



---

# PRODUCT SALES ANALYSIS

---



EMP ID:2320097

NAME: ADINA JOSHI SATYA VARDAN  
Cohort Id: CSDAIA24AZ003

## **Problem Statement:**

- Fetch data from source URL & load in raw layer as zip file.
- Fetch files that has been arrived in last 24 hrs from source URL.
- Unzip files into refined container which is staging layer and
- Processed or clean data & load into ADLS processed container in parquet file as delta table.

## **Requirements 1 - Data Preparation (using ADF):**

### **1. Fetching and Loading Raw Data:**

- The source data will be fetched from the provided URL.
- The fetched data will be loaded into the raw container in ADLS in zip file format.

### **2. Unzipping and Loading Data into Staging:**

- Unzip the files that have arrived in the last 24 hours from the source in raw container.
- Load the unzipped files into the ADLS refine container in the path:  
refine/<participant\_name>/Unzippeddata/file.

### **3. Processing and Loading Clean Data:**

- Process or clean the data from the unzipped files using Azure Databricks (ADB).
- Load the processed or clean data under the processed container using Azure Databricks in the form of parquet files.

## **Requirements 2 - Data Transformation (using ADB):**

### **1. Accessing Data Lake using Secret Keys:**

- Access the data lake using secret keys by creating secrets in Key Vault.

### **2. Adding Header to Raw File:**

- Add headers to the raw file "sourceorderheader" with the provided schema:

SalesOrderID,RevisionNumber,OrderDate,DueDate,ShipDate,Status,OnlineOrderFlag,SalesOrderNumber,PurchaseOrderNumber,AccountNumber,CustomerID,ShipToAddressID,BillToAddressID,ShipMethod,CreditCardApprovalCode,SubTotal,TaxAmt,Freight,TotalDue,Comment,rowguid,ModifiedDate.

### **3. Joining and Cleaning Data:**

- Join the "product" file and "salesorderdetails" file.
- Perform necessary cleaning:
  - Drop all entries where color is null.

- Drop all entries where size is null.
- Check the difference between StandardCost and ListPrice for each sales order.
- Replace product sizes: S -> 30, M -> 32, L -> 34, XL -> 36, XXL -> 38 to display available product groups by size.

#### 4. Creating Managed Delta Tables:

- Create managed delta tables in Azure Databricks for performing analysis using SQL queries.

#### 5. Creating External Delta Tables:

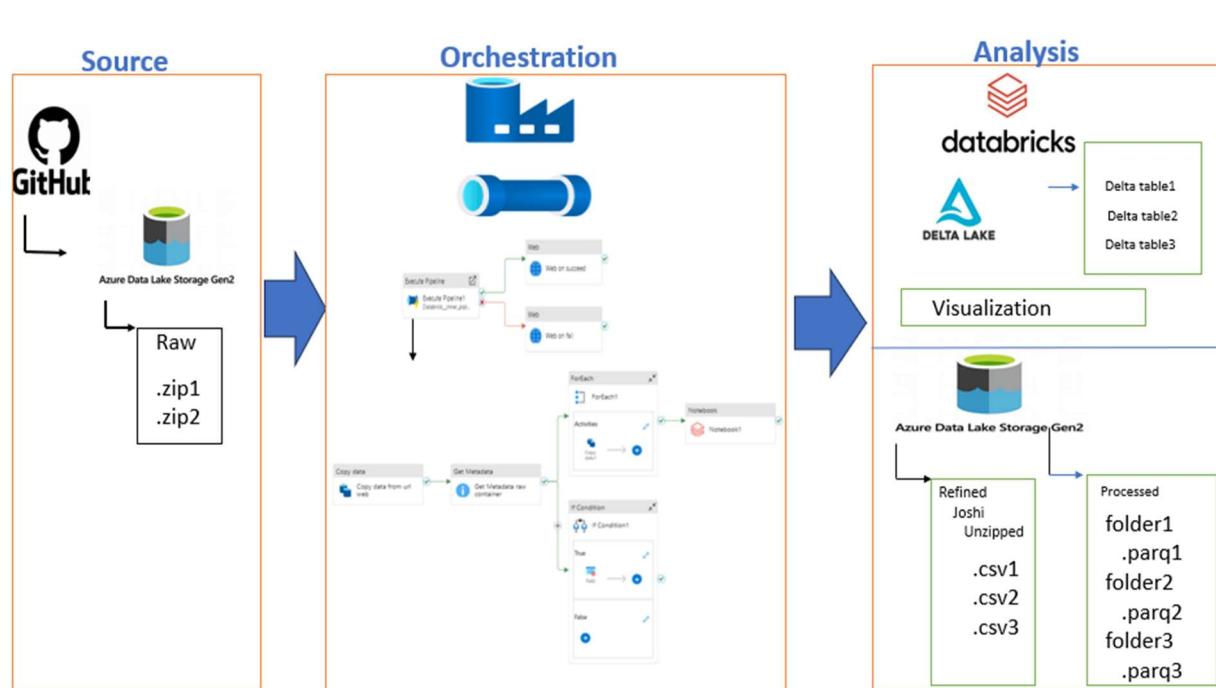
- Create external delta tables in Azure Databricks and load data in the form of parquet files into the processed data lake container.

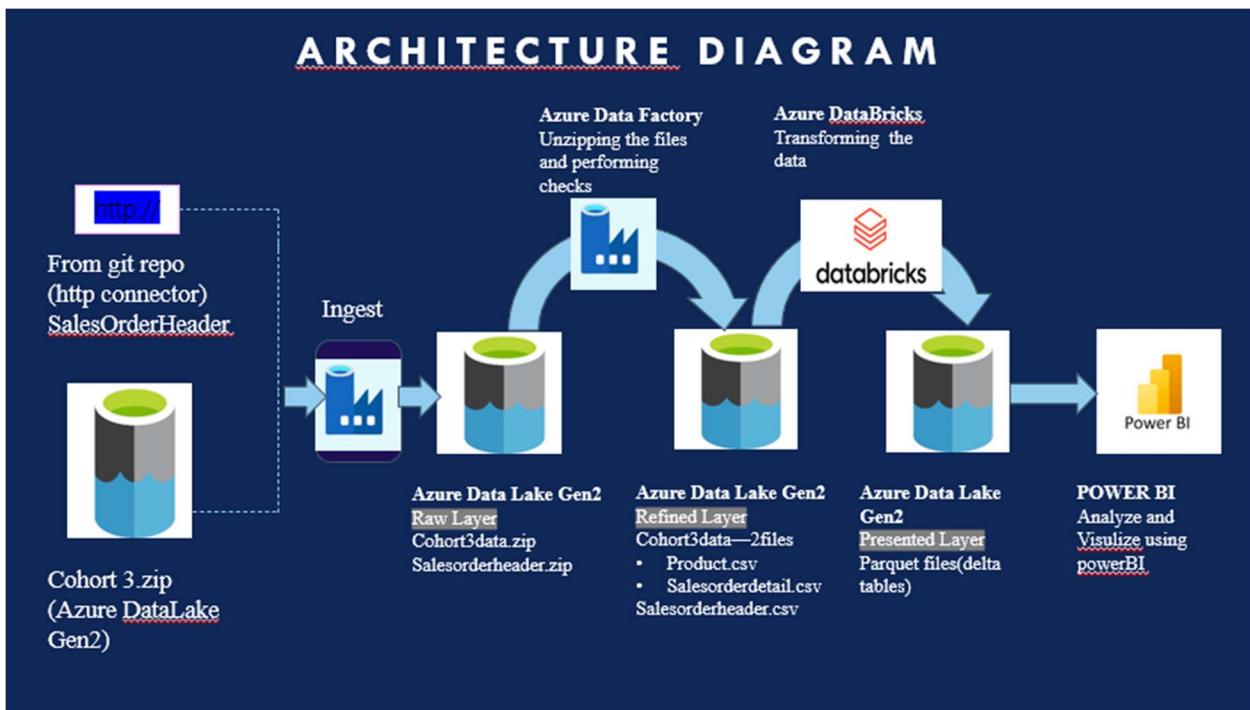
#### 6. Visualization and Analysis:

- Create visualizations on the cleaned data and generate graphs to understand product order sales details easily.

## Setup Email Configuration:

- Configure email notifications for pipeline failure and success.
- The email should contain pipeline name, run ID, and failure/success message.





Microsoft Azure

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

19891A03Q6@srkrec.ac.in SRKR ENGINEERING COLLEGE

**databricks\_resource\_group** Resource group

Overview Resources Recommendations

Filter for any field... Type equals all Location equals all Add filter

Showing 1 to 7 of 7 records. Show hidden types No grouping List view

Name	Type	Location
AzureDatabricksVaults	Key vault	East US
databricks_workspace	Azure Databricks Service	East US
DatabricksLogicApp	Logic app	East US
databrickspratice	Storage account	East US
DatafactoryDatabrick452	Data factory (V2)	East US
outlook	API Connection	East US
outlook-1	API Connection	East US

< Previous Page 1 of 1 Next >

Give feedback

Cloud ENG IN 11:56 PM 4/5/2024

Step 1: create resource group “databricks\_resource\_group” which is used to hold resources.

The screenshot shows the Microsoft Azure Storage account interface for 'databrickspratice'. The left sidebar includes options like Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, and Storage browser. Under Data storage, 'Containers' is selected. The main area displays a table of containers:

Name	Last modified	Anonymous access level	Lease state
\$logs	31/03/2024, 11:33:22	Private	Available
processed	05/04/2024, 18:42:43	Private	Available
raw	31/03/2024, 12:28:45	Private	Available
refine	03/04/2024, 17:29:48	Private	Available

Step 2: Create storage account (Azure datalake ) with name “databrickspratice” for hierarchical namespace for storing raw data ,refined data, processed data in different folders inside containers.

The screenshot shows the Microsoft Azure Databricks workspace interface for 'databricks\_workspace'. The left sidebar includes Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Virtual Network Peerings, Encryption, Networking, Security & compliance, Properties, Locks), and Monitoring. The main area displays workspace details:

- Resource group: databricks\_resource\_group
- Location: East US
- Subscription: Azure for Students
- Subscription ID: 264b3f1e-91c0-4e91-a1db-7f81c7e72a39
- Tags: (edit) Add.tags

A large red Databricks logo is centered, and a blue 'Launch Workspace' button is at the bottom right. The URL https://adb-5869407656176356.16.azuredatabricks.net is also shown.

Step 3:Create data bricks workspace with name “databricks\_workspace”.

The screenshot shows the Microsoft Azure Logic App interface for a resource named "DatabricksLogicApp". The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Development Tools (Logic app designer, Logic app code view, Run History, Versions, API connections, Quick start guides), and Settings. The main content area displays the "Essentials" section with details such as Resource group (move to "databricks\_resource\_group"), Location (East US), Subscription (Azure for Students), Subscription ID (264b3f1e-91c0-4e91-a1db-7f81c7e72a39), Workflow URL (https://prod-94.eastus.logic.azure.com:443/workflows/fcb7f7b02677...), and Tags (edit, Add tags). Below this are tabs for Get started, Runs history (selected), Trigger history, and Metrics. The top navigation bar includes a search bar, a menu icon, and account information (19B91A03Q6@srkrec.ac.in, SRKR ENGINEERING COLLEGE (S...)). The bottom taskbar shows various icons for Windows 10.

Step 4: Create logic app with name “DatabricksLogicApp”.

The screenshot shows the Microsoft Azure Key Vault interface for a resource named "AzureDatabricksVaults". The left sidebar includes Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Access policies, Events, Objects (Keys, Secrets selected), Certificates, Settings, and Access configuration. The main content area displays a table of secrets with columns Name, Type, Status, and Expiration date. The secrets listed are: DatalakeAccessKey (Enabled), DatalakeSASToken (Enabled), datalakeServicePrinClientID (Enabled), datalakeServicePrinClientScrete (Enabled), and datalakeServicePrinTenantID (Enabled). The top navigation bar includes a search bar, a menu icon, and account information (19B91A03Q6@srkrec.ac.in, SRKR ENGINEERING COLLEGE (S...)). The bottom taskbar shows various icons for Windows 10.

Step 5: Create Azure key vault with name “AzureDatabricksVaults” to store secrets in key valuts for project .

The screenshot shows the Microsoft Azure Data Factory Studio interface. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, the URL 'Home > databricks\_resource\_group > DatafactoryDatabrick452' is visible. The main content area displays the 'DatafactoryDatabrick452' resource group details, including its type as 'Data factory (V2)', status as 'Succeeded', location as 'East US', and subscription information. A large blue icon of a factory building is centered below the details. The bottom of the screen shows a taskbar with various icons and the system tray indicating the date and time as 4/6/2024.

Step 6: Create data factory with name "DatafactoryDatabricks452" from creating pipelines and scheduling pipelines.

## Related to Azure DataFactory:

The screenshot shows the 'Linked services' configuration page within the Microsoft Azure Data Factory studio. The left sidebar lists various settings like General, Factory settings, Connections, and Linked services. The 'Linked services' section is currently selected. It shows a list of three linked services: 'AzureDataLakeStorage1' (Type: Azure Data Lake Storage Gen2), 'HttpServer1' (Type: HTTP), and 'Ls\_AzureDatabricks' (Type: Azure Databricks). The bottom of the screen shows a taskbar with various icons and the system tray indicating the date and time as 4/6/2024.

