mod4_assigment-uc1

October 23, 2024

1 Module4: Numpy, Pandas, Matplotlib

1.1 Assignment: Use-Case I

[1]:
```
#1
import pandas as pd
import numpy as np


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

# Print basic information about the data (Optional)
print('DATA INFO: ')
print(data.info())
print('\nDATA HEAD: ')
print(data.head())

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

# Extract each column into a separate NumPy array
salary = np.array(data['Salary'])
gender = np.array(data['Gender'])
age = np.array(data['Age'])
phd = np.array(data['PhD'])

print("Salary Array: ", salary)
print("Gender Array: ", gender)
print("Age Array: ", age)
print("PhD Array: ", phd)
```

```
DATA INFO:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
```

```
 0   Salary  100 non-null    float64
 1   Gender  100 non-null    int64
 2   Age     100 non-null    int64
 3   PhD     100 non-null    int64
dtypes: float64(1), int64(3)
memory usage: 3.2 KB
None

DATA HEAD:
   Salary  Gender  Age  PhD
0   140.0       1   47    1
1    30.0       0   65    1
2    35.1       0   56    0
3    30.0       1   23    0
4    80.0       0   53    1
Salary Array:  [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.  ]
Gender Array:  [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]
Age Array:  [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]
PhD Array:  [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]
```

[30]:
```
#2 Find:
# The number of men with a PhD
# The number of women with a PhD

# Men with a PhD
men_with_phd = np.sum((gender == 1) & (phd == 1))
```

```
print("Number of men with a PhD:", men_with_phd)

# Women with a PhD
women_with_phd = np.sum((gender == 0) & (phd == 1))
print("Number of women with a PhD:", women_with_phd)
```

```
Number of men with a PhD: 24
Number of women with a PhD: 15
```

[36]:
```
#3 Store the "Age" and "Ph.D." columns in one DataFrame and delete the data of␣
 ↪all people who don't have a PhD

# Create a new DataFrame with 'Age' and 'PhD' columns
age_phd_df = data[['Age', 'PhD']]

# Filter out people who don't have a PhD
age_phd_with_phd = age_phd_df[age_phd_df['PhD'] == 1]
print(age_phd_with_phd)
age_phd_with_phd.shape
```

```
    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
```

[36]:
```
(39, 2)
```

[32]:
```
#4 Calculate the total number of people who have a PhD degree from the␣
 ↪SalaryGender CSV file
# Total number of people with a PhD
total_with_phd = np.sum(phd == 1)
print("Total number of people with a PhD:", total_with_phd)
```

```
Total number of people with a PhD: 39
```

[43]:
```
#5 Count The Number Of Times Each Value Appears In An Array Of Integers Input:␣
 ↪[0, 5, 4, 0, 4, 4, 3, 0, 0, 5, 2, 1, 1, 9] Expected Output: array([4, 2, 1,␣
 ↪1, 3, 2, 0, 0, 0, 1])
arr = np.array([0, 5, 4, 0, 4, 4, 3, 0, 0, 5, 2, 1, 1, 9])

# Count the occurrences of each number from 0 to 9
counts = np.bincount(arr, minlength=10)

print("Counts: ", counts)
#set(sorted(arr))
np.arange(10)
```

```
Counts:  [4 2 1 1 3 2 0 0 0 1]
```

[43]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[47]:
```
6# Create a NumPy array [[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]] and␣
 ↪filter the elements greater than 5
# Create the array
arr = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]])

# Filter elements greater than 5
print(arr>5)
filtered_arr = arr[arr > 5]
print("Elements greater than 5: ", filtered_arr)
```

```
[[False False False]
 [False False False]
 [ True  True  True]
 [ True  True  True]]
Elements greater than 5:  [ 6  7  8  9 10 11]
```

