

# Project Gladiator

## Healthcare Case Study

<b>Team Members</b>		
<b>Sr. No.</b>	<b>Name of Member</b>	<b>PS No.</b>
1	Garima Singh	10683205
2	Numan Salim Shaikh	10683782
3	Atharva Jadhav	10683762
4	Mukesh Parmar	10683809

# **INDEX**

<b>Sr. No.</b>	<b>Contents</b>	<b>Page No.</b>
1.	Project problem statement	3
2.	Importing data into Snowflake warehouse	4
3.	Dataset Segregation and Importing	5
4.	Database, schema and table creation and loading	6
5.	Findings	9
6.	External Stage and Data Loading	10
7.	Creation of Integration and Snowpipe	12
8.	Creation of Streams and Tasks	17
9.	Secure Share and New Reader account	21
10.	Clone table using Time Travel	24

## Problem Statement

Recent Covid-19 Pandemic has raised alarms over one of the most overlooked areas to focus: Healthcare Management. While healthcare management has various use cases for using data science, patient length of stay is one critical parameter to observe and predict if one wants to improve the efficiency of the healthcare management in a hospital.

This parameter helps hospitals to identify patients of high LOS risk (patients who will stay longer) at the time of admission. Once identified, patients with high LOS risk can have their treatment plan optimized to minimize LOS and lower the chance of staff/visitor infection. Also, prior knowledge of LOS can aid in logistics such as room and bed allocation planning.

Suppose you have been hired as Data Scientist of HealthMan – a not for profit organization dedicated to manage the functioning of Hospitals in a professional and optimal manner.

As a snowflake developer/ consultant, you are helping the data analytical team to explore the dataset using snowflake , history data is loaded and current data is loaded with snowpipes into database , Merge all the datasets based on the common key and create a single table and Check for noise or missing data.

The task is to accurately predict the Length of Stay for each patient on a case by case basis so that the Hospitals can use this information for optimal resource allocation and better functioning. The length of stay is divided into 11 different classes ranging from 0-10 days to more than 100 days. COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University

## Data Ingestion

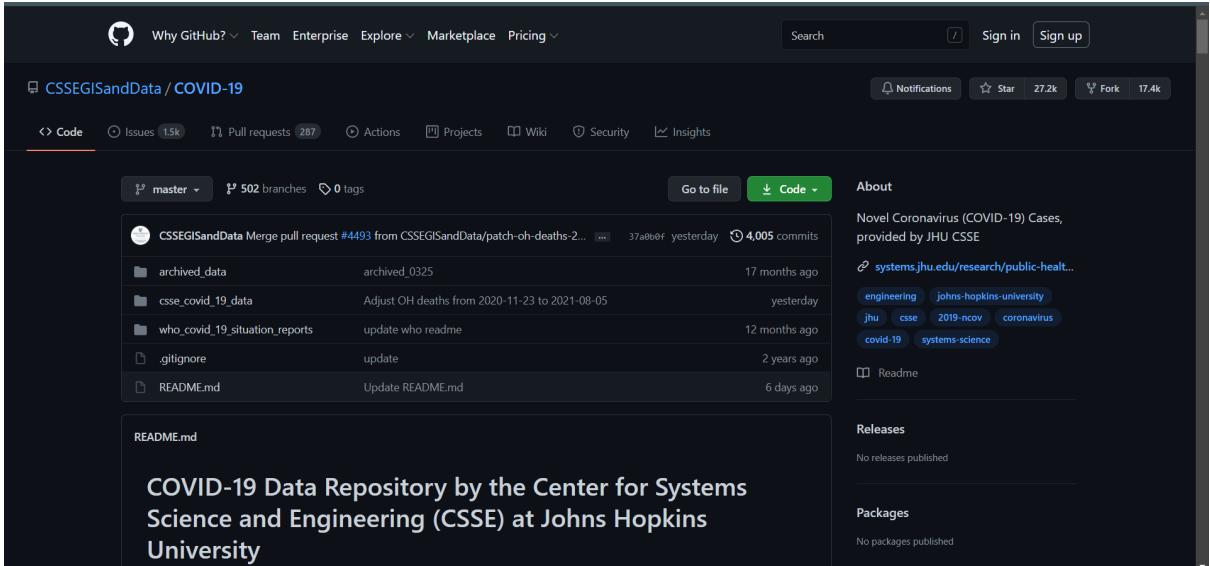
Data ingestion is a process by which data is moved from one or more sources to a destination where it can be stored and further analysed. The data might be in different formats and come from various sources, including RDBMS, other types of databases, S3 buckets, CSVs, or from streams. Since the data comes from different places, it needs to be cleansed and transformed in a way that allows you to analyse it together with data from other sources. Otherwise, your data is like a bunch of puzzle pieces that don't fit together. You can ingest data in real time, in batches, or in a combination of the two (this is called lambda architecture). When you ingest data in batches, data is imported at regularly scheduled intervals. This can be very useful when you have processes that run on a schedule, such as reports that run daily at a specific time. Real-time ingestion is useful when the information gleaned is very time-sensitive, such as data from a power grid that must be monitored moment-to-moment. Of course, you can also ingest data using a lambda architecture. This approach attempts to balance the benefits of batch and real-time modes by using batch processing to provide comprehensive views of batch data, while also using real-time processing to provide views of time-sensitive data.

We will be adding data in by internal stage and external stage. For the internal stage we will create a stage called **@covid\_int** and will feed data through our local device. We have also created the external stage **@covid\_ext** and for that we have used AWS S3 buckets. We load our data in the buckets and then we connect it to the external stage. Through that we can load data into our tables.

## -----SEGREGATION-----

Name of table	Purpose	Date wise segregation	Rows & columns observed
COVID_HISTORY	To store historical covid statistics	1st July 2020-31st Dec 2020	7,27,364 rows and 14 columns
COVID_CURRENT	To store current covid statistics	1st Jan 2021- 8th Aug 2021	8,77,102 rows and 14 columns
COVID_CONSUMPTION	To store the cumulative data	1st July 2020 - 8th Aug 2021	16,04,466 rows and 15 Columns
COVID_CONSUMPTION_HISTORY	To store the historical data	1st July 2020 - 8th Aug 2021	16,04,466 rows and 15 Columns

## DATASET -



The screenshot shows the GitHub repository page for CSSEGISandData / COVID-19. The repository has 4.005 commits, 502 branches, and 0 tags. The README.md file is visible, containing the text: "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University". The repository has 27.2k stars, 17.4k forks, and 1.5k issues. It also has 287 pull requests. The repository is provided by JHU CSSE and is associated with the URL [systems.jhu.edu/research/public-health/covid-19](https://systems.jhu.edu/research/public-health/covid-19). Tags include engineering, johns-hopkins-university, jhu, cse, 2019-ncov, coronavirus, covid-19, and systems-science.

Our dataset consists of a repository from the CSSEGIS and Data, that is the Covid-19 dataset which is segregated into three parts, historical or archived data, the current esse data and the situation reports from WHO. The historical data consists of the covid-19 data from July 2020 till December 2020 and the current covid-19 data consists of the data from January 2021 till starting of August 2021. We also have the reports generated by the WHO from the start of the pandemic.

Here is how the data for January 2020 looks locally.

Name	Date modified	Type	Size
07-01-2020.csv	09-08-2021 21:51	CSV File	502 KB
07-02-2020.csv	09-08-2021 21:51	CSV File	503 KB
07-03-2020.csv	09-08-2021 21:51	CSV File	503 KB
07-04-2020.csv	09-08-2021 21:51	CSV File	503 KB
07-05-2020.csv	09-08-2021 21:51	CSV File	504 KB
07-06-2020.csv	09-08-2021 21:51	CSV File	504 KB
07-07-2020.csv	09-08-2021 21:51	CSV File	505 KB
07-08-2020.csv	09-08-2021 21:51	CSV File	505 KB
07-09-2020.csv	09-08-2021 21:51	CSV File	506 KB
07-10-2020.csv	09-08-2021 21:51	CSV File	508 KB
07-11-2020.csv	09-08-2021 21:51	CSV File	508 KB
07-12-2020.csv	09-08-2021 21:51	CSV File	508 KB
07-13-2020.csv	09-08-2021 21:51	CSV File	508 KB
07-14-2020.csv	09-08-2021 21:51	CSV File	508 KB
07-15-2020.csv	09-08-2021 21:51	CSV File	509 KB
07-16-2020.csv	09-08-2021 21:51	CSV File	509 KB
07-17-2020.csv	09-08-2021 21:51	CSV File	512 KB
07-18-2020.csv	09-08-2021 21:51	CSV File	522 KB
07-19-2020.csv	09-08-2021 21:51	CSV File	523 KB
07-20-2020.csv	09-08-2021 21:51	CSV File	524 KB
07-21-2020.csv	09-08-2021 21:51	CSV File	524 KB
07-22-2020.csv	09-08-2021 21:51	CSV File	524 KB
07-23-2020.csv	09-08-2021 21:51	CSV File	525 KB
07-24-2020.csv	09-08-2021 21:51	CSV File	525 KB



```

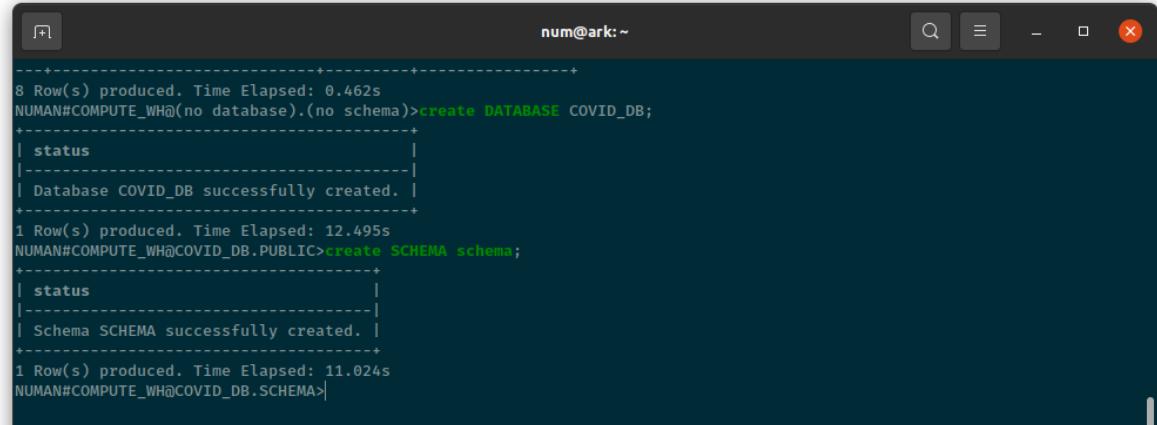
1 import os
2 import shutil as sh
3 count=0
4
5 #Path of directory where all csv files exists
6 for file in os.listdir("/home/num/Numan/LTI/newData/COVID-19/csse_covid_19_data/csse_covid_19_daily_reports"):
7     if file.endswith(".csv"):
8         month=file[0:2]
9         date=file[3:5]
10        year=file[6:10]
11        print(month+year)
12
13        if not os.path.exists("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports/"+year):
14            os.mkdir("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports//"+year)
15
16        if not os.path.exists("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports/"+year+"/"+month):
17            os.mkdir("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports//"+year+"/"+month)
18
19        sh.copy("/home/num/Numan/LTI/newData/COVID-19/csse_covid_19_data/csse_covid_19_daily_reports/"+file,"/home/num/Numan/LTI/newData/ncov19-"
20        csse_covid_19_daily_reports//"+year+"/"+month+"/"+file)
21        count+=1
22
22 print(count)

```

## DATABASE AND SCHEMA CREATION

```
CREATE DATABASE covid_db;
```

```
CREATE SCHEMA "COVID_DB"."SCHEMA";
```



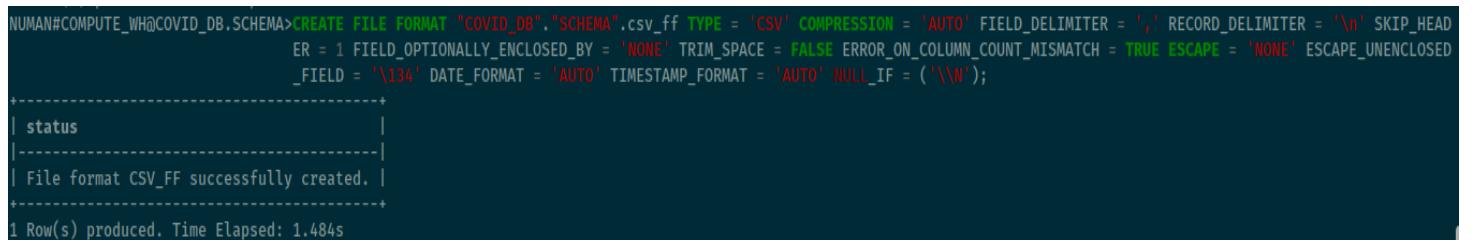
The terminal window shows the following command and its output:

```
num@ark:~  
+-----+  
8 Row(s) produced. Time Elapsed: 0.462s  
NUMAN#COMPUTE_WH@(no database).(no schema)>create DATABASE COVID_DB;  
+-----+  
| status |  
+-----+  
| Database COVID_DB successfully created. |  
+-----+  
1 Row(s) produced. Time Elapsed: 12.495s  
NUMAN#COMPUTE_WH@COVID_DB.PUBLIC>create SCHEMA schema;  
+-----+  
| status |  
+-----+  
| Schema SCHEMA successfully created. |  
+-----+  
1 Row(s) produced. Time Elapsed: 11.024s  
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

### Creation of file format -

Created a file format named “CSV\_FF” in our COVID\_DB database. It describes a set of staged data to access or load into Snowflake tables. For our project, we have focused solely on .csv file formats.

