HANDS-ON LAB GUIDE FOR Build Agile Modern Data Applications with Snowflake and AWS

To be used with the Snowflake free 30-day trial at:
https://trial.snowflake.com

Works for any Snowflake edition or cloud provider
Approximate duration: 90 minutes. Approximately 7 credits used

Table of Contents

Lab Overview
Module 1: Prepare Your Lab Environment
Module 2: The Snowflake User Interface & Lab “Story”
Module 3: Preparing to Load the Data and Loading the Data
Module 4: Deploy CodePipeline and Serverless application
Module 5: Test the application and deploy second version
Module 6: Fix the faulty deployment
Module 7: Bonus: Test concurrency
Summary & Next Steps
Cleaning-up

Lab Overview

This lab introduces you to the devops and data apps capabilities of Snowflake and Amazon CodePipeline, and is designed specifically for use with the Snowflake and Amazon services, free 30-day trial at https://trial.snowflake.com and AWS Trial account.

Target Audience

Database engineers and Data Warehouse Administrators and cloud engineers, devops engineers

What you'll learn

The exercises in this lab will walk you through the steps to:

- Ingest structured and semi-structured data for Snowflake powered REST application
- Deploy an Amazon CodePipeline
- Configure Snowflake with SQL code over CICD pipeline
- Deploy API gateway and serverless app with Lambda via CICD and infrastructure as a code
- Learn how Snowflake fits into Data Application and DevOps deployments
- Join the on-demand Q&A with your Snowflake and Amazon peers

Prerequisites

- Use of the Snowflake free 30-day trial environment
- Basic knowledge of SQL, and database concepts and objects
- Familiarity with CSV comma-delimited files and JSON semi-structured data
- Use of an AWS Account with the ability to launch a CloudFormation template, create a S3 bucket, IAM Roles and CodePipeline stack

Module 1: Prepare Your Lab Environment

1.1 Steps to Prepare Your Lab Environment

1.1.1 If not yet done, register for a Snowflake free 30-day trial at https://trial.snowflake.com

- The Snowflake edition (Standard, Enterprise, e.g.), cloud provider (GCP, AWS, or Azure), and Region (US Central, Europe West, e.g.) do *not* matter for this lab. But we suggest you select the region which is physically closest to you. And select the Enterprise edition so you can leverage some advanced capabilities that are not available in the Standard Edition.

- After registering, you will receive an email with an activation link and your Snowflake account URL. Bookmark this URL for easy, future access. After activation, you will create a user name and password. Write down these credentials.

1.1.2 Resize your browser windows so you can view this lab guide PDF and your web browser side-by-side to more easily follow the lab instructions. If possible, even better is to use a secondary display dedicated to the lab guide.

1.1.3 If you do not already have a AWS account please create a new account using this link - Create AWS Account. Make sure you have the permissions to use a Cloudformation Template, create a S3 bucket, IAM Roles and launch Lambda & API Gateway. Once logged in to your account, select an AWS region closest to your Snowflake account, US-WEST-2(Oregon), US-EAST-2 (Ohio) and US-EAST-1(N. Virginia) are good choices.

1.1.4 Click on https://aws-snowflake-workshop.s3.amazonaws.com/lab_scripts_online_mda.sql and download the “lab_scripts_online_mda.sql” file to your local machine. This file contains pre-written SQL commands and we will use this file later in the lab.

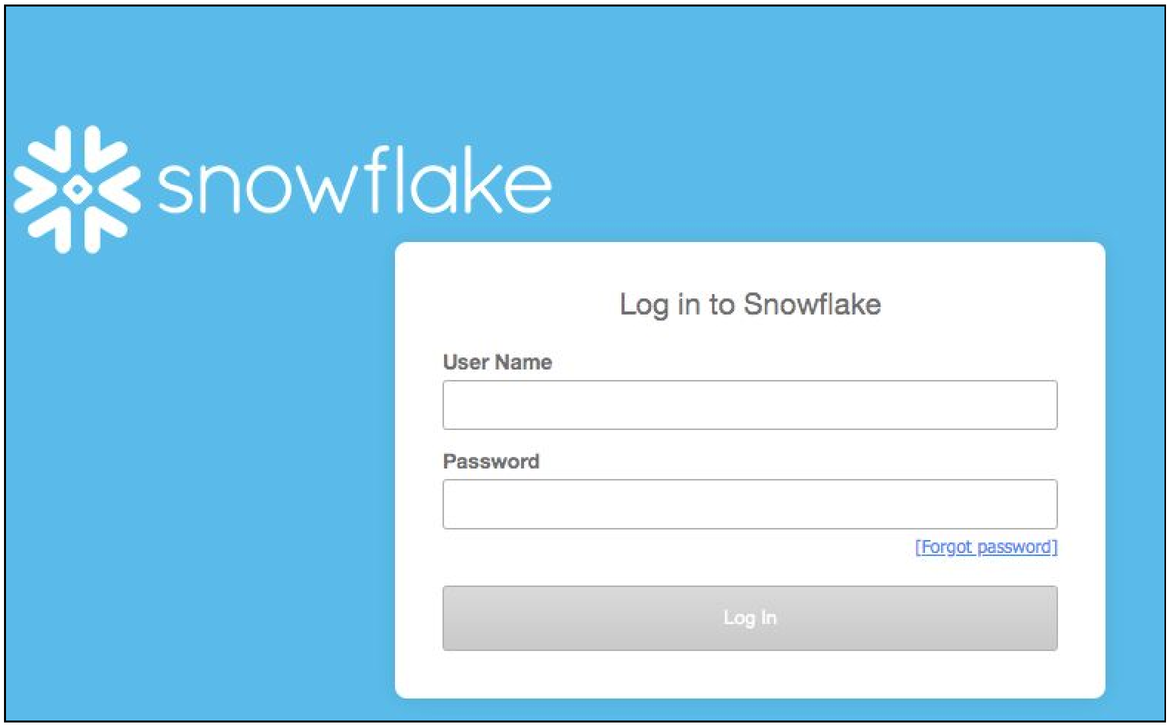
Module 2: The Snowflake User Interface & Lab "Story"

About the screen captures, sample code, and environment
Screen captures in this lab depict examples and results that may slightly vary from what you may see when you complete the exercises.

2.1 Logging Into the Snowflake User Interface (UI)

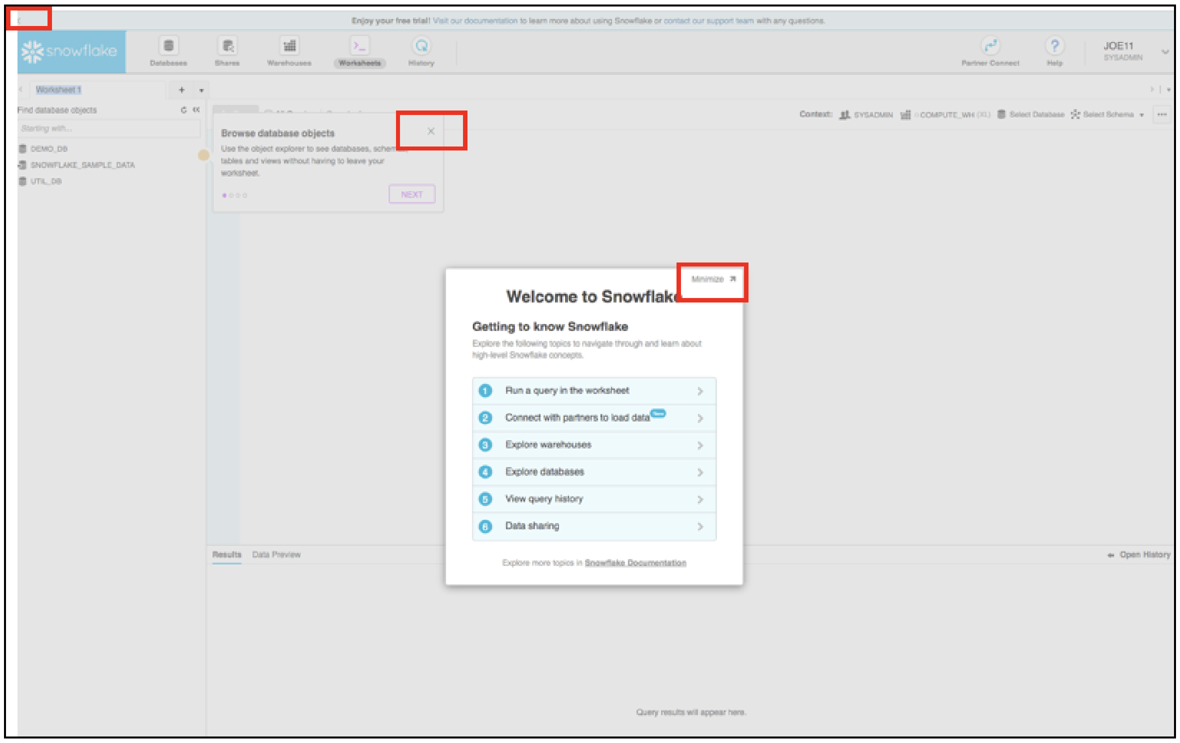
2.1.1 Open a browser window and enter the URL of your Snowflake 30-day trial environment.

2.1.2 You should see the login screen below. Enter your unique credentials to log in.



2.2 Close any Welcome Boxes and Tutorials

2.2.1 You may see "welcome" and "helper" boxes in the UI when you log in for the first time. Also a "Enjoy your free trial…" ribbon at the top of the UI. Minimize and close them by clicking on the items in the red boxes in the screenshot below.


Enjoy your free trial! Visit our documentation to learn more about using Snowflake or contact our support team with any questions.

Browse database objects

Use the object explorer to see databases, schemas, tables and views without having to leave your worksheet.

Welcome to Snowflake

Getting to know Snowflake

Explore the following topics to navigate through and learn about high-level Snowflake concepts.

1. Run a query in the worksheet
2. Connect with partners to load data
3. Explore warehouses
4. Explore databases
5. View query history
6. Data sharing

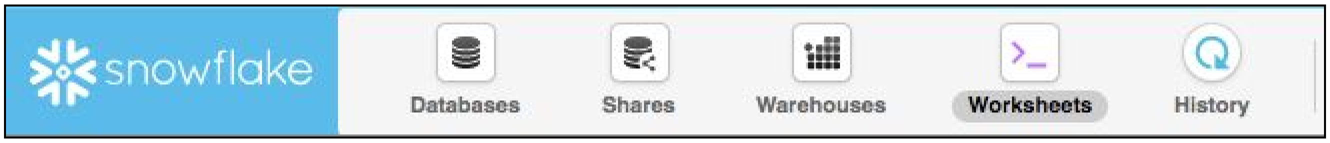
Explore more topics in Snowflake Documentation

Query results will appear here.

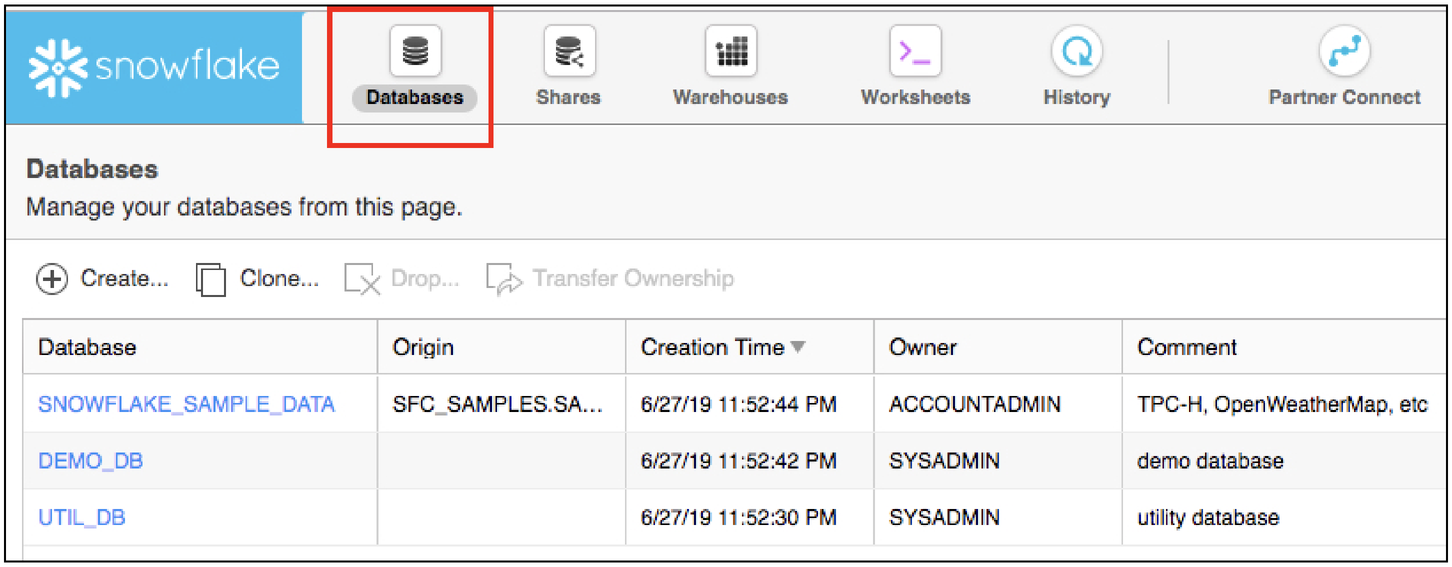
2.3 Navigating the Snowflake UI

First let’s get you acquainted with Snowflake! This section covers the basic components of the user interface to help you orient yourself. We will move left to right in the top of the UI.

2.3.1 The top menu allows you to switch between the different areas of Snowflake:



2.3.2 The **Databases** tab shows information about the databases you have created or have privileges to access. You can create, clone, drop, or transfer ownership of databases as well as load data (limited) in the UI. Notice several databases already exist in your environment. However, we will not be using these in this lab.

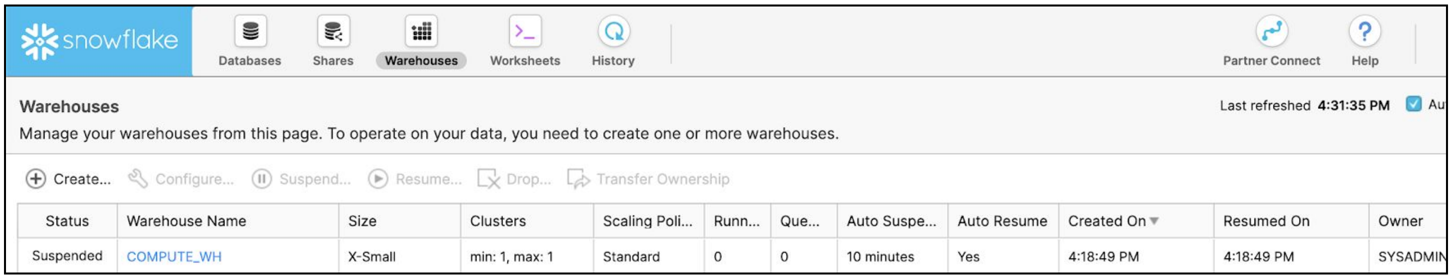


Databases
Manage your databases from this page.

Database	Origin	Creation Time ▼	Owner	Comment
SNOWFLAKE_SAMPLE_DATA	SFC_SAMPLES.SA...	6/27/19 11:52:44 PM	ACCOUNTADMIN	TPC-H, OpenWeatherMap, etc
DEMO_DB		6/27/19 11:52:42 PM	SYSADMIN	demo database
UTIL_DB		6/27/19 11:52:30 PM	SYSADMIN	utility database

2.3.3 The **Shares** tab is where data sharing can be configured to easily and securely share Snowflake table(s) among separate Snowflake accounts or external users, without having to create a second copy of the table data. At the end of this lab is a module on data sharing.

2.3.4 The **Warehouses** tab is where you set up and manage compute resources (virtual warehouses) to load or query data in Snowflake. Note a warehouse called “COMPUTE_WH (XS)” already exists in your environment.



Warehouses
Manage your warehouses from this page. To operate on your data, you need to create one or more warehouses.

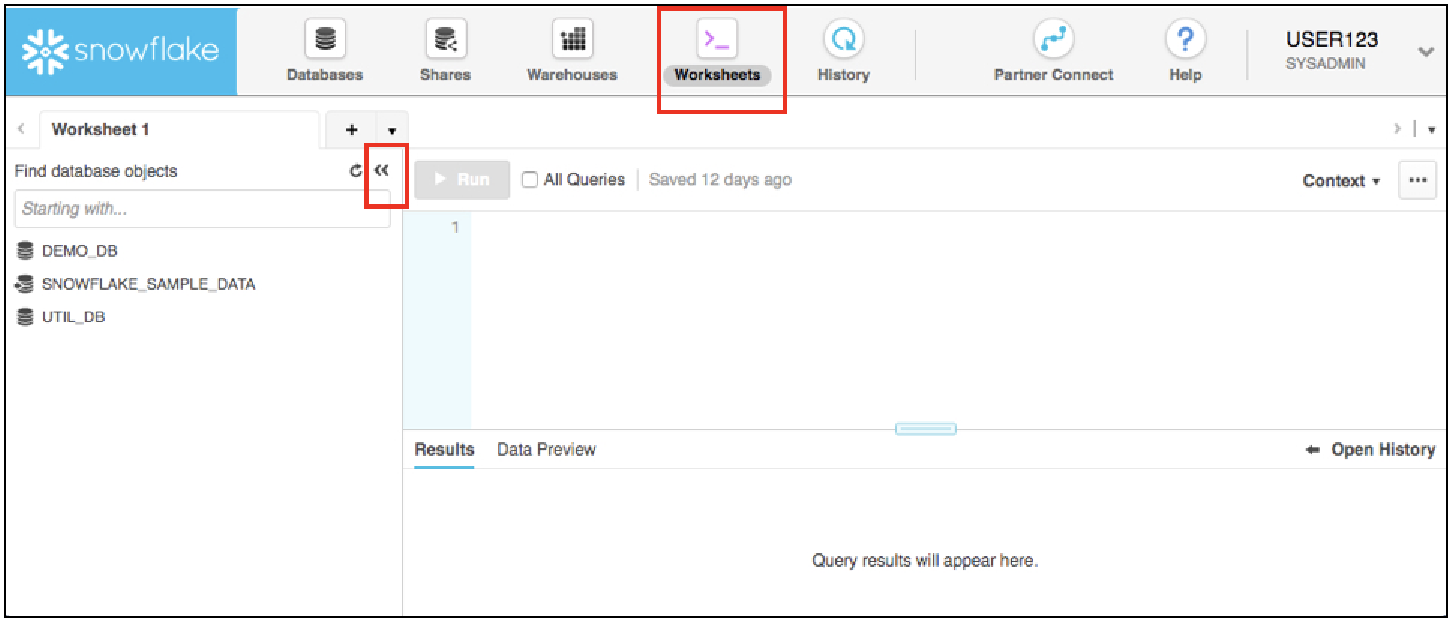
Last refreshed **4:31:35 PM**

Status	Warehouse Name	Size	Clusters	Scaling Poli...	Runn...	Que...	Auto Suspe...	Auto Resume	Created On ▼	Resumed On	Owner
Suspended	COMPUTE_WH	X-Small	min: 1, max: 1	Standard	0	0	10 minutes	Yes	4:18:49 PM	4:18:49 PM	SYSADMIN

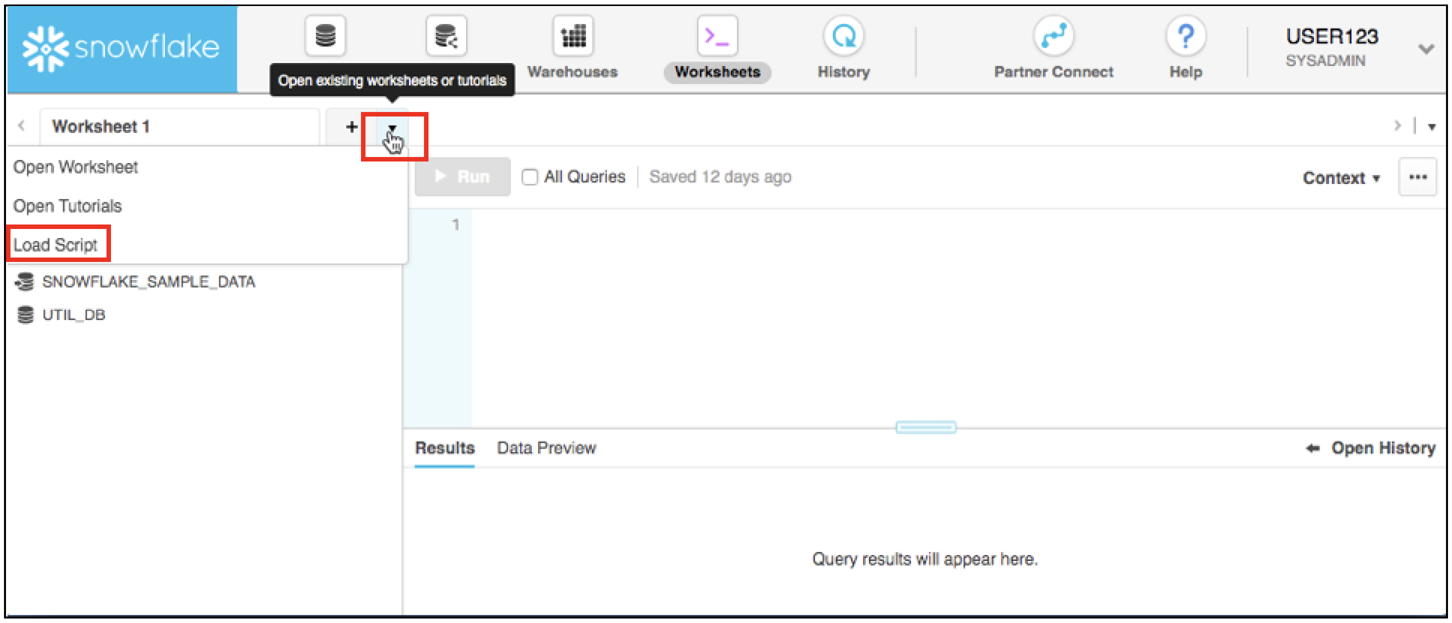
2.3.5 The **Worksheets** tab provides an interface for submitting SQL queries, performing DDL and DML operations and viewing results as your queries/operations complete. The default “Worksheet 1” appears.

In the left pane is the database objects browser which enables users to explore all databases, schemas, tables, and views accessible by the role selected for a worksheet. The bottom pane shows results of queries and operations.

The various windows on this page can be resized by moving the small sliders on them. And if during the lab you need more room to work in the worksheet, collapse the database objects browser in the left pane. Many of the screenshots in this guide will have this database objects browser closed.



2.3.6 At the top left of the default "Worksheet 1," just to the right of the worksheet tab, click on the small, downward facing arrow, select "Load Script", then browse to the "lab_scripts.sql" file you downloaded in the prior module and select "Open". All of the SQL commands you need to run for the remainder of this lab will now appear on the new worksheet. Do not run any of the SQL commands yet. We will come back to them later in the lab and execute them one at a time.

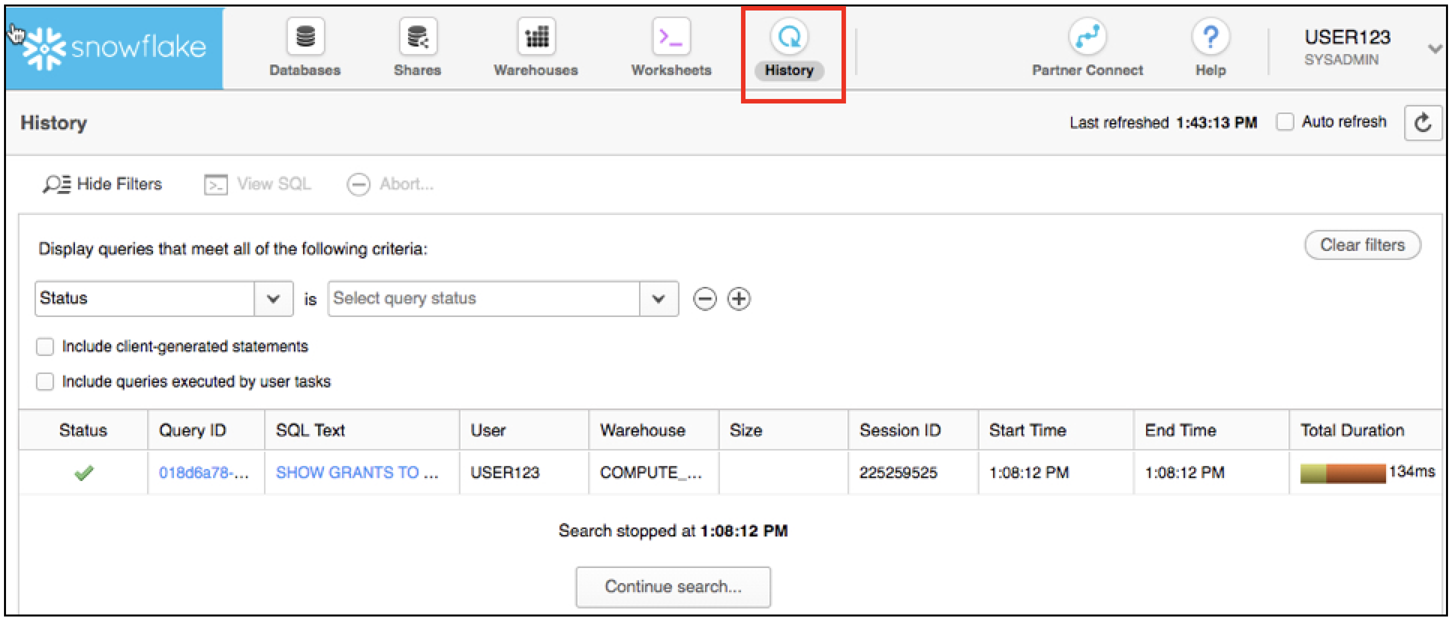


Warning - Do Not Copy/Paste SQL From This PDF to a Worksheet
Copy-pasting the SQL code from this PDF into a Snowflake worksheet will result in formatting errors and the SQL will not run correctly. Make sure to use the "Load Script" method just covered.

On older or locked-down browsers, this "load script" step may not work as the browser will prevent you from opening the .sql file. If this is the case, open the .sql file with a text editor and then copy/paste all the text from the .sql file to the "Worksheet 1"

Worksheets vs the UI
Much of the configurations in this lab will be executed via this pre-written SQL in the Worksheet in order to save time. These configurations could also be done via the UI in a less technical manner but would take more time.

2.3.7 The **History** tab allows you to view the details of all queries executed in the last 14 days in the Snowflake account (click on a Query ID to drill into the query for more detail).



History

Last refreshed **1:43:13 PM** Auto refresh

Hide Filters View SQL Abort...

Display queries that meet all of the following criteria: Clear filters

Status is Select query status

Include client-generated statements

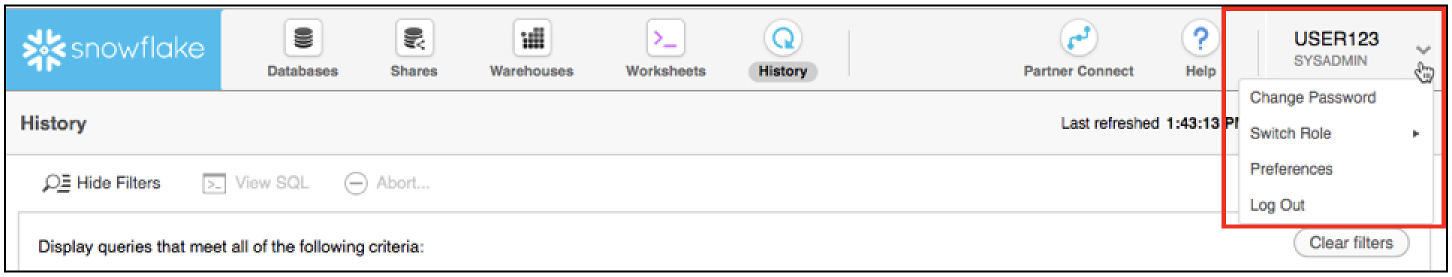
Include queries executed by user tasks

Status	Query ID	SQL Text	User	Warehouse	Size	Session ID	Start Time	End Time	Total Duration
✓	018d6a78-...	SHOW GRANTS TO ...	USER123	COMPUTE_...		225259525	1:08:12 PM	1:08:12 PM	134ms

Search stopped at **1:08:12 PM**

Continue search...

2.3.8 If you click on the top right of the UI where your user name appears, you will see that here you can do things like change your password, roles, or preferences. Snowflake has several system defined roles. You are currently in the default role of SYSADMIN.



SYSADMIN
For most this lab you will remain in the IMDB_ROLE role which has custom privileges to alter warehouses and create databases and other objects in an account.

In a real-world environment, you would use different roles for the tasks in this lab, and assign the roles to your users. More on access control in Snowflake is in towards the end of this lab and also at https://docs.snowflake.net/manuals/user-guide/security-access-control.html

2.4 The Lab story

This Snowflake lab will be done as part of a theoretical real-world “story” to help you better understand why we are performing the steps in this lab and in the order they appear.

The “story” of this lab is based on the team building a data app presenting movie data over REST interface. It’s a backend serverless model that consumes data available in Snowflake for all customers that have access to the application.

We will first load structured .csv data from IMDB dataset into Snowflake. Then later we will deploy AWS CodePipeline and a serverless stack that our application is made of. Once we have the application working, we will load semi-structured JSON movie data, enrich our existing views in an automated manner using CI/CD. We will also learn how to take advantage of native devops capabilities of Amazon Web Services and Snowflake.

Module 3: Preparing to Load the Data and Loading the Data

Let’s start by preparing to load the structured IMDB data into Snowflake.

This module will walk you through the steps to:

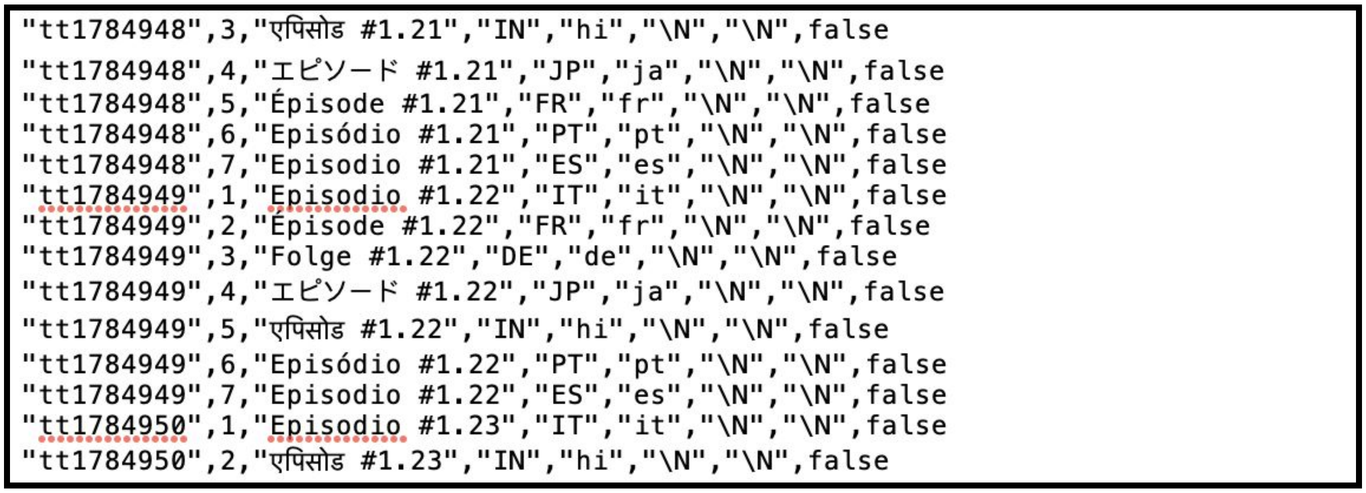
- Create a database and table
- Create a role and user
- Grant permissions to the role
- Create a virtual warehouse
- Create an external stage
- Create a file format for the data
- Load data to Snowflake tables
- Create a view

Getting Data into Snowflake
There are many ways to get data into Snowflake from many locations including the COPY command, Snowpipe auto-ingestion, an external connector, or a third-party ETL/ELT product. More information on getting data into Snowflake, see https://docs.snowflake.net/manuals/user-guide-data-load.html

We are using the COPY command and GCS storage for this module in a manual process so you can see and learn from the steps involved. In the real-world, a customer would likely use an automated process or ETL product to make the data loading process fully automated and much easier.

The data we will be using is movie data available at Kaggle. The data has been exported and pre-staged for you in an Amazon S3 bucket in the us-west-1 (N. California) region. The data consists of 7 tables, and contains information about titles, crew, ratings, etc. The data represents 95M rows, 145 objects, and 0.9GB total size compressed.

Below is a snippet from one of the “title” CSV data files:

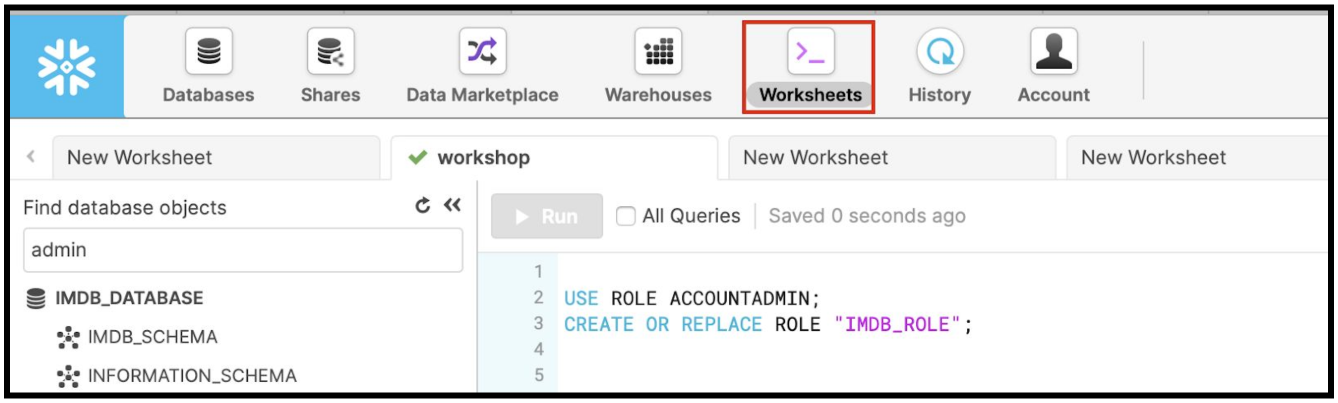


```
"tt1784948",3,"एपिसोड #1.21","IN","hi","\N","\N",false
"tt1784948",4,"エピソード #1.21","JP","ja","\N","\N",false
"tt1784948",5,"Épisode #1.21","FR","fr","\N","\N",false
"tt1784948",6,"Episódio #1.21","PT","pt","\N","\N",false
"tt1784948",7,"Episodio #1.21","ES","es","\N","\N",false
"tt1784949",1,"Episodio #1.22","IT","it","\N","\N",false
"tt1784949",2,"Épisode #1.22","FR","fr","\N","\N",false
"tt1784949",3,"Folge #1.22","DE","de","\N","\N",false
"tt1784949",4,"エピソード #1.22","JP","ja","\N","\N",false
"tt1784949",5,"एपिसोड #1.22","IN","hi","\N","\N",false
"tt1784949",6,"Episódio #1.22","PT","pt","\N","\N",false
"tt1784949",7,"Episodio #1.22","ES","es","\N","\N",false
"tt1784950",1,"Episodio #1.23","IT","it","\N","\N",false
"tt1784950",2,"एपिसोड #1.23","IN","hi","\N","\N",false
```

It is in comma-delimited format with double quote enclosing and no header line. This will come into play later in this module as we configure the Snowflake tables which will store this data.

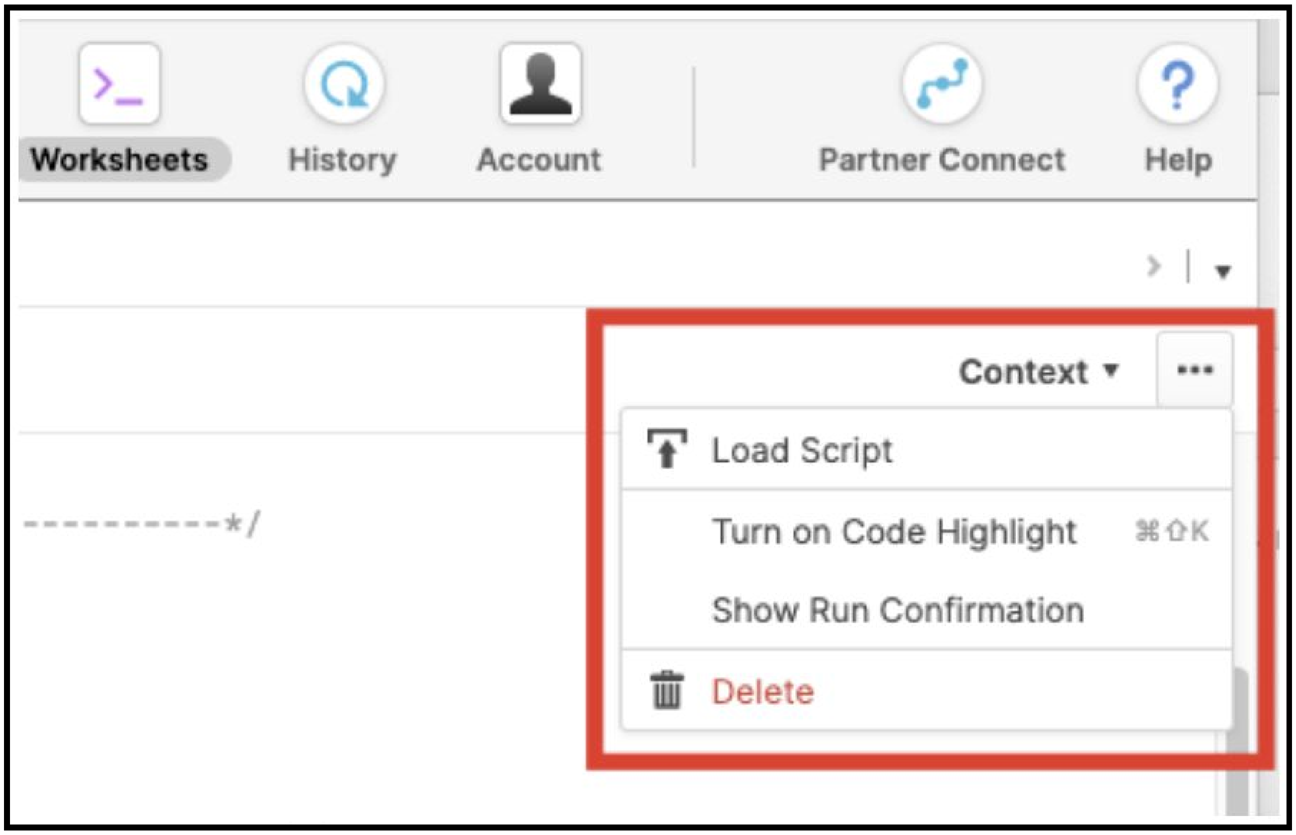
3.1 Create database and warehouse

3.1.1 At the top of the Snowflake UI, click the Worksheets tab. You should see the worksheet with all the SQL we loaded in a prior step.

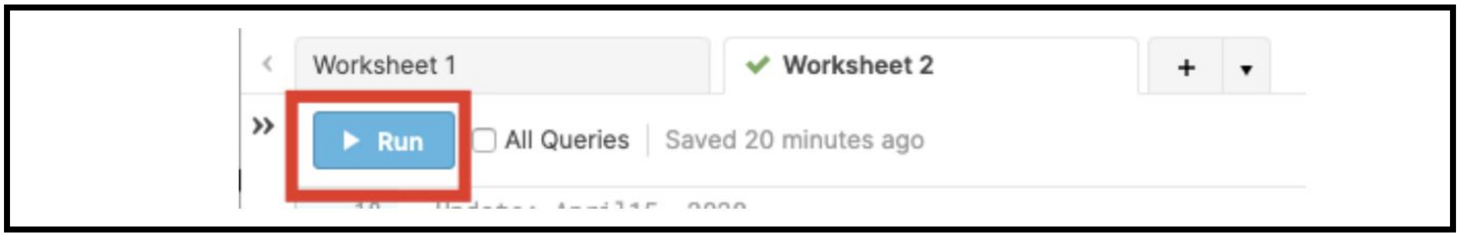


```

USE ROLE ACCOUNTADMIN;
CREATE OR REPLACE ROLE "IMDB_ROLE";


```

3.1.2 Before we start using SQL in Worksheets we will turn on Code Highlight by clicking on the 3 dots on the top right hand corner of the worksheet, and then clicking on Turn on Code Highlight. This will make it easier to identify the SQL that will be executed.



3.1.3 To execute a SQL command or commands, click or select multiple commands with your mouse. The SQL command(s) will be highlighted in BLUE. You can now either press COMMAND & RETURN on a Mac or CONTROL & ENTER on Windows; or you can click the RUN button towards the top left hand slide of the Worksheet.



Warning
In this lab, never check the “All Queries” box at the top of the worksheet. We want to run SQL queries one at a time in a specific order; not all at once.

3.1.4 Next we will briefly switch roles to the ACCOUNTADMIN role primarily to allow us to create a specific role for this workshop. Execute the SQL command shown below.

```
USE ROLE ACCOUNTADMIN;
```

3.1.5 First, let's create a database called IMDB_DATABASE that will be used for loading the movies data. We want to also create a dedicated SCHEMA that will be used later on.

```
CREATE OR REPLACE DATABASE  "IMDB_DATABASE";

CREATE SCHEMA "IMDB_DATABASE"."IMDB_SCHEMA";
```

3.1.6 Let's create a Warehouse called IMDB_WH.
Note: that the Warehouse is configured to auto suspend and resume, this prevents the unnecessary use of credits if the Warehouse is not being used with the convenience that it will automatically resume when needed

```
CREATE OR REPLACE WAREHOUSE IMDB_WH

WITH WAREHOUSE_SIZE = 'XSMALL'

WAREHOUSE_TYPE = 'STANDARD' AUTO_SUSPEND = 60

AUTO_RESUME = TRUE MIN_CLUSTER_COUNT = 1 MAX_CLUSTER_COUNT = 5
SCALING_POLICY = 'STANDARD';
```

3.2 Create role and user

3.2.1 Now we will create a role named IMDB_ROLE that will be used to control access to objects in Snowflake. We will also GRANT all privileges on the IMDB_WH Warehouse and IMDB_DATABASE to this role. We want to also make sure that IMDB_ROLE will have permissions over all future created tables. We will also grant the IMDB_ROLE to the ACCOUNTADMIN role to allow ACCOUNTADMIN to have all the IMDB_ROLE privileges.

```
CREATE OR REPLACE ROLE "IMDB_ROLE";

GRANT ROLE "IMDB_ROLE" TO ROLE "ACCOUNTADMIN";
GRANT USAGE,MODIFY,OPERATE ON WAREHOUSE "IMDB_WH" TO ROLE
"IMDB_ROLE";

GRANT USAGE ON SCHEMA "IMDB_DATABASE"."IMDB_SCHEMA" TO ROLE
"IMDB_ROLE";

GRANT SELECT, DELETE, TRUNCATE, UPDATE ON FUTURE TABLES IN
SCHEMA "IMDB_DATABASE"."IMDB_SCHEMA" TO ROLE "IMDB_ROLE";

GRANT SELECT ON FUTURE VIEWS IN SCHEMA
"IMDB_DATABASE"."IMDB_SCHEMA" TO ROLE "IMDB_ROLE";

GRANT CREATE TABLE, CREATE VIEW, CREATE FILE FORMAT, CREATE
STAGE, CREATE STREAM, CREATE TASK ON SCHEMA
"IMDB_DATABASE"."IMDB_SCHEMA" TO ROLE "IMDB_ROLE";

GRANT CREATE DATABASE ON ACCOUNT TO ROLE IMDB_ROLE;

GRANT OWNERSHIP ON DATABASE IMDB_DATABASE TO ROLE IMDB_ROLE;
```

3.2.2 The next step is to create a user that will be used by external services to connect to the Snowflake account. The user IMDB_ENGINEER will be created and assigned a default role, warehouse and database to establish the default context when connected to the Snowflake account.
Note that the SQL statement has a password “P@$$w0rd1234!”, you can change the password to one that you prefer. Do note the password you use if you do change it as it will be required for the next portion of the lab when Amazon Lambda will connect to Snowflake.We will also grant the IMDB_ROLE to the IMDB_ENGINEER user.Finally we will switch to the IMDB_ROLE role for the next steps.

```
CREATE or REPLACE USER "IMDB_ENGINEER" PASSWORD = 'P@$$w0rd1234!'

DEFAULT_ROLE = "IMDB_ROLE" DEFAULT_WAREHOUSE = 'IMDB_WH'
DEFAULT_NAMESPACE = 'IMDB_DATABASE.IMDB_SCHEMA'
MUST_CHANGE_PASSWORD = FALSE;

GRANT ROLE "IMDB_ROLE" TO USER "IMDB_ENGINEER";

USE ROLE "IMDB_ROLE";
```

DDL operations are free!
Note that all the DDL operations we have done so far do NOT require compute resources, so we can create all our objects for free.

3.3 Create a file format and external stage

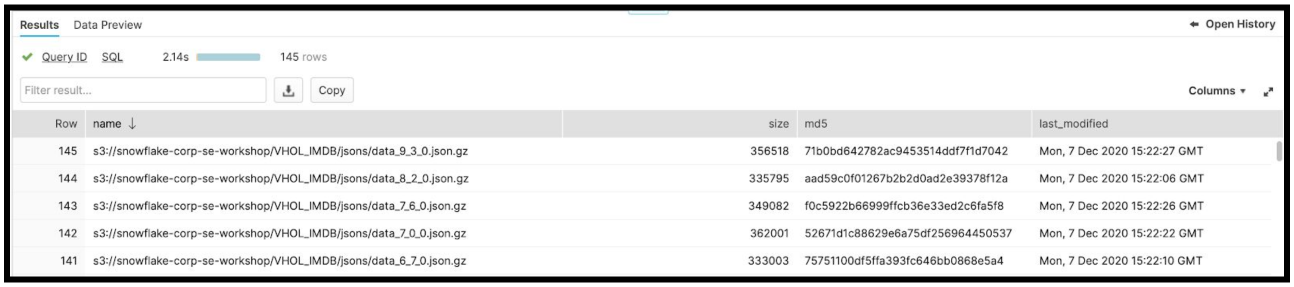
We are working with structured, comma-delimited data that has already been staged in a public, external S3 bucket. Before we can use this data, we first need to create an External Stage that specifies the location of our external bucket. We also need to create a File Format for the comma-delimited data.
NOTE - For this lab we are using an AWS S3 bucket located in US-West-1 (N. California). In the real-world, to prevent data egress/transfer costs, you would want to select a staging location from the same region that your Snowflake environment is in.

3.3.1 First we will create an External Stage to an existing Amazon S3 Location. The data is located in a public S3 bucket and does not require credentials for ease of use with the lab. In practice you will need to configure secure access to Amazon S3. For more information see the Snowflake documentation https://docs.snowflake.com/en/user-guide/data-load-s3.html

```
CREATE OR REPLACE STAGE
"IMDB_DATABASE"."IMDB_SCHEMA"."IMDB_SOURCE_STAGE"

URL = 's3://snowflake-corp-se-workshop/VHOL_IMDB' ;
```

3.3.2 We can list the content of our External Stage

```
LIST @IMDB_SOURCE_STAGE;
```



Row	name	size	md5	last_modified
145	s3://snowflake-corp-se-workshop/VHOL_IMDB/jsons/data_9_3_0.json.gz	356518	71b0bd642782ac9453514ddf7f1d7042	Mon, 7 Dec 2020 15:22:27 GMT
144	s3://snowflake-corp-se-workshop/VHOL_IMDB/jsons/data_8_2_0.json.gz	335795	aad59c0f01267b2b2d0ad2e39378f12a	Mon, 7 Dec 2020 15:22:06 GMT
143	s3://snowflake-corp-se-workshop/VHOL_IMDB/jsons/data_7_6_0.json.gz	349082	f0c5922b66999ffcb36e33ed2c6fa5f8	Mon, 7 Dec 2020 15:22:26 GMT
142	s3://snowflake-corp-se-workshop/VHOL_IMDB/jsons/data_7_0_0.json.gz	362001	52671d1c88629e6a75df256964450537	Mon, 7 Dec 2020 15:22:22 GMT
141	s3://snowflake-corp-se-workshop/VHOL_IMDB/jsons/data_6_7_0.json.gz	333003	75751100df5ffa393fc646bb0868e5a4	Mon, 7 Dec 2020 15:22:10 GMT

3.3.3 We can also measure basic volumetric parameters like number of files, average file size and total storage used, with this SQL:

```
SELECT FLOOR(SUM($2)/POWER(1024, 3),1)
TOTAL_COMPRESSED_STORAGE_GB,
   FLOOR(AVG($2)/POWER(1024, 2),1) AVG_FILE_SIZE_MB,
  COUNT(*) AS NUM_FILES
FROM TABLE (RESULT_SCAN(LAST_QUERY_ID()));
```

3.3.4 Next we will create the File Format that will be used. The CSV data does not have a header.

```
CREATE FILE FORMAT "IMDB_DATABASE"."IMDB_SCHEMA"."CSV_IMDB"
TYPE = 'CSV' COMPRESSION = 'AUTO' FIELD_DELIMITER = ','
RECORD_DELIMITER = '\n' SKIP_HEADER = 0
FIELD_OPTIONALLY_ENCLOSED_BY = '\042'
TRIM_SPACE = FALSE ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE
ESCAPE = 'NONE' ESCAPE_UNENCLOSED_FIELD = '\134' DATE_FORMAT = 'AUTO'
TIMESTAMP_FORMAT = 'AUTO' NULL_IF = ('\\N');
```

3.4 Create and load tables

3.4.1 Next we will create the TITLE, TITLE_BASICS, TITLE_CREW, TITLE_EPISODE, TITLE_PRINCIPALS, TITLE_BASICS, NAME_BASICS tables to load movie data from a S3 bucket.

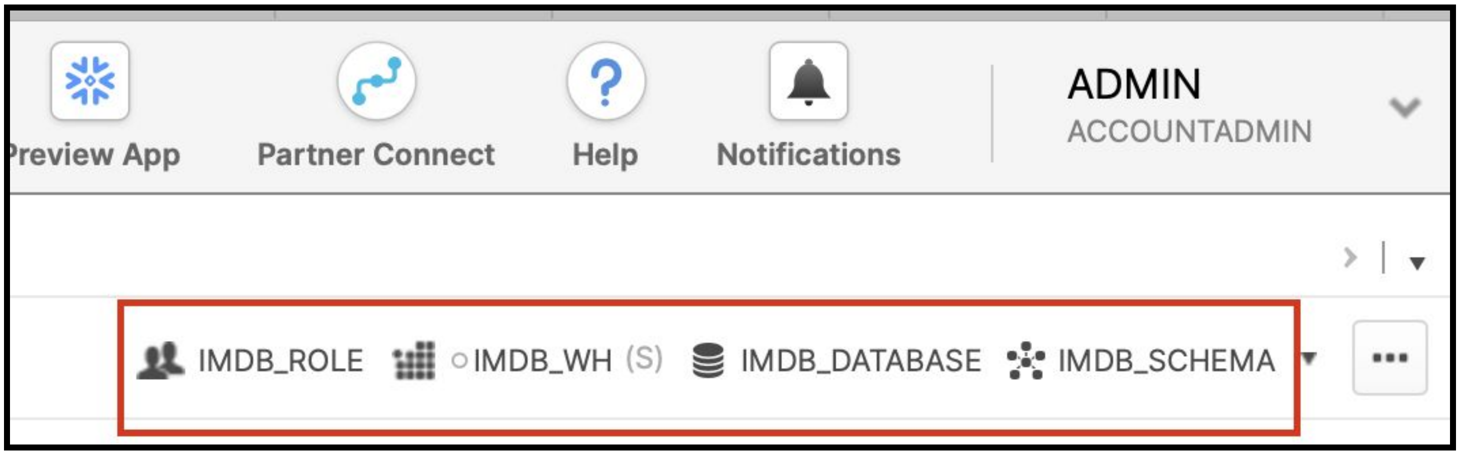
```
CREATE OR REPLACE TABLE title(titleid varchar(500),ordering integer,title
varchar(1000), region varchar(500),"language" varchar(500),types
varchar(500),attributes varchar(500),isoriginaltitle boolean);

CREATE OR REPLACE TABLE title_basics(tconst varchar(500), titletype varchar(500),
primarytitle varchar(500),originaltitle varchar(500),isadult boolean,startyear date,endyear
date, runtimeminutes int,genres varchar(500));

CREATE OR REPLACE TABLE title_crew(tconst varchar, directors varchar,writers
varchar);

CREATE OR REPLACE TABLE title_episode(tconst varchar,parent
varchar,seasonnumber int,epiosdenumber int);

CREATE OR REPLACE TABLE title_principals(tconst varchar, ordering int,nconst
varchar,category varchar,job varchar,characters varchar);

CREATE OR REPLACE TABLE title_ratings(tconst varchar(500),averagerating
double,numvotes int);

CREATE OR REPLACE TABLE name_basics(nconst varchar, primaryname
varchar,birthyear date,deathyear date,"primary_proffesion" varchar,knownfortitles
varchar);
```

Many Options to Run Commands.
SQL commands can be executed through the UI (limited), via the Worksheets tab, using our SnowSQL command line tool, a SQL editor of your choice via ODBC/JDBC, or through our Python or Spark connectors.

As mentioned earlier, in this lab we will run operations via pre-written SQL in the worksheet (as opposed to using the UI) to save time.

3.4.2 Let's now switch context to use the IMDB_WH warehouse. We want to also increase the warehouse size to SMALL, this will allow us to load our data faster into the tables. The context for executing SQL commands is shown in the top right hand corner of the worksheets.

```
ALTER WAREHOUSE IMDB_WH SET WAREHOUSE_SIZE=SMALL;
```



3.4.3 Finally let's load the movie data from the S3 bucket into Snowflake. We will load the data using the COPY command.

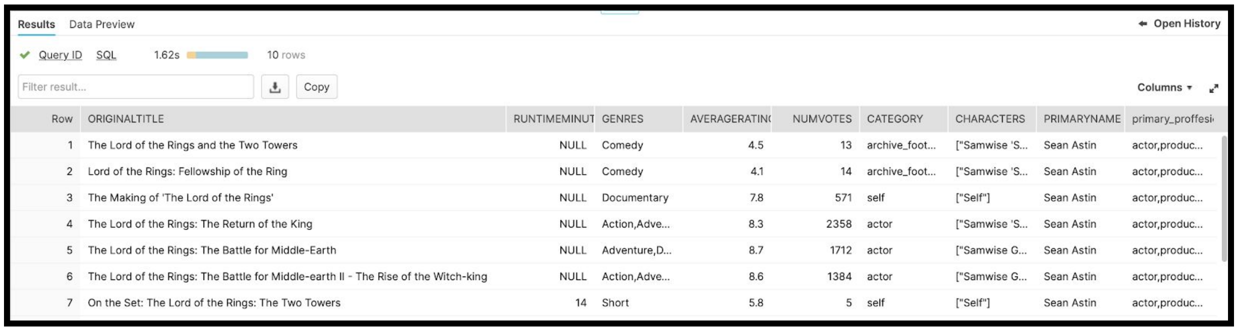
```
COPY INTO  name_basics FROM @IMDB_SOURCE_STAGE/imdb/name_basics/
FILE_FORMAT=(FORMAT_NAME='CSV_imdb');

COPY INTO  title FROM @IMDB_SOURCE_STAGE/imdb/title/
FILE_FORMAT=(FORMAT_NAME='CSV_imdb');

COPY INTO title_crew FROM @IMDB_SOURCE_STAGE/imdb/title_crew/
FILE_FORMAT=(FORMAT_NAME='CSV_imdb');

COPY INTO title_basics FROM @IMDB_SOURCE_STAGE/imdb/title_basics/
```

```
FILE_FORMAT=(FORMAT_NAME='CSV_imdb');

COPY INTO  title_episode FROM @"IMDB_SOURCE_STAGE"/imdb/title_episode/
FILE_FORMAT=(FORMAT_NAME='CSV_imdb');

COPY INTO title_ratings FROM @"IMDB_SOURCE_STAGE"/imdb/title_ratings/
FILE_FORMAT=(FORMAT_NAME='CSV_imdb');

COPY INTO  title_principals FROM @"IMDB_SOURCE_STAGE"/imdb/title_principals/
FILE_FORMAT=(FORMAT_NAME='CSV_imdb');
```

3.4.4 Let’s look at the data by running a SELECT with JOIN and WHERE command, we can limit the results to Lord of the Rings movie and Sean Astin as a person who participated in the movie.

```
SELECT
originaltitle,runtimeminutes,genres,averagerating,numvotes,category,characters,primary
name,"primary_proffesion" FROM imdb_database.imdb_schema.title_basics b
LEFT JOIN imdb_database.imdb_schema.title_ratings r
ON b.tconst=r.tconst
LEFT JOIN imdb_database.imdb_schema.title_principals p
ON r.tconst=p.tconst
LEFT JOIN imdb_database.imdb_schema.title_crew c
ON p.tconst = c.tconst
LEFT JOIN imdb_database.imdb_schema.name_basics ba
ON p.nconst = ba.nconst
WHERE originaltitle ILIKE '%lord of the rings%'
AND PRIMARYNAME = 'Sean Astin'
LIMIT 10;
```

The results are displayed in the frame below the Worksheet.

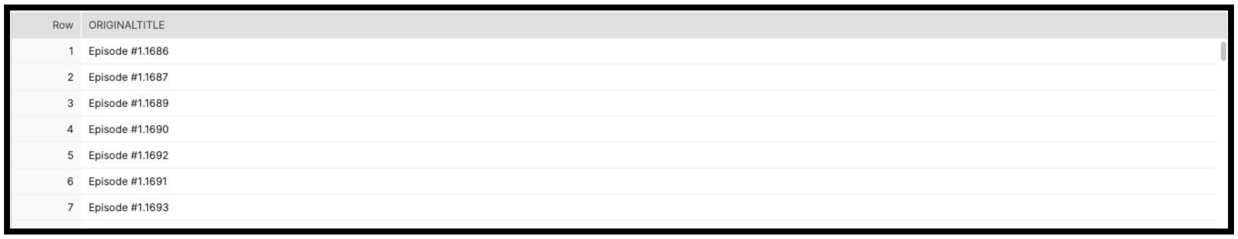


1.62s 10 rows

Row	ORIGINALTITLE	RUNTIMEMINUT	GENRES	AVERAGERATIN(	NUMVOTES	CATEGORY	CHARACTERS	PRIMARYNAME	primary_proffesi
1	The Lord of the Rings and the Two Towers	NULL	Comedy	4.5	13	archive_foot...	["Samwise 'S...	Sean Astin	actor,produc...
2	Lord of the Rings: Fellowship of the Ring	NULL	Comedy	4.1	14	archive_foot...	["Samwise 'S...	Sean Astin	actor,produc...
3	The Making of 'The Lord of the Rings'	NULL	Documentary	7.8	571	self	["Self"]	Sean Astin	actor,produc...
4	The Lord of the Rings: The Return of the King	NULL	Action,Adve...	8.3	2358	actor	["Samwise 'S...	Sean Astin	actor,produc...
5	The Lord of the Rings: The Battle for Middle-Earth	NULL	Adventure,D...	8.7	1712	actor	["Samwise G...	Sean Astin	actor,produc...
6	The Lord of the Rings: The Battle for Middle-earth II - The Rise of the Witch-king	NULL	Action,Adve...	8.6	1384	actor	["Samwise G...	Sean Astin	actor,produc...
7	On the Set: The Lord of the Rings: The Two Towers	14	Short	5.8	5	self	["Self"]	Sean Astin	actor,produc...

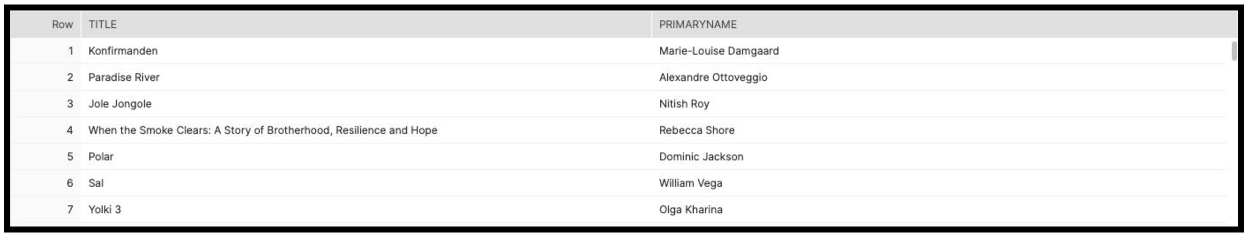
3.5 Create views for our application

We will expose three views to our application.
Information about movies under MOVIES view:



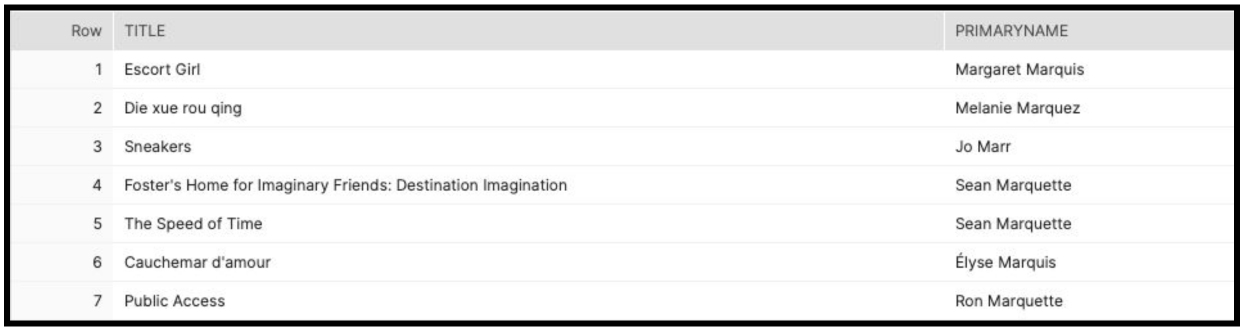
Row	ORIGINALTITLE
1	Episode #1.1686
2	Episode #1.1687
3	Episode #1.1689
4	Episode #1.1690
5	Episode #1.1692
6	Episode #1.1691
7	Episode #1.1693

Information about directors under DIRECTOR view:



Row	TITLE	PRIMARYNAME
1	Konfirmanden	Marie-Louise Damgaard
2	Paradise River	Alexandre Ottoveggio
3	Jole Jongole	Nitish Roy
4	When the Smoke Clears: A Story of Brotherhood, Resilience and Hope	Rebecca Shore
5	Polar	Dominic Jackson
6	Sal	William Vega
7	Yolki 3	Olga Kharina

And lastly information about actors in ACTOR view:



Row	TITLE	PRIMARYNAME
1	Escort Girl	Margaret Marquis
2	Die xue rou qing	Melanie Marquez
3	Sneakers	Jo Marr
4	Foster's Home for Imaginary Friends: Destination Imagination	Sean Marquette
5	The Speed of Time	Sean Marquette
6	Cauchemar d'amour	Élyse Marquis
7	Public Access	Ron Marquette

Each one will be available for querying under different API Resource name.

3.5.1 Let's create the first VIEW with movie titles. We want to display only the original title of the movie.

```
CREATE OR REPLACE VIEW movies AS (SELECT originaltitle FROM
imdb_database.imdb_schema.title_basics);
```

3.5.2 Now, we can create a VIEW with information about movie directors. We would like to have details on the movie title that the person directed.

```
CREATE OR REPLACE VIEW director AS (SELECT distinct(TITLE),PRIMARYNAME
FROM NAME_BASICS N
LEFT JOIN TITLE_PRINCIPALS P
on N.NCONST=P.NCONST

LEFT JOIN TITLE T
ON P.TCONST=T.TITLEID

JOIN TITLE_RATINGS R
ON T.TITLEID=R.TCONST

WHERE CATEGORY='director'
AND ISORIGINALTITLE='TRUE');
```

3.5.3 Lastly, let’s prepare the view with actors information, we want to have similar information as in DIRECTOR view.

```
CREATE OR REPLACE VIEW actor AS (SELECT distinct(TITLE),PRIMARYNAME
FROM NAME_BASICS N

LEFT JOIN TITLE_PRINCIPALS P
on N.NCONST=P.NCONST

LEFT JOIN TITLE T
ON P.TCONST=T.TITLEID

JOIN TITLE_RATINGS R
ON T.TITLEID=R.TCONST

WHERE CATEGORY LIKE ANY ('actor','actress')
AND ISORIGINALTITLE='TRUE')
```

Module 4: Deploy CodePipeline and Serverless application

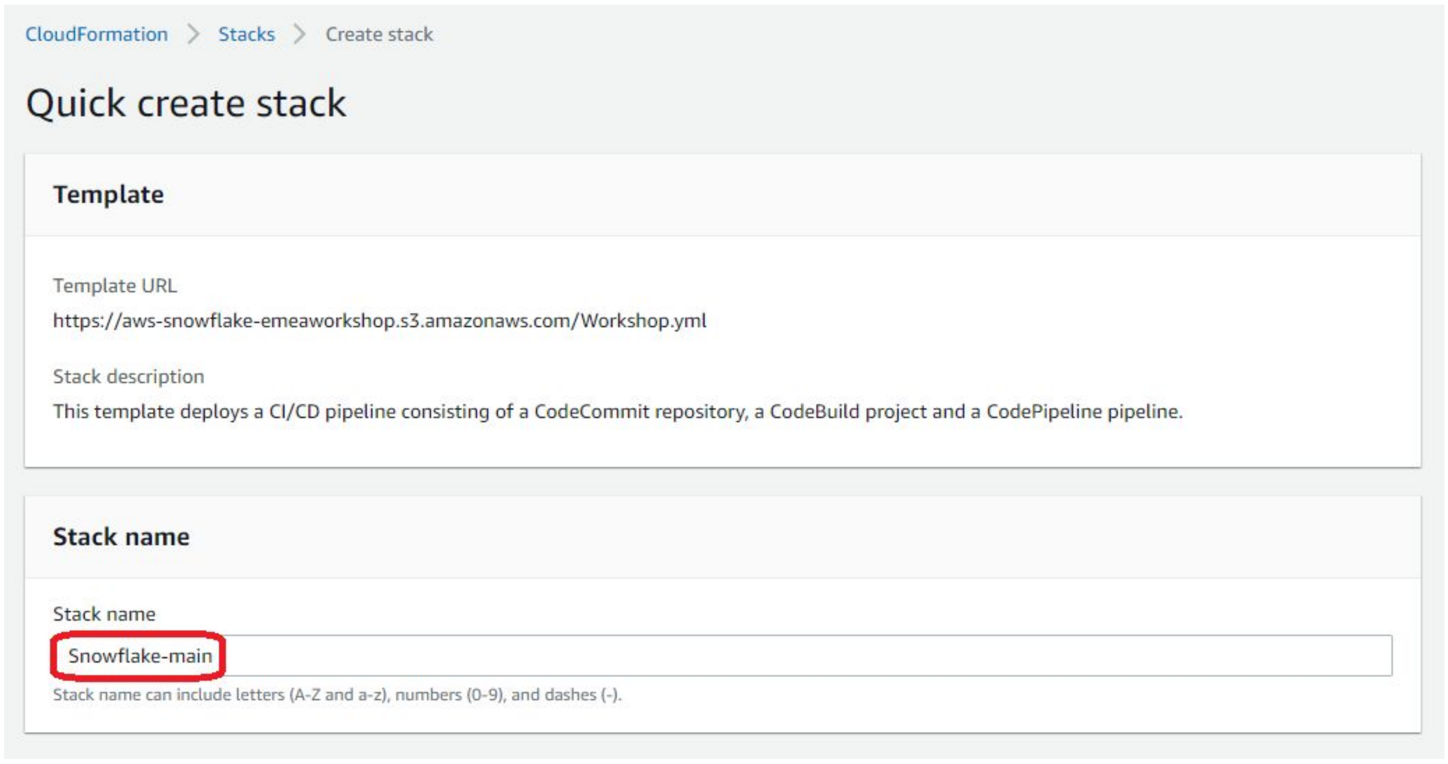
To get started, you will be following instructions to setup your AWS Account. You will be leveraging AWS CloudFormation, which allows you to write infrastructure as code.

Warning
By executing these templates, you are taking responsibility for the lifecycle and costs associated with provisioning them. Please follow the workshop clean-up instructions to remove all resources from your AWS account once you have finished the workshop to avoid unexpected costs.

1. Launch the AWS CloudFormation stack

Region	Launch stack
US East (N. Virginia)	Launch
US West (Oregon)	Launch
Europe (Frankfurt)	Launch
Europe (Ireland)	Launch
Asia Pacific (Singapore)	Launch
Asia Pacific (Sydney)	Launch

2. Enter a stack name (or just keep the default name)



Quick create stack

Template

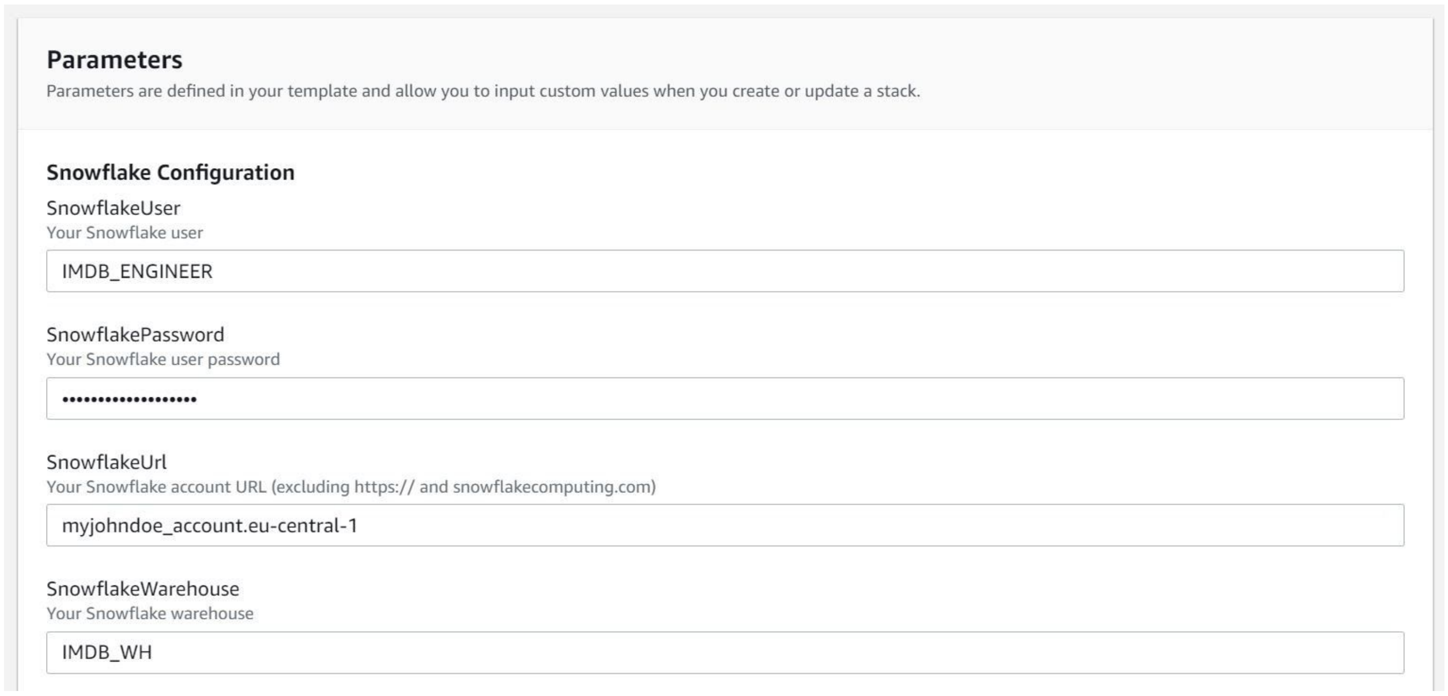
Template URL
https://aws-snowflake-emeaworkshop.s3.amazonaws.com/Workshop.yml

Stack description
This template deploys a CI/CD pipeline consisting of a CodeCommit repository, a CodeBuild project and a CodePipeline pipeline.

Stack name

Stack name
Snowflake-main
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

3. Input your Snowflake credentials in the configuration section



Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Snowflake Configuration

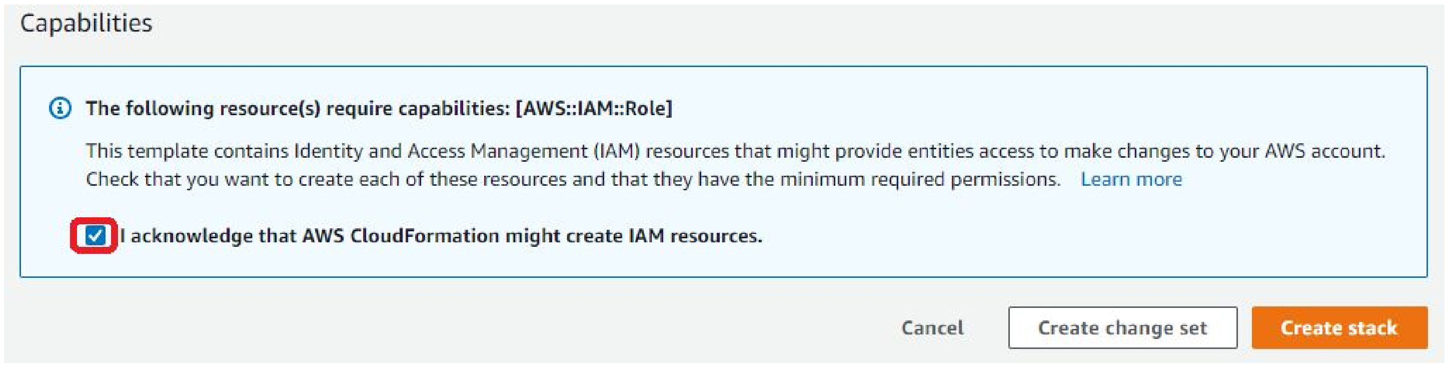
SnowflakeUser
Your Snowflake user
IMDB_ENGINEER

SnowflakePassword
Your Snowflake user password
••••••••••••••••••

SnowflakeUrl
Your Snowflake account URL (excluding https:// and snowflakecomputing.com)
myjohndoe_account.eu-central-1

SnowflakeWarehouse
Your Snowflake warehouse
IMDB_WH

4. Check the boxes in the Capabilities section



Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]

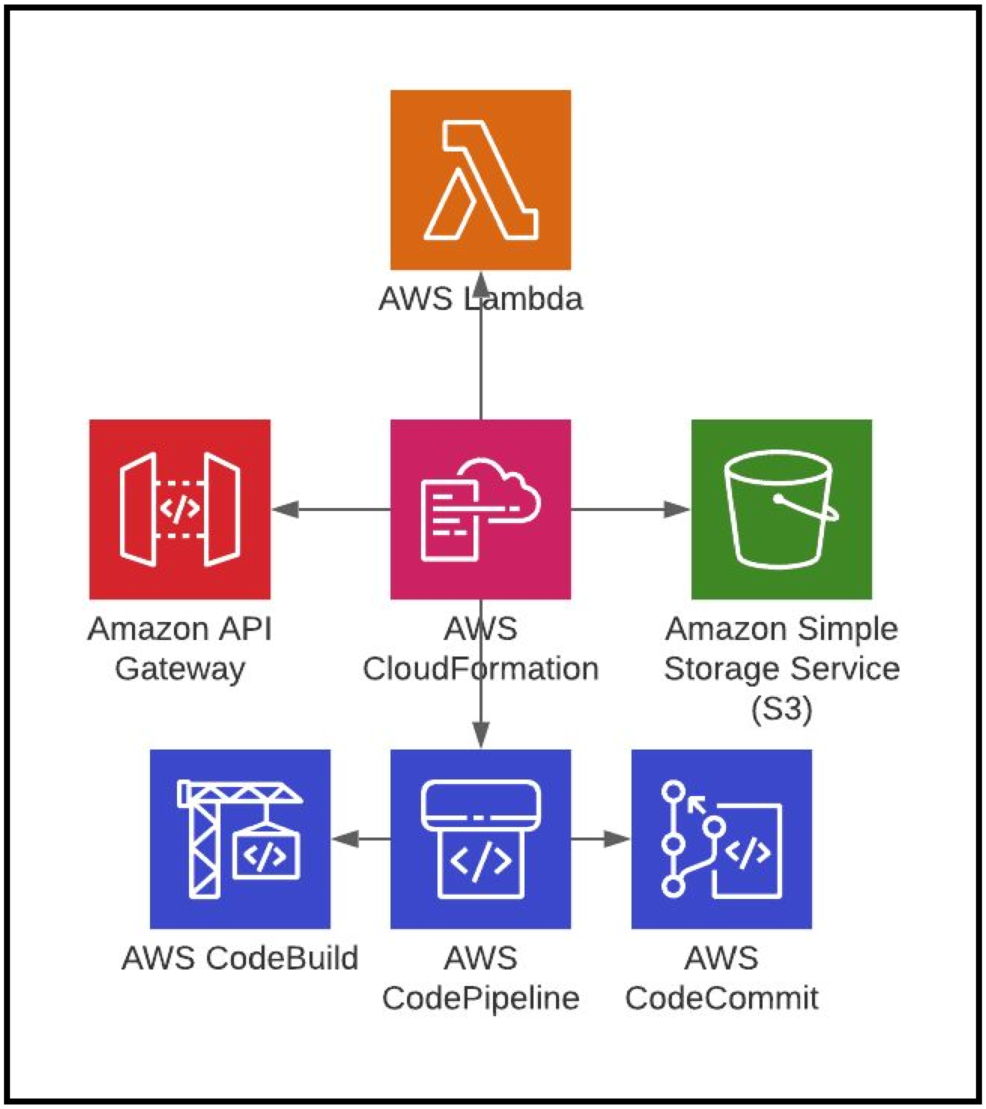
This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. Learn more

[✓] **I acknowledge that AWS CloudFormation might create IAM resources.**

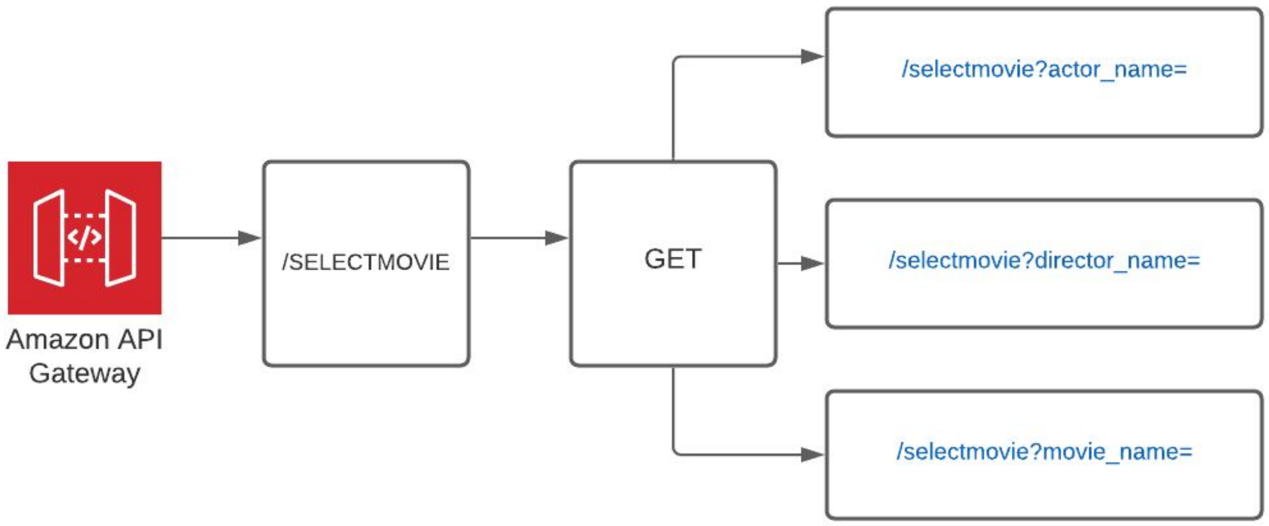
5. Click Create stack

The stack should take about 10 minutes to complete deployment and will create the following components in your AWS account:

- An AWS CodeCommit repository and S3 bucket to host the files
- An AWS CodeBuild project that will build the files
- An AWS CodePipeline pipeline that will orchestrate the CI/CD deployment
- An API Gateway that uses a Lambda function to interact with Snowflake
- IAM policies and roles required for the underlying components



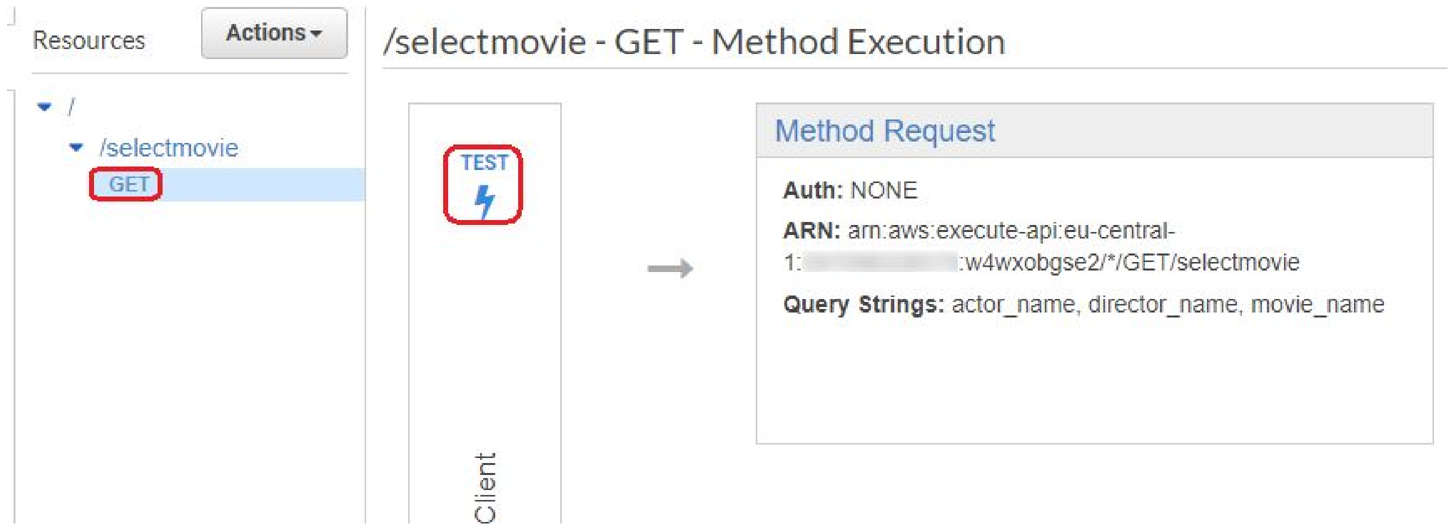
The API gateway deploys a GET method and a RESOURCE that will allow us to query the Snowflake database for movies by movie name, actor or director name.



Module 5: Test the application and deploy second version

First, let's test the application to ensure we can use the API Gateway and AWS Lambda to talk to our Snowflake warehouse and query for movies and actors.

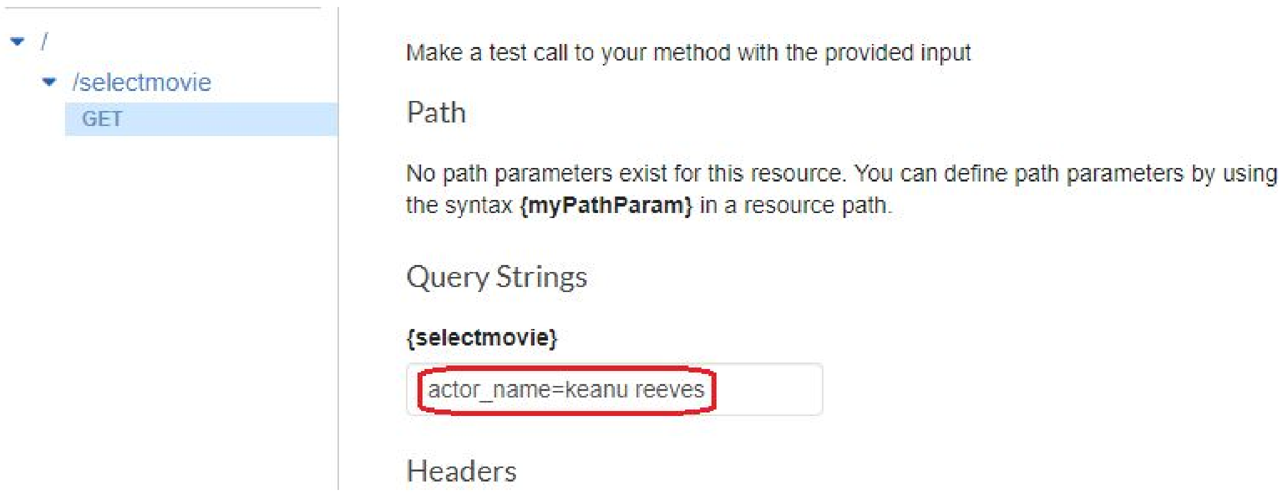
1. Go to the API Gateway console and click on **Application-backend.**
2. In the resources section go to **/selectmovie** and then to **GET,** then click on the TEST button



Auth: NONE
ARN: arn:aws:execute-api:eu-central-1:[redacted]:w4wxobgse2/*/GET/selectmovie
Query Strings: actor_name, director_name, movie_name

Let's do a search by actor name, and let's check the top 10 movies where Keanu Reeves starred in:

1. Type “**actor_name**” in the Query strings section and then the name of the actor



Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

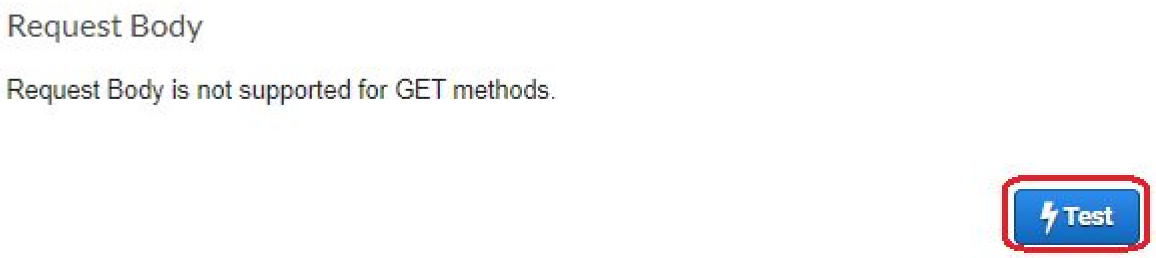
Query Strings

{selectmovie}

actor_name=keanu reeves

Headers

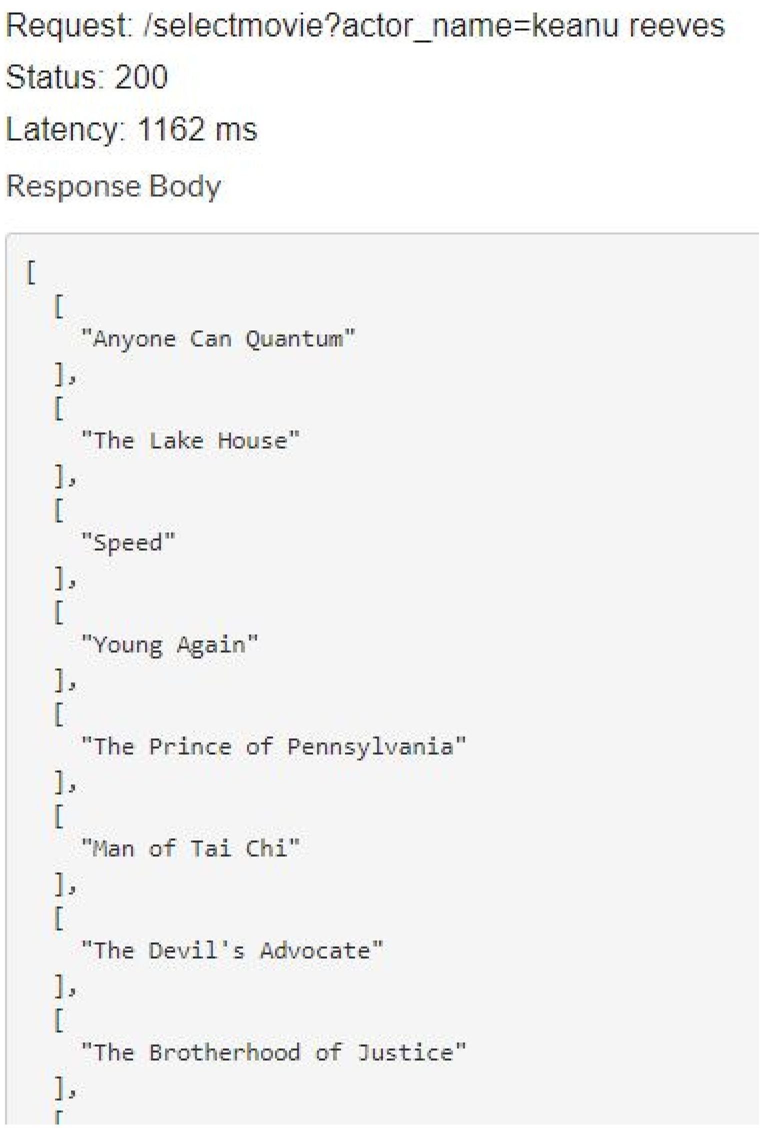
2. Click on the test button to get the results



Request Body

Request Body is not supported for GET methods.

After a few seconds, you should see the list of movies to the right, displayed in a json-like format but easy to read.



Request: /selectmovie?actor_name=keanu reeves

Status: 200

Latency: 1162 ms

Response Body

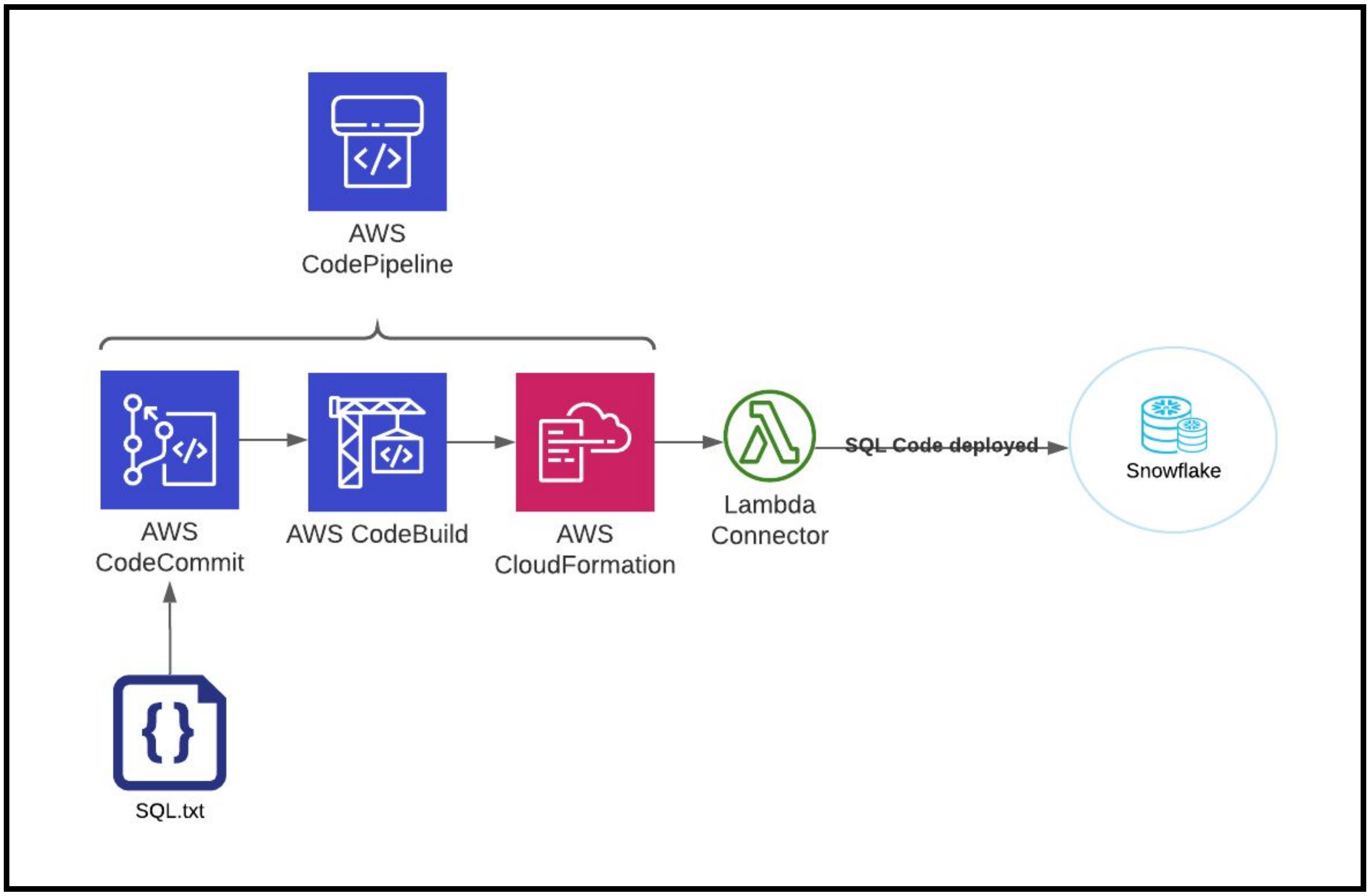
```
[
  [
    "Anyone Can Quantum"
  ],
  [
    "The Lake House"
  ],
  [
    "Speed"
  ],
  [
    "Young Again"
  ],
  [
    "The Prince of Pennsylvania"
  ],
  [
    "Man of Tai Chi"
  ],
  [
    "The Devil's Advocate"
  ],
  [
    "The Brotherhood of Justice"
  ],
```

Try searching for movies by director name, e.g. searching for **Jan de Bont** should return Speed, while searching for **Wachowski** would return the Matrix series.

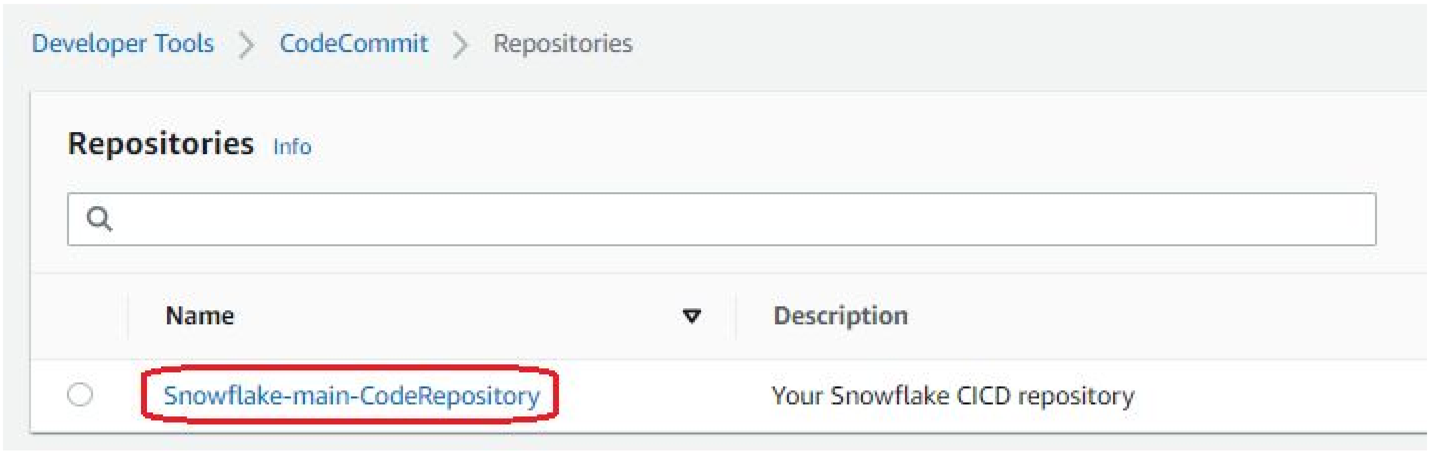
Deploying the second version

Next, we wish to enrich the data that we can retrieve and get additional information such as a movie overview, taglines, duration etc. The deployment won't require any changes in the frontend, and our database team has already prepared the necessary modifications in the backend and provided a sql.txt file with the required changes.

We are using the same Lambda function that queries



1. Open the drop-down services menu. type **CodeCommit** and click enter.
2. Click on the repository that has the description Your Snowflake CICD repository



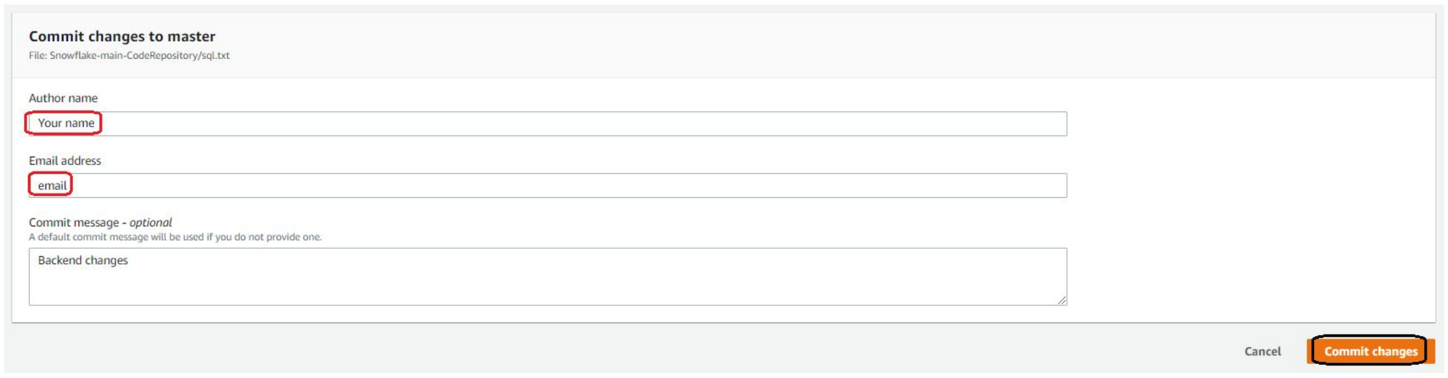
	Name	Description
○	Snowflake-main-CodeRepository	Your Snowflake CICD repository

3. In the Code section, click on the sql.txt file to open it, the file should have a single line that does nothing

4. Click on edit and then copy paste the content from the text file below into CodeCommit

https://aws-snowflake-workshop.s3.amazonaws.com/sql_v2.txt

5. In the commit changes section, complete the author name, email and optionally leave a commit message. Click **Commit changes**.



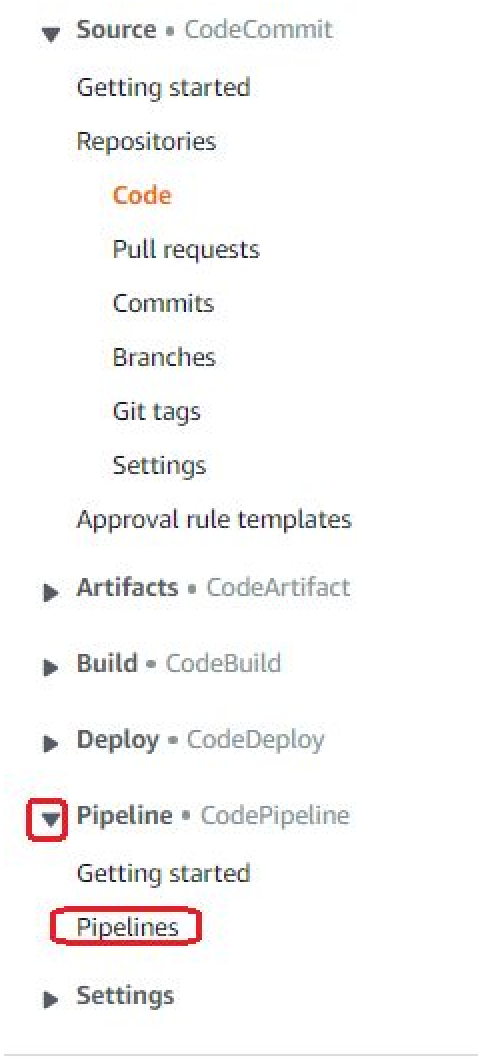
Commit changes to master
File: Snowflake-main-CodeRepository/sql.txt

Author name
Your name

Email address
email

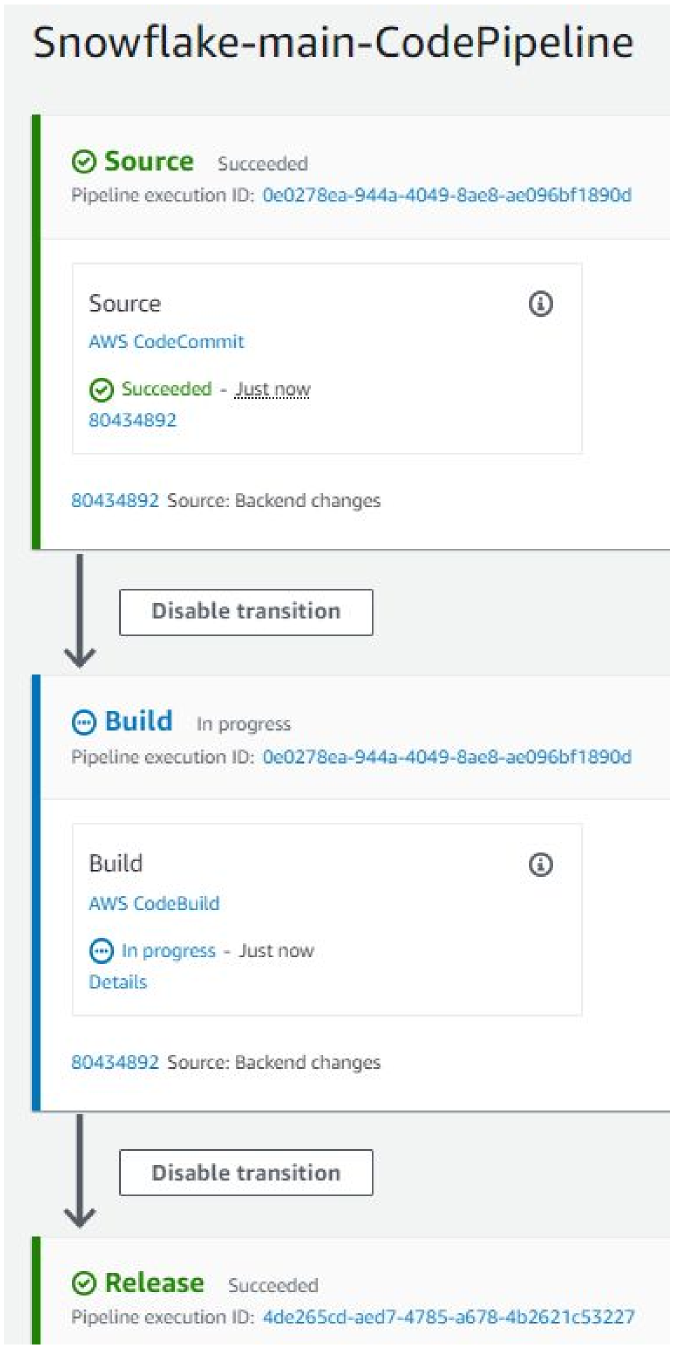
Commit message - *optional*
A default commit message will be used if you do not provide one.
Backend changes

AWS CodePipeline will now detect the changes in the CodeCommit repository and trigger the build project. To check what is happening, expand the Pipeline menu on the left side and click on Pipelines.



Alternatively click on the services drop-down bar, type CodePipeline and then enter
Click on the pipeline named ***Snowflake-main-CodePipeline*** (note, that this might be different, if you changed the stack name when you created the workshop resources).

You should see CodePipeline has already picked up the repository change and started the build process and will perform the release if successful.



Snowflake-main-CodePipeline

Source Succeeded
Pipeline execution ID: 0e0278ea-944a-4049-8ae8-ae096bf1890d

Source
AWS CodeCommit
Succeeded - Just now
80434892

80434892 Source: Backend changes

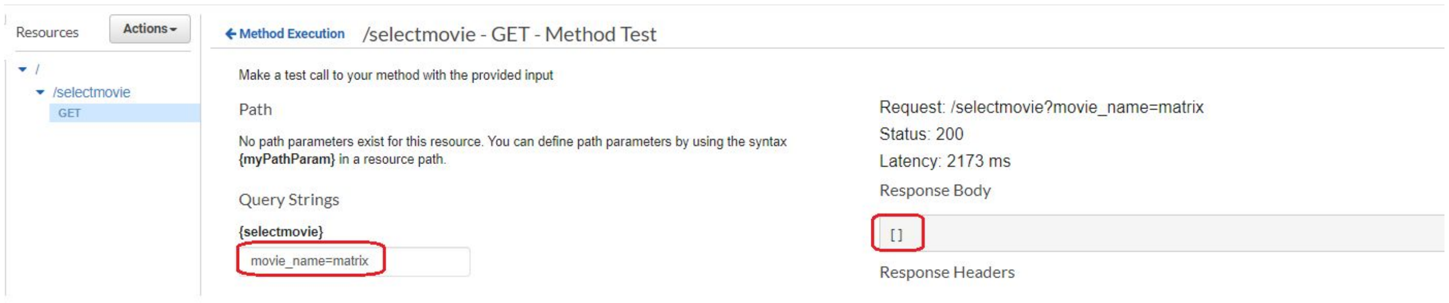
Build In progress
Pipeline execution ID: 0e0278ea-944a-4049-8ae8-ae096bf1890d

Build
AWS CodeBuild
In progress - Just now
Details

80434892 Source: Backend changes

Release Succeeded
Pipeline execution ID: 4de265cd-aed7-4785-a678-4b2621c53227

Once the pipeline has finished successfully, let's go again to the API Gateway Console and search by movie name for Matrix.



/selectmovie - GET - Method Test

Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

{selectmovie}

movie_name=matrix

Request: /selectmovie?movie_name=matrix
Status: 200
Latency: 2173 ms
Response Body

```
[]
```

Response Headers

For some reason, we aren't getting any data this time! Let's understand what happened wrong in the next module.

Module 6: Fix the faulty deployment

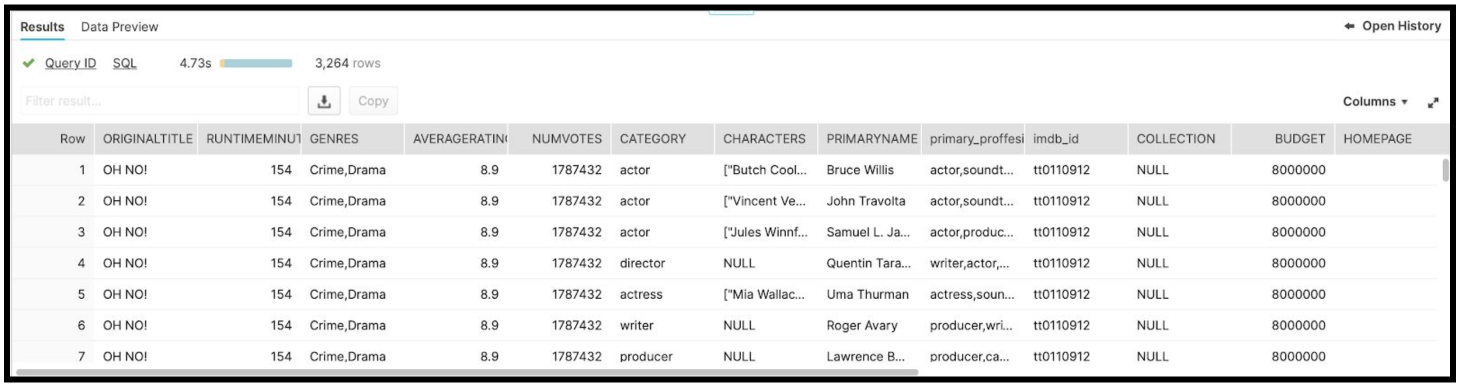
As it has been raised by the application team, recent deployment broke the data. We can confirm that we haven't implemented any unit and regression testing into our CICD solution. Usually, in situations like that we should have a fail-back plan that would allow us to revert the changes. Snowflake native capabilities fit perfectly into devops methodology. Let's review the deployment code and try to revert changes.

6.1 Understand the problem

QUERY_HISTORY!
Note that all query history can be also reviewed under ACCOUNT_USAGE schema inside of QUERY_HISTORY view or under HISTORY in SNOWFLAKE UI navi bar

6.1.1 Let's check first what happened. We can query the view and try to locate the problem. Let's run a simple query and check the results.

```
SELECT * FROM movies

WHERE averagerating IS NOT NULL
AND numvotes > 100000
ORDER BY averagerating DESC,numvotes;
```



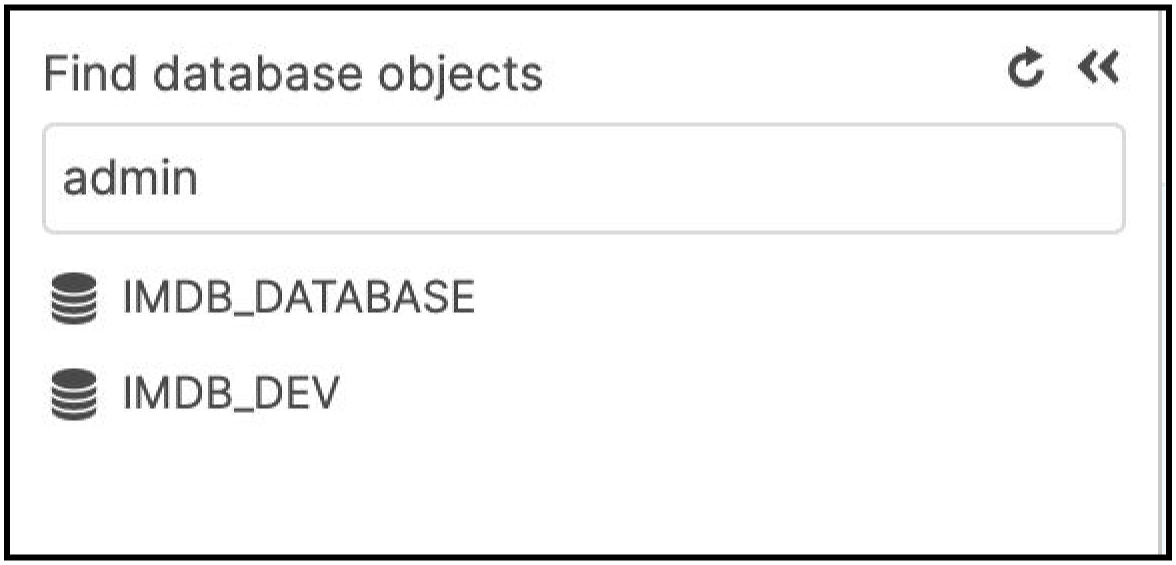
Row	ORIGINALTITLE	RUNTIMEMINUT	GENRES	AVERAGERATIN	NUMVOTES	CATEGORY	CHARACTERS	PRIMARYNAME	primary_proffesi	imdb_id	COLLECTION	BUDGET	HOMEPAGE
1	OH NO!	154	Crime,Drama	8.9	1787432	actor	["Butch Cool...	Bruce Willis	actor,soundt...	tt0110912	NULL	8000000	
2	OH NO!	154	Crime,Drama	8.9	1787432	actor	["Vincent Ve...	John Travolta	actor,soundt...	tt0110912	NULL	8000000	
3	OH NO!	154	Crime,Drama	8.9	1787432	actor	["Jules Winnf...	Samuel L. Ja...	actor,produc...	tt0110912	NULL	8000000	
4	OH NO!	154	Crime,Drama	8.9	1787432	director	NULL	Quentin Tara...	writer,actor,...	tt0110912	NULL	8000000	
5	OH NO!	154	Crime,Drama	8.9	1787432	actress	["Mia Wallac...	Uma Thurman	actress,soun...	tt0110912	NULL	8000000	
6	OH NO!	154	Crime,Drama	8.9	1787432	writer	NULL	Roger Avary	producer,wri...	tt0110912	NULL	8000000	
7	OH NO!	154	Crime,Drama	8.9	1787432	producer	NULL	Lawrence B...	producer,ca...	tt0110912	NULL	8000000	

It looks like the ORIGINALTITLE has been overwritten with "OH NO!", well, it may happen if no testing, reviews or approvals are done during deployments. Let's analyse the deployment code to understand what happened.

