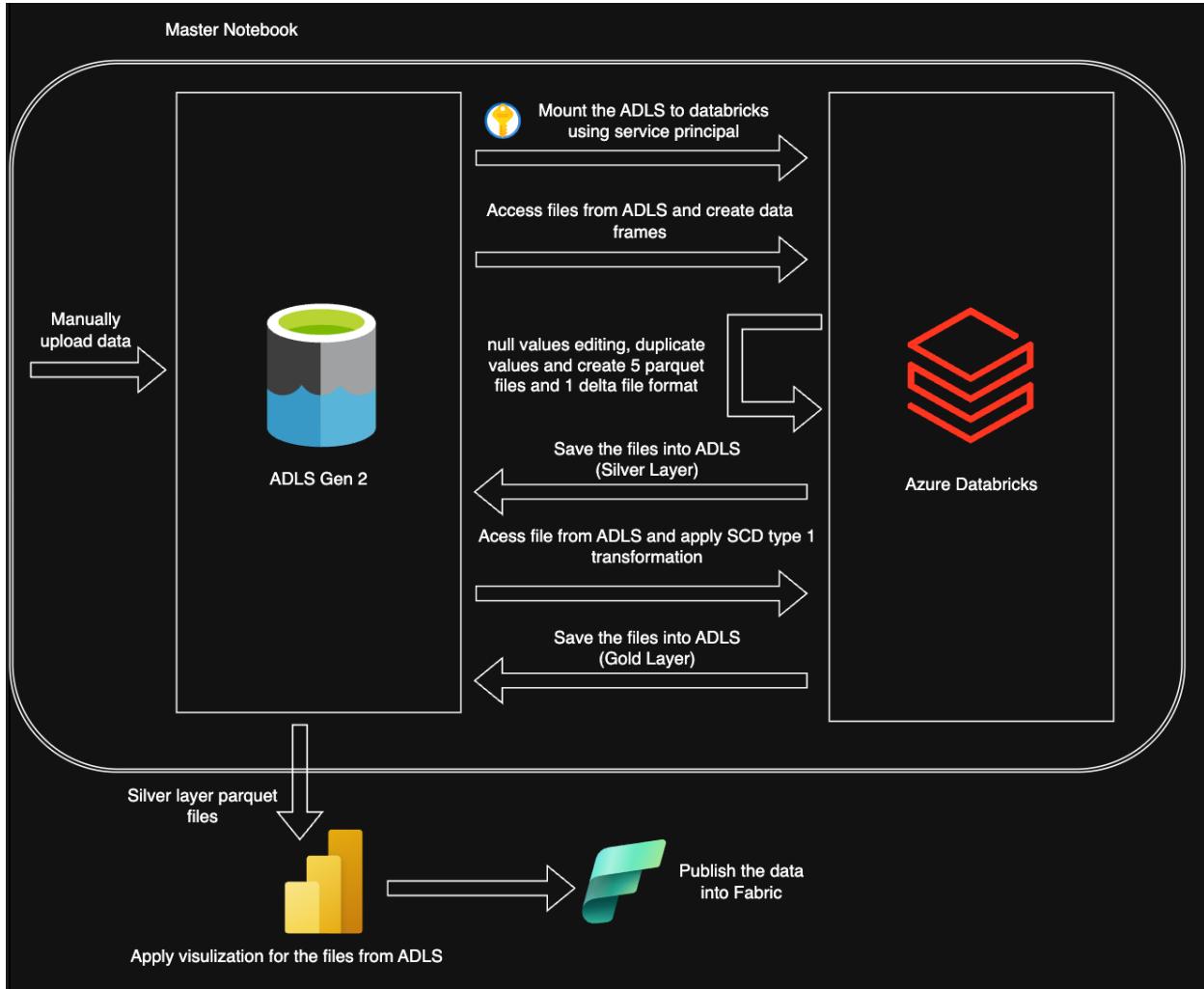


PROJECT 2

The project focuses on designing and implementing a comprehensive data pipeline to process customer account data. It involves extracting data from ADLS GEN2 (Bronze layer), transforming it in the silver layer using Databricks Notebooks, and then storing the results in a Delta Table formatted as SCD Type 1 in ADLS GEN2 (Gold layer). The goal is to provide efficient, accurate, and scalable data processing to support subsequent analytics and reporting requirements.

