CLOUD COMPUTING PROJECT

Cloud Computing

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.

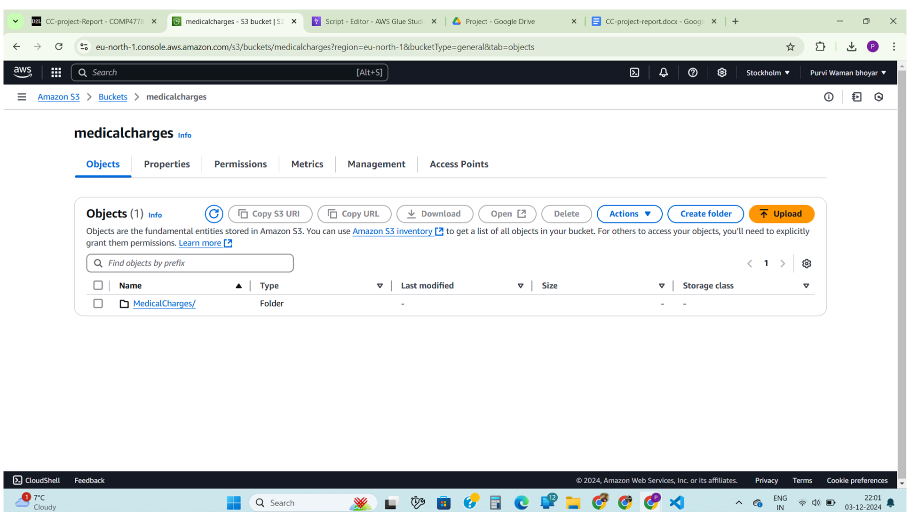
2. Processing:
   PySpark aggregated the dataset into average charges by region.
   The output was stored in s3://medicalcharges/MedicalCharges/Output_data/region_stats/.

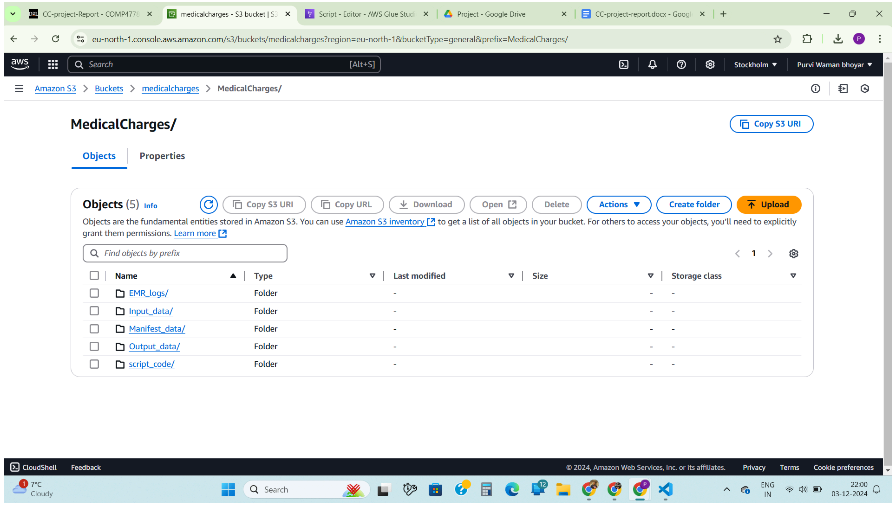
3. Manifest:
   manifest files were created in S3 bucket, sample path is given below :
s3://medicalcharges/MedicalCharges/Manifest_data/region_stats_manifest.json

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.

<u>Screenshots as per the Workflow :</u>

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 > ...

```python
import sys
import json
import boto3
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, count, max, min, stddev, when

# Initialize Spark
spark = SparkSession.builder.appName("ExtendedMedicalChargesAnalysisWithManifest").getOrCreate()

# Initialize S3 client
s3_client = boto3.client('s3')

input_path = "s3://medicalcharges/MedicalCharges/Input_data/Charges.csv"
output_path_base = "s3://medicalcharges/MedicalCharges/Output_data/"
manifest_base_path = "s3://medicalcharges/MedicalCharges/Manifest_data/"

def generate_and_upload_manifest(s3_output_path, manifest_s3_path):
    """
    Generates a JSON manifest file for QuickSight visualization and uploads it to S3.
    Args:
        s3_output_path (str): S3 path of the output data.
        manifest_s3_path (str): S3 path to store the manifest file.
    """
    manifest_data = {
        "fileLocations": [{"URIs": [s3_output_path]}],
        "globalUploadSettings": {
            "format": "CSV",
            "delimiter": ",",
            "textQualifier": "\"",
            "containsHeader": True
        }
    }

    # Create a local manifest file
    local_manifest_path = "/tmp/manifest.json"
```

new_source_code.py 1

C: > Users > Purvi > OneDrive > Desktop > CSNL_Assignments > CC > CC_Project > new_source_code.py > ...

```python
def generate_and_upload_manifest(s3_output_path, manifest_s3_path):
    with open(local_manifest_path, 'w') as manifest_file:
        json.dump(manifest_data, manifest_file, indent=4)

    # Parse bucket and key from manifest_s3_path
    bucket_name = manifest_s3_path.split("/")[2]
    manifest_key = "/".join(manifest_s3_path.split("/")[3:])

    # Upload the manifest file to S3
    s3_client.upload_file(local_manifest_path, bucket_name, manifest_key)
    print(f"Manifest file uploaded to: {manifest_s3_path}")

# Load data into a DataFrame
data = spark.read.csv(input_path, header=True, inferSchema=True)

# Filter out null or invalid rows for required columns
data_filtered = data.filter((col("region").isNotNull()) &
                            (col("smoker").isNotNull()) &
                            (col("charges").isNotNull()) &
                            (col("sex").isNotNull()) &
                            (col("bmi").isNotNull()))

# Define a function to process and generate manifests for each output
def process_and_generate_manifest(output_folder, dataset_name):
    output_path = f"{output_path_base}{output_folder}/"
    manifest_path = f"{manifest_base_path}{dataset_name}_manifest.json"
    return output_path, manifest_path

# 6. Average, max, and min charges by region
region_stats_output, region_stats_manifest = process_and_generate_manifest("region_stats", "region_stats")
data_filtered.groupBy("region").agg(
    avg("charges").alias("average_charge"),
    max("charges").alias("max_charge"),
    min("charges").alias("min_charge")
).write.csv(region_stats_output, mode="overwrite", header=True)
generate_and_upload_manifest(region_stats_output, region_stats_manifest)
```

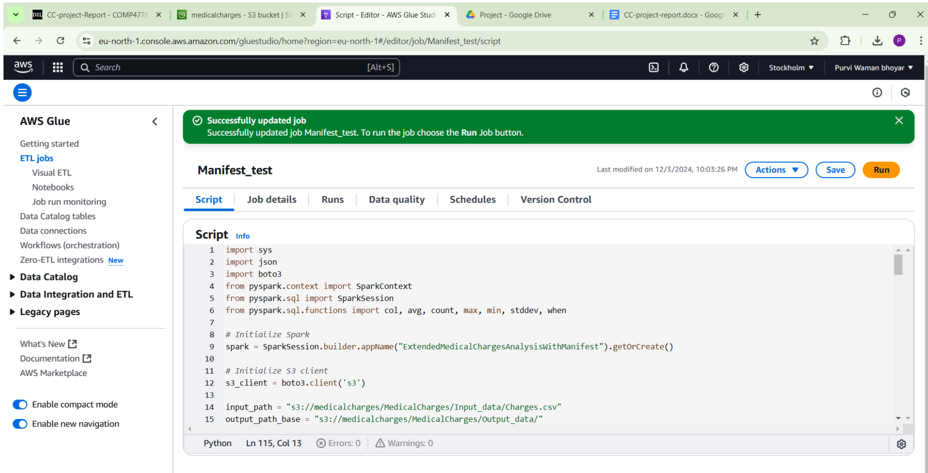
new_source_code.py 1

C: > Users > Purvi > OneDrive > Desktop > CSNL_Assignments > CC > CC_Project > new_source_code.py > ...

