National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Nikhil Kumar Singh
Student ID: 19202491

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nikhil Kumar Singh
Student ID:	19202491
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	16/12/2021
Project Title:	Configuration Manual
Word Count:	7836
Page Count:	25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Contents

Configuration Manual

Nikhil Kumar Singh
19202491

1 Overview:

This paper contains a comprehensive, step-by-step guide for implementing this research study. It will detail all of the steps I took to accomplish my study and how those actions may be repeated.

2 Accounts:

For this research, I created 4 trail accounts (Snowflake, AWS, Azure and GCP) and the steps for creating their trial accounts are mentioned in the further sections.

For the implementation of the infrastructure for the project, I took some inspiration majorly from Khedekar & Tian (2020), Wang et al. (2021) and from other papers as well including Wan et al. (2021) and Xin et al. (2019)

2.1 Snowflake Trial Account:

Please have your contact information (name, email, company/business name, and country) available before beginning this step, since you will be utilizing these throughout the sign-up process. Your email address will be used to deliver a link to your Snowflake instance, so ensure it is accurate. You do not need to submit payment information at this time, but you will be required to do so after your trial period ends (after 30 days).

To begin, let's create a Snowflake account. The Snowflake website has an intuitive user interface for establishing and maintaining your account. Additionally, it provides a free-to-use (for 30 days) account with a 400USD compute credit.

Go to the Snowflake website, www.Snowflake.com, and look for the START FOR FREE button. Then, on that page, click the START FOR FREE option to begin the provisioning process.

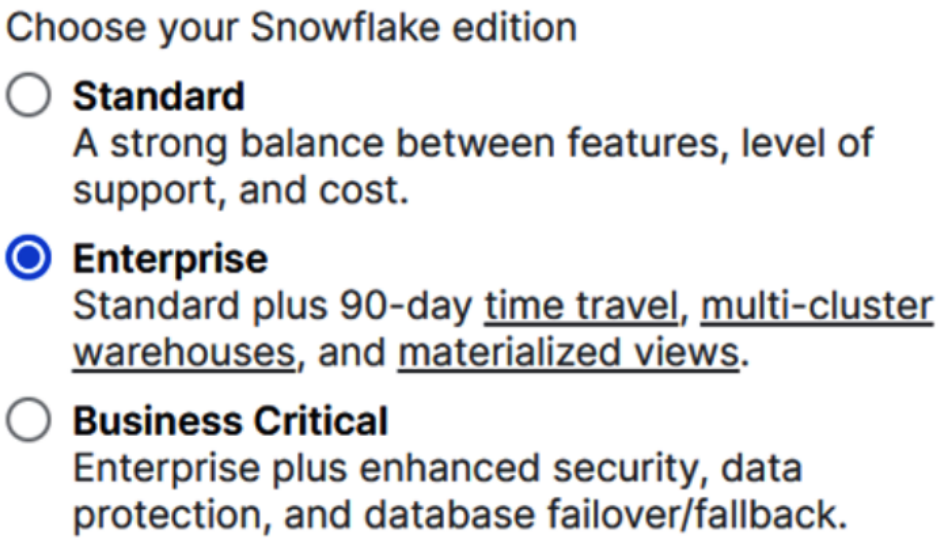


Figure 1: Types of Instance

You will be sent to a register page, where you may begin your 30-day trial with 400USD in credit. Please complete the needed contact information on this page and proceed.

The following steps outline the various choices for instance type and cloud platform selection. These two choices are crucial from a cost and efficiency standpoint.

On the next page, you'll be invited to pick your Snowflake edition and public cloud provider, as well as the location in which your Snowflake instance will be hosted. Which Snowflake edition you choose is mostly determined by your use case. Standard is the entry-level version and includes all required SQL warehouse capability. It does not, however, allow multi-cluster virtual warehouses or materialized views, and supports just one day of time travel (Snowflake's method of updating data versions as new data arrives). Enterprise edition is a suitable fit for the majority of enterprises because to its support for multi-cluster virtual warehouses, materialized views, and time travel up to 90 days. As seen in the following screenshot, we chose Enterprise as shown in Figure 1:

Business Critical has various new security measures and expanded failover capabilities, ensuring that your business maintains more continuity.

A critical point to remember is that Snowflake is a Software as a Service (SaaS), which means that regardless of the public cloud platform you choose, you will access your Snowflake instance through a URL and will not be required to log into your cloud provider's dashboard. Snowflake's service is available on all three main public cloud vendors: Amazon Web Services (AWS), Microsoft Azure (Microsoft Azure), and Google Cloud Platform (Google Cloud Platform) (GCP). These three platforms are broadly comparable in terms of capabilities, with certain limitations imposed by the platform

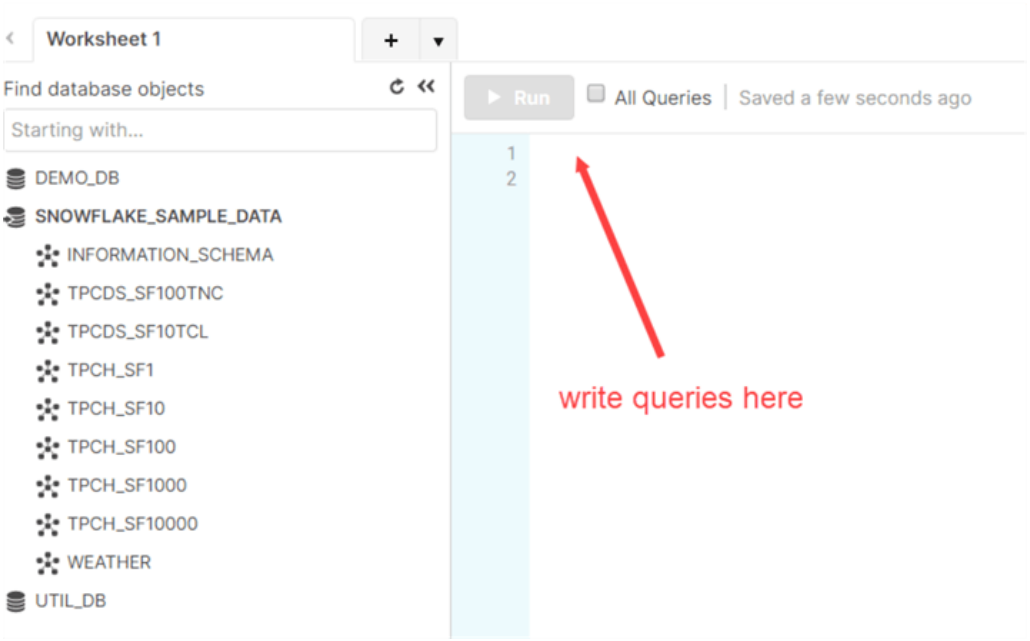


Figure 2: Snowflake Workspace

design.

Which public cloud you pick is critical when considering which public cloud hosts the remainder of your apps and data. Generally, it is preferable to use the same public cloud and area as the rest of your data, since this results in reduced overall costs and improved performance for your firm.

We've chosen AWS as the platform here. Unless otherwise noted, all of the samples supplied operate on AWS. After making that selection, go to the following page, where you will receive a notification indicating that Account Setup is in Progress. Once the instance has been created, this will change to a success message. You will get an activation email with instructions on how to activate your account and create a temporary username and password:

The activation email contains an activation link that will activate your account and take you to a site where you can establish a username and password. This username will serve as the Account Administrator for your instance, so please pick wisely and safeguard your login and password. Once your username and password have been configured, you will be sent to the Snowflake webUI.

After this, you'll be able to login to the snowflake account.