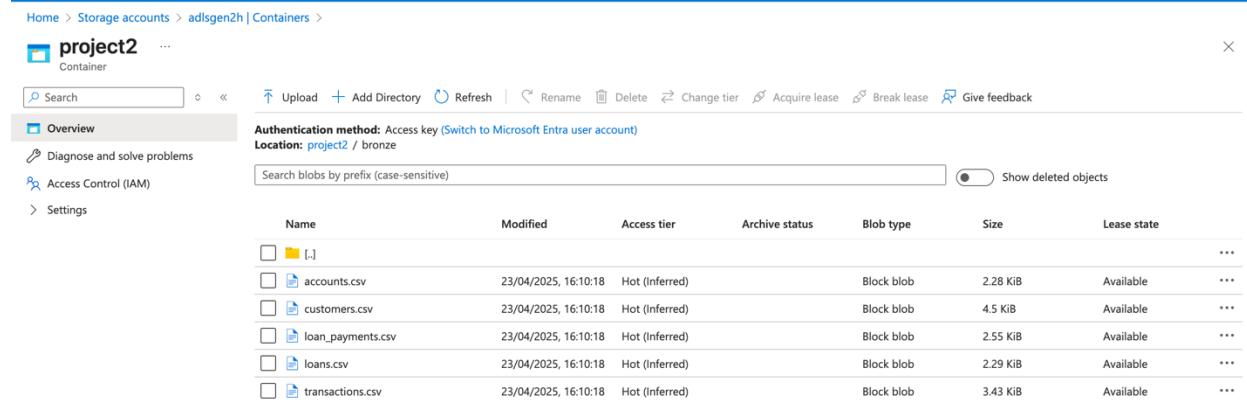


Architecture Diagram

Step 1:

First we need to manually upload data into container in ADLS gen2 and then create secrets in keyvault and also create a service principal for databricks to allow reading and writing

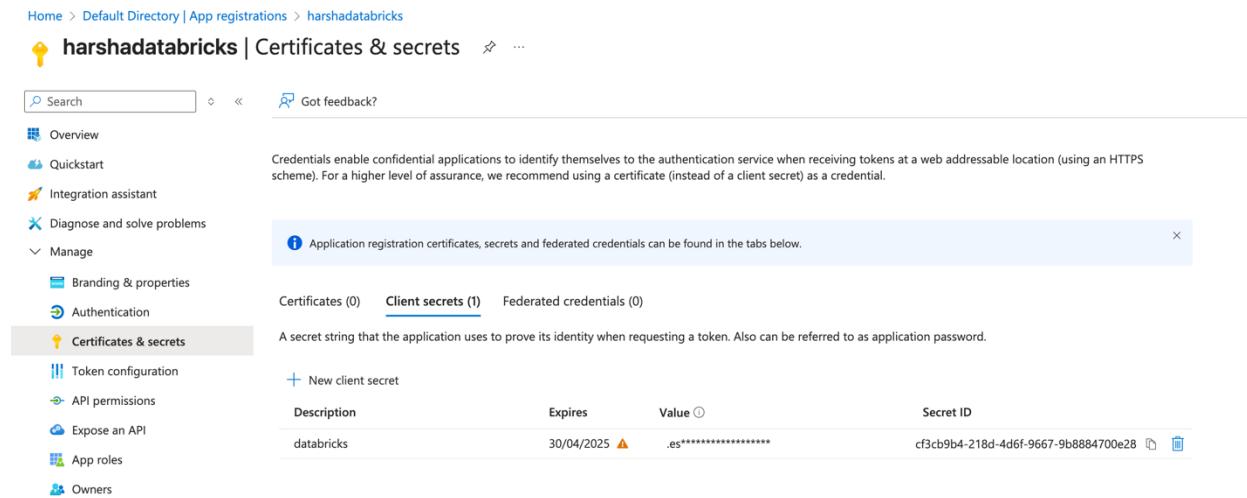
into ADLS finally we create Scope in databricks to create mount and link the databricks and ADLS



