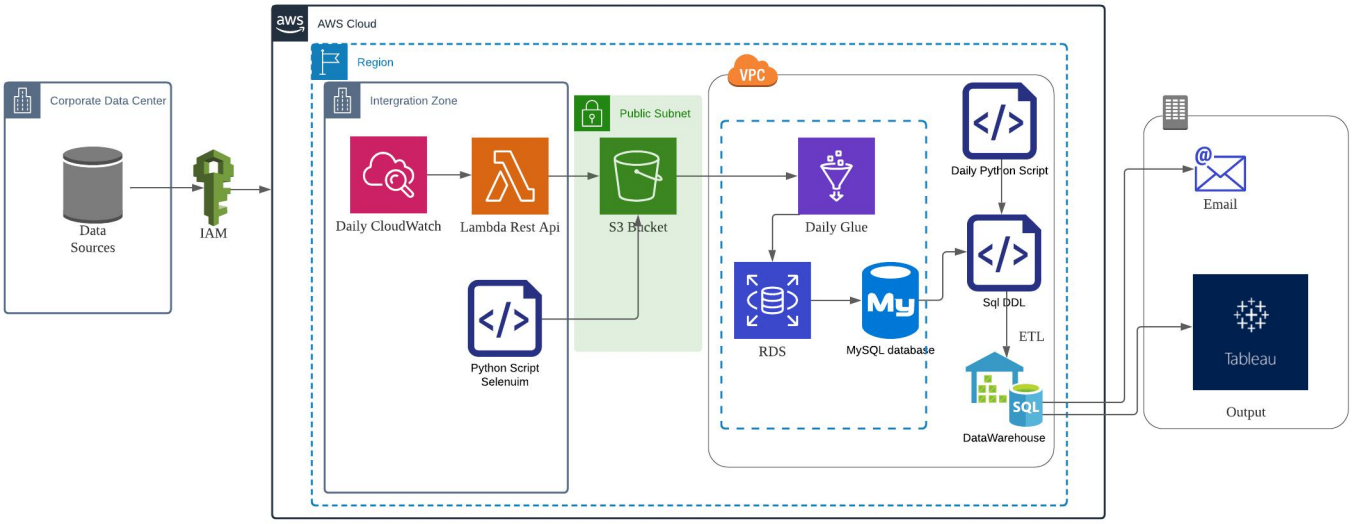
6100 Information Architecture Summer 2021 Final Project Report

Student: Xiaolan Li, Bernard Cooper

Professor: Brandon Chiazza

Architecture:



1. Using Chrome Driver Selenium method to obtain the unstructured NYC median income Data set.

https://data.cccnewyork.org/data/table/66/median-incomes#66/107/62/a/a

[1]:
```
import awscli
import selenium
import boto3
import os
import s3fs
import pandas as pd
import time
from selenium import webdriver
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
repo = os.path.dirname(os.path.abspath(''))
```

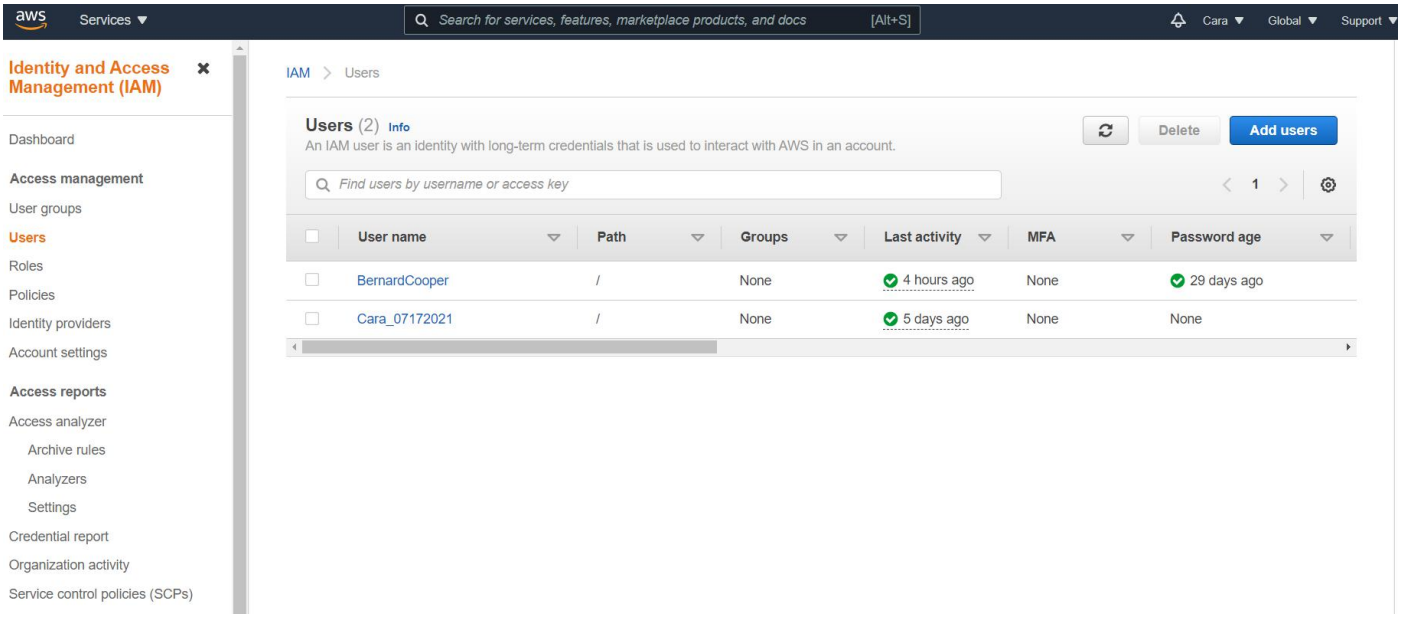
[3]:
```
browser = webdriver.Chrome(os.path.join(repo, "ResourceDatasets", "chromedriver.exe"))

#enter the url path that needs to be accessed by webdriver
browser.get('https://data.cccnewyork.org/data/table/66/median-incomes#66/107/62/a/a')
time.sleep(5)
#identify xpath of location to select element
table = browser.find_element_by_xpath("/html/body/div[1]/div[2]/div[2]/div[3]/div/table")
df =[]

#loop through dataframe to export table
for row in table.find_elements_by_css_selector('tr'):
      cols = df.append([cell.text for cell in row.find_elements_by_css_selector('td')])

defined_columns = ['location','all_households','families','families_with_children','families_without_children']
```

[4]:
```
df_median_BOROUGHS = pd.DataFrame(df, columns=df[1])[4:9].reset_index(drop=True)
df_median_BOROUGHS.columns = defined_columns
df_median_BOROUGHS
```

[4]:

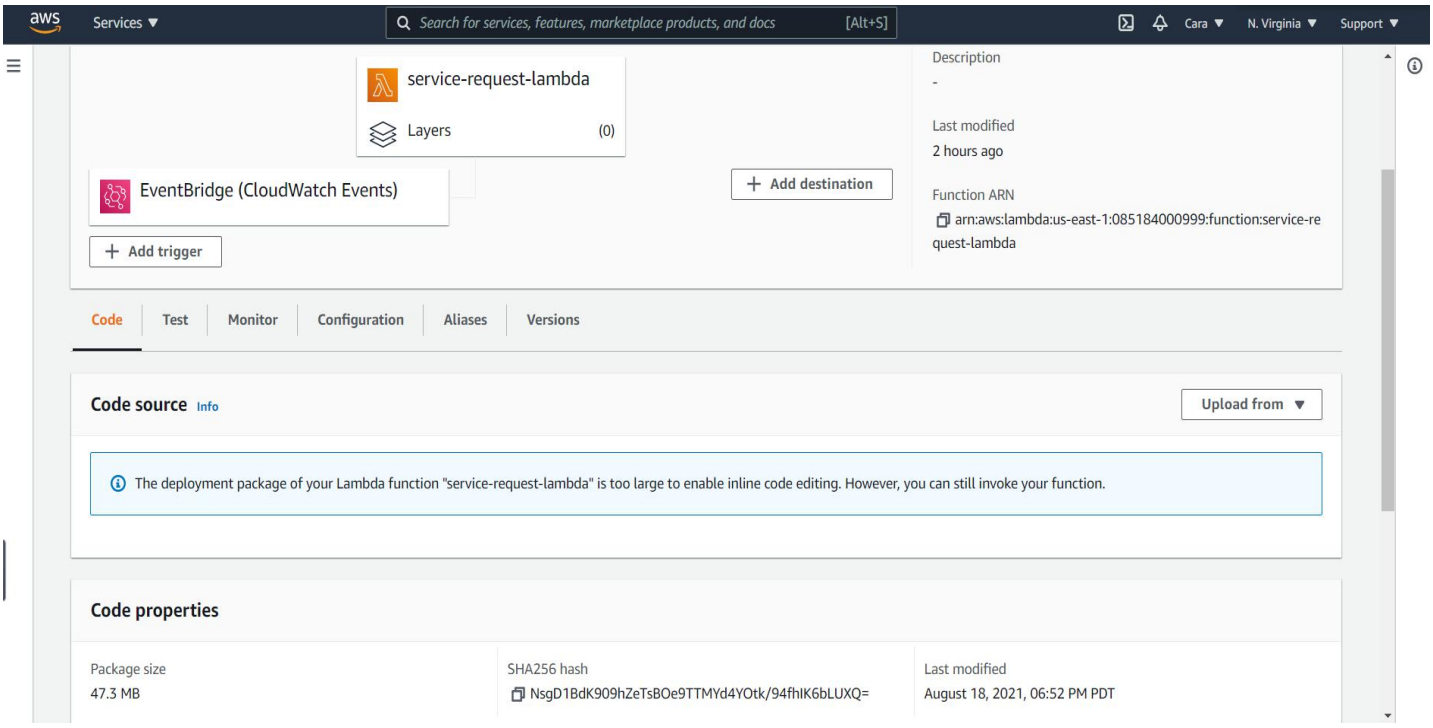
	location	all_households	families	families_with_children	families_without_children
0	Bronx	$41,432	$50,835	$41,129	$61,248
1	Brooklyn	$66,937	$74,422	$66,936	$79,400
2	Manhattan	$93,651	$126,690	$140,841	$121,669
3	Queens	$73,696	$82,534	$75,501	$86,501
4	Staten Island	$89,821	$105,438	$104,641	$106,015

2. Using IAM service to manage the users authorization and role

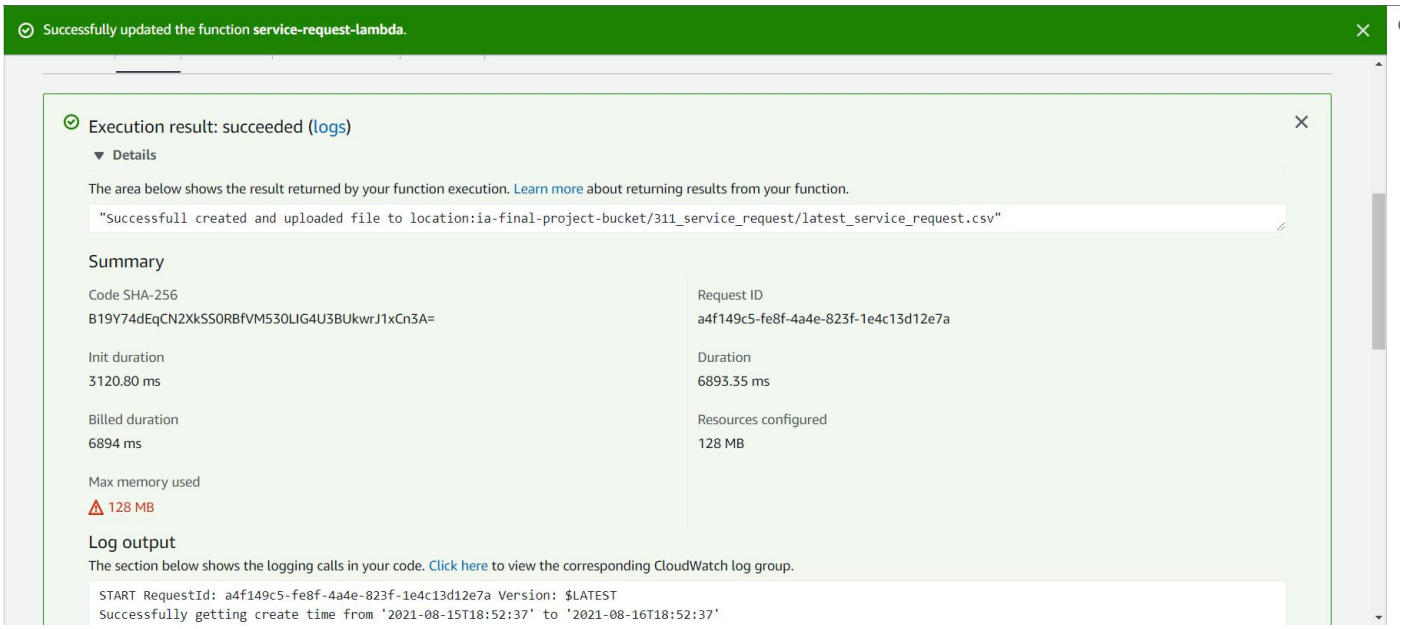


3. Using Lambda Service with REST API method to daily obtain the last day incident resource from 311 service request structured database

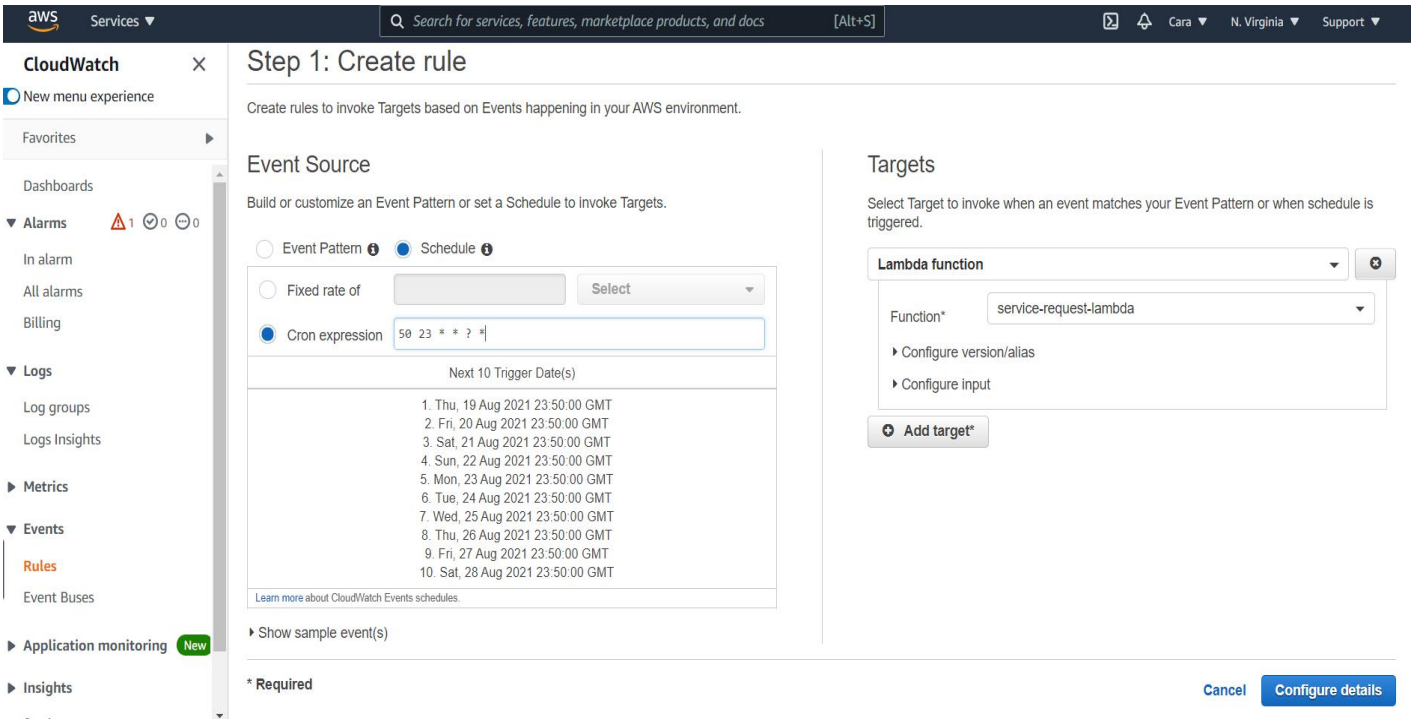
https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