Click Get Started to get to your personal workspace as shown 2:

You'll have a unique URL with your account name in it to access your Snowflake environment as shown in figure 3:

During the account set-up, some sample databases were provided as well as shown in

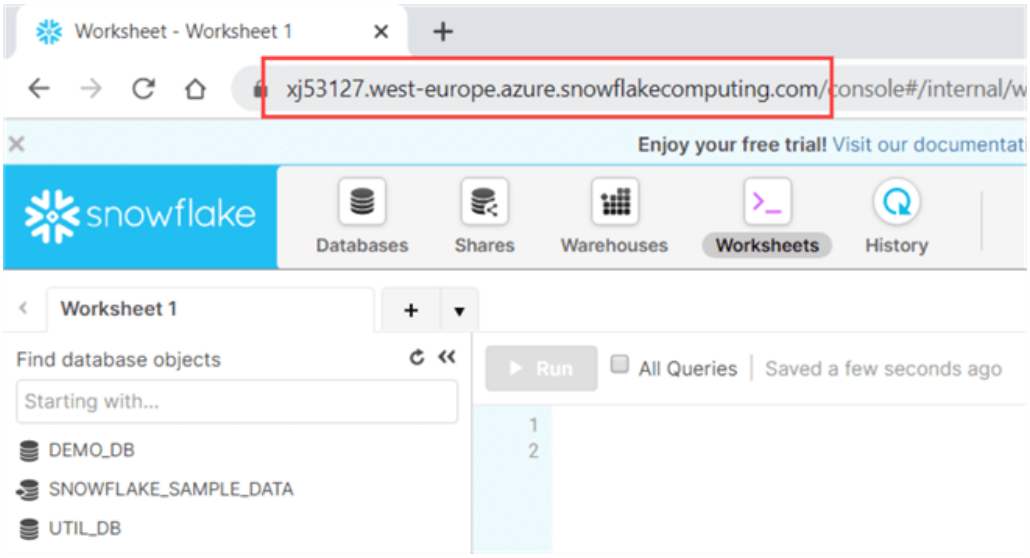


Figure 3: URL to access the workspace

figure **??**:

You are currently signed in with the SYSADMIN role. To access and modify account information, however, you must first move to the ACCOUNTADMIN position, which is the super admin role as shown in figure 5:

2.2 AWS, Azure and GCP Trial Accounts

In the similar fashion accounts need to be activate for Azure, AWS and GCP. The links are added in the footnotes.

- AWS trial account setup steps [1]
- Azure trial account setup steps [2]
- GCP trial account setup steps [3]

3 Collecting data-sets:

For this research, there is a requirement to upload huge data-set on the cloud. Now, you could use any dataset large enough. For my research I used few public datasets from kaggle and combined them together to have a big dataset of about 14GB in size.

[1]https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/
[2]https://k21academy.com/microsoft-azure/create-free-microsoft-azure-trial-account/
[3]https://k21academy.com/google-cloud/create-google-cloud-free-tier-account/

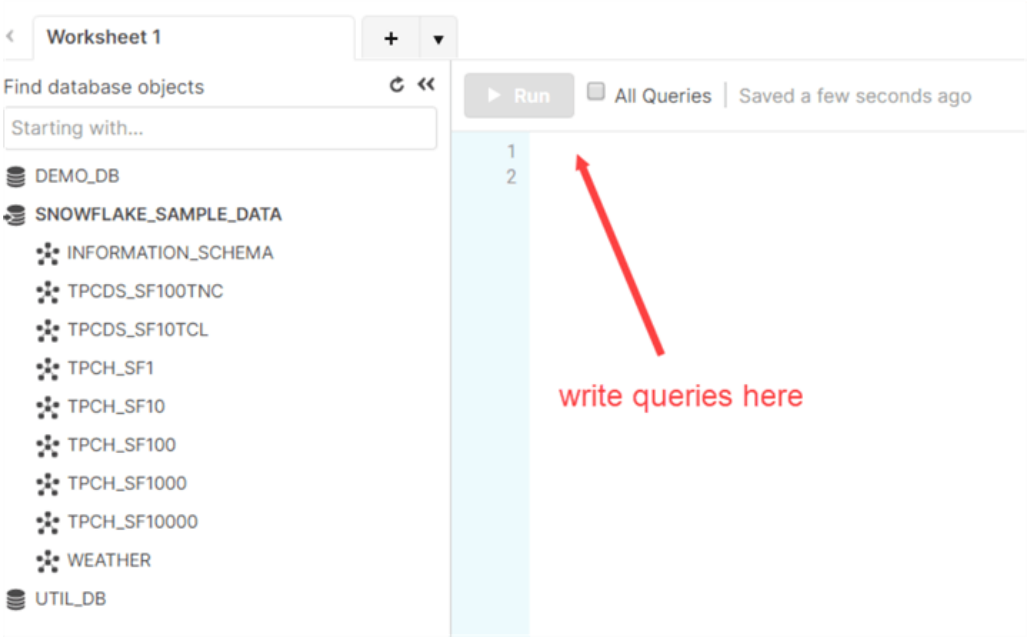


Figure 4: Worksheet and default databases

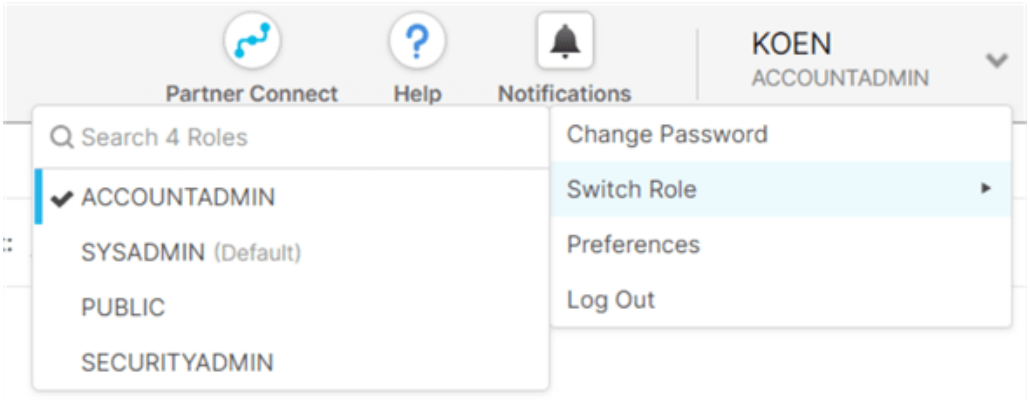


Figure 5: Roles in snowflake

Following are the datasets that I used for my research.

1. F1 2020 race data [4]
2. Brazilian E-Commerce Public Dataset by Olist [5]
3. Bitcoin Historical Data [6]
4. The Movies Dataset [7]
5. Trending YouTube Video Statistics [8]
6. 120 years of Olympic history: athletes and results [9]
7. 5000 Movie Dataset [10]

4 Uploading Datasets to the cloud:

The same datasets then needs to be uploaded to three cloud storage's and the steps to do that are mentioned in this section.

4.1 AWS:

To upload the files to a bucket on Amazon S3

- Create an Amazon S3 bucket.
- Log in to the AWS Management Console and go to https://console.aws.amazon.com/s3/ to access the Amazon S3 console.
- To create a bucket, click Create Bucket.
- Type a bucket name in the Create a Bucket dialog box's Bucket Name field.
- The bucket name you choose must be unique among all bucket names already in use in Amazon S3. One strategy for ensuring uniqueness is to prefix your bucket names with your organization's name. Bucket names must adhere to specific guidelines. For further details, see the Amazon Simple Storage Service User Guide's section on bucket constraints and limitations.