```
CREATE FILE FORMAT "COVID_DB"."SCHEMA".CSV_FF SET COMPRESSION =  
'AUTO' FIELD_DELIMITER = ',' RECORD_DELIMITER = '\n' SKIP_HEADER = 1  
FIELD_OPTIONALLY_ENCLOSED_BY = 'NONE' TRIM_SPACE = FALSE  
ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE ESCAPE = 'NONE'  
ESCAPE_UNENCLOSED_FIELD = '\134' DATE_FORMAT = 'AUTO'  
TIMESTAMP_FORMAT = 'AUTO' NULL_IF = ('\\N');
```



The terminal window shows the following command and its output:

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE FILE FORMAT "COVID_DB"."SCHEMA".CSV_FF TYPE = 'CSV' COMPRESSION = 'AUTO' FIELD_DELIMITER = ',' RECORD_DELIMITER = '\n' SKIP_HEADER = 1 FIELD_OPTIONALLY_ENCLOSED_BY = 'NONE' TRIM_SPACE = FALSE ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE ESCAPE = 'NONE' ESCAPE_UNENCLOSED_FIELD = '\134' DATE_FORMAT = 'AUTO' TIMESTAMP_FORMAT = 'AUTO' NULL_IF = ('\\N');  
+-----+  
| status |  
+-----+  
| File format CSV_FF successfully created. |  
+-----+  
1 Row(s) produced. Time Elapsed: 1.484s
```

### Creation of internal stage COVID\_INT

```
CREATE STAGE "COVID.DB"."SCHEMA".covid_int;
```

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE STAGE "COVID_DB"."SCHEMA".covid_int;
+-----+
| status
|-
| Stage area COVID_INT successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 1.226s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

## TABLE CREATION AND LOADING

```

CREATE TABLE "COVID_DB"."SCHEMA"."COVID_HISTORY" ("province/state" STRING, "country/region" STRING, "LAST_UPDATE" TIMESTAMP, "CONFIRMED" INTEGER, "DEATHS" INTEGER, "RECOVERED" INTEGER, "SUSPECTED" INTEGER, "CONFNSUSP" INTEGER, "NOTES" STRING);

```

```

1 Row(s) produced. Time Elapsed: 0.296s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE TABLE "COVID_DB"."SCHEMA"."COVID_HISTORY" ("FIPS" NUMBER (7), "ADMIN2" VARCHAR (50), "PROVINCE_STATE" VARCHAR (50), "COUNTRY_REGION" VARCHAR (50), "LAST_UPDATE" DATE, "LAT" NUMBER (4,4), "LONG" NUMBER (4,4), "CONFIRMED" INTEGER, "DEATHS" INTEGER, "RECOVERED" INTEGER, "ACTIVE" INTEGER, "COMBINED_KEY" VARCHAR (100), "INCIDENT_RATE" FLOAT, "CASE_FATALITY_RATIO" FLOAT);
+-----+
| status
|-
| Table COVID_HISTORY successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 23.477s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>desc table COVID_HISTORY;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| name | type | kind | null? | default | primary key | unique key | check | expression | comment | policy name |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| FIPS | NUMBER(7,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| ADMIN2 | VARCHAR(50) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| PROVINCE_STATE | VARCHAR(50) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| COUNTRY_REGION | VARCHAR(50) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| LAST_UPDATE | DATE | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| LAT | NUMBER(4,4) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| LONG | NUMBER(4,4) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| CONFIRMED | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| DEATHS | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| RECOVERED | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| ACTIVE | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| COMBINED_KEY | VARCHAR(100) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| INCIDENT_RATE | FLOAT | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
| CASE_FATALITY_RATIO | FLOAT | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
14 Row(s) produced. Time Elapsed: 4.559s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

Putting the .csv files of the covid statistics for the month of **November,2020** into the internal stage that we created for smooth loading.

```

put file://D:\Project_Gladiator\COVID-19-master\hist_data_monthly\2020\11\*.csv
@covid_int/2020/11;

```

```

31 Row(s) produced. Time Elapsed: 19.089s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>put file:///home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports/2020/11/*.csv @covid_int/2020/11;
11-03-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.215s, 0.16MB/s).
11-09-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.228s, 0.16MB/s).
11-28-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.251s, 0.16MB/s).
11-22-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.294s, 0.15MB/s).
11-02-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (0.880s, 0.22MB/s).
11-17-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (0.864s, 0.23MB/s).
11-25-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.115s, 0.18MB/s).
11-06-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.401s, 0.14MB/s).
11-18-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.816s, 0.11MB/s).
11-10-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.792s, 0.11MB/s).
11-29-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.303s, 0.15MB/s).
11-12-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.378s, 0.14MB/s).
11-20-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (0.709s, 0.28MB/s).
11-15-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.180s, 0.16MB/s).
11-01-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (0.929s, 0.21MB/s).
11-23-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.026s, 0.19MB/s).
11-21-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (0.846s, 0.23MB/s).
11-19-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.163s, 0.17MB/s).
11-07-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (0.947s, 0.20MB/s).
11-16-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (1.157s, 0.17MB/s).
11-26-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (0.776s, 0.25MB/s).
11-30-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (0.827s, 0.24MB/s).
11-05-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (0.919s, 0.21MB/s).
11-08-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.910s, 0.10MB/s).
11-24-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (0.792s, 0.25MB/s).
11-14-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.299s, 0.15MB/s).
11-11-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.384s, 0.14MB/s).
11-04-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (1.166s, 0.17MB/s).
11-27-2020.csv_c.gz(0.20MB): [#####] 100.00% Done (0.934s, 0.21MB/s).
11-13-2020.csv_c.gz(0.19MB): [#####] 100.00% Done (0.817s, 0.24MB/s).

+-----+-----+-----+-----+-----+-----+-----+-----+
| source | target | source_size | target_size | source_compression | target_compression | status | message |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 11-01-2020.csv | 11-01-2020.csv.gz | 558013 | 201897 | NONE | GZIP | uploaded | |
| 11-02-2020.csv | 11-02-2020.csv.gz | 558216 | 202177 | NONE | GZIP | uploaded | |
| 11-03-2020.csv | 11-03-2020.csv.gz | 558290 | 202319 | NONE | GZIP | uploaded | |
| 11-04-2020.csv | 11-04-2020.csv.gz | 558685 | 202554 | NONE | GZIP | uploaded | |
| 11-05-2020.csv | 11-05-2020.csv.gz | 558701 | 202568 | NONE | GZIP | uploaded | |

```

## Loading into table COVID\_HISTORY -

```

COPY INTO COVID_HISTORY FROM (SELECT t.$1, t.$2, t.$3, t.$4, t.$5, t.$6, t.$7, t.$8,
t.$9, t.$10, t.$11, to_char(t.$12), t.$13, t.$14 FROM @covid_int (file_format => 'CSV_FF' )
t) on_error = 'ABORT_STATEMENT' PURGE = FALSE;

```

We copy the staged .csv files from the internal stage @covid\_int to the table covid\_history.

file_error_column_name	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_e
covid_int/2020/07/07-22-2020.csv.gz	LOADED	3935	3935	1	0	NULL		NULL	NULL
covid_int/2020/08/08-11-2020.csv.gz	LOADED	3952	3952	1	0	NULL		NULL	NULL
covid_int/2020/08/08-24-2020.csv.gz	LOADED	3959	3959	1	0	NULL		NULL	NULL
covid_int/2020/09/09-20-2020.csv.gz	LOADED	3963	3963	1	0	NULL		NULL	NULL
covid_int/2020/09/09-27-2020.csv.gz	LOADED	3965	3965	1	0	NULL		NULL	NULL
covid_int/2020/10/10-20-2020.csv.gz	LOADED	3968	3968	1	0	NULL		NULL	NULL
covid_int/2020/11/11-04-2020.csv.gz	LOADED	3970	3970	1	0	NULL		NULL	NULL
covid_int/2020/11/11-26-2020.csv.gz	LOADED	3984	3984	1	0	NULL		NULL	NULL
covid_int/2020/12/12-09-2020.csv.gz	LOADED	3984	3984	1	0	NULL		NULL	NULL
covid_int/2020/12/12-21-2020.csv.gz	LOADED	3984	3984	1	0	NULL		NULL	NULL

**Total Rows Inserted - 7,27,364**

## FINDINGS

We found out that on copying the .csv files from our internal stage, our query was separating the combined key values in the dataset which led to duplication of the columns in our dataset. Even the data was mislabeled due to this error.

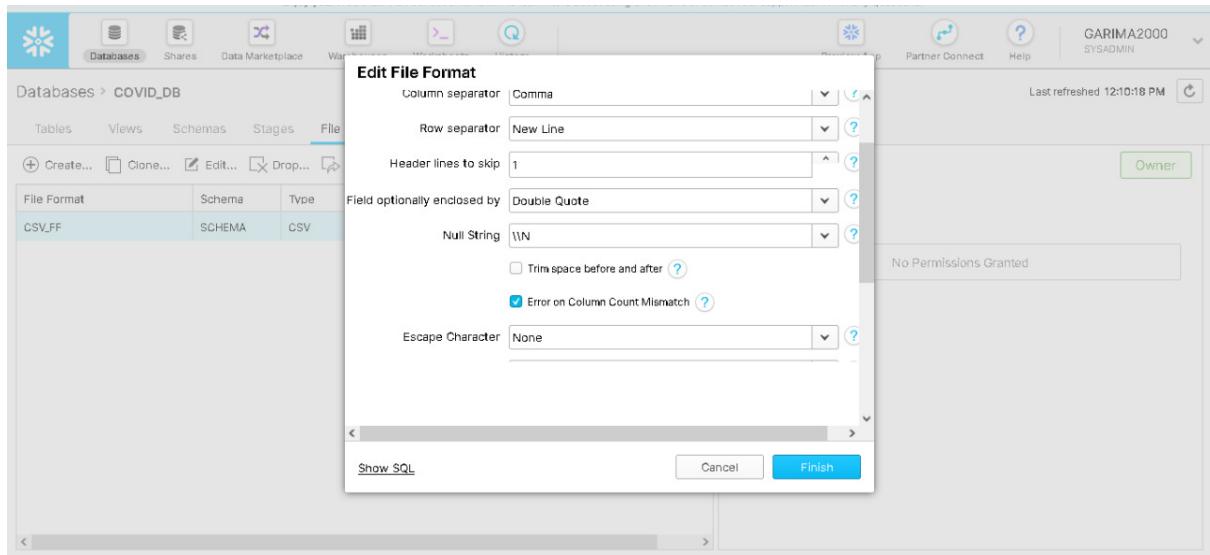
Row	\$1	\$2	\$3	\$4	\$5	\$6	\$7	\$8	\$9	\$10	\$11	\$12	\$13	\$14
1	FIPS	Admin2	Province_Sta...	Country_Reg...	lastUpdate	Lat	Long...	Confirmed	Deaths	Recovered	Active	Combined_K...	Incidence_R...	Case-Fatality...
2	45001	Abbeville	South Carolina	US	2020-07-02 ...	34.22333378	-82.46170658	113	0	0	113	"Abbeville	South Carolina	US"
3	22001	Acadia	Louisiana	US	2020-07-02 ...	30.2950649	-92.41419698	919	37	0	882	"Acadia	Louisiana	US"
4	51001	Accomack	Virginia	US	2020-07-02 ...	37.76707161	-75.63234615	1043	14	0	1029	"Accomack	Virginia	US"
5	16001	Ada	Idaho	US	2020-07-02 ...	43.4526575	-116.241551...	2288	23	0	2265	"Ada	Idaho	US"
6	19001	Adair	Iowa	US	2020-07-02 ...	41.33075609	-94.47105874	15	0	0	15	"Adair	Iowa	US"
7	21001	Adair	Kentucky	US	2020-07-02 ...	37.10459774	-85.28129668	121	19	0	102	"Adair	Kentucky	US"
8	29001	Adair	Missouri	US	2020-07-02 ...	40.19058551	-92.60079167	90	0	0	90	"Adair	Missouri	US"
9	40001	Adair	Oklahoma	US	2020-07-02 ...	35.88494195	-94.65895267	115	4	0	111	"Adair	Oklahoma	US"
10	8001	Adams	Colorado	US	2020-07-02 ...	39.87432092	-104.3382578	4226	156	0	4070	"Adams	Colorado	US"
11	16003	Adams	Idaho	US	2020-07-02 ...	44.89333571	-116.4545247	11	0	0	11	"Adams	Idaho	US"
12	17001	Adams	Illinois	US	2020-07-02 ...	39.98815591	-91.18766813	93	1	0	92	"Adams	Illinois	US"
13	18001	Adams	Indiana	US	2020-07-02 ...	40.7457653	-84.93671406	45	1	0	44	"Adams	Indiana	US"
14	19003	Adams	Iowa	US	2020-07-02 ...	41.02903567	-94.69832645	8	0	0	8	"Adams	Iowa	US"

To tackle this issue, we had to make a correction in our file format “CSV\_FF” and apply an optional delimiter for double quotes.

