**Olympic Data Analysis Using Azure**

Tools Used: MySQL, HDFS, Sqoop, Hive, HBase, Snowflake, Snowpipe, Azure Data Factory, Azure Databricks, AWS, PowerBI

Project Description:

* There were two datasets that had information about the Olympic games and the athlete's performance.
* The main objective was to build different kinds of pipelines to analyze the Olympic data.
* In the first pipeline, we loaded the data from the local file system into MySQL tables, and then using the SQOOP import we moved the data into HIVE tables.
* We partitioned the Hive tables based on the country and game types for faster query performance.
* Then we analyzed the datasets using Hive Queries.
* In the second pipeline, we utilized Azure and Snowflake. Data was stored in Azure and using copyInto activity we copied the data from ADLS to the snowflake, then analyzed the data using the snowflake query.
* In the third pipeline, we utilized AWS and Snowflake. Data was stored in AWS and using snowpipe activity we copied the data from S3 to the snowflake, then analyzed the data using the snowflake query. The advantage of Snowpipe is that whenever data is available at the source it will load that data into the target automatically.
* Finally we connected the snowflake with PowerBI to visualize the Olympic data and draw insight from it.

**Dataset Info:**

**Athletes\_event:**

15 columns and 2,71,117 rows

* Id: Unique ID of each athlete
* Name: Name of the athlete
* Sex: Gender of the athlete
* Age: Age of the athlete
* Height: Height of the athlete
* Weight: Weight of the athlete
* Team: The Country belongs to
* Noc: Short Code for the country
* Games: Year and Type of Olympic (summer or winter)
* Year: The year in which the Olympics held
* Season: Summer or winter
* City: The city where the Olympics held
* Sport: The sport in which the athlete participated
* Event: The complete Event info
* Medal: Gold, Silver, Bronze or NA

**Noc\_Regions:**

3 columns, 230 rows

* Noc: Shortcode for country
* Region: Country name

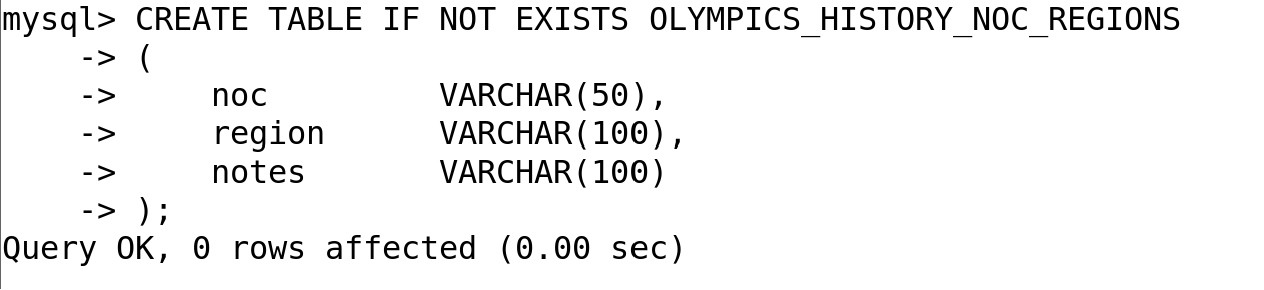
**Pipeline 1: Importing data from MySQL into HIVE using SQOOP import.**

1. **Create database and tables in MySQL**

**CREATE** **TABLE** **IF** **NOT** **EXISTS** OLYMPICS\_HISTORY  
( **id** INT, **name** VARCHAR(300), sex VARCHAR(10), age varchar(20), height varchar(20), weight varchar(20), team VARCHAR(50), noc VARCHAR(50), games VARCHAR(50), **year** INT, season VARCHAR(50), city VARCHAR(50), sport VARCHAR(100), **event** VARCHAR(200), medal VARCHAR(50) );



**CREATE** **TABLE** **IF** **NOT** **EXISTS** OLYMPICS\_HISTORY\_NOC\_REGIONS  
( noc VARCHAR(50), region VARCHAR(100), notes VARCHAR(100) );

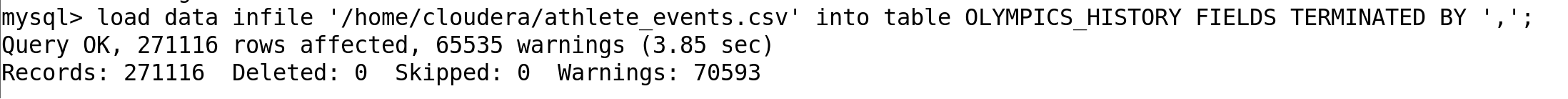


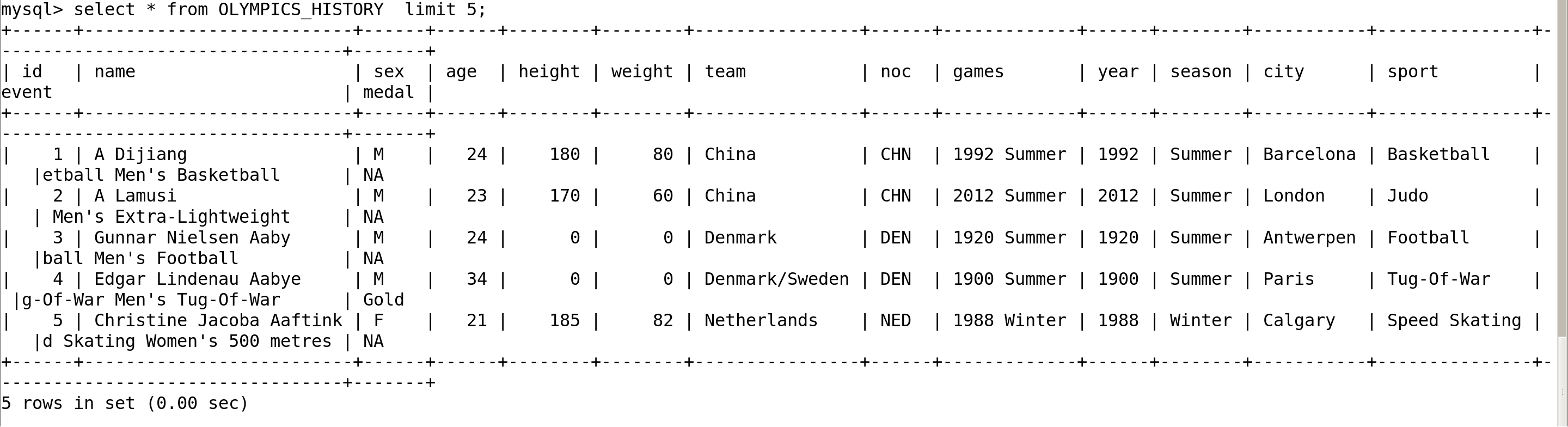
1. **Load data from the local file system into MySQL table**

**load** **data** **infile** '/home/cloudera/athlete\_events.csv'

**into** **table** OLYMPICS\_HISTORY

**FIELDS** **TERMINATED** **BY** ',';



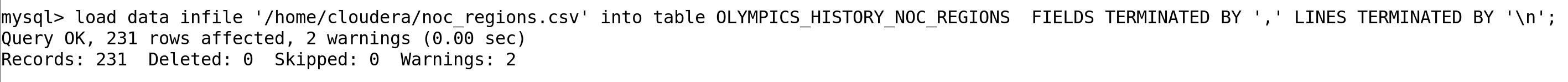


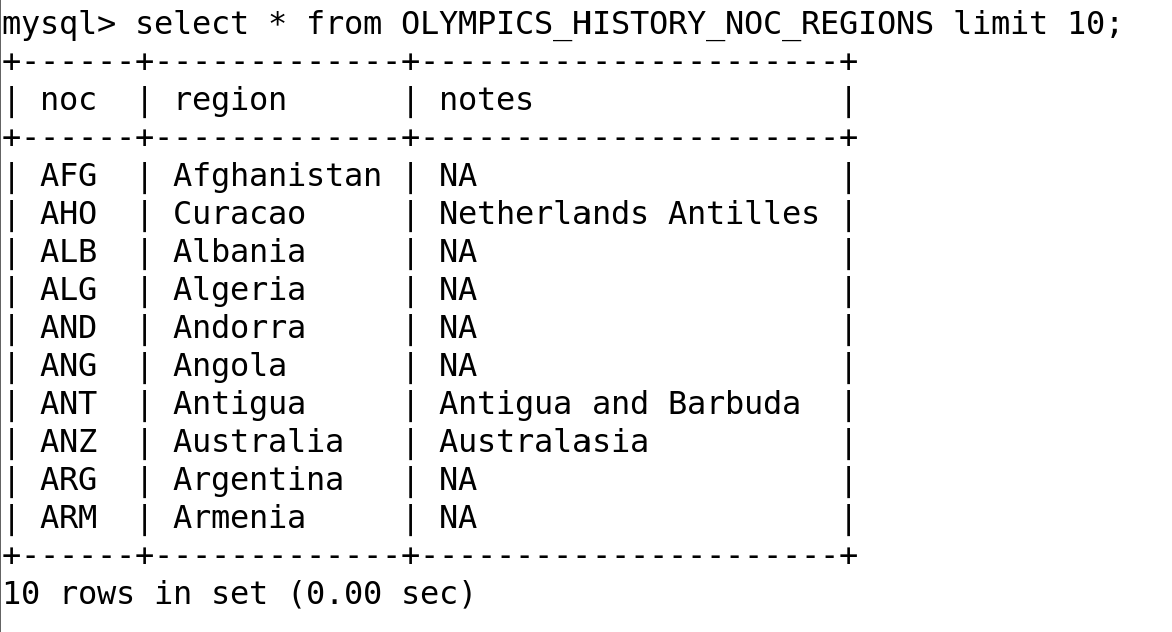
**load** **data** **infile** '/home/cloudera/noc\_regions.csv'

**into** **table** OLYMPICS\_HISTORY\_NOC\_REGIONS

**FIELDS** **TERMINATED** **BY** ','

**LINES** **TERMINATED** **BY** '\n';





1. **Load the data from MySQL table into Hive tables using Sqoop Import**

**This will directly import the table into the hive with the table name OLYMPICS\_HISTORY**

sqoop import

--connect jdbc:mysql://localhost:3306/olympic

--username root -P

--table OLYMPICS\_HISTORY

--hive-import

-m 1

**If we want to import the MySQL table into a specific database and table in hive we can use this command**

sqoop import

--connect jdbc:mysql://localhost:3306/olympic

--username root -P

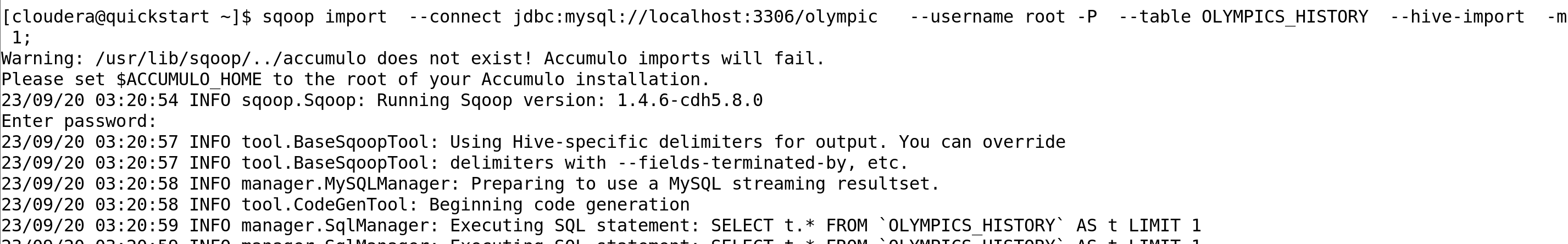
--table OLYMPICS\_HISTORY

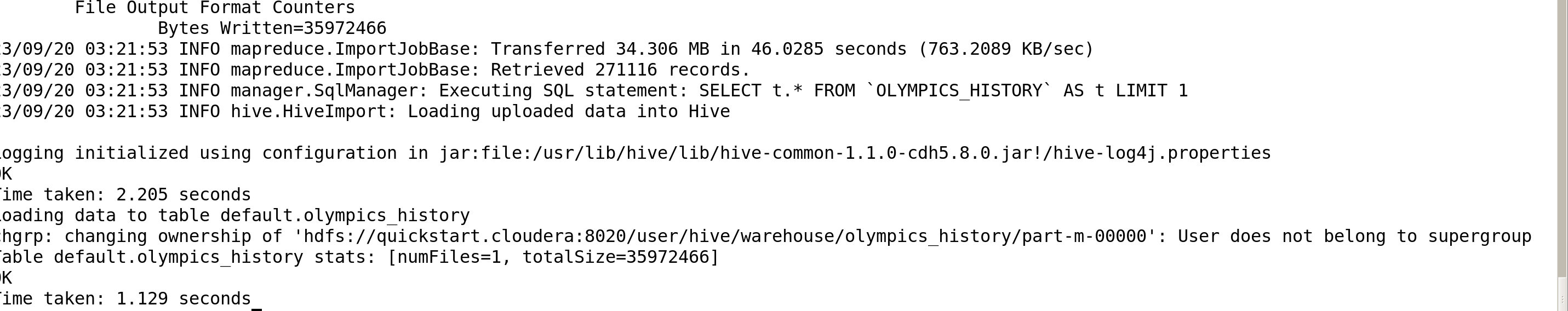
--hive-import

--hive-database olympic\_hive

--hive-table olympic\_**history**

-m 1;