[4]https://www.kaggle.com/coni57/f1-2020-race-data
[5]https://www.kaggle.com/olistbr/brazilian-ecommerce?select=olist$_order_payments_dataset.csv$
[6]https://www.kaggle.com/mczielinski/bitcoin-historical-data
[7]https://www.kaggle.com/rounakbanik/the-movies-dataset?select=ratings.csv
[8]https://www.kaggle.com/datasnaek/youtube-new
[9]https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results
[10]https://www.kaggle.com/tmdb/tmdb-movie-metadata

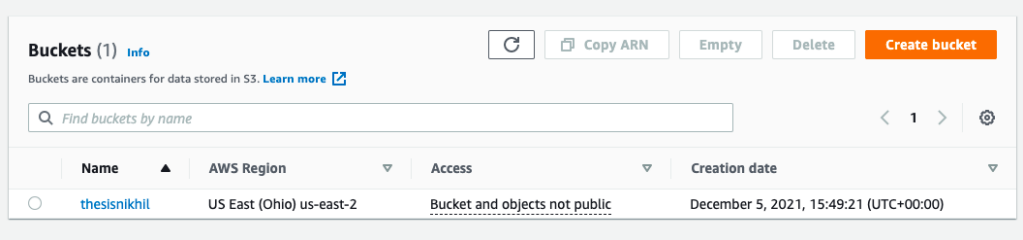


Figure 6: Bucket in AWS S3

- Choose a Region.
- Create the bucket in the region where your cluster is located. If your cluster is located in the United States of America's West (Oregon) Region, choose US West (Oregon) Region (us-west-2).
- Select Create.
- When Amazon S3 successfully builds your bucket, the console's Buckets section shows your empty bucket.
- Create a folder and Choose the name of the new bucket.
- Select the Actions button and from the drop-down box, select Create Folder.
- Load a new folder with a name.

Upload the data files to the new Amazon S3 bucket.

- Choose the name of the data folder.
- In the Upload - Select Files wizard, choose Add Files.
- Follow the Amazon S3 console instructions to upload all of the files you downloaded and extracted,
- Choose Start Upload.

4.2 Azure:

The steps to upload data on Azure are slightly different.

It Azure, you'll first need to create a storage account as shown in Figure 7

After creating a storage account, we will then need to create a container inside that storage account as shown in Figure 8

Then we can create a folder and upload the dataset in that folder as shown in Figure 9

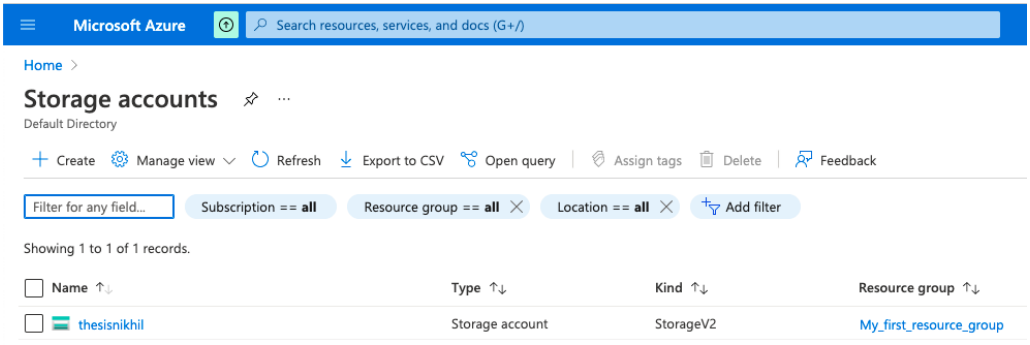


Figure 7: Azure Storage Account

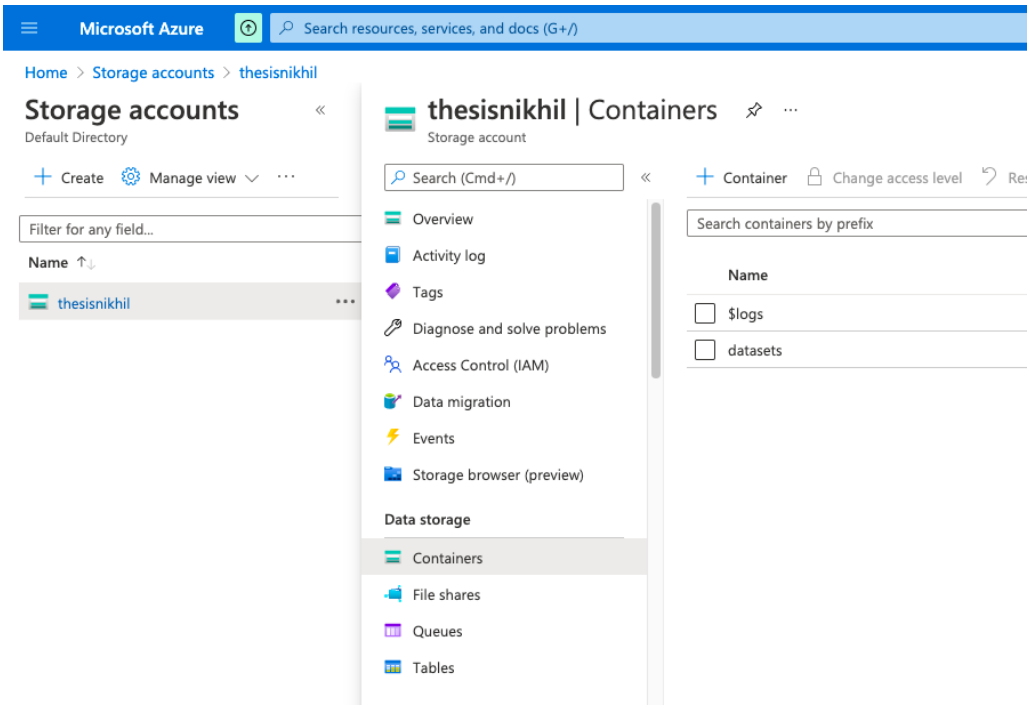


Figure 8: Container in Azure

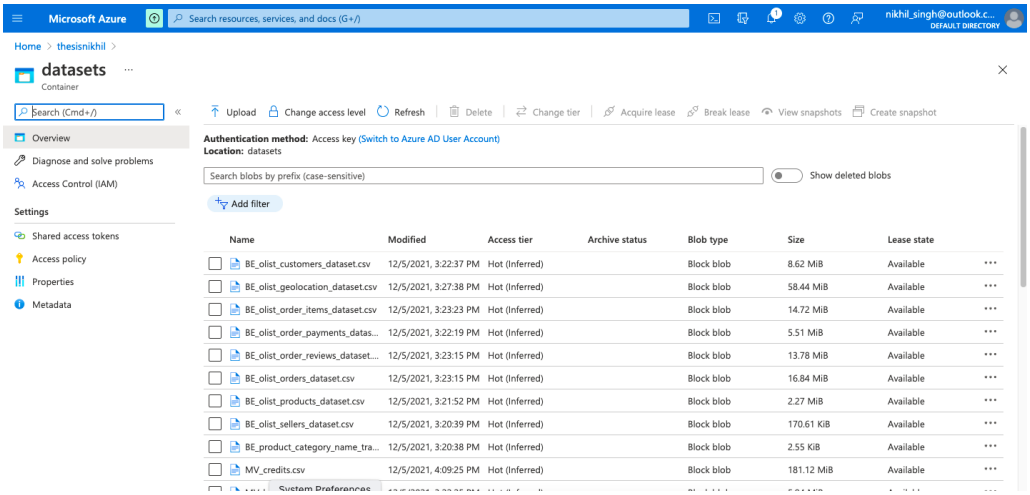


Figure 9: Dataset uploaded in azure

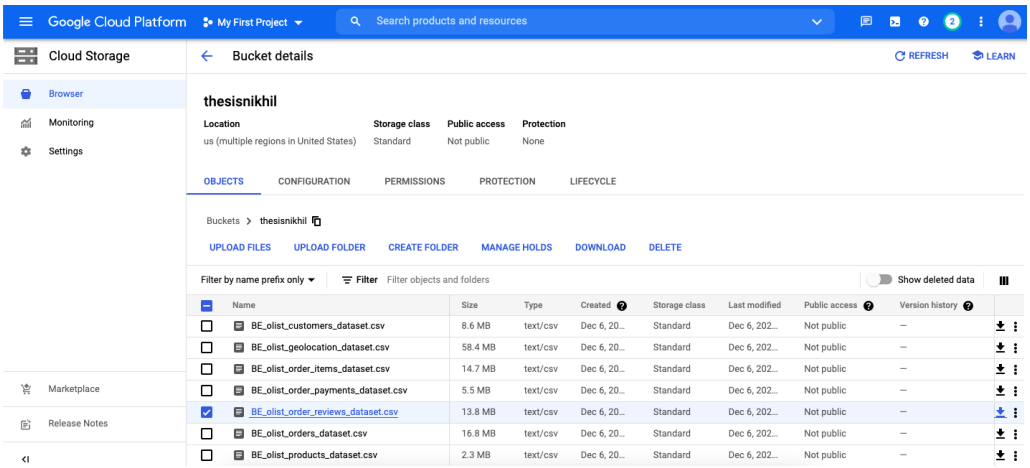


Figure 10: Dataset uploaded in GCP

4.3 GCP:

Just like AWS and Azure, you can create a folder in GCP cloud storage and upload the dataset there as well as shown in the figure.

