CSSE Covid19 Daily Reports - Healthcare Case Study

JIVIN VARGHESE PORTHUKARAN

Problem Statement:

Recent Covid-19 Pandemic has raised alarms over one of the most overlooked areas to focus: Healthcare Management. Center for Systems Science and

Engineering (CSSE) at Johns Hopkins University has a huge repository of data of COVID-19 for dashboardsThe 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.As a snowflake developer we can help non-technical people to understand

this huge dataset in very simple and efficient way.This can be done by snowflake by loading, exploring, cleaning, merging using pipes, streams. End result is to get some meaningful insights from this huge data and also answer some business questions.

Dataset

Source data is downloaded as a CSV file (comma-separated values) from the CSSE github repository and the metadata for the same is as follows:

Historical Data(2nd Jan 2021 - 1st May 2021):

Rows	Columns
480,363	14

Current Data(2nd May 2021 - 1st Jun 2021):

Rows	Columns
124,434	14

Column Name	Description	Data Type
FIPS	US only. Federal Information Processing Standards code that uniquely identifies counties within the USA.	Number
Admin2	County name. US only.	Plain Text

Province_State	Province, state or dependency name.	Plain Text
Country_Region	Country , region or sovereignty name. The names of locations included on the Website correspond with the official designations used by the U.S. Department of State.	Plain Text
Last Update	MM/DD/YYYY HH:mm:ss (24 hour format, in UTC).	Date
Lat and Long	Dot locations on the dashboard. All points (except for Australia) shown on the map are based on geographic centroids, and are not representative of a specific address, building or any location at a spatial scale finer than a province/state. Australian dots are located at the centroid of the largest city in each state.	Float
Confirmed	Counts include confirmed and probable (where reported).	Number
Deaths	Counts include confirmed and probable (where reported).	Number
Recovered	Recovered cases are estimates based on local media reports, and state and local reporting when available, and therefore may be substantially lower than the true number. US state-level recovered cases are from COVID Tracking Project.	Number

Active	Active cases = total cases - total recovered - total deaths.	Number
Incident_Rate	Incident_Rate	Number
Combined_Key	Clubbed values of Admin2, Province_State & Country_Region	Plain Text
Case_Fatality_Ratio (%)	Incidence Rate = cases per 100,000 persons.	Number

Project development Steps

1. Create database Covid_db and schema covid_ schema.

a. Database

```
//Database Creation
create database 'covid_db';
```

b. Schema

```
//Schema Creation
create schema 'covid_db'.'covid_schema';
```

2. Creating File Format for the CSV Data

```
//File Format Craetion
CREATE FILE FORMAT "COVID_DB"."COVID_SCHEMA".CSV_FF 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');
```

3. Creating Covid History Table

a. Table Creation

A table named '**COVID_HISTORY**' has been created into which the historical dataset is loaded.

```
create or replace table covid_history (
    FIPS                number          ,
    Admin2              varchar(500)    ,
    Province_State      varchar(500)    ,
    Country_Region      varchar(500)    not null,
    Last_Update         date            ,
    Lat                 double precision        ,
    Long                double precision        ,
    Confirmed           number          ,
    Deaths              number          ,
    Recovered           number          ,
    Active              number          ,
   Combined_Key         varchar(500)    ,
   Incident_Rate        number          ,
   Case_Fatality_Ratio  number
);
```

b. Description of Covid_History Table

```
//Describe Table
desc table covid_history;
```