```

ALTER FILE FORMAT "COVID_DB"."SCHEMA".CSV_FF SET COMPRESSION =
'AUTO' FIELD_DELIMITER = ',' RECORD_DELIMITER = '\n' SKIP_HEADER = 1
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');

```



## EXTERNAL STAGE -

We use AWS S3 as our external stage. We create a bucket **snow-extstage-grp1** and create a folder "**covid/**". In this folder we add half of the "current" data in a month wise manner. All .csvs belonging to the same month were in the same folder.

Buckets (82) [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

	Name	AWS Region	Access	Creation date
<input type="checkbox"/>	shriyash20	east-1	Objects can be public	(UTC+05:30)
<input type="checkbox"/>	snow-extstage-group4	US East (N. Virginia) us-east-1	Objects can be public	August 9, 2021, 14:48:00 (UTC+05:30)
<input type="checkbox"/>	snow-extstage-group7	US East (N. Virginia) us-east-1	Objects can be public	August 9, 2021, 12:35:13 (UTC+05:30)
<input type="checkbox"/>	snow-extstage-grp1	US East (N. Virginia) us-east-1	Objects can be public	August 10, 2021, 13:07:08 (UTC+05:30)
<input type="checkbox"/>	snow-extstage-grp8	US East (N. Virginia) us-east-1	Bucket and objects not public	August 10, 2021, 17:26:26 (UTC+05:30)
<input type="checkbox"/>	soumya99	US East (N. Virginia) us-east-1	Objects can be public	July 26, 2021, 12:33:06 (UTC+05:30)
<input type="checkbox"/>	supriyamishra	US East (N. Virginia) us-east-1	Objects can be public	August 5, 2021, 11:48:10 (UTC+05:30)
<input type="checkbox"/>	tanvijadhav	US East (N. Virginia) us-east-1	Objects can be public	August 5, 2021, 11:45:10 (UTC+05:30)
<input type="checkbox"/>	testpipesnow	US East (N. Virginia) us-east-1	Objects can be public	July 29, 2021, 17:51:14 (UTC+05:30)

Amazon S3 > snow-extstage-grp1

**snow-extstage-grp1** [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

**Objects (1)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	covid/	Folder	-	-	-

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	01/	Folder	-	-	-
<input type="checkbox"/>	02/	Folder	-	-	-
<input type="checkbox"/>	03/	Folder	-	-	-

We create a table COVID\_CURRENT to store this current data

```
CREATE OR REPLACE TABLE "COVID_DB"."SCHEMA"."COVID_CURRENT"
("FIPS" NUMBER (7), "ADMIN2" VARCHAR (50), "PROVINCE_STATE" VARCHAR (50),
"COUNTRY_REGION" VARCHAR (50), "LAST_UPDATE" TIMESTAMP, "LAT" FLOAT,
"LONG" FLOAT, "CONFIRMED" INTEGER, "DEATHS" INTEGER,
"RECOVERED" INTEGER, "ACTIVE" INTEGER, "COMBINED_KEY" VARCHAR (100),
"INCIDENT_RATE" FLOAT, "CASE_FATALITY_RATIO" FLOAT);
```

### Create external stage COVID\_EXT in Snowflake using AWS access and secret key

```
CREATE STAGE "COVID_DB"."SCHEMA".covid_ext URL =
's3://snow-extstage-grp1/covid/' CREDENTIALS = (AWS_KEY_ID =
'AKIAYLYEKLCYL63VYJQP' AWS_SECRET_KEY =
'q+jI2Xkm6Jq33MSqaSc+4U1vRcYFdB0rE+*****');
```

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE STAGE "COVID_DB"."SCHEMA".covid_ext URL = 's3://snow-extstage-grp1/covid/' CREDENTIALS = (AWS_KEY_ID = 'AKIAYLYEKLCYL63VYJQP' AWS_SECRET_KEY = 'q+jI2Xkm6Jq33MSqaSc+4U1vRcYFdB0rE+*****');
|-----+
| status
|-----+
| Stage area COVID_EXT successfully created.
|-----+
1 Row(s) produced. Time Elapsed: 1.349s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>list @covid_ext;
+-----+-----+-----+-----+
| name | size | md5 | last_modified |
+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/2021/01/01-01-2021.csv | 566983 | 14e83f73cbaaf0555e332666c5c27fe5 | Tue, 10 Aug 2021 07:38:02 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-02-2021.csv | 567295 | d49a19a7020c1ec11c50395d47ed4cc8a | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-03-2021.csv | 567359 | 5ca3dfb457290d67d428806e28b0acc0 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-04-2021.csv | 567399 | 256e2a093148f3988f39b5f9066d58ac8 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-05-2021.csv | 567607 | a78228e0a1eb1f2beba75f4a467fcda | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-06-2021.csv | 567624 | e221e6423461221d2db7d2818203e657 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-07-2021.csv | 567672 | 8c9a59b00cd2f9b72384aa9c3a4b18d | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-08-2021.csv | 567831 | 2fee15f5f3fa602752dab26e93fa20a | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-09-2021.csv | 567795 | 12dcf920ac65528cb27624b83bb9ea5 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-10-2021.csv | 567838 | 7d8150d6e02f76e3b6586ae2efb089d3a | Tue, 10 Aug 2021 07:38:02 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-11-2021.csv | 567989 | 37b8321fc807e4496b6ffe26a3129d6b | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-12-2021.csv | 568015 | 8af3c485b37e1fa3710c3d722e1497ce | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-13-2021.csv | 568118 | 044724c88002346693e2789563194d14 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-14-2021.csv | 567648 | 037429ae68cc3ad2e98c88c64493f7e4 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-15-2021.csv | 568402 | 29031144d5eba2d2e98c88c64493f7e4 | Tue, 10 Aug 2021 07:38:03 GMT |
|-----+
```

### Copying data into COVID\_CURRENT from external stage COVID\_EXT -

```
COPY INTO "COVID_DB"."SCHEMA"."COVID_CURRENT" FROM
'@"COVID_DB"."SCHEMA"."COVID_EXT"' FILE_FORMAT =
'"COVID_DB"."SCHEMA".CSV_FF"' ON_ERROR = 'CONTINUE' PURGE = FALSE;
```

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>COPY INTO "COVID_DB"."SCHEMA"."COVID_CURRENT" FROM '@"COVID_DB"."SCHEMA"."COVID_EXT"' FILE_FORMAT = '"COVID_DB"."SCHEMA".CSV_FF"' ON_ERROR = 'CONTINUE' PURGE = FALSE;
+-----+-----+-----+-----+-----+-----+-----+-----+
| file | status | rows_parsed | rows_loaded | error_limit | errors_seen | first_error |
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/01/01-04-2021.csv | LOADED | 3985 | 3985 | 3985 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/01/01-16-2021.csv | LOADED | 3986 | 3986 | 3986 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/01/01-25-2021.csv | LOADED | 3987 | 3987 | 3987 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/02/02-08-2021.csv | LOADED | 3987 | 3987 | 3987 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/03/03-09-2021.csv | LOADED | 3987 | 3987 | 3987 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/01/01-12-2021.csv | LOADED | 3985 | 3985 | 3985 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/01/01-24-2021.csv | LOADED | 3987 | 3987 | 3987 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/02/02-05-2021.csv | LOADED | 3987 | 3987 | 3987 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
| s3://snow-extstage-grp1/covid/02/02-27-2021.csv | LOADED | 3987 | 3987 | 3987 | 0 | NULL |
|-----+-----+-----+-----+-----+-----+-----+-----+
```

## Creating storage integration s3\_integration -

We change our user role to **ACCOUNTADMIN** and create an integration. A storage integration is a Snowflake object that stores a generated identity and access management (IAM) entity for your external cloud storage, along with an optional set of allowed or blocked storage locations (Amazon S3, Google Cloud Storage, or Microsoft Azure). Cloud provider administrators in your organization grant permissions on the storage locations to the generated entity. This option allows users to avoid supplying credentials when creating stages or when loading or unloading data. A single storage integration can support multiple external stages.

```
CREATE STORAGE INTEGRATION s3_integration
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = S3
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::574997813424:role/lti_numan_935'
  storage_allowed_locations = ('s3://snow-extstage-grp1/covid/')
```

```
|-----|
| 358792 |
+-----+
1 Row(s) produced. Time Elapsed: 1.271s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>use role accountadmin;
+-----+
| status |
|-----|
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 1.229s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>create storage integration s3_pipe
  type = external_stage
  storage_provider = s3
  enabled = true
  storage_aws_role_arn = 'arn:aws:iam::574997813424:role/lti_numan_935'
  storage_allowed_locations = ('s3://snow-extstage-grp1/covid/');
+-----+
| status |
|-----|
| Integration S3_PIPE successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 1.383s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

## Creating pipe s3\_pipe -

```
CREATE PIPE "COVID_DB"."SCHEMA".S3_PIPE AS COPY INTO
"COVID_DB"."SCHEMA"."COVID_CURRENT" FROM
@"COVID_DB"."SCHEMA"."COVID_EXT" FILE_FORMAT = ( FORMAT_NAME =
"COVID_DB"."SCHEMA"."CSV_FF");
```

## Edit the trust relationship -

Steps :

1. Get the **ARN** from Snowflake and add it to the policy document. Access points have **Amazon Resource Names (ARNs)**. Access point ARNs are similar to bucket ARNs, but they are explicitly typed and encode the access point's Region and the AWS account ID of the access point's owner.
2. Also, get the **notification channel ID** and add to the trust relationship policy.

---

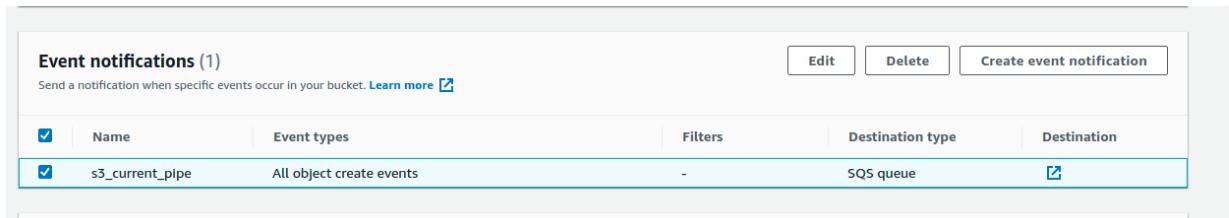
## Edit Trust Relationship

You can customize trust relationships by editing the following access control policy document.

### Policy Document

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Sid": "",
6        "Effect": "Allow",
7        "Principal": {
8          "AWS": "arn:aws:iam::099555862548:user/ovzt-s-v2sa7501"
9        },
10       "Action": "sts:AssumeRole",
11       "Condition": {
12         "StringEquals": {
13           "sts:ExternalId": "NLA77504_SFCRole=3_H08J6U/rQ1li9xAR0ycyhNQ2Z9Q="
14         }
15       }
16     }
17   ]
18 }
```

## Add SQS Event notification -

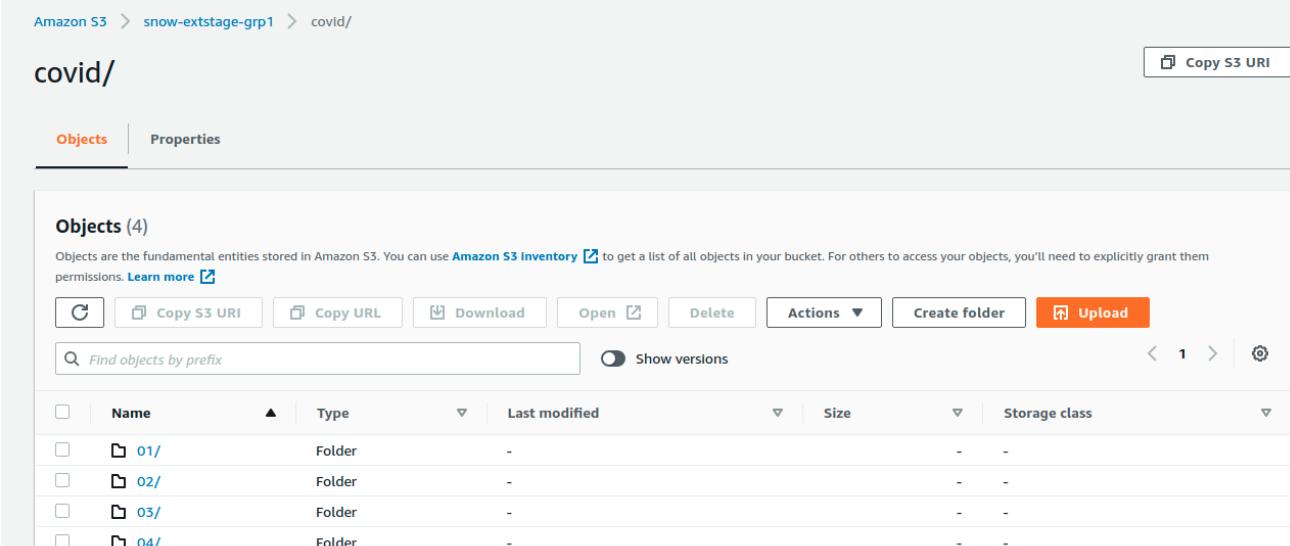


Event notifications (1)		Edit	Delete	Create event notification
Send a notification when specific events occur in your bucket. <a href="#">Learn more</a>				
Name	Event types	Filters	Destination type	Destination
<input checked="" type="checkbox"/> s3_current_pipe	All object create events	-	SQS queue	<a href="#">Edit</a>

## Checking pipe status -

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select system$pipe_Status('s3_pipe');
+-----+
| SYSTEM$PIPE_STATUS('S3_PIPE') |
+-----+
| {"executionState":"RUNNING","pendingFileCount":0,"notificationChannelName":"arn:aws:sqs:us-east-1:099555862548:sf-snowpipe-AIDAROLP55AKIPNEZKLYY-sjq7f0wnn4GRodf5chha_g","numOutstandingMessagesOnChannel":3,"lastReceivedMessageTimestamp":"2021-08-10T10:05:42.98Z","lastForwardedMessageTimestamp":"2021-08-10T10:05:43.033Z"} |
+-----+
1 Row(s) produced. Time Elapsed: 1.227s
```

Add data into the bucket and see if it reflects in the table -



The screenshot shows the Amazon S3 console interface. The path is 'Amazon S3 > snow-extstage-grp1 > covid/'. The 'covid/' folder is selected. The 'Objects' tab is active, showing 4 objects. The objects are 01/, 02/, 03/, and 04/, all of which are folders. The table below shows their details: Name (01/, 02/, 03/, 04/), Type (Folder), Last modified (-), Size (-), and Storage class (-). There are buttons for Actions, Create folder, and Upload.

Name	Type	Last modified	Size	Storage class
01/	Folder	-	-	-
02/	Folder	-	-	-
03/	Folder	-	-	-
04/	Folder	-	-	-

Before adding the data into the AWS S3 bucket -

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select count(*) from covid_current;
+-----+
| COUNT(*) |
+-----+
| 478402 |
+-----+
1 Row(s) produced. Time Elapsed: 3.782s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

On adding the fifth month (folder 05) data (May 2021) we can see the change in count of rows, that's how we know the pipe is functioning properly.

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select count(*) from covid_current;
+-----+
| COUNT(*) |
+-----+
| 478402 |
+-----+
1 Row(s) produced. Time Elapsed: 3.782s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select count(*) from covid_current;
+-----+
| COUNT(*) |
+-----+
| 601999 |
+-----+
1 Row(s) produced. Time Elapsed: 1.498s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

### Setting clustering key for both the tables -

In general, Snowflake produces well-clustered data in tables; however, over time, particularly as DML occurs on very large tables (as defined by the amount of data in the table, not the number of rows), the data in some table rows might no longer cluster optimally on desired dimensions.

To improve the clustering of the underlying table micro-partitions, you can always manually sort rows on key table columns and re-insert them into the table.

For our tables we use the columns **country\_region** and **province\_state**.

Below is the implementation of the query to alter the tables with manually created cluster keys.

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>alter table covid_history cluster by (COUNTRY_REGION,province_state);
+-----+
| status |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 4.082s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

Changes in both the tables can be seen below -

## Changing retention period -

By default, the maximum retention period is 1 day (i.e. one 24 hour period). With Snowflake Enterprise Edition (and higher), the default for your account can be set to any value up to 90 days. The account default can be overridden using the **DATA\_RETENTION\_TIME\_IN\_DAYS** parameter in the command.

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>alter table covid_history set data_retention_time_in_days=30;
+-----+
| status          |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 1.349s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>alter table covid_current set data_retention_time_in_days=30;
+-----+
| status          |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.395s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

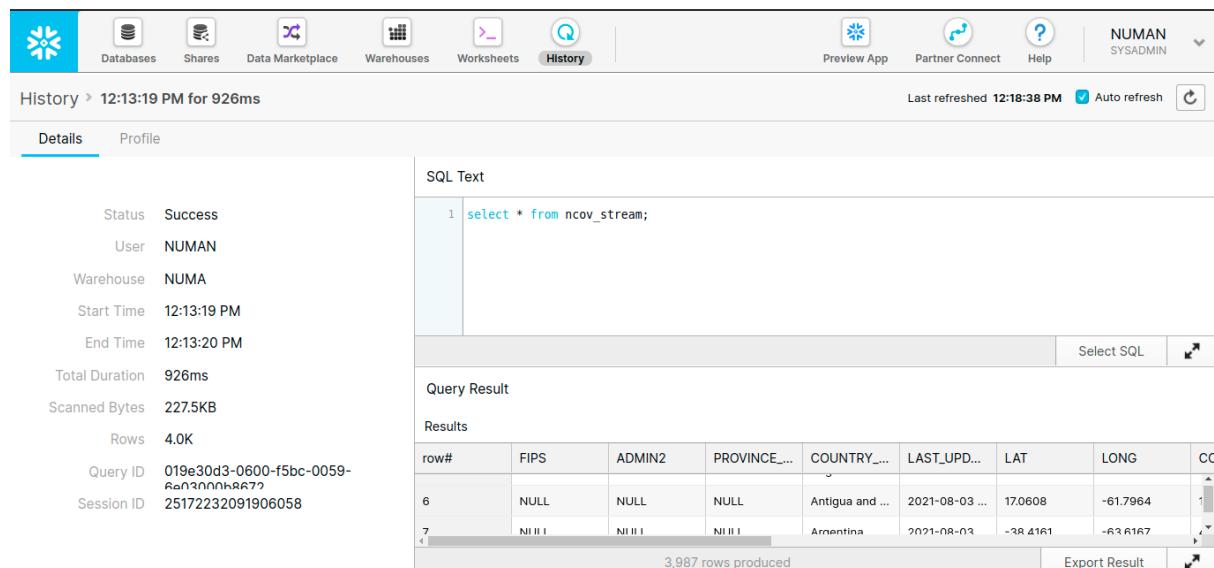
## Creation of Streams -

A stream object records data manipulation language (DML) changes made to tables, including inserts, updates, and deletes, as well as metadata about each change, so that actions can be taken using the changed data. This process is referred to as **change data capture (CDC)**. An individual table stream tracks the changes made to rows in a *source table*. A **table stream** (also referred to as simply a “**stream**”) makes a “change table” available of what changed, at the row level, between two transactional points of time in a table. This allows querying and consuming a sequence of change records in a transactional fashion.

We created a stream named “**ncov\_stream**” on the table “**covid\_merger**” so that all the changes that were made in the external stage (AWS S3 bucket) could be captured and the same changes can be incorporated in the “**covid\_current**” table.

```
CREATE TABLE covid_merger LIKE covid_current;
CREATE OR REPLACE STREAM ncov_stream ON TABLE covid_merger;
```

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE OR REPLACE STREAM ncovstreamm ON TABLE COVID_MERGER;
+-----+
| status
+-----+
| Stream NCOVSTREAMM successfully created.
+-----+
1 Row(s) produced. Time Elapsed: 0.463s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```



The screenshot shows the Snowflake UI interface. At the top, there are navigation icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets, and History. On the right, there are links for Preview App, Partner Connect, Help, and a user account (NUMAN SYSADMIN). The History tab is selected, showing a recent run at 12:13:19 PM for 926ms. The SQL query executed was:

```
1 select * from ncov_stream;
```

The results section shows a table with the following data:

row#	FIPS	ADMIN2	PROVINCE...	COUNTRY...	LAST_UPD...	LAT	LONG	CC
6	NULL	NULL	NULL	Antigua and ...	2021-08-03 ...	17.0608	-61.7964	1
7	NI 111	NI 111	NI 111	Argentina	2021-08-03	-38.4161	-63.6167	1

3,987 rows produced. There is also an 'Export Result' button.

## Creation of Task -

Currently, a task can execute a single SQL statement, including a call to a stored procedure. Tasks can be combined with table streams for continuous ELT workflows to process recently changed table rows. Tasks can also be used independently to generate periodic reports by inserting or merging rows into a report table or perform other periodic work.

