A blue and red text

Description automatically generated

**POV Leveraging Iceberg Tables in Snowflake Data Cloud**

**Team Owners** (Part-Time):

Dhamu / Syamsundar Gudibandi

**Support Team** (Part-Time)

Syamsundar Gudibandi

Mounika Koduru

Srisai Maddukuri

Uha Priya Akunuri

**Dedicated Team**

Swetha Appisetty Himabindhu Ogirala

Gayathri Arepalli Anil Bitra

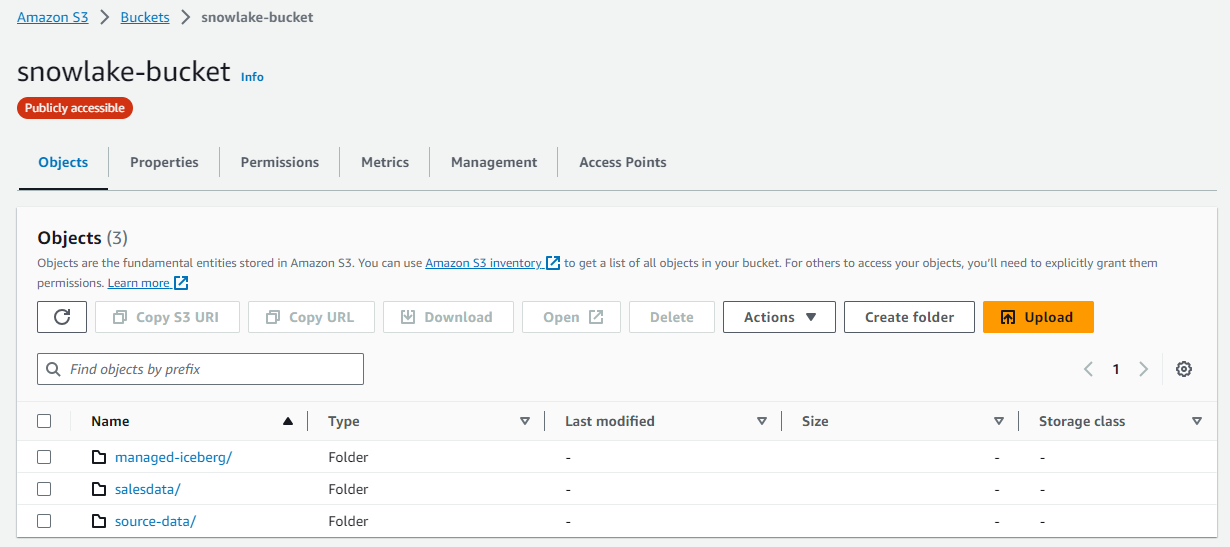
Akshitha Veesam Surendra Reddy Sanayapalli

|  |  |  |
| --- | --- | --- |
| Index | Table of contents | Page No |
| **1** | **STEPS FOLLOWED FOR UNMANAGED ICEBERG TABLES** | 3 |
| 1.1 | Creating s3 bucket and load datasets in folder | 3 |
| 1.2 | Crawler Creation | 4 |
| 1.3 | Iceberg Table Creation in Athena | 5 |
| 1.4 | Creating an External Volume and Catalog Integration | 6 |
| 1.5 | Iceberg Table Creation in Snowflake | 9 |
| 1.6 | Features applied on unmanaged iceberg table | 10 |
|  | 1.6.1 Schema Evolution | 10 |
|  | 1.6.2 Data Masking and RBAC | 10 |
|  | 1.6.3 Secure Data Share | 11 |
| **2** | **STEPS FOLLOWED FOR UNMANAGED ICEBERG TABLES** | 12 |
| 2.1 | Creating s3 bucket and load datasets in folder | 12 |
| 2.2 | Creating an External volume | 13 |
| 2.3 | Creating Storage Integration | 16 |
| 2.4 | Creating External Stage | 17 |
| 2.5 | Creating Iceberg Tables | 17 |
| 2.6 | Metadata Information | 18 |
| 2.7 | Incremental Loading | 19 |
| 2.8 | Interoperability | 19 |
| 2.9 | Features Applied on Iceberg Table | 20 |
|  | 2.9.1 Schema Evolution | 20 |
|  | 2.9.2 Data Masking and RBAC | 21 |
|  | 2.9.3 Multi-Table Transaction | 22 |
|  | 2.9.4 Joins | 23 |
|  | 2.9.5 Secure Data Share | 23 |
| **3** | **STEPS FOLLOWED FOR EXTERNAL TABLE CREATION** | 25 |
| 3.1 | Create integration, external stage, external table | 25 |
| **4** | **PERFORMANCE PARITY** | 26 |
| 4.1 | Queries | 26 |
| 4.2 | Metrics | 26 |
| **5** | **REFERENCES** | 27 |

1. STEPS FOLLOWED FOR UNMANAGED ICEBERG TABLES

**1.1 Creating s3 bucket and load datasets in folder**

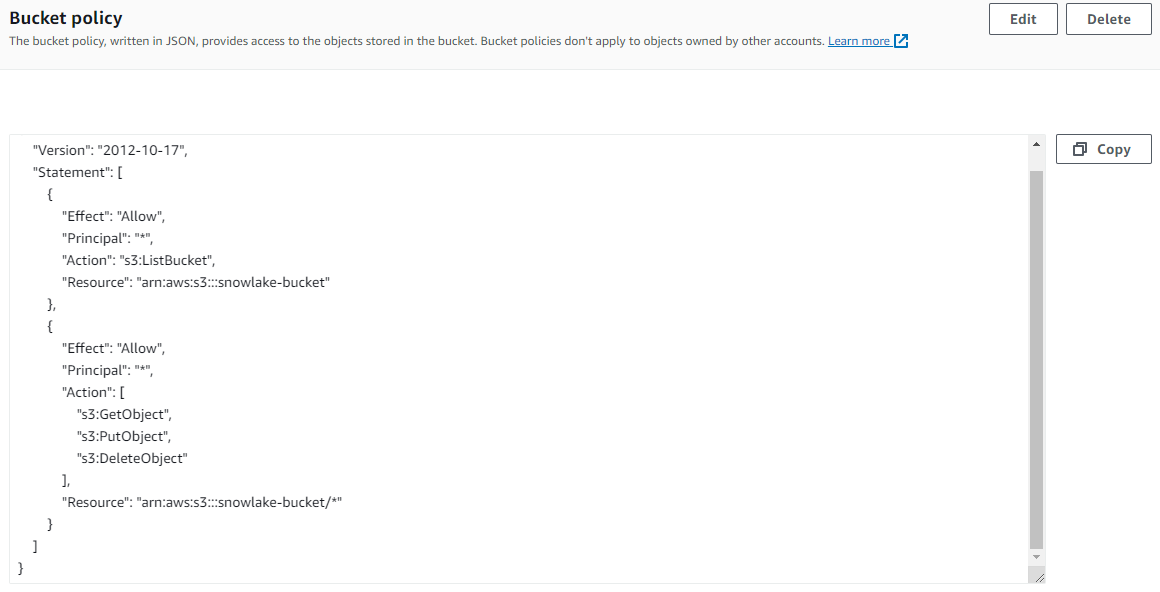
First, we created an s3 bucket(snowlake-bucket) and then created a folder in s3bucket(salesdata), In salesdata we uploaded a parquet file.

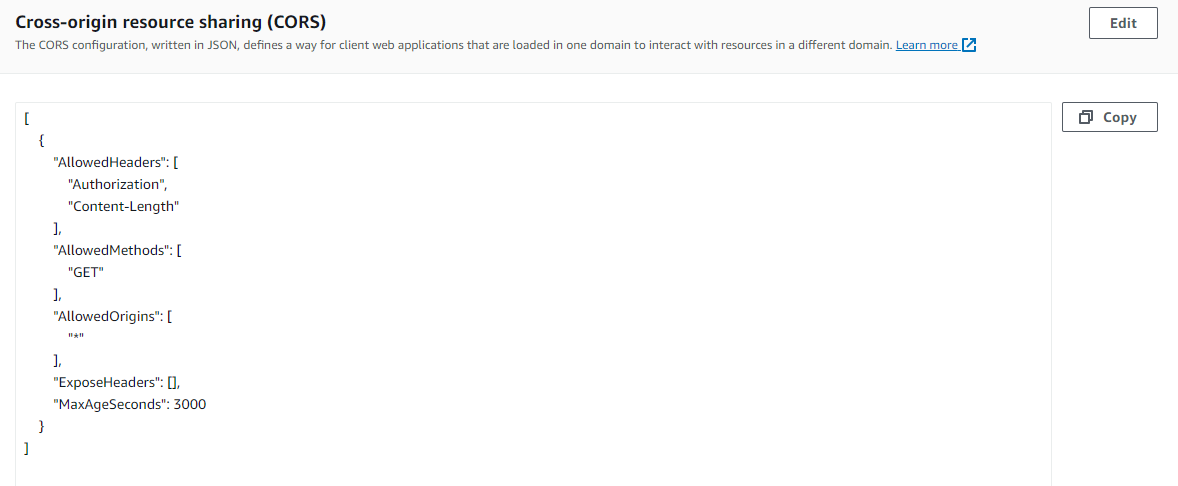


After creation of s3 bucket we must make sure the permissions are updated

1)Block public access (bucket settings) - while creating s3 bucket we must make sure the block all public access to be off

2) Edit the bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts.



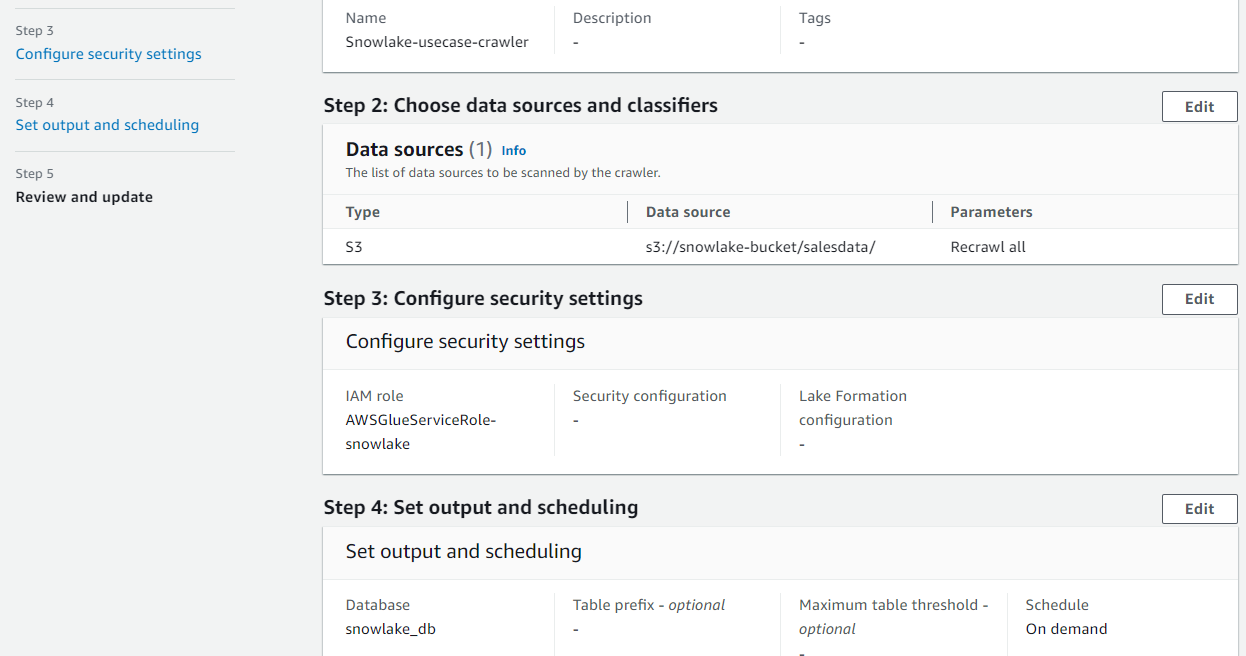
  3) And edit the Cross-origin resource sharing (CORS), The CORS configuration, written in JSON, defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. 

**1.2 CRAWLER CREATION**

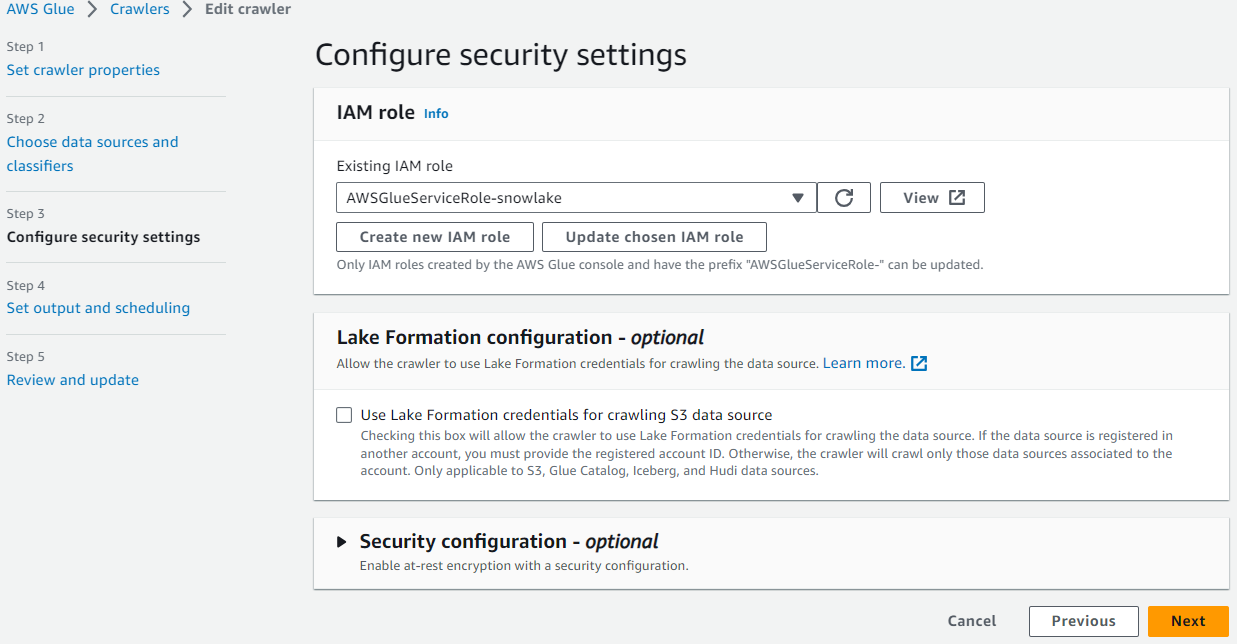
Steps:

       1) Go to AWS glue console, create a crawler with the name snowlake-unmanaged-crawler.

       2) Next data source and classifiers, in this data sources choose created s3 bucket



  3) Next configure security settings, we have created a new IAM role with name AWSGlueServicesRole-snowlake for glue.



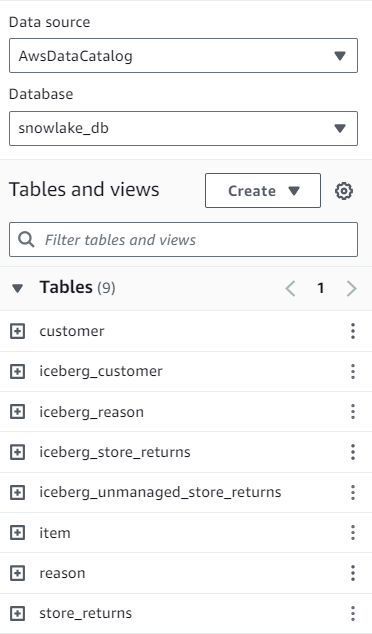
        4) Next, we click on add database and created a new database in step 4 and click on next.

         5) After that click on review and update.

         6) Next run the crawler and check database&tables in the data catalog are created or not.

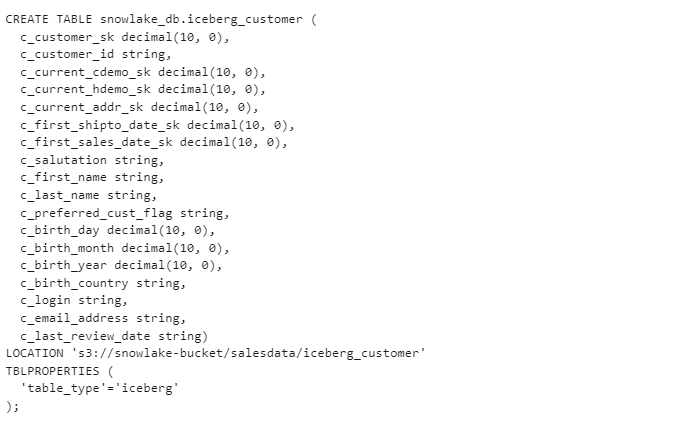
**1.3 Iceberg table creation in Athena**

Now go to AWS Athena and select database and check the tables in database on left panel.

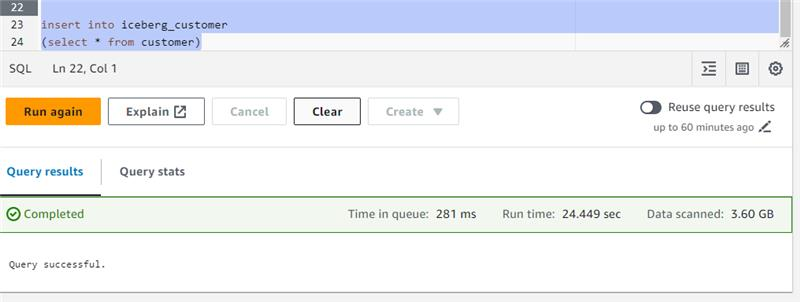
.

1)Go to the settings and make sure the query result location has a specified s3 bucket path which is used to store the query results of Athena.

2)In Athena open a new query editor and create an iceberg table with s3 bucket location.



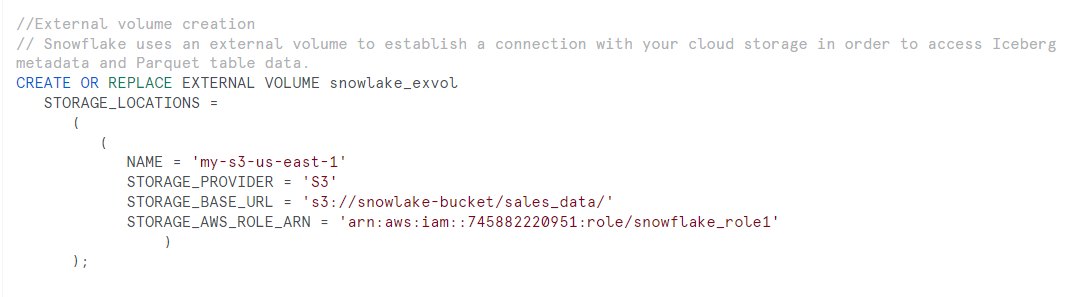
3)After that insert the data from customer data (table in database) into the created iceberg table.



**1.4 External volume and Catalog integration**

Now move to snowflake, open a worksheet and create an External volume.

Code:



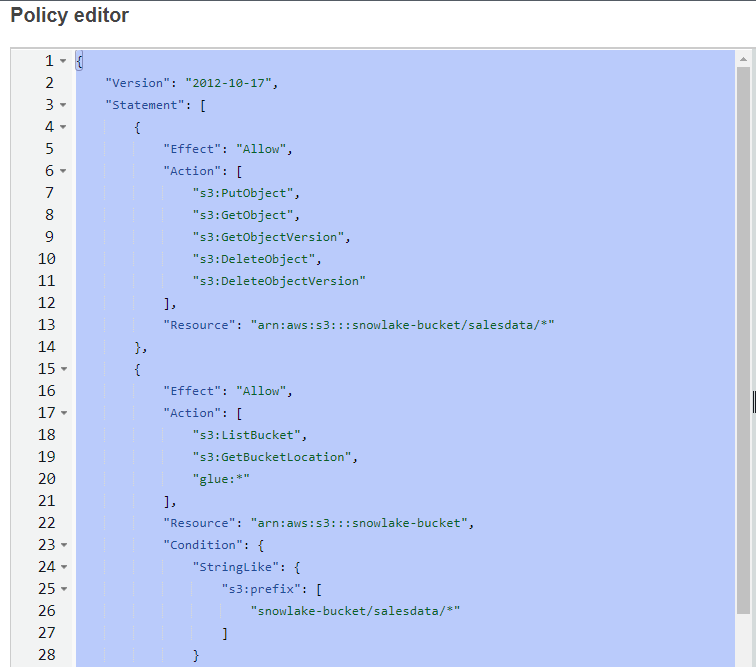
1)After creation we must describe the external volume



2)From this property value we must get “STORAGE\_AWS\_IAM\_USER\_ARN” and “STORAGE\_AWS\_EXTERNAL\_ID” values to place them in Trust relationship   for the role (AWSGlueServiceRole-snowlake).

3)We have created 2 roles, each of them is used for s3 and glue access.

1) Role for S3 bucket used in external volume in this role clicks on add permissions and add a policy, click on create new policy and click Json to add bucket details



4)Click on next and save changes and add policy to role.

5)Click on role trust relationships and edit trust policy and add the “STORAGE\_AWS\_IAM\_USER\_ARN and “STORAGE\_AWS\_EXTERNAL\_ID” from description of external volume.



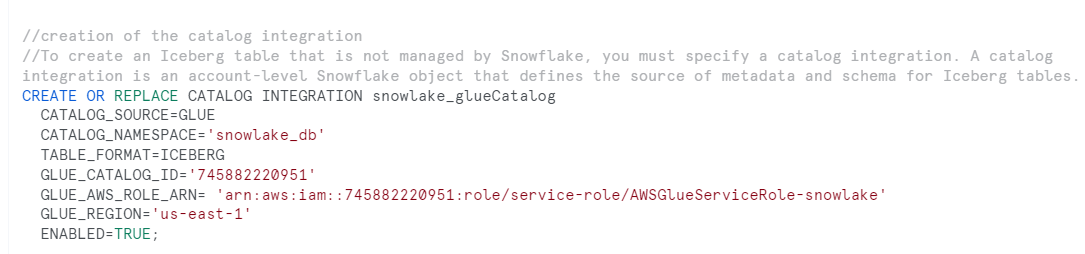
6)After that go to another role created for glue (AWSGlueServiceRole-snowlake) and click on add permissions and click on create a policy and click json to add glue details,



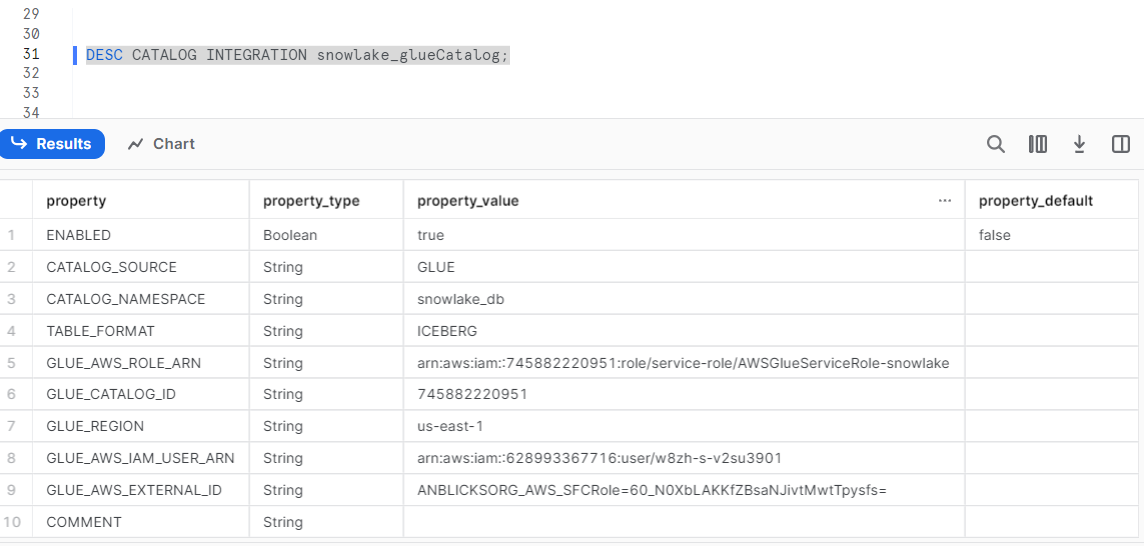
7)Click on next and save changes and add policy to the role.

8)Next go to snowflake and create a catalog integration.

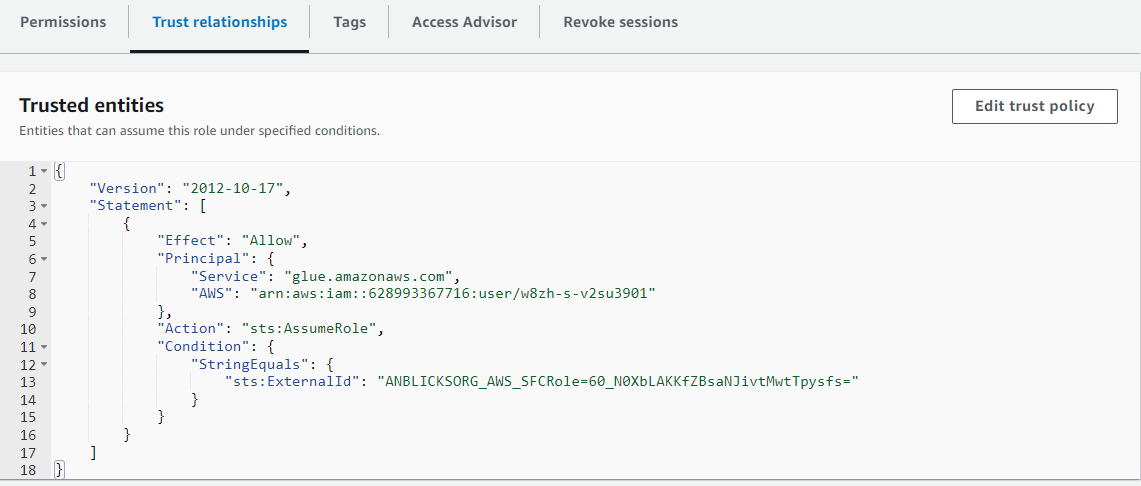
Code:



Next describe the catalog integration



9)From this get “GLUE\_AWS\_IAM\_USER\_ARN” and “GLUE\_AWS\_EXTERNAL\_ID” and place that in trust relationships of role AWSGlueServiceRole-snowlake

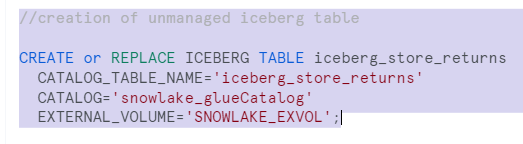


**1.5 Iceberg table creation in snowflake**

After that create an iceberg table

**Note: The catalog table name as well as iceberg table name in snowflake should be same as the iceberg table that is created in Athena**

code:



**1.6 Features applied on unmanaged iceberg table**

**1.6.1 Schema Evolution**

1) In the Athena, we performed the schema evolution for the iceberg table.

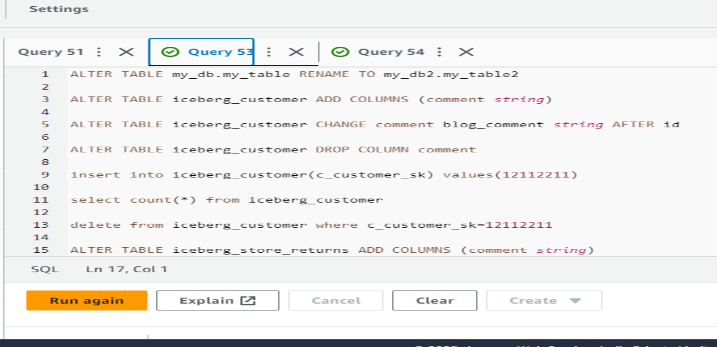
**Schema evolution** includes

  ADD

  DROP

  CHANGE

RENAME



2)In order to see these changes in the snowflake iceberg table, we need to refresh the table.

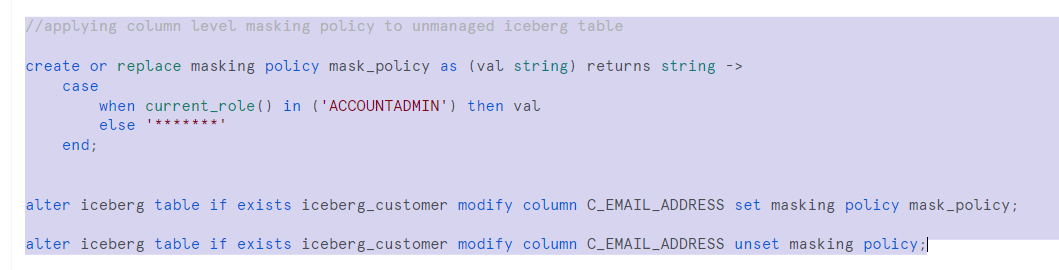
  -**- Alter iceberg table iceberg\_store\_returns refresh;**

3)To get the location of the metadata file and status of the snapshot generation

-- **Select SYSTEM$GET\_ICEBERG\_TABLE\_INFORMATION('iceberg\_store\_returns');**

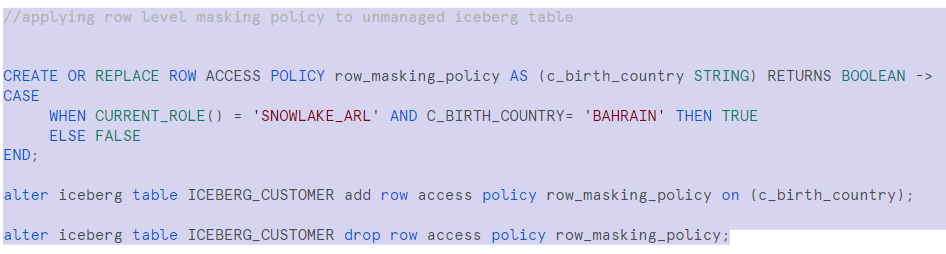
**1.6.2 Data Masking and RBAC**

1)After that we have created the column masking policy to the iceberg table.



We applied masking policy to the C\_EMAIL\_ADDRESS column of iceberg\_customer table.

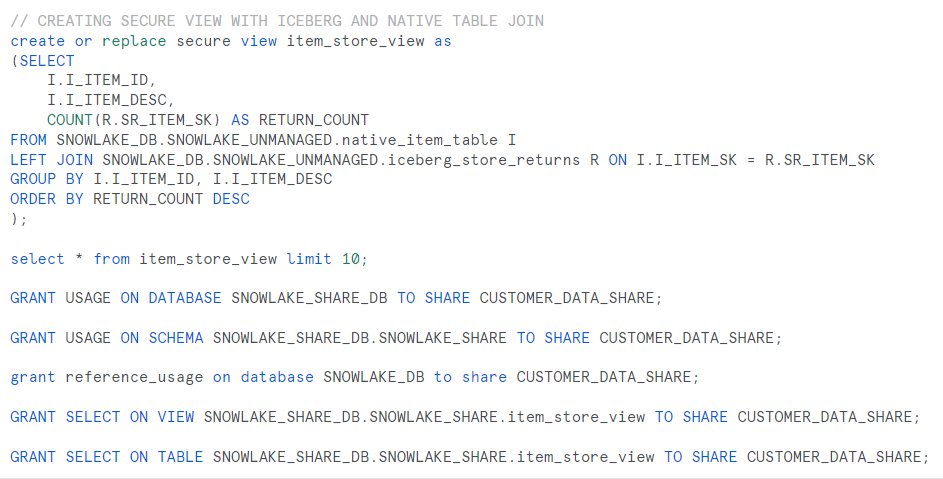
2)After that we applied row access policies to the iceberg table.



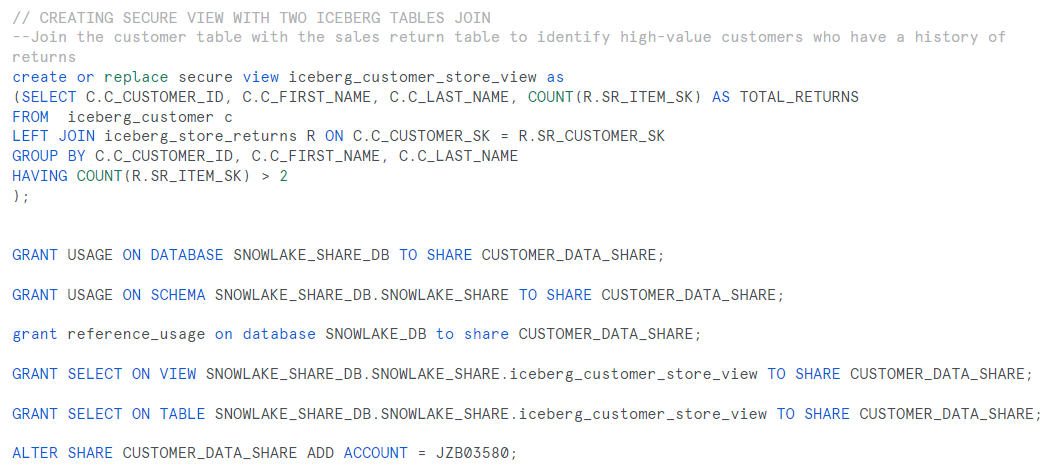
We applied row access policies to access the rows of the iceberg\_customer table whose C\_BIRTH\_COUNTRY =’BAHRAIN’

**1.6.3 Secure Data Share**

Share created on view



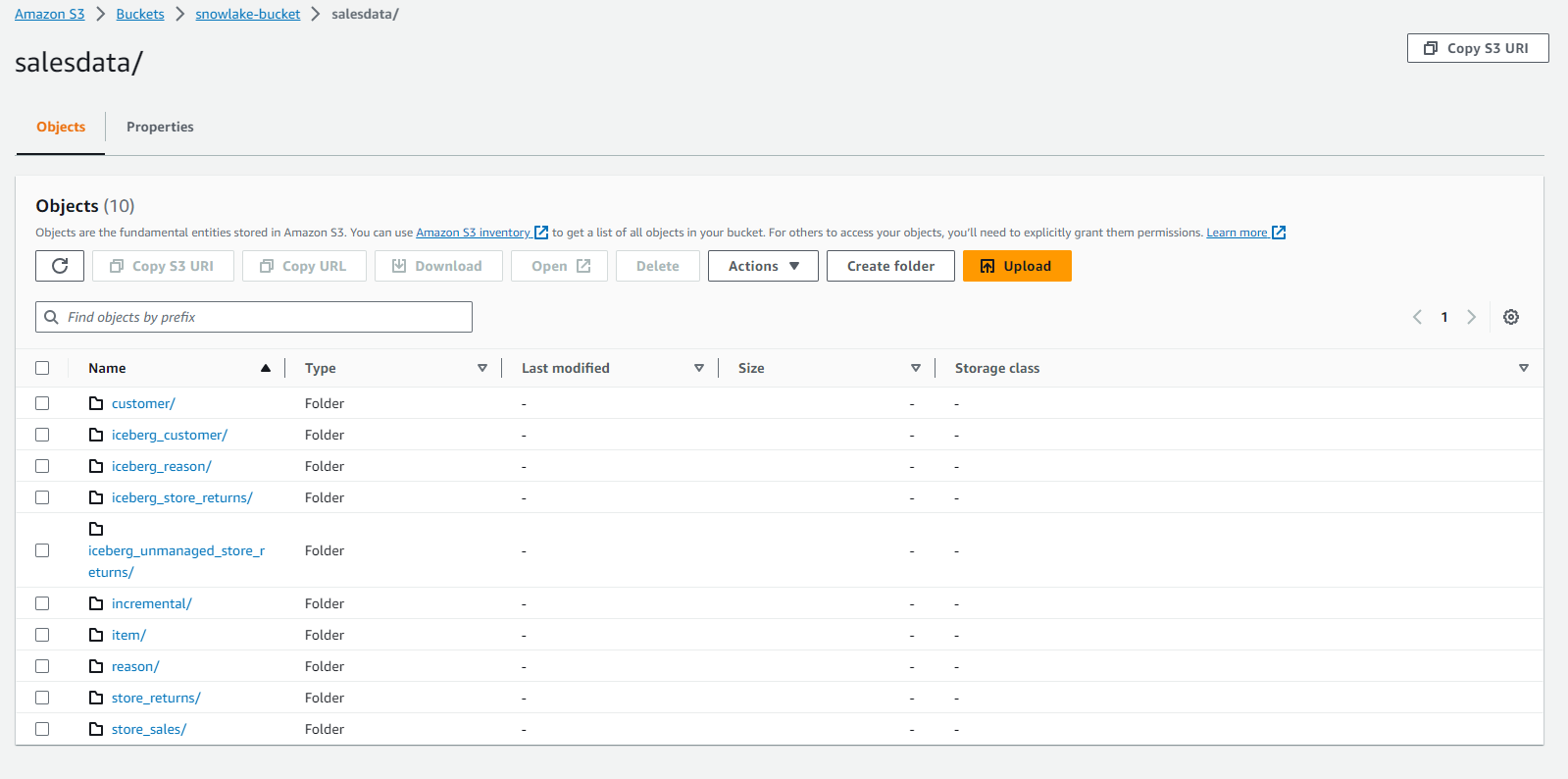
Share created on view on top of two iceberg tables



2. STEPS FOLLOWED FOR UNMANAGED ICEBERG TABLES

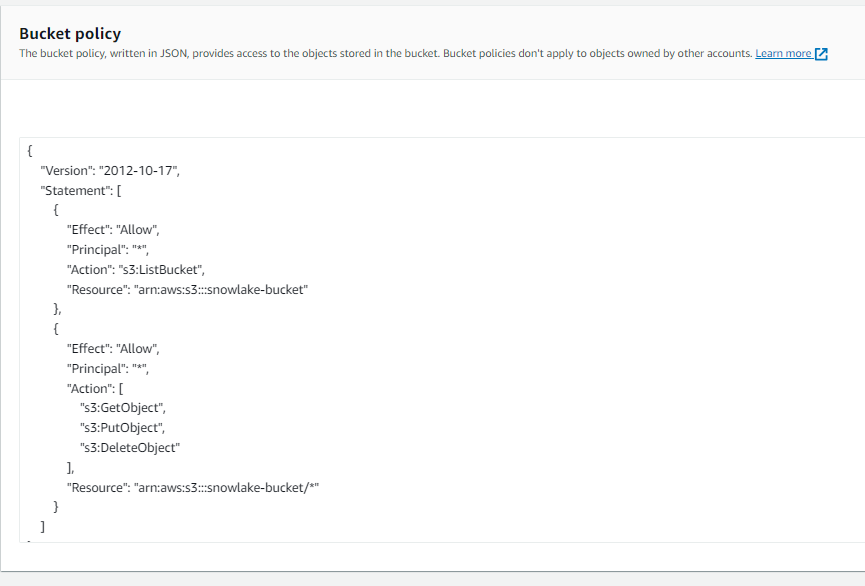
 2.1 Creating s3 bucket and load datasets in folder

First, we have created an s3 bucket(snowlake-bucket) and then created a folder in s3bucket(salesdata), In salesdata we have uploaded parquet files of store returns data.

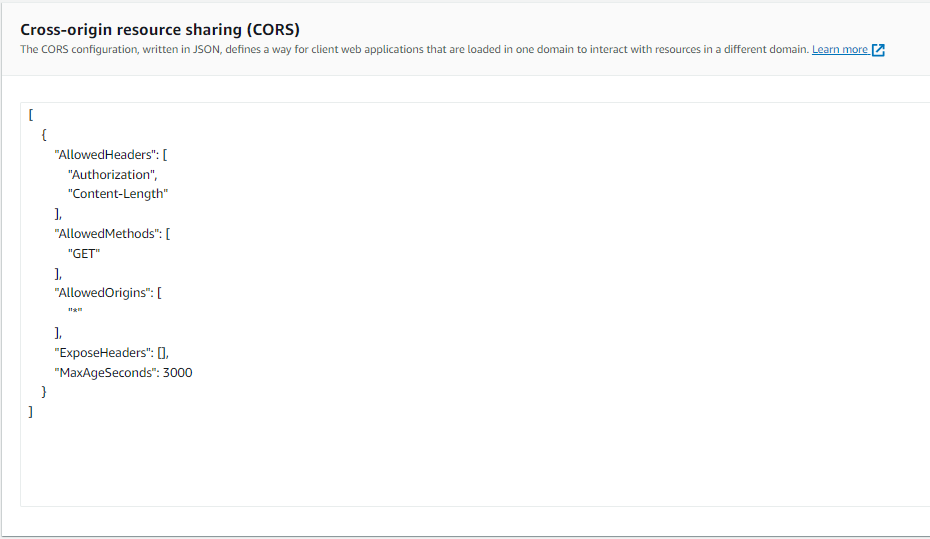


* After creation of s3 bucket we must make sure the permissions are updated  
  1)Block public access (bucket settings) - while creating s3 bucket we must make sure the block all public access to be off

2) Permissions --> Edit the bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts.



3) And edit the Cross-origin resource sharing (CORS), The CORS configuration, written in JSON, defines a way for client web applications that are loaded in one domain to interact with resources in a different domain.



**2.2 Creating an External volume**

**Step1**: Creating External Volume

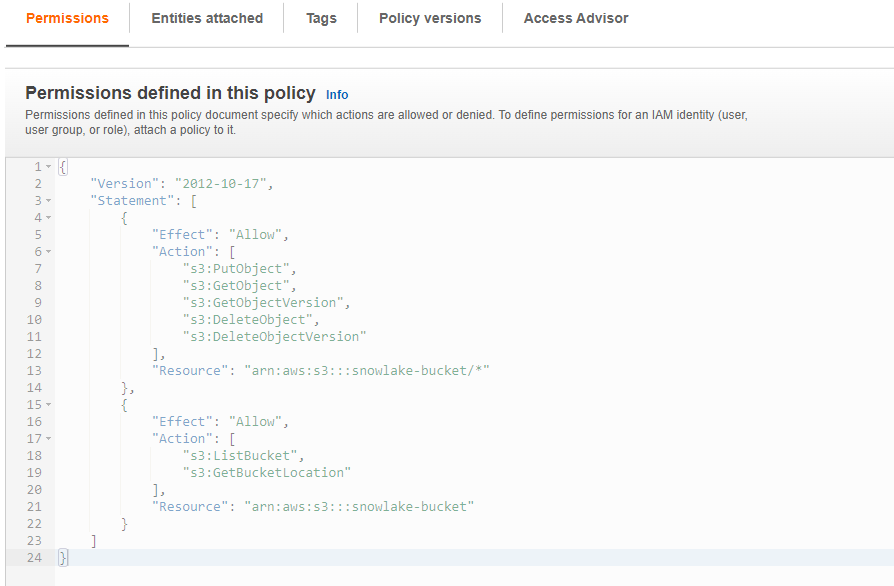
External Volume establishes connectivity to cloud storage for reading and writing iceberg metadata and parquet table data.

1) Log into the AWS Management Console --> Select IAM

2)Choose Policy and create a policy --> Select Create policy

3)Select JSON tab and edit the code in policy editor according to the documentation.

4)Give the bucket name and folder path in the code.



5)Name the policy as managed-access and enter create policy.

6)Select Roles from left-hand navigation pane --> Create role

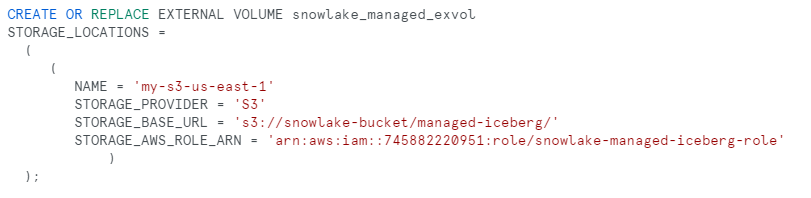
7)Choose aws account --> Select option external id with dummy id (‘0000’) --> Select Next

8)Select policy which was created in step 5 --> Select Next

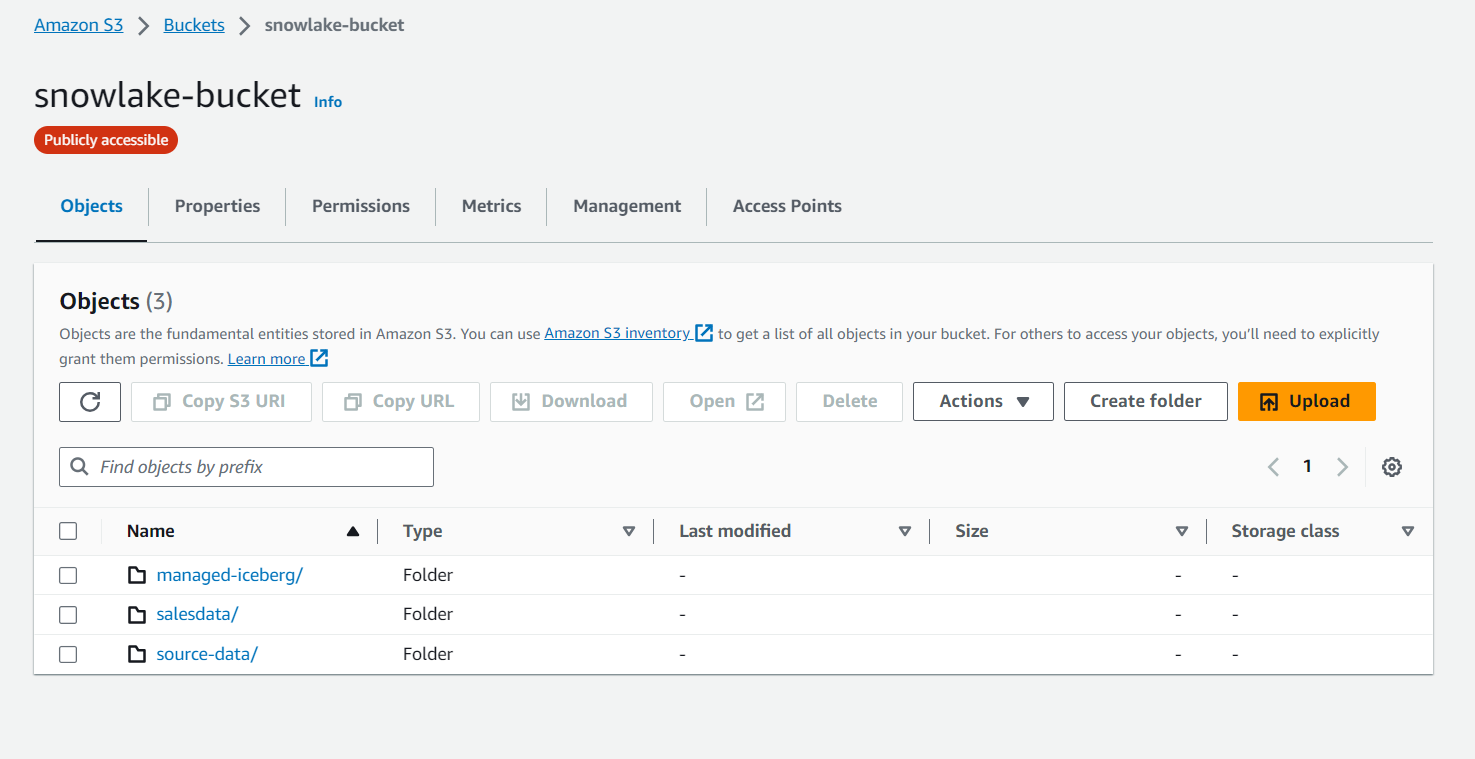
9)Give a role name --> Select create role

10)Copy the role ARN value.

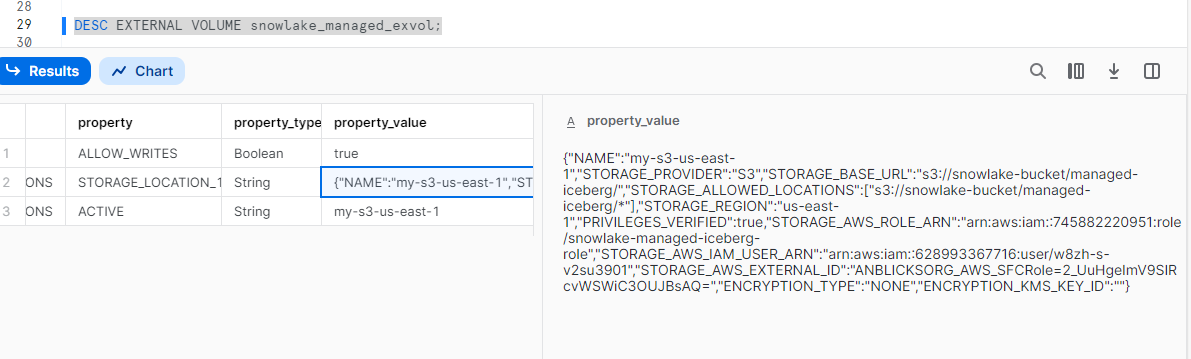
11)Open snowflake account --> Write the following code syntax



12)Create a folder in the snowlake\_bucket as managed iceberg to give this location as the storage\_base\_url in the external volume where we can access the iceberg table metadata and parquet data.

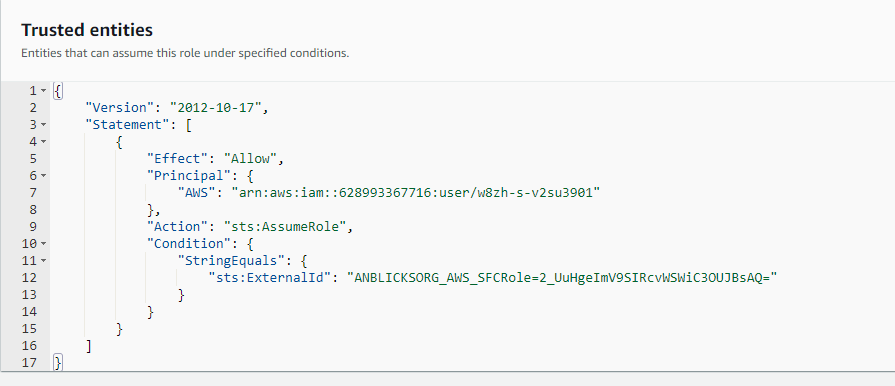


13)Describe the external volume and copy the values of STORAGE\_AWS\_IAM\_USER\_ARN and STORAGE\_AWS\_EXTERNAL\_ID.



14)Select the role you created in IAM --> Trust Relationship --> Edit Trust Relationship

15)Modify the trust relationship with the values recorded in step 12.



16)Select Update Trust Policy.

**2.3 Creating Storage integration**

**Step 2**: Creating Storage integration

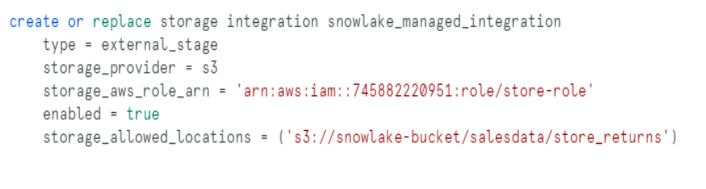
1)IAM --> ROLES --> Create Role

2)Select policy --> AmazonS3FullAccess

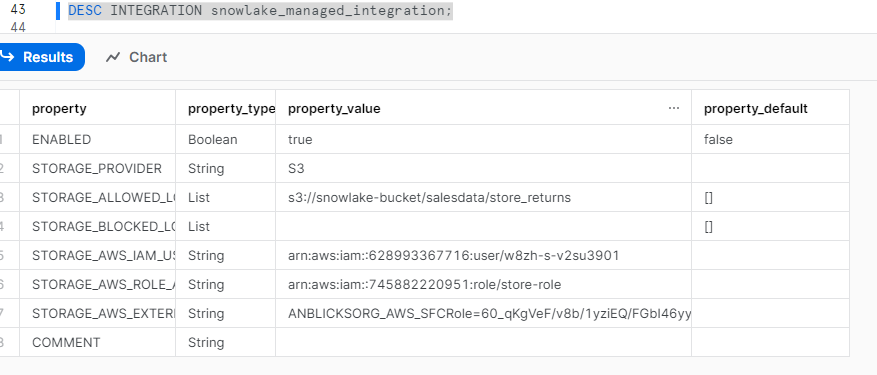
3)Give a role name --> Select Create Role

4)Copy the ARN value --> Open snowflake --> Create storage integration.

5)Give the storage\_allowed \_locations as the location where the source data for the iceberg table is present i.e., the location of store\_returns folder.



6)Describe the storage integration and copy the STORAGE\_AWS\_IAM\_USER\_ARN value.

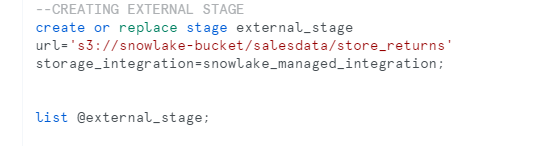


7)IAM --> Select the role created in step 3 --> Edit Trust relationship --> Update

**2.4 Creating an External Stage**

**Step 3:** Creating an external stage

1)Create external stage using s3 bucket location and storage integration created in Step 2



2)List the stage to see the files located in bucket

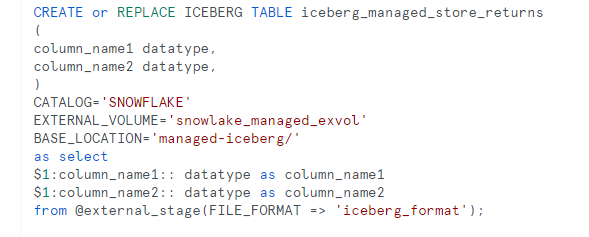
**2.5 Creating Iceberg Table**

**Step 4:** Creating an Iceberg Table

1)Create Iceberg Table using external volume.

2)Give the catalog as snowflake

3)Give the base location name as managed-iceberg (folder newly created in snowlake-bucket).



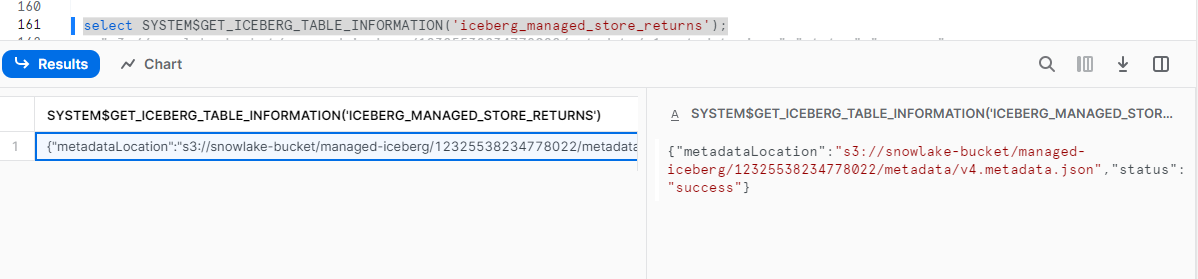
4)Create a file format for datatype parquet

5) Insert the data into iceberg table from external stage

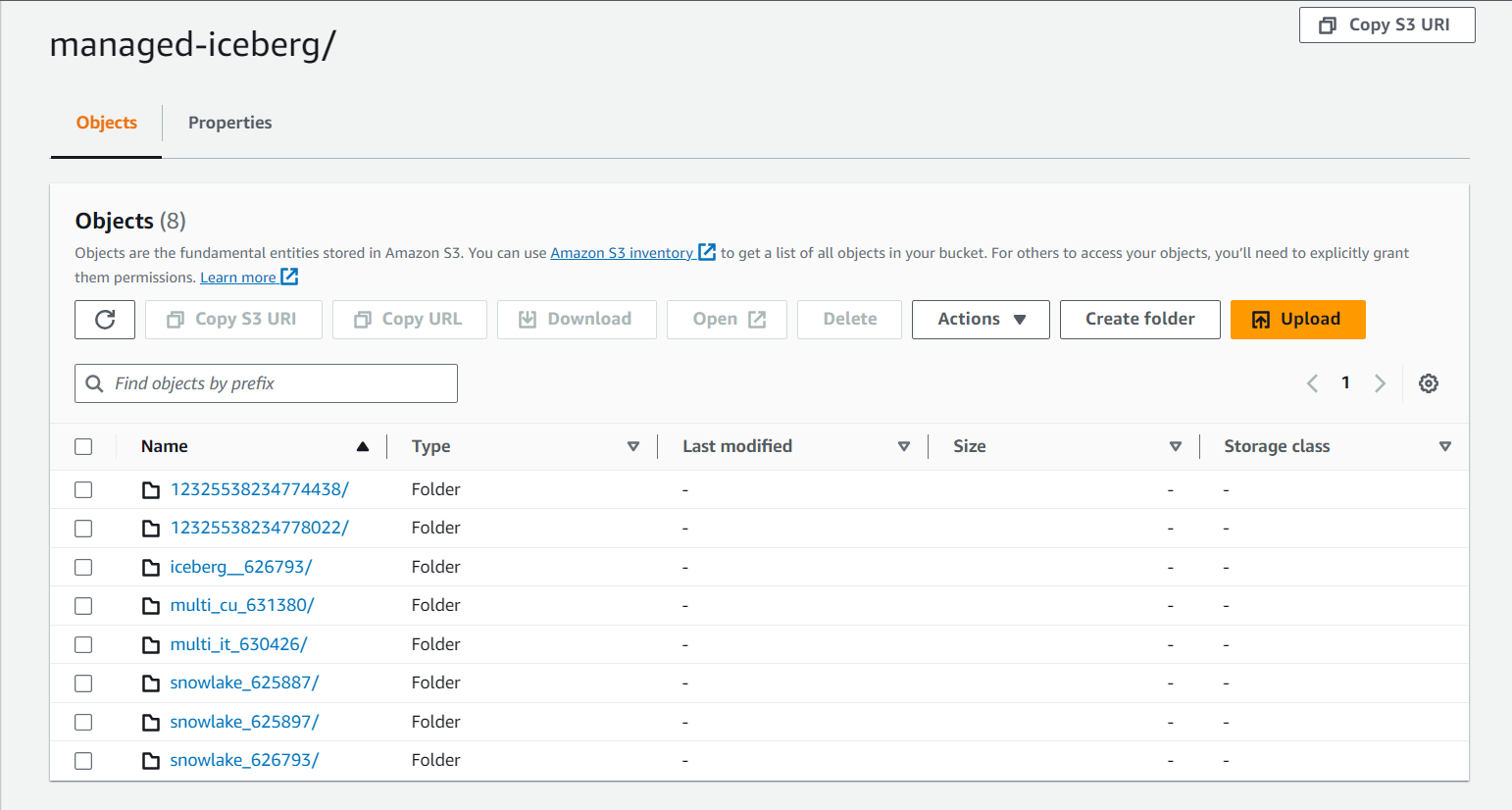
**2.6 Metadata Information**

**Step 5:** To sync the metadata

1)To sync the metadata in aws s3 bucket which we created



2)We would be getting the metadata files in the location specified in the external volume.



3)Run the above query to get the recent metadata snapshot.

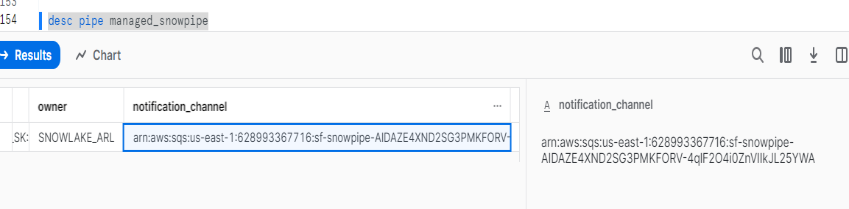
**2.7 Incremental loading**

**Step 6: Incremental load into the iceberg table**

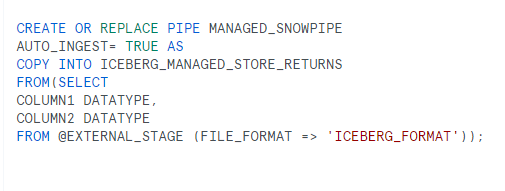
1)As long as a new file is added into the source data location then the data in the new file is automatically ingested into the iceberg table.

2)Create a snow Pipe as managed\_snowpipe with auto ingest as true on the external stage that is used to create the iceberg table.

3)Now describe the pipe and copy the notification channel for event notifications for sqs in s3 bucket.



4)The code snippet below shows,how to create a pipe for incremental loading



5)Check the count of iceberg table after the incremental load.

**2.8 Interoperability**

**Step 7 Interoperability using Spark**

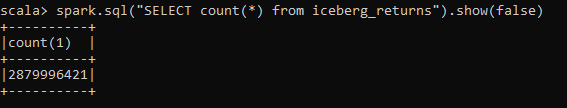
1)Give the required dependencies for iceberg aws and apache spark.

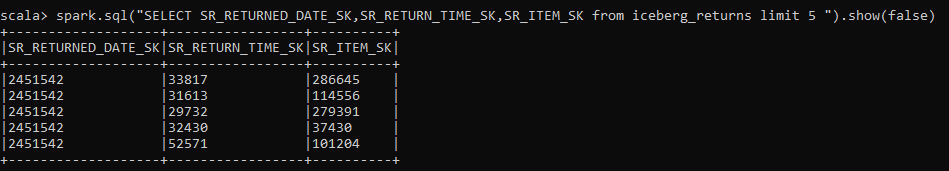
2)Import the required packages to start the spark session

3)Build a spark session by giving the required parameters to connect the hadoop and aws by giving aws credentials.

4)Give the connection parameters for snowflake

5)Create a temporary view with snapshot id and query it.





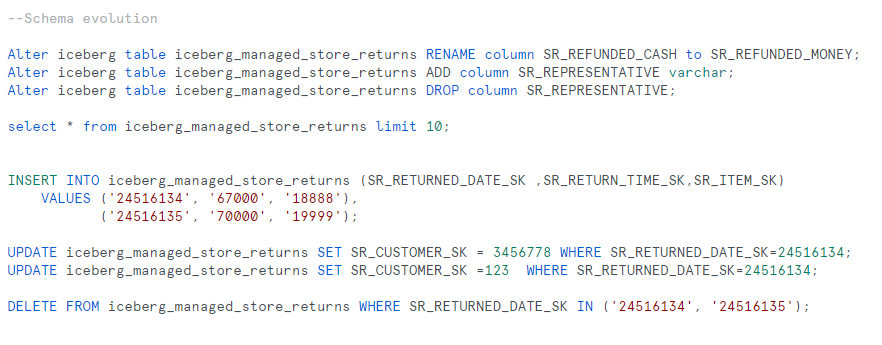
**2.9 Features Applied on iceberg tables**

**2.9.1 Schema Evolution**

**Step 8:** Schema Evolution in snowflake

1)In schema evolution the following operations performed on the managed iceberg table.

* Alter
* Update
* Drop
* Delete
* Rename



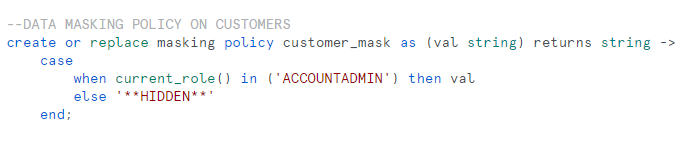
2)In order to sync the metadata and get the recent snapshot id,we need to perform the query



**2.9.2 Data Masking Policy and RBAC**

**Step 9: Data Masking Policy**

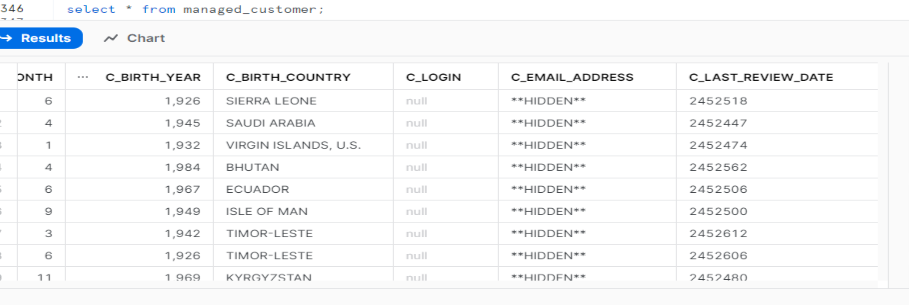
1)Create a masking policy to hide the email address of the customer table for any role other than accountadmin



2) Set the masking policy to the managed iceberg table



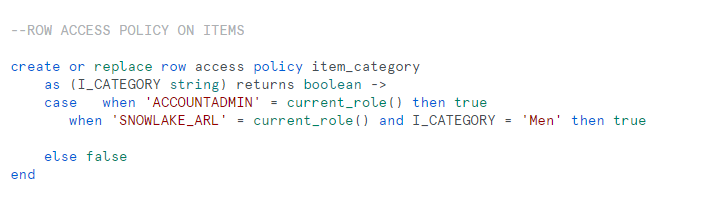
3) After setting the masking policy to the table and accessing the table, the result will be



**Step 10: Row Access Policy**

1) In the item table we are having different categories like men,women,music and sports.

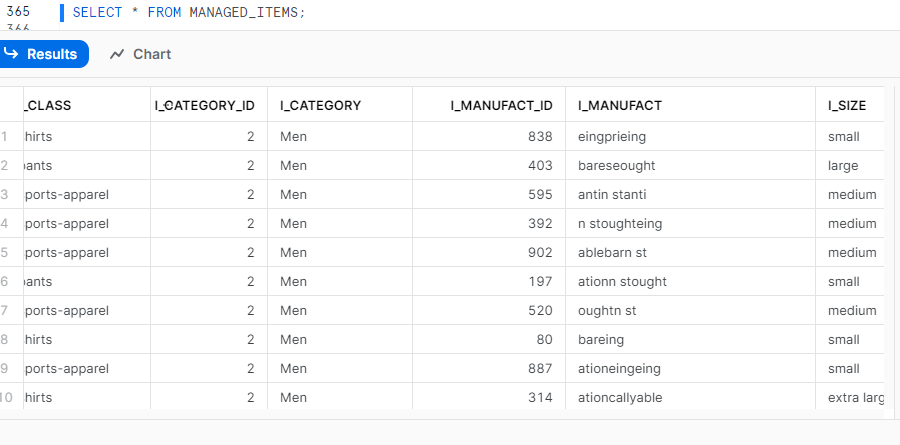
2) We are setting row access policy on category men to get the rows which are having men as category.



3)Set the row access policy on the iceberg table.



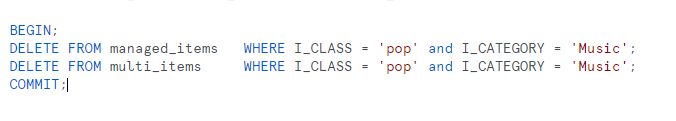
4) After setting the row access policy to the table and accessing the table, the result will be



**2.9.3 Multi Table Transaction**

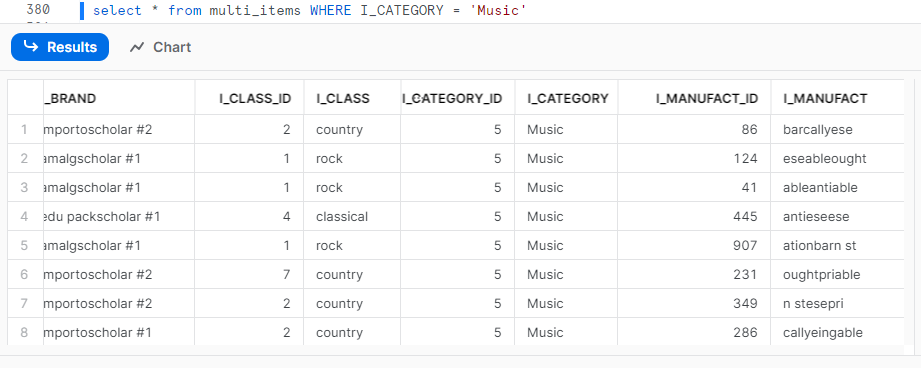
**Step 11: Multi Table Transactions**

1)Snowflake managed iceberg tables support multi table transaction where we can delete the data from two iceberg tables.



2)After running the above query, the tables manged\_items and multi\_items does not contain the rows where I\_class is pop and I\_category is music.

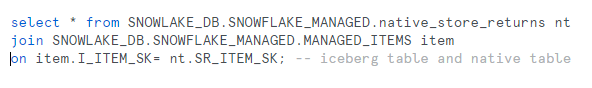
3)The data in the table will be as shown below.



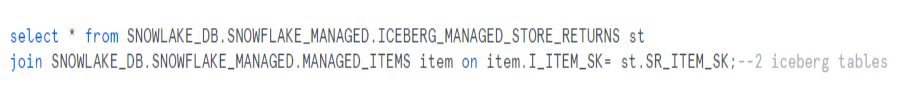
**2.9.4 Joins**

**Step 12: Creating Joins**

1)Joining a snowflake native table with managed iceberg table.



2)Joining two snowflake managed iceberg tables.

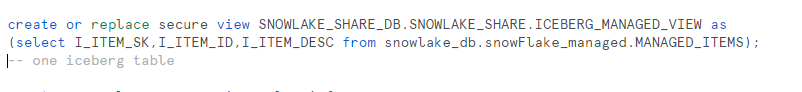


**2.9.5 Secure Data Share**

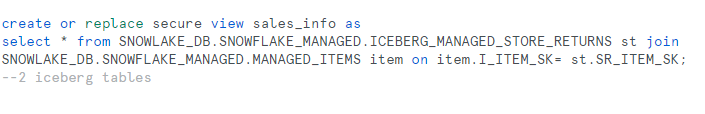
**Step 13: Sharing**

1)Created a database and schema for sharing purposes.

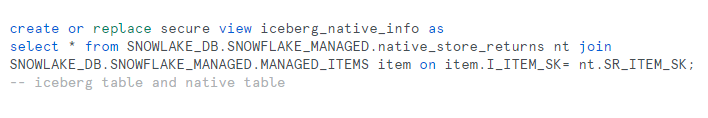
2)Created a secure view called ICEBERG\_MANAGED\_VIEW with only one managed iceberg table MANAGED\_ITEMS.



3)Created a secure view called SALES\_INFO with two managed iceberg tables called ICEBERG\_MANAGED\_STORE\_RETURNS and MANAGED\_ITEMS.



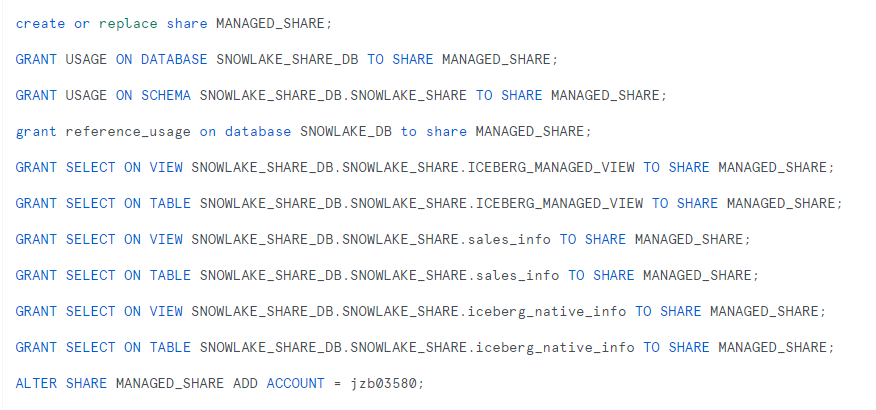
4)Created a secure view called ICEBERG\_NATIVE\_INFO with one managed iceberg table called ICEBERG\_MANAGED\_STORE\_RETURNS and a snowflake native table called native\_store\_returns.



5)Created a share called managed\_share.

6)Grant permissions for share to use database, schema and select permissions on table and view.

7) Add the account locator for alter share command to share the secure views.



**3 STEPS FOLLOWED FOR EXTERNAL TABLE CREATION**

**3.1 Create integration, external stage, external table**

* Creating storage integration with following steps

IAM --> ROLES --> Create Role

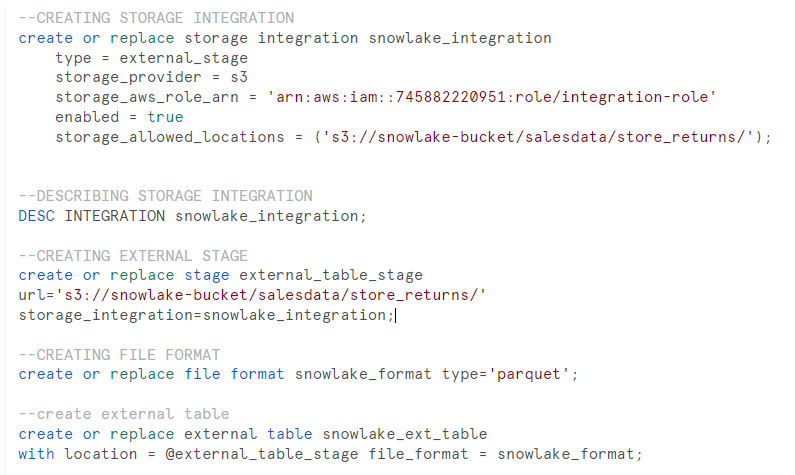
Select policy --> AmazonS3FullAccess

Give a role name --> Select Create Role

Copy the ARN value --> Open snowflake --> Create storage integration.

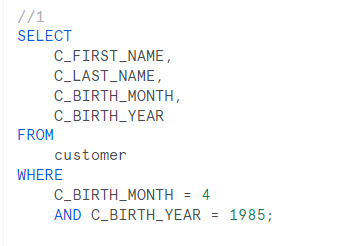
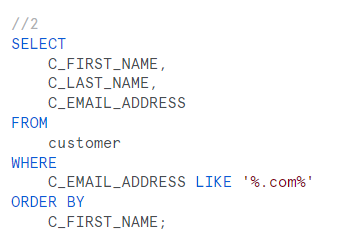
Give the storage\_allowed \_locations as the location where the source data for the iceberg table is present i.e., the location of store\_returns folder.

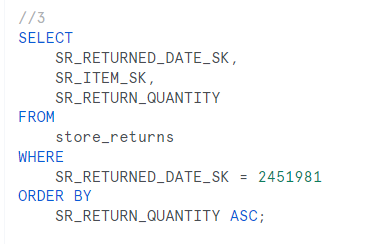
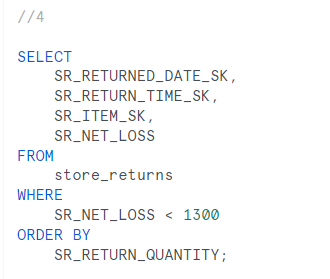
Create External stage as well as create external table using that stage as location and add file format



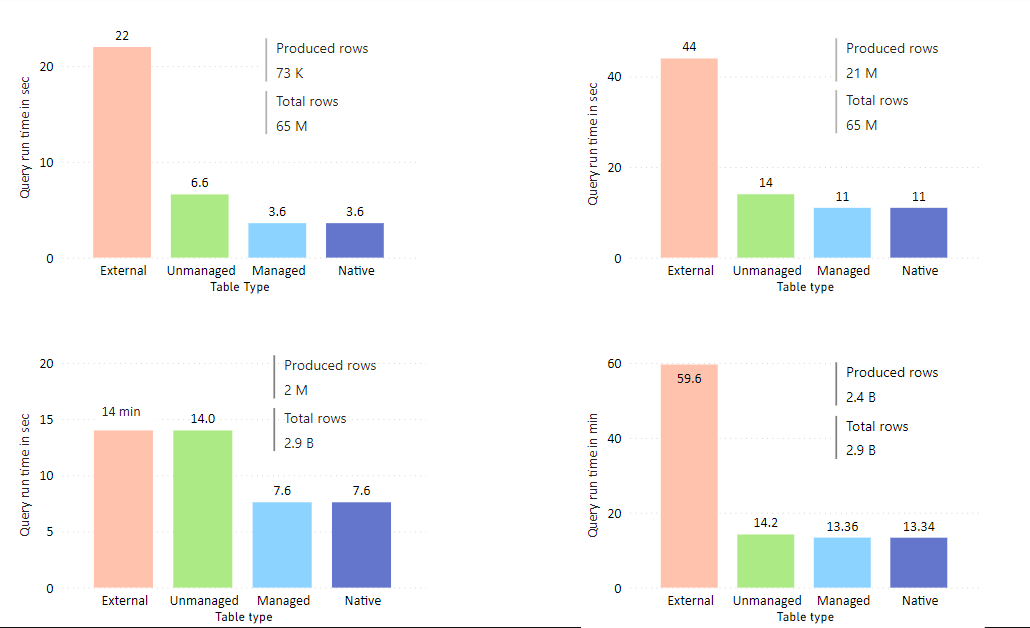
**4 PERFORMANCE PARITY**

**4.1 Queries**

**4.2 Metrics**



**5 REFERENCES**

<https://docs.snowflake.com/LIMITEDACCESS/iceberg-2023/tables-iceberg-getting-started>