sqoop import

--connect jdbc:mysql://localhost:3306/olympic

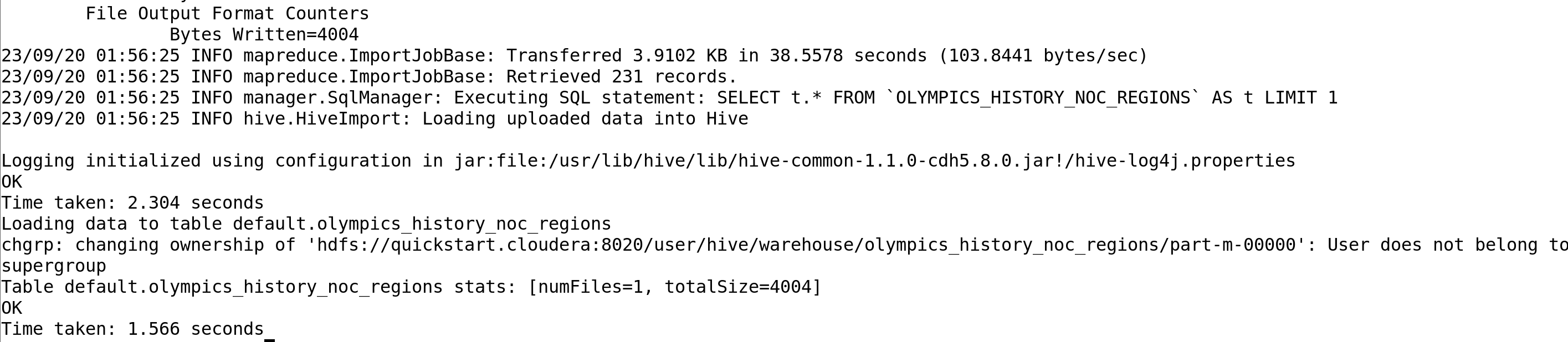
--username root -P

--table OLYMPICS\_HISTORY\_NOC\_REGIONS

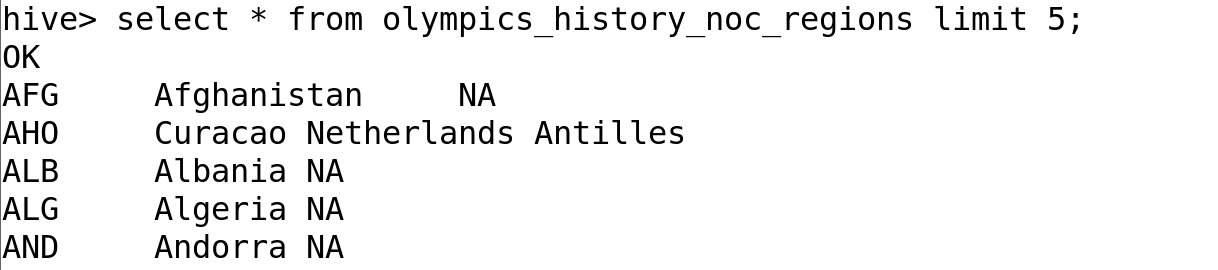
--hive-import

-m 1





27 sec, 1 map 0 reduce



1. **Partitioning the Hive table for optimization purposes**

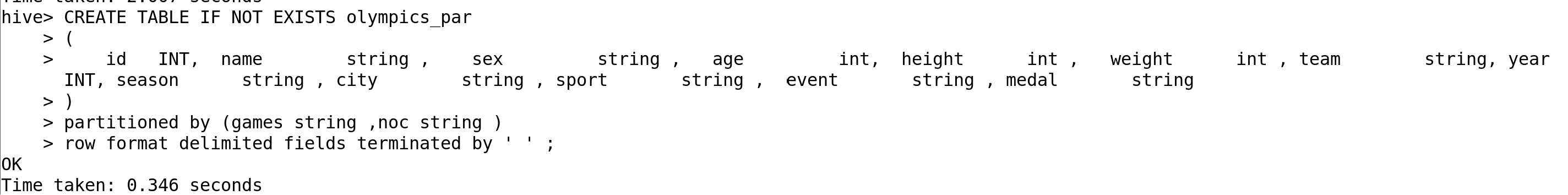
**Distinct Noc count – 230**

**Distinct Games count - 51**

**SET hive.exec.dynamic.partition = true ;  
SET hive.exec.dynamic.partition.mode = nonstrict ;**

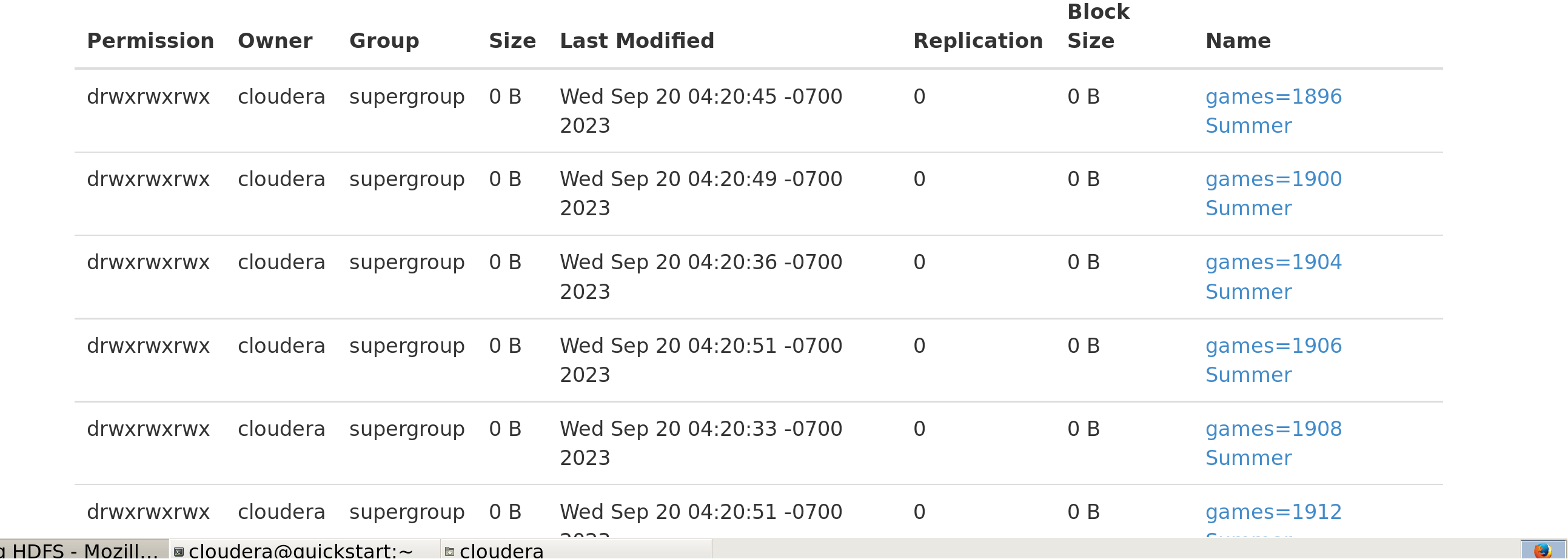
**SET hive.exec.max.dynamic.partitions = 5000;  
SET hive.exec.max.dynamic.partitions.pernode = 5000;**

**CREATE** **TABLE** **IF** **NOT** **EXISTS** olympics\_par1  
(**id** INT, **name** **string**, sex **string**, age string, height string, weight string, team **string**, **year** INT, season **string**, city **string**, sport **string**, **event** **string**, medal **string**)  
partitioned **by** (games **string**, noc **string** )   
**row** **format** **delimited** **fields** **terminated** **by** ',' ;

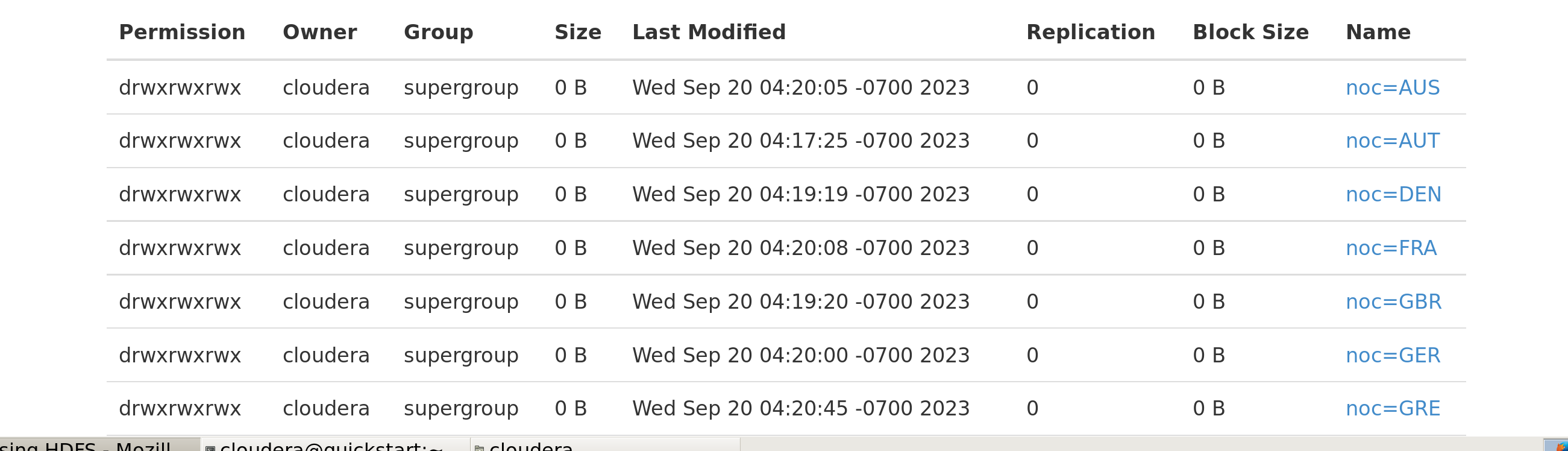


**insert** overwrite **table** olympics\_par1 **partition** (games, noc)

**select** **id**, **name**, sex, age, height, weight, team, **year**, season, city, sport, **event**, medal, games, noc **from** OLYMPICS\_HISTORY

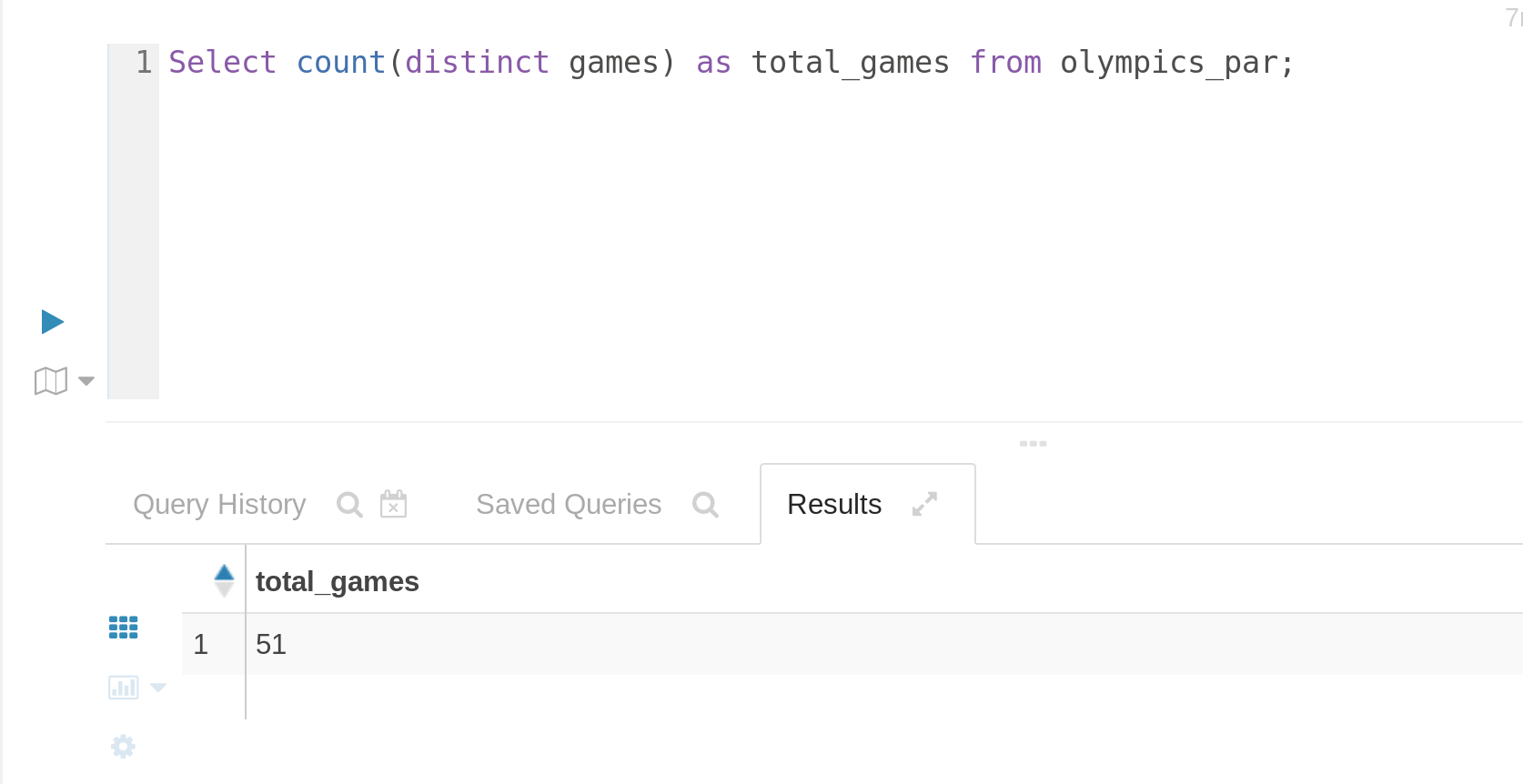


51 partitions



1. **Solving the client Questions using the HiveQL**
2. **How many Olympics games have been held?**

**Select** count(**distinct** games) **from** olympics\_par;



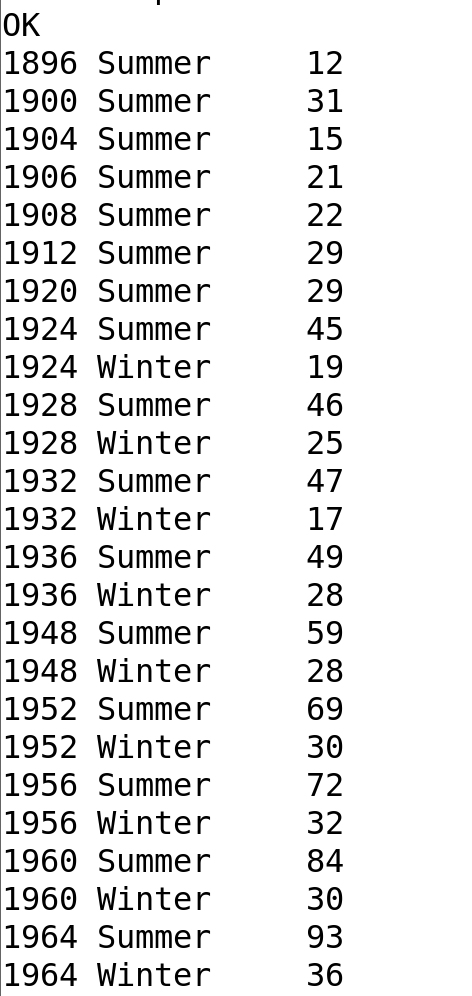
1. **List down all Olympics games held so far.**

Select distinct games **from** Olympics\_par1



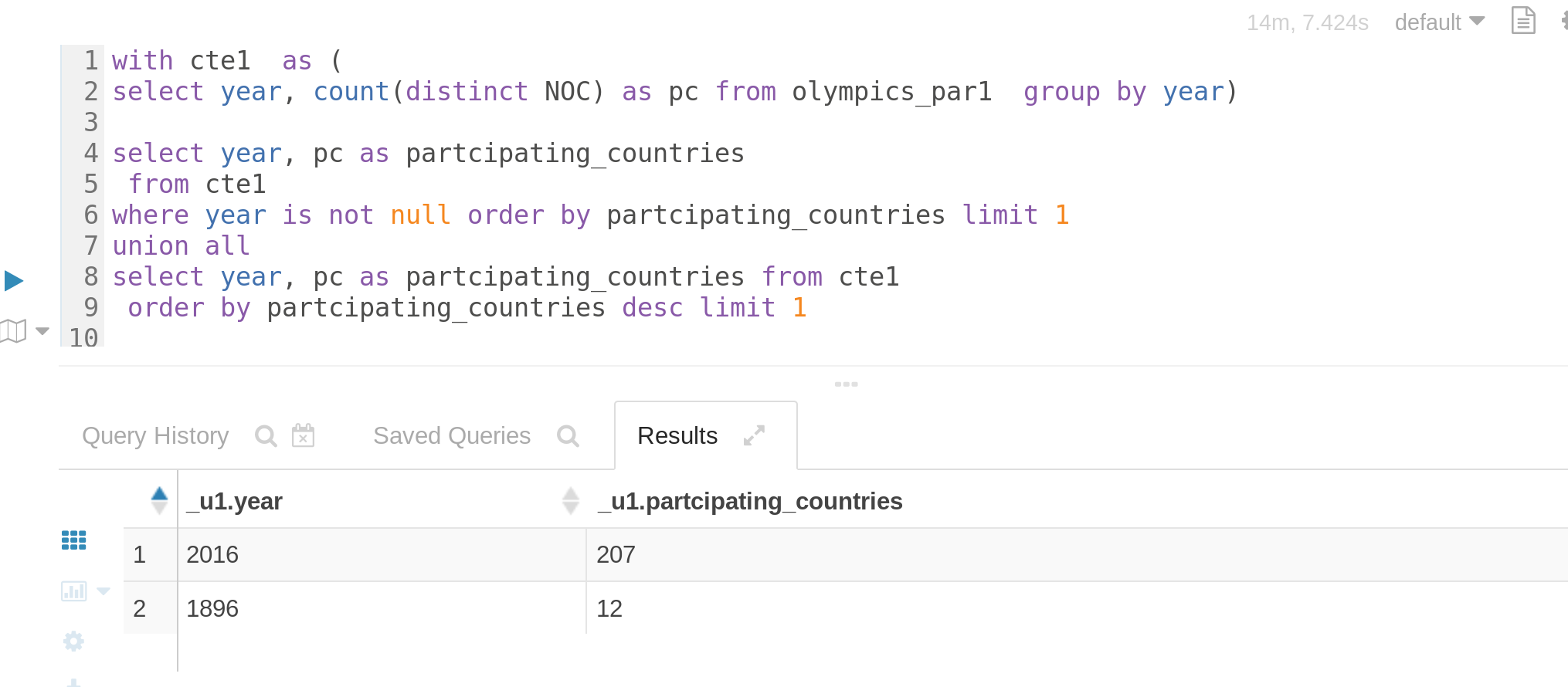
1. **Mention the total no of nations who participated in each olympics game?**

Select games , count(distinct noc) as total\_no\_nations **from** Olympics\_par1 group by games;



1. **Which year saw the highest and lowest no of countries participating in olympics?**

**with** cte1 **as** (  
**select** **year**, **count**(**distinct** NOC) **as** pc **from** olympics\_par1   
**group** **by** **year**)  
  
**select** **year**, pc **as** partcipating\_countries  
**from** cte1   
**where** **year** **is** **not** null **order** **by** partcipating\_countries **limit** 1  
**union** **all**  
**select** **year**, pc **as** partcipating\_countries **from** cte1  
**order** **by** partcipating\_countries **desc** **limit** 1

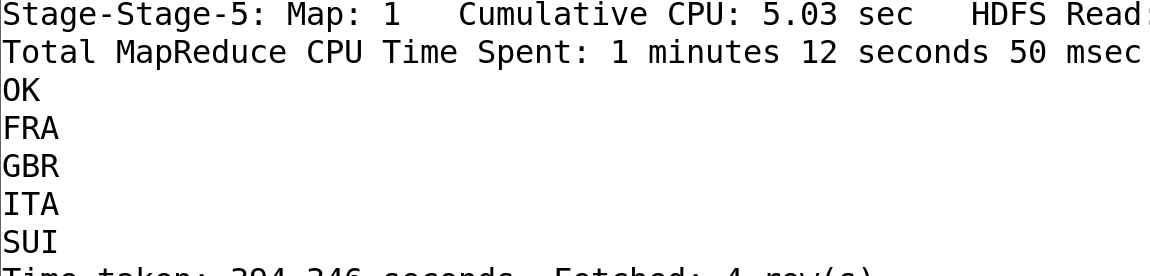


1. **Which nation has participated in all of the olympic games?**

**with** appear\_count **as** (  
**select** NOC, **count**(**distinct** games) **as** appeared **from** olympics\_par1 **group** **by** noc),seasons **as**  
(**select** **count**(**distinct** games) **as** total\_games **from** olympics\_par1)

**select** r.region **from** seasons, appear\_count a inner join olympics\_history\_noc\_regions r on a.noc = r.noc

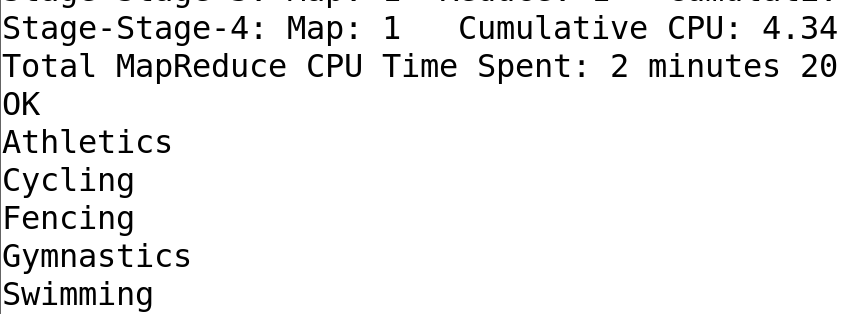
**where** appeared=total\_games



1. **Identify the sport which was played in all summer olympics.**

**with** play\_count **as** (**select** sport, **count**(**distinct** **year**) **as** played **from** olympics\_par1 **where** season="Summer" **group** **by** sport),  
total **as** (**select** **count**(**distinct** **year**) **as** c **from** olympics\_par1 **where** season **like** "Summer")

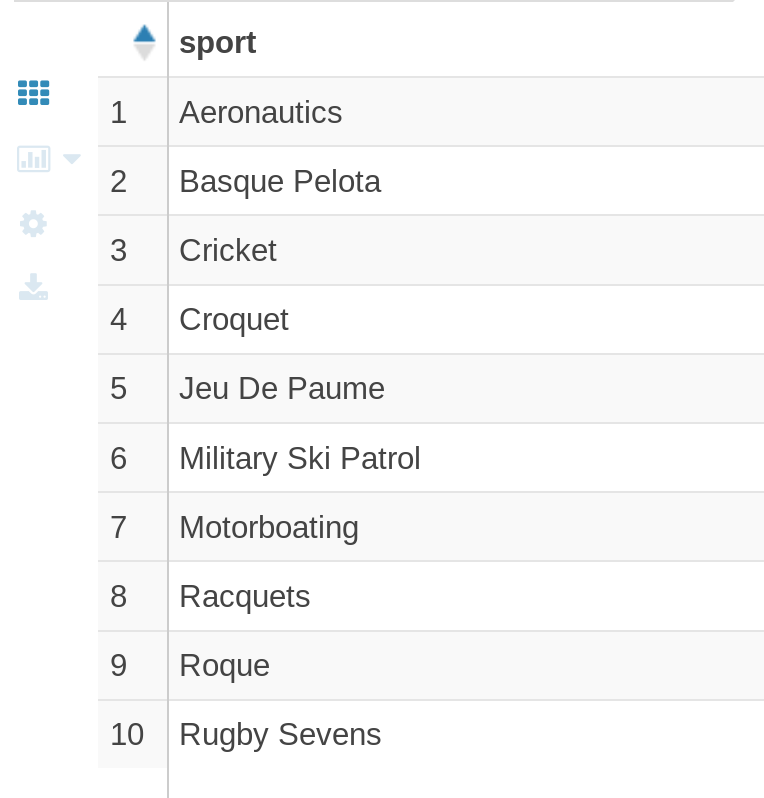
**select** sport **from** play\_count , total **where** played=c;



1. **Which Sports were just played only once in the olympics?**

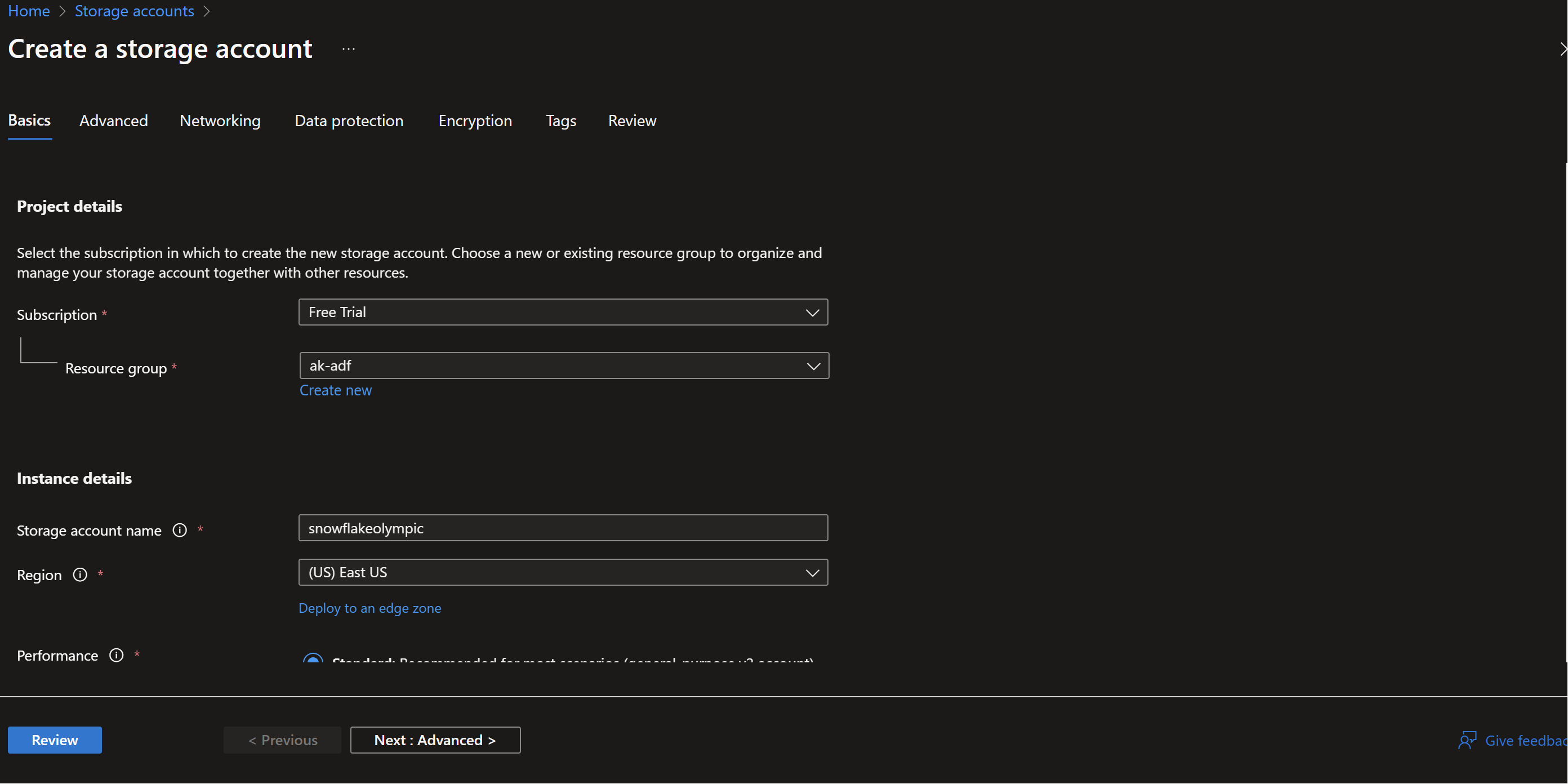
with sport\_count as (select sport, count(distinct year) as played **from** olympics\_par1 group by sport)  
select sport **from** sport\_count where **played**=1;

8 min 22 sec , 1 map , 1 reduce



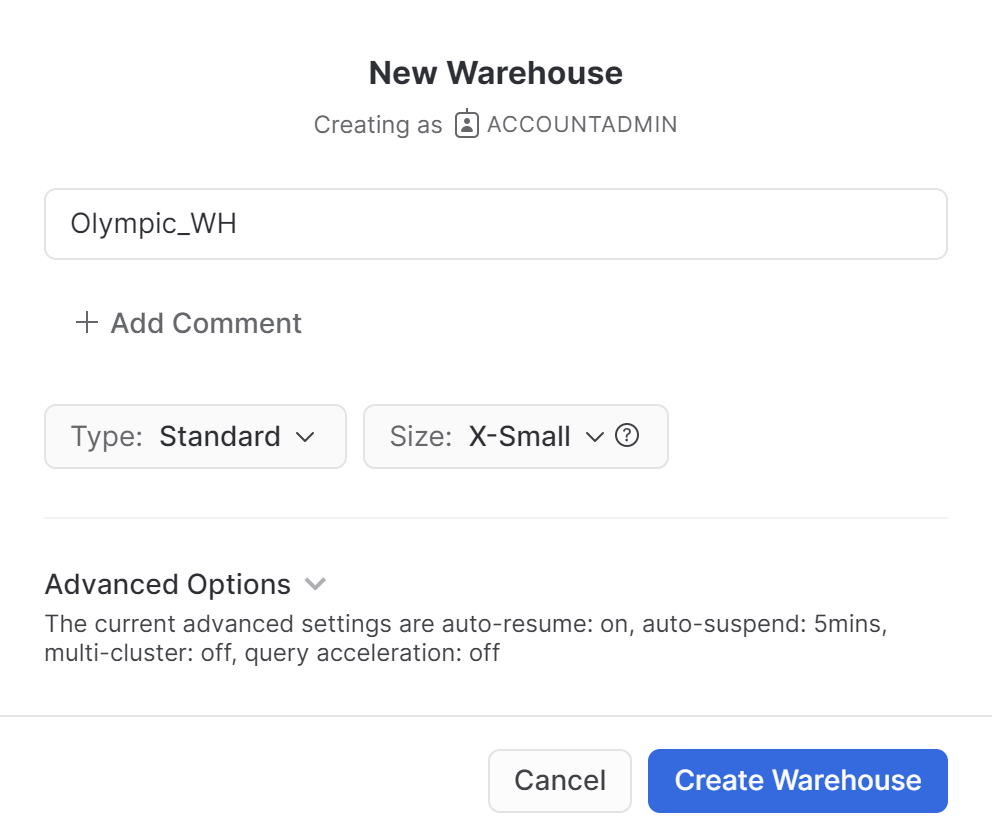
**Pipeline 2: Import the data from Azure into Snowflake using COPYINTO**

