

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 dataset.

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

```
[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 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>

The screenshot displays the AWS Lambda console interface for the function 'service-request-lambda'. The top navigation bar includes the AWS logo, 'Services' dropdown, a search bar, and regional settings for 'N. Virginia'. The main content area is divided into several sections:

- EventBridge (CloudWatch Events)**: This section shows the function is configured as a target for an EventBridge event. It includes an 'Add trigger' button and an 'Add destination' button.
- Code source**: This section contains a message indicating that the deployment package of the Lambda function is too large to enable inline code editing. It also features an 'Upload from' dropdown menu.
- Code properties**: This section provides details about the function's code, including the package size (47.3 MB), the SHA256 hash, and the last modified date (August 18, 2021, 06:52 PM PDT).

The function's description is empty, and it was last modified 2 hours ago. The function ARN is listed as 'arn:aws:lambda:us-east-1:085184000999:function:service-request-lambda'.

3. Using CloudWatch Serice to schedule the time to run lambda function

aws

Services

Search for services, features, marketplace products, and docs

[Alt+S]

Cara

N. Virginia

Support

CloudWatch

New menu experience

Favorites

Dashboards

Alarms

In alarm

All alarms

Billing

Logs

Log groups

Logs Insights

Metrics

Events

Rules

Event Buses

Application monitoring

Insights

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

Cron expression

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

Targets

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

Lambda function

Function*

Configure version/alias

Configure input

Add target*

* Required

Cancel

Configure details

aws

Services

Search for services, features, marketplace products, and docs

[Alt+S]

Cara

N. Virginia

Support

CloudWatch

New menu experience

Favorites

In alarm

All alarms

Billing

Logs

Log groups

Logs Insights

Metrics

Events

Rules

Event Buses

Application monitoring

Insights

Settings

Getting Started

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.

Create rule

Actions

Status

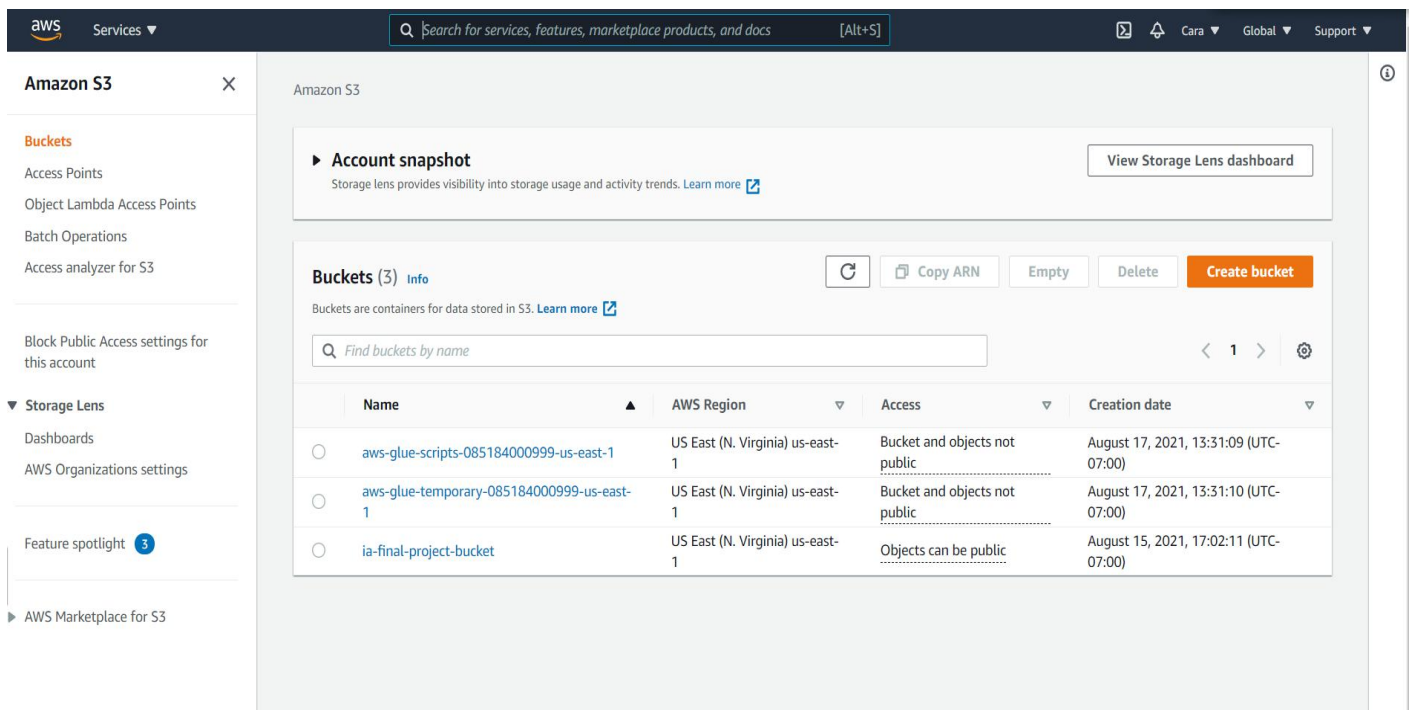
All

Name

<< < Viewing 1 to 1 of 1 Rules > >>

Status	Name	Description
<div></div>	Daily-schedule-run-lambda-service-request	Final Project Daily schedule run lambda service request function from 311

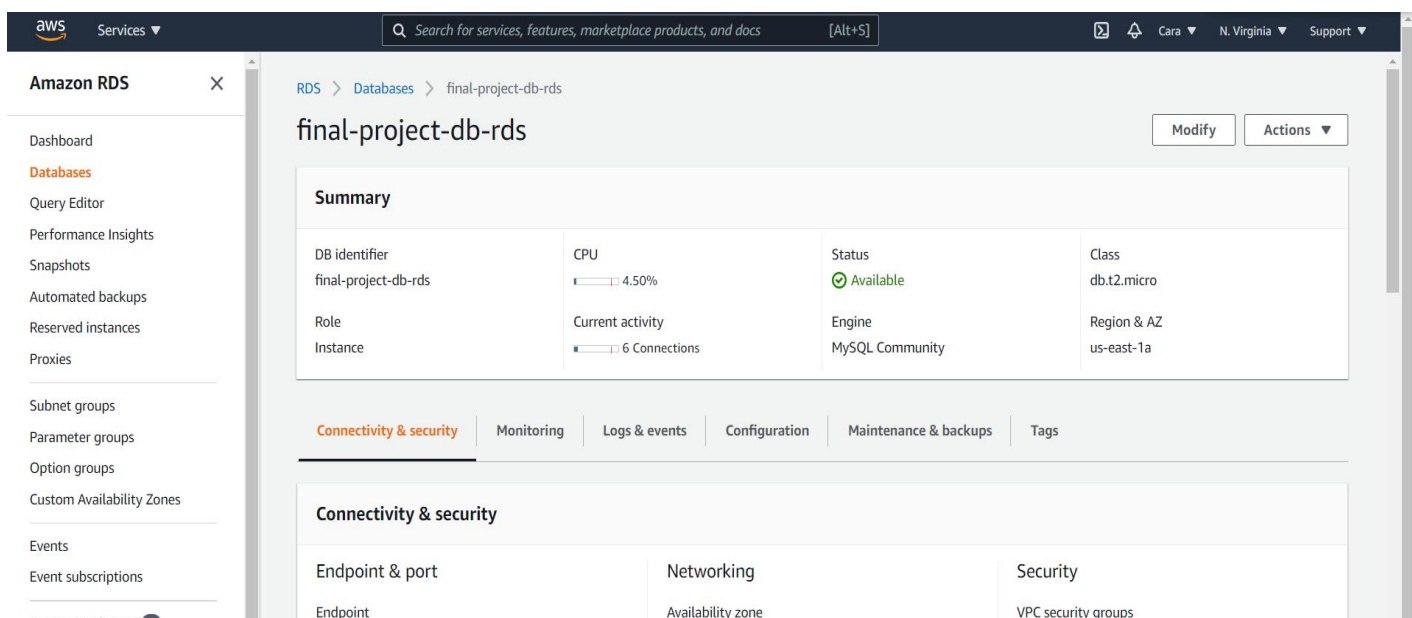
4. Using S3 Service to store the two data sources in `ia-final-project-bucket` bucket



The screenshot shows the Amazon S3 console interface. On the left, the 'Amazon S3' sidebar is visible with options like Buckets, Access Points, and Storage Lens. The main content area displays the 'Account snapshot' and a list of buckets. The bucket 'ia-final-project-bucket' is selected, showing its details in a table.

