

PROJECT-07

End -to-End Pipeline Design:SCD TYPE 1 with ADLSGen2,Delta Lake ,Databricks & ADF.

Read data from ADLS GEN2

- Apply slowly changing dimension (SCD) Type 1 logic.
- Store the data in Delta Lake format as an external table.
- Parameterize and trigger it from Azure Data Factory (ADF) using dynamic parameters and Databricks notebook activity.

NANDINI RATHORE

NANDINI RATHORE

1.creating key vault.

2.creating app registration.

06c5c529-e87f-4089-ad59-9328fcc5d19b client id

NANDINI RATHORE

Client secret Jxt8Q~ubtFOSer39kU4SCGy2_S4afWOioHKFkbBV

511e51ab-918d-4a74-8cc6-089756a6a26b tenant id.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various options like Overview, Quickstart, Integration assistant, Diagnose and solve problems, Manage, Certificates & secrets (which is selected), Token configuration, API permissions, Expose an API, App roles, and Owners. The main content area is titled 'appregistration | Certificates & secrets'. It shows a section for 'Client secrets' with a table header: 'Certificates (0)', 'Client secrets (0)', 'Federated credentials (0)'. Below the table, it says 'No client secrets have been created for this application.' A modal window titled 'Add a client secret' is overlaid on the page. It has fields for 'Description' (set to 'clientsecret') and 'Expires' (set to '90 days (3 months)'). There are 'Add' and 'Cancel' buttons at the bottom of the modal.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with options like Create, Group by none, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Access policies, Resource visualizer, Events, Objects (with Keys, Secrets, Certificates), and Settings. The main content area is titled 'nvaultkey | Secrets'. It shows a message: 'The secret 'clientsecret' has been successfully created.' Below this, there's a table with columns: Name, Type, Status, and Expiration date. The table contains two rows: 'clientsecret' (Type: Secret, Status: Enabled) and 'clientid' (Type: Secret, Status: Enabled).

Vault uri:- <https://nvaultkey.vault.azure.net/>

NANDINI RATHORE

Add role assignment

Role **Members** **Conditions** **Review + assign**

Selected role Storage Blob Data Contributor

Assign access to User, group, or service principal Managed identity

Members [+ Select members](#)

Name	Object ID	Type
appregisteration	aa1842ac-3615-43bd-b960-355c3df3c9...	App

Description Optional

Review + assign **Previous** **Next** **Feedback**

Create an Azure Databricks workspace

Review + create

Summary

Basics

Workspace name	NDATABRICKS
Subscription	Azure subscription 1
Resource group	firstresourcegroup
Region	East US
Pricing Tier	premium
Managed Resource Group name	

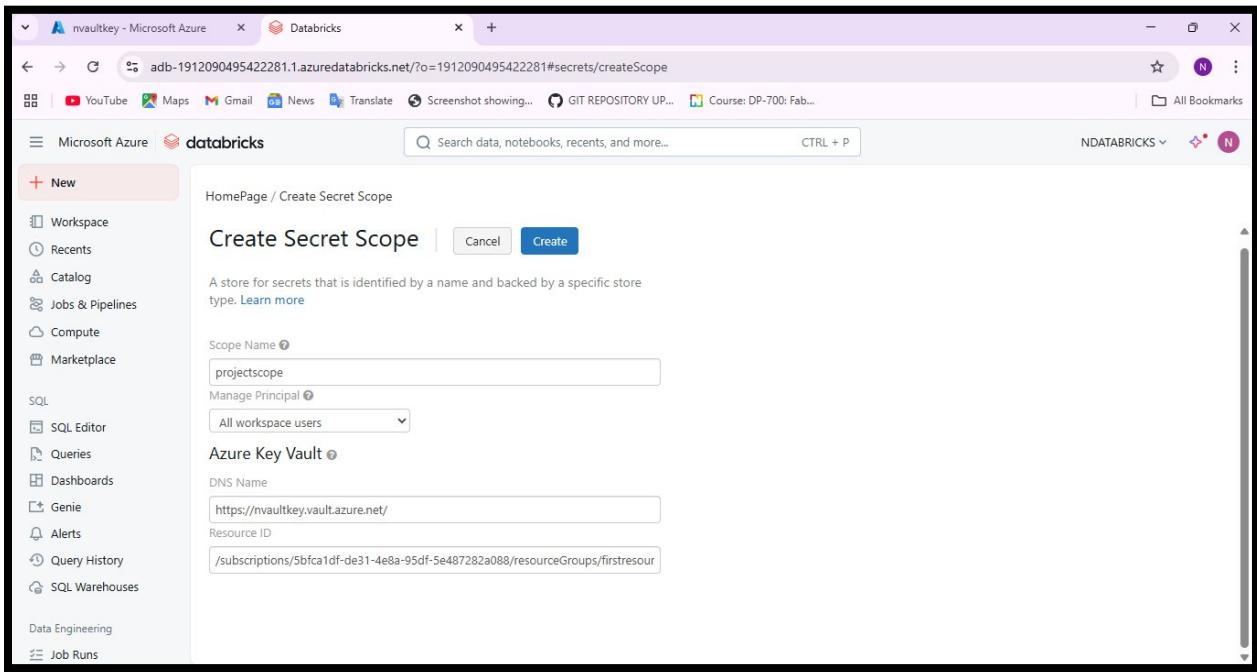
Networking

Create **< Previous**

For creating scope:-

<https://adb-1912090495422281.1.azuredatabricks.net/?o=1912090495422281#secrets/createScope>

Copy resource ID .



Paste here resource ID and azure vault URI. It is the **unique URL** used to access a specific **Azure Key Vault** in your Azure environment.

```

dbutils.help()

This module provides various utilities for users to interact with the rest of Databricks.

credentials: DatabricksCredentialUtils -> Utilities for interacting with credentials within notebooks
data: DataUtils -> Utilities for understanding and interacting with datasets (EXPERIMENTAL)
fs: DBFileUtils -> Manipulates the Databricks filesystem (DBFS) from the console
jobs: JobUtils -> Utilities for leveraging jobs features
library: LibraryUtils -> Utilities for session isolated libraries
meta: MetaUtils -> Methods to hook into the compiler (EXPERIMENTAL)
notebook: NotebookUtils -> Utilities for the control flow of a notebook (EXPERIMENTAL)
preview: Preview -> Utilities under preview category
secrets: SecretUtils -> Provides utilities for leveraging secrets within notebooks
widgets: WidgetsUtils -> Methods to create and get bound value of input widgets inside notebooks
  
```

`Dbutils.help()` command is used in Databricks notebooks to display a list of available utility commands provided by the `dbutils` module.

To check list of scopes

```

▶ ✓ 03:55 PM (<1s) 4
dbutils.secrets.listScopes()

[SecretScope(name='projectscope')]

▶ ✓ 03:55 PM (1s) 5
dbutils.secrets.list("projectscope")

[SecretMetadata(key='clientid'), SecretMetadata(key='clientsecret')]

```

Untitled Notebook 2025-07-28 15:37:01 Python Tabs: OFF NANDINI RATHORE's C... Schedule Share

File Edit View Run Help Last edit was 1 minute ago

Mount adlsgen2 with databricks using mount point 'ncplstorage'.

```

::: 7
config = {"fs.azure.account.auth.type": "OAuth",
           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
           "fs.azure.account.oauth2.client.id": dbutils.secrets.get(scope="projectscope",key="clientid"),
           "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="projectscope",key="clientSecret"),
           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/511e51ab-918d-4a74-8cc6-089756a6a26b/oauth2/token"}

dbutils.fs.mount(
    source = "abfss://container@nandiniadls.dfs.core.windows.net/",
    mount_point = "/mnt/ncplstorage",
    extra_configs = config)

True

```

```

configs = {"fs.azure.account.auth.type": "OAuth",
           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
           "fs.azure.account.oauth2.client.id": dbutils.secrets.get(scope="projectscope",key="clientid"),
           "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="projectscope",key="clientSecret"),
           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/511e51ab-918d-4a74-8cc6-089756a6a26b/oauth2/token"}

dbutils.fs.mount(
    source = "abfss://container@nandiniadls.dfs.core.windows.net/",
    mount_point = "/mnt/ncplstorage",
    extra_configs = config)

```

To check the mount.

```
▶ ✓ 2 minutes ago (<1s) 9 Python [ ] ⚙ 🗑
dbutils.fs.mounts()

[MountInfo(mountPoint='/databricks-datasets', source='databricks-datasets', encryptionType=''),
 MountInfo(mountPoint='/Volumes', source='UnityCatalogVolumes', encryptionType=''),
 MountInfo(mountPoint='/databricks/mlflow-tracking', source='databricks/mlflow-tracking', encryptionType=''),
 MountInfo(mountPoint='/databricks-results', source='databricks-results', encryptionType=''),
 MountInfo(mountPoint='/databricks/mlflow-registry', source='databricks/mlflow-registry', encryptionType=''),
 MountInfo(mountPoint='/Volume', source='DbfsReserved', encryptionType=''),
 MountInfo(mountPoint='/volumes', source='DbfsReserved', encryptionType=''),
 MountInfo(mountPoint='/', source='DatabricksRoot', encryptionType=''),
 MountInfo(mountPoint='/mnt/ncplstorage', source='abfss://container@nandiniadls.dfs.core.windows.net/', encryptionType=''),
 MountInfo(mountPoint='/volume', source='DbfsReserved', encryptionType='')]
```

to check number of files in adls using mount point.

```
▶ ✓ 4 minutes ago (1s) 9
dbutils.fs.ls("/mnt/ncplstorage")

[FileInfo(path='dbfs:/mnt/ncplstorage/Azurecapacities.csv', name='Azurecapacities.csv', size=49, modificationTime=175329487000),
 FileInfo(path='dbfs:/mnt/ncplstorage/firstfile.csv', name='firstfile.csv', size=98, modificationTime=1753663867000),
 FileInfo(path='dbfs:/mnt/ncplstorage/folder1/', name='folder1/', size=0, modificationTime=1753292131000),
 FileInfo(path='dbfs:/mnt/ncplstorage/secondfile.csv', name='secondfile.csv', size=92, modificationTime=1753663884000)]
```

Read a file from adlsgen2.

```
▶ ✓ 1 minute ago (2s) 14
df=spark.read.format("csv").option("header",True).option("inferSchema",True).load("/mnt/ncplstorage/firstfile.csv")
display(df)
```

▶ (3) Spark Jobs

▶ df: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 2 more fields]

Table +

\downarrow id	\downarrow name	\downarrow city	\downarrow phonenumbers
1	nandini	ptl	814653852
2	lakhwinder	ptl	754123694
3	dimple	hry	888742012

NANDINI RATHORE

Home > Storage accounts > nandinirathi | Containers >

container Container

Search Add Directory Upload Refresh Delete Copy Paste Rename Acquire lease Break lease Edit columns

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

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

Search blobs by prefix (case-sensitive) Only show active objects

Showing all 4 items

Name	Last modified	Access tier	Blob type	Size	Lease state
folder1	23/07/2025, 13:35:31		Block blob	49 B	Available
Azurecapacities.csv	23/07/2025, 14:21:10	Hot (Inferred)	Block blob	98 B	Available
firstfile.csv	27/07/2025, 20:51:07	Hot (Inferred)	Block blob	92 B	Available
secondfile.csv	27/07/2025, 20:51:24	Hot (Inferred)	Block blob	92 B	Available

Connect Databricks with SQLDB.

```
05:22 PM (2s) 16
dbutils.secrets.list("projectscope")
[SecretMetadata(key='clientid'),
SecretMetadata(key='clientsecret'),
SecretMetadata(key='password'),
SecretMetadata(key='username')]
```

```
05:23 PM (2s) 17
username=dbutils.secrets.get(scope="projectscope",key="username")
password=dbutils.secrets.get(scope="projectscope",key="password")
```

File Edit View Run Help Python Tabs: ON Last edit was now

Run all NANDINI RATHORE's C... Schedule Share

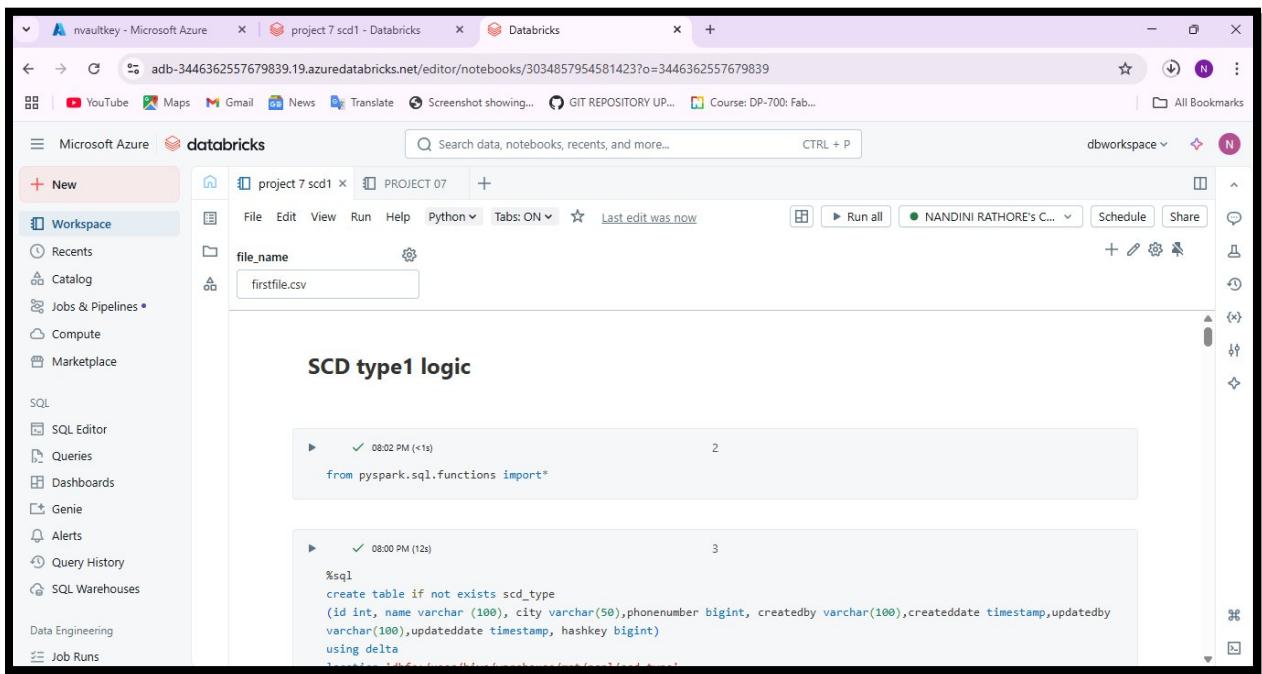
```
Just now (3s) 18
df=(spark.read
    .format("jdbc")
    .option("url", "jdbc:sqlserver://nandinisqlserver.database.windows.net:1433;database=sqldb;")
    .option("dbtable", "SalesLT.Product")
    .option("username", username)
    .option("password", password)
    .load()
)
display(df)

(1) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [ProductID: integer, Name: string ... 15 more fields]
```

Table +

ProductID	Name	ProductNumber	Color	StandardCost	ListPrice	Unit
680	HL Road Frame - Black, 58	FR-R92B-58	Black	1059.3100	1431.5000	58
706	HL Road Frame - Red, 58	FR-R92R-58	Red	1059.3100	1431.5000	58
3						

NANDINI RATHORE



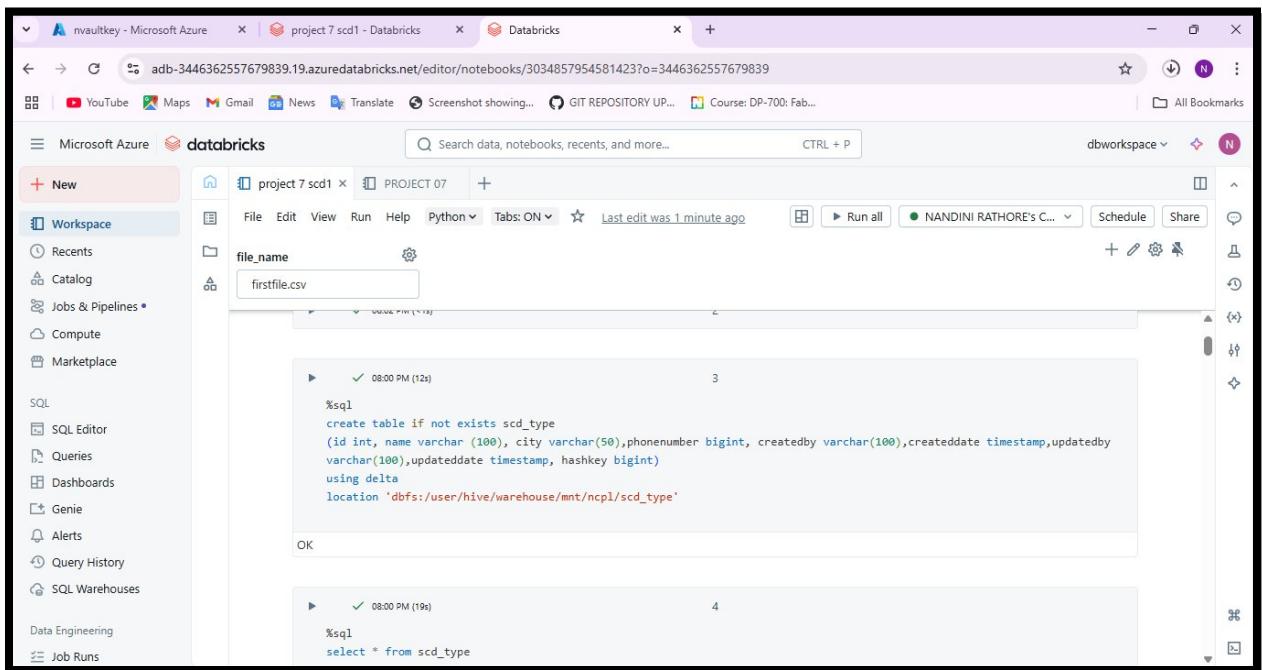
The screenshot shows the Databricks SQL Editor interface. On the left, the sidebar includes options like Workspace, Catalog, Jobs & Pipelines, Compute, Marketplace, and Data Engineering. The main area has tabs for 'project 7 scd1' and 'PROJECT 07'. A search bar at the top right says 'Search data, notebooks, recents, and more...'. Below it, there's a toolbar with 'File', 'Edit', 'View', 'Run', 'Help', 'Python', 'Tabs: ON', and 'Last edit was now'. To the right of the toolbar are buttons for 'Run all', 'Schedule', and 'Share'. The notebook content starts with a title 'SCD type1 logic' and two code snippets:

```

2
from pyspark.sql.functions import*
3
%sql
create table if not exists scd_type
(id int, name varchar(100), city varchar(50), phonenumbers bigint, createdby varchar(100), createddate timestamp, updatedby
varchar(100), updateddate timestamp, hashkey bigint)
using delta
1
    id      name   city      phonenumbers  createdby  createddate  updatedby  updateddate  hashkey
  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---

```

Creating a target table in databricks.



This screenshot continues from the previous one. The code has been modified to include a location for the Delta table:

```

3
%sql
create table if not exists scd_type
(id int, name varchar(100), city varchar(50), phonenumbers bigint, createdby varchar(100), createddate timestamp, updatedby
varchar(100), updateddate timestamp, hashkey bigint)
using delta
location 'dbfs:/user/hive/warehouse/mnt/ncpl/scd_type'
OK
4
%sql
select * from scd_type

```

The execution steps are labeled 3 and 4. Step 3 shows the creation of the table with the specified location. Step 4 shows a query being run to select all data from the newly created table.

Creating delta table because while creating I gave location for the table and in delta table if we delete table its data is not deleted it remain in storage.

The screenshot shows the Databricks SQL interface. At the top, there is a file selector labeled 'file_name' with 'firstfile.csv' selected. Below it, a code editor window displays a query:

```
%sql
select * from scd_type
```

The output pane shows the result of the query:

```
_sqldf: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 7 more fields]
```

Below the output is a table view with the following columns:

	Δ id	Δ name	Δ city	Δ phonenumbers	Δ createdby	Δ createddate	Δ update...	Δ updatedate
No rows returned								

to check type of the table we use this command.

The screenshot shows the Databricks SQL interface. The code editor window displays the command:

```
%sql desc formatted scd_type
```

The output pane shows the detailed schema information:

```
_sqldf: pyspark.sql.dataframe.DataFrame = [col_name: string, data_type: string ... 1 more field]
```

Below the output is a table view with the following columns:

	Δ col_name	Δ data_type
1	id	int
2	name	varchar(100)
3	city	varchar(50)
4	phonenumbers	bigint
5	createdby	varchar(100)

The screenshot shows the 'Detailed Table Information' view for the 'scd_type' table. The table has 22 rows of information:

10		
11	# Detailed Table Information	
12	Catalog	spark_catalog
13	Database	default
14	Table	scd_type
15	Created Time	Mon Jul 28 23:37:32 UTC 2025
16	Last Access	UNKNOWN
17	Created By	Spark 3.5.0
18	Type	EXTERNAL
19	Location	dbfs:/user/hive/warehouse/mnt/ncpl/scd_type
20	Provider	delta
21	Owner	root
22	Table Properties	[delta.enableDeletionVectors=true,delta.feature.deletionVectors=supported,delta.minReaderVersion=3,delta.minWriterVersion=7]

Type and location of the table.

```
dbutils.widgets.text("file_name", " ")
file_name=dbutils.widgets.get("file_name")
print(file_name)

firstfile.csv
```

```
src_path="/mnt/ncplstorage" + file_name
print(src_path)
```

dbutils.widgets are used to **create input controls** in your Databricks notebook, so we can make code dynamic, interactive, and reusable.

Pass different values (like filenames, dates) without changing the code.

Run the same notebook with different inputs.

```
src_path="/mnt/ncplstorage" + file_name
print(src_path)
tgt_path="/mnt/ncplstorage/tgt_table"
print(tgt_path)

/mnt/ncplstorage firstfile.csv
```

```
tgt_path="/mnt/ncplstorage/tgt_table"
```

```

file_name
firstfile.csv

read file

08:00 PM (12s) 12
df_src=spark.read.format("csv").option("header",True).option("inferSchema",True).load("/mnt/ncplstorage/firstfile.csv")
display(df_src)

(3) Spark Jobs
df_src: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 2 more fields]

Table + 
+---+---+---+---+
| 1 | 1 | nandini | ptl |
| 2 | 2 | lakhwinder | ptl |
| 3 | 3 | dimple | hry |
+---+---+---+---+

```

Read the csvfile.

```

file_name
firstfile.csv

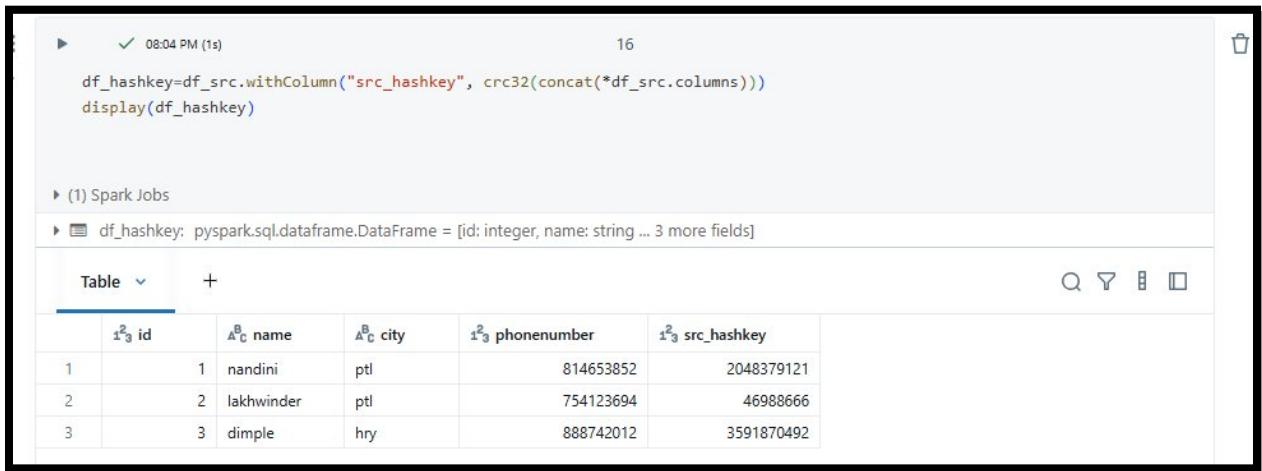
Create data in target table.

08:00 PM (2s) 14
from delta import DeltaTable
dbtable=DeltaTable.forPath(spark,"dbfs:/user/hive/warehouse/mnt/ncpl/scd_type")
dbtable.toDF().show()

+---+---+---+---+---+---+---+---+
| id|name|city|phonenumer|createdby|createddate|updatedby|updateddate|hashkey|
+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+

```

At first target data is empty because no new data is added.



```
08:04 PM (1s) 16
df_hashkey=df_src.withColumn("src_hashkey", crc32(concat(*df_src.columns)))
display(df_hashkey)

(1) Spark Jobs
df_hashkey: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 3 more fields]

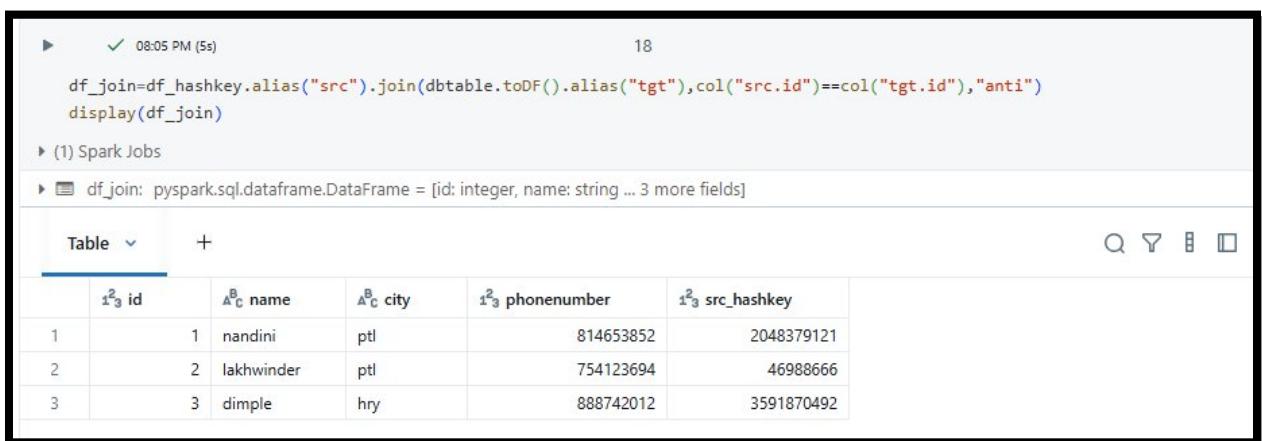
Table + Q Y E □
+---+---+---+---+---+
| id | name | city | phonenumer | src_hashkey |
+---+---+---+---+---+
| 1  | nandini | ptl | 814653852 | 2048379121 |
| 2  | lakhwinder | ptl | 754123694 | 46988666 |
| 3  | dimple | hry | 888742012 | 3591870492 |
+---+---+---+---+---+
```

Generating hashkey



```
08:05 PM (5s) 18
df_join=df_hashkey.alias("src").join(dbtable.toDF().alias("tgt"),col("src.id")==col("tgt.id"),"anti")
display(df_join)

(1) Spark Jobs
df_join: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 3 more fields]
```

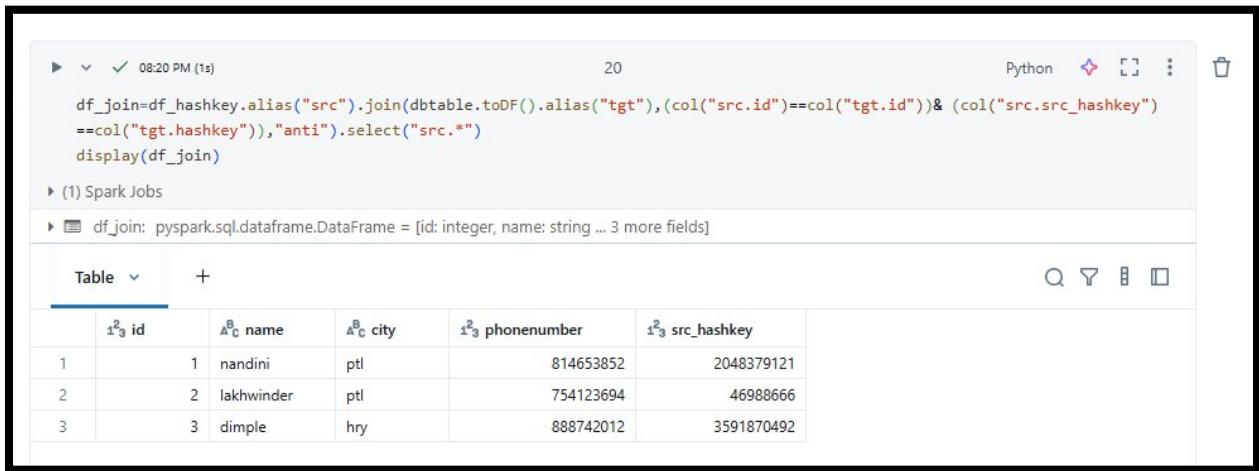


```
08:05 PM (5s) 18
df_join=df_hashkey.alias("src").join(dbtable.toDF().alias("tgt"),col("src.id")==col("tgt.id"),"anti")
display(df_join)

(1) Spark Jobs
df_join: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 3 more fields]

Table + Q Y E □
+---+---+---+---+---+
| id | name | city | phonenumer | src_hashkey |
+---+---+---+---+---+
| 1  | nandini | ptl | 814653852 | 2048379121 |
| 2  | lakhwinder | ptl | 754123694 | 46988666 |
| 3  | dimple | hry | 888742012 | 3591870492 |
+---+---+---+---+---+
```

"anti" ⚲ Left anti join: returns only the rows from src (left side) that do not have a matching row in tgt based on the condition.



```

08:20 PM (1s) 20 Python
df_join=df_hashkey.alias("src").join(dbtable.toDF().alias("tgt"),(col("src.id")==col("tgt.id"))& (col("src.src_hashkey") ==col("tgt.hashkey")),"anti").select("src.*")
display(df_join)

(1) Spark Jobs
df_join: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 3 more fields]

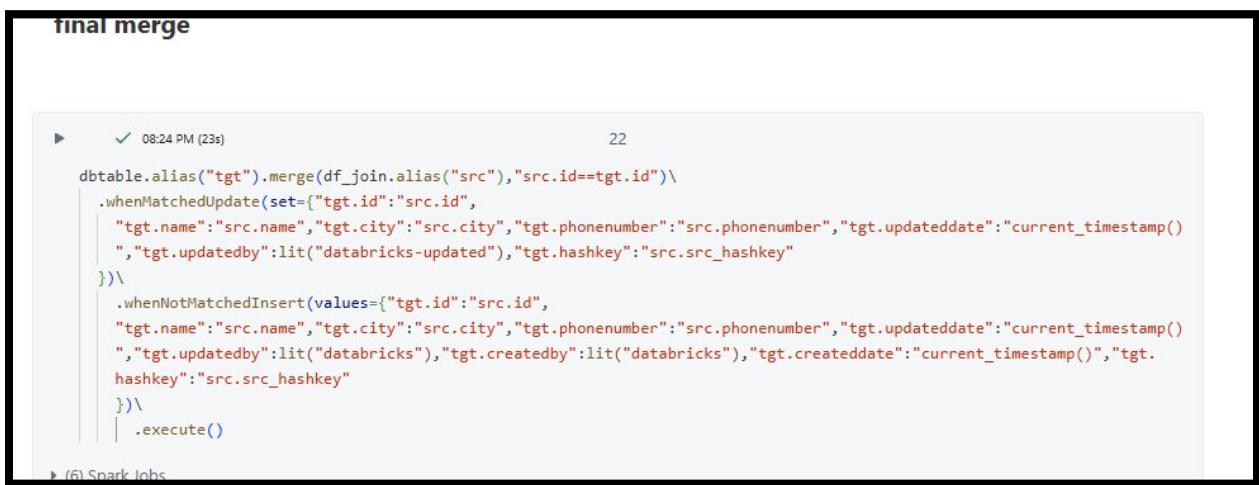
Table + 
+---+---+---+---+---+
| id | name | city | phononenumber | src_hashkey |
+---+---+---+---+---+
| 1 | nandini | ptl | 814653852 | 2048379121 |
| 2 | lakhwinder | ptl | 754123694 | 46988666 |
| 3 | dimple | hry | 888742012 | 3591870492 |
+---+---+---+---+---+

```

Join src table with target table, on the basis of id and hashkey.

Src_id==tgt_id

Src_hashkey==tgt_hashkey.



```

final merge
08:24 PM (23s) 22 Python
dbtable.alias("tgt").merge(df_join.alias("src"),"src.id==tgt.id")\
.whenMatchedUpdate(set={"tgt.id": "src.id",
"tgt.name": "src.name", "tgt.city": "src.city", "tgt.phonenumber": "src.phonenumber", "tgt.updateddate": "current_timestamp()",
", "tgt.updatedby": lit("databricks-updated"), "tgt.hashkey": "src.src_hashkey"
})\
.whenNotMatchedInsert(values={"tgt.id": "src.id",
"tgt.name": "src.name", "tgt.city": "src.city", "tgt.phonenumber": "src.phonenumber", "tgt.updateddate": "current_timestamp()",
", "tgt.updatedby": lit("databricks"), "tgt.createdby": lit("databricks"), "tgt.createddate": "current_timestamp()", "tgt.
hashkey": "src.src_hashkey"
})\
.execute()

(6) Spark Jobs

```

```
%sql
select * from scd_type
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 7 more fields]

id	name	city	phonenum	createddate	updatedate	user	hashkey
1	nandini	ptl	814653852	databricks	2025-07-29T0...	databricks	2048379121
2	lakhwinder	ptl	754123694	databricks	2025-07-29T0...	databricks	46988666
3	dimple	hry	888742012	databricks	2025-07-29T0...	databricks	3591870492

Target table gives data of new inserted values.

I used another file from adls storage account and update,insert values.

Now add new values and update 1 row.

```
file_name
secondfile.csv
```

```
Just now(<1s)
dbutils.widgets.text("file_name", "")
file_name=dbutils.widgets.get("file_name")
print(file_name)
```

```
Just now(<1s)
src_path="/mnt/ncplstorage" + file_name
print(src_path)
tgt_path="/mnt/ncplstorage/tgt_table"
```

The screenshot shows a Databricks notebook titled "project 7 scd1". The sidebar on the left is open, showing options like Workspace, Recents, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. The main area displays a code cell in Python:

```
Just now (2s) 12
df_src=spark.read.format("csv").option("header",True).option("inferSchema",True).load("/mnt/ncplstorage/secondfile.csv")
display(df_src)
```

Below the code, a table is displayed with the following data:

x ² id	x ² name	x ² city	x ² phonenumer
1	sahil	hyd	123456789
2	priya	chd	258741369
3	somya	pune	456789123

The screenshot shows a Databricks notebook titled "project 7 scd1". The sidebar on the left is open, showing options like Workspace, Recents, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. The main area displays a code cell in Python:

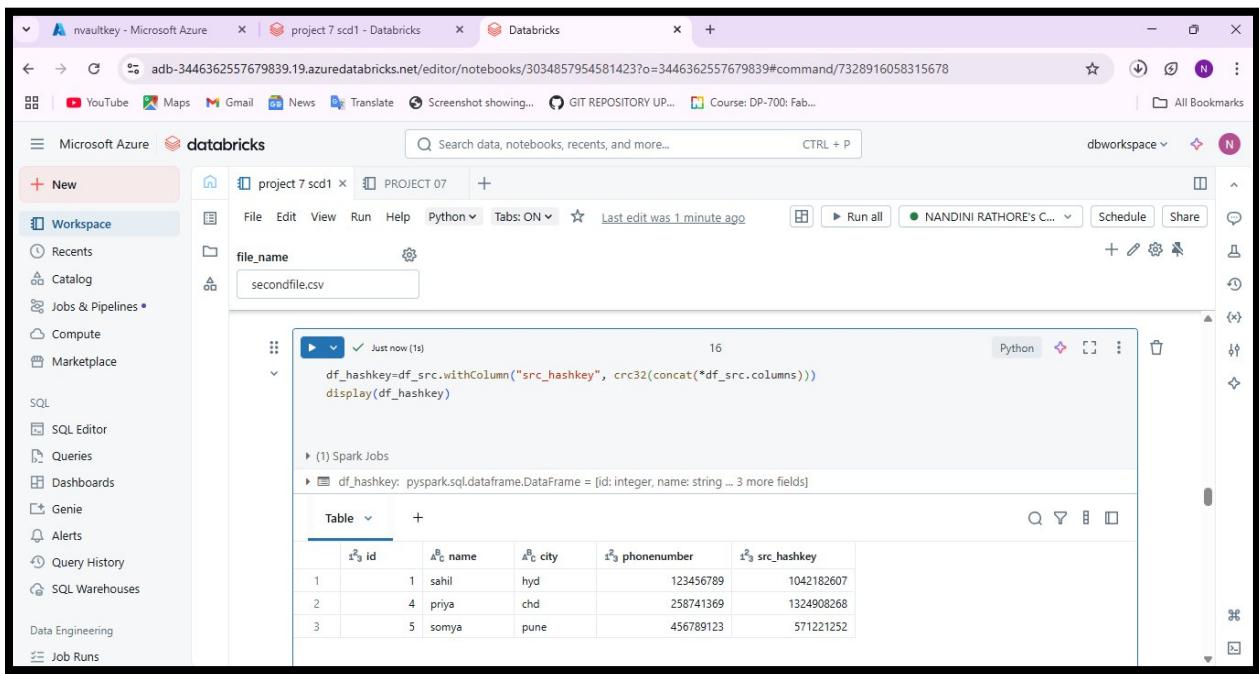
```
Run cell (Ctrl + Enter) Just now (1s) 14
from delta import DeltaTable
dbtable=DeltaTable.forName(spark,"dbfs:/user/hive/warehouse/mnt/ncpl/scd_type")
dbtable.toDF().show()
```

Below the code, a table is displayed with the following data:

id	name	city	phonenumer	createdby	createddate	updatedby	updateddate	hashkey
1	nandini	ptl	814653852	databricks	2025-07-29 00:24:...	databricks	2025-07-29 00:24:...	2048379121
2	lakhwinder	ptl	754123694	databricks	2025-07-29 00:24:...	databricks	2025-07-29 00:24:...	46988666
3	dimple	hryl	888742012	databricks	2025-07-29 00:24:...	databricks	2025-07-29 00:24:...	3591870492

Now it is showing data in target table.

NANDINI RATHORE



Screenshot of a Databricks notebook titled "project 7 scd1". The notebook interface includes a sidebar with options like Workspace, Catalog, and Compute. The main area shows a code cell with Python code and its output.

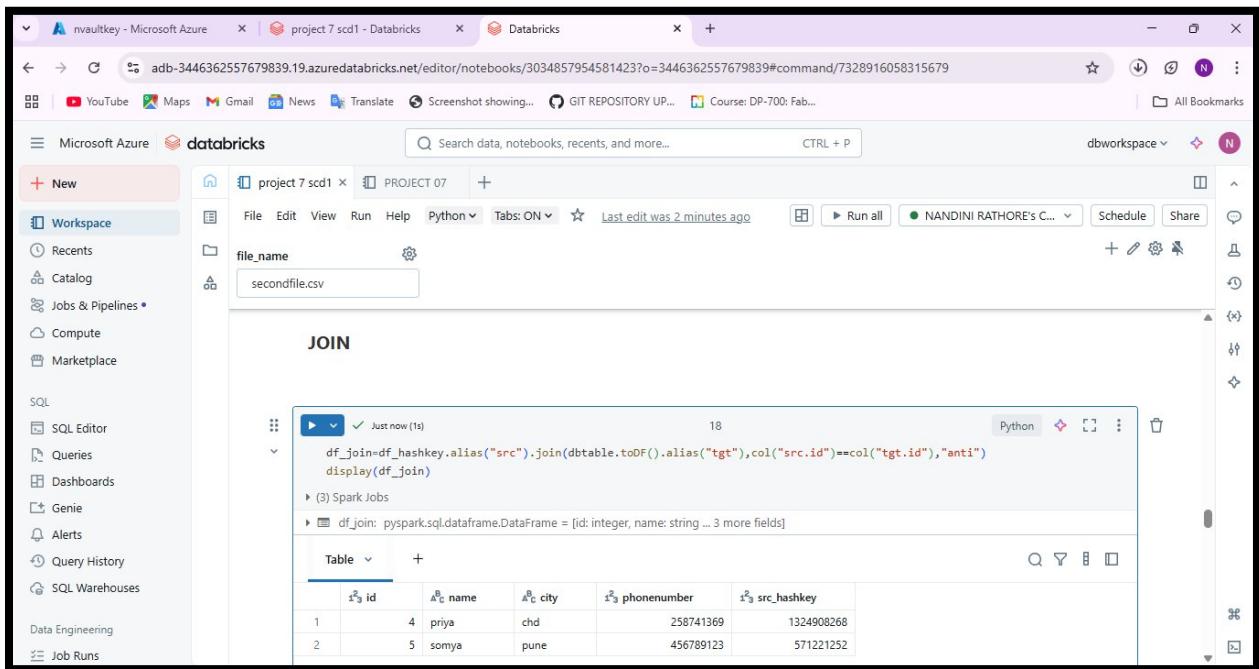
```
Just now (1s) 16
df_hashkey=df_src.withColumn("src_hashkey", crc32(concat(*df_src.columns)))
display(df_hashkey)
```

(1) Spark Jobs

```
df_hashkey: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 3 more fields]
```

Table

id	name	city	phononenumber	src_hashkey
1	sahil	hyd	123456789	1042182607
2	priya	chd	258741369	1324908268
3	somya	pune	456789123	571221252



Screenshot of a Databricks notebook titled "project 7 scd1". The notebook interface includes a sidebar with options like Workspace, Catalog, and Compute. The main area shows a code cell with Python code and its output table.

```
Just now (1s) 18
df_join=df_hashkey.alias("src").join(dbtable.toDF().alias("tgt"),col("src.id") == col("tgt.id"), "anti")
display(df_join)
```

(3) Spark Jobs

```
df_join: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 3 more fields]
```

Table

id	name	city	phononenumber	src_hashkey
4	priya	chd	258741369	1324908268
5	somya	pune	456789123	571221252

NANDINI RATHORE

The screenshot shows a Databricks workspace interface. On the left, there's a sidebar with various options like Workspace, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, and Data Engineering. The main area has tabs for 'project 7 scd1' and 'PROJECT 07'. A search bar at the top says 'Search data, notebooks, recents, and more...'. Below it, a table titled 'secondfile.csv' is displayed. The table has columns: id, name, city, phonenumbers, and src_hashkey. The data is as follows:

	<code>s²_id</code>	<code>^B_name</code>	<code>^B_city</code>	<code>^B_phonenum...</code>	<code>^B_src_hashkey</code>
1	5	somya	pune	456789123	571221252
2	1	sahil	hyd	123456789	1042182607
3	4	priya	chd	258741369	1324908268

This screenshot shows a similar Databricks workspace setup. The sidebar and tabs are identical. The main area displays a table titled 'secondfile.csv' with a different schema and data. The columns are: id, name, city, phonenumbers, createddate, updatedby, updateddate, and hash. The data is as follows:

	<code>s²_id</code>	<code>^B_name</code>	<code>^B_city</code>	<code>^B_phonenum...</code>	<code>^B_created...</code>	<code>^B_updatedby</code>	<code>^B_updateddate</code>	<code>^B_hash</code>	
1	1	sahil	hyd	123456789	databricks	> 2025-07-29T0...	databricks-updated	> 2025-07-29...	104
2	2	lakhwinder	ptl	754123694	databricks	> 2025-07-29T0...	databricks	> 2025-07-29...	4
3	3	dimple	hry	888742012	databricks	> 2025-07-29T0...	databricks	> 2025-07-29...	359
4	4	priya	chd	258741369	databricks	> 2025-07-29T0...	databricks	> 2025-07-29...	132
5	5	somya	pune	456789123	databricks	> 2025-07-29T0...	databricks	> 2025-07-29...	57

1 row is updated as it get “ databricks-updated” in updatedby column And row 4,5 are new inserted values. In scd typ 1 it keeps the record of new inserted and updated rows, didn’t keep historical data.

NANDINI RATHORE

The screenshot shows the Databricks workspace interface. On the left, the sidebar includes options like Workspace, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, Data Engineering, and Job Runs. The main area displays a notebook titled "project 7 scd1". A cell is currently executing, showing the command:

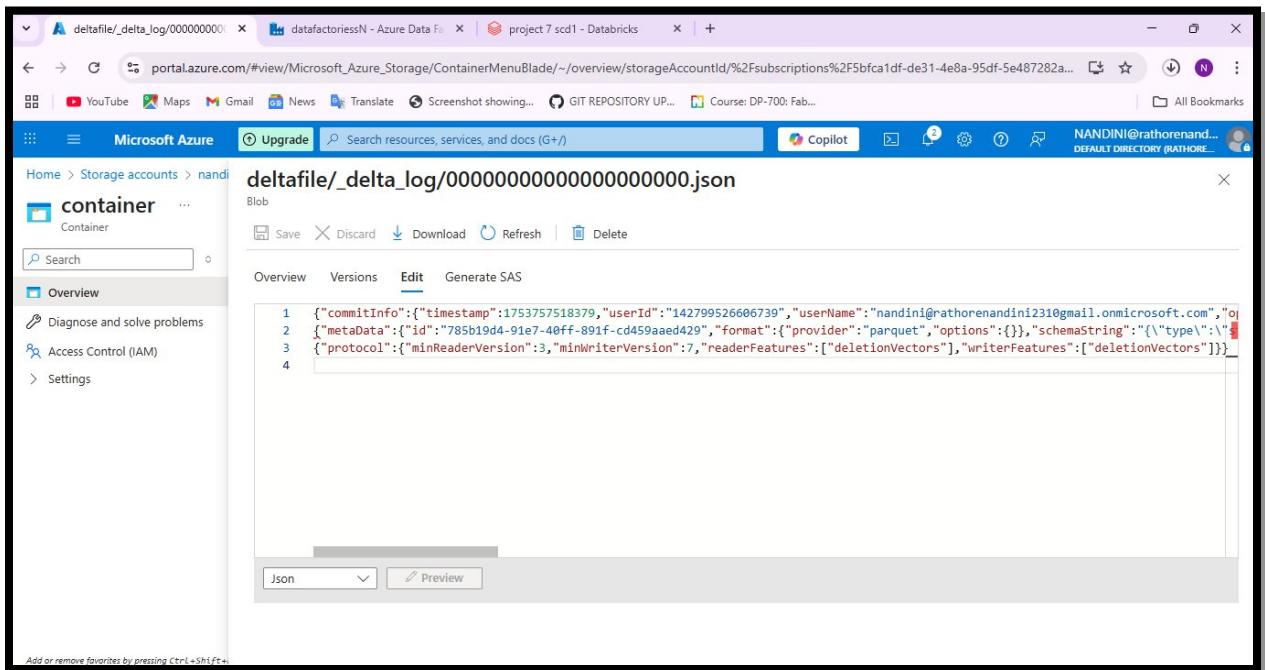
```
df._join=df._join.write.format("parquet").save("/mnt/ncplstorage/scd_type")
```

Below the cell, it says "Just now (3s)" and "25". The status bar indicates "Python". A tooltip provides keyboard shortcuts: [Shift+Enter] to run and move to next cell, [Ctrl+Shift+P] to open the command palette, and [Esc H] to see all keyboard shortcuts.

The screenshot shows the Azure Storage Explorer interface. It displays a list of blobs in the "scd_type" container. The blobs are:

Name	Last modified	Access tier	Blob type	Size	Lease state
[..]	28/07/2025, 22:32:30	Hot (Inferred)	Block blob	0	Available
_SUCCESS	28/07/2025, 22:32:29	Hot (Inferred)	Block blob	124 B	Available
_committed_8185899185133962374...	28/07/2025, 22:32:29	Hot (Inferred)	Block blob	0	Available
_started_8185899185133962374	28/07/2025, 22:32:29	Hot (Inferred)	Block blob	0	Available
part-00000-tid-8185899185133962374...	28/07/2025, 22:32:29	Hot (Inferred)	Block blob	847 B	Available

A tooltip highlights the last row: "part-00000-tid-8185899185133962374-e9ea699f-5757-4e86-ba4a-3ffcd1aa0fd-170-1.c000.snappy.parquet".



The _delta_log/ folder and its .json files:

- Are required for Delta Lake to function
- Store metadata and transactional history
- Make Delta format more powerful than plain Parquet.
- Delta lake are formed on the top of parquet format supports ACID transactions,schema evolution, time travel, efficient reads.

The screenshot shows the Databricks Access tokens settings page. The left sidebar has a 'New' button and categories like Workspace, Recents, Catalog, Jobs & Pipelines, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. Under 'User', it lists Profile, Preferences, Developer (which is selected), Linked accounts, and Notifications. The main content area is titled 'Access tokens' and says: 'Personal access tokens can be used for secure authentication to the [Databricks API](#) instead of passwords.' It shows a table with one row: Comment: ACCESS TOKEN, Creation: 2025-07-28 20:51:18 EDT, and Expiration: 2025-10-26 20:51:18 EDT.

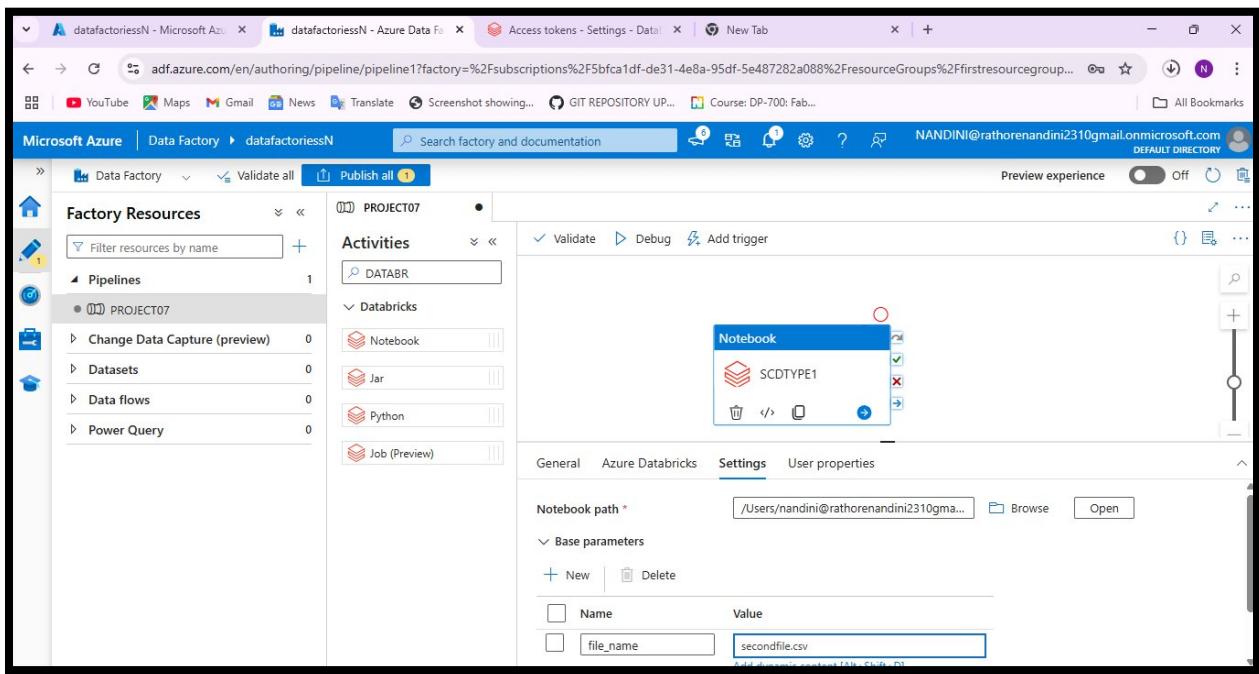
For run pipeline in adf we need access token .

Profileàsettingsàusersàdevelopersàmanageàaccess token.

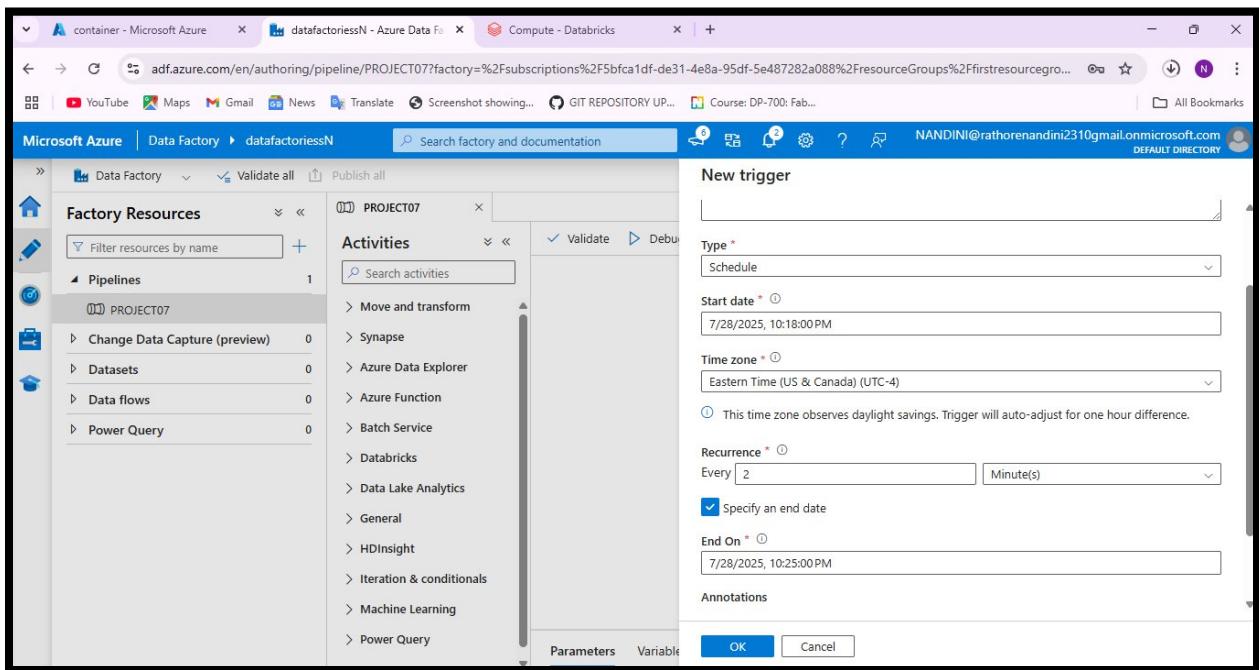
dapi936b47130cd59c9732f9ac8ecc82752f

The screenshot shows the 'New linked service' dialog in Azure Data Factory. On the left, the 'Factory Resources' sidebar shows 'Pipelines' (1), 'PROJECT07', 'Change Data Capture (preview)', 'Datasets', 'Data flows', and 'Power Query'. In the center, under 'PROJECT07', 'Activities' are listed: DATA BRICKS, Notebook, Jar, Python, and Job (Preview). On the right, the 'Databricks linked service' configuration is shown. It includes fields for 'Databricks workspace' (set to 'dbworkspace'), 'Select cluster' (radio buttons for 'New job cluster', 'Existing interactive cluster' (selected), and 'Existing instance pool'), 'Databrick Workspace URL' (set to <https://adb-3446362557679839.19.azuredatabricks.net>), 'Authentication type' (set to 'Access Token'), 'Access token' (a redacted input field), and 'Choose from existing clusters' (a dropdown menu showing 'NANDINI RATHORE's Cluster 2025-07-28 15:33:59'). At the bottom, there are 'Create' and 'Cancel' buttons, and a status message 'Connection successful' with a green checkmark.

Creating linked service in azure data factory.



Select the notebook path and give parameters which is also used in notebooks.



NANDINI RATHORE

The screenshot shows the Microsoft Azure Data Factory Pipeline runs page. The left sidebar navigation includes Dashboards, Runs, Pipeline runs (selected), Trigger runs, Change Data Capture (preview), Runtimes & sessions, Integration runtimes, Data flow debug, Notifications, and Alerts & metrics. The main content area displays a table of Pipeline runs. The table has columns: Pipeline name, Run start, Run end, Duration, Triggered by, Status, and Run. There are four rows for PROJECT07, all triggered by trigger1 and succeeded. The status column shows green checkmarks.

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
PROJECT07	7/28/2025, 10:16:08 PM	7/28/2025, 10:19:49 PM	3m 42s	Manual trigger	✓ Succeeded	Original
PROJECT07	7/28/2025, 10:20:00 PM	7/28/2025, 10:20:38 PM	38s	trigger1	✓ Succeeded	Original
PROJECT07	7/28/2025, 10:22:00 PM	7/28/2025, 10:22:38 PM	38s	trigger1	✓ Succeeded	Original
PROJECT07	7/28/2025, 10:24:00 PM	7/28/2025, 10:24:38 PM	38s	trigger1	✓ Succeeded	Original

The screenshot shows the Microsoft Azure Data Factory Activities page. The left sidebar navigation includes Data Factory (selected), Validate all, Pipelines, PROJECT07, Change Data Capture (preview), Datasets, Data flows, and Power Query. The main content area shows a list of activities under PROJECT07. One activity, 'DATABR', is selected, showing its details. The details pane shows the activity type is 'Monitor real-time execution in Azure Databricks', duration is 00:02:18, and the run page URL is <https://adb-3446362557679839.19.azuredatabricks.net/?o=3446362557679839#job/1052405368639734/run>. Below the details, there is a preview of the run results, showing one item named 'SCDTYPE1' with a status of 'Succeeded'.

Trigger pipeline and by clicking on this URL, to open the notebook manually to check the code or debug.

Jobs run in databricks.

The screenshot shows the Databricks interface with the 'Jobs & Pipelines' section selected. On the left sidebar, under 'Data Engineering', 'Job Runs' is highlighted. In the main area, there are three cards: 'Ingestion pipeline', 'ETL pipeline', and 'Job'. Below these cards, the 'Job runs' tab is selected, showing a timeline from 27 Jul, 12 AM to 28 Jul, 12 PM. A single job run is listed, starting at 28 Jul, 09:35 PM, run by 'NANDINI RAT...', and completed successfully ('Succeeded').

The screenshot shows the Azure Data Factory interface. On the left, the 'Factory Resources' pane shows a 'Pipelines' section with one item named 'PROJECT07'. In the main pane, under 'Activities', a 'Databricks' section is expanded, showing 'Notebook', 'Jar', 'Python', and 'Job (Preview)'. A specific notebook activity named 'SCDTYPE1' is selected. At the bottom, the 'Pipeline status' is shown as 'Succeeded'. The pipeline run details table shows one item: 'SCDTYPE1' (Status: Succeeded, Activity type: Notebook, Run start: 7/28/2025, 9:33:58 PM, Duration: 2m 22s, Integration: AutoResolve).

In this project we use Azure Data Factory , because it is good at moving data, but Databricks Notebooks are better at transforming or cleaning the data using Python or Spark.

CONCLUSION:->

- dbutils.widgets are used to **create input controls** in your Databricks notebook, so we can make code dynamic, interactive, and reusable.
 - Pass different values (like filenames, dates) without changing the code.
 - Run the same notebook with different inputs.
-
- In this project we use Azure Date Factory , because it is good at moving data, but Databricks Notebooks are better at transforming **or** cleaning the data using Python or Spark.
 - We use ADF to run notebooks so we can automate and combine: Data copying (by ADF)
 - Data cleaning/processing (by notebook),In one pipeline.
 - dbutils.fs.mount() doesn't work with Unity Catalog:

Unity Catalog is a new system in Databricks that helps manage data access securely and centrally.

But It blocks dbutils.fs.mount() because:

- Mounted storage can't be tracked or controlled by Unity Catalog.
- Unity Catalog wants full control and security over data access.
- Mounts could bypass security rules, so they are not allowed.