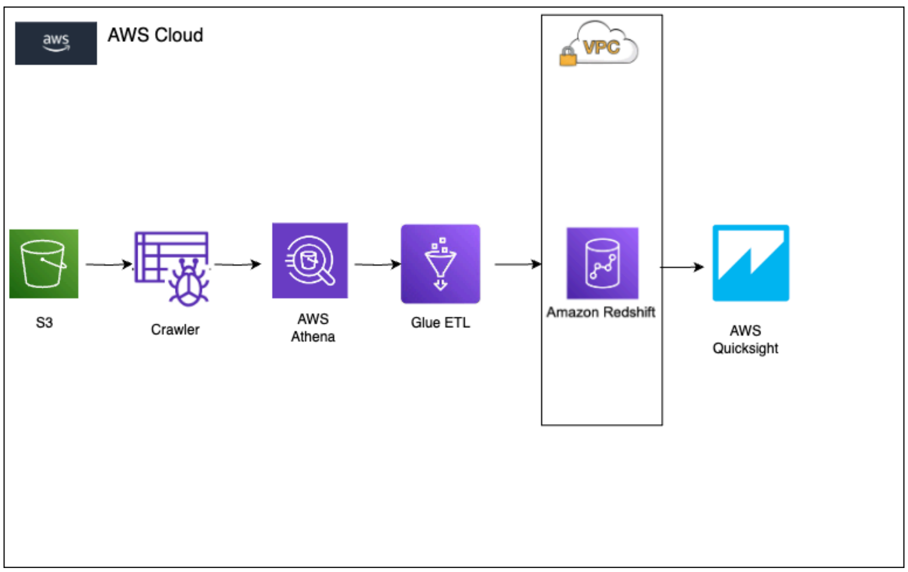
COVID - 19 End to End Data Engineering Project

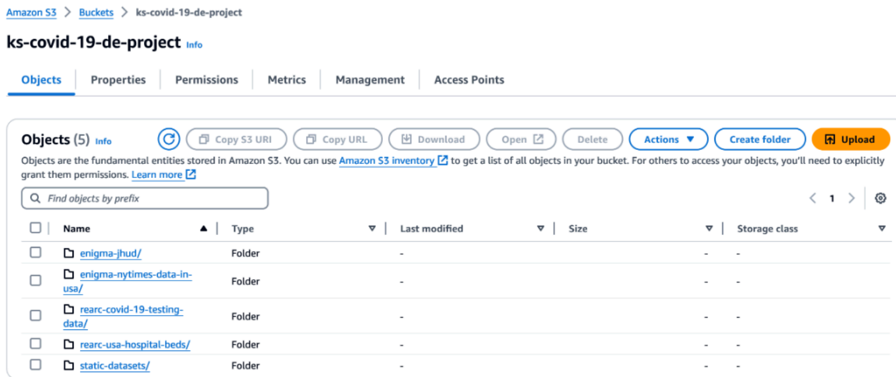
- This study employs data mining techniques to analyze the COVID-19 project dataset, aiming to uncover patterns, identify trends, and determine risk factors associated with SARS-CoV-2 infection.
- By examining the provided data, the research seeks to extract valuable insights into the dynamics of COVID-19 transmission and susceptibility.
- Data set used: Available on AWS Open Dataset - https://aws.amazon.com/covid-19-data-lake/

ARCHITECTURE:



- This Architecture depicts a data processing and analytics pipeline in AWS Cloud.
- Here's an explanation of the flow:
  - **S3 (Simple Storage Service):** The starting point, likely where raw data is initially stored.
  - **Crawler**: A tool that scans the data in S3 to determine its structure and schema.
  - **AWS Athena**: A query service that allows analysis of data directly in S3 using SQL.
  - **Glue ETL**: Performs Extract, Transform, Load operations to prepare and structure the data.
  - **Amazon Redshift**: A data warehouse service where the processed data is stored for analytics.
  - **AWS QuickSight**: A business intelligence tool for creating visualizations and dashboards from the data in Redshift.
- All of these services (except QuickSight) are shown within a **VPC** (Virtual Private Cloud), indicating a secure, isolated network environment.
- This pipeline allows for ingesting raw data, processing it, storing it in a structured format, and then analyzing and visualizing it - all within the AWS ecosystem.
- This kind of architecture is for big data analytics and business intelligence applications.

