

ASSIGNMENT – 4 PART A

GitLab Link:

<https://git.cs.dal.ca/umatiya/csci5410-f23-b00899642/-/tree/A4>

I followed steps below to solve the part A of the assignment:

The common steps I took is given below:

Step 1:

- Create an Account and login to the AWS Academy as shown in fig 1 & 2.

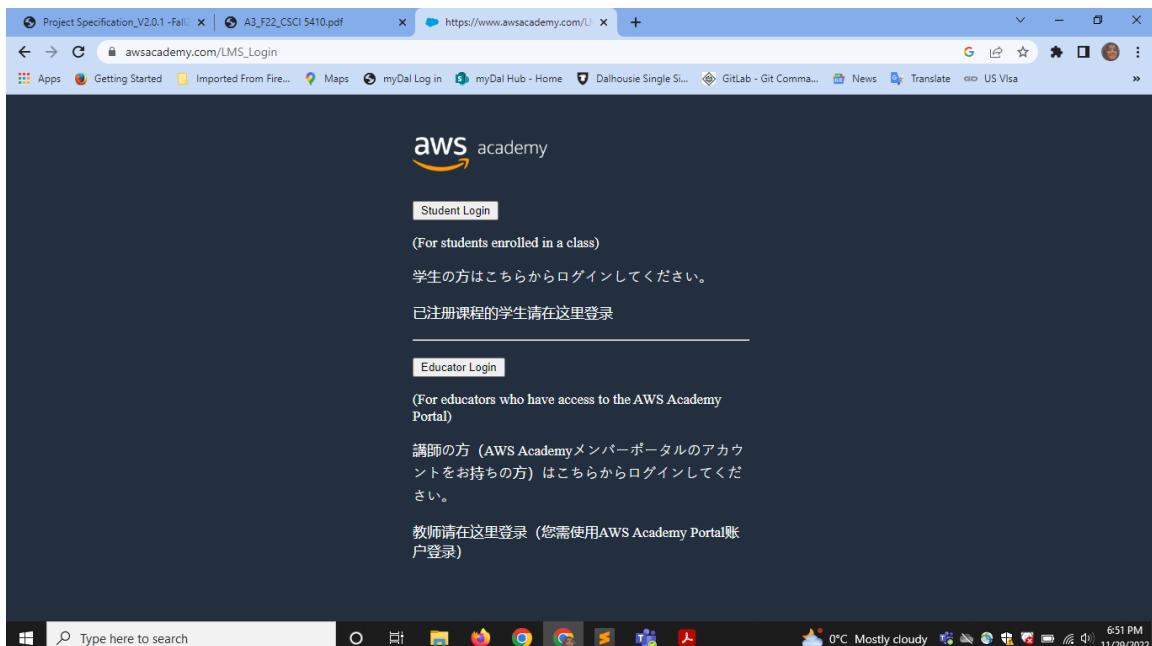


Fig 1: AWS Academy account login page

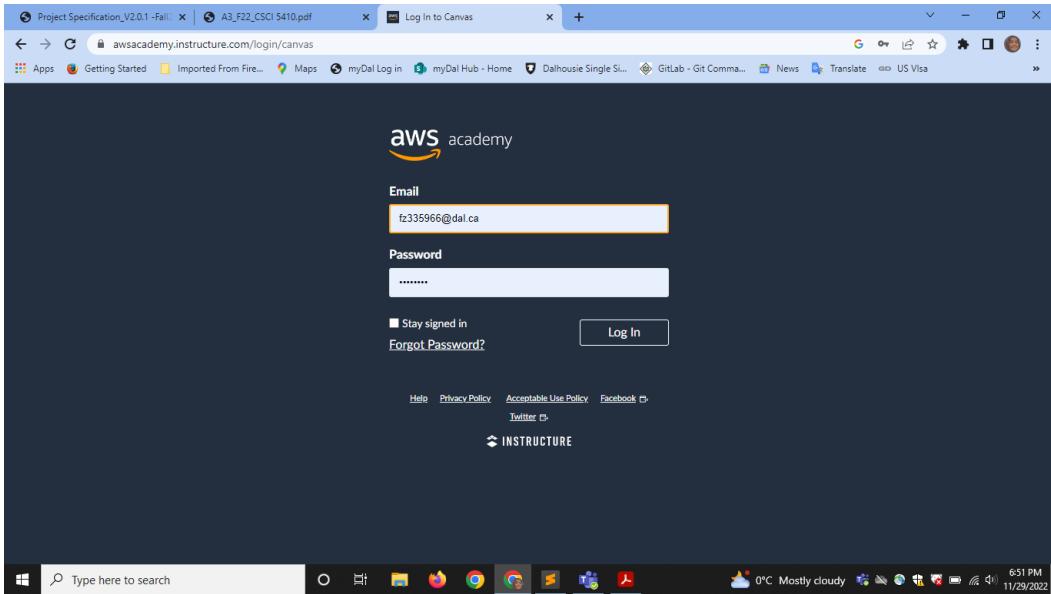


Fig 2: Student Account login page

- After login to the AWS academy, go to AWS Academy Learner Lab as shown in the fig 3.

Fig 3: AWS Dashboard

- The first window you'll see open is the AWS Dashboard as shown in the fig 3.
- Click on the AWS Academy Learner Lab shown in the fig. 3 and the new window will be open which is shown in the fig 4.

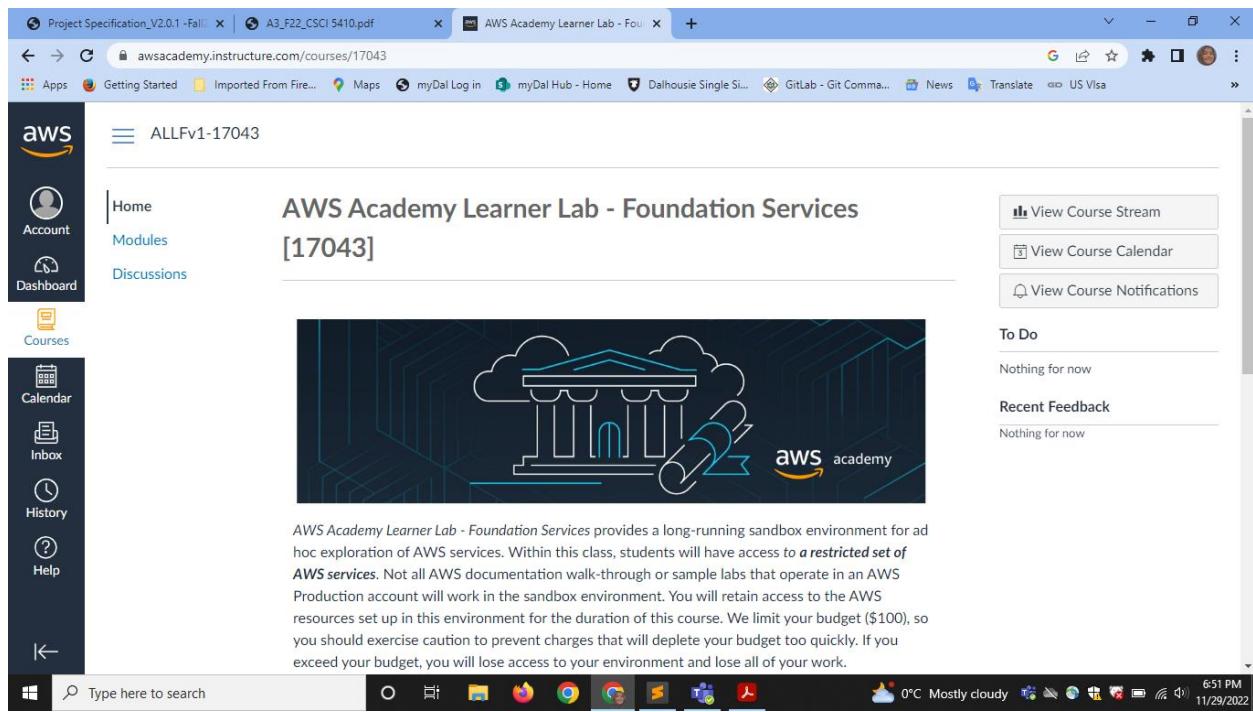


Fig 4: AWS home page

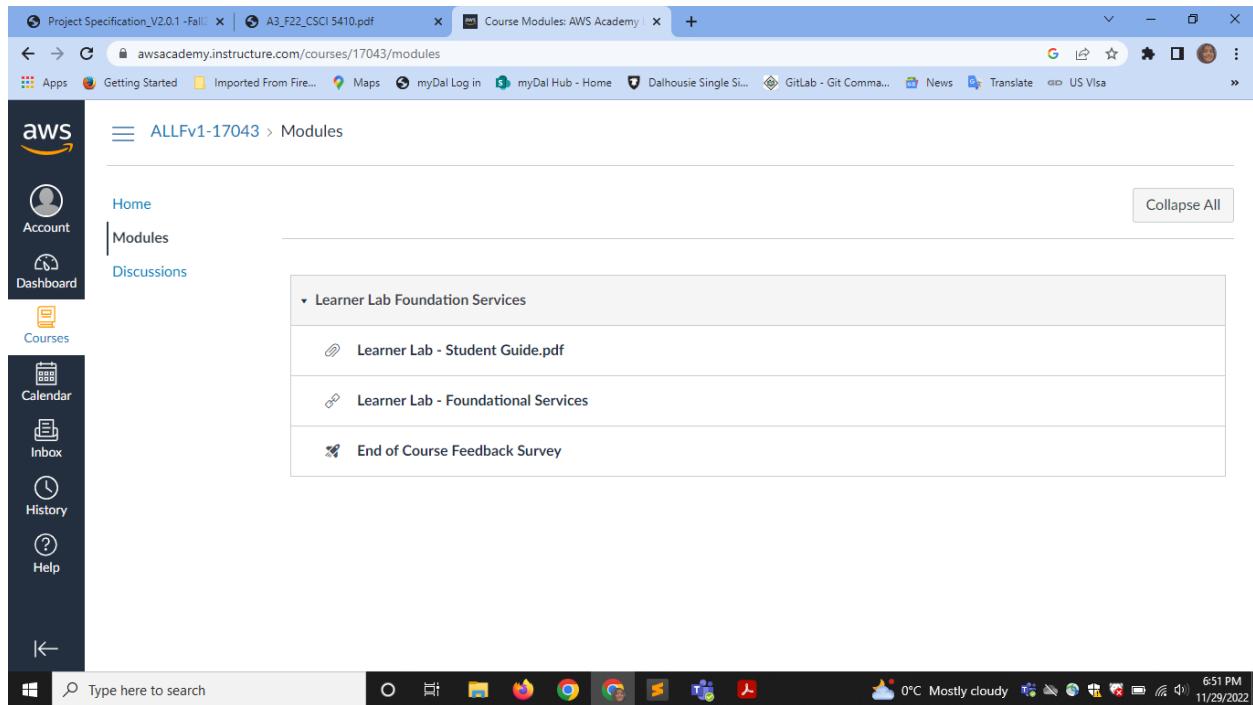


Fig 5: AWS Modules Page

- Go to modules as shown in the fig 4. After clicking on the modules, AWS modules page will open as shown in the fig 5.

- Go to “Learner Lab – Foundational Services” as shown in fig 5. After clicking on it, the start lab and end lab page will open as shown in the fig 6.
- Click on the Start Lab to start the lab (The red dot represents the lab is not started yet) as shown in the fig 6.

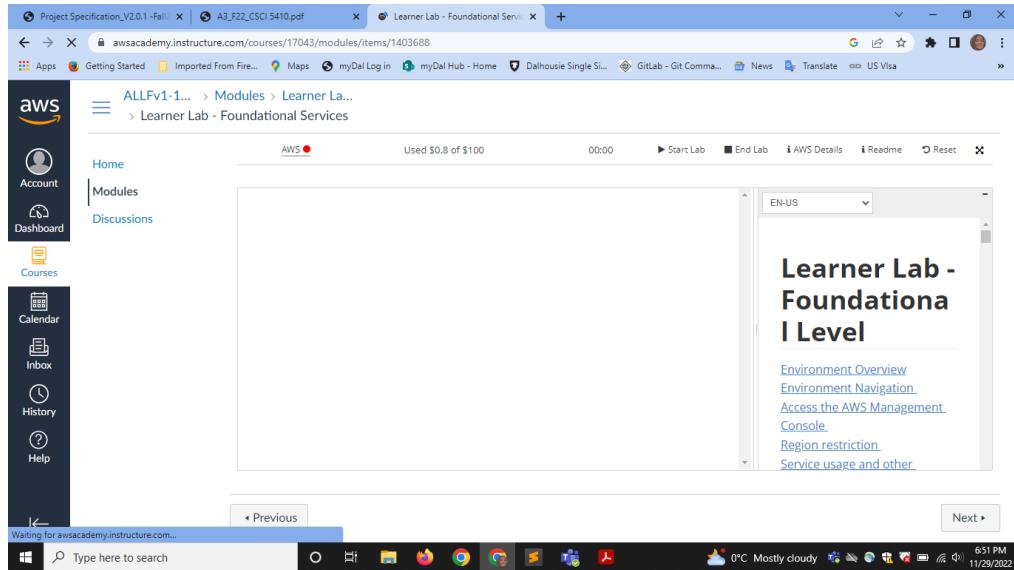


Fig 6: AWS End lab

- Lab has started because the green dot indicates the lab has started as shown in the fig 7.
- On clicking the AWS green icon which is shown in the fig. 7, it will redirect the user to the AWS Console Home as shown in the fig. 7.1. It contains all different services like Amazon Lex, DynamoDB, S3, Lambda and many more.

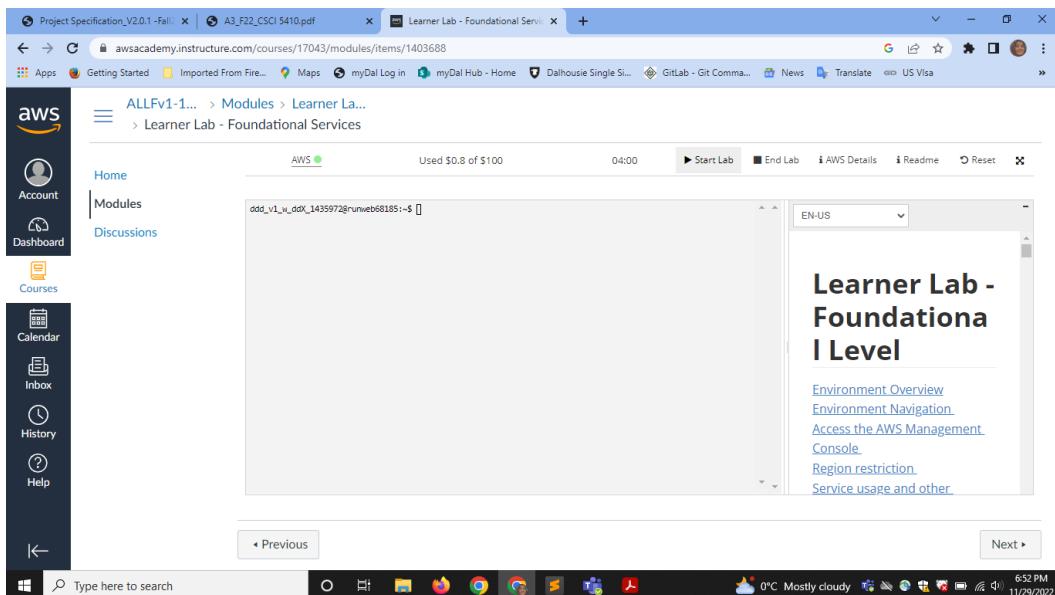


Fig 7: AWS start lab

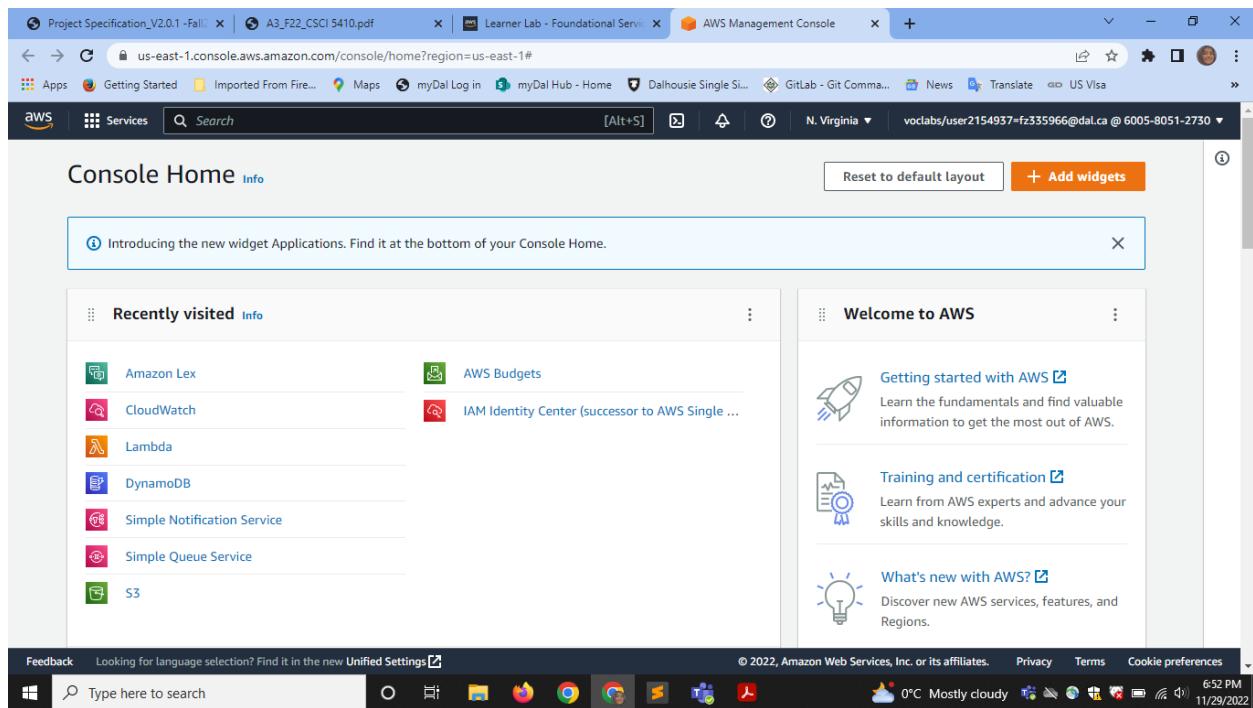


Fig 7.1: AWS Console Home

Step 2: Create SQS

- Go to AWS Console (follow step 1 to get here) and search for Amazon SQS as shown in the fig 8.
- Click on the “Create queue” to create queue as shown in the fig 9 or 10.
- Fill the details in the “Create queue” form as shown in the fig 11 – 17.
- The queue “PartA_SQSqueue” has been created and its details has been mentioned as shown in the fig 18.

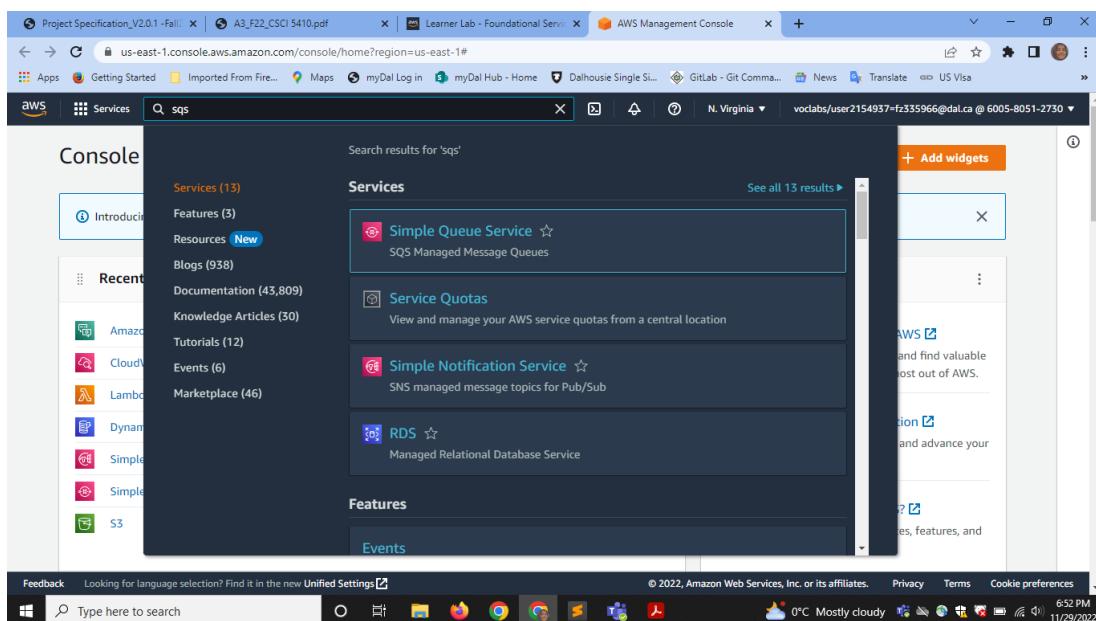


Fig 8: Search for Amazon SQS from search bar

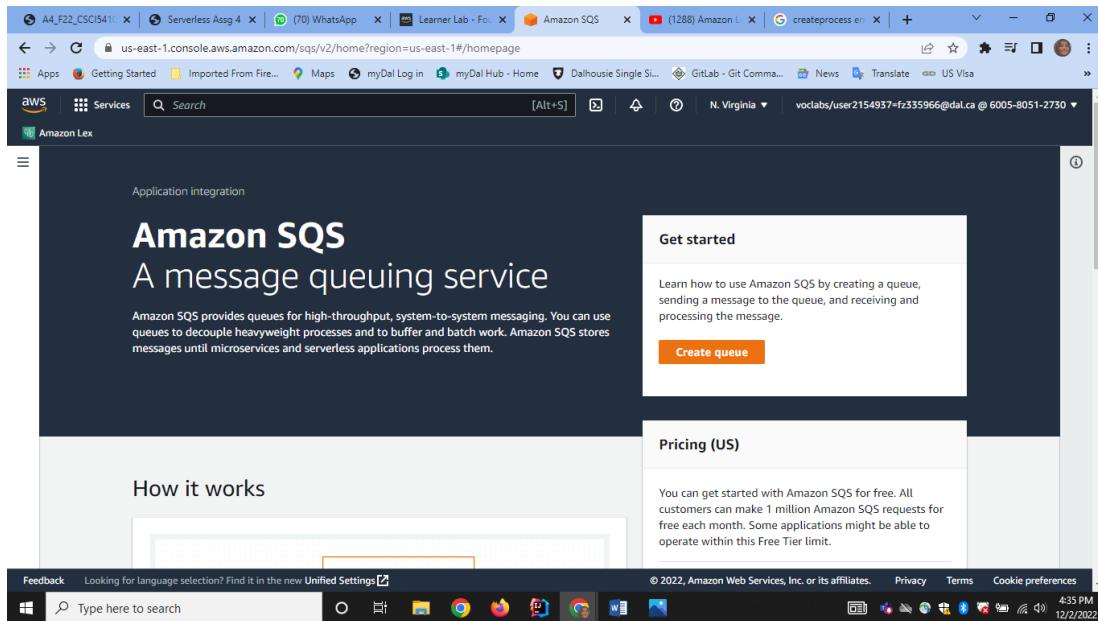


Fig 9: Amazon SQS home page

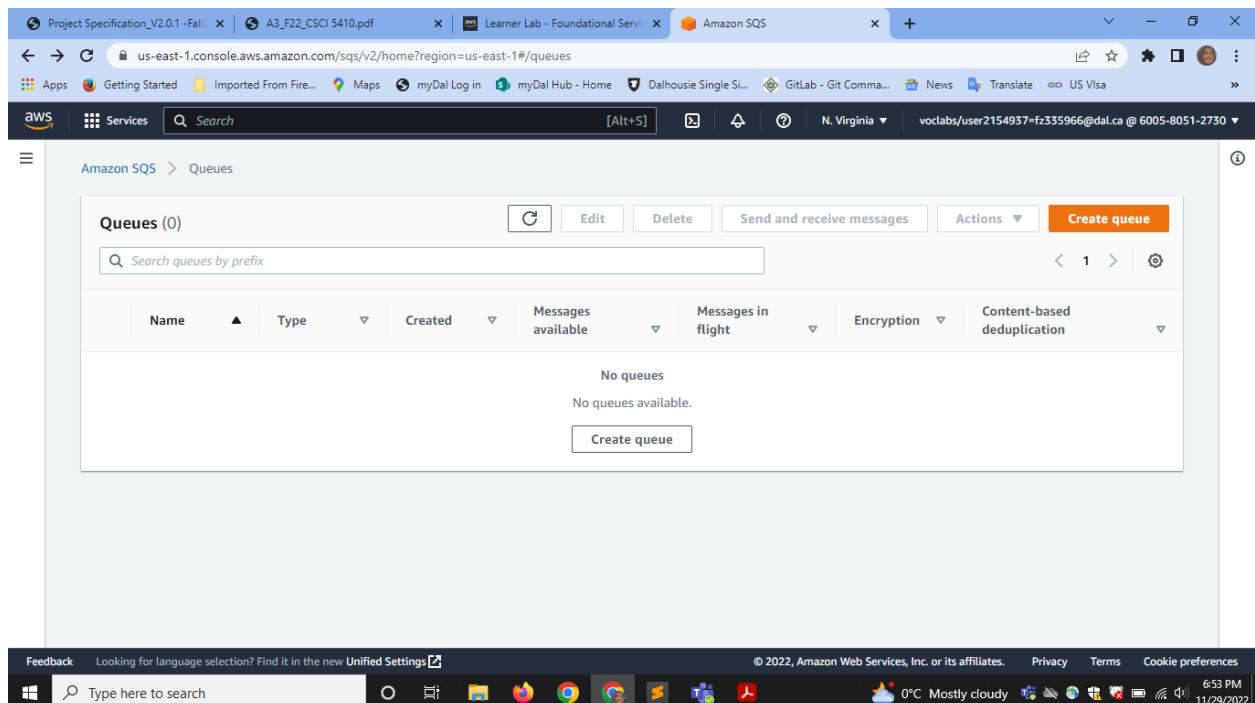


Fig 10: Queues page

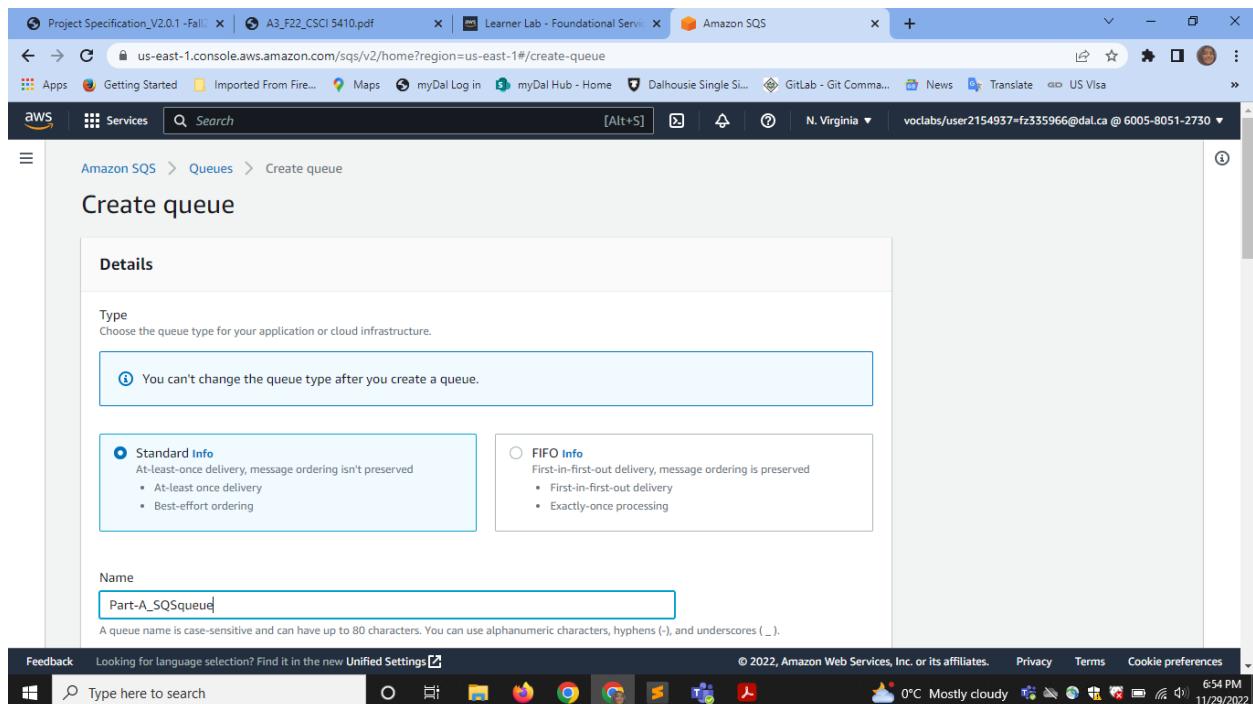


Fig 11: “Create Queue” details

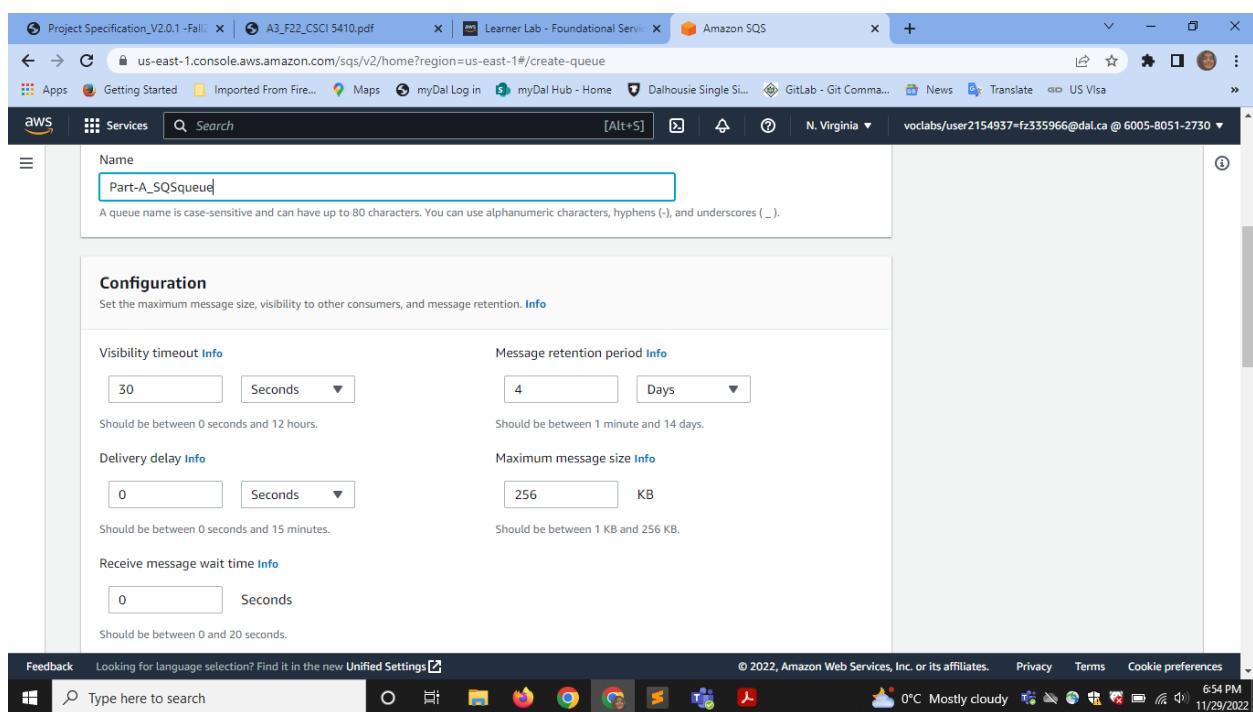


Fig 12: “Create Queue” details

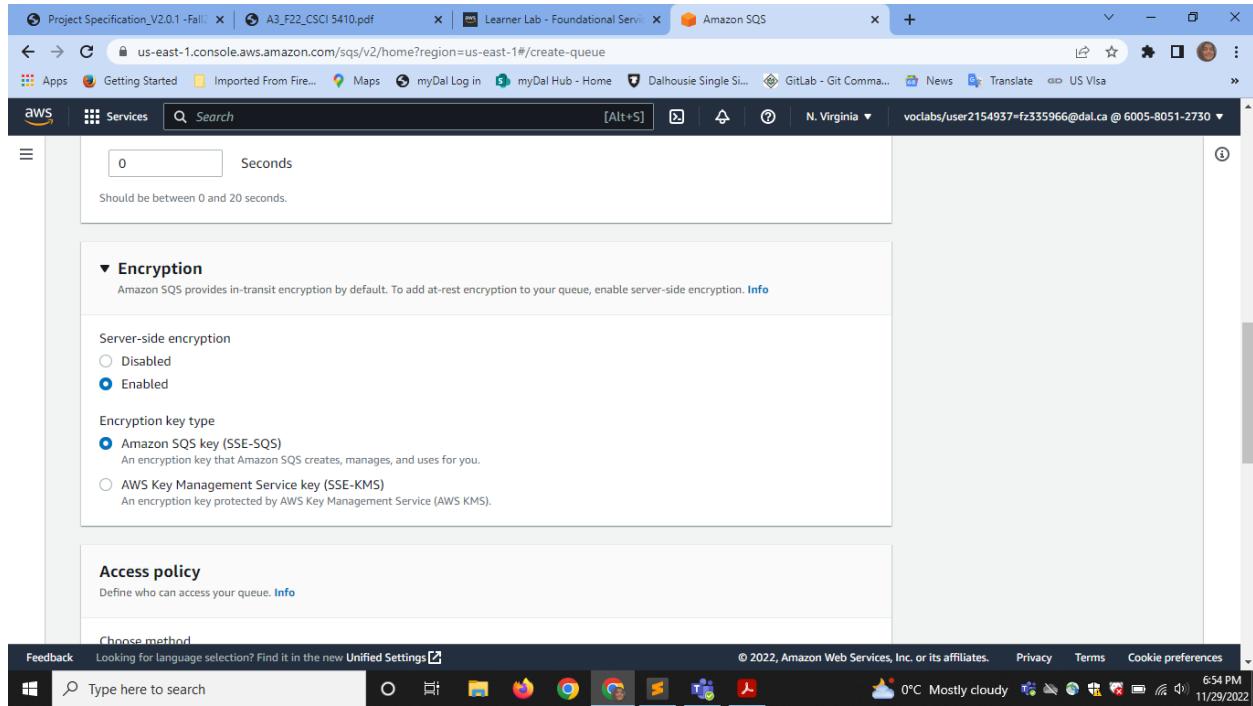


Fig 13: "Create Queue" details

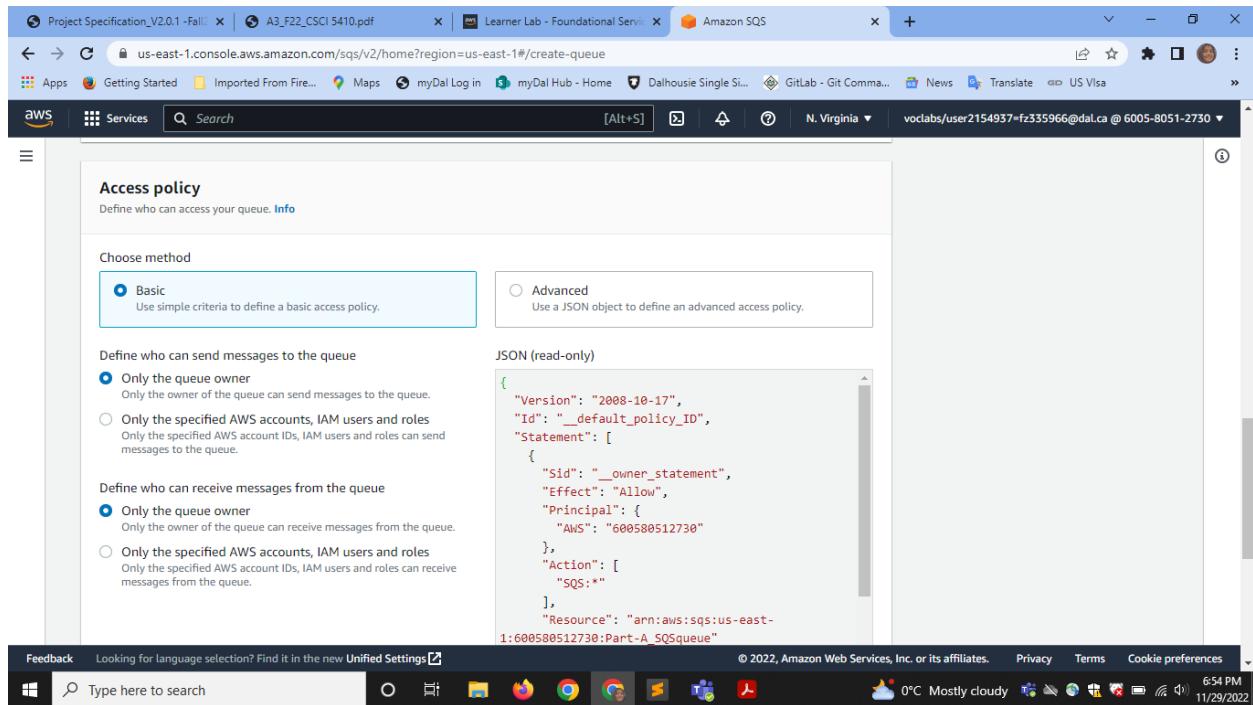


Fig 14: "Create Queue" details

The screenshot shows the AWS Lambda console with the 'Create Queue' page open. The 'Redrive allow policy' section is expanded, displaying a JSON policy document:

```
Id": "__default_policy_ID",
"Statement": [
{
  "Sid": "__owner_statement",
  "Effect": "Allow",
  "Principal": {
    "AWS": "600580512730"
  },
  "Action": [
    "SQS:*"
  ],
  "Resource": "arn:aws:sqs:us-east-1:600580512730:Part-A_SQSqueue"
}
```

The 'Dead Letter queue' section is collapsed.

Fig 15: “Create Queue” details

The screenshot shows the AWS Lambda console with the 'Create Queue' page open. The 'Dead-letter queue' section is expanded, showing options for setting the queue to receive undeliverable messages:

- Send undeliverable messages to a dead-letter queue. [Info](#)
- Set this queue to receive undeliverable messages.
- Disabled
- Enabled

The 'Tags' section is collapsed.

Fig 16: “Create Queue” details

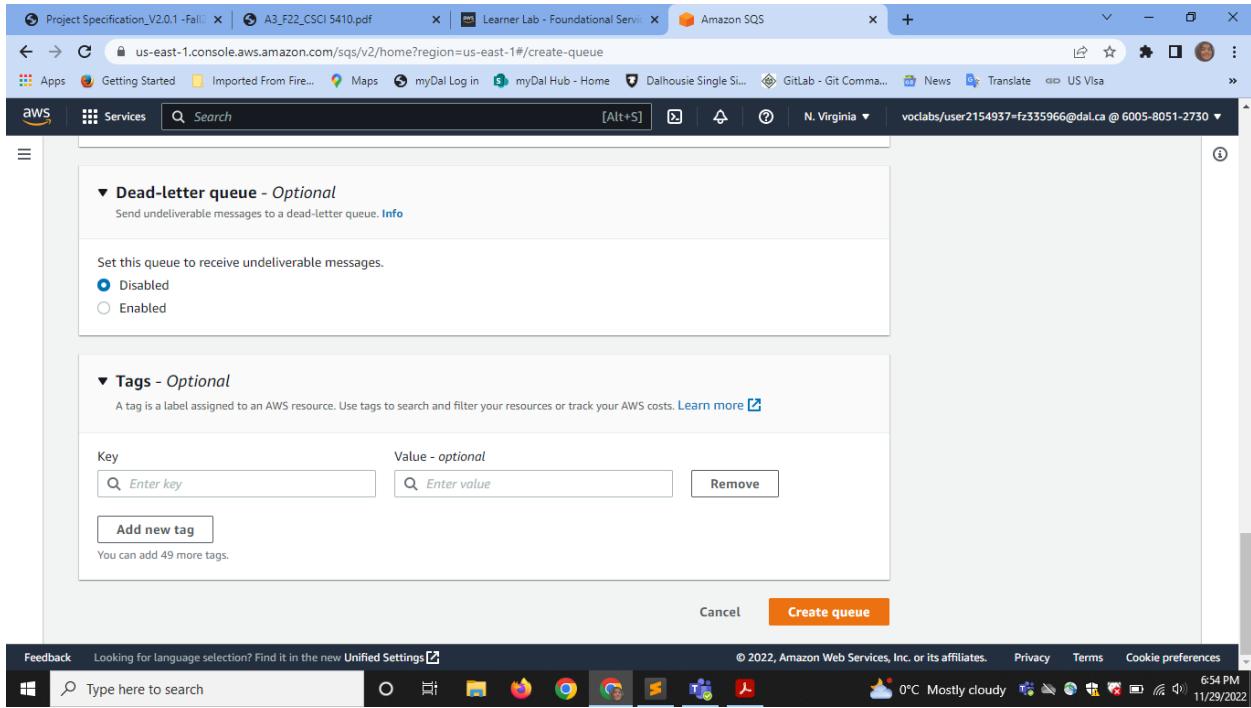


Fig 17: “Create Queue” details

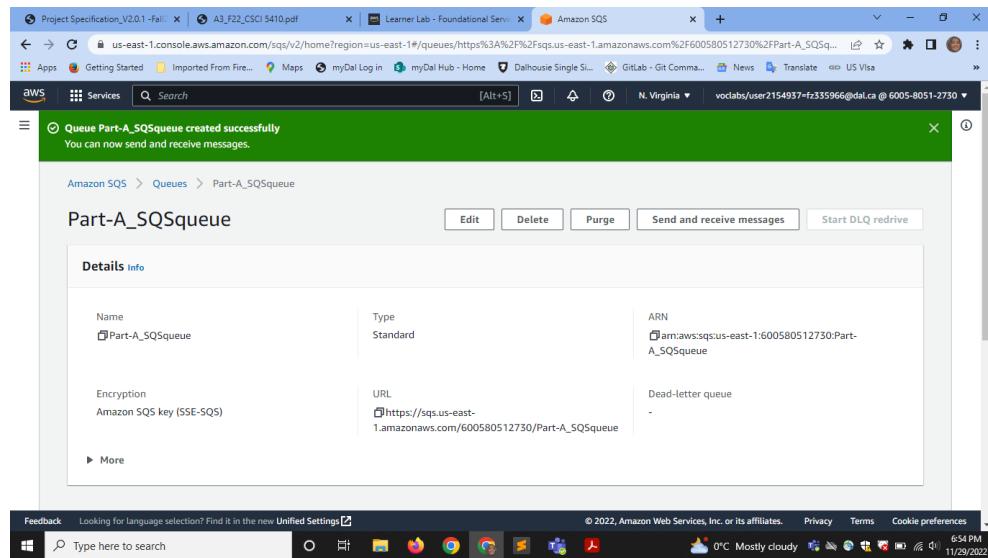


Fig 18: Queue has been created

Step 3: Create SNS

- Go to AWS Console (follow step 1 to get here) and search for Amazon SNS as shown in the fig 19.
- Click on the “Create topic” to create topic as shown in the fig 20.
- Fig 21 represents the dashboard for the SNS.
- Fill the details in the “Create queue” form as shown in the fig 22 – 32.

