Cloud Computing

Cloud Computing Project

Project Details

Student Name:	**Maximilian Girt**		
Student ID:	**22205093**		
Project Title:	**Data Warehouse Dashboard**	CC Project:	**1**
Due Date:	**04 / 12 / 2023**	Submitted Date:	**Mon Dec 4**

Instructions

1. **Project:** HealthCare Data Warehouse Analysis
2. **Project Report:** produce a project report describing the data set application being studied. This report should contain the following items:
   a. Application overview:
      1. The application aims to display data on a dashboard showing various analytics in the forms of charts and tables. It has the ability to switch between various data sets as well. The application focuses on Heart Disease, Stroke, and Diabetes and uses three separate datasets to achieve this. The reason for focusing on these three illnesses is that they are inherently linked as people at risk of one are also at risk of the other two.
      2. The application utilises the cloud via Amazon AWS's EC2 to host the website. Nginx is used to allow connections from outside the ec2. Tmux is used to keep the website running even when I am not SSH into the ec2.
      3. The data warehouse is hosted via Amazon AWS RedShift. RedShift has tables for each data set. The tables are saved in the default dev/public schema.
      4. RedShift retrieves the data from an Amazon S3 Bucket. This bucket is used to store the csv files from kaggle.

   b. **Objectives**: Briefly describe each goal/objective and whether they have been completed:
      1. **Develop a Data Dashboard for Healthcare Analytics:**
         1. **Goal**: To create an interactive dashboard capable of displaying analytics in the form of charts and tables, focusing on Heart Disease, Stroke, and Diabetes.
         2. **Status**: Completed. The dashboard has been successfully developed to display relevant data on these illnesses, providing insights through various visual formats.
      2. **Develop a Data Dashboard for Healthcare Analytics:**
         1. **Goal**: To design the dashboard with the capability to switch between multiple datasets, allowing for a comprehensive analysis of different health metrics.
         2. **Status**: Completed. The application now allows users to switch between different datasets for Heart Disease, Stroke, and Diabetes.
      3. **Utilise Cloud Infrastructure for Hosting**:

1. **Goal**: To use Amazon AWS's EC2 for hosting the website, ensuring reliable access and scalability.
2. **Status**: Completed. The application is hosted on AWS EC2, with Nginx for external connections and Tmux for continuous operation.

4. **Establish a Data Warehouse on Amazon AWS RedShift**:

    1. **Goal**: To set up a data warehouse on AWS RedShift for efficient data management, with separate tables for each health dataset.

    2. **Status**: Completed. The data warehouse is operational with data organized in the dev/public schema on RedShift.

5. **Integrate Data from Amazon S3 Bucket**:

    1. **Goal**: To retrieve and manage datasets stored as CSV files in an Amazon S3 Bucket, sourced from Kaggle.
    2. **Status**: Completed. The data warehouse successfully retrieves and manages data from the S3 Bucket, ensuring a seamless data pipeline.

c. Describe the problem and the collection of the datasets.

1. **Problem**:

    1. The primary problem addressed by the dashboard is the need for effective and accessible analytics in healthcare, specifically relating to chronic diseases such as Heart Disease, Stroke, and Diabetes. It is also paramount that patient data is secure which we can utilise the cloud to allow that. Ideally, this data can be used to better predict and understand what causes these illnesses and how they are linked.

    2. Healthcare professionals and researchers require an intuitive and dynamic platform to analyse various health metrics and trends from these datasets. However, the complexity and volume of healthcare data pose significant challenges in terms of data management, analysis, and visualisation. The Dashboard aims to address these challenges by providing an interactive, cloud-based solution for efficient data handling and visualisation, thereby enhancing decision-making and research capabilities in the medical field.

2. **Source of DataSets**:

    1. The datasets are all collected from Kaggle.

        a. https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset

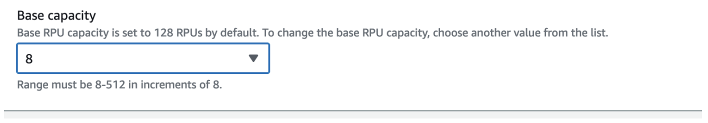
        b. https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

        c. https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

    2. It was paramount that the datasets had a high suitability score and were not too big in storage size as I did not want to go over the free tier of AWS for the EC2 or RedShift.

    3. Once the datasets are downloaded on my local machine as CSV files I uploaded them to the Amazon S3 Data Bucket which supports direct uploads of CSV files. Then, in the redshift query editor I could create tables and use the COPY command to copy the data from a CSV file into a table I have already created.

d. Explain your methodology and implementation.