To automate the procedure of adding the contents of the streams into the target table, we create a task “**covid\_task**”. They automatically insert the data in the final table whenever there is new data added in the stream.

```

CREATE OR REPLACE TASK covid_task
WAREHOUSE = COMPUTE_WH
SCHEDULE = '1 MINUTE'
WHEN
    SYSTEM$STREAM_HAS_DATA('ncov_stream')
AS
    INSERT INTO covid_current SELECT fips, admin2, province_state, country_region,
last_update, lat, long, confirmed, deaths, recovered, active, combined_key, incident_rate,
case_fatality_ratio FROM ncov_stream;
  
```

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>create or replace task covid_task
    warehouse = compute_wh
    schedule = '1 minute'
    when
        system$stream_has_data('ncov_stream')
    as
        insert into covid_current select fips,admin2,province_state,COUNTRY_REGION,last_update,lat,long,confirmed,deaths,recovered,active,c
        ombined_key,incident_rate,case_fatality_ratio
            from ncov_stream;
+-----+
| status |
+-----+
| Task COVID_TASK successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.423s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
  
```

Row	created_on	name	id	database_name	schema_name	owner	comment	warehouse	schedule	predecessors	state	definition
1	2021-08-11 ...	COVID_TASK	019e3119-c1...	COVID_DB	SCHEMA	ACCOUNTA...		COMPUTE...	1 minute	NULL	suspended	insert into covid_current SELECT fips, admin2, province_state, country_region, last_update, lat, long, confirmed, deaths, recovered, active, combined_key, incident_rate, case_fatality_ratio FROM ncov_stream;

## Adding data in the external stage -

We added data in the S3 bucket that is the external bucket. Covid data of the month of **August 2021** is added into the stage.

### Before adding the data into the bucket -

row#	COUNT(*)
1	3987

### After adding the data in the external bucket-

row#	COUNT(*)
1	7974

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>--after adding day-8.csv of month 8
      select count(*) from covid_merger;

      select count(*) from covid_current;

+-----+
| COUNT(*) |
+-----+
| 27909 |
+-----+
1 Row(s) produced. Time Elapsed: 1.288s
+-----+
| COUNT(*) |
+-----+
| 865240 |
+-----+
1 Row(s) produced. Time Elapsed: 1.239s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>--after adding day-9.csv of month 8
      select count(*) from covid_merger;

      select count(*) from covid_current;

+-----+
| COUNT(*) |
+-----+
| 31896 |
+-----+
1 Row(s) produced. Time Elapsed: 1.200s
+-----+
| COUNT(*) |
+-----+
| 869227 |
+-----+
1 Row(s) produced. Time Elapsed: 1.232s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

### Creation of Secured share and Reader Account -

To share our database for the non-Snowflake user in order to use the data for various analytics, we create a secure share for the table which has the appropriate data. In secure share, the reader cannot perform any DML operations on the shared table reader can only see the data, that is, in **Read Only** mode.

```

CREATE SHARE "COVID_ANALYSIS" COMMENT="";
GRANT USAGE ON DATABASE "COVID_DB" TO SHARE "COVID_ANALYSIS";
GRANT USAGE ON SCHEMA "COVID_DB"."SCHEMA" TO SHARE "COVID_ANALYSIS";
GRANT SELECT ON VIEW "COVID_DB"."SCHEMA"."COVID_CONSUMPTION" TO SHARE "COVID_ANALYSIS"

```

Create a Secure Share and add Database objects to it

A Secure Share is a package or container comprising Database objects that contain the data you want to share. Once you setup Tables or Secure Views in the Database, ready with the data you intend to share, you can add them to the Secure Share.

Have you prepared the data? Identify and prepare data you want to include in the Secure Share. [Learn more](#) about preparing data.

**Create**

Select a Database and Schemas. Select Tables or Views within each Schema to add to the Secure Share.

Secure Share Name: COVID\_ANALYSIS

Database: COVID\_DB

Tables & Views: Select Tables & Secure Views | 1 Table, 0 Secure Views  
COVID\_DB.SCHEMA.COVID\_CONSUMPTION

Comment:

Show SQL | Cancel | Create

Review the Secure Share, Preview Tables & Validate Secure Views

Before you add consumer accounts to access the Secure Share, it's always a good practice to review the contents of your share and validate the data they will see.

✓ "COVID\_ANALYSIS" Secure share was created successfully!

Data Preview Secure Share Overview

Warehouse: COMPUTE\_WH (S) Suspended (auto resume)

Table or View: Select a Table or View

Preview Data

With a warehouse running, we run  
SELECT \* FROM <NAME> LIMIT 10;  
to generate a preview of the data in your Table

Show SQL | Done | Next! Add Consumers

After creating the secure share, we now create a reader account for the non snowflake user. For that we mention the name of the account and all the details like the username and the password for the account.

**Create Reader Account**

**Reader Account Details**

Provide a name for your new Reader account. The edition, region and cloud of your new Reader account will be the same as your current account.

Account Name: COVID\_READER

Comments: reader acc for non-snowflake user

Edition: ENTERPRISE

Region: US East (Virginia)

**Admin Login Information**

These are your reader account credentials, used to log in and manage the Reader account as the ACCOUNTADMIN role.

User Name: PowerBI

Password: \*\*\*\*\* Show Password

Confirm Password: \*\*\*\*\* Show Password

[Show SQL](#) Cancel Create Account

**COVID\_ANALYSIS**

Type: Outbound

Owner: ACCOUNTADMIN

Creation Time: 8/12/2021

Database: COVID\_DB

Add consumers to access your Secure Data Share

**Account**

Reader Accounts

Reader accounts enable providers to share data with consumers who...

[Create Reader Account](#) [Drop](#)

Search Reader Accounts: 2 Reader Accounts

Account Name	Locator
COVID_READER	ZLA65565
NUMANREADER	RUA50630

**Congratulations!**

You have created a new reader account "COVID\_READER".

Account URL: <https://zla65565.us-east-1.snowflakecomputing.com>

Locator: ZLA65565 [?](#)

What's next?

- Visit the [secure shares page](#) to create and provide access to your secure share.
- Once you are able to access your reader account, log in and ensure it's ready for your data consumers.

[Done](#)

**Secure Shares**

Inbound Outbound [Create](#) [Add Consumers](#) [Edit](#) [Drop](#)

Search Outbound Secure Shares: 5 Outbound Secure Shares

Secure Share Name	Shared With
COVID_ANALYSIS	<a href="#">Add Consumers</a>
COVID	QAA75167,ZLA36440
READKAR	RUA50630
OPMT	<a href="#">Add Consumers</a>
SH_NMN_DB	OFA91449,VKA00283

**Add Consumers**

Once added, the secure share is immediately "visible" and the account can create a database from the share and start querying the tables and secure views in the database. Currently you will only be able to share with accounts in the same cloud and region.

Secure Share: COVID\_ANALYSIS

Account Type:  Reader  Full

[?](#)

Create a Reader account and add it to this Secure Share.

[Show SQL](#) Cancel Add

**COVID\_ANALYSIS**

Type: Outbound

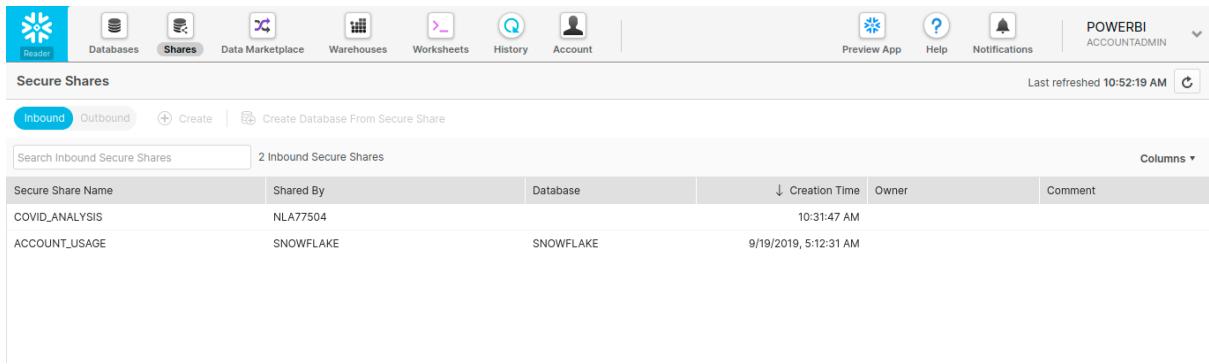
Owner: ACCOUNTADMIN

Creation Time: 8/12/2021

Database: COVID\_DB

Add consumers to access your Secure Data Share

After login in the Reader account, we change the role to ACCOUNTADMIN in order to create a shared database for the shared table to be used. Also we create a warehouse for the reader account, that can be used to run the queries to retrieve data from the shared table.



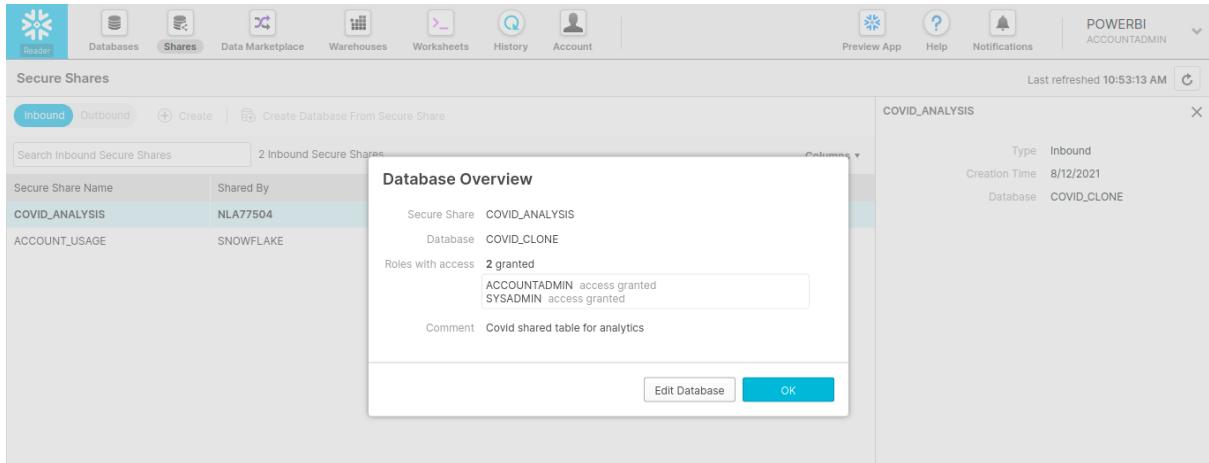
Secure Shares

Last refreshed 10:52:19 AM

Inbound Outbound + Create Create Database From Secure Share

Search Inbound Secure Shares 2 Inbound Secure Shares Columns ▾

Secure Share Name	Shared By	Database	↓ Creation Time	Owner	Comment
COVID_ANALYSIS	NLA77504		10:31:47 AM		
ACCOUNT_USAGE	SNOWFLAKE	SNOWFLAKE	9/19/2019, 5:12:31 AM		



Secure Shares

Last refreshed 10:53:13 AM

Inbound Outbound + Create Create Database From Secure Share

Search Inbound Secure Shares 2 Inbound Secure Shares

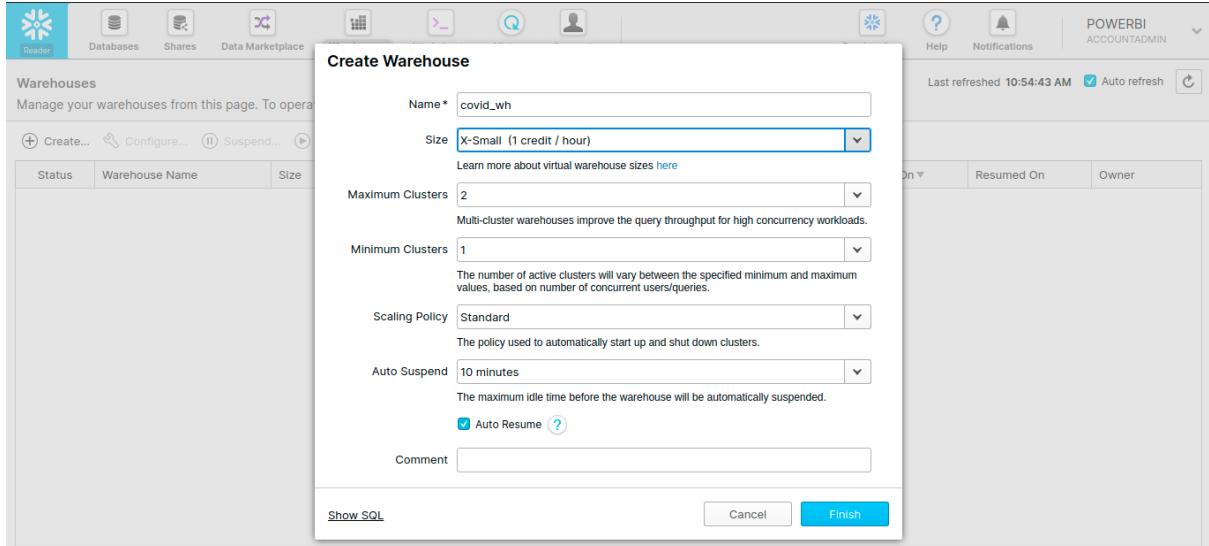
Secure Share Name	Shared By
COVID_ANALYSIS	NLA77504
ACCOUNT_USAGE	SNOWFLAKE

**Database Overview**

Secure Share: COVID\_ANALYSIS  
Database: COVID\_CLONE  
Roles with access: 2 granted  
Comment: Covid shared table for analytics

Type: Inbound  
Creation Time: 8/12/2021  
Database: COVID\_CLONE

Edit Database OK



Warehouses

Manage your warehouses from this page. To opera

+ Create... Configure... Suspend...

Status	Warehouse Name	Size
--------	----------------	------

**Create Warehouse**

Name\*: covid\_wh

Size: X-Small (1 credit / hour)

Learn more about virtual warehouse sizes [here](#)

Maximum Clusters: 2

Multi-cluster warehouses improve the query throughput for high concurrency workloads.

Minimum Clusters: 1

The number of active clusters will vary between the specified minimum and maximum values, based on number of concurrent users/queries.

Scaling Policy: Standard

The policy used to automatically start up and shut down clusters.

Auto Suspend: 10 minutes

The maximum idle time before the warehouse will be automatically suspended.

Auto Resume

Comment:

Show SQL Cancel Finish

Last refreshed 10:54:43 AM Auto refresh

The screenshot shows the Snowflake Worksheet interface. The top navigation bar includes 'Reader', 'Databases', 'Shares', 'Data Marketplace', 'Warehouses', 'Worksheets' (selected), 'History', and 'Account'. On the right, there are 'Preview App', 'Help', 'Notifications', and a 'POWERBI' section for 'ACCOUNTADMIN'. The main area shows a 'New Worksheet' with a 'Run' button and a query editor containing the following SQL:

```

1 use role accountadmin;
2
3 select * from "COVID_CLONE"."SCHEMA"."COVID_CONSUMPTION" where FIPS is not null limit 1000;

```

The results section shows a table with 1,000 rows. The columns are: Row, FIPS, ADMIN2, PROVINCE\_STATE, COUNTRY\_REGION, LAST\_UPDATE, LAT, LONG, CONFIRMED, DEATHS, and RECOVERED. The data is as follows:

Row	FIPS	ADMIN2	PROVINCE_STATE	COUNTRY_REGION	LAST_UPDATE	LAT	LONG	CONFIRMED	DEATHS	RECOVERED
1	1001	Autauga	Alabama	US	2021-08-08 ...	32.53952745	-86.64408227	7745	114	NUL
2	1003	Baldwin	Alabama	US	2021-08-08 ...	30.72774901	-87.72207058	27098	330	NUL
3	1005	Barbour	Alabama	US	2021-08-08 ...	31.868263	-85.3871286	2610	63	NUL
4	1007	Bibb	Alabama	US	2021-08-08 ...	32.99642064	-87.1251146	2990	66	NUL
5	1009	Blount	Alabama	US	2021-08-08 ...	33.98210918	-86.56790593	7567	140	NUL
6	1011	Bullock	Alabama	US	2021-08-08 ...	32.10030533	-85.71265535	1270	41	NUL
7	1013	Butler	Alabama	US	2021-08-08 ...	31.75300095	-86.68057478	2485	72	NUL
8	1015	Calhoun	Alabama	US	2021-08-08 ...	33.77483727	-85.82630386	15741	336	NUL
9	1017	Chambers	Alabama	US	2021-08-08 ...	32.91360079	-85.39072749	4088	125	NUL

## Creation of History table to Implement SCD (Type - 2) -

We have seen that the data has been added to the table “covid\_consumption”, but in order to implement the SCD(Slowly Changing Dimensions) and to get all the history of data stored. Hence we have another table named as “covid\_consumption\_history” which has 3 extra columns i.e the start\_date, end\_date and the current\_flag. These parameters tell us what was the time period when the data was current.

```

100 Row(s) produced. Time Elapsed: 2.987s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE OR REPLACE TABLE
    "COVID_DB"."SCHEMA"."COVID_CONSUMPTION"
    ("FIPS" NUMBER (7),
     "ADMIN2" VARCHAR (50),
     "PROVINCE_STATE" VARCHAR (50),
     "COUNTRY_REGION" VARCHAR (50),
     "LAST_UPDATE" TIMESTAMP,
     "LAT" FLOAT,
     "LONG" FLOAT,
     "CONFIRMED" INTEGER,
     "DEATHS" INTEGER,
     "RECOVERED" INTEGER,
     "ACTIVE" INTEGER,
     "COMBINED_KEY" VARCHAR (100),
     "INCIDENT_RATE" FLOAT,
     "CASE_FATLITY_RATIO" FLOAT,
     "UPDATE_TIME" TIMESTAMP_NTZ);
+-----+-----+
| status |-----|
|-----| Table COVID_CONSUMPTION successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 5.310s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE OR REPLACE TABLE
    "COVID_DB"."SCHEMA"."COVID_CONSUMPTION_HISTORY"
    ("FIPS" NUMBER (7),
     "ADMIN2" VARCHAR (50),
     "PROVINCE_STATE" VARCHAR (50),
     "COUNTRY_REGION" VARCHAR (50),
     "LAST_UPDATE" TIMESTAMP,
     "LAT" FLOAT,
     "LONG" FLOAT,
     "CONFIRMED" INTEGER,
     "DEATHS" INTEGER,
     "RECOVERED" INTEGER,
     "ACTIVE" INTEGER,
     "COMBINED_KEY" VARCHAR (100),
     "INCIDENT_RATE" FLOAT,
     "CASE_FATALITY_RATIO" FLOAT,
     "START_TIME" TIMESTAMP_NTZ,
     "END_TIME" TIMESTAMP_NTZ,
     "CURRENT" INTEGER
    );
+-----+
| status |
+-----+
| Table COVID_CONSUMPTION_HISTORY successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.452s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

We have created a stream that will monitor all the changes in the covid\_consumption table and also store the metadata for the same. After creating the stream, we have made a view that will combine all the DML operations like insert, update and delete and add the data in the three columns that are start\_date, end\_date and the current\_flag.

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>create or replace stream consume_stream on table covid_consumption;
+-----+
| status |
+-----+
| Stream CONSUME_STREAM successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.899s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>create or replace view covid_change_data as
  select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio,start_time,end_time,current_flag,'I' as dml_type
  from (select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio,
  update_timestamp as start_time,
  lag(update_timestamp) over (partition by (last_update,combined_key) order by update_timestamp desc) as end_time_raw,
  case when end_time_raw is null then '9999-12-31'::timestamp_ntz else end_time_raw end as end_time,
  case when end_time_raw is null then 1 else 0 end as current_flag
  from (select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio, update_timestamp
  from consume_stream
  where metadataaction = 'INSERT'
  and metadataisupdate = 'FALSE'))
  union

  select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio, start_time, end_time, current_flag, dml_type
  from (select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio,
  update_timestamp as start_time,
  lag(update_timestamp) over (partition by (last_update,combined_key) order by update_timestamp desc) as end_time_raw,
  case when end_time_raw is null then '9999-12-31'::timestamp_ntz else end_time_raw end as end_time,
  case when end_time_raw is null then 1 else 0 end as current_flag,
  dml_type
  from (
  select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio, update_timestamp, 'I' as dml_type
  from consume_stream
  where metadataaction = 'INSERT'
  and metadataisupdate = 'TRUE'
  union

  select null,null,null,null,last_update,null,null,null,
  null,null,combined_key,null,null,start_time, 'U' as dml_type
  from covid_consumption_history
  where (last_update,combined_key) in (select last_update,combined_key
  from consume_stream
  where metadataaction = 'INSERT'
  and metadataisupdate = 'TRUE')
  union

  select null,null,null,null,null,null,null,null,
  null,null,combined_key,null,null,start_time, 'U' as dml_type
  from covid_consumption_history
  where (last_update,combined_key) in (select last_update,combined_key
  from consume_stream
  where metadataaction = 'INSERT'
  and metadataisupdate = 'TRUE')
  and current_flag = 1))
  union

  select null,null,null,null,cs.last_update,null,null,null,
  null,null,cs.combined_key,null,null,cch.start_time, current_timestamp()::timestamp_ntz, null, 'D'
  from covid_consumption_history cch
  inner join consume_stream cs
  on cs.last_update = cch.last_update and
  cs.combined_key = cch.combined_key
  where cs.metadata_action = 'DELETE'
  and cs.metadata_isupdate = 'FALSE'
  and cch.current_flag = 1;
```

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>union
  select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio, start_time, end_time, current_flag, dml_type
  from (select fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio,
  update_timestamp as start_time,
  lag(update_timestamp) over (partition by (last_update,combined_key) order by update_timestamp desc) as end_time_raw,
  case when end_time_raw is null then '9999-12-31'::timestamp_ntz else end_time_raw end as end_time,
  case when end_time_raw is null then 1 else 0 end as current_flag,
  dml_type
  from (
  select fips,admin2,province state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio, update_timestamp, 'I' as dml_type
  from consume_stream
  where metadataaction = 'INSERT'
  and metadataisupdate = 'TRUE'
  union

  select null,null,null,null,null,null,null,null,
  null,null,combined_key,null,null,start_time, 'U' as dml_type
  from covid_consumption_history
  where (last_update,combined_key) in (select last_update,combined_key
  from consume_stream
  where metadataaction = 'INSERT'
  and metadataisupdate = 'TRUE')
  and current_flag = 1))
  union

  select null,null,null,null,cs.last_update,null,null,null,
  null,null,cs.combined_key,null,null,cch.start_time, current_timestamp()::timestamp_ntz, null, 'D'
  from covid_consumption_history cch
  inner join consume_stream cs
  on cs.last_update = cch.last_update and
  cs.combined_key = cch.combined_key
  where cs.metadata_action = 'DELETE'
  and cs.metadata_isupdate = 'FALSE'
  and cch.current_flag = 1;
```

```
num@ark:~
```

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>merge into covid_consumption_history cch
  using covid_change_data ccd
  on cch.last_update=ccd.last_update
  and cch.combined_key=ccd.combined_key
  and cch.start_time=ccd.start_time

  when matched and ccd.dml_type = 'U' then update -- Indicates the record has been updated and is no longer current and the end_time need
  s to be stamped
    set cch.end_time = ccd.end_time,
    cch.current_flag = 0
  when matched and ccd.dml_type = 'D' then delete -- Deletes are essentially logical deletes. The record is stamped and no newer version
  is inserted
    set cch.end_time = ccd.end_time,
    cch.current_flag = 0
  when not matched and ccd.dml_type = 'I' then insert -- Inserting a new n_nationkey and updating an existing one both result in an inser
  t
    (fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
  recovered,active,combined_key,incident_rate,case_fatality_ratio, start_time, end_time, current_flag)
    values (ccd.fips,ccd.admin2,ccd.province_state,ccd.country_region,ccd.last_update,ccd.lat,ccd.long,ccd.confirmed,ccd.deaths,
  ccd.recovered,ccd.active,ccd.combined_key,ccd.incident_rate,ccd.case_fatality_ratio, ccd.start_time, ccd.end_time, ccd.current_flag);

  number of rows inserted | number of rows updated |
  | 1593095 | 0 |
  1593095 Row(s) produced. Time Elapsed: 5.519s
  HUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>desc TABLE COVID_CONSUMPTION_HISTORY;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| name | type | kind | null? | default | primary key | unique key | check | expression | comment | policy name |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| FIPS | NUMBER(7,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| ADMIN2 | VARCHAR(50) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| PROVINCE_STATE | VARCHAR(50) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| COUNTRY_REGION | VARCHAR(50) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| LAST_UPDATE | TIMESTAMP_NTZ(9) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| LAT | FLOAT | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| LONG | FLOAT | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| CONFIRMED | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| DEATHS | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| RECOVERED | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| ACTIVE | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| COMBINED_KEY | VARCHAR(100) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| INCIDENT_RATE | FLOAT | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| CASE_FATALITY_RATIO | FLOAT | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| START_TIME | TIMESTAMP_NTZ(9) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| END_TIME | TIMESTAMP_NTZ(9) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
| CURRENT | NUMBER(38,0) | COLUMN | Y | NULL | N | N | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
17 Row(s) produced. Time Elapsed: 0.400s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

Using this view we will create a task that will merge the data in the view and set the right data in the covid\_consumption\_history table, inserting , updating and deleting data as required. In the view we have used a composite key to uniquely identify each row in the table so that we can change the data in the history table according to the changes in the covid\_consumption\_history table.