6.2 Analyse the code

6.2.1 As an initial deployment step, we've deployed ZERO-COPY CLONE of our IMDB_DATABASE database. CLONE is a unique solution, Snowflake users are able to clone their tables, schema or databases without creating additional copies. Snowflake stores data in files that are immutable, and encrypted, as part of architecture. The cloud services layer, with the metadata repository, records the information regarding the files being stored, the file locations, and a reference to a certain version of the data. This is also kept encrypted. In addition, when any data changes, the Metadata repository is automatically updated to provide a pointer to the changed data. All of this is performed in the background by the software without any involvement from the user. The metadata repository still retains the record for all versions of the data set.

```
CREATE OR REPLACE DATABASE IMDB_DEV CLONE IMDB_DATABASE;
```



ZERO-COPY CLONE IS EXTREMELY FAST!
Zero-copy clone doesn't move or copy any physical data and can be completed extremely quickly during deployment purposes. There is no need to keep duplicated QA, DEV, UAT environments 24/7.

6.2.2 After the ZERO-COPY CLONE has been created, the code assumed the CONTEXT of IMDB_DEV database and IMDB_SCHEMA that reflects the content of IMDB_DATABASE.IMDB_SCHEMA.

```
USE SCHEMA IMDB_DEV.IMDB_SCHEMA;
```

6.2.3 As the new data is available in JSON format, the code deployed a new FILE FORMAT for JSON files named JSON_IMDB. Data to a new table will be load using that type.

```
CREATE OR REPLACE  FILE FORMAT JSON_IMDB TYPE = 'JSON'
COMPRESSION = 'AUTO' ENABLE_OCTAL = FALSE
ALLOW_DUPLICATE = FALSE STRIP_OUTER_ARRAY = FALSE
          STRIP_NULL_VALUES = FALSE IGNORE_UTF8_ERRORS = FALSE;
```

6.2.4 For the data available in JSON format we've deployed a TABLE with a single column of VARIANT type. Snowflake has native support for semi-structured data. Of the supported file formats, JSON is one of the most widely used due to its relatively lightweight data-interchange format and the ease with which it can be written and read by both humans and machines. JSON data can be loaded directly into table columns of type VARIANT, and then queried using SQL SELECT statements that reference JSON document elements by their hierarchical paths.

```
CREATE TABLE JSON_DATA(v variant);
```

VARIANT FORMAT FLEXIBILITY!
Native Variant format can store your PARQUET, XML, JSON, AVRO, ORC files for which you can define a SCHEMA-ON-READ and still take advantage of columnar storage.

6.2.5 As the next step, our developer engineer automated the data COPY from EXTERNAL STAGE to JSON_DATA table using JSON_IMDB file format.

```
COPY INTO IMDB_DEV.IMDB_SCHEMA.JSON_DATA FROM
                @IMDB_DEV.IMDB_SCHEMA.IMDB_SOURCE_STAGE/jsons
FILE_FORMAT=(FORMAT_NAME=IMDB_DEV.IMDB_SCHEMA.JSON_IMDB);
```

6.2.6 We can query the new table using familiar SQL syntax and hierarchical paths. Notice how easy is to extract the data and filter what we need. The schema is defined ON-READ giving us full flexibility.

```
SELECT
                v:imdb_id,v:belongs_to_collection,v:budget,v:homepage,v:overview,v:pop
                ularity,v:release_date,v:revenue,v:runtime,v:tagline,v:vote_average,
                v:vote_count
 FROM JSON_DATA
WHERE GET (v,'original_title') ilike '%matrix%'
AND v:imdb_id NOT LIKE ''
AND v:revenue > 0;
```

6.2.7 To simplify querying our deployment released a code for a new VIEW called JSON_VIEW, the view extracts information with a key that can be used to join this view with our formerly created views. Collection name, budget of a movie, homepage url, short overview, popularity, release date, revenue, runtime, tagline, average rating and number of votes.

```
CREATE OR REPLACE VIEW JSON_VIEW  AS (SELECT v:imdb_id::string AS
                "imdb_id",
                  v:belongs_to_collection.name::variant AS collection,
                  v:budget::int AS budget,
                  v:homepage::varchar AS homepage,
                  v:overview::varchar AS overview,
                  v:popularity::float AS popularity,
                  v:release_date::varchar AS release_date,
                  v:revenue::int AS revenue,
                  v:runtime::integer AS runtime,
                    v:tagline::Varchar AS tagline,
                    v:vote_average::float AS vote_avg,
                    v:vote_count::int AS voters
FROM JSON_DATA
WHERE "imdb_id" NOT LIKE ''
AND revenue > 0);
```

6.2.8 For the consistency between our application and database, developer decided to merge newly created JSON_VIEW with MOVIES view and add more data to DIRECTOR and ACTOR views. This way the change is transparent for the application.

```
CREATE or REPLACE VIEW MOVIES AS
(SELECT
        originaltitle,runtimeminutes,genres,averagerating,numvotes,category,char
        acters,primaryname,

        "primary_proffesion","imdb_id",COLLECTION,BUDGET,HOMEPAGE,OV
        ERVIEW,POPULARITY,RELEASE_DATE,REVENUE,RUNTIME,TAGLIN
        E,VOTE_AVG,VOTERS
FROM title_basics b

LEFT JOIN title_ratings r
ON b.tconst=r.tconst

LEFT JOIN title_principals p
ON r.tconst=p.tconst

LEFT JOIN title_crew c
ON p.tconst = c.tconst

LEFT JOIN name_basics ba
ON p.nconst = ba.nconst

INNER JOIN json_view j
ON b.tconst = j."imdb_id"
);

CREATE OR REPLACE VIEW actor AS (SELECT distinct(TITLE),PRIMARYNAME,
        CHARACTERS,AVERAGERATING,NUMVOTES,CATEGORY FROM
        NAME_BASICS N

LEFT JOIN TITLE_PRINCIPALS P
on N.NCONST=P.NCONST

LEFT JOIN TITLE T
ON P.TCONST=T.TITLEID
JOIN TITLE_RATINGS R
ON T.TITLEID=R.TCONST

WHERE CATEGORY LIKE ANY ('actor','actress')
AND ISORIGINALTITLE='TRUE');

CREATE OR REPLACE VIEW director AS (SELECT distinct(TITLE),PRIMARYNAME,
            CHARACTERS,AVERAGERATING,NUMVOTES,CATEGORY FROM
            NAME_BASICS N

LEFT JOIN TITLE_PRINCIPALS P
on N.NCONST=P.NCONST

LEFT JOIN TITLE T
ON P.TCONST=T.TITLEID

JOIN TITLE_RATINGS R
ON T.TITLEID=R.TCONST

WHERE CATEGORY='director'
AND ISORIGINALTITLE='TRUE');
```

6.2.9 To finalize these changes our developer used SWAP command to replace IMDB_DEV name with IMDB_DATABASE, so all end-users can take advantage of changes transparently.

```
ALTER DATABASE IMDB_DATABASE SWAP WITH IMDB_DEV;
```

Warning

In real-life scenario applying a critical change like this one, should include unit testing, regression testing and additional checks or approvals and data merge. Eventually, once the code has been deployed, tested and accepted on IMDB_DEV, the same code would be deployed to IMDB_DATABASE.

6.2.10 Here we can spot the mistake made during deployment, a single UPDATE operation caused all ORIGINALTITLE fields to change to ‘OH NO!’

```
USE SCHEMA IMDB_DATABASE.IMDB_SCHEMA;
UPDATE TITLE_BASICS
    SET ORIGINALTITLE = 'OH NO!';
```

6.3 Revert the changes with TIME-TRAVEL

Traditional solutions would require an administrator to revert the changes using restores form snapshots, backups, logs or to reload the data. Snowflake has a robust set of capabilities that support companies in recovering from disasters on different levels. From undroping deleted objects with UNDROP command, multi-region replication, multi-availability zone or TIME-TRAVEL capability that we will use to retrieve ORIGINALTITLE.
Snowflake Time Travel enables accessing historical data (i.e. data that has been changed or deleted) at any point within a defined period. It serves as a powerful tool for performing the following tasks:

- Restoring data-related objects (tables, schemas, and databases) that might have been accidentally or intentionally deleted.
- Duplicating and backing up data from key points in the past.
- Analyzing data usage/manipulation over specified periods of time.

6.3.1 You will retrieve the query_id, unique query identifier allocated to every query by running a SELECT on query_history_by_warehouse function. We are looking for last UPDATE for which we would like to know the query_id this will be the UPDATE made during deployment. We want to store the retrieved query_id as a variable QUERY_ID

```
SET query_id =
  (SELECT query_id
FROM TABLE(information_schema.query_history_by_warehouse(result_limit => 100))
WHERE query_text like 'UPDATE%' ORDER BY start_time DESC LIMIT 1);
```

6.3.2 We can display the populated VARIABLE with simple SELECT

```
SELECT $query_id;
```

6.3.3 Knowing the query_id of the UPDATE we can compare the current state of our table with the state of our table before the query_id was committed. This feature takes advantage of TIME-TRAVEL feature that holds the story of a table up to 90 days back. You can retrieve or restore the information from the past using statements like BEFORE, AT, AFTER.

```
WITH a AS (SELECT * FROM IMDB_DATABASE.IMDB_SCHEMA.MOVIES LIMIT 10),
b AS    (SELECT * FROM IMDB_DATABASE.IMDB_SCHEMA.MOVIES  BEFORE
                (statement => $query_id) LIMIT 10)
SELECT * FROM a UNION ALL SELECT  * FROM b;
```

6.3.4 Now, you will restore the TITLE_BASICS table to TITLE_BASICS_CLONE table using the CLONE function retrieving the TITLE_BASICS table state from TIME-TRAVEL before our UPDATE happened.

```
CREATE TABLE IMDB_DATABASE.IMDB_SCHEMA.TITLE_BASICS_CLONE CLONE
          IMDB_DATABASE.IMDB_SCHEMA.TITLE_BASICS BEFORE
          (STATEMENT => $query_id);
```

6.3.5 We should check the TITLE_BASICS_CLONE table with basic SELECT to confirm if proper data is in ORIGINALTITLE column.

```
SELECT * FROM IMDB_DATABASE.IMDB_SCHEMA.TITLE_BASICS_CLONE
limit 10;
```

6.3.6 As you noticed, data in TITLE_BASICS_CLONE looks right. You will now SWAP the TITLE_BASICS table with TITLE_BASICS_CLONE table using ALTER TABLE statement.

```
ALTER TABLE  IMDB_DATABASE.IMDB_SCHEMA.TITLE_BASICS SWAP WITH
          IMDB_DATABASE.IMDB_SCHEMA.TITLE_BASICS_CLONE;
```

6.3.7 You can DROP the TITLE_BASICS_CLONE table now. We can still retrieve it with UNDROP if required.

```
DROP TABLE IMDB_DATABASE.IMDB_SCHEMA.TITLE_BASICS_CLONE;
```

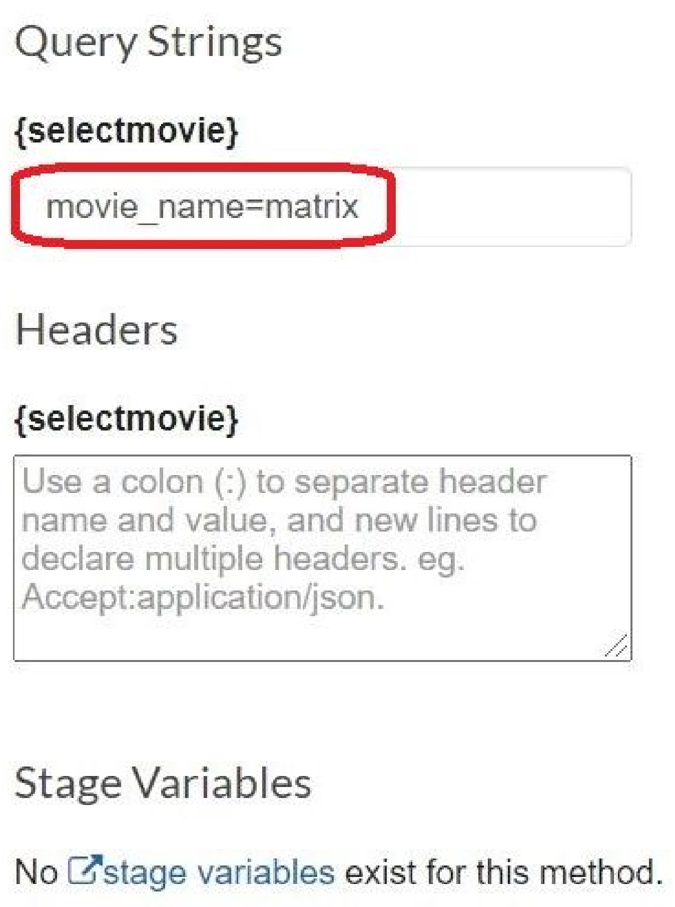
6.4 Test the application

Now that we’ve identified the “problem” that caused our deployment, let’s re-test again and see if we can get results with the additional data.

1. Go to the API Gateway console and click on **Application-backend.**
2. In the resources section go to **/selectmovie** and then to **GET,** then click on the TEST button

Let’s repeat the same search as before:

3. Type “**movie_name**” in the Query strings section and then type Matrix



Query Strings

{selectmovie}

movie_name=matrix

Headers

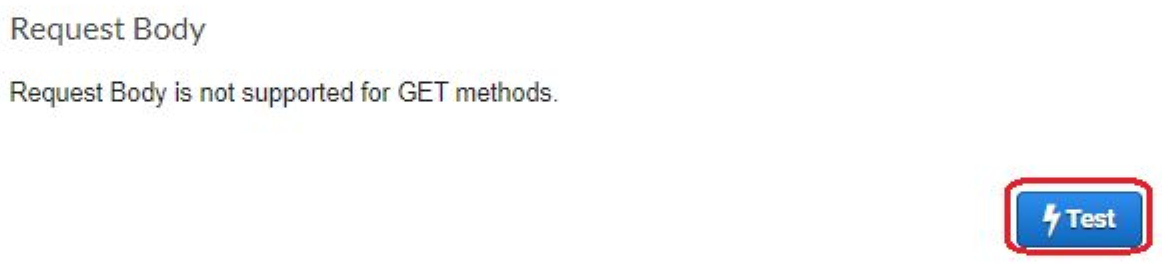
{selectmovie}

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.

Stage Variables

No stage variables exist for this method.

4. Click on the test button to get the results

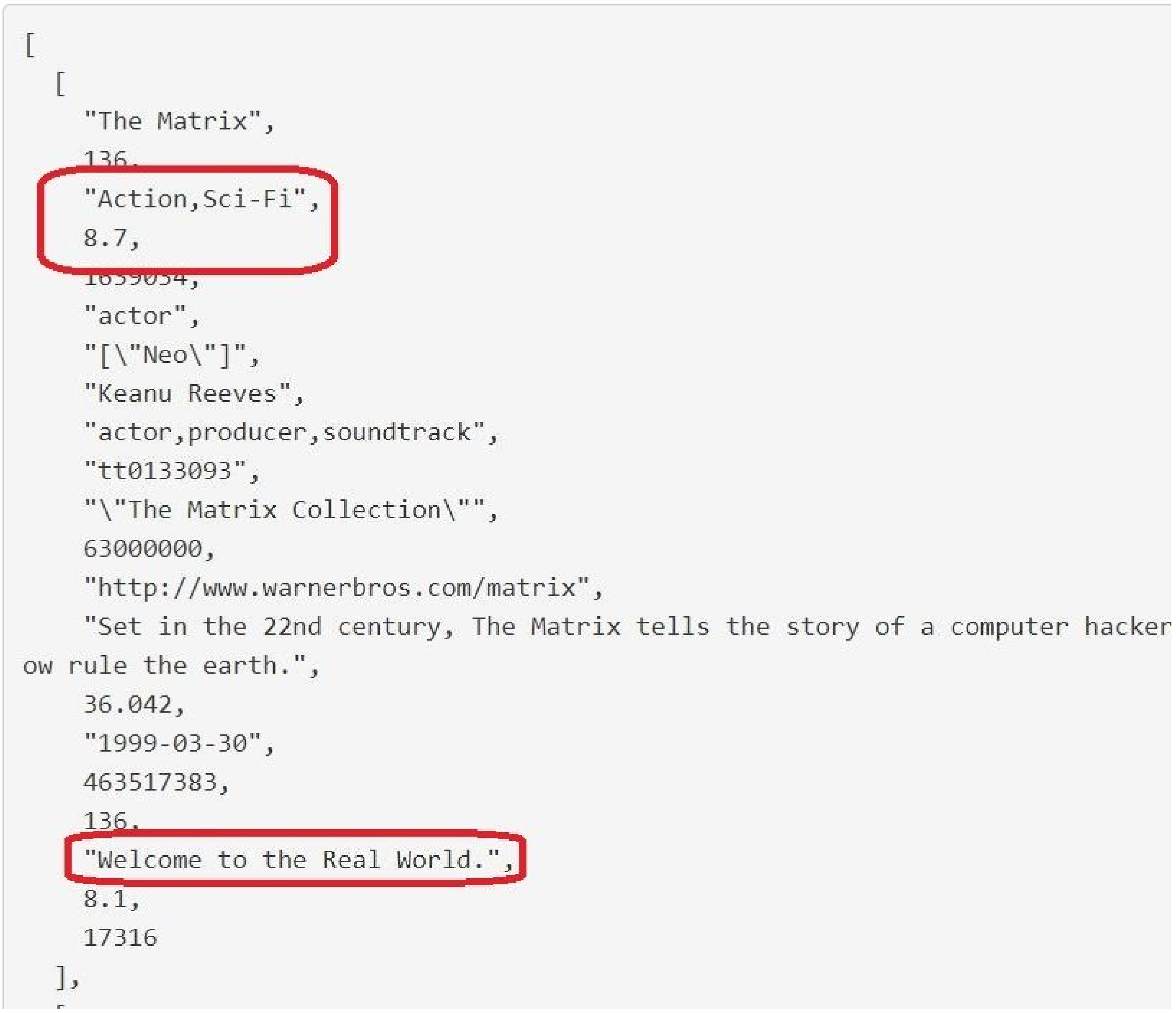


Request Body

Request Body is not supported for GET methods.