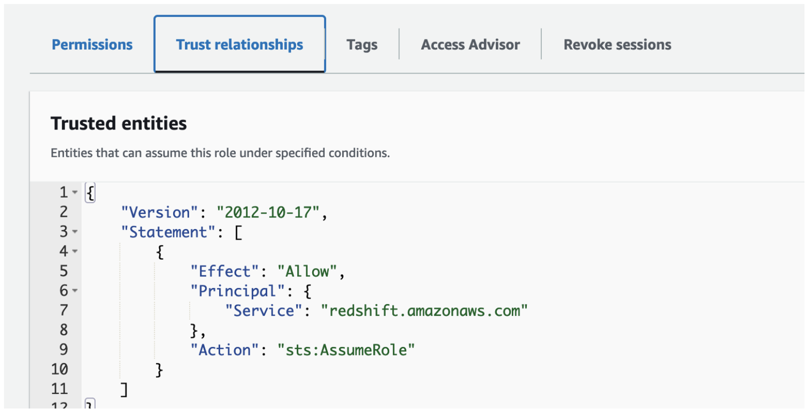
1. Cloud Infrastructure Planning
    1. The cloud infrastructure using Amazon AWS services was planned to ensure scalability, reliability, and security. This included choosing AWS EC2 for hosting, AWS RedShift for data warehousing, and AWS S3 for data storage.
    2. First I created an Amazon RedShift instance which required me to create a workgroup. I chose the lowest baes capacity used to process data warehouse workloads which is 8 RPUs or Redshift processing units. The reason I did this was to lower any potential costs as the less cloud resources I used the less money I would be charged!
        a. 

Base capacity

Base RPU capacity is set to 128 RPUs by default. To change the base RPU capacity, choose another value from the list.

8

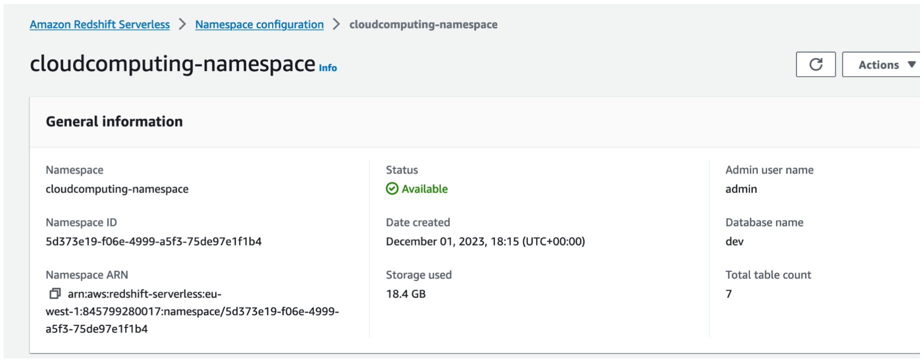
Range must be 8-512 in increments of 8.

    3. Next I had to create a virtual private cloud and a new security group. I stuck with the default for these which allowed me to add three subnet groups.
    4. Next I created my namespace which allowed me to choose a database name. I went with dev for the name which is now where my data is stored.
    5. I had to create an Associated IAM Role. For the purpose of my dashboard I created a role that would only be allowed to read from RedShift. Additionally, I had to create a trust relationship to make this work.
        a. 

Trusted entities

Entities that can assume this role under specified conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

        b. AmazonS3ReadOnlyAccess
    6. 

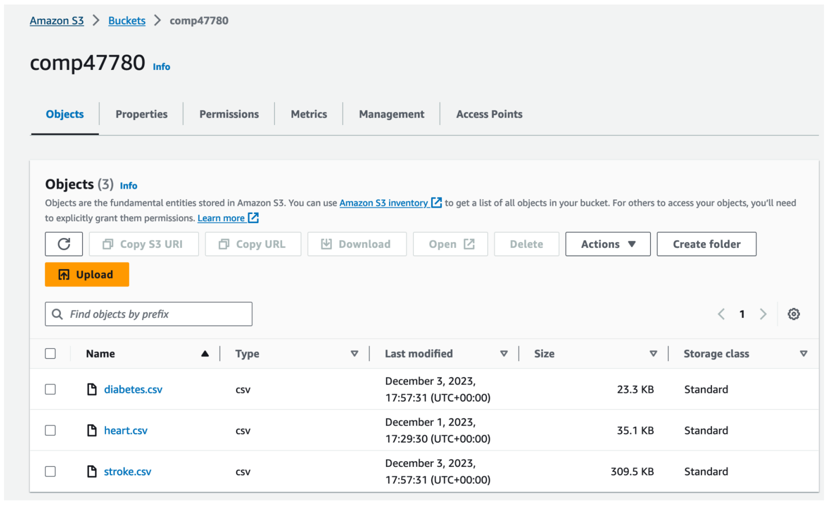
Amazon Redshift Serverless > Namespace configuration > cloudcomputing-namespace

