Project Deliverable-2

Group Members:
Aneela Gannarapu
Shravan Naidu Gollu
Indu Narsingula
Manogna Chennuru
Mounika kasaragadda

A phase wise approach is followed to achieve the preprocessing of data that includes ETL transformations and later we performed few visualizations.
Phase-1: S3 Set up

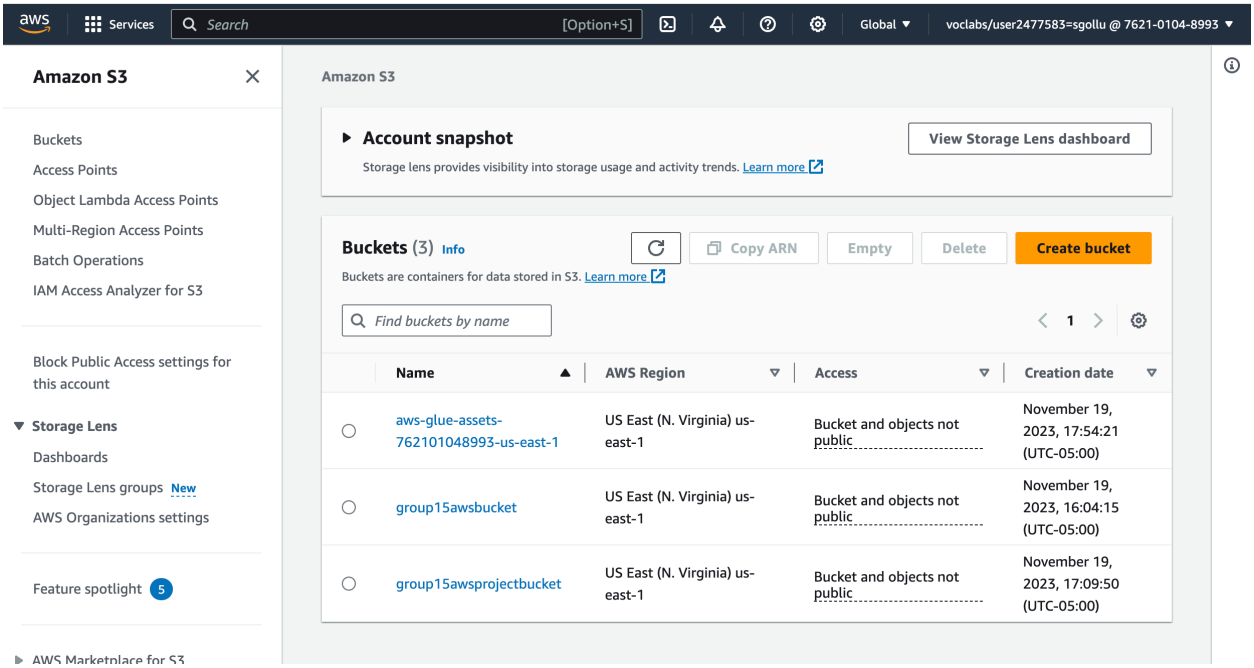


Fig: S3 buckets creation to import the dataset from kaggle

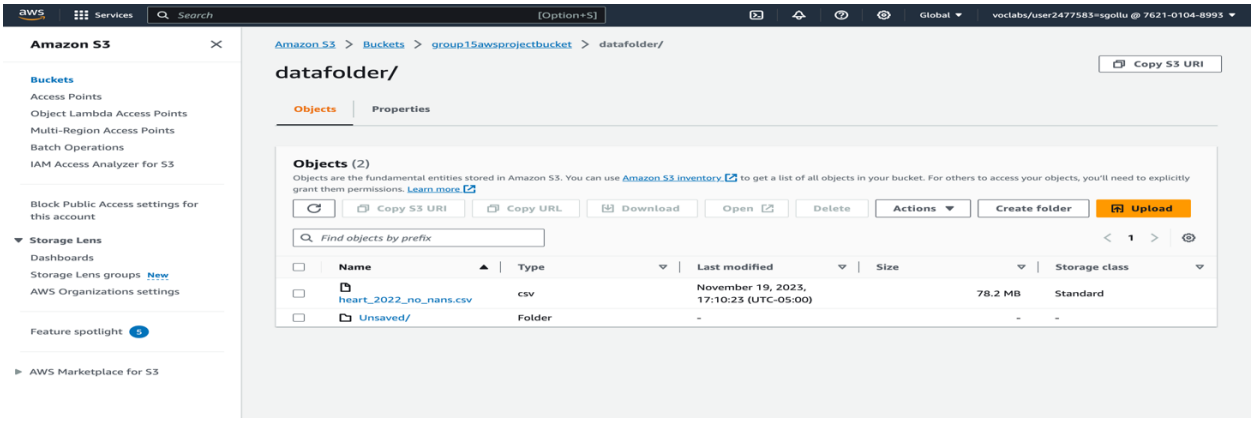


Fig: Uploaded the dataset file to bucket

After the data is placed on to S3 bucket, Phase-2&3 is started where we first created Query Result location Athena and later utilized glue to create new schema for the database. Visualise ETL feature of Glue is utilized to clean the data set. Then a crawler is created to establish new table in the dataset. Later multiple SQL queries were written to better understand the data distribution.