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)

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



The screenshot shows the Amazon RDS console interface. The left sidebar displays 'Amazon RDS' with various options like Dashboard, Databases, and Query Editor. The main content area shows the details for the 'final-project-db-rds' instance, including a summary of its status and configuration.

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

The 'Connectivity & security' section is expanded, showing details for Endpoint & port, Networking, and Security.

Endpoint & port	Networking	Security
Endpoint	Availability zone	VPC security groups

6. 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 Glue

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.

[Add tables](#) [Action](#) [Save view](#) Showing: 1 - 4

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

AWS Glue

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](#).

[Add crawler](#) [Run crawler](#) [Action](#) [User preferences](#) Showing: 1 - 1

<input type="checkbox"/>	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
<input type="checkbox"/>	final-project-crawler		Ready	Logs	53 secs	53 secs	0	1

AWS Glue

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](#) 

Q Filter by tags and attributes

Showing: 1 - 4 < > ↺ ?

<input type="checkbox"/> Name	Type	ETL language	Script location	Last modified	Job bookmark
<input type="checkbox"/> load-311-to-rds-mysql	Spark	python	s3://aws-glue-s...	18 August 2021 7:28 PM ...	Disable
<input type="checkbox"/> nyc_BOROUGH median_income_info	Spark	python	s3://aws-glue-s...	17 August 2021 2:48 PM ...	Disable
<input type="checkbox"/> nyc_DISTRICT median_income_info	Spark	python	s3://aws-glue-s...	17 August 2021 2:56 PM ...	Disable
<input type="checkbox"/> nyc_ZipCodes median_income_info	Spark	python	s3://aws-glue-s...	17 August 2021 2:52 PM ...	Disable

AWS Glue

Workflows (2)

