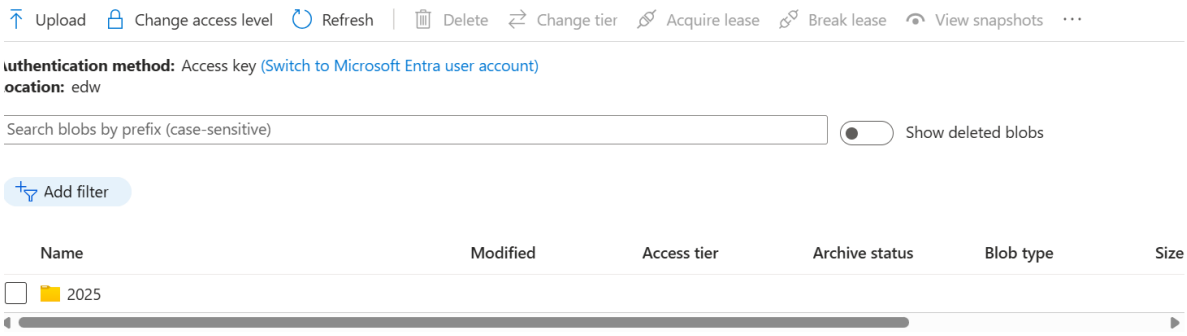
SCD Type 1 in Data Bricks

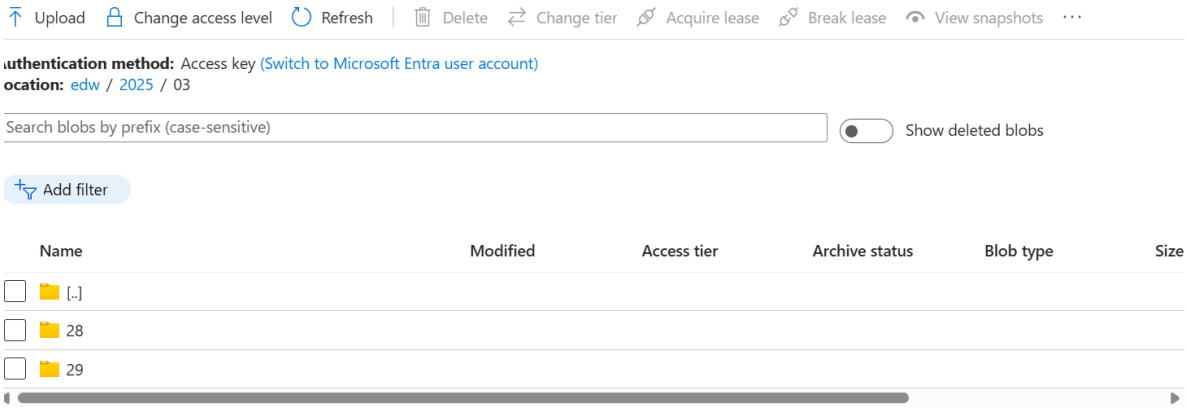
Upload files to storage account

Uploaded Day 1 and Day 2 files to ADLS storage into SCD TYPE folder with day 1 and Day 2 as subfolders

