

# 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

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various navigation options like Buckets, Access Points, Object Lambda Access Points, etc. The main area is titled 'Amazon S3' and shows an 'Account snapshot' with a link to 'View Storage Lens dashboard'. Below that is a table titled 'Buckets (3) Info' with columns for Name, AWS Region, Access, and Creation date. The buckets listed are:

Name	AWS Region	Access	Creation date
aws-glue-assets-762101048993-us-east-1	US East (N. Virginia) us-east-1	Bucket and objects not public	November 19, 2023, 17:54:21 (UTC-05:00)
group15awsbucket	US East (N. Virginia) us-east-1	Bucket and objects not public	November 19, 2023, 16:04:15 (UTC-05:00)
group15awsprojectbucket	US East (N. Virginia) us-east-1	Bucket and objects not public	November 19, 2023, 17:09:50 (UTC-05:00)

Fig: S3 buckets creation to import the dataset from kaggle

This screenshot shows the AWS S3 console with a more detailed view. It's navigating through 'Amazon S3 > Buckets > group15awsprojectbucket > datafolder/'. The 'Objects' tab is selected, showing a list of files. There are two objects: 'heart\_2022\_no\_nans.csv' (a CSV file) and 'Unsaved/' (a folder). The table has columns for Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
heart_2022_no_nans.csv	csv	November 19, 2023, 17:10:23 (UTC-05:00)	78.2 MB	Standard
Unsaved/	Folder	-	-	-

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.

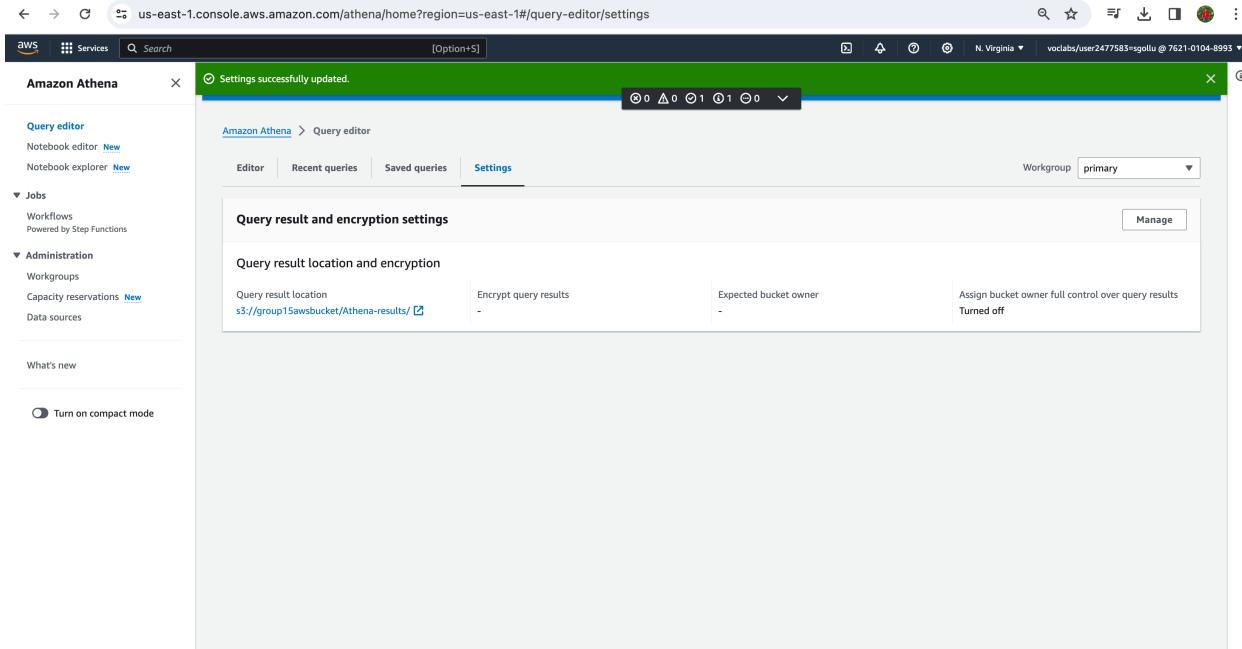
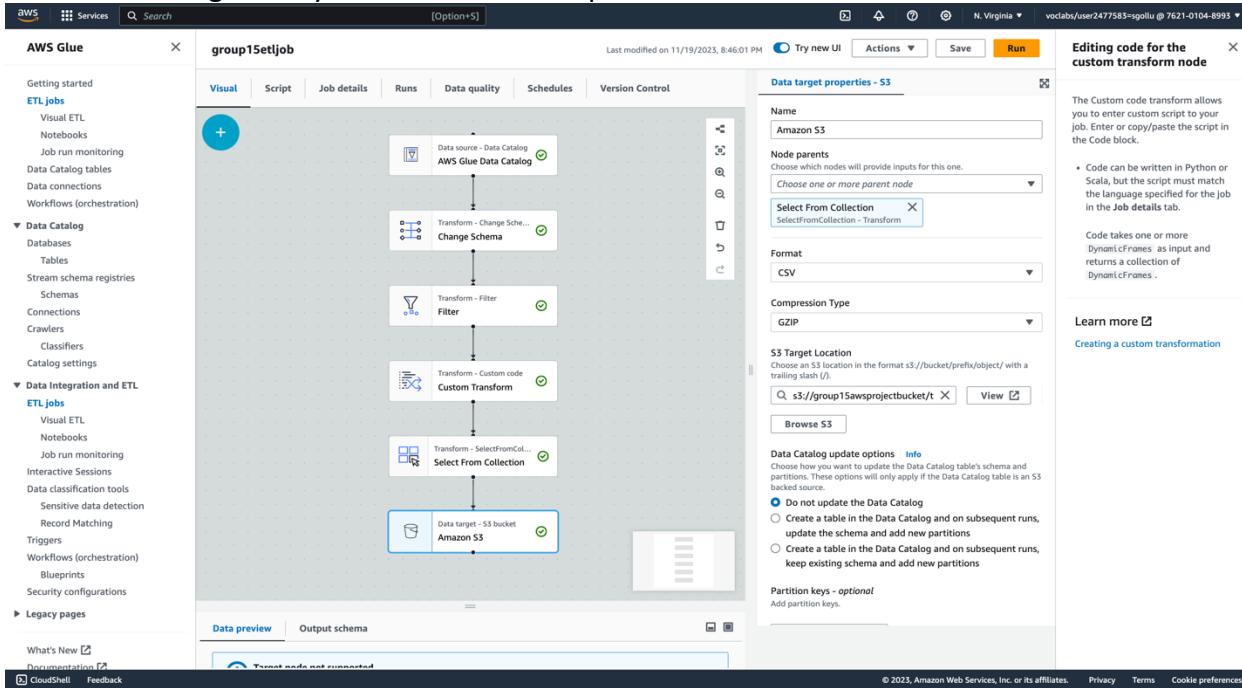