The screenshot shows the Azure Storage Explorer interface. On the left, there's a sidebar with options like Overview, Diagnose and solve problems, Access Control (IAM), and Settings. The main area displays a list of files in the 'project2' container. The files are:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]	23/04/2025, 16:10:18	Hot (Inferred)		Block blob	2.28 KiB	Available
accounts.csv	23/04/2025, 16:10:18	Hot (Inferred)		Block blob	4.5 KiB	Available
customers.csv	23/04/2025, 16:10:18	Hot (Inferred)		Block blob	2.55 KiB	Available
loan_payments.csv	23/04/2025, 16:10:18	Hot (Inferred)		Block blob	2.29 KiB	Available
loans.csv	23/04/2025, 16:10:18	Hot (Inferred)		Block blob	3.43 KiB	Available
transactions.csv	23/04/2025, 16:10:18	Hot (Inferred)		Block blob	2.28 KiB	Available

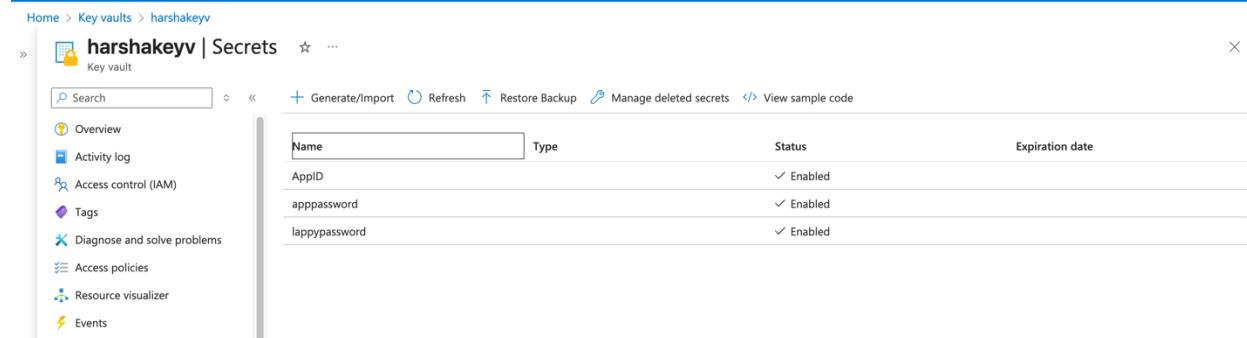
Uploaded the files



The screenshot shows the Azure App registrations page. The sidebar includes options like Overview, Quickstart, Integration assistant, Diagnose and solve problems, Manage, Branding & properties, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners, and App registrations. The 'Certificates & secrets' section is selected. It shows a table for 'Client secrets' with one entry:

Description	Expires	Value	Secret ID
databricks	30/04/2025	.es*****	cf3cb9b4-218d-4d6f-9667-9b8884700e28

Created service principal for databricks



The screenshot shows the Azure Key vaults page. The sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Access policies, Resource visualizer, and Events. The 'Secrets' section is selected. It shows a table with three secrets:

Name	Type	Status	Expiration date
AppID		✓ Enabled	
apppassword		✓ Enabled	
lappypassword		✓ Enabled	

Created secrets using app id and password in keyvault to access adls

HomePage / Create Secret Scope

Create Secret Scope

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

Scope Name [?](#)
askvconnections

Manage Principal [?](#)
Creator

Azure Key Vault [?](#)

DNS Name
https://harshakeyv.vault.azure.net/

Resource ID
/subscriptions/.../resourceGroups/.../providers/Microsoft.KeyVault/vaults/...