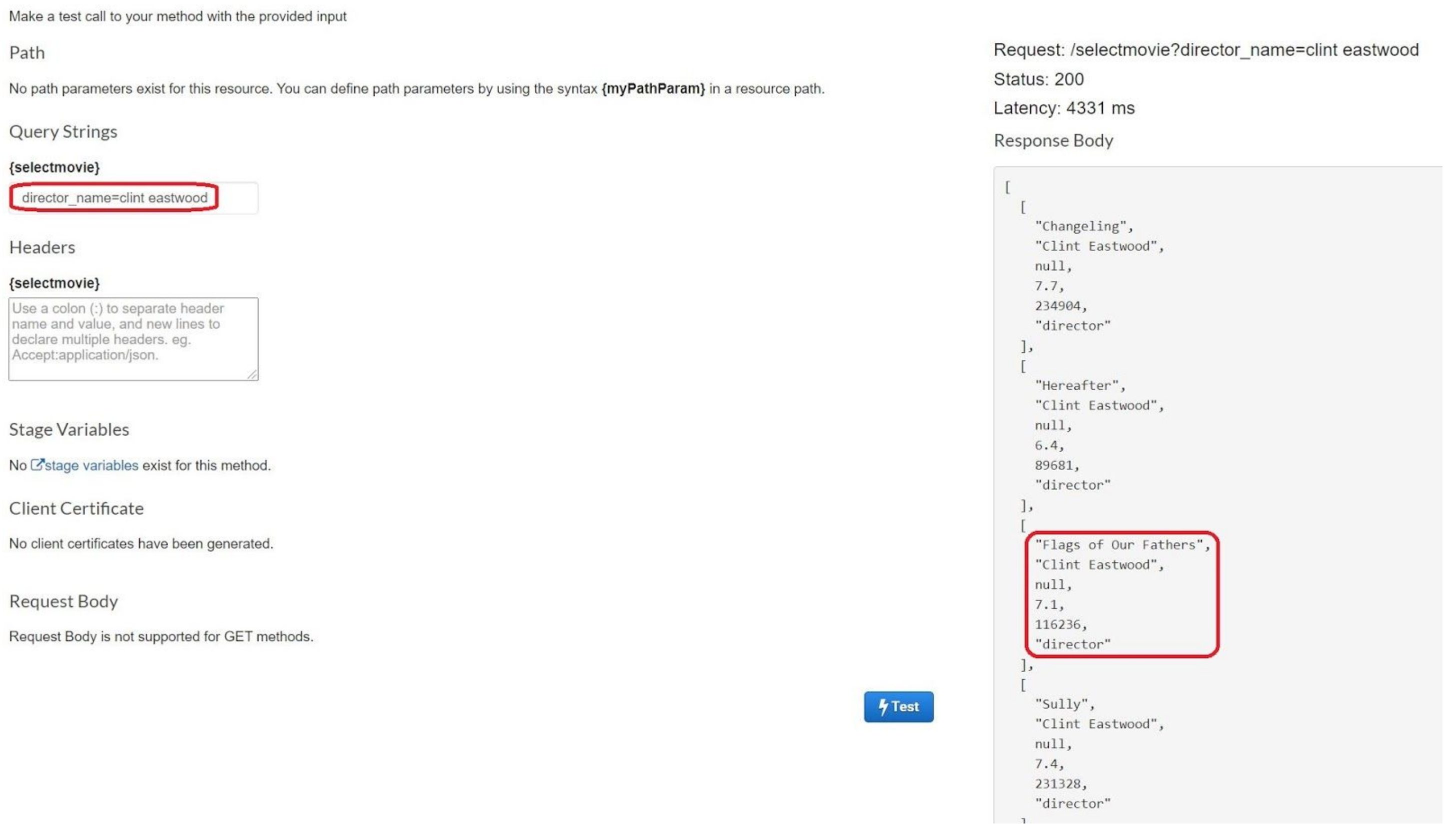
After a few seconds, you should see the list of movies with the enriched data such as run time, genre, average rating, category, tagline and description.

Response Body



```
[
  [
    "The Matrix",
    136,
    "Action,Sci-Fi",
    8.7,
    1639034,
    "actor",
    "[\"Neo\"]",
    "Keanu Reeves",
    "actor,producer,soundtrack",
    "tt0133093",
    "\"The Matrix Collection\"",
    63000000,
    "http://www.warnerbros.com/matrix",
    "Set in the 22nd century, The Matrix tells the story of a computer hacker
ow rule the earth.",
    36.042,
    "1999-03-30",
    463517383,
    136,
    "Welcome to the Real World.",
    8.1,
    17316
  ],
  [
```

Similarly, try searching for movies by director name, e.g. searching for **Clint Eastwood** should return additional fields such as the average rating for the movie directed or the number of votes on IMDB.


Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

{selectmovie}

director_name=clint eastwood

Headers

{selectmovie}

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.

Stage Variables

No stage variables exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

Request Body is not supported for GET methods.

Test

Request: /selectmovie?director_name=clint eastwood

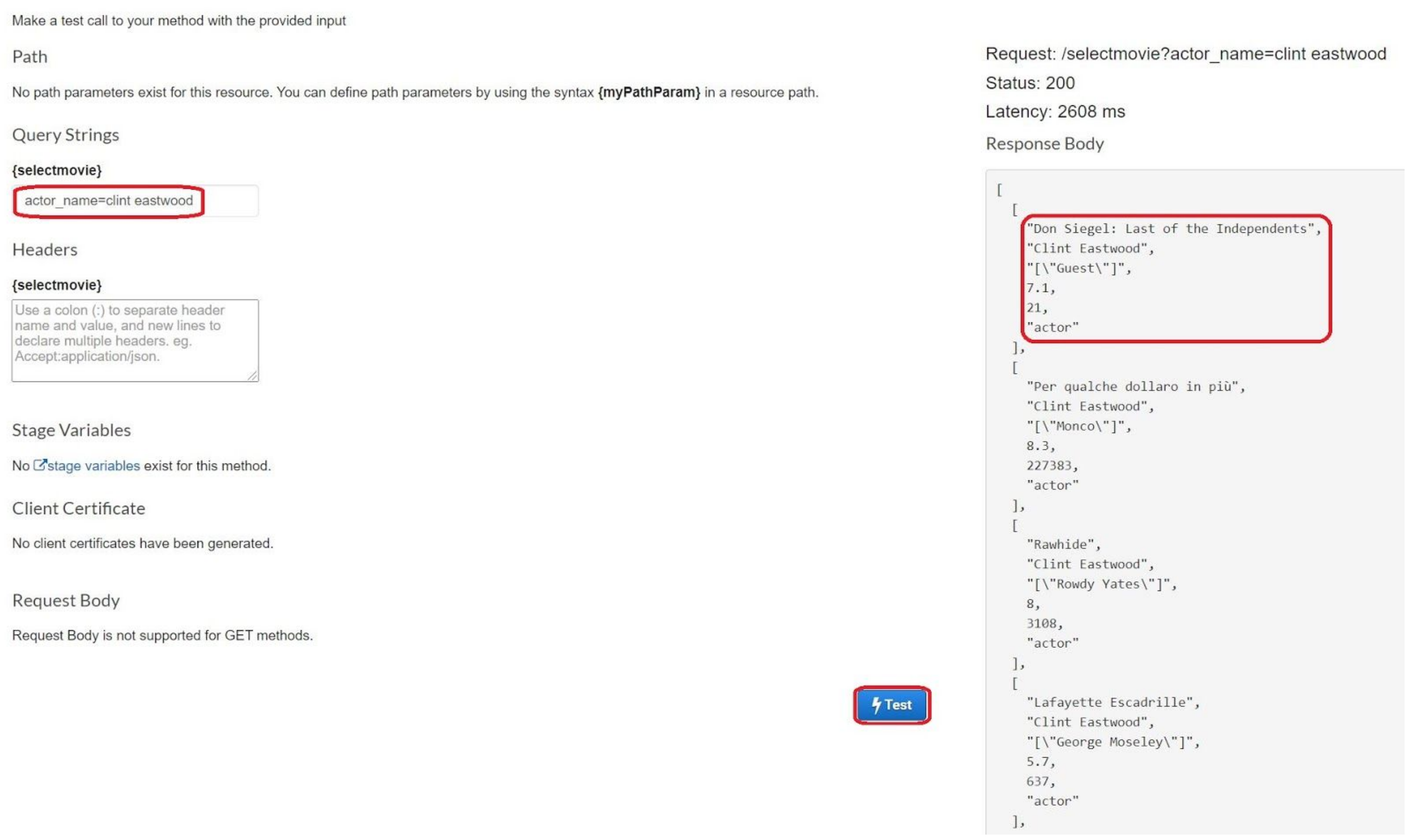
Status: 200

Latency: 4331 ms

Response Body

```
[
  [
    "Changeling",
    "Clint Eastwood",
    null,
    7.7,
    234904,
    "director"
  ],
  [
    "Hereafter",
    "Clint Eastwood",
    null,
    6.4,
    89681,
    "director"
  ],
  [
    "Flags of Our Fathers",
    "Clint Eastwood",
    null,
    7.1,
    116236,
    "director"
  ],
  [
    "Sully",
    "Clint Eastwood",
    null,
    7.4,
    231328,
    "director"
```

Let’s now search for results for **Clint Eastwood** but instead of searching for movies he directed, let’s look at movies he acted in.
Change the query string to ***actor_name=Clint Eastwood*** and click on the test button.


Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

{selectmovie}

actor_name=clint eastwood

Headers

{selectmovie}

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.

Stage Variables

No stage variables exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

Request Body is not supported for GET methods.

Request: /selectmovie?actor_name=clint eastwood

Status: 200

Latency: 2608 ms

Response Body

```
[
  [
    "Don Siegel: Last of the Independents",
    "Clint Eastwood",
    "[\"Guest\"]",
    7.1,
    21,
    "actor"
  ],
  [
    "Per qualche dollaro in più",
    "Clint Eastwood",
    "[\"Monco\"]",
    8.3,
    227383,
    "actor"
  ],
  [
    "Rawhide",
    "Clint Eastwood",
    "[\"Rowdy Yates\"]",
    8,
    3108,
    "actor"
  ],
  [
    "Lafayette Escadrille",
    "Clint Eastwood",
    "[\"George Moseley\"]",
    5.7,
    637,
    "actor"
  ],
```

We can see **Clint Eastwood** was an actor in a movie called ***Don Siegel: Last of the Independents***, where he played the role of *Guest*, the movie has a rating of **7.1** and 21 votes on IMDB.

Congratulations, you are now done with this lab! Let's wrap things up in the next, and final, section.

Module 7: Bonus: Test concurrency

In this section facilitators will run a concurrency test on Snowflake via Amazon API Gateway using an Amazon solution for Distributed Load testing and Apache Jmeter. The configuration and execution of this section is not in scope of online training but will be presented to highlight scaling capabilities of Amazon serverless services and Snowflake multi-cluster configuration.

For better understanding of the topic, please take a look at the command executed by the Snowflake trainer.

7.1.1 Set context to Accountadmin role and turn off Results Cache

```
USE ROLE accountadmin;
```

```
ALTER ACCOUNT SET use_cached_result=false;
```

7.1.2 Alter IMDB_WH warehouse to scale out&in automatically if there is a demand for more concurrent query executions, we expect at least 100 concurrent executions

```
ALTER WAREHOUSE "IMDB_WH" SET WAREHOUSE_SIZE = 'XSMALL'
AUTO_SUSPEND = 60 AUTO_RESUME = TRUE
MIN_CLUSTER_COUNT = 1 MAX_CLUSTER_COUNT = 10
SCALING_POLICY = 'STANDARD' ;
```

7.1.3 Upload a JMX file with valid REST API URL to Distributed Load Testing Console and launch a JMETER test against the application.

7.1.4 Take a look at Snowflake history and warehouse menu to understand how warehouses are satisfying the need for more concurrency.

Summary & Next Steps

This tutorial was designed as a hands-on introduction to Snowflake to simultaneously teach you how to use it, while showcasing some of its key capabilities and differentiators. We covered how to navigate the UI, create databases and warehouses, load & query structured and semi-structured data, perform zero-copy cloning, undo user errors, RBAC, and data sharing.

We encourage you to continue with your free trial by loading in your own sample or production data and by using some of the more advanced capabilities of Snowflake not covered in this lab. There are several ways Snowflake can help you with this:

- At the very top of the UI click on the "Partner Connect" icon to get access to trial/free ETL and BI tools to help you get more data into Snowflake and then analyze it
- Read the "Definitive Guide to Maximizing Your Free Trial" document at: https://www.snowflake.com/test-driving-snowflake-the-definitive-guide-to-maximizing-your-free-trial/
- Attend a Snowflake virtual or in-person event to learn more about our capabilities and how customers use us https://www.snowflake.com/about/events/
- Contact Sales to learn more https://www.snowflake.com/free-trial-contact-sales/

Resetting Your Snowflake Environment

Lastly, if you would like to reset your environment by deleting all the objects created as part of this lab, run the SQL below in a worksheet.

Run this SQL to set the worksheet context:

```
USE ROLE accountadmin;
```

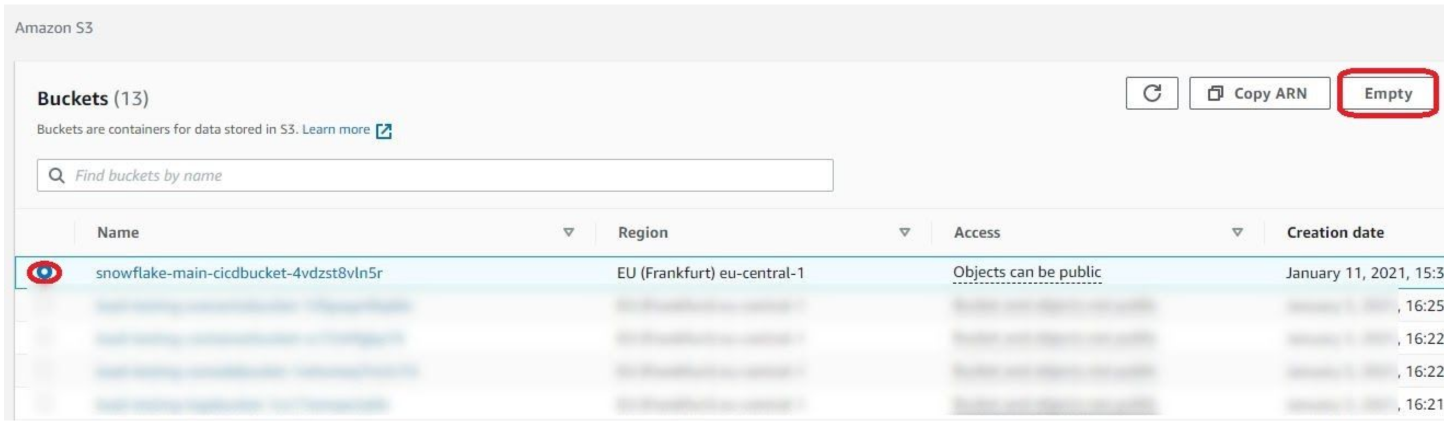
Then run this SQL to drop all the objects we created in the lab:

```
DROP DATABASE IF EXISTS IMDB_DATABASE;
DROP DATABASE IF EXISTS IMDB_DEV;
DROP WAREHOUSE IF EXISTS IMDB_WH;
DROP ROLE IF EXISTS IMDB_ROLE;
DROP USER IF EXISTS IMDB_ENGINEER;
```

Cleaning-up your AWS Environment

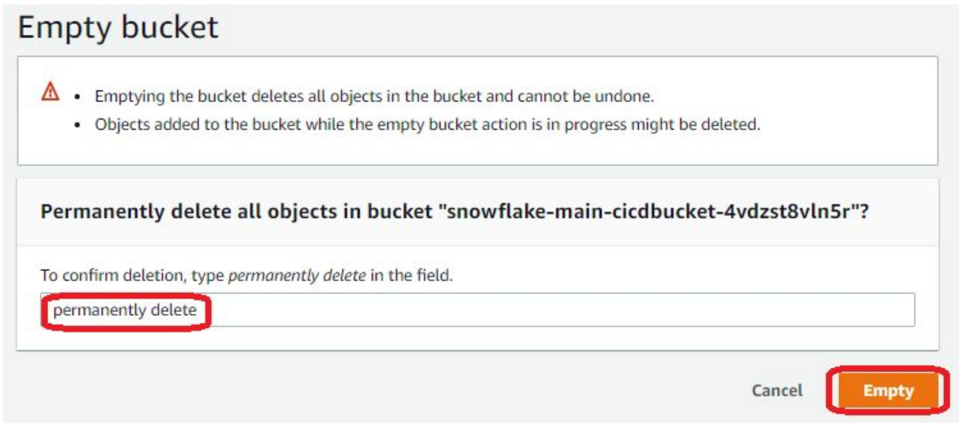
1. Delete S3 objects in the workshop bucket

    a. Open the AWS CloudFormation Console and select the stack named Snowflake-main (note, that this might be different, if you changed the stack name when you created the workshop resources)
    b. Find the cicdbucket Logical ID under the Resources tab, and make a note of the bucket name
    c. Open the Amazon S3 console and check the box next to the bucket and click on empty



Name	Region	Access	Creation date
snowflake-main-cicdbucket-4vdzst8vln5r	EU (Frankfurt) eu-central-1	Objects can be public	January 11, 2021, 15:3
[illegible]	[illegible]	[illegible]	[illegible], 16:25
[illegible]	[illegible]	[illegible]	[illegible], 16:22
[illegible]	[illegible]	[illegible]	[illegible], 16:22
[illegible]	[illegible]	[illegible]	[illegible], 16:21

    d. In the empty bucket screen, type "**permanently delete**" (without quotes)
    e. Click on the **empty** button


Empty bucket

- Emptying the bucket deletes all objects in the bucket and cannot be undone.
- Objects added to the bucket while the empty bucket action is in progress might be deleted.

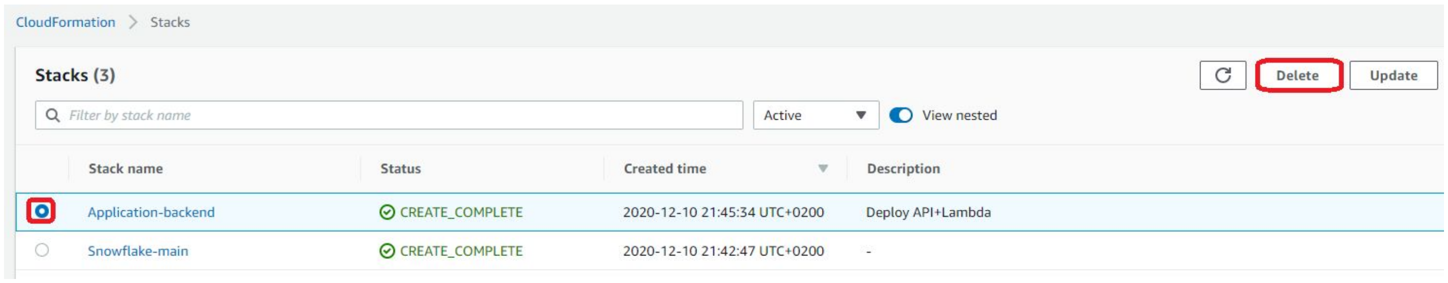
Permanently delete all objects in bucket "snowflake-main-cicdbucket-4vdzst8vln5r"?

To confirm deletion, type *permanently delete* in the field.

permanently delete

    f. Return to the CloudFormation Console.

2. Delete the AWS CloudFormation Stacks
    a. Select the stack named Application-backend and click the **Delete** button to delete the stack
    b. Select the stack named Snowflake-Main and click the **Delete** button to delete the stack (note, that this might be different, if you changed the stack name when you created the workshop resources)



Stacks (3)

Stack name	Status	Created time	Description
Application-backend	CREATE_COMPLETE	2020-12-10 21:45:34 UTC+0200	Deploy API+Lambda
Snowflake-main	CREATE_COMPLETE	2020-12-10 21:42:47 UTC+0200	-

That's it! All done.