



Microsoft Graph Reports API

Module Tutorial & Setup

How to use the Microsoft Graph Reports API module in your own Azure Synapse workspace

1. Create a linked service to Graph Reports API

- In your Azure, first go to AAD -> App registrations -> New registration and create a new app registration specifically for accessing the Graph Reports API from Synapse.
- Then click “API Permission” -> “Add Permission” and select Microsoft Graph -> select Application Permissions -> choose the “User.Read.All” permission, and click on the “Add permissions” button.
- Follow the same steps above, except now add the “Directory.Read.All” and “Reports.Read.All” permissions.
- After adding these permissions, you should see something similar to the following screenshot in your own Azure Portal:

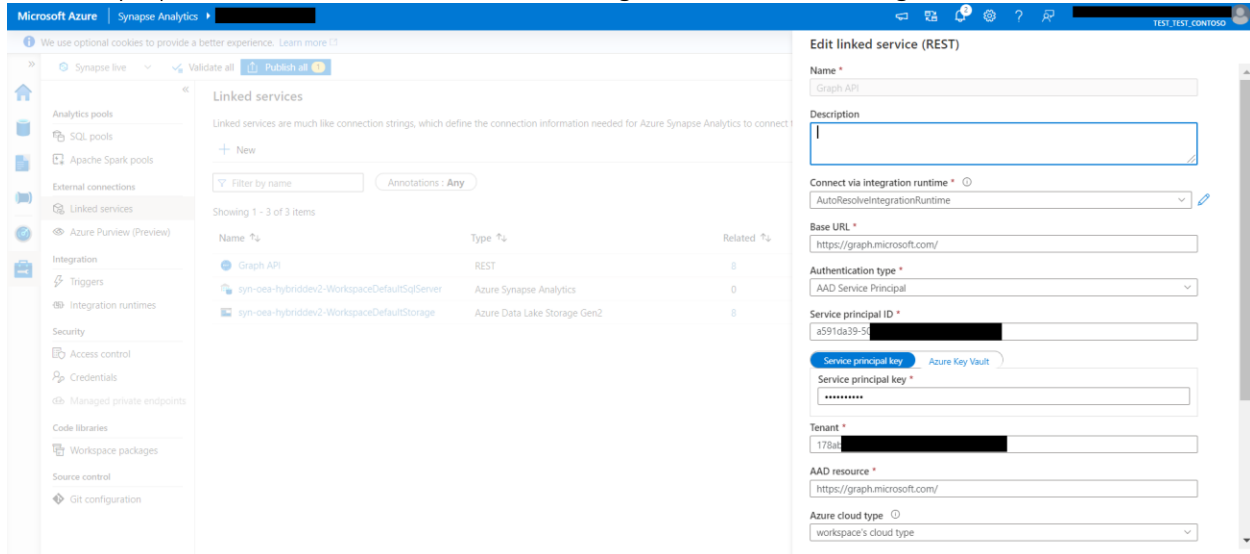
The screenshot shows the 'API permissions' page in the Azure Portal for an application named 'ms_graph_api'. The left sidebar contains navigation links: Overview, Quickstart, Integration assistant, Manage (Branding, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners, Roles and administrators | Preview, Manifest), and Support + Troubleshooting (Troubleshooting, New support request). The main content area shows a list of configured permissions for Microsoft Graph. A message at the top states: 'The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the actual status of the permission.' Below this, a table lists the permissions:

API / Permissions name	Type	Description	Admin consent required	Status
▼ Microsoft Graph (4)				
Directory.Read.All	Application	Read directory data	Yes	Granted for test_test_Co... ***
Reports.Read.All	Application	Read all usage reports	Yes	Granted for test_test_Co... ***
User.Read	Delegated	Sign in and read user profile	No	Granted for test_test_Co... ***
User.Read.All	Application	Read all users' full profiles	Yes	Granted for test_test_Co... ***

At the bottom of the table, a note says: 'To view and manage permissions and user consent, try Enterprise applications.'