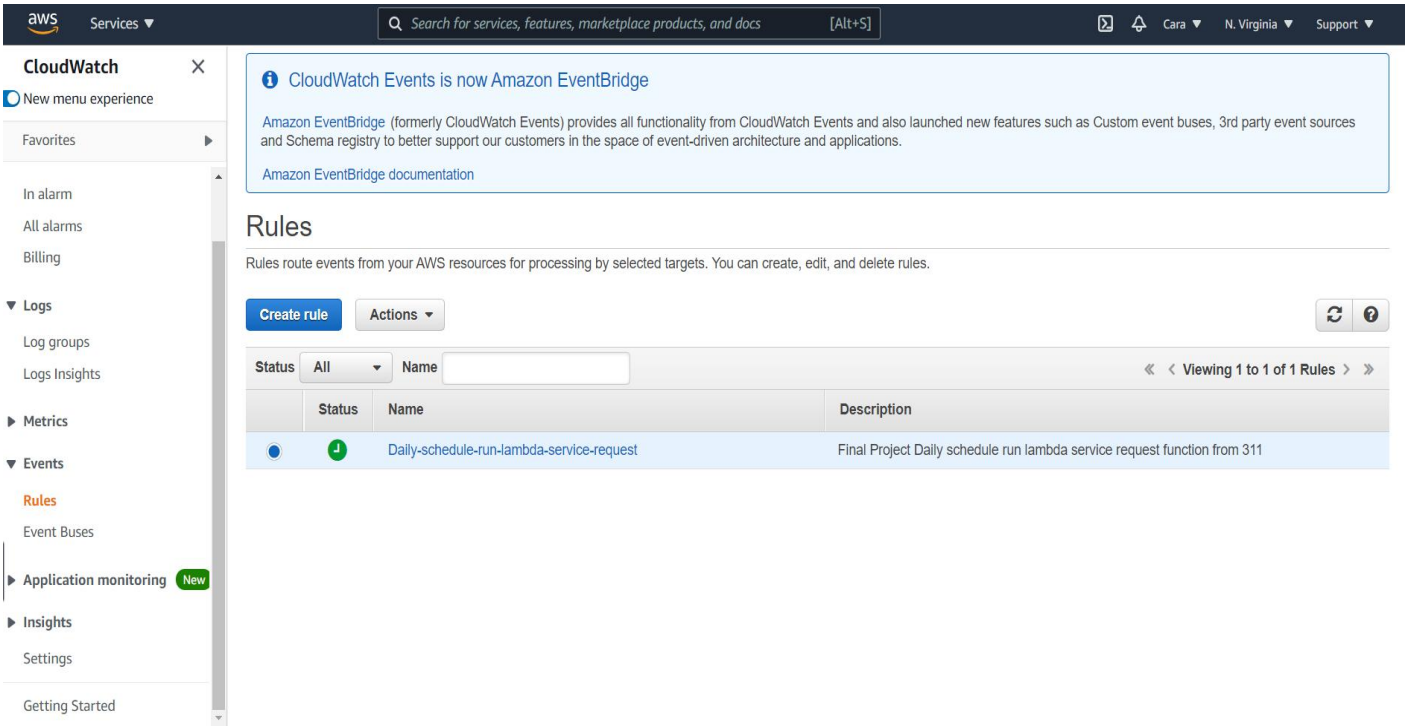


```python
    Previous_twoday = str(datetime.datetime.today() - datetime.timedelta(days=2))[:19]
    Previous_twoday = Previous_twoday[:10] + 'T' + Previous_twoday[11:]
    Previous_oneday = str(datetime.datetime.today() - datetime.timedelta(days=1))[:19]
    Previous_oneday = Previous_oneday[:10] + 'T' + Previous_oneday[11:]
    response = client.get("erm2-nwe9",
                          where="created_date BETWEEN '" + Previous_twoday + "' AND '" + Previous_oneday + "'",
                          limit=10000)
    results_df = pd.DataFrame.from_records(response)
    client.close()
    print('Successfully getting create time from \'{}\' to \'{}\''.format(Previous_twoday, Previous_oneday))

    s3 = boto3.client('s3')
    try:
        file_name = "311_service_request/latest_service_request.csv"
        # check if the csv file exist in s3 bucket
        # get the existing file
        obj = s3.get_object(Bucket='ia-final-project-bucket', Key=file_name)
        df_current = pd.read_csv(io.BytesIO(obj['Body'].read()), index_col=0)
        current_data = df_current.to_csv(None, index=False).encode()
        # append
        bytes_to_write = results_df.to_csv(None, header=None, index=False).encode()
        appended_data = current_data + bytes_to_write
        # overwrite
        s3.put_object(Body=appended_data, Bucket='ia-final-project-bucket', Key=file_name)
        return 'Successfully found existing file and appended data to {}'.format(file_name)

    except ClientError:
        # Not found
        pathname = 'ia-final-project-bucket/'  # specify location of s3:/{my-bucket}/
```

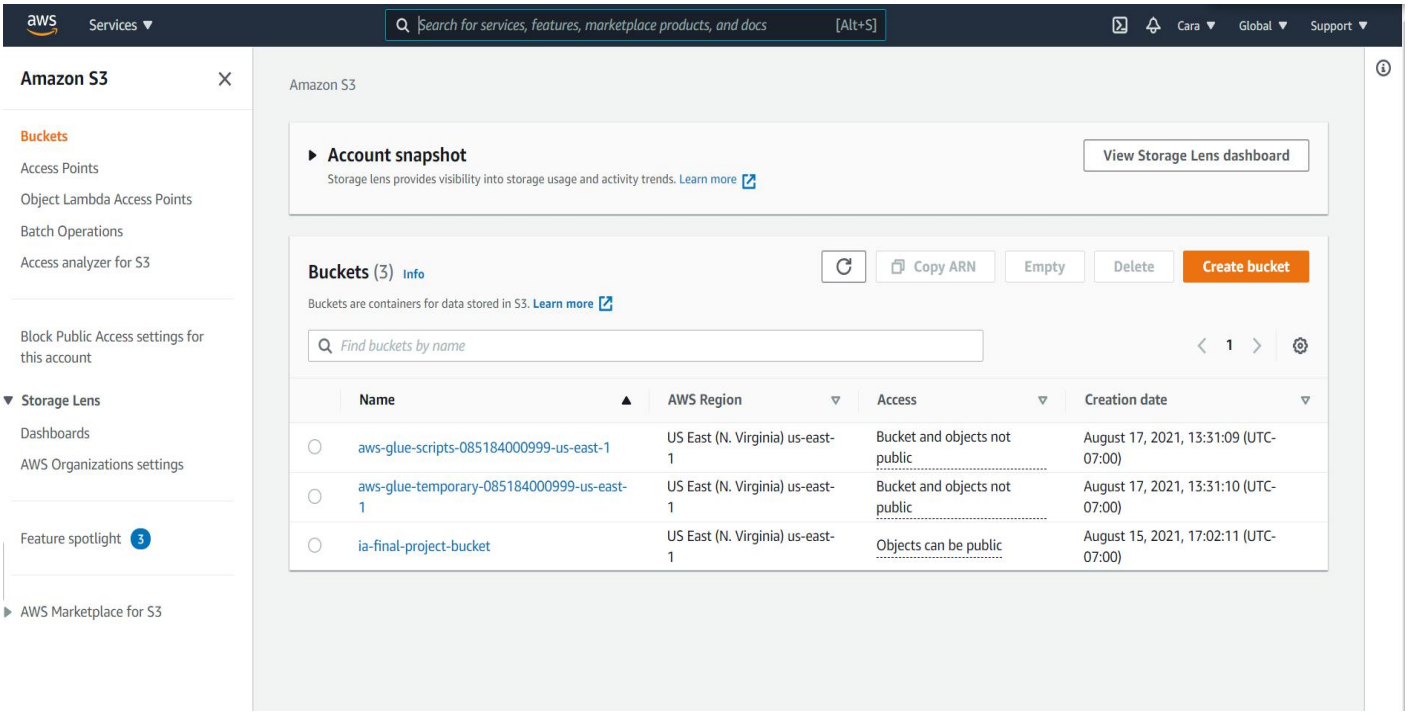


4. Using Cloud Watch Service to schedule the time to run lambda function

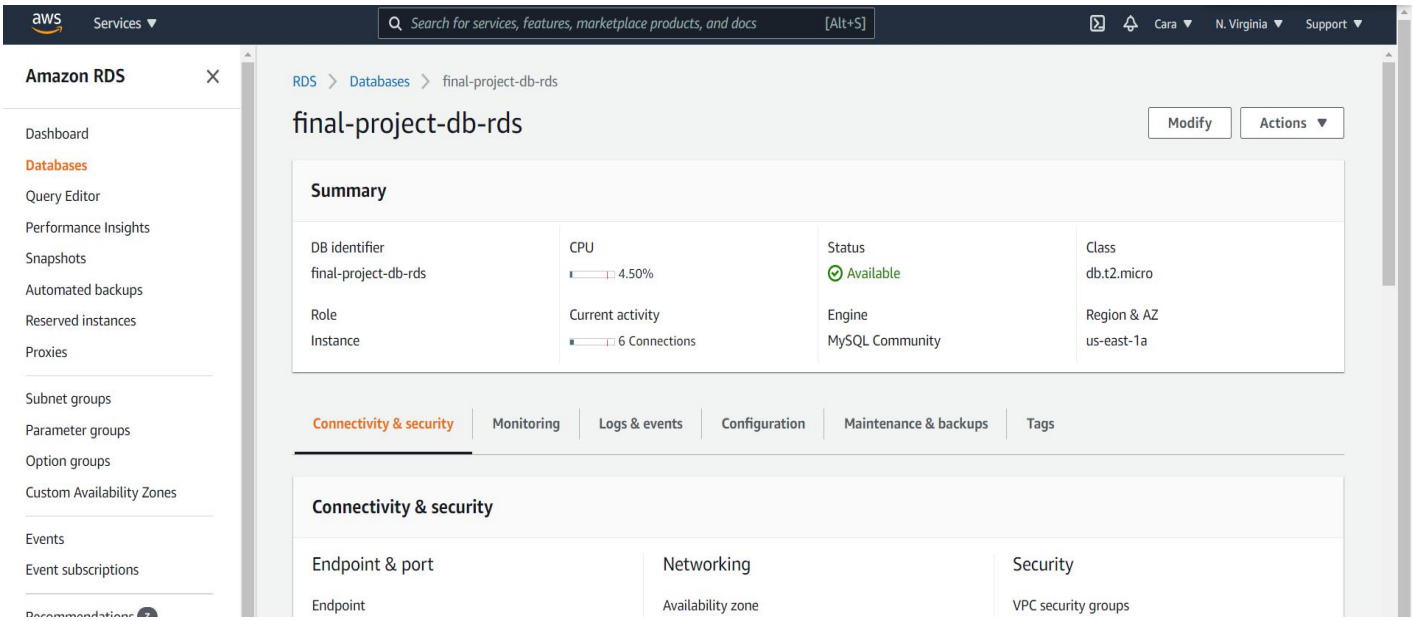




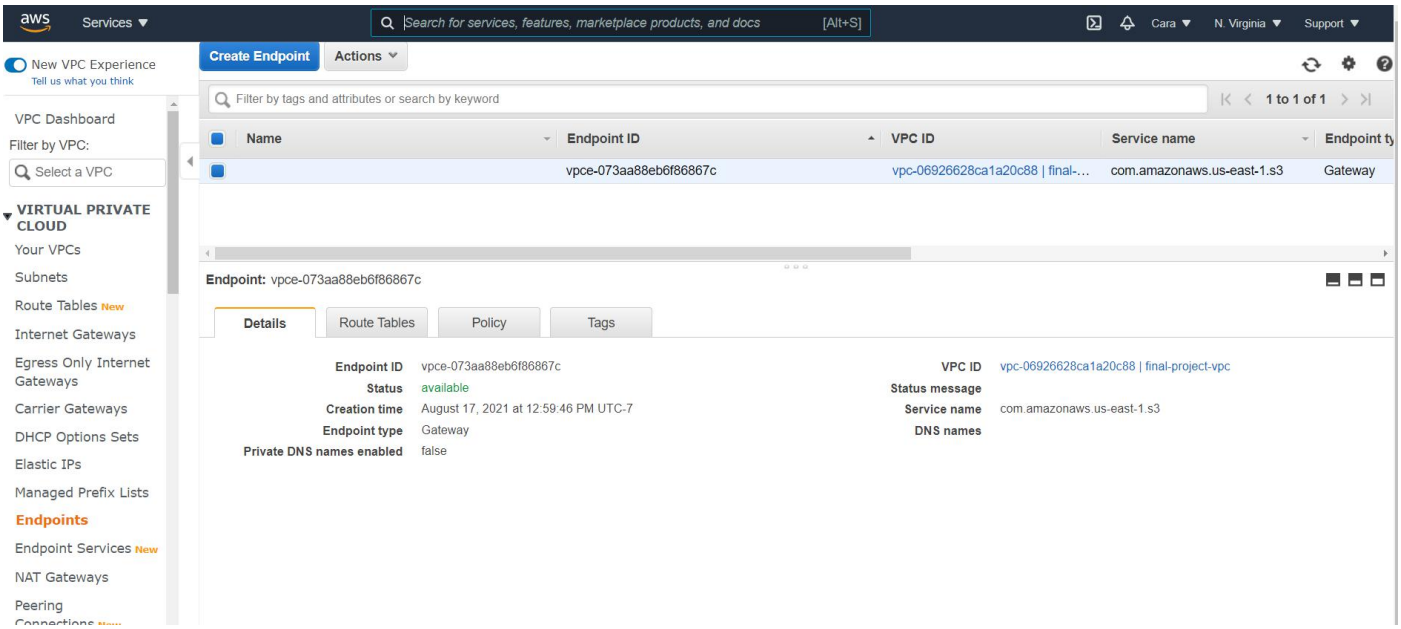
5. Using S3 Service to store the two data sources in `ia-final-project-bucket` bucket



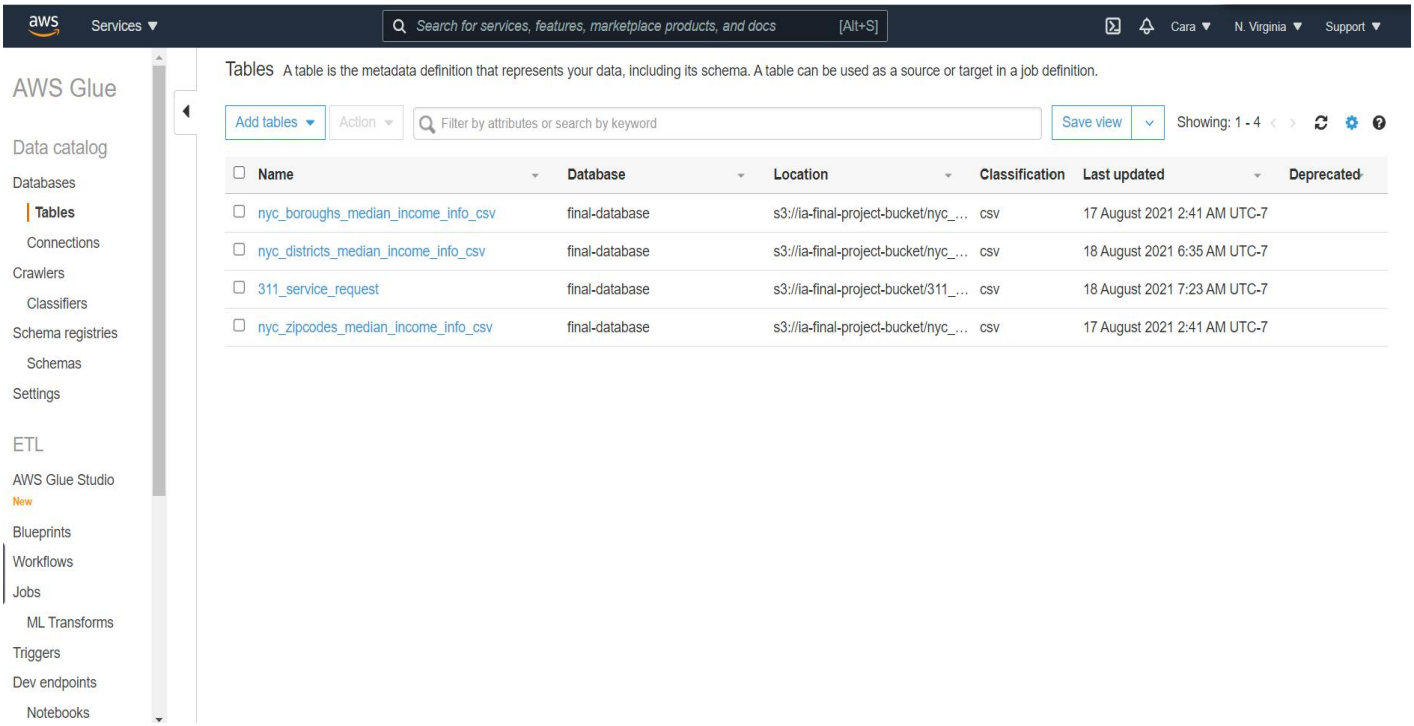
6. Using RDS to store the sources schema and data warehouse

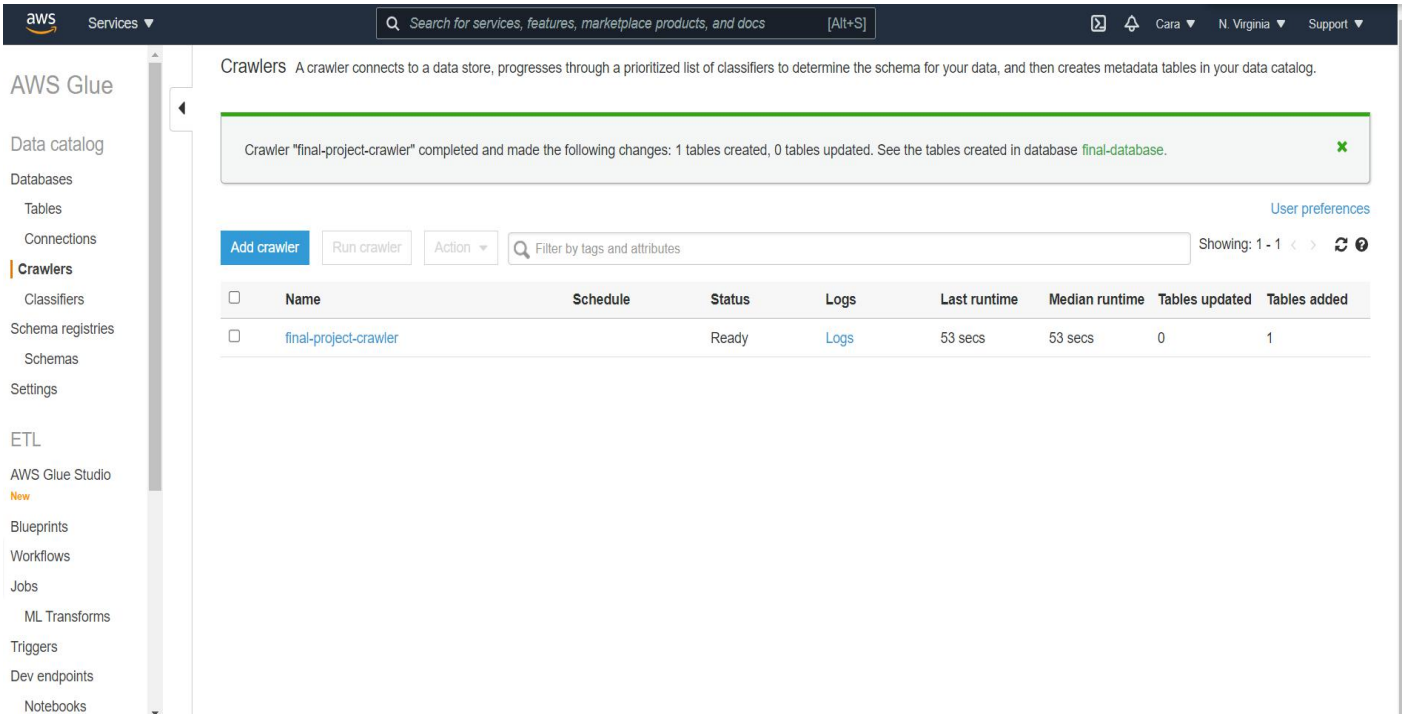


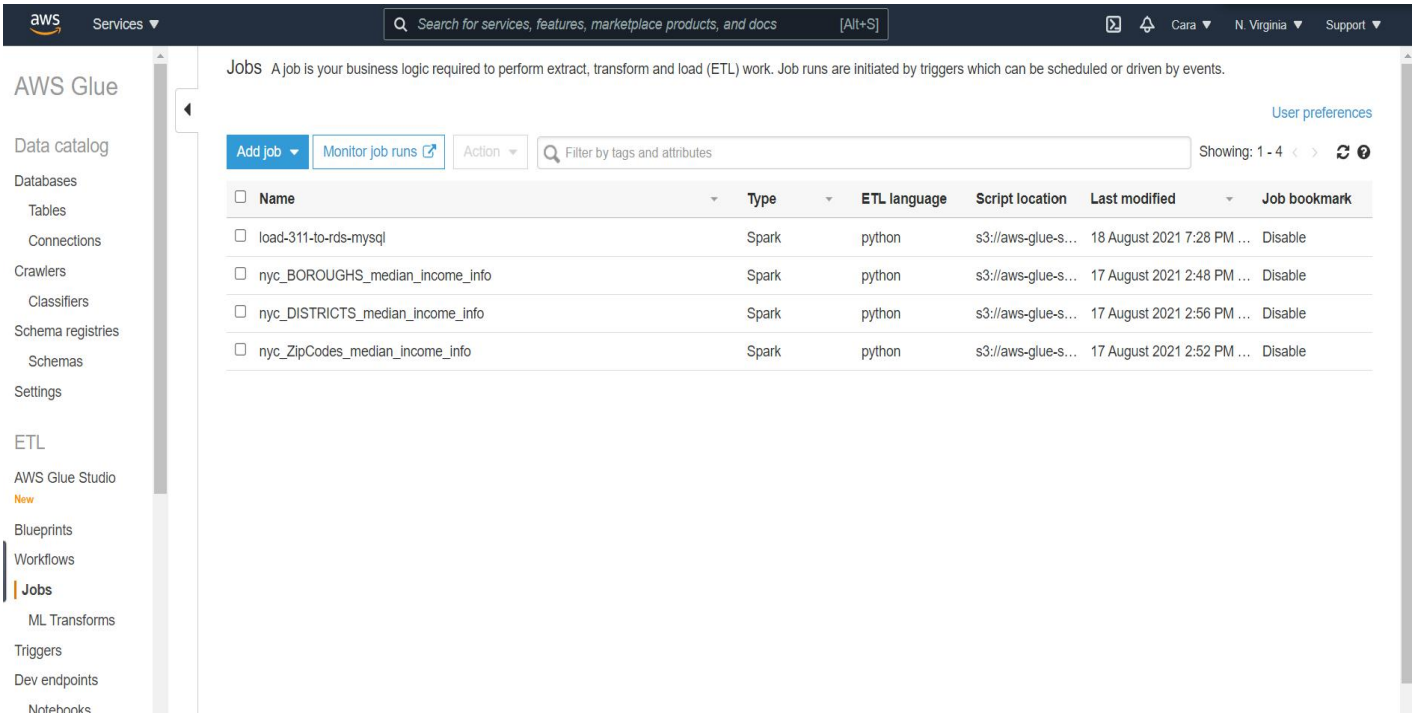
7. Using VPC service to create an endpoint for connecting the S3 Gateway

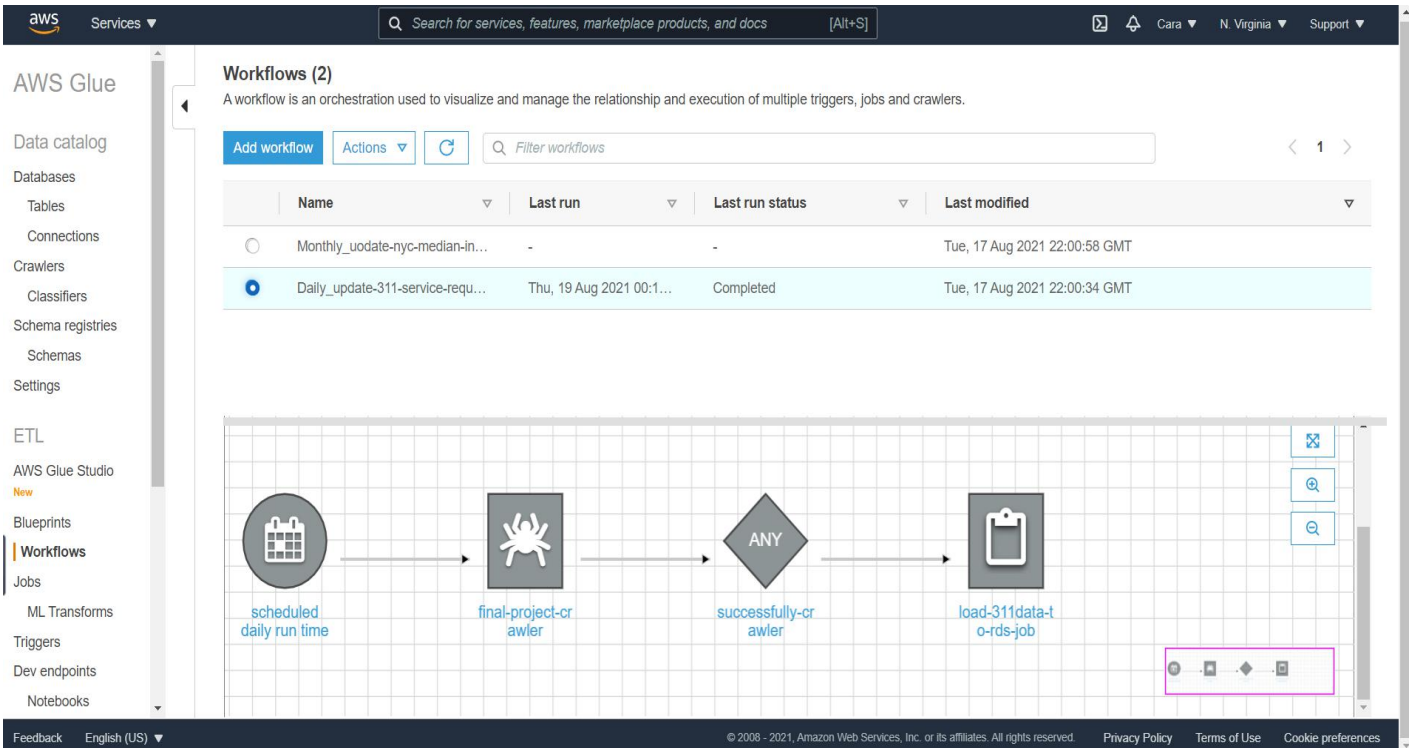


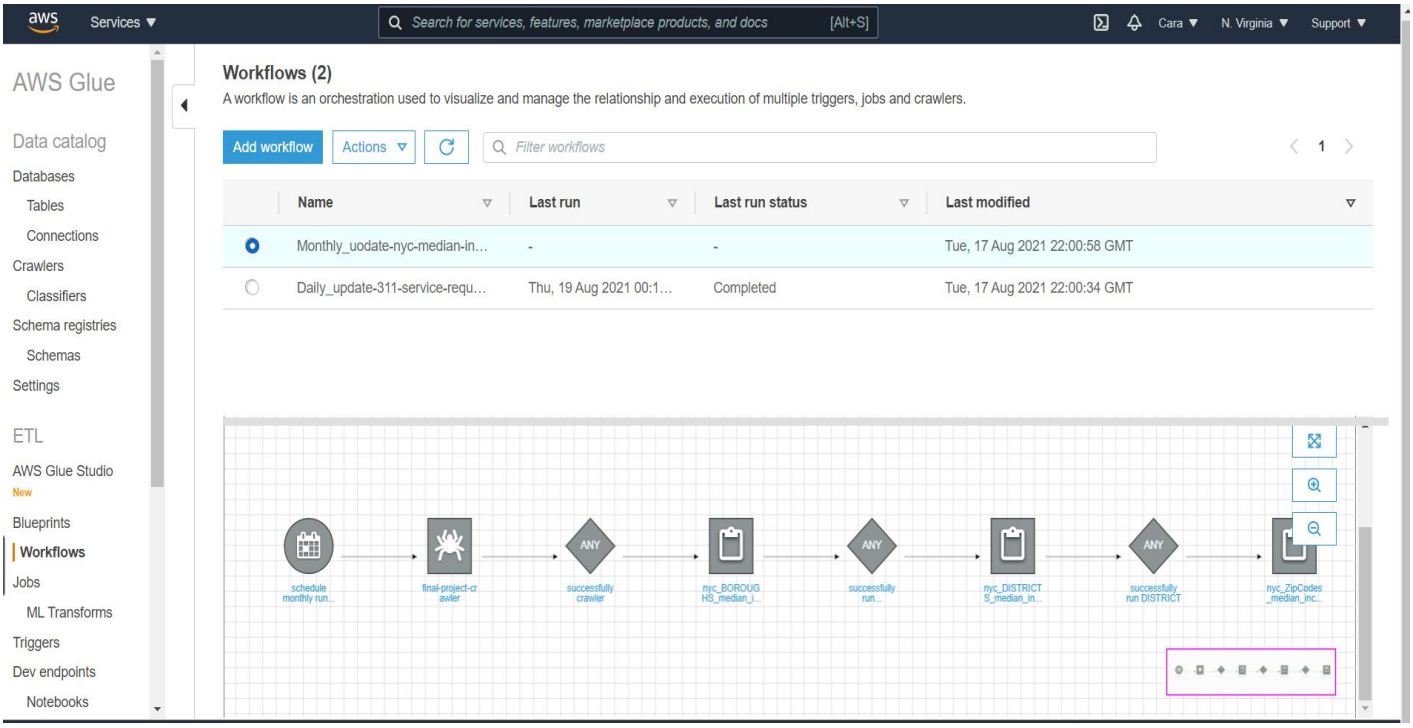
8. Using GLUE service to load the data into RDS with MySQL Work Bench in daily schedule with workflow steps crawler table from s3 and running jobs for each table.




aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
Cara
N. Virginia
Support
AWS Glue
Data catalog
Databases
Tables
Connections
Crawlers
Classifiers
Schema registries
Schemas
Settings
ETL
AWS Glue Studio
New
Blueprints
Workflows
Jobs
ML Transforms
Triggers
Dev endpoints
Notebooks
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.
Crawler "final-project-crawler" completed and made the following changes: 1 tables created, 0 tables updated. See the tables created in database final-database.
User preferences
Add crawler
Run crawler
Action
Filter by tags and attributes
Showing: 1 - 1
Name
Schedule
Status
Logs
Last runtime
Median runtime
Tables updated
Tables added
final-project-crawler
Ready
Logs
53 secs
53 secs
0
1

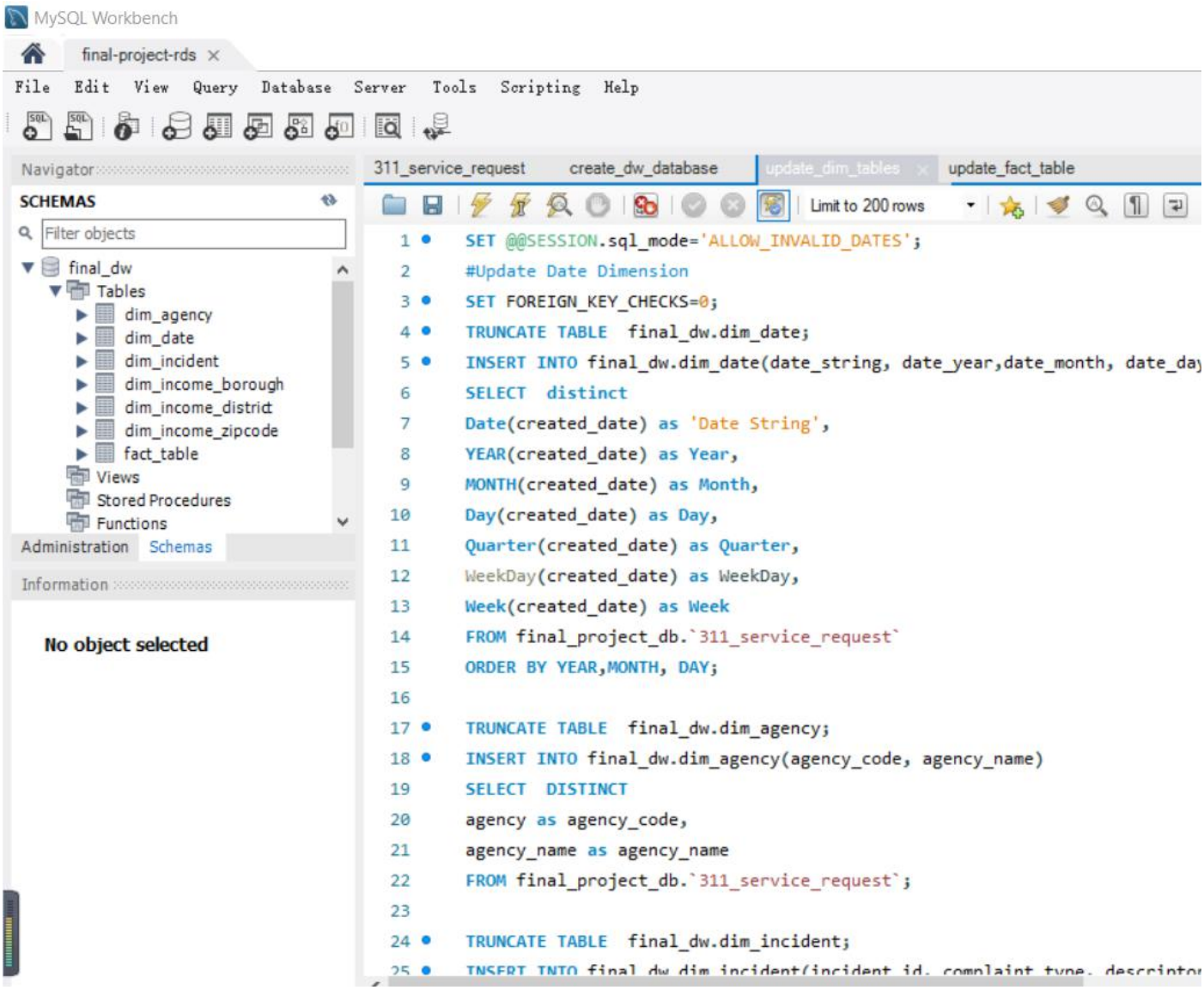

aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
Cara
N. Virginia
Support
AWS Glue
Data catalog
Databases
Tables
Connections
Crawlers
Classifiers
Schema registries
Schemas
Settings
ETL
AWS Glue Studio
New
Blueprints
Workflows
Jobs
ML Transforms
Triggers
Dev endpoints
Notebooks
Jobs A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.
User preferences
Add job
Monitor job runs
Action
Filter by tags and attributes
Showing: 1 - 4
Name
Type
ETL language
Script location
Last modified
Job bookmark
load-311-to-rds-mysql
Spark
python
s3://aws-glue-s…
18 August 2021 7:28 PM …
Disable
nyc_BOROUGHS_median_income_info
Spark
python
s3://aws-glue-s…
17 August 2021 2:48 PM …
Disable
nyc_DISTRICTS_median_income_info
Spark
python
s3://aws-glue-s…
17 August 2021 2:56 PM …
Disable
nyc_ZipCodes_median_income_info
Spark
python
s3://aws-glue-s…
17 August 2021 2:52 PM …
Disable


aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
Cara
N. Virginia
Support
AWS Glue
Data catalog
Databases
Tables
Connections
Crawlers
Classifiers
Schema registries
Schemas
Settings
ETL
AWS Glue Studio
New
Blueprints
Workflows
Jobs
ML Transforms
Triggers
Dev endpoints
Notebooks
Workflows (2)
A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.
Add workflow
Actions
Filter workflows
1
Name
Last run
Last run status
Last modified
Monthly_uodate-nyc-median-in…
-
-
Tue, 17 Aug 2021 22:00:58 GMT
Daily_update-311-service-requ…
Thu, 19 Aug 2021 00:1…
Completed
Tue, 17 Aug 2021 22:00:34 GMT
scheduled daily run time
final-project-cr awler
ANY
successfully-cr awler
load-311data-t o-rds-job
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

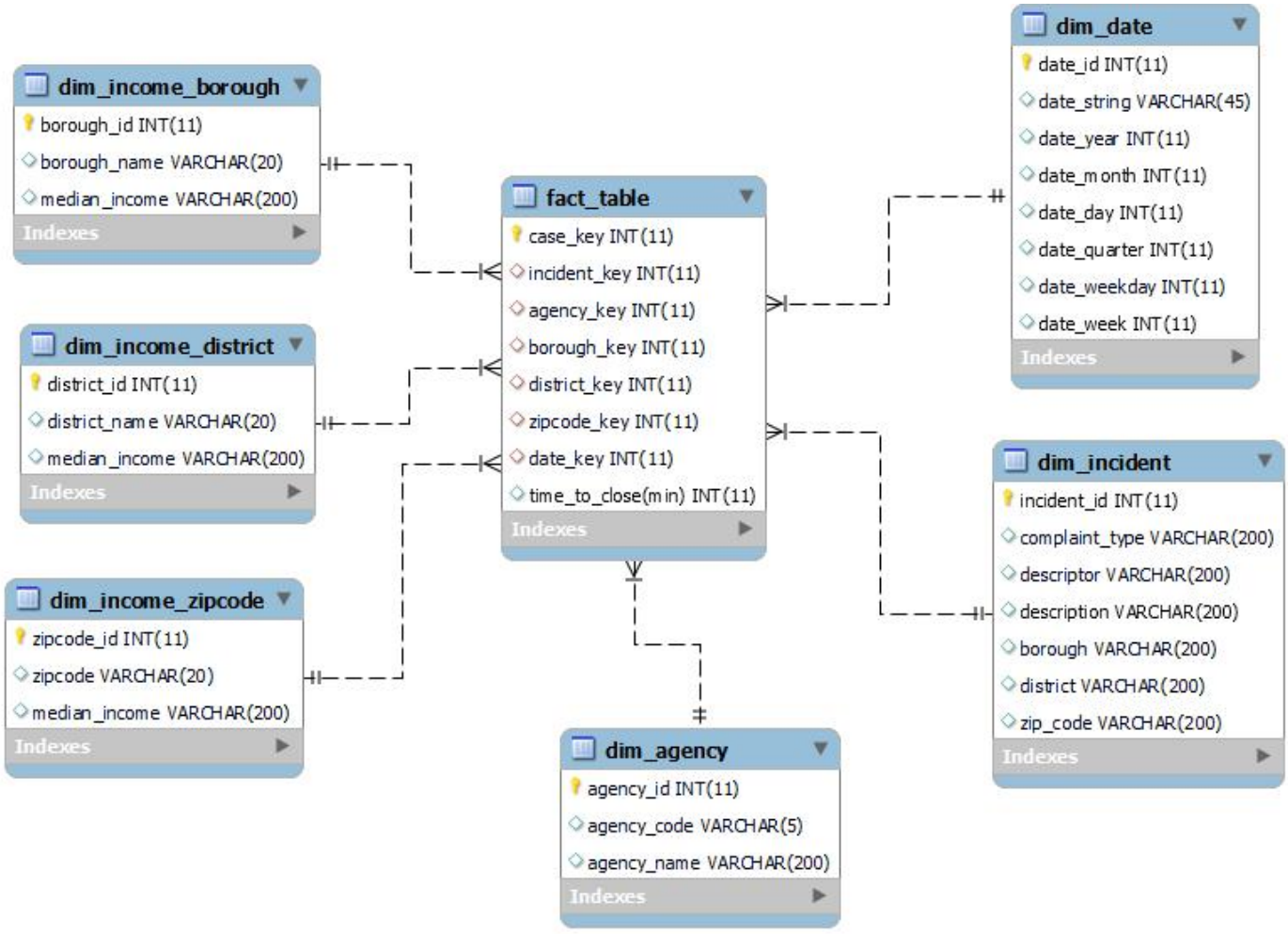

aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
Cara
N. Virginia
Support
AWS Glue
Data catalog
Databases
Tables
Connections
Crawlers
Classifiers
Schema registries
Schemas
Settings
ETL
AWS Glue Studio
New
Blueprints
Workflows
Jobs
ML Transforms
Triggers
Dev endpoints
Notebooks
Workflows (2)
A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.
Add workflow
Actions
Filter workflows
1
Name
Last run
Last run status
Last modified
Monthly_uodate-nyc-median-in…
-
-
Tue, 17 Aug 2021 22:00:58 GMT
Daily_update-311-service-requ…
Thu, 19 Aug 2021 00:1…
Completed
Tue, 17 Aug 2021 22:00:34 GMT
schedule monthly run…
final-project-cr awler
ANY
successfully crawler
nyc_BOROUG HS_median_i…
ANY
successfully run…
nyc_DISTRICT S_median_in…
ANY
successfully run DISTRICT
nyc_ZipCodes _median_inc…

9. Database Reverse Engineer ER diagram for Resource database `final_project_db`

10. Create Star schema Data warehouse DDL in sql, create update the dimensional table DDL, create update fact table DDL.



11. Database Reverse Engineer ER diagram for star schema Data Warehouse Database `final_dw`

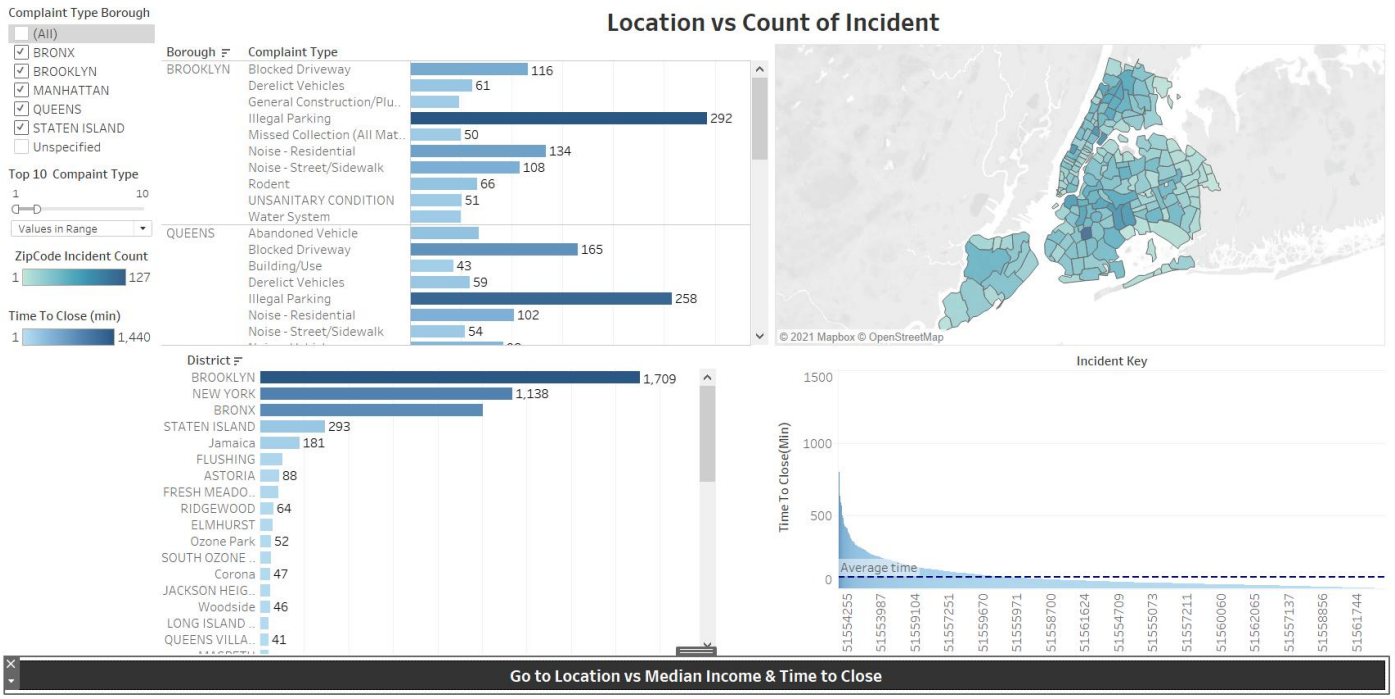


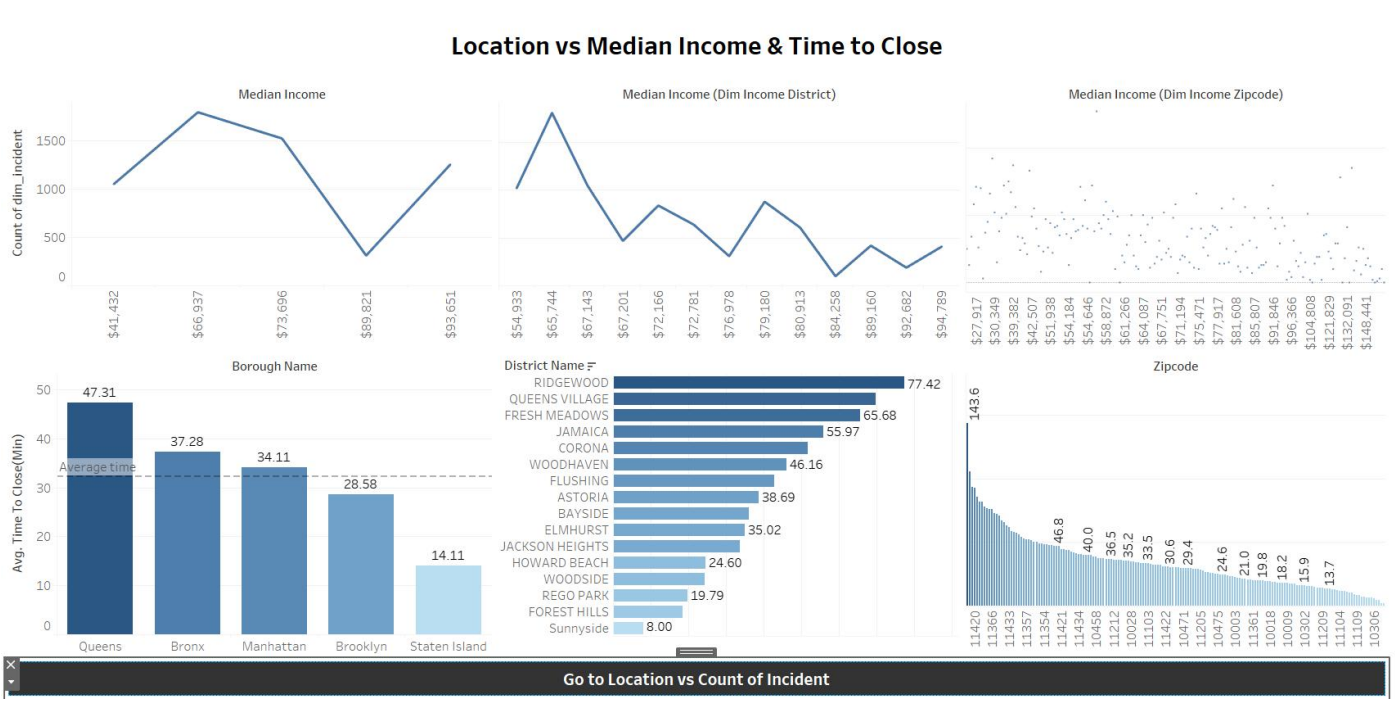
12. Using python connect to the RDS MySql and schedule to run the updated data warehouse sql script daily time. Connecting to the gmail module and send the notification log when it successfully updated data warehouse.

```
##Test query to see if you can connect
query = "select * from final_dw.dim_date;"
cursor.execute(query)
result = cursor.fetchall()
print(result)

#Call Dimension Tables Stored Procedure
with open("./update_dim_tables.sql", encoding="utf-8") as f:
    commands = f.read().split(';')

for command in commands:
    cursor.execute(command)
    print(command)

print('Dimension tables updated.')


#Call Fact Tables Stored Procedure
with open("./update_fact_table.sql", encoding="utf-8") as f:
    commands = f.read().split(';')

for command in commands:
    cursor.execute(command)
    print(command)

print('Fact tables updated.')


connection.commit()
connection.close()
print('Disconnected from database.')
```

```
sent_from = gmail_user
to = ['xiaolancara@gmail.com','bhcooper@mail.yu.edu']
subject = 'Batch Process Completed on: ' + now.strftime("%m/%d/%Y - %H:%M:%S")
body = 'Your Final Project Batch Job Processed with 0 Errors'

email_text = """\
From: %s
To: %s
Subject: %s

%s
""" % (sent_from, to, subject, body)

try:
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login(gmail_user, gmail_password)
    server.sendmail(sent_from, to, email_text)
    server.close()
    print('Email successfully sent')
except:
    print('Error: your email did not send')
```

13. Using Tableau to Analyze the data warehouse





Analysis conclusion:

- Illegal Parking is the highest frequent incident in service request.
- Brooklyn, Newyork and Bronx are the highest frequent service request districts.
- Zipcode 11226 has the highest frequent service request.
- The distribution of time to close in all incidents are right skew with a long tail.
- There's no strong correlation between median income and Borough as well as in zipcode but has a negative correlation with districts.
- Borough Queens, District Ridgewood, zipcode 11411 have highest average time to close incidents.