Step 7: Create Linked Services for data factory to external services provided in azure like data lake (**AzureDataLakeStorage**), data bricks (**LS\_AzureDatabricks**) and http server (**HTTPSServer1**).

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar has 'General', 'Connections', and 'Linked services' selected. The main area shows a list of 'Linked services' with three items: 'AzureDataLakeStorage1' (Type: Azure Data Lake Storage Gen2), 'HttpServer1' (Type: HTTP), and 'Ls\_AzureDatabricks' (Type: Azure Databricks). On the right, a modal window titled 'Edit linked service' is open for 'Azure Data Lake Storage Gen2'. It includes fields for 'Name' (set to 'AzureDataLakeStorage1'), 'Description', 'Connect via integration runtime' (set to 'AutoResolveIntegrationRuntime'), 'Authentication type' (set to 'Account key'), 'Account selection method' (radio button for 'Enter manually' selected), 'URL' (set to 'https://databrickspratice.dfs.core.windows.net/'), 'Storage account key' (selected), 'Storage account key \*' (set to '\*\*\*\*\*'), and a 'Test connection' button which is currently 'Test connection'.

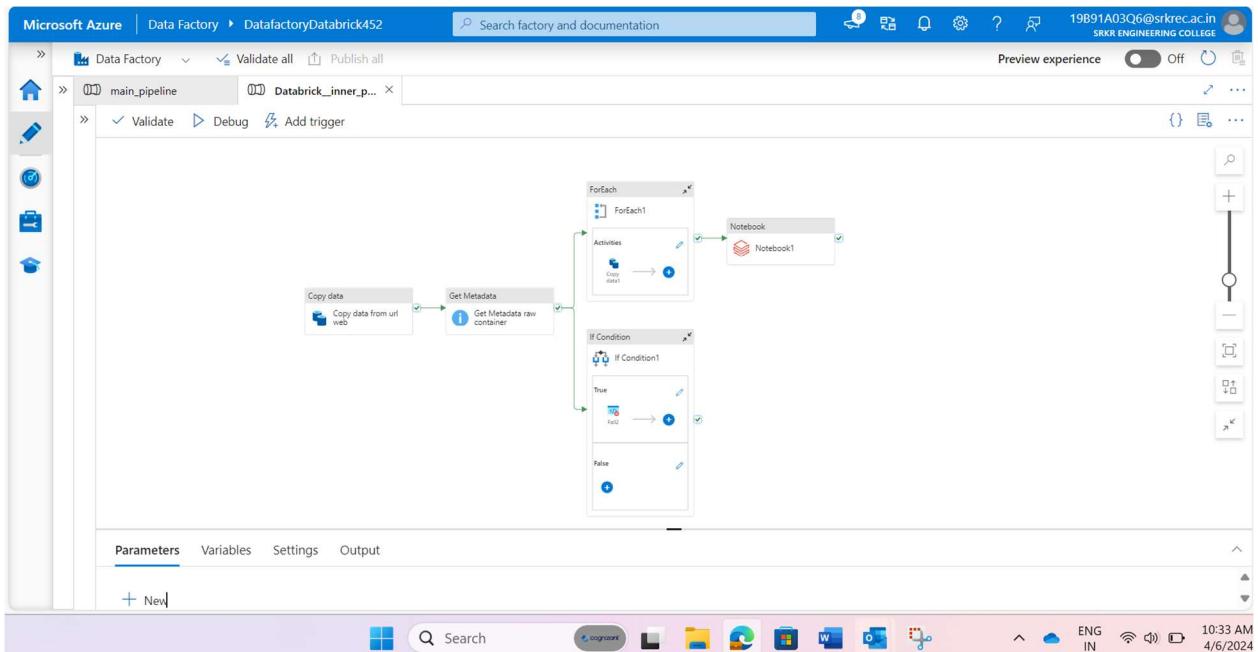
Step 8: Create Linked service between data lake and data factory with name “**AzureDataLakeStorage**”.

The screenshot shows the Microsoft Azure Data Factory interface. The left sidebar has 'General', 'Connections', and 'Linked services' selected. The main area shows a list of 'Linked services' with three items: 'AzureDataLakeStorage1' (Type: Azure Data Lake Storage Gen2), 'HttpServer1' (Type: HTTP), and 'Ls\_AzureDatabricks' (Type: Azure Databricks). On the right, a modal window titled 'Edit linked service' is open for 'HTTP'. It includes fields for 'Name' (set to 'HttpServer1'), 'Description', 'Connect via integration runtime' (set to 'AutoResolveIntegrationRuntime'), 'Base URL' (set to 'https://raw.githubusercontent.com/microsoft/sql-server-samples/master/samples/databases/'), a warning message about trusting the URL, 'Server certificate validation' (radio button for 'Enable' selected), 'Authentication type' (set to 'Anonymous'), 'Auth headers' (empty), and a 'Save' button which is currently 'Save'.

**Step 9: Creating Linked Service between external data source to data factory with name “HTTPSServer1”.**

The screenshot shows the Microsoft Azure Data Factory interface. On the left sidebar, 'Linked services' is selected. In the main area, a table lists three existing linked services: 'AzureDataLakeStorage1' (Type: Azure Data Lake Storage Gen2), 'HttpServer1' (Type: HTTP), and 'Ls\_AzureDatabricks' (Type: Azure Databricks). To the right, the 'Edit linked service' dialog is open for 'HttpServer1'. It shows the configuration for connecting via an integration runtime ('AutoResolveIntegrationRuntime'), selecting an account manually ('From Azure subscription' is selected), specifying a Databricks workspace URL ('https://adb-5869407656176356.16.azuredatabricks.net'), choosing 'Access Token' authentication type, and entering an access token ('.....'). A green checkmark indicates a 'Connection successful' test was run. The status bar at the bottom shows the date and time as 4/6/2024, 10:45 AM.

**Step 10: Create Linked Service between data bricks and data factory with name “LS\_AzureDatabricks”.**



Above picture will show inner pipeline designed for data ingestion ,data transformation and data loading.

The screenshot shows the Microsoft Azure Data Factory interface. A dataset named "DS\_adls\_raw\_zip\_web" is selected. The "Connection" tab is active, showing the following configuration:

- Linked service:** HttpServer1 (selected)
- Base URL:** https://raw.githubusercontent.com/micro...
- Relative URL:** SalesOrderHeader.csv
- Compression type:** Select...
- Column delimiter:** Comma (,)
- Row delimiter:** Default (\r\n or \n\r)
- Encoding:** Default(UTF-8)

The "Test connection" button shows a green checkmark indicating success.

Step 11: Create source dataset for fetching data from https URL .

The screenshot shows the Microsoft Azure Data Factory interface with a pipeline step selected. The "Source" tab is active for the "Copy data" activity, showing the following configuration:

- Source dataset:** DS\_adls\_raw\_zip\_web
- Request method:** GET
- Additional headers:** (empty)
- Request body:** (empty)

The pipeline also includes a "Get Metadata" activity and an "If Condition" activity, which is part of a larger conditional logic block.

Step 12: given that “DS\_adls\_raw\_zip\_web” dataset for source side of copy data activity with name as “Copy data from URL web”.

The screenshot shows the Microsoft Azure Data Factory interface. In the top navigation bar, it says "Microsoft Azure | Data Factory > DatafactoryDatabrick452". The search bar contains "Search factory and documentation". On the right, there are user details "19B91A03Q6@srkrec.ac.in SRKR ENGINEERING COLLEGE" and a "Preview experience" toggle set to "Off". Below the navigation, there are tabs for "Data Factory", "Validate all", and "Publish all". The main workspace shows a dataset named "DS\_adls\_raw\_source\_cp\_url" which is a "DelimitedText" type with a "CSV" icon. The "Connection" tab is selected, showing a linked service "AzureDataLakeStorage1" which is connected successfully. The "File path" is set to "raw / Directory / File name". Other settings include "Compression type: ZipDeflate (zip)", "Column delimiter: Comma (,), Row delimiter: Default (\r\n or \n\r)", and "Encoding: Default(UTF-8)". The bottom status bar shows the date and time as "12:11 AM 4/6/2024".

Step 13: Create sink data set for storing data set into raw container in data lake in zip format by giving compression type as (ZIP).

The screenshot shows the Microsoft Azure Data Factory interface with a pipeline named "main\_pipeline". The pipeline editor displays a sequence of activities: "Copy data" (source: "DS\_adls\_raw\_zip\_file\_web", sink: "DS\_adls\_raw\_zip\_web"), "Get Metadata" (source: "Get Metadata raw container"), "Copy data1" (source: "Copy data1" output, sink: "Notebook1"), and an "If Condition" activity. The "Sink" tab is selected for the "Copy data" activity, showing the "Sink dataset" as "DS\_adls\_raw\_source\_cp\_url". The "Copy behavior" dropdown is set to "Select...". The bottom status bar shows the date and time as "12:10 AM 4/6/2024".

Step 14: given that “**DS\_adls\_raw\_source\_cp\_url**” dataset for sink side of copy data activity “**Copy data from URL web**”.

The screenshot shows the Microsoft Azure Data Factory interface. A dataset named "DS\_adls\_raw\_zip\_file\_exist" is selected. The "Connection" tab is active, showing the following configuration:

- Linked service:** AzureDataLakeStorage1 (selected)
- File path:** raw / Directory / File name
- Compression type:** Select...
- Column delimiter:** Comma (,)
- Row delimiter:** Default (\r\n, \n or \r\r\n)
- Encoding:** Default(UTF-8)
- Quote character:** Double quote (")

The "Preview experience" toggle is off. The status bar at the bottom right shows "12:11 AM 4/6/2024".

Step 15: Create dataset for get metadata activity to point out to raw container for fetching metadata about files and folders in that container.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A pipeline named "main\_pipeline" is displayed. The pipeline consists of the following steps:

- A "Copy data" activity (Copy data from url/web) is connected to a "Get Metadata" activity (Get Metadata raw container).
- The "Get Metadata" activity is connected to an "If Condition" activity (If Condition1).
- The "If Condition1" activity is connected to a "Copy data1" activity (Copy data from url/web), which is then connected to a "Notebook1" activity.

The "Get Metadata" activity has the following settings:

- Dataset:** DS\_adls\_raw\_zip\_file\_exist
- Field list:** Argument, Child items
- Start time (UTC):** @getPastTime(24,'hour')
- End time (UTC):** @utcnow()

The "If Condition1" activity has the following settings:

- Condition:** If Condition1

The "Copy data1" activity has the following settings:

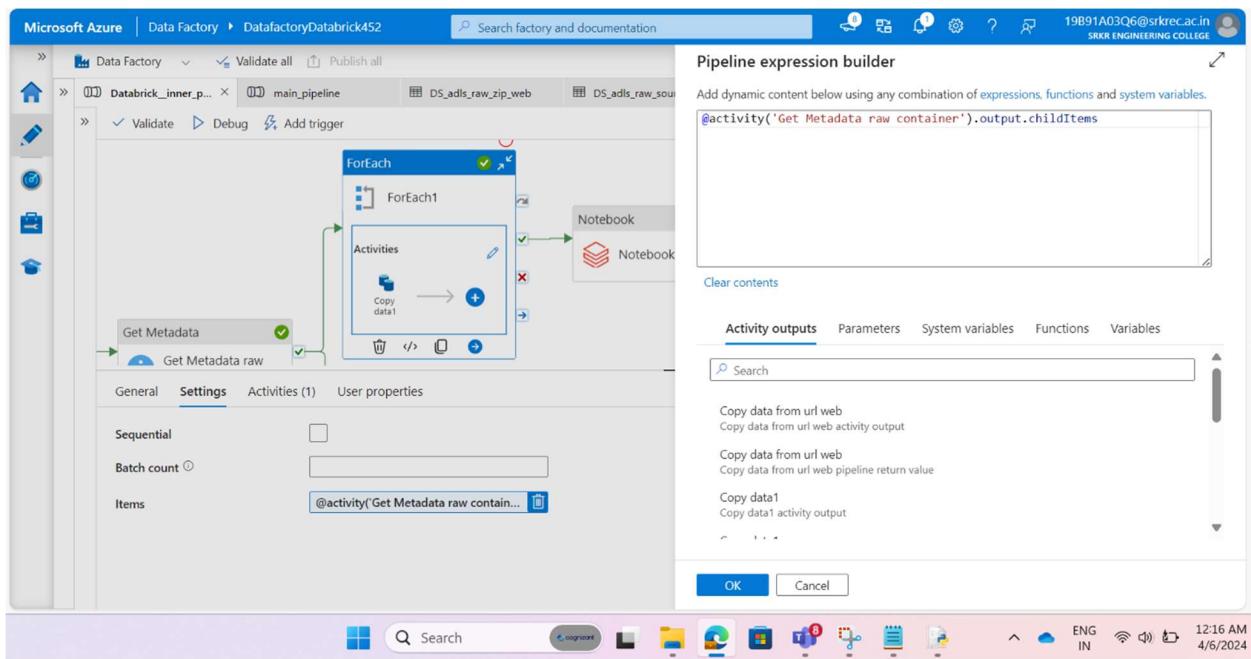
- Source:** DS\_adls\_raw\_zip\_file\_exist
- Sink:** DS\_adls\_raw\_source...