- Click on the “Grant admin consent” link once these permissions have been successfully added, so that you can verify all permissions show a status of “Granted” under the status column.
- Now go to “certificates & secrets” and add a new client secret -> copy that value.
- Now in your Synapse Studio, go to “linked services” under “Manage”, and add a REST service for Graph Reports API -> select AAD Service Principal as the Authentication type.
- Under the base URL and AAD resource fields, type in: <https://graph.microsoft.com/>
- Under “Service principal key”, paste the value of the secret you copied previously.

- Under “Service principal ID”, enter the “Application (client) ID” of the app registration created in the previous step (go to the Overview section of the app registration). After doing so, your Synapse environment should look something similar to the following screenshot:

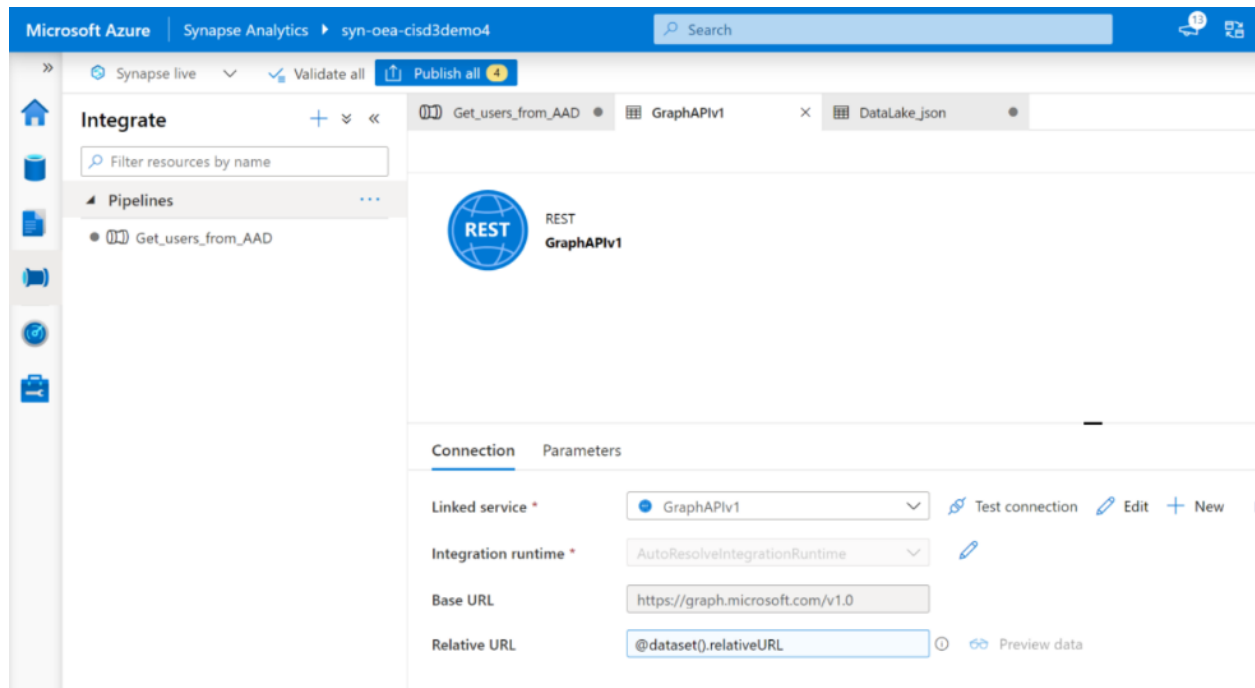


2. Notes

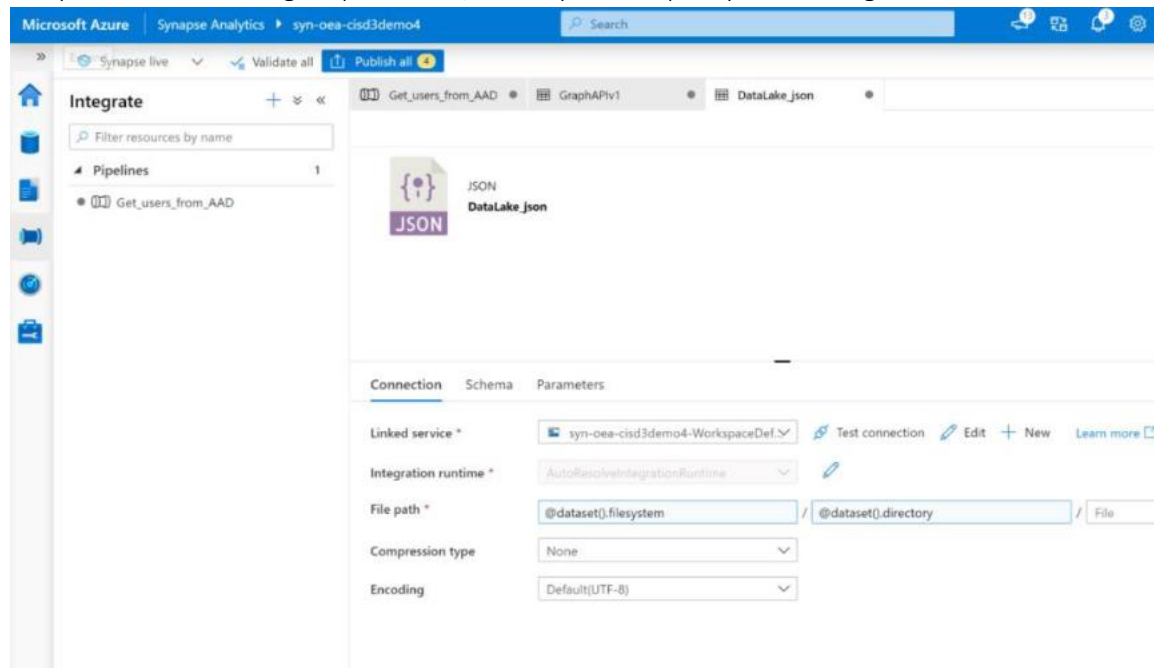
- Make sure that you're Synapse environment is working within the OEA framework (i.e. the initial setup).
- Before moving forward in this tutorial, you can choose to either proceed with step 3 or step 4 (step 3 explains the process of creating a Graph Reports API pipeline from scratch; step 4 uses the sample datasets and pipeline template provided in this module). You should start with step 4 to familiarize yourself with the pipeline, and easily use the sample notebook.
 - If you want to use your own Graph Reports API data, you can start with step 3. To use the notebook provided, you will have to make sure that the pipeline is landing data in stage1np under a folder "GraphAPI" -> the JSON files being landed in this folder should be "users.json", "m365_app_user_detail.json", and "teams_activity_user_details.json" in order to run the notebook seamlessly (the queries used for these three files can be found in the GitHub dataset folder of this module).
 - As of September 1st, 2021, you will also have to sign in to <https://admin.microsoft.com/AdminPortal> to use your own data (due to the default hashing of userPrincipalNames); go to the admin center and login -> go to Settings -> Org Settings -> "Services" page. Select "Reports". Uncheck the statement "Display concealed user, group, and site names in all reports," and then save your changes. [Activity Reports in the Microsoft 365 admin center - Microsoft 365 admin | Microsoft Docs](#).