Cancel Create

Created scope in databricks

After creating scope the step 1 is done we now proceed to step 2.

```
1
dbutils.secrets.listScopes()
[SecretScope(name='askvconnection')]

2
dbutils.secrets.list("askvconnection")
[SecretMetadata(key='AppID'),
 SecretMetadata(key='apppassword'),
 SecretMetadata(key='lappypassword')]
```

We are checking the secrets are accessible or not

```
 2 days ago (1s) 3
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="askvconnection", key="AppID"),
    "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="askvconnection", key="apppassword"),
    "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/2dfe05fa-e532-47e7-95c8-56f0f4f444c8/oauth2/token"
}
mount_point = "/mnt/project2"

if not any(m.mountPoint == mount_point for m in dbutils.fs.mounts()):
    dbutils.fs.mount(
        source="abfss://project2@adlsgen2h.dfs.core.windows.net/",
        mount_point=mount_point,
        extra_configs=configs
)
else:
    print(f"{mount_point} is already mounted.")

/mnt/project2 is already mounted.
```

We are configuring our adls to databricks using secrets

```
 2 days ago (20s) 5
df_account = spark.read.format("csv") \
    .option("header", "true") \
    .load("/mnt/project2/bronze/accounts.csv")
df_customer = spark.read.format("csv") \
    .option("header", "true") \
    .load("/mnt/project2/bronze/customers.csv")
df_loanpayments = spark.read.format("csv") \
    .option("header", "true") \
    .load("/mnt/project2/bronze/loan_payments.csv")
df_loan = spark.read.format("csv") \
    .option("header", "true") \
    .load("/mnt/project2/bronze/loans.csv")
df_transaction = spark.read.format("csv") \
    .option("header", "true") \
    .load("/mnt/project2/bronze/transactions.csv")
display(df_account)
display(df_customer)
display(df_loanpayments)
display(df_loan)
display(df_transaction)
▶ (10) Spark Jobs
```

We are reading the data in our container

```
 2 days ago (3s) 6
  df_account_clean = df_account.dropDuplicates()
  df_customer_clean = df_customer.dropDuplicates()
  df_loanpayments_clean = df_loanpayments.dropDuplicates()
  df_loan_clean = df_loan.dropDuplicates()
  df_transaction_clean = df_transaction.dropDuplicates()
  display(df_account_clean)
  display(df_customer_clean)
  display(df_loanpayments_clean)
  display(df_loan_clean)
  display(df_transaction_clean)

  ▶ (10) Spark Jobs
  ▶ df_account_clean: pyspark.sql.dataframe.DataFrame = [account_id: string, customer_id: string ... 2 more fields]
  ▶ df_customer_clean: pyspark.sql.dataframe.DataFrame = [customer_id: string, first_name: string ... 5 more fields]
  ▶ df_loan_clean: pyspark.sql.dataframe.DataFrame = [loan_id: string, customer_id: string ... 3 more fields]
  ▶ df_loanpayments_clean: pyspark.sql.dataframe.DataFrame = [payment_id: string, loan_id: string ... 2 more fields]
  ▶ df_transaction_clean: pyspark.sql.dataframe.DataFrame = [transaction_id: string, account_id: string ... 3 more fields]
```

Removing the duplicates

```
 2 days ago (2s) 7
  df_account_filled = df_account_clean.fillna("Unknown")
  display(df_account_filled)
  df_customer_filled = df_customer_clean.fillna("Unknown")
  display(df_customer_filled)
  df_loanpayments_filled = df_loanpayments_clean.fillna("Unknown")
  display(df_loanpayments_filled)
  df_loan_filled = df_loan_clean.fillna("Unknown")
  display(df_loan_filled)
  df_transaction_filled = df_transaction_clean.fillna("Unknown")
  display(df_transaction_filled)

  ▶ (10) Spark Jobs
  ▶ df_account_filled: pyspark.sql.dataframe.DataFrame = [account_id: string, customer_id: string ... 2 more fields]
  ▶ df_customer_filled: pyspark.sql.dataframe.DataFrame = [customer_id: string, first_name: string ... 5 more fields]
  ▶ df_loan_filled: pyspark.sql.dataframe.DataFrame = [loan_id: string, customer_id: string ... 3 more fields]
  ▶ df_loanpayments_filled: pyspark.sql.dataframe.DataFrame = [payment_id: string, loan_id: string ... 2 more fields]
  ▶ df_transaction_filled: pyspark.sql.dataframe.DataFrame = [transaction_id: string, account_id: string ... 3 more fields]
```