- Login to GCP account.
- Open the cloud storage browser in GCP console.
- Click on the CREATE BUCKET option.
- Enter the requested details and click on Done.

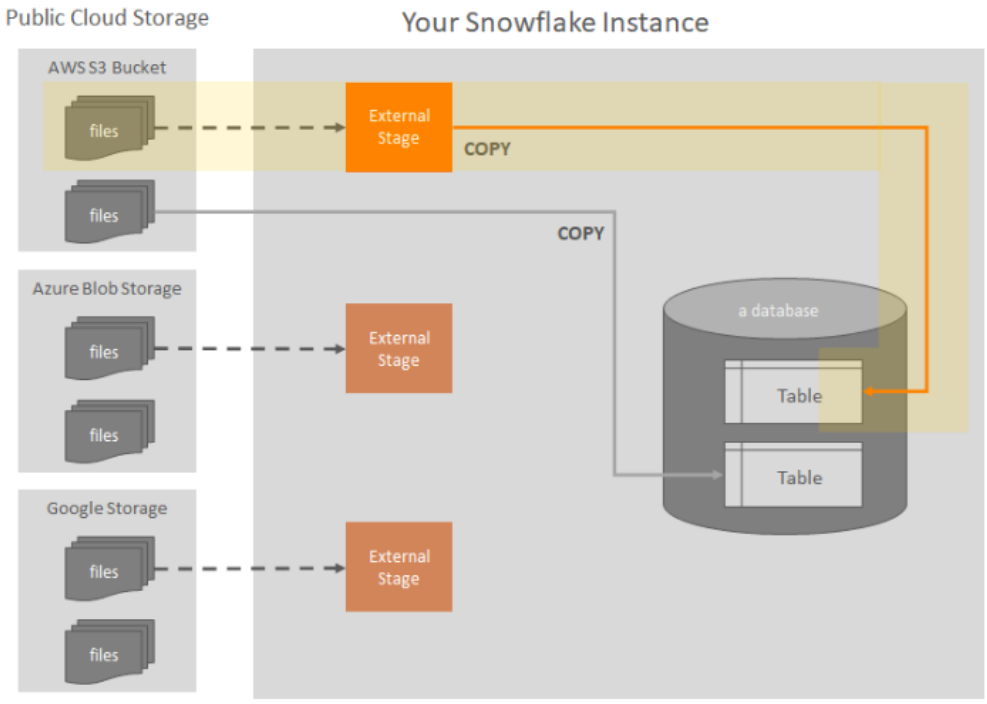


Figure 11: Loading from cloud storage via external stage

5 Loading data into snowflake:

In this section, we will see how the data from cloud storage can be loaded into the snowflake application.

Figure 11 shows the architecture of how the data is loaded into the snowflake application from the three sources.

5.1 Loading Data from cloud into snowflake:

We need to create a policy so that the data on the cloud can be accessed by the Snowflake application.

Overall it's a 4 step process to load the data into S3 and into the snowflake from S3. The details steps are mentioned in the official snowflake documentation

Basically, it's a 3-4 step process depending on the cloud service provider and the links for the official snowflake documentations are listed below.

- Bulk loading from Amazon S3 [11].

[11] https://docs.snowflake.com/en/user-guide/data-load-s3.html

```
// Create storage integration object

create or replace storage integration s3_int
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = S3
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN = ''
  STORAGE_ALLOWED_LOCATIONS = ('s3://<your-bucket-name>/<your-path>/', 's3://<your-bucket-name>/<your-path>/')
   COMMENT = 'This an optional comment'


// See storage integration properties to fetch external_id so we can update it in S3
DESC integration s3_int;
```

Figure 12: AWS Integration Object

Create Stage.txt

```
---- Create file format & stage objects ----

-- create file format
create or replace file format demo_db.public.fileformat_azure
    TYPE = CSV
    FIELD_DELIMITER = ','aws
    SKIP_HEADER = 1;

-- create stage object
create or replace stage demo_db.public.stage_aws
    STORAGE_INTEGRATION = aws_integration
    URL = 'aws://storageaccountsnow.blob.core.windows.net/snowflakecsv'
    FILE_FORMAT = fileformat_aws;


-- list files
LIST @demo_db.public.stage_aws;
```

Figure 13: Creating a stage object

- Bulk loading from Microsoft Azure [12].
- Bulk loading from Google GCP [13].

Next we create an integration and staging objects to load the files into a stage in snowflake from where we can later load in into the database.

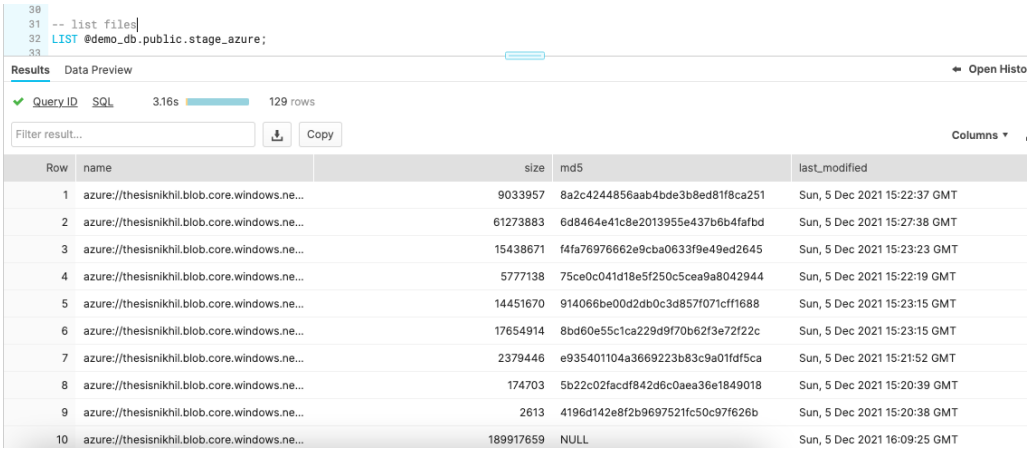
Figure 12 shows the commands to create the integration object before being able to access the data in snowflake.

Next we create a stage object as shown in the figure 13

Next, we list the stage object and see the files loaded into the stage from the S3 bucket using the command "LIST @demo_db.public.stage_aws;"

Now that we see the list of files as shown in figure 14.

[12]https://docs.snowflake.com/en/user-guide/data-load-azure.html
[13]https://docs.snowflake.com/en/user-guide/data-load-gcs.html



```
30
31 -- list files
32 LIST @demo_db.public.stage_azure;
33
```

Results | Data Preview | Open Histo

Query ID | SQL | 3.16s | 129 rows

Filter result... | Copy | Columns

Row	name	size	md5	last_modified
1	azure://thesisnikhil.blob.core.windows.ne...	9033957	8a2c4244856aab4bde3b8ed81f8ca251	Sun, 5 Dec 2021 15:22:37 GMT
2	azure://thesisnikhil.blob.core.windows.ne...	61273883	6d8464e41c8e2013955e437b6b4fafbd	Sun, 5 Dec 2021 15:27:38 GMT
3	azure://thesisnikhil.blob.core.windows.ne...	15438671	f4fa76976662e9cba0633f9e49ed2645	Sun, 5 Dec 2021 15:23:23 GMT
4	azure://thesisnikhil.blob.core.windows.ne...	5777138	75ce0c041d18e5f250c5cea9a8042944	Sun, 5 Dec 2021 15:22:19 GMT
5	azure://thesisnikhil.blob.core.windows.ne...	14451670	914066be00d2db0c3d857f071cff1688	Sun, 5 Dec 2021 15:23:15 GMT
6	azure://thesisnikhil.blob.core.windows.ne...	17654914	8bd60e55c1ca229d9f70b62f3e72f22c	Sun, 5 Dec 2021 15:23:15 GMT
7	azure://thesisnikhil.blob.core.windows.ne...	2379446	e935401104a3669223b83c9a01fdf5ca	Sun, 5 Dec 2021 15:21:52 GMT
8	azure://thesisnikhil.blob.core.windows.ne...	174703	5b22c02facdf842d6c0aea36e1849018	Sun, 5 Dec 2021 15:20:39 GMT
9	azure://thesisnikhil.blob.core.windows.ne...	2613	4196d142e8f2b9697521fc50c97f626b	Sun, 5 Dec 2021 15:20:38 GMT
10	azure://thesisnikhil.blob.core.windows.ne...	189917659	NULL	Sun, 5 Dec 2021 16:09:25 GMT

Figure 14: List of files in external stage

Next we create a table in which we will load the data from this stage and then we would be able to access the contents.

