Assignment Class 4

NumPy

In [1]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\hites\appdata\local\programs\python\python311\lib\site-packages
(1.24.3)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\hites\appdata\local\programs\python\python311\lib\site-package
s (2.0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hites\appdata\local\programs\python\python311\
lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hites\appdata\local\programs\python\python311\lib\site-p
ackages (from pandas) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\hites\appdata\local\programs\python\python311\lib\site
-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\hites\appdata\local\programs\python\python311\lib\site-
packages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\users\hites\appdata\local\programs\python\python311\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
'''Q1 Create a NumPy array containing integers from 1 to 10.'''
```

Out[3]:
```
'Q1 Create a NumPy array containing integers from 1 to 10.'
```

In [4]:
```
# Ans1
import numpy as np
a = np.arange(1,11)
print(a)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

In [5]:
```
'''Q2. Create a 2D NumPy array with shape (3, 4) filled with random float values between 0
and 1.'''
```

Out[5]:
```
'Q2. Create a 2D NumPy array with shape (3, 4) filled with random float values between 0\nand 1.'
```

In [6]:
```
# Ans2
np.random.random((3,4))
#np.random.randint(0,1,size = (3,4))
```

Out[6]:
```
array([[0.88938088, 0.00679188, 0.31623687, 0.64028212],
       [0.89258516, 0.90352362, 0.86309588, 0.67747313],
       [0.74265669, 0.05689596, 0.71839777, 0.21235991]])
```

In [7]:
```
'''Q3. Given the following NumPy array:
arr = np.array([10, 20, 30, 40, 50])
Add 5 to each element of the array.'''
```

Out[7]:
```
'Q3. Given the following NumPy array:\narr = np.array([10, 20, 30, 40, 50])\nAdd 5 to each element of the array
.'
```

In [8]:
```
# Ans3
arr = np.array([10, 20, 30, 40, 50])
print(arr+5)
```

```
[15 25 35 45 55]
```

In [9]:
```
'''Q4. Given two NumPy arrays:
arr1 = np.array([1, 2, 3, 4]) arr2 = np.array([5, 6, 7, 8])
Concatenate them into a single array'''
```

Out[9]:
```
'Q4. Given two NumPy arrays:\narr1 = np.array([1, 2, 3, 4]) arr2 = np.array([5, 6, 7, 8])\nConcatenate them int
o a single array'
```

In [10]:
```
# Ans4
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([5, 6, 7, 8])
np.concatenate((arr1,arr2))
```

Out[10]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [11]:
```
'''Q5 Create a NumPy array of 10 elements with equally spaced values from 0 to 9.'''
```

Out[11]: 'Q5 Create a NumPy array of 10 elements with equally spaced values from 0 to 9.'

In [12]:
```
# Ans5
arr = np.linspace(0,9,10)
print(arr)
```

```
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

In [13]:
```
'''Q6. Given a NumPy array:
arr = np.array([1, 2, 3, 4, 5, 6])
Reverse the elements in the array.'''
```

Out[13]: 'Q6. Given a NumPy array:\narr = np.array([1, 2, 3, 4, 5, 6])\nReverse the elements in the array.'

In [14]:
```
# Ans6
arr = np.array([1, 2, 3, 4, 5, 6])
print(arr[::-1])
```

```
[6 5 4 3 2 1]
```

In [15]:
```
'''Q7 Create a 3x3 identity matrix using NumPy.'''
```

Out[15]: 'Q7 Create a 3x3 identity matrix using NumPy.'

In [16]:
```
# Ans7
identity_matrix = np.eye(3)
print(identity_matrix)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [17]:
```
'''Q8. Calculate the mean, median, and standard deviation of the following NumPy array:
arr = np.array([15, 20, 25, 30, 35])
'''
```

Out[17]: 'Q8. Calculate the mean, median, and standard deviation of the following NumPy array:\narr = np.array([15, 20, 25, 30, 35])\n'

In [18]:
```
# Ans8
arr = np.array([15, 20, 25, 30, 35])
Mean = np.mean(arr)
Median = np.median(arr)
Std = np.std(arr)
print(Mean)
print(Median)
print(Std)
```

```
25.0
25.0
7.0710678118654755
```

In [19]:
```
'''Q9. Given a 2D NumPy array:
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
Get the diagonal elements of the array.
'''
```

Out[19]: 'Q9. Given a 2D NumPy array:\narr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])\nGet the diagonal elements of the array.\n'

In [20]:
```
# Ans9
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
np.diagonal(arr)
```

Out[20]: array([1, 5, 9])

In [21]:
```
'''Q10. Create a NumPy array of 10 random integers between 1 and 100 (inclusive).
'''
```

Out[21]: 'Q10. Create a NumPy array of 10 random integers between 1 and 100 (inclusive).\n'

In [22]:
```
# Ans10
x = np.random.randint(1,100, 10)
print(x)
```

```
[26 34  2 30 19 67 39 50 76 13]
```

In [23]:
```
'''Q11. Create a 2D NumPy array of shape (5, 5) with random integers between 1 and 50
(inclusive).'''
```

Out[23]: 'Q11. Create a 2D NumPy array of shape (5, 5) with random integers between 1 and 50\n(inclusive).'

In [24]:
```
# Ans11
```

```
x = np.random.randint(1,50, size=(5,5))
print(x)
```

```
[[15 23 26 19 24]
 [35 13 38 29 15]
 [27 31 17 31 21]
 [38  7 37 35 35]
 [17  8 27 26 18]]
```

In [25]:
```
'''Q12. Given a 2D NumPy array:
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
Calculate the sum of all the elements in the array'''
```

Out[25]: 'Q12. Given a 2D NumPy array:\narr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])\nCalculate the sum of all the elements in the array'

In [26]:
```
# Ans12
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
x = np.sum(arr)
print(x)
```

```
45
```

In [27]:
```
'''Q13. Create a NumPy array with 100 evenly spaced values between 0 and 1 (inclusive) and
reshape it into a 10x10 matrix.'''
```

Out[27]: 'Q13. Create a NumPy array with 100 evenly spaced values between 0 and 1 (inclusive) and\nreshape it into a 10x10 matrix.'

In [28]:
```
# Ans13
x = np.linspace(0, 1, 100).reshape(10, 10)
print(x)
```

```
[[0.         0.01010101 0.02020202 0.03030303 0.04040404 0.05050505
  0.06060606 0.07070707 0.08080808 0.09090909]
 [0.1010101  0.11111111 0.12121212 0.13131313 0.14141414 0.15151515
  0.16161616 0.17171717 0.18181818 0.19191919]
 [0.2020202  0.21212121 0.22222222 0.23232323 0.24242424 0.25252525
  0.26262626 0.27272727 0.28282828 0.29292929]
 [0.3030303  0.31313131 0.32323232 0.33333333 0.34343434 0.35353535
  0.36363636 0.37373737 0.38383838 0.39393939]
 [0.4040404  0.41414141 0.42424242 0.43434343 0.44444444 0.45454545
  0.46464646 0.47474747 0.48484848 0.49494949]
 [0.50505051 0.51515152 0.52525253 0.53535354 0.54545455 0.55555556
  0.56565657 0.57575758 0.58585859 0.5959596 ]
 [0.60606061 0.61616162 0.62626263 0.63636364 0.64646465 0.65656566
  0.66666667 0.67676768 0.68686869 0.6969697 ]
 [0.70707071 0.71717172 0.72727273 0.73737374 0.74747475 0.75757576
  0.76767677 0.77777778 0.78787879 0.7979798 ]
 [0.80808081 0.81818182 0.82828283 0.83838384 0.84848485 0.85858586
  0.86868687 0.87878788 0.88888889 0.8989899 ]
 [0.90909091 0.91919192 0.92929293 0.93939394 0.94949495 0.95959596
  0.96969697 0.97979798 0.98989899 1.        ]]
```

In [29]:
```
'''Q14. Given two NumPy arrays:
import numpy as np
arr1 = np.array([1, 2, 3, 4]) arr2 = np.array([5, 6, 7, 8])
Find the common elements between the two arrays'''
```

Out[29]: 'Q14. Given two NumPy arrays:\nimport numpy as np\narr1 = np.array([1, 2, 3, 4]) arr2 = np.array([5, 6, 7, 8])\nFind the common elements between the two arrays'

In [30]:
```
# Ans14
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([5, 6, 7, 8])
x = np.intersect1d(arr1,arr2)
print(x)
```

```
[]
```

In [31]:
```
'''Q15. Create a NumPy array containing 10 random integers between -50 and 50 (inclusive).
Replace all negative values in the array with 0'''
```

Out[31]: 'Q15. Create a NumPy array containing 10 random integers between -50 and 50 (inclusive).\nReplace all negative values in the array with 0'

In [32]:
```
# Ans15
x = np.random.randint(-50,50,10)
print(x)
x[x<0] = 0
print(x)
```

```
[  1 -40  31   2   0  27 -44 -44 -30  23]
[ 1  0 31  2  0 27  0  0  0 23]
```

In [33]:
```
'''Q16  Given a 2D NumPy array:
```

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
Calculate the sum of each row and each column separately.'''
```

Out[33]: 'Q16 Given a 2D NumPy array:\nimport numpy as np\narr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])\nCalculate the sum of each row and each column separately.'

In [34]:
```
# Ans16
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
x = np.sum(arr, axis =0)
y = np.sum(arr, axis =1)
print(x)
print(y)
```

```
[12 15 18]
[ 6 15 24]
```

In [35]:
```
'''Q17. Create a NumPy array of shape (6, 6) with diagonal elements as 1, 2, 3, 4, 5, and 6, and
all other elements as 0.
'''
```

Out[35]: 'Q17. Create a NumPy array of shape (6, 6) with diagonal elements as 1, 2, 3, 4, 5, and 6, and\nall other elements as 0.\n'

In [36]:
```
# Ans17
x = np.diag(np.arange(1,7))
print(x)
```

```
[[1 0 0 0 0 0]
 [0 2 0 0 0 0]
 [0 0 3 0 0 0]
 [0 0 0 4 0 0]
 [0 0 0 0 5 0]
 [0 0 0 0 0 6]]
```

In [37]:
```
'''Q18. Given a NumPy array:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
Normalize the array so that the values range from 0 to 1.
'''
```

Out[37]: 'Q18. Given a NumPy array:\nimport numpy as np\narr = np.array([1, 2, 3, 4, 5])\nNormalize the array so that the values range from 0 to 1.\n'

In [38]:
```
# Ans18
arr = np.array([1, 2, 3, 4, 5])
y = []
for i in arr:
    a = i-min(arr)/(max(arr)-min(arr))
    y.append(a)
nor = np.array(y)
print(nor)
```

```
[0.75 1.75 2.75 3.75 4.75]
```

In [39]:
```
'''Q19. Create a NumPy array with 20 random integers between 1 and 100 (inclusive). Find the
maximum value and its index in the array'''
```

Out[39]: 'Q19. Create a NumPy array with 20 random integers between 1 and 100 (inclusive). Find the\nmaximum value and its index in the array'

In [40]:
```
# Ans19
x = np.random.randint(1,100,20)
y = np.max(x)
z = np.where(x==y)
print(y,z)
```

```
97 (array([10], dtype=int64),)
```

In [41]:
```
'''Q20. Given two NumPy arrays:
import numpy as np
arr1 = np.array([1, 2, 3, 4]) arr2 = np.array([5, 6, 7, 8])
Compute the element-wise product of the two arrays.'''
```

Out[41]: 'Q20. Given two NumPy arrays:\nimport numpy as np\narr1 = np.array([1, 2, 3, 4]) arr2 = np.array([5, 6, 7, 8])\nCompute the element-wise product of the two arrays.'

In [42]:
```
# Ans20
arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([5, 6, 7, 8])
x = arr1*arr2
print(x)
```

```
[ 5 12 21 32]
```

PANDAS

PANDAS

In [43]:
```
import pandas as pd
```

In [44]:
```
'''Q.1 Write a Pandas program to add, subtract, multiple and divide two Pandas Series.
Input: [2, 4, 6, 8, 10], [1, 3, 5, 7, 9]
'''
```

Out[44]:
```
'Q.1 Write a Pandas program to add, subtract, multiple and divide two Pandas Series.\nInput: [2, 4, 6, 8, 10], [1, 3, 5, 7, 9]\n'
```

In [45]:
```
# Ans1
x = pd.Series([2, 4, 6, 8, 10])
y = pd.Series([1, 3, 5, 7, 9])
print('Sum: ',x+y)
print('Subtract: ',x-y)
print('Multiply: ',x*y)
print('Devide: ',x/y)
```

```
Sum:  0     3
1     7
2    11
3    15
4    19
dtype: int64
Subtract:  0    1
1    1
2    1
3    1
4    1
dtype: int64
Multiply:  0     2
1    12
2    30
3    56
4    90
dtype: int64
Devide:  0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```

In [46]:
```
'''Q.2 Create a Pandas DataFrame from the following dictionary, where the keys represent
columns and the values represent the data:
data = { 'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [25, 30, 22, 28], 'City': ['New York',
'London', 'Paris', 'Tokyo'] }'''
```

Out[46]:
```
"Q.2 Create a Pandas DataFrame from the following dictionary, where the keys represent\ncolumns and the values represent the data:\ndata = { 'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [25, 30, 22, 28], 'City': ['New York',\n'London', 'Paris', 'Tokyo'] }"
```

In [47]:
```
# Ans2
x = { 'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [25, 30, 22, 28], 'City': ['New York', 'London', 'Pa
df = pd.DataFrame(x)
print(df)
```

```
      Name  Age      City
0    Alice   25  New York
1      Bob   30    London
2  Charlie   22     Paris
3    David   28     Tokyo
```

In [48]:
```
'''Q.3 Given a Pandas DataFrame 'df', select the first 5 rows of the DataFrame.'''
```

Out[48]:
```
"Q.3 Given a Pandas DataFrame 'df', select the first 5 rows of the DataFrame."
```

In [49]:
```
# Ans 3
df = pd.DataFrame(x)
df.head() #x has only 4 rows.
```

Out[49]:

	Name	Age	City
0	Alice	25	New York
1	Bob	30	London
2	Charlie	22	Paris
3	David	28	Tokyo

In [50]:
```
'''Q.4 Create a new column 'Salary' in the DataFrame 'df' with random integer values
```

```
between 50000 and 80000 (inclusive).'''
```

Out[50]:
```
"Q.4 Create a new column 'Salary' in the DataFrame 'df' with random integer values\nbetween 50000 and 80000 (in
clusive)."
```

In [51]:
```
# Ans4
df1 = pd.DataFrame({'Name':['Hi','Bi','Ci','Ki','Ti','Li','Gi','Di','Ni','Si']})
y = np.random.randint(50000,80000,10)
df1['Salary'] = y
print(df1)
```

```
  Name  Salary
0   Hi   71920
1   Bi   69084
2   Ci   79184
3   Ki   52893
4   Ti   52710
5   Li   58852
6   Gi   78638
7   Di   50361
8   Ni   79840
9   Si   65022
```

In [52]:
```
'''Q.5 Given two Pandas DataFrames 'df1' and 'df2' with the same columns, concatenate
them vertically.'''
```

Out[52]:
```
"Q.5 Given two Pandas DataFrames 'df1' and 'df2' with the same columns, concatenate\nthem vertically."
```

In [53]:
```
df1 = pd.DataFrame({'Name':['Hitesh','Rajesh'],'Age':[20,30]})
df2 = pd.DataFrame({'Name':['Shifa','Kamla'],'Age':[25,24]})
pd.concat([df1,df2])
```

Out[53]:

	Name	Age
0	Hitesh	20
1	Rajesh	30
0	Shifa	25
1	Kamla	24

In [54]:
```
'''Q.6 Create a new DataFrame named 'df_filtered' from 'df', containing only rows where
the 'Age' column is greater than 25.
'''
```

Out[54]:
```
"Q.6 Create a new DataFrame named 'df_filtered' from 'df', containing only rows where\nthe 'Age' column is grea
ter than 25.\n"
```

In [55]:
```
# Ans6
df = pd.DataFrame({'Name':['Hi','Bi','ci','ki'],'Age':[10,24,45,40]})
df_filtered = df[df['Age']>25]
print(df_filtered)
```

```
  Name  Age
2   ci   45
3   ki   40
```

In [56]:
```
'''Q.7 Given a Pandas DataFrame 'df', sort the DataFrame based on the 'Age' column in
ascending order'''
```

Out[56]:
```
"Q.7 Given a Pandas DataFrame 'df', sort the DataFrame based on the 'Age' column in\nascending order"
```

In [57]:
```
# Ans7
df.sort_values(by = 'Age',ascending=True)
```

Out[57]:

	Name	Age
0	Hi	10
1	Bi	24
3	ki	40
2	ci	45

In [58]:
```
'''Q.8 Calculate the mean and median of the 'Salary' column in the DataFrame 'df'.
'''
```

Out[58]:
```
"Q.8 Calculate the mean and median of the 'Salary' column in the DataFrame 'df'.\n"
```

In [59]:
```
# Ans 8
x = [['Hi',10,2500],['Bi',20,2600],['Ci',15,3500]]
df = pd.DataFrame(x,columns=['Name','Age',"Salary"])
```

```
print(df)
print(df['Salary'].mean())
print(df['Salary'].median())
print(df['Salary'].std())
```

```
  Name  Age  Salary
0   Hi   10    2500
1   Bi   20    2600
2   Ci   15    3500
2866.6666666666665
2600.0
550.7570547286102
```

In [60]:
```
'''Q.9 Group the DataFrame 'df' by the 'City' column and calculate the mean 'Age' for each
group.'''
```

Out[60]:
```
"Q.9 Group the DataFrame 'df' by the 'City' column and calculate the mean 'Age' for each\ngroup."
```

In [61]:
```
# Ans9
df = pd.DataFrame({'Name':['Hi','Bi','Ci','Di'],'City':['Jhansi','Nlk','Jhansi','Nlk'],'Age':[25,24,27,65]})
print(df)
Group = df.groupby('City')
x = Group['Age'].mean()
y = pd.DataFrame(x)
print(y)
```

```
  Name    City  Age
0   Hi  Jhansi   25
1   Bi     Nlk   24
2   Ci  Jhansi   27
3   Di     Nlk   65
         Age
City
Jhansi  26.0
Nlk     44.5
```

In [62]:
```
'''Q.10 Read the csv provide(nba.csv) into a Dataframe?Display:
*Number of rows and columns in dataframe
* Top 10 and bottom 10 rows
* Summary statistics for numerical columns
*Datatype of columns in dataframe
* Length of the data
* Print all the Rows from index 100 to 200'''
```

Out[62]:
```
'Q.10 Read the csv provide(nba.csv) into a Dataframe\x02Display:\n*Number of rows and columns in dataframe\n* T
op 10 and bottom 10 rows\n* Summary statistics for numerical columns\n*Datatype of columns in dataframe\n* Leng
th of the data\n* Print all the Rows from index 100 to 200'
```

In [63]:
```
df = pd.read_csv(r"C:\Users\hites\OneDrive\Desktop\Data analyst Bootcamp\Python\Class 4\user_course_file_648da6
```

In [64]:
```
df.shape
```

Out[64]:
```
(457, 8)
```

In [65]:
```
df.head(10)
```

Out[65]:

	Name	Team	Number	Position	Age	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	180	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	235	Marquette	6796117.0
2	John Holland	Boston Celtics	30	SG	27	205	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	185	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	231	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	240	NaN	12000000.0
6	Jordan Mickey	Boston Celtics	55	PF	21	235	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41	C	25	238	Gonzaga	2165160.0
8	Terry Rozier	Boston Celtics	12	PG	22	190	Louisville	1824360.0
9	Marcus Smart	Boston Celtics	36	PG	22	220	Oklahoma State	3431040.0

In [66]:
```
df.tail(10)
```

Out[66]:

	Name	Team	Number	Position	Age	Weight	College	Salary
447	Rudy Gobert	Utah Jazz	27	C	23	245	NaN	1175880.0
448	Gordon Hayward	Utah Jazz	20	SF	26	226	Butler	15409570.0
449	Rodney Hood	Utah Jazz	5	SG	23	206	Duke	1348440.0
450	Joe Ingles	Utah Jazz	2	SF	28	226	NaN	2050000.0
451	Chris Johnson	Utah Jazz	23	SF	26	206	Dayton	981348.0
452	Trey Lyles	Utah Jazz	41	PF	20	234	Kentucky	2239800.0
453	Shelvin Mack	Utah Jazz	8	PG	26	203	Butler	2433333.0
454	Raul Neto	Utah Jazz	25	PG	24	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21	C	26	256	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24	C	26	231	Kansas	947276.0

In [67]:
```
df.describe()
```

Out[67]:

	Number	Age	Weight	Salary
count	457.000000	457.000000	457.000000	4.460000e+02
mean	17.678337	26.938731	221.522976	4.842684e+06
std	15.966090	4.404016	26.368343	5.229238e+06
min	0.000000	19.000000	161.000000	3.088800e+04
25%	5.000000	24.000000	200.000000	1.044792e+06
50%	13.000000	26.000000	220.000000	2.839073e+06
75%	25.000000	30.000000	240.000000	6.500000e+06
max	99.000000	40.000000	307.000000	2.500000e+07

In [68]:
```
df.dtypes
```

Out[68]:
```
Name         object
Team         object
Number        int64
Position     object
Age           int64
Weight        int64
College      object
Salary      float64
dtype: object
```

In [69]:
```
len(df)
```

Out[69]:
```
457
```

In [70]:
```
df.iloc[100:200]
```

Out[70]:

	Name	Team	Number	Position	Age	Weight	College	Salary
100	Chris Paul	Los Angeles Clippers	3	PG	31	175	Wake Forest	21468695.0
101	Paul Pierce	Los Angeles Clippers	34	SF	38	235	Kansas	3376000.0
102	Pablo Prigioni	Los Angeles Clippers	9	PG	39	185	NaN	947726.0
103	JJ Redick	Los Angeles Clippers	4	SG	31	190	Duke	7085000.0
104	Austin Rivers	Los Angeles Clippers	25	PG	23	200	Duke	3110796.0
...	...	...	...	...	...	...	...	...
195	Anthony Tolliver	Detroit Pistons	43	PF	31	240	Creighton	3000000.0
196	Lavoy Allen	Indiana Pacers	5	PF	27	255	Temple	4050000.0
197	Rakeem Christmas	Indiana Pacers	25	PF	24	250	Syracuse	1007026.0
198	Monta Ellis	Indiana Pacers	11	SG	30	185	NaN	10300000.0
199	Paul George	Indiana Pacers	13	SF	26	220	Fresno State	17120106.0

100 rows × 8 columns

In [ ]:
```

```

In [ ]:
```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js