Replacing the null values

```

▶ ✓ 2 days ago (13s) 8
df_account_filled.write.format("parquet") \
    .mode("overwrite") \
    .save("/mnt/project2/silver/accounts")
df_customer_filled.write.format("parquet") \
    .mode("overwrite") \
    .save("/mnt/project2/silver/customers")
df_loanpayments_filled.write.format("parquet") \
    .mode("overwrite") \
    .save("/mnt/project2/silver/loan_payments")
df_loan_filled.write.format("parquet") \
    .mode("overwrite") \
    .save("/mnt/project2/silver/loans")
df_transaction_filled.write.format("parquet") \
    .mode("overwrite") \
    .save("/mnt/project2/silver/transactions")

▶ (10) Spark Jobs

```

Name	Type	Owner	Created at
Notebook_1	Notebook	Harsha Vardhan	Apr 23, 2025, 04:01 PM
Notebook_2	Notebook	Harsha Vardhan	Apr 23, 2025, 07:40 PM
Notebook_3_goldlayer	Notebook	Harsha Vardhan	Apr 23, 2025, 08:23 PM

Converting the file into parquet format

```

▶ ✓ Yesterday (5s) 2
from pyspark.sql.functions import col

# Step 1: Join all DataFrames
df_joined = df_accounts_parquet.join(df_transactions_parquet, on="account_id", how="left") \
    .join(df_customers_parquet, on="customer_id", how="left") \
    .join(df_loans_parquet, on="customer_id", how="left") \
    .join(df_loanpayments_parquet, on="loan_id", how="left")

# Step 2: Define grouped order of columns
ordered_columns = [
    # IDs
    "account_id", "transaction_id", "customer_id", "loan_id", "payment_id",

    # Dates grouped
    "transaction_date", "payment_date", "loan_term",

    # Amounts grouped
    "transaction_amount", "payment_amount", "loan_amount", "balance",

    # Customer info grouped
    "first_name", "last_name", "address", "city", "state", "zip"
]

# Step 3: Get any remaining columns
all_columns = df_joined.columns
remaining_columns = [c for c in all_columns if c not in ordered_columns]

```

Now joining all the columns and convert the file into parquet for further visualization and reporting it in fabric

```

▶ ✓ 10:32 PM (4s) 5
df_clean.write.format("parquet").save("/mnt/project2/silver/customer_loan_paymentsjoined")

```

► (7) Spark Jobs

Converted the joined code into parquet format

Home > adlsgen2h | Containers >

project2 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: project2 / silver

Search blobs by prefix (case-sensitive)

Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[...]						...
accounts	23/04/2025, 18:16:00				-	...
customer_loan_paymentsjoined	25/04/2025, 22:35:31				-	...
customers	23/04/2025, 18:17:24				-	...
loan_payments	23/04/2025, 18:17:26				-	...
loans	23/04/2025, 18:17:28				-	...
transactions	23/04/2025, 18:17:31				-	...

Files created after running the notebook

Step 3:

```

2 days ago (9s) 1 SQL
```
CREATE TABLE if not exists delta.`/mnt/project2/gold/accountsscdtype1` (
 account_id STRING,
 customer_id STRING,
 account_type STRING,
 balance STRING,
 hashkey BIGINT,
 created_by STRING,
 created_date TIMESTAMP,
 updated_by STRING,
 updated_date TIMESTAMP
);

