



SCHOOL OF INFORMATICS & IT
Individual Solution

Student Name (Matric Number) : 2302276G
Tutorial Group : PC02
Tutor : Goh Rui Quan
Submission Date : 27/7/24

Declaration of Originality

I am the originator of this work and I have appropriately acknowledged all other original sources used as my references for this work.

I understand that Plagiarism is the act of taking and using the whole or any part of another person's work, including work generated by AI, and presenting it as my own.

I understand that Plagiarism is an academic offence and if I am found to have committed or abetted the offence of plagiarism in relation to this submitted work, disciplinary action will be enforced.

Declaration on the use of Generative AI tools for assignments

Describe how you have used Generative AI tools such as ChatGPT or Dall.E-2 in your assignment.

Show snapshots of the conversations with the AI tool (i.e., the prompts you used and the response you get from the AI tool).

<https://chatgpt.com/share/ae6db300-c143-4f03-9d81-6b6bc328197a>

Used it for steps on how to download the request library in AWS Lambda

<https://chatgpt.com/share/f47c0c9d-10db-4ad5-a0fb-97099f2fb843>

Asked ChatGPT for ideas on how to fix the error where the data does not show up in Athena.

How do you indicate the reference?

The content generated by AI tools are not retrievable except by the user who generated them, so they are considered non-recoverable sources. Although non-recoverable data or quotations in APA Style papers are usually cited as personal communications, with ChatGPT-generated text there is no person communicating. Quoting text from ChatGPT chat is therefore more like sharing the output of an algorithm, with a reference list entry and the corresponding in-text citation.

According to the official APA Style site, ChatGPT references should be cited as:

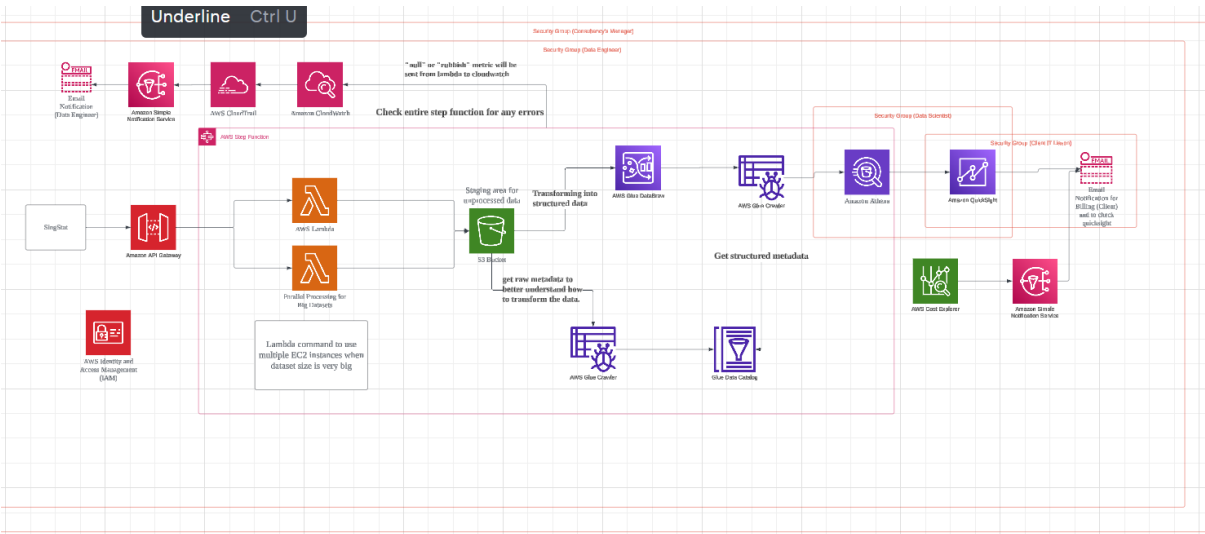
E.g. OpenAI. (2023). *ChatGPT* (Sep 25 version) [Large language model].

<https://chat.openai.com/chat>

Important Note:

- Do not copy answers produced by the AI tool in totality as it is considered as plagiarism.
- Do not rely on any information produced by the AI tool blindly. You should always verify the answer with other sources. Do not assume that these answers provided by the AI tool are correct.
- To achieve quality outputs from the AI tool, you should provide good prompt that is clear and specific. Be precise and provide context. Avoid asking open-ended questions.

Pipeline Overview:



Solution Implementation

Component	Setup, reason, additional notes (for data integrity, budget, etc)																								
API Gateway	<div><div>API Gateway > APIs</div><div><div>APIs (3/3)</div><div><div>Find APIs</div></div><table><tr><th>Name</th><th>Description</th><th>ID</th><th>Protocol</th><th>API endpoint type</th><th>Created</th></tr><tr><td>monthly_singstat_data</td><td></td><td>xik9d3xxa2</td><td>HTTP</td><td>Regional</td><td>2024-07-23</td></tr><tr><td>quarterly_singstat_data</td><td></td><td>d0zmb6i05</td><td>HTTP</td><td>Regional</td><td>2024-07-23</td></tr><tr><td>yearly_singstat_data</td><td></td><td>5w25jxhpb1</td><td>HTTP</td><td>Regional</td><td>2024-07-23</td></tr></table></div></div> <div>Created three Api's for the 3 datasets</div>	Name	Description	ID	Protocol	API endpoint type	Created	monthly_singstat_data		xik9d3xxa2	HTTP	Regional	2024-07-23	quarterly_singstat_data		d0zmb6i05	HTTP	Regional	2024-07-23	yearly_singstat_data		5w25jxhpb1	HTTP	Regional	2024-07-23
Name	Description	ID	Protocol	API endpoint type	Created																				
monthly_singstat_data		xik9d3xxa2	HTTP	Regional	2024-07-23																				
quarterly_singstat_data		d0zmb6i05	HTTP	Regional	2024-07-23																				
yearly_singstat_data		5w25jxhpb1	HTTP	Regional	2024-07-23																				
Lambda	<div><div><div>yearlysingstatdata</div><div>Format JSON</div></div><div><div>Event JSON</div><div><pre>1 { 2 "json_url": "https://tablebuilder.singstat.gov.sg/api/table/tabledata/M651351?isTestApi=true", 3 "s3_key": "singstat_data/UntransformedSingstatData/Yearly/Yearly_transport.json" 4 }</pre></div></div></div>																								

Code source info

Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment Var Execution results lambda_function

Environment

venvstat /

bin

certifi-2024.7.4.dist-info

charset-normalizer

charset-normalizer-3.3.2

idna

idna-3.7.dist-info

requests

requests-2.32.3.dist-info

urllib3

urllib3-2.2.2.dist-info

lambda_function.py

```
1 import json
2 import boto3
3 import requests
4
5 def lambda_handler(event, context):
6     # Extract parameters from the event
7     json_url = event.get('json_url')
8     s3_key = event.get('s3_key')
9     test_large_dataset = event.get('test_large_dataset', False)
10
11     if not json_url or not s3_key:
12         print('Missing json_url or s3_key in the event payload')
13         return {
14             'statusCode': 400,
15             'body': json.dumps('Missing json_url or s3_key in the event payload')
16         }
17
18     headers = {
19         'User-Agent': 'Mozilla/5.0',
20         # Add other headers if necessary
21     }
22
23     try:
24         # Fetch the JSON file
25         print(f'Fetching data from {json_url}')
26         response = requests.get(json_url, headers=headers)
27         response.raise_for_status()
28         data = response.json()
29         print('Data fetched successfully')
30
31         # Convert dictionary to list for size check
32         data_items = list(data.items())
33
34         # Initialize the Lambda client
35         lambda_client = boto3.client('lambda')
36
37         # S3 bucket name
38
39         # S3 bucket name
40         s3_bucket = 'singstat'
41
42         # Check if the dataset size is very big or if testing large dataset
43         if len(data_items) > 10000 or test_large_dataset:
44             print(f'Large dataset detected: {len(data_items)} items. Launching EC2 instances for processing.')
45             # Invoke Lambda function to launch EC2 instances for very large datasets
46             lambda_client.invoke(
47                 FunctionName='ProcessingLargeDatasetSingstat', # Name of the Lambda function to launch EC2 instances
48                 InvocationType='Event',
49                 Payload=json.dumps({'action': 'launch_ec2', 'data_chunks': data_items, 's3_bucket': s3_bucket, 's3_key': s3_key})
50             )
51             return {
52                 'statusCode': 200,
53                 'body': json.dumps('Large dataset detected, launching EC2 instances for processing')
54             }
55         else:
56             print(f'Dataset size is within limits: {len(data_items)} items. Uploading to S3.')
57             # Normal processing: upload the entire dataset to S3
58             s3_client = boto3.client('s3')
59
60             # Upload to S3
61             s3_client.put_object(
62                 Bucket=s3_bucket,
63                 Key=s3_key,
64                 Body=json.dumps(data),
65                 ContentType='application/json'
66             )
67             print(f'Data uploaded to S3 bucket {s3_bucket} with key {s3_key}')
68
69             # Invoke the next Lambda function for Glue Crawler and DataBrew
70             lambda_client.invoke(
71                 FunctionName='AutomateGlueCrawlerAndDataBrew', # Replace with your next Lambda function name
72                 InvocationType='Event', # Asynchronous invocation
73                 Payload=json.dumps({'s3_key': s3_key}) # Pass any required parameters
74             )
75             print('AutomateGlueCrawlerAndDataBrew Lambda function invoked successfully')
76
77             return {
78                 'statusCode': 200,
79                 'body': json.dumps('Data fetched and uploaded successfully, next Lambda function invoked')
80             }
81
82     except requests.exceptions.RequestException as e:
83         print(f'Error fetching data: {e}')
84         return {
85             'statusCode': 500,
86             'body': json.dumps(f'Error fetching data: {e}')
87         }
88     except boto3.exceptions.BotoError as e:
89         print(f'Error with AWS services: {e}')
90         return {
91             'statusCode': 500,
92             'body': json.dumps(f'Error with AWS services: {e}')
93         }
94     except Exception as e:
95         print(f'Error processing data: {e}')
96         return {
97             'statusCode': 500,
98             'body': json.dumps(f'Error processing data: {e}')
99         }
100 }
```

10:5 Python Spaces: 4

Created a Lambda function named Singstat which uploads JSON data to an S3 bucket after processing it from a given URL. It begins by obtaining the parameters (s3_key and json_url) from the test event. This is where the user will put the link to the json data from singstat and the s3 destination. If either is absent, an error is returned.

To check the amount of the dataset, it then uses the requests library to retrieve the JSON data and converts it to a list of objects. It calls another Lambda function (ProcessingLargeDatasetSingstat) to process the data using EC2 instances if the dataset is large.

For datasets of a reasonable size, it uploads the information to the S3 bucket (singstat) and into a folder called untransformedsingstatdata, in either the monthly, yearly or quarterly folder. The user specifies which folder in the test event. This is done because most singstat data are either monthly, annually, or quarterly.

It then uses the AutomateGlueCrawlerAndDataBrew Lambda function for more processing.

ProcessingLargeDatasetSingstat function

Code source Info Upload from ▾

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P) lambda_function Environment Var Execution results

ProcessingLargeDat lambda_function.py

```
1 import json
2 import boto3
3 from botocore.exceptions import ClientError
4
5 def lambda_handler(event, context):
6     ec2_client = boto3.client('ec2')
7     s3_client = boto3.client('s3')
8     lambda_client = boto3.client('lambda')
9     s3_bucket = 'singstat'
10
11     # Extract the s3_key from the event
12     s3_key = event.get('s3_key')
13
14     # Parameters to define the EC2 instance
15     instance_params = {
16         'ImageId': 'ami-0b72821e2f351e396', # Replace with your AMI ID
17         'InstanceType': 't2.micro', # Choose an appropriate instance type
18         'MaxCount': 1,
19         'MinCount': 1,
20         'KeyName': 'key-pair-for-large-datasets', # Replace with your key pair name
21         'SecurityGroupIds': ['sg-82e9f0a859a817065'], # Replace with your security group ID
22         'TagSpecifications': [
23             {
24                 'ResourceType': 'Instance',
25                 'Tags': [
26                     {'Key': 'Name', 'Value': 'DataProcessingInstance'}
27                 ]
28             }
29         ],
30         'UserData': f"""#!/bin/bash
31         echo '{json.dumps(event.get('data_chunks', []))}' > /tmp/data_chunks.json
32         # Include your script here to process data_chunks.json
33         """
34     }
35
36     try:
37         # Launch the EC2 instance
38         instance = ec2_client.run_instances(**instance_params)
39
40         # Extract the instance ID
41         instance_id = instance['Instances'][0]['InstanceId']
42         print(f"EC2 instance launched with ID: {instance_id}")
43
44         # Upload the data to S3 using the provided s3_key
45         s3_client.put_object(
46             Bucket=s3_bucket,
47             Key=s3_key,
48             Body=json.dumps(event.get('data_chunks', [])),
49             ContentType='application/json'
50         )
51         print(f'Data uploaded to S3 bucket {s3_bucket} with key {s3_key}')
52
53         # Invoke the next Lambda function for Glue Crawler and DataBrew
54         lambda_client.invoke(
55             FunctionName='AutomateGlueCrawlerAndDataBrew', # Replace with your next Lambda function name
56             InvocationType='Event', # Asynchronous invocation
57             Payload=json.dumps({'s3_key': s3_key}) # Pass any required parameters
58         )
59         print('AutomateGlueCrawlerAndDataBrew Lambda function invoked successfully')
60
61         # Return the instance ID and confirmation
62         return {
63             'statusCode': 200,
64             'body': json.dumps({'instance_id': instance_id, 'message': 'Data uploaded to S3 and next Lambda function invoked successfully'})
65         }
66
67     except ClientError as e:
68         error_message = e.response['Error']['Message']
69         print(f"ClientError launching EC2 instance: {error_message}")
70         return {
71             'statusCode': 500,
72             'body': json.dumps({'error': 'ClientError', 'message': error_message})
73         }
74
75 except Exception as e:
76     error_message = str(e)
77     print(f"Error launching EC2 instance: {error_message}")
78     return {
79         'statusCode': 500,
80         'body': json.dumps({'error': 'Exception', 'message': error_message})
81     }
```

1:1 Python Spaces: 4

EC2 > Instances > i-0811d2579aaec24bc

Instance summary for i-0811d2579aaec24bc (ProcessLargeDatasetsSingstat) [Info](#) Refresh Connect Instance state ▾

Updated less than a minute ago

Instance ID i-0811d2579aaec24bc (ProcessLargeDatasetsSingstat)	Public IPv4 address 35.174.156.194 open address	Private IPv4 addresses 172.31.89.88
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-35-174-156-194.compute-1.amazonaws.com open address
Hostname type IP name: ip-172-31-89-88.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-89-88.ec2.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for EC2 Learn more
Auto-assigned IP address 35.174.156.194 [Public IP]	VPC ID vpc-0be88d780f840a518	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-0f74d4ca5151b9e9d	
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:609225191640:instance/i-0811d2579aaec24bc	

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

▼ Instance details [Info](#)

Platform	AMI ID	Monitoring

This AWS Lambda function launches an EC2 instance to process large datasets, uploads the data to an S3 bucket and then calls another Lambda function to process the data even more.

The first step involves obtaining the `s3_key` from the test event in the `singstat` function and specifying the AMI ID, instance type, key pair, security group, and user data script to manage the data chunks for the EC2 instance.

After that, the function tries to start the EC2 instance and uploads the data chunks to the designated S3 bucket (`singstat`) and into the `untransformedsingstatdata` folder and into either `monthly`, `quarterly`, or `yearly` just like the `singstat` function when it succeeds. Following a successful upload of the data, it immediately calls the `AutomateGlueCrawlerAndDatabrew` Lambda function to handle the data further.

When problems occur, the method returns the relevant error messages and status codes and handles possible errors using `ClientError` from `boto3` and general exceptions.

AutomateGlueCrawlerAndDatabrew function

```
1 import json
2 import boto3
3 import time
4
5 def lambda_handler(event, context):
6     glue_client = boto3.client('glue')
7     databrew_client = boto3.client('databrew')
8     athena_client = boto3.client('athena')
9
10    # Extract parameters from the event
11    s3_key = event.get('s3_key')
12
13    if not s3_key:
14        print("Error: Missing s3_key in the event payload")
15        return {
16            'statusCode': 400,
17            'body': json.dumps('Missing s3_key in the event payload')
18        }
19
20    print(f"Received s3_key: {s3_key}")
21
22    # Convert s3_key to lowercase for case insensitive comparison
23    s3_key_lower = s3_key.lower()
24
25    # Names for the Glue Crawler and DataBrew Job
26    untransformed_crawler_name = 'crawl_untransformed_singstat_data' # Name of your Glue Crawler for untransformed data
27    transformed_crawler_name = 'crawl_transformed_singstat_data' # Name of your Glue Crawler for transformed data
28
29    # Determine the job name and Athena query based on the prefix
30    if s3_key_lower.startswith('singstat_data/untransformedsingstatdata/monthly/'):
31        databrew_job_name = 'MonthlyTransformSingstatData'
32        athena_query = "SELECT * FROM monthly"
33    elif s3_key_lower.startswith('singstat_data/untransformedsingstatdata/quarterly/'):
34        databrew_job_name = 'QuarterlyTransformSingstatData'
35        athena_query = "SELECT * FROM quarterly"
36    elif s3_key_lower.startswith('singstat_data/untransformedsingstatdata/yearly/'):
37        databrew_job_name = 'YearlyTransformSingstatData'
```

1:1 Python Spaces: 4

```
    databrew_job_name = 'YearlyTransformSingstatData'
    athena_query = "SELECT * FROM yearly"
else:
    print("Error: File does not match any known prefixes")
    return {
        'statusCode': 400,
        'body': json.dumps('File does not match any known prefixes')
    }

try:
    # Start the Glue Crawler for untransformed data
    glue_client.start_crawler(Name=untransformed_crawler_name)
    print('Glue Crawler for untransformed data started successfully')

    # Wait for the untransformed data crawler to finish
    while True:
        response = glue_client.get_crawler(Name=untransformed_crawler_name)
        state = response['Crawler']['State']
        if state == 'READY':
            print('Glue Crawler for untransformed data has finished')
            break
        print('Waiting for the untransformed data crawler to finish...')
        time.sleep(30)

    # Start the DataBrew Job
    databrew_client.start_job_run(Name=databrew_job_name)
    print('DataBrew job started successfully')

    # Wait for the DataBrew job to finish
    while True:
        job_run = databrew_client.list_job_runs(Name=databrew_job_name, MaxResults=1)
        state = job_run['JobRuns'][0]['State']
        if state == 'SUCCEEDED':
            print('DataBrew job has finished successfully')
            break
        elif state in ['FAILED', 'STOPPED']:
            print(f'DataBrew job did not complete successfully. State: {state}')
            return f
```

1:1 Python Spaces: 4

```
lambda_function * Environment Var * Execution results *
74     return {
75         'statusCode': 500,
76         'body': json.dumps(f"DataBrew job did not complete successfully. State: {state}")
77     }
78     print('Waiting for the DataBrew job to finish...')
79     time.sleep(30)
80
81     # Start the Glue Crawler for transformed data
82     glue_client.start_crawler(Name=transformed_crawler_name)
83     print('Glue Crawler for transformed data started successfully')
84
85     # Wait for the transformed data crawler to finish
86     while True:
87         response = glue_client.get_crawler(Name=transformed_crawler_name)
88         state = response['Crawler']['State']
89         if state == 'READY':
90             print('Glue Crawler for transformed data has finished')
91             break
92         print('Waiting for the transformed data crawler to finish...')
93         time.sleep(30)
94
95     # Run the Athena query
96     print(f"Running Athena query: {athena_query}")
97
98     # Start the Athena query execution
99     response = athena_client.start_query_execution(
100         QueryString=athena_query,
101         QueryExecutionContext={
102             'Database': 'transformed_singstat_data'
103         },
104         ResultConfiguration={
105             'OutputLocation': 's3://singstat/singstat_data/QueriedSingstatData/'
106         }
107     )
108     query_execution_id = response['QueryExecutionId']
109     print(f"Started Athena query with execution ID: {query_execution_id}")
110
111
112     # Check the Athena query execution status
113     state = 'RUNNING'
114     while state in ['RUNNING', 'QUEUED']:
115         response = athena_client.get_query_execution(QueryExecutionId=query_execution_id)
116         state = response['QueryExecution']['Status']['State']
117         if state == 'SUCCEEDED':
118             print("Athena query succeeded!")
119             break
120         elif state in ['FAILED', 'CANCELLED']:
121             print(f"Athena query did not complete successfully. State: {state}")
122             return {
123                 'statusCode': 500,
124                 'body': json.dumps(f"Athena query did not complete successfully. State: {state}")
125             }
126     print('Waiting for the Athena query to finish...')
127     time.sleep(10)
128
129     return {
130         'statusCode': 200,
131         'body': json.dumps('Glue Crawler, DataBrew job, and Athena query started and finished successfully')
132     }
133
134 except Exception as e:
135     print(f"Error: {e}")
136     return {
137         'statusCode': 500,
138         'body': json.dumps(f"Error: {e}")
139     }
```

This AWS Lambda function consists of AWS Glue, DataBrew, and Athena operations to process and query data stored in the untransformedsingstatdata folder.

Amazon S3 > Buckets > singstat > singstat_data > UntransformedSingstatData/

UntransformedSingstatData/ [Copy S3 URI](#)

Database properties

Name	Description	Location	Created on (UTC)
untransformed_singstat_data	-	-	July 22, 2024 at 10:25:22

Tables (3)

Name	Database	Location	Classification	Deprecated	View data	Data quality
monthly	untransformed_singstat	s3://singstat/singstat_d	.JSON	-	Table data	View data quality
quarterly	untransformed_singstat	s3://singstat/singstat_d	.JSON	-	Table data	View data quality
yearly	untransformed_singstat	s3://singstat/singstat_d	.JSON	-	Table data	View data quality

First, it crawls the untransformedsingstatdata folder using the crawl_untransformed_singstat_data crawler and uploads it to the untransformed_singstat_data database.

Recipe jobs Profile jobs Schedules

Recipe jobs (3) info

Find jobs Show all

Job name	Status	Job input	Job output	Last run	Created on	Created by	Tags
YearlyTransformSingstatData	Succeeded	YearlyTranfor... Project (YearlySingsta... Dataset + YearlyTranfor... Recipe)	1 output	a few seconds ago July 28, 2024, 6:32:41 pm	5 days ago July 23, 2024, 7:27:00 pm	voclabs	-
QuarterlyTransformSingstatData	Succeeded	QuarterlyTra... Project (QuarterlySin... Dataset + QuarterlyTra... Recipe)	1 output	5 hours ago July 28, 2024, 1:16:14 pm	5 days ago July 23, 2024, 7:23:55 pm	voclabs	-
MonthlyTransformSingstatData	Succeeded	MonthlyTrans... Project (MonthlySing... Dataset + MonthlyTrans... Recipe)	1 output	5 hours ago July 28, 2024, 1:26:27 pm	5 days ago July 23, 2024, 6:42:05 pm	voclabs	-

MonthlyTransformSingstat

Dataset: MonthlySingstatData Sample: First n sample (500 rows)

Last job run 5 hours ago, no job runs scheduled Run job

UNDO FILTER SORT COLUMN FORMAT CLEAN EXTRACT MISSING UNWIND DUPLICATES OUTLIERS SPLIT MERGE CREATE FUNCTIONS CONDITIONS NEST-UNNEST PIVOT GROUP JOIN UNION TEXT SCALE MAPPING ENCODE MORE

Viewing 3 columns 500 rows

ABC Data Series	ABC Month	# Value
Electricity Generation	1975 Jan	341.8
Electricity Generation	1975 Feb	289.6
Electricity Generation	1975 Mar	342.8
Electricity Generation	1975 Apr	348.6
Electricity Generation	1975 May	351.4
Electricity Generation	1975 Jun	343.3
Electricity Generation	1975 Jul	359.7
Electricity Generation	1975 Aug	363
Electricity Generation	1975 Sep	359.1
Electricity Generation	1975 Oct	368.5
Electricity Generation	1975 Nov	345.6
Electricity Generation	1975 Dec	362.3

Recipe (11)

- Unnest values for Data_row_unnested_columns_unnested, replacing Data_row_unnested_columns_u
- Delete column Data.between, Data.dataLastUpda
- Delete column Data.frequency, Data.generatedBy, Data.id, Da
- ata.limit, Data.offset, Data_row_unnested.footerNo, D
- ata_row_unnested.seriesNo, Data.search, Data.sortB
- y, Data.timeFilter, Data.title, Data.Count, Message, St
- atusCode, Data_row_unnested.uoM
- Rename Data_row_unnested_columns_unnested.v
- alue to Value
- Rename Data_row_unnested_columns_unnested.k
- ey to Month
- Change type of Value to Double
- Move Data_row_unnested.rowText to Start of ta
- ble
- Rename Data_row_unnested.rowText to Data Se
- ries

It then takes the s3_key out of the test event in singstat function and uses its prefix to identify which DataBrew task and Athena query to use. For example, singstat_data/untransformedsingstatdata/monthly/ for the MonthlyTransformSingstatData databrew job.

Amazon S3 Buckets s3bucket singstat_data/ TransformedSingstatData/

AWS Glue Database transformed_singstat_data

transformed_singstat_data

Database properties

Name	Description	Location	Created on (UTC)
transformed_singstat_data	-	-	July 24, 2024 at 09:25:23

Tables (3)

Name	Database	Location	Classification	Deprecated	View data	Data quality
monthly	transformed_singstat_d	s3://singstat/singstat_d	JSON	-	Table data	View data quality
quarterly	transformed_singstat_d	s3://singstat/singstat_d	JSON	-	Table data	View data quality
yearly	transformed_singstat_d	s3://singstat/singstat_d	JSON	-	Table data	View data quality

Once the DataBrew job completes successfully, the transformed data is then uploaded to the transformedsingstatdata folder and into either the monthly, quarterly, or yearly folder. Then, the crawl_transformed_singstat_data crawler is used to crawl the transformedsingstat data folder and uploads it to the transformed_singstat_data database.

Amazon S3

>

Buckets

>

singstat

>

singstat_data/

>

QueriedSingstatData/

QueriedSingstatData/

Copy S3 URI

Objects

Properties

Objects (13)

Info

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

Show versions

1

<

>

Refresh

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	01f657ac-adce-4f06-bbc4-fa055df9f46f.csv	csv	July 28, 2024, 11:49:18 (UTC+08:00)	9.2 KB	Standard
<input type="checkbox"/>	01f657ac-adce-4f06-bbc4-fa055df9f46f.csv.metadata	metadata	July 28, 2024, 11:49:18 (UTC+08:00)	163.0 B	Standard
<input type="checkbox"/>	0efa2667-63c7-497b-94a9-62abd9559f6e.csv	csv	July 28, 2024, 13:29:17 (UTC+08:00)	25.9 KB	Standard
<input type="checkbox"/>	0efa2667-63c7-497b-94a9-62abd9559f6e.csv.metadata	metadata	July 28, 2024, 13:29:17 (UTC+08:00)	165.0 B	Standard
<input type="checkbox"/>	12b7d8da-6d96-4e72-a787-c0cf56361054.csv	csv	July 28, 2024, 18:36:00 (UTC+08:00)	9.2 KB	Standard

To analyse the data that has been processed, a select Athena query is finally run. The queries are different for each type of dataset. For example, `SELECT * FROM monthly` for month and `SELECT * FROM quarterly` for quarter. All the Athena queries go to the QueriedSingstatData folder.

In addition to providing suitable error handling and status updates throughout, the function incorporates checks at each stage to guarantee that the tasks are completed successfully.

SingstatNotifications function

```

import boto3
import json
import datetime
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    # Set up AWS clients
    s3_client = boto3.client('s3')
    ce_client = boto3.client('ce')
    sns_client = boto3.client('sns')

    # Define the S3 bucket and file keys
    s3_bucket = 'singstat' # Replace with your S3 bucket name
    powerbi_chart_key = 'singstat_data/PowerBIChartCostExplorerReport/YearlyTransformSingstatData.pbix' # Replace with the path to your Power BI file in S3

    # Generate the AWS Cost Explorer report
    try:
        end = datetime.date.today()
        start = end - datetime.timedelta(days=5)
        response = ce_client.get_cost_and_usage(
            TimePeriod={
                'Start': start.strftime('%Y-%m-%d'),
                'End': end.strftime('%Y-%m-%d')
            },
            Granularity='DAILY',
            Metrics=['UnblendedCost'],
            Filter={
                'Not': {
                    'Dimensions': {
                        'Key': 'RECORD_TYPE',
                        'Values': ['Credit']
                    }
                }
            }
        )
    
```

12:35 Python Spaces: 4

```

    )
except Exception as e:
    logger.error(f"Error fetching cost and usage data: {e}")
    return {
        'statusCode': 500,
        'body': json.dumps('Error fetching cost and usage data')
    }

# Extract the total cost
try:
    total_cost = response['ResultsByTime'][0]['Total']['UnblendedCost']['Amount']
except KeyError as e:
    logger.error(f"Key error while extracting total cost: {e}")
    return {
        'statusCode': 500,
        'body': json.dumps('Error extracting total cost')
    }

# Create the cost report content
cost_report_content = f"Total cost for {start} to {end} (excluding credits): ${total_cost}"

# Generate a unique key for the cost report based on the date
cost_report_key = f"singstat_data/PowerBIChartCostExplorerReport/cost_report_{end.strftime('%Y-%m-%d')}.csv"

# Upload the cost report to S3
try:
    s3_client.put_object(Bucket=s3_bucket, Key=cost_report_key, Body=cost_report_content)
except Exception as e:
    logger.error(f"Error uploading cost report to S3: {e}")
    return {
        'statusCode': 500,
        'body': json.dumps('Error uploading cost report to S3')
    }

# Generate a pre-signed URL for the cost report (expires in 7 days)
try:
    cost_report_url = s3_client.generate_presigned_url(

```

12:35 Python Spaces: 4

```

    cost_report_url = s3_client.generate_presigned_url(
        'get_object',
        Params={'Bucket': s3_bucket, 'Key': cost_report_key},
        ExpiresIn=604800 # Link expires in 7 days
    )
except Exception as e:
    logger.error(f"Error generating pre-signed URL: {e}")
    return {
        'statusCode': 500,
        'body': json.dumps('Error generating pre-signed URL')
    }

# Generate a pre-signed URL for the Power BI chart (expires in 7 days)
try:
    powerbi_chart_url = s3_client.generate_presigned_url(
        'get_object',
        Params={'Bucket': s3_bucket, 'Key': powerbi_chart_key},
        ExpiresIn=604800 # Link expires in 7 days
    )
except Exception as e:
    logger.error(f"Error generating pre-signed URL for Power BI chart: {e}")
    return {
        'statusCode': 500,
        'body': json.dumps('Error generating pre-signed URL for Power BI chart')
    }

# Construct the SNS message
message = f"Total cost for {start} to {end} (excluding credits): ${total_cost}\n\n"
message += f"Please check the updated Power BI charts and AWS Cost Explorer reports:\n"
message += f"Power BI Chart: {powerbi_chart_url}\n"
message += f"Cost Explorer Report: {cost_report_url}\n"

# Publish the message to SNS
try:
    sns_client.publish(
        TopicArn="arn:aws:sns:us-east-1:609225191640:SingstatNotifications",
        Message=message,
        Subject="Daily AWS Cost Report and Power BI Charts"
    )
except Exception as e:
    logger.error(f"Error publishing message to SNS: {e}")
    return {
        'statusCode': 500,
        'body': json.dumps('Error publishing message to SNS')
    }

return {
    'statusCode': 200,
    'body': json.dumps('Notification sent successfully')
}

```

12:35 Python Spaces: 4

```

# Publish the message to SNS
try:
    sns_client.publish(
        TopicArn='arn:aws:sns:us-east-1:609225191640:SingstatNotifications',
        Message=message,
        Subject='Daily AWS Cost Report and Power BI Charts'
    )
except Exception as e:
    logger.error(f"Error publishing message to SNS: {e}")
    return {
        'statusCode': 500,
        'body': json.dumps('Error publishing message to SNS')
    }

return {
    'statusCode': 200,
    'body': json.dumps('Notification sent successfully')
}

```

Using AWS Cost Explorer, this Lambda function creates a daily cost report and uploads it to the s3 folder named PowerBIChartCostExplorerReport. It then pre-signs URLs for the report and a Power BI chart and sends an SNS notification.

PowerBIChartCostExplorerReport/



Objects Properties

Objects (2) Info

  Copy S3 URI  Copy URL  Download  Open  Delete  Actions  Create folder  Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to set permissions. [Learn more](#)

☐ Show versions

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 cost_report_2024-07-26.csv	csv	July 26, 2024, 18:55:22 (UTC+08:00)	74.0 B	Standard
<input type="checkbox"/>	 YearlyTransformSingstatData.pbix	pbix	July 26, 2024, 17:49:22 (UTC+08:00)	51.7 KB	Standard

It retrieves the last five days' worth of expenses, calculates the overall cost, and uploads the report to S3. Both the report and the Power BI chart have pre-signed URLs created by the function that make them available for seven days.

SingstatNotifications

Edit

Delete

Details

Name	Display name
SingstatNotifications	-
ARN	Topic owner
arn:aws:sns:us-east-1:609225191640:SingstatNotifications	609225191640
Type	
Standard	

[Subscriptions](#) [Access policy](#) [Data protection policy](#) [Delivery policy \(HTTP/S\)](#) [Delivery status logging](#) [Encryption](#) [Tags](#)


Subscriptions (1)

Edit

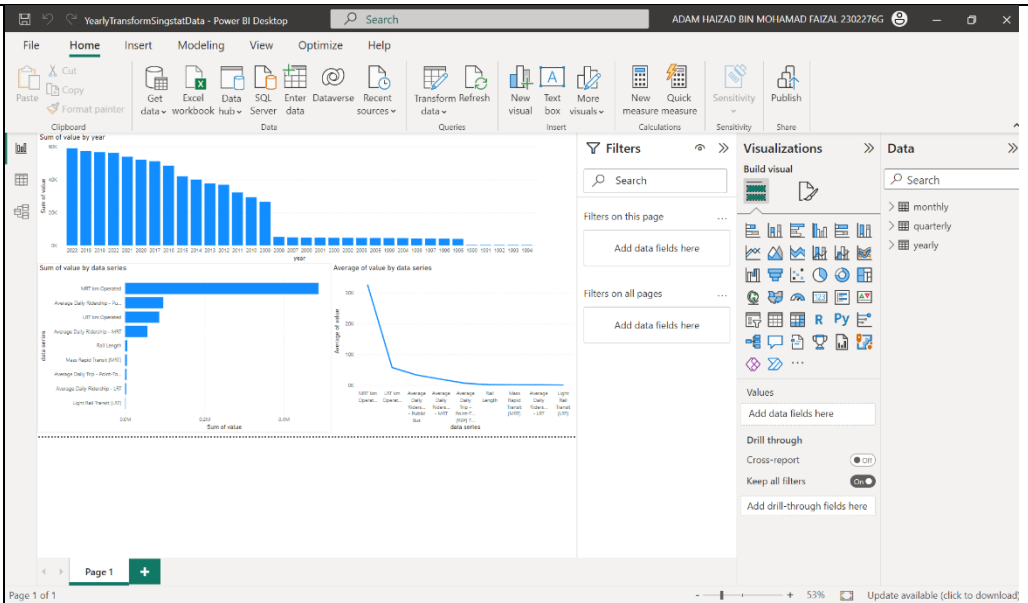
Delete

Request confirmation

Confirm subscription

<input type="radio"/>	ID	Endpoint	Status	Protocol
<input type="radio"/>	d71c8e00-5a0a-47cf-a445-dd7c8c...	2302276G@student.tp.edu.sg	 Confirmed	EMAIL

QuickSight (Power BI)



After querying in Athena, the queried data is then displayed in Power BI. The connection from Athena to Power BI is done through ODBC driver. After creating visualizations, the Power BI file is saved and uploaded to the PowerBIChartCostExplorerReport s3 folder.

CloudWatch

CloudWatch Alarms (12)						
Name	State	Last state update (Local)	Conditions	Actions		
AutomateGlueCrawlerAndDat abrewLambdaError	Insufficient data	2024-07-28 18:41:58	Errors > 0 for 1 datapoints within 5 minutes	Actions enabled		
AutomateGlueCrawlerAndDat abrewLambdaDuration	Insufficient data	2024-07-28 18:41:55	Duration > 900000 for 1 datapoints within 5 minutes	Actions enabled		
AutomateGlueCrawlerAndDat abrewLambdaThrottle	Insufficient data	2024-07-28 18:41:20	Throttles > 0 for 1 datapoints within 5 minutes	Actions enabled		
SingstatLambdaThrottle	Insufficient data	2024-07-28 18:41:19	Throttles > 0 for 1 datapoints within 5 minutes	Actions enabled		
SingstatLambdaDuration	Insufficient data	2024-07-28 18:41:12	Duration > 150000 for 1 datapoints within 5 minutes	Actions enabled		
SingstatLambdaError	Insufficient data	2024-07-28 18:41:11	Errors > 0 for 1 datapoints within 5 minutes	Actions enabled		
ProcessingLargeDatasetSings tatiLambdaThrottle	Insufficient data	2024-07-28 14:11:41	Throttles > 0 for 1 datapoints within 5 minutes	Actions enabled		
ProcessingLargeDatasetSings tatiLambdaError	Insufficient data	2024-07-28 14:11:13	Errors > 0 for 1 datapoints within 5 minutes	Actions enabled		
ProcessingLargeDatasetSings tatiLambdaDuration	Insufficient data	2024-07-28 14:11:10	Duration > 100000 for 1 datapoints within 5 minutes	Actions enabled		
SingstatNotificationsLambda Duration	Insufficient data	2024-07-28 13:19:07	Duration > 20000 for 1 datapoints within 5 minutes	Actions enabled		
SingstatNotificationsLambda Throttle	Insufficient data	2024-07-27 20:57:16	Throttles > 0 for 1 datapoints within 5 minutes	Actions enabled		
SingstatNotificationsLambda Error	Insufficient data	2024-07-27 20:56:18	Errors > 0 for 1 datapoints within 5 minutes	Actions enabled		

Created alarms for all the 4 lambda functions. The alarms include errors, throttles, and duration to guarantee the dependability and efficiency of our Lambda services.

When a Lambda function experiences execution errors, including exceptions or timeouts, the error alarms are set off, signalling possible problems with the code or environment that need to be fixed.

Throttle alarms help us detect and handle capacity restrictions by alerting us when a function above its concurrency limits and rejects or delays further invocations.

Duration alarms monitor how long Lambda functions execute and notify us if they take longer than anticipated. This could be a sign of wasteful code or

performance bottlenecks. When combined, these alerts offer thorough oversight to preserve the effectiveness and well-being of our Lambda operations.

Step 2: Configure actions

Actions

Notification
When In alarm, send a notification to "SingstatNotifications"

ALARM: "SingstatLambdaError" in US East (N. Virginia)

AN

AWS Notifications<no-reply@sns.amazonaws.com>

To: ADAM HAIZAD BIN MOHAMAD FAIZAL

☺

↩ Reply

↩ Reply all

➡ Forward

📧

⋮

Sun 28/07/2024 21:02

Caution: This is an Internet email. If you are unsure of the content, please check the source before you respond.

You are receiving this email because your Amazon CloudWatch Alarm "SingstatLambdaError" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [1.0 (28/07/24 12:57:00)] was greater than the threshold (0.0) (minimum 1 datapoint for OK -> ALARM transition)," at "Sunday 28 July, 2024 13:02:11 UTC".

View this alarm in the AWS Management Console:
<https://us-east-1.console.aws.amazon.com/cloudwatch/deeplink/s?region=us-east-1#alarmsV2:alarm/SingstatLambdaError>

Alarm Details:

- Name: SingstatLambdaError
- Description:
- State Change: INSUFFICIENT_DATA -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [1.0 (28/07/24 12:57:00)] was greater than the threshold (0.0) (minimum 1 datapoint for OK -> ALARM transition).
- > ALARM transition).
- Timestamp: Sunday 28 July, 2024 13:02:11 UTC
- AWS Account: 609225191640
- Alarm Arn: arn:aws:cloudwatch:us-east-1:609225191640:alarm:SingstatLambdaError

Threshold:

- The alarm is in the ALARM state when the metric is GreaterThanThreshold 0.0 for at least 1 of the last 1 period(s) of 300 seconds.

Monitored Metric:

- MetricNamespace: AWS/Lambda
- MetricName: Errors
- Dimensions: [FunctionName = singstat]
- Period: 300 seconds
- Statistic: Sum
- Unit: not specified
- TreatMissingData: missing

State Change Actions:

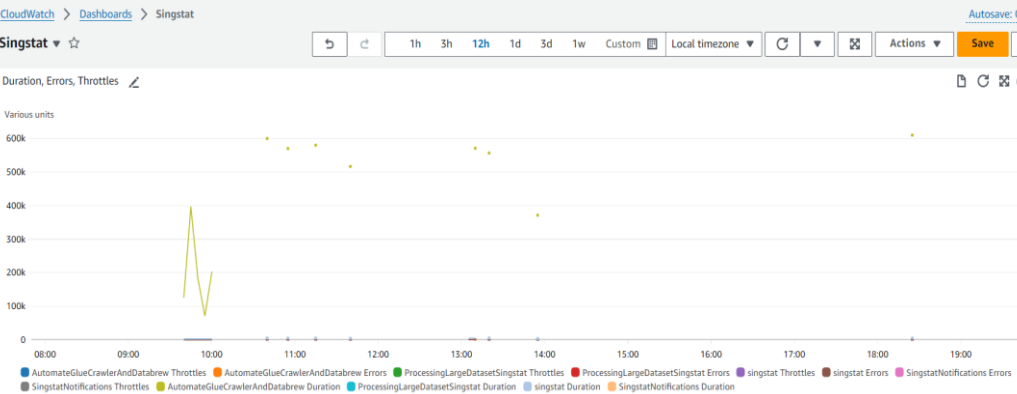
- OK:
- ALARM: [arn:aws:sns:us-east-1:609225191640:SingstatNotifications]
- INSUFFICIENT_DATA:

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:609225191640:SingstatNotifications:d71c8e00-5a0a-47cf-a445-da7c8cbe1f08&Endpoint=2302276G@student.tpo.edu.sg>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

For all these alarms, when the alarm state goes into the ‘in alarm’ state, an email will be sent through SNS to notify about the alarm.



I made a dashboard to track all the data with alarms in addition to the alerts. We can swiftly detect and fix problems with this dashboard's real-time view of important metrics for each Lambda function, such as durations, throttles, and errors.

Amazon EventBridge > Rules > ReingestHistoricalData

ReingestHistoricalData

Edit Disable Delete CloudFormat

Rule details info

Rule name	Status	Event bus name	Type
ReingestHistoricalData	Enabled	default	Scheduled Standard
Description	Rule ARN	Event bus ARN	
	arn:aws:events:us-east-1:609225191640:rule/ReingestHistoricalData	arn:aws:events:us-east-1:609225191640:event-bus/default	

Event schedule Targets Monitoring Tags

Event schedule info

Cron expression

0 0 1 1 ? *

Next 10 trigger date(s)

UTC

Wed, 01 Jan 2025 00:00:00 UTC
Thu, 01 Jan 2026 00:00:00 UTC
Fri, 01 Jan 2027 00:00:00 UTC
Sat, 01 Jan 2028 00:00:00 UTC
Mon, 01 Jan 2029 00:00:00 UTC
Tue, 01 Jan 2030 00:00:00 UTC
Wed, 01 Jan 2031 00:00:00 UTC
Thu, 01 Jan 2032 00:00:00 UTC
Sat, 01 Jan 2033 00:00:00 UTC

Event schedule Targets Monitoring Tags

Targets

Details	Target Name	Type	Arn	Input	Role
▼	singstat	Lambda function	arn:aws:lambda:us-east-1:609225191640:function:singstat	Matched event	-
Input to target: Matched event					
Additional parameters: --					
Dead-letter queue (DLQ): -					

	<p>Created a CloudWatch rule which will run the Singstat lambda function every year. The reason this is crucial is that Singstat refreshes its data on an annual basis, thus it ensures that the data is the most recent. This automated procedure reduces the need for human interaction while maintaining data relevance and accuracy.</p>																																																		
CloudTrail	<div><div><div><div><div>CloudTrail > Trails > arn:aws:cloudtrail:us-east-1:609225191640:trail/Singstat-data</div><div>Singstat-data</div><div><div>Delete</div><div>Stop logging</div></div></div><div><div>General details</div><div>Edit</div><div><table><tr><td>Trail logging</td><td>Trail log location</td><td>Log file validation</td><td>SNS notification delivery</td></tr><tr><td>Logging</td><td>aws-cloudtrail-logs-609225191640-576709e6/AWSLogs/609225191640</td><td>Enabled</td><td>Disabled</td></tr><tr><td>Trail name</td><td></td><td>Last file validation delivered</td><td>Last SNS notification</td></tr><tr><td>Singstat-data</td><td></td><td>July 27, 2024, 14:36:55 (UTC+08:00)</td><td>-</td></tr><tr><td>Multi-region trail</td><td>Last log file delivered</td><td></td><td></td></tr><tr><td>Yes</td><td>July 27, 2024, 13:39:25 (UTC+08:00)</td><td></td><td></td></tr><tr><td>Apply trail to my organization</td><td>Log file SSE-KMS encryption</td><td></td><td></td></tr><tr><td>Not enabled</td><td>Not enabled</td><td></td><td></td></tr></table></div></div></div><div><div>Amazon S3 > Buckets > aws-cloudtrail-logs-609225191640-576709e6 > AWSLogs/ > 609225191640/ > CloudTrail/ > us-east-1/ > 2024/ > 07/28/</div><div><div>Objects</div><div>Properties</div></div><div><div>Objects (2) Info</div><div><div>Refresh</div><div>Copy S3 URI</div><div>Copy URL</div><div>Download</div><div>Open</div><div>Delete</div><div>Actions</div><div>Create folder</div><div>Upload</div></div><div>Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to exp permissions. Learn more</div><div><div>Find objects by prefix</div><div>Show versions</div></div><table><tr><th><input type="checkbox"/></th><th>Name</th><th>Type</th><th>Last modified</th><th>Size</th><th>Storage</th></tr><tr><td><input type="checkbox"/></td><td>609225191640_CloudTrail_u s-east- 1_20240728T1205Z_zYT2Re TIU6WwOOIU.json.gz</td><td>gz</td><td>July 28, 2024, 20:01:08 (UTC+08:00)</td><td>1.4 KB</td><td>Standard</td></tr><tr><td><input type="checkbox"/></td><td>609225191640_CloudTrail_u s-east- 1_20240728T1210Z_jyDKw WZ9LcmFF1n4.json.gz</td><td>gz</td><td>July 28, 2024, 20:05:59 (UTC+08:00)</td><td>3.0 KB</td><td>Standard</td></tr></table></div></div><div><p>To monitor and track user activity across different AWS regions, the "Singstat-data" AWS CloudTrail trail is actively logging events. This trail uses log file validation to guarantee the integrity of the logs and saves log files in a designated S3 bucket named aws-cloudtrail-logs-609225191640-576709e6.</p><p>All activities inside the AWS environment are logged and can be examined for operational and security troubleshooting thanks to this configuration, which is utilised for auditing and compliance purposes.</p><p>Additionally, Insights are enabled to get a deeper analysis of the API activity to detect unusual activities.</p></div></div></div>	Trail logging	Trail log location	Log file validation	SNS notification delivery	Logging	aws-cloudtrail-logs-609225191640-576709e6/AWSLogs/609225191640	Enabled	Disabled	Trail name		Last file validation delivered	Last SNS notification	Singstat-data		July 27, 2024, 14:36:55 (UTC+08:00)	-	Multi-region trail	Last log file delivered			Yes	July 27, 2024, 13:39:25 (UTC+08:00)			Apply trail to my organization	Log file SSE-KMS encryption			Not enabled	Not enabled			<input type="checkbox"/>	Name	Type	Last modified	Size	Storage	<input type="checkbox"/>	609225191640_CloudTrail_u s-east- 1_20240728T1205Z_zYT2Re TIU6WwOOIU.json.gz	gz	July 28, 2024, 20:01:08 (UTC+08:00)	1.4 KB	Standard	<input type="checkbox"/>	609225191640_CloudTrail_u s-east- 1_20240728T1210Z_jyDKw WZ9LcmFF1n4.json.gz	gz	July 28, 2024, 20:05:59 (UTC+08:00)	3.0 KB	Standard
Trail logging	Trail log location	Log file validation	SNS notification delivery																																																
Logging	aws-cloudtrail-logs-609225191640-576709e6/AWSLogs/609225191640	Enabled	Disabled																																																
Trail name		Last file validation delivered	Last SNS notification																																																
Singstat-data		July 27, 2024, 14:36:55 (UTC+08:00)	-																																																
Multi-region trail	Last log file delivered																																																		
Yes	July 27, 2024, 13:39:25 (UTC+08:00)																																																		
Apply trail to my organization	Log file SSE-KMS encryption																																																		
Not enabled	Not enabled																																																		
<input type="checkbox"/>	Name	Type	Last modified	Size	Storage																																														
<input type="checkbox"/>	609225191640_CloudTrail_u s-east- 1_20240728T1205Z_zYT2Re TIU6WwOOIU.json.gz	gz	July 28, 2024, 20:01:08 (UTC+08:00)	1.4 KB	Standard																																														
<input type="checkbox"/>	609225191640_CloudTrail_u s-east- 1_20240728T1210Z_jyDKw WZ9LcmFF1n4.json.gz	gz	July 28, 2024, 20:05:59 (UTC+08:00)	3.0 KB	Standard																																														

Key Management Service (KMS)

KMS > Customer-managed keys > Key ID: 7f124d15-38d5-479d-9d9d-de820a6756e0

7f124d15-38d5-479d-9d9d-de820a6756e0

Key actions

General configuration

Alias

Singstat-encryption-key

ARN

arn:aws:kms:us-east-1:609225191640:key/7f124d15-38d5-479d-9d9d-de820a6756e0

Status

Enabled

Description

-

Creation date

Jul 26, 2024 21:08 GMT+8

Regionality

Single region

Key policy

Cryptographic configuration

Tags

Key rotation

Aliases

Key policy

Switch

Key administrators (1)

Add

Choose the IAM users and roles who can administer this key through the KMS API. You might need to add additional permissions for the users or roles to administer this key from this console page.

Search Key administrators

☐

Name

Path

Type

☐

LabRole

/

Role

☒ Encrypt query results

Encryption type

☐ SSE_S3

Specifies server-side encryption (SSE) with S3-managed encryption keys.

☒ SSE_KMS

Specifies server-side encryption (SSE) with AWS KMS-managed keys.

☐ CSE_KMS

Specifies client-side encryption (CSE) with KMS-managed keys.

Choose an AWS KMS key

This key will be used to encrypt and decrypt your resources. [Learn more](#)

☐ Use an AWS owned key (aws/s3)

A key that AWS owns and manages for you.

☒ Choose a different AWS KMS key (advanced)

Choose a key that you have permission to use, or create a new one.

arn:aws:kms:us-east-1:609225191640:key/7f124d15-38d5-479d-9d9d-de820a6756e0

Create an AWS KMS key

AWS KMS key details

Key ARN

arn:aws:kms:us-east-1:609225191640:key/7f124d15-38d5-479d-9d9d-de820a6756e0

Key status

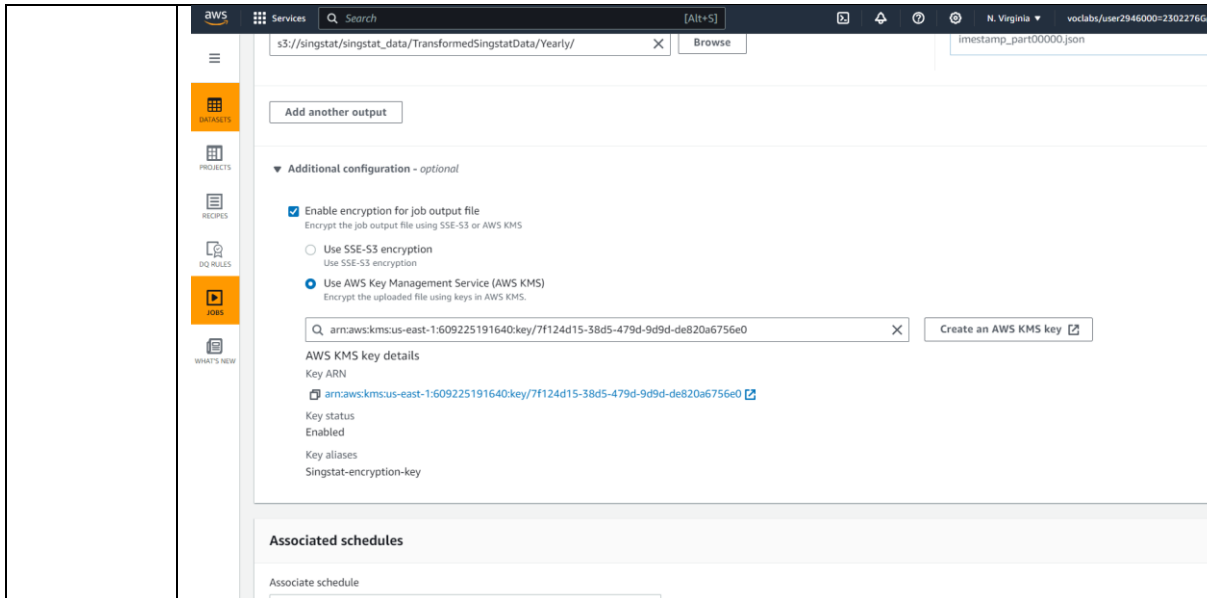
Enabled

Key action

Switch

Add

Enabled



I made an encryption key that can be used with S3, Athena, and DataBrew, among other services named Singstat-encryption-key. By encrypting data both in transit and at rest, this key improves data security by guaranteeing that private data is safeguarded during various processing and storage phases. We adhere to data protection laws and uphold uniform security standards by combining this encryption key with different services.

User's process:

First, the user will need to create an API gateway to access the singstat data. Then, go to the singstat lambda function and edit any of the test event and modify the s3_key, which is the destination you want the file to be in. Make sure that the destination matches with the dataset. For example, monthly dataset goes to monthly folder. Also modify the json_url which is the link to your singstat data, Run the function and the function automates up till Athena for you.

Then, the user needs to connect Athena to Power BI using ODBC driver. To do this, the user needs to get the access key id, secret access key id and session token, which can be found on the details for the learner lab. After connecting to Athena, the user can perform any visualizations they like to send to the client.

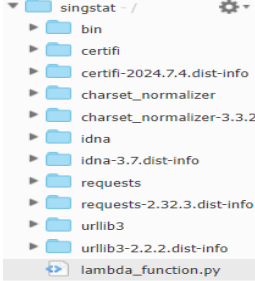
When the user is done, download the file and upload it to the PowerBIChartCostExplorerReport. Then go to the SingstatNotifications lambda function and make sure to replace the powerbi_chart key with the s3 link to the file. Then, run the function and the client will receive an email on the visualizations and costs.


SDL Learning Journal

Task 1:

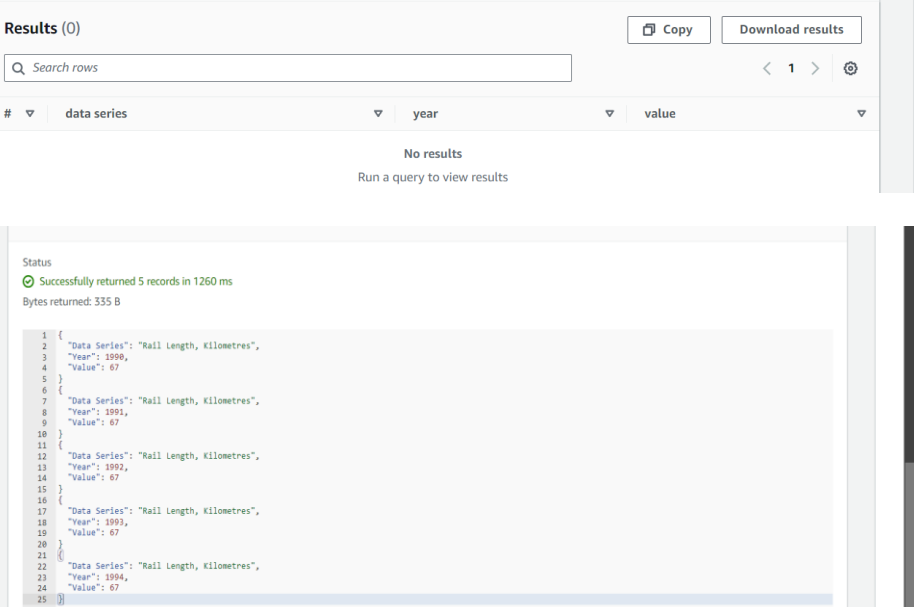
Cycle:	1
--------	---

Plan: Task & Resources	Implement AWS Lambda for ingestion, planned to use AWS academy lab and materials and reference to code provided by the teacher on teams
Perform:	Carried out the task on 22 June 2024, modifying from the code given by the teacher.
Monitor:	It was not successful. Asked ChatGPT for more information on the error and suggestions on what to do. (https://chatgpt.com/share/ae6db300-c143-4f03-9d81-6b6bc328197a). The error was due to the requests library not being installed.
Reflect:	Learned that the requests library is not included in the default Lambda environment. Decided to continue with installing of the library the next day.

Cycle:	2
Plan: Task & Resources	Install the request library with help from ChatGPT and YouTube videos
Perform:	Carried out the task on 23 June 2024, which required the use of the command prompt and jupyter notebook to download the library and zip the file to be uploaded to the lambda function. 
Monitor:	The function was executed with no errors and has been assigned to an s3 bucket for further processing down the pipeline.
Reflect:	Learned how to package and deploy third party libraries in AWS Lambda, which is a useful skill to have. The next cycle will focus on adding more features to the lambda function so that it will be able to handle larger datasets.
Cycle:	3
Plan: Task & Resources	Add another function that will be invoked in the main lambda function when there is a large dataset. I will be using aws materials and the lucid chart diagram done during the group project.

Perform:	Carried out the task on the same day and created ec2 instances which will be used to ingest the larger dataset in the lambda function and then upload it to the untransformedsingstatdata s3 folder, for data that is untransformed.
	
Monitor:	The function was executed with no errors and has been assigned to an s3 bucket for further processing down the pipeline.
Reflect:	Learned how to integrate lambda with ec2 for scalable data processing.

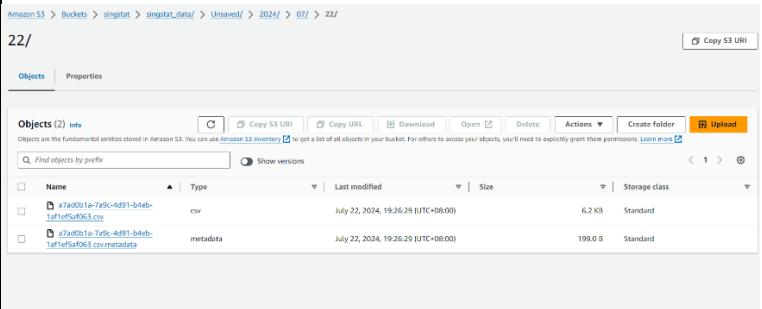
Task 2:

Cycle:	1
Plan: Task & Resources	After implementing the automategluecrawlerdatabrew lambda function, the next step would be to query the data in Athena. I will use aws learner lab and aws materials for the task.
Perform:	Performed the SELECT * FROM query in Athena to show the table and its data.
Monitor:	<p>The table was shown but there was no data inside it. This was weird because I could view the data in s3 query select.</p>  <pre> 1 { 2 "Data Series": "Rail Length, Kilometres", 3 "Year": 1990, 4 "Value": 67 5 } 6 { 7 "Data Series": "Rail Length, Kilometres", 8 "Year": 1991, 9 "Value": 67 10 } 11 { 12 "Data Series": "Rail Length, Kilometres", 13 "Year": 1992, 14 "Value": 67 15 } 16 { 17 "Data Series": "Rail Length, Kilometres", 18 "Year": 1993, 19 "Value": 67 20 } 21 { 22 "Data Series": "Rail Length, Kilometres", 23 "Year": 1994, 24 "Value": 67 25 } </pre> <p>Asked for ChatGPT's help (https://chatgpt.com/share/f47c0c9d-10db-4ad5-a0fb-97099f2fb843). ChatGPT gave some ideas on what could have went wrong.</p>

Reflect:	The issue could be due to the data format or something could be wrong with the steps I did in glue databrew. Decided to check all these the next day.
----------	---

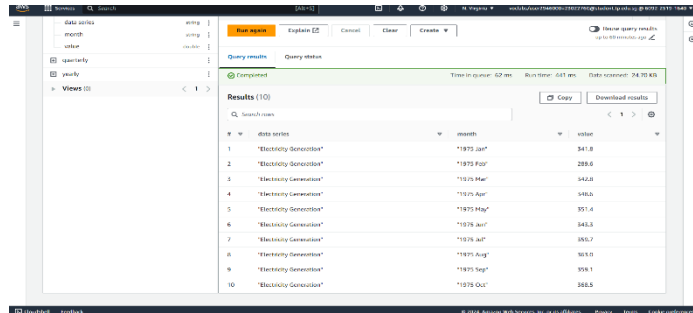
Cycle:	2
Plan: Task & Resources	Try out all the suggestions that ChatGPT gave and YouTube videos.
Perform:	Changed the format of the data and even remove the step where I merge the data which was a possible cause of error.
Monitor:	However, despite the multiple amount of attempts to try to figure out the error, I could not solve the problem.
Reflect:	I learned how to persevere and keep calm as I felt as if I wasted a lot of time trying to figure the error out but fail in the end. I planned to ask my teacher for help during the next class.

Cycle:	3
Plan: Task & Resources	Planned to ask my teacher for help during class.
Perform:	Asked my teacher and he told me that it could be a file error and that I should try again with a new crawler, new bucket, and new database.
Monitor:	I did what he asked me to do but the same thing happened. I then asked him again and after multiple attempts, it did not work. The problem was deeper than I thought.
Reflect:	Even though it was not working, it was already late, and I had to go home. I recognised that some issues are very complex and require a lot of patience and require multiple attempts to solve. I planned to go home and try fixing the error.

Cycle:	4
Plan: Task & Resources	Troubleshoot the error with the help of ChatGPT, YouTube and visiting websites for potential solutions.
Perform:	<p>When going through the s3 bucket, I noticed an unsaved folder in the same folder as the data used for Athena. I deleted the unsaved folder as I felt it may be the root cause of the error.</p> 

Monitor:

When I deleted the folder, I ran the automategluecrawlerdatabrew lambda function again and when I ran the SELECT * FROM query, the data was displayed.



The screenshot shows the Amazon Athena console interface. On the left, there's a sidebar with navigation options like 'data catalog', 'queries', 'views', and 'schemas'. The main area displays the 'Query results' for a specific query. The query is 'SELECT * FROM electricity_generation'. The results are shown in a table with columns 'data series', 'month', and 'value'. The table contains 10 rows of data, representing electricity generation for different months from 1970 to 1979.

#	data series	month	value
1	"Electricity Generation"	"1970 Jan"	341.8
2	"Electricity Generation"	"1970 Feb"	288.6
3	"Electricity Generation"	"1970 Mar"	342.0
4	"Electricity Generation"	"1970 Apr"	348.0
5	"Electricity Generation"	"1970 May"	351.4
6	"Electricity Generation"	"1970 Jun"	343.3
7	"Electricity Generation"	"1970 Jul"	350.7
8	"Electricity Generation"	"1970 Aug"	345.0
9	"Electricity Generation"	"1970 Sep"	346.1
10	"Electricity Generation"	"1970 Oct"	348.5

The unsaved folder was likely interfering with the data processing and querying in Athena.

Reflect:

The experience taught me to carefully inspect all the components in the data pipeline to ensure I can identify the causes of error. It also taught me a minor issue such as a folder can have a significant impact.