1. **Create a Storage account and container in Azure and upload .csv files**

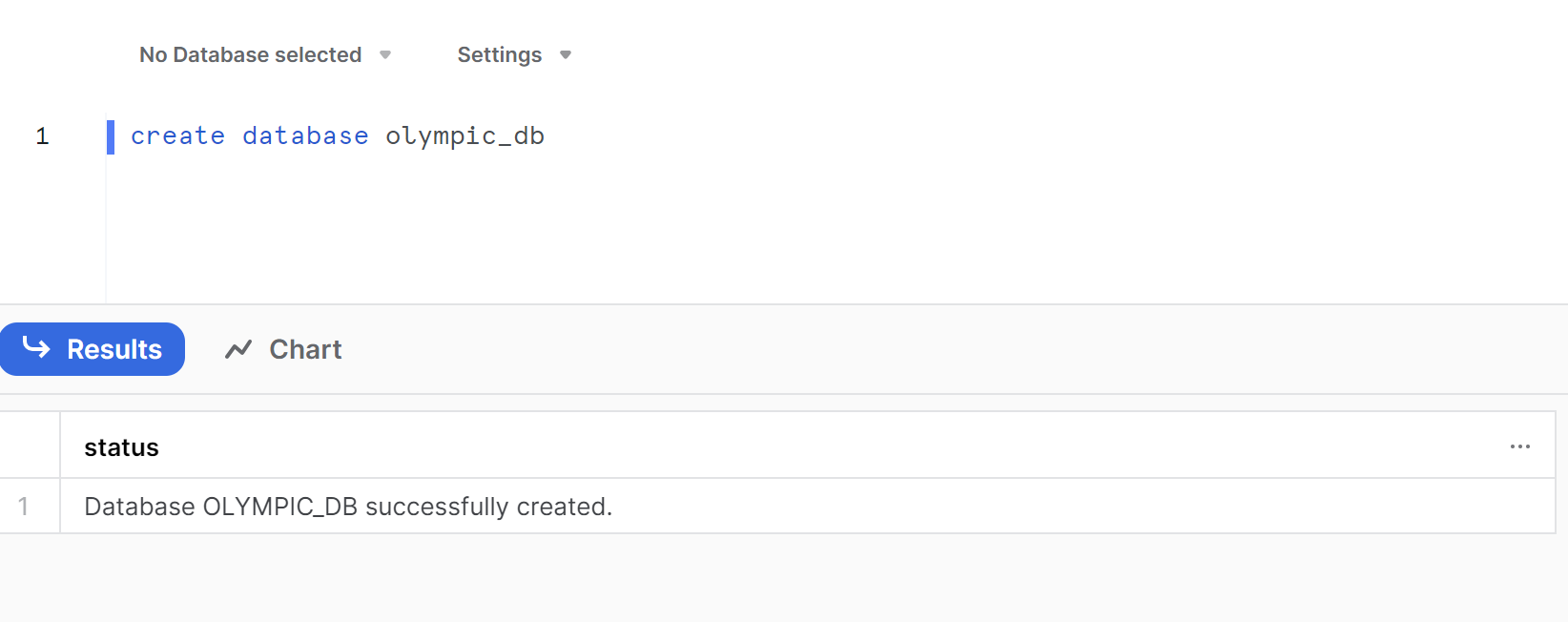


Enable hierarchical namespace

1. **Create a Data Warehouse in snowflake**

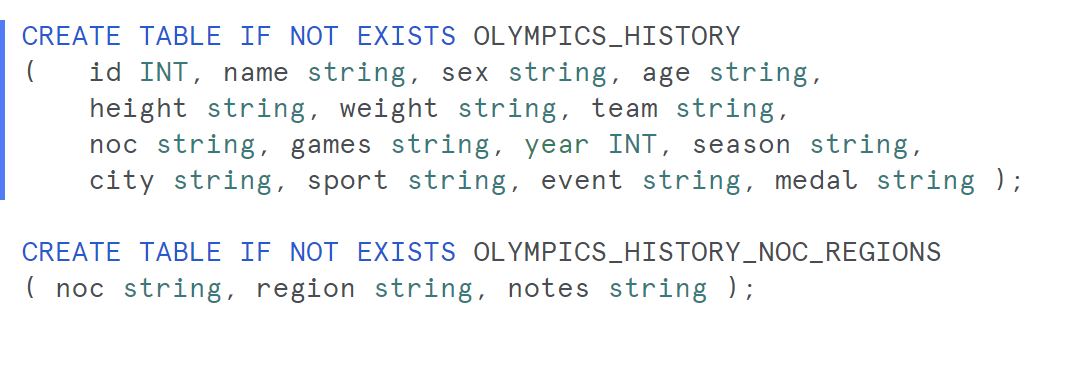


1. **Create a Database in snowflake**

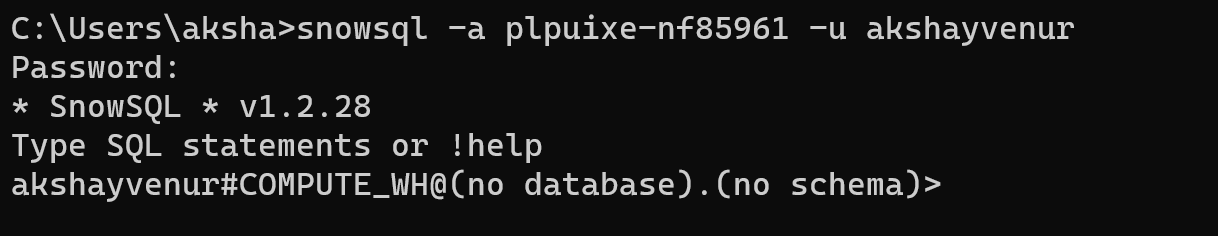
****

1. **Create a table schema**

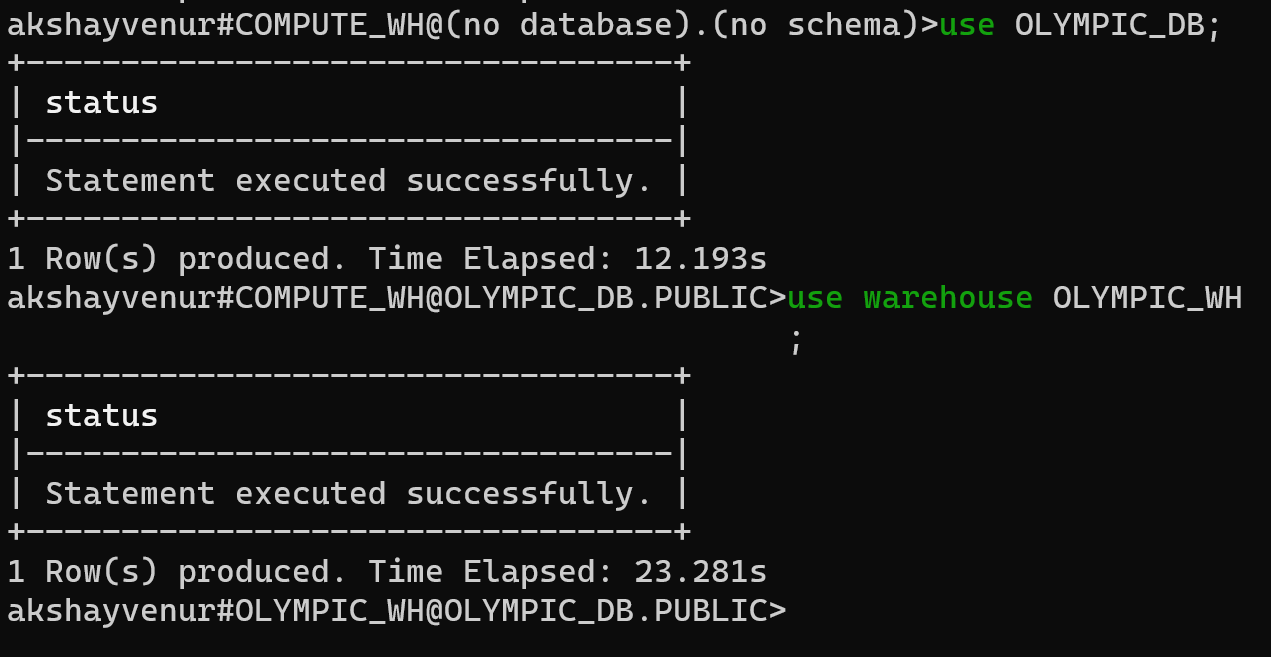
**CREATE** **TABLE** **IF** **NOT** **EXISTS** OLYMPICS\_HISTORY  
( **id** INT, **name** **string**, sex **string**, age **string**,  
 height **string**, weight **string**, team **string**,   
 noc **string**, games **string**, **year** INT, season **string**,   
 city **string**, sport **string**, **event** **string**, medal **string** );  
  
**CREATE** **TABLE** **IF** **NOT** **EXISTS** OLYMPICS\_HISTORY\_NOC\_REGIONS  
( noc **string**, region **string**, notes **string** );

****

1. **Login to SnowSQL**

****

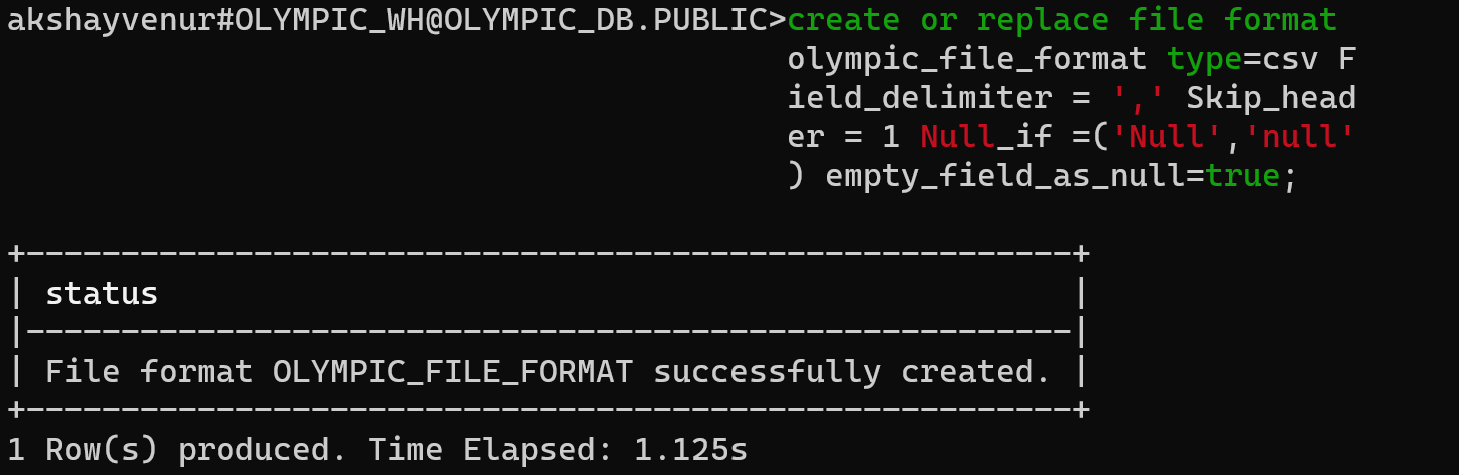
1. **Use the warehouse and database**

****

1. **Create a file format.**

By creating or replacing this file format, you are defining the rules for how Snowflake should interpret and process CSV files that use this format. You can then use this file format when loading data into Snowflake, specifying it in the corresponding data loading operations, such as the COPY INTO statement

**create** **or** **replace** **file** **format** olympic\_file\_format   
 **type**=csv   
 Field\_delimiter = ','   
 Skip\_header = 1   
 Null\_if =('Null','null')  
 empty\_field\_as\_null=true;

****

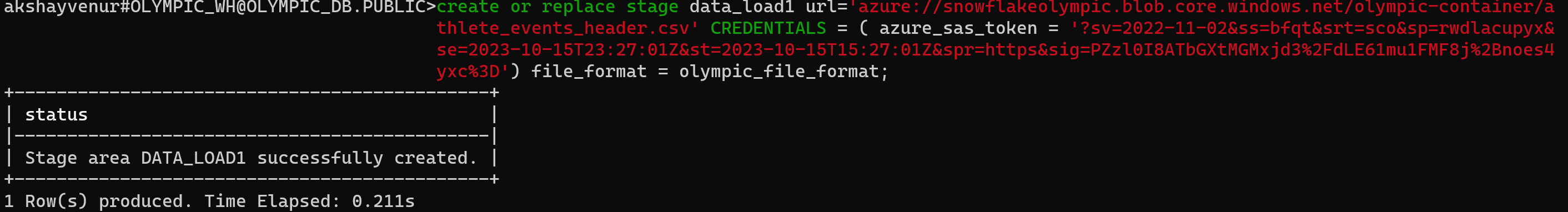
* Null\_if = ('Null', 'null'): This option tells Snowflake to treat values that are equal to 'Null' or 'null' (case-insensitive) as NULL values. In other words, if a field in the CSV file contains one of these values, it will be interpreted as a NULL value in Snowflake.
* empty\_field\_as\_null=true: This setting specifies that if a field in the CSV file is empty (contains no value), it should be treated as a NULL value in Snowflake.

1. **Creating the external stage**

create or replace stage data\_load1 url='azure://snowflakeolympic.blob.core.windows.net/olympic-container/athlete\_events\_header.csv'

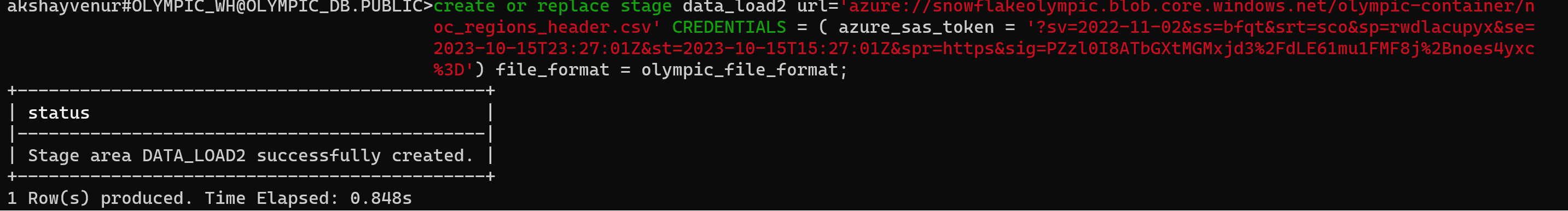
CREDENTIALS = ( azure\_sas\_token = '?sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupyx&se=2023-10-15T23:27:01Z&st=2023-10-15T15:27:01Z&spr=https&sig=PZzl0I8ATbGXtMGMxjd3%2FdLE61mu1FMF8j%2Bnoes4yxc%3D')

file\_format = olympic\_file\_format;

****

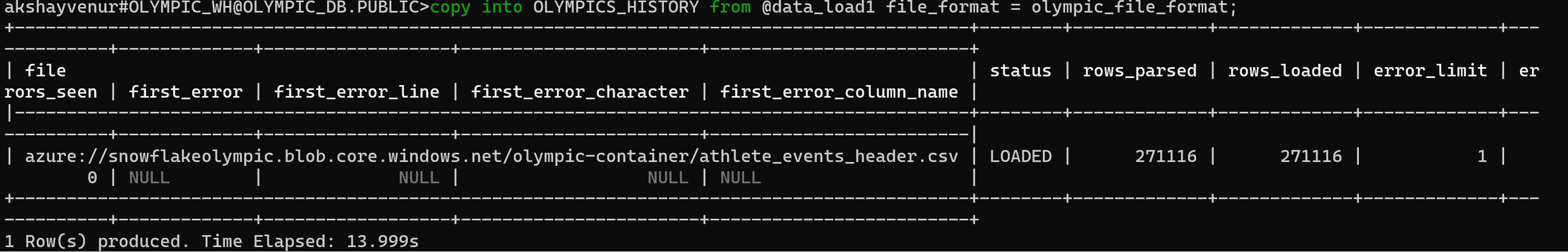
create or replace stage data\_load2 url='azure://snowflakeolympic.blob.core.windows.net/olympic-container/noc\_regions\_header.csv' CREDENTIALS = ( azure\_sas\_token = '?sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupyx&se=2023-10-15T23:27:01Z&st=2023-10-15T15:27:01Z&spr=https&sig=PZzl0I8ATbGXtMGMxjd3%2FdLE61mu1FMF8j%2Bnoes4yxc%3D')

file\_format = olympic\_file\_format;

