: Azure Cosmos DB provides a high level of security through features such as encryption at rest and in transit, role-based access control, and network isolation.

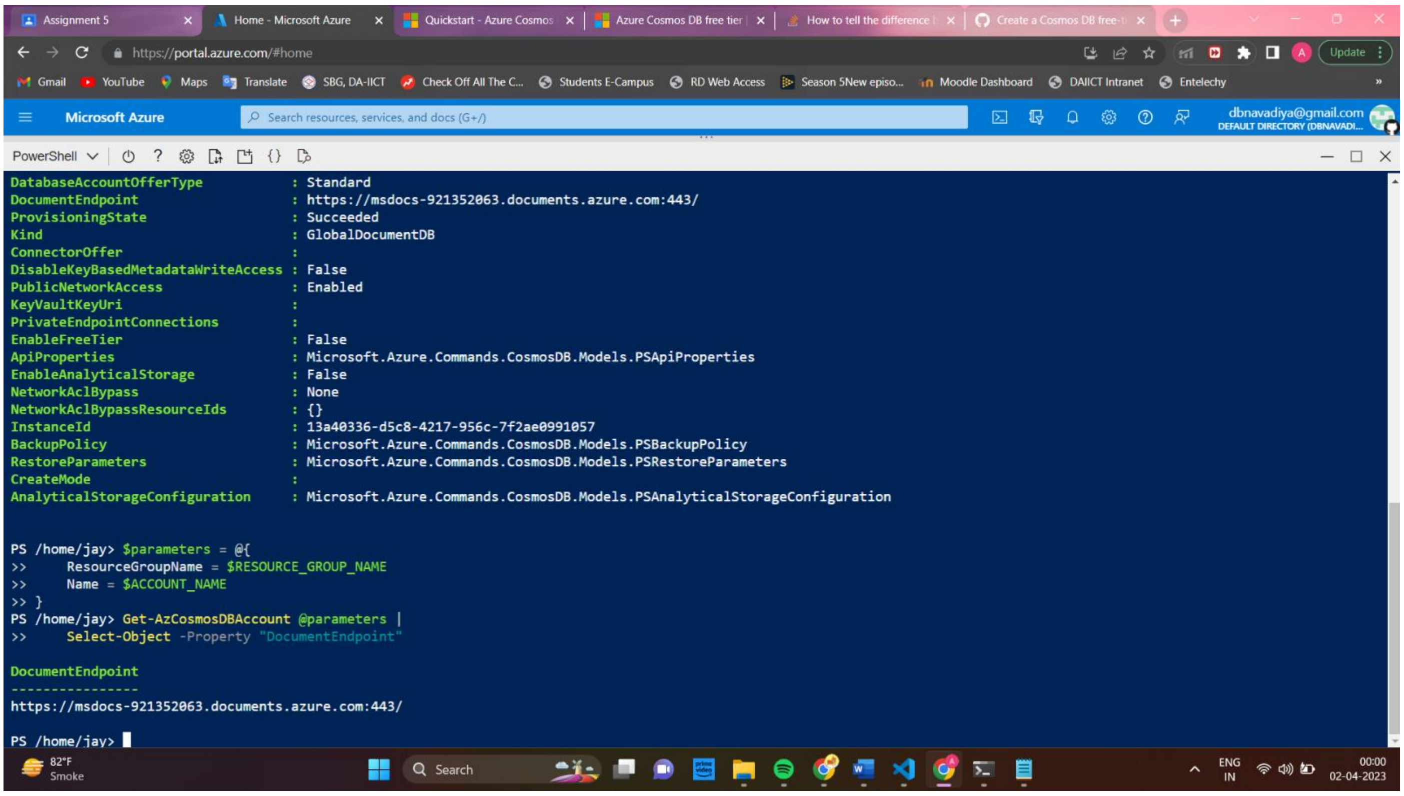
vi. **Integration** with other Azure services: Azure Cosmos DB integrates with other Azure services, such as Azure Functions, Azure Stream Analytics, and Azure Search, making it easy to build modern, cloud-native applications.

2b. Azure Cosmos DB for NoSQL client library for Python:

i. Initialising:

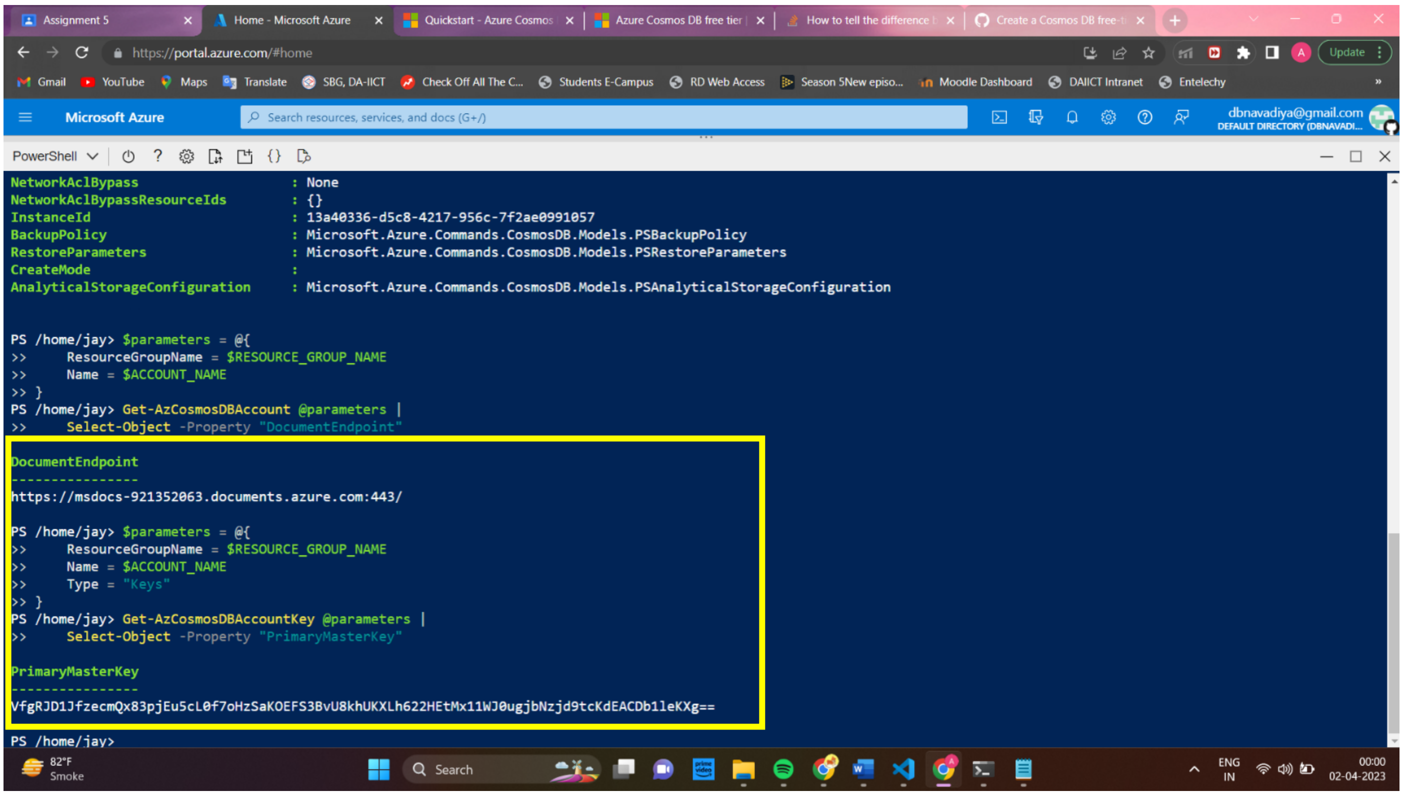


```
ResourceGroupName : msdocs-cosmos-quickstart-rg-aditi
Location          : westus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/a9688224-b60c-4f4a-ac91-7c96e7e843d9/resourceGroups/msdocs-cosmos-quickstart-rg-aditi


PS /home/jay> $parameters = @{
>>     ResourceGroupName = $RESOURCE_GROUP_NAME
>>     Name = $ACCOUNT_NAME
>>     Location = $LOCATION
>>     EnableFreeTier = true
>> }
PS /home/jay> New-AzCosmosDBAccount @parameters

Id                                   : /subscriptions/a9688224-b60c-4f4a-ac91-7c96e7e843d9/resourceGroups/msdocs-cosmos-quickstart-rg-aditi/providers/Microsoft.DocumentDB/da
                                       tabaseAccounts/msdocs-921352063
Name                                 : msdocs-921352063
Location                             : West US
Tags                                 : {}
EnableCassandraConnector             :
EnableMultipleWriteLocations         : False
VirtualNetworkRules                  : {}
FailoverPolicies                     : {msdocs-921352063-westus}
Locations                            : {msdocs-921352063-westus}
ReadLocations                        : {msdocs-921352063-westus}
WriteLocations                       : {msdocs-921352063-westus}
Capabilities                         : {}
ConsistencyPolicy                    : Microsoft.Azure.Management.CosmosDB.Models.ConsistencyPolicy
EnableAutomaticFailover              : False
IsVirtualNetworkFilterEnabled        : False
IpRules                              : {}
```



```
DatabaseAccountOfferType             : Standard
DocumentEndpoint                     : https://msdocs-921352063.documents.azure.com:443/
ProvisioningState                    : Succeeded
Kind                                 : GlobalDocumentDB
ConnectorOffer                       :
DisableKeyBasedMetadataWriteAccess   : False
PublicNetworkAccess                  : Enabled
KeyVaultKeyUri                       :
PrivateEndpointConnections           :
EnableFreeTier                       : False
ApiProperties                        : Microsoft.Azure.Commands.CosmosDB.Models.PSApiProperties
EnableAnalyticalStorage              : False
NetworkAclBypass                     : None
NetworkAclBypassResourceIds          : {}
InstanceId                           : 13a40336-d5c8-4217-956c-7f2ae0991057
BackupPolicy                         : Microsoft.Azure.Commands.CosmosDB.Models.PSBackupPolicy
RestoreParameters                    : Microsoft.Azure.Commands.CosmosDB.Models.PSRestoreParameters
CreateMode                           :
AnalyticalStorageConfiguration       : Microsoft.Azure.Commands.CosmosDB.Models.PSAnalyticalStorageConfiguration


PS /home/jay> $parameters = @{
>>     ResourceGroupName = $RESOURCE_GROUP_NAME
>>     Name = $ACCOUNT_NAME
>> }
PS /home/jay> Get-AzCosmosDBAccount @parameters |
>>     Select-Object -Property "DocumentEndpoint"

DocumentEndpoint
----------------
https://msdocs-921352063.documents.azure.com:443/

PS /home/jay>
```

ii. Getting URL endpoints:



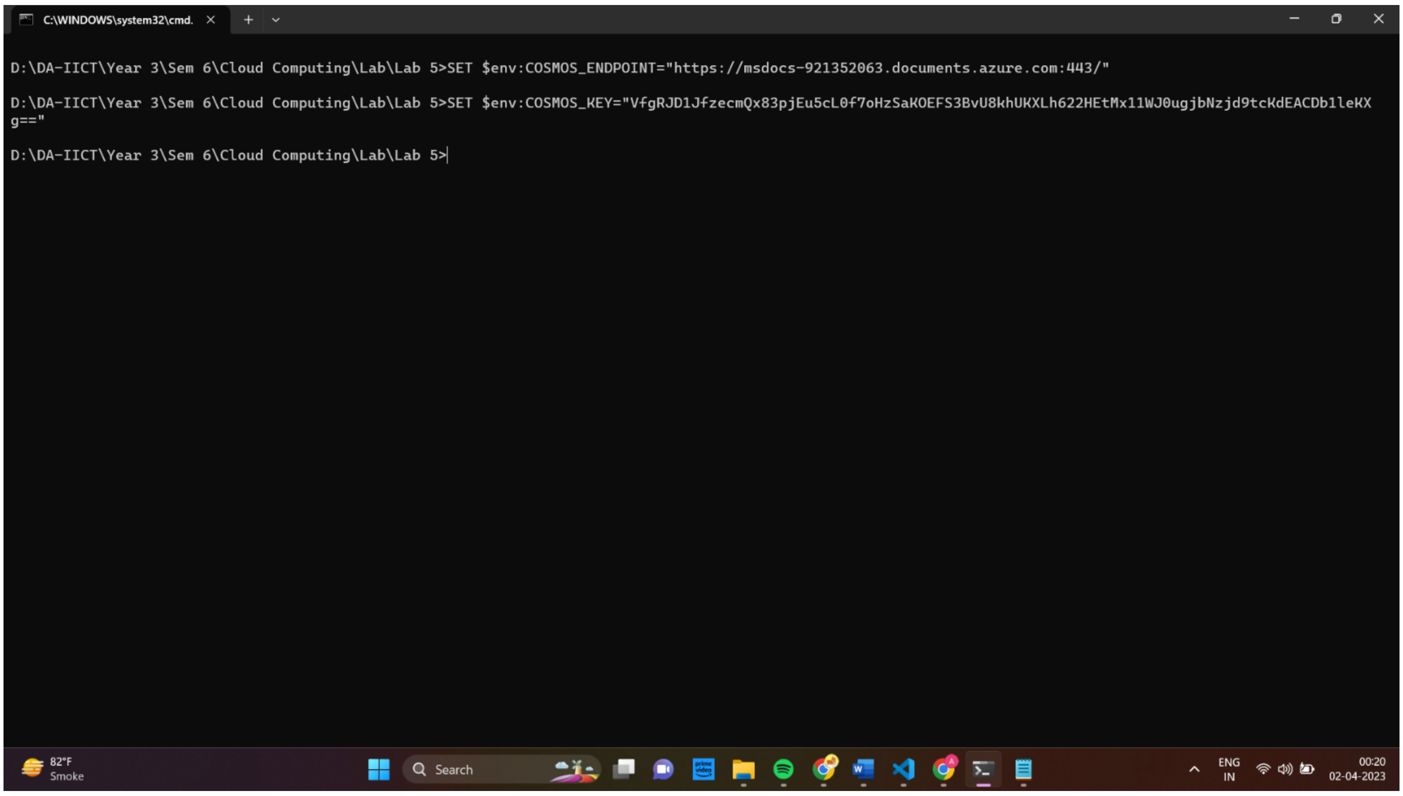
```
NetworkAclBypass                    : None
NetworkAclBypassResourceIds         : {}
InstanceId                          : 13a40336-d5c8-4217-956c-7f2ae0991057
BackupPolicy                        : Microsoft.Azure.Commands.CosmosDB.Models.PSBackupPolicy
RestoreParameters                   : Microsoft.Azure.Commands.CosmosDB.Models.PSRestoreParameters
CreateMode                          :
AnalyticalStorageConfiguration      : Microsoft.Azure.Commands.CosmosDB.Models.PSAnalyticalStorageConfiguration


PS /home/jay> $parameters = @{
>>     ResourceGroupName = $RESOURCE_GROUP_NAME
>>     Name = $ACCOUNT_NAME
>> }
PS /home/jay> Get-AzCosmosDBAccount @parameters |
>>     Select-Object -Property "DocumentEndpoint"

DocumentEndpoint
----------------
https://msdocs-921352063.documents.azure.com:443/

PS /home/jay> $parameters = @{
>>     ResourceGroupName = $RESOURCE_GROUP_NAME
>>     Name = $ACCOUNT_NAME
>>     Type = "Keys"
>> }
PS /home/jay> Get-AzCosmosDBAccountKey @parameters |
>>     Select-Object -Property "PrimaryMasterKey"

PrimaryMasterKey
----------------
VfgRJD1JfzecmQx83pjEu5cL0f7oHzSaKOEFS3BvU8khUKXLh622HEtMx11WJ0ugjbNzjd9tcKdEACDb1leKXg==

PS /home/jay>
```