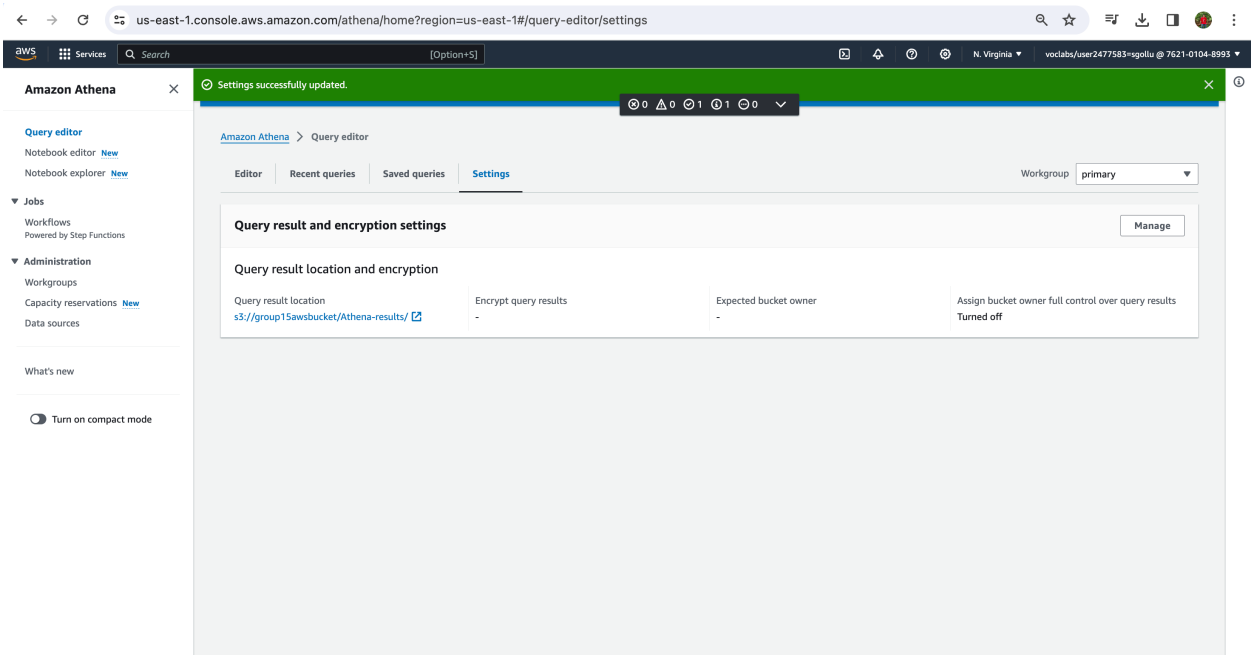


Fig : Query Result location setup for SQL Queries executed in Athena

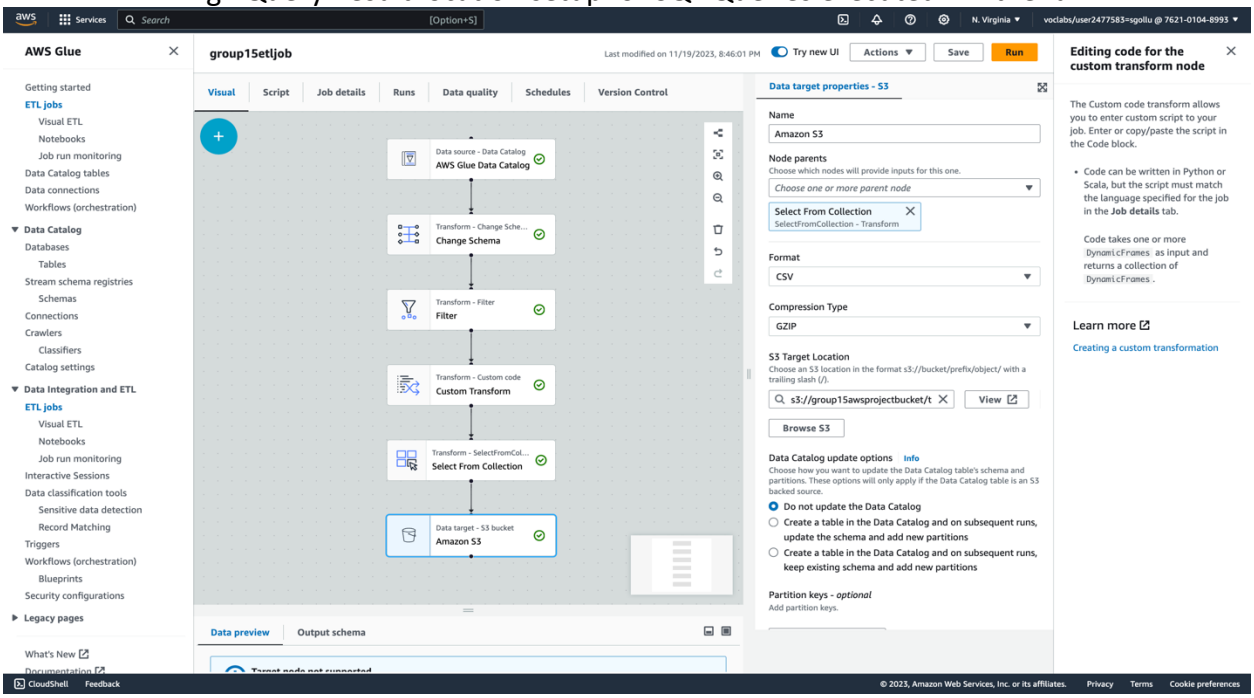


Fig: Steps involved in ETL

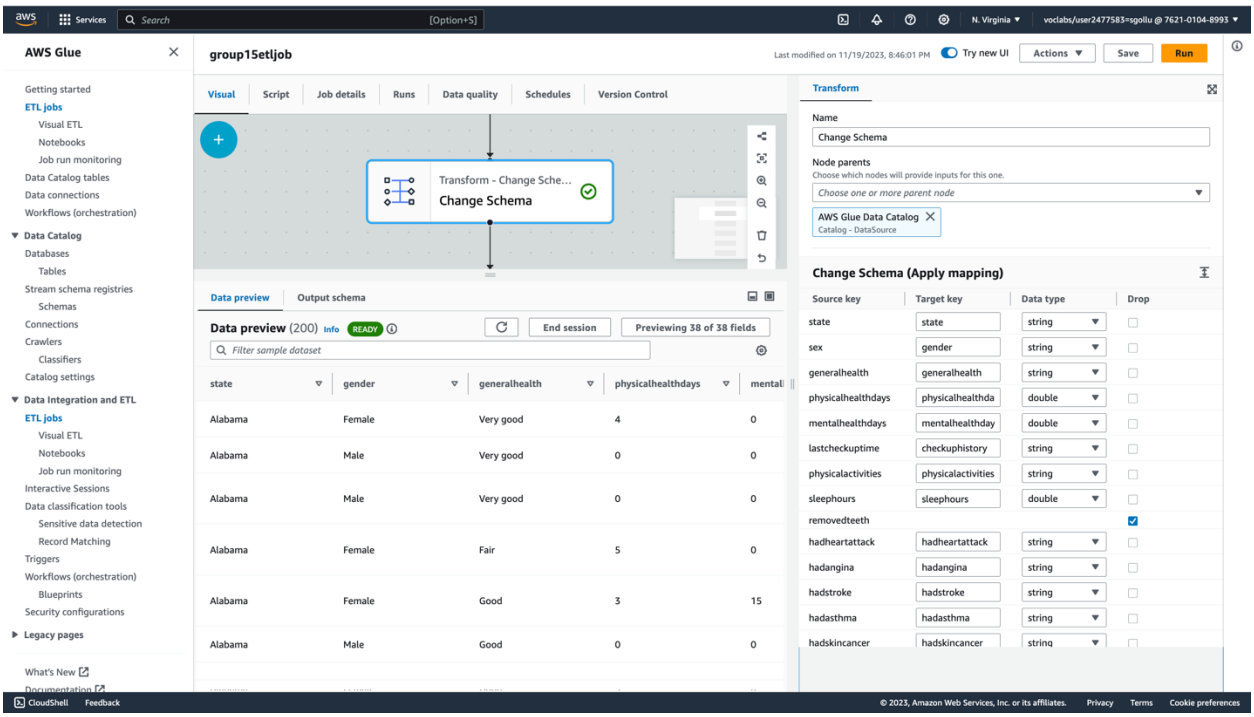


Fig: The transform Change Schema is used to perform Update and delete operations on database

Operations performed: Dropped Covid Test result and Removed Teeth columns. First one indicates whether the patient is tested positive or negative and the later indicates number of teeth removed till now. Both are insignificant when focusing on heart disease prediction. Renamed few columns. To make it easier for us code and to increase readability.

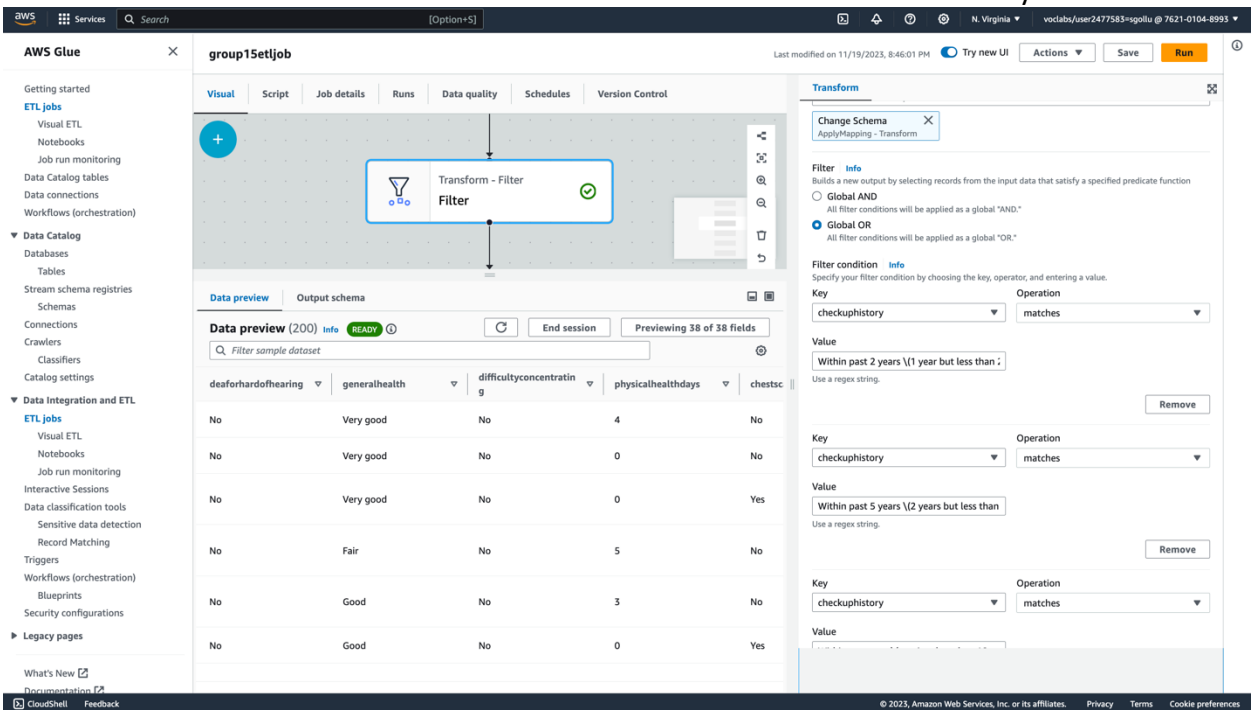


Fig: Screenshot indicates the filter that has been implemented on Checkup history of patients.