WORKFLOW:

- **Step 1:** Manually cleaned the data and uploaded to a S3 bucket in folders.
- 

Amazon S3 > Buckets > ks-covid-19-de-project

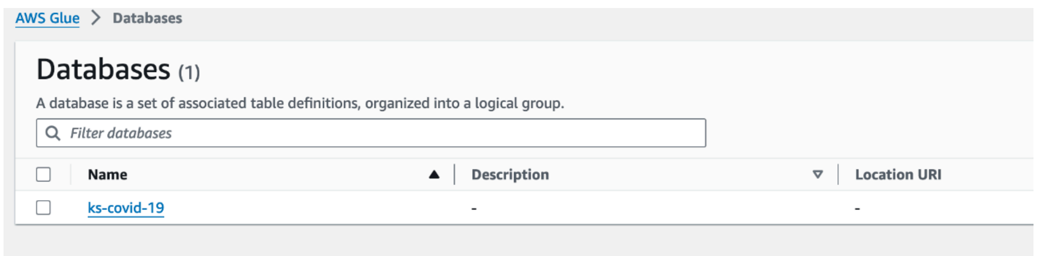
ks-covid-19-de-project Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (5) Info

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

Name	Type	Last modified	Size	Storage class
enigma-jhud/	Folder	-	-	-
enigma-nytimes-data-in-usa/	Folder	-	-	-
rearc-covid-19-testing-data/	Folder	-	-	-
rearc-usa-hospital-beds/	Folder	-	-	-
static-datasets/	Folder	-	-	-

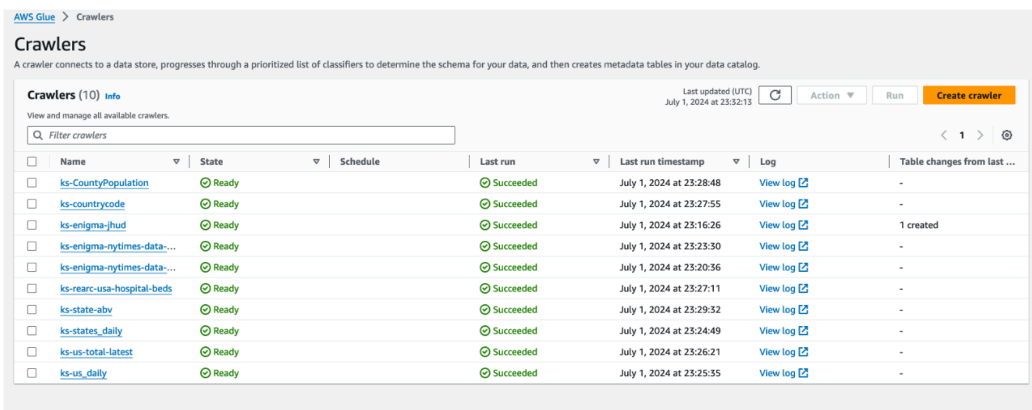
- **Step 2**: Ran Crawler on all the S3 folders to get Metadata information in the form of tables to be used with Athena for Analysis
- 

AWS Glue > Databases

Databases (1)

A database is a set of associated table definitions, organized into a logical group.