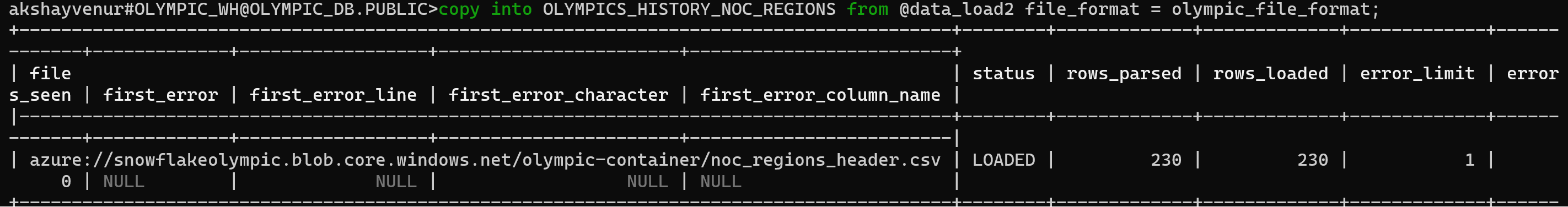


1. **Running the copy Into Command to push the data from Azure to snowflake**

**copy** **into** OLYMPICS\_HISTORY **from** @data\_load1 file\_format = olympic\_file\_format;

****

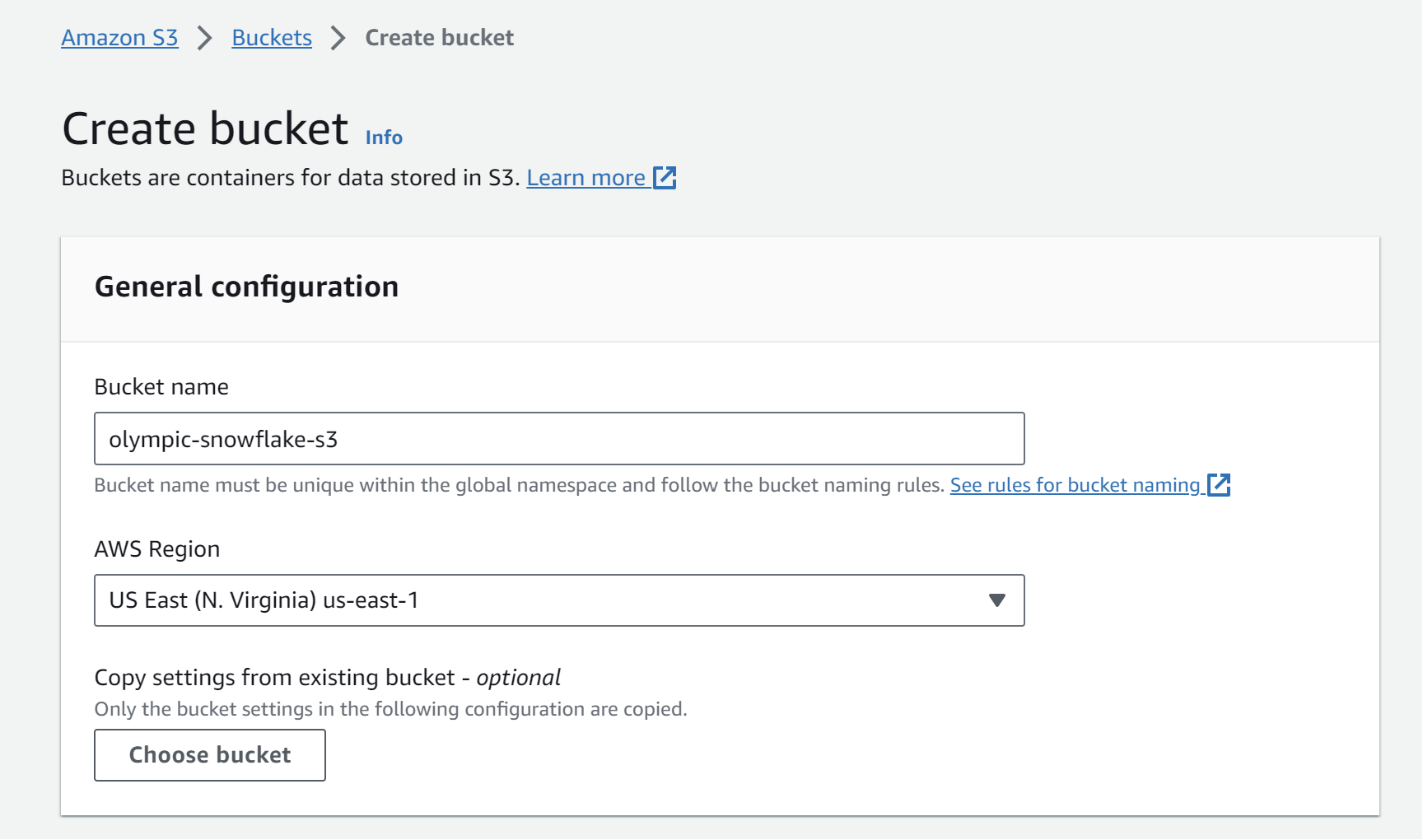
**copy** **into** OLYMPICS\_HISTORY\_NOC\_REGIONS **from** @data\_load2 file\_format = olympic\_file\_format;

****

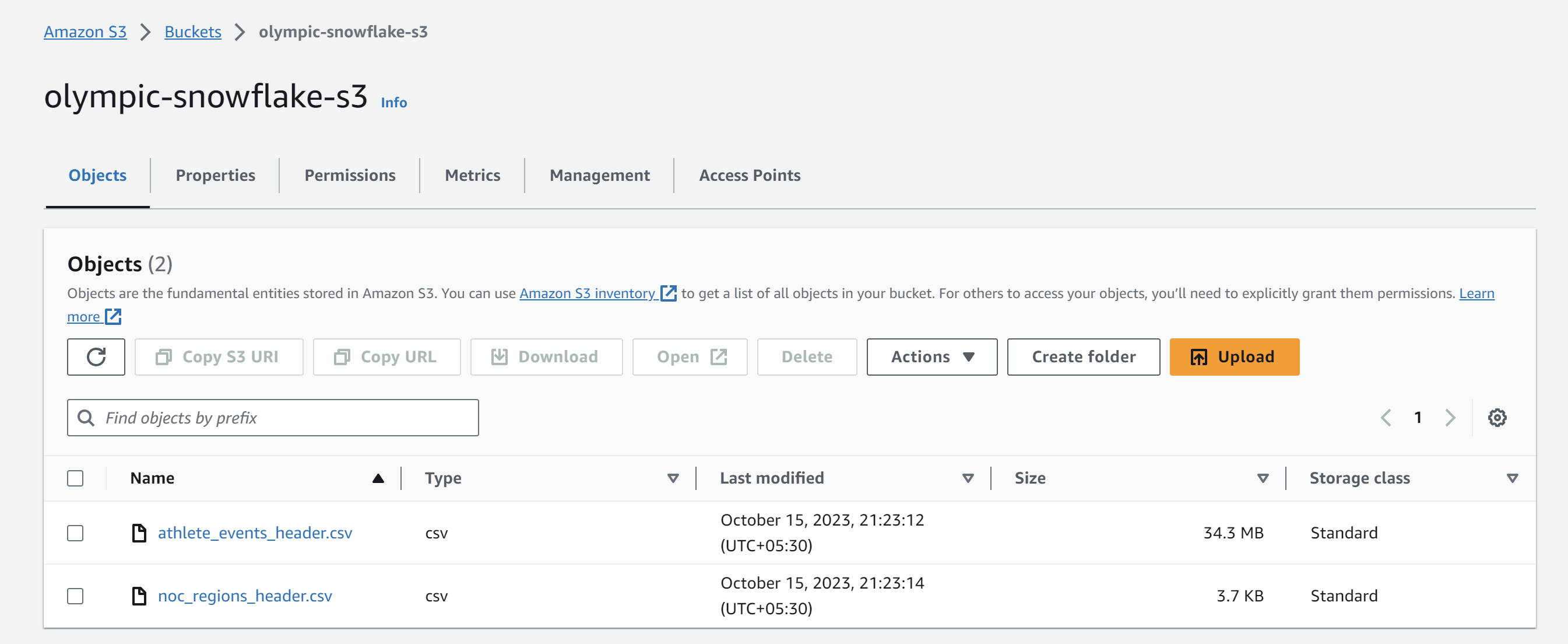
1. **Doing Analysis using Snowflake query.**

**Pipeline 3: Import the data from AWS into Snowflake using SNOWPIPE**

1. **Create a S3 bucket in AWS**

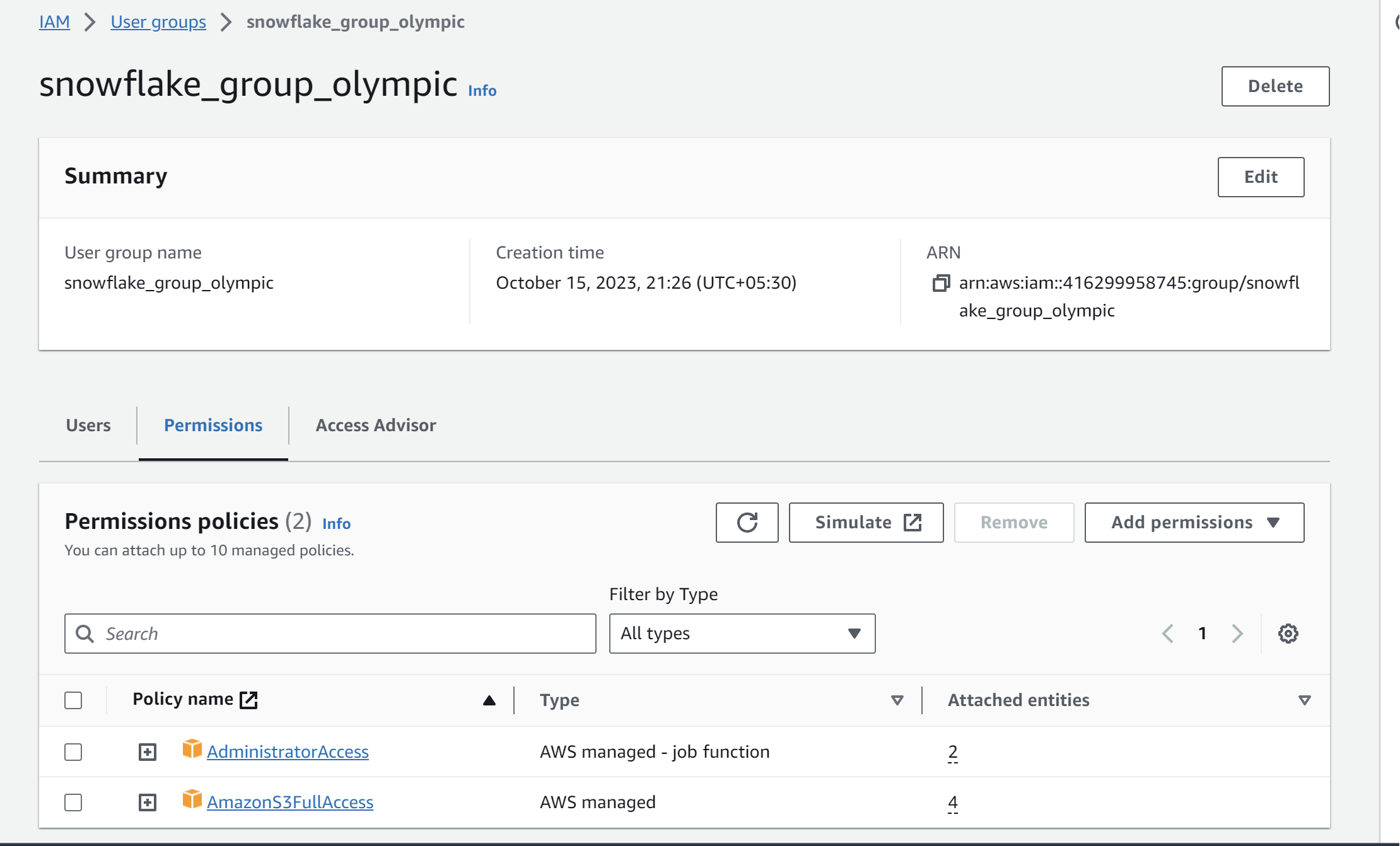
****

1. **Upload files into S3 bucket**

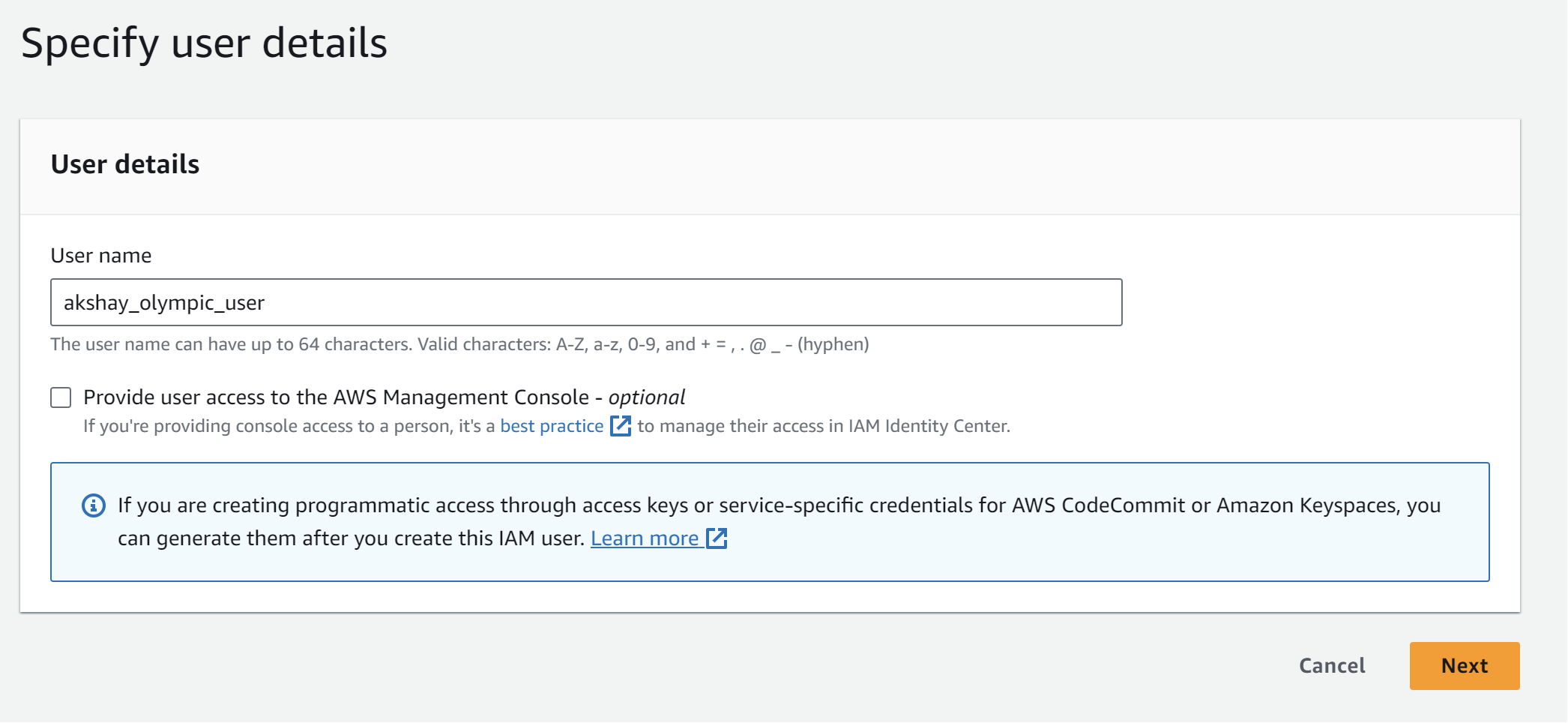
****

1. **Create a user group in IAM**

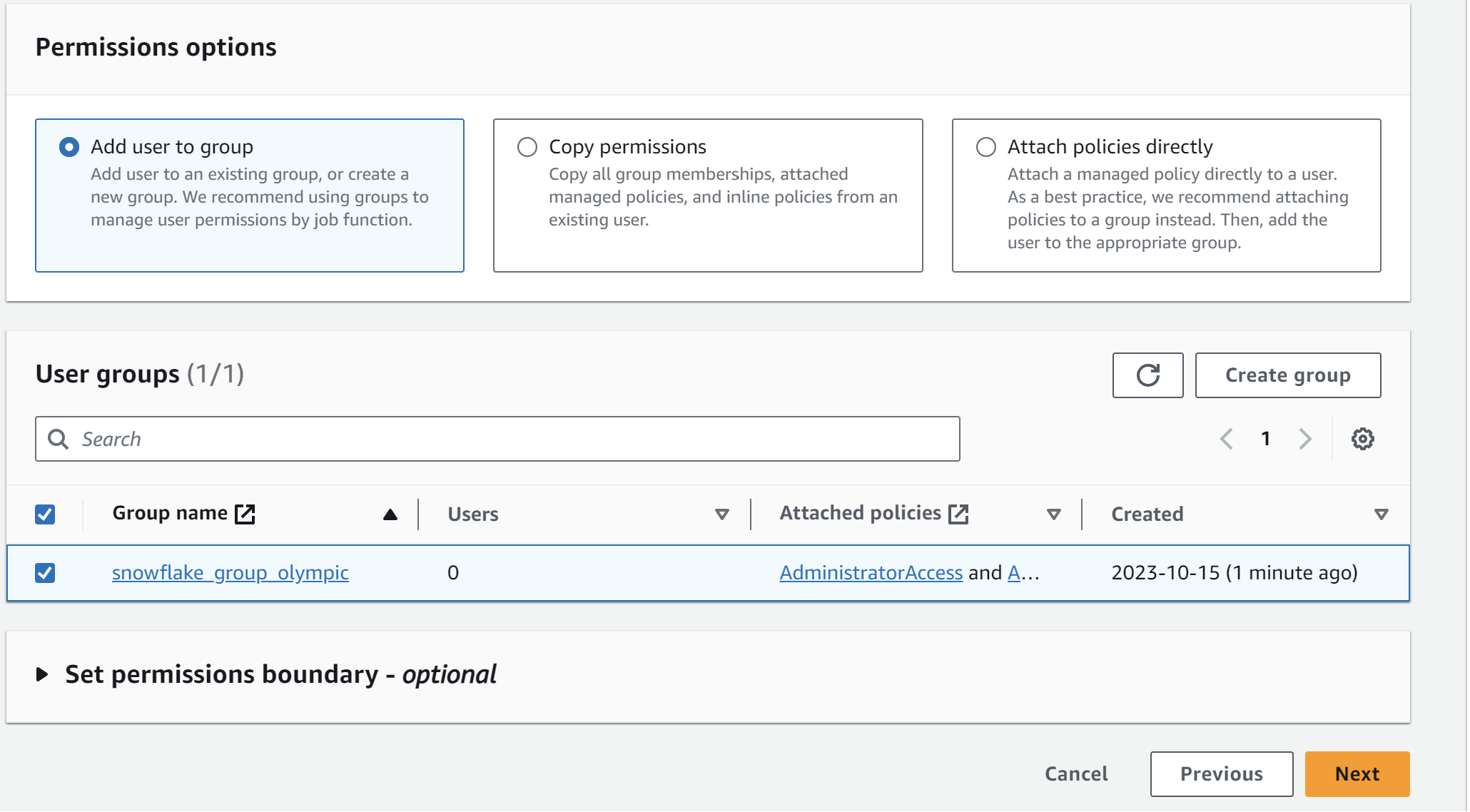
**Give S3 full access and administrative access to this user group**

****

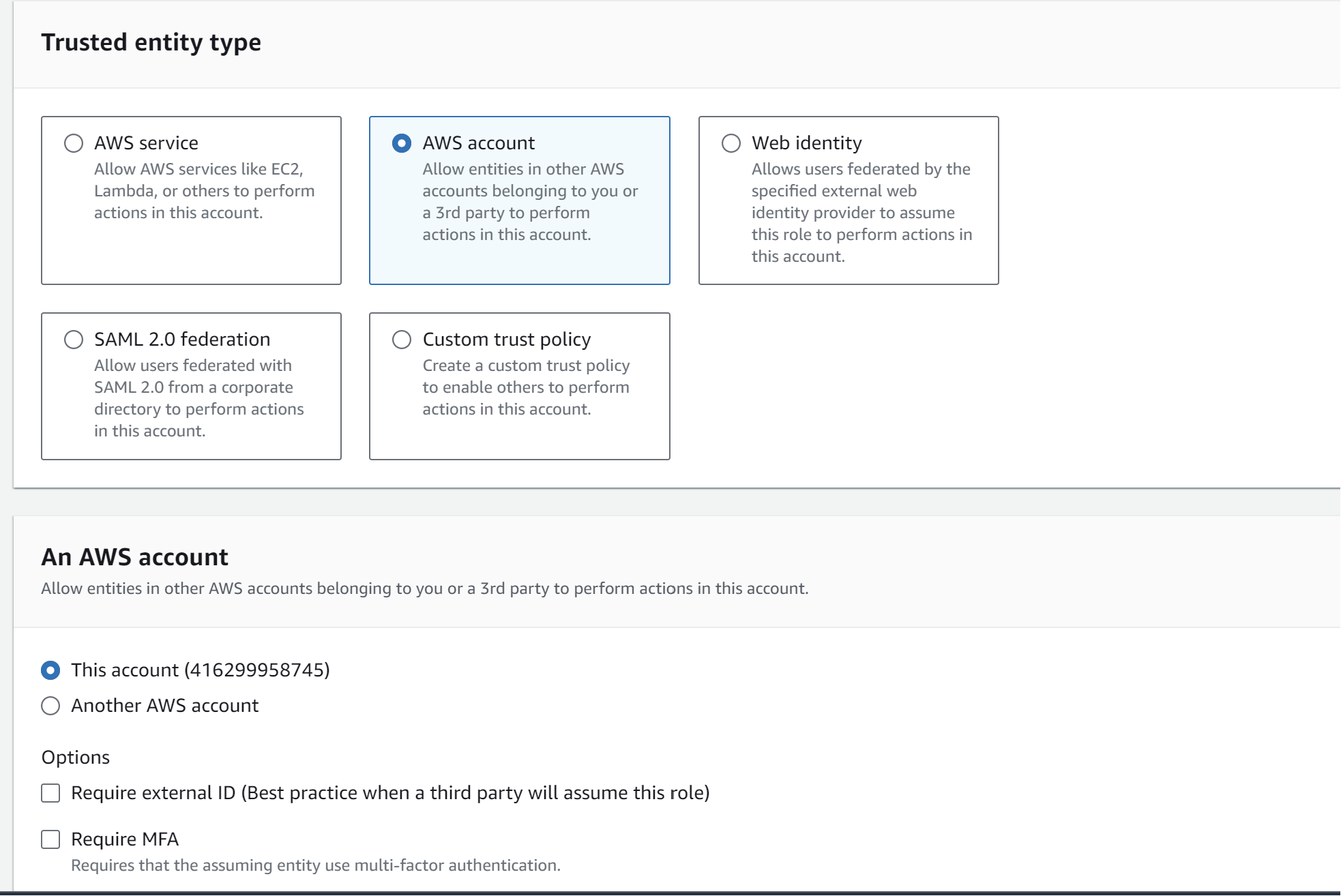
1. **Create a user in IAM**

****

**Assign the user to the user group which is created**

****

1. **Create a role in IAM**

****

**Give S3 full access and administrative access to this role**

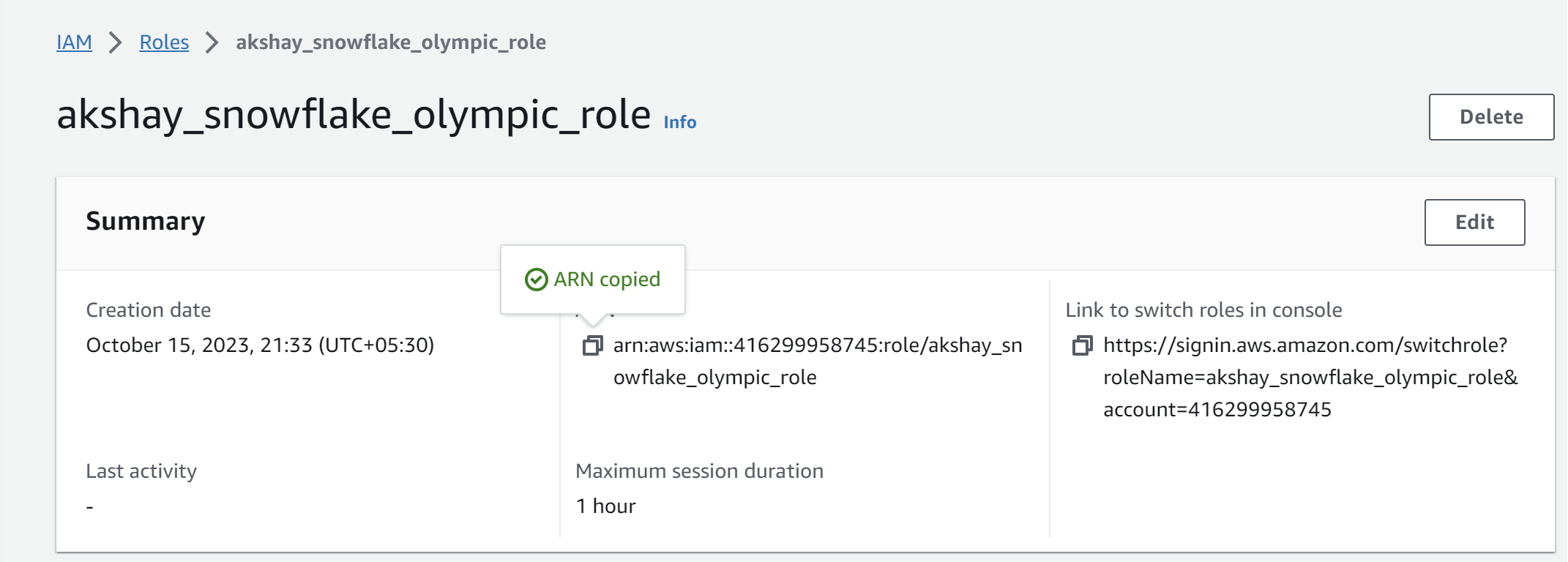
1. **Login to snowSQL**

Create database and table schema

1. **Create a file format**

**create** **or** **replace** **file** **format** olympic\_file\_format   
 **type**=csv   
 Field\_delimiter = ','   
 Skip\_header = 1   
 Null\_if =('Null','null')  
 empty\_field\_as\_null=true;