Reason to establish the filter: The dataset contains results of patient checkup history range starting from less than year to more than five years.
Estimating one's health based on their history dated more than five years will not give feasible solutions hence filtered out the attribute data.

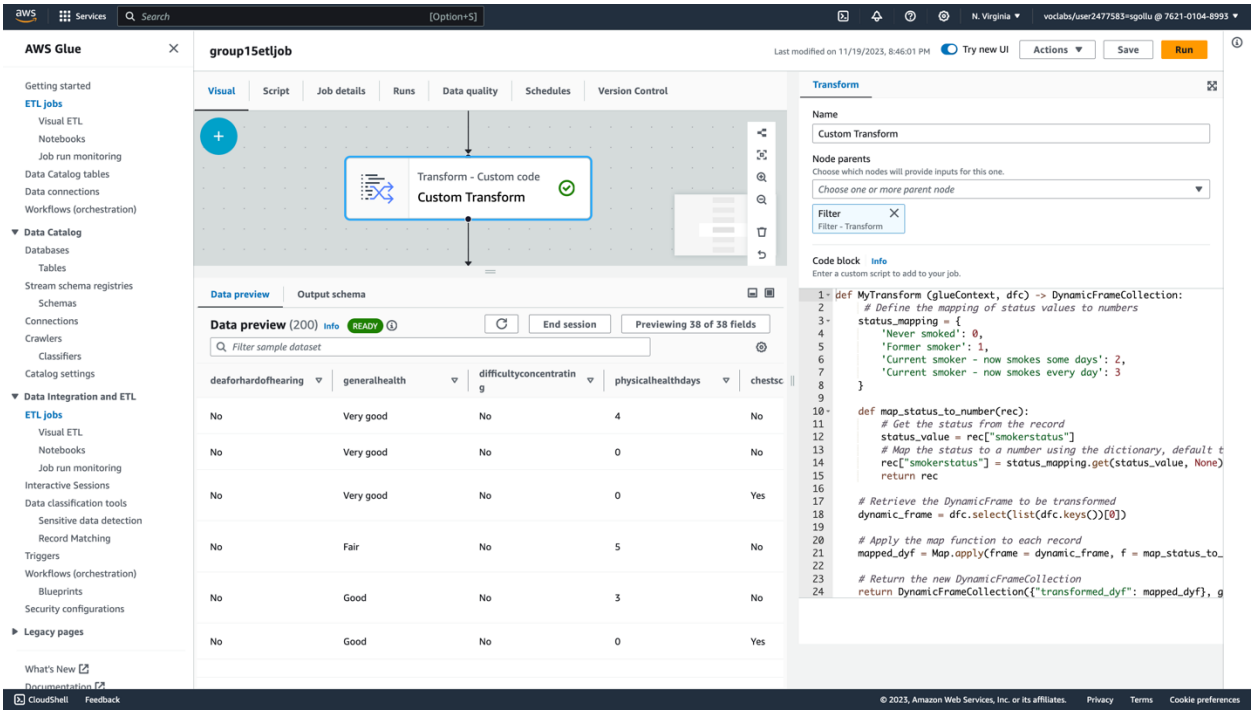


Fig : Screenshot details of custom transform applied to the dataset

Applied On: Smokerstatus column.
Reason: Smoking is one the key factors when talking about the causes of heart disease. But the dataset has categorical values for the attribute (Smokerstatus) and the values describe the frequency of smoking. Hence, we decided to convert the string values to numeric values by writing a python script to meet the necessary requirements. The script is as follows:

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    # Define the mapping of status values to numbers
    status_mapping = {
        'Never smoked': 0,
        'Former smoker': 1,
        'Current smoker - now smokes some days': 2,
        'Current smoker - now smokes every day': 3
    }

    def map_status_to_number(rec):
        # Get the status from the record
        status_value = rec["smokerstatus"]
```

```
    # Map the status to a number using the dictionary, default to None if not found
    rec["smokerstatus"] = status_mapping.get(status_value, None)
    return rec

# Retrieve the DynamicFrame to be transformed
dynamic_frame = dfc.select(list(dfc.keys())[0])

# Apply the map function to each record
mapped_dyf = Map.apply(frame = dynamic_frame, f = map_status_to_number)

# Return the new DynamicFrameCollection
return DynamicFrameCollection({"transformed_dyf": mapped_dyf}, glueContext)
```

Convention employed for conversion:

```
'Never smoked': 0,
'Former smoker': 1,
'Current smoker - now smokes some days': 2,
'Current smoker - now smokes every day': 3
```

All the transformed data is stored as a csv file in a new folder of S3. After populating the S3 details the job is executed.

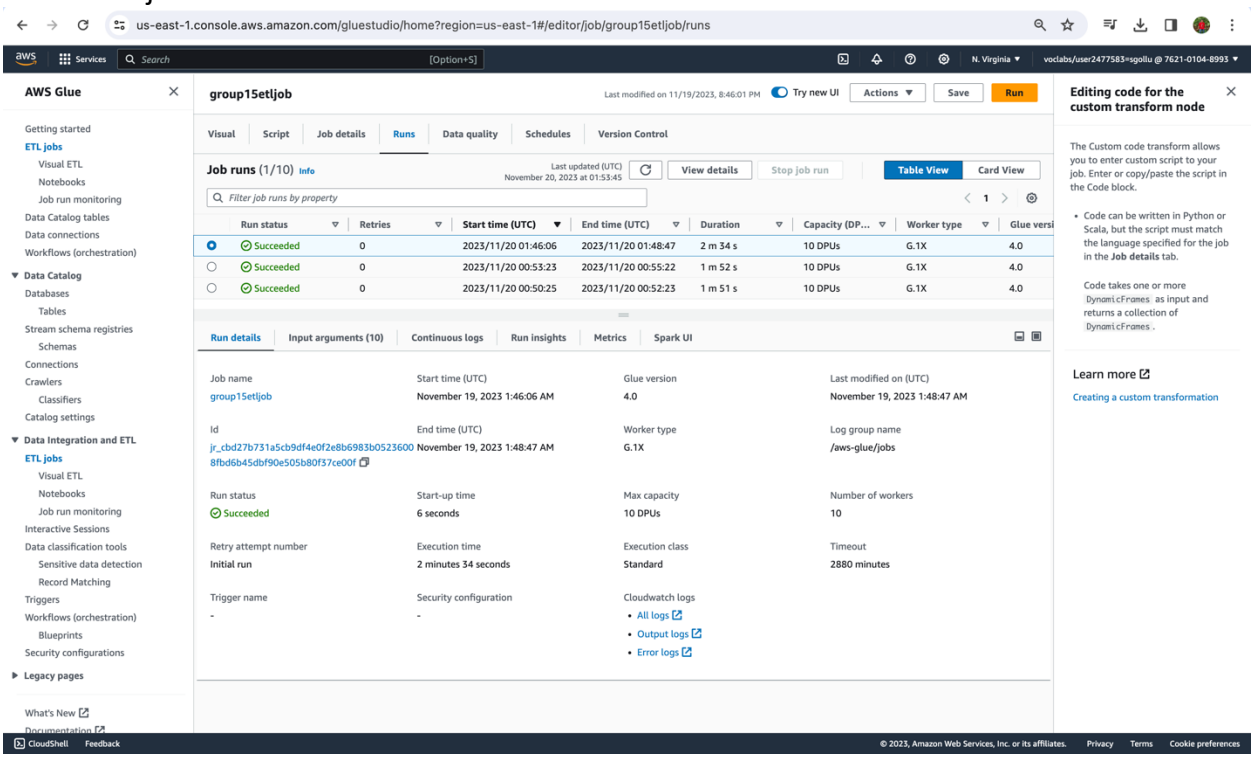


Fig: Shows the run progress of ETL Jobs in Glue

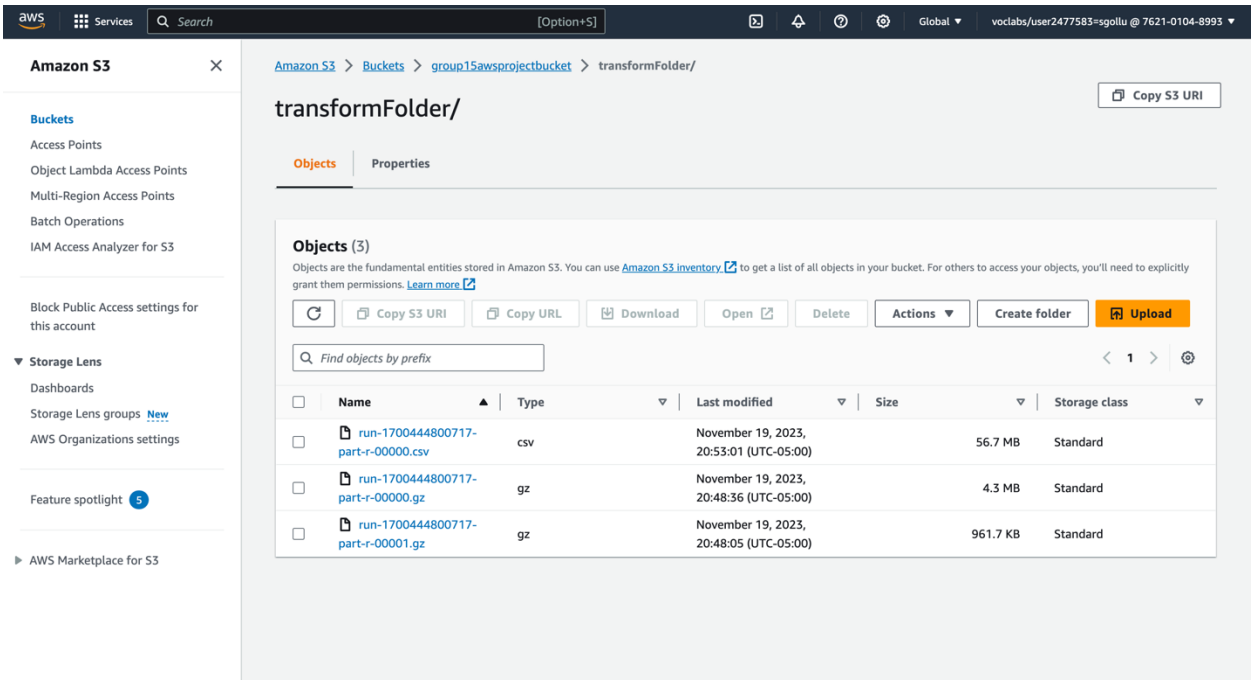


Fig: Transformed files are populated in a separate S3 folder.

After job execution the transformed file is stored as .gz file upon extraction gives the desired csv file.

