

*****25/09/2023*****

Agenda:

- Azure
 - Cloud concepts
 - Types
 - Lass, paas, Saas
 - Diff between on premises & cloud based'
 - Job responsibilities as azure data engineer
 - Types of data
-

Azure data engineer also help ensure that data pipeline and data stores are high performing, efficient, organized, and reliable, given a set of business.

Server

If we are going to use cloud, In cloud no need to maintain any server, with internet we can access locally.

No need to take care of infrastructure, cloud is cheaper.

To access azure login: <https://portal.azure.com/#home>

[GitHub - deeksharm/DP203](#)

<https://github.com/deeksharm/DP203>

[DP203/Azure_Sep25.xlsx at main · deeksharm/DP203 · GitHub](#)

Big data, hadoop: [Big Data Tutorial – Learn Big Data from Scratch - DataFlair \(data-flair.training\)](#)

AZURE DOCUMENTATION: [Azure documentation | Microsoft Learn](#)

PAYMENT MODEL

1. Pay-as-you-go
2. 1 year reservation -->
up to 30% discount
3. 3 years reservation -->
up to 50-70%

Learning link:

[Data redundancy - Azure Storage | Microsoft Learn](#)

[Introduction to Azure Storage - Cloud storage on Azure | Microsoft Learn](#)

[Azure cross-region replication | Microsoft Learn](#)

[Dynamic data masking - SQL Server | Microsoft Learn](#)

[What is a Public Cloud - Definition | Microsoft Azure](#)

[Storage account overview - Azure Storage | Microsoft Learn](#)

[Azure Storage Explorer – cloud storage management | Microsoft Azure](#)

[Adding Users to Your SQL Azure Database | Azure Blog | Microsoft Azure](#)

[Azure portal: Dynamic data masking - Azure SQL Database | Microsoft Learn](#)

[Dynamic data masking - Azure SQL Database | Microsoft Learn](#)

Pricing [Pricing Calculator | Microsoft Azure](#)

Cosmos db:

[Databases, containers, and items - Azure Cosmos DB | Microsoft Learn](#)

Azure resource manager:

[Create template - Visual Studio Code - Azure Resource Manager | Microsoft Learn](#)

[Hadoop Ecosystem and Their Components - A Complete Tutorial - DataFlair \(data-flair.training\)](#)

Refer:

Python, Big data → data flair.com([Free Online Certification Courses – Learn Today. Lead Tomorrow. - DataFlair \(data-flair.training\)](#))

The Azure Storage platform includes the following data services:

- [Azure Blobs](#): A massively scalable object store for text and binary data. Also includes support for big data analytics through Data Lake Storage Gen2.
- [Azure Files](#): Managed file shares for cloud or on-premises deployments.
Existing configuration file we can use no need to do configuration for all 50 virtual machines . in files we can store configuration files.
SMB protocol, NFS.
- [Azure Elastic SAN](#) (preview): A fully integrated solution that simplifies deploying, scaling, managing, and configuring a SAN in Azure.
- [Azure Queues](#): data stored in message format.
- [Azure Tables](#): A NoSQL store for schemaless storage of structured data.
- [Azure managed Disks](#): Block-level storage volumes for Azure V

To check pricing of service we are going to create:

[Pricing Calculator | Microsoft Azure](#)

STORAGE ACCOUNT:

CREATE RESOURCE GROUP:

- 1) Home → Subscriptions → Pay-as-you go → Resource Group → + create → Resource group: RG_UST_Training → Region : EAST(US)
- 2) Tags → Project : UST_Training, Name: Monisha (it can be any name)
- 3) Click Review : create
- 4) Create

DATA LAKE(HIERARCHICAL NAMESPACE):

- 1) Go inside the resource group created(RG_UST_Training) → +create → search: storage account → expand create → click storage account → Account Name(should be in small letters, globally unique, alpha numeric value because the url will use this name so it should be unique) : adlsusstraining(azure datalake storage)
- 2) 
- 3) Above tick the check box for Standard(if you premium expensive you can choose that also)
- 4) Redundancy : Choose LRS(Locally Redundant Storage) → less cost
- 5) Click Next: Advanced
- 6) Go with default option: Tick 1st : https secure and 3rd : authentication method key access
- 7) Click Next
- 8) Tick Enable Storage
- 9) Hierarchical Namespace **Tick it** (DATA LAKE)
- 10) Access Tier by default Hot go with that option
- 11) Click Next: Networking
- 12) Networking Connectivity: **Tick** 1st: Enable public access
- 13) Network Routing : **Tick** Microsoft Network Routing
- 14) Click Next

15) **Tick** Enable soft delete or blobs

16) Encryption go with default

17) Tags:

Project: UST_Training

Owner : Monisha

18)Review and create 19)Click Create

CREATE VIRTUAL MACHINE:((WINDOWS))

- 1) Go to the Resource Group(RG_UST_Training) ➔ +Create ➔ search : Virtual Machine ➔ Click create ➔ click virtual machine ➔ VM name: vmmoni, Region : EAST US
- 2) Availability Option : No infrastructure redundancy required (As of now because we r not creating for production)
- 3) Security type : Trusted launch virtual machine
- 4) Image ➔ configuration file Windows server 2022 datacenter: Azure Edition - * 64 Gen 2
- 5) Spot instance : B1'S
- 6) Administrator Account :
USER NAME : Monishab
PASSWORD : *****
- 7) Public Inbound ports: Allow selected ports
- 8) Select inbound ports: RDP: 3389
- 9) Os disk type : Premium SSD
- 10) Key ➔ platform managed key
- 11) Default
- 12) Next
- 13) Tags

Project: UST_Training

Owner : Monisha

14) Review and create

15) Click Create

16) How to connect vm(windows):

17) Start

18) Connect

19) Click select

20) Download RDP file ➔ click open file

21) Connect

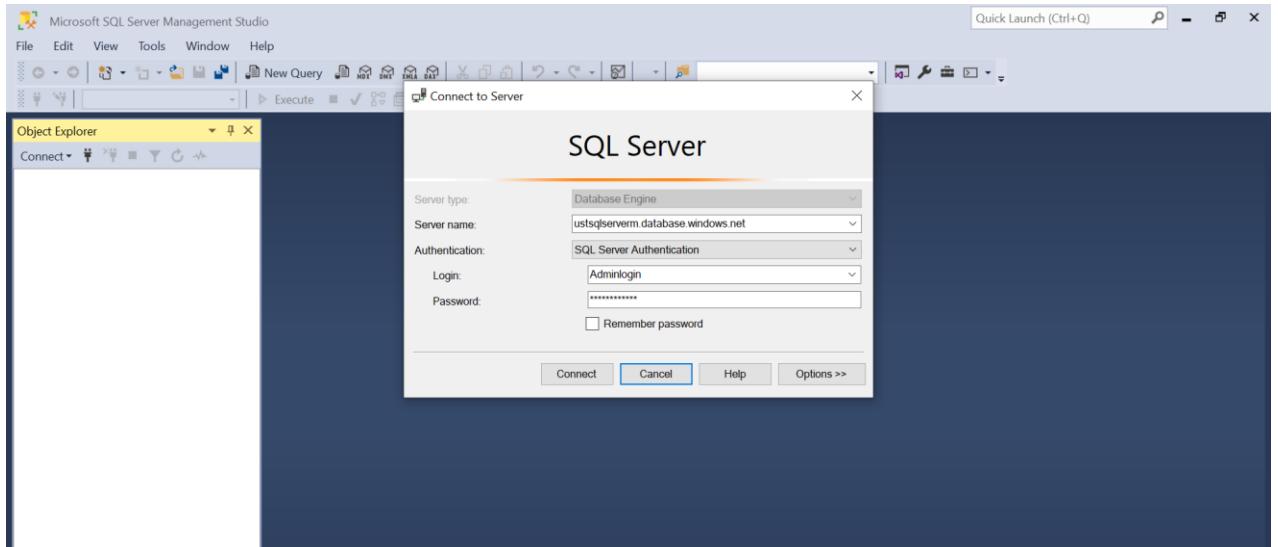
22) Enter username and password

23) trust

CREATE VIRTUAL MACHINE:((LINUX))

- 1) Go to the Resource Group(RG_UST_Training) ➔ +Create ➔ search : Virtual Machine ➔ Click create ➔ click virtual machine ➔ VM name: ustvm, Region : South central

- 2)** Availability Option : No infrastructure redundancy required (As of now because we r not creating for production)
- 3)** Security type : Trusted launch virtual machine
Image → configuration file UBUNTU server 22.04 LTS
- 4)** Spot instance : B1'S to increase use **DS1_V2**
- 5)** SSH:
USER NAME : MonishaB
Key value : monisha(anyname we can give)
- 6)** Public Inbound ports: Allow selected ports
- 7)** Select inbound ports: SSH:22
- 8)** Os disk type : Premium SSD
- 9)** Key → platform managed key
- 10)** Default
- 11)** Next
- 12)** Tags
Project: UST_Training
Owner : Monisha
- 13)** Review and create
- 14)** Click Create
- 15)** Download pem file store in a local
How to connect :LINUX
 - 1)Start
 - 2)connect
 - 3)Search puttygen:
 - 4)Load → choose .pem file → open → save private key
 - 5)Search putty :
 - 6)Ssh → auth → credential → browse → ppk file → open
 - 7)Go to session → port number 22 → IPAddress: username@ipaddress
 - 8)Open
 - 9)Connect once
 - 10)Type hostname and check hostname
- 11)Stop(After using linux vm stop it)



AZURE SQL:

TYPES:

- 1) SQL with vm → (SQL Server + vm) **IAAS**
- 2) Managed Instance → (Fully managed, SQL Server) **PAAS**
- 3) SQL Database → (Complete SQL DB) **PAAS**
 - 1) Singleton DB
 - 2) Elastic Pool

LAB AZURE SQL:

- 1) Search Azure SQL
- 2) Click create → AZURE SQL
- 3) Choose SQL Databases → single database → create
- 4) Choose Resource group → RG_UST_Training
- 5) Database name : USTSqldb
- 6) Server : Create New : → Server Name (lowercase should be unique): ustsqlserverm
- 7) Location : (US) East US
- 8) Authentication method : 1)SQL Authentication
 - 2) Azure active directory auth → Microsoft entra authentication
→ use both sql & Microsoft entra auth
- 9) Choose sql authentication Server Admin Name : Adminlogin , Password : Monisha@ishu
- 10) Ok
- 11) Want to use sql Elastic Pool : No
- 12) Work Environment : Tick Development,
Production.
- 13) Compute + Storage : Service & compute tier : choose basic (DTU'S 5 DATA SIZE 2GB) → Apply
Standard
Premium
VCore

- 14) Backup Storage Redundancy : Tick LRS,
GRS,
ZRS.
- 15) Connectivity Method : Tick Public Endpoint → allow azure service : yes , hybrid benefit: yes
No access,
Private Endpoint
- 16) Connection Policy: Tick Default (uses ReDirect Policy)
Proxy
Redirect
- 17) Microsoft Defender for SQL: Enable defender → Tick Not now,
Start Free Trial
- 18) Next
- 19) Additional settings: Data source → Use Existing Data → Choose Sample → ok
- 20) Tags
Project : ust_training
Owner : monisha
- 21) Review + create
- 22) Create
- AZURE SQL QUERING:**
- 1) Overview → setserverfirewall → +create → 0.0.0.0 254.254.254.254 → save
 - 2) Query editor → username : , password : → ok
 - 3) +new query → Select * from [SalesLT].[Address]
 - 4) Run

HOW TO CONNECT AZURE SQL TO SSMS :

- 1) In azure sql → overview → copy server name
 - 2) Open SSMS Software → + connect → Database Engine → server : paste server name → SQL Server Authentication → server_name: password: from azure portal adminlogin and password
 - 3) Connect
- DIFFERENT MASKING RULE:
HOW TO ADD MASKING RULE:

The screenshot shows the Azure portal interface with the URL [Home > USTSqlMoniDB \(ustsqlserverm/USTSqlMoniDB\) | Dynamic Data Masking](#). A modal window titled "Add masking rule" is open. The "Mask name" field contains "SalesLT.Customer.NameStyle". Under "Select what to mask", "Schema" is set to "SalesLT" and "Table" is set to "Customer". The "Column" dropdown shows "NameStyle (bit)". In the "Select how to mask" section, "Masking field format" is set to "Number (random number range)". Below this, "From" is set to "0" and "To" is set to "100". There are "Add" and "Delete" buttons at the top left of the modal.

*****28/9/2023*****

USE CASE 1:

ADF

BLOB =====> DATALAKE

E T load

CREATE DATA FACTORY:

- 1) Go to resource group ➔ + create ➔ search : data factory ➔ create ➔ click data factory
- 2) Name(unique) : Adfmonisha
- 3) As of now Repository type : Tick Git later. Tick (Repository type : tick github or azure
- 4) Connect via : Tick public endpoint,
Private endpoint
- 5) Tags :
Project ➔ ust_training
Owner ➔ Monisha
- 6) Review + Create
- 7) Create

We need blob storage and datalake service to copy data via data factory: blob to data lake

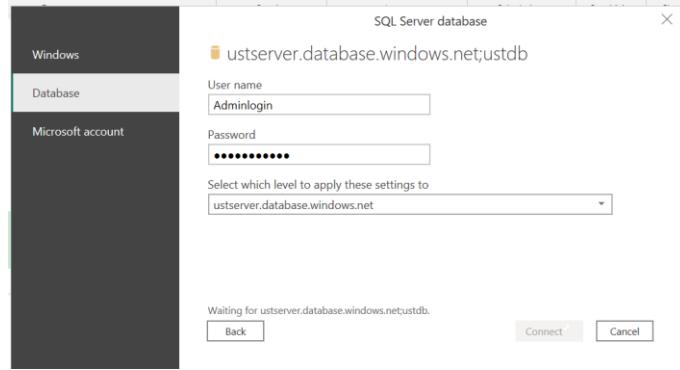
- 1) Click container in blob ➔ +container ➔ Create container named raw in blob storage.
Name:raw ➔ create
- 2) Click container in datalake ➔ +container ➔ Create container named refined in default storage.
- 3) Name:refined ➔ create
- 4) In blob storage ➔ click upload ➔ browse for files ➔ choose person.csv file ➔ open ➔ upload
- 5) Go to data factory(adfmonisha) ➔ Overview ➔ click launch studio

- 6) Click Manage → Linked service → + New → search:blob → choose Azure Blob Storage → continue
- 7) Name (LS is best practice): LS_AzureBlobStorage
- 8) **Connect via integration runtime :** Default (AutoResolveIntegrationRuntime)
- 9) **Authentication type:** Account Type → Connection String →
- 10) **Tick** From Azure subscription(((Enter manually to connect with other account)))
- 11) Azure Subscription : Choose pay as you go
- 12) Storage account name : Blob Storage name(absmonishaustr)
- 13) Test connection:To linked Service
- 14) Click create
- 15) Click Manage → Linked service → + New → search:data → choose Azure Data Lake Storage Gen 2 → continue
- 16) Name (LS is best practice): LS_AzureDataLakeStorage
- 17) **Connect via integration runtime :** Default (AutoResolveIntegrationRuntime)
- 18) **Authentication type:** Account Type → Connection String →
- 19) **Tick** From Azure subscription(((Enter manually to connect with other account)))
- 20) Azure Subscription : Choose pay as you go
- 21) Storage account name : Data Lake Storage name(adlsmonishaustr)
- 22) Test connection:To linked Service
- 23) Click create
- 24) Create DATASETS → Author → dataset → ... → new dataset → search:blob → choose Azure Blob Storage → continue
- 25) Select format : choose csv → continue
- 26) Name(DS is a best practice) : DS_Source_raw
- 27) LinkedService: LS_AzureBlobStorage
- 28) Browse → select raw → person.csv → ok → ok
- 29) ... → new dataset → search:data → Azure Data Lake Storage Gen 2 → continue
- 30) Choose json → Name : DS_Target_refined → Linkedservice: LS_AzureDataLakeStorage
- 31) Browse → select refined → ok → ok
- 32) Pipelines → ... → Name: PL_Raw_to_Refined → Activities → search:copy(Becoz here we are going to copy person.csv from blob to person.json to data lake) → drag and drop copy data → give Name : Raw_to_refined →
- 33) Source → Source dataset : DS_Source_raw
- 34) Sink → Sink dataset : DS_Target_refined
- 35) Click Validate → it will validate and resolve if any error is there means
- 36) Click debug → in debug we can do validation without saving (publish) files once debug is succeeded
- 37) Click publish all → then click Add trigger → trigger now → to trigger pipeline
- 38) Add Trigger → new/edit → schedule , tumbling window, storage events, custom events

HOW TO CONNECT POWERBI WITH AZURE SQL DATABASE:

- 1) Go to PowerBI → Get Data → more → Azure → choose Azure SQL database → connect

- 2) In portal → go to azure sql (resource_group: RG_UST_Training, ustldb:azure sql) → overview → copy server name and paste in power bi server → database name also copy from portal and paste in powerBi → ok
- 3) Choose database → username: Adminlogin, password: Welcome@123 → connect



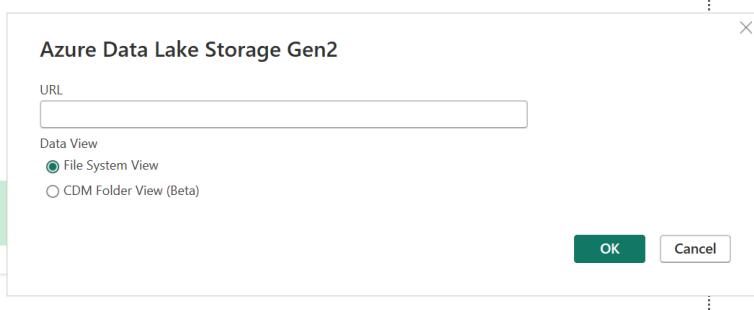
- 4) Connect mobile data(UST_SECURE is not)

HOW TO CONNECT DATALAKE WITH POWERBI:

- 1) Go to data lake storage → endpoint → copy datalake storage account 2nd url → ok

- 2) In Power Bi → Get Data → Get More → Azure → Azure Data Lake Storage Gen 2 → connect

3) Paste the datalake url ➔ click ok



- 4) Authentication : Organisational Account,
Account key,
SAS ➔ Access choose ➔ generate SAS ➔ Token
5) Choose Account key ➔ copy and paste account key ➔ connect
6) Combine or load or transform data

AZURE COSMOS DB:

The screenshot shows the Microsoft Azure Marketplace search results for 'cosmos db'. The search bar at the top has 'cosmos db' entered. Below the search bar, there are filters for 'Service Providers' (checkbox for 'Azure services only') and 'Publisher name : All'. The results section displays four items:

- Azure Cosmos DB** (Microsoft): Globally-distributed, multi-model database service.
- Azure Cosmos DB Reserved Capacity** (Microsoft): Reserved instances (RIs) for Cosmos DB significantly reduce your Cosmos DB costs compared to pay-as-you-go prices.
- Azure Cosmos DB for MongoDB** (Microsoft): Quickly setup a MongoDB-compatible API database with built-in security and scalability.
- Alpaqa Studio - Cosmos DB Data Studio** (APC Labs GmbH): The professional data studio for Microsoft Azure Cosmos DB. It includes a 'Free trial' badge.

- Go to resource group ➔ search azure cosmos db ➔ create ➔ azure cosmos db
- Azure cosmos db for nosql ➔ create
- Account Name : moniustcosmosdb

- Location: Canada East(Account level region)
- Capacity mode : tick provisioned throughput,
Serverless
Limit total account tck it
- Next
- GRS Disable
- Next
- Connectivity network : tick all network(default)
- next
- Backup policy: periodic(default)
- Next'
- Data encryption: tick service managed key
- Tags
- Review +create
- Create

How to create container and item, database:

- Click data explorer ➔ Launch quick start ➔ create new ➔ SampleDB (Default)
- Tick share throughput across container
- Database throughput: tick Autoscale ➔ 1000
- Container id : SampleContainer
- Partition key: /categoryid
- Click ok
- Click items ➔ new item ➔ paste item json from cosmosdbjsonformat ➔ new sql query:

```
SELECT *
FROM c
WHERE CONTAINS(c.name, "Helmet")
```

```

1   "name": "Road Helmet,45",
2   "id": "123456789",
3   "categoryID": "123456789",
4   "SKU": "AB-1234-56",
5   "description": "The product called \"Road Helmet,45\"",
6   "price": 48.74,
7   "_rid": "21EaANPG2zUoAQA=AAAAAA",
8   "_self": "dbs/21EaAAA=/colls/21EaANPG2zU=/docs/21EaANPG2zUoAQA=AAAAAA/",
9   "_etag": "\"0000a75-0000-0000-0000-651bbd720000\"",
10  "_attachments": "attachments/",
11  "_ts": 1696316786

```

MOVING DATA FROM ONE CONTAINER TO ANOTHER CONTAINER USING DATA FACTORY:

- ... → New container →

New Container

S(c.name,"Helmet")

Results	Query Stats
1 - 4	
<pre>{ "id": "601A5234-644D-4B83-9FDB-326C22C10510", "categoryID": "14A1AD50-59EA-4B63-A1B9-67B077783B0E", "categoryName": "Accessories, Helmets", }</pre>	

- Container id : HelmetData, partition key: /itemid → click ok

-

The screenshot shows the Microsoft Azure Data Explorer interface for the 'moniustcosmosdb' database account. On the left, the navigation menu includes 'Cost Management', 'Quick start', 'Notifications', 'Data Explorer' (which is selected), 'Settings' (with options like 'Features', 'Replicate data globally', 'Default consistency', 'Backup & Restore', 'Networking', 'CORS', 'Dedicated Gateway', and 'Keys'), and 'Keys'. The main workspace displays the 'NOSQL API' section under 'DATA'. A 'SampleDB' database is selected, showing its 'Scale' and 'Items' sections. A 'SampleContainer' is expanded, showing 'Stored Procedures', 'User Defined Functions', 'Triggers', and 'NOTEOBOOKS' (which is currently not available). On the right, a 'New Container' dialog is open, prompting for a 'Database id' (set to 'SampleDB'), 'Container id' (set to 'HelmetData'), and 'Partition key' (set to '/item'). Below the dialog, there's a checkbox for 'Provision dedicated throughput for this container' and a 'Unique keys' section with a '+ Add unique key' button. The bottom of the screen shows the Windows taskbar with various pinned icons.

• Go to data factory → Linked service → cosmosdb nosql → continue

The screenshot shows the Microsoft Azure Data Factory interface. On the left, a sidebar lists various options: General, Factory settings, Connections (selected), Integration runtimes, Microsoft Purview, Source control, Git configuration, ARM template, Author, Triggers, Global parameters, Data flow libraries, and Security.

The main area is titled "Linked services" and contains a search bar with "cosmos" and a filter bar with "Annotations: Any". It displays four items:

Name	Type
LS_AzureBlobStorage	Azure Blob Storage
LS_AzureDataLakeStorage	Azure Data Lake Storage
LS_AzureSqlDatabase	Azure SQL Database
LS_OnPremServer	File system

On the right, a modal window titled "New linked service" is open. It shows the "Data store" tab selected, with "Azure Cosmos DB for NoSQL" highlighted in the search results. The modal includes tabs for Compute, Data store, Database, File, Generic protocol, NoSQL, and Services and apps. Below the tabs, there are two options: "Azure Cosmos DB for MongoDB" and "Azure Cosmos DB for NoSQL".

The "New linked service" configuration page has the following fields:

- Authentication type:** Account key (selected)
- Account selection method:** From Azure subscription (selected)
- Azure subscription:** Pay-As-You-Go (45f63aab-e46c-4be8-bb77-43fe31240084)
- Azure Cosmos DB account name:** moniustcosmosdb
- Database name:** SampleDB
- Additional connection properties:** New

At the bottom of the configuration page are "Create", "Back", "Test connection", and "Cancel" buttons.

- Create dataset

https://adf.azure.com/en/authoring/pipeline/PL_OnPrem_to_Azure?factory=%2Fsubscriptions%2F45f6...

Microsoft Azure | Data Factory > AdfMonisha

Search factory and documentation

User4@hk0178210@gmail.onmicrosoft.com DEFAULT DIRECTORY

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us en...

Data Factory > Validate all > Publish all

Factory Resources

- + Filter resources by name
- Pipelines: PL_OnPrem_to_Azure (1)
- Change Data Capture (preview): 0
- Datasets: DS_AzureSQL, DS_Deleteblob, DS_DeleteDL, DS_OnPrem, DS_Raw, DS_Refined (6)
- Data flows: 0
- Power Query: 0

Activities

- + Search activities
- < Valid
- < Copy
- < Other
- < Pipeline
- < Pipeliner
- < All
- < Show in catalog
- < Activities

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

Select a data store

cosmos

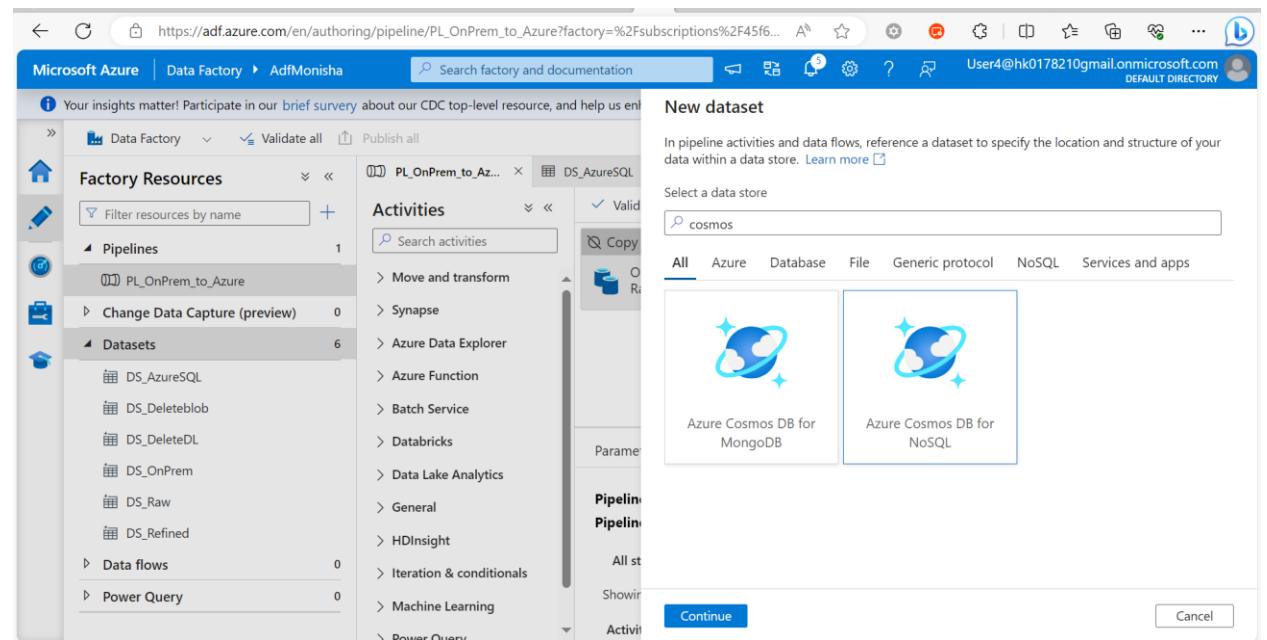
All Azure Database File Generic protocol NoSQL Services and apps

New dataset

Azure Cosmos DB for MongoDB

Azure Cosmos DB for NoSQL

Continue Cancel



Set properties

Name

Linked service *

Container

  Edit

Import schema

 From connection/store None

Set properties

Name

Linked service *

Container

   Edit

Import schema

 From connection/store None

Microsoft Azure | Data Factory > AdfMonisha

Your insights matter! Participate in our [brief survey](#) about our CDC top-level resource, and help us enhance your experience.

Validate all Publish all

PL_OnPrem_to_Azure DS_AzureSQL DS_SampleContainer DS_Helmet_Target pipeline1

Validate Validate copy runtime Debug Add trigger

Copy data

Cosmosdb C1 to Cosmosdb C2

Source dataset * DS_SampleContainer

Use query Container Query

Query

```
SELECT *  
FROM c  
WHERE CONTAINS(c.name,"Helmet")
```

Page size

Properties

Name * pipeline1

Description

Annotations

General Related

Name * pipeline1

Description

Annotations

Preview experience Off

Microsoft Azure https://adf.azure.com/en/authoring/pipeline/pipeline1?factory=%2Fsubscriptions%2F45f63aab-e46c... User4@hk01782@gmail.onmicrosoft.com DEFAULT DIRECTORY

Your insights matter! Participate in our [brief survey](#) about our CDC top-level resource, and help us enhance your experience.

Validate all Publish all

PL_OnPrem_to_Azure DS_AzureSQL DS_SampleContainer DS_Helmet_Target pipeline1

Validate Validate copy runtime Debug Add trigger

Copy data

Cosmosdb C1 to Cosmosdb C2

Sink dataset * DS_Helmet_Target

Write behavior Insert

Write batch timeout

Write batch size

Max concurrent connections

Disable performance metrics

Properties

Name * pipeline1

Description

Annotations

The image displays two screenshots of the Microsoft Azure portal. The top screenshot shows the Data Factory pipeline run status for a 'Copy data' activity named 'Cosmosdb C1 to'. The pipeline run ID is 85d3a82-6028-4f64-bbfa-4ce50c2f83b1, and the status is 'Succeeded'. The bottom screenshot shows the Azure Cosmos DB Data Explorer interface for the 'moniustcosmosdb' account. It lists the 'SampleDB' database, 'Scale' container, and 'HelmetData' item. A query results pane shows a single document with the following JSON structure:

```

{
  "id": "601A5234-644D-4B83-9FDB-326C22C1051D",
  "categoryId": "14A1AD50-59EA-4B63-A189-67B077783B0E",
  "categoryName": "Accessories, Helmets"
}

```

HOW TO CONNECT COSMOSDB TO LOCAL:

Inside vm

- Download and install vs code: [Download Visual Studio Code - Mac, Linux, Windows](#)
- Download .net runtime *64
- <https://dotnet.microsoft.com/en-us/download/dotnet/7.0>
- Download .net SDK
- In desktop ➔ create folder named : cosmos

- <https://github.com/microsoftlearning/dp-420-cosmos-db-dev> from this link if u have git clone this file if not download zip and extract that file inside Cosmos folder
- Open that extracted file with VS Code
- Go to 04-sdk-Connect ➔ open script.cs ➔

put below code in script.cs

```
using System;
using System.Linq;

using Microsoft.Azure.Cosmos;

string endpoint = "<cosmos-endpoint>";
string key = "<cosmos-key>";

CosmosClient client = new (endpoint, key);

AccountProperties account = await client.ReadAccountAsync();

Console.WriteLine($"Account Name:\t{account.Id}");
Console.WriteLine($"Primary Region:\t{account.WritableRegions.FirstOrDefault()?.Name}");
```

- Go to cosmosdb ➔ key ➔ copy url
- `string endpoint = "https://moniustcosmosdb.documents.azure.com:443/";`
- copy primary key ➔ paste
- `string key =
 "FIOw6BoLOhS89PCff8nqYD7ELRn8RJ3ISWAUeQdBWRlAqp379JY1vswJoj99nwWD2iWX2sXyS
 W2zACDbXkpFEQ==";`
- save
- right click on script.cs ➔ open in integrated terminal ➔ paste the below command and run
dotnet add package Microsoft.Azure.Cosmos --version 3.22.1
- dotnet run

```

    04-sdk-connect > script.cs
    1  using System;
    2  using System.Linq;
    3
    4  using Microsoft.Azure.Cosmos;
    5
    6  string endpoint = "https://moniustcosmosdb.documents.azure.com:443/";
    7  string key = "FIOw6BoLohS89PCfF8nqYD7ELRn8RJ3ISwAueQdBWRiaqp379JY1vswJoj99nwD2iWx2sXySW2zACDbXkpFEQ==";
    8
    9  CosmosClient client = new (endpoint, key);

info : Package 'Microsoft.Azure.Cosmos' is compatible with all the specified frameworks in project 'C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect\app.csproj'.
info : PackageReference for package 'Microsoft.Azure.Cosmos' version '3.22.1' added to file 'C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect\app.csproj'.
info : Generating MSBuild file C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect\obj\app.csproj.nuget.g.props.
info : Generating MSBuild file C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect\obj\app.csproj.nuget.g.targets.
info : Writing assets file to disk. Path: C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect\obj\project.assets.json
log : Restored C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect\app.csproj
(in 52.46 sec).
● PS C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect> dotnet run
  Account Name: moniustcosmosdb
  Primary Region: Canada East
○ PS C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\04-sdk-connect> dotnet run

```

- HOW TO DO CRUD, READ/WRITE OPERATION:
- Open script.cs from 06-sdk-offline → paste script → change url and key → type dotnet run → check in cosmosdb to see new product container

using System;

using Microsoft.Azure.Cosmos;

```
string endpoint = "https://madhangicosmosdb.documents.azure.com:443/";
```

```
string key =
```

```
"QCh8V9HVhkDiePiYVRgezRIK04yFb2N6eX3yLmEA00jr9m3S23yZi98ProAmwLykbxXxLRzAnbLA
CDbfS3KFA==";
```

```
CosmosClient client = new CosmosClient(endpoint, key);
```

```
Database database = await client.CreateDatabaseIfNotExistsAsync("SampleDB");
```

```
Container container = await database.CreateContainerIfNotExistsAsync("products",
"/categoryId", 400);
```

```
Product saddle = new()
```

```
{
```

```
  id = "706cd7c6-db8b-41f9-aea2-0e0c7e8eb009",
```

```
  categoryId = "9603ca6c-9e28-4a02-9194-51cdb7fea816",
```

```
  name = "Road Saddle",
```

```
  price = 45.99d,
```

```

tags = new string[]
{
    "tan",
    "new",
    "crisp"
};

await container.CreateItemAsync<Product>(saddle);

```

The screenshot shows a Windows Remote Desktop Connection window for a machine named 'MonishaVMforSHIR'. The title bar says 'dp-420-cosmos-db-dev-main [Administrator]'. The left sidebar shows a project tree for 'DP-420-COSMOS-DB-DEV-MAIN' containing several folders like '04-sdk-connect', '05-sdk-offline', '06-sdk-crud', etc., and files like 'script.cs'. The 'script.cs' file is selected in the Explorer view. The main editor area shows a C# script for creating a database and container in Azure Cosmos DB. Below the editor is a terminal window showing the command 'dotnet run' being executed, which finds an item named 'Road Saddle'. The bottom status bar shows the path 'C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\10-paginate-results-sdk'.

```

script.cs 04-sdk-connect script.cs 06-sdk-crud script.cs 10-paginate-results-sdk
File Edit Selection View Go Run ...
dp-420-cosmos-db-dev-main [Administrator]
EXPLORER
DP-420-COSMOS-DB-DEV-MAIN
script.cs
05-sdk-offline
06-sdk-crud
bin
obj
app.csproj
product.cs
script.cs
07-sdk-batch
08-sdk-bulk
09-execute-query-sdk
10-paginate-results-sdk
bin
obj
app.csproj
product.cs
script.cs
12-custom-index-policy
13-change-feed
16-measure-performance
17-denormalize
20-sdk-regions
21-sdk-consistency-model
script.cs
04-sdk-connect > script.cs
1  using System;
2  using Microsoft.Azure.Cosmos;
3
4  string endpoint = "https://moniustcosmosdb.documents.azure.com:443/";
5  string key = "FIOw6BoL0hs89PCFF8nqYD7ELRn8RJ3ISWAUeQdBWRiaqp379JY1vswJoj99nwD2iWX2sXySw2zACDbXkpFEQ";
6
7  CosmosClient client = new CosmosClient(endpoint, key);
8
9  Database database = await client.CreateDatabaseIfNotExistsAsync("SampleDB");
10
11  Container container = await database.CreateContainerIfNotExistsAsync("products", "/categoryId", 400);
12
13  Product moni = new()
14  {
15      id = "706cd7c6-db8b-41f9-aea2-0e0c7e8eb009",
16      categoryId = "9603ca6c-9e28-4a02-9194-51cdb7fea816".
17
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\10-paginate-resu
● lts-sdk> dotnet run
Found item: Road Saddle
PS C:\Users\Monishavm\Desktop\Cosmos\dp-420-cosmos-db-dev-main\dp-420-cosmos-db-dev-main\10-paginate-resu
○ lts-sdk> []

```

- In 10-paginate-results-sdk, script.cs paste script → change url, key → dotnet run using System;
using Microsoft.Azure.Cosmos;

```
string endpoint = "https://madhangicosmosdb.documents.azure.com:443/";
```

```
string key =
"QCh8V9HVhkDiePiYVRrgezRIK04yFb2N6eX3yLmEA00jr9m3S23yZi98ProAmwLykbxXxLRzAnbLA
CDbfS3KFA==";
```

```
CosmosClient client = new CosmosClient(endpoint, key);
```

```

Database database = await client.CreateDatabaseIfNotExistsAsync("SampleDB");

Container container = await database.CreateContainerIfNotExistsAsync("products",
    "/categoryId");

using FeedIterator<Product> feed = container.GetItemQueryIterator<Product>(
    queryText: "SELECT * FROM products"
);

// Iterate query result pages
while (feed.HasMoreResults)
{
    FeedResponse<Product> response = await feed.ReadNextAsync();

    // Iterate query results
    foreach (Product item in response)
    {
        Console.WriteLine($"Found item:\t{item.name}");
    }
}

```

The screenshot shows the Azure Cosmos DB PowerShell interface. The title bar reads "MonishaVMforSHIR - 172.178.66.0:3389 - Remote Desktop Connection". The window contains three tabs: "script.cs 04-sdk-connect", "script.cs 06-sdk-crud", and "script.cs 10-paginate-results-sdk". The "10-paginate-results-sdk" tab is active, displaying C# code for paginating results. The code uses the Microsoft.Azure.Cosmos library to connect to a database and container, and then iterates through query results using a FeedIterator. The PowerShell command "dotnet run" is shown at the bottom, along with the output: "Found item: Road Saddle". The left sidebar shows the project structure under "DP-420-COSMOS-DB-DEV-MAIN", including files like "script.cs", "05-sdk-offline", "06-sdk-crud", "app.csproj", "product.cs", and "script.cs" (the current file being edited).

To add one more container and new database:

In 06-sdk, change database name to FourOctober and Container name to address.

```

using System;
using Microsoft.Azure.Cosmos;

```

```

string endpoint = "https://moniustcosmosdb.documents.azure.com:443/";
string key =
"FI0w6BoLOhS89PCff8nqYD7ELRn8RJ3ISWAUeQdBWRlAqp379JY1vswJoj99nwWD2iWX2sXySW2zACDb
XkpFEQ==";

CosmosClient client = new CosmosClient(endpoint, key);

Database database = await client.CreateDatabaseIfNotExistsAsync("FourOctober");

Container container = await database.CreateContainerIfNotExistsAsync("address",
"/categoryId", 400);

Product moni = new()
{
    id = "406cd7c6-db8b-41f9-aea2-0e0c7e8eb012",
    categoryId = "2745ca6c-9e28-4a02-9194-51cdb7fea815",
    name = "Address",
    price = 45.99d,
    tags = new string[]
    {
        "tan",
        "new",
        "crisp"
    }
};

await container.CreateItemAsync<Product>(moni);

```

To Query go to 10-sdk,

```

using System;
using Microsoft.Azure.Cosmos;

string endpoint = "https://moniustcosmosdb.documents.azure.com:443/";
string key =
"FI0w6BoLOhS89PCff8nqYD7ELRn8RJ3ISWAUeQdBWRlAqp379JY1vswJoj99nwWD2iWX2sXySW2zACDb
XkpFEQ==";

CosmosClient client = new CosmosClient(endpoint, key);

Database database = await client.CreateDatabaseIfNotExistsAsync("FourOctober");

```

```

Container container = await database.CreateContainerIfNotExistsAsync("address",
"/categoryId");

using FeedIterator<Product> feed = container.GetItemQueryIterator<Product>(
    queryText: "SELECT * FROM address"
);

// Iterate query result pages
while (feed.HasMoreResults)
{
    FeedResponse<Product> response = await feed.ReadNextAsync();

    // Iterate query results
    foreach (Product item in response)
    {
        Console.WriteLine($"Found item:\t{item.name}");
    }
}

```

To add one more item, in 06-sdk-crud, change item name and name:

```

using System;
using Microsoft.Azure.Cosmos;

string endpoint = "https://moniustcosmosdb.documents.azure.com:443/";
string key =
"FI0w6BoLOhS89PCFF8nqYD7ELRn8RJ3ISWAUeQdBWRlAqp379JY1vswJoj99nwWD2iWX2sXySW2zACDb
XkpFEQ==";

CosmosClient client = new CosmosClient(endpoint, key);

Database database = await client.CreateDatabaseIfNotExistsAsync("FourOctober");

Container container = await database.CreateContainerIfNotExistsAsync("address",
"/categoryId", 400);

Product monisha = new()
{
    id = "506cd7c6-db8b-41f9-aea2-0e0c7e8eb012",
    categoryId = "1745ca6c-9e28-4a02-9194-51cdb7fea815",
    name = "Mayiladuthurai",
    price = 45.99d,
    tags = new string[]
{
}

```

```

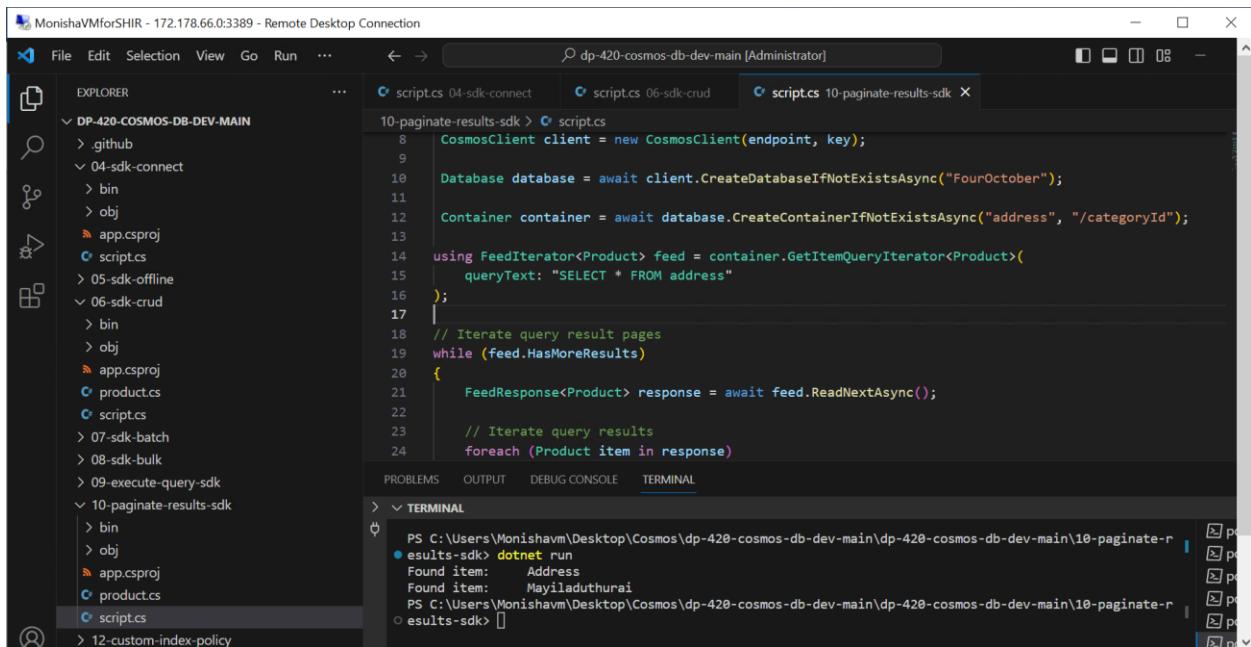
        "tan",
        "new",
        "crisp"
    }

};

await container.CreateItemAsync<Product>(monisha);

```

then if u run query: we can see two items

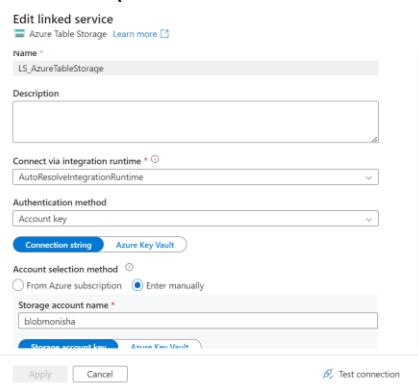


The screenshot shows a Windows Remote Desktop Connection window titled 'MonishaVMforSHIR - 172.178.66.0:3389 - Remote Desktop Connection'. The taskbar at the bottom has icons for File Explorer, Task View, Start, Task Manager, and File History.

The main area displays a code editor with three tabs: 'script.cs 04-sdk-connect', 'script.cs 06-sdk-crud', and 'script.cs 10-paginate-results-sdk'. The 'script.cs 10-paginate-results-sdk' tab is active, showing C# code for interacting with a CosmosDB container. Below the code editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab shows the command 'dotnet run' being executed, followed by output indicating two items found in the database: 'Address' and 'Mayiladuthurai'.

TO LOAD CONTAINER DATA TO TABLE STORAGE:

1. Go to blob storage ➔ Click Table
2. +Table ➔ Table Name: Monisha Table
3. Go to data factory ➔ Create Linked service ➔ choose Table Storage ➔ Table Name
➔ Create (use datalake or blob to create table)



5) Create a dataset → Table storage → create

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (PL_Cosmos_to_Table_datalake, PL_CosmosDb, PL_OnPrem_to_Azure), 'Datasets' (DS_AzureSQL, DS_AzureTable, DS_DataLake_Target, DS_Deleteblob, DS_DeleteDL, DS_Helmet_Target, DS_OnPrem, DS_Raw), and 'Change Data Capture (preview)'.

The main workspace displays a 'Copy data' activity. It has two 'Cosmos to Table Storage' source nodes and one 'Cosmos to Data Lake' sink node. The 'Source' tab of the activity configuration pane is selected, showing the 'Source dataset' dropdown set to 'DS_SampleContainer'. The 'Query' section contains the following SQL:

```
SELECT *
FROM c
WHERE CONTAINS(c.name,"Helmet")
```

The 'Connection' tab of the dataset configuration pane is selected, showing the 'Linked service' dropdown set to 'LS_AzureTableStorage' and the 'Table' dropdown set to 'MonishaTable'.

6) Another dataset → cosmos nosql →

The screenshot shows the Azure Data Factory Studio interface. On the left, the 'Factory Resources' pane is open, displaying various datasets and data flows. A specific dataset, 'DS_SampleContainer', is selected. On the right, the details for this dataset are shown under the 'Connection' tab. The connection is set to 'LS_CosmosDbNoSql' and the container is 'SampleContainer'. There are tabs for 'Schema' and 'Parameters' as well.

7)
8) Query →

```
SELECT *
FROM c
WHERE CONTAINS(c.name,"Helmet")
```

9)sink → Ds_AzureTable

10)debug

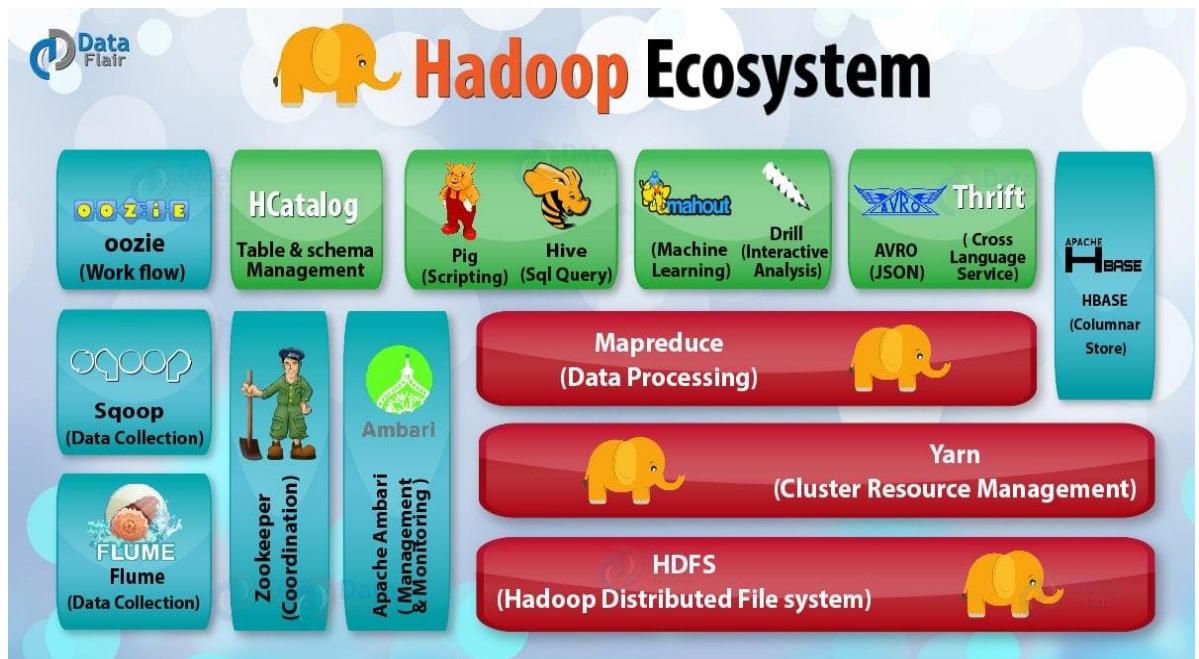
11) to check in table data stored or not : go to datalake → click Storage browser → tables
→ click on table name → validate data by checking data with cosmos

The screenshot shows the Microsoft Azure Storage browser interface. The left sidebar shows navigation options like Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, and Storage browser. The Storage browser is currently selected. In the main area, the 'Tables' section is highlighted. A table titled 'MonishaTable' is displayed, showing four rows of data. The columns are PartitionKey, RowKey, Timestamp, and _attachments. The data is as follows:

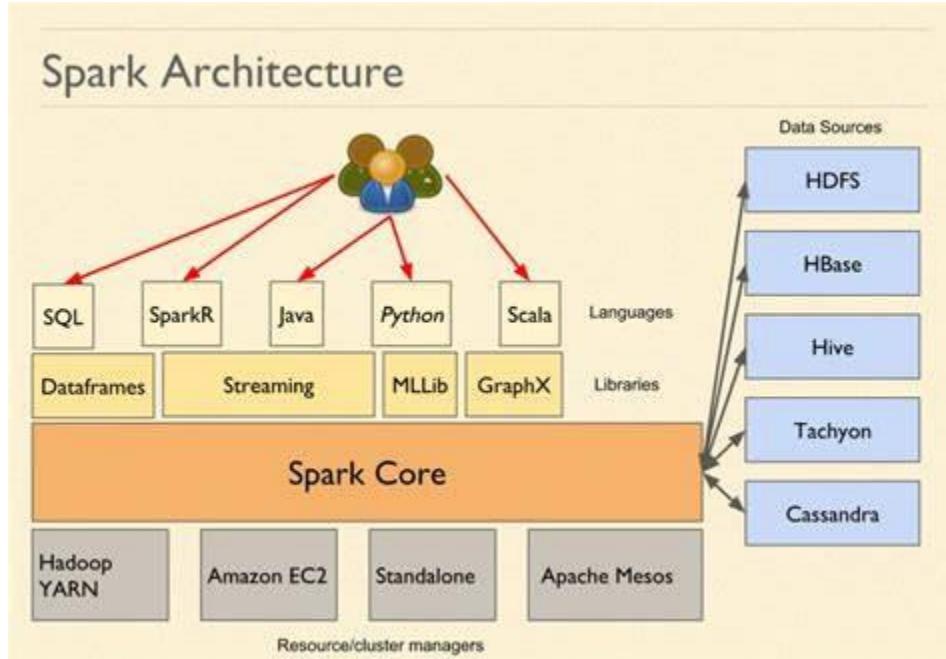
PartitionKey	RowKey	Timestamp	_attachments
DefaultPartitionKey	0a3a90fc-0b9b-4133-95...	2023-10-04T07:06:12.59...	attachments/
DefaultPartitionKey	8b073464-2262-4e2f-85...	2023-10-04T07:06:12.59...	attachments/
DefaultPartitionKey	8c6f5982-5196-4256-b1...	2023-10-04T07:06:12.59...	attachments/
DefaultPartitionKey	ca401015-96ff-481a-99...	2023-10-04T07:06:12.59...	attachments/

BIG DATA:

Hadoop Ecosystem:



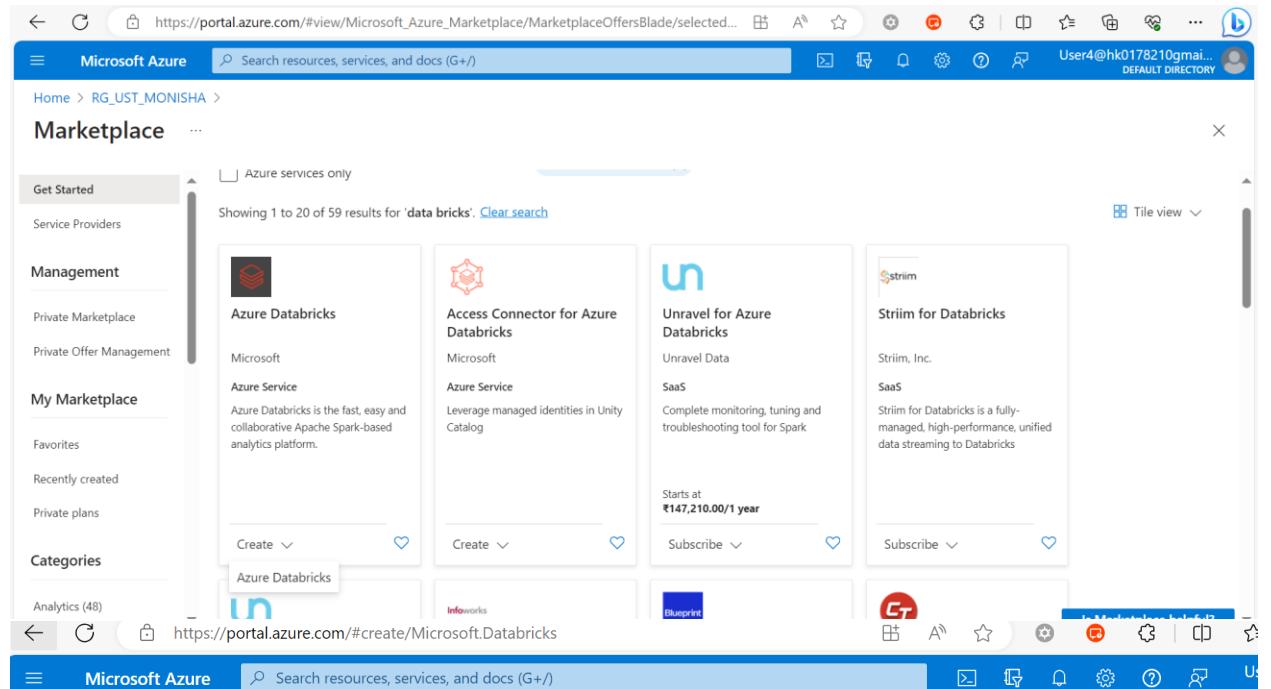
Spark Architecture:



HOW TO CREATE DATA BRICKS:

- Go inside Resource Group ➔ click +create
- Workspace Name : adbmonisha
- Region: west India

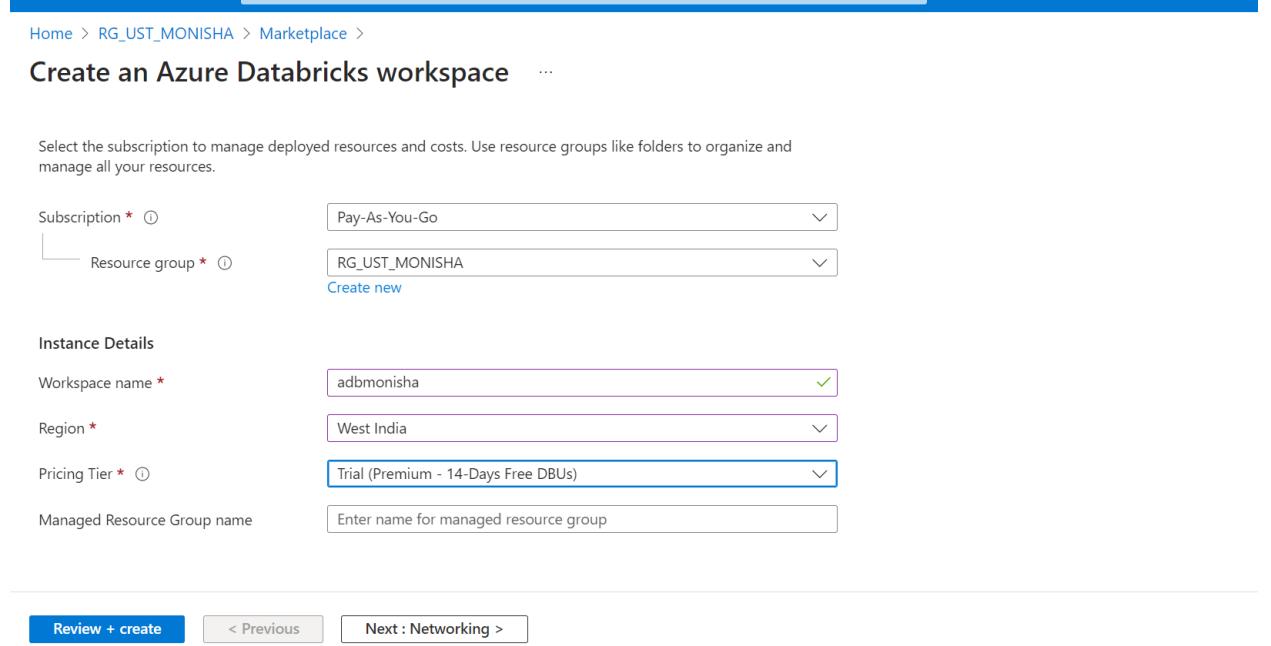
- Pricing Tier : Trial (as we are testing)
- Rest everything go with default
- Tags: Project: ust_training , Owner: Monisha
- Review+ create
- create



The screenshot shows the Microsoft Azure Marketplace search results for 'data bricks'. The search bar at the top has 'data bricks' entered. Below it, there are four cards displayed:

- Azure Databricks** by Microsoft: Description: Azure Databricks is the fast, easy and collaborative Apache Spark-based analytics platform. Buttons: Create, Blueprint.
- Access Connector for Azure Databricks** by Microsoft: Description: Leverage managed identities in Unity Catalog. Buttons: Create, Blueprint.
- Unravel for Azure Databricks** by Unravel Data: Description: Complete monitoring, tuning and troubleshooting tool for Spark. Starts at ₹147,210.00/1 year. Buttons: Subscribe.
- Striim for Databricks** by Striim, Inc.: Description: Striim for Databricks is a fully-managed, high-performance, unified data streaming to Databricks. Buttons: Subscribe.

On the left sidebar, under 'My Marketplace', the 'Create' button is highlighted over the 'Azure Databricks' card.



The screenshot shows the 'Create an Azure Databricks workspace' configuration page. It includes sections for 'Subscription' (Pay-As-You-Go), 'Resource group' (RG_UST_MONISHA), 'Instance Details' (Workspace name: adbmonisha, Region: West India, Pricing Tier: Trial (Premium - 14-Days Free DBUs)), and 'Managed Resource Group name' (Enter name for managed resource group). At the bottom, there are navigation buttons: 'Review + create', '< Previous', and 'Next : Networking >'.



Microsoft Azure



Search resources, services, and docs (G+/)

[Home](#) > [RG_UST_MONISHA](#) > [Marketplace](#) >

Create an Azure Databricks workspace

✓ Validation Succeeded

[Basics](#) [Networking](#) [Encryption](#) [Tags](#) [Review + create](#)

Summary

Basics

Workspace name	adbmonisha
Subscription	Pay-As-You-Go
Resource group	RG_UST_MONISHA
Region	West India
Pricing Tier	trial
Managed Resource Group name	

Networking

Deploy Azure Databricks workspace with [No](#)

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

HOW TO CREATE COMPUTE:

1. LAUNCH workspace ➔ COMPUTE ➔

2.

https://adb-89634/056//2986.6.azuredatabricks.net/?o=89634/056//2986#setting/clusters

Microsoft Azure | databricks Search data, notebooks, recents, and more... CTRL + P

Compute

All-purpose compute Job compute SQL warehouses Pools Policies

Filter compute you ... Created by

State ↑ Name Policy Runtime Active ...

No compute

Create compute to run workloads from your notebooks and compute configurations

Create compute

2.

https://adb-896347056772986.6.azuredatabricks.net/?o=896347056772986#create/cluster

Microsoft Azure | databricks Search data, notebooks, recents, and more... CTRL + P adbmonisha user4@hk0178210@gmail.onmicrosoft.com

Compute > New compute > UI preview Send feedback

ADB Moni cluster

Access mode Single user access

Single user User4

Performance

Databricks runtime version

Runtime: 13.3 LTS (Scala 2.12, Spark 3.4.1)

Use Photon Acceleration

Node type

Standard_DS3_v2 14 GB Memory, 4 Cores

Terminate after 15 minutes of inactivity

Tags

Create compute Cancel

1 Driver 14 GB Mem 13.3.x-scala2.12

Photon Standard_DS3_v2 1.5 DBU/h

3.

Another way of querying:

The screenshot shows the Databricks Catalog Explorer interface. On the left sidebar, under the 'Catalog' section, there is a tree view of databases: 'hive_metastore', 'default', 'samples', 'default', and 'nyctaxi'. Under 'nyctaxi', the 'trips' table is selected. On the right, a detailed view of the 'trips' table is shown with columns: 'tpep_pickup_datetime', 'tpep_dropoff_datetime', 'trip_distance', 'fare_amount', 'pickup_zip', and 'dropoff_zip'. A yellow box highlights the 'Notebook' button in the top right corner of the table details panel.

To create notebook → new → notebook

The screenshot shows a Databricks Notebook titled 'ADB Moni 1St NoteBook' in Python. The notebook has two cells: 'Cmd 1' and 'Cmd 2'. In 'Cmd 1', the code `print("Hi moni, Welcome!!!!")` is run, and the output 'Hi moni, Welcome!!!!' is displayed. In 'Cmd 2', the SQL command `SELECT getdate()` is run, and the result is a single row: 'getdate()' with value '2023-10-05T07:02:41.148+0000'. A yellow box highlights the 'Run Cell' button in the top right corner of the notebook interface.

To run cell : run cell Or shift + enter

The screenshot shows a Databricks Notebook with a context menu open over the first cell. The menu options are 'Run Cell', 'Run All Above', and 'Run All Below'. A yellow box highlights the 'Run Cell' option.

Shift+Ctrl+Enter to run selected text

To go to recent notebook:

Screenshot of the Microsoft Azure Databricks interface showing the 'Recents' section. A yellow box highlights the 'Recents' link in the sidebar. The main area shows a table of recent notebooks:

Name	Last viewed	Type
ADB Moni 1st NoteBook	2 minutes ago	Notebook

Screenshot of the Microsoft Azure Databricks interface showing the 'Catalog' section. A yellow box highlights the 'Catalog' link in the sidebar. The main area shows the 'hive metastore' catalog structure:

- hive metastore
 - default
 - samples

Click create or modify table →

Screenshot of the Microsoft Azure Databricks interface showing the 'Add data' page for creating or modifying a table from file upload. A yellow box highlights the 'Add data' button in the top right corner. The main area shows a form for uploading files:

Add data > Create or modify table from file upload Preview

Drop one or more files here, or browse
Maximum of 10 files and total upload size of 256MB

Requires a SQL warehouse or a cluster with Databricks Runtime 10.3 and above

For file uploads greater than 256MB, please [upload to DBFS](#).

Click upload to dbfs →

The screenshot shows the Microsoft Azure Databricks interface. The left sidebar contains navigation links for Workspace, Recents, Catalog, Workflows, Compute, SQL (SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses), Data Engineering (Job Runs, Data Ingestion), and a New section. The main content area is titled "Add data > DBFS" and "Upload File". It includes a "DBFS Target Directory" input field with the value "/FileStore/tables/" and a "Select" button. Below this is a message stating "Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more". A large blue oval highlights the "Drop files to upload, or click to browse" area. The "Files" section shows a file named "Person.csv" with a green checkmark, a size of "0.3 KB", and a "Remove file" link. A message at the bottom says "✓File uploaded to /FileStore/tables/Person.csv". At the bottom right are buttons for "Create Table with UI" (highlighted with a yellow box) and "Create Table in Notebook".

Cluster : choose cluster(ADB Moni Cluster) → PREVIEW table → table name: person, tick first row, tick infer schema

Microsoft Azure | databricks

Add data > DBFS Specify Table Attributes

Table Name: person

Create in Database: default

File Type: CSV

Column Delimiter: ,

First row is header

Infer schema

Multi-line

Create Table

PersonID	LastName	FirstName	Address	City
1	Zoe	kl	USA	USA
2	Kal	kl	NY	NC
3	krm	fgfd	NY	NY
4	jay	bc	IN	IN
5	ajay	g	CAL	CAL
6	aman	ffb	PUN	PUN
7	kalnana	frv	AIIS	AIIS

```
%sql
SELECT * FROM `hive_metastore`.`default`.`person`;
```

Microsoft Azure | databricks

Explore hive_metastore.default.person Python

File Edit View Run Help Last edit was now Provide feedback

Cmd 1

```
1 %sql
2 SELECT * FROM `hive_metastore`.`default`.`person`;
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [PersonID: integer, LastName: string ... 3 more fields]

PersonID	LastName	FirstName	Address	City
1	Zoe	kl	USA	USA
2	Kal	kl	NY	NC
3	krm	fgfd	NY	NY
4	jay	bc	IN	IN
5	ajay	g	CAL	CAL
6	aman	ffb	PUN	PUN
7	kalnana	frv	AIIS	AIIS

14 rows | 0.68 seconds runtime

New result table: OFF

This result is stored as PySpark data frame `_sqldf` and in the IPython output cache as `Out[1]`. Learn more

OR

Spark.sql ("query").Show():

```
spark.sql("SELECT * FROM `hive_metastore`.`default`.`person`;").show()
```

The screenshot shows the Databricks workspace interface. On the left, there's a sidebar with various navigation options like Recents, Catalog, Workflows, Compute, SQL, and Dashboards. The main area is titled "Explore hive_metastore.default.person" and shows a command line interface (Cmd 1) with the following code and output:

```
1 spark.sql("SELECT * FROM `hive_metastore`.`default`.`person`;").show()
```

(1) Spark Jobs

PersonID	LastName	FirstName	Address	City
1	Zoe	kl	USA	USA
2	Kal	kl	NY	NC
3	krm	fgfd	NY	NY
4	jay	bc	IN	IN
5	ajay	g	CAL	CAL
6	aman	ffb	PUN	PUN
7	kalpana	fcv	AUS	AUS
8	sayali	gddg	ZIM	ZIM
9	shyam	ggd	RUS	RUS
10	baban	gdg	KOL	KOL
11	tarun	wr	USA	USA
12	ram	ad	NY	NC
13	morti	tyu	NY	NY
14	roy	wsa	IN	IN

OR

```
display(spark.sql("SELECT * FROM `hive_metastore`.`default`.`person`;"))
```

The screenshot shows the Databricks workspace interface. The sidebar and top navigation bar are identical to the previous screenshot. The main area is titled "Explore hive_metastore.default.person" and shows a command line interface (Cmd 1) with the following code and output:

```
1 display(spark.sql("SELECT * FROM `hive_metastore`.`default`.`person`;"))
```

(1) Spark Jobs

Table

PersonID	LastName	FirstName	Address	City
1	Zoe	kl	USA	USA
2	Kal	kl	NY	NC
3	krm	fgfd	NY	NY
4	jay	bc	IN	IN
5	ajay	g	CAL	CAL
6	aman	ffb	PUN	PUN
7	kalpana	fcv	AUS	AUS
8	sayali	gddg	ZIM	ZIM
9	shyam	ggd	RUS	RUS
10	baban	gdg	KOL	KOL
11	tarun	wr	USA	USA
12	ram	ad	NY	NC
13	morti	tyu	NY	NY
14	roy	wsa	IN	IN

14 rows | 0.32 seconds runtime

Refreshed now

Command took 0.32 seconds -- by user4@hk0178210@gmail.onmicrosoft.com at 10/5/2023, 1:10:22 PM on ADB Moni Cluster

Shift+Enter to run
Shift+Ctrl+Enter to run selected text

HOW TO CREATE TABLE WITH NOTEBOOK:

1. `spark.sql("CREATE TABLE cities (name STRING, population INT) USING PARQUET")`
2. `spark.sql("INSERT INTO cities VALUES ('Seattle', 730400), ('San Francisco', 881549), ('Beijing', 21540000), ('Bangalore', 10540000)")`
3. `display(spark.sql("SELECT * FROM cities ORDER BY name"))`

The screenshot shows a Databricks workspace interface. On the left is a sidebar with various navigation options like Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, and SQL Warehouses. The main area is titled "Explore hive_metastore.default.person" and shows three command cells (Cmd 2, Cmd 3, Cmd 4) in Python.

- Cmd 2:** Contains the code `spark.sql("CREATE TABLE cities (name STRING, population INT) USING PARQUET")`. The output shows a DataFrame with no rows.
- Cmd 3:** Contains the code `spark.sql("INSERT INTO cities VALUES ('Seattle', 730400), ('San Francisco', 881549), ('Beijing', 21540000), ('Bangalore', 10540000)")`. The output shows a DataFrame with four rows, each representing a city with its name and population.
- Cmd 4:** Contains the code `display(spark.sql("SELECT * FROM cities ORDER BY name"))`. The output shows a DataFrame with the same four rows, ordered by name.

Explore hive_metastore.default.person

Python

File Edit View Run Help Last edit was 5 minutes ago Provide feedback

Run all ADB Moni Cluster Schedule Share

Command took 1.22 seconds -- by user4@hk0178210@gmail.onmicrosoft.com at 10/5/2023, 1:14:58 PM on ADB Moni Cluster

Cmd 4

1 display(spark.sql("SELECT * FROM cities ORDER BY name"))

(1) Spark Jobs

Table +

New result table: OFF

	name	population
1	Bangalore	10540000
2	Beijing	21540000
3	San Francisco	881549
4	Seattle	730400

4 rows | 0.74 seconds runtime Refreshed 5 minutes ago

Command took 0.74 seconds -- by user4@hk0178210@gmail.onmicrosoft.com at 10/5/2023, 1:15:16 PM on ADB Moni Cluster

Cmd 5

Microsoft Azure | databricks Search data, notebooks, recents, and more... CTRL + P adbmonishaust user4@hk0178210@gmail.onmicrosoft.com

New Workspace Recents Catalog Workflows Compute SQL SQL Editor Queries Dashboards Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Delta Live Tables

Explore hive_metastore.default.person

Python

File Edit View Run Help Last edit was 5 minutes ago Provide feedback

Run all ADB Moni Cluster Schedule Share

Command took 1.22 seconds -- by user4@hk0178210@gmail.onmicrosoft.com at 10/5/2023, 1:14:58 PM on ADB Moni Cluster

Cmd 4

1 display(spark.sql("SELECT * FROM cities ORDER BY name"))

(1) Spark Jobs

Table +

New result table: OFF

	name	population
1	Bangalore	10540000
2	Beijing	21540000
3	San Francisco	881549
4	Seattle	730400

4 rows | 0.74 seconds runtime Refreshed 5 minutes ago

Command took 0.74 seconds -- by user4@hk0178210@gmail.onmicrosoft.com at 10/5/2023, 1:15:16 PM on ADB Moni Cluster

Cmd 5

Microsoft Azure | databricks Search data, notebooks, recents, and more... CTRL + P adbmonishaust user4@hk0178210@gmail.onmicrosoft.com

New Workspace Recents Catalog Workflows Compute SQL SQL Editor Queries Dashboards Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Delta Live Tables

MOUNT STORAGE TO DATA BRICKS:

1. Create data lake storage
2. In cluster notebook → New → Notebook → Name : ADB to ADLS Connectivity
3. In cell, To add markdown:
%md
Mount DataLake with DataBricks → bold
4. To configure data lake or connect:

```
spark.conf.set(  
    'fs.azure.account.key.adlsmonisha.dfs.core.windows.net'  
    ,  
    'bs+6/GW6Ps+g8kKPGEeITgCZ00eHu1Fe2WAyeizdR9s0090o/TlQCGs90tWD80foRMi1YedWKKQl+'  
    'AstavXOMg=='  
)  
  
Syntax : spark.conf.set('fs.azure.account.key(or)SAS.endpoint  
of storage','account key or SAS')  
Fs → file system  
Authentication method → account key or SAS  
5. To copy endpoint : Go to storage account → end points →  
DataLake Storage url without https:// → copy  
6. To copy account key: Go to storage account → Access key → key  
→ click show → copy key  
7. Shift enter and run the cell  
8. To list the files in the storage account :  
dbutils.fs.ls('abfss://raw@adlsmonisha.dfs.core.windows.net')  
syntax:  
dbutils.fs.ls('abfss://containername@storagename.endpoints/fold  
er/subfolder/file.exe')  
abfss → azure blob file system
```

The screenshot shows the Microsoft Azure Databricks interface. On the left, the sidebar includes options like New, Workspace, Recents, Catalog, Workflows, Compute, SQL Editor, Queries, Dashboards, Alerts, Query History, and SQL Warehouses. The main area is titled 'ADB TO ADLS Connectivity' and is set to Python. It displays the following code in Cmd 1:

```
spark.conf.set(
  'fs.azure.account.key.adlsmmonisha.dfs.core.windows.net',
  'bs+6/GW6Ps+g8kKPGeeITgCZ00eHu1Fe2WAye1zdR9s0090o/TlQCGs90tWD80foRMi1YedWKkQl+AStavXOMg=='
)
```

A message at the top of the notebook states: "Free trial ends in 14 days. Upgrade to Premium in Azure Portal". Below the code, it says "Command took 0.47 seconds -- by user4@hk0178210@gmail.onmicrosoft.com at 10/5/2023, 3:40:24 PM on ADB Moni Cluster". In Cmd 2, the following command is run:

```
dbutils.fs.ls('abfss://raw@adlsmmonisha.dfs.core.windows.net')
```

The output shows a single file: "[FileInfo(path='abfss://raw@adlsmmonisha.dfs.core.windows.net/Person.csv', name='Person.csv', size=316, modificationTime=1696500951000)]". Below this, it says "Command took 1.89 seconds -- by user4@hk0178210@gmail.onmicrosoft.com at 10/5/2023, 3:46:58 PM on ADB Moni Cluster".

HOW TO READ THE FILE:

df → data frame
read → to read data
inferSchema → by default csv data will be string but if u correct data type.
inferSchema will auto deduct data type
header → header of table is true
delimiter → it can be , : . this file is comma separated.
Read File from ADLS
source_file = 'abfss://raw@adlsmmonisha.dfs.core.windows.net/Person.csv'

```
df = spark.read\  
.format('csv')\  
.option('inferSchema', True)\  
.option('header', True)\  
.option('delimiter', ',')\  
.load(source_file)
```

To view data:

`df.show()`

A screenshot of a Databricks notebook titled "ADB TO ADLS Connectivity". The notebook interface includes a sidebar with various workspace options like Recents, Catalog, Workflows, and Compute. The main area shows the command `df.show()` and its output, which is a table with 14 rows and 6 columns. The columns are PersonID, LastName, FirstName, Address, and City. The data is as follows:

PersonID	LastName	FirstName	Address	City
1	Zoe	kl	USA	USA
2	Kal	kl	NY	NC
3	krm	fgfd	NY	NY
4	jay	bc	IN	IN
5	ajay	g	CAL	CAL
6	aman	ffb	PUN	PUN
7	kalpana	fcv	AUS	AUS
8	sayali	gddg	ZIM	ZIM
9	shyam	ggd	RUS	RUS
10	baban	gdg	KOL	KOL
11	tarun	wr	USA	USA
12	ram	ad	NY	NC
13	morti	tyu	NY	NY
14	roy	wsa	IN	IN

Below the table, a message indicates the command took 1.09 seconds to run.

`display(df)`

A screenshot of a Databricks notebook titled "ADB TO ADLS Connectivity". The sidebar and notebook structure are identical to the previous screenshot. The main area shows the command `display(df)` and its output, which is a table with 14 rows and 6 columns. The columns are PersonID, LastName, FirstName, Address, and City. The data is the same as the previous screenshot. The table has a header row and 14 data rows. A message at the bottom indicates the command took 0.97 seconds to run.

PersonID	LastName	FirstName	Address	City
1	Zoe	kl	USA	USA
2	Kal	kl	NY	NC
3	krm	fgfd	NY	NY
4	jay	bc	IN	IN
5	ajay	g	CAL	CAL
6	aman	ffb	PUN	PUN
7	kalpana	fcv	AUS	AUS
8	sayali	gddg	ZIM	ZIM
9	shyam	ggd	RUS	RUS
10	baban	gdg	KOL	KOL
11	tarun	wr	USA	USA
12	ram	ad	NY	NC
13	morti	tyu	NY	NY
14	roy	wsa	IN	IN

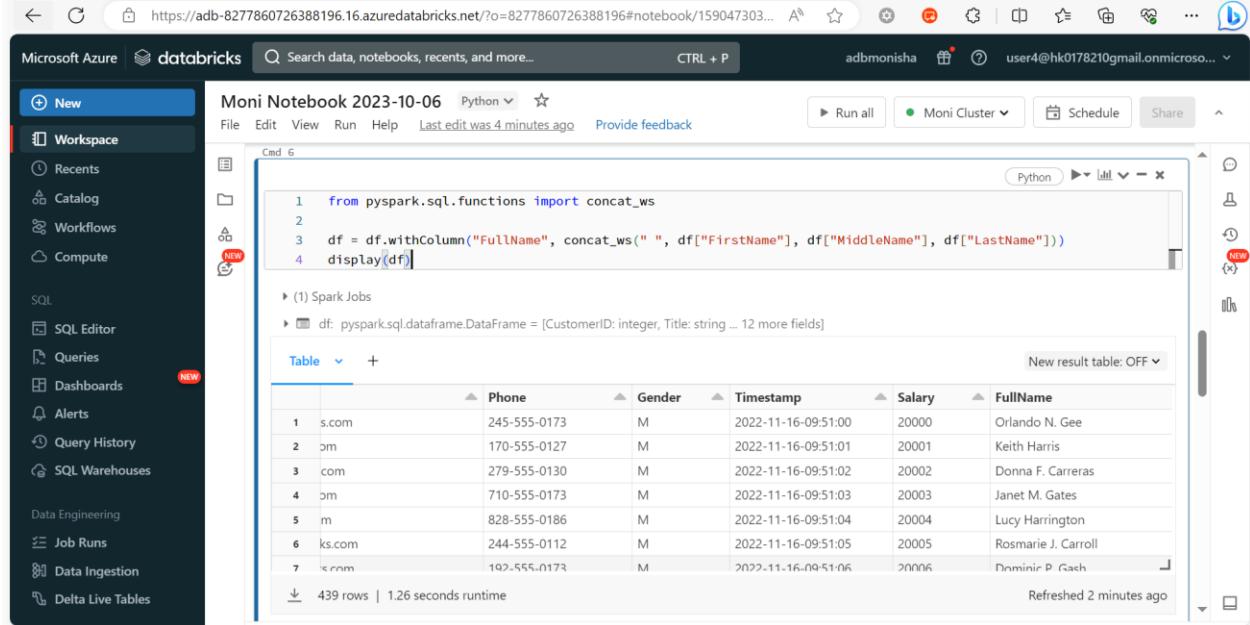
Below the table, a message indicates the command took 0.97 seconds to run.

ASSIGNMENT:

1)

```
from pyspark.sql.functions import concat_ws

df = df.withColumn("FullName", concat_ws(" ", df["FirstName"], df["MiddleName"], df["LastName"]))
display(df)
```



The screenshot shows a Databricks workspace interface. On the left, there's a sidebar with various navigation options like Workspace, Recents, Catalog, Workflows, Compute, SQL, and more. The main area is titled 'Moni Notebook 2023-10-06'. It contains a code cell labeled 'Cmd 6' with the following Python code:

```
1 from pyspark.sql.functions import concat_ws
2
3 df = df.withColumn("FullName", concat_ws(" ", df["FirstName"], df["MiddleName"], df["LastName"]))
4 display(df)
```

Below the code cell, it says '(1) Spark Jobs' and shows a preview of the DataFrame 'df' with the following data:

	Phone	Gender	Timestamp	Salary	FullName
1	s.com	M	2022-11-16-09:51:00	20000	Orlando N. Gee
2	om	M	2022-11-16-09:51:01	20001	Keith Harris
3	com	M	2022-11-16-09:51:02	20002	Donna F. Carreras
4	om	M	2022-11-16-09:51:03	20003	Janet M. Gates
5	m	M	2022-11-16-09:51:04	20004	Lucy Harrington
6	ks.com	M	2022-11-16-09:51:05	20005	Rosmarie J. Carroll
7	com	M	2022-11-16-09:51:06	20006	Dominic P. Gach

At the bottom of the preview, it says '439 rows | 1.26 seconds runtime' and 'Refreshed 2 minutes ago'.

2)

```
from pyspark.sql.functions import when
```

```
df = df.withColumn("Gender", when(df["Gender"] == "F",  
"Female").otherwise("Male"))
display(df)
```

Moni Notebook 2023-10-06 Python

```

1 from pyspark.sql.functions import when
2
3 df = df.withColumn("Gender", when(df["Gender"] == "F", "Female").otherwise("Male"))
4 display(df)

```

(1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [CustomerID: integer, Title: string ... 12 more fields]

	EmailAddress	Phone	Gender	Timestamp	Salary
17	kermit@adventure-works.com	216-555-0122	Male	2022-11-16-09:51:16	20016
18	kevin5@adventure-works.com	926-555-0164	Male	2022-11-16-09:51:17	20017
19	donald0@adventure-works.com	357-555-0161	Male	2022-11-16-09:51:18	20018
20	jackie0@adventure-works.com	972-555-0163	Female	2022-11-16-09:51:19	20019
21	bryan2@adventure-works.com	344-555-0144	Female	2022-11-16-09:51:20	20020
22	todd0@adventure-works.com	783-555-0110	Female	2022-11-16-09:51:21	20021
23	barbara4@adventure-works.com	1 (11) 500 555-0181	Female	2022-11-16-09:51:22	20022

439 rows | 1.16 seconds runtime

Refreshed 3 minutes ago

3)

```

pdf = df.select('CustomerID', 'FullName', 'CompanyName', 'SalesPerson',
'EmailAddress', 'Phone', 'Gender')
pdf.createOrReplaceTempView("Customerdata")
display(spark.sql("SELECT * FROM Customerdata"))

```

Moni Notebook 2023-10-06 Python

```

1 pdf = df.select('CustomerID', 'FullName', 'CompanyName', 'SalesPerson', 'EmailAddress', 'Phone', 'Gender')
2 pdf.createOrReplaceTempView("Customerdata")
3 display(spark.sql("SELECT * FROM Customerdata"))

```

(1) Spark Jobs

pdf: pyspark.sql.dataframe.DataFrame = [CustomerID: integer, FullName: string ... 5 more fields]

	CustomerID	FullName	CompanyName	SalesPerson	EmailAddress
1	1	Orlando N. Gee	A Bike Store	adventure-works\\pamela0	orla
2	2	Keith Harris	Progressive Sports	adventure-works\\david8	keit
3	3	Donna F. Carreras	Advanced Bike Components	adventure-works\\jillian0	doni
4	4	Janet M. Gates	Modular Cycle Systems	adventure-works\\jillian0	jane
5	5	Lucy Harrington	Metropolitan Sports Supply	adventure-works\\shu0	lucy
6	6	Rosmarie J. Carroll	Aerobic Exercise Company	adventure-works\\linda3	rosn

439 rows | 1.40 seconds runtime

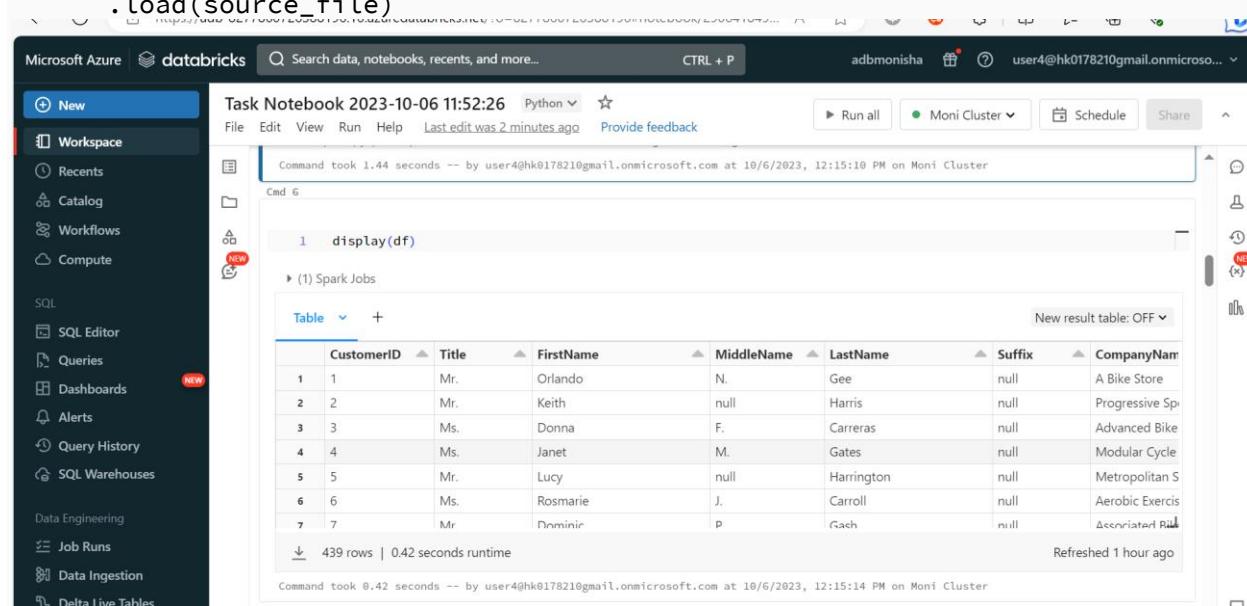
Refreshed 3 minutes ago

Command took 1.40 seconds -- by user4@hk0178210gmail.onmicrosoft.com at 10/6/2023, 10:39:45 AM on Moni Cluster

Task:

1)

```
# TODO  
# Replace <<FILL_IN>> with your code.  
source_file =  
'abfss://raw@adlsmonisha.dfs.core.windows.net/Source/customerdemo.parquet'  
  
pdf = spark.read\  
    .parquet(source_file)  
    .load(source_file)
```



The screenshot shows a Microsoft Azure Databricks Task Notebook interface. The left sidebar contains navigation links for New, Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, and Delta Live Tables. The main area is titled "Task Notebook 2023-10-06 11:52:26 Python". It displays a command history with one entry: "display(df)". Below the command is a table view showing data from a DataFrame. The table has columns: CustomerID, Title, FirstName, MiddleName, LastName, Suffix, and CompanyName. The data consists of 439 rows. The table is styled with alternating row colors and column headers. A message at the bottom right indicates the data was refreshed 1 hour ago.

	CustomerID	Title	FirstName	MiddleName	LastName	Suffix	CompanyName
1	1	Mr.	Orlando	N.	Gee	null	A Bike Store
2	2	Mr.	Keith	null	Harris	null	Progressive Sp
3	3	Ms.	Donna	F.	Carreras	null	Advanced Bike
4	4	Ms.	Janet	M.	Gates	null	Modular Cycle
5	5	Mr.	Lucy	null	Harrington	null	Metropolitan S
6	6	Ms.	Rosmarie	J.	Carroll	null	Aerobic Exercis
7	7	Mr	Dominic	P	Gach	null	Decorated Rad

2)

```
source_file =  
'abfss://raw@adlsmonisha.dfs.core.windows.net/Source/customerdemo.parquet'  
  
df = spark.read\  
    .parquet(source_file)\\  
    .select("FirstName")\\  
    .distinct()
```

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P adbmonisha user4@hk0178210gmail.onmicrosoft.com

Task Notebook 2023-10-06 11:52:26 Python

```

5 ... .select("FirstName") \
6 ... .distinct()
7 # Identify the total number of records remaining.

(4) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [FirstName: string]
Command took 2.01 seconds -- by user4@hk0178210gmail.onmicrosoft.com at 10/6/2023, 12:20:06 PM on Moni Cluster

Cmd 8

1 display(df)

(2) Spark Jobs
Table + New result table: OFF
+-----+
| FirstName |
+-----+
| 1 Chad   |
| 2 Shannon |
| 3 Carolyn |
| 4 Scott   |
| 5 Keith   |
+-----+

```

3)

```
totalArticles = df.count()
print("Distinct Articles: {}".format(totalArticles))
```

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P adbmonisha user4@hk0178210gmail.onmicrosoft.com

Task Notebook 2023-10-06 11:52:26 Python

```

1 totalArticles = df.count()
2 print("Distinct Articles: {}".format(totalArticles))

(3) Spark Jobs
Distinct Articles: 313
Command took 0.96 seconds -- by user4@hk0178210gmail.onmicrosoft.com at 10/6/2023, 12:23:03 PM on Moni Cluster

Cmd 10

1 from pyspark.sql.functions import concat_ws
2
3 pdf = pdf.withColumn("FullName", concat_ws(" ", pdf["FirstName"], pdf["MiddleName"], pdf["LastName"]))
4 display(pdf)

(1) Spark Jobs
pdf: pyspark.sql.dataframe.DataFrame = [CustomerID: integer, Title: string ... 12 more fields]
Table + New result table: OFF
+-----+
| CustomerID | Title | FirstName | MiddleName | LastName | Suffix | CompanyName |
+-----+
| 1           | Mr.  | Orlando   | N.          | Gee      | null   | A Bike Store |
| 2           | Mr.  | Keith      | null        | Harris   | null   | Progressive Sp |
+-----+

```

4)

```
from pyspark.sql.functions import concat_ws

pdf = pdf.withColumn("FullName", concat_ws(" ", pdf["FirstName"],
pdf["MiddleName"], pdf["LastName"]))
display(pdf)
```

The screenshot shows a Microsoft Azure Databricks Task Notebook titled "Task Notebook 2023-10-06 11:52:26" in Python. The notebook interface includes a sidebar with options like New, Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, and Delta Live Tables. The main area displays a code cell with the following Python code:

```
1 from pyspark.sql.functions import concat_ws
2
3 pdf = pdf.withColumn("FullName", concat_ws(" ", pdf["FirstName"], pdf["MiddleName"], pdf["LastName"]))
4 display(pdf)
```

Below the code, it says "(1) Spark Jobs" and "pdf: pyspark.sql.dataframe.DataFrame = [CustomerID: integer, Title: string ... 12 more fields]". A table view shows the data with columns: CustomerID, Title, FirstName, MiddleName, LastName, Suffix, CompanyName. The data is as follows:

	CustomerID	Title	FirstName	MiddleName	LastName	Suffix	CompanyName
1	1	Mr.	Orlando	N.	Gee	null	A Bike Store
2	2	Mr.	Keith	null	Harris	null	Progressive Spi
3	3	Ms.	Donna	F.	Carreras	null	Advanced Bike
4	4	Ms.	Janet	M.	Gates	null	Modular Cycle
5	5	Mr.	Lucy	null	Harrington	null	Metropolitan S
6	6	Ms.	Rosmarie	J.	Carroll	null	Aerobic Exercis
7	7	Mr	Dominic	P	Gash	null	Decorated Rul

At the bottom, it says "439 rows | 0.96 seconds runtime" and "Refreshed 1 hour ago". The status bar at the bottom indicates "Command took 0.96 seconds -- by user4@hk0178210gmail.onmicrosoft.com at 10/6/2023, 12:25:00 PM on Moni Cluster".

5)

To rename column and change data type:

```
from pyspark.sql.functions import *
from pyspark.sql.types import StringType
tdf = pdf.withColumnRenamed('TimeStamp',
'CreatedTime').withColumn('TimeStamp', unix_timestamp(col('TimeStamp'), "yyyy-MM-dd 't' HH:mm:ss").cast(StringType()))
```

The screenshot shows a Microsoft Azure Databricks Task Notebook titled "Task Notebook 2023-10-06 11:52:26" in Python. The sidebar is identical to the previous screenshot. The main area shows the command "tdf.printSchema()" executed in a cell, resulting in the following schema output:

```
|-- TimeStamp: string (nullable = true)
|-- Salary: integer (nullable = true)
|-- FullName: string (nullable = false)

Command took 0.03 seconds -- by user4@hk0178210gmail.onmicrosoft.com at 10/6/2023, 1:33:42 PM on Moni Cluster
```

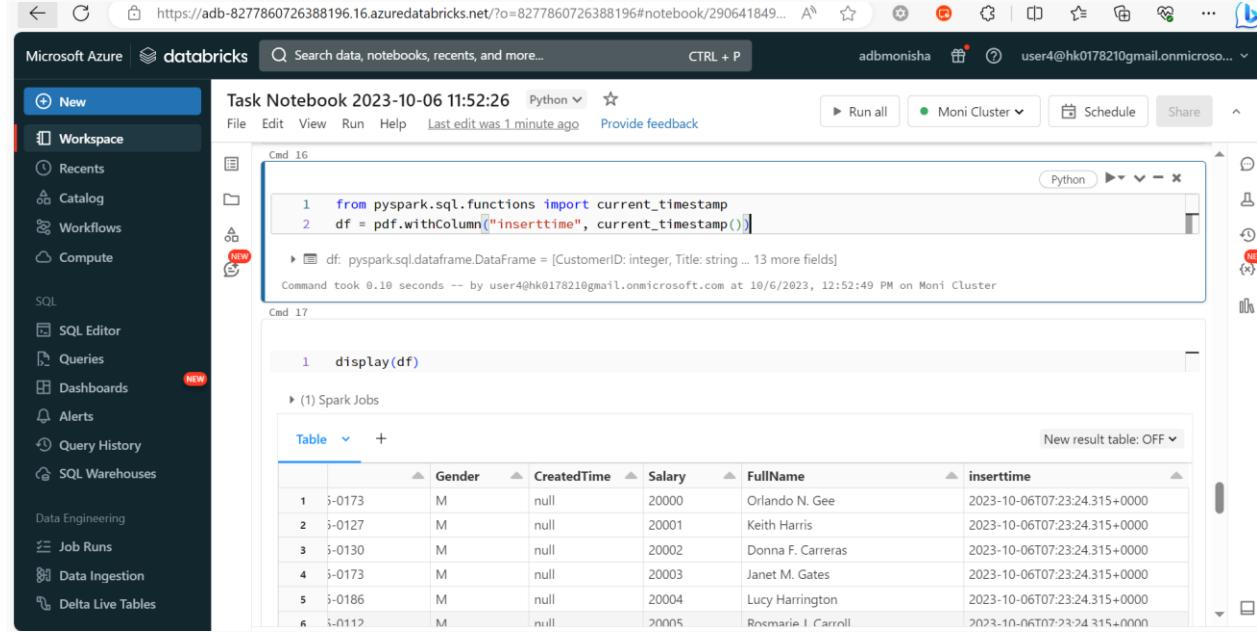
Below this, another cell labeled "Cmd 15" shows the same schema output again:

```
1 tdf.printSchema()
```

```
root
 |-- CustomerID: integer (nullable = true)
 |-- Title: string (nullable = true)
 |-- FirstName: string (nullable = true)
 |-- MiddleName: string (nullable = true)
 |-- LastName: string (nullable = true)
 |-- Suffix: string (nullable = true)
 |-- CompanyName: string (nullable = true)
 |-- SalesPerson: string (nullable = true)
 |-- EmailAddress: string (nullable = true)
 |-- Phone: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- TimeStamp: string (nullable = true)
 |-- Salary: integer (nullable = true)
 |-- FullName: string (nullable = false)
```

6)

```
from pyspark.sql.functions import current_timestamp
df = pdf.withColumn("inserttime", current_timestamp())
```



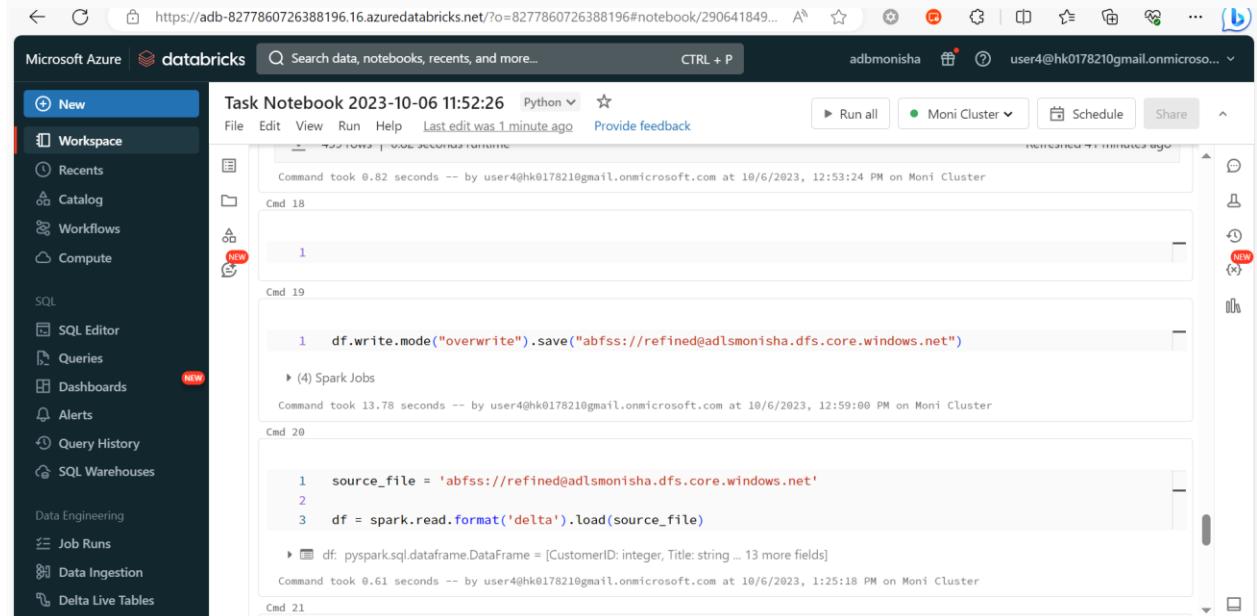
The screenshot shows a Databricks Task Notebook interface. The sidebar on the left includes options like New, Workspace, Recents, Catalog, Workflows, Compute, SQL, and Data Engineering. The main area displays a command cell (Cmd 16) containing Python code to add an 'inserttime' column to a DataFrame. Below the code, the output shows the DataFrame structure and the resulting data with the new timestamp column.

ID	CustomerID	Gender	CreatedTime	Salary	FullName	inserttime
1	3-0173	M	null	20000	Orlando N. Gee	2023-10-06T07:23:24.315+0000
2	3-0127	M	null	20001	Keith Harris	2023-10-06T07:23:24.315+0000
3	3-0130	M	null	20002	Donna F. Carreras	2023-10-06T07:23:24.315+0000
4	3-0173	M	null	20003	Janet M. Gates	2023-10-06T07:23:24.315+0000
5	3-0186	M	null	20004	Lucy Harrington	2023-10-06T07:23:24.315+0000
6	3-0112	M	null	20005	Rosmarie I. Carroll	2023-10-06T07:23:24.315+0000

7)

To write the file and store it in datalake another container:

```
df.write.mode("overwrite").save("abfss://refined@adlsmonisha.dfs.core.windows.net")
```



The screenshot shows a Databricks Task Notebook interface. The sidebar on the left includes options like New, Workspace, Recents, Catalog, Workflows, Compute, SQL, and Data Engineering. The main area displays a command cell (Cmd 18) containing Python code to save a DataFrame to Azure Data Lake Storage. Below the code, the output shows the command took 0.82 seconds and the resulting data.

ID	CustomerID	Gender	CreatedTime	Salary	FullName	inserttime
1	3-0173	M	null	20000	Orlando N. Gee	2023-10-06T07:23:24.315+0000
2	3-0127	M	null	20001	Keith Harris	2023-10-06T07:23:24.315+0000
3	3-0130	M	null	20002	Donna F. Carreras	2023-10-06T07:23:24.315+0000
4	3-0173	M	null	20003	Janet M. Gates	2023-10-06T07:23:24.315+0000
5	3-0186	M	null	20004	Lucy Harrington	2023-10-06T07:23:24.315+0000
6	3-0112	M	null	20005	Rosmarie I. Carroll	2023-10-06T07:23:24.315+0000

8)

To read the delta file:

```
source_file = 'abfss://refined@adlsmonisha.dfs.core.windows.net'
```

```
df = spark.read.format('delta').load(source_file)
```

The screenshot shows a Microsoft Azure Databricks Task Notebook interface. The left sidebar contains navigation links for Workspace, Recents, Catalog, Workflows, Compute, SQL, and Data Engineering. The main area has a header bar with 'Task Notebook 2023-10-06 11:52:26' and 'Python'. Below the header is a code editor with three lines of Python code:

```
1 source_file = 'abfss://refined@adlsmonisha.dfs.core.windows.net'
2
3 df = spark.read.format('delta').load(source_file)
```

Below the code editor is a command history section labeled 'Cmd 22' with one entry:

```
1 display(df)
```

At the bottom, there is a table preview titled '(1) Spark Jobs' with the following data:

	CustomerID	Title	FirstName	MiddleName	LastName	Suffix	CompanyName
1	1	Mr.	Orlando	N.	Gee	null	A Bike Store
2	2	Mr.	Keith	null	Harris	null	Progressive Sp
3	3	Ms.	Donna	F.	Carreras	null	Advanced Bike
4	4	Ms.	Janet	M.	Gates	null	Modular Cycle
5	5	Mr.	Lucy	null	Harrington	null	Metropolitan S
6	6	Ms.	Rosmarie	J.	Carroll	null	Aerobic Exercis

7/10/2023

- 1) Create data lake storage(adlsmonisha)
- 2) Create sql database use server existing(SQLMonisha)
- 3) Data factory(adfmonisha)
- 4) Create a container named IncrementalDataLoad in DataLake Storage

Microsoft Azure | Containers

New container

Name * ✓

Anonymous access level

The access level is set to private because anonymous access is disabled on this storage account.

Advanced

Create Give feedback ↗

SQL Server Management Studio

Object Explorer

SQL Server

Server type: Database Engine
 Server name: usstrainserver database windows.net
 Authentication: SQL Server Authentication
 Login: Adminlogin
 Password:
 Remember password

Connect

mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name	
1	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
2	15	8	350	165	3693	11.5	70	1	buck skylark 320
3	18	8	318	150	3436	11	70	1	plymouth satellite
4	16	8	304	150	3433	12	70	1	amc rebel sst
5	17	8	302	140	3449	10.5	70	1	ford torino
6	15	8	429	198	4341	10	70	1	ford galaxie 500
7	14	8	454	220	4354	9	70	1	chevrolet impala
8	14	8	440	215	4312	8.5	70	1	plymouth fury ii
9	14	8	455	225	4425	10	70	1	pontiac catalina
10	15	8	390	190	3850	8.5	70	1	amc ambassador dpl
11	15	8	383	170	3563	10	70	1	dodge challenger se
12	14	8	340	160	3600	8	70	1	nkmouth vega 2d

Results Messages

Disconnected.

Ready

```

create table data_source_table
(
    PersonID int,
    Name varchar(255),
    LastModifytime datetime
);

INSERT INTO data_source_table
    (PersonID, Name, LastModifytime)
VALUES

```

```

(1, 'aaaa','9/1/2017 12:56:00 AM'),
(2, 'bbbb','9/2/2017 5:23:00 AM'),
(3, 'cccc','9/3/2017 2:36:00 AM'),
(4, 'dddd','9/4/2017 3:21:00 AM'),
(5, 'eeee','9/5/2017 8:06:00 AM');

```

```
select * from data_source_table
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like tables and system tables. The central pane contains a T-SQL script with an INSERT statement followed by a SELECT statement. The Results pane below shows the output of the SELECT statement, which retrieves five rows of data with columns PersonID, Name, and LastModifytime. The status bar at the bottom indicates the query was executed successfully.

PersonID	Name	LastModifytime
1	aaaa	2017-09-01 00:56:00.000
2	bbbb	2017-09-02 05:23:00.000
3	cccc	2017-09-03 02:36:00.000
4	dddd	2017-09-04 03:21:00.000
5	eeee	2017-09-05 08:06:00.000

```

5)
6) create table watermarktable
7) (
8)
9) TableName varchar(255),
10)WatermarkValue datetime,
11));
12)
13)-----
14)
15)INSERT INTO watermarktable
16)VALUES ('data_source_table','1/1/2010 12:00:00 AM')
17)
18)-----
19)
20)Select * from watermarktable
21)

```

SQLQuery2.sql - usstrainer.database.windows.net.SQLmonisha (Adminlogin (73)) - Microsoft SQL Server Management Studio

```

create table watermarktable
(
    TableName varchar(255),
    WatermarkValue datetime,
);

INSERT INTO watermarktable
VALUES ('data_source_table', '1/1/2010 12:00:00 AM')

```

Results

TableName	WatermarkValue
data_source_table	2010-01-01 00:00:00.000

Query executed successfully.

22)

23)CREATE PROCEDURE usp_write_watermark @LastModifiedtime datetime,

 @TableName varchar(50)

24)AS

25)

26)BEGIN

27)

28)UPDATE watermarktable

29)SET [WatermarkValue] = @LastModifiedtime

30)WHERE [TableName] = @TableName

31)

32)END

SQLQuery2.sql - usstrainer.database.windows.net.SQLmonisha (Adminlogin (73)) - Microsoft SQL Server Management Studio

```

Select * from watermarktable

CREATE PROCEDURE usp_write_watermark @LastModifiedtime datetime, @TableName varchar(50)
AS
BEGIN
    UPDATE watermarktable
    SET [WatermarkValue] = @LastModifiedtime
    WHERE [TableName] = @TableName
END

```

Commands completed successfully.

Completion time: 2023-10-07T13:21:56.8419772+05:30

Query executed successfully.

33)

34)

New linked service

Data store Compute

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

Continue Cancel

35)

New linked service

Azure SQL Database Learn more

Connect via integration runtime *

Account selection method From Azure subscription Enter manually

Azure subscription Pay-As-You-Go (45f63aab-e46c-4be8-bb77-43fe31240084)

Server name * usstrainserver

Database name * SQLmonisha

Authentication type * SQL authentication

Create Back Test connection Cancel

Microsoft Azure | Data Factory > adfmonisha

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us en...

Validate all Publish all

Linked services

Linked service defines the connection information to a data store.

+ New

Filter by name Annotations : Any

Showing 1 - 1 of 1 items

Name	Type
LS_AzureSqlDatabase	Azure SQL Database

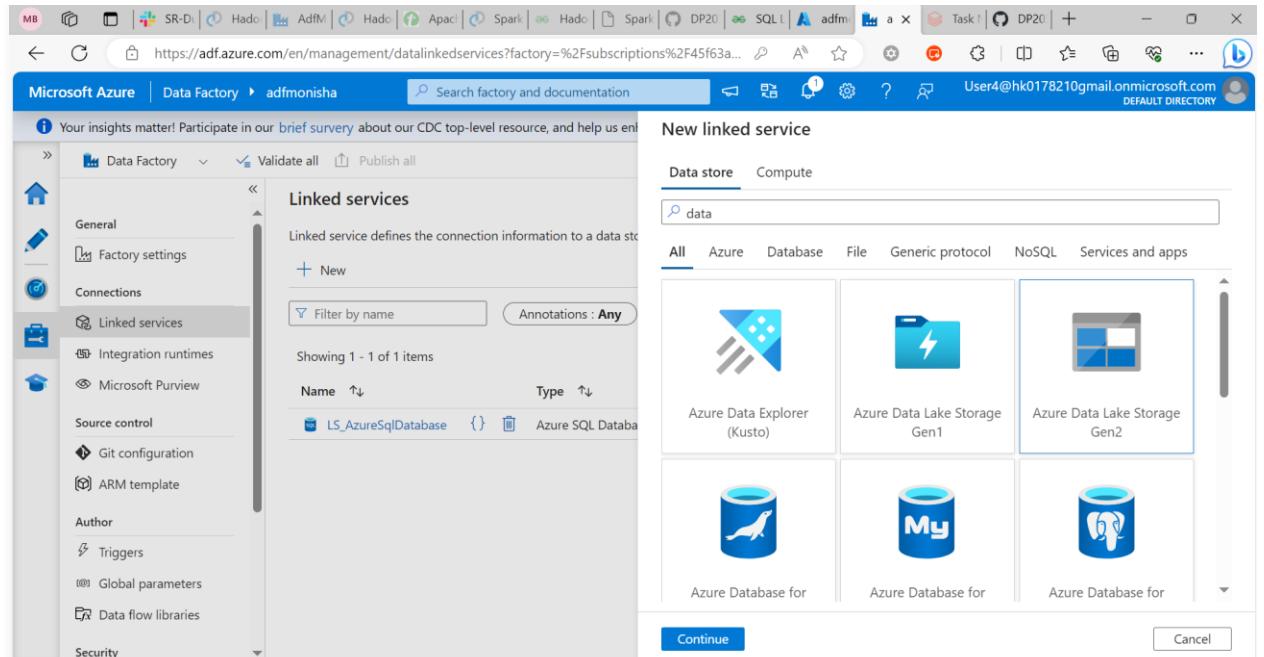
Data store Compute

data

All Azure Database File Generic protocol NoSQL Services and apps

 Azure Data Explorer (Kusto)	 Azure Data Lake Storage Gen1	 Azure Data Lake Storage Gen2
 Azure Database for MySQL	 Azure Database for PostgreSQL	 Azure Database for Oracle

Continue Cancel



Microsoft Azure | Data Factory > adfmonisha

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us en...

Validate all Publish all

Linked services

Linked service defines the connection information to a data store.

+ New

Filter by name Annotations : Any

Showing 1 - 1 of 1 items

Name	Type
LS_AzureSqlDatabase	Azure SQL Database

New linked service

LS_AzureDataLakeStorage

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Authentication type

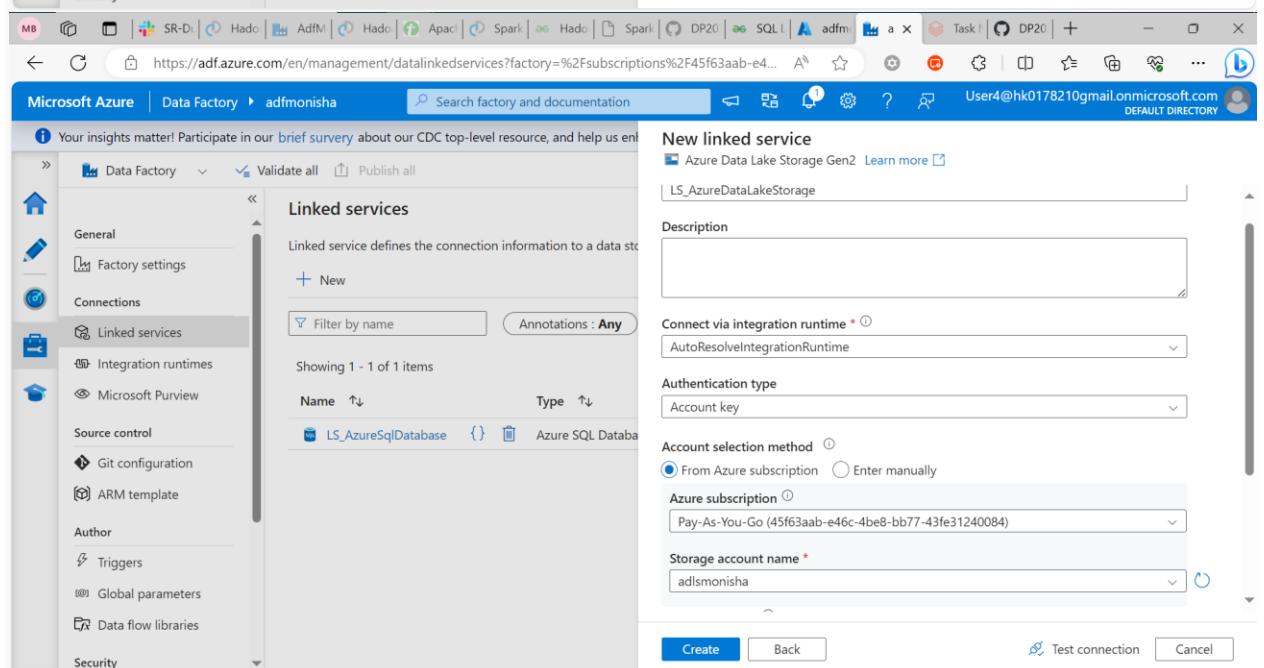
Account key

Account selection method From Azure subscription Enter manually

Azure subscription

Storage account name *

Create Back Test connection Cancel



The screenshot shows the Microsoft Azure Data Factory interface. On the left, there's a sidebar with icons for Home, Pipelines, Change Data Capture (preview), Datasets (selected), Data flows, and Power Query. The main area has tabs for Data Factory, Validate all, and Publish all. A search bar at the top right says "Search factory and documentation". The user is in the "Datasets" section, and a modal dialog titled "New dataset" is open. The dialog has a sub-header "In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. Learn more" and a "Select a data store" dropdown with "sql" selected. Below it is a grid of data store options under the "All" tab, with "Azure Database for PostgreSQL" and "Azure SQL Database" highlighted. Other options include Amazon RDS for SQL Server, Azure Cosmos DB for NoSQL, Azure Database for MySQL, and Azure SQL Database Managed Instance. At the bottom of the dialog are "Continue" and "Cancel" buttons.

This screenshot shows the "Set properties" dialog for a dataset named "DS_AzureSqlTable_Source". The dialog includes fields for "Name" (set to "DS_AzureSqlTable_Source"), "Linked service" (set to "LS_AzureSqlDatabase"), "Table name" (set to "dbo.data_source_table"), and "Import schema" (set to "From connection/store"). There are "OK", "Back", and "Cancel" buttons at the bottom. The background shows the same Data Factory interface as the previous screenshot, with the "Datasets" section selected in the sidebar.

Microsoft Azure | Data Factory > adfmonisha

Set properties

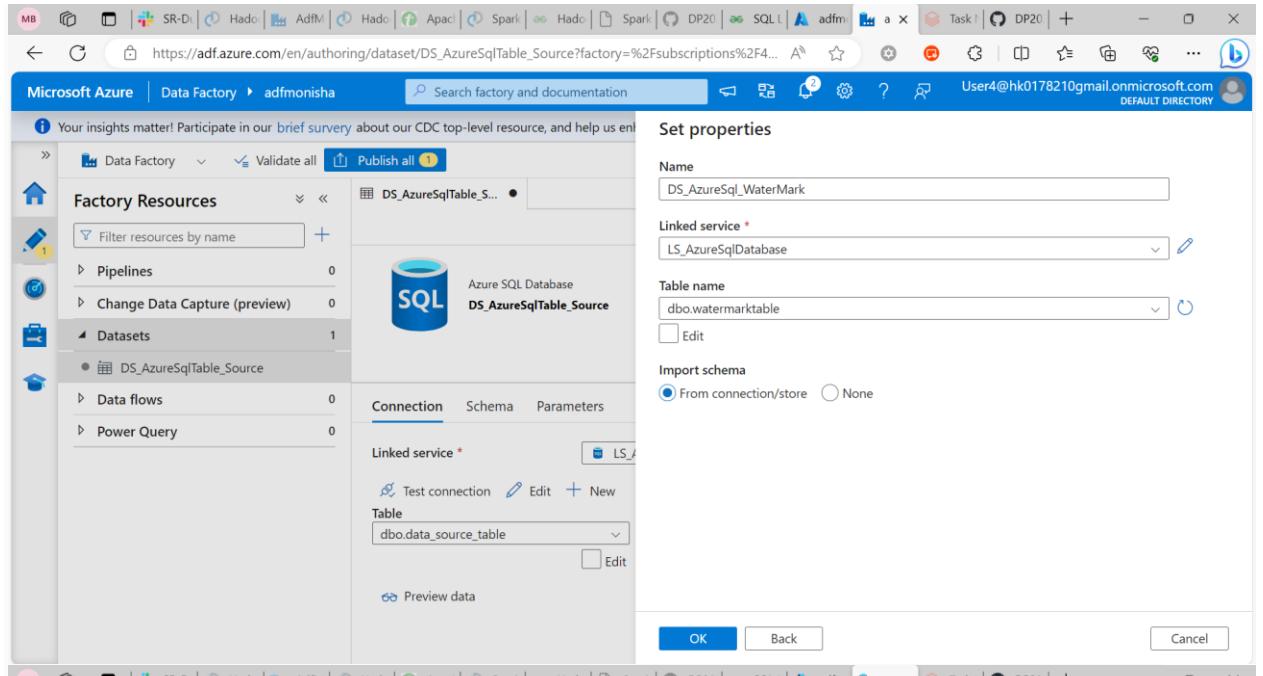
Name: DS_AzureSql_WaterMark

Linked service: LS_AzureSqlDatabase

Table name: dbo.watermarktable

Import schema: From connection/store (radio button selected)

OK Back Cancel



Microsoft Azure | Data Factory > adfmonisha

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](#)

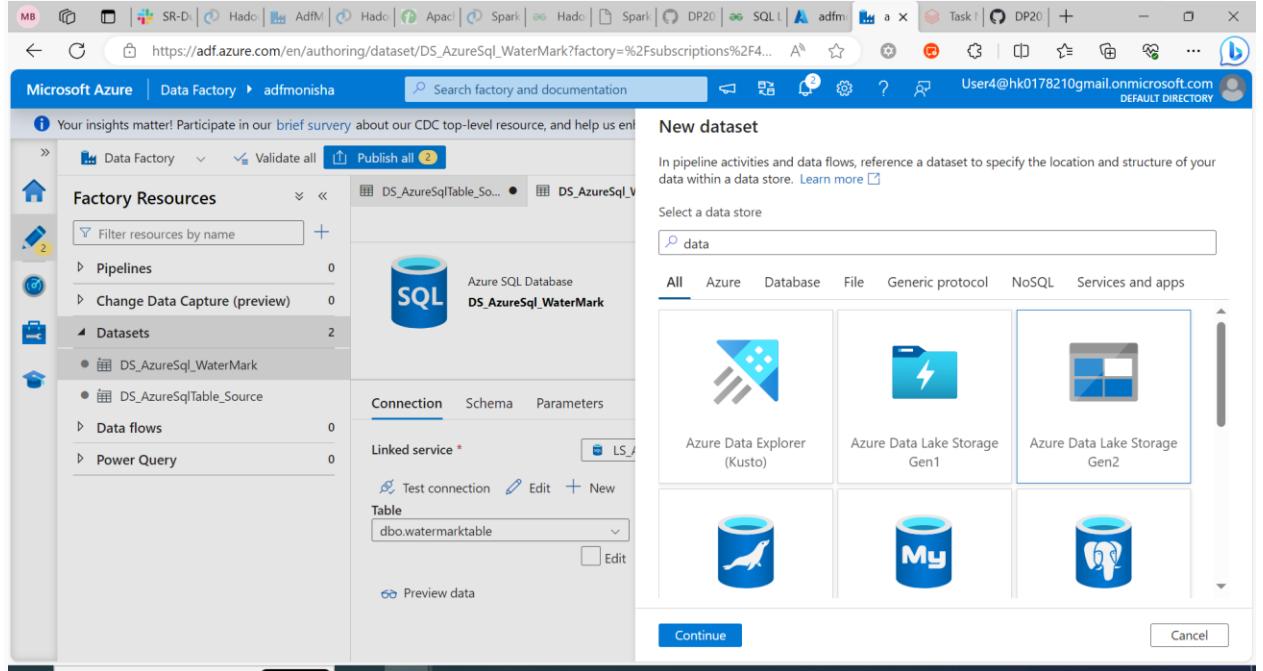
Select a data store

data

All Azure Database File Generic protocol NoSQL Services and apps

Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen1	Azure Data Lake Storage Gen2

Continue Cancel



Microsoft Azure | Data Factory > adfmonisha

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us en...

Data Factory > Validate all > Publish all (2)

Factory Resources > Datasets > DS_AzureSql_WaterMark

DS_AzureSql_WaterMark

Azure SQL Database

DS_AzureSql_WaterMark

Connection Schema Parameters

Linked service * LS_AzureDataLakeStorage

Test connection Edit + New

Table dbo.watermarktable

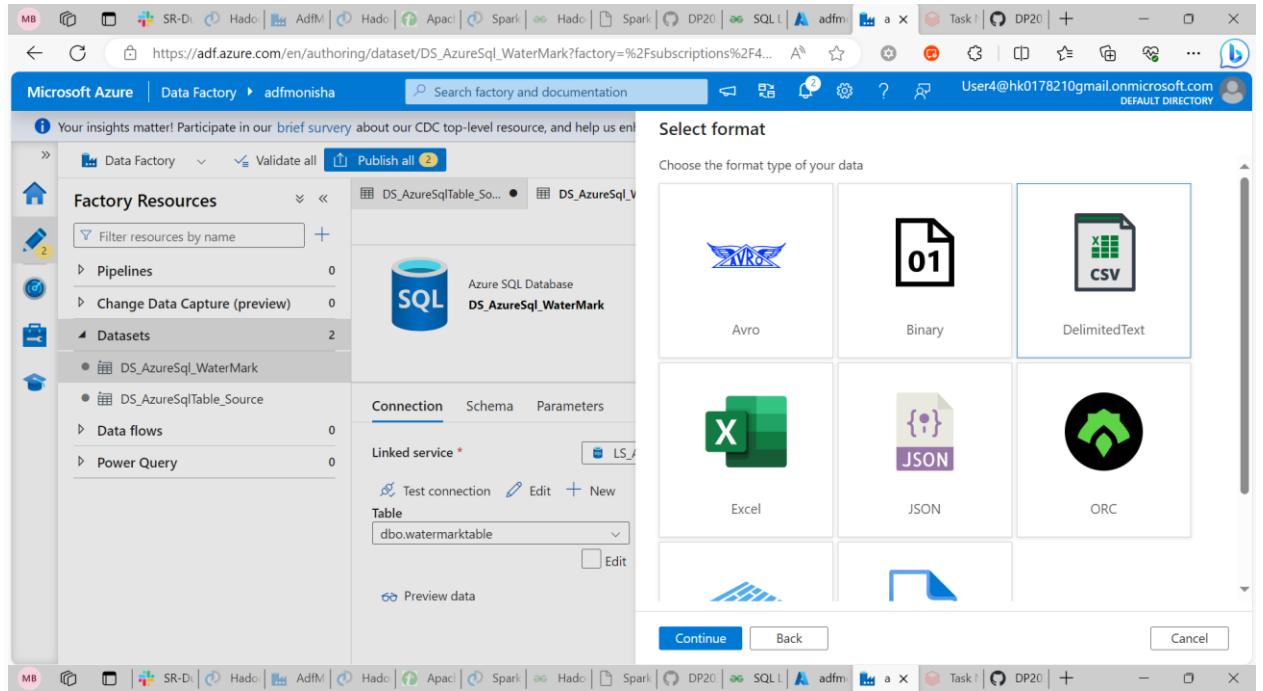
Preview data

Select format

Choose the format type of your data

Avro	Binary	DelimitedText
Excel	JSON	ORC

Continue Back Cancel



Microsoft Azure | Data Factory > adfmonisha

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us en...

Data Factory > Validate all > Publish all (2)

Factory Resources > Datasets > DS_AzureSql_WaterMark

DS_AzureSql_WaterMark

Azure SQL Database

DS_AzureSql_WaterMark

Connection Schema Parameters

Linked service * LS_AzureDataLakeStorage

File path incrementaldataload / Directory / File name

First row as header

Import schema

From connection/store From sample file None

Set properties

Name DS_Datalake

Linked service * LS_AzureDataLakeStorage

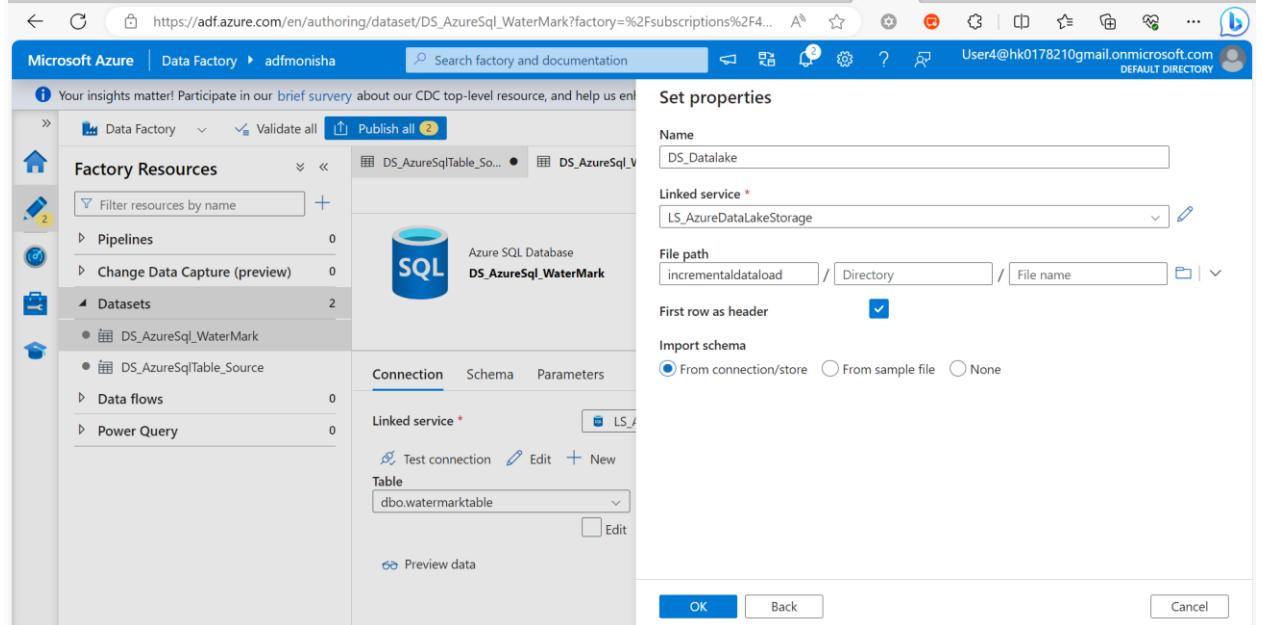
File path incrementaldataload / Directory / File name

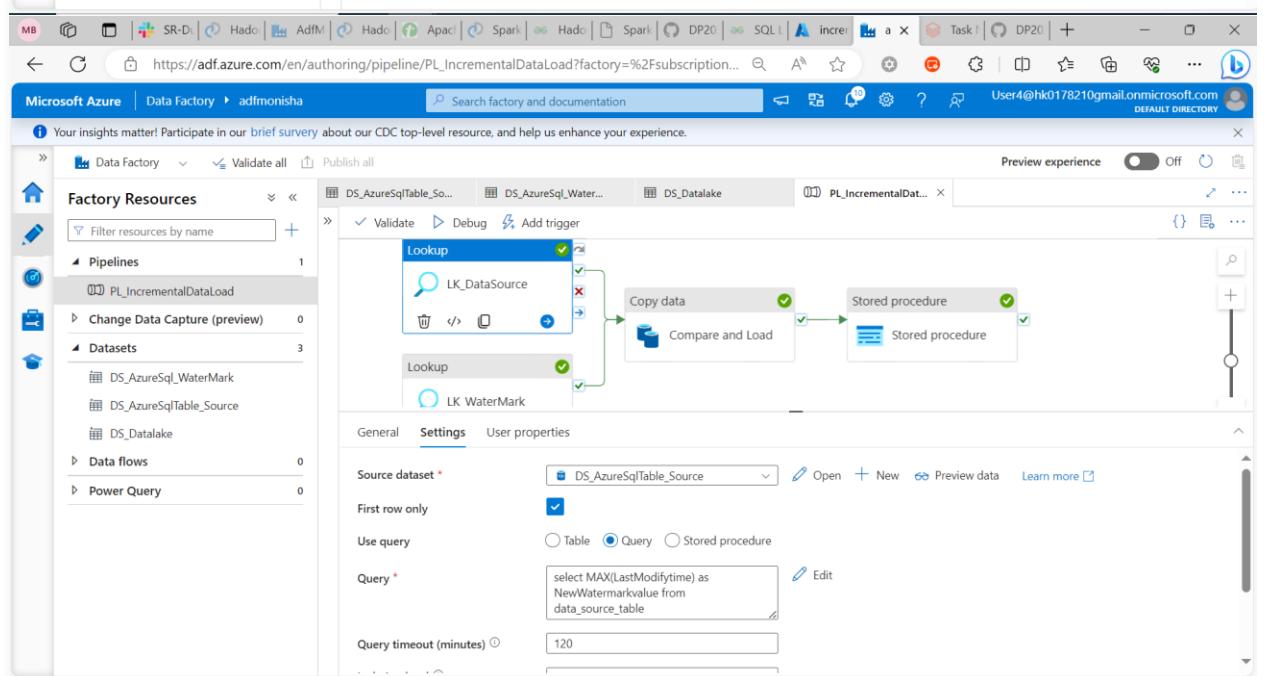
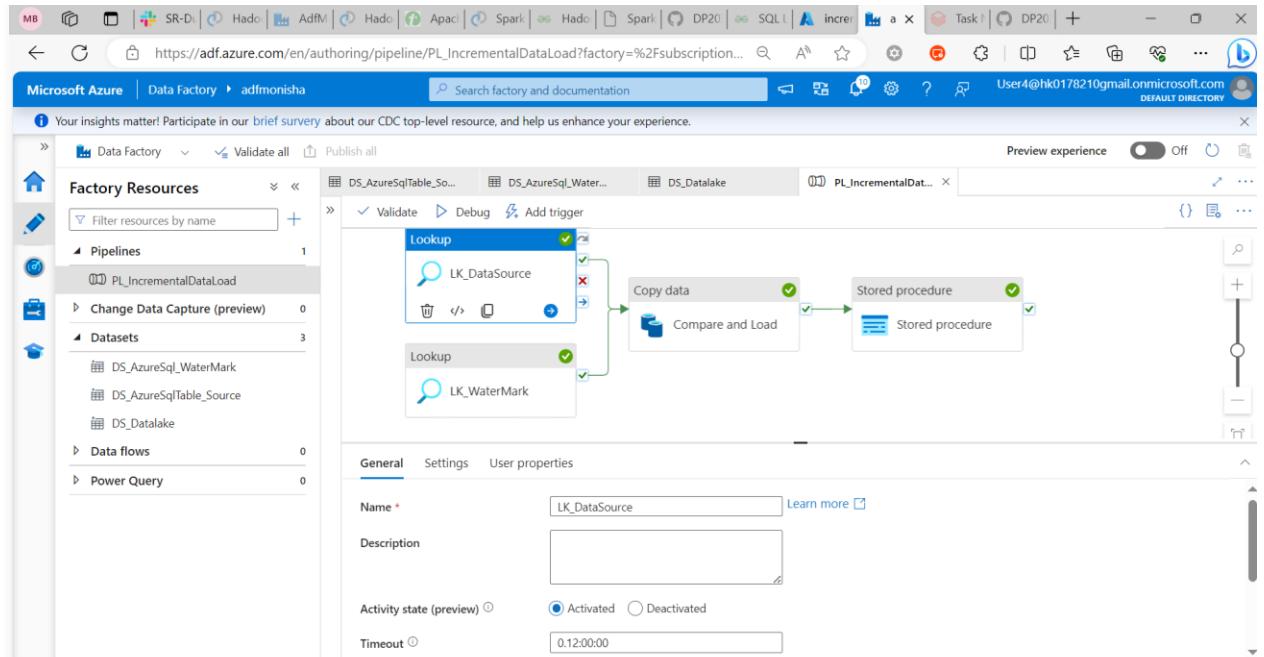
First row as header

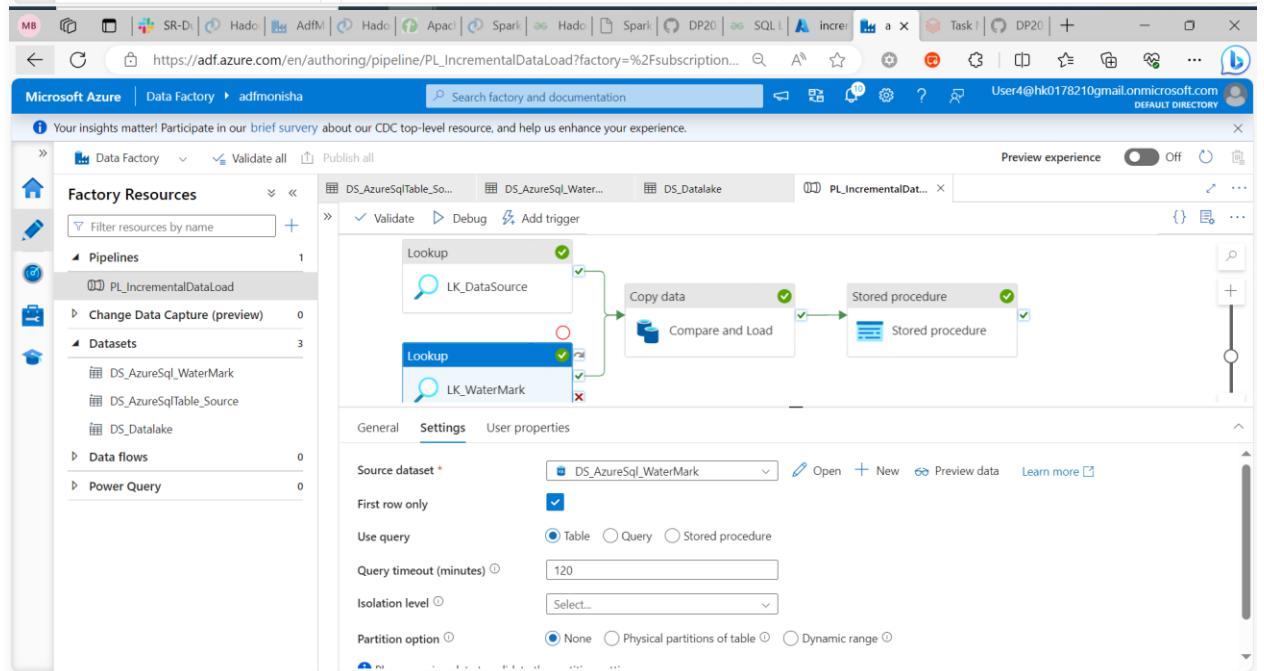
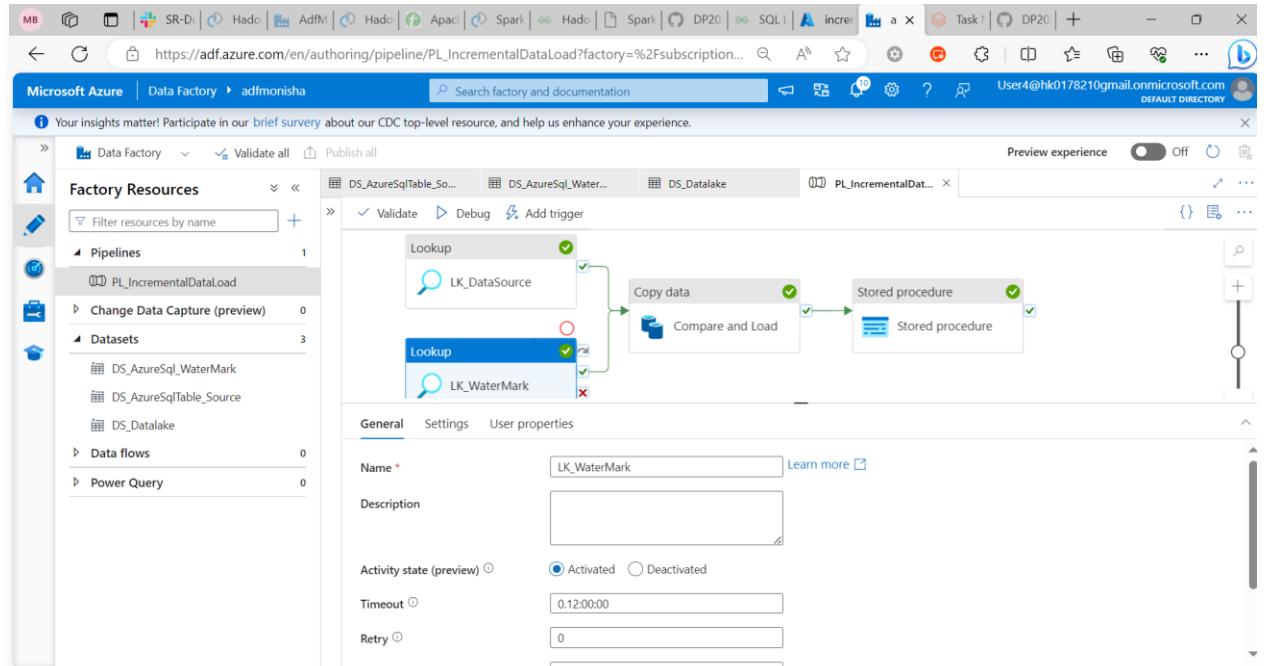
Import schema

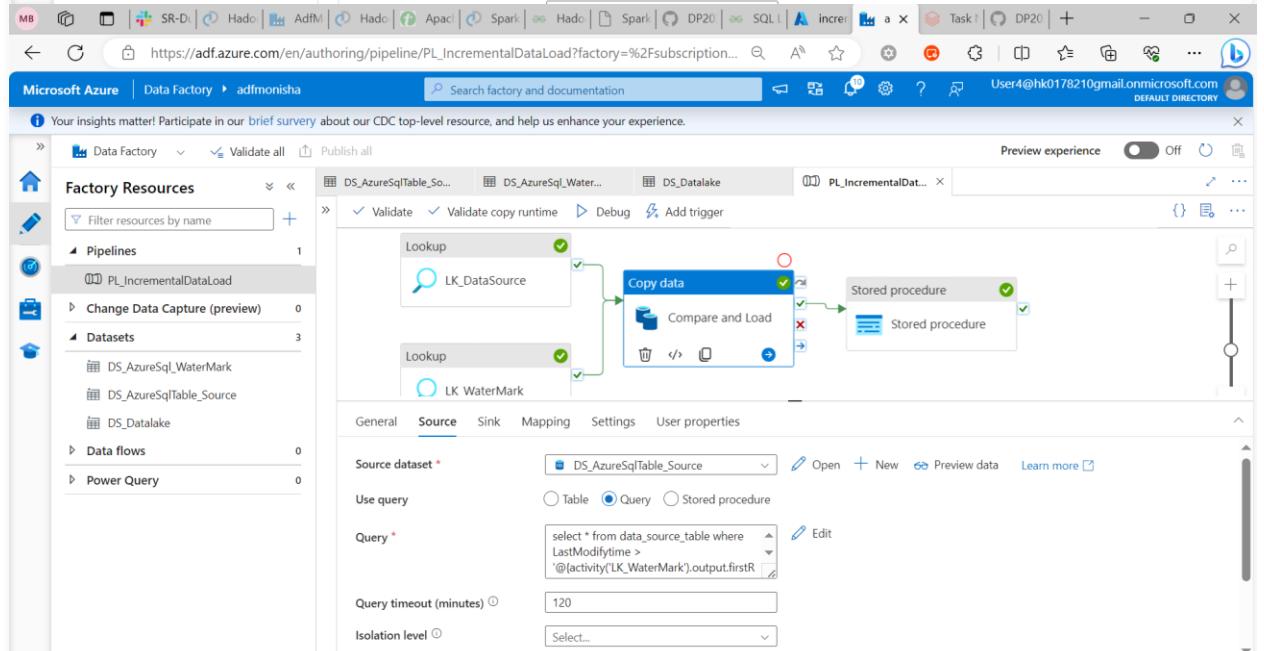
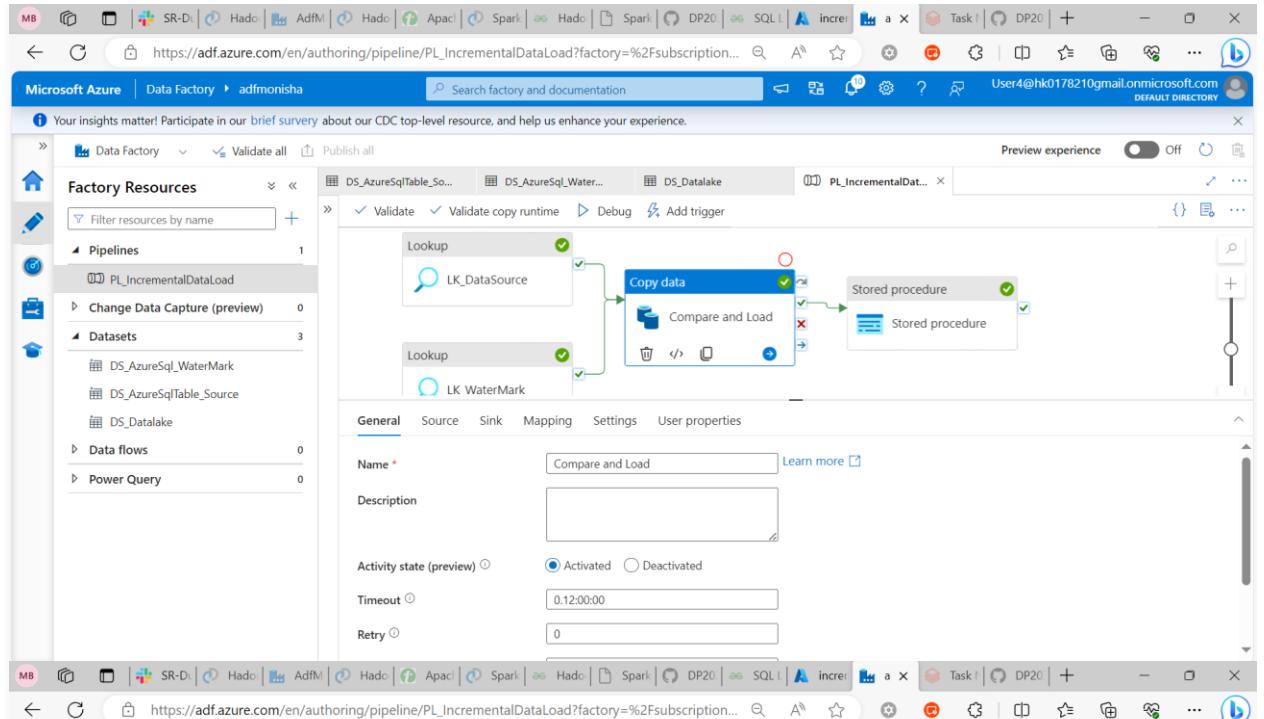
From connection/store From sample file None

OK Back Cancel









```

select * from data_source_table where LastModifytime >
'@{activity('LK_WaterMark').output.firstRow.WatermarkValue}' and
LastModifytime <=
'@{activity('LK_DataSource').output.firstRow.NewWatermarkvalue}'

```

Microsoft Azure | Data Factory > adfmonisha | https://adf.azure.com/en/authoring/pipeline/PL_IncrementalDataLoad?factory=%2Fsubscriptions%2F45f63a... | User4@hk0178210@gmail.onmicrosoft.com | DEFAULT DIRECTORY

Copy to clipboard

OK Cancel

Microsoft Azure | Data Factory > adfmonisha | https://adf.azure.com/en/authoring/dataset/DS_Datalake?factory=%2Fsubscriptions%2F45f63a... | User4@hk0178210@gmail.onmicrosoft.com | DEFAULT DIRECTORY

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us enhance your experience.

Factory Resources

- Pipelines: PL_IncrementalDataLoad (1)
- Datasets: DS_AzureSqlTable_Source, DS_AzureSqlTable_WaterMark, DS_Datalake (3)
- Data flows: 0
- Power Query: 0

DS_Datalake

DelimitedText

CSV

Connection Schema Parameters

Linked service: LS_AzureDataLakeStorage

File path: incrementaldataload / Directory: @CONCAT('Incremental-', pipeline().RunId, '.txt')

Compression type: Select...

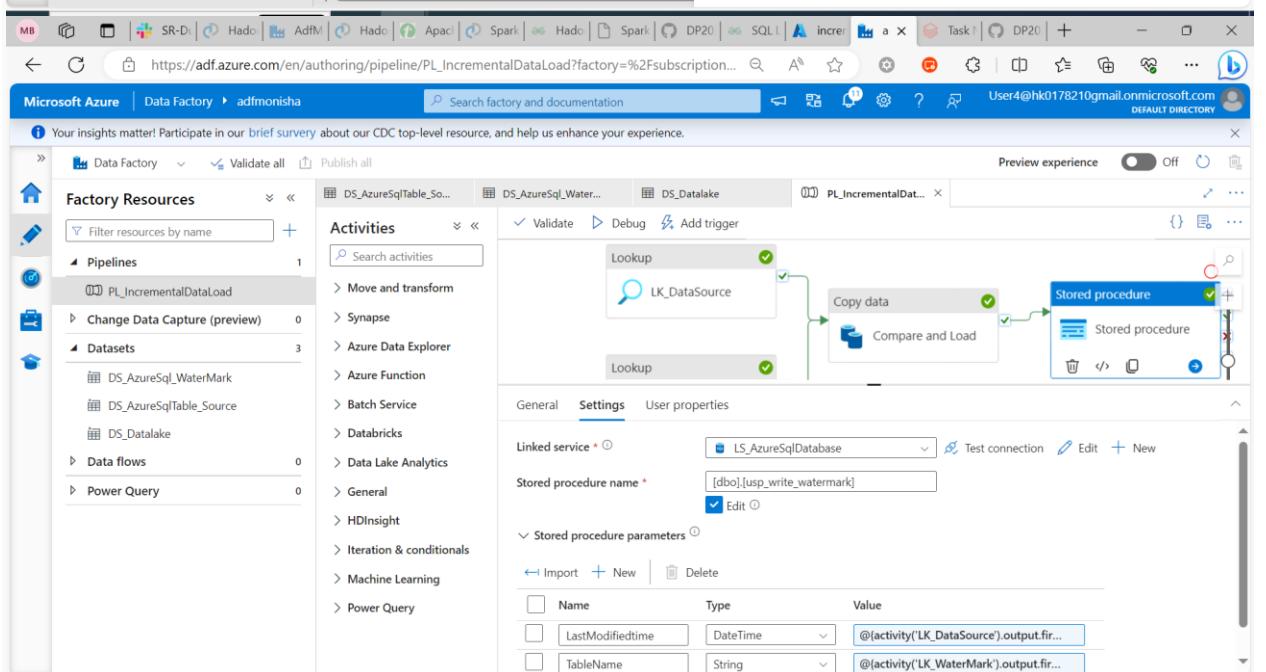
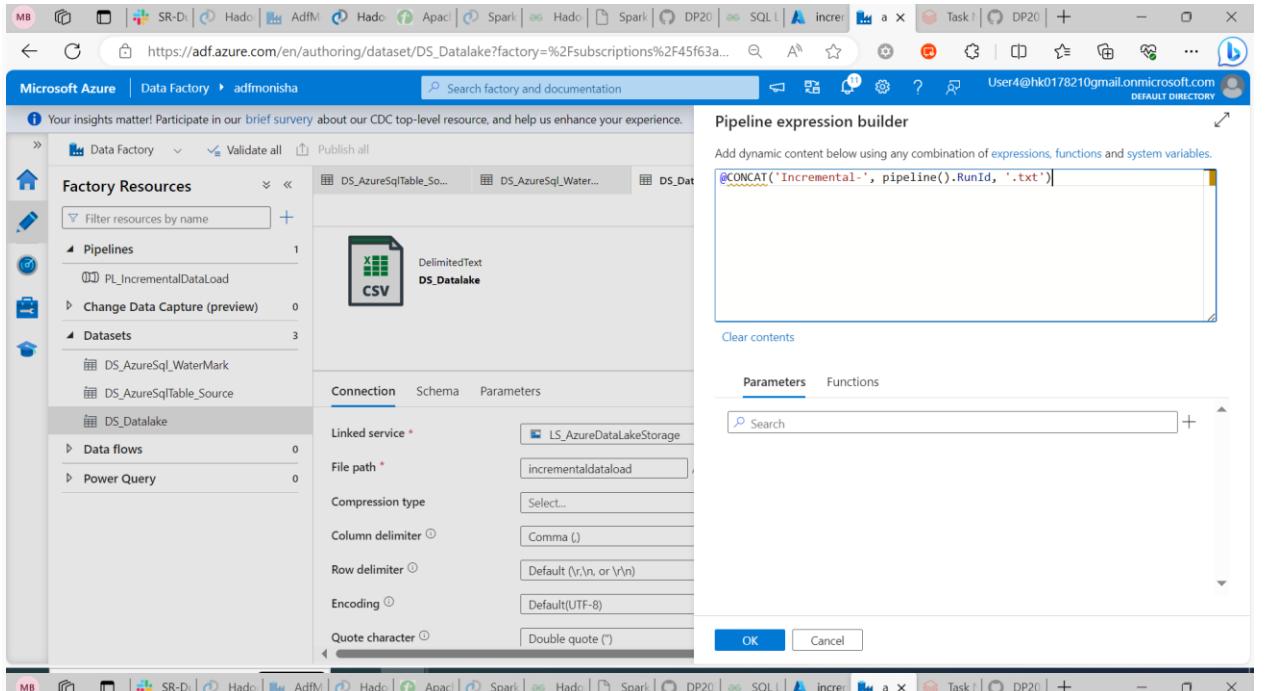
Column delimiter: Comma (,)

Row delimiter: Default (\r\n, or \n\r)

Encoding: Default(UTF-8)

Quote character: Double quote ("")

@CONCAT('Incremental-', pipeline().RunId, '.txt')

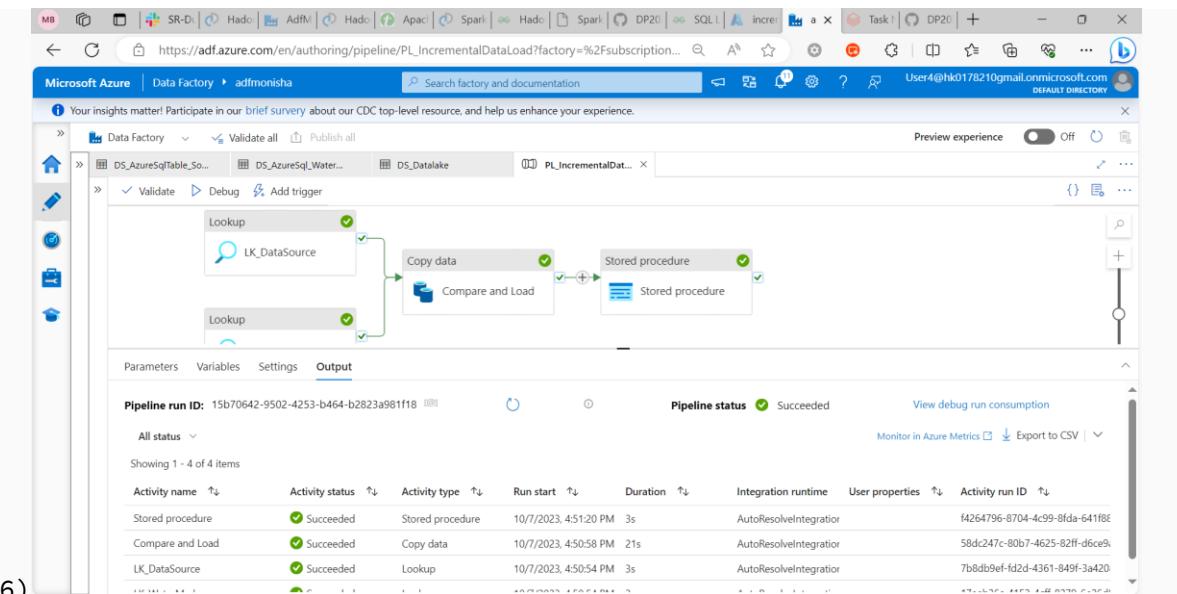


```
@{activity('LK_DataSource').output.firstRow.NewWatermarkvalue}
```

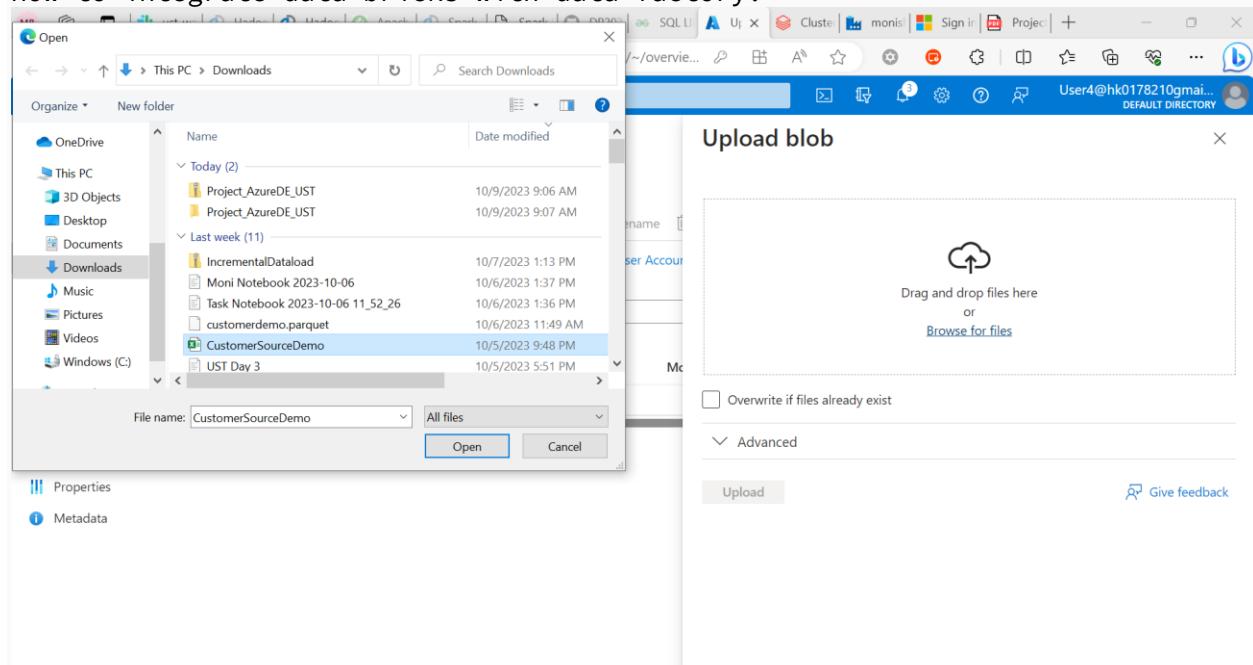
The screenshot shows the Microsoft Azure Data Factory Pipeline expression builder. A yellow box highlights the 'Activity outputs' tab in the bottom navigation bar. Under 'LK_WaterMark' activity outputs, the 'firstRow.TableName' output is selected. The pipeline expression in the main editor is: `@{activity('LK_WaterMark').output.firstRow.TableName}`. The pipeline name is PL_IncrementalDataLoad.

```
@{activity('LK_WaterMark').output.firstRow.TableName}
```

The screenshot shows the Microsoft Azure Data Factory Pipeline expression builder. A yellow box highlights the 'Activity outputs' tab in the bottom navigation bar. Under 'LK_WaterMark' activity outputs, the 'firstRow.TableName' output is selected. The pipeline expression in the main editor is: `@{activity('LK_WaterMark').output.firstRow.TableName}`. The pipeline name is PL_IncrementalDataLoad.



How to integrate data bricks with data factory:



Create new notebook →

The screenshot shows the Microsoft Azure Databricks interface. The left sidebar contains navigation links such as Workspace, Recents, Catalog, Workflows (selected), Compute, SQL, and various engineering and data management tools. The main area displays a 'Jobs' workflow named 'ScheduleNB'. A single task named 'ADLSConnectivity' is listed, defined as a Notebook type task running on a 'Job_cluster' cluster. An 'Unspecified path' is indicated. A modal window titled 'Select Notebook' is open, showing a list of workspace items: 'Repos', 'Shared', and 'Users'. The 'Confirm' button is visible at the bottom right of the modal.

Microsoft Azure | databricks

Workflows > Jobs >

ScheduleNB

Runs Tasks

Task name*

Type*

Source*

Path*

Cluster*

Cancel Create

Microsoft Azure | databricks

Select Notebook

Workspace Recents

Repos Shared Users

Trash Moni Notebook 2023-10-09

Cancel Confirm

The screenshot shows the Databricks interface for creating a new workflow. The left sidebar is titled 'Workflows' and includes options like Workspace, Recents, Catalog, and Workflows. The main area shows a task named 'ADLSConnectivity' with a sub-task 'Job_cluster'. The configuration form below includes fields for Type (Notebook), Source (Workspace), Path (/Users/user4@hk0178210gmail.onmicrosoft.com/Moni Notebook 2023-10-09), Cluster (Job_cluster), and Dependent libraries. Buttons for 'Cancel' and 'Create' are at the bottom right.

Schedule → add schedule

The screenshot shows the 'Schedule' configuration dialog. It includes fields for Trigger Status (Active), Trigger type (Scheduled), and a schedule entry for 'Every Day at 11:00' in UTC+05:30. A checkbox for 'Show cron syntax' is also present. Buttons for 'Cancel' and 'Save' are at the bottom right. A note at the bottom indicates the driver and workers configuration: 'Driver: Standard_DS3_v2 · Workers: Standard_DS3_v2 · 8 workers · 13.3 LTS Photon (includes Apache Spark 3.4.1, Scala 2.12)'.

ScheduleNB

Job details

- Job ID: 471810833626260
- Creator: User4
- Run as: User4
- Tags: +Tag

Git

Not configured

Schedule

At 11:00 AM (UTC+05:30) — Chennai, Kolkata, Mumbai, ...

Tasks

```
graph LR; ADLSConnectivity[ADLSConnectivity] --> notebook[notebook]
```

notebook

Path*: ..ers/user4@hk0178210gmail.onmicrosoft.com/Notebook 2023-10-09 12:01:59

Cluster*: Moni Cluster

Cancel Create task

ScheduleNB run

Job run details

- Job ID: 471810833626260
- Job run ID: 1072899809229998
- Launched: Manually
- Started: 10/09/2023, 12:07:38 PM
- Ended: 10/09/2023, 12:08:02 PM
- Duration: 24s
- Queue duration: -
- Status: Succeeded

Compute

Moni Cluster

Driver: Standard_DS3_v2 - Workers: Standard_DS3_v2 - 0 workers - 13.3 LTS Photon (includes Apache Spark 3.4.1,

DATA FACTORY TO DATABRICKS:

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, Data flows, and Power Query. The main workspace displays a pipeline named 'ADF TO DATABRICKS' with a single 'Copy data' activity selected. The 'Properties' pane on the right shows the activity's name as 'blob to datalake'. The 'Source' tab indicates the source is 'blob to datalake'. The 'Sink' tab shows a connection to 'Databricks'. The 'Mapping' tab is visible at the bottom.

This screenshot shows the 'Databricks linked service' configuration dialog within the Azure Data Factory pipeline editor. It is a modal window with several tabs: 'General', 'Azure Databricks' (selected), and 'Databricks linked service'. Under 'Azure Databricks', the 'Account selection method' section has 'From Azure subscription' selected. It shows an 'Azure subscription' dropdown set to 'Pay-As-You-Go (45f63aab-e46c-4be8-bb77-43fe31240084)', a 'Databricks workspace' dropdown set to 'databrickmonisha', and a 'Select cluster' section with 'Existing interactive cluster' selected. The 'Authentication type' section uses 'Access Token' and has an 'Access token' input field. At the bottom, there are 'Create' and 'Cancel' buttons, and a 'Test connection' link.

The screenshot shows the Microsoft Azure Databricks Settings page. On the left, there's a sidebar with various navigation options like Workspace, Recents, Catalog, Workflows, Compute, SQL, Data Engineering, Job Runs, Data Ingestion, and Delta Live Tables. The main area is titled 'Settings' under 'User'. A modal window titled 'Generate new token' is open. It has a 'Comment' field containing 'adf to databrick connectivity' and a 'Lifetime (days)' field with a value of 1. At the bottom of the modal are 'Cancel' and 'Generate' buttons. To the right of the modal, a message says 'Free trial ends in 14 days. Upgrade to Premium in Azure Portal'.

This screenshot shows the same Microsoft Azure Databricks Settings page after generating a token. The modal window now displays the generated token 'dapiac598d563badb4c64fb4e1d8543d83e8' in a text input field. There is a 'Copy' button next to the token. Below the token, a message says 'Make sure to copy the token now. You won't be able to see it again.' At the bottom of the modal are 'Done' and 'Expiration' buttons. The main settings page shows a message 'No tokens exist.' The bottom of the screen shows a taskbar with various icons and system status information.

The screenshot shows the Microsoft Azure Data Factory pipeline editor interface. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, Data flows, and Power Query. A pipeline named 'ADF TO DATABRICKS' is selected. In the main workspace, a 'Copy data' activity is being configured, with its source set to 'blob to datalake'. A modal dialog titled 'New linked service' is open, specifically for 'Databricks workspace'. The 'Databricks linked service' tab is selected. The configuration includes:

- Data workspace:** databricksonmonisha
- Select cluster:** Existing interactive cluster (radio button selected)
- Databrick Workspace URL:** https://adb-6462883714603594.14.azuredatabricks.net
- Authentication type:** Access Token
- Access token:** (Input field containing a redacted access token)
- Choose from existing clusters:** Moni Cluster

At the bottom of the modal are 'Create' and 'Cancel' buttons, along with a 'Test connection' link.

The screenshot shows the same Data Factory interface. A 'Browse' dialog is open, prompting 'Select a file or folder'. The path 'Root folder > Users > user4@hk0178210gmail.onmicrosoft.com' is shown. Two items are listed: 'Moni Notebook 2023-10-09' and 'Notebook 2023-10-09 12:01:59'. At the bottom of the dialog are 'OK' and 'Cancel' buttons, along with a note 'Showing 1 - 2 of 2 items'.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (including 'ADF TO DATABRICKS'), 'Datasets', 'Data flows', and 'Power Query'. The main workspace displays the 'Activities' section for the 'ADF TO DATABRICKS' pipeline. It shows a 'Copy data' activity ('blob to datalake') and a 'Notebook' activity ('notebook'). Both activities are marked as 'Succeeded'. The 'Properties' panel on the right shows the pipeline name as 'ADF TO DATABRICKS'.

To check:

The screenshot shows the Microsoft Azure Databricks interface. The left sidebar includes 'Workspace', 'Recents', 'Catalog', 'Workflows', 'Compute', 'SQL', 'SQL Editor', 'Queries', 'Dashboards', 'Alerts', 'Query History', 'SQL Warehouses', 'Data Engineering', 'Job Runs' (which is selected), and 'Data Ingestion'. The main area is titled 'Workflows' and shows a timeline from October 8 to October 9, 2023. A green bar indicates a successful job run. Below the timeline is a table of job runs:

Start time	Job	Run as	Launched	Duration	Status	Run parameters
Oct 09, 2023, 12:17 PM	ADF_adfpravi_Data...	User5	By runs submit API	11s	Succeeded	
Oct 09, 2023, 12:17 PM	ADF_adfmonisha_...	User4	By runs submit API	12s	Succeeded	
Oct 09, 2023, 12:08 PM	Schedule Notebook	User5	Manually	25s	Succeeded	
Oct 09, 2023, 12:07 PM	schedule notebook	User5	Manually	24s	Succeeded	

nono

SQL TO DATABRICKS CONNECTION:

- 1) Go to sql database → in query editor → create table :

```
CREATE TABLE [dbo].[topmovies_comma] (
    [Index] [BIGINT] NULL,
    [MovieTitle] varchar(500) NULL
);
```

The screenshot shows the Microsoft Azure portal interface for a SQL database named 'SQLMonisha'. On the left, there's a navigation sidebar with options like Overview, Activity log, Tags, Diagnose and solve problems, and Query editor (preview). The main area displays a query editor titled 'Query 1' with the following SQL code:

```
1 CREATE TABLE [dbo].[topmovies_comma] (
2     [Index] [BIGINT] NULL,
3     [MovieTitle] varchar(500) NULL
4 );
5
6 SELECT * FROM [dbo].[topmovies_comma]
```

Below the code, the 'Results' tab shows the message: 'Query succeeded: Affected rows: 0'. At the bottom right of the editor, it says 'Query succeeded | 0s'.

- 2) Go to datalake storage → in refined container upload
top_movies_part3.csv file → upload

The screenshot shows the Microsoft Azure portal interface for a storage account container named 'refined'. The left sidebar includes options like Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access tokens, Manage ACL, Access policy, Properties, and Metadata. On the right, there's an 'Upload blob' dialog box. Inside the dialog, a file named 'TopMovies_Part3.csv' is selected and ready to be uploaded. The dialog also includes an 'Overwrite if files already exist' checkbox and an 'Upload' button at the bottom right.

The screenshot shows the Microsoft Azure Storage Container blade for the 'refined' container. The 'Overview' tab is selected. A single blob named 'TopMovies_Part3.csv' is listed. The blob has a modified date of 10/9/2023, 12:33:37, an access tier of Hot (Inferred), and is a Block blob. The size is 1.06 MB.

3) In sql data base → connection strings → jdbc

⇒ Give password we can connect

⇒ (OR)

4) In data bricks notebook → connect → read file →

```
spark.conf.set(
    'fs.azure.account.key.adlsmonisha.dfs.core.windows.net',
    'RNJlp5aBBdPy5cUmh229dWq6hBPRj7iRl3pR6LPUmAoiPUwm/itC4Xm3qmwIxYd
WGGScplHn8Wxe+ASTem7ZDg=='
)
```

```
dbutils.fs.ls('abfss://refined@adlsmonisha.dfs.core.windows.net')
```

```
source_file =
'abfss://refined@adlsmonisha.dfs.core.windows.net/TopMovies_Part3.csv'
df = spark.read\
    .format('csv')\
    .option('inferSchema', True)\
    .option('header', True)\
    .option('delimiter', ',')\
    .load(source_file)
```

```
display(df)
```

```

spark.conf.set(
    'fs.azure.account.key.adlsmonisha.dfs.core.windows.net',
    'RNJlp5aBBdPy5cUmh229dWq6hBPRj7iRL3pR6LPUmAoIPUwm/itC4Xm3qmwIxYdWGGScp1Hn8Wxe+ASem7ZDg=='
)

```

```

dbutils.fs.ls('abfss://refined@adlsmonisha.dfs.core.windows.net')

```

```

source_file = 'abfss://refined@adlsmonisha.dfs.core.windows.net/TopMovies_Part3.csv'
df = spark.read\
    .format('csv')\
    .option('inferSchema', True)\ 
    .option('header', True)\ 
    .option('delimiter', ',')\ 
    .load(source_file)

```

Output:

Index	MovieTitle
1	12 Years a Slave
2	Ben-Hur
3	Persona
4	The Grand Budapest Hotel
5	Million Dollar Baby
6	Amores Perros
7	The Masacre
50	rows 0.60 seconds runtime

- 5) Go to data bricks note book → change logicalservername, databasename, tablename, username, password

```

logicalServername = "monishadb.database.windows.net"
databaseName = "SQLMonisha"
tableName = "topmovies_comma"
userName = "Adminlogin"
password = "Monisha@ishu" # Please specify password here

```

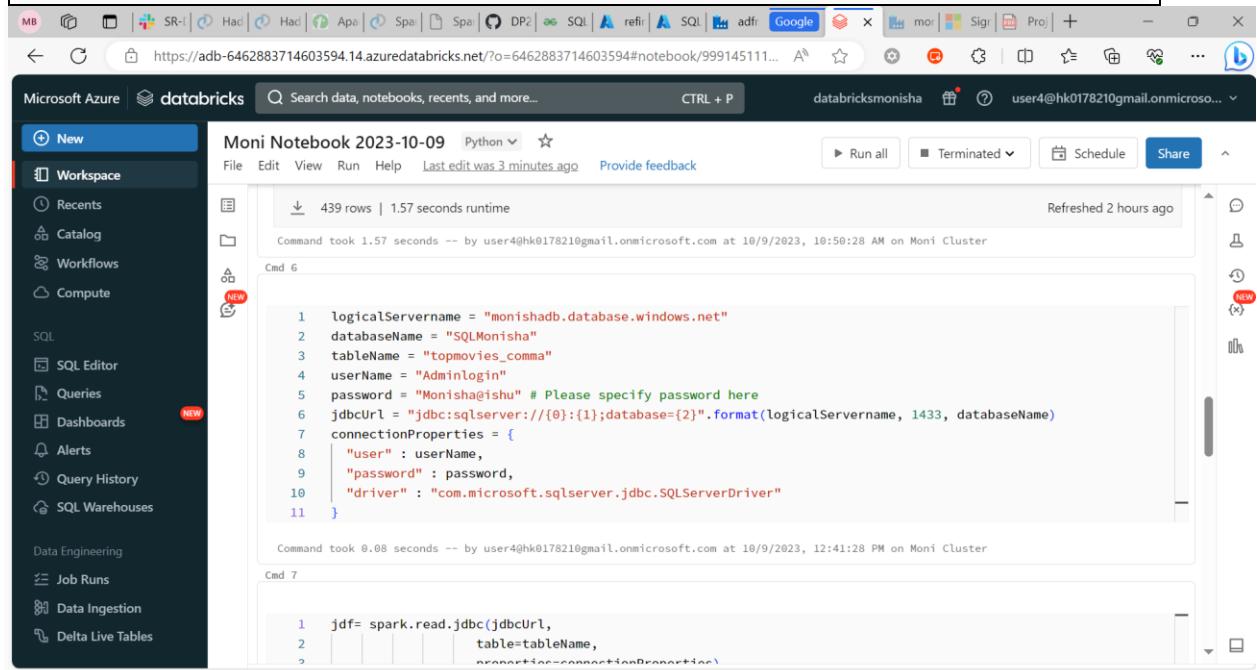
```
jdbcUrl =  
"jdbc:sqlserver://{0}:{1};database={2}".format(logicalServername  
, 1433, databaseName)  
connectionProperties = {  
    "user" : userName,  
    "password" : password,  
    "driver" : "com.microsoft.sqlserver.jdbc.SQLServerDriver"  
}
```

TO READ THE FILE:

```
jdf= spark.read.jdbc(jdbcUrl,  
                      table=tableName,  
                      properties=connectionProperties)
```

TO WRITE THE FILE:

```
df.write.jdbc(jdbcUrl,  
             mode = "append",  
             table=tableName,  
             properties=connectionProperties)
```



The screenshot shows two separate sessions of a Databricks notebook named "Moni Notebook 2023-10-09" in Python.

Session 1 (Top):

- Cmd 9:** A command to write data to a JDBC table. It uses `df.write.jdbc(jdbcUrl, ..., mode="append", ..., table=tableName, properties=connectionProperties)`.
- Cmd 10:** A command to read data from a JDBC table. It uses `jdf= spark.read.jdbc(jdbcUrl, ..., table=tableName, properties=connectionProperties)`.
- Result of Cmd 10:** Displays the DataFrame `jdf` as a table with columns `Index` and `MovieTitle`. The data is as follows:

Index	MovieTitle
1	12 Years a Slave
2	Ben-Hur
3	Persona
4	The Grand Budapest Hotel
5	Million Dollar Baby
6	Amores Perros
7	The Maccanns

Session 2 (Bottom):

- Cmd 11:** A command to display the DataFrame `jdf`, which is shown as a table with the same data as above.

6)

```
# Details about connection string
logicalServername = "sqlserver2023.database.windows.net"
databaseName = "sqldb"
tableName = "CustomerSource"
userName = "Adminlogin"
password = "Welcome@123" # Please specify password here
jdbcUrl =
"jdbc:sqlserver://{}:{};database={}".format(logicalServername, 1433,
databaseName)
```

```

connectionProperties = {
    "user" : userName,
    "password" : password,
    "driver" : "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

df.write.jdbc(jdbcUrl,
              mode ="append",
              table=tableName,
              properties=connectionProperties)

jdf= spark.read.jdbc(jdbcUrl,
                     table=tableName,
                     properties=connectionProperties)

```

ANOTHER WAY TO CONNECT SQL:

- 1) Go to SQL Database → Connection strings → jdbc → COPY THAT url
- 2) Go to databricks → paste jdbc url and change password

```

sqlconnect =
"jdbc:sqlserver://monishadb.database.windows.net:1433;data
base=SQLMonisha;user=Adminlogin@monishadb;password=Monisha
@ishu;encrypt=true;trustServerCertificate=false;hostNameIn
Certificate=*.database.windows.net;loginTimeout=30;"
```

- 3) READ THE FILE : CHANGE variable name as sqlconnect and table name can be anything.

```

jjdf= spark.read.jdbc(sqlconnect,
                     table="topmovies_comma")
```

```

Moni Notebook 2023-10-09 Python ⚡
File Edit View Run Help Last edit was 4 minutes ago Provide feedback
Run all • Moni Cluster v Schedule Share
1 sqlconnect = "jdbc:sqlserver://monishadb.database.windows.net:1433;database=SQLMonisha; user=Adminloggingmonishadb;password=Monisha@ishu;encrypt=true;trustServerCertificate=false; hostNameInCertificate=*.database.windows.net;loginTimeout=30;" Command took 0.05 seconds -- by user4@hk0178210gmail.onmicrosoft.com at 10/9/2023, 1:22:09 PM on Moni Cluster
2 Cmd 13
3 Cmd 14
4 display(jjdf)
5 (1) Spark Jobs
6 Table + New result table: OFF v
7 Index MovieTitle
8 1 201 12 Years a Slave

```

HOW TO CREATE EVENT HUB NAMESPACE:

Showing 1 to 20 of 43 results for 'EVENT HUB'. [Clear search](#)

Service	Description	Provider	Action
Mandiant Managed Service & Azure Event Hub	Integrate Azure Event Hub to enable service delivery with Managed Defense for Defender for Endpoint.	Mandiant	Create
Event Hubs	Cloud-scale fully managed streaming platform supporting Apache Kafka and AMQP clients and services.	Azure Service	Create
Event Hubs Cluster	Ingest, route and store Apache Kafka and AMQP streams with millions of events for real-time analytics and apps.	Azure Service	Create
Event Grid Domain	Manage Event Grid topics with a unified authentication model.	Azure Service	Create

Create Namespace

Event Hubs

Resource group * RG_UST_MONISHA

Namespace name * ehubmoninamespace2023

Location * East US

Pricing tier * Standard (~\$22 USD per TU per Month)

Throughput Units 1

Enable Auto-Inflate

Review + create < Previous Next: Advanced >

ehubmoninamespace23 | Overview

Deployment

Deployment succeeded

Deployment 'ehubmoninamespace23' to resource group 'RG_UST_MONISHA' was successful.

Overview

Deployment name : ehubmoninamespace23
Subscription : Pay-As-You-Go
Resource group : RG_UST_MONISHA

Start time : 10/9/2023, 3:28:04 PM
Correlation ID : a827d77e-07b7-4949-8a82-4e...

Deployment details

Next steps

Cost management
Get notified to stay within your budget and prevent unexpected charges on your bill.
[Set up cost alerts >](#)

Microsoft Defender for Cloud
Secure your apps and infrastructure
[Go to Microsoft Defender for Cloud >](#)

Free Microsoft tutorials
[Start learning today >](#)

Work with an expert
Azure experts are service provider partners

Go to resource

The screenshot shows two instances of the Microsoft Azure portal interface. Both instances are for the same Event Hubs Namespace, "ehubmoninamespace23".

Left Instance (Top):

- The left sidebar shows navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Generate data (preview), Events, Scale, Geo-Recovery, Networking, and Encryption.
- The main area displays "Shared access policies" under "Settings". A single policy named "RootManageSharedAccessKey" is listed, with the claim "Manage, Send, Listen".

Right Instance (Bottom):

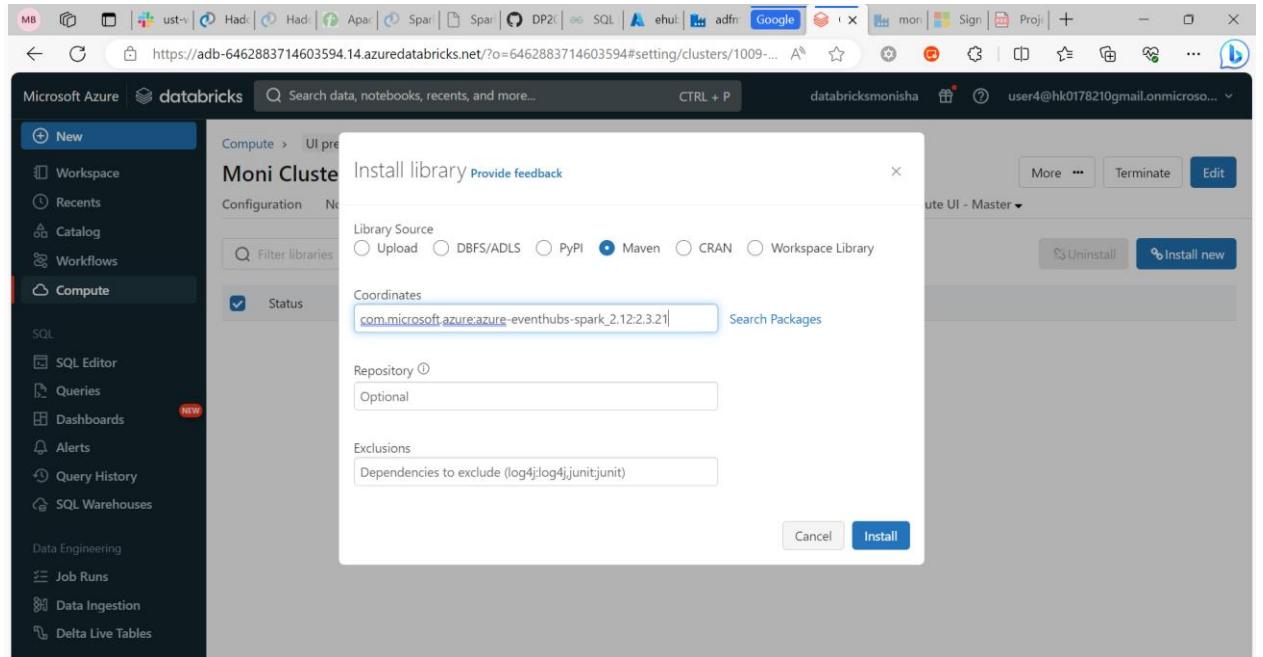
- The right sidebar shows the same navigation options.
- The main area displays the "SAS Policy: RootManageSharedAccessKey" configuration page.
- Checkboxes for "Manage", "Send", and "Listen" are checked.
- The "Primary key" field contains the value: "+NiRaLsL8dRiuOAtrma8uupK+4QwMX+vXC+A EhDnyZ+o=".
- The "Secondary key" field contains the value: "b8kALHX3t4WedmxN/1mJxmoSHGTmGrDw7+A EhDMgcX0=".
- A yellow box highlights the "Connection string-primary key" field, which contains the value: "Endpoint=sb://ehubmoninamespace23.servicebus.windows.net/SharedAccessKeyNam...". A "Copy" button is shown next to it.
- The "Connection string-secondary key" field contains the value: "Endpoint=sb://ehubmoninamespace23.servicebus.windows.net/SharedAccessKeyNam...".
- The "SAS Policy ARM ID" field contains the value: "/subscriptions/45f63aab-e46c-4be8-bb77-43fe31240084/resourcegroups/RG_UST_MO...".

EVENT CONSUMER: CONNECT WITH EVENTHUB:

New → notebook → Name : ADB to Eventhub

Compute → cluster → libraries → install new

com.microsoft.azure:azure-eventhubs-spark_2.12:2.3.21



```
import pyspark
from pyspark.sql.functions import *
from pyspark.sql import *
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json
from pyspark.sql.types import StructType, StructField, StringType,
IntegerType

##connect with evethub
ehConf = {}
ehConf['eventhubs.connectionString'] =
sc._jvm.org.apache.spark.eventhubs.EventHubsUtils.encrypt("Endpoint=sb
://ehubmoninamespace23.servicebus.windows.net/;SharedAccessKeyName=Roo
tManageSharedAccessKey;SharedAccessKey=+NiRaLsL8dRlu0Atma8uupK+4QwMX+v
XC+AEhDnyZ+o=;EntityPath=ehubmoni")

##read stream data

df_read_stream = (spark.readStream
    .format("eventhubs")
    .options(**ehConf)
    .load())

bodyDF = df_read_stream.select(col("body").cast("STRING"))
display(bodyDF, streamName= "bodyDF")
```

File Edit Selection View Go Run Terminal Help sendevent_eventhub_python_script.py - Downloads - Visual Studio Code

EXPLORER DOWNLOADS event.py sendevent_eventhub_python_script.py

```

1 import time
2 import os
3 import uuid
4 import datetime
5 import random
6 import json
7
8 import azure.eventhub
9 from azure.eventhub import EventHubProducerClient, EventData
10
11 # This script simulates the production of events for 10 devices.
12 devices = []
13 for x in range(0, 10):
14     devices.append(str(uuid.uuid4()))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python + ⚡

Ln 23, Col 153 Spaces: 4 UTF-8 LF (Python 3.11.5 64-bit (system))

<https://adb-6462883714603594.14.azuredatabricks.net/>/?o=6462883714603594#notebook/315256246...

Microsoft Azure | databricks Search data, notebooks, recents, and more... CTRL + P databricksmonisha Run all Moni Cluster Schedule Share

ADB TO EventHub Python

New Workspace Recents Catalog Workflows Compute SQL Editor Queries Dashboards Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Delta Live Tables

File Edit View Run Help Last edit was 6 minutes ago Provide feedback

Run all Moni Cluster Schedule Share

Table + New result table: OFF

body

1	{"id": "fe5bd042-ec35-4e49-90c2-0c36fb605d57", "timestamp": "2023-10-09 12:42:14.224999", "temperature": 70, "humidity": 89}
2	{"id": "34196288-2aff-4430-bdf1-c027c307d0e2", "timestamp": "2023-10-09 12:42:14.226099", "temperature": 80, "humidity": 80}
3	{"id": "97a36868-f764-47f4-af18-06d4179f96c7", "timestamp": "2023-10-09 12:42:14.227125", "temperature": 85, "humidity": 98}
4	{"id": "a2ab7c5f-b5ce-458a-b587-e10119520bba", "timestamp": "2023-10-09 12:42:14.228121", "temperature": 75, "humidity": 84}
5	{"id": "bfd9e820-01cb-4c16-86cc-90b625f74003", "timestamp": "2023-10-09 12:42:14.229120", "temperature": 80, "humidity": 100}
6	{"id": "42d34724-662a-435d-b560-6cb9d0639b28", "timestamp": "2023-10-09 12:42:14.230118", "temperature": 76, "humidity": 99}
7	{"id": "49b8dd58-fcf3-4d34-aee4-884124e41b52", "timestamp": "2023-10-09 12:42:14.231113", "temperature": 86, "humidity": 99}
8	{"id": "d6dbe71e-b2ee-427c-84b9-047ea6bf808f", "timestamp": "2023-10-09 12:42:14.231113", "temperature": 70, "humidity": 96}
9	{"id": "217704f5-649a-40dd-b153-d43fad0b2327", "timestamp": "2023-10-09 12:42:14.232124", "temperature": 87, "humidity": 87}

1000 rows | Truncated data

Cancel after getting 1000 rows

TO AUTOGENERATE SCRIPTS :

```
import time
import os
import uuid
import datetime
import random
import json

import azure.eventhub
from azure.eventhub import EventHubProducerClient, EventData

# This script simulates the production of events for 10 devices.
devices = []
for x in range(0, 10):
    devices.append(str(uuid.uuid4()))

# Create a producer client to produce and publish events to the event hub.
producer =
EventHubProducerClient.from_connection_string(conn_str="Endpoint=sb://ehubmoninam
espace23.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;Sh
aredAccessKey=+NiRaLsL8dRluOAtma8uupK+4QwMX+vXC+AEhDnyZ+o=",
eventhub_name="ehubmoni")

for y in range(0,2000):      # For each device, produce 20 events.
    event_data_batch = producer.create_batch() # Create a batch. You will add
events to the batch later.
    for dev in devices:
        # Create a dummy reading.
        reading = {'id': dev, 'timestamp': str(datetime.datetime.utcnow()),
'temperature': random.randint(70, 100), 'humidity': random.randint(70, 100)}
        s = json.dumps(reading) # Convert the reading into a JSON string.
        event_data_batch.add(EventData(s)) # Add event data to the batch.
        print("events proccessing", s)
    producer.send_batch(event_data_batch) # Send the batch of events to the event
hub.

# Close the producer.
producer.close()
```

In visual studio code: right click on the file ➔ click run python file in terminal

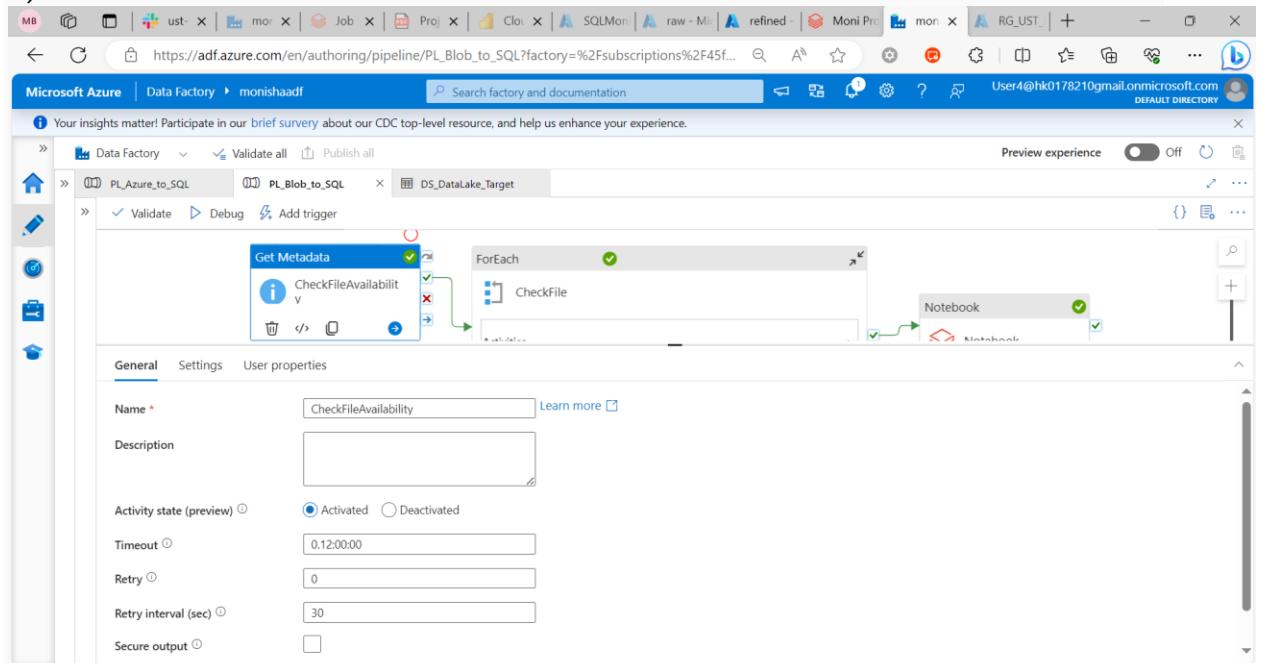
In terminal, cd to downloads ➔ cmd(command prompt) ➔ python file_name.py(python sendevent_eventhub_python_script.py)

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Terminal:** sendevent_eventhub_python_script.py - Downloads - Visual Studio Code
- Explorer:** Shows a list of files in the Downloads folder, including:
 - Monisha.Baskar.ipynb
 - monisha.pem
 - MonishaVMforSHIR.rdp
 - ms.monishabaskarblrtrv.pdf
 - MySQL.pdf
 - neweventhubfile.py
 - OCIF2023CA (1).jpg
 - omprenvmmoni.rdp
 - Person.csv
 - {} Person.json
 - Power Bi Assignment.pdf
 - Practice 1.ipynb
 - Project_AzureDE_UST.zip
 - python-3.11.5-amd64.exe
 - query (1).sql
 - query.sql
 - report.csv
 - sales.ass.pbix
 - Sales.csv
 - sendevent_eventhub_python...
 - Shruti_Oracle_certificate.pdf
 - Slides1234.pdf
- Code Editor:** The script file contains Python code for sending events to Azure Event Hub. It imports time, os, uuid, datetime, random, json, and azure.eventhub. It defines a function to simulate event production for 10 devices. The loop iterates from 0 to 10, generating a unique UUID for each device and appending it to a list. Finally, it prints a series of JSON objects representing events with various IDs, timestamps, temperatures, and humidity levels.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, along with a Python icon and other standard VS Code icons.

ASSIGNMENT:

1)



Microsoft Azure | Data Factory | monishaadf

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us enhance your experience.

Preview experience: Off

DS_Blob_Filename

Connection Schema Parameters

Linked service: LS_AzureBlobStorage

File path: raw / @dataset().FileName

Compression type: Select...

Column delimiter: Comma (,)

Row delimiter: Default (\r\n, or \n)

Encoding: Default(UTF-8)

Quote character: Double quote (")

Microsoft Azure | Data Factory | monishaadf

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us enhance your experience.

Preview experience: Off

PL_Blob_to_SQL

General Settings User properties

Dataset: DS_Blob_Filename

Dataset properties:

Name	Value	Type
FileName	@variables('FileName')	string

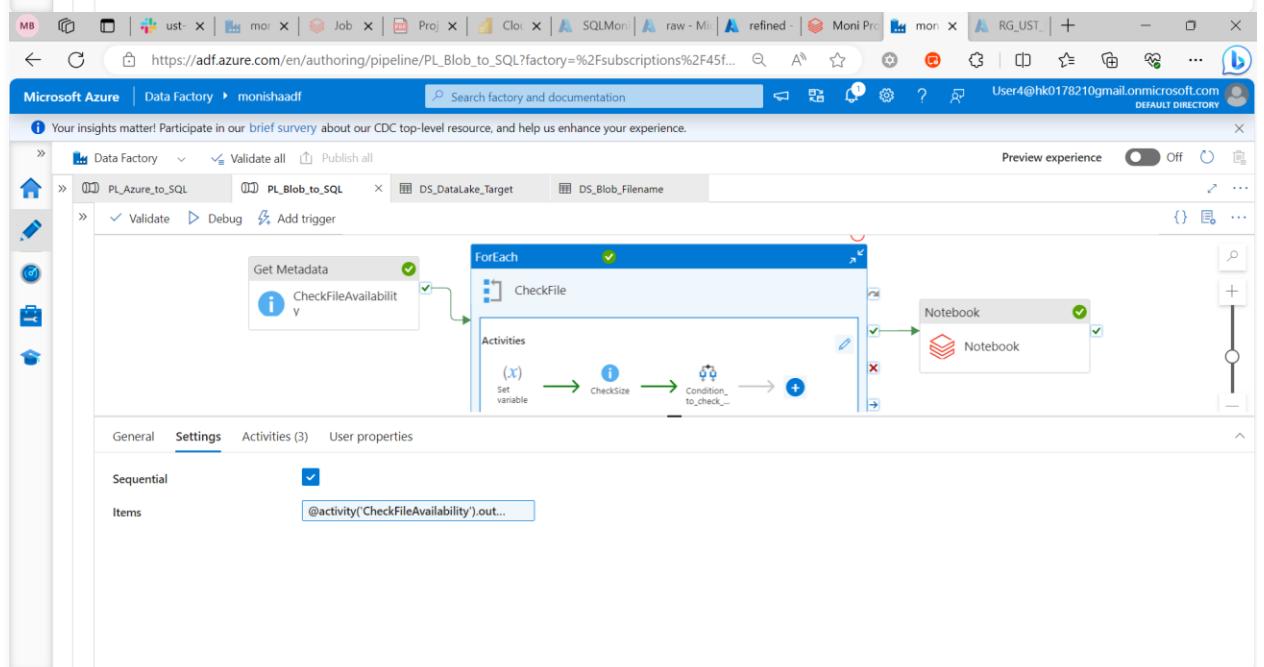
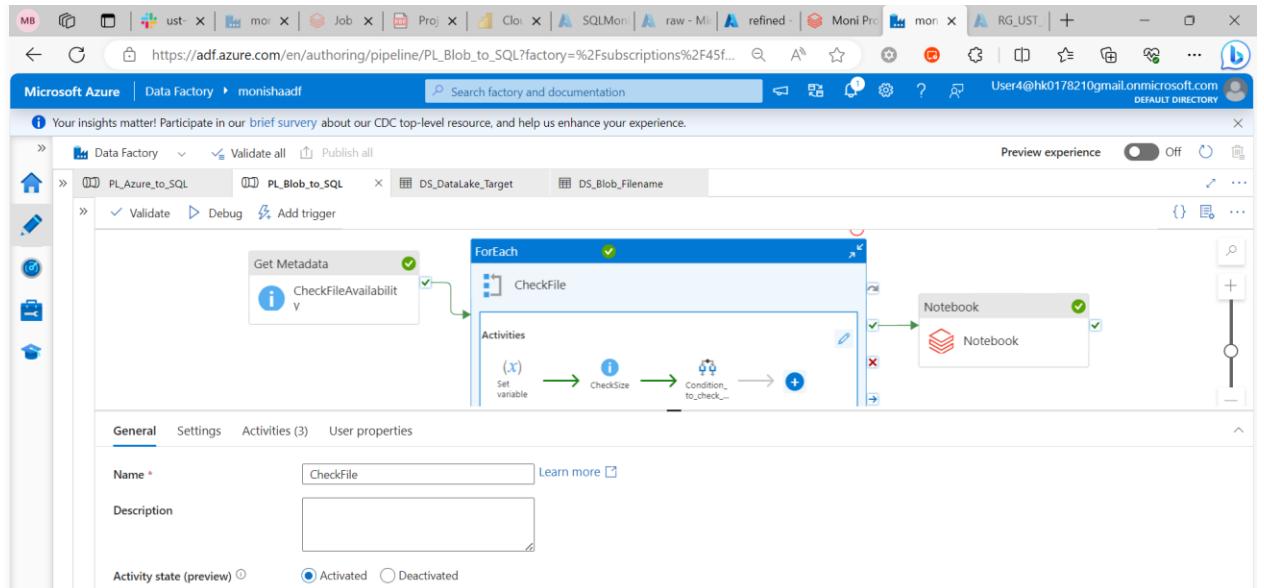
Field list:

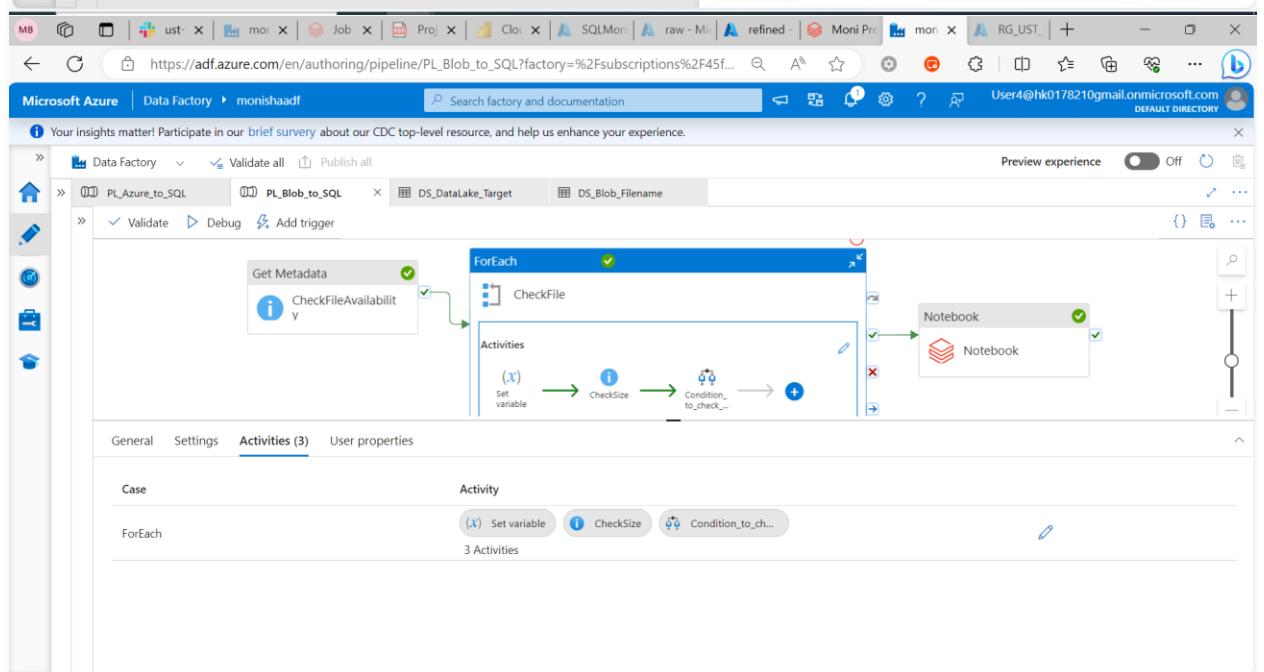
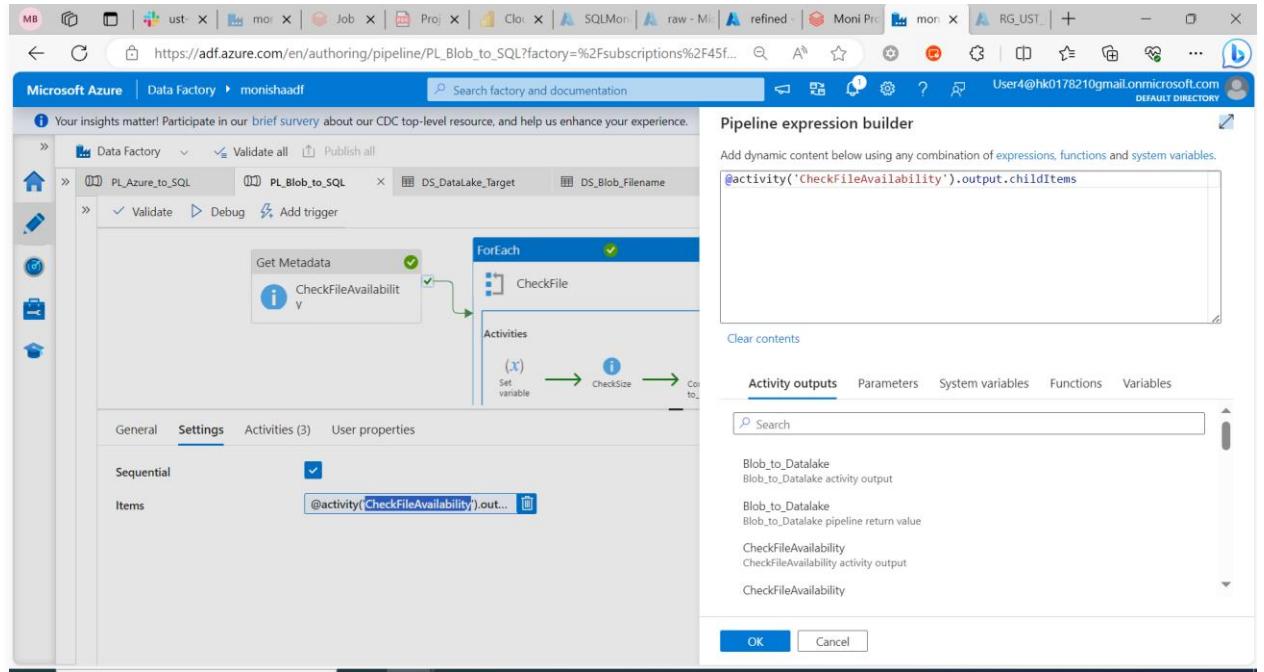
- + New | Delete
- Argument
- Child items

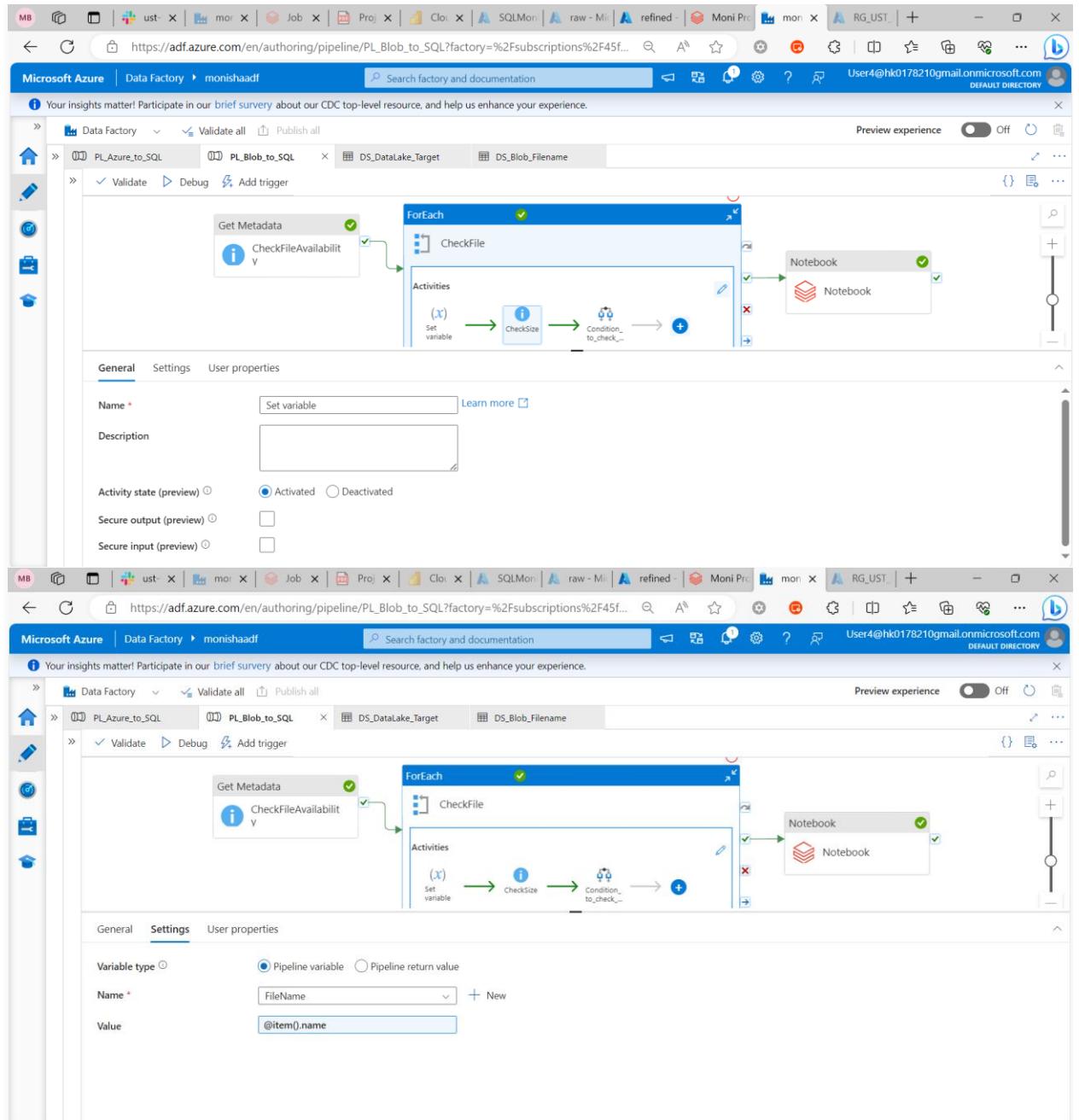
Filter by last modified:

Start time (UTC) End time (UTC)

The screenshot displays two main sections of the Azure Data Factory interface. The top section shows the configuration for a dataset named 'DS_Blob_Filename'. It includes fields for connection (LS_AzureBlobStorage), file path (raw / @dataset().FileName), and various file format settings like compression, delimiters, encoding, and quote characters. The bottom section shows a pipeline named 'PL_Blob_to_SQL' with a preview view. This pipeline consists of a 'Get Metadata' activity followed by a 'ForEach' loop. Inside the loop, there is a 'CheckFile' activity and a 'Notebook' activity. The pipeline is currently set to 'Validate' mode. Both sections include a 'Settings' tab for further configuration.







Microsoft Azure | Data Factory > monishaadf

https://adf.azure.com/en/authoring/pipeline/PL_Blob_to_SQL?factory=%2Fsubscriptions%2F45f...

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us enhance your experience.

Preview experience Off

Data Factory > PL_Blob_to_SQL > DS_DataLake_Target > DS_Blob_Filename

Validate all Publish all

CheckFile

Get Metadata

Set variable

If Condition

Condition_to_check

True

Blob_to_DataLake

General Settings User properties

Name * CheckSize Learn more

Description

Activity state (preview) Activated Deactivated

Timeout 0.12:00:00

Retry 0

Microsoft Azure | Data Factory > monishaadf

https://adf.azure.com/en/authoring/pipeline/PL_Blob_to_SQL?factory=%2Fsubscriptions%2F45f...

Your insights matter! Participate in our brief survey about our CDC top-level resource, and help us enhance your experience.

Preview experience Off

Data Factory > PL_Blob_to_SQL > DS_DataLake_Target > DS_Blob_Filename

Validate all Publish all

CheckFile

Get Metadata

Set variable

If Condition

Condition_to_check

True

DS_Blob_Filename

General Settings User properties

Dataset * DS_Blob_Filename Open New Learn more

Dataset properties

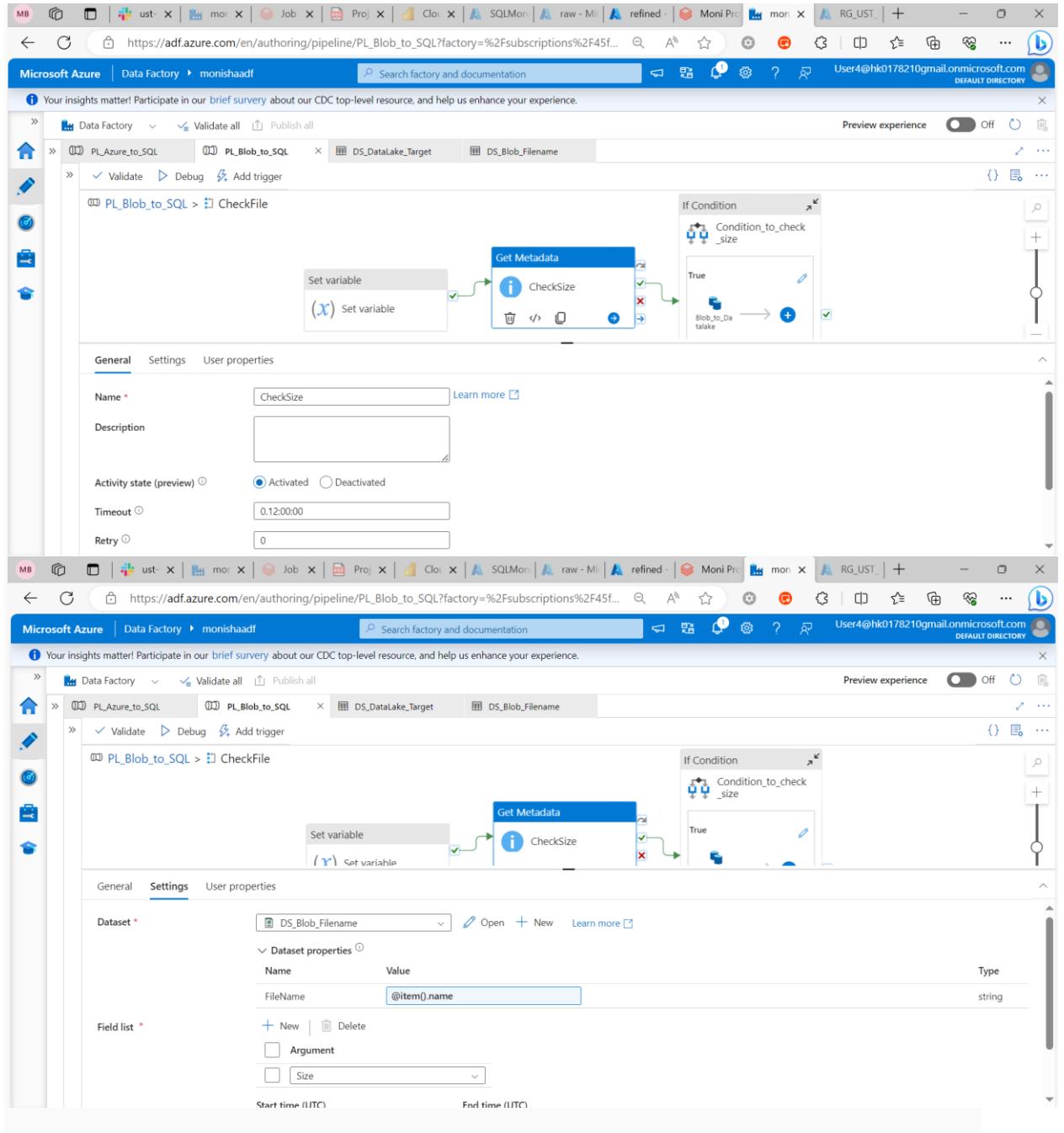
Name	Value	Type
FileName	@item().name	string

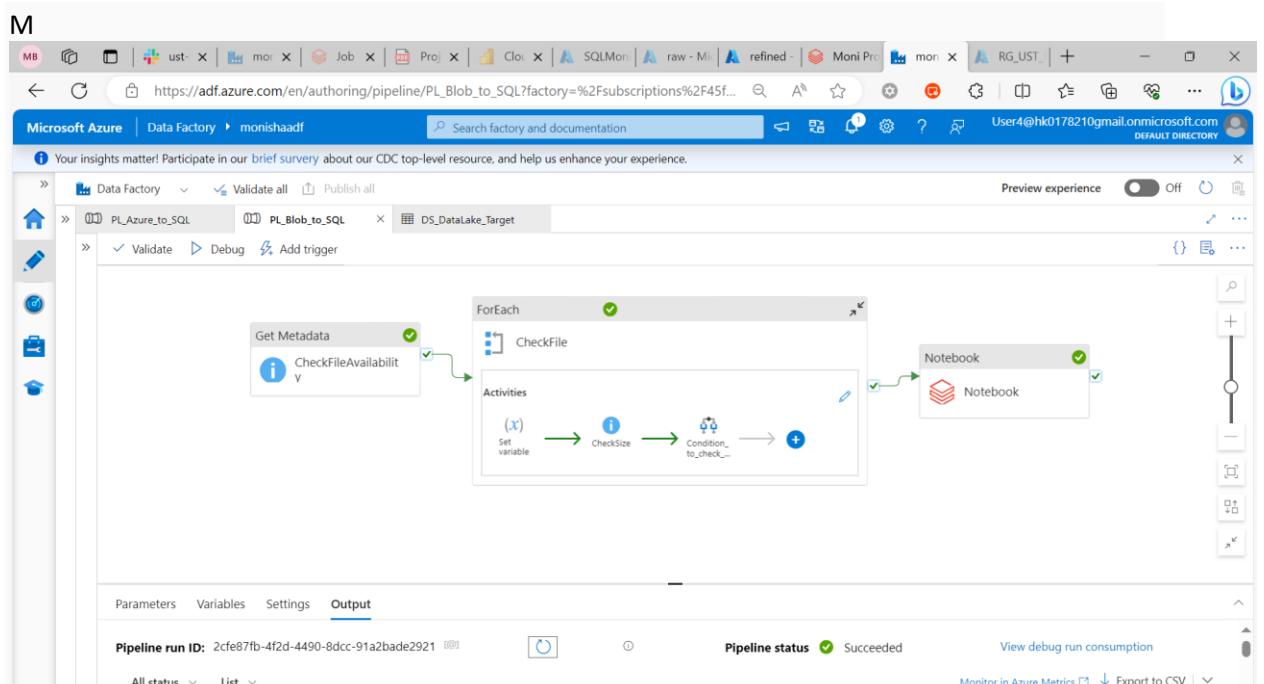
Field list

New Delete Argument Size

Start time (UTC)

End time (UTC)





IN DATAFACTORY → MANAGE → ARM TEMPLATE → EXPORT

Resource group level also we can take backup

AFTER THAT EXTRACT → BUILD UR TEMPLATE → ARM → LOAD FILE → OPEN → SAVE

Custom deployment:

(IT will not save password or credentials we need to provide)

MANAGE → ARM TEMPLATE → EXPORT ARM TEMPLATE