The similar steps needs to be carried out for azure and GCP and the queries for the same are attached in the artifacts under code/loading_data

6 Creating a snowpipe:

6.1 Creating a task:

To create a pipeline we will first create a task in snowflake and the steps are menitoned as follows.

- To simplify the process for you, we've utilized Snowflake's example data and built an aggregate query on top of it. (Please note that sample data is supplied with your Snowflake instance and is located in the database SNOWFLAKE SAMPLE DATA.) On the sample data, we will run the following fictitious query:

```
SELECT C.C_NAME,SUM(L_EXTENDEDPRICE),SUM(L_TAX)
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.CUSTOMER C
INNER JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS O
ON O.O_CUSTKEY = C.C_CUSTKEY
INNER JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.LINEITEM LI
ON LI.L_ORDERKEY = O.O_ORDERKEY
GROUP BY C.C_NAME;
```

- The result for the above query is is shown in figure 15

- Now we'll establish a target table to store the results of this query. Notably, we have added a Reporting Time column to the columns returned by the previous query. Each time we enter data, we will include the current timestamp in this column:

```
CREATE DATABASE task_demo;
USE DATABASE task_demo;
CREATE OR REPLACE TABLE ordering_customers
(
Reporting_Time TIMESTAMP,
Customer_Name STRING,
Revenue NUMBER(16,2),
Tax NUMBER(16,2)
);
```

- We will now create a task using the preceding SQL statement to insert data into the ordering_customers table. To start with, we will configure the task to run every 2 minutes:

```
CREATE TASK refresh_ordering_customers
WAREHOUSE = COMPUTE_WH
SCHEDULE = '30 MINUTE'
COMMENT = 'Update Ordering_Customers Table with latest
 data'
AS
INSERT INTO ordering_customers
SELECT CURRENT_TIMESTAMP, C.C_NAME,
SUM(L_EXTENDEDPRICE), SUM(L_TAX)
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.CUSTOMER C
INNER JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS O
ON O.O_CUSTKEY = C.C_CUSTKEY
INNER JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.LINEITEM LI
ON LI.L_ORDERKEY = O.O_ORDERKEY
GROUP BY CURRENT_TIMESTAMP, C.C_NAME;
```

- Let's validate that the task has been created correctly by running the DESC command:

```
DESC TASK refresh_ordering_customers;
```

- The following figure 16 is the output of the previous code, which has been broken into numerous lines for readability. Notably, a new task is generated by default in a suspended state, as seen in the output:

- If you are executing your code through a role other than ACCOUNTADMIN, you must provide that role the following privileges:

Figure 15: Sample output

Figure 16: DESC Command output

```
USE ROLE ACCOUNTADMIN;
GRANT EXECUTE TASK ON ACCOUNT TO ROLE SYSADMIN;
```

- We now need to change the task's status to Resumed so that it may resume execution on time. This will be accomplished by executing the describe command once again to verify that the task has been properly transferred from the suspended to the begun state. Nota bene, the next step must be performed as ACCOUNTADMIN; otherwise, you may provide the needed permissions to another role, as described in the previous step:

  ```
  ALTER TASK refresh_ordering_customers RESUME;
  DESC TASK refresh_ordering_customers;
  ```

- The accompanying code produces the following output figure 17:

- Note that now, the state of the task is set to started, which means the task is now scheduled and should execute in the next 2 minutes.

- We will now keep an eye on the task execution to validate that it runs successfully. To do that, we need to query task_history, as follows:

  ```
  SELECT name, state,
  completed_time, scheduled_time,
  ```

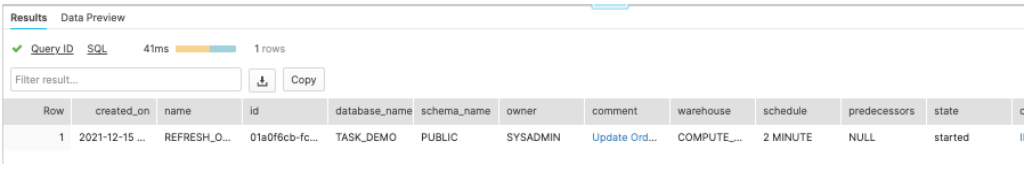


Figure 17: DESC Command output

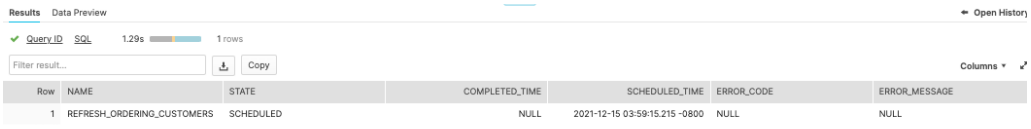


Figure 18: Output of task_history

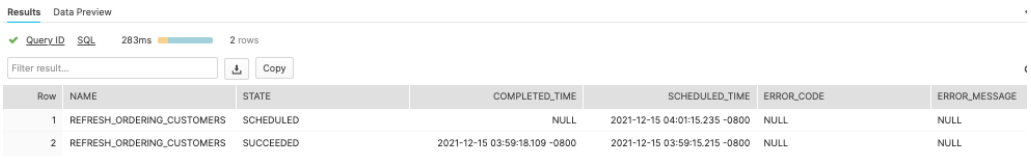


Figure 19: Output of task_history

```
error_code, error_message
FROM TABLE(information_schema.task_history())
WHERE name = 'REFRESH_ORDERING_CUSTOMERS';
```

- The output from the query is shown in figure 18
- As seen by the output, the job has not yet been completed, and the planned time for execution is displayed. In another two minutes, we'll check the status, and this job should have completed.
- Rerun the above query after 2 minutes. As predicted, the job was successfully completed, as shown by the following output 19, and the next instance is scheduled for execution:
- Let's validate that the task has indeed executed successfully by selecting from the ordering_customers table:

  ```
  SELECT * FROM ordering_customers;
  ```

- As shown in the figure 20, the query returned around 100,000 records, which indicates that the task executed successfully:

More examples of creating tasks are present in folder code/tasks as shown in figure