Name	Description	Location URI
ks-covid-19	-	-

- 

AWS Glue > Crawlers

Crawlers

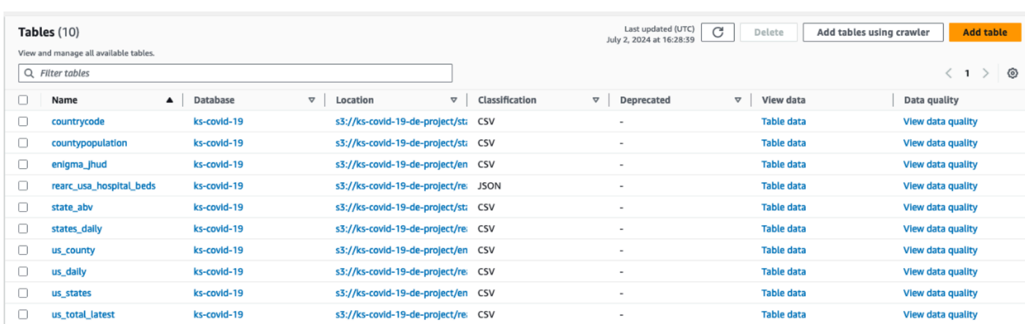
A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawlers (10) Info

Last updated (UTC) July 1, 2024 at 23:32:13

View and manage all available crawlers.

Name	State	Schedule	Last run	Last run timestamp	Log	Table changes from last ...
ks-CountyPopulation	Ready		Succeeded	July 1, 2024 at 23:28:48	View log	-
ks-countrycode	Ready		Succeeded	July 1, 2024 at 23:27:55	View log	-
ks-enigma-jhud	Ready		Succeeded	July 1, 2024 at 23:16:26	View log	1 created
ks-enigma-nytimes-data-...	Ready		Succeeded	July 1, 2024 at 23:23:30	View log	-
ks-enigma-nytimes-data-...	Ready		Succeeded	July 1, 2024 at 23:20:36	View log	-
ks-rearc-usa-hospital-beds	Ready		Succeeded	July 1, 2024 at 23:27:11	View log	-
ks-state-abv	Ready		Succeeded	July 1, 2024 at 23:29:32	View log	-
ks-states_daily	Ready		Succeeded	July 1, 2024 at 23:24:49	View log	-
ks-us-total-latest	Ready		Succeeded	July 1, 2024 at 23:26:21	View log	-
ks-us_daily	Ready		Succeeded	July 1, 2024 at 23:25:35	View log	-