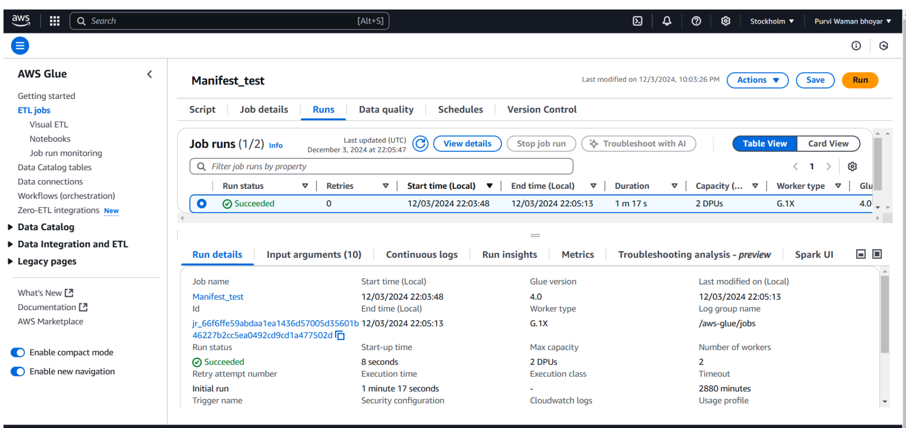
```python
# 7. Standard deviation of charges by BMI category
bmi_stddev_output, bmi_stddev_manifest = process_and_generate_manifest("bmi_stddev_analysis", "bmi_stddev")
data_filtered.withColumn(
    "bmi_category",
    when(col("bmi") < 18.5, "Underweight")
    .when((col("bmi") >= 18.5) & (col("bmi") < 25), "Normal weight")
    .when((col("bmi") >= 25) & (col("bmi") < 30), "Overweight")
    .otherwise("Obese")
).groupBy("bmi_category").agg(
    stddev("charges").alias("stddev_charge")
).write.csv(bmi_stddev_output, mode="overwrite", header=True)
generate_and_upload_manifest(bmi_stddev_output, bmi_stddev_manifest)

# 8. Average charges by age range
age_range_output, age_range_manifest = process_and_generate_manifest("age_range_analysis", "age_range")
data_filtered.withColumn(
    "age_range",
    when(col("age") < 20, "Below 20")
    .when((col("age") >= 20) & (col("age") < 30), "20-29")
    .when((col("age") >= 30) & (col("age") < 40), "30-39")
    .when((col("age") >= 40) & (col("age") < 50), "40-49")
    .otherwise("50 and above")
).groupBy("age_range").agg(
    avg("charges").alias("average_charge")
).write.csv(age_range_output, mode="overwrite", header=True)
generate_and_upload_manifest(age_range_output, age_range_manifest)

# 9. Count of smokers and non-smokers by region
smoker_region_output, smoker_region_manifest = process_and_generate_manifest("smoker_region_analysis", "smoker_region")
data_filtered.groupBy("region", "smoker").agg(
    count("*").alias("count")
).write.csv(smoker_region_output, mode="overwrite", header=True)
generate_and_upload_manifest(smoker_region_output, smoker_region_manifest)

# 10. Charges comparison between males and females, grouped by smoker status
gender_smoker_output, gender_smoker_manifest = process_and_generate_manifest("gender_smoker_analysis", "gender_smoker")
data_filtered.groupBy("sex", "smoker").agg(
```

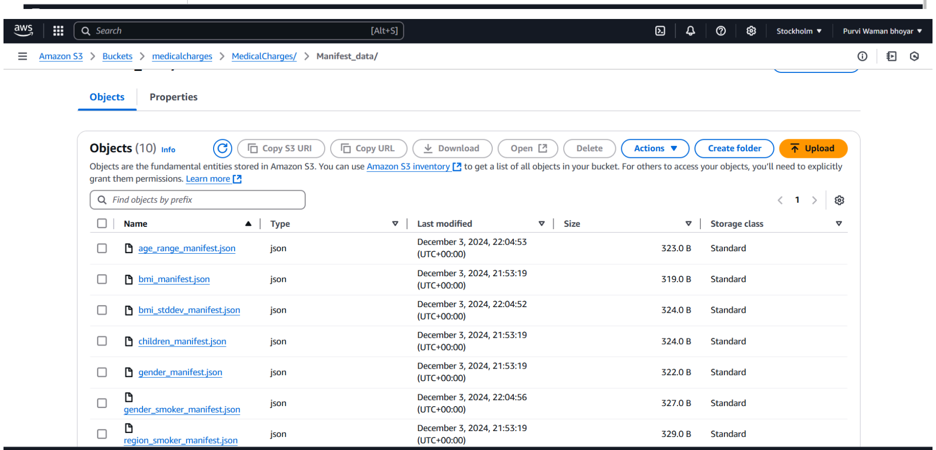
```python
# 10. Charges comparison between males and females, grouped by smoker status
gender_smoker_output, gender_smoker_manifest = process_and_generate_manifest("gender_smoker_analysis", "gender_smoker")
data_filtered.groupBy("sex", "smoker").agg(
    avg("charges").alias("average_charge")
).write.csv(gender_smoker_output, mode="overwrite", header=True)
generate_and_upload_manifest(gender_smoker_output, gender_smoker_manifest)

# Stop the Spark session
spark.stop()
```

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

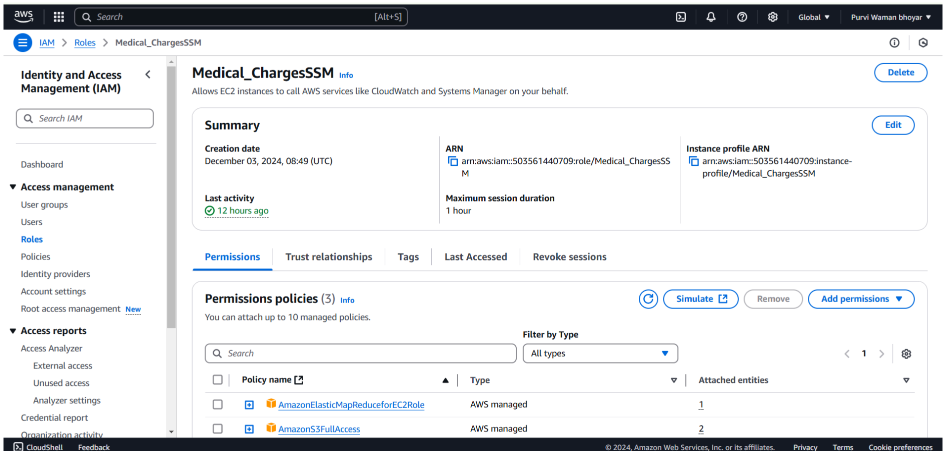


Succeeded Glue Job along with output files on the mentioned paths.

