Real-Time Data Analytics Pipeline using AWS, Apache NiFi, and Snowflake

By: Thrivikram Kotharu

Project Type: End-to-End Near Real-Time Data Engineering Project
Technologies: AWS EC2, Docker, Jupyter Notebook, Apache NiFi, Amazon S3, Snowflake, Snowpipe, Snowflake Tasks, Python (Faker)

1. Executive Summary

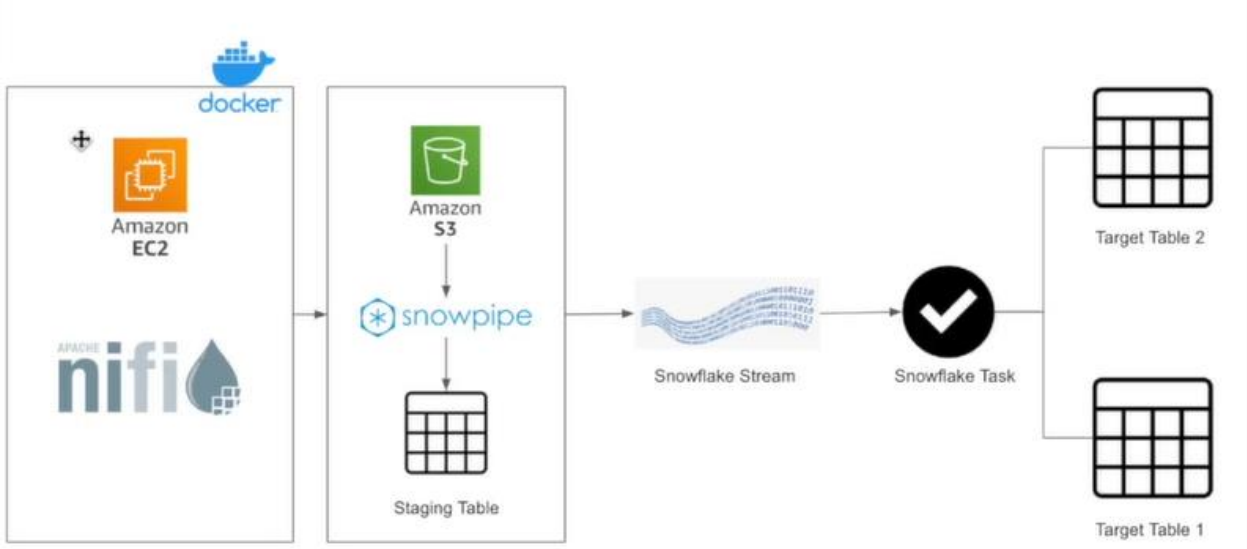
This project demonstrates a near real-time data ingestion and analytics pipeline built on AWS and Snowflake. Synthetic customer data is generated using Python (Faker) in Jupyter Notebook, streamed to Amazon S3 using Apache NiFi, and automatically ingested into Snowflake using Snowpipe. A scheduled Snowflake Task runs every minute to process the data and maintain a curated customer table using Slowly Changing Dimension (SCD) Type 1 logic.

The architecture simulates a real-world streaming ingestion pattern with micro-batch processing and automated orchestration.

2. Architecture Overview

High-Level Flow

1. EC2 (Amazon Linux) hosts Docker containers
2. Docker runs:
    - Jupyter Notebook (data generation)
    - Apache NiFi (file movement)
3. Python (Faker) generates CSV files (10,000 records per file)
4. Apache NiFi uploads files to Amazon S3
5. S3 Event Notification triggers Snowpipe
6. Snowpipe loads data into customer_raw
7. Snowflake Task (every 1 minute) calls a stored procedure
8. Stored procedure merges data into customer table (SCD Type 1)

