Project Gladiator

Healthcare Case Study

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

INDEX

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.

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.

Data Ingestion

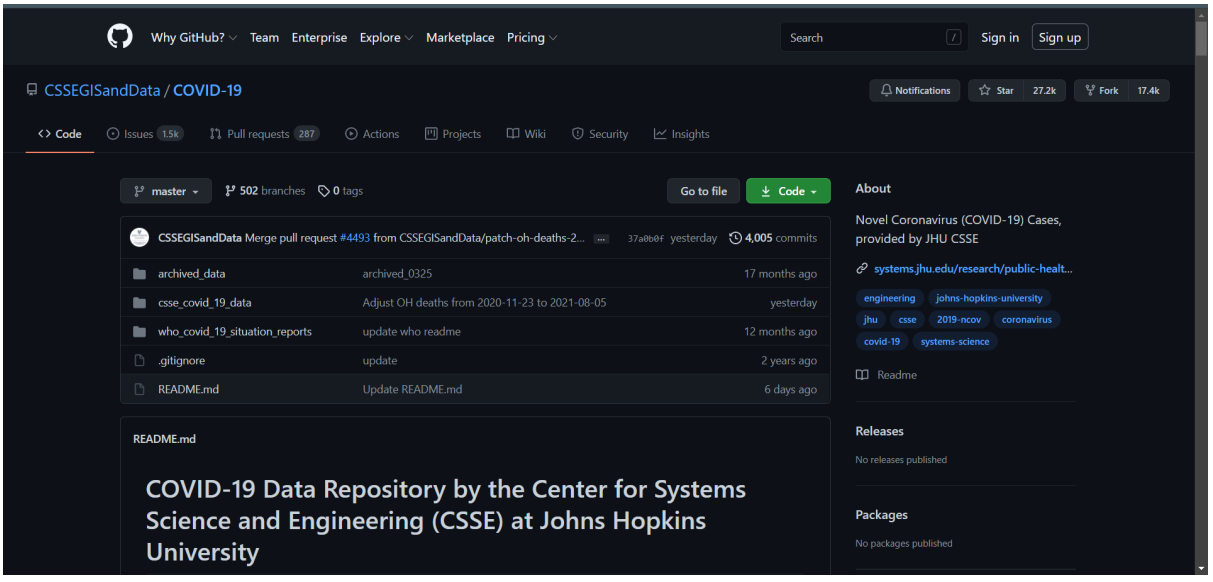
Data ingestion is a process by which data is moved from one or more sources to a destination where it can be stored and further analysed. The data might be in different formats and come from various sources, including RDBMS, other types of databases, S3 buckets, CSVs, or from streams. Since the data 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----------

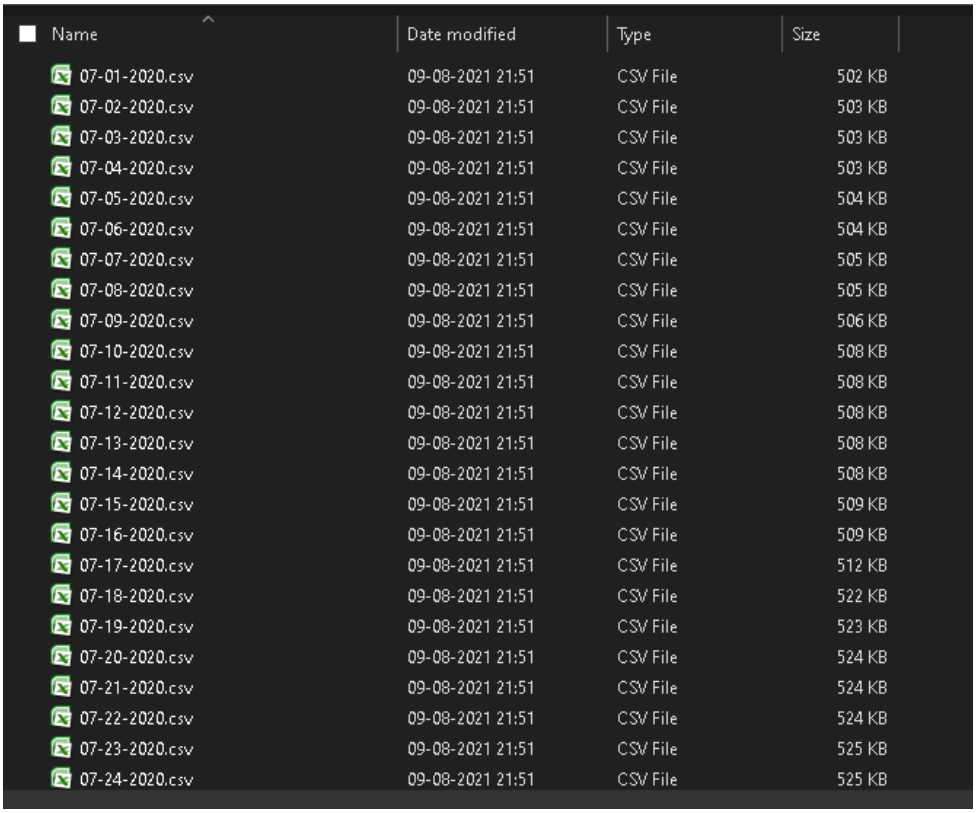
Name of table	Purpose	Date wise segregation	Rows & columns observed
COVID_HISTORY	To store historical covid statistics	1st July 2020-31st Dec 2020	7,27,364 rows and 14 columns
COVID_CURRENT	To store current covid statistics	1st Jan 2021- 8th Aug 2021	8,77,102 rows and 14 columns
COVID_CONSUMPTION	To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

DATASET -



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 csse 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.



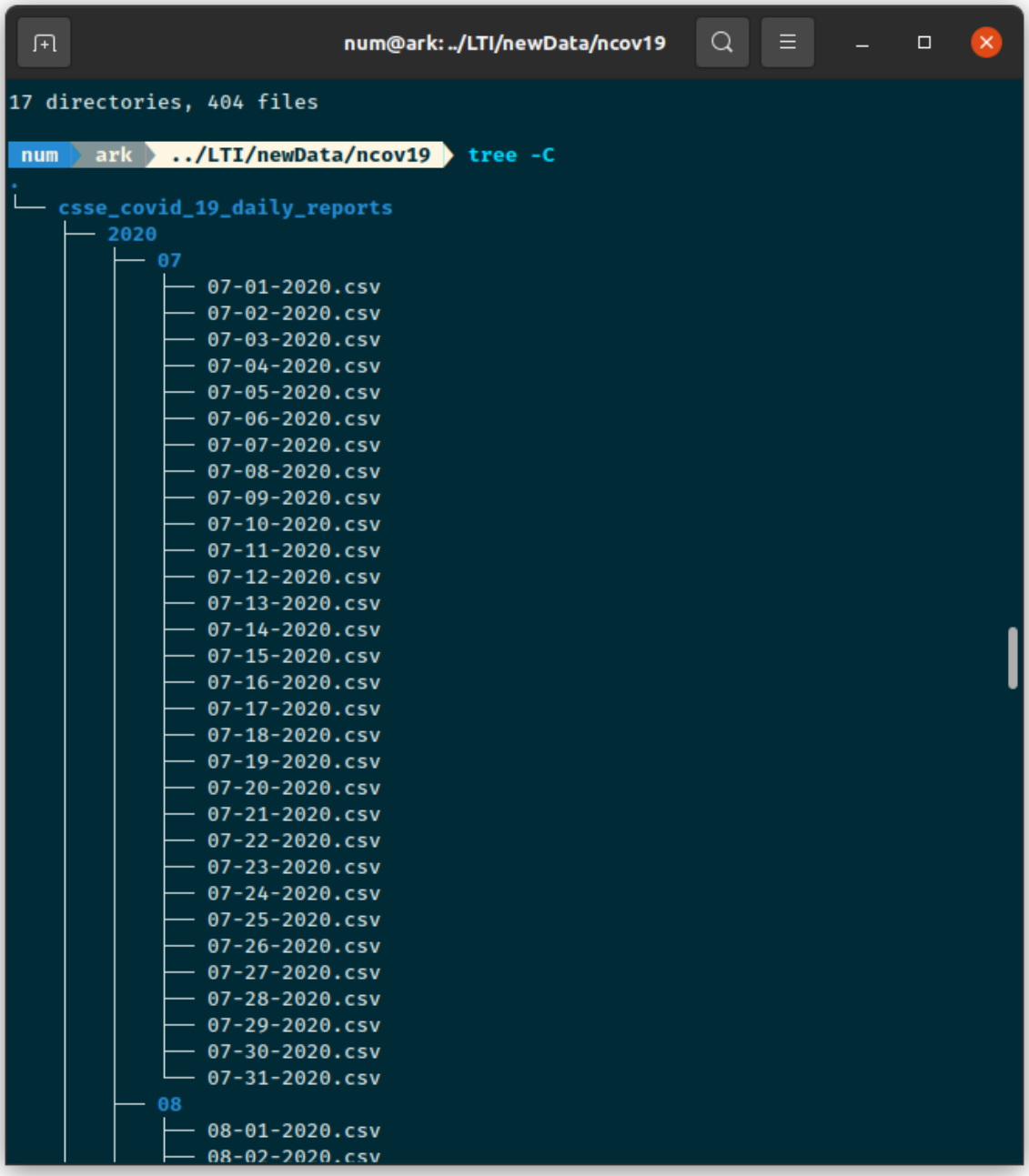
Name	Date modified	Type	Size
07-01-2020.csv	09-08-2021 21:51	CSV File	502 KB
07-02-2020.csv	09-08-2021 21:51	CSV File	503 KB
07-03-2020.csv	09-08-2021 21:51	CSV File	503 KB
07-04-2020.csv	09-08-2021 21:51	CSV File	503 KB
07-05-2020.csv	09-08-2021 21:51	CSV File	504 KB
07-06-2020.csv	09-08-2021 21:51	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

The above csv files in the data set were singular or we can say ungrouped, so file- tracking would be tedious. Hence, in order to overcome this, a Python script was created to categorize the files into a folder on the basis of the month of the csv files. This also helps us in the internal staging area to load the files in a systematic manner. The folder hierarchy can be seen as below


```
import os
import shutil as sh
count=0

#Path of directory where all csv files exists
for file in os.listdir("/home/num/Numan/LTI/newData/COVID-19/csse_covid_19_data/csse_covid_19_daily_reports"):
    if file.endswith(".csv"):
        month=file[0:2]
        #date=file[3:5]
        year=file[6:10]
        print(month+year)

        if not os.path.exists("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports/"+year):
            os.mkdir("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports//"+year)

        if not os.path.exists("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports/"+year+"/"+month):
            os.mkdir("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports//"+year+"/"+month)

        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/-
csse_covid_19_daily_reports/"+year+"/"+month+"/"+file)
        count+=1

print(count)
```

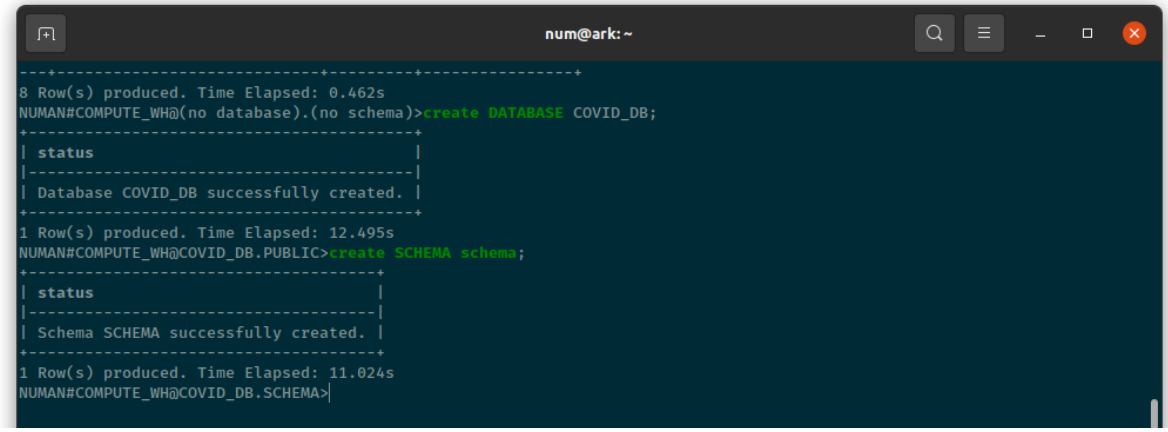

```
import os
import shutil as sh
count=0

#Path of directory where all csv files exists
for file in
os.listdir("/home/num/Numan/LTI/newData/COVID-19/csse_covid_19_data/csse_cov
id_19_daily_reports"):
    if file.endswith(".csv"):
        month=file[0:2]
        #date=file[3:5]
        year=file[6:10]
        print(month+year)
                if not
os.path.exists("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_repor
ts/"+year):

os.mkdir("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports//"+
year)
                if not
os.path.exists("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_repor
ts/"+year+"/"+month):

os.mkdir("/home/num/Numan/LTI/newData/ncov19/csse_covid_19_daily_reports//"+
year+"/"+month)

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/csse_covid_19_da
ily_reports/"+year+"/"+month+"/"+file)
        count+=1
```

```
print(count)
```



DATABASE AND SCHEMA CREATION

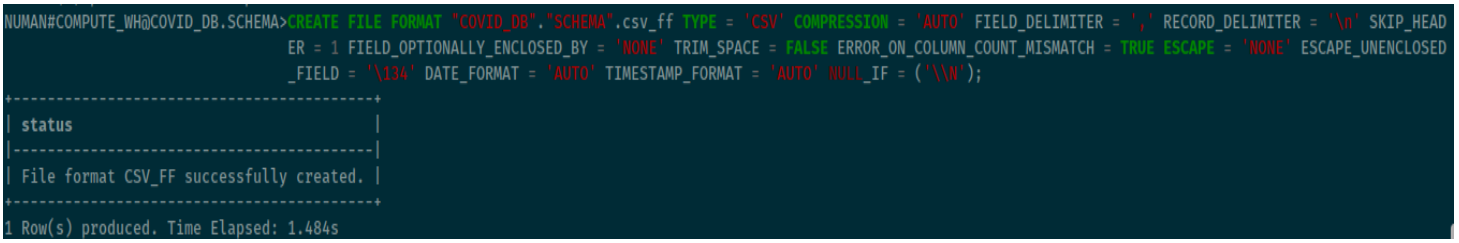
```
CREATE DATABASE covid_db;

CREATE SCHEMA "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"."PUBLIC".bgy TYPE = 'CSV'
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');
```



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>
```

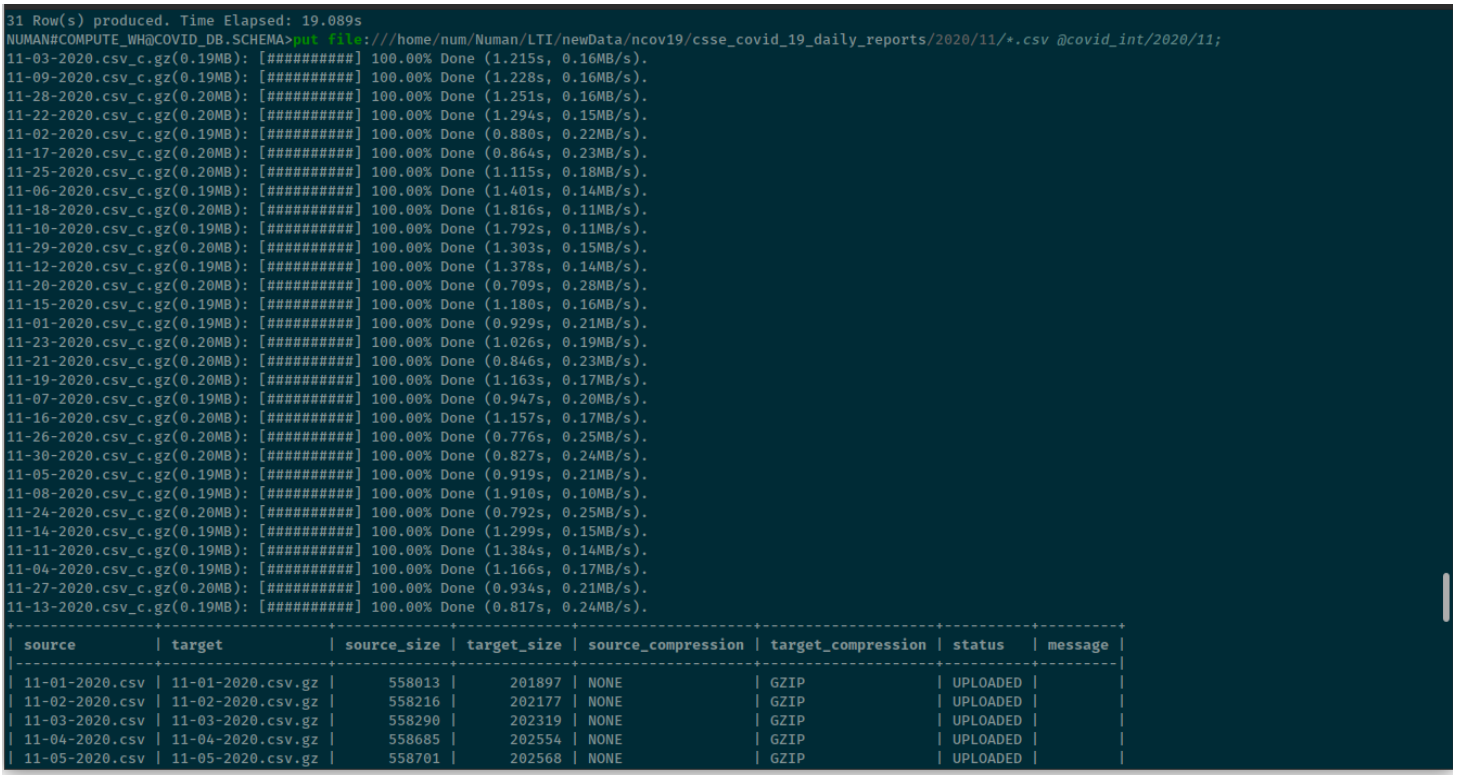
TABLE CREATION AND LOADING

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);

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE TABLE "COVID_DB"."SCHEMA"."COVID_HISTORY" ("FIPS" NUMBER (7), "ADMIN2" VARCHAR (50), "PROVINCE_STATE" VARCHAR (50), "COUNTRY_REG
                                ION" VARCHAR (50), "LAST_UPDATE" DATE, "LAT" NUMBER (4,4), "LONG" NUMBER (4,4), "CONFIRMED" INTEGER, "DEATHS" INTEGER, "RECOVERED" INTE
                                GER, "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        |
| 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         | DATE         | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    | NULL        |
| LAT                 | NUMBER(4,4)  | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       | NULL    | NULL        |
| LONG                | NUMBER(4,4)  | 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        |
+---------------------+--------------+--------+-------+---------+-------------+------------+-------+------------+---------+-------------+
14 Row(s) produced. Time Elapsed: 4.559s
```

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;
```

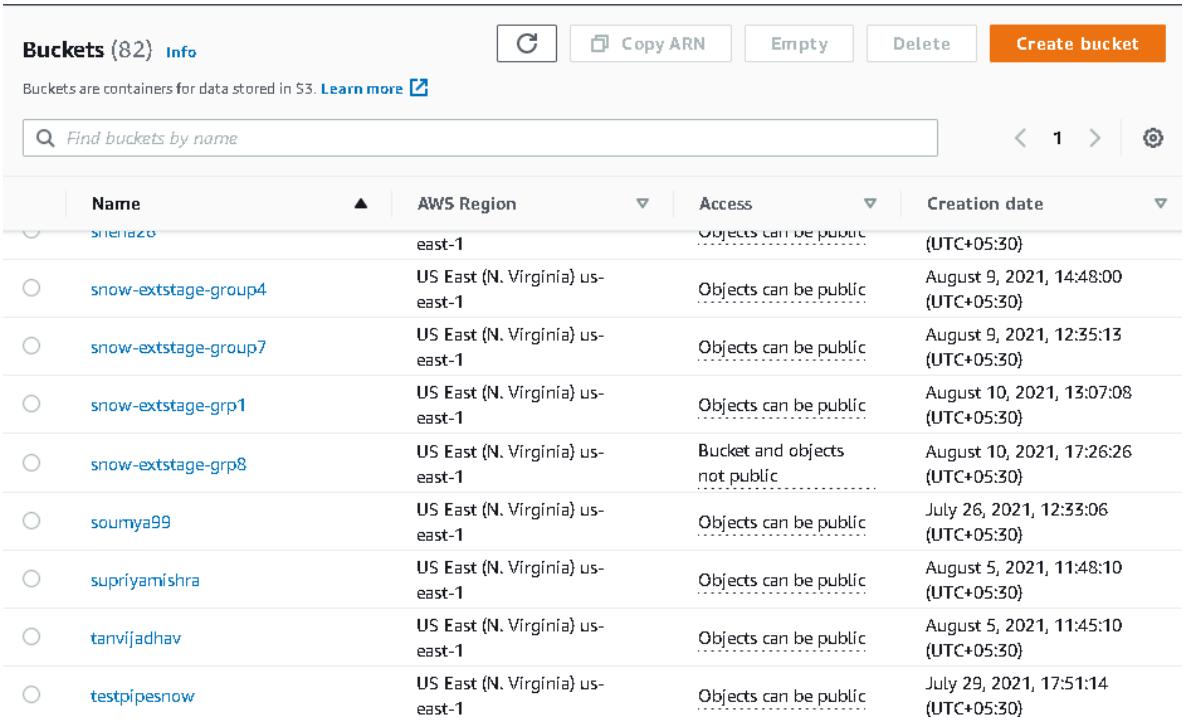
We copy the staged .csv files from the internal stage @covid_int to the table covid_history.

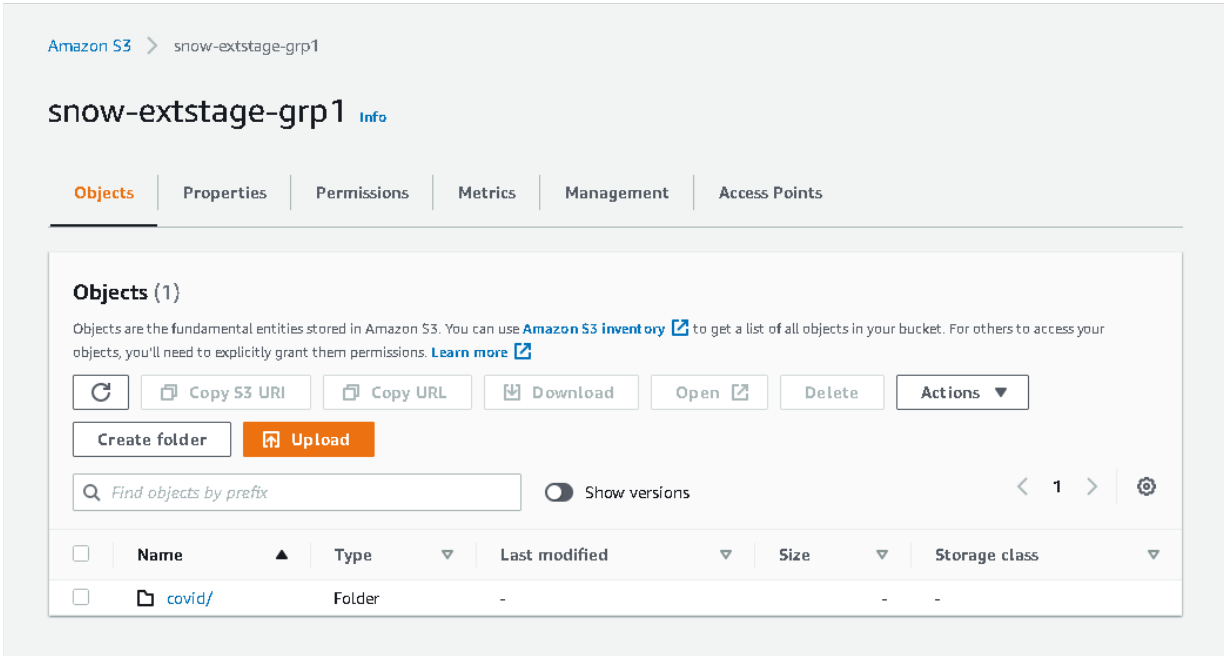
```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>COPY INTO COVID_HISTORY FROM (SELECT t.$1,t.$2,t.$3,t.$4,t.$5,t.$6,t.$7,t.$8,t.$9,t.$10,t.$11,to_char(t.$12),t.$13,t.$14
                                FROM @covid_int (file_format => 'csv_ff' ) t) on_error = 'ABORT_STATEMENT' PURGE = FALSE;
+-----------------------------------+--------+-------------+-------------+-------------+-------------+-------------+------------------+----------------------+--------
----------------+
| file                              | status | rows_parsed | rows_loaded | error_limit | errors_seen | first_error | first_error_line | first_error_character | first_e
rror_column_name |
|-----------------------------------+--------+-------------+-------------+-------------+-------------+-------------+------------------+----------------------+--------
----------------|
| covid_int/2020/07/07-22-2020.csv.gz | LOADED |        3935 |        3935 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/08/08-11-2020.csv.gz | LOADED |        3952 |        3952 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/08/08-24-2020.csv.gz | LOADED |        3959 |        3959 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/09/09-20-2020.csv.gz | LOADED |        3963 |        3963 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/09/09-27-2020.csv.gz | LOADED |        3965 |        3965 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/10/10-20-2020.csv.gz | LOADED |        3968 |        3968 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/11/11-04-2020.csv.gz | LOADED |        3970 |        3970 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/11/11-26-2020.csv.gz | LOADED |        3984 |        3984 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/12/12-09-2020.csv.gz | LOADED |        3984 |        3984 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
| covid_int/2020/12/12-21-2020.csv.gz | LOADED |        3984 |        3984 |           1 |           0 | NULL        |             NULL |                 NULL | NULL
                |
```

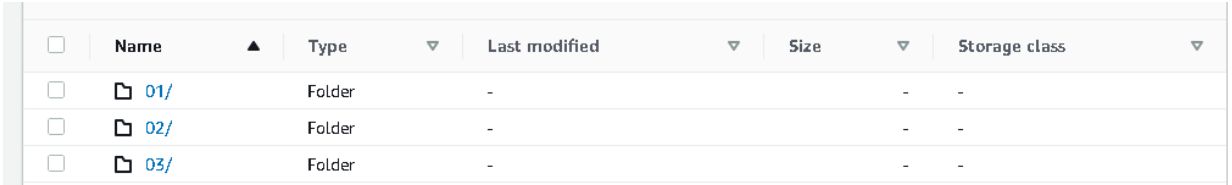
Total Rows Inserted - 7,27,364

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.





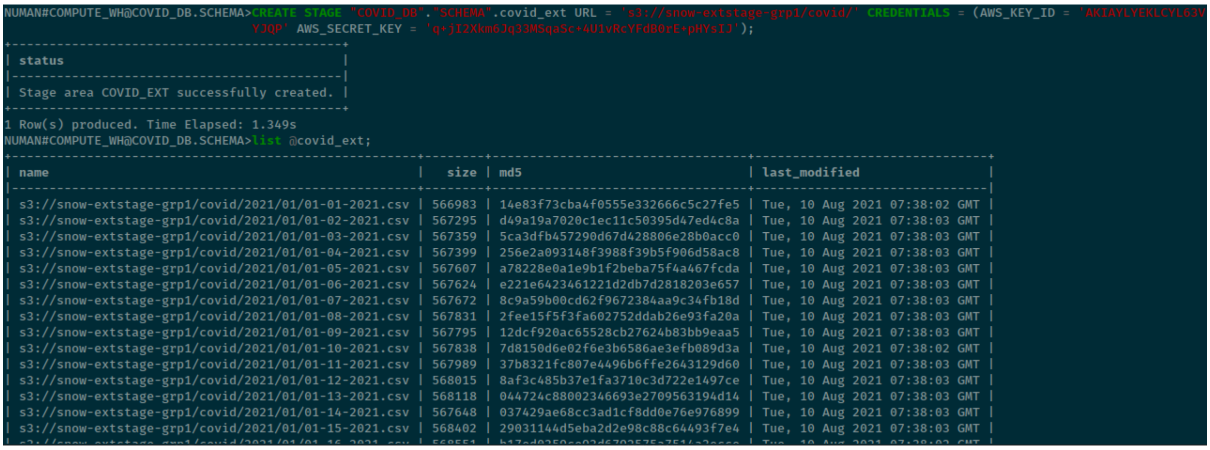


We create a table COVID_CURRENT to store this current data

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

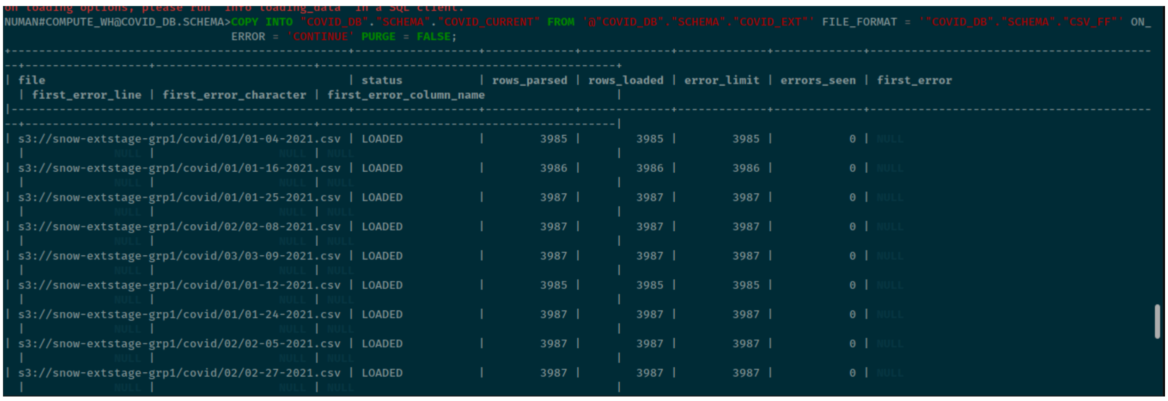
CREATE STAGE "COVID_DB"."SCHEMA".covid_ext URL =
's3://snow-extstage-grp1/covid/' CREDENTIALS = (AWS_KEY_ID =
'AKIAYLYEKLCYL63VYJQP' AWS_SECRET_KEY =
'q+jI2Xkm6Jq33MSqaSc+4U1vRcYFdB0rE+******');



```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE STAGE "COVID_DB"."SCHEMA".covid_ext URL = 's3://snow-extstage-grp1/covid/' CREDENTIALS = (AWS_KEY_ID = 'AKIAYLYEKLCYL63V
                                 YJQP' AWS_SECRET_KEY = 'q+jI2Xkm6Jq33MSqaSc+4U1vRcYFdB0rE+pHYsIJ');
+-----------------------------------------+
| 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 | 14e83f73cba4f0555e332666c5c27fe5 | Tue, 10 Aug 2021 07:38:02 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-02-2021.csv | 567295 | d49a19a7020c1ec11c50395d47ed4c8a | 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 | 256e2a093148f3988f39b5f906d58ac8 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-05-2021.csv | 567607 | a78228e0a1e9b1f2beba75f4a467fcda | 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 | 8c9a59b00cd62f9672384aa9c34fb18d | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-08-2021.csv | 567831 | 2fee15f5f3fa602752ddab26e93fa20a | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-09-2021.csv | 567795 | 12dcf920ac65528cb27624b83bb9eaa5 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-10-2021.csv | 567838 | 7d8150d6e02f6e3b6586ae3efb089d3a | Tue, 10 Aug 2021 07:38:02 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-11-2021.csv | 567989 | 37b8321fc807e4496b6ffe2643129d60 | 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 | 044724c88002346693e2709563194d14 | Tue, 10 Aug 2021 07:38:03 GMT |
| s3://snow-extstage-grp1/covid/2021/01/01-14-2021.csv | 567648 | 037429ae68cc3ad1cf8dd0e76e976899 | 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
  | first_error_line | first_error_character | first_error_column_name                 |
| s3://snow-extstage-grp1/covid/01/01-04-2021.csv | LOADED |  3985 |  3985 |  3985 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/01/01-16-2021.csv | LOADED |  3986 |  3986 |  3986 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/01/01-25-2021.csv | LOADED |  3987 |  3987 |  3987 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/02/02-08-2021.csv | LOADED |  3987 |  3987 |  3987 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/03/03-09-2021.csv | LOADED |  3987 |  3987 |  3987 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/01/01-12-2021.csv | LOADED |  3985 |  3985 |  3985 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/01/01-24-2021.csv | LOADED |  3987 |  3987 |  3987 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/02/02-05-2021.csv | LOADED |  3987 |  3987 |  3987 |  0 | NULL
  |             NULL |                  NULL | NULL                                    |
| s3://snow-extstage-grp1/covid/02/02-27-2021.csv | LOADED |  3987 |  3987 |  3987 |  0 | NULL
  |             NULL |                  NULL | 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/')
```

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>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/');
+-------------------------------------------------+
```

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** through the pipes and add to the event notification.

Edit Trust Relationship

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

Policy Document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::099555862548:user/ovzt-s-v2sa7501"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "NLA77504_SFCRole=3_HO8J6U/rQ1li9xAR0ycyhNQ2Z9Q="
        }
      }
    }
  ]
}
```

Add SQS Event notification -

Event notifications (1)

Send a notification when specific events occur in your bucket. Learn more

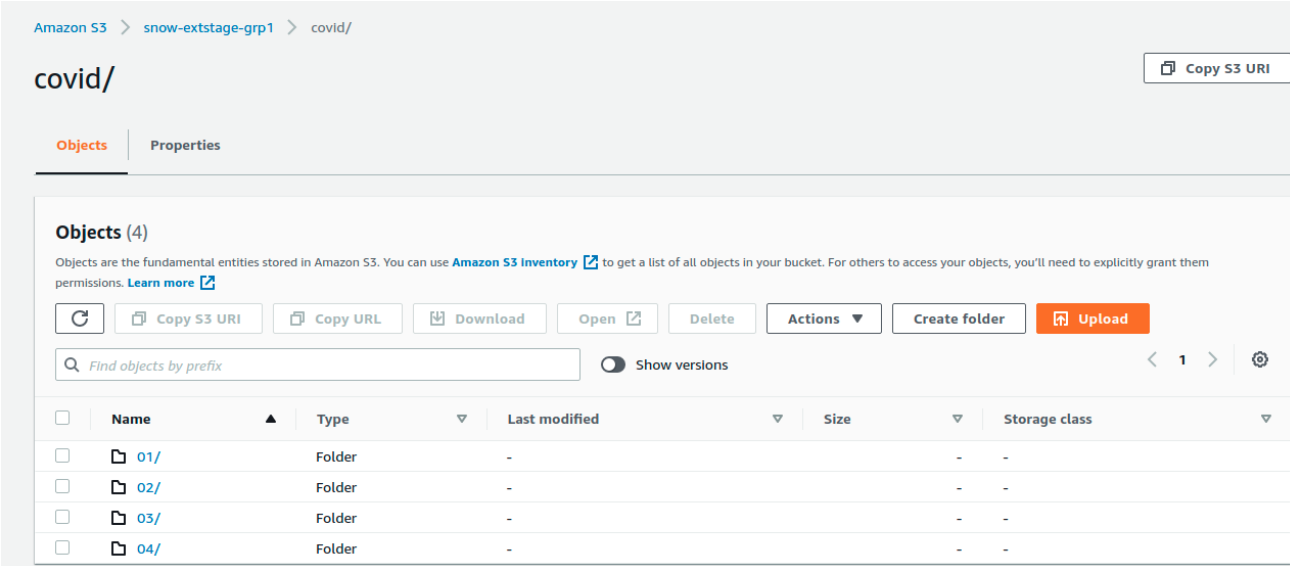
Edit | Delete | Create event notification

	Name	Event types	Filters	Destination type	Destination
☑	s3_current_pipe	All object create events	-	SQS queue	

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-sjq7fOwwn4GRodf5chha_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 -



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 -

```
| 2021-08-10 02:36:38.964 -0700 | COVID_CURRENT | COVID_DB      | SCHEMA      | TABLE |         |                                       | 877102 | 31853568 | SYSADMIN |
 1              | OFF                   | OFF             | OFF                  |                        NULL |                         NULL | N           |
| 2021-08-09 23:38:11.295 -0700 | COVID_HISTORY | COVID_DB      | SCHEMA      | TABLE |         | LINEAR(COUNTRY_REGION,province_state) | 727364 | 23845376 | SYSADMIN |
 1              | ON                    | OFF             | OFF                  |                        NULL |                         NULL | N           |

2 Row(s) produced. Time Elapsed: 1.259s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>show tables;

| created_on                    | name          | database_name | schema_name | kind  | comment | cluster_by                            |   rows |    bytes | owner    |
 retention_time | automatic_clustering | change_tracking | search_optimization | search_optimization_progress | search_optimization_bytes | is_external |

| 2021-08-10 02:36:38.964 -0700 | COVID_CURRENT | COVID_DB      | SCHEMA      | TABLE |         | LINEAR(COUNTRY_REGION,province_state) | 877102 | 26112000 | SYSADMIN |
 1              | ON                    | OFF             | OFF                  |                        NULL |                         NULL | N           |
| 2021-08-09 23:38:11.295 -0700 | COVID_HISTORY | COVID_DB      | SCHEMA      | TABLE |         | LINEAR(COUNTRY_REGION,province_state) | 727364 | 23845376 | SYSADMIN |
 1              | ON                    | OFF             | OFF                  |                        NULL |                         NULL | N           |

2 Row(s) produced. Time Elapsed: 4.837s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

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>
```

----------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.

Results Data Preview — Open History

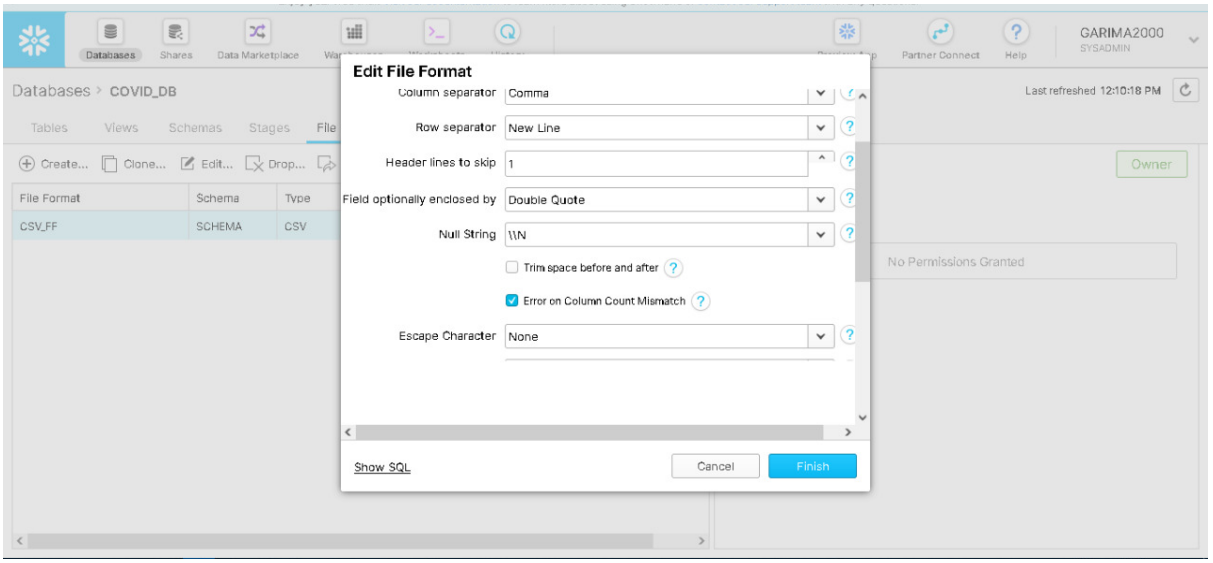
Query ID SQL 567ms 3,800 rows

Filter result... Copy Columns

Row	$1	$2	$3	$4	$5	$6	$7	$8	$9	$10	$11	$12	$13	$14
1	FIPS	Admin2	Province_Sta...	Country_Reg...	...t_Update	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	-92.60078167	90	0	0	90	"Adair	Missouri	US"
9	40001	Adair	Oklahoma	US	2020-07-02 ...	35.88494195	-94.65859267	115	4	0	111	"Adair	Oklahoma	US"
10	8001	Adams	Colorado	US	2020-07-02 ...	39.87432092	-104.3362578	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	-91.18786813	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	-94.69932645	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');
```



Creation of Streams -

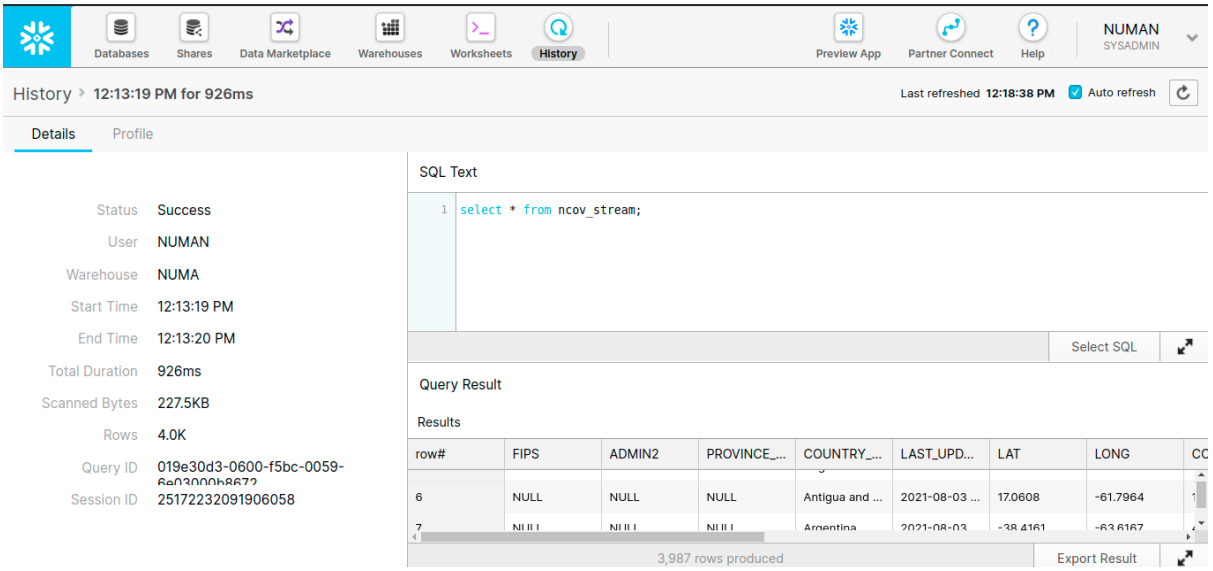
A stream object records data manipulation language (DML) changes made to tables, including inserts, updates, and deletes, as well as metadata about each change, so that actions can be taken using the changed data. This process is referred to as **change data capture (CDC)**. An individual table stream 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>
```

METADATA$ACTION	METADATA$ISUPDATE	METADATA$ROW_ID
INSERT	TRUE	943f305c36accd512bc8d90700ea96d1bee777de
DELETE	TRUE	943f305c36accd512bc8d90700ea96d1bee777de



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

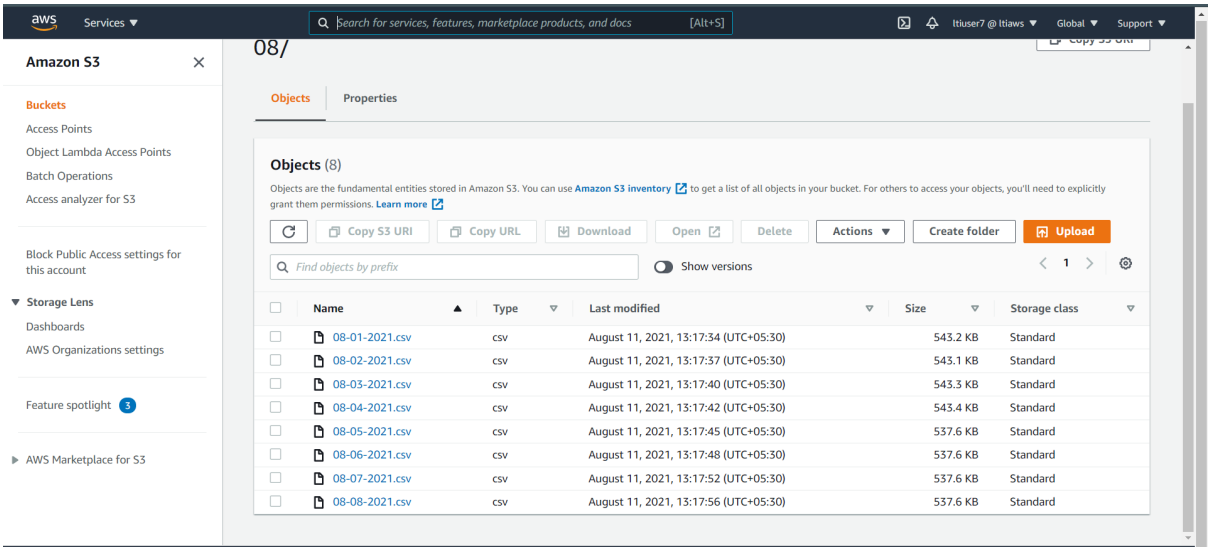
To automate the procedure of adding the contents of the streams into the target table, we create a task "**covid_task**". They automatically insert the data in the final table whenever there is new data added in the stream.

```
CREATE OR REPLACE TASK covid_task
WAREHOUSE = COMPUTE_WH
SCHEDULE = '15 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_ke
y,incident_rate, case_fatality_ratio  FROM ncov_stream;
```

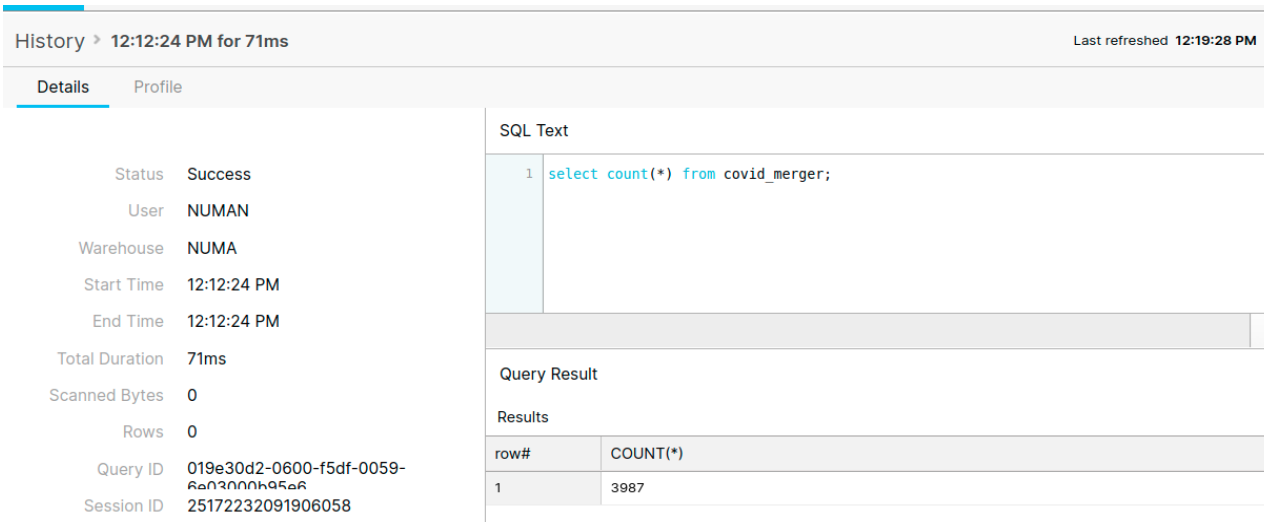
```
| status
|----------------------------------------|
| Task COVID_TASK successfully created.  |
+----------------------------------------+
1 Row(s) produced. Time Elapsed: 0.423s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

Adding data in the external stage -

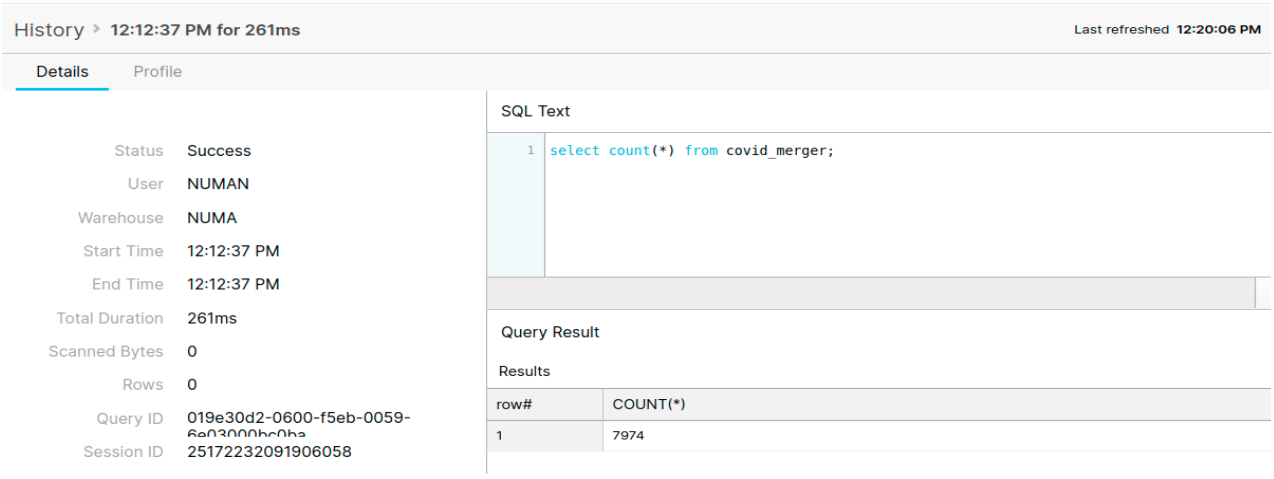
We added data in the S3 bucket that is the external bucket. Covid data of the month of **August 2021** is added into the stage.



Before adding the data into the bucket -



After adding the data in the external bucket-



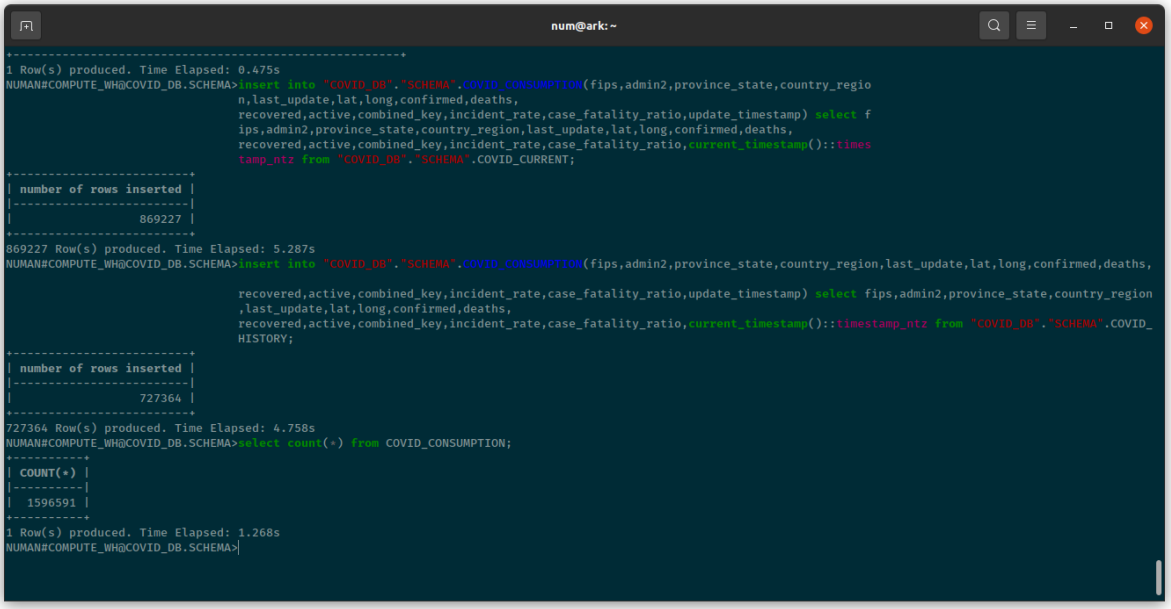
```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>--after adding day-8.csv of month 8
                                select count(*) from covid_merger;

                                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 History table to Implement SCD (Type - 2) -

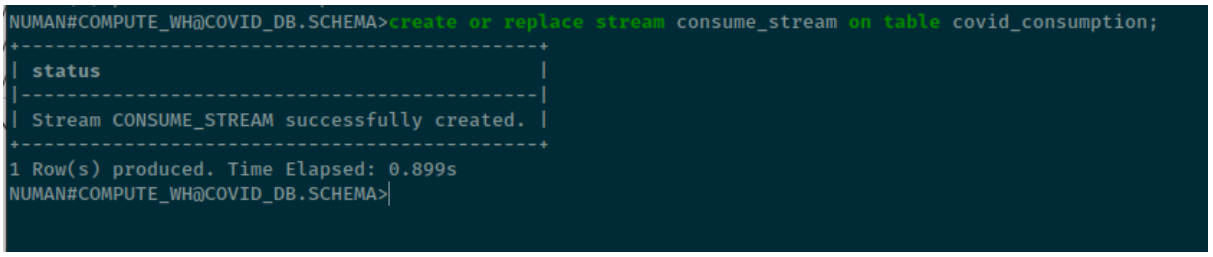
We have seen that the data has been added to the table “covid_consumption”, but in order to implement the SCD(Slowly Changing Dimensions) and to get all the history of data stored. Hence we have another table named as “covid_consumption_history” which has 3 extra columns i.e the start_date, end_date 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_FATALITY_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 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 metadata$action = 'INSERT'
                                            and metadata$isupdate = '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 metadata$action = 'INSERT'
                                            and metadata$isupdate = 'TRUE'
                                            union

                                            select null,null,null,null,last_update,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
```

```
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 metadata$action = 'INSERT'
                                            and metadata$isupdate = 'TRUE'
                                            union

                                            select null,null,null,null,last_update,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 metadata$action = 'INSERT'
                                                                                 and metadata$isupdate = 'TRUE')
                                    and current_flag = 1))
                                union


                                select null,null,null,null,cs.last_update,null,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 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 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
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>
```

Code for view:

```
create or replace view covid_change_data as
--this query figures out what to do when insert takes place
select
fips,admin2,province_state,country_region,last_update,lat,long,confirmed,de
aths,
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,de
aths,
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,de
aths,
recovered,active,combined_key,incident_rate,case_fatality_ratio,
update_timestamp
            from consume_stream
            where metadata$action = 'INSERT'
            and metadata$isupdate = 'FALSE'))
union
--this query figures out what to do when update takes place

select
fips,admin2,province_state,country_region,last_update,lat,long,confirmed,de
aths,
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,de
aths,
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,de
aths,
recovered,active,combined_key,incident_rate,case_fatality_ratio,
update_timestamp, 'I' as dml_type
            from consume_stream
            where metadata$action = 'INSERT'
            and metadata$isupdate = 'TRUE'
            union

         select null,null,null,null,last_update,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 metadata$action = 'INSERT'
                                    and metadata$isupdate = 'TRUE')
     and current_flag = 1))
union

--this query figures out what to do when delete takes place
select null,null,null,null,cs.last_update,null,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;
```

Stream is initially empty:

```
show streams;
select * from consume_stream;
```

Data Preview — Open History

ery ID | SQL | 1.16s | 0 rows

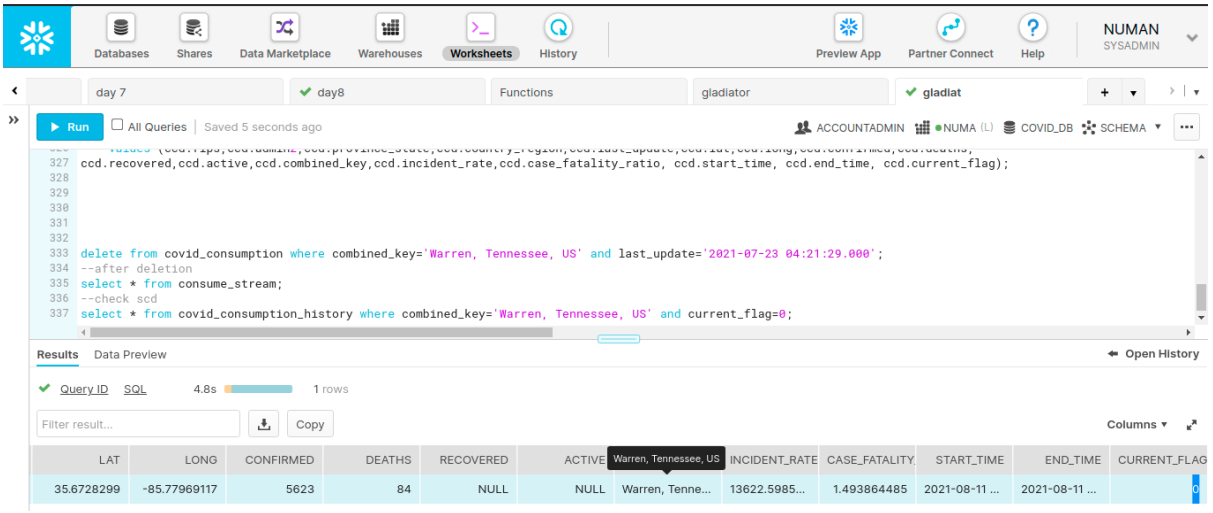
esult... | Copy | Columns

ED	DEATHS	RECOVERED	ACTIVE	COMBINED_KEY	INCIDENT_RATE	CASE_FATALITY	UPDATE_TIMES	METADATA$ACT	METADATA$ISU	METADATA$ROV

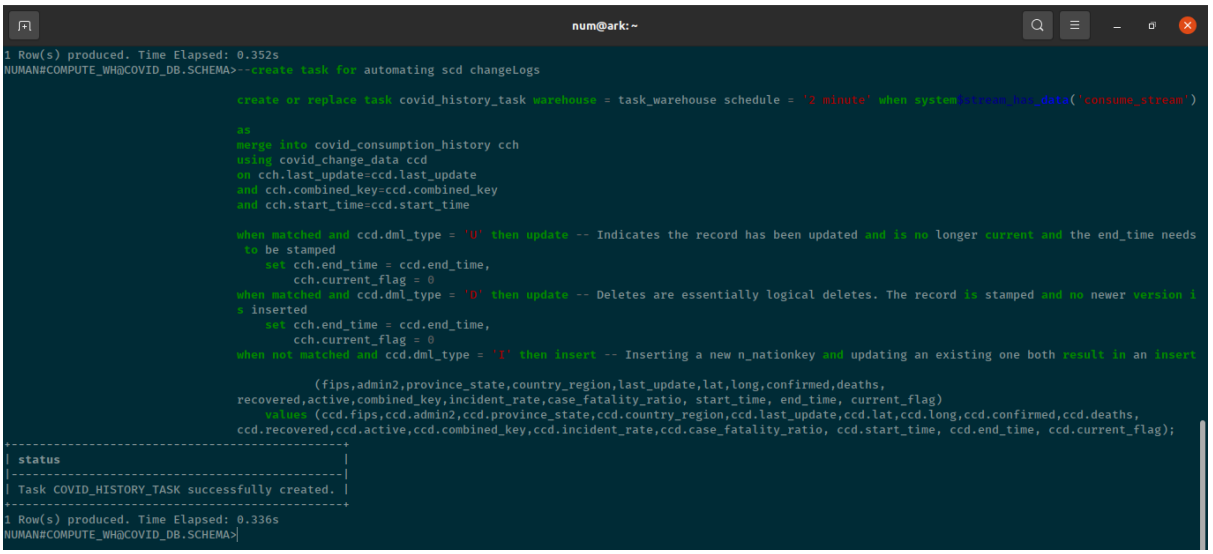
```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>desc TABLE COVID_CONSUMPTION_HISTORY;
+--------------------+------------------+--------+-------+---------+-------------+------------+-------+------------+---------+-------------+
| name               | type             | kind   | null? | default | primary key | unique 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>
```



Now to automate the merge statements 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.



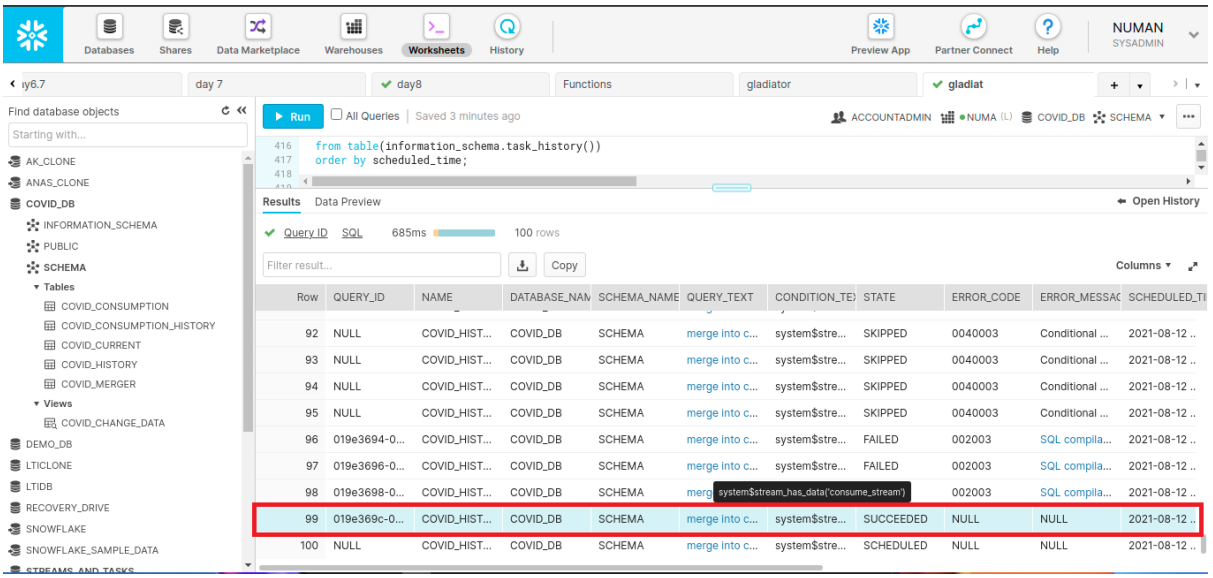
Code for Tasks:

```
--create task for automating scd changeLogs

create or replace task covid_history_task warehouse =
task_warehouse schedule = '10 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
    set cch.end_time = ccd.end_time,
        cch.current_flag = 0
when not matched and ccd.dml_type = 'I' then insert --
(fips,admin2,province_state,country_region,last_update,lat,long,co
nfirmed,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.las
t_update,ccd.lat,ccd.long,ccd.confirmed,ccd.deaths,
ccd.recovered,ccd.active,ccd.combined_key,ccd.incident_rate,ccd.ca
se_fatality_ratio, ccd.start_time, ccd.end_time,
ccd.current_flag);
```

TASK covid_history_task has been scheduled and subsequently it succeeded as shown:

```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select * from consume_stream;
+------+---------+----------------+----------------+-------------------------+-------------+-------------+-----------+--------+-----------+--------+-------------------
--+----------------+--------------------+-------------------------+
| FIPS | ADMIN2  | PROVINCE_STATE | COUNTRY_REGION | LAST_UPDATE             |         LAT |        LONG | CONFIRMED | DEATHS | RECOVERED | ACTIVE | COMBINED_KEY
  |  INCIDENT_RATE | CASE_FATALITY_RATIO | UPDATE_TIMESTAMP        | METADATA$ACTION | METADATA$ISUPDATE | METADATA$ROW_ID                          |
|------+---------+----------------+----------------+-------------------------+-------------+-------------+-----------+--------+-----------+--------+-------------------
--+----------------+--------------------+-------------------------+-----------------+-------------------+------------------------------------------|
| 1001 | Numan   | Alabama        | US             | 2020-07-09 04:34:23.000 | 32.53952745 | -86.64408227 |       663 |     12 |         0 |    631 | Autauga, Alabama, U
S | 1150.906585047 |        1.866251944 | 2021-08-12 00:15:24.285 | INSERT          | True              | db61e8a836478507c0615ffb783476d7ae47b9da |
| 1001 | Atharva | Alabama        | US             | 2020-07-12 04:34:30.000 | 32.53952745 | -86.64408227 |       708 |     14 |         0 |    662 | Autauga, Alabama, U
S | 1209.973330469 |        2.071005917 | 2021-08-12 00:15:27.659 | INSERT          | True              | 603827587e6f8a7d7665170e1eaa747fc90b20ae |
| 1001 | Garima  | Alabama        | US             | 2020-07-21 04:38:46.000 | 32.53952745 | -86.64408227 |       872 |     20 |         0 |    818 | Autauga, Alabama, U
S | 1499.937353452 |        2.386634845 | 2021-08-12 00:15:31.054 | INSERT          | True              | 594d913850a6643095a9d3e2b979dd16085fd2d0 |
| 1001 | Mukesh  | Alabama        | US             | 2020-07-31 04:35:18.000 | 32.53952745 | -86.64408227 |      1029 |     20 |         0 |    943 | Autauga, Alabama, U
S | 1723.675025506 |        2.076843198 | 2021-08-12 00:15:32.269 | INSERT          | True              | d76f6b7a5838153c662b85714798887095b96e11 |
| 1001 | Autauga | Alabama        | US             | 2020-07-09 04:34:23.000 | 32.53952745 | -86.64408227 |       663 |     12 |         0 |    631 | Autauga, Alabama, U
S | 1150.906585047 |        1.866251944 | 2021-08-11 05:54:29.469 | DELETE          | True              | db61e8a836478507c0615ffb783476d7ae47b9da |
| 1001 | Autauga | Alabama        | US             | 2020-07-12 04:34:30.000 | 32.53952745 | -86.64408227 |       708 |     14 |         0 |    662 | Autauga, Alabama, U
S | 1209.973330469 |        2.071005917 | 2021-08-11 05:54:29.469 | DELETE          | True              | 603827587e6f8a7d7665170e1eaa747fc90b20ae |
| 1001 | Autauga | Alabama        | US             | 2020-07-21 04:38:46.000 | 32.53952745 | -86.64408227 |       872 |     20 |         0 |    818 | Autauga, Alabama, U
S | 1499.937353452 |        2.386634845 | 2021-08-11 05:54:29.469 | DELETE          | True              | 594d913850a6643095a9d3e2b979dd16085fd2d0 |
| 1001 | Autauga | Alabama        | US             | 2020-07-31 04:35:18.000 | 32.53952745 | -86.64408227 |      1029 |     20 |         0 |    943 | Autauga, Alabama, U
S | 1723.675025506 |        2.076843198 | 2021-08-11 05:54:29.469 | DELETE          | True              | d76f6b7a5838153c662b85714798887095b96e11 |
+------+---------+----------------+----------------+-------------------------+-------------+-------------+-----------+--------+-----------+--------+-------------------
--+----------------+--------------------+-------------------------+-----------------+-------------------+------------------------------------------+
8 Row(s) produced. Time Elapsed: 1.234s
```

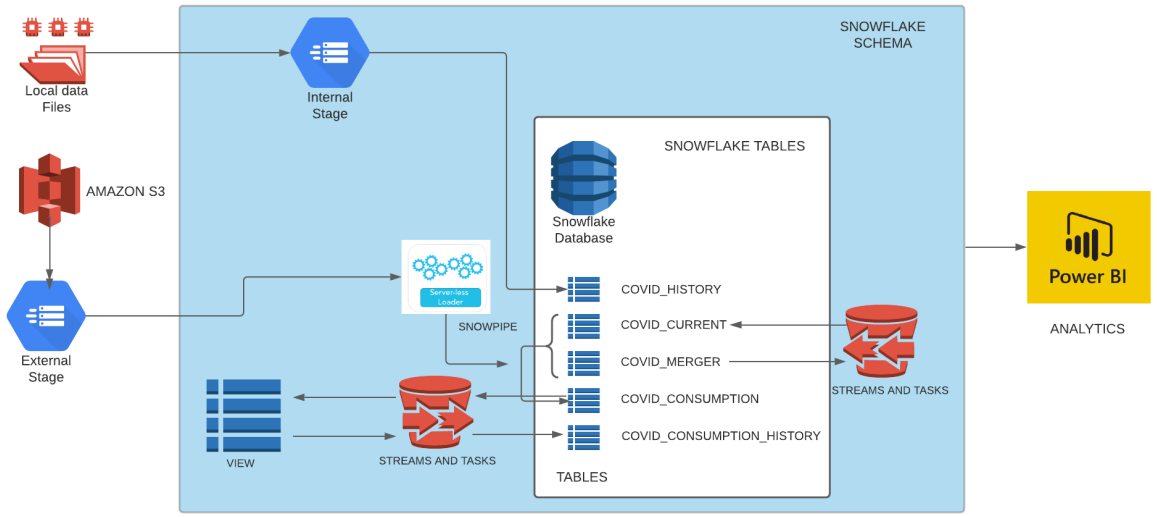


End Result:

```
num@ark: ~
+------+---------+-------------------------+-----------+--------------+
8 Row(s) produced. Time Elapsed: 2.223s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select fips,admin2,last_update,confirmed from covid_consumption where FIPS = 1001 and confirmed=663
                                union
                                select fips,admin2,last_update,confirmed from covid_consumption where FIPS = 1001 and confirmed=708
                                union
                                select fips,admin2,last_update,confirmed from covid_consumption where FIPS = 1001 and confirmed=872
                                union
                                select fips,admin2,last_update,confirmed from covid_consumption where FIPS = 1001 and confirmed=1029;
+------+---------+-------------------------+-----------+
| FIPS | ADMIN2  | LAST_UPDATE             | CONFIRMED |
|------+---------+-------------------------+-----------|
| 1001 | Mukesh  | 2020-07-31 04:35:18.000 |      1029 |
| 1001 | Numan   | 2020-07-09 04:34:23.000 |       663 |
| 1001 | Atharva | 2020-07-12 04:34:30.000 |       708 |
| 1001 | Garima  | 2020-07-21 04:38:46.000 |       872 |
+------+---------+-------------------------+-----------+
4 Row(s) produced. Time Elapsed: 1.170s
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>select fips,admin2,last_update,confirmed,current_flag from covid_consumption_history where FIPS = 1001 and confirmed=663
                                union
                                select fips,admin2,last_update,confirmed,current_flag from covid_consumption_history where FIPS = 1001 and confirmed=708
                                union
                                select fips,admin2,last_update,confirmed,current_flag from covid_consumption_history where FIPS = 1001 and confirmed=872
                                union
                                select fips,admin2,last_update,confirmed,current_flag from covid_consumption_history where FIPS = 1001 and confirmed=1029;
+------+---------+-------------------------+-----------+--------------+
| FIPS | ADMIN2  | LAST_UPDATE             | CONFIRMED | CURRENT_FLAG |
|------+---------+-------------------------+-----------+--------------|
| 1001 | Atharva | 2020-07-12 04:34:30.000 |       708 |            1 |
| 1001 | Numan   | 2020-07-09 04:34:23.000 |       663 |            1 |
| 1001 | Garima  | 2020-07-21 04:38:46.000 |       872 |            1 |
| 1001 | Mukesh  | 2020-07-31 04:35:18.000 |      1029 |            1 |
| 1001 | Autauga | 2020-07-31 04:35:18.000 |      1029 |            0 |
| 1001 | Autauga | 2020-07-12 04:34:30.000 |       708 |            0 |
| 1001 | Autauga | 2020-07-21 04:38:46.000 |       872 |            0 |
| 1001 | Autauga | 2020-07-09 04:34:23.000 |       663 |            0 |
+------+---------+-------------------------+-----------+--------------+
8 Row(s) produced. Time Elapsed: 0.306s
```

Here the results indicate that we have successfully implemented the Slowly changing dimension of Type-2.

The following image shows the clear view of data flow and scds working in production:



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>
```

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
```

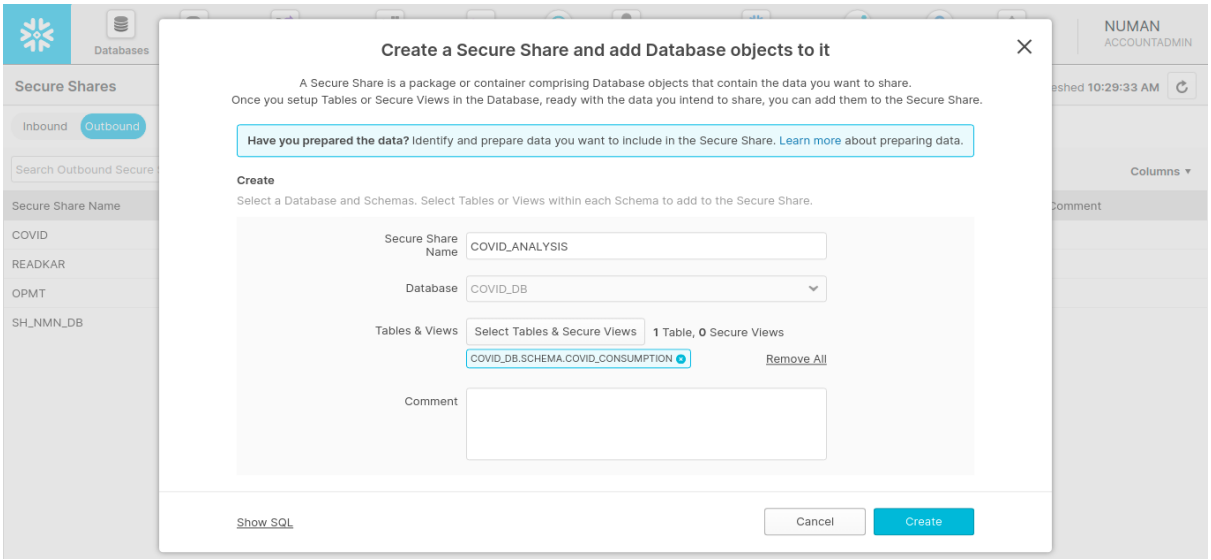
After this we create a clone table with the data from the covid_consumption table before this update was done in order to demonstrate **Time Travel**.

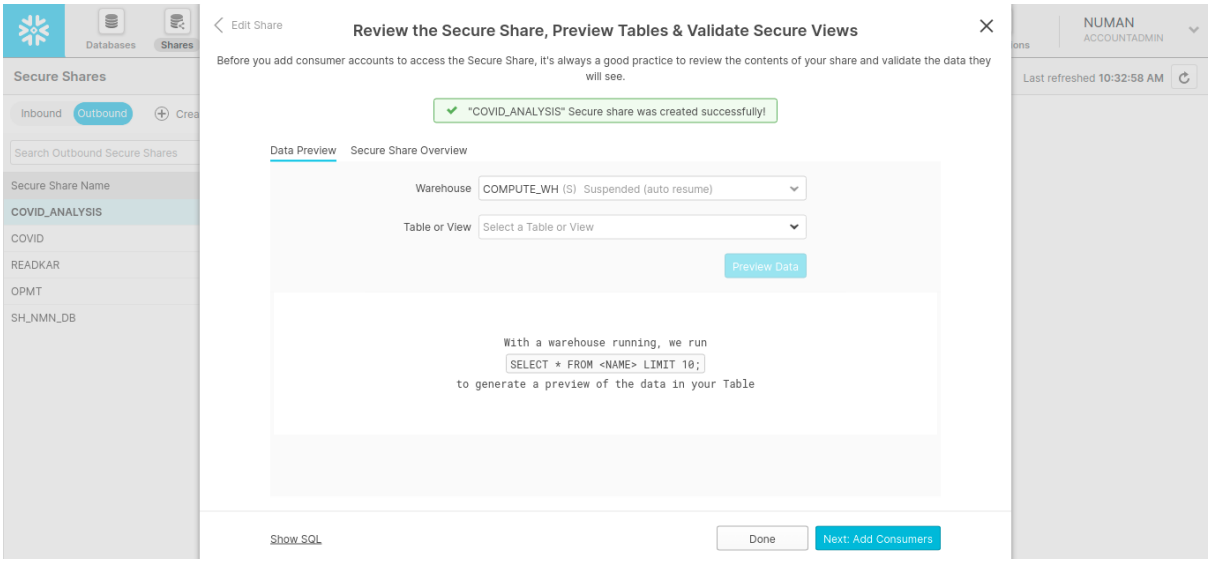
```
NUMAN#COMPUTE_WH@COVID_DB.SCHEMA>CREATE TABLE "COVID_DB"."SCHEMA".COVID_CLONE_TT CLONE "COVID_DB"."SCHEMA"."COVID_CONSUMPTION"
                                before(TIMESTAMP=>'2021-08-12 00:15:00.000 +0530' :: timestamp_tz);
+-----------------------------------------+
| status                                  |
|-----------------------------------------|
| 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>
```

Creation of Secured share and Reader Account -

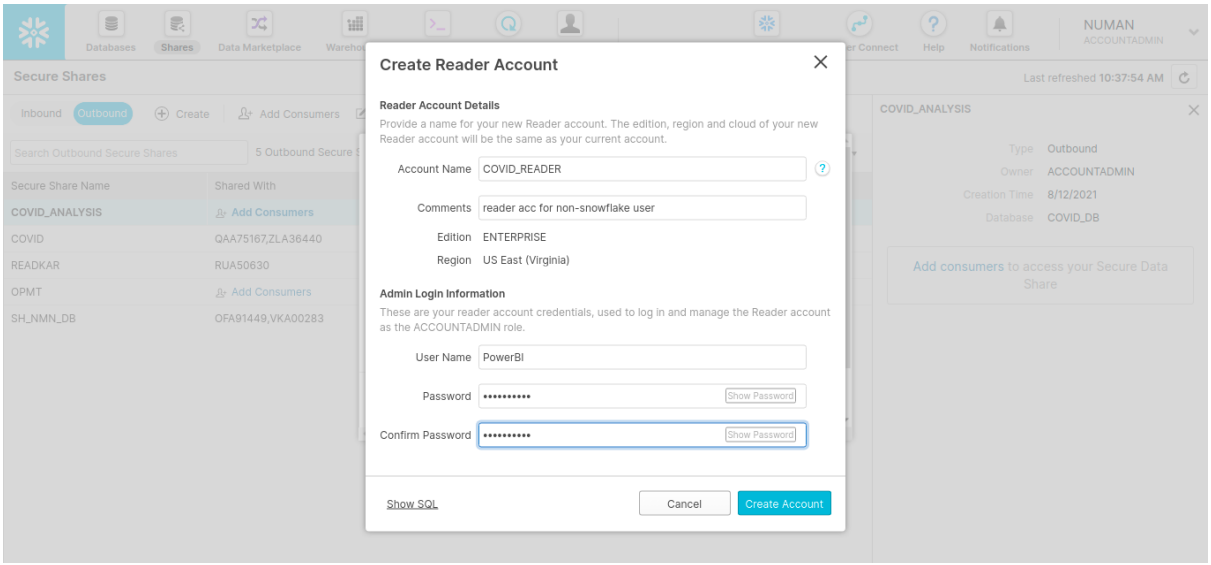
To share our database for the non-Snowflake user in order to use the data for various analytics, we create a secure share for the table which has the appropriate data. In secure share, the reader cannot perform any DML operations on the shared table reader can only see the data, that is, in **Read Only**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

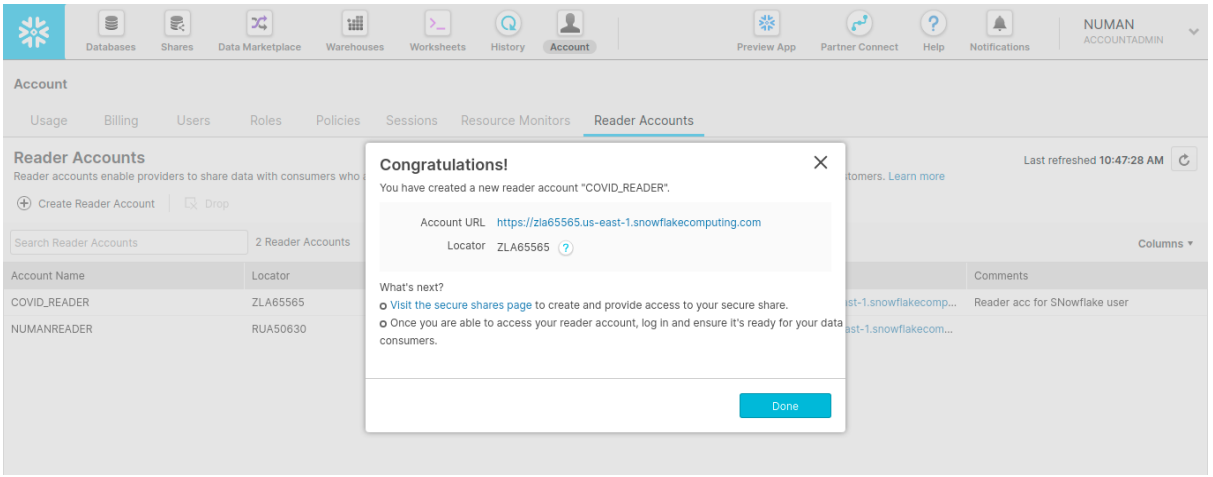
CREATE SHARE "COVID_ANALYSIS" COMMENT='';
GRANT USAGE ON DATABASE "COVID_DB" TO SHARE "COVID_ANALYSIS";
GRANT USAGE ON SCHEMA "COVID_DB"."SCHEMA" TO SHARE "COVID_ANALYSIS";
GRANT SELECT ON VIEW "COVID_DB"."SCHEMA"."COVID_CONSUMPTION" TO SHARE "COVID_ANALYSIS"

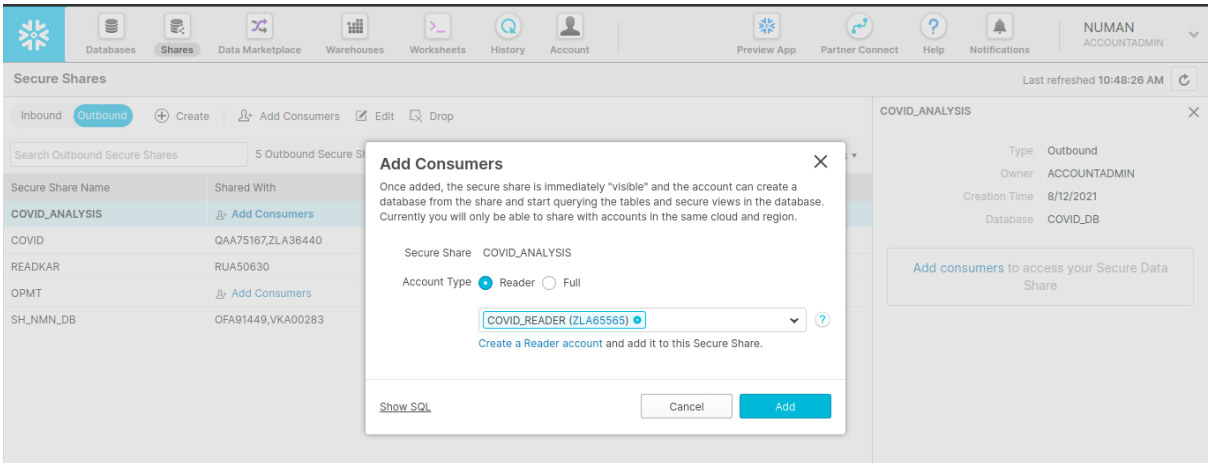




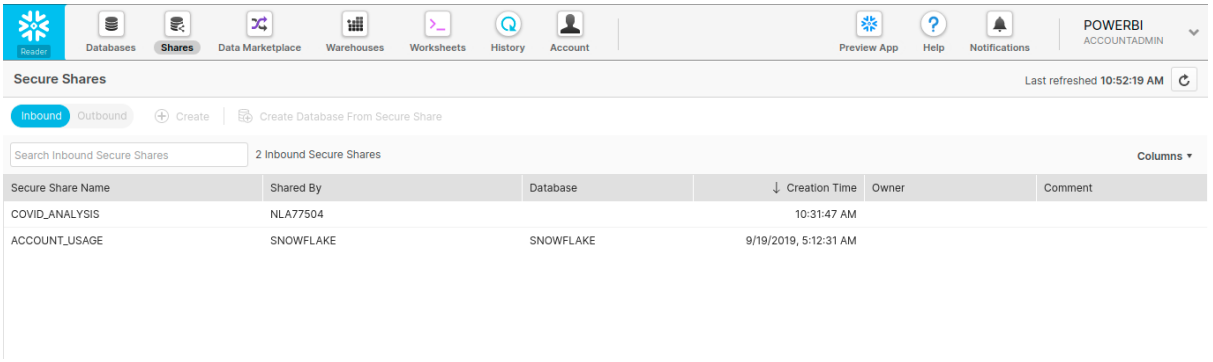
After creating the secure share, we now create a reader account for the non snowflake user. For that we mention the name of the account and all the details like the username and the password for the account.

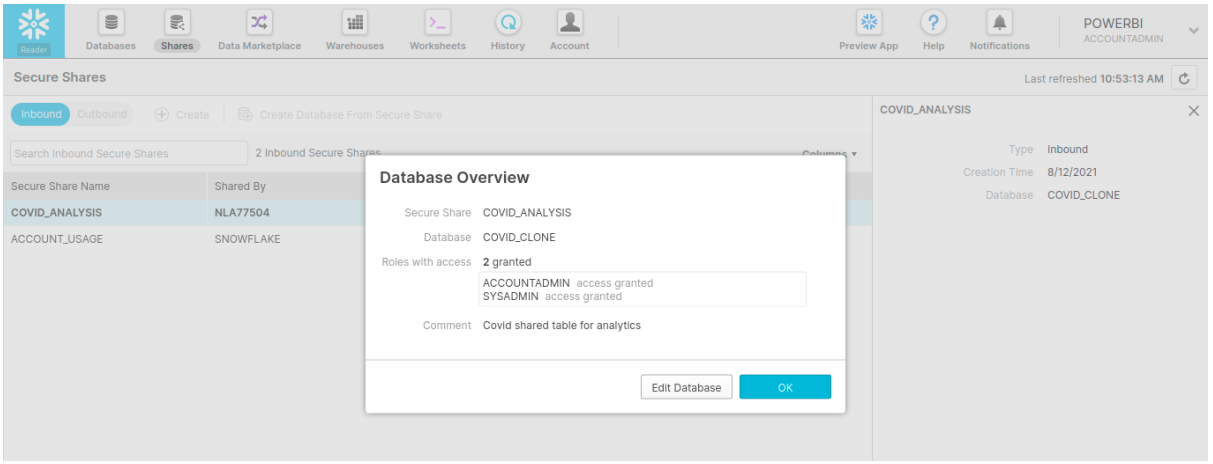


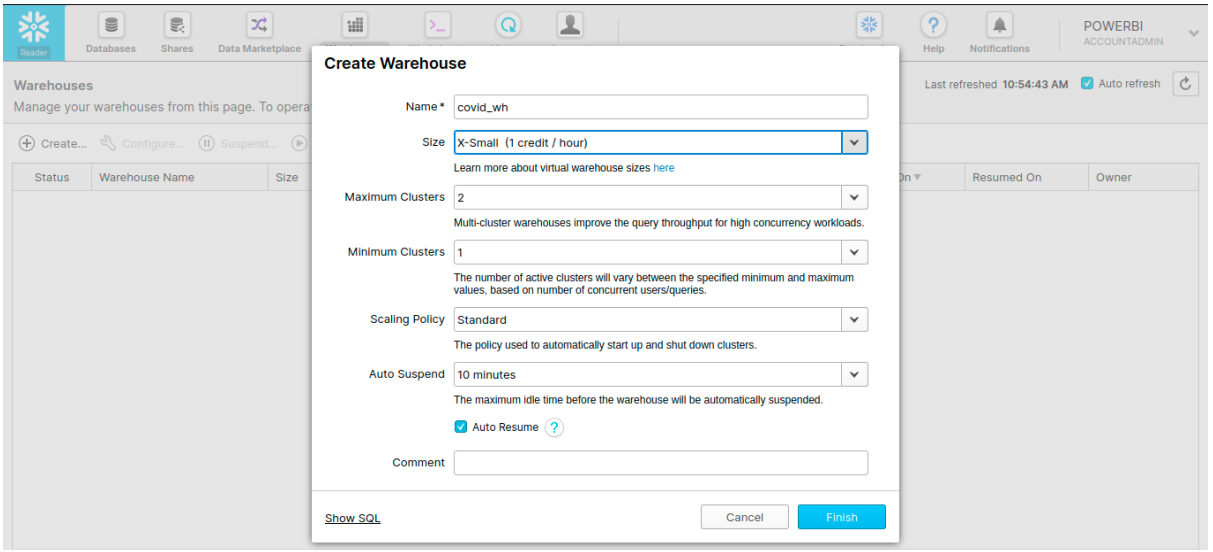


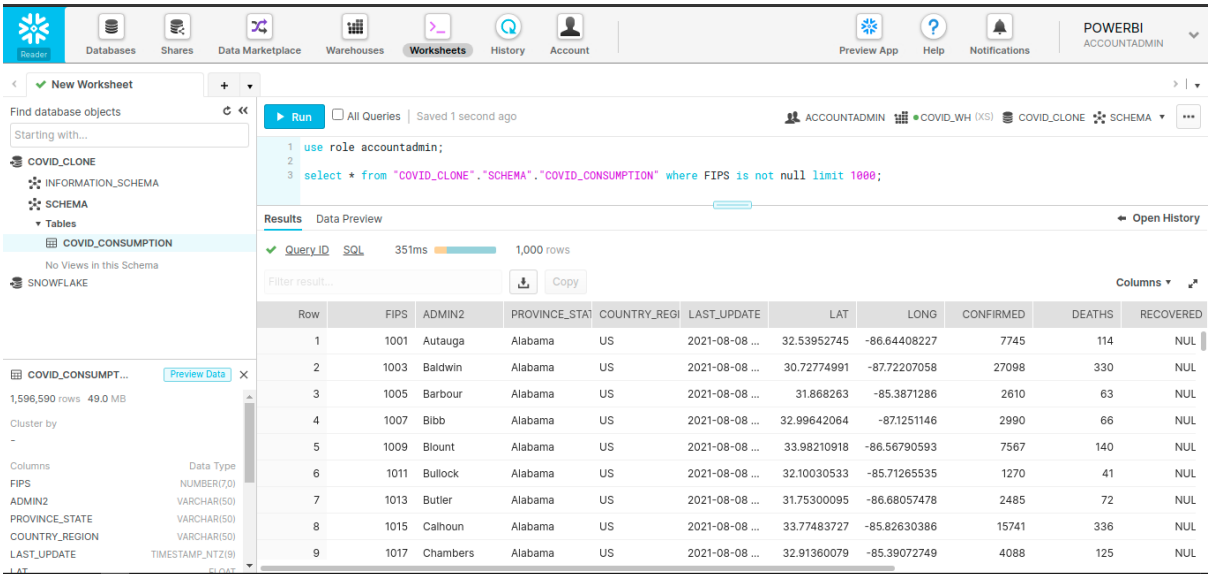


After login in the Reader account, we change the role to ACCOUNTADMIN in order to create a shared database for the shared table to be used. Also we create a warehouse for the reader account, that can be used to run the queries to retrieve data from the shared table.




Secure Shares
Database Overview
Secure Share COVID_ANALYSIS
Database COVID_CLONE
Roles with access 2 granted
ACCOUNTADMIN access granted
SYSADMIN access granted
Comment Covid shared table for analytics
Edit Database
OK
COVID_ANALYSIS
Type Inbound
Creation Time 8/12/2021
Database COVID_CLONE

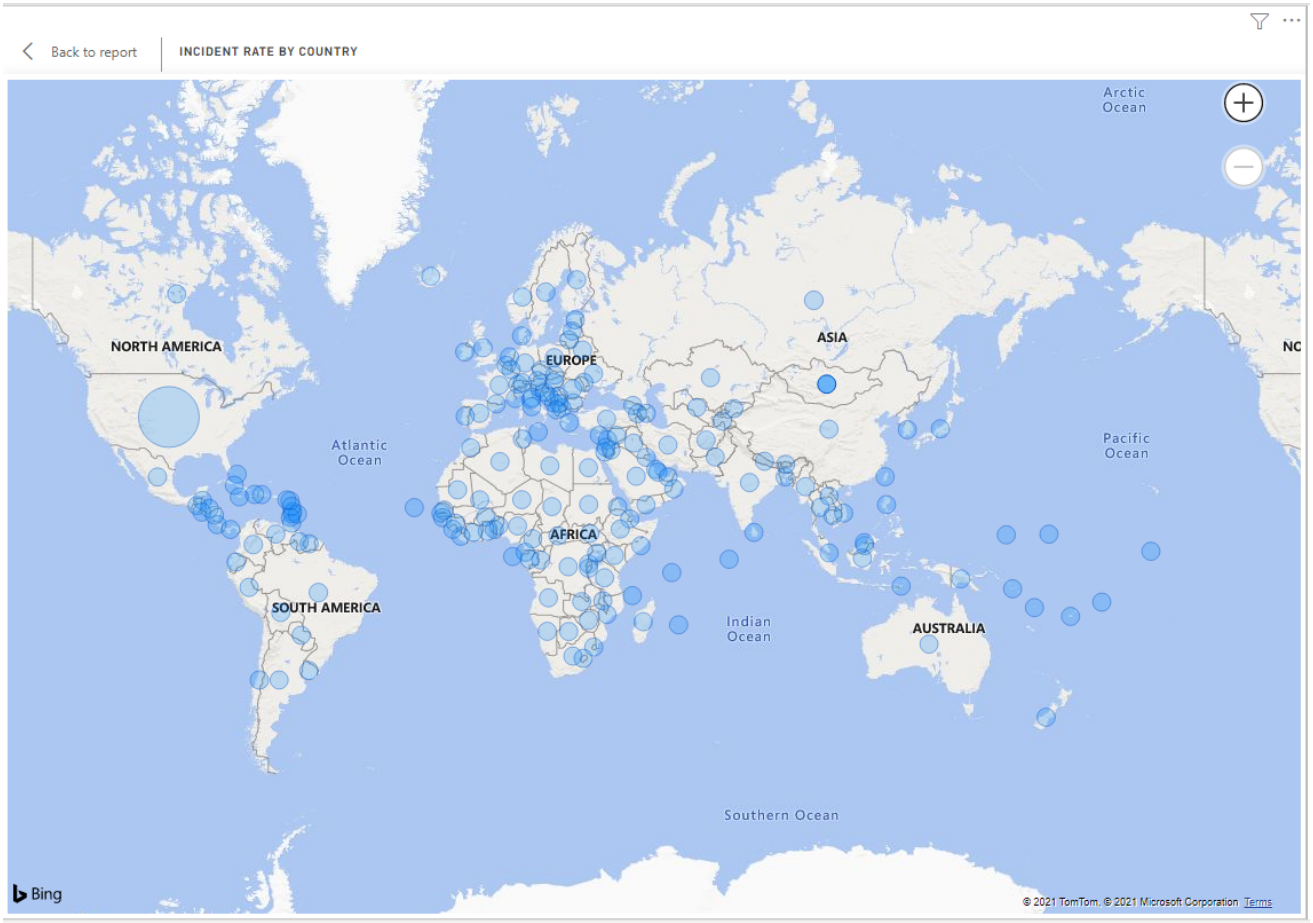

Warehouses
Create Warehouse
Name * covid_wh
Size X-Small (1 credit / hour)
Learn more about virtual warehouse sizes here
Maximum Clusters 2
Multi-cluster warehouses improve the query throughput for high concurrency workloads.
Minimum Clusters 1
The number of active clusters will vary between the specified 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


New Worksheet
Run
use role accountadmin;
select * from "COVID_CLONE"."SCHEMA"."COVID_CONSUMPTION" where FIPS is not null limit 1000;
Results
Data Preview
Open History
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 ... 30.72774991 -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

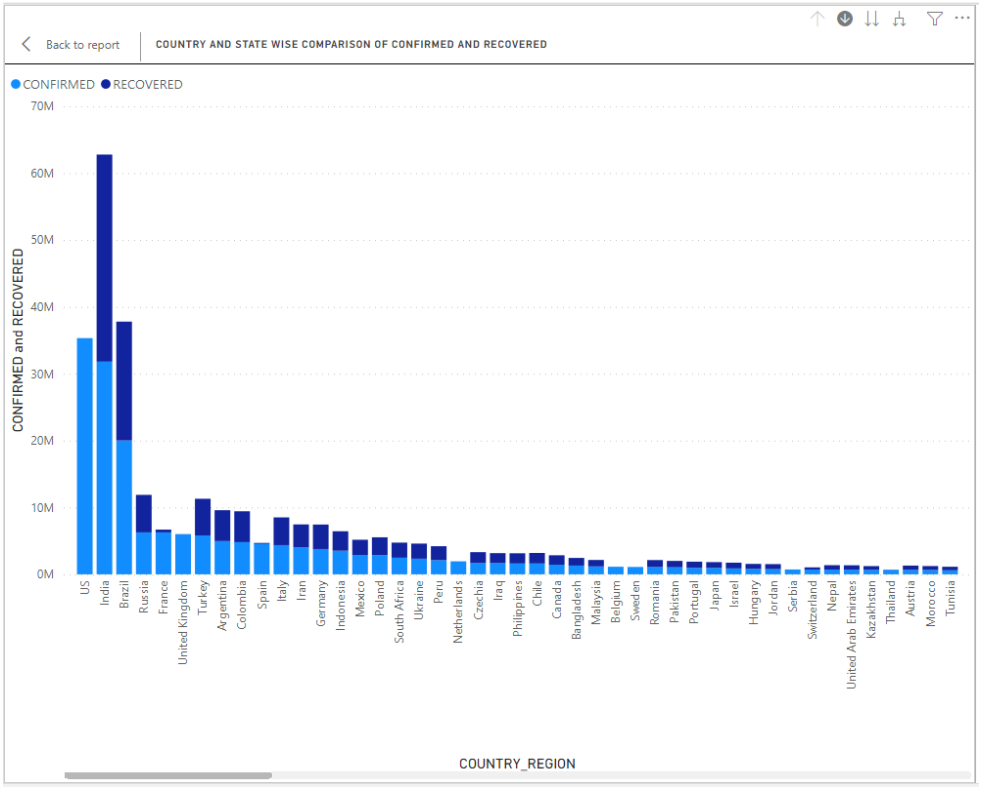
REPORTS

INCIDENT RATE BY COUNTRY REGION

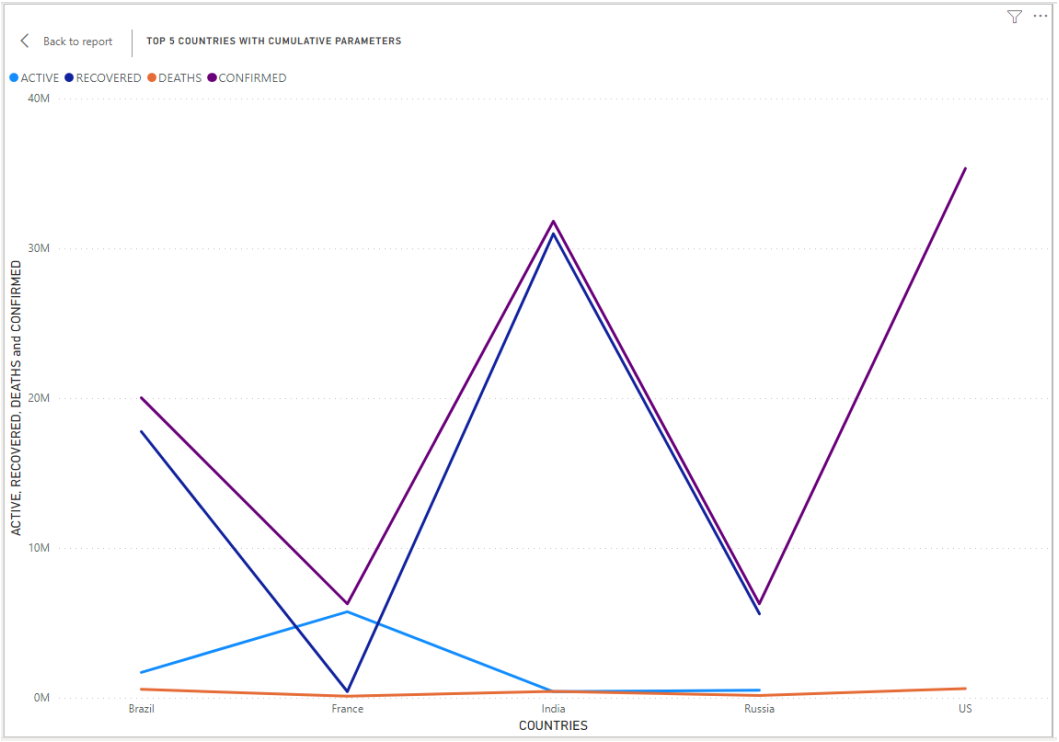
The **incidence rate** is a measure of the frequency with which the event, in this case COVID-19, occurs over a specific period. Numerically, it is defined as the number of new cases for the disease within a time frame, as a proportion of the number of people at risk for the disease.



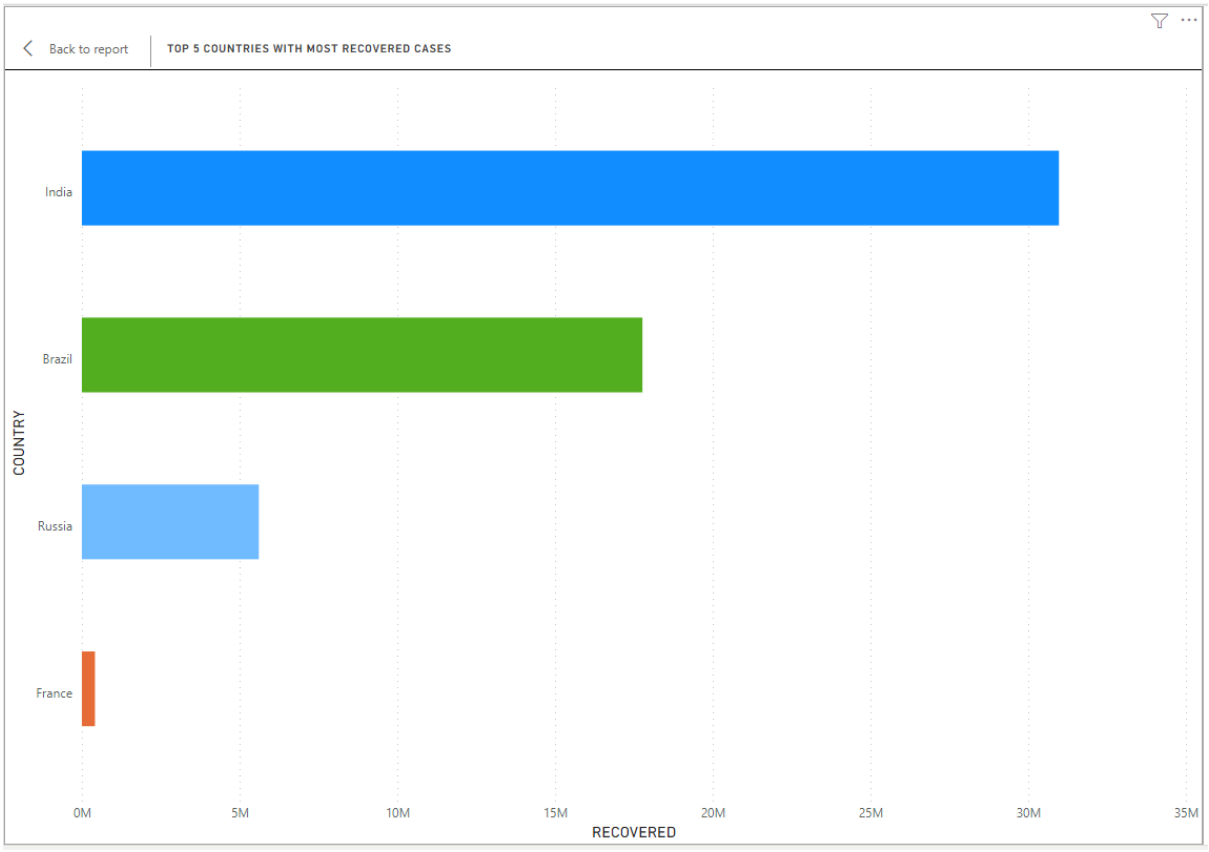
COUNTRY WISE COMPARISON OF CONFIRMED AND RECOVERED CASES



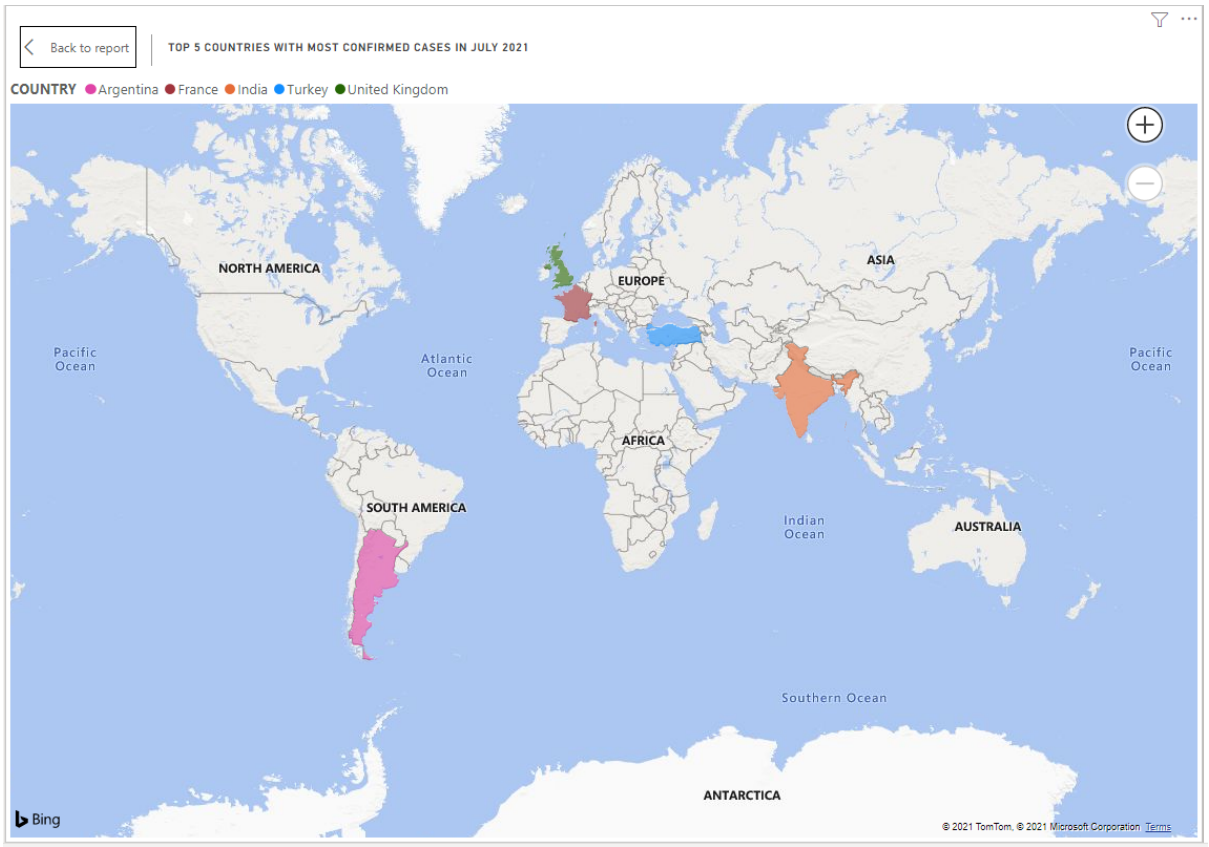
TOP 5 COUNTRIES WITH CUMULATIVE PARAMETERS



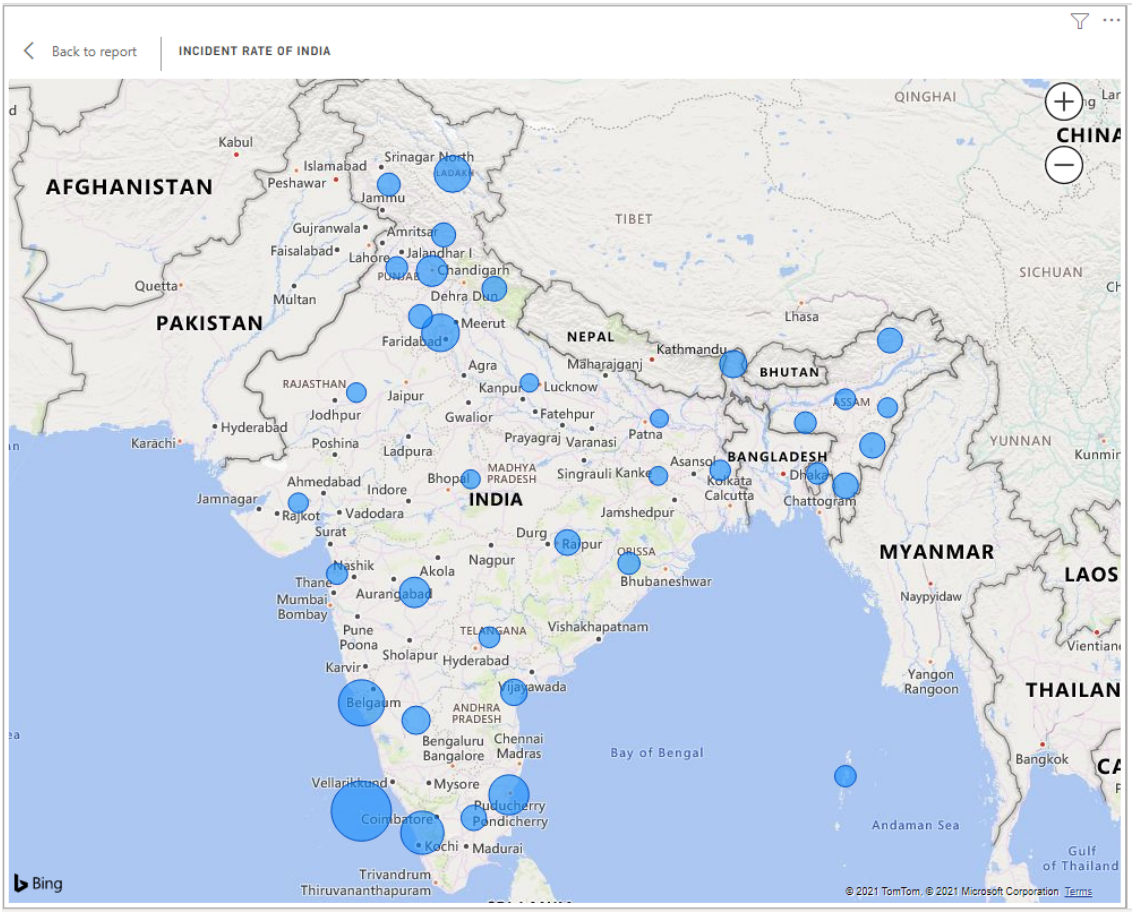
TOP 5 COUNTRIES WITH MOST RECOVERED CASES



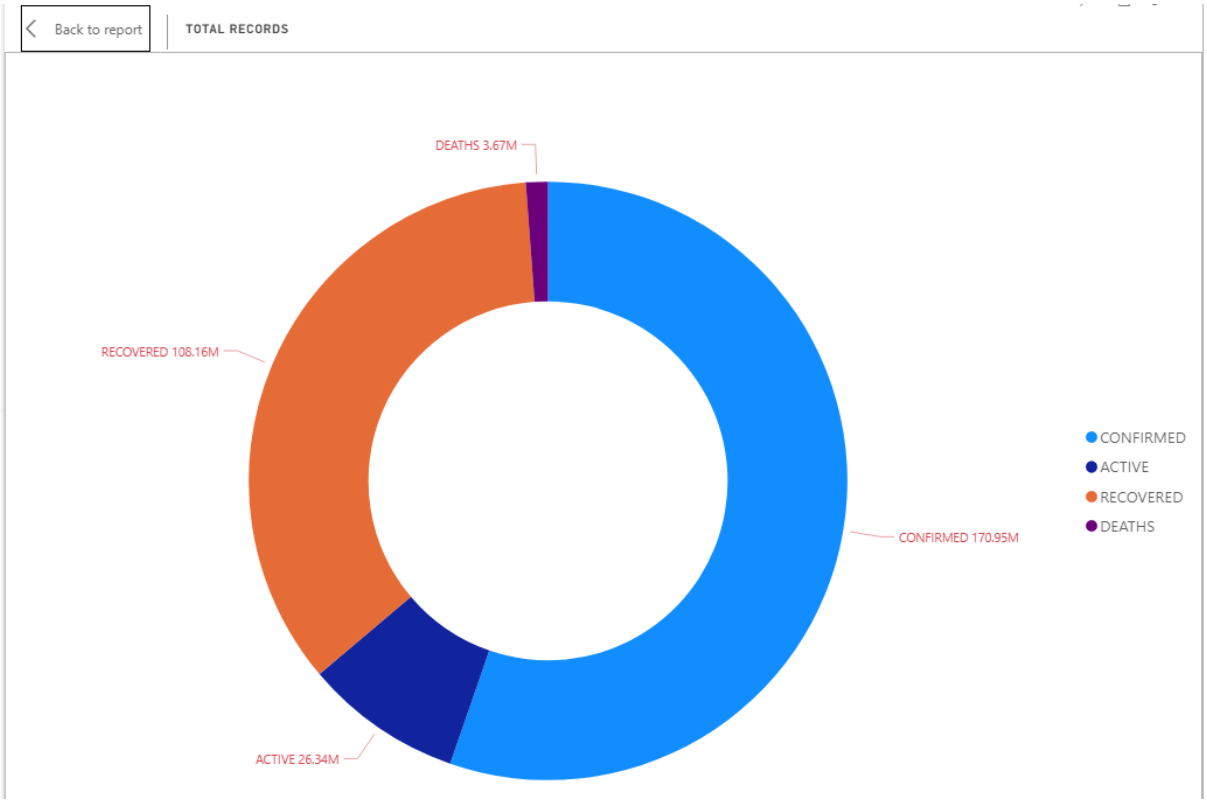
TOP 5 COUNTRIES WITH MOST CONFIRMED CASES IN JULY 2021



INCIDENT RATE OF INDIA BASED ON STATES



TOTAL RECORD OF INDIA WITH CUMULATIVE PARAMETERS

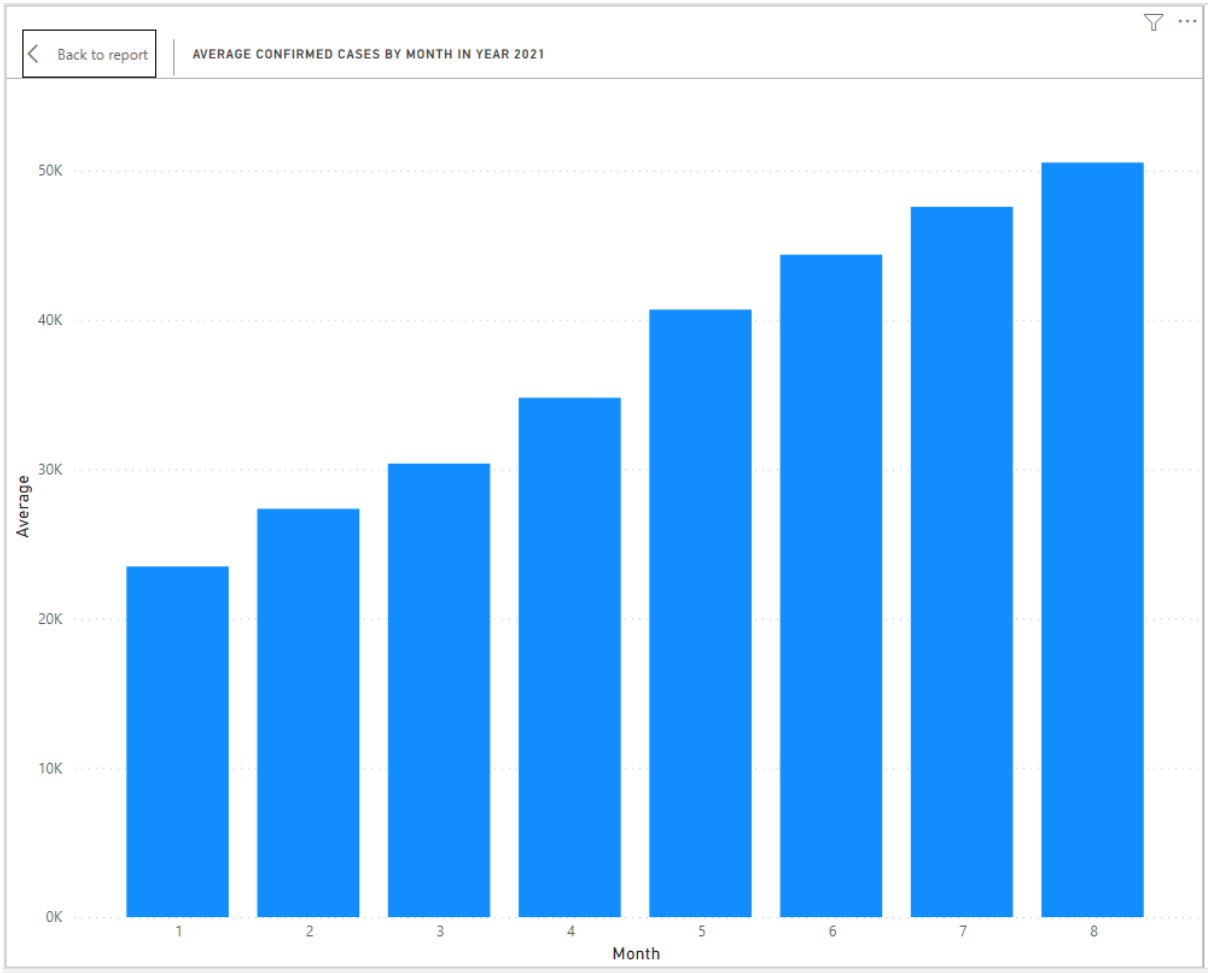


TOP 10 STATES OF INDIA WITH MOST NUMBER OF CONFIRMED CASES

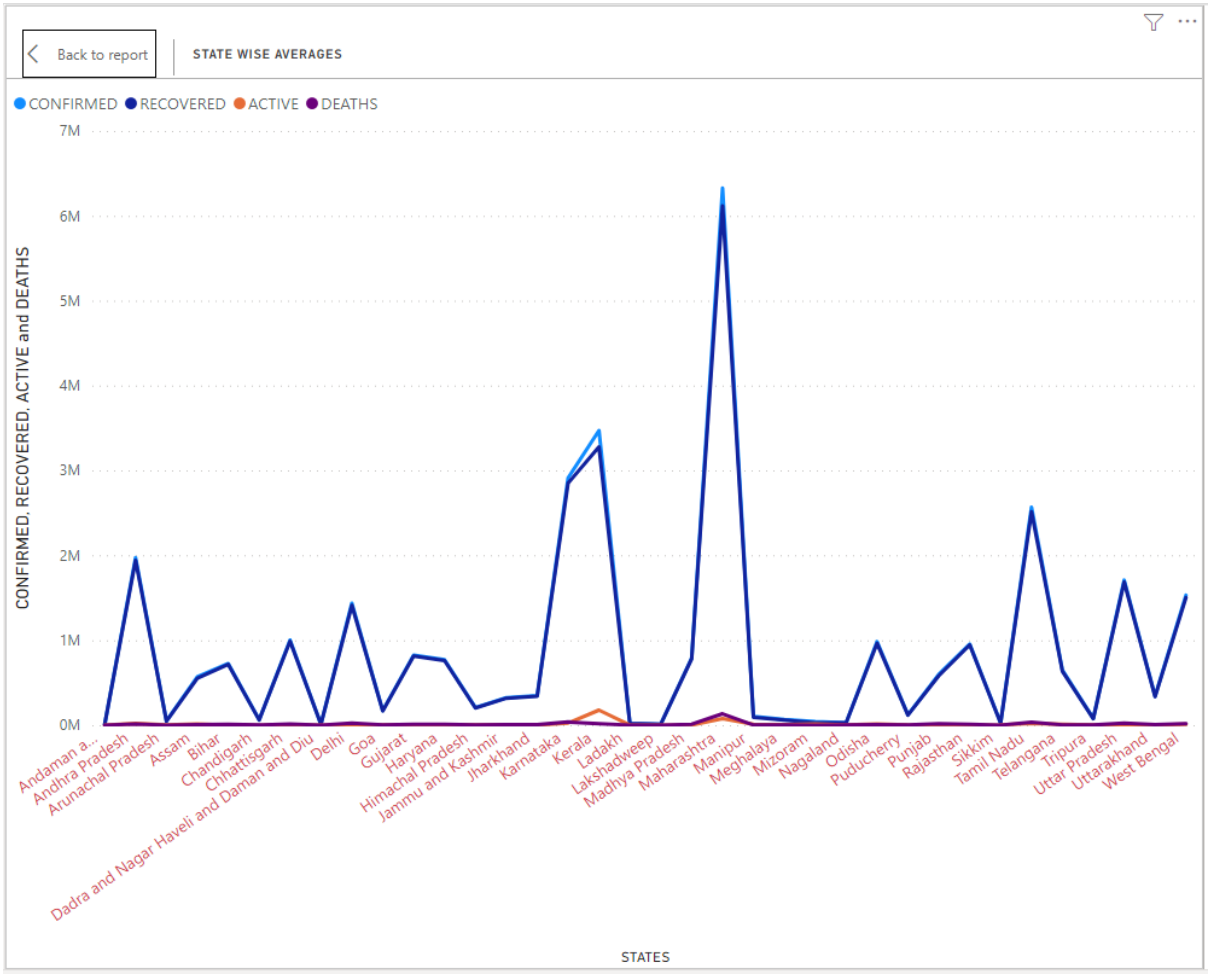
TOP 5 STATES WITH MOST NUMBER OF CASES

STATE	CONFIRMED	RECOVERED	ACTIVE	DEATHS
Maharashtra	5,746,892.00	5,395,370.00	256,178.00	95,344.00
Karnataka	2,604,431.00	2,261,590.00	313,751.00	29,090.00
Delhi	1,426,240.00	1,390,963.00	11,040.00	24,237.00
Tamil Nadu	2,096,516.00	1,770,503.00	301,781.00	24,232.00
Uttar Pradesh	1,691,488.00	1,633,947.00	37,044.00	20,497.00
West Bengal	1,376,377.00	1,273,788.00	87,048.00	15,541.00
Chhattisgarh	971,463.00	922,674.00	35,741.00	13,048.00
Andhra Pradesh	1,693,085.00	1,528,360.00	153,795.00	10,930.00
Kerala	2,526,579.00	2,310,385.00	207,379.00	8,815.00
Rajasthan	939,958.00	888,919.00	42,654.00	8,385.00

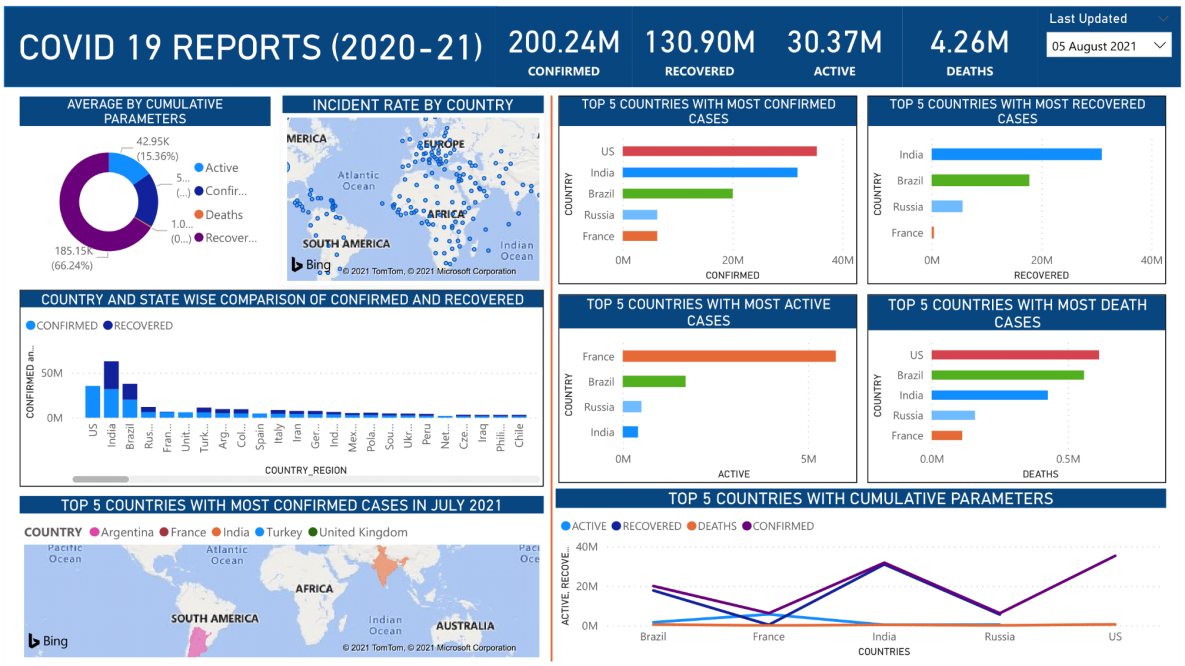
AVERAGE CONFIRMED CASES BY MONTH IN YEAR 2021



STATE WISE AVERAGES WITH CUMULATIVE PARAMETERS



DASHBOARD - 1 (GLOBAL)

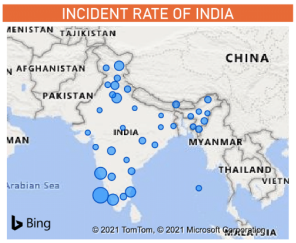


DASHBOARD - 2 (INDIA)

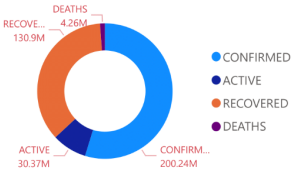
COVID 19 REPORTS (2020-21) - INDIA

Last Updated
05 August 2021

INCIDENT RATE OF INDIA



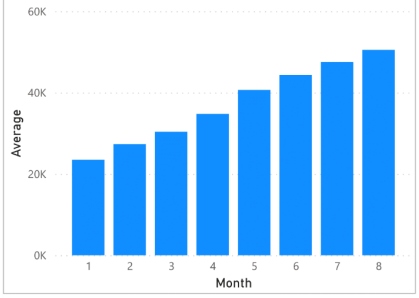
TOTAL RECORDS



TOP 5 STATES WITH MOST NUMBER OF CASES

STATE	CONFIRMED	RECOVERED	ACTIVE	DEATHS
Maharashtra	63,27,194.00	61,17,560.00	76,224.00	1,33,410.00
Karnataka	29,11,727.00	28,50,717.00	24,330.00	36,680.00
Tamil Nadu	25,67,401.00	25,13,087.00	20,117.00	34,197.00
Delhi	14,36,518.00	14,10,947.00	513.00	25,058.00
Uttar Pradesh	17,08,623.00	16,85,170.00	686.00	22,767.00
West Bengal	15,30,850.00	15,01,925.00	10,745.00	18,180.00
Kerala	34,71,563.00	32,77,788.00	1,76,564.00	17,211.00
Chhattisgarh	10,02,735.00	9,87,298.00	1,906.00	13,531.00
Andhra Pradesh	19,73,996.00	19,40,368.00	20,184.00	13,444.00
Odisha	9,82,181.00	9,63,718.00	12,295.00	6,168.00

AVERAGE CONFIRMED CASES BY MONTH IN YEAR 2021



STATE WISE AVERAGES