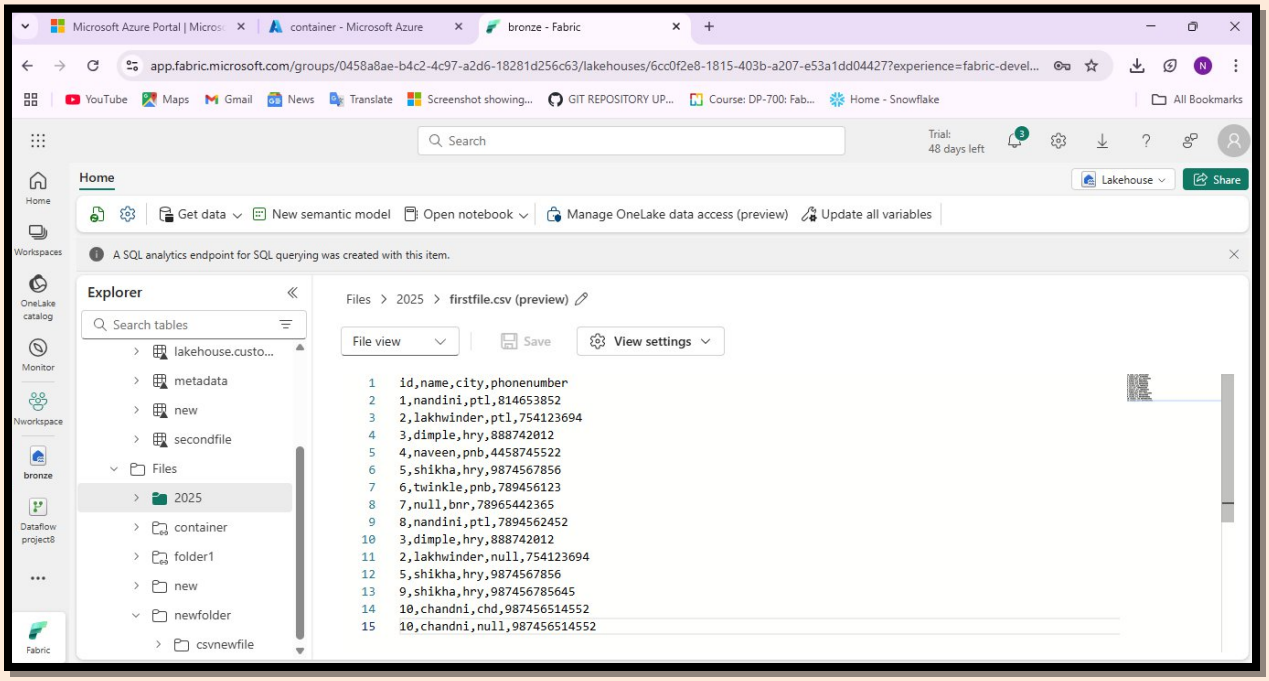
Project – 8

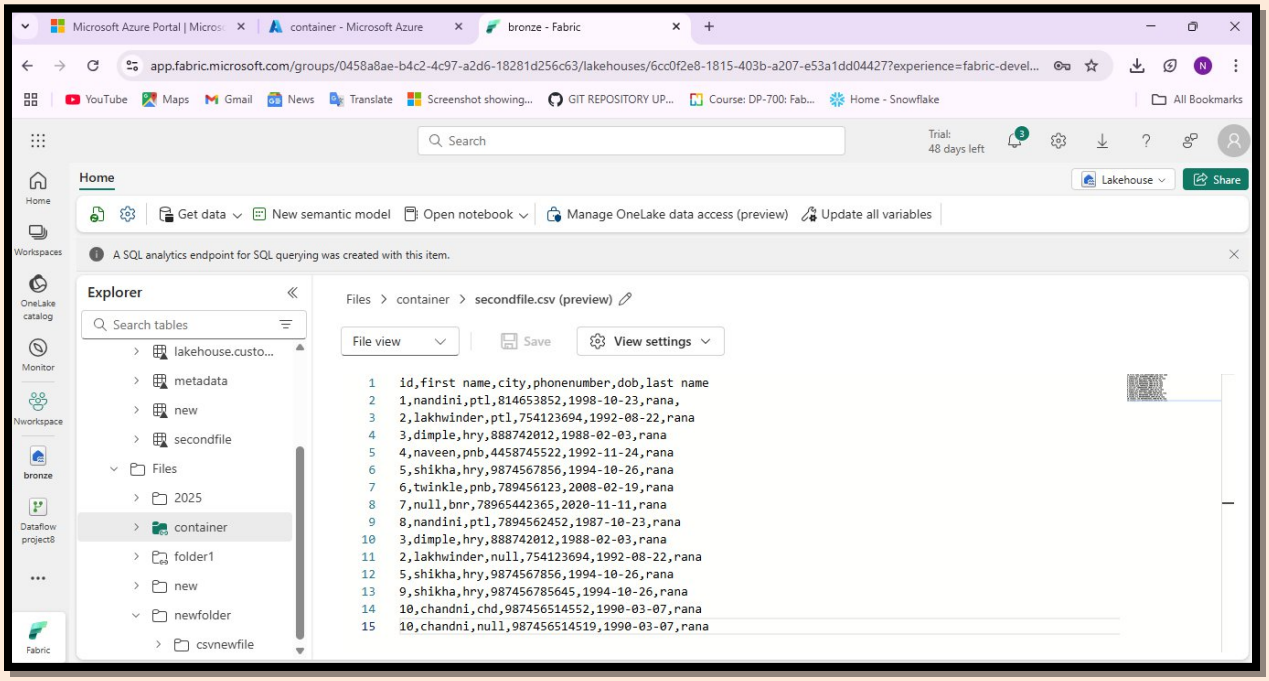
Title: Data Cleaning and SCD Type 1 Processing using Dataflow Gen2 and Spark Notebooks in Microsoft Fabric.

The goal is to build a modern data pipeline using Microsoft Fabric components where raw data is ingested and cleaned using Dataflow Gen2, finally processed in a Spark notebook to apply Slowly Changing Dimension (SCD) Type 1 logic and store the clean, transformed data into a Fabric Warehouse.

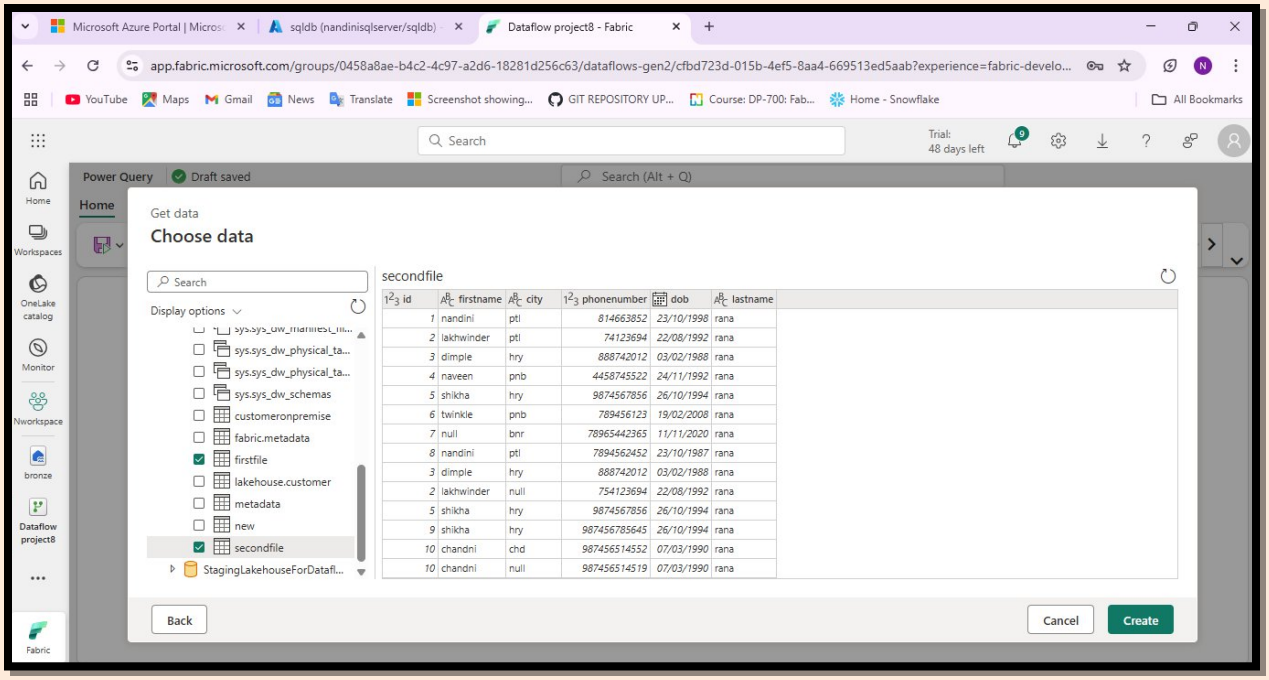
NANDINI RATHORE



CSV FILE IN BRONZE (LAKEHOUSE).

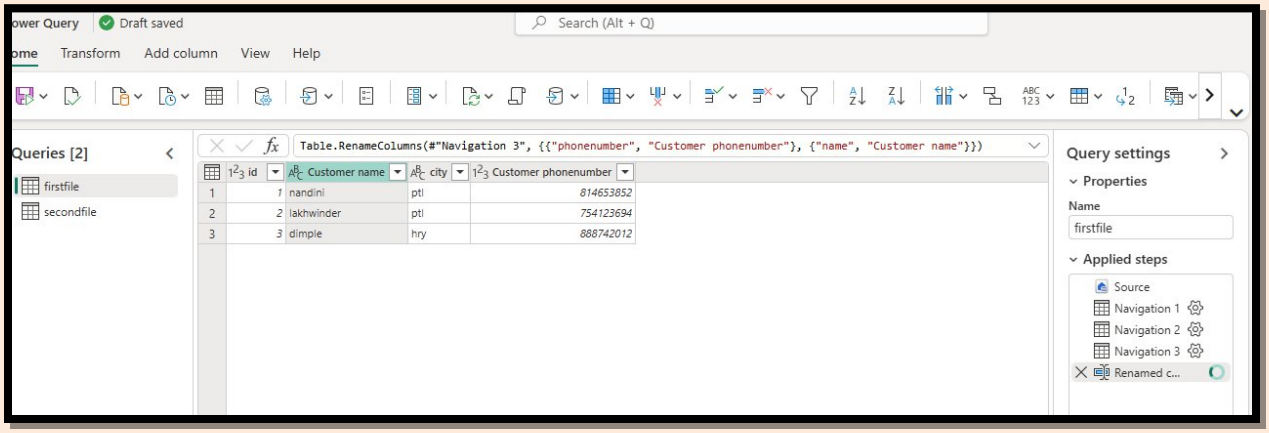


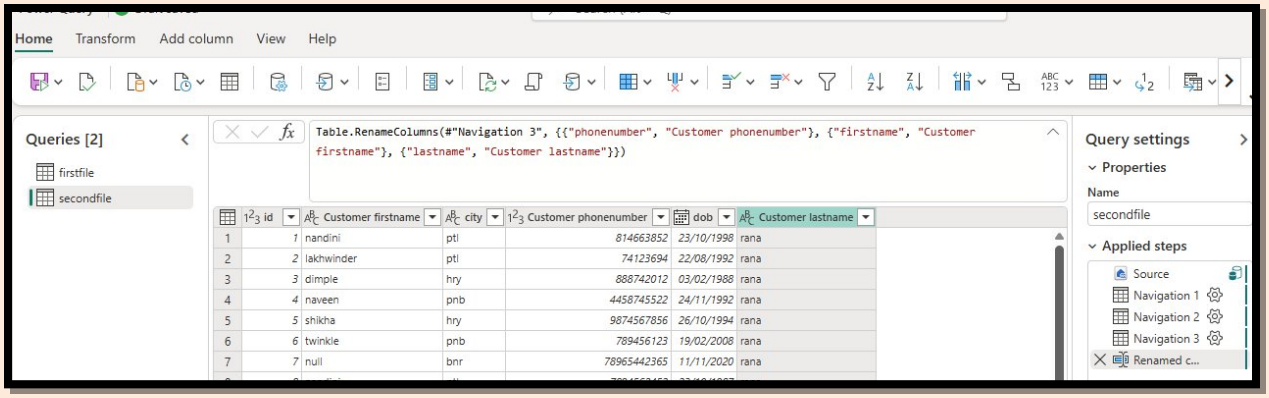
CSV FILE IN BRONZE (LAKEHOUSE).

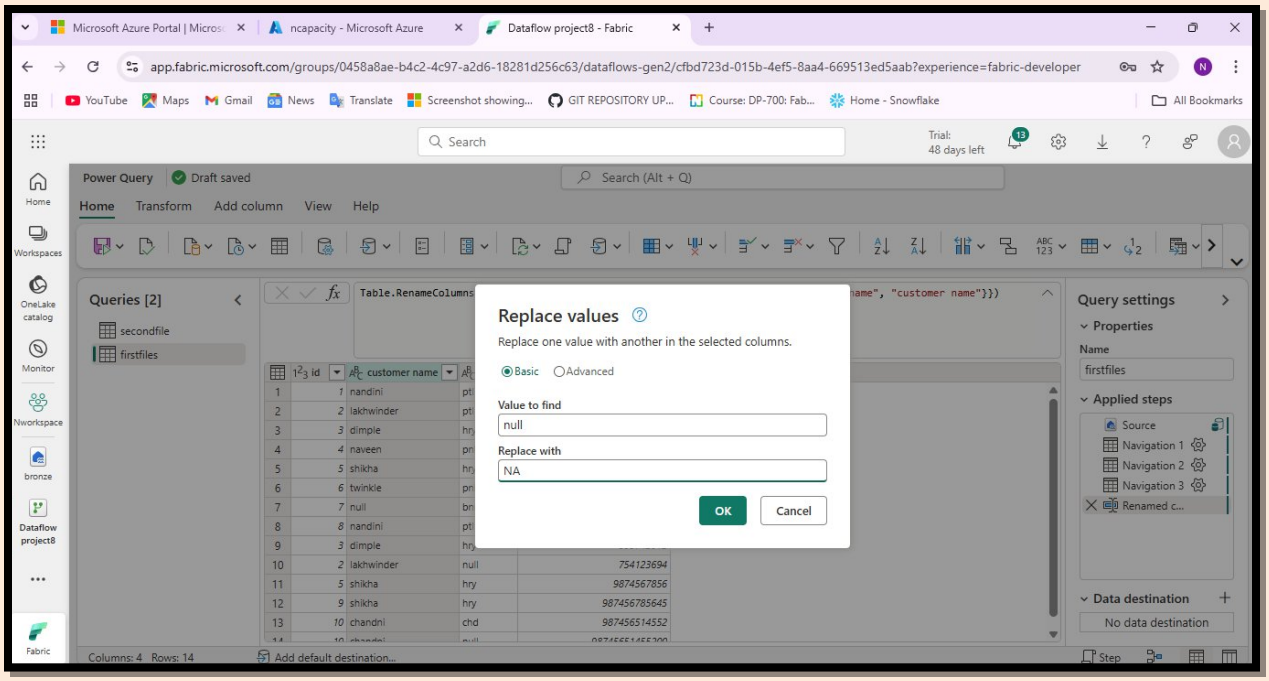


OPEN WORKSPACEàNEW ITEMàDATAFLOWGEN2.

SELECT TWO TABLES FROM LAKEHOUSE.

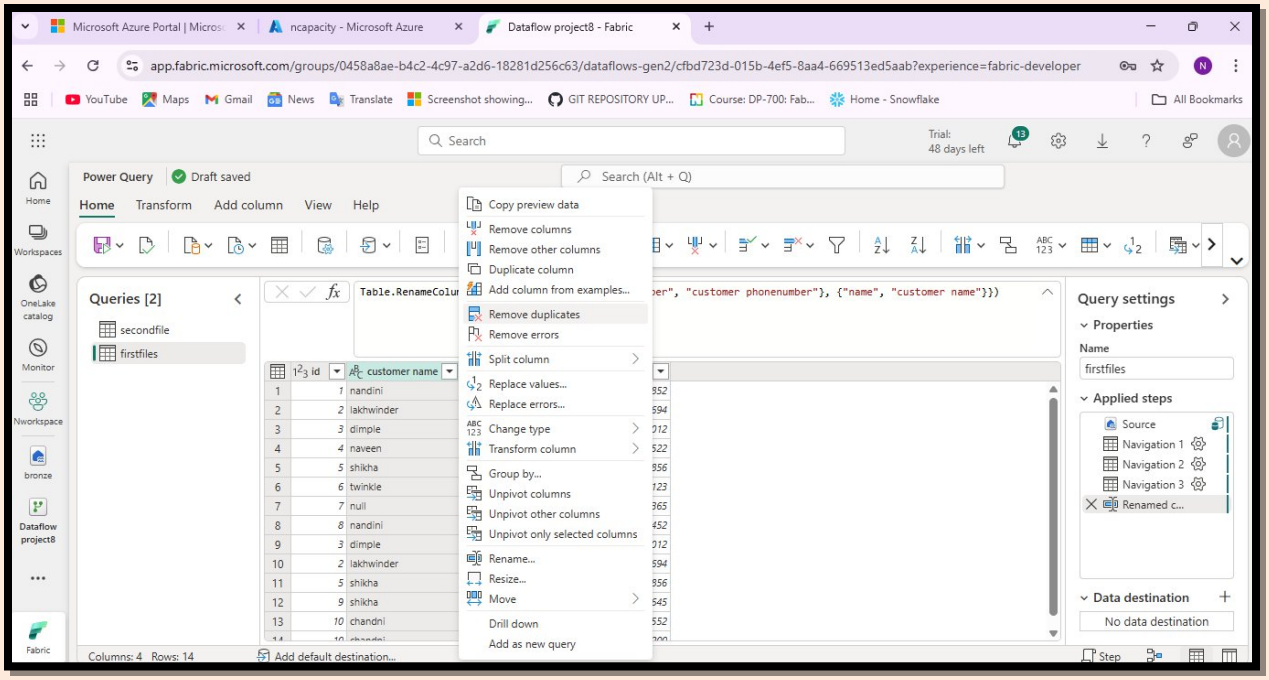




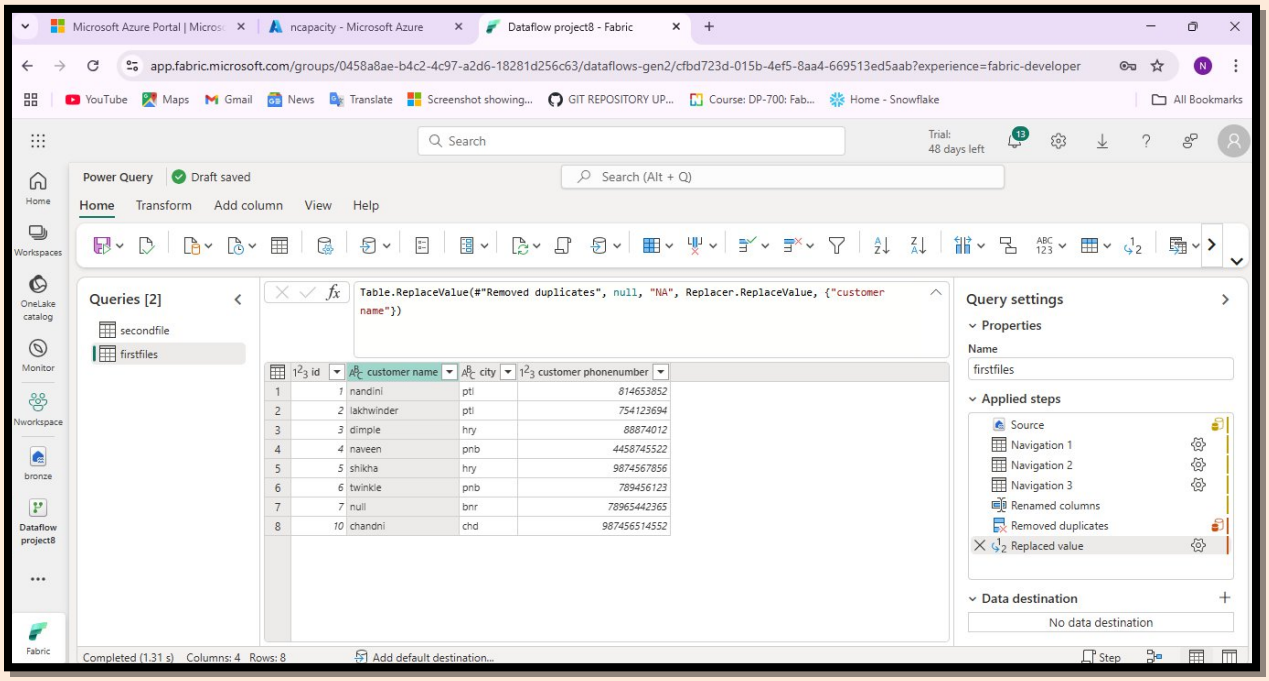


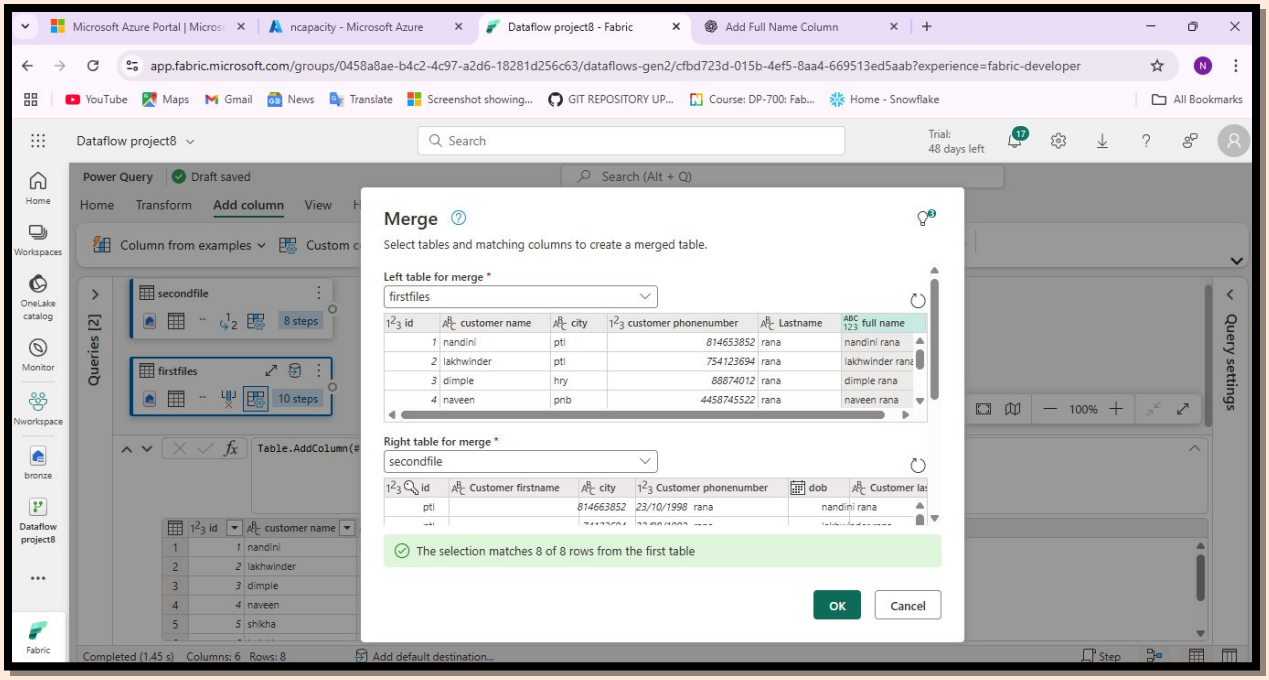
NOW CLEAN THE DATA.

REMOVE NULL VALUES.

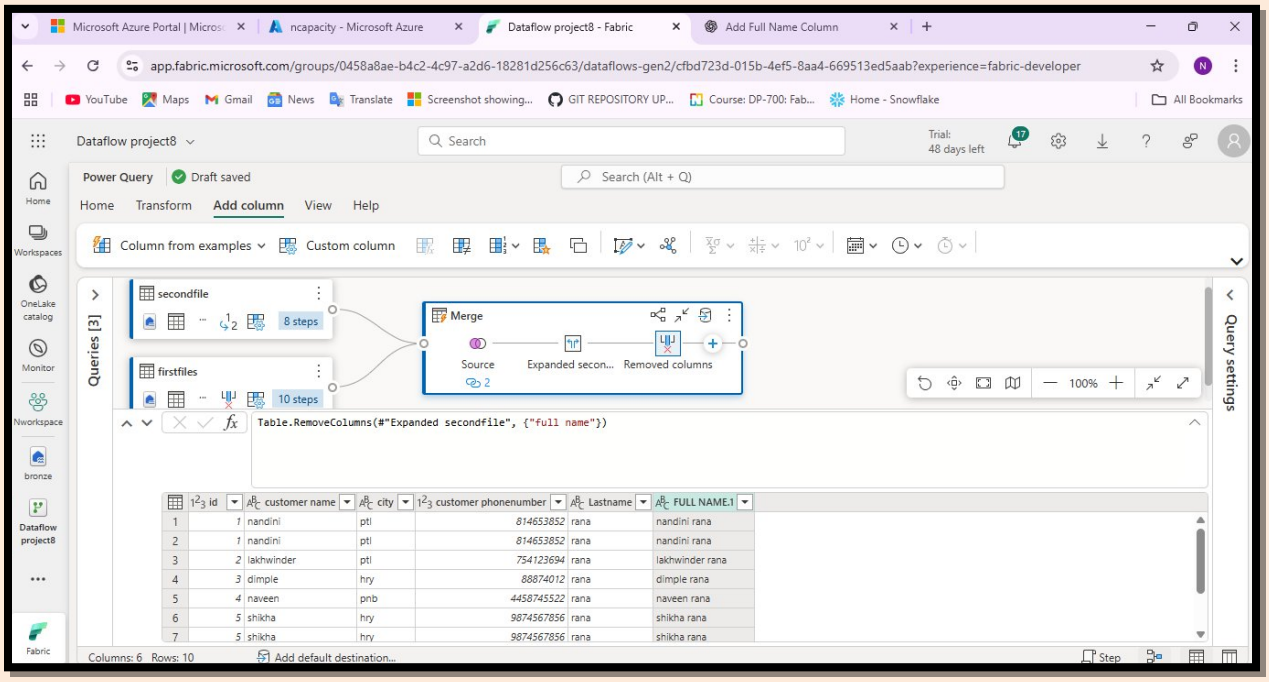


REMOVE DUPLICATES: : Duplicate rows are identified based on key columns and removed to ensure data uniqueness.

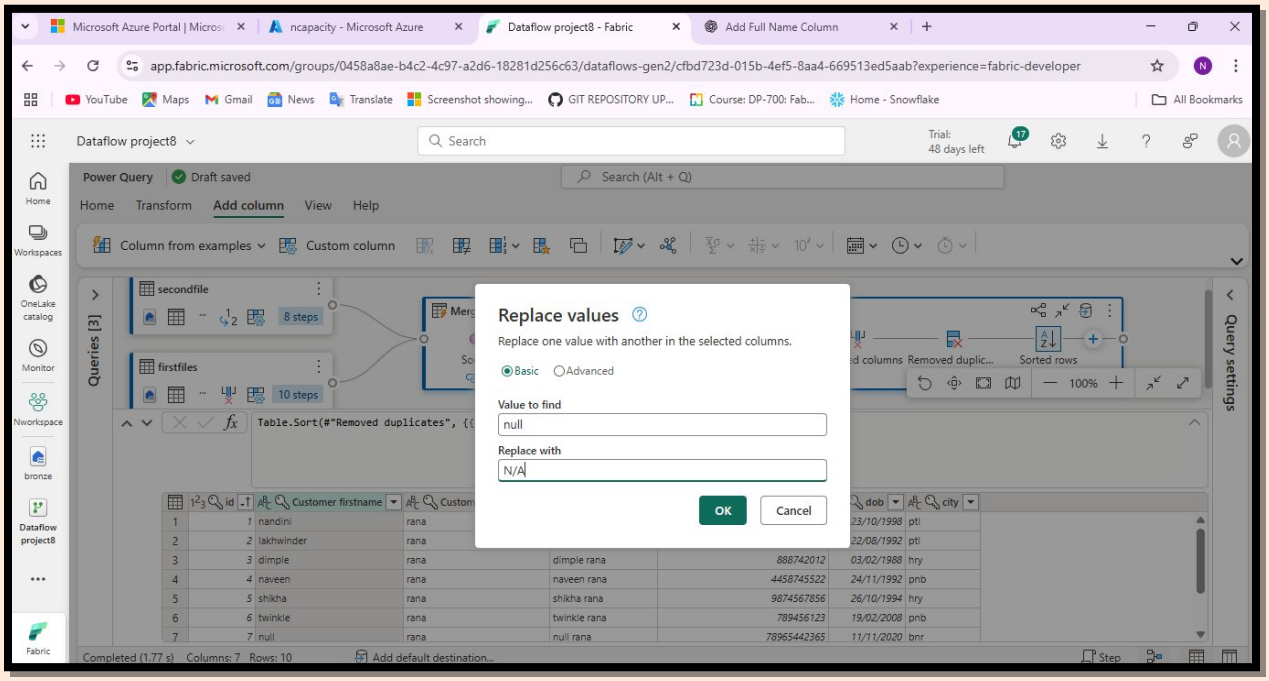


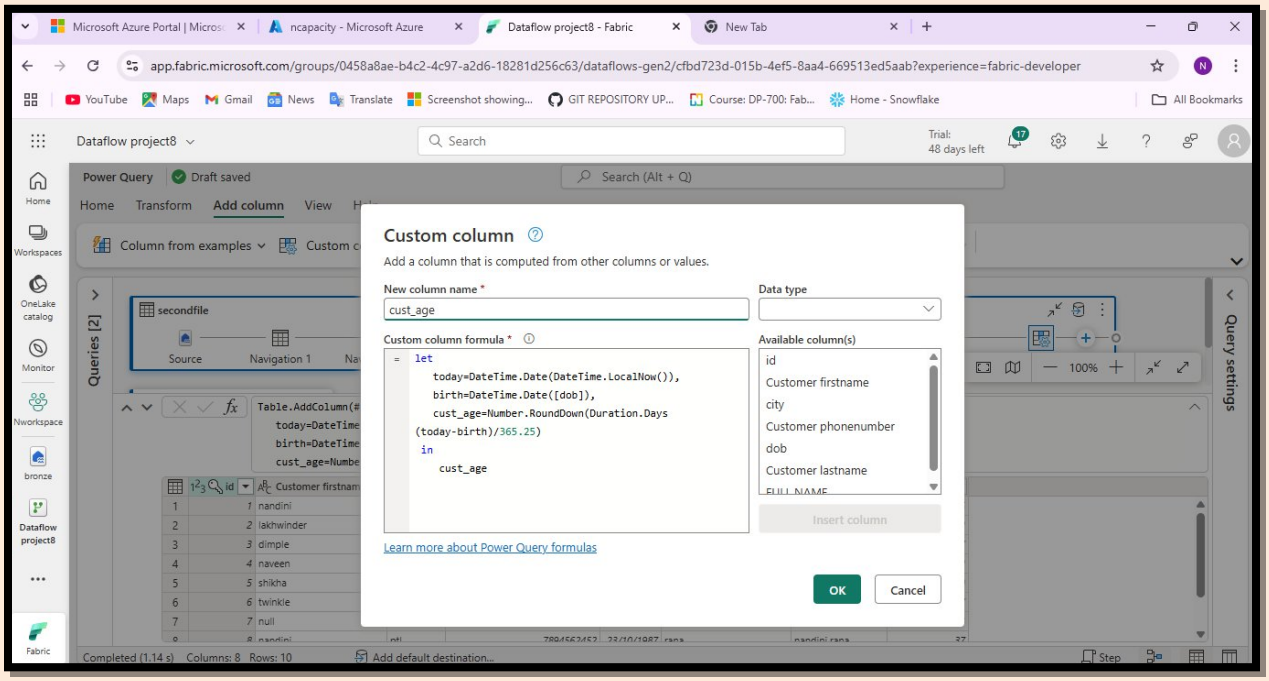


MERGE TWO TABLES ON THE BASIS OF ONE COLUMN: Data from different tables or sources is joined to create a consolidated dataset using common key columns.