The "Notebook1" activity has the following settings:

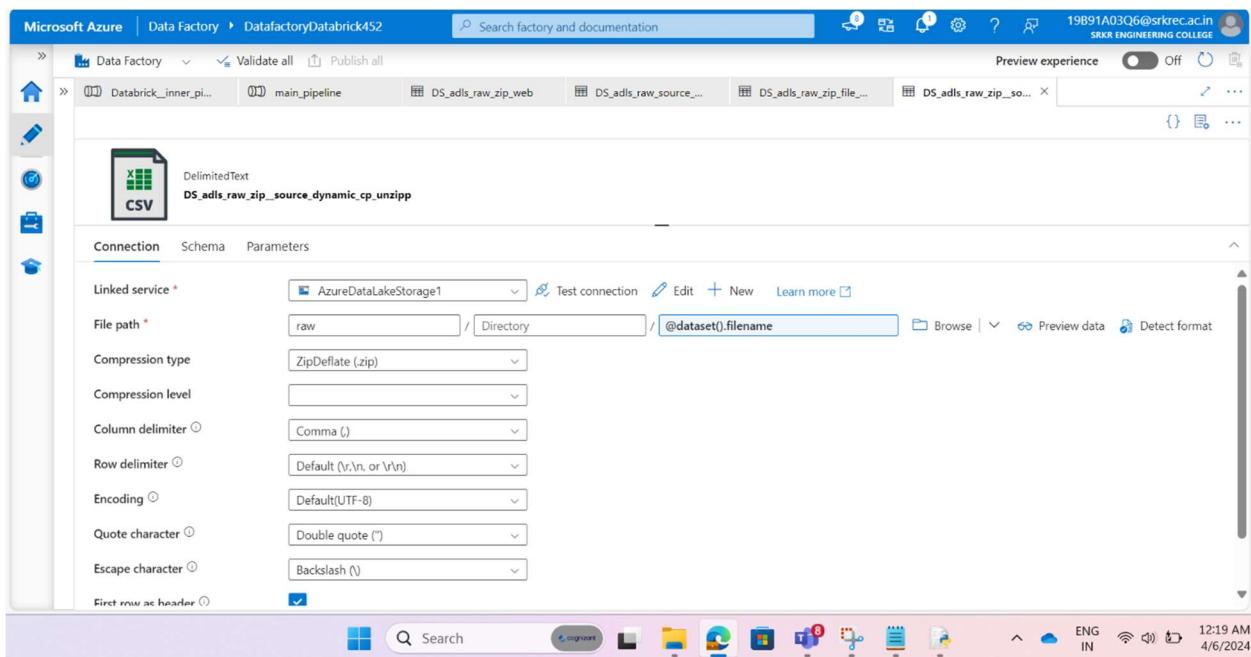
- Notebook:** Notebook1

The "Get Metadata" activity is highlighted with a red circle. The "Settings" tab is selected for the "Get Metadata" activity. The status bar at the bottom right shows "12:12 AM 4/6/2024".

Step 16 : Given that “DS\_adls\_raw\_zip\_file\_exist” dataset for picking childitems of files in raw containers within 24 hours from present time when pipeline start .



Step 17: Drag and drop for each activity for iterate over those child items pick by get metadata. Connect get metadata output with for each activity.



Step 18: Create source dataset for copy activity inside for each activity by passing file name dynamically by creating parameter with “filename” and passing value “@dataset().filename” at dataset level dynamically and giving compression type as .Zip .

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A 'Copy data' activity is selected, with its configuration pane open. The 'Source dataset' is set to 'DS\_adls\_raw\_zip\_source\_dynamically\_cp'. The 'File path type' is set to 'File path in dataset'. The 'Start time (UTC)' and 'End time (UTC)' fields are empty. The 'Filter by last modified' and 'Recursively' checkboxes are checked. The 'Enable partitions discovery' checkbox is unchecked. The pipeline name is 'main\_pipeline'.

Step 19: Given that “**DS\_adls\_raw\_zip\_source\_dynamically\_cp**” dataset at source side to copy data activity inside for each activity.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A 'DelimitedText' dataset is selected, with its configuration pane open. The 'Connection' tab is active. The 'Linked service' is set to 'AzureDataLakeStorage1'. The 'File path' is set to 'refine/joshi/unzipped/'. The 'Compression type' is set to 'None'. The 'Column delimiter' is set to 'Comma (,)'. The 'Row delimiter' is set to 'Default (\r,\n, or \v\n)'. The 'Encoding' is set to 'Default(UTF-8)'. The 'Quote character' is set to 'Double quote (")'. The 'Escape character' is set to 'Backslash (\)'. The 'First row as header' checkbox is checked. The 'Null value' field is empty. The pipeline name is 'main\_pipeline'.

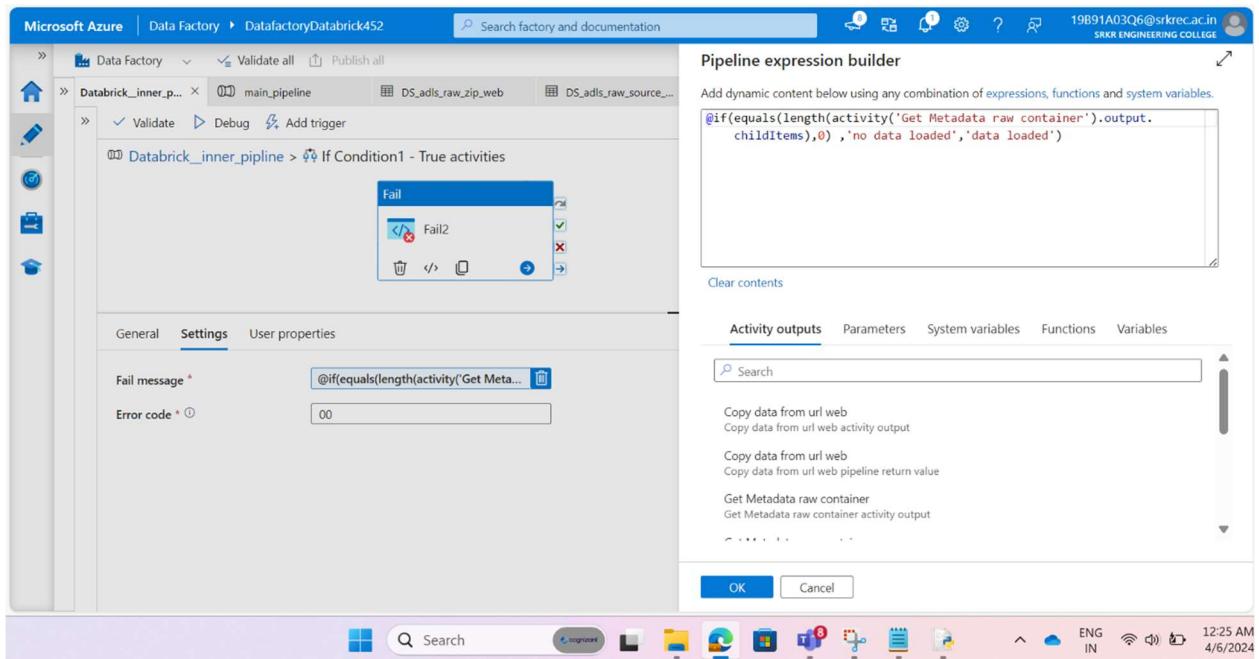
Step 20: Create dataset for sink side of copy data activity inside for each activity for copying zip file data from raw container to refine container inside folder name “**joshi/unzipped**” by unzipping files and keeping only **.csv files** inside refined container.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A 'Copy data' activity is selected, with its configuration pane open. The 'Sink' tab is active. The 'Sink dataset' dropdown is set to 'DS\_adls\_raw\_unzipped\_sink\_cp\_unzipped'. Other visible settings include 'Copy behavior' (set to 'Select...'), 'Max concurrent connections', 'Block size (MB)', 'Metadata', 'Quote all text', 'File extension (.csv)', and 'Max rows per file'.

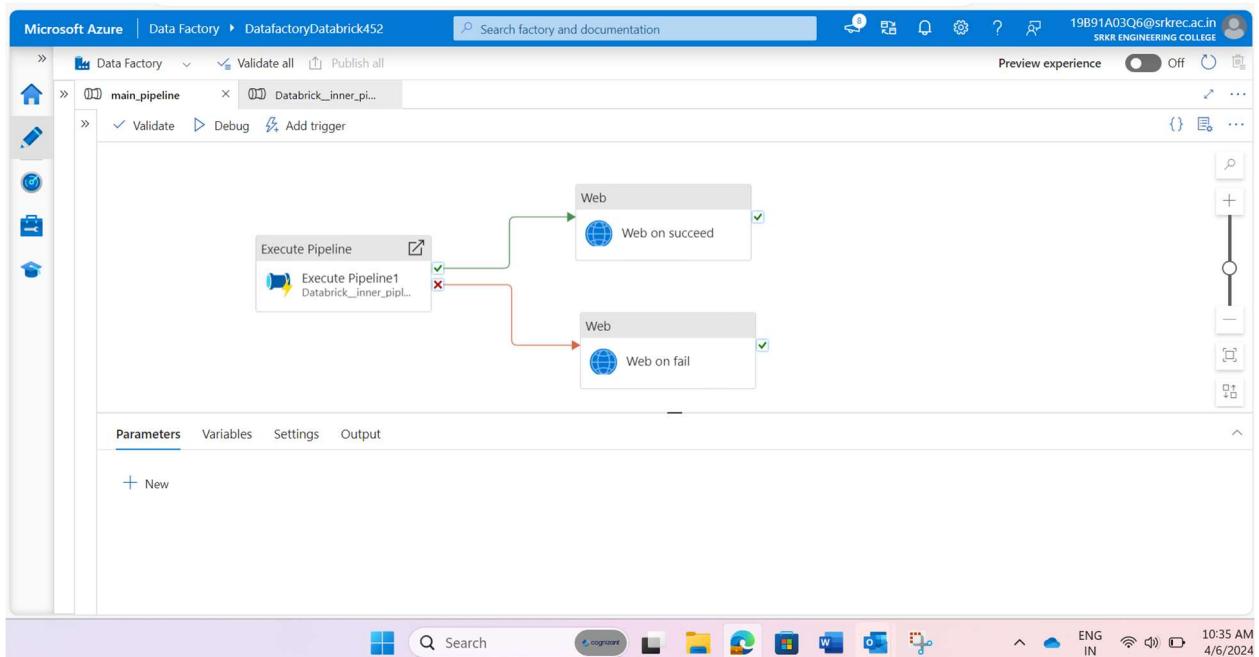
Step 21: given that “**DS\_adls\_raw\_unzipped\_sink\_cp**” dataset to sink side of copy data activity inside for each activity.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. An 'If Condition' activity is selected. The condition expression is set to '@equals(length(activity('Get Metadata raw container').output.childItems),0)'. The 'True' branch of the if condition leads to a 'Fail2' activity. The 'Activities (1)' tab in the left pane indicates one activity is present. The right pane displays the 'Pipeline expression builder' interface, which includes a search bar and a list of available activities like 'Copy data from url web' and 'Fail2'.

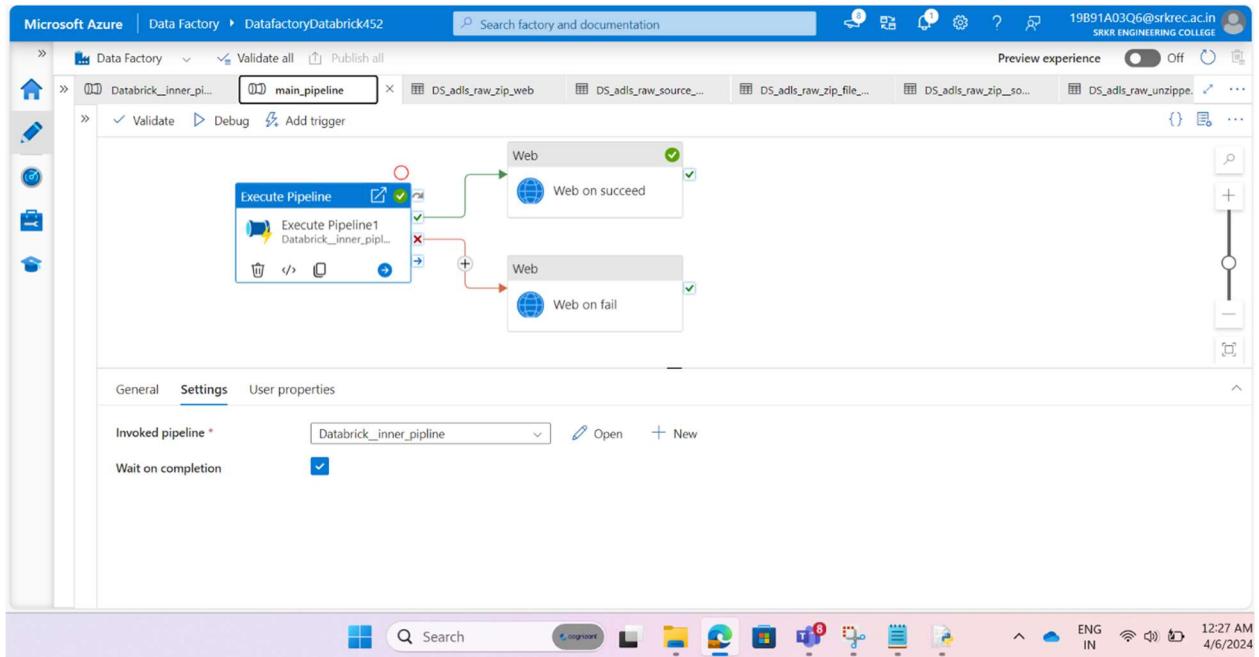
Step 22: Drag and drop if activity for checking child items length return by get metadata activity by giving condition like above it will check whether child item length is equals to zero or not.



