

SNOWFLAKE ASSIGNMENT

Name: Polireddi Govind

Employee Id: AS1552

Problem Statement 1:

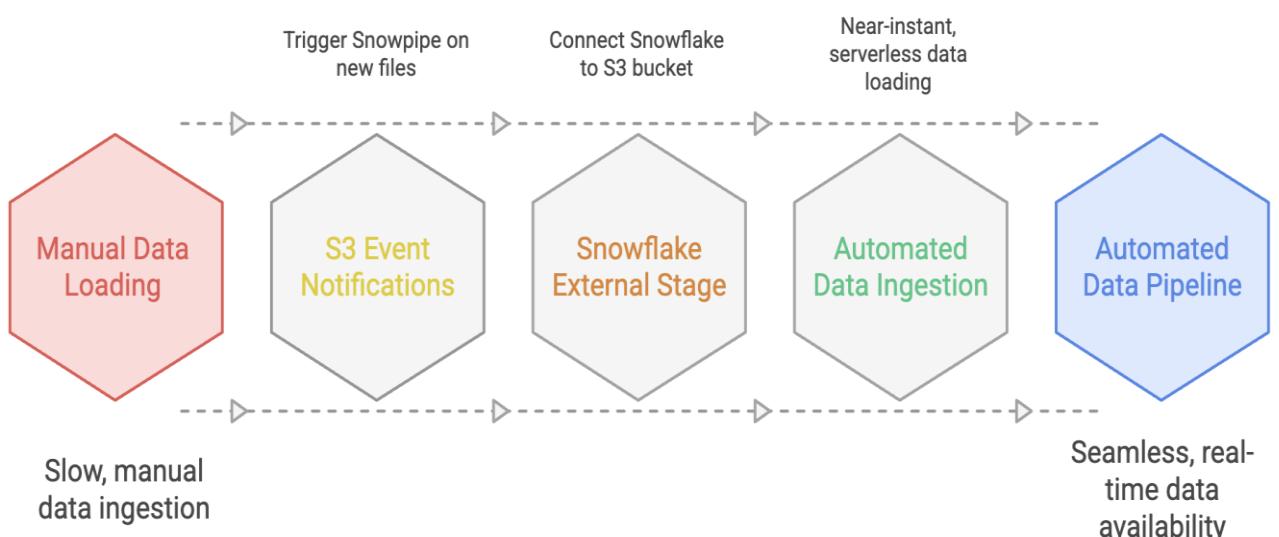
Automate Real-time Log Data Ingestion from AWS S3 into Snowflake using Snowpipe and Monitor via Snowsight

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.

Automate Log Data Ingestion into Snowflake

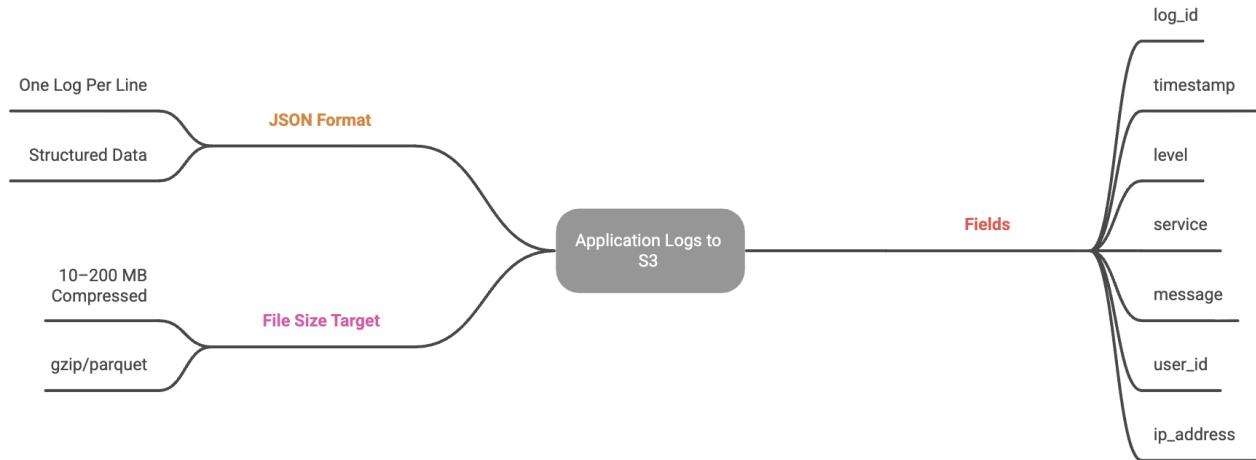


Architecture Overview

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.

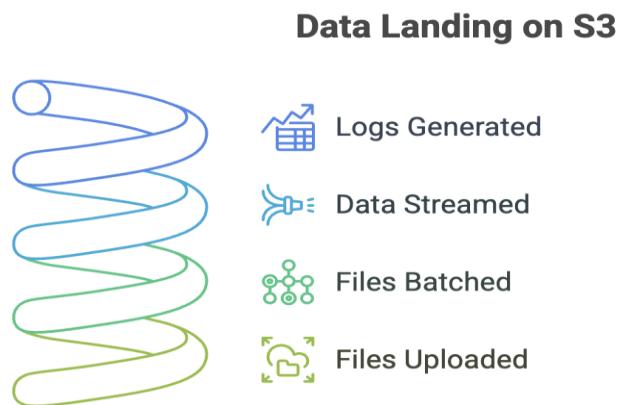
4. Monitoring and Optimization

- **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

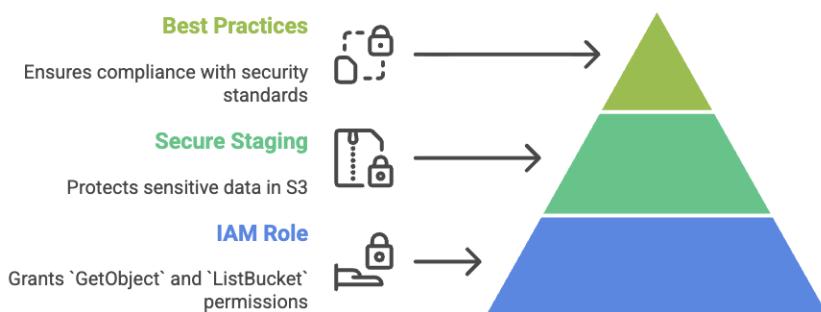
- Logs generated by different services are streamed to the S3 bucket.
- Files are batched and upload to the S3 bucket.



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.

Secure Data Access Hierarchy

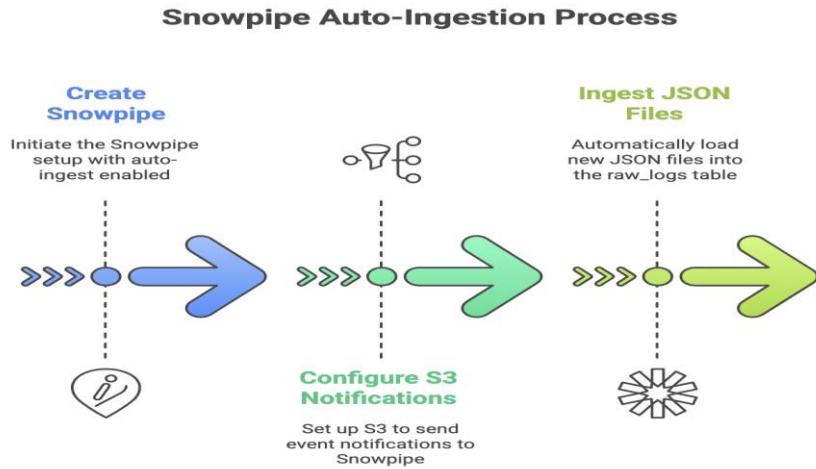


Step 3: Defining File Format and Copy Options

- 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.

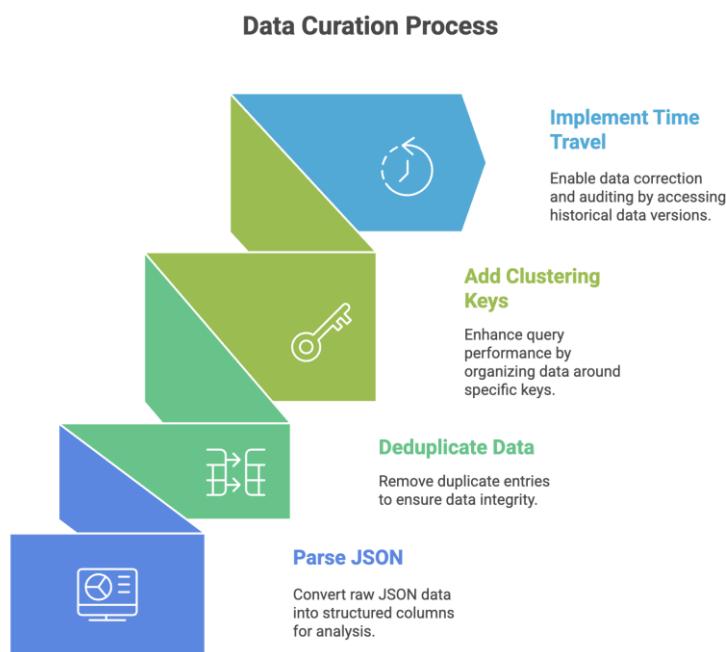
Step 4: Automating Ingestion with Snowpipe

- 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.



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.

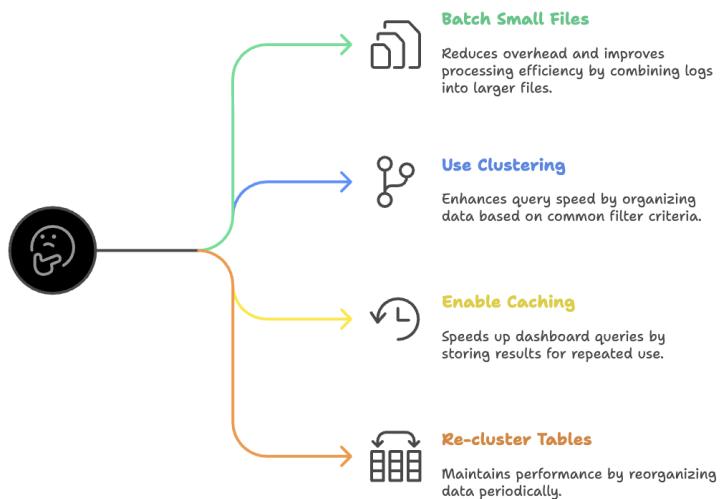


Step 6: Performance Optimization

- 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.

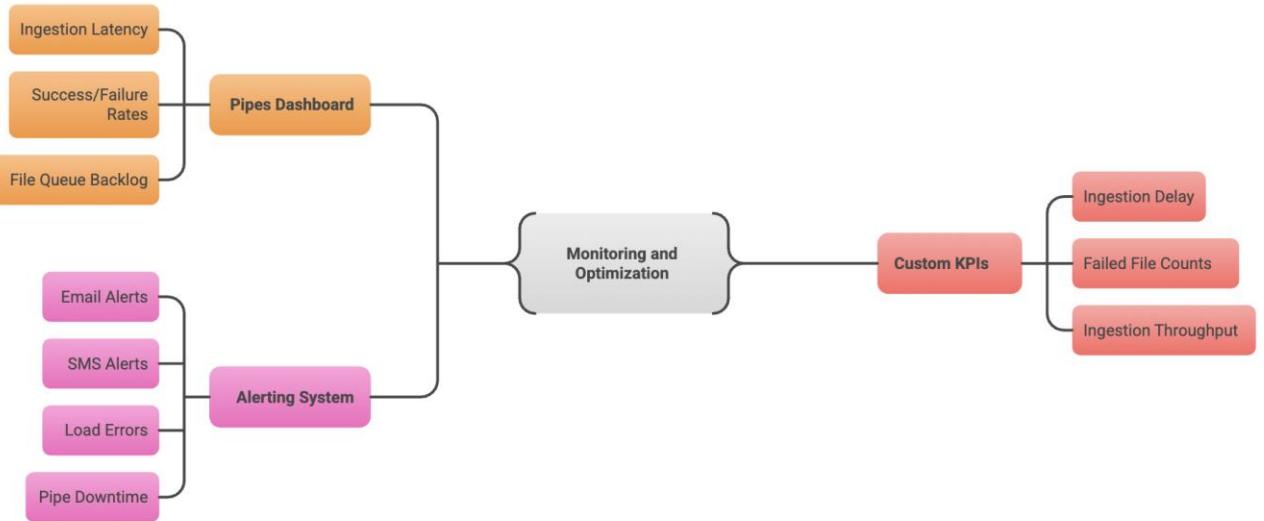
How to optimize Snowflake performance?



Step 7: Monitoring with Snowsight

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

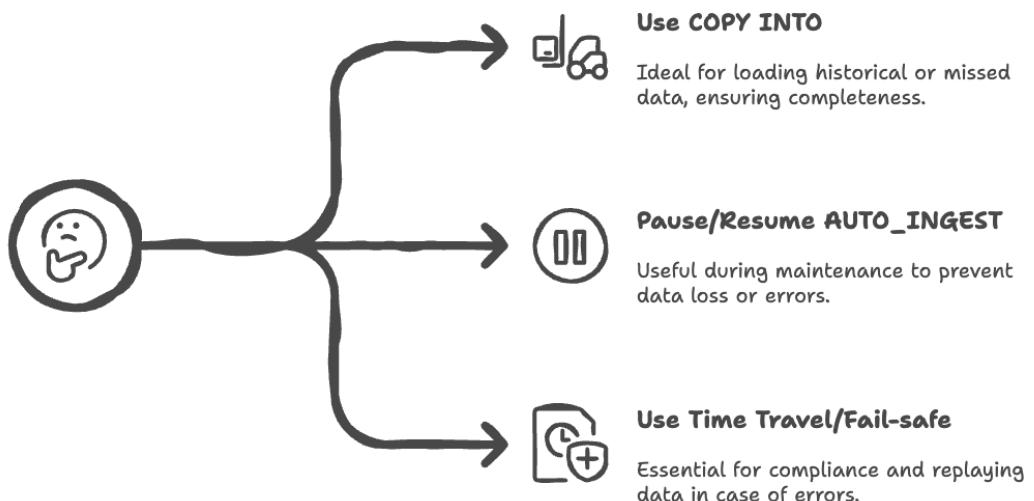
Monitoring and Optimization Strategies



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?

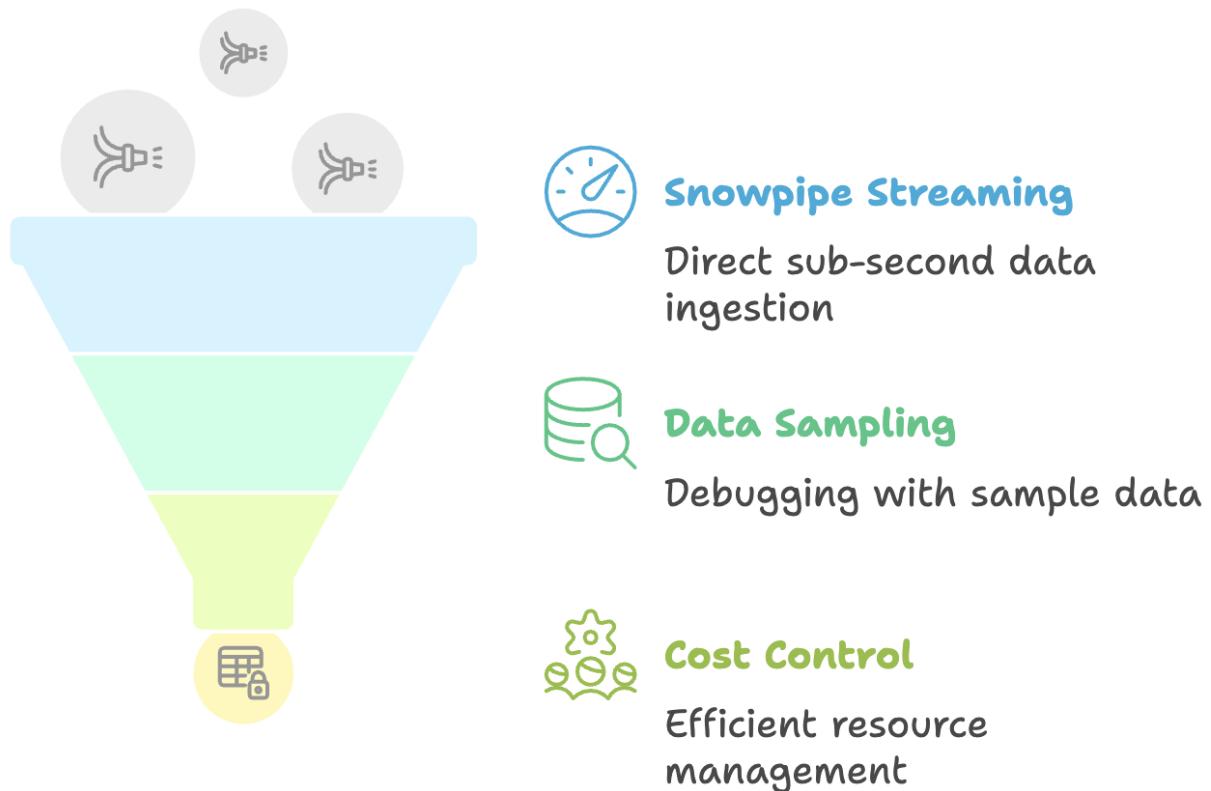


Step 9: Advanced Capabilities

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

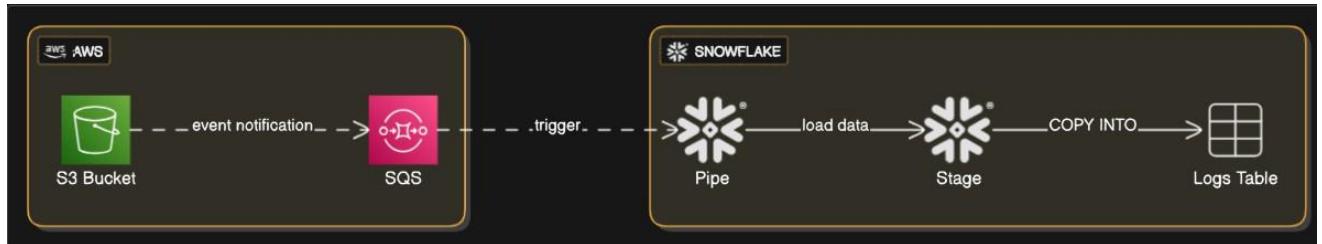
- **Cost Control:**
 - Use Snowpipe for ingestion (Snowflake-managed compute).
 - Use auto-suspend warehouses for transformation to minimize compute cost.

Optimizing Data Ingestion and Transformation



ARCHITECTURE:

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



IMPLEMENTATION

S3 BUCKET:

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

The screenshot shows the AWS S3 console. On the left, the navigation pane lists 'General purpose buckets' including 'Amazon S3', 'Directory buckets', 'Table buckets', 'Vector buckets', 'Access Grants', 'Access Points (General Purpose Buckets, FSx file systems)', 'Access Points (Directory Buckets)', 'Object Lambda Access Points', 'Multi-Region Access Points', and 'Batch Annotations'. The main area displays the 'snowflake-logs-data-bucket' with an 'info' button. Below it, there are tabs for 'Objects', 'Metadata', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. Under the 'Objects' tab, there is a table with one item: a folder named 'logs/'. The table has columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'.

SNOWFAKE WAREHOUSE:

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

The screenshot shows the Snowflake SQL editor. The left sidebar shows 'My Workspace' with files like 'Advanced snowflake lab-1.sql', 'case-study.sql', 'Snowflake-assignment.sql', 'Untitled 1.sql', 'Untitled.sql', and 'Week2_Day_5.sql'. The main area shows the SQL code for creating a warehouse, database, and schema:

```

1 -- Create Warehouse, Database, Schema
2 CREATE OR REPLACE WAREHOUSE Sigmoid
3   WAREHOUSE_SIZE = 'XSMALL'
4   AUTO_SUSPEND = 300
5   AUTO_RESUME = TRUE;
6
7 CREATE OR REPLACE DATABASE logsdb;
8 CREATE OR REPLACE SCHEMA logsdb.raw;
9 USE SCHEMA logsdb.raw;

```

The status bar at the bottom indicates the session is 'ACCOUNTADMIN - SIGMOID (X-Small)'.

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

The screenshot shows the Snowflake SQL editor. The left sidebar shows 'My Workspace' with files like 'Advanced snowflake lab-1.sql', 'case-study.sql', 'Snowflake-assignment.sql', 'Untitled 1.sql', 'Untitled.sql', and 'Week2_Day_5.sql'. The main area shows the SQL code for creating a target table and a JSON file format:

```

11 -- Create the target table
12 CREATE OR REPLACE TABLE log_events (
13   log_id STRING,
14   timestamp TIMESTAMP_LTZ,
15   level STRING,
16   service STRING,
17   message STRING,
18   user_id INT,
19   ip_address STRING
20 );
21
22 -- Create JSON file format
23 CREATE OR REPLACE FILE FORMAT json_ff
24   TYPE = JSON
25   STRIP_OUTER_ARRAY = TRUE;

```

The status bar at the bottom indicates the session is 'ACCOUNTADMIN - SIGMOID (X-Small)'.

S3 READ POLICY(Role):

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

The screenshot shows the AWS IAM Roles page. The role 'Snowflake-s3-storage-logs-access' is selected. The 'Summary' tab is active, displaying details like creation date (September 28, 2025), ARN (arn:aws:iam::687185077902:role/Snowflake-s3-storage-logs-access), and last activity (4 hours ago). The 'Permissions' tab is selected, showing one managed policy: 'AmazonS3ReadOnlyAccess'. A link to switch roles in the console is also present.

S3 and SNOWFLAKE INTEGRATION:

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

```
-- Create Storage Integration
CREATE OR REPLACE STORAGE INTEGRATION s3_logs_integ
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = 'S3'
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::687185077902:role/Snowflake-s3-storage-Logs-access'
STORAGE_ALLOWED_LOCATIONS = ('s3://snowflake-Logs-data-bucket/logs');
```

- ✓ Describe the created integration.

```
DESC INTEGRATION s3_Logs_integ;
```

- ✓ Copy the ARN and External_ID from the OUTPUT.

property	property_type	property_value	property_default
ENABLED	Boolean	true	false
STORAGE_PROVIDER	String	S3	
STORAGE_ALLOWED_LOCATIONS	List	s3://snowflake-Logs-data-bucket/logs/	[]
STORAGE_BLOCKED_LOCATIONS	List		[]
STORAGE_AWS_IAM_USER_ARN	String	arn:aws:iam::974916068036:user/externalstages/ci9s7c0000	
STORAGE_AWS_ROLE_ARN	String	arn:aws:iam::687185077902:role/Snowflake-s3-storage-Logs-access	
STORAGE_AWS_EXTERNAL_ID	String	JK33786_SFRole=2_tEY1K8FkFUGRUItzOpTfZRDriQ0=	
USE_PRIVATELINK_ENDPOINT	Boolean	false	false
COMMENT	String		

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

```

1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Principal": {
7          "AWS": "<STORAGE_AWS_IAM_USER_ARN>"
8        },
9        "Action": "sts:AssumeRole",
10       "Condition": {
11         "StringEquals": {
12           "sts:ExternalId": "<STORAGE_AWS_EXTERNAL_ID>"
13         }
14       }
15     ]
16   }
17 }

```

STAGE CREATION:

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

```

36 -- Create Stage pointing to your S3 bucket
37 CREATE OR REPLACE STAGE s3_logs_stage
38   URL = 's3://snowflake-logs-data-bucket/Logs/'
39   STORAGE_INTEGRATION = s3_logs_integ
40   FILE_FORMAT = json_ff;
41
42

```

- ✓ Successfully created a stage.

Database Explorer

Databases

LOGSDB / RAW / S3_LOGS_STAGE

Details

- Region: us-east-1
- Location: AWS
- URL: s3://snowflake-logs-dat...
- Has Access Credentials: No
- Directory Table: Enabled
- Storage Integration: S3_LOGS_INTEG

Privileges

ACCOUNTADMIN (Current Role)

SNOWPIPE:

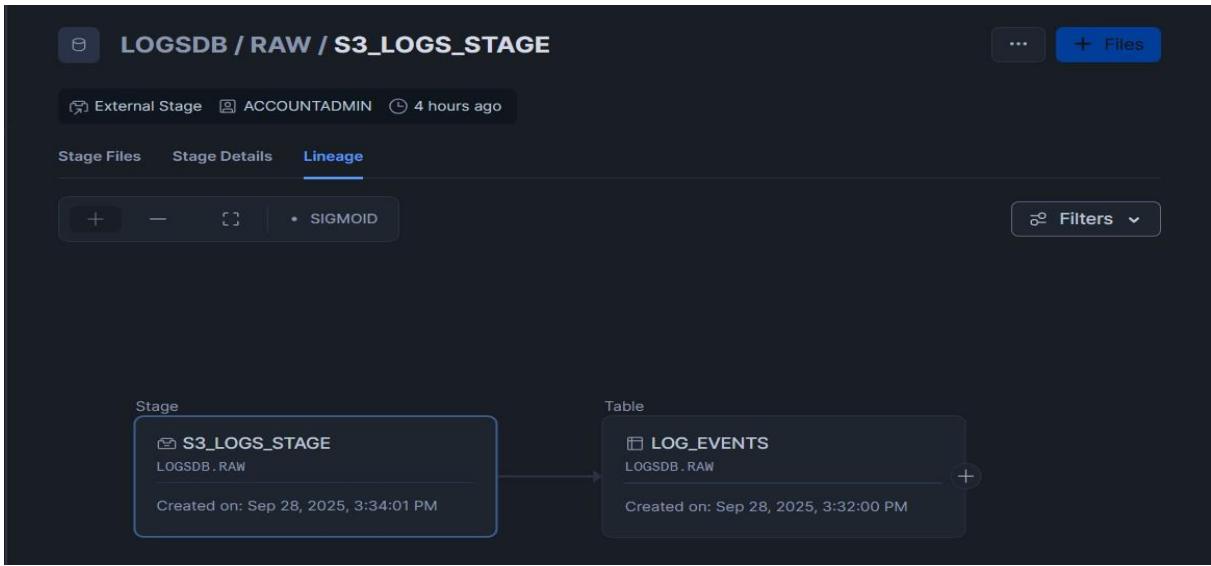
- ✓ Create a Snowpipe to ingest the data continuously into the logs table.

```

42
43   -- Create Snowpipe for auto-ingest
44   CREATE OR REPLACE PIPE log_pipe
45     AUTO_INGEST = TRUE
46   AS
47     COPY INTO Log_events
48     FROM @s3_logs_stage
49     FILE_FORMAT = (FORMAT_NAME = json_ff)
50     MATCH_BY_COLUMN_NAME = CASE_INSENSITIVE
51     ON_ERROR = 'CONTINUE';
52
53 DESC PIPE log_pipe;
54

```

- ✓ 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" page in the AWS S3 console. It includes sections for "General configuration" (Event name: "snowflake-logs-pipeline", Prefix: "logs/"), "Event types" (Put objects), and "Object creation" (All object create events). A note at the bottom says "Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events."

- ✓ Successfully created the SQS notification.

The screenshot shows the "Event notifications (1)" section in the AWS S3 console. It lists a single rule named "snowpipe" that triggers an "All object create events" event type on the prefix "logs/" to an SQS queue with the URL "arn:aws:sqs:us-east-1:974916068036:sf-snowpipe-AIDA6F7MGULCITXZYKBFX-yWhpMtZ5AtEdxhvlDjyw1g". There is also a section for "Amazon EventBridge" with a note about using it for event-driven applications.

FILE UPLOAD:

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

The screenshot shows the AWS S3 console interface. The top navigation bar includes the AWS logo, a search bar with 'iam', and account information for 'Account ID: 6871-8507-7902' and 'govind-admin'. Below the navigation bar, the path 'Amazon S3 > Buckets > snowflake-logs-data-bucket > logs/' is visible. The main area displays a table titled 'Objects (1)'. The table has columns for Name, Type, Last modified, Size, and Storage class. One object, 'logs_dataset1.json', is listed with a size of 20.9 KB and a storage class of Standard. The file was last modified on September 28, 2025, at 19:50:17 (UTC+05:30). Action buttons like Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload are available at the top of the table.

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

The screenshot shows the Snowflake assignment SQL workspace. The top navigation bar includes 'My Workspace > Snowflake-assignment.sql'. The workspace contains a single query: 'SELECT count(*) FROM log_events;'. The results section shows a table with one row, indicating 100 rows. The table has a single column '# COUNT(*)' with the value 100. The results are displayed in a tabular format with columns for search, sort, and export options.

ID	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-bf22-4ccb2fd5dd2	2025-09-28 07:44:53.993 -0700	DEBUG	payments	Low stock warning	3413	142.3.81.216
4	810ff445-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-4c07-92a7-023c1e928160	2025-09-28 07:55:51.993 -0700	ERROR	payments	Payment declined	2455	107.136.36.87
9	5e92ab64-1584-41da-ba27-64e2e0b9316c	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-0d9f-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.

The screenshot shows the AWS S3 console interface. The top navigation bar includes the AWS logo, a search bar with 'iam', and account information for 'Account ID: 6871-8507-7902' and 'govind-admin'. Below the navigation bar, the path 'Amazon S3 > Buckets > snowflake-logs-data-bucket > logs/' is visible. The main area displays a table titled 'Objects (2)'. The table has columns for Name, Type, Last modified, Size, and Storage class. Two objects, 'logs_dataset1.json' and 'logs_dataset2.json', are listed with a size of 20.9 KB and a storage class of Standard. Both files were last modified on September 28, 2025, at 19:50:17 (UTC+05:30).

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

Snowflake-assignment.sql

```

54
55     | SELECT count(*) FROM log_events;
56
57

```

Results (just now)

	# COUNT(*)
1	200

Results (just now)

	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-bdfc-43b7-92ac-a84ad583e0b	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-4c07-92a7-023c1e92eb160	2025-09-28 07:55:51.993 -0700	ERROR	payments	Payment declined	2455	107.136.36.87
9	5e92ab64-1584-41da-ba27-64e2eb09316f	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	e296f56a-0df9-48ef-9dc7-70c0c7161777	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	c1832f8-1202-4598-b5d4-1002fc382bb	2025-09-28 07:39:27.993 -0700	INFO	auth	Order placed	3570	81.216.32.197
14	3119372f-f6db-471d-a1c4-7150f202a90	2025-09-28 08:30:32.993 -0700	DEBUG	shipping	Invalid credentials	4977	5.58.136.174
15	f56e8c36-a070-4168-ac2d-b739ad8dd0b0	2025-09-28 09:07:25.993 -0700	ERROR	inventory	Order placed	2858	1.134.91.54
...	Address validation failed	1014	78.101.89.0

TIME TRAVEL:

- ✓ Retention period is 7 days.

My Workspace

Snowflake-assignment.sql

```

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)

	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

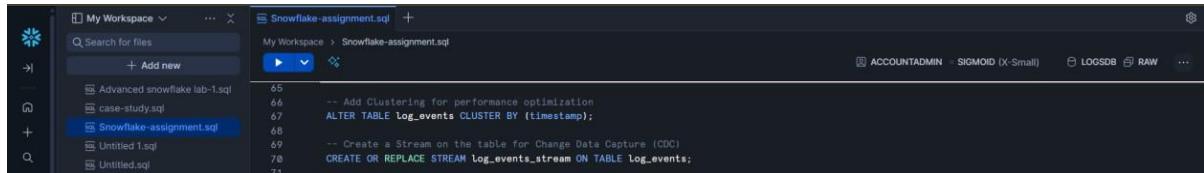
- ✓ Table logs data is of 100 Rows.(past of [400] seconds)

Results (just now)

	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

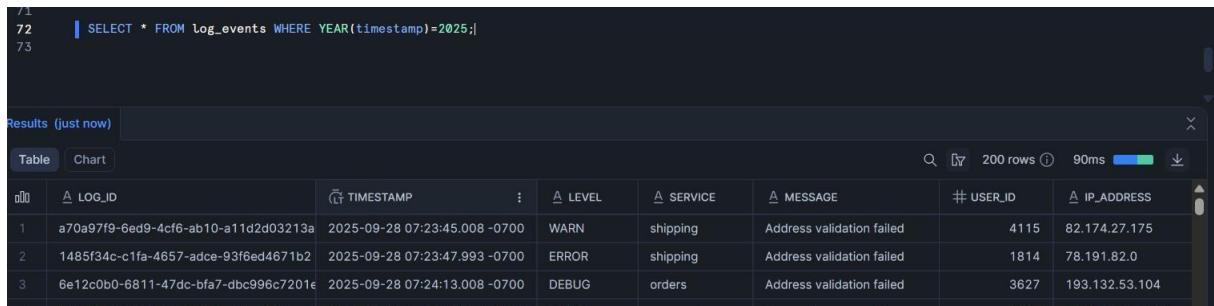
CACHE AND QUERY OPTIMIZATION:

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



```
My Workspace > Snowflake-assignment.sql +  
My Workspace > Snowflake-assignment.sql  
65 -- Add Clustering for performance optimization  
66 ALTER TABLE log_events CLUSTER BY (timestamp);  
67  
68 -- Create a Stream on the table for Change Data Capture (CDC)  
69 CREATE OR REPLACE STREAM log_events_stream ON TABLE log_events;  
70  
71
```

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

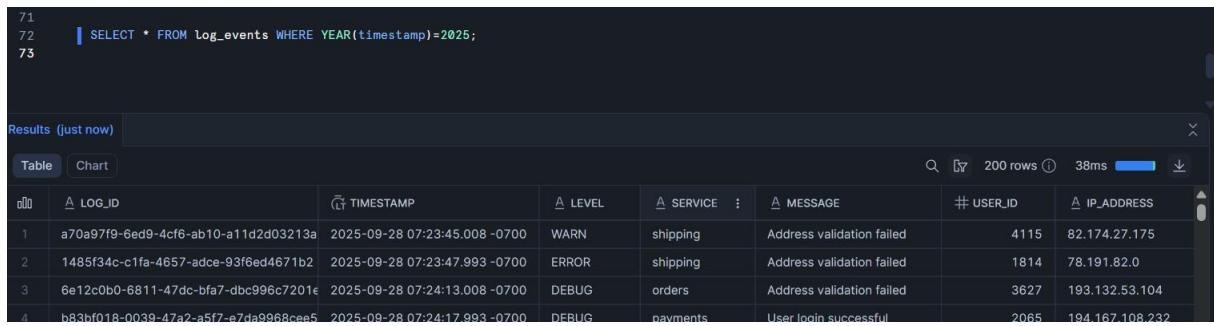


```
71  
72 | SELECT * FROM log_events WHERE YEAR(timestamp)=2025;  
73
```

Results (just now)

	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

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



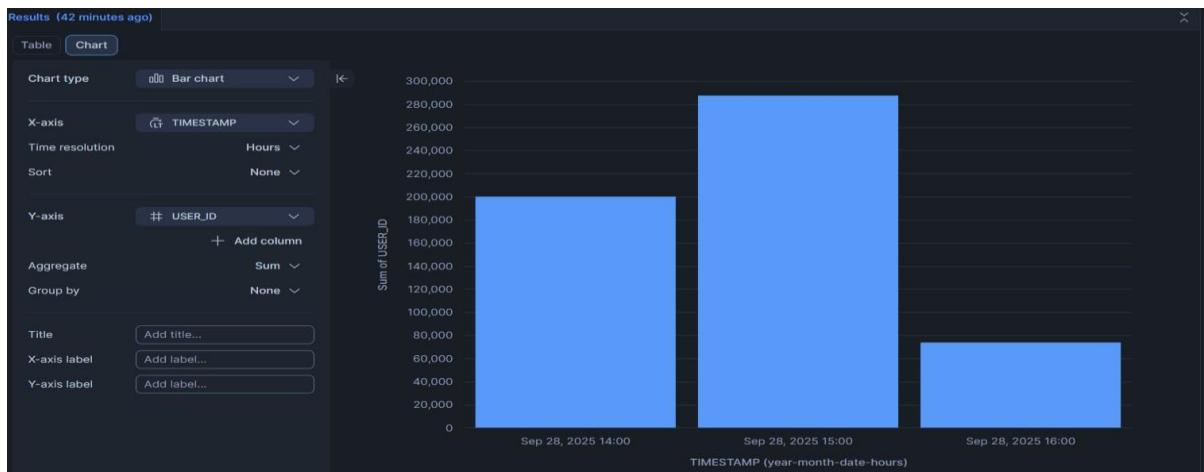
```
71  
72 | SELECT * FROM log_events WHERE YEAR(timestamp)=2025;  
73
```

Results (just now)

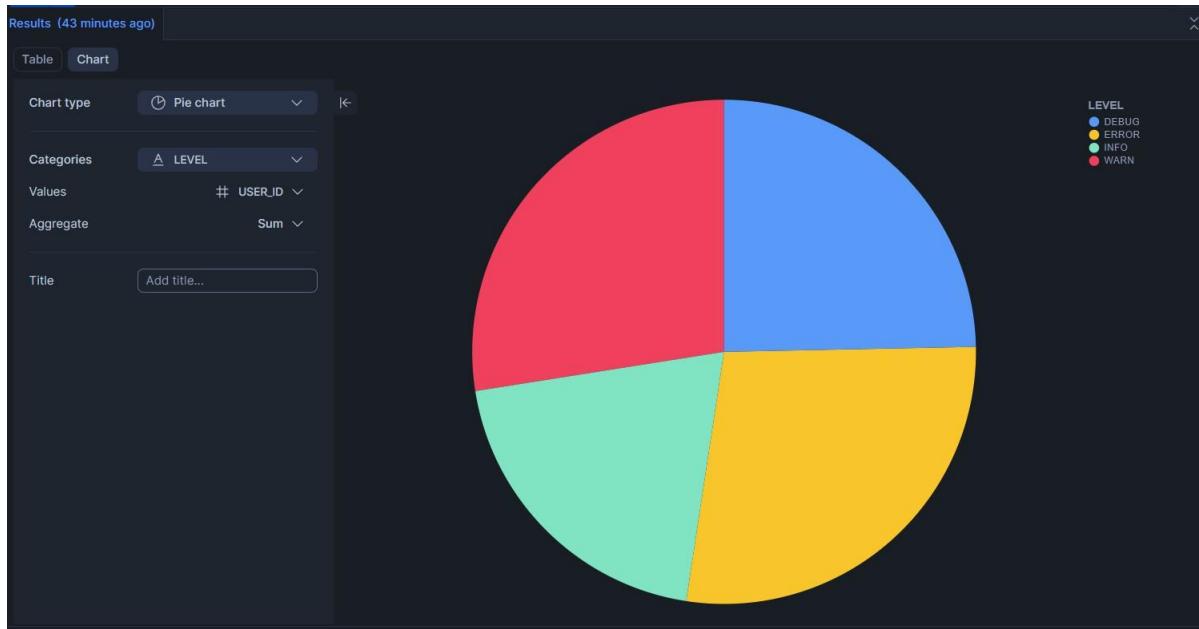
	LOG_ID	TIMESTAMP	LEVEL	SERVICE	MESSAGE	USER_ID	IP_ADDRESS
1	a70a97f9-6ed9-4cf6-ab10-a11d2d03213a	2025-09-28 07:23:45.008 -0700	WARN	shipping	Address validation failed	4115	82.174.27.175
2	1485f34c-c1fa-4657-adce-93f6ed4671b2	2025-09-28 07:23:47.993 -0700	ERROR	shipping	Address validation failed	1814	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
4	b83bf018-0039-47a2-a5f7-e7da9968ce5	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.

