Class 6

In [1]:
```
'''Q1- Create a Pandas Data frame from the given data and create a new column “Voter” based on
voter age, i.e., if age >18 then voter column should be “Yes” otherwise if age <18 then voter
column should be “No”
raw_Data = {'Voter_name': ['Geek1', 'Geek2', 'Geek3', 'Geek4',
'Geek5', 'Geek6', 'Geek7', 'Geek8'],
'Voter_age': [15, 23, 25, 9, 67, 54, 42, np.NaN]}
'''
```

Out[1]:
```
"Q1- Create a Pandas Data frame from the given data and create a new column “Voter” based on\nvoter age, i.e.,
if age >18 then voter column should be “Yes” otherwise if age <18 then voter\ncolumn should be “No”\nraw_Data =
{'Voter_name': ['Geek1', 'Geek2', 'Geek3', 'Geek4',\n'Geek5', 'Geek6', 'Geek7', 'Geek8'],\n'Voter_age': [15, 23
, 25, 9, 67, 54, 42, np.NaN]}\n"
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
# Ans1
raw_Data = {'Voter_name': ['Geek1', 'Geek2', 'Geek3', 'Geek4',
                           'Geek5', 'Geek6', 'Geek7', 'Geek8'],
            'Voter_age': [15, 23, 25, 9, 67, 54, 42, np.NaN]}
df = pd.DataFrame(raw_Data)
print(df)
def Voter_g(age):
    if age > 18:
        return 'Yes'
    else:
        return 'No'
df['Voter'] = df['Voter_age'].apply(Voter_g)
print(df)
```

```
  Voter_name  Voter_age
0      Geek1       15.0
1      Geek2       23.0
2      Geek3       25.0
3      Geek4        9.0
4      Geek5       67.0
5      Geek6       54.0
6      Geek7       42.0
7      Geek8        NaN
  Voter_name  Voter_age Voter
0      Geek1       15.0    No
1      Geek2       23.0   Yes
2      Geek3       25.0   Yes
3      Geek4        9.0    No
4      Geek5       67.0   Yes
5      Geek6       54.0   Yes
6      Geek7       42.0   Yes
7      Geek8        NaN    No
```

In [4]:
```
'''Q2 – Create a Pandas Data frame from the given data and collapse First and Last column into one
column as Full Name, so the output contains Full Name and Age, then convert column age to
index
raw_Data = {'First': ['Manan ', 'Raghav ', 'Sunny '],
'Last': ['Goel', 'Sharma', 'Chawla'],
'Age' : [12, 24, 56]}
'''
```

Out[4]:
```
"Q2 – Create a Pandas Data frame from the given data and collapse First and Last column into one\ncolumn as Ful
l Name, so the output contains Full Name and Age, then convert column age to\nindex\nraw_Data = {'First': ['Man
an ', 'Raghav ', 'Sunny '],\n'Last': ['Goel', 'Sharma', 'Chawla'],\n'Age' : [12, 24, 56]}\n"
```

In [5]:
```
# Ans2
# Creating as new DataFrame
raw_Data = {'First': ['Manan ', 'Raghav ', 'Sunny '],
            'Last': ['Goel', 'Sharma', 'Chawla'],
            'Age' : [12, 24, 56]}
df= pd.DataFrame(raw_Data)
print(df)

# Adding a new collumn by joining two columns
df['Full_Name'] = df['First'] + df['Last']
print(df)

# Droppinng First & Last column
df.drop(['First','Last'],axis=1,inplace=True)
print(df)
```

```
# Converting column age to index
df.set_index(['Age'],inplace=True)
print(df)
```

```
    First    Last  Age
0   Manan     Goel   12
1  Raghav   Sharma   24
2   Sunny   Chawla   56
     First    Last  Age      Full_Name
0   Manan     Goel   12     Manan Goel
1  Raghav   Sharma   24  Raghav Sharma
2   Sunny   Chawla   56   Sunny Chawla
   Age      Full_Name
0   12     Manan Goel
1   24  Raghav Sharma
2   56   Sunny Chawla
         Full_Name
Age
12      Manan Goel
24   Raghav Sharma
56    Sunny Chawla
```

In [6]:
```
'''Q3- Create a Pandas Data frame from the given data -
raw_Data = {'Date':['10/2/2011', '11/2/2011', '12/2/2011', '13/2/2011'],
'Product':['Umbrella', 'Matress', 'Badminton','Shuttle'],
'Price':[1250, 1450, 1550, 400],
'Expense': [ 21525220.653, 31125840.875, 23135428.768, 56245263.942]}
a- Add Index as Item1, Item2, Item3, Item4
b- Find the index labels of all items whose 'Price' is greater than 1000.
c- Replace products using Map() with respective codes- Umbrella : 'U', Matress : 'M', Badminton
: 'B', Shuttle: 'S'
d- Round off the Expense column values to two decimal places.
e- Create a new