[47]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

[50]:
```
#7 Create a NumPy array having NaN (Not a Number) and print it. Then print the same array omitting all elements which are NaN
# Create a NumPy array with NaN values
arr_with_nan = np.array([np.nan, 1, 2, np.nan, 3, 4, 5])

# Print the array with NaN
print("Array with NaN: ", arr_with_nan)

# Print the array omitting NaN values
print(np.isnan(arr_with_nan))
print(~np.isnan(arr_with_nan))
arr_without_nan = arr_with_nan[~np.isnan(arr_with_nan)]
print("Array without NaN: ", arr_without_nan)
```

```
Array with NaN:  [nan  1.  2. nan  3.  4.  5.]
[ True False False  True False False False]
[False  True  True False  True  True  True]
Array without NaN:  [1. 2. 3. 4. 5.]
```

[53]:
```
#8 Create a 10x10 array with random values and find the minimum and maximum value
# Create a 10x10 array with random values
random_array = np.random.random((10, 10))
print("10x10 array with random values", random_array)
# Find the minimum and maximum values
min_value = random_array.min()
max_value = random_array.max()

print("Minimum value: ", min_value)
print("Maximum value: ", max_value)
```

```
10x10 array with random values [[0.9065555  0.77404733 0.33314515 0.08110139
0.40724117 0.23223414
  0.13248763 0.05342718 0.72559436 0.01142746]
 [0.77058075 0.14694665 0.07952208 0.08960303 0.67204781 0.24536721
  0.42053947 0.55736879 0.86055117 0.72704426]
 [0.27032791 0.1314828  0.05537432 0.30159863 0.26211815 0.45614057
  0.68328134 0.69562545 0.28351885 0.37992696]
```

```
 [0.18115096 0.78854551 0.05684808 0.69699724 0.7786954  0.77740756
  0.25942256 0.37381314 0.58759964 0.2728219 ]
 [0.3708528  0.19705428 0.45985588 0.0446123  0.79979588 0.07695645
  0.51883515 0.3068101  0.57754295 0.95943334]
 [0.64557024 0.03536244 0.43040244 0.51001685 0.53617749 0.68139251
  0.2775961  0.12886057 0.39267568 0.95640572]
 [0.18713089 0.90398395 0.54380595 0.45691142 0.88204141 0.45860396
  0.72416764 0.39902532 0.90404439 0.69002502]
 [0.69962205 0.3277204  0.75677864 0.63606106 0.24002027 0.16053882
  0.79639147 0.9591666  0.45813883 0.59098417]
 [0.85772264 0.45722345 0.95187448 0.57575116 0.82076712 0.90884372
  0.81552382 0.15941446 0.62889844 0.39843426]
 [0.06271295 0.42403225 0.25868407 0.84903831 0.03330463 0.95898272
  0.35536885 0.35670689 0.0163285  0.18523233]]
Minimum value:  0.011427458625031028
Maximum value:  0.9594333408334251
```

[55]:
```
#9 Create a random vector of size 30 and find the mean value
# Create a random vector of size 30
random_vector = np.random.random(30)
print("random vector of size 30: ", random_vector)

# Find the mean value
mean_value = np.mean(random_vector)
print("Mean value: ", mean_value)
```

```
random vector of size 30:  [0.24141862 0.66250457 0.24606318 0.66585912
0.51730852 0.42408899
 0.55468781 0.28705152 0.70657471 0.41485687 0.36054556 0.82865691
 0.92496691 0.04600731 0.23262699 0.34851937 0.81496648 0.98549143
 0.9689717  0.90494835 0.29655627 0.99201124 0.24942004 0.10590615
 0.95095261 0.23342026 0.68976827 0.05835636 0.7307091  0.88172021]
Mean value:  0.5441645142627912
```

[56]:
```
#10 Create a NumPy array having elements 0 to 10 and negate all the elements␣
 ↪between 3 and 9
# Create a NumPy array with elements from 0 to 10
arr = np.arange(11)

# Negate all the elements between 3 and 9
arr[(arr >= 3) & (arr <= 9)] *= -1

print("Modified array: ", arr)
```

```
Modified array:  [ 0  1  2 -3 -4 -5 -6 -7 -8 -9 10]
```

