I. OLTP

A. Table Creation Script

```
CREATE TABLE IF NOT EXISTS countries (
    id SERIAL PRIMARY KEY,
    location TEXT PRIMARY KEY,
    iso_code TEXT NOT NULL,
    continent TEXT,
    population FLOAT,
    population_density FLOAT,
    median_age FLOAT,
    aged_65_older FLOAT,
    aged_70_older FLOAT,
    gdp_per_capita FLOAT,
    extreme_poverty FLOAT,
    cardiovasc_death_rate FLOAT,
    diabetes_prevalence FLOAT,
    female_smokers FLOAT,
    male_smokers FLOAT,
    handwashing_facilities FLOAT,
    hospital_beds_per_thousand FLOAT,
    life_expectancy FLOAT,
    human_development_index FLOAT
);

CREATE TABLE IF NOT EXISTS cases (
    id SERIAL PRIMARY KEY,
    location TEXT REFERENCES countries(location),
    date TEXT NOT NULL,
    total_cases BIGINT,
    new_cases BIGINT,
    new_cases_smoothed FLOAT,
    total_cases_per_million FLOAT,
    new_cases_per_million FLOAT,
    new_cases_smoothed_per_million FLOAT
);

CREATE TABLE IF NOT EXISTS deaths (
    id SERIAL PRIMARY KEY,
    location TEXT REFERENCES countries(location),
    date TEXT NOT NULL,
    total_deaths BIGINT,
    new_deaths BIGINT,
    new_deaths_smoothed FLOAT,
    total_deaths_per_million FLOAT,
    new_deaths_per_million FLOAT,
    new_deaths_smoothed_per_million FLOAT
);

CREATE TABLE IF NOT EXISTS hospitals (
    id SERIAL PRIMARY KEY,
    location TEXT REFERENCES countries(location),
    date TEXT NOT NULL,
    icu_patients BIGINT,
    icu_patients_per_million FLOAT,
    hosp_patients BIGINT,
    hosp_patients_per_million FLOAT,
    weekly_icu_admissions BIGINT,
    weekly_icu_admissions_per_million FLOAT,
    weekly_hosp_admissions BIGINT,
    weekly_hosp_admissions_per_million FLOAT
);
```

```
CREATE TABLE IF NOT EXISTS tests (
    id SERIAL PRIMARY KEY,
    location TEXT REFERENCES countries(location),
    date TEXT NOT NULL,
    total_tests BIGINT,
    new_tests BIGINT,
    total_tests_per_thousand FLOAT,
    new_tests_per_thousand FLOAT,
    new_tests_smoothed FLOAT,
    new_tests_smoothed_per_thousand FLOAT,
    positive_rate FLOAT,
    tests_per_case FLOAT,
    tests_units TEXT
);

CREATE TABLE IF NOT EXISTS vaccinations (
    id SERIAL PRIMARY KEY,
    location TEXT REFERENCES countries(location),
    date TEXT NOT NULL,
    total_vaccinations BIGINT,
    people_vaccinated BIGINT,
    people_fully_vaccinated BIGINT,
    total_boosters BIGINT,
    new_vaccinations BIGINT,
    new_vaccinations_smoothed FLOAT,
    total_vaccinations_per_hundred FLOAT,
    people_vaccinated_per_hundred FLOAT,
    people_fully_vaccinated_per_hundred FLOAT,
    total_boosters_per_hundred FLOAT,
    new_vaccinations_smoothed_per_million FLOAT,
    new_people_vaccinated_smoothed FLOAT,
    new_people_vaccinated_smoothed_per_hundred FLOAT
);
```

B. List of Tables

In [2]:
```
%sql \dt
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
6 rows affected.
```

Out[2]:

Schema	Name	Type	Owner
public	cases	table	postgres
public	countries	table	postgres
public	deaths	table	postgres
public	hospitals	table	postgres
public	tests	table	postgres
public	vaccinations	table	postgres

C. Schema for each table

Cases

In [17]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'cases';
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
9 rows affected.
```

Out[17]:

column_name	data_type	character_maximum_length	column_default	is_nullable
new_cases_smoothed_per_million	double precision	None	None	YES
new_cases_per_million	double precision	None	None	YES
id	integer	None	nextval('cases_id_seq'::regclass)	NO
total_cases	bigint	None	None	YES
new_cases	bigint	None	None	YES
new_cases_smoothed	double precision	None	None	YES
total_cases_per_million	double precision	None	None	YES
location	text	None	None	YES
date	text	None	None	NO

Countries

In [18]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'countries';
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
18 rows affected.
```

Out[18]:

column_name	data_type	character_maximum_length	column_default	is_nullable
human_development_index	double precision	None	None	YES
handwashing_facilities	double precision	None	None	YES
hospital_beds_per_thousand	double precision	None	None	YES
life_expectancy	double precision	None	None	YES
population	double precision	None	None	YES
population_density	double precision	None	None	YES
median_age	double precision	None	None	YES
aged_65_older	double precision	None	None	YES
aged_70_older	double precision	None	None	YES
gdp_per_capita	double precision	None	None	YES
extreme_poverty	double precision	None	None	YES
cardiovasc_death_rate	double precision	None	None	YES
diabetes_prevalence	double precision	None	None	YES
female_smokers	double precision	None	None	YES
male_smokers	double precision	None	None	YES
iso_code	text	None	None	NO
continent	text	None	None	YES
location	text	None	None	NO

Deaths

In [19]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'deaths';
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
9 rows affected.
```

Out[19]:

column_name	data_type	character_maximum_length	column_default	is_nullable
new_deaths_smoothed_per_million	double precision	None	None	YES
new_deaths_per_million	double precision	None	None	YES
id	integer	None	nextval('deaths_id_seq'::regclass)	NO
total_deaths	bigint	None	None	YES
new_deaths	bigint	None	None	YES
new_deaths_smoothed	double precision	None	None	YES
total_deaths_per_million	double precision	None	None	YES
location	text	None	None	YES
date	text	None	None	NO

Hospitals

In [20]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'hospitals';
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
11 rows affected.
```

Out[20]:

column_name	data_type	character_maximum_length	column_default	is_nullable
weekly_hosp_admissions_per_million	double precision	None	None	YES
weekly_hosp_admissions	bigint	None	None	YES
id	integer	None	nextval('hospitals_id_seq'::regclass)	NO
icu_patients	bigint	None	None	YES
icu_patients_per_million	double precision	None	None	YES
hosp_patients	bigint	None	None	YES
hosp_patients_per_million	double precision	None	None	YES
weekly_icu_admissions	bigint	None	None	YES
weekly_icu_admissions_per_million	double precision	None	None	YES
location	text	None	None	YES
date	text	None	None	NO

Tests

In [21]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'tests';
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
12 rows affected.
```

Out[21]:

column_name	data_type	character_maximum_length	column_default	is_nullable
id	integer	None	nextval('tests_id_seq'::regclass)	NO
new_tests_smoothed_per_thousand	double precision	None	None	YES
positive_rate	double precision	None	None	YES
tests_per_case	double precision	None	None	YES
total_tests	bigint	None	None	YES
new_tests	bigint	None	None	YES
total_tests_per_thousand	double precision	None	None	YES
new_tests_per_thousand	double precision	None	None	YES
new_tests_smoothed	double precision	None	None	YES
location	text	None	None	YES
date	text	None	None	NO
tests_units	text	None	None	YES

Vaccinations

In [22]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'vaccinations';
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
16 rows affected.
```

Out[22]:

column_name	data_type	character_maximum_length	column_default	is_nullable
new_people_vaccinated_smoothed_per_hundred	double precision	None	None	YES
new_people_vaccinated_smoothed	double precision	None	None	YES
id	integer	None	nextval('vaccinations_id_seq'::regclass)	NO
total_vaccinations	bigint	None	None	YES
people_vaccinated	bigint	None	None	YES
people_fully_vaccinated	bigint	None	None	YES
total_boosters	bigint	None	None	YES
new_vaccinations	bigint	None	None	YES
new_vaccinations_smoothed	double precision	None	None	YES
total_vaccinations_per_hundred	double precision	None	None	YES
people_vaccinated_per_hundred	double precision	None	None	YES
people_fully_vaccinated_per_hundred	double precision	None	None	YES
total_boosters_per_hundred	double precision	None	None	YES
new_vaccinations_smoothed_per_million	double precision	None	None	YES
location	text	None	None	YES
date	text	None	None	NO