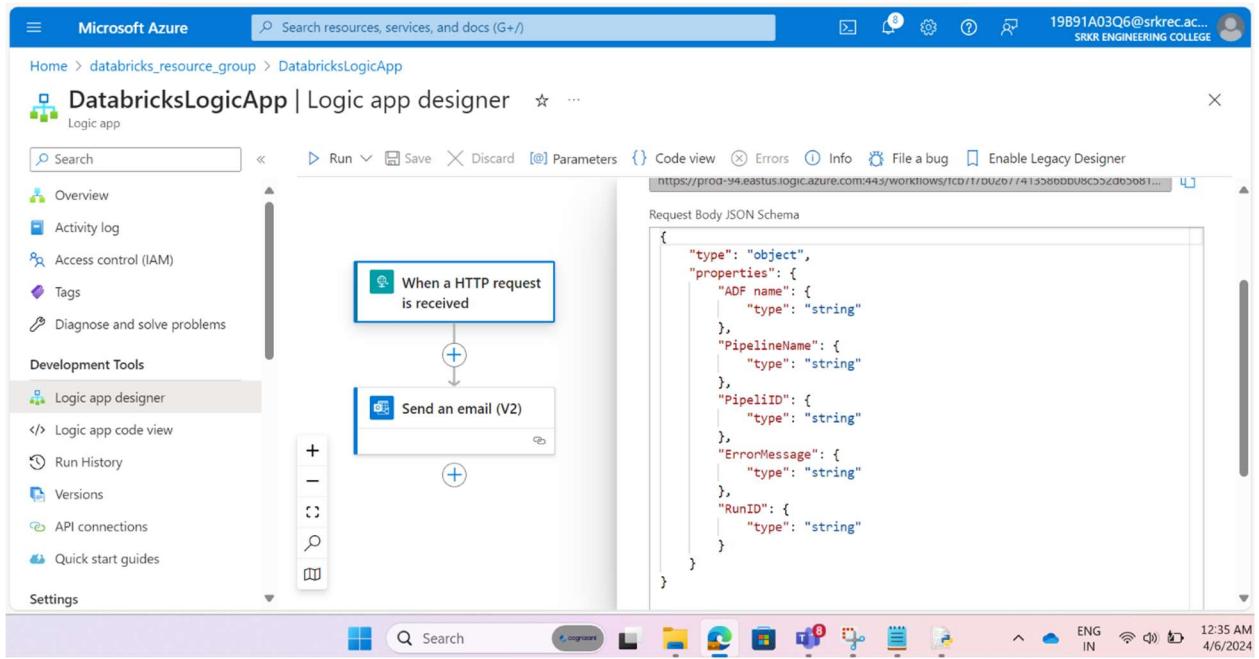
Step 23: Drag and drop Fail activity inside if condition while condition is true fail activity will work. Check for child item length whether zero or not depending upon that fail activity will fail pipeline when condition is true.



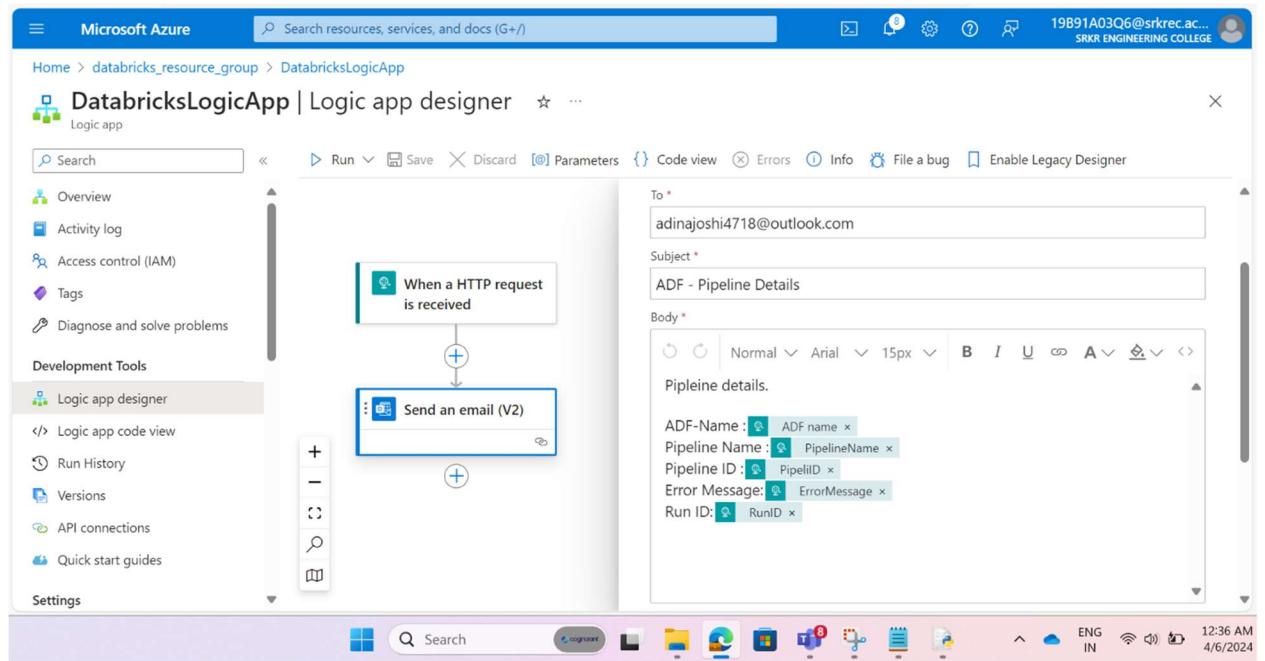
Step 24: Create another main pipeline with Execute pipeline activity, web activity.



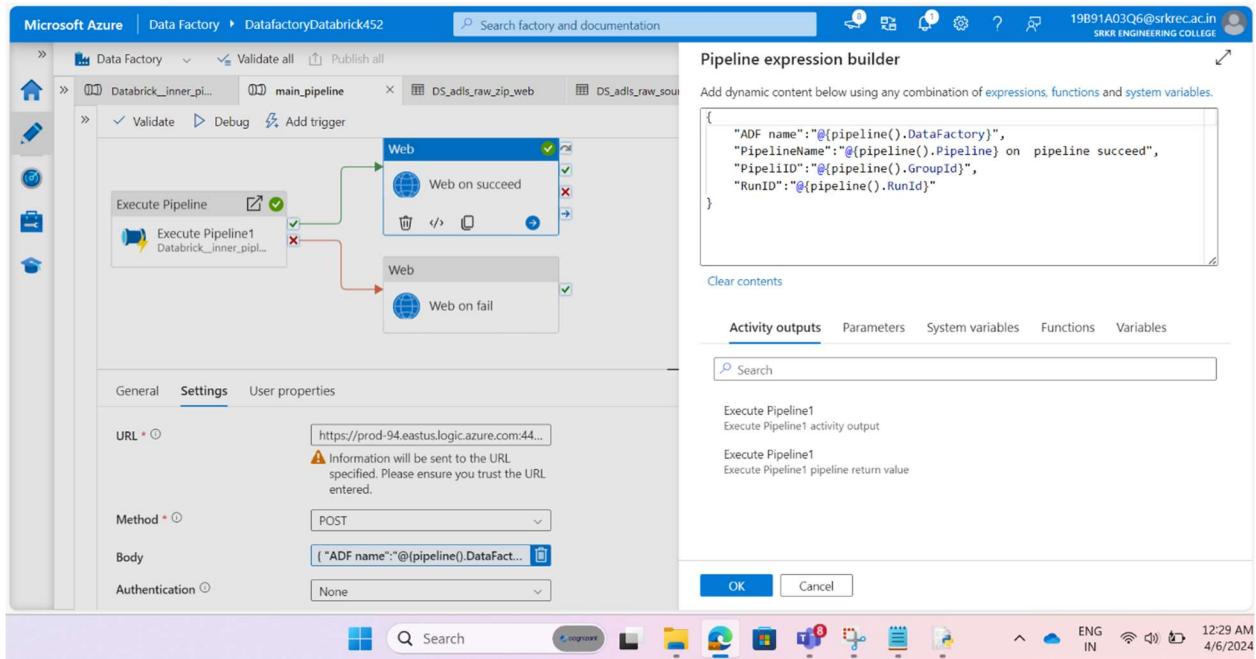
Step 25: given another pipeline to this activity which is nothing previous pipeline with name “Databrick\_inner\_pipeline”.



Step 26: Design log app for triggering mail when pipeline fail or on pipeline succeed by passing Json format which are to be appear in mail like pipeline id, ADF name, pipeline name, error message.



**Step 27: Design custom mail by giving personal mail id and craeting body with requiried details reagarding pipeline fails details.**



Step 28: given ADF name, pipeline name, pipeline id, Run id dynamically to https request to trigger mail in logic app on pipeline succeed.

The screenshot shows the Microsoft Azure Data Factory Pipeline expression builder interface. On the left, there's a navigation pane with icons for Home, Workspace, Repos, Favorites, and Trash. The main area displays a logic app flow with two parallel branches. The top branch, labeled 'Execute Pipeline', has a green checkmark and points to a 'Web' activity named 'Web on succeed'. The bottom branch, also labeled 'Execute Pipeline', has a red X and points to a 'Web' activity named 'Web on fail'. Below the logic app, there are tabs for General, Settings (which is selected), and User properties. Under the Settings tab, there are fields for URL (https://prod-94.eastus.logic.azure.com:44...), Method (POST), Body ({"ADF name": "@{pipeline().DataFact..."}), and Authentication (None). To the right of the logic app, a 'Pipeline expression builder' window is open, containing a JSON expression that dynamically constructs a URL for an https request based on pipeline information. At the bottom right of the pipeline builder window are 'OK' and 'Cancel' buttons.

Step 29: given ADF name, pipeline name, pipeline id, Run id, error message dynamically to https request to trigger mail in logic app on pipeline fail.

## Related to Data bricks:

The screenshot shows the Databricks workspace interface. On the left, there's a sidebar with icons for Home, Workspace, Repos, Favorites, and Trash. The main area shows a list of items under the user '19b91a03q6@srkrec.ac.in'. The list includes:

Name	Type	Owner	Created at
db_practice	Folder	VARADA VINAY	2024-03-31 12:08:57
product_sales_visualization	Dashboard	VARADA VINAY	2024-04-05 21:30:02
Project Notebook 2024-04-03 14:06...	Notebook	VARADA VINAY	2024-04-03 14:07:03

At the bottom, there's a toolbar with various icons for search, copy, paste, and other operations. The status bar at the bottom right shows the date and time (12:40 AM 4/6/2024).

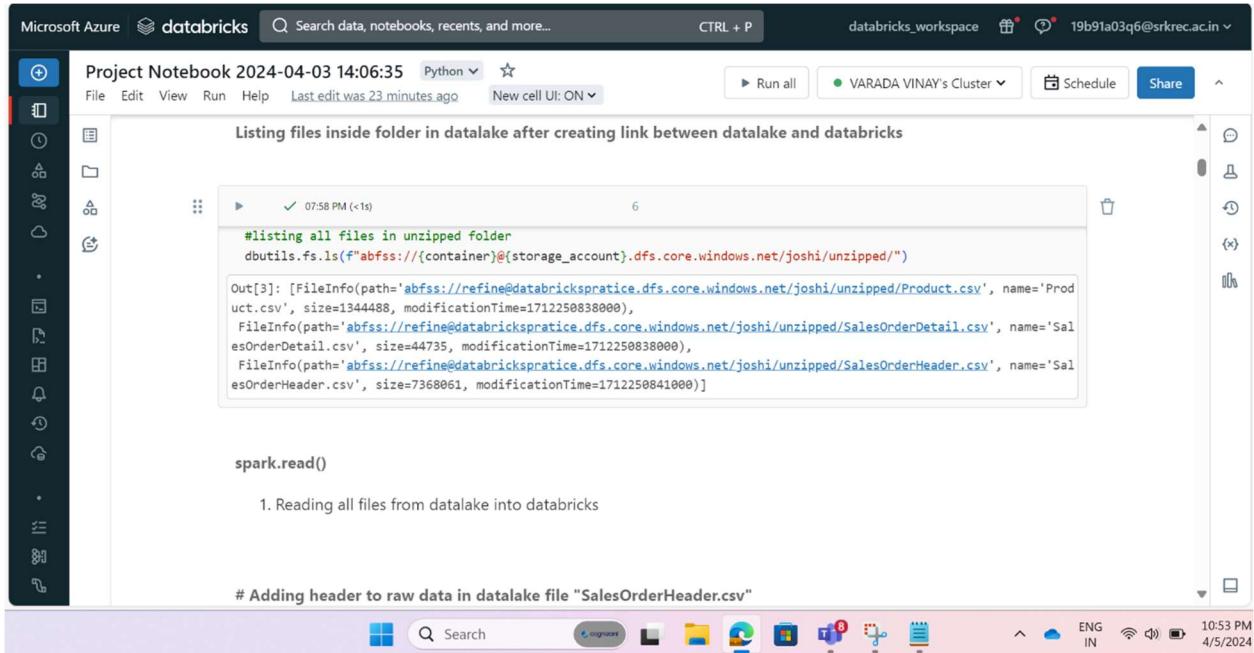
Step 30: Create workspace for doing data cleaning and data transformation on raw data.