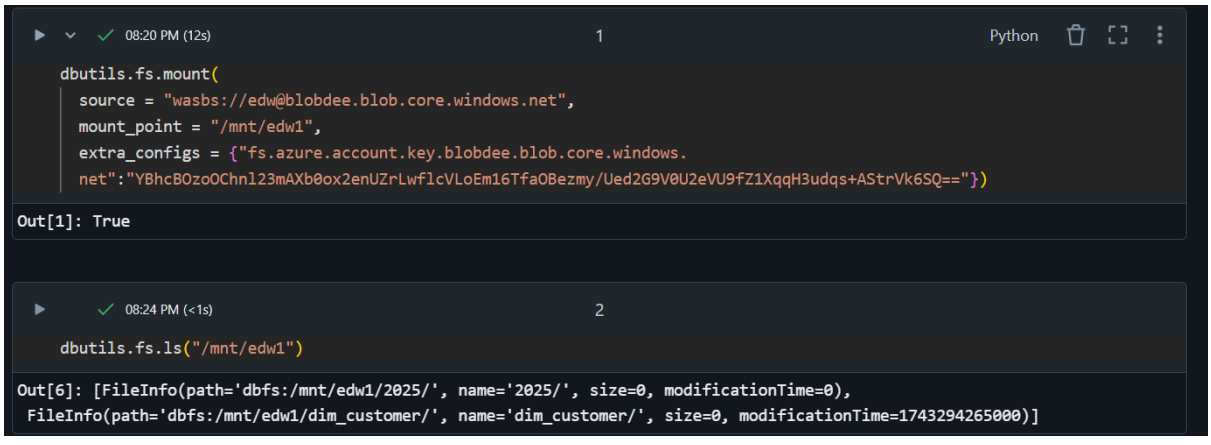
2025 folder



28 and 29 sub folders

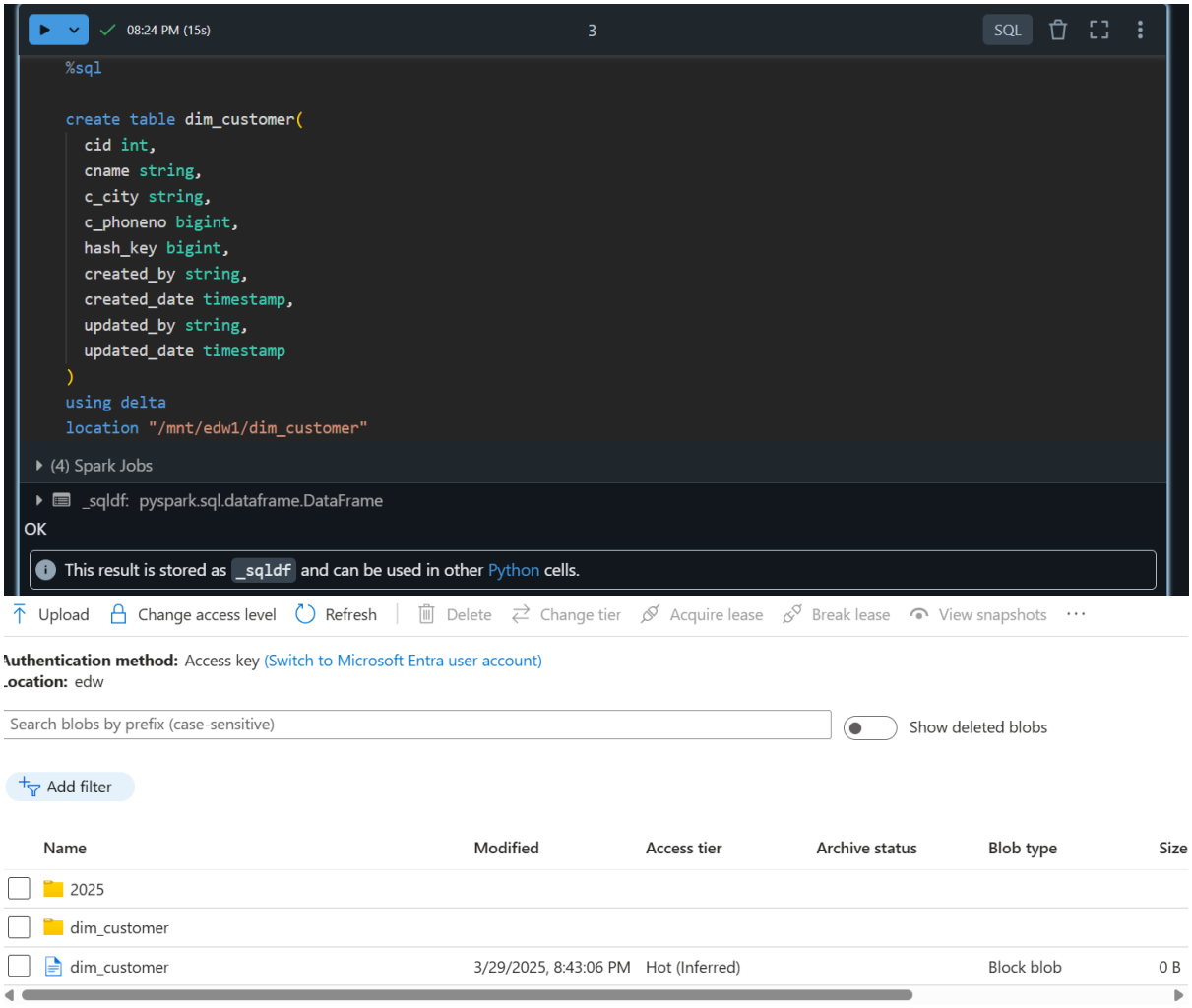


Check if the mount point is existed if not create one, used data bricks community edition



