In [ ]:
```
# Import necessary libraries:
import boto3
import pandas as pd
from io import StringIO
```

In [ ]:
```
# Set AWS credentials and configuration:
AWS_ACCESS_KEY = "YOUR_ACCESS_KEY"
AWS_SECRET_KEY = "YOUR_SECRET_KEY"
AWS_REGION = "eu-west-2"
SCHEMA_NAME = "covid19_db"
S3_STAGING_DIR = "s3://nehal-dev-test-bucket/output6/"
S3_BUCKET_NAME = "nehal-dev-test-bucket"
S3_OUTPUT_DIRECTORY = "output6"
```

In [ ]:
```
# Create an Athena client:
athena_client = boto3.client('athena',
                      region_name=AWS_REGION,
                      aws_access_key_id=AWS_ACCESS_KEY,
                      aws_secret_access_key=AWS_SECRET_KEY)
```

In [57]:
```
# Define a function to download and load query results into a DataFrame
# This function uses the boto3 library to download the query results from S3 and load them into a DataFrame using pandas.
Dict = {}
def download_and_load_query_results(
    client:boto3.client, query_response: Dict
):
    while True:
        try:
            # This function only loads the first 1000 rows
            client.get_query_results(
                QueryExecutionId = query_response["QueryExecutionId"]
            )
            break
        except Exception as err:
            if "not yet finished" in str(err):
                time.sleep(0.001)
            else:
                raise err

    temp_file_location: str="athena_query_results.csv"

    s3_client = boto3.client("s3",
                region_name=AWS_REGION,
                aws_access_key_id=AWS_ACCESS_KEY,
                aws_secret_access_key=AWS_SECRET_KEY )

    s3_client.download_file(
        S3_BUCKET_NAME,
        f"{S3_OUTPUT_DIRECTORY}/{query_response['QueryExecutionId']}.csv",
        temp_file_location,
    )
    return pd.read_csv(temp_file_location)
```

In [ ]:
```
# Perform Athena queries and retrieve the results:
response = athena_client.start_query_execution(
    QueryString = "SELECT * FROM enigma_jhud",
    QueryExecutionContext={'Database': SCHEMA_NAME},
    ResultConfiguration={
        'OutputLocation': S3_STAGING_DIR,
        'EncryptionConfiguration' : {"EncryptionOption" : "SSE_S3"}
    }
)
```

In [ ]:
```
enigma_jhud = download_and_load_query_results(athena_client, response)
enigma_jhud.head()
```

In [ ]:
```
response = athena_client.start_query_execution(
    QueryString = "SELECT * FROM nytimes_data_in_usa_us_statesus_states",
    QueryExecutionContext={'Database': SCHEMA_NAME},
    ResultConfiguration={
        'OutputLocation': S3_STAGING_DIR,
        'EncryptionConfiguration' : {"EncryptionOption" : "SSE_S3"}
    }
)
```

In [ ]:
```
nytimes_data_in_usa_us_states = download_and_load_query_results(athena_client, response)
nytimes_data_in_usa_us_states.head()
```

In [ ]:
```
response = athena_client.start_query_execution(
    QueryString = "SELECT * FROM rearc_covid_19_testing_data_states_daily",
    QueryExecutionContext={'Database': SCHEMA_NAME},
    ResultConfiguration={
        'OutputLocation': S3_STAGING_DIR,
        'EncryptionConfiguration' : {"EncryptionOption" : "SSE_S3"}
    }
)
```

In [ ]:
```
rearc_covid_19_testing_data_states_daily = download_and_load_query_results(athena_client, response)
rearc_covid_19_testing_data_states_daily.head()
```

In [ ]:
```
response = athena_client.start_query_execution(
    QueryString = "SELECT * FROM rearc_usa_hospital_beds",
    QueryExecutionContext={'Database': SCHEMA_NAME},
    ResultConfiguration={
        'OutputLocation': S3_STAGING_DIR,
        'EncryptionConfiguration' : {"EncryptionOption" : "SSE_S3"}
    }
)
```

In [ ]:
```
rearc_usa_hospital_beds = download_and_load_query_results(athena_client, response)
rearc_usa_hospital_beds.head()
```

In [ ]:
```
response = athena_client.start_query_execution(
    QueryString = "SELECT * FROM nytimes_data_in_usa_us_county",
    QueryExecutionContext={'Database': SCHEMA_NAME},
    ResultConfiguration={
        'OutputLocation': S3_STAGING_DIR,
        'EncryptionConfiguration' : {"EncryptionOption" : "SSE_S3"}
    }
)
```

In [ ]:
```
nytimes_data_in_usa_us_county = download_and_load_query_results(athena_client, response)
nytimes_data_in_usa_us_county.head()
```

In [ ]:
```
# Perform data processing and merge operations:
factCovid_1 = enigma_jhud[['fips','province_state','country_region','confirmed','deaths','recovered','active']]
factCovid_2 = rearc_covid_19_testing_data_states_daily[['fips','date','positive','negative','hospitalizedcurrently','hospitalized','hospitalizeddischarged']]
factCovid = pd.merge(factCovid_1, factCovid_2, on='fips', how='inner')
factCovid.head()

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')
dimRegion.head()

dimHospital = rearc_usa_hospital_beds[['fips','state_name','latitude','longtitude','hq_address','hospital_name','hospital_type','hq_city','hq_state']]
dimHospital.head()
```

