

## Context

|                           |         |
|---------------------------|---------|
| Create SQL Server         | 2 – 4   |
| Create SQL database       | 4 – 6   |
| Create a Linked Services  | 6 – 7   |
| Create Dataset            | 8 – 9   |
| If Condition Activity     | 9 – 14  |
| For Each Activity         | 15 – 18 |
| Filter Activity           | 18 – 21 |
| Set Variable Activity     | 21 – 23 |
| Append Variable           | 23 – 25 |
| Until Activity            | 26 – 30 |
| Web Activity              | 31 – 33 |
| Switch Activity           | 33 – 38 |
| Lookup Activity           | 39 – 45 |
| Wait Activity             | 46 – 47 |
| Script Activity           | 48 – 51 |
| Stored Procedure Activity | 52 – 54 |

# Create SQL Server

1. On the Home page click on SQL servers.



2. Click on Create SQL Server as shown below.

3. Under basic give the below properties as shown below.

Basics Networking Additional settings Tags Review + create

SQL database server is a logical container for managing databases and elastic pools. Complete the Basic tab, then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

**Project details**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*  Resource group \*  [Create new](#)

**Server details**

Enter required settings for this server, including providing a name and location.

Server name \*  .database.windows.net

Location \*

4. Scroll down and give the login name and password then click on the Next option.

**Authentication**

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Azure AD authentication [Learn more](#) using an existing Azure AD user, group, or application as Azure AD admin [Learn more](#), or select both SQL and Azure AD authentication.

Authentication method  Use only Azure Active Directory (Azure AD) authentication  Use both SQL and Azure AD authentication  Use SQL authentication

Server admin login \*

Password \*

Confirm password \*

[Review + create](#) | [Next : Networking >](#)

5. Set the below properties and click on Review+Create

## Create SQL Database Server

Microsoft

Basics Networking Additional settings Tags Review + create

Configure networking access for your server.

### Firewall rules

Allow Azure services and resources to access this server

Yes  No

[Review + create](#) [< Previous](#) [Next : Additional settings >](#)

6. Next click on Create.

Basics Networking Additional settings Tags Review + create

### Product details

SQL Database Server  
by Microsoft  
[Terms of use](#) | [Privacy policy](#)

**Estimated cost per month**  
No additional charges

### Terms

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Market frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and third-party offerings. For additional details see [Azure Marketplace Terms](#).

### Basics

|                       |                           |
|-----------------------|---------------------------|
| Subscription          | Microsoft Partner Network |
| Resource group        | ADFResource               |
| Server name           | adfsqservertrail          |
| Authentication method | SQL authentication        |
| Server admin login    | adminsql                  |
| Location              | East US                   |

### Networking

[Create](#) [< Previous](#) [Download a template for automation](#)

7. After some time you will see a screen like below.

The screenshot shows the Microsoft Azure Deployment Overview page for a deployment named "Microsoft.SQLServer.createServer\_fca55494a9eb4c27a5fcf5aee35d70f". The status is "Your deployment is complete". Deployment details include: Deployment name: Microsoft.SQLServer.createServer\_fca55494a9eb4c27a5fcf5aee35d70f, Start time: 7/20/2023, 1:25:21 PM, Subscription: Microsoft Partner Network, Resource group: ADFResource, Correlation ID: 5c1fa646-8fc8-4155-9ebf-9db2786e9654. Navigation links include Search, Delete, Cancel, Redeploy, Download, Refresh, Overview, Inputs, Outputs, and Template.

## Create SQL database

1. On the Home page click on SQL database.

The screenshot shows the Microsoft Azure Home page with the "Azure services" navigation bar. It includes icons for Create a resource, SQL databases, Storage accounts, Data factories, App registrations, Quickstart center, Virtual machines, App Services, Azure Cosmos DB, and More services.

2. Next click on create sql database.

The screenshot shows the Microsoft Azure SQL databases blade. It displays a message: "No SQL databases to display. Try changing or clearing your filters." Below this is a "Create SQL database" button. The blade also includes filters for Name, Server, Replica type, Pricing tier, Location, and Subscription.

3. Give the Database name as sqldb and select the server that we created before then set the properties as shown below. And click on Review+Create.

The screenshot shows the "Create SQL Database" wizard. In the "Database details" step, the "Database name" is set to "sqldb" and the "Server" is set to "adfsqlservertrail (East US)". Other options include "Want to use SQL elastic pool?" (No selected), "Workload environment" (Development selected), and a note about default settings for Development workloads. Buttons at the bottom include "Review + create" and "Next : Networking >".

4. Next click on Create.
5. After some time you will see the below screen as shown.

The screenshot shows the Microsoft Azure portal's 'Deployment' overview page. The title bar reads 'Microsoft.SQLDatabase.newDatabaseExistingServer\_02f8fa5d79fd47e5 | Overview'. The main content area displays a green checkmark icon and the message 'Your deployment is complete'. Below this, deployment details are listed: Deployment name: Microsoft.SQLDatabase.newDatabaseExistingServer\_0..., Start time: 7/20/2023, 1:31:52 PM, Subscription: Microsoft Partner Network, Correlation ID: a1e52811-272f-4ad2-af61-b2d67b4ef720, and Resource group: ADFResource.

6. Now click on the SQL Database that we created just now.
7. Go to Query editor, give login credentials that we created in the sql server click on Ok.
8. If it asks to allow access to an IP address allow it.

The screenshot shows the 'Query editor (preview)' interface for the 'sqldatabase (sampleadmins1/sqldatabase)' database. The top navigation bar includes 'Home > sqldatabase (sampleadmins1/sqldatabase) | Query editor (preview)'. The left sidebar lists database management tasks like 'Overview', 'Activity log', 'Tags', 'Diagnose and solve problems', 'Getting started', and 'Query editor (preview)'. The main area displays a 'Welcome to SQL Database Query Editor' message and two authentication options: 'SQL server authentication' (Login: admins1, Password: [redacted]) and 'Active Directory authentication' (Continue as Lacchi.Balaji@gisul.co.in). A large 'OK' button is at the bottom right.

9. A Query editor will open so that we can create tables and export tables also.
10. I have created a table using sql query.

The screenshot shows the 'Query editor (preview)' interface for the 'sqldatabase (admins1)' database. The left sidebar shows 'Overview', 'Activity log', 'Tags', 'Diagnose and solve problems', 'Getting started', and 'Query editor (preview)'. The main area shows a 'Query 1' window with the following SQL code:

```

CREATE TABLE TotalSale (
    id int NOT NULL,
    SalePersonName varchar(100) NULL,
    ProductName varchar(100) NULL,
    ItemsSold int NULL,
    SoldPrice int NULL,
    SoldDate Date,
    Country varchar(100) NULL,
    Region varchar(100) NULL
)
  
```

The 'Results' tab is selected at the bottom.

11. Below is the table I have created.

The screenshot shows the Azure Data Studio interface. On the left, there's a tree view of the database schema under 'sqldatabase (adminsql1)'. It shows a 'Tables' node expanded, revealing a 'dbo.TotalSale' table with columns: id, SalePersonFName, SalePersonLName, ProductName, ItemsSold, SoldPrice, Country, and Region. Below 'Tables' are 'Views' and 'Stored Procedures'. On the right, a 'Query 1' tab is open with the following SQL query:

```
SELECT * FROM dbo.TotalSale
```

The results pane displays 10 rows of data from the 'TotalSale' table:

| id | SalePersonFName | SalePersonLName | ProductName | ItemsSold | SoldPrice | Country  | Region        |
|----|-----------------|-----------------|-------------|-----------|-----------|----------|---------------|
| 1  | Aamir           | Shahzad         | TV          | 1         | 700       | USA      | North America |
| 2  | M               | Raza            | Cell Phone  | 2         | 800       | USA      | North America |
| 3  | Christy         | Ladson          | TV          | 3         | 1600      | USA      | North America |
| 4  | John            | Rivers          | Laptop      | 5         | 2400      | USA      | North America |
| 5  | Najaf           | Ali             | Computer    | 1         | 300       | Pakistan | Asia          |
| 6  | Sukhjeet        | Singh           | TV          | 2         | 900       | India    | Asia          |
| 7  | Chirag          | Patel           | Cell Phone  | 5         | 1500      | India    | Asia          |
| 8  | Aleena          | Aman            | Laptop      | 2         | 800       | Pakistan | Asia          |
| 9  | Petra           | Henry           | TV          | 10        | 5000      | France   | Europe        |
| 10 | Rita            | Roger           | Laptop      | 7         | 2100      | France   | Europe        |

## Create a Linked Services

1. Go to Azure Data Factory.
2. Go to the Manage tab and select Linked Services then click on the plus symbol to create.

The screenshot shows the 'Linked services' page in the Azure Data Factory portal. The left sidebar has 'General' selected. The main area shows a table with one item:

| Name                 | Type  |
|----------------------|-------|
| AzureBlobStorage_txt | Azure |

3. Select the SQL Azure Database and click on Continue.

The screenshot shows the 'Data store' selection screen in the Azure Data Factory portal. The left sidebar has 'Compute' selected. The main area shows a grid of database options:

| All                           | Azure                     | Database                            | File | Generic protocol | NoSQL | Services and apps |
|-------------------------------|---------------------------|-------------------------------------|------|------------------|-------|-------------------|
| Amazon RDS for SQL Server     | Azure Cosmos DB for NoSQL | Azure Database for MySQL            |      |                  |       |                   |
| Azure Database for PostgreSQL | Azure SQL Database        | Azure SQL Database Managed Instance |      |                  |       |                   |
| MySQL                         | Oracle                    | SQL Server                          |      |                  |       |                   |

At the bottom, there are 'Continue' and 'Cancel' buttons.

4. Give the Name and Runtime.

New linked service

Azure SQL Database [Learn more](#)

Name \*

Description

Connect via integration runtime \* ⓘ

AutoResolveIntegrationRuntime

5. And give the below properties and test the connection then click on create.

Connect via integration runtime

AutoResolveIntegrationRuntime

**Connection string** **Azure Key Vault**

Account selection method ⓘ

From Azure subscription  Enter manually

Azure subscription

Select all

Server name \*

sampleadmins1sql

Database name \*

sqldatabase

Authentication type \*

SQL authentication

User name \*

admins1sql1

Add dynamic content [Alt+Shift+D]

**Password** **Azure Key Vault**

Password \*

\*\*\*\*\*

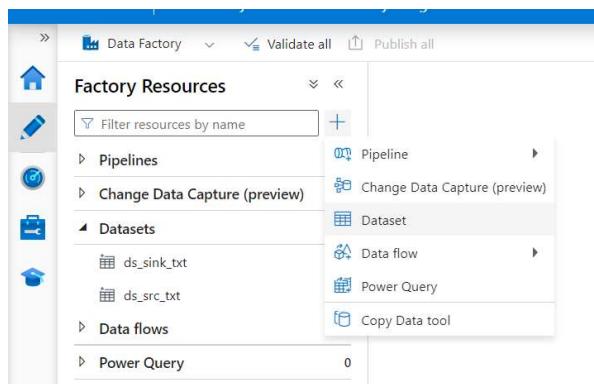
 Connection successful

 Test connection

Create Back Cancel

# Create Dataset

1. Go to the Author tab and click on Plus symbol then click on Dataset.

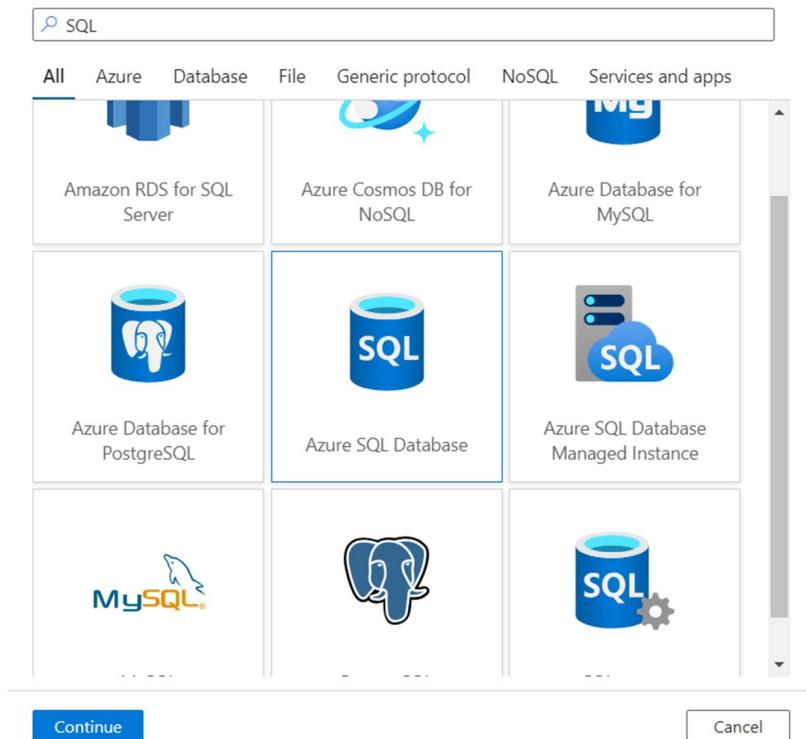


2. Select Azure SQL Database and click on Continue.

## New dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store



SQL

All Azure Database File Generic protocol NoSQL Services and apps

|                               |                           |                                     |
|-------------------------------|---------------------------|-------------------------------------|
| Amazon RDS for SQL Server     | Azure Cosmos DB for NoSQL | Azure Database for MySQL            |
| Azure Database for PostgreSQL | Azure SQL Database        | Azure SQL Database Managed Instance |
| MySQL                         | Oracle                    | Azure Synapse Analytics             |

Continue Cancel

- Give the Name and select the linked service and table that we created before. Then check the none option and click on Ok.

