

SNOWFLAKE ASSIGNMENT

Name: Polireddi Govind

Employee Id: AS1552

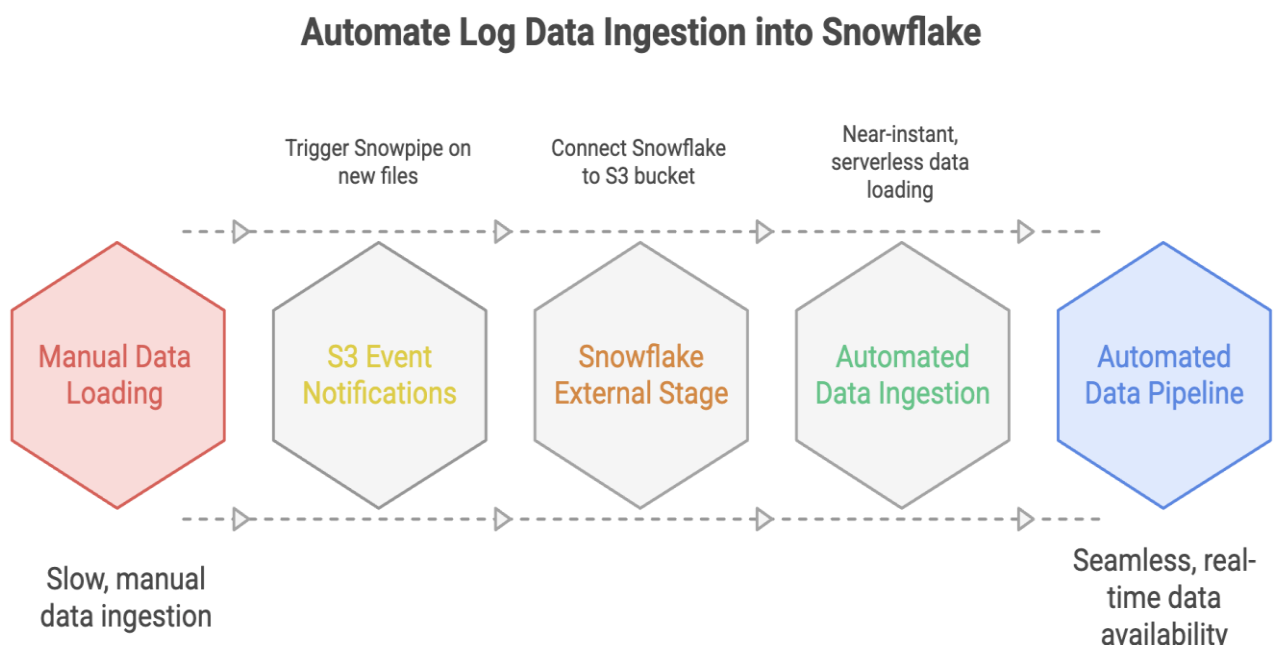
Problem Statement 1:

Automate Real-time Log Data Ingestion from AWS S3 into Snowflake using Snowpipe and Monitor via Snowsight

Topics Covered: Snowpipe, Internal/External Stage, Copy Options, Streaming, SnowSQL, Performance Optimization, Virtual Warehouse, Caching, Monitoring (Snowsight), AWS S3 Integration, Time Travel, Clustering, Data Sampling

Project Overview:

This project aims to automate real-time log data ingestion from AWS S3 into Snowflake using Snowpipe, establishing a seamless, serverless, and near-instant data loading pipeline. By connecting Snowflake with S3 event notifications, newly uploaded log files are automatically ingested, processed, and made available for analytics with minimal manual effort. The solution focuses on performance optimization, cost efficiency, and operational visibility, leveraging Snowflake's capabilities such as external/internal stages, copy options, virtual warehouses, clustering, caching, time travel, and Snowsight monitoring.

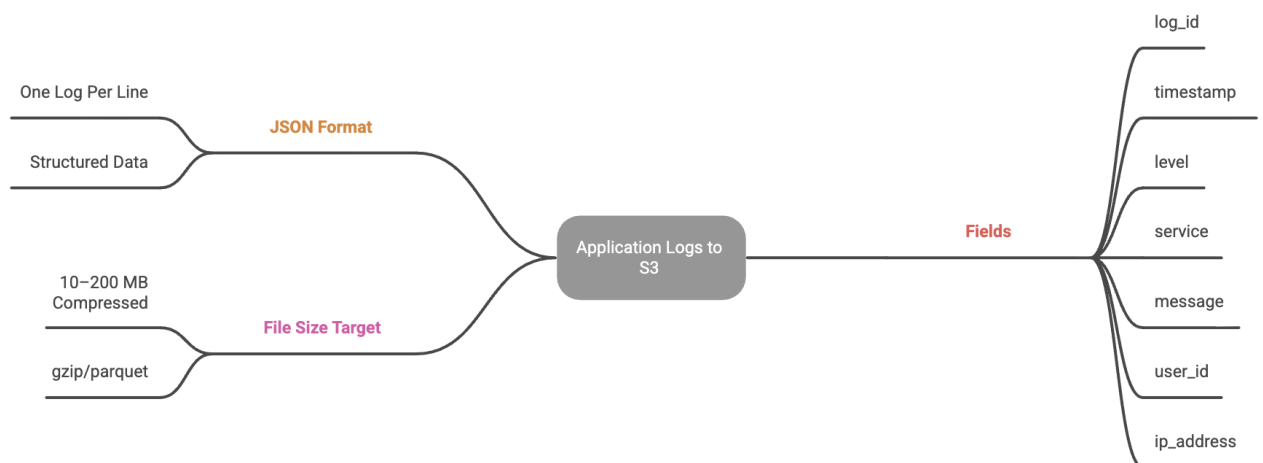


Architecture Overview

1. Log Producers → AWS S3

- Application logs (e.g., auth, payments, orders, shipping) are written to S3 in **JSON format**, one log per line.
- Each JSON contains fields like:
 - log_id, timestamp, level, service, message, user_id, ip_address.
- File size target: **10–200 MB compressed (parquet)** for efficient Snowpipe ingestion.

Application Logs to S3: Structure and Format



2. S3 Event Notifications → Snowflake Snowpipe

- Each new file triggers an **S3 event notification (via SNS/SQS)**.
- Snowpipe automatically loads data into Snowflake without manual intervention.

3. Snowflake Staging and Curation

- **External Stage:** Secure pointer to the S3 bucket/prefix.
- **Snowpipe:** Automatically ingests JSON logs into a **raw staging table**.
- **Curation Layer:** Transforms, normalizes, and deduplicates data into curated analytics tables.

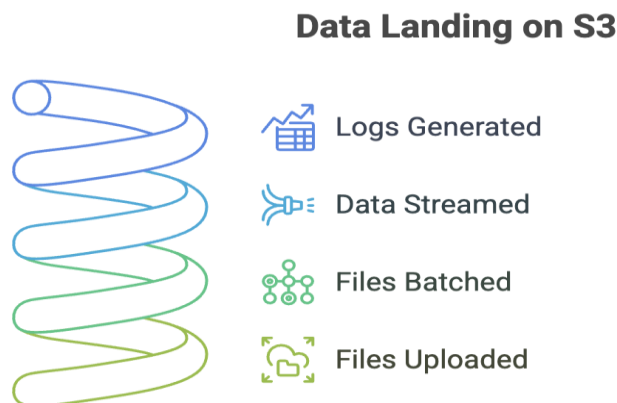
4. Monitoring and Optimization

- **Snowsight:** Monitors ingestion latency, file load history, and error counts.
- **Virtual Warehouses:** Handle downstream transformations.
- **Clustering & Time Travel:** Improve query speed and allow recovery from data issues.