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select count(*) from consume_stream;
+-----+
| COUNT(*) |
+-----+
| 1596591 |
+-----+
1 Row(s) produced. Time Elapsed: 7.677s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

```

1 Row(s) produced. Time Elapsed: 0.475s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>insert into "COVID_DB"."SCHEMA".COVID_CONSUMPTION(fips,admin2,province_state,country_region,
n,last_update,lat,long,confirmed,deaths,
recovered,active,combined_key,incident_rate,case_fatality_ratio,update_timestamp) select f
ips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
recovered,active,combined_key,incident_rate,case_fatality_ratio,current_timestamp():::timestamp_ntz from "COVID_DB"."SCHEMA".COVID_CURRENT;
+-----+
| number of rows inserted |
+-----+
| 869227 |
+-----+
869227 Row(s) produced. Time Elapsed: 5.287s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>insert into "COVID_DB"."SCHEMA".COVID_CONSUMPTION(fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
recovered,active,combined_key,incident_rate,case_fatality_ratio,update_timestamp) select fips,admin2,province_state,country_region
, last_update,lat,long,confirmed,deaths,
recovered,active,combined_key,incident_rate,case_fatality_ratio,current_timestamp_ntz from "COVID_DB"."SCHEMA".COVID_HISTORY;
+-----+
| number of rows inserted |
+-----+
| 727364 |
+-----+
727364 Row(s) produced. Time Elapsed: 4.758s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select count(*) from COVID_CONSUMPTION;
+-----+
| COUNT(*) |
+-----+
| 1596591 |
+-----+
1 Row(s) produced. Time Elapsed: 1.268s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

```

The screenshot shows the Snowflake Worksheets interface. At the top, there are navigation icons for Databases, Shares, Data Marketplace, Warehouses, Worksheets (which is selected), and History. On the right, there are links for Preview App, Partner Connect, Help, and a user account labeled NUMAN SYSADMIN. The main area shows a query editor with a code editor and a results table. The code editor contains a series of SQL statements, including a view definition, a delete statement, and a select statement. The results table shows a single row of data with columns: LAT, LONG, CONFIRMED, DEATHS, RECOVERED, ACTIVE, INCIDENT\_RATE, CASE\_FATALITY, START\_TIME, END\_TIME, and CURRENT\_FLAG. The data for the row is: 35.6728299, -85.77969117, 5623, 84, NULL, NULL, Warren, Tennessee, US, 13622.5985..., 1.493864485, 2021-08-11, 2021-08-11, 0.

Now to automate this merge statement we created a task named “covid\_history\_task” that will conduct the merge statement that will add the proper data from the view into the covid\_consumption\_history table along with start\_time, end\_time and the current\_flag.

The screenshot shows a terminal window with the command: `NUMAN@COMPUTE_WH:~$ create task for automating scd changeLogs`. The terminal then displays the SQL code for the task:

```

1 Row(s) produced. Time Elapsed: 0.352s
NUMAN@COMPUTE_WH:~$ create task for automating scd changeLogs
create or replace task covid_history_task warehouse = task_warehouse schedule = '2 minute' when system$stream_has_data('consume_stream')
as
merge into covid_consumption_history cch
using covid_change_data ccd
on cch.last_update=ccd.last_update
and cch.combined_key=ccd.combined_key
and cch.start_time=ccd.start_time
when matched and ccd.dml_type = 'U' then update -- Indicates the record has been updated and is no longer current and the end_time needs
      to be stamped
      set cch.end_time = ccd.end_time,
          cch.current_flag = 0
when matched and ccd.dml_type = 'D' then update -- Deletes are essentially logical deletes. The record is stamped and no newer version is
      inserted
      set cch.end_time = ccd.end_time,
          cch.current_flag = 0
when not matched and ccd.dml_type = 'I' then insert -- Inserting a new n_nationkey and updating an existing one both result in an insert
      (fips,admin2,province_state,country_region,last_update,lat,long,confirmed,deaths,
      recovered,active,combined_key,incident_rate,case_fatality_ratio, start_time, end_time, current_flag)
      values (ccd.fips,ccd.admin2,ccd.province_state,ccd.country_region,ccd.last_update,ccd.lat,ccd.long,ccd.confirmed,ccd.deaths,
      ccd.recovered,ccd.active,ccd.combined_key,ccd.incident_rate,ccd.case_fatality_ratio, ccd.start_time, ccd.end_time, ccd.current_flag);
-----+
| status
|-----+
| Task COVID_HISTORY_TASK successfully created.
-----+
1 Row(s) produced. Time Elapsed: 0.336s
NUMAN@COMPUTE_WH:~$
```

Row	QUERY_ID	NAME	DATABASE_NAME	SCHEMA_NAME	QUERY_TEXT	CONDITION_TYPE	STATE	ERROR_CODE	ERROR_MESSAGE	SCHEDULED_TIME
92	NULL	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	SKIPPED	0040003	Conditional ...	2021-08-12 ..
93	NULL	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	SKIPPED	0040003	Conditional ...	2021-08-12 ..
94	NULL	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	SKIPPED	0040003	Conditional ...	2021-08-12 ..
95	NULL	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	SKIPPED	0040003	Conditional ...	2021-08-12 ..
96	019e3694-0...	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	FAILED	002003	SQL compila...	2021-08-12 ..
97	019e3696-0...	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	FAILED	002003	SQL compila...	2021-08-12 ..
98	019e3698-0...	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stream_h...	002003	SQL compila...	2021-08-12 ..	
99	019e369c-0...	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	SUCCEEDED	NULL	NULL	2021-08-12 ..
100	NULL	COVID_HIST...	COVID_DB	SCHEMA	merge into c...	system\$stre...	SCHEDULED	NULL	NULL	2021-08-12 ..

## Time Travel -

In order to show the time travel in a proper way, we set the retention period of both the tables “covid\_consumption” and “covid\_consumption\_history” to 30 days.

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>alter table covid_consumption set data_retention_time_in_days=30;
                                         alter table covid_consumption_history set data_retention_time_in_days=30;
+-----+-----+
| status |-----+
|-----| Statement executed successfully. |
+-----+-----+
1 Row(s) produced. Time Elapsed: 0.300s
+-----+-----+
| status |-----+
|-----| Statement executed successfully. |
+-----+-----+
1 Row(s) produced. Time Elapsed: 0.325s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

To show the time travel feature in the snowflake, we create a clone table named “covid\_clone\_tt” which will have the data from the “covid\_consumption” table before a particular timestamp.

We updated the covid\_consumption table then after a particular time, we created a clone table from the covid\_consumption table before that update.

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>--dml statement for task
--update
begin;
update covid_consumption
set admin2 = 'Numan', update_timestamp = current_timestamp()::timestamp_ntz
where FIPS = 1001 and confirmed=663;

update covid_consumption
set admin2 = 'Atharva', update_timestamp = current_timestamp()::timestamp_ntz
where FIPS = 1001 and confirmed=708;

update covid_consumption
set admin2 = 'Garima', update_timestamp = current_timestamp()::timestamp_ntz
where FIPS = 1001 and confirmed=872;

update covid_consumption
set admin2 = 'Mukesh', update_timestamp = current_timestamp()::timestamp_ntz
where FIPS = 1001 and confirmed=1029;
commit;
+-----+
| status |
+-----+
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 1.263s
+-----+
| number of rows updated | number of multi-joined rows updated |
+-----+
| 1 | 0 |
+-----+
1 Row(s) produced. Time Elapsed: 3.312s

```

As we can see the stream got updated with the updated values from the covid\_consumption table. Now using this stream we can see the data was updated from the table at a certain time.

```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select * from consume_stream;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| FIPS | ADMIN2 | PROVINCE_STATE | COUNTRY_REGION | LAST_UPDATE | METADATA$ACTION | METADATA$ISUPDATE | METADATA$ROW_ID |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1001 | Numan | Alabama | US | 2020-07-09 04:34:23.000 | INSERT | True | db61e8a30478507c0615ffb783476d7ae47b9da |
| 1001 | Atharva | Alabama | US | 2020-07-12 00:15:24.285 | INSERT | True | 708 | 14 | 0 | 662 | Autauga, Alabama, U
| 1001 | Garima | Alabama | US | 2020-07-12 00:15:27.659 | INSERT | True | 603827587e6f8a7d7665170e1eaa747fc90b20ae |
| 1001 | Mukesh | Alabama | US | 2020-07-31 04:35:18.000 | INSERT | True | 594d913850a6643095a9d3e2b979dd16085fd2d0 |
| 1001 | Autauga | Alabama | US | 2020-07-31 04:35:18.000 | INSERT | True | d76f6b7a5838153c662b85714798887095b96e11 |
| 1001 | Autauga | Alabama | US | 2020-07-09 04:34:23.000 | INSERT | True | 663 | 12 | 0 | 631 | Autauga, Alabama, U
| 1001 | Autauga | Alabama | US | 2020-07-11 05:54:29.469 | DELETE | True | db61e8a30478507c0615ffb783476d7ae47b9da |
| 1001 | Autauga | Alabama | US | 2020-07-12 04:34:30.000 | INSERT | True | 708 | 14 | 0 | 662 | Autauga, Alabama, U
| 1001 | Autauga | Alabama | US | 2020-07-21 04:38:46.000 | INSERT | True | 603827587e6f8a7d7665170e1eaa747fc90b20ae |
| 1001 | Autauga | Alabama | US | 2020-07-31 04:35:18.000 | INSERT | True | 872 | 20 | 0 | 818 | Autauga, Alabama, U
| 1001 | Autauga | Alabama | US | 2020-07-31 04:35:18.000 | INSERT | True | 594d913850a6643095a9d3e2b979dd16085fd2d0 |
| 1001 | Autauga | Alabama | US | 2020-07-31 04:35:18.000 | INSERT | True | 1029 | 20 | 0 | 943 | Autauga, Alabama, U
| 1001 | Autauga | Alabama | US | 2020-07-31 04:35:18.000 | DELETE | True | d76f6b7a5838153c662b85714798887095b96e11 |
+-----+
8 Row(s) produced. Time Elapsed: 1.234s

```

After this we create a clone table with the data from the covid\_consumption table before this update was done in order to demonstrate **Time Travel**.

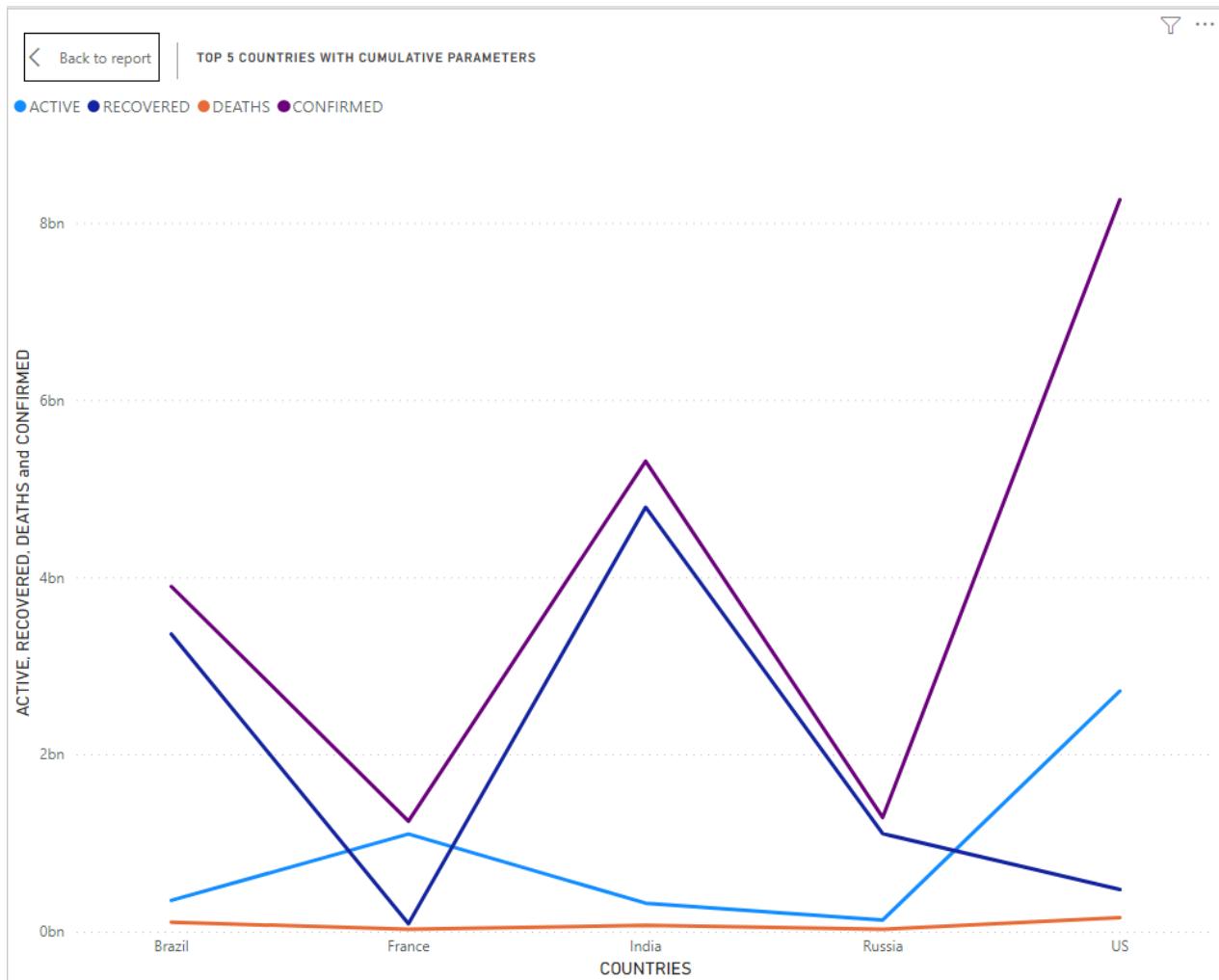
```

NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE TABLE "COVID_DB".SCHEMA".COVID_CLONE_TT CLONE "COVID_DB".SCHEMA".COVID_CONSUMPTION"
before(TIMESTAMP=>'2021-08-12 00:15:00.000 +0530' :: timestamp_tz);
+-----+
| status |
+-----+
| Table COVID_CLONE_TT successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 6.770s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select fips,admin2,last_update,confirmed from covid_clone_tt where FIPS = 1001 and confirmed=663
union
select fips,admin2,last_update,confirmed from covid_clone_tt where FIPS = 1001 and confirmed=708
union
select fips,admin2,last_update,confirmed from covid_clone_tt where FIPS = 1001 and confirmed=872
union
select fips,admin2,last_update,confirmed from covid_clone_tt where FIPS = 1001 and confirmed=1029;
+-----+-----+-----+-----+
| FIPS | ADMIN2 | LAST_UPDATE | CONFIRMED |
+-----+-----+-----+-----+
| 1001 | Autauga | 2020-07-09 04:34:23.000 | 663 |
| 1001 | Autauga | 2020-07-31 04:35:18.000 | 1029 |
| 1001 | Autauga | 2020-07-12 04:34:30.000 | 708 |
| 1001 | Autauga | 2020-07-21 04:38:46.000 | 872 |
+-----+
4 Row(s) produced. Time Elapsed: 2.053s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>

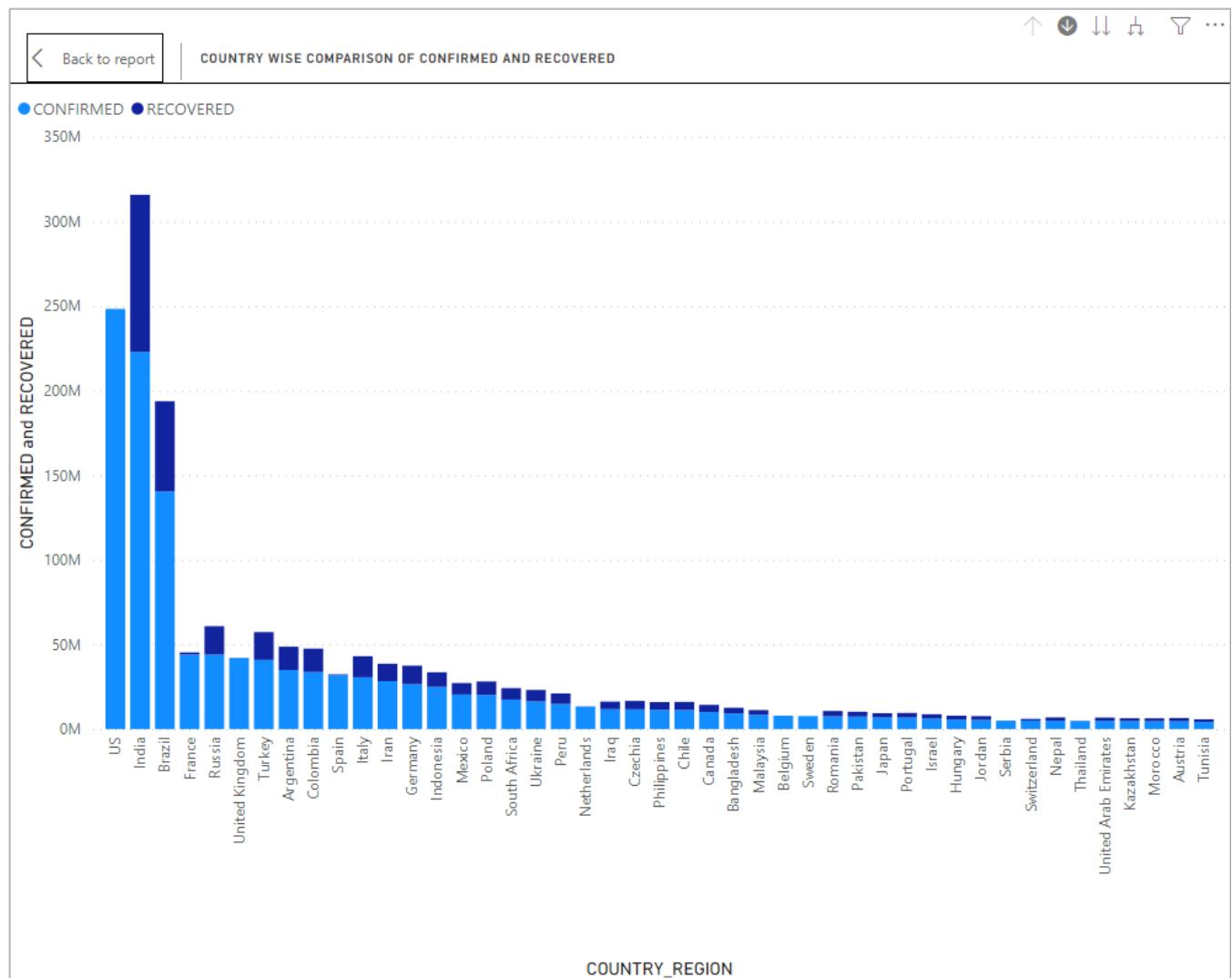
```

# REPORTS

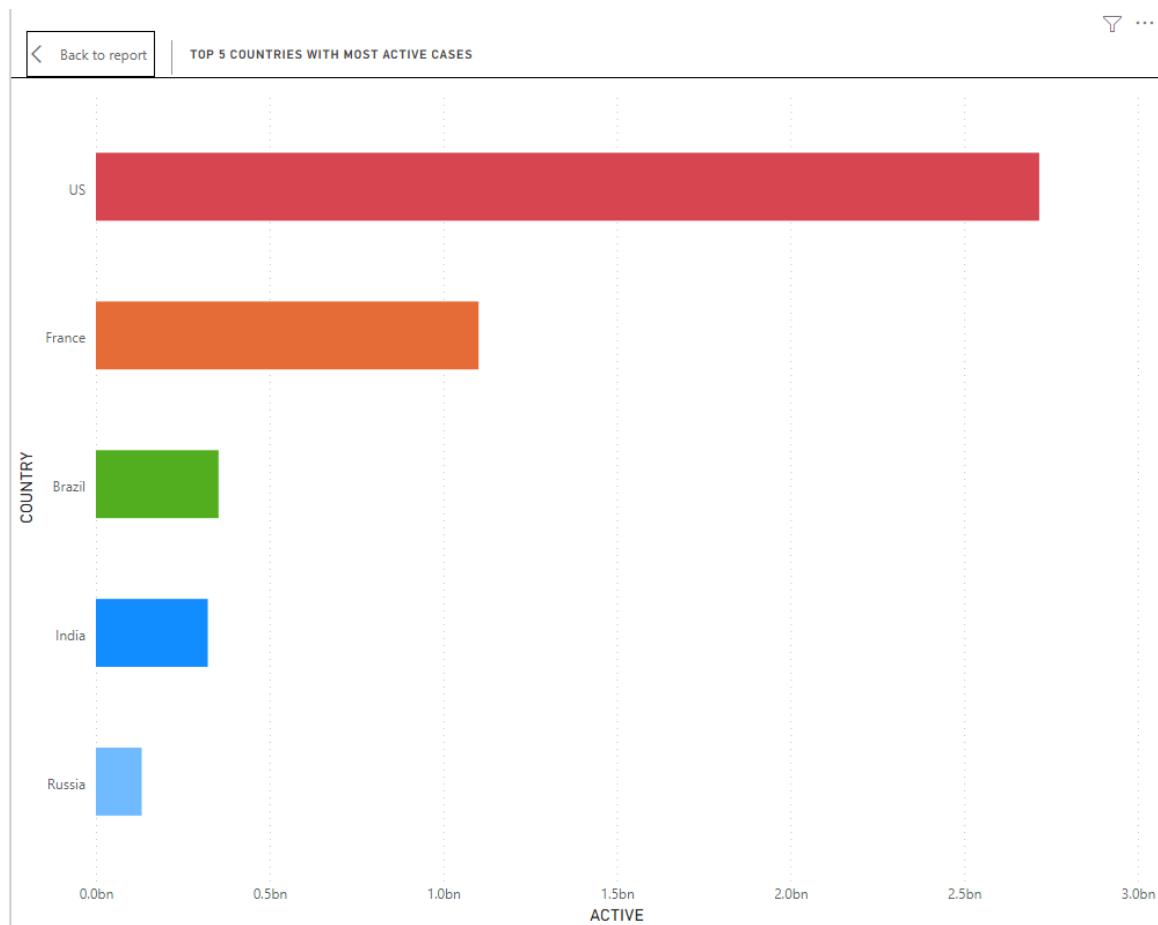
## TOP 5 COUNTRIES WITH CUMULATIVE PARAMETERS



## COUNTRY WISE COMPARISON OF CONFIRMED AND RECOVERED CASES



## TOP 5 COUNTRIES WITH MOST ACTIVE CASES



## INCIDENT RATE BY COUNTRY REGION