Results | Data Preview | Open History

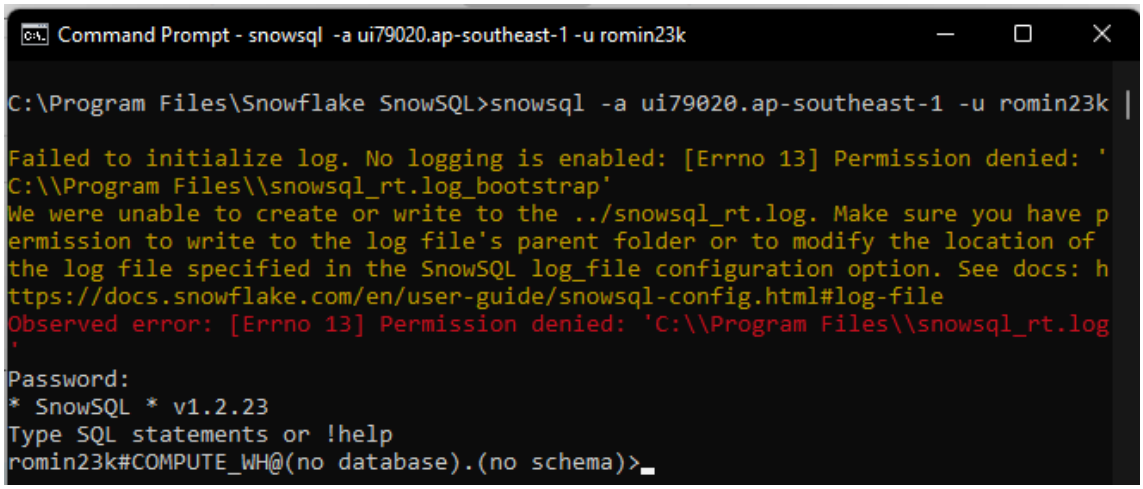
Query ID | SQL | 47ms | 14 rows

Filter result... | Copy | Columns

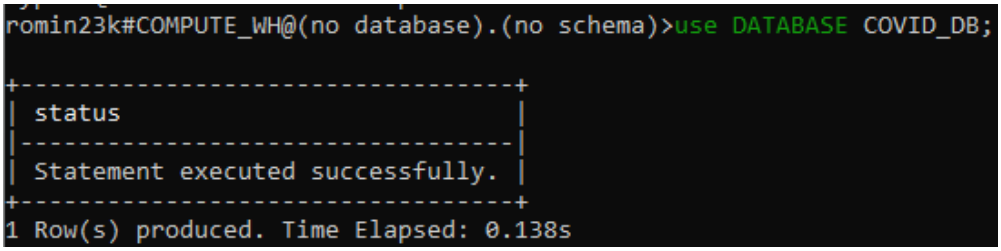
Row	name	type	kind	null?	default	primary key	unique key	check	expression	comment	policy name
1	FIPS	NUMBER(38,0)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
2	CITY	VARCHAR(500)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
3	PROVINCE_STATE	VARCHAR(500)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
4	COUNTRY_REGION	VARCHAR(500)	COLUMN	N	NULL	N	N	NULL	NULL	NULL	NULL
5	LAST_UPDATE	DATE	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
6	LAT	FLOAT	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
7	LONG	FLOAT	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
8	CONFIRMED	NUMBER(38,0)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
9	DEATHS	NUMBER(38,0)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
10	RECOVERED	NUMBER(38,0)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
11	ACTIVE	NUMBER(38,0)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
12	COMBINED_KEY	VARCHAR(500)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
13	INCIDENT_RATE	NUMBER(38,0)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL
14	CASE_FATALITY_RA...	NUMBER(38,0)	COLUMN	Y	NULL	N	N	NULL	NULL	NULL	NULL

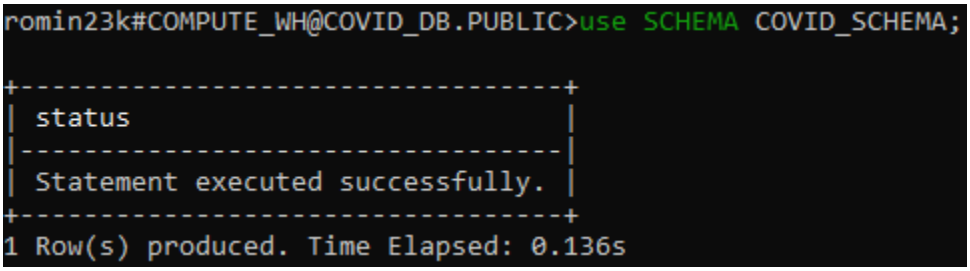
4. Create an internal stage and load the historical data from local.

a. SnowCLI Setup: Login into snowcli using snowflake account locator and snowflake credentials.



b. **Setting Context for SnowCLI:**





c. **Putting Covid Historical data into Internal Stage:** The PUT command has been used to load the data from local storage to the internal stage.

```
romin23k#COMPUTE_WH@COVID_DB.COVID_SCHEMA>put file://C:\Users\romin\Desktop\LTI\Gladiator\jan-march21\*.csv @~;
+---------------+------------------+-------------+-------------+--------------------+--------------------+----------+---------+
| source        | target           | source_size | target_size | source_compression | target_compression | status   | message |
|---------------+------------------+-------------+-------------+--------------------+--------------------+----------+---------|
| 01-01-2021.csv | 01-01-2021.csv.gz |      570038 |      209232 | NONE               | GZIP               | UPLOADED |         |
| 01-02-2021.csv | 01-02-2021.csv.gz |      570267 |      209472 | NONE               | GZIP               | UPLOADED |         |
| 01-03-2021.csv | 01-03-2021.csv.gz |      570440 |      209536 | NONE               | GZIP               | UPLOADED |         |
| 01-04-2021.csv | 01-04-2021.csv.gz |      570450 |      209520 | NONE               | GZIP               | UPLOADED |         |
| 01-05-2021.csv | 01-05-2021.csv.gz |      570624 |      209552 | NONE               | GZIP               | UPLOADED |         |
| 01-06-2021.csv | 01-06-2021.csv.gz |      570636 |      209520 | NONE               | GZIP               | UPLOADED |         |
| 01-07-2021.csv | 01-07-2021.csv.gz |      570666 |      209760 | NONE               | GZIP               | UPLOADED |         |
| 01-08-2021.csv | 01-08-2021.csv.gz |      570839 |      209632 | NONE               | GZIP               | UPLOADED |         |
| 01-09-2021.csv | 01-09-2021.csv.gz |      570802 |      209840 | NONE               | GZIP               | UPLOADED |         |
| 01-10-2021.csv | 01-10-2021.csv.gz |      570842 |      209600 | NONE               | GZIP               | UPLOADED |         |
| 01-11-2021.csv | 01-11-2021.csv.gz |      570962 |      209904 | NONE               | GZIP               | UPLOADED |         |
| 01-12-2021.csv | 01-12-2021.csv.gz |      570993 |      209664 | NONE               | GZIP               | UPLOADED |         |
| 01-13-2021.csv | 01-13-2021.csv.gz |      571124 |      209904 | NONE               | GZIP               | UPLOADED |         |
```

```
| 04-18-2021.csv | 04-18-2021.csv.gz |      558043 |      201680 | NONE               | GZIP               | UPLOADED |         |
| 04-19-2021.csv | 04-19-2021.csv.gz |      558094 |      201568 | NONE               | GZIP               | UPLOADED |         |
| 04-20-2021.csv | 04-20-2021.csv.gz |      558091 |      201744 | NONE               | GZIP               | UPLOADED |         |
| 04-21-2021.csv | 04-21-2021.csv.gz |      558089 |      201824 | NONE               | GZIP               | UPLOADED |         |
| 04-22-2021.csv | 04-22-2021.csv.gz |      558036 |      201536 | NONE               | GZIP               | UPLOADED |         |
| 04-23-2021.csv | 04-23-2021.csv.gz |      558119 |      201744 | NONE               | GZIP               | UPLOADED |         |
| 04-24-2021.csv | 04-24-2021.csv.gz |      558191 |      201792 | NONE               | GZIP               | UPLOADED |         |
| 04-25-2021.csv | 04-25-2021.csv.gz |      558184 |      201792 | NONE               | GZIP               | UPLOADED |         |
| 04-26-2021.csv | 04-26-2021.csv.gz |      558300 |      201808 | NONE               | GZIP               | UPLOADED |         |
| 04-27-2021.csv | 04-27-2021.csv.gz |      558164 |      201936 | NONE               | GZIP               | UPLOADED |         |
| 04-28-2021.csv | 04-28-2021.csv.gz |      558200 |      201856 | NONE               | GZIP               | UPLOADED |         |
| 04-29-2021.csv | 04-29-2021.csv.gz |      558134 |      201808 | NONE               | GZIP               | UPLOADED |         |
| 04-30-2021.csv | 04-30-2021.csv.gz |      558235 |      201744 | NONE               | GZIP               | UPLOADED |         |
+---------------+------------------+-------------+-------------+--------------------+--------------------+----------+---------+
120 Row(s) produced. Time Elapsed: 33.166s
romin23k#COMPUTE_WH@COVID_DB.COVID_SCHEMA>
```

d. COPY the CSV data into snowflake table:

```
22 copy into "COVID_DB"."COVID_SCHEMA"."COVID_HISTORY"
23 from @~
24 file_format = ( format_name= csv_ff);
```