Document Endpoint :
https://msdocs-921352063.documents.azure.com:443/

Primary Master Key :
VfgRJD1JfzecmQx83pjEu5cL0f7oHzSaKOEFS3BvU8khUKXLh622HEtMx11WJ0ugjbNzjd9tcKdEACDb1leKXg==

iii. Working with the Database:

- After this, I set the two environment variables COSMOS_KEY and COSMOS_ENDPOINT as follows:



```
D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>SET $env:COSMOS_ENDPOINT="https://msdocs-921352063.documents.azure.com:443/"

D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>SET $env:COSMOS_KEY="VfgRJD1JfzecmQx83pjEu5cL0f7oHzSaKOEFS3BvU8khUKXLh622HEtMx11WJ0ugjbNzjd9tcKdEACDb1leKX
g=="

D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>
```

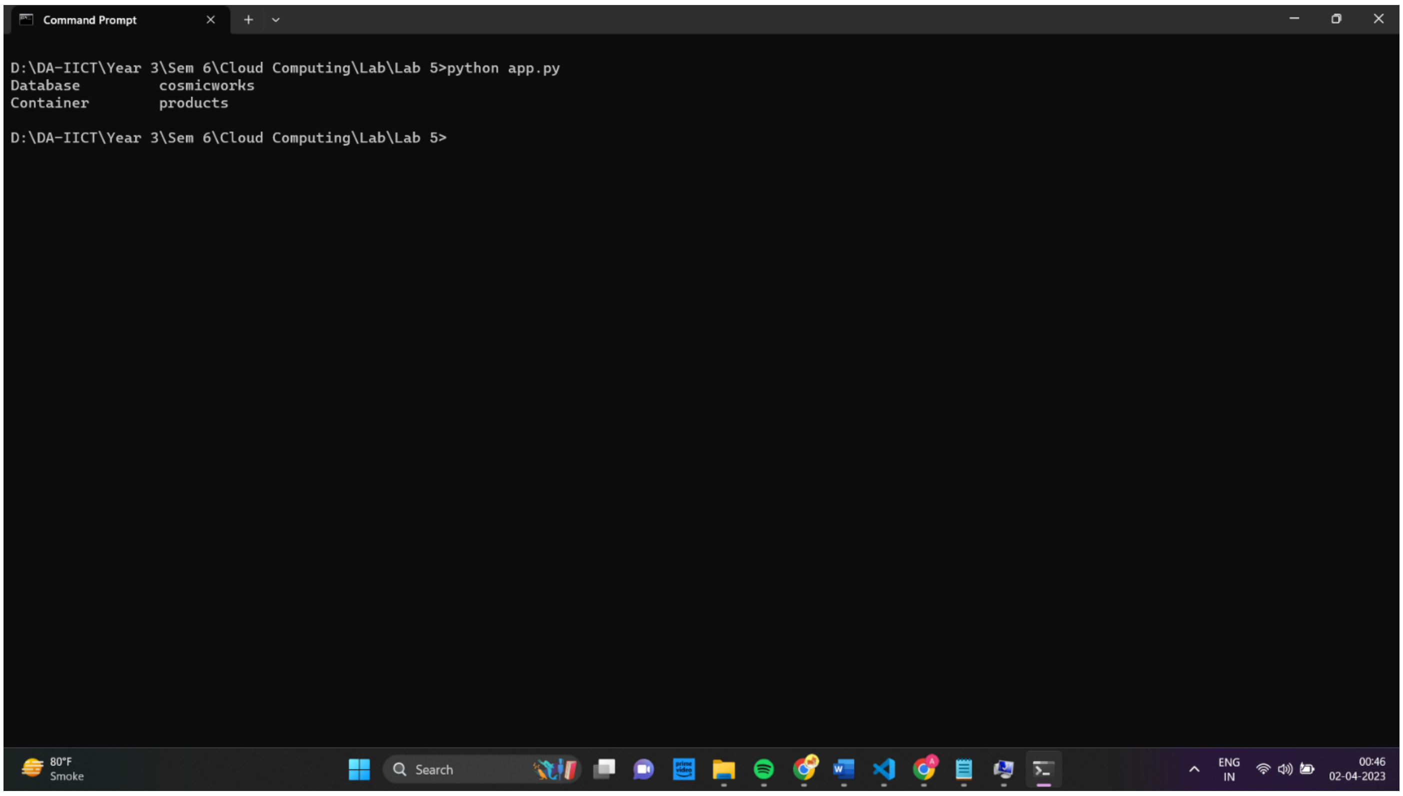
- I then created a client using Python using the value of these environment variables as follows:

```
rom azure.cosmos import CosmosClient, PartitionKey

# </imports>

# <environment_variables>
ENDPOINT = os.environ["COSMOS_ENDPOINT"]
KEY = os.environ["COSMOS_KEY"]
# </environment_variables>

# <constants>
DATABASE_NAME = "cosmicworks"
CONTAINER_NAME = "products"
# </constants>

# <create_client>
client = CosmosClient(url=ENDPOINT, credential=KEY)
# </create_client>
```

- I then created a database and a container using the following lines of code:

```
# <create_database>
database = client.create_database_if_not_exists(id=DATABASE_NAME)
print("Database\t", database.id)
# </create_database>

# <create_partition_key>
key_path = PartitionKey(path="/categoryId")
# </create_partition_key>

# <create_container>
container = database.create_container_if_not_exists(
    id=CONTAINER_NAME, partition_key=key_path, offer_throughput=400
)
print("Container\t", container.id)
# </create_container>
```

The output for the same is as follows:



```
D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>python app.py
Database        cosmicworks
Container       products

D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>
```

- After that I inserted 6 items across 2 categories with partition keys as 24042002-aditiDas-Category1 and 24042002-aditiDas-Category2.