Step-by-Step Approach for Implementation

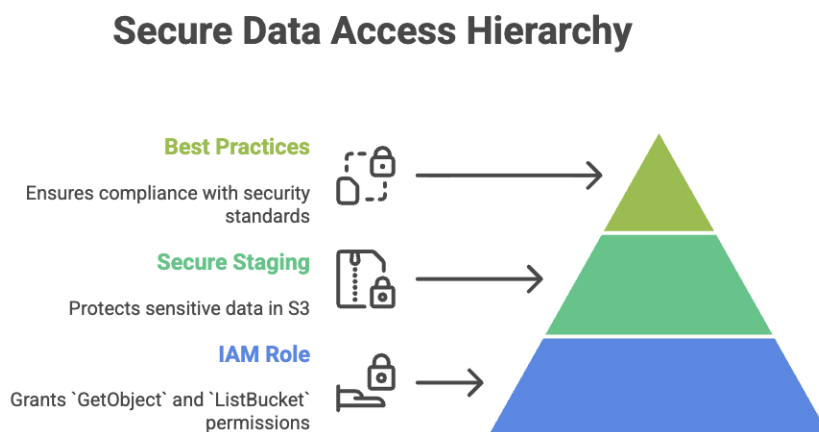
Step 1: Data Landing on S3

- Logs generated by different services are streamed to the S3 bucket.
- Files are batched and upload to the S3 bucket.



Step 2: Setting Up the Snowflake Stage

- Create an **AWS IAM role** with GetObject and ListBucket permissions.
- Stage points securely to the S3 log location.



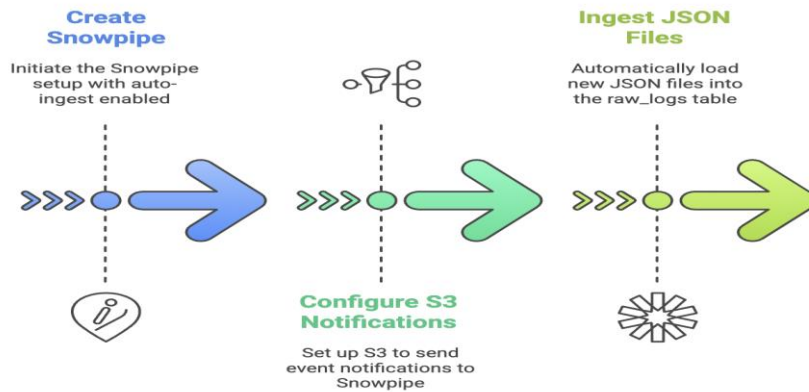
Step 3: Defining File Format and Copy Options

- Define a **JSON file format** to handle the log data structure:
- COPY options:
 - Use `ON_ERROR='CONTINUE'` to skip corrupted records.
 - Enable `PURGE=FALSE` for reprocessing flexibility.

Step 4: Automating Ingestion with Snowpipe

- Create a **Snowpipe** with `AUTO_INGEST = TRUE`:
- Configure S3 event notifications to publish to the Snowflake pipe endpoint.
- Each new JSON file is ingested into the `raw_logs` table.

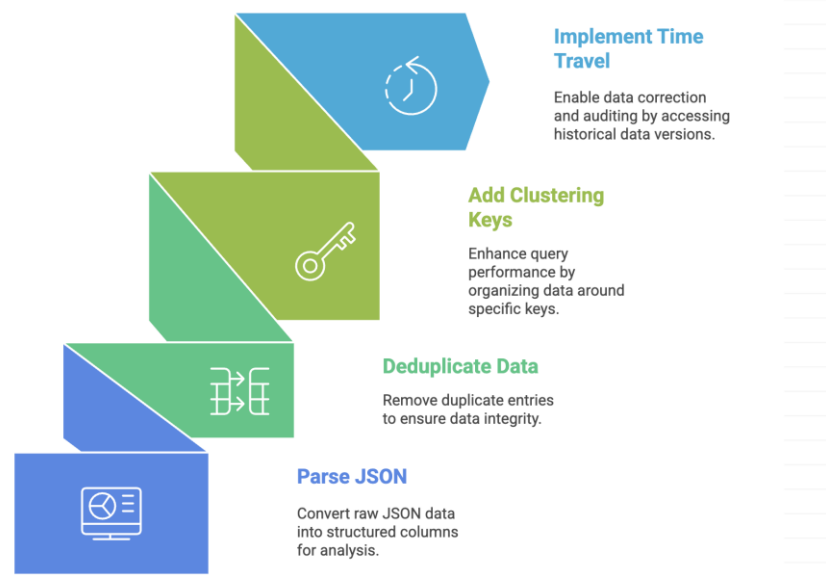
Snowpipe Auto-Ingestion Process



Step 5: Transforming and Curating Data

- Parse JSON data into structured columns:
- Apply **deduplication** using `DISTINCT` or a **merge on log_id**.
- Add **clustering keys**: (`event_time`, `service_name`).
- Use **Time Travel** for data correction and auditing.

Data Curation Process

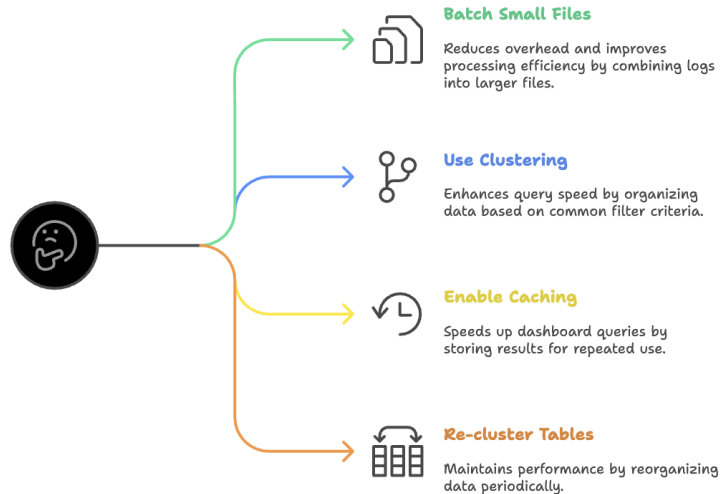


Step 6: Performance Optimization

- Avoid small files — batch logs into optimal file sizes.
- Use **clustering** and **search optimization** for fast filtering (e.g., by `service_name` or `log_level`).

- Enable **caching** for repeated dashboard queries.
- Periodically **re-cluster** curated tables to maintain performance.