Results Data Preview Open History

Query ID SQL 3.89s 122 rows

Filter result... Copy Columns

Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_column_name
1	01-01-2021.csv.gz	LOADED	4011	4011	1	0	NULL	NULL	NULL	NULL
2	01-10-2021.csv.gz	LOADED	4012	4012	1	0	NULL	NULL	NULL	NULL
3	01-22-2021.csv.gz	LOADED	4014	4014	1	0	NULL	NULL	NULL	NULL
4	01-30-2021.csv.gz	LOADED	4014	4014	1	0	NULL	NULL	NULL	NULL
5	02-07-2021.csv.gz	LOADED	4014	4014	1	0	NULL	NULL	NULL	NULL
6	02-14-2021.csv.gz	LOADED	4014	4014	1	0	NULL	NULL	NULL	NULL
7	02-20-2021.csv.gz	LOADED	4014	4014	1	0	NULL	NULL	NULL	NULL
8	02-25-2021.csv.gz	LOADED	4014	4014	1	0	NULL	NULL	NULL	NULL

5. Clustering for the Covid_History Data was performed with 2 columns of the table: Last_update,Combined_key.

Clustering is a key factor in queries because table data that is not sorted or is only partially sorted may impact query performance, particularly on very large tables.

In Snowflake, as data is inserted/loaded into a table, clustering metadata is collected and recorded for each micro-partition created during the process. Snowflake then leverages this clustering information to avoid unnecessary scanning of micro-partitions during querying, significantly accelerating the performance of queries that reference these columns

```
31 //Clustering
32 alter table covid_history cluster
33 by (Last_Update, Combined_Key);
```

6. Performing Transformation :

After querying the table it was found that there were few discrepancies in the data. Hence, cleaning operations were performed to remove them.

a. Changing NULL to appropriate type

```
48 update covid_history set RECOVERED = ifnull(RECOVERED, 0);
49 update covid_history set INCIDENT_RATE = ifnull(INCIDENT_RATE, 0);
50 alter table covid_history rename column admin2 to  city;
51 update covid_history set CITY = ifnull(CITY, 'NA');
52 update covid_history set PROVINCE_STATE = ifnull(PROVINCE_STATE, 'NA');
53 update covid_history set ACTIVE = ifnull(ACTIVE, 0);
54 update covid_history set DEATHS = ifnull(DEATHS, 0);
55 update covid_history set case_fatality_ratio = ifnull(case_fatality_ratio, 0);
56 update covid_history set INCIDENT_RATE = ifnull(INCIDENT_RATE, 0);
57 update covid_history set lat = ifnull(lat, 0);
58 update covid_history set long = ifnull(long, 0);
```

b. Deleting any data other then year 2021

```
60 select * from covid_current
61 where LAST_UPDATE NOT like  '2021%';
62 delete from covid_history where LAST_UPDATE NOT like '2021%';
```

7. External Named Stage and Data Loading:

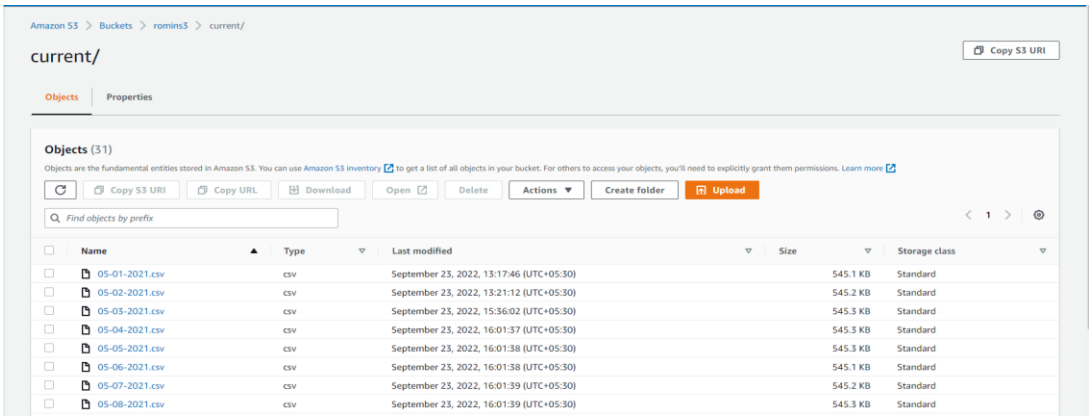
An external stage is created to store the current data. Later the staging area is used as a source to load the current data into the ‘COVID_CURRENT’ table.\

Following are the steps:

a. Created External Stage to load data from external source i.e AWS S3

```
67 //Staage External(S3)
68 CREATE STAGE "COVID_DB"."COVID_SCHEMA".PIPE_STAGE URL = 's3://romins3/current'
69 CREDENTIALS = (AWS_KEY_ID = 'AKIAYLYEKLCYB2MNQ76K'
70                AWS_SECRET_KEY = '****************************************');
```

b. Uploading Data in the AWS S3 bucket



c. Storage Integration

```
118 create or replace storage integration covid_integration_current
119   type = external_stage
120   storage_provider = s3
121   enabled = true
122   storage_aws_role_arn = 'arn:aws:iam::574997813424:role/romin_s3_role'
123   storage_allowed_locations = ('s3://romins3/current/01-07-2022.csv');
124 Desc integration covid_integration_current;
```

d. Table Creation for Current Covid Data

```
create or replace table covid_current (
   FIPS                number              ,
   City              varchar(500)          ,
   Province_State      varchar(500)        ,
   Country_Region      varchar(500)      not null,
   Last_Update         date        ,
   Lat                 double precision            ,
   Long               double precision             ,
   Confirmed          number              ,
   Deaths              number              ,
   Recovered           number              ,
   Active              number              ,
  Combined_Key         varchar(500)        ,
  Incident_Rate        number              ,
  Case_Fatality_Ratio  number
);
```

8. **Creating a snowpipe to continuously load new data from the external stage from s3 bucket to the staging table.**

Snowpipe enables loading data from files as soon as they're available in a stage. This means you can load data from files in micro-batches, making it available to users within minutes, rather than manually executing COPY statements on a schedule to load larger batches.

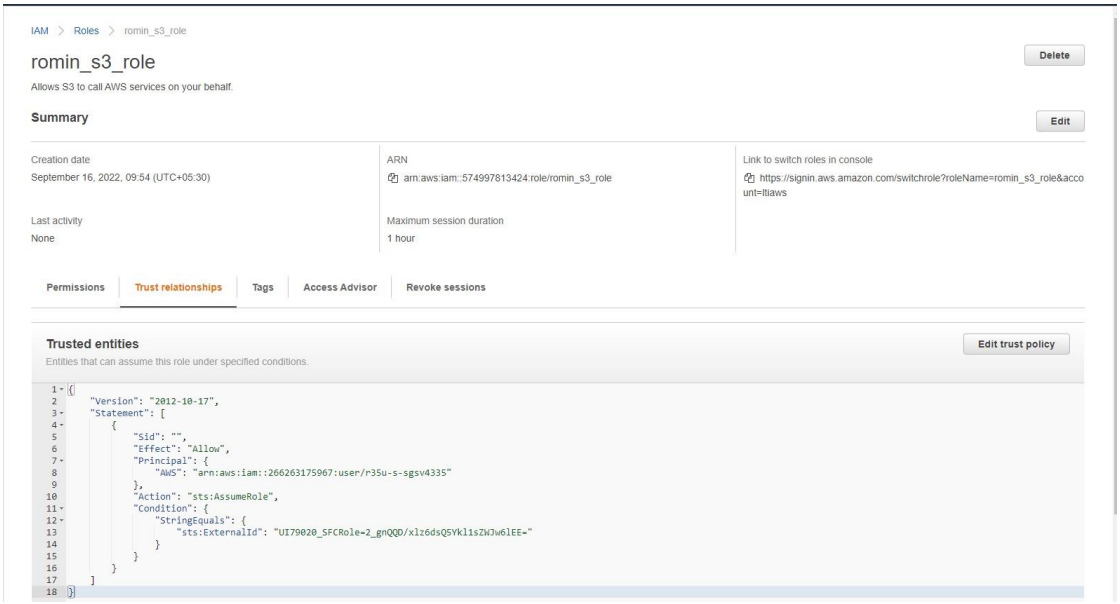
Automated data loads leverage event notifications for cloud storage to inform Snowpipe of the arrival of new data files to load. Snowpipe copies the files into a queue, from which they are loaded into the target table in a continuous, serverless fashion based on parameters defined in a specified pipe object.

Following are the steps:

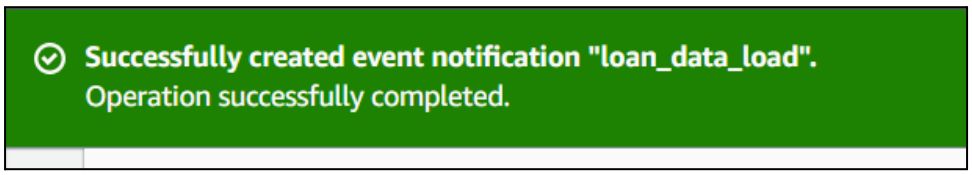
a. Creating a snowpipe

```
create or replace pipe snowpipe auto_ingest=true
as
copy into COVID_CURRENT
from (select ifnull(t.$1,0),ifnull(t.$2,'NA'),ifnull(t.$3,'NA'),
      t.$4,t.$5,t.$6,t.$7,t.$8,t.$9,ifnull(t.$10,0),ifnull(t.$11,0),
      t.$12,t.$13,t.$14 from @PIPE_STAGE t)
file_format=csv_ff;
show pipes;
list @PIPE_STAGE;
```

b. Editing Trust Relationship



c. Creating Event Notification



9. Maintaining Change Data Capture and Slowly Changing Data

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 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.

A **Slowly Changing Dimension (SCD)** is a dimension that stores and manages both current and historical data over time in a data warehouse. It is considered one of the most **critical** ETL tasks in **tracking the history** of dimension records.
A **Type 2** SCD retains the full history of values. When the value of a chosen attribute changes, the current record is closed and a new record is created with the changed data values. This new record becomes the current record. Each record contains the effective time and expiration time to identify the time period between which the record was active.

a. **Creating the Stream : 'covid_stream' on 'covid_current' table**
   As soon as the newly added current covid data is imported into the snowflake tables via snowpipe, the stream captures it and merges it to the **"covid_history"** table.

```
141 //Creating Stream
142 create or replace stream covid_stream on table covid_current
143 insert_only = True;
144
145 show streams; --display streams present
146 desc stream covid_stream;
147 select system$stream_has_data('covid_db.covid_schema.covid_stream'); --Gives Boolean
148 select * from covid_stream; -- Data inside stream
```

b. **Creating task :**To copy data from stream to 'Covid_history' table a task is created to append the data to the covid history table

```
//Tasks for merging
create or replace task task_covid_history
warehouse = compute_wh
schedule ='1 minute'
when system$stream_has_data('covid_stream')
as
merge into covid_history
using covid_stream on
covid_history.last_update=covid_stream.last_update and
covid_history.COMBINED_KEY=covid_stream.COMBINED_KEY
when not matched and covid_stream.METADATA$ACTION='INSERT' then
INSERT(FIPS,city,Province_State,Country_Region,Last_Update,Lat,Long,
       Confirmed,Deaths,Recovered,Active,Combined_Key,Incident_Rate,
       Case_Fatality_Ratio)
       values(
   covid_stream.FIPS,
   covid_stream.city,
   covid_stream.Province_State,
   covid_stream.Country_Region,
   covid_stream.Last_Update,
   covid_stream.Lat,
   covid_stream.Long,
   covid_stream.Confirmed,
   covid_stream.Deaths,
   covid_stream.Recovered,
   covid_stream.Active,
  covid_stream.Combined_Key,
  covid_stream.Incident_Rate,
  covid_stream.Case_Fatality_Ratio
 );
show tasks; --Show available tasks
alter task task_covid_history resume; --start the yask if suspended
```

10. Sharing Data with Non SnowFlake user :

Reader accounts provide a quick, easy, and cost-effective way to share data without requiring the consumer to become a Snowflake customer.

Each reader account belongs to the provider account that created it. Similar to standard consumer accounts, the provider account uses shares to share databases with reader accounts; however, a reader account can only consume data from the provider account that created it

a. Creating outbound share :

```
228 //Share Cretion
229 create share COVID_share;
```

b. Granting privileges:

```
231 //Granting objects to share
232 grant usage on database "COVID_DB" to share COVID_share;
233 grant usage on schema "COVID_DB"."COVID_SCHEMA" to share COVID_share;
234 grant select on table "COVID_DB"."COVID_SCHEMA"."COVID_CURRENT" to share COVID_share;
235 grant select on table "COVID_DB"."COVID_SCHEMA"."COVID_HISTORY" to share COVID_share;
236 grant select on table "COVID_DB"."COVID_SCHEMA"."POPULATION" to share COVID_share;
```

c. Creating Reader Account:

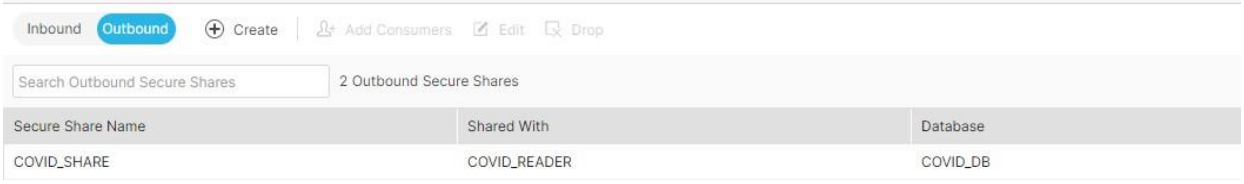
```
225 CREATE MANAGED ACCOUNT covid_reader
226 admin_name='covid_reader',
227 admin_password='Abc123**',
228 type=reader;
```

d. Assigning Share to Reader Account:

```
242 //add reader account to share object
243 ALTER SHARE "COVID_SHARE" ADD ACCOUNTS = UB93461;
```

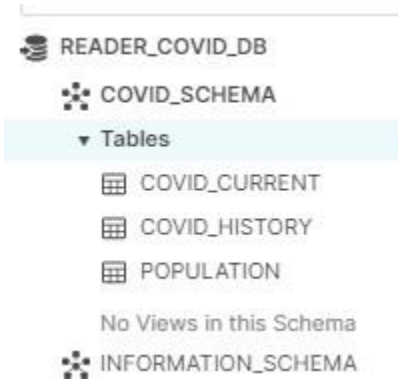
e. Inbounds and Outbounds:

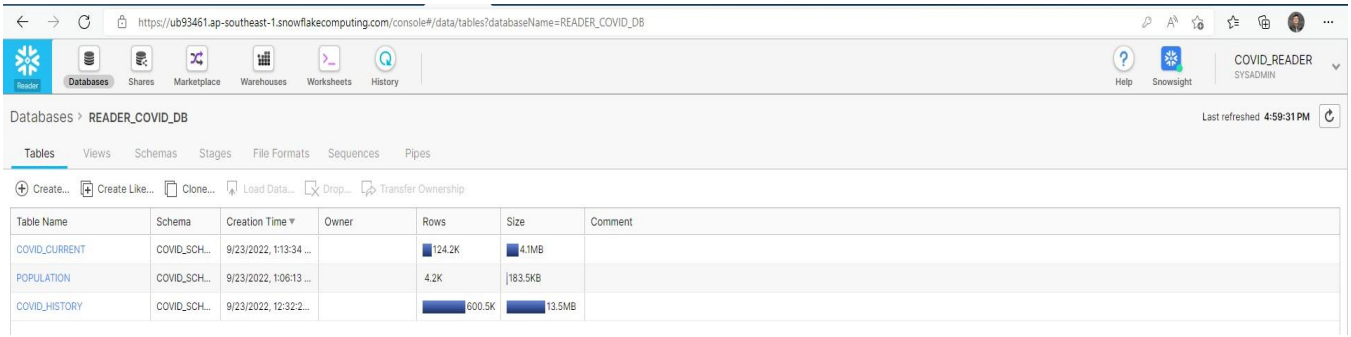
i. <u>Outbound from Main Acc</u> :



ii. <u>Inbound from Reader Account</u> :

f. Reader account can see the tables :





11. Create clone of the table with time travel before one day and get history data of a table

Zero Copy Cloning : Cloning, also referred to as “zero-copy cloning” , creates a copy of a database, schema or table. A snapshot of data present in the source object is taken when the clone is created and is made available to the cloned object. It just updates the metadata which references the source data and prevents creation of physical copy of the data.

Time Travel: Snowflake Time Travel enables accessing historical data (i.e. data that

has been changed or deleted) at any point within a defined period. It serves as a powerful tool
for performing the following tasks:

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

a. Clone Table created

```
186 Select current_timestamp();
187 --22022-09-23 01:11:56.987 -0700
188 Create or replace Table Covid_Clone Clone Covid_Current
189 AT (Timestamp => '2022-09-23 16:11:56.987 +0530'::timestamp); --yesterday timestamp
```

b. Time Travel

```
210 --with respect to statement and query id
211 Select * from Covid_Clone
212 BEFORE(STATEMENT => '01a72779-3200-8b15-0001-d0be0002fff2')
213 where FIPS=101;  ---01a72779-3200-8b15-0001-d0be0002fff2 (Query id)
214
215 --time travel WRT OFFSET
216 Select * from Covid_Clone AT(OFFSET => -60*1)
217 where country_region='Afghanistan';
```

12. Create stored procedure to insert the data into table after typecasting date format

A **stored procedure** ‘data_insert_into’ is written in javascript language. The procedure helps to insert a new record into the loan_data_clone table. The insert command with all typecasting for date columns are written as sql command and executed.

```
//=============================================================================================================
create or replace procedure insert_data()
returns varchar
Language javascript
EXECUTE As caller
As $$
try{
    snowflake.execute(
        {sqlText: "begin transaction;"}
        );
        snowflake.execute(
        {sqlText: `INSERT INTO covid_current VALUES(08,'','','Afghanistan',TO_DATE('2022-06-02', 'yyyy-mm-dd'),
        33.93911,67.709953,180419,7707,20,25,'Afghanistan',463.4643,4.24155);`}
        );

       snowflake.execute (
       {sqlText: "commit;"}
       );
       return_value = 'sucess';
}
catch(err)
{
    snowflake.execute (
    {sqlText: "rollback;"}
    );
    return_value = err;
}

return return_value;
$$;

call insert_data();
//=============================================================================================================
```

13. Creating data for easy visualization

a. Populations table

```
--
create table temporary  COUNTRY_POPULATON(
    COUNTRY_REGION VARCHAR(500),
    POPULATION NUMBER
    ) AS
select POPULATION.COUNTRY_REGION, max(POPULATION.POPULATION) from COVID_HISTORY
inner join POPULATION
on  COVID_HISTORY.COUNTRY_REGION = POPULATION.COUNTRY_REGION
group by POPULATION.COUNTRY_REGION;
```

b. Table for initial confirmed cases

```
create table temporary  initial_confirmed (
    COUNTRY_REGION VARCHAR(500),
    INITIAL_CONFIRMED NUMBER
    ) AS
select COVID_HISTORY.COUNTRY_REGION, sum(COVID_HISTORY.CONFIRMED) from COVID_HISTORY
where last_update = '2021-01-02'
group by COVID_HISTORY.COUNTRY_REGION
order by COVID_HISTORY.COUNTRY_REGION;
```

c. Table for final covid status

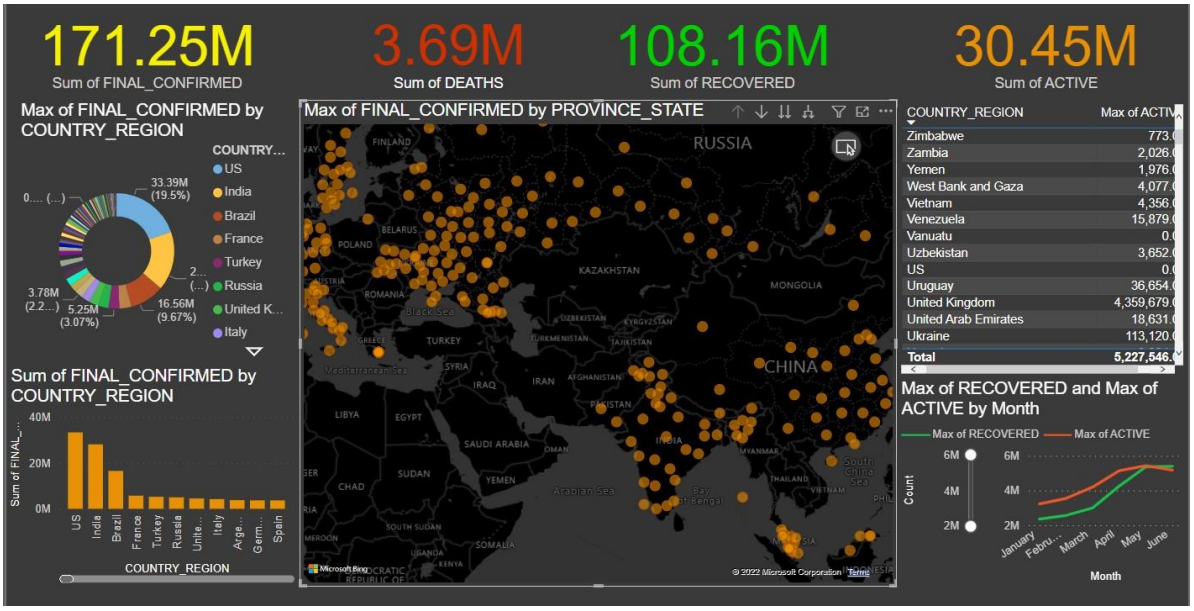
```
create OR REPLACE TEMPORARY table FINAL_status (
    COUNTRY_REGION VARCHAR(500),
    confirmed number
    death NUMBER,
    active NUMBER,
    recovered NUMBER
    ) AS
select COVID_HISTORY.COUNTRY_REGION,sum(COVID_HISTORY.CONFIRMED), sum(COVID_HISTORY.deaths),
sum(COVID_HISTORY.active), sum(COVID_HISTORY.recovered) from COVID_HISTORY
where last_update = '2021-06-01'
group by COVID_HISTORY.COUNTRY_REGION
order by COVID_HISTORY.COUNTRY_REGION;
```

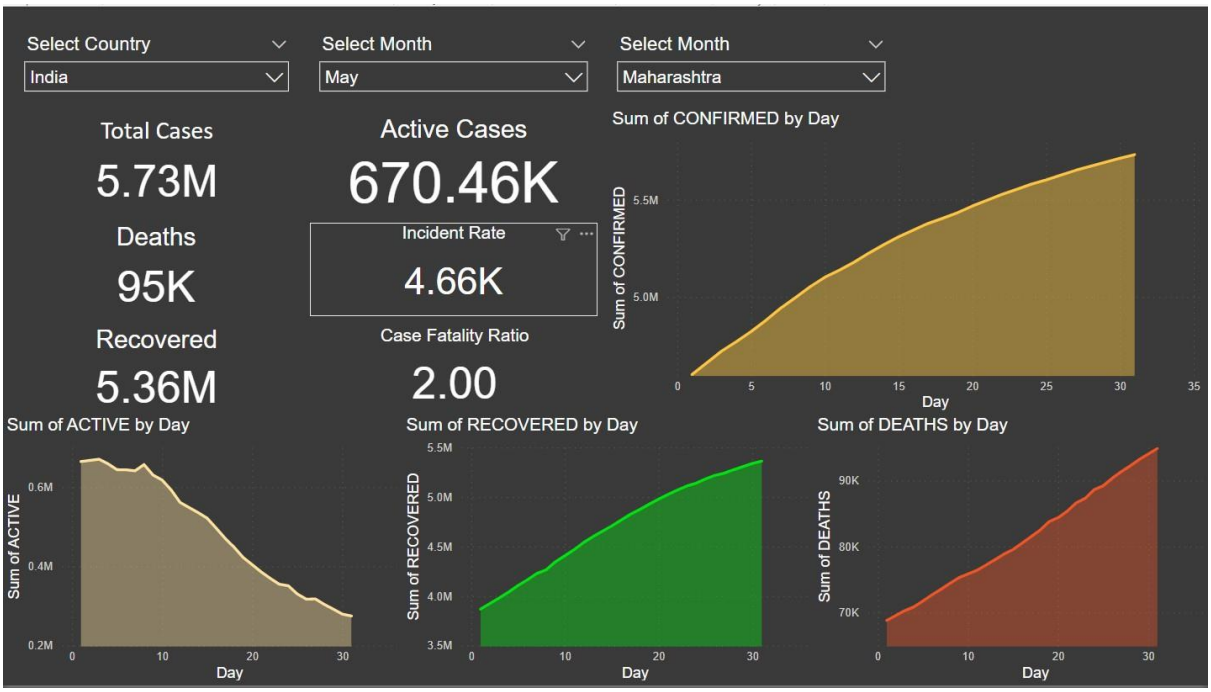
d. Final table used for visualizations

```
CREATE OR REPLACE TRANSIENT TABLE BUSINESS_QUE_v2(
    COUNTRY_REGION VARCHAR(500),
    POPULATION NUMBER,
    INITIAL_CONFIRMED NUMBER,
    FINAL_CONFIRMED DOUBLE PRECISION,
    active NUMBER,
    recovered NUMBER,
    deaths NUMBER,
    PERCT_INCR NUMBER,
    CONF_PER_CAPITA DOUBLE PRECISION
)AS
SELECT COUNTRY_POPULATON.COUNTRY_REGION, COUNTRY_POPULATON.POPULATION, INITIAL_CONFIRMED.INITIAL_CONFIRMED,
FINAL_CONFIRMED.FINAL_CONFIRMED, FINAL_status.ACTIVE, FINAL_status.RECOVERED, FINAL_status.DEATH,
BUSINESS_QUE.PERCT_INCR, BUSINESS_QUE.CONF_PER_CAPITA FROM COUNTRY_POPULATON
INNER JOIN INITIAL_CONFIRMED
ON COUNTRY_POPULATON.COUNTRY_REGION = INITIAL_CONFIRMED.COUNTRY_REGION
INNER JOIN FINAL_CONFIRMED
ON COUNTRY_POPULATON.COUNTRY_REGION = FINAL_CONFIRMED.COUNTRY_REGION
INNER JOIN FINAL_STATUS
ON COUNTRY_POPULATON.COUNTRY_REGION = FINAL_STATUS.COUNTRY_REGION
INNER JOIN BUSINESS_QUE
ON COUNTRY_POPULATON.COUNTRY_REGION = BUSINESS_QUE.COUNTRY_REGION;
```

Visualizations Using PowerBI

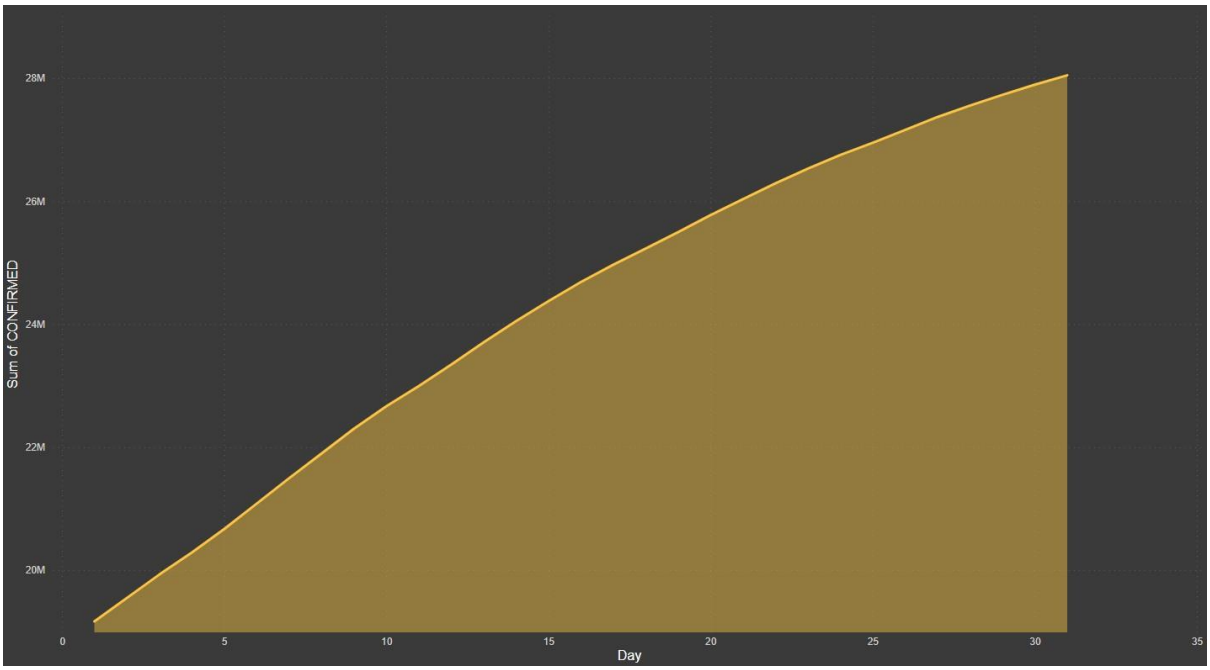
Dashboard



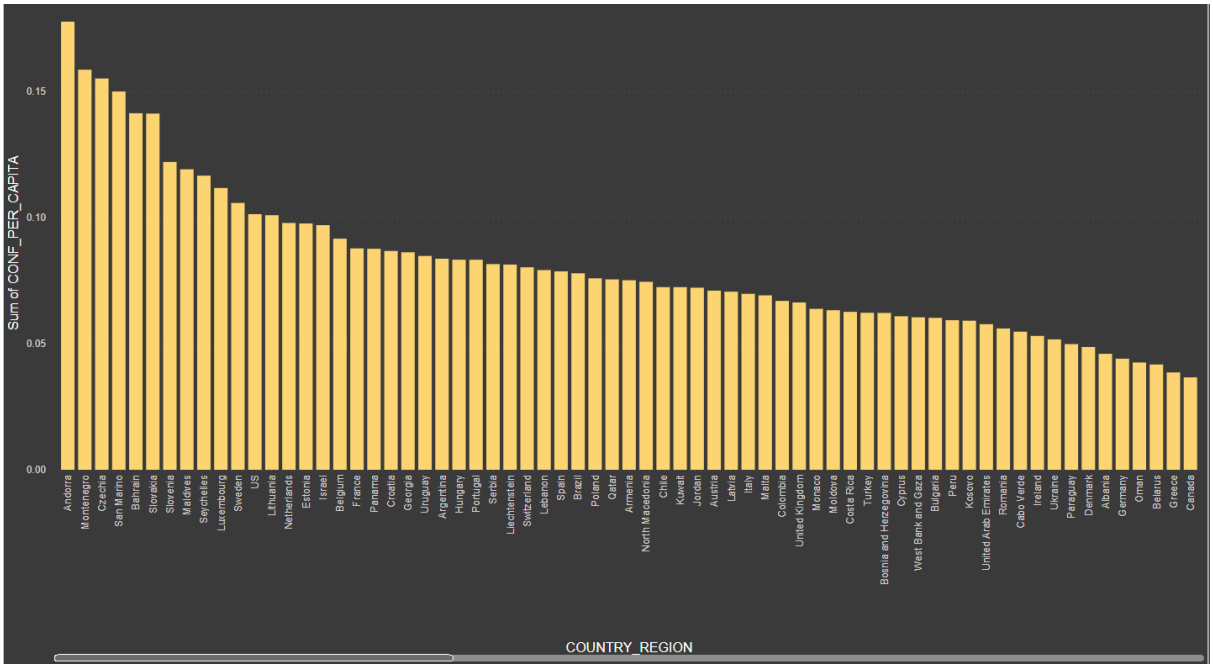


Business Queries and Visualizations :

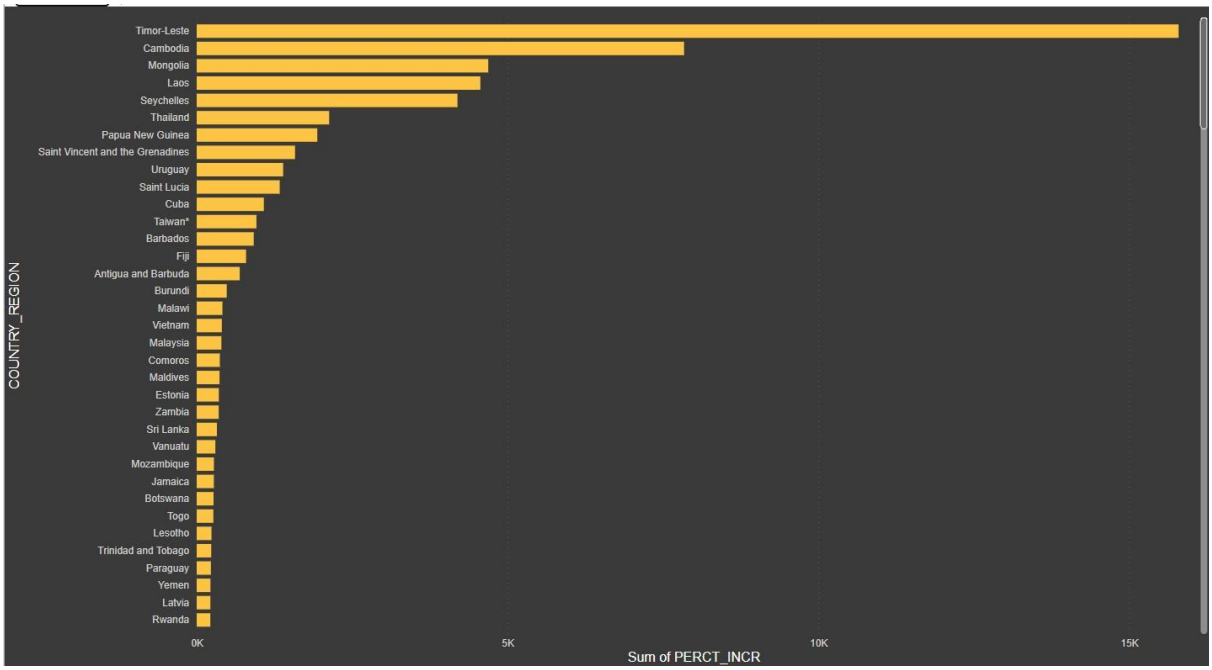
1. **What is the daily number of confirmed cases(Month: May, Country: India)?**



2. **Which countries with the highest number of confirmed cases have the most per capita?**



3. **Where are confirmed cases increasing most rapidly?**



4. Confirmed ,Active, Recovered cases by month :