D. Count of rows for each table

In [11]:
```
%sql select count(0) from countries
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
1 rows affected.
```

Out[11]:

count
244

In [12]:
```
%sql select count(0) from deaths
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
1 rows affected.
```

Out[12]:

count
194031

In [13]:
```
%sql select count(0) from cases
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
1 rows affected.
```

Out[13]:

count
194031

In [14]:
```
%sql select count(0) from hospitals
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
1 rows affected.
```

Out[14]:

count
194031

In [15]:
```
%sql select count(0) from tests
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
1 rows affected.
```

Out[15]:

count
194031

In [16]:
```
%sql select count(0) from vaccinations
```

```
 * postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
1 rows affected.
```

Out[16]:

count
194031

II. OLAP

A. Table Creation Script

```
CREATE TABLE IF NOT EXISTS dim_locations (
    id INT PRIMARY KEY UNIQUE,
    location TEXT UNIQUE,
    iso_code TEXT NOT NULL,
    continent TEXT,
    population FLOAT,
    population_density FLOAT,
    median_age FLOAT,
    aged_65_older FLOAT,
    aged_70_older FLOAT,
    gdp_per_capita FLOAT,
    extreme_poverty FLOAT,
    cardiovasc_death_rate FLOAT,
    diabetes_prevalence FLOAT,
    female_smokers FLOAT,
    male_smokers FLOAT,
    handwashing_facilities FLOAT,
    hospital_beds_per_thousand FLOAT,
    life_expectancy FLOAT,
    human_development_index FLOAT
);

CREATE TABLE IF NOT EXISTS dim_dates (
    date TEXT PRIMARY KEY,
    year INT NOT NULL,
    month INT NOT NULL,
    day INT NOT NULL
);

CREATE TABLE IF NOT EXISTS fact_cases (
    location TEXT REFERENCES dim_locations(location),
    date TEXT REFERENCES dim_dates(date),
    total_cases BIGINT,
    new_cases BIGINT,
    new_cases_smoothed FLOAT,
    total_cases_per_million FLOAT,
    new_cases_per_million FLOAT,
    new_cases_smoothed_per_million FLOAT,
    primary key(location, date)

);

CREATE TABLE IF NOT EXISTS fact_deaths (
    location TEXT REFERENCES dim_locations(location),
    date TEXT REFERENCES dim_dates(date),
    total_deaths BIGINT,
    new_deaths BIGINT,
    new_deaths_smoothed FLOAT,
    total_deaths_per_million FLOAT,
    new_deaths_per_million FLOAT,
    new_deaths_smoothed_per_million FLOAT,
    primary key(location, date)
);

CREATE TABLE IF NOT EXISTS fact_hospitals (
    location TEXT REFERENCES dim_locations(location),
    date TEXT REFERENCES dim_dates(date),
    icu_patients BIGINT,
    icu_patients_per_million FLOAT,
```

```
    hosp_patients BIGINT,
    hosp_patients_per_million FLOAT,
    weekly_icu_admissions BIGINT,
    weekly_icu_admissions_per_million FLOAT,
    weekly_hosp_admissions BIGINT,
    weekly_hosp_admissions_per_million FLOAT,
    primary key(location, date)
);

CREATE TABLE IF NOT EXISTS fact_tests (
    location TEXT REFERENCES dim_locations(location),
    date TEXT REFERENCES dim_dates(date),
    total_tests BIGINT,
    new_tests BIGINT,
    total_tests_per_thousand FLOAT,
    new_tests_per_thousand FLOAT,
    new_tests_smoothed FLOAT,
    new_tests_smoothed_per_thousand FLOAT,
    positive_rate FLOAT,
    tests_per_case FLOAT,
    tests_units TEXT,
    primary key(location, date)
);

CREATE TABLE IF NOT EXISTS fact_vaccinations (
    location TEXT REFERENCES dim_locations(location),
    date TEXT REFERENCES dim_dates(date),
    total_vaccinations BIGINT,
    people_vaccinated BIGINT,
    people_fully_vaccinated BIGINT,
    total_boosters BIGINT,
    new_vaccinations BIGINT,
    new_vaccinations_smoothed FLOAT,
    total_vaccinations_per_hundred FLOAT,
    people_vaccinated_per_hundred FLOAT,
    people_fully_vaccinated_per_hundred FLOAT,
    total_boosters_per_hundred FLOAT,
    new_vaccinations_smoothed_per_million FLOAT,
    new_people_vaccinated_smoothed FLOAT,
    new_people_vaccinated_smoothed_per_hundred FLOAT,
    primary key(location, date)
);
```

B. List of Tables

In [39]:
```
%sql \d
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
   postgresql://postgres:***@covid.cunofrcjnuto.us-east-1.rds.amazonaws.com/postgres
7 rows affected.
```

Out[39]:

schema	name	type	owner
public	dim_dates	table	admin
public	dim_locations	table	admin
public	fact_cases	table	admin
public	fact_deaths	table	admin
public	fact_hospitals	table	admin
public	fact_tests	table	admin
public	fact_vaccinations	table	admin

C. Schema for Each Table

dim_dates

In [5]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'dim_dates';
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
4 rows affected.
```

Out[5]:

column_name	data_type	character_maximum_length	column_default	is_nullable
day	integer	None	None	NO
month	integer	None	None	NO
year	integer	None	None	NO
date	character varying	256	None	NO

dim_locations

In [6]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'dim_locations';
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
18 rows affected.
```

Out[6]:

column_name	data_type	character_maximum_length	column_default	is_nullable
human_development_index	double precision	None	None	YES
life_expectancy	double precision	None	None	YES
hospital_beds_per_thousand	double precision	None	None	YES
handwashing_facilities	double precision	None	None	YES
male_smokers	double precision	None	None	YES
female_smokers	double precision	None	None	YES
diabetes_prevalence	double precision	None	None	YES
cardiovasc_death_rate	double precision	None	None	YES
extreme_poverty	double precision	None	None	YES
gdp_per_capita	double precision	None	None	YES
aged_70_older	double precision	None	None	YES
aged_65_older	double precision	None	None	YES
median_age	double precision	None	None	YES
population_density	double precision	None	None	YES
population	double precision	None	None	YES
continent	character varying	256	None	YES
iso_code	character varying	256	None	NO
location	character varying	256	None	NO

fact_cases

In [7]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'fact_cases';
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
8 rows affected.
```

Out[7]:

column_name	data_type	character_maximum_length	column_default	is_nullable
new_cases	bigint	None	None	YES
total_cases	bigint	None	None	YES
new_cases_smoothed_per_million	double precision	None	None	YES
new_cases_per_million	double precision	None	None	YES
total_cases_per_million	double precision	None	None	YES
new_cases_smoothed	double precision	None	None	YES
date	character varying	256	None	NO
location	character varying	256	None	NO

fact_deaths

In [8]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'fact_deaths';
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
8 rows affected.
```

Out[8]:

column_name	data_type	character_maximum_length	column_default	is_nullable
new_deaths	bigint	None	None	YES
total_deaths	bigint	None	None	YES
new_deaths_smoothed_per_million	double precision	None	None	YES
new_deaths_per_million	double precision	None	None	YES
total_deaths_per_million	double precision	None	None	YES
new_deaths_smoothed	double precision	None	None	YES
date	character varying	256	None	NO
location	character varying	256	None	NO

fact_hospitals

In [9]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'fact_hospitals';
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
10 rows affected.
```

Out[9]:

column_name	data_type	character_maximum_length	column_default	is_nullable
weekly_hosp_admissions	bigint	None	None	YES
weekly_icu_admissions	bigint	None	None	YES
hosp_patients	bigint	None	None	YES
icu_patients	bigint	None	None	YES
weekly_hosp_admissions_per_million	double precision	None	None	YES
weekly_icu_admissions_per_million	double precision	None	None	YES
hosp_patients_per_million	double precision	None	None	YES
icu_patients_per_million	double precision	None	None	YES
date	character varying	256	None	NO
location	character varying	256	None	NO

fact_tests

In [10]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'fact_tests';
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
11 rows affected.
```