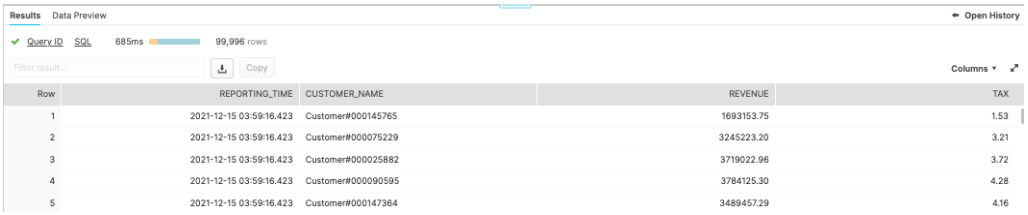


Figure 20: Validating the task

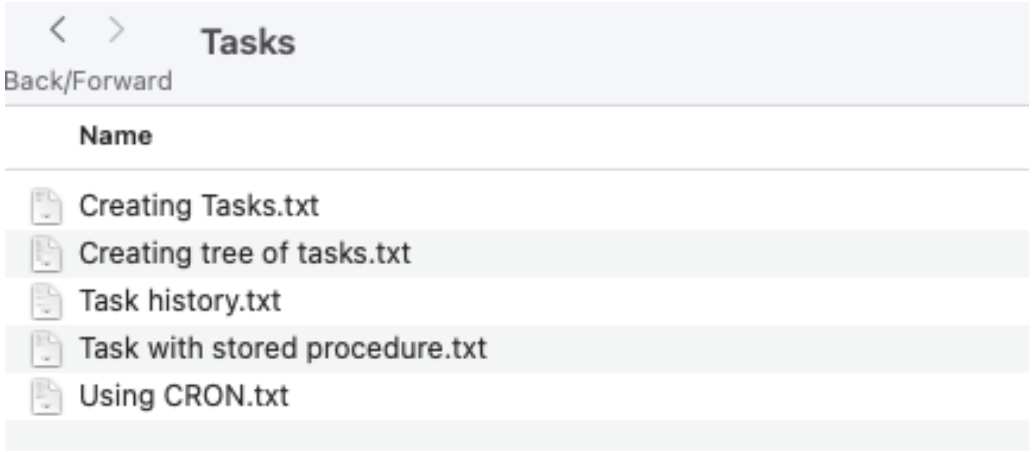


Figure 21: Tasks files

6.2 Creating a stream:

- Let's start by creating a database and a staging table on which we will create our stream object. We create a staging table to simulate data arriving from outside Snowflake and being processed further through a stream object:

```
CREATE DATABASE stream_demo;
USE DATABASE stream_demo;
CREATE TABLE customer_staging
(
ID INTEGER,
Name STRING,
State STRING,
Country STRING
);
```

- The process of creating a stream is quite straightforward with a simple command such as the following:

```
CREATE STREAM customer_changes ON TABLE customer_staging;
```

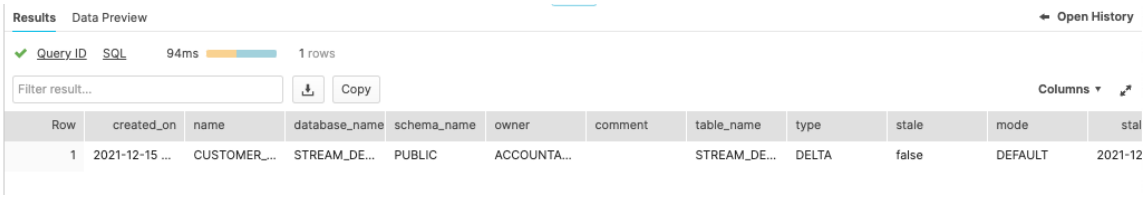
- Let's describe the stream to see what has been created:

```
DESC STREAM customer_changes;
```

- As shown in figure 22 the mode of the stream is set to DEFAULT, which indicates it will track inserts, updates, and deletes that are performed on the table:

- Let's insert some data into the staging table to simulate data arriving into Snowflake:

```
INSERT INTO customer_staging VALUES (1,'Jane
Doe','NSW','AU');
INSERT INTO customer_staging VALUES
(2,'Alpha','VIC','AU');
```



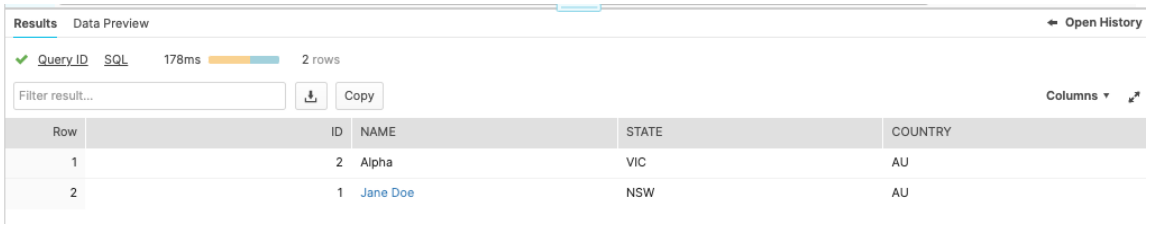
Results Data Preview — Open History

Query ID SQL 94ms 1 rows

Filter result... Copy Columns

Row	created_on	name	database_name	schema_name	owner	comment	table_name	type	stale	mode	stal
1	2021-12-15 ...	CUSTOMER_...	STREAM_DE...	PUBLIC	ACCOUNTA...		STREAM_DE...	DELTA	false	DEFAULT	2021-12

Figure 22: The created stream



Results Data Preview — Open History

Query ID SQL 178ms 2 rows

Filter result... Copy Columns

Row	ID	NAME	STATE	COUNTRY
1	2	Alpha	VIC	AU
2	1	Jane Doe	NSW	AU

Figure 23: Validating whether the data is inserted into the table

- Validate that the data is indeed inserted into the staging table by selecting it from the table:

```
SELECT * FROM customer_staging;
```

- As expected, two rows are present in the staging table as shown in figure 23:
- Now, let's view how the changing data has been captured through the stream. We can view the data by simply selecting from the stream object itself, which is shown as follows :

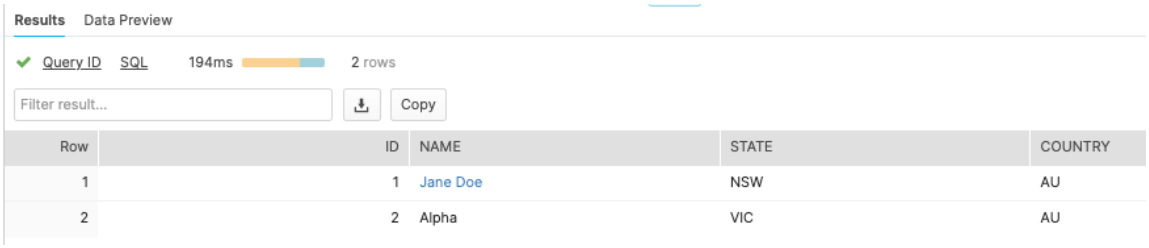
```
SELECT * FROM customer_changes;
```

- The following output (figure 24) shows the data that has been recorded by the stream. Notice both rows have been recorded with an action code of INSERT:
- Now that we have our stream set up successfully, we can process the data from the stream into another table. You would usually do this as part of a data pipeline. Create a table first in which we will insert the recorded data:

```
CREATE TABLE customer
(
```

Row	ID	NAME	STATE	COUNTRY	METADATA$ACT	METADATA$ISUP	METADATA$ROW
1	1	Jane Doe	NSW	AU	INSERT	FALSE	546c1da8c5e...
2	2	Alpha	VIC	AU	INSERT	FALSE	562a8a2d053...

Figure 24: Viewing the changing data



Results Data Preview

Query ID SQL 194ms 2 rows

Filter result... Copy

Row	ID	NAME	STATE	COUNTRY
1	1	Jane Doe	NSW	AU
2	2	Alpha	VIC	AU

Figure 25: Validating the data from the customer table

```
ID INTEGER,
Name STRING,
State STRING,
Country STRING
);
```

- Retrieving data from a stream and inserting it into another table is as simple as performing a query on the stream itself. As shown ahead, the query selects the required columns from the stream and inserts them into the customer table. Do note that we have used a WHERE clause on metadata$action equal to INSERT. This is to ensure that we only process new data:

```
INSERT INTO customer
SELECT ID,Name,State,Country
FROM customer_changes
WHERE metadata$action = 'INSERT';
```

- Let's select the data from the customer table to validate that the correct data appears there:

```
SELECT * FROM customer;
```

- As expected, the two rows that were originally inserted into the staging table appear here as shown in figure 25:
- Let's find out what happens to the stream after data has been processed from it. If we perform SELECT now, there will be zero rows returned since that data has already been processed:

```
SELECT * FROM customer_changes;
```

- The output shows zero rows returned (figure 26):
- Let's update a row in the staging table. We are then going to see in the stream how that update appears:

```
UPDATE customer_staging SET name = 'John Smith' WHERE
ID = 1;
```