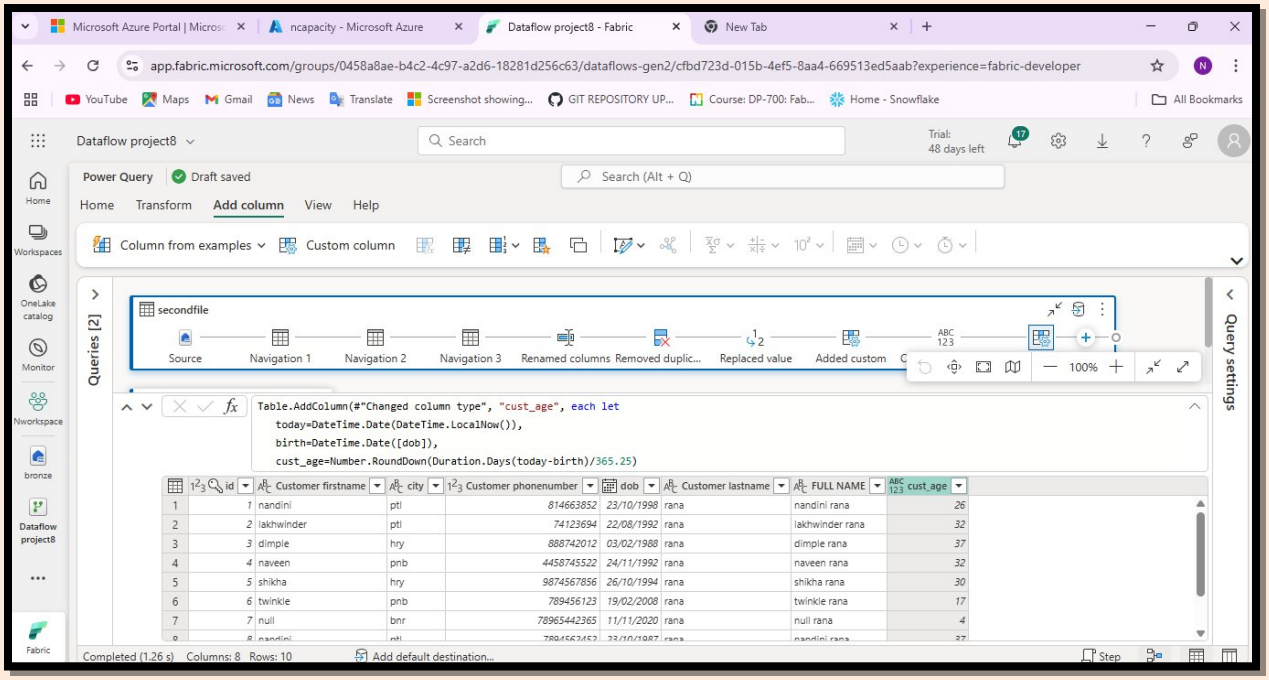


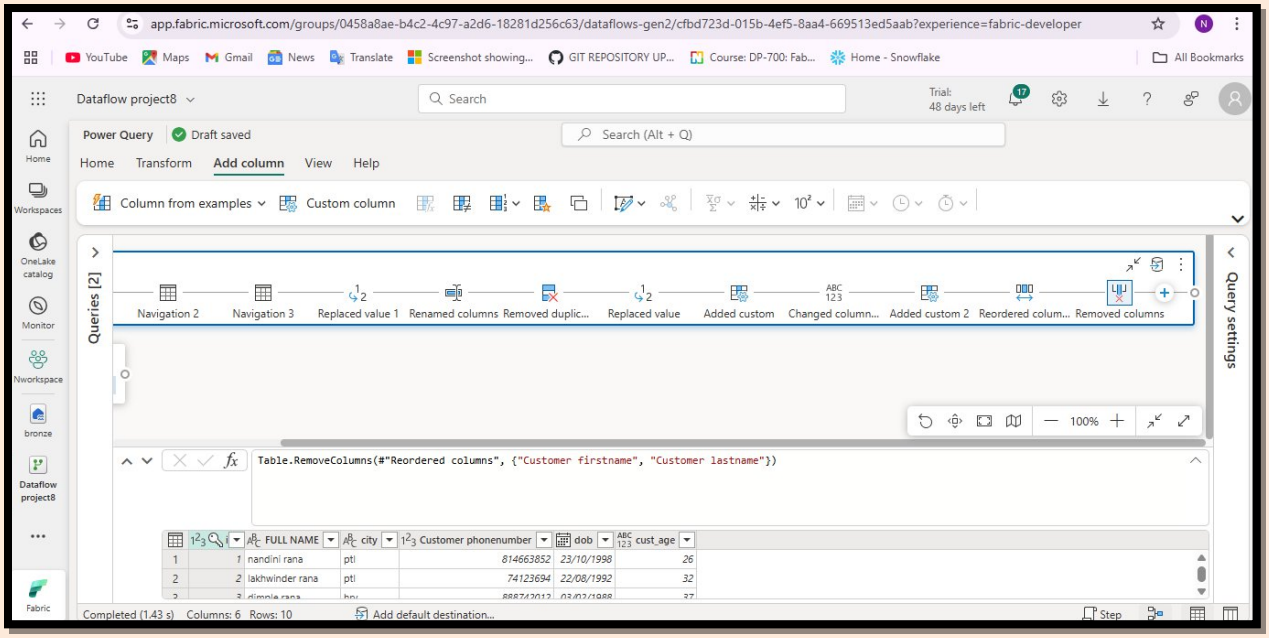
REMOVE UNNECESSARY COLUMNS: Unused or irrelevant columns are removed to reduce data size and improve performance.



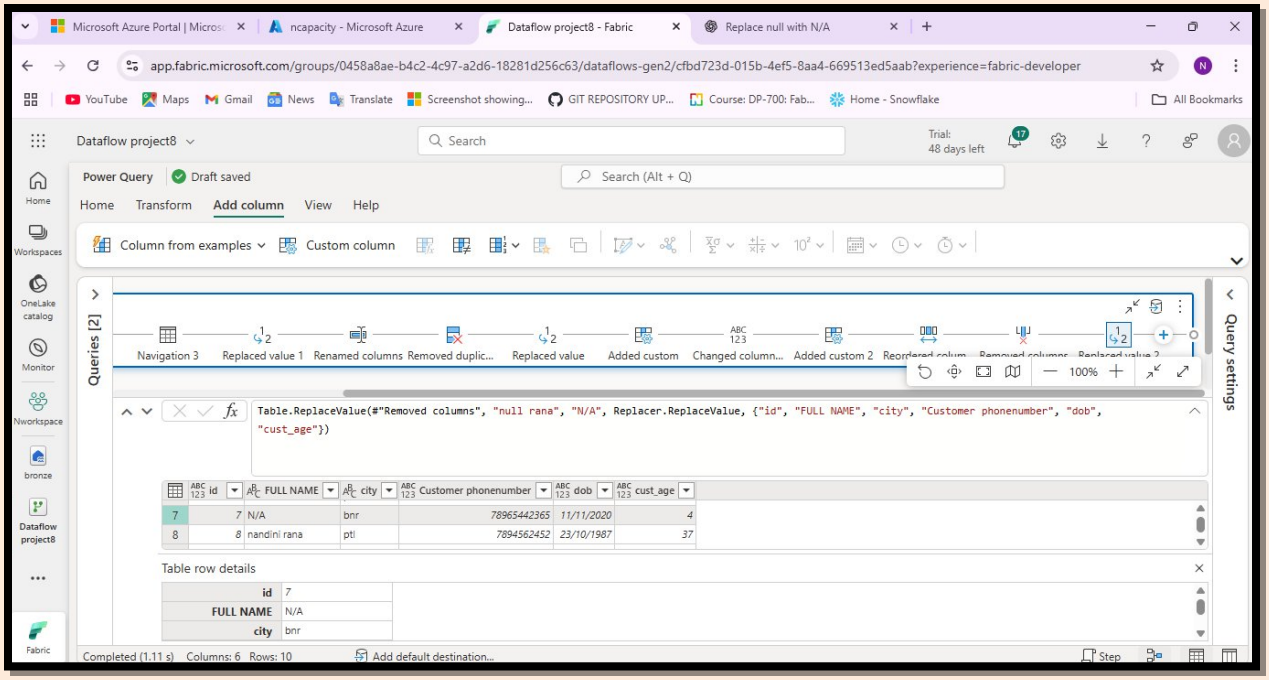


age calculated from date of birth.

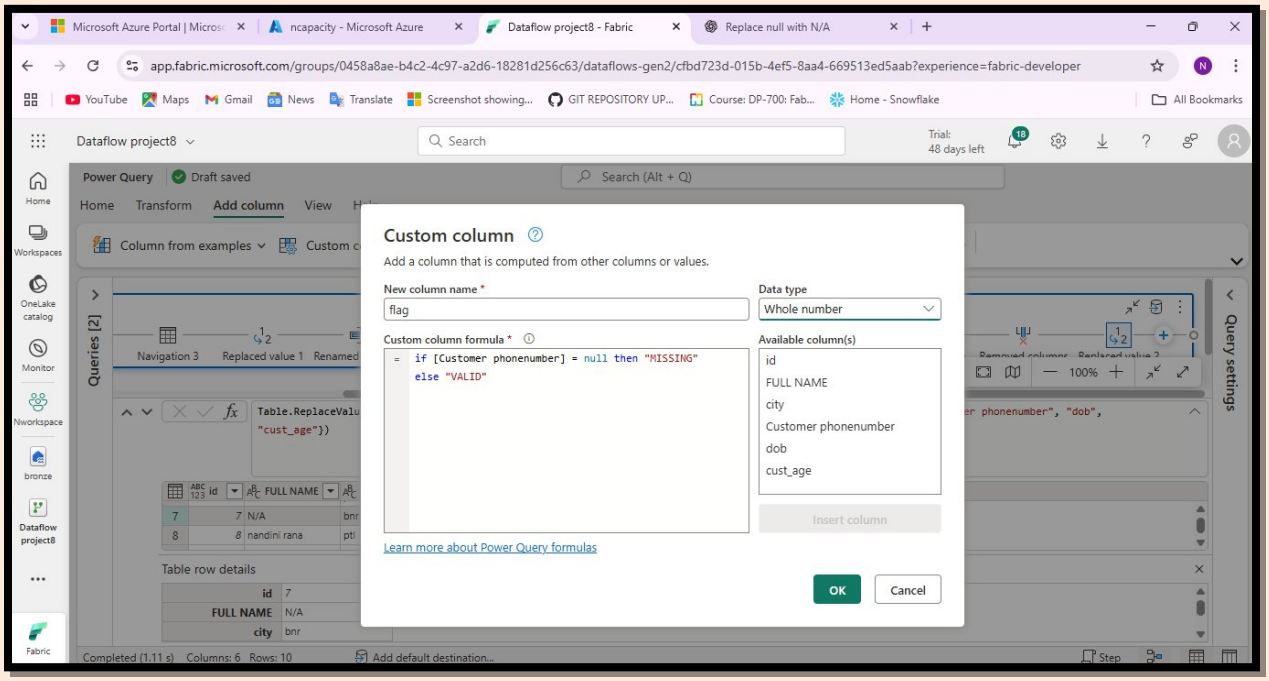




Full name columns derived from first and last name.

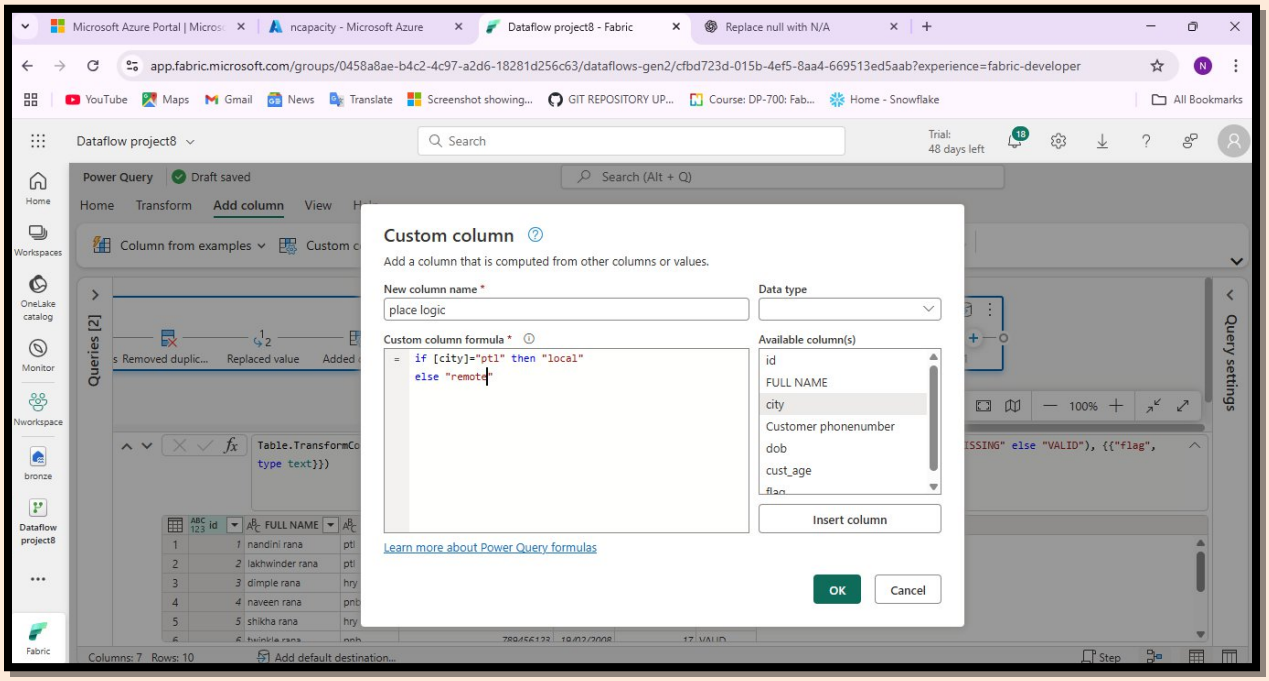


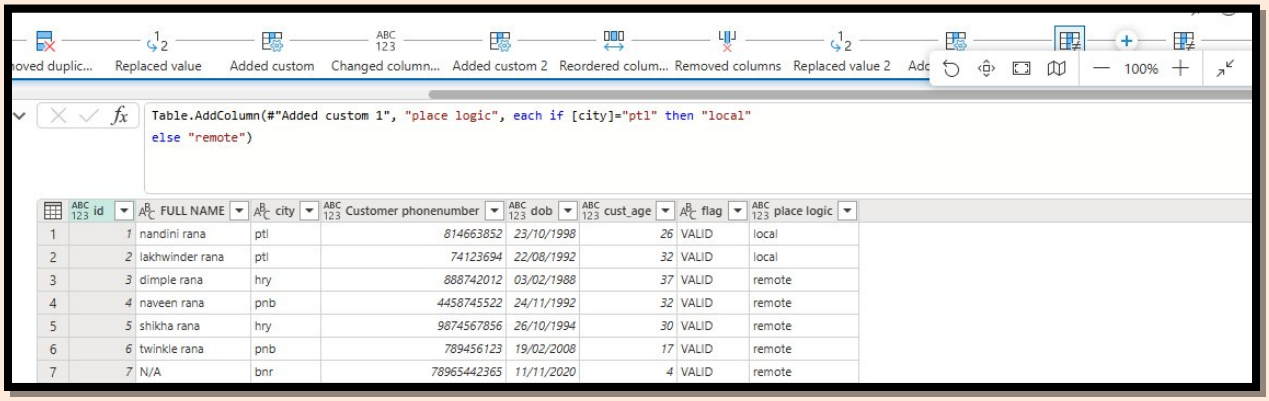
Replacing Null Values: Missing or null values in the dataset are replaced with appropriate default values, such as zeros or placeholder text.



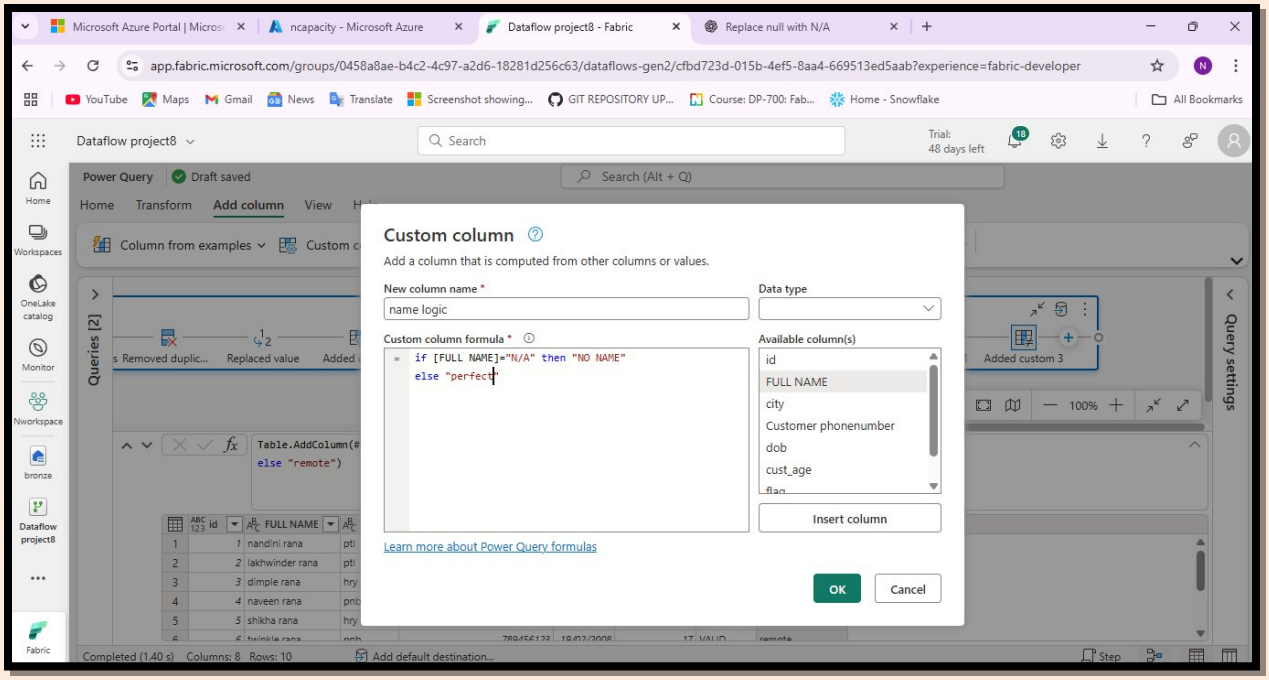
Flags or status indicators based on conditional logic.

ADD COLUMNSàCUSTOM COLUMNSàWRITE THE FORMULA TO GET CONDITIONAL LOGIC.



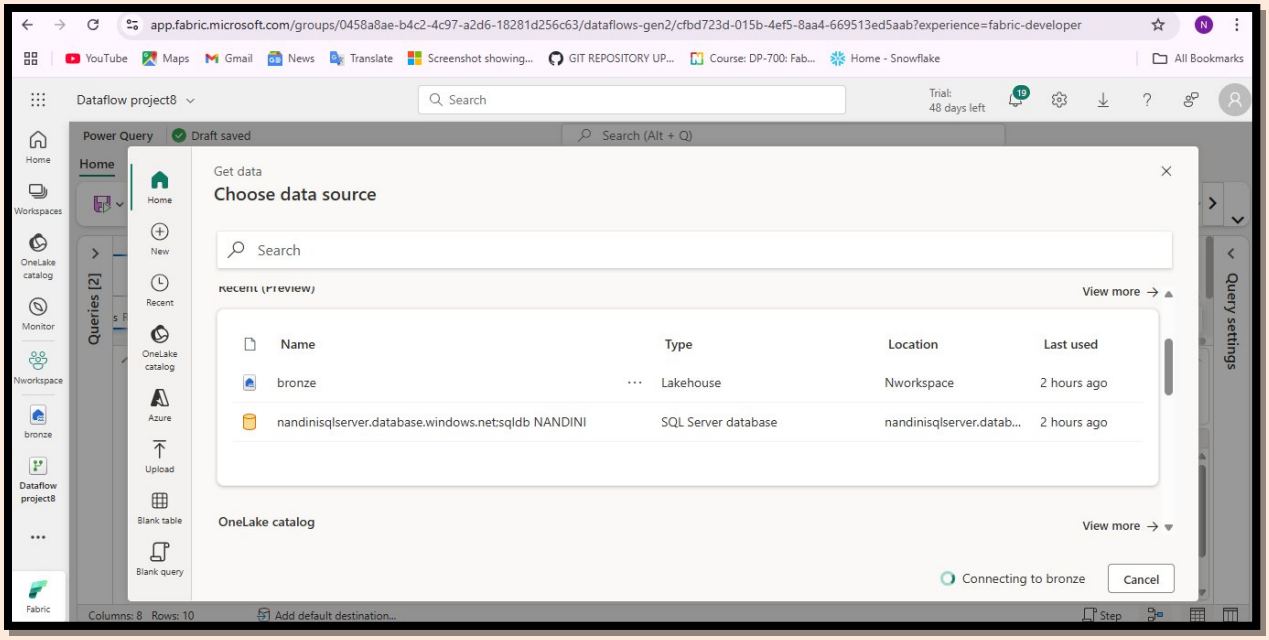


	id	FULL NAME	city	Customer phonenumber	dob	cust_age	flag	place logic
1	1	nandini rana	ptl	814663852	23/10/1998	26	VALID	local
2	2	lakhwinder rana	ptl	74123694	22/08/1992	32	VALID	local
3	3	dimple rana	hry	888742012	03/02/1988	37	VALID	remote
4	4	naveen rana	pnb	4458745522	24/11/1992	32	VALID	remote
5	5	shikha rana	hry	9874567856	26/10/1994	30	VALID	remote
6	6	twinkle rana	pnb	789456123	19/02/2008	17	VALID	remote
7	7	N/A	bnr	78965442365	11/11/2020	4	VALID	remote

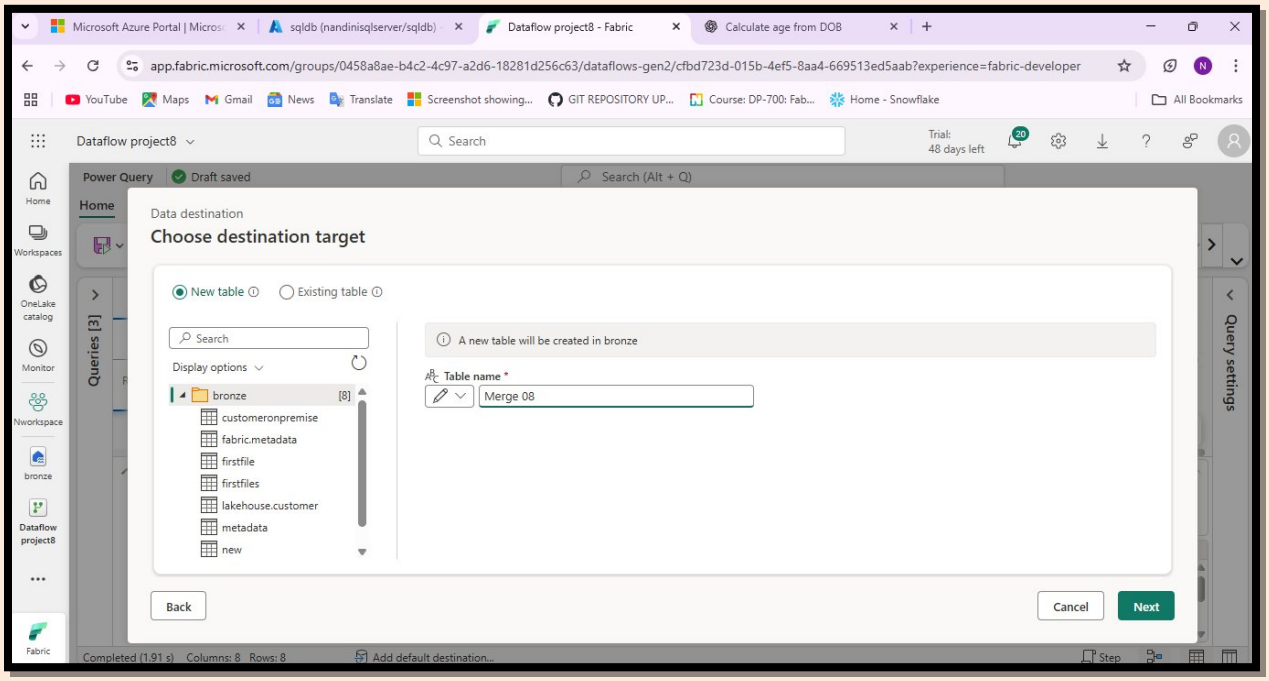



```
Table.AddColumn(#"Added custom 3", "name logic", each if [FULL NAME]="N/A" then "NO NAME"
else "perfect")
```

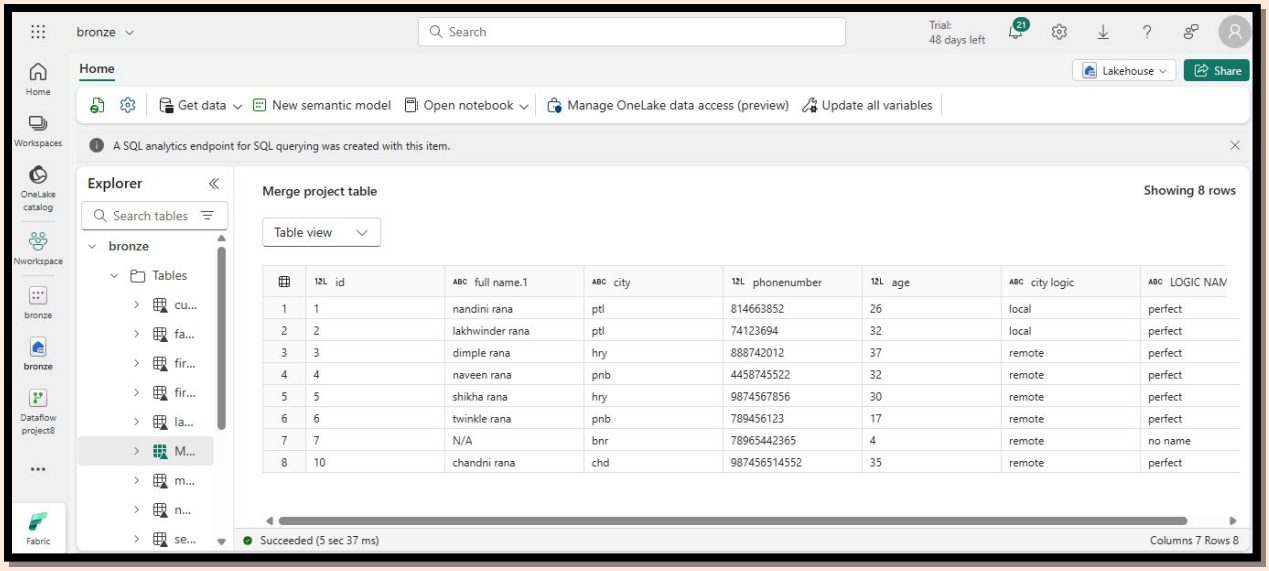
id	FULL NAME	city	Customer phonenumber	dob	cust_age	flag	place logic	name logic
3	dimple rana	hry	888742012	03/02/1988	37	VALID	remote	perfect
4	naveen rana	pnb	4458745522	24/11/1992	32	VALID	remote	perfect
5	shikha rana	hry	9874567856	26/10/1994	30	VALID	remote	perfect
6	twinkle rana	pnb	789456123	19/02/2008	17	VALID	remote	perfect
7	N/A	bnr	78965442365	11/11/2020	4	VALID	remote	NO NAME
8	nandini rana	ptl	7894562452	23/10/1987	37	VALID	local	perfect

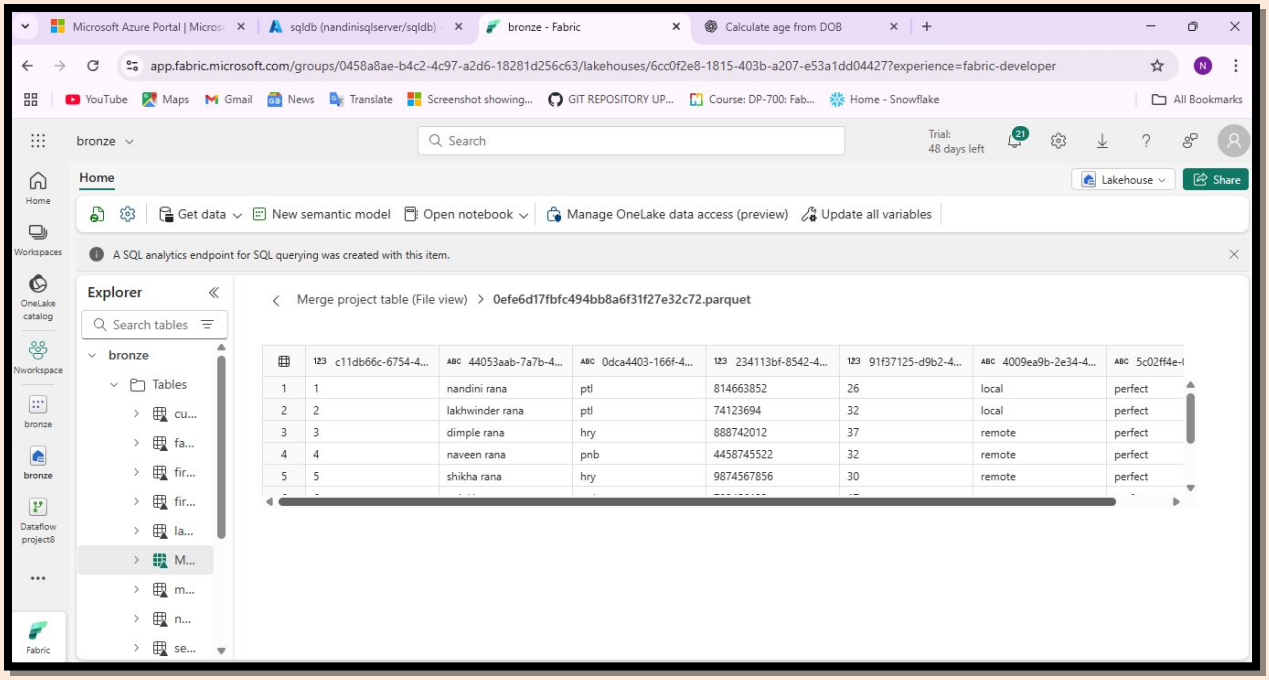


NOW LOAD THIS CLEAN DATA TO FINAL DESTINATION àLAKEHOUSE.

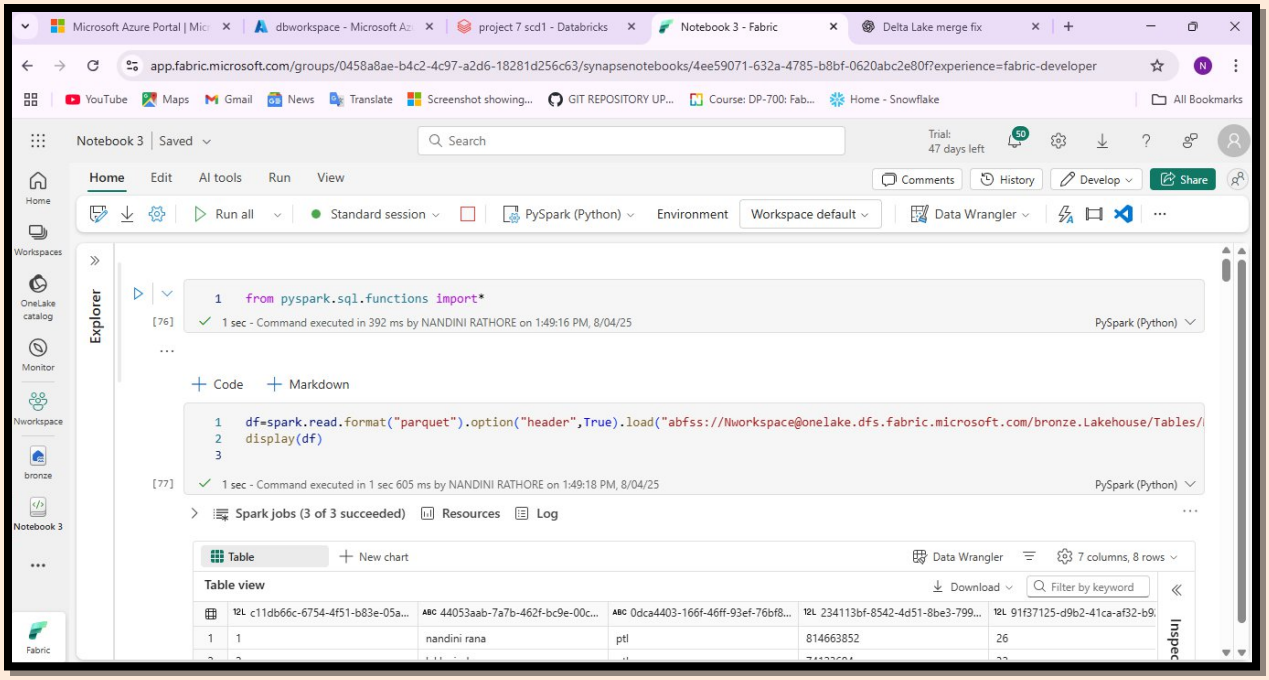


SAVE TABLE.





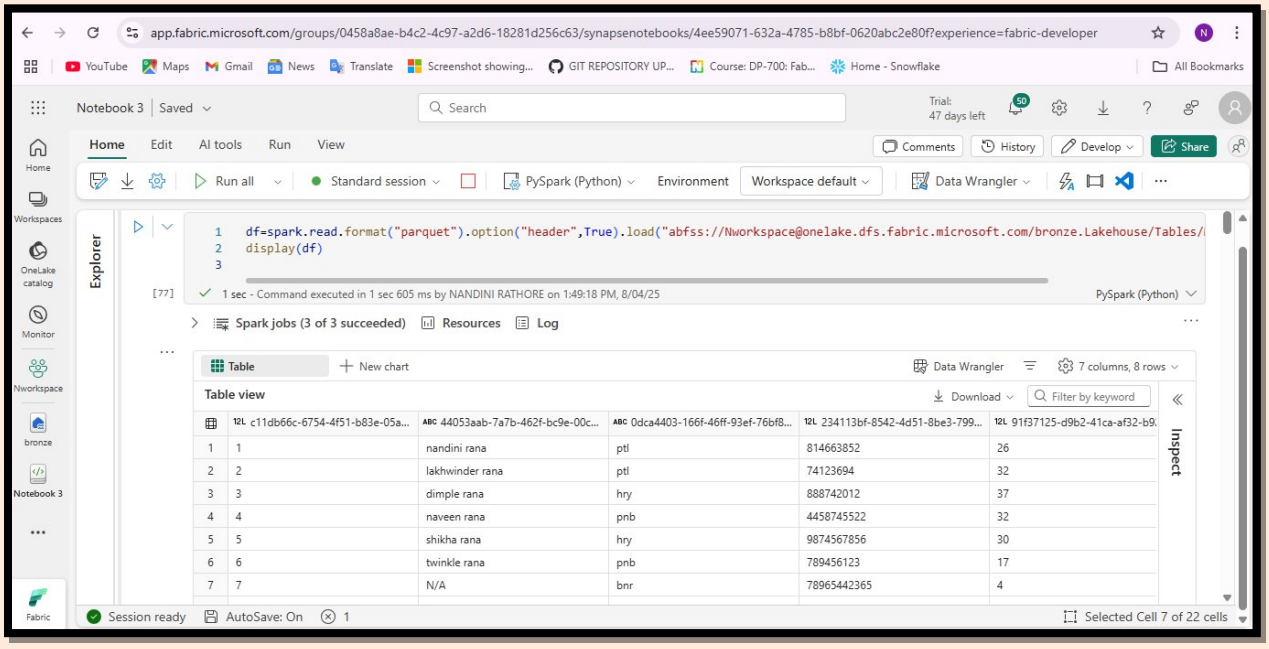
WHEN CLICK ON FILE VIEW IT SHOWS PARQUET FILE FORMAT.



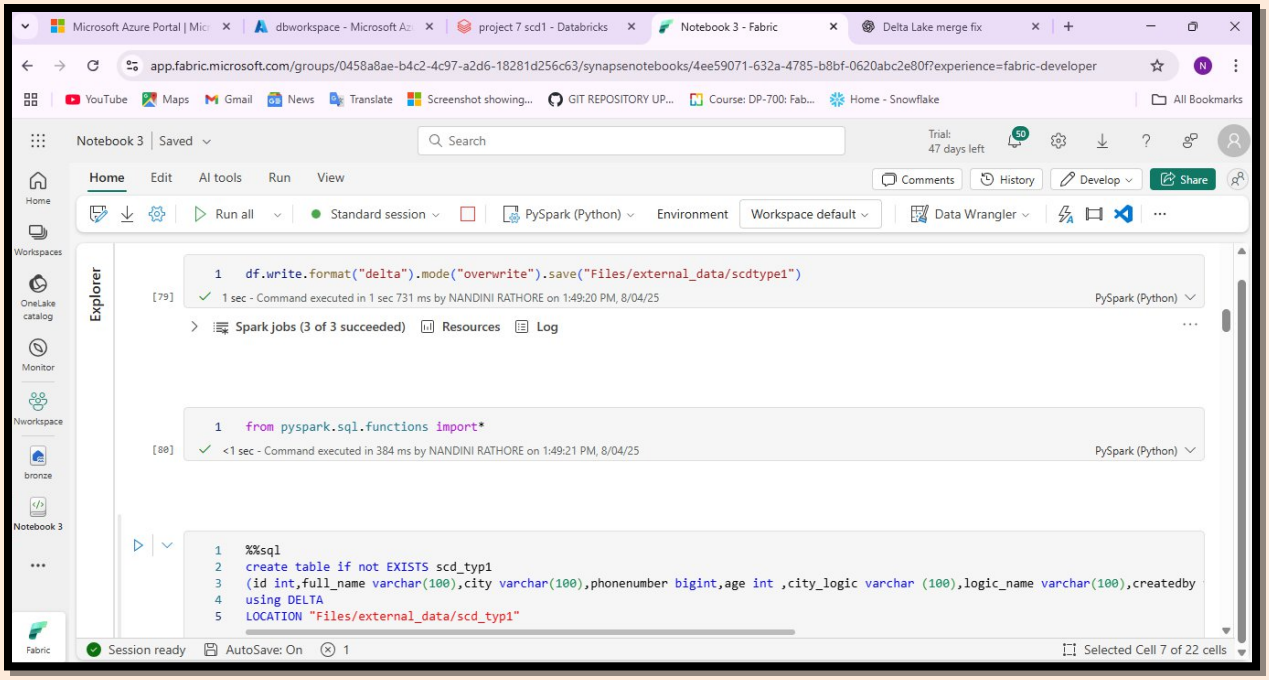
NOW CLICK ON NEW ITEMàNOTEBOOK.

PERFORM Scdtype1.

IMPORT FUNCTIONS.



READ SOURCE FILE.

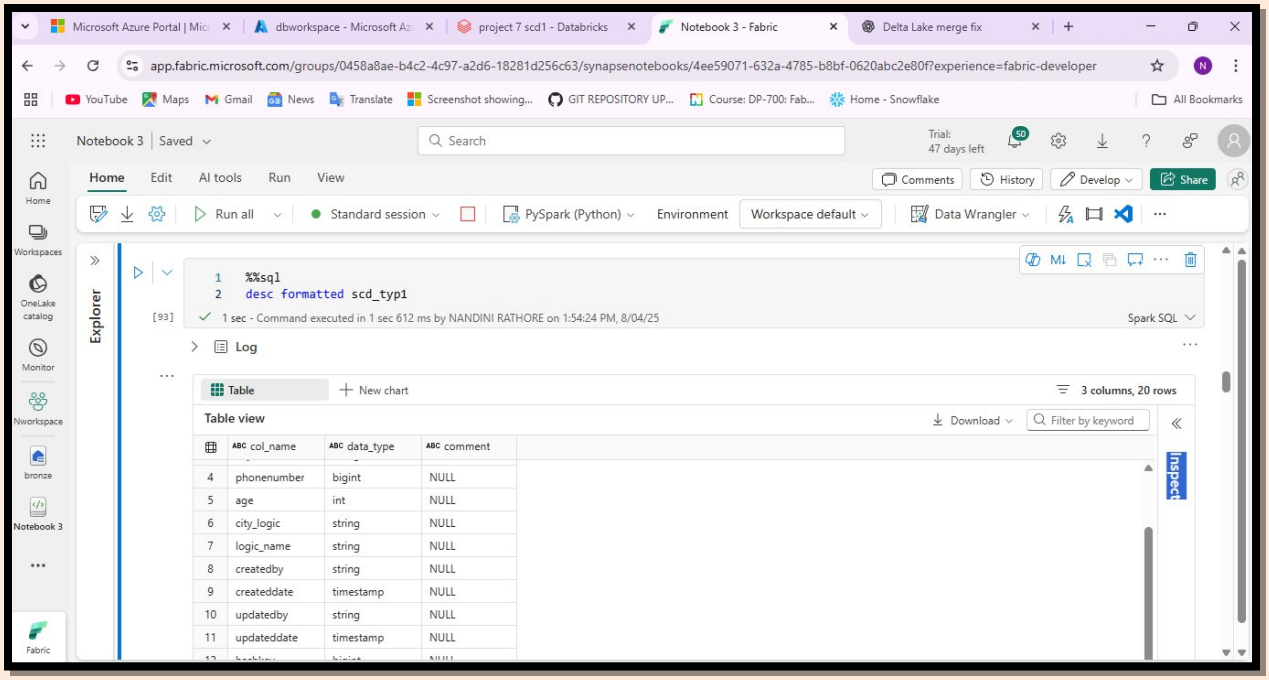


WRITE THAT FILE IN DELTA FORMAT.

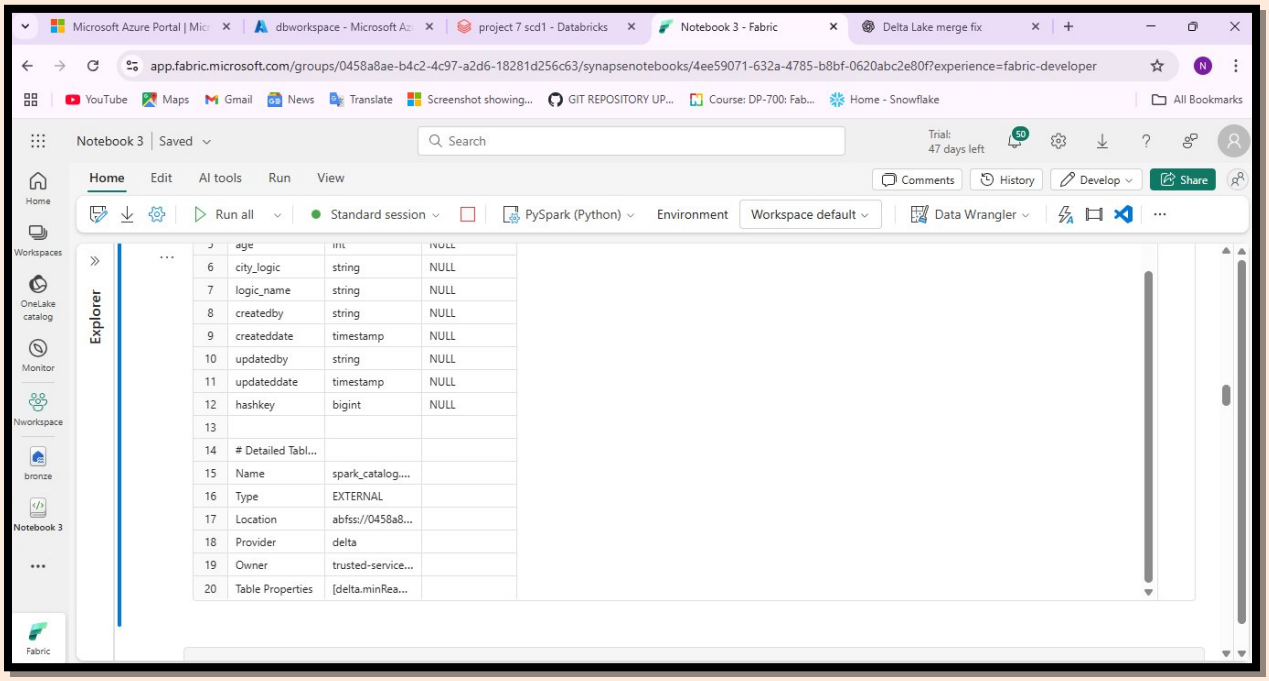
```
df=spark.read.format("parquet").option("header",True).load("abfss://Nworkspace@onelake.dfs.fabric.microsoft.com/bronze.Lakehouse/Tables/Merge project table")
display(df)
```

CREATE A TARGET EXTERNAL DELTA TABLE .

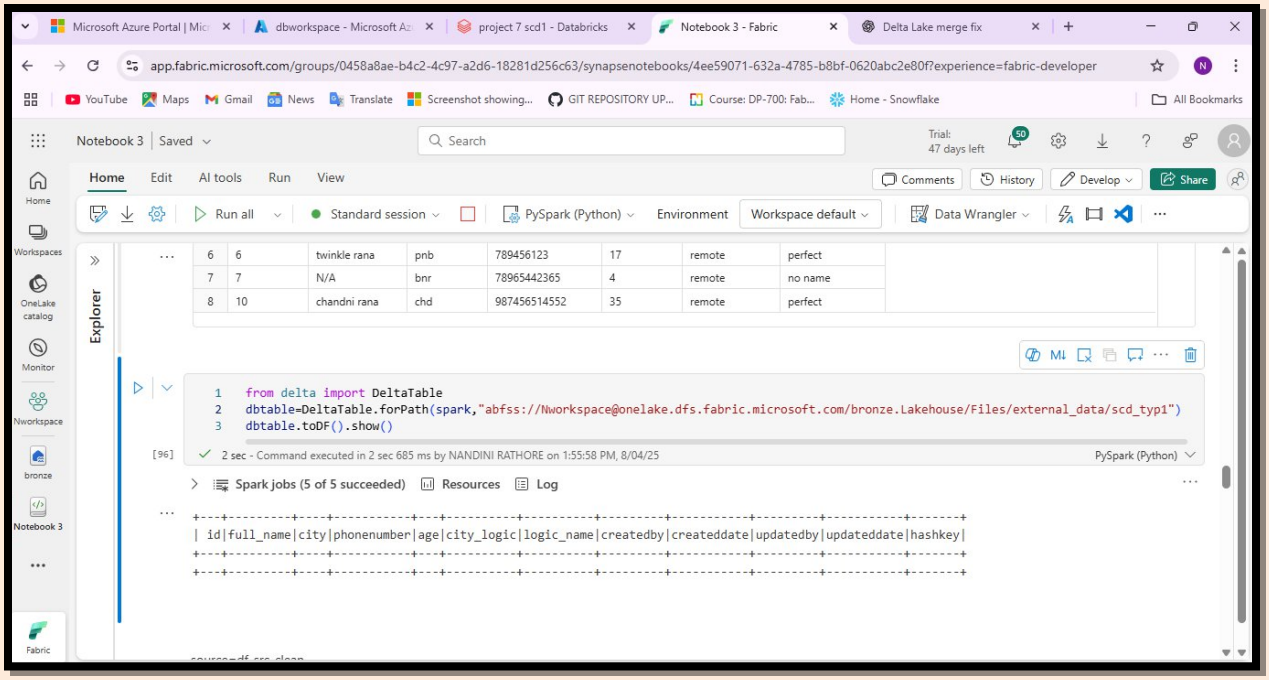
```
%%sql
create table if not EXISTS scd_typ1
(id int,full_name varchar(100),city varchar(100),phonenumber bigint,age int ,city_logic varchar (100),logic_name varchar(100),createdby varchar(100),createddate timestamp,updatedby varchar(100),updateddate timestamp,hashkey bigint)
using DELTA
LOCATION "Files/external_data/scd_typ1"
```



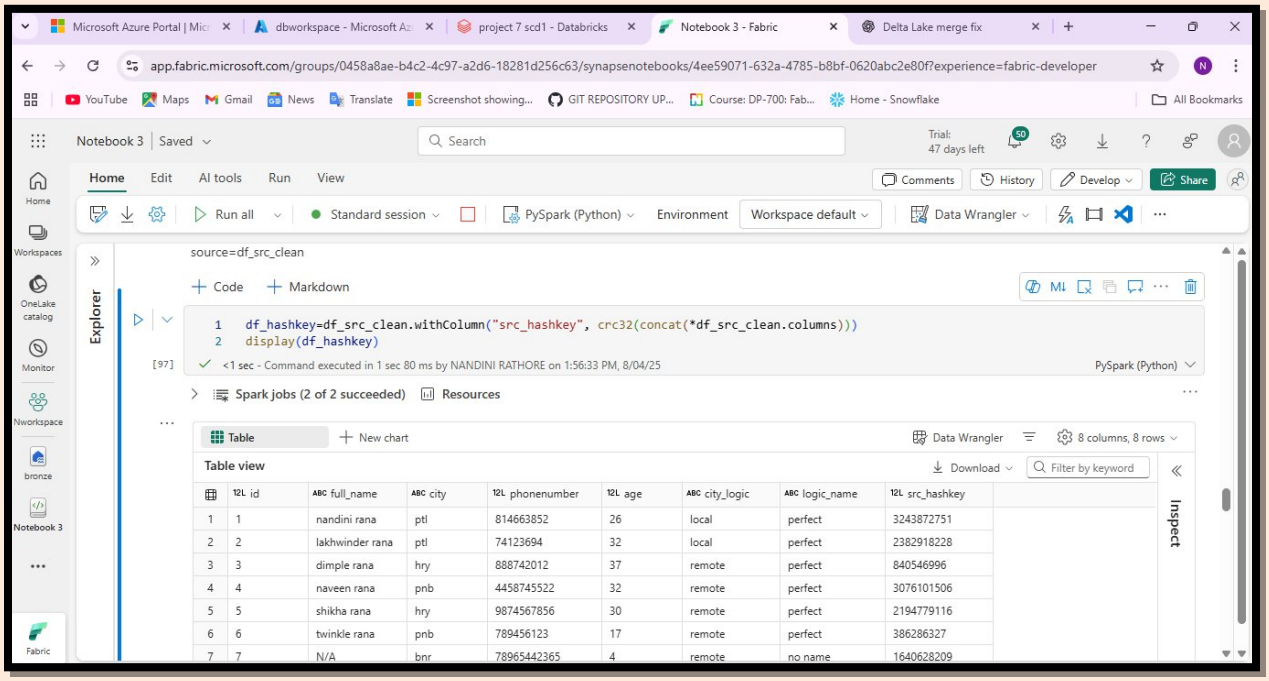
WE USE THIS COMMAND TO KNOW TYPE OF THE TABLE.



DELTA TABLE IS CREATED.

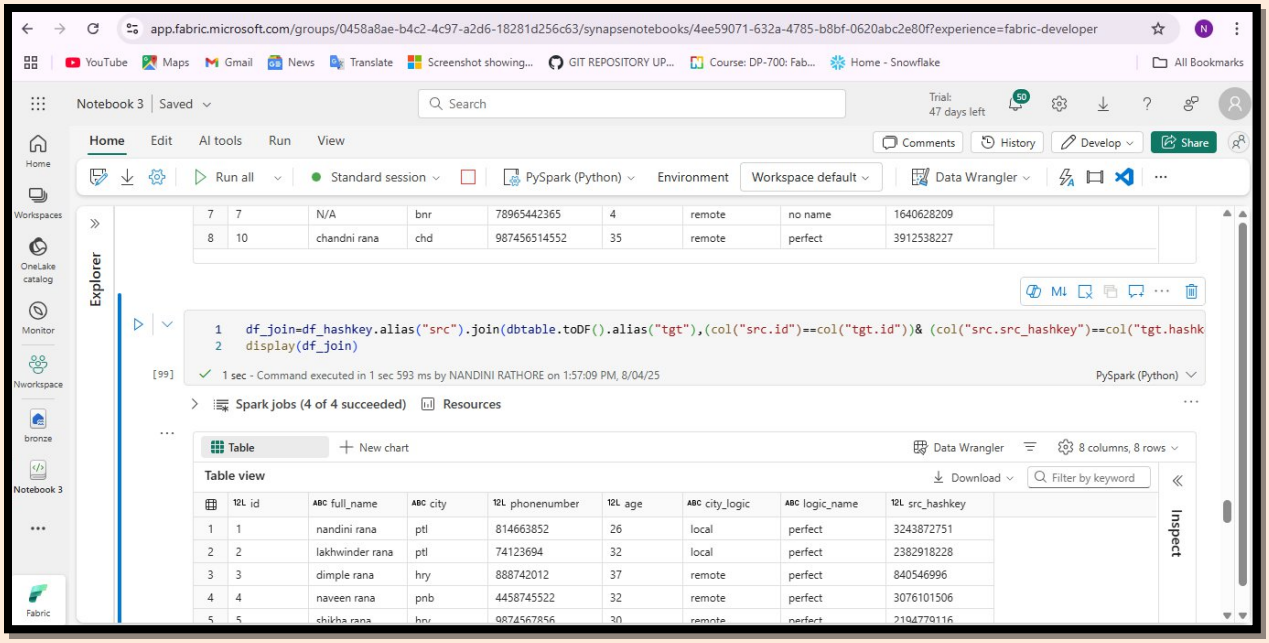


At first target data is empty because no new data is added.

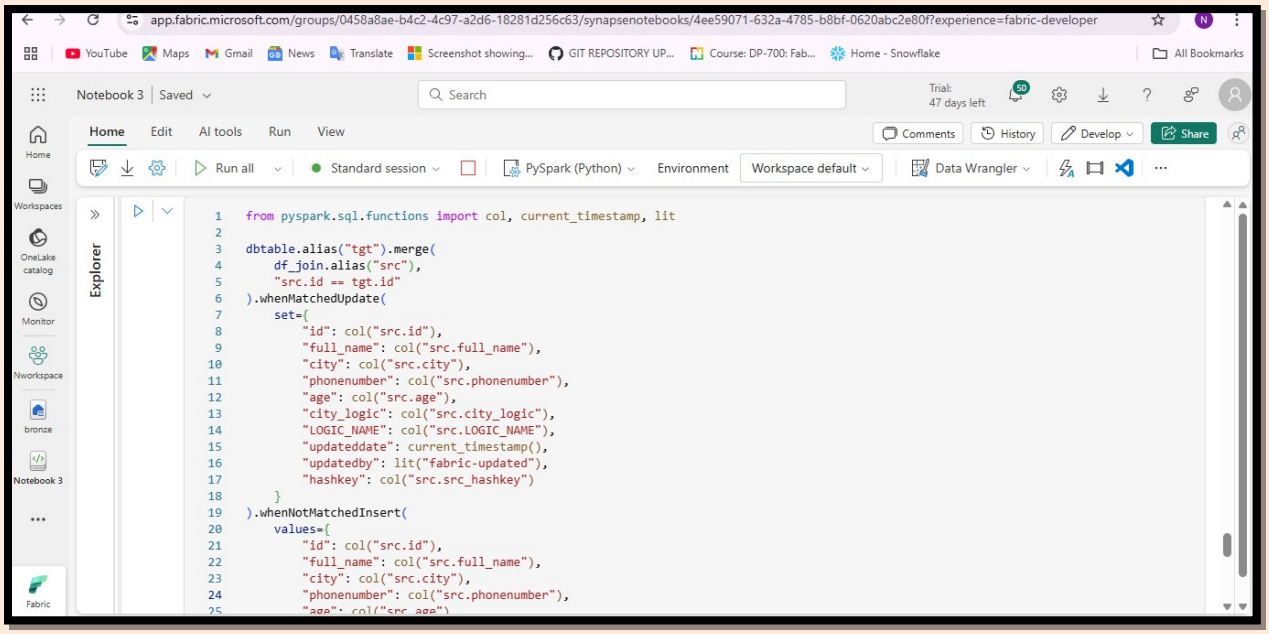


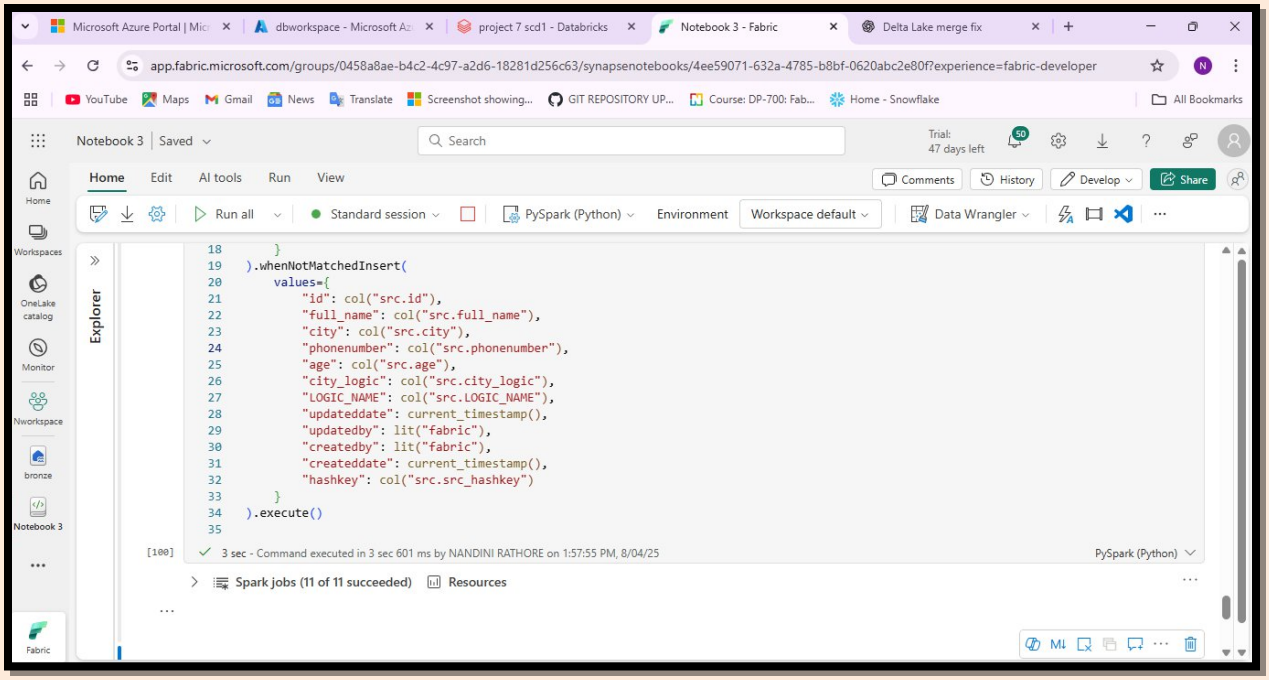
GENERATING HASHKEY:

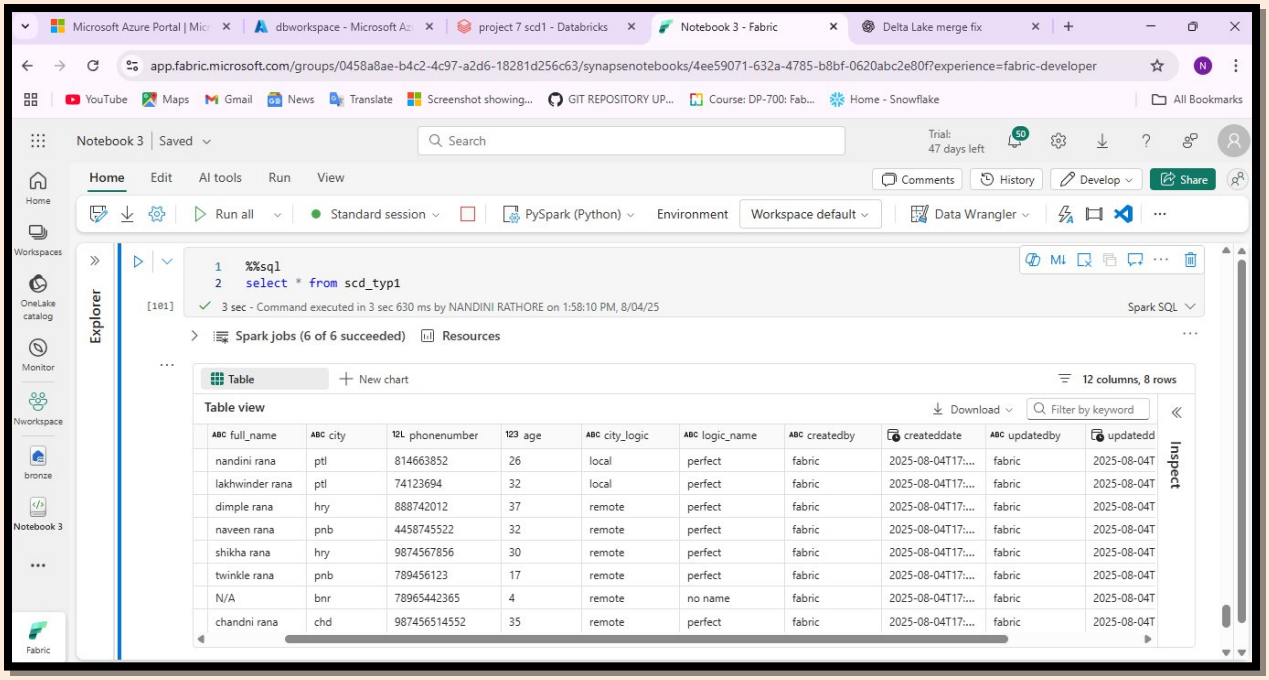
```
df_hashkey=df_src_clean.withColumn("src_hashkey", crc32(concat(*df_src_clean.columns)))
display(df_hashkey)
```



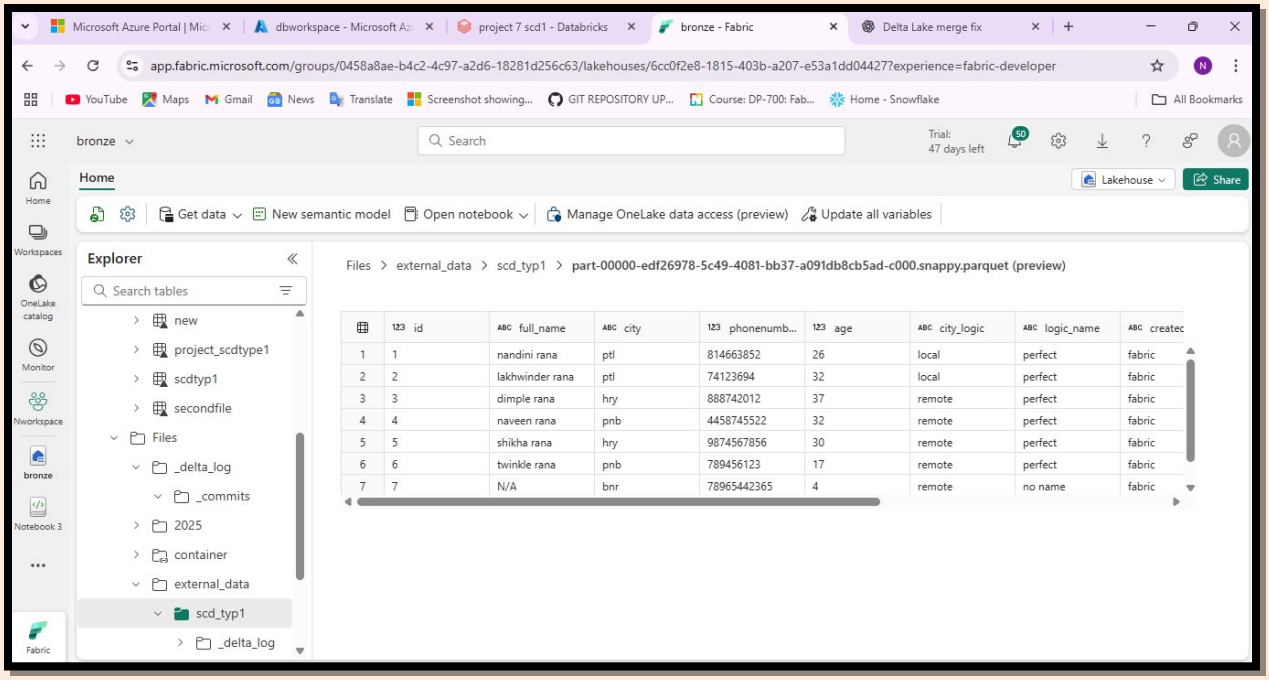
- df_hashkey ☾ Your source DataFrame containing new or updated records.
- .alias ("src") ☾ Assigns the alias src to the source DataFrame so you can reference it in joins.
- dbtable. toDF () ☾ Converts a database table into a PySpark DataFrame.
- .alias("tgt") ☾ Assigns the alias tgt to the target (existing) table.
- col("src.id") == col("tgt.id") ☾ Join condition: matches records with the same ID in both source and target.
- "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.



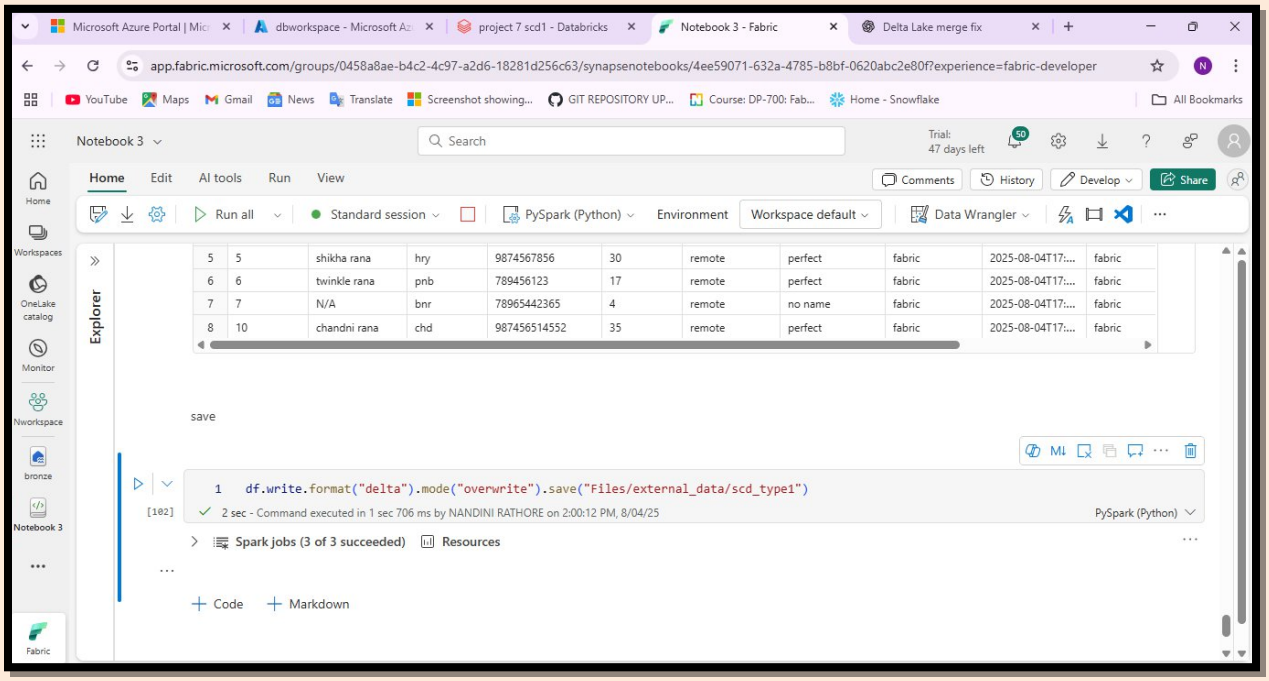


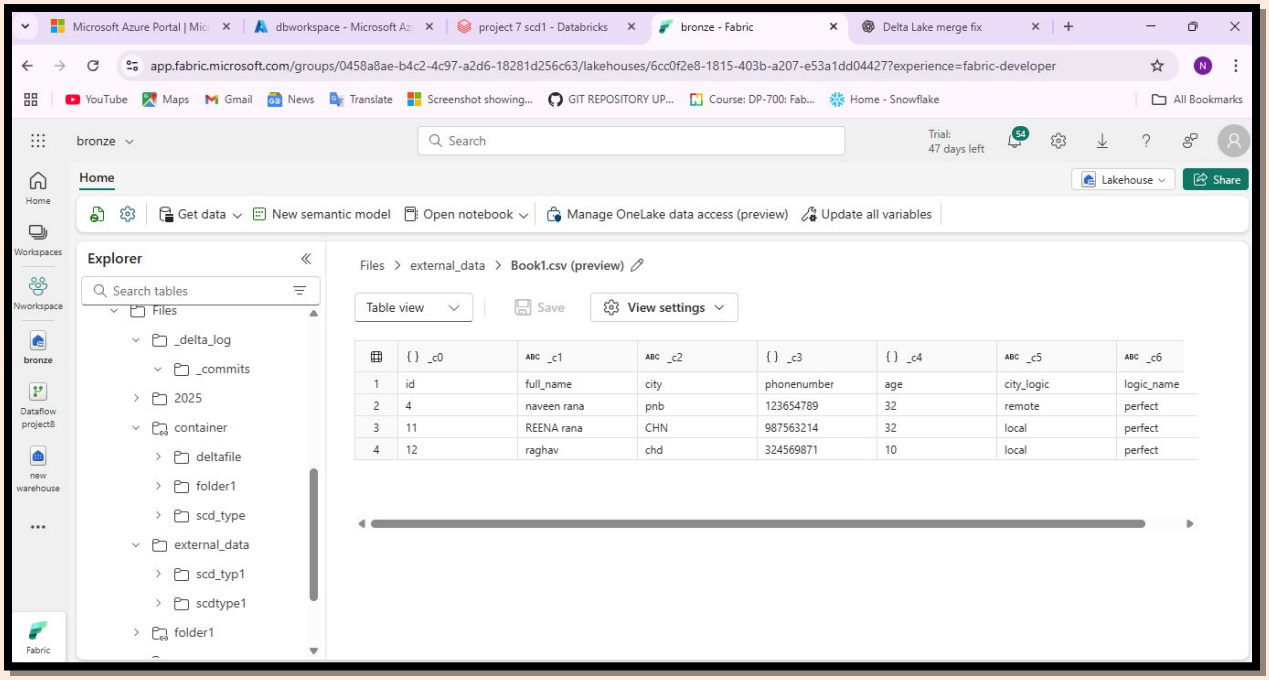


Target table gives data of new inserted values.
I used another file from adls storage account and update,insert values.
We are unable to create widgets in fabric as it didn't use dbutils and sparkutils command.

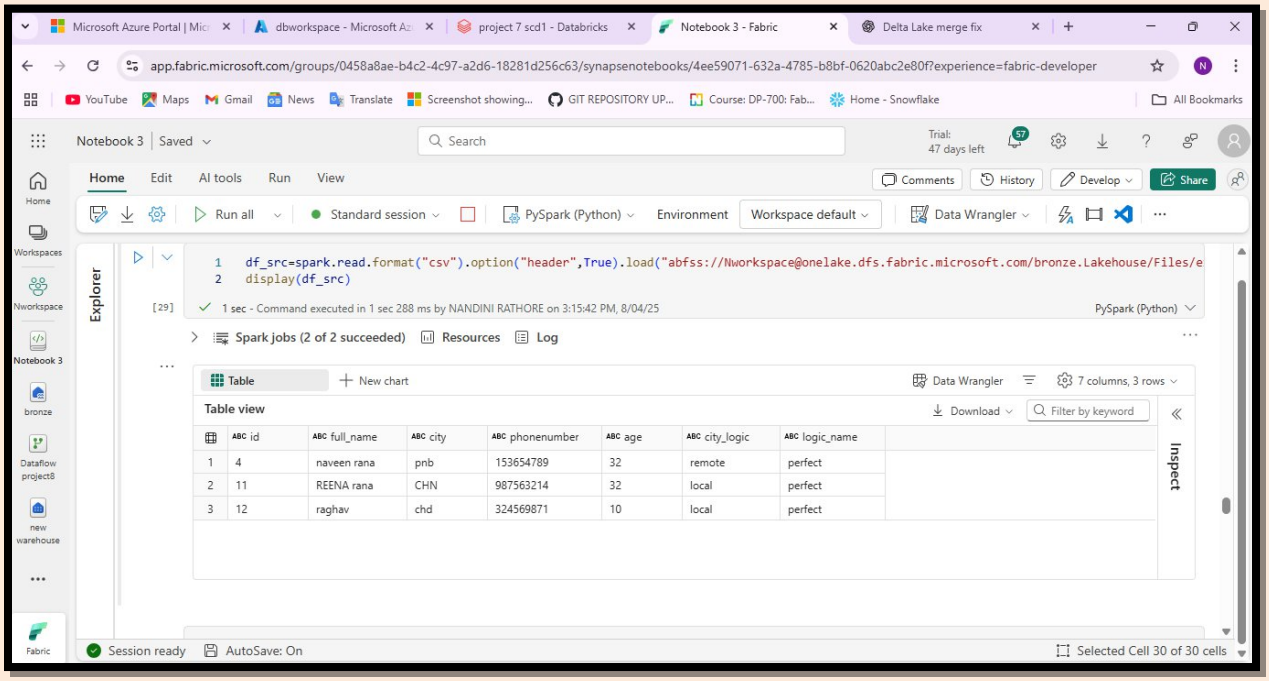


This is the parquet file format of scd_typ1 having all the newly inserted rows.





This is the other file I took as source table.