Fig : Query Result location setup for SQL Queries executed in Athena



## Fig: Steps involved in ETL

The screenshot shows the AWS Glue Visual ETL interface for a job named "group15etljob". The left sidebar contains navigation links for AWS Glue, ETL jobs, Data Catalog, and Data Integration & ETL. The main area displays a visual workflow with a single node labeled "Transform - Change Schema". Below the workflow is a "Data preview" section showing 200 rows of sample data from Alabama. To the right, the "Transform" configuration pane is open, specifically the "Change Schema (Apply mapping)" tab. This tab lists source keys and target keys with their corresponding data types. For example, "state" is mapped to "string", "gender" to "string", and "generalhealth" to "string". A "removedteeth" column is listed with its target key "removedteeth" checked. The bottom right of the screen includes standard AWS footer links for privacy, terms, and cookie preferences.

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.

This screenshot shows the same AWS Glue Visual ETL interface for the "group15etljob" job. The workflow now includes a "Filter" node. The "Transform" configuration pane is expanded to show the "Filter" configuration. It defines two filter conditions using the "checkuphistory" key: one for "matches" within the past 2 years and another for "matches" within the past 5 years. The data preview section shows rows where the "checkuphistory" value is "No" for all patients, indicating they have no recent checkups.

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.

The screenshot shows the AWS Glue visual editor interface. On the left, there's a sidebar with navigation links for ETL jobs, Data Catalog, and Data Integration and ETL. The main workspace displays a 'group1SetJob' job. A 'Transform - Custom code' step is selected, labeled 'Custom Transform'. Below it, a 'Data preview' section shows 200 rows of data with columns like 'deaforhardofhearing', 'generalhealth', 'difficultyconcentratin', 'physicalhealthdays', and 'chestsc'. To the right, the 'Transform' panel is open, showing the code block:

```

1 def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
2     # Define the mapping of status values to numbers
3     status_mapping = {
4         'Never smoked': 0,
5         'Former smoker': 1,
6         'Current smoker - now smokes some days': 2,
7         'Current smoker - now smokes every day': 3
8     }
9
10    def map_status_to_number(rec):
11        # Get the status from the record
12        status_value = rec["smokerstatus"]
13        # Map the status to a number using the dictionary, default
14        # rec["smokerstatus"] = status_mapping.get(status_value, None)
15        # return rec
16
17        # Retrieve the DynamicFrame to be transformed
18        dynamic_frame = dfc.select(list(dfc.keys())[0])
19
20        # Apply the map function to each record
21        mapped_dyf = Map.applyVFrame = dynamic_frame, f = map_status_to_
22
23        # Return the new DynamicFrameCollection
24        return DynamicFrameCollection({"transformed_dyf": mapped_dyf}, g

```

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.

Run details	Input arguments (10)	Continuous logs	Run insights	Metrics	Spark UI
Job name group15etljob	Start time (UTC) November 19, 2023 1:46:06 AM	Glue version 4.0	Last modified on (UTC) November 19, 2023 1:48:47 AM		
Id jr_cbd27b731a5cb9df4e0f2e8b6983b0523600	End time (UTC) November 19, 2023 1:48:47 AM	Worker type G.1X	Log group name /aws-glue/jobs		
	Run status Succeeded	Start-up time 6 seconds	Max capacity 10 DPU	Number of workers 10	
	Retry attempt number Initial run	Execution time 2 minutes 34 seconds	Execution class Standard	Timeout 2880 minutes	
	Trigger name -	Security configuration -	Cloudwatch logs		
			<ul style="list-style-type: none"> <li>All logs</li> <li>Output logs</li> <li>Error logs</li> </ul>		

Fig: Shows the run progress of ETL Jobs in Glue

The screenshot shows the AWS S3 console interface. On the left, the navigation pane for 'Amazon S3' is visible, showing options like Buckets, Storage Lens, and AWS Marketplace. The main content area displays a bucket named 'group15awsprojectbucket' with a folder 'transformFolder'. Inside this folder, there are three objects: 'run-1700444800717-part-r-00000.csv', 'run-1700444800717-part-r-00000.gz', and 'run-1700444800717-part-r-00001.gz'. The objects are listed in a table with columns for Name, Type, Last modified, Size, and Storage class. A 'Copy S3 URI' button is located at the top right of the object list.

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

The screenshot shows the AWS Glue console interface. The left sidebar lists various Glue services like ETI jobs, Data Catalog, and Data Integration & ETL. The main area is titled 'Add crawler' and contains five steps: Step 1: Set crawler properties, Step 2: Choose data sources and classifiers, Step 3: Configure security settings, Step 4: Set output and scheduling, and Step 5: Review and create. The 'Step 2: Choose data sources and classifiers' step is currently active, showing a table with one data source entry: Type 'S3', Data source 's3://group15awsbucket/heart\_2022\_with\_nans.csv', and Parameters 'Recrawl all'. The 'Step 3: Configure security settings' step shows IAM role 'LabRole' and Security configuration ' '. The 'Step 4: Set output and scheduling' step shows Database 'group15db', Table prefix - optional ' ', Maximum table threshold - optional ' ', and Schedule 'On demand'. At the bottom right, there are 'Cancel', 'Previous', and 'Create crawler' buttons.

Fig: Crawler Creation

us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/crawlers

us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/crawlers/view/group15crawler

Fig: Created crawler is ready to use

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/e6ceb56f-673e-4a15-bbea-a42cf66ef42d

#	state	sex	generalhealth	physicalhealthdays	mentalhealthdays	lastcheckuptime
1	Michigan	Female	Fair	0.0	3.0	Within past year (anytime less than 12 months ago)
2	Michigan	Female	Very good	3.0	0.0	Within past year (anytime less than 12 months ago)
3	Michigan	Female	Good	3.0	0.0	Within past year (anytime less than 12 months ago)
4	Michigan	Female	Excellent	3.0	0.0	Within past year (anytime less than 12 months ago)
5	Michigan	Female	Good	3.0	0.0	Within past year (anytime less than 12 months ago)
6	Michigan	Female	Excellent	3.0	0.0	Within past year (anytime less than 12 months ago)
7	Michigan	Female	Good	3.0	0.0	Within past year (anytime less than 12 months ago)
8	Michigan	Female	Excellent	3.0	0.0	Within past year (anytime less than 12 months ago)
9	Michigan	Female	Good	3.0	0.0	Within past year (anytime less than 12 months ago)
10	Michigan	Female	Excellent	3.0	0.0	Within past year (anytime less than 12 months ago)