Out[10]:

column_name	data_type	character_maximum_length	column_default	is_nullable
new_tests	bigint	None	None	YES
total_tests	bigint	None	None	YES
tests_per_case	double precision	None	None	YES
positive_rate	double precision	None	None	YES
new_tests_smoothed_per_thousand	double precision	None	None	YES
new_tests_smoothed	double precision	None	None	YES
new_tests_per_thousand	double precision	None	None	YES
total_tests_per_thousand	double precision	None	None	YES
tests_units	character varying	256	None	YES
date	character varying	256	None	NO
location	character varying	256	None	NO

fact_vaccinations

In [11]:
```
%%sql
select column_name, data_type, character_maximum_length, column_default, is_nullable
from INFORMATION_SCHEMA.COLUMNS where table_name = 'fact_vaccinations';
```

```
 * postgresql://admin:***@final-project-olap.cggw3rhsilm3.us-east-1.redshift.amazonaws.com:5439/dev
16 rows affected.
```

Out[11]:

column_name	data_type	character_maximum_length	column_default	is_nullable
new_vaccinations	bigint	None	None	YES
total_boosters	bigint	None	None	YES
people_fully_vaccinated	bigint	None	None	YES
people_vaccinated	bigint	None	None	YES
total_vaccinations	bigint	None	None	YES
id	integer	None	"identity"(105779, 0, '1,1'::text)	NO
new_people_vaccinated_smoothed_per_hundred	double precision	None	None	YES
new_people_vaccinated_smoothed	double precision	None	None	YES
new_vaccinations_smoothed_per_million	double precision	None	None	YES
total_boosters_per_hundred	double precision	None	None	YES
people_fully_vaccinated_per_hundred	double precision	None	None	YES
people_vaccinated_per_hundred	double precision	None	None	YES
total_vaccinations_per_hundred	double precision	None	None	YES
new_vaccinations_smoothed	double precision	None	None	YES
date	character varying	256	None	YES
location	character varying	256	None	YES

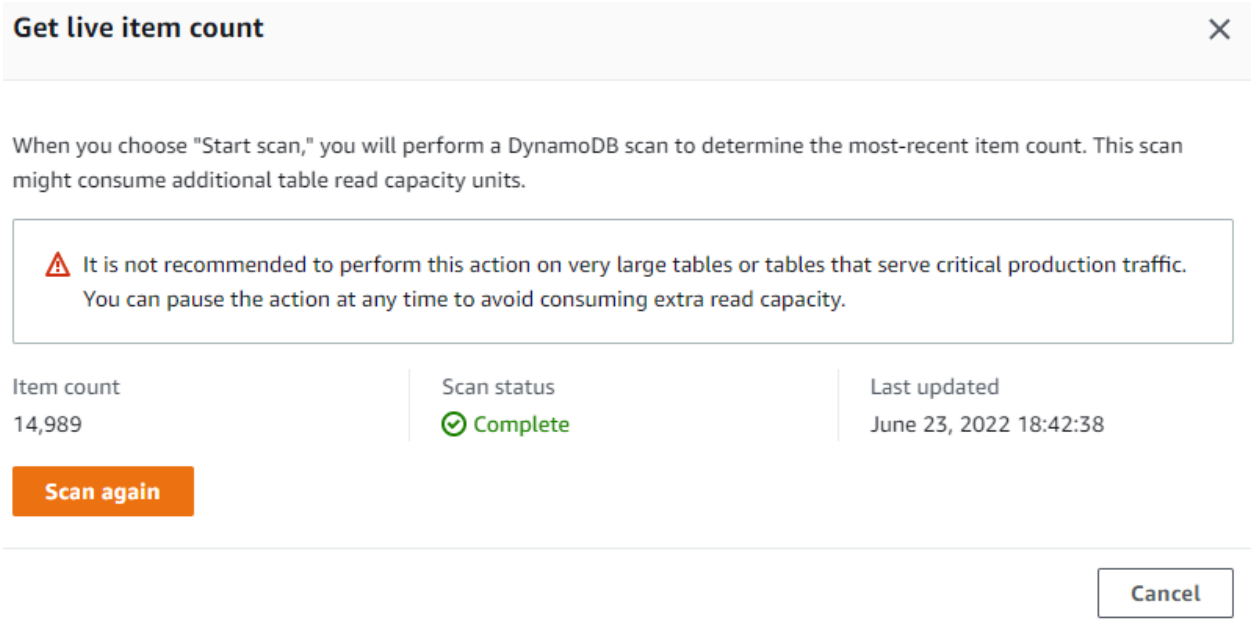
III. NoSQL

A. Python code for adding the items (See tweet_data.py)

```
def insert_to_dynamo():
    s3 = boto3.resource('s3')
    my_bucket = s3.Bucket(s3_bucket)

    s3_client = boto3.client('s3',
                             aws_access_key_id=aws_access_key_id,
                             aws_secret_access_key=aws_secret_access_key)

    comprehend = boto3.client('comprehend',
        aws_access_key_id=aws_access_key_id,
        aws_secret_access_key=aws_secret_access_key)



    for my_bucket_object in my_bucket.objects.all():
        if re.match(f'{landing_path}{datetime.now().strftime("%Y-%m-%d")}.json',
                    my_bucket_object.key):

            s3_response_object = s3_client.get_object(Bucket=s3_bucket,
                                                      Key=my_bucket_object.key)

            object_content = s3_response_object['Body']
            df = pd.read_json(object_content, lines=True)
            df['date'] = df.date.dt.strftime("%Y-%m-%d")
            df_tweets = df

            client = boto3.client('dynamodb')

            # replace nan to None for easier processing
            df_tweets = df_tweets.replace(np.nan, None, regex=False)
            # remove links
            df_tweets['content'] = df_tweets.content.str.replace(r'http.*?\b', '', regex=True)

            # remove hashtags; should be captured by hashtag feature
            df_tweets['content'] = df_tweets.content.str.replace(r'#.*?\b', '', regex=True)

            # remove non-alphanumeric for compatability with dynamodb
            df_tweets['content'] = df_tweets.content.str.replace(r'[^A-Za-z0-9 ]+', '', regex=True)
            df_tweets['tokenized_content'] = df_tweets.content.str.split()

            print('start sentiment analysis... ')
            df_sentiment = pd.json_normalize(df_tweets.content.apply(lambda x:
                                                 comprehend.detect_sentiment(Text=x,
                                                                             LanguageCode='en')))

            print('end sentiment analysis... ')

            df_dynamo = pd.concat([df_tweets.reset_index(drop=True),
                                   df_sentiment], axis=1)

            # get only relevant features
            df_dynamo = df_dynamo.loc[:, ['id', 'date', 'content', 'replyCount',
                                          'retweetCount', 'likeCount', 'quoteCount',
                                          'hashtags', 'Sentiment', 'SentimentScore.Positive',
                                           'SentimentScore.Negative', 'SentimentScore.Neutral',
                                           'SentimentScore.Mixed', 'tokenized_content'
                                    ]].drop_duplicates(subset=['id', 'date']).to_dict(orient='records')

            print('start insert dynamodb... ')
            for row in df_dynamo:
                try:
                    client.execute_statement(
                        Statement=f"""
                        INSERT INTO covid_tweets VALUE {str(row)}
                        """
                    )
                except:
                    continue

            print('end insert dynamodb... ')
```

B. Output of the contents of the table

Count: 14989



Columns

```
import boto3
import pandas as pd
```

```
client = boto3.client('dynamodb')
```

```
output = client.execute_statement(
Statement="""
SELECT * FROM covid_tweets
"""
)['Items']

output = pd.DataFrame(output)
```

```
display(output.columns.tolist())
```

```
['content',
 'retweetCount',
 'hashtags',
 'tokenized_content',
 'SentimentScore.Negative',
 'Sentiment',
 'SentimentScore.Positive',
 'likeCount',
 'SentimentScore.Mixed',
 'date',
 'SentimentScore.Neutral',
 'replyCount',
 'id',
 'quoteCount']
```

covid_tweets sample table items

Items returned (300)

Actions ▼ | Create item

1 ...