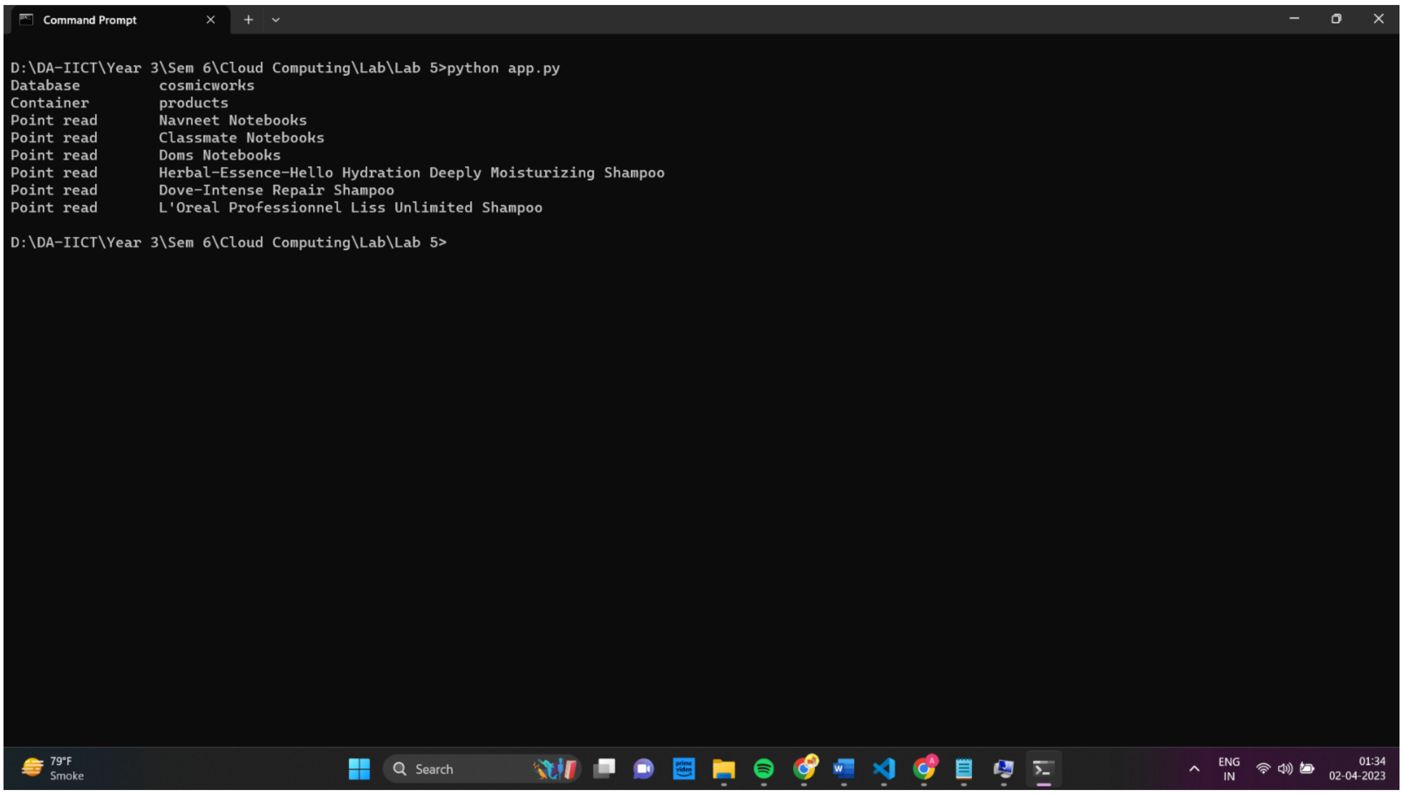
They are as follows:

```
# <new_items>

new_item1 = {
    "id": "24042002-aditiDas-Category1-Item1",
    "categoryId": "24042002-aditiDas-Category1",
    "categoryName": "student-notebooks",
    "name": "Navneet Notebooks",
    "quantity": 24,
    "sale": False,
}

new_item2 = {
    "id": "24042002-aditiDas-Category2-Item1",
    "categoryId": "24042002-aditiDas-Category2",
    "categoryName": "Shampoos",
    "name": "Herbal-Essence-Hello Hydration Deeply Moisturizing Shampoo",
    "quantity": 10,
    "sale": True,
}

new_item3 = {
    "id": "24042002-aditiDas-Category1-Item2",
    "categoryId": "24042002-aditiDas-Category1",
    "categoryName": "student-notebooks",
    "name": "Classmate Notebooks",
    "quantity": 30,
    "sale": False,
}

new_item4 = {
    "id": "24042002-aditiDas-Category2-Item2",
    "categoryId": "24042002-aditiDas-Category2",
    "categoryName": "Shampoos",
    "name": "Dove-Intense Repair Shampoo",
    "quantity": 24,
    "sale": True,
}

new_item5 = {
    "id": "24042002-aditiDas-Category1-Item3",
    "categoryId": "24042002-aditiDas-Category1",
    "categoryName": "student-notebooks",
    "name": "Doms Notebooks",
    "quantity": 8,
    "sale": True,
}


new_item6 = {
    "id": "24042002-aditiDas-Category2-Item3",
    "categoryId": "24042002-aditiDas-Category2",
    "categoryName": "Shampoos",
    "name": "L'Oreal Professionnel Liss Unlimited Shampoo",
    "quantity": 14,
    "sale": False,
}

# </new_items>


# <create_item>
container.create_item(new_item1)
container.create_item(new_item2)
container.create_item(new_item3)
container.create_item(new_item4)
container.create_item(new_item5)
container.create_item(new_item6)
# </create_item>
```