- The queue “Part-A_SNS” has been created and its details has been mentioned as shown in the fig 33.

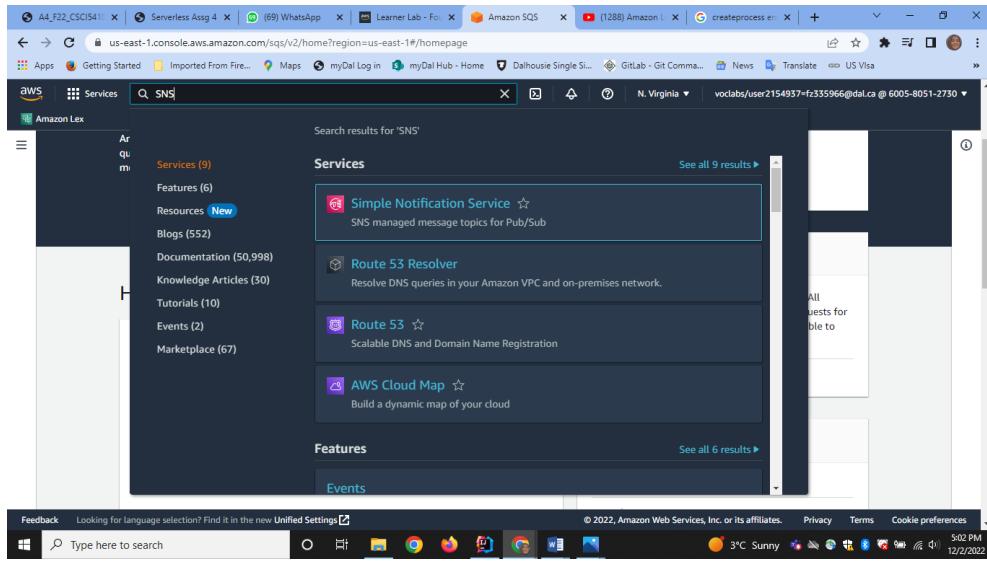


Fig 19: Search for Amazon SNS from search bar

A screenshot of the Amazon Simple Notification Service (SNS) homepage. The title 'Amazon Simple Notification Service' is prominently displayed. Below it, a large heading reads 'Pub/sub messaging for microservices and serverless applications.' A descriptive paragraph explains that Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service. To the right, there's a 'Create topic' form with a 'Topic name' field containing 'Part-A_SNS' and a 'Next step' button. Another button, 'Start with an overview', is also visible. At the bottom left, a 'Benefits and features' section is partially visible. The bottom of the page follows the standard AWS footer layout with links for Feedback, Unified Settings, and various AWS services.

Fig 20: Amazon SNS home page

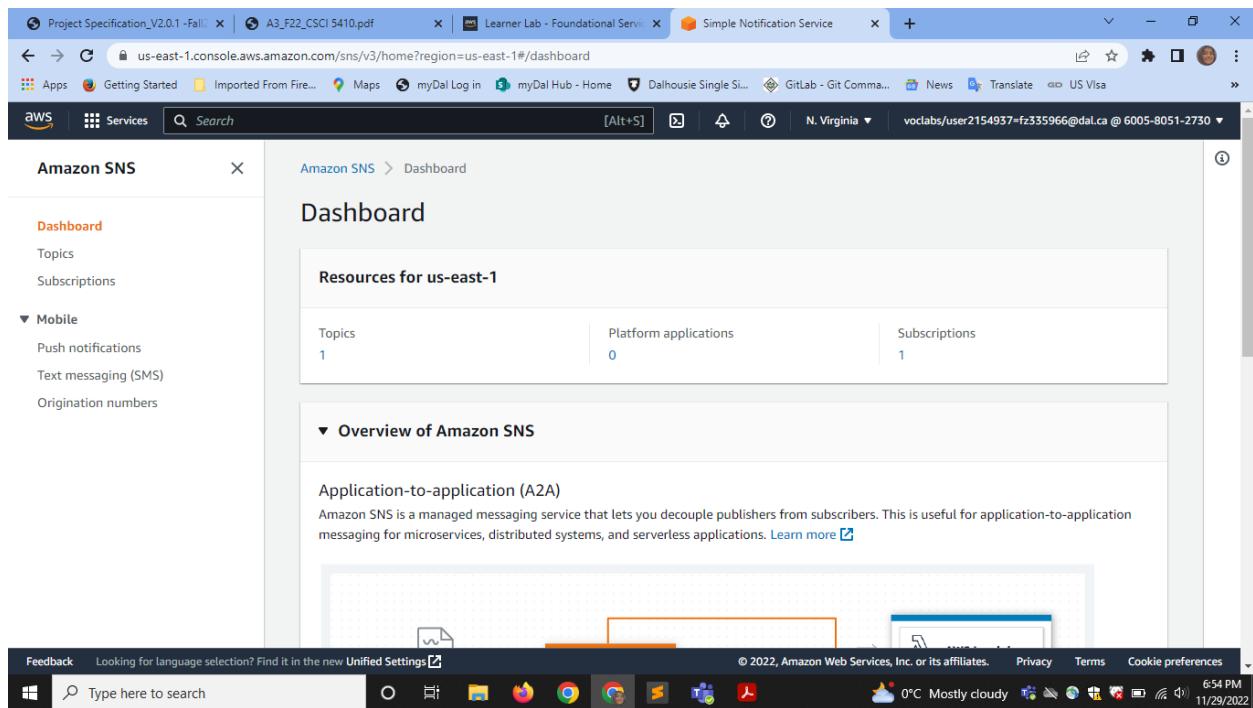


Fig 21: SNS Dashboard

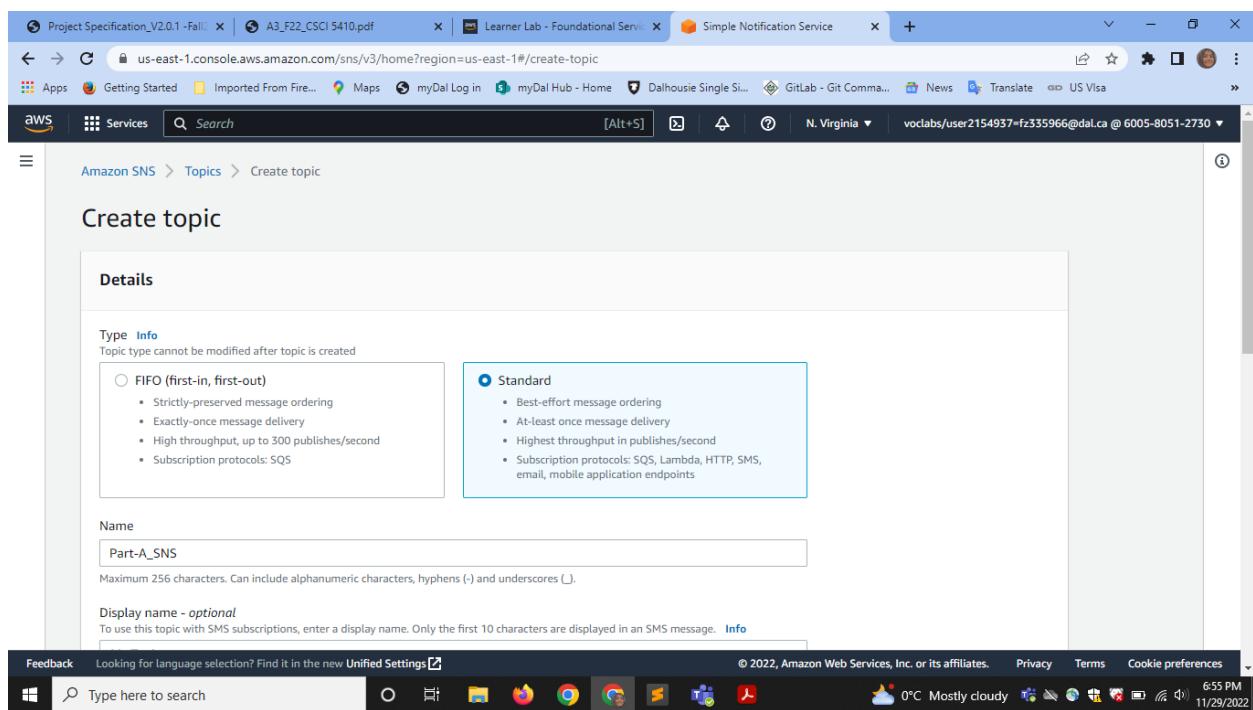


Fig 22: "Create topic" for SNS page

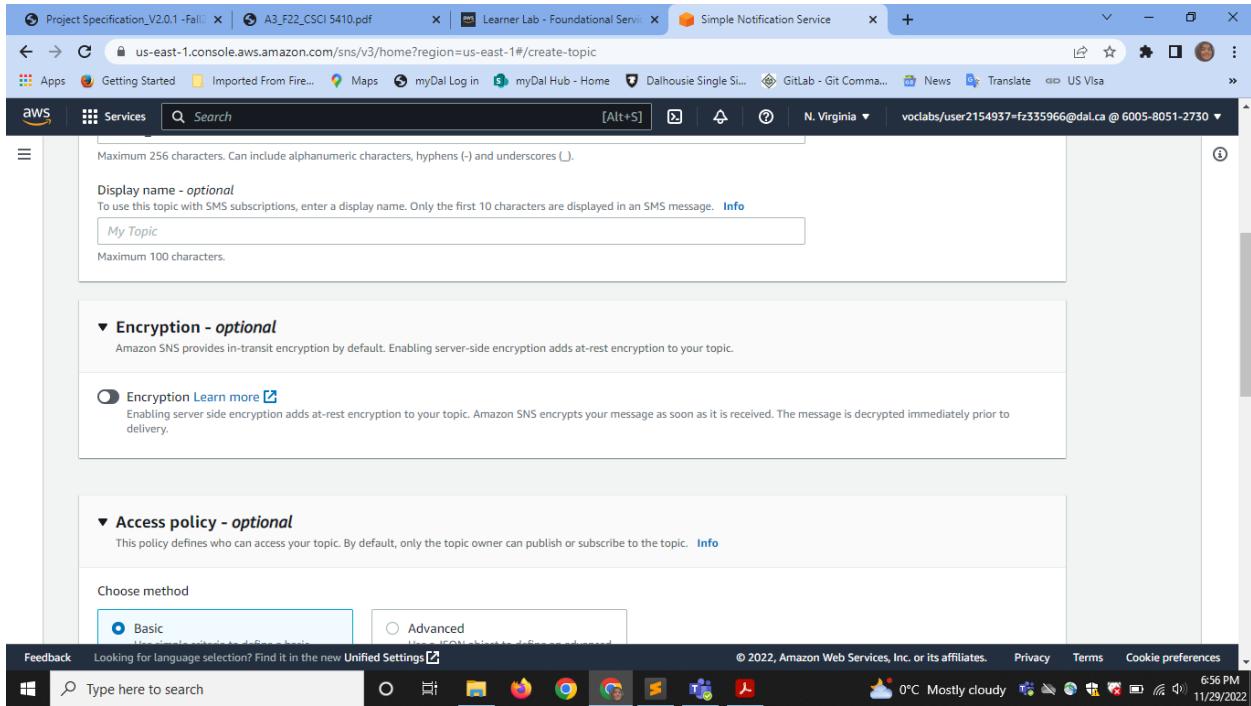


Fig 23: “Create topic” form

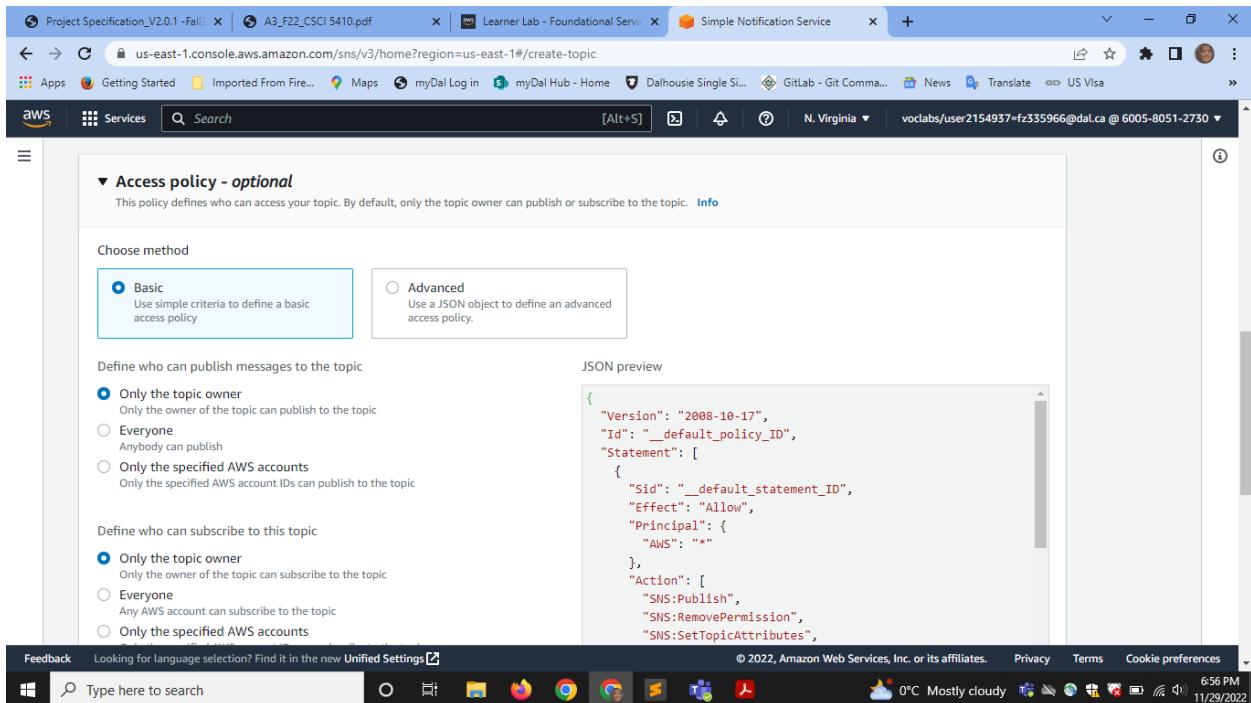


Fig 24: “Create topic” form

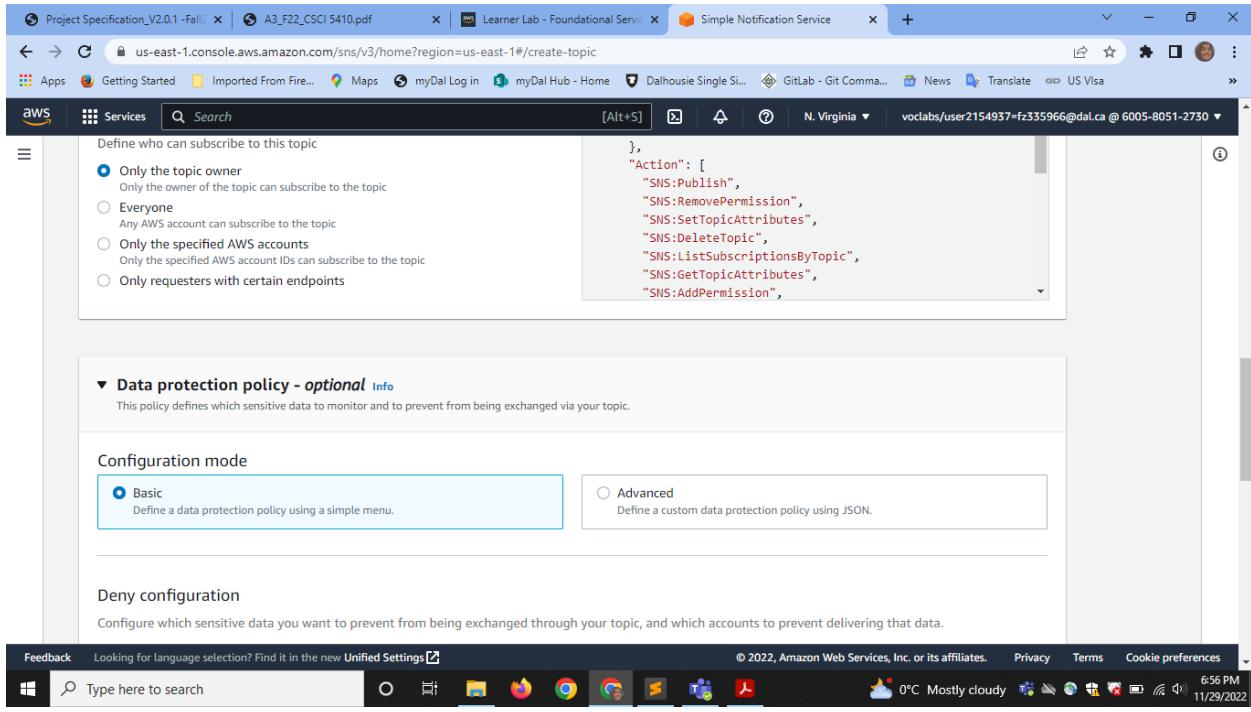


Fig 25: "Create topic" form

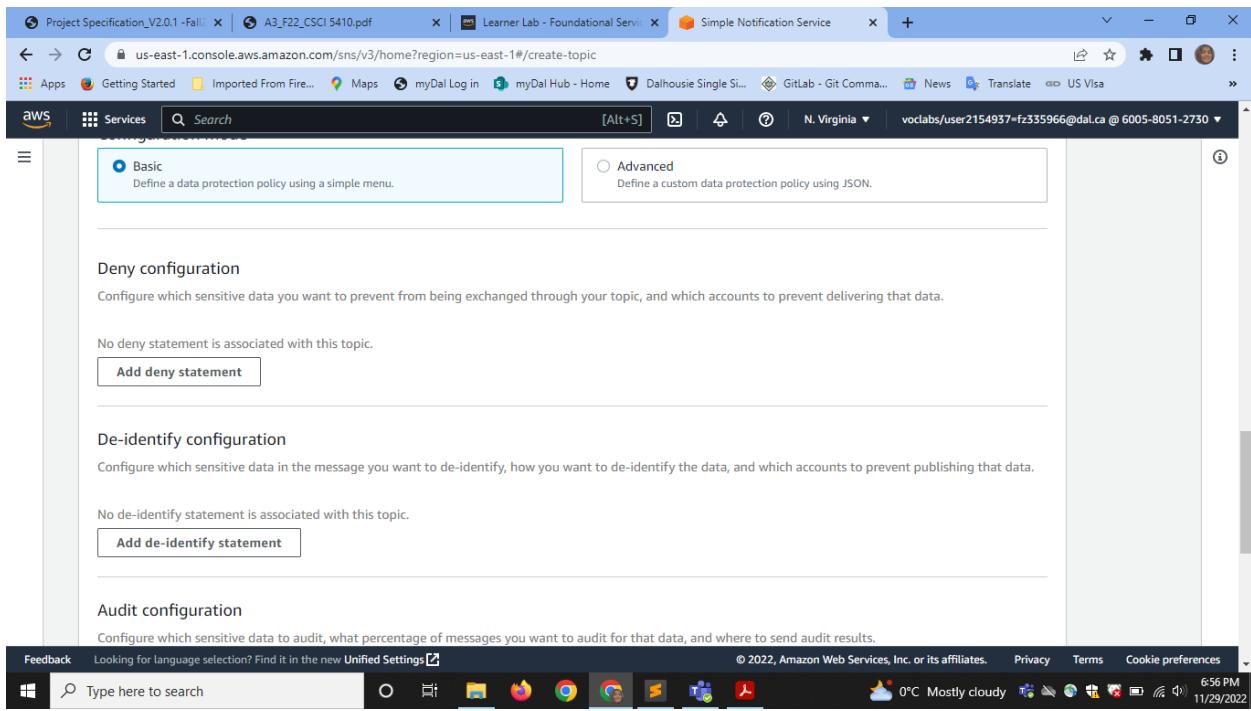


Fig 26: "Create topic" form

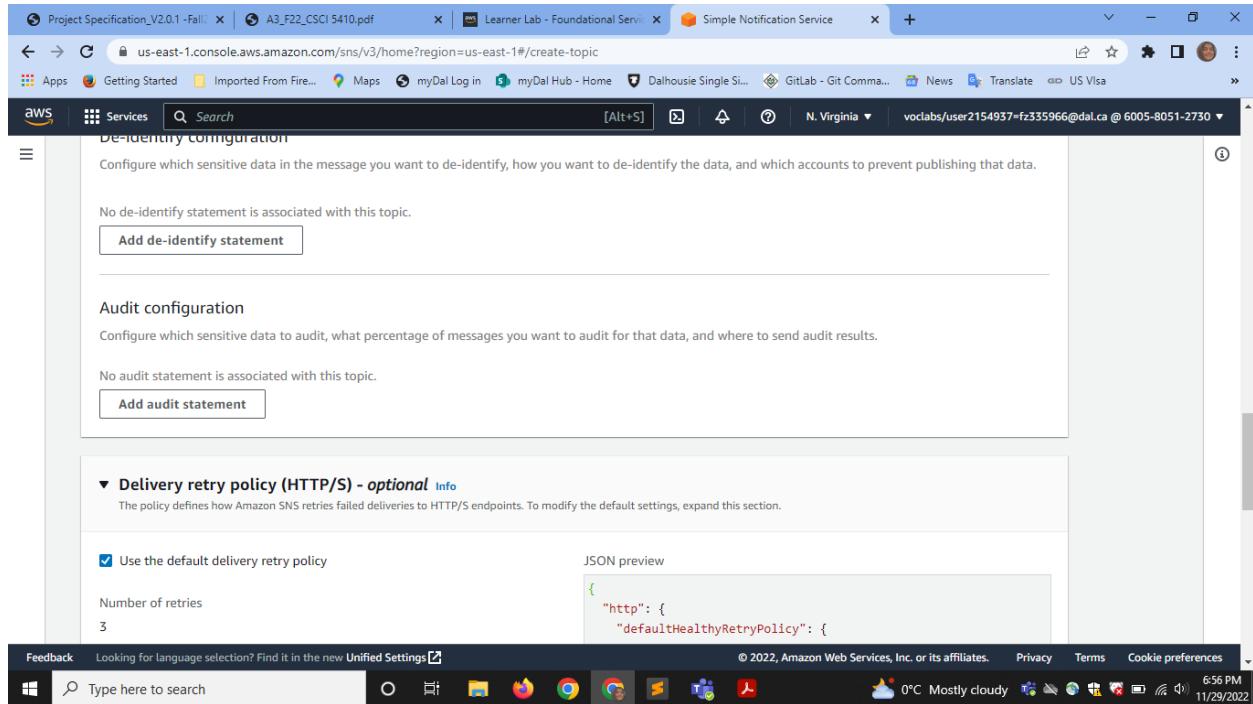


Fig 27: “Create topic” form

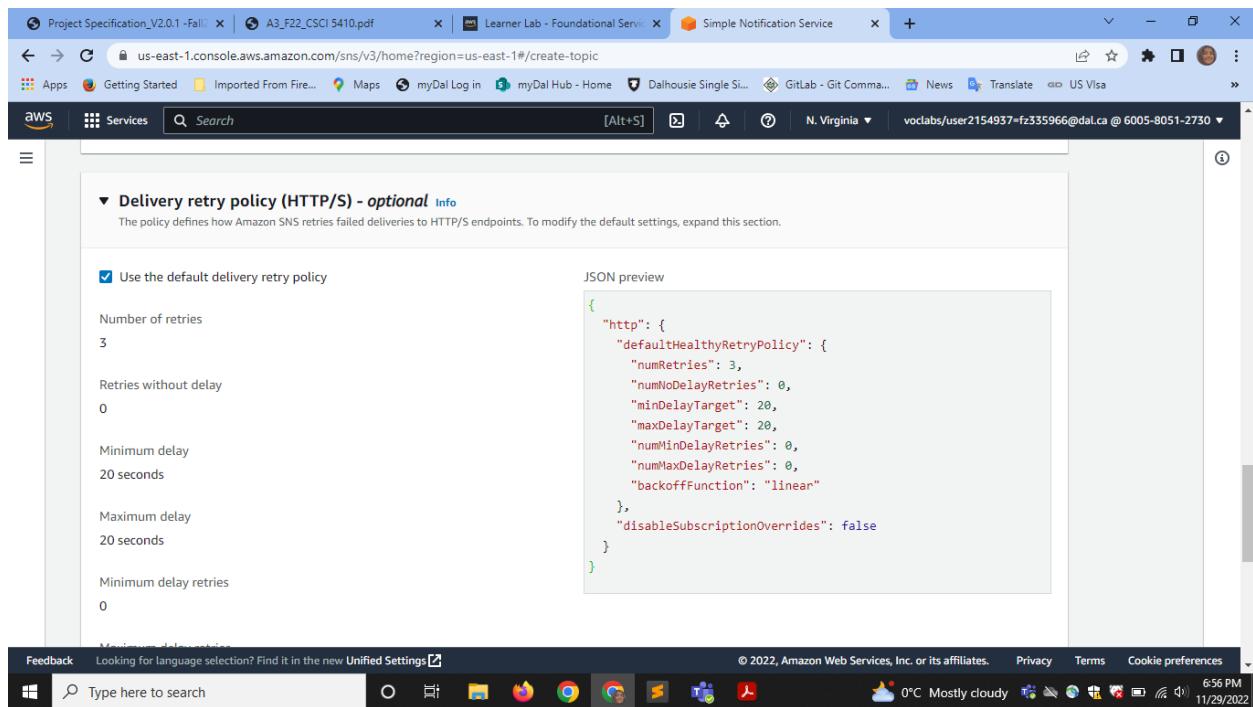


Fig 28: “Create topic” form

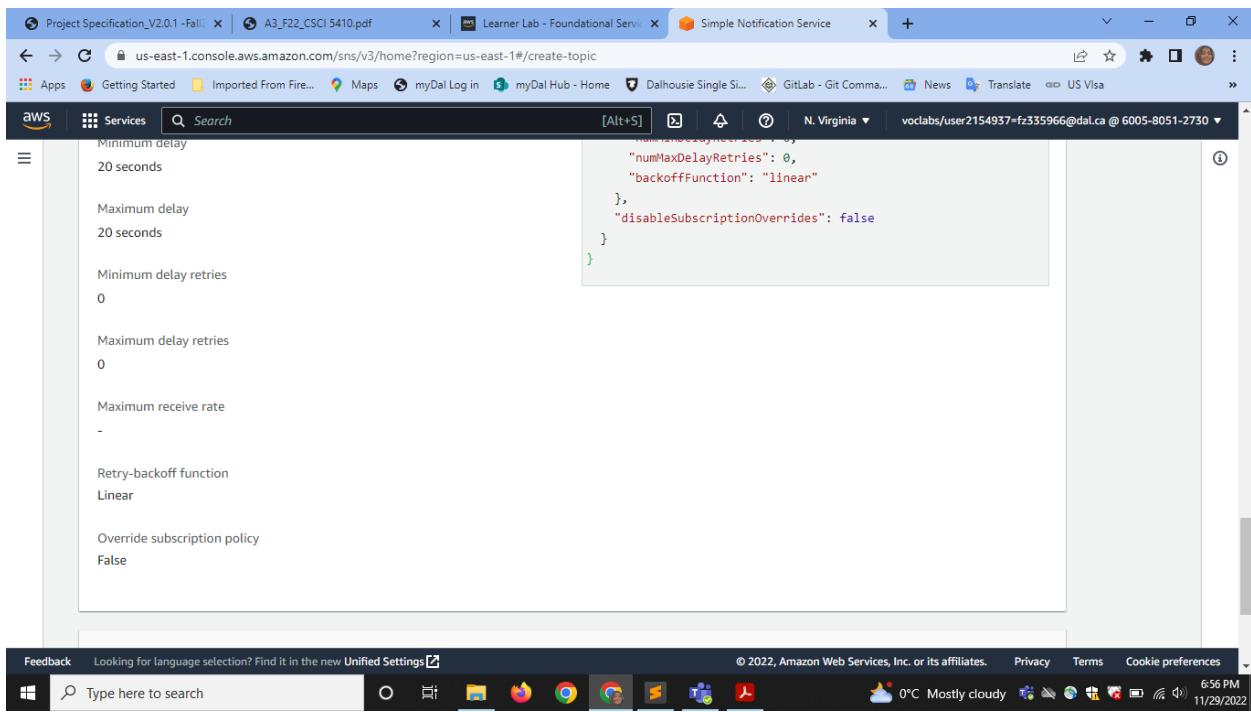


Fig 29: “Create topic” form

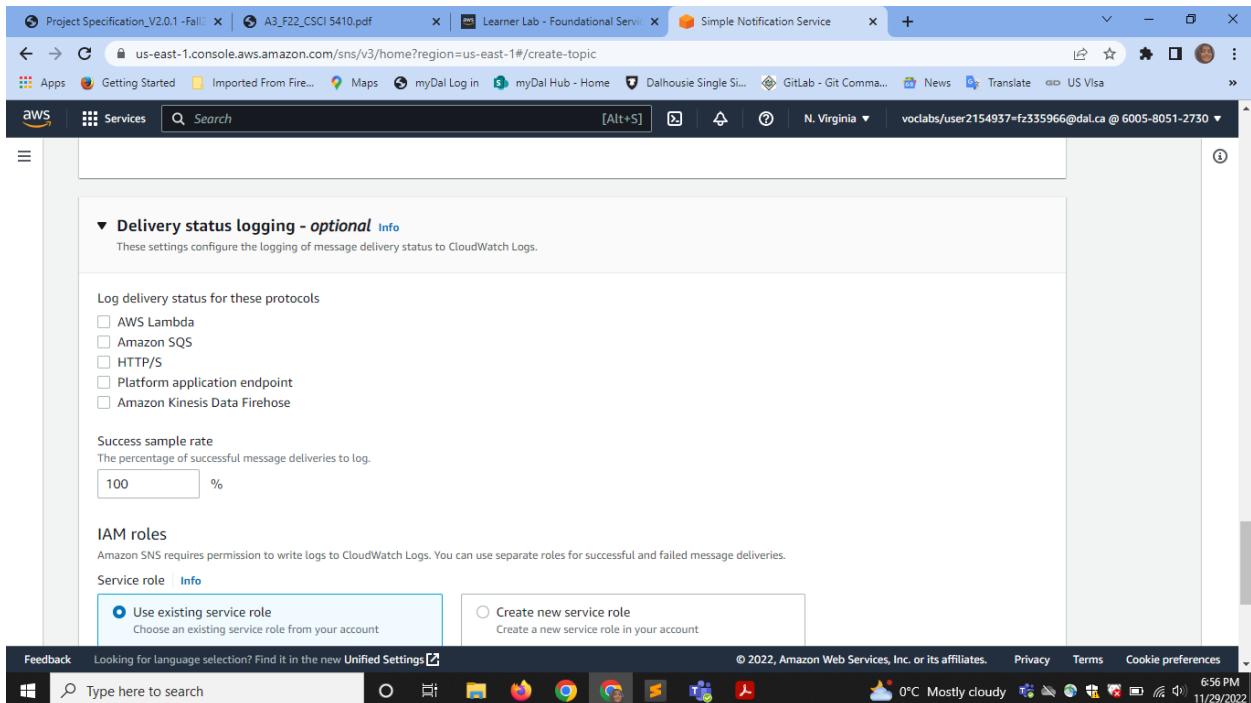


Fig 30: “Create topic” form

The screenshot shows the AWS Simple Notification Service (SNS) 'Create topic' form. At the top, there are two radio button options for IAM roles: 'Use existing service role' (selected) and 'Create new service role'. Below these are two input fields for ARNs: 'IAM role for successful deliveries' and 'IAM role for failed deliveries'. A section for 'Tags - optional' follows, containing a table for adding key-value pairs. At the bottom right are 'Cancel' and 'Create topic' buttons.

IAM roles
Amazon SNS requires permission to write logs to CloudWatch Logs. You can use separate roles for successful and failed message deliveries.

Service role | Info

Use existing service role
Choose an existing service role from your account

Create new service role
Create a new service role in your account

IAM role for successful deliveries
Enter a role ARN

IAM role for failed deliveries
Enter a role ARN

▼ Tags - optional

A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

Key	Value - optional
<input type="text"/> Enter key	<input type="text"/> Enter value
Add tag	

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

6:56 PM 11/29/2022

Fig 31: “Create topic” form

This screenshot is identical to Fig 31, showing the 'Create topic' form. The 'Create topic' button at the bottom right is highlighted in orange, indicating it is the primary action to be taken.

IAM roles
Amazon SNS requires permission to write logs to CloudWatch Logs. You can use separate roles for successful and failed message deliveries.

Service role | Info

Use existing service role
Choose an existing service role from your account

Create new service role
Create a new service role in your account

IAM role for successful deliveries
Enter a role ARN

IAM role for failed deliveries
Enter a role ARN

▼ Tags - optional

A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. [Learn more](#)

Key	Value - optional
<input type="text"/> Enter key	<input type="text"/> Enter value
Add tag	

Cancel **Create topic**

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

6:56 PM 11/29/2022

Fig 32: “Create topic” form

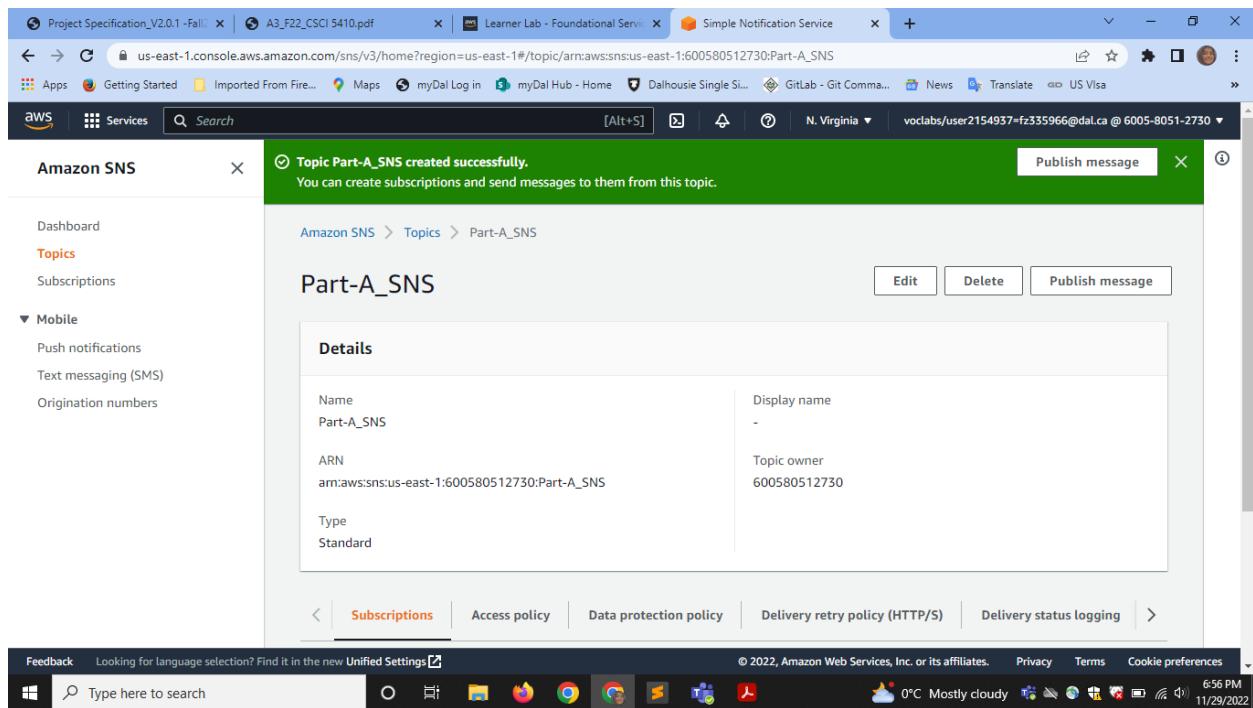


Fig 33: Topics “Part-A_SNS” has been created

Step 4: Create subscription

- Click on the “Create subscription” button as shown on the page as shown in the fig 34.
- Fill the details in the “Create subscription” form as shown in the fig 35-37.
- The subscription “Part-A_SNS” has been created and its details has been mentioned as shown in the fig 38 and 39.
- The subscription is created to Part-A_SNS as shown in the fig 38. The status for the subscription is currently showing “pending confirmation” as shown in the fig 39.
- Now check the email which you provided while creating subscription (Refer fig 36 and 40).
- Confirm the subscription you received on email by AWS Notifications as shown in the fig 40.
- Go to Subscription page AWS to see the status changes from pending to confirmed on confirming the subscription through email notifications from AWS (Refer fig 42).

The screenshot shows the AWS Simple Notification Service (SNS) interface. On the left, there's a sidebar with links like 'Dashboard', 'Topics' (which is selected), and 'Subscriptions'. Below that is a 'Mobile' section with 'Push notifications', 'Text messaging (SMS)', and 'Origination numbers'. The main content area has tabs for 'Subscriptions' (selected), 'Access policy', 'Data protection policy', 'Delivery retry policy (HTTP/S)', and 'Delivery status logging'. Under 'Subscriptions', it says '(0)' and has buttons for 'Edit', 'Delete', 'Request confirmation', 'Confirm subscription', and 'Create subscription'. A search bar is above this. Below the search bar, it says 'No subscriptions found' and 'You don't have any subscriptions to this topic.' There's also a 'Create subscription' button. At the bottom of the page, there's a feedback link, copyright information (© 2022, Amazon Web Services, Inc. or its affiliates.), and a navigation bar with icons for Windows, search, and system status.

Fig 34: “Create subscription” page

The screenshot shows the 'Create subscription' form. At the top, there's a warning message: 'Important changes for sending text messages (SMS) to US destinations' with a note about mobile carriers changing regulations. To the right is a 'View origination numbers' button. Below the warning, the path 'Amazon SNS > Subscriptions > Create subscription' is shown. The main form has a 'Details' section with fields for 'Topic ARN' (set to 'arn:aws:sns:us-east-1:600580512730:Part-A_SNS'), 'Protocol' (set to 'Email'), and 'Endpoint' (an email address 'fz335966@dal.ca'). At the bottom, there's a feedback link, copyright information (© 2022, Amazon Web Services, Inc. or its affiliates.), and a navigation bar with icons for Windows, search, and system status.

Fig 35: “Create subscription” form

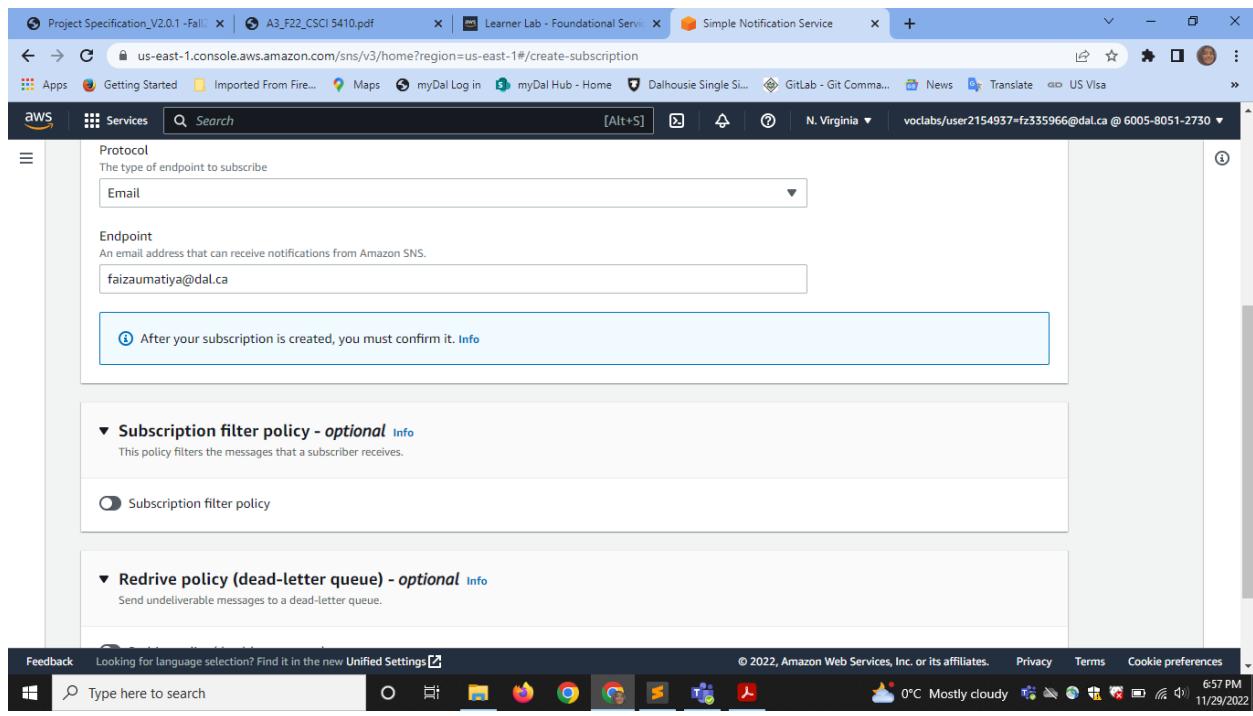


Fig 36: “Create subscription” form

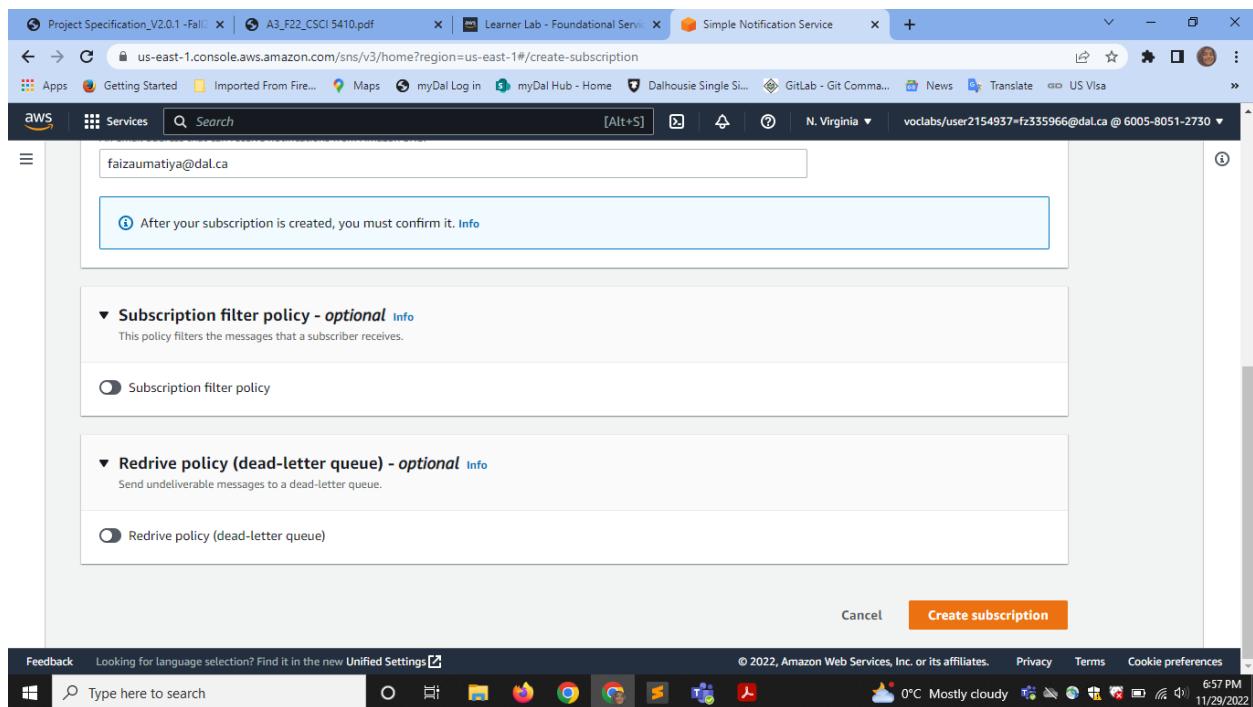


Fig 37: “Create subscription” form

The screenshot shows the AWS SNS service interface. On the left, a sidebar lists options like Dashboard, Topics, Subscriptions (which is selected), and Mobile. The main content area displays a success message: "Subscription to Part-A_SNS created successfully." It includes the ARN of the subscription: "arn:aws:sns:us-east-1:600580512730:Part-A_SNS:7fae9797-d593-4680-9492-4de06ead4d11". Below this, a "Details" section shows the ARN, Status (Pending confirmation), Protocol (EMAIL), Endpoint (faizamatya@dal.ca), and Topic (Part-A_SNS). A warning message at the top right states: "Important changes for sending text messages (SMS) to US destinations. US mobile carriers have recently changed their regulations, and will require that all toll-free numbers (TFNs) complete a registration process with a regulatory body before September 30, 2022. If you currently have a toll-free number you must register your toll-free number by September 30, 2022 or you will no longer be able to use the toll-free number. Learn more." A "View origination numbers" button is also present. The bottom of the screen shows a Windows taskbar with various icons and the date/time: "6:57 PM 11/29/2022".