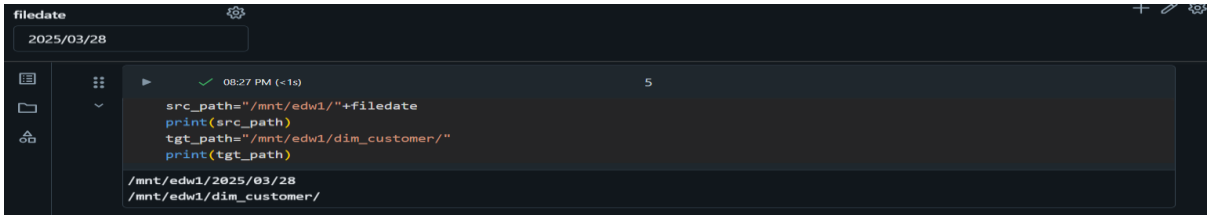
Write a sql query to create a delta table

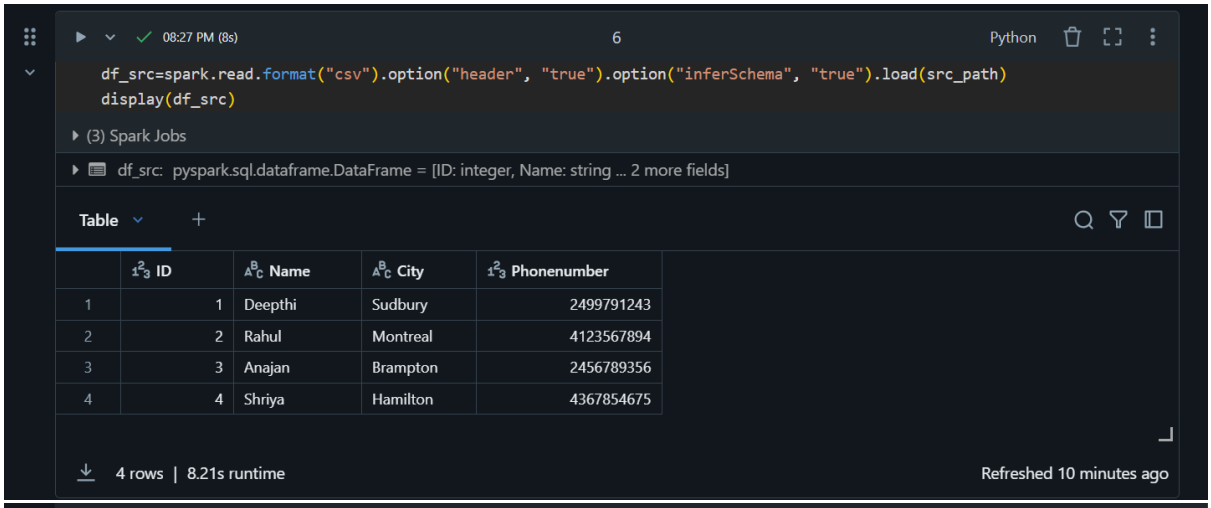
```
%sql
create table dim_customer(
  cid int,
  cname string,
  c_city string,
  c_phoneno bigint,
  hash_key bigint,
  created_by string,
  created_date timestamp,
  updated_by string,
  updated_date timestamp
)
using delta
location "/mnt/edw1/dim_customer"
```