- These items were read using the following code :

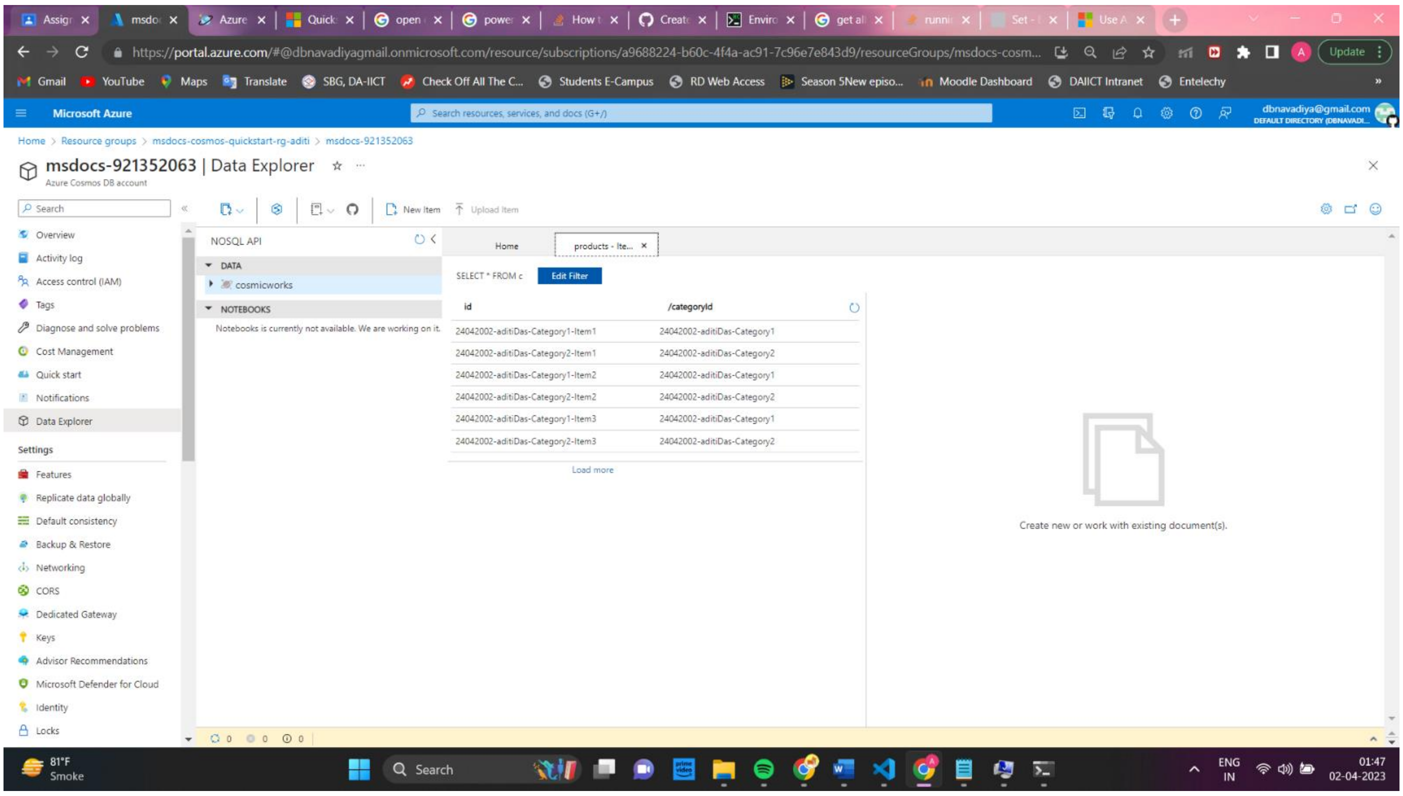
```
#<read_item>

categories = ["Category1", "Category2"]
items = ["Item1", "Item2", "Item3"]

for category in categories:
    for item in items:
        item_id = f"24042002-aditiDas-{category}-{item}"
        partition_key = f"24042002-aditiDas-{category}"
        existing_item = container.read_item(item=item_id, partition_key=partition_key)
        print("Point read\t", existing_item["name"])

#</read_item>
```

