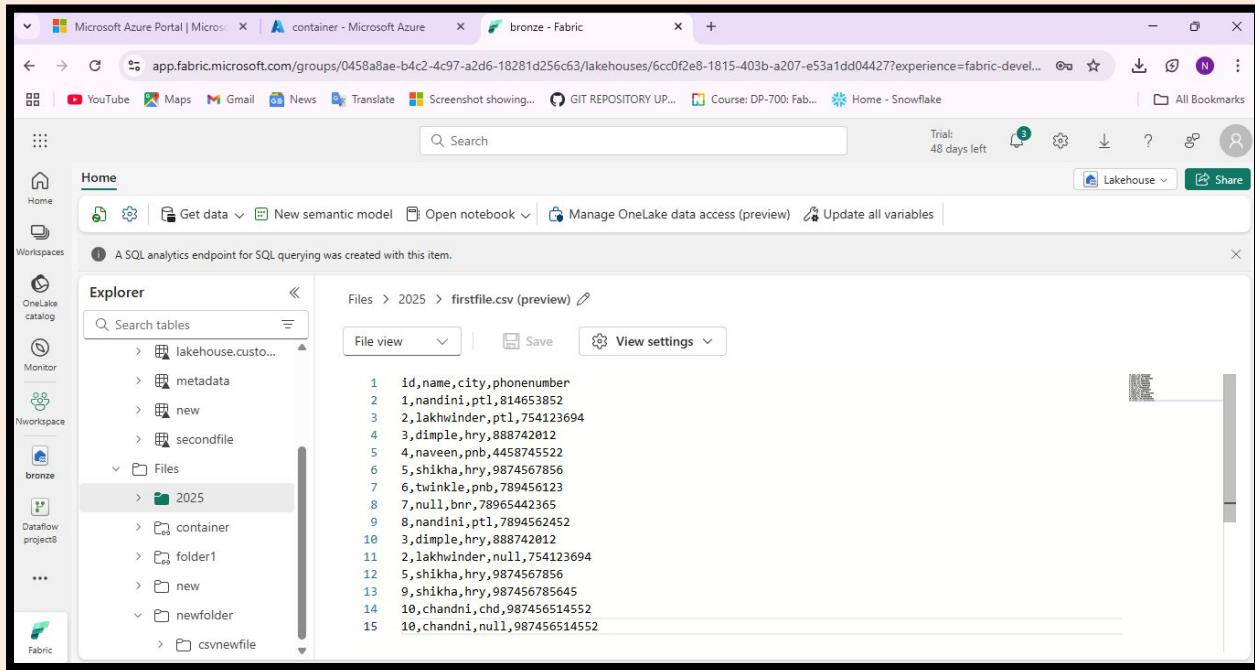


## Project - 8

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.

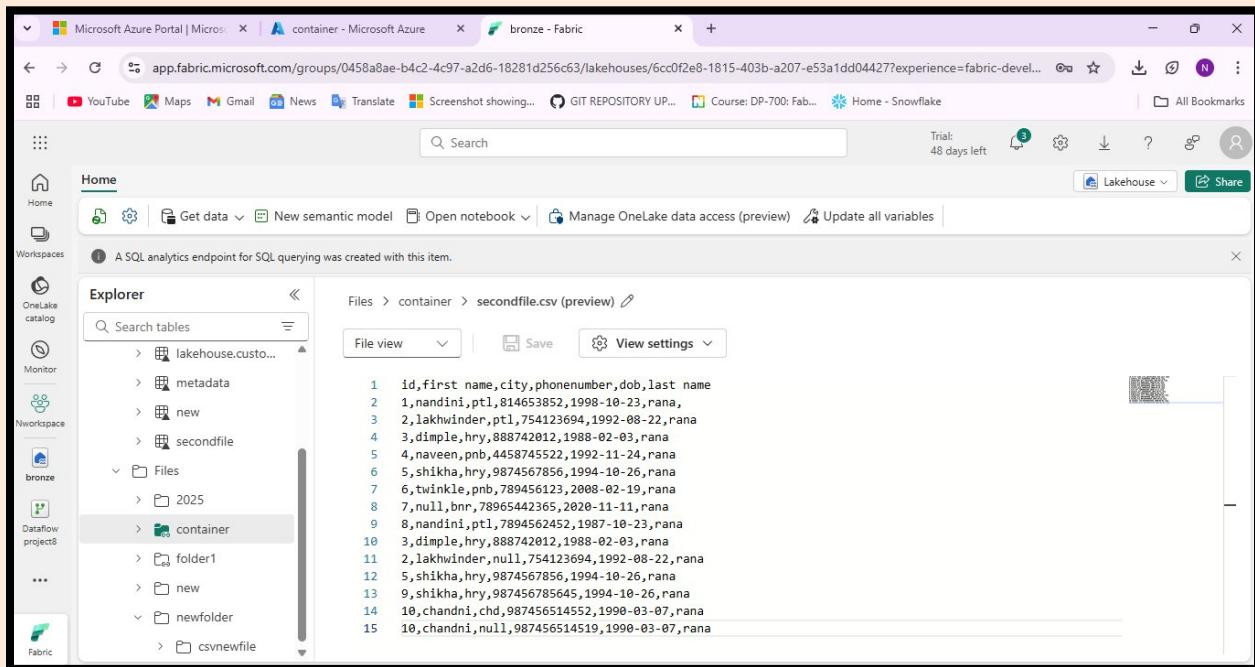
NANDINI RATHORE



The screenshot shows the Microsoft Fabric Data Explorer interface. The left sidebar lists workspaces: OneLake catalog, Monitor, Workspace (bronze), Dataflow project8, and Fabric. The main area displays the contents of a CSV file named 'firstfile.csv' located in the '2025' folder under the 'bronze' workspace. The file view shows 15 rows of data:

	id	name	city	phonenumber
1	1	nandini	ptl	814653852
2	2	lakhwinder	ptl	754123694
3	3	dimple	hry	888742012
4	4	naveen	pnb	445874552
5	5	shikha	hry	9874567856
6	6	twinkle	pnb	789456123
7	7	null	bnr	78965442365
8	8	nandini	ptl	7894562452
9	9	dimple	hry	888742012
10	10	lakhwinder	null	754123694
11	11	shikha	hry	9874567856
12	12	twinkle	pnb	789456123
13	13	shikha	hry	987456785645
14	14	chandni	chd	987456514552
15	15	chandni	null	987456514552

### CSV FILE IN BRONZE (LAKEHOUSE).



The screenshot shows the Microsoft Fabric Data Explorer interface. The left sidebar lists workspaces: OneLake catalog, Monitor, Workspace (bronze), Dataflow project8, and Fabric. The main area displays the contents of a CSV file named 'secondfile.csv' located in the 'container' folder under the 'bronze' workspace. The file view shows 15 rows of data:

	id	first name	city	phonenumber	dob	last name
1	1	nandini	ptl	814653852	1998-10-23	rana
2	2	lakhwinder	ptl	754123694	1992-08-22	rana
3	3	dimple	hry	888742012	1988-02-03	rana
4	4	naveen	pnb	445874552	1992-11-24	rana
5	5	shikha	hry	9874567856	1994-10-26	rana
6	6	twinkle	pnb	789456123	2008-02-19	rana
7	7	null	bnr	78965442365	2020-11-11	rana
8	8	nandini	ptl	7894562452	1987-10-23	rana
9	9	dimple	hry	888742012	1988-02-03	rana
10	10	lakhwinder	null	754123694	1992-08-22	rana
11	11	shikha	hry	9874567856	1994-10-26	rana
12	12	twinkle	pnb	789456123	2008-02-19	rana
13	13	shikha	hry	987456785645	1994-10-26	rana
14	14	chandni	chd	987456514552	1990-03-07	rana
15	15	chandni	null	987456514519	1990-03-07	rana

### CSV FILE IN BRONZE (LAKEHOUSE).

The screenshot shows the Microsoft Power Query interface. On the left, there's a sidebar with options like Home, Workspaces, OneLake catalog, Monitor, Nworkspace, bronze, Dataflow project8, and Fabric. The main area has a 'Power Query' tab and a 'Draft saved' message. A 'Get data' button is at the top, followed by a 'Choose data' dialog box. This dialog lists 'Display options' and shows two tables: 'firstfile' and 'secondfile'. The 'secondfile' table is selected and its preview grid is visible on the right, containing 10 rows of data with columns: id, firstname, city, phononenumber, dob, and lastname.

	id	firstname	city	phononenumber	dob	lastname
1	nandini	ptl		814663852	23/10/1998	rana
2	lakhwinder	ptl		74123694	22/08/1992	rana
3	dimple	hry		888742012	03/02/1988	rana
4	naveen	pnb		4458745522	24/11/1992	rana
5	shikha	hry		9874567856	26/10/1994	rana
6	twinkle	pnb		789456123	19/02/2008	rana
7	null	bnr		78965442365	11/11/2020	rana
8	nandini	ptl		7894562452	23/10/1987	rana
9	dimple	hry		888742012	03/02/1988	rana
10	lakhwinder	null		754123694	22/08/1992	rana
11	shikha	hry		9874567856	26/10/1994	rana
12	shikha	hry		987456785645	26/10/1994	rana
13	chandni	chd		987456514552	07/03/1990	rana
14	chandni	null		987456514519	07/03/1990	rana

OPEN WORKSPACEàNEW ITEMàDATAFLOWGEN2.

SELECT TWO TABLES FROM LAKEHOUSE.

This screenshot shows the Microsoft Power Query editor with two queries listed in the 'Queries [2]' pane: 'firstfile' and 'secondfile'. The 'firstfile' query is currently selected and its preview grid is shown, containing 3 rows with columns: Customer name, city, and Customer phononenumber. The 'Applied steps' pane on the right shows a history of steps, including 'Source', 'Navigation 1', 'Navigation 2', 'Navigation 3', and a step named 'Renamed c...'. The 'Properties' pane shows the query is named 'firstfile'.

Table.RenameColumns(#"Navigation 3", {"phonenumber", "Customer phonenumber"}, {"firstname", "Customer firstname"}, {"lastname", "Customer lastname"})

id	Customer firstname	city	Customer phonenumber	dob	Customer lastname
1	nandini	ptl	814663852	23/10/1998	rana
2	lakhwinder	ptl	74123694	22/08/1992	rana
3	dimple	hry	888742012	03/02/1988	rana
4	navneen	pnb	4458745522	24/11/1992	rana
5	shikha	hry	9874567856	26/10/1994	rana
6	twinkle	pnb	789456123	19/02/2008	rana
7	null	bnr	78965442365	11/11/2020	rana
8			9874567856	22/08/1992	rana
9			9874567856	22/08/1992	rana
10			9874567856	22/08/1992	rana
11			9874567856	22/08/1992	rana
12			9874567856	22/08/1992	rana
13			9874567856	22/08/1992	rana
14			9874567856	22/08/1992	rana

Replace one value with another in the selected columns.

Basic Advanced

Value to find: null  
Replace with: NA

OK Cancel

id	customer name	city
1	nandini	ptl
2	lakhwinder	ptl
3	dimple	hry
4	navneen	pnb
5	shikha	hry
6	twinkle	pnb
7	null	bnr
8	nandini	ptl
9	dimple	hry
10	lakhwinder	null
11	shikha	hry
12	shikha	hry
13	chandni	chd
14	shikha	null

NOW CLEAN THE DATA.

REMOVE NULL VALUES.

The screenshot shows the Microsoft Power Query Editor interface. On the left, there's a sidebar with icons for Home, Workspaces, OneLake catalog, Monitor, Nworkspace, bronze, Dataflow project8, and Fabric. The main area shows two queries: 'secondfile' and 'firstfiles'. The 'firstfiles' query has 14 rows and 4 columns. A context menu is open over the 'customer name' column, with 'Remove duplicates' highlighted. The 'Applied steps' pane on the right shows a step named 'Renamed c...'. The status bar at the bottom indicates 'Columns: 4 Rows: 14'.

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

This screenshot shows the Power Query Editor after the 'Remove duplicates' step has been applied. The 'firstfiles' query now has 8 unique rows. The 'Applied steps' pane now includes 'Removed duplicates' and 'Replaced value'. The status bar at the bottom indicates 'Completed (1.31 s)'. The data table shows columns: '1~3 id', 'customer name', 'city', and '1~3 customer phonenumbers'.

1~3 id	customer name	city	1~3 customer phonenumbers
1	nandini	pti	814653852
2	lakhwinder	pti	754123694
3	dimple	hry	88874012
4	naveen	pnb	445874552
5	shikha	hry	9874567856
6	twinkle	pnb	789456123
7	null	bnr	78965442365
10	chandni	chd	987456514552

The screenshot shows the Microsoft Power Query interface within the Azure Dataflow project. A 'Merge' dialog is open, prompting the user to select tables and matching columns for merging. The 'Left table for merge' is set to 'firstfiles' and the 'Right table for merge' is set to 'secondfile'. Both tables have one row selected. A green status bar at the bottom indicates 'The selection matches 8 of 8 rows from the first table'. The 'OK' button is visible at the bottom right of the dialog.

**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.

The screenshot shows the Microsoft Power Query interface after the merge operation. The 'Queries' pane now contains three steps: 'secondfile', 'firstfiles', and a new 'Merge' step. The 'Merge' step has a 'Source' connection to 'firstfiles' and an 'Expanded secondfile' connection. Below the 'Merge' step, a 'Table.RemoveColumns' step is shown with the formula '#"Expanded secondfile", {"full name"}'. The resulting table in the preview pane contains all columns except the 'full name' column, which is highlighted in red. The status bar at the bottom shows 'Columns: 6 Rows: 10'.

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

**Replace values**

Replace one value with another in the selected columns.

Basic  Advanced

Value to find  
null

Replace with  
N/A

OK Cancel

**Custom column**

Add a column that is computed from other columns or values.

New column name \*

Data type

Custom column formula \*

```
= let
    today=DateTime.Date(DateTime.LocalNow()),
    birth=DateTime.Date([dob]),
    cust_age=Number.RoundDown(Duration.Days (today-birth)/365.25)
in
    cust_age
```

Available column(s)

- id
- Customer firstname
- city
- Customer phononenumber
- dob
- Customer lastname
- FULL NAME

Insert column

OK Cancel

age calculated from date of birth.

The screenshot shows the Microsoft Power Query Editor interface. A query named "secondfile" is open, which contains a single step: "Table.AddColumn(#"Changed column type", "cust\_age", each let today=DateTime.Date(DateTime.LocalNow()), birth=DateTime.Date([dob]), cust\_age=Number.RoundDown(Duration.Days(today-birth))/365.25)". The resulting table has columns: id, Customer firstname, city, Customer phonenumer, dob, Customer lastname, FULL NAME, and cust\_age. The data includes rows for various customers with their names, cities, phone numbers, dates of birth, and calculated ages.

	Customer firstname	city	Customer phonenumer	dob	Customer lastname	FULL NAME	cust_age
1	nandini	ptl	814663852	23/10/1998	rana	nandini rana	26
2	lakhwinder	ptl	74123694	22/08/1992	rana	lakhwinder rana	32
3	dimple	hry	888742012	03/02/1988	rana	dimple rana	37
4	naveen	pnb	445874552	24/11/1992	rana	naveen rana	32
5	shikha	hry	9874567856	26/10/1994	rana	shikha rana	30
6	twinkle	pnb	789456123	19/02/2008	rana	twinkle rana	17
7	null	bnr	78965442365	11/11/2020	rana	null rana	4
8	nandini	ptl	7891562457	22/10/1987	rana	nandini rana	32

The screenshot shows the Microsoft Power Query Editor interface. A query named "secondfile" is open, which contains a step: "Table.RemoveColumns(#"Reordered columns", {"Customer firstname", "Customer lastname"}). The resulting table has columns: id, FULL NAME, city, Customer phonenumer, dob, and cust\_age. The data includes rows for various customers with their names, cities, phone numbers, dates of birth, and calculated ages.

	FULL NAME	city	Customer phonenumer	dob	cust_age
1	nandini rana	ptl	814663852	23/10/1998	26
2	lakhwinder rana	ptl	74123694	22/08/1992	32
3	dimple rana	hry	888742012	03/02/1988	37

Full name columns derived from first and last name.

Power Query Draft saved

Home Transform Add column View Help

Queries [2]

Navigation 3 Replaced value 1 Renamed columns Removed duplicates Replaced value Added custom Changed column... Added custom 2 Rearranged columns Removed columns Renamed columns

Table.ReplaceValue(#"Removed columns", "null rana", "N/A", Replacer.ReplaceValue, {"id", "FULL NAME", "city", "Customer phonenumber", "dob", "cust\_age"})

	id	FULL NAME	city	Customer phonenumber	dob	cust_age
7	7	N/A	bnr	78965442365	11/11/2020	4
8	8	nandini rana	ptl	7894562452	23/10/1987	37

Table row details

	id	FULL NAME	city
	7	N/A	bnr

Completed (1.11 s) Columns: 6 Rows: 10 Add default destination...

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

Power Query Draft saved

Home Transform Add column View Help

Queries [2]

Navigation 3 Replaced value 1 Renamed columns Removed duplicates Replaced value Added custom Changed column... Added custom 2 Rearranged columns Removed columns Renamed columns

Custom column

New column name \* flag Data type Whole number

Custom column formula \* = if [Customer phonenumber] = null then "MISSING" else "VALID"

Available column(s)

- id
- FULL NAME
- city
- Customer phonenumber
- dob
- cust\_age

Learn more about Power Query formulas

OK Cancel

Completed (1.11 s) Columns: 6 Rows: 10 Add default destination...

Flags or status indicators based on conditional logic.

## ADD COLUMNSàCUSTOM COLUMNSàWRITE THE FORMULA TO GET CONDITIONAL LOGIC.

**Custom column**

Add a column that is computed from other columns or values.

New column name\*: place logic

Data type:

Custom column formula\*:

```
= if [city] = "ptl" then "local"
else "remote"
```

Available column(s):

- id
- FULL NAME
- city
- Customer phonenumer
- dob
- cust\_age
- flag

Learn more about Power Query formulas

OK Cancel

ABC 123	ABC id	ABC FULL NAME	ABC city	ABC 123 Customer phonenumer	ABC 123 dob	ABC 123 cust_age	ABC 123 flag	ABC 123 place logic
1	1	nandini rana	ptl			26	VALID	local
2	2	lakhwinder rana	ptl	814663852	23/10/1998	32	VALID	local
3	3	dimple rana	hry	74123694	22/08/1992	37	VALID	remote
4	4	naveen rana	pnb	888742012	03/02/1988	32	VALID	remote
5	5	shikha rana	hry	445874522	24/11/1992	30	VALID	remote
6	6	twinkle rana	pnb	9874567856	26/10/1994	17	VALID	remote
7	7	N/A	bnr	789456123	19/02/2008	4	VALID	remote

ABC 123	ABC id	ABC FULL NAME	ABC city	ABC 123 Customer phonenumer	ABC 123 dob	ABC 123 cust_age	ABC 123 flag	ABC 123 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	445874522	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

**Custom column**

New column name: name logic

Custom column formula:

```
= if [FULL NAME] = "N/A" then "NO NAME"
else "perfect"
```

Available column(s): id, FULL NAME, city, Customer phonenumer, dob, cust\_age, flag

ABC 123	id	ABC FULL NAME	ABC city	ABC 123 Customer phonenumer	ABC 123 dob	ABC 123 cust_age	ABC flag	ABC 123 place logic	ABC 123 name logic
1	1	nandini rana	ptl						
2	2	lakhwinder rana	ptl						
3	3	dimple rana	hry						
4	4	naveen rana	pnb	4458745522	24/11/1992	32	VALID	remote	perfect
5	5	shikha rana	hry	9874567856	26/10/1994	30	VALID	remote	perfect
6	6	twinkle rana	pnb	789456123	19/02/2008	17	VALID	remote	perfect
7	7	N/A	bnn	78965442365	11/11/2020	4	VALID	remote	NO NAME
8	8	nandini rana	ptl	7894562452	23/10/1987	37	VALID	local	perfect

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

ABC 123	id	ABC FULL NAME	ABC city	ABC 123 Customer phonenumer	ABC 123 dob	ABC 123 cust_age	ABC flag	ABC 123 place logic	ABC 123 name logic
3	3	dimple rana	hry	888742012	03/02/1988	37	VALID	remote	perfect
4	4	naveen rana	pnb	4458745522	24/11/1992	32	VALID	remote	perfect
5	5	shikha rana	hry	9874567856	26/10/1994	30	VALID	remote	perfect
6	6	twinkle rana	pnb	789456123	19/02/2008	17	VALID	remote	perfect
7	7	N/A	bnn	78965442365	11/11/2020	4	VALID	remote	NO NAME
8	8	nandini rana	ptl	7894562452	23/10/1987	37	VALID	local	perfect

The screenshot shows the Microsoft Fabric Dataflow interface. On the left, there's a sidebar with 'Queries [2]' selected. The main area shows a 'Get data' step with a 'Choose data source' modal open. The modal lists two data sources: 'bronze' (Lakehouse) and 'nandinisqlserver.database.windows.net:sqldb NANDINI' (SQL Server database). The 'bronze' entry is highlighted. At the bottom of the modal, it says 'Connecting to bronze'.

NOW LOAD THIS CLEAN DATA TO FINAL DESTINATION àLAKEHOUSE.

The screenshot shows the Microsoft Fabric Dataflow interface. On the left, there's a sidebar with 'Queries [3]' selected. The main area shows a 'Data destination' step with a 'Choose destination target' modal open. The modal shows 'New table' selected and displays a tree view of the 'bronze' folder. A new table named 'Merge 08' is being created under the 'bronze' folder. The 'Next' button is visible at the bottom right of the modal.

SAVE TABLE.

A SQL analytics endpoint for SQL querying was created with this item.

Merge project table

Showing 8 rows

	id	full name	city	phononenumber	age	city logic	logic name
1	1	nandini rana	ptl	814663852	26	local	perfect
2	2	lakhwinder rana	ptl	74123694	32	local	perfect
3	3	dimple rana	hry	888742012	37	remote	perfect
4	4	naveen rana	pnb	4458745522	32	remote	perfect
5	5	shikha rana	hry	9874567856	30	remote	perfect
6	6	twinkle rana	pnb	788456123	17	remote	perfect
7	7	N/A	bnr	78965442365	4	remote	no name
8	10	chandni rana	chd	98745614552	35	remote	perfect

Succeeded (5 sec 37 ms)

Columns 7 Rows 8

A SQL analytics endpoint for SQL querying was created with this item.

Merge project table (File view) > 0efe6d17fbfc494bb8a631f27e32c72.parquet

	c11db66c-6754-4...	44053aab-7a7b-4...	Odca4403-166f-4...	234113bf-8542-4...	91f37125-d9b2-4...	4009ea9b-2e34-4...	5c02ff4e-1
1	1	nandini rana	ptl	814663852	26	local	perfect
2	2	lakhwinder rana	ptl	74123694	32	local	perfect
3	3	dimple rana	hry	888742012	37	remote	perfect
4	4	naveen rana	pnb	4458745522	32	remote	perfect
5	5	shikha rana	hry	9874567856	30	remote	perfect

WHEN CLICK ON FILE VIEW IT SHOWS PARQUET FILE FORMAT.

The screenshot shows a Microsoft Fabric Notebook interface. In the code editor, two commands are run:

```

1   from pyspark.sql.functions import*
[76] ✓ 1 sec - Command executed in 392 ms by NANDINI RATHORE on 1:49:16 PM, 8/04/25
...
+---+
| df=spark.read.format("parquet").option("header",True).load("abfss://Nworkspace@onelake.dfs.fabric.microsoft.com/bronze.Lakehouse/Tables/bronze.parquet") |
| display(df) |
| 3 |
[77] ✓ 1 sec - Command executed in 1 sec 605 ms by NANDINI RATHORE on 1:49:18 PM, 8/04/25

```

The resulting table view displays the following data:

	c11db66c-6754-4f51-b83e-05a...	ABC 4405aab-7a7b-462f-bc9e-00...	ABC 0dca4403-166f-46ff-93ef-76bf8...	12L 234113bf-8542-4d51-8be3-799...	12L 91f37125-d9b2-41ca-af32-b9...
1	1	nandini rana	ptl	814663852	26
2	2	lakhwinder rana	ptl	74123694	32
3	3	dimple rana	hry	888742012	37
4	4	naveen rana	pnb	445874522	32
5	5	shikha rana	hry	9874567856	30
6	6	twinkle rana	pnb	789456123	17
7	7	N/A	bnr	78965442365	4

NOW CLICK ON NEW ITEMàNOTEBOOK.

PERFORM Scdtype1.

IMPORT FUNCTIONS.

The screenshot shows a Microsoft Fabric Notebook interface. In the code editor, the same PySpark code is run again, resulting in the same table view. The data has been updated with new rows:

	c11db66c-6754-4f51-b83e-05a...	ABC 4405aab-7a7b-462f-bc9e-00...	ABC 0dca4403-166f-46ff-93ef-76bf8...	12L 234113bf-8542-4d51-8be3-799...	12L 91f37125-d9b2-41ca-af32-b9...
1	1	nandini rana	ptl	814663852	26
2	2	lakhwinder rana	ptl	74123694	32
3	3	dimple rana	hry	888742012	37
4	4	naveen rana	pnb	445874522	32
5	5	shikha rana	hry	9874567856	30
6	6	twinkle rana	pnb	789456123	17
7	7	N/A	bnr	78965442365	4
8	8	anup rana	ptl	74123694	32
9	9	lakshmi rana	ptl	74123694	32
10	10	prachi rana	ptl	74123694	32
11	11	komal rana	ptl	74123694	32
12	12	komal rana	ptl	74123694	32
13	13	komal rana	ptl	74123694	32
14	14	komal rana	ptl	74123694	32
15	15	komal rana	ptl	74123694	32
16	16	komal rana	ptl	74123694	32
17	17	komal rana	ptl	74123694	32
18	18	komal rana	ptl	74123694	32
19	19	komal rana	ptl	74123694	32
20	20	komal rana	ptl	74123694	32
21	21	komal rana	ptl	74123694	32
22	22	komal rana	ptl	74123694	32

READ SOURCE FILE.

```

1 df.write.format("delta").mode("overwrite").save("Files/external_data/scdtype1")
[79] ✓ 1 sec - Command executed in 1 sec 731 ms by NANDINI RATHORE on 1:49:20 PM, 8/04/25
PySpark (Python) ▾

> Spark jobs (3 of 3 succeeded) Resources Log

1 from pyspark.sql.functions import*
[80] ✓ <1 sec - Command executed in 384 ms by NANDINI RATHORE on 1:49:21 PM, 8/04/25
PySpark (Python) ▾

1 %%sql
2 create table if not EXISTS scd_typ1
3 (id int,full_name varchar(100),city varchar(100),phonenumber bigint,age int ,city_logic varchar (100),logic_name varchar(100),createdby
4 using DELTA
5 LOCATION "Files/external_data/scd_typ1"

```

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"

```

```
%sql
desc formatted scd_typ1
```

1 sec - Command executed in 1 sec 612 ms by NANDINI RATHORE on 1:54:24 PM, 8/04/25

**Table view**

ABC col_name	ABC data_type	ABC comment
4 phonenumber	bigint	NULL
5 age	int	NULL
6 city_logic	string	NULL
7 logic_name	string	NULL
8 createdby	string	NULL
9 createddate	timestamp	NULL
10 updatedby	string	NULL
11 updateddate	timestamp	NULL
12 hashkey	bigint	NULL

WE USE THIS COMMAND TO KNOW TYPE OF THE TABLE.

```
# Detailed Table...
```

6 age	int	NULL
7 city_logic	string	NULL
8 logic_name	string	NULL
9 createdby	string	NULL
10 createddate	timestamp	NULL
11 updatedby	string	NULL
12 updateddate	timestamp	NULL
13 hashkey	bigint	NULL
14 # Detailed Table...		
15 Name	spark_catalog...	
16 Type	EXTERNAL	
17 Location	abfss://0458a8...	
18 Provider	delta	
19 Owner	trusted-service...	
20 Table Properties	[delta.minRea...	

DELTA TABLE IS CREATED.

```

from delta import DeltaTable
dbtable=DeltaTable.forName(spark,"abfss://Nworkspace@onelake.dfs.fabric.microsoft.com/bronze.Lakehouse/Files/external_data/scd_typ1")
dbtable.toDF().show()

```

[96] ✓ 2 sec - Command executed in 2 sec 685 ms by NANDINI RATHORE on 1:55:58 PM, 8/04/25

> Spark jobs (5 of 5 succeeded) Resources Log

	id	full_name	city	phononenumber	age	city_logic	logic_name	src_hashkey
1	1	nandini rana	ptl	814663852	26	local	perfect	3243872751
2	2	lakhwinder rana	ptl	74123694	32	local	perfect	2382918228
3	3	dimple rana	hry	888742012	37	remote	perfect	840546996
4	4	naiveen rana	pnb	445874552	32	remote	perfect	3076101506
5	5	shikha rana	hry	9874567856	30	remote	perfect	2194779116
6	6	twinkle rana	pnb	789456123	17	remote	perfect	386286327
7	7	N/A	bnr	78965442365	4	remote	no name	1640628209

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

```

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

```

[97] ✓ <1 sec - Command executed in 1 sec 80 ms by NANDINI RATHORE on 1:56:33 PM, 8/04/25

> Spark jobs (2 of 2 succeeded) Resources

Table view

	id	full_name	city	phononenumber	age	city_logic	logic_name	src_hashkey
1	1	nandini rana	ptl	814663852	26	local	perfect	3243872751
2	2	lakhwinder rana	ptl	74123694	32	local	perfect	2382918228
3	3	dimple rana	hry	888742012	37	remote	perfect	840546996
4	4	naiveen rana	pnb	445874552	32	remote	perfect	3076101506
5	5	shikha rana	hry	9874567856	30	remote	perfect	2194779116
6	6	twinkle rana	pnb	789456123	17	remote	perfect	386286327
7	7	N/A	bnr	78965442365	4	remote	no name	1640628209

### GENERATING HASHKEY:

```

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

```

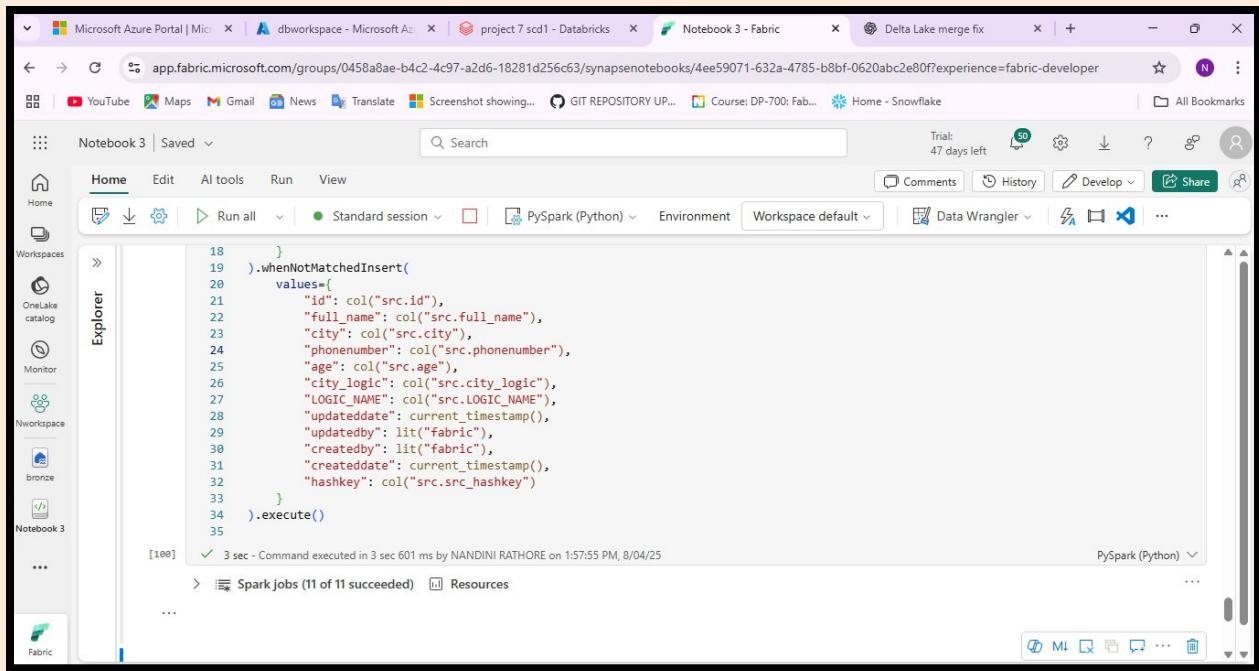
	id	full_name	city	phononenumber	age	city_logic	logic_name	src_hashkey
1	1	nandini rana	ptl	814663852	26	local	perfect	3243872751
2	2	lakhwinder rana	ptl	74123694	32	local	perfect	2382918228
3	3	dimple rana	hry	888742012	37	remote	perfect	840546996
4	4	naaveen rana	pnb	4458745522	32	remote	perfect	3076101506
5	5	chikha rana	hry	0874567856	30	remote	perfect	2104770116

- `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.

```

1   from pyspark.sql.functions import col, current_timestamp, lit
2
3   dbtable.alias("tgt").merge(
4       df_hashkey.alias("src"),
5       "src.id == tgt.id"
6   ).whenMatchedUpdate(
7       set={
8           "id": col("src.id"),
9           "full_name": col("src.full_name"),
10          "city": col("src.city"),
11          "phononenumber": col("src.phonenumber"),
12          "age": col("src.age"),
13          "city_logic": col("src.city_logic"),
14          "LOGIC_NAME": col("src.LOGIC_NAME"),
15          "updateddate": current_timestamp(),
16          "updatedby": lit("fabric-updated"),
17          "hashkey": col("src.src_hashkey")
18      }
19  ).whenNotMatchedInsert(
20      values={
21          "id": col("src.id"),
22          "full_name": col("src.full_name"),
23          "city": col("src.city"),
24          "phononenumber": col("src.phonenumber"),
25          "age": col("src.age")
26      }
27 )

```



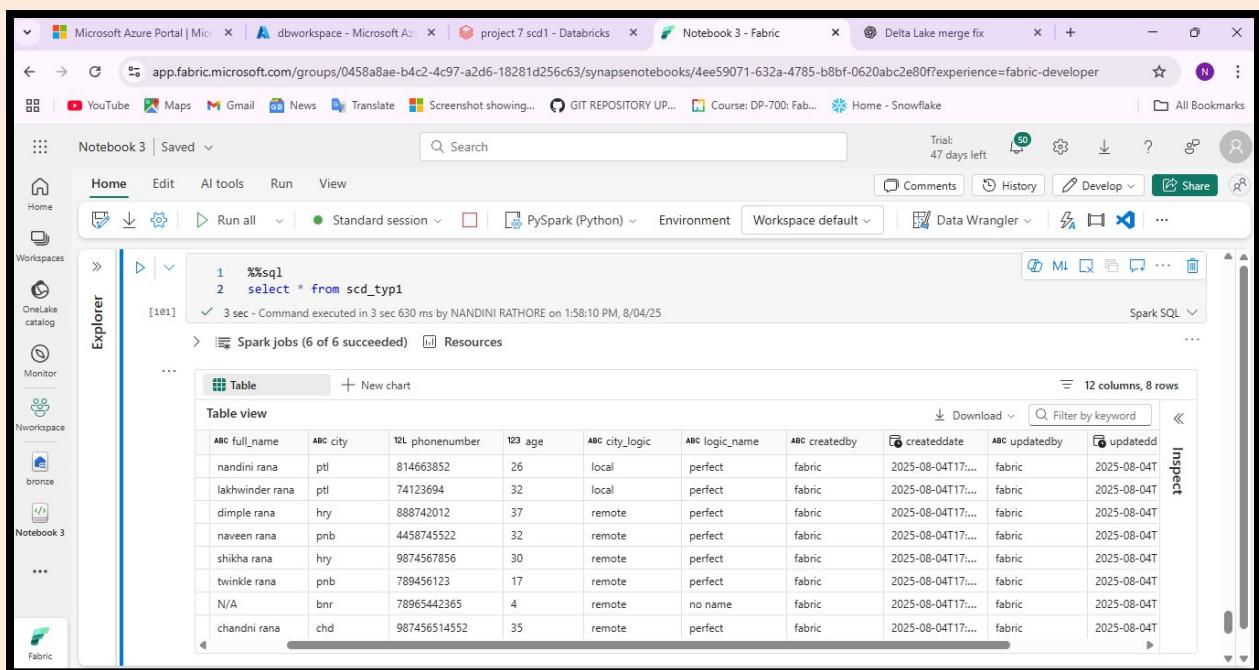
The screenshot shows a Databricks notebook titled "Notebook 3 | Saved". The code in the editor is:

```

18
19     ).whenNotMatchedInsert(
20         values=(
21             "id": col("src.id"),
22             "full_name": col("src.full_name"),
23             "city": col("src.city"),
24             "phonenumber": col("src.phonenumber"),
25             "age": col("src.age"),
26             "city_logic": col("src.city_logic"),
27             "LOGIC_NAME": col("src.LOGIC_NAME"),
28             "updatedate": current_timestamp(),
29             "updatedby": lit("fabric"),
30             "createdby": lit("fabric"),
31             "createddate": current_timestamp(),
32             "hashkey": col("src.src_hashkey")
33         )
34     ).execute()
35

```

The output cell [100] shows the command was executed in 3 sec 601 ms by NANDINI RATHORE on 1:57:55 PM, 8/04/25. Below the notebook interface, there is a "Spark jobs (11 of 11 succeeded)" section.



The screenshot shows a Databricks notebook titled "Notebook 3 | Saved". The code in the editor is:

```

1 %%sql
2 select * from scd_typ1

```

The output cell [101] shows the command was executed in 3 sec 630 ms by NANDINI RATHORE on 1:58:10 PM, 8/04/25. Below the notebook interface, there is a "Spark jobs (6 of 6 succeeded)" section. A "Table view" section displays the results of the query:

ABC full_name	ABC city	121 phonenumber	123 age	ABC city_logic	ABC logic_name	ABC createdby	ABC createddate	ABC updatedby	ABC updatedd
nandini rana	ptl	814663852	26	local	perfect	fabric	2025-08-04T17...	fabric	2025-08-04T
lakhwinder rana	ptl	74123694	32	local	perfect	fabric	2025-08-04T17...	fabric	2025-08-04T
dimple rana	hry	888742012	37	remote	perfect	fabric	2025-08-04T17...	fabric	2025-08-04T
naveen rana	pnb	4458745522	32	remote	perfect	fabric	2025-08-04T17...	fabric	2025-08-04T
shikha rana	hry	9874567856	30	remote	perfect	fabric	2025-08-04T17...	fabric	2025-08-04T
twinkle rana	pnb	789456123	17	remote	perfect	fabric	2025-08-04T17...	fabric	2025-08-04T
N/A	bnr	78965442365	4	remote	no name	fabric	2025-08-04T17...	fabric	2025-08-04T
chandni rana	chd	987456514552	35	remote	perfect	fabric	2025-08-04T17...	fabric	2025-08-04T

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.

The screenshot shows the Microsoft Fabric Data Explorer interface. On the left, the 'Explorer' sidebar lists various datasets and files under the 'bronze' workspace. A table preview is displayed for the file 'part-00000-edf26978-5c49-4081-bb37-a091db8cb5ad-c000.snappy.parquet (preview)'. The table has columns: id, full\_name, city, phonenumb, age, city\_logic, logic\_name, and created. The data includes rows for Nandini Ratha, Lakhwinder Rana, Dimple Rana, Neaveen Rana, Shikha Rana, Twinkle Rana, and an N/A entry.

	id	full_name	city	phonenumb	age	city_logic	logic_name	created
1	1	nandini rana	ptl	814663852	26	local	perfect	fabric
2	2	lakhwinder rana	ptl	74123694	32	local	perfect	fabric
3	3	dimple rana	hry	888742012	37	remote	perfect	fabric
4	4	neaveen rana	pnb	445874552	32	remote	perfect	fabric
5	5	shikha rana	hry	9874567856	30	remote	perfect	fabric
6	6	twinkle rana	pnb	789456123	17	remote	perfect	fabric
7	7	N/A	bnr	78965442365	4	remote	no name	fabric

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

The screenshot shows a PySpark notebook titled 'Notebook 3'. The code cell contains the following command:

```
1 df.write.format("delta").mode("overwrite").save("Files/external_data/scd_type1")
```

The output of the cell shows the command was executed successfully in 2 seconds by NANDINI RATHORE on 2:00:12 PM, 8/04/25. Below the code cell, there is a section for 'Spark jobs' which shows 3 of 3 succeeded.

The screenshot shows the Microsoft Fabric Data Explorer interface. On the left, the 'Explorer' sidebar lists workspaces, including 'bronze' (selected), 'new\_warehouse', and 'Fabric'. Under 'bronze', there are several containers and files, such as '\_delta\_log', '2025', 'container', 'external\_data', and 'Book1.csv'. The main area displays the contents of 'Book1.csv' in 'Table view'. The table has seven columns: '\_c0', 'ABC \_c1', 'ABC \_c2', '\_c3', '\_c4', 'ABC \_c5', and 'ABC \_c6'. The data rows are:

_c0	ABC _c1	ABC _c2	_c3	_c4	ABC _c5	ABC _c6
1	full_name	city	phonenumber	age	city_logic	logic_name
2	naveen rana	pnb	123654789	32	remote	perfect
3	REENA rana	CHN	987563214	32	local	perfect
4	raghav	chd	324569871	10	local	perfect

This is the other file I took as source table.

The screenshot shows a Databricks Notebook titled 'Notebook 3 - Fabric'. The notebook interface includes a top navigation bar with tabs like Home, Edit, AI tools, Run, View, and a search bar. Below the navigation is a toolbar with various icons. The main workspace contains a code cell with the following PySpark code:

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

Below the code cell, a message indicates the command was executed successfully: "1 sec - Command executed in 1 sec 288 ms by NANDINI RATHORE on 3:15:42 PM, 8/04/25". The workspace also shows a section for 'Spark jobs' and 'Resources'. To the right, there is a preview of the data in 'Table view', which matches the structure shown in the previous screenshot. The data rows are:

ABC id	ABC full_name	ABC city	ABC phonenumber	ABC age	ABC city_logic	ABC logic_name
4	naveen rana	pnb	123654789	32	remote	perfect
11	REENA rana	CHN	987563214	32	local	perfect
12	raghav	chd	324569871	10	local	perfect

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)
```

```

1 df_hashkey=df_src.withColumn("src_hashkey", crc32(concat(*df_src.columns)))
2 display(df_hashkey)
✓ 1 sec - Command executed in 884 ms by NANDINI RATHORE on 3:16:36 PM, 8/04/25
> Spark jobs (1 of 1 succeeded) Resources
Table view Data Wrangler Download Filter by keyword
8 columns, 3 rows
ABC id ABC full_name ABC city ABC phonenumber ABC age ABC city_logic ABC logic_name 12L 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

```

```

1 df_join=df_hashkey.alias("src").join(dbtable.toDF().alias("tgt"),(col("src.id")==col("tgt.id"))&(col("src.src_hashkey")==col("tgt.hashkey")))
2 display(df_join)
✓ 1 sec - Command executed in 1 sec 585 ms by NANDINI RATHORE on 3:16:52 PM, 8/04/25
> Spark jobs (2 of 2 succeeded) Resources
Table view Data Wrangler Download Filter by keyword
8 columns, 3 rows
ABC id ABC full_name ABC city ABC phonenumber ABC age ABC city_logic ABC logic_name 12L 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

```

Join with target table.

```

1 from pyspark.sql.functions import col, current_timestamp, lit
2
3 dbtable.alias("tgt").merge(
4     df_join.alias("src"),
5     "src.id == tgt.id"
6 ).whenMatchedUpdate(
7     set={
8         "id": col("src.id"),
9         "full_name": col("src.full_name"),
10        "city": col("src.city"),
11        "phonenumber": col("src.phonenumber"),
12        "age": col("src.age"),
13        "city_logic": col("src.city_logic"),
14        "LOGIC_NAME": col("src.LOGIC_NAME"),
15        "updateddate": current_timestamp(),
16        "updatedby": lit("fabric-updated"),
17        "hashkey": col("src.src_hashkey")
18    }

```

```

19     ).whenNotMatchedInsert(
20     values={
21         "id": col("src.id"),
22         "full_name": col("src.full_name"),
23         "city": col("src.city"),
24         "phonenumber": col("src.phonenumber"),
25         "age": col("src.age"),
26         "city_logic": col("src.city_logic"),
27         "LOGIC_NAME": col("src.LOGIC_NAME"),
28         "updateddate": current_timestamp(),
29         "updatedby": lit("fabric"),
30         "createdby": lit("fabric"),
31         "createddate": current_timestamp(),
32         "hashkey": col("src.src_hashkey")
33     }
34     ).execute()

```

1 %%sql  
2 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

123 id ↑	ABC full_name	ABC city	12L phonenumber	123 age	ABC city_logic	ABC logic_name	ABC createdby	createddate	ABC 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”.

YouTube Maps Gmail News Translate Screenshot showing... GIT REPOSITORY UP... Course: DP-700: Fab... Home - Snowflake All Bookmarks

Notebook 3 | Saved

Home Edit Run View

Comments History Develop Share

Explorer Data items Resources

+ Add data items

Items

- bronze
  - Tables
- Files
  - \_delta\_log
  - 2025
  - container
  - external\_data
  - final\_clean\_file
  - folder1

1 '.save("abfss://Nworkspace@onelake.dfs.fabric.microsoft.com/bronze.Lakehouse/Files/final\_clean\_file")

[37] ✓ - Command executed in 2 sec 591 ms by NANDINI RATHORE on 3:31:21 PM, 8/04/25

PySpark (Python)

Write that file into parquet format in lakehouse.

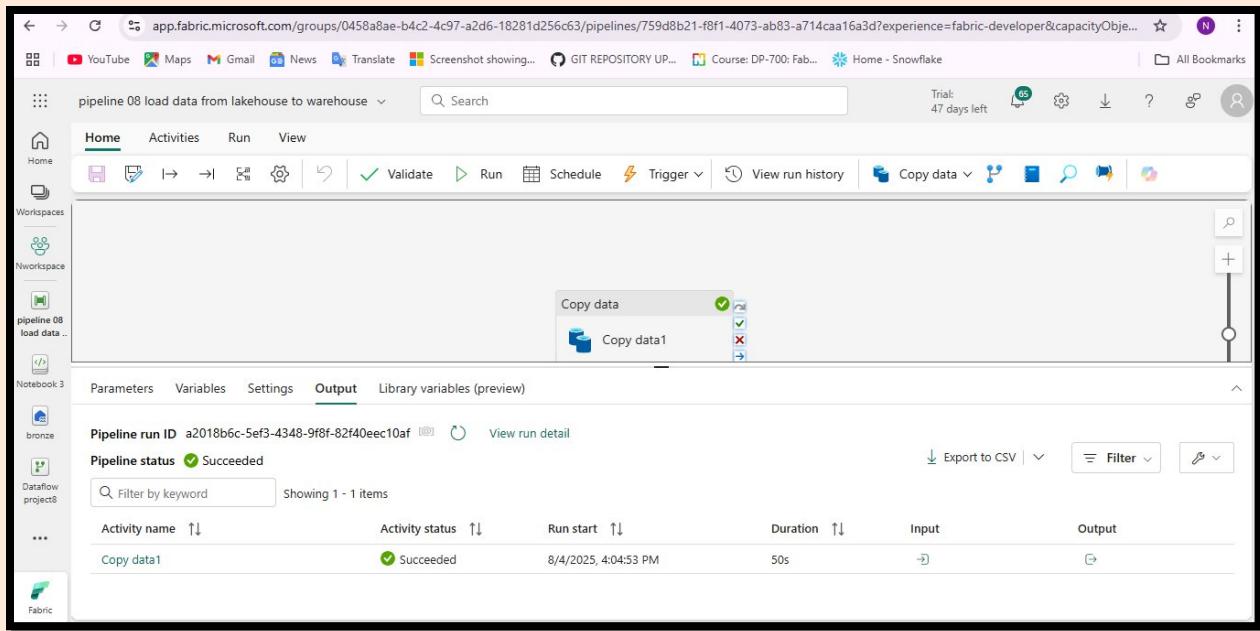
The screenshot shows the 'Copy data' interface in Microsoft Fabric. The 'Source' tab is selected. The 'Connection' dropdown is set to 'bronze'. Under 'Root folder', 'Files' is selected. Under 'File path type', 'File path' is selected with the value 'final\_clean\_file'. The 'Recursively' checkbox is checked. Under 'File format', 'Parquet' is selected. Other tabs like 'Destination', 'Mapping', and 'Settings' are visible but not selected.

After that create a new pipeline.

Select source lakehouse ,choose file path,file formatàparquet.

The screenshot shows the 'Copy data' interface in Microsoft Fabric. The 'Destination' tab is selected. The 'Connection' dropdown is set to 'new warehouse'. Under 'Table option', 'Auto create table' is selected. Under 'Table', 'dbo' is selected and 'final project' is typed into the input field. A tooltip 'Add dynamic content [Alt+Shift+D]' is shown next to the table name input. Other tabs like 'General', 'Source', and 'Mapping' are visible but not selected.

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.