Later Crawler is created to load schema from file in S3 to SQL

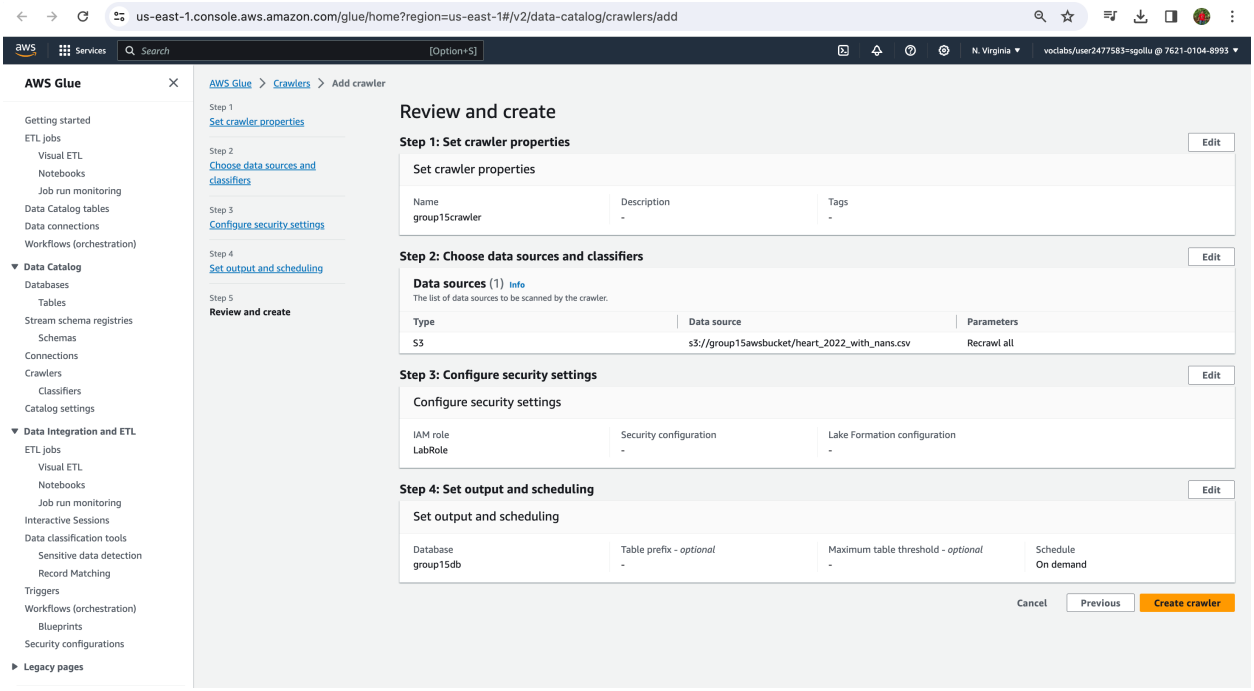
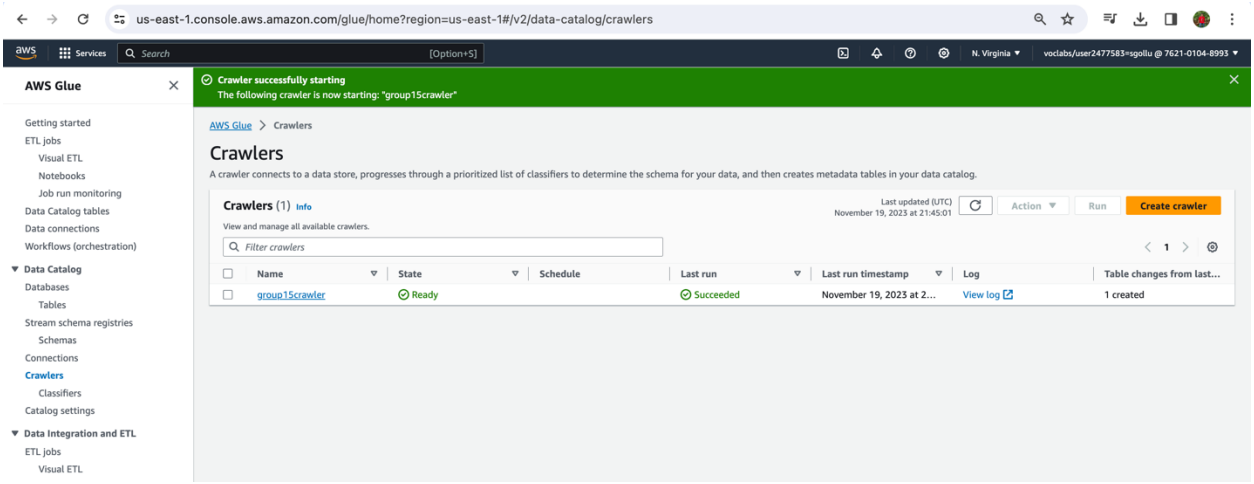


Fig: Crawler Creation



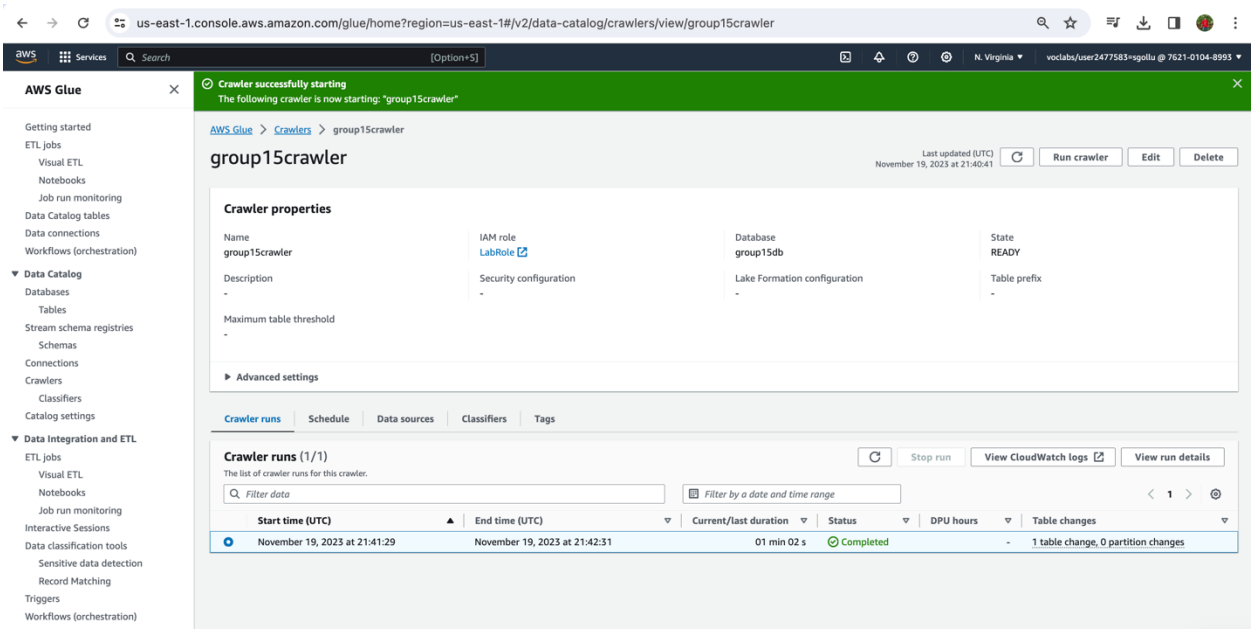


Fig: Created crawler is ready to use

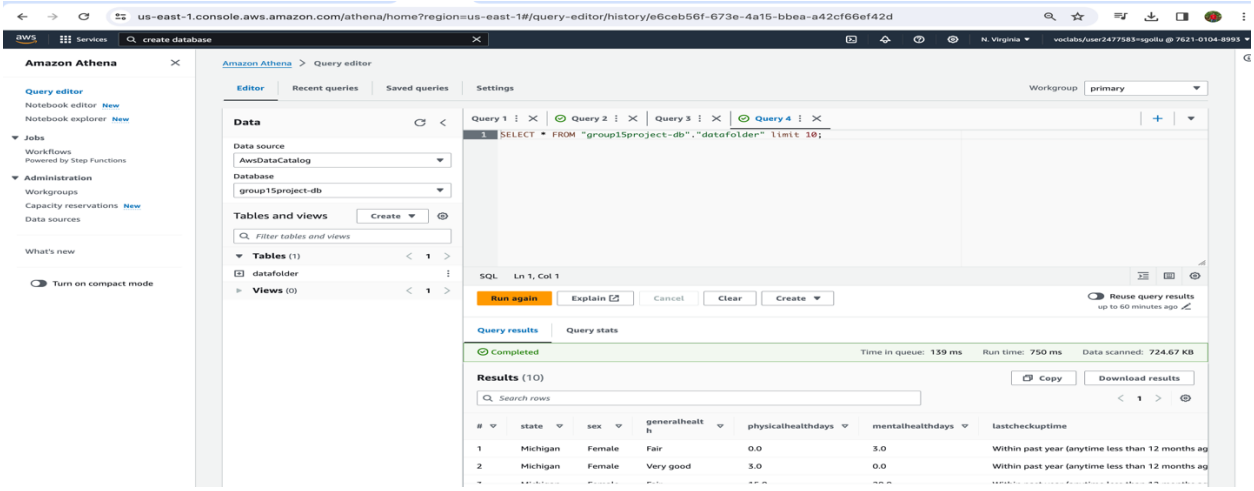


Fig: Query to verify the data imported from S3

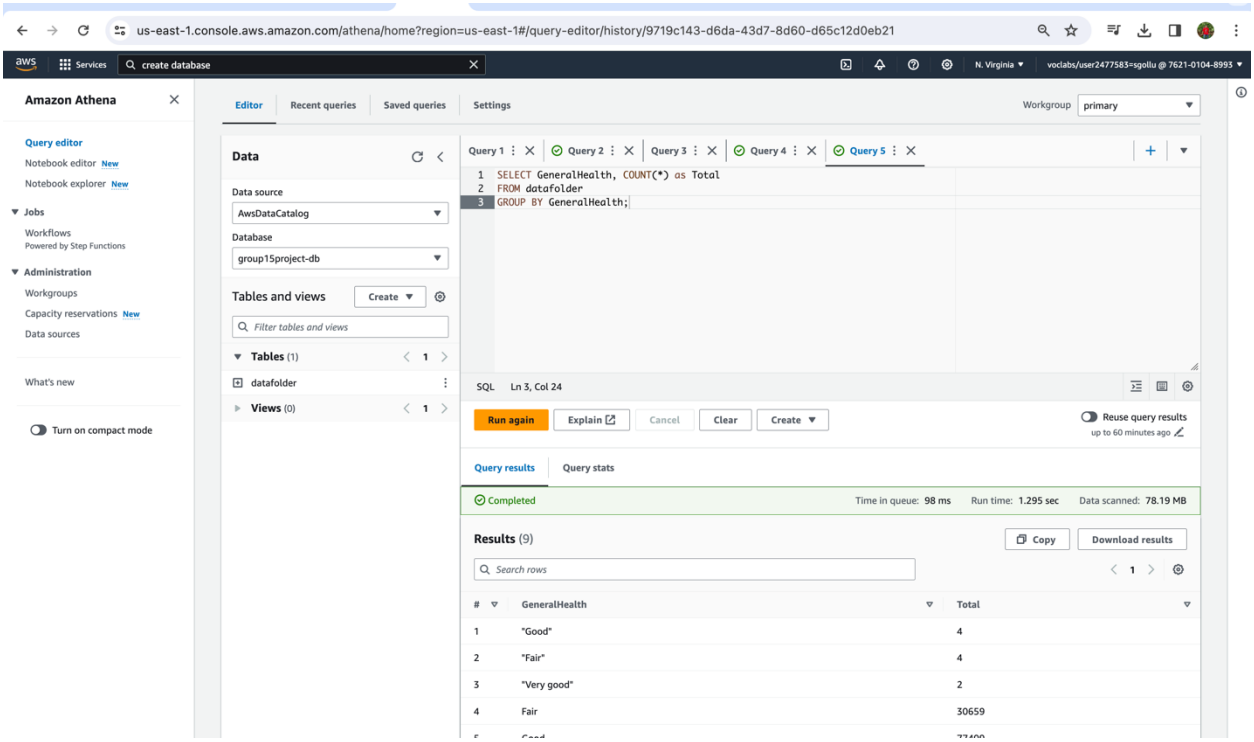


Fig: Query to understand the general health status distribution

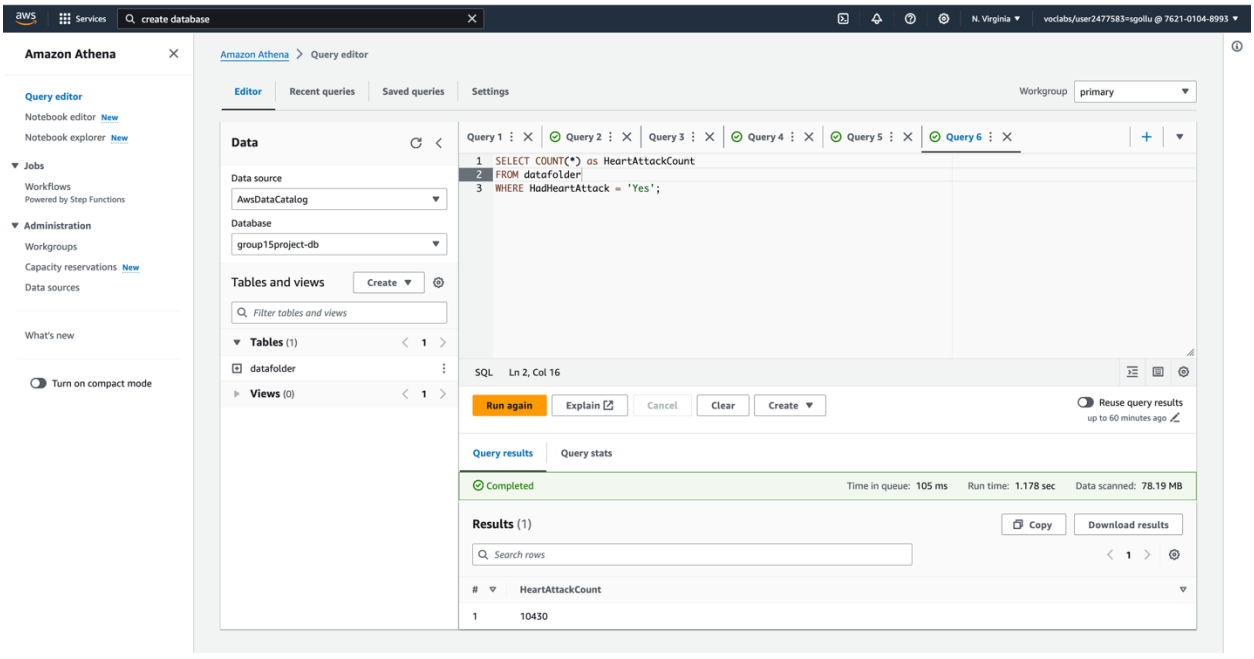


Fig: Finding Out no of records for Heart attacks in the dataset