The screenshot shows the Databricks Cluster Configuration page. The configuration is set to "Unrestricted". It includes options for "Multi node" and "Single node", "Access mode" (set to "Single user access"), and "User" (set to "VARADA VINAY"). The "Performance" section shows "Databricks Runtime Version" as "11.3 LTS (includes Apache Spark 3.3.0, Scala 2.12)", "Node type" as "Standard\_D3\_v2" (14 GB Memory, 4 Cores), and a checkbox for "Terminate after 20 minutes of inactivity" which is checked. The "Summary" panel indicates "1 Driver" with "14 GB Memory, 4 Cores", "Runtime" as "11.3.x-scala2.12", and "Standard\_D3\_v2" with "0.75 DBU/h". The status bar at the bottom shows "12:38 AM 4/6/2024".

Step 31: Create cluster with single node for running queries written in notebook and to give output.

The screenshot shows a Microsoft Azure Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" in Python. The notebook contains two code cells. The first cell, titled "Accessing Secret Keys", contains the following code:dbutils.secrets.help()  
dbutils.secrets.list(scope='datalake\_scope')  
dbutils.secrets.get(scope='datalake\_scope',key='DatalakeAccessKey')The second cell, titled "Creating link between datalake and databricks", contains the following code:#Creating link between datalake and databricks  
secret\_access\_key=dbutils.secrets.get(scope='datalake\_scope',key='DatalakeAccessKey')  
storage\_account="databrickspratice"  
container="refine"  
spark.conf.set("fs.azure.account.key.{storage\_account}.dfs.core.windows.net",secret\_access\_key)The status bar at the bottom shows "10:52 PM 4/5/2024".

Step 32: By using above code we can fetch secrets creating in data bricks and we can create link and authentication for data lake to data bricks.



Project Notebook 2024-04-03 14:06:35 Python

```
#listing all files in unzipped folder
dbutils.fs.ls(f"abfss://{container}@{storage_account}.dfs.core.windows.net/joshi/unzipped/")

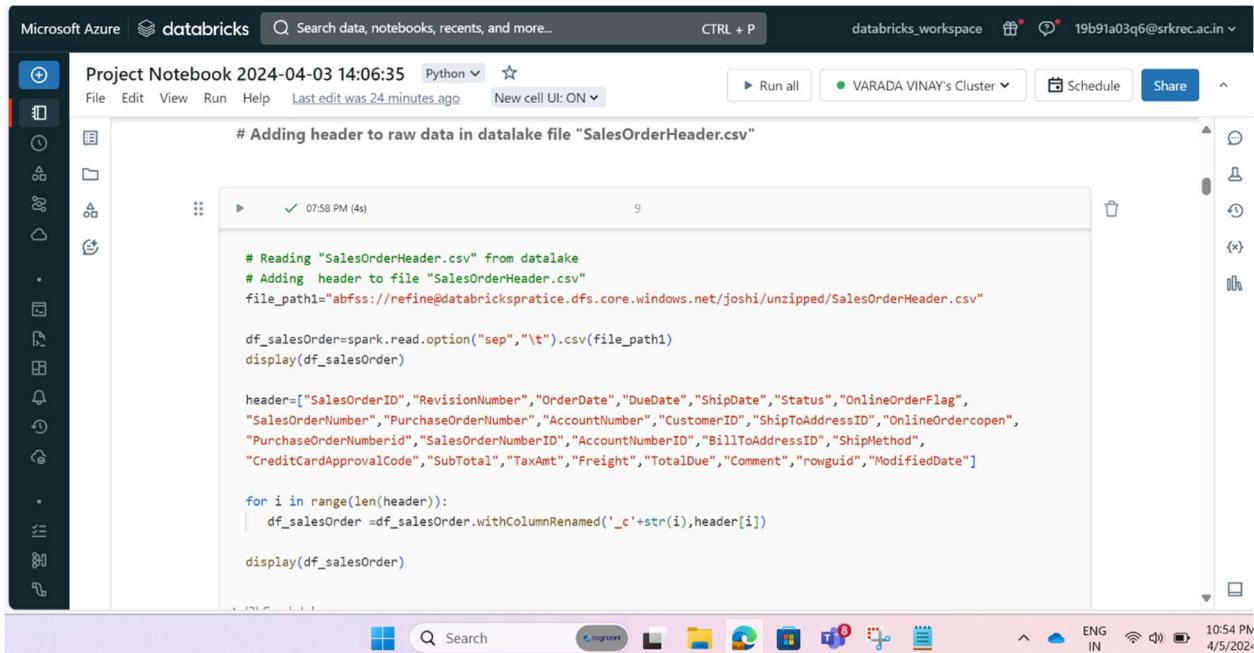
Out[3]: [FileInfo(path='abfss://refine@databrickspratice.dfs.core.windows.net/joshi/unzipped/Product.csv', name='Product.csv', size=1344488, modificationTime=1712250838000),
 FileInfo(path='abfss://refine@databrickspratice.dfs.core.windows.net/joshi/unzipped/SalesOrderDetail.csv', name='SalesOrderDetail.csv', size=44735, modificationTime=1712250838000),
 FileInfo(path='abfss://refine@databrickspratice.dfs.core.windows.net/joshi/unzipped/SalesOrderHeader.csv', name='SalesOrderHeader.csv', size=7368061, modificationTime=1712250841000)]
```

spark.read()

1. Reading all files from datalake into databricks

# Adding header to raw data in datalake file "SalesOrderHeader.csv"

Step 33: Above code is used to fetch files from data lake into data bricks to picks file for cleaning and transformation.



Project Notebook 2024-04-03 14:06:35 Python

```
# Adding header to raw data in datalake file "SalesOrderHeader.csv"

# Reading "SalesOrderHeader.csv" from datalake
# Adding header to file "SalesOrderHeader.csv"
file_path1="abfss://refine@databrickspratice.dfs.core.windows.net/joshi/unzipped/SalesOrderHeader.csv"

df_salesOrder=spark.read.option("sep","\t").csv(file_path1)
display(df_salesOrder)

header=["SalesOrderID","RevisionNumber","OrderDate","DueDate","ShipDate","Status","OnlineOrderFlag",
"SalesOrderNumber","PurchaseOrderNumber","AccountNumber","CustomerID","ShipToAddressID","OnlineOrderOpen",
"PurchaseOrderNumberId","SalesOrderNumberID","AccountNumberID","BillToAddressID","ShipMethod",
"CreditCardApprovalCode","SubTotal","TaxAmt","Freight","TotalDue","Comment","rowguid","ModifiedDate"]

for i in range(len(header)):
    df_salesOrder=df_salesOrder.withColumnRenamed('_c'+str(i),header[i])

display(df_salesOrder)
```

Step 34: Above code is used to read raw file from data lake and add header to that raw file.

The screenshot shows a Databricks Project Notebook interface. At the top, it displays "Project Notebook 2024-04-03 14:06:35" and "Python". Below the header are standard menu options: File, Edit, View, Run, Help, and a note that "Last edit was 25 minutes ago". To the right of the menu are buttons for "Run all", "Schedule", and "Share". The main workspace contains two data frames. The first data frame has columns labeled \_c0 through \_c5. The second data frame, titled "New result table: OFF", has columns labeled SalesOrderID, RevisionNumber, OrderDate, DueDate, ShipDate, and Status. Both data frames show 6,854 rows of data. The status bar at the bottom indicates "Refreshed 3 hours ago" and shows the date and time as 4/5/2024 10:55 PM.

Step 35: Above picture will show the output for above code to add header to raw file.

The screenshot shows a Databricks Project Notebook interface with two code cells. The top cell contains the following Python code:

```
#Reading "SalesOrderDetail.csv" file from datalake by using "spark.read.csv()" command
file_path2 ="abfss://refine@databrickspratice.dfs.core.windows.net/joshi/unzipped/SalesOrderDetail.csv"
df_SalesOrderDetail=spark.read.option('header','True').csv(file_path2)
#display(df_product)
```

The bottom cell contains the following Python code:

```
#Reading "Product.csv" file from datalake by using "spark.read.csv()" command
file_path2 ="abfss://refine@databrickspratice.dfs.core.windows.net/joshi/unzipped/Product.csv"
df_product=spark.read.option('header','True').csv(file_path2)
#display(df_product)
```

Both cells show a success message with a duration of less than 1 second. The status bar at the bottom indicates "10:56 PM 4/5/2024" and shows the date and time as 4/5/2024 10:56 PM.

Step 36: Above code is used for reading csv files from datalake into data bricks.

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks.workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python Last edit was 28 minutes ago New cell UI: ON

Join()

1. i need to join "df\_product" with "df\_SalesOrderDetail" using "productID"
2. new\_df=lefttable\_df.join(righttable\_df,common column,join type)

```
#using join() function by joining "product" and "SalesOrderDetail" data
df_pjsod = df_product.join(df_SalesOrderDetail,df_product.ProductID == df_SalesOrderDetail.ProductID,"inner")
display(df_pjsod)
```

(2) Spark Jobs

df\_pjsod: pyspark.sql.dataframe.DataFrame = [ProductID: string, Name: string ... 24 more fields]

Table New result table: OFF

ProductID	Name	ProductNumber	Color	StandardCost	ListPrice
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99

10:57 PM 4/5/2024

Step 37: Above code is used for apply join between to datasets “product” and “salesorder details” files.

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks.workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python Last edit was 30 minutes ago New cell UI: ON

(2) Spark Jobs

df\_pjsod: pyspark.sql.dataframe.DataFrame = [ProductID: string, Name: string ... 24 more fields]

Table New result table: OFF

ProductID	Name	ProductNumber	Color	StandardCost	ListPrice
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99
707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99

10:59 PM 4/5/2024

Step 38: After applying join by above code the output appear is like this.

The screenshot shows a Microsoft Azure Databricks Project Notebook interface. The notebook title is "Project Notebook 2024-04-03 14:06:35" and the language is set to Python. The code cell contains the following Python code:

```
#Select required columns from dataframe "df_pjsod"
df_main_product_SalesOrderDetail=df_pjsod.select(df_product.ProductID,(df_pjsod.Name).alias("ProductName"),
df_pjsod.ProductNumber,df_pjsod.SalesOrderID,df_pjsod.Color,df_pjsod.Size,df_pjsod.StandardCost,df_pjsod.ListPrice)

display(df_main_product_SalesOrderDetail)
```

The output of the cell shows a table with 15 rows and 7 columns. The columns are: ProductID, ProductName, ProductNumber, SalesOrderID, Color, Size, and StandardCost. The first few rows of the table are:

	ProductID	ProductName	ProductNumber	SalesOrderID	Color	Size
1	707	Sport-100 Helmet, Red	HL-US09-R	71938	Red	NULL
2	707	Sport-100 Helmet, Red	HL-US09-R	71936	Red	NULL
3	707	Sport-100 Helmet, Red	HL-US09-R	71902	Red	NULL
4	707	Sport-100 Helmet, Red	HL-US09-R	71797	Red	NULL
5	707	Sport-100 Helmet, Red	HL-US09-R	71784	Red	NULL
6	707	Sport-100 Helmet, Red	HL-US09-R	71783	Red	NULL
7						

Step 39: Above code is to select required columns from that join table for our data cleaning and data transformation .

The screenshot shows the same Databricks Project Notebook interface. The code cell now contains the command `display(df_main_product_SalesOrderDetail)`. The output shows the same table as before, with 542 rows and 7 columns. The table header is identical to the one in the previous screenshot.