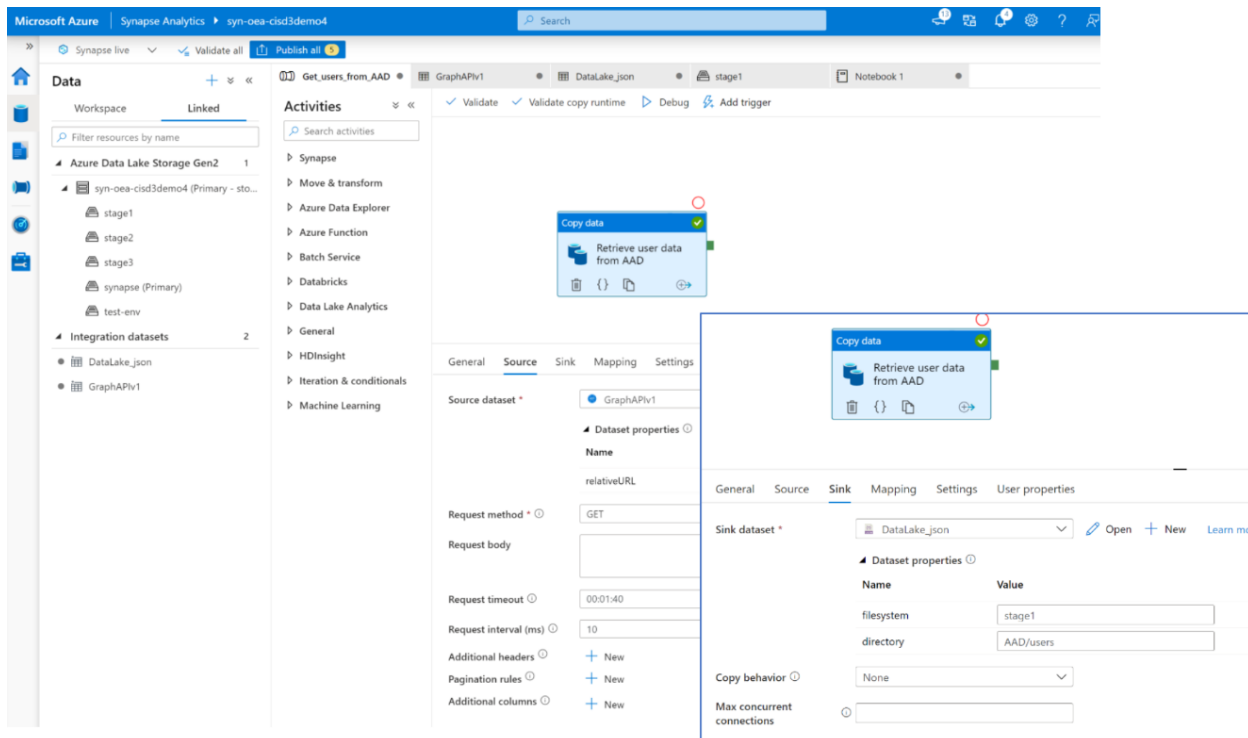
3. Creating the Graph Reports API pipeline from scratch within your Synapse environment

- Create a new pipeline under “Integrate” -> Add “Copy data” activity -> In “Source” tab, create a new REST dataset, referring to the Graph Reports API linked service you created.



- Select “Azure Blob Storage” and click “Continue”. In the “Select format” prompt, select “JSON” format -> In the “Set properties” prompt, select your Azure Storage Account -> For “File path”, you should use: stage1np/XXX/YYY, where you can specify the landing of this JSON data.

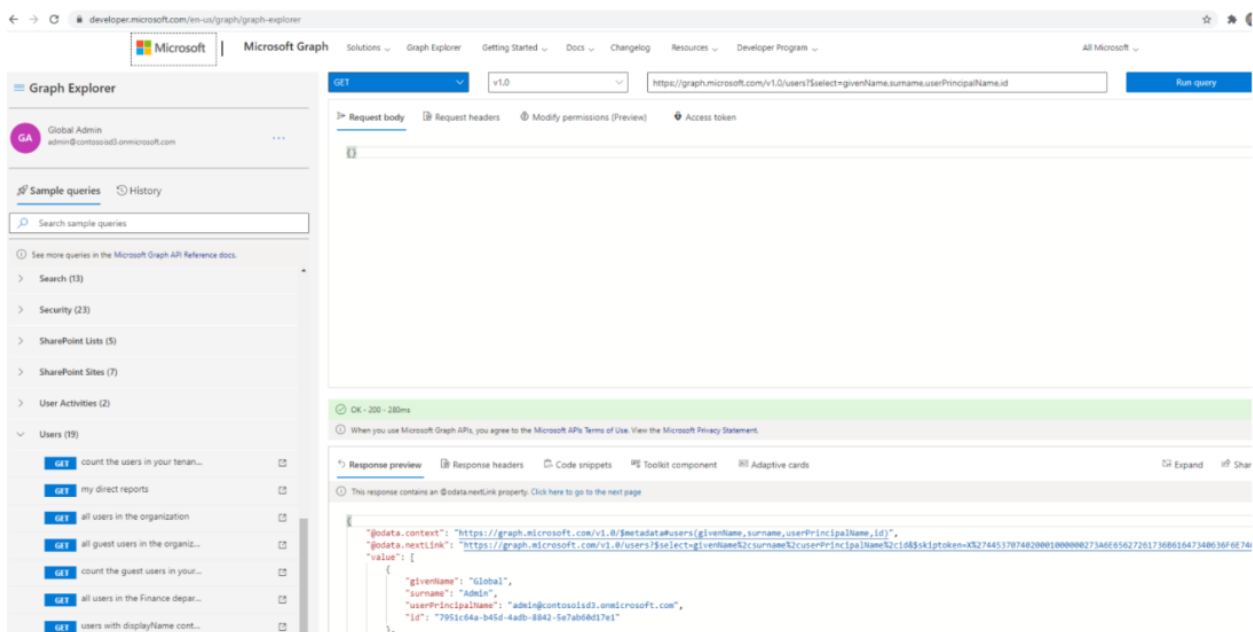




- You can reduce the data returned by selecting specific attributes in the relativeURL, like this: `beta/users?$select=givenName,surname,userPrincipalName,id`

Make sure you put this relativeURL in the “dynamic contents” section in order to retrieve the data from that query.

- Use the Graph Explorer to try it out, and see the data the query is pulling, found at <https://developer.microsoft.com/en-us/graph/graph-explorer>



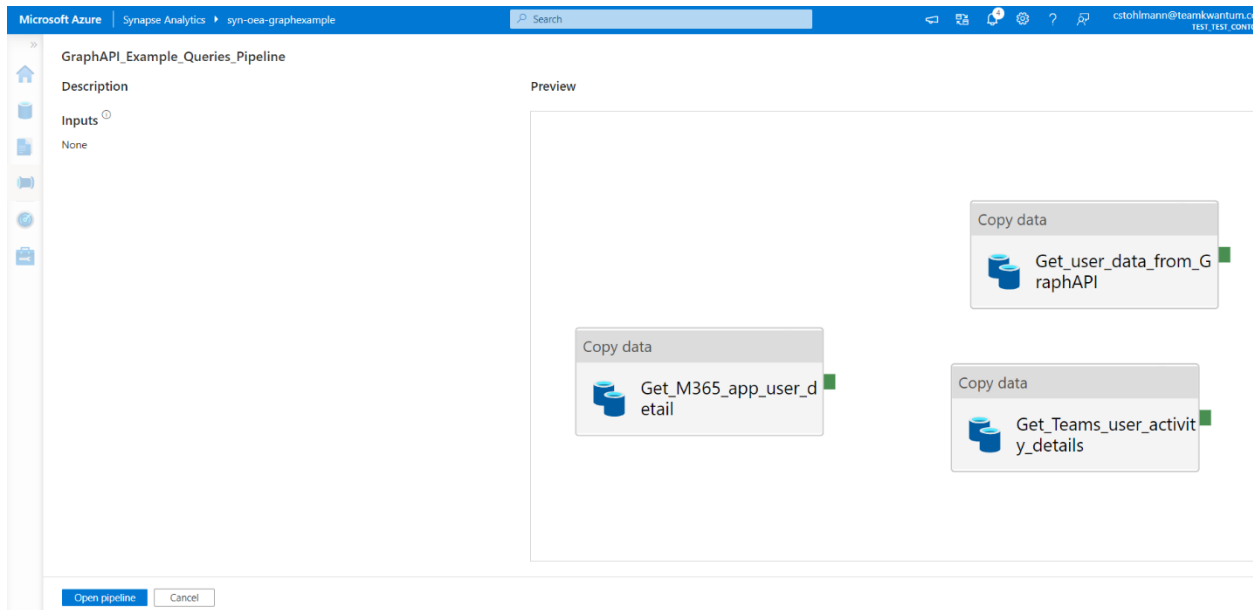
- If you run the pipeline and get a failure because of lack of access like this:

```
{ "errorCode": "2200", "message": "Failure happened on 'Source' side.  
ErrorCode=RestSourceCallFailed,'Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=The  
HttpStatusCode 403 indicates failure.\nRequest URL: https://graph.microsoft.com/v1.0/users?\$top=3\nResponse  
payload:{\"error\":{\"code\": \"Authorization_RequestDenied\", \"message\": \"Insufficient privileges to complete the  
operation.\", \"innerError\": {\"date\": \"2021-05-28T19:31:00\", \"request-id\": \"abc221e5-a4d1-4845-8a5d-  
c26353b99af3\", \"client-request-id\": \"abc221e5-a4d1-4845-8a5d-  
c26353b99af3\"}}},Source=Microsoft.DataTransfer.ClientLibrary,\"\", \"failureType\": \"UserError\", \"target\": \"Copy data1\",  
\"details\": [] }
```