Fig 38: Subscription created successfully page

This screenshot is identical to Fig 38, but the status of the subscription is now "Pending confirmation" instead of "Pending confirmation". The rest of the interface, including the details section and the warning message, remains the same.

Fig 39: Subscription created successfully page

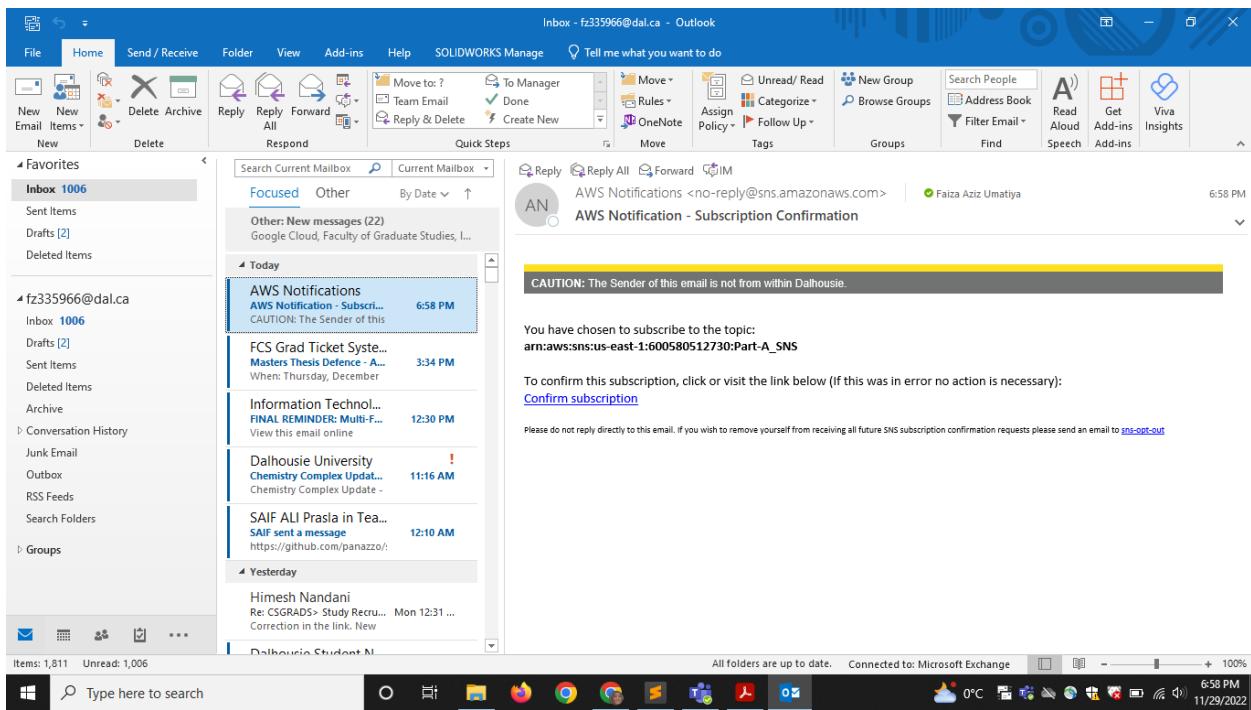


Fig 40: AWS notifications email for confirmation page

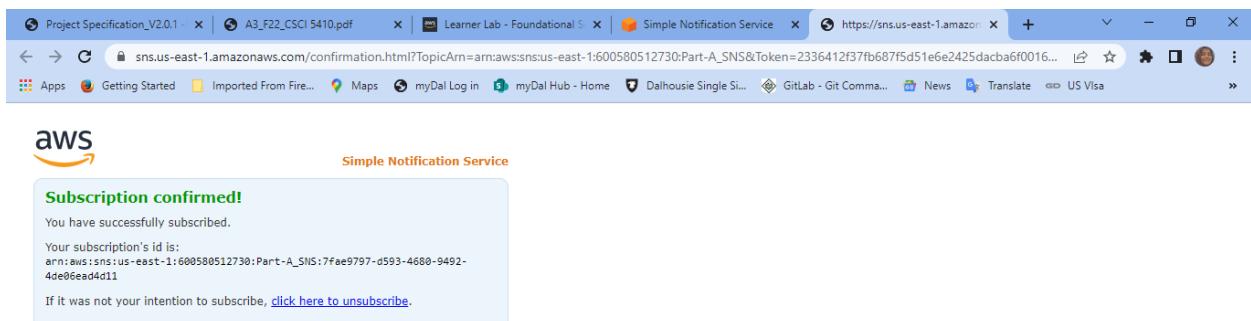


Fig 41: AWS subscription confirmation page

The screenshot shows the AWS SNS Subscriptions page. On the left, there's a sidebar with links like Dashboard, Topics, and Subscriptions. The main area has a header 'Amazon SNS > Subscriptions' and a table titled 'Subscriptions (1)'. The table has columns for ID, Endpoint, Status, Protocol, and Topic. One row is shown: '7fae9797-d593-46...' with 'faizaumatiya@dal.ca' as the endpoint, 'Confirmed' status, 'EMAIL' protocol, and 'Part-A_SNS' topic. There are buttons for Edit, Delete, Request confirmation, Confirm subscription, and Create subscription.

ID	Endpoint	Status	Protocol	Topic
7fae9797-d593-46...	faizaumatiya@dal.ca	Confirmed	EMAIL	Part-A_SNS

Fig 42: AWS subscription status confirmed

Step 5: Create a lambda function

- Go to search bar of the AWS console (Refer step 1 to get here) and search for AWS Lambda as shown in the fig 43.
- Fill the details to create lambda function as shown in the fig 44-47.
- Fig 48 shows lambda function “SQS_SNS” is created.
- Fig 49, 50, 51 and 51.1 shows the lambda function.

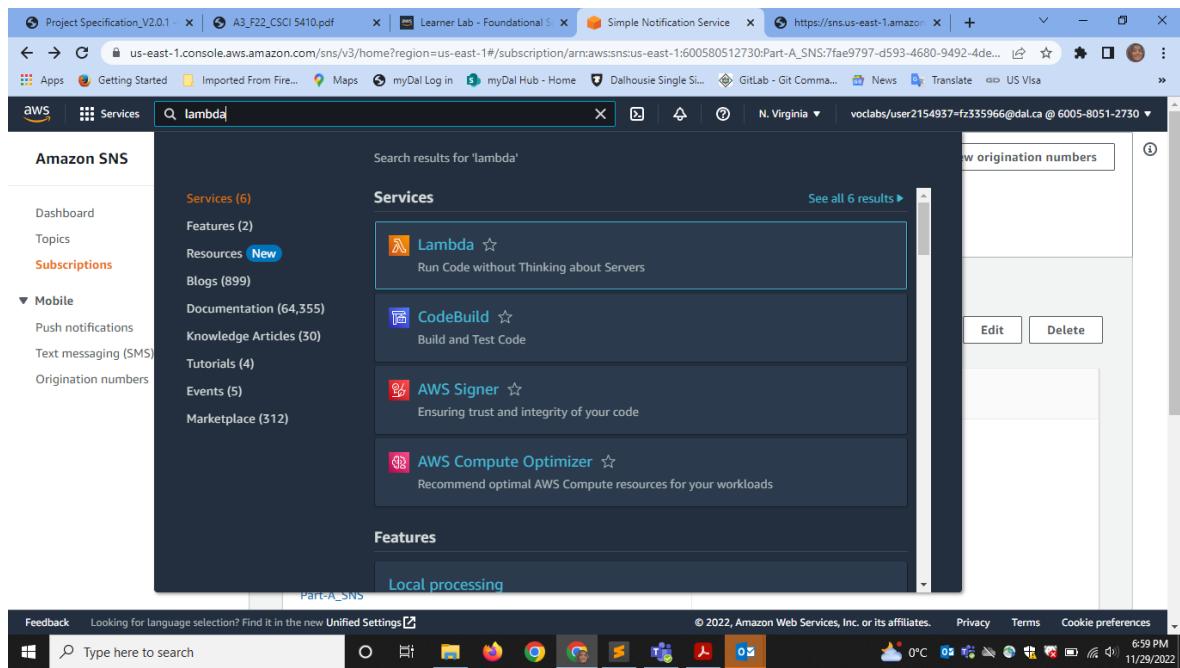


Fig 43: Searching for AWS lambda

The screenshot shows the AWS Lambda Functions page. The left sidebar lists navigation options such as Dashboard, Applications, Functions (which is selected and highlighted in orange), Additional resources, and Related AWS resources. The main content area displays a table titled 'Functions (4)'. The table includes columns for Function name, Description, Package type, Runtime, and Last modified. The listed functions are:

Function name	Description	Package type	Runtime	Last modified
LightsailMonitoringFunction	-	Zip	Python 3.8	2 months ago
verifyStudentIdentity	-	Zip	Python 3.9	19 days ago
ratingFoodOrderFunction	-	Zip	Python 3.9	1 hour ago
MainMonitoringFunction	-	Zip	Python 3.8	2 months ago

Fig 44: Create function page

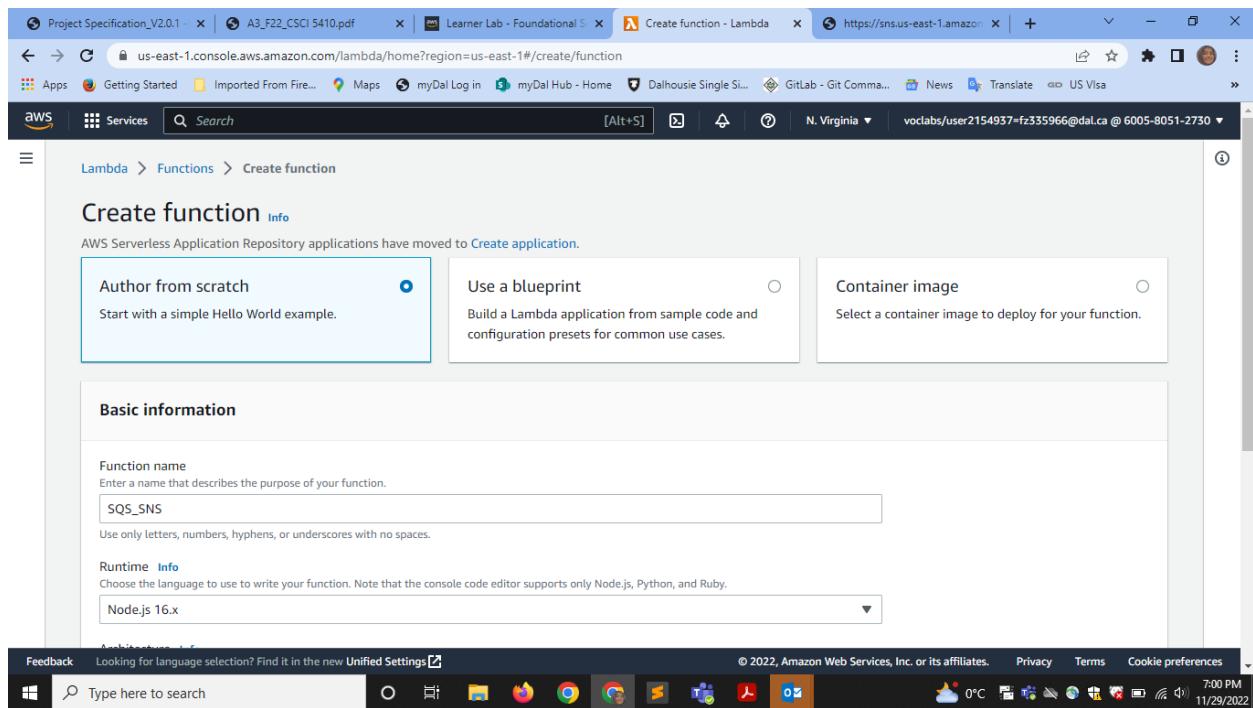


Fig 45: Create function form

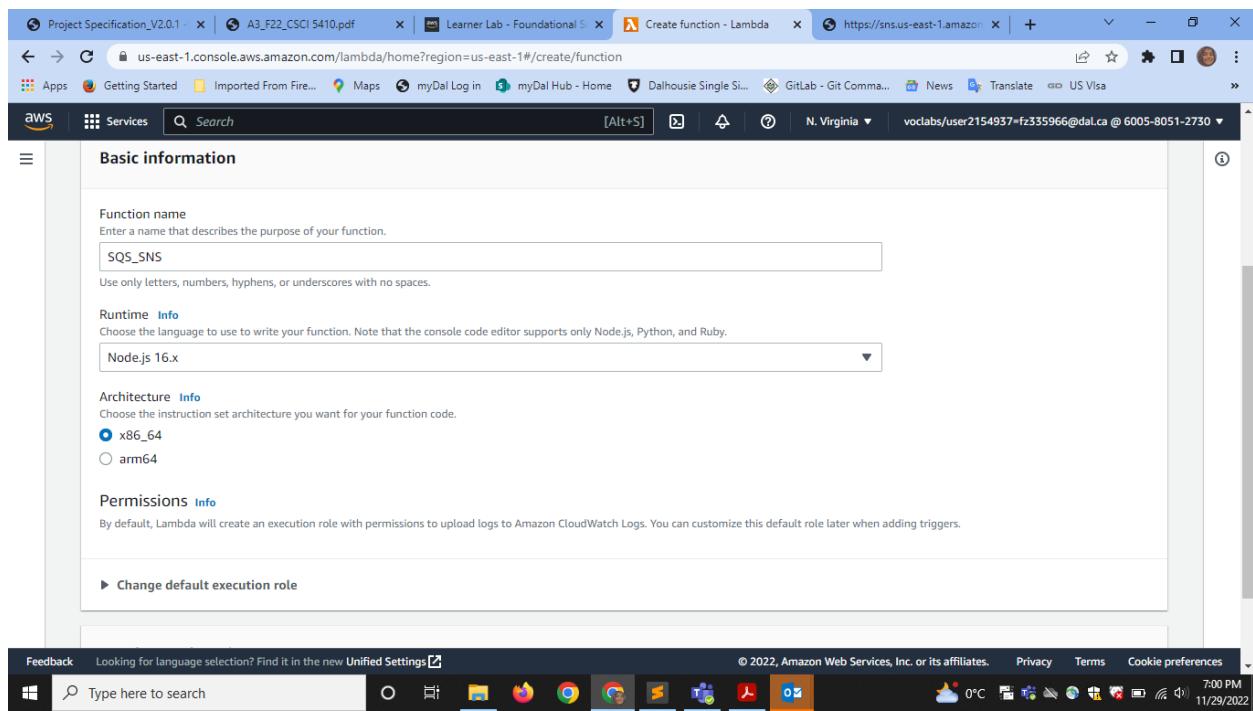


Fig 46: Create function form

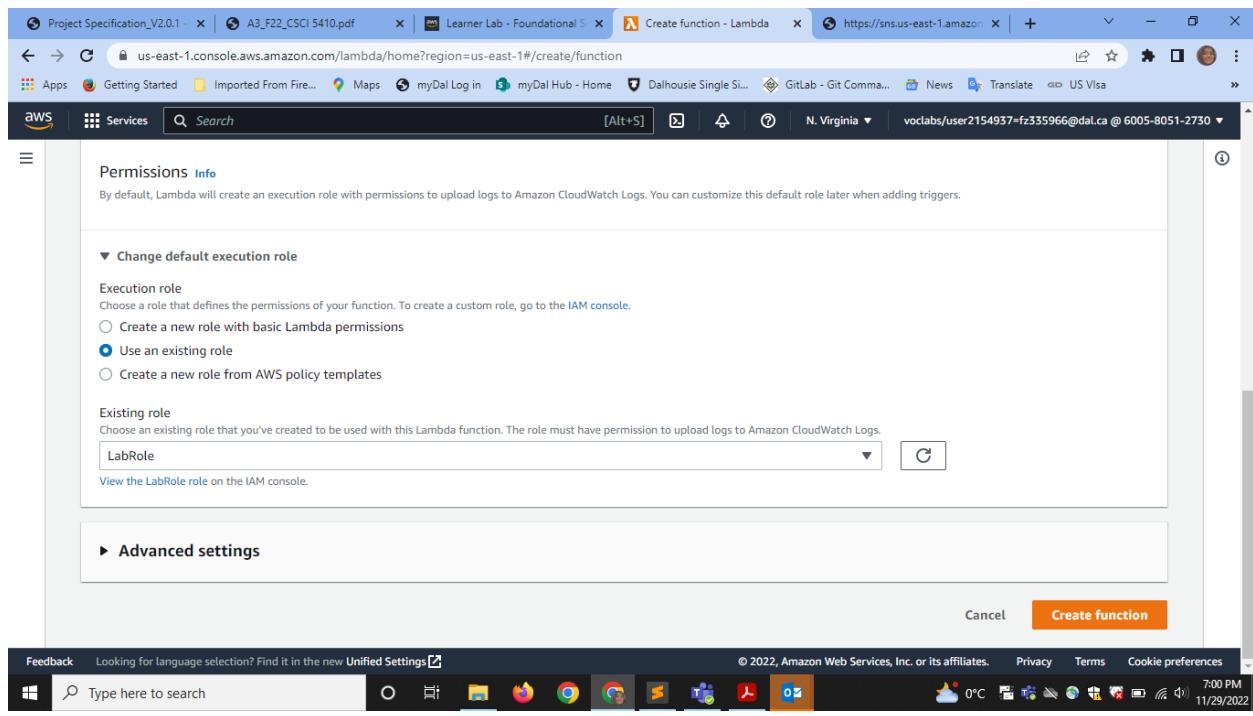


Fig 47: Create function form

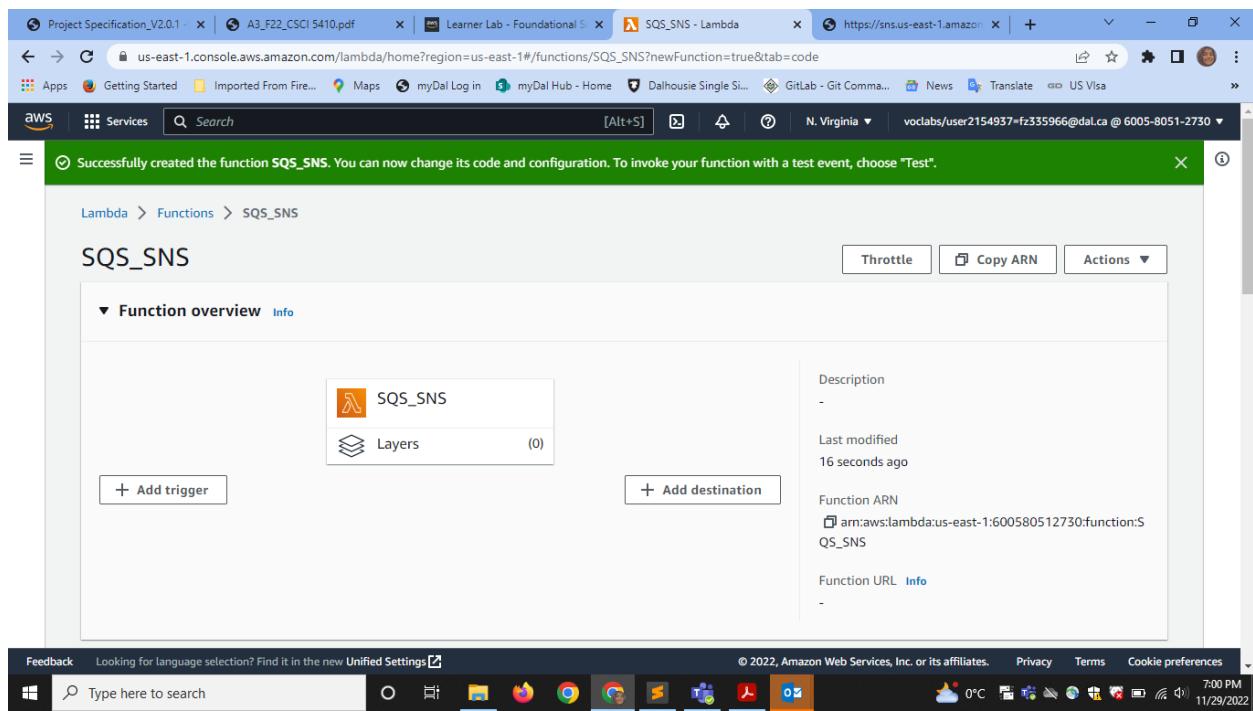


Fig 48: Lambda function "SQS_SNS" created

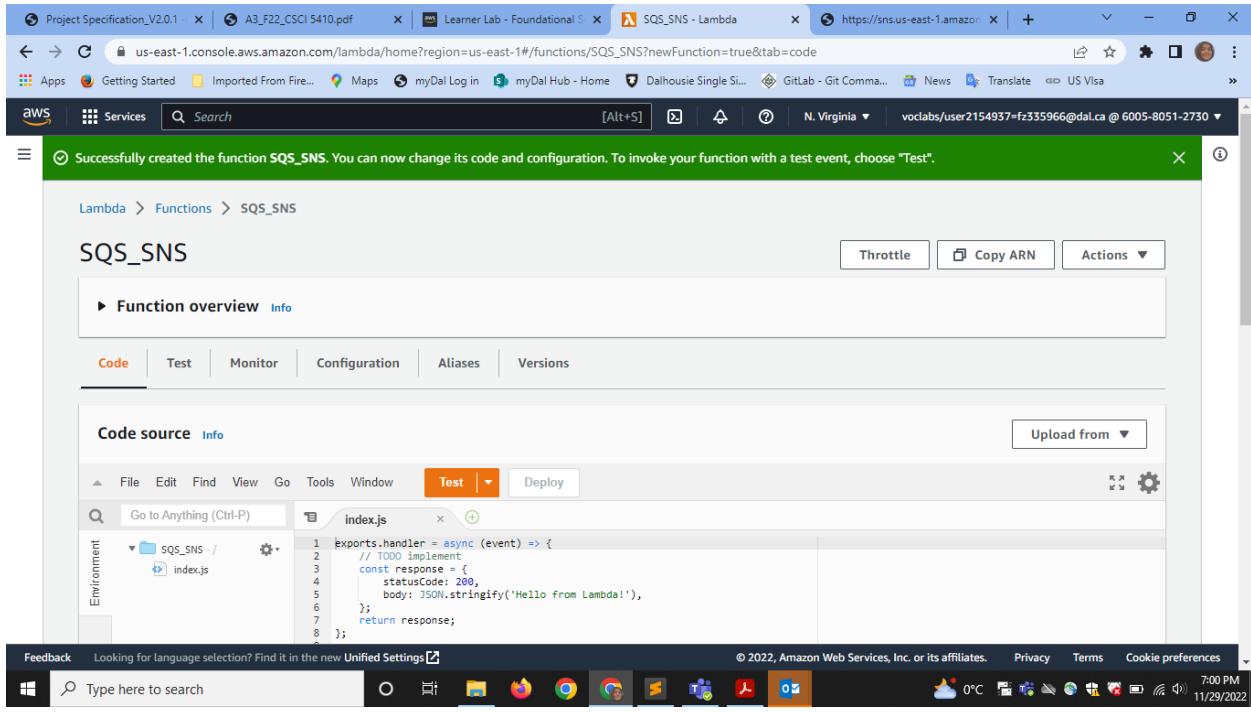


Fig 49: Lambda function “SQS_SNS” created

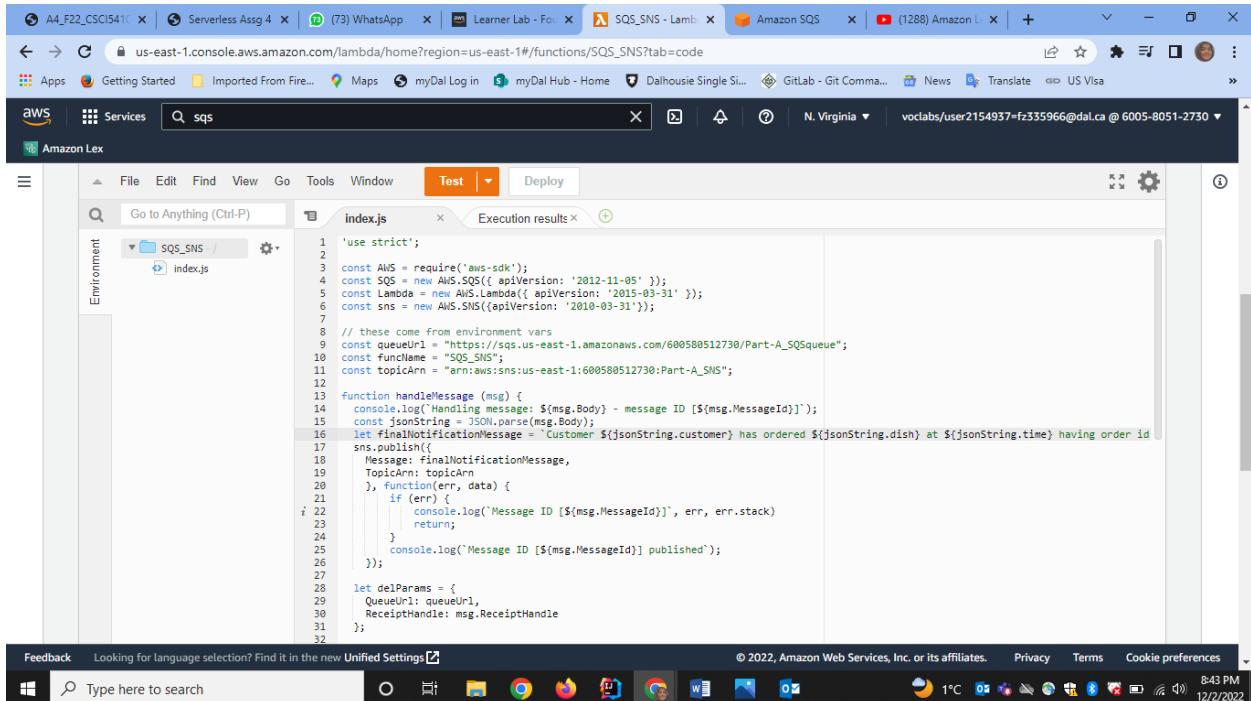


Fig 50: Lambda function

The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes tabs for 'A4_F22_CSCI5410', 'Serverless Assg 4', '(73) WhatApp', 'Learner Lab - F...', 'SQS_SNS - Lamb...', 'Amazon SQS', and '(1288) Amazon L...'. The search bar at the top right contains the query 'sq...'. The main workspace displays the 'index.js' file under the 'SQS_SNS /' environment. The code is as follows:

```
31  };
32
33   return SQS
34   .deleteMessage(delParams)
35   .promise()
36   .then(() => console.log(`Message ID [${msg.MessageId}] deleted`))
37   .catch(err => console.log(`Message ID [${msg.MessageId}]`, err, err.stack));
38 }
39
40 function recurse () {
41   let params = {
42     FunctionName: funcName,
43     InvokeArgs: "{}"
44   };
45
46   return Lambda
47   .invokeAsync(params)
48   .promise()
49   .then((data) => console.log("Lambda Function recursed"));
50 }
51
52 module.exports.handler = function(event, context) {
53   let params = {
54     QueueUrl1 : queueUrl1,
55     MaxNumberOfMessages : 10,
56     VisibilityTimeout : 6,
57     WaitTimeSeconds : 20
58   };
59
60   SQS
61   .receiveMessage(params)
62 }
```

Below the code editor, there's a 'Feedback' section and a search bar. The bottom status bar shows the date and time: '© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 8:44 PM 12/2/2022'.

Fig 51: Lambda function

This screenshot is identical to Fig 51, showing the AWS Lambda function editor with the same code in the 'index.js' file under the 'SQS_SNS /' environment. The code is identical to what was shown in Fig 51.

Fig 51.1: Lambda function

Step 6: To add trigger for receiving messaging at some time interval.

- To add trigger, click on the “Add trigger” option as shown in the fig 48.
- Fill in the details like adding rule name and schedule expression as shown in the fig 52 and 53.
- Click on “Add trigger” as shown in the fig 53.
- The EventBridge trigger has been added as shown in the fig 54.
- On clicking the “test” (Refer fig 60), I was prompt to “configure test event page” as shown in the fig 54.1 and 54.2.
- Fill the details as shown in the fig 54.1 and 54.2.

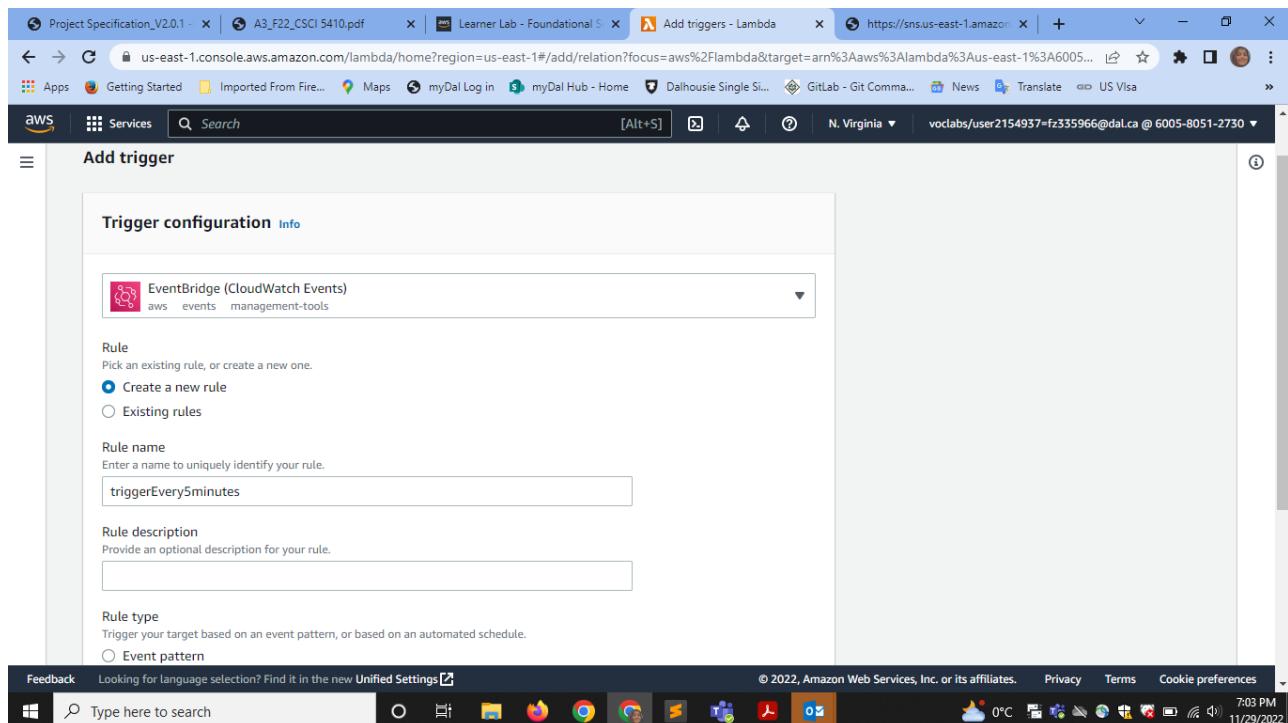


Fig 52: “Trigger configuration” form

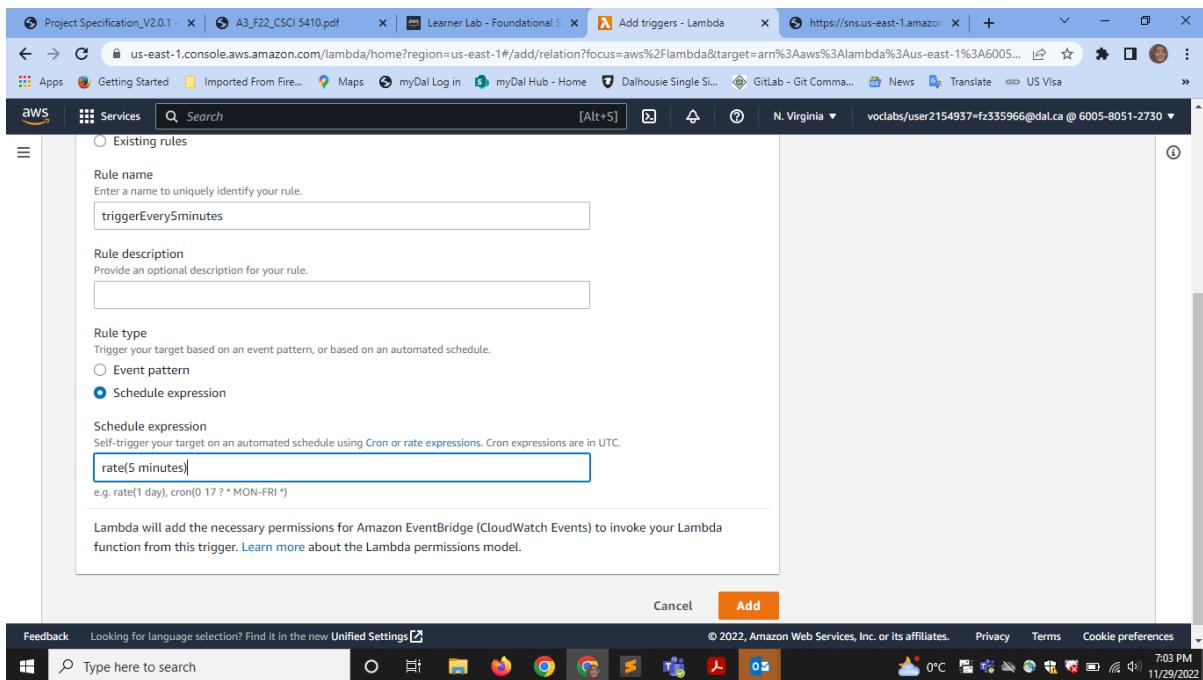


Fig 53: “Trigger configuration” form

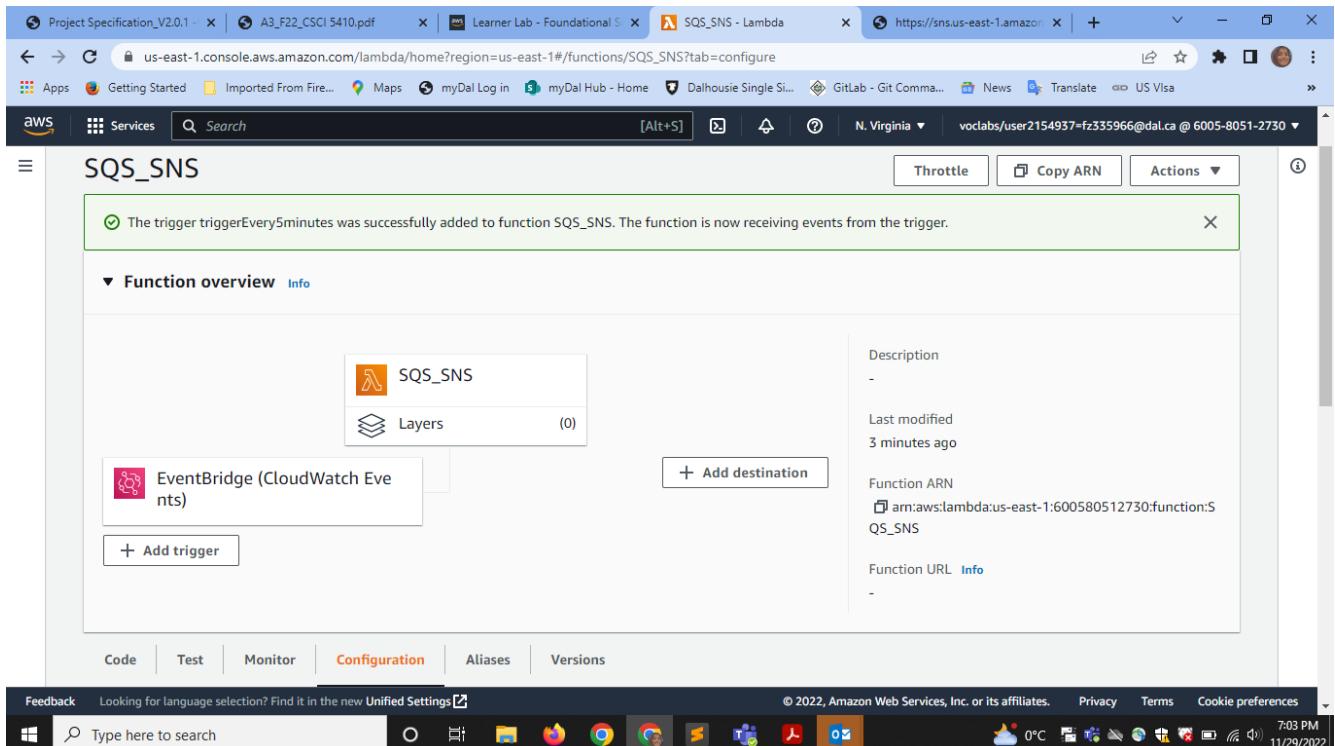


Fig 54: “EventBridge” trigger added

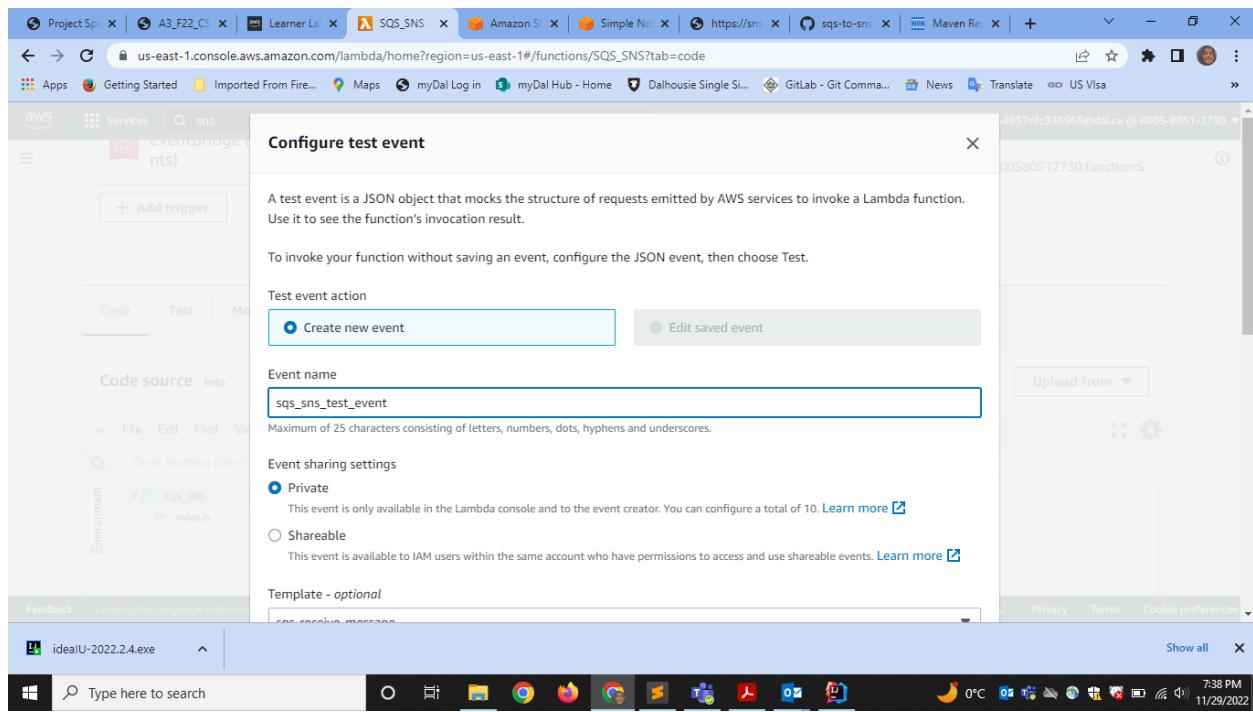


Fig 54.1: Configure test event page

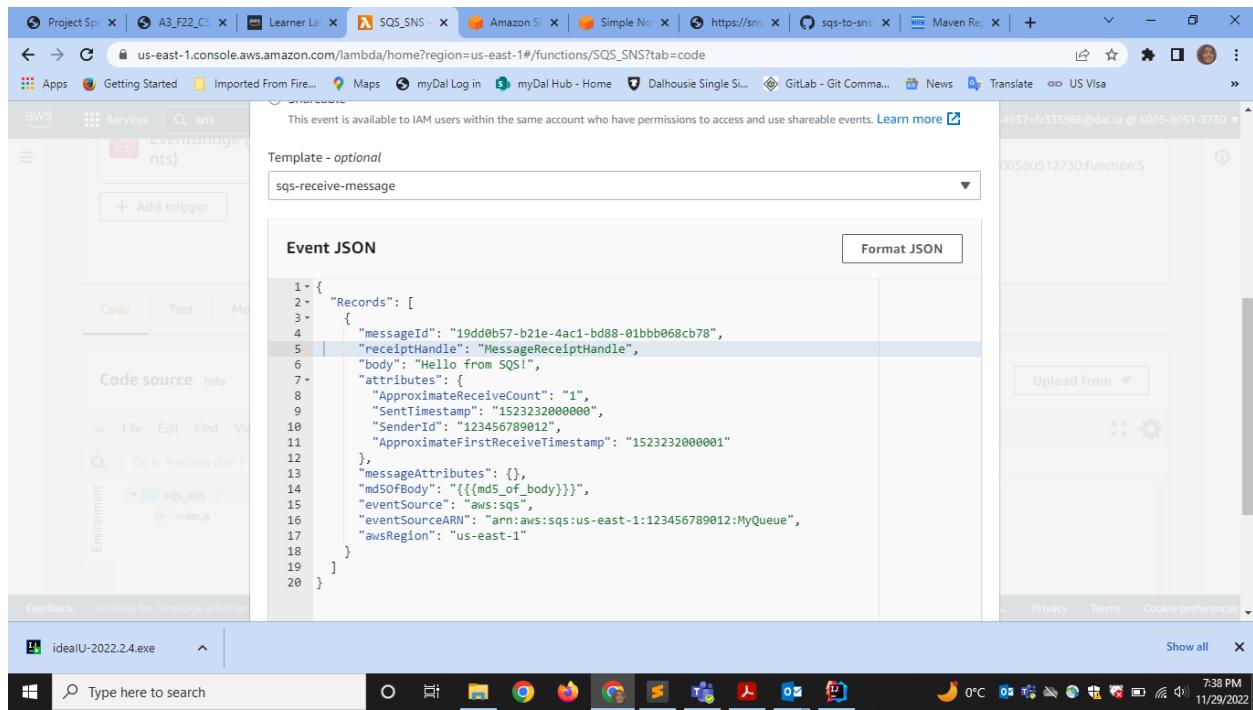
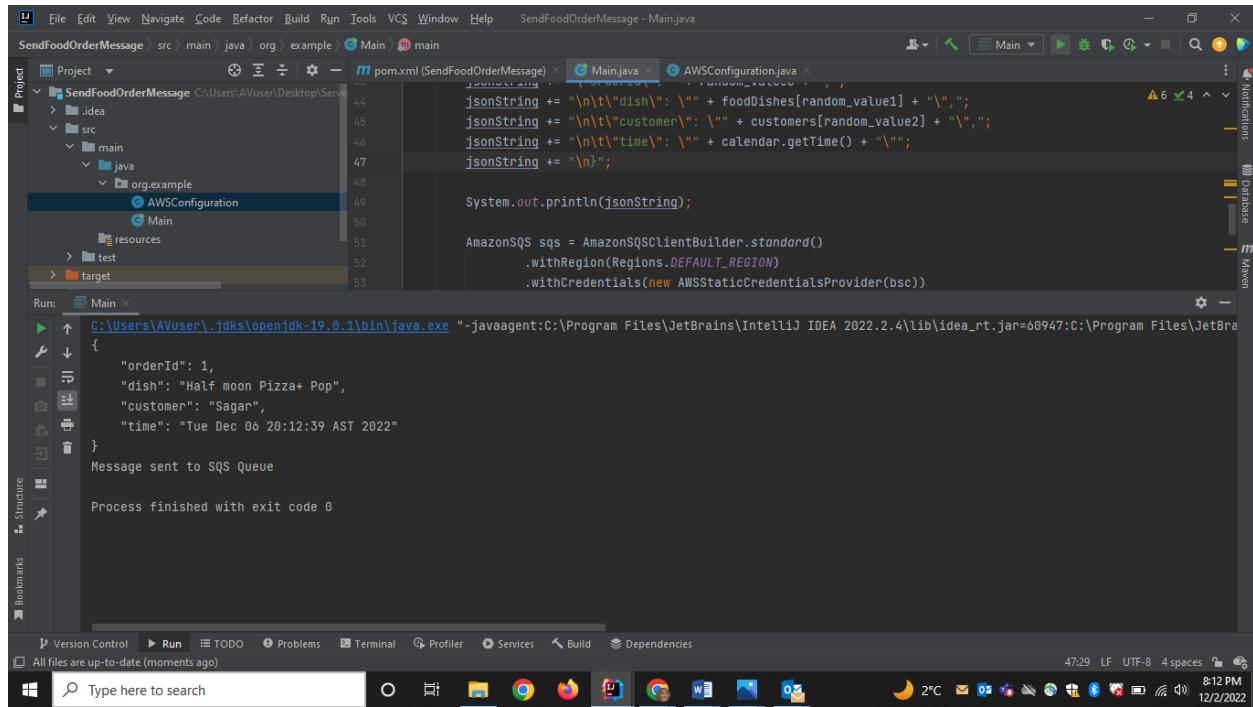


Fig 54.2: Template for Event JSON page

Step 7: Code Execution

- Write the java code and run the code as shown in the fig 55.
- Go to “Part-A_SQSqueue” (Refer fig 56) and click on “send and receive message” option as shown in the fig 56.
- To check whether we received message or not which is been sent, click on “Polling message” option which is shown in fig 57 and 58.
- On clicking the “polling message”, it shows the status of message receiving as shown in fig 58.
- After receiving the message (Refer fig 58) click on the message and you will see the actual json format message which is shown in fig 59.
- Go to the lambda function and test and deploy to get the message on your provided email as shown in the fig 60.
- Fig 61, shows that the message is been received after every interval of minutes with the order message i.e. previous mail was at 8.10pm and the next mail was at 8.15 pm.



