

PROJECT DETAILS

Student Name: **Purvi Waman Bhoyar**

Student ID: **24210585**

Project Title: **Medical Cost Charges Analysis using AWS cloud services**

CC Project: **2**

Due Date: **06/12/2024**

Submitted Date: **06/12/2024**

Application Overview -

The aim of this project is to design and implement a cloud-based solution for the analysis of medical datasets using distributed computing frameworks. The analysis of charges, demographics, smoking habits, and BMI from the medical data focuses on finding the trends and insights. This project uses AWS services like EMR, S3, Glue, AWS Session Manager Agent and QuickSight to efficiently process and visualize the data.

Objectives -

The objectives of the project were to:

1. **Data Preprocessing :**

- 1.a) Implement Distributed Data Processing with PySpark on AWS EMR.
- 1.b) Conduct the following focused analyses:
Average Charges by Region, Smoker Status, BMI Categories, and Age.
- 1.c) Provide statistics about charges - standard deviation and most extreme values.

2. **Visualization**

The processed data is to be visualized using AWS QuickSight for easier interpretation.

3. **Cloud-based Solution**

To implement a scalable, cost-effective cloud-based solution for handling medical datasets.
Automation

4. Automation

Automate processing and manifest creation for quick integration with QuickSight.

Status of the Project : Successful completion of all the above objectives and their verification based on AWS services.

Problem Statement and Dataset Description

Problem Statement

Medical datasets are usually huge and contain complex interrelations between various variables like demographic attributes, lifestyle factors, and medical expenses. The efficient processing of such data is rather vital to actionable insights, which can help healthcare providers in decision-making and in developing policies.

Dataset Description

The proposed dataset contains the following fields:

1. **Age:** Age of the person
2. **Sex:** Sex/Gender of the person
3. **BMI:** Body Mass Index.
4. **Children:** Number of children covered by the insurance.
5. **Smoker:** Smoking status (Yes/No).
6. **Region:** Geographic region.
7. **Charges:** Medical charges incurred.

The dataset contains 1,300 rows and provides a comprehensive snapshot of healthcare costs across different demographic and lifestyle factors.

Methodology and Implementation

Tools and Technologies

1. AWS S3:

Used for storing the raw medical dataset, processed outputs, AWS services log files and manifest files for visualization.

2. AWS EMR:

Utilized PySpark for distributed data processing and analysis of the data

3. AWS Glue:

It was utilized to test the functionality of the pyspark script, ensuring that the data processing logic produced accurate outputs and generated the required manifest files.

4. AWS QuickSight:

Used for data visualization and generating insights from processed data.

5. PySpark:

Allowed for efficient and easy data aggregation, transformation, and statistical analysis.

6. Boto3:

Automated the generation and upload of manifest files for QuickSight integration.

7. AWS EC2:

EC2 instance was vital in hosting the master and core nodes of EMR, thus ensuring stability and performance in the distributed environment.

Implementation Steps

1. Data Preprocessing:

Cleaned the dataset by removing rows containing missing or null values.
The cleaned data was stored in S3.

2. Data Analysis:

Conducted the following analyses:

- 2.a) Average, Maximum, and Minimum Charges by Region
- 2.b) Standard Deviation of Charges by Different Categories Based on BMI
- 2.c) Average Charges by Different Age Groups
- 2.d) Count of Smokers and Non-smokers in Different Regions
- 2.e) Comparison of Charges by Gender in Smokers and Non-smokers

3. Manifest Generation:

JSON manifest files were generated for all the outputted datasets for easy integration into QuickSight.

4. Visualization:

Imported manifest files into QuickSight in order to create dashboards and visualizations, including but not limited to bar charts, pie charts, and line graphs.

Worked Example

Analysis: Average Charges by Region, Sex, bmi, Smoking status and other features of the dataset.

1. Input:

Raw dataset is in s3://medicalcharges/MedicalCharges/Input_data/Charges.csv.

2. Processing:

PySpark aggregated the dataset into average charges by region.

The output was stored in s3://medicalcharges/MedicalCharges/Output_data/region_stats/.

3. Manifest:

manifest files were created in S3 bucket, sample path is given below :

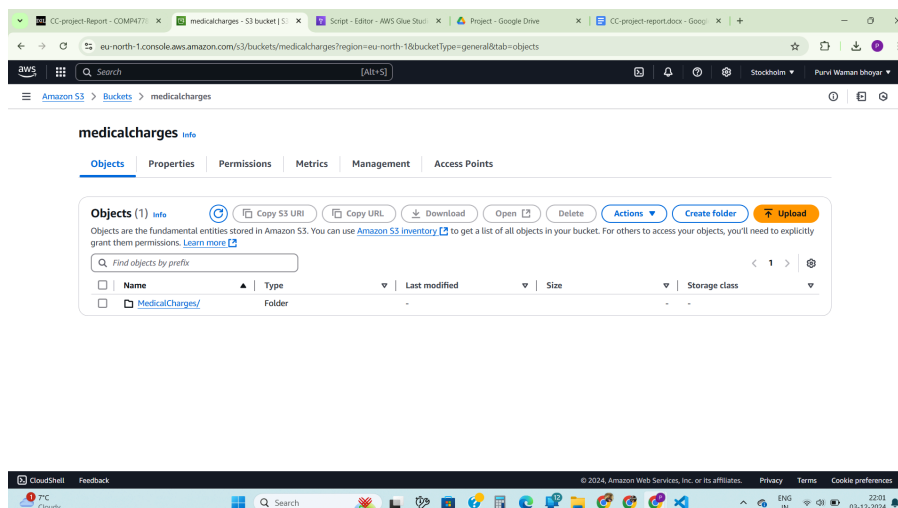
s3://medicalcharges/MedicalCharges/Manifest_data/region_stats_manifest.json

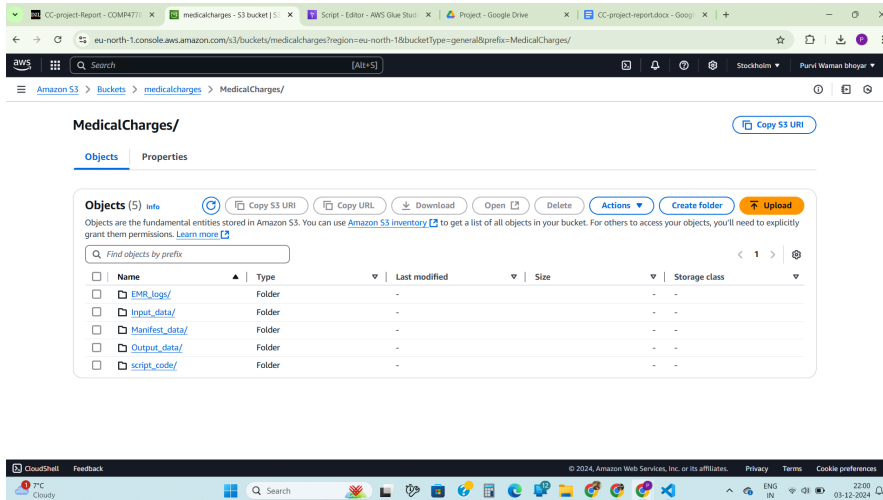
4. Visualization:

Loaded the dataset into QuickSight to perform visualisation of average charges across the regions along with other important plots such as pie charts, vertical and horizontal bar plots, donut shaped plots.

Screenshots as per the Workflow :

AWS S3 bucket main folder, containing every subfolder with input, output, logs and manifest files location.