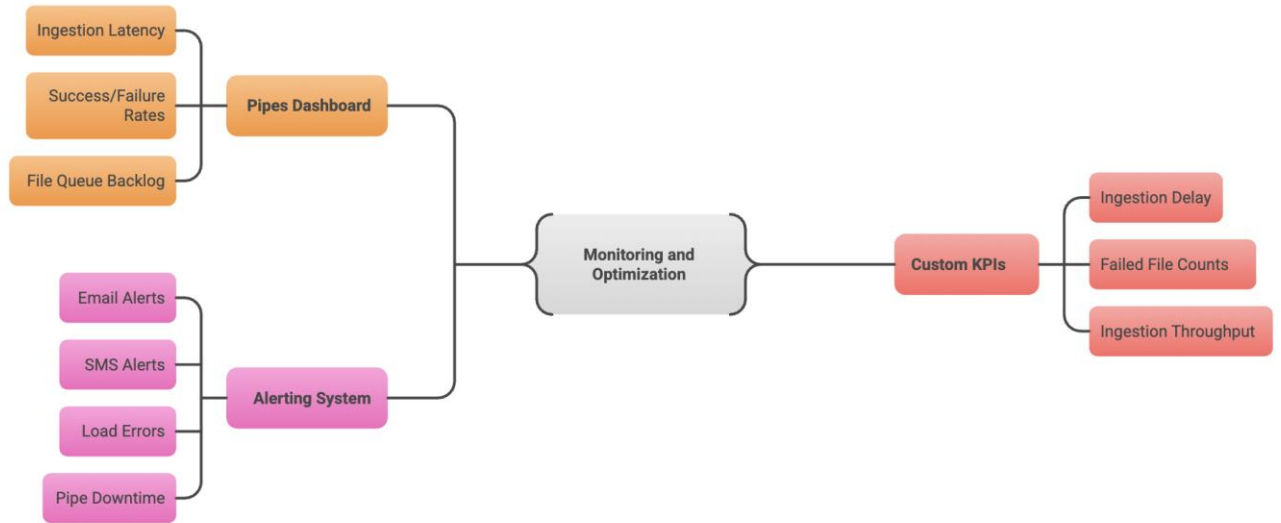
How to optimize Snowflake performance?



Step 7: Monitoring with Snowsight

- Use the **Pipes Dashboard** to track:
 - Ingestion latency
 - Success/failure rates
 - File queue backlog
- Set up **custom KPIs**:
 - Ingestion delay = `CURRENT_TIMESTAMP - file_last_modified`
 - Failed file counts, ingestion throughput
- Configure **email/SMS alerts** for load errors or pipe downtime.

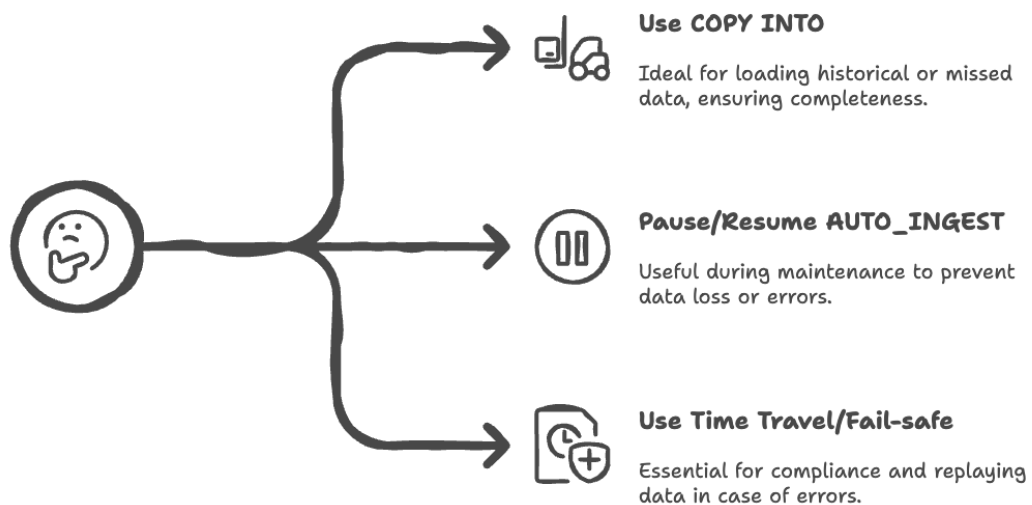
Monitoring and Optimization Strategies



Step 8: Backfills and Reprocessing

- Use **COPY INTO** for historical or missed data loads:
- Pause/resume `AUTO_INGEST` during maintenance.
- Use **Time Travel** and **Fail-safe** for replay or compliance scenarios.

How to manage data ingestion and compliance?



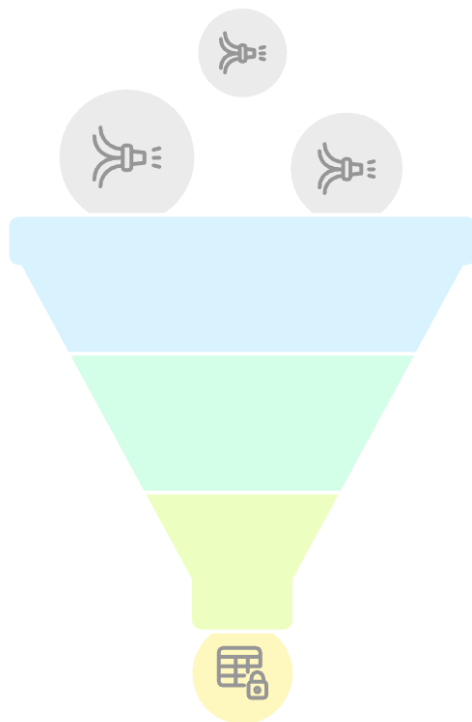
Step 9: Advanced Capabilities

- **Snowpipe Streaming:** For sub-second ingestion directly from producers (optional).
- **Data Sampling:** Use `SAMPLE` or partition filters for debugging.

- **Cost Control:**

- Use Snowpipe for ingestion (Snowflake-managed compute).
- Use auto-suspend warehouses for transformation to minimize compute cost.

Optimizing Data Ingestion and Transformation



Snowpipe Streaming

Direct sub-second data ingestion



Data Sampling

Debugging with sample data

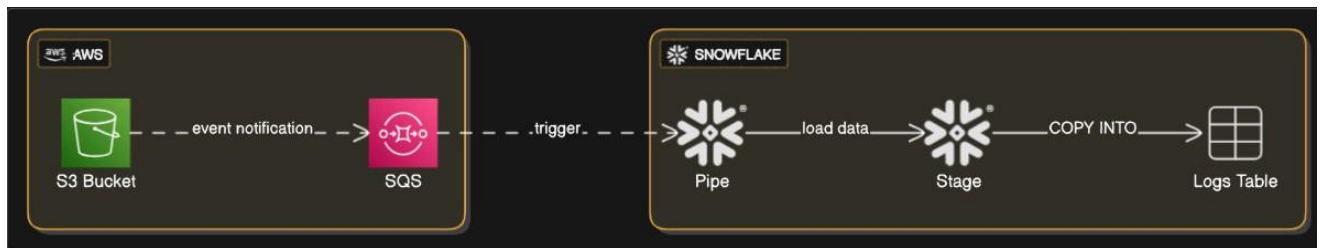


Cost Control

Efficient resource management

ARCHITECTURE:

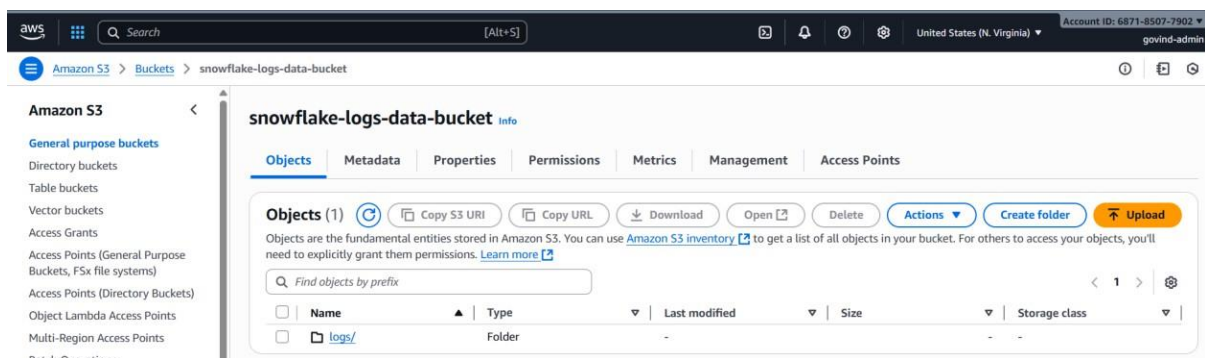
S3 → SQS → Snowpipe → Snowflake Stage → Copy Into Logs Table (Snowflake)



IMPLEMENTATION

S3 BUCKET:

- ✓ Created a S3 bucket “snowflake-logs-data-bucket”

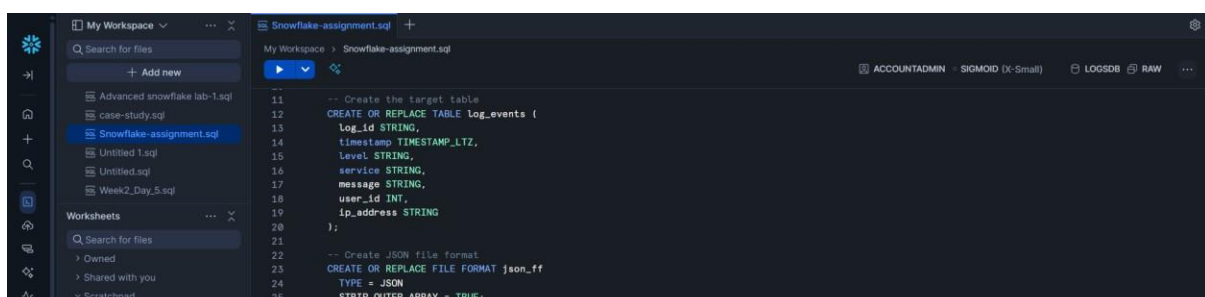


SNOWFAKE WAREHOUSE:

- ✓ Created a Warehouse (Sigmoid) , Database(logsdb) and Schema (RAW).

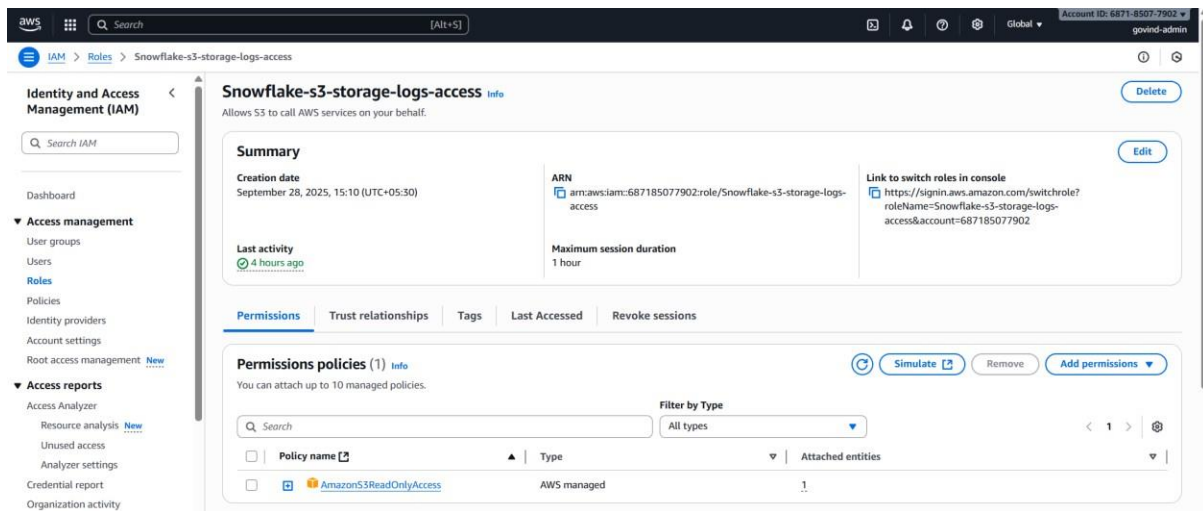


- ✓ Created a log_events Table to store the logs from S3.
- ✓ Create a JSON file format.



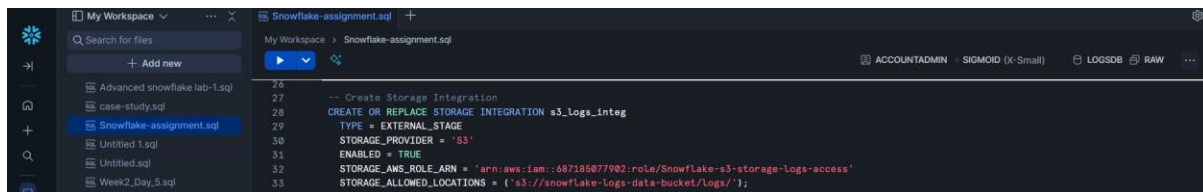
S3 READ POLICY(Role):

- ✓ Create a new role in IAM for S3 read access and assign to the Snowflake.



S3 and SNOWFLAKE INTEGRATION:

- ✓ Integrate S3 and Snowflake, assign the role and the location of the bucket.



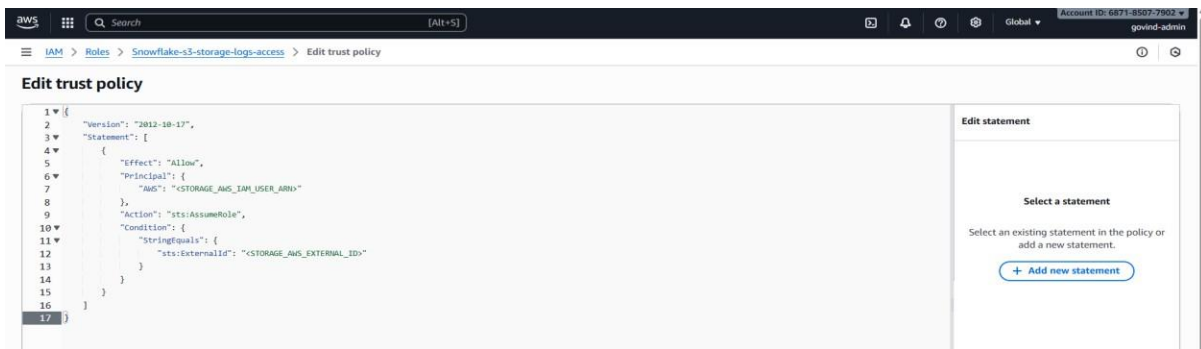
- ✓ Describe the created integration.



- ✓ Copy the ARN and External_ID from the OUTPUT.