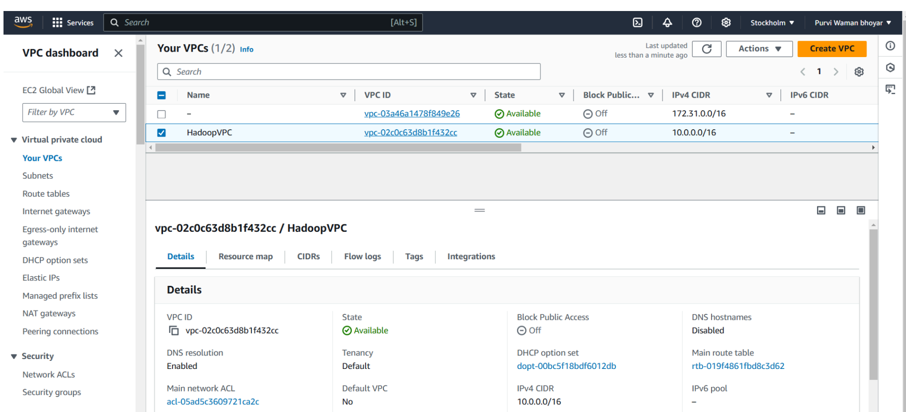




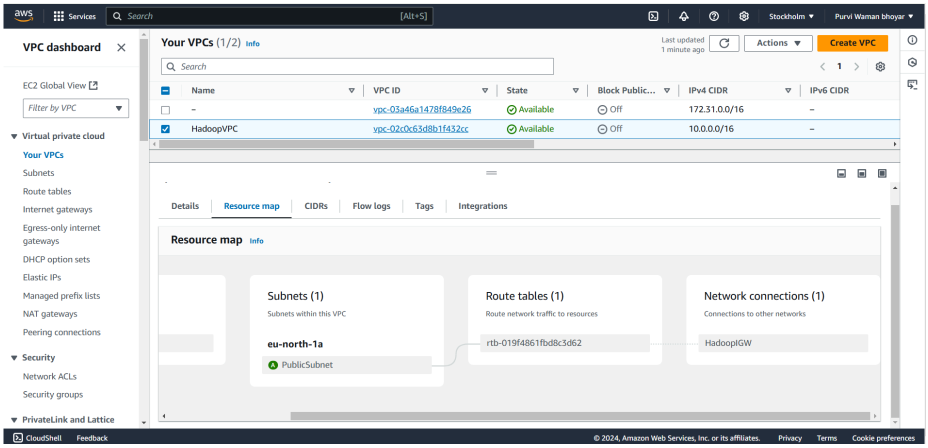
AWS IAM Role creation given full access for using AWS Session Manager, S3 Bucket and Elastic MapReduce.



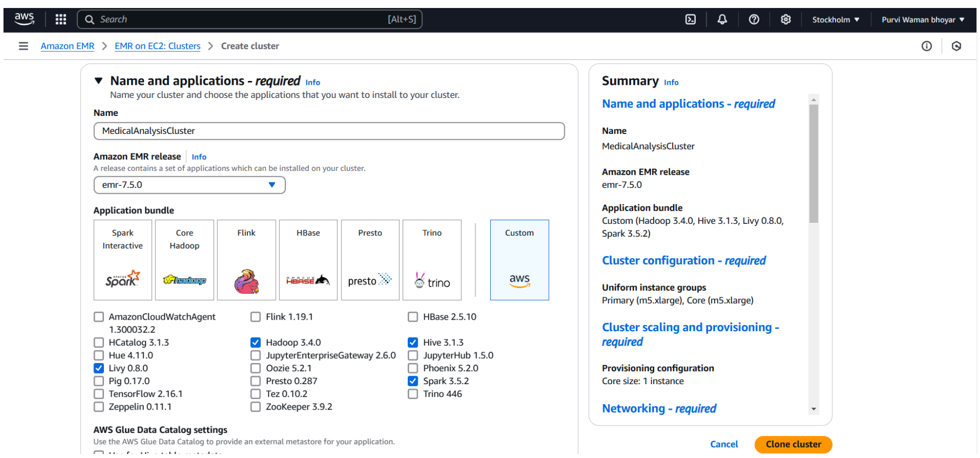
Using AWS VPC service, creating own VPC (Virtual Private Cloud) , Subnet, Routing table and Internet Gateway for creating EMR clusters further in the process .

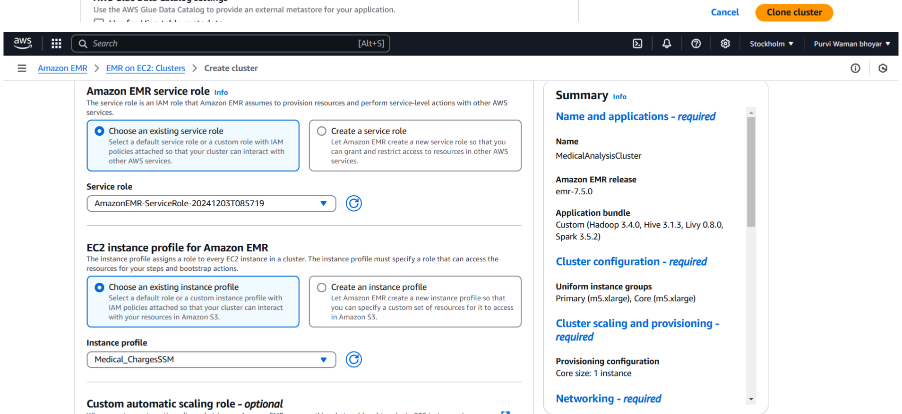


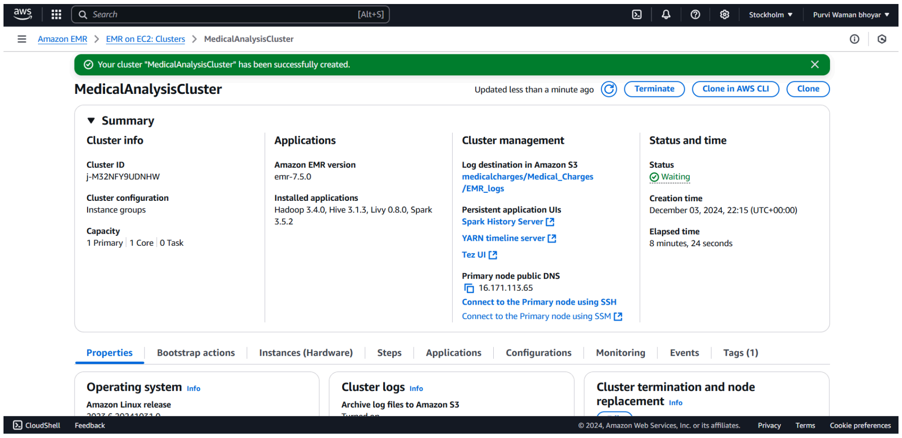
VPC->Subnet->Routing Table> Internet Gateway flow shown below



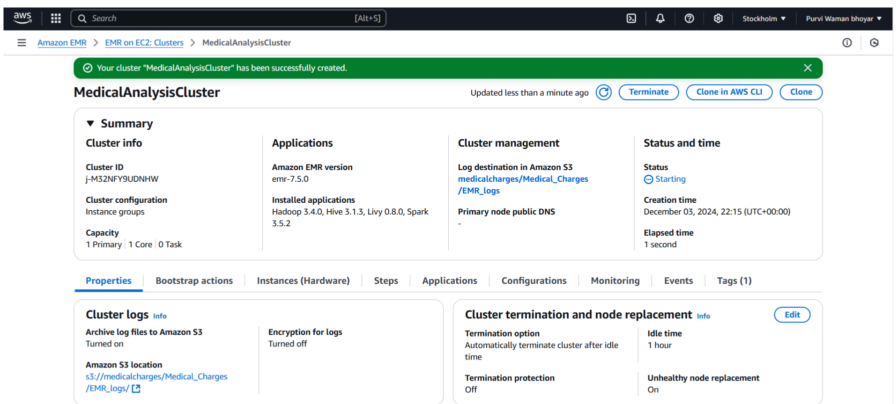
AWS EMR on EC2 cluster creation with necessary requirements and roles selected



