**Apache Airflow DAG scripts:**

1. **Real time data pipeline:**

from airflow import DAG

from airflow.operators.python import PythonOperator, BranchPythonOperator

from airflow.operators.bash import BashOperator

from datetime import datetime,timedelta

import airflow.utils

from airflow.providers.amazon.aws.operators.glue import GlueJobOperator

from airflow.operators.dummy import DummyOperator

import boto3

import os

from airflow.models.connection import Connection

### glue job specific variables

glue\_job\_name = "supplementary-data-pipeline"

glue\_iam\_role = "db-project-glue-role"

region\_name = "us-east-1"

email\_recipient = "[saumya.varshney@sjsu.edu](mailto:saumya.varshney@sjsu.edu)"

bucket\_name = "db-project-source-files"

bucket\_name\_del = "db-project-staging-area"

# Define the default arguments for the DAG

default\_args = {

'owner': 'airflow',

'depends\_on\_past': False,

'start\_date': airflow.utils.dates.days\_ago(0),

'retries': 1,

'retry\_delay': timedelta(minutes=5),

}

def get\_files\_matching\_prefix(prefix, s3\_client):

response = s3\_client.list\_objects\_v2(Bucket=bucket\_name\_del, Prefix=prefix)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("These are the files matching prefix \n"+str(response))

files\_in\_folder = response["Contents"]

files\_to\_delete = []

# We will create Key array to pass to delete\_objects function

for f in files\_in\_folder:

files\_to\_delete.append({"Key": f["Key"]})

return files\_to\_delete

def delete\_objects\_from\_supplementary\_bucket():

s3\_client = boto3.client("s3",

aws\_access\_key\_id = 'AKIAQB3UYZWL5LPL3EH7',

aws\_secret\_access\_key= '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',

region\_name="us-east-1")

all\_files\_to\_delete = []

# First we list all files matching a prefix

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("datatype/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("date/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("location/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("locationcategory/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("station/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("stationrelation/",s3\_client=s3\_client))

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("All files matching prefix"+str(all\_files\_to\_delete))

if len(all\_files\_to\_delete) > 0:

response = s3\_client.delete\_objects(

Bucket=bucket\_name\_del,

Delete={"Objects": all\_files\_to\_delete}

)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("Response from Delete"+str(response))

def delete\_objects\_from\_realtime\_bucket():

s3\_client = boto3.client("s3",

aws\_access\_key\_id = 'AKIAQB3UYZWL5LPL3EH7',

aws\_secret\_access\_key= '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',

region\_name="us-east-1")

all\_files\_to\_delete = []

# First we list all files matching a prefix

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2023/",s3\_client=s3\_client))

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("All files matching prefix"+str(all\_files\_to\_delete))

if len(all\_files\_to\_delete) > 0:

response = s3\_client.delete\_objects(

Bucket=bucket\_name\_del,

Delete={"Objects": all\_files\_to\_delete}

)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("Response from Delete"+str(response))

# Instantiate a DAG object

dag = DAG('Pipeline\_Orchestration\_DAG\_realtime', default\_args=default\_args, description='Orchestrate Realtime jobs', schedule\_interval=None)

# Start task

start\_task = DummyOperator(task\_id='start\_task', dag=dag)

delete\_objects\_task = PythonOperator(

task\_id='delete\_objects\_supplementry',

python\_callable=delete\_objects\_from\_supplementary\_bucket,

dag=dag,

)

delete\_objects\_task\_real\_time = PythonOperator(

task\_id='delete\_objects\_realtime',

python\_callable=delete\_objects\_from\_realtime\_bucket,

dag=dag,

)

load\_supplementary\_data\_task = GlueJobOperator(

task\_id='load\_supplementary\_data\_task',

job\_name='supplementary-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/supplementary-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

create\_job\_kwargs={"GlueVersion": "3.0", "NumberOfWorkers": 2, "WorkerType": "G.1X"},

dag=dag, verbose=False)

# # Job for loading data for 2023

load\_realtime\_data\_task = GlueJobOperator(

task\_id='load\_realtime\_data\_task',

job\_name='realtime-data-pipeline',

script\_location=f"s3://{bucket\_name}/python\_scripts/realtime-data-pipeline.py",

s3\_bucket=bucket\_name,

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

create\_job\_kwargs={"GlueVersion": "3.0", "NumberOfWorkers": 2, "WorkerType": "G.1X"},

dag=dag, verbose=False)