The output is as follows:



```
D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>python app.py
Database          cosmicworks
Container         products
Point read        Navneet Notebooks
Point read        Classmate Notebooks
Point read        Doms Notebooks
Point read        Herbal-Essence-Hello Hydration Deeply Moisturizing Shampoo
Point read        Dove-Intense Repair Shampoo
Point read        L'Oreal Professionnel Liss Unlimited Shampoo

D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>
```

Output on Azure Portal:



Home > Resource groups > msdocs-cosmos-quickstart-rg-aditi > msdocs-921352063

msdocs-921352063 | Data Explorer

Azure Cosmos DB account

SELECT * FROM c

id	/categoryId
24042002-aditiDas-Category1-Item1	24042002-aditiDas-Category1
24042002-aditiDas-Category2-Item1	24042002-aditiDas-Category2
24042002-aditiDas-Category1-Item2	24042002-aditiDas-Category1
24042002-aditiDas-Category2-Item2	24042002-aditiDas-Category2
24042002-aditiDas-Category1-Item3	24042002-aditiDas-Category1
24042002-aditiDas-Category2-Item3	24042002-aditiDas-Category2

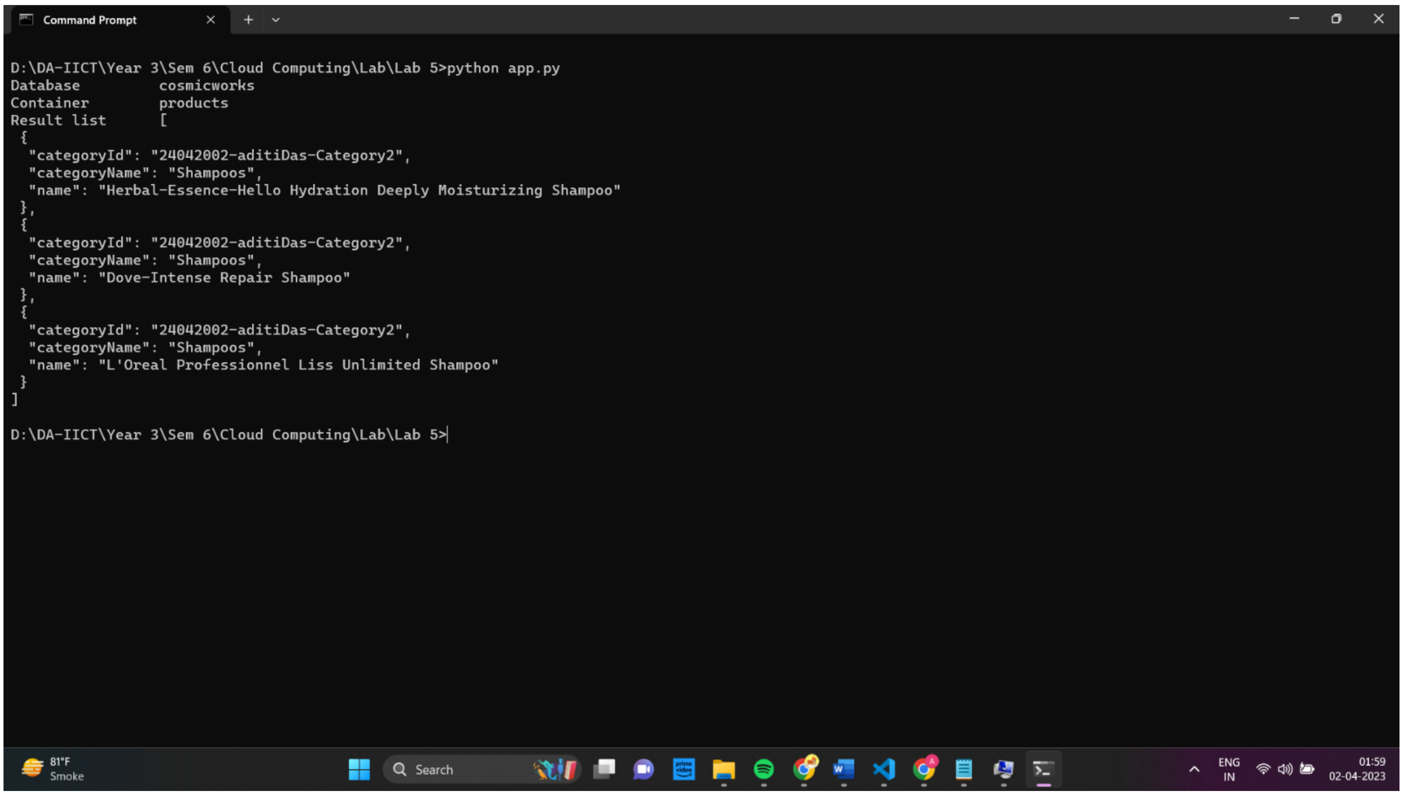
Notebooks is currently not available. We are working on it.

Create new or work with existing document(s).

- I then queried these items to obtain the name, ProductId and CategoryId of items with **categoryId = “24042002-aditiDas-Category2”** using the following code.

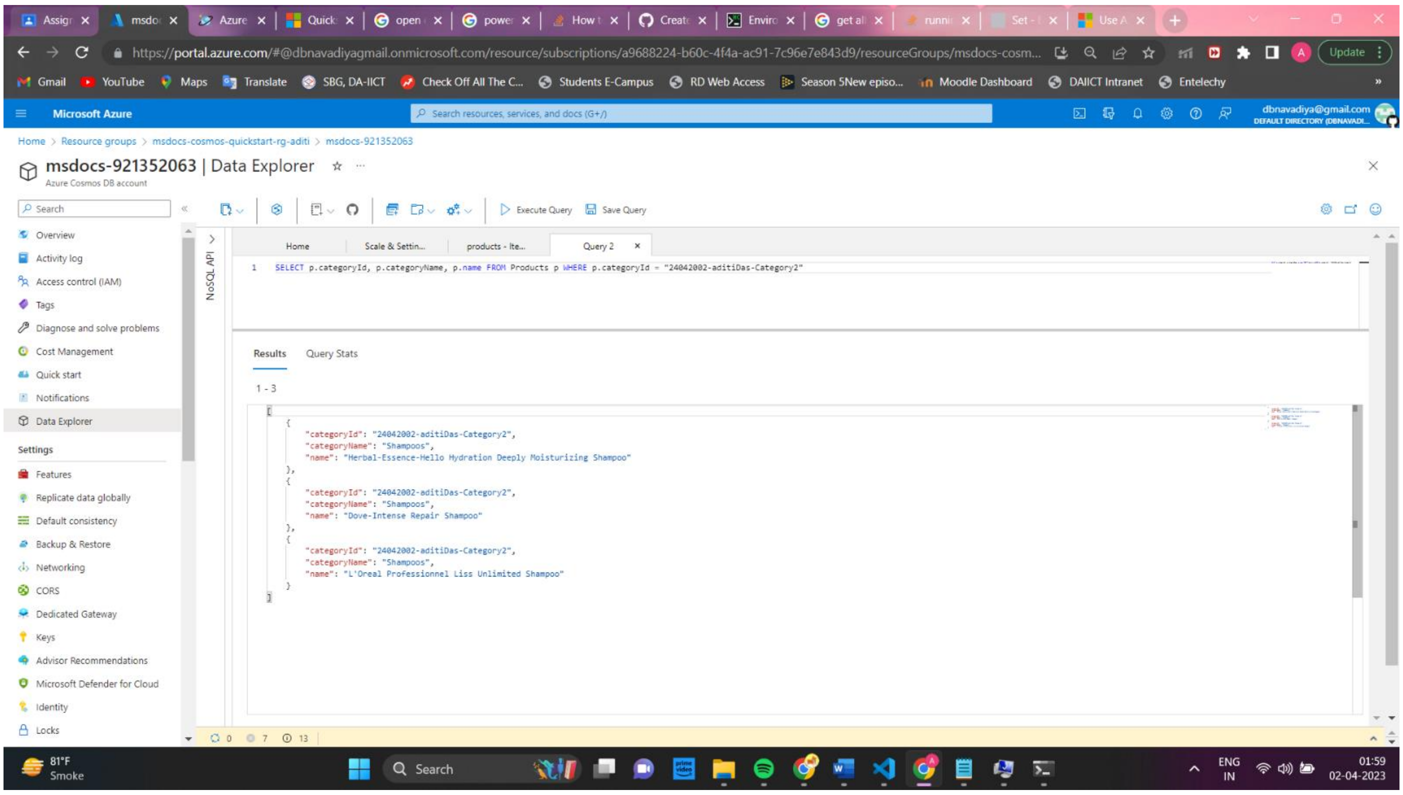
```
# <build_query>

QUERY = "SELECT p.categoryId, p.categoryName, p.name FROM products p WHERE p.categoryId =
@categoryId"
CATEGORYID = "24042002-aditiDas-Category2"
params = [dict(name="@categoryId", value=CATEGORYID)]

# </build_query>


# <query_items>

results = container.query_items(
    query=QUERY, parameters=params, enable_cross_partition_query=False
)

# </query_items>
```

Output in CMD:



```
D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>python app.py
Database          cosmicworks
Container         products
Result list       [
  {
   "categoryId": "24042002-aditiDas-Category2",
   "categoryName": "Shampoos",
   "name": "Herbal-Essence-Hello Hydration Deeply Moisturizing Shampoo"
  },
  {
   "categoryId": "24042002-aditiDas-Category2",
   "categoryName": "Shampoos",
   "name": "Dove-Intense Repair Shampoo"
  },
  {
   "categoryId": "24042002-aditiDas-Category2",
   "categoryName": "Shampoos",
   "name": "L'Oreal Professionnel Liss Unlimited Shampoo"
  }
]

D:\DA-IICT\Year 3\Sem 6\Cloud Computing\Lab\Lab 5>
```

Output on Azure Portal:



```
SELECT p.categoryId, p.categoryName, p.name FROM Products p WHERE p.categoryId = "24042002-aditiDas-Category2"
```

```
[
    {
        "categoryId": "24042002-aditiDas-Category2",
        "categoryName": "Shampoos",
        "name": "Herbal-Essence-Hello Hydration Deeply Moisturizing Shampoo"
    },
    {
        "categoryId": "24042002-aditiDas-Category2",
        "categoryName": "Shampoos",
        "name": "Dove-Intense Repair Shampoo"
    },
    {
        "categoryId": "24042002-aditiDas-Category2",
        "categoryName": "Shampoos",
        "name": "L'Oreal Professionnel Liss Unlimited Shampoo"
    }
]
```