CREATE TABLE if not exists delta.`/mnt/project2/gold/customersscdtype1` (
 customer_id STRING,
 first_name STRING,
 last_name STRING,
 address STRING,
 city STRING,
 state STRING,
 zip STRING,
 hashkey BIGINT,
 created_by STRING,
 created_date TIMESTAMP,
 updated_by STRING,
 updated_date TIMESTAMP
);
```

```

Applying SCD type 1 transformation to parquet file in silver layer

```

  ▶ ✓ 2 days ago (18s) 6
  from pyspark.sql.functions import current_timestamp, lit

  dbtable_account.alias("tgt").merge(
    df_src1_account.alias("src"),
    "tgt.account_id = src.account_id"
  ).whenMatchedUpdate(set={
    "account_id": "src.account_id",
    "customer_id": "src.customer_id",
    "account_type": "src.account_type",
    "balance": "src.balance",
    "hashkey": "src.hash_key",
    "updated_date": current_timestamp(),
    "updated_by": lit("databricks_Updated")
  }).whenNotMatchedInsert(values={
    "account_id": "src.account_id",
    "customer_id": "src.customer_id",
    "account_type": "src.account_type",
    "balance": "src.balance",
    "hashkey": "src.hash_key",
    "created_date": current_timestamp(),
    "created_by": lit("databricks"),
    "updated_date": current_timestamp(),
    "updated_by": lit("databricks")
  }).execute()

```

```

  ▶ ✓ 2 days ago (3s) 7
  display(spark.read.format("delta").option("header","true").load(tgt_path_account))
  display(spark.read.format("delta").option("header","true").load(tgt_path_customer))
  display(spark.read.format("delta").option("header","true").load(tgt_path_loanpayments))
  display(spark.read.format("delta").option("header","true").load(tgt_path_loan))
  display(spark.read.format("delta").option("header","true").load(tgt_path_transactions))

```

▶ (5) Spark Jobs

	Δ account_id	Δ customer_id	Δ account_type	Δ balance	Δ hashkey	Δ created_by	Δ created_date
1	48	6	Checking	4900.00	1456590466	databricks	2025-04-24T04:12:06
2	21	53	Savings	300.25	2459936851	databricks	2025-04-24T04:12:06

Successfully converted all the files into scd type 1 and stored In gold layer.

Workflows

Jobs Job runs Pipelines

Send feedback

Filter jobs Owned by me Accessible by me Favorites Tags

Load tutorial Create job

Name Tags Created by Trigger Recent runs

Project2 master notebook Harsha Vardhan

Workflows > Jobs > Project2 master notebook > Project2 master notebook run

Send feedback Delete job run Repair run

Graph Timeline List

Job run details

- Job ID: 154773372709884
- Job run ID: 813843330459608
- Launched: Manually
- Started: Apr 23, 2025, 09:27 PM
- Ended: Apr 23, 2025, 09:29 PM
- Duration: 1m 47s
- Queue duration: -
- Status: Succeeded
- Lineage: No lineage information for this job. Learn more

Compute

- Harsha Vardhan's Cluster
- Single node: Standard_D4ds_v5 - Release: 15.4.13

View details Spark UI Logs Metrics

I have created 3 notebook so I created a job to run them all one after the other and it successfully executed.

Home > adlsgen2h | Containers >

project2 Container

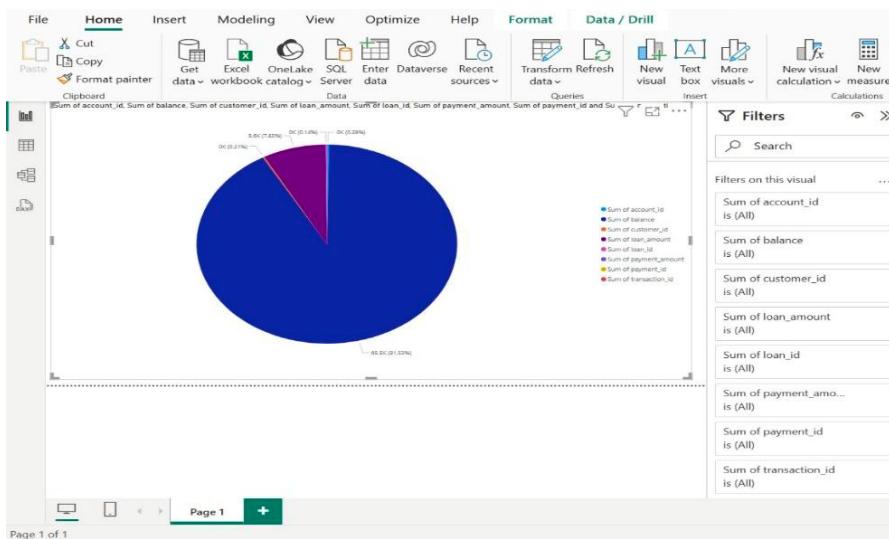
Search Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: project2 / gold