Read that file by using:

df_src=spark.read.format("csv").option("header",True).load("abfss://Nworkspace@onelake.dfs.fabric.microsoft.com/bronze.Lakehouse/Files/external_data/Book1.csv")

```
display(df_src)
```

```
df_hashkey=df_src.withColumn("src_hashkey", crc32(concat(*df_src.columns)))
display(df_hashkey)
```

[32] ✓ 1 sec - Command executed in 884 ms by NANDINI RATHORE on 3:16:36 PM, 8/04/25 PySpark (Python)

Spark jobs (1 of 1 succeeded) Resources

Table + New chart Data Wrangler 8 columns, 3 rows

Table view Download Filter by keyword

	id	full_name	city	phonenumber	age	city_logic	logic_name	src_hashkey
1	4	naveen rana	pnb	153654789	32	remote	perfect	3278921524
2	11	REENA rana	CHN	987563214	32	local	perfect	2639066754
3	12	raghav	chd	324569871	10	local	perfect	3344871274

Inspect

```
df_join=df_hashkey.alias("src").join(dbtable.toDF().alias("tgt"),(col("src.id")==col("tgt.id"))& (col("src.src_hashkey")==col("tgt.hashk
display(df_join)
```

✓ 1 sec - Command executed in 1 sec 585 ms by NANDINI RATHORE on 3:16:52 PM, 8/04/25 PySpark (Python)

Spark jobs (2 of 2 succeeded) Resources

Table + New chart Data Wrangler 8 columns, 3 rows

Table view Download Filter by keyword

	id	full_name	city	phonenumber	age	city_logic	logic_name	src_hashkey
1	4	naveen rana	pnb	153654789	32	remote	perfect	3278921524
2	11	REENA rana	CHN	987563214	32	local	perfect	2639066754
3	12	raghav	chd	324569871	10	local	perfect	3344871274

Inspect

Join with target table.

```
from pyspark.sql.functions import col, current_timestamp, lit

dbtable.alias("tgt").merge(
    df_join.alias("src"),
    "src.id == tgt.id"
).whenMatchedUpdate(
    set={
        "id": col("src.id"),
        "full_name": col("src.full_name"),
        "city": col("src.city"),
        "phonenumber": col("src.phonenumber"),
        "age": col("src.age"),
        "city_logic": col("src.city_logic"),
        "LOGIC_NAME": col("src.LOGIC_NAME"),
        "updateddate": current_timestamp(),
        "updatedby": lit("fabric-updated"),
        "hashkey": col("src.src_hashkey")
    }
```

```
).whenNotMatchedInsert(
    values={
        "id": col("src.id"),
        "full_name": col("src.full_name"),
        "city": col("src.city"),
         "phonenumber": col("src.phonenumber"),
        "age": col("src.age"),
        "city_logic": col("src.city_logic"),
        "LOGIC_NAME": col("src.LOGIC_NAME"),
        "updateddate": current_timestamp(),
        "updatedby": lit("fabric"),
        "createdby": lit("fabric"),
        "createddate": current_timestamp(),
        "hashkey": col("src.src_hashkey")
    }
).execute()
```

```
%%sql
select * from scd_typ1
```

3 sec - Command executed in 3 sec 660 ms by NANDINI RATHORE on 3:17:23 PM, 8/04/25 — Spark SQL

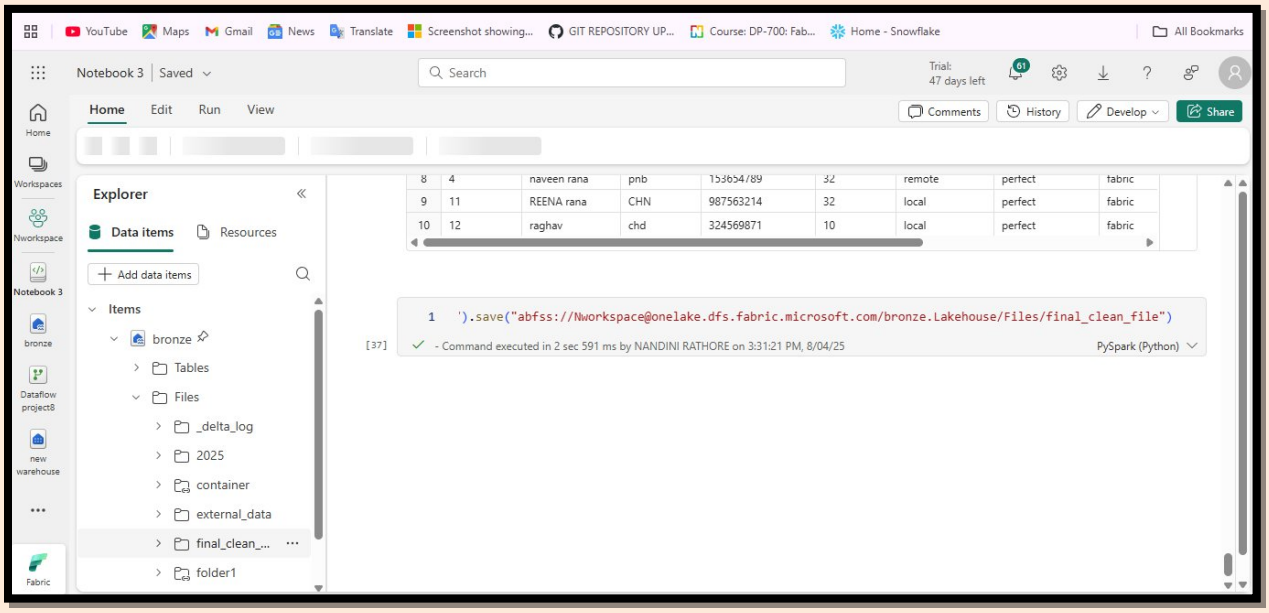
Spark jobs (7 of 7 succeeded) Resources

Table | New chart | 12 columns, 10 rows

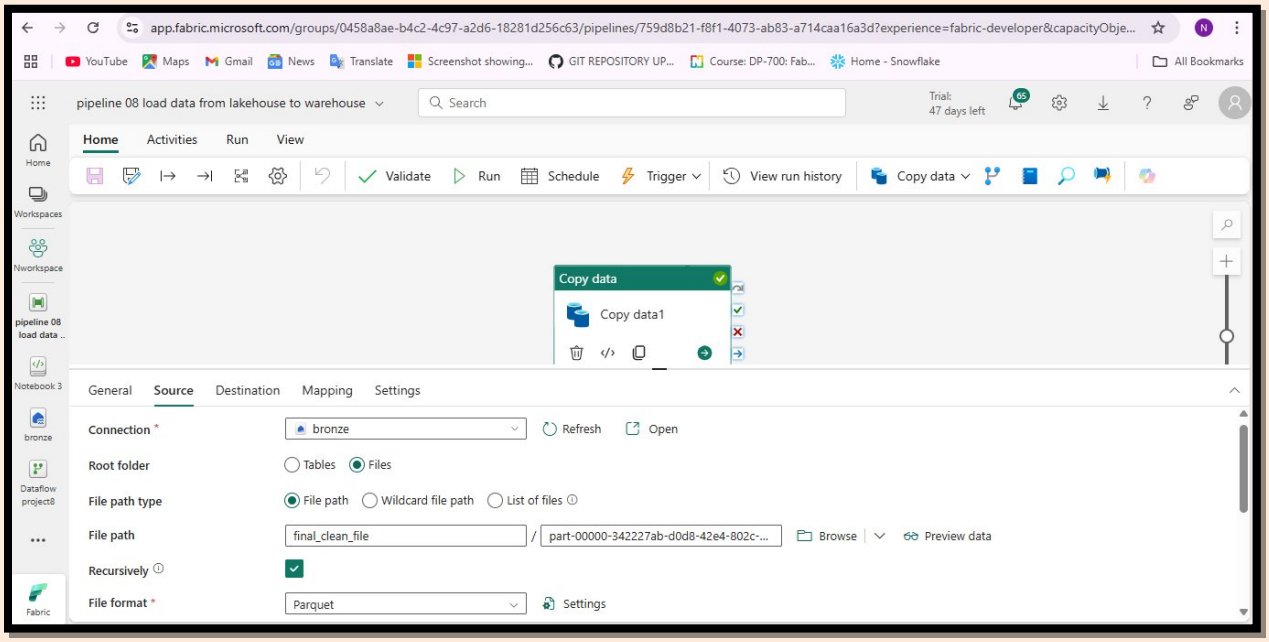
Table view — Download — Filter by keyword — Inspect

id	full_name	city	phonenumber	age	city_logic	logic_name	createdby	createddate	updatedby
1	nandini rana	ptl	814663852	26	local	perfect	fabric	2025-08-04T17:...	fabric
2	lakhwinder rana	ptl	74123694	32	local	perfect	fabric	2025-08-04T17:...	fabric
3	dimple rana	hry	888742012	37	remote	perfect	fabric	2025-08-04T17:...	fabric
4	naveen rana	pnb	153654789	32	remote	perfect	fabric	2025-08-04T17:...	fabric-updated
5	shikha rana	hry	9874567856	30	remote	perfect	fabric	2025-08-04T17:...	fabric
6	twinkle rana	pnb	789456123	17	remote	perfect	fabric	2025-08-04T17:...	fabric

All the rows are inserted except one one row which shows “updatedby=fabric-updated”.

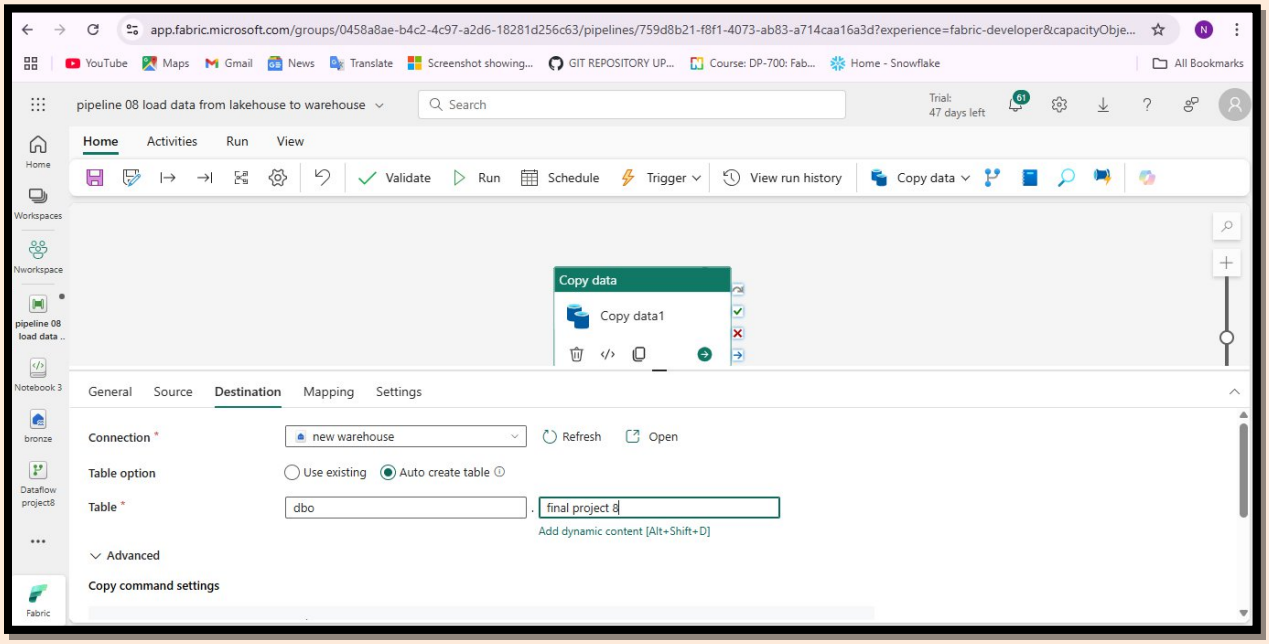


Write that file into parquet format in lakehouse.

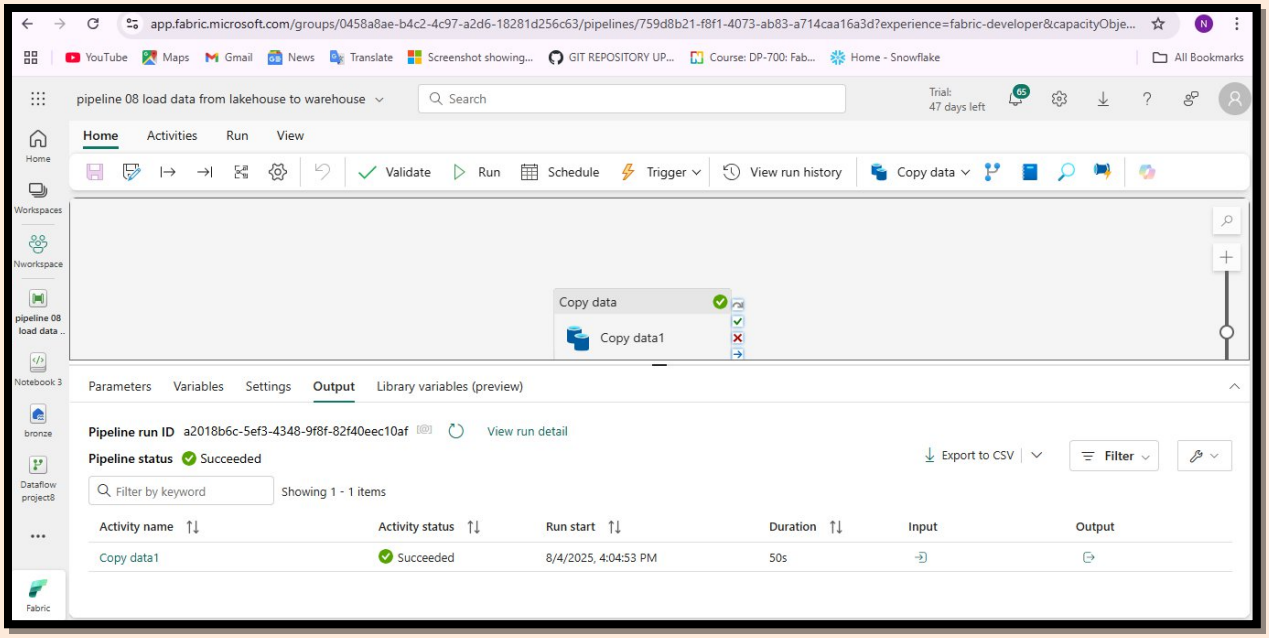


After that create a new pipeline.

Select source lakehouse ,choose file path,file formatàparquet.



Select sink warehouseàauto create tableàwrite schema and name of the table.



Save & run the pipeline.

Now the raw data after cleaning in notebook is successfully loaded into warehouse.

Conclusion:-- SCD Type 1 ensures that when changes are detected in dimension data ,the existing record is updated directly without maintaining historical versions.

The key operations involved in this step include:

•Reading Cleaned Data: The final cleaned and enriched dataset is loaded from the Lakehouse.

•Comparing with Target Data: Existing records in the Warehouse table are compared against the new data based on key identifiers.

•Merging Data: Records that have changes are updated, and new records are inserted. No historical versioning is maintained in SCD Type 1.

•Saving Updated Data: The final merged dataset is written back into the Warehouse, making it available for reporting and analytics.

This end-to-end process streamlines the journey of raw data to clean, enriched, and analytically ready data stored in the Warehouse.

- Dataflow Gen2: Performs initial data cleaning and transformation. Removing null values,remove duplicate values,conditional logics.
- Spark Notebook: Applies SCD Type 1 logic and loads the final output to the Warehouse.
- Lakehouse: Acts as the intermediate staging area for all transformations.
- Warehouse: Serves as the final destination for the transformed and up-to-date data.

This approach ensures:

- **Data Accuracy** – Clean, current data is always available for reporting.
- **Efficiency** – Separation of concerns across components improves maintainability.
- **Scalability** – The architecture supports growing data volumes and complex logic.
- **Business Relevance** – By adding calculated fields and applying SCD logic, we meet real-world needs like tracking customer updates or generating timely reports.

This project demonstrates how Microsoft Fabric can be leveraged to build end-to-end pipelines that convert raw data into meaningful insights through structured, governed, and automated processes.