[59]:
```
#11 Create a random array of 3 rows and 3 columns and sort it according to 1st␣
 ↪column, 2nd column, or 3rd column
```

```
# Create a random 3x3 array
random_array = np.random.rand(3, 3)

print("Original Array:\n", random_array)

# Sort according to the 1st column
print(random_array[:, 0])
print(random_array[:, 0].argsort())
sorted_by_first = random_array[random_array[:, 0].argsort()]
print("Sorted by 1st column:\n", sorted_by_first)

# Sort according to the 2nd column
sorted_by_second = random_array[random_array[:, 1].argsort()]
print("Sorted by 2nd column:\n", sorted_by_second)

# Sort according to the 3rd column
sorted_by_third = random_array[random_array[:, 2].argsort()]
print("Sorted by 3rd column:\n", sorted_by_third)
```

```
Original Array:
 [[0.99440079 0.45182168 0.07086978]
 [0.29279403 0.15235471 0.41748637]
 [0.13128933 0.6041178  0.38280806]]
[0.99440079 0.29279403 0.13128933]
[2 1 0]
Sorted by 1st column:
 [[0.13128933 0.6041178  0.38280806]
 [0.29279403 0.15235471 0.41748637]
 [0.99440079 0.45182168 0.07086978]]
Sorted by 2nd column:
 [[0.29279403 0.15235471 0.41748637]
 [0.99440079 0.45182168 0.07086978]
 [0.13128933 0.6041178  0.38280806]]
Sorted by 3rd column:
 [[0.99440079 0.45182168 0.07086978]
 [0.13128933 0.6041178  0.38280806]
 [0.29279403 0.15235471 0.41748637]]
```

[61]:
```
#12 Create a four-dimensional array and get the sum over the last two axes at␣
 ↪once
# Create a 4D array
arr_4d = np.random.rand(3, 3, 3, 3)
print(arr_4d.shape)

# Sum over the last two axes (axis -2 and axis -1)
sum_over_last_two_axes = arr_4d.sum(axis=(-2, -1))
```

```
print("4D array:\n", arr_4d)
print("Sum over the last two axes:\n", sum_over_last_two_axes)
```

```
(3, 3, 3, 3)
4D array:
 [[[[0.82989737 0.96828641 0.91978281]
    [0.03603382 0.174772   0.38913468]
    [0.9521427  0.30002892 0.16046764]]

   [[0.88630467 0.44639442 0.90787559]
    [0.16023047 0.66111751 0.44026375]
    [0.07648677 0.69646314 0.24739876]]

   [[0.03961552 0.0599443  0.06107854]
    [0.90773296 0.73988392 0.89806236]
    [0.67258231 0.52893993 0.30444636]]]


  [[[0.99796225 0.36218906 0.47064895]
    [0.37824517 0.97952693 0.17465839]
    [0.327988   0.68034867 0.06320762]]

   [[0.60724937 0.4776465  0.28399998]
    [0.23841328 0.51451274 0.36792758]
    [0.45651989 0.33747738 0.97049369]]

   [[0.13343943 0.09680395 0.34339173]
    [0.5910269  0.65917647 0.39725675]
    [0.99927799 0.351893   0.72140667]]]


  [[[0.63758269 0.81305386 0.97622566]
    [0.88979366 0.76456197 0.69824848]
    [0.33549817 0.14768558 0.062636  ]]

   [[0.2419017  0.43228148 0.52199627]
    [0.77308355 0.95874092 0.11732048]
    [0.10700414 0.58969472 0.74539807]]

   [[0.84815038 0.93583208 0.98342624]
    [0.39980169 0.38033518 0.14780868]
    [0.68493444 0.65676196 0.8620626 ]]]]
Sum over the last two axes:
 [[4.73054635 4.52253508 4.2122862 ]
 [4.43477503 4.25424043 4.29367289]
 [5.32528608 4.48742135 5.89911325]]
```

