

Cloud Computing Assignment 2

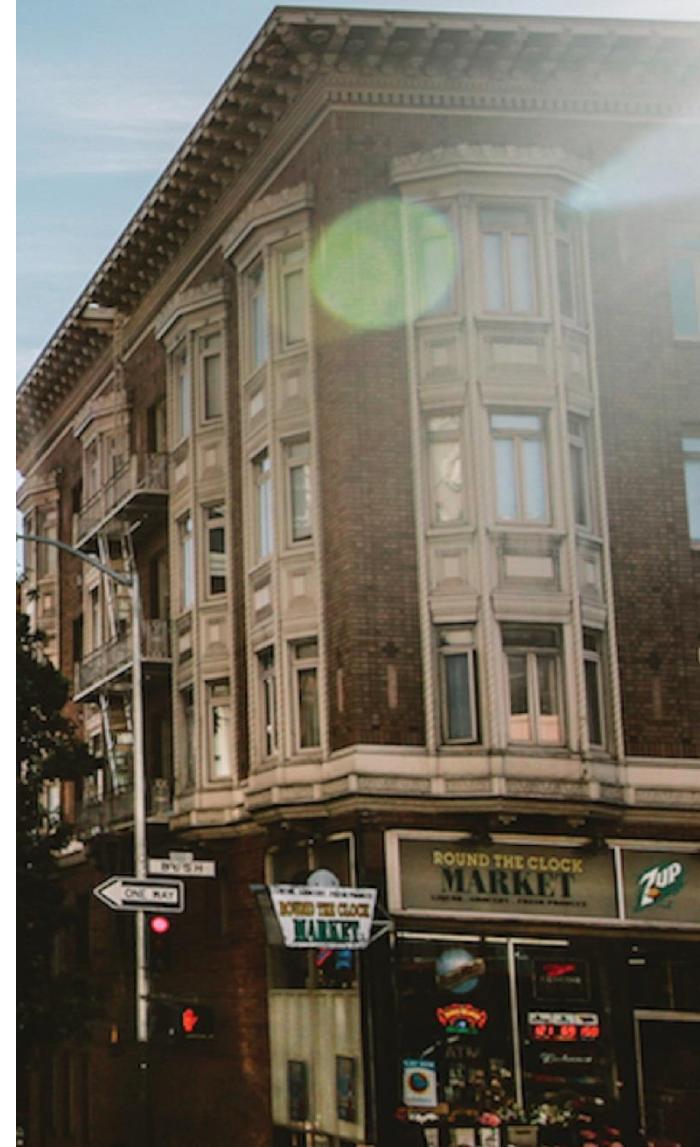
Victoria Road Crash Statistics

MAY 24

COSC2640 – Group 28

Authored by:

- Sridevi Pamarthi (s3778317)
- Divya Ulaganathan (s3759465)



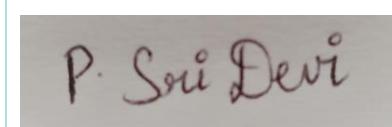
Contents

Signed Contribution Agreement	4
Links	4
Summary	4
Introduction	5
Related work	7
Difficulties faced	7
Software Design/ Architecture	7
Implementation	10
1. Setup S3 buckets:	10
2. Setup AWS Lake formation	11
3. Setup AWS Glue	11
a) Create Crawler	11
b) Create Glue ETL jobs.....	12
c) Validate the data using Athena.....	20
4. Setup Redshift	21
a) Create the Cluster:	21
b) Configure VPC	24
c) Create connection in AWS Glue	26
d) Create Glue job to load data to Redshift.....	26
e) Verify the data in Redshift.....	29
f) Create transformation views.....	30
5. Setup Quick Sight	31
a) VPC connection to access Redshift	31
b) Create a data set from Redshift to build visual	31
6. Set up Elastic Beanstalk	32
a) Register a domain (Route 53)	32
b) Request SSL certificate (Certification Manager):.....	33

c) Create an EBS environment	33
7. QuickSight Sign-On with Cognito User Pools	37
a) Run Cloud Formation stack CognitoQuickSight	37
b) Upload content into S3 static website	37
c) Cognito Configuration:.....	38
d) run CloudFront URL.....	38
A small user manual	41
References.....	43
Appendix	44

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
<i>By signing below, I certify all information is true and correct to the best of my knowledge.</i>	<i>By signing below, I certify all information is true and correct to the best of my knowledge.</i>
Signature: Divya Ulaganathan Date: 24/05/2020	Signature: Sridevi Pamarthi Date: 24/05/2020
 Divya Ulaganathan Miss	

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.

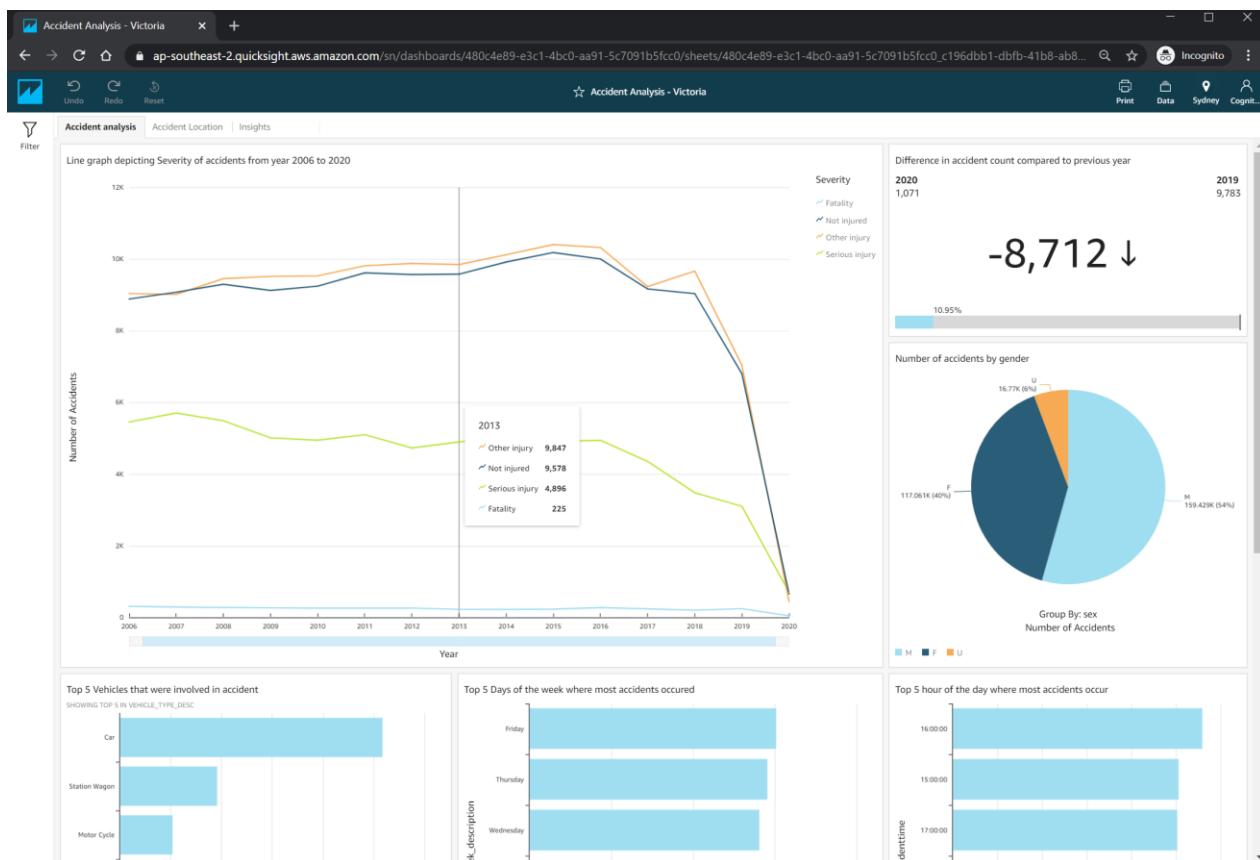
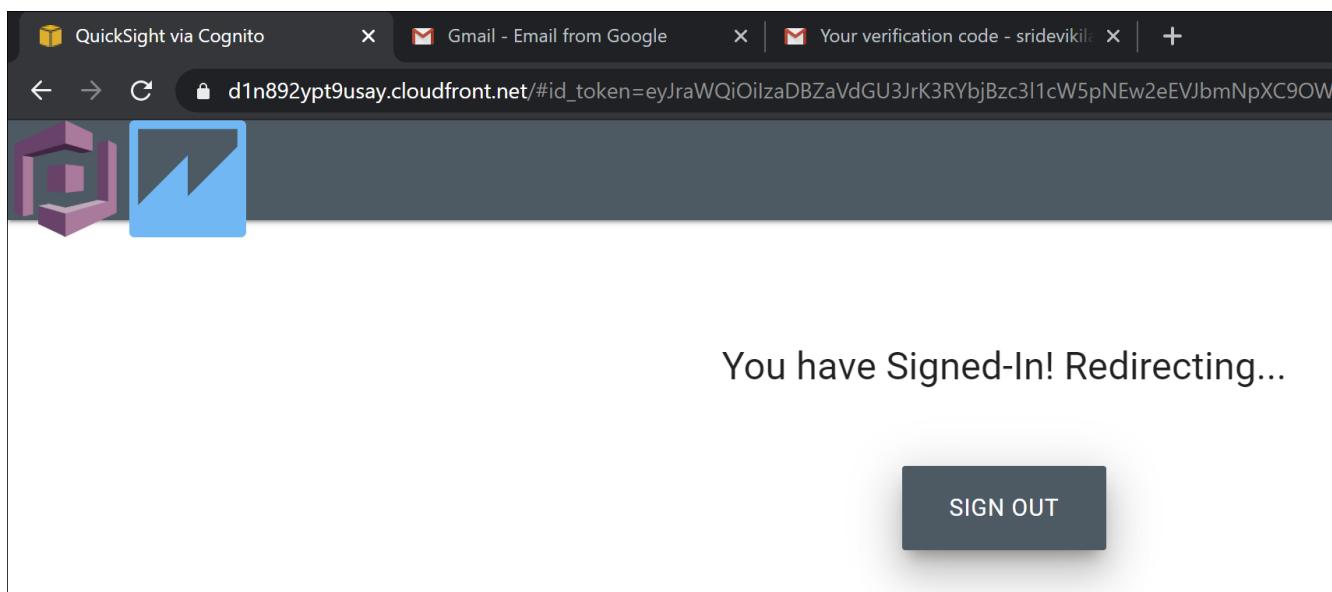
As per design, the CloudFront and Cognito services are used to handle the public access to this application. The user should sign up and authenticate the identity first, only then s/he should be able to access the statistics and insights published in QuickSight application related to road accidents of 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:



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.