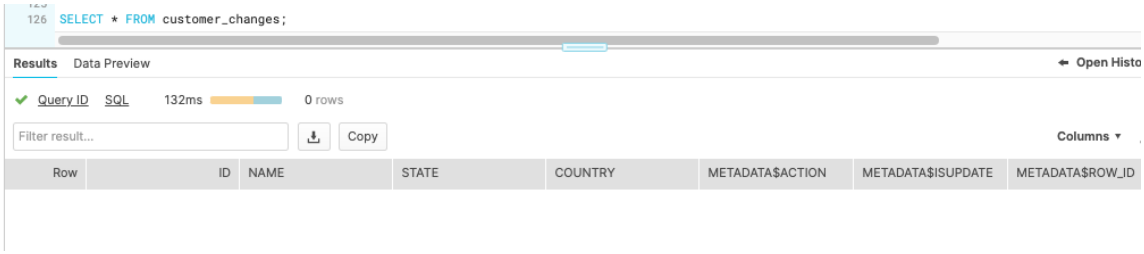


Figure 26: Zero rows returned

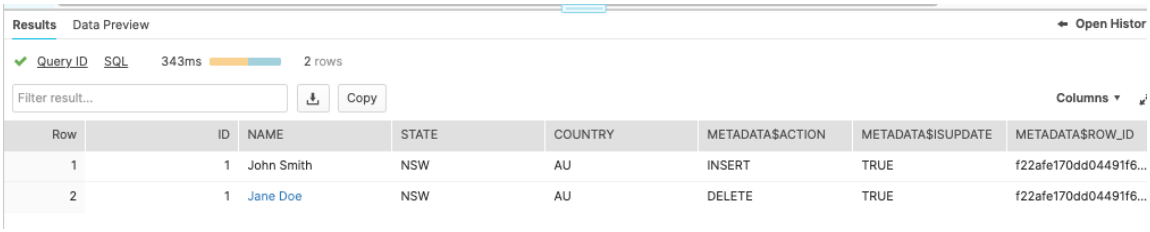


Figure 27: Selecting the data from the stream

- Select the data from the stream:

  ```
  SELECT * FROM customer_changes;
  ```

- You will see two records appear in the result set, as shown in figure 27:
- An update operation is essentially captured as DELETE followed by INSERT. Therefore, you will see both INSERT and UPDATE appear in the result. If you are processing the stream for deletes as well, you will need additional logic in the consuming code to process DELETE correctly.

6.3 Creating snowpipe using task and stream

- Let's start by creating a database and a staging table on which we will create our stream object. We will be creating a staging table to simulate data arriving from outside Snowflake and being processed further through a stream object:

  ```
  CREATE DATABASE stream_demo;
  USE DATABASE stream_demo;
  CREATE TABLE customer_staging
  (
  ID INTEGER,
  Name STRING,
  State STRING,
  Country STRING
  );
  ```

name	database_name	schema_name	owner	comment	table_name	type	stale	mode
CUSTOMER_...	STREAM_DE...	PUBLIC	SYSADMIN		STREAM_DE...	DELTA	false	APPEND_ONLY

Figure 28: The created stream

- Next, create a stream on the table that captures only the inserts. The insert-only mode is achieved by setting APPEND_ONLY to TRUE:

```
CREATE STREAM customer_changes ON TABLE
customer_staging APPEND_ONLY = TRUE;
```

- Let's describe the stream to see what has been created:

```
DESC STREAM customer_changes;
```

- As shown in the following output (figure 28), notice that the mode of the stream is set to APPEND_ONLY, which indicates it will only track inserts:

- So, we have created a staging table and a stream on top of it. Now, we are going to create the actual table into which all the new customer data will be processed:

```
CREATE TABLE customer
(
ID INTEGER,
Name STRING,
State STRING,
Country STRING
);
```

- Let's now create a task that we will use to insert any new data that appears in the stream:

```
CREATE TASK process_new_customers
WAREHOUSE = COMPUTE_WH
COMMENT = 'Process new data into customer'
AS
INSERT INTO customer
SELECT ID,Name,State,Country
FROM customer_changes
WHERE metadata$action = 'INSERT';
```

- Let's schedule this task to run every 1 minutes. We are assuming that new customer data is getting inserted into the staging table all the time and we need to process it every 1 minutes. Please note that to resume a task, you will need to run the command as ACCOUTNADMIN or another role with the appropriate privileges:

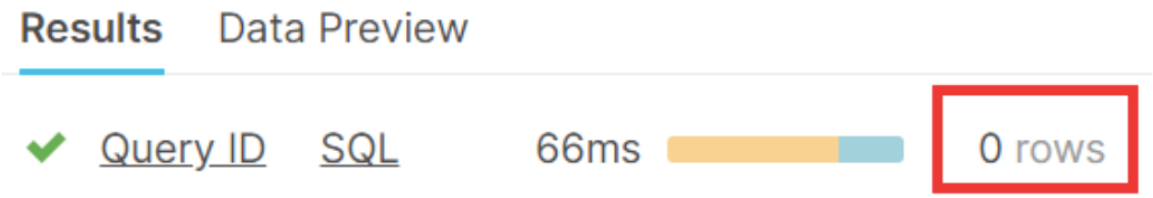


Figure 29: Customer table

ID	NAME	STATE	COUNTRY	METADATA$ACTI	METADATA$ISUP	METADATA$ROW
2	Alpha	VIC	AU	INSERT	FALSE	c9801d993af...
1	Jane Doe	NSW	AU	INSERT	FALSE	49e84564a18...

Figure 30: Viewing the changing data

```
ALTER TASK process_new_customers
SET SCHEDULE = '1 MINUTE';
ALTER TASK process_new_customers RESUME;
```

- Let's check that the target table, that is, customer, is empty:

```
SELECT * FROM customer;
```

- As expected, no rows are present in the customer table as shown in figure 29:
- We will now insert some data into the staging table (effectively simulating data that has arrived into Snowflake from an external source):

```
INSERT INTO customer_staging VALUES (1,'Jane
Doe','NSW','AU');
INSERT INTO customer_staging VALUES
(2,'Alpha','VIC','AU');
```

- Now, let's view how the changing data has been captured through the stream. We can view the data by adding SELECT from the stream object itself:

```
SELECT * FROM customer_changes;
```

- The following output (figure 30 shows the data that has been recorded by the stream. Notice both rows have been recorded with an action code of INSERT:
- Retrieving data from a stream and inserting it into another table is as simple as performing a query on the stream itself. As shown ahead, the query selects the required columns from the stream and inserts them into the customer table. Do note that we have used a WHERE clause on metadata$action set to INSERT. This is to ensure that we only process new data:

ID	NAME	STATE	COUNTRY
2	Alpha	VIC	AU
1	Jane Doe	NSW	AU

Figure 31: Two rows inserted into the staging table

```
INSERT INTO customer
SELECT ID,Name,State,Country
FROM customer_changes
WHERE metadata$action = 'INSERT';
```

- Let's select the data from the customer table to validate that the correct data appears there:

```
SELECT * FROM customer;
```

- As expected, the two rows that were originally inserted into the staging table appear here (figure 31):
- We will now insert some more data into the staging table and let it be processed by the scheduled task:

```
INSERT INTO customer_staging VALUES
(3,'Mike','ACT','AU');
INSERT INTO customer_staging VALUES
(4,'Tango','NT','AU');
```

- Now, we wait for our scheduled task to run, which will process this staging data into the target table. You can also keep an eye on the execution and the next scheduled time by running the following query. Once the scheduled task has executed, the results will look like what is shown as follows:

```
SELECT * FROM
TABLE(information_schema.task_history(
task_name => 'PROCESS_NEW_CUSTOMERS'))
ORDER BY SCHEDULED_TIME DESC;
```

- Once the task has been successfully executed, you will see output similar to the following screenshot (figure 32). The task that has been run successfully will have a QUERY_ID value assigned and a STATE value of SUCCEEDED:

QUERY_ID	NAME	DATABASE_NAM	SCHEMA_NAME	QUERY_TEXT	CONDITION_TE	STATE	ERROR_CODE	ERROR_MESSAC
NULL	PROCESS_N...	STREAM_DE...	PUBLIC	INSERT INT...	NULL	SCHEDULED	NULL	NULL
019a4e7d-0...	PROCESS_N...	STREAM_DE...	PUBLIC	INSERT INT...	NULL	SUCCEEDED	NULL	NULL

Figure 32: Processing the staging data into the target table

ID	NAME	STATE ↓	COUNTRY
4	Tango	NT	AU
2	Alpha	VIC	AU
1	Jane Doe	NSW	AU
3	Mike	ACT	AU

Figure 33: Validating the rows in the staging table

- Once the task has been successfully executed, select the data from the target table to validate that the rows in the staging table have been inserted into the target table:

  ```
  SELECT * FROM customer;
  ```

- You will see two additional records appear in the result set, indicating that the data from the staging table was processed through a combination of tasks and streams and inserted into the target table (figure 33):

7 Access Management:

Figure 35 shows the roles in snowflake application.

The detailed steps for working with each of the roles are documented and present inside code/Access_management folder (figure 35).

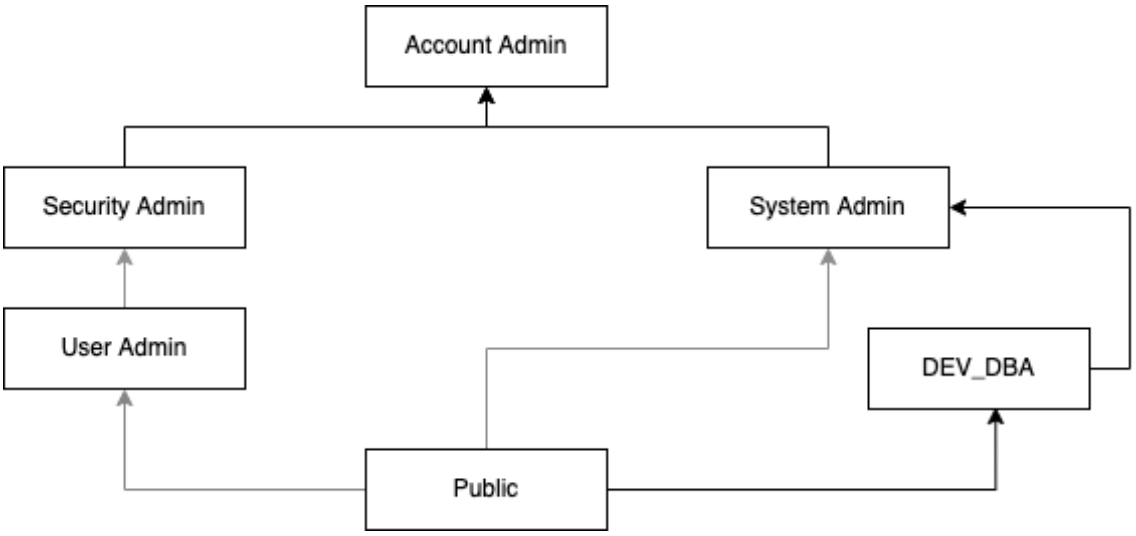


Figure 34: Role Hierarchy

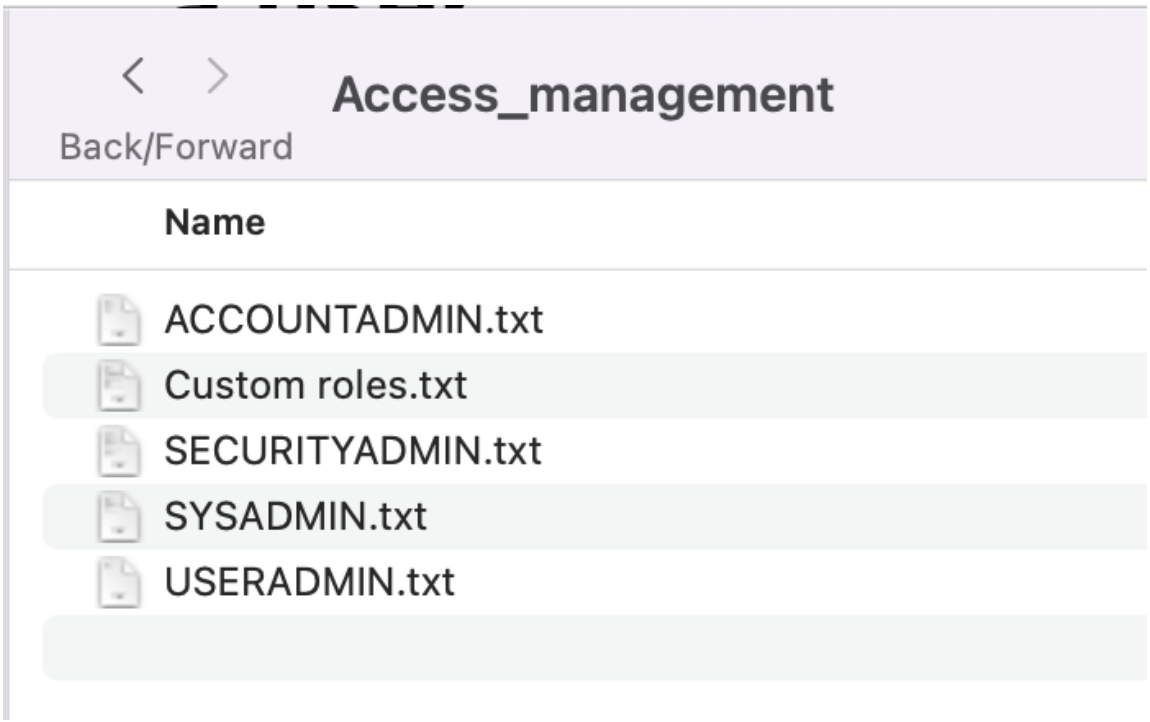


Figure 35: code/Access_management

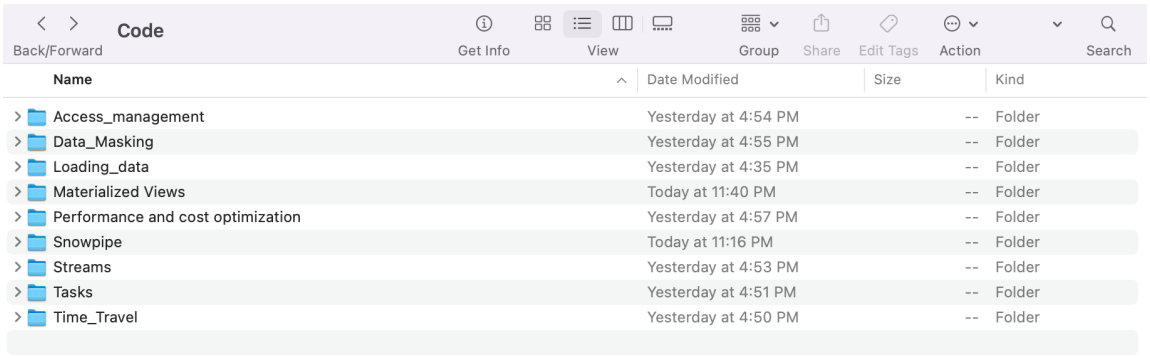


Figure 36: Folder Structure

Their are numerous other parts of the implementation as well and they are shared in the artifacts as shown in figure 36. Also, I would be covering some important tasks in the demo video as well.

References

Khedekar, V. & Tian, Y. (2020), 'Multi-tenant big data analytics on aws cloud platform'.

Wan, W., Du, X., Zhao, X. & Yang, Z. (2021), 'A cloud-enabled collaborative hub for analysis of geospatial big data'.

Wang, F., Wang, H. & Xue, L. (2021), 'Research on data security in big data cloud computing environment'.

Xin, Li, J. & Guo (2019), 'Research on ship data big data parallel scheduling algorithm based on cloud computing', *Journal of Coastal Research* **94(sp1)**, 535–539.