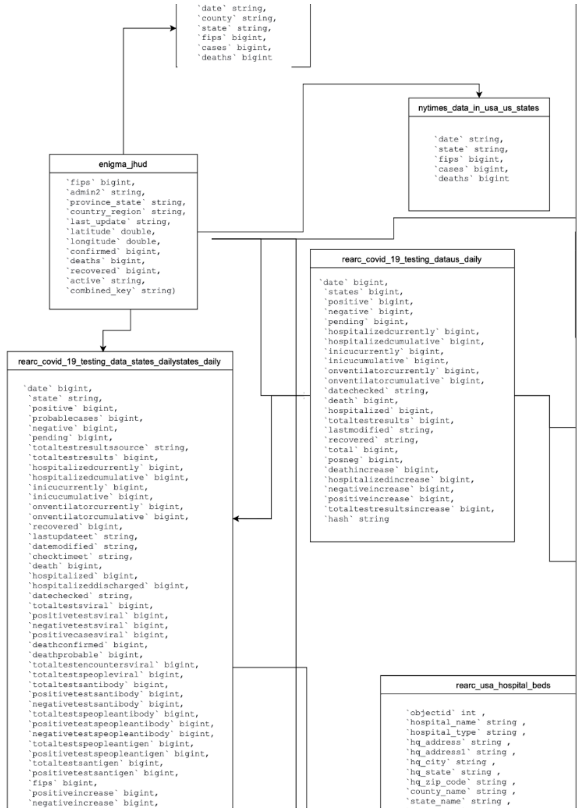
- 

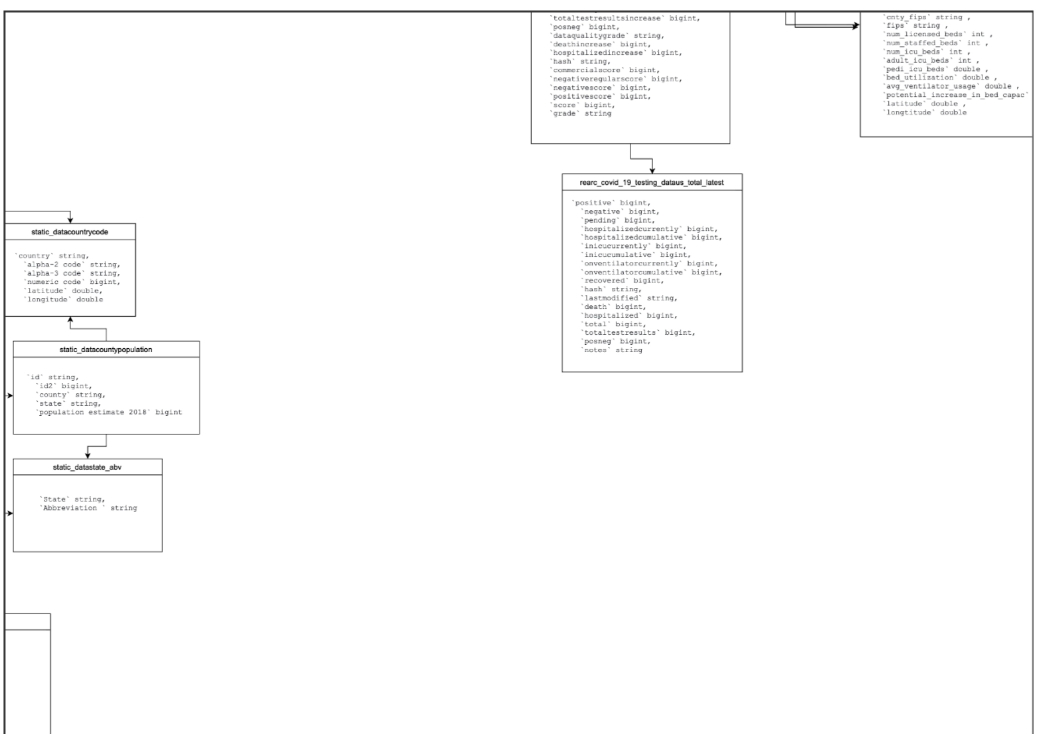
Tables (10)

Last updated (UTC) July 2, 2024 at 16:28:39

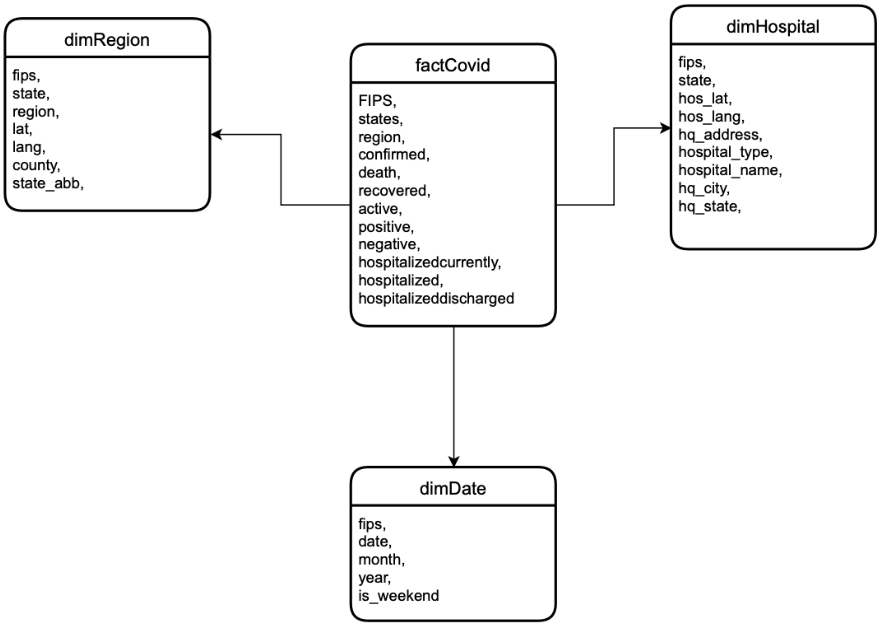
View and manage all available tables.

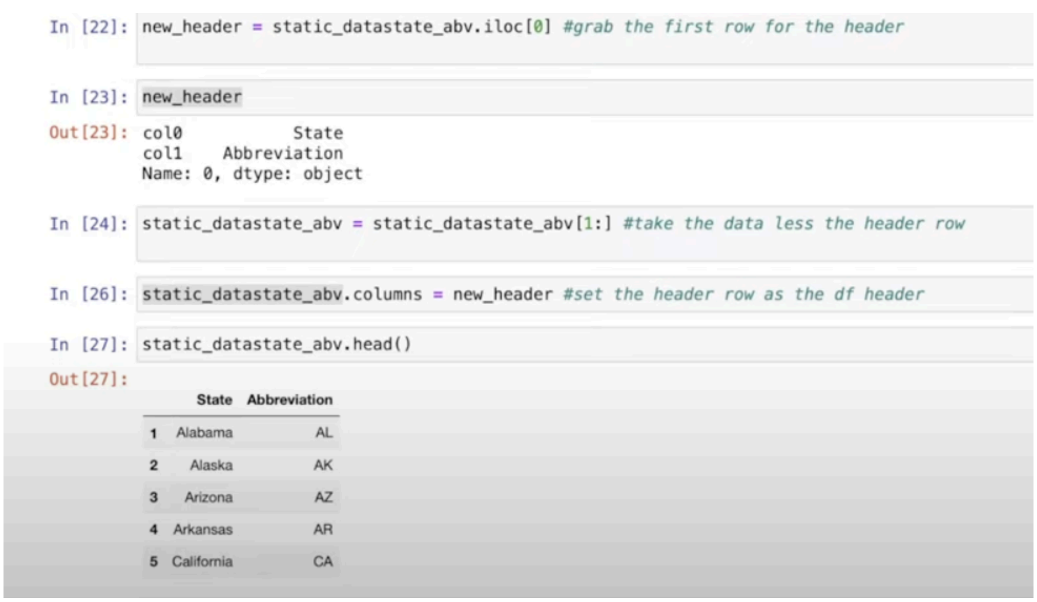
Name	Database	Location	Classification	Deprecated	View data	Data quality
countrycode	ks-covid-19	s3://ks-covid-19-de-project/st	CSV	-	Table data	View data quality
countypopulation	ks-covid-19	s3://ks-covid-19-de-project/st	CSV	-	Table data	View data quality
enigma_jhud	ks-covid-19	s3://ks-covid-19-de-project/en	CSV	-	Table data	View data quality
rearc_usa_hospital_beds	ks-covid-19	s3://ks-covid-19-de-project/re	JSON	-	Table data	View data quality
state_abv	ks-covid-19	s3://ks-covid-19-de-project/st	CSV	-	Table data	View data quality
states_daily	ks-covid-19	s3://ks-covid-19-de-project/re	CSV	-	Table data	View data quality
us_county	ks-covid-19	s3://ks-covid-19-de-project/en	CSV	-	Table data	View data quality
us_daily	ks-covid-19	s3://ks-covid-19-de-project/re	CSV	-	Table data	View data quality
us_states	ks-covid-19	s3://ks-covid-19-de-project/en	CSV	-	Table data	View data quality
us_total_latest	ks-covid-19	s3://ks-covid-19-de-project/re	CSV	-	Table data	View data quality

- **Step 3:** Once all the tables were created by Crawler, used Athena to preview the tables created and get the DDL scripts of those table to create Data Model.
- 

- 

- **Step 4:** With the help of the above Data Model, created a Dimension Model using Star Schema.
-



- **Step 4**: Using Python, Connected to Athena to query the data and storing the data in the tables created using Panda
- **Step 5:** With Pandas, transformed the data into the appropriate format: like replacing nulls, having the right column header, converted integer date into proper date format (yyyy-mm-dd), etc.
- 

```
In [22]: new_header = static_datastate_abv.iloc[0] #grab the first row for the header

In [23]: new_header
Out[23]: col0            State
         col1     Abbreviation
         Name: 0, dtype: object

In [24]: static_datastate_abv = static_datastate_abv[1:] #take the data less the header row

In [26]: static_datastate_abv.columns = new_header #set the header row as the df header

In [27]: static_datastate_abv.head()
Out[27]:
```