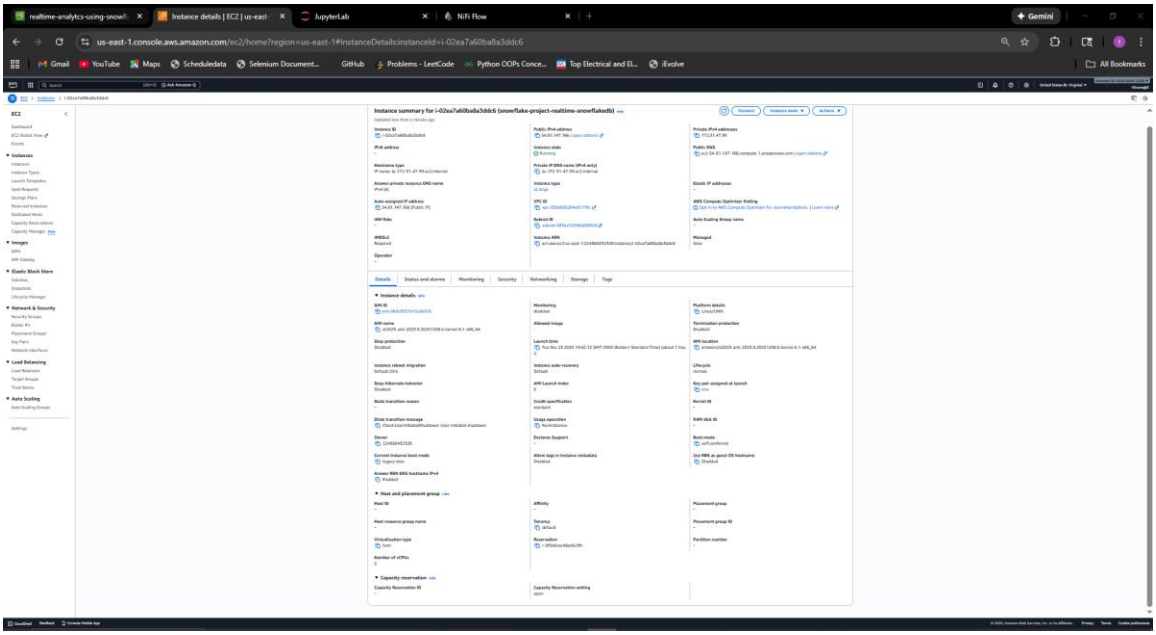


3. Infrastructure Setup

3.1 EC2 and Docker Setup

- EC2 instance launched with Amazon Linux
- Docker installed on EC2
- Docker images used:
    - Jupyter Notebook
    - Apache NiFi

Docker allows isolation and portability of services.



4. Data Generation using Python (Faker)

Objective

Generate realistic customer data to simulate streaming ingestion.

Key Characteristics

- 10,000 records per run
- Timestamp-based file naming
- Fields include customer name, email, address, and geography

Python Code (Faker)

```
from faker import Faker

import csv

from datetime import datetime

RECORD_COUNT = 10000

fake = Faker()

current_time = datetime.now().strftime("%Y%m%d%H%M%S")


def create_csv_file():

with open(f'data/customer_{current_time}.csv', 'w', newline='') as csvfile:

fieldnames = [

"customer_id","first_name","last_name","email",

"street","city","state","country"

]

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

writer.writeheader()

for i in range(RECORD_COUNT):

writer.writerow({

"customer_id": i,

"first_name": fake.first_name(),
```

"last_name": fake.last_name(),

"email": fake.email(),

"street": fake.street_address(),

"city": fake.city(),

"state": fake.state(),

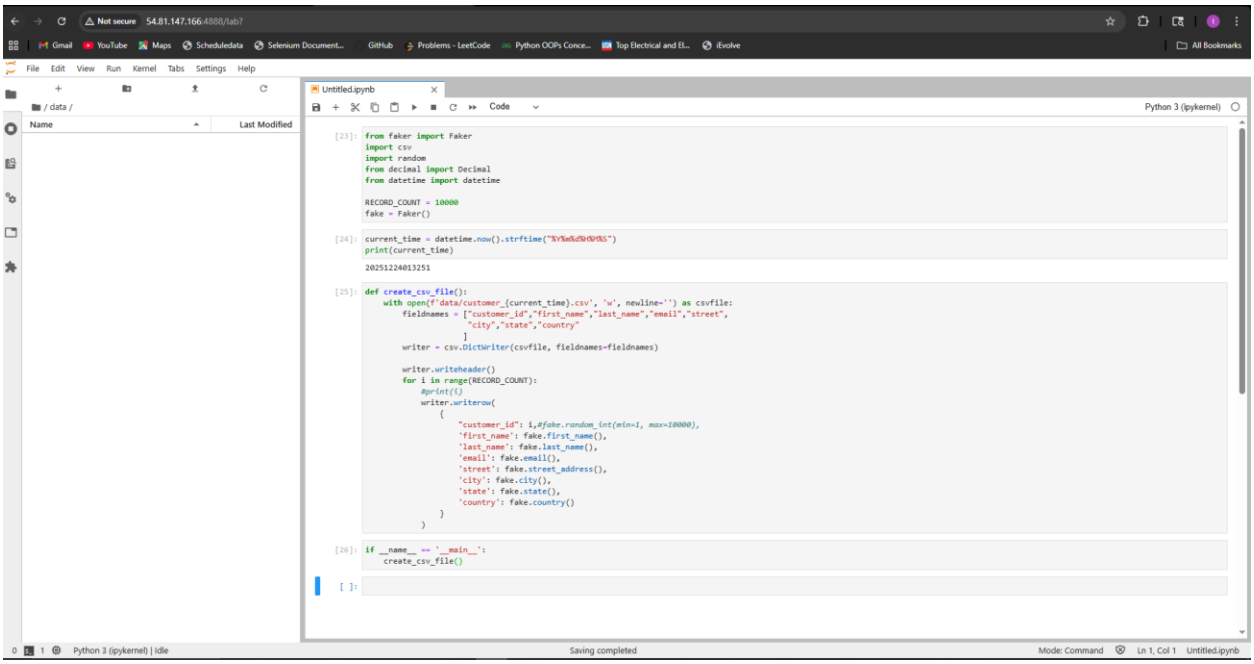
"country": fake.country()

})

