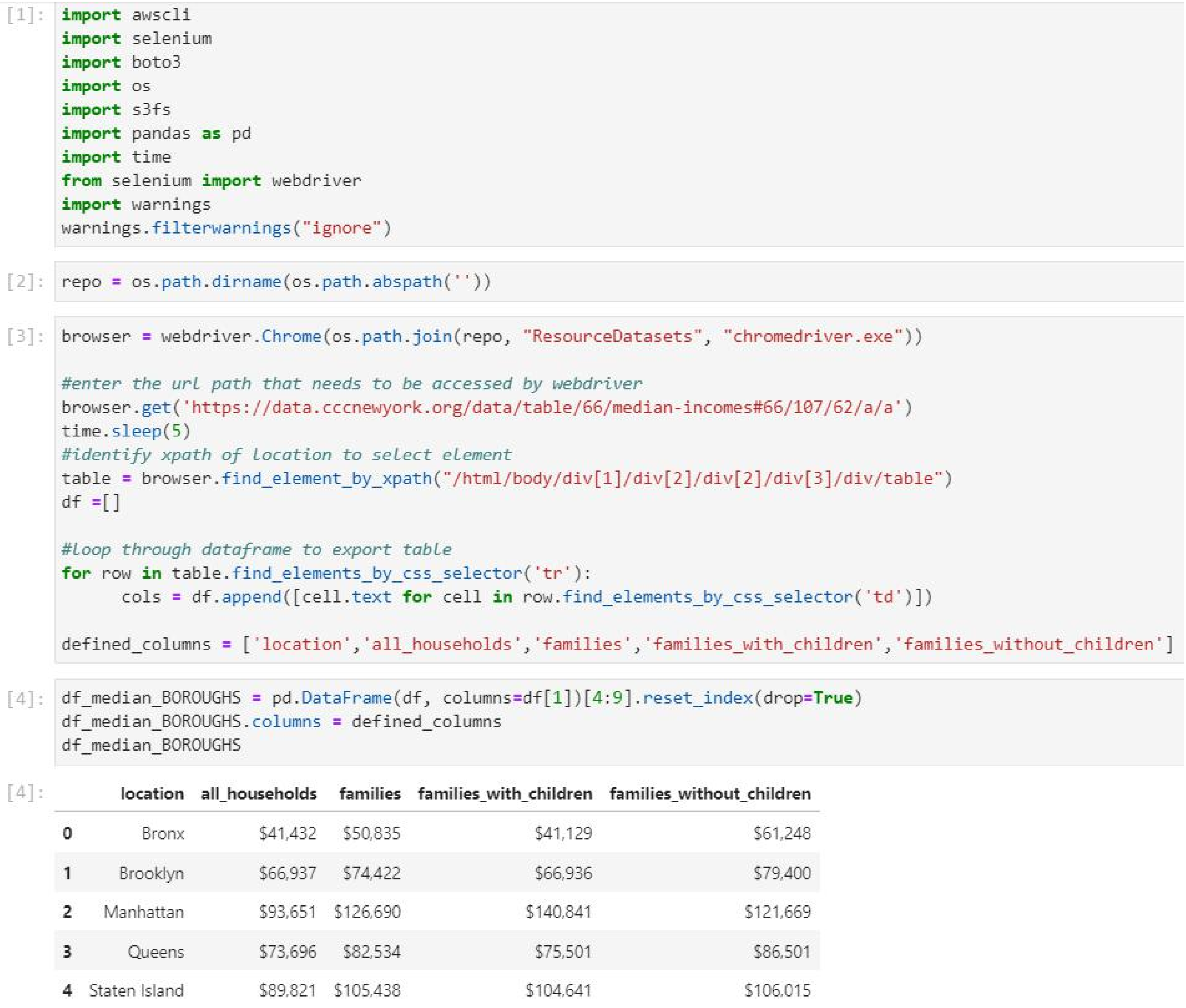
6100 Information Architecture Summer 2021 Final Project Report

Student: Xiaolan Li, Bernard Cooper

Professor: Brandon Chiazza

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']
```

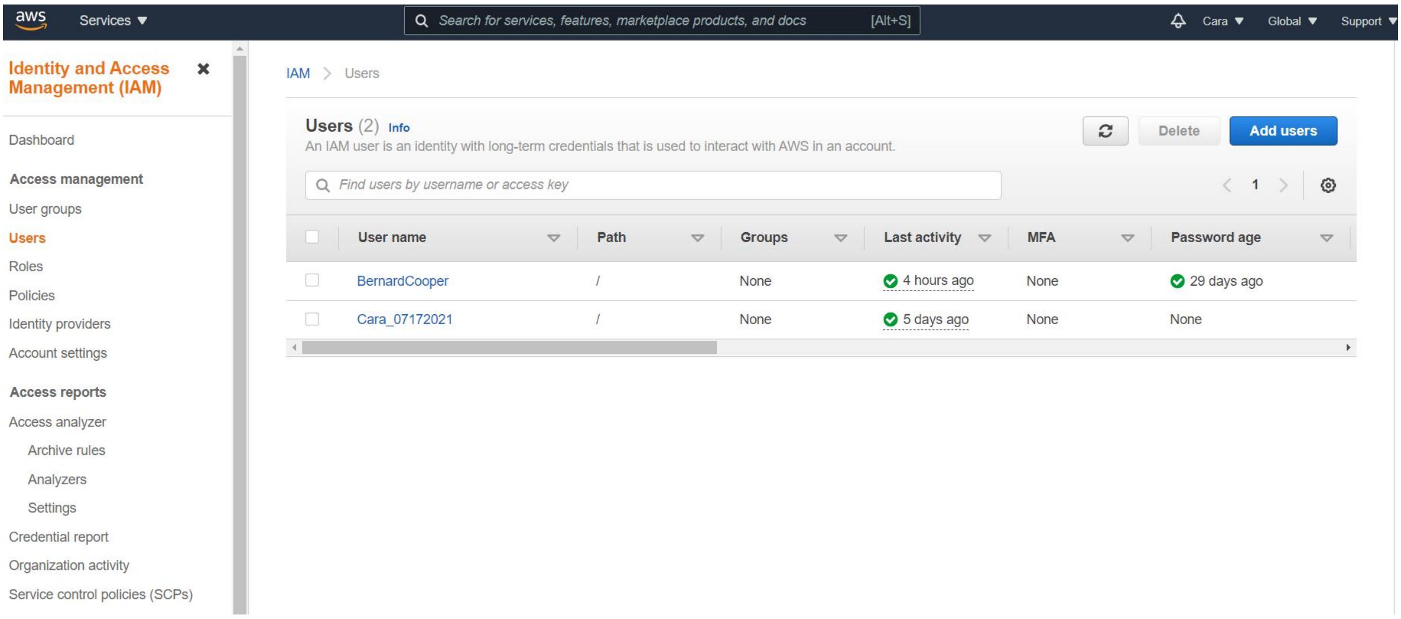
[4]:

```
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



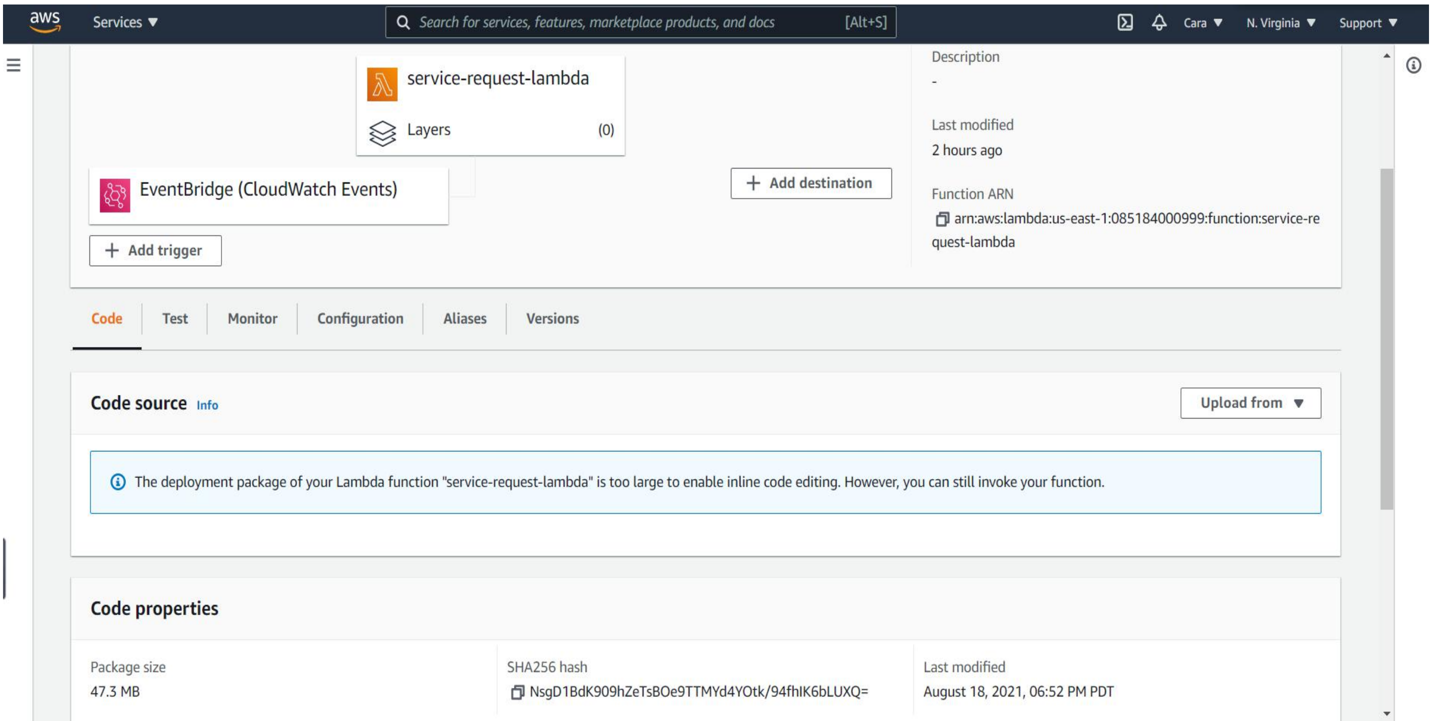
Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age
BernardCooper	/	None	✓ 4 hours ago	None	✓ 29 days ago
Cara_07172021	/	None	✓ 5 days ago	None	None

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



Description
-

Last modified
2 hours ago

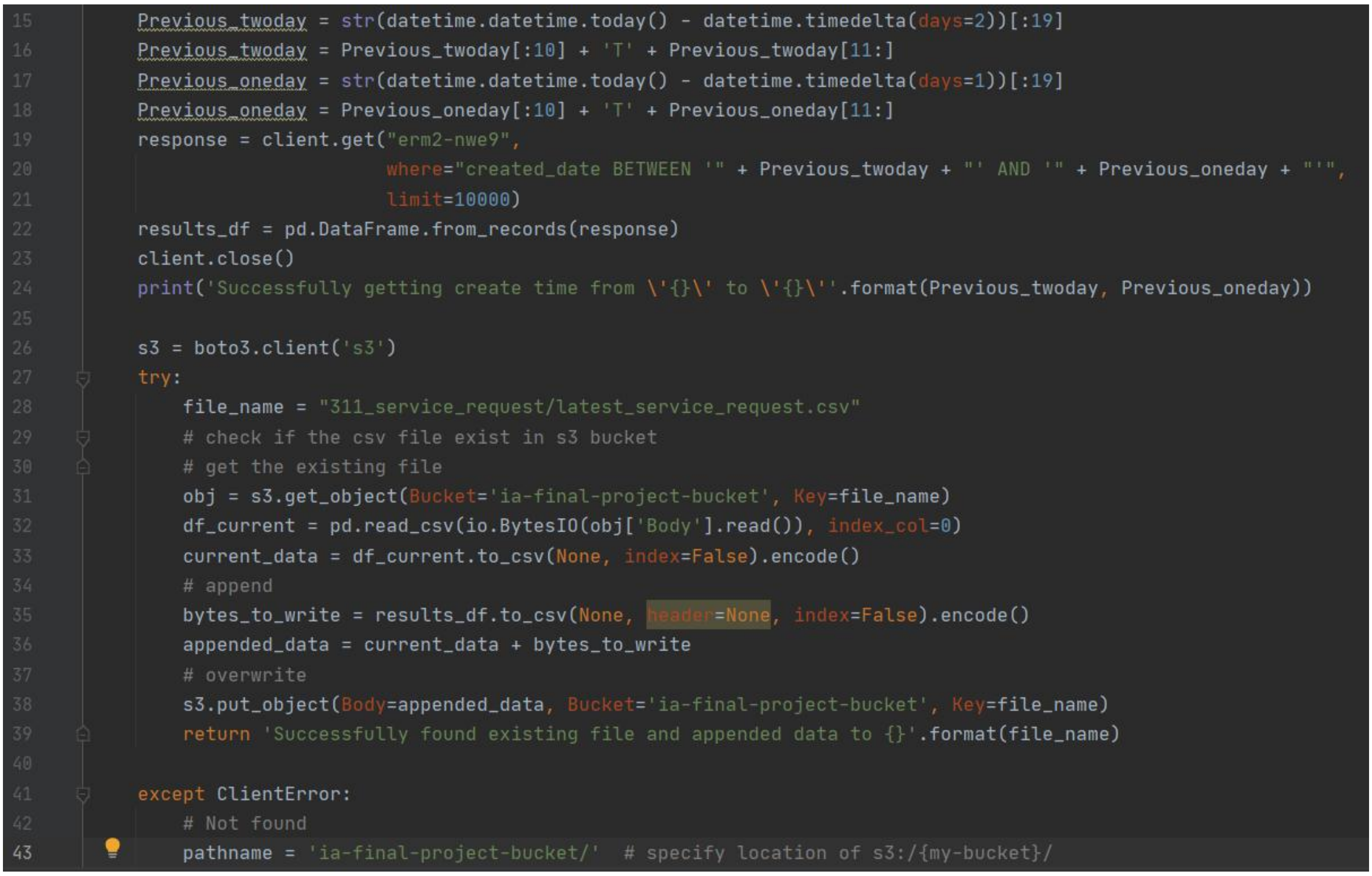
Function ARN
arn:aws:lambda:us-east-1:085184000999:function:service-request-lambda

Code source Info

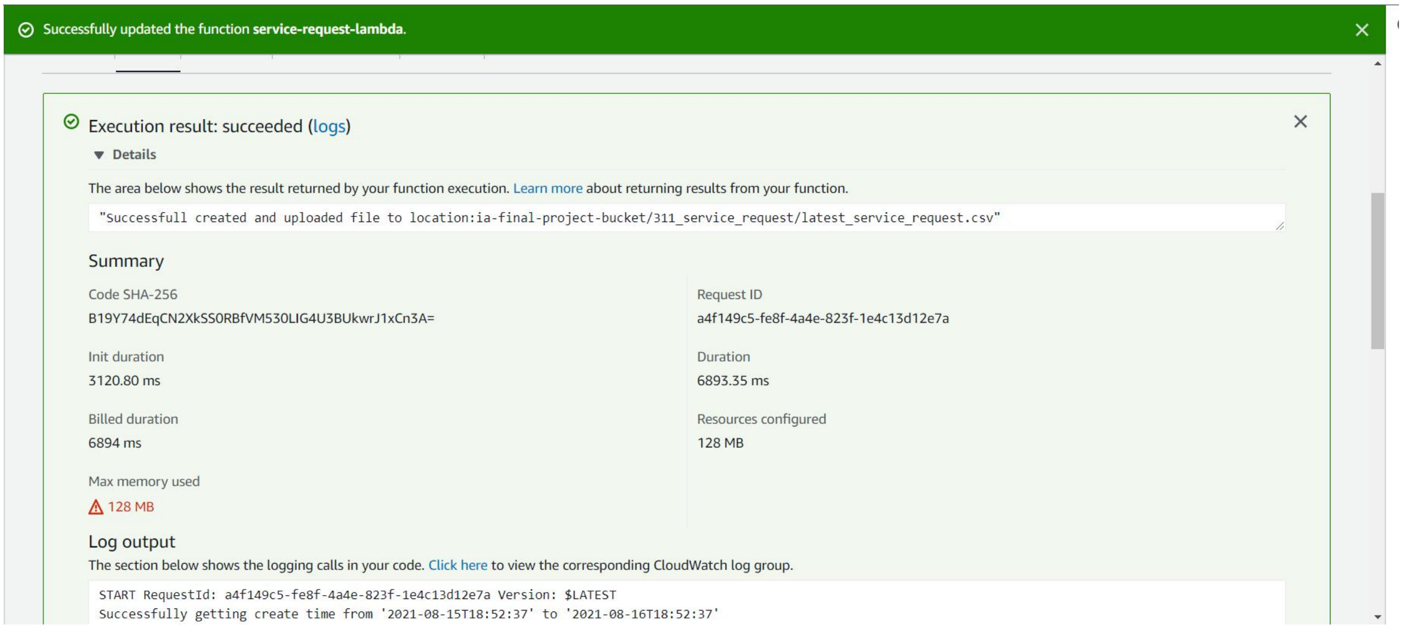
The deployment package of your Lambda function "service-request-lambda" is too large to enable inline code editing. However, you can still invoke your function.

Code properties

Package size	SHA256 hash	Last modified
47.3 MB	NsgD1BdK909hZeTsBOe9TTMYd4YOtk/94fhIK6bLUXQ=	August 18, 2021, 06:52 PM PDT



```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}/
```



Successfully updated the function **service-request-lambda**.

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
"Successfull created and uploaded file to location:ia-final-project-bucket/311_service_request/latest_service_request.csv"
```

Summary

Code SHA-256
B19Y74dEqCN2XkSS0RBfVM530LIG4U3BUkwrJ1xCn3A=

Request ID
a4f149c5-fe8f-4a4e-823f-1e4c13d12e7a

Init duration
3120.80 ms

Duration
6893.35 ms

Billed duration
6894 ms

Resources configured
128 MB

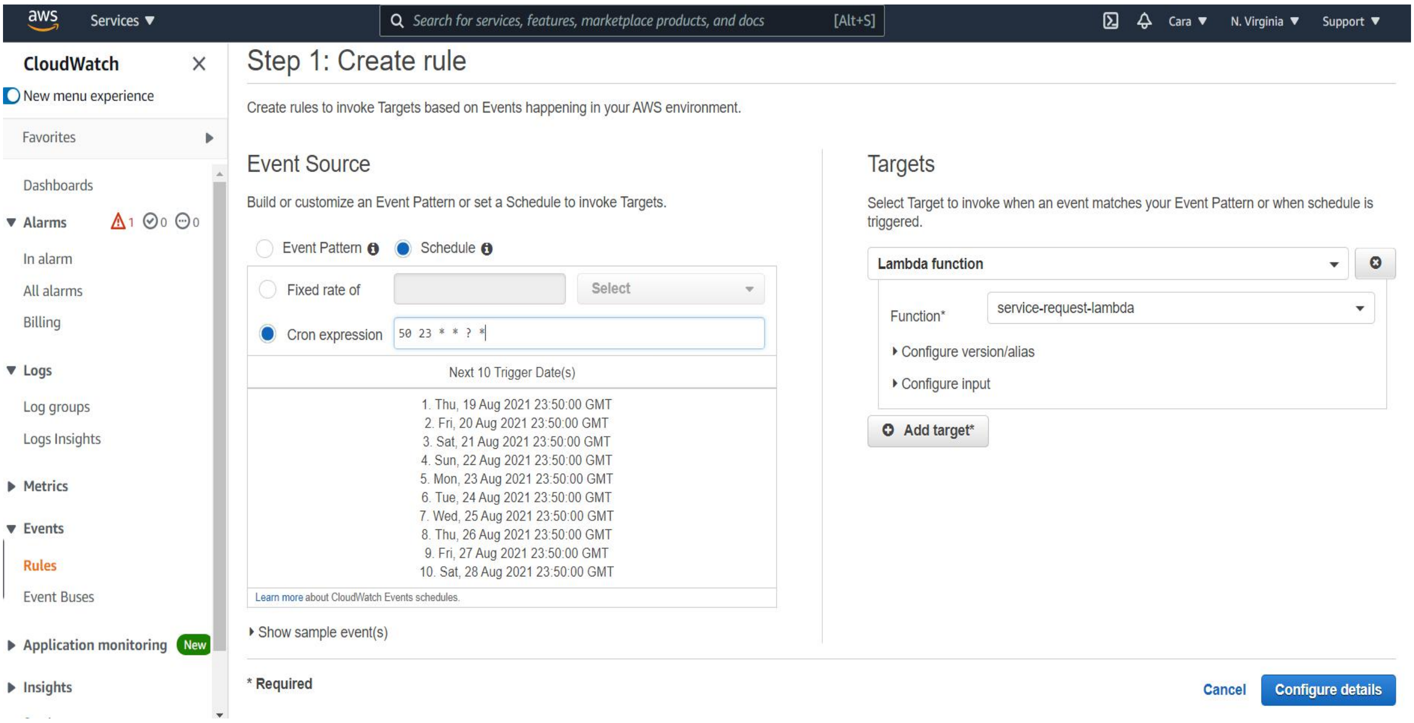
Max memory used
⚠ 128 MB

Log output

The section below shows the logging calls in your code. Click here to view the corresponding CloudWatch log group.

```
START RequestId: a4f149c5-fe8f-4a4e-823f-1e4c13d12e7a Version: $LATEST
Successfully getting create time from '2021-08-15T18:52:37' to '2021-08-16T18:52:37'
```

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



Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

○ Event Pattern ● Schedule

○ Fixed rate of [ ] Select

● Cron expression `50 23 * * ? *`

Next 10 Trigger Date(s)

1. Thu, 19 Aug 2021 23:50:00 GMT
2. Fri, 20 Aug 2021 23:50:00 GMT
3. Sat, 21 Aug 2021 23:50:00 GMT
4. Sun, 22 Aug 2021 23:50:00 GMT
5. Mon, 23 Aug 2021 23:50:00 GMT
6. Tue, 24 Aug 2021 23:50:00 GMT
7. Wed, 25 Aug 2021 23:50:00 GMT
8. Thu, 26 Aug 2021 23:50:00 GMT
9. Fri, 27 Aug 2021 23:50:00 GMT
10. Sat, 28 Aug 2021 23:50:00 GMT

Learn more about CloudWatch Events schedules.

▸ Show sample event(s)

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

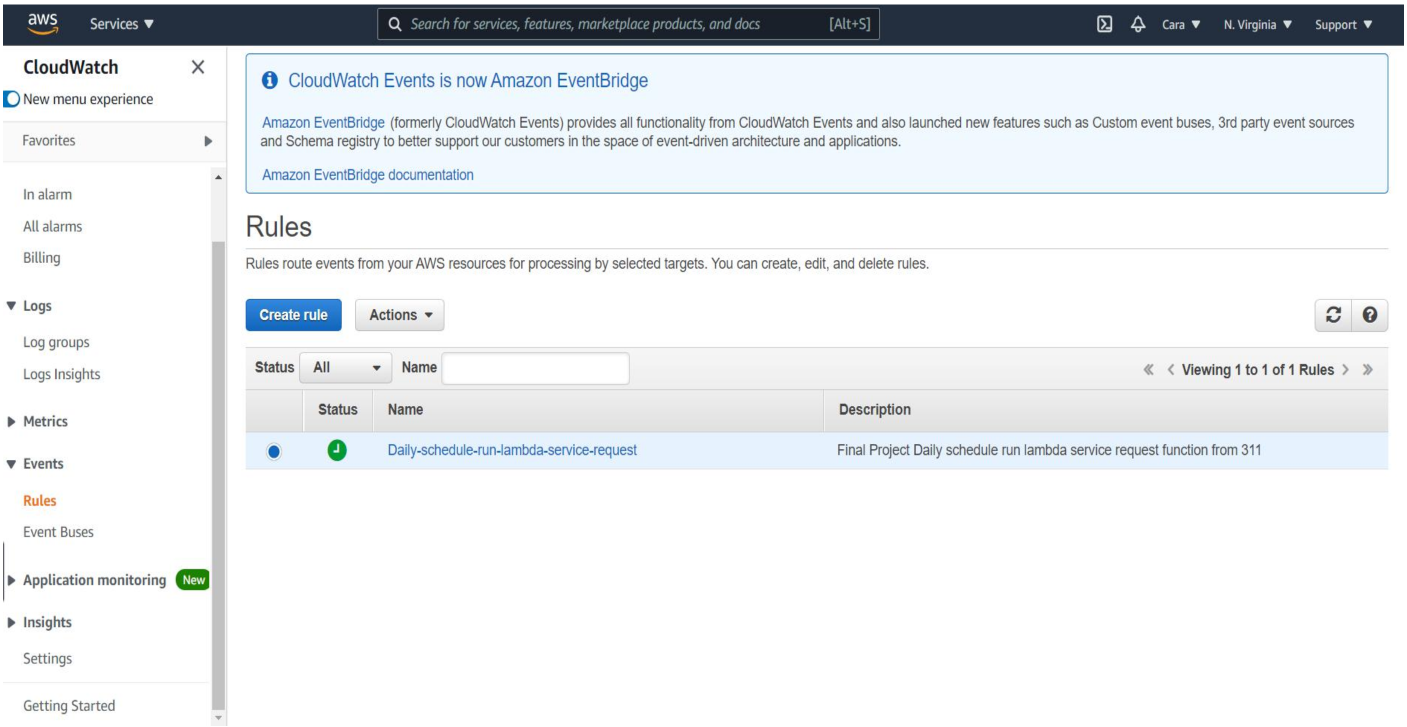
Lambda function

Function*: service-request-lambda

▸ Configure version/alias

▸ Configure input

* Required



CloudWatch Events is now Amazon EventBridge

Amazon EventBridge (formerly CloudWatch Events) provides all functionality from CloudWatch Events and also launched new features such as Custom event buses, 3rd party event sources and Schema registry to better support our customers in the space of event-driven architecture and applications.

Amazon EventBridge documentation

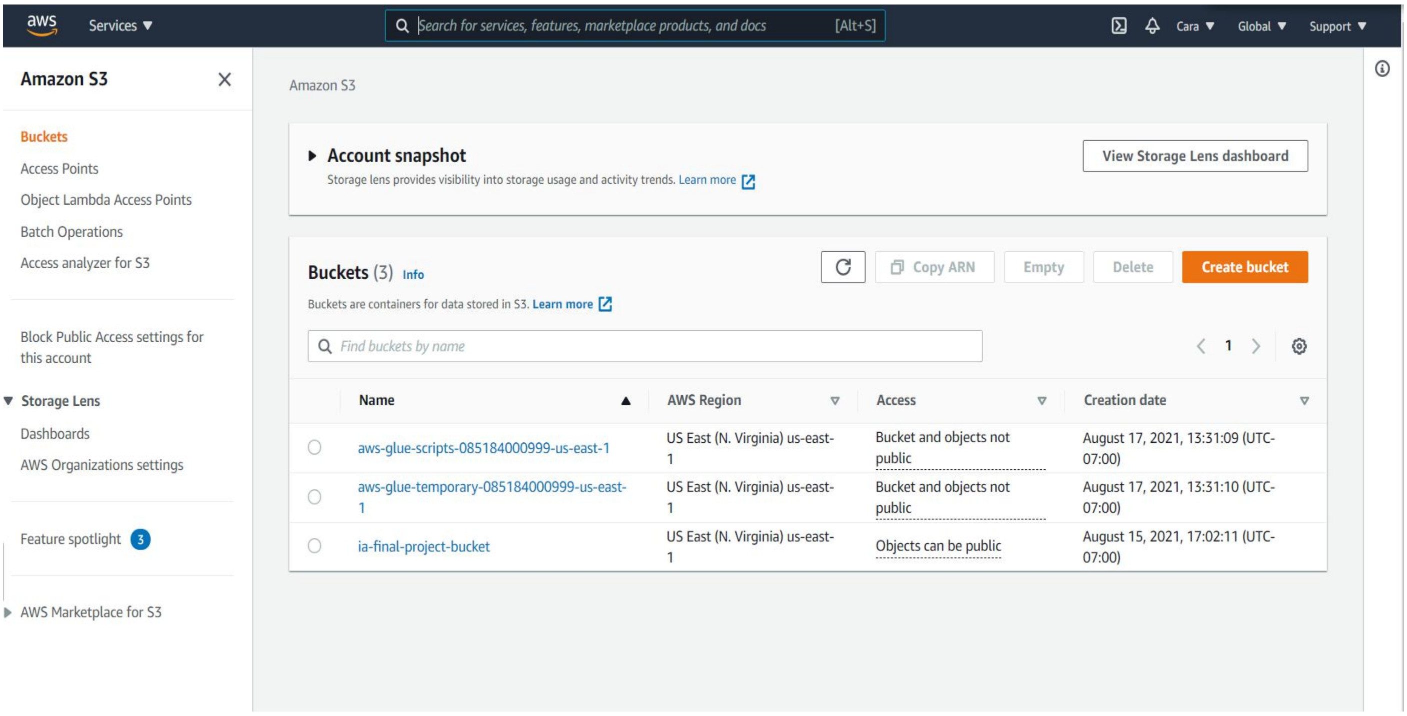
Rules

Rules route events from your AWS resources for processing by selected targets. You can create, edit, and delete rules.

Status: All   Name: Viewing 1 to 1 of 1 Rules

	Status	Name	Description
●	✓	Daily-schedule-run-lambda-service-request	Final Project Daily schedule run lambda service request function from 311

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



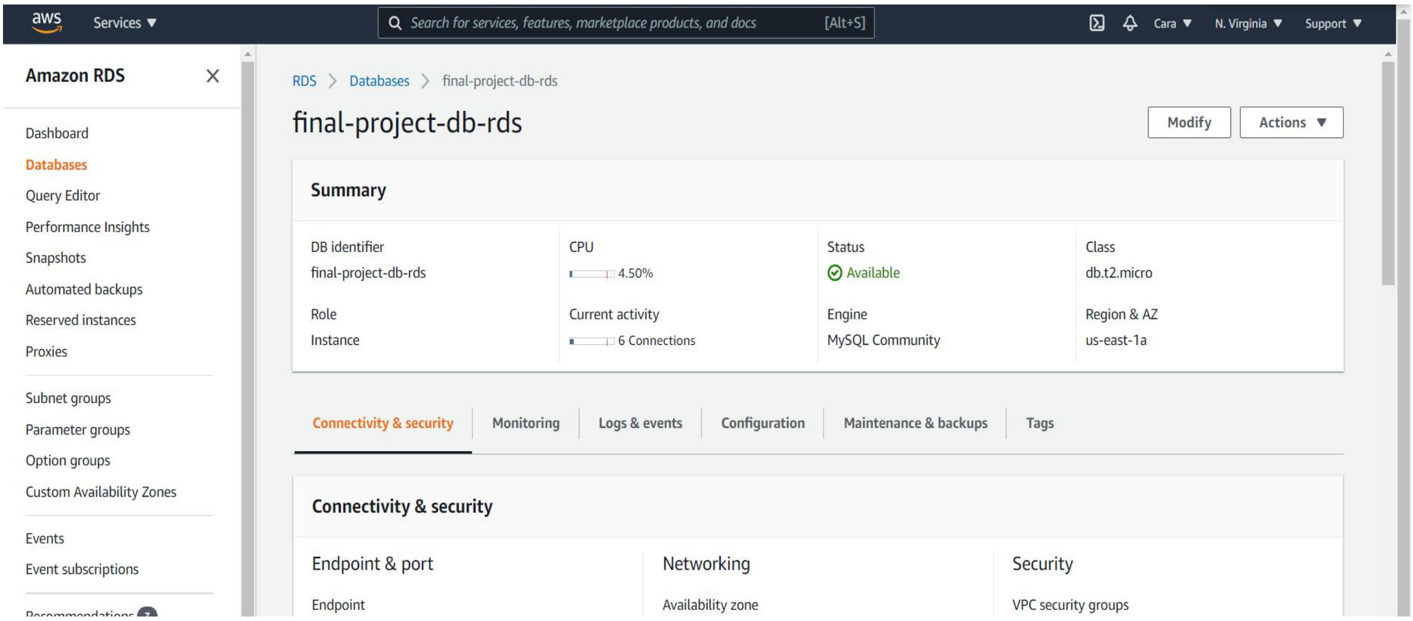
Storage lens provides visibility into storage usage and activity trends. Learn more

Buckets (3) Info

Buckets are containers for data stored in S3. Learn more

	Name	AWS Region	Access	Creation date
○	aws-glue-scripts-085184000999-us-east-1	US East (N. Virginia) us-east-1	Bucket and objects not public	August 17, 2021, 13:31:09 (UTC-07:00)
○	aws-glue-temporary-085184000999-us-east-1	US East (N. Virginia) us-east-1	Bucket and objects not public	August 17, 2021, 13:31:10 (UTC-07:00)
○	ia-final-project-bucket	US East (N. Virginia) us-east-1	Objects can be public	August 15, 2021, 17:02:11 (UTC-07:00)

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



final-project-db-rds

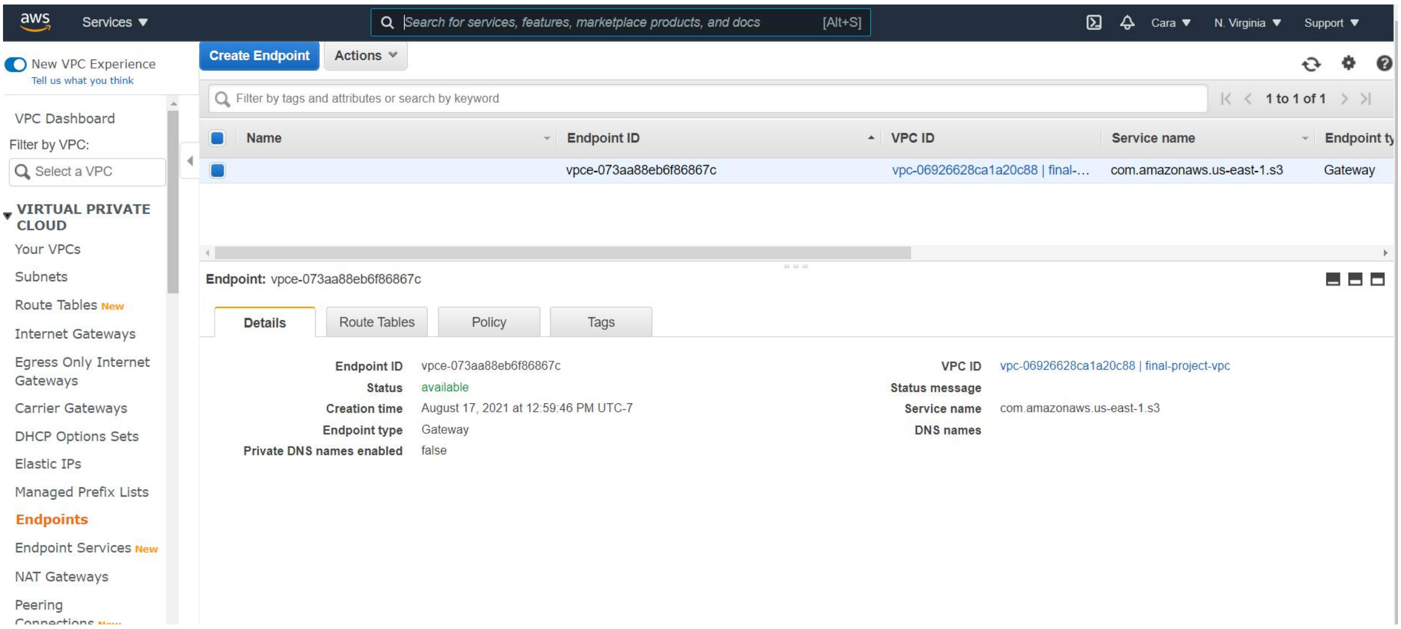
Summary

DB identifier	CPU	Status	Class
final-project-db-rds	4.50%	✓ Available	db.t2.micro
Role	**Current activity**	**Engine**	**Region & AZ**
Instance	6 Connections	MySQL Community	us-east-1a

Connectivity & security

Endpoint & port	Networking	Security
Endpoint	Availability zone	VPC security groups

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

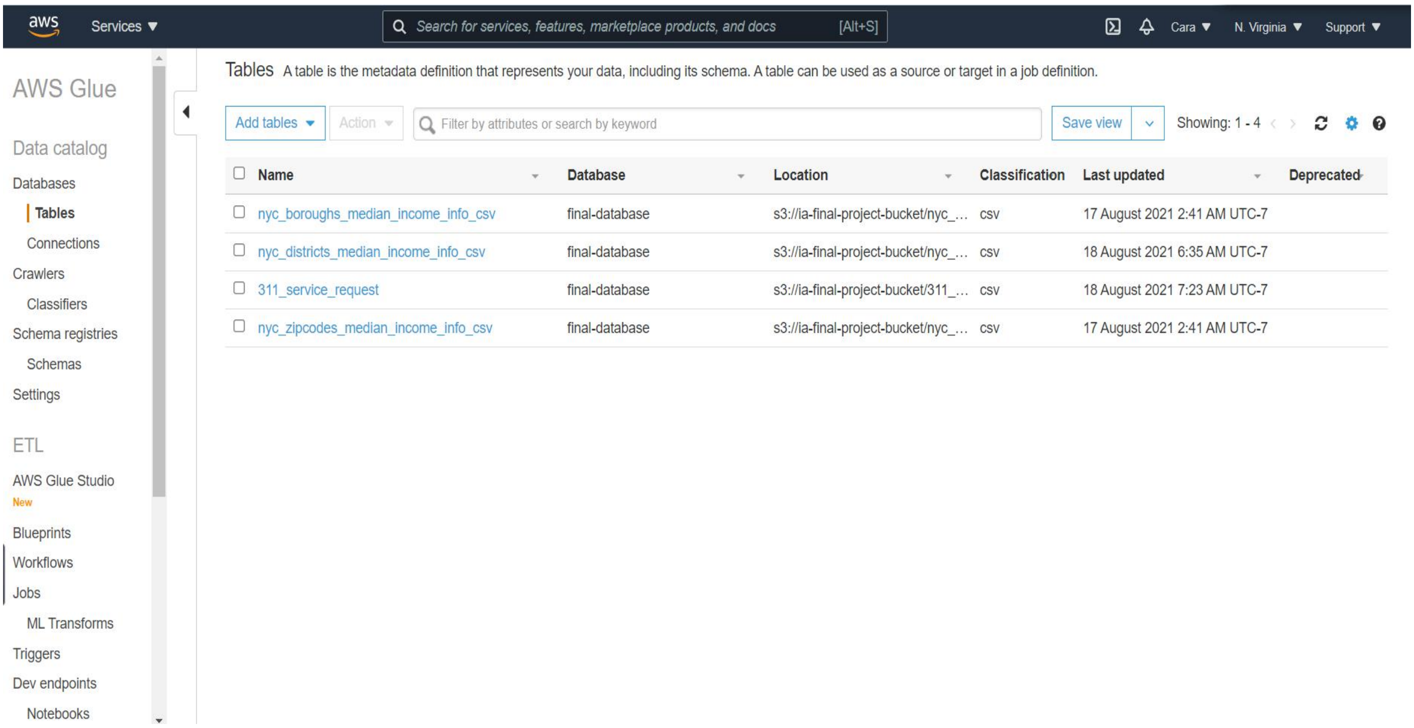


Name	Endpoint ID	VPC ID	Service name	Endpoint ty
	vpce-073aa88eb6f86867c	vpc-06926628ca1a20c88 | final-…	com.amazonaws.us-east-1.s3	Gateway

Endpoint: vpce-073aa88eb6f86867c

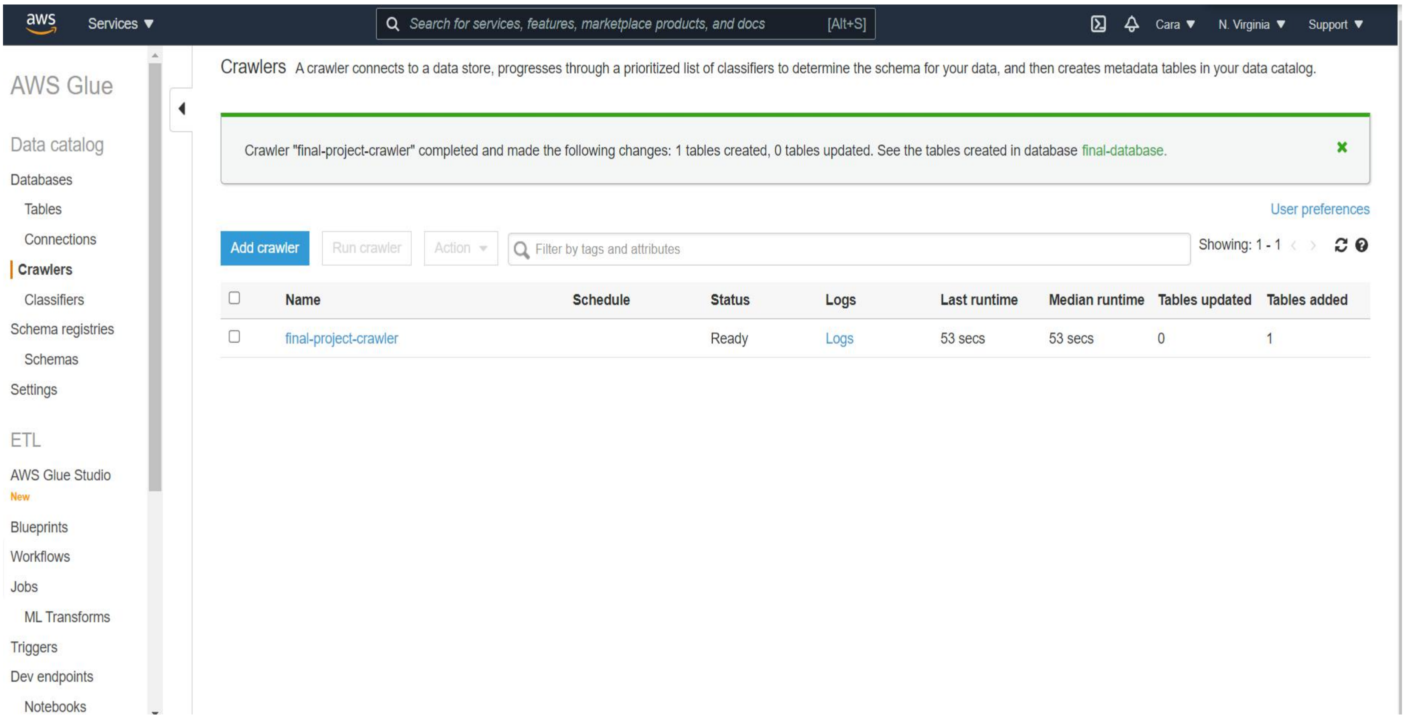
Endpoint ID	vpce-073aa88eb6f86867c	VPC ID	vpc-06926628ca1a20c88 | final-project-vpc
Status	available	Status message	
Creation time	August 17, 2021 at 12:59:46 PM UTC-7	Service name	com.amazonaws.us-east-1.s3
Endpoint type	Gateway	DNS names	
Private DNS names enabled	false		

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.



Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

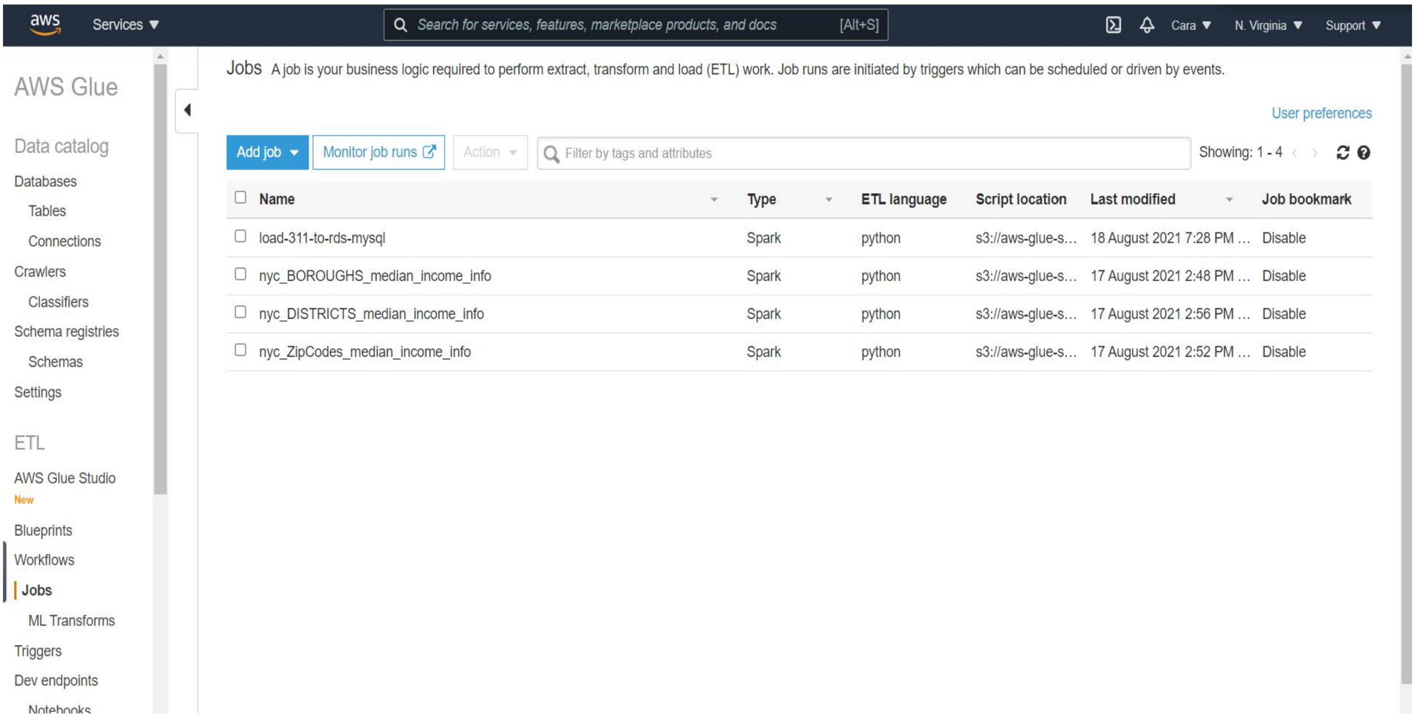
Name	Database	Location	Classification	Last updated	Deprecated
nyc_boroughs_median_income_info_csv	final-database	s3://ia-final-project-bucket/nyc_…	csv	17 August 2021 2:41 AM UTC-7	
nyc_districts_median_income_info_csv	final-database	s3://ia-final-project-bucket/nyc_…	csv	18 August 2021 6:35 AM UTC-7	
311_service_request	final-database	s3://ia-final-project-bucket/311_…	csv	18 August 2021 7:23 AM UTC-7	
nyc_zipcodes_median_income_info_csv	final-database	s3://ia-final-project-bucket/nyc_…	csv	17 August 2021 2:41 AM UTC-7	



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.

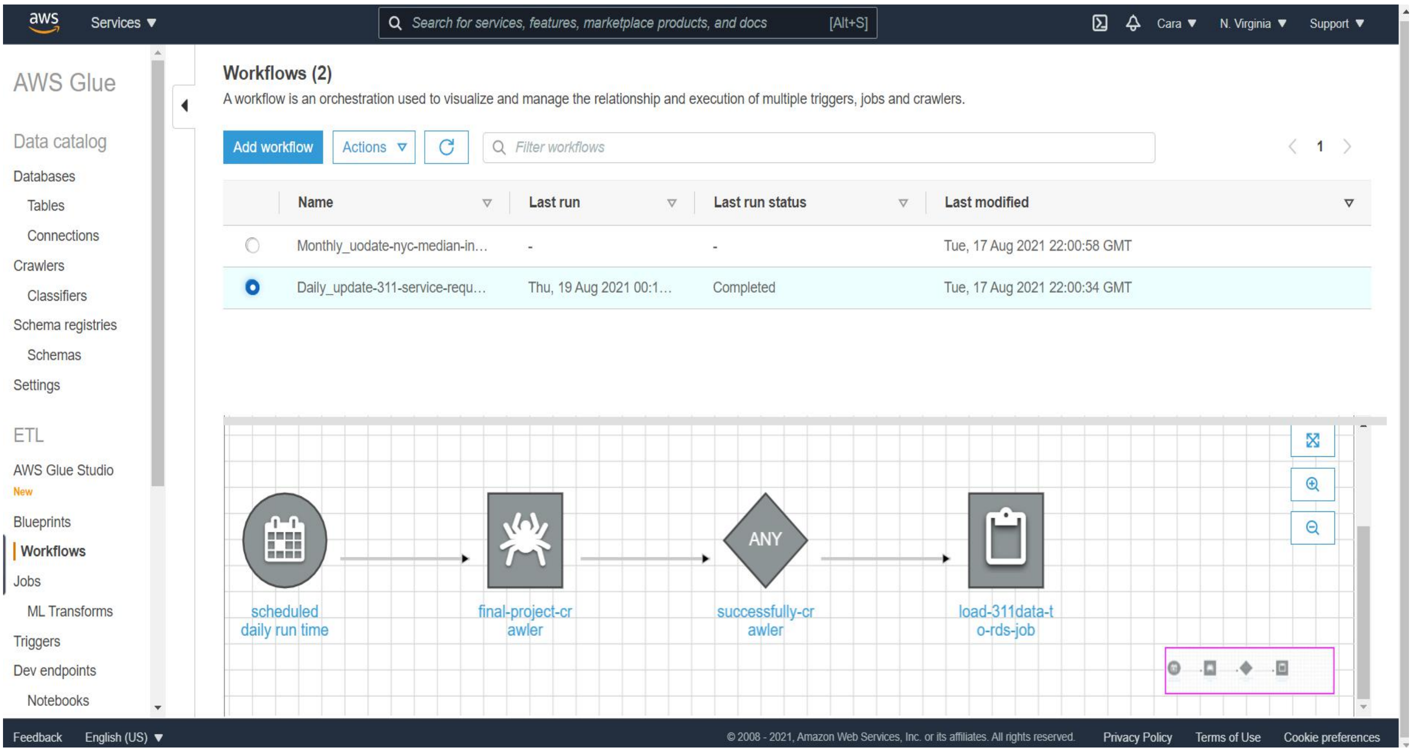
Crawler "final-project-crawler" completed and made the following changes: 1 tables created, 0 tables updated. See the tables created in database final-database.

□	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
□	final-project-crawler		Ready	Logs	53 secs	53 secs	0	1



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.

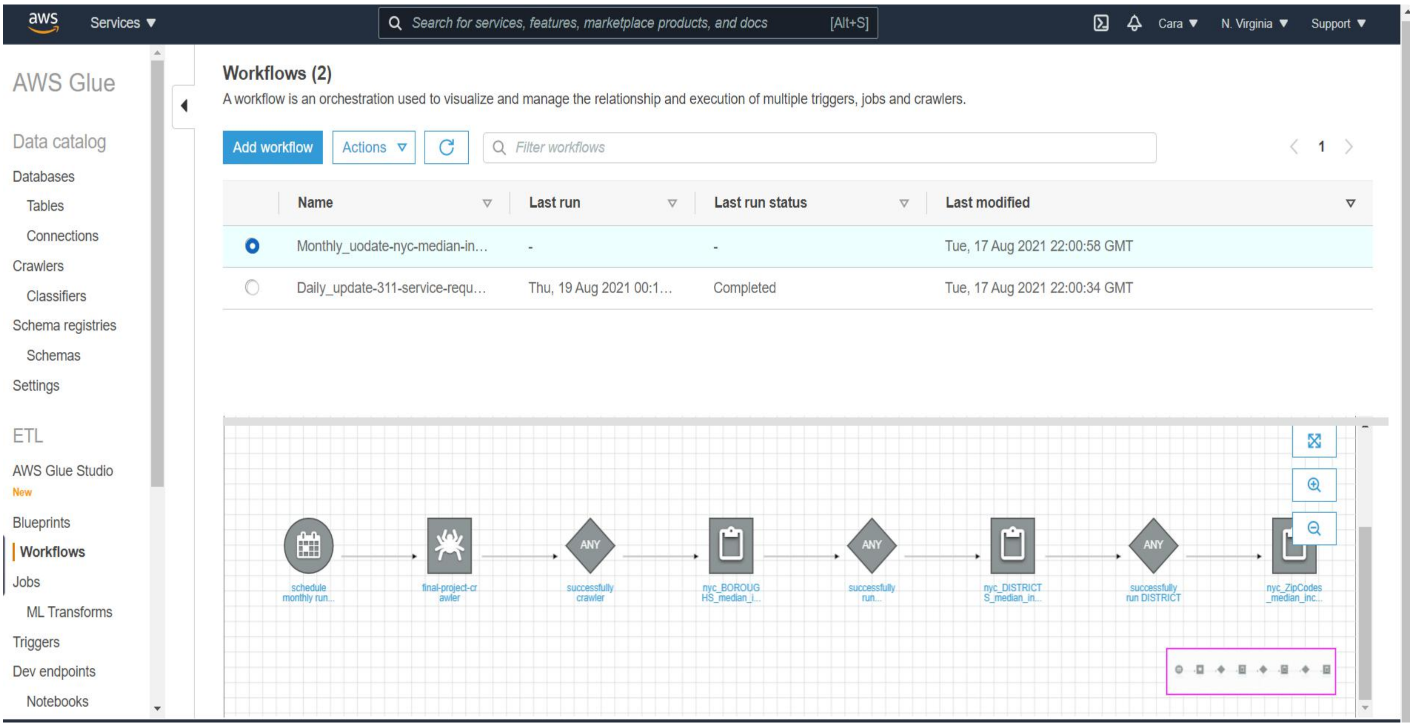
□	Name	Type	ETL language	Script location	Last modified	Job bookmark
□	load-311-to-rds-mysql	Spark	python	s3://aws-glue-s...	18 August 2021 7:28 PM ...	Disable
□	nyc_BOROUGHS_median_income_info	Spark	python	s3://aws-glue-s...	17 August 2021 2:48 PM ...	Disable
□	nyc_DISTRICTS_median_income_info	Spark	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



Workflows (2)

A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.

	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