The screenshot shows the IntelliJ IDEA interface during code execution. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, and the current file name, SendFoodOrderMessage - Main.java. The Project tool window on the left shows the project structure with a src folder containing main, java, org.example, AWSConfiguration, and Main classes, along with pom.xml and resources. The Run tool window at the bottom shows the command used to run the application: C:\Users\AVUser\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.4\lib\idea_rt.jar=60947:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.4\bin". The terminal output pane displays the following text:
{"orderId": 1,
 "dish": "Half moon Pizza+ Pop",
 "customer": "Sagar",
 "time": "Tue Dec 06 20:12:39 AST 2022"}
Message sent to SQS Queue
Process finished with exit code 0

Fig 55: Terminal represents message sent to SQS

The screenshot shows the AWS SQS console with the URL https://us-east-1.console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues/https%3A%2F%2Fsqs.us-east-1.amazonaws.com%2F600580512730%2FPart-A_SQSqueue. The page displays the details of the 'Part-A_SQSqueue'. Key information includes:

- Name:** Part-A_SQSqueue
- Type:** Standard
- ARN:** arn:aws:sqs:us-east-1:600580512730:Part-A_SQSqueue
- Encryption:** Amazon SNS key (SSE-SQS)
- URL copied:** https://sqs.us-east-1.amazonaws.com/600580512730/Part-A_SQSqueue
- Dead-letter queue:** -

Below the details, there are tabs for SNS subscriptions, Lambda triggers, Dead-letter queue, Monitoring, Tagging, Access policy, Encryption, and Dead-letter queue redrive tasks. The browser status bar at the bottom shows the date and time as 7:36 PM 12/2/2022.

Fig 56: Part-A_SQSqueue page

The screenshot shows the AWS SQS console with the URL https://us-east-1.console.aws.amazon.com/sqs/v2/home?region=us-east-1#/queues/https%3A%2F%2Fsqs.us-east-1.amazonaws.com%2F600580512730%2FPart-A_SQSqueue/sendReceiveMessages. The page is titled "Send and receive messages" and contains the following fields:

- Send message Info:** A text area labeled "Enter message" with a "Clear content" button and a "Send message" button.
- Delivery delay Info:** A field where "0" is selected in a dropdown menu next to "Seconds". A note states "Should be between 0 seconds and 15 minutes."
- Message attributes - Optional Info:** A link to optional message attributes.

Below the form, there is a feedback message: "Feedback Looking for language selection? Find it in the new Unified Settings". The browser status bar at the bottom shows the date and time as 7:36 PM 12/2/2022.

Fig 57: Send and receive message page

The screenshot shows the AWS Lambda function configuration page. At the top, there are tabs for 'Lambda' and 'SQS'. Below the tabs, the 'Handler' section is visible, showing the path `lambda_function.lambda_handler`. The 'Code' section indicates the code is deployed using the AWS Lambda Python runtime. The 'Environment' section shows environment variables like `AWS_LAMBDA_FUNCTION_NAME`, `AWS_LAMBDA_FUNCTION_MEMORY_SIZE`, and `AWS_LAMBDA_FUNCTION_TIMEOUT`. The 'Logs' section shows log entries for the function's execution.

Fig 58: Send and receive message page – Message received

The screenshot shows the AWS Lambda function configuration page. At the top, there are tabs for 'Lambda' and 'SQS'. Below the tabs, the 'Handler' section is visible, showing the path `lambda_function.lambda_handler`. The 'Code' section indicates the code is deployed using the AWS Lambda Python runtime. The 'Environment' section shows environment variables like `AWS_LAMBDA_FUNCTION_NAME`, `AWS_LAMBDA_FUNCTION_MEMORY_SIZE`, and `AWS_LAMBDA_FUNCTION_TIMEOUT`. The 'Logs' section shows log entries for the function's execution.

Fig 59: Send and receive message page – Message received

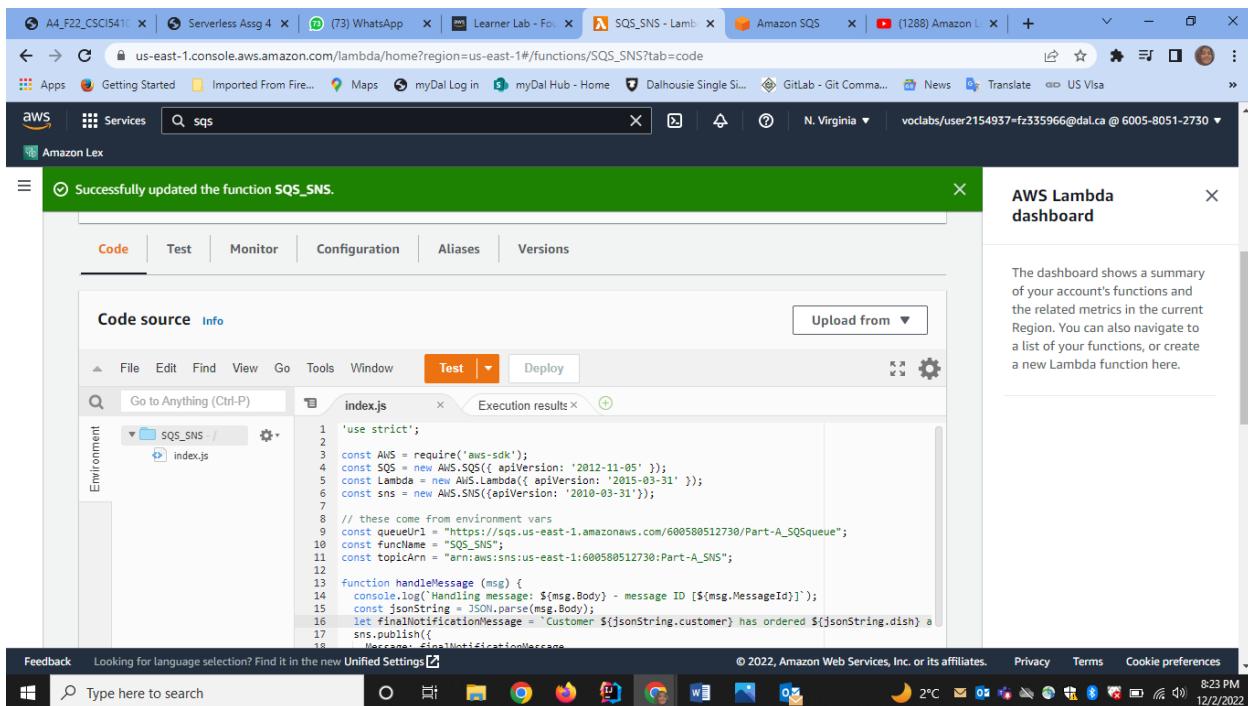


Fig 60: Test and deploy Lambda Function

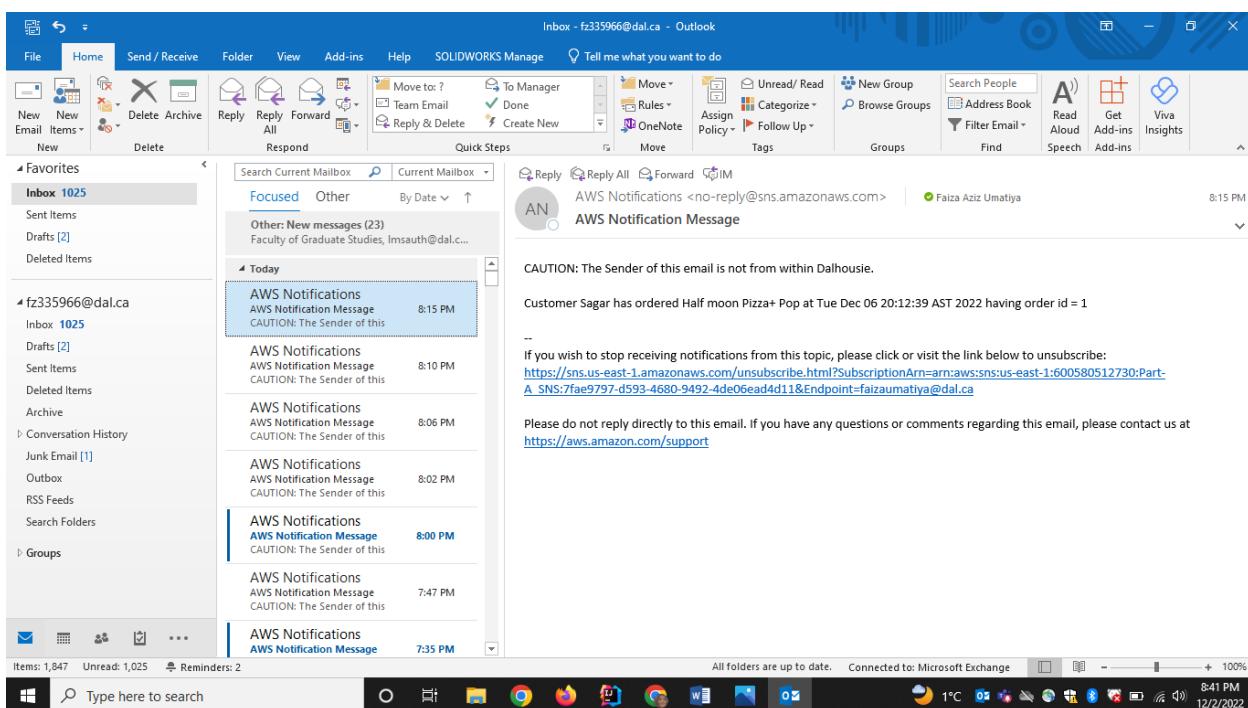


Fig 61: AWS notification having customer order message

Code Screenshots:

Main.java:

```
package org.example;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.SendMessageRequest;
import com.amazonaws.services.sqs.model.SendMessageResult;
import nscapes.javascript.JSONObject;
import java.util.Calendar;
import java.util.Date;

public class Main {
    public static void main(String[] args) {
        AWSConfiguration awsConfiguration = new AWSConfiguration();
        // AWS credentials
        BasicSessionCredentials bsc = new BasicSessionCredentials(
            awsConfiguration.AWS_ACCESS_KEY,
            awsConfiguration.AWS_SECRET_KEY,
            awsConfiguration.AWS_SESSION_TOKEN
        );
        // making String array of food dishes and customers
        String[] foodDishes = {"Pizza + Pop", "Pasta + Pop", "Pizza + Pasta", "Half moon Pizza+ Pop"};
        String[] customers = {"Faiza", "Sagar", "Swinkhal", "Ariya", "Parul"};
    }
}
```

Fig 62: Main.java

```
// making String array of food dishes and customers
String[] foodDishes= {"Pizza + Pop", "Pasta + Pop", "Pizza + Pasta", "Half moon Pizza+ Pop"};
String[] customers = {"Faiza", "Sagar", "Swinkhal", "Ariya", "Parul"};

// Generating random values

/* Reference:
[1]"How to generate random numbers in Java," Eduative: Interactive Courses for
Software Developers. [Online]. Available: https://www.educative.io/answers/how-to-
generate-random-numbers-in-java. [Accessed: 03-Dec-2022].
*/

int random_value1 = (int) Math.floor(Math.random()*(4-0+1)+0);
int random_value2 = (int) Math.floor(Math.random()*(4-0+1)+0);
int random_value3 = (int) Math.floor(Math.random()*(4-0+1)+0);

/*Reference:
[2]GeeksforGeeks, "Calendar getInstance() Method in Java with Examples", GeeksforGeeks [Online].
Available: https://www.geeksforgeeks.org/calendar-getinstance-method-in-java-with-examples/
[Accessed: 02-Dec-2022].
*/

Calendar calendar = Calendar.getInstance();
calendar.setTime(new Date());
calendar.add(Calendar.DATE, amount (int) (Math.random()*(4-0+1)+1));
```

Fig 63: Main.java

The screenshot shows the IntelliJ IDEA interface with the Main.java file open. The code generates a JSON string representing a food order and sends it to an AWS SQS queue using the AmazonSQSClientBuilder.

```
calendar.setTime(new Date());
calendar.add(Calendar.DATE, amount: (int) (Math.random()*(4-0+1)+0)+1);

//Creating Json string
String jsonString= "\n\t";
jsonString += "\n\t\"orderId\": " + random_value3 + ",";
jsonString += "\n\t\"dish\": \"\" + foodDishes[random_value1] + "\"";
jsonString += "\n\t\"customer\": \"\" + customers[random_value2] + "\"";
jsonString += "\n\t\"time\": \"\" + calendar.getTime() + "\"";
jsonString += "\n}";

System.out.println(jsonString);

/* Reference:
[3]Amazon Web Services, Inc., [Online].
Available: https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/sqs/AmazonSQS.html
[Accessed: 03-Dec-2022].
*/

AmazonSQS sqs = AmazonSQSClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .withCredentials(new AWSStaticCredentialsProvider(bsc))
    .build();

/*Reference:
[4]Amazon Web Services, Inc., "Sending, Receiving, and Deleting Amazon SQS Messages", Amazon Web Services,
2022 [Online]. Available: https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/examples-sqs-messages.html
[Accessed: 01-Dec-2022].
*/
```

Fig 64: Main.java

The screenshot shows the IntelliJ IDEA interface with the Main.java file open. The code continues from the previous snippet, creating a SendMessageRequest, setting its queue URL and message body to the generated JSON string, and then sending the message to the SQS queue.

```
AmazonSQS sqs = AmazonSQSClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .withCredentials(new AWSStaticCredentialsProvider(bsc))
    .build();

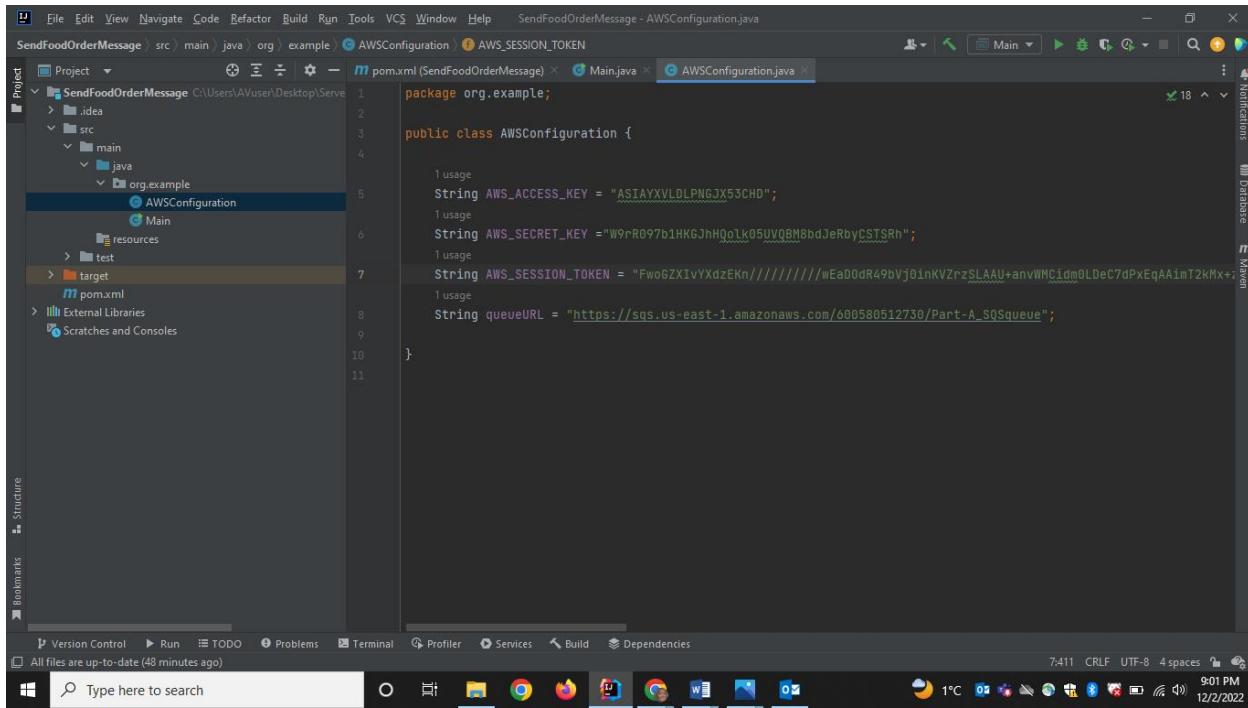
/*Reference:
[4]Amazon Web Services, Inc., "Sending, Receiving, and Deleting Amazon SQS Messages", Amazon Web Services,
2022 [Online]. Available: https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/examples-sqs-messages.html
[Accessed: 01-Dec-2022].
*/
```

```
SendMessageRequest messageRequest = new SendMessageRequest()
    .withQueueUrl(awsConfiguration.queueURL)
    .withMessageBody(jsonString)
    .withDelaySeconds(5);

SendMessageResult messageResult = sqs.sendMessage(messageRequest);
System.out.println("Message sent to SQS Queue");
```

Fig 65: Main.java

AWSConfiguration.java:



The screenshot shows the IntelliJ IDEA interface with the AWSConfiguration.java file open. The code defines a class AWSConfiguration with static fields for AWS access key, secret key, session token, and queue URL. The code is annotated with usage counts for each variable.

```
package org.example;

public class AWSConfiguration {

    String AWS_ACCESS_KEY = "ASIAJXVLDLPGJX53CHD";
    String AWS_SECRET_KEY = "W9rR097b1HKGJhH0olk05UV0BM8bdJeRbyCSTSrh";
    String AWS_SESSION_TOKEN = "FwoGZXIvYXdzEKn//////////wEa00dR49bVj0inVZrzSLAAU+anvWMcidm0LDeC7dPxEqAAimT2kMx+";
    String queueURL = "https://sqs.us-east-1.amazonaws.com/600580512730/Part-A_SQSqueue";
}
```

Fig 66: AWSConfiguration.java

Code Script:

I created 2 java class:

Main.java:

```
package org.example;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.SendMessageRequest;
import com.amazonaws.services.sqs.model.SendMessageResult;
import netscape.javascript.JSONObject;

import java.util.Calendar;
import java.util.Date;

public class Main {
    public static void main(String[] args) {

        AWSConfiguration awsConfiguration = new AWSConfiguration();
        // AWS credentials
        BasicSessionCredentials bsc = new BasicSessionCredentials(

```

```

awsConfiguration.AWS_ACCESS_KEY,
awsConfiguration.AWS_SECRET_KEY,
awsConfiguration.AWS_SESSION_TOKEN
);
// making String array of food dishes and customers
String[] foodDishes= {"Pizza + Pop", "Pasta + Pop", "Pizza + Pasta", "Half moon Pizza+ Pop"};
String[] customers = {"Faiza", "Sagar", "Swinkhal", "Ariya", "Parul"};

// Generating random values

/* Reference:
[1]“How to generate random numbers in Java,” Eduative: Interactive Courses for
Software Developers. [Online]. Available: https://www.educative.io/answers/how-to-generate-random-numbers-in-java. [Accessed: 03-Dec-2022].
*/
int random_value1 = (int) Math.floor(Math.random()*(4-0+1)+0);
int random_value2 =(int) Math.floor(Math.random()*(4-0+1)+0);
int random_value3 = (int) Math.floor(Math.random()*(4-0+1)+0);

/*Reference:
[2]GeeksforGeeks, “Calendar getInstance() Method in Java with Examples”, GeeksforGeeks [Online].
Available: https://www.geeksforgeeks.org/calendar-getinstance-method-in-java-with-examples/
[Accessed: 02-Dec-2022].
*/
Calendar calendar = Calendar.getInstance();
calendar.setTime(new Date());
calendar.add(Calendar.DATE, (int) (Math.random()*(4-0+1)+0)+1);

//Creating Json string
String jsonString="{\n\t";
jsonString += "\"orderId\": " + random_value3 + ",";
jsonString += "\n\t\"dish\": \"\""+ foodDishes[random_value1] + "\",";
jsonString += "\n\t\"customer\": \"\""+ customers[random_value2] + "\",";
jsonString += "\n\t\"time\": \"\""+ calendar.getTime() + "\"";
jsonString += "\n}";

System.out.println(jsonString);

/* Reference:
[3]Amazon Web Services, Inc., [Online].
Available:
https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/sqs/AmazonSQSClientBuilder.html.
[Accessed: 03-Dec-2022].
*/
AmazonSQS sqs = AmazonSQSClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .withCredentials(new AWSStaticCredentialsProvider(bsc))
    .build();

/*Reference:
[4]Amazon Web Services, Inc., “Sending, Receiving, and Deleting Amazon SQS Messages”, Amazon Web

```

```

Services, Inc..
2022 [Online]. Available: https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/examples-sqs-messages.html
[Accessed: 01-Dec-2022].
*/
SendMessageRequest messageRequest = new SendMessageRequest()
    .withQueueUrl(awsConfiguration.queueURL)
    .withMessageBody(jsonString)
    .withDelaySeconds(5);

SendMessageResult messageResult = sqs.sendMessage(messageRequest);
System.out.println("Message sent to SQS Queue");

}
}

```

AWSConfiguration.java:

```

package org.example;

public class AWSConfiguration {

    String AWS_ACCESS_KEY = "ASIAYXVLDLPNGJX53CHD";
    String AWS_SECRET_KEY = "W9rR097b1HKGJhHQolk05UVQBM8bdJeRbyCSTSrh";
    String AWS_SESSION_TOKEN =
    "FwoGZXIvYXdzEKn//////////wEaDOdR49bVj0inKVZrzSLAAU+anvWMCidm0LDeC7dPxEqAAimT2kMx+zeB
    YSZB0wYcMMf82CUqOOp4h41UGBhD/URriDiOYiLf/NDz0+30kJCHazXmrOGBTLilJ+eNjnD6D/Ec1faMN
    Y3j6nr9TNLGSIulHG1mY4eDxexPOk1SPbHxguepTGVU5IHL4StXDWqxcc+02XG4CD3No5ZbE1+TZftGhhjW
    2yWfdSsByTd/UDSfUt2FDPf6DvP597U795hpwDjFJq2QvFG4eLKpd5Ci3laqcBjItzfe0fcCL9ibP3o71iCGLSRy0f
    4DEI8EMvU6ZfzLh32e+JVCWCOhAI3KxVhAo";
    String queueURL = "https://sqs.us-east-1.amazonaws.com/600580512730/Part-A_SQSqueue";

}

```

Lambda function: index.js (code reference [5])

```

'use strict';

const AWS = require('aws-sdk');

const SQS = new AWS.SQS({ apiVersion: '2012-11-05' });

const Lambda = new AWS.Lambda({ apiVersion: '2015-03-31' });

const sns = new AWS.SNS({ apiVersion: '2010-03-31' });

// these come from environment vars

const queueUrl = "https://sqs.us-east-1.amazonaws.com/600580512730/Part-A_SQSqueue";

const funcName = "SQS_SNS";

const topicArn = "arn:aws:sns:us-east-1:600580512730:Part-A_SNS";

```

```

function handleMessage (msg) {
  console.log(`Handling message: ${msg.Body} - message ID [${msg.MessageId}]`);
  const jsonString = JSON.parse(msg.Body);
  let finalNotificationMessage = `Customer ${jsonString.customer} has ordered ${jsonString.dish} at ${jsonString.time} having order id = ${jsonString.orderId}`;
  sns.publish({
    Message: finalNotificationMessage,
    TopicArn: topicArn
  }, function(err, data) {
    if (err) {
      console.log(`Message ID [${msg.MessageId}]`, err, err.stack)
      return;
    }
    console.log(`Message ID [${msg.MessageId}] published`);
  });
}

let delParams = {
  QueueUrl: queueUrl,
  ReceiptHandle: msg.ReceiptHandle
};

return SQS
  .deleteMessage(delParams)
  .promise()
  .then(() => console.log(`Message ID [${msg.MessageId}] deleted`))
  .catch(err => console.log(`Message ID [${msg.MessageId}]`, err, err.stack));
}

function recurse () {

```

```
let params = {
    FunctionName: funcName,
    InvokeArgs: "{}"
};

return Lambda
    .invokeAsync(params)
    .promise()
    .then((data) => console.log("Lambda Function recursed"));
}
```

```
module.exports.handler = function(event, context) {
    let params = {
        QueueUrl      : queueUrl,
        MaxNumberOfMessages : 10,
        VisibilityTimeout  : 6,
        WaitTimeSeconds   : 20
    };
}
```

```
SQS
    .receiveMessage(params)
    .promise()
    .then(res => {
        if (res.Messages) {
            return Promise.all(res.Messages.map(handleMessage));
        }
    })
// handle any errors and restore the chain so we always get
// to the next step - which is to recurse
.catch(err => console.log(err, err.stack))
```

```

.then(() => recurse())
.then(() => context.succeed())
// only fail the function if we couldn't recurse, which we
// can then monitor via CloudWatch and trigger
.catch(err => context.fail(err, err.stack));
};


```

References:

- [1] Amazon Web Services, Inc., “Creating an Amazon SNS topic”, Amazon Web Services, Inc., 2022 [Online]. Available: <https://docs.aws.amazon.com/sns/latest/dg/sns-create-topic.html> [Accessed: 01-Dec-2022].
- [2] Amazon Web Services, Inc., “Create a queue (console)”, Amazon Web Services, Inc., 2022 [Online]. Available: <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/step-create-queue.html> [Accessed: 01-Dec-2022].
- [3] Amazon Web Services, Inc., “Sending, Receiving, and Deleting Amazon SQS Messages”, Amazon Web Services, Inc., 2022 [Online]. Available: <https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/examples-sqs-messages.html> [Accessed: 01-Dec-2022].
- [4] GeeksforGeeks, “Calendar getInstance() Method in Java with Examples”, GeeksforGeeks [Online]. Available: <https://www.geeksforgeeks.org/calendar-getinstance-method-in-java-with-examples/> [Accessed: 02-Dec-2022].
- [5] R. P. Roizman, “SQS-TO-SNS-LAMBDA,” GitHub, 2022[Online]. Available: <https://github.com/panazzo/sqs-to-sns-lambda/blob/master/index.js> [Accessed: 01-Dec-2022].

ASSIGNMENT 4 – PART B

I read and understood GCP BigQueryML, which helped me create KMeans cluster of the given dataset

Following are the steps I followed to solve the assignment:

Step 1:

- Open GCP BigQuery homepage as shown in the fig 1.

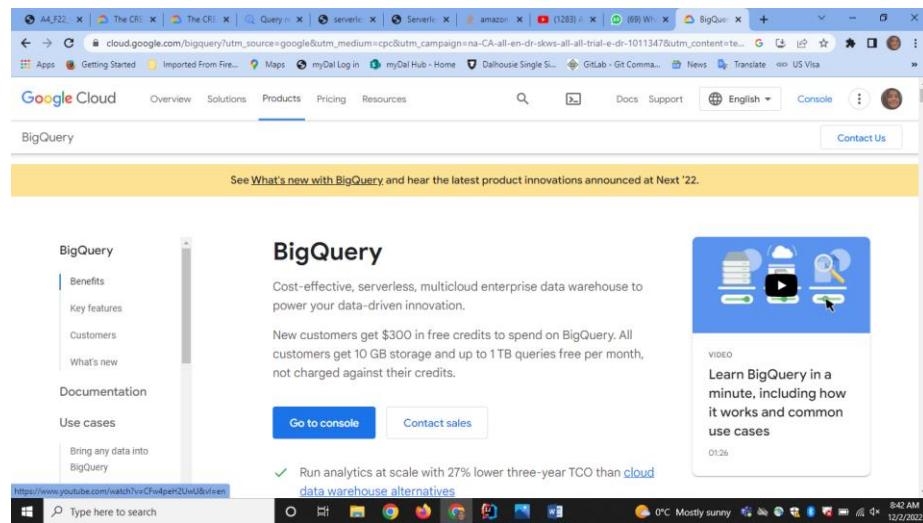


Fig 1: BigQuery home page

Step 2:

- Go to GCP console as shown in the fig 2.

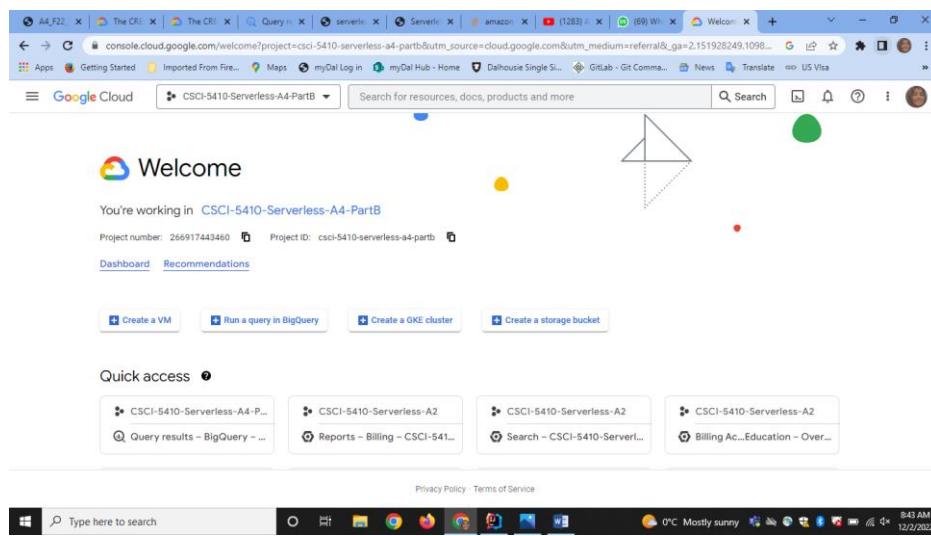


Fig 2: GCP console page

Step 3:

- Create a new project named “CSCI-5410-Serverless-A4-PartB” as shown in the fig 3, 4 & 5.

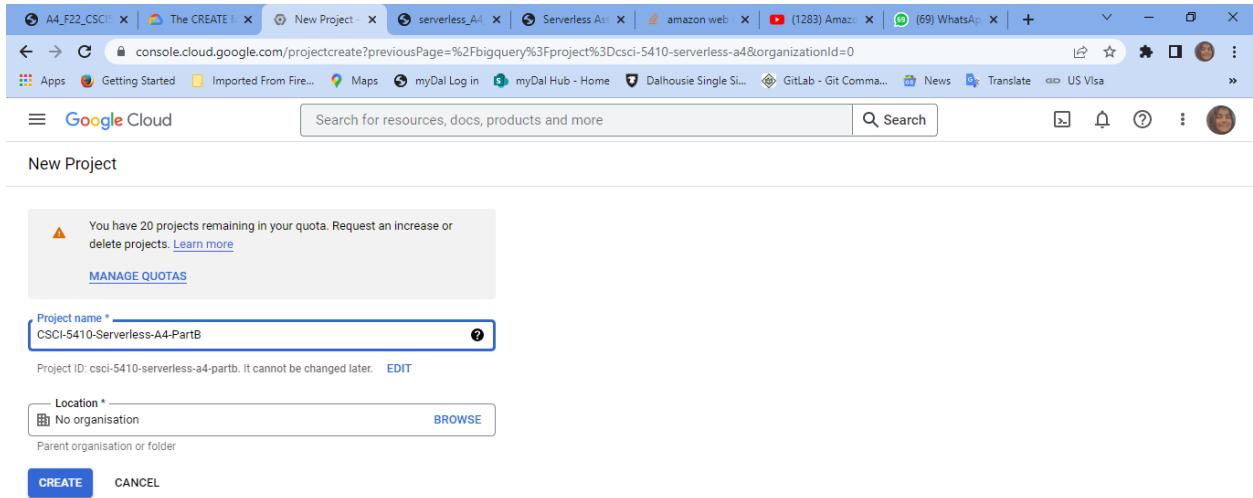


Fig 3: New project creation on GCP console

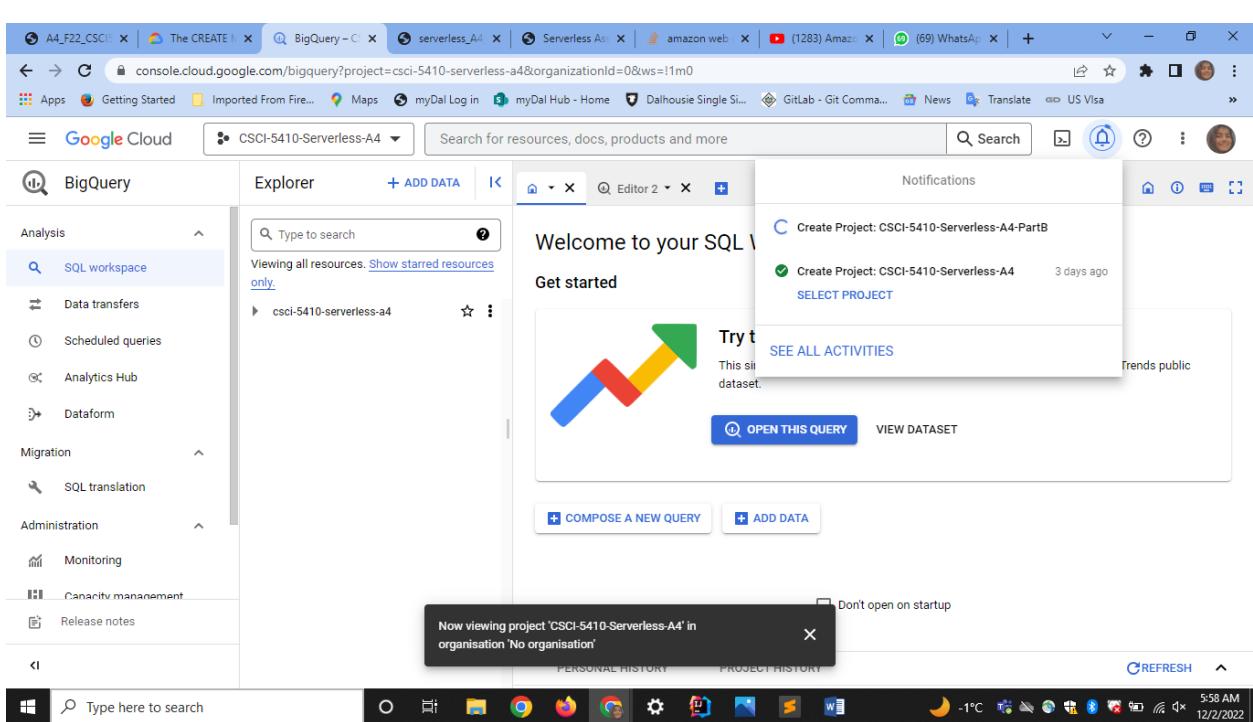


Fig 4: Created new project on GCP console

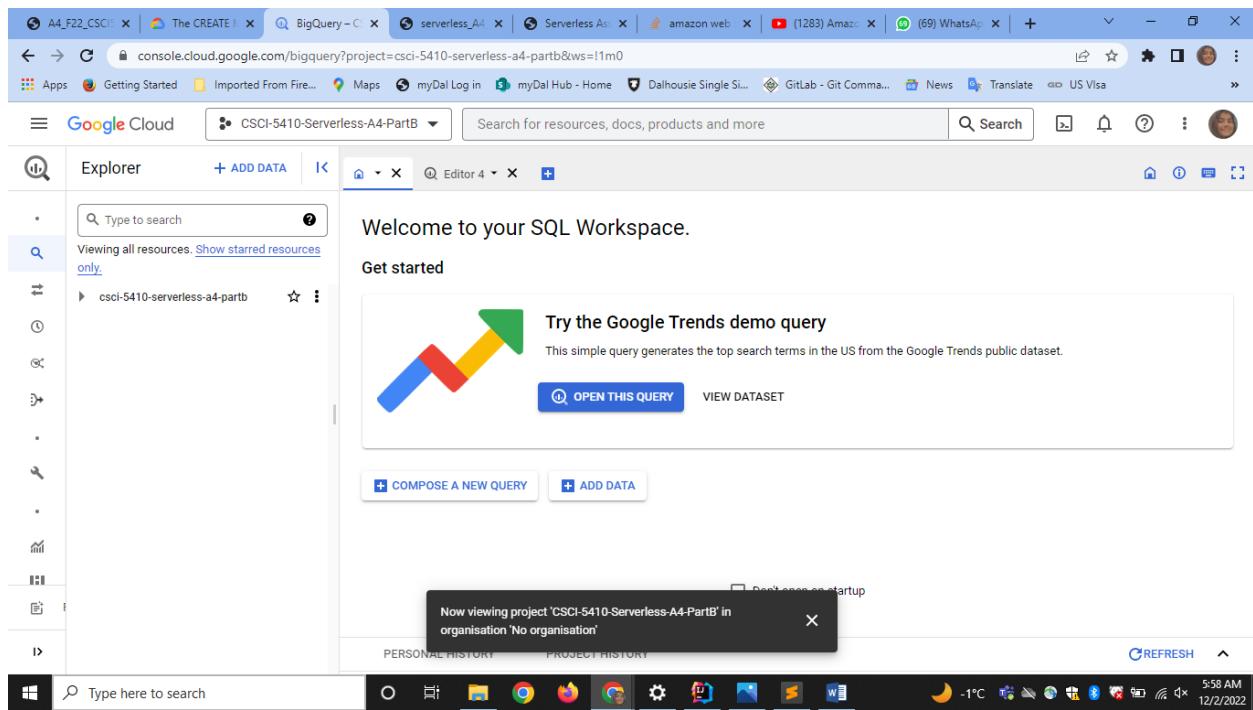


Fig 5: Created new project on GCP console

Step 4:

- Create training dataset as shown in the fig 6, 7. Fig 8 and 9 shows dataset has been created.

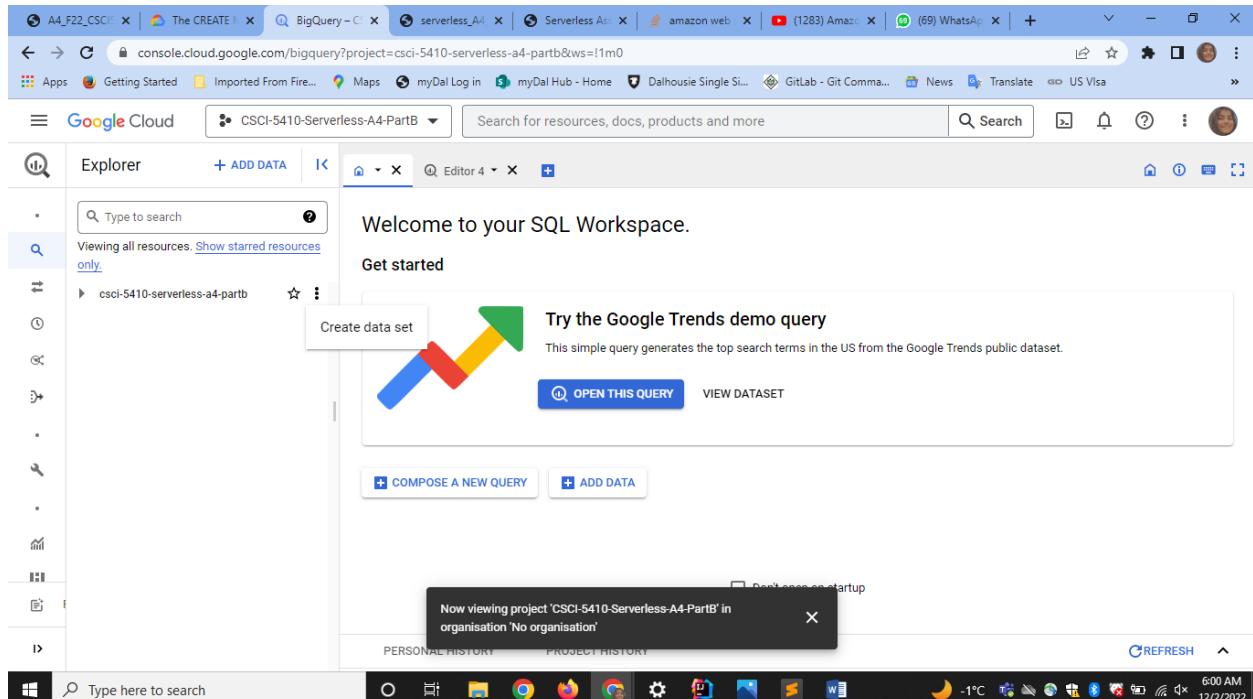


Fig 6: select “create data set” option

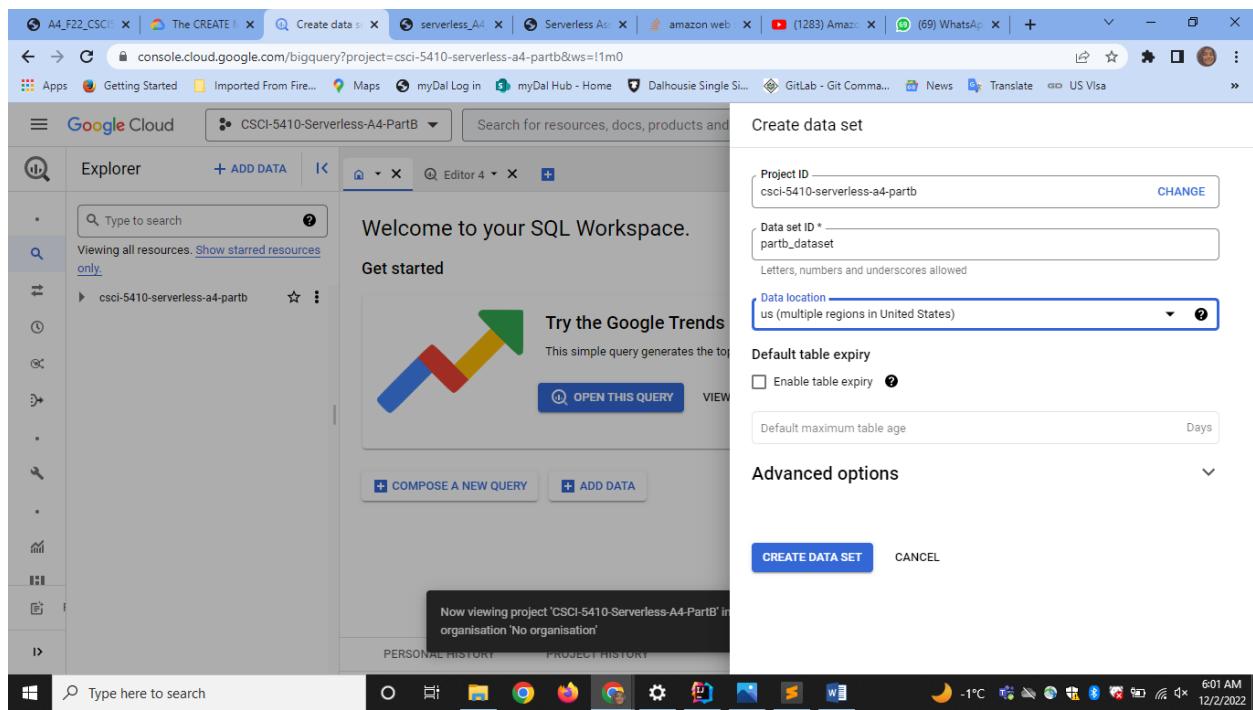


Fig 7: “create data set” form

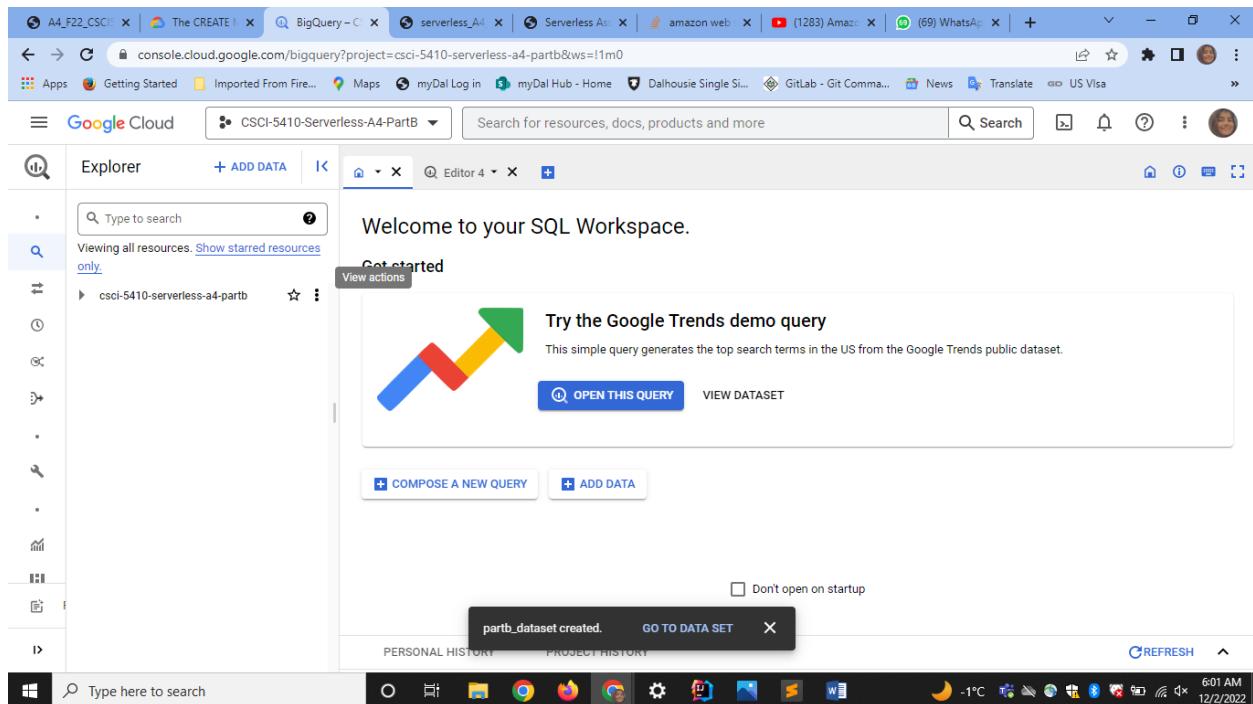


Fig 8: Dataset has been created

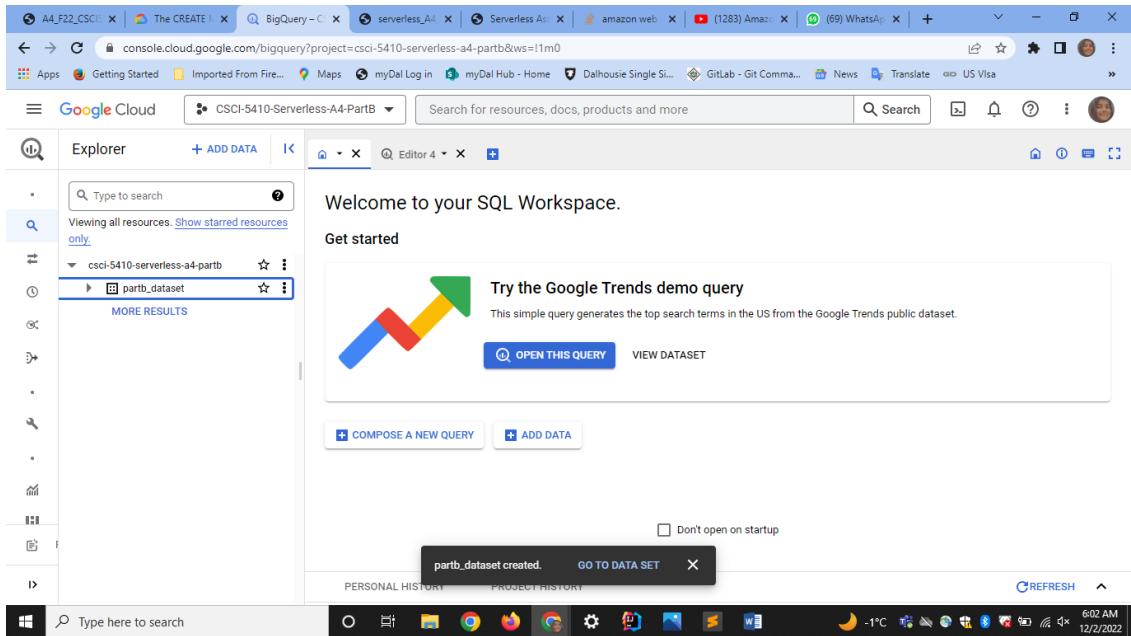


Fig 9: Dataset “partb_dataset” has been created

Step 5:

- Now we need to create table and upload data into the table.
- Fig 10 and 11 are the provided data set named “SDey_FTP_input” which is to be uploaded.

SDey_FTP_input - Excel	
File Home Insert Page Layout Formulas Data Review View Help SOLIDWORKS Manage Tell me what you want to do	
Clipboard	
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.	
A1	V_1 V_2
1	0.000002 0.001652
2	0.001652 0.000007
3	0.000007 0.000065
4	0.000065 0.000003
5	0.000003 0.001489
6	0.001489 0.000004
7	0.000004 0.000123
8	0.000123 0.000004
9	0.000004 0.001431
10	0.001431 0.000004
11	0.000004 0.000001
12	0.000001 0.000001
13	0.000001 0.001815
14	0.001815 0.000005
15	0.000005 0.002027
16	0.002027 0.000004
17	0.000004 0.000002
18	0.000002 0.000001
19	0.000001 0.001927
20	0.001927 0.000004
21	

Fig 10: Dataset

The screenshot shows a Microsoft Excel spreadsheet titled "SDey_FTP_input - Excel". The table has columns labeled A through T. The first few rows of data are as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
57982	0.000005	0.000233																		
57983	0.000233	0.000005																		
57984	0.000005	0.000001																		
57985	0.000001	0.000002																		
57986	0.000002	0.003137																		
57987	0.003137	0.000006																		
57988	0.000006	0.003735																		
57989	0.003735	0.000004																		
57990	0.000004	0.000001																		
57991	0.000001	0.000002																		
57992	0.000002	0.000001																		
57993	0.000001	0.013146																		
57994	0.013146	0.001157																		
57995	0.001157	0.000005																		
57996	0.000005	0.004293																		
57997	0.004293	0.002671																		
57998	0.002671	0.000005																		
57999	0.000005	0.000838																		
58000	0.000838	0.000004																		
58001	0.000004	0.000195																		
58002	0.000195	0.005018																		

Fig 11: Dataset

Step 6:

- To add data, fill the details and upload file as shown in the figure 12, 13 and 14.
- Fig 15 shows “partb_dataset_table” has been created.
- Fig 16 shows the field values.

The screenshot shows the Google Cloud BigQuery interface. On the left, there is a sidebar with the 'Explorer' tab selected, showing a project structure with 'cscl-5410-serverless-a4-partb' and 'partb_dataset'. The main area is titled 'Add data' and contains the following sections:

- Source:** A search bar labeled 'Search for data sources'.
- Popular sources:** Three options: 'Local file' (Upload a local file), 'Google Cloud Storage' (Google object storage service), and 'Connections to external data sources' (Connection from BigQuery to an external data source).
- Additional sources:** Two search bars: 'Search for and star a project' and 'Star a project by name'.
- PERSONA:** A 'CLOSE' button.

Fig 12: “Add data” form

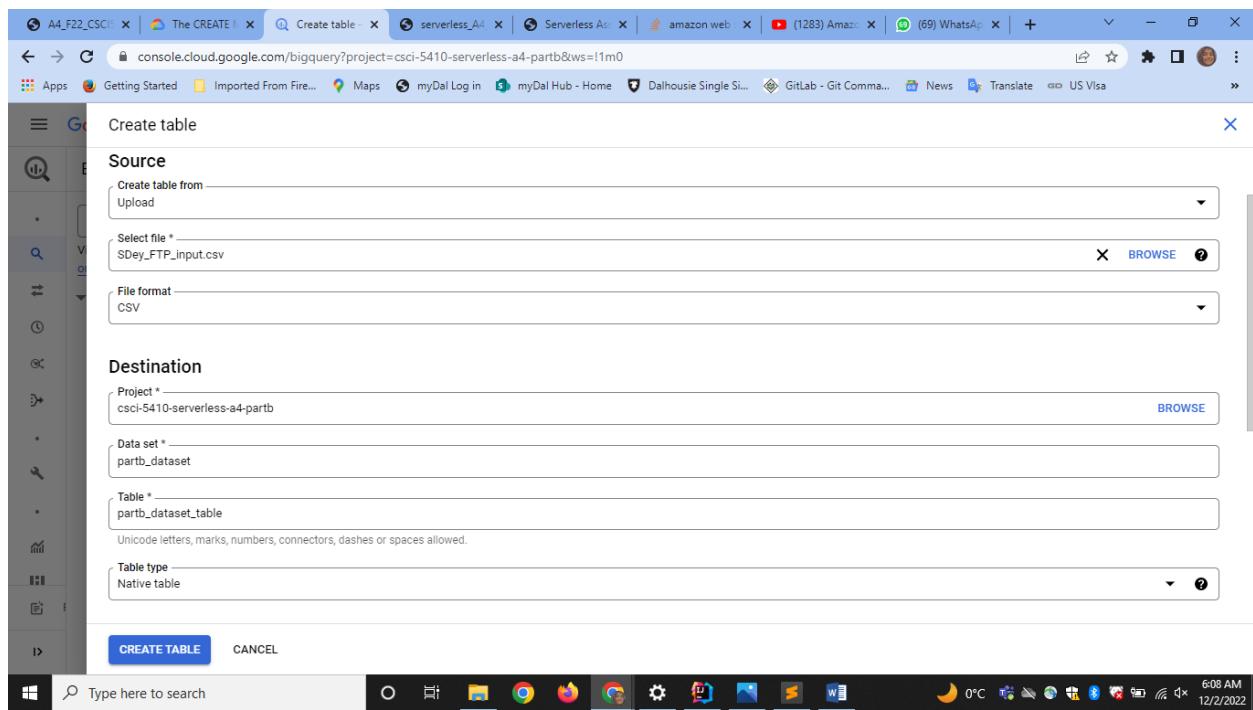


Fig 13: “Add data” form

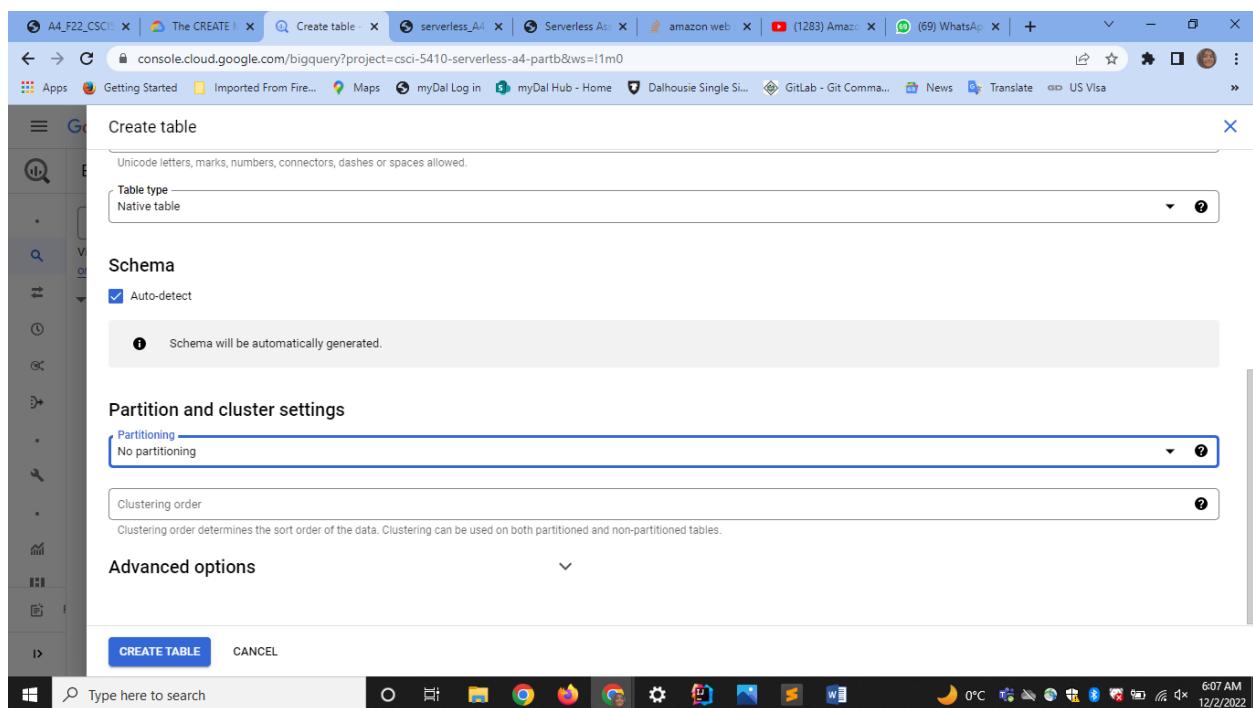


Fig 14: “Add data” form

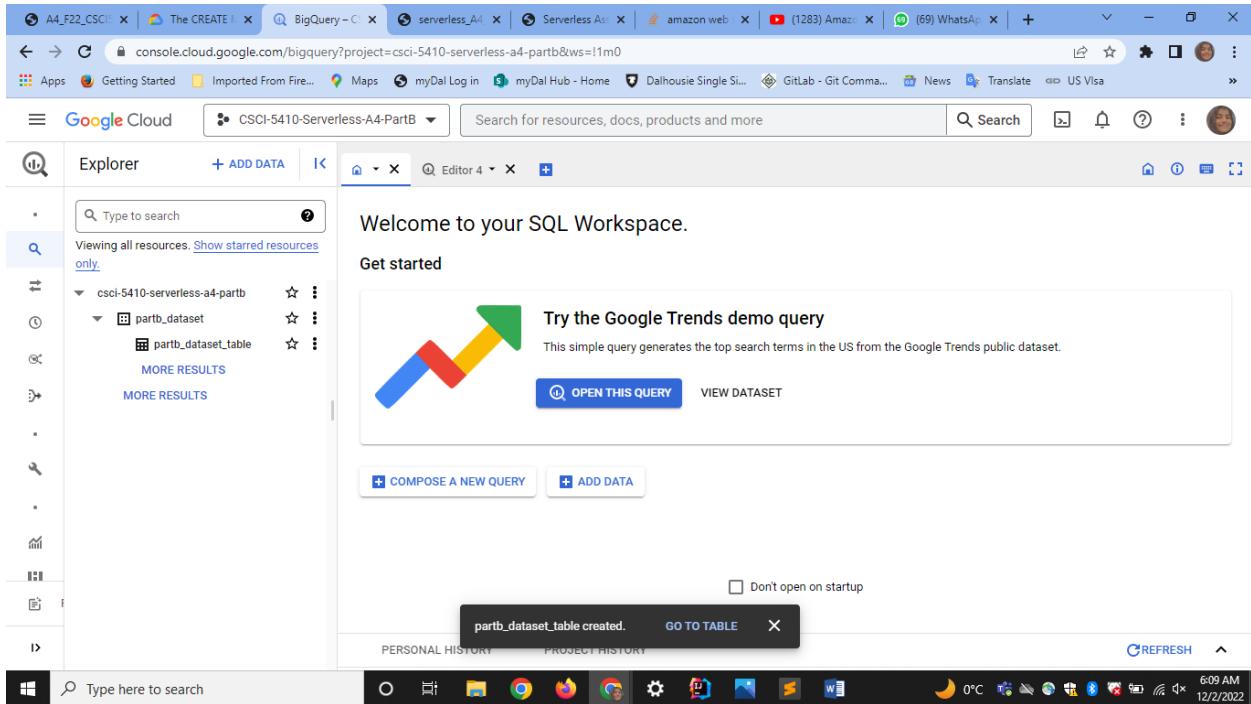


Fig 15: “partb_dataset_table” has been created

Field name	Type	Mode	Collation	Default value	Policy tags	Description
V_1	FLOAT	NULLABLE				
V_2	FLOAT	NULLABLE				

Fig 16: Field values of the “partb_dataset_table”

Step 7:

- Change the settings for the query by clicking on the “more” option under “partb_dataset_table” as shown in the fig 17. Then go to “Query settings” as shown in the fig. 17.
- Fill the “Query settings” details as shown in the fig 18, 19, 20.

- Keep the location same as the “partb_dataset” which is “US”.

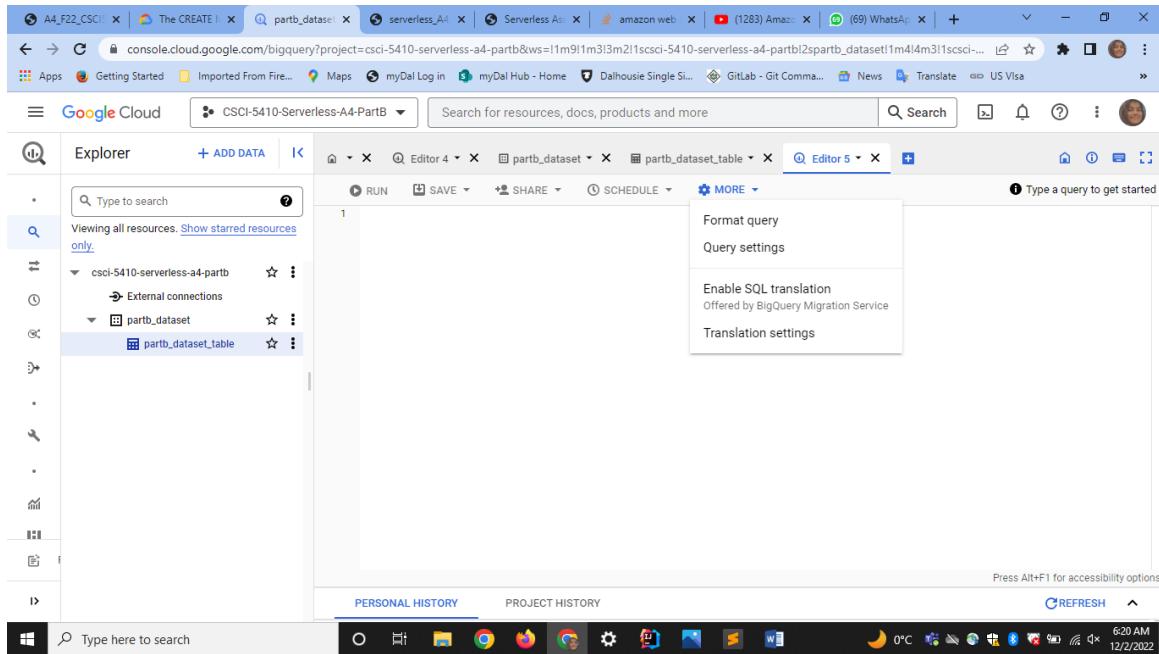


Fig 17: “Query setting” option

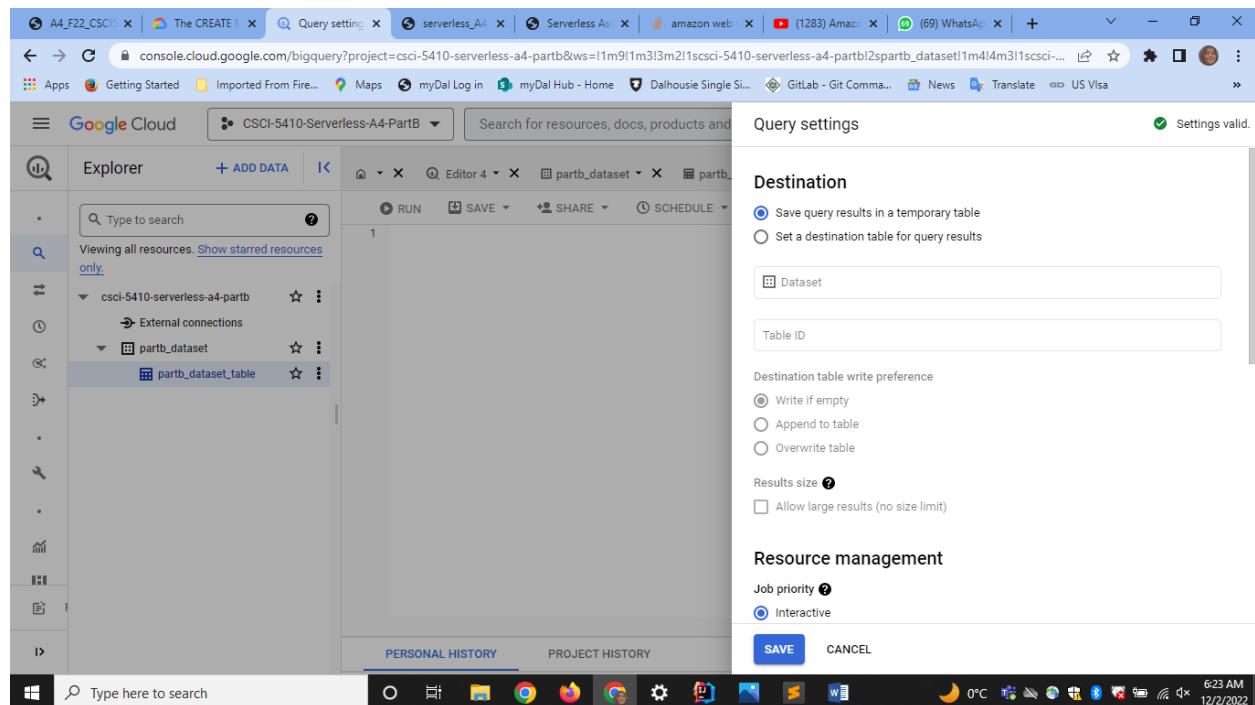


Fig 18: “Query settings” form

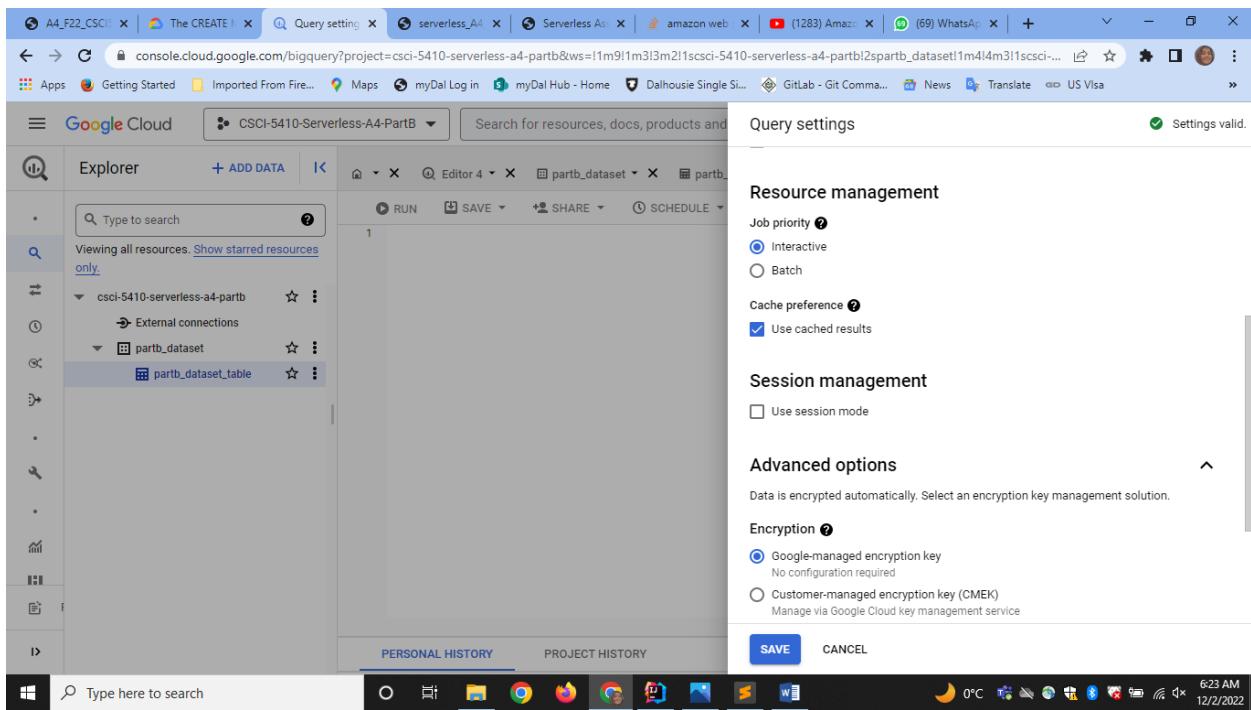


Fig 19: “Query settings” form

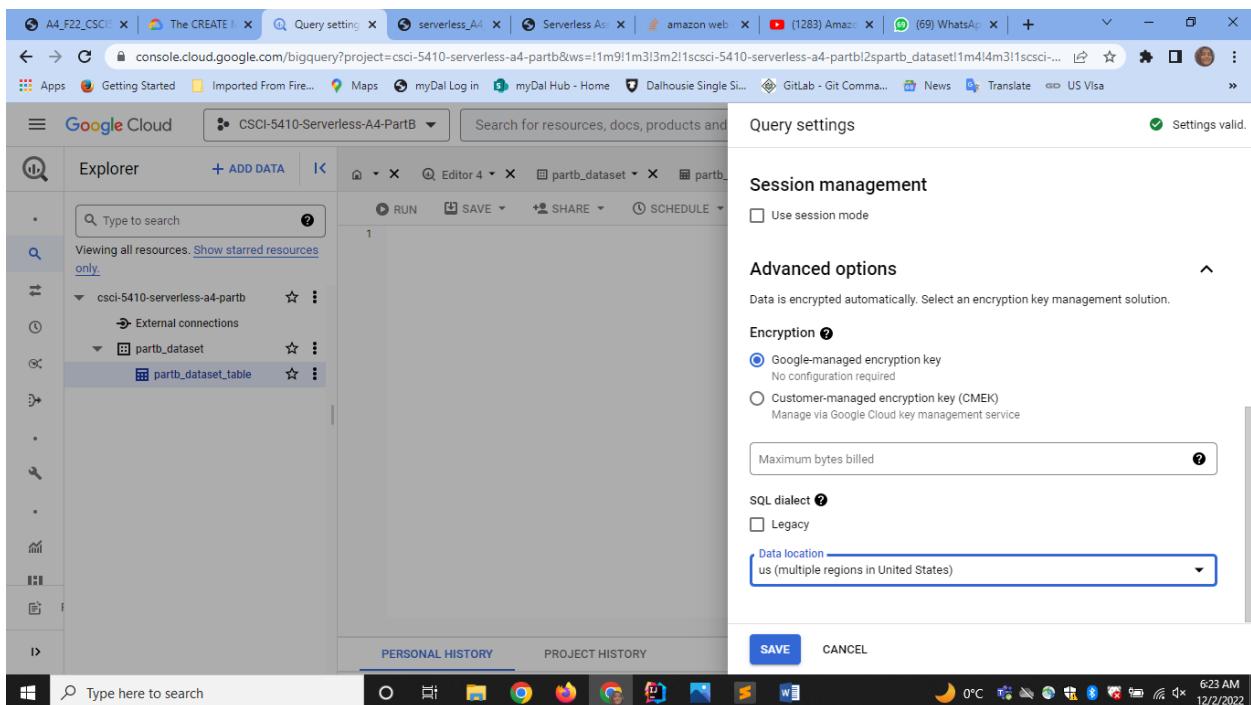
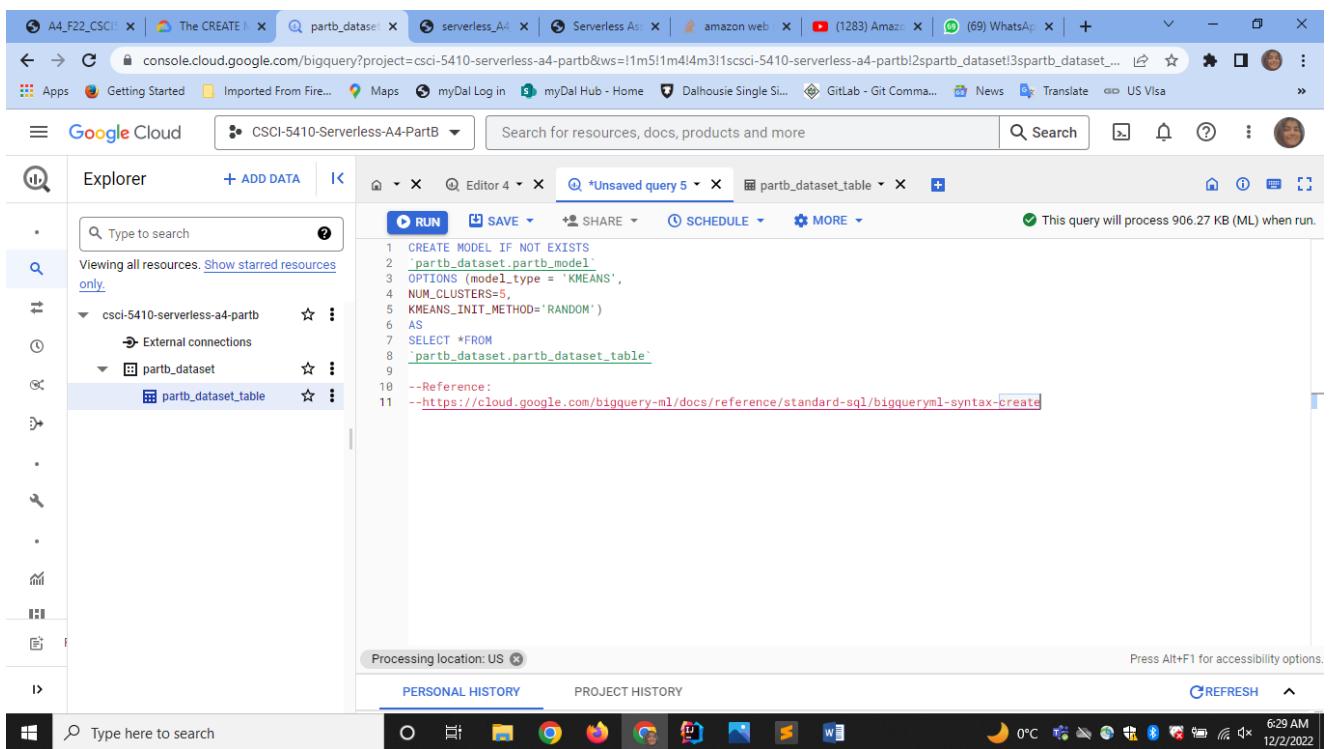


Fig 20: “Query settings” form

Step 8:

- I wrote a query to train the model as shown in the fig 21. The query creates a k-means model with 5 clusters using the default “DISTANCE_TYPE” value of “EUCLIDEAN_DISTANCE” [1].
- Fig 22 shows query running with the elapsed time.
- Fig 23 shows that query is executed.
- Fig 24 and 25 provides the model details which is been created.
- Fig 26, 27, 28, 29, 30 and 31 gives the graphical view, table view of the results.
- Fig 32 represents evaluation details of the “partb_model”.
- Fig 33 represents job details of the “partb_model”.



The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays the project structure under 'csci-5410-serverless-a4-partb'. The main area shows an 'Editor' tab with an 'Unsaved query 5' tab open. The query code is as follows:

```
1 CREATE MODEL IF NOT EXISTS
2   `partb_dataset.partb_model`
3 OPTIONS (model_type = 'KMEANS',
4   NUM_CLUSTERS=5,
5   KMEANS_INIT_METHOD='RANDOM')
6 AS
7 SELECT *FROM
8   `partb_dataset.partb_dataset_table`
9
10 --Reference:
11 --https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create
```

The status bar at the bottom indicates the processing location is US. The taskbar at the bottom right shows the date and time as 12/2/2022, 6:29 AM.

Fig 21: Query to train the model

The screenshot shows the Google Cloud BigQuery interface. On the left, the Explorer sidebar displays a project structure with a dataset named 'partb_dataset' containing a table named 'partb_dataset_table'. In the center, a query editor window is open with the following SQL code:

```

1 CREATE MODEL IF NOT EXISTS
2   `partb_dataset.partb_model`
3   OPTIONS (model_type = 'KMEANS',
4           NUM_CLUSTERS=5,
5           KMEANS_INIT_METHOD='RANDOM')
6   AS
7   SELECT * FROM
8   `partb_dataset.partb_dataset_table`
9
10 --Reference:
11 -->https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create

```

The status bar at the bottom indicates "Query running (4.4 sec - Stage: Preprocess)". Below the editor, the "Query results" section shows job information: Elapsed time 4 sec, Slot time consumed 0 sec, Stages 0 (Preprocess), and Training iterations Completed: 0, Planned: 0.

Fig 22: Query running

The screenshot shows the Google Cloud BigQuery interface after the query has completed. The Explorer sidebar now includes a "Saved queries" section with entries for "2022-12-02 06:55:21" and "2022-12-02 07:07:32". The query editor window shows the same SQL code as in Fig 22, but the status bar now says "This query will process 0 B (ML) when run." The "Query results" section shows job information: Elapsed time 1 min 5 sec, Slot time consumed 13 min 41 sec, Stages 0 (Preprocess, Train, Evaluate), and Training iterations Completed: 7, Planned: 20. Below the results, a chart titled "Loss" vs "Duration (seconds)" shows a single data point at 2 seconds.

Fig 23: Query results

The screenshot shows the Google Cloud Platform interface with the search bar set to 'CSCI-5410-Serverless-A4-PartB'. The main view displays the 'partb_model' details page under the 'partb_dataset_table' project. The 'DETAILS' tab is selected, showing the following information:

Model type	KMEANS	Data location	US
Model details	EDIT		
Model ID	csci-5410-serverless-a4-partb.partb_dataset.partb_model		
Description			
Labels			
Date created	2 Dec 2022, 06:33:07 UTC-4		
Model expiry	Never		
Date modified	2 Dec 2022, 06:33:07 UTC-4		
Data location	US		
Model type	KMEANS		

Below the details, the 'Training options' section is visible, containing optional parameters used in the script to create the model. The 'PERSONAL HISTORY' and 'PROJECT HISTORY' tabs are also present at the bottom of the details page.

Fig 24: Model details for “partb_model”

This screenshot is identical to Fig 24, showing the 'partb_model' details page in the Google Cloud Platform interface. The 'DETAILS' tab is selected, displaying the same model information and training options as in Fig 24. The 'PERSONAL HISTORY' and 'PROJECT HISTORY' tabs are also visible at the bottom.

Fig 25: Model details for “partb_model”

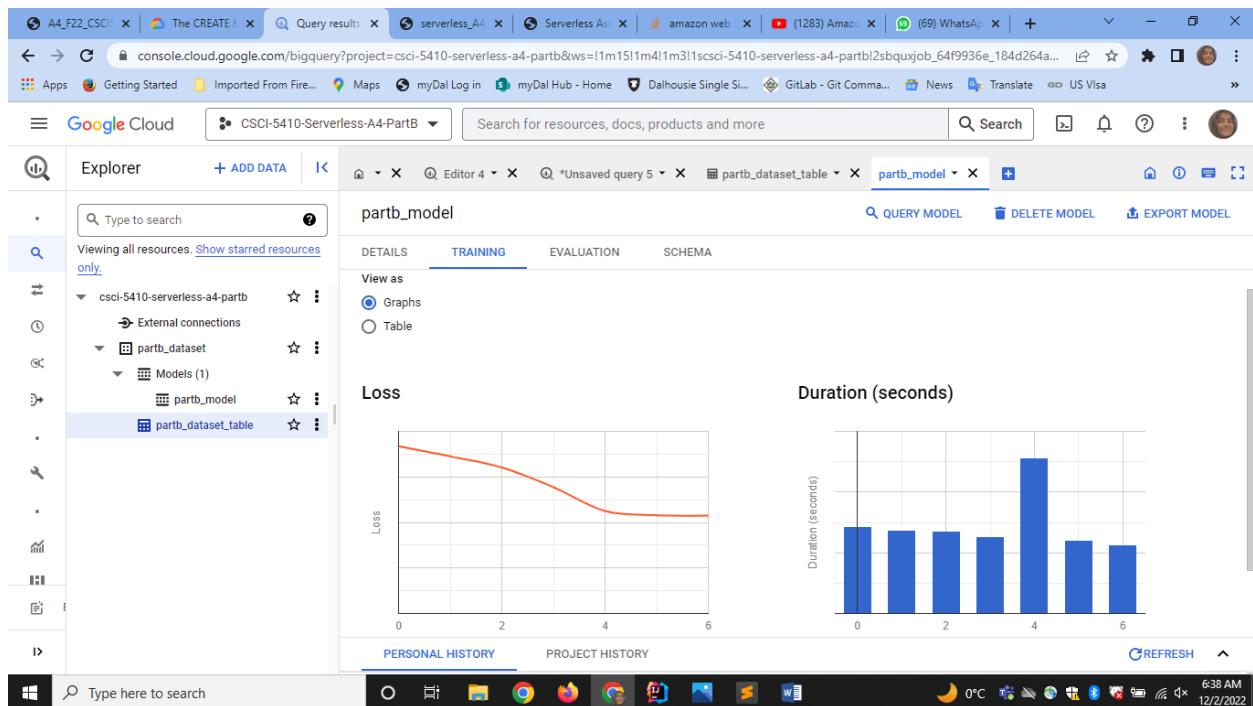


Fig 26: Training details for “partb_model” – Graph view

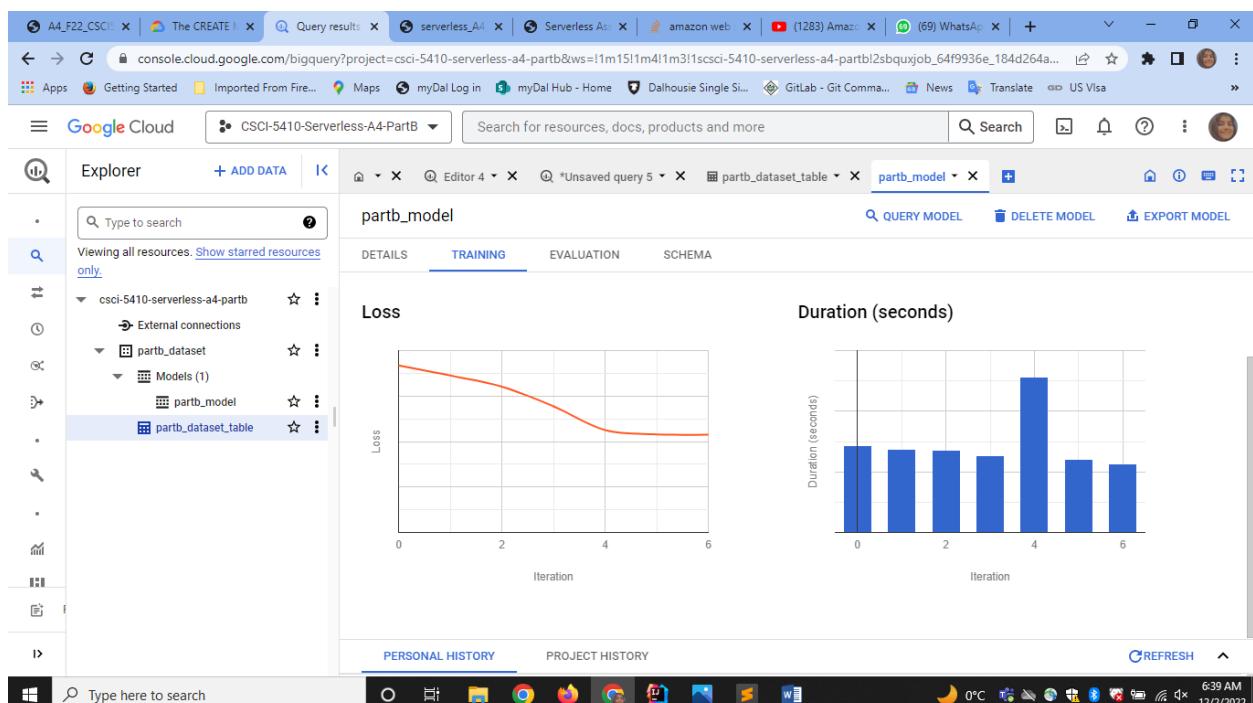


Fig 27: Training details for “partb_model” – Graph view

The screenshot shows the Google Cloud Platform interface for a project named 'CSCI-5410-Serverless-A4-PartB'. In the left sidebar, under 'Explorer', there is a tree view showing a dataset named 'csci-5410-serverless-a4-partb' which contains a 'Saved queries (1)' node and an 'External connections' node. Under 'partb_dataset', there is a 'Models (1)' node which contains a 'partb_model' node. The main content area displays a table titled 'partb_model' with the 'TRAINING' tab selected. The table has columns: Iteration, Training data loss, Duration (seconds), Cluster centroid ID, Cluster radius, and Cluster size. The data shows multiple iterations with varying values across these metrics.

Iteration	Training data loss	Duration (seconds)	Cluster centroid ID	Cluster radius	Cluster size
6	1.0748	2.26	1	0.50143377	5175
			2	17.71623155	166
			3	1.38367642	1096
			4	0.40970695	39686
			5	0.12172496	11878
5	1.0789	2.42	1	0.41424954	5536
			2	17.72395901	166
			3	1.34845274	1353
			4	0.41792731	39281
			5	0.11736526	11665

Fig 28: Training details for “partb_model” – table view

This screenshot is identical to Fig 28, showing the same Google Cloud Platform interface and training details for the 'partb_model' in the 'csci-5410-serverless-a4-partb' dataset. The table structure and data values are also identical.

Iteration	Training data loss	Duration (seconds)	Cluster centroid ID	Cluster radius	Cluster size
6	1.0748	2.26	1	0.50143377	5175
			2	17.71623155	166
			3	1.38367642	1096
			4	0.40970695	39686
			5	0.12172496	11878
5	1.0789	2.42	1	0.41424954	5536
			2	17.72395901	166
			3	1.34845274	1353
			4	0.41792731	39281
			5	0.11736526	11665

Fig 29: Training details for “partb_model” – table view

The screenshot shows the Google Cloud Platform interface with the search bar set to 'partb_dataset_table'. The main view displays the 'partb_model' training details table. The table has columns: DETAILS, TRAINING, EVALUATION, and SCHEMA. The TRAINING column contains values such as 3, 2, 1, etc., and the EVALUATION column contains values like 1.3874, 1.6037, 1.7260, etc. The SCHEMA column lists various numerical values. The bottom right of the table shows 'PERSONAL HISTORY' and 'PROJECT HISTORY' buttons, and a 'REFRESH' button.

DETAILS	TRAINING	EVALUATION	SCHEMA
	3	1.3874	2.54
	2		1 0.29625951 7489
			2 17.13951765 248
			3 0.90853282 2293
			4 0.35312711 40084
			5 0.09150513 7887
	2	1.6037	2.72
	1		1 0.23417653 8147
			2 11.80470554 627
			3 0.73764161 3087
			4 0.29221384 40833
			5 0.07264775 5307
	1	1.7260	2.74
	2		1 0.17114555 7373
			2 7.60310662 1666
			3 0.59100772 4306
			4 0.22538646 40283
			5 0.08808484 4373

Fig 30: Training details for “partb_model” – table view

This screenshot is nearly identical to Fig 30, showing the same 'partb_model' training details table. The data rows are slightly different, reflecting updated training results. The bottom right of the table shows 'PERSONAL HISTORY' and 'PROJECT HISTORY' buttons, and a 'REFRESH' button.

DETAILS	TRAINING	EVALUATION	SCHEMA
	5		0.07264775 5307
	1	1.7260	2.74
	2		1 0.17114555 7373
			2 7.60310662 1666
			3 0.59100772 4306
			4 0.22538646 40283
			5 0.08808484 4373
	0	1.8367	2.86
	1		1 0.10913171 6519
			2 4.89317648 4376
			3 0.33411633 4029
			4 0.20248316 39148
			5 0.11107073 3929

Fig 31: Training details for “partb_model” – table view

Type to search

Viewing all resources. [Show starred resources only.](#)

csci-5410-serverless-a4-partb

- External connections
- partb_dataset**
- Models (1)
 - partb_model**
- partb_dataset_table**

partb_model

EVALUATION

Metrics

Davies–Bouldin index	1.6145
Mean squared distance	1.0748

Numeric features

This table shows the centroid value for each feature. Use the select menu to view more numeric features.

Selected features: V_1, V_2

Centroid ID	Count	V_1	V_2
1	5,175	0.0011	0.0037
2	166	0.0723	0.0718
3	1,096	0.0014	0.0115
4	39,686	0.0011	0.0000

REFRESH

Fig 32: Evaluation details for “partb_model”

Type to search

Viewing all resources. [Show starred resources only.](#)

csci-5410-serverless-a4-partb

- External connections
- partb_dataset**
- Models (2)
 - partb_model**
 - partb_model1**
- partb_dataset_table**

partb_model

RUN

2022-12-02 06:55:21

```

1 CREATE MODEL IF NOT EXISTS
2 `partb_dataset.partb_model`
```

Processing location: US

Query results

JOB INFORMATION

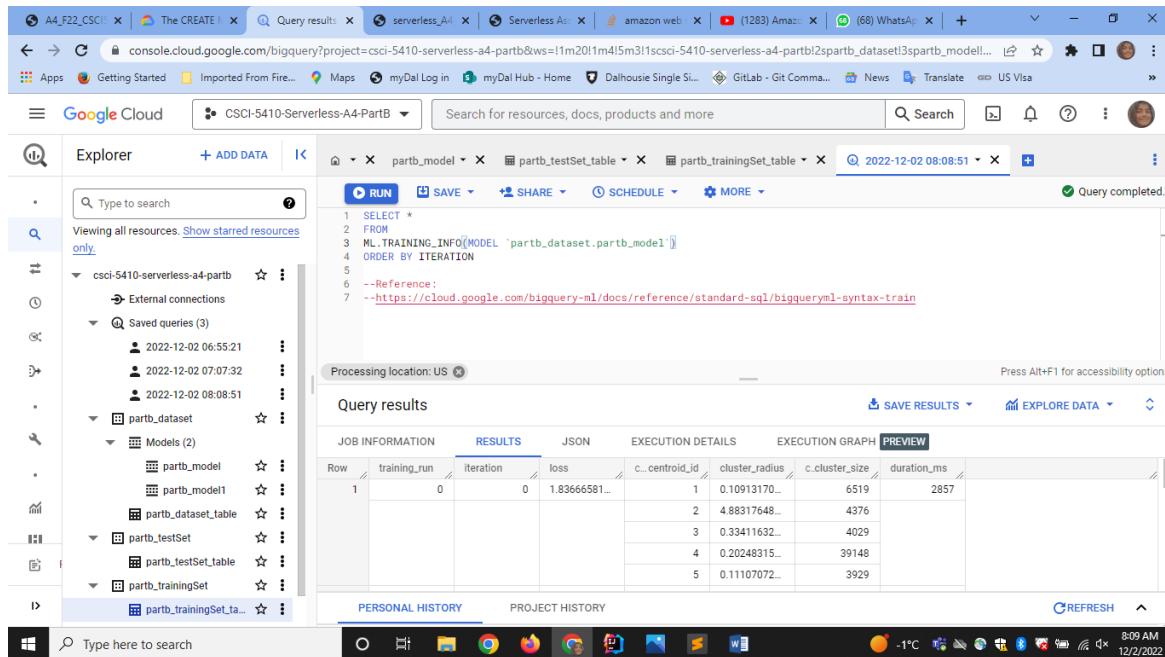
Job ID	csci-5410-serverless-a4-partb:US.bqjob_64f9936e_184d264aa0b
User	faizumatiya@gmail.com
Location	US
Creation time	2 Dec 2022, 06:32:02 UTC-4
Start time	2 Dec 2022, 06:32:02 UTC-4
End time	2 Dec 2022, 06:33:07 UTC-4
Duration	1 min 5 sec
Bytes processed	906.27 KB
Bytes billed	10 MB
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	csci-5410-serverless-a4-partb.partb_dataset.partb_model

REFRESH

Fig 33: Job details for “partb_model”

EXTRA:

- We can also check the training model training statics by writing the query as shown in the fig 33.1.
- Fig 33.2, 33.3, 33.4 and 33.5 shows the query results for training model.



The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are several tabs: A4_F22_CSCI, The CREATE, Query results, serverless_A4, Serverless A..., amazon web, (1283) Amazon, (68) WhatsApp, and a plus sign for new tabs. Below the tabs, the URL is console.cloud.google.com/bigquery?project=csci-5410-serverless-a4-partb&ws=!1m2!1m4!5m3!1scsci-5410-serverless-a4-partb!2spartb_dataset!3spartb_model!... . The main area has a search bar "Search for resources, docs, products and more" and a "Search" button. On the left, there's an "Explorer" sidebar with a tree view of resources under "csci-5410-serverless-a4-partb". The tree includes "Saved queries (3)", "partb_dataset" (with "Models (2)" containing "partb_model" and "partb_model1"), and "partb_testSet" (with "partb_testSet_table" and "partb_trainingSet" (with "partb_trainingSet_table"). The main panel shows a query editor with the following SQL code:

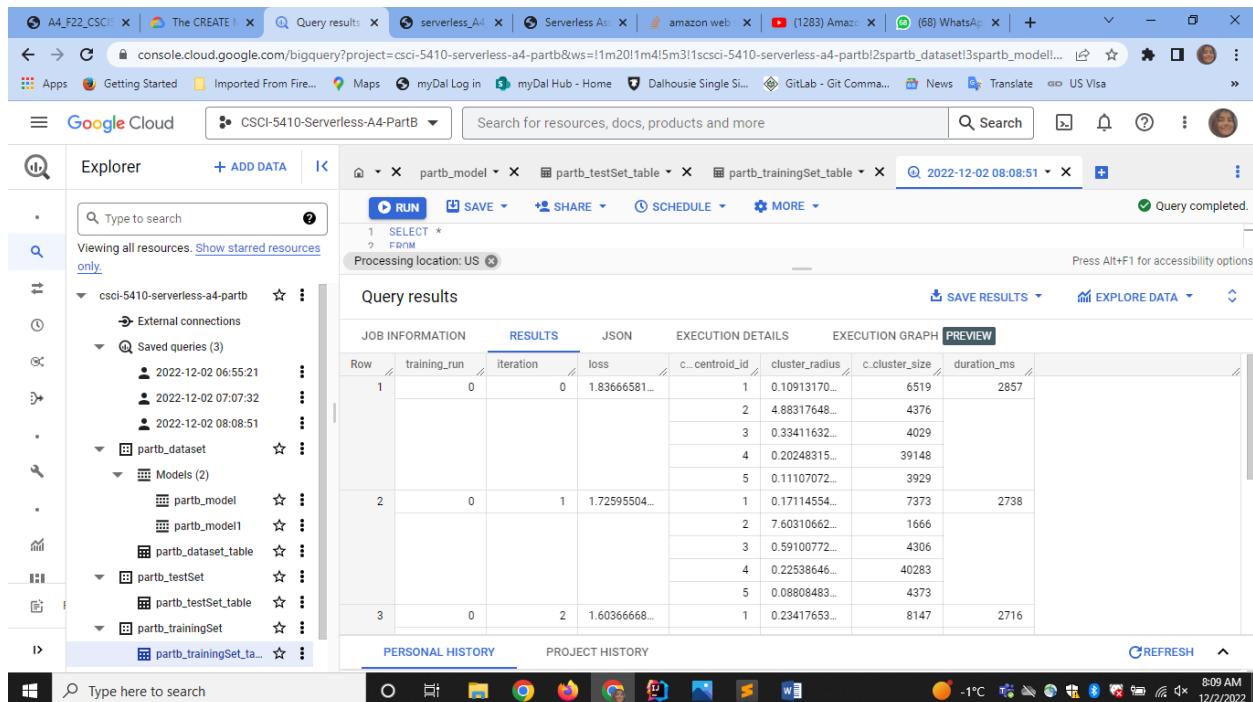
```

1 SELECT *
2 FROM
3 ML.TRAINING_INFO(MODEL `partb_dataset.partb_model`)
4 ORDER BY ITERATION
5
6 --Reference:
7 --https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-train
  
```

The status bar at the bottom right indicates "Query completed." Below the code, the "Query results" section displays a table with the following data:

Row	training_run	iteration	loss	c...centroid_id	cluster_radius	c.cluster_size	duration_ms
1	0	0	1.83666581...	1	0.10913170...	6519	2857
				2	4.88317648...	4376	
				3	0.33411632...	4029	
				4	0.20248315...	39148	
				5	0.11107072...	3929	

Fig 33.1: Query to check model training



This screenshot is identical to Fig 33.1, showing the same Google Cloud BigQuery interface and query results. The query results table is identical to the one in Fig 33.1, displaying five rows of data for the first training run (iteration 0).

Fig 33.2: Query results for training statistics

The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays a project structure under 'csci-5410-serverless-a4-partb'. The main area shows a query results page for a job named 'partb_model' with a timestamp of '2022-12-02 08:08:51'. The results table has columns: Row, training_run, iteration, loss, c...centroid_id, cluster_radius, c.cluster_size, and duration_ms. The data shows multiple iterations for different training runs, with the final row being iteration 4 of training run 0.

Row	training_run	iteration	loss	c...centroid_id	cluster_radius	c.cluster_size	duration_ms
3		0	2	1.6036668...	1	0.23417653...	8147
					2	11.8047055...	627
					3	0.73764161...	3087
					4	0.29221384...	40833
					5	0.07264774...	5307
4		0	3	1.38735068...	1	0.29625951...	7489
					2	17.1395176...	248
					3	0.90853282...	2293
					4	0.35312710...	40084
					5	0.09150512...	7887
5		0	4	1.12561709...	1	0.35113798...	6111
							5135

Fig 33.3: Query results for training statistics

This screenshot is nearly identical to Fig 33.3, showing the same query results for the 'partb_model' job. The data in the results table is very similar, with slight differences in the execution details and duration_ms values. The final row is iteration 6 of training run 0.

Row	training_run	iteration	loss	c...centroid_id	cluster_radius	c.cluster_size	duration_ms
5		0	4	1.12561709...	1	0.35113798...	6111
					2	17.7683295...	176
					3	1.16208843...	1791
					4	0.40439785...	39261
					5	0.10980606...	10662
6		0	5	1.07892693...	1	0.41424954...	5536
					2	17.7239590...	166
					3	1.34845273...	1353
					4	0.41792730...	39281
					5	0.11736525...	11665
7		0	6	1.07478810...	1	0.50143376...	5175
							2264

Fig 33.4: Query results for training statistics

The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays project resources, including a saved query named '2022-12-02 06:55:21'. The main area shows the results of a query for 'partb_trainingSet_table'. The query details are:

```

1 SELECT *
2 FROM `csci-5410-serverless-a4-partb.partb_dataset.partb_trainingSet_table`@US
  
```

The results table has columns: Row, training_run, iteration, loss, c...centroid_id, cluster_radius, c.cluster_size, duration_ms. The data is as follows:

Row	training_run	iteration	loss	c...centroid_id	cluster_radius	c.cluster_size	duration_ms
6		0	5	1.07892693...	1	0.41424954...	5536
					2	17.7239590...	166
					3	1.34845273...	1353
					4	0.41792730...	39281
					5	0.11736525...	11665
7		0	6	1.07478810...	1	0.50143376...	5175
					2	17.7162315...	166
					3	1.38367642...	1096
					4	0.40970694...	39686
					5	0.12172496...	11878

Fig 33.5: Query results for training statistics

Alternative query:

Follow the same step as Step 7:

Step 8:

- Change the settings for the query by clicking on the “more” option under “partb_dataset_table” as shown in the fig 34. Then go to “Query settings” as shown in the fig. 34.
- Fill the “Query settings” details as shown in the fig 35, 36, 37 and 38.
- Keep the location same as the “partb_dataset” which is “US”.

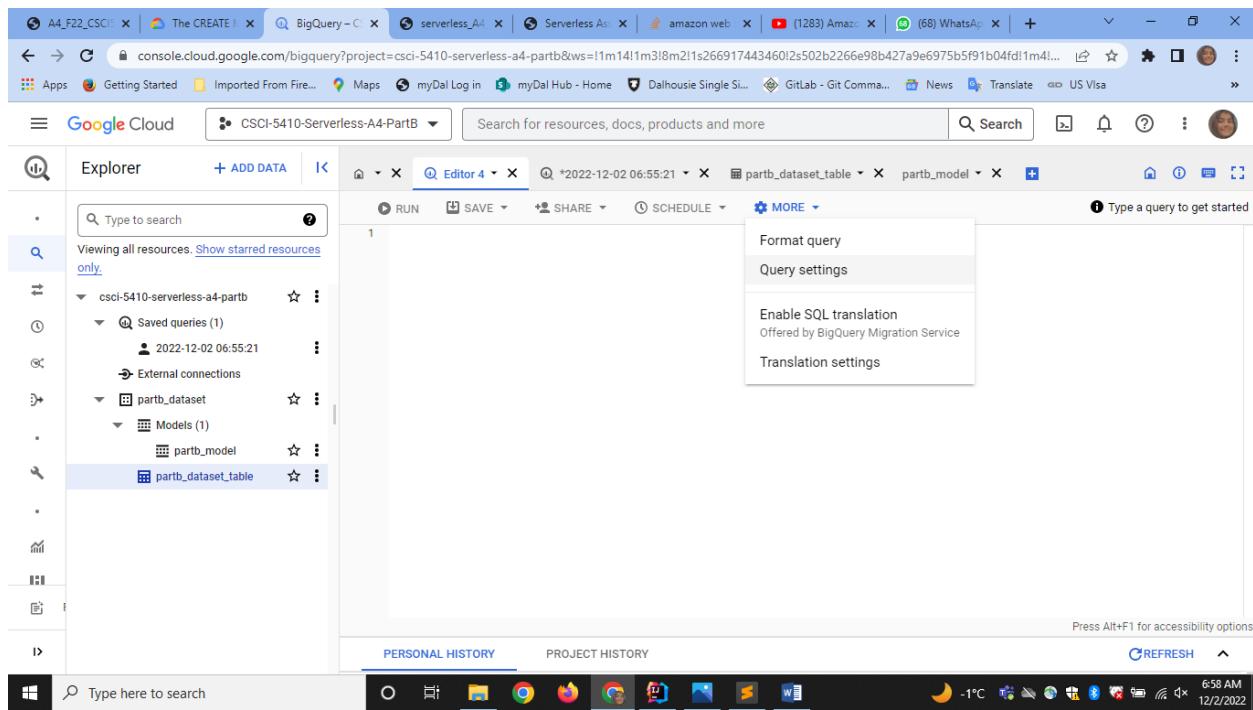


Fig 34: “Query setting” option

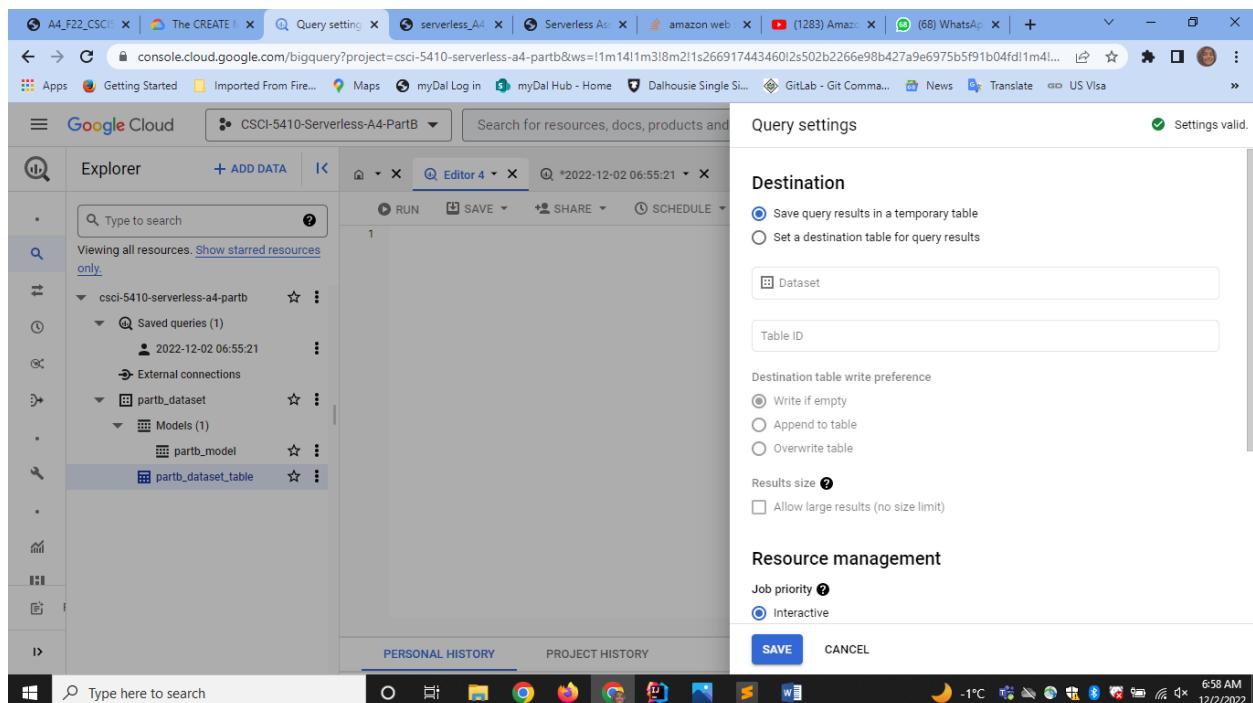


Fig 35: “Query settings” form

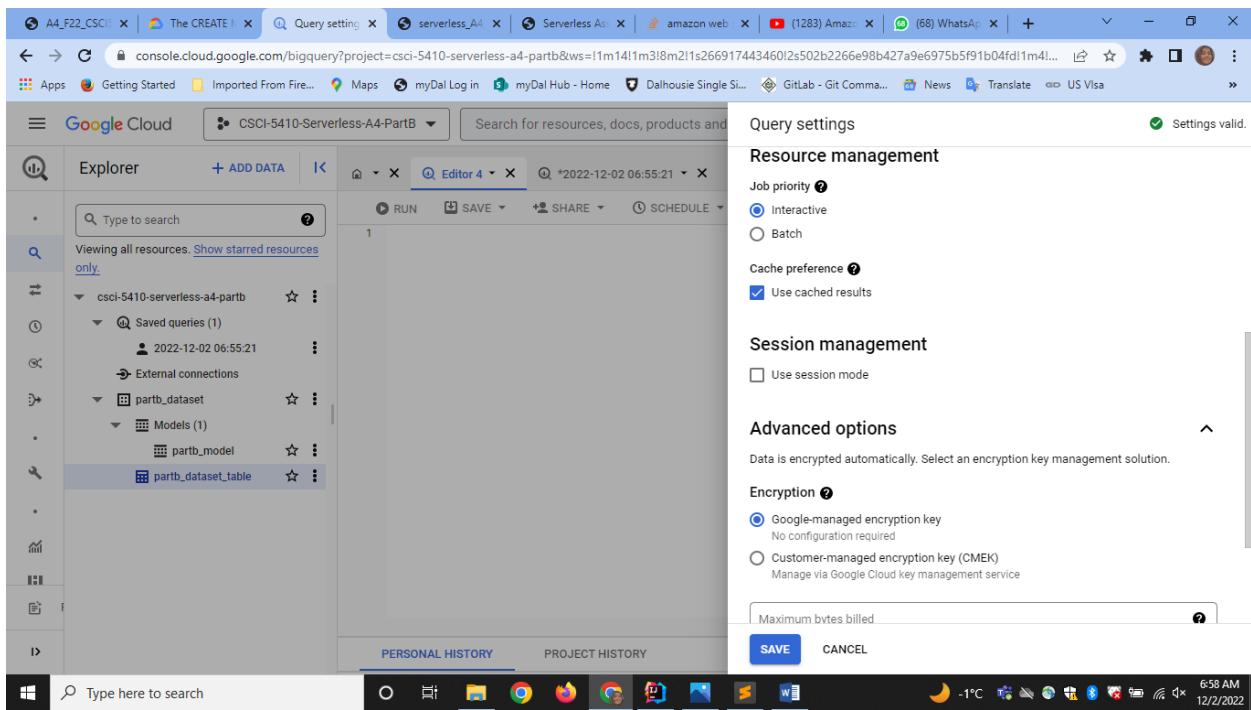


Fig 36: “Query settings” form

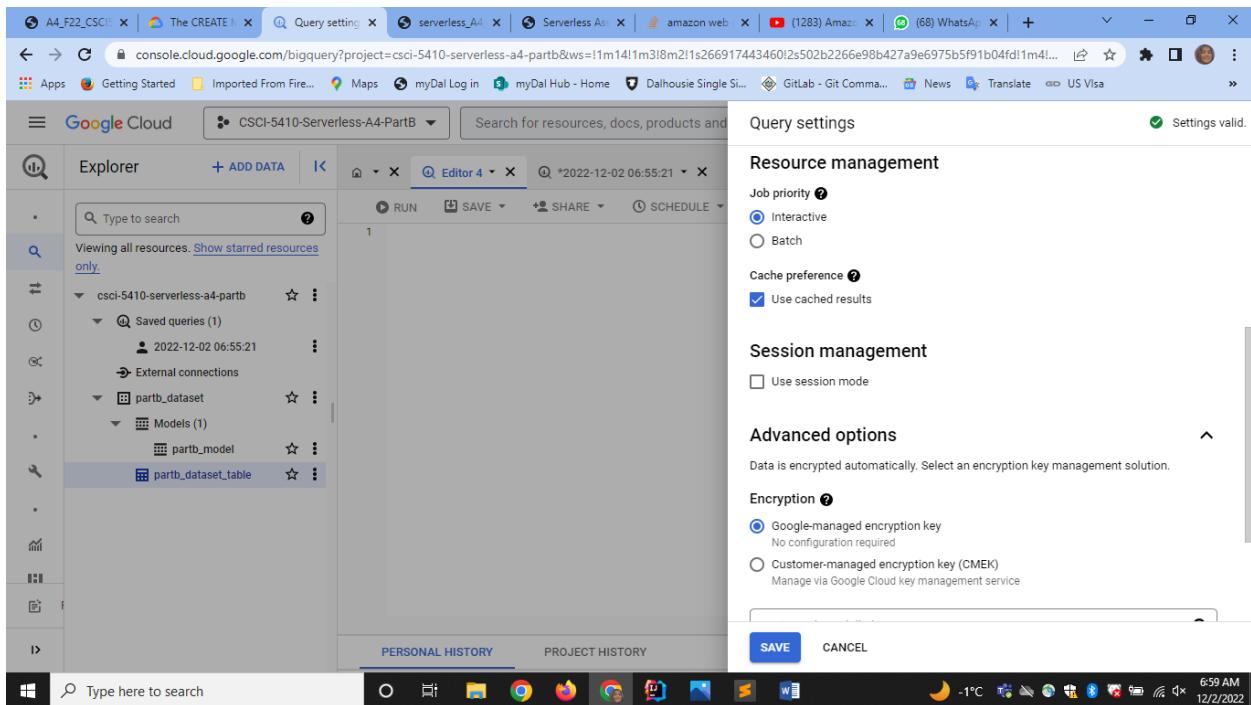


Fig 37: “Query settings” form

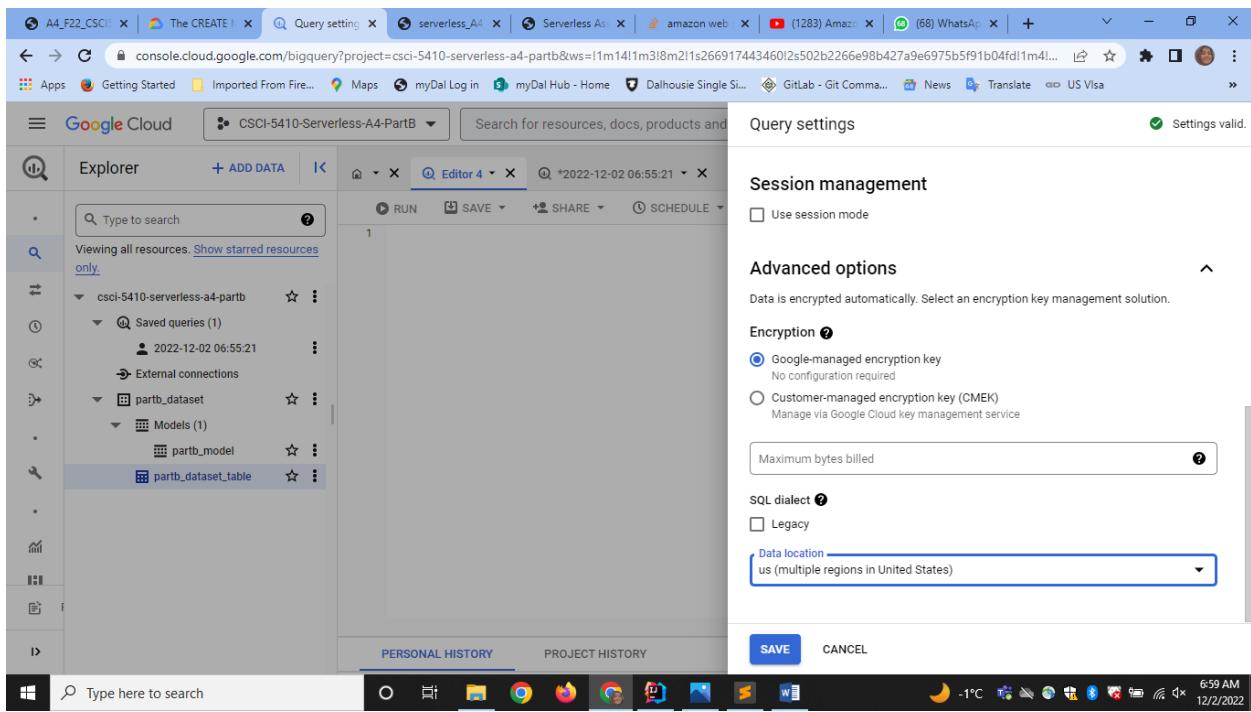


Fig 38: “Query settings” form

Step 9:

- I wrote a query to train the model as shown in the fig 39. The query creates a k-means model with 11 clusters using the “DISTANCE_TYPE” value of “EUCLIDEAN_DISTANCE” [1].
- Fig 40 shows query running with the elapsed time.
- Fig 41 shows that query is executed.
- Fig 42 and 43 provides the model details which is been created.
- Fig 44, 45, 46, 47, 48, 49 and 50 gives the graphical view, table view of the results.
- Fig 51 and 52 represents evaluation details of the “partb_model1”.
- Fig 53 represents job details of the “partb_model1”.

The screenshot shows the Google Cloud BigQuery interface. On the left, the Explorer sidebar displays a project named 'csci-5410-serverless-a4-partb' containing two saved queries and one model named 'partb_model'. The main area shows a query editor with the following SQL code:

```

1 CREATE MODEL IF NOT EXISTS
2   `partb_dataset.partb_model1`
3   OPTIONS (model_type = 'KMEANS',
4           NUM_CLUSTERS=11,
5           KMEANS_INIT_METHOD='RANDOM',
6           DISTANCE_TYPE= 'EUCLIDEAN')
7   AS
8   SELECT *FROM
9   `partb_dataset.partb_dataset_table`
10
11 --Reference:
12 --https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create

```

The status bar at the bottom indicates the processing location is US.

Fig 39: Query to train the model

The screenshot shows the Google Cloud BigQuery interface with the same project and dataset structure as Fig 39. The main area now displays the 'Query results' tab, indicating the query is running. The status bar at the bottom shows the elapsed time as 2 sec.

JOB INFORMATION	RESULTS	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Elapsed time 2 sec	Slot time consumed 0 sec	Stages Preprocess 0	Training iterations Completed: 0 Planned: 0	

Fig 40: Query running

The screenshot shows the Google Cloud BigQuery interface. At the top, there are several tabs and a search bar. Below the search bar, the main area displays a query results page. The query details are as follows:

- Project:** CSCI-5410-Serverless-A4-PartB
- Job ID:** 2022-12-02_07:07:32
- Processing location:** US
- Elapsed time:** 2 min 13 sec
- Slot time consumed:** 28 min 5 sec
- Stages:**
 - Preprocess: 0
 - Train: 0
 - Evaluate: 0
- Training iterations:** Completed: 17, Planned: 20

Below the execution details, there is a chart titled "Loss" and "Duration (seconds)". The chart shows two horizontal bars: one for Loss (ranging from 2 to 15) and one for Duration (seconds) (ranging from 15 to 2). The "PERSONAL HISTORY" tab is selected at the bottom.

Fig 41: Query completed

The screenshot shows the Google Cloud BigQuery interface. The left sidebar shows the project structure, including a saved query named "2022-12-02_07:07:32". The main panel displays the details for a K-Means model named "partb_model1".

Details	Value
Model type	KMEANS
Data location	US
Model details	<p>Model ID: csci-5410-serverless-a4-partb.partb_dataset.partb_model1</p> <p>Description: (empty)</p> <p>Labels: (empty)</p> <p>Date created: 2 Dec 2022, 07:10:21 UTC-4</p> <p>Model expiry: Never</p> <p>Date modified: 2 Dec 2022, 07:10:21 UTC-4</p> <p>Data location: US</p> <p>Model type: KMEANS</p>

Below the details, there is a "Training options" section which is currently empty. The "PERSONAL HISTORY" tab is selected at the bottom.

Fig 42: Model details for "partb_model1"

The screenshot shows the Google Cloud Platform interface for a KMEANS model named "partb_model1".

- Explorer:** Shows the project structure under "csci-5410-serverless-a4-partb".
- Details Tab:** Displays basic information like Data location (US) and Model type (KMEANS).
- Training Options:**
 - Max allowed iterations: 20
 - Actual iterations: 17
 - Early stop: true
 - Min relative progress: 0.01
 - Distance type: Euclidean
 - Number of clusters: 11
 - Centroids initialisation method: Random
- Graphs:** Two line graphs are shown:
 - Loss:** A red line graph showing the loss decreasing from approximately 10 at iteration 0 to near zero by iteration 10.
 - Duration (seconds):** A blue bar chart showing the duration of each iteration, with most iterations taking between 0 and 5 seconds, and one outlier at approximately 15 seconds.

Fig 43: Model details for “partb_model1”

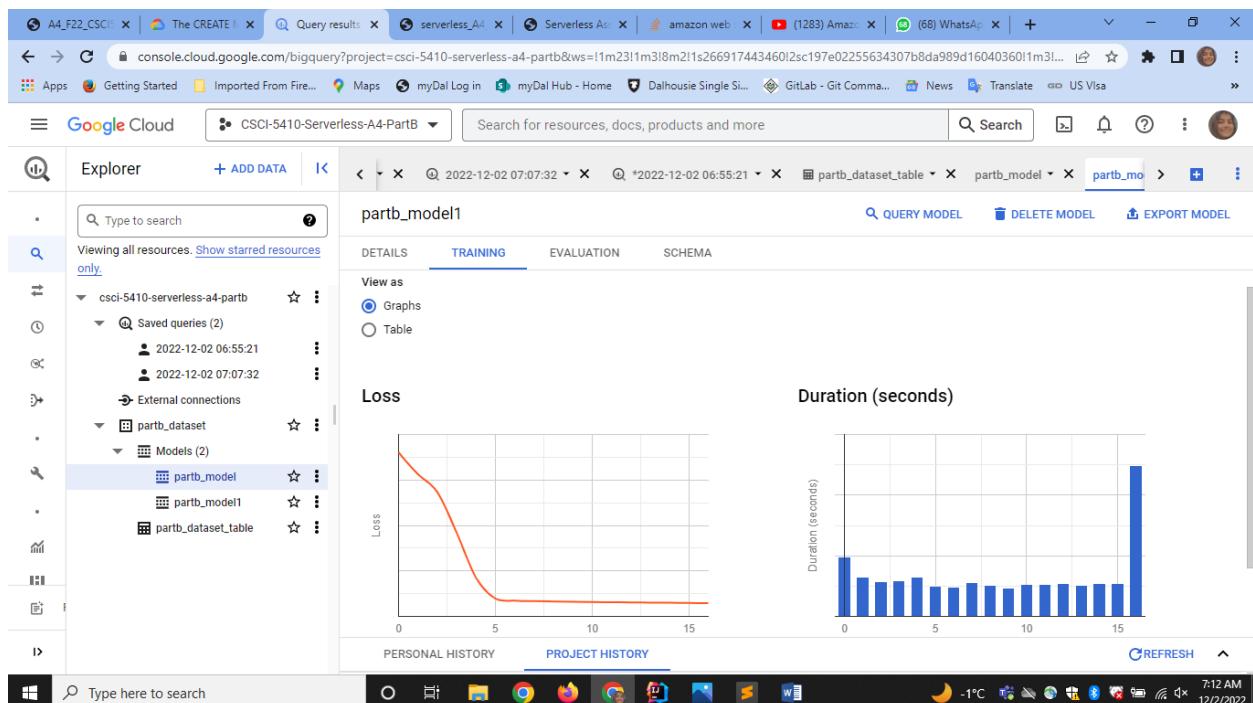


Fig 44: Training details for “partb_model1” – Graph view

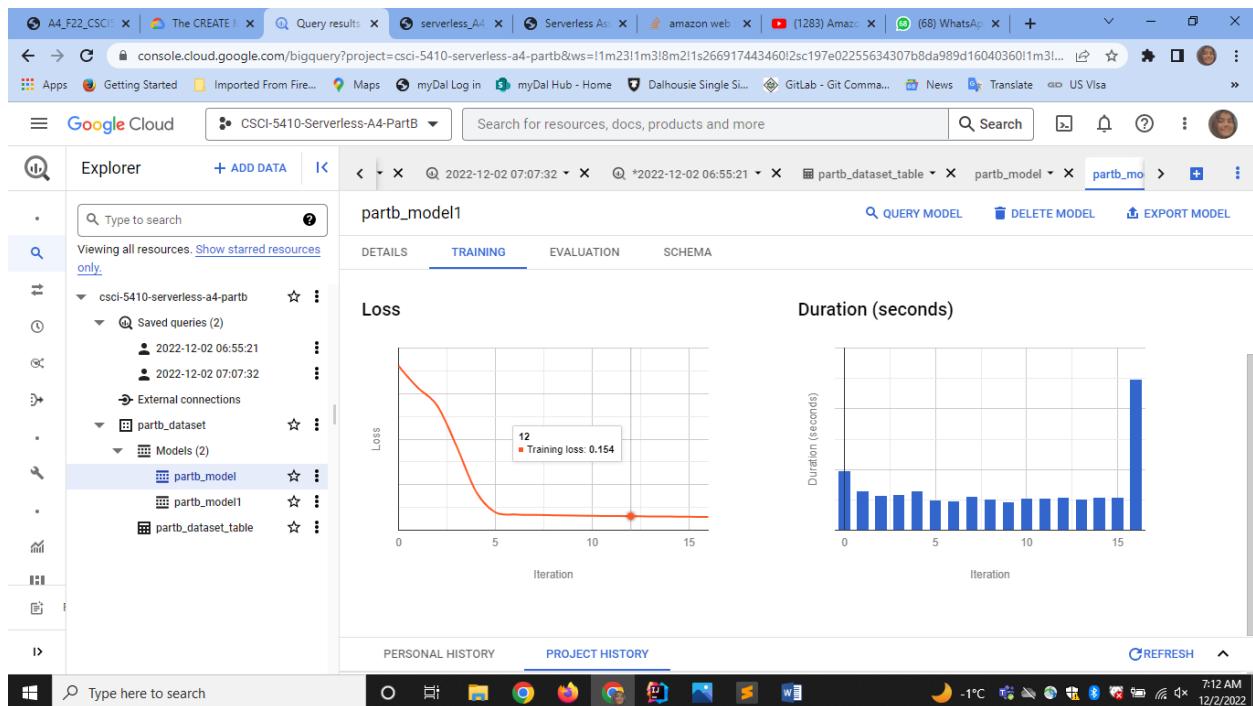


Fig 45: Training details for “partb_model1” – Graph view

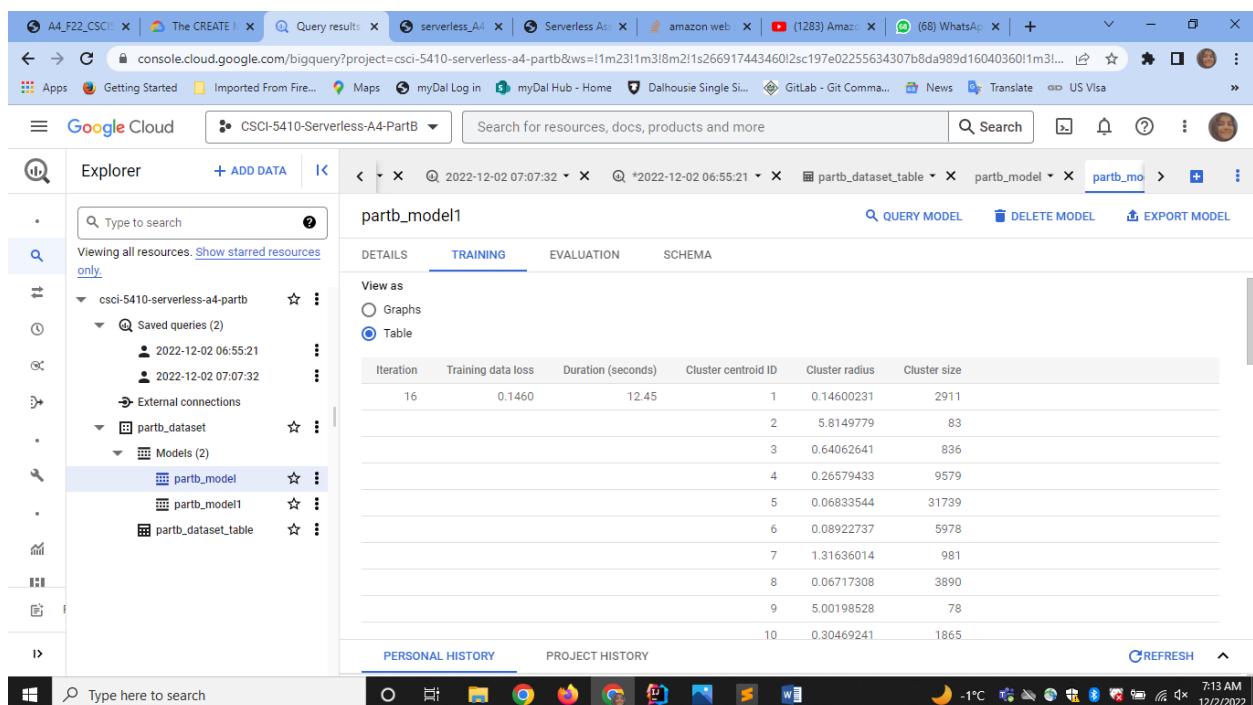


Fig 46: Training details for “partb_model1” – Table view

The screenshot shows the Google Cloud BigQuery interface. The left sidebar has an 'Explorer' section with a search bar and a list of resources under 'csci-5410-serverless-a4-partb'. The main area displays a table titled 'partb_model1' with three tabs: DETAILS, TRAINING (selected), EVALUATION, and SCHEMA. The TRAINING tab shows training statistics: 15 examples, 0.1471 error, and 2.75 AUC. The table lists 15 rows of training data with columns for index, error, and AUC. The bottom navigation bar includes 'PERSONAL HISTORY' and 'PROJECT HISTORY'.

	error	AUC
1	0.13955799	2896
2	5.8149779	83
3	0.59284892	920
4	0.26456765	9650
5	0.06737513	31641
6	0.08630026	5907
7	1.31258781	989
8	0.06519799	3686
9	5.00349455	78
10	0.29273559	2078
11	2.94457077	73
12	0.13318603	2958
13	5.8149779	83
14	0.56696557	1007
15	0.26342203	9736
16	0.0661664	31524
17	0.08373992	5725
18	1.30948348	998
19	0.06286818	3464
20	5.09330634	79
21	0.28005644	2343
22	5.06692044	94

Fig 47: Training details for “partb_model1” – Table view

This screenshot is identical to Fig 47, showing the same Google Cloud BigQuery interface and training details for 'partb_model1'. The table data is identical, showing 15 rows of training data with columns for index, error, and AUC.

	error	AUC
1	0.13955799	2896
2	5.8149779	83
3	0.59284892	920
4	0.26456765	9650
5	0.06737513	31641
6	0.08630026	5907
7	1.31258781	989
8	0.06519799	3686
9	5.00349455	78
10	0.29273559	2078
11	2.94457077	73
12	0.13318603	2958
13	5.8149779	83
14	0.56696557	1007
15	0.26342203	9736
16	0.0661664	31524
17	0.08373992	5725
18	1.30948348	998
19	0.06286818	3464
20	5.09330634	79
21	0.28005644	2343
22	5.06692044	94

Fig 48: Training details for “partb_model1” – Table view

The screenshot shows the Google Cloud BigQuery interface. The left sidebar has an 'Explorer' section with a tree view of datasets and models. The main area displays a table titled 'partb_model1' under the 'TRAINING' tab. The table has columns: ID, Loss, and Accuracy. The data rows are as follows:

ID	Loss	Accuracy
9	5.09330634	79
10	0.28005644	2343
11	2.85653844	84
13	0.1518	2.57
1	0.12506662	2966
2	5.8149779	83
3	0.523208	1089
4	0.26162728	9842
5	0.06467929	31396
6	0.08464737	5495
7	1.30580577	1010
8	0.06039865	3243
9	5.25893778	81
10	0.26398935	2685
11	2.4139145	111

Fig 49: Training details for “partb_model1” – Table view

The screenshot shows the Google Cloud BigQuery interface. The left sidebar has an 'Explorer' section with a tree view of datasets and models. The main area displays a table titled 'partb_model1' under the 'TRAINING' tab. The table has columns: ID, Loss, and Accuracy. The data rows are as follows:

ID	Loss	Accuracy
7	1.30580577	1010
8	0.06039865	3243
9	5.25893778	81
10	0.26398935	2685
11	2.4139145	111
12	0.1537	2.69
1	0.10882535	3385
2	5.8149779	83
3	0.48067677	1198
4	0.25839309	9963
5	0.06280106	31238
6	0.08581488	4867

Fig 50: Training details for “partb_model1” – Table view

Metrics

	Davies-Bouldin index	Mean squared distance
	0.9127	0.146

Numeric features

Centroid ID	Count	V_1	V_2
1	2,911	0.0003	0.0032
2	83	0.1433	0.0018
3	836	0.0008	0.0113
4	9,579	0.0030	0.0002

Fig 51: Evaluation details for “partb_model1”

Centroid ID	Count	V_1	V_2
1	2,911	0.0003	0.0032
2	83	0.1433	0.0018
3	836	0.0008	0.0113
4	9,579	0.0030	0.0002
5	31,739	0.0002	0.0000
6	5,978	0.0002	0.0019
7	981	0.0130	0.0011
8	3,890	0.0001	0.0011
9	78	0.0004	0.1465
10	1,865	0.0008	0.0054
11	61	0.0026	0.0383

Fig 52: Evaluation details for “partb_model1”

The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are several tabs: A4_F22_CSC, The CREATE, BigQuery - C, serverless_A4, Serverless A..., amazon web..., (1283) Amazon..., (68) WhatsApp..., and others. The main window displays the 'Explorer' view under 'Google Cloud'. On the left, a tree view shows a project named 'csci-5410-serverless-a4-partB' containing a 'Saved queries' folder with two entries: '2022-12-02 06:55:21' and '2022-12-02 07:07:32'. The '2022-12-02 07:07:32' query is selected. The right side of the screen shows the 'Query results' section for this query. The query details are as follows:

```

2 partb_dataset.partb_model1
3 OPTIONS (model type = 'KMFANS'
Processing location: US

```

JOB INFORMATION

Job ID	csci-5410-serverless-a4-partb:US.bqxxjob_3283c4ed_184d285b1a9
User	faizamatiya@gmail.com
Location	US
Creation time	2 Dec 2022, 07:08:08 UTC-4
Start time	2 Dec 2022, 07:08:08 UTC-4
End time	2 Dec 2022, 07:10:21 UTC-4
Duration	2 min 13 sec
Bytes processed	906.27 KB
Bytes billed	10 MB
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	csci-5410-serverless-a4-partb.partb_dataset.partb_model1

PERSONAL HISTORY and **PROJECT HISTORY** buttons are also present.

Fig 53: Job details for “partb_model1”

Step 10:

- First divide the provided dataset i.e. “SDey_FTP_input” into 25% test set and 75% training set.
- “SDey_FTP_input” contains total records of 58002.
- I made a separate file for 25% test set named as “SDey_FTP_input_25PercentTestSet” which contains 14500 records (25% of 58002) (Refer fig 54).
- And remaining 75% of the record i.e. 43502 record is in the other file named as “SDey_FTP_input_75PercentTrainingSet” (Refer fig 54).

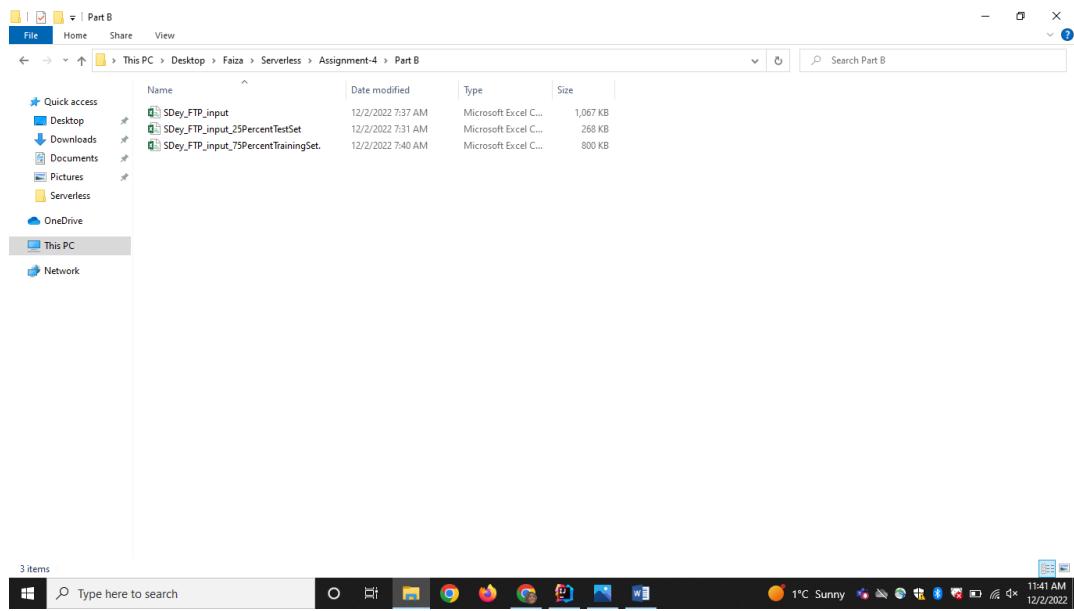


Fig 54: 3 different dataset

Step 11:

- **Create a dataset with 25% dataset** as test set by filling the details as shown in the fig 55 and 56.
- I used the “SDey_FTP_input_25PercentTestSet” dataset as shown in the fig 59.
- Now we need to add data. But in order to add data we need to create a table to upload file as shown in the fig 58, 59, 60 and 61.
- Refer fig 62, to see the partb_testSet_table has been created.

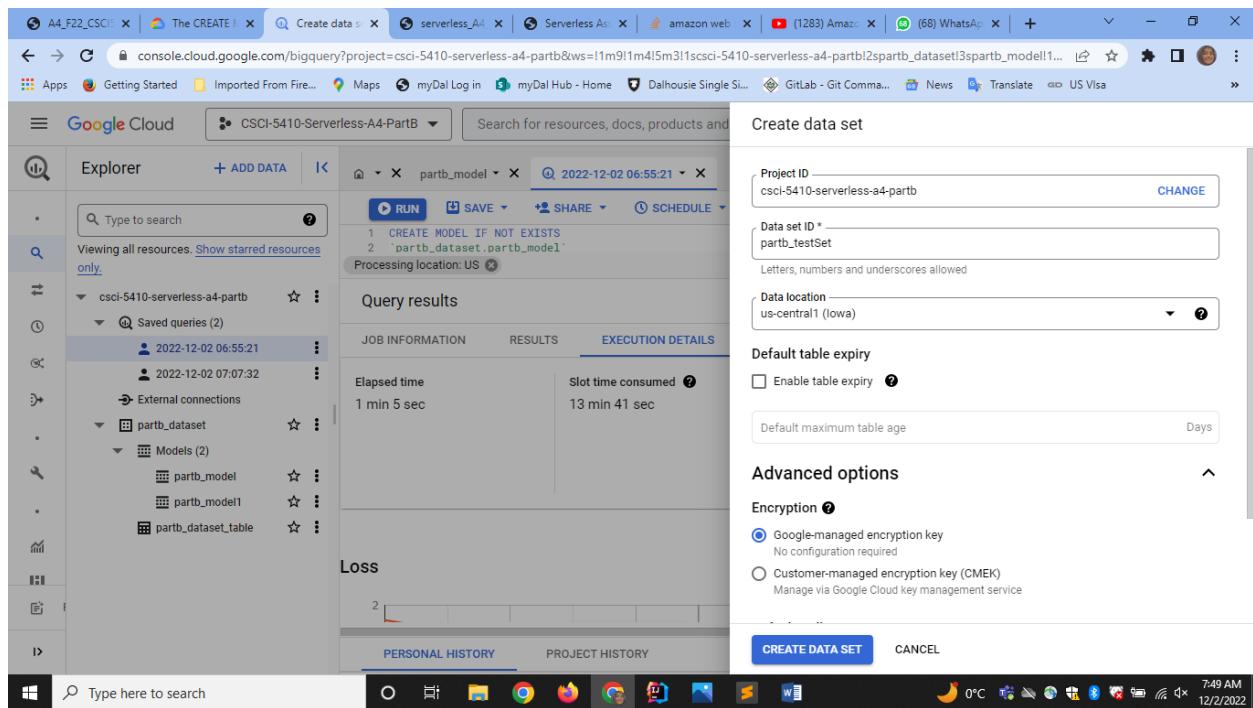


Fig 55: “Create data set” form

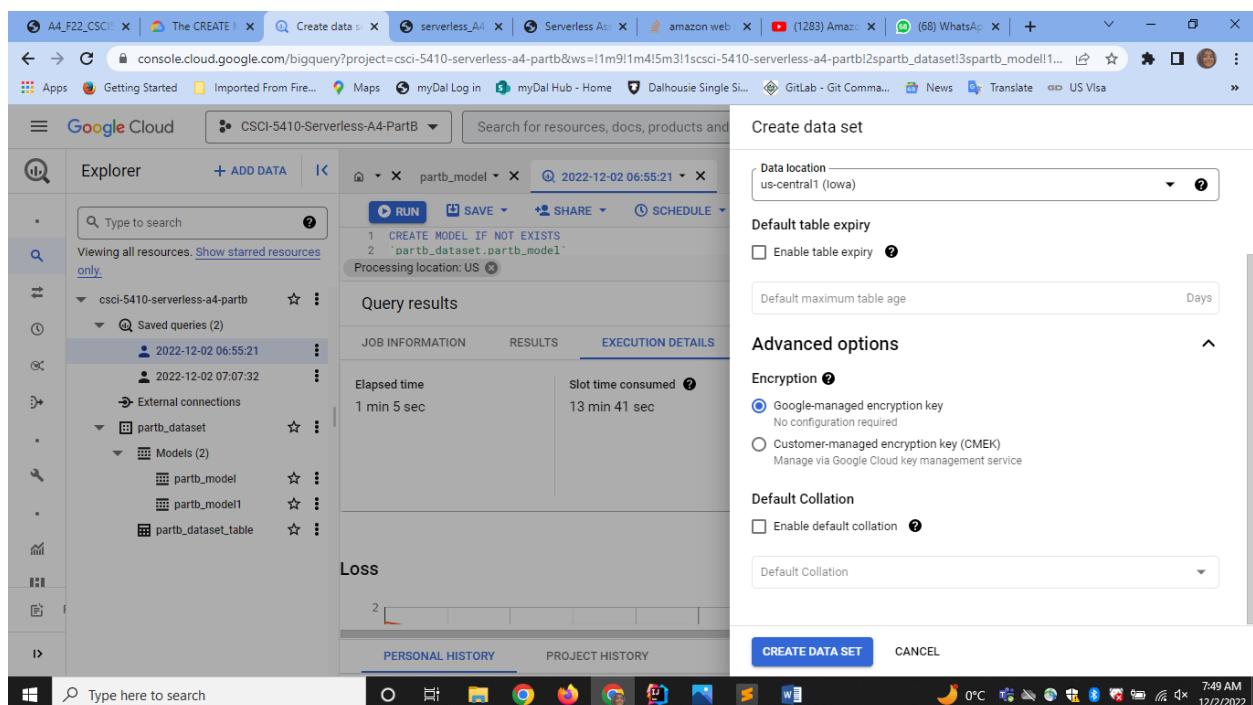


Fig 56: “Create data set” form

Google Cloud Explorer

Query results

Execution Details

Elapsed time: 1 min 5 sec

Slot time consumed: 13 min 41 sec

Stages:

- Preprocess: 0
- Train: 0
- Evaluate: 0

Training Iterations:

- Completed: 7
- Planned: 20

Loss Duration (seconds)

partb_testSet created. GO TO DATA SET

Fig 57: “partb_testSet” dataset has been created

Add data

Source

Search for data sources

Popular sources

- Local file: Upload a local file
- Google Cloud Storage: Google object storage service
- Connections to external data sources: Connection from BigQuery to an external data source

Additional sources

Viewing all 24 results.

Search for and star a project

Star a project by name

CLOSE

Fig 58: “Add data” form

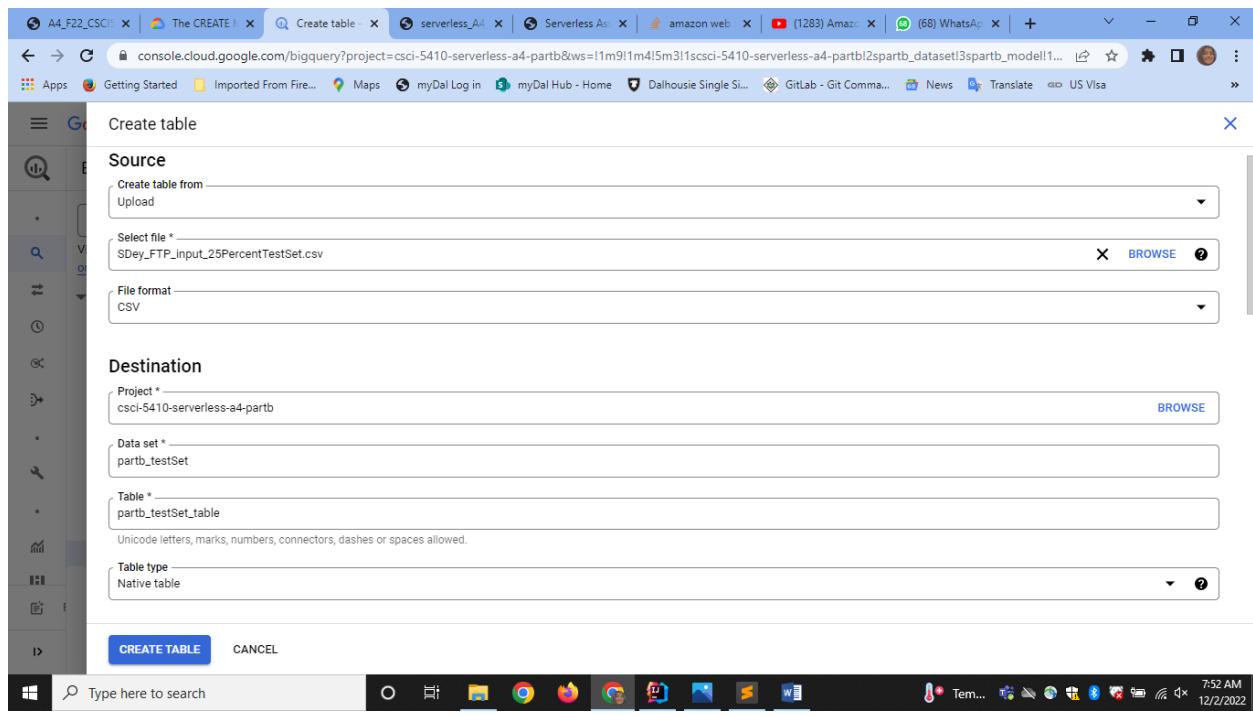


Fig 59: “Add data” form

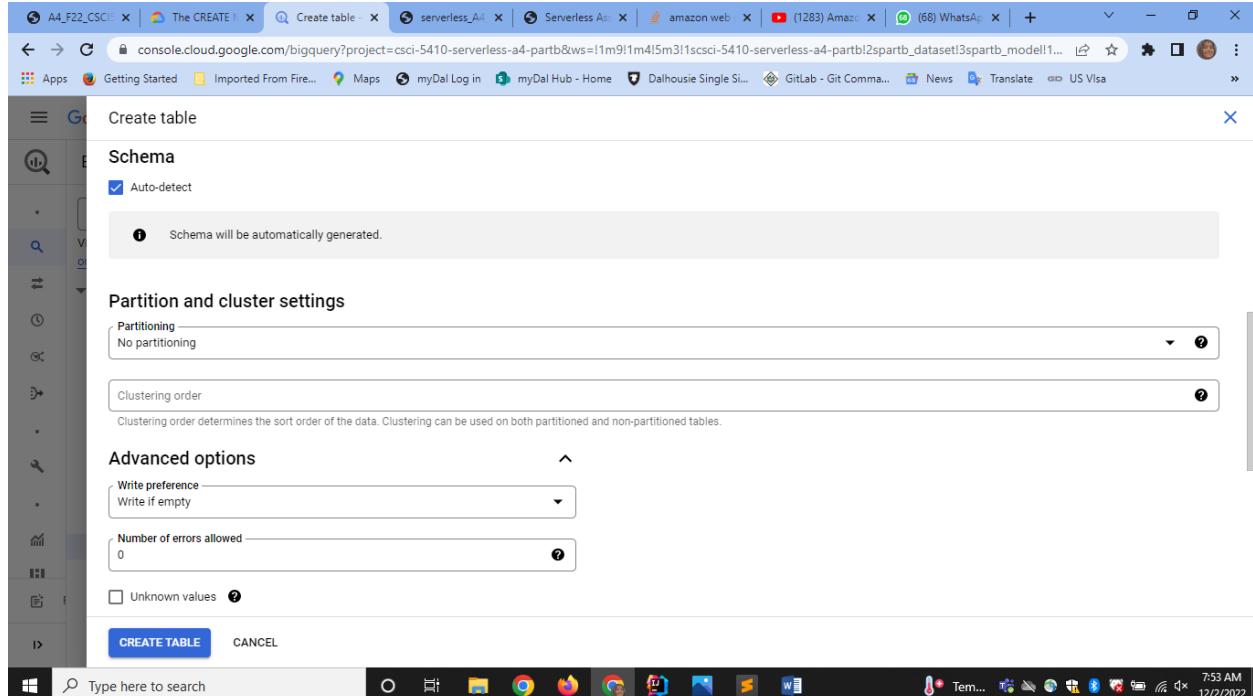


Fig 60: “Add data” form

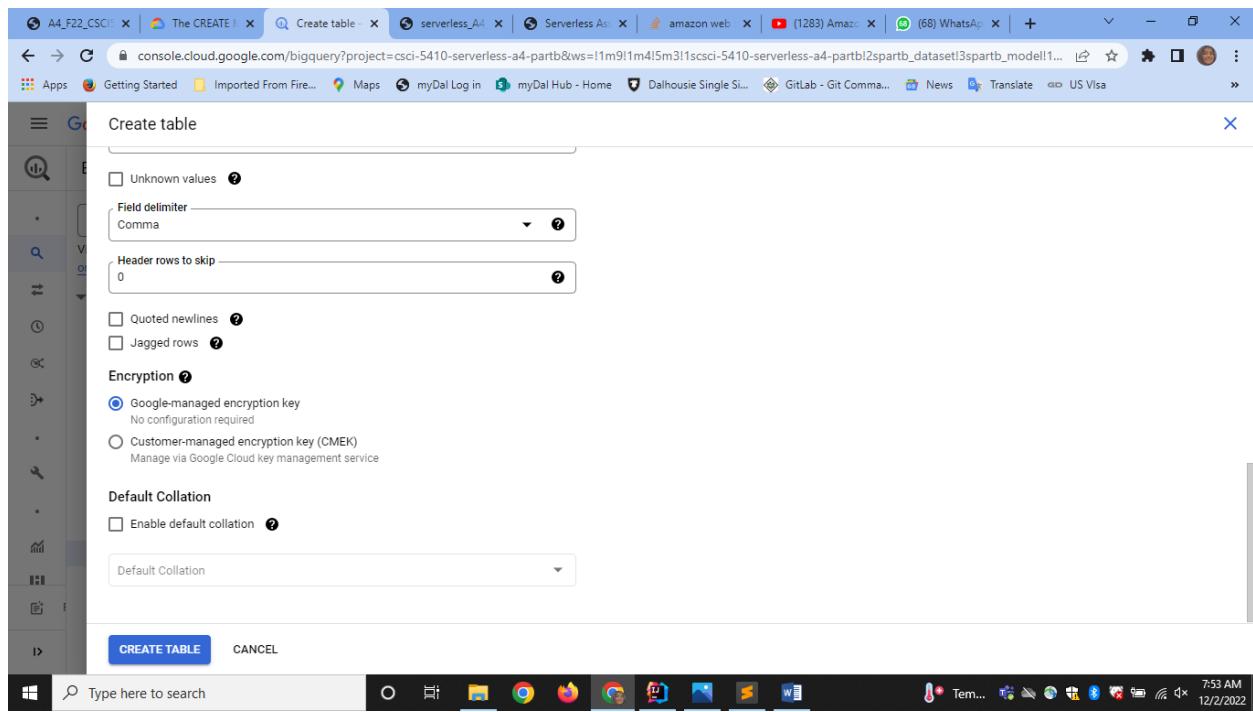


Fig 61: “Add data” form

The screenshot shows the Google Cloud Platform Explorer interface. A new dataset named 'partb_testSet' has been created under the 'csci-5410-serverless-a4-partb' project. The dataset info table includes:

Data set ID	csci-5410-serverless-a4-partb.partb_testSet
Created	2 Dec 2022, 07:49:36 UTC-4
Default table expiry	Never
Last modified	2 Dec 2022, 07:49:36 UTC-4
Data location	us-central1
Description	
Default collation	

A confirmation message 'partb_testSet_table created.' is displayed at the bottom of the screen.

Fig 62: “partb_testSet_table” has been created

Step 12:

- Change the settings for the query by clicking on the “more” option under “partb_testSet_table” as shown in the fig 63. Then go to “Query settings” as shown in the fig. 63.
- Fill the “Query settings” details as shown in the fig 64, 65 and 66.
- Keep the location same as the “partb_testSet” which is “us-central-1 (Iowa)”.

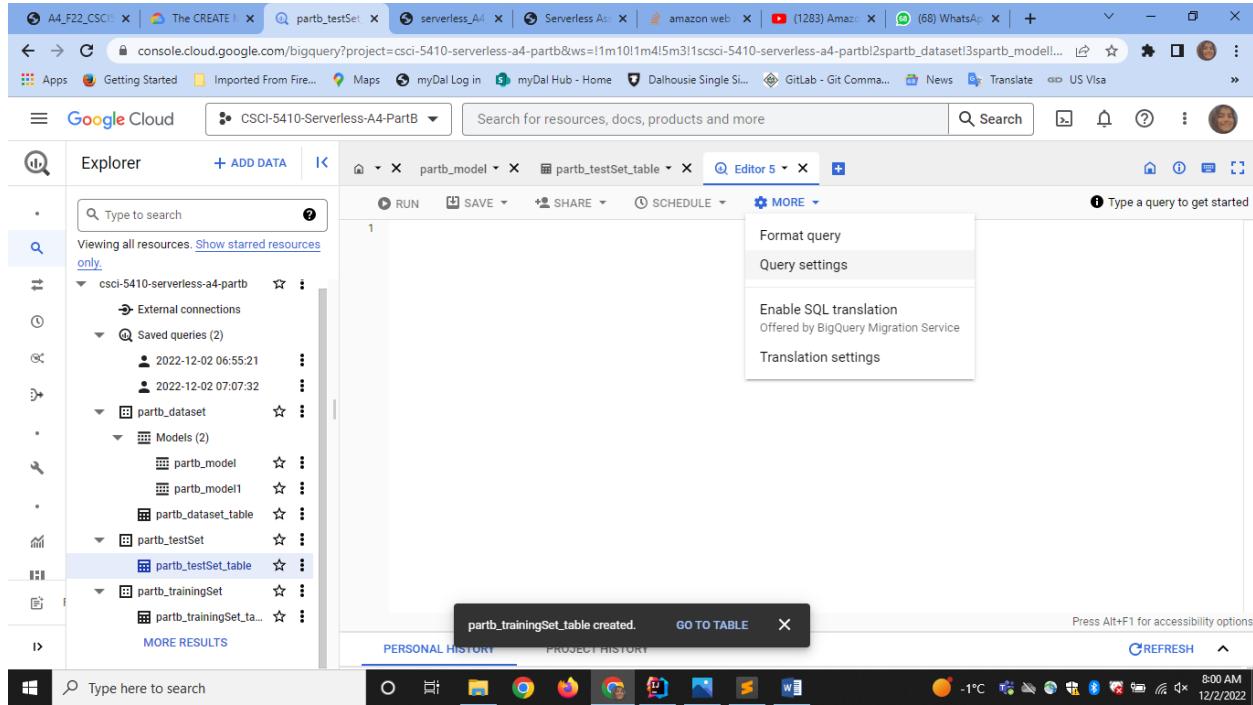


Fig 63: "Query settings" option

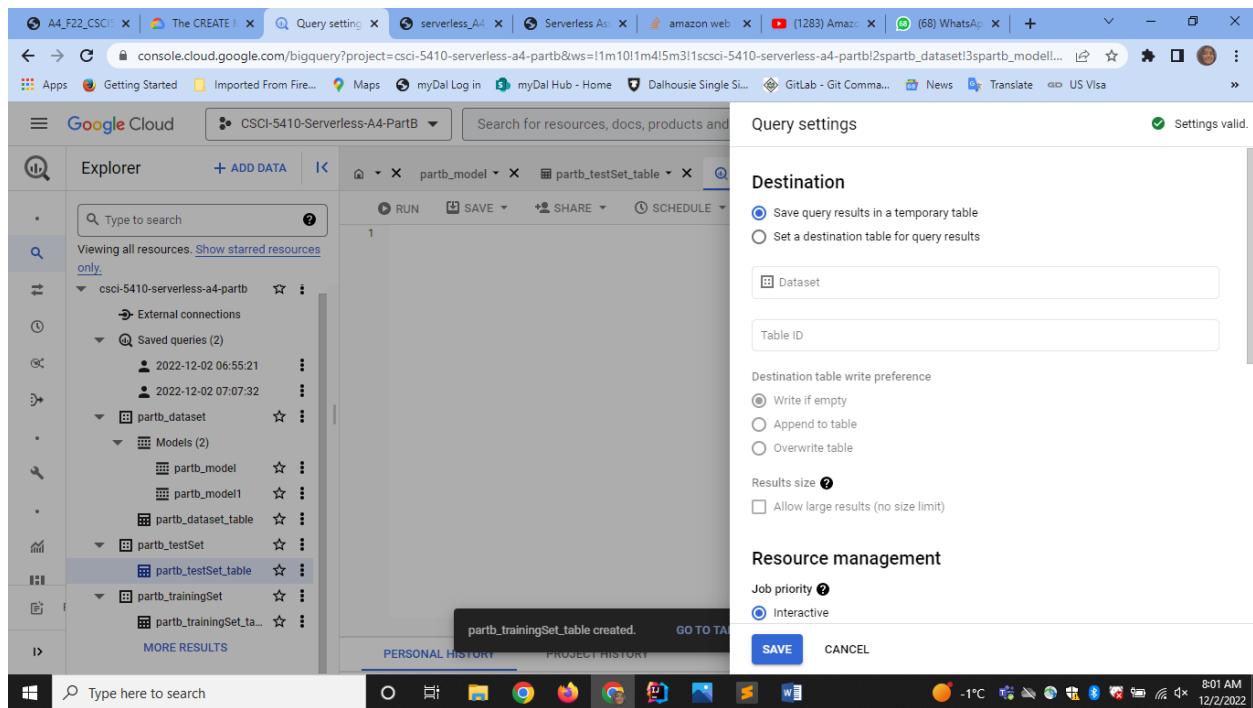


Fig 64: “Query settings” option

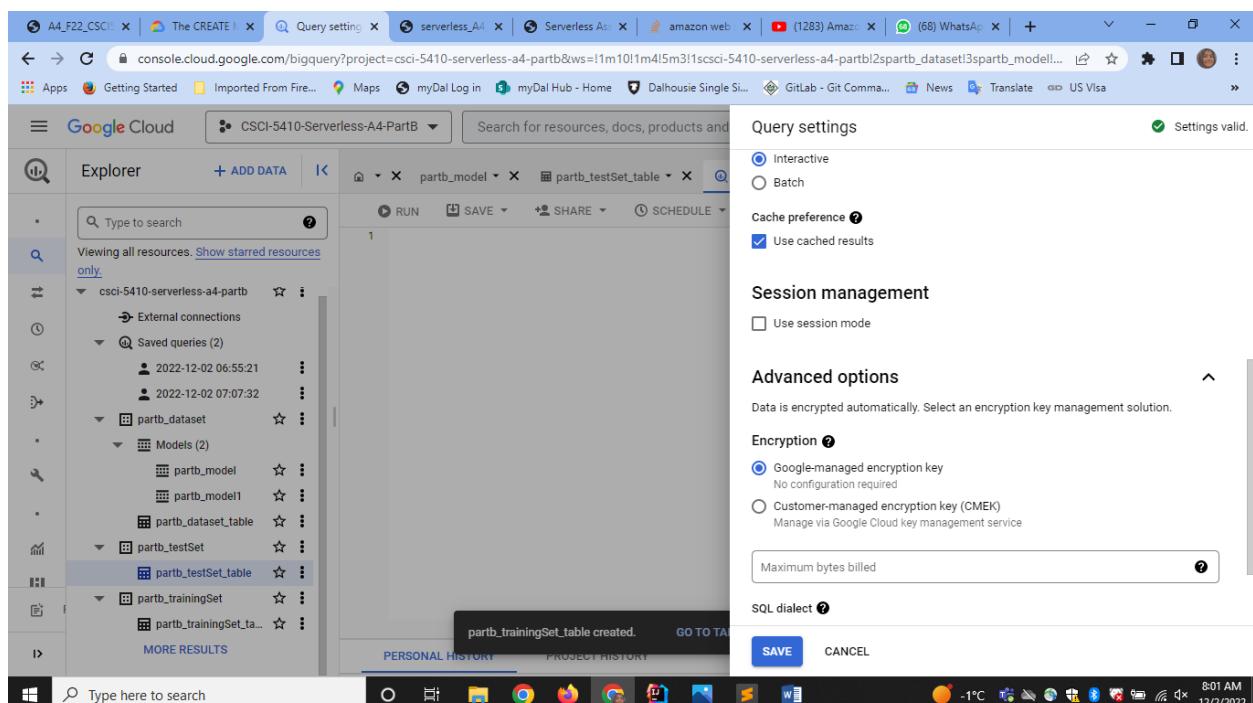


Fig 65: “Query settings” option

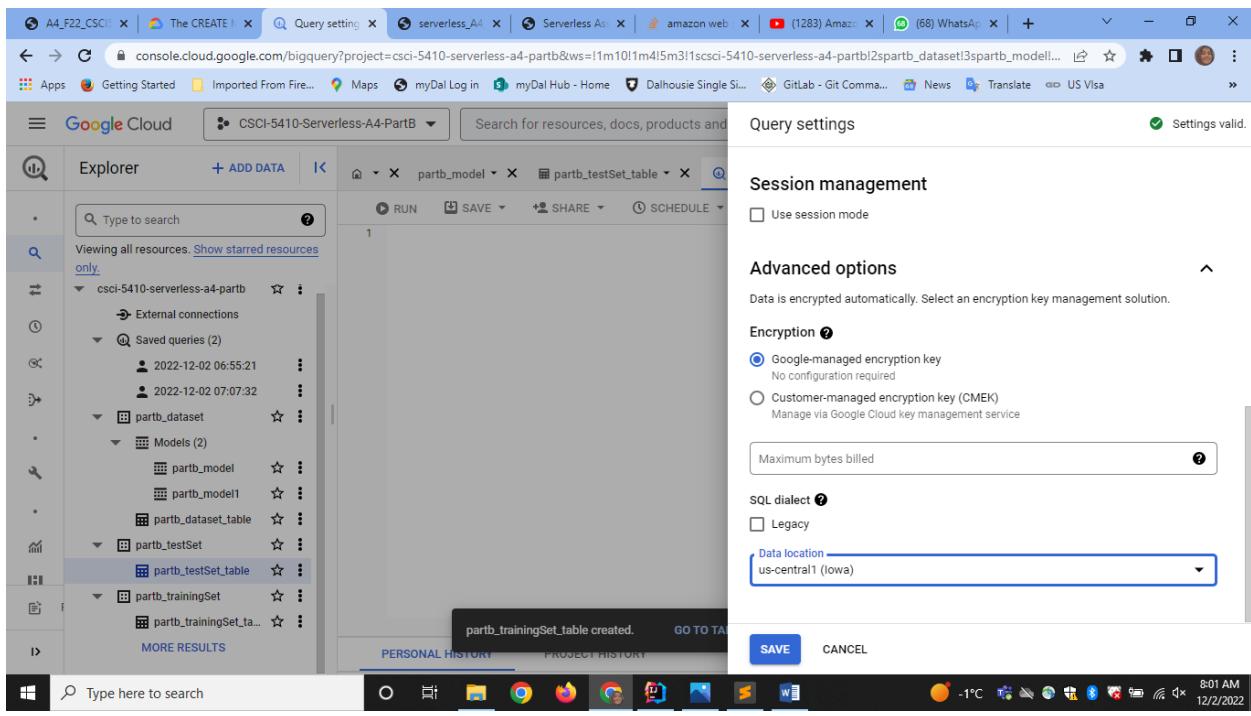


Fig 66: “Query settings” option

Step 13:

- I wrote a query to train the model as shown in the fig 67. The query creates a k-means model with 25 clusters using the “DISTANCE_TYPE” value of “EUCLIDEAN_DISTANCE” and standardize features [1].
- Fig 68 shows query running with the elapsed time.
- Fig 69 and 70 provides the model details which is been created after query completes its execution.
- Fig 71, 72, 73, 74, 75 and 76 gives the graphical view, table view of the results.
- Fig 77, 78 and 79 represents evaluation details of the “partb_model3”.
- Fig 53 represents job details of the “partb_model3”.

The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays the project structure under 'partb_dataset'. In the main area, a query is being run to create a K-Means model named 'partb_model' from the 'partb_testSet' dataset. The code is as follows:

```

1 CREATE MODEL IF NOT EXISTS
2 'partb_testSet.partb_model3'
3 OPTIONS (model_type = 'KMEANS',
4 NUM_CLUSTERS=25,
5 KMEANS_INIT_METHOD='RANDOM',
6 DISTANCE_TYPE='EUCLIDEAN',
7 STANDARDIZE_FEATURES = TRUE)
8 AS
9 SELECT * FROM
10 'partb_testSet.partb_testSet_table'
11
12 --Reference:
13 --https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create

```

The status bar at the bottom indicates the job is running with a duration of 5.9 seconds.

Fig 67: Query to create model

The screenshot shows the Google Cloud BigQuery interface with the same query running. The status bar at the bottom indicates the job is running with a duration of 5.9 seconds. The 'EXECUTION DETAILS' tab is selected, showing the following metrics:

Elapsed time	Slot time consumed	Stages	Training iterations
5 sec	0 sec	Preprocess 0	Completed: 0 Planned: 0

Fig 68: Query running

The screenshot shows the Google Cloud Platform interface for a K-Means model named 'partb_model3'. The left sidebar displays the 'Explorer' view with a tree structure of datasets and models. The main panel shows the 'DETAILS' tab for 'partb_model3', which is a K-Means model located in 'us-central1'. The 'Model details' section includes fields for Model ID (csci-5410-serverless-a4-partb.partb_testSet.partb_model3), Description, Labels, Date created (2 Dec 2022, 08:18:48 UTC-4), Model expiry (Never), Date modified (2 Dec 2022, 08:18:48 UTC-4), Data location (us-central1), and Model type (KMEANS). The 'Training options' section lists optional parameters such as Max allowed iterations (20), Actual iterations (16), Early stop (true), Min relative progress (0.01), Distance type (Euclidean), Number of clusters (25), and Centroids initialisation method (Random).

Fig 69: Model details for “partb_model3”

This screenshot is identical to Fig 69, showing the Google Cloud Platform interface for the 'partb_model3' K-Means model. The 'DETAILS' tab is selected, displaying the same information: Model ID (csci-5410-serverless-a4-partb.partb_testSet.partb_model3), Model type (KMEANS), and Data location (us-central1). The 'Training options' section also shows the same parameters as in Fig 69.

Fig 70: Model details for “partb_model3”

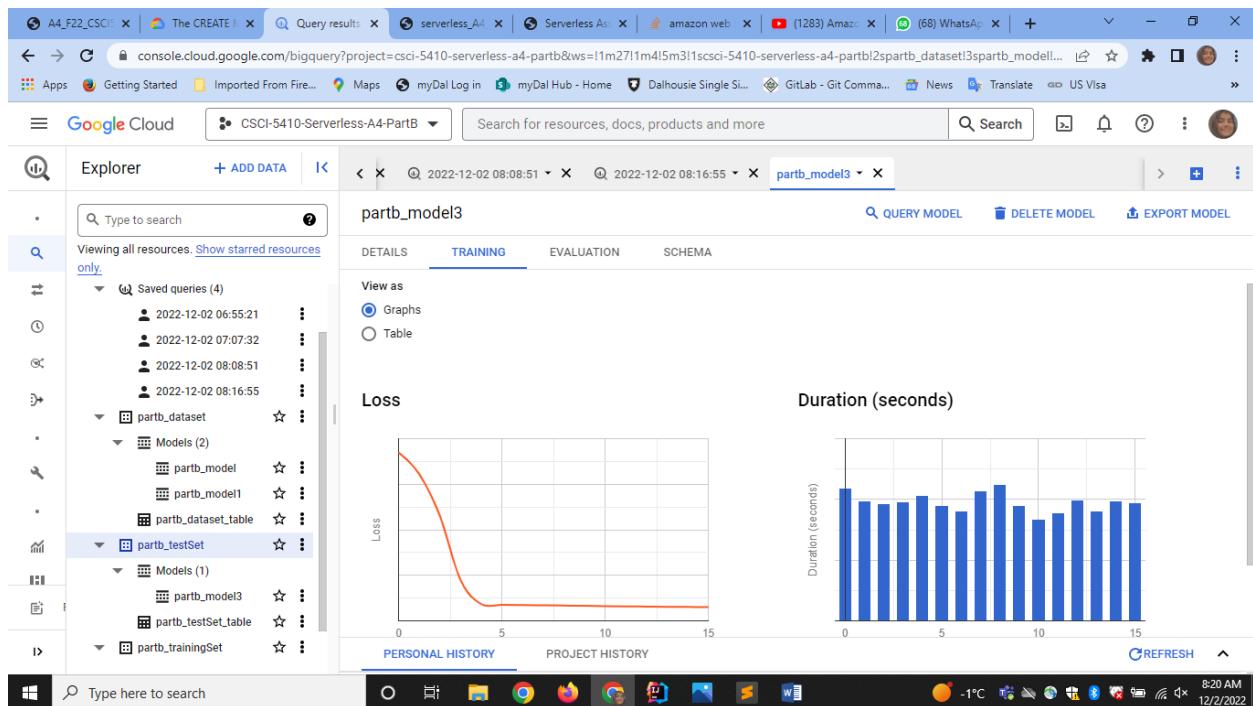


Fig 71: Training details for “partb_model3” – Graph view

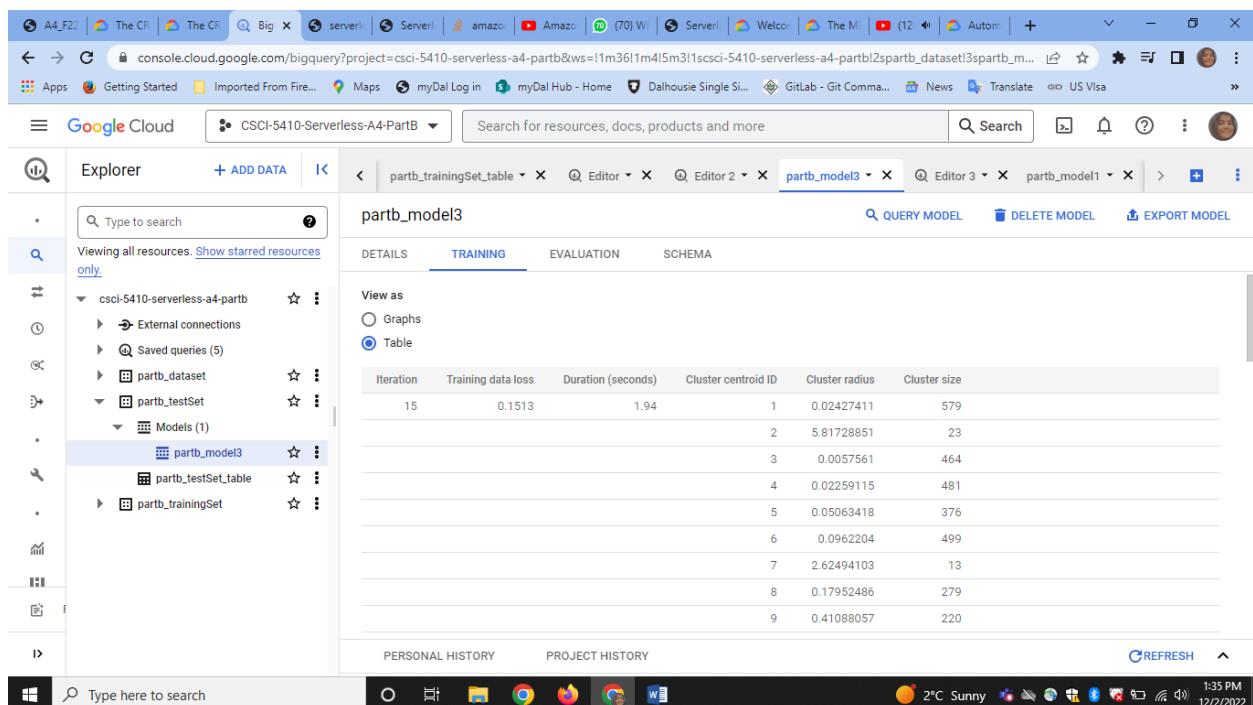


Fig 72: Training details for “partb_model3” – Table view

The screenshot shows the Google Cloud BigQuery interface. The top navigation bar includes tabs for 'A4_F22', 'The CR', 'Big', 'server...', 'Server...', 'amazon...', 'Amazo...', '(70) W...', 'Server...', 'Welco...', 'The M...', 'YouTube', 'Autom...', and a '+' button. Below the navigation bar is a search bar and a 'Search' button. The main area has tabs for 'partb_trainingSet_table', 'Editor', 'Editor 2', 'partb_model3' (which is active), 'Editor 3', and 'partb_model1'. The 'partb_model3' tab displays a table with columns 'DETAILS', 'TRAINING', 'EVALUATION', and 'SCHEMA'. The 'TRAINING' tab is selected, showing 23 rows of data. The table includes columns for ID, Training Loss, Evaluation Loss, and Model ID. The bottom of the interface shows 'PERSONAL HISTORY' and 'PROJECT HISTORY' buttons, and a 'REFRESH' button.

DETAILS	TRAINING	EVALUATION	SCHEMA
	10	0.03218753	1292
	11	2.57934835	15
	12	0.01378295	448
	13	0.01954347	191
	14	0.43540806	154
	15	0.16169961	515
	16	6.78230125	24
	17	0.01416309	469
	18	0.00283811	5437
	19	0.06108363	238
	20	0.03150882	793
	21	0.02171158	552
	22	0.1341323	377
	23	0.0093875	268

Fig 73: Training details for “partb_model3” – Table view

This screenshot is identical to Fig 73, showing the Google Cloud BigQuery interface for the 'partb_model3' table. The 'TRAINING' tab is selected, displaying 23 rows of training data. The columns are 'DETAILS', 'TRAINING', 'EVALUATION', and 'SCHEMA'. The data rows show various training parameters and model IDs.

DETAILS	TRAINING	EVALUATION	SCHEMA
	23	0.0093875	268
	24	0.35585506	101
	25	0.07043114	691
	14	0.1523	1.98
	1	0.02129581	548
	2	5.81728851	23
	3	0.00570924	445
	4	0.02393538	470
	5	0.04744865	371
	6	0.08891893	515
	7	2.57011379	14
	8	0.17206318	296
	9	0.39903385	241
	10	0.03010719	1251
	11	2.55907003	17

Fig 74: Training details for “partb_model3” – Table view

The screenshot shows the Google Cloud BigQuery interface. The top navigation bar includes tabs for 'A4_F22', 'The CR', 'Big', 'server...', 'Server...', 'amazon...', 'Amazo...', '(70) W...', 'Server...', 'Welco...', 'The M...', 'YouTube', 'Autom...', and a '+' button. Below the navigation bar is a toolbar with icons for 'Apps', 'Getting Started', 'Imported From Fire...', 'Maps', 'myDal Log in', 'myDal Hub - Home', 'Dalhousie Single Si...', 'GitLab - Git Comm...', 'News', 'Translate', and 'US Visa'. The main area has a header 'Google Cloud' and a search bar 'Search for resources, docs, products and more'. A sub-header 'CSCI-5410-Serverless-A4-PartB' is displayed above the main content. The main content area is titled 'partb_model3' and shows a table with four columns: DETAILS, TRAINING, EVALUATION, and SCHEMA. The TRAINING tab is selected, displaying 24 rows of training data. The table has a header row with columns '11', '2.55907003', and '17'. The data rows are numbered 11 through 24, each with three numerical values. Below the table are buttons for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button. On the left side, there is an 'Explorer' sidebar with a tree view of resources under 'csci-5410-serverless-a4-partb', including 'External connections', 'Saved queries (5)', 'partb_dataset', 'partb_testSet', and 'Models (1)' which contains 'partb_model3'. The bottom of the screen shows a Windows taskbar with icons for File Explorer, Task View, Start, Taskbar settings, and various pinned applications. The system tray shows the date and time as '1:39 PM 12/2/2022'.

Fig 75: Training details for “partb_model3” – Table view

This screenshot is identical to Fig 75, showing the Google Cloud BigQuery interface for the 'partb_model3' table. It displays the same 24 rows of training data with columns '11', '2.55907003', and '17'. The interface includes the same navigation bar, toolbar, and sidebar as Fig 75. The bottom of the screen shows a Windows taskbar with icons for File Explorer, Task View, Start, Taskbar settings, and various pinned applications. The system tray shows the date and time as '1:39 PM 12/2/2022'.

Fig 76: Training details for “partb_model3” – Table view

The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays the 'Explorer' section with various saved queries and datasets. The main panel is titled 'partb_model3' under the 'EVALUATION' tab. It shows two sections: 'Metrics' and 'Numeric features'. The 'Metrics' section includes Davies-Bouldin index (0.9525) and Mean squared distance (0.1513). The 'Numeric features' section lists 'V_1, V_2' as selected features, with a table showing Centroid ID, Count, V_1, and V_2 values for centroids 1, 2, and 3.

Centroid ID	Count	V_1	V_2
1	579	0.0000	0.0021
2	23	0.0003	0.1692
3	464	0.0000	0.0002

Fig 77: Evaluation details for “partb_model3”

This screenshot shows the same Google Cloud BigQuery interface for 'partb_model3' evaluation. The 'Numeric features' table has been updated to show a larger dataset with 17 rows, each containing Centroid ID, Count, V_1, and V_2 values. The results are identical to those shown in Fig 77, indicating no significant change in the model's performance metrics.

Centroid ID	Count	V_1	V_2
4	481	0.0001	0.0016
5	376	0.0001	0.0027
6	499	0.0002	0.0039
7	13	0.0004	0.0363
8	279	0.0004	0.0063
9	220	0.0102	0.0006
10	1,292	0.0018	0.0000
11	15	0.0336	0.0012
12	448	0.0000	0.0018
13	191	0.0000	0.0007
14	154	0.0006	0.0114
15	515	0.0048	0.0002
16	24	0.1675	0.0061
17	160	0.0000	0.0000

Fig 78: Evaluation details for “partb_model3”

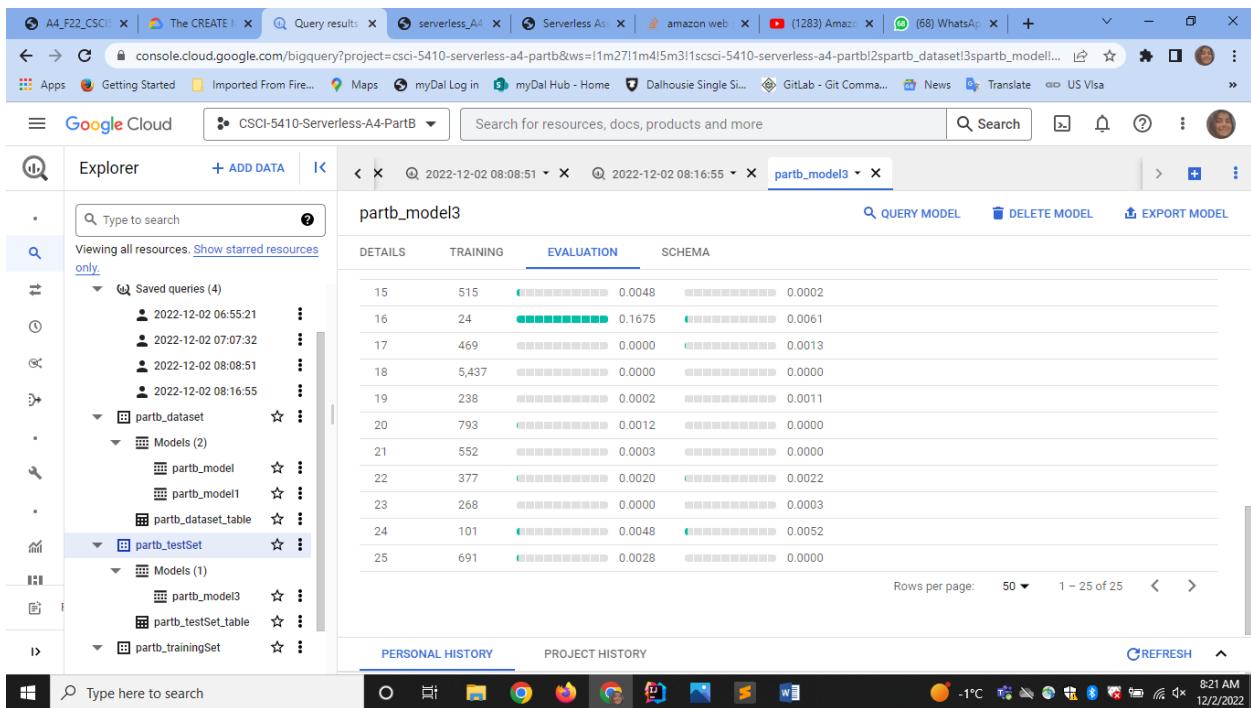


Fig 79: Evaluation details for “partb_model3”

Step 14:

- **Create a dataset with 75% dataset** as test set by filling the details as shown in the fig 80 and 81.
- The dataset “partb_trainingSet” created as shown in the fig 82.
- I used the “SDey_FTP_input_75PercentTrainingSet” dataset as shown in the fig 84.
- Now we need to add data. But in order to add data we need to create a table to upload file as shown in the fig 83, 84, 85 and 86.
- After clicking on the “create table” as shown in the fig 86, the partb_trainingSet_table is created.

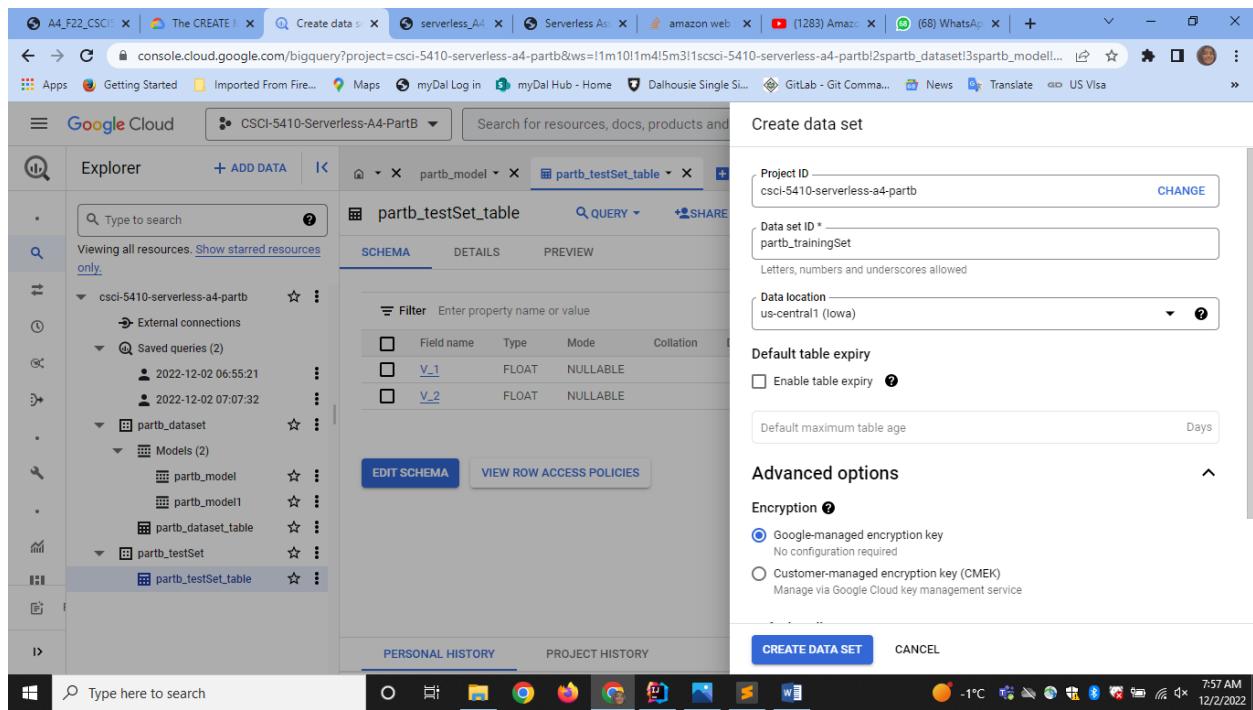


Fig 80: “Create data set” form

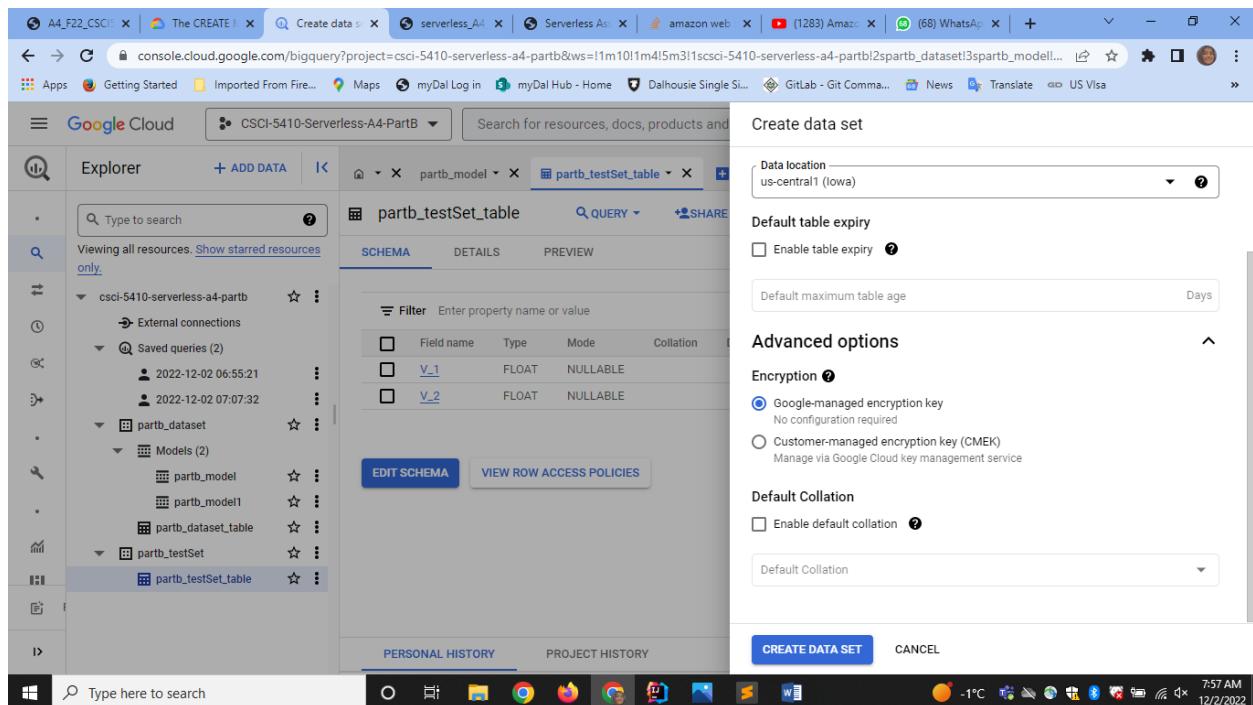


Fig 81: “Create data set” form

The screenshot shows the Google Cloud BigQuery interface. In the left sidebar, under the 'Explorer' tab, there is a tree view of datasets and tables. A context menu is open over the 'partb_trainingSet' dataset, with a tooltip indicating 'partb_trainingSet created.' Below the tree view, a modal window titled 'PERSONAL HISTORY' is visible. The main content area shows the schema for the 'partb_testSet_table' table, which has two columns: 'V_1' and 'V_2', both of type FLOAT and nullable. At the bottom of the screen, the Windows taskbar shows the date and time as 7:58 AM 12/2/2022.

Fig 82: “*partb_trainingSet*” dataset created

The screenshot shows the Google Cloud BigQuery interface with the 'ADD DATA' button selected in the top navigation bar. A modal window titled 'Add data' is open. The 'Source' section contains a search bar labeled 'Search for data sources'. Below it, the 'Popular sources' section lists three options: 'Local file' (with a 'Upload a local file' button), 'Google Cloud Storage' (described as a 'Google object storage service'), and 'Connections to external data sources' (described as 'Connection from BigQuery to an external data source'). The left sidebar shows the same dataset structure as Fig 82. The Windows taskbar at the bottom shows the date and time as 7:58 AM 12/2/2022.

Fig 83: “*Add data*” form

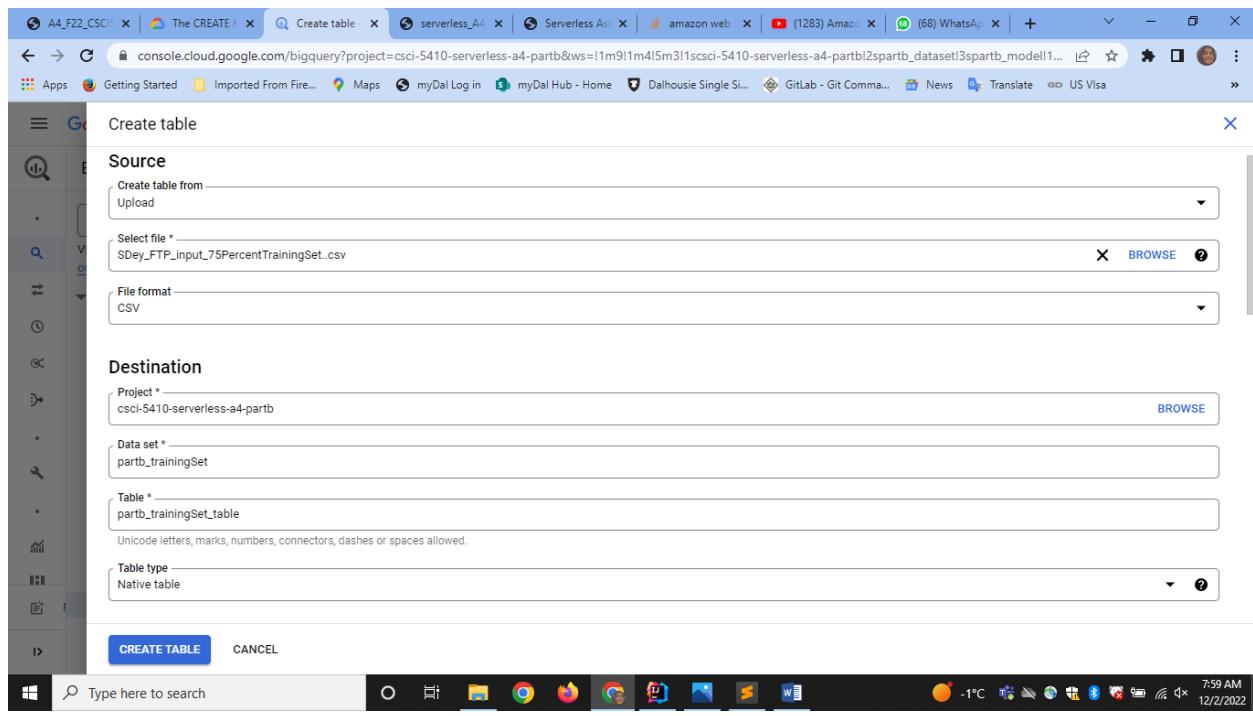


Fig 84: "Add data" form

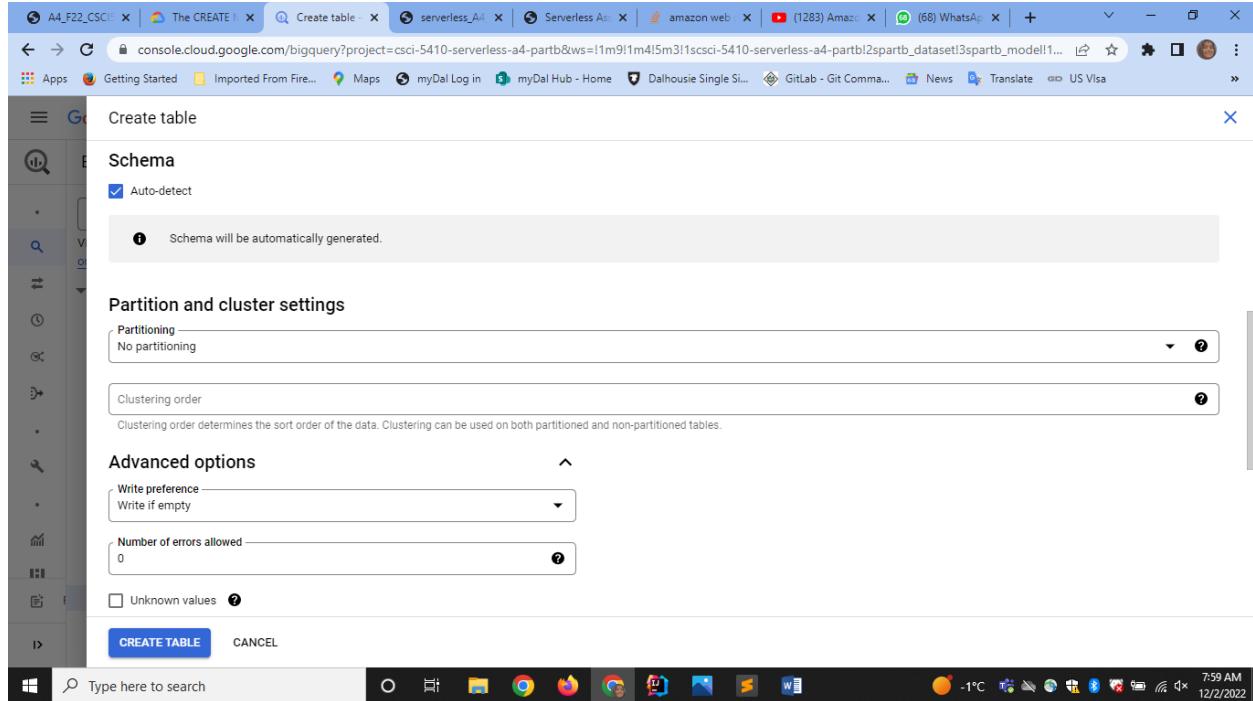


Fig 85: "Add data" form

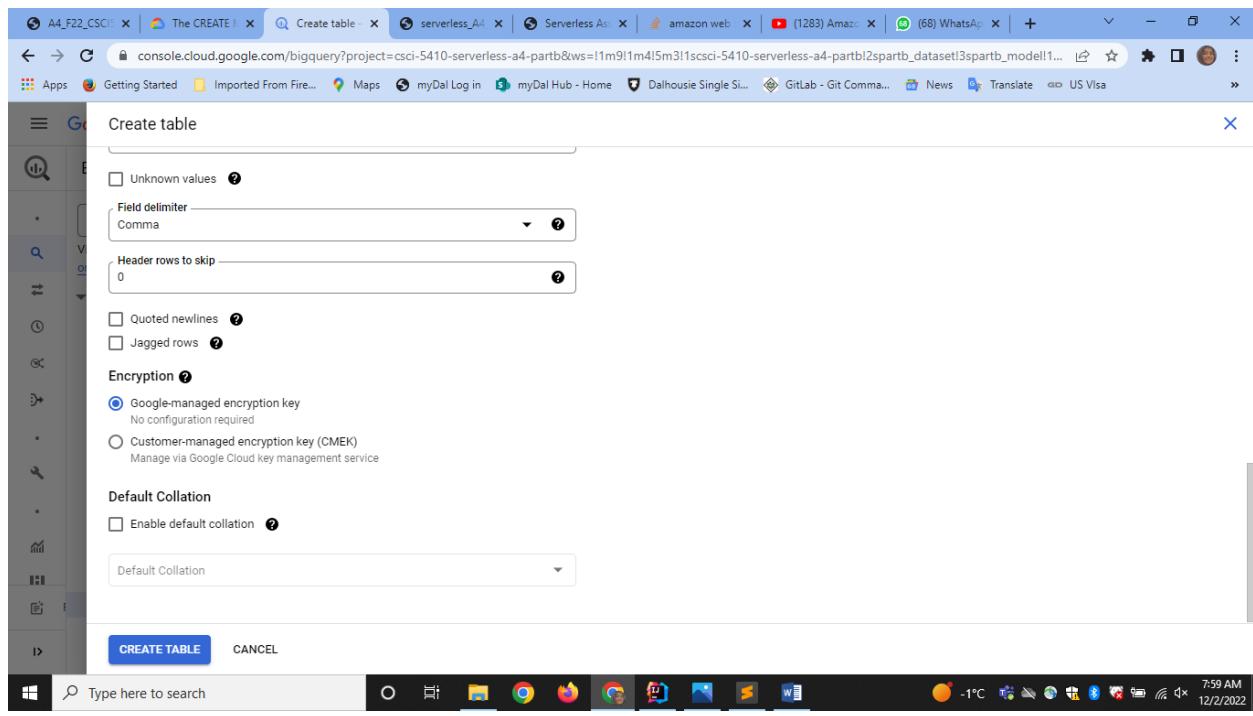


Fig 86: “Add data” form

Step 15:

- Keep the location same as the “partb_trainingSet” which is “us-central-1 (Iowa) as shown in the fig 87”.
- I wrote a query to train the model as shown in the fig 88. The query creates a k-means model with 25 clusters using the “DISTANCE_TYPE” value of “EUCLIDEAN_DISTANCE” and standardize features [1].
- Fig 89 shows query running with the elapsed time.
- Fig 90 and 91 provides the model details which is been created after query completes its execution.
- Fig 92, 93, 94, 95, 96 and 97 gives the graphical view, table view of the results.
- Fig 98 and 99 represents evaluation details of the “partb_model4”.

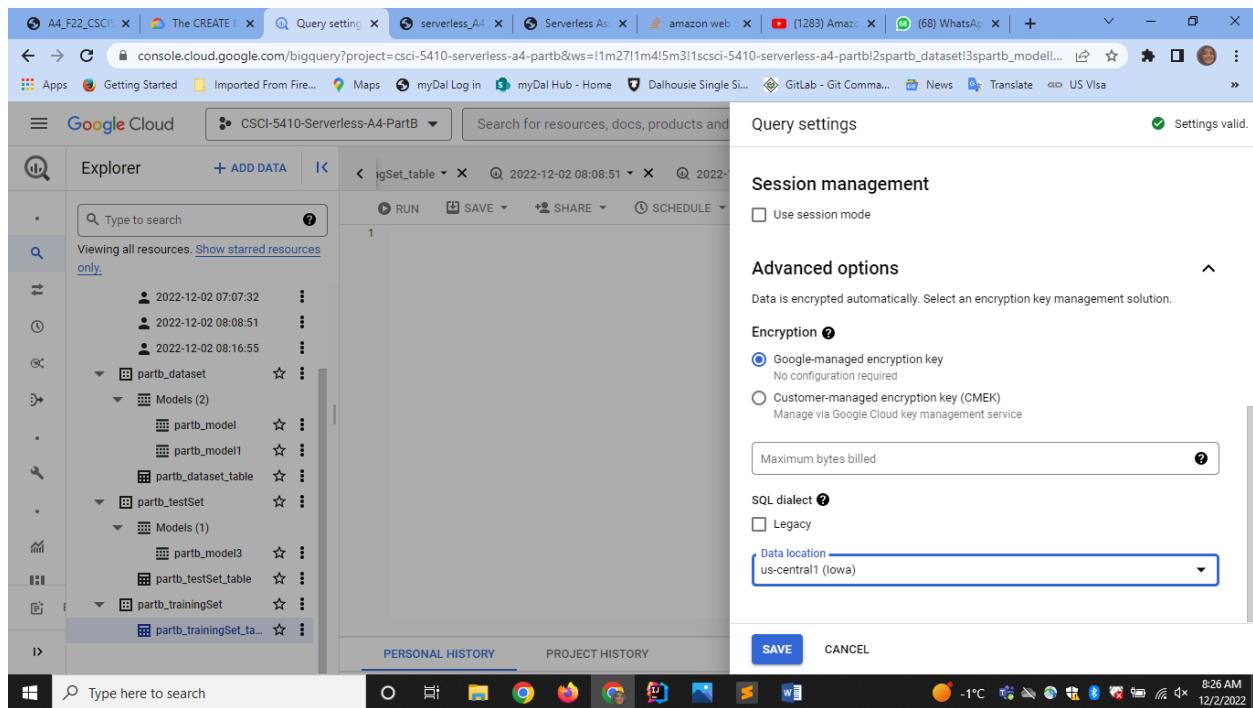


Fig 87: Query settings

```

1 CREATE MODEL IF NOT EXISTS
2   `partb_trainingSet.partb_model14`
3   OPTIONS (model_type = 'KMEANS',
4           NUM_CLUSTERS=17,
5           DISTANCE_TYPE='EUCLIDEAN',
6           STANDARDIZE_FEATURES = TRUE,
7           MAX_ITERATIONS = 6)
8   AS
9   SELECT *FROM
10   `partb_trainingSet.partb_trainingSet_table`
11
12 --Reference:
13 --https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create
  
```

The screenshot shows the BigQuery query editor with an unsaved query. The code above creates a K-Means model named 'partb_model14' from the 'partb_trainingSet_table'. The processing location is set to 'us-central1'.

Fig 88: Query to create model for training set

The screenshot shows the Google Cloud BigQuery interface. The top navigation bar has tabs for 'A4_F22_CSCI', 'The CREATE', 'Query results', 'serverless_A4', 'Serverless A...', 'amazon web...', '(1283) Amaz...', '(69) What...', and others. Below the tabs, the URL is `console.cloud.google.com/bigquery?project=csci-5410-serverless-a4-partb&ws=1m32l1m4l5m3lscsci-5410-serverless-a4-partb!2spartb_dataset!3spartb_model...!`. The main area shows the 'Google Cloud' logo and 'CSCI-5410-Serverless-A4-PartB' project. The 'Explorer' sidebar lists resources like '2022-12-02 07:07:32', '2022-12-02 08:08:51', '2022-12-02 08:16:55' (selected), and datasets 'partb_dataset' containing 'partb_model', 'partb_model1', 'partb_dataset_table', 'partb_testSet', 'partb_model3', 'partb_testSet_table', 'partb_trainingSet', and 'partb_trainingSet_ta...'. The central pane displays a query editor with the following SQL code:

```

1 CREATE MODEL IF NOT EXISTS `partb_trainingSet.partb_model4`
2   OPTIONS (model_type = 'KMEANS',
3           NUM_CLUSTERS=17,
4           DISTANCE_TYPE='EUCLEDIAN',
5           STANDARDIZE_FEATURES = TRUE,
6           MAX_ITERATIONS = 6)
7   AS
8   SELECT * FROM
9     `partb_trainingSet.partb_trainingSet_table`
10
11 Processing location: us-central1

```

The status bar indicates 'Query running (5.5 sec - Stage: Preprocess)'. Below the code, the 'Query results' section shows 'JOB INFORMATION', 'RESULTS' (selected), 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The execution details table includes columns for 'Elapsed time' (5 sec), 'Slot time consumed' (0 sec), 'Stages' (Preprocess, 0), and 'Training iterations' (Completed: 0, Planned: 0). The bottom of the screen shows a Windows taskbar with various icons.

Fig 89: Query running

The screenshot shows the Google Cloud BigQuery interface. The top navigation bar and URL are identical to Fig 89. The main area shows the 'Google Cloud' logo and 'CSCI-5410-Serverless-A4-PartB' project. The 'Explorer' sidebar lists resources like '2022-12-02 06:55:21', '2022-12-02 07:07:32', '2022-12-02 08:08:51', '2022-12-02 08:16:55' (selected), and datasets 'partb_dataset' containing 'partb_model', 'partb_model1', 'partb_dataset_table', 'partb_testSet', and 'Models (1)'. The central pane displays the 'partb_model4' model details page. The 'DETAILS' tab is selected, showing:

- Model type:** KMEANS
- Data location:** us-central1
- Model details:**
 - Model ID:** csci-5410-serverless-a4-partb.partb_trainingSet.partb_model4
 - Description:** (empty)
 - Labels:** (empty)
 - Date created:** 2 Dec 2022, 08:32:28 UTC-4
 - Model expiry:** Never
 - Date modified:** 2 Dec 2022, 08:32:28 UTC-4
 - Data location:** us-central1
 - Model type:** KMEANS
- Training options:** (described as optional parameters added to the script)

The bottom of the screen shows a Windows taskbar with various icons.

Fig 90: Model details for "partb_model4"

Google Cloud Explorer

Type to search

Viewing all resources. Show starred resources only.

2022-12-02 08:16:55

partb_dataset

Models (2)

partb_model

partb_model1

partb_dataset_table

partb_testSet

Models (1)

partb_model4

DETAILS TRAINING EVALUATION SCHEMA

Data location us-central1

Model type KMEANS

Training options

Training options are the optional parameters that were added in the script to create this model.

Max allowed iterations	6
Actual iterations	6
Early stop	true
Min relative progress	0.01
Distance type	Euclidean
Number of clusters	17
Centroids initialisation	Random method

PERSONAL HISTORY PROJECT HISTORY

Fig 91: Model details for “partb_model4”

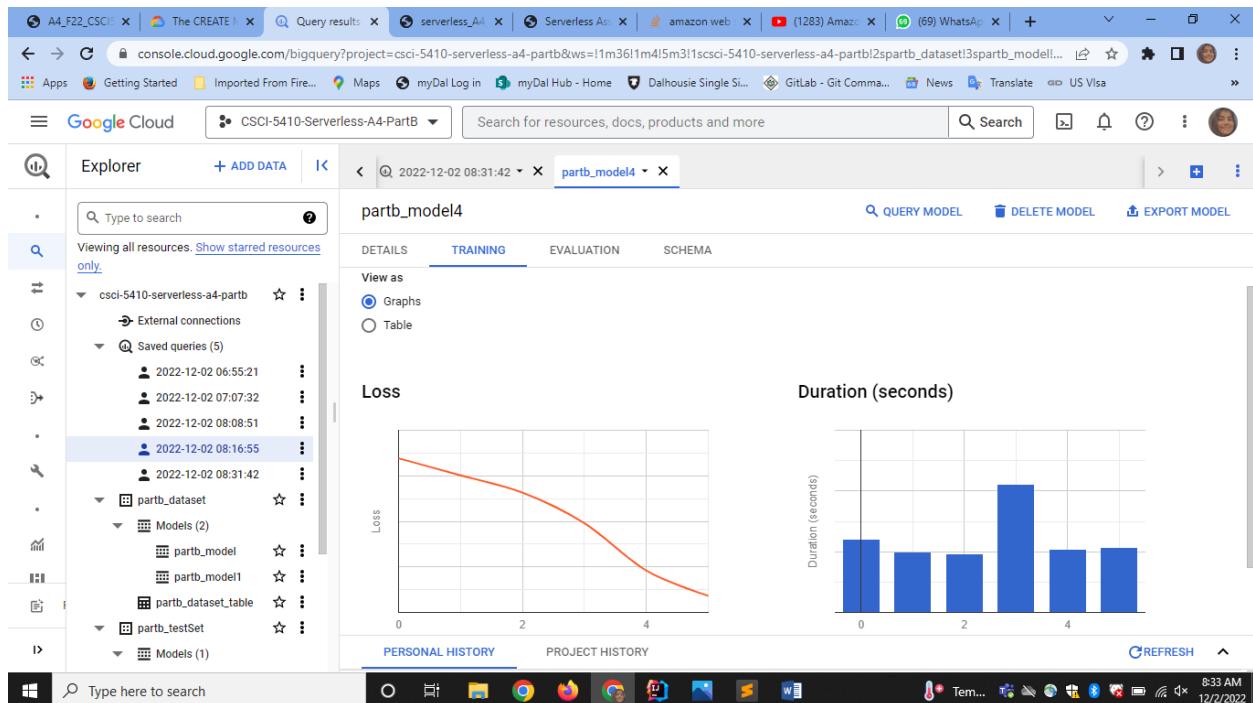


Fig 92: Training details for “partb_model4” – Graph view

The screenshot shows the Google Cloud Platform interface for a project named 'CSCI-5410-Serverless-A4-PartB'. In the center, a table titled 'partb_model4' is displayed under the 'TRAINING' tab. The table has columns: Iteration, Training data loss, Duration (seconds), Cluster centroid ID, Cluster radius, and Cluster size. The data shows 10 iterations with varying performance metrics and cluster sizes.

Iteration	Training data loss	Duration (seconds)	Cluster centroid ID	Cluster radius	Cluster size
5	0.1818	2.15	1	0.08699394	1528
			2	1.41369542	849
			3	0.02777212	20088
			4	0.02082418	792
			5	0.01933118	1279
			6	0.01767936	1119
			7	3.99823811	60
			8	0.02666607	548
			9	0.24187269	2210
			10	0.3628783	601

Fig 93: Training details for “partb_model4” – Table view

This screenshot is identical to Fig 93, showing the same training details for 'partb_model4' in BigQuery. The table structure and data values are the same, indicating no significant changes between the two captures.

Fig 94: Training details for “partb_model4” – Table view

The screenshot shows the Google Cloud Platform interface with the search bar set to 'partb_model4'. The main view displays the 'TRAINING' tab for the 'partb_model4' table. The table has four columns: DETAILS, TRAINING, EVALUATION, and SCHEMA. The TRAINING column contains numerical values such as 4, 0.4597, and 2.08. The EVALUATION column contains values like 1, 0.0596802, and 1751. The SCHEMA column contains values such as 2, 1.22959491, and 1155. The DETAILS column contains values from 3 to 14. The table also includes 'PERSONAL HISTORY' and 'PROJECT HISTORY' tabs at the bottom.

Fig 95: Training details for “partb_model4” – Table view

This screenshot is identical to Fig 95, showing the Google Cloud Platform interface with the search bar set to 'partb_model4'. It displays the 'TRAINING' tab for the 'partb_model4' table. The table structure and data are the same as in Fig 95, with columns for DETAILS, TRAINING, EVALUATION, and SCHEMA. The data rows range from 3 to 17, showing various performance metrics for the model training process.

Fig 96: Training details for “partb_model4” – Table view

The screenshot shows the Google Cloud BigQuery interface for a dataset named 'partb_model4'. The 'TRAINING' tab is selected. The table has four columns: index, value, count, and row_id. The data is as follows:

index	value	count	row_id
6	0.01609954	1115	
7	12.62321601	102	
8	0.02712017	537	
9	0.13456092	1947	
10	0.29472158	699	
11	0.10551848	651	
12	0.01368217	1640	
13	0.01858523	1437	
14	0.08165579	186	
15	0.09226978	6476	
16	11.15847416	192	

Fig 97: Training details for “partb_model4” – Table view

The screenshot shows the Google Cloud BigQuery interface for a dataset named 'partb_model4'. The 'EVALUATION' tab is selected. It displays the following metrics:

- Davies-Bouldin index: 1.1076
- Mean squared distance: 0.1818

Under the 'Numeric features' section, it says: "This table shows the centroid value for each feature. Use the select menu to view more numeric features." A dropdown menu shows 'Selected features: V_1, V_2'. The table below shows the centroid ID, count, and values for V_1 and V_2.

Centroid ID	Count	V_1	V_2
1	1,528	0.0000	0.0023
2	849	0.0009	0.0109
3	20,088	0.0000	0.0000
4	792	0.0000	0.0010

Fig 98: Evaluation details for “partb_model4”

Iteration	Training Examples	Evaluation Score	Schema
4	792	0.0000	0.0010
5	1,279	0.0000	0.0015
6	1,119	0.0000	0.0013
7	60	0.0008	0.1263
8	548	0.0000	0.0006
9	2,210	0.0001	0.0041
10	601	0.0030	0.0035
11	789	0.0018	0.0017
12	1,636	0.0000	0.0002
13	2,162	0.0000	0.0018
14	337	0.0009	0.0012
15	7,713	0.0020	0.0000
16	61	0.1045	0.0005
17	1,730	0.0072	0.0005

Fig 99: Evaluation details for “partb_model4”

Code/Query Script:

1) Creation of Model (partb_model) having dataset “SDeY FTP input.csv”

```
CREATE MODEL IF NOT EXISTS
`partb_dataset.partb_model`
OPTIONS (model_type = 'KMEANS',
NUM_CLUSTERS=5,
KMEANS_INIT_METHOD='RANDOM')
AS
SELECT *FROM
`partb_dataset.partb_dataset_table`
```

2) Evaluate model using “SDeY FTP input.csv” dataset

```
SELECT *
FROM
ML.TRAINING_INFO(MODEL `partb_dataset.partb_model`)
ORDER BY ITERATION
```

3) Creation of Model (partb_model1) having dataset “SDev FTP input.csv”

```
CREATE MODEL IF NOT EXISTS
`partb_dataset.partb_model1`
OPTIONS (model_type = 'KMEANS',
NUM_CLUSTERS=11,
KMEANS_INIT_METHOD='RANDOM',
DISTANCE_TYPE= 'EUCLIDEAN')
AS
SELECT *FROM
`partb_dataset.partb_dataset_table`
```

4) Creation of model using test dataset i.e. “SDev FTP input 25PercentTestSet.csv”:

```
CREATE MODEL IF NOT EXISTS
`partb_testSet.partb_model3`
OPTIONS (model_type = 'KMEANS',
NUM_CLUSTERS=25,
KMEANS_INIT_METHOD='RANDOM',
DISTANCE_TYPE='EUCLIDEAN',
STANDARDIZE_FEATURES = TRUE)
AS
SELECT *FROM
`partb_testSet.partb_testSet_table`
```

5) Creation of model using training dataset “SDev FTP input 75PercentTestSet”:

```
CREATE MODEL IF NOT EXISTS
`partb_trainingSet.partb_model4`
OPTIONS (model_type = 'KMEANS',
NUM_CLUSTERS=17,
DISTANCE_TYPE='EUCLIDEAN',
STANDARDIZE_FEATURES = TRUE,
MAX_ITERATIONS = 6)
AS
SELECT *FROM
`partb_trainingSet.partb_trainingSet_table`
```

Reference:

[1] “The CREATE MODEL statement for K-means models,” *Google Cloud*. [Online]. Available: <https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create-kmeans>. [Accessed: 02-Dec-2022].