	ProductID	ProductName	ProductNumber	SalesOrderID	Color	Size
1	707	Sport-100 Helmet, Red	HL-US09-R	71938	Red	NULL
2	707	Sport-100 Helmet, Red	HL-US09-R	71936	Red	NULL
3	707	Sport-100 Helmet, Red	HL-US09-R	71902	Red	NULL
4	707	Sport-100 Helmet, Red	HL-US09-R	71797	Red	NULL
5	707	Sport-100 Helmet, Red	HL-US09-R	71784	Red	NULL
6	707	Sport-100 Helmet, Red	HL-US09-R	71783	Red	NULL
7						

Step 40: After selecting required columns by using above code we got this output.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" in Python. The notebook interface includes a sidebar with various icons, a top bar with navigation and search options, and a main workspace. In the workspace, a cell titled "Replace" contains the following code:

```
# Replace size value with required values

from pyspark.sql.functions import when
df_replace_size = df_main_product_SalesOrderDetail.withColumn("Size",
    when(df_main_product_SalesOrderDetail["Size"]=="S",
        "30")
    .when(df_main_product_SalesOrderDetail["Size"] == "M",
        "32")
    .when(df_main_product_SalesOrderDetail["Size"] == "L",
        "34")
    .when(df_main_product_SalesOrderDetail["Size"] == "XL",
        "36")
    .when(df_main_product_SalesOrderDetail["Size"] == "XXL",
        "38")
    .otherwise(
        df_main_product_SalesOrderDetail["Size"]))
```

Step 41: Above code is used for cleaning data by replace size values which are letter with a number.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" in Python. The notebook interface includes a sidebar with various icons, a top bar with navigation and search options, and a main workspace. In the workspace, a cell displays a table titled "(2) Spark Jobs" containing data from the "df\_main\_product\_SalesOrderDetail" DataFrame. The table has the following columns: ProductNumber, SalesOrderID, Color, Size, StandardCost, and ListPrice. The data shows multiple rows where the "Size" column contains letter codes like "M" and "L".

	ProductNumber	SalesOrderID	Color	Size	StandardCost	ListPrice
1	y, M	LJ-0192-M	Multi	M	38.4923	49.99
2	y, M	LJ-0192-M	Multi	M	38.4923	49.99
3	y, M	LJ-0192-M	Multi	M	38.4923	49.99
4	y, M	LJ-0192-M	Multi	M	38.4923	49.99
5	y, M	LJ-0192-M	Multi	M	38.4923	49.99
6	y, M	LJ-0192-M	Multi	M	38.4923	49.99
7	y, M	LJ-0192-M	Multi	M	38.4923	49.99

Step 42: picture is showing size column data before applying transformation.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" in Python. The notebook interface includes a sidebar with various icons, a search bar at the top, and a header with "databricks.workspace" and user information. The main area displays a table of data with columns: ProductNumber, SalesOrderID, Color, Size, StandardCost, and ListPrice. The data shows several rows of product information, including some with null values in the Color and Size columns. Below the table, a section titled "Filter()" contains two steps: 1. by using filter (filtering null values in color column) and 2. df\_filtered=old\_df.filter(old\_df["Color"].isNotNull()).

ProductNumber	SalesOrderID	Color	Size	StandardCost	ListPrice
32	LJ-0192-M	Multi	32	38.4923	49.99
33	LJ-0192-M	Multi	32	38.4923	49.99
34	LJ-0192-M	Multi	32	38.4923	49.99
35	LJ-0192-M	Multi	32	38.4923	49.99
36	LJ-0192-M	Multi	32	38.4923	49.99
37	LJ-0192-L	Multi	34	38.4923	49.99

Step 43: picture is showing size column data after applying transformation.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" in Python. The notebook interface includes a sidebar with various icons, a search bar at the top, and a header with "databricks.workspace" and user information. The main area displays a code cell containing Python code for filtering NULL values from the "Color" and "Size" columns. The code uses the filter() function with a condition where both columns are not equal to "NULL". Below the code cell, a section titled "Filter()" contains four steps: 1. by using filter (filtering null values in color column), 2. df\_filtered=old\_df.filter(old\_df["Color"].isNotNull()), 3. but we have null in string format so we use, and 4. df\_filtered=old\_df.filter((old\_df["Color"]!="NULL") & (old\_df["Size"]!="NULL")). The notebook also shows a table of data with the same columns as before, but with fewer rows due to the filtering applied in the code.

```
# filter NULL values in color and size column

df_filtered_data=df_replace_size.filter((df_replace_size["Color"]!="NULL") & (df_replace_size["Size"]!="NULL"))
display(df_filtered_data)
```

Step 44: Above code is used for filtering color,size column recorders whivh are NULL.

The screenshot shows a Databricks Project Notebook interface. The notebook title is "Project Notebook 2024-04-03 14:06:35" and the language is Python. The code cell contains:

```
df_replace_size: pyspark.sql.dataframe.DataFrame = [ProductID: string, ProductName: string ... 6 more fields]
```

The resulting DataFrame table view shows 542 rows. The columns are ProductNumber, SalesOrderID, Color, Size, StandardCost, and ListPrice. Many rows have null values in the Color and Size columns. The status bar at the bottom indicates "Refreshed 3 hours ago".

Step 45 : Above picture shows output before filtering null values.

The screenshot shows the same Databricks Project Notebook interface. The notebook title is "Project Notebook 2024-04-03 14:06:35" and the language is Python. The code cell contains:

```
df_filtered_data=df_replace_size.filter((df_replace_size["Color"]!="NULL") & (df_replace_size["Size"]!="NULL"))
display(df_filtered_data)
```

The resulting DataFrame table view shows 393 rows. The columns are Name, ProductNumber, SalesOrderID, Color, Size, StandardCost, and ListPrice. All rows now have non-null values in the Color and Size columns. The status bar at the bottom indicates "Refreshed 3 hours ago".

Step 46: Above picture shows output after filtering null values.

The screenshot shows a Microsoft Azure Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" running in Python. The notebook contains the following code:

```
#Renaming ProductName by removing color and size
from pyspark.sql.functions import split
df_ProdctRename = df_filtered_data.withColumn("ProductName",split(df_filtered_data["ProductName"],'[-]+')[0])
display(df_ProdctRename)
```

The notebook interface includes a sidebar with various icons, a toolbar at the top, and a status bar at the bottom indicating the date and time.

Step 47: Above code is to clean product name is useful manner by using split () function in pyspark.

The screenshot shows a Microsoft Azure Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" running in Python. The notebook displays the following table:

ProductID	ProductName	ProductNumber	SalesOrderID	Color	Size
193	885 HL Touring Frame - Yellow, 60	FR-T98Y-60	71796	Yellow	60
194	885 HL Touring Frame - Yellow, 60	FR-T98Y-60	71784	Yellow	60
195	885 HL Touring Frame - Yellow, 60	FR-T98Y-60	71782	Yellow	60
196	886 LL Touring Frame - Yellow, 62	FR-T67Y-62	71782	Yellow	62
197	889 HL Touring Frame - Yellow, 54	FR-T98Y-54	71898	Yellow	54
198	889 HL Touring Frame - Yellow, 54	FR-T98Y-54	71784	Yellow	54
199					

The notebook interface includes a sidebar with various icons, a toolbar at the top, and a status bar at the bottom indicating the date and time.

Step 48: Above picture shows output before applying split() function.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" running in Python. The notebook displays a table of data with columns: ProductID, ProductName, ProductNumber, SalesOrderID, Color, Size, and Standart. The data consists of 393 rows, all of which have "HL Touring Frame" as the ProductName. The table is titled "(2) Spark Jobs" and includes a note about "df\_ProdctRename". The status bar at the bottom indicates "Refreshed 3 hours ago".

Step 49: Above picture shows output after applying split() function.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" running in Python. The notebook contains the following code:

```
from pyspark.sql.types import IntegerType
df_ProdctRename.printSchema()

df_datatypecast = df_ProdctRename.withColumn("Size", df_ProdctRename["Size"].cast(IntegerType())) \
    .withColumn("StandardCost", df_ProdctRename["StandardCost"].cast(IntegerType())) \
    .withColumn("ListPrice", df_ProdctRename["ListPrice"].cast(IntegerType())) \
    .withColumn("ProductID", df_ProdctRename["ProductID"].cast(IntegerType())) \
    .withColumn("SalesOrderID", df_ProdctRename["SalesOrderID"].cast(IntegerType()))

df_datatypecast.printSchema()
```

The status bar at the bottom indicates "Refreshed 3 hours ago".

Step 50: above code is to type cat datatype from string to integers.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" in Python. The notebook interface includes a sidebar with various icons, a search bar at the top, and a main workspace with code cells and a sidebar on the right. The current cell (cell 23) displays the schema of a DataFrame named "df\_datatypecast". The schema is defined as follows:

```
root
 |-- ProductID: string (nullable = true)
 |-- ProductName: string (nullable = true)
 |-- ProductNumber: string (nullable = true)
 |-- SalesOrderID: string (nullable = true)
 |-- Color: string (nullable = true)
 |-- Size: string (nullable = true)
 |-- StandardCost: string (nullable = true)
 |-- ListPrice: string (nullable = true)

root
 |-- ProductID: integer (nullable = true)
 |-- ProductName: string (nullable = true)
 |-- ProductNumber: string (nullable = true)
 |-- SalesOrderID: integer (nullable = true)
 |-- Color: string (nullable = true)
 |-- Size: integer (nullable = true)
 |-- StandardCost: integer (nullable = true)
 |-- ListPrice: integer (nullable = true)
```

Step 51: result of above code after and before type casting schema.

The screenshot shows a Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35" in Python. The notebook interface includes a sidebar with various icons, a search bar at the top, and a main workspace with code cells and a sidebar on the right. The current cell (cell 25) displays the schema of a DataFrame named "df\_final\_data" after type casting. The schema is defined as follows:

```
--> ProductNumber: string (nullable = true)
--> SalesOrderID: integer (nullable = true)
--> Color: string (nullable = true)
--> Size: integer (nullable = true)
--> StandardCost: integer (nullable = true)
--> ListPrice: integer (nullable = true)
```

Below the schema, there is a comment: "Discount cost of product sold by diff between "ListPrice" and "StandardCost"

The code cell contains the following Python code:

```
# finding difference between "ListPrice" and "StandardCost"

from pyspark.sql.functions import col,round
df_final_data=df_datatypecast.withColumn("DiffPrice",round(col("ListPrice")-col("StandardCost"),2))
```

The final output of the cell is:

```
--> df_final_data: pyspark.sql.dataframe.DataFrame = [ProductID: string, ProductName: string ... 7 more fields]
```

Step 52: Find different between listprice and standardprice .

A screenshot of a Microsoft Azure Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35". The notebook is running Python. The code cell displays a DataFrame named "df\_final\_data" using the command `display(df_final_data)`. The DataFrame contains 393 rows and 8 columns: ProductNumber, SalesOrderID, Color, Size, StandardCost, ListPrice, and DiffPrice. The data shows multiple entries for ProductNumber LJ-0192-M, with various values for SalesOrderID, Color, Size, and Price.

ProductNumber	SalesOrderID	Color	Size	StandardCost	ListPrice	DiffPrice
1	LJ-0192-M	Multi	32	38	49	11
2	LJ-0192-M	Multi	32	38	49	11
3	LJ-0192-M	Multi	32	38	49	11
4	LJ-0192-M	Multi	32	38	49	11
5	LJ-0192-M	Multi	32	38	49	11
6	LJ-0192-M	Multi	32	38	49	11
7	LJ-0192-M	Multi	32	38	49	11

Step 53: Output for above code using to find difference between listprice and standardprice.

**Creating delta tables (manage tables) in data bricks in hive meta store which are stored in DBFS.**

A screenshot of a Microsoft Azure Databricks Project Notebook titled "Project Notebook 2024-04-03 14:06:35". The notebook is running Python. The code cell creates view tables for "df\_final\_data", "df\_salesOrder", and "df\_product" using the command `#creating viewtables`. It then saves these as managed tables in the Hive Metastore under the names "manage\_ProductSalesOrderDetails", "manage\_SalesOrderDetails", and "manage\_Product".

```
Creating view tables of "df_final_data", "df_salesOrder", "df_product"

#creating viewtables
df_salesOrder.createOrReplaceTempView("df_salesOrder")
df_final_data.createOrReplaceTempView("df_final_data")
df_product.createOrReplaceTempView("df_product")

Created Managed Tables "manage_ProductSalesOrderDetails", "manage_SalesOrderDetails", "manage_Product"
```

The notebook also includes a section for Spark Jobs, though no specific code is shown there.