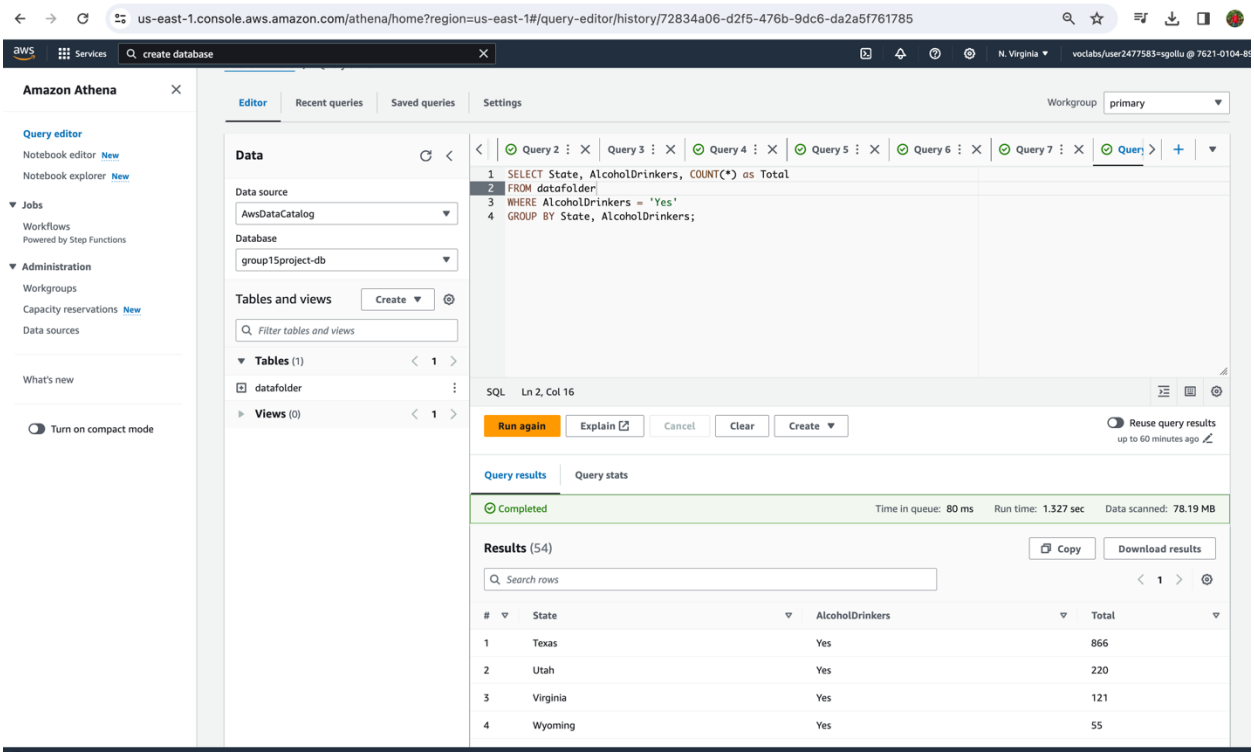


Fig: Query to understand alcohol drinkers in each state

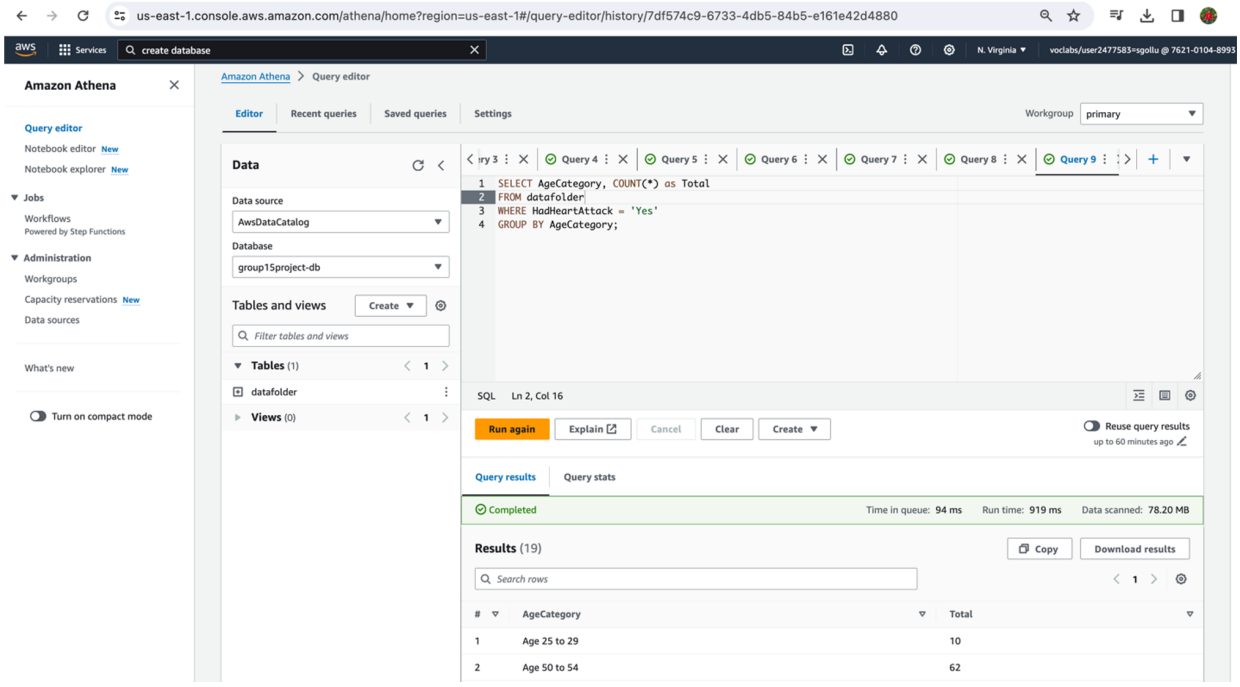


Fig: Query to understand heart attack trends in different age groups

After querying the results, visualizations are to be carried out in AWS Quicksight module, but the access levels didn't support the utilization of the module. Hence utilized sage maker to analyse the findings of data.

[12]:
```
import boto3
import pandas as pd
import sagemaker
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

[6]:
```
bucket = 'group15awsprojectbucket'


file_key = 'transformFolder/run-1700444800717-part-r-00000.csv'

# Construct the full S3 path
data_location = 's3://{}/{}'.format(bucket, file_key)

# Read the data into a Pandas DataFrame
df = pd.read_csv(data_location)


df.head()
```

[6]:

	deaforhardofhearing	generalhealth	difficultyconcentrating	physicalhealthdays	chestscan	hadskincancer	hivtesting	pneumovaxever	bli
0	No	Very good	No	4.0	No	No	No	Yes	
1	No	Very good	No	0.0	No	No	No	Yes	
2	No	Very good	No	0.0	Yes	No	No	Yes	
3	No	Fair	No	5.0	No	Yes	No	Yes	
4	No	Good	No	3.0	No	No	No	Yes	

5 rows × 38 columns

Fig: Conversion to data frames

[7]:
```
df.describe()
```

[7]:

	physicalhealthdays	heightinmeters	bmi	mentalhealthdays	weightinkilograms	smokerstatus	sleephours
count	192372.000000	192372.000000	192372.000000	192372.000000	192372.000000	192372.000000	192372.000000
mean	4.159145	1.704134	28.732025	4.116753	83.712744	0.603622	7.028445
std	8.440267	0.106620	6.532323	8.037662	21.407868	0.903312	1.446745
min	0.000000	0.910000	12.020000	0.000000	29.480000	0.000000	1.000000
25%	0.000000	1.630000	24.330000	0.000000	68.040000	0.000000	6.000000
50%	0.000000	1.700000	27.460000	0.000000	81.650000	0.000000	7.000000
75%	3.000000	1.780000	31.930000	4.000000	95.250000	1.000000	8.000000
max	30.000000	2.410000	97.650000	30.000000	292.570000	3.000000	24.000000

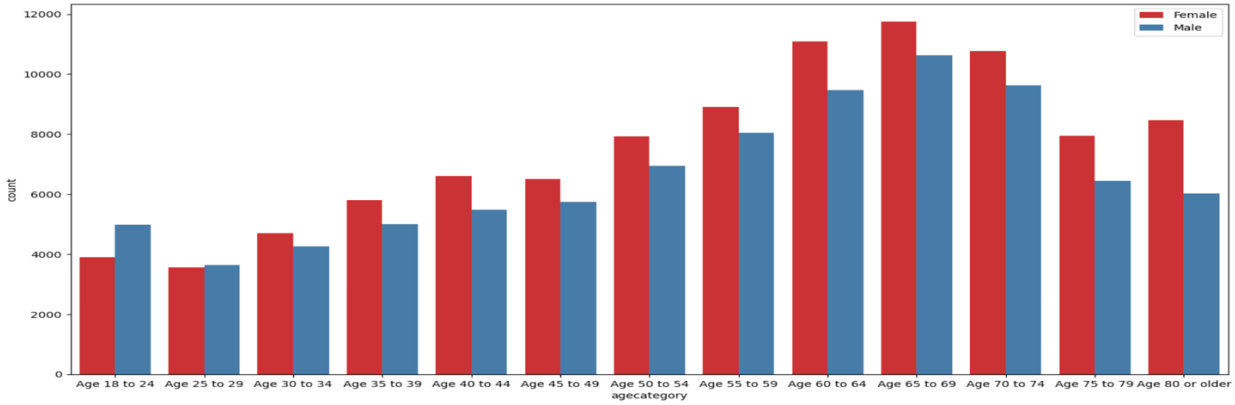
Fig: Verifying the data frame contents

```
# Define colors for the plot
irises_colors = ['rgb(33, 75, 99)', 'rgb(79, 129, 102)', 'rgb(151, 179, 100)',
                 'rgb(175, 49, 35)', 'rgb(36, 73, 147)']

cols = ['alcoholdrinkers', 'physicalactivities']

# Looping through columns to create pie charts
for i in cols:
    # Filter the DataFrame
    df_yes = df[df[i] == 'Yes']
    df_no = df[df[i] == 'No']
    fig = make_subplots(rows=1, cols=2, specs=[[{'type': 'domain'}, {'type': 'domain'}]])
    fig.add_trace(
        go.Pie(values=df_no.highrisklastyear.value_counts().values,
               labels=[f'No {i}', f'with {i}'],
               marker_colors=irises_colors),
        row=1, col=1
    )
    fig.add_trace(
        go.Pie(values=df_yes.highrisklastyear.value_counts().values,
               labels=[f'No {i}', f'with {i}'],
               marker_colors=irises_colors),
        row=1, col=2
    )
    fig.update_layout(
        legend_title="Labels",
        annotations=[
            dict(text=f'No {i}', x=0.15, y=1.10, font_size=30),
            dict(text=f'with {i}', x=0.80, y=1.10, font_size=30)
        ]
    )
    fig.show()
```

Fig: Code to understand the impact of alcohol and physical activity on heart disease

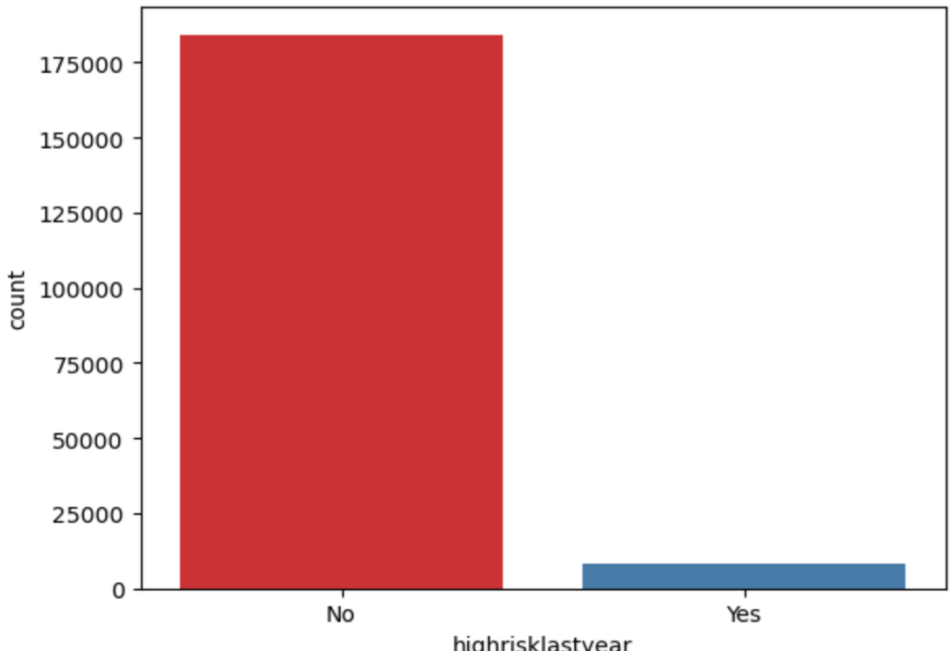

```
import matplotlib.pyplot as plt
import seaborn as sns
order = pd.unique(df.agecategory.values)
order.sort()
plt.figure(figsize = (17,8))
sns.countplot(data=df[df['highrisklastyear']=='No'], x='agecategory', hue='gender', palette='Set1', order=order)
plt.legend(['Female','Male'])
plt.show()
```

Fig: Code to understand the number of cases by segregating data with respect to age and gender.



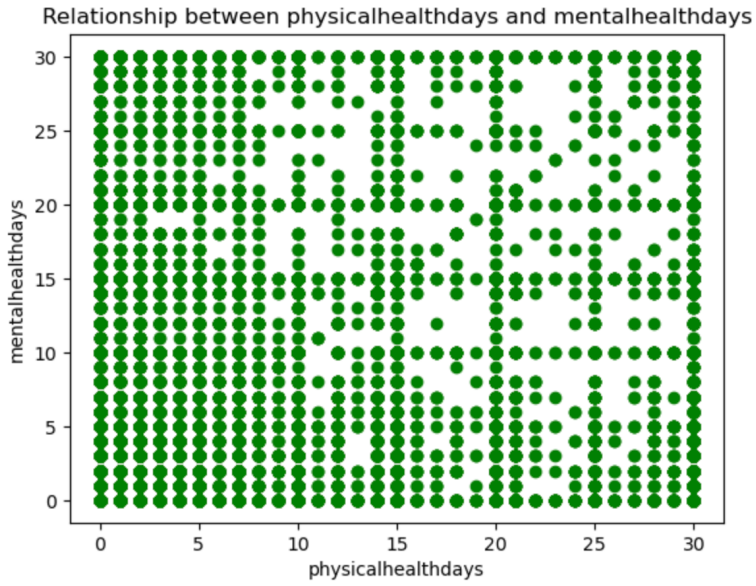
```
sns.countplot(data=df,x='highrisklastyear',palette='Set1')
```