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

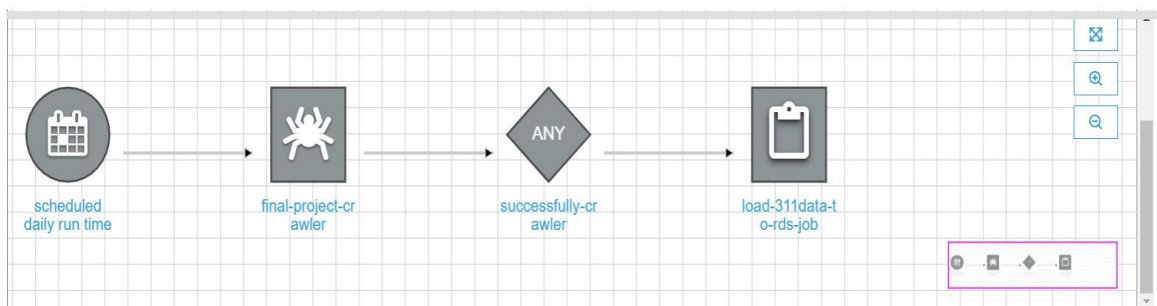
Add workflow

Actions ▾

🔍 *Filter workflows*

< 1 >

	Name	Last run	Last run status	Last modified
	Monthly_update-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



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

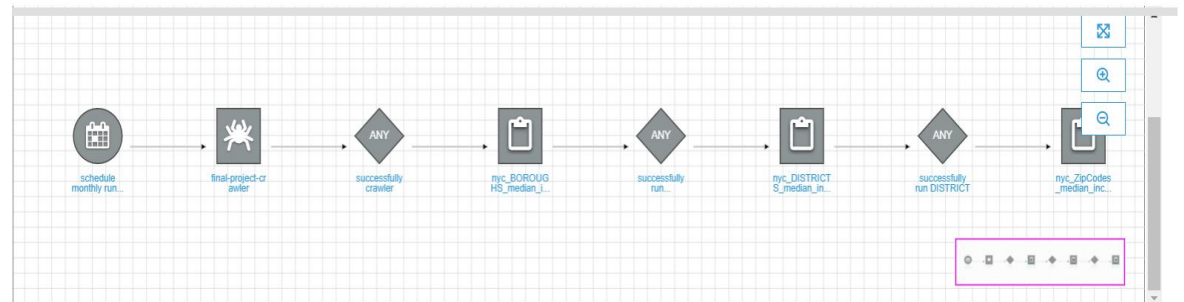
Workflows (2)

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

	Name	Last run	Last run status	Last modified
	Monthly_update-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



7. 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)
families 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)
families VARCHAR(225)
families_with_children VARCHAR(225)
families_without_children VARCHAR(225)

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

MySQL Workbench

final-project-rds x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

final_dw

- Tables
 - dim_agency
 - dim_date
 - dim_incident
 - dim_income_borough
 - dim_income_district
 - dim_income_zipcode
 - fact_table
- Views
- Stored Procedures
- Functions

Administration Schemas

Information

No object selected

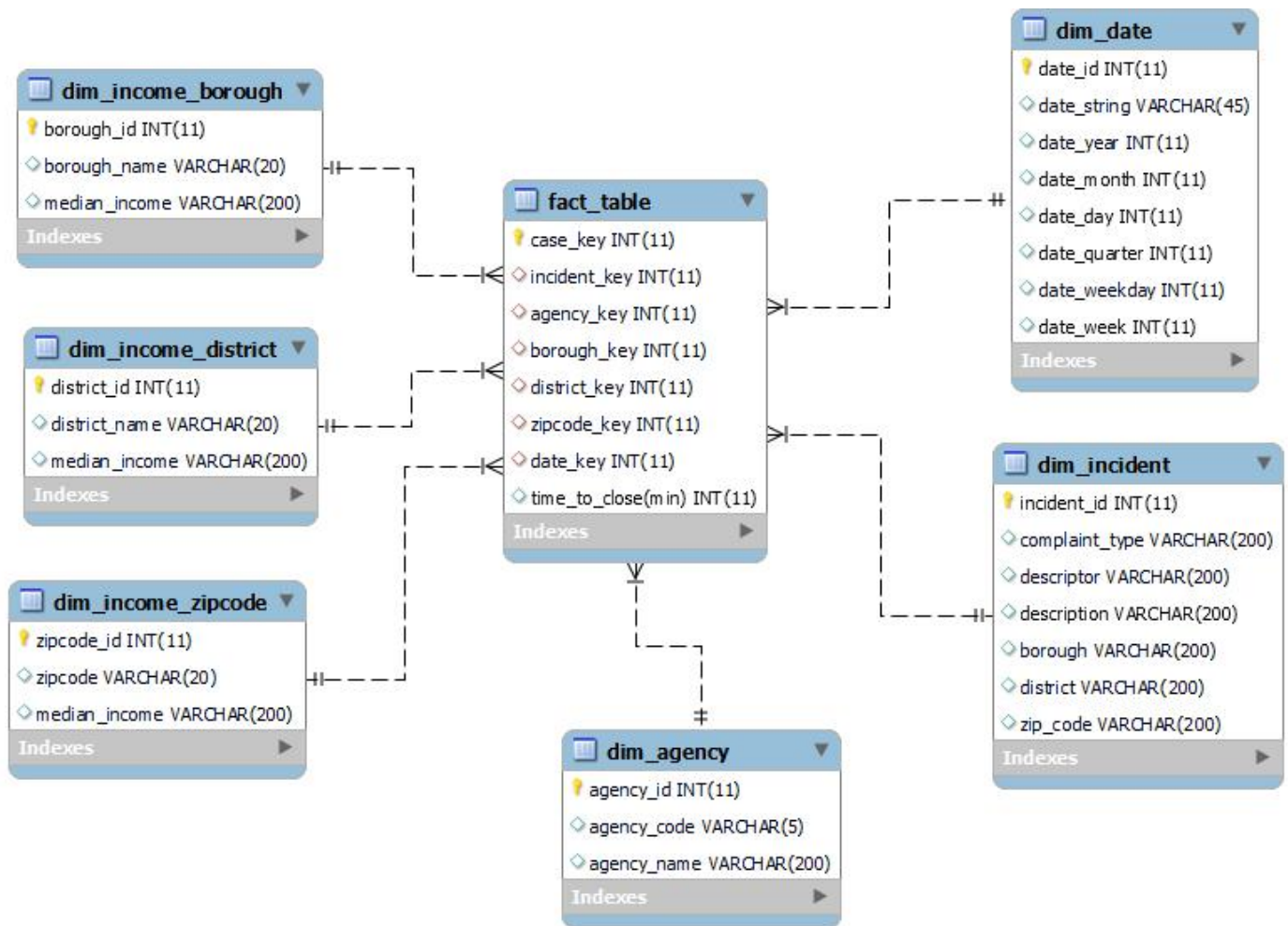
311_service_request create_dw_database update_dim_tables x update_fact_table

Limit to 200 rows

```
1 • SET @@SESSION.sql_mode='ALLOW_INVALID_DATES';
2 #Update Date Dimension
3 • SET FOREIGN_KEY_CHECKS=0;
4 • TRUNCATE TABLE final_dw.dim_date;
5 • INSERT INTO final_dw.dim_date(date_string, date_year,date_month, date_day)
6 SELECT distinct
7 Date(created_date) as 'Date String',
8 YEAR(created_date) as Year,
9 MONTH(created_date) as Month,
10 Day(created_date) as Day,
11 Quarter(created_date) as Quarter,
12 WeekDay(created_date) as WeekDay,
13 Week(created_date) as Week
14 FROM final_project_db.`311_service_request`
15 ORDER BY YEAR,MONTH, DAY;
16
17 • TRUNCATE TABLE final_dw.dim_agency;
18 • INSERT INTO final_dw.dim_agency(agency_code, agency_name)
19 SELECT DISTINCT
20 agency as agency_code,
21 agency_name as agency_name
22 FROM final_project_db.`311_service_request`;
23
24 • TRUNCATE TABLE final_dw.dim_incident;
25 • INSERT INTO final_dw.dim_incident(incident_id, complaint_type, descriptor
```

9. Database Reverse Engineer ER diagram for star schema Data Warehouse database

`final_dw`



10. 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.

```
13  ##Test query to see if you can connect
14  query = "select * from final_dw.dim_date;"
15  cursor.execute(query)
16  result = cursor.fetchall()
17  print(result)
18
19  #Call Dimension Tables Stored Procedure
20  with open("./update_dim_tables.sql", encoding="utf-8") as f:
21      commands = f.read().split(';')
22
23  for command in commands:
24      cursor.execute(command)
25      print(command)
26
27  print('Dimension tables updated.')
28
29
30  #Call Fact Tables Stored Procedure
31  with open("./update_fact_table.sql", encoding="utf-8") as f:
32      commands = f.read().split(';')
33
34  for command in commands:
35      cursor.execute(command)
36      print(command)
37
38  print('Fact tables updated.')
39
40
41  connection.commit()
42  connection.close()
43  print('Disconnected from database.')
```

```
10  sent_from = gmail_user
11  to = ['xiaolancara@gmail.com', 'bhcooper@mail.yu.edu']
12  subject = 'Batch Process Completed on: ' + now.strftime("%m/%d/%Y - %H:%M:%S")
13  body = 'Your Final Project Batch Job Processed with 0 Errors'
14
15  email_text = """\
16  From: %s
17  To: %s
18  Subject: %s
19
20  %s
21  """ % (sent_from, to, subject, body)
22
23  try:
24      server = smtplib.SMTP('smtp.gmail.com', 587)
25      server.ehlo()
26      server.starttls()
27      server.login(gmail_user, gmail_password)
28      server.sendmail(sent_from, to, email_text)
29      server.close()
30      print_('Email successfully sent')
31  except:
32      print_('Error: your email did not send')
```

11. Using Tableau to Analyze the data warehouse