Cloud Computing Assignment 2

Victoria Road Crash Statstics

MAY 24

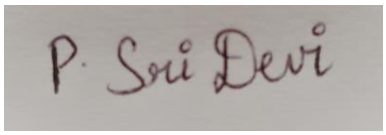
COSC2640 – Group 28
Authored by:
- Sridevi Pamarthi (s3778317)
- Divya Ulaganathan (s3759465)

RMIT UNIVERSITY

Contents

Signed Contribution Agreement

Group name: Group 28

Student Name: Divya Ulaganathan	Student Name: Sridevi Pamarthi
Student ID: s3759465	Student ID: s3778317
Contribution Percentage: 50	Contribution Percentage: 50
By signing below, I certify all information is true and correct to the best of my knowledge. Signature: Divya Ulaganathan Date: 24/05/2020 X Divya Ulaganathan Miss	*By signing below, I certify all information is true and correct to the best of my knowledge.* Signature: Sridevi Pamarthi Date: 24/05/2020 

Links

VicRoads dataset URL used for this assignment is as follows:
https://vicroadsopendatastorehouse.vicroads.vic.gov.au/opendata/Road_Safety/ACCIDENT.zip

Application URL:
http://sridevidivya.ap-southeast-2.elasticbeanstalk.com/

Quicksight dashboard URL:
https://ap-southeast-2.quicksight.aws.amazon.com/sn/dashboards/480c4e89-e3c1-4bc0-aa91-5c7091b5fcc0

Summary

The aim of this project is to provide statistics and insights of a big dataset in a secured manner.

The application utilises all possible AWS cloud infrastructure and services like compute, storage, database, Networking & Content Delivery, Analytics and Security, Identity, & Compliance. By using this application, the admin user can share their analysis to specific user group by authenticating their identity.

Introduction

VicRoads provides crash statistics data to organizations, researchers, and the general public to help with education, research and the development of road safety programs and initiatives.

The main objective of this project is to provide statistics and insights related to Victoria state crashes to the public which is to educate and prevent any possible accidents in the future. Given the population of state is over 6 million, we anticipate that this application could be accessed by many people across the state over a period of time and so this application is designed accordingly to be able to scale up and down based on the demand.

This application processes the Victoria state crash history data from the year 2006 to date (2020). It extracts or pulls the information from the VicRoads website. The dataset constitutes of 11 csv files with records ranging from 1,92,886 to 3,50,000. It provides statistics and insights related to Victoria state crashes to the public which is to educate and prevent any possible accidents in the future.

AWS cloud computing services were designed and implemented to analyze such large datasets quickly and also to support the growth of data volumes and public access demand.

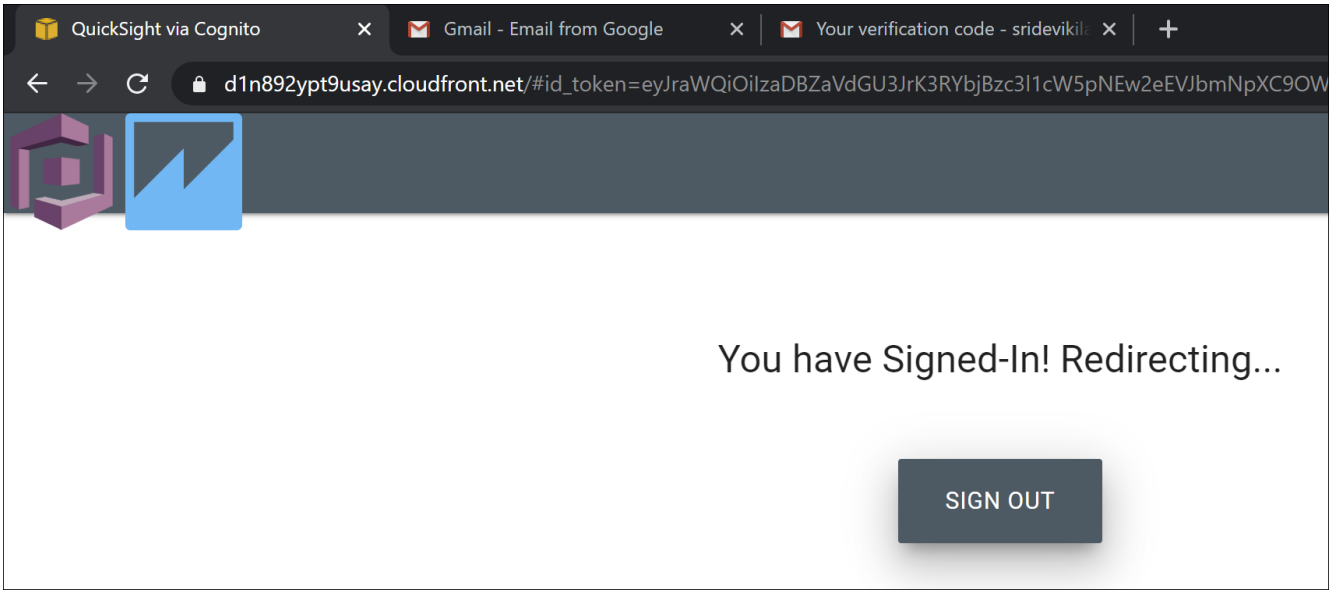
As per design, the CloudFront and Cognito services are used to handle the public access to this application. The user should sign up and authenticate the identity first, only then s/he should be able to access the statistics and insights published in QuickSight application related to road accidents of Victoria state. The website is powered by the Elastic Beanstalk service which automatically supports the capacity provisioning and health monitoring of the application.

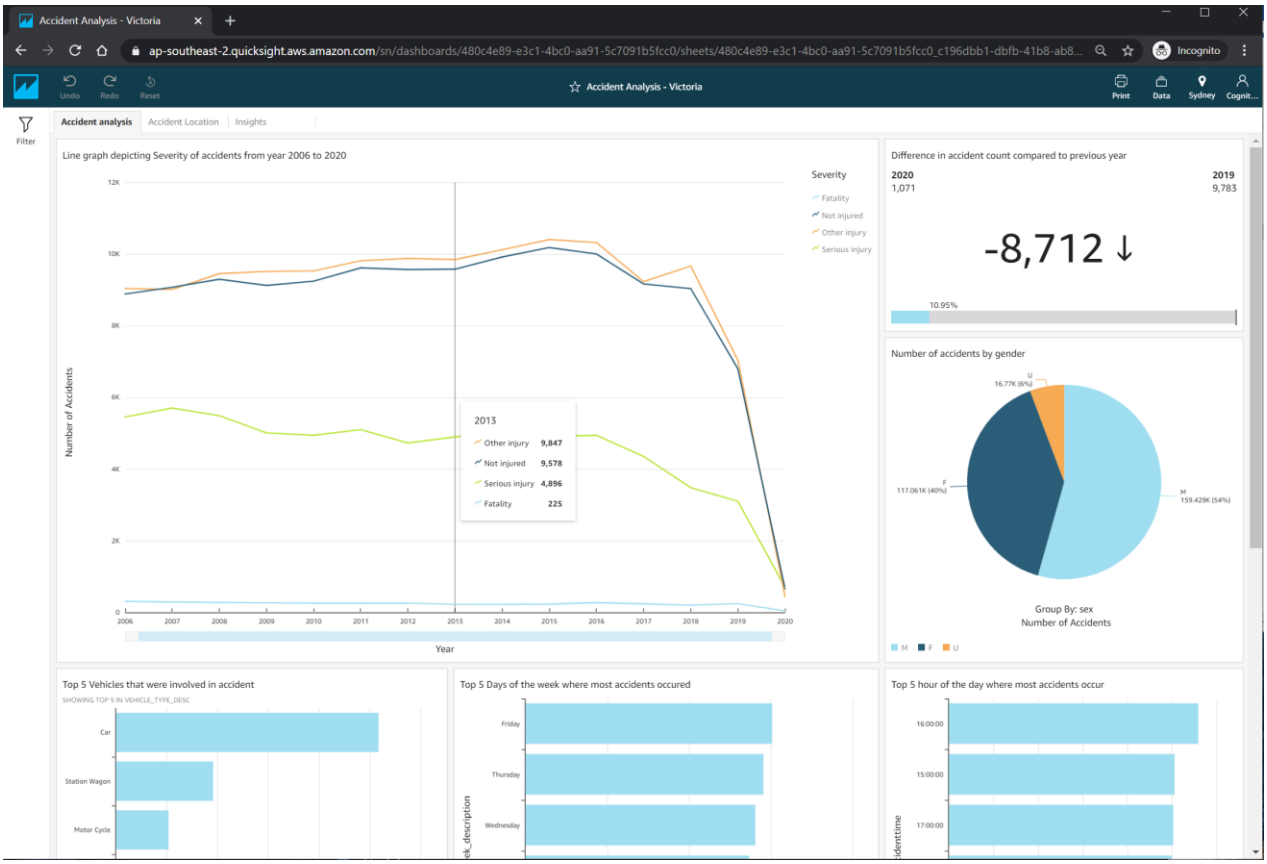
While the Elastic Beanstalk handles the demand from the website or front end automatically, the backend part of the application must also be compatible and equally powerful enough to make this project success. As a result, the highly scalable (massively parallel architecture) Redshift database solution is used to store and provide the data to QuickSight application which is a powerful cloud business intelligence service for producing the right insights and forecast on time.

Overall, with the objective of this project and its technology, more people will be able to access this application from their smart devices like laptop, desktop, mobile and tablet to gain the knowledge of the Victoria crashes and the locations so that they could act carefully to prevent accidents in those regions.

Industries that deal with highly sensitive information and analyses big data on daily manner like banking and finance, health sectors etc., can utilize this application architecture to access and shared their analysis in a secured infrastructure.

Upon successful validation, the user has now redirected to QuickSight visuals as follows:


QuickSight via Cognito
Gmail - Email from Google
Your verification code - sridevikil
d1n892ypt9usay.cloudfront.net/#id_token=eyJraWQiOiIzaDBZaVdGU3JrK3RYbjBzc3l1cW5pNEw2eEVJbmNpXC9OW
You have Signed-In! Redirecting...
SIGN OUT


Accident Analysis - Victoria
ap-southeast-2.quicksight.aws.amazon.com/sn/dashboards/480c4e89-e3c1-4bc0-aa91-5c7091b5fcc0/sheets/480c4e89-e3c1-4bc0-aa91-5c7091b5fcc0_c196dbb1-dbfb-41b8-ab8...
Incognito
Accident Analysis - Victoria
Print
Data
Sydney
Filter
Accident analysis
Accident Location
Insights
Line graph depicting Severity of accidents from year 2006 to 2020
Severity
Fatality
Not injured
Other injury
Serious injury
Number of Accidents
Year
2013
Other injury 9,847
Not injured 9,578
Serious injury 4,896
Fatality 225
Difference in accident count compared to previous year
2020
1,071
2019
9,783
-8,712 ↓
10.95%
Number of accidents by gender
16.77K (6%)
117.061K (40%)
159.429K (54%)
Group By: sex
Number of Accidents
Top 5 Vehicles that were involved in accident
Car
Station Wagon
Motor Cycle
Top 5 Days of the week where most accidents occured
Friday
Thursday
Wednesday
Top 5 hour of the day where most accidents occur
16:00:00
15:00:00
17:00:00

Related work

The below work published in GitHub is similar to this project.

https://github.com/ajdeziel/crash-heat

Difficulties faced

Given the volume of the services used in this project are huge, the challenges we faced mainly were related to the connectivity such as VPC /network settings. To overcome the connectivity challenge between QuickSight and Redshift, the QuickSight has been upgraded (for ex: QuickSight) to the enterprise edition.

To whitelist the web application URL in QuickSight, the Elasitic BeanStalk application should be compatible with HTTPS protocol. To achieve it, we require the SSL certificate which can only be obtained if there is registered domain.

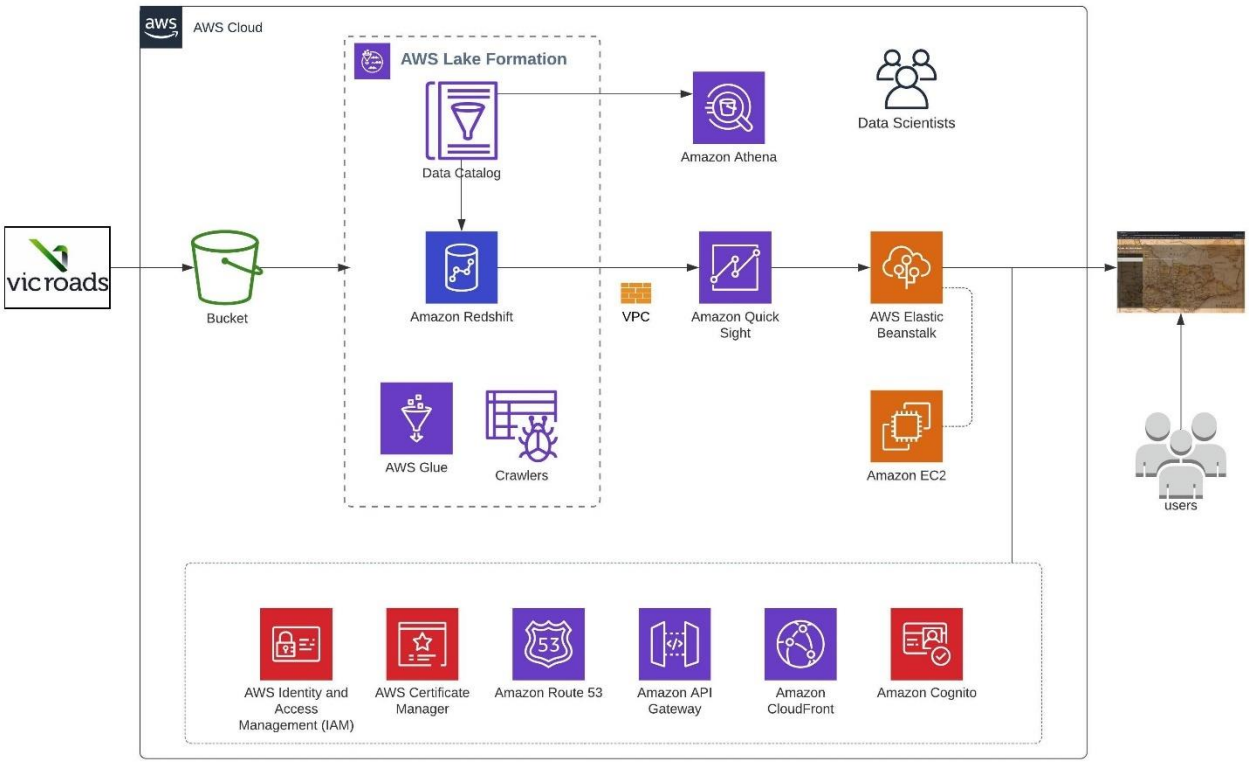
We initially thought of embedding the QuickSight dashboards directly in the website, however, there is not enough information available in the internet to get the Embed URL of the dashboard using a PHP application.

From the knowledge learnt from Google Cloud Platform before, we understood that the composer to be installed in the PHP application directory to access the SDK of the cloud provider within the code. However, the composer installation is not required for Elastic BeanStalk and instead the composer.json to be uploaded with the application code.

Software Design/ Architecture

The below high-level architecture diagram depicts the various cloud components used to achieve this project and the purpose of every component has explained below.

Victoria Road Crash Analytics Architecture



Purpose of AWS cloud components:

1. **S3 Bucket**
    a. to store the data pulled from the VicRoads website
    b. to support data lake, to store the Glue job logs
    c. to host the static website required for CloudFront distribution

2. **Lake Formation**
    a. to register the data lake location
    b. to create the database(s) on S3 storage

3. **Glue Crawler**
    a. to scan the S3 bucket (the VicRoads dataset) and create the corresponding table schema in a Database (AWS Lake) to populate the Data Catalog

4. **Glue Data Catalog**
    a. to verify the table schema, location, and other metadata
    b. to modify the table schema (column names, data types)
    c. to create the connections to other databases of type JDBC/ RDS/ **Redshift (in this case)**/Kafka etc. Glue ETL job makes use this connection to load the data into Redshift

5. **Glue ETL Jobs**
    a. to automate data retrieval from VicRoads website to the S3 bucket
    b. to load the raw data from S3 bucket to the tables created by the Crawler, in AWS Lake/Data Catalog
    c. to load the data from Data Catalog to AWS Redshift database

6. **Redshift**
    a. MPP data warehouse (massively parallel processing /architecture) solution to store the data in the transformed form as required for analytics and reporting.
    b. to support the ever-growing data requirements and demand, as this database solution can scale out quickly to meet the needs. Also, the number of nodes can scale up and down easily.

7. **Athena**
    a. to explore the datasets in AWS Lake/ Data Catalog before bringing into the AWS Redshift database solution.

8. **Quick Sight**
    a. to build the interactive dashboards or reports using the data from AWS redshift and other data sources
    b. to publish the dashboards to the users who can then access from computer/ mobile/ tablet
    c. to publish the dashboards to embed in the websites or web applications
    d. to whitelist the external website access to Quick Sight dashboards

9. **Elastic Beanstalk**
    a. to build the highly scalable web application to meet the demand. The web application embeds the dashboards from the Quick Sight.

10. **EC2**
    a. to support the Elastic Beanstalk architecture and application hosting

11. **Route 53**
    a. to register a domain which is required to use HTTPS protocol for the web application deployed in the Elastic Beanstalk.

12. **Certification Manager**
    a. to obtain the SSL certificate using the domain, and to enable the HTTPS protocol for the web application.

13. **Cognito**
    a. to authenticate/integrate user access to different AWS services. This is used for authenticating the users through CloudFront when accessing the Quick sight dashboards.

14. **CloudFront**
    a. to serve a static website hosted on S3 bucket, this has public access. When users access the CloudFront URL, they get authenticated via Cognito and then redirects to Quick Sight (in our case) or any other services.

15. **VPC**
    a. to manage the subnets, security groups, end points as required for the current solution.

16. **IAM**
    a. to create the groups, users, roles, policies as required for the current solution.

Description of your dataset/data structure/APIs/sensors you used for your project (if any):
Coding Language:

1. **Spark** for Glue ETL jobs to load the data into Data lake and Redshift
2. **Python** for Glue ETL job to retrie the data from Vicroads website
3. **Javascript** for web application

4. **PHP** for web application
5. **Html** for web application
6. **SQL** for data exploration and creating the views with transformations

Cloud based Storage solution

- Amazon S3

Cloud based reporting solution

- Amazon Quick Sight

Cloud based data lake solution

- Amazon Data Lake

Cloud based hosting solution

- Elastic Beanstalk

APIs

- Quick Sight API
- Redshift API

Cloud based data warehousing solution

- Amazon Redshift
- Data structures related to views and tables are available in the Appendix section

Implementation

Below are the step-by-step guidelines to reproduce this project and make it live.

1. Setup S3 buckets:

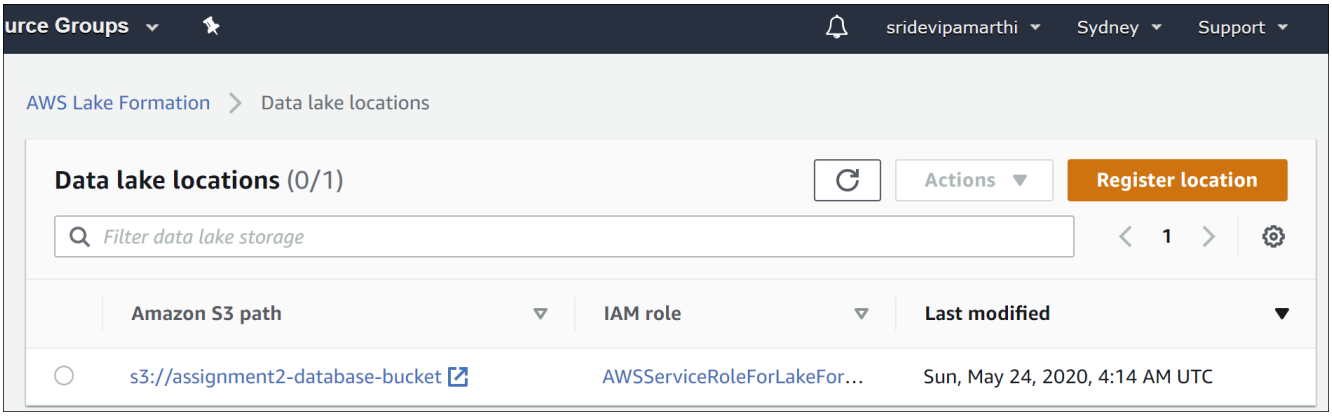
Bucket Name	Purpose
assignment2-data-bucket	for the initial accident dataset
assignment2-database-bucket	for the Data Lake storage
assignment2-jobs-bucket	for the Glue job executions
assignment2-misc-bucket	for miscellaneous
assignment2-query-bucket	for executing queries from Athena

Create the following IAM roles

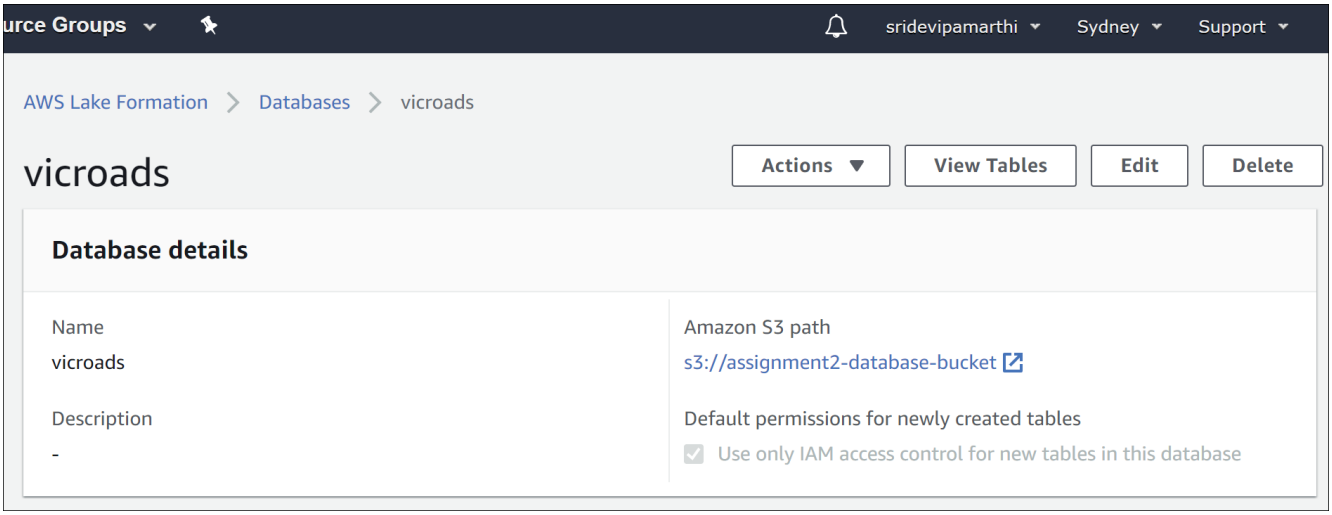
Role Name	Role Description
aws-assignment2-glue	To provide the access to s3 bucket, Redshift and glue services
aws-assignment2-s3	To provide the access to s3 bucket

2. Setup AWS Lake formation

a) Register Data Lake location



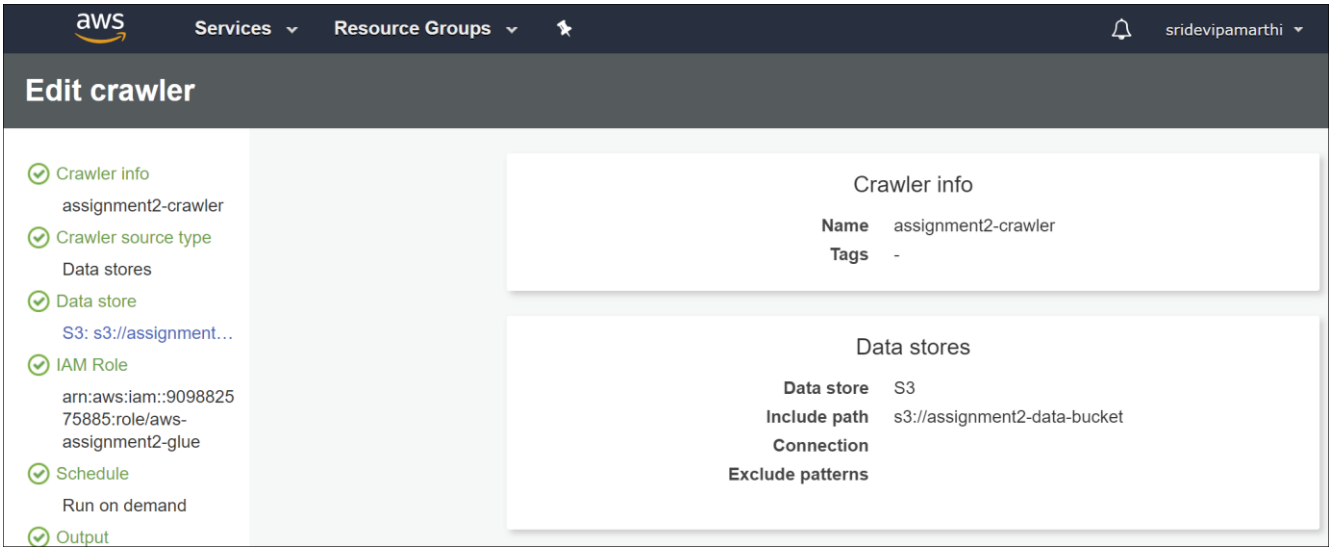
b) Create the Database 'vicroads' in AWS Lake