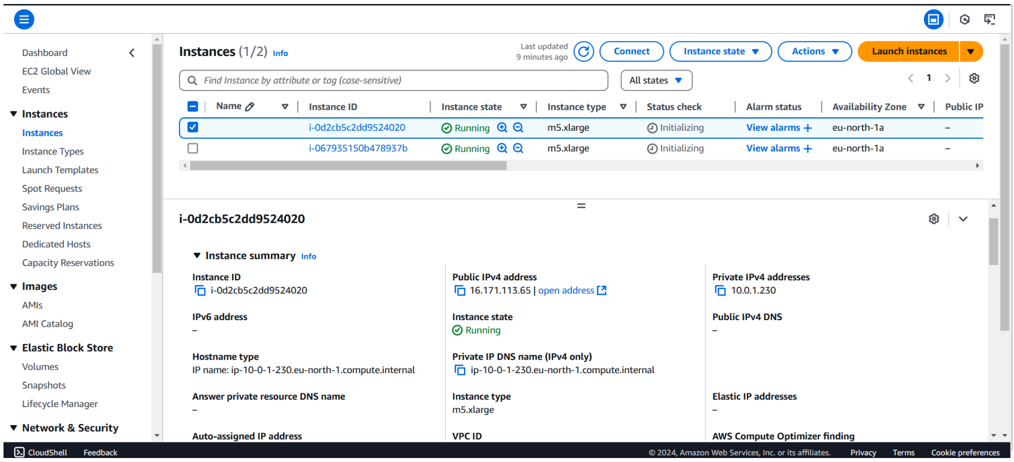




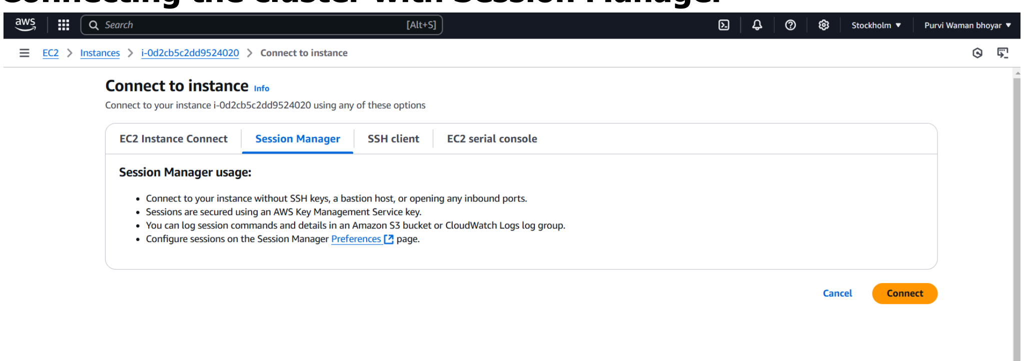
Successful EMR cluster created -



EC2 instance running simultaneously



Connecting the cluster with Session Manager

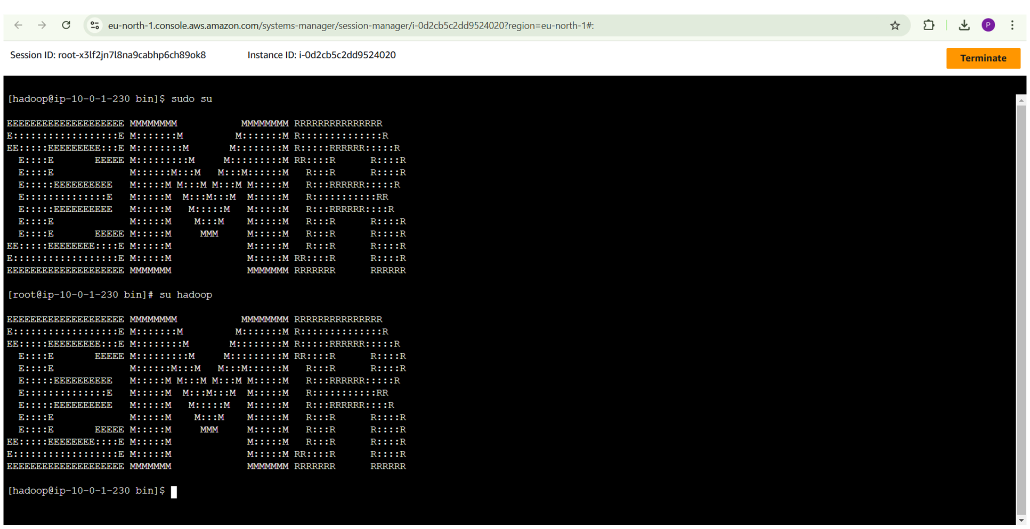


EMR Session Manager session starting, initial commands and outputs.

Steps:

1. sudo su

2. su Hadoop



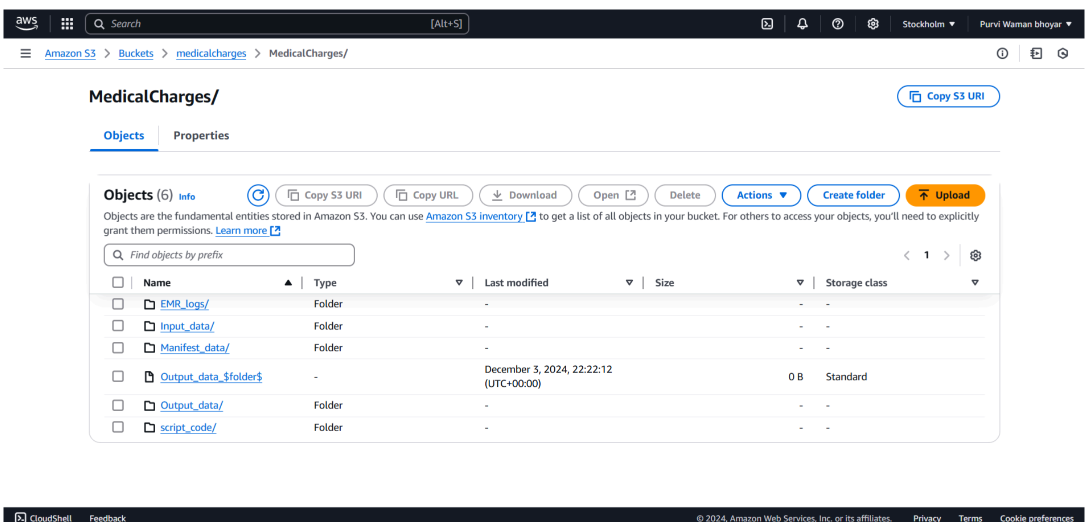
Making directory “Medicalproject” and copying the source_code.py into it .