cloudcomputing-namespace Info

General information

Namespace cloudcomputing-namespace	Status Available	Admin user name admin
Namespace ID 5d373e19-f06e-4999-a5f3-75de97e1f1b4	Date created December 01, 2023, 18:15 (UTC+00:00)	Database name dev
Namespace ARN arn:aws:redshift-serverless:eu-west-1:845799280017:namespace/5d373e19-f06e-4999-a5f3-75de97e1f1b4	Storage used 18.4 GB	Total table count 7

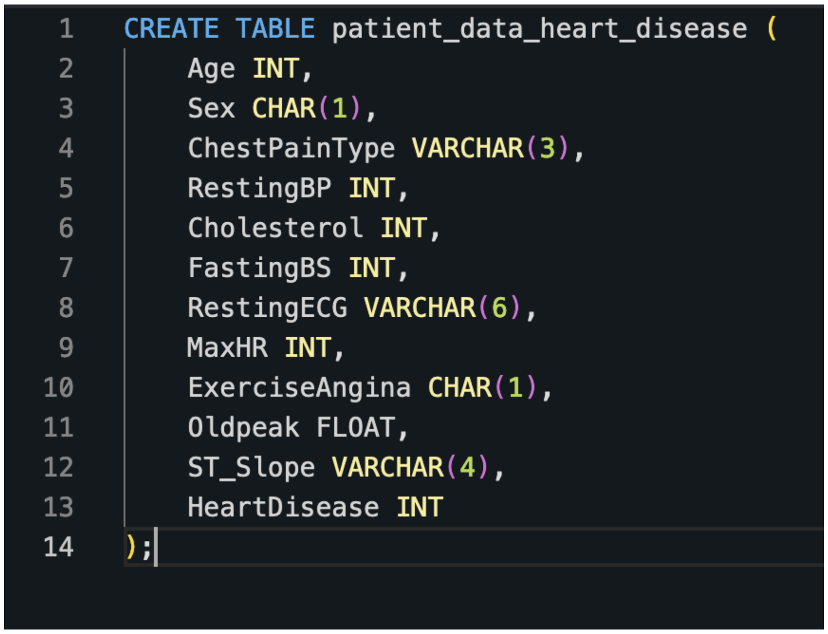
    7. Once My RedShift was created I needed to upload my data so I created an S3 Bucket. By creating the bucket in the same region as RedShift I can import the data directly into RedShift using its query editor!

a. 

Objects (3) Info

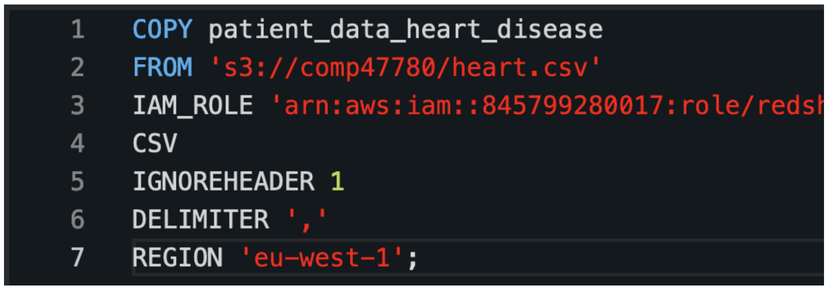
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
diabetes.csv	csv	December 3, 2023, 17:57:31 (UTC+00:00)	23.3 KB	Standard
heart.csv	csv	December 1, 2023, 17:29:30 (UTC+00:00)	35.1 KB	Standard
stroke.csv	csv	December 3, 2023, 17:57:31 (UTC+00:00)	309.5 KB	Standard

b. 

```
CREATE TABLE patient_data_heart_disease (
    Age INT,
    Sex CHAR(1),
    ChestPainType VARCHAR(3),
    RestingBP INT,
    Cholesterol INT,
    FastingBS INT,
    RestingECG VARCHAR(6),
    MaxHR INT,
    ExerciseAngina CHAR(1),
    Oldpeak FLOAT,
    ST_Slope VARCHAR(4),
    HeartDisease INT
);
```

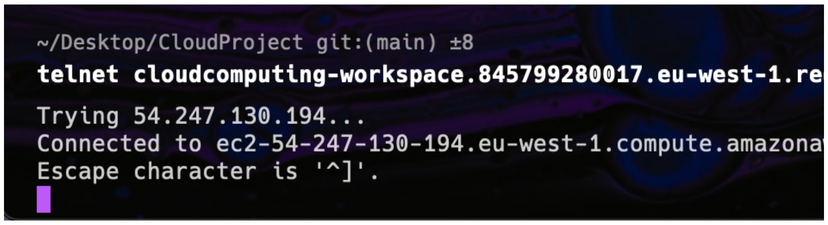
i. This is how I created my heart table

c. 

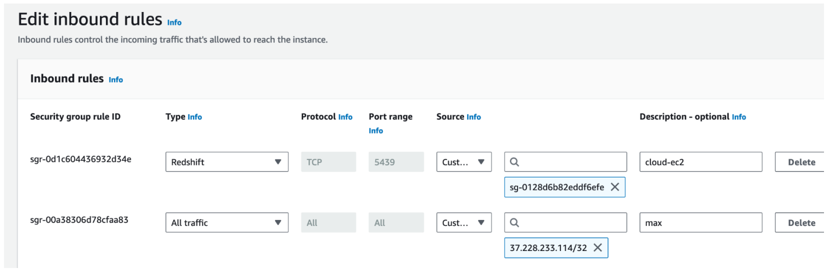
```
COPY patient_data_heart_disease
FROM 's3://comp47780/heart.csv'
IAM_ROLE 'arn:aws:iam::845799280017:role/redsh
CSV
IGNOREHEADER 1
DELIMITER ','
REGION 'eu-west-1';
```

i. This is how I copied the data from my S3 Bucket into my table in redshift.

8. Once I had the data set up I had to check that my data was accessible outside of the application. I used telnet on my local machine in the terminal to ping the RedShift Server via its ip. This only worked after I allowed public access and set up the security rules.

a. 

```
~/Desktop/CloudProject git:(main) ±8
telnet cloudcomputing-workspace.845799280017.eu-west-1.re
Trying 54.247.130.194...
Connected to ec2-54-247-130-194.eu-west-1.compute.amazona
Escape character is '^]'.
```

b. 

Edit inbound rules Info

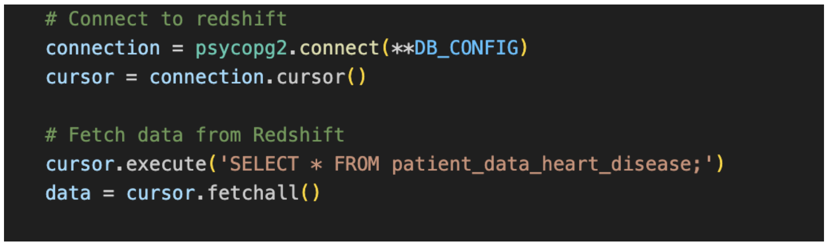
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-0d1c604436932d34e	Redshift	TCP	5439	Cust...	sg-0128d6b82eddf6efe	cloud-ec2	Delete
sgr-00a38306d78cfaa83	All traffic	All	All	Cust...	37.228.233.114/32	max	Delete

i. As seen in the picture I allow my personal IP to ping the RedShift server. However, I also have my EC2's security group set up so the EC2 can access redshift.

9. The next step was to build my dashboard. I have a lot of experience using React and Flask so those were my tools of choice. The back end used Flask which allowed me to connect to my Redshift Server and query data. I wrote sql queries that returned all the data in JSON format for each table. This data would then be taken to the front end react app to display it.

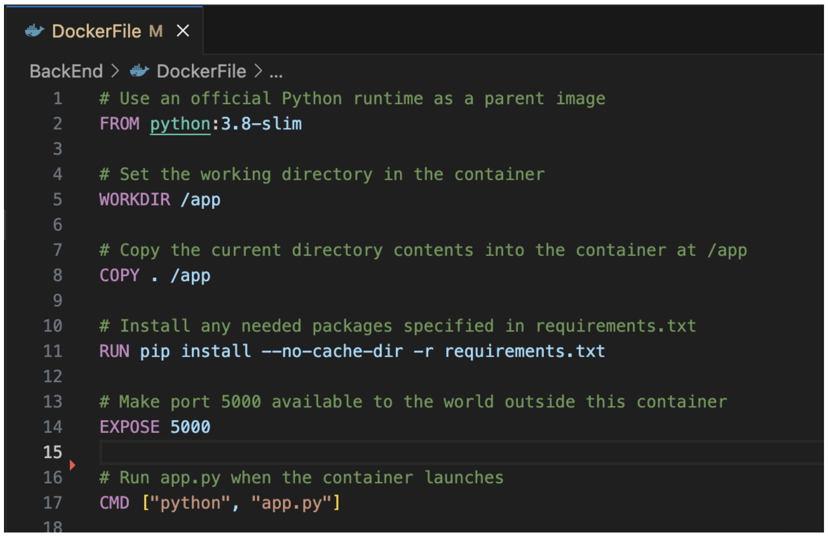
a. 

```python
# Connect to redshift
connection = psycopg2.connect(**DB_CONFIG)
cursor = connection.cursor()

# Fetch data from Redshift
cursor.execute('SELECT * FROM patient_data_heart_disease;')
data = cursor.fetchall()
```

10. To build the front-end I used React and the Chart.js library to visualise my data. I built different components for all aspects of my site. To get the data from the backend I used axios. To have that data accessible to my other components I had to use a context in react which allows me to pass data between different components in the app but also keeps different aspects of the app separate for better coding practices.

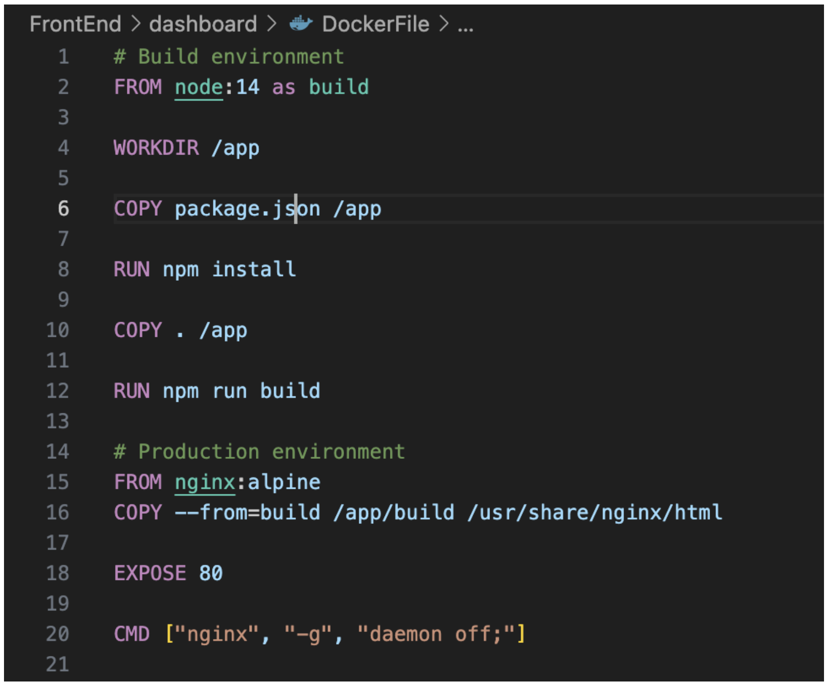
11. The Website is reactive so the user can change between the different data sets and they will immediately see the charts updating with the new data.

12. The last step was to deploy the website. To do this I decided to use docker for easier deployment. I created two DockerFiles, one for the front-end and one for the backend. I then used dockercompose to create a container.



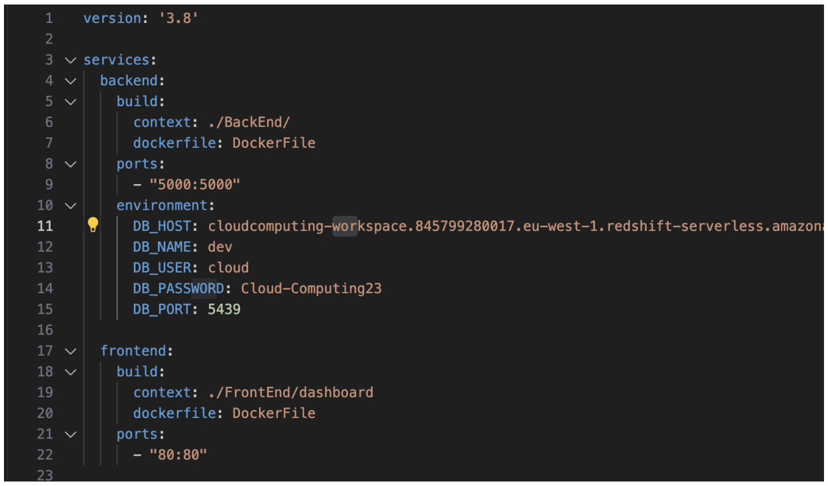
```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Run app.py when the container launches
CMD ["python", "app.py"]
```

a.



```
# Build environment
FROM node:14 as build

WORKDIR /app

COPY package.json /app

RUN npm install

COPY . /app

RUN npm run build

# Production environment
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

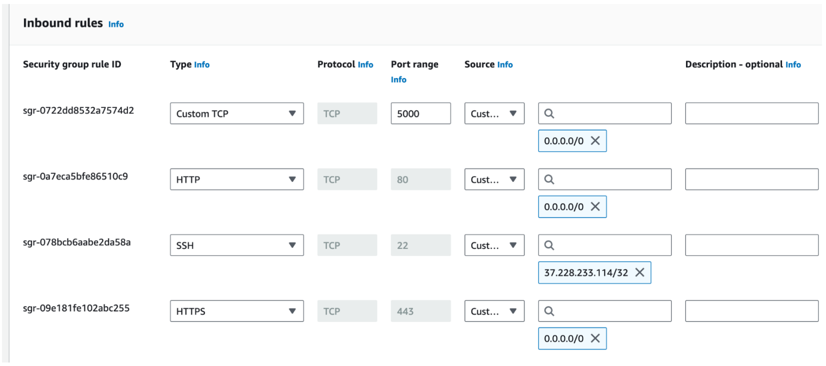
b.



```
version: '3.8'

services:
  backend:
    build:
      context: ./BackEnd/
      dockerfile: DockerFile
    ports:
      - "5000:5000"
    environment:
      DB_HOST: cloudcomputing-workspace.845799280017.eu-west-1.redshift-serverless.amazona
      DB_NAME: dev
      DB_USER: cloud
      DB_PASSWORD: Cloud-Computing23
      DB_PORT: 5439

  frontend:
    build:
      context: ./FrontEnd/dashboard
      dockerfile: DockerFile
    ports:
      - "80:80"
```

c.

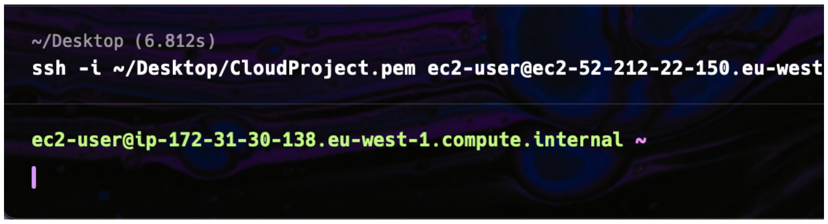
            i. I used an environment variable to secure my DB information.

13. The last step was to create an EC2 instance and host my website on it. I first created an Amazon Linux 2023 AMI Instance.
14. I created a new security group that was just for my EC2, I did not want to use the same one as RedShift as they serve different purposes.
    a. 

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info
sgr-0722dd8532a7574d2	Custom TCP	TCP	5000	Cust...	0.0.0.0/0	
sgr-0a7eca5bfe86510c9	HTTP	TCP	80	Cust...	0.0.0.0/0	
sgr-078bcb6aabe2da58a	SSH	TCP	22	Cust...	37.228.233.114/32	
sgr-09e181fe102abc255	HTTPS	TCP	443	Cust...	0.0.0.0/0	

        i. Currently, the front-end and back-end can be accessed by anyone with the link to my ec2. This allows me to access the website from anywhere.

15. I then SHH into my EC2 on my local terminal. I used SCP to transfer my project over to the EC2:
    a. scp -i ~/Desktop/CloudProject.pem -r ~/Desktop/CloudProject.zip ec2-user@ec2-52-212-22-150.eu-west-1.compute.amazonaws.com: Project
    b. 

```
~/Desktop (6.812s)
ssh -i ~/Desktop/CloudProject.pem ec2-user@ec2-52-212-22-150.eu-west

ec2-user@ip-172-31-30-138.eu-west-1.compute.internal ~
```

16. Once in the EC2 I built my project just running the command:
    a. docker compose up.
17. I had to change the Axios get requests to use the link for the EC2 instead of just saying "localhost"
18. In order to have the site running even when I am not using the EC2 terminal I utilised a detached TMUX session to keep my website running.

e. Discuss the suitability of the tool being used to solve the problem.

   1. Amazon RedShift

      1. This was used as the data warehouse solution for storing and managing large datasets related to Heart Disease, Stroke, and Diabetes.
      2. This is highly suitable due to its ability to handle large volumes of data efficiently, perform fast query processing, and scale seamlessly. RedShift's columnar storage and data compression capabilities make it ideal for the kind of analytical workload required in healthcare data analysis. RedShift handles much of the maintenance, backup, and scaling automatically as well.

    3. It is really handy that RedShift uses SQL for all its queries as its easy to use and means most people can help build the application if I ever expand it.

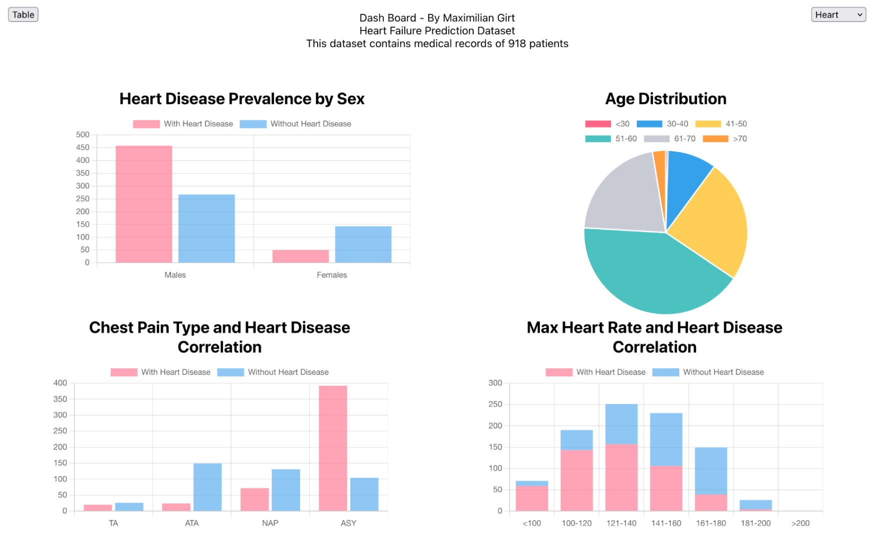
2. Amazon S3 Data Bucket
    1. This was used for storing CSV files from Kaggle, which are the primary data sources for the dashboard
    2. This was a great choice for the project's data storage given its high durability, scalability, and security. It effectively serves as a reliable and accessible data lake for the project.
3. Amazon AWS EC2
    1. I used this to host the web application of the dashboard.
    2. This was ideal as it integrated really well with Amazon Redshift which made it really easy to access the data. I only had to change a few lines of code for the axios get requests in order to get the data onto my web application. This was already shown in the methodology section.
4. NGINX
    1. I used this as a web server to manage external connections to the dashboard and was defined in my dockerfile!
    2. This tool provides a secure and efficient gateway into the application and protects against security attacks such as DDOS.
5. TMUX
    1. Used to ensure the dashboard is operating continuously. What good is a dashboard that isn't running when you need it?
    2. This allows me to keep the application running without manual intervention.
6. React / Chart.js
    1. This allowed me to create interactive and reactive charts on the dashboard that are visually appealing. This made the complex data easy to look at and understand.
7. Docker & Docker Compose
    1. I containerized the application to ensure consistency across different environments and simplifying deployment.

f. Describe the features of your software.

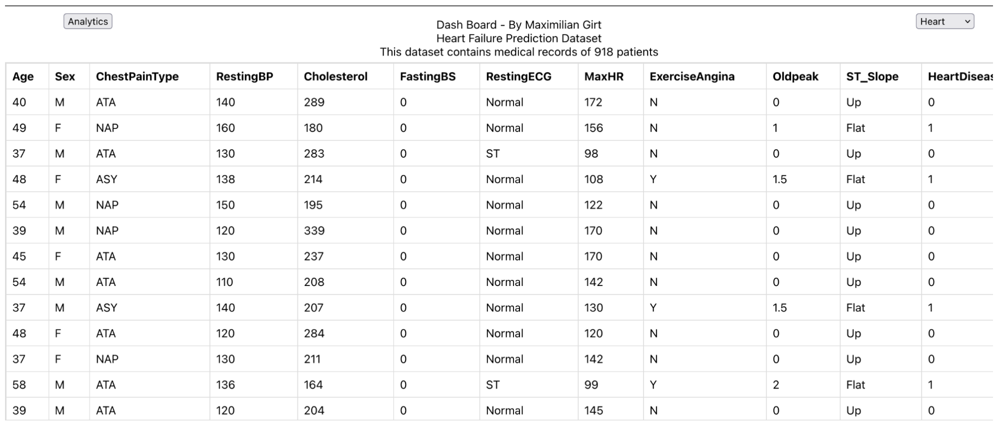
1. Interactive Data Visualization
    1. The dashboard provides dynamic and interactive charts, graphs, and tables. Users can visualize data related to Heart Disease, Stroke, and Diabetes in various formats, bar charts, pie charts, and histograms.
    2. Users are able to interact with the charts with their mouse to hide data points, or hover over them to see how many patients are affected.
2. Reactive Website
    1. Users can immediately switch between what data they are looking at with no lag. They can also switch between viewing the data visualisations or just looking at the table.
3. User Friendly Interface

4. Cloud Based Access
    1. As a cloud-based application, it is accessible from anywhere, at any time, provided there is internet connectivity. This enhances flexibility and convenience for users.
5. Scalable Architecture
    1. with the use of AWS Redshift, AWS S3 Buckets, and EC2, the software can handle increasing amounts of data and users without a significant drop in performance. This is crucial for adapting to growing data analysis needs.
6. Due to it being on the cloud, users with the proper access, (currently only me) can enter the AWS management system and upload more data.

g. Give a worked example.

1. http://ec2-52-212-22-150.eu-west-1.compute.amazonaws.com/
    1. Please let me know once this has been graded so I can take everything down.
    2. The video shows me accessing the website as well as everything I set up in AWS.
2. 

Dash Board - By Maximilian Girt
Heart Failure Prediction Dataset
This dataset contains medical records of 918 patients

3. 

Dash Board - By Maximilian Girt
Heart Failure Prediction Dataset
This dataset contains medical records of 918 patients

Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDiseas
40	M	ATA	140	289	0	Normal	172	N	0	Up	0
49	F	NAP	160	180	0	Normal	156	N	1	Flat	1
37	M	ATA	130	283	0	ST	98	N	0	Up	0
48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
54	M	NAP	150	195	0	Normal	122	N	0	Up	0
39	M	NAP	120	339	0	Normal	170	N	0	Up	0
45	F	ATA	130	237	0	Normal	170	N	0	Up	0
54	M	ATA	110	208	0	Normal	142	N	0	Up	0
37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
48	F	ATA	120	284	0	Normal	120	N	0	Up	0
37	F	NAP	130	211	0	Normal	142	N	0	Up	0
58	M	ATA	136	164	0	ST	99	Y	2	Flat	1
39	M	ATA	120	204	0	Normal	145	N	0	Up	0

h. Conclusion

1. The link between diabetes, stroke, and heart disease is crucial and complex. Diabetes, a condition characterised by high blood sugar levels, can damage blood

vessels and nerves throughout the body. This damage increases the risk of cardiovascular diseases, including heart disease, which remains one of the leading causes of death globally. Furthermore, diabetes elevates the risk of developing blood clots or the buildup of plaques in arteries, both of which are significant risk factors for stroke. Stroke and heart disease share common risk factors and can exacerbate each other, creating a cycle that can be challenging to break. Understanding these links is vital for effective prevention, management, and treatment strategies (https://www.niddk.nih.gov/health-information/diabetes/overview/preventing-problems/heart-disease-stroke).

2. The Dashboard and Data Warehouse Solution addresses the need to understand the links between Heart Disease, Stroke, and Diabetes by providing a way to see what causes these illnesses.

3. The use of Amazon RedShift, S3, and the EC2 is pivotal to the success of this project. Amazon RedShift plays a crucial role as the project's data warehouse, offering a robust, scalable, and efficient platform for storing and analysing the vast amounts of health data associated with diabetes, stroke, and heart disease. Its fast, powerful query processing capabilities enable the handling of complex analytical workloads with ease, providing the necessary speed and efficiency for real-time data analysis. This is essential for the dashboard's objective of delivering timely and accurate healthcare insights. Amazon EC2 serves as the backbone of the application hosting environment. It provides the necessary computing power and scalability to ensure that the Dashboard is consistently available and performs optimally, regardless of user load. EC2's flexibility in resource allocation and its ability to scale on demand are fundamental in managing the varying traffic and data processing needs of the dashboard

i. References

1. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html
2. https://docs.aws.amazon.com/redshift/latest/gsg/new-user-serverless.html
3. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
4. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
5. https://flask.palletsprojects.com/en/3.0.x/
6. https://www.chartjs.org/docs/latest/
7. https://legacy.reactjs.org/docs/getting-started.html
8. https://www.niddk.nih.gov/health-information/diabetes/overview/preventing-problems/heart-disease-stroke
9. https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset
10. https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction
11. https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
12. https://stackoverflow.com/questions/63708035/installing-docker-compose-on-amazon-ec2-linux-2-9kb-docker-compose-file

3. **Submission:** The deadline is 04 December 2023. Passed this deadline penalty will apply.

   a. Please submit your project report, the source code, and all the necessary files to execute your implementation.

b. You may need to prepare a README file to explain how to execute it.
c. Please submit ONLY one ZIP file, containing all the necessary files and directories, including the project report.