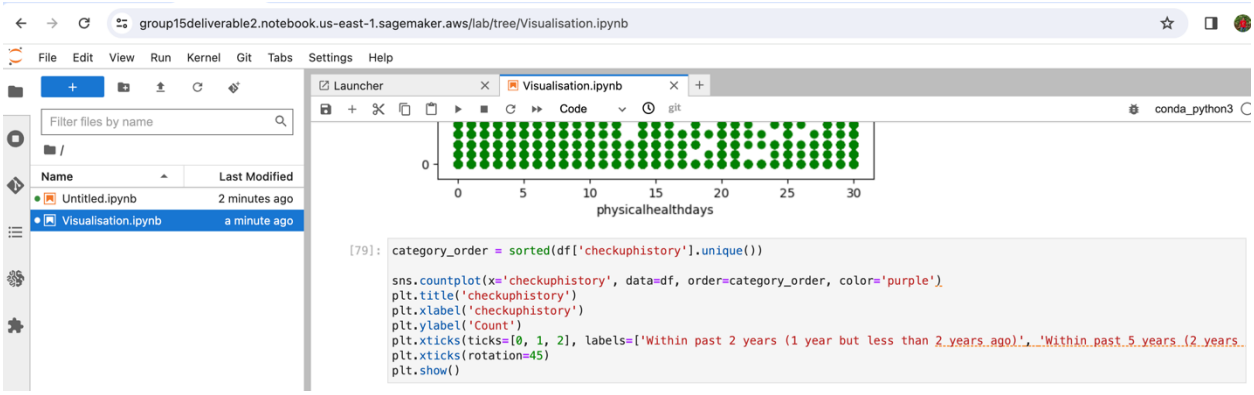
Code to understand data bias of the decision attribute.



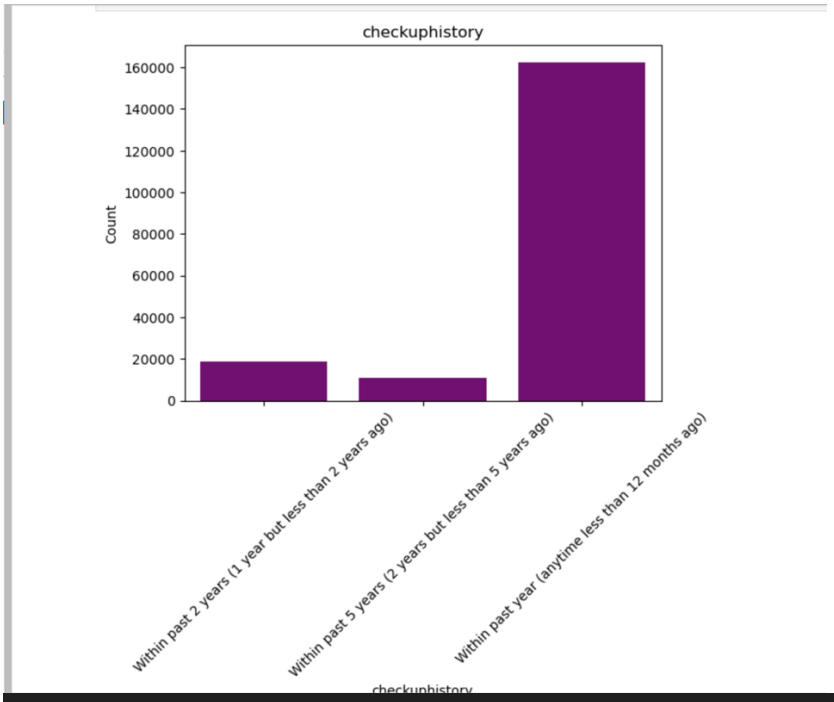

```
[78]: plt.scatter(df['physicalhealthdays'], df['mentalhealthdays'], color='green')
plt.title('Relationship between physicalhealthdays and mentalhealthdays')
plt.xlabel('physicalhealthdays')
plt.ylabel('mentalhealthdays')
plt.show()
```

Code to understand the relationship between physical and mental health of a patient



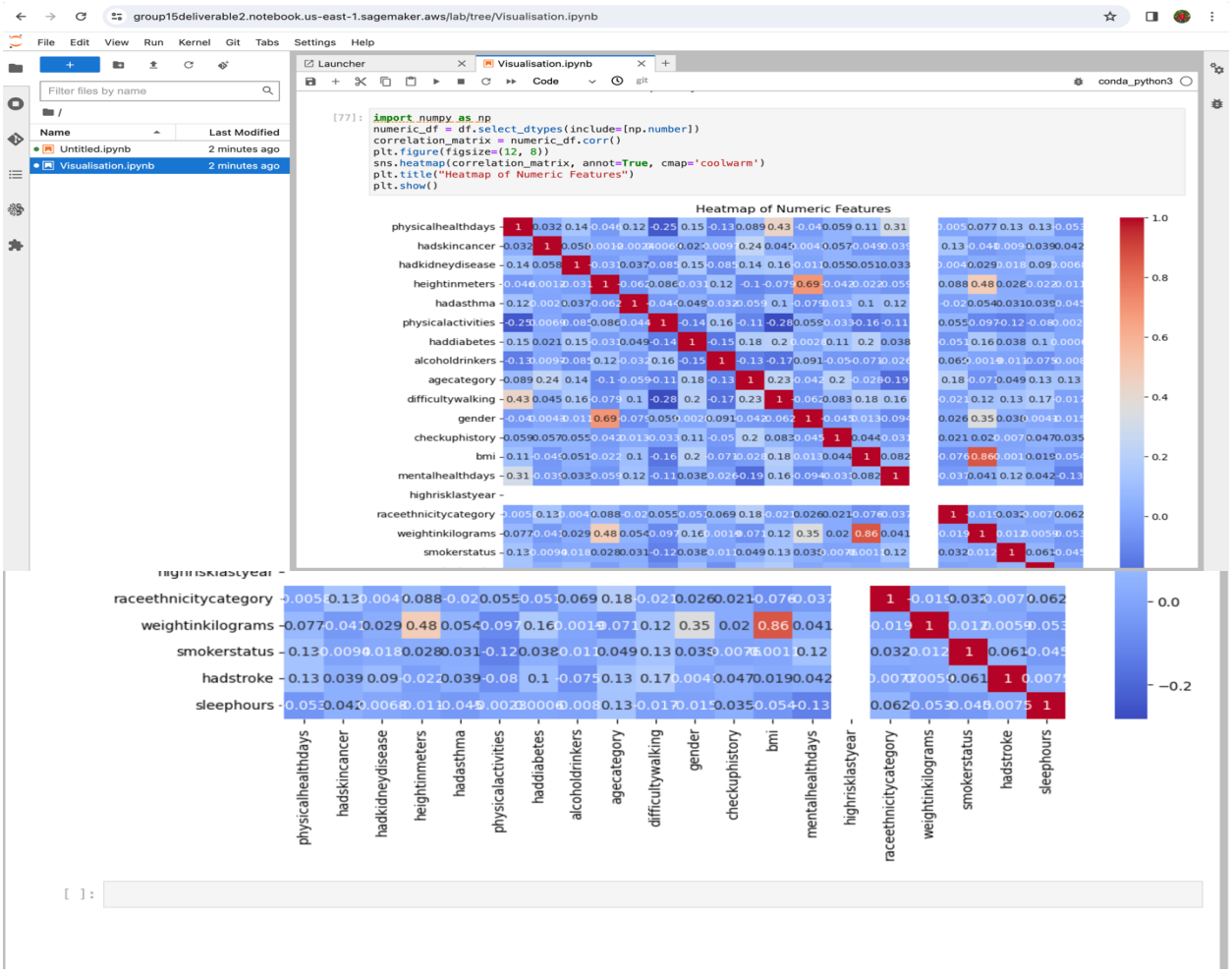


Code to understand Data understand number of heart attacks and frequency of health check ups




```
import numpy as np
numeric_df = df.select_dtypes(include=[np.number])
correlation_matrix = numeric_df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Heatmap of Numeric Features")
plt.show()
```

Code to understand the correlation map between different attributes of data set.



Correlation Matrix

AWS Pipeline for the above would be as follows: