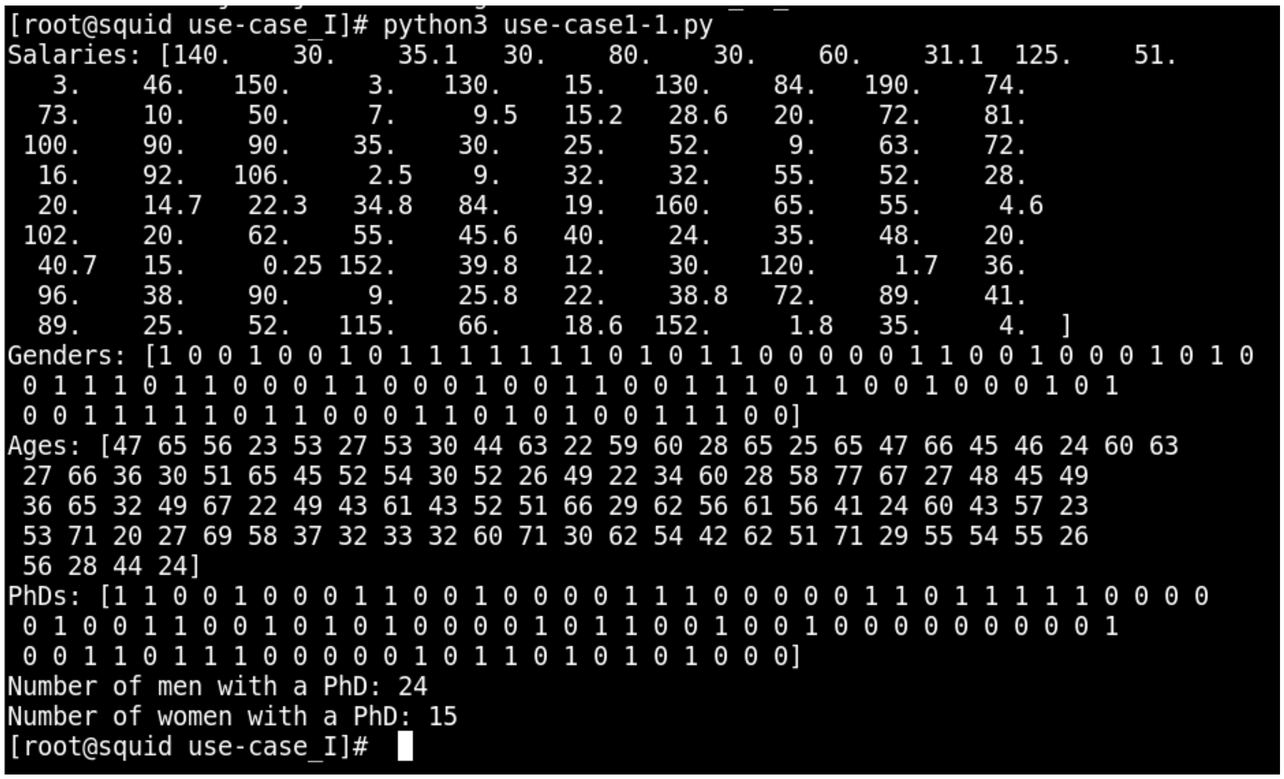
Module 4: Introduction to NumPy & Pandas

Case Study I

1. Extract data from the given SalaryGender CSV file and store the data from each column in a separate NumPy array

2. Find:
1. The number of men with a PhD
2. The number of women with a PhD

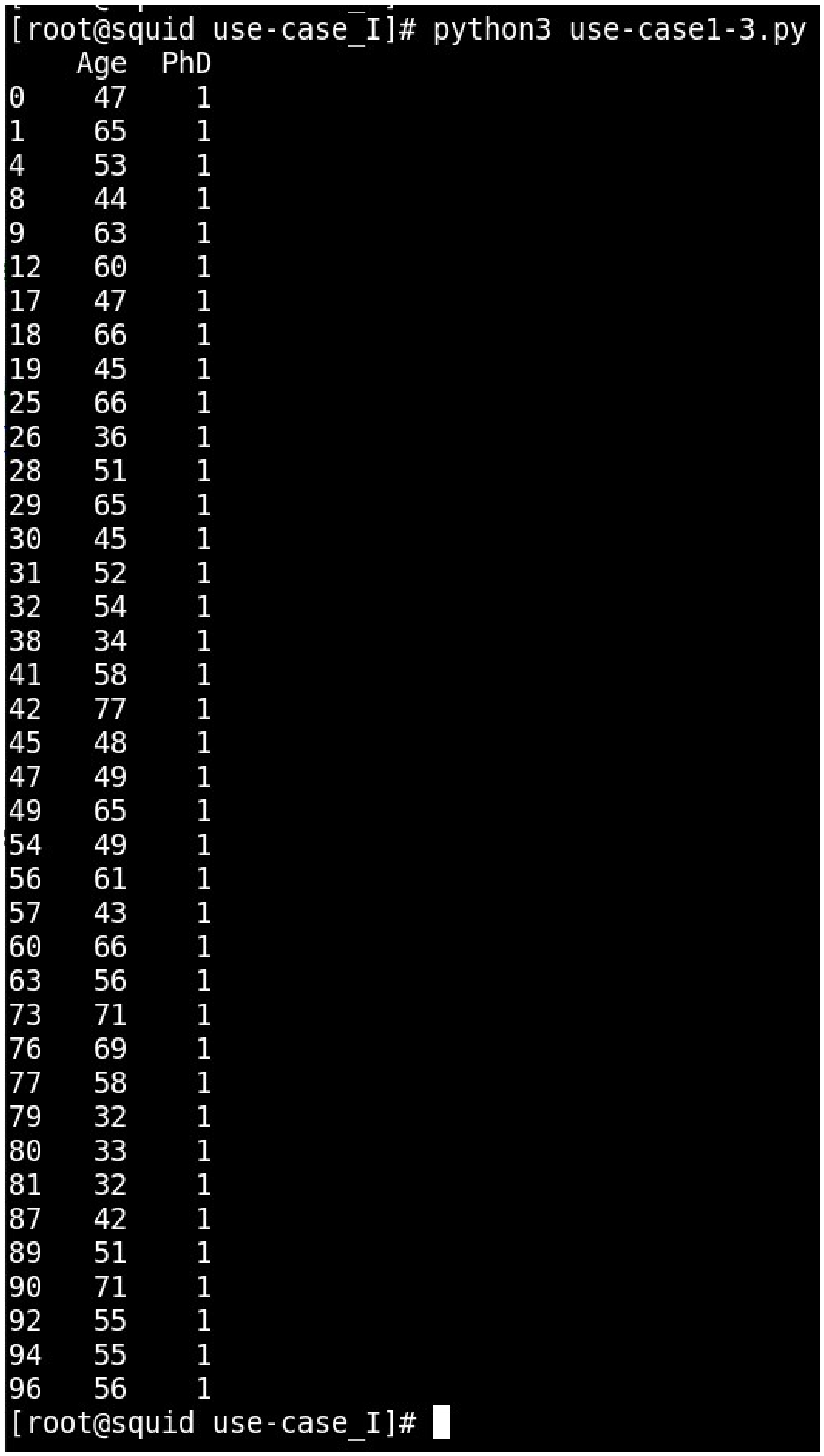
```
import numpy as np

# Path to the CSV file
file_path = 'SalaryGender.csv'

# Use numpy.genfromtxt to read the data, skipping the header row
data = np.genfromtxt(file_path, delimiter=',', skip_header=1)

# Extract each column into a separate NumPy array
# Ensuring salaries are treated as floats
salaries = data[:, 0].astype(float)  # Already float by default, but emphasizing for clarity
genders = data[:, 1].astype(int)  # Convert to int for consistency
ages = data[:, 2].astype(int)  # Convert to int for consistency
phds = data[:, 3].astype(int)  # Convert to int for consistency

# Print the arrays (optional)
print("Salaries:", salaries)
print("Genders:", genders)
print("Ages:", ages)
print("PhDs:", phds)

# Number of men with a PhD
men_with_phd = np.sum((genders == 1) & (phds == 1))

# Number of women with a PhD
women_with_phd = np.sum((genders == 0) & (phds == 1))

# Print the results
print("Number of men with a PhD:", men_with_phd)
print("Number of women with a PhD:", women_with_phd)
```



```
[root@squid use-case_I]# python3 use-case1-1.py
Salaries: [140.    30.    35.1   30.    80.    30.    60.    31.1  125.    51.
   3.    46.   150.     3.   130.    15.   130.    84.   190.    74.
  73.    10.    50.     7.     9.5   15.2   28.6   20.    72.    81.
 100.    90.    90.    35.    30.    25.    52.     9.    63.    72.
  16.    92.   106.     2.5    9.    32.    32.    55.    52.    28.
  20.    14.7   22.3   34.8   84.    19.   160.    65.    55.     4.6
 102.    20.    62.    55.    45.6   40.    24.    35.    48.    20.
  40.7   15.     0.25 152.    39.8   12.    30.   120.     1.7   36.
  96.    38.    90.     9.    25.8   22.    38.8   72.    89.    41.
  89.    25.    52.   115.    66.    18.6  152.     1.8   35.     4.  ]
Genders: [1 0 0 1 0 0 1 0 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 1 0
 0 1 1 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 0 1 0 0 0 1 0 1
 0 0 1 1 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 0 0]
Ages: [47 65 56 23 53 27 53 30 44 63 22 59 60 28 65 25 65 47 66 45 46 24 60 63
 27 66 36 30 51 65 45 52 54 30 52 26 49 22 34 60 28 58 77 67 27 48 45 49
 36 65 32 49 67 22 49 43 61 43 52 51 66 29 62 56 61 56 41 24 60 43 57 23
 53 71 20 27 69 58 37 32 33 32 60 71 30 62 54 42 62 51 71 29 55 54 55 26
 56 28 44 24]
PhDs: [1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 1 0 0 0 0
 0 1 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1
 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0]
Number of men with a PhD: 24
Number of women with a PhD: 15
[root@squid use-case_I]#
```

3. Store the “Age” and “PhD” columns in one DataFrame and delete the data of all people who don’t have a PhD from SalaryGender CSV file.

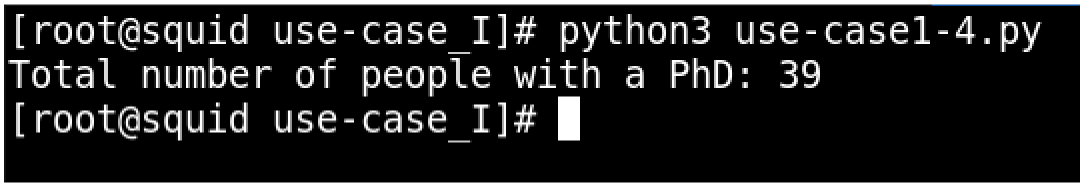
```
import pandas as pd

# Load the CSV file into a DataFrame
df = pd.read_csv('SalaryGender.csv')

# Filter the DataFrame to keep only those with a PhD
df_with_phd = df[df['PhD'] == 1]

# Select only the "Age" and "PhD" columns
age_phd_df = df_with_phd[['Age', 'PhD']]

# Display the resulting DataFrame
print(age_phd_df)
```



```
[root@squid use-case_I]# python3 use-case1-3.py
    Age  PhD
0    47    1
1    65    1
4    53    1
8    44    1
9    63    1
12   60    1
17   47    1
18   66    1
19   45    1
25   66    1
26   36    1
28   51    1
29   65    1
30   45    1
31   52    1
32   54    1
38   34    1
41   58    1
42   77    1
45   48    1
47   49    1
49   65    1
54   49    1
56   61    1
57   43    1
60   66    1
63   56    1
73   71    1
76   69    1
77   58    1
79   32    1
80   33    1
81   32    1
87   42    1
89   51    1
90   71    1
92   55    1
94   55    1
96   56    1
[root@squid use-case_I]# 
```

4. Calculate the total number of people who have a PhD degree from SalaryGender CSV file

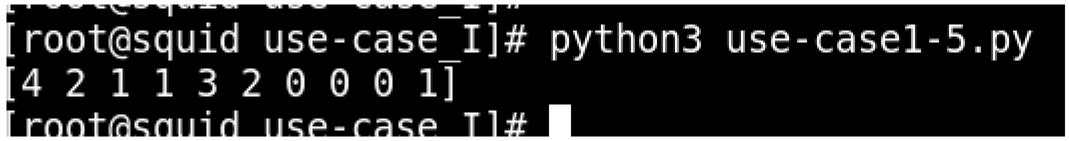
```
# Calculate the total number of people with a PhD
total_phd = len(df_with_phd)

# Print the total number
print("Total number of people with a PhD:", total_phd)
```



```
[root@squid use-case_I]# python3 use-case1-4.py
Total number of people with a PhD: 39
[root@squid use-case_I]#
```

5. How do you Count The Number Of Times Each Value Appears In An Array Of Integers?
[0, 5, 4, 0, 4, 4, 3, 0, 0, 5, 2, 1, 1, 9]
Answer should be array([4, 2, 1, 1, 3, 2, 0, 0, 0, 1]) which means 0 comes 4 times,
1 comes 2 times, 2 comes 1 time, 3 comes 1 time and so on.

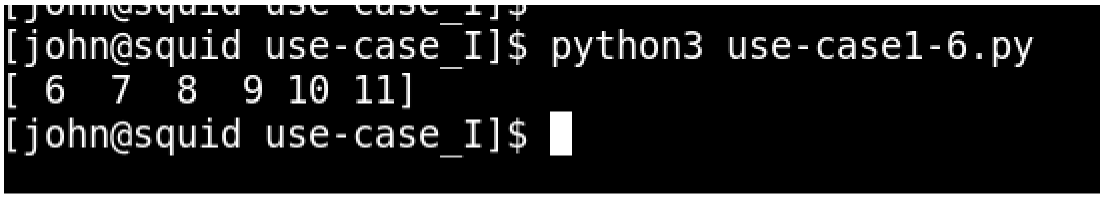
```
import numpy as np

# Given array of integers
arr = np.array([0, 5, 4, 0, 4, 4, 3, 0, 0, 5, 2, 1, 1, 9])

# Use np.bincount to count occurrences of each value
counts = np.bincount(arr)

print(counts)
```



```
[root@squid use-case_I]# python3 use-case1-5.py
[4 2 1 1 3 2 0 0 0 1]
[root@squid use-case_I]#
```

6. Create a numpy array [[0, 1, 2], [3, 4, 5], [6, 7, 8],[9, 10, 11]]) and filter the elements greater than 5.

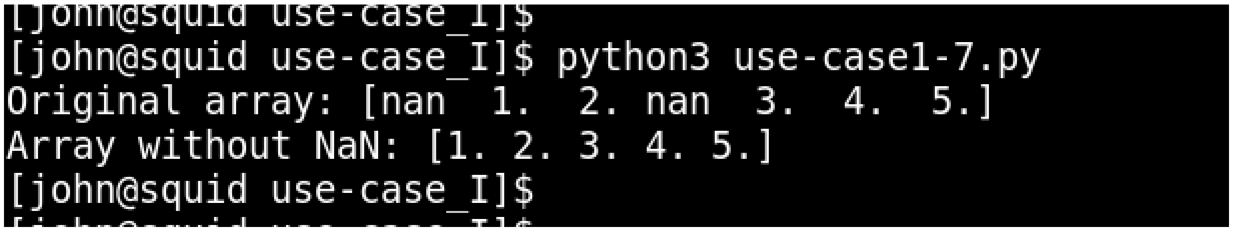
```
import numpy as np

# Create the numpy array
arr = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]])

# Filter elements greater than 5
filtered_arr = arr[arr > 5]

print(filtered_arr)
```



```
[john@squid use-case_I]$ python3 use-case1-6.py
[ 6  7  8  9 10 11]
[john@squid use-case_I]$
```

7. Create a numpy array having NaN (Not a Number) and print it. array([nan, 1., 2., nan, 3., 4., 5.]) Print the same array omitting all elements which are nan

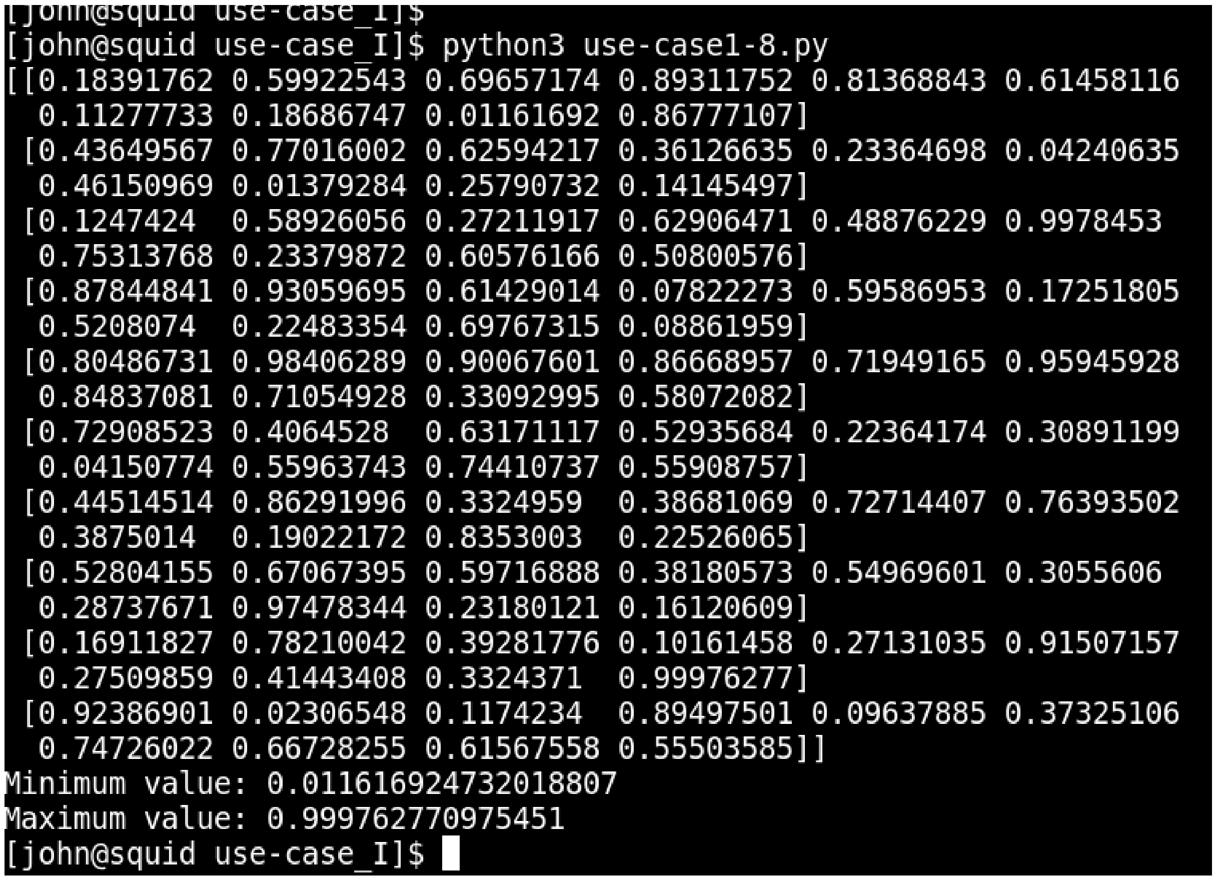
```
import numpy as np

# Create the numpy array with NaN values
arr = np.array([np.nan, 1., 2., np.nan, 3., 4., 5.])

# Print the original array
print("Original array:", arr)

# Omit NaN elements and print
filtered_arr = arr[~np.isnan(arr)]
print("Array without NaN:", filtered_arr)
```



```
[john@squid use-case_I]$
[john@squid use-case_I]$ python3 use-case1-7.py
Original array: [nan  1.  2. nan  3.  4.  5.]
Array without NaN: [1. 2. 3. 4. 5.]
[john@squid use-case_I]$
```

8. Create a 10x10 array with random values and find the minimum and maximum values.

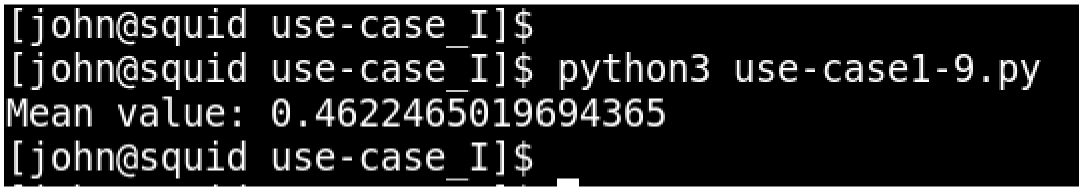
```
import numpy as np

# Create a 10x10 array with random values
arr = np.random.random((10, 10))

# Find the minimum and maximum values
min_val = arr.min()
max_val = arr.max()

print(arr)

print("Minimum value:", min_val)
print("Maximum value:", max_val)
```



```
[john@squid use-case_I]$
[john@squid use-case_I]$ python3 use-case1-8.py
[[0.18391762 0.59922543 0.69657174 0.89311752 0.81368843 0.61458116
  0.11277733 0.18686747 0.01161692 0.86777107]
 [0.43649567 0.77016002 0.62594217 0.36126635 0.23364698 0.04240635
  0.46150969 0.01379284 0.25790732 0.14145497]
 [0.1247424  0.58926056 0.27211917 0.62906471 0.48876229 0.9978453
  0.75313768 0.23379872 0.60576166 0.50800576]
 [0.87844841 0.93059695 0.61429014 0.07822273 0.59586953 0.17251805
  0.5208074  0.22483354 0.69767315 0.08861959]
 [0.80486731 0.98406289 0.90067601 0.86668957 0.71949165 0.95945928
  0.84837081 0.71054928 0.33092995 0.58072082]
 [0.72908523 0.4064528  0.63171117 0.52935684 0.22364174 0.30891199
  0.04150774 0.55963743 0.74410737 0.55908757]
 [0.44514514 0.86291996 0.3324959  0.38681069 0.72714407 0.76393502
  0.3875014  0.19022172 0.8353003  0.22526065]
 [0.52804155 0.67067395 0.59716888 0.38180573 0.54969601 0.3055606
  0.28737671 0.97478344 0.23180121 0.16120609]
 [0.16911827 0.78210042 0.39281776 0.10161458 0.27131035 0.91507157
  0.27509859 0.41443408 0.3324371  0.99976277]
 [0.92386901 0.02306548 0.1174234  0.89497501 0.09637885 0.37325106
  0.74726022 0.66728255 0.61567558 0.55503585]]
Minimum value: 0.011616924732018807
Maximum value: 0.999762770975451
[john@squid use-case_I]$
```

9. Create a random vector of size 30 and find the mean value.

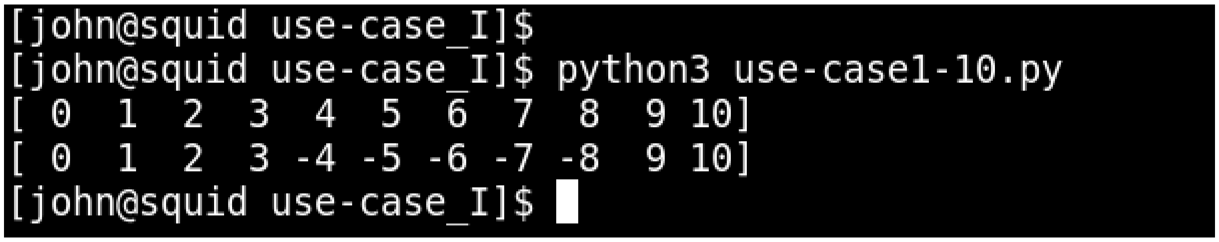
```
import numpy as np

# Create a random vector of size 30
vector = np.random.random(30)

# Find the mean value
mean_val = vector.mean()

print("Mean value:", mean_val)
```



```
[john@squid use-case_I]$
[john@squid use-case_I]$ python3 use-case1-9.py
Mean value: 0.4622465019694365
[john@squid use-case_I]$
```

10. Create numpy array having elements 0 to 10 And negate all the elements between 3 and 9

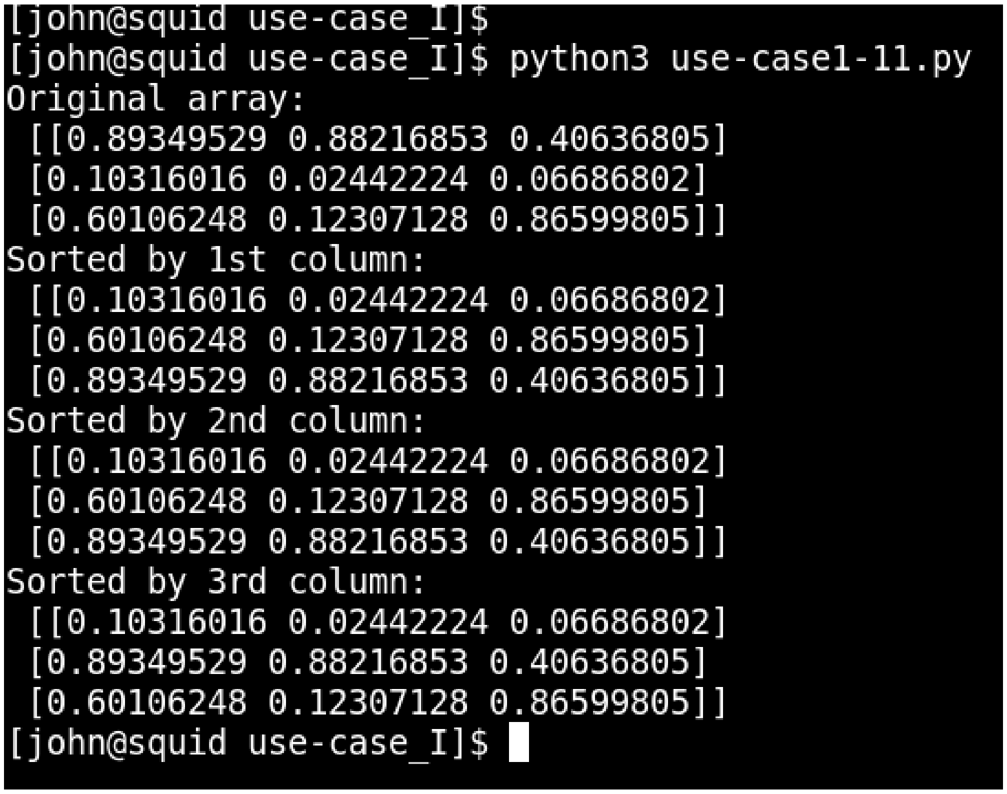
```
import numpy as np

# Create numpy array with elements 0 to 10
arr = np.arange(11)
print(arr)
# Negate all elements between 3 and 9
arr[(arr > 3) & (arr < 9)] *= -1

print(arr)
```



```
[john@squid use-case_I]$
[john@squid use-case_I]$ python3 use-case1-10.py
[ 0  1  2  3  4  5  6  7  8  9 10]
[ 0  1  2  3 -4 -5 -6 -7 -8  9 10]
[john@squid use-case_I]$
```

11. Create a random array of 3 rows and 3 columns and sort it according to 1 st column, 2 nd column or 3 rd column.

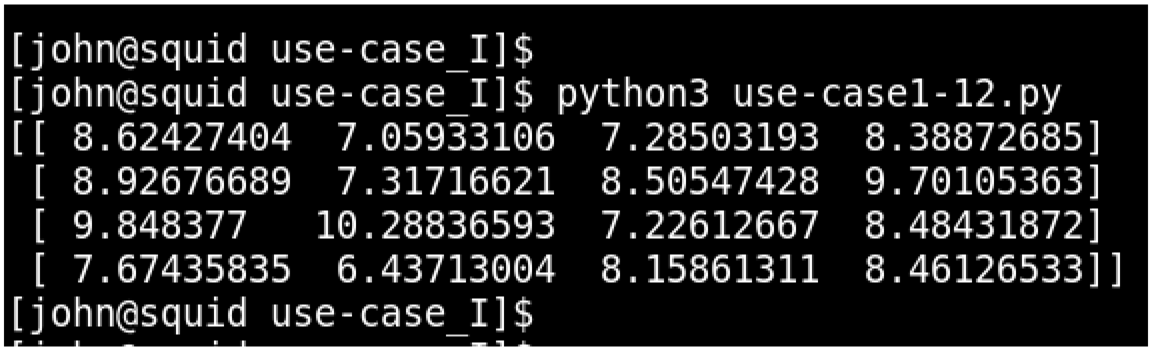
```
import numpy as np

# Create a random array of 3 rows and 3 columns
arr = np.random.random((3, 3))

# Sort according to the 1st column
arr_sorted_1st_col = arr[arr[:, 0].argsort()]

# Sort according to the 2nd column
arr_sorted_2nd_col = arr[arr[:, 1].argsort()]

# Sort according to the 3rd column
arr_sorted_3rd_col = arr[arr[:, 2].argsort()]

print("Original array:\n", arr)
print("Sorted by 1st column:\n", arr_sorted_1st_col)
print("Sorted by 2nd column:\n", arr_sorted_2nd_col)
print("Sorted by 3rd column:\n", arr_sorted_3rd_col)
```



```
[john@squid use-case_I]$
[john@squid use-case_I]$ python3 use-case1-11.py
Original array:
 [[0.89349529 0.88216853 0.40636805]
 [0.10316016 0.02442224 0.06686802]
 [0.60106248 0.12307128 0.86599805]]
Sorted by 1st column:
 [[0.10316016 0.02442224 0.06686802]
 [0.60106248 0.12307128 0.86599805]
 [0.89349529 0.88216853 0.40636805]]
Sorted by 2nd column:
 [[0.10316016 0.02442224 0.06686802]
 [0.60106248 0.12307128 0.86599805]
 [0.89349529 0.88216853 0.40636805]]
Sorted by 3rd column:
 [[0.10316016 0.02442224 0.06686802]
 [0.89349529 0.88216853 0.40636805]
 [0.60106248 0.12307128 0.86599805]]
[john@squid use-case_I]$
```

12. Create a four dimensions array get sum over the last two axis at once.

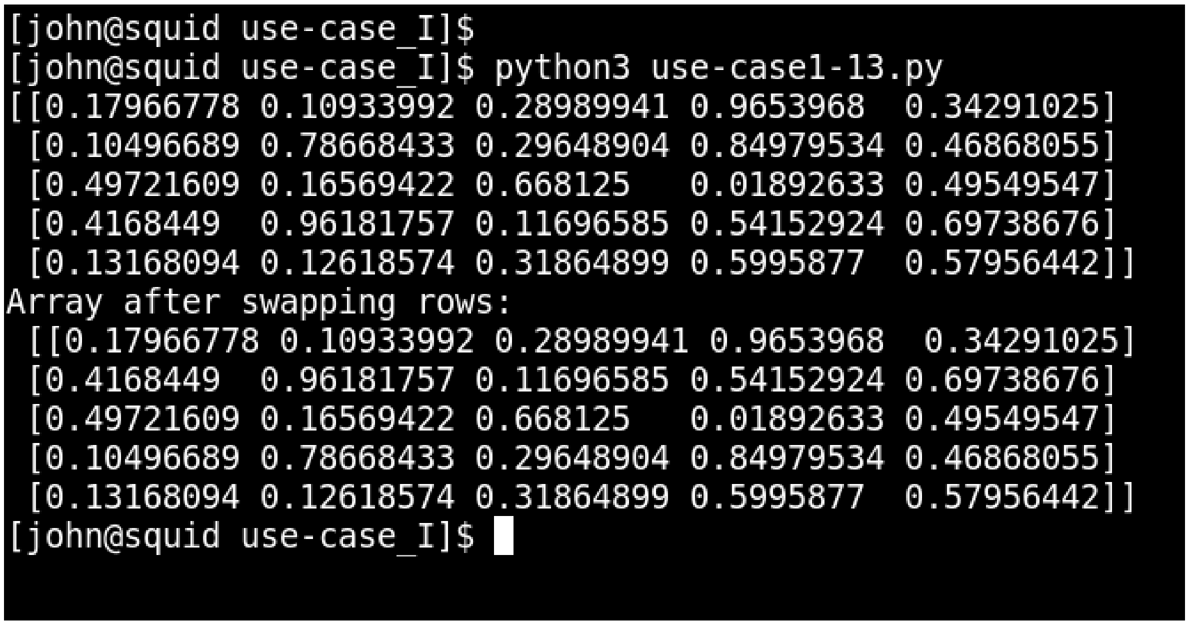
```
import numpy as np

# Create a four dimensions array
arr = np.random.rand(4, 4, 4, 4)
print(arr)
# Get sum over the last two axis at once
sum_over_last_two = arr.sum(axis=(-2, -1))

print(sum_over_last_two)
```



```
[john@squid use-case_I]$
[john@squid use-case_I]$ python3 use-case1-12.py
[[ 8.62427404  7.05933106  7.28503193  8.38872685]
 [ 8.92676689  7.31716621  8.50547428  9.70105363]
 [ 9.848377   10.28836593  7.22612667  8.48431872]
 [ 7.67435835  6.43713004  8.15861311  8.46126533]]
[john@squid use-case_I]$
```

13. Create a random array and swap two rows of an array.

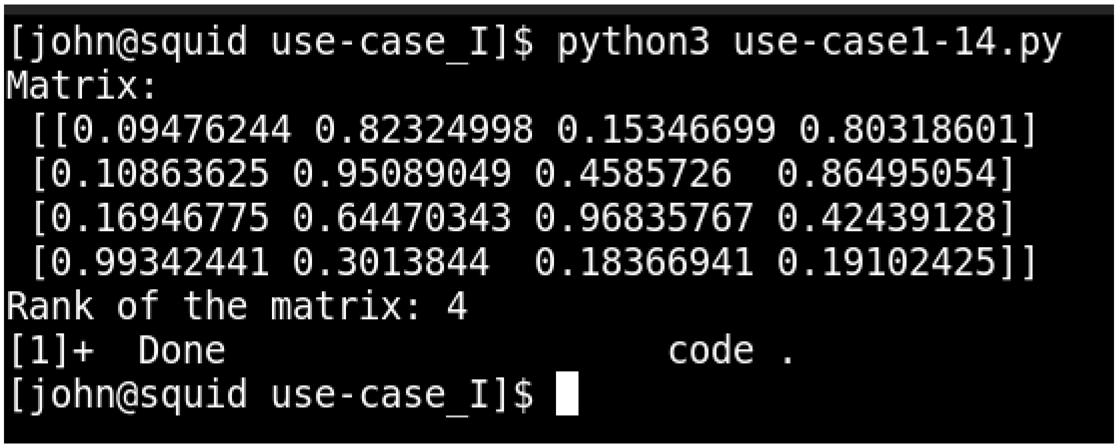
```
import numpy as np

# Create a random array of shape (5, 5) for demonstration
arr = np.random.rand(5, 5)

print(arr)

# Swap row 1 with row 3 (0-indexed)
arr[[1, 3]] = arr[[3, 1]]

print("Array after swapping rows:\n", arr)
```



```
[john@squid use-case_I]$
[john@squid use-case_I]$ python3 use-case1-13.py
[[0.17966778 0.10933992 0.28989941 0.9653968  0.34291025]
 [0.10496689 0.78668433 0.29648904 0.84979534 0.46868055]
 [0.49721609 0.16569422 0.668125   0.01892633 0.49549547]
 [0.4168449  0.96181757 0.11696585 0.54152924 0.69738676]
 [0.13168094 0.12618574 0.31864899 0.5995877  0.57956442]]
Array after swapping rows:
 [[0.17966778 0.10933992 0.28989941 0.9653968  0.34291025]
 [0.4168449  0.96181757 0.11696585 0.54152924 0.69738676]
 [0.49721609 0.16569422 0.668125   0.01892633 0.49549547]
 [0.10496689 0.78668433 0.29648904 0.84979534 0.46868055]
 [0.13168094 0.12618574 0.31864899 0.5995877  0.57956442]]
[john@squid use-case_I]$
```

14. Create a random matrix and Compute a matrix rank.

```
import numpy as np
from numpy.linalg import matrix_rank

# Create a random matrix
matrix = np.random.rand(4, 4)

# Compute the matrix rank
rank = matrix_rank(matrix)

print("Matrix:\n", matrix)
print("Rank of the matrix:", rank)
```



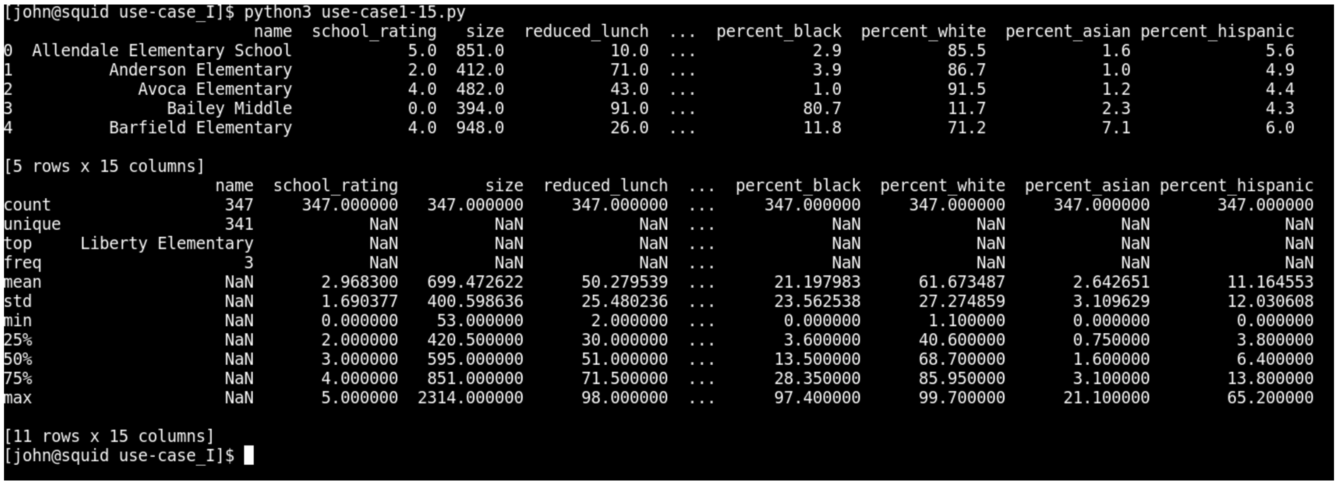
```
[john@squid use-case_I]$ python3 use-case1-14.py
Matrix:
 [[0.09476244 0.82324998 0.15346699 0.80318601]
 [0.10863625 0.95089049 0.4585726  0.86495054]
 [0.16946775 0.64470343 0.96835767 0.42439128]
 [0.99342441 0.3013844  0.18366941 0.19102425]]
Rank of the matrix: 4
[1]+  Done                    code .
[john@squid use-case_I]$
```

15 Analyse various school outcomes in Tennessee using pandas.

Let's start by loading the data into a pandas DataFrame and then exploring it to understand the details and statistics. We will also group the data by `school_rating` and describe the statistics for the `reduced_lunch` variable.

First, let's load the data and describe it, to find more details about it.

```
import pandas as pd

# Load the data
data = pd.read_csv('middle_tn_schools.csv')

# Display the first few rows of the data
print(data.head())

# Describe the data
data_description = data.describe(include='all')
print(data_description)
```



```
[john@squid use-case_I]$ python3 use-case1-15.py
                          name  school_rating   size  reduced_lunch  ...  percent_black  percent_white  percent_asian percent_hispanic
0  Allendale Elementary School            5.0  851.0           10.0  ...            2.9           85.5            1.6              5.6
1          Anderson Elementary            2.0  412.0           71.0  ...            3.9           86.7            1.0              4.9
2             Avoca Elementary            4.0  482.0           43.0  ...            1.0           91.5            1.2              4.4
3                Bailey Middle            0.0  394.0           91.0  ...           80.7           11.7            2.3              4.3
4          Barfield Elementary            4.0  948.0           26.0  ...           11.8           71.2            7.1              6.0

[5 rows x 15 columns]
                      name  school_rating         size  reduced_lunch  ...  percent_black  percent_white  percent_asian percent_hispanic
count                  347     347.000000   347.000000     347.000000  ...     347.000000     347.000000     347.000000       347.000000
unique                 341            NaN          NaN            NaN  ...            NaN            NaN            NaN              NaN
top     Liberty Elementary            NaN          NaN            NaN  ...            NaN            NaN            NaN              NaN
freq                     3            NaN          NaN            NaN  ...            NaN            NaN            NaN              NaN
mean                   NaN       2.968300   699.472622      50.279539  ...      21.197983      61.673487       2.642651        11.164553
std                    NaN       1.690377   400.598636      25.480236  ...      23.562538      27.274859       3.109629        12.030608
min                    NaN       0.000000    53.000000       2.000000  ...       0.000000       1.100000       0.000000         0.000000
25%                    NaN       2.000000   420.500000      30.000000  ...       3.600000      40.600000       0.750000         3.800000
50%                    NaN       3.000000   595.000000      51.000000  ...      13.500000      68.700000       1.600000         6.400000
75%                    NaN       4.000000   851.000000      71.500000  ...      28.350000      85.950000       3.100000        13.800000
max                    NaN       5.000000  2314.000000      98.000000  ...      97.400000      99.700000      21.100000        65.200000

[11 rows x 15 columns]
[john@squid use-case_I]$ █
```

Initial Data Analysis

The data from Middle Tennessee schools has been successfully loaded and analyzed. Here are the key components of the initial exploratory analysis:

Data Overview

The dataset includes 347 schools with various attributes such as school ratings, size, percentage of students on reduced lunch, state percentiles, student-teacher ratios, school type, average scores, and demographic information.

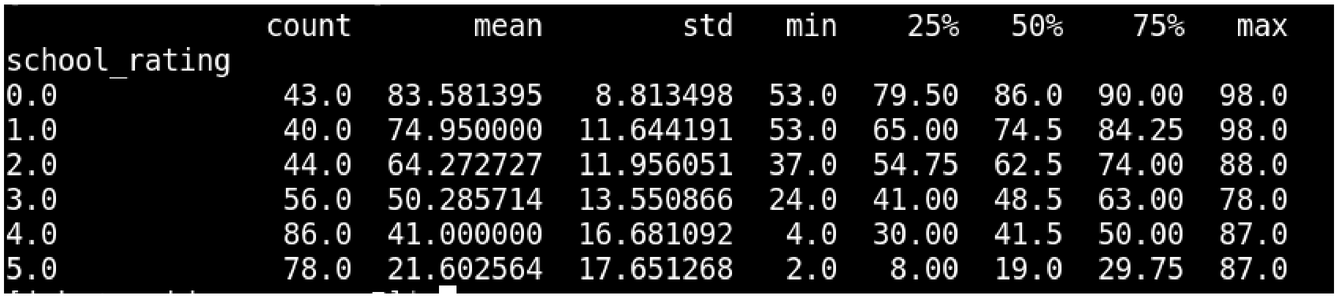
Descriptive Statistics

The descriptive statistics provide insights into the central tendencies, dispersions, and distributions of the data:

- **School Rating:** Ranges from 0 to 5, with a mean of approximately 3.
- **Size:** Schools vary significantly in size, from as few as 53 students to as many as 2,314.
- **Reduced Lunch:** A measure of the socio-economic status of students, with values ranging from 2% to 98%.
- **State Percentiles:** Indicate the school's performance relative to others in the state.
- **Student-Teacher Ratio:** Varies from 11.6 to 18.7.
- **Demographic Data:** Percentages of different ethnic groups within the schools

Now, we will group the data by `school_rating` and describe the `reduced_lunch` variable.

```
import pandas as pd

# Load the data
data = pd.read_csv('middle_tn_schools.csv')

# Display the first few rows of the data
print(data.head())

# Describe the data
data_description = data.describe(include='all')
print(data_description)


# Group data by school_rating and describe the reduced_lunch variable
grouped_data = data.groupby('school_rating')['reduced_lunch'].describe()
print(grouped_data)
```



```
               count       mean        std   min    25%   50%    75%   max
school_rating
0.0             43.0  83.581395   8.813498  53.0  79.50  86.0  90.00  98.0
1.0             40.0  74.950000  11.644191  53.0  65.00  74.5  84.25  98.0
2.0             44.0  64.272727  11.956051  37.0  54.75  62.5  74.00  88.0
3.0             56.0  50.285714  13.550866  24.0  41.00  48.5  63.00  78.0
4.0             86.0  41.000000  16.681092   4.0  30.00  41.5  50.00  87.0
5.0             78.0  21.602564  17.651268   2.0   8.00  19.0  29.75  87.0
```

Grouped Data by School Rating

The `reduced_lunch` variable, which is a proxy for household income, was analyzed by grouping schools based on their ratings:

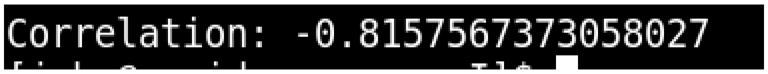
- **School Rating 0:** High mean reduced lunch percentage (83.58%), indicating a high proportion of students from low-income households.
- **School Rating 1:** Slightly lower mean reduced lunch percentage (74.95%).
- **School Rating 2:** Reduced lunch percentage decreases further (64.27%).
- **School Rating 3:** Mean reduced lunch percentage (50.29%).
- **School Rating 4:** Even lower mean reduced lunch percentage (41.00%).
- **School Rating 5:** Lowest mean reduced lunch percentage (21.60%).

This analysis shows a clear trend where schools with higher ratings tend to have a lower percentage of students on reduced lunch, suggesting a correlation between household income and school performance.

Phase 3 – Correlation Analysis

First, let's calculate the correlation between `reduced_lunch` and `school_rating`.

```
# Phase 3 - Correlation Analysis
correlation = data['reduced_lunch'].corr(data['school_rating'])
print(f"Correlation: {correlation}")
```



```
Correlation: -0.8157567373058027
```

The computed correlation value is approximately -0.82, indicating a strong negative correlation between `reduced_lunch` and `school_rating`. This means that as the percentage of students on reduced lunch increases, the school rating tends to decrease.

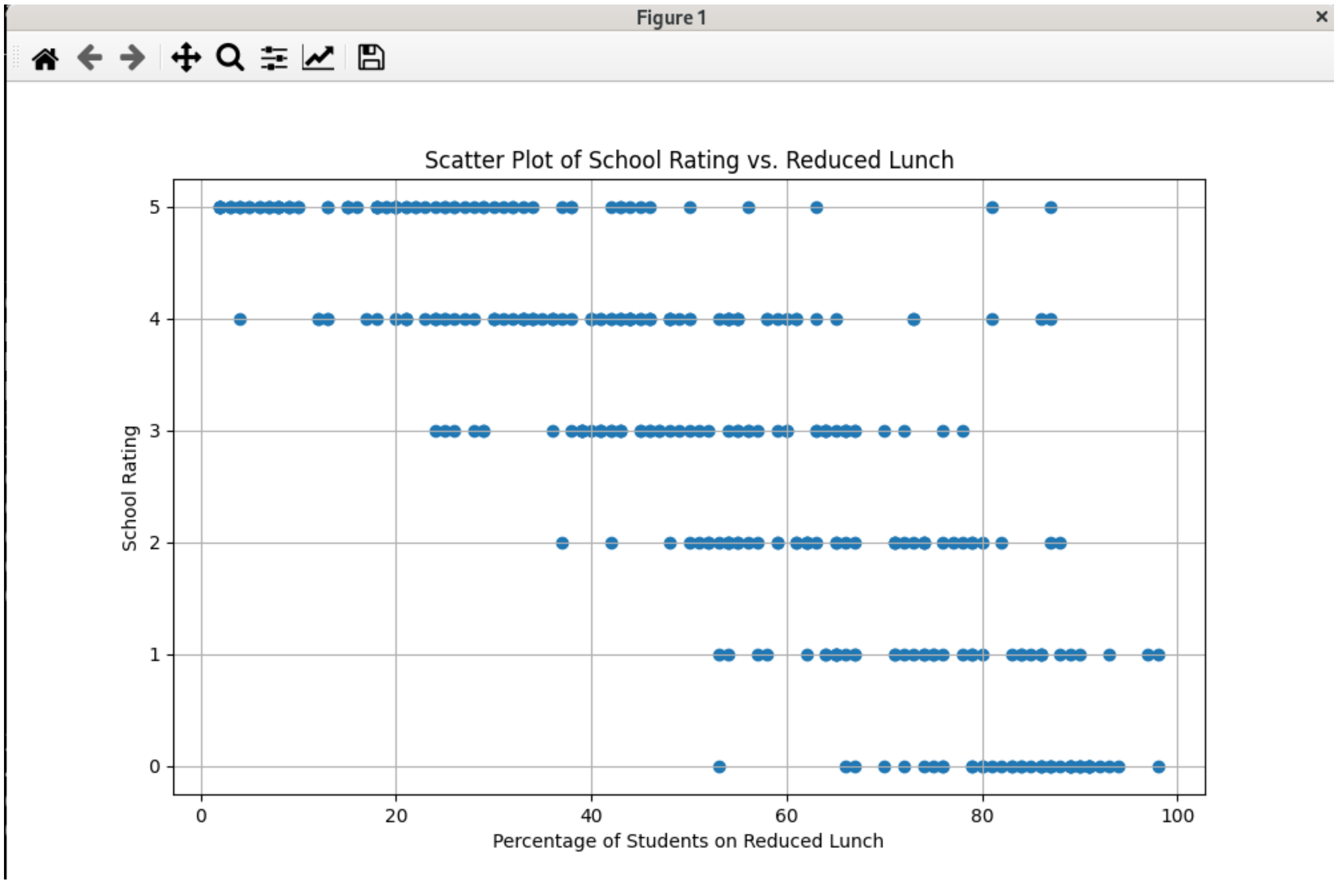
Phase 4 – Scatter Plot

Next, we will create a scatter plot to visualize the relationship between `reduced_lunch` and `school_rating`.

```
# Phase 4 - Scatter Plot
plt.figure(figsize=(10, 6))
plt.scatter(data['reduced_lunch'], data['school_rating'])
plt.title('Scatter Plot of School Rating vs. Reduced Lunch')
plt.xlabel('Percentage of Students on Reduced Lunch')
plt.ylabel('School Rating')
plt.grid(True)
plt.show()
```

The scatter plot shows the relationship between `school_rating` and `reduced_lunch`. Each dot represents a school, with the x-axis showing the percentage of students on reduced lunch and the y-axis showing the school rating.

The scatter plot is displayed below:



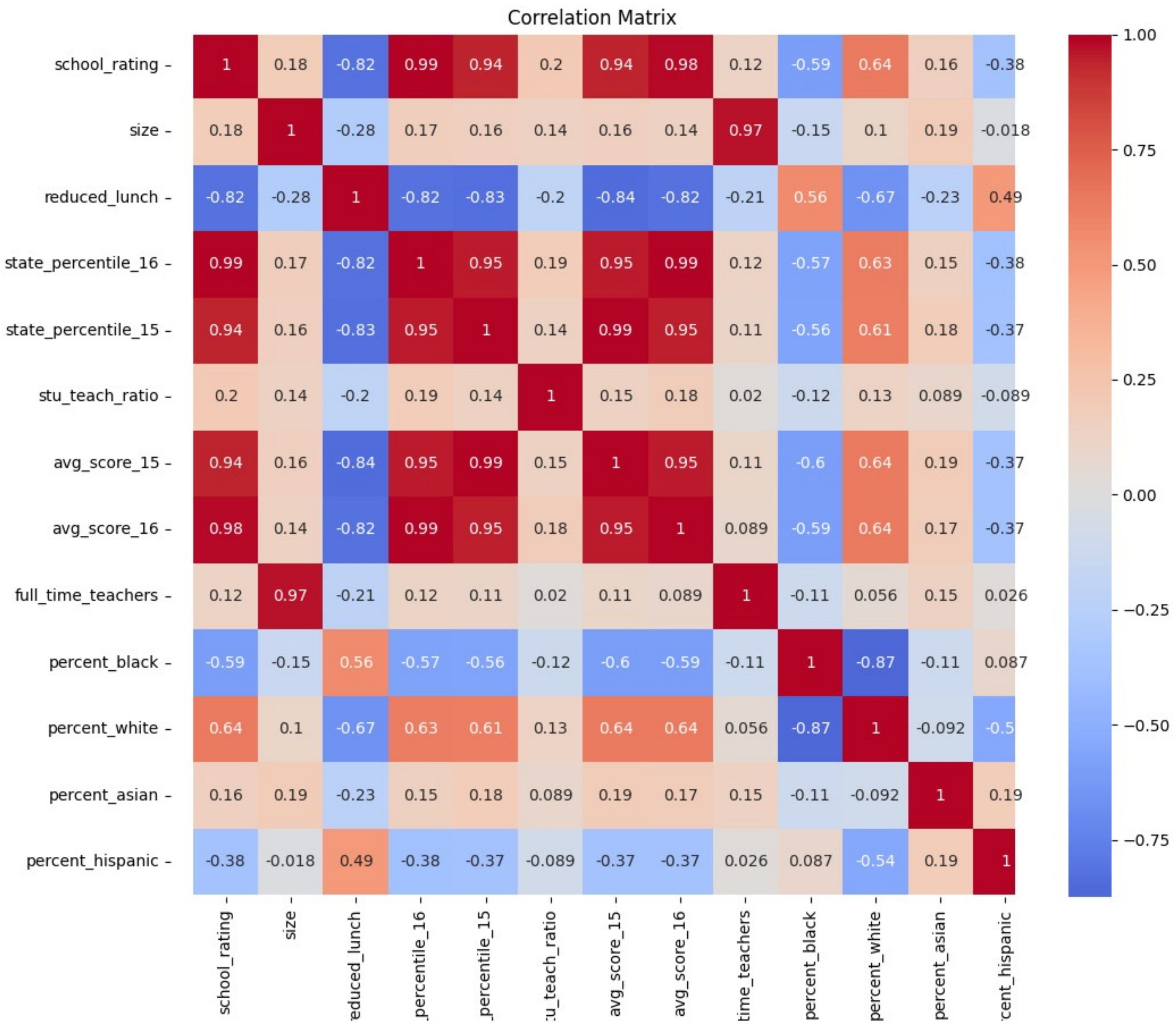
Phase 5 – Correlation Matrix

Finally, we'll create a correlation matrix to visualize the correlations between various variables in the dataset.

```
# Phase 5 - Correlation Matrix
# Select only numeric columns
numeric_data = data.select_dtypes(include=[float, int])

# Compute the correlation matrix
plt.figure(figsize=(12, 10))
correlation_matrix = numeric_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```

The correlation matrix visualizes the correlations between various variables in the dataset. The heatmap provides a color-coded view of the correlations, with red indicating positive correlation, blue indicating negative correlation, and white indicating no correlation. The intensity of the colors reflects the strength of the correlations.

The correlation matrix is displayed below:



	school_rating	size	educed_lunch	percentile_16	percentile_15	u_teach_ratio	avg_score_15	avg_score_16	time_teachers	percent_black	percent_white	percent_asian	cent_hispanic
school_rating	1	0.18	-0.82	0.99	0.94	0.2	0.94	0.98	0.12	-0.59	0.64	0.16	-0.38
size	0.18	1	-0.28	0.17	0.16	0.14	0.16	0.14	0.97	-0.15	0.1	0.19	-0.018
reduced_lunch	-0.82	-0.28	1	-0.82	-0.83	-0.2	-0.84	-0.82	-0.21	0.56	-0.67	-0.23	0.49
state_percentile_16	0.99	0.17	-0.82	1	0.95	0.19	0.95	0.99	0.12	-0.57	0.63	0.15	-0.38
state_percentile_15	0.94	0.16	-0.83	0.95	1	0.14	0.99	0.95	0.11	-0.56	0.61	0.18	-0.37
stu_teach_ratio	0.2	0.14	-0.2	0.19	0.14	1	0.15	0.18	0.02	-0.12	0.13	0.089	-0.089
avg_score_15	0.94	0.16	-0.84	0.95	0.99	0.15	1	0.95	0.11	-0.6	0.64	0.19	-0.37
avg_score_16	0.98	0.14	-0.82	0.99	0.95	0.18	0.95	1	0.089	-0.59	0.64	0.17	-0.37
full_time_teachers	0.12	0.97	-0.21	0.12	0.11	0.02	0.11	0.089	1	-0.11	0.056	0.15	0.026
percent_black	-0.59	-0.15	0.56	-0.57	-0.56	-0.12	-0.6	-0.59	-0.11	1	-0.87	-0.11	0.087
percent_white	0.64	0.1	-0.67	0.63	0.61	0.13	0.64	0.64	0.056	-0.87	1	-0.092	-0.5
percent_asian	0.16	0.19	-0.23	0.15	0.18	0.089	0.19	0.17	0.15	-0.11	-0.092	1	0.19
percent_hispanic	-0.38	-0.018	0.49	-0.38	-0.37	-0.089	-0.37	-0.37	0.026	0.087	-0.54	0.19	1

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
data = pd.read_csv('middle_tn_schools.csv')

# Display the first few rows of the data
print(data.head())

# Describe the data
data_description = data.describe(include='all')
print(data_description)

# Group data by school_rating and describe the reduced_lunch variable
grouped_data = data.groupby('school_rating')['reduced_lunch'].describe()
print(grouped_data)

# Phase 3 - Correlation Analysis
correlation = data['reduced_lunch'].corr(data['school_rating'])
print(f"Correlation: {correlation}")

# Phase 4 - Scatter Plot
plt.figure(figsize=(10, 6))
plt.scatter(data['reduced_lunch'], data['school_rating'])
plt.title('Scatter Plot of School Rating vs. Reduced Lunch')
plt.xlabel('Percentage of Students on Reduced Lunch')
plt.ylabel('School Rating')
plt.grid(True)
plt.show()

# Phase 5 - Correlation Matrix
# Select only numeric columns
numeric_data = data.select_dtypes(include=[float, int])

# Compute the correlation matrix
plt.figure(figsize=(12, 10))
correlation_matrix = numeric_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```