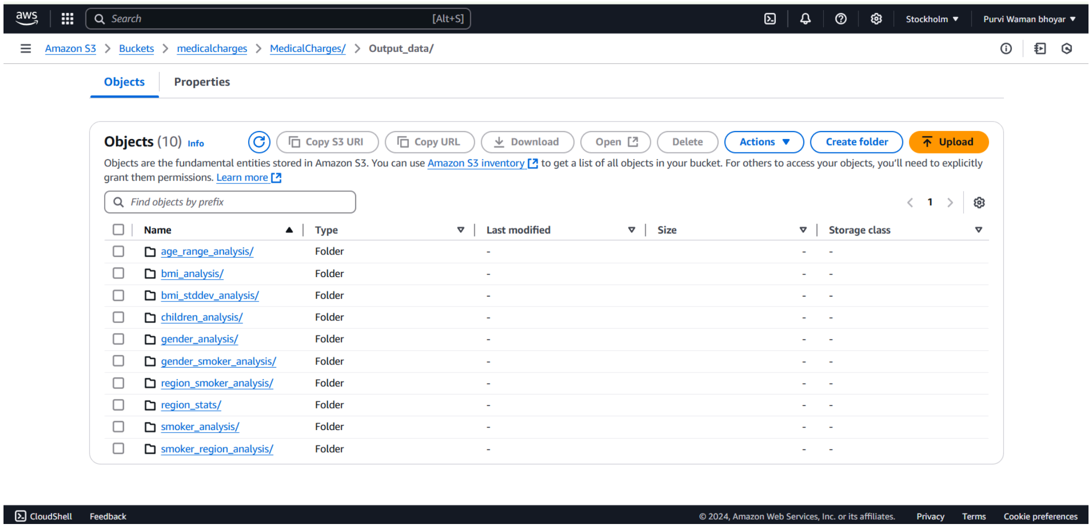
```
[hadoop@ip-10-0-1-230 bin]$ cd
[hadoop@ip-10-0-1-230 ~]$ mkdir Medicalproject
[hadoop@ip-10-0-1-230 ~]$ cd Medicalproject
```

```
[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
```

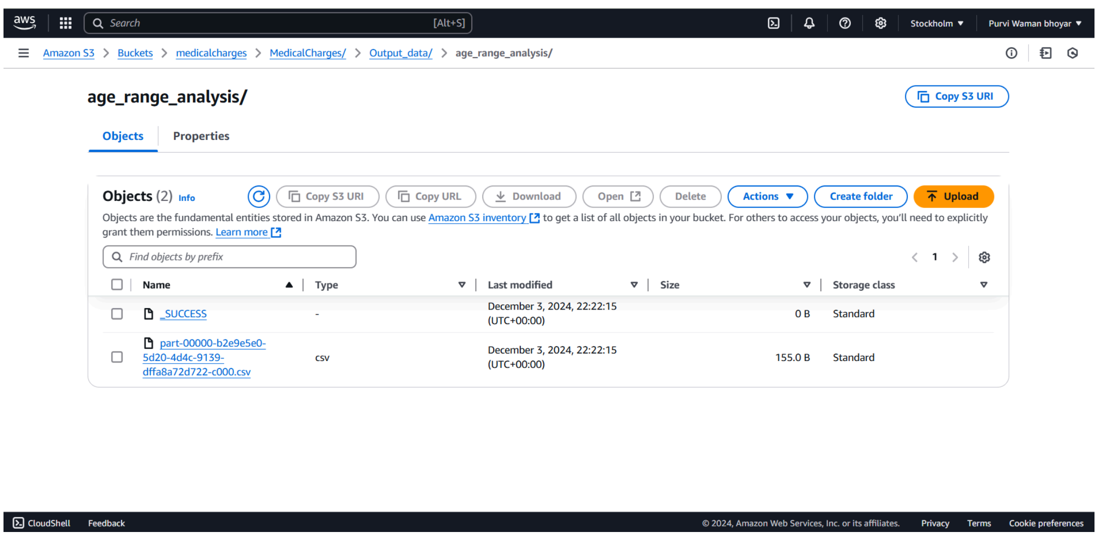
```
[hadoop@ip-10-0-1-230 Medicalproject]$ spark-submit source_code.py
24/12/03 22:20:59 INFO EMRParamSideChannel: Setting FGAC mode to false
24/12/03 22:20:59 INFO SparkContext: Running Spark version 3.5.2-amzn-1
24/12/03 22:20:59 INFO SparkContext: OS info Linux, 6.1.112-124.190.amzn2023.x86_64, amd64
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 .

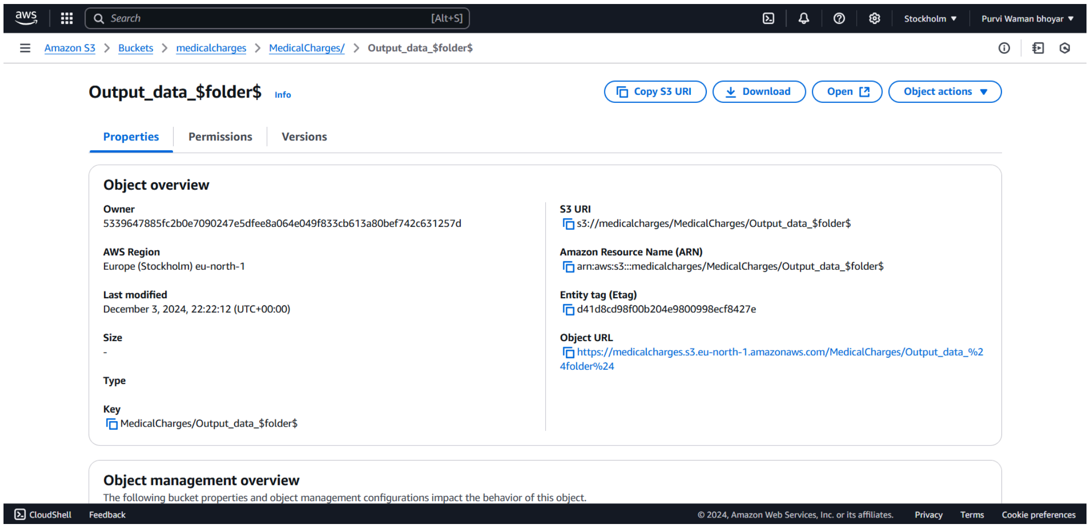


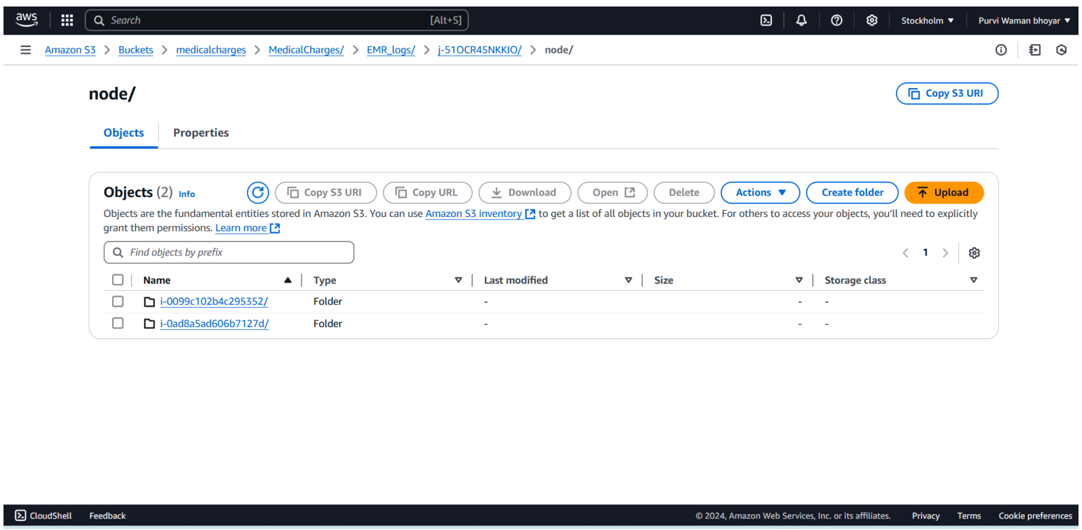


Snapshots of one of the output files created on the dedicated path.

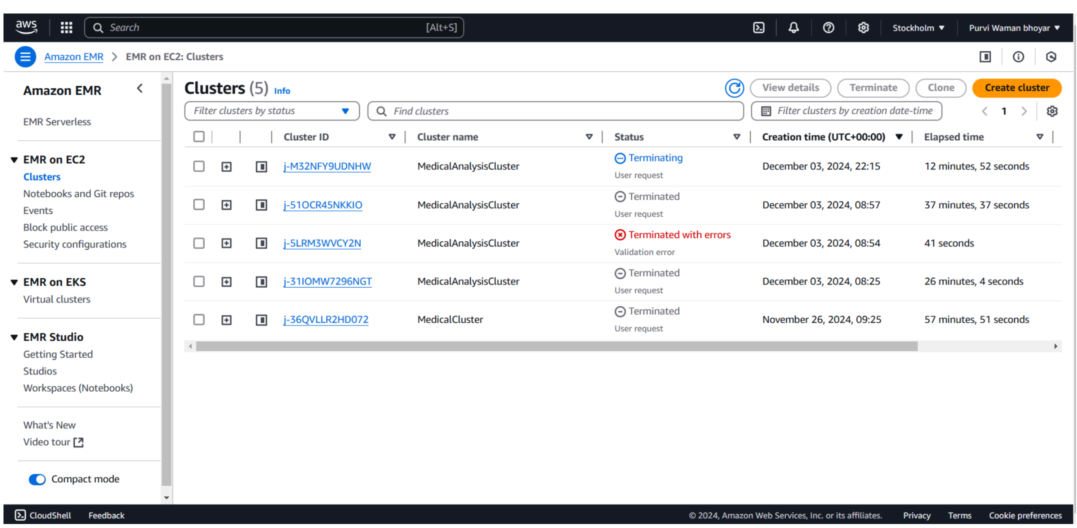


Snapshots showing output log file and EMR logs files getting stored in S3 bucket during the execution of EMR cluster.

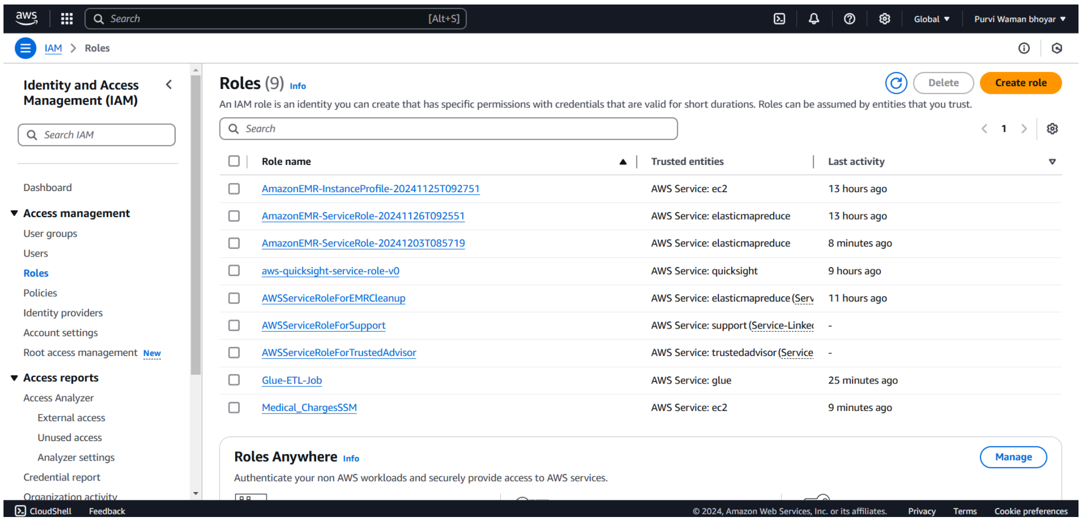




Below Snapshot is the list of all EMR on EC2 clusters created for the testing /implementation of the project.

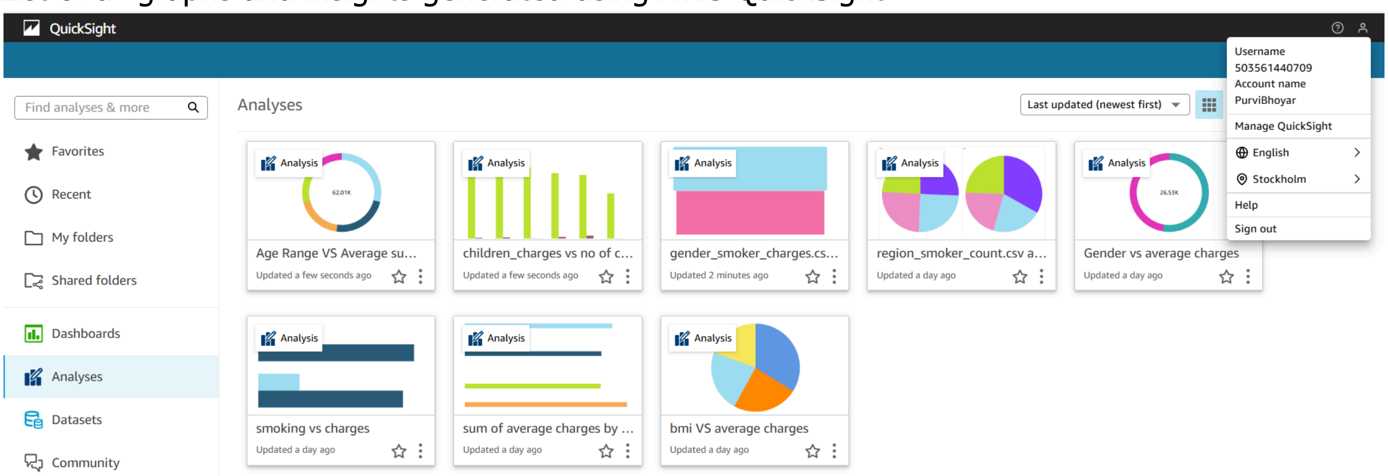


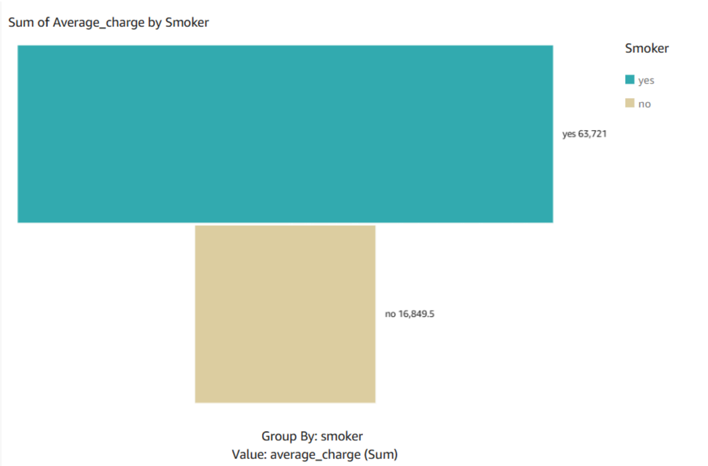
Below Snapshot is the list of all IAM roles created for the testing of scripts/Glue jobs/ EMR and S3 access specific clusters .

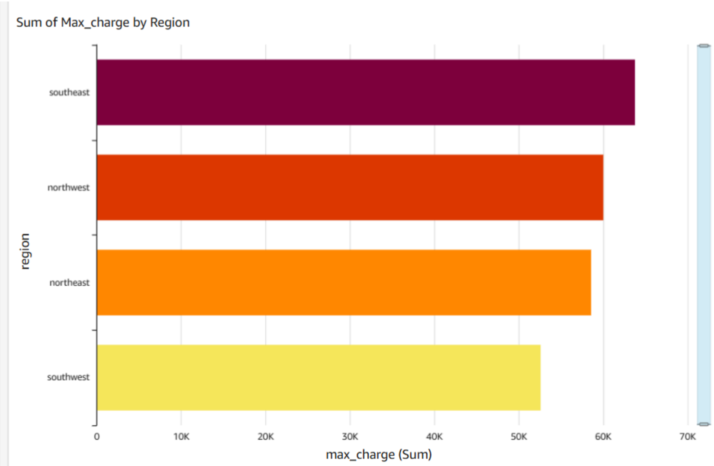


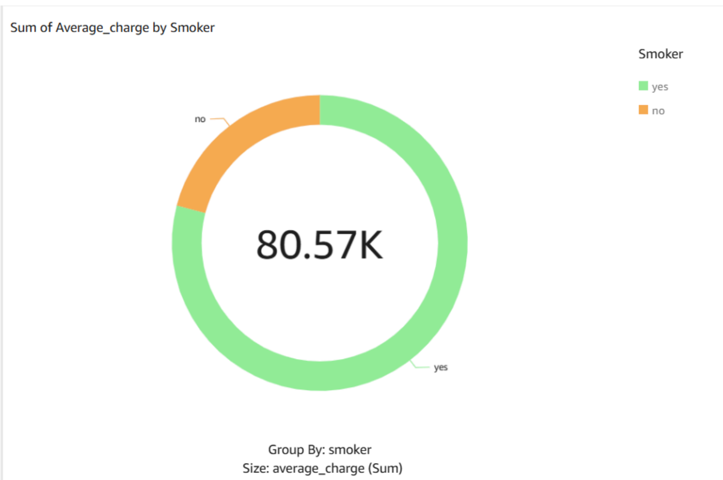
QuickSight Snapshots -

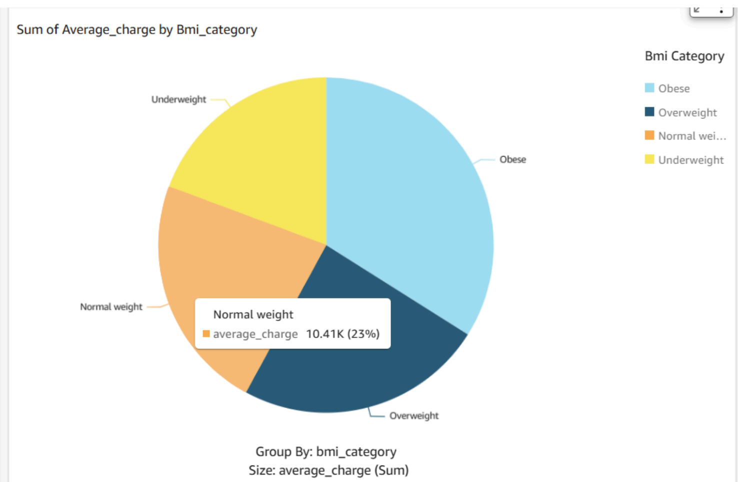
List of all graphs and insights generated using AWS QuickSight.

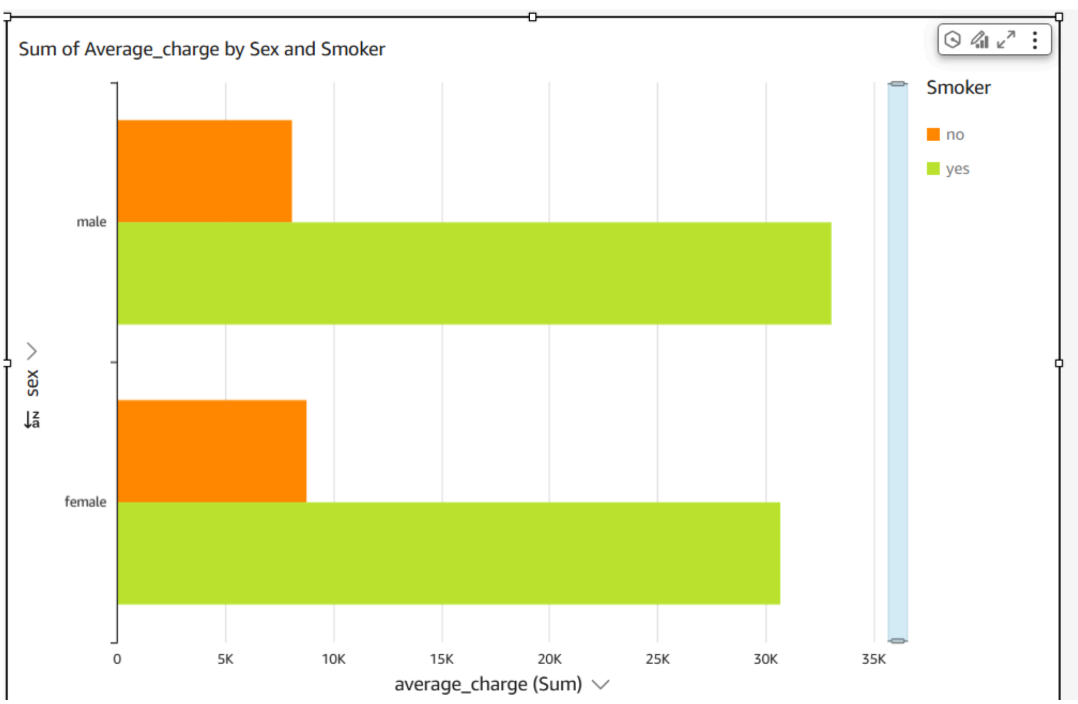


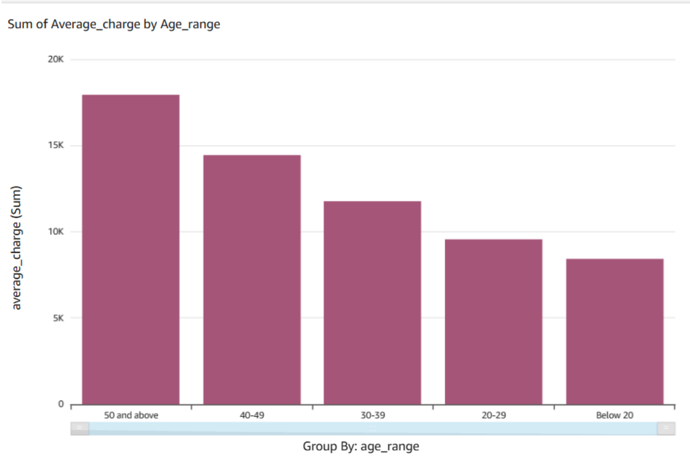

Sum of Average_charge by Smoker
Smoker
yes
no
yes 63,721
no 16,849.5
Group By: smoker
Value: average_charge (Sum)

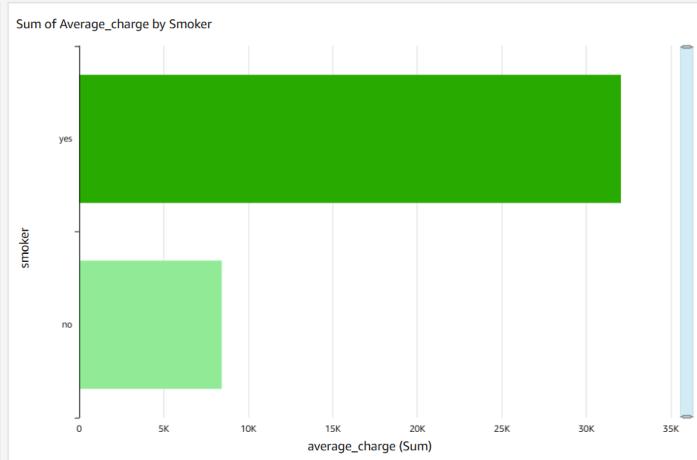

Sum of Max_charge by Region
region
southeast
northwest
northeast
southwest
0
10K
20K
30K
40K
50K
60K
70K
max_charge (Sum)

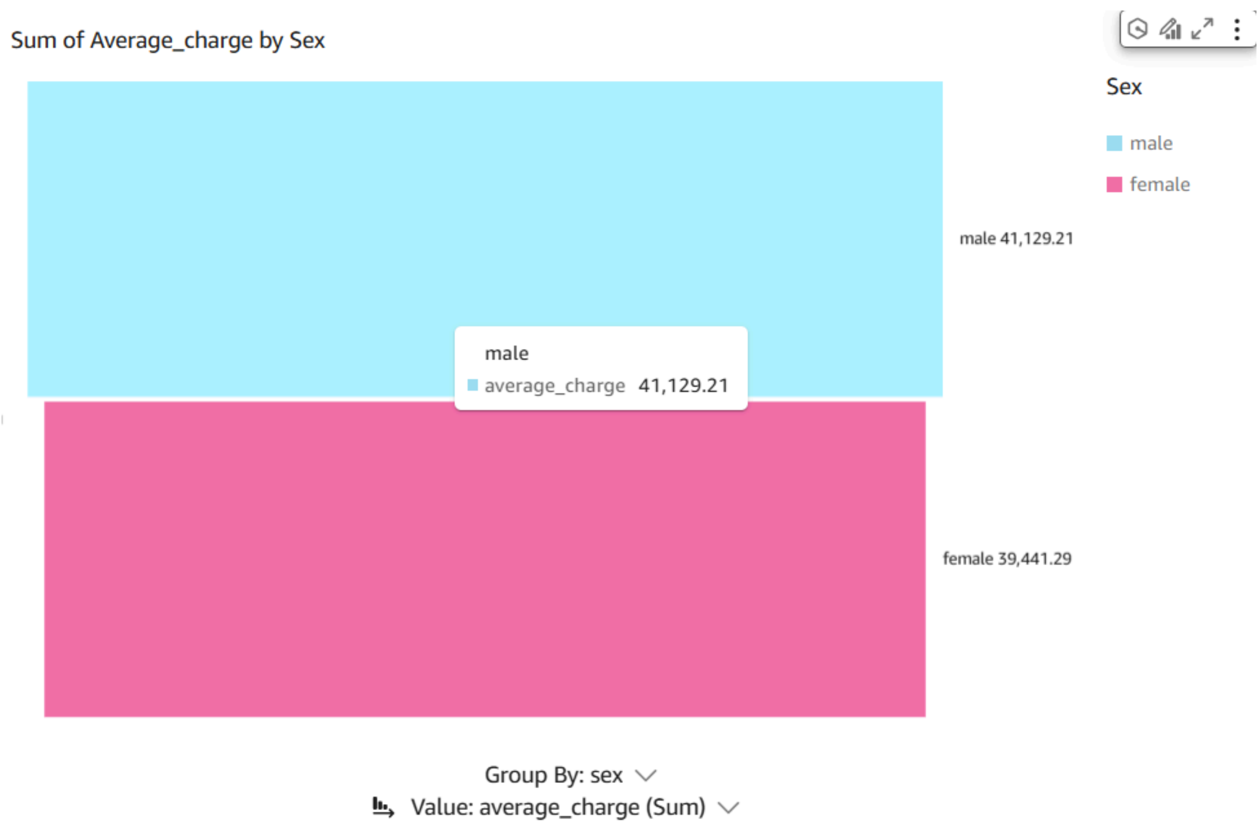

Sum of Average_charge by Smoker
Smoker
yes
no
no
80.57K
yes
Group By: smoker
Size: average_charge (Sum)

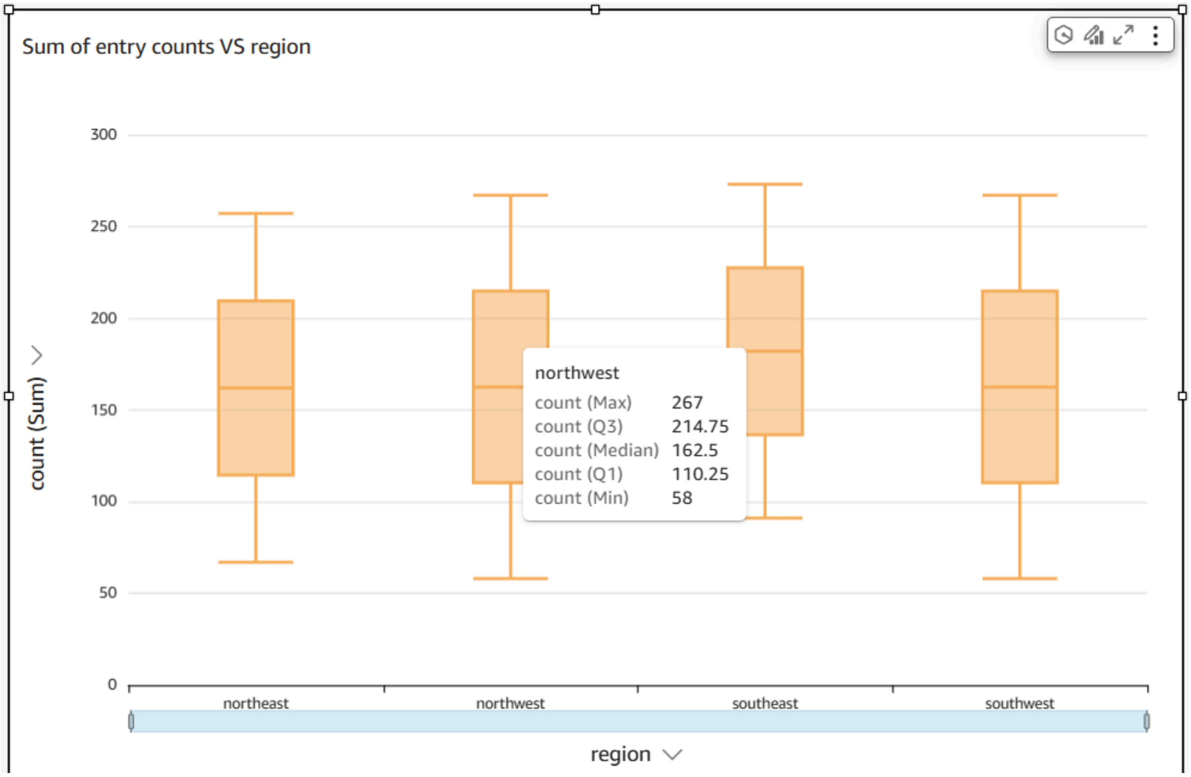

Sum of Average_charge by Bmi_category
Bmi Category
Obese
Overweight
Normal wei...
Underweight
Underweight
Obese
Normal weight
Normal weight
average_charge 10.41K (23%)
Overweight
Group By: bmi_category
Size: average_charge (Sum)


Sum of Average_charge by Sex and Smoker
Smoker
no
yes
male
female
sex
0
5K
10K
15K
20K
25K
30K
35K
average_charge (Sum)


Sum of Average_charge by Age_range
20K
15K
10K
5K
0
average_charge (Sum)
50 and above
40-49
30-39
20-29
Below 20
Group By: age_range


Sum of Average_charge by Smoker
yes
no
smoker
0
5K
10K
15K
20K
25K
30K
35K
average_charge (Sum)


Sum of Average_charge by Sex
Sex
male
female
male 41,129.21
male
average_charge 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.