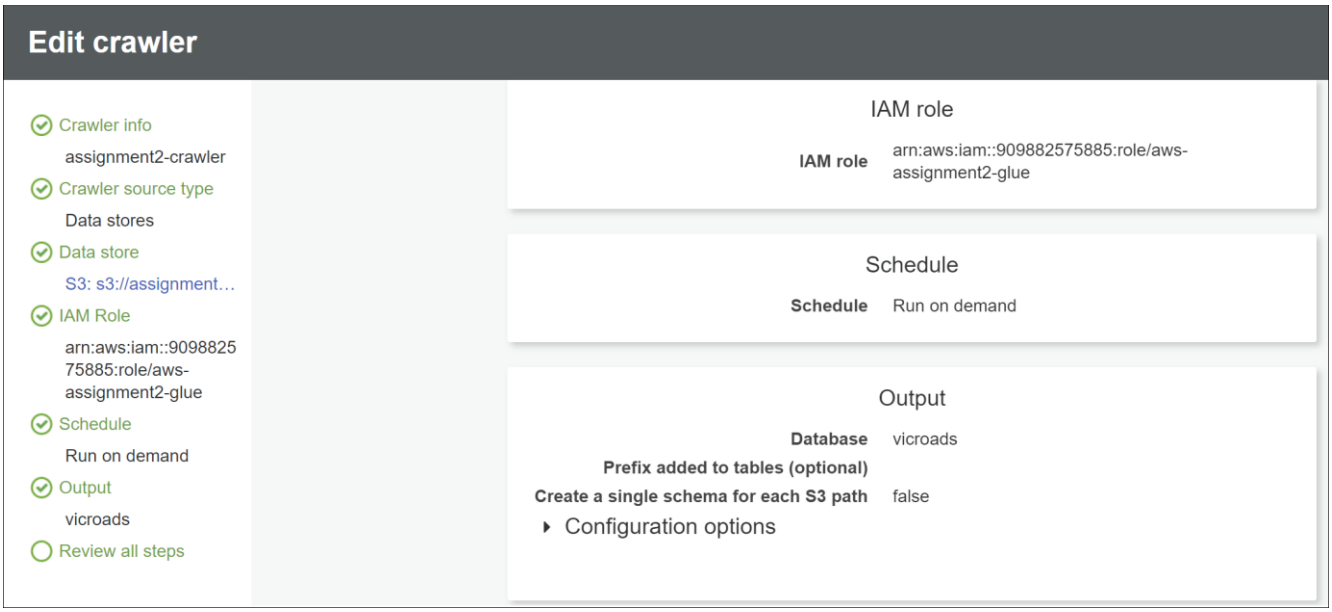


3. Setup AWS Glue

a) Create Crawler

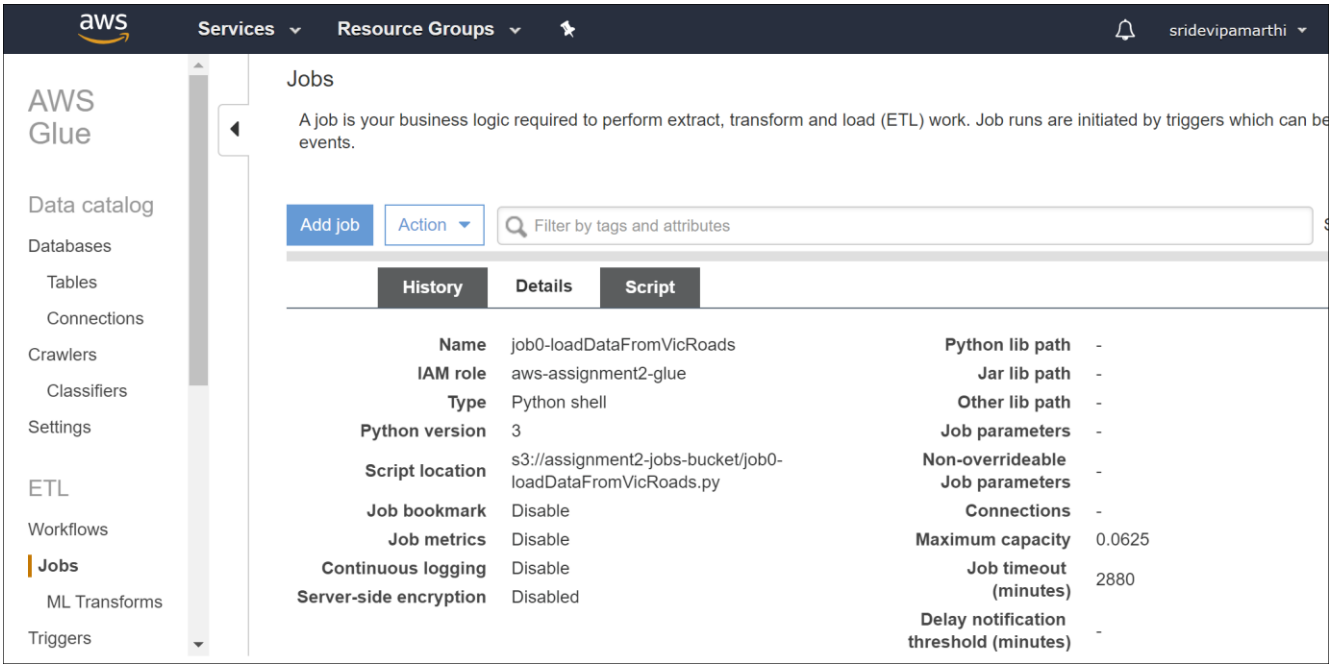
- Create the Crawler 'assignment2-crawler' in AWS Glue





b) Create Glue ETL jobs

i. Create Glue ETL job to retrieve data from VicRoads



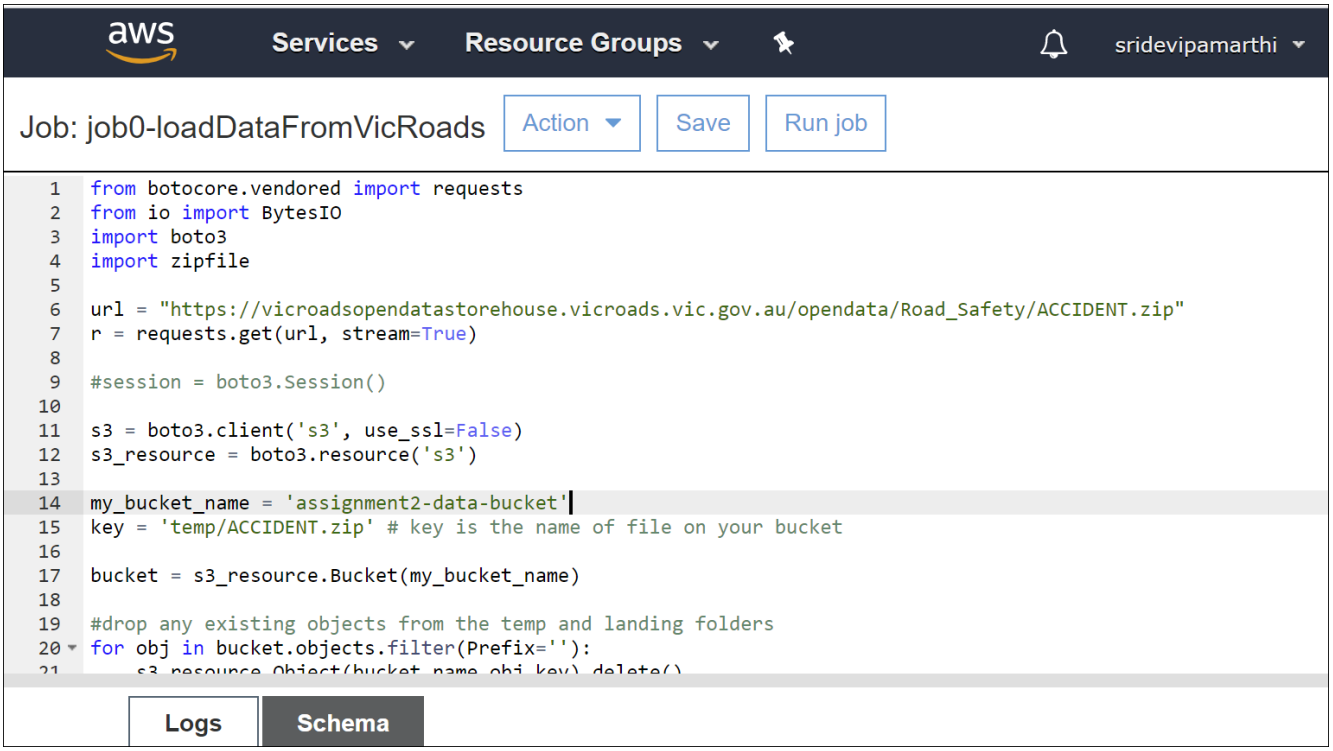
Python script: Go to Action and select edit script to update the following script in the Script tab of the job. This script does the following steps:

1. It retrieves the zip file from the Vicroads and place in the temp folder of the bucket (the folder gets created automatically)
2. it unzips the file and place the datasets into the root location of the bucket

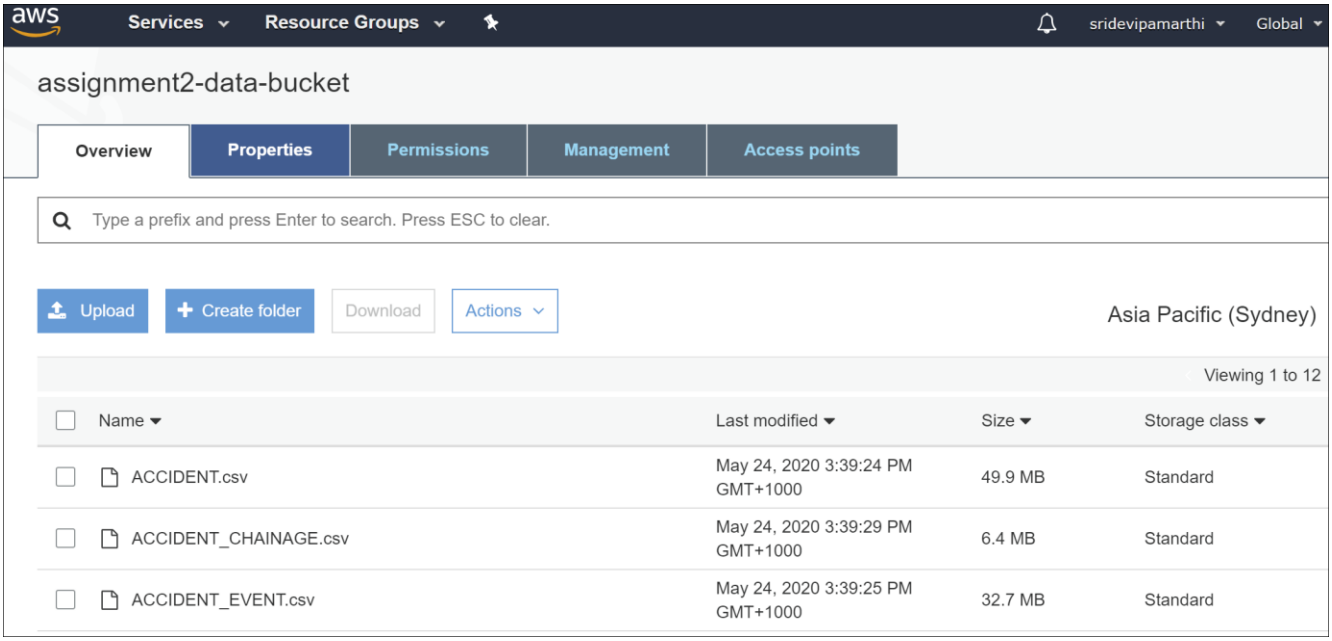
```
from botocore.vendored import requests
from io import BytesIO
import boto3
import zipfile

url = "https://vicroadsopendatastorehouse.vicroads.vic.gov.au/opendata/Road_Safety/ACCIDENT.zip"
r = requests.get(url, stream=True)

#session = boto3.Session()

s3 = boto3.client('s3', use_ssl=False)
s3_resource = boto3.resource('s3')

my_bucket_name = 'assignment2-data-bucket'
key = 'temp/ACCIDENT.zip' # key is the name of file on your bucket

bucket = s3_resource.Bucket(my_bucket_name)

#drop any existing objects from the temp and landing folders
for obj in bucket.objects.filter(Prefix=''):
    s3_resource.Object(bucket.name,obj.key).delete()


#upload zip file from the URL to S3
bucket.upload_fileobj(r.raw, key)


prefix      = "temp/"
zipped_keys =  s3.list_objects_v2(Bucket=my_bucket_name, Prefix=prefix, Delimiter = "/")
```

```
file_list = []
for key in zipped_keys['Contents']:
    file_list.append(key['Key'])

#Now create zip object one by one, this below is for 1st file in file_list
zip_obj = s3_resource.Object(bucket_name=my_bucket_name, key=file_list[0])
print (zip_obj)
buffer = BytesIO(zip_obj.get()["Body"].read())

z = zipfile.ZipFile(buffer)
for filename in z.namelist():
    file_info = z.getinfo(filename)
    s3_resource.meta.client.upload_fileobj(
        z.open(filename),
        Bucket=my_bucket_name,
        Key= f'{filename}')
```

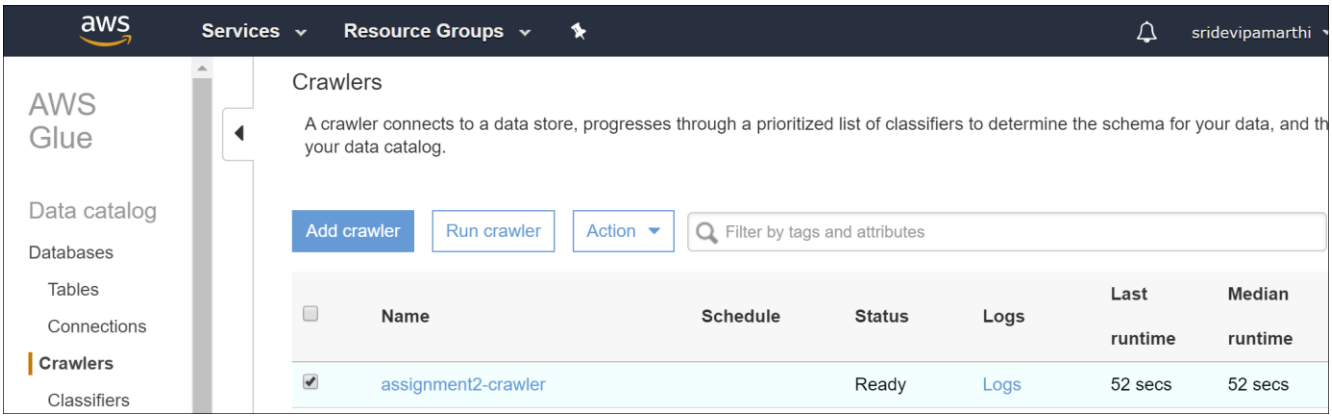
ii. Run the above job to get the data into S3 bucket.



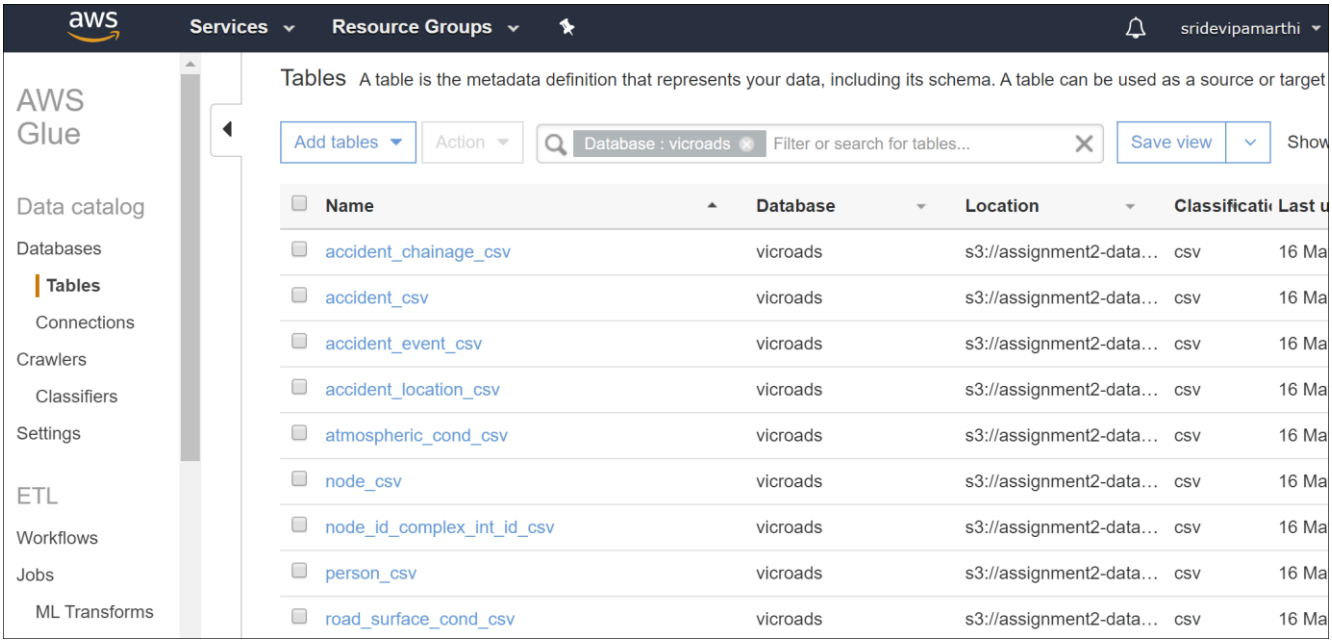
iii. The data gets loaded into the S3 bucket as follows:



iv. Run the crawler crated in the above step 3a. This will create the table structures in Data Lake database 'vicroads'



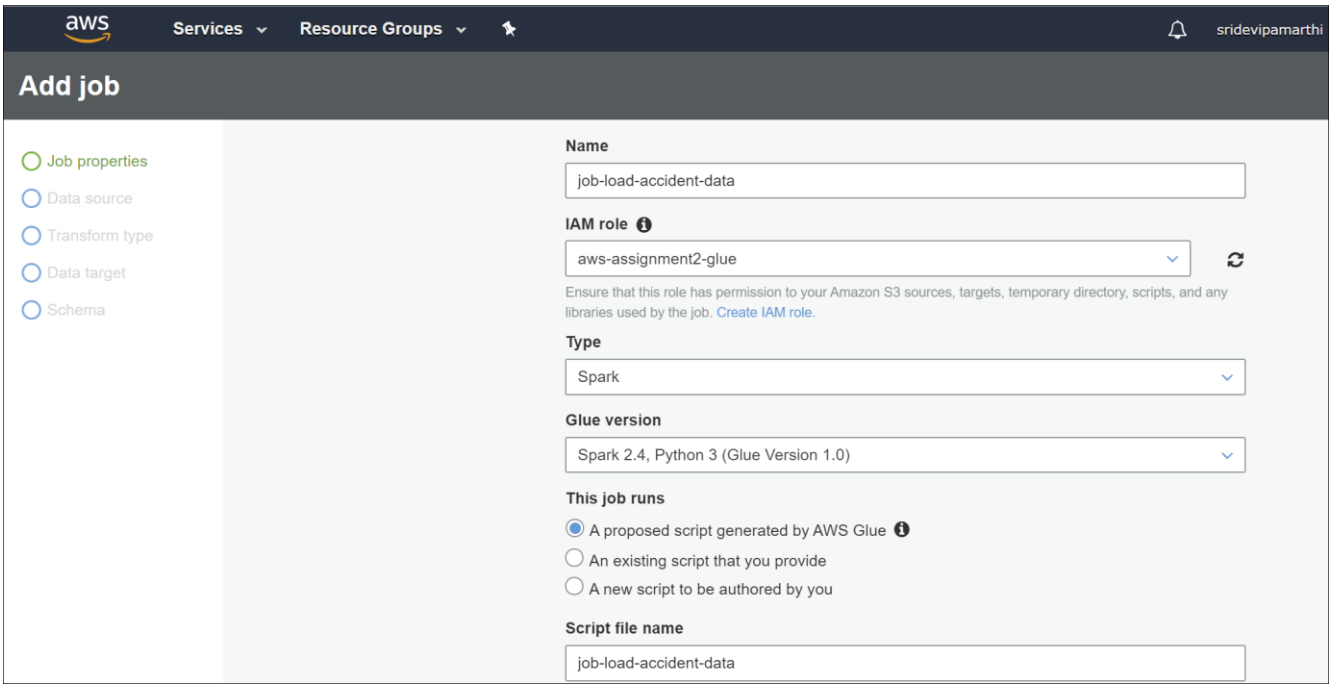
v. The table structures get created now and we can edit the table definition (renaming columns, changing data types) if require.

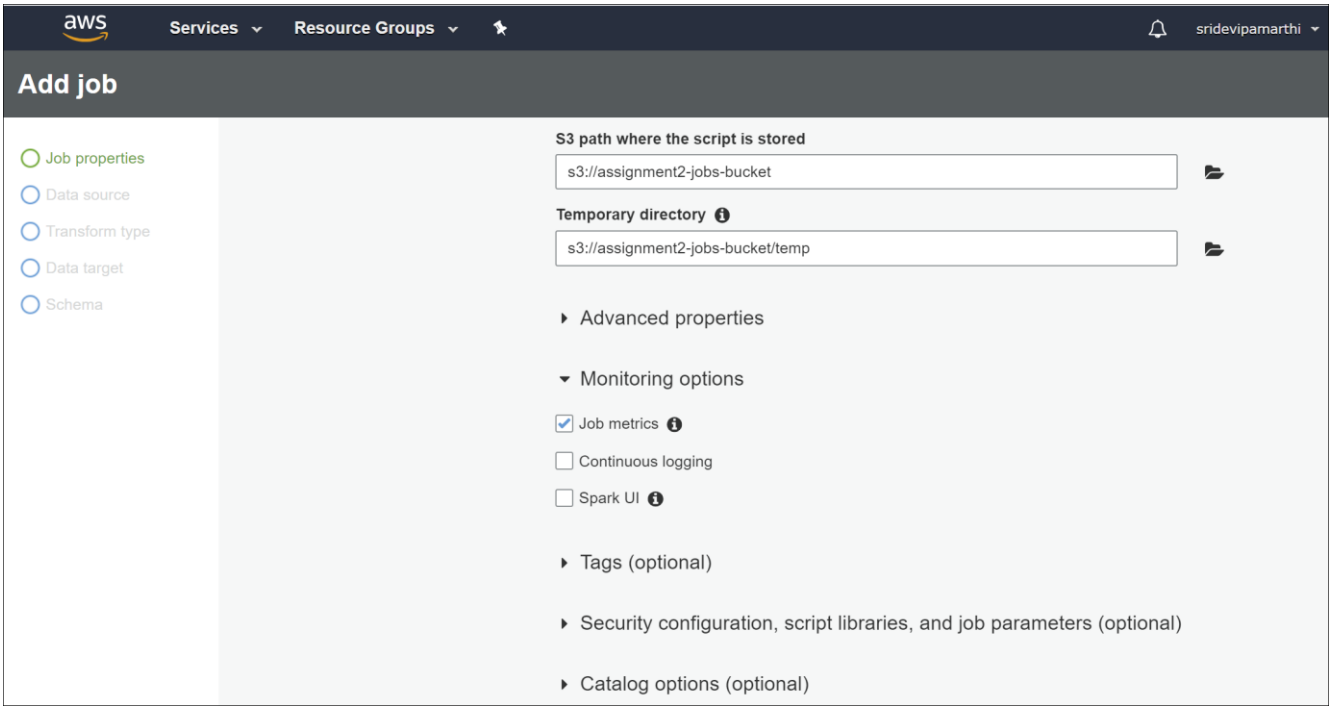


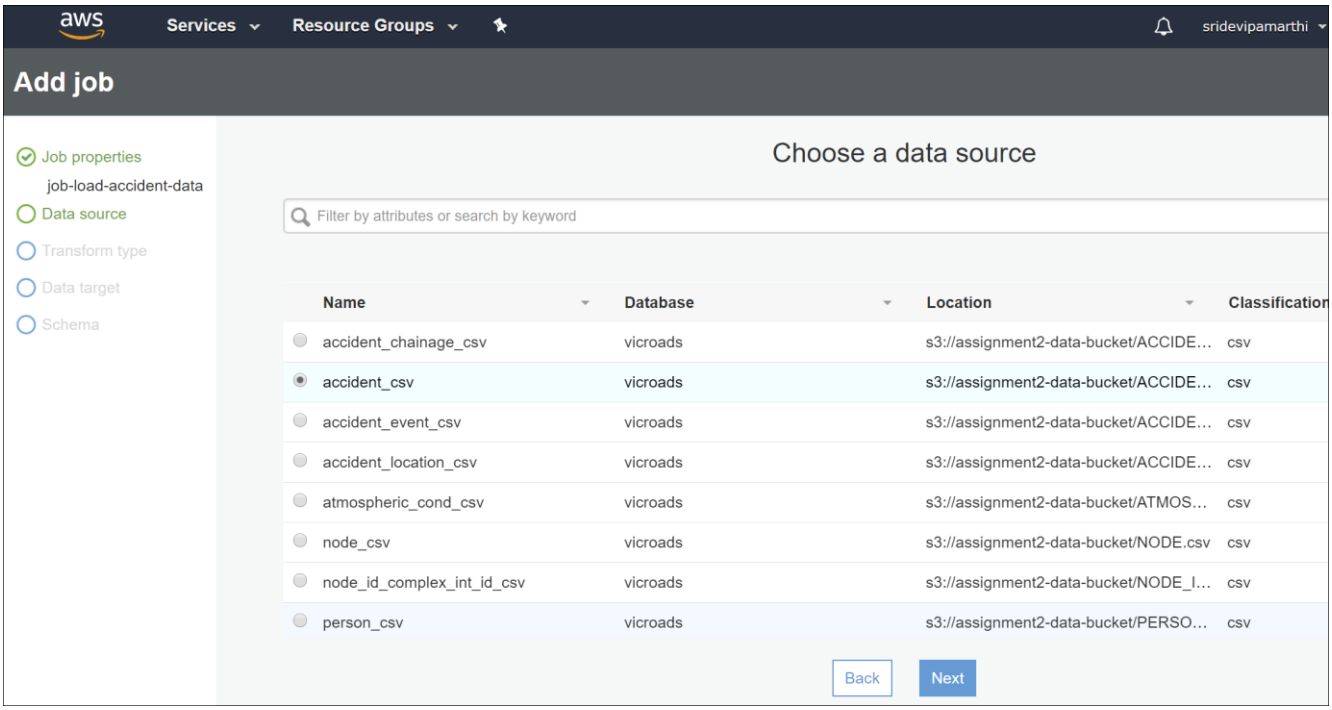
vi. Create Glue jobs to load the data from corresponding file in S3 bucket to the above freshly created tables of the vicroads database (in Data Lake).