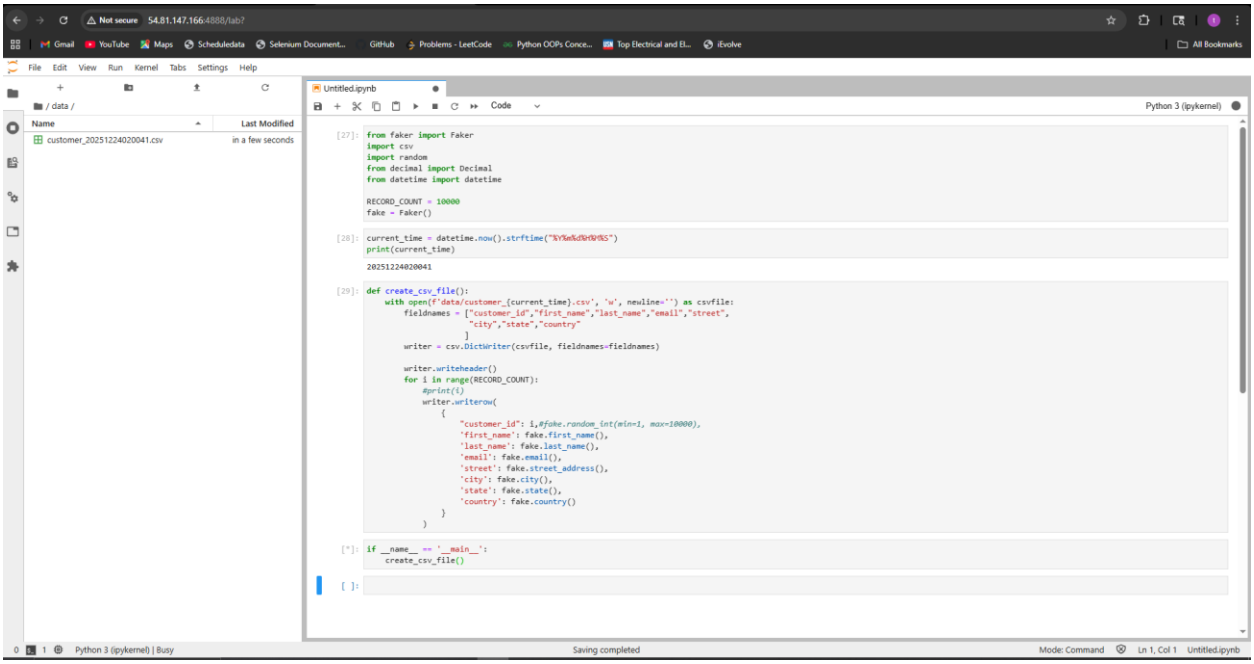
if __name__ == '__main__':

create_csv_file()

Before Running the code:



After Running the code:



Untitled.ipynb | customer_20251224020041.c

Delimiter: ,

	customer_id	first_name	last_name	email	street	city	state	country
1	0	Randy	Torres	imiller@example.net	Alexander Branch Apt. 813	East Sandra	South Carolina	Egypt
2	1	Ashley	Black	ghtelizabeth@example.net	63931 Alexander Curve	Gardnerchester	New Hampshire	Netherlands Antilles
3	2	Billy	Wright	ersonyvette@example.org	4602 Cannon Freeway	West Anna	Louisiana	Netherlands
4	3	Lori	Hughes	brownalexis@example.org	15884 Angela Summit	West Shane	California	Somalia
5	4	Whitney	Atkins	eric26@example.net	Jeremy Estates Suite 099	New Carlosstad	Washington	Saint Lucia
6	5	Brittany	Pearson	jason69@example.com	70919 Daniel Island	Lewiston	Louisiana	Montserrat
7	6	Debra	Sanchez	bishopgina@example.org	73978 Campbell Terrace	Rowlandtown	Nebraska	Chad
8	7	Danny	Dixon	bphillips@example.net	02 Moore Locks Suite 344	Turnerbury	New Hampshire	Eritrea
9	8	Heidi	Lee	wross@example.net	James Squares Suite 831	Port Andrew	Maine	Italy
10	9	Joseph	Woods	phillip01@example.com	671 Thomas Spurs	Smithhaven	Indiana	Libyan Arab Jamahiriya
11	10	Kimberly	Smith	kimberly25@example.com	93879 Karen Parks	Port Benjamin	Minnesota	Norway
12	11	Christopher	Dunn	chadtaylor@example.net	155 Wendy Junctions	West Mark	Missouri	Solomon Islands
13	12	David	Woods	laurawad@example.com	33 Reeves Roads Apt. 907	Andrewville	Texas	Moldova
14	13	Laura	Hill	lpham@example.org	38293 Brown Fort Apt. 778	New Brittanymouth	West Virginia	Cuba
15	14	Holly	Martinez	jjohnson@example.net	008 Sheila Cliffs Suite 965	North Johnhaven	Missouri	Zimbabwe
16	15	Courtney	Castillo	sonmallory@example.com	66105 Burke Run	Wrightshire	Utah	Tokelau
17	16	Gina	Williams	nkmartinez@example.com	74588 Calhoun Motorway	East Holly	West Virginia	Fiji
18	17	Angel	Fowler	acharylopez@example.net	679 Johnson Drives	East Ashley	Washington	Oman
19	18	Leslie	Martin	emily39@example.org	730 Donald Rest Suite 260	North Joseph	Nevada	Central African Republic
20	19	Christopher	Randall	dayariel@example.org	7146 Burgess Lake	Keithbury	Connecticut	Greece
21	20	Alexandria	Cortez	janet44@example.com	250 Peter Knolls Suite 079	Port Stephanieview	Oregon	Mauritania
22	21	Sean	Yang	qjohnson@example.net	2073 Christine Meadow	Shortstad	Montana	Cook Islands
23	22	Melissa	Duke	awilson@example.net	Bowman Stream Suite 215	Acostaville	Arizona	Luxembourg
24	23	Michael	Smith	sellpalmer@example.com	Hendricks Views Suite 721	Billyton	Alaska	Burkina Faso
25	24	Steven	Carrillo	orichardson@example.org	874 Elliott Views Suite 457	West Scott	Florida	Mexico
26	25	Monica	Aguirre	kellycombs@example.net	9397 James Haven	East James	Florida	Sri Lanka
27	26	Michael	Daniels	cheryl29@example.org	Margaret Villages Apt. 668	Port Courtneymouth	Minnesota	Thailand
28	27	Elizabeth	Peterson	ricerobert@example.net	aymond Gardens Apt. 078	New David	Kansas	Romania
29	28	Jeffrey	Jones	alkerjoseph@example.net	4 Darlene Glens Suite 402	North Manuelville	Massachusetts	Montserrat
30	29	Jason	Curtis	jsantos@example.net	6878 Simpson Spurs	Walterport	Utah	Haiti
31	30	Natalie	Merritt	daniel41@example.org	3654 Morrison Station	West Noah	Colorado	Maldives
32	31	Dominique	Thompson	coxzachary@example.org	554 Bryan Fork	Hollandport	Idaho	Gibraltar

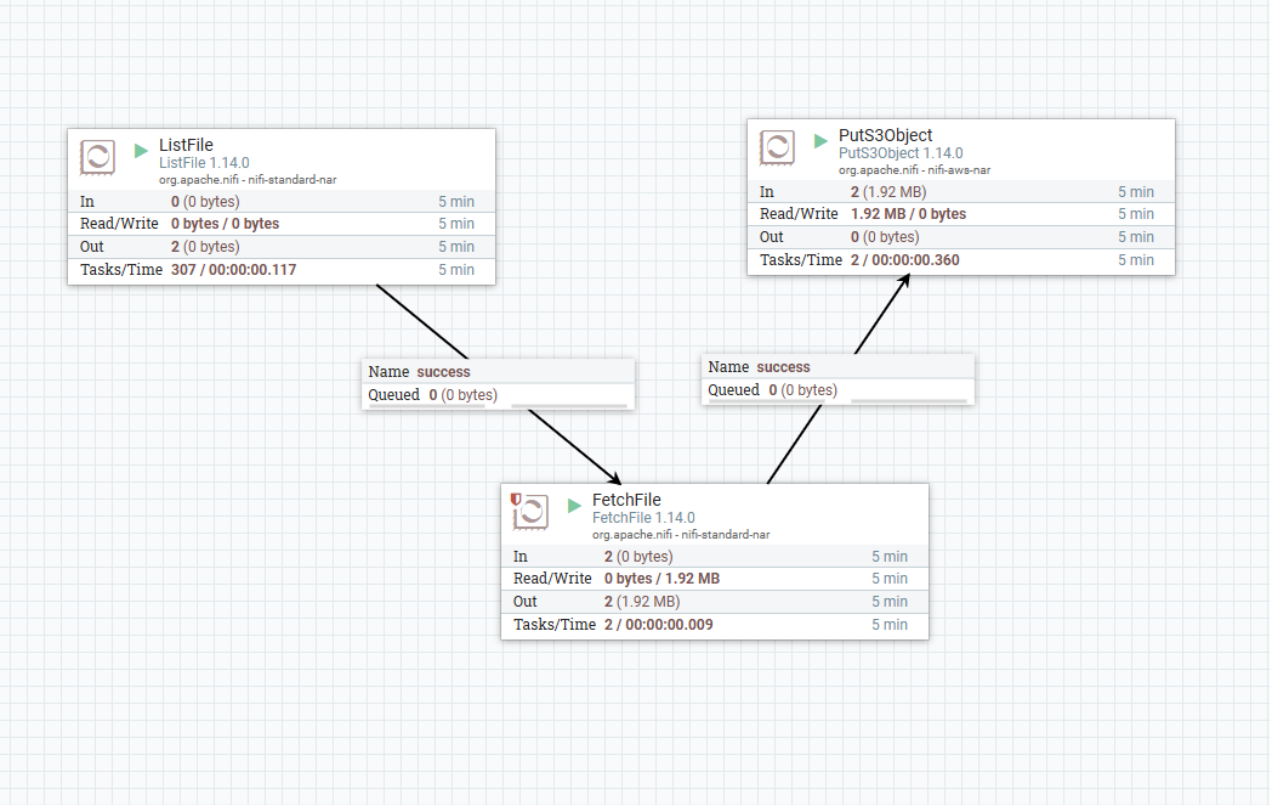
5. Apache NiFi Flow

NiFi Processors Used

- **ListFile** – Detect new CSV files
- **FetchFile** – Read file content
- **PutS3Object** – Upload files to Amazon S3

Purpose

NiFi automates file movement and ensures new data is pushed to S3 without manual intervention.