# # # Job for loading archive data to redshift

load\_realtime\_data\_to\_redshift\_task = GlueJobOperator(

task\_id='load\_realtime\_data\_to\_redshift\_task',

job\_name='realtime-data-to-redshift',

script\_location=f"s3://{bucket\_name}/python\_scripts/realtime-data-to-redshift.py",

s3\_bucket=bucket\_name,

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

# # # Job for loading supplementary to redshift

load\_supplementary\_data\_to\_redshift\_task = GlueJobOperator(

task\_id='load\_supplementary\_data\_to\_redshift\_task',

job\_name='supplementary-data-to-redshift',

script\_location=f"s3://{bucket\_name}/python\_scripts/supplementary-data-to-redshift.py",

s3\_bucket=bucket\_name,

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

# End task

end\_task = DummyOperator(task\_id='end\_task', dag=dag)

# Set up dependencies for Realtime data pipeline

start\_task >> delete\_objects\_task >> load\_supplementary\_data\_task >> load\_supplementary\_data\_to\_redshift\_task >> end\_task

start\_task >> delete\_objects\_task >> load\_supplementary\_data\_task >> delete\_objects\_task\_real\_time >> load\_realtime\_data\_task >> load\_realtime\_data\_to\_redshift\_task >> end\_task

1. **Archive data pipeline:**

from airflow import DAG

from airflow.operators.python import PythonOperator, BranchPythonOperator

from airflow.operators.bash import BashOperator

from datetime import datetime,timedelta

import airflow.utils

from airflow.providers.amazon.aws.operators.glue import GlueJobOperator

from airflow.operators.dummy import DummyOperator

import boto3

import os

from airflow.models.connection import Connection

### glue job specific variables

glue\_job\_name = "supplementary-data-pipeline"

glue\_iam\_role = "db-project-glue-role"

region\_name = "us-east-1"

email\_recipient = "[saumya.varshney@sjsu.edu](mailto:saumya.varshney@sjsu.edu)"

bucket\_name = "db-project-source-files"

bucket\_name\_del = "db-project-staging-area"

# Define the default arguments for the DAG

default\_args = {

'owner': 'airflow',

'depends\_on\_past': False,

'start\_date': airflow.utils.dates.days\_ago(0),

'retries': 1,

'retry\_delay': timedelta(minutes=5),

}

def get\_files\_matching\_prefix(prefix, s3\_client):

response = s3\_client.list\_objects\_v2(Bucket=bucket\_name\_del, Prefix=prefix)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("These are the files matching prefix \n"+str(response))

files\_in\_folder = response["Contents"]

files\_to\_delete = []

# We will create Key array to pass to delete\_objects function

for f in files\_in\_folder:

files\_to\_delete.append({"Key": f["Key"]})

return files\_to\_delete

def delete\_objects\_from\_archive\_bucket():

s3\_client = boto3.client("s3",

aws\_access\_key\_id = 'AKIAQB3UYZWL5LPL3EH7',

aws\_secret\_access\_key= '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',

region\_name="us-east-1")

all\_files\_to\_delete = []

# First we list all files matching a prefix

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2010/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2011/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2012/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2013/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2014/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2015/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2016/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2017/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2018/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2019/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2020/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2021/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("data/year=2022/",s3\_client=s3\_client))

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("All files matching prefix"+str(all\_files\_to\_delete))

if len(all\_files\_to\_delete) > 0:

response = s3\_client.delete\_objects(

Bucket=bucket\_name\_del,

Delete={"Objects": all\_files\_to\_delete}

)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("Response from Delete"+str(response))

def delete\_objects\_from\_supplementary\_bucket():

s3\_client = boto3.client("s3",

aws\_access\_key\_id = 'AKIAQB3UYZWL5LPL3EH7',

aws\_secret\_access\_key= '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',

region\_name="us-east-1")

all\_files\_to\_delete = []

# First we list all files matching a prefix

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("datatype/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("date/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("location/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("locationcategory/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("station/",s3\_client=s3\_client))

all\_files\_to\_delete.extend(get\_files\_matching\_prefix("stationrelation/",s3\_client=s3\_client))

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("All files matching prefix"+str(all\_files\_to\_delete))

if len(all\_files\_to\_delete) > 0:

response = s3\_client.delete\_objects(

Bucket=bucket\_name\_del,

Delete={"Objects": all\_files\_to\_delete}

)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("Response from Delete"+str(response))

# Instantiate a DAG object

dag = DAG('Pipeline\_Orchestration\_DAG\_archive', default\_args=default\_args, description='Orchestrate Archive jobs', schedule\_interval=None)

