Data Handling with SQL Database, ADLS Gen2, and Databricks

Objective:

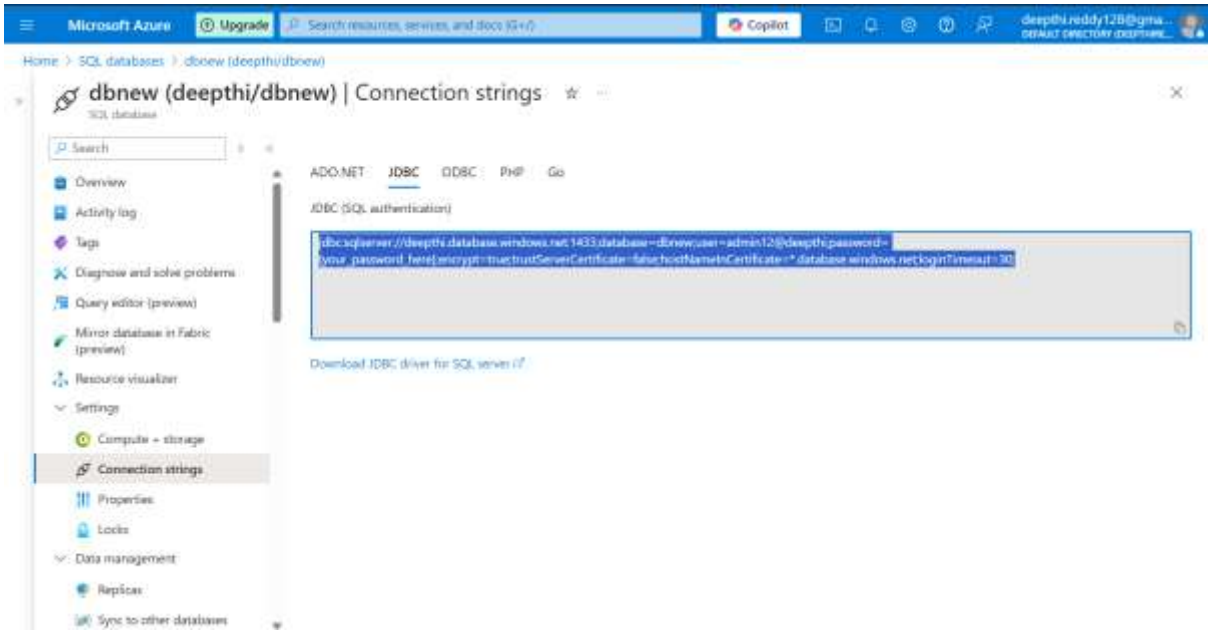
- Build a data pipeline to extract data from an SQL database, clean it by removing duplicates, and store it in Azure Data Lake Storage (ADLS) Gen2 using Databricks.
- Create SCD TYPE 1 logic using the Delta Tables in Databricks

Extract Data from SQL Database

Create a connection to SQL database using JDBC

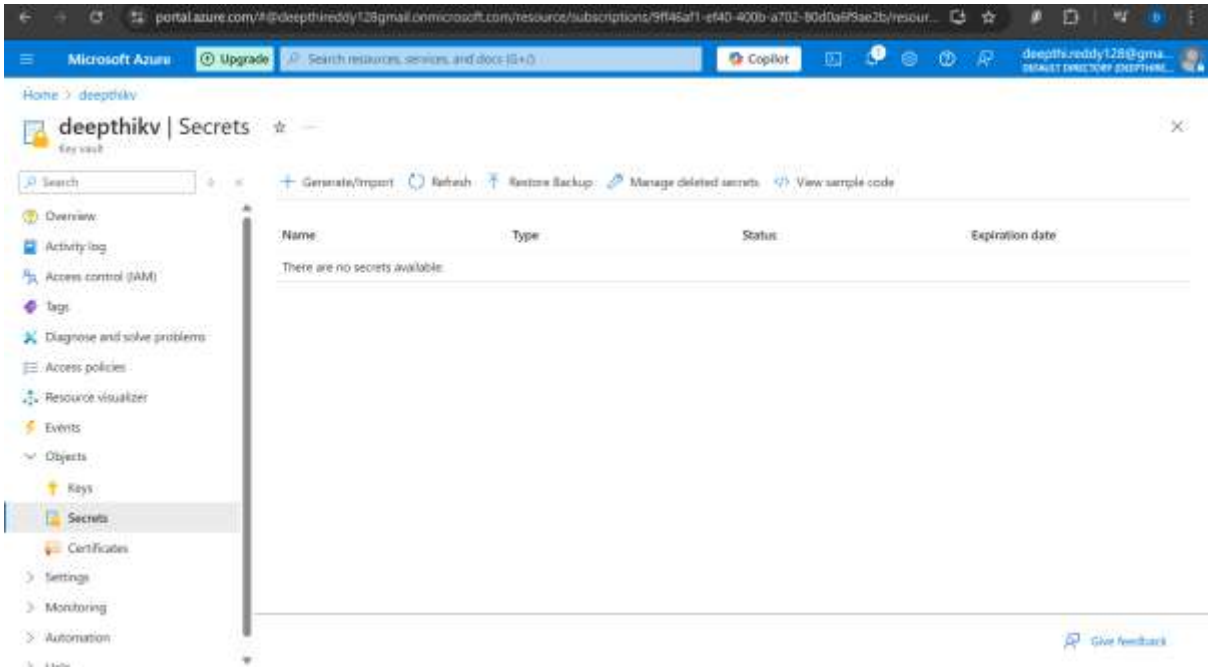
Go to Azure SQL database-> Settings-> Connection strings -> JDBC and get the data from there, like username, password and URL

jdbc:sqlserver://deepthi.database.windows.net:1433;database=dbnew;user=admin12@deepthi;password={your_password_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;

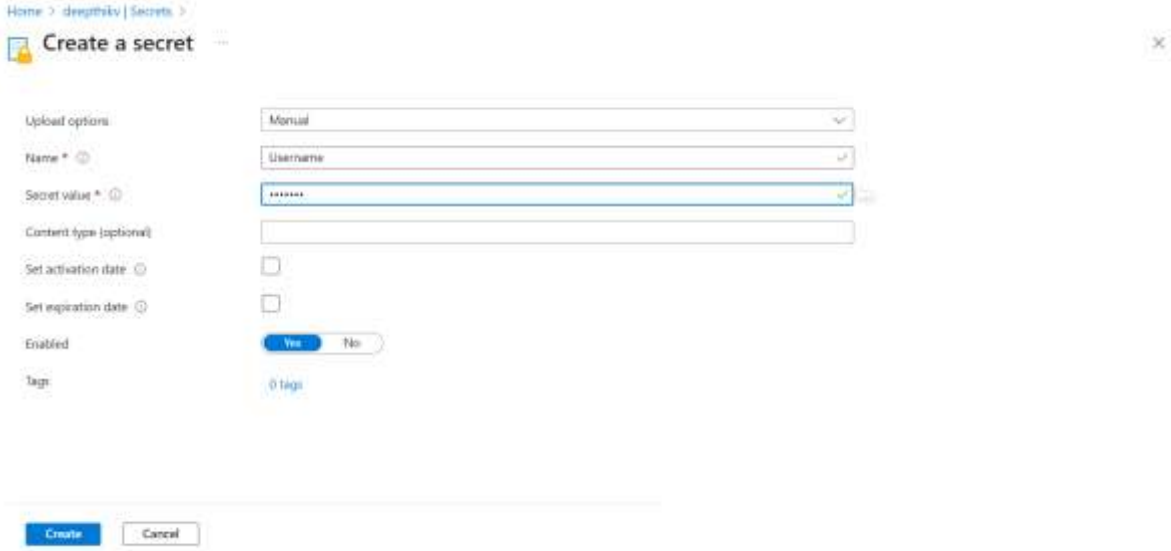


Now got to key vault to generate keys using these details

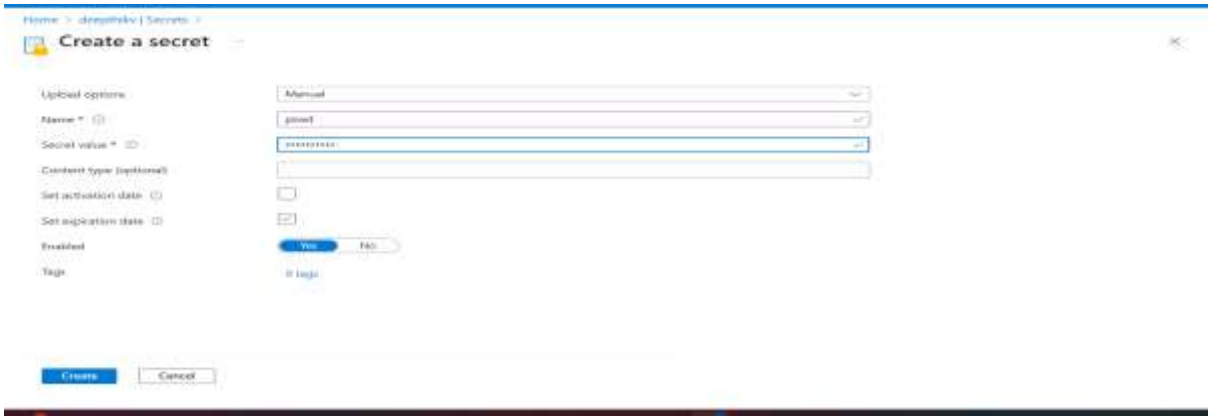
Azure key vault-> Objects-> Secrets -> Generate/Import



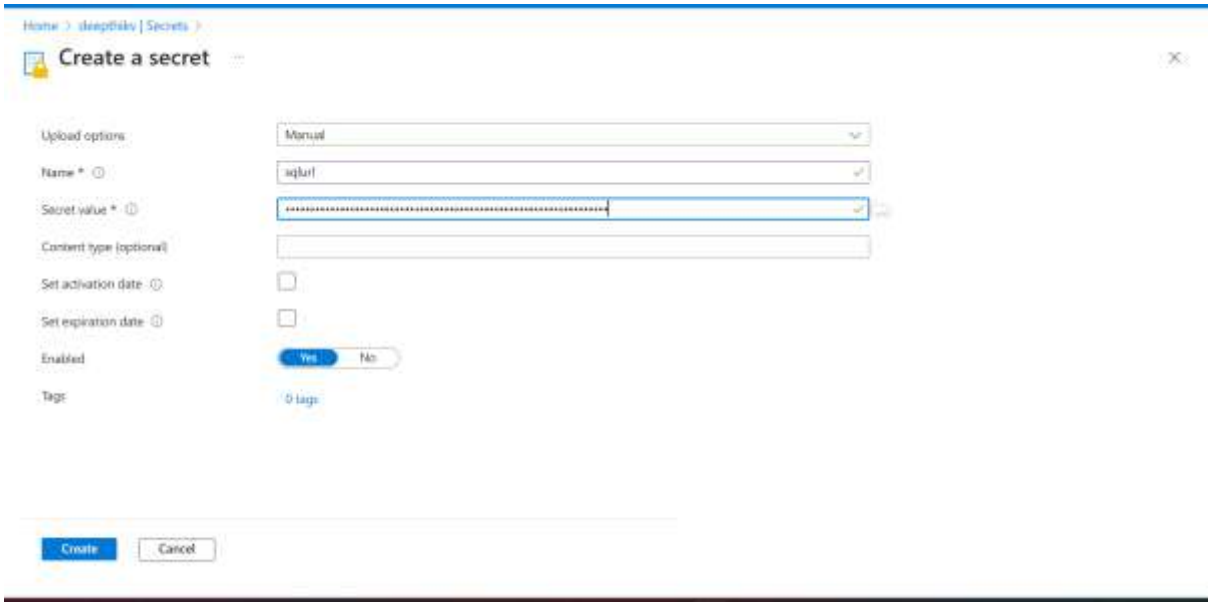
Create Username secret first



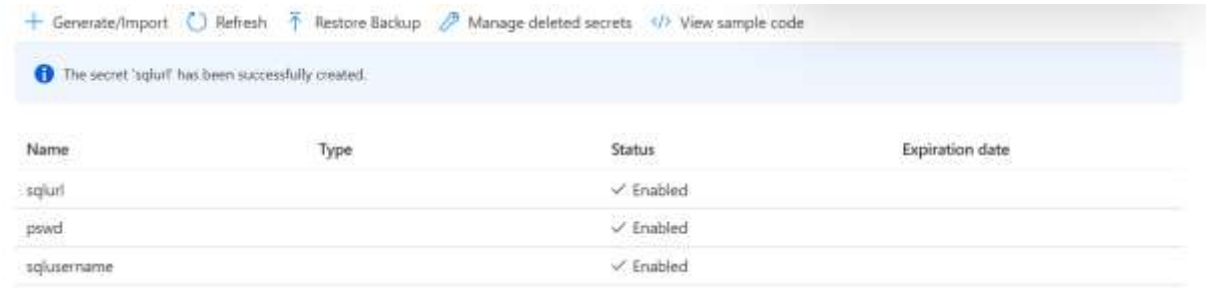
Create Password secret



Create secret for URL

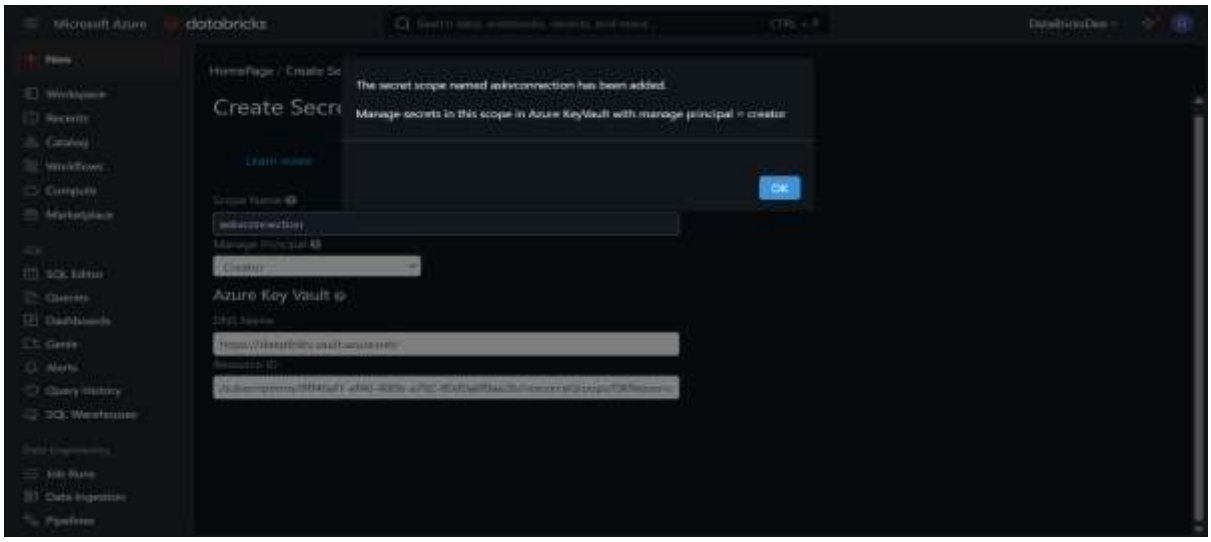


All the secrets are created



Now go to data bricks Scope and create a scope

Using https://adb-4390551724307171.11.azuredatabricks.net/?o=4390551724307171#secrets/createScope



Now check if the scope is created correctly or not with dbutils.secrets.listScope() command

```
dbutils.secrets.listScopes()
```

```
[SecretScope(name='askvconnection')]
```

```
dbutils.secrets.list('askvconnection')
```

```
[SecretMetadata(key='pswd'),
 SecretMetadata(key='sqlurl'),
 SecretMetadata(key='sqlusername')]
```

Create a table in SQL DB to perform the tasks

```
create table dbo.credit(
CID int, Name varchar(20), C_Score int, C_type varchar(20)

)
```

```
Messages
Commands completed successfully.

Completion time: 2025-03-31T14:12:35.6514550-04:00
```

Insert values into the table

```
insert into dbo.credit values (1,'Deepthi',989,'Visa')
insert into dbo.credit values (2,'Rahul',1124,'Visa')
insert into dbo.credit values (3,NULL,789,'Visa')
insert into dbo.credit values (4,'Shirya',1124,'Visa')
insert into dbo.credit values (5,'Priya',NULL,'Visa')
insert into dbo.credit values (6,'Mounika',989,'Visa')
insert into dbo.credit values (7,NULL,678,'Visa')
```

```
Messages
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
```

Check the values

```
select * from dbo.credit
```

100 %

Results | Messages

	CID	Name	C_Score	C_type
1	1	Deepthi	989	Visa
2	1	Deepthi	989	Visa
3	2	Rahul	1124	Visa
4	3	NULL	789	Visa
5	4	Shirya	1124	Visa
6	5	Priya	NULL	Visa
7	6	Mounika	989	Visa
8	7	NULL	678	Visa

Now create a connection from SQL to Databricks using below code

```
df = (spark.read
.format("jdbc")
.option("url",  dbutils.secrets.get(scope = "askvconnection", key = "sqlurl") )
.option("dbtable", "dbo.credit")
.option("user", dbutils.secrets.get(scope = "askvconnection", key = "sqlusername"))
.option("password",  dbutils.secrets.get(scope = "askvconnection", key = "pswd") )
.load() )
```

Here we are using our scope and key and creating a connection between SQL DB and Data bricks and reading the data from table **credit** which we created.

Just now (2s) 4 Python

```
df = (spark.read
.format("jdbc")
.option("url",  dbutils.secrets.get(scope = "askvconnection", key = "sqlurl") )
.option("dbtable", "dbo.credit")
.option("user", dbutils.secrets.get(scope = "askvconnection", key = "sqlusername"))
.option("password",  dbutils.secrets.get(scope = "askvconnection", key = "pswd") )
.load()
)
```

df: pyspark.sql.dataframe.DataFrame = [CID: integer, Name: string ... 2 more fields]

Get data from data frame using display(df)

Just now (5s) 5 Python

```
display(df)
```

(1) Spark Jobs

Table

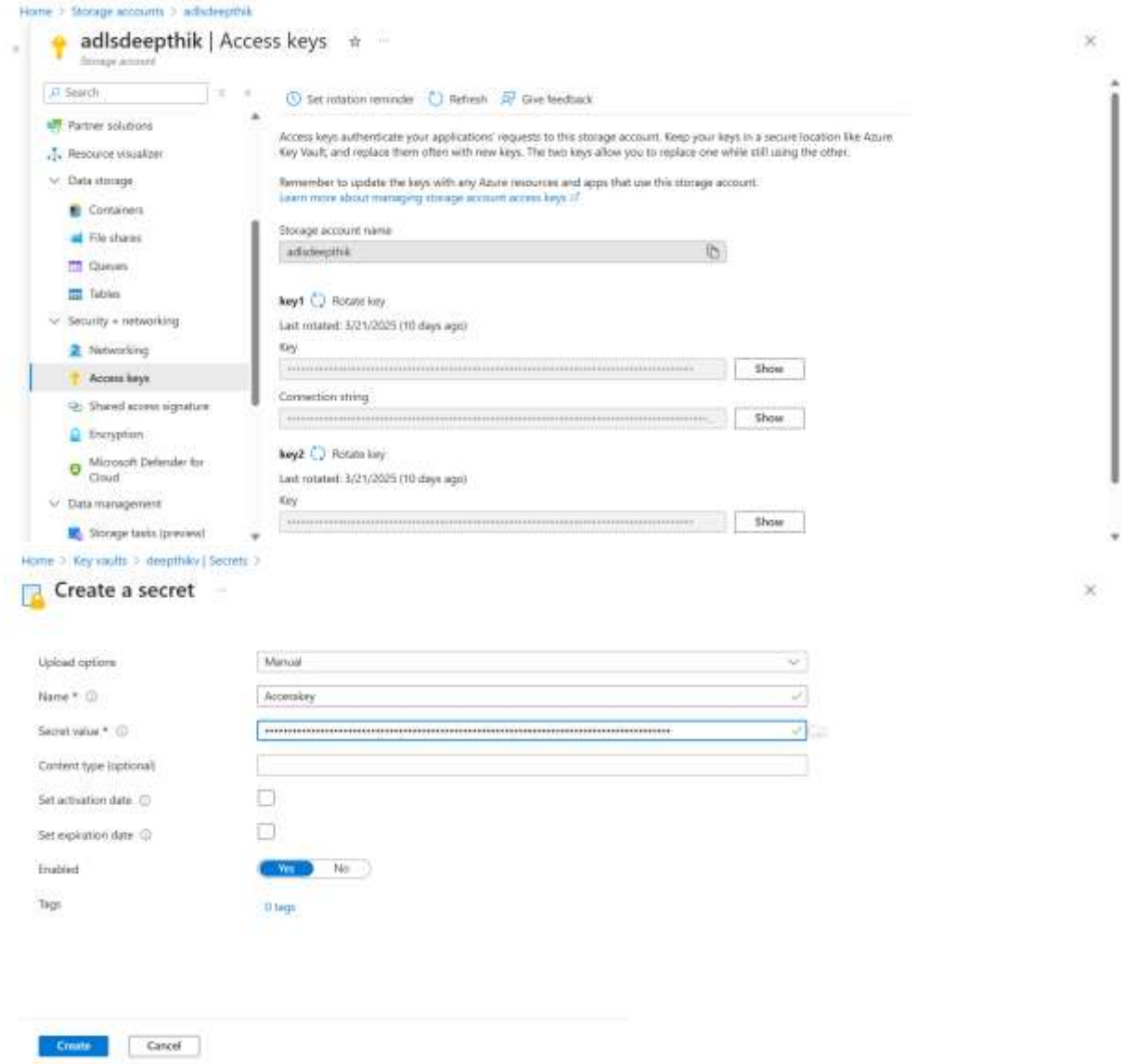
	CID	Name	C_Score	C_type
1	1	Deepthi	989	Visa
2	1	Deepthi	989	Visa
3	2	Rahul	1124	Visa
4	3	null	789	Visa
5	4	Shirya	1124	Visa
6	5	Priya	null	Visa
7	6	Mounika	989	Visa
8	7	null	678	Visa

8 rows | 5.04s runtime Refreshed now

Set Up ADLS Gen2 Mount Point

To create a mount to ADLS storage create secret keys using access key to get connected to Data bricks

Go to ADLS -> Security + Networking -> Access key and copy the access key to use it in Key Vault



Access key scope is also created

Now create a mount for ADLS using this scope and key

```
dbutils.fs.mount(
  source = "wasbs://container1@adlsdeepthik.blob.core.windows.net",
  mount_point = "/mnt/container1",
```

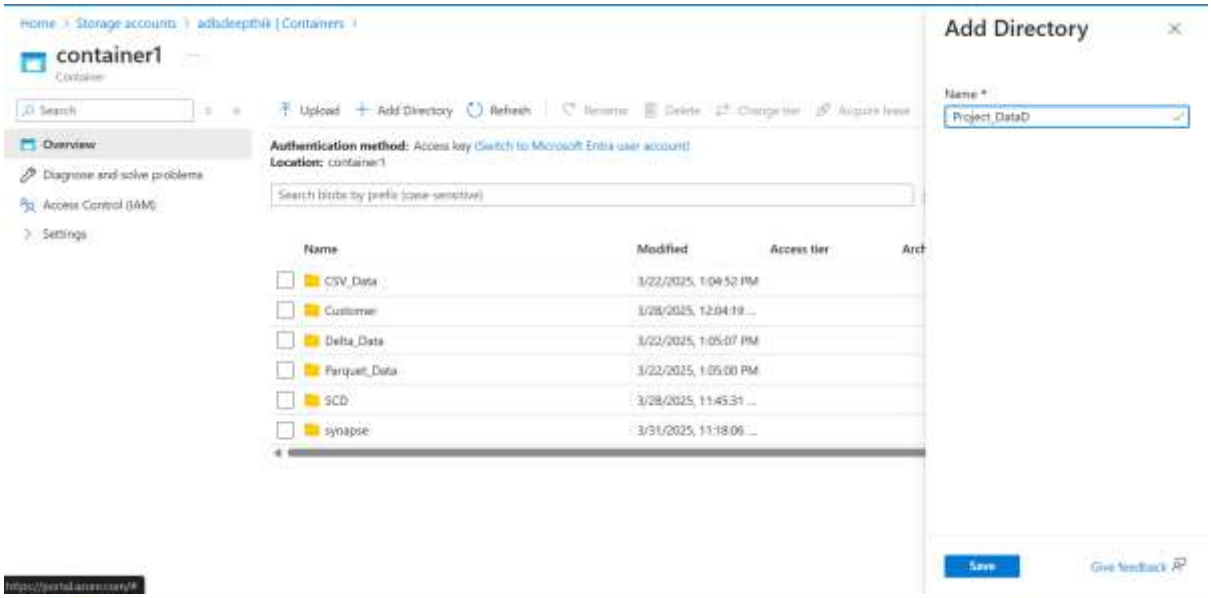
extra_configs = {"fs.azure.account.key.adlsdeepthik.blob.core.windows.net":dbutils.secrets.get(scope = "askvconnection", key = "Accesskey")})

```
dbutils.fs.mount(
  source = "wasbs://container1@adlsdeepthik.blob.core.windows.net",
  mount_point = "/mnt/container1",
  extra_configs = {"fs.azure.account.key.adlsdeepthik.blob.core.windows.net":dbutils.secrets.get(scope =
  "askvconnection", key = "Accesskey")})
```

```
True
```

Now write the data frame which we created to ADLS container

Before that create a folder in storage account to store that data




```
dbutils.fs.ls("/mnt/container1")
```

```
[FileInfo(path='dbfs:/mnt/container1/CSV_Data/', name='CSV_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Customer/', name='Customer/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Delta_Data/', name='Delta_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Parquet_Data/', name='Parquet_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Project_DataD/', name='Project_DataD/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/SCD /', name='SCD /', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/synapse/', name='synapse/', size=0, modificationTime=0)]
```

Now write the data to ADLS

```
df.write.format("csv").option("header", "true").mode("overwrite").save("/mnt/container1/Project_DataD/")
```

(1) Spark Jobs

Read the file which is added to ADLS

```
df_ADSL = spark.read.format("csv").option("header", "true").load("/mnt/container1/Project_DataD/")
```

(1) Spark Jobs

```
display(df_ADSL)
```

(1) Spark Jobs

Table

	CID	Name	C_Score	C_type
1	1	Deepthi	989	Visa
2	1	Deepthi	989	Visa
3	2	Rahul	1124	Visa
4	3	null	789	Visa
5	4	Shirya	1124	Visa
6	5	Priya	null	Visa
7	6	Mounika	989	Visa
8	7	null	678	Visa

8 rows | 0.47s runtime

Refreshed now

Remove nulls and duplicates

Removing NULLS, this function removes all the NULLS from the data frame

```
df_ADSL=df_ADSL.na.drop()
```

df_ADSL: pyspark.sql.dataframe.DataFrame = [CID: string, Name: string ... 2 more fields]

```
display(df_ADSL)
```

(1) Spark Jobs

Table

	CID	Name	C_Score	C_type
1	1	Deepthi	989	Visa
2	1	Deepthi	989	Visa
3	2	Rahul	1124	Visa
4	4	Shirya	1124	Visa
5	6	Mounika	989	Visa

5 rows | 0.45s runtime

Refreshed now

Now remove Duplicates, below function remove all duplicate records

```
df_ADSL= df_ADSL.dropDuplicates()
display(df_ADSL)
```

(2) Spark Jobs

df_ADSL: pyspark.sql.dataframe.DataFrame = [CID: string, Name: string ... 2 more fields]

Table

	CID	Name	C_Score	C_type
1	2	Rahul	1124	Visa
2	1	Deepthi	989	Visa
3	6	Mounika	989	Visa
4	4	Shirya	1124	Visa

4 rows | 1.23s runtime

Refreshed now

Now writing this new data frame to different folder which we can use for further tasks

```
Just now (3s)                                                    15
df_ADSL.write.format("csv").option("header", "true").mode("overwrite").save("/mnt/container1/cleaned_Data/")
(2) Spark Jobs
```

Now read and display the cleaned data

```
Just now (1s)                                                    16                                                    Python
df_clean = spark.read.format("csv").option("header", "true").load("/mnt/container1/cleaned_Data/")
display(df_clean)
(2) Spark Jobs
df_clean: pyspark.sql.dataframe.DataFrame = [CID: string, Name: string ... 2 more fields]
```

Table +

	CID	Name	C_Score	C_type
1	2	Rahul	1124	Visa
2	1	Deepthi	989	Visa
3	6	Mounika	989	Visa
4	4	Shirya	1124	Visa

Create SCD type 1 Logic using the cleaned data

Now using Data bricks community edition to perform SCD type 1

Create a mount to ADLS storage account in data bricks community edition

```
03:03 PM (16s)                                                    1
dbutils.fs.mount(
    source = "wasbs://container1@adlsdeepthik.blob.core.windows.net",
    mount_point = "/mnt/container1",
    extra_configs = {"fs.azure.account.key.adlsdeepthik.blob.core.windows.net":"aG5rHQ12QMCkYf0qfqpesBNpAXA5qqY/JvHi/
    I3zrqUy6GObRWji+5yYna0BXCReGeQeIX09JDHU+AStdzKfrQ==""})
Out[1]: True
```

```
4 minutes ago (3s)                                                    2
dbutils.fs.ls("/mnt/container1")
Out[2]: [FileInfo(path='dbfs:/mnt/container1/CSV_Data/', name='CSV_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Customer/', name='Customer/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Delta_Data/', name='Delta_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Parquet_Data/', name='Parquet_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/Project_DataD/', name='Project_DataD/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/SCD /', name='SCD /', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/cleaned_Data/', name='cleaned_Data/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/mnt/container1/synapse/', name='synapse/', size=0, modificationTime=0)]
```

Now create a Delta Table

```sql
%sql
create table cleanData(
C_id int,
C_name string,
score string,
C_type bigint,
C_hash_key bigint,
created_by string,
created_date timestamp,
updated_by string,
updated_date timestamp
)
using delta
location "/mnt/container1/SCD_Project"
```

(4) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame

OK

This result is stored as `_sqldf` and can be used in other Python cells.

	_$azuretmpfolder$	3/31/2025, 2:36:24 PM
	cleaned_Data	3/31/2025, 2:52:49 PM
	CSV_Data	3/22/2025, 1:04:52 PM
	Customer	3/28/2025, 12:04:19 ...
	Delta_Data	3/22/2025, 1:05:07 PM
	Parquet_Data	3/22/2025, 1:05:00 PM
	Project_DataD	3/31/2025, 2:42:56 PM
	SCD	3/28/2025, 11:45:31 ...
	SCD_Project	3/31/2025, 3:10:08 PM
	synapse	3/31/2025, 11:18:06 ...

Now read the clean data and assign it to a data frame

```python
df_src=spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/mnt/container1/cleaned_Data/")
display(df_src)
```

(3) Spark Jobs

df_src: pyspark.sql.dataframe.DataFrame = [CID: integer, Name: string ... 2 more fields]

Table

	CID	Name	C_Score	C_type
1	2	Rahul	1124	Visa
2	1	Deepthi	989	Visa
3	6	Mounika	989	Visa
4	4	Shirya	1124	Visa

Now create hash key for each record

Just now (2s) 5 Python

```python
from pyspark.sql.functions import *
df_src1=df_src.withColumn("hash_key",crc32(concat(*df_src.columns)))
display(df_src1)
```

(1) Spark Jobs

df_src1: pyspark.sql.dataframe.DataFrame = [CID: integer, Name: string ... 3 more fields]

Table

	CID	Name	C_Score	C_type	hash_key
1	2	Rahul	1124	Visa	3433401949
2	1	Deepthi	989	Visa	2651650240
3	6	Mounika	989	Visa	1503936933
4	4	Shirya	1124	Visa	29636849

4 rows | 1.63s runtime Refreshed now

Just now (5s) 6 Python

```python
from delta.tables import DeltaTable
dbtable = DeltaTable.forPath(spark, "/mnt/container1/SCD_Project/")
dbtable.toDF().show()
```

(1) Spark Jobs

```
+----+------+-----+------+----------+----------+------------+----------+------------+
|C_id|C_name|score|C_type|C_hash_key|created_by|created_date|updated_by|updated_date|
+----+------+-----+------+----------+----------+------------+----------+------------+
+----+------+-----+------+----------+----------+------------+----------+------------+
```

Just now (3s) 7 Python

```python
df_src1=df_src1.alias("src").join(dbtable.toDF().alias("tgt"), ((col("src.CID") == col("tgt.C_id")) & (col("src.
hash_key") == col("tgt.C_hash_key"))), "anti").select(col("src.*"))
df_src1.show()
```

(1) Spark Jobs

df_src1: pyspark.sql.dataframe.DataFrame = [CID: integer, Name: string ... 3 more fields]

```
+---+-------+-------+------+----------+
|CID|   Name|C_Score|C_type|  hash_key|
+---+-------+-------+------+----------+
|  2|  Rahul|   1124|  Visa|3433401949|
|  1|Deepthi|    989|  Visa|2651650240|
|  6|Mounika|    989|  Visa|1503936933|
|  4| Shirya|   1124|  Visa|  29636849|
+---+-------+-------+------+----------+
```

1 minute ago (19s) 8

```python
dbtable.alias("tgt").merge(df_src1.alias("src"),"tgt.C_id = src.CID")\
    .whenMatchedUpdate(set={"tgt.C_id":"src.CID","tgt.C_name":"src.NAME","tgt.score":"src.C_Score","tgt.C_type":"src.
    C_type","tgt.C_hash_key":"src.hash_key","tgt.updated_date":current_timestamp(),"tgt.updated_by":lit
    ("databricks")})\
        .whenNotMatchedInsert(values={"tgt.C_id":"src.CID","tgt.C_name":"src.NAME","tgt.score":"src.C_Score","tgt.
        C_type":"src.C_type","tgt.C_hash_key":"src.hash_key","tgt.created_date":current_timestamp(),"tgt.
        created_by":lit("databricks"),"tgt.updated_date":current_timestamp(),"tgt.updated_by":lit("databricks")}).
        execute()
```

(8) Spark Jobs

Just now (8s) 9 Python

```python
display(spark.read.format("delta").option("header","true").load("/mnt/container1/SCD_Project/"))
```

(2) Spark Jobs

Table

	C_id	C_name	score	C_type	C_hash_key	created_by	created_date
1	2	Rahul	1124	null	3433401949	databricks	2025-03-31T19:21:06.632+00:0
2	1	Deepthi	989	null	2651650240	databricks	2025-03-31T19:21:06.632+00:0
3	6	Mounika	989	null	1503936933	databricks	2025-03-31T19:21:06.632+00:0
4	4	Shirya	1124	null	29636849	databricks	2025-03-31T19:21:06.632+00:0

4 rows | 8.22s runtime Refreshed now