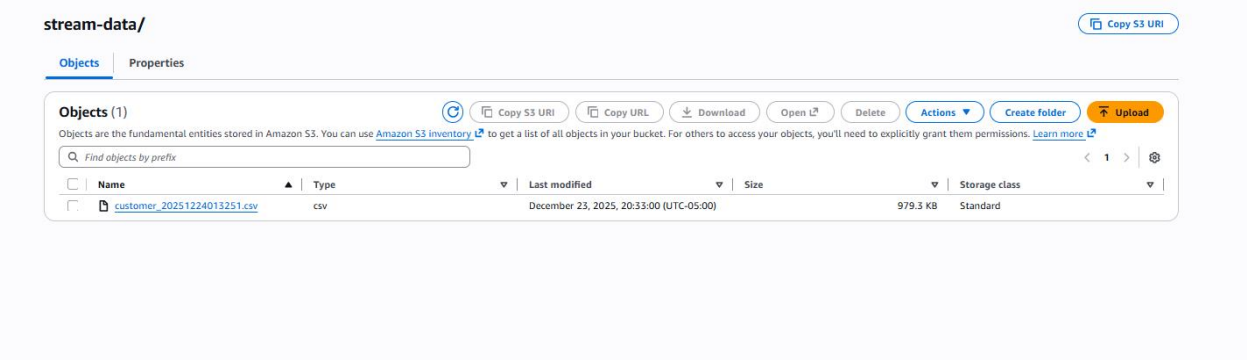


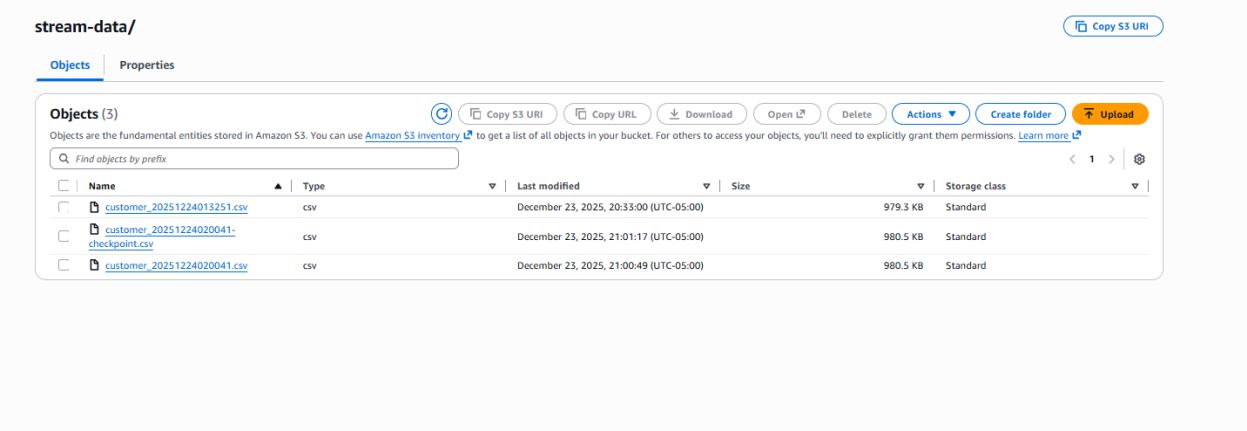
6. Amazon S3 (Landing Zone)

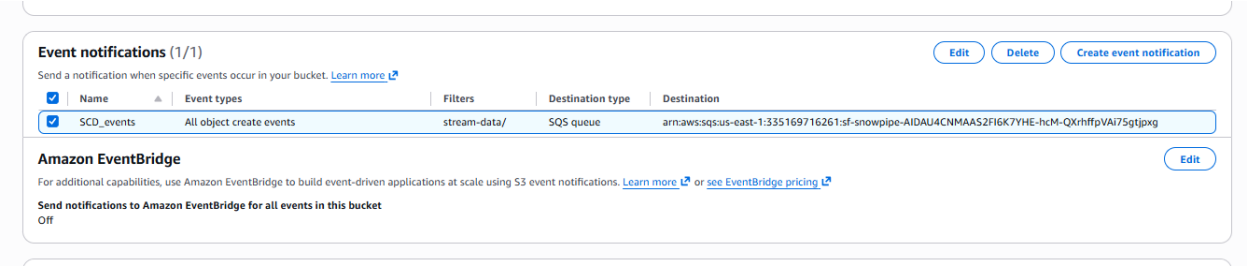
Bucket Structure

- Bucket: realtime-analytics-using-snowflake
- Prefix: stream-data/

Each CSV file uploaded triggers an S3 event notification.