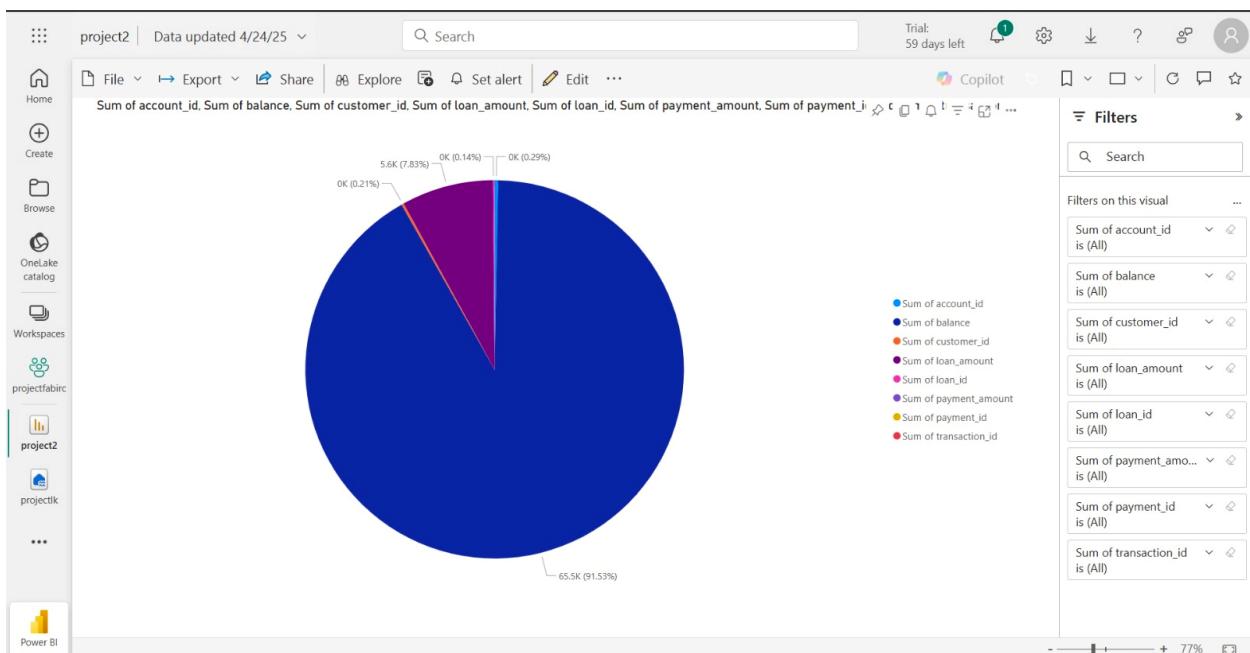
Search blobs by prefix (case-sensitive)

Show deleted objects

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[...]	23/04/2025, 20:55:41				-	***
accountsscdtype1	23/04/2025, 20:55:42				-	***
customersscdtype1	23/04/2025, 20:55:44				-	***
loan_paymentsscdtype1	23/04/2025, 20:55:46				-	***
loansscdtype1	23/04/2025, 20:55:46				-	***
transactionsscdtype1	23/04/2025, 20:55:48				-	***



Visualization created in power bi.



Published in fabric