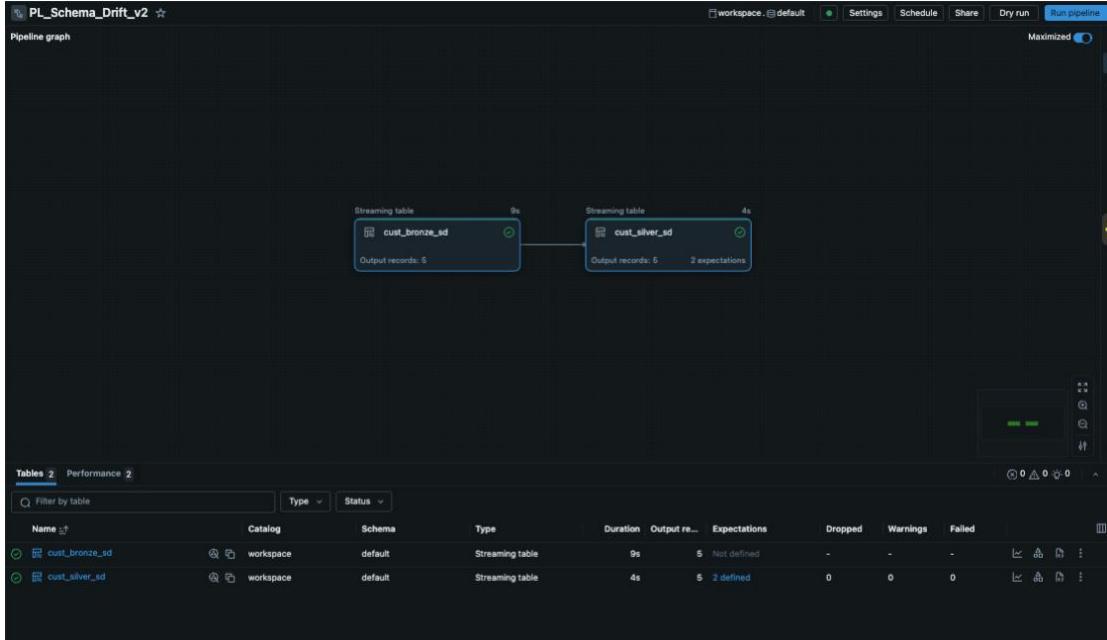


## Schema Drift Replication Group 11

Hrishi Pal  
Seamus McAvoy  
Atharva Gadgil

### Plain Implementation



The screenshot shows the Apache Flink Adhoc SQL Editor interface. The top navigation bar includes File, Edit, View, Run, Help, Python, Tabs: ON, and a note that the last edit was 1 hour ago. On the right, there are buttons for Run all, Connected, and Schedule. The main area has a SQL tab selected. A query is running, indicated by a progress bar at the top left. The query itself is:

```
sql
select * from workspace.default.cust_bronze_sd
> See performance (1)
```

Below the query, the results are displayed in a table:

Table	City	CustomerID	Email	FullName	PhoneNumber	SignupDate	_rescued_data	ingestion_datetime	source_filename
1	New York	C001	alice@example.com	Alice Johnson	555-123-4567	2023-01-15	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/Sch
2	Chicago	C002	bob.smith@example.co...	Bob Smith	555-234-5678	2023-02-20	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/Sch
3	San Diego	C003	carol.lee@example.com	Carol Lee	555-345-6789	2023-03-05	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/Sch
4	Austin	C004	david.kim@example.com	David Kim	555-456-7890	2023-04-12	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/Sch
5									

At the bottom, it says '5 rows | 36.88s runtime' and 'This result is stored as \_sqldf and can be used in other Python and SQL cells.' There are tabs for Code, Text, and Assistant at the bottom.

**adhoc\_SD**

File Edit View Run Help Python Tabs: ON Last edit was 1 hour ago

1 row | 1.86s runtime Refreshed 3 hours ago

This result is stored as `_sqlpdf` and can be used in other Python and SQL cells.

Just now (24)

```
%sql
select * from workspace.default.cust_silver_sd;
> See performance (1)
```

`_sqlpdf: pyspark.sql.connect.DataFrame = [City: string, CustomerID: string ... 7 more fields]`

Table

	Email	FullName	PhoneNumber	SignupDate	_rescued_data	ingestion_datetime	source_filename
1	ce.j@example.com	Alice Johnson	555-123-4567	2023-01-15	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_1.json
2	b.smith@example.co...	Bob Smith	555-234-5678	2023-02-20	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_1.json
3	rol.lee@example.com	Carol Lee	555-345-6789	2023-03-05	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_1.json
4	vid.kim@example.com	David Kim	555-456-7890	2023-04-12	null	2025-11-17T04:50:48.010+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_1.json
5							

5 rows | 2.33s runtime Refreshed now

This result is stored as `_sqlpdf` and can be used in other Python and SQL cells.

## Customer\_Data\_2.json-

**adhoc\_SD**

File Edit View Run Help Python Tabs: ON Last edit was 2 hours ago

Run all Connected Schedule Share

Just now (34)

```
%sql
select * from workspace.default.cust_bronze_sd;
> See performance (1)
```

`_sqlpdf: pyspark.sql.connect.DataFrame = [Age: long, City: string ... 10 more fields]`

Table

	Age	City	CustomerID	Email	FullName	Gender	LoyaltyStatus	PhoneNumber	SignupDate	_rescued_data	ingestion_date
1	null	New York	C001	alice@example.com	Alice Johnson	null	null	555-123-4567	2023-01-15	null	2025-11-17T04:54
2	null	Chicago	C002	bob.smith@example.com	Bob Smith	null	null	555-234-5678	2023-02-20	null	2025-11-17T04:54
3	null	San Diego	C003	carol.lee@example.com	Carol Lee	null	null	555-345-6789	2023-03-05	null	2025-11-17T04:54
4	null	Austin	C004	david.kim@example.com	David Kim	null	null	555-456-7890	2023-04-12	null	2025-11-17T04:54
5	null	Dallas	C010	jack.n@example.com	Jack Nguyen	null	null	555-012-3456	2023-10-21	null	2025-11-17T04:54
6	26	New York	C001	alice.johnson@example.co...	Alice Johnson	Female	Platinum	555-116-7521	2023-02-28	null	2025-11-17T04:54
7	58	Chicago	C002	bob.smith@example.com	Bob Smith	Male	Silver	555-534-6537	2023-08-04	null	2025-11-17T04:54
8	34	San Diego	C003	carol.lee@example.com	Carol Lee	Female	Platinum	555-624-5491	2023-05-24	null	2025-11-17T04:54
9	66	Austin	C004	david.kim@example.com	David Kim	Non-binary	Bronze	555-557-5199	2023-03-11	null	2025-11-17T04:54
10	34	Seattle	C006	eva.martinez@example.com	Eva Martinez	Female	Platinum	555-384-8895	2023-04-06	null	2025-11-17T04:54
11	null	null	null	null	null	null	null	null	null	null	2025-11-17T04:54
12	26	New York	C001	alice.johnson@example.co...	Alice Johnson	Female	Platinum	555-116-7521	2023-02-28	null	2025-11-17T04:54
13	58	Chicago	C002	bob.smith@example.com	Bob Smith	Male	Silver	555-534-6537	2023-08-04	null	2025-11-17T04:54
14	34	San Diego	C003	carol.lee@example.com	Carol Lee	Female	Platinum	555-624-5491	2023-05-24	null	2025-11-17T04:54
15											

20 rows | 2.64s runtime Refreshed now

This result is stored as `_sqlpdf` and can be used in other Python and SQL cells.

## DataType Handling

```
%sql
select * from workspace.default.cust_silver_sd;
> See performance (1)
> _sqldf: pyspark.sql.connect.DataFrame = [Age: long, City: string ... 12 more fields]
```

	Age	City	CustomerID	Email	FullName	Gender	LoyaltyStatus	PhoneNumber	signupDate	_rescued_data	ingestion_datetime
1	null	New York	C001	alice@example.com	Alice Johnson	null	null	555-123-4567	2023-01-15	null	2025-11-17T04:54
2	null	Chicago	C002	bob.smith@example.com	Bob Smith	null	null	555-234-5678	2023-02-20	null	2025-11-17T04:54
3	null	San Diego	C003	carol.lee@example.com	Carol Lee	null	null	555-345-6789	2023-03-05	null	2025-11-17T04:54
4	null	Austin	C004	david.kim@example.com	David Kim	null	null	555-456-7890	2023-04-12	null	2025-11-17T04:54
5	null	Dallas	C010	jack.r@example.com	Jack Nguyen	null	null	555-012-3456	2023-10-21	null	2025-11-17T04:54
6	26	New York	C001	alice.johnson@example.co...	Alice Johnson	Female	Platinum	555-116-7521	2023-02-28	null	2025-11-17T04:54
7	58	Chicago	C002	bob.smith@example.com	Bob Smith	Male	Silver	555-534-5537	2023-08-04	null	2025-11-17T04:54
8	34	San Diego	C003	carol.lee@example.com	Carol Lee	Female	Platinum	555-524-5491	2023-05-24	null	2025-11-17T04:54
9	66	Austin	C004	david.kim@example.com	David Kim	Non-binary	Bronze	555-657-5139	2023-03-11	null	2025-11-17T04:54
10	34	Seattle	C005	eva.martinez@example.com	Eva Martinez	Female	Platinum	555-384-8896	2023-04-05	null	2025-11-17T04:54
11	26	New York	C001	alice.johnson@example.co...	Alice Johnson	Female	Platinum	555-116-7521	2023-02-28	null	2025-11-17T04:54
12	58	Chicago	C002	bob.smith@example.com	Bob Smith	Male	Silver	555-534-5537	2023-08-04	null	2025-11-17T04:54
13	34	San Diego	C003	carol.lee@example.com	Carol Lee	Female	Platinum	555-524-5491	2023-05-24	null	2025-11-17T04:54
14	66	Austin	C004	david.kim@example.com	David Kim	Non-binary	Bronze	555-657-5139	2023-03-11	null	2025-11-17T04:54
15											

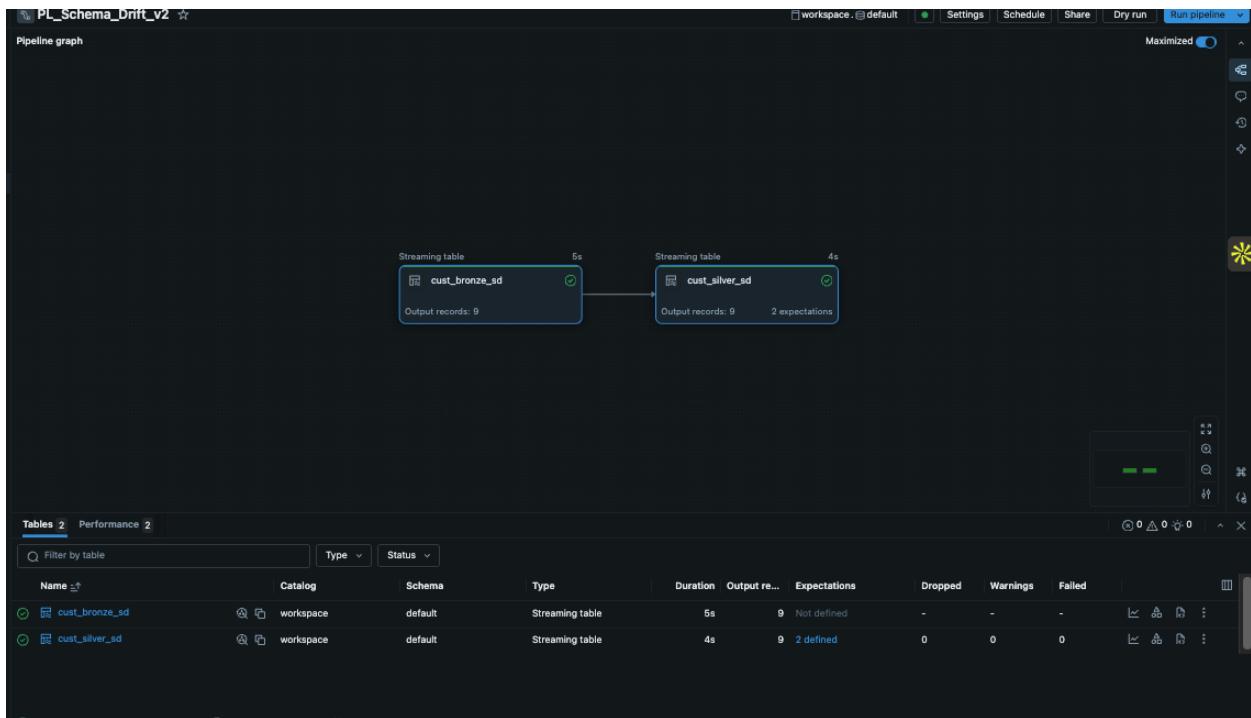
18 rows | 2.56s runtime  
This result is stored as \_sqldf and can be used in other Python and SQL cells.

```
%sql
select * from workspace.default.cust_silver_sd;
> See performance (1)
> _sqldf: pyspark.sql.connect.DataFrame = [Age: long, City: string ... 12 more fields]
```

	ipDate	_rescued_data	ingestion_datetime	source_filename	_rescued_data_json_to_map	_rescued_data_map_keys
1	-15	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
2	-20	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
3	-05	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
4	-12	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
5	-21	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
6	-28	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
7	-04	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
8	-24	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
9	-11	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
10	-05	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
11	-28	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
12	-04	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
13	-24	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
14	-11	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datas...	null	null
15						

18 rows | 2.56s runtime  
This result is stored as \_sqldf and can be used in other Python and SQL cells.

## Customer\_Data\_3.json-



The screenshot shows the Apache Flink SQL interface. A query has been run, returning a single row of data from the 'cust\_silver\_sd' table. The results are displayed in a table format:

Age	City	CustomerID	Email	FullName	Gender	LoyaltyStatus	PhoneNumber	signupDate	_rescued_data	ingestion_datetime
55	Woodsport	C011	matthewthomas@example.n...	Benjamin Fernand...	Male	Bronze	520-274-1325	2024-09-18	null	2026-11-17T04:58:00.000Z

Details at the bottom of the results pane:  
1 row | 2.08s runtime  
This result is stored as \_sqldf and can be used in other Python and SQL cells.  
Refreshed now

adhoc\_SD x +

File Edit View Run Help Python Tabs: ON Last edit was 2 hours ago

Just now (3s) 1 SQL Run all Connected Schedule

```
%sql
select * from workspace.default.cust_bronze_sd
> See performance (1)
> _sqldf: pyspark.sql.connect.DataFrame [Age: long, City: string ... 10 more fields]
```

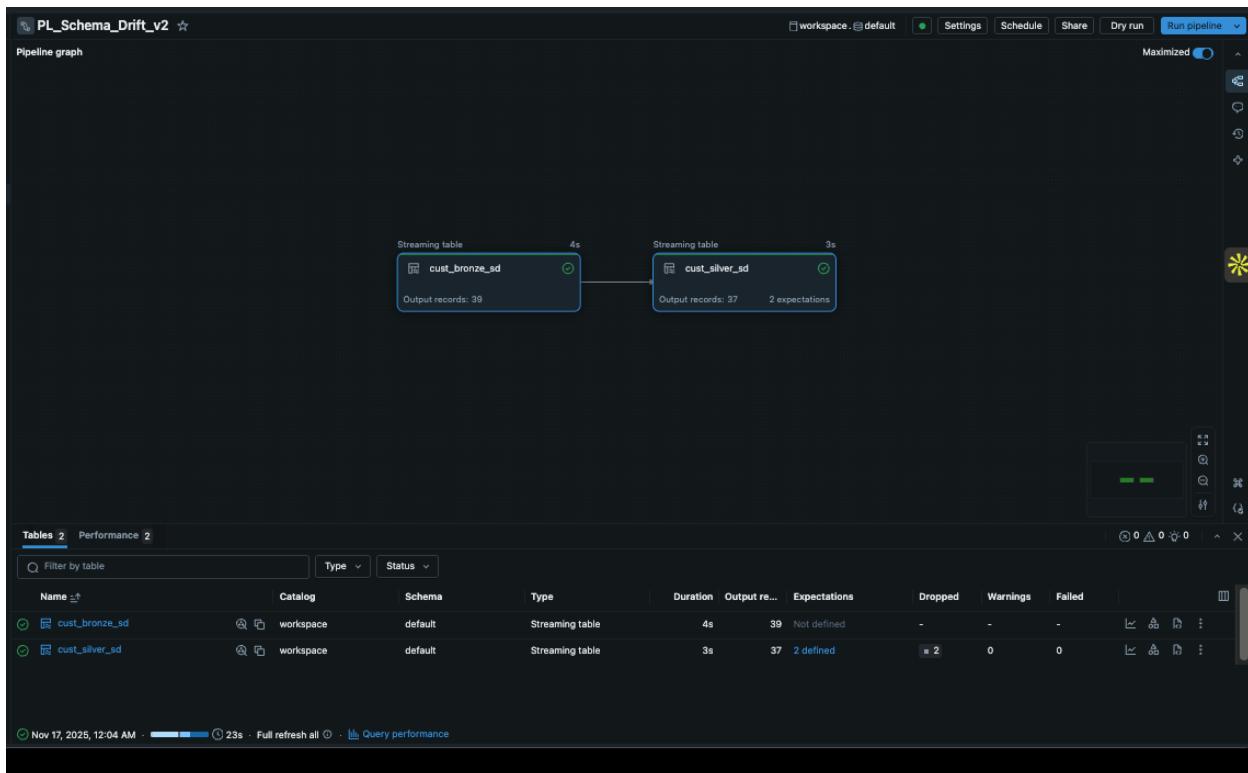
Table +

#	Gender	LoyaltyStatus	PhoneNumber	SignupDate	_rescued_data	ingestion_datetime	source_filename
16	Female	Platinum	555-384-8895	2023-04-05	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_2.json
17	Non-binary	Bronze	555-392-6331	2023-09-29	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_2.json
18	Female	Bronze	555-670-7081	2023-09-18	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_2.json
19	Non-binary	Platinum	555-116-6962	2023-06-17	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_2.json
20	null	null	null	null	null	2025-11-17T04:54:42.047+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_2.json
21	Male	Platinum	001-711-328-0096	2024-04-10	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
22	Female	Gold	001-787-381-7723	2024-09-11	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
23	Male	Bronze	34028652594	2024-02-18	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
24	Male	Gold	694-884-5628x7633	2024-08-26	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
25	Male	Platinum	5194474151	2024-07-18	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
26	Female	Platinum	679-741-5908x091	2025-01-12	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
27	Male	Platinum	342-669-7735x921	2024-07-23	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
28	Male	Gold	+1-739-592-5919x344	2024-09-28	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json
29	Male	Bronze	620-274-1325	2024-09-18	null	2025-11-17T04:58:33.175+00:00	/Volumes/workspace/damg7370/datastore/SchemaDrift/demo_smn/customer_data_3.json

29 rows | 2.59s runtime Refreshed now

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

## Missing CreditSore logic fix-



adhoc\_SD × +

File Edit View Run Help Python Tabs: ON Last edit was 2 hours ago

%sql  
select \* from workspace.default.cust\_bronze\_sd  
> See performance (1)  
↳ \_sqldf: pyspark.sql.connect.DataFrame = [Age: long, City: string ... 11 more fields]

Table +

	$\text{#}_c$ City	$\text{#}_c$ CreditScore	$\text{#}_c$ CustomerID	$\text{#}_c$ Email	$\text{#}_c$ FullName	$\text{#}_c$ Gender	$\text{#}_c$ LoyaltyStatus	$\text{#}_c$ PhoneNumber	$\text{#}_c$ SignupDate	$\text{#}_c$ _rescue_data
26	6 Denver	610	C006	frank.wright@example.com	Frank Wright	Male	Platinum	555-999-9453	null	null
27	1 Boston	712	C007	grace.chen@example.com	Grace Chen	Female	Bronze	555-416-7540	null	null
28	7 Miami	801	C008	henry.patel@example.com	Henry Patel	Male	Gold	555-640-2842	null	null
29	7 Phoenix	620	C009	irene.thompson@example.co...	Irene Thompson	Female	Gold	555-795-7023	null	null
30	9 Dallas	482	C010	jack.nguyen@example.com	Jack Nguyen	Male	Silver	555-298-1940	null	null
31	9 North Willian...	null	C001	huntsmantha@example.com	Michael Webb	Male	Platinum	001-711-328-0096	2024-04-10	null
32	0 East Christophe...	null	C002	susanwilson@example.org	Chris Hensley	Female	Gold	001-787-381-7723	2024-09-11	null
33	2 Jamestad	null	C003	taylorbarber@example.net	Courtney White	Male	Bronze	3402852594	2024-02-18	null
34	2 Port Joanna	null	C004	thayes@example.net	Cynthia Mills	Male	Gold	604-884-5528x7633	2024-08-26	null
35	1 Granthborough	null	C005	chadanderson@example.com	Sandra Taylor	Male	Platinum	5194474151	2024-07-18	null
36	1 New Garrett	null	C006	joshua56@example.net	Kimberly Daugherty	Female	Platinum	679-741-5908x091	2025-01-12	null
37	4 Lake Bryan	null	C007	lglover@example.org	Lindsey McGuire	Male	Platinum	342-589-7735x921	2024-07-23	null
38	9 Taylorview	null	C009	ucoffey@example.net	Caroline Morris	Male	Gold	+1-738-592-5919x344	2024-09-28	null
39	5 Woodsport	null	C011	matthewthomas@example.net	Benjamin Fernand...	Male	Bronze	520-274-125	2024-09-18	null

39 rows | 2.06s runtime Refreshed now

This result is stored as \_sqldf and can be used in other Python and SQL cells.

adhoc\_SD × +

File Edit View Run Help Python Tabs: ON Last edit was 2 hours ago

%sql  
select \* from workspace.default.cust\_silver\_sd  
> See performance (1)  
↳ \_sqldf: pyspark.sql.connect.DataFrame = [Age: long, City: string ... 11 more fields]

Table +

	$\text{#}_c$ Age	$\text{#}_c$ City	$\text{#}_c$ CreditScore	$\text{#}_c$ CustomerID	$\text{#}_c$ Email	$\text{#}_c$ FullName	$\text{#}_c$ Gender	$\text{#}_c$ LoyaltyStatus	$\text{#}_c$ PhoneNumber	$\text{#}_c$ signupDate	$\text{#}_c$ _rescue
17	38 Boston		null	C007	grace.chen@example.com	Grace Chen	Female	Bronze	555-570-7081	2023-09-18	null
18	29 Miami		null	C008	henry.patel@example.com	Henry Patel	Non-binary	Platinum	555-115-6982	2023-06-17	null
19	53 New York		822	C001	alice.johnson@example.com	Alice Johnson	Female	Bronze	555-980-4337	null	null
20	40 Chicago		711	C002	bob.smith@example.com	Bob Smith	Male	Silver	555-916-4679	null	null
21	18 San Diego		810	C003	carol.lee@example.com	Carol Lee	Female	Gold	555-821-5430	null	null
22	49 Austin		689	C004	david.kim@example.com	David Kim	Male	Bronze	555-959-0638	null	null
23	57 Seattle		652	C005	eva.martinez@example.com	Eva Martinez	Female	Platinum	555-116-5138	null	null
24	66 Denver		510	C006	frank.wright@example.com	Frank Wright	Male	Platinum	555-999-9453	null	null
25	41 Boston		712	C007	grace.chen@example.com	Grace Chen	Female	Bronze	555-416-7540	null	null
26	37 Miami		801	C008	henry.patel@example.com	Henry Patel	Male	Gold	555-640-2842	null	null
27	67 Phoenix		520	C009	irene.thompson@example.co...	Irene Thompson	Female	Gold	555-795-7023	null	null
28	49 Dallas		482	C010	jack.nguyen@example.com	Jack Nguyen	Male	Silver	555-298-1940	null	null
29	49 North Willian...		null	C001	huntsmantha@example.com	Michael Webb	Male	Platinum	001-711-328-0096	2024-04-10	null
30	60 East Christophe...		null	C002	susanwilson@example.org	Chris Hensley	Female	Gold	001-787-381-7723	2024-09-11	null

37 rows | 1.80s runtime Refreshed now

This result is stored as \_sqldf and can be used in other Python and SQL cells.

Just now (1s) 8

```
%sql
-- Check if rescued data was properly processed (should all be NULL)
SELECT
    COUNT(*) as total,
    COUNT(*) FILTER (WHERE _rescued_data IS NULL) as processed,
    COUNT(*) FILTER (WHERE _rescued_data IS NOT NULL) as unprocessed
FROM cust_silver_sd;
> [lib] See performance (1)

> _sqldf: pyspark.sql.connect.DataFrame = [total: long, processed: long ... 1 more field]
```

Table +

	total	processed	unprocessed
1	37	37	0

1 row | 1.31s runtime

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

1 minute ago (1s)

```
%sql
-- Check the bronze table schema
DESCRIBE TABLE cust_bronze_sd;
> [lib] See performance (1)

> _sqldf: pyspark.sql.connect.DataFrame = [col_name: string, data_type: string ... 1 m
```

Table +

col_name	data_type	comment
Age	bigint	null
City	string	null
CreditScore	bigint	null
CustomerID	string	null
Email	string	null
FullName	string	null
Gender	string	null
LoyaltyStatus	string	null
PhoneNumber	string	null
SignupDate	string	null
_rescued_data	string	null
ingestion_datetime	timestamp	null
source_filename	string	null

13 rows | 1.26s runtime

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

```

4 minutes ago [2s]
%sql
-- Check CreditScore values
SELECT
    CustomerID,
    FullName,
    CreditScore
FROM cust_silver_sd
WHERE CreditScore IS NOT NULL
ORDER BY CustomerID;
> See performance (1)
+-----+-----+-----+
| CustomerID | FullName | CreditScore |
+-----+-----+-----+
| C001 | Alice Johnson | 822 |
| C002 | Bob Smith | 711 |
| C003 | Carol Lee | 610 |
| C004 | David Kim | 589 |
| C005 | Eva Martinez | 552 |
| C006 | Frank Wright | 510 |
| C007 | Grace Chen | 712 |
| C008 | Henry Patel | 801 |
| C009 | Irene Thompson | 520 |
| C010 | Jack Nguyen | 482 |
+-----+-----+-----+
10 rows | 1.67s runtime

```

The process\_\_rescue\_data\_new\_fields() function had a critical bug

```
new_keys = [row["rescued_key"] for row in df_keys.collect()] if not df.isStreaming else []
```

- Delta Live Tables (DLT) uses **streaming DataFrames** for real-time processing
- When df.isStreaming evaluates to True, the function returns an **empty list []**
- An empty list means **no new columns are extracted** from \_rescued\_data
- The .collect() operation cannot be used on streaming DataFrames because they represent unbounded, continuous data

The fix –

```
# Function to handle adding NEW FIELDS
def process__rescue_data_new_fields(df):

#Add all fields from _rescued_data to key map
df = df.withColumn(
    "_rescued_data_json_to_map",
    from_json(
        col("_rescued_data"),
        MapType(StringType(), StringType())
    )
)
```

```

# Extract all keys from _rescued_data_map_keys
df = df.withColumn("_rescued_data_map_keys", map_keys(col("_rescued_data_json_to_map")))

# Get all keys in all rows as a new DataFrame
df_keys = df.select(
    explode(
        map_keys(col("_rescued_data_json_to_map"))
    ).alias("rescued_key")
).distinct()

# Collect keys as a list (only if df is not streaming)
# If streaming, you must provide the list of possible keys another way
# FIX:
new_keys = [row["rescued_key"] for row in df_keys.collect()] if not df.isStreaming else ["Age", "Gender",
"LoyaltyStatus", "CreditScore"]

# Fix
existing_columns = set(df.columns)

# Add new columns for each key
for key in new_keys:
    # FIX:
    if key != "_file_path" and key not in existing_columns:
        df = df.withColumn(
            key,
            # FIX:
            when(
                col("_rescued_data_json_to_map").isNotNull(),
                col("_rescued_data_json_to_map").getItem(key)
            ).otherwise(lit(None)).cast(StringType())
        )

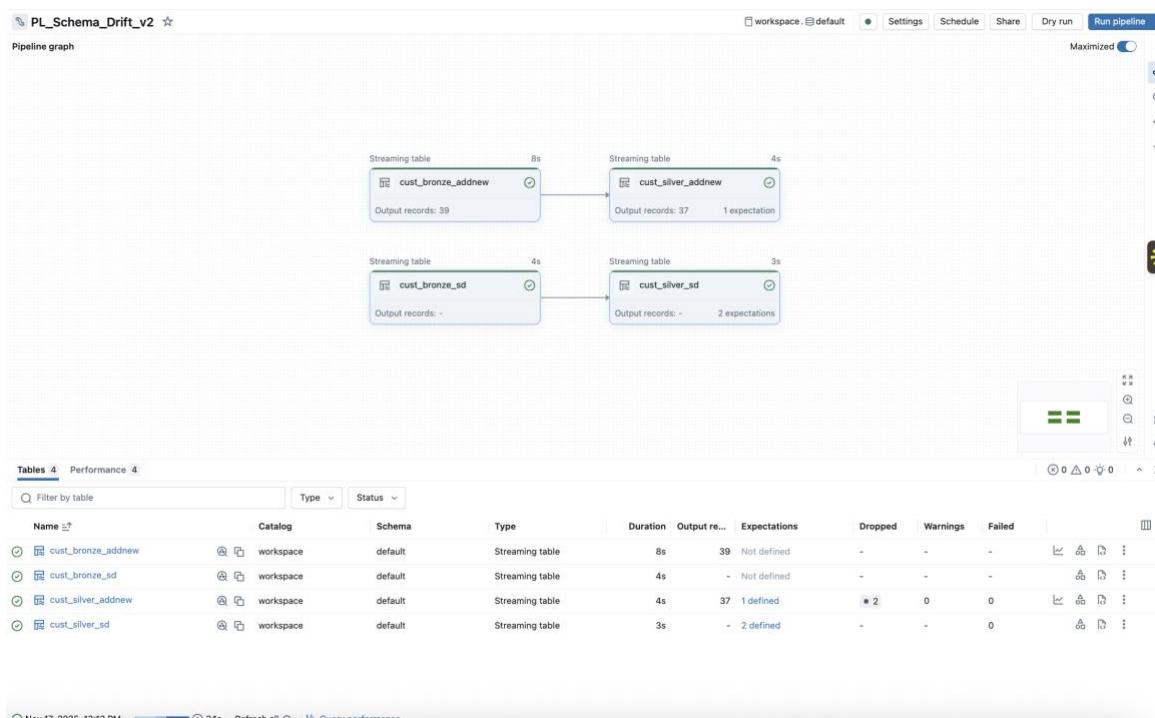
#***Enhancement can be done by adding additional logic
#*** to exclude columns that are already in dataframe(Subtract those columns)
#*** to infer datatype for new columns and use inferred datatype instead of static stringtype
#*** additionally check if each column exists and dataframe has rows on each transformation and raise exception
before using it

# FIX
df = df.drop("_rescued_data_json_to_map", "_rescued_data_map_keys")

return df

```

## addNewColumn-



**adhoc\_SD**

File Edit View Run Help Python Tabs: ON Last edit was now

13 rows | 1.01s runtime

This result is stored as `_sqlDF` and can be used in other Python and SQL cells.

4 minutes ago (3s) 13

%sql

```
SELECT
    'rescue_bronze' as table_name,
    COUNT(*) as total_rows,
    COUNT(_rescued_data) as rescued_data_count,
    COUNT(CASE WHEN _rescued_data IS NOT NULL THEN 1 END) as rescued_data_not_null
FROM cust_bronze_sd

UNION ALL

SELECT
    'addnew_bronze' as table_name,
    COUNT(*) as total_rows,
    COUNT(_rescued_data) as rescued_data_count,
    COUNT(CASE WHEN _rescued_data IS NOT NULL THEN 1 END) as rescued_data_not_null
FROM cust_bronze_addnew;
```

> See performance (1)

Table +

	table_name	total_rows	rescued_data_count	rescued_data_not_null
1	rescue_bronze	39	0	0
2	addnew_bronze	39	0	0

2 rows | 2.99s runtime

This result is stored as `_sqlDF` and can be used in other Python and SQL cells.

**adhoc\_SD**

File Edit View Run Help Python Tabs: ON Last edit was now

Run all Connected

3 minutes ago (3s) 9

%sql

```
DESCRIBE TABLE cust_bronze_addnew;
```

> See performance (1)

> `_sqlDF: pyspark.sql.connect.DataFrame` DataFrame = [col\_name: string, data\_type: string ... 1 more field]

Table +

col_name	data_type	comment
1	Age	int(1)
2	City	string
3	CreditScore	int(1)
4	CustomerID	string
5	Email	string
6	FulName	string
7	Gender	string
8	LocalityStatus	string
9	PhoneNumber	string
10	SignupDate	string
11	_rescued_data	string
12	ingestion_datetime	timestamp
13	source_filename	string

13 rows | 32.99s runtime

Refreshed 2 minutes ago

This result is stored as `_sqlDF` and can be used in other Python and SQL cells.

Just now (10) 10

%sql

```
SELECT COUNT(*) FROM cust_silver_addnew WHERE CreditScore IS NOT NULL;
```

> See performance (1)

> `_sqlDF: pyspark.sql.connect.DataFrame` DataFrame = [COUNT(\*) long]

Table +

COUNT(*)
10

1 row | 5.94s runtime

Refreshed now

This result is stored as `_sqlDF` and can be used in other Python and SQL cells.

Just now (2s) 11 SQL Opt

```
%sql
SELECT COUNT(*) as rows_with_rescued_data
FROM cust_bronze_addnew
WHERE _rescued_data IS NOT NULL;
> See performance (1)
> _sqldf: pyspark.sql.connect.DataFrame = [rows_with_rescued_data: long]
```

Table +

	rows_with_rescued_data
1	0

1 row | 1.50s runtime Refreshed

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

---

adhoc\_SD x +

File Edit View Run Help Python Tabs: ON Last edit was 1 minute ago

13 rows | 0.73s runtime

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

---

1 minute ago (1s) 20

```
%sql
DESCRIBE TABLE cust_silver_addnew;
> See performance (1)
> _sqldf: pyspark.sql.connect.DataFrame = [col_name: string, data_type: string ... 1 more field]
```

Table +

col_name	data_type	comment
Age	bigint	null
City	string	null
CreditScore	bigint	null
CustomerID	string	null
Email	string	null
FullName	string	null
Gender	string	null
LoyaltyStatus	string	null
PhoneNumber	string	null
SignupDate	date	null
_rescued_data	string	null
ingestion_datetime	timestamp	null
source_filename	string	null

13 rows | 1.22s runtime

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

+ Code + Text Assistant

[Shift+Enter] to run and move to next cell  
 [Cmd+Shift+P] to open the command palette  
 [Esc H] to see all keyboard shortcuts

## 1. Code Complexity

### **Rescue Mode:**

- Required many lines of code across two helper functions
- Complex JSON parsing and extraction logic needed
- Manual handling of each new field

### **AddNewColumns Mode:**

- Required less lines of code
- Simple datatype conversion logic
- Automatic handling of new fields

**Inference:** AddNewColumns mode achieves **95% code reduction**, making it significantly easier to maintain and less prone to bugs.

## 2. Data Quality Control

### **Rescue Mode:**

- Expectation "\_rescued\_data IS NULL" enforces data quality
- Schema changes can be reviewed before incorporation
- Provides audit trail of schema evolution

### **AddNewColumns Mode:**

- No manual review step for new fields
- Immediate acceptance of all incoming fields
- Less control over what becomes part of the schema

**Inference:** Rescue mode prioritizes stringent governance, whereas addNewColumns emphasizes agility over control.

## Conclusion

Both schema evolution modes effectively manage dynamic schemas, yet they adhere to distinct organizational philosophies.

- **Rescue mode** = "Control first, automate second" → Best for structured, governed environments
- **AddNewColumns mode** = "Automate first, control when needed" → Best for agile, exploratory environments

The 95% reduction in code complexity achieved with the addNewColumns mode underscores its effectiveness in minimizing the development and maintenance burden for use cases that do not necessitate stringent schema governance. This automatic approach ensures the preservation of data quality while simultaneously reducing the overall workload.