Fig: Query to verify the data imported from S3

The screenshot shows the AWS Athena Query Editor interface. The left sidebar includes sections for Services, Query editor (selected), Notebook editor, Jobs, Administration, and What's new. The main area has tabs for Editor, Recent queries, Saved queries, and Settings. The Workgroup is set to primary. The Data section shows a Data source of AwsDataCatalog and a Database of group15project-db. The Tables and views section lists one table named datafolder. The SQL editor contains the following query:

```
1 SELECT GeneralHealth, COUNT(*) as Total
2 FROM datafolder
3 GROUP BY GeneralHealth;
```

The Query results tab shows the output of the query:

#	GeneralHealth	Total
1	"Good"	4
2	"Fair"	4
3	"Very good"	2
4	Fair	30659

Fig: Query to understand the general health status distribution

The screenshot shows the AWS Athena Query Editor interface, identical to the previous one but with a different query. The left sidebar and Data section are the same. The SQL editor contains the following query:

```
1 SELECT COUNT(*) as HeartAttackCount
2 FROM datafolder
3 WHERE HadHeartAttack = 'Yes';
```

The Query results tab shows the output of the query:

#	HeartAttackCount
1	10430

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

The screenshot shows the AWS Athena Query Editor interface. The left sidebar includes links for Notebook editor, Jobs, Administration, and What's new. The main area has tabs for Data, Recent queries, Saved queries, and Settings. The Data tab is selected, showing a Data source (AwsDataCatalog) and Database (group15project-db). Under Tables and views, there is one table named 'datafolder'. The SQL query entered is:

```

1 SELECT State, AlcoholDrinkers, COUNT(*) as Total
2 FROM datafolder
3 WHERE AlcoholDrinkers = 'Yes'
4 GROUP BY State, AlcoholDrinkers;

```

The results table shows the following data:

#	State	AlcoholDrinkers	Total
1	Texas	Yes	866
2	Utah	Yes	220
3	Virginia	Yes	121
4	Wyoming	Yes	55

Fig: Query to understand alcohol drinkers in each state

This screenshot is identical to the first one, showing the same interface and query results. The query is:

```

1 SELECT AgeCategory, COUNT(*) as Total
2 FROM datafolder
3 WHERE HadHeartAttack = 'Yes'
4 GROUP BY AgeCategory;

```

The results table shows the following data:

#	AgeCategory	Total
1	Age 25 to 29	10
2	Age 50 to 54	62

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']

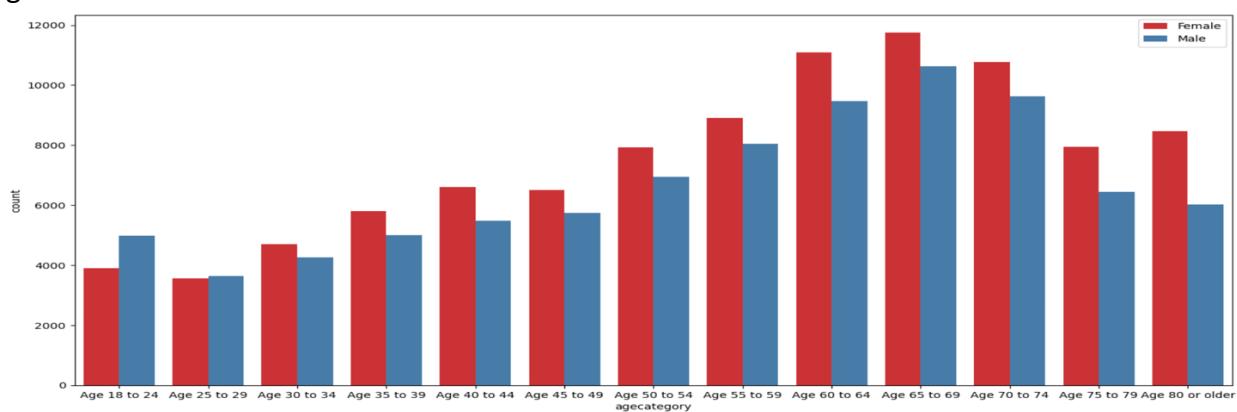
# Looping through columns to create pie charts
for i in cols:
    # Filter the DataFrame
    df_yes = df[df[i] == 'Yes']
    df_no = df[df[i] == 'No']
    fig = make_subplots(rows=1, cols=2, specs=[[{'type': 'domain'}, {'type': 'domain'}]])
    fig.add_trace(
        go.Pie(values=df_no.highrisklastyear.value_counts().values,
               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