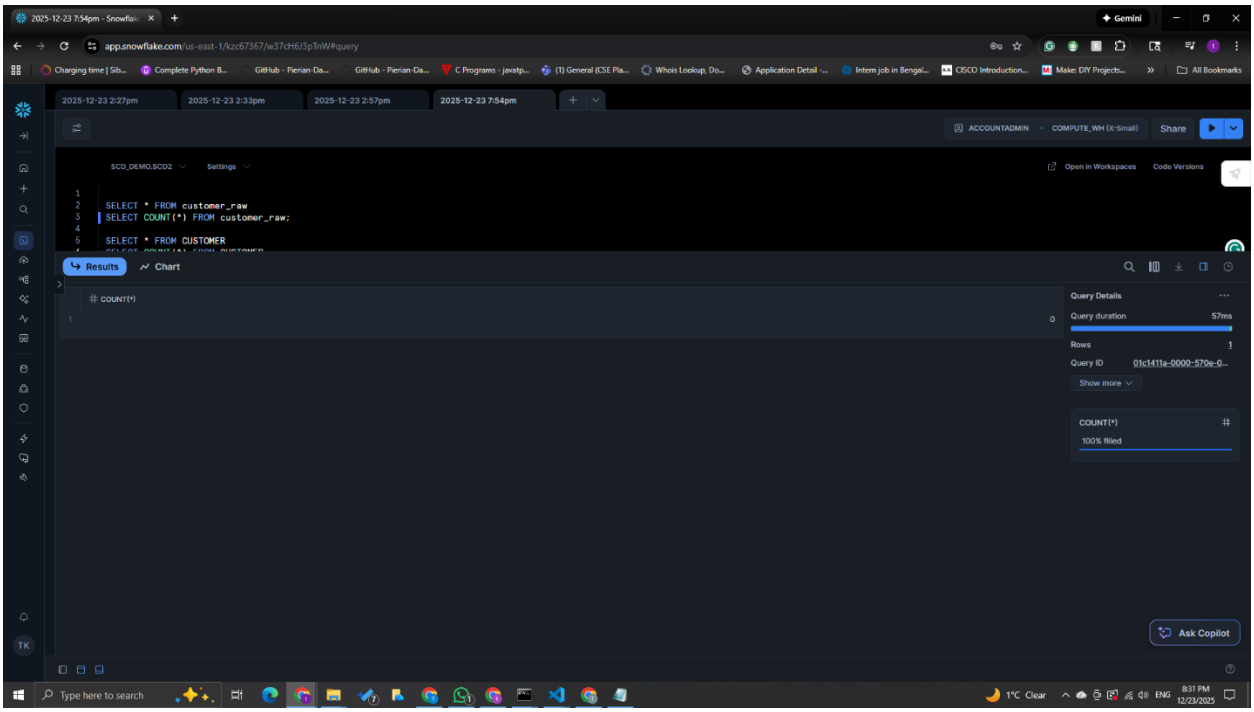
7. Snowflake Ingestion using Snowpipe

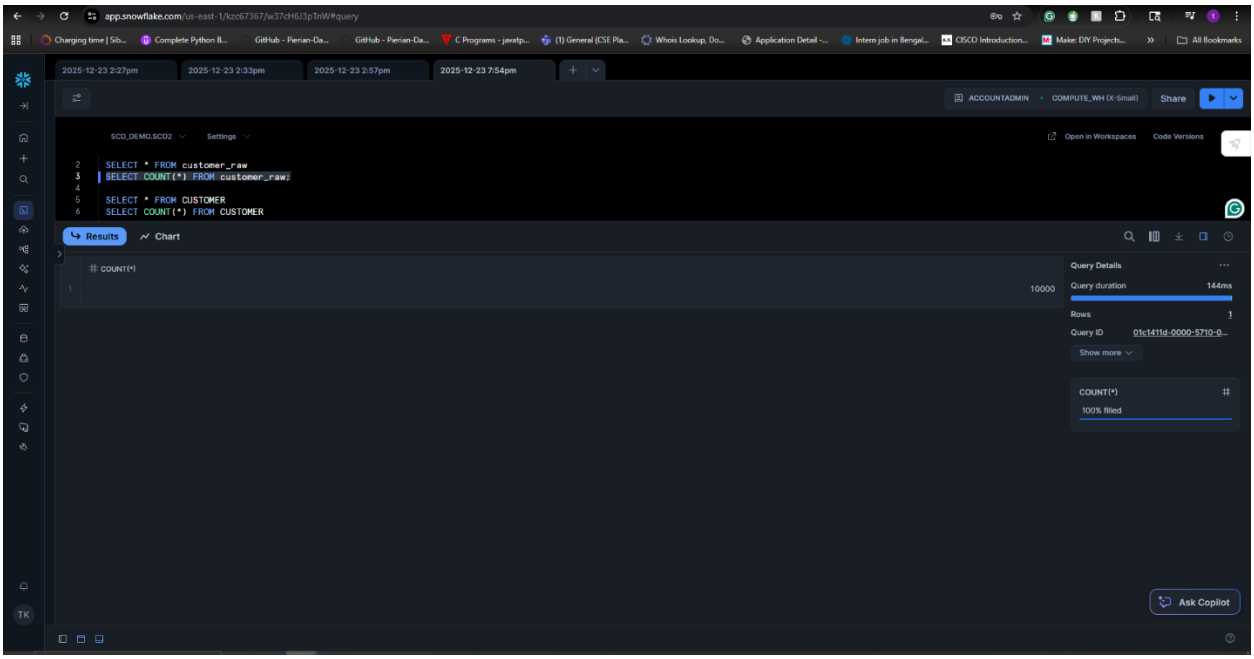
Snowpipe Functionality

- Auto-ingest enabled
- Triggered by S3 event notifications
- Loads new files automatically into staging table

Staging Table

```
CREATE OR REPLACE TABLE customer_raw (

customer_id NUMBER,

first_name VARCHAR,

last_name VARCHAR,

email VARCHAR,

street VARCHAR,

city VARCHAR,

state VARCHAR,

country VARCHAR

);
```





8. Data Modeling (SCD Type 1)

Target Table

CREATE OR REPLACE TABLE customer (

customer_id NUMBER,

first_name VARCHAR,

last_name VARCHAR,

email VARCHAR,

street VARCHAR,

city VARCHAR,

state VARCHAR,

country VARCHAR,

update_timestamp TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP()

);

SCD Type 1 Explanation

- Only latest version of a customer is retained
- Updates overwrite existing values
- No historical versions are stored

```
SELECT * FROM CUSTOMER
SELECT COUNT(*) FROM CUSTOMER
```