[4]:
```
#13 Create a random array and swap two rows of an array
# Create a random 5x5 array
arr = np.random.rand(5, 5)

print("Original array:\n", arr)

print(arr[[0,2]])
# Swap the first and third row
arr[[0, 2]] = arr[[2, 0]]

print("Array after swapping first and third row:\n", arr)
```

```
Original array:
 [[1.89152073e-01 1.96626252e-01 3.65948707e-01 8.76776091e-01
  7.76790978e-02]
 [7.52615299e-01 1.78932447e-01 9.94956263e-01 1.03324310e-01
  5.85217421e-01]
 [6.64687215e-01 5.87745711e-01 9.59305608e-01 2.85732083e-02
  8.15608816e-01]
 [4.08820235e-01 4.18236707e-01 3.89478732e-01 1.35041905e-01
  9.14065228e-01]
 [4.89470263e-01 1.91618268e-01 8.80028242e-01 3.12385668e-04
  9.34146003e-01]]
[[0.18915207 0.19662625 0.36594871 0.87677609 0.0776791 ]
 [0.66468722 0.58774571 0.95930561 0.02857321 0.81560882]]
Array after swapping first and third row:
 [[6.64687215e-01 5.87745711e-01 9.59305608e-01 2.85732083e-02
  8.15608816e-01]
 [7.52615299e-01 1.78932447e-01 9.94956263e-01 1.03324310e-01
  5.85217421e-01]
 [1.89152073e-01 1.96626252e-01 3.65948707e-01 8.76776091e-01
  7.76790978e-02]
 [4.08820235e-01 4.18236707e-01 3.89478732e-01 1.35041905e-01
  9.14065228e-01]
 [4.89470263e-01 1.91618268e-01 8.80028242e-01 3.12385668e-04
  9.34146003e-01]]
```

[63]:
```
#14 Create a random matrix and compute a matrix rank
# Create a random matrix (5x5)
matrix = np.random.rand(5, 5)

print("Matrix:\n", matrix)

# Compute the rank of the matrix
matrix_rank = np.linalg.matrix_rank(matrix)

print("Rank of the matrix: ", matrix_rank)
```

```
Matrix:
 [[0.82240674 0.65342116 0.72634246 0.536923   0.11047711]
 [0.40503561 0.40537358 0.32104299 0.02995032 0.73725424]
 [0.10978446 0.60630813 0.7032175  0.63478632 0.95914225]
 [0.10329816 0.86716716 0.02919023 0.53491685 0.40424362]
 [0.52418386 0.36509988 0.19056691 0.0191229  0.51814981]]
Rank of the matrix:  5
```