Microsoft Azure | databricks | Search data, notebooks, recents, and more... | CTRL + P | databricks\_workspace | 19b91a03q6@srkrec.ac.in | + Add | Browse DBFS | VARADA VINAY's Cl... 14 GB, 4 Cores | Create

### Catalog Explorer

Type to filter

- hive\_metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails
- samples

default > **default.manage\_productdetails**

Overview Sample Data Details Permissions History

Created At	Fri Apr 05 17:05:17 UTC 2024
Last Access	UNKNOWN
Created By	Spark 3.3.0
Type	MANAGED
Location	dbfs:/user/hive/warehouse/manage_productdetails
Is Managed Location	true
Properties	delta.minReaderVersion=1 delta.minWriterVersion=2

Use with BI tools | Create

Search | Microsoft Edge | File Explorer | Task View | Start | Cloud | ENG IN | 12:45 AM | 4/6/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... | CTRL + P | databricks\_workspace | 19b91a03q6@srkrec.ac.in | + Add | Browse DBFS | VARADA VINAY's Cl... 14 GB, 4 Cores | Create

### Catalog Explorer

Type to filter

- hive\_metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails
- samples

default > **default.manage\_productsalesorderdetails**

Overview Sample Data Details Permissions History

Created At	Fri Apr 05 17:05:10 UTC 2024
Last Access	UNKNOWN
Created By	Spark 3.3.0
Type	MANAGED
Location	dbfs:/user/hive/warehouse/manage_productsalesorderdetails
Is Managed Location	true
Properties	delta.minReaderVersion=1 delta.minWriterVersion=2

Use with BI tools | Create

Search | Microsoft Edge | File Explorer | Task View | Start | Cloud | ENG IN | 12:47 AM | 4/6/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Catalog Explorer Send feedback

Type to filter

default > default.manage\_salesorderdetails

Overview Sample Data Details Permissions History

Created At Fri Apr 05 17:05:14 UTC 2024

Last Access UNKNOWN

Created By Spark 3.3.0

Type MANAGED

Location dbfs:/user/hive/warehouse/manage\_salesorderdetails

Is Managed Location true

Properties delta.minReaderVersion=1  
delta.minWriterVersion=2

+ Add Browse DBFS Use with BI tools Create

12:46 AM 4/6/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python New cell UI: ON

File Edit View Run Help Last edit was 59 minutes ago Run all Terminated Schedule Share

Describe table schema

```
%sql
show create table manage_SalesOrderDetails;
show create table manage_ProductSalesOrderDetails;
show create table manage_ProductDetails;
```

\_sqldf: pyspark.sql.dataframe.DataFrame = [createtab\_stmt: string]

Table + New result table: OFF

createtab\_stmt

```
CREATE TABLE spark_catalog.default.manage_product ( ProductID STRING, Name STRING, ProductNumber STRING, Color STRING, StandardCost STRING, ListPrice STRING, Size STRING, Weight STRING, ProductCategoryID STRING,
1 ProductModelID STRING, SellStartDate STRING, SellEndDate STRING, DiscontinuedDate STRING, ThumbNailPhoto STRING,
ThumbnailPhotoFileName STRING, rowguid STRING, ModifiedDate STRING) USING delta TBLPROPERTIES (
'delta.minReaderVersion' = '1', 'delta.mi...
```

1 row | 0.37 seconds runtime Refreshed 3 hours ago

11:28 PM 4/5/2024

## Creating delta tables (external tables) in data bricks and there results of parquet files in datalake.

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python New cell UI: ON

File Edit View Run Help Last edit was 59 minutes ago Run all Terminated Schedule Share

Creating external table "productdetails" load data into datalake into processed container with file name "productdetails"

```
spark.sql(''' CREATE TABLE if NOT EXISTS ProductDetails ( ProductID STRING, Name STRING, ProductNumber STRING, Color STRING, StandardCost STRING, ListPrice STRING, Size STRING, Weight STRING, ProductCategoryID STRING, ProductModelID STRING, SellStartDate STRING, SellEndDate STRING, DiscontinuedDate STRING, ThumbNailPhoto STRING, ThumbnailPhotoFileName STRING, rowguid STRING, ModifiedDate STRING)
USING DELTA LOCATION 'abfss://processed@databrickspratice.dfs.core.windows.net/productdetails' ''')
```

(4) Spark Jobs

Out[24]: DataFrame[]

Inserting records into external table "productdetails" from "df\_product" viewtable

11:29 PM 4/5/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python New cell UI: now

File Edit View Run Help Last edit was now Run all Terminated Schedule Share

(4) Spark Jobs

Out[24]: DataFrame[]

Inserting records into external table "productdetails" from "df\_product"

```
%sql
insert into ProductDetails
select * from df_product;

/*
%python
df_product.write.option("path","abfss://processed@databrickspratice.dfs.core.windows.net/productdetails").
saveAsTable("ProductDetails")
*/
```

Creating external table "FinalProductSalesOrderDetails" load data into datalake into processed container with file name "FinalProductSalesOrderDetails"

11:36 PM 4/5/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

### Catalog Explorer

Type to filter

- hive\_metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails
- samples

+ Add Browse DBFS Use with BI tools Create

default > default.productdetails

Overview Sample Data Details Permissions History

Filter columns...

Column	Type	Comment
ProductID	string	
Name	string	
ProductNumber	string	
Color	string	
StandardCost	string	
ListPrice	string	
Size	string	

About this table

Owner: Not set

Data source format: Delta

Last Updated: 13 hours ago

Size: 247.5KiB, 1 file

Comment: Add comment

Search ENG IN 11:46 AM 4/6/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

### Catalog Explorer

Type to filter

- hive\_metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails
- samples

+ Add Browse DBFS Use with BI tools Details Create

default > default.productdetails

Overview Sample Data Details Permissions History

Created At	Fri Apr 05 17:05:20 UTC 2024
Last Access	UNKNOWN
Created By	Spark 3.3.0
Type	EXTERNAL
Location	abfss://processed@databrickspratice.dfs.core.windows.net/productdetails
Properties	delta.minReaderVersion=1 delta.minWriterVersion=2

Search ENG IN 12:42 AM 4/6/2024

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

Home > databrickspratice | Containers >

**processed** Container

Search Overview Diagnose and solve problems Access Control (IAM) Settings Shared access tokens Manage ACL Access policy Properties Metadata

Authentication method: Access key (Switch to Microsoft Entra user account)  
Location: processed / productdetails

Search blobs by prefix (case-sensitive) Show deleted objects

Name	Modified	Access tier	Archive status	Blob type
[...]				
_delta_log				
part-00000-5b582ea4-bdc4-4c82-852e-0620e1789...	05/04/2024, 22:35:21	Hot (Inferred)		Block blob

19B91A03Q6@srkrec.ac.in SRKR ENGINEERING COLLEGE (S...

Search 11:49 AM 4/6/2024

Microsoft Azure databricks Search data, notebooks, recents, and more... CTRL + P databricks.workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python Last edit was 1 minute ago New cell UI: ON

Run all Terminated Schedule Share

Creating external table "FinalProductSalesOrderDetails" load data into datalake into processed container with file name "FinalProductSalesOrderDetails"

08:37 PM (3s) 37

```
spark.sql('''CREATE TABLE if NOT EXISTS FinalProductSalesOrderDetails ( ProductID STRING, ProductName STRING, ProductNumber STRING, SalesOrderID STRING, Color STRING, Size INT, StandardCost INT, ListPrice INT, DiffPrice INT) USING delta LOCATION 'abfss://processed@databrickspratice.dfs.core.windows.net/FinalProductSalesOrderDetails''' )
```

(4) Spark Jobs

Out[75]: DataFrame[1]

Inserting records into external table "FinalProductSalesOrderDetails" from "df\_final\_data"

11:36 PM 4/5/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python New cell UI: ON Run all Terminated Schedule Share

Inserting records into external table "FinalProductSalesOrderDetails" from "df\_final\_data"

```
%sql
insert into FinalProductSalesOrderDetails
select * from df_final_data;

/*
%python
df_final_data.write.option("path","abfss://processed@databrickspratice.dfs.core.windows.net/
FinalProductSalesOrderDetails").saveAsTable("FinalProductSalesOrderDetails")
*/
(6) Spark Jobs
_sqlid: pyspark.sql.DataFrame
    num_affected_rows: long
    num_inserted_rows: long
```

Table + New result table: OFF

Search ENG IN 11:38 PM 4/5/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Catalog Explorer Send feedback + Add Browse DBFS VARADA VINAY's Cl... 14 GB, 4 Cores

Type to filter

- hive metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails
- samples

default > default.finalproductsalesorderdetails

Overview Sample Data Details Permissions History

Filter columns...

Column	Type	Comment
ProductID	string	
ProductName	string	
ProductNumber	string	
SalesOrderID	string	
Color	string	
Size	int	
StandardCost	int	

About this table

Owner: Not set

Data source format: Delta

Last Updated: 13 hours ago

Size: 6.3KiB, 1 file

Comment: Add comment

Search ENG IN 11:51 AM 4/6/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTR + P databricks\_workspace 19b91a03q6@srkrec.ac.in

### Catalog Explorer

Type to filter

- hive metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails
- samples

default > **default.finalproductsalesorderdetails**

Overview Sample Data Details Permissions History

Created At	Fri Apr 05 17:05:24 UTC 2024
Last Access	UNKNOWN
Created By	Spark 3.3.0
Type	EXTERNAL
Location	abfss://processed@databrickspratice.dfs.core.windows.net/FinalProductSalesOrderDetails
Properties	delta.minReaderVersion=1 delta.minWriterVersion=2

+ Add Browse DBFS Create

Use with BI tools

12:44 AM 4/6/2024

Microsoft Azure | Search resources, services, and docs (G+/-) 19B91A03Q6@srkrec.ac.in SRKR ENGINEERING COLLEGE (S...

Home > databrickspratice | Containers >

### processed

Container

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Overview

Authentication method: Access key (Switch to Microsoft Entra user account)  
Location: processed / FinalProductSalesOrderDetails

Search blobs by prefix (case-sensitive)

Show deleted objects

Name	Modified	Access tier	Archive status	Blob type
[..]				
_delta_log				
part-00000-9f57290a-c604-49ca-8fe7-0ed56d02eb...	05/04/2024, 22:35:24	Hot (Inferred)		Block blob

11:52 AM 4/6/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python New cell UI: ON

File Edit View Run Help Last edit was 3 minutes ago

Run all Terminated Schedule Share

Creating external table "SalesOrderDetails" load data into datalake into processed container with file name "SalesOrderDetails"

```
spark.sql("""CREATE TABLE IF NOT EXISTS SalesOrderDetails (SalesOrderID STRING, RevisionNumber STRING, OrderDate STRING, DueDate STRING, ShipDate STRING, Status STRING, OnlineOrderFlag STRING, SalesOrderNumber STRING, PurchaseOrderNumber STRING, AccountNumber STRING, CustomerID STRING, ShipToAddressID STRING, OnlineOrderOpen STRING, PurchaseOrderNumberID STRING, SalesOrderNumberID STRING, AccountNumberID STRING, BillToAddressID STRING, ShipMethod STRING, CreditCardApprovalCode STRING, SubTotal STRING, TaxAmt STRING, Freight STRING, TotalDue STRING, Comment STRING, Rowguid STRING, ModifiedDate STRING) USING delta LOCATION 'abfss://processed@databrickspratice.dfs.core.windows.net/SalesOrderDetails' """)
```

(4) Spark Jobs

Out[87]: DataFrame[]

11:39 PM 4/5/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks\_workspace 19b91a03q6@srkrec.ac.in

Project Notebook 2024-04-03 14:06:35 Python New cell UI: ON

File Edit View Run Help Last edit was 4 minutes ago

Run all Terminated Schedule Share

Inserting records into external table "SalesOrderDetails" from "df\_salesOrder"

```
%sql
insert into SalesOrderDetails
select * from df_salesOrder;

/*
%python
df_salesOrder.write.option("path","abfss://processed@databrickspratice.dfs.core.windows.net/SalesOrderDetails")
.saveAsTable("SalesOrderDetails")
*/
```

(5) Spark Jobs

\_sqldf: pyspark.sql.DataFrame = [num\_affected\_rows: long, num\_inserted\_rows: long]

Table + New result table: OFF

	num_affected_rows	num_inserted_rows
1	31465	31465

11:40 PM 4/5/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... | CTRL + P | databricks\_workspace | 19b91a03q6@srkrec.ac.in

### Catalog Explorer

Type to filter

- hive\_metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails**
- samples

default > **default.salesorderdetails**

Overview Sample Data Details Permissions History

Filter columns...

Column	Type	Comment
SalesOrderID	string	(+)
RevisionNumber	string	(+)
OrderDate	string	(+)
DueDate	string	(+)
ShipDate	string	(+)
Status	string	(+)
OnlineOrderFlag	string	(+)

About this table

Owner: Not set [Edit](#)

Data source format: **Delta**

Last Updated: 13 hours ago

Size: 2.9MiB, 2 files

Comment: [Add comment](#)

Use with BI tools | Create

Search | cogment | File | Home | Help | ENG IN | 11:53 AM | 4/6/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... | CTRL + P | databricks\_workspace | 19b91a03q6@srkrec.ac.in

### Catalog Explorer

Type to filter

- hive\_metastore
- default
  - finalproductsalesorderdetails
  - manage\_productdetails
  - manage\_productsalesorderdetails
  - manage\_salesorderdetails
  - productdetails
  - salesorderdetails**
- samples

default > **default.salesorderdetails**

Overview Sample Data Details Permissions History

Created At	Fri Apr 05 17:05:28 UTC 2024
Last Access	UNKNOWN
Created By	Spark 3.3.0
Type	EXTERNAL
Location	abfss://processed@databrickspractice.dfs.core.windows.net/SalesOrderDetails
Properties	delta.minReaderVersion=1 delta.minWriterVersion=2

Use with BI tools | Create

Search | cogment | File | Home | Help | ENG IN | 12:43 AM | 4/6/2024

The screenshot shows the Microsoft Azure Storage Explorer interface. The left sidebar displays the 'processed' container with options like Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access tokens, Manage ACL, Access policy, Properties, and Metadata. The main area shows a list of blobs with columns for Name, Modified, Access tier, Archive status, and Blob type. Two files are listed: '\_delta\_log' (Modified 04/04/2024, 22:35:30) and two part files ('part-00000...' and 'part-00001...') both modified on 04/04/2024, 22:35:30, with Hot (Inferred) access tier and Block blob type. A search bar at the top right allows searching by prefix (case-sensitive). The bottom status bar shows the date and time (11:54 AM, 4/6/2024).

# Pipeline Result on success:

The screenshot shows the Microsoft Azure Data Factory pipeline results page. The pipeline run ID is 4de27cac-2f21-435e-b85f-1b8ae2ecbe6a. The pipeline status is Succeeded. The output section shows two items: 'Web on succeed' (Succeeded) and 'Execute Pipeline1' (Succeeded). The pipeline editor on the left shows a flow starting with an 'Execute Pipeline' activity pointing to a 'Web' activity labeled 'Web on succeed' (green checkmark) and another 'Web' activity labeled 'Web on fail' (red X). The bottom status bar shows the date and time (10:37 PM, 4/5/2024).

Microsoft Azure | Data Factory > DatafactoryDatabrick452

Search factory and documentation

Preview experience Off

Data Factory Databrick\_inner\_p... main\_pipeline DS\_adls\_raw\_zip\_file\_...

Validate all Publish all

Validate Debug Add trigger

Activity name Activity status Activity type Run start Duration Integration runtime User properties Activity run ID

Notebook1	Succeeded	Notebook	4/5/2024, 10:34:51 PM	49s	AutoResolveIntegrator	f4021ef4-79ff-4a50-90aa-
Copy data1	Succeeded	Copy data	4/5/2024, 10:33:41 PM	1m 7s	AutoResolveIntegrator	8746ab1b-6489-4af5-b9e
Copy data1	Succeeded	Copy data	4/5/2024, 10:33:41 PM	1m 4s	AutoResolveIntegrator	e95cadf1-68e1-43f1-a76c
If Condition1	Succeeded	If Condition	4/5/2024, 10:33:41 PM	Less than 1s		c6aed204-379e-467f-b04
ForEach1	Succeeded	ForEach	4/5/2024, 10:33:41 PM	1m 10s		c5f0a8f2-eb3b-487c-b83e
Get Metadata raw container	Succeeded	Get Metadata	4/5/2024, 10:33:36 PM	4s	AutoResolveIntegrator	84e9a644-ff01-4085-a811
Copy data from url web	Succeeded	Copy data	4/5/2024, 10:32:07 PM	1m 29s	AutoResolveIntegrator	53ffa70e-59cc-4384-a873

Search

ENG IN 10:39 PM 4/5/2024

Microsoft Azure | Search resources, services, and docs (G+)

19B91A03Q6@srkrec.ac... SRKR ENGINEERING COLLEGE

Home > databrickspratice | Containers >

raw ...

Container

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Manage ACL

Access policy

Properties

Metadata

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: raw

Search blobs by prefix (case-sensitive)

Show deleted objects

Name	Modified	Access tier	Archive status	Blob type
cohort3data.zip	05/04/2024, 22:16:18	Hot (Inferred)		Block blob
SalesOrderHeader.csv.zip	05/04/2024, 22:33:33	Hot (Inferred)		Block blob

https://portal.azure.com/#

Search

ENG IN 10:41 PM 4/5/2024

Microsoft Azure

Search resources, services, and docs (G+)

Home > databrickspratice | Containers >

**refine** Container

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Authentication method: Access key (Switch to Microsoft Entra user account)  
Location: refine / joshi / unzipped

Search blobs by prefix (case-sensitive) Show deleted objects

Name	Modified	Access tier	Archive status	Blob type
[...]	05/04/2024, 22:34:46	Hot (Inferred)		Block blob
Product.csv	05/04/2024, 22:34:47	Hot (Inferred)		Block blob
SalesOrderDetail.csv	05/04/2024, 22:34:47	Hot (Inferred)		Block blob
SalesOrderHeader.csv	05/04/2024, 22:34:44	Hot (Inferred)		Block blob

Search ENG IN 10:43 PM 4/5/2024

Microsoft Azure

Search resources, services, and docs (G+)

Home > databrickspratice | Containers >

**processed** Container

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Authentication method: Access key (Switch to Microsoft Entra user account)  
Location: processed

Search blobs by prefix (case-sensitive) Show deleted objects

Name	Modified	Access tier	Archive status	Blob type
FinalProductSalesOrderDetails				
productdetails				
SalesOrderDetails				

https://portal.azure.com/#

Search ENG IN 10:44 PM 4/5/2024

Microsoft Azure | databricks | Search data, notebooks, recents, and more... CTRL + P databricks.workspace 19b91a03q6@srukrec.ac.in

Workflows > Runs > ADF\_DatafactoryDatabrick452\_Databrick\_inner\_pipeline\_Notebook1\_f4021ef4-79ff-4a50-90aa-9e7af834bb44 run Delete job run

Output Accessing Secret Keys

```
✓ 1.49 seconds
dbutils.secrets.help()
dbutils.secrets.list(scope='datalake_scope')
dbutils.secrets.get(scope='datalake_scope',key='DataLakeAccessKey')
```

Out[1]: '[REDACTED]'

Creating link between datalake and databricks

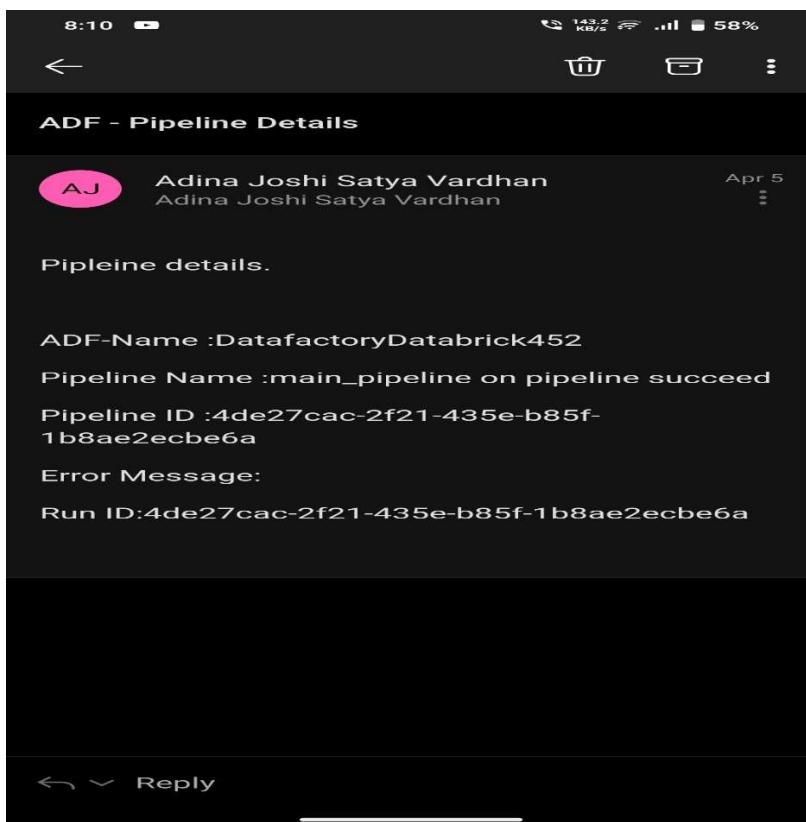
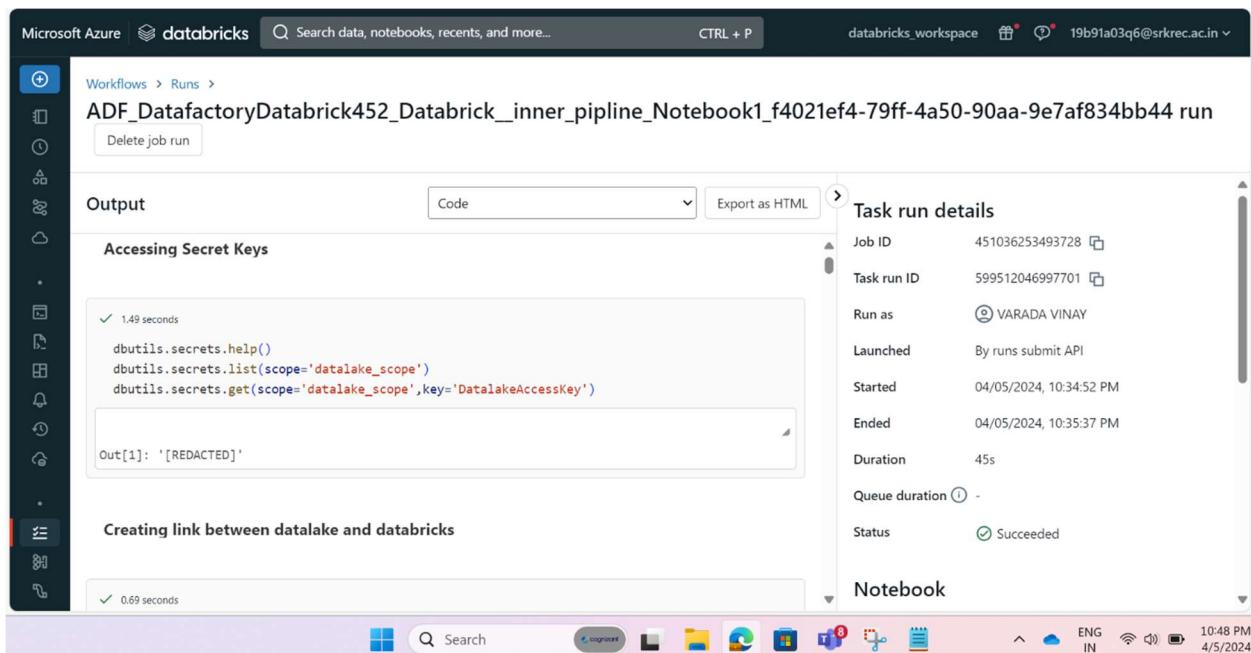
```
✓ 0.69 seconds
```

Task run details

Job ID	451036253493728
Task run ID	599512046997701
Run as	VARADA VINAY
Launched	By runs submit API
Started	04/05/2024, 10:34:52 PM
Ended	04/05/2024, 10:35:37 PM
Duration	45s
Queue duration	-
Status	Succeeded

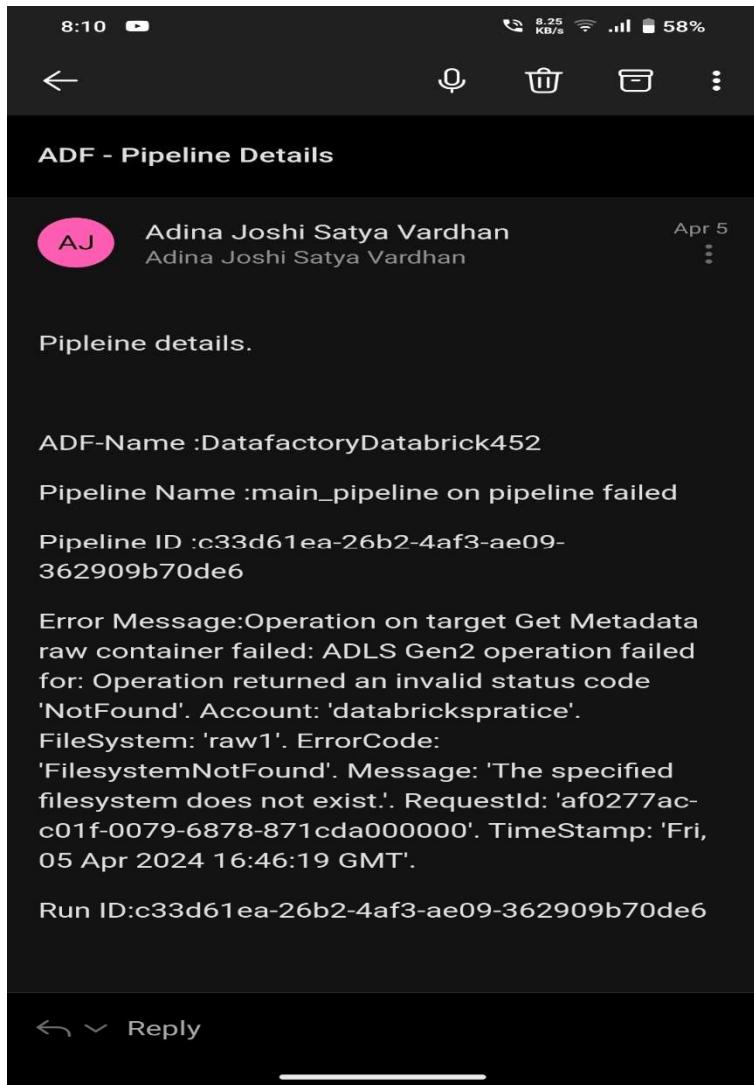
Notebook

Search ENG IN 10:48 PM 4/5/2024



Mail on pipeline succeed

## Mail on pipeline failed:



# Visualization:

