**IT457** – Cloud Computing **Date** –March 31, 2023

**Lab Report - 5**

**Name** – Divya Kirtikumar Patel

**Student ID** - **202001420**

1. **Complete NoSQL design for following problems**
   1. **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?**

To design a data solution in NoSQL for this scenario, the following steps will be undertaken:

1. High-resolution images will be stored in an Azure Storage Blob and their URL endpoints will be obtained.
2. A table named “All-Items” will be created with “Category Type” as the Partition Key. This will serve as both a group key and a balanced key. The Row Key for this table will contain the unique product id for the given Partition Key. For example, a valid Partition Key would be “Clothes” and a valid Row Key for it would be “Shirt-1”. Other valid key sets might include {“Clothes”,”Shirt-2”} and {“Toys”,”Doll-1”}.
3. The “All-Items” table will store properties in a Key-Value form, including the URL endpoint for the high-resolution image. It will also contain properties such as name, price, available quantity (SKU), etc. This will prevent looping in case the user wants to view the image in higher resolution.
4. Additional tables in this data solution will depend on the specific requirements of the client. For example, if a client (gift shop owner/manager) wants to be notified when a product is running low on stock (let us assume that this is true when the quantity of the product is less than 4), a table named “Restock-Urgently” could be designed.
5. The “Restock-Urgently” table would have the SKU of an item as its Partition Key to prevent looping. The Row Key for this table would be the combination of the Partition Key and Row Key of the “All-Items” table. Data will automatically be added to this table when the SKU for a product falls below 4 and removed when it rises above 3.
6. Having such a table prevents an O(n^2) complexity where one has to iterate through all the Partition Keys and then within them, the Row Keys for “All-Items”.

**b). 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:

1. The primary objective of this solution is to track the progress of individual employees in their training modules. Azure Blob Storage can be used to store the URL endpoints of these training modules.
2. The modules can be identified as M-1, M-2,…,M-500 and the employees as E-1, E-2,…, E-N.
3. A table named “Module-Information” will be created with the module ids (M-1,M-2,…M-500) as the Partition Key and a random value (e.g. 1) as the Row Key. The URL endpoints will be stored alongside these Partition Keys.
4. Although not necessary for the problem statement mentioned here, another table named “Employee-Details” can be designed with the Division that the employee works in as the Partition Key and the Employee-ID as the Row Key.
5. To track an employee’s progress in their training modules, a table named “Employee-Training-Status” will be created with the Employee-ID as the Partition Key and the Module-ID as the Row Key. This avoids an O(n^2) complexity when tracking a single employee’s progress.
6. The progress of each module will be stored in a property named “Progress” and an internal API will convert the duration of the video watched or number of pages read to a percentage 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).

Here are some additional points that can be included in the design of a data solution for tracking employee progress in training modules:

* The “Employee-Training-Status” table can also include additional properties such as the date and time when the employee started and completed each module. This information can be used to generate reports on employee progress and identify areas where additional training may be needed.
* An internal API can be used to update the “Progress” property in real-time as the employee completes each module. This can provide managers with up-to-date information on employee progress and allow them to take timely action if needed.
* The “Employee-Details” table can include additional information about each employee such as their name, job title, and contact information. This can facilitate communication between managers and employees and help managers to provide personalized feedback and support.
* The solution can be designed to generate alerts or notifications when an employee completes a module or reaches a certain level of progress. This can help managers to stay informed and provide timely recognition and feedback to employees.

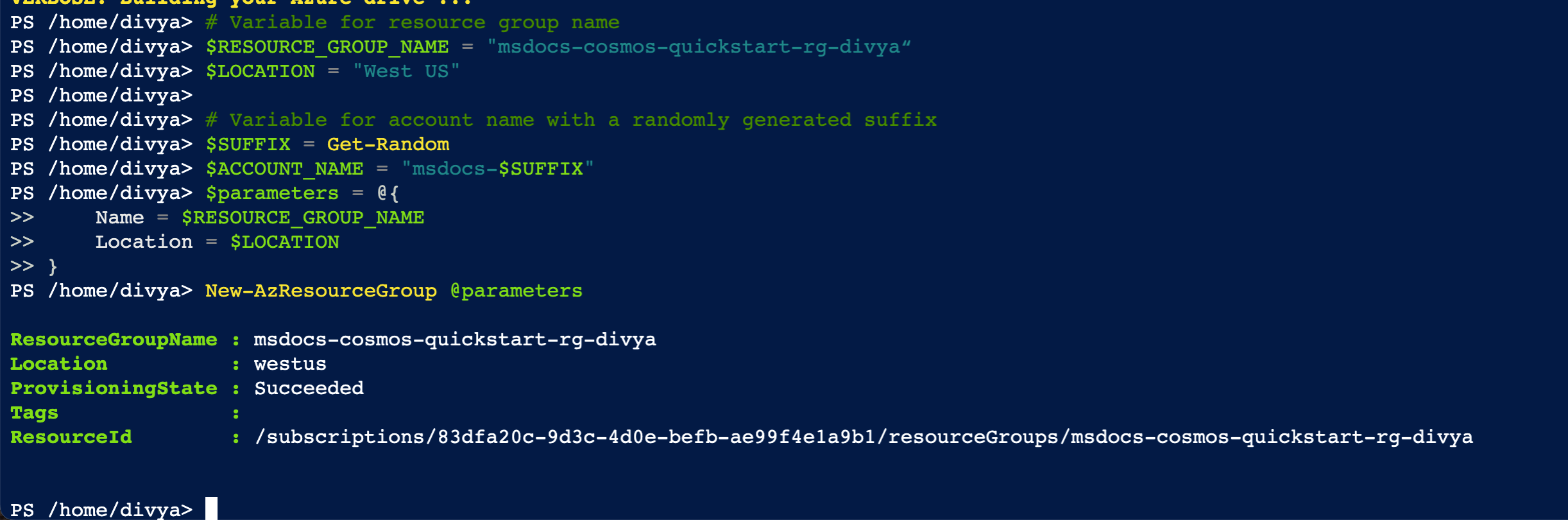
1. **Go through the following tutorials:**

**Noteworthy Features about Azure Cosmos DB based on the tutorial:**

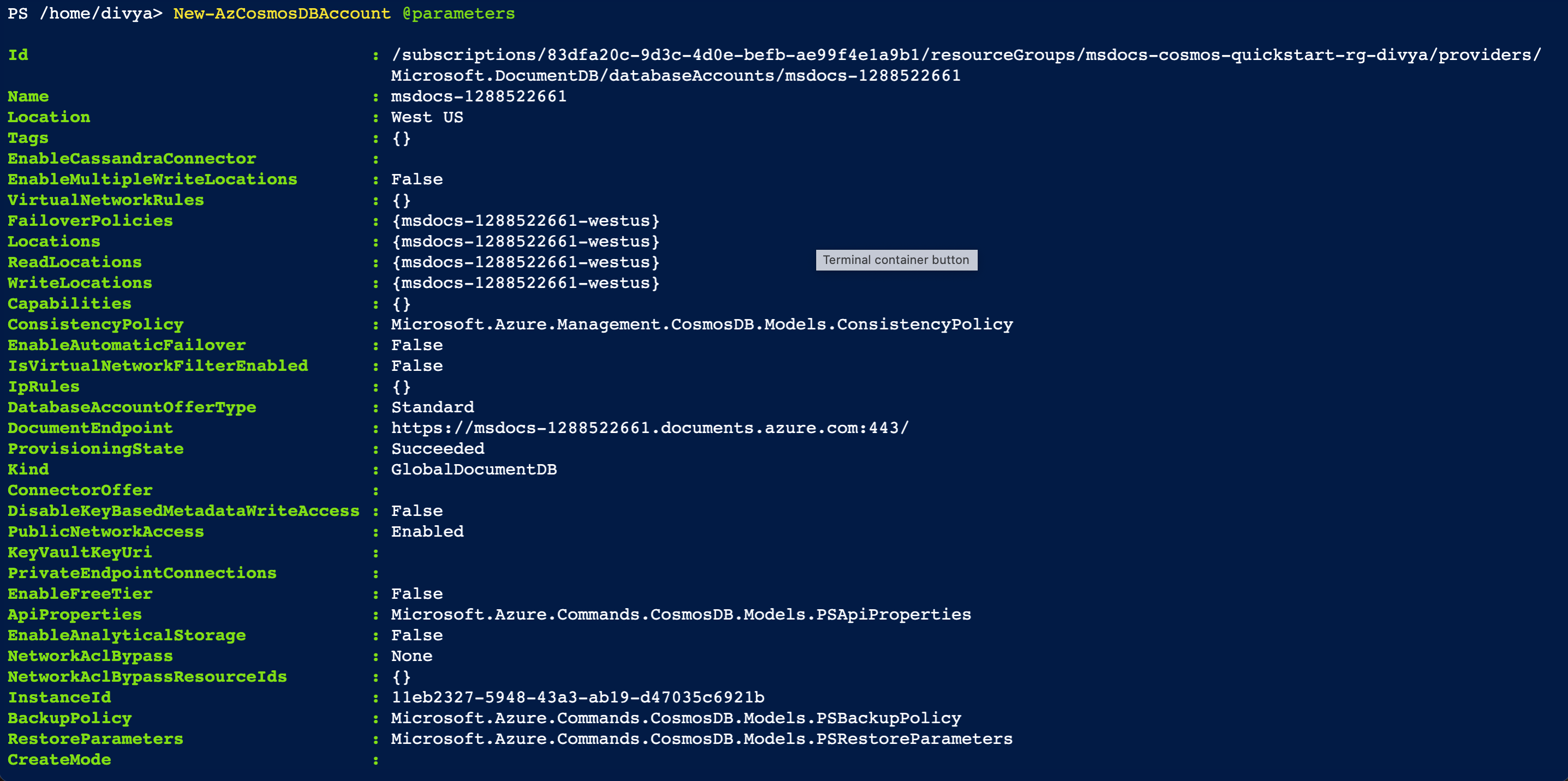
1. **Global distribution**: Azure Cosmos DB is a globally distributed database service that allows you to easily store and manage data across multiple regions around the world. This feature ensures low latency access to data and high availability of your application, regardless of the geographic location of your users.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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.

**2. Azure Cosmos DB for NoSQL client library for Python**

**Creating A Resource Group**

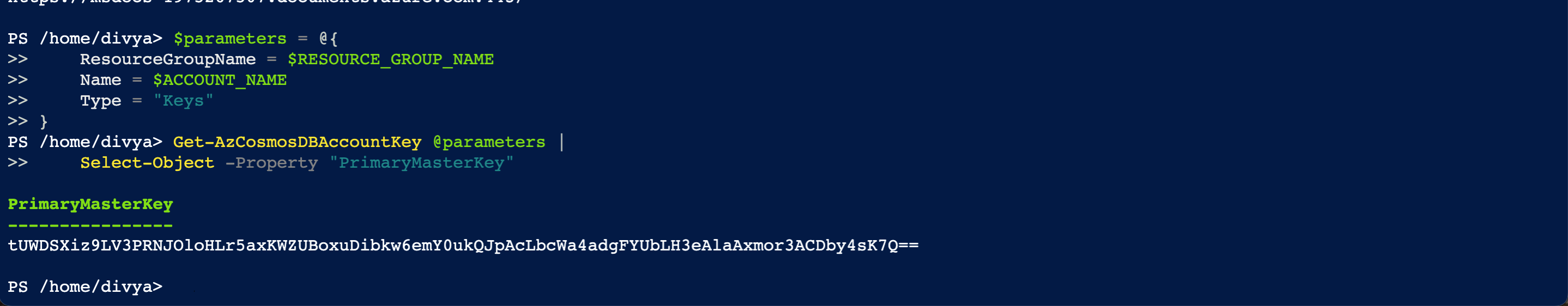
****

**Creating CosmosDBAccount:**

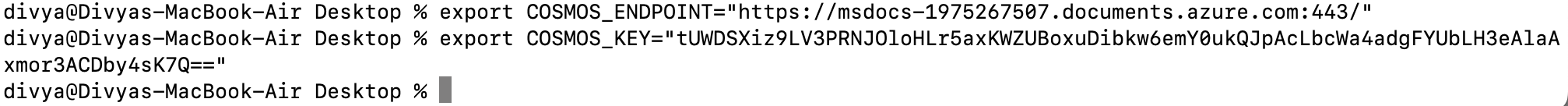
****

**Getting Endpoint URLS :**

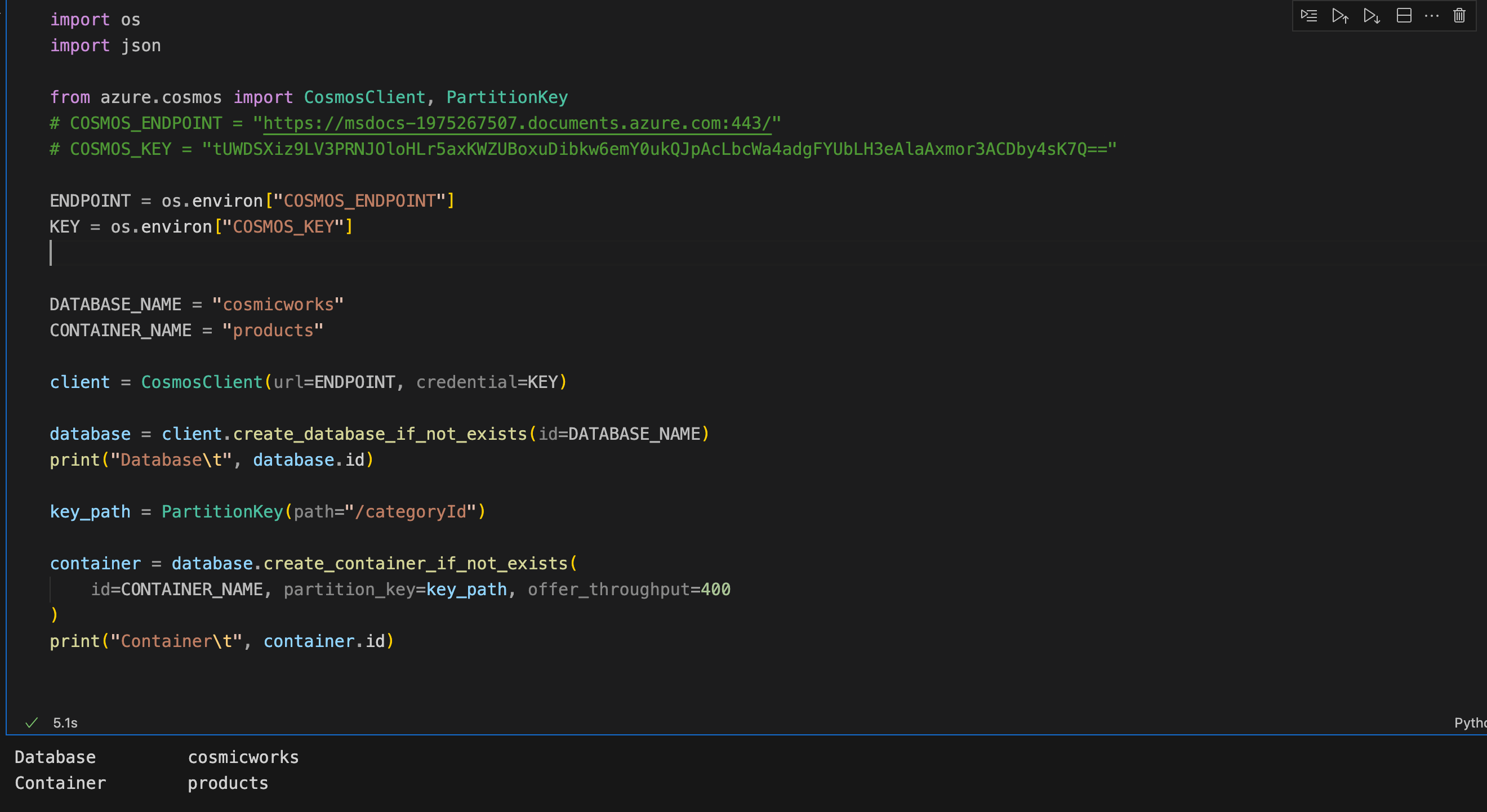
****

****

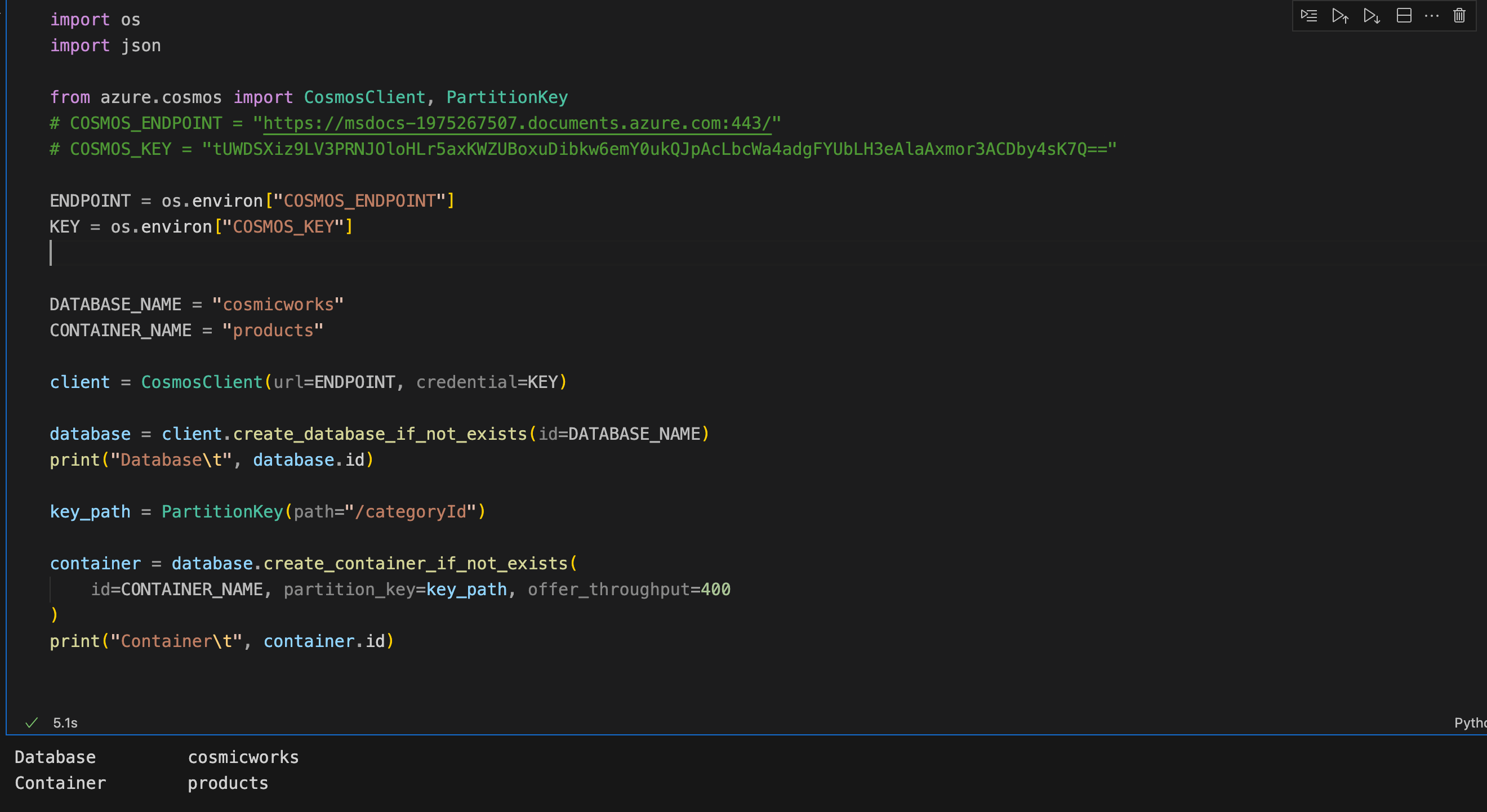
**Exporting Environment Variables:**

****

client using Python using the value of these environment variables as follows

****

database and a container

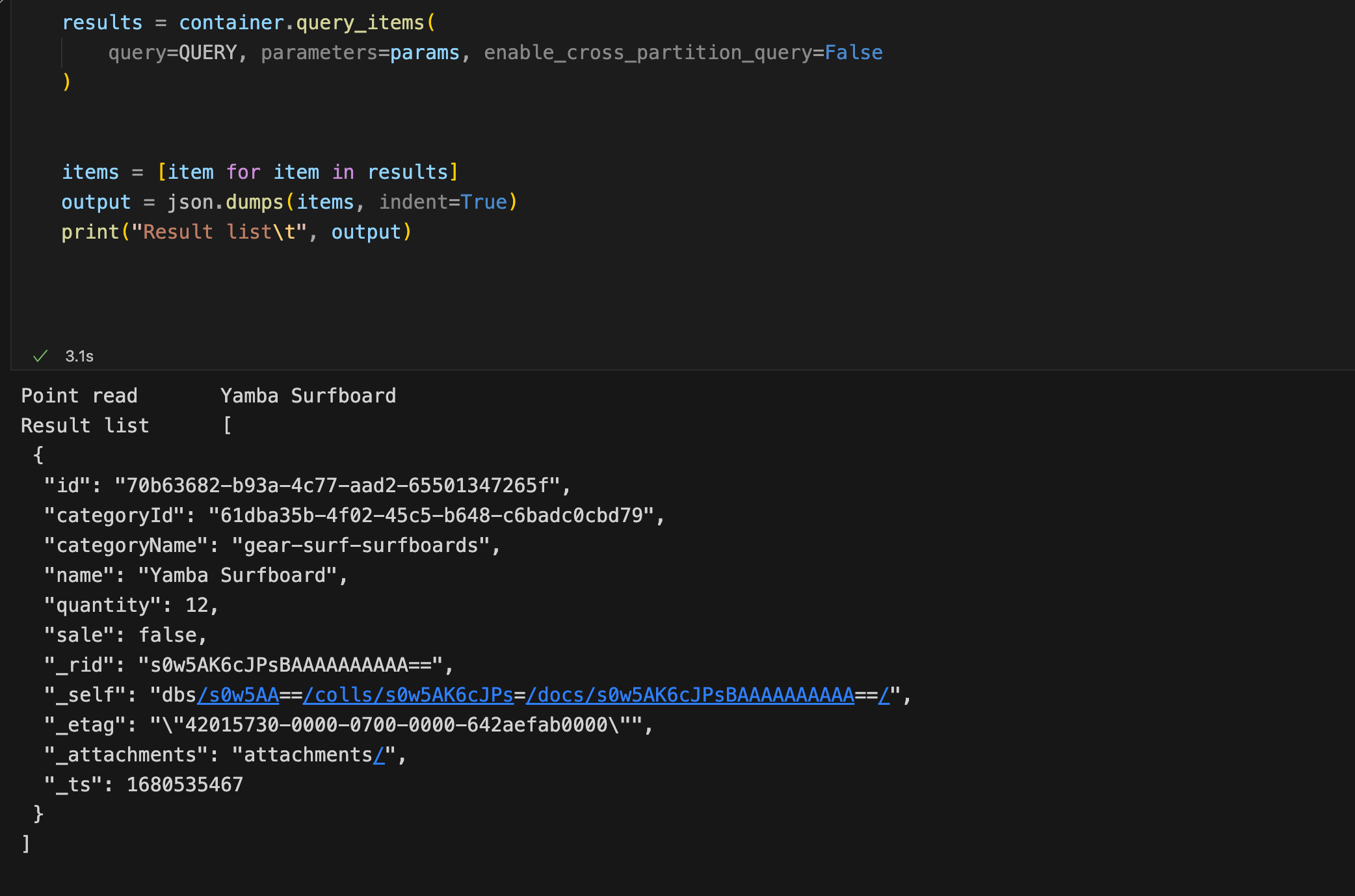
****

**Output**

**Inserting And Creating New Entry in Database**

****

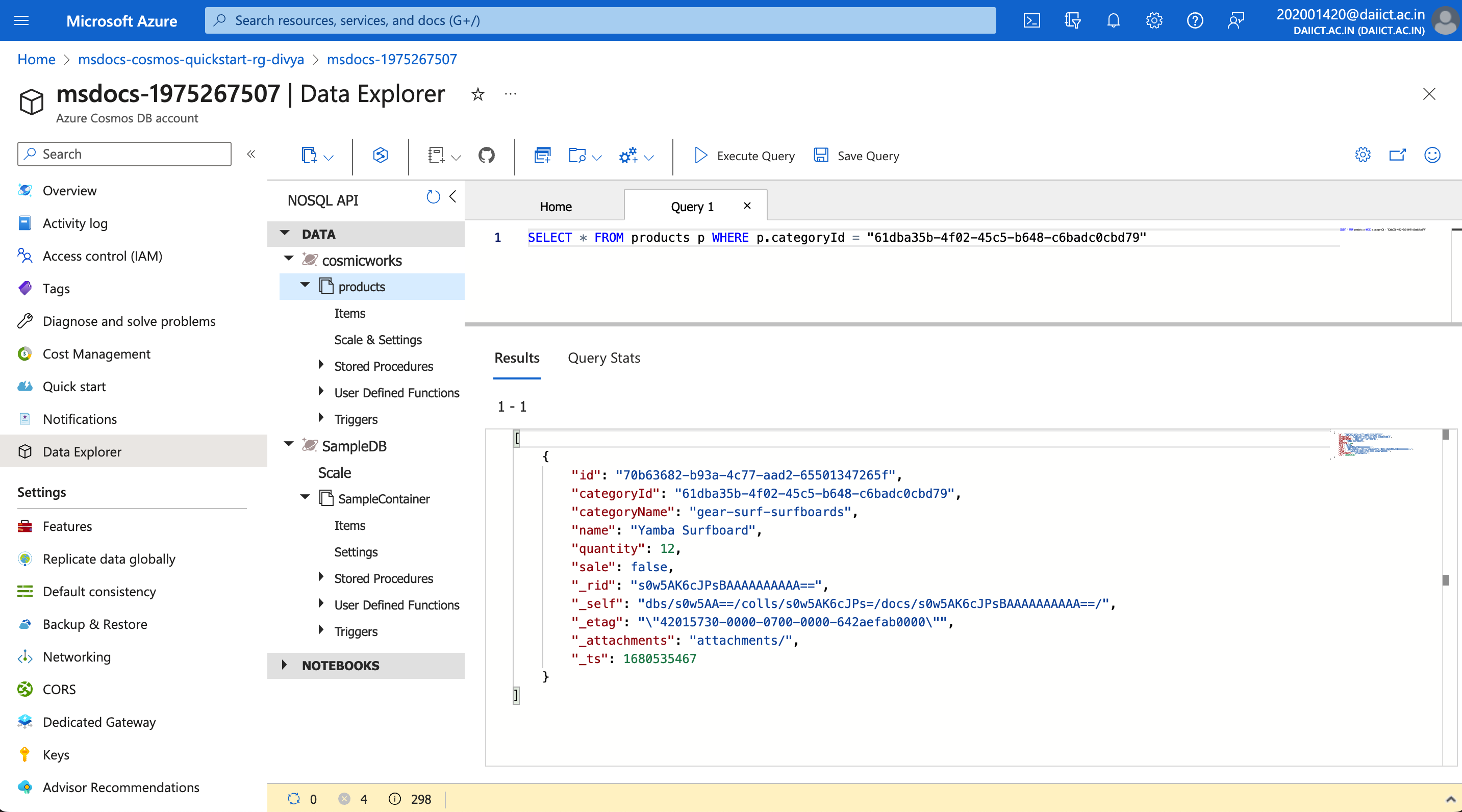
**Querying the items**

****

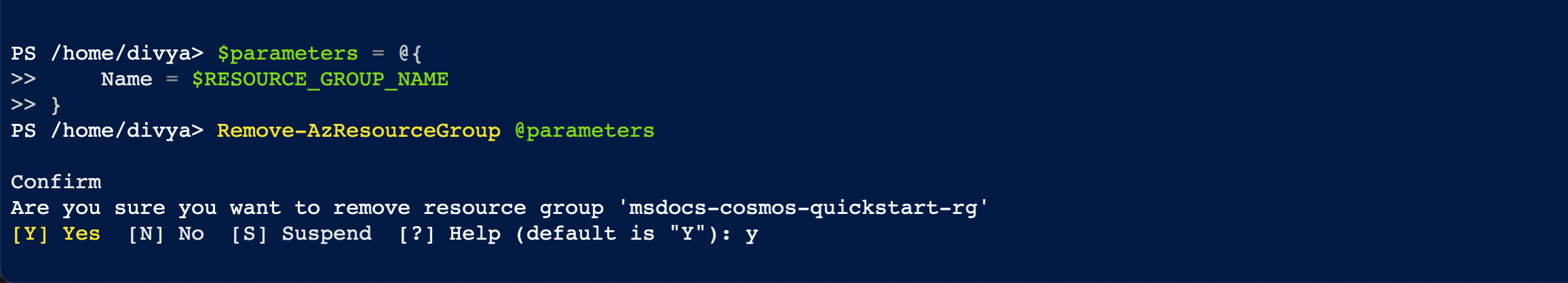
**Output in Local Environment (Red Box)**

**[Using Jupyter Notebook for Python]**

**Output on Azure Portal Using Data Explorer**

****

**Cleaning Up the Resources:**

****