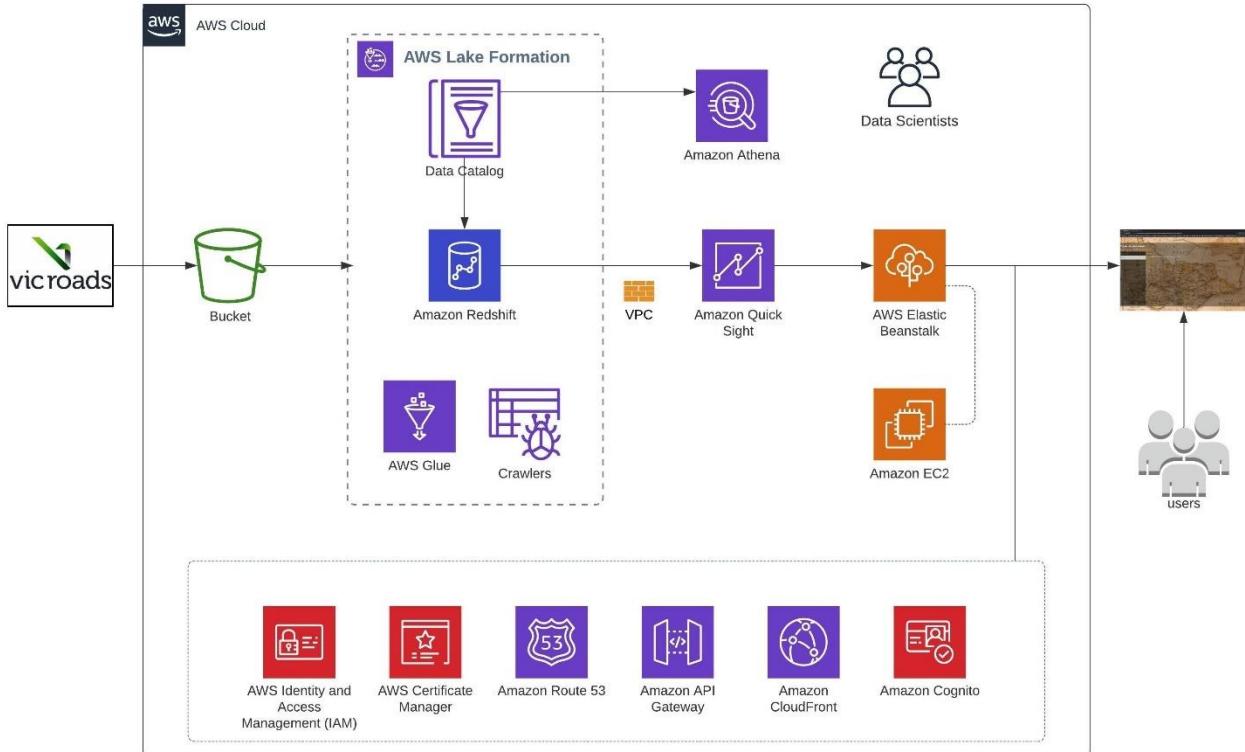
We initially thought of embedding the QuickSight dashboards directly in the website, however, there is not enough information available in the internet to get the Embed URL of the dashboard using a PHP application.

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 retrieve 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

The screenshot shows the AWS Lake Formation interface. In the top navigation bar, there are links for 'Source Groups' (with a dropdown arrow), a bell icon, user 'sridevipamarthi' (with a dropdown arrow), region 'Sydney' (with a dropdown arrow), and 'Support' (with a dropdown arrow). Below the navigation, the path 'AWS Lake Formation > Data lake locations' is shown. The main area is titled 'Data lake locations (0/1)' with a search bar containing 'Filter data lake storage'. There is a table with columns: 'Amazon S3 path', 'IAM role', and 'Last modified'. One row is listed: 's3://assignment2-database-bucket' (with a blue link icon), 'AWSServiceRoleForLakeFor...', and 'Sun, May 24, 2020, 4:14 AM UTC'. At the top right of the table are buttons for 'Actions' (with a dropdown arrow) and 'Register location' (in orange).

b) Create the Database 'vicroads' in AWS Lake

The screenshot shows the AWS Lake Formation interface. In the top navigation bar, there are links for 'Source Groups' (with a dropdown arrow), a bell icon, user 'sridevipamarthi' (with a dropdown arrow), region 'Sydney' (with a dropdown arrow), and 'Support' (with a dropdown arrow). Below the navigation, the path 'AWS Lake Formation > Databases > vicroads' is shown. The main area shows a database named 'vicroads' with buttons for 'Actions' (with a dropdown arrow), 'View Tables', 'Edit', and 'Delete'. Under the heading 'Database details', there are two columns of information: 'Name' (vicroads) and 'Amazon S3 path' (s3://assignment2-database-bucket with a blue link icon). Below these, there are sections for 'Description' (empty) and 'Default permissions for newly created tables' (checkbox checked, 'Use only IAM access control for new tables in this database').

3. Setup AWS Glue

a) Create Crawler

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

Edit crawler

- Crawler info
assignment2-crawler
- Crawler source type
Data stores
- Data store
S3: s3://assignment...
- IAM Role
arn:aws:iam::9098825
75885:role/aws-
assignment2-glue
- Schedule
Run on demand
- Output

Crawler info

Name: assignment2-crawler
Tags: -

Data stores

Data store: S3
Include path: s3://assignment2-data-bucket
Connection
Exclude patterns

Edit crawler

- Crawler info
assignment2-crawler
- Crawler source type
Data stores
- Data store
S3: s3://assignment...
- IAM Role
arn:aws:iam::9098825
75885:role/aws-
assignment2-glue
- Schedule
Run on demand
- Output
vicroads
- Review all steps

IAM role

IAM role: arn:aws:iam::909882575885:role/aws-
assignment2-glue

Schedule

Schedule: Run on demand

Output

Database: vicroads
Prefix added to tables (optional)
Create a single schema for each S3 path: false
▶ Configuration options

b) Create Glue ETL jobs

- i. Create Glue ETL job to retrieve data from VicRoads

Name	job0-loadDataFromVicRoads	Python lib path	-
IAM role	aws-assignment2-glue	Jar lib path	-
Type	Python shell	Other lib path	-
Python version	3	Job parameters	-
Script location	s3://assignment2-jobs-bucket/job0-loadDataFromVicRoads.py	Non-overrideable	-
Job bookmark	Disable	Job parameters	-
Job metrics	Disable	Connections	-
Continuous logging	Disable	Maximum capacity	0.0625
Server-side encryption	Disabled	Job timeout (minutes)	2880
		Delay notification threshold (minutes)	-

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 = "/")

```

```

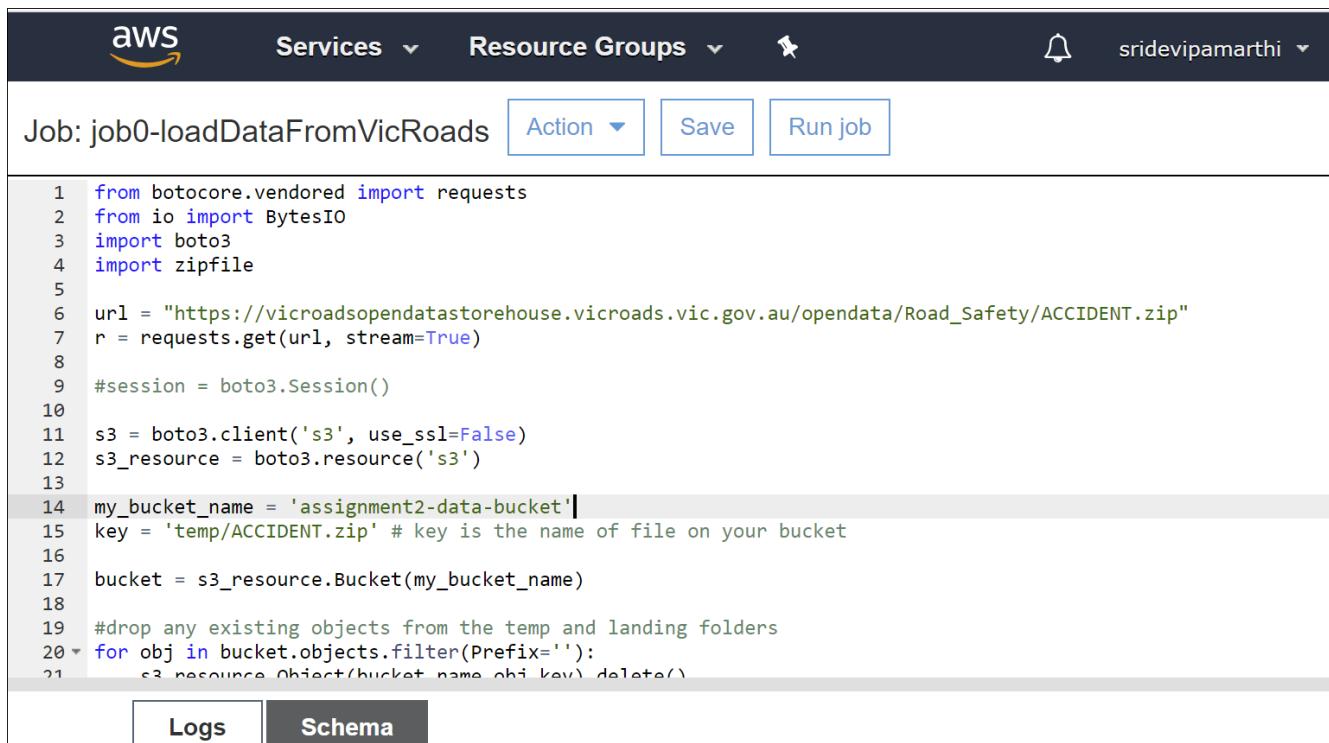
file_list = []
for key in zipped_keys['Contents']:
    file_list.append(key['Key'])

#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.



The screenshot shows the AWS Lambda function editor interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and a user profile 'sridevipamarthi'. Below the navigation bar, the job name is 'Job: job0-loadDataFromVicRoads'. There are three buttons: 'Action', 'Save', and 'Run job'. The main area contains the Python code for the function:

```

1  from botocore.vendored import requests
2  from io import BytesIO
3  import boto3
4  import zipfile
5
6  url = "https://vicroadsopendatastorehouse.vicroads.vic.gov.au/opendata/Road_Safety/ACCIDENT.zip"
7  r = requests.get(url, stream=True)
8
9  #session = boto3.Session()
10
11 s3 = boto3.client('s3', use_ssl=False)
12 s3_resource = boto3.resource('s3')
13
14 my_bucket_name = 'assignment2-data-bucket'
15 key = 'temp/ACCIDENT.zip' # key is the name of file on your bucket
16
17 bucket = s3_resource.Bucket(my_bucket_name)
18
19 #drop any existing objects from the temp and landing folders
20 for obj in bucket.objects.filter(Prefix=''):
21     s3_resource.Object(bucket_name=obj.key).delete()

```

At the bottom, there are two tabs: 'Logs' and 'Schema', with 'Logs' being the active tab.

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

The screenshot shows the AWS S3 console for the 'assignment2-data-bucket'. The top navigation bar includes 'Services', 'Resource Groups', and a user dropdown for 'sridevipamarthi'. The main content area displays the bucket's name, 'assignment2-data-bucket'. Below it are tabs for 'Overview', 'Properties' (which is selected), 'Permissions', 'Management', and 'Access points'. A search bar at the top says 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are buttons for 'Upload', 'Create folder', 'Download', and 'Actions'. To the right, it shows the region as 'Asia Pacific (Sydney)' and 'Viewing 1 to 12' results. The table lists three objects:

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	ACCIDENT.csv	May 24, 2020 3:39:24 PM GMT+1000	49.9 MB	Standard
<input type="checkbox"/>	ACCIDENT_CHAINAGE.csv	May 24, 2020 3:39:29 PM GMT+1000	6.4 MB	Standard
<input type="checkbox"/>	ACCIDENT_EVENT.csv	May 24, 2020 3:39:25 PM GMT+1000	32.7 MB	Standard

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

The screenshot shows the AWS Glue console under the 'Crawlers' section. The left sidebar has a navigation menu with options: AWS Glue, Data catalog, Databases, Tables, Connections, Crawlers (which is selected and highlighted in orange), and Classifiers. The main content area is titled 'Crawlers' and contains the following information: 'A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then adds the data to your data catalog.' Below this are buttons for 'Add crawler', 'Run crawler', 'Action', and a search bar 'Filter by tags and attributes'. A table lists the crawlers:

<input type="checkbox"/>	Name	Schedule	Status	Logs	Last runtime	Median runtime
<input checked="" type="checkbox"/>	assignment2-crawler		Ready	Logs	52 secs	52 secs

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

The screenshot shows the AWS Glue Data Catalog Tables page. On the left sidebar, under the 'Tables' section, there is a list of tables: accident_chainage_csv, accident_csv, accident_event_csv, accident_location_csv, atmospheric_cond_csv, node_csv, node_id_complex_int_id_csv, person_csv, and road_surface_cond_csv. Each table entry includes its name, database (vicroads), location (S3 path), file type (csv), and last modified date (16 Ma).

- 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 required.

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

The screenshot shows the 'Add job' configuration page. On the left, there is a sidebar with tabs: Job properties (selected), Data source, Transform type, Data target, and Schema. The main form fields are:

- Name:** job-load-accident-data
- IAM role:** aws-assignment2-glue
- 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

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)

Add job

Job properties Data source Transform type Data target Schema

job-load-accident-data

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
Change schema

Data target
accident_csv

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
accident_csv

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 default. AWS Glue generates your script with the defined mappings.

Source			Target	
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_desc	string	accident_type_desc	accident_type_desc	string

```

1  import sys
2  from awsglue.transforms import *
3  from awsglue.utils import getResolvedOptions
4  from pyspark.context import SparkContext
5  from awsglue.context import GlueContext
6  from awsglue.job import Job
7
8  ## @params: [JOB_NAME]
9  args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16 ## @type: DataSource
17 ## @args: [database = "vicroads", table_name = "accident_csv", transformation_ctx = "datasource0"]
18 ## @return: datasource0
19 ## @inputs: []
20 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "vicroads", table_name = "accident_csv", transformation_ctx = "datasource0")
21 ## @type: ApplyMapping
22 ## @args: [mapping = [{"accident_no": "string", "accident_no": "string"}, {"accidentdate": "string", "accidentdate": "string"}, {"accidentdate": "string", "accidentdate": "string"}], @return: applymapping1
23 ## @inputs: [frame = datasource0]
24 applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [{"accident_no": "string", "accident_no": "string"}, {"accidentdate": "string", "accidentdate": "string"}, {"accidenttime": "string", "accidenttime": "string"}, {"accident_type": "string", "accident_type": "string"}, {"accident_type_desc": "string", "accident_type_desc": "string"}, {"day_of_week": "string", "day_of_week": "string"}], @return: selectfields2
25 ## @type: SelectFields
26 ## @args: [paths = ["accident_no", "accidentdate", "accidenttime", "accident_type", "accident_type_desc", "day_of_week"]]
27 ## @return: selectfields2
28 ## @inputs: [frame = applymapping1]
29 ## @outputs: [frame = selectfields2]
30

```

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

The screenshot shows the AWS Glue Jobs console. On the left, there's a sidebar with navigation links for AWS Glue, Data catalog, Databases, Tables, Connections, Crawlers, Classifiers, Settings, ETL, Workflows, and Jobs. The 'Jobs' link is highlighted. The main area displays a table of jobs with columns: Name, Type, ETL language, Script location, and Last modified. There are 10 entries listed, all using Spark as the ETL type and Python as the language.

Name	Type	ETL language	Script location	Last modified
job0-loadDataFromVicRoads	Spark	python	s3://assignme...	17 May 2020 5:38 PM ...
job1-accident	Spark	python	s3://assignme...	16 May 2020 4:13 PM ...
job10-subdca	Spark	python	s3://assignme...	16 May 2020 4:34 PM ...
job11-vehicle	Spark	python	s3://assignme...	16 May 2020 4:36 PM ...
job2-accident-chainage	Spark	python	s3://assignme...	16 May 2020 4:16 PM ...
job3-accident-event	Spark	python	s3://assignme...	16 May 2020 4:18 PM ...
job4-accident-location	Spark	python	s3://assignme...	16 May 2020 4:21 PM ...
job5-atmospheric-cond	Spark	python	s3://assignme...	16 May 2020 4:23 PM ...
job6-node	Spark	python	s3://assignme...	16 May 2020 4:25 PM ...
job7-node-complex	Spark	python	s3://assignme...	16 May 2020 4:27 PM ...

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.

The screenshot shows the AWS Athena Query Editor. On the left, there's a sidebar with Database (set to vicroads), Tables (12), and Views (0). The main area has tabs for Athena, Query Editor (which is selected), Saved Queries, History, AWS Glue Data Catalog, and Workgroup : primary. The Query Editor shows a 'New query 1' tab with the following SQL code:

```

1 SELECT *
2 FROM "vicroads"."accident_csv"
3 limit 10;

```

Below the code, there are buttons for Run query, Save as, and Create. A status message indicates a run time of 1.3 seconds and data scanned of 150.82 KB. The Results section displays the output of the query:

accident_no	accidentdate	accidenttime	accident_type	accident_type_desc	day_of_week	day_week_desc
T2006000010	13/01/2006	"12:42:00 "	1	"Collision with vehicle"	6	Friday
T2006000018	13/01/2006	"19:10:00 "	1	"Collision with vehicle"	6	Friday
T2006000022	14/01/2006	"12:10:00 "	7	"Fall from or in moving vehicle"	7	Saturday
T2006000023	14/01/2006	"11:49:00 "	1	"Collision with vehicle"	7	Saturday

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.

The screenshot shows the 'Create cluster' configuration page in the AWS Amazon Redshift service. The left sidebar has a 'CLUSTERS' button highlighted in orange. The main area shows 'Cluster configuration' with a 'Cluster identifier' field containing 'redshift-cluster-2'. Under 'Node type', there are two sections: 'RA3' (Recommended) and 'DC2'. The 'RA3' section lists 'ra3.4xlarge' and 'ra3.16xlarge' node types. The 'DC2' section lists 'dc2.large' and 'dc2.8xlarge' node types, with 'dc2.large' selected. Both sections provide details like price per node/hour and storage capacity.

Node Type	Storage	Price (\$/node/hour)
ra3.4xlarge	Managed storage: up to 64 TB/node	\$3.909/node/hour \$0.026/GB/month
ra3.16xlarge	Managed storage: up to 256 TB/node	\$15.636/node/hour \$0.026/GB/month
dc2.large	Storage: 160 GB/node	\$0.33/node/hour
dc2.8xlarge	Storage: 2.6 TB/node	\$6.40/node/hour

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

Events

Services ▾ Resource Groups ▾

WHAT'S NEW

ra3.4xlarge
12 vCPU (gen 3)

dc2.large
2 vCPU (gen 2)

► Show legacy dense storage node types

Nodes
Enter the number of nodes that you need.
1 Range (1-32)

Configuration summary
dc2.large | 1 node

\$237.60/month Estimated on-demand compute price Save more than 60% of your costs by purchasing reserved nodes. Learn more	160 GB Total compressed storage The total storage capacity for the cluster if you deploy the number of nodes that you chose.
--	---

Configure the database name, port, master credentials:

Database configurations

Database name (optional)
Specify a database name to create an additional database.
vicroadsdw

The name must be 1-64 alphanumeric characters (lowercase only), and it can't be a [reserved word](#).

Database port (optional)
Port number where the database accepts inbound connections. You can't change the port after the cluster has been created.
5439

The port must be numeric (1150-65535).

Master user name
Enter a login ID for the master user of your DB instance.
awsuser

The name must be 1-128 alphanumeric characters, and it can't be a [reserved word](#).

Master user password
.....

Show password

- The value must be 8-64 characters.
- The value must contain at least one uppercase letter.
- The value must contain at least one lowercase letter.
- The value must contain at least one number.
- The master password can only contain ASCII characters (ASCII codes 33-126), except '(single quotation mark)', '' (double quotation mark)', '/', '\, or @.'

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.

The master password can only contain ASCII characters (ASCII codes 33-126), except - (single quotation mark), - (double quotation mark), /, \, or @.

Cluster permissions (optional)

Additional configurations Use defaults

These configurations are optional, and default settings have been defined to help you get started with your cluster. Turn off "Use defaults" to modify these settings now.

Network Using default VPC (vpc-33b8b554) and default subnet	Security Using default (sg-919f4de4) cluster security group
Backup Automated snapshots are created about every eight hours or following every 5 GB per node of data changes, whichever comes first.	Configuration Using default.redshift-1.0 parameter group with no database encryption
Maintenance Using current maintenance track	

Create cluster

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.

Amazon Redshift > Clusters > redshift-cluster-1

redshift-cluster-1

Available dc2.large 1 total nodes

Actions Query cluster

Cluster performance | Query monitoring | Maintenance and monitoring | Backup | **Properties** | Schedule

Cluster permissions

Your cluster needs permissions to access other AWS services on your behalf. For the required permissions, add IAM roles with the principal "redshift.amazonaws.com". You can associate up to 10 IAM roles with this cluster. [Learn more](#)

Attach IAM roles

Connection details

Master user name: awsuser [Change master user password](#)

Endpoint: redshift-cluster-1.c1m7dek5Sub7.ap-southeast-2.redshift.amazonaws.com:5439/vicroadsdw [Copy](#)

[View all connection details](#).

The screenshot shows the AWS Redshift console for a cluster named 'redshift-cluster-1'. It displays cluster details, database configurations, network and security settings, and connectivity URLs.

Cluster details:

Cluster identifier	Node type	Number of nodes	Total available storage
redshift-cluster-1	dc2.large	1	160 GB

Database configurations:

Database name	Port	Master user name
vicroadsdw	5439	awsuser

Network and security:

Virtual private cloud (VPC)	Subnet
vpc-33b8b554	default
Availability Zone	Enhanced VPC routing
ap-southeast-2c	Forces cluster traffic through a VPC.
	Disabled
VPC security group	
sg-919f4de4	Specify which instances and devices can connect to the cluster.
Publicly accessible	Edit
No	Allow instances and devices outside the VPC connect to your database through the cluster endpoint

JDBC URL: jdbc:redshift://redshift-cluster-1.c1m7dek5ub7.ap-southeast-2.redshift.amazonaws.com:5439/vicroadsdw

ODBC URL: Driver={Amazon Redshift (x64)}; Server=redshift-cluster-1.c1m7dek5ub7.ap-southeast-2.redshift.amazonaws.com; Database=vicroadsdw; UID=awsuser; Pwd=insert_your_master_user_password_here; Port=5439

SSH ingestion settings: Cluster public key

Node IP addresses:

Node role	Public IP address	Private IP address
Shared	13.238.176.115	172.31.18.89

Tags: Manage tags
No tags are associated with this resource.

b) Configure VPC

Make sure the VPC settings are in place as follows:

The screenshot shows the AWS VPC console with the 'Your VPCs' section selected. It displays a list of VPCs and their details.

VPC Dashboard:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set	Main Route table
vpc-33b8b554	vpc-33b8b554	available	172.31.0....	-	dopt-5d96b93a	rtb-11f76177

VPC: vpc-33b8b554:

Description	CIDR Blocks	Flow Logs	Tags
VPC ID: vpc-33b8b554 State: available IPv4 CIDR: 172.31.0/16 IPv6 CIDR: - DNS resolution: Enabled DNS hostnames: Enabled ClassicLink DNS Support: Disabled Owner: 909882575885			Tenancy: default Default VPC: Yes Classic link: Disabled IPv6 Pool: - Network ACL: acl-130ba075 DHCP options set: dopt-5d96b93a Route table: rtb-11f76177

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

The screenshot shows the AWS VPC Security Groups page. On the left, the navigation pane includes options like Internet Gateways, Egress Only Internet Gateways, DHCP Options Sets, Elastic IPs, Endpoints, Endpoint Services, NAT Gateways, Peering Connections, SECURITY (Network ACLs, Security Groups), and VIRTUAL PRIVATE NETWORK (VPN) (Customer Gateways, Virtual Private Gateways, Site-to-Site VPN Connections). The main content area displays the 'Security Groups (1/1)' section. A search bar at the top allows filtering by security group ID (sg-919f4de4) or name. Below the search bar is a table with columns: Security group ID, Security group name, VPC ID, Description, Owner, and Inbound rules. One row is visible: sg-919f4de4, default, vpc-33b8b554, default VPC security gr..., 909882575885, 2 Permiss. Below the table, tabs for Details, Inbound rules (which is selected), Outbound rules, and Tags are present. Under the Inbound rules tab, a table lists Type (All TCP, All traffic), Protocol (TCP, All), Port range (0 - 65535, All), Source (sg-0baf786052af56f6f (Amazon-QuickSight-access), sg-919f4de4 (default)), and Description - optional (empty). A 'Edit inbound rules' button is located at the top right of this table.

Add following outbound rules:

The screenshot shows the AWS VPC Security Groups page, similar to the previous one but with different selected tabs. The navigation pane is identical. The main content area displays the 'Security Groups (1/1)' section. The 'Outbound rules' tab is selected. A table lists Type (All TCP, All TCP, All traffic), Protocol (TCP, TCP, All), Port range (0 - 65535, 0 - 65535, All), Destination (sg-0baf786052af56f6f (Amazon-QuickSight-access), sg-919f4de4 (default), 0.0.0.0/0), and Description - optional (empty). A 'Edit outbound rules' button is located at the top right of this table.

c) Create connection in AWS Glue

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

Connection properties

Name	Redshift
Type	JDBC
Require SSL connection	false
Description (optional)	-

Connection access

JDBC URL	jdbc:redshift://redshift-cluster-1.c1m7dekk55ub7.ap-southeast-2.redshift.amazonaws.com:5439/vicroadsdw
Username	awsuser
VPC ID	vpc-33b8b554
Subnet	subnet-513ddb19
Security groups	sg-0ba786052af56f6f, sg-919f4de4

Back Finish

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

Connections A connection contains the properties needed to connect to your data.

Testing Redshift access to your data store is in progress. This can take a few moments.

Name	Type	Date created	Last updated
Redshift	JDBC	21 May 2020 1:12 PM UTC+10	21 May 2020 1:28 PM UTC

Redshift connected successfully to your instance.

Name	Type	Date created	Last updated
Redshift	JDBC	21 May 2020 1:12 PM UTC+10	21 May 2020 1:28 PM UTC

d) Create Glue job to load data to Redshift

Follow the configuration given in below screenshots to create the job

Add job

Job properties
 Data source
 Transform type
 Data target
 Schema

Configure the job properties

Name

IAM role

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

Glue version

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

Add job

Job properties
 job-load-accident-data-redshift
 Data source
 Transform type
 Data target
 Schema

Choose a data source

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](#)

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](#)

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

Database name vicroadsdw

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	
Column name	Data type	Map to target	
accident_no	string	accident_no	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
accidentdate	string	accidentdate	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
accidenttime	string	accidenttime	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
accident_type	bigint	accident_type	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
accident_type_desc	string	accident_type_desc	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
day_of_week	bigint	day_of_week	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
day_week_desc	string	day_week_description	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
dca_code	bigint	dca_code	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
dca_description	string	dca_description	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>
directory	string	directory	<input type="button" value="Add column"/> <input type="button" value="Clear"/> <input type="button" value="Reset"/>

Job: job-load-accident-data-redshift was added. Edit and save your Python script.

Action Save Run job Generate diagram Insert template at cursor Source Target Target Location

```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 ## @params: [TempDir, JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['TempDir','JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16 ## @type: DataSource
17 ## @args: [database = "vicroads", table_name = "accident_csv", transformation_ctx = "datasource0"]
18 ## @return: datasource0
19 ## @inputs: []
20 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "vicroads", table_name = "accident_csv", transformation_ctx = "datasource0")
21 ## @type: ApplyMapping
22 ## @args: [mapping = [{"accident_no": "string", "accident_no": "string"}, {"accidentdate": "string", "accidentdate": "string"}], @return: applymapping1
23 ## @inputs: [datasource0]
24 ## @outputs: [frame = datasource0]
25 applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [{"accident_no": "string", "accident_no": "string"}, {"accidentdate": "string", "accidentdate": "string"}])
26 ## @type: ResolveChoice
27 ## @args: [choice = "make_cols", transformation_ctx = "resolvechoice2"]
28 ## @return: resolvechoice2
29

```

Logs Schema

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

Add job Action Filter by tags and attributes Showing: 1 - 11

Name	Type	ETL language	Script location	Last modified
redshift-job01-accident	Spark	python	s3://assignme...	21 May 2020 4:37 PM ...
redshift-job02-accident-chainage	Spark	python	s3://assignme...	21 May 2020 4:41 PM ...
redshift-job03-accident-event	Spark	python	s3://assignme...	21 May 2020 4:45 PM ...
redshift-job04-accident-location	Spark	python	s3://assignme...	21 May 2020 4:47 PM ...
redshift-job05-atmospheric-cond	Spark	python	s3://assignme...	21 May 2020 4:49 PM ...
redshift-job06-node	Spark	python	s3://assignme...	21 May 2020 4:51 PM ...
redshift-job07-node-complex	Spark	python	s3://assignme...	21 May 2020 4:54 PM ...
redshift-job08-person	Spark	python	s3://assignme...	21 May 2020 4:56 PM ...
redshift-job09-road-surface	Spark	python	s3://assignme...	21 May 2020 4:58 PM ...
redshift-job10-subdca	Spark	python	s3://assignme...	21 May 2020 4:59 PM ...
redshift-job11-vehicle	Spark	python	s3://assignme...	21 May 2020 5:01 PM ...

e) Verify the data in Redshift

The screenshot shows the AWS Redshift Data Objects interface. On the left, there's a sidebar with icons for Dashboard, Clusters, Queries, Editor (which is selected), and Config. The main area has tabs for Data objects and Query editor. In the Data objects tab, it says 'Select schema' with 'public' selected. Below that is a search bar for 'Filter tables and views' with 'accident_chainage_csv', 'accident_csv', 'accident_event_csv', and 'accident_location_csv' listed. In the Query editor tab, there's a 'Query 1' section with the following SQL code:

```

1 SELECT TOP 10 *
2 FROM ACCIDENT_CSV
3

```

Below the query editor is a table titled 'Rows returned (10)' with columns: accident_no, accidentdate, accidenttime, accident_type, accident_type_desc, day_of_week, and day_week. The data shows three rows of accident records.

accident_no	accidentdate	accidenttime	accident_type	accident_type_desc	day_of_week	day_week
T2006000010	13/01/2006	12:42:00	1	Collision with vehicle	6	Friday
T2006000022	14/01/2006	12:10:00	7	Fall from or in moving vehicle	7	Saturday
T2006000026	14/01/2006	10:45:00	1	Collision with vehicle	7	Saturday

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.ACCEPT_NO) AS ACCIDENT_COUNT

FROM PERSON_CSV P
JOIN ACCIDENT_CSV A ON P.ACCEPT_NO = A.ACCEPT_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

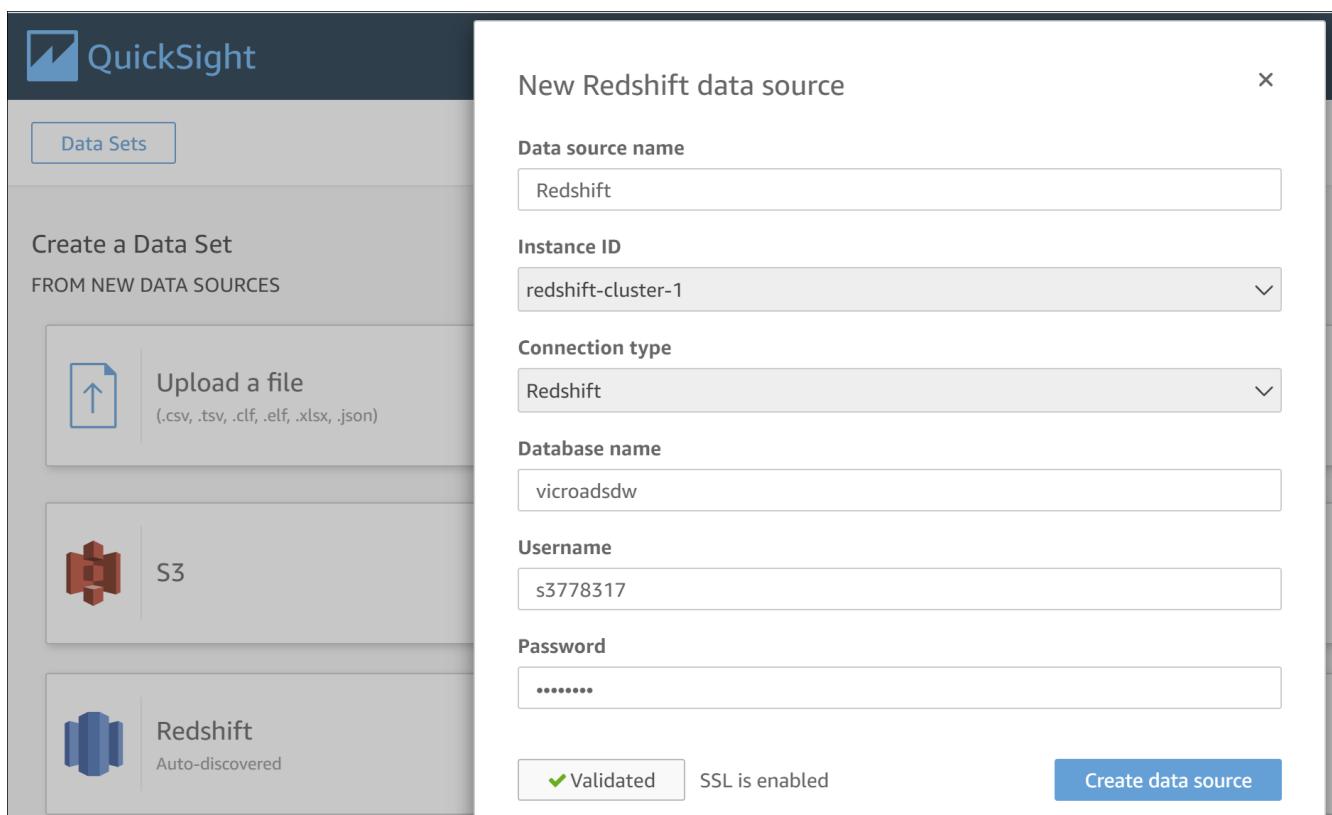
The screenshot shows the QuickSight interface with the title "Manage VPC connections". It displays a table with one row for a Redshift connection, showing details like ARN, Subnet ID, Security group ID, and DNS resolvers. Buttons for "Manage VPC connections" and "Add VPC connection" are visible.

VPC connection name	VPC connection ARN	Subnet ID	Security group ID	DNS resolvers
Redshift	arn:aws:quicksight:ap-southeast-2:909882575885:vpcConnection/Redshift	subnet-513ddb19	sg-0baef786052af56f6f	

b) Create a data set from Redshift to build visual

The screenshot shows the "Create a Data Set" section under "FROM NEW DATA SOURCES". It lists various data sources: Upload a file, Salesforce, S3 Analytics, S3, Athena, RDS, Redshift (Auto-discovered), Redshift (Manual connect), and MySQL. The "Redshift (Auto-discovered)" option is highlighted.

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)

The screenshot shows the AWS Route 53 dashboard. On the left sidebar, there are several navigation links: Dashboard, Hosted zones, Health checks, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Pending requests, Resolver, and VPCs. The main content area has a section titled "Hosted zones" with a brief description: "endpoints in complex configurations." It includes two buttons: "Create policy" and "Create health check". Below this is a "Register domain" section with a sub-instruction: "Find and register an available domain, or transfer your existing domains to Route 53." It features a search bar for "Type a domain name" and a dropdown for "TLD" (e.g., .com - \$12.00). A large blue "Check" button is prominent. Under "Alerts", there is a table showing one alert for "srirmit.com" with status "Domain registration successful" and last update "2020-05-22 23:13:01".

b) Request SSL certificate (Certification Manager):

The screenshot shows the AWS Certificate Manager interface. The left sidebar lists "Certificates" and "Certificate Manager" (which is selected). Other options include "Private certificate authority" and "Private CAs". The main content area is titled "Certificates" and contains a message about AWS Certificate Manager logs. It features three buttons: "Request a certificate", "Import a certificate", and "Actions". Below these is a table listing certificates. One row is highlighted for "srirmit.com", showing it was issued by "Amazon Issued" and is "In use? Yes".

c) Create an EBS environment

The screenshot shows the AWS Elastic Beanstalk environment creation wizard. The left sidebar has "Elastic Beanstalk" selected, with "Environments" and "Applications" listed under "Recent environments". The main content area shows the "Create environment" step. It starts with a "Select environment tier" section, which explains that there are two types: "Web server environment" (selected) and "Worker environment". Both descriptions mention running over port 80. Below this is a detailed description of the "Web server environment". At the bottom right are "Cancel" and "Select" buttons.

The screenshot shows the 'Create a web server environment' wizard. On the left, a sidebar lists 'Environments', 'Applications', and 'Recent environments' (vicroadsapp-env). The main area is titled 'Create a web server environment' and contains a sub-section 'Application information'. It has a field 'Application name' with the value 'vicroadsapplication' and a note below it: 'Up to 100 Unicode characters, not including forward slash (/)'.

The screenshot shows the 'Environment information' step. The sidebar remains the same. The main area is titled 'Environment information' and asks for 'Choose the name, subdomain, and description for your environment. These cannot be changed later.' It has fields for 'Environment name' (Vicroadsapplication-env), 'Domain' (sridevi-divya-ebs.ap-southeast-2.elasticbeanstalk.com), and a 'Check availability' button which shows a success message: 'sridevi-divya-ebs.ap-southeast-2.elasticbeanstalk.com is available.'

The screenshot shows the 'Platform' configuration step. The sidebar includes 'Applications' and 'Recent environments' (vicroadsapp-env). The main area has two options: 'Managed platform' (selected) and 'Custom platform'. Under 'Managed platform', it says 'Platforms published and maintained by AWS Elastic Beanstalk. Learn more' and lists 'Platform' (PHP), 'Platform branch' (PHP 7.4 running on 64bit Amazon Linux 2), and 'Platform version' (3.0.1 (Recommended)).

Below this is the 'Application code' section, which includes 'Sample application' (selected), 'Existing version', and 'Upload your code'. A note at the bottom states: 'Managed updates are now enabled by default for new environments on supporting platforms.'

At the bottom right are buttons for 'Cancel', 'Configure more options', and 'Create environment'.

Select configure more options:

Elastic Beanstalk

Environments Applications

Recent environments
vicroadsapp-env

Configure Vicroadsapplication-env

Presets

Start from a preset that matches your use case or choose *Custom configuration* to unset recommended values and use the service's default values.

Configuration presets

- Single instance (*Free Tier eligible*)
- Single instance (using Spot instance)
- High availability
- High availability (using Spot and On-Demand instances)
- Custom configuration

Elastic Beanstalk

Environments Applications

Recent environments
vicroadsapp-env

Modify load balancer

Application Load Balancer
Application layer load balancer—routing HTTP and HTTPS traffic based on protocol, port, and route to environment processes.

Classic Load Balancer
Previous generation — HTTP, HTTPS, and TCP

Network Load Balancer
Ultra-high performance and static IP addresses for your application.

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

Elastic Beanstalk

Environments Applications

vicroadsapp
Application versions Saved configurations

Application Load Balancer

You can specify listeners for your load balancer. Each listener routes incoming client traffic on a specified port using a specified protocol to your environment processes. By default, we've configured your load balancer with a standard web server on port 80.

<input type="checkbox"/>	Port	Protocol	SSL certificate	Enabled
<input type="checkbox"/>	443	HTTPS	sirimt.com - 134fd945-53e5-4820-b8b9-3dff479e4672	<input checked="" type="checkbox"/>
<input type="checkbox"/>	80	HTTP	--	<input checked="" type="checkbox"/>

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

Elastic Beanstalk

Environments Applications

vicroadsapp
Application versions Saved configurations

Rules

Your load balancer routes requests to environment processes based on rules. Rules with lower priority numbers have higher precedence. If a request doesn't match any rule's pattern, the request is routed to the process associated with the default rule.

<input type="checkbox"/>	Listener port	Name	Priority	Path pattern	Process
<input type="checkbox"/>	443	default	--	/*	default
<input type="checkbox"/>	80	default	--	/*	default

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

Elastic Beanstalk

Environments Applications

▼ vicroadsapp Application versions Saved configurations

Modify network

Virtual private cloud (VPC)

VPC
vpc-33b8b554 (172.31.0.0/16)

Elastic Beanstalk

Environments Applications

▼ vicroadsapp Application versions Saved configurations

▼ vicroadsapp-env

Visibility
Make your load balancer internal if your application serves requests only from connected VPCs. Public load balancers serve requests from the Internet.

Public

Load balancer subnets

Availability Zone	Subnet	CIDR	Name
ap-southeast-2a	subnet-513ddb19	172.31.32.0/20	
ap-southeast-2b	subnet-86b541e0	172.31.0.0/20	
ap-southeast-2c	subnet-3f7d1a67	172.31.16.0/20	

Elastic Beanstalk

Environments Applications

▼ vicroadsapp Application versions Saved configurations

▼ vicroadsapp-env

Go to environment Configuration Logs Health Monitoring Alarms

Instance settings
Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances.

Public IP address
Assign a public IP address to the Amazon EC2 instances in your environment.

Instance subnets

Availability Zone	Subnet	CIDR	Name
ap-southeast-2a	subnet-513ddb19	172.31.32.0/20	
ap-southeast-2b	subnet-86b541e0	172.31.0.0/20	
ap-southeast-2c	subnet-3f7d1a67	172.31.16.0/20	

Upload and deploy the application

Elastic Beanstalk

Environments Applications

▼ vicroadsapp Application versions Saved configurations

▼ vicroadsapp-env

Go to environment

vicroadsapp-env
sridevipamarthi.ap-southeast-2.elasticbeanstalk.com (e-c7k35mmnxy)
Application name: vicroadsapp

Health
Ok

Running version
Sample Application-65

Platform
PHP 7.4 running on 64bit.
Linux 2/3.0.1

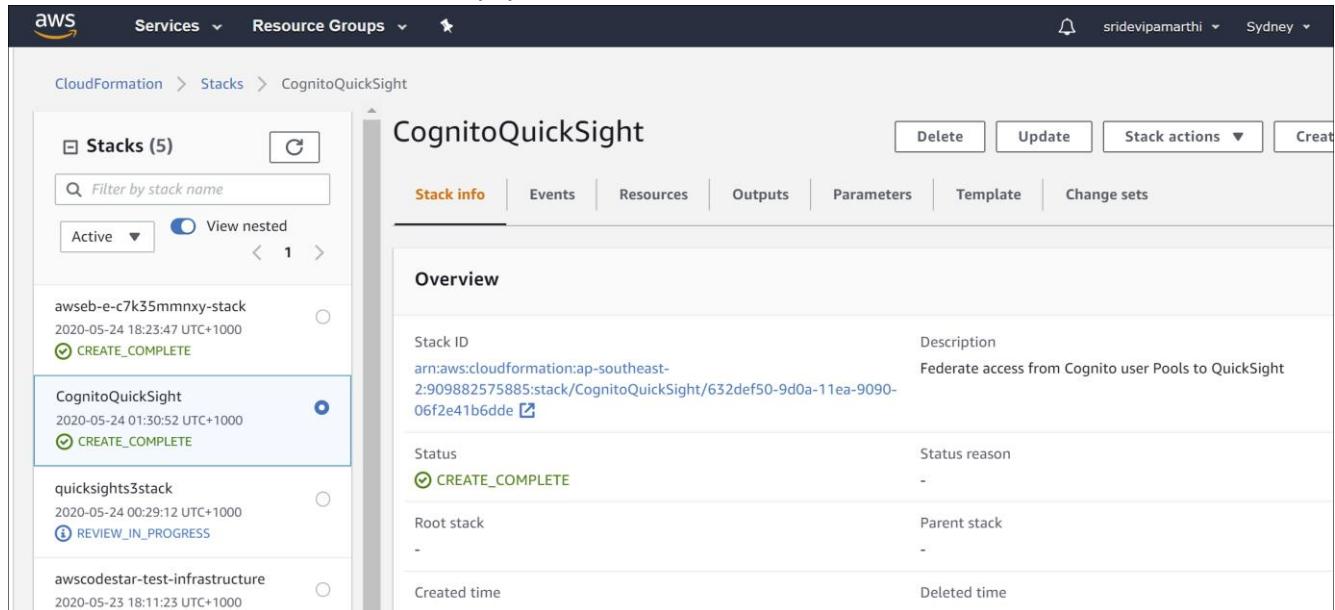
Upload and deploy

7. QuickSight Sign-On with Cognito User Pools

This has been 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

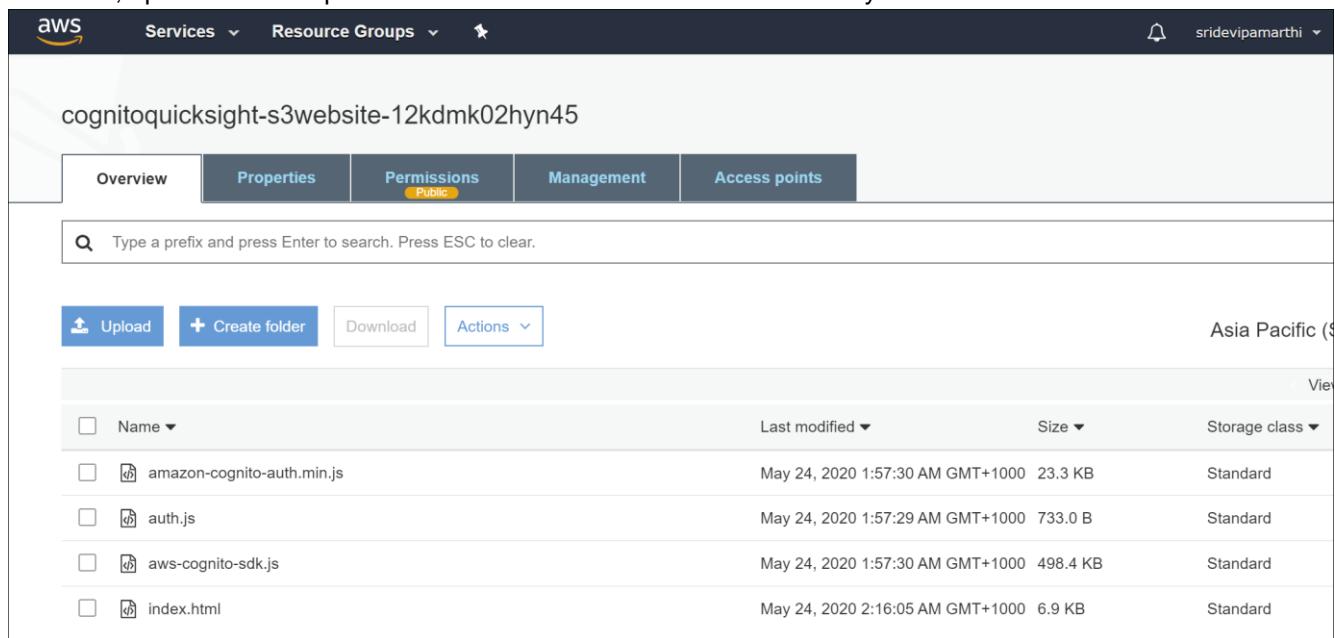
To achieve this, the cloud formation script provided in the above URL has been executed.



The screenshot shows the AWS CloudFormation console with the 'Stacks' page. On the left, a sidebar lists five stacks: 'awseb-e-c7k35mmnny-stack' (CREATE_COMPLETE), 'CognitoQuickSight' (CREATE_COMPLETE, highlighted in blue), 'quicksights3stack' (REVIEW_IN_PROGRESS), and 'awscodestar-test-infrastructure' (CREATE_IN_PROGRESS). The main panel displays the 'CognitoQuickSight' stack details. The 'Stack info' tab is selected, showing the stack ID as 'arn:aws:cloudformation:ap-southeast-2:909882575885:stack/CognitoQuickSight/632def50-9d0a-11ea-9090-06f2e41b6dde'. Other tabs include Events, Resources, Outputs, Parameters, Template, and Change sets. The 'Overview' section provides a summary of the stack's status, root stack, and creation time.

b) Upload content into S3 static website

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



The screenshot shows the AWS S3 console with the 'cognitoquicksight-s3website-12kdmk02hyn45' bucket selected. The 'Properties' tab is active. The bucket contains four objects: 'amazon-cognito-auth.min.js', 'auth.js', 'aws-cognito-sdk.js', and 'index.html'. The 'Actions' dropdown menu is open, showing options like 'Upload', 'Create folder', 'Download', and 'Actions'. The 'Permissions' section indicates 'Public' access. The 'Access points' section shows 'Asia Pacific (Singapore)' as the endpoint. The table lists the object names, last modified dates, sizes, and storage classes.

Name	Last modified	Size	Storage class
amazon-cognito-auth.min.js	May 24, 2020 1:57:30 AM GMT+1000	23.3 KB	Standard
auth.js	May 24, 2020 1:57:29 AM GMT+1000	733.0 B	Standard
aws-cognito-sdk.js	May 24, 2020 1:57:30 AM GMT+1000	498.4 KB	Standard
index.html	May 24, 2020 2:16:05 AM GMT+1000	6.9 KB	Standard

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

The screenshot shows the AWS Cognito User Pools console. The navigation bar at the top includes the AWS logo, Services (dropdown), Resource Groups (dropdown), a bell icon, sridevapamarthi (username), Sydney (region), and Support. The main navigation on the left lists General settings, Users and groups, Attributes, Policies, MFA and verifications, Advanced security, Message customizations, Tags, Devices, App clients, Triggers, Analytics, and App integration. Under App integration, 'Domain name' is highlighted with a yellow background. The central content area has a title 'What domain would you like to use?'. It contains a note about domain prefixes and an 'Amazon Cognito domain' section with a 'Domain prefix' input field containing 'https://sri-divya-qs.auth.ap-southeast-2.amazoncognito.com'. A 'Delete domain' button is located to the right of the input field. Below this is a 'Your own domain' section with a note about associated certificates and a 'Use your domain' button.

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.

The screenshot shows a web browser window titled 'QuickSight via Cognito'. The address bar displays the URL 'd1n892ypt9usay.cloudfront.net'. The page content features two large, stylized icons (one purple, one blue) on the left. In the center, the text 'Use Cognito User Pools credentials to access QuickSight' is displayed. At the bottom right is a dark blue button with the text 'SIGN IN / SIGN UP' in white.

Signin

sri-divya-qs.auth.ap-southeast-2.amazoncognito.com/signup?redirect_uri=https%3A%2Fd1n892ypt9usay.cl

Sign up with a new account

Username

s3778317

Email

sridevikilari95@gmail.com

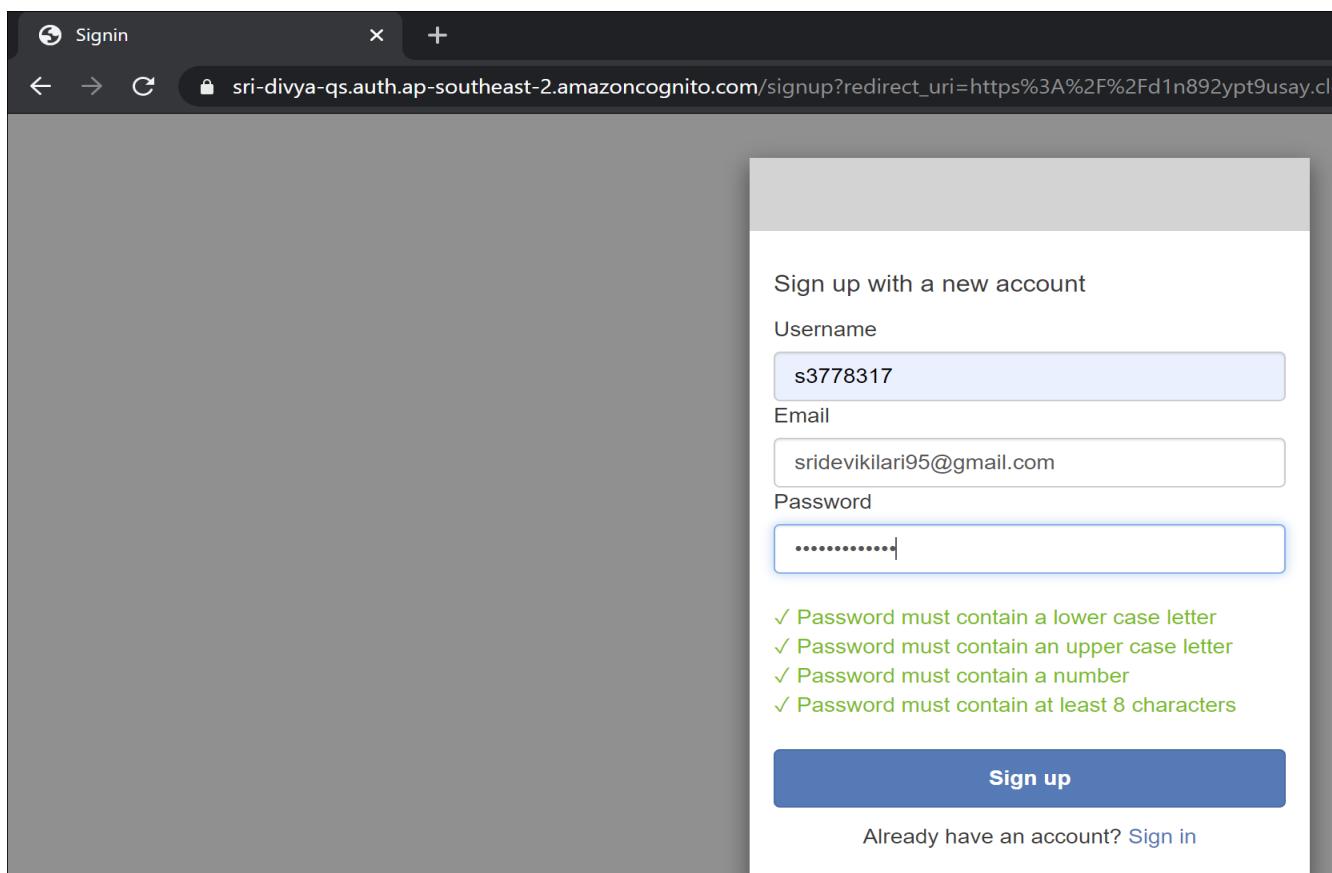
Password

.....

✓ 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](#)

The screenshot shows a web browser window with a sign-up form for an Amazon Cognito user. The URL in the address bar is sri-divya-qs.auth.ap-southeast-2.amazoncognito.com/signup?redirect_uri=https%3A%2Fd1n892ypt9usay.cl. The form includes fields for Username (s3778317), Email (sridevikilari95@gmail.com), and Password (a masked string). Below the password field are four validation messages indicating requirements: a lowercase letter, an uppercase letter, a number, and a minimum length of 8 characters. A blue "Sign up" button is at the bottom, and a link to "Sign in" is below it.

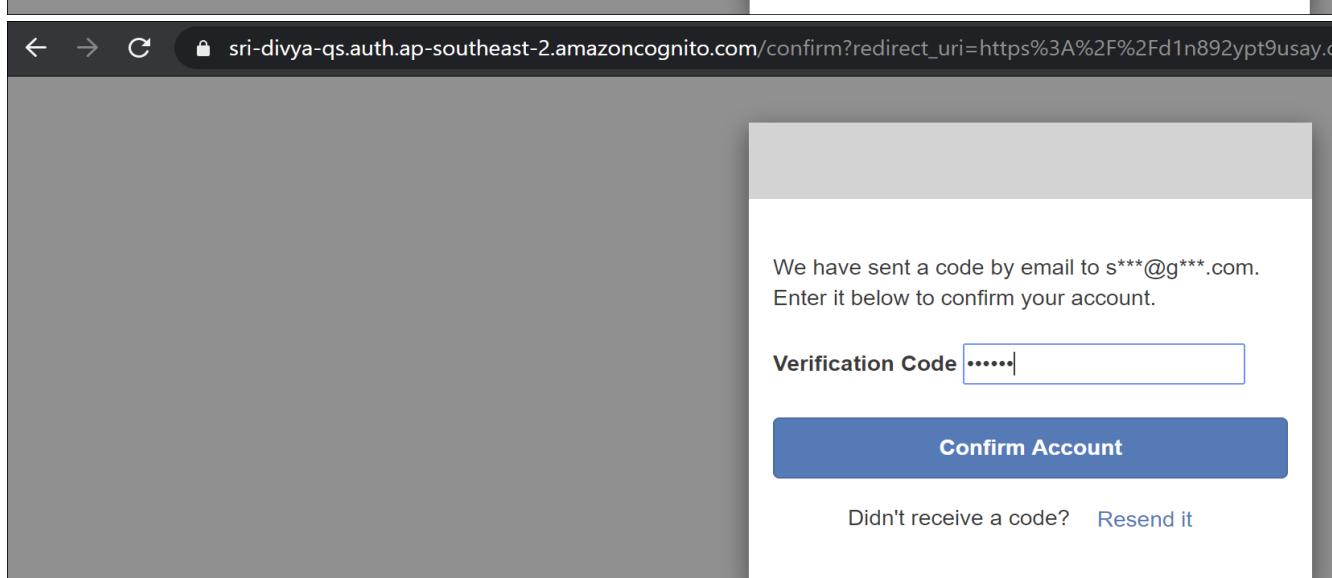
sri-divya-qs.auth.ap-southeast-2.amazoncognito.com/confirm?redirect_uri=https%3A%2Fd1n892ypt9usay.cl

We have sent a code by email to s***@g***.com. Enter it below to confirm your account.

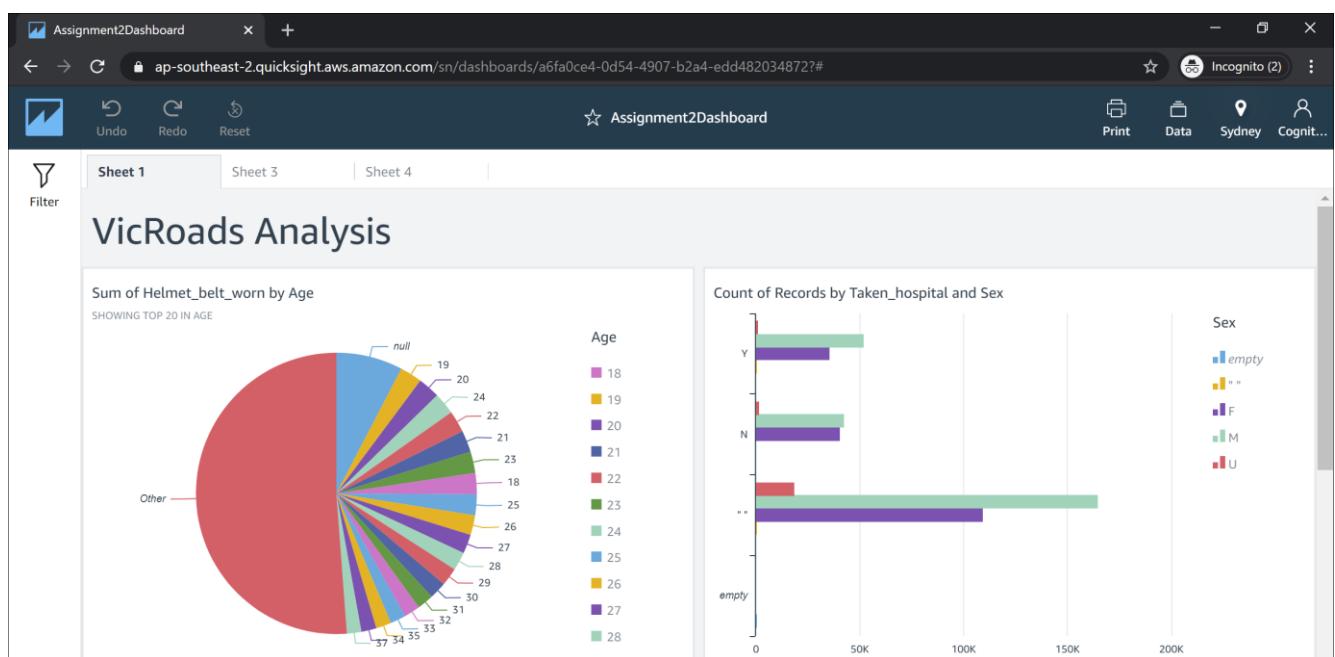
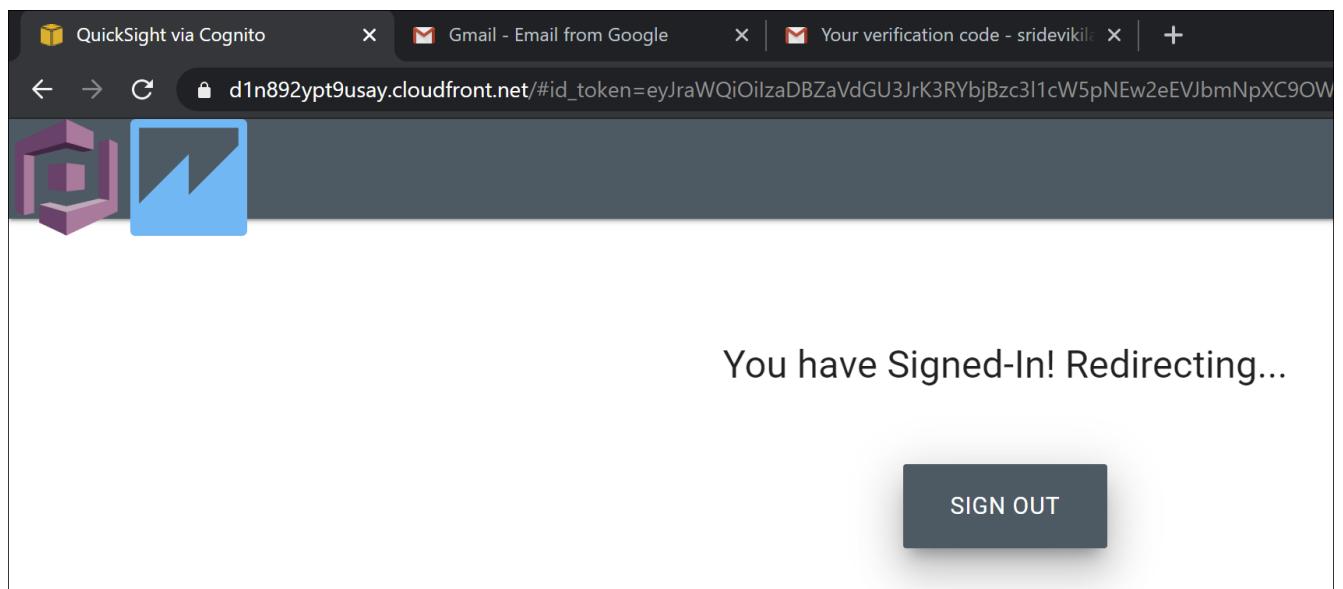
Verification Code

Confirm Account

Didn't receive a code? [Resend it](#)

The screenshot shows a confirmation screen for an account. The URL in the address bar is sri-divya-qs.auth.ap-southeast-2.amazoncognito.com/confirm?redirect_uri=https%3A%2Fd1n892ypt9usay.cl. It displays a message telling the user a code has been sent via email and asks them to enter it to confirm their account. A "Verification Code" input field contains several masked characters. A blue "Confirm Account" button is at the bottom, and a link to "Resend it" is below it.

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

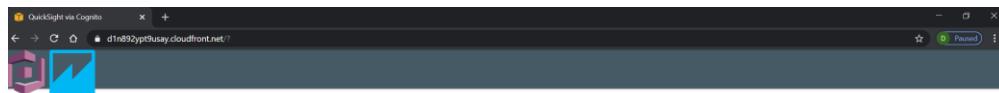
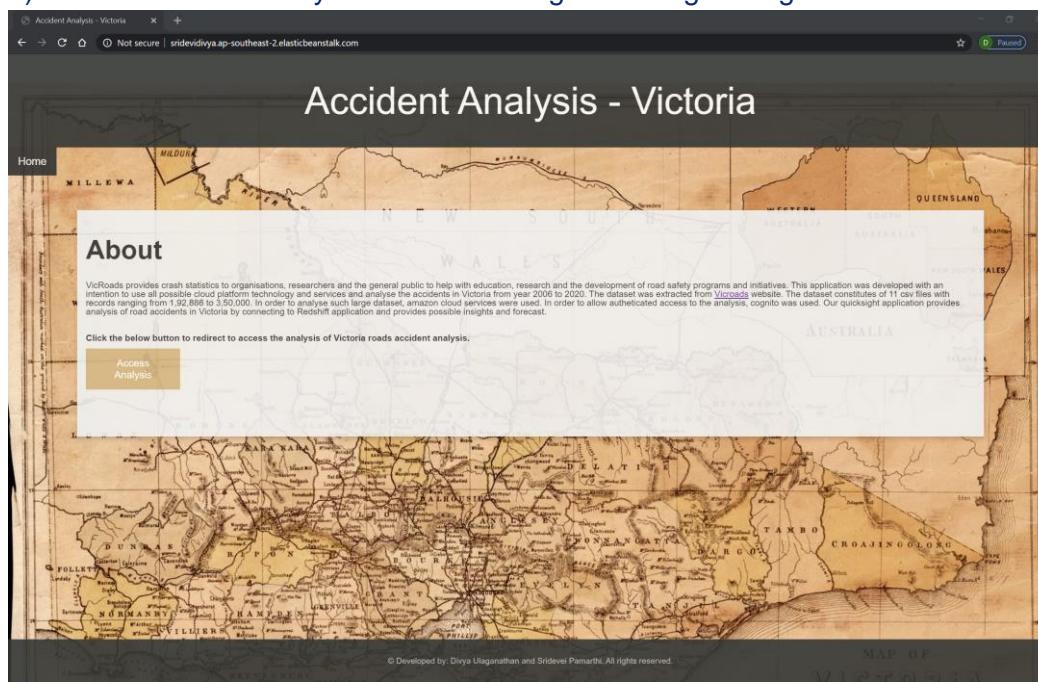


A small user manual

a) Navigate to the link :

<http://sridevidivya.ap-southeast-2.elasticbeanstalk.com/>

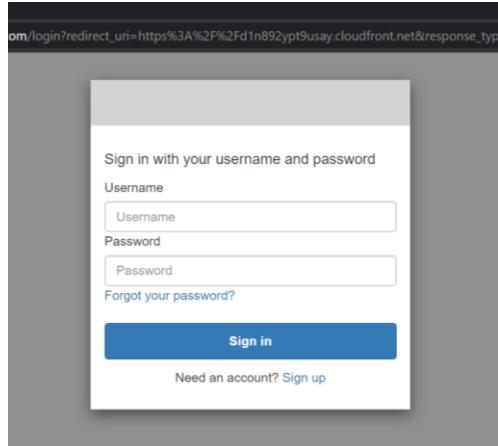
b) Click on Access analysis button to navigate to Cognito login.



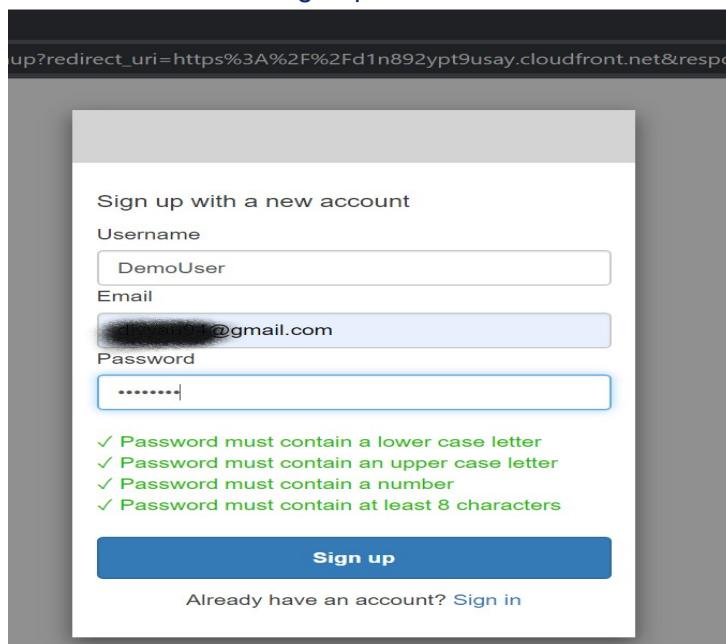
Use Cognito User Pools credentials to access QuickSight

SIGN IN / SIGN UP

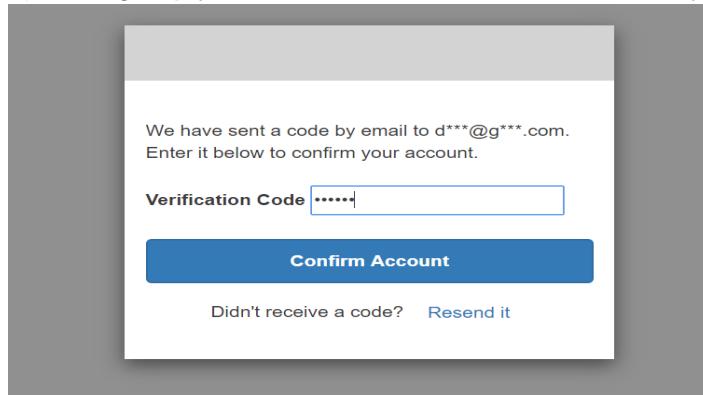
c) Click on “Sign In/ Sign Up” button. You will receive a popup like below.



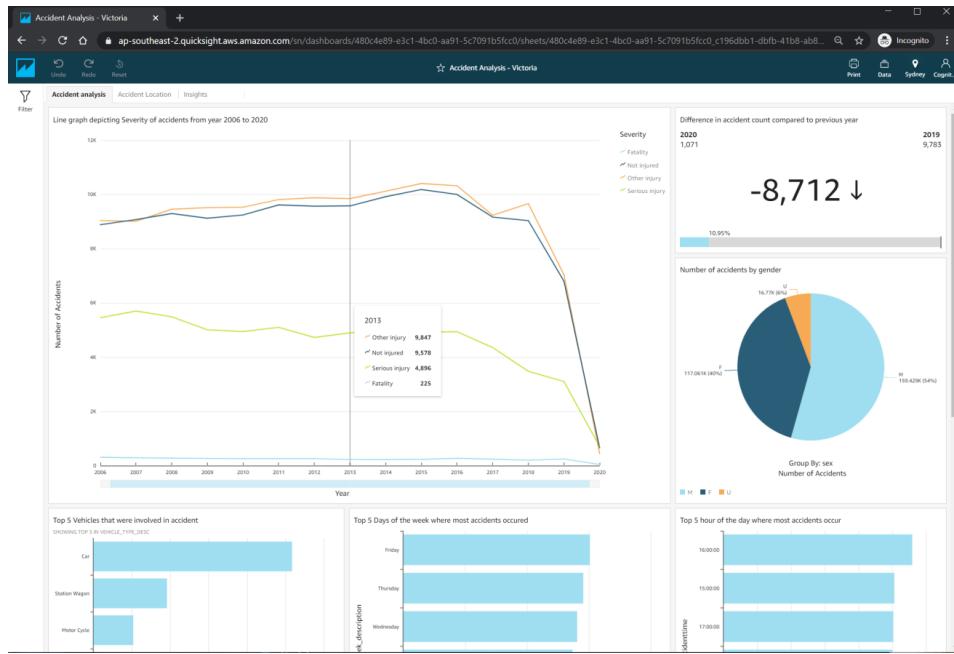
d) If you are a new user to this application enter your details to sign-in. Click Sign up and enter you details like below to Sign up.



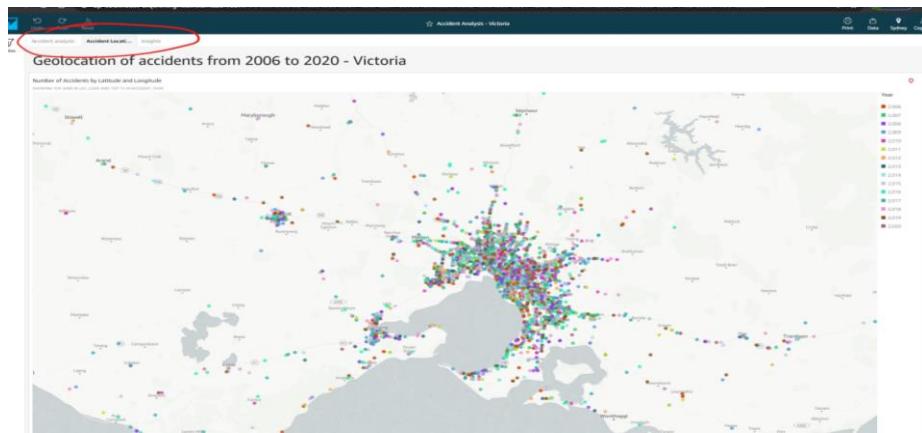
e) On Sign up you will receive a verification code on your respective email to confirm your account



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