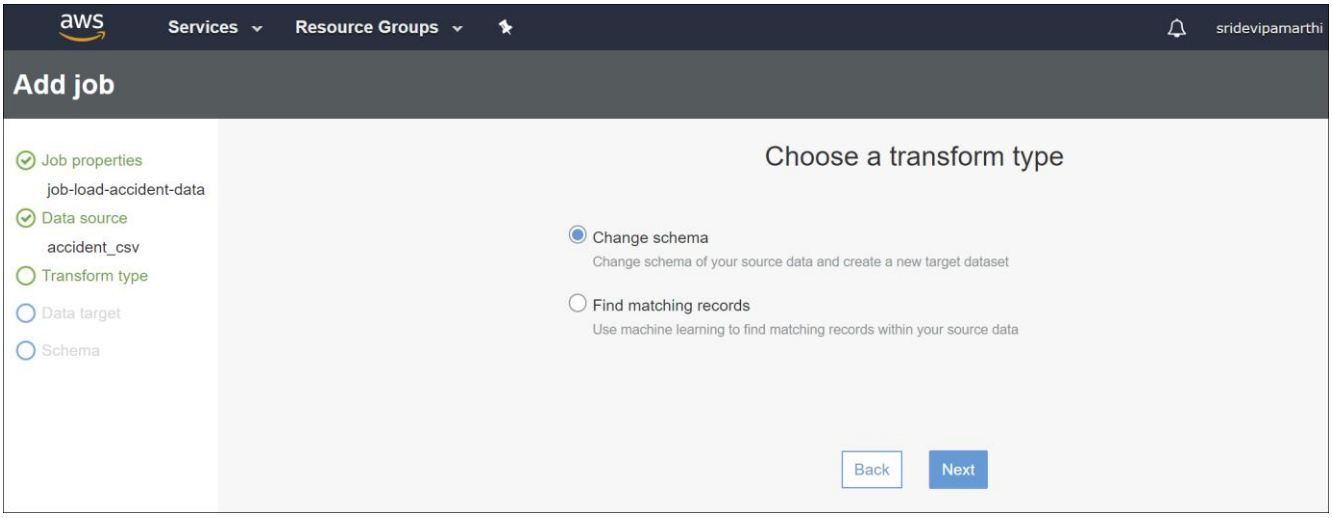
We created a specific job for every table, however, all the underlying job scripts can be combined into a single job, if require.

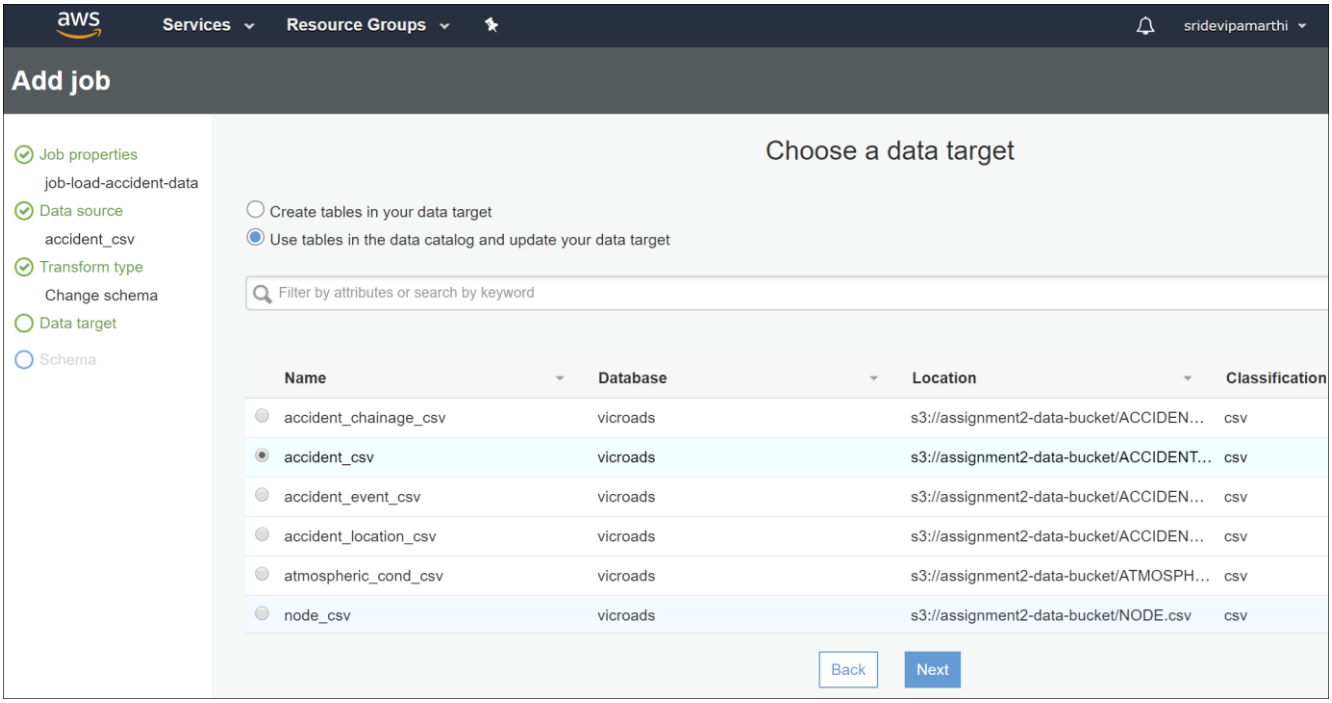
- **Sample steps to create a Glue job for accident dataset. and these steps to be repeated for all the tables.**

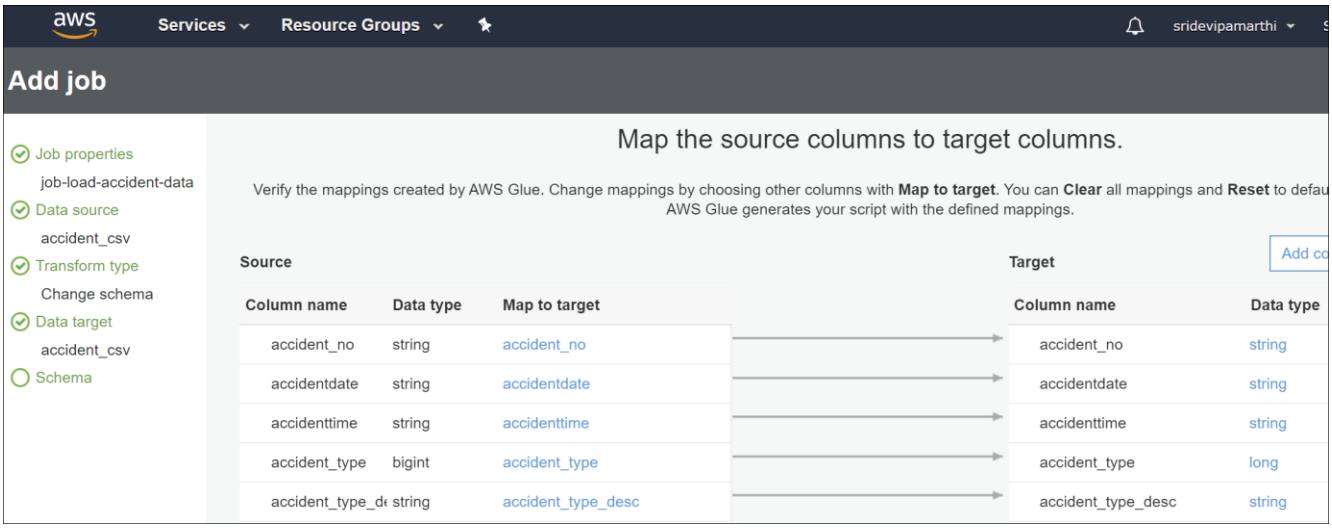


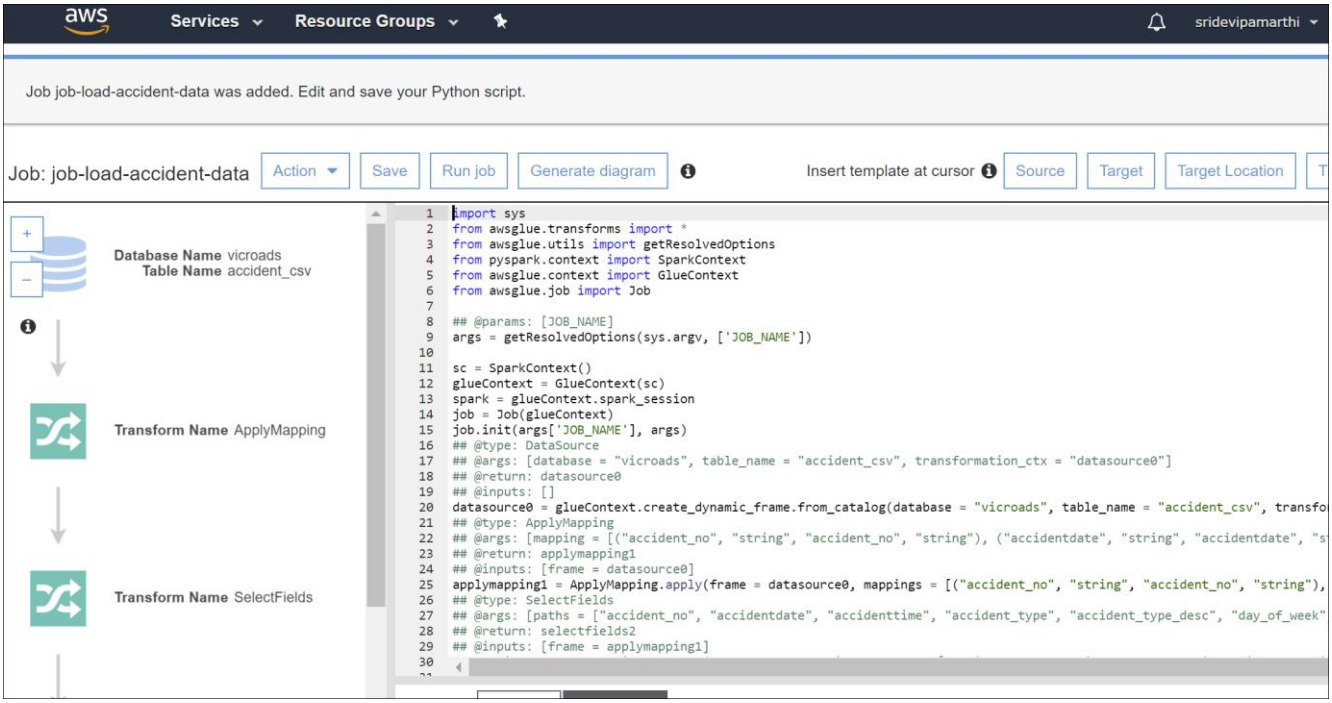

aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
Data source
Transform type
Data target
Schema
S3 path where the script is stored
s3://assignment2-jobs-bucket
Temporary directory
s3://assignment2-jobs-bucket/temp
Advanced properties
Monitoring options
Job metrics
Continuous logging
Spark UI
Tags (optional)
Security configuration, script libraries, and job parameters (optional)
Catalog options (optional)


aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
job-load-accident-data
Data source
Transform type
Data target
Schema
Choose a data source
Filter by attributes or search by keyword
Name
Database
Location
Classification
accident_chainage_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
accident_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
accident_event_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
accident_location_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
atmospheric_cond_csv
vicroads
s3://assignment2-data-bucket/ATMOS...
csv
node_csv
vicroads
s3://assignment2-data-bucket/NODE.csv
csv
node_id_complex_int_id_csv
vicroads
s3://assignment2-data-bucket/NODE_I...
csv
person_csv
vicroads
s3://assignment2-data-bucket/PERSO...
csv
Back
Next


aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
job-load-accident-data
Data source
accident_csv
Transform type
Data target
Schema
Choose a transform type
Change schema
Change schema of your source data and create a new target dataset
Find matching records
Use machine learning to find matching records within your source data
Back
Next


aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
job-load-accident-data
Data source
accident_csv
Transform type
Change schema
Data target
Schema
Choose a data target
Create tables in your data target
Use tables in the data catalog and update your data target
Filter by attributes or search by keyword
Name
Database
Location
Classification
accident_chainage_csv
vicroads
s3://assignment2-data-bucket/ACCIDEN...
csv
accident_csv
vicroads
s3://assignment2-data-bucket/ACCIDENT...
csv
accident_event_csv
vicroads
s3://assignment2-data-bucket/ACCIDEN...
csv
accident_location_csv
vicroads
s3://assignment2-data-bucket/ACCIDEN...
csv
atmospheric_cond_csv
vicroads
s3://assignment2-data-bucket/ATMOSPH...
csv
node_csv
vicroads
s3://assignment2-data-bucket/NODE.csv
csv
Back
Next


aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
job-load-accident-data
Data source
accident_csv
Transform type
Change schema
Data target
accident_csv
Schema
Map the source columns to target columns.
Verify the mappings created by AWS Glue. Change mappings by choosing other columns with Map to target. You can Clear all mappings and Reset to defau
AWS Glue generates your script with the defined mappings.
Source
Target
Add co
Column name
Data type
Map to target
Column name
Data type
accident_no
string
accident_no
accident_no
string
accidentdate
string
accidentdate
accidentdate
string
accidenttime
string
accidenttime
accidenttime
string
accident_type
bigint
accident_type
accident_type
long
accident_type_de
string
accident_type_desc
accident_type_desc
string

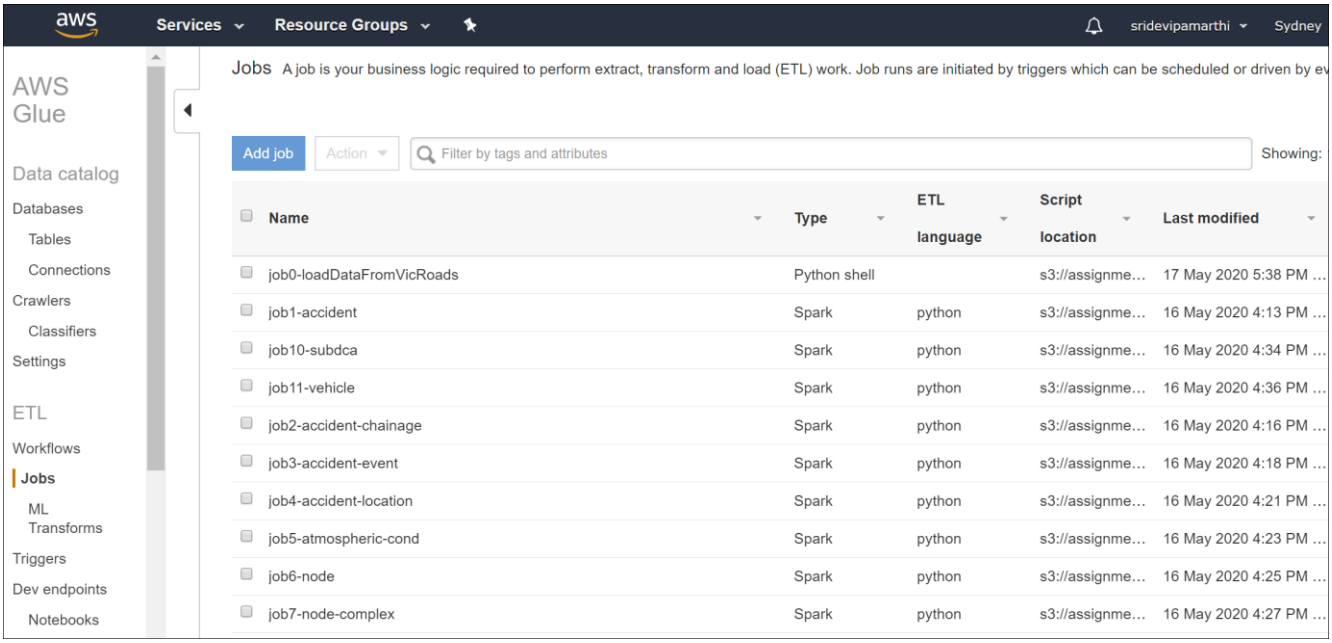


After running the above job, the data gets loaded into the database **vicroads** hosted in AWS lake.

Repeat the same steps for all the remaining tables too and run those jobs to load the data into the database.

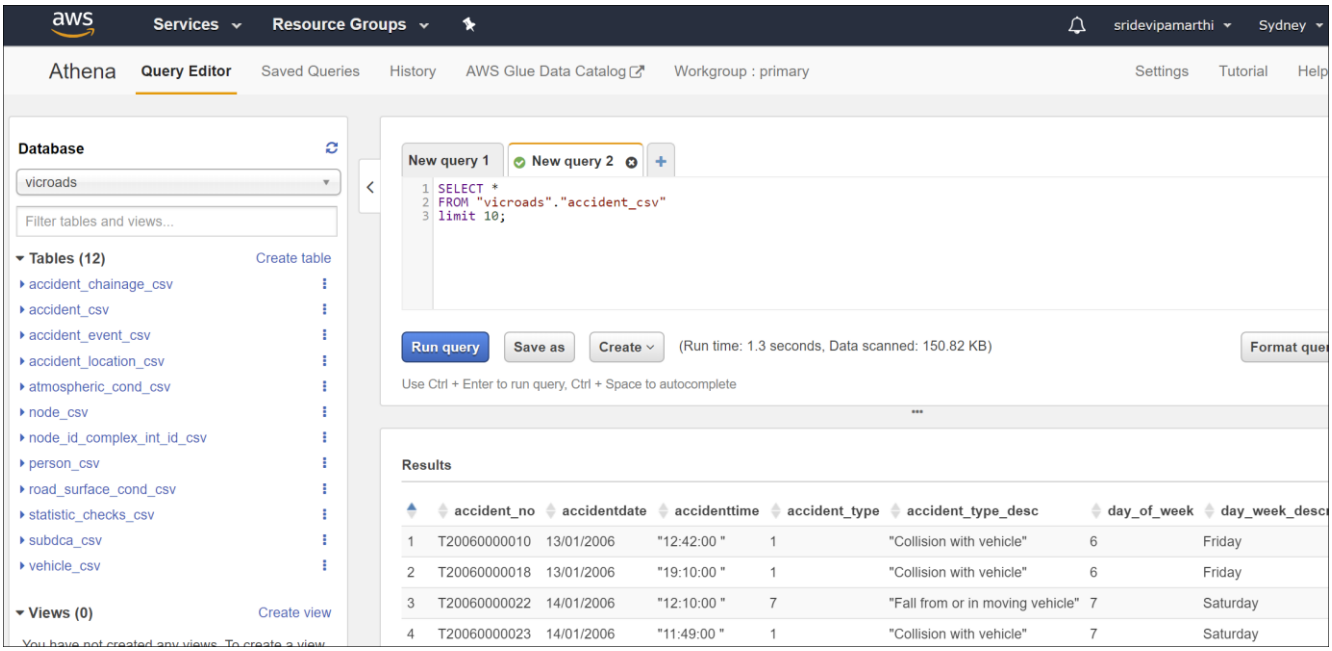
Below is the list of jobs created in this assignment.

- job1-accident
- job2-accident-chainage
- job3-accident-event
- job4-accident-location
- job5-atmospheric-cond
- job6-node
- job7-node-complex
- job8-person
- job9-road-surface
- job10-subdca
- job11-vehicle



c) Validate the data using Athena

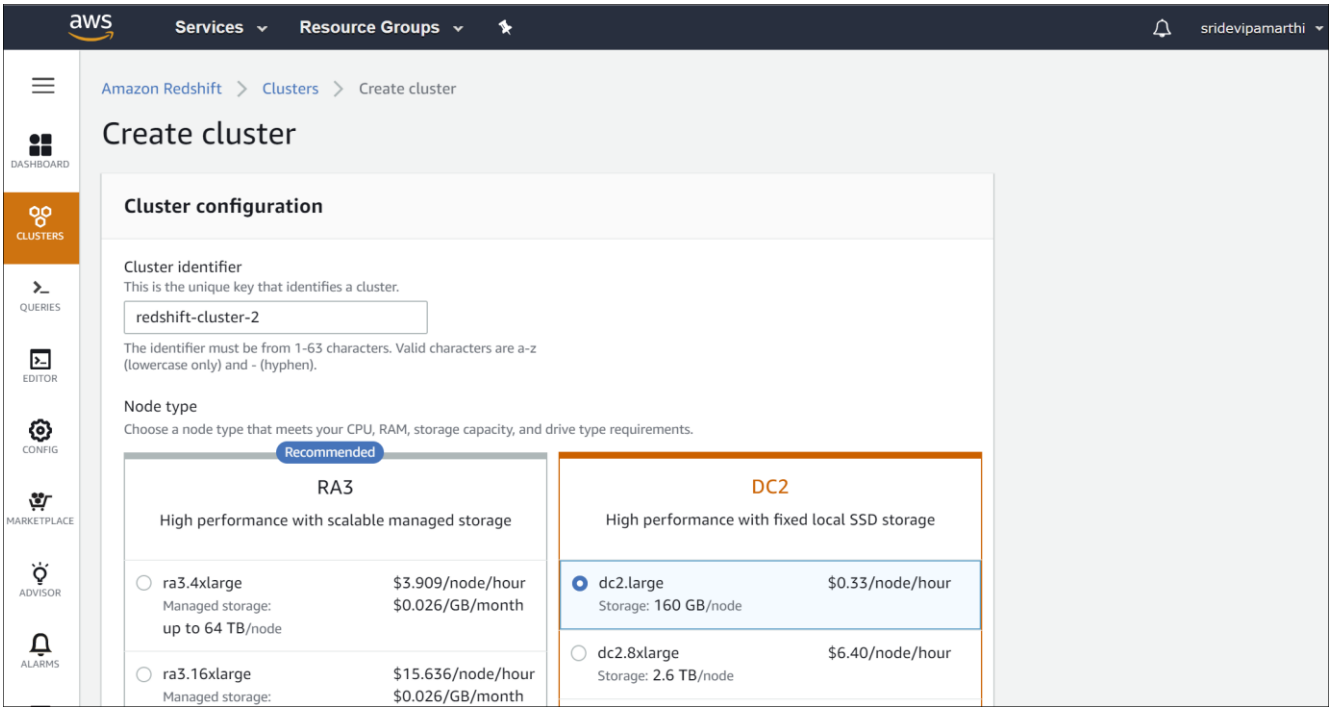
Athena is used to validate the data across all the tables before proceeding further. This is the sample screenshot to show how to use.



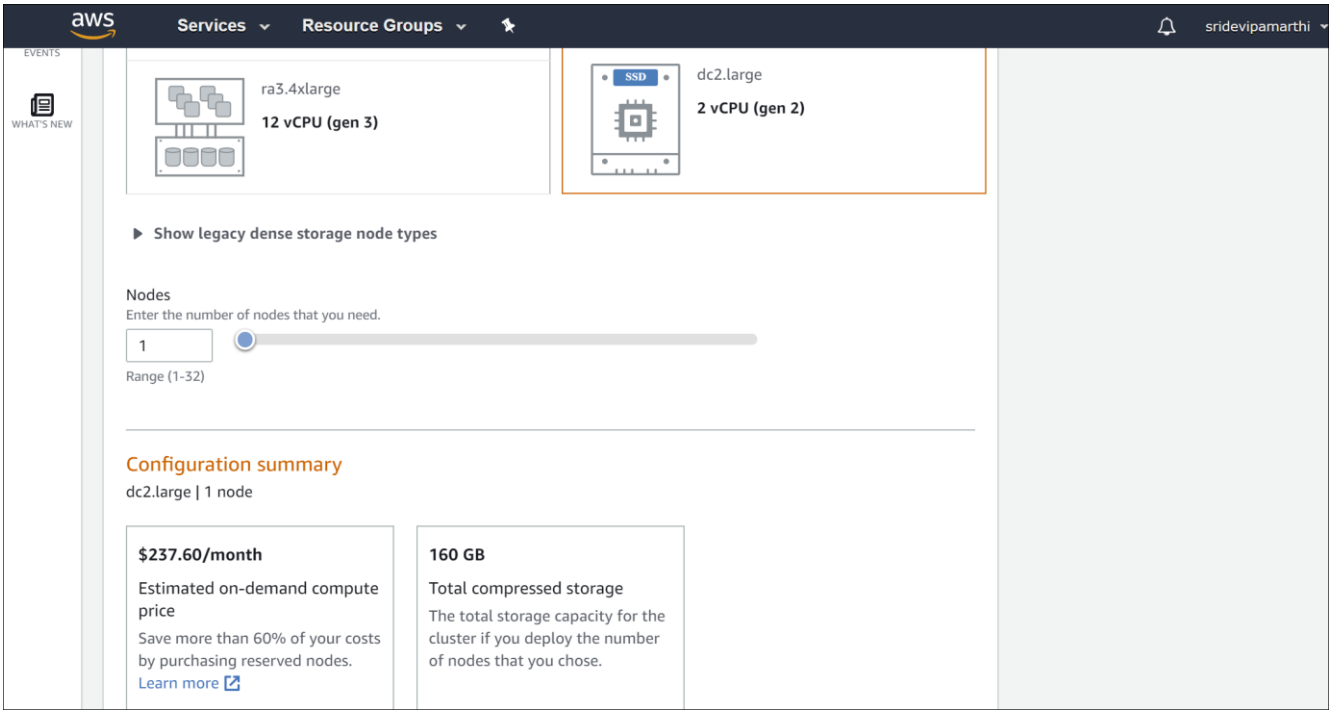
4. Setup Redshift

a) Create the Cluster:

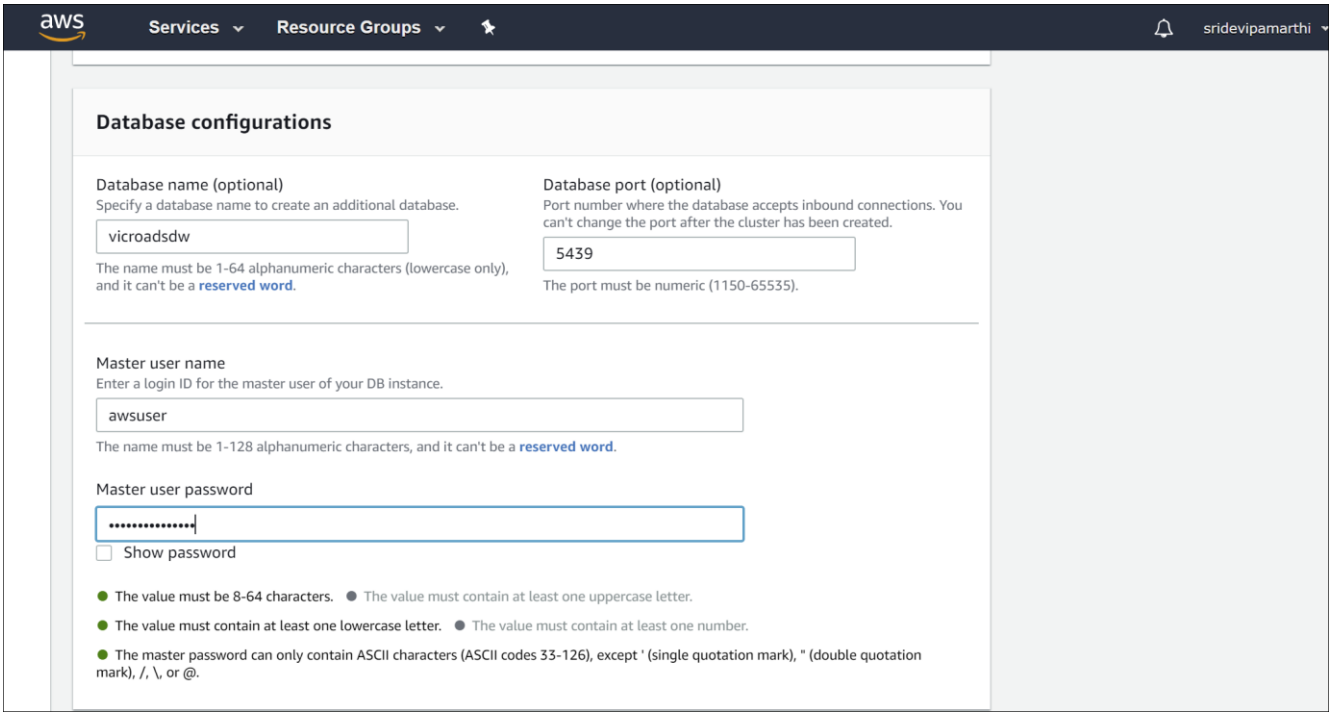
Choose the specification of node based on the complexity of the requirement, we chose the cheapest node available to control the cost.