1. **Next go to the AWS role and copy the ARN**

****

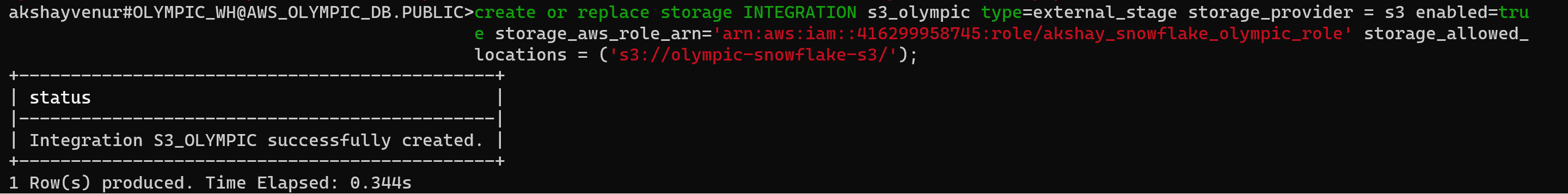
1. **Create the storage integration and copy the S3 file path and ARN**

**create** **or** **replace** **storage** INTEGRATION s3\_olympic

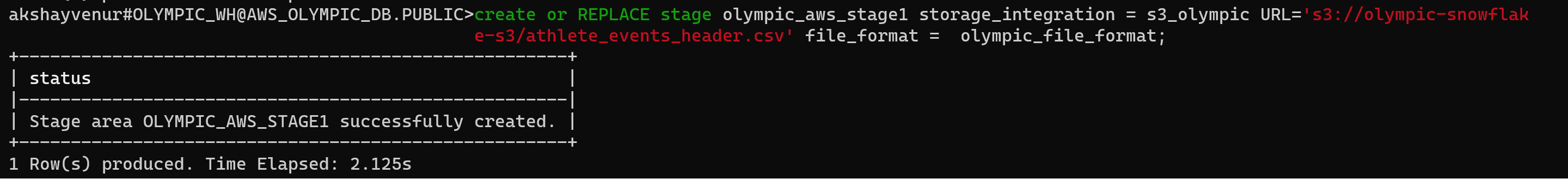
**type**=external\_stage storage\_provider = s3

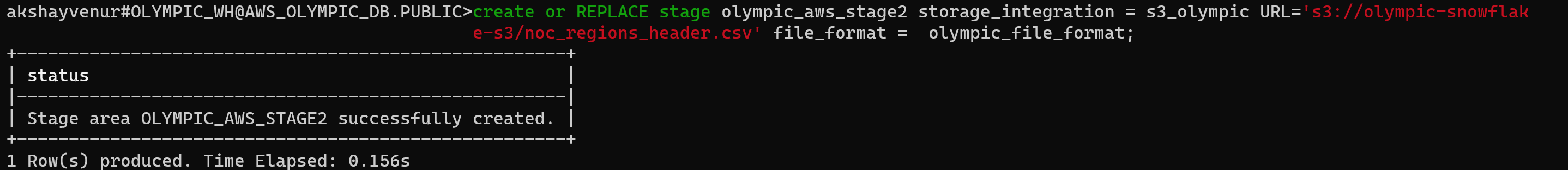
enabled=true storage\_aws\_role\_arn='arn:aws:iam::416299958745:role/akshay\_snowflake\_olympic\_role'

storage\_allowed\_locations = ('s3://olympic-snowflake-s3/');

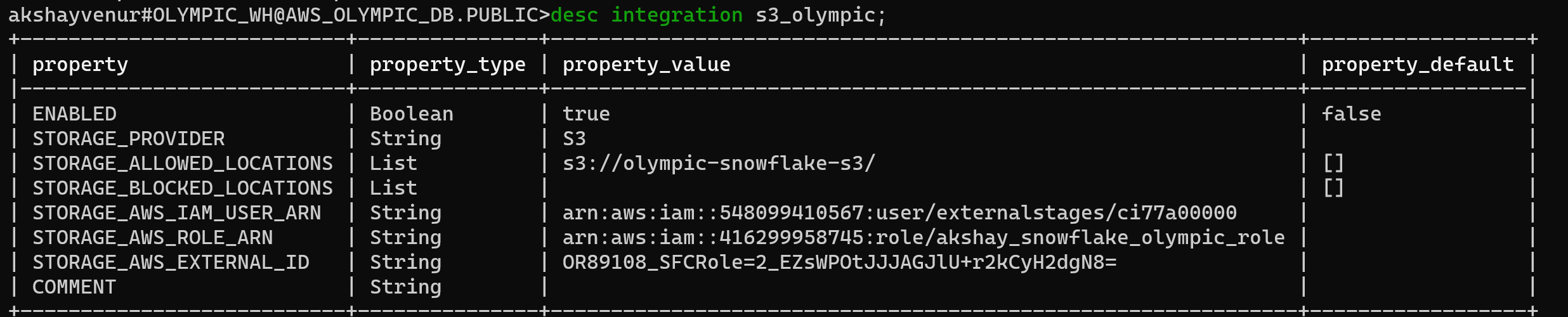
****

1. **Create a stage**

****

****

1. **Change the trust relation in AWS role**

****

**In the above " Storage\_AWS\_IAM\_User\_ARN " and " External ID " <- copy both of them and paste it in trust relationship :**

**Go to : -- > role > > Trust relationship --> Edit trust policy**

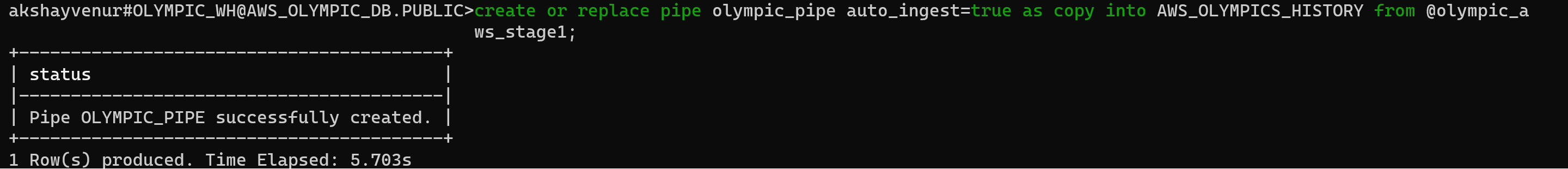
**Below are the "Storage\_AWS\_IAM\_User\_ARN " and " External ID " values which we have got from the above, change them accordingly as shown below :**

****

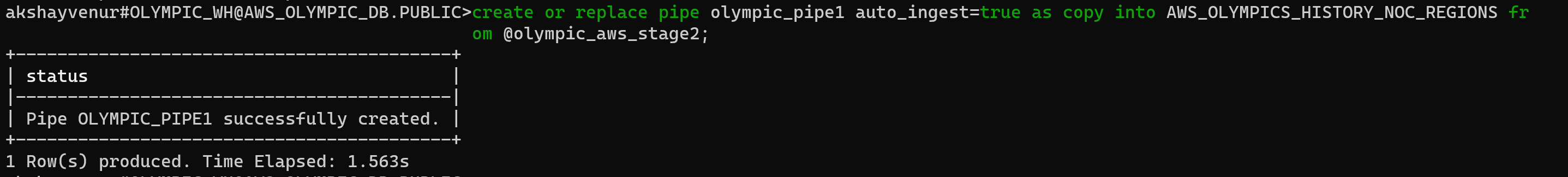
**Click on Update policy**

1. **Create the pipe**

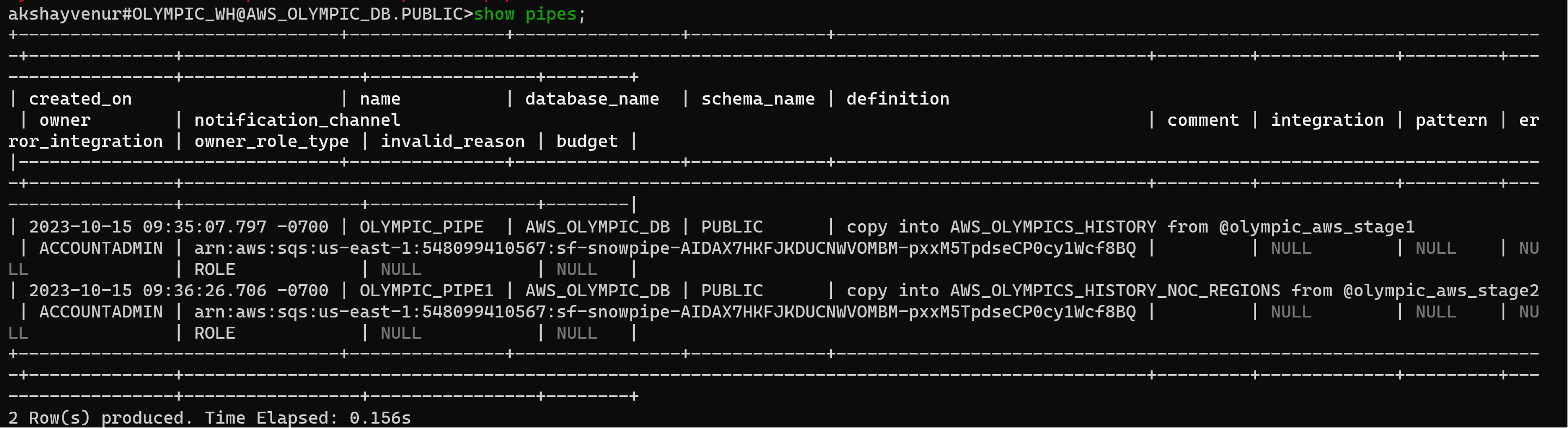
**create** **or** **replace** **pipe** olympic\_pipe auto\_ingest=true **as** copy **into** AWS\_OLYMPICS\_HISTORY **from** @olympic\_aws\_stage1;



**create** **or** **replace** **pipe** olympic\_pipe1 auto\_ingest=true **as** copy **into** AWS\_OLYMPICS\_HISTORY\_NOC\_REGIONS from @olympic\_aws\_stage2;



1. **Copy the notification channel from Show pipe code**

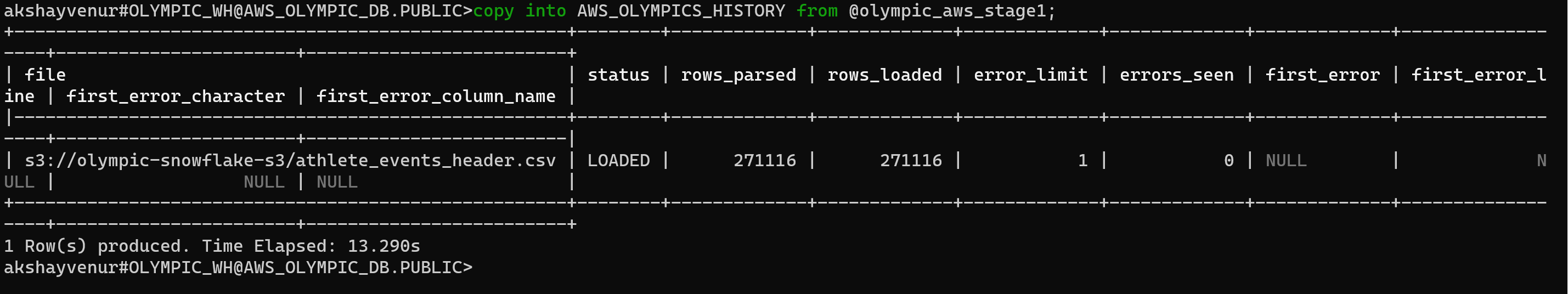


1. **Go to AWS and create event notification and SQS settings**

* Go to S3, and then click on the properties
* Click on the event notification
* Give event name
* Click on ‘all object create event’
* Next, click on SQS Queue(Destination)
* In ‘Enter SQS queue ARN’ , enter the ARN copied
* Click on save changes
* Upload the updated data in S3

1. **Load the data from S3 to Snowflake tables**

**copy** **into** AWS\_OLYMPICS\_HISTORY **from** @olympic\_aws\_stage1;



**copy** **into** AWS\_OLYMPICS\_HISTORY\_NOC\_REGIONS **from** @olympic\_aws\_stage2;