PySpark Source code -

```
C:\> Users \Purvi> OneDrive \Desktop \CSNL_Assignments \CC \CC_Project \new_source_code.py > ...
1  import sys
2  import json
3  import boto3
4  from pyspark.context import SparkContext
5  from pyspark.sql import SparkSession
6  from pyspark.sql.functions import col, avg, count, max, min, stddev, when
7
8  # Initialize Spark
9  spark = SparkSession.builder.appName("ExtendedMedicalChargesAnalysisWithManifest").getOrCreate()
10
11 # Initialize S3 client
12 s3_client = boto3.client('s3')
13
14 input_path = "s3://medicalcharges/MedicalCharges/Input_data/Charges.csv"
15 output_path_base = "s3://medicalcharges/MedicalCharges/Output_data/"
16 manifest_base_path = "s3://medicalcharges/MedicalCharges/Manifest_data/"
17
18 def generate_and_upload_manifest(s3_output_path, manifest_s3_path):
19     """
20     Generates a JSON manifest file for QuickSight visualization and uploads it to S3.
21     Args:
22         s3_output_path (str): S3 path of the output data.
23         manifest_s3_path (str): S3 path to store the manifest file.
24     """
25     manifest_data = {
26         "fileLocations": [{"URIs": [s3_output_path]}],
27         "globalUploadSettings": {
28             "format": "CSV",
29             "delimiter": ",",
30             "textQualifier": "\"",
31             "containsHeader": True
32         }
33     }
34
35     # Create a local manifest file
36     local_manifest_path = "/tmp/manifest.json"
```

```

new_source_code.py 1 X
C:\Users\Purvi> OneDrive\ Desktop\ CSNL_Assignments\ CC\ CC_Project\ new_source_code.py > ...
18 def generate_and_upload_manifest(s3_output_path, manifest_s3_path):
19     local_manifest_csv_path = f"manifest_{manifest_s3_path}.json"
20     with open(local_manifest_path, 'w') as manifest_file:
21         json.dump(manifest_data, manifest_file, indent=4)
22
23     # Parse bucket and key from manifest_s3_path
24     bucket_name = manifest_s3_path.split("/")[2]
25     manifest_key = "/" + ".join(manifest_s3_path.split("/")[:3])
26
27     # Upload the manifest file to S3
28     s3_client.upload_file(local_manifest_path, bucket_name, manifest_key)
29     print(f"Manifest file uploaded to: {manifest_s3_path}")
30
31 # Load data into a DataFrame
32 data = spark.read.csv(input_path, header=True, inferSchema=True)
33
34 # Filter out null or invalid rows for required columns
35 data_filtered = data.filter((col("region").isNull()) &
36                             (col("smoker").isNull()) &
37                             (col("charges").isNull()) &
38                             (col("sex").isNull()) &
39                             (col("bmi").isNull()))
40
41 # Define a function to process and generate manifests for each output
42 def process_and_generate_manifest(output_folder, dataset_name):
43     output_path = f"{output_path_base}{output_folder}/"
44     manifest_path = f"{manifest_base_path}{dataset_name}_manifest.json"
45     return output_path, manifest_path
46
47 # 6. Average, max, and min charges by region
48 region_stats_output, region_stats_manifest = process_and_generate_manifest("region_stats", "region_stats")
49 data_filtered.groupBy("region").agg(
50     avg("charges").alias("average_charge"),
51     max("charges").alias("max_charge"),
52     min("charges").alias("min_charge")
53 ).write.csv(region_stats_output, mode="overwrite", header=True)
54 generate_and_upload_manifest(region_stats_output, region_stats_manifest)

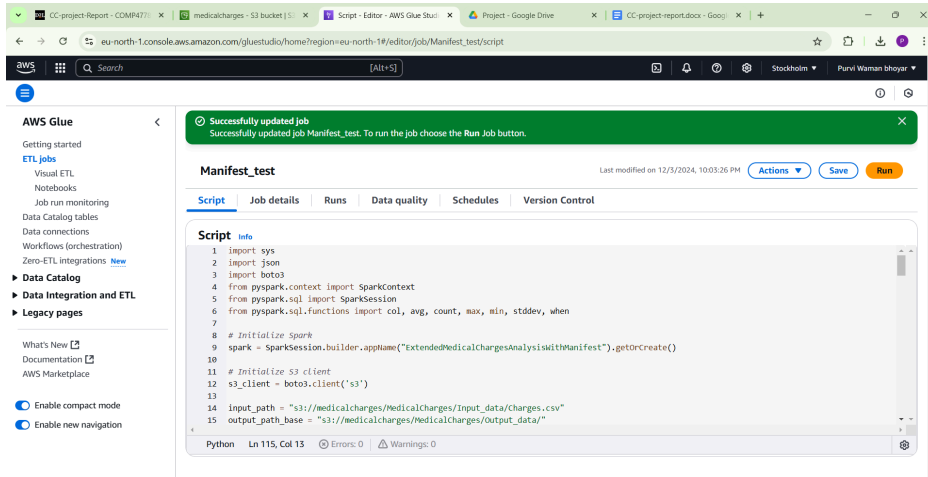
```

```

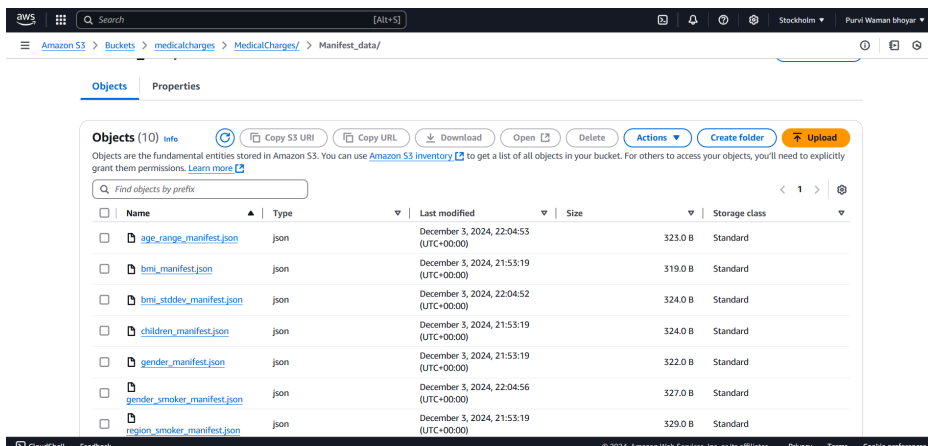
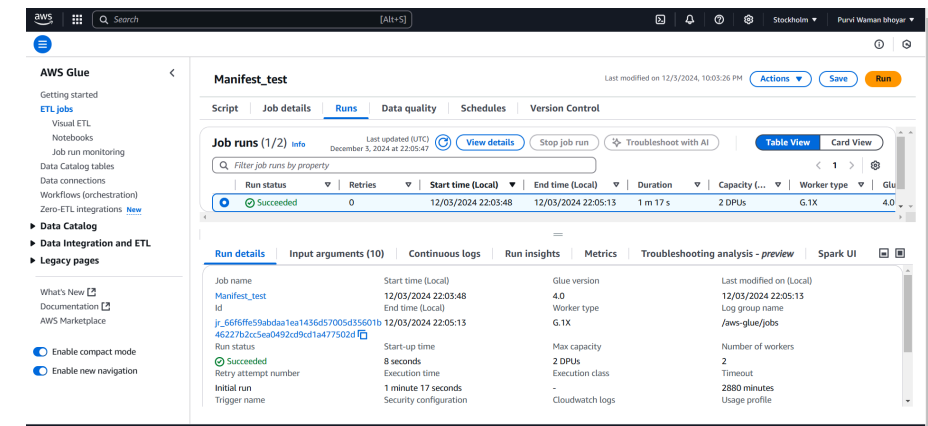
new_source_code.py 1 X
C:\Users\Purvi> OneDrive\ Desktop\ CSNL_Assignments\ CC\ CC_Project\ new_source_code.py > ...
73 # 7. Standard deviation of charges by BMI category
74 bmi_stddev_output, bmi_stddev_manifest = process_and_generate_manifest("bmi_stddev_analysis", "bmi_stddev")
75 data_filtered.withColumn(
76     "bmi_category",
77     when(col("bmi") < 18.5, "Underweight")
78     .when((col("bmi") >= 18.5) & (col("bmi") < 25), "Normal weight")
79     .when((col("bmi") >= 25) & (col("bmi") < 30), "Overweight")
80     .otherwise("Obese")
81 ).groupBy("bmi_category").agg(
82     stddev("charges").alias("stddev_charge")
83 ).write.csv(bmi_stddev_output, mode="overwrite", header=True)
84 generate_and_upload_manifest(bmi_stddev_output, bmi_stddev_manifest)
85
86 # 8. Average charges by age range
87 age_range_output, age_range_manifest = process_and_generate_manifest("age_range_analysis", "age_range")
88 data_filtered.withColumn(
89     "age_range",
90     when(col("age") < 20, "Below 20")
91     .when((col("age") >= 20) & (col("age") < 30), "20-29")
92     .when((col("age") >= 30) & (col("age") < 40), "30-39")
93     .when((col("age") >= 40) & (col("age") < 50), "40-49")
94     .otherwise("50 and above")
95 ).groupBy("age_range").agg(
96     avg("charges").alias("average_charge")
97 ).write.csv(age_range_output, mode="overwrite", header=True)
98 generate_and_upload_manifest(age_range_output, age_range_manifest)
99
100 # 9. Count of smokers and non-smokers by region
101 smoker_region_output, smoker_region_manifest = process_and_generate_manifest("smoker_region_analysis", "smoker_region")
102 data_filtered.groupBy("region", "smoker").agg(
103     count("*").alias("count")
104 ).write.csv(smoker_region_output, mode="overwrite", header=True)
105 generate_and_upload_manifest(smoker_region_output, smoker_region_manifest)
106
107 # 10. Charges comparison between males and females, grouped by smoker status
108 gender_smoker_output, gender_smoker_manifest = process_and_generate_manifest("gender_smoker_analysis", "gender_smoker")
109 data_filtered.groupBy("sex", "smoker").agg(
110     avg("charges").alias("average_charge")
111 ).write.csv(gender_smoker_output, mode="overwrite", header=True)
112 generate_and_upload_manifest(gender_smoker_output, gender_smoker_manifest)
113
114 # Stop the Spark session
115 spark.stop()

```

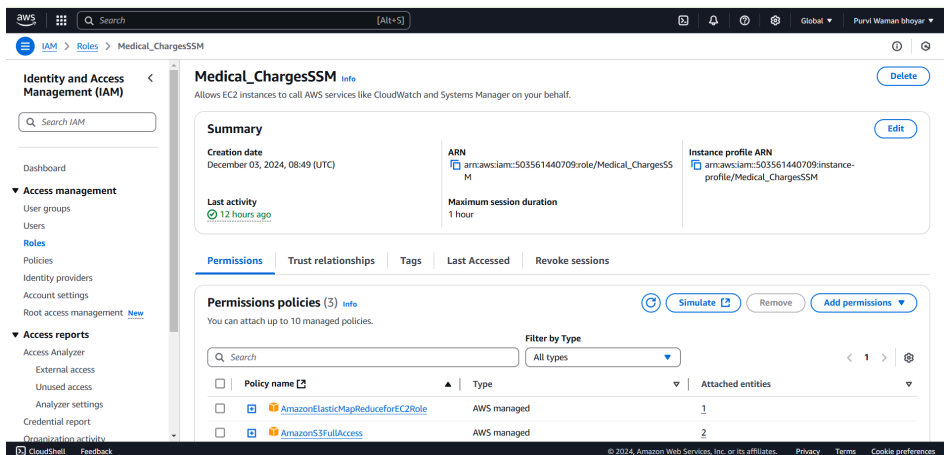
AWS Glue Service used for testing the source code -"Manifest_test" job



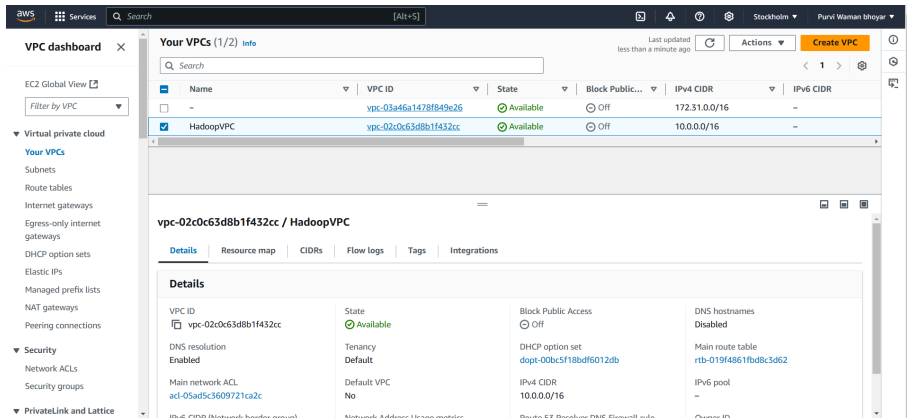
Succeeded Glue Job along with output files on the mentioned paths.



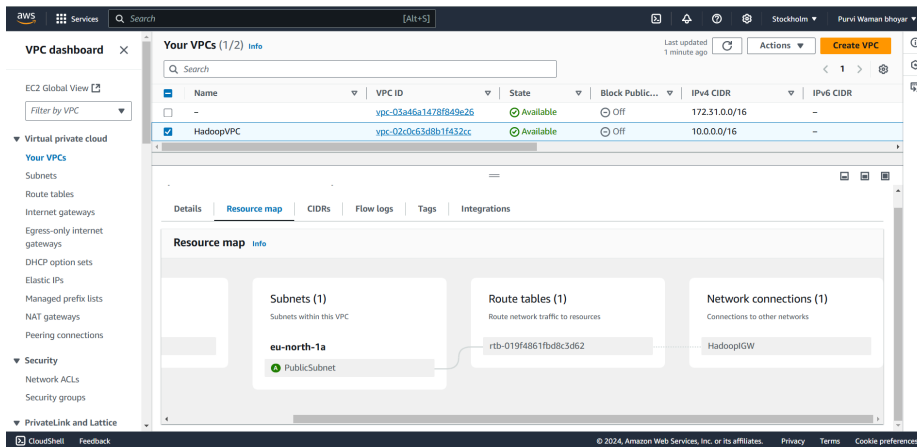
AWS IAM Role creation given full access for using AWS Session Manager, S3 Bucket and Elastic MapReduce.



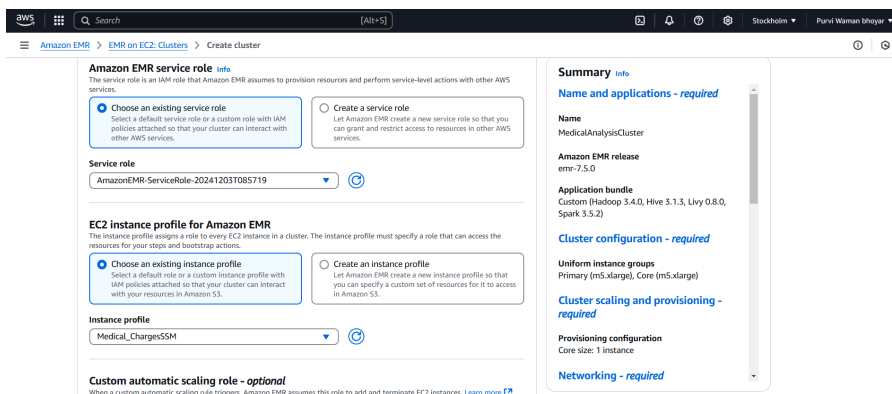
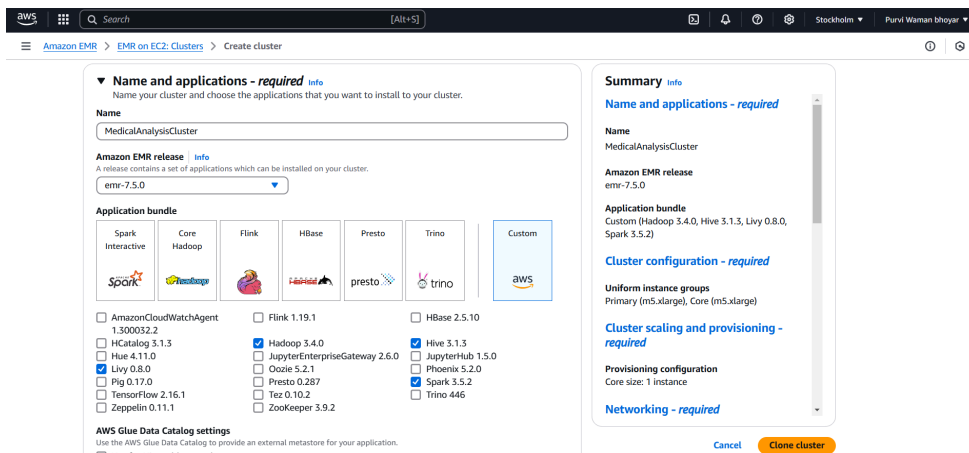
Using AWS VPC service, creating own VPC (Virtual Private Cloud) , Subnet, Routing table and Internet Gateway for creating EMR clusters further in the process .



VPC->Subnet->Routing Table> Internet Gateway flow shown below



AWS EMR on EC2 cluster creation with necessary requirements and roles selected



Amazon EMR

EMR on EC2 Clusters

MedicalAnalysisCluster

Your cluster "MedicalAnalysisCluster" has been successfully created.

Updated less than a minute ago

Terminate

Clone in AWS CLI

Clone

MedicalAnalysisCluster

Summary

Cluster info

Cluster ID
j-M32NFY9UDNHW

Cluster configuration
Instance groups

Capacity
1 Primary 1 Core 0 Task

Applications

Amazon EMR version
emr-7.5.0

Installed applications
Hadoop 3.4.0, Hive 3.1.3, Lily 0.8.0, Spark 3.5.2

Cluster management

Log destination in Amazon S3
medicalcharges/Medical_Charges/EMR_logs

Persistent application Uls
Spark History Server
YARN timeline server
Tez UI

Primary node public DNS
16.171.115.65
Connect to the Primary node using SSH
Connect to the Primary node using SSM

Status and time

Status
Waiting

Creation time
December 03, 2024, 22:15 (UTC+00:00)

Elapsed time
8 minutes, 24 seconds

Properties

Bootstrap actions

Instances (Hardware)

Steps

Applications

Configurations

Monitoring

Events

Tags (1)

Operating system

Amazon Linux release 2023.4_2024.02.01

Cluster logs

Archive log files to Amazon S3

Cluster termination and node replacement

CloudShell

Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Successful EMR cluster created -

Amazon EMR

EMR on EC2 Clusters

MedicalAnalysisCluster

Your cluster "MedicalAnalysisCluster" has been successfully created.

Updated less than a minute ago

Terminate

Clone in AWS CLI

Clone

MedicalAnalysisCluster

Summary

Cluster info

Cluster ID
j-M32NFY9UDNHW

Cluster configuration
Instance groups

Capacity
1 Primary 1 Core 0 Task

Applications

Amazon EMR version
emr-7.5.0

Installed applications
Hadoop 3.4.0, Hive 3.1.3, Lily 0.8.0, Spark 3.5.2

Cluster management

Log destination in Amazon S3
medicalcharges/Medical_Charges/EMR_logs

Primary node public DNS
-

Status and time

Status
Starting

Creation time
December 03, 2024, 22:15 (UTC+00:00)

Elapsed time
1 second

Properties

Bootstrap actions

Instances (Hardware)

Steps

Applications

Configurations

Monitoring

Events

Tags (1)

Cluster logs

Archive log files to Amazon S3
Turned on

Amazon S3 location
s3://medicalcharges/Medical_Charges/EMR_logs/

Encryption for logs
Turned off

Cluster termination and node replacement

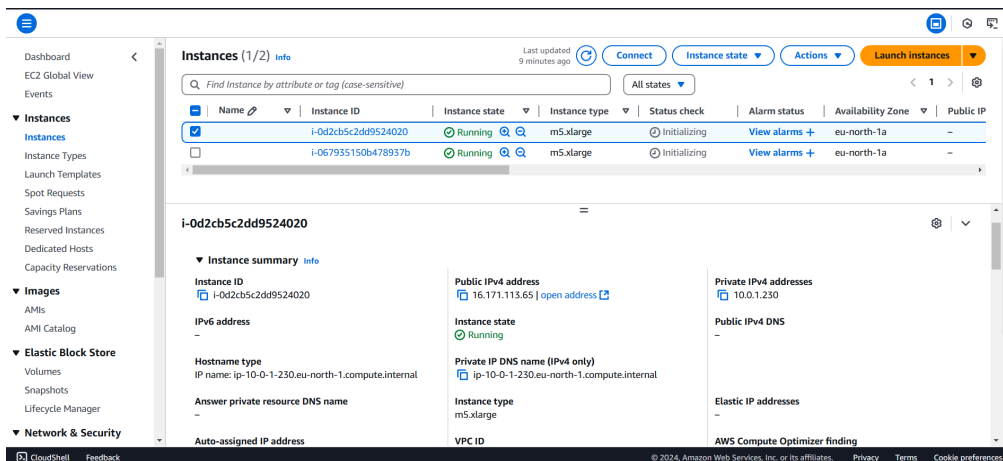
Termination option
Automatically terminate cluster after idle time

Termination protection
Off

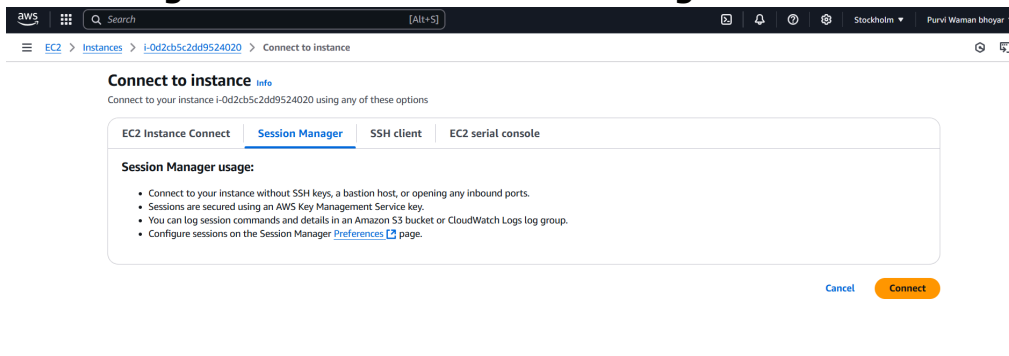
Idle time
1 hour

Unhealthy node replacement
On

EC2 instance running simultaneously



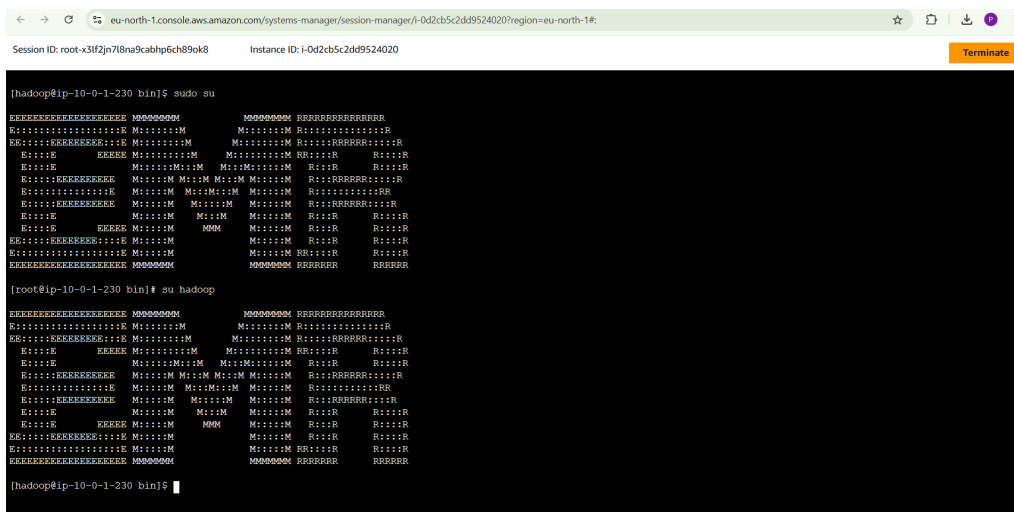
Connecting the cluster with Session Manager



EMR Session Manager session starting, initial commands and outputs.

Steps:

1. `sudo su`
2. `su Hadoop`



Making directory “Medicalproject” and copying the source_code.py into it .

```
[hadoop@ip-10-0-1-230 bin]$ cd
[hadoop@ip-10-0-1-230 ~]$ mkdir Medicalproject
[hadoop@ip-10-0-1-230 ~]$ cd Medicalproject
```

```
[hadoop@ip-10-0-1-230 Medicalproject]$ aws s3 cp s3://medicalcharges/MedicalCharges/script_code/source_code.py .
download: s3://medicalcharges/MedicalCharges/script_code/source_code.py to ./source_code.py
```

```
[hadoop@ip-10-0-1-230 Medicalproject]$ spark-submit source_code.py
24/12/03 22:20:59 INFO EMRParamsSideChannel: Setting RGAC mode to false
24/12/03 22:20:59 INFO SparkContext: Running Spark version 3.5.2-amzn-1
24/12/03 22:20:59 INFO SparkContext: OS info Linux, 6.1.112-124.190.amzn2023.x86_64, amd64
24/12/03 22:20:59 INFO SparkContext: Java version 17.0.13
24/12/03 22:20:59 INFO ResourceUtils: =====
24/12/03 22:20:59 INFO ResourceUtils: No custom resources configured for spark.driver.
24/12/03 22:20:59 INFO ResourceUtils: =====
24/12/03 22:20:59 INFO SparkContext: Submitted application: ExtendedMedicalChargesAnalysisWithManifest
24/12/03 22:20:59 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(executorType -> name: executorType, amount: 1, script: , vendor: , cores -
> name: cores, amount: 2, script: , vendor: , memory -> name: memory, amount: 4743, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task reso
urces: Map(cpus -> name: cpus, amount: 1.0)
24/12/03 22:20:59 INFO ResourceProfile: Limiting resource is cpus at 2 tasks per executor
24/12/03 22:20:59 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/12/03 22:21:00 INFO ResourceProfile: User executor ResourceProfile created, executor resources: Map(executorType -> name: executorType, amount: 1, script: , vendor: , c
ores -> name: cores, amount: 2, script: , vendor: , memory -> name: memory, amount: 4743, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), tas
k resources: Map(cpus -> name: cpus, amount: 1.0)
24/12/03 22:21:00 INFO ResourceProfile: Limiting resource is cpus at 2 tasks per executor
24/12/03 22:21:00 INFO ResourceProfileManager: Added ResourceProfile id: 1
24/12/03 22:21:00 INFO SecurityManager: Changing view acls to: hadoop
24/12/03 22:21:00 INFO SecurityManager: Changing modify acls to: hadoop
24/12/03 22:21:00 INFO SecurityManager: Changing view acls groups to:
24/12/03 22:21:00 INFO SecurityManager: Changing modify acls groups to:
```

As we can see from the below screenshot, the output files and manifest files. Generated files are getting stored in respective folders .

aws

Search

[Alt+S]

Stockholm

Purvi Waman bhoir

Amazon S3

Buckets

medicalcharges

MedicalCharges/

MedicalCharges/

Copy S3 URI

Objects

Properties

Objects (6)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	EMR_logs/	Folder	-	-	-
<input type="checkbox"/>	Input_data/	Folder	-	-	-
<input type="checkbox"/>	Manifest_data/	Folder	-	-	-
<input type="checkbox"/>	Output_data_folders	-	December 3, 2024, 22:22:12 (UTC+00:00)	0 B	Standard
<input type="checkbox"/>	Output_data/	Folder	-	-	-
<input type="checkbox"/>	script_code/	Folder	-	-	-

CloudShell

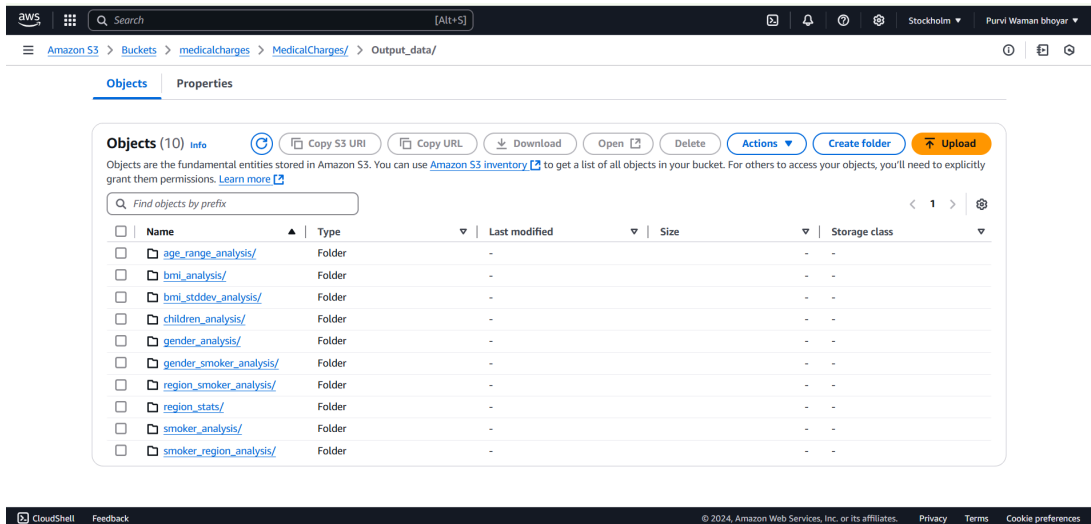
Feedback

© 2024 Amazon Web Services, Inc. or its affiliates

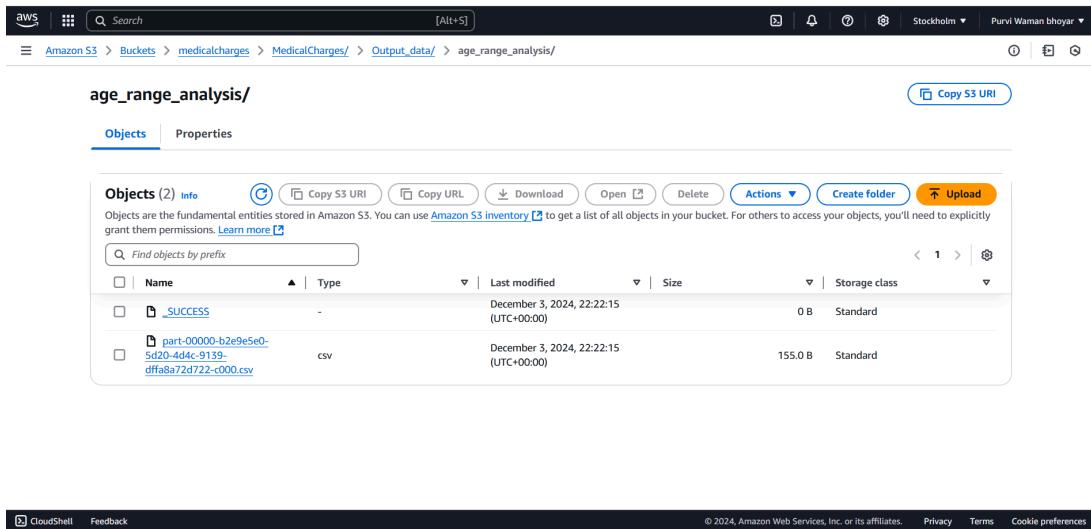
Privacy

Terms

Cookie preferences



Snapshots of one of the output files created on the dedicated path.



Snapshots showing output log file and EMR logs files getting stored in S3 bucket during the execution of EMR cluster.

aws

Search

[Alt+S]

Stockholm

Purni Waman bhojar

Amazon S3> Buckets> medicalcharges> MedicalCharges/> Output_data_\$folder\$

Output_data_\$folder\$ Info

Copy S3 URICopy S3 URIDownloadDownloadOpenOpenObject actionsObject actions

PropertiesPermissionsVersions

Object overview

Owner

5339647885fc2b0e7090247e5dfee8a064e049f833cb613a80bef742c631257d

AWS Region

Europe (Stockholm) eu-north-1

Last modified

December 3, 2024, 22:22:12 (UTC+00:00)

Size

-

Type

-

Key

MedicalCharges/Output_data_\$folder\$

S3 URI

s3://medicalcharges/MedicalCharges/Output_data_\$folder\$

Amazon Resource Name (ARN)

arn:aws:s3::medicalcharges/MedicalCharges/Output_data_\$folder\$

Entity tag (Etag)

d41d8cd98f00b204e9800998ecf8427e

Object URL

https://medicalcharges.s3.eu-north-1.amazonaws.com/MedicalCharges/Output_data_%24folder%24

Object management overview

The following bucket properties and object management configurations impact the behavior of this object.

CloudShellFeedback

© 2024, Amazon Web Services, Inc. or its affiliates. PrivacyTermsCookie preferences

aws

Search

[Alt+S]

Stockholm

Purni Waman bhojar

Amazon S3> Buckets> medicalcharges> MedicalCharges/> EMR_logs/> i-51OCR45NKKIO/> node/

node/

Copy S3 URICopy S3 URI

ObjectsPermissions

Objects (2) Info

Copy S3 URICopy S3 URIDownloadDownloadOpenOpenDeleteDeleteActionsActionsCreate folderCreate folderUploadUpload

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 explicitly grant them permissions. [Learn more](#)

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	i-0099c102b4c295352/	Folder	-	-	-
<input type="checkbox"/>	i-0ad8a5ad606b7127d/	Folder	-	-	-

Below Snapshot is the list of all EMR on EC2 clusters created for the testing /implementation of the project.

aws

Search

[Alt+S]

Stockholm

Purni Waman bhojar

Amazon EMR> EMR on EC2: Clusters

Amazon EMR

EMR Serverless

EMR on EC2 Clusters

Notebooks and Git repos

Events

Block public access

Security configurations

EMR on EKS

Virtual clusters

EMR Studio

Getting Started

Studios

Workspaces (Notebooks)

What's New

Video tour

Compact mode

Clusters (5) Info

View detailsView detailsTerminateTerminateCloneCloneCreate clusterCreate cluster

Filter clusters by statusFilter clusters by statusFind clustersFind clustersFilter clusters by creation date-timeFilter clusters by creation date-time

	Cluster ID	Cluster name	Status	Creation time (UTC+00:00)	Elapsed time
<input type="checkbox"/>	i-M32NFY9UDNHW	MedicalAnalysisCluster	Terminating User request	December 03, 2024, 22:15	12 minutes, 52 seconds
<input type="checkbox"/>	i-51OCR45NKKIO	MedicalAnalysisCluster	Terminated User request	December 03, 2024, 08:57	37 minutes, 37 seconds
<input type="checkbox"/>	i-5LRM3WVCY2N	MedicalAnalysisCluster	Terminated with errors Validation error	December 03, 2024, 08:54	41 seconds
<input type="checkbox"/>	i-31IOMW7296NGT	MedicalAnalysisCluster	Terminated User request	December 03, 2024, 08:25	26 minutes, 4 seconds
<input type="checkbox"/>	i-36QVLLR2HD072	MedicalCluster	Terminated User request	November 26, 2024, 09:25	57 minutes, 51 seconds

Below Snapshot is the list of all IAM roles created for the testing of scripts/Glue jobs/ EMR and S3 access specific clusters .

aws

Search

[Alt+S]

Global

Purvi Waman bhojar

IAM > Roles

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

Users groups

Users

Roles

Policies

Identity providers

Account settings

Root access management

Access reports

Access Analyzer

External access

Unused access

Analyzer settings

Credential report

Organization activity

Roles (9)

Info

Delete

Create role

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Search

< 1 >

<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	AmazonEMR-InstanceProfile-20241125T092751	AWS Service: ec2	13 hours ago
<input type="checkbox"/>	AmazonEMR-ServiceRole-20241126T092551	AWS Service: elasticmapreduce	13 hours ago
<input type="checkbox"/>	AmazonEMR-ServiceRole-20241203T085719	AWS Service: elasticmapreduce	8 minutes ago
<input type="checkbox"/>	aws-quicksight-service-role-v0	AWS Service: quicksight	9 hours ago
<input type="checkbox"/>	AWSServiceRoleForEMRCleanup	AWS Service: elasticmapreduce (Ser...	11 hours ago
<input type="checkbox"/>	AWSServiceRoleForSupport	AWS Service: support (Service-Link...	-
<input type="checkbox"/>	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service...	-
<input type="checkbox"/>	Glue-ETL-Job	AWS Service: glue	25 minutes ago
<input type="checkbox"/>	Medical_ChargesSSM	AWS Service: ec2	9 minutes ago

Roles Anywhere

info

Authenticate your non AWS workloads and securely provide access to AWS services.

Manage

CloudShell

Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

QuickSight Snapshots -

List of all graphs and insights generated using AWS QuickSight.

QuickSight

Username: 503561440709
Account name: PurviBhojar
Manage QuickSight
English
Stockholm
Help
Sign out

Find analyses & more

Favorites

Recent

My folders

Shared folders

Dashboards

Analyses

Datasets

Community

Analyses

Last updated (newest first)

Analysis

62.01K

Age Range VS Average su...

Updated a few seconds ago

Analysis

children_charges vs no of c...

Updated a few seconds ago

Analysis

gender_smoker_charges.cs...

Updated 2 minutes ago

Analysis

region_smoker_count.csv a...

Updated a day ago

Analysis

Gender vs average charges

Updated a day ago

Analysis

smoking vs charges

Updated a day ago

Analysis

sum of average charges by ...

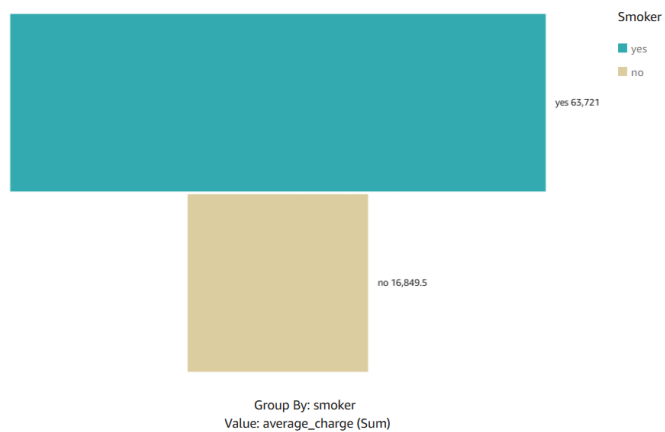
Updated a day ago

Analysis

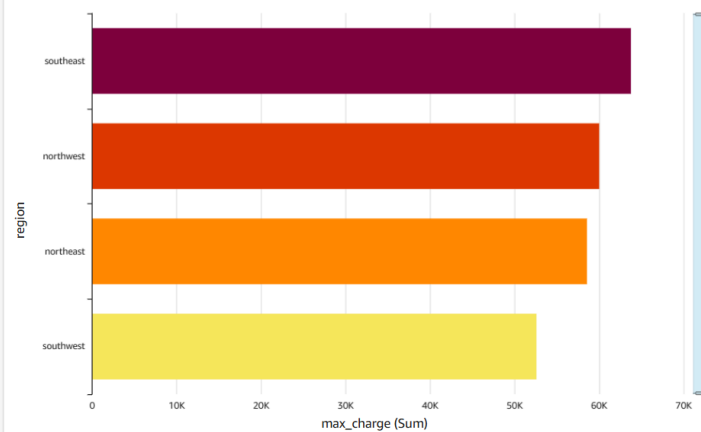
bmi VS average charges

Updated a day ago

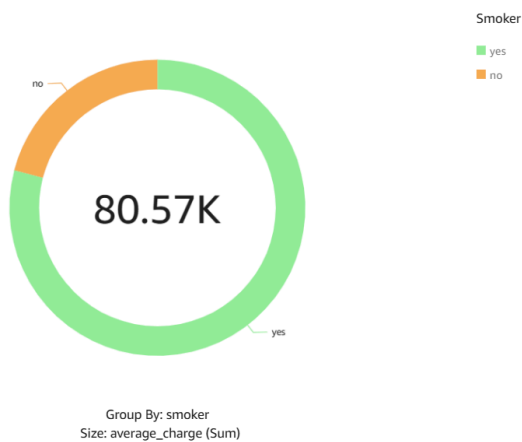
Sum of Average_charge by Smoker



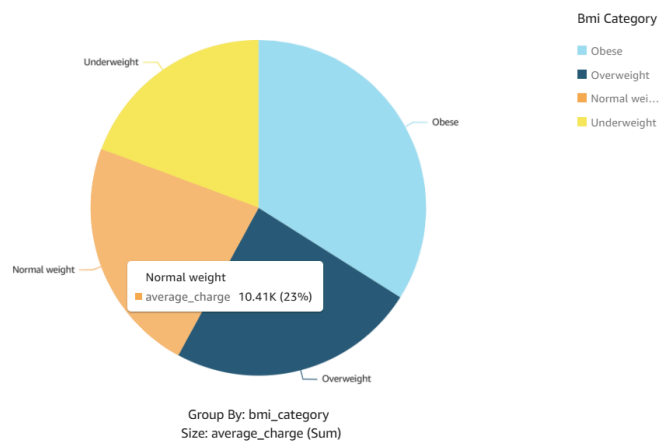
Sum of Max_charge by Region



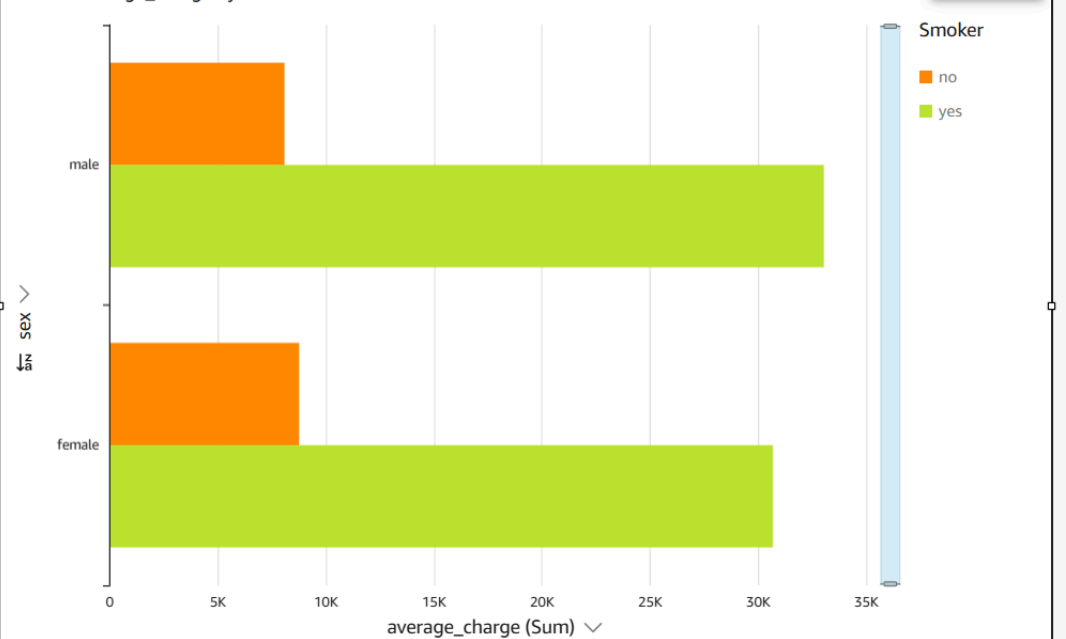
Sum of Average_charge by Smoker



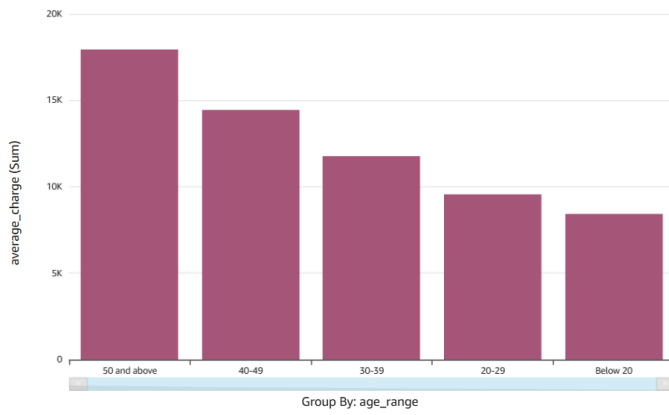
Sum of Average_charge by Bmi_category



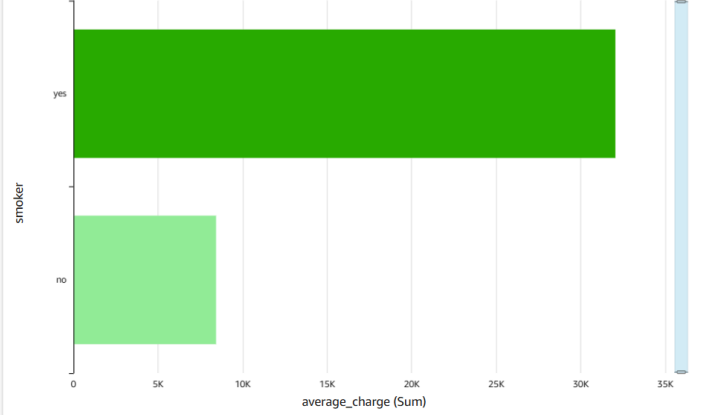
Sum of Average_charge by Sex and Smoker



Sum of Average_charge by Age_range



Sum of Average_charge by Smoker



Sum of Average_charge by Sex



Sex

■ male

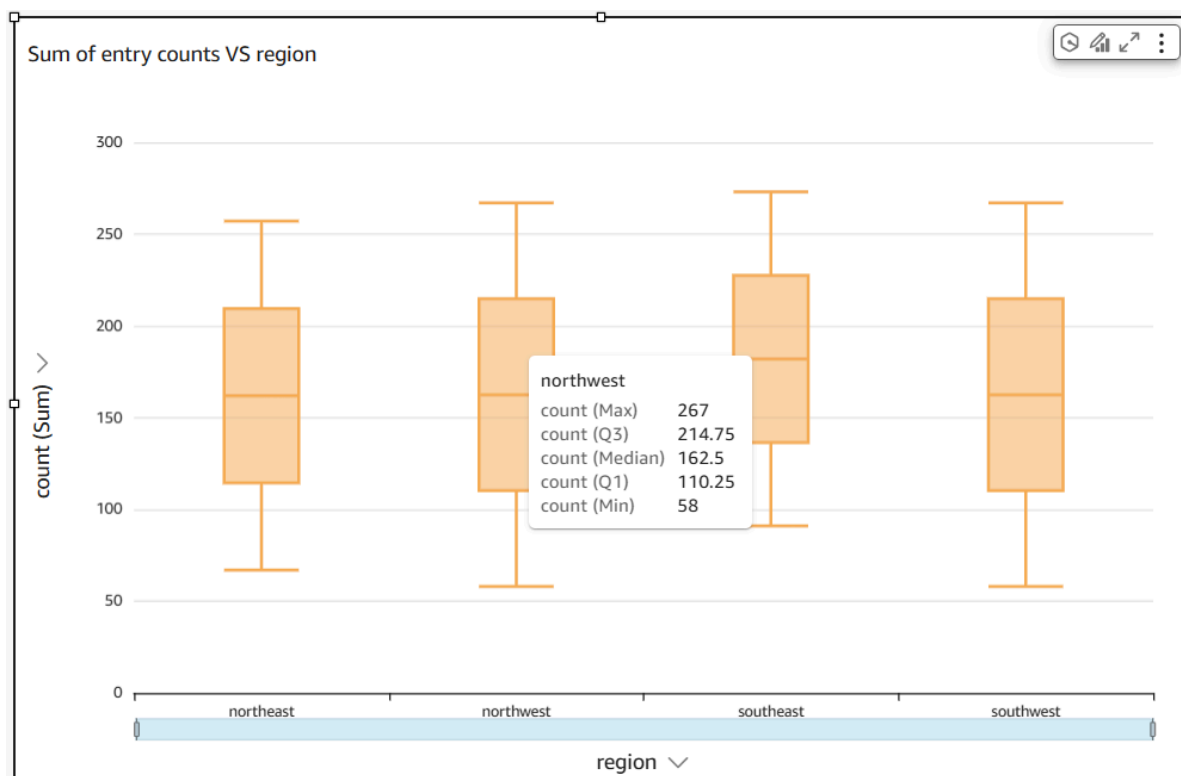
■ female

male 41,129.21

female 39,441.29

Group By: sex

Value: average_charge (Sum)



Suitability of Tools

AWS EMR and QuickSight proved to be highly suitable for this project owing to their scalability and ease of integration:

1. EMR:

Distributed processing was possible, thus reducing runtime for data analysis.

2. QuickSight:

Complex data visualization was easier, with provision for interaction in data.

3. S3:

Reliable cost-effective storage both for input and output data.

4. Glue:

Glue efficiently, accurately and in less time tested the pyspark script of the project and generated the output files.

5. AWS VPC:

Setup networking for the EMR cluster to be isolated but in a secure manner to enable all services to communicate and share data securely.

6. AWS EC2:

EC2 instance was vital in hosting the master and core nodes of EMR, thus ensuring stability and performance in the distributed environment.

7. Boto3:

Automated the generation and upload of manifest files for QuickSight integration.

Features of the Software

1. Automated Analysis:

PySpark allows for efficient processing of large medical datasets.

2. Dynamic Manifest Generation:

It automatically generates manifest files that can be integrated with QuickSight.

3. Scalability:

The cloud-based solution is highly scalable for larger datasets.

4. Interactive Visualization:

Easy-to-understand dashboards and visualizations are possible with QuickSight.

Conclusion

This project demonstrates how AWS services can be leveraged in the analysis and visualization of medical datasets. It was effective, as well as scalable, to handle the solution on the cloud for seamless integration between data processing and visualization tools. Insights derived from this dataset can be of immense help in understanding healthcare cost patterns and driving informed data-driven decisions.

References

1. AWS Documentation: [Amazon EMR](#)
Official documentation for Amazon Elastic MapReduce (EMR), detailing its functionality and use cases.
2. AWS Documentation: [Amazon S3](#)
Comprehensive guide on Amazon Simple Storage Service (S3) for object storage and data management.
3. AWS Documentation: [Amazon Glue](#)
Overview and technical documentation for AWS Glue, covering ETL workflows and integration capabilities.
4. AWS Documentation: [Amazon QuickSight](#)
Information about Amazon QuickSight, its features, and steps for creating interactive dashboards.
5. PySpark Documentation: [PySpark](#)
Official API documentation for PySpark, including distributed data processing and analysis features.
6. Dataset Source: [Medical Dataset](#)
Dataset used in this project, sourced from Kaggle.