date	id	content	hashtags	likeCount	quoteCount	replyCount	retweet
2022-05-19	15271696470275604...	Coronavirus...	[{ "S" : "cor...	0	0	0	0
2022-05-19	15271841954995650...	Coronavirus...	[{ "S" : "cor...	0	0	0	1
2022-05-19	15272004084768317...	Coronavirus...	[{ "S" : "cor...	0	0	0	0
2022-05-19	15272087284575109...	Coronavirus...	[{ "S" : "cor...	0	0	0	0
2022-05-19	15274139090486763...	1028146 d...	[{ "S" : "Re...	2	0	0	1
2022-05-18	15267143822515896...	COVID19us...	[{ "S" : "CO...	0	1	0	0
2022-05-18	15268056831312732...	Coronavirus...	[{ "S" : "cor...	0	0	1	0
2022-05-18	15268216617811312...	Coronavirus...	[{ "S" : "cor...	0	0	1	0
2022-05-18	15268378109213736...	Coronavirus...	[{ "S" : "cor...	0	0	0	0
2022-05-18	15268448280588861...	Coronavirus...	[{ "S" : "cor...	0	0	0	0
2022-05-18	15270525684085063...	1027616 d...	[{ "S" : "Re...	2	0	0	1
2022-05-18	15270747808007618...	COVID19us...	[{ "S" : "CO...	0	0	0	0
2022-05-14	15252865614447697...	COVID19us...	[{ "S" : "CO...	0	0	0	0
2022-05-14	15253590937409781...	Coronavirus...	[{ "S" : "cor...	0	0	0	0
2022-05-14	15253716678113812...	Coronavirus...	[{ "S" : "cor...	0	0	0	0

C. Screenshot of each tab of the console of the DynamoDB table

DynamoDB > Tables > covid_tweets

Tables (2)

Any table tag

Find tables by table name

1

- covid_tweets
- sessions

covid_tweets

Actions ▼ | Explore table items

Overview | Indexes | Monitor | Global tables | Backups | Exports and streams | Additional settings

General information

Partition key	Sort key	Capacity mode	Table status
date (String)	id (Number)	On-demand	Active No active alarms

▶ Additional info

Items summary

DynamoDB updates the following information approximately every six hours.

Get live item count

Item count	Table size	Average item size
14,989	10.1 megabytes	676.85 bytes

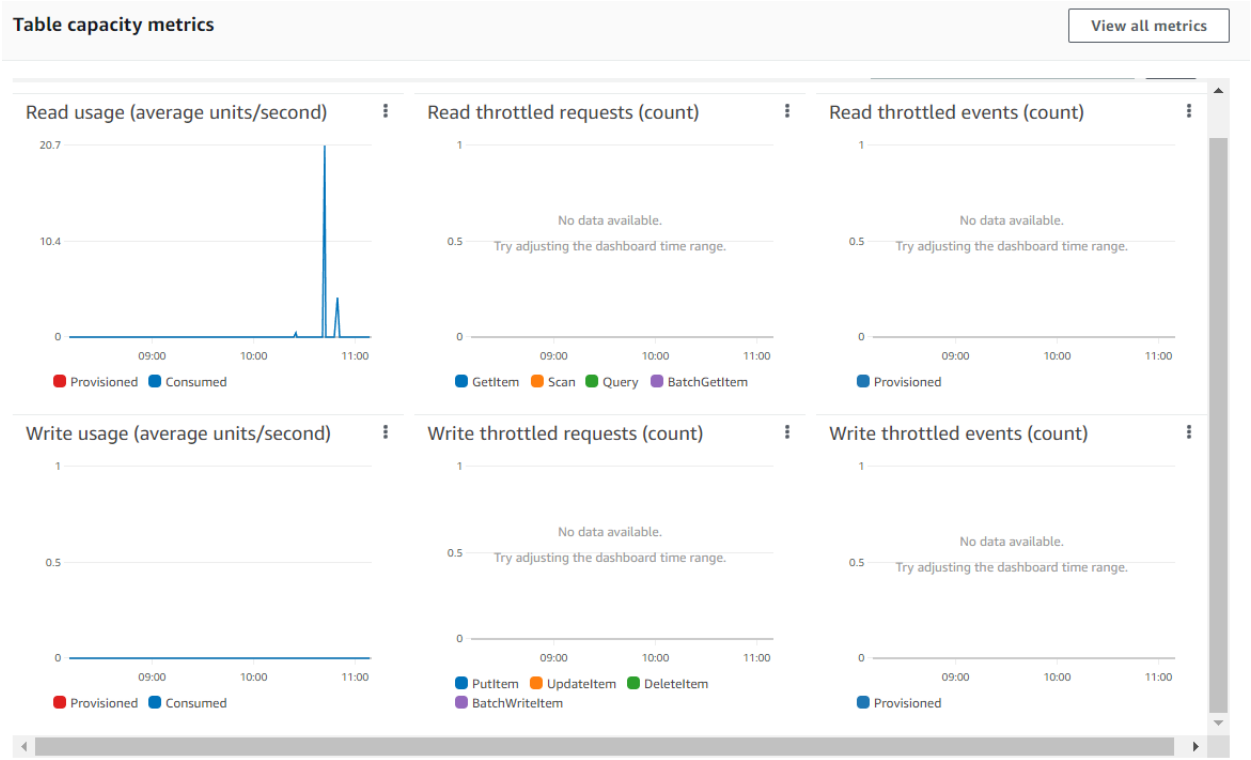
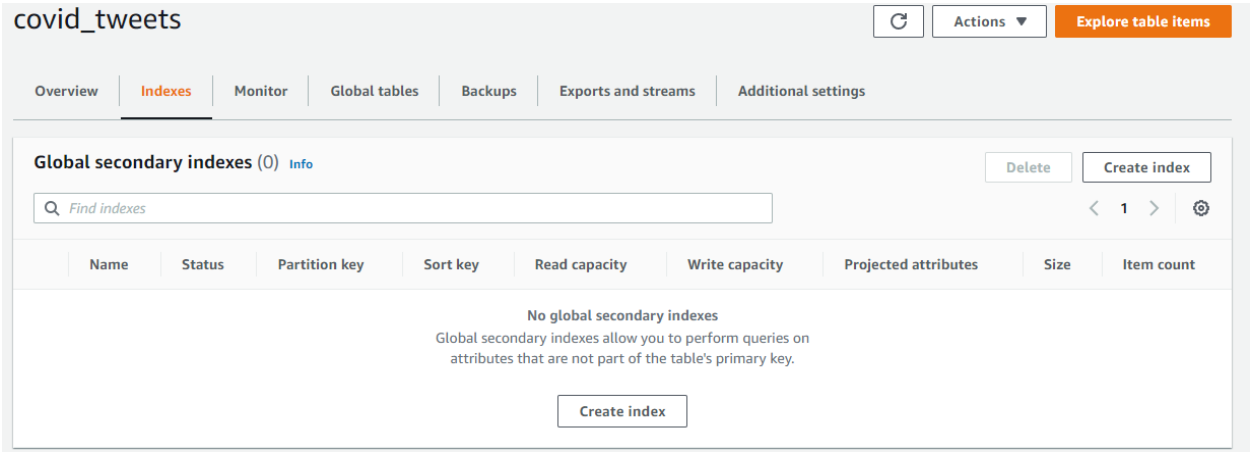


Table capacity metrics
View all metrics
Read usage (average units/second)
20.7
10.4
0
09:00
10:00
11:00
Provisioned
Consumed
Read throttled requests (count)
1
0.5
0
No data available.
Try adjusting the dashboard time range.
09:00
10:00
11:00
GetItem
Scan
Query
BatchGetItem
Read throttled events (count)
1
0.5
0
No data available.
Try adjusting the dashboard time range.
09:00
10:00
11:00
Provisioned
Write usage (average units/second)
1
0.5
0
09:00
10:00
11:00
Provisioned
Consumed
Write throttled requests (count)
1
0.5
0
No data available.
Try adjusting the dashboard time range.
09:00
10:00
11:00
PutItem
UpdateItem
DeleteItem
BatchWriteItem
Write throttled events (count)
1
0.5
0
No data available.
Try adjusting the dashboard time range.
09:00
10:00
11:00
Provisioned

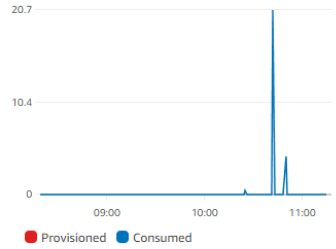

covid_tweets
Actions
Explore table items
Overview
Indexes
Monitor
Global tables
Backups
Exports and streams
Additional settings
Global secondary indexes (0) Info
Delete
Create index
Find indexes
1
Name
Status
Partition key
Sort key
Read capacity
Write capacity
Projected attributes
Size
Item count
No global secondary indexes
Global secondary indexes allow you to perform queries on attributes that are not part of the table's primary key.
Create index

CloudWatch metrics

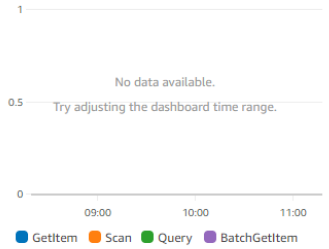
View all in CloudWatch

Table: covid_tweets

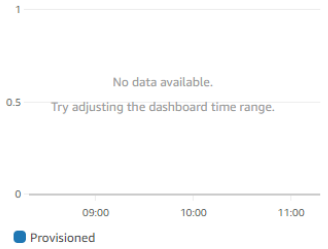
Read usage (average units/second)