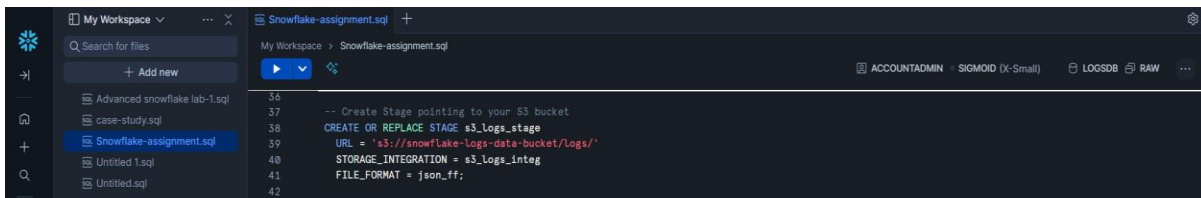
	property	property_type	property_value	property_default
1	ENABLED	Boolean	true	false
2	STORAGE_PROVIDER	String	S3	
3	STORAGE_ALLOWED_LOCATIONS	List	s3://snowflake-logs-data-bucket/logs/	[]
4	STORAGE_BLOCKED_LOCATIONS	List		[]
5	STORAGE_AWS_IAM_USER_ARN	String	arn:aws:iam::974916068036:user/externalstages/ci9s7c0000	
6	STORAGE_AWS_ROLE_ARN	String	arn:aws:iam::687185077902:role/Snowflake-s3-storage-logs-access	
7	STORAGE_AWS_EXTERNAL_ID	String	JK33786_SFCRole=2_tEy1K8FkFUGRUlftzOptfZRDtQ0=	
8	USE_PRIVATELINK_ENDPOINT	Boolean	false	false
9	COMMENT	String		

- ✓ Update the Policy with ARN and External_ID in the Role policy.

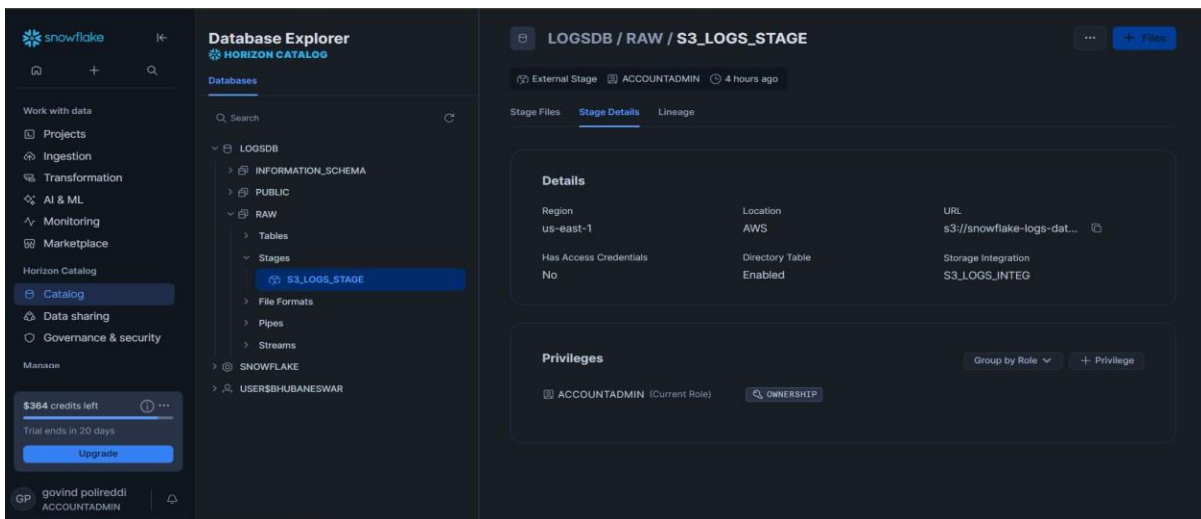


STAGE CREATION:

- ✓ Create a stage to point it to the S3 bucket to fetch the Json files.



- ✓ Successfully created a stage.

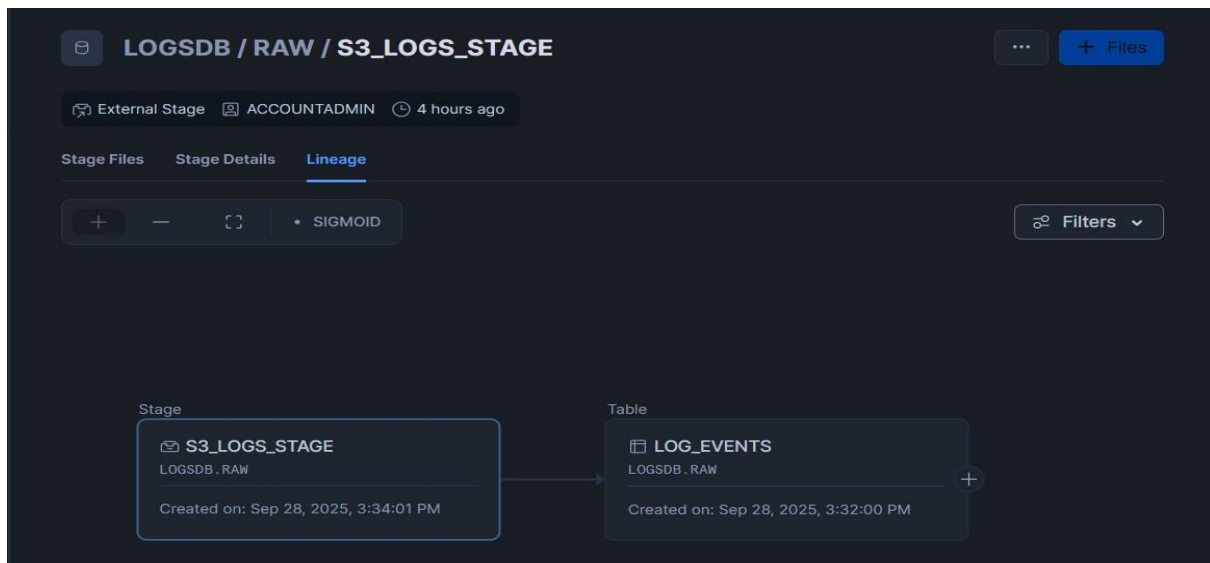


SNOWPIPE:

- ✓ Create a Snowpipe to ingest the data continuously into the logs table.



- ✓ Lineage graph is created from the Stage to the Table



SQS EVENT NOTIFICATION:

- ✓ Create SQS Event notification ,when there is a new file added into the S3 bucket it gets trigger the Snowpipe.

The screenshot shows the 'Create event notification' form in the AWS IAM console. The form is titled 'Create event notification' and includes a sub-header 'General configuration'. The 'Event name' field is set to 'snowflake-logs-pipeline'. The 'Prefix - optional' field is set to 'logs/'. The 'Suffix - optional' field is set to '.jpg'. The 'Event types' section is expanded, showing 'Object creation' with the following options: 'All object create events' (s3:ObjectCreated:*), 'Put' (s3:ObjectCreated:Put), and 'Delete' (s3:ObjectDeleted:Delete). The 'Destination' field is set to 'arn:aws:sqs:us-east-1:974916068036:sf-snowpipe-AIDA6F7MGULCITXZYKBFX-yWhpMEZ5AtEdvhlDjyw1g'.