#### Set properties

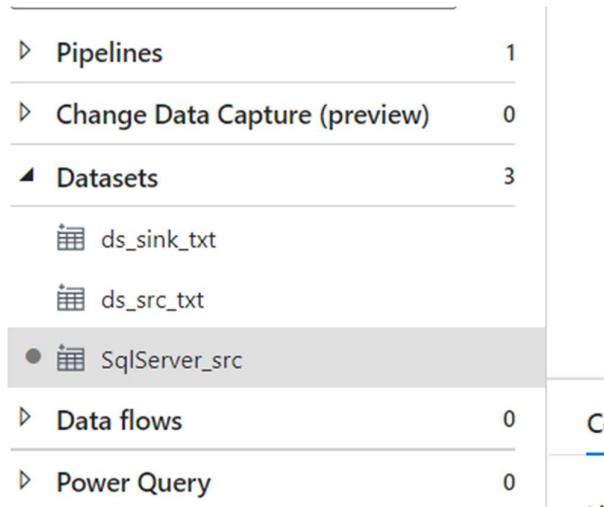
Name  
SqlServer\_src

Linked service \*  
AzureSqlDatabase

Table name  
dbo.TotalSale

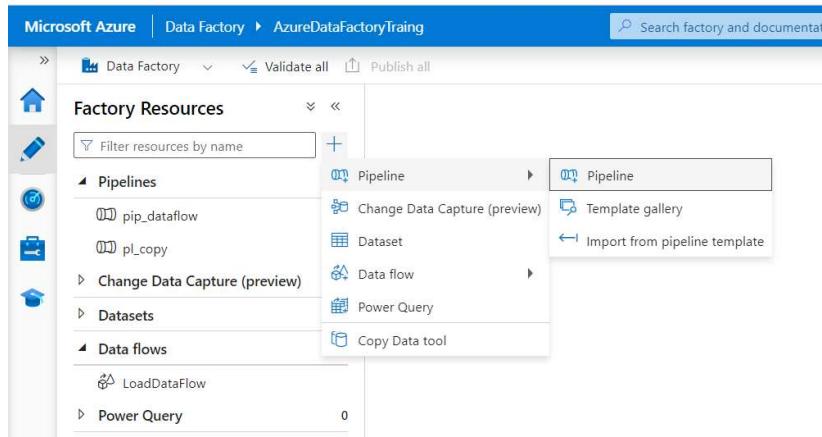
Import schema  
 From connection/store  None

- Under the dataset you can see our sql server dataset as shown below.



## If Condition Activity

- Go to Azure Data Factory and create a Pipeline.



2. Under the Parameter create three parameters as shown below.

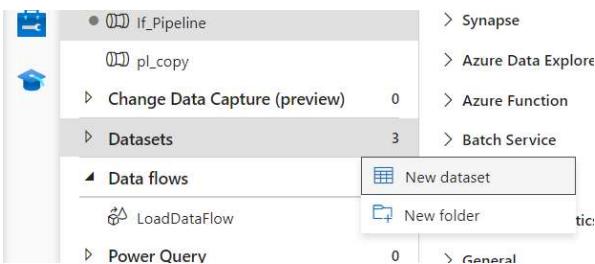
Parameters    Variables    Settings    Output

---

**New** | **Delete**

| Name          | Type   | Default value |
|---------------|--------|---------------|
| copytoOutput1 | String | true          |
| Output1Folder | String | Output1       |
| Output2Folder | String | Output2       |

3. Now create a Dataset for the destination.



4. Select Azure Blob Storage and click on continue. Select the Delimited text and click on continue.  
 5. Give the Name, select the Linked Service, and under the file path give the container name that we created in the previous session. Then check on the None option.

## Set properties

Name

Linked service \*



File path

 /  / 


First row as header



Import schema

From connection/store     From sample file     None

6. Under Parameter create a below parameter.

Connection Schema Parameters

New Delete

| Name         | Type   | Default value |
|--------------|--------|---------------|
| outputfolder | String | Value         |

7. Under Connection, in the Directory give the below expression.

Exp: @dataset().outputfolder

Connection Schema Parameters

Linked service \* AzureBlobStorage\_txt

Test connection Edit New Learn more

File path \* f-demo / @dataset().outputfolder / File name

Compression type None

Column delimiter Comma (,)

8. Now go to pipeline drag and drop the If condition activity as shown below.

If\_Pipeline

If\_Conditon\_dataset

Activities

if

Iteration & conditionals

If Condition

If Condition1

True  
No activities

False  
No activities

General Activities (0) User properties

Name \* If Condition1

9. Under Activity, in the expression section give the below expression.

Exp: @bool(pipeline().parameters.copytoOutput1)

General Activities (0) User properties

Expression: @bool(pipeline().parameters.copytoOutput1)

| Case  | Activity      |
|-------|---------------|
| True  | No activities |
| False | No activities |

10. Next click on the True case pencil symbol.

11. Drag and drop the Copy data as shown below.

If\_Pipeline If\_Conditon\_dataset

Activities

cop

Move & transform

Copy data

If\_Pipeline > If Condition1 > True activities

Copy data1

12. Under the Source select the source dataset that we created in the previous session.

General Source Sink (1) Mapping Settings User properties

Source dataset: ds\_src\_txt

File path type: File path in dataset

Start time (UTC)

End time (UTC)

13. Under sink select the destination target that we created before.

14. Under the outputfolder value give the below expression.

Exp: @pipeline().parameters.Output1Folder

General Source Sink Mapping Settings User properties

Sink dataset: If\_Conditon\_dataset

Dataset properties

| Name         | Value                                |
|--------------|--------------------------------------|
| outputfolder | @pipeline().parameters.Output1Folder |

Copy behavior: None

15. Next go to pipeline and click on the False pencil symbol as shown below.

General Activities (1) User properties

Expression ⓘ @bool(pipeline().parameters.copyto...)

| Case  | Activity                 |
|-------|--------------------------|
| True  | Copy data1<br>1 Activity |
| False | No activities            |

16. Next drag and drop the copy data as shown below.

If\_Pipeline If\_Conditon\_dataset

Activities

cop

Move & transform

Copy data

If\_Pipeline > If Condition1 > False activities

Copy data2

17. Under the source select the source dataset.

General Source Sink Mapping Settings User properties

Source dataset \*

ds\_src\_txt

File path type

File path in dataset

Start time (UTC)

End time (UTC)

Filter by last modified ⓘ

18. Under sink select the destination target that we created before.

19. Under the outputfolder value give the below expression.

Exp: @pipeline().parameters.Output2Folder

General Source Sink Mapping Settings User properties

Sink dataset \*

If\_Conditon\_dataset

Dataset properties ⓘ

| Name         | Type   |
|--------------|--------|
| outputfolder | string |

Copy behavior ⓘ

None

20. Go to Pipeline validate and click on Debug.

21. Our pipeline execute successfully.

The screenshot shows the Azure Data Factory Pipeline validation interface. At the top, there are tabs for 'Validate' (which is checked), 'Debug' (highlighted in blue), and 'Add trigger'. Below these are sections for 'Activities' (listing 'Copy data' and 'Move & transform' options) and a detailed view of the 'If Condition' activity. The 'If Condition' activity has a 'True' branch containing a 'Copy data' activity named 'Copy data1'. The 'False' branch is currently empty. Below this, the 'Output' tab is selected, showing a table of pipeline run details. The table includes columns for Activity name, Status, Activity type, Run start, and Duration. One row is listed: 'Copy data1' with a status of 'Succeeded' (indicated by a green checkmark), run at 7/20/2023, 5:15:56 PM, and duration of 13s.

| Activity name | Status    | Activity type | Run start             | Duration |
|---------------|-----------|---------------|-----------------------|----------|
| Copy data1    | Succeeded | Copy data     | 7/20/2023, 5:15:56 PM | 13s      |

22. As we all know in the pipeline parameter we have given the value as true.

23. So, it means the data will be stored in the output 1 folder.

24. Go to your container you will see an Output1 folder created.

The screenshot shows the Azure Blob Storage container 'f-demo'. The left sidebar has sections for 'Overview', 'Diagnose and solve problems', and 'Access Control (IAM)'. The main area shows a search bar, upload, change access level, refresh, and delete buttons. It also displays the 'Authentication method' as 'Access key (Switch to Azure AD User Account)' and the 'Location' as 'f-demo'. A search bar for blobs by prefix is present. Below these, a table lists blobs in the container. Two entries are shown: 'Output' and 'Output1', both of which are modified on 7/20/2023, 5:16:07 PM.

| Name    | Modified              |
|---------|-----------------------|
| Output  | 7/20/2023, 5:16:07 PM |
| Output1 | 7/20/2023, 5:16:07 PM |

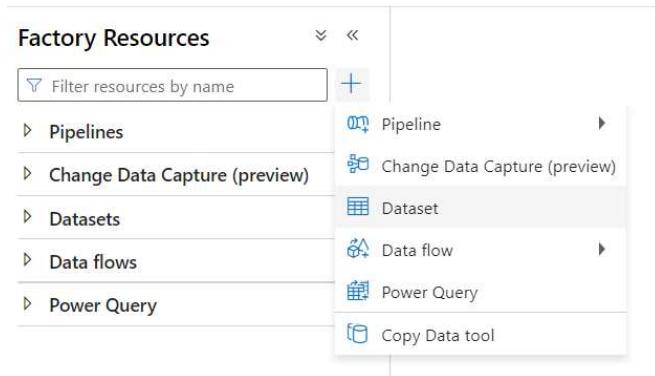
25. If you go inside the folder you will see a Source file.

The screenshot shows the contents of the 'Output1' folder. A 'Search' bar and a 'Add filter' button are at the top. Below is a table listing files. Two files are shown: 'Std\_details.txt' and '[]'. The 'Std\_details.txt' file was modified on 7/20/2023, 5:16:07 PM.

| Name            | Modified              |
|-----------------|-----------------------|
| Std_details.txt | 7/20/2023, 5:16:07 PM |
| [.]             |                       |

# For Each Activity

1. Create a Dataset.



2. Select Azure Blob Storage and click continue. Then select Delimited Text and click on Continue.
3. Give name, linked service, and container name then click ok.

## Set properties

Name

Linked service \*  
 

File path  
 /  /   

First row as header

Import schema  
 From connection/store  From sample file  None

4. Under parameters create a New Parameter as shown below.

| Parameters               |            |        |
|--------------------------|------------|--------|
| <input type="checkbox"/> | Name       | Type   |
| <input type="checkbox"/> | FolderName | String |

5. Under connection in the directory give the below expression.

Exp: @dataset().FolderName

Connection Parameters

Linked service \*  

Test connection  Edit

File path \*  /  /

Compression type

6. Create a Pipeline.

The screenshot shows the 'Factory Resources' pane on the left with 'Pipelines' selected. A modal dialog is open in the center, titled 'Pipeline'. It contains a list of options: 'Pipeline' (selected), 'Change Data Capture (preview)', 'Dataset', 'Data flow', 'Power Query', and 'Copy Data tool'. Below this is a link 'Import from pipeline template'.

7. Under parameter create a Parameter with the below value.

Value: ["output\_1","output\_2","output\_3"]

The screenshot shows the 'Parameters' tab. It has tabs for 'Parameters', 'Variables', 'Settings', and 'Output'. Under 'Parameters', there is a table:

| Name         | Type  | Default value                  |
|--------------|-------|--------------------------------|
| OutputFolder | Array | ["output_1","output_2","outp"] |

8. Drag and drop the ForEach Activity as shown below.

The screenshot shows the 'Activities' pane on the left with sections for 'Move & transform' (Copy data, Data flow) and 'Iteration & conditionals' (ForEach). A 'ForEach' activity is being added to the main canvas. The canvas shows a 'ForEach' activity with a child 'ForEach1' activity. The 'ForEach1' activity has a placeholder 'Activities' with a note 'No activities' and a plus sign to add more.

9. Under Settings give the below Expression.

Exp: @pipeline().parameters.OutputFolder

The screenshot shows the 'Settings' tab. It has tabs for 'General', 'Settings' (selected), 'Activities (0)', and 'User properties'. Under 'Settings', there are two sections: 'Sequential' (checked) and 'Items' (containing the expression '@pipeline().parameters.OutputFolder').

10. Under Activity click on the pencil symbol.

General   Settings   **Activities (0)**   User properties

---

| Case    | Activity      |
|---------|---------------|
| ForEach | No activities |



11. Drag and drop the copy data activity.

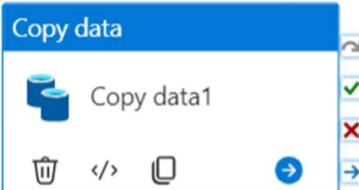
000 **ForEachAct\_pipeline** ●

✓ Validate   ✓ Validate copy runtime   ▶ Debug   ⚡ Add trigger

000 [ForEachAct\\_pipeline](#) >  **ForEach1**

**Copy data**

Copy data1



12. Under the source select the Employee Dataset.

General   **Source**   Sink   Mapping   Settings   User properties

---

Source dataset \*  ds\_src\_txt  Open  New  Preview data 

File path type  File path in dataset  Prefix  Wildcard file path  List of files ⓘ

13. Under Sink select the Binary Dataset that we created before and in the FolderName give the below expression.

Exp: @item()

General   Source   **Sink**   Mapping   Settings   User properties

---

Sink dataset \*  Delimited\_Dataset  Open  New 

▼ Dataset properties ⓘ

| Name       | Value   |
|------------|---------|
| FolderName | @item() |

14. Next go to Pipeline and validate it then click on debug.

15. Here Our Pipeline was executed successfully.

The screenshot shows the Azure Data Factory Pipeline run status page. At the top, there are three buttons: 'Validate' (green checkmark), 'Debug' (blue play icon), and 'Add trigger'. Below this is a 'ForEach' activity node with a green checkmark. Inside the node, there is a 'ForEach1' sub-node with its own green checkmark. Under 'Activities', there is a 'Copy data1' activity with a green checkmark. Below the pipeline run details, there is a table showing the execution history:

| Activity name | Activity status | Activity type | Run start              | Duration |
|---------------|-----------------|---------------|------------------------|----------|
| Copy data1    | Succeeded       | Copy data     | 7/28/2023, 10:20:46 AM | 8s       |
| Copy data1    | Succeeded       | Copy data     | 7/28/2023, 10:20:36 AM | 9s       |
| Copy data1    | Succeeded       | Copy data     | 7/28/2023, 10:20:25 AM | 10s      |
| ForEach1      | Succeeded       | ForEach       | 7/28/2023, 10:20:24 AM | 32s      |

16. Go to Destination and check whether the folders are created or not.



## Filter Activity

1. Here in the folder I have 2 Txt files and 1 CSV file. So, I need to get the txt files.

**Authentication method:** Access key ([Switch to Azure AD User Account](#))

**Location:** f-demo / Switch1

Search blobs by prefix (case-sensitive)

Add filter

| Name  | Modified                | Access tier    | Archive |
|---|-------------------------|----------------|---------|
| <input type="checkbox"/> ..                   |                         |                |         |
| <input type="checkbox"/> Customer_Details.csv | 7/28/2023, 10:33:14 ... | Hot (Inferred) |         |
| <input type="checkbox"/> EmpDetails.txt       | 7/28/2023, 10:33:24 ... | Hot (Inferred) |         |
| <input type="checkbox"/> Std_details.txt      | 7/21/2023, 10:25:48 ... | Hot (Inferred) |         |

2. Create a Dataset for this folder location and do not select any file.
3. Select Azure Blob Storage and click on continue. Then select Delimited Text and click on Continue.
4. Give the name, linked service, and container then folder.

Name  
Filter\_Dataset

Linked service \*  
AzureBlobStorage\_txt

File path  
f-demo / Switch1 / File name

First row as header

Import schema  
 From connection/store  From sample file  None

5. Create a pipeline.

The screenshot shows the 'Factory Resources' blade. In the center, there is a search bar with 'Filter\_Dataset' and a dropdown menu. Below the search bar, there is a list of resources: Pipelines, Change Data Capture (preview), Datasets, Data flows, and Power Query. A 'Pipeline' item is selected and highlighted in blue. A tooltip for 'Pipeline' shows options: Pipeline, Change Data Capture (preview), Template gallery, Import from pipeline template, and a copy data tool. To the left, there is a sidebar with a search bar and a list of categories: Pipelines, Change Data Capture (preview), Datasets, Data flows, and Power Query.

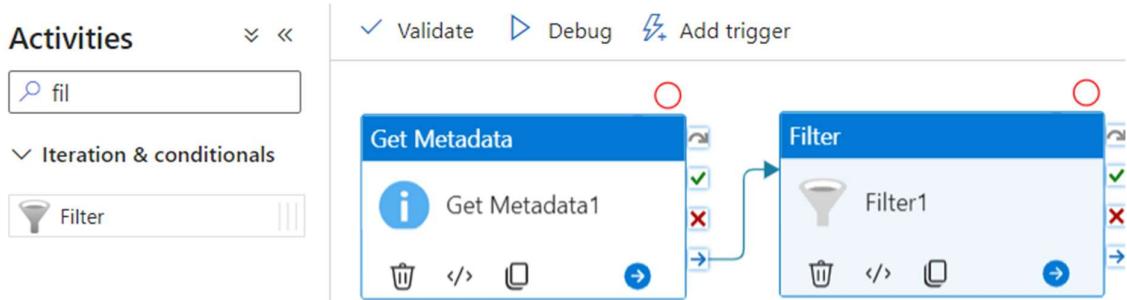
6. Drag and drop the Get Metadata.

The screenshot shows the 'Activities' blade. On the left, there is a search bar with 'me' and a list of activities under 'General': Get Metadata. On the right, there is a list of activities: Get Metadata, Change Data Capture, Pipeline, Data flow, Power Query, and Copy Data tool. The 'Get Metadata' activity is selected and highlighted in blue. A tooltip for 'Get Metadata' shows icons for edit, validate, debug, add trigger, and delete.

7. Under Settings select the Dataset that we created before and select the field list.

The screenshot shows the 'Settings' blade for the 'Get Metadata' activity. At the top, there are tabs: General, Settings (which is selected), and User properties. Below the tabs, there is a 'Dataset \*' dropdown set to 'Filter\_Dataset'. There are also 'Open', 'New', and 'Learn more' buttons. The 'Field list \*' section contains a 'New' button, a 'Delete' button, an 'Argument' checkbox, and a 'Child items' dropdown.

8. Drag and drop the filter activity and make the connection as shown below.



9. Under settings for the Item give the below expression.

Exp: @activity('Get Metadata1').output.childItems

10. For the Condition give the below expression.

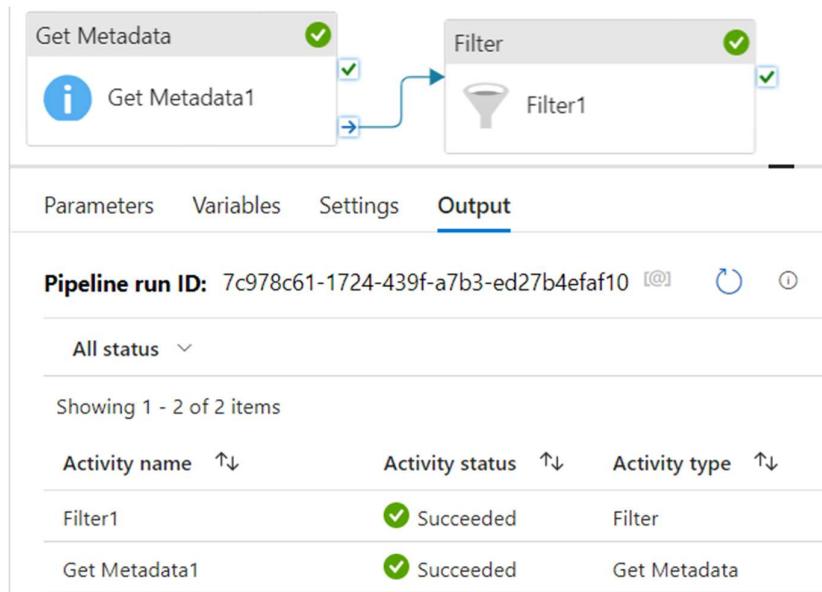
Exp: @endswith(item().name,'txt')

General    Settings    User properties

|           |  |
|-----------|--|
| Items     | @activity('Get Metadata1').output.chi... |
| Condition | @endswith(item().name,'txt')             |

11. Now validate the pipeline and click on debug.

12. Here Our Pipeline was executed successfully.



13. Now click on filter output and check the txt file names.

The screenshot shows the 'Output' section of a pipeline run. It displays the results of two tasks: 'Filter1' and 'Get Metadata1'. Both tasks are marked as 'Succeeded'. The 'Filter1' task has a 'Run start' date of 7/28/2023. The 'Get Metadata1' task also has a 'Run start' date of 7/28/2023. The output JSON shows that the 'Value' array contains two items, each representing a file: 'EmpDetails.txt' and 'Std\_details.txt', both of which are of type 'File'.

## Set Variable Activity

1. Create a Pipeline.

The screenshot shows the 'Factory Resources' blade in the Azure Data Factory portal. A context menu is open over the 'Pipelines' item, with 'Pipeline' selected. Other options in the menu include 'Change Data Capture (preview)', 'Dataset', 'Data flow', 'Power Query', and 'Copy Data tool'.

2. Create a parameter with the below value.

Value: Address: 221 M.G. Road Road. City: Kolkata. State: West Bengal

The screenshot shows the 'Parameters' blade. A new parameter is being created with the name 'addressLine', type 'String', and default value 'Address: 221 M.G. Road R'.

| Name        | Type   | Default value            |
|-------------|--------|--------------------------|
| addressLine | String | Address: 221 M.G. Road R |

3. Under Variables create a variable as shown below.

| Name     | Type   | Default value |
|----------|--------|---------------|
| cityName | String |               |

4. Drag and drop the set variable.

5. Under setting select the name and give the below expression.

```
Exp:@substring(pipeline().parameters.addressLine,add(indexof(pipeline().parameters.addressLine,'City:'), 6),sub(sub(indexof(pipeline().parameters.addressLine,'State:'), 2),add(indexof(pipeline().parameters.addressLine,'City:'), 6)))
```

|               |  |
|---------------|--|
| Variable type | <input checked="" type="radio"/> Pipeline variable <input type="radio"/> Pipeline return value |
| Name *        | cityName   |
| Value         | @substring(pipeline().parameters.ad...)  |

6. Next validate the pipeline and click on debug.

7. Here our pipeline was executed successfully.

| Activity name | Activity status | Activity type |
|---------------|-----------------|---------------|
| Set variable1 | Succeeded       | Set variable  |

- In the “Output” tab, the output is displayed as “Kolkata”.

```

Output
Copy to clipboard

{
  "name": "cityName",
  "value": "Kolkata"
}

Set variable1 Succeeded Set variable
  
```

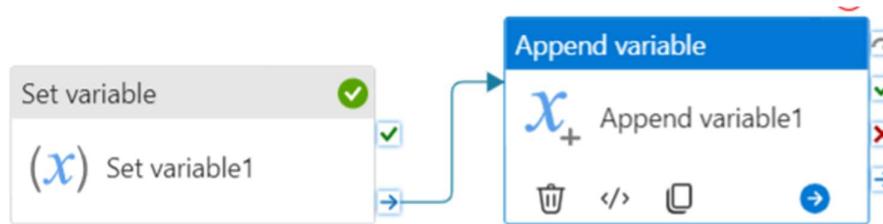
## Append Variable

- In this example we are going to use the same pipeline that we created before for the set variable active.
- So open the set variable pipeline that we created just before. Under variables add two more variables.
- For the listofCities give the below value.

Value: ["New York","London","Tokyo","Singapore City"]

| Name           | Type   | Default value                    |
|----------------|--------|----------------------------------|
| cityName       | String | Value                            |
| listOfCities   | Array  | ["New York", "London", "Tokyo",] |
| appendedCities | Array  | Value                            |

- Drag and drop the Append Variable Activity and make the connection as shown below.



5. Under Settings select the name and give the value.

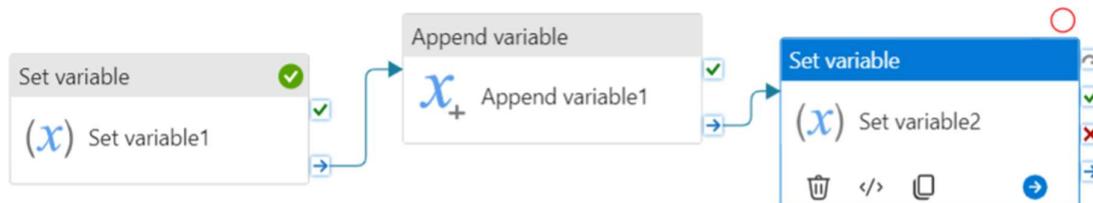
Value: @variables('cityName')

General    **Settings**    User properties

Name \*  + New

Value

6. Drag and drop the set variable activity and make connections as shown below.



7. Under Settings give the name and value.

Value: @variables('listOfCities')

General    **Settings**    User properties

Variable type ⓘ  Pipeline variable  Pipeline return value

Name \*  + New

Value

8. Next validate the Pipeline and click on Debug.

9. Here our Pipeline was executed successfully.

✓ Validate ▷ Debug ⚡ Add trigger

The screenshot shows the Pipeline interface with the following details:

- Activities:** Set variable1, Append variable1, Set variable2.
- Variables:** appendedCities (Pipeline variable).
- Output:** Pipeline run ID: 79f1bfa0-ac7a-40ea-93d8-bb72064e59c3, Pipeline status: Succeeded.
- Table:** Shows the execution details for three items.

| Activity name    | Activity status | Activity type   | Run start             | Duration     |
|------------------|-----------------|-----------------|-----------------------|--------------|
| Set variable2    | Succeeded       | Set variable    | 7/28/2023, 3:56:36 PM | Less than 1s |
| Append variable1 | Succeeded       | Append variable | 7/28/2023, 3:56:35 PM | Less than 1s |
| Set variable1    | Succeeded       | Set variable    | 7/28/2023, 3:56:35 PM | Less than 1s |

10. In the Output tab, the output is displayed as the value of the cityName Variable, appended to the Variable listOfCities of Array Type.

Output

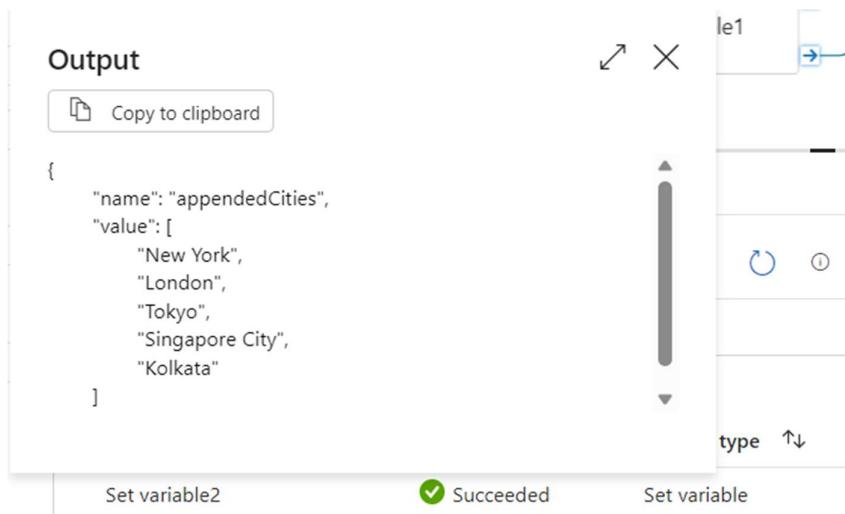
Copy to clipboard

```
{  
  "name": "appendedCities",  
  "value": [  
    "New York",  
    "London",  
    "Tokyo",  
    "Singapore City",  
    "Kolkata"  
  ]  
}
```

le1

Set variable2    Succeeded    Set variable

type ↑↓



# Until Activity

1. Go to Container and go to Output folder then delete the file.

The screenshot shows the Azure Storage Explorer interface. A container named 'f-demo' is selected. Inside it, there is a folder named 'Output' which contains three files: '\_SUCCESS', 'part-00000-e6862a16-47ef-4712-a2dc-d150acf6f...', and 'Std\_details.txt'. The 'Std\_details.txt' file has a context menu open, displaying various actions such as View/edit, Download, Properties, Generate SAS, View versions, View snapshots, Create snapshot, Change tier, Acquire lease, Break lease, and Delete.

2. Create a Pipeline.

The screenshot shows the Microsoft Azure Data Factory portal. On the left, there is a sidebar titled 'Factory Resources' with options like Pipelines, Change Data Capture (preview), Datasets, Data flows, and Power Query. Under Pipelines, a tooltip is displayed, showing 'Pipeline' (selected), 'Change Data Capture (preview)', 'Dataset', 'Data flow', 'Power Query', and 'Copy Data tool'. At the top, there is a search bar and a 'Publish all' button.

3. Under variable create a variable as shown below.

The screenshot shows the 'Variables' tab in the Azure Data Factory Variables section. It displays a table with columns: Name, Type, and Default value. There is one entry: 'FileAvailable' of type String with a default value of 'false'.

| Name          | Type   | Default value |
|---------------|--------|---------------|
| FileAvailable | String | false         |

4. Drag and drop the Until activity.

The screenshot shows the Azure Data Factory Pipeline designer. A pipeline named 'Until\_Pipe' is selected. In the 'Activities' section, there is a search bar with 'unti' typed in. Below it, an 'Until' activity is selected, showing its configuration pane. The pane includes tabs for Validate, Debug, and Add trigger. The 'Until' activity itself has a sub-section named 'Until1' which contains the message 'Activities No activities'.

5. Under settings, in the expression give the below expression.

Exp: @bool(variables('FileAvailable'))

General    **Settings**    Activities (0)    User properties

---

Expression ⓘ   

Timeout ⓘ   

6. Under Activity click on the pencil symbol.

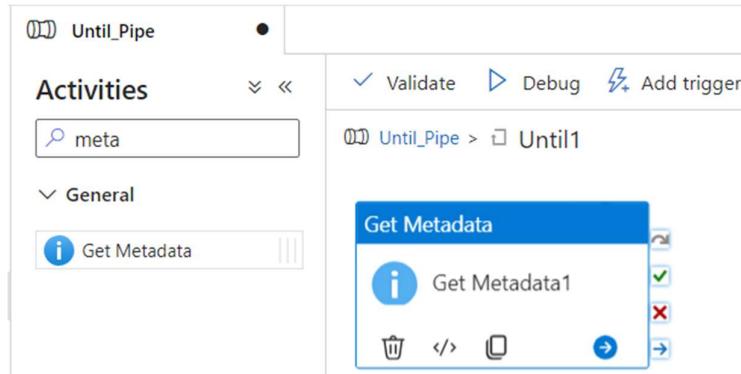
General    Settings    **Activities (0)**    User properties

---

| Case  | Activity      |
|-------|---------------|
| Until | No activities |



7. Drag and drop the get metadata activity.



The screenshot shows the 'Until' activity configuration. In the 'Activities' section, 'Get Metadata' is selected. The 'Get Metadata' activity is highlighted in a blue box. The 'Activity' pane shows 'Get Metadata1' with a green checkmark next to it. The top right of the screen has buttons for Validate, Debug, and Add trigger.

8. Under setting select the source dataset. And click on new to add an Argument.

General    **Settings**    User properties

---

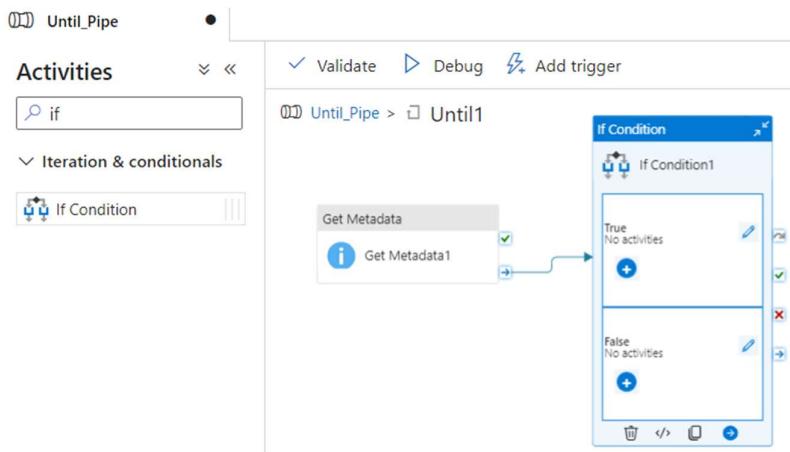
Dataset \*      Open  New 

Field list \*     New | 

Argument

Exists

9. Drag and drop the If condition and make the connection as shown below.



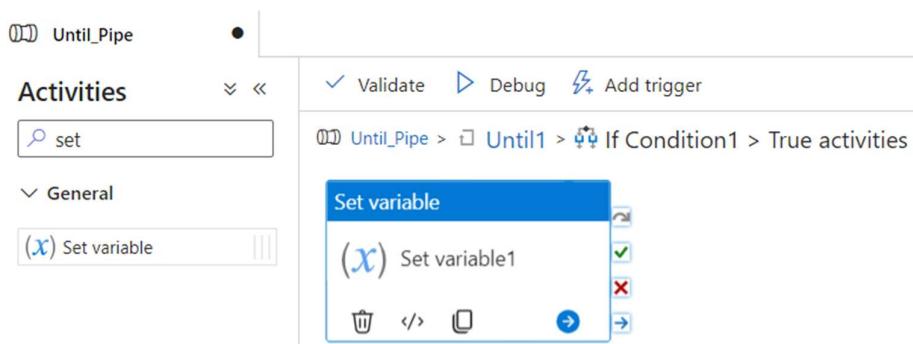
10. Under the activity, in the expression give the below expression.

Exp: @bool(activity('Get Metadata1').output.exists)

This screenshot shows the 'Activities' tab for the 'If Condition' activity. The 'Expression' field contains the expression '@bool(activity('Get Metadata1').output.exists)'. Below the expression, there are two sections: 'Case' and 'Activity'. The 'Case' section is empty. The 'Activity' section contains a single 'Activity' row, which is the 'Get Metadata' activity.

11. Click on the True condition pencil symbol.

12. Drag and drop the set variable activity as shown below.



13. Under setting give the Name and value as shown below.

This screenshot shows the 'Settings' tab for the 'Set variable' activity. Under 'Variable type', 'Pipeline variable' is selected. The 'Name' field is set to 'FileAvailable' and the 'Value' field is set to 'true'.

14. Now go back to Until and click on the False activity pencil symbol.

The screenshot shows the 'Until' activity configuration. At the top, there is an 'Expression' input field containing the expression '@bool(activity('Get Metadata1').output...'. Below it, there are two rows for 'Case' and 'Activity':

| Case  | Activity                        |
|-------|---------------------------------|
| True  | (x) Set variable1<br>1 Activity |
| False | No activities                   |

Each row has a blue pencil icon to its right.

15. Drag and drop the wait activity.

The screenshot shows the 'Until\_Pipe' pipeline. In the 'Activities' pane, a search bar contains 'wait'. A 'Wait' activity is selected and shown in the main pane. The pipeline structure is: Until\_Pipe > Until1 > If Condition1 > Wait1.

16. Under the settings give the time as 60 sec.

The screenshot shows the 'Settings' tab for the 'Wait' activity. The 'Wait time in seconds \*' field is set to '60'.

17. Go back to the pipeline and drag and drop the Copy data activity and make the connection.

The screenshot shows the 'Until\_Pipe' pipeline. In the 'Activities' pane, a search bar contains 'cop'. A 'Copy data' activity is selected and shown in the main pane. It is connected to the 'Wait1' activity.

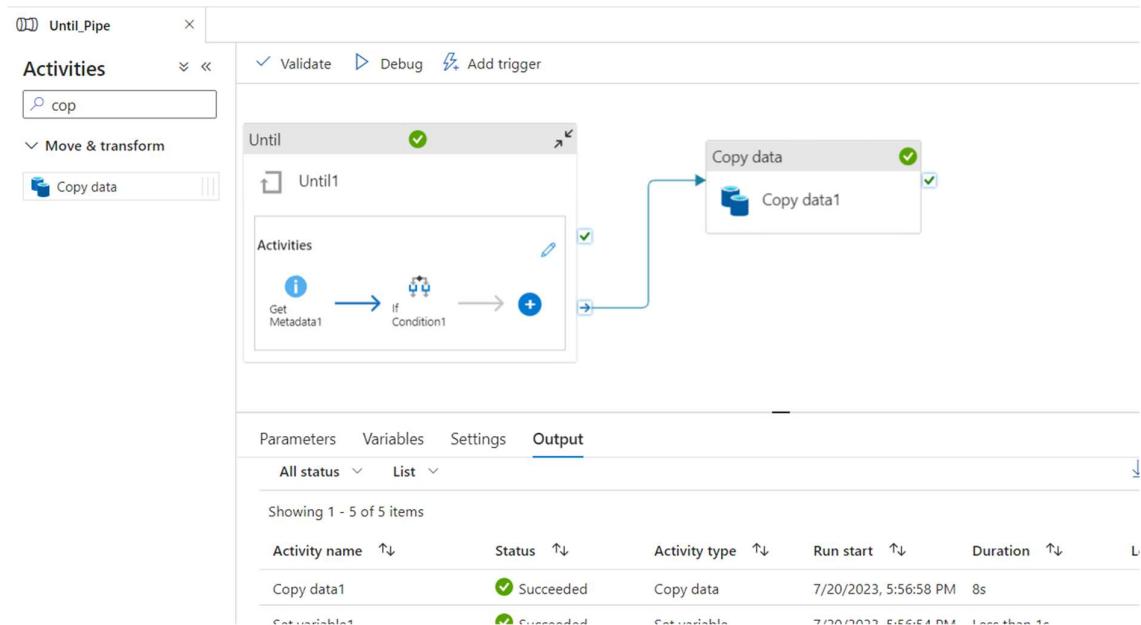
18. Under the Source select the source.

The screenshot shows the 'Source' tab for the 'Copy data' activity. The 'Source dataset \*' dropdown is set to 'ds\_src\_txt'. The 'File path type' dropdown is set to 'File path in dataset'. There are fields for 'Start time (UTC)' and 'End time (UTC)'. A 'Filter by last modified' checkbox is checked.

19. Under sink select the destination dataset.

The screenshot shows the 'Sink' tab of a dataset configuration. The 'Sink dataset' dropdown is set to 'ds\_sink\_txt'. The 'Copy behavior' dropdown is set to 'None'. The 'Max concurrent connections' input field is empty.

20. Now validate the pipeline and click on Debug. Here our pipeline runs successfully as shown below.



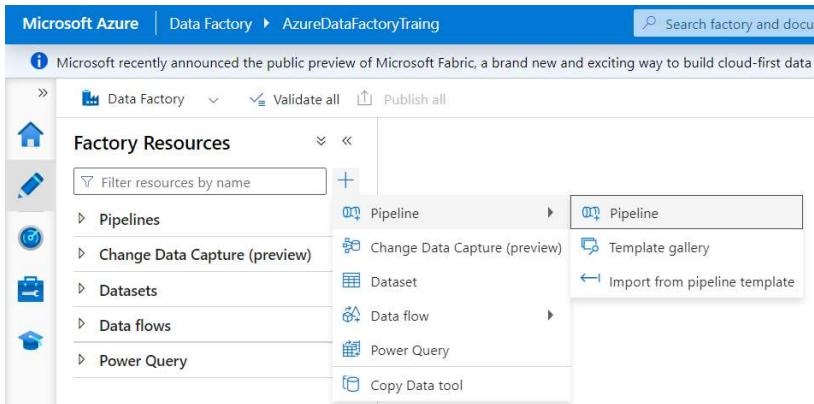
21. So in our case we already have a file in our source so it will directly copy into the target folder.

The screenshot shows the Azure Storage Explorer interface. It displays a list of files in a folder. The files listed are '\_SUCCESS', 'part-00000-e6862a16-47ef-4712-a2dc-d150acfb6f...', and 'Std\_details.txt'. The '\_SUCCESS' file was modified on 7/20/2023, 4:09:54 PM and is in the 'Hot (Inferred)' access tier. The other two files were modified on 7/20/2023, 5:57:05 PM and are also in the 'Hot (Inferred)' access tier.

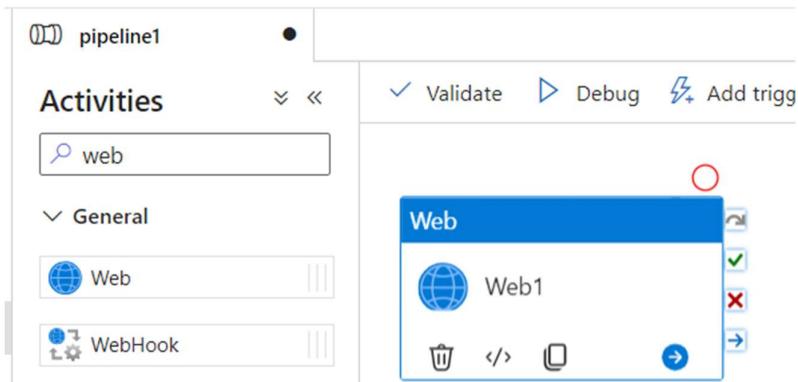
| Name   | Modified              | Access tier    | Archive status |
|--|-----------------------|----------------|----------------|
| [...]  |                       |                |                |
| _SUCCESS   | 7/20/2023, 4:09:54 PM | Hot (Inferred) |                |
| part-00000-e6862a16-47ef-4712-a2dc-d150acfb6f... | 7/20/2023, 4:09:54 PM | Hot (Inferred) |                |
| Std_details.txt                                  | 7/20/2023, 5:57:05 PM | Hot (Inferred) |                |

# Web Activity

1. Go to Azure Data Factory and go to Author then create a pipeline.



2. Drag and drop the web activity as shown below.



3. Under setting give the URL and set the method.

URL: <http://dummy.restapiexample.com/api/v1/employees>

General    Settings    User properties

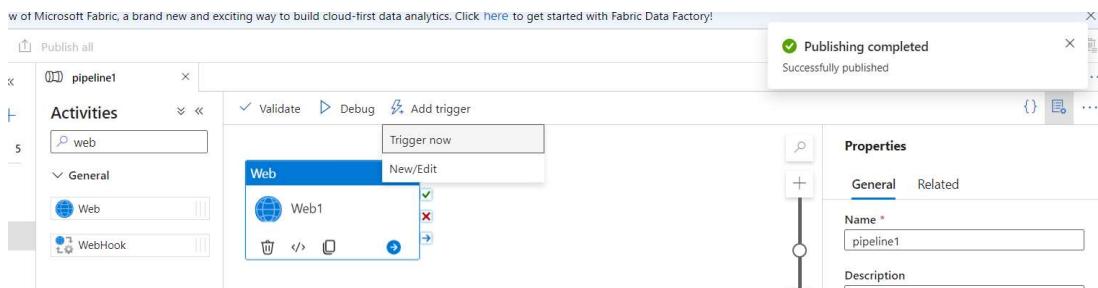
---

URL \* ⓘ  ⚠ Information will be sent to the URL specified. Please ensure you trust the URL entered.

Method \* ⓘ

Authentication ⓘ

4. Publish the pipeline and click on Add trigger then click on Trigger now.



5. Next click Ok.

6. Now go to the Monitor tab and you can see our pipeline status is succeeded.

A screenshot of the Microsoft Fabric Monitor tab. The left sidebar shows "Runs" and "Pipeline runs" selected. The main area is titled "Pipeline runs" and shows a table of runs. One run is listed: "pipeline1" with a run start of 7/21/2023, 9:50:17 AM, a run end of 7/21/2023, 9:50:23 AM, a duration of 6s, triggered by "Manual trigger", and a status of "Succeeded".

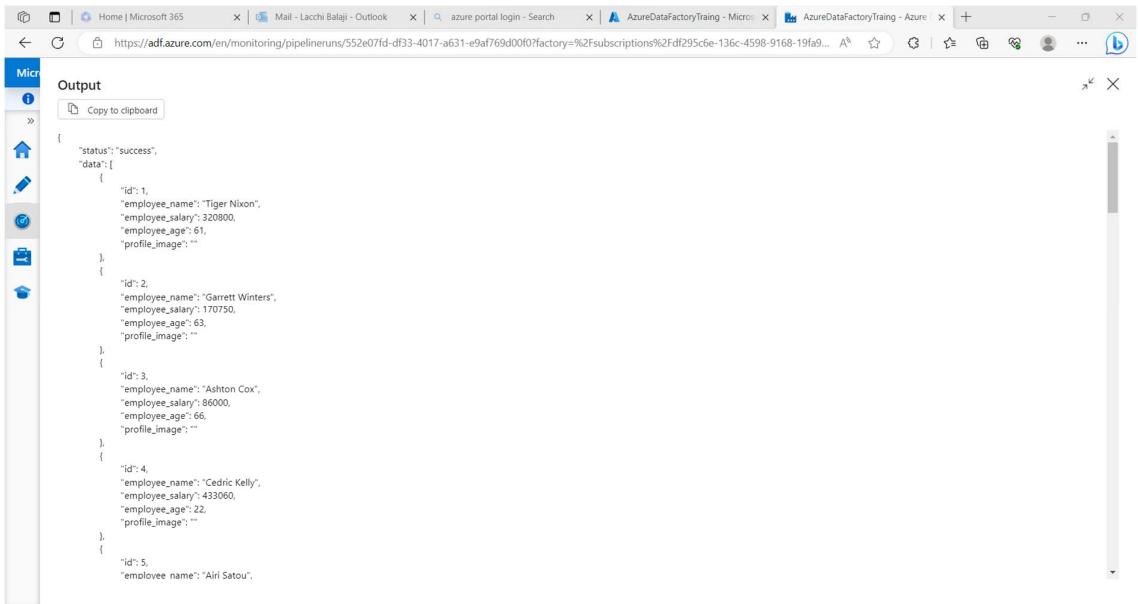
7. Next click on the Pipeline. Inside that click on Input you will see below input.

A screenshot of the Microsoft Fabric Pipeline run details. The left sidebar shows "Pipeline runs" selected. The main area shows a "Web" activity named "Web1" with a green checkmark. Below it, under "Input", is a JSON configuration block:

```
{  
  "url": "http://dummy.restapiexample.com/api/v1/employees",  
  "method": "GET",  
  "headers": {}  
}
```

The bottom of the screen shows a table of activity runs for "Web1", with one row showing a "Run start" of 7/21/2023, 9:50:19 AM, a "Duration" of 3s, and a status of "Succeeded".

8. Now if you click on Output you will see an output as shown below.

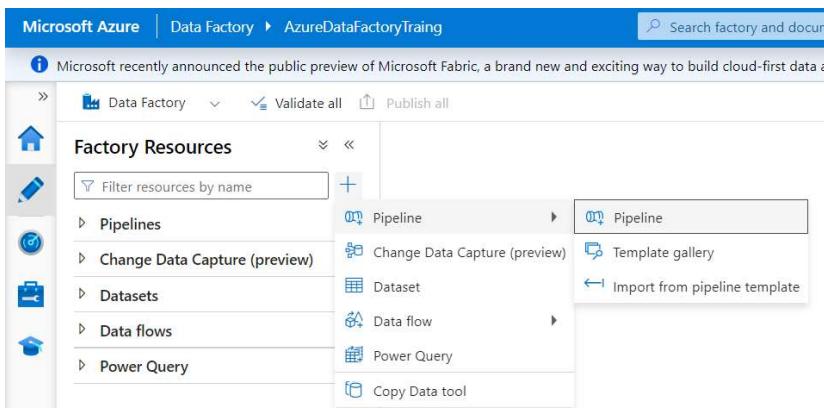


The screenshot shows a browser window with multiple tabs open. The active tab displays a JSON response under the heading 'Output'. The JSON data represents a successful pipeline run with five data items, each containing employee information: name, salary, age, and profile image URL. The JSON structure is as follows:

```
{ "status": "success", "data": [ { "id": 1, "employee_name": "Tiger Nixon", "employee_salary": 320800, "employee_age": 61, "profile_image": ""}, { "id": 2, "employee_name": "Garrett Winters", "employee_salary": 170750, "employee_age": 63, "profile_image": ""}, { "id": 3, "employee_name": "Ashton Cox", "employee_salary": 86000, "employee_age": 66, "profile_image": ""}, { "id": 4, "employee_name": "Cedric Kelly", "employee_salary": 430600, "employee_age": 22, "profile_image": ""}, { "id": 5, "employee_name": "Airi Satou", "employee_salary": 112000, "employee_age": 33, "profile_image": ""} ] }
```

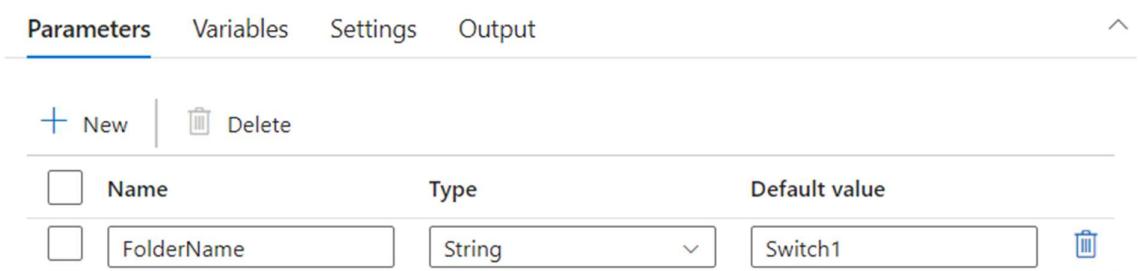
## Switch Activity

1. Create a pipeline.



The screenshot shows the 'Factory Resources' section of the Azure Data Factory interface. Under the 'Pipelines' category, a new pipeline is being created. A modal dialog is open, showing options for creating a new pipeline or importing from a template. The 'Pipeline' option is selected.

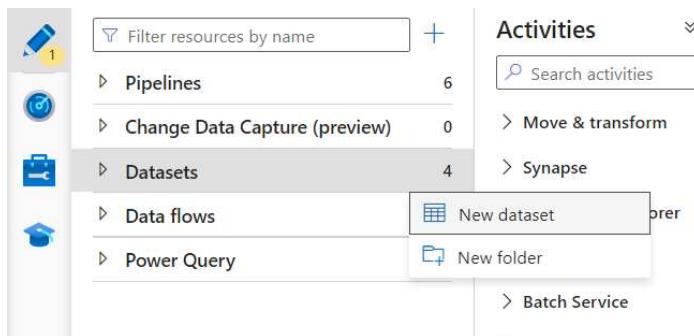
2. Under the parameter create the below parameter.



The screenshot shows the 'Parameters' tab of the pipeline configuration. It lists a single parameter named 'FolderName' of type 'String' with a default value of 'Switch1'. There are buttons for 'New' and 'Delete' at the top of the list.

| Name       | Type   | Default value |
|------------|--------|---------------|
| FolderName | String | Switch1       |

3. Create a Dataset.
4. Select the Azure Blob Storage and click on Continue. Then select the Binary and click on continue.



5. In the properties give the name, linked service, and the container name as shown below.

#### Set properties

Name  
SwitchDataset

Linked service \*  
AzureBlobStorage\_txt

File path  
f-demo / Directory / File name

First row as header

Import schema  
 From connection/store  From sample file  None

6. Now under parameter create a below parameter.

| Connection                      | Parameters             |
|---------------------------------|------------------------|
| <a href="#">New</a>             | <a href="#">Delete</a> |
| <input type="checkbox"/> Name   | Type                   |
| <input type="checkbox"/> folder | String                 |

7. Under connection, in the directory give the below expression as shown below.

Exp: @dataset().folder

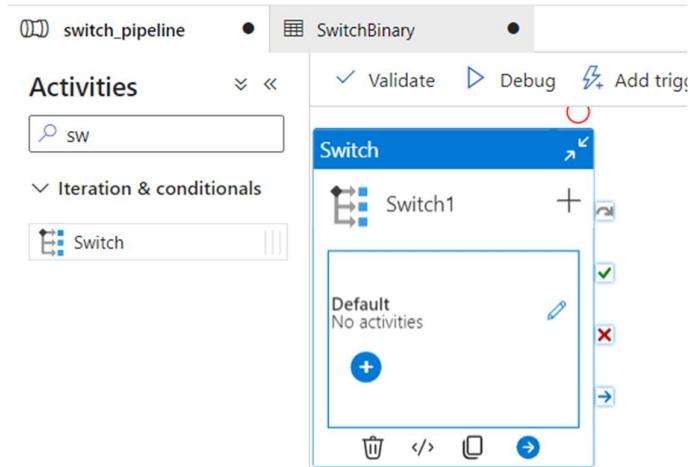
Connection Parameters

Linked service \*  
AzureBlobStorage\_txt

File path \*  
f-demo / @dataset().folder / File name

Compression type  
Select...

8. Go to Pipeline drag and drop the switch activity.



9. Under Activity give the below expression.

Exp: @pipeline().parameters.FolderName

A screenshot of the 'Activities' tab for the 'Switch1' activity. The 'Expression' field is set to '@pipeline().parameters.FolderName'. The 'Case' column shows 'Default' and the 'Activity' column shows 'No activities'.

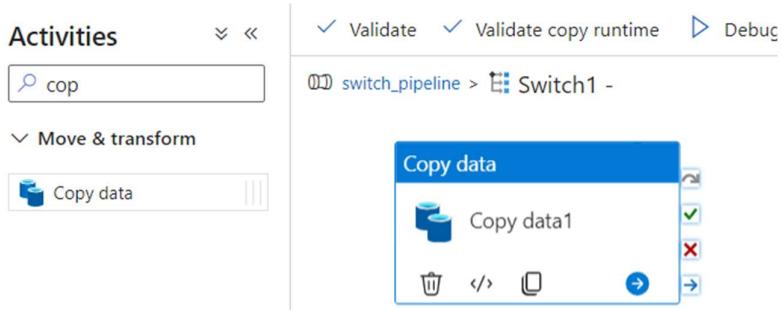
10. Now click on the plus symbol to add a case. Then click on the pencil symbol.

A screenshot of the 'Activities' tab for the 'Switch1' activity. It shows two cases: 'Default' and 'Case1'. Both cases have 'No activities' assigned. There are edit icons next to each case entry.

11. Give the case name Switch1 as shown below.

A screenshot of the 'General' tab for the 'Case1' row. The 'Case' field is filled with 'Switch1'.

12. Drag and drop the copy data activity as shown below.



13. Under the Source select the source dataset that we created in the day 1 session.

A screenshot of the 'Source' tab for a 'Copy data' activity. The tab is active and shows the 'Source dataset' dropdown set to 'ds\_src\_txt'. Below the dropdown are buttons for 'Open', 'New', 'Preview data', and 'Learn more'. Underneath the dropdown, there are four radio button options for 'File path type': 'File path in dataset' (selected), 'Prefix', 'Wildcard file path', and 'List of f...'. There is also a scroll bar on the right side of the screen.

14. Under the sink select the dataset that we created before and under the folder name give the below expression.

Exp: @pipeline().parameters.FolderName

A screenshot of the 'Sink' tab for a 'Copy data' activity. The tab is active and shows the 'Sink dataset' dropdown set to 'SwitchDataset'. Below the dropdown are buttons for 'Open', 'New', and 'Learn more'. Underneath the dropdown, there is a section titled 'Dataset properties' with a table. The table has columns for 'Name', 'Value', and 'Type'. One row in the table shows 'folder' with a value of '@pipeline().parameters.FolderName'. There is also a 'Copy behavior' dropdown set to 'None'.

15. Now go to pipeline and add another case.

A screenshot of the 'Activities (1)' tab for a pipeline. The tab is active and shows a table of cases. The first case is 'Default' with an activity 'No activities'. The second case is 'Switch1' with an activity 'Copy data1'. The third case is 'Case2' with an activity 'No activities'. Each row in the table has edit and delete icons.

16. Name it Switch2.

The screenshot shows the 'General' tab of a pipeline component configuration. The 'Case' field is highlighted and contains the value 'Switch2'.

17. Drag and drop the copy data activity as shown below.

The screenshot shows the 'Activities' pane of a pipeline editor. A 'Copy data' activity is selected and highlighted with a blue border. It is labeled 'Copy data2'.

18. Under the Source select the source dataset that we created in the day 1 session.

The screenshot shows the 'Source' tab of a pipeline component configuration. The 'Source dataset' dropdown is set to 'ds\_src\_txt'. Below it, under 'File path type', the radio button for 'File path in dataset' is selected.

19. Under the sink select the dataset that we created before and under the folder name give the below expression.

Exp: @pipeline().parameters.FolderName

The screenshot shows the 'Sink' tab of a pipeline component configuration. The 'Sink dataset' dropdown is set to 'SwitchDataset'. Under 'Dataset properties', there is a table:

| Name   | Value                             | Type |
|--------|-----------------------------------|------|
| folder | @pipeline().parameters.FolderName |      |

20. Next go to Pipeline and validate it and click on debug.

21. Here our pipeline executes successfully.

The screenshot shows the Azure Data Factory Pipeline status page. At the top, there are buttons for Validate, Debug, and Add trigger. Below that is a tree view of the pipeline structure under a 'Switch' node, with 'Switch1' selected. The pipeline run ID is e34003d0-7fed-4d0d-847e-1d42a749e19a, and the Pipeline status is Succeeded. The table below lists two activities: 'Copy data1' and 'Switch1', both of which succeeded. The table has columns for Activity name, Activity status, Activity type, Run start, and Duration.

| Activity name | Activity status | Activity type | Run start              | Duration |
|---------------|-----------------|---------------|------------------------|----------|
| Copy data1    | Succeeded       | Copy data     | 7/21/2023, 10:25:41 AM | 9s       |
| Switch1       | Succeeded       | Switch        | 7/21/2023, 10:25:41 AM | 10s      |

22. As initially give the parameter value as switch1 so if you go to your container you will see it as a Switch1 folder.

The screenshot shows the Azure Storage Blob Container 'f-demo'. The container has an 'Overview' tab selected. It displays the authentication method as 'Access key (Switch to Azure AD User Account)' and the location as 'f-demo'. The blob list shows a folder named 'Switch1' and other files like 'Output' and 'Output1'. The table has columns for Name, Modified, and Access tier.

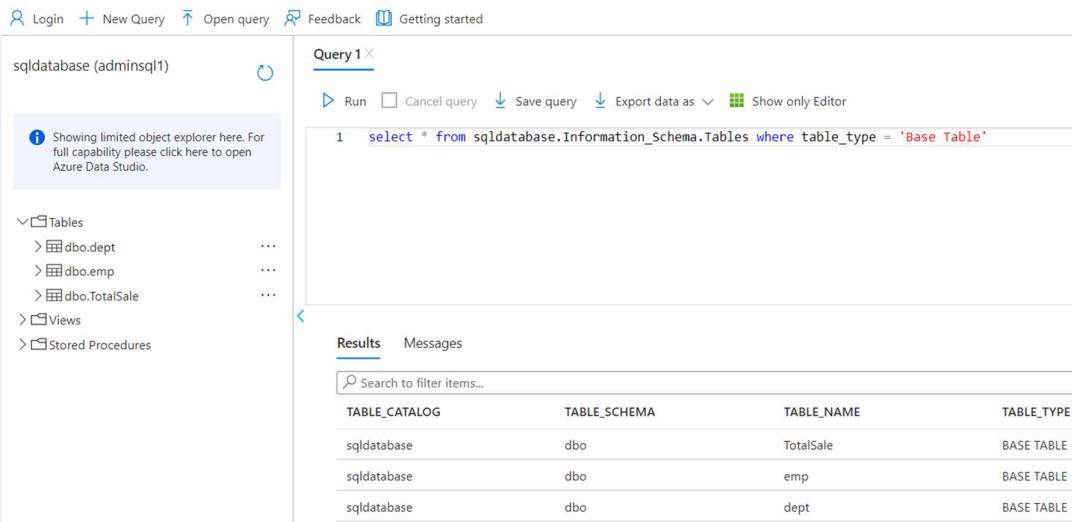
| Name    | Modified              | Access tier    |
|---------|-----------------------|----------------|
| Output  |                       |                |
| Output1 |                       |                |
| Switch1 |                       |                |
| Output  | 7/20/2023, 4:09:54 PM | Hot (Inferred) |

23. Inside the folder the file is copied from source to destination.

| Name            | Modifi |
|-----------------|--------|
| [..]            |        |
| Std_details.txt | 7/21/2 |

# Lookup Activity

1. Open the SQL database that we created in the previous session.
2. Here we have three tables. Here dynamically we will store these tables in the Azure Blob storage.



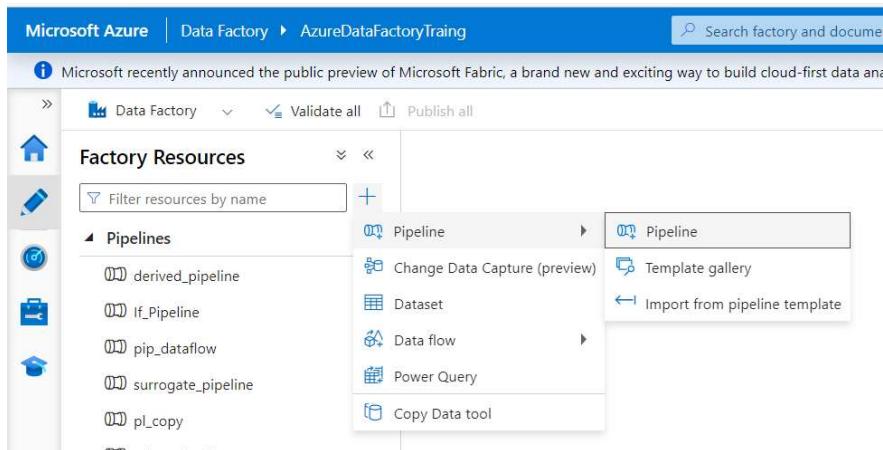
The screenshot shows the Microsoft Data Studio interface. On the left, there's a tree view of a database named 'sqldatabase' containing 'Tables', 'Views', and 'Stored Procedures'. In the center, a query editor window titled 'Query 1' displays the following SQL code:

```
1 select * from sqldatabase.Information_Schema.Tables where table_type = 'Base Table'
```

The results pane below shows a table with four columns: TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and TABLE\_TYPE. The data is as follows:

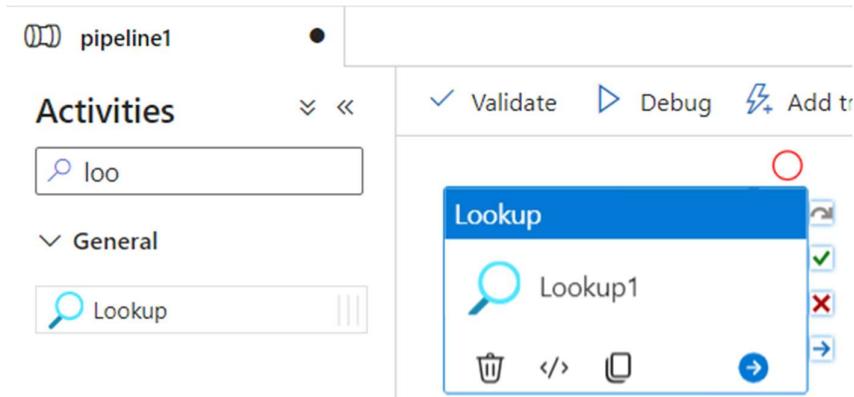
| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME | TABLE_TYPE |
|---------------|--------------|------------|------------|
| sqldatabase   | dbo          | TotalSale  | BASE TABLE |
| sqldatabase   | dbo          | emp        | BASE TABLE |
| sqldatabase   | dbo          | dept       | BASE TABLE |

3. Create a Pipeline.



The screenshot shows the Microsoft Azure Data Factory interface. The top navigation bar says 'Microsoft Azure | Data Factory > AzureDataFactoryTraining'. The main area is titled 'Factory Resources' and shows a list of 'Pipelines'. A context menu is open over one of the pipelines, with options like 'Pipeline', 'Change Data Capture (preview)', 'Dataset', 'Data flow', 'Power Query', and 'Copy Data tool'.

4. Drag and drop the lookup activity as shown below.



The screenshot shows the Microsoft Azure Data Factory pipeline editor. A pipeline named 'pipeline1' is open. In the 'Activities' section, a search bar contains the text 'loo'. Under the 'General' category, a 'Lookup' activity is selected. A tooltip for the 'Lookup' activity is displayed, showing its name 'Lookup1' and various icons for configuration and validation.

5. Under settings click on New.

General    **Settings** <sup>1</sup>    User properties

---

Source dataset \*

First row only

6. Select the Azure SQL Database and click continue.  
7. Just give the name and select the linked service and click ok.

Set properties

Name

Linked service \*

Table name

Import schema  
 From connection/store  None

> Advanced

8. Next check the query option and give the below query.

Query: select \* from sqldatabase.Information\_Schema.Tables where table\_type = 'Base Table'

9. Uncheck the First row-only option.

General    **Settings**    User properties

---

Source dataset \*

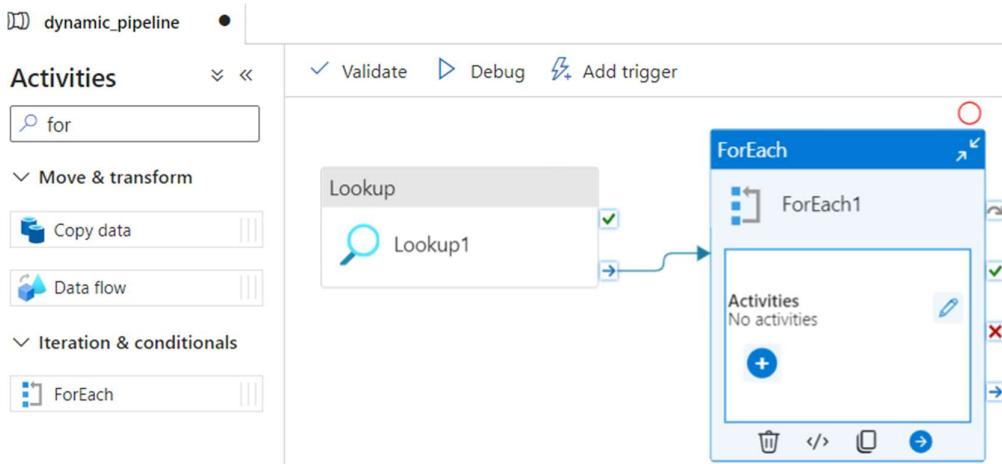
First row only

Use query  
 Table  Query  Stored procedure

Query \*

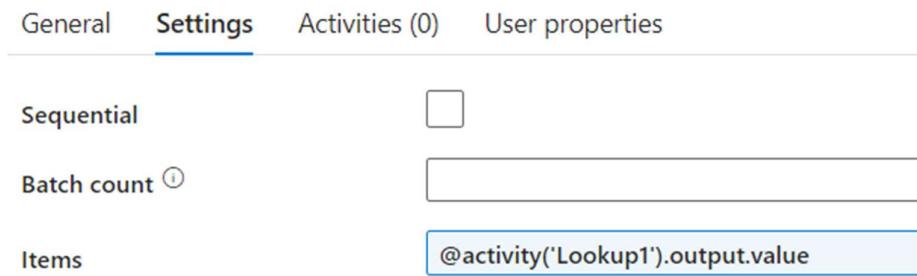
Query timeout (minutes)

10. Drag and drop the foreach activity and make the connection as shown below.

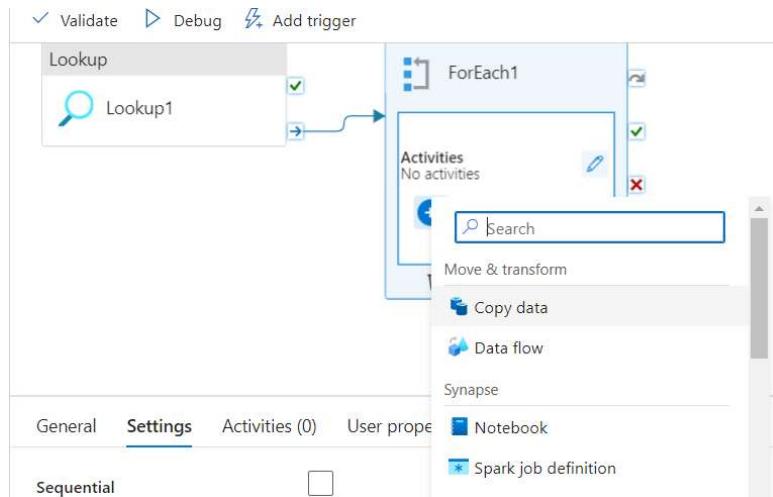


11. Under setting in the Items give the below expression as shown below.

Exp: @activity('Lookup1').output.value



12. Click on the plus symbol in the foreach loop and click on copy data option.



13. Under the source click on New.

General    **Source** <sup>1</sup>    Sink <sup>1</sup>    Mapping    Settings    User properties

Source dataset \*

14. Select the Azure SQL Database and click on continue.

15. Then give the name and linked service then click on Ok.

### Set properties

Name

Dynamic\_Source

Linked service \*

AzureSqlDatabase

Table name



Edit

Import schema

From connection/store     None

> Advanced

16. Click on Open.

General    **Source** <sup>1</sup>    Sink <sup>1</sup>    Mapping    Settings    User properties

Source dataset \*

Use auerv  Table  Query  Stored procedure

17. Under parameters add two parameters as shown below.

Connection    Schema    **Parameters**

| <input type="checkbox"/> Name       | Type   | Default value                      | <input type="button" value=""/> |
|-------------------------------------|--------|------------------------------------|---------------------------------|
| <input type="checkbox"/> TableName  | String | <input type="text" value="Value"/> | <input type="button" value=""/> |
| <input type="checkbox"/> SchemaName | String | <input type="text" value="Value"/> | <input type="button" value=""/> |

18. Now go to connections and check the edit option and give the expression as shown below.

Exp: @dataset().SchemaName

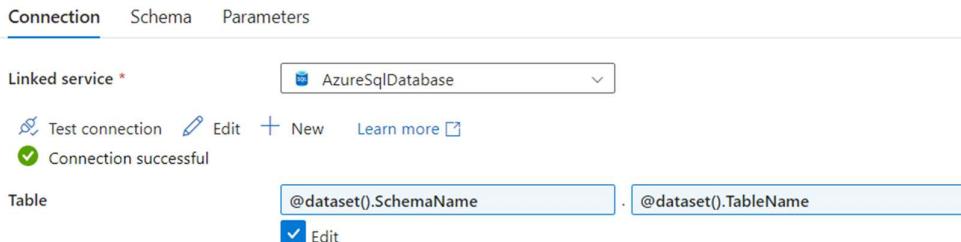
Exp: @dataset().TableName

Connection Schema Parameters

Linked service \*  AzureSqlDatabase

Connection successful

Table  .   
 Edit



19. Under source for the TableName and SchemaName give the below expression.

Exp: @item().table\_name

Exp: @item().table\_schema

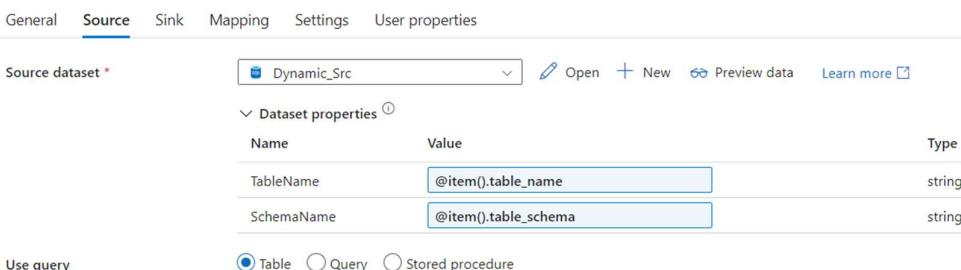
General Source Sink Mapping Settings User properties

Source dataset \*  Dynamic\_Src

Dataset properties

| Name       | Type   |
|------------|--------|
| TableName  | string |
| SchemaName | string |

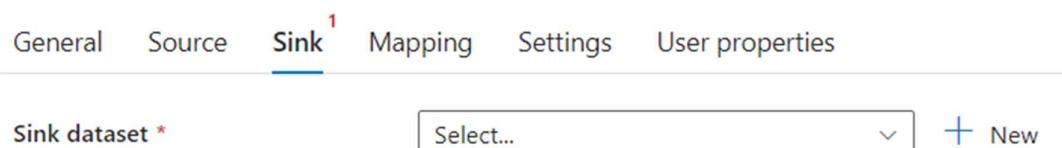
Use query  Table  Query  Stored procedure



20. Under sink click on new.

General Source Sink <sup>1</sup> Mapping Settings User properties

Sink dataset \*



21. Select Azure Blob Storage and click on continue. Then click on delimited text and click on continue.

22. Give the Name, Linked service, and Container name.

#### Set properties

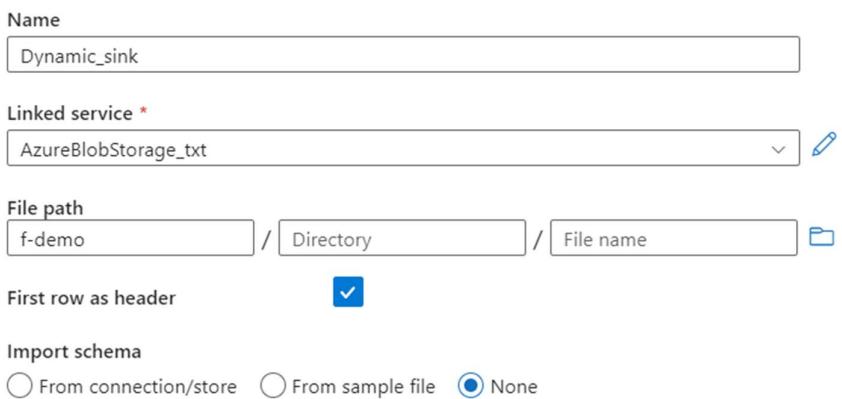
Name

Linked service \*  AzureBlobStorage\_txt

File path  /  /

First row as header

Import schema  
 From connection/store  From sample file  None



23. Click on Open.

The screenshot shows the 'Sink' tab of a dataset configuration. The 'Sink dataset' dropdown is set to 'Dynamic\_sink'. Below it, the 'Copy behavior' dropdown is set to 'None'. To the right, there are buttons for 'Open', 'New', and 'Learn more'.

24. Under parameter create below three parameters.

The screenshot shows the 'Parameters' tab with three new parameters defined:

| Name       | Type   | Default value |
|------------|--------|---------------|
| TopFolder  | String | Value         |
| FolderName | String | Value         |
| FileName   | String | Value         |

25. Under connection, in the directory give the below expression.

Exp: @concat(dataset().TopFolder,dataset().FolderName)

26. And in the File Name give the below expression.

Exp: @dataset().FileName

The screenshot shows the 'Connection' tab for an Azure Blob Storage linked service. The 'File path' field contains the expression '@concat(dataset().TopFolder,dataset().FolderName) / @dataset().FileName'. The 'Row delimiter' field is set to 'Default (\r,\n, or \r\n)'.

27. Now go to the pipeline under the sink for the top folder and give Dynamic\_Pipeline.

28. For the Folder Name give this expression: @item().table\_name

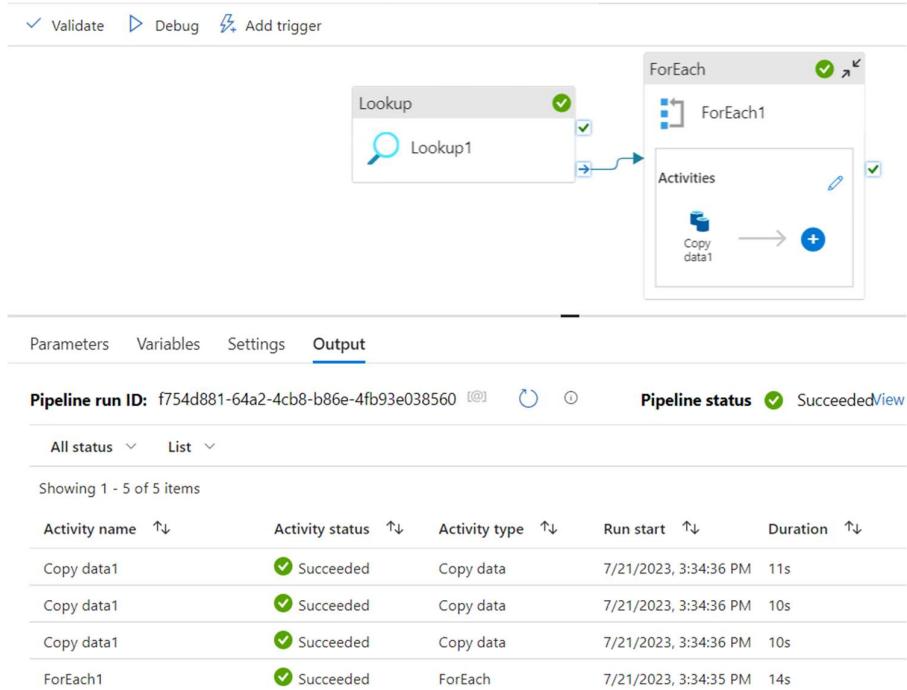
29. For the File Name give this expression: @concat(item().table\_schema,  
item().table\_name)

The screenshot shows the 'Sink' tab for the 'Dynamic\_sink' dataset. Under 'Dataset properties', the following values are set:

| Name       | Value                                    |
|------------|--|
| TopFolder  | Dynamic_Pipeline                         |
| FolderName | @item().table_name                       |
| FileName   | @concat(item().table_schema, item()....) |

30. Now validate the pipeline and click on debug.

31. Our pipeline was executed successfully.



32. Now go to container and you will see three folders as shown below. Each folder has a table.

Upload Change access level Refresh | Delete Change

**Authentication method:** Access key ([Switch to Azure AD User Account](#))

**Location:** f-demo

Search blobs by prefix (case-sensitive)

Add filter

|                          | Name                      | Modified |
|--------------------------|---------------------------|----------|
| <input type="checkbox"/> | Dynamic_Pipelinedept      |          |
| <input type="checkbox"/> | Dynamic_Pipelineemp       |          |
| <input type="checkbox"/> | Dynamic_PipelineTotalSale |          |

# Wait Activity

1. Create a Pipeline.

The screenshot shows the 'Factory Resources' pane in the Azure Data Factory interface. Under the 'Pipelines' category, 'Pipeline' is selected. Other options like 'Change Data Capture (preview)', 'Dataset', 'Data flow', 'Power Query', and 'Copy Data tool' are also listed.

2. Create Parameters as shown below.

| Name              | Type   | Default value |
|-------------------|--------|---------------|
| waitTimeInSecond  | Int    | 75            |
| waitFinishMessage | String | Wait is over  |

3. Create a Variable as shown below.

| Name       | Type   | Default value |
|------------|--------|---------------|
| waitOutput | String | Value         |

4. Drag and drop the wait as shown below.

The screenshot shows the pipeline editor for 'Wait\_pipeline'. A 'Wait' activity is selected and highlighted with a red circle. The activity has a name 'Wait1' and a small icon of a hourglass. Below the activity are standard control icons: a trash can, a close button, a copy/paste button, and a next/previous step button.

5. Under settings give the below expression.

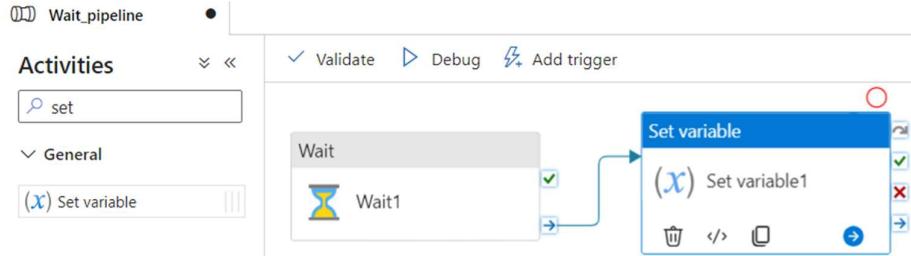
Exp: @pipeline().parameters.waitTimeInSecond

General    **Settings**    User properties

Wait time in seconds \*

@pipeline().parameters.waitTimeInSe...

6. Drag and drop the Set Variable and make connections as shown below.



7. Under settings, select the name and give the below expression.

Exp: @pipeline().parameters.waitFinishMessage

General    **Settings**    User properties

Variable type ⓘ

Pipeline variable  Pipeline return value

Name \*

waitOutput

+ New

Value

@pipeline().parameters.waitFinishMe...

8. Now validate the pipeline and click on Debug.

9. Our Pipeline was executed successfully.

10. It can be seen that the "Wait" Activity executed for 75 seconds, i.e., 1 minute and 15 seconds.

11. The "Wait" Activity transferred control to the "Set variable" Activity on the 76th second, i.e., when the time spent by the "Wait" Activity was 1 minute 16 seconds.

✓ Validate    ▶ Debug    ⚡ Add trigger



Parameters    Variables    Settings    **Output**

Pipeline run ID: dab85b0c-2bcf-4524-a317-ecff77ff1e35 ⏪ ⏴ ⓘ    Pipeline status: ✓ Succeeded View debug ↴ ↴

All status ▾

Showing 1 - 2 of 2 items

| Activity name ↑↓ | Activity status ↑↓ | Activity type ↑↓ | Run start ↑↓         | Duration ↑↓  | Log |
|------------------|--------------------|------------------|----------------------|--------------|-----|
| Set variable1    | ✓ Succeeded        | Set variable     | 8/4/2023, 4:56:07 PM | Less than 1s |     |
| Wait1            | ✓ Succeeded        | Wait             | 8/4/2023, 4:54:51 PM | 1m 16s       |     |

# Script Activity

1. Create below tables as shown below.

Code: create table emp\_det(  
id int,  
name varchar(20),  
gender varchar(20))

```
insert into emp_det values(1, 'Mahesh', 'Male');  
insert into emp_det values(2, 'Anu', 'Female');
```

Create table dept\_det(  
id int,  
name varchar(20))

```
insert into dept_det values(1, 'IT');  
insert into dept_det values(2, 'HR');  
insert into dept_det values(1, 'Payroll');
```

```
Select * from emp_det;  
Select * from dept_det;
```

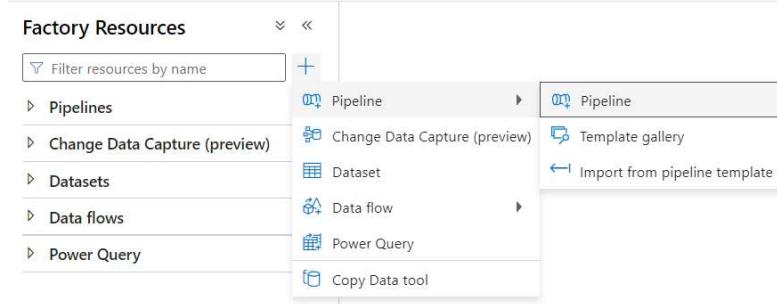
```
17  select * from emp_det;
```

Results    Messages

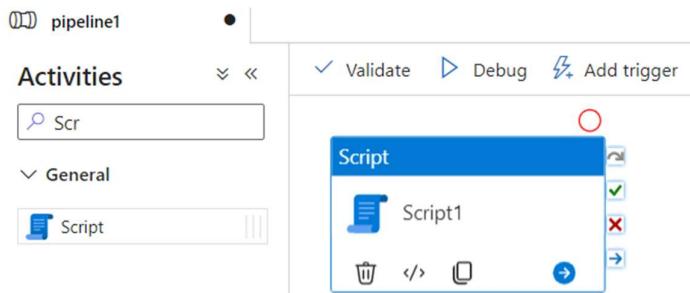
Search to filter items...

| id | name   | gender |
|----|--------|--------|
| 1  | Mahesh | Male   |
| 2  | Anu    | Female |

2. Create a Pipeline.



3. Drag and drop the Script Activity.



4. Under setting, give the below script.

Code: Select \* from emp\_det;  
Select \* from dept\_det;  
Print 'Queries Executed'

A screenshot of the 'Settings' tab for the Script activity. It shows a 'Linked service' dropdown set to 'AzureSqlDatabase'. Below it, there's a 'Script' section with a 'Query' radio button selected. The script text area contains the following code:

```
Select * from emp_det;
Select * from dept_det;
Print 'Queries Executed'
```

5. Scroll down under Advanced, check the Enable logging and select the linked service then set the location.

A screenshot of the 'Advanced' settings tab. It includes fields for 'Script block execution timeout (minutes)' (set to 120), 'Enable logging' (checkbox checked), and 'Script log output' (radio button set to 'External storage'). Below that is a 'Logging settings' section with a 'Logging account linked service' dropdown set to 'AzureBlobStorage\_txt'. It also shows a 'Folder path' input field with 'f-demo/Output' and a 'Browse' button.

6. Validate the pipeline and click on Debug.
7. Here our pipeline was executed successfully.

The screenshot shows the Azure Pipeline interface. At the top, there are three buttons: 'Validate' (with a checkmark), 'Debug' (with a play icon), and 'Add trigger'. Below this is a list of activities under the heading 'Script'. A single activity, 'Script1', is listed with a green checkmark indicating success. The interface includes tabs for 'Parameters', 'Variables', 'Settings', and 'Output', with 'Output' being the active tab. Below the activities, a summary table provides details about the execution:

| Activity name | Activity status                                | Activity type |
|---------------|--|---------------|
| Script1       | <span style="color: green;">✓ Succeeded</span> | Script        |

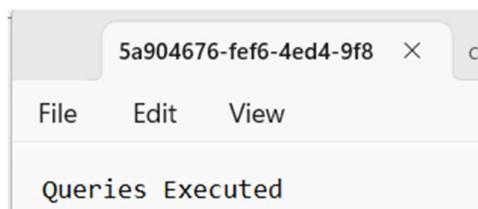
8. Go to the location and download the file as shown below.

The screenshot shows the Azure Storage Blob container interface. It displays a list of blobs with one item selected: '5a904676-fef6-4ed4-9f83-08973dbda42e\_0.txt'. The blob details pane on the right shows the following information:

- Authentication method: Access key (Switch to Azure AD User Account)
- Location: f-demo / Output / scriptactivity-logs / Script1 / 5a904676-fef6-4ed4-9f83-08973dbda42e
- File size: 17 B
- File type: Append blob
- Status: Available

A context menu is open over the blob, listing options such as View/edit, Download, Properties, Generate SAS, and Delete.

9. When you open the document, you will see the below message.



10. Now go back to Script Activity settings and change the query then add the below script parameter.

Query: select @name as NameCol

The screenshot shows the 'Settings' tab of a Script Activity configuration. The 'Connection' dropdown is set to 'AzureSqlDatabase'. The 'Type' radio button is selected for 'Query'. The query editor contains the SQL statement: 'select @name as NameCol'. Below the editor, under 'Script parameters', there is a table:

| Index | Name | Type   | Value  | Direction |
|-------|------|--------|--------|-----------|
| 1     | name | String | Mahesh | Input     |

There is also a note: 'Add dynamic content [Alt+Shift+D]'. The 'Test connection' button is visible at the top right.

11. Validate the pipeline and click on Debug.

12. Here our pipeline was executed successfully.

The screenshot shows the 'Output' tab for a pipeline run. The 'Pipeline run ID' is f5fb156f-0213-4923-ae53-fa40c0c280cf. The run status is 'Succeeded'. The table below lists the activity details:

| Activity name | Activity status | Activity type | Run start     |
|---------------|-----------------|---------------|---------------|
| Script1       | Succeeded       | Script        | 8/22/2023, 1: |

13. In the Output you will see the Name as shown below.

The screenshot shows the 'Output' pane for the pipeline run. It displays the JSON result set from the 'Script1' activity. The result is a single row with the key 'NameCol' and value 'Mahesh'.

```
{"resultSetCount": 1, "recordsAffected": 0, "resultSets": [ { "rowCount": 1, "rows": [ { "NameCol": "Mahesh" } ] } ]}
```

# Stored Procedure Activity

1. In this example we are using the below table. Create a Procedure for the Table.

Code: create procedure deleteEmp @id int  
AS  
delete FROM emp\_det where id = @id  
GO;

The screenshot shows a database query editor interface. At the top, there are buttons for Run, Cancel query, Save query, Export data as, and Show only Editor. Below the buttons is a code editor window containing the following T-SQL script:

```
1 select * from [dbo].[emp_det]
2
3 create procedure deleteEmp @id int
4 AS
5 delete FROM emp_det where id = @id
6 GO;
```

Below the code editor is a results grid titled "Results". It has columns for id, name, and gender. The data is as follows:

| id | name   | gender |
|----|--------|--------|
| 1  | Mahesh | Male   |
| 2  | Anu    | Female |

2. Create a Pipeline.

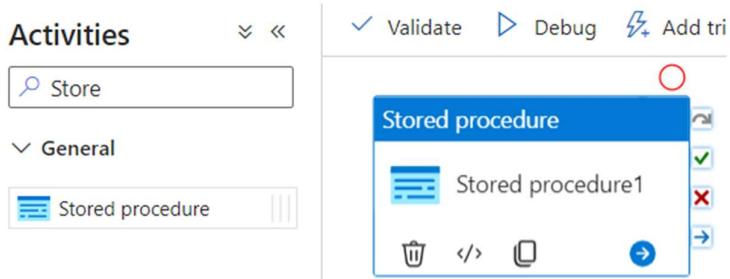
The screenshot shows the "Factory Resources" pane in the Power BI service. On the left is a sidebar with options: Pipelines, Change Data Capture (preview), Datasets, Data flows, and Power Query. On the right, a list of resources is shown, with "Pipeline" selected. A context menu is open over the "Pipeline" item, showing options: Pipeline, Change Data Capture (preview), Dataset, Data flow, Power Query, and Copy Data tool.

3. Under parameter create a below parameter.

The screenshot shows the "Parameters" section in the Power BI service. At the top, there are tabs for Parameters, Variables, Settings, and Output. Below the tabs, there is a "New" button and a "Delete" button. A table lists parameters with columns for Name, Type, and Default value. One row is visible, showing a parameter named "id" of type "String" with an empty "Value" field.

| Name | Type   | Default value |
|------|--------|---------------|
| id   | String |               |

4. Drag and drop the stored procedure activity.



5. Now select the SQLServer linked service and select the Stored Procedure that we created before.
6. Then under parameters click on import and give the below expression.

Exp: @pipeline().parameters.id

This screenshot shows the 'Settings' tab for a stored procedure activity. At the top, there are tabs for 'General', 'Settings' (which is selected), and 'User properties'. Below the tabs, the 'Linked service' dropdown is set to 'AzureSqlDatabase'. Under 'Stored procedure name', the value is '[dbo].[deleteEmp]'. There is also a 'Refresh' button. The 'Stored procedure parameters' section is expanded, showing a table with one row. The table has columns for 'Name', 'Type', and 'Value'. The 'Name' column contains 'id', the 'Type' column contains 'Int32', and the 'Value' column contains '@pipeline().parameters.id'.

| Name | Type  | Value                     |
|------|-------|---------------------------|
| id   | Int32 | @pipeline().parameters.id |

7. Validate the pipeline and click on debug.
8. It will ask the id value, give 1.

### Pipeline run

#### Parameters

| Name | Type   | Value |
|------|--------|-------|
| id   | string | 1     |

9. Our pipeline executed successfully.

The screenshot shows the 'Output' tab of a pipeline run. At the top, there's a list of activities: 'Stored procedure' (green checkmark) and 'Stored procedure1' (blue checkmark). Below this, tabs for 'Parameters', 'Variables', 'Settings', and 'Output' are visible, with 'Output' being the active tab. The pipeline run ID is 46485243-4836-4c9b-b122-98ac457cfb8a. The activity list shows 'Stored procedure1' with a green checkmark and the status 'Succeeded'. The activity type is 'Stored procedure'.

10. Now go to SQL database and run the table, id 1 row has been deleted.

The screenshot shows an Azure SQL Database query results page. At the top, there are buttons for 'Run', 'Cancel query', 'Save query', 'Export data as', and 'Show only Editor'. The query entered is 'select \* from [dbo].[emp\_det]'. Below the results table, there are 'Results' and 'Messages' tabs, with 'Results' being the active tab. A search bar at the top of the results table says 'Search to filter items...'. The results table has columns 'id', 'name', and 'gender'. One row is shown: id 2, name Anu, gender Female.

| id | name | gender |
|----|------|--------|
| 2  | Anu  | Female |