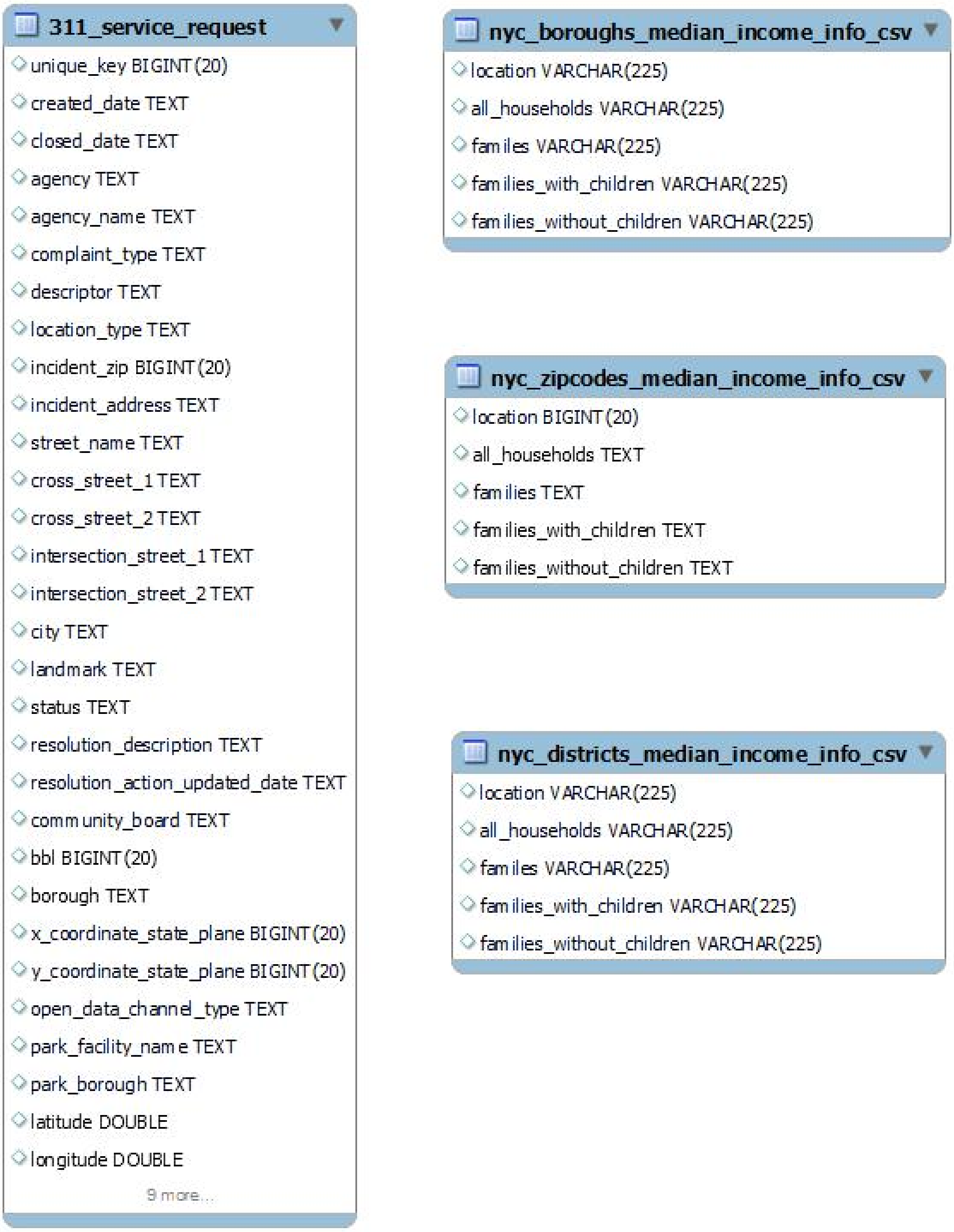


Workflows (2)

A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.

	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

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



311_service_request

- unique_key BIGINT(20)
- created_date TEXT
- closed_date TEXT
- agency TEXT
- agency_name TEXT
- complaint_type TEXT
- descriptor TEXT
- location_type TEXT
- incident_zip BIGINT(20)
- incident_address TEXT
- street_name TEXT
- cross_street_1 TEXT
- cross_street_2 TEXT
- intersection_street_1 TEXT
- intersection_street_2 TEXT
- city TEXT
- landmark TEXT
- status TEXT
- resolution_description TEXT
- resolution_action_updated_date TEXT
- community_board TEXT
- bbl BIGINT(20)
- borough TEXT
- x_coordinate_state_plane BIGINT(20)
- y_coordinate_state_plane BIGINT(20)
- open_data_channel_type TEXT
- park_facility_name TEXT
- park_borough TEXT
- latitude DOUBLE
- longitude DOUBLE

9 more...

nyc_boroughs_median_income_info_csv

- location VARCHAR(225)
- all_households VARCHAR(225)
- familes VARCHAR(225)
- families_with_children VARCHAR(225)
- families_without_children VARCHAR(225)

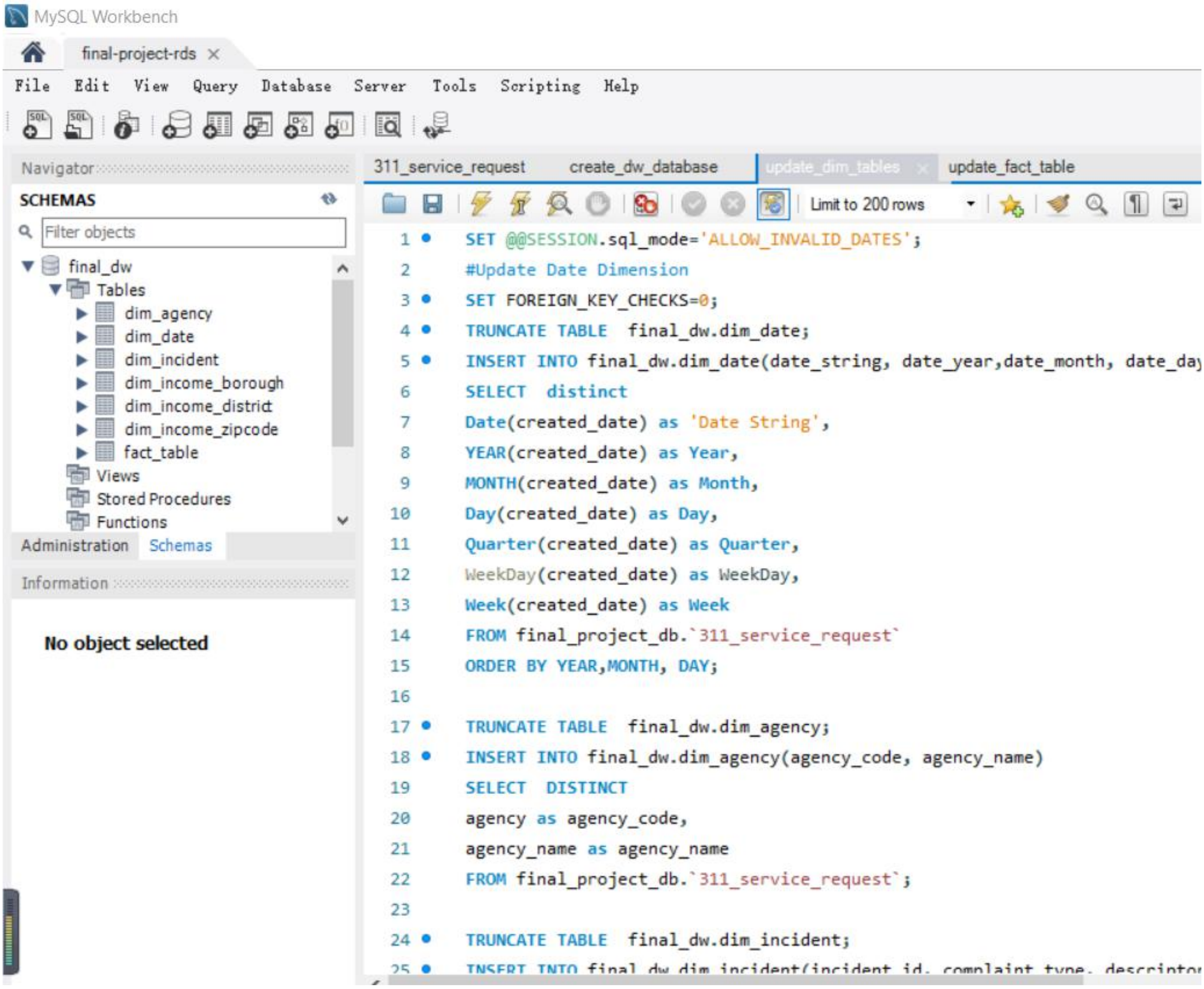
nyc_zipcodes_median_income_info_csv

- location BIGINT(20)
- all_households TEXT
- families TEXT
- families_with_children TEXT
- families_without_children TEXT

nyc_districts_median_income_info_csv

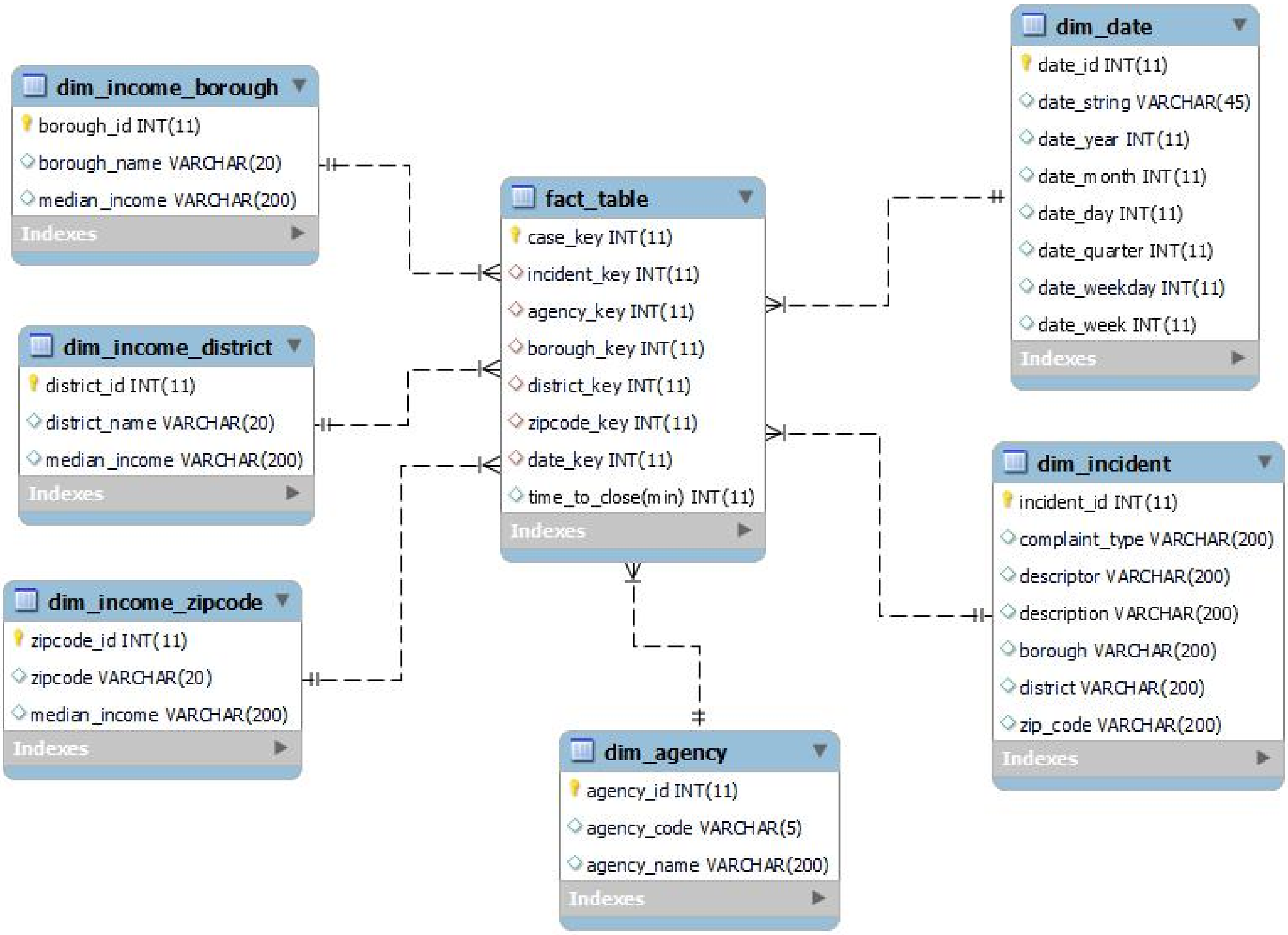
- location VARCHAR(225)
- all_households VARCHAR(225)
- familes VARCHAR(225)
- families_with_children VARCHAR(225)
- families_without_children VARCHAR(225)

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

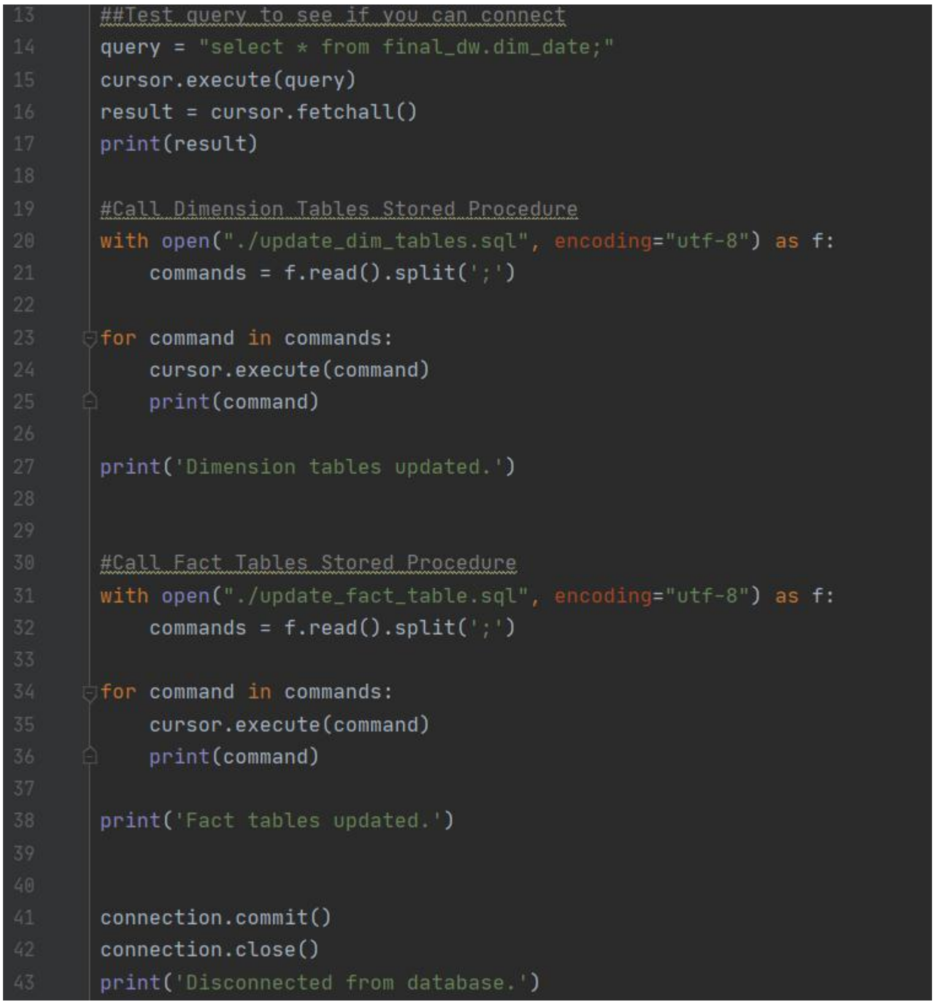


```
SET @@SESSION.sql_mode='ALLOW_INVALID_DATES';
#Update Date Dimension
SET FOREIGN_KEY_CHECKS=0;
TRUNCATE TABLE  final_dw.dim_date;
INSERT INTO final_dw.dim_date(date_string, date_year,date_month, date_day
SELECT  distinct
Date(created_date) as 'Date String',
YEAR(created_date) as Year,
MONTH(created_date) as Month,
Day(created_date) as Day,
Quarter(created_date) as Quarter,
WeekDay(created_date) as WeekDay,
Week(created_date) as Week
FROM final_project_db.`311_service_request`
ORDER BY YEAR,MONTH, DAY;

TRUNCATE TABLE  final_dw.dim_agency;
INSERT INTO final_dw.dim_agency(agency_code, agency_name)
SELECT  DISTINCT
agency as agency_code,
agency_name as agency_name
FROM final_project_db.`311_service_request`;

TRUNCATE TABLE  final_dw.dim_incident;
INSERT INTO final_dw.dim_incident(incident_id, complaint_type, descriptor
```

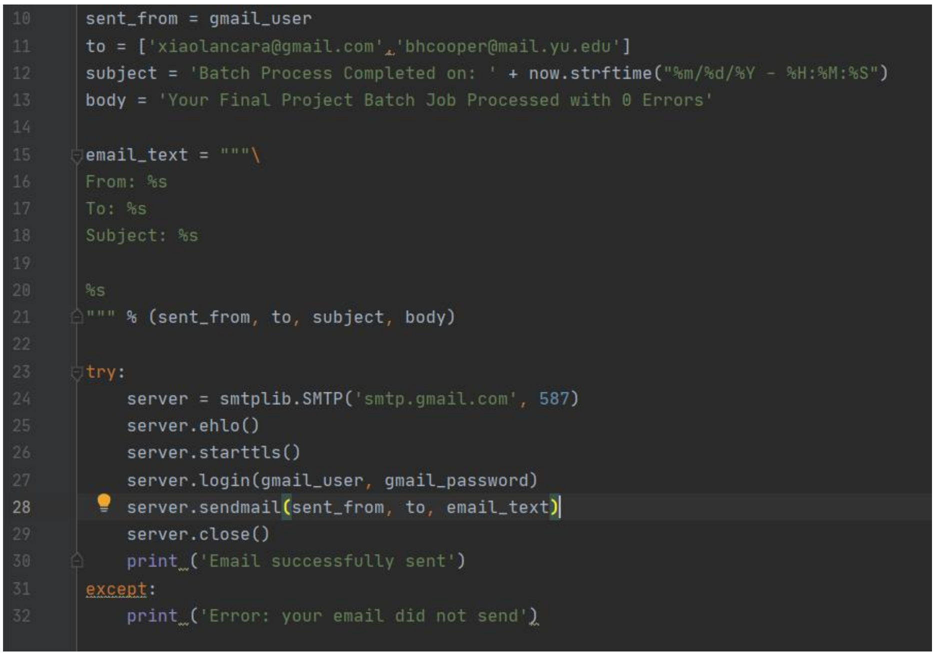
11. Database Reverse Engineer ER diagram for star schema Data Warehouse Database `final_dw`



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.

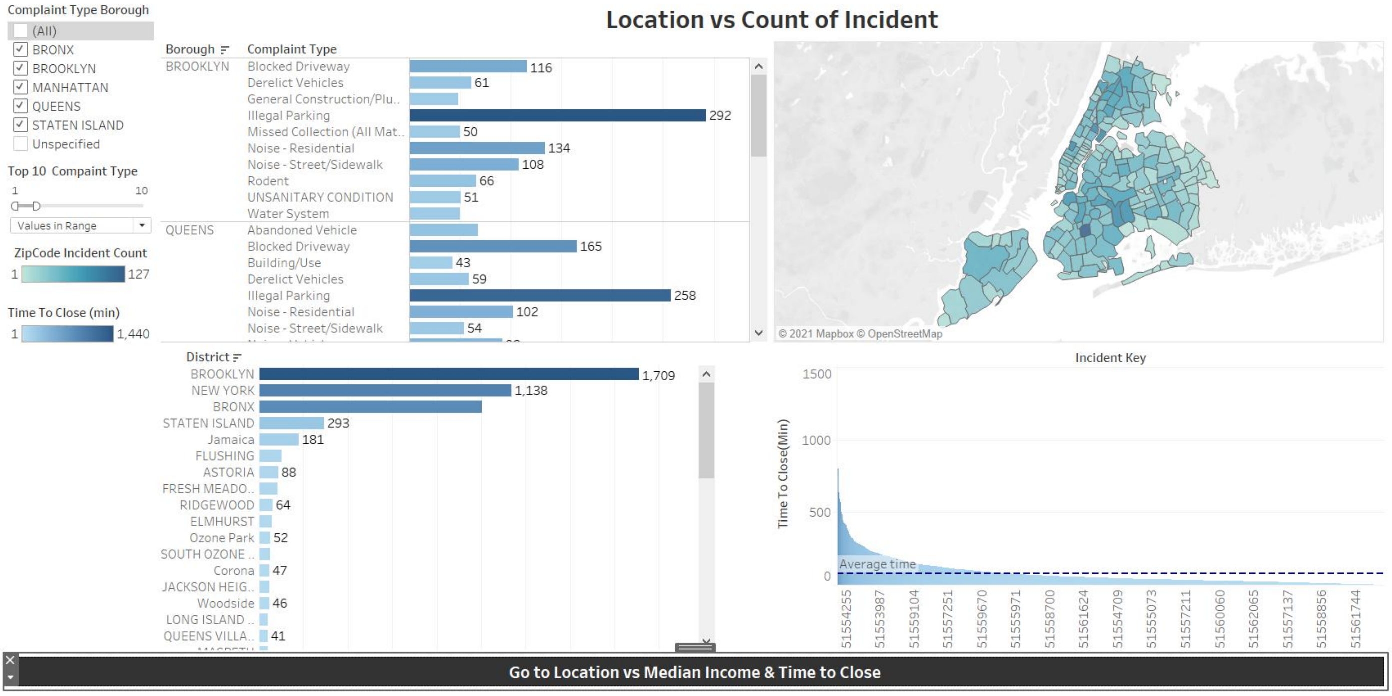


```python
##Test query to see if you can connect
query = "select * from final_dw.dim_date;"
cursor.execute(query)
result = cursor.fetchall()
print(result)

#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.')
```



```python
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



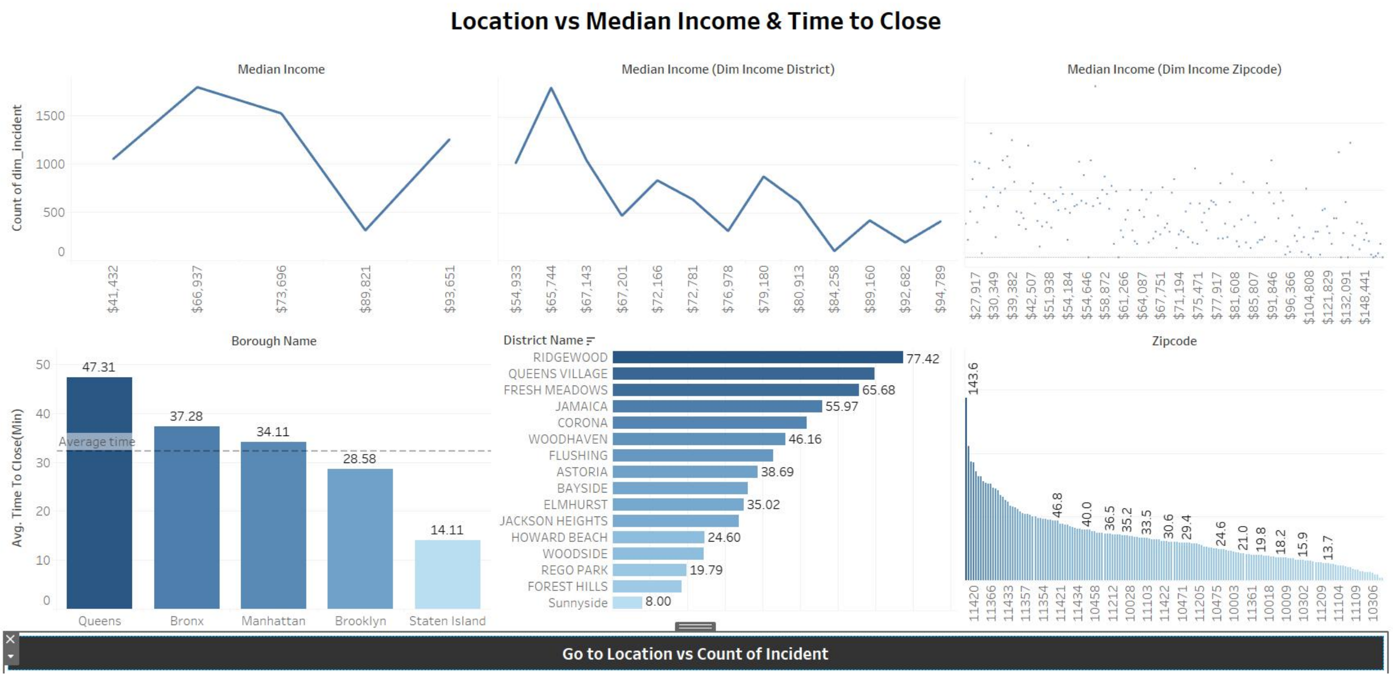
Location vs Count of Incident

Complaint Type Borough
□ (All)
☑ BRONX
☑ BROOKLYN
☑ MANHATTAN
☑ QUEENS
☑ STATEN ISLAND
□ Unspecified

Top 10 Compaint Type
1 10
Values in Range

ZipCode Incident Count
1 127

Time To Close (min)
1 1,440



Location vs Median Income & Time to Close