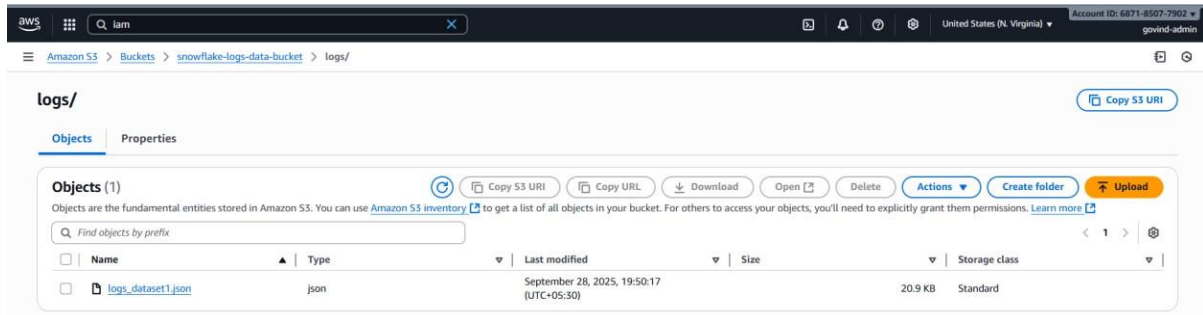
- ✓ Successfully created the SQS notification.

The screenshot shows the 'Event notifications' table in the AWS IAM console. The table has columns for 'Name', 'Event types', 'Filters', 'Destination type', and 'Destination'. The table contains one row with the name 'snowpipe', event types 'All object create events', filters 'logs/', destination type 'SQS queue', and destination 'arn:aws:sqs:us-east-1:974916068036:sf-snowpipe-AIDA6F7MGULCITXZYKBFX-yWhpMEZ5AtEdvhlDjyw1g'. Below the table, there is a section for 'Amazon EventBridge' with a toggle switch set to 'Off'.

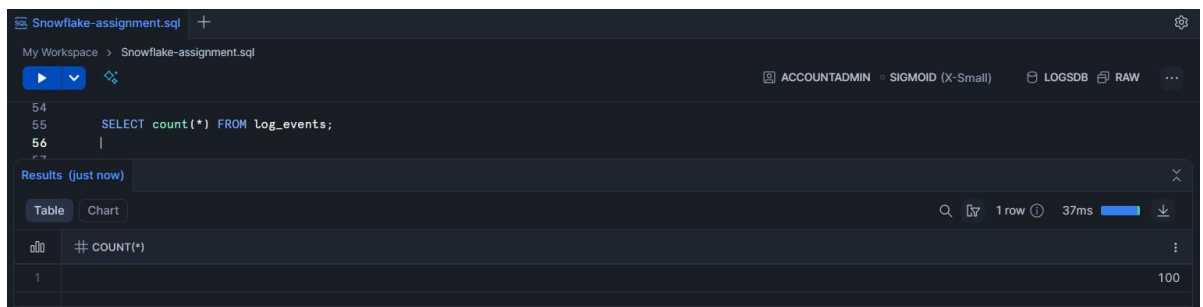
Name	Event types	Filters	Destination type	Destination
snowpipe	All object create events	logs/	SQS queue	arn:aws:sqs:us-east-1:974916068036:sf-snowpipe-AIDA6F7MGULCITXZYKBFX-yWhpMEZ5AtEdvhlDjyw1g

FILE UPLOAD:

- ✓ Uploaded a “logs_dataset1.json” file to the S3 bucket.

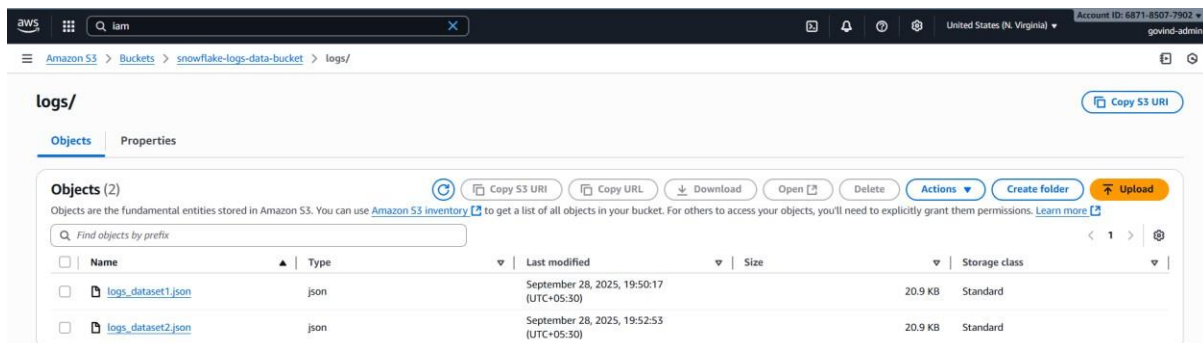


- ✓ Successfully loaded the logs data into the Table (100 Rows)

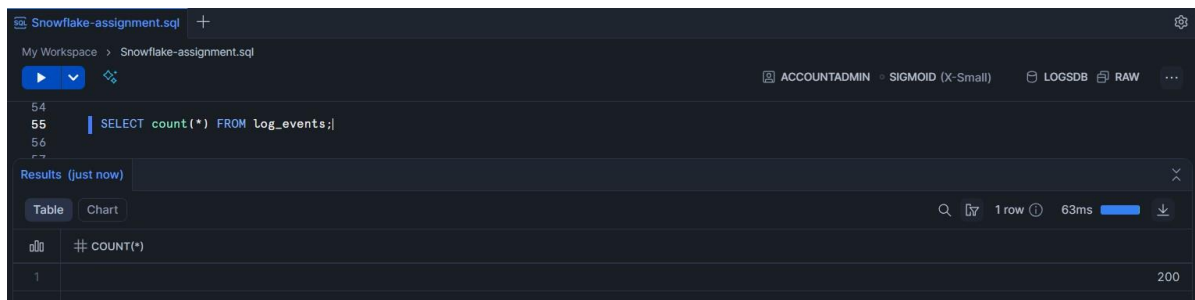


	LOG_ID	TIMESTAMP	LEVEL	SERVICE	MESSAGE	USER_ID	IP_ADDRESS
1	587ca500-b7c5-4c48-a611-f9d99f03368e	2025-09-28 07:55:20.993 -0700	INFO	auth	Invalid credentials	2003	114.71.52.44
2	62525b58-6883-45d3-a9c6-4424ece7f6de	2025-09-28 08:02:01.993 -0700	DEBUG	auth	User login successful	1383	111.119.13.101
3	f39a1fa3-a046-4aa8-bf22-4ccb2f9d5dd2	2025-09-28 07:44:53.993 -0700	DEBUG	payments	Low stock warning	3413	142.3.81.216
4	8f0ff445-af10-40bd-b30c-f75ee15d6914	2025-09-28 08:36:11.993 -0700	ERROR	payments	Inventory updated	4920	172.52.47.194
5	726a24df-4373-445d-8230-1f0929fb79cd	2025-09-28 09:09:26.993 -0700	ERROR	orders	Address validation failed	2083	22.235.63.193
6	930d7160-bdfe-43b7-92ac-ae84d558ee0b	2025-09-28 09:11:53.993 -0700	ERROR	shipping	Payment declined	3364	98.35.23.116
7	b6c0aae3-c787-43be-9709-9a7b1629313f	2025-09-28 07:37:06.993 -0700	ERROR	auth	Inventory updated	4549	51.194.142.232
8	4cb80cdc-a39d-4c07-92a7-023c1e9281b0	2025-09-28 07:55:51.993 -0700	ERROR	payments	Payment declined	2455	107.136.36.87
9	5e92ab64-1584-41da-ba27-64e2e0b9316f	2025-09-28 08:09:43.993 -0700	WARN	payments	Low stock warning	2554	138.112.166.28
10	12f47a31-a833-427b-98eb-4ae48b59c02f	2025-09-28 08:51:22.993 -0700	INFO	orders	Order failed	2096	33.108.161.108
11	e2965c6a-0df9-48ef-9dc7-70c0c716f777	2025-09-28 07:53:09.993 -0700	DEBUG	inventory	Low stock warning	1585	135.71.126.134
12	1389f297-bb12-4eeb-a598-583edab7ce70	2025-09-28 07:40:39.993 -0700	DEBUG	shipping	Order failed	2482	112.70.252.46
13	c1832fb8-1202-4596-b5d4-1002fcc382bb	2025-09-28 07:39:27.993 -0700	INFO	auth	Order placed	3570	81.216.32.197
14	3119372f-f6d8-471d-a1c4-7150f0202a90	2025-09-28 08:30:32.993 -0700	DEBUG	shipping	Invalid credentials	4977	5.58.136.174
15	f56e8c36-a070-4168-ac2d-b739ad8bd0b0	2025-09-28 09:07:25.993 -0700	ERROR	inventory	Order placed	2858	1.134.91.54

- ✓ Uploaded a “logs_dataset2.json” file to the S3 bucket.



- ✓ Successfully loaded the logs data into the Table (200 Rows)



```
54
55 SELECT count(*) FROM log_events;
56
```

Results (just now)

Table	Chart	1 row	63ms	Download
# COUNT(*)		200		

Results (just now)

TableChart

200 rows140ms

	LOG_ID	TIMESTAMP	LEVEL	SERVICE	MESSAGE	USER_ID	IP_ADDRESS
1	587ca500-b7c5-4c48-a611-f9d99f03368e	2025-09-28 07:55:20.993 -0700	INFO	auth	Invalid credentials	2003	114.71.52.44
2	62525b58-6883-45d3-a9c6-4424ece7f6de	2025-09-28 08:02:01.993 -0700	DEBUG	auth	User login successful	1383	111.119.13.101
3	f39a1fa3-a046-4aa8-bf22-4ccb2f9d5dd2	2025-09-28 07:44:53.993 -0700	DEBUG	payments	Low stock warning	3413	142.3.81.216
4	8f0ff445-af10-40bd-b30c-f75ee15d6914	2025-09-28 08:36:11.993 -0700	ERROR	payments	Inventory updated	4920	172.52.47.194
5	726a24df-4373-445d-8230-1f0929fb79cd	2025-09-28 09:09:26.993 -0700	ERROR	orders	Address validation failed	2083	22.235.63.193
6	930d7160-bdfe-43b7-92ac-ae84d558ee0b	2025-09-28 09:11:53.993 -0700	ERROR	shipping	Payment declined	3364	98.35.23.116
7	b6c0aee3-c787-43be-9709-9a7b1629313f	2025-09-28 07:37:06.993 -0700	ERROR	auth	Inventory updated	4549	51.194.142.232
8	4cb80cdc-a39d-4c07-92a7-023c1e928160	2025-09-28 07:55:51.993 -0700	ERROR	payments	Payment declined	2455	107.136.36.87
9	5e92ab64-1584-41da-ba27-64e2e0b9316e	2025-09-28 08:09:43.993 -0700	WARN	payments	Low stock warning	2554	138.112.166.28
10	12f47a31-a833-427b-98eb-4ae48b59c02f	2025-09-28 08:51:22.993 -0700	INFO	orders	Order failed	2096	33.108.161.108
11	e2965c6a-0df9-48ef-9dc7-70c0c716f777	2025-09-28 07:53:09.993 -0700	DEBUG	inventory	Low stock warning	1585	135.71.126.134
12	1389f297-bb12-4eeb-a598-583edab7ce70	2025-09-28 07:40:39.993 -0700	DEBUG	shipping	Order failed	2482	112.70.252.46
13	c1832fb8-1202-4596-b5d4-1002fcc382bb	2025-09-28 07:39:27.993 -0700	INFO	auth	Order placed	3570	81.216.32.197
14	3119372f-f6d8-471d-a1c4-7150f0202a90	2025-09-28 08:30:32.993 -0700	DEBUG	shipping	Invalid credentials	4977	5.58.136.174
15	f56e8c36-a070-4168-ac2d-b739ad8bd0b0	2025-09-28 09:07:25.993 -0700	ERROR	inventory	Order placed	2858	1.134.91.54
16	1495f34c-c1fa-4657-adce-93f6ed4671b2	2025-09-28 07:23:47.993 -0700	ERROR	shipping	Address validation failed	1814	78.191.82.0

TIME TRAVEL:

- ✓ Retention period is 7 days.



```
57
58 -- Enable Time Travel on the table (7 days retention)
59 ALTER TABLE log_events
60 SET DATA_RETENTION_TIME_IN_DAYS = 7;
61
62 SELECT * FROM log_events;
63
64 SELECT * FROM log_events BEFORE (OFFSET => -400);
65
```

- ✓ Table logs data is of 200 Rows.(Present)

Results (just now)

TableChart

200 rows106ms

	LOG_ID	TIMESTAMP	LEVEL	SERVICE	MESSAGE	USER_ID	IP_ADDRESS
1	a70a97f9-6ed9-4cf6-ab10-a11d2d03213a	2025-09-28 07:23:45.008 -0700	WARN	shipping	Address validation failed	4115	82.174.27.175
2	1485f34c-c1fa-4657-adce-93f6ed4671b2	2025-09-28 07:23:47.993 -0700	ERROR	shipping	Address validation failed	1814	78.191.82.0
3	6e12c0b0-6811-47dc-bfa7-dbc996c7201e	2025-09-28 07:24:13.008 -0700	DEBUG	orders	Address validation failed	3627	193.132.53.104

- ✓ Table logs data is of 100 Rows.(past of [400] seconds)

Results (just now)

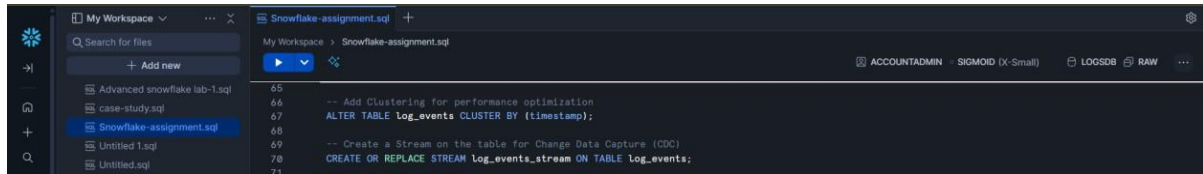
TableChart

100 rows527ms

	LOG_ID	TIMESTAMP	LEVEL	SERVICE	MESSAGE	USER_ID	IP_ADDRESS
1	587ca500-b7c5-4c48-a611-f9d99f03368e	2025-09-28 07:55:20.993 -0700	INFO	auth	Invalid credentials	2003	114.71.52.44
2	62525b58-6883-45d3-a9c6-4424ece7f6de	2025-09-28 08:02:01.993 -0700	DEBUG	auth	User login successful	1383	111.119.13.101
3	f39a1fa3-a046-4aa8-bf22-4ccb2f9d5dd2	2025-09-28 07:44:53.993 -0700	DEBUG	payments	Low stock warning	3413	142.3.81.216
4	8f0ff445-af10-40bd-b30c-f75ee15d6914	2025-09-28 08:36:11.993 -0700	ERROR	payments	Inventory updated	4920	172.52.47.194

CACHE AND QUERY OPTIMIZATION:

- ✓ Cluster the Column(timestamp) for faster optimization.