	State	Abbreviation
1	Alabama	AL
2	Alaska	AK
3	Arizona	AZ
4	Arkansas	AR
5	California	CA

-

0	2	20210307
1	1	20210307
2	5	20210307
3	60	20210307
4	4	20210307

```
In [35]: dimDate['date'] = pd.to_datetime(dimDate['date'], format='%Y%m%d')

/var/folders/0n/nb074wzd4kl1nmdvkbdnpgym0000gn/T/ipykernel_40148/572748324.py:1: SettingWithC
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
ng-a-view-versus-a-copy
  dimDate['date'] = pd.to_datetime(dimDate['date'], format='%Y%m%d')

In [36]: dimDate.head()
Out[36]:
```

	fips	date
0	2	2021-03-07
1	1	2021-03-07
2	5	2021-03-07
3	60	2021-03-07
4	4	2021-03-07

- **Step 6:** Using Pandas, created Fact and Dimension Tables from the above Dimension model and transformed the data.
-

```
In [28]: factCovid_1 = enigma_jhud[['fips','province_state','country_region','confirmed','deaths','recovered','active']]
         factCovid_2 = rearc_covid_19_testing_data_states_dailystates_daily[['fips','date','positive','negative','hospitalize
         factCovid = pd.merge(factCovid_1, factCovid_2, on='fips', how='inner')

In [30]: factCovid.shape
Out[30]: (26418, 13)

In [31]: dimRegion_1 = enigma_jhud[['fips','province_state','country_region','latitude','longitude']]
         dimRegion_2 = nytimes_data_in_usa_us_county[['fips','county','state']]
         dimRegion = pd.merge(dimRegion_1, dimRegion_2, on='fips', how='inner')

In [32]: eds[['fips','state_name','latitude','longtitude','hq_address','hospital_name','hospital_type','hq_city','hq_state']]

In [33]: dimDate = rearc_covid_19_testing_data_states_dailystates_daily[['fips','date']]
```

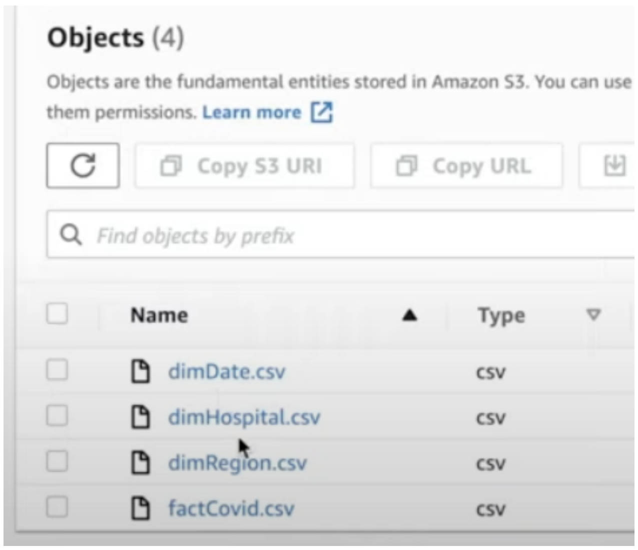
- Created Date dim table:
-

```
In [37]: dimDate['year'] = dimDate['date'].dt.year
         dimDate['month'] = dimDate['date'].dt.month
         dimDate["day_of_week"] = dimDate['date'].dt.dayofweek

/var/folders/0n/nb074wzd4kl1nmdvkbdnpgym0000gn/T/ipykernel_40148/2445661104.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy
  dimDate['year'] = dimDate['date'].dt.year
/var/folders/0n/nb074wzd4kl1nmdvkbdnpgym0000gn/T/ipykernel_40148/2445661104.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy
  dimDate['month'] = dimDate['date'].dt.month
/var/folders/0n/nb074wzd4kl1nmdvkbdnpgym0000gn/T/ipykernel_40148/2445661104.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy
  dimDate["day_of_week"] = dimDate['date'].dt.dayofweek
```

-

```
In [38]: dimDate.head()
Out[38]:
```

	fips	date	year	month	day_of_week
0	2	2021-03-07	2021	3	6
1	1	2021-03-07	2021	3	6
2	5	2021-03-07	2021	3	6
3	60	2021-03-07	2021	3	6
4	4	2021-03-07	2021	3	6

- **Step 7:** Stored the output of the Fact and Dimension tables in another S3 bucket.
- 

- **Step 8:** Create Schemas from the tables created above (Fact and Dimension) using pandas
-

Out[133]:

	fips	date	year	month	day_of_week
0	2	2021-03-07	2021	3	6
1	1	2021-03-07	2021	3	6
2	5	2021-03-07	2021	3	6
3	60	2021-03-07	2021	3	6
4	4	2021-03-07	2021	3	6

In [132]:
```
dimDatesql = pd.io.sql.get_schema(dimDate.reset_index(), 'dimDate')
print(''.join(dimDatesql))
```

```
CREATE TABLE "dimDate" (
"index" INTEGER,
  "fips" INTEGER,
  "date" TIMESTAMP,
  "year" INTEGER,
  "month" INTEGER,
  "day_of_week" INTEGER
)
```

In [ ]:
```
factCovidsql = pd.io.sql.get_schema(factCovid.reset_index(), 'factCovid')
print(''.join(factCovidsql))
```

In [ ]:
```
dimRegionsql = pd.io.sql.get_schema(dimRegion.reset_index(), 'dimRegion')
print(''.join(dimRegionsql))
```

In [ ]:
```
dimHospitalsql = pd.io.sql.get_schema(dimHospital.reset_index(), 'dimHospital')
print(''.join(dimHospitalsql))
```

- **Step 9:** Using Redshift Connector library , connect to Redshift programmatically and create these tables in Redshift.
- In [137]:
```
import redshift_connector
```

In [138]:
```
conn = redshift_connector.connect(
     host='redshift-cluster-2.ctlwvzbuur6m.ap-south-1.redshift.amazonaws.com',
     database='dev',
     user="awsuser",
     password='Passw0rd123'
  )
```

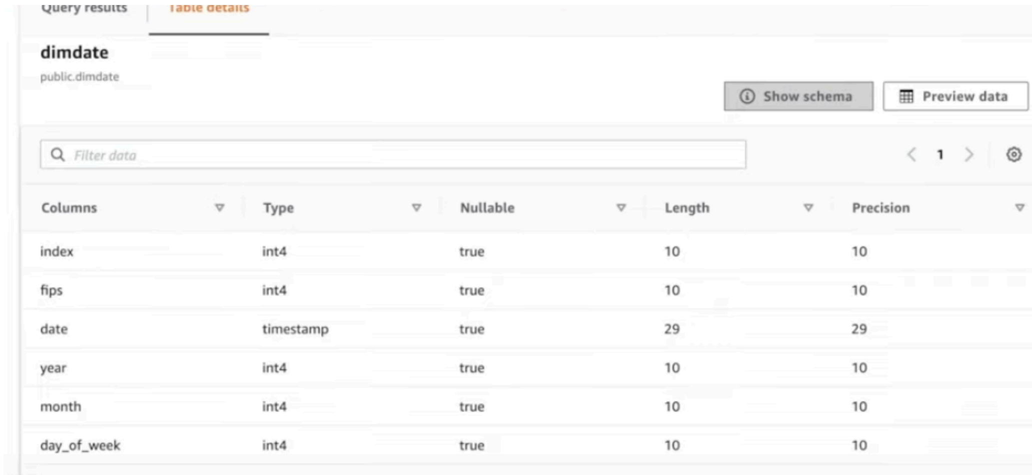
In [139]:
```
conn.autocommit = True
```

In [140]:
```
cursor= redshift_connector.Cursor = conn.cursor()
```

In [141]:
```
cursor.execute("""
CREATE TABLE "dimDate" (
"index" INTEGER,
  "fips" INTEGER,
  "date" TIMESTAMP,
  "year" INTEGER,
  "month" INTEGER,
  "day_of_week" INTEGER
)
""")
```

Out[141]:
```
<redshift_connector.cursor.Cursor at 0x12fe311e0>
```

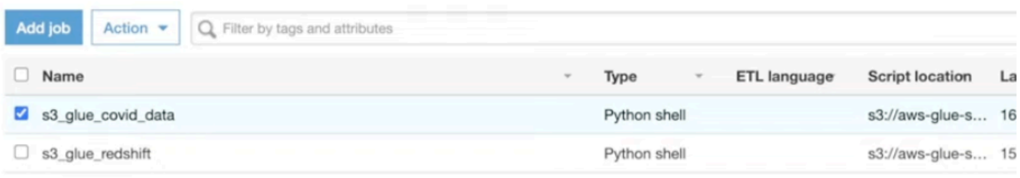
-



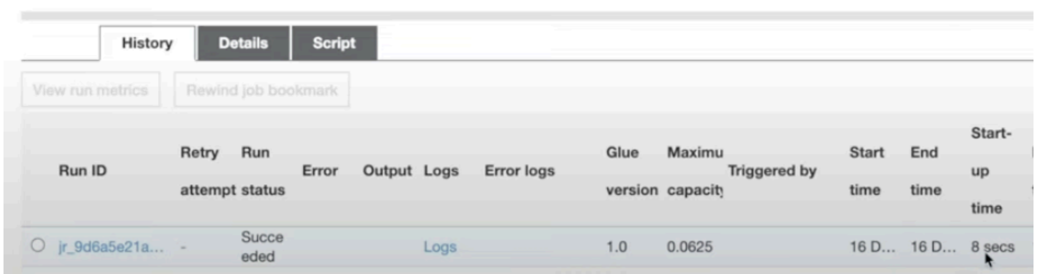
Columns	Type	Nullable	Length	Precision
index	int4	true	10	10
fips	int4	true	10	10
date	timestamp	true	29	29
year	int4	true	10	10
month	int4	true	10	10
day_of_week	int4	true	10	10

- **Step 10:** Using Copy command copy the data from S3 to Redshift cluster.

- **Step 11:** Create Glue Job using Redshift connector and create script
-



Name	Type	ETL language	Script location	La
s3_glue_covid_data	Python shell		s3://aws-glue-s...	16
s3_glue_redshift	Python shell		s3://aws-glue-s...	15



Run ID	Retry attempt	Run status	Error	Output	Logs	Error logs	Glue version	Maximu capacity	Triggered by	Start time	End time	Start-up time
jr_9d6a5e21a...	-	Succeeded			Logs		1.0	0.0625		16 D...	16 D...	8 secs

- **Step 12**: Now once the data is in Redshift, you can use Quick-sight to visualize it.

SUMMARY:

- This document outlines an end-to-end data engineering project focused on analyzing COVID-19 data using AWS services.
- The project aims to uncover patterns, trends, and risk factors associated with SARS-CoV-2 infection using data mining

techniques.

- **Key takeaways:**
    - The project uses a dataset available on AWS Open Dataset.
    - The architecture involves several AWS services: S3, Crawler, AWS Athena, Glue ETL, Amazon Redshift, and AWS QuickSight, mostly within a VPC for security.
    - The workflow consists of 12 steps, including:
        - Data cleaning and uploading to S3
        - Using Crawler to create metadata tables
        - Analyzing data with Athena
        - Creating a dimensional model using Star Schema
        - Data transformation using Python and Pandas
        - Creating fact and dimension tables
        - Storing processed data in S3
        - Creating schemas and tables in Redshift
        - Copying data from S3 to Redshift
        - Creating a Glue Job
        - Visualizing data with QuickSight
- This project demonstrates a comprehensive approach to handling big data, from ingestion and processing to analysis and visualization, all within the AWS ecosystem.
- It showcases the integration of various AWS services to create a robust data pipeline for COVID-19 data analysis.