[26]:
```
#15
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv('middle_tn_schools.csv')
data.info()
print( data.describe() )
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 347 entries, 0 to 346
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   name                 347 non-null    object
 1   school_rating        347 non-null    float64
 2   size                 347 non-null    float64
 3   reduced_lunch        347 non-null    float64
 4   state_percentile_16  347 non-null    float64
 5   state_percentile_15  341 non-null    float64
 6   stu_teach_ratio      347 non-null    float64
 7   school_type          347 non-null    object
 8   avg_score_15         341 non-null    float64
 9   avg_score_16         347 non-null    float64
 10  full_time_teachers   347 non-null    float64
 11  percent_black        347 non-null    float64
 12  percent_white        347 non-null    float64
 13  percent_asian        347 non-null    float64
 14  percent_hispanic     347 non-null    float64
dtypes: float64(13), object(2)
memory usage: 40.8+ KB
       school_rating         size  reduced_lunch  state_percentile_16  \
count     347.000000   347.000000     347.000000           347.000000
mean        2.968300   699.472622      50.279539            58.801729
std         1.690377   400.598636      25.480236            32.540747
min         0.000000    53.000000       2.000000             0.200000
25%         2.000000   420.500000      30.000000            30.950000
50%         3.000000   595.000000      51.000000            66.400000
75%         4.000000   851.000000      71.500000            88.000000
```

```
max          5.000000  2314.000000      98.000000            99.800000
```

```
       state_percentile_15  stu_teach_ratio  avg_score_15  avg_score_16  \
count           341.000000       347.000000    341.000000    347.000000
mean             58.249267        15.461671     57.004692     57.049856
std              32.702630         5.725170     26.696450     27.968974
min               0.600000         4.700000      1.500000      0.100000
25%              27.100000        13.700000     37.600000     37.000000
50%              65.800000        15.000000     61.800000     60.700000
75%              88.600000        16.700000     79.600000     80.250000
max              99.800000       111.000000     99.000000     98.900000

       full_time_teachers  percent_black  percent_white  percent_asian  \
count          347.000000     347.000000     347.000000     347.000000
mean            44.939481      21.197983      61.673487       2.642651
std             22.053386      23.562538      27.274859       3.109629
min              2.000000       0.000000       1.100000       0.000000
25%             30.000000       3.600000      40.600000       0.750000
50%             40.000000      13.500000      68.700000       1.600000
75%             54.000000      28.350000      85.950000       3.100000
max            140.000000      97.400000      99.700000      21.100000

       percent_hispanic
count        347.000000
mean          11.164553
std           12.030608
min            0.000000
25%            3.800000
50%            6.400000
75%           13.800000
max           65.200000
```

[26]:

```
                        name  school_rating   size  reduced_lunch  \
0  Allendale Elementary School            5.0  851.0           10.0
1          Anderson Elementary            2.0  412.0           71.0
2             Avoca Elementary            4.0  482.0           43.0
3                Bailey Middle            0.0  394.0           91.0
4          Barfield Elementary            4.0  948.0           26.0

   state_percentile_16  state_percentile_15  stu_teach_ratio    school_type  \
0                 90.2                 95.8             15.7         Public
1                 32.8                 37.3             12.8         Public
2                 78.4                 83.6             16.6         Public
3                  1.6                  1.0             13.1  Public Magnet
4                 85.3                 89.2             14.8         Public

   avg_score_15  avg_score_16  full_time_teachers  percent_black  \
```

```
0          89.4          85.2                54.0           2.9
1          43.0          38.3                32.0           3.9
2          75.7          73.0                29.0           1.0
3           2.1           4.4                30.0          80.7
4          81.3          79.6                64.0          11.8

   percent_white  percent_asian  percent_hispanic
0           85.5            1.6               5.6
1           86.7            1.0               4.9
2           91.5            1.2               4.4
3           11.7            2.3               4.3
4           71.2            7.1               6.0
```

[58]:
```
#grouped_data = data.groupby('school_rating')['reduced_lunch']
#print(type(grouped_data))

# Iterate through each group and print all the entries for that rating
#for rating, group in grouped_data:
#    print(f"School Rating: {rating}")
#    print(group)
#    print("\n")

# Phase 2: Group data by school_rating and describe reduced_lunch statistics␣
 ↪for each rating group
grouped_data = data.groupby('school_rating')['reduced_lunch'].describe()
print("Grouped Data by School Rating:\n", grouped_data)
```

```
Grouped Data by School Rating:
                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
```

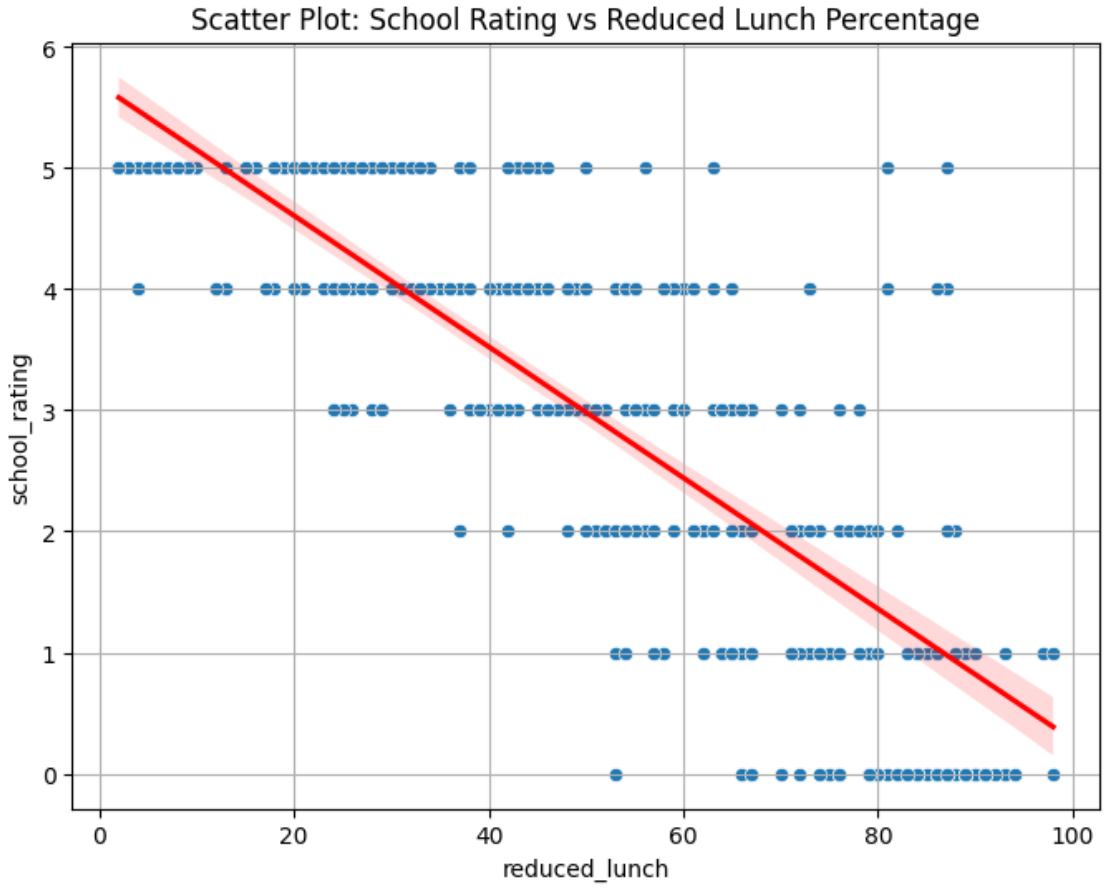
[59]:
```
# Phase 3: Correlation analysis between reduced_lunch and school_rating
correlation = data[['reduced_lunch', 'school_rating']].corr()
print("\nCorrelation between Reduced Lunch and School Rating:\n", correlation)
```

```
Correlation between Reduced Lunch and School Rating:
                reduced_lunch  school_rating
reduced_lunch        1.000000      -0.815757
school_rating       -0.815757       1.000000
```

[60]:
```
# Phase 4: Scatter plot between reduced_lunch and school_rating
plt.figure(figsize=(8, 6))
sns.scatterplot(x='reduced_lunch', y='school_rating', data=data)
plt.title('Scatter Plot: School Rating vs Reduced Lunch Percentage')
plt.xlabel('Reduced Lunch Percentage (%)')
plt.ylabel('School Rating')
plt.grid(True)

# Display a trend line (regression line) for the scatter plot
sns.regplot(x='reduced_lunch', y='school_rating', data=data, scatter=False,␣
 ↪color='red')

plt.show()
```

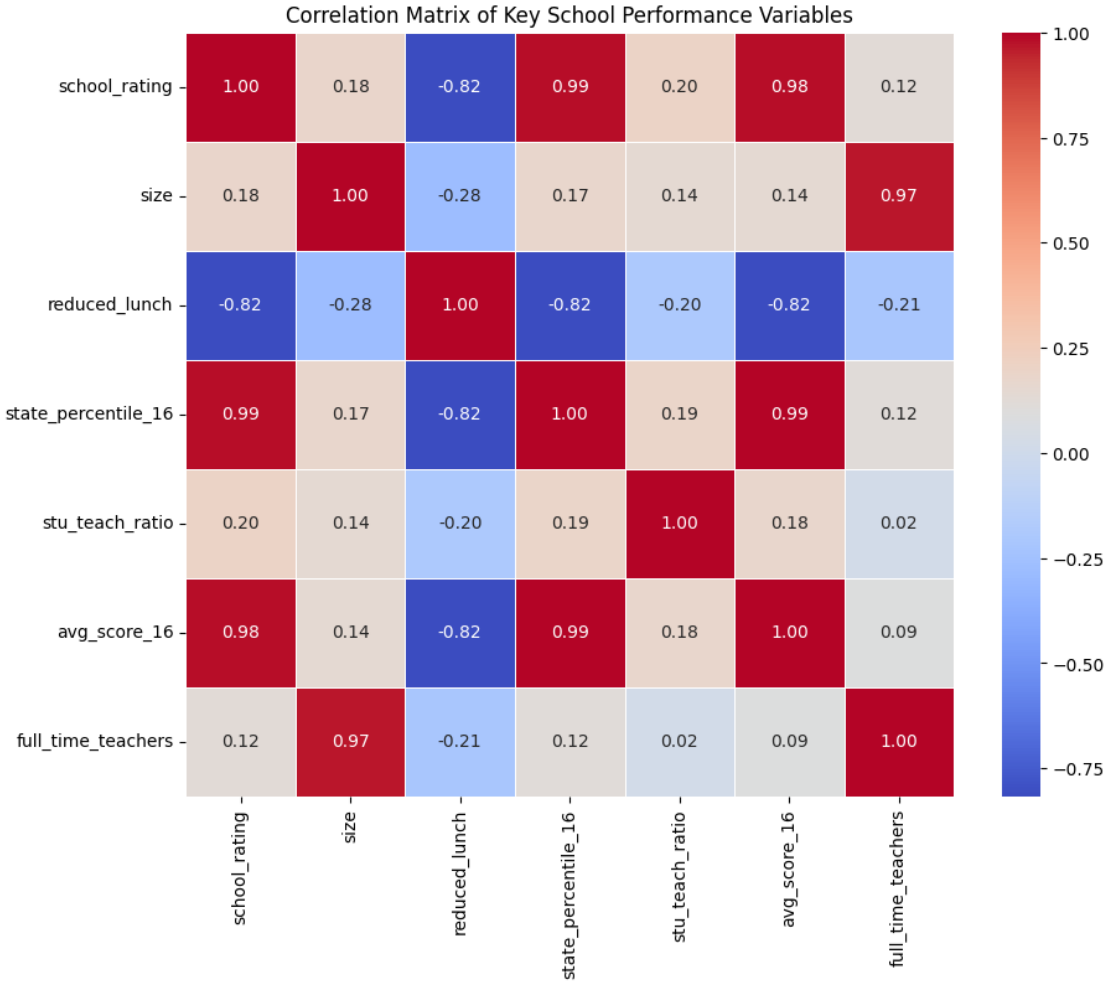


[61]:
```
# Phase 5: Correlation Matrix Visualization (including important fields)
plt.figure(figsize=(10, 8))

# Create a correlation matrix of important fields in the dataset
```

```
important_columns = ['school_rating', 'size', 'reduced_lunch',␣
 ↪'state_percentile_16',
                     'stu_teach_ratio', 'avg_score_16', 'full_time_teachers']
correlation_matrix = data[important_columns].corr()

# Visualize the correlation matrix using a heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪linewidths=0.5)
plt.title('Correlation Matrix of Key School Performance Variables')
plt.show()
```



Akram MTir 23-10-2024