- ✓ 90ms time taken to retrieve the data as per the Query.

The screenshot shows the results of a query: `SELECT * FROM log_events WHERE YEAR(timestamp)=2025;`. The results are displayed in a table with 8 columns: LOG_ID, TIMESTAMP, LEVEL, SERVICE, MESSAGE, USER_ID, and IP_ADDRESS. The table contains 3 rows of data. The execution time is 90ms.

LOG_ID	TIMESTAMP	LEVEL	SERVICE	MESSAGE	USER_ID	IP_ADDRESS
a70a97f9-6ed9-4cf6-ab10-a11d2d03213a	2025-09-28 07:23:45.008 -0700	WARN	shipping	Address validation failed	4115	82.174.27.175
1485f34c-c1fa-4657-adce-93f6ed4671b2	2025-09-28 07:23:47.993 -0700	ERROR	shipping	Address validation failed	1814	78.191.82.0
6e12c0b0-6811-47dc-bfa7-dbc996c7201e	2025-09-28 07:24:13.008 -0700	DEBUG	orders	Address validation failed	3627	193.132.53.104

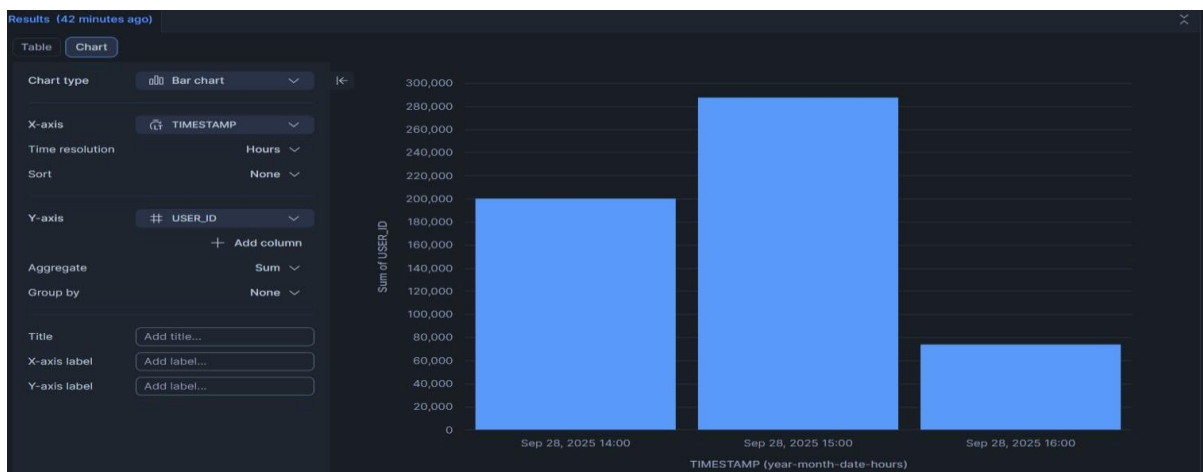
- ✓ 38ms time taken to retrieve the data as per the Query as it cached the Result(Result Cache).

The screenshot shows the results of the same query: `SELECT * FROM log_events WHERE YEAR(timestamp)=2025;`. The results are displayed in a table with 8 columns: LOG_ID, TIMESTAMP, LEVEL, SERVICE, MESSAGE, USER_ID, and IP_ADDRESS. The table contains 4 rows of data. The execution time is 38ms.

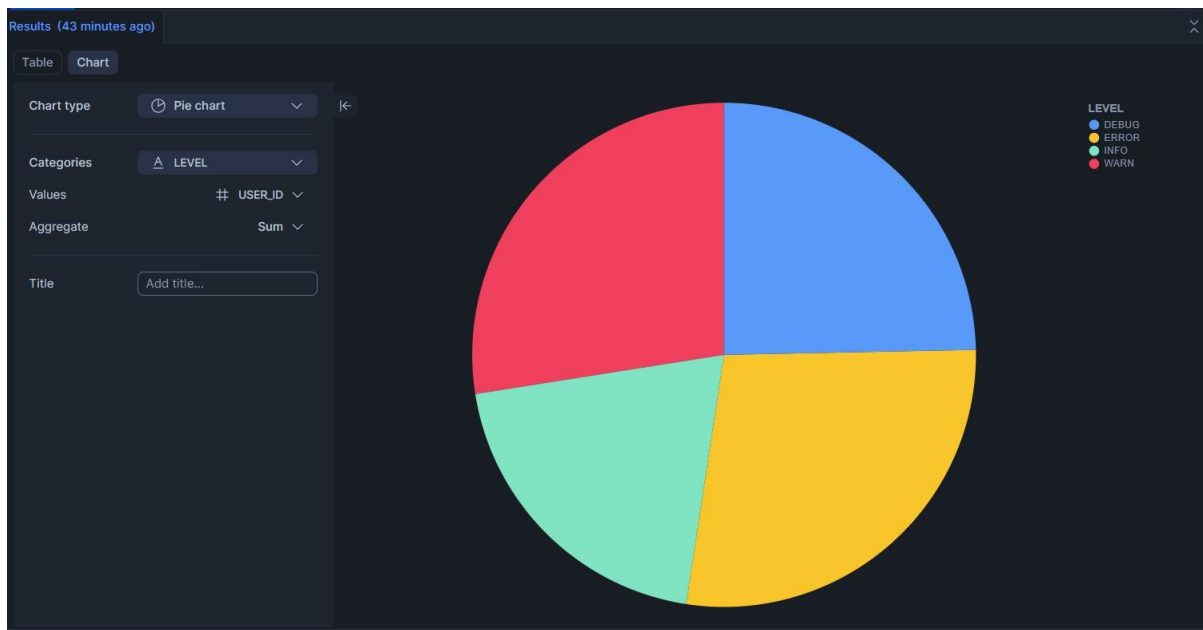
LOG_ID	TIMESTAMP	LEVEL	SERVICE	MESSAGE	USER_ID	IP_ADDRESS
a70a97f9-6ed9-4cf6-ab10-a11d2d03213a	2025-09-28 07:23:45.008 -0700	WARN	shipping	Address validation failed	4115	82.174.27.175
1485f34c-c1fa-4657-adce-93f6ed4671b2	2025-09-28 07:23:47.993 -0700	ERROR	shipping	Address validation failed	1814	78.191.82.0
6e12c0b0-6811-47dc-bfa7-dbc996c7201e	2025-09-28 07:24:13.008 -0700	DEBUG	orders	Address validation failed	3627	193.132.53.104
b83bf018-0038-47a2-a5f7-a7da9968cae5	2025-09-28 07:24:17.993 -0700	DEBUG	payments	User login successful	2065	194.167.108.232

MONITORING(SNOWSIGHT):

- ✓ Number of logs per day. 28th September got more logs.



✓ Error is having the main issue in the logs.



✓ Bar chart showing the message wise total logs.