In [ ]:
```
# Extract columns from the DataFrame rearc_covid_19_testing_data_states_daily and create a new DataFrame dimDate:
dimDate = rearc_covid_19_testing_data_states_daily[['fips','date']]

# Convert the 'date' column to datetime format:
dimDate['date'] = pd.to_datetime(dimDate['date'], format='%Y%m%d')

# Extract year, month, and day of the week from the 'date' column and add them as new columns:
dimDate['year'] = dimDate['date'].dt.year
dimDate['month'] = dimDate['date'].dt.month
dimDate['day_of_week'] = dimDate['date'].dt.dayofweek
```

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

In [ ]:
```
# factCovid.to_csv(r'D:\1.IT PROJECT WORK\4.Data Engineer\Darsheel Parmar\COVID 19 - Build End to End Data Engineering Project\factCovid.csv')
# dimRegion.to_csv(r'D:\1.IT PROJECT WORK\4.Data Engineer\Darsheel Parmar\COVID 19 - Build End to End Data Engineering Project\dimRegion.csv')
# dimHospital.to_csv(r'D:\1.IT PROJECT WORK\4.Data Engineer\Darsheel Parmar\COVID 19 - Build End to End Data Engineering Project\dimHospital.csv')
# dimDate.to_csv(r'D:\1.IT PROJECT WORK\4.Data Engineer\Darsheel Parmar\COVID 19 - Build End to End Data Engineering Project\dimDate.csv')
```

In [ ]:
```
# Create an S3 resource object:
s3_resource = boto3.resource('s3')

# Define the S3 bucket name:
bucket = 'nehal-covid-de-project'
```

In [ ]:
```
# Convert and upload DataFrames to S3 as CSV files:
csv_buffer = StringIO()
factCovid.to_csv(csv_buffer)
s3_resource.Object(bucket, 'output/factCovid.csv').put(Body=csv_buffer.getvalue())

csv_buffer = StringIO()
dimRegion.to_csv(csv_buffer)
s3_resource.Object(bucket, 'output/dimRegion.csv').put(Body=csv_buffer.getvalue())

csv_buffer = StringIO()
dimHospital.to_csv(csv_buffer)
s3_resource.Object(bucket, 'output/dimHospital.csv').put(Body=csv_buffer.getvalue())

csv_buffer = StringIO()
dimDate.to_csv(csv_buffer)
s3_resource.Object(bucket, 'output/dimDate.csv').put(Body=csv_buffer.getvalue())
```

In [ ]:
```
# Generate and print SQL schemas for DataFrames:
factCovidsql = pd.io.sql.get_schema(factCovid.reset_index(), 'factCovid')
print(''.join(factCovidsql))

dimHospitalsql = pd.io.sql.get_schema(dimHospital.reset_index(), 'dimHospital')
print(''.join(dimHospitalsql))

dimDatesql = pd.io.sql.get_schema(dimDate.reset_index(), 'dimDate')
print(''.join(dimDatesql))
```

In [ ]:
```
# Install the redshift-connector library:
pip install redshift-connector
```

In [ ]:
```
import redshift_connector
```

In [ ]:
```
# Connect to Amazon Redshift:
conn = redshift_connector.connect(
    host='my-first-redshift.cvxs7hla7yh8.eu-west-2.redshift.amazonaws.com',
    database='myfirstdb',
    user='user',
    password="Password"
)
conn.autocommit = True
cursor= redshift_connector.Cursor = conn.cursor()
```

In [ ]:
```
# Execute SQL queries to create tables into Redshift:
cursor.execute("""
    CREATE TABLE "factCovid" (
"index" INTEGER,
  "fips" REAL,
  "province_state" TEXT,
  "country_region" TEXT,
  "confirmed" REAL,
  "deaths" REAL,
  "recovered" REAL,
  "active" REAL,
  "date" INTEGER,
  "positive" REAL,
  "negative" REAL,
  "hospitalizedcurrently" REAL,
  "hospitalized" REAL,
  "hospitalizeddischarged" REAL
)
""")

cursor.execute("""
    CREATE TABLE "dimHospital" (
"index" INTEGER,
  "fips" REAL,
  "state_name" TEXT,
  "latitude" REAL,
  "longtitude" REAL,
  "hq_address" TEXT,
  "hospital_name" TEXT,
  "hospital_type" TEXT,
  "hq_city" TEXT,
  "hq_state" TEXT
)
""")

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

In [ ]:
```
# Execute SQL queries to copy data from S3 into Redshift:
cursor.execute("""
    copy dimDate from 's3://nehal-covid-de-project/output/dimDate.csv'
    credentials 'aws_iam_role=arn:aws:iam::127656700581:role/Redshift-S3-access'
    delimiter ','
    region 'eu-west-2'
    IGNOREHEADER 1
    """)

cursor.execute("""
    copy dimHospital from 's3://nehal-covid-de-project/output/dimHospital.csv'
    credentials 'aws_iam_role=arn:aws:iam::127658009581:role/Redshift-S3-access'
    delimiter ','
    region 'eu-west-2'
    IGNOREHEADER 1
    """)

cursor.execute("""
    copy factCovid from 's3://nehal-covid-de-project/output/factCovid.csv'
    credentials 'aws_iam_role=arn:aws:iam::127654009581:role/Redshift-S3-access'
    delimiter ','
    region 'eu-west-2'
    IGNOREHEADER 1
    """)
```