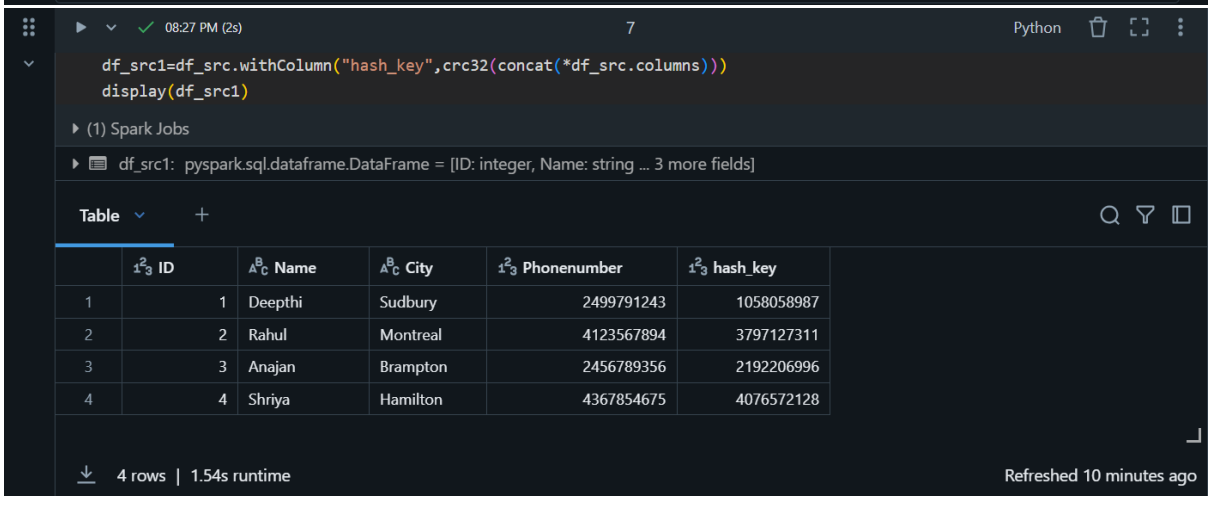


New folder dim_customer is created on blob storage


filedate
08:27 PM (<1s)
4
from pyspark.sql.functions import *
dbutils.widgets.text('filedate','')
filedate=dbutils.widgets.get('filedate')


filedate
2025/03/28
08:27 PM (<1s)
5
src_path="/mnt/edw1/"+filedate
print(src_path)
tgt_path="/mnt/edw1/dim_customer/"
print(tgt_path)
/mnt/edw1/2025/03/28
/mnt/edw1/dim_customer/


08:27 PM (8s)
6
Python
df_src=spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(src_path)
display(df_src)
(3) Spark Jobs
df_src: pyspark.sql.dataframe.DataFrame = [ID: integer, Name: string ... 2 more fields]
Table
ID
Name
City
Phonenumber
1 1 Deepthi Sudbury 2499791243
2 2 Rahul Montreal 4123567894
3 3 Anajan Brampton 2456789356
4 4 Shriya Hamilton 4367854675
4 rows | 8.21s runtime
Refreshed 10 minutes ago


08:27 PM (2s)
7
Python
df_src1=df_src.withColumn("hash_key",crc32(concat(*df_src.columns)))
display(df_src1)
(1) Spark Jobs
df_src1: pyspark.sql.dataframe.DataFrame = [ID: integer, Name: string ... 3 more fields]
Table
ID
Name
City
Phonenumber
hash_key
1 1 Deepthi Sudbury 2499791243 1058058987
2 2 Rahul Montreal 4123567894 3797127311
3 3 Anajan Brampton 2456789356 2192206996
4 4 Shriya Hamilton 4367854675 4076572128
4 rows | 1.54s runtime
Refreshed 10 minutes ago

08:28 PM (5s) 8

```
from delta.tables import DeltaTable
dbtable = DeltaTable.forPath(spark, tgt_path)
dbtable.toDF().show()
```

(1) Spark Jobs

```
+---+-----+------+---------+--------+----------+------------+----------+------------+
|cid|cname|c_city|c_phoneno|hash_key|created_by|created_date|updated_by|updated_date|
+---+-----+------+---------+--------+----------+------------+----------+------------+
+---+-----+------+---------+--------+----------+------------+----------+------------+
```

08:28 PM (2s) 9 Python

```
df_src1=df_src1.alias("src").join(dbtable.toDF().alias("tgt"), ((col("src.ID") == col("tgt.cid")) & (col("src.
hash_key") == col("tgt.hash_key"))), "anti").select(col("src.*"))
df_src1.show()
```

(1) Spark Jobs

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

```
+---+-------+--------+-----------+----------+
| ID|   Name|    City|Phonenumber|  hash_key|
+---+-------+--------+-----------+----------+
|  1|Deepthi| Sudbury| 2499791243|1058058987|
|  2|  Rahul|Montreal| 4123567894|3797127311|
|  3| Anajan|Brampton| 2456789356|2192206996|
|  4| Shriya|Hamilton| 4367854675|4076572128|
+---+-------+--------+-----------+----------+
```

08:32 PM (19s) 10

```
dbtable.alias("tgt").merge(df_src1.alias("src"),"tgt.cid = src.ID")\
    .whenMatchedUpdate(set={"tgt.cid":"src.ID","tgt.cname":"src.Name","tgt.c_city":"src.City","tgt.c_phoneno":"src.
    Phonenumber","tgt.hash_key":"src.hash_key","tgt.updated_date":current_timestamp(),"tgt.updated_by":lit
    ("databricks_Updated")})\
        .whenNotMatchedInsert(values={"tgt.cid":"src.ID","tgt.cname":"src.Name","tgt.c_city":"src.City","tgt.
        c_phoneno":"src.Phonenumber","tgt.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

Day 1 data

08:33 PM (9s) 11 Python

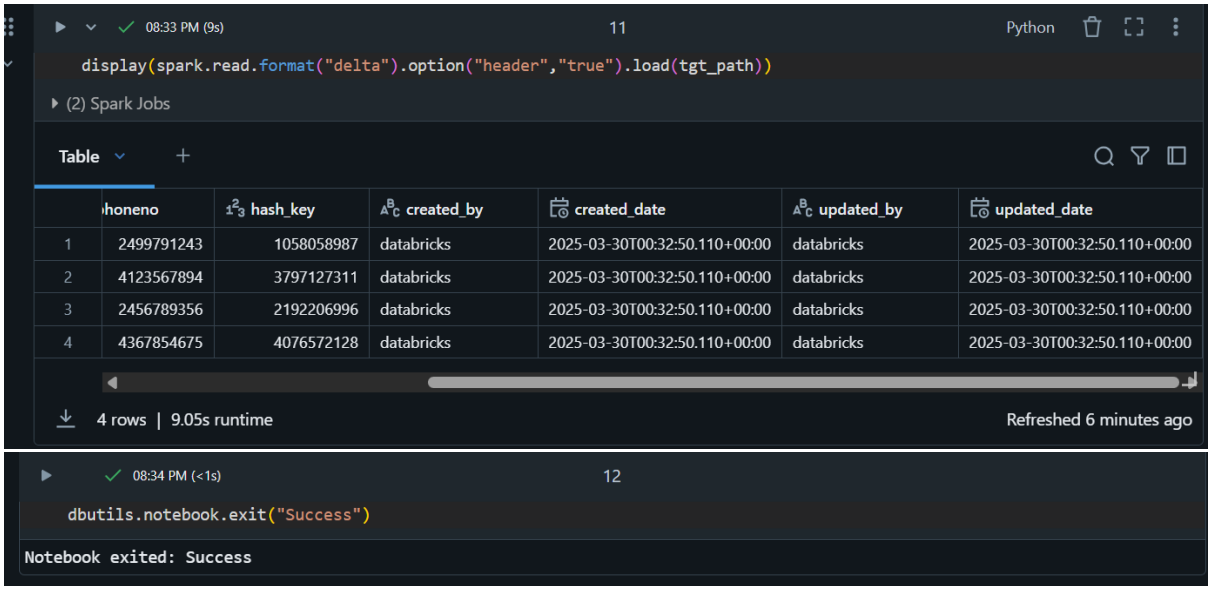
```
display(spark.read.format("delta").option("header","true").load(tgt_path))
```

(2) Spark Jobs

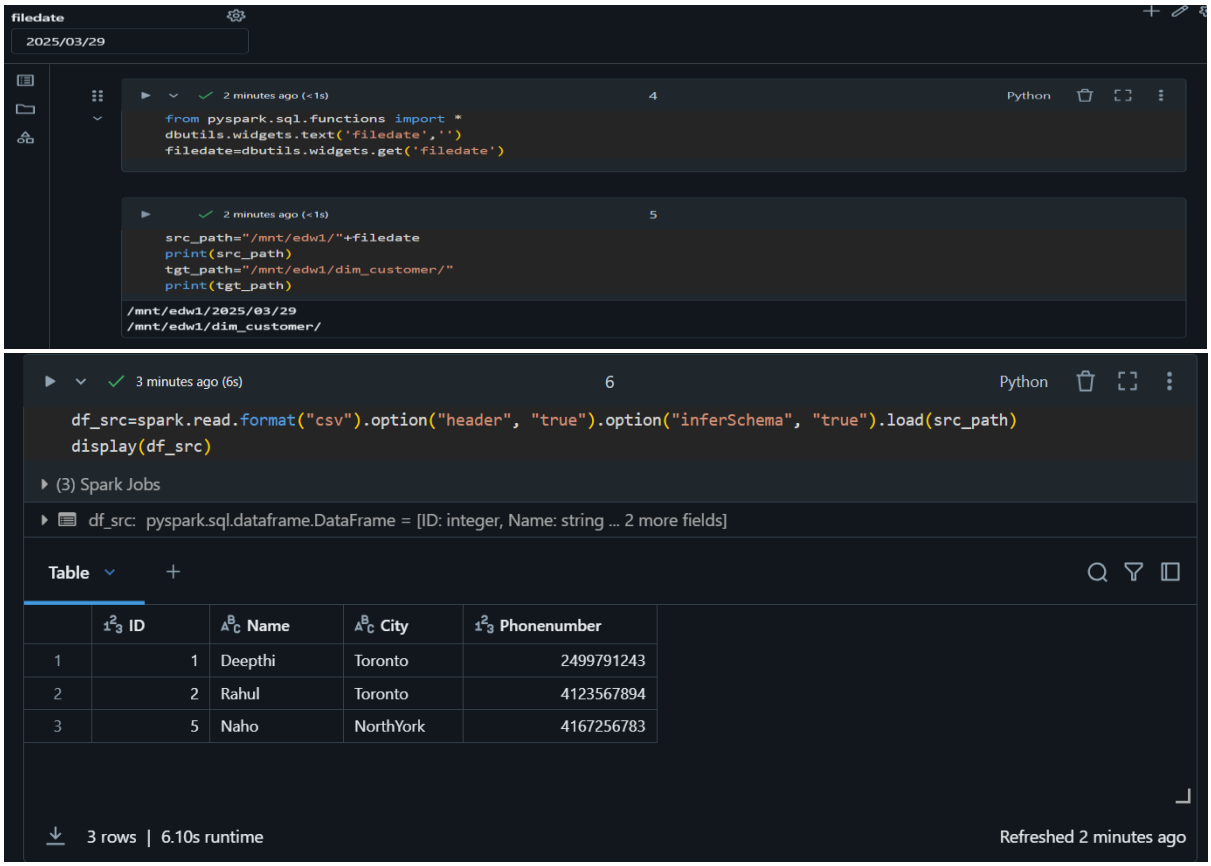
Table +

	c_city	c_phoneno	hash_key	created_by	created_date	updated_by	
1	Sudbury	2499791243	1058058987	databricks	2025-03-30T00:32:50.110+00:00	databricks	202
2	Montreal	4123567894	3797127311	databricks	2025-03-30T00:32:50.110+00:00	databricks	202
3	Brampton	2456789356	2192206996	databricks	2025-03-30T00:32:50.110+00:00	databricks	202
4	Hamilton	4367854675	4076572128	databricks	2025-03-30T00:32:50.110+00:00	databricks	202

4 rows | 9.05s runtime Refreshed 5 minutes ago


08:33 PM (9s)
11
Python
display(spark.read.format("delta").option("header","true").load(tgt_path))
(2) Spark Jobs
Table
honeno
hash_key
created_by
created_date
updated_by
updated_date
1 2499791243 1058058987 databricks 2025-03-30T00:32:50.110+00:00 databricks 2025-03-30T00:32:50.110+00:00
2 4123567894 3797127311 databricks 2025-03-30T00:32:50.110+00:00 databricks 2025-03-30T00:32:50.110+00:00
3 2456789356 2192206996 databricks 2025-03-30T00:32:50.110+00:00 databricks 2025-03-30T00:32:50.110+00:00
4 4367854675 4076572128 databricks 2025-03-30T00:32:50.110+00:00 databricks 2025-03-30T00:32:50.110+00:00
4 rows | 9.05s runtime
Refreshed 6 minutes ago
08:34 PM (<1s)
12
dbutils.notebook.exit("Success")
Notebook exited: Success

Day 2 data


filedate
2025/03/29
2 minutes ago (<1s)
4
Python
from pyspark.sql.functions import *
dbutils.widgets.text('filedate','')
filedate=dbutils.widgets.get('filedate')
2 minutes ago (<1s)
5
src_path="/mnt/edw1/"+filedate
print(src_path)
tgt_path="/mnt/edw1/dim_customer/"
print(tgt_path)
/mnt/edw1/2025/03/29
/mnt/edw1/dim_customer/
3 minutes ago (6s)
6
Python
df_src=spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(src_path)
display(df_src)
(3) Spark Jobs
df_src: pyspark.sql.dataframe.DataFrame = [ID: integer, Name: string ... 2 more fields]
Table
ID
Name
City
Phonenumber
1 1 Deepthi Toronto 2499791243
2 2 Rahul Toronto 4123567894
3 5 Naho NorthYork 4167256783
3 rows | 6.10s runtime
Refreshed 2 minutes ago

2 minutes ago (1s) 7

```
df_src1=df_src.withColumn("hash_key",crc32(concat(*df_src.columns)))
display(df_src1)
```

(1) Spark Jobs

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

Table

	ID	Name	City	Phonenumber	hash_key
1	1	Deepthi	Toronto	2499791243	650429413
2	2	Rahul	Toronto	4123567894	3692243240
3	5	Naho	NorthYork	4167256783	2378172573

2 minutes ago (5s) 8

```
from delta.tables import DeltaTable
dbtable = DeltaTable.forPath(spark, tgt_path)
dbtable.toDF().show()
```

(1) Spark Jobs

```
+---+-------+--------+----------+----------+----------+--------------------+----------+--------------------+
|cid|  cname|  c_city| c_phoneno|  hash_key|created_by|        created_date|updated_by|        updated_date|
+---+-------+--------+----------+----------+----------+--------------------+----------+--------------------+
|  1|Deepthi| Sudbury|2499791243|1058058987|databricks|2025-03-30 00:32:...|databricks|2025-03-30 00:32:...|
|  2|  Rahul|Montreal|4123567894|3797127311|databricks|2025-03-30 00:32:...|databricks|2025-03-30 00:32:...|
|  3| Anajan|Brampton|2456789356|2192206996|databricks|2025-03-30 00:32:...|databricks|2025-03-30 00:32:...|
|  4| Shriya|Hamilton|4367854675|4076572128|databricks|2025-03-30 00:32:...|databricks|2025-03-30 00:32:...|
+---+-------+--------+----------+----------+----------+--------------------+----------+--------------------+
```

2 minutes ago (4s) 9 Python

```
df_src1=df_src1.alias("src").join(dbtable.toDF().alias("tgt"), ((col("src.ID") == col("tgt.cid")) & (col("src.
hash_key") == col("tgt.hash_key"))), "anti").select(col("src.*"))
df_src1.show()
```

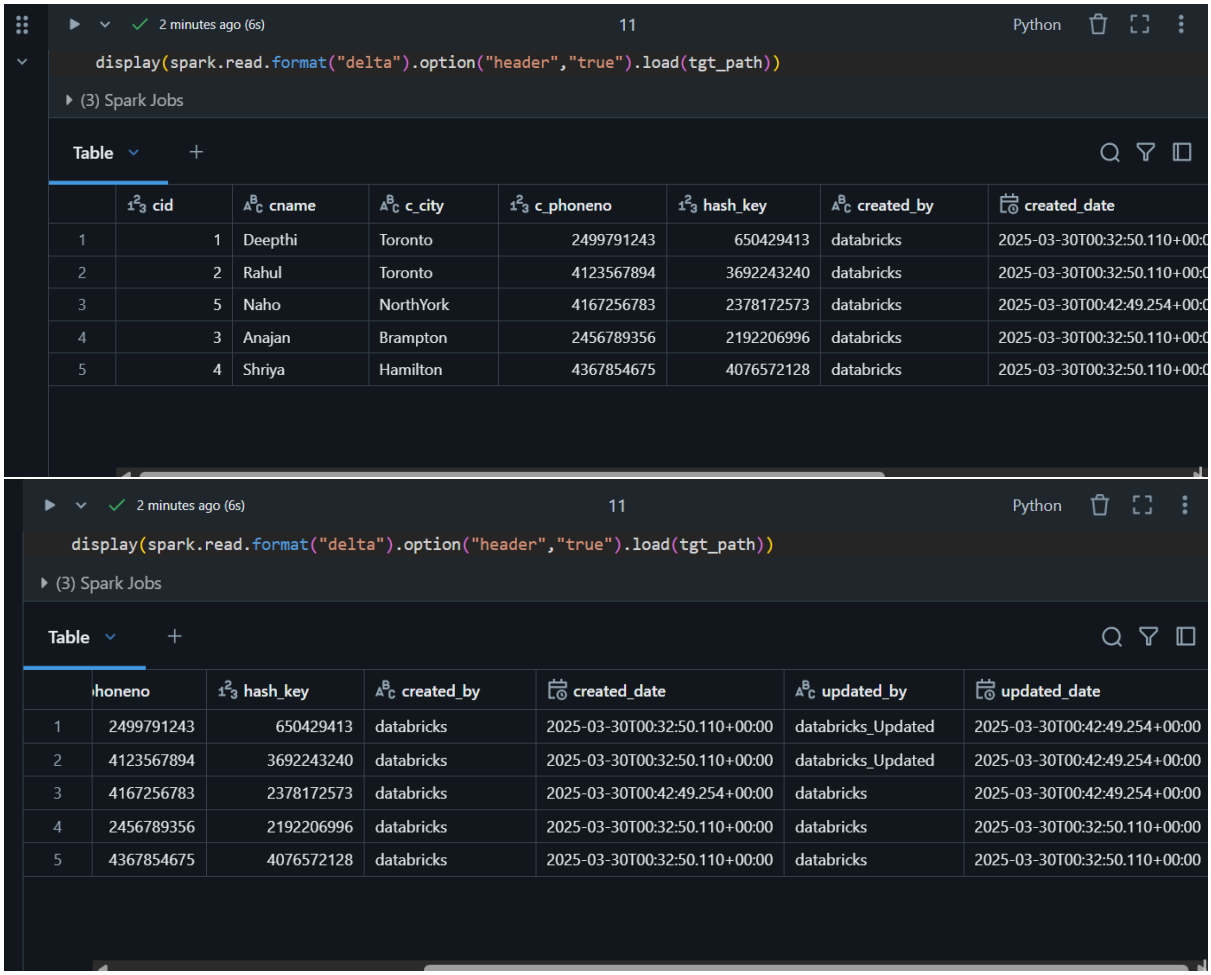
(2) Spark Jobs

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

```
+---+-------+---------+-----------+----------+
| ID|   Name|     City|Phonenumber|  hash_key|
+---+-------+---------+-----------+----------+
|  1|Deepthi|  Toronto| 2499791243| 650429413|
|  2|  Rahul|  Toronto| 4123567894|3692243240|
|  5|   Naho|NorthYork| 4167256783|2378172573|
+---+-------+---------+-----------+----------+
```

2 minutes ago (27s) 10 Python

```
dbtable.alias("tgt").merge(df_src1.alias("src"),"tgt.cid = src.ID")\
    .whenMatchedUpdate(set={"tgt.cid":"src.ID","tgt.cname":"src.Name","tgt.c_city":"src.City","tgt.c_phoneno":"src.
    Phonenumber","tgt.hash_key":"src.hash_key","tgt.updated_date":current_timestamp(),"tgt.updated_by":lit
    ("databricks_Updated")})\
        .whenNotMatchedInsert(values={"tgt.cid":"src.ID","tgt.cname":"src.Name","tgt.c_city":"src.City","tgt.
        c_phoneno":"src.Phonenumber","tgt.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()
```



```
2 minutes ago (6s)  11  Python
display(spark.read.format("delta").option("header","true").load(tgt_path))
(3) Spark Jobs
```

Table

	cid	cname	c_city	c_phoneno	hash_key	created_by	created_date
1	1	Deepthi	Toronto	2499791243	650429413	databricks	2025-03-30T00:32:50.110+00:0
2	2	Rahul	Toronto	4123567894	3692243240	databricks	2025-03-30T00:32:50.110+00:0
3	5	Naho	NorthYork	4167256783	2378172573	databricks	2025-03-30T00:42:49.254+00:0
4	3	Anajan	Brampton	2456789356	2192206996	databricks	2025-03-30T00:32:50.110+00:0
5	4	Shriya	Hamilton	4367854675	4076572128	databricks	2025-03-30T00:32:50.110+00:0


```
2 minutes ago (6s)  11  Python
display(spark.read.format("delta").option("header","true").load(tgt_path))
(3) Spark Jobs
```

Table

	honeno	hash_key	created_by	created_date	updated_by	updated_date
1	2499791243	650429413	databricks	2025-03-30T00:32:50.110+00:00	databricks_Updated	2025-03-30T00:42:49.254+00:00
2	4123567894	3692243240	databricks	2025-03-30T00:32:50.110+00:00	databricks_Updated	2025-03-30T00:42:49.254+00:00
3	4167256783	2378172573	databricks	2025-03-30T00:42:49.254+00:00	databricks	2025-03-30T00:42:49.254+00:00
4	2456789356	2192206996	databricks	2025-03-30T00:32:50.110+00:00	databricks	2025-03-30T00:32:50.110+00:00
5	4367854675	4076572128	databricks	2025-03-30T00:32:50.110+00:00	databricks	2025-03-30T00:32:50.110+00:00

If we see above results, ID 1 and 2 are updated with new city names and ID 5 is new record which is added