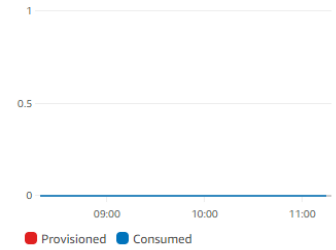
Read throttled requests (count)



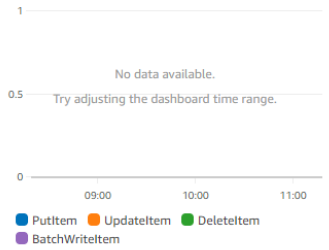
Read throttled events (count)



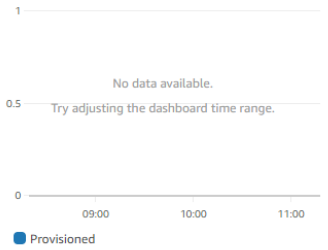
Write usage (average units/second)



Write throttled requests (count)

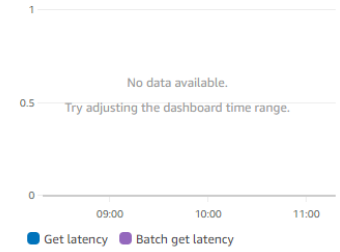


Write throttled events (count)

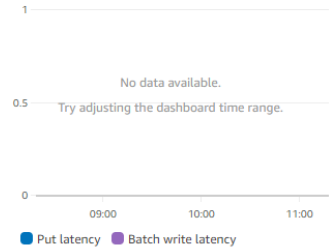


Latency

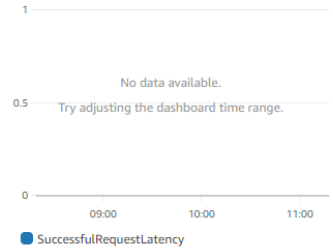
Get latency (milliseconds)



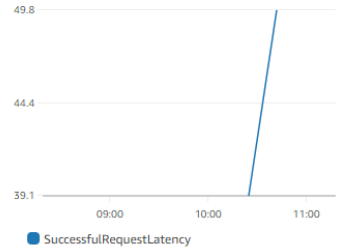
Put latency (milliseconds)



Query latency (milliseconds)



Scan latency (milliseconds)

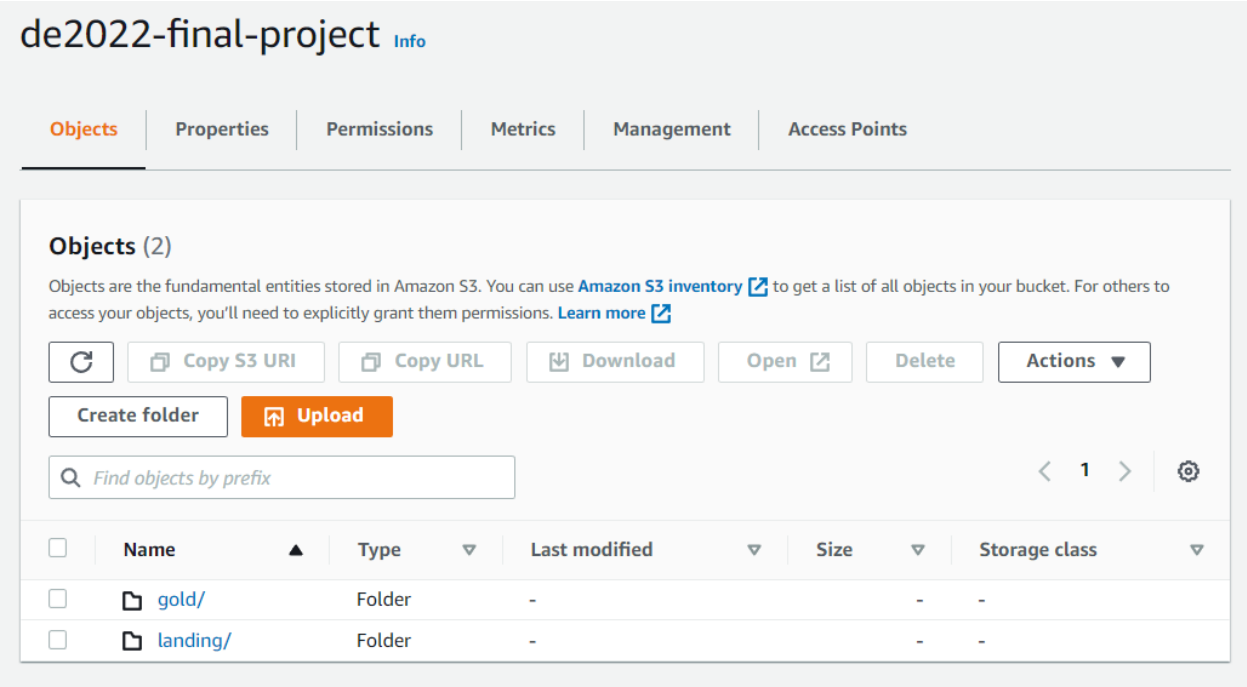


Time to Live (TTL)

TTL deleted items (count)