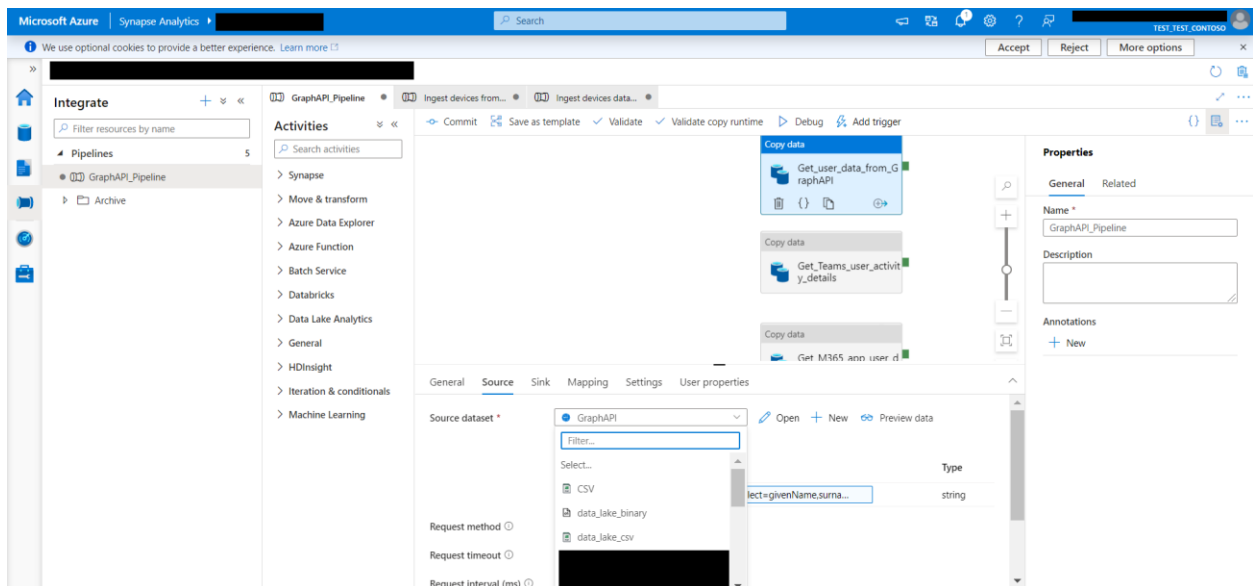
You'll need to double check the app registration you created and the API permissions to it.

4. Using the pipeline template in Synapse for the Graph Reports API

- Navigate to the cloned pipeline template zip file titled “GraphAPI_Pipeline” in “pipelines” from this module.
- Next, under “Integrate” and select “Add resource” -> then select “Import from pipeline template” -> select the pipeline zip you cloned to your local environment.



- Now you will just have to connect the REST data source to your linked service. As in the screenshot below, you should navigate to “Source” once loading in the template -> click on “Open”.
- Then, for the “GraphAPI” REST source uploaded, you will connect the newly created linked service from the “Linked service” drop-down. Make sure to publish/commit these changes.



- After you've done this for each activity, you can manually trigger the pipeline to run.

- Now you need to process this user data and write it to stage 2 in a more usable way, and create a spark database that can be easily queried from Power BI.