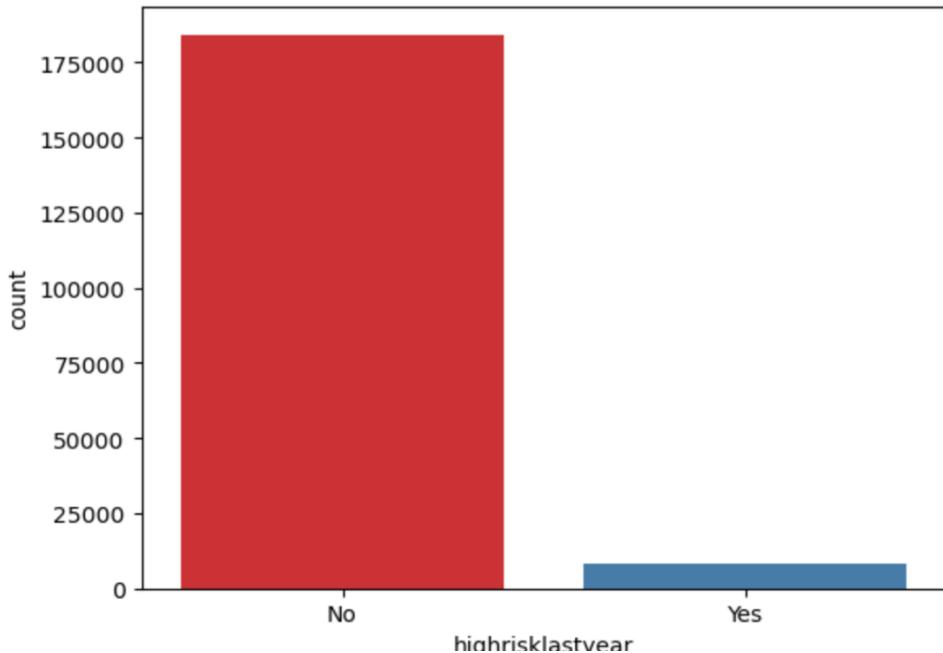
```
import matplotlib.pyplot as plt
import seaborn as sns
order = pd.unique(df['agecategory'].values)
order.sort()
plt.figure(figsize=(17,8))
sns.countplot(data=df[df['highrisklastyear']=='No'], x='agecategory', hue='gender', palette='Set1', order=order)
plt.legend(['Female','Male'])
plt.show()
```

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



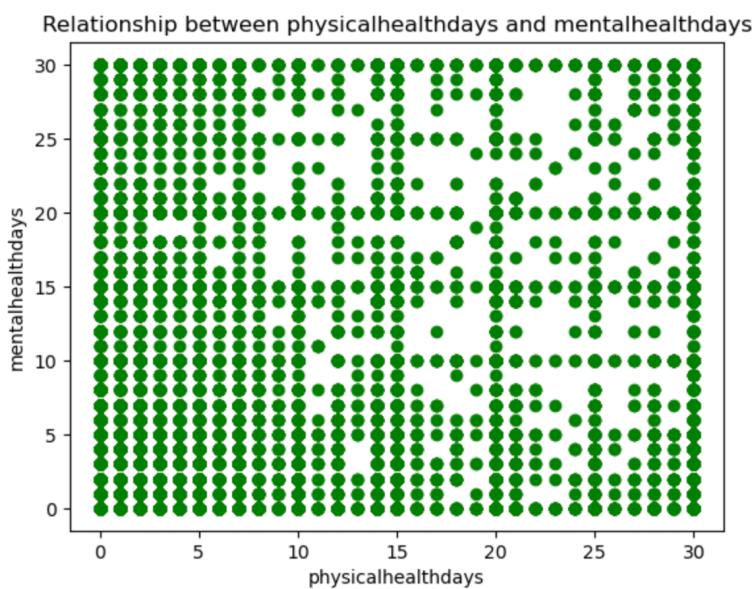
```
sns.countplot(data=df,x='highrisklastyear',palette='Set1')
```

Code to understand data bias of the decision attribute.



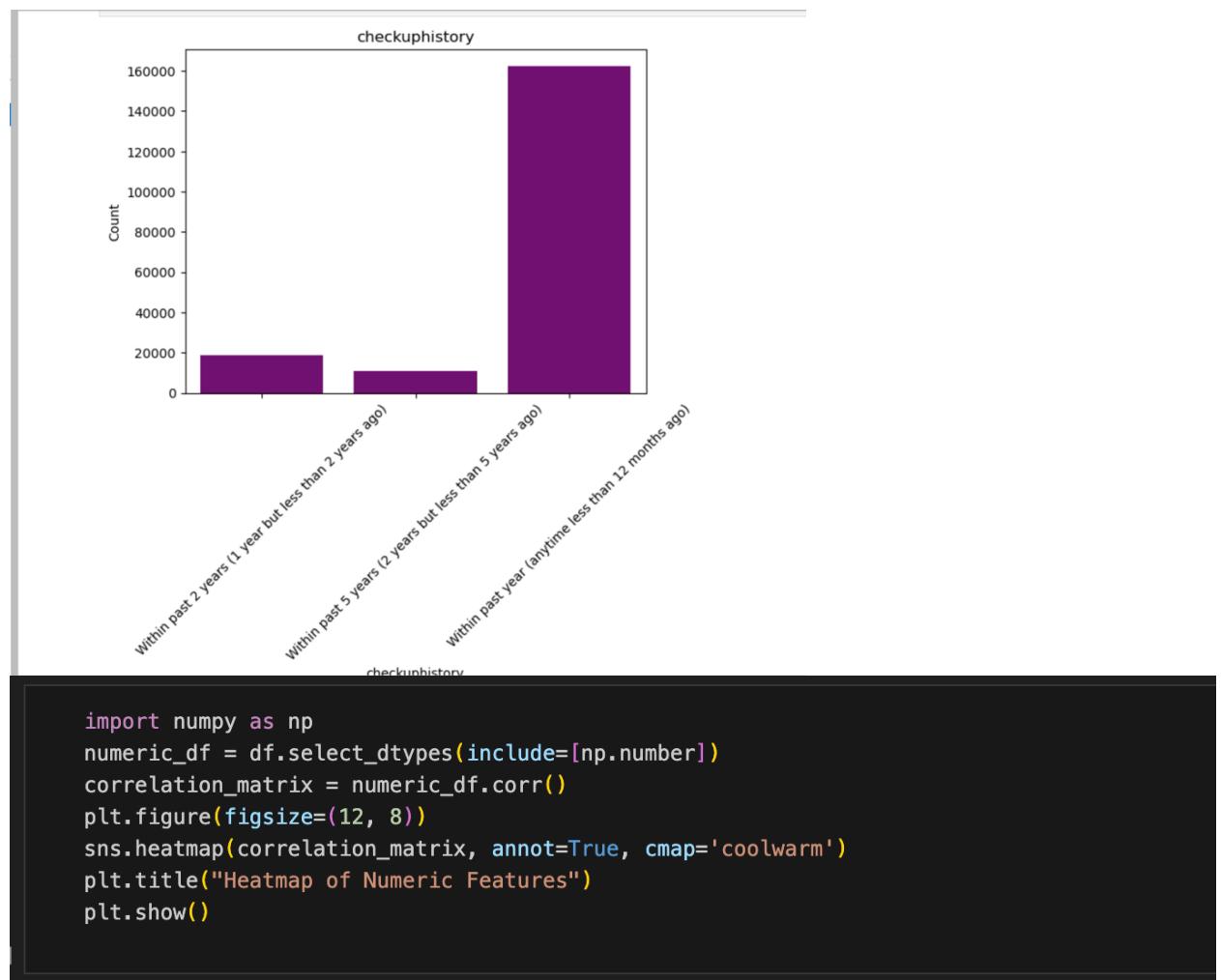
```
[78]: plt.scatter(df['physicalhealthdays'], df['mentalhealthdays'], color='green')
plt.title('Relationship between physicalhealthdays and mentalhealthdays')
plt.xlabel('physicalhealthdays')
plt.ylabel('mentalhealthdays')
plt.show()
```

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

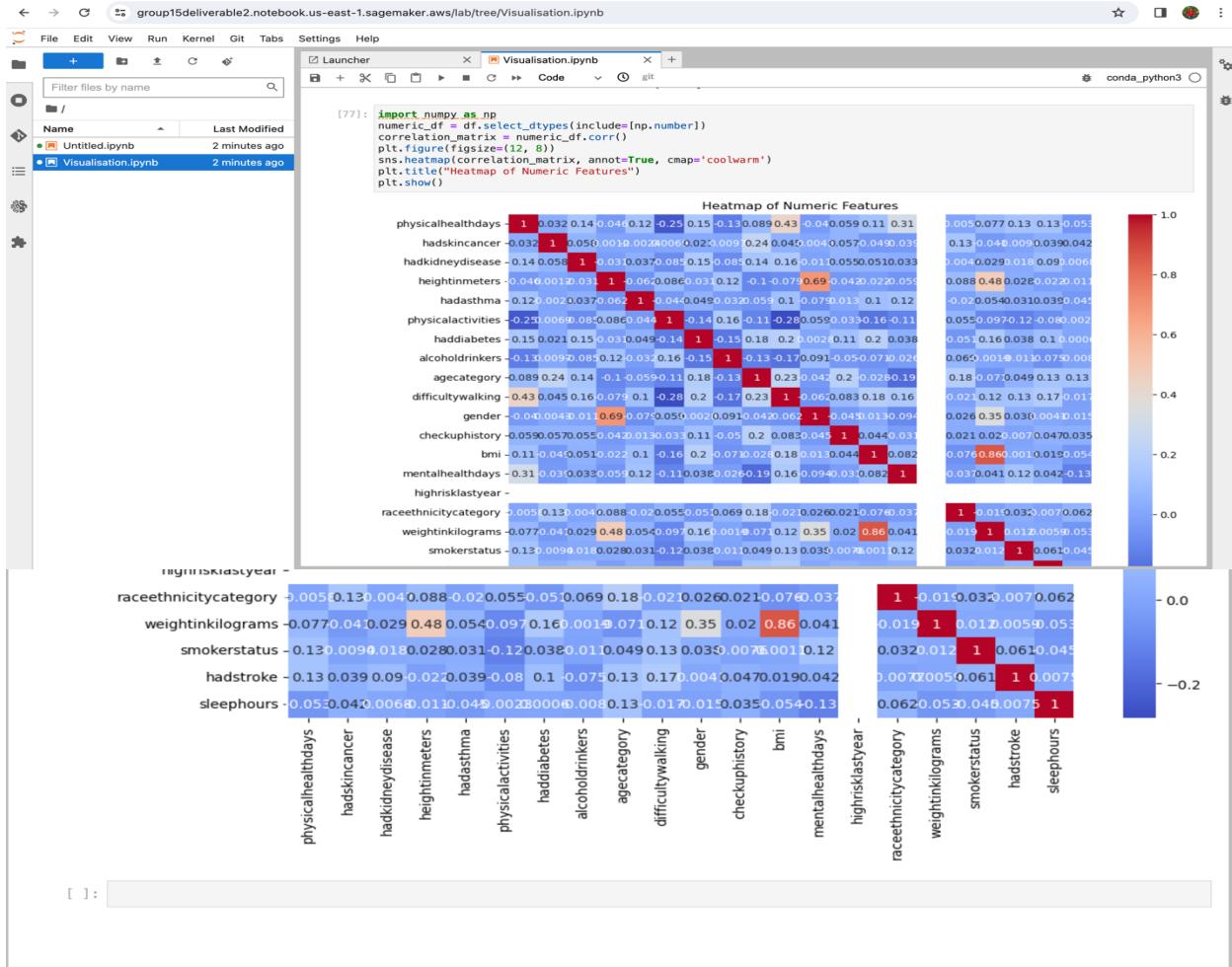


```
[79]: category_order = sorted(df['checkuphistory'].unique())
sns.countplot(x='checkuphistory', data=df, order=category_order, color='purple')
plt.title('checkuphistory')
plt.xlabel('checkuphistory')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1, 2], labels=['Within past 2 years (1 year but less than 2 years ago)', 'Within past 5 years (2 years but less than 5 years ago)', 'Within past year (anytime less than 12 months ago)'])
plt.show()
```

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



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



Correlation Matrix

AWS Pipeline for the above would be as follows:

