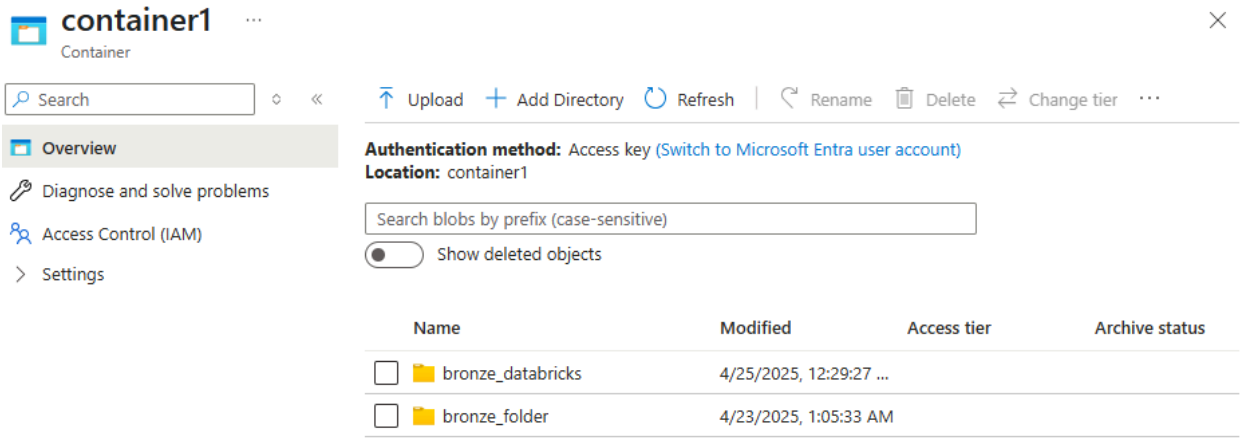


# Project 2

1. For the **bronze layer**, we have to store 5 files from the local machine to storage account. For that I have created a new directory named **bronze\_databricks**.

[Home](#) > [Storage accounts](#) > [sabrpadls | Containers](#) >



container1

Container

Search

Upload Add Directory Refresh Rename Delete Change tier

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

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

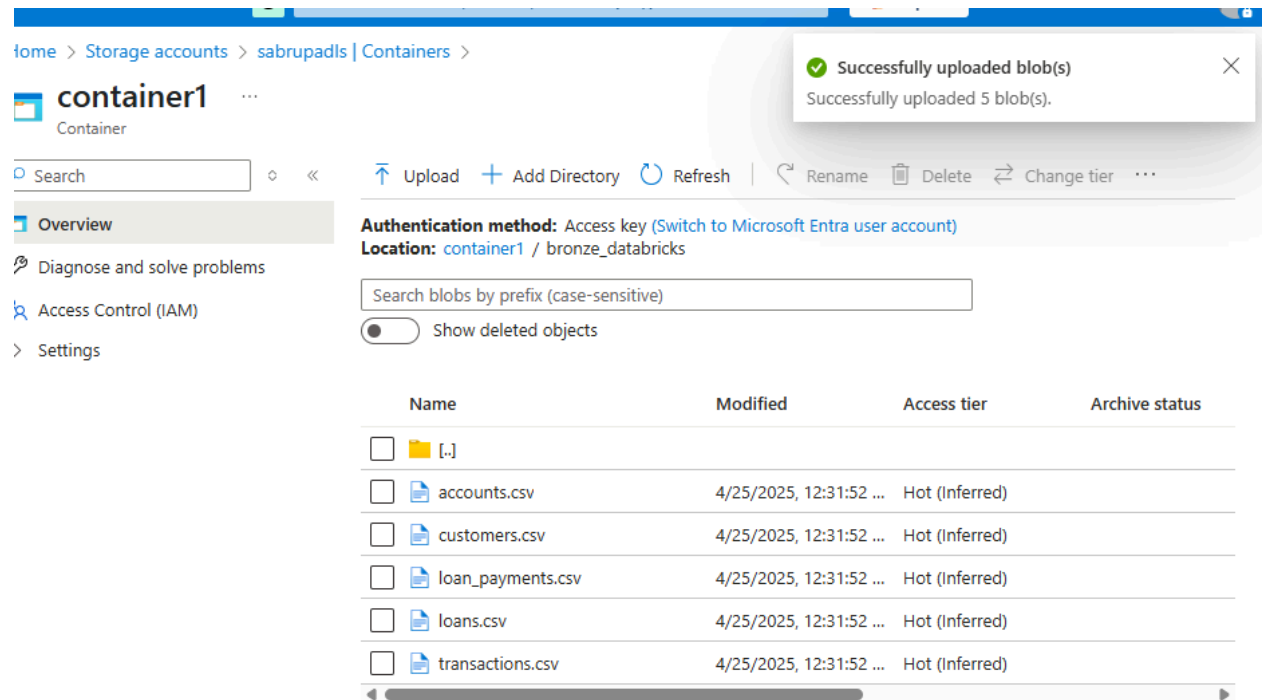
Location: container1

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name              | Modified                | Access tier | Archive status |
|-------------------|-------------------------|-------------|----------------|
| bronze_databricks | 4/25/2025, 12:29:27 ... |             |                |
| bronze_folder     | 4/23/2025, 1:05:33 AM   |             |                |

And uploaded all the files manually.



container1

Container

Search

Upload Add Directory Refresh Rename Delete Change tier

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

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

Location: container1 / bronze\_databricks

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name              | Modified                | Access tier    | Archive status |
|-------------------|-------------------------|----------------|----------------|
| [.]               | 4/25/2025, 12:31:52 ... |                |                |
| accounts.csv      | 4/25/2025, 12:31:52 ... | Hot (Inferred) |                |
| customers.csv     | 4/25/2025, 12:31:52 ... | Hot (Inferred) |                |
| loan_payments.csv | 4/25/2025, 12:31:52 ... | Hot (Inferred) |                |
| loans.csv         | 4/25/2025, 12:31:52 ... | Hot (Inferred) |                |
| transactions.csv  | 4/25/2025, 12:31:52 ... | Hot (Inferred) |                |

Now all the files are in csv format and in a single folder.

2. **The second step is the silver layer where I performed two tasks.** One is to fetch all each of the file and perform some transformation on them to clean and store in parquet format in a new directory.

Also have to join all the files based on the primary key and **store in delta** format.

To establish the connection between ADLS and the databricks, I used **service principle**:

- **First of all, a new registration is created under Microsoft Entra ID-> Manage-> App Registration**

The screenshot shows the Microsoft Entra ID portal for an application named 'sabrapp'. The left sidebar contains navigation links: Overview, Quickstart, Integration assistant, Diagnose and solve problems, Manage (with sub-links for Branding & properties, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners, Roles and administrators, and Manifest), and Support + Troubleshooting. The main content area has a search bar and action buttons (Delete, Endpoints, Preview features). A warning banner states: 'A certificate or secret is expiring soon. Create a new one'. Below this is the 'Essentials' section with key-value pairs: Display name (sabrapp), Application (client) ID (7d90342b-0062-458c-9b2b-947fd0483cdf), Object ID (3c6b228f-1d81-44f9-9de3-220a8b956b7c), Directory (tenant) ID (a7dda661-784c-41b1-aa59-db8a4cc0c8d3), and Supported account types (My organization only). To the right of these are links for Client credentials (0 certificate, 1 secret), Redirect URIs (Add a Redirect URI), Application ID URI (Add an Application ID URI), and Managed application in local directory (sabrapp). A blue information banner at the bottom of the Essentials section states: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more'. At the bottom of the page, there is a section titled 'Build your application with the Microsoft identity platform' with a description: 'The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. Learn more'.

Home > App registrations >

sabrapp

Search

Delete Endpoints Preview features

Overview

Quickstart

Integration assistant

Diagnose and solve problems

Manage

Branding & properties

Authentication

Certificates & secrets

Token configuration

API permissions

Expose an API

App roles

Owners

Roles and administrators

Manifest

Support + Troubleshooting

A certificate or secret is expiring soon. Create a new one →

Essentials

Display name  
sabrapp

Application (client) ID  
7d90342b-0062-458c-9b2b-947fd0483cdf

Object ID  
3c6b228f-1d81-44f9-9de3-220a8b956b7c

Directory (tenant) ID  
a7dda661-784c-41b1-aa59-db8a4cc0c8d3

Supported account types  
My organization only

Client credentials  
0 certificate, 1 secret

Redirect URIs  
Add a Redirect URI

Application ID URI  
Add an Application ID URI

Managed application in local directory  
sabrapp

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)

Get Started Documentation

## Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. [Learn more](#)

**As in the SC new-app I have created.**

- **Secondly, we need to copy Application ID.**

- After that under Certificates and Secrets, we have to create new client secret with custom expiry date.

Home / App registrations / sabrapp

## sabrapp | Certificates & secrets

Search Got feedback?

- Overview
- Quickstart
- Integration assistant
- Diagnose and solve problems
- Manage
  - Branding & properties
  - Authentication
  - Certificates & secrets**
  - Token configuration
  - API permissions
  - Expose an API
  - App roles
  - Owners
  - Roles and administrators
  - Manifest
- Support + Troubleshooting



Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) **Client secrets (1)** Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

| Description | Expires     | Value ⓘ  | Secret ID  |
|-------------|-------------|----------|--|
| sabrclient  | 5/10/2025 ⚠ | moW***** | 13c6c8b3-fbea-4d7...   |

- As here I have created “sabrclient”. And copy the value and the secret ID for further use while connecting.
- Further, key Vaults have to be created to hide the IDs

Home > Key vaults > ssabrkv

## ssabrkv | Secrets

Key vault

Search

+ Generate/Import Refresh Restore Backup Manage deleted secrets

| Name              | Type | Status    | Expiration date |
|-------------------|------|-----------|-----------------|
| applicationId     |      | ✓ Enabled |                 |
| secretclientValue |      | ✓ Enabled |                 |
| secretId          |      | ✓ Enabled |                 |
| tenantId          |      | ✓ Enabled |                 |

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Access policies
- Resource visualizer
- Events
- Objects
- Keys
- Secrets**
- Certificates
- Settings
- Access configuration

As I have created “ssabrkv” key Vault with app-id, app-value and secretid.

adp-3974609196074706.6.azuredatabricks.net/To=3974609196074706#secrets/createScope

Microsoft Azure databricks

Search data, notebooks, recents, and more...

+ New

- Workspace
- Recents
- Catalog
- Workflows
- Compute
- Marketplace
- SQL
- SQL Editor
- Queries
- Dashboards
- Genie
- Alerts
- Query History
- SQL Warehouses
- Data Engineering
- Job Runs

HomePage / Create Secret Scope

### Create Secret Scope

Cancel Create

A store for secrets that is identified by a name and backed by a specific store type. [Learn more](#)

Scope Name

Manage Principal

Creator

Azure Key Vault

DNS Name

https://xxx.vault.azure.net/

Resource ID

/subscriptions/xxxxx/...

Here DNS Name and the Resource ID we can get from the “Properties” tab

in Key Vault:

|                               |  |
|-------------------------------|--|
| Name                          | ssabrkv  |
| Sku (Pricing tier)            | Standard   |
| Location                      | canadacentral  |
| Vault URI                     | https://ssabrkv.vault.azure.net/   |
| Resource ID                   | /subscriptions/d74c8e4f-2be5-4   |
| Subscription ID               | d74c8e4f-2be5-4e1d-9321-ecca   |
| Subscription Name             | Azure subscription 1   |
| Directory ID                  | a7dda661-784c-41b1-aa59-db8  |
| Directory Name                | Default Directory  |
| <b>Soft-delete</b>            | <b>Soft delete has been enabled on</b>   |
| Days to retain deleted vaults | 90   |
| Purge protection              | <input checked="" type="radio"/> Disable purge protection (all<br>purged during retention period)<br><input type="radio"/> Enable purge protection (enfo<br>for deleted vaults and vault o |

- Then in databricks run the following command to check the scope:

▶

✓ 12:08 AM (<1s)

1

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

```
[SecretScope(name='sabradscope'),  
SecretScope(name='sabrscope'),  
SecretScope(name='sabrscope2')]
```

- Now, it's time to connect the storage account with the databricks using mounting. For that I have created a mount:

```
12:17 AM (14s) 3 Python

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": "7d90342b-0062-458c-9b2b-947fd0483cdf",
    "fs.azure.account.oauth2.client.secret":
    "mow8Q~XNrZC1dGpj1hwOXCL2CuP5dB102aI9yc9m",
    "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/
a7dda661-784c-41b1-aa59-db8a4cc0c8d3/oauth2/token"
}

dbutils.fs.mount(
    source = "abfss://container1@sabrupadls.dfs.core.windows.net/",
    mount_point = "/mnt/container1",
    extra_configs = configs
)

True
```

- The below SC showing all the files in bronze\_databricks folder:

```
Just now (4s) 4 Python

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

[FileInfo(path='dbfs:/mnt/container1/bronze_databricks/accounts.csv', name='accounts.cs
v', size=2331, modificationTime=174555512000),
  FileInfo(path='dbfs:/mnt/container1/bronze_databricks/customers.csv', name='customers.c
sv', size=4603, modificationTime=174555512000),
  FileInfo(path='dbfs:/mnt/container1/bronze_databricks/loan_payments.csv', name='loan_pa
yments.csv', size=2613, modificationTime=174555512000),
  FileInfo(path='dbfs:/mnt/container1/bronze_databricks/loans.csv', name='loans.csv', siz
e=2340, modificationTime=174555512000),
  FileInfo(path='dbfs:/mnt/container1/bronze_databricks/transactions.csv', name='transact
ions.csv', size=3513, modificationTime=174555512000)]
```

- Here I created 5 dataframes to store the data from the 5 files:

```
df_accounts=spark.read.format("csv").option("header", "true").option
("inferSchema", "true").load("/mnt/container1/bronze_folder/accounts.csv")

df_customers=spark.read.format("csv").option("header", "true").option
("inferSchema", "true").load("/mnt/container1/bronze_folder/customers.csv")

df_loan_payments=spark.read.format("csv").option("header", "true").option
("inferSchema", "true").load("/mnt/container1/bronze_folder/loan_payments.csv")

df_loans=spark.read.format("csv").option("header", "true").option("inferSchema",
"true").load("/mnt/container1/bronze_folder/loans.csv")

df_transactions=spark.read.format("csv").option("header", "true").option
("inferSchema", "true").load("/mnt/container1/bronze_folder/transactions.csv")
```

▶ (10) Spark Jobs

- ▶ df\_accounts: pyspark.sql.dataframe.DataFrame = [account\_id: integer, customer\_id: integer ... 2 more fields]
- ▶ df\_customers: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, first\_name: string ... 5 more fields]
- ▶ df\_loan\_payments: pyspark.sql.dataframe.DataFrame = [payment\_id: integer, loan\_id: integer ... 2 more fields]
- ▶ df\_loans: pyspark.sql.dataframe.DataFrame = [loan\_id: integer, customer\_id: integer ... 3 more fields]
- ▶ df\_transactions: pyspark.sql.dataframe.DataFrame = [transaction\_id: integer, account\_id: integer ... 3 more fields]

+ Code    + Text

- If I display customers dataframe, it has null values:

|    |  |          |                  |                |      |        |
|----|--|----------|------------------|----------------|------|--------|
| 74 |  | Graham   | 7373 Oak Dr      | Bala           | ON   | P0C0A1 |
| 75 |  | Sullivan | 7474 Pine Rd     | Bracebridge    | ON   | P1L0A1 |
| 76 |  | Wallace  | 7575 Birch Blvd  | Huntsville     | ON   | P1H0A1 |
| 77 |  | Woods    | 7676 Spruce Ln   | Burks Falls    | ON   | P0A0A1 |
| 78 |  | Cole     | 7777 Fir St      | Sundridge      | ON   | P0A0A1 |
| 79 |  | West     | 7878 Redwood Dr  | South River    | ON   | P0A0A1 |
| 80 |  | Jordan   | 7979 Cypress Ave | North Bay      | ON   | P1B0A1 |
| 81 |  | Owens    | 8080 Willow Rd   | Mattawa        | ON   | P0H0A1 |
| 82 |  | Reynolds | 8181 Poplar St   | Sturgeon Falls | ON   | P2B0A1 |
| 83 |  | Fisher   | 8282 Ash Blvd    | Verner         | ON   | P0H0A1 |
| 84 |  | Ellis    | 8383 Beech Dr    | Field          | ON   | P0H0A1 |
| 85 |  | Harrison | 8484 Cedar Ln    | Temagami       | ON   | P0H0A1 |
| 86 |  | Gibson   | 8585 Elm St      | New Liskeard   | ON   | P0J0A1 |
| 87 |  | McDonald | 8686 Maple Ave   | Haileybury     | null | null   |



- The following command removed the null values. Actually it removes the whole row.

```
df_customers_clean = df_customers.na.drop()
display(df_customers_clean)
```

01:15 AM (<1s) 7 Python

```
df_customers_clean = df_customers.na.drop()
display(df_customers_clean)
```

▶ (1) Spark Jobs

▶ df\_customer\_notnull: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, first\_name: string ... more fields]

|    | customer_id | first_name | last_name | address          | city           |
|----|-------------|------------|-----------|------------------|----------------|
| 73 | 73          | Andrew     | Hamilton  | 7272 Maple Ave   | Gravenhurst    |
| 74 | 74          | Harper     | Graham    | 7373 Oak Dr      | Bala           |
| 75 | 75          | Joshua     | Sullivan  | 7474 Pine Rd     | Bracebridge    |
| 76 | 76          | Evelyn     | Wallace   | 7575 Birch Blvd  | Huntsville     |
| 77 | 77          | Daniel     | Woods     | 7676 Spruce Ln   | Burks Falls    |
| 78 | 78          | Abigail    | Cole      | 7777 Fir St      | Sundridge      |
| 79 | 79          | James      | West      | 7878 Redwood Dr  | South River    |
| 80 | 80          | Emily      | Jordan    | 7979 Cypress Ave | North Bay      |
| 81 | 81          | Michael    | Owens     | 8080 Willow Rd   | Mattawa        |
| 82 | 82          | Elizabeth  | Reynolds  | 8181 Poplar St   | Sturgeon Falls |
| 83 | 83          | David      | Fisher    | 8282 Ash Blvd    | Verner         |
| 84 | 84          | Sophia     | Ellis     | 8383 Beech Dr    | Field          |
| 85 | 85          | John       | Harrison  | 8484 Cedar Ln    | Temagami       |
| 86 | 86          | Olivia     | Gibson    | 8585 Elm St      | New Liskeam    |

The below SC showing the commands to remove all the null values from all the 5 dataframes.

▶

▼

✓

Just now (<1s)

9

Python

🗑️

🌟

🔍

⋮

```
df_customers_clean = df_customers.na.drop()
df_accounts_clean = df_accounts.na.drop()
df_loan_payments_clean = df_loan_payments.na.drop()
df_loans_clean = df_loans.na.drop()
df_transactions_clean = df_transactions.na.drop()
```

▶

📄

df\_accounts\_clean: pyspark.sql.dataframe.DataFrame = [account\_id: integer, customer\_id: integer ... 2 more fields]

▶

📄

df\_customers\_clean: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, first\_name: string ... 5 more fields]

▶

📄

df\_loan\_payments\_clean: pyspark.sql.dataframe.DataFrame = [payment\_id: integer, loan\_id: integer ... 2 more fields]

▶

📄

df\_loans\_clean: pyspark.sql.dataframe.DataFrame = [loan\_id: integer, customer\_id: integer ... 3 more fields]

▶

📄

df\_transactions\_clean: pyspark.sql.dataframe.DataFrame = [transaction\_id: integer, account\_id: integer ... 3 more fields]

Also to remove the duplicates:

▶

▼

✓

1 minute ago (<1s)

6

Python

🗑️

🌟

🔍

⋮

```
df_customers_clean = df_customers.na.drop().dropDuplicates()
df_accounts_clean = df_accounts.na.drop().dropDuplicates()
df_loan_payments_clean = df_loan_payments.na.drop().dropDuplicates()
df_loans_clean = df_loans.na.drop().dropDuplicates()
df_transactions_clean = df_transactions.na.drop().dropDuplicates()
```

▶

📄

df\_accounts\_clean: pyspark.sql.dataframe.DataFrame = [account\_id: integer, customer\_id: integer ... 2 more fields]

▶

📄

df\_customers\_clean: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, first\_name: string ... 5 more fields]

▶

📄

df\_loan\_payments\_clean: pyspark.sql.dataframe.DataFrame = [payment\_id: integer, loan\_id: integer ... 2 more fields]

▶

📄

df\_loans\_clean: pyspark.sql.dataframe.DataFrame = [loan\_id: integer, customer\_id: integer ... 3 more fields]

▶

📄

df\_transactions\_clean: pyspark.sql.dataframe.DataFrame = [transaction\_id: integer, account\_id: integer ... 3 more fields]

After cleaning the data, we have to write back to storage account but in parquet format.

Just now (6s)





8

Python

```
df_customers_clean.write.mode("overwrite").parquet("/mnt/container1/silver_databricks/customers")
df_accounts_clean.write.mode("overwrite").parquet("/mnt/container1/silver_databricks/accounts")
df_loan_payments_clean.write.mode("overwrite").parquet("/mnt/container1/silver_databricks/loan_payments")
df_loans_clean.write.mode("overwrite").parquet("/mnt/container1/silver_databricks/loans")
df_transactions_clean.write.mode("overwrite").parquet("/mnt/container1/silver_databricks/transactions")
```

(10) Spark Jobs

Here's the silver\_databricks folder:

|                          | Name  | Modified                | Access tier | Archive s |
|--------------------------|---|-------------------------|-------------|-----------|
| <input type="checkbox"/> |  bronze_databricks | 4/25/2025, 12:29:27 ... |             |           |
| <input type="checkbox"/> |  bronze_folder     | 4/23/2025, 1:05:33 AM   |             |           |
| <input type="checkbox"/> |  bronzeFiles       | 4/23/2025, 11:54:08 ... |             |           |
| <input type="checkbox"/> |  silver_databricks | 4/25/2025, 1:37:14 AM   |             |           |

All the 5 files in parquet format:

Search

Upload Add Directory Refresh | Rename Delete Change tier ...

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

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

Location: container1 / silver\_databricks

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name                                   | Modified              | Access tier | Archive status |
|--|-----------------------|-------------|----------------|
| <input type="checkbox"/> [..]          |                       |             |                |
| <input type="checkbox"/> accounts      | 4/25/2025, 1:37:15 AM |             |                |
| <input type="checkbox"/> customers     | 4/25/2025, 1:37:14 AM |             |                |
| <input type="checkbox"/> loan_payments | 4/25/2025, 1:37:16 AM |             |                |
| <input type="checkbox"/> loans         | 4/25/2025, 1:37:17 AM |             |                |
| <input type="checkbox"/> transactions  | 4/25/2025, 1:37:18 AM |             |                |

Upload Add Directory Refresh | Rename Delete Change tier ...

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

Location: container1 / silver\_databricks / accounts

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name  | Modified              | Access tier    | Archive status |
|---|-----------------------|----------------|----------------|
| <input type="checkbox"/> [..]                             |                       |                |                |
| <input type="checkbox"/> _committed_8361869868515931...   | 4/25/2025, 1:37:15 AM | Hot (Inferred) |                |
| <input type="checkbox"/> _started_8361869868515931629     | 4/25/2025, 1:37:15 AM | Hot (Inferred) |                |
| <input type="checkbox"/> _SUCCESS                         | 4/25/2025, 1:37:15 AM | Hot (Inferred) |                |
| <input type="checkbox"/> part-00000-tid-83618698685159... | 4/25/2025, 1:37:15 AM | Hot (Inferred) |                |

Next step is to join all the files on the common column and store in Delta format in ADLS storage account.

The steps are below:

1. As the customers, accounts and loans all have common column “customer\_id”. Thus, can be joined.
2. This joined df can be joined with transaction since both have account\_id common.
3. Lastly, loan\_payment can be joined on loan\_id.

The below Screen Short shows the commands:

```
# Now join with transactions on account_id
joined_df = joined_df.join(
    df_transactions_clean, on="account_id", how="inner"
)

# Join with loans on customer_id
joined_df = joined_df.join(
    df_loans_clean, on="customer_id", how="inner"
)

# Finally, join with loan_payments on loan_id
joined_df = joined_df.join(
    df_loan_payments_clean, on="loan_id", how="inner"
)
display(joined_df)
```

▶ (10) Spark Jobs

joined\_df: pyspark.sql.dataframe.DataFrame = [loan\_id: integer, customer\_id: integer ... 19 more fields]

|   | loan_id | customer_id | account_id | first_name  | last_name | address         | city           | state | zip    | account_type | balance |
|---|---------|-------------|------------|-------------|-----------|-----------------|----------------|-------|--------|--------------|---------|
| 1 | 30      | 32          | 30         | Sophia      | Morris    | 3131 Oak Dr     | Belleville     | ON    | K8N0A1 | Checking     | 3100.2  |
| 2 | 4       | 34          | 4          | Olivia      | Reed      | 3333 Birch Blvd | Orillia        | ON    | L3V0A1 | Checking     | 3000.2  |
| 3 | 26      | 25          | 26         | Daniel      | Campbell  | 2424 Willow Rd  | St. Catharines | ON    | L2R0A1 | Checking     | 2800.2  |
| 4 | 74      | 43          | 74         | Joseph      | Cox       | 4242 Cedar Ln   | Aurora         | ON    | L4G0A1 | Checking     | 7500.2  |
| 5 | 11      | 3           | 11         | Michael     | Johnson   | 789 Oak Dr      | Montreal       | QC    | H1A1A1 | Savings      | 1100.2  |
| 6 | 40      | 19          | 40         | Christopher | Baker     | 1818 Pine Rd    | Thunder Bay    | ON    | P7A0A1 | Checking     | 4100.2  |

Now this joined data we have to store in the form of Delta.

```
joined_df.write.format("delta").mode("overwrite").save("/mnt/container1/silver_databricks/joined_data")
```

▶ (13) Spark Jobs

Here's the successfully running command

Got joined\_data folder:

|                          | Name          | Modified               | Access tier | Archive status | Blob type |
|--------------------------|---------------|------------------------|-------------|----------------|-----------|
| <input type="checkbox"/> | accounts      | 4/25/2025, 1:37:15 AM  |             |                |           |
| <input type="checkbox"/> | customers     | 4/25/2025, 1:37:14 AM  |             |                |           |
| <input type="checkbox"/> | joined_data   | 4/27/2025, 12:27:59 AM |             |                |           |
| <input type="checkbox"/> | loan_payments | 4/25/2025, 1:37:16 AM  |             |                |           |
| <input type="checkbox"/> | loans         | 4/25/2025, 1:37:17 AM  |             |                |           |
| <input type="checkbox"/> | transactions  | 4/25/2025, 1:37:18 AM  |             |                |           |

Containers >

r1 ...

×

◦ ◀ [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:** [container1](#) / [silver\\_databricks](#) / [joined\\_data](#)

problems

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

|                          | Name  | Modified               | Access tier    | Archive status | Blob type  | Size     | Lease state   |
|--------------------------|---|------------------------|----------------|----------------|------------|----------|---------------|
| <input type="checkbox"/> | delta_log   | 4/27/2025, 12:27:59 AM |                |                |            |          | - ...         |
| <input type="checkbox"/> | part-00000-442fec65-e7bb-49d6-9891-4f1af34dcce1.c000.snappy.parquet | 4/27/2025, 12:28:02 AM | Hot (Inferred) |                | Block blob | 12.3 KiB | Available ... |

### Step 3: Storing the joined data as SCD Type 1

1. First of all, we have to create a SCD Type1 table.

▶ ▼ ✓ 12:18 AM (8s) 13 SQL 🗑️ ✨ ⌂ ⋮

```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.default.accounts_scdtype1 (
    customer_id INT,
    account_id INT,
    account_type STRING,
    balance FLOAT,
    hashkey BIGINT,
    createdby STRING,
    createddate TIMESTAMP,
    updatedby STRING,
    updateddate TIMESTAMP
)
USING DELTA
LOCATION '/mnt/container1/gold_folder/accounts_scdtype1';

SELECT * FROM hive_metastore.default.accounts_scdtype1;
```

▶ 📄 \_sqldf: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, account\_id: integer ... 7 more fields]

Table ▼ + 🔍 ⚙️ 📄

|                  | 1 <sup>2</sup> <sub>3</sub> customer_id | 1 <sup>2</sup> <sub>3</sub> account_id | A <sup>B</sup> <sub>C</sub> account_type | 1.2 balance | 1 <sup>2</sup> <sub>3</sub> hashkey |
|------------------|---|--|--|-------------|-------------------------------------|
| No rows returned |   |  |  |             |                                     |

2. Here's the delta table for accounts that we just created:

Container

Container

Search

UploadAdd DirectoryRefreshRenameDeleteChange tier

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

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

Location: container1 / gold\_databricks

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name              | Modified                | Access tier | Archive status |
|-------------------|-------------------------|-------------|----------------|
| [.]               |                         |             |                |
| accounts_scdtype1 | 4/29/2025, 12:18:50 ... |             |                |

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

Location: container1 / gold\_databricks / accounts\_scdtype1

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name       | Modified                | Access tier | Archive : |
|------------|-------------------------|-------------|-----------|
| [.]        |                         |             |           |
| _delta_log | 4/29/2025, 12:18:50 ... |             |           |

3. Adding Hashkey:



▶

✓ 2 minutes ago (1s)

16

Python

🗑️

🌟

🔍

⋮

```
# adding hash key
src_accounts = src_accounts.withColumn("hashkey", crc32(concat(*src_accounts.
columns)))
display(src_accounts)
```

▶ (1) Spark Jobs

▶ 📄 src\_accounts: pyspark.sql.dataframe.DataFrame = [account\_id: integer, customer\_id: integer ... 3 more fields]

Table ▼

+

🔍 🔍 📄 🗑️

|    | t_id | customer_id | account_type | balance | hashkey    |
|----|------|-------------|--------------|---------|------------|
| 3  | 82   | 2           | Checking     | 8300.5  | 3247491094 |
| 4  | 52   | 10          | Checking     | 5300    | 2280475706 |
| 5  | 65   | 69          | Savings      | 550.25  | 3693472785 |
| 6  | 76   | 22          | Checking     | 7700    | 3359048414 |
| 7  | 62   | 35          | Checking     | 6300.5  | 800230545  |
| 8  | 98   | 49          | Checking     | 9900.5  | 3893038149 |
| 9  | 53   | 86          | Savings      | 400.25  | 1685401127 |
| 10 | 66   | 26          | Checking     | 6700.5  | 372242038  |
| 11 | 72   | 17          | Checking     | 7300    | 1425050024 |
| 12 | 86   | 21          | Checking     | 8700.5  | 2919860885 |

**4. Defining source and the target path and reading the parquet file from the silver layer:**

▶

✓ 12:46 AM (1s)

14

```
src_path = "/mnt/container1/silver_databricks/accounts"
tgt_path = "/mnt/container1/gold_databricks/accounts_scdtype1"
src_accounts = spark.read.format("parquet").load(src_path)
```

▶ (1) Spark Jobs

▶ 📄 src\_accounts: pyspark.sql.dataframe.DataFrame = [account\_id: integer, customer\_id: integer ... 2 more fields]

**5. After importing pyspark.sql functions, I added another column, hashkey:**


▶ ✓ 01:02 AM (<1s) 15

```
from pyspark.sql.functions import *
```

▶ ✓ 01:02 AM (3s) 16

```
# adding hash key
src_accounts = src_accounts.withColumn("hashkey", crc32(concat(*src_accounts.
columns)))
display(src_accounts)
```

▶ (1) Spark Jobs

▶  src\_accounts: pyspark.sql.dataframe.DataFrame = [account\_id: integer, customer\_id: integer ... 3 more fields]

| Table ▾ |      | +           |              |         |            | 🔍 🔍 📄 📄 |  |
|---------|------|-------------|--------------|---------|------------|---------|--|
|         | t_id | customer_id | account_type | balance | hashkey    |         |  |
| 52      | 20   | 21          | Checking     | 2000    | 2020209643 |         |  |
| 53      | 71   | 73          | Savings      | 625.75  | 2590228591 |         |  |
| 54      | 13   | 29          | Savings      | 1300.25 | 3025925162 |         |  |
| 55      | 64   | 12          | Checking     | 6500    | 1796018383 |         |  |
| 56      | 61   | 52          | Savings      | 500.25  | 2662432370 |         |  |
| 57      | 32   | 9           | Checking     | 3300    | 2957212870 |         |  |
| 58      | 89   | 54          | Savings      | 850.25  | 540724915  |         |  |
| 59      | 31   | 71          | Savings      | 125.75  | 352326581  |         |  |
| 60      | 90   | 38          | Checking     | 9100.5  | 2053712775 |         |  |
| 61      | 10   | 92          | Checking     | 1800.5  | 533177005  |         |  |

## 6. Import the Delta Table class:

```
▶ ✓ 12:46 AM (2s) 17  
  
from delta.tables import DeltaTable  
deltatable = DeltaTable.forPath(spark, tgt_path)  
deltatable.toDF().show
```

```
<bound method DataFrame.show of DataFrame[customer_id: int, account_id: int, account_type: string, balance: float, hashkey: bigint, createdby: string, createddate: timestamp, updatedby: string, updateddate: timestamp]>
```

## 7. Merging the delta table with dataframe and map the columns

01:06 AM (<1s)

18

```
from pyspark.sql.functions import col
src_accounts = src_accounts.alias("account").join(deltatable.toDF().alias("tgt"),
((col("account.account_id")==col("tgt.account_id")) & (col("account.hashkey")==col
("tgt.hashkey"))),"anti").select("account.*")
```

src\_accounts: pyspark.sql.dataframe.DataFrame = [account\_id: integer, customer\_id: integer ... 3 mo  
fields]

01:14 AM (13s)

19

Python



```
deltatable.alias("tgt").merge(
    src_accounts.alias("src"),
    "tgt.account_id = src.account_id")\
    .whenMatchedUpdate(set={"tgt.account_id": "src.account_id ", "tgt.customer_id":
"src.customer_id", "tgt.account_type": "src.account_type", "tgt.balance": "src.
balance", "tgt.hashkey": "src.hashkey", "tgt.updatedby": lit("databricks"), "tgt.
updateddate": current_timestamp()})\
    .whenNotMatchedInsert(values={"tgt.account_id": "src.account_id ", "tgt.
customer_id": "src.customer_id", "tgt.account_type": "src.account_type", "tgt.
balance": "src.balance", "tgt.hashkey": "src.hashkey", "tgt.createdby": lit
("databricks"), "tgt.createddate": current_timestamp(), "tgt.updatedby": lit
("databricks"), "tgt.updateddate": current_timestamp()})\
    .execute()
display(spark.read.format("delta").load(tgt_path))
```

(8) Spark Jobs

| Table ▾ |             | +        |  |              |         |      |
|---------|-------------|----------|--|--------------|---------|------|
|         | customer_id | accou... |  | account_type | balance | hasl |
| 1       | 27          | 36       |  | Checking     | 3700    | 1    |
| 2       | 65          | 85       |  | Savings      | 800.25  | 3    |
| 3       | 2           | 82       |  | Checking     | 8300.5  | 3    |

8. Here's the file has created in the gold\_databricks folder:

<<

↑

Upload

+

Add Directory

↺

Refresh

|

↶

Rename

🗑

Delete

↔

Change tier

...

**Authentication method:** Access key [\(Switch to Microsoft Entra user account\)](#)

**Location:** [container1](#) / gold\_databricks

Search blobs by prefix (case-sensitive)

Show deleted objects

↑ Upload

+ Add Directory

↻ Refresh

|

↶ Rename

🗑 Delete

↔ Change tier

⋮

**Authentication method:** Access key [\(Switch to Microsoft Entra user account\)](#)

**Location:** [container1](#) / [gold\\_databricks](#) / [accounts\\_scdtype1](#)

Search blobs by prefix (case-sensitive)

☐ Show deleted objects

|                          | Name                              | Modified                | Access tier    | Archive statu |
|--------------------------|-----------------------------------|-------------------------|----------------|---------------|
| <input type="checkbox"/> | 📁 [..]                            |                         |                |               |
| <input type="checkbox"/> | 📁 _delta_log                      | 4/29/2025, 12:18:50 ... |                |               |
| <input type="checkbox"/> | 📄 part-00000-ac0474da-c187-40d... | 4/29/2025, 1:14:44 AM   | Hot (Inferred) |               |

9. The same steps have to repeat for all other 4 files.

**Next step is to Schedule the pipeline:**

1. There is an option “Schedule at the top right corner.
2. Give the Job Name.

3. Select the appropriate time. Even can select how often the pipeline should be executed:

Run all

terminated

Schedule

### New schedule

Job name\*

Project\_2

SimpleAdvanced

Schedule

Every

Day

at

10

:

30

☐ Show details

Minute

Hour

✓ Day

Week

Month

Timezone

(UTC-07:00) Mountain Standard Time (North America)DST

Compute type

Serverless

More options

CancelCreate

#### 4. Select Timezone -> Create

Schedules (1)

Add schedule

● At 01:34 (UTC)

Silver\_layer

Time=174555512000),  
.3, modificationTime=174555512000)]