Results | Chart

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	STREET	CITY	STATE	COUNTRY	UPDATE_TIMESTAMP
1	0	Cassidy	Wolfe	jennifer59@example.net	48938 Omar Prairie	New Rhonda	New York	Algeria	2025-12-23 17:34:09.279
2	1	Becky	Jones	jestes@example.net	913 Elizabeth Shoals	Knightchester	Indiana	Anguilla	2025-12-23 17:34:09.279
3	2	Diane	Hughes	floresshawn@example.org	4970 Harmon Summit	Fletcherstad	Montana	Nauru	2025-12-23 17:34:09.279
4	3	Tara	Rangel	usmith@example.org	11524 Wilson Mill Apt. 513	North Daniel	West Virginia	Serbia	2025-12-23 17:34:09.279
5	4	Jonathon	Dawson	corr@example.com	278 Kimberly Corner Apt. 93	Navarroland	Rhode Island	Portugal	2025-12-23 17:34:09.279
6	5	Kathleen	Davidson	tammy13@example.net	5055 Dickson Mission	Thompsonmouth	Hawaii	Guernsey	2025-12-23 17:34:09.279
7	6	Jane	Long	tevans@example.org	92256 Sharon Locks	New Andrew	Mississippi	Central African Republ	2025-12-23 17:34:09.279
8	7	Robin	Willis	stoneedward@example.com	62843 Autumn Avenue	New Amanda	Idaho	Jordan	2025-12-23 17:34:09.279
9	8	Susan	Kelly	gcole@example.net	9233 Thomas Manor	Brendastad	Oregon	Marshall Islands	2025-12-23 17:34:09.279
10	9	James	Brown	johnkaufman@example.org	45930 Matthew Locks Apt. 2	North Sueville	Arizona	French Guiana	2025-12-23 17:34:09.279
11	10	Craig	Chavez	susancraig@example.net	96587 Reed Center Apt. 987	Jesseberg	Georgia	Mongolia	2025-12-23 17:34:09.279
12	11	Amanda	Moore	franklinjohn@example.org	213 Christian Fields Suite 73	Port Elizabeth	Hawaii	Belarus	2025-12-23 17:34:09.279
13	12	Melissa	Cruz	daniel90@example.org	903 Michelle Ferry Suite 401	Bethstad	Arkansas	United States of Amer	2025-12-23 17:34:09.279
14	13	Cory	Wilson	nicholaslee@example.net	3172 Kemp Ferry Suite 406	Hansenmouth	Louisiana	Bouvet Island (Bouvet	2025-12-23 17:34:09.279
15	14	Sandra	Stewart	valdezcrystal@example.net	271 Cook Park Apt. 593	Blakestad	New York	North Macedonia	2025-12-23 17:34:09.279
16	15	Oscar	Rodriguez	davidsmith@example.com	9192 Zachary Ridges Apt. 19	Port Glen	Iowa	Brazil	2025-12-23 17:34:09.279
17	16	Diane	Powell	liukelly@example.net	82288 Benjamin Mountains A	Jackieport	Nevada	Russian Federation	2025-12-23 17:34:09.279
18	17	Angela	Powers	danielwhite@example.org	806 Ferguson Lakes	Karachester	Wisconsin	Benin	2025-12-23 17:34:09.279
19	18	Amy	Sanchez	melissarodriguez@example.net	6806 Catherine Mills	Lake Johnshire	Texas	Barbados	2025-12-23 17:34:09.279

9. **Merge Logic (SCD Type 1)**

```
MERGE INTO customer c

USING customer_raw cr

ON c.customer_id = cr.customer_id

WHEN MATCHED AND (

c.first_name <> cr.first_name OR

c.last_name <> cr.last_name OR

c.email <> cr.email OR

c.street <> cr.street OR

c.city <> cr.city OR

c.state <> cr.state OR

c.country <> cr.country
```

)

THEN UPDATE SET

c.first_name = cr.first_name,

c.last_name = cr.last_name,

c.email = cr.email,

c.street = cr.street,

c.city = cr.city,

c.state = cr.state,

c.country = cr.country,

c.update_timestamp = CURRENT_TIMESTAMP()

WHEN NOT MATCHED THEN INSERT (

customer_id, first_name, last_name, email,

street, city, state, country

)

VALUES (

cr.customer_id, cr.first_name, cr.last_name,

cr.email, cr.street, cr.city, cr.state, cr.country

);

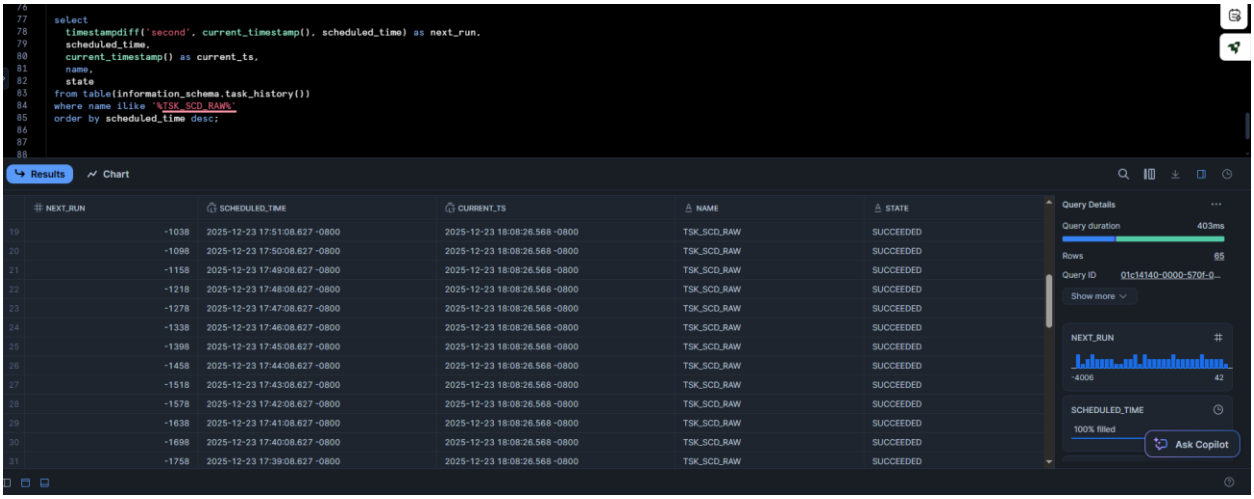
10. Automation with Stored Procedure and Task

- Stored Procedure Encapsulates merge logic and truncates staging table after processing.

Snowflake Task

- Runs every 1 minute
- Enables near real-time micro-batch processing

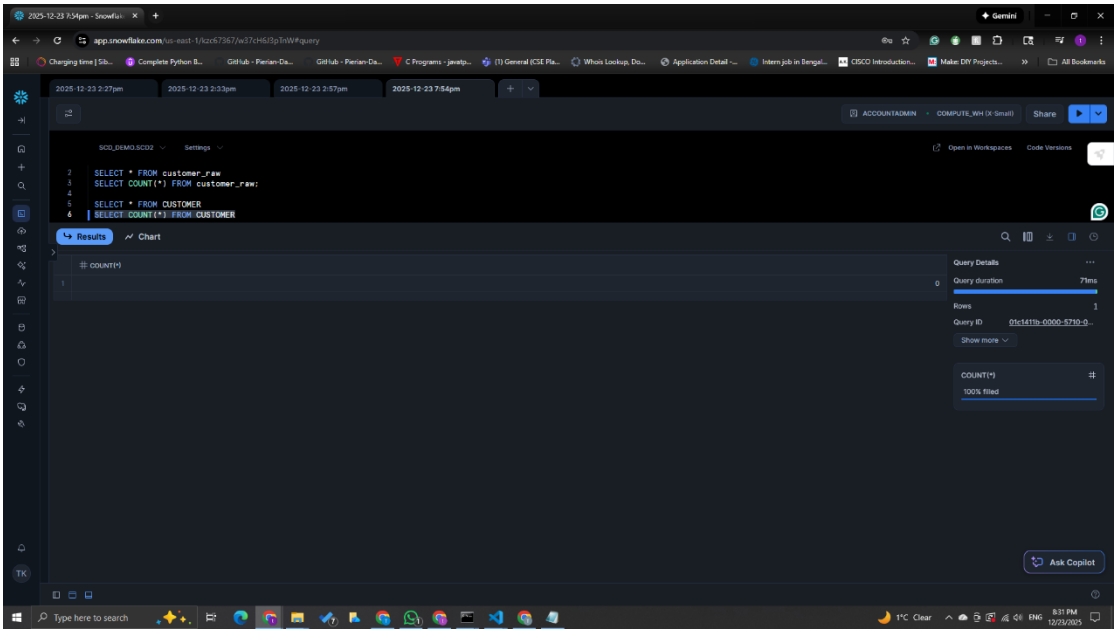
```
CREATE OR REPLACE TASK tsk_scd_raw
WAREHOUSE = COMPUTE_WH
 SCHEDULE = '1 minute'
 AS
 CALL pdr_scd_demo();
```

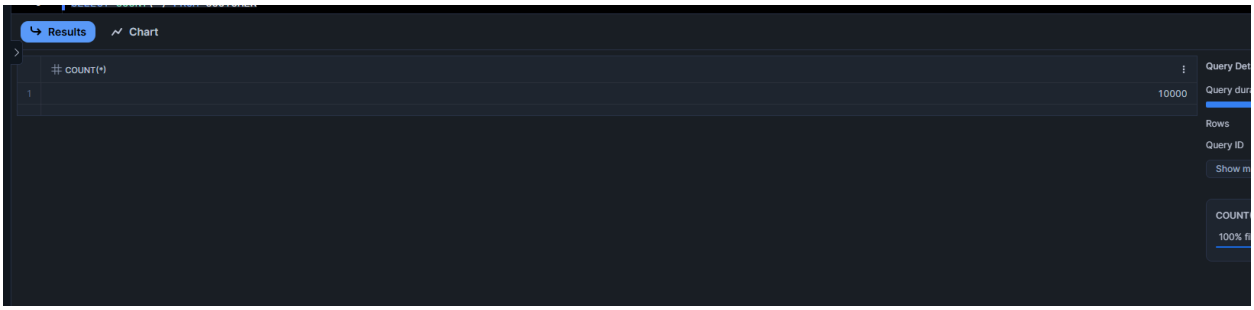


11. Validation and Results

Observations

- **Before Faker execution:**
    - **customer_raw = 0 rows**
    - **customer = 0 rows**
- **After Faker + NiFi + Snowpipe:**
    - **customer_raw = 10,000 rows**
    - **customer = 10,000 rows**





12. Conclusion

This project successfully demonstrates:

- End-to-end real-time ingestion using AWS and Snowflake
- Event-driven ingestion with Snowpipe
- Micro-batch processing using Snowflake Tasks
- Correct implementation of SCD Type 1
- Production-style orchestration using NiFi and Docker

The design is extensible and can be enhanced to full SCD Type 2 by leveraging history tables and streams.

13. Future Enhancements

- Implement full SCD Type 2 with customer_history
- Add data quality checks
- Integrate monitoring and alerting
- Parameterize record volume and schedules
- Visualize metrics using dashboards