IV. Data Lake

A. Document detailing the S3 buckets



B. Directory Hierarchy

```
(base) ubuntu@ip-172-31-84-45:~$ aws s3 ls  --human-readable --summarize --recursive s3://de2022-final-project/
2022-06-15 08:09:51    0 Bytes gold/
2022-06-15 15:31:20    0 Bytes gold/nosql/
2022-06-23 11:54:57    1.8 MiB gold/nosql/tweets_db.json
2022-06-15 09:50:49    0 Bytes gold/olap/
2022-06-23 07:57:55   18.5 KiB gold/olap/dim_dates_000
2022-06-23 07:57:51   34.7 KiB gold/olap/dim_locations_000
2022-06-23 07:58:04    1.1 MiB gold/olap/fact_cases_000
2022-06-23 07:58:08  899.0 KiB gold/olap/fact_deaths_000
2022-06-23 07:58:13   32.2 KiB gold/olap/fact_hospitals_000
2022-06-23 07:58:21  366.5 KiB gold/olap/fact_tests_000
2022-06-23 07:58:25  552.5 KiB gold/olap/fact_vaccinations_000
2022-06-15 09:04:13    0 Bytes gold/oltp/
2022-06-23 07:57:01   11.6 MiB gold/oltp/cases.csv
2022-06-23 07:57:10   23.7 KiB gold/oltp/countries.csv
2022-06-23 07:57:03    9.7 MiB gold/oltp/deaths.csv
2022-06-23 07:57:05    6.2 MiB gold/oltp/hospitals.csv
2022-06-23 07:57:07   11.0 MiB gold/oltp/tests.csv
2022-06-23 07:57:08   11.3 MiB gold/oltp/vaccinations.csv
2022-06-15 04:38:26    0 Bytes landing/
2022-06-15 07:18:13    0 Bytes landing/owid-data/
2022-06-23 07:56:47   53.3 MiB landing/owid-data/owid-covid-data.csv
2022-06-15 15:33:55    0 Bytes landing/tweet-data/
2022-06-16 06:19:48  297.0 KiB landing/tweet-data/2020-06-01.json
2022-06-16 06:19:50  236.6 KiB landing/tweet-data/2020-06-02.json
2022-06-16 06:19:51  165.4 KiB landing/tweet-data/2020-06-03.json
2022-06-16 06:19:53  141.8 KiB landing/tweet-data/2020-06-04.json
2022-06-16 06:19:54  152.3 KiB landing/tweet-data/2020-06-05.json
2022-06-16 06:19:55  127.6 KiB landing/tweet-data/2020-06-06.json
2022-06-16 06:19:56  104.1 KiB landing/tweet-data/2020-06-07.json
2022-06-16 06:19:57   92.4 KiB landing/tweet-data/2020-06-08.json
2022-06-16 06:19:59  163.6 KiB landing/tweet-data/2020-06-09.json
2022-06-16 06:20:00  119.7 KiB landing/tweet-data/2020-06-10.json
2022-06-16 06:20:02  233.0 KiB landing/tweet-data/2020-06-11.json
2022-06-16 06:20:03  125.6 KiB landing/tweet-data/2020-06-12.json
2022-06-16 06:20:04  150.6 KiB landing/tweet-data/2020-06-13.json
2022-06-16 06:20:06  218.0 KiB landing/tweet-data/2020-06-14.json
2022-06-16 06:20:07  143.4 KiB landing/tweet-data/2020-06-15.json
2022-06-16 06:20:09  210.2 KiB landing/tweet-data/2020-06-16.json
2022-06-16 06:20:10  199.0 KiB landing/tweet-data/2020-06-17.json
2022-06-16 06:20:12  221.1 KiB landing/tweet-data/2020-06-18.json
2022-06-16 06:20:13  168.4 KiB landing/tweet-data/2020-06-19.json
2022-06-16 06:20:15  170.4 KiB landing/tweet-data/2020-06-20.json
2022-06-16 06:20:17  248.3 KiB landing/tweet-data/2020-06-21.json
2022-06-16 06:20:19  253.7 KiB landing/tweet-data/2020-06-22.json
2022-06-16 06:20:21  300.7 KiB landing/tweet-data/2020-06-23.json
2022-06-16 06:20:22  273.7 KiB landing/tweet-data/2020-06-24.json
2022-06-16 06:20:25  326.9 KiB landing/tweet-data/2020-06-25.json
2022-06-16 06:20:27  425.1 KiB landing/tweet-data/2020-06-26.json
2022-06-16 06:20:30  305.8 KiB landing/tweet-data/2020-06-27.json
2022-06-16 06:20:33  489.7 KiB landing/tweet-data/2020-06-28.json
2022-06-16 06:20:37  576.6 KiB landing/tweet-data/2020-06-29.json
2022-06-16 06:20:41  653.5 KiB landing/tweet-data/2020-06-30.json
2022-06-16 06:20:45  587.3 KiB landing/tweet-data/2020-07-01.json
```

```
2022-06-16 06:30:27   14.3 KiB landing/tweet-data/2022-04-30.json
2022-06-16 06:30:27   22.0 KiB landing/tweet-data/2022-05-01.json
2022-06-16 06:30:28   23.2 KiB landing/tweet-data/2022-05-02.json
2022-06-16 06:30:29   19.0 KiB landing/tweet-data/2022-05-03.json
2022-06-16 06:30:29   21.4 KiB landing/tweet-data/2022-05-04.json
2022-06-16 06:30:30   27.4 KiB landing/tweet-data/2022-05-05.json
2022-06-16 06:30:31   29.9 KiB landing/tweet-data/2022-05-06.json
2022-06-16 06:30:31   16.0 KiB landing/tweet-data/2022-05-07.json
2022-06-16 06:30:32   18.6 KiB landing/tweet-data/2022-05-08.json
2022-06-16 06:30:32   21.3 KiB landing/tweet-data/2022-05-09.json
2022-06-16 06:30:33   16.3 KiB landing/tweet-data/2022-05-10.json
2022-06-16 06:30:34   18.8 KiB landing/tweet-data/2022-05-11.json
2022-06-16 06:30:34   18.5 KiB landing/tweet-data/2022-05-12.json
2022-06-16 06:30:35   24.4 KiB landing/tweet-data/2022-05-13.json
2022-06-16 06:30:35   16.3 KiB landing/tweet-data/2022-05-14.json
2022-06-16 06:30:36   19.8 KiB landing/tweet-data/2022-05-15.json
2022-06-16 06:30:37   16.3 KiB landing/tweet-data/2022-05-16.json
2022-06-16 06:30:37   18.7 KiB landing/tweet-data/2022-05-17.json
2022-06-16 06:30:38   19.2 KiB landing/tweet-data/2022-05-18.json
2022-06-16 06:30:38   13.4 KiB landing/tweet-data/2022-05-19.json
2022-06-16 06:30:39   21.6 KiB landing/tweet-data/2022-05-20.json
2022-06-16 06:30:39   19.2 KiB landing/tweet-data/2022-05-21.json
2022-06-16 06:30:40   18.9 KiB landing/tweet-data/2022-05-22.json
2022-06-16 06:30:41   22.3 KiB landing/tweet-data/2022-05-23.json
2022-06-16 06:30:41   16.3 KiB landing/tweet-data/2022-05-24.json
2022-06-16 06:30:42   24.0 KiB landing/tweet-data/2022-05-25.json
2022-06-16 06:30:42   13.4 KiB landing/tweet-data/2022-05-26.json
2022-06-16 06:30:43   16.3 KiB landing/tweet-data/2022-05-27.json
2022-06-16 06:30:44   16.3 KiB landing/tweet-data/2022-05-28.json
2022-06-16 06:30:44   20.5 KiB landing/tweet-data/2022-05-29.json
2022-06-16 06:30:45   18.9 KiB landing/tweet-data/2022-05-30.json
2022-06-16 06:30:45   23.1 KiB landing/tweet-data/2022-05-31.json
2022-06-16 06:30:46   24.6 KiB landing/tweet-data/2022-06-01.json
2022-06-16 06:30:47   21.4 KiB landing/tweet-data/2022-06-02.json
2022-06-16 06:30:47   21.5 KiB landing/tweet-data/2022-06-03.json
2022-06-16 06:30:48   16.3 KiB landing/tweet-data/2022-06-04.json
2022-06-16 06:30:49   16.3 KiB landing/tweet-data/2022-06-05.json
2022-06-16 06:30:49   18.5 KiB landing/tweet-data/2022-06-06.json
2022-06-16 06:30:50   16.3 KiB landing/tweet-data/2022-06-07.json
2022-06-16 06:30:51   16.3 KiB landing/tweet-data/2022-06-08.json
2022-06-16 06:30:51   26.4 KiB landing/tweet-data/2022-06-09.json
2022-06-16 06:30:52   19.2 KiB landing/tweet-data/2022-06-10.json
2022-06-16 06:30:53   16.3 KiB landing/tweet-data/2022-06-11.json
2022-06-16 06:30:54   21.3 KiB landing/tweet-data/2022-06-12.json
2022-06-16 06:30:54   16.2 KiB landing/tweet-data/2022-06-13.json
2022-06-16 06:30:55   24.7 KiB landing/tweet-data/2022-06-14.json
2022-06-16 06:30:56   23.8 KiB landing/tweet-data/2022-06-15.json
2022-06-18 08:08:12   18.8 KiB landing/tweet-data/2022-06-17.json
2022-06-20 06:39:24   21.9 KiB landing/tweet-data/2022-06-19.json
2022-06-23 11:54:33   23.9 KiB landing/tweet-data/2022-06-22.json

Total Objects: 770
   Total Size: 155.5 MiB
```

C. Policies

GiveFullAccessToGoldZone

defines the AWS permissions that you can assign to a user, group, or role. You can create and ed

l editor | JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "s3-object-lambda:*"
            ],
            "Resource": "arn:aws:s3:::de2022-final-project/gold/*"
        }
    ]
}
```

GiveFullAccessToLandingZone

defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a po

l editor | JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "s3-object-lambda:*"
            ],
            "Resource": "arn:aws:s3:::de2022-final-project/landing/*"
        }
    ]
}
```

D. User Groups

data-scientists

Delete

Summary

Edit

User group name	Creation time	ARN
data-scientists	June 23, 2022, 21:03 (UTC+08:00)	arn:aws:iam::960715734278:group/data-scientists

Users | Permissions | Access Advisor

Permissions policies (1) Info
You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Filter policies by property or policy name and press enter

1

Policy name	Type	Description
GiveFullAccessToGoldZone	Customer managed	

business-analysts

Delete

Summary

Edit

User group name	Creation time	ARN
business-analysts	June 23, 2022, 21:04 (UTC+08:00)	arn:aws:iam::960715734278:group/business-analysts

Users | Permissions | Access Advisor

Permissions policies (1) Info
You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Filter policies by property or policy name and press enter

1

Policy name	Type	Description
GiveFullAccessToGoldZone	Customer managed	

data-engineers

Delete

Summary

Edit

User group name	Creation time	ARN
data-engineers	June 23, 2022, 21:05 (UTC+08:00)	arn:aws:iam::960715734278:group/data-engineers

Users | Permissions | Access Advisor

Permissions policies (1) Info
You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Filter policies by property or policy name and press enter

1

Policy name	Type	Description
GiveFullAccessToLandingZone	Customer managed	

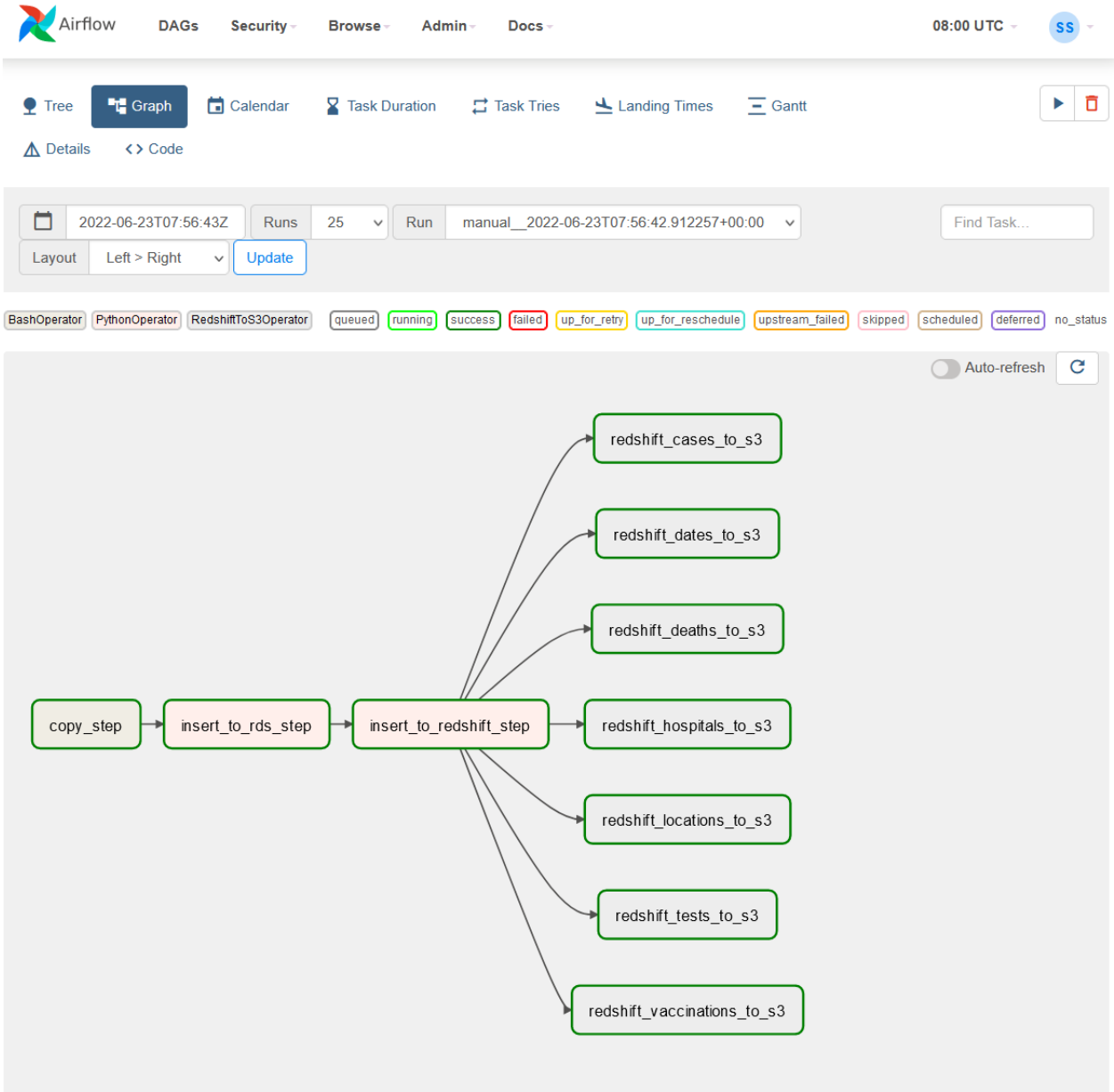
V. ETL Jobs

A. Dag for extracting COVID data and updating the OLTP and OLAP databases

1. Airflow Script

Please see attached file extract_covid_data.py

2. DAG Details Screenshots



Airflow DAGs Security Browse Admin Docs 08:06 UTC SS

DAG: extract_covid_data

Schedule: 1 day, 0:00:00 Next Run: 2022-06-23, 08:06:26

Tree Graph Calendar Task Duration Task Tries Landing Times Gantt

Details Code

DAG Details

failed 34 queued 3 removed 2 scheduled 7 success 53 upstream_failed 53

Schedule Interval	1 day, 0:00:00
Catchup	True
Start Date	2022-06-23, 08:06:26
End Date	None
Max Active Runs	3 / 16
Concurrency	16
Default Args	{}
Tasks Count	10
Task IDs	['copy_step', 'insert_to_rds_step', 'insert_to_redshift_step', 'redshift_locations_to_s3', 'redshift_dates_to_s3', 'redshift_cases_to_s3', 'redshift_deaths_to_s3', 'redshift_hospitals_to_s3', 'redshift_tests_to_s3', 'redshift_vaccinations_to_s3']
Relative file location	extract_covid_data.py
Owner	airflow
DAG Run Timeout	None
Tags	None

Airflow DAGs Security Browse Admin Docs 08:07 UTC SS

DAG: extract_covid_data

Schedule: 1 day, 0:00:00 Next Run: 2022-06-23, 08:07:02

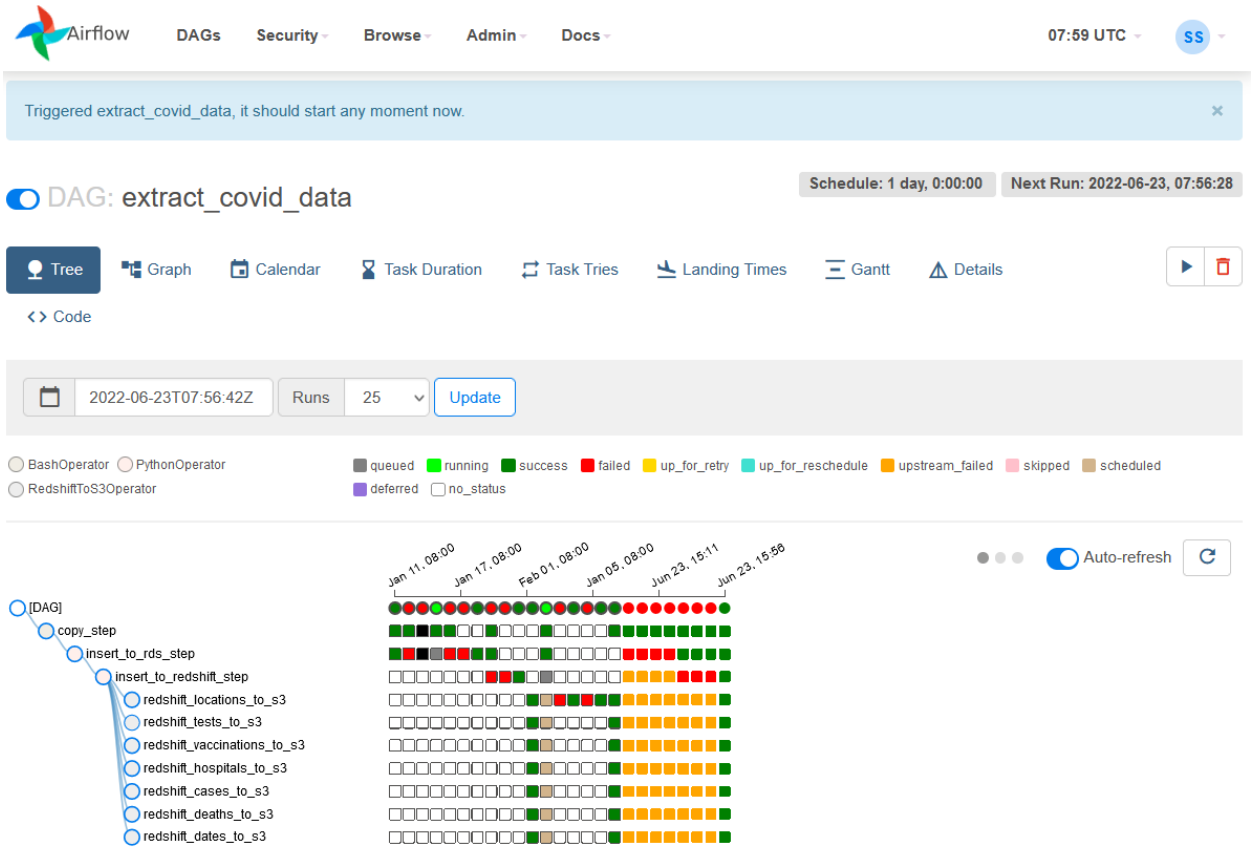
Tree Graph Calendar Task Duration Task Tries Landing Times Gantt

Details Code

Toggle Wrap

```python
import datetime
import os
import pandas as pd
import s3fs
import sqlite3


from airflow import DAG
from airflow.decorators import task
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator
from airflow.providers.postgres.operators.postgres import PostgresOperator
from airflow.providers.postgres.hooks.postgres import PostgresHook
from airflow.providers.amazon.aws.operators.rds import (RdsCreateDbSnapshotOperator , RdsStartExportTaskOperator)
from airflow.providers.amazon.aws.transfers.redshift_to_s3 import RedshiftToS3Operator
from airflow.providers.amazon.aws.hooks.redshift_sql import RedshiftSQLHook
from dateutil.relativedelta import relativedelta

from smart_open import smart_open
from sqlalchemy import create_engine


with DAG(
    dag_id='extract_covid_data',
    start_date=datetime.datetime.now()
) as dag:

    #### Step 1: Copy from OWID to S3
    copy_step = BashOperator(
        task_id='copy_step',
        bash_command=('curl "https://covid.ourworldindata.org/data/owid-covid-data.csv" '
                      '| aws s3 cp - s3://de2022-final-project/landing/owid-data/owid-covid-data.csv')
    )


#     #### Step 2: Insert CSV data to RDS
    def insert_to_rds():

        # Connect to postgres using pre-defined connection
        conn = PostgresHook(postgres_conn_id='final_project_rds')
        engine = conn.get_sqlalchemy_engine()
```

3. Success Runs

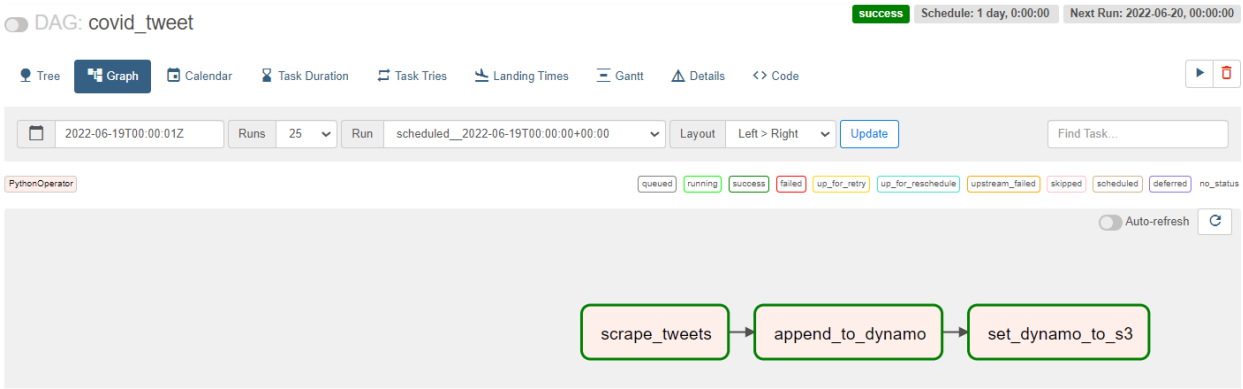


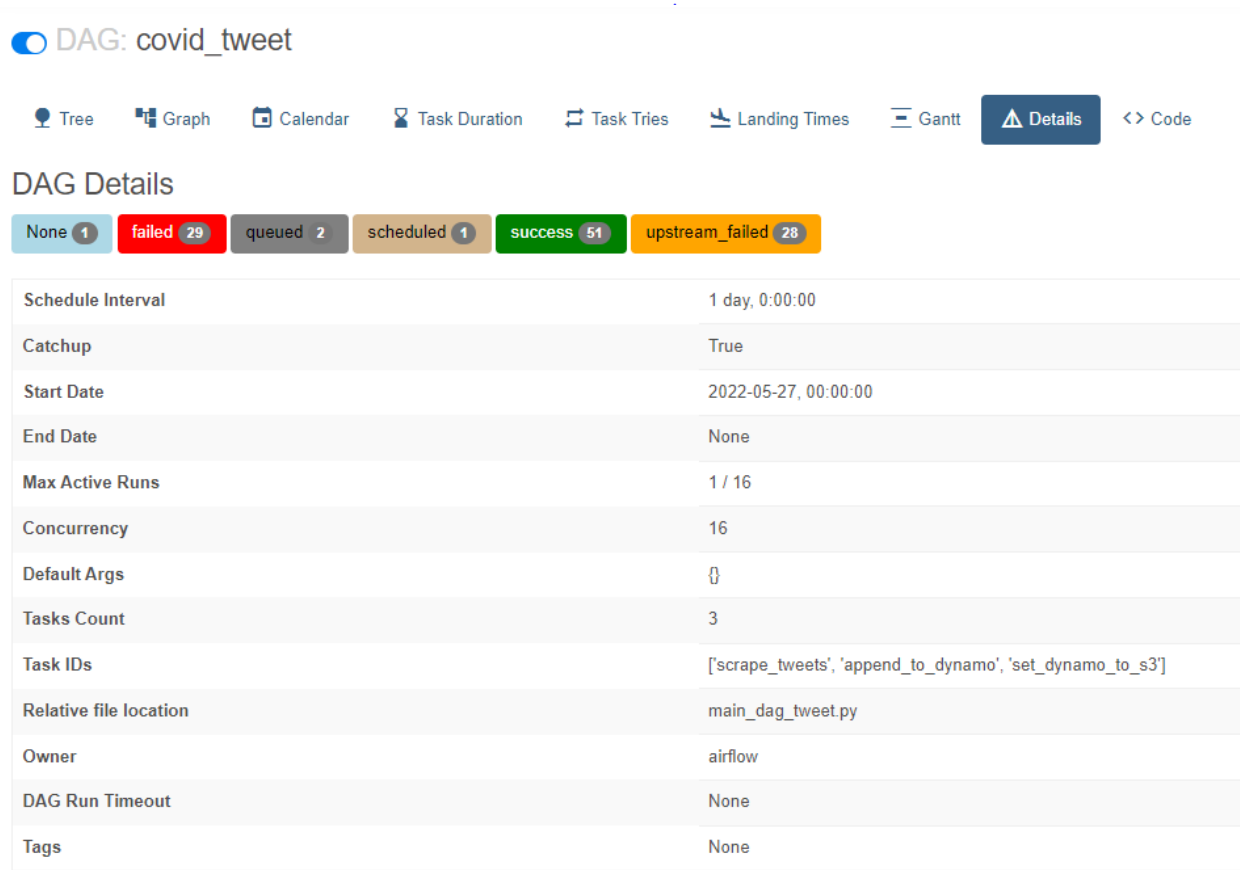
B. Dag for scraping tweets and inserting them into a DynamoDB table

1. Airflow Script

Please see attached file tweet_data.py and main_dag_tweet.py.

2. DAG Details Screenshots





Schedule Interval	1 day, 0:00:00
Catchup	True
Start Date	2022-05-27, 00:00:00
End Date	None
Max Active Runs	1 / 16
Concurrency	16
Default Args	{}
Tasks Count	3
Task IDs	['scrape_tweets', 'append_to_dynamo', 'set_dynamo_to_s3']
Relative file location	main_dag_tweet.py
Owner	airflow
DAG Run Timeout	None
Tags	None

DAG: covid_tweet

Tree | Graph | Calendar | Task Duration | Task Tries | Landing Times | Gantt | Details | Code

```python
import datetime

from airflow import DAG
from airflow.decorators import task
from airflow.operators.bash import BashOperator
from airflow.operators.python import PythonOperator
from airflow.providers.postgres.operators.postgres import PostgresOperator

from tweet_data import *

with DAG(
    dag_id='covid_tweet',
    start_date=datetime(2022, 5, 27)
) as dag:

    scrape_tweets = PythonOperator(
        task_id='scrape_tweets',
        python_callable=scrape_tweets
    )


    append_to_dynamo = PythonOperator(
        task_id='append_to_dynamo',
        python_callable=insert_to_dynamo
    )

    set_dynamo_to_s3 = PythonOperator(
        task_id='set_dynamo_to_s3',
        python_callable=dynamo_to_s3
    )


    scrape_tweets >> append_to_dynamo >> set_dynamo_to_s3
```

3. Success Runs