# Start tasks

start\_task = DummyOperator(task\_id='start\_task', dag=dag)

delete\_objects\_task = PythonOperator(

task\_id='delete\_objects\_supplementry',

python\_callable=delete\_objects\_from\_supplementary\_bucket,

dag=dag,

)

delete\_objects\_task\_archive = PythonOperator(

task\_id='delete\_objects\_archive',

python\_callable=delete\_objects\_from\_archive\_bucket,

dag=dag,

)

# Loading supplementary data

load\_Supplementary\_data\_task = GlueJobOperator(

task\_id='load\_Supplementary\_data\_task',

job\_name='supplementary-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/supplementary-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

create\_job\_kwargs={"GlueVersion": "3.0", "NumberOfWorkers": 2, "WorkerType": "G.1X"},

dag=dag, verbose=False)

# Job for loading data from 2010-2022

#2010

load\_2010\_data\_task = GlueJobOperator(

task\_id='load\_2010\_data\_task',

job\_name='2010-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2010-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2011

load\_2011\_data\_task = GlueJobOperator(task\_id='load\_2011\_data\_task',

job\_name='2011-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2011-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2012

load\_2012\_data\_task = GlueJobOperator(task\_id='load\_2012\_data\_task',

job\_name='2012-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2012-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2013

load\_2013\_data\_task = GlueJobOperator(task\_id='load\_2013\_data\_task',

job\_name='2013-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2013-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2014

load\_2014\_data\_task = GlueJobOperator(task\_id='load\_2014\_data\_task',

job\_name='2014-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2014-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2015

load\_2015\_data\_task = GlueJobOperator(task\_id='load\_2015\_data\_task',

job\_name='2015-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2015-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2016

load\_2016\_data\_task = GlueJobOperator(task\_id='load\_2016\_data\_task',

job\_name='2016-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2016-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2017

load\_2017\_data\_task = GlueJobOperator(task\_id='load\_2017\_data\_task',

job\_name='2017-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2017-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2018

load\_2018\_data\_task = GlueJobOperator(task\_id='load\_2018\_data\_task',

job\_name='2018-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2018-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2019

load\_2019\_data\_task = GlueJobOperator(task\_id='load\_2019\_data\_task',

job\_name='2019-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2019-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2020

load\_2020\_data\_task = GlueJobOperator(task\_id='load\_2020\_data\_task',

job\_name='2020-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2020-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2021

load\_2021\_data\_task = GlueJobOperator(task\_id='load\_2021\_data\_task',

job\_name='2021-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2021-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

#2022

load\_2022\_data\_task = GlueJobOperator(task\_id='load\_2022\_data\_task',

job\_name='2022-data-pipeline',

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

script\_location=f"s3://{bucket\_name}/python\_scripts/2022-data-pipeline.py",

s3\_bucket=bucket\_name,

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

# Job for loading archive data to redshift

load\_archive\_data\_to\_redshift\_task = GlueJobOperator(task\_id='load\_archive\_data\_to\_redshift\_task',

job\_name='archive-data-to-redshift',

script\_location=f"s3://{bucket\_name}/python\_scripts/realtime-data-to-redshift.py",

s3\_bucket=bucket\_name,

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

# Job for loading supplementary to redshift

load\_supplementary\_data\_to\_redshift\_task = GlueJobOperator(task\_id='load\_supplementary\_data\_to\_redshift\_task',

job\_name='supplementary-data-to-redshift',

script\_location=f"s3://{bucket\_name}/python\_scripts/supplementary-data-to-redshift.py",

s3\_bucket=bucket\_name,

aws\_conn\_id= 'noaa\_db\_project\_aws\_connection\_1',

iam\_role\_name='db-project-glue-role',

region\_name="us-east-1",

dag=dag, verbose=False)

# End task

end\_task = DummyOperator(task\_id='end\_task', dag=dag)

# Set up dependencies for Archive data pipeline

start\_task >> delete\_objects\_task >> load\_Supplementary\_data\_task >> load\_supplementary\_data\_to\_redshift\_task >> end\_task

start\_task >> delete\_objects\_task >> load\_Supplementary\_data\_task >> delete\_objects\_task\_archive >> load\_2010\_data\_task >> load\_2011\_data\_task >> load\_2012\_data\_task \

>> load\_2013\_data\_task >> load\_2014\_data\_task >> load\_2015\_data\_task >> load\_2016\_data\_task >> load\_2017\_data\_task \

>> load\_2018\_data\_task >> load\_2019\_data\_task >> load\_2020\_data\_task >> load\_2021\_data\_task >> load\_2022\_data\_task \

>> load\_archive\_data\_to\_redshift\_task >> end\_task