6. Execute the Notebook

- Navigate to the cloned notebook from this Graph Reports API module -> import the notebook under “Develop”
- Make sure you change the code in the code block “Provision storage accounts”, so that you’re using your own storage account name.
- Attach to your spark pool and now run this notebook:

The screenshot shows the Microsoft Azure Synapse Analytics interface. The top bar indicates the environment is 'syn-oea-graphexample'. The left sidebar shows the 'Develop' environment with a search bar and a list of notebooks, including 'GraphAPI_module_example_notebook'. The main area displays the notebook content, which includes a title 'Graph API Module Example Notebook', a description 'This notebook creates 3 tables (users, m365_app_user_detail and teams_activity_user_details) into a new Spark database called graphapi.', and a code block titled 'Provision storage accounts'. The code block contains Python code for setting up Spark SQL types, functions, and window functions, and for defining storage accounts and test environments. The code is as follows:

```
1 from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, ArrayType
2 from pyspark.sql.functions import *
3 from pyspark.sql.window import Window
4
5
6 # data lake and container information
7 storage_account = 'stoeahybriddev2'
8 use_test_env = False
9
10 if use_test_env:
11     stage1np = 'abfss://test-env@' + storage_account + '.dfs.core.windows.net/stage1np'
12     stage2np = 'abfss://test-env@' + storage_account + '.dfs.core.windows.net/stage2np'
13 else:
14     stage1np = 'abfss://stage1np@' + storage_account + '.dfs.core.windows.net'
15     stage2np = 'abfss://stage2np@' + storage_account + '.dfs.core.windows.net'
```

Below the code block, there is a status bar indicating the execution of the code: '[1] ✓ - Apache Spark session started in 2 min 23 sec 906 ms. Command executed in 156 ms on 1:11:30 PM, 10/06/21'.

- Running this notebook creates two spark databases: stage 2np and stage 2p. The stage 2p data is used within the scope of the PowerBI dashboard provided for this module.

7. View Data in Stage 2np and Stage 2p

- Now you're able to query the data from the newly created spark databases.

The screenshot displays the Microsoft Azure Synapse Analytics workspace. On the left, the 'Data' pane shows a tree view of resources, including 's2p_graphapi (Spark)' and its tables: 'm365_app_user_detail', 'teams_activity_user_details', and 'users'. The main query editor shows a SQL script named 'SQL script 2' with the following query:

```
1 SELECT TOP (100) [surname]
2 , [givenName]
3 , [userPrincipalName_pseudonym]
4 , [id]
5 FROM [s2p_graphapi].[dbo].[users]
```

The 'Results' pane shows the output of the query, displaying a table with four columns: 'surname', 'givenName', 'userPrincipalName_pseudonym', and 'id'. The table contains 10 rows of data.

surname	givenName	userPrincipalName_pseudonym	id
*	*	6a88b5a86c10257521c9413dd6...	0d444980-3123-46f4-85d3-649e4619ea5
*	*	e110dd065c9b9b47a4a1141681...	c19234e7-a816-41af-9524-4be4092c559
*	*	1a13cbd3535d389ad59bb9ffa33...	7a2ca37f-d1f3-4be8-8abe-8711d4eea8d6
*	*	0f622cd93f5bbee426ba3b4b8ee...	154f2b2c-9e73-4ecc-917e-06c0e440dea15
*	*	46f098301c98ceaf66b35a50e5...	dbc10b91-8d2c-4cf2-ac08-985ad3b4d6b4
*	*	c02ef6235e16cfeadd64969de05...	3f901837-788d-407b-b6e5-3dd9018011fe
*	*	6cb0324d1e8d57b4c720401b523...	8ec9eba7-aab5-4f79-b798-47320dbef0d
*	*	cf901e5a50a682ca98c48b567891...	ce02f59b-7ab4-426d-be15-1831bef543a8
*	*	4e4dc17aed0f5fe61b3628935698...	1e9fb7e6-a12b-43d9-bf5a-c0c32226db8
*	*	522478eb754bc718b4c0fcb4361...	ebddec9f-bfd9-4a32-bc62-07554666ec00

The 'Properties' pane on the right shows the details of the query script, including its name 'SQL script 2', description, type 'sql script', and size '111 bytes'. The 'Results settings per query' are set to 'First 5000 rows (default)'.