Practical Assignment 3

Name: Mohit Muley

Roll no.: 276

PRN no.: 202201040192

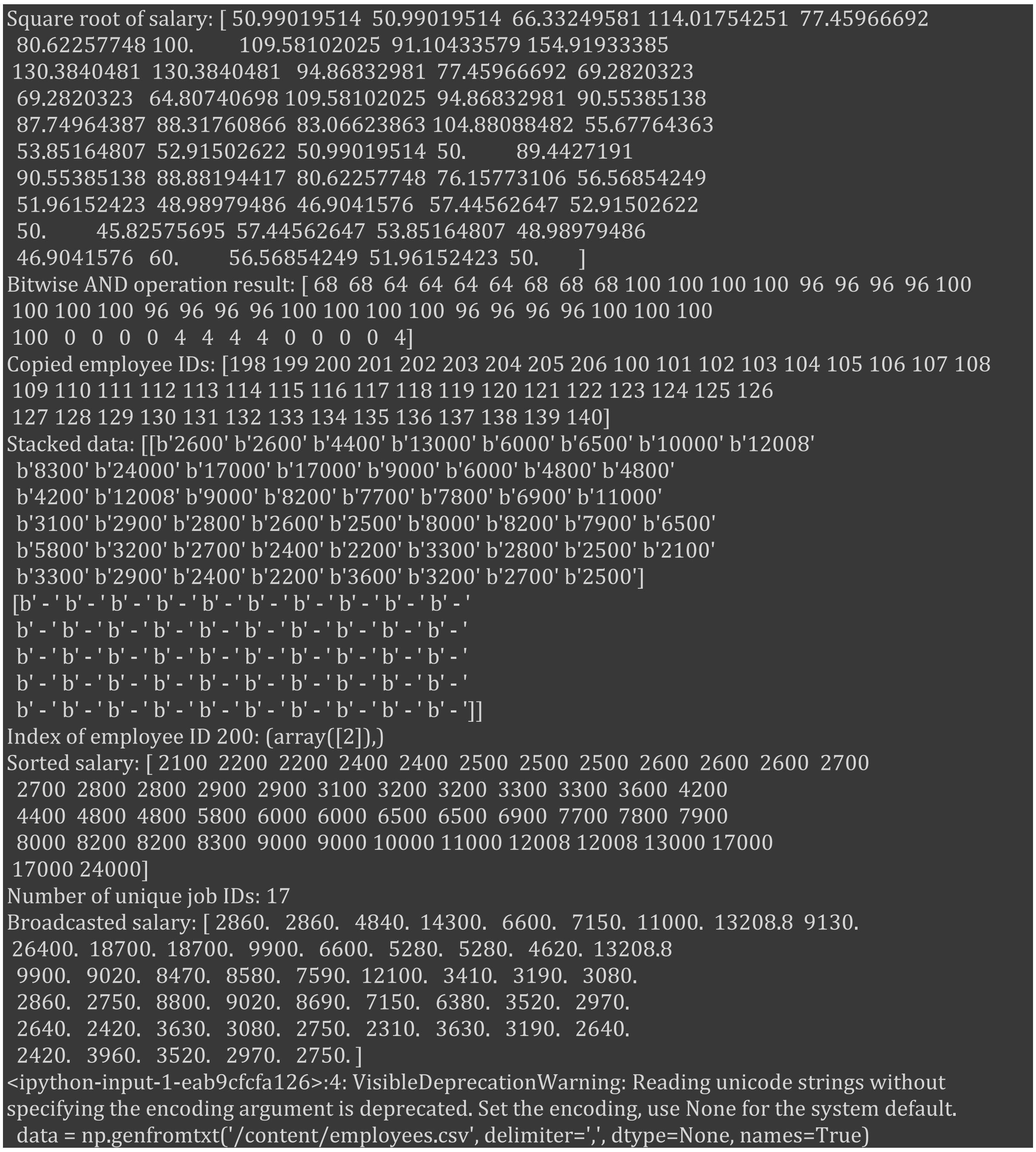
Division: B   Batch: B4

Subject: EDS

```python
import numpy as np

# Read the CSV file into a NumPy array
data = np.genfromtxt('/content/employees.csv', delimiter=',',
dtype=None, names=True)

# Accessing columns
employee_ids = data['EMPLOYEE_ID']
first_names = data['FIRST_NAME']
last_names = data['LAST_NAME']

# Perform matrix operations
# Example: Multiplying the 'SALARY' column by 2
salary = data['SALARY']
result = salary * 2

# Horizontal stacking of arrays
# Example: Stacking 'EMPLOYEE_ID' and 'SALARY' columns horizontally
stacked = np.hstack((employee_ids.reshape(-1, 1), salary.reshape(-1,
1)))

# Vertical stacking of arrays
# Example: Stacking 'FIRST_NAME' and 'LAST_NAME' columns vertically
stacked = np.vstack((first_names, last_names))

# Custom sequence generation
# Example: Generating a sequence from 1 to 10
sequence = np.arange(1, 11)

# Arithmetic and Statistical Operations
# Example: Finding the mean of the 'SALARY' column
mean_salary = np.mean(salary)

# Mathematical Operations
# Example: Calculating the square root of the 'SALARY' column
sqrt_salary = np.sqrt(salary)

# Bitwise Operators
# Example: Performing a bitwise AND operation on the 'EMPLOYEE_ID'
column
and_operation = employee_ids & 100

# Copying and viewing arrays
# Example: Creating a copy of the 'EMPLOYEE_ID' column
employee_ids_copy = employee_ids.copy()

# Data Stacking, Searching, Sorting, Counting, Broadcasting
# Example: Stacking 'SALARY' and 'COMMISSION_PCT' columns vertically
stacked_data = np.vstack((salary, data['COMMISSION_PCT']))

# Example: Searching for the index of a specific value in the
'EMPLOYEE_ID' column
index = np.where(employee_ids == 200)

# Example: Sorting the 'SALARY' column in ascending order
sorted_salary = np.sort(salary)

# Example: Counting the number of unique values in the 'JOB_ID' column
unique_jobs = np.unique(data['JOB_ID'])
count_jobs = len(unique_jobs)

# Example: Broadcasting the 'SALARY' column with a scalar value
broadcasted = salary * 1.1

# Print the information
print("Employee IDs:", employee_ids)
print("First Names:", first_names)
print("Last Names:", last_names)
print("Result of matrix operation:", result)
print("Horizontally stacked array:", stacked)
print("Vertically stacked array:", stacked)
print("Custom sequence:", sequence)
print("Mean salary:", mean_salary)
print("Square root of salary:", sqrt_salary)
print("Bitwise AND operation result:", and_operation)
print("Copied employee IDs:", employee_ids_copy)
print("Stacked data:", stacked_data)
print("Index of employee ID 200:", index)
print("Sorted salary:", sorted_salary)
print("Number of unique job IDs:", count_jobs)
print("Broadcasted salary:", broadcasted)
```

Output:

```
Employee IDs: [198 199 200 201 202 203 204 205 206 100 101 102 103 104 105 106 107 108
 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
 127 128 129 130 131 132 133 134 135 136 137 138 139 140]
First Names: [b'Donald' b'Douglas' b'Jennifer' b'Michael' b'Pat' b'Susan' b'Hermann'
 b'Shelley' b'William' b'Steven' b'Neena' b'Lex' b'Alexander' b'Bruce'
 b'David' b'Valli' b'Diana' b'Nancy' b'Daniel' b'John' b'Ismael'
 b'Jose Manuel' b'Luis' b'Den' b'Alexander' b'Shelli' b'Sigal' b'Guy'
 b'Karen' b'Matthew' b'Adam' b'Payam' b'Shanta' b'Kevin' b'Julia' b'Irene'
 b'James' b'Steven' b'Laura' b'Mozhe' b'James' b'TJ' b'Jason' b'Michael'
 b'Ki' b'Hazel' b'Renske' b'Stephen' b'John' b'Joshua']
Last Names: [b'OConnell' b'Grant' b'Whalen' b'Hartstein' b'Fay' b'Mavris' b'Baer'
 b'Higgins' b'Gietz' b'King' b'Kochhar' b'De Haan' b'Hunold' b'Ernst'
 b'Austin' b'Pataballa' b'Lorentz' b'Greenberg' b'Faviet' b'Chen'
 b'Sciarra' b'Urman' b'Popp' b'Raphaely' b'Khoo' b'Baida' b'Tobias'
 b'Himuro' b'Colmenares' b'Weiss' b'Fripp' b'Kaufling' b'Vollman'
 b'Mourgos' b'Nayer' b'Mikkilineni' b'Landry' b'Markle' b'Bissot'
 b'Atkinson' b'Marlow' b'Olson' b'Mallin' b'Rogers' b'Gee' b'Philtanker'
 b'Ladwig' b'Stiles' b'Seo' b'Patel']
Result of matrix operation: [ 5200  5200  8800 26000 12000 13000 20000 24016 16600 48000 34000
34000
 18000 12000  9600  9600  8400 24016 18000 16400 15400 15600 13800 22000
  6200  5800  5600  5200  5000 16000 16400 15800 13000 11600  6400  5400
  4800  4400  6600  5600  5000  4200  6600  5800  4800  4400  7200  6400
  5400  5000]
Horizontally stacked array: [[b'Donald' b'Douglas' b'Jennifer' b'Michael' b'Pat' b'Susan' b'Hermann'
  b'Shelley' b'William' b'Steven' b'Neena' b'Lex' b'Alexander' b'Bruce'
  b'David' b'Valli' b'Diana' b'Nancy' b'Daniel' b'John' b'Ismael'
  b'Jose Manuel' b'Luis' b'Den' b'Alexander' b'Shelli' b'Sigal' b'Guy'
  b'Karen' b'Matthew' b'Adam' b'Payam' b'Shanta' b'Kevin' b'Julia'
  b'Irene' b'James' b'Steven' b'Laura' b'Mozhe' b'James' b'TJ' b'Jason'
  b'Michael' b'Ki' b'Hazel' b'Renske' b'Stephen' b'John' b'Joshua']
 [b'OConnell' b'Grant' b'Whalen' b'Hartstein' b'Fay' b'Mavris' b'Baer'
  b'Higgins' b'Gietz' b'King' b'Kochhar' b'De Haan' b'Hunold' b'Ernst'
  b'Austin' b'Pataballa' b'Lorentz' b'Greenberg' b'Faviet' b'Chen'
  b'Sciarra' b'Urman' b'Popp' b'Raphaely' b'Khoo' b'Baida' b'Tobias'
  b'Himuro' b'Colmenares' b'Weiss' b'Fripp' b'Kaufling' b'Vollman'
  b'Mourgos' b'Nayer' b'Mikkilineni' b'Landry' b'Markle' b'Bissot'
  b'Atkinson' b'Marlow' b'Olson' b'Mallin' b'Rogers' b'Gee' b'Philtanker'
  b'Ladwig' b'Stiles' b'Seo' b'Patel']]
Vertically stacked array: [[b'Donald' b'Douglas' b'Jennifer' b'Michael' b'Pat' b'Susan' b'Hermann'
  b'Shelley' b'William' b'Steven' b'Neena' b'Lex' b'Alexander' b'Bruce'
  b'David' b'Valli' b'Diana' b'Nancy' b'Daniel' b'John' b'Ismael'
  b'Jose Manuel' b'Luis' b'Den' b'Alexander' b'Shelli' b'Sigal' b'Guy'
  b'Karen' b'Matthew' b'Adam' b'Payam' b'Shanta' b'Kevin' b'Julia'
  b'Irene' b'James' b'Steven' b'Laura' b'Mozhe' b'James' b'TJ' b'Jason'
  b'Michael' b'Ki' b'Hazel' b'Renske' b'Stephen' b'John' b'Joshua']
 [b'OConnell' b'Grant' b'Whalen' b'Hartstein' b'Fay' b'Mavris' b'Baer'
  b'Higgins' b'Gietz' b'King' b'Kochhar' b'De Haan' b'Hunold' b'Ernst'
  b'Austin' b'Pataballa' b'Lorentz' b'Greenberg' b'Faviet' b'Chen'
  b'Sciarra' b'Urman' b'Popp' b'Raphaely' b'Khoo' b'Baida' b'Tobias'
  b'Himuro' b'Colmenares' b'Weiss' b'Fripp' b'Kaufling' b'Vollman'
  b'Mourgos' b'Nayer' b'Mikkilineni' b'Landry' b'Markle' b'Bissot'
  b'Atkinson' b'Marlow' b'Olson' b'Mallin' b'Rogers' b'Gee' b'Philtanker'
  b'Ladwig' b'Stiles' b'Seo' b'Patel']]
Custom sequence: [ 1  2  3  4  5  6  7  8  9 10]
Mean salary: 6182.32
```



```
Square root of salary: [ 50.99019514  50.99019514  66.33249581 114.01754251  77.45966692
  80.62257748 100.         109.58102025  91.10433579 154.91933385
 130.3840481  130.3840481   94.86832981  77.45966692  69.2820323
  69.2820323   64.80740698 109.58102025  94.86832981  90.55385138
  87.74964387  88.31760866  83.06623863 104.88088482  55.67764363
  53.85164807  52.91502622  50.99019514  50.         89.4427191
  90.55385138  88.88194417  80.62257748  76.15773106  56.56854249
  51.96152423  48.98979486  46.9041576   57.44562647  52.91502622
  50.         45.82575695  57.44562647  53.85164807  48.98979486
  46.9041576   60.         56.56854249  51.96152423  50.        ]
Bitwise AND operation result: [ 68  68  64  64  64  64  68  68  68 100 100 100 100  96  96  96  96 100
 100 100 100  96  96  96  96 100 100 100 100  96  96  96  96 100 100 100
 100   0   0   0   0   4   4   4   4   0   0   0   0   4]
Copied employee IDs: [198 199 200 201 202 203 204 205 206 100 101 102 103 104 105 106 107 108
 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
 127 128 129 130 131 132 133 134 135 136 137 138 139 140]
Stacked data: [[b'2600' b'2600' b'4400' b'13000' b'6000' b'6500' b'10000' b'12008'
  b'8300' b'24000' b'17000' b'17000' b'9000' b'6000' b'4800' b'4800'
  b'4200' b'12008' b'9000' b'8200' b'7700' b'7800' b'6900' b'11000'
  b'3100' b'2900' b'2800' b'2600' b'2500' b'8000' b'8200' b'7900' b'6500'
  b'5800' b'3200' b'2700' b'2400' b'2200' b'3300' b'2800' b'2500' b'2100'
  b'3300' b'2900' b'2400' b'2200' b'3600' b'3200' b'2700' b'2500']
 [b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - '
  b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - '
  b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - '
  b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - '
  b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ' b' - ']]
Index of employee ID 200: (array([2]),)
Sorted salary: [ 2100  2200  2200  2400  2400  2500  2500  2500  2600  2600  2600  2700
  2700  2800  2800  2900  2900  3100  3200  3200  3300  3300  3600  4200
  4400  4800  4800  5800  6000  6000  6500  6500  6900  7700  7800  7900
  8000  8200  8200  8300  9000  9000 10000 11000 12008 12008 13000 17000
 17000 24000]
Number of unique job IDs: 17
Broadcasted salary: [ 2860.   2860.   4840.  14300.   6600.   7150.  11000.  13208.8  9130.
 26400.  18700.  18700.   9900.   6600.   5280.   5280.   4620.  13208.8
  9900.   9020.   8470.   8580.   7590.  12100.   3410.   3190.   3080.
  2860.   2750.   8800.   9020.   8690.   7150.   6380.   3520.   2970.
  2640.   2420.   3630.   3080.   2750.   2310.   3630.   3190.   2640.
  2420.   3960.   3520.   2970.   2750. ]
<ipython-input-1-eab9cfcfa126>:4: VisibleDeprecationWarning: Reading unicode strings without
specifying the encoding argument is deprecated. Set the encoding, use None for the system default.
  data = np.genfromtxt('/content/employees.csv', delimiter=',', dtype=None, names=True)
```