**Lab 2: Account for and Count the Accounts**

Woodgrove Bank provided schemas for their member and account data so you and your team can generate sample data to load into the new Azure Cosmos DB account.

**Objective**

* Run the console application to generate and load the sample data into Cosmos DB.
* Verify that the data was successfully loaded.

**Task 1: Load sample account and member data into the Azure Cosmos DB account**

You can run the website and the REST API that supports it locally. You need to first update your local configuration and then you can run the solution in the debugger using Visual Studio.

1. Click on the Start button, search for +++**Visual Studio**+++ and select **Visual Studio 2022**.

A screenshot of a computer

Description automatically generated

1. Sign in with your **Azure login credentials** if prompted.

**A screenshot of a computer

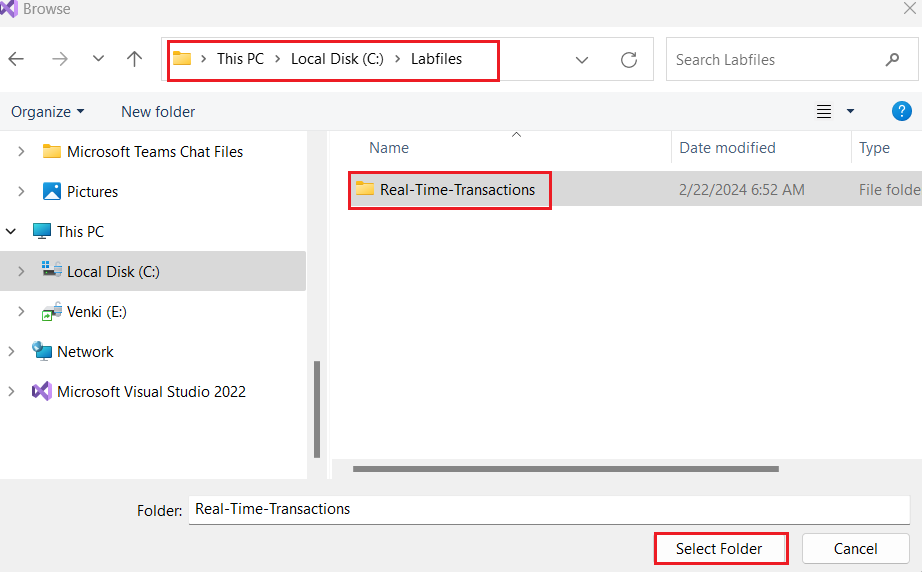
Description automatically generated**

1. Click on **Open a local folder**.

A screenshot of a computer

Description automatically generated

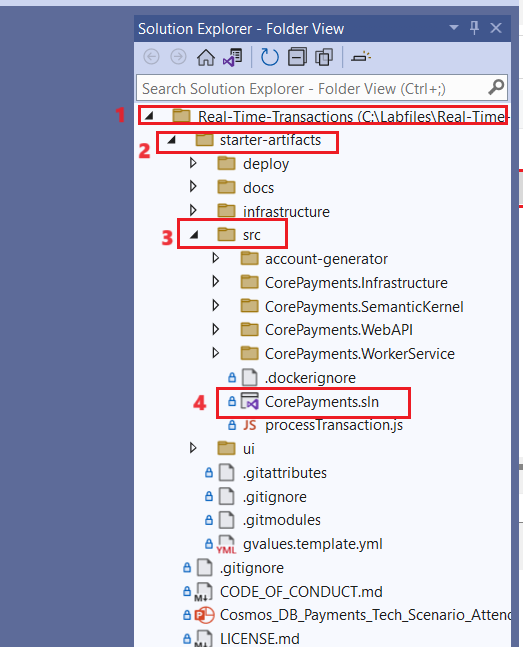
1. Navigate and select **Real-time-Payment-Transaction** folder from **C:\LabFiles** and click on the **Select Folder** button.

****

**A blue screen with a yellow line

Description automatically generated**

1. In the Visual Studio,in **Solution Explorer-Folder view** navigate and double click on **src ->** **CorePayments.sln** file.

****

1. Wait for the **NuGetpackegs** to get restored.

**A screenshot of a computer

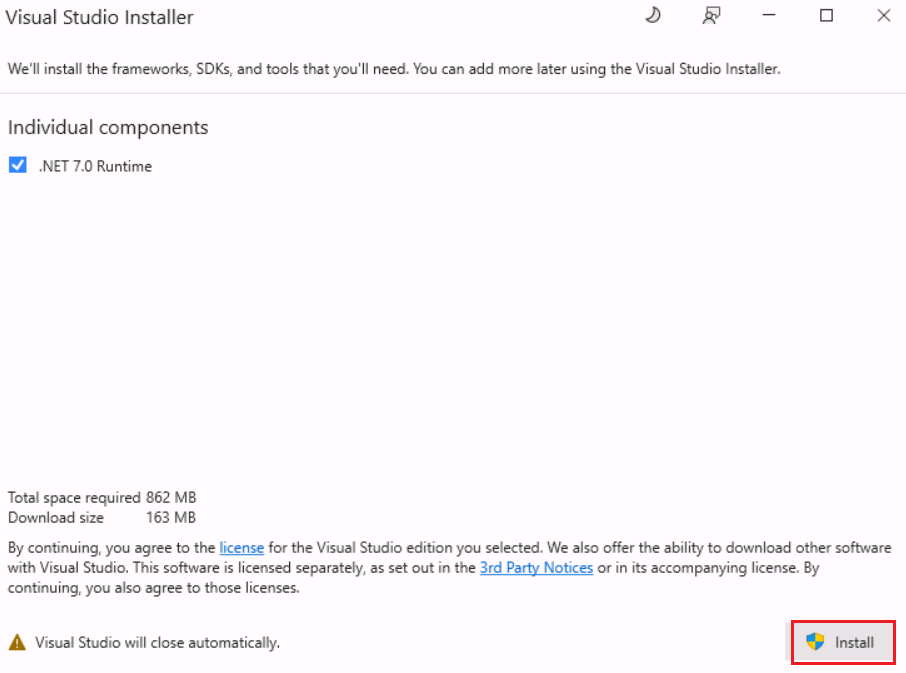
Description automatically generated**

1. Once done and it says **Ready** in the foot bar, click on **Install** in the warning, **This Project is targeting a version of .NET** which is not installed.

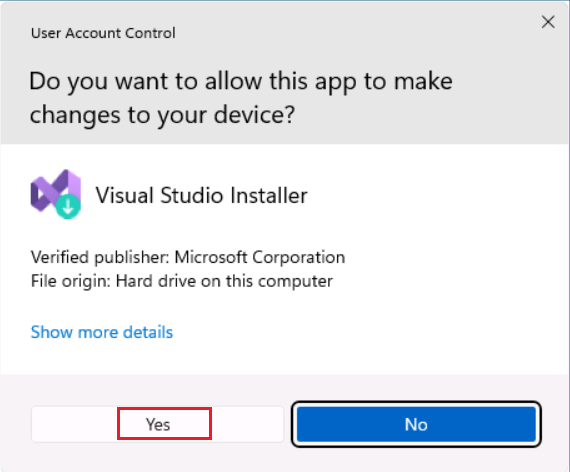
**A computer screen shot of a blue screen

Description automatically generated**

1. On the Visual Studio Installer tab, select **.NET 7.0 Runtime** and click onthe **Install** button.

****

1. Click on the **yes** button.

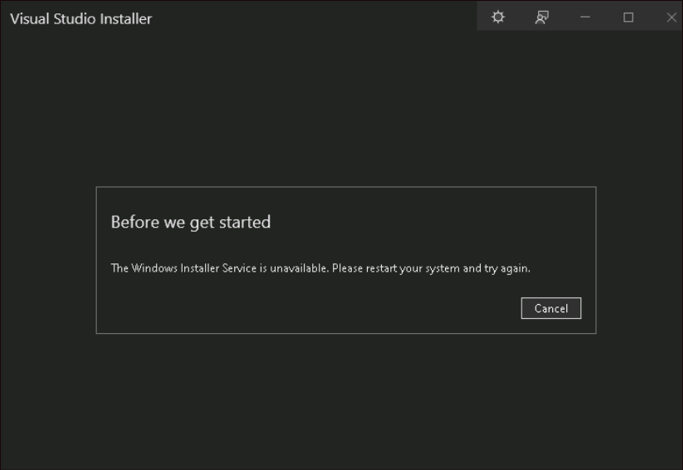
****

1. Click on **Close** once done.

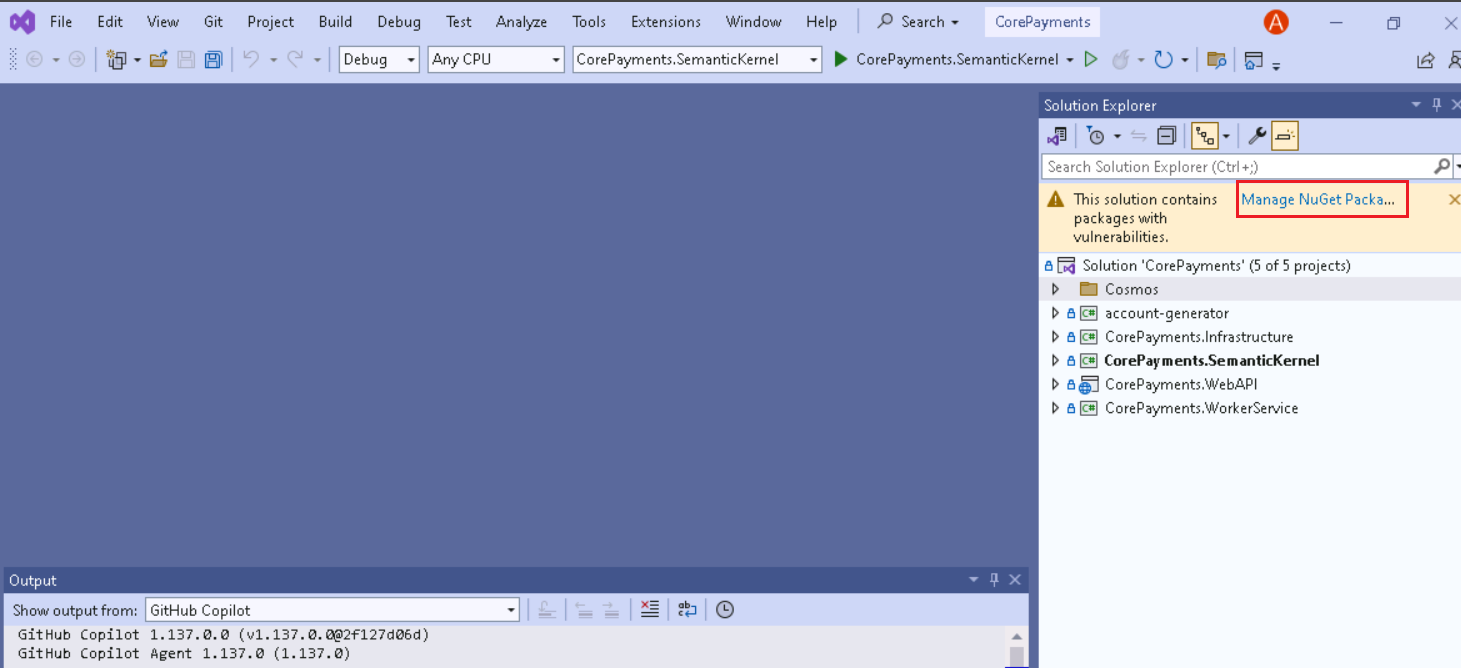
**A screenshot of a computer

Description automatically generated**

**Note:** If there is any issue while installing the .NET 7 or the packages due to network as below, please restart your VM and then continue with the lab execution.



1. Select **Manage NuGet package** in the warning, **This solution contains packages with vulnerabilities**.

****

1. In the NuGet – Solution window, click on the **Updates**, select the checkbox against **Azure.identity** and then select **Update**.

**A screenshot of a computer

Description automatically generated**

1. Click on **Apply** and then **I Accept** in theLicense Acceptance window.

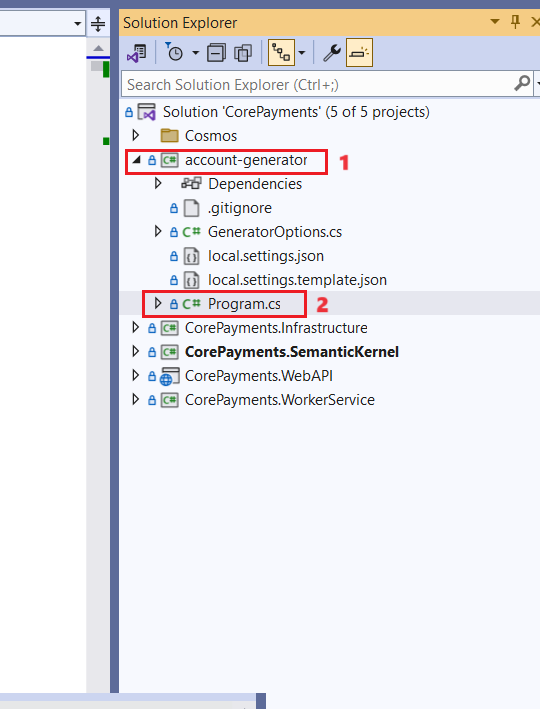
**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. Select the **Program.cs** file from the **account-generator** folder.



1. Go through the **MainAsync** and **CreateMembersAsync** methods to get familiar with the Cosmos DB SDK and understand the data model, including the global index.

public static async Task MainAsync(string[] args)

{

var configuration = new ConfigurationBuilder().SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("local.settings.json")

.AddJsonFile("settings.json", optional: true, reloadOnChange: true)

.AddCommandLine(args, new Dictionary<string, string>

{

{"-m", $"{nameof(GeneratorOptions)}:{nameof(GeneratorOptions.RunMode)}"},

{"-s", $"{nameof(GeneratorOptions)}:{nameof(GeneratorOptions.SleepTime)}"},

{"-c", $"{nameof(GeneratorOptions)}:{nameof(GeneratorOptions.BatchSize)}"},

{"-v", $"{nameof(GeneratorOptions)}:{nameof(GeneratorOptions.Verbose)}"}

})

.AddEnvironmentVariables()

.Build();

GeneratorOptions options = new();

configuration.GetSection(nameof(GeneratorOptions))

.Bind(options);

Console.WriteLine("To STOP press CTRL+C...");

Console.CancelKeyPress += Console\_CancelKeyPressHandler;

cosmosClient = new CosmosClient(configuration["CosmosDbConnectionString"],

new CosmosClientOptions() {AllowBulkExecution = true, EnableContentResponseOnWrite = false});

// TODO: Instantiate the transactionsContainer and membersContainer with new Container objects from the CosmosClient.

// Use the "payments" database and "transactions", "members", and "globalIndex" containers.

transactionsContainer = cosmosClient.GetContainer("payments", "transactions");

membersContainer = cosmosClient.GetContainer("payments", "members");

globalIndexContainer = cosmosClient.GetContainer("payments", "globalIndex");

// Generate Members if they don't already exist:

var memberList = await CreateMembersAsync();

var tasks = new List<Task>();

try

{

for (var i = 1; i <= 5; i++)

{

tasks.Add(LoadAsync(i, options, memberList));

}

Task.WhenAll(tasks).GetAwaiter().GetResult();

}

catch (Exception ex)

{

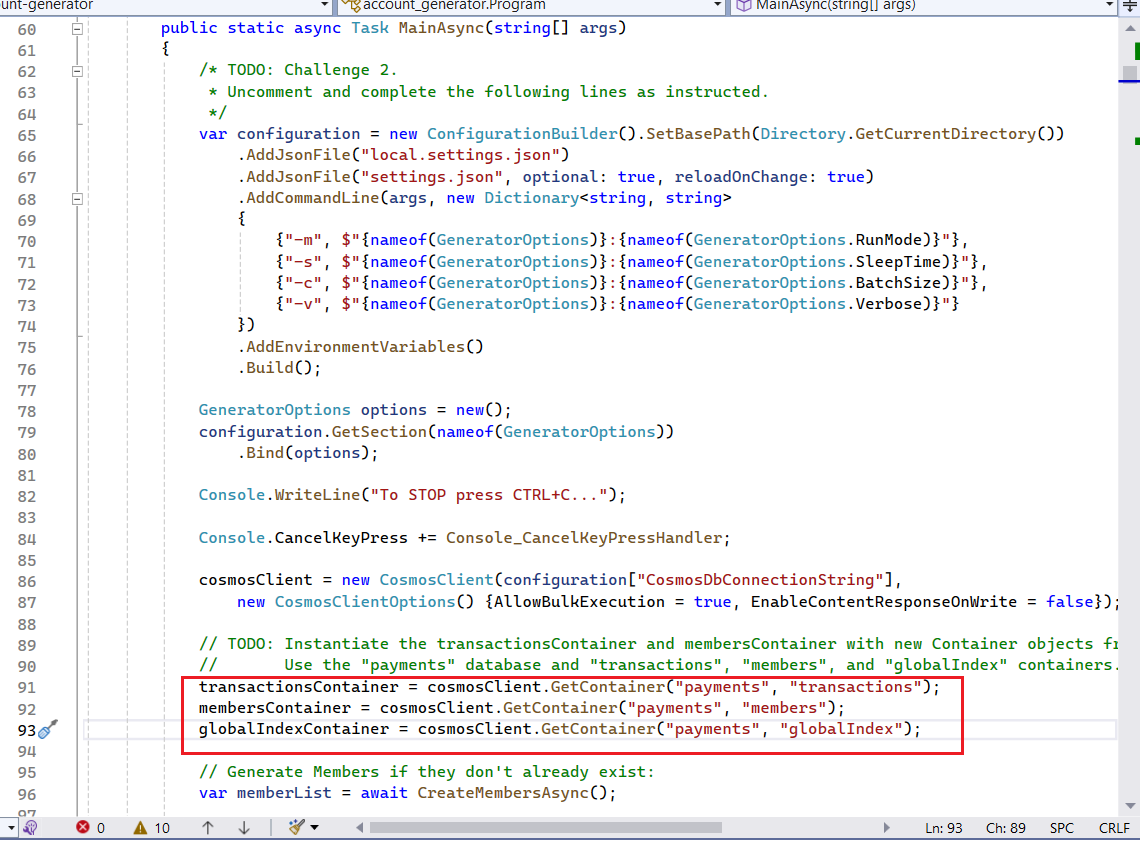
Console.WriteLine(ex.Message);

}

Console.WriteLine("Completed generating data.");

cosmosClient.Dispose();

}



private static async Task<List<Member>> CreateMembersAsync()

{

var memberList = new List<Member>();

var query = "SELECT \* FROM c";

var queryDefinition = new QueryDefinition(query);

var resultIterator = membersContainer.GetItemQueryIterator<Member>(queryDefinition);

while (resultIterator.HasMoreResults)

{

var response = await resultIterator.ReadNextAsync();

memberList.AddRange(response);

}

if (memberList.Count > 0)

{

Console.WriteLine("Skipping Member record generation since records already exist.");

return memberList;

}

for (var i = 0; i <= 350; i++)

{

var memberId = Guid.NewGuid().ToString();

var memberFaker = new Faker<Member>()

.RuleFor(u => u.memberId, (f, u) => memberId)

.RuleFor(u => u.firstName, (f, u) => f.Name.FirstName())

.RuleFor(u => u.lastName, (f, u) => f.Name.LastName())

.RuleFor(u => u.email, (f, u) => f.Internet.Email())

.RuleFor(u => u.phone, (f, u) => f.Phone.PhoneNumber())

.RuleFor(u => u.address, (f, u) => f.Address.StreetAddress())

.RuleFor(u => u.city, (f, u) => f.Address.City())

.RuleFor(u => u.state, (f, u) => f.Address.State())

.RuleFor(u => u.zipcode, (f, u) => f.Address.ZipCode("#####"))

.RuleFor(u => u.country, (f, u) => "USA")

.RuleFor(u => u.type, (f, u) => Constants.DocumentTypes.Member)

.RuleFor(u => u.memberSince, (f, u) => f.Date.Past(20));

await \_pollyRetryPolicy.ExecuteAsync(async () =>

{

// TODO: Finish the code below to Upsert the Member item in the Members container.

// Set the partition key to the memberId. It is important that you upsert, not insert.

var member = memberFaker.Generate();

await membersContainer.UpsertItemAsync(member, new PartitionKey(memberId));

memberList.Add(member);

// Create a global index lookup for this member.

var globalIndex = new GlobalIndex

{

partitionKey = memberId,

targetDocType = Constants.DocumentTypes.Member,

id = memberId

};

await globalIndexContainer.UpsertItemAsync(globalIndex, new PartitionKey(globalIndex.partitionKey));

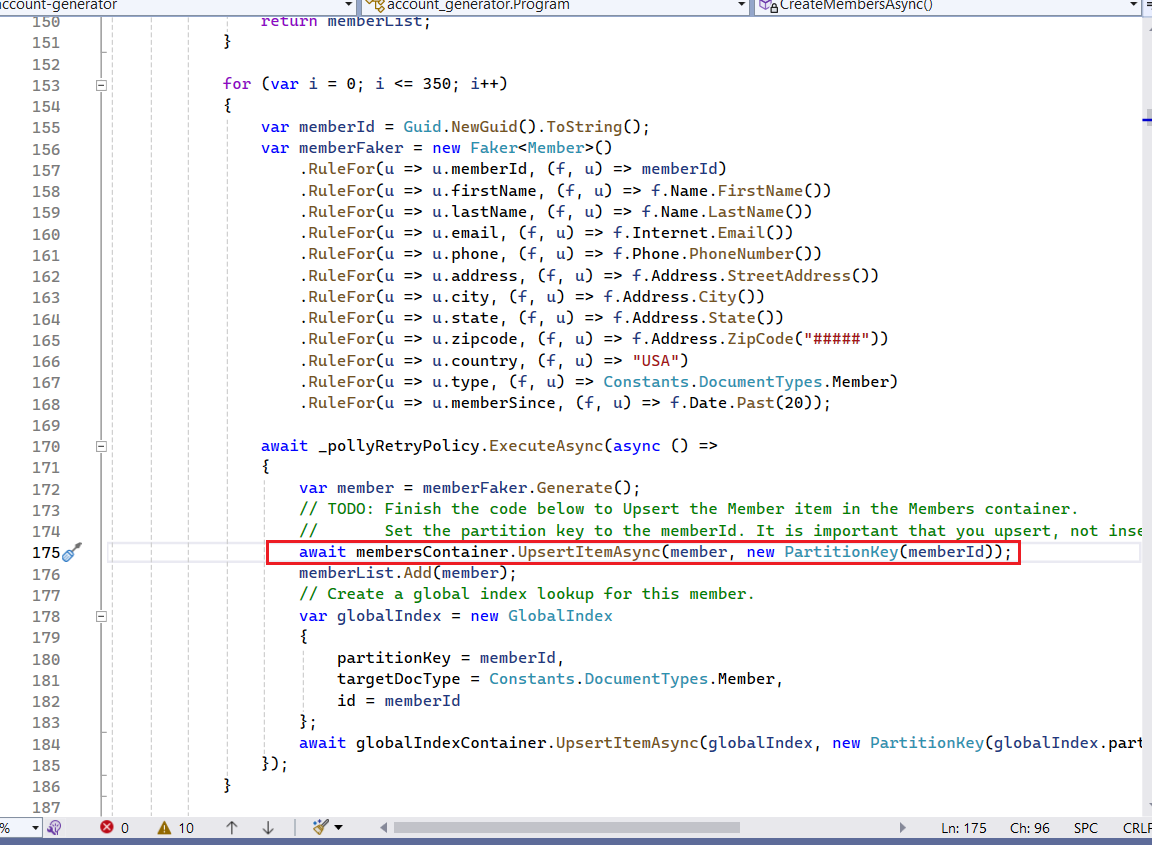
});

}

Console.WriteLine("Finished generating Members.");

return memberList;

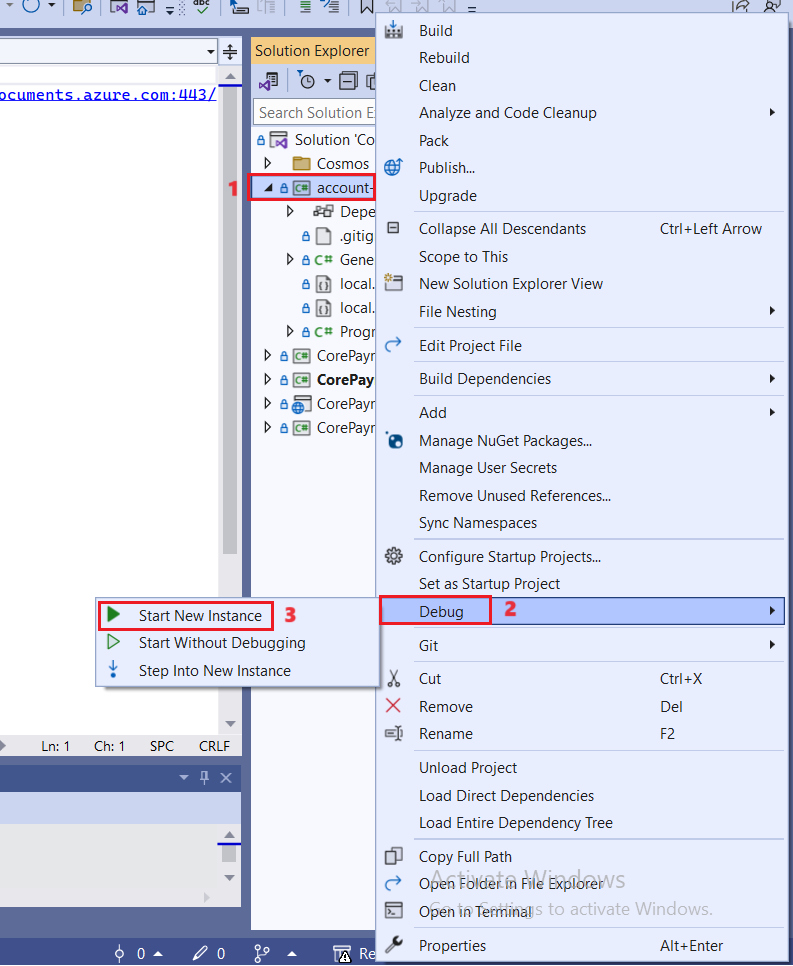
}



1. Ensure the **local.settings.json** file has been created for the account generator project. If not, copy the **local.settings.template.json**file and add the **CosmosDbConnectionString**value as in the screenshot below.



1. In the **Solution Explorer-Folder view,** right-click the **account-generator** project, select **Debug**, then **Start new instance**.



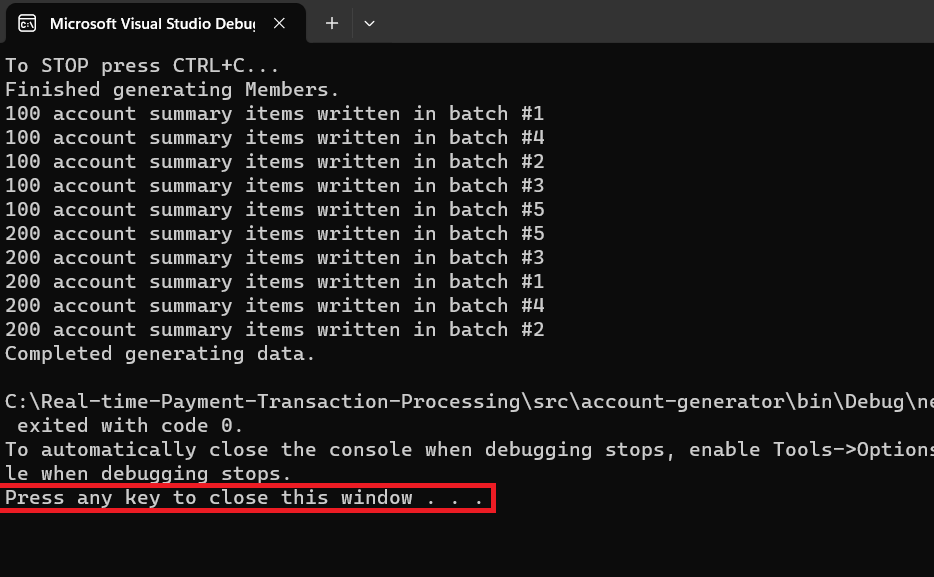
1. The account generator should start running and generating data. It will take a maximum of 15 minutes to generate all the data. You can monitor the progress in the console window.

A screenshot of a computer

Description automatically generated

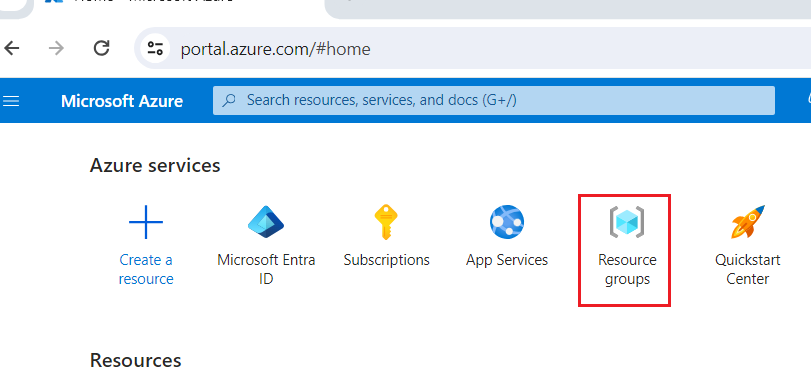
A screenshot of a computer

Description automatically generated

19. Press any key, when it says **completed generating data** and prompts to **Press any key to close this window**.

## Task 2: Verify the data loaded into Azure Cosmos DB from the Azure portal

1. Open a browser and go to +++[**https://portal.azure.com**](https://portal.azure.com)+++ and sign in with your Azure login credentials, if not logged in already.
2. From the Home page, click on **Resource groups** tile.



1. Click on your resource group name and then click on **CosmosDB account** name from the list of resources.

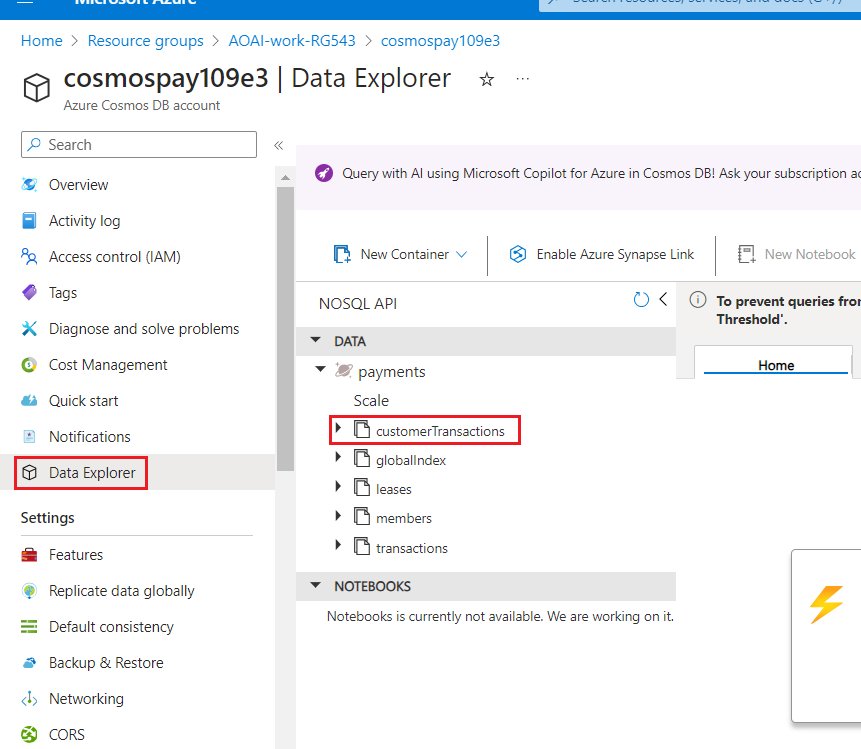
A screenshot of a computer

Description automatically generated

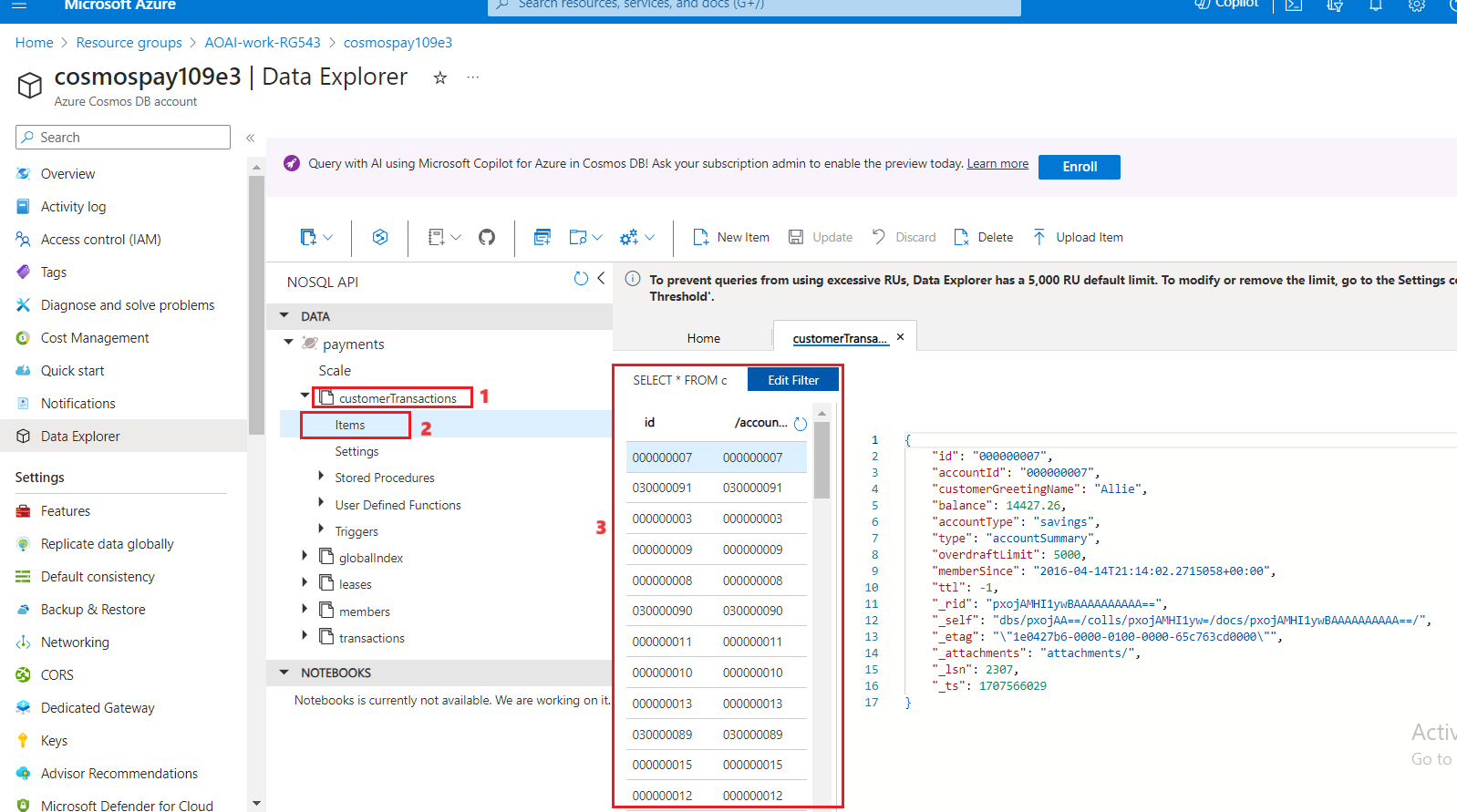
A screenshot of a computer

Description automatically generated

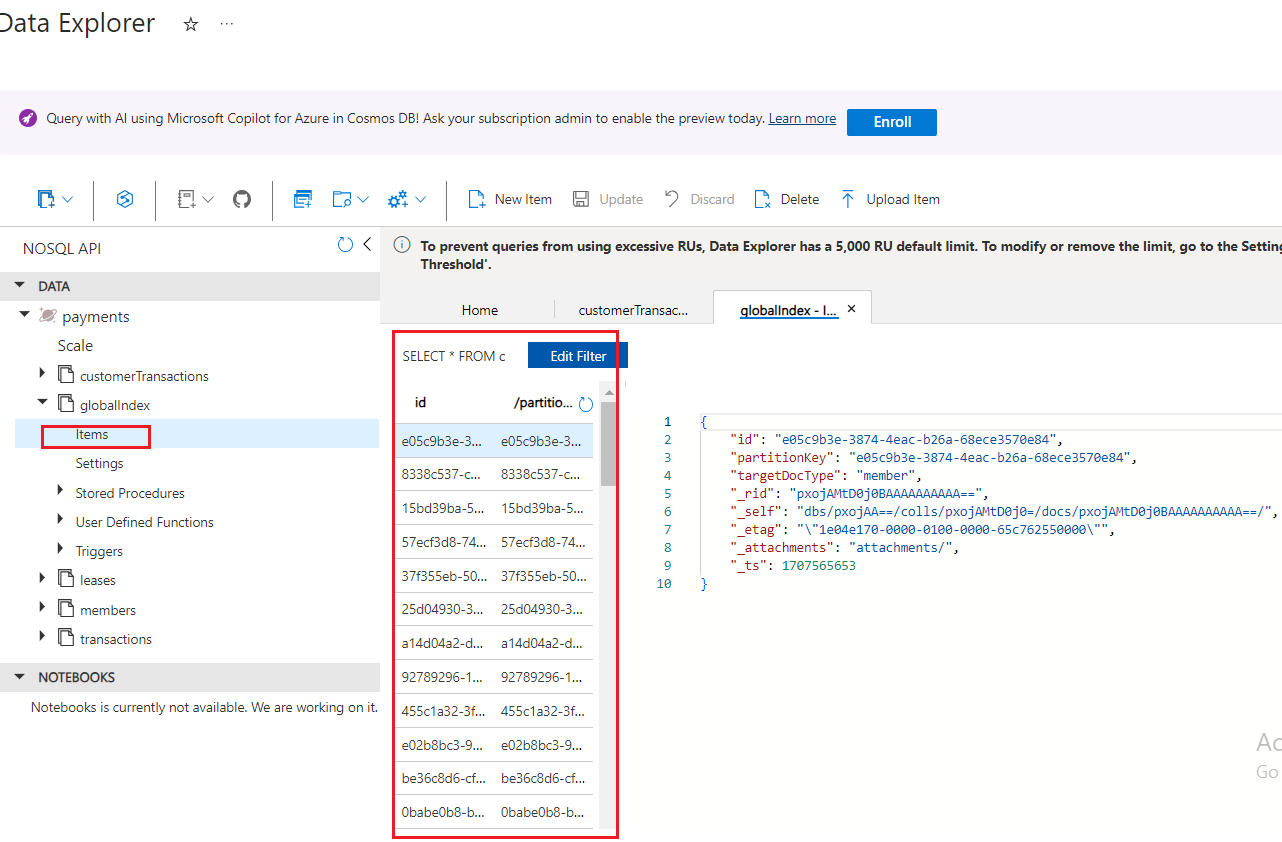
1. Click on **Data Explorer**. Select the **CustomerTransactions** container data.



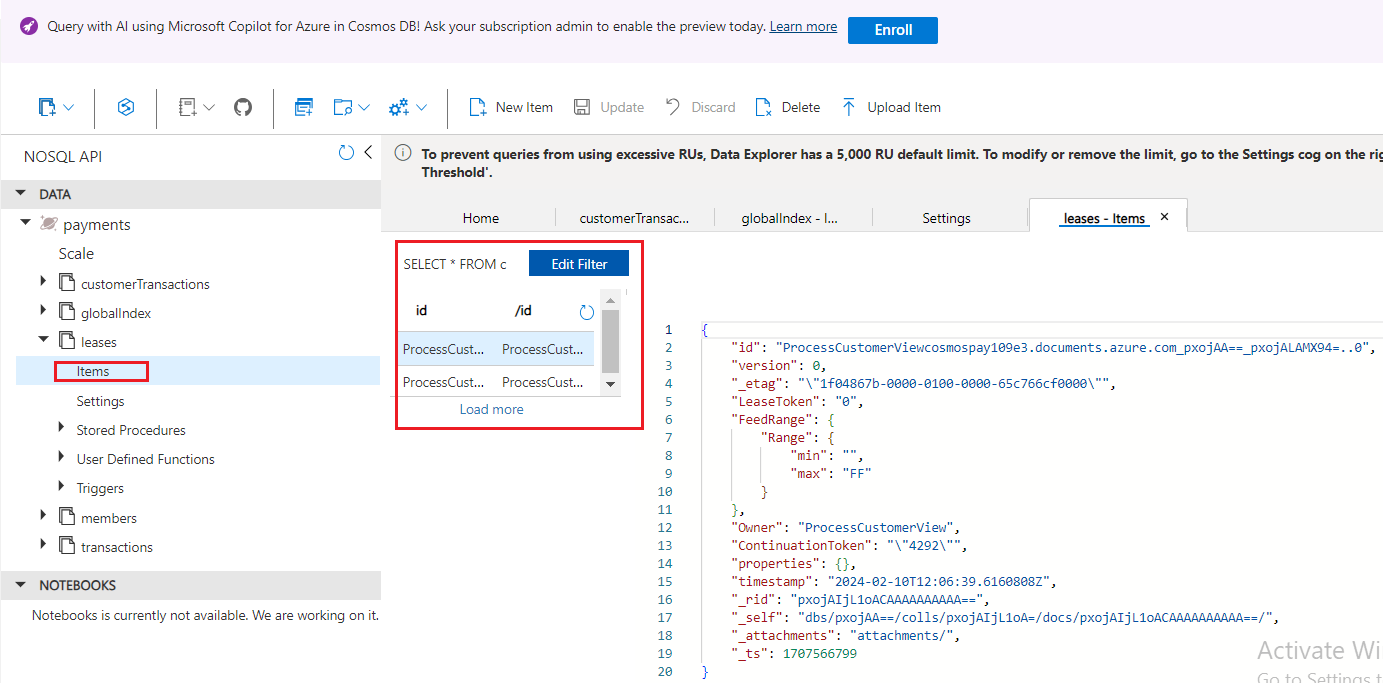
1. Click on **Items** under the Container and ensure that it has been loaded with the data.



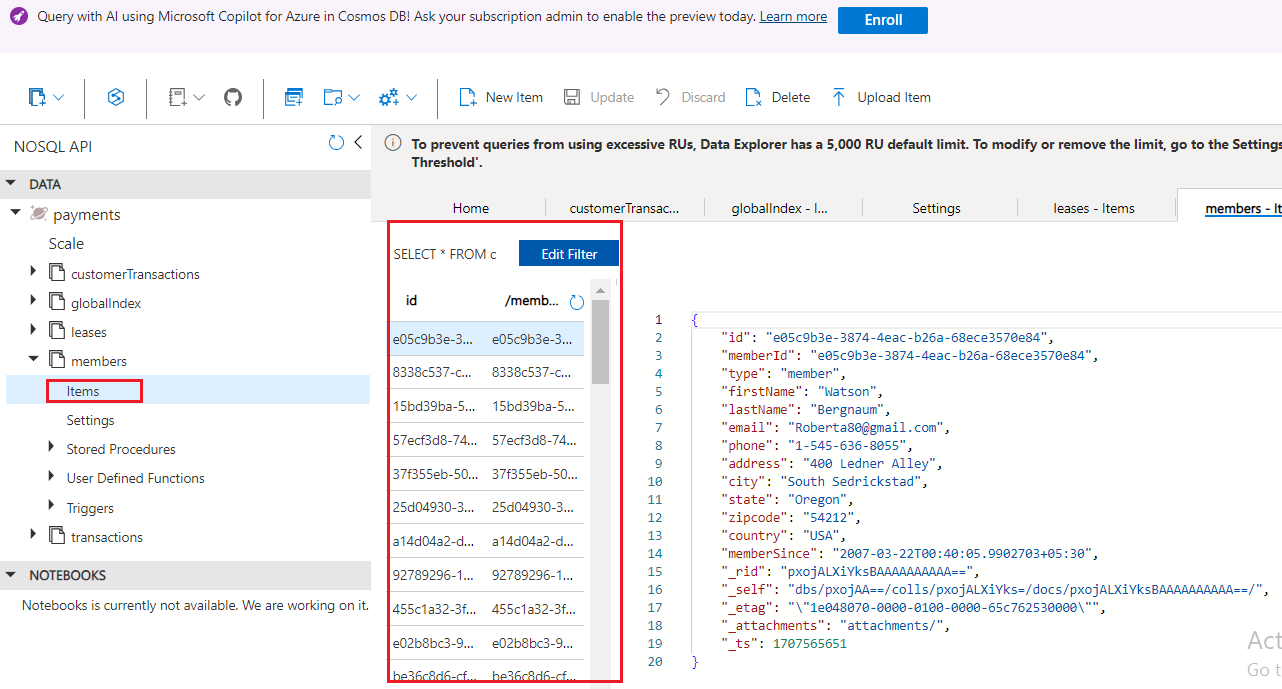
1. Verify that **globalindex Items** have been loaded.

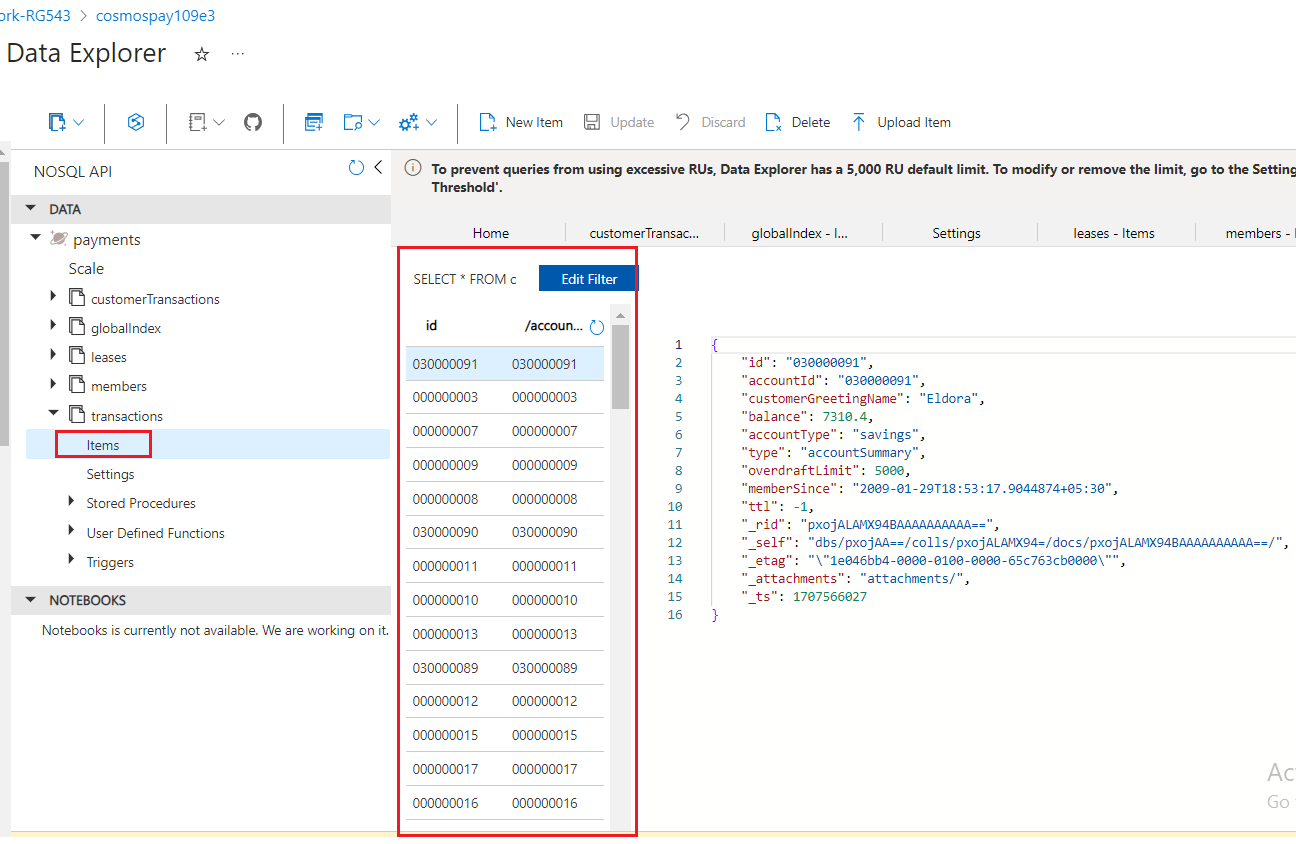


1. Now verify that **leases Items** have been loaded.

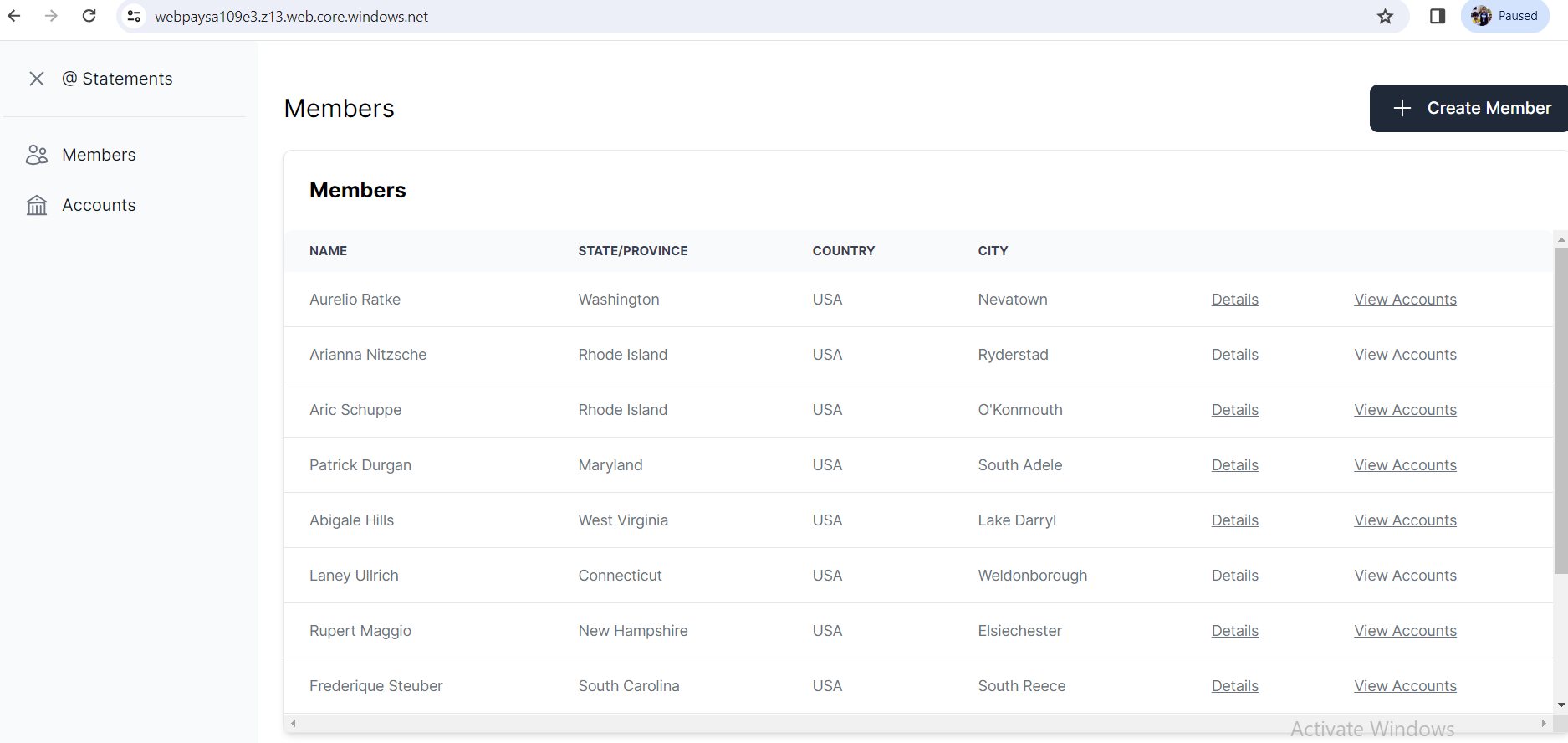


1. Verify that **members Items** and **transactions Items** have been loaded.

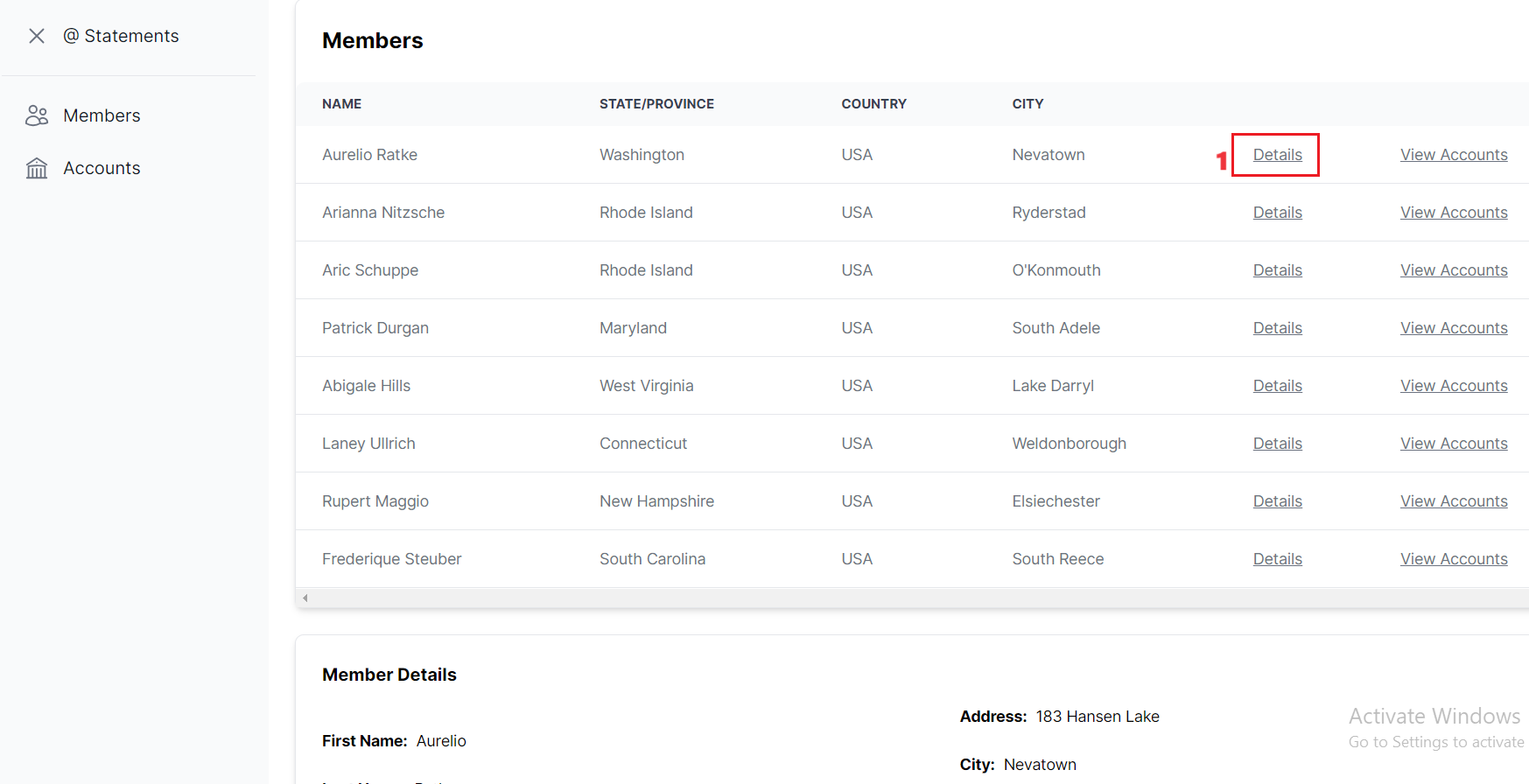




1. Open the **Static URL** (**(**The URL that you have saved in your notepad in the **Lab 1)** it into a browser.



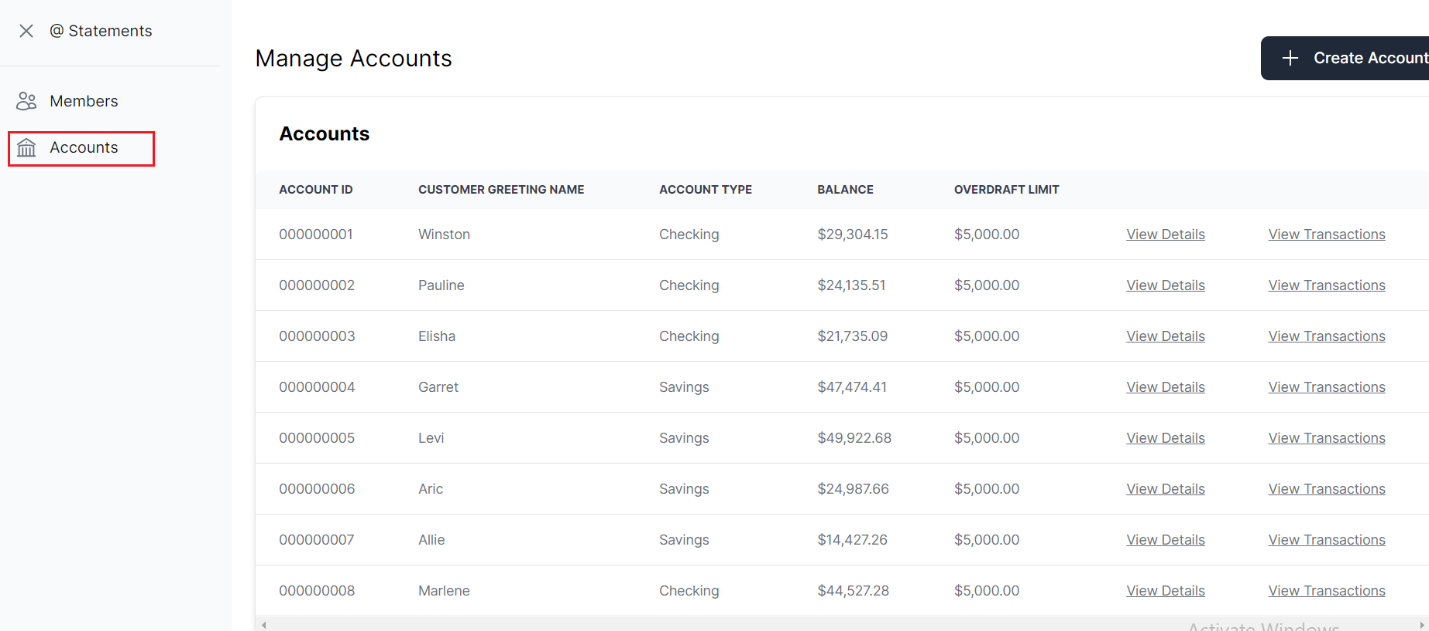
1. Click on the **Details** link of any row and it will display the all the details of the member in the lower part of the screen.

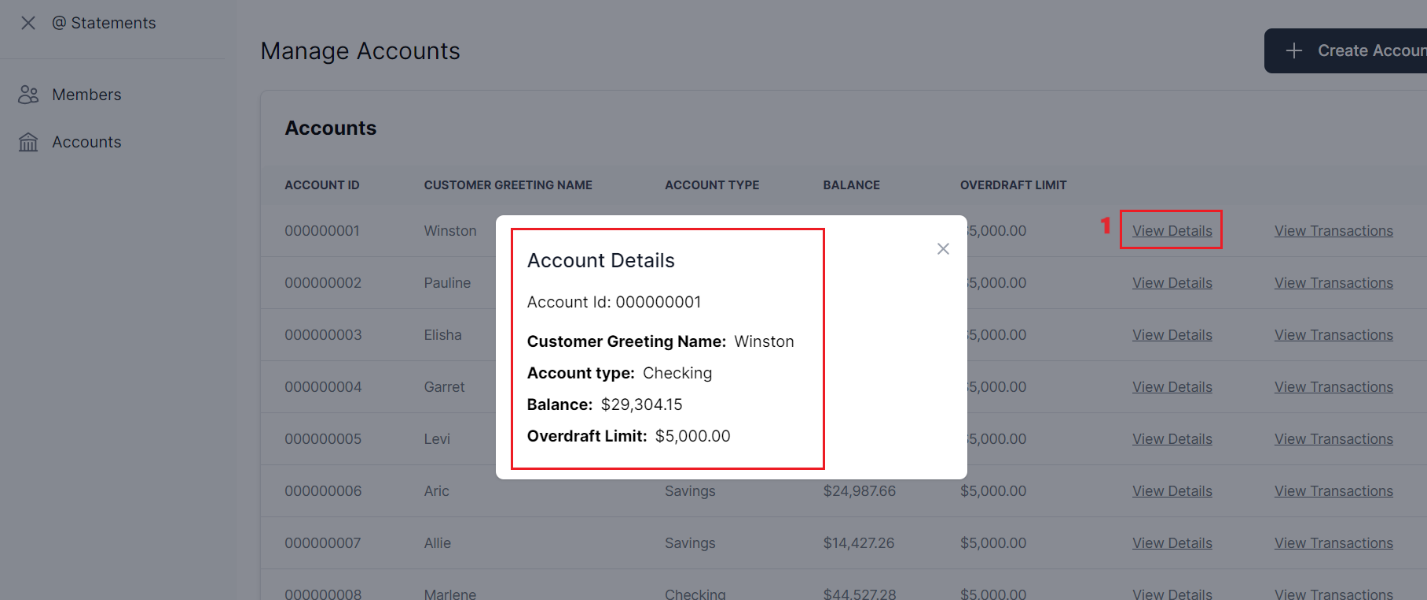


A screenshot of a computer

Description automatically generated

1. Click on **Accounts** from left navigation pane. Click on **view Details** of any record and verify the account details





**Summary**

In this lab, you have completed the loading of the data that generates customer and transaction data and verified loaded data in the CosmosDB.