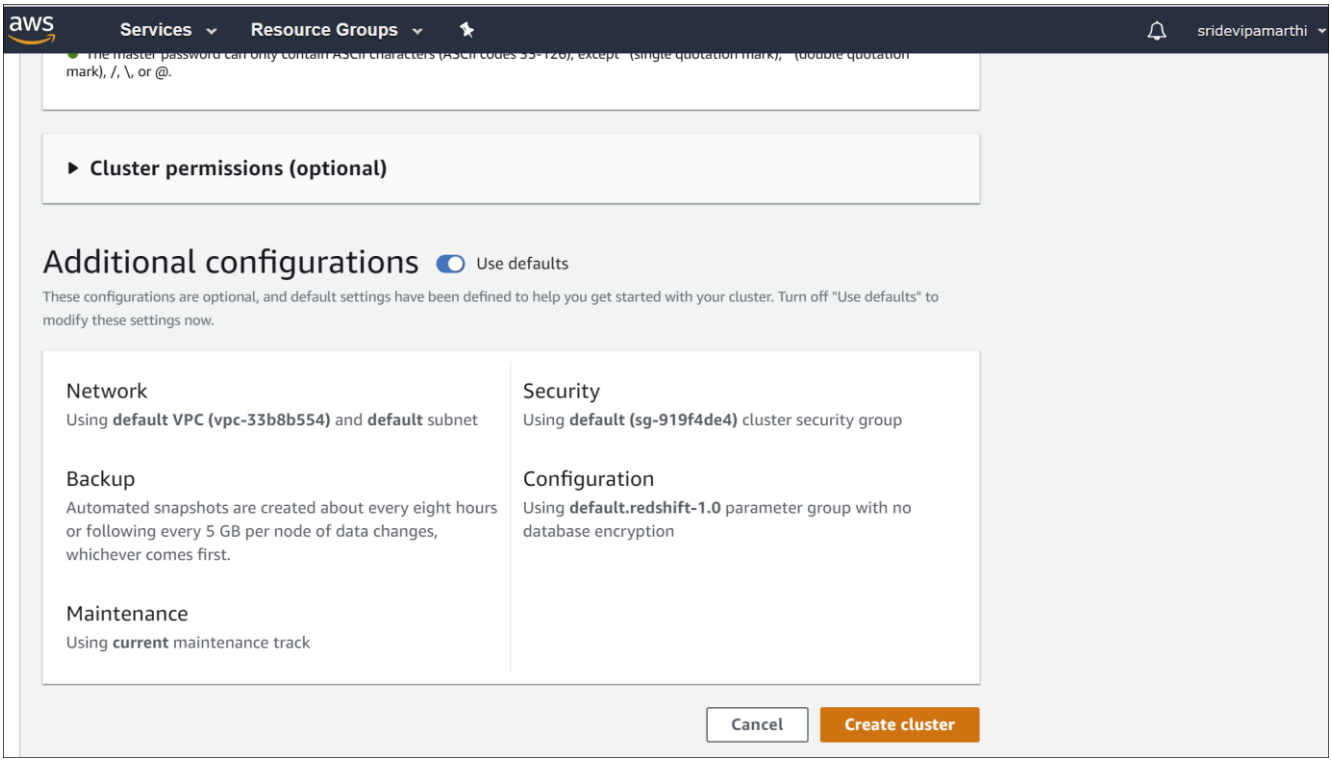
Select the number of nodes for parallel processing, this can be configured on demand and so we stick to 1 node.



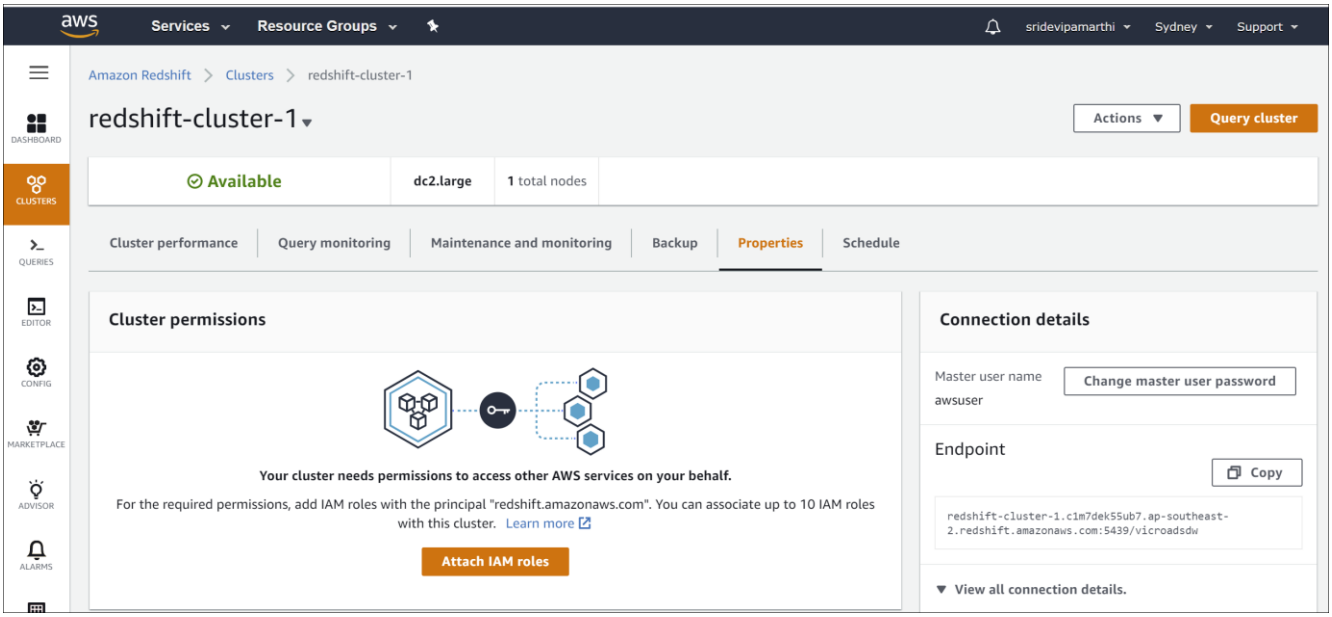
Configure the database name, port, master credentials:

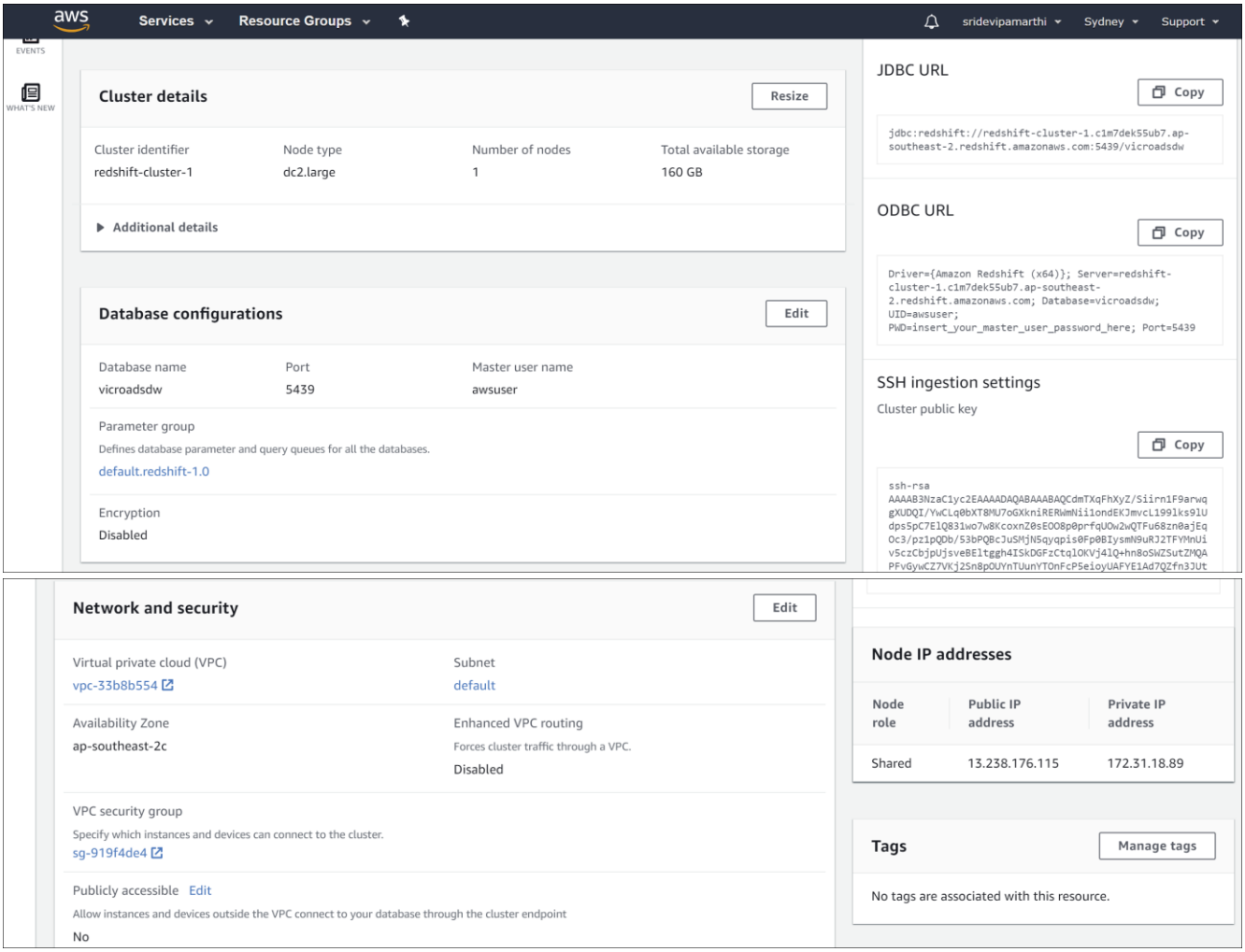


Make a note of default VPC and Security group details, these will be required for firewall configuration to enable the Quick sight access to Redshift database. Select ‘Create cluster’ to proceed.



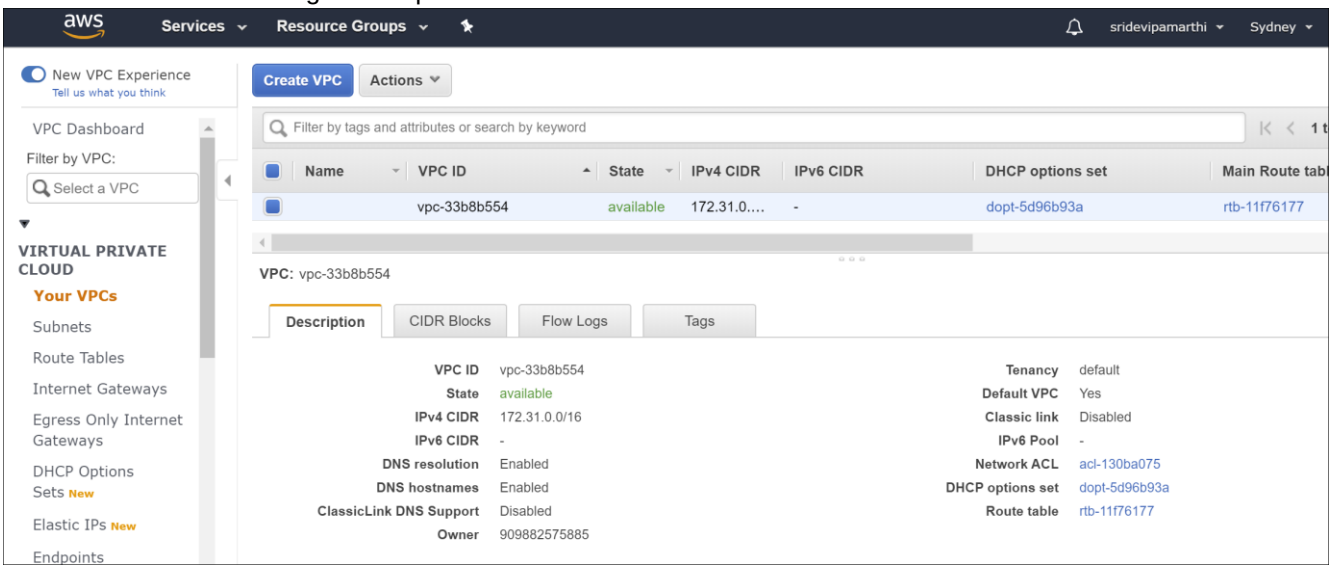
Once created, navigate to cluster properties to find all the technical information required for accessing Redshift from various services. Such as JDBC/ODBC connection strings, IP addresses etc.



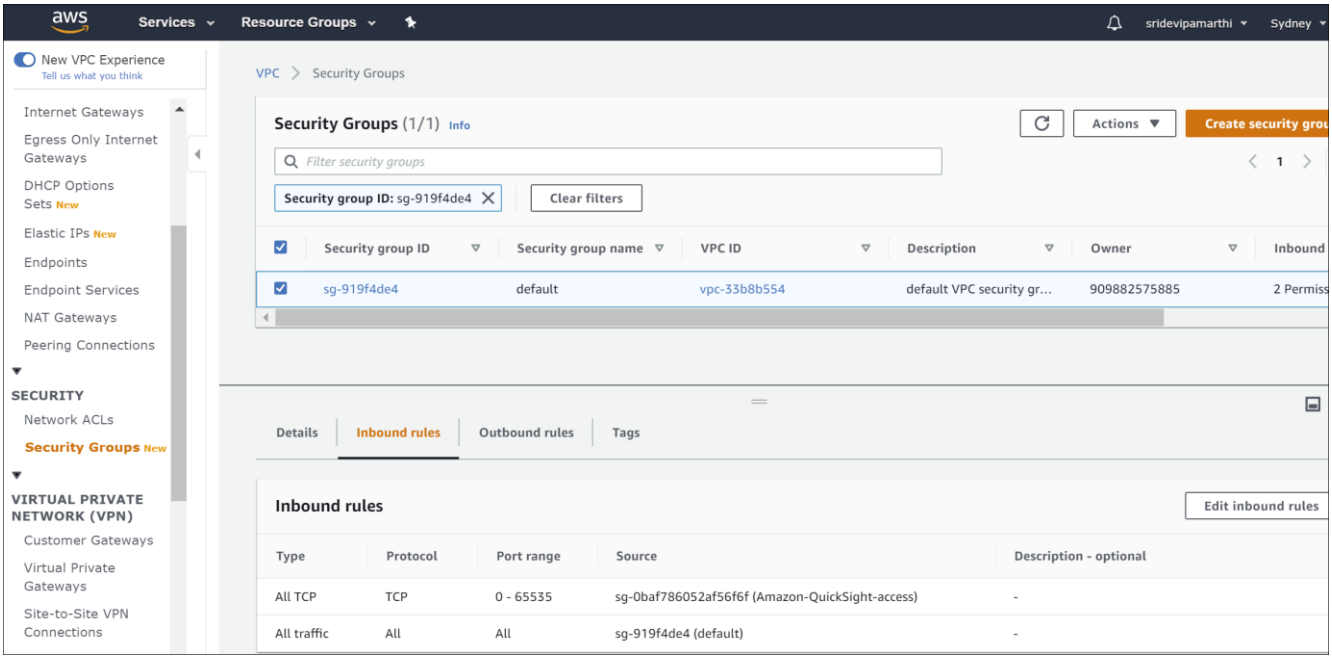


b) Configure VPC

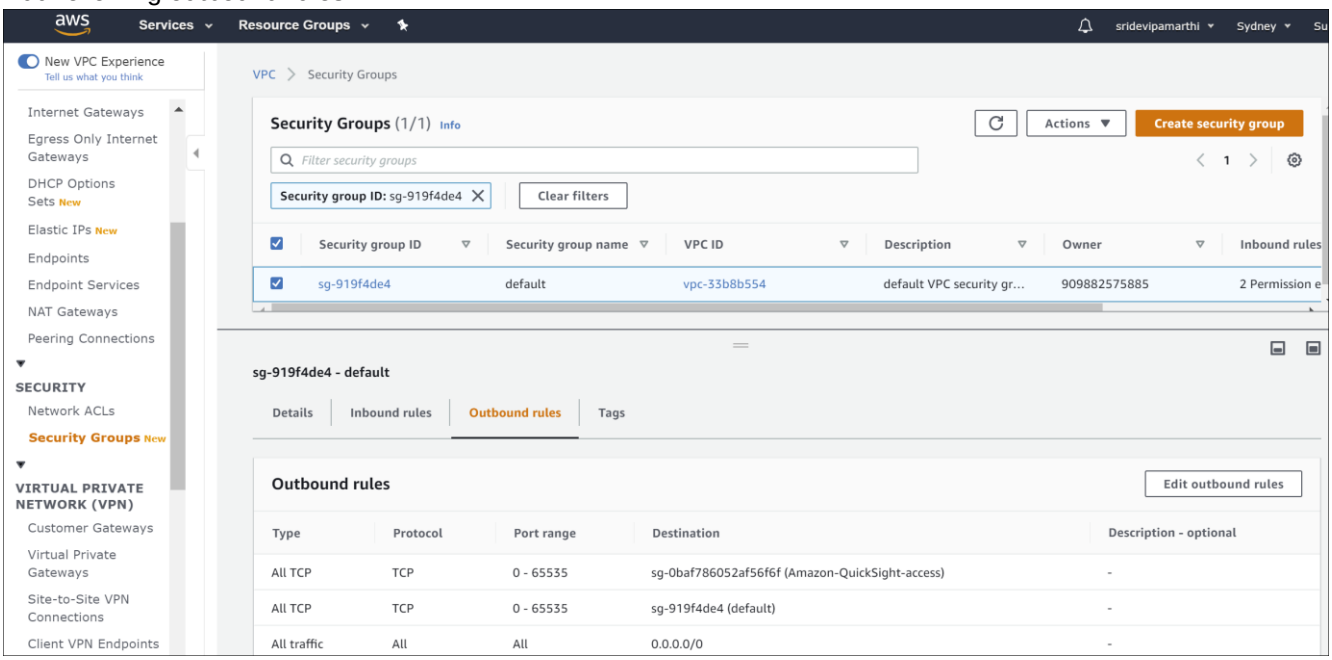
Make sure the VPC settings are in place as follows:



Add following inbound rules to default Security groups to allow network traffic/ access

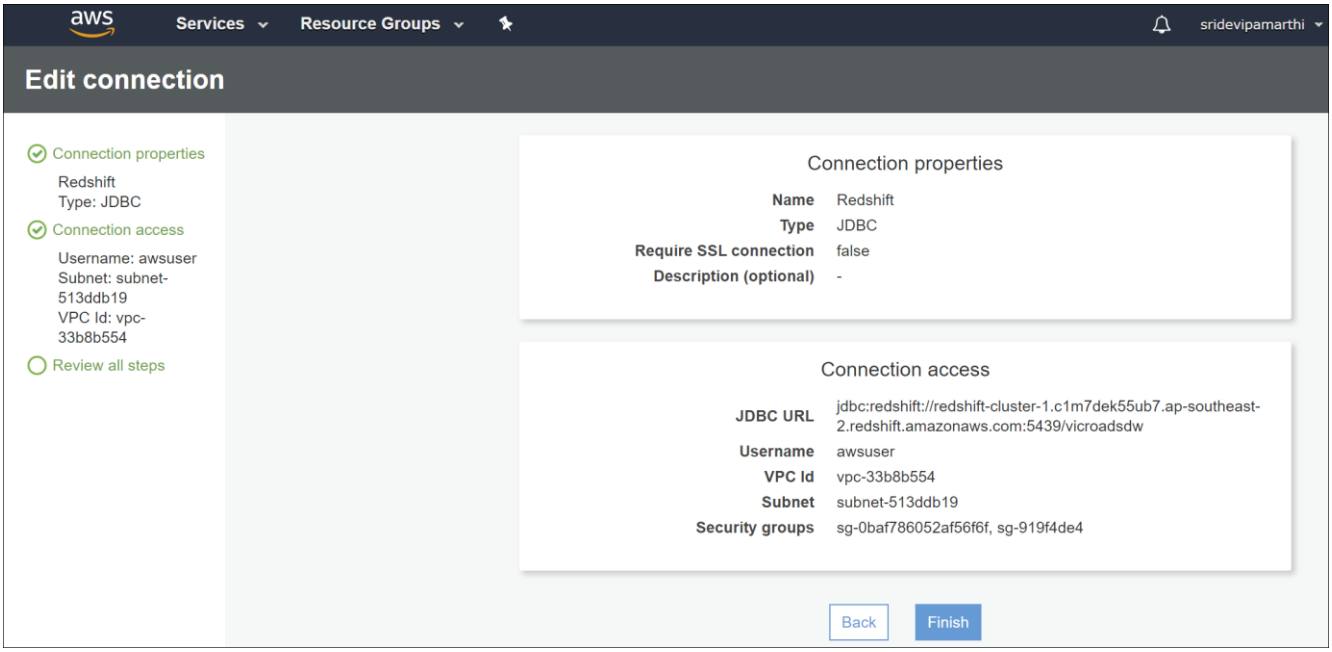


Add following outbound rules:

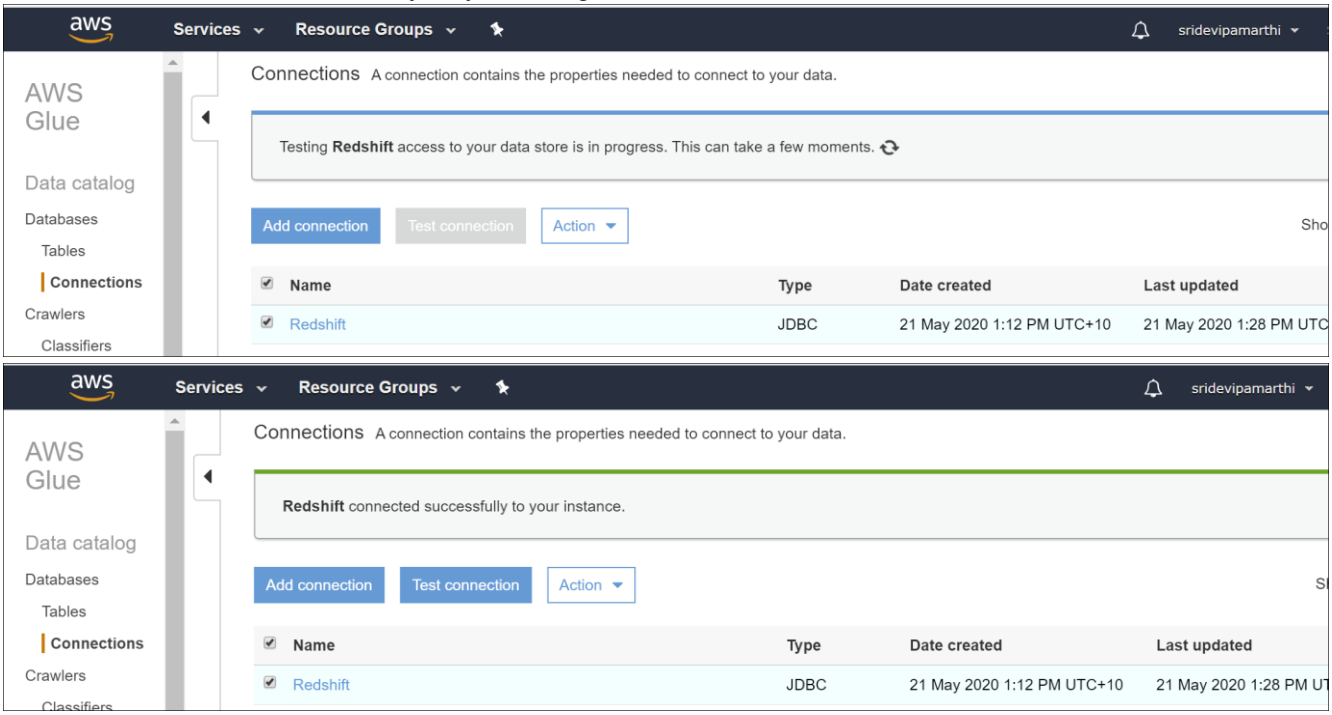


c) Create connection in AWS Glue

It is needed to load data from **vicroads** database to Redshift database **vicroadsdw** using Glue job.

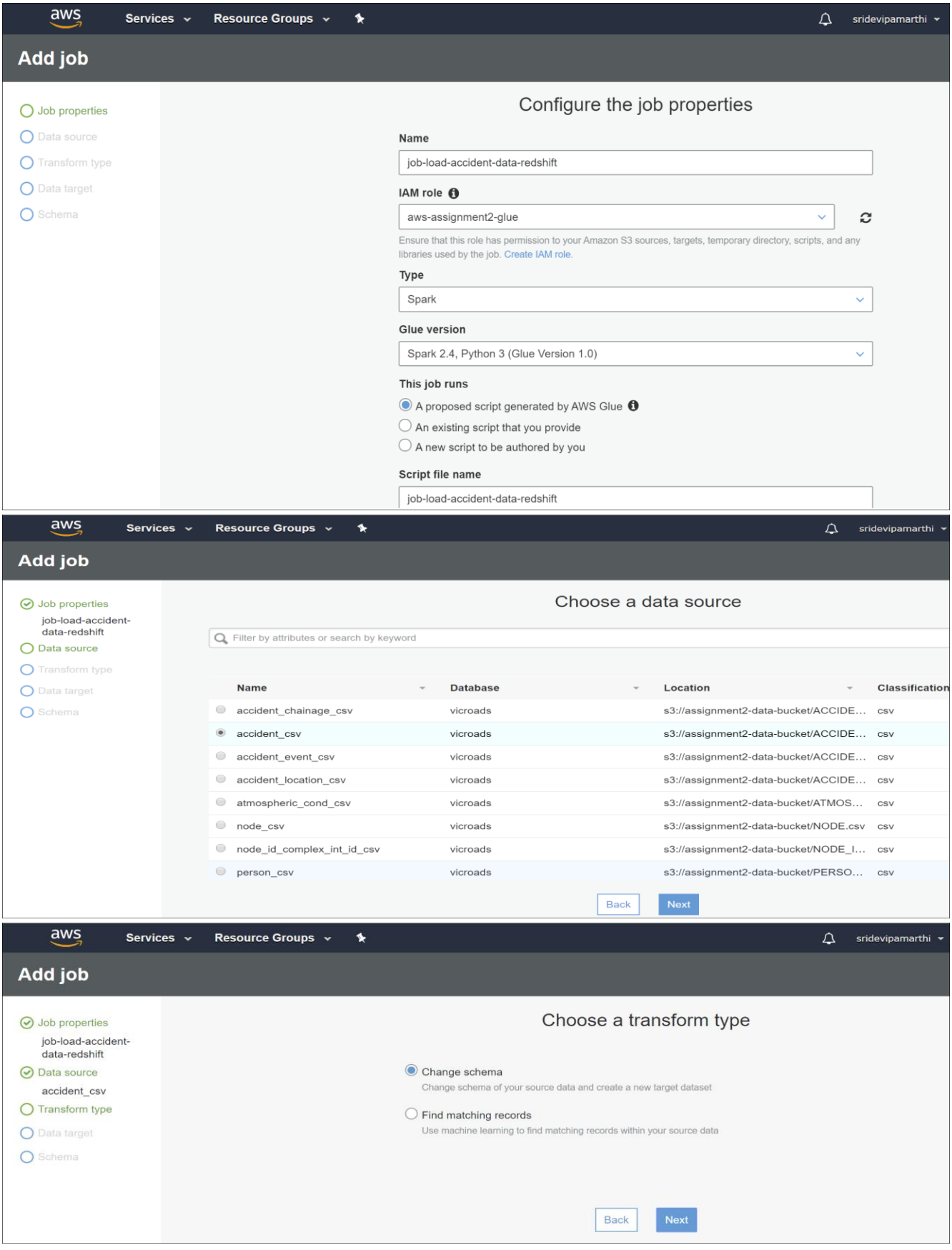


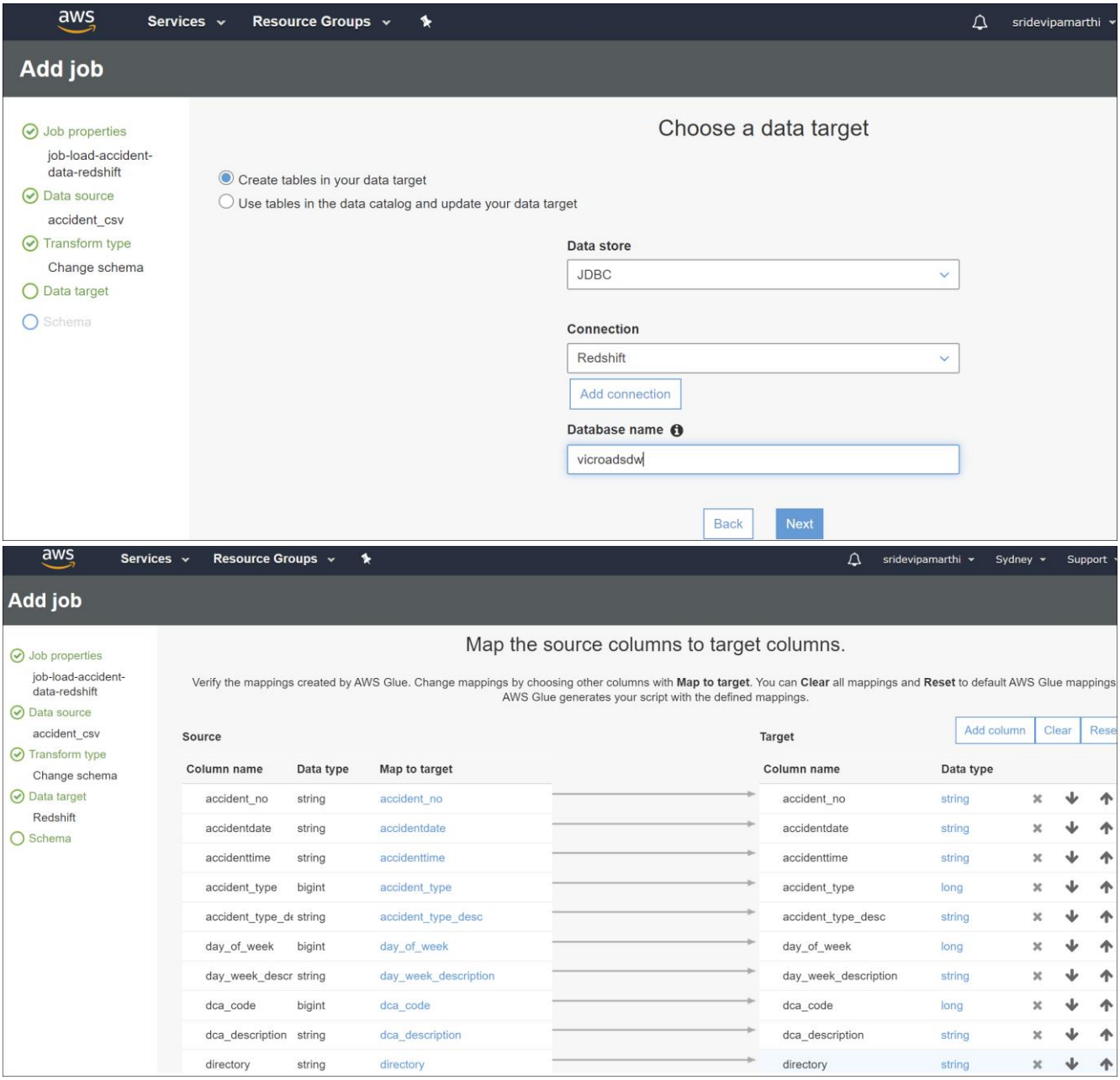
Once the connection created, verify it by selecting 'test connection' as shown below.

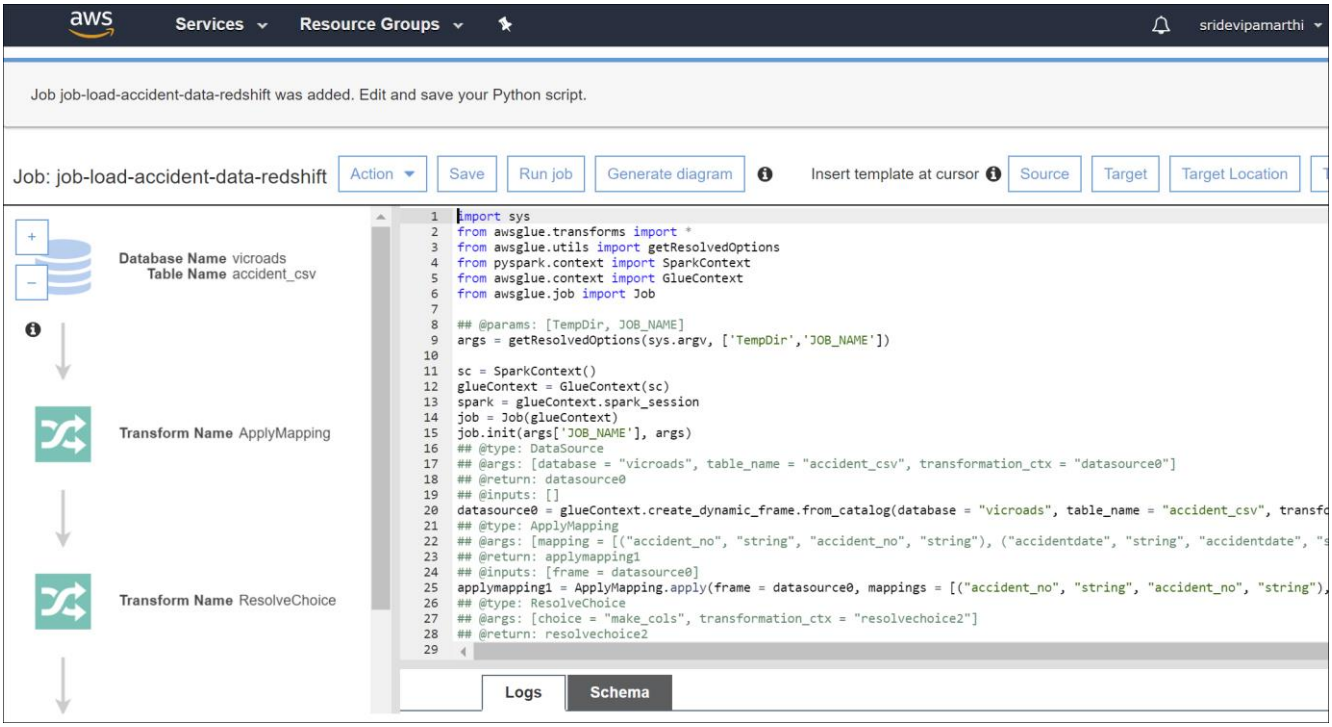


d) Create Glue job to load data to Redshift

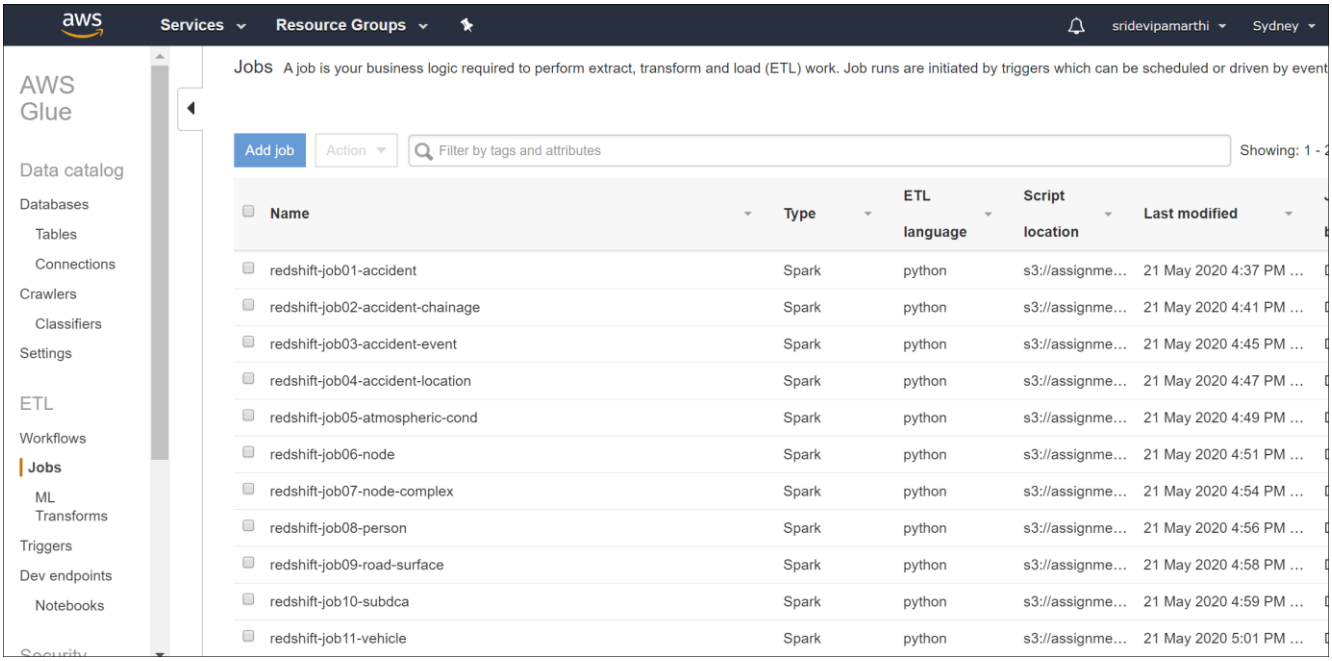
Follow the configuration given in below screenshots to create the job


aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
Data source
Transform type
Data target
Schema
Configure the job properties
Name
job-load-accident-data-redshift
IAM role
aws-assignment2-glue
Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. Create IAM role.
Type
Spark
Glue version
Spark 2.4, Python 3 (Glue Version 1.0)
This job runs
A proposed script generated by AWS Glue
An existing script that you provide
A new script to be authored by you
Script file name
job-load-accident-data-redshift
aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
job-load-accident-data-redshift
Data source
Transform type
Data target
Schema
Choose a data source
Filter by attributes or search by keyword
Name
Database
Location
Classification
accident_chainage_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
accident_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
accident_event_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
accident_location_csv
vicroads
s3://assignment2-data-bucket/ACCIDE...
csv
atmospheric_cond_csv
vicroads
s3://assignment2-data-bucket/ATMOS...
csv
node_csv
vicroads
s3://assignment2-data-bucket/NODE.csv
csv
node_id_complex_int_id_csv
vicroads
s3://assignment2-data-bucket/NODE_I...
csv
person_csv
vicroads
s3://assignment2-data-bucket/PERSO...
csv
Back
Next
aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
job-load-accident-data-redshift
Data source
accident_csv
Transform type
Data target
Schema
Choose a transform type
Change schema
Change schema of your source data and create a new target dataset
Find matching records
Use machine learning to find matching records within your source data
Back
Next

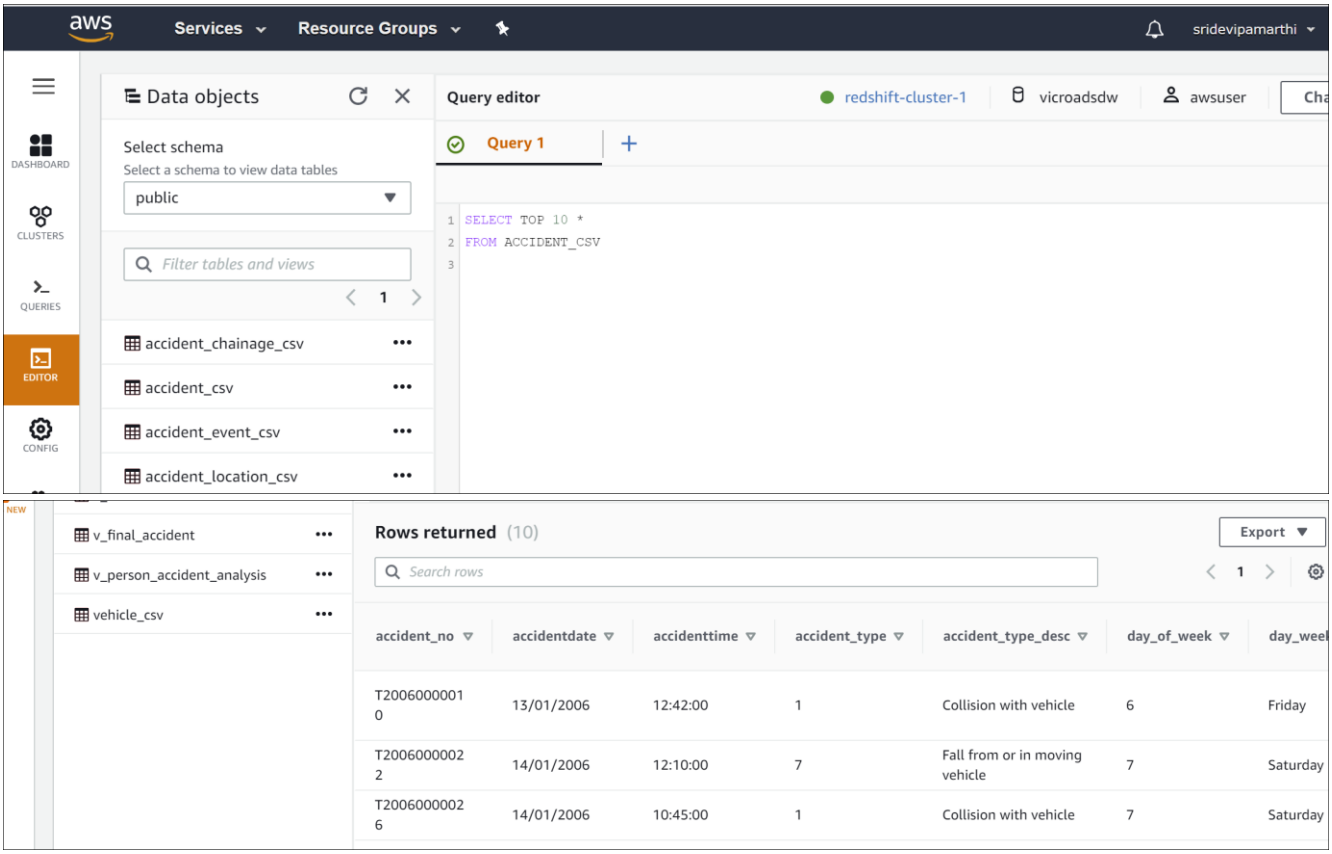

aws
Services
Resource Groups
sridevipamarthi
Add job
Job properties
job-load-accident-data-redshift
Data source
accident_csv
Transform type
Change schema
Data target
Schema
Choose a data target
Create tables in your data target
Use tables in the data catalog and update your data target
Data store
JDBC
Connection
Redshift
Add connection
Database name
vicroadsdw
Back
Next
aws
Services
Resource Groups
sridevipamarthi
Sydney
Support
Add job
Job properties
job-load-accident-data-redshift
Data source
accident_csv
Transform type
Change schema
Data target
Redshift
Schema
Map the source columns to target columns.
Verify the mappings created by AWS Glue. Change mappings by choosing other columns with Map to target. You can Clear all mappings and Reset to default AWS Glue mappings
AWS Glue generates your script with the defined mappings.
Source
Target
Add column
Clear
Column name
Data type
Map to target
Column name
Data type
accident_no string accident_no accident_no string
accidentdate string accidentdate accidentdate string
accidenttime string accidenttime accidenttime string
accident_type bigint accident_type accident_type long
accident_type_de string accident_type_desc accident_type_desc string
day_of_week bigint day_of_week day_of_week long
day_week_descr string day_week_description day_week_description string
dca_code bigint dca_code dca_code long
dca_description string dca_description dca_description string
directory string directory directory string



Run the job to load data into Redshift database. The following jobs are created for this assignment specific to each table, as follows.



e) Verify the data in Redshift



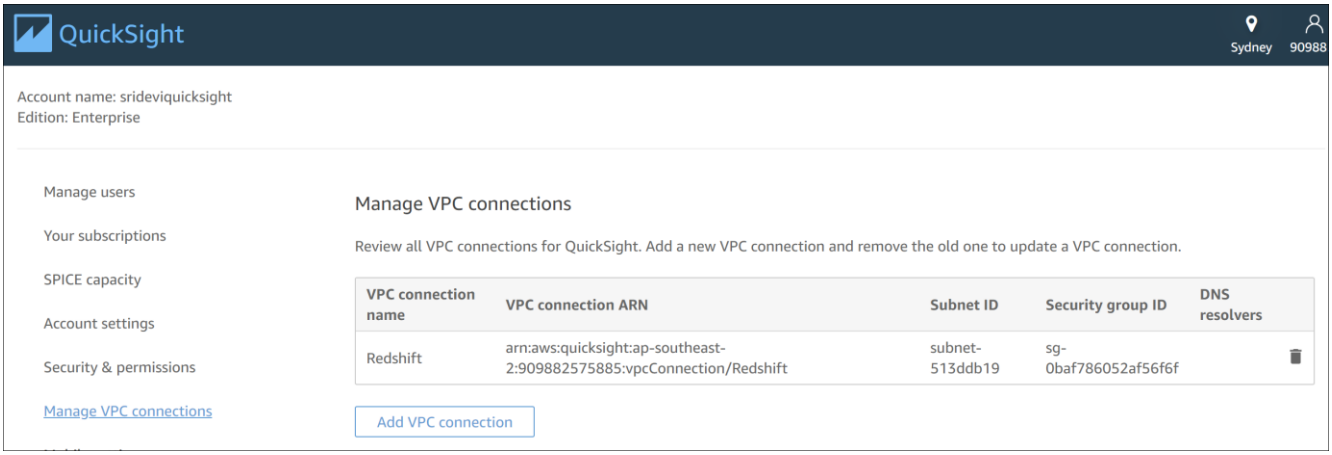
f) Create transformation views

Views can be created by joining the associated tables which returns the dataset required for reporting and analytics. Below is the sample view definition v_Person_Accident_Analysis to show how the data is transformed for the next steps.

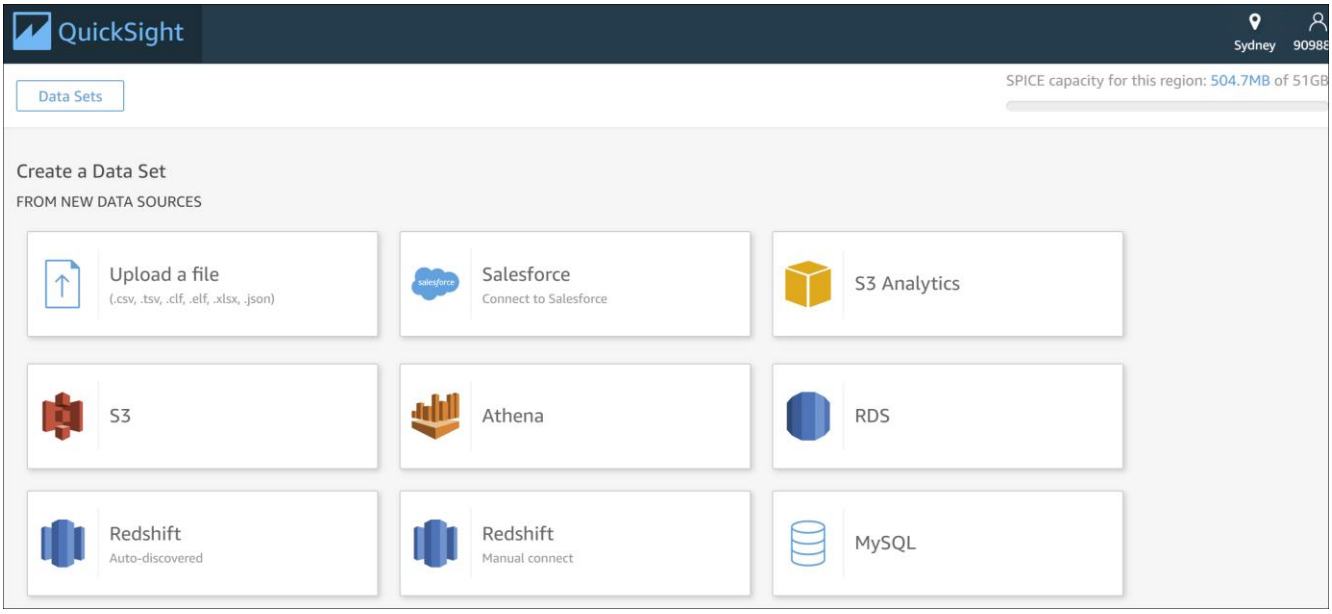
```
CREATE VIEW v_Person_Accident_Analysis AS

SELECT
  COALESCE(P.INJ_LEVEL_DESC,'Unknown') AS INJ_LEVEL_DESC
, COALESCE(P.AGE_GROUP,'unknown') AS AGE_GROUP
, COALESCE(A.ACCIDENT_TYPE,'0') AS ACCIDENT_TYPE
, COALESCE(A.ACCIDENT_TYPE_DESC,'Unknown') AS ACCIDENT_TYPE_DESC
, COALESCE(P.SEX,'U') AS SEX
, COUNT(P.ACCIDENT_NO) AS ACCIDENT_COUNT

FROM PERSON_CSV P
JOIN ACCIDENT_CSV A ON P.ACCIDENT_NO = A.ACCIDENT_NO

GROUP BY COALESCE(P.INJ_LEVEL_DESC,'Unknown')
, COALESCE(P.AGE_GROUP,'unknown')
, COALESCE(A.ACCIDENT_TYPE,'0')
, COALESCE(A.ACCIDENT_TYPE_DESC,'Unknown')
, COALESCE(P.SEX,'U')
```

5. Setup Quick Sight

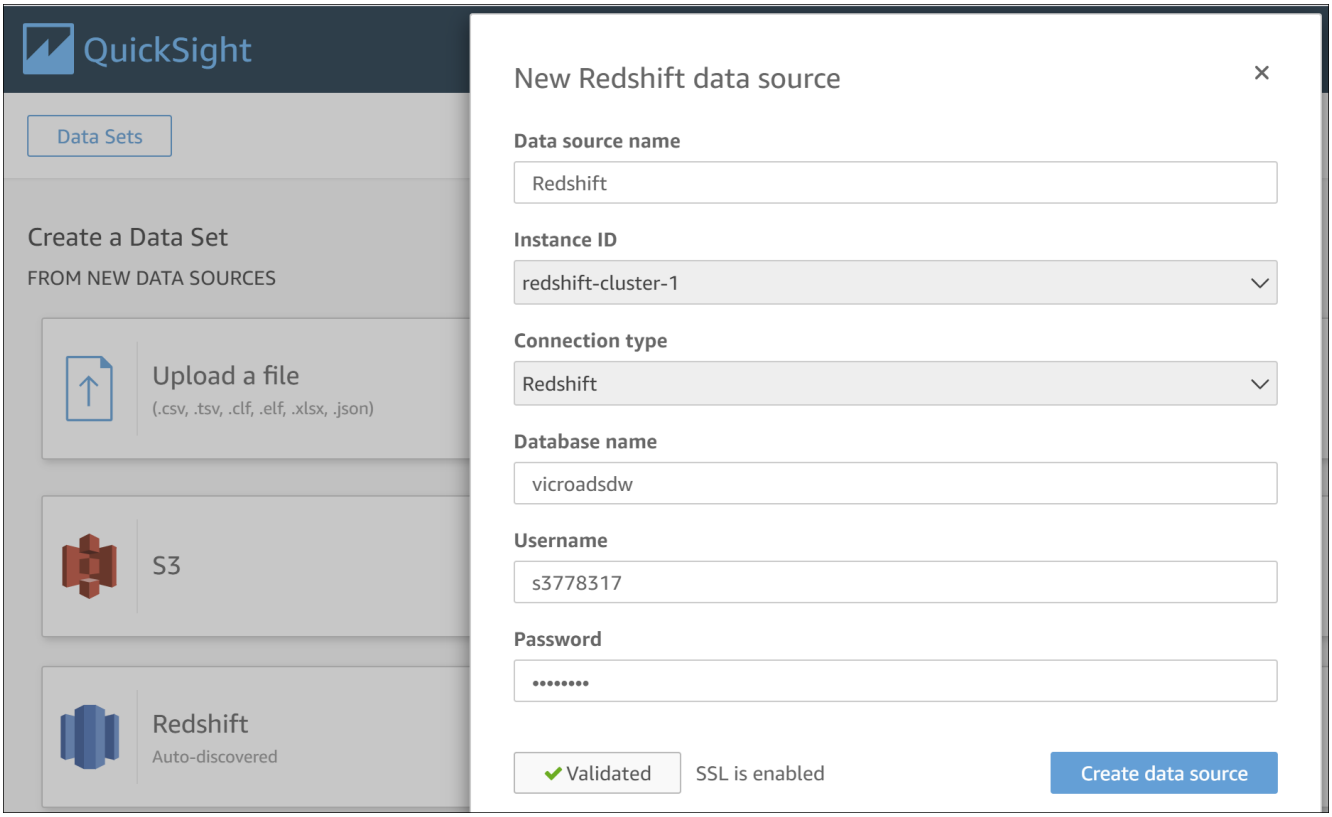
a) VPC connection to access Redshift



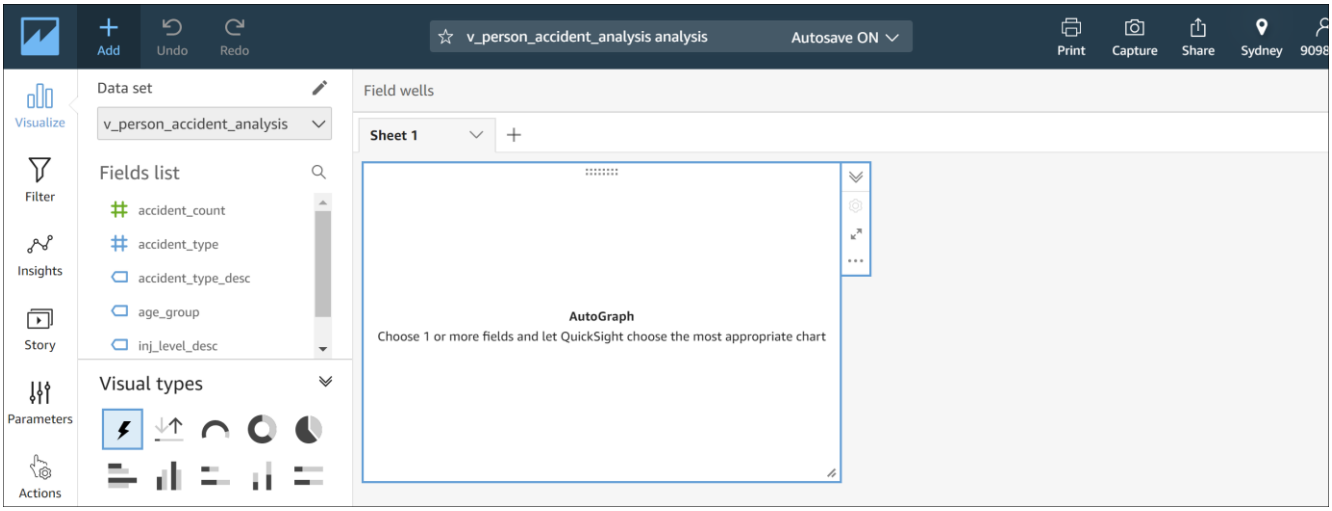
b) Create a data set from Redshift to build visual



Select Redshift (Auto discovered), input details and validate connection to create data source.



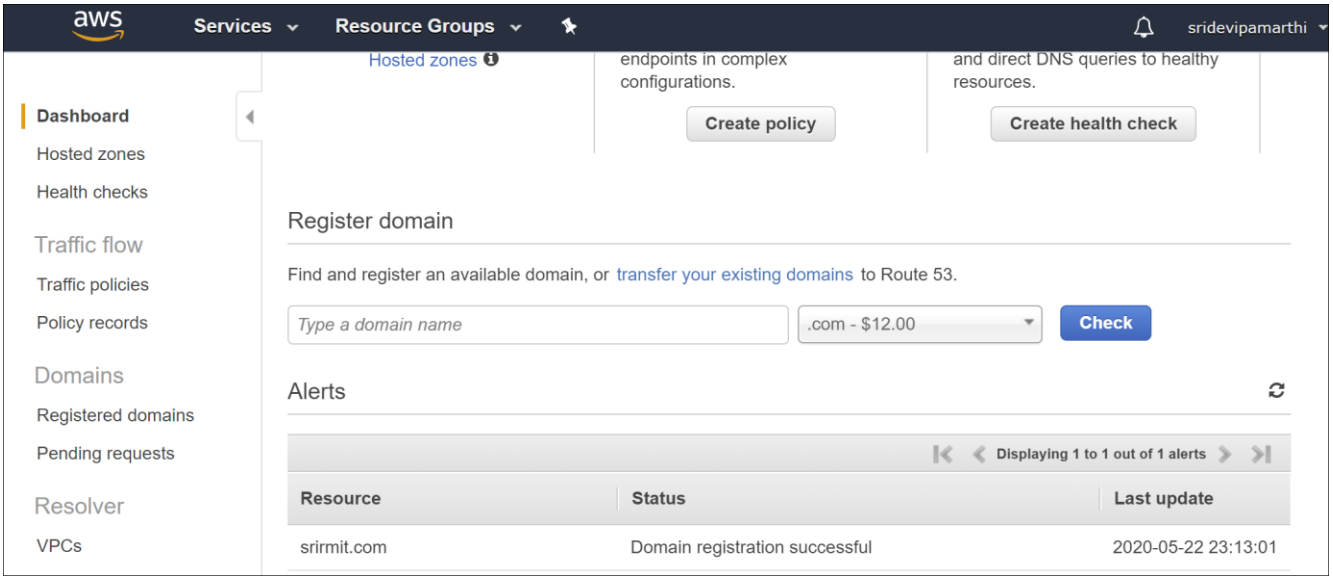
Now, we can access the view (or any other objects) created in Redshift database to build the visual.



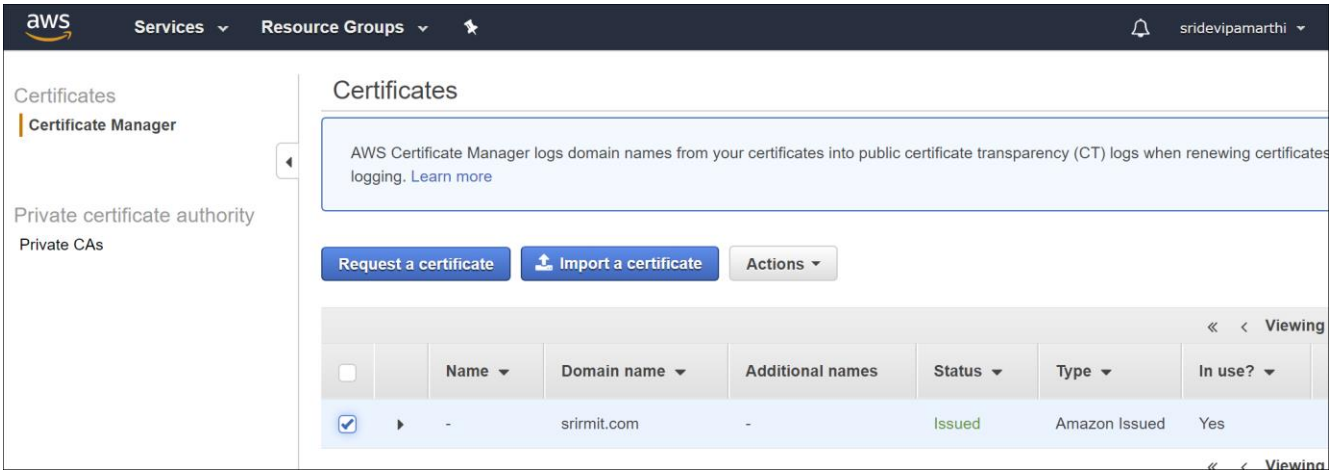
6. Set up Elastic Beanstalk

a) Register a domain (Route 53)

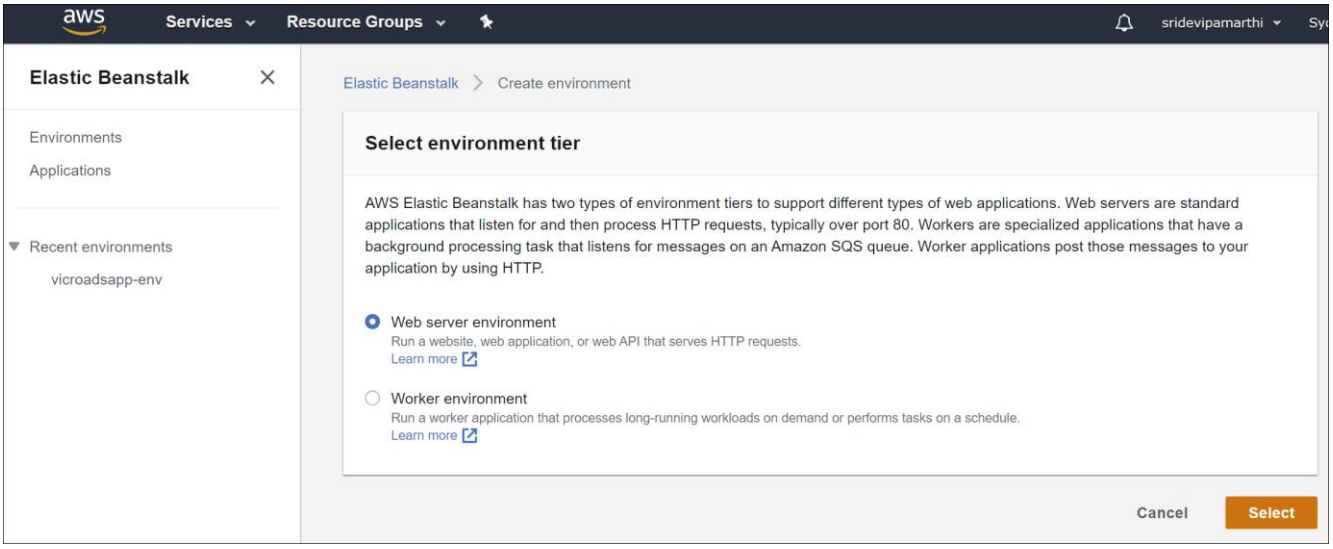
To whitelist the Elastic Beanstalk web application URL in Quick sight, the application should support HTTPS protocol and that requires an SSL certificate (which requires a domain to be setup)

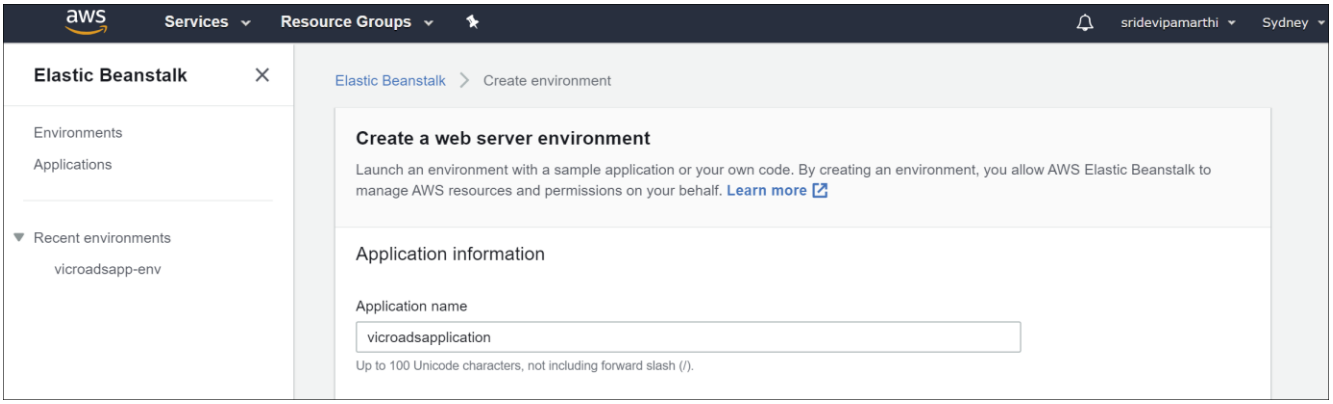


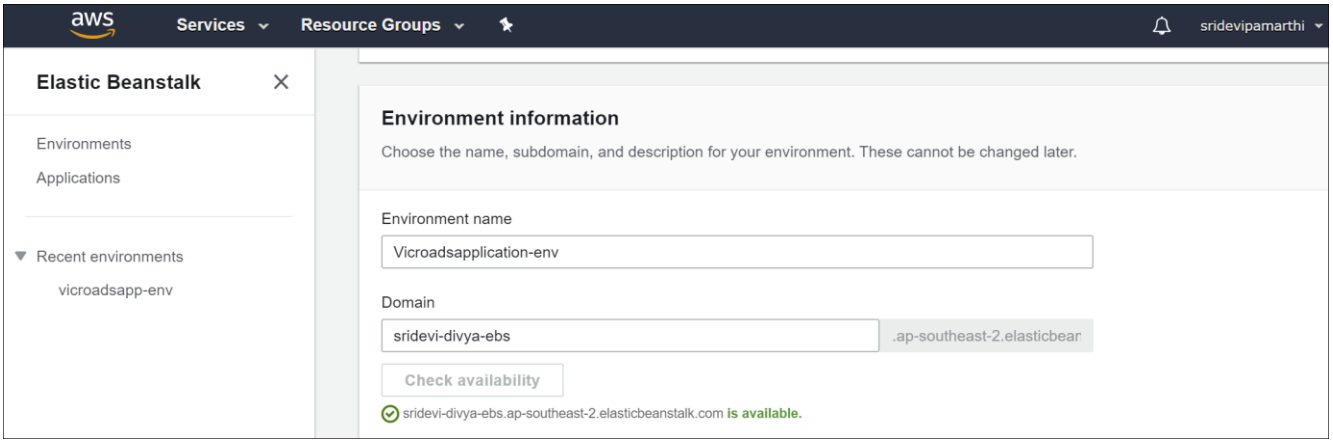
b) Request SSL certificate (Certification Manager):

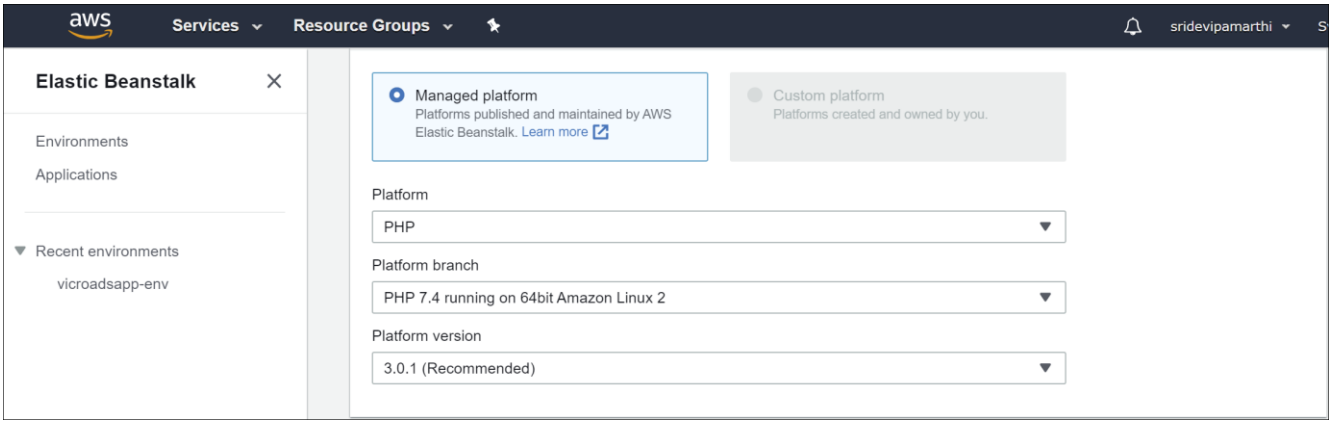


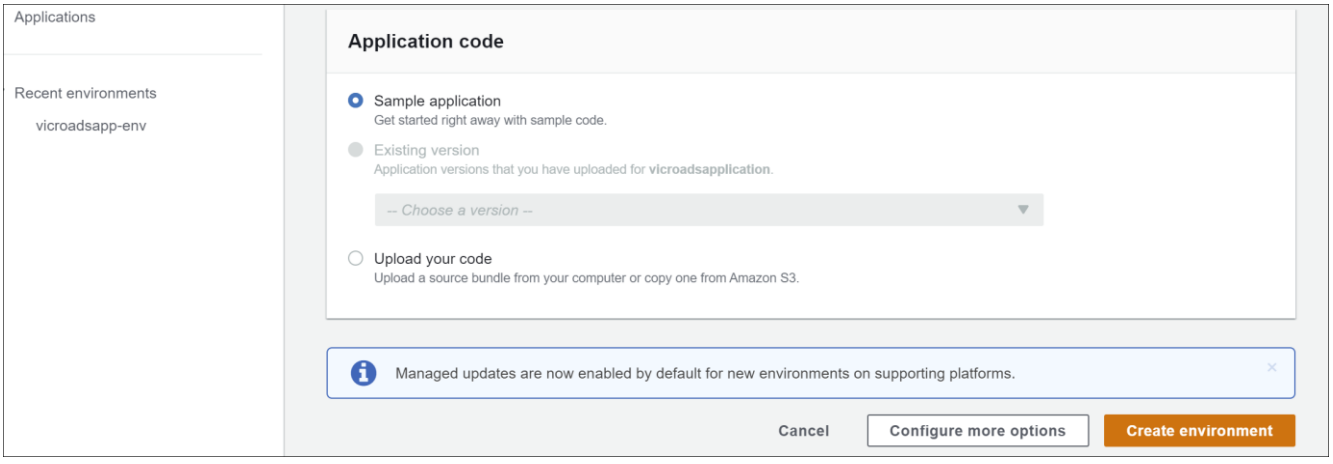
c) Create an EBS environment



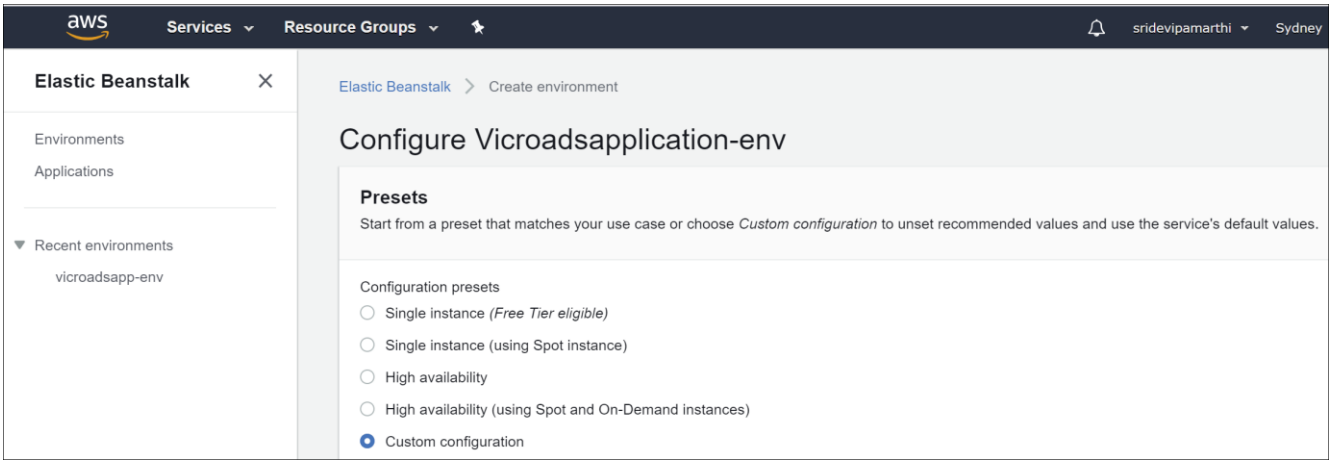


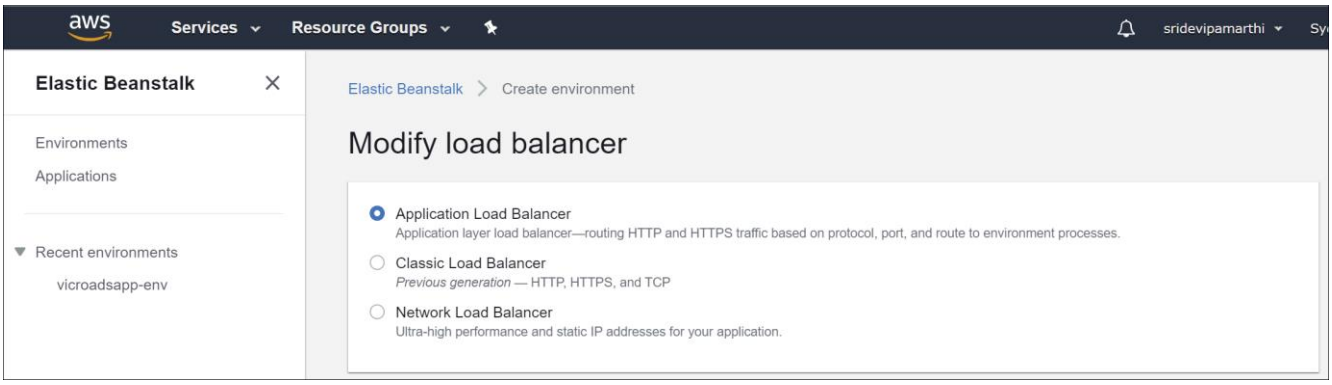




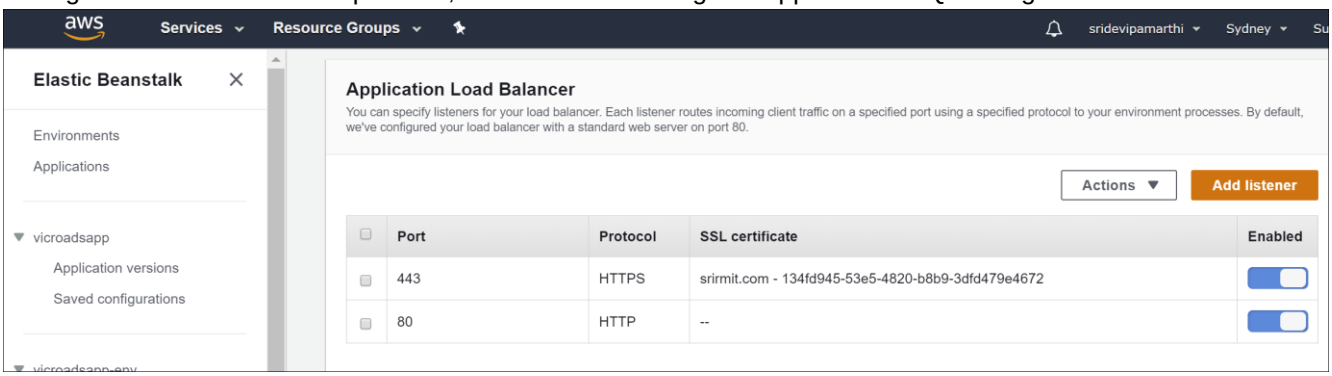


Select configure more options:

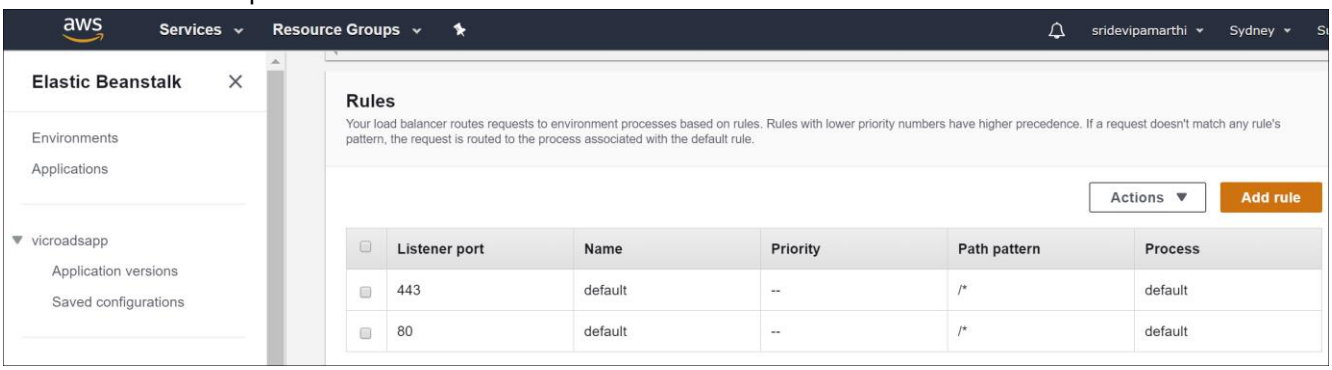




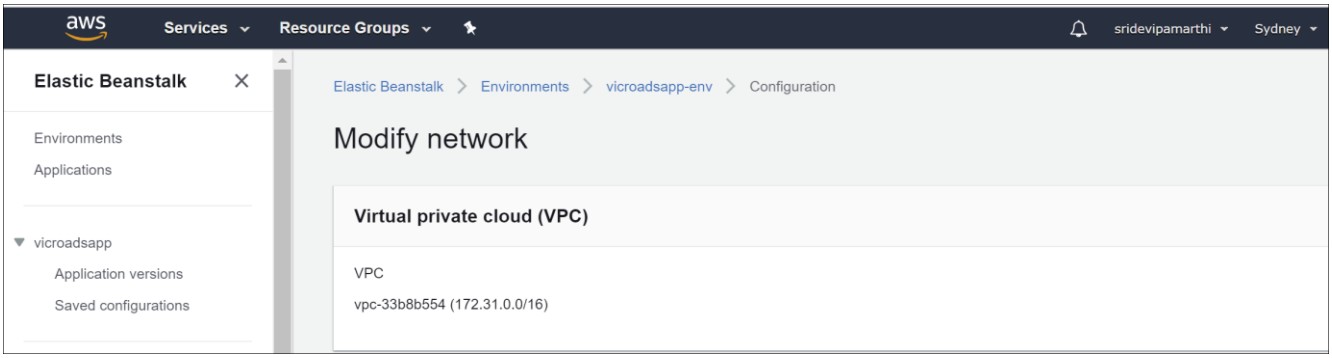
Configure listener for HTTPS protocol, this is for whitelisting this application in Quick sight.

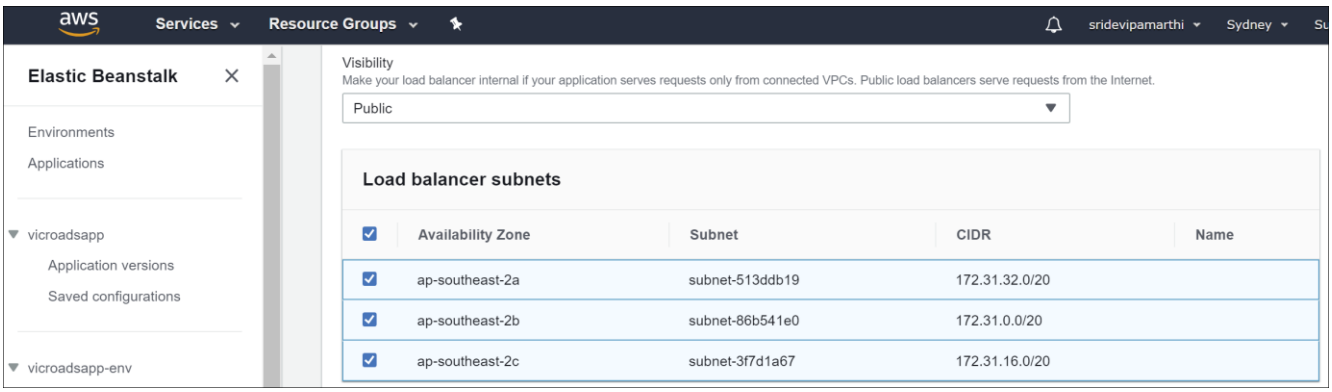


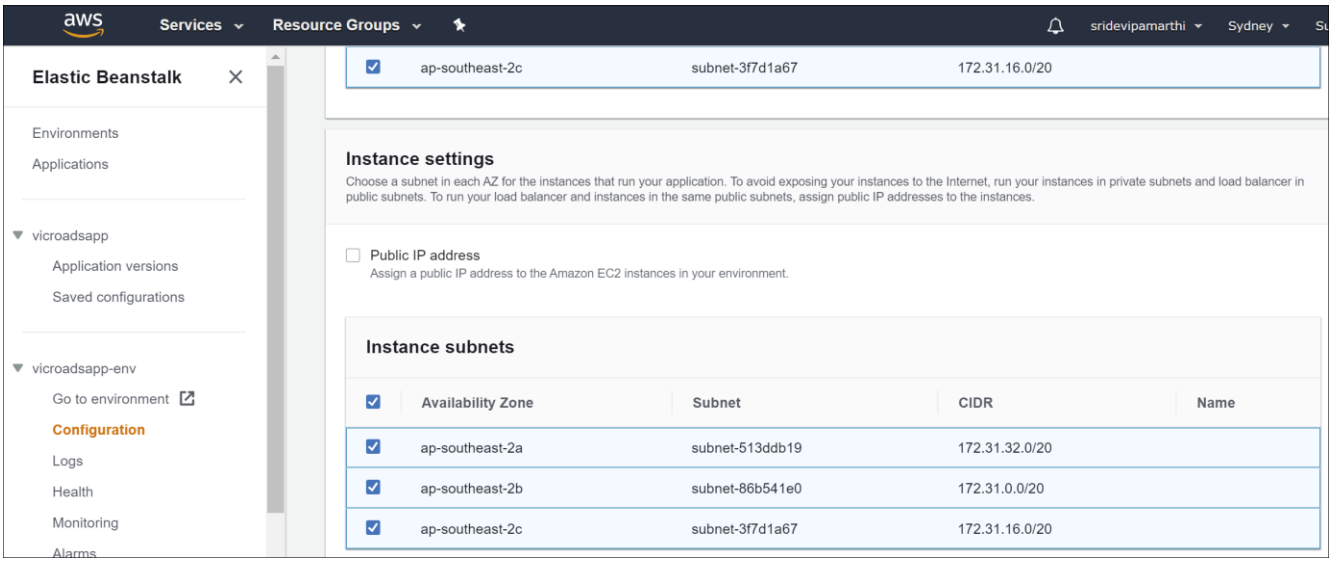
Add the new rule for port 443 for the above said reason.



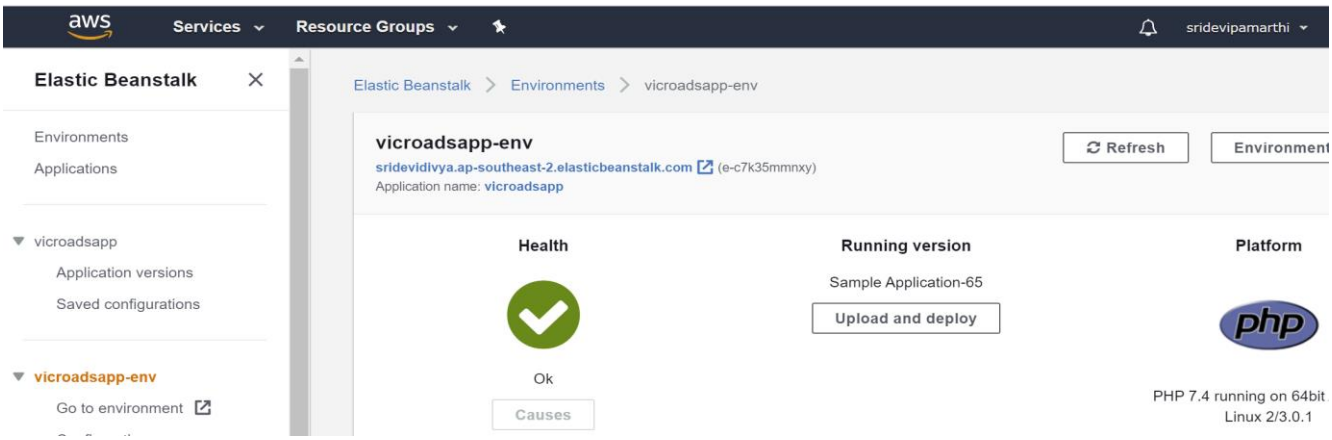
Modify VPC to add this environment into the available subnets and then create the environment.







Upload and deploy the application

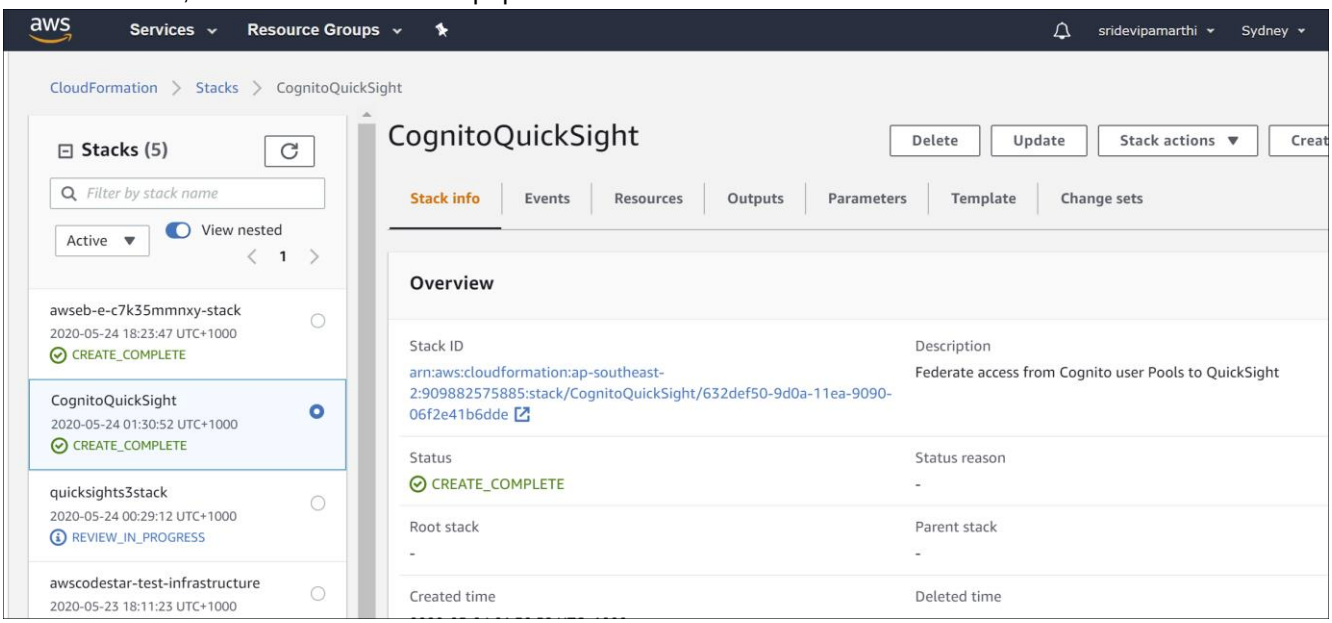


7. QuickSight Sign-On with Cognito User Pools

This has done by following the exact steps given in the URL here. The objective of this step is to allow public access to the application through CloudFront URL. With this approach, the user can sign up and validate their identity and then access the QuickSight dashboard information.

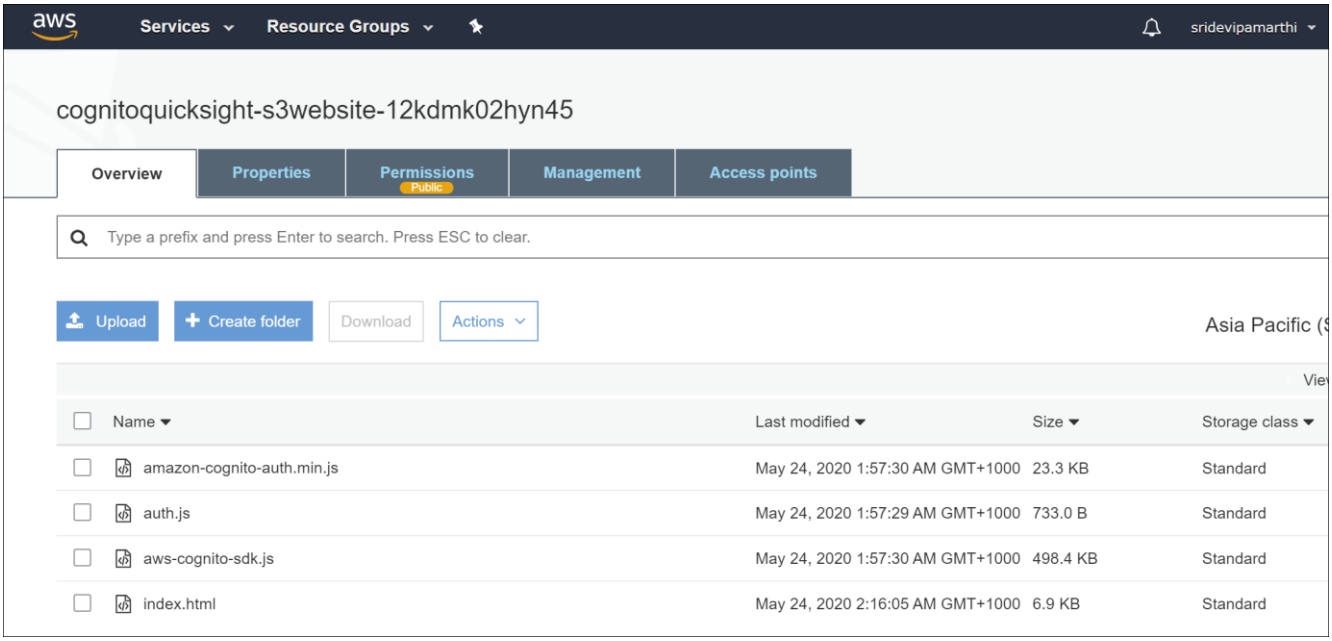
a) Run Cloud Formation stack CognitoQuickSight

To achieve this, the cloud formation script provided in the above URL has been executed.



b) Upload content into S3 static website

And then, uploaded the required files into the static S3 website created by the cloud formation stack.

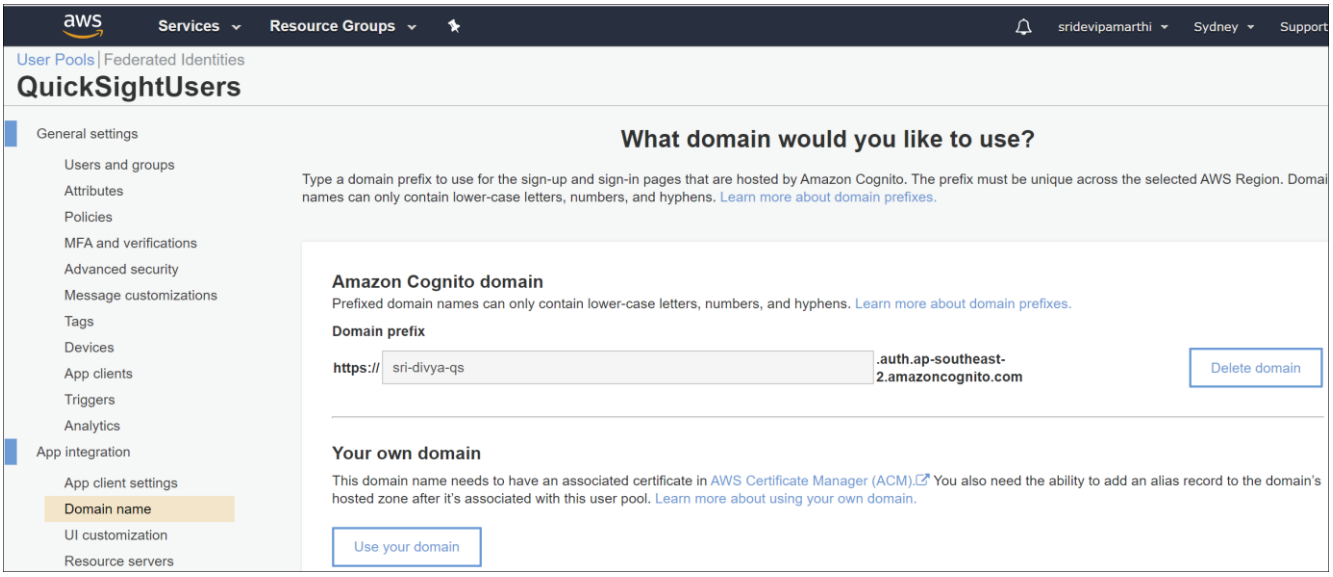


This process configured the following components automatically:

- Amazon CloudFront distribution
- S3 static website
- Amazon Cognito user pool
- Amazon Cognito identity pool
- IAM role for authenticated users
- API Gateway API
- Lambda function

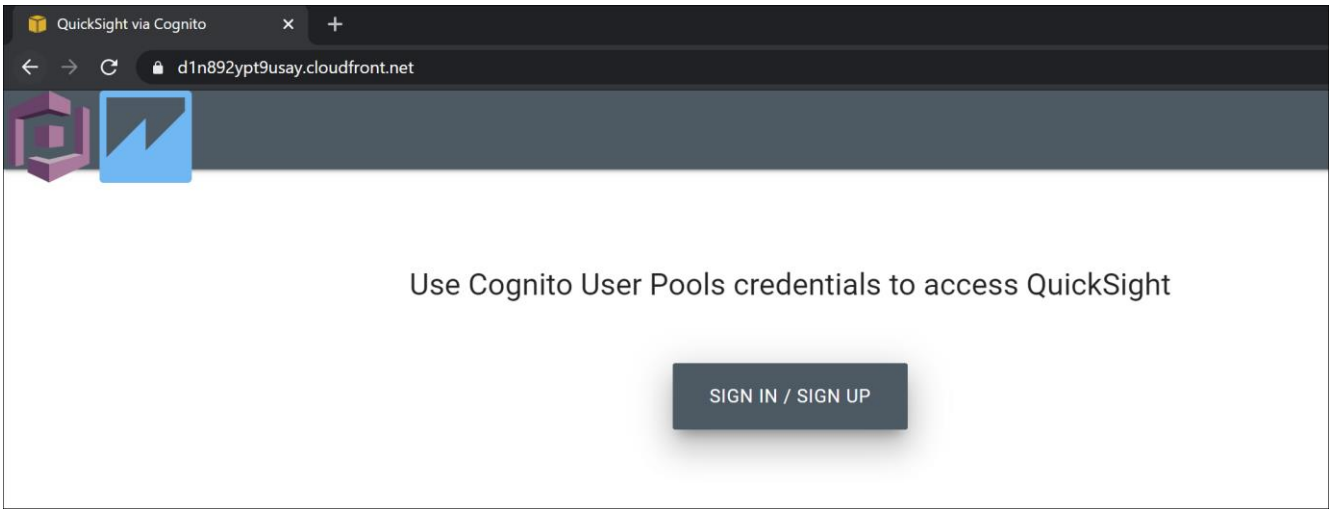
c) Cognito Configuration:

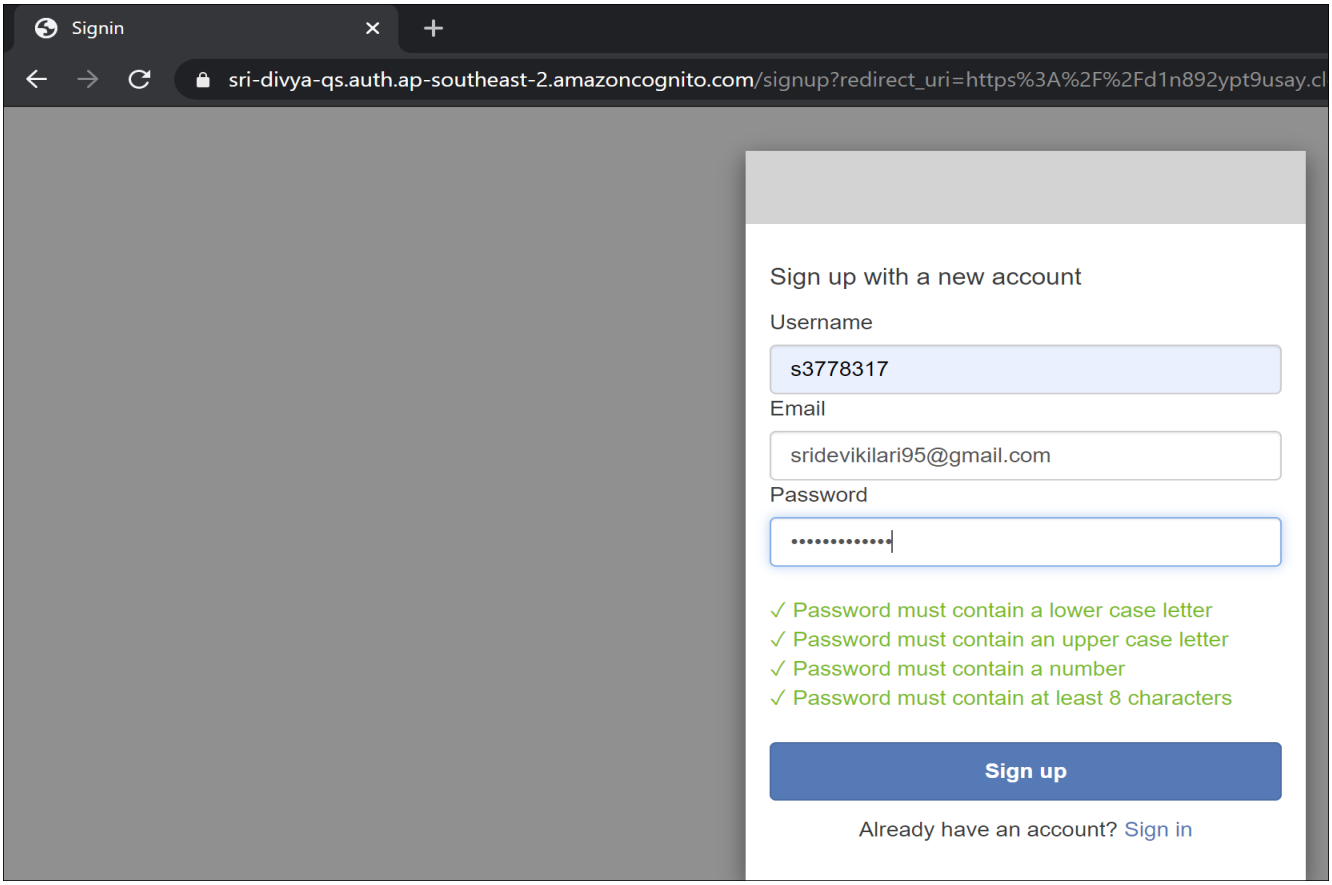
configure Cognito domain configuration in User Pool QuickSightUsers

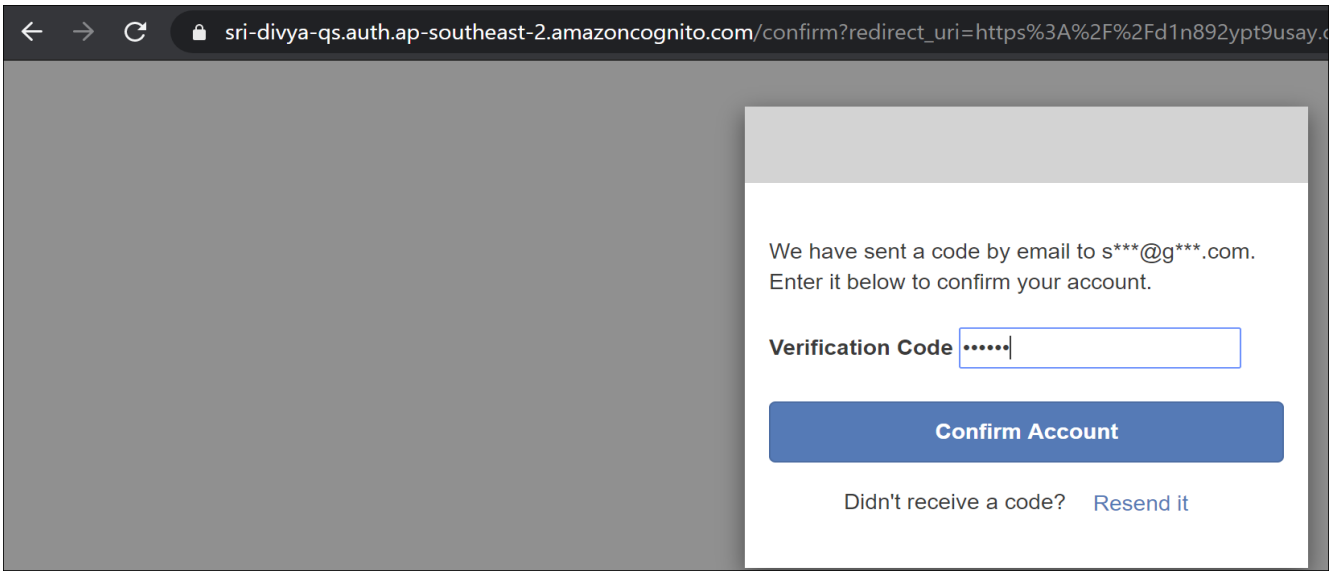


d) run CloudFront URL

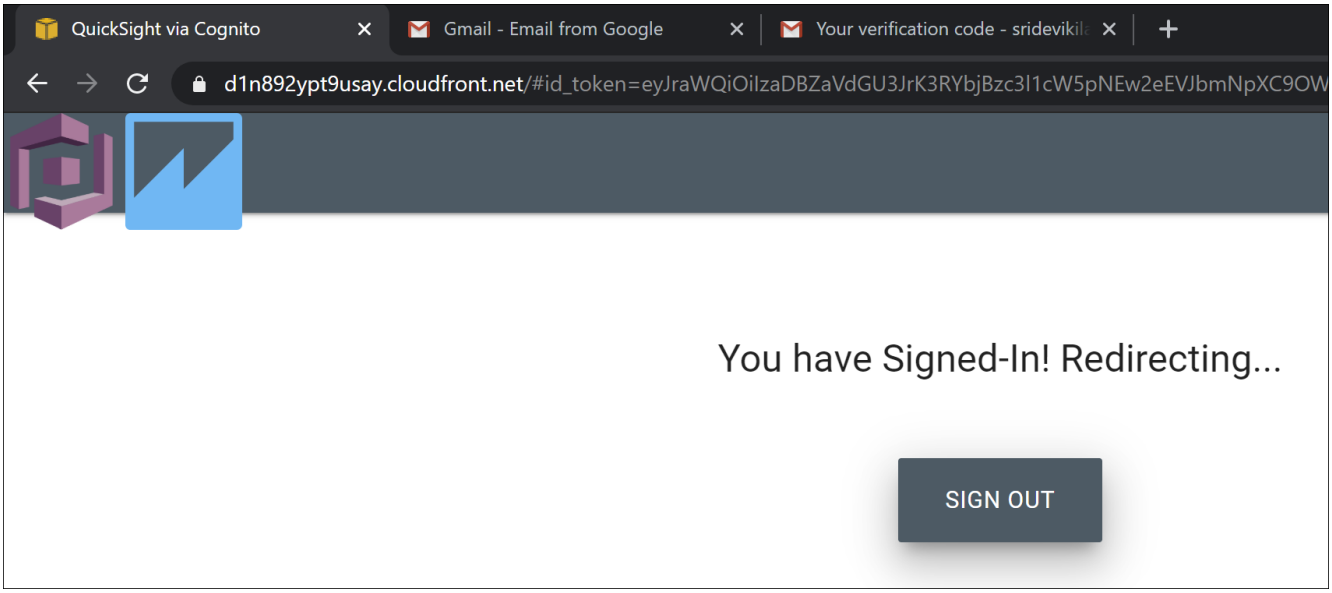
When the user accesses the CloudFront URL, it provides the sign in and sign up options as follows for validating the user before allowing to access Quick sight visuals.

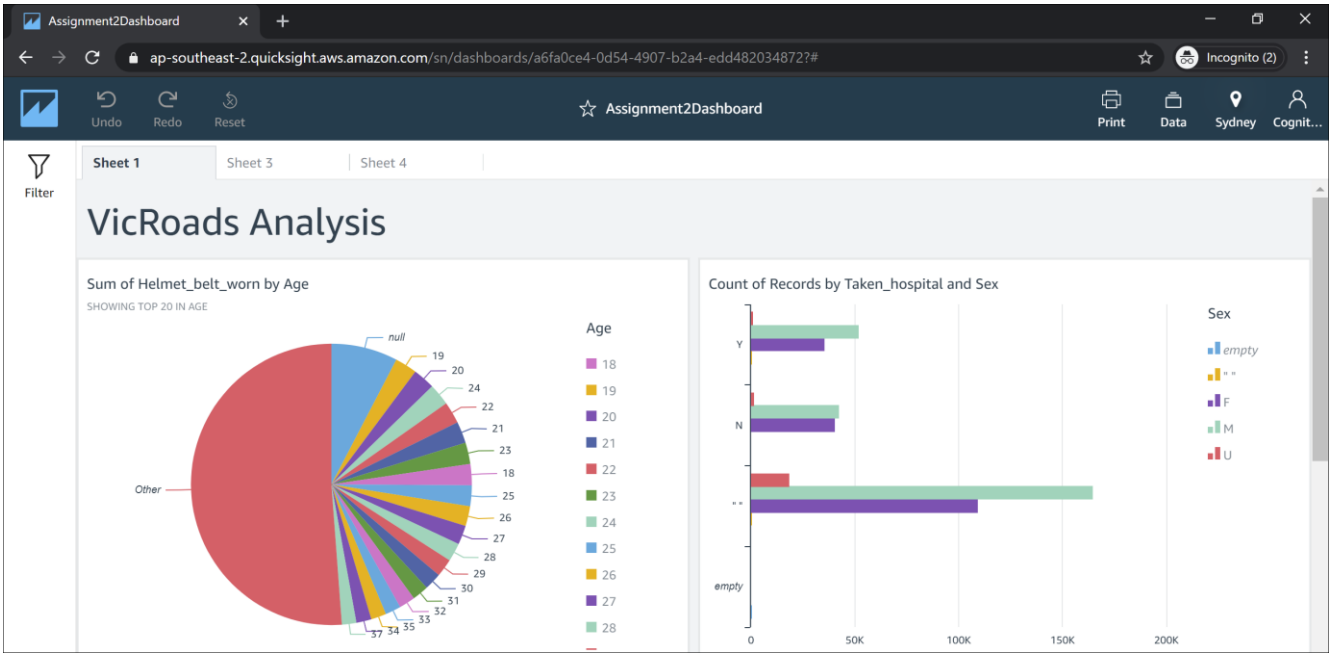



Signin
sri-divya-qs.auth.ap-southeast-2.amazoncognito.com/signup?redirect_uri=https%3A%2F%2Fd1n892ypt9usay.cl
Sign up with a new account
Username
s3778317
Email
sridevikilari95@gmail.com
Password
✓ Password must contain a lower case letter
✓ Password must contain an upper case letter
✓ Password must contain a number
✓ Password must contain at least 8 characters
Sign up
Already have an account? Sign in


sri-divya-qs.auth.ap-southeast-2.amazoncognito.com/confirm?redirect_uri=https%3A%2F%2Fd1n892ypt9usay.c
We have sent a code by email to s***@g***.com.
Enter it below to confirm your account.
Verification Code
Confirm Account
Didn't receive a code? Resend it

Upon successful validation, the user has now redirected to QuickSight visuals as follows:

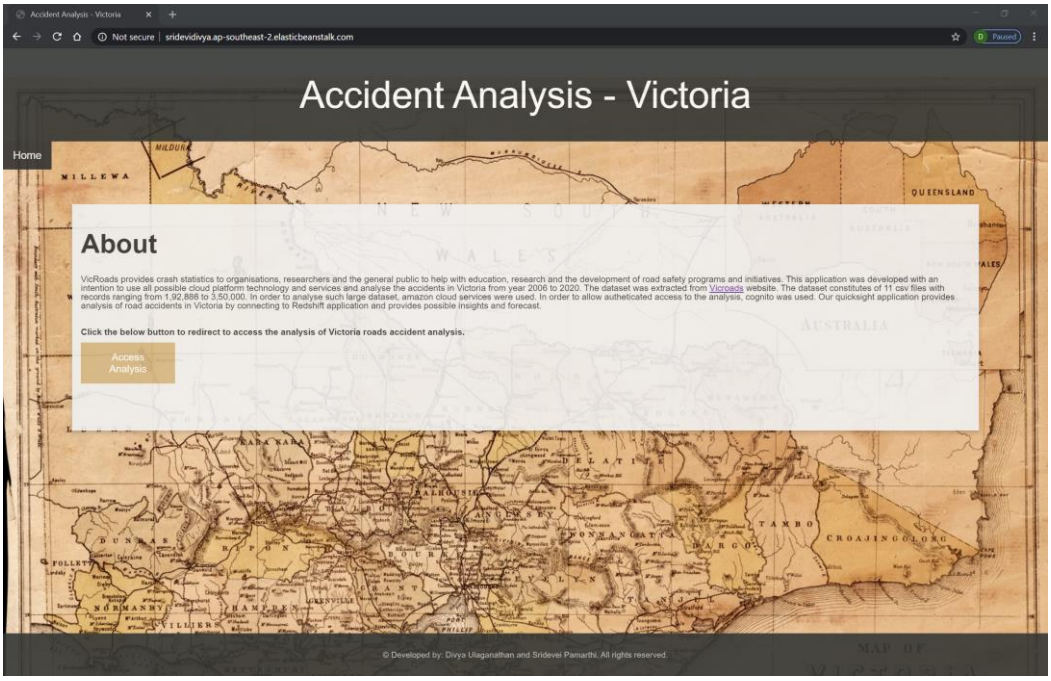


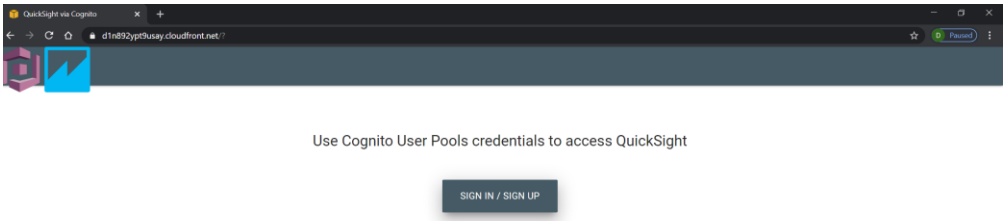


A small user manual

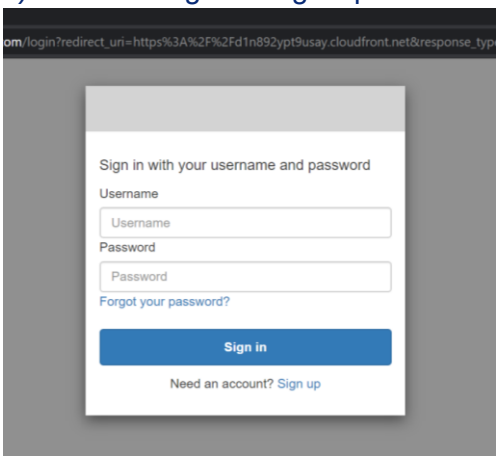
a) Navigate to the link :
http://sridevidivya.ap-southeast-2.elasticbeanstalk.com/

b) Click on Access analysis button to navigate to Cognito login.

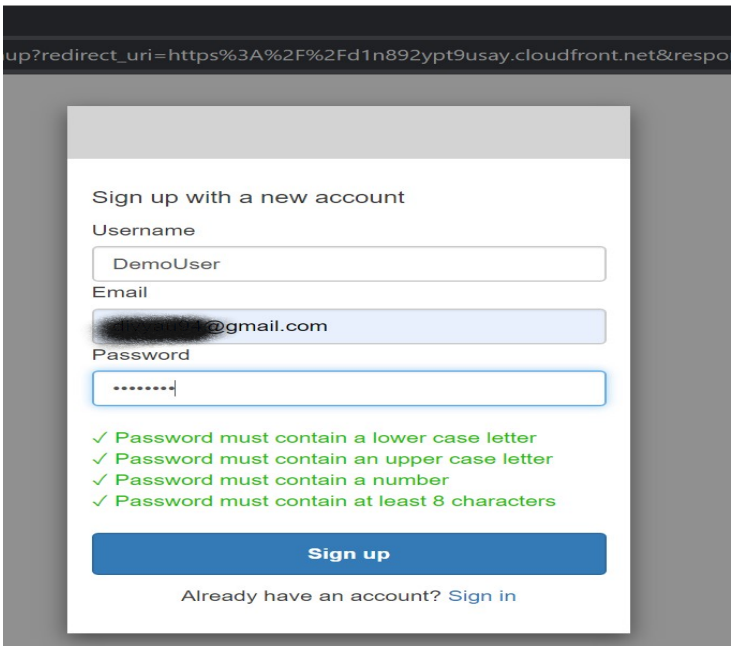




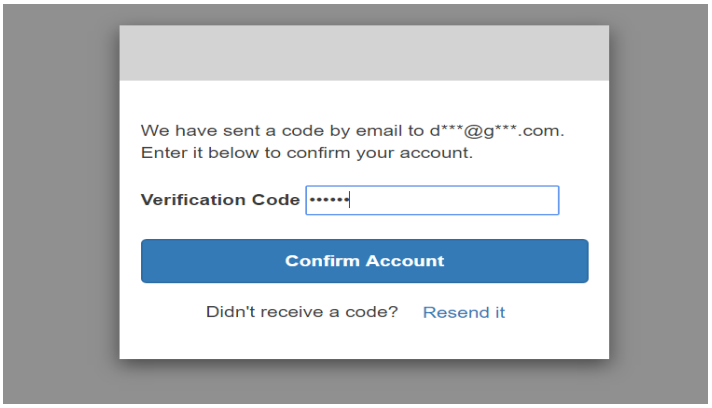
c) Click on “Sign In/ Sign Up” button. You will receive a popup like below.



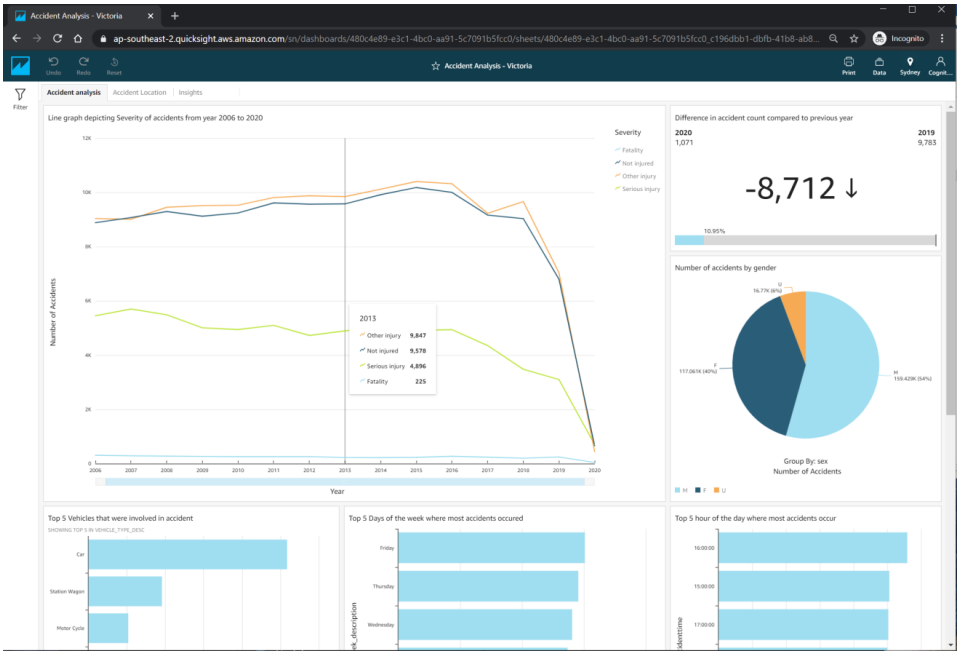
d) If you are a new user to this application enter your details to sign-in. Click Sign up and enter you details like below to Sign up.



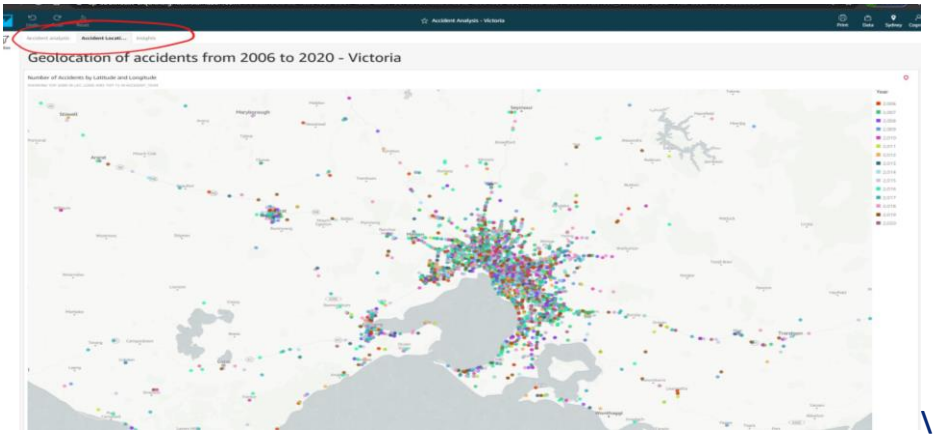
e) On Sign up you will receive a verification code on your respective email to confirm your account



f. On confirming, you will be redirected to the Quicksight application to see the analysis of accidents in Victoria.



g. Interact with the charts by hovering on them. Scroll down to see additional charts. Click on the tabs on to left corner of the application to view additional visuals and insights.



References

[1] Vicroadsopendatastorehouse.vicroads.vic.gov.au. 2020. [online] Available at: <https://vicroadsopendatastorehouse.vicroads.vic.gov.au/opendata/Road_Safety/ACCIDENT.zip> [Accessed 24 May 2020].

[2] Amazon Web Services. 2020. *Embed Interactive Dashboards In Your Application With Amazon Quicksight | Amazon Web Services*. [online] Available at: <https://aws.amazon.com/blogs/big-data/embed-interactive-dashboards-in-your-application-with-amazon-quicksight/> [Accessed 24 May 2020].

[3] Docs.aws.amazon.com. 2020. *Configuring Your Elastic Beanstalk Environment's Load Balancer To Terminate HTTPS - AWS Elastic Beanstalk*. [online] Available at: <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https-elb.html> [Accessed 24 May 2020].

[4] Data Visualization tool for vehicular crash data across the State of Victoria (Australia). [online] Available at: <https://github.com/ajdeziel/crash-heat> [Accessed 24 May 2020].

[5]. Use Amazon QuickSight Federated Single Sign-On with Amazon Cognito User Pools. [online] Available at: <https://aws.amazon.com/blogs/big-data/use-amazon-quicksight-federated-single-sign-on-with-amazon-cognito-user-pools/> [Accessed 24 May 2020].

[6]. Udemy course: Mastering AWS Glue, QuickSight, Athena, & Redshift Spectrum by Siddharth Mehta. [online] Available at: <https://www.udemy.com/course/aws-serverless-glue-redshift-spectrum-athena-quicksight-training/> [Accessed 24 May 2020].

Appendix

Views schema:

table_name	column_name	ordinal_position	data_type
v_accident	node_id	1	bigint
v_accident	route_no	2	bigint
v_accident	chainage_seq	3	bigint
v_accident	route_link_no	4	bigint
v_accident	chainage	5	bigint
v_final_accident	accident_year	1	integer
v_final_accident	accidentdate	2	character varying
v_final_accident	severity	3	bigint
v_final_accident	accident_no	4	character varying
v_final_accident	age_group	5	character varying
v_final_accident	sex	6	character varying
v_final_accident	inj_level_desc	7	character varying
v_final_accident	road_user_type_desc	8	character varying
v_final_accident	accidenttime	9	character varying
v_final_accident	day_week_description	10	character varying
v_final_accident	vehicle_type_desc	11	character varying
v_final_accident	lat	12	double precision
v_final_accident	long	13	double precision
v_final_accident	postcode_no	14	bigint
v_person_accident_analysis	inj_level_desc	1	character varying
v_person_accident_analysis	age_group	2	character varying
v_person_accident_analysis	accident_type	3	bigint
v_person_accident_analysis	accident_type_desc	4	character varying
v_person_accident_analysis	sex	5	character varying

v_person_accident_analysis	accident_count	6	bigint

Tables schema:

table_name	column_name	ordinal_position	data_type
accident_chainage_csv	node_id	1	bigint
accident_chainage_csv	route_no	2	bigint
accident_chainage_csv	chainage_seq	3	bigint
accident_chainage_csv	route_link_no	4	bigint
accident_chainage_csv	chainage	5	bigint
accident_csv	accident_no	1	character varying
accident_csv	accidentdate	2	character varying
accident_csv	accidenttime	3	character varying
accident_csv	accident_type	4	bigint
accident_csv	accident_type_desc	5	character varying
accident_csv	day_of_week	6	bigint
accident_csv	day_week_description	7	character varying
accident_csv	dca_code	8	bigint
accident_csv	dca_description	9	character varying
accident_csv	directory	10	character varying
accident_csv	edition	11	bigint
accident_csv	page	12	character varying
accident_csv	grid_reference_x	13	character varying
accident_csv	grid_reference_y	14	bigint
accident_csv	light_condition	15	bigint
accident_csv	light_condition_desc	16	character varying
accident_csv	node_id	17	bigint
accident_csv	no_of_vehicles	18	bigint
accident_csv	no_persons	19	bigint
accident_csv	no_persons_inj_2	20	bigint
accident_csv	no_persons_inj_3	21	bigint
accident_csv	no_persons_killed	22	bigint
accident_csv	no_persons_not_inj	23	bigint
accident_csv	police_attend	24	bigint
accident_csv	road_geometry	25	bigint
accident_csv	road geometry desc	26	character varying
accident_csv	severity	27	bigint
accident_csv	speed_zone	28	bigint
accident_event_csv	accident_no	1	character varying
accident_event_csv	event_seq_no	2	bigint
accident_event_csv	event_type	3	character varying

accident_event_csv	event_type_desc	4	character varying
accident_event_csv	vehicle_1_id	5	character varying
accident_event_csv	vehicle_1_coll_pt	6	character varying
accident_event_csv	vehicle_1_coll_pt_desc	7	character varying
accident_event_csv	vehicle_2_id	8	character varying
accident_event_csv	vehicle_2_coll_pt	9	character varying
accident_event_csv	vehicle_2_coll_pt_desc	10	character varying
accident_event_csv	person_id	11	bigint
accident_event_csv	object_type	12	bigint
accident_event_csv	object_type_desc	13	character varying
accident_location_csv	accident_no	1	character varying
accident_location_csv	node_id	2	bigint
accident_location_csv	road_route_1	3	bigint
accident_location_csv	road_name	4	character varying
accident_location_csv	road_type	5	character varying
accident_location_csv	road_name_int	6	character varying
accident_location_csv	road_type_int	7	character varying
accident_location_csv	distance_location	8	bigint
accident_location_csv	direction_location	9	character varying
accident_location_csv	nearest_km_post	10	character varying
accident_location_csv	off_road_location	11	character varying
atmospheric_cond_csv	accident_no	1	character varying
atmospheric_cond_csv	atmosph_cond	2	bigint
atmospheric_cond_csv	atmosph_cond_seq	3	bigint
atmospheric_cond_csv	atmosph_cond_desc	4	character varying
node_csv	accident_no	1	character varying
node_csv	node_id	2	bigint
node_csv	node_type	3	character varying
node_csv	amg_x	4	double precision
node_csv	amg_y	5	double precision
node_csv	lga_name	6	character varying
node_csv	lga_name_all	7	character varying
node_csv	region_name	8	character varying
node_csv	deg_urban_name	9	character varying
node_csv	lat	10	double precision
node_csv	long	11	double precision
node_csv	postcode_no	12	bigint
node_id_complex_int_id_csv	accident_no	1	character varying
node_id_complex_int_id_csv	node_id	2	bigint
node_id_complex_int_id_csv	complex_int_no	3	bigint
person_csv	accident_no	1	character varying
person_csv	person_id	2	character varying
person_csv	vehicle_id	3	character varying
person_csv	sex	4	character varying
person_csv	age	5	bigint
person_csv	age_group	6	character varying

person_csv	inj_level	7	bigint
person_csv	inj_level_desc	8	character varying
person_csv	seating_position	9	character varying
person_csv	helmet_belt_worn	10	bigint
person_csv	road_user_type	11	bigint
person_csv	road_user_type_desc	12	character varying
person_csv	licence_state	13	character varying
person_csv	pedest_movement	14	bigint
person_csv	postcode	15	bigint
person_csv	taken_hospital	16	character varying
person_csv	ejected_code	17	bigint
road_surface_cond_csv	accident_no	1	character varying
road_surface_cond_csv	surface_cond	2	bigint
road_surface_cond_csv	surface_cond_desc	3	character varying
road_surface_cond_csv	surface_cond_seq	4	bigint
subdca_csv	accident_no	1	character varying
subdca_csv	sub_dca_code	2	character varying
subdca_csv	sub_dca_seq	3	bigint
subdca_csv	sub_dca_code_desc	4	character varying
vehicle_csv	accident_no	1	character varying
vehicle_csv	vehicle_id	2	character varying
vehicle_csv	vehicle_year_manuf	3	bigint
vehicle_csv	vehicle_dca_code	4	bigint
vehicle_csv	initial_direction	5	character varying
vehicle_csv	road_surface_type	6	bigint
vehicle_csv	road_surface_type_desc	7	character varying
vehicle_csv	reg_state	8	character varying
vehicle_csv	vehicle_body_style	9	character varying
vehicle_csv	vehicle_make	10	character varying
vehicle_csv	vehicle_model	11	character varying
vehicle_csv	vehicle_power	12	character varying
vehicle_csv	vehicle_type	13	bigint
vehicle_csv	vehicle_type_desc	14	character varying
vehicle_csv	vehicle_weight	15	bigint
vehicle_csv	construction_type	16	character varying
vehicle_csv	fuel_type	17	character varying
vehicle_csv	no_of_wheels	18	bigint
vehicle_csv	no_of_cylinders	19	bigint
vehicle_csv	seating_capacity	20	bigint
vehicle_csv	tare_weight	21	bigint
vehicle_csv	total_no_occupants	22	bigint
vehicle_csv	carry_capacity	23	bigint
vehicle_csv	cubic_capacity	24	character varying
vehicle_csv	final_direction	25	character varying
vehicle_csv	driver_intent	26	bigint
vehicle_csv	vehicle_movement	27	bigint

vehicle_csv	trailer_type	28	character varying
vehicle_csv	vehicle_colour_1	29	character varying
vehicle_csv	vehicle_colour_2	30	character varying
vehicle_csv	caught_fire	31	bigint
vehicle_csv	initial_impact	32	character varying
vehicle_csv	lamps	33	bigint
vehicle_csv	level_of_damage	34	bigint
vehicle_csv	owner_postcode	35	bigint
vehicle_csv	towed_away_flag	36	bigint
vehicle_csv	traffic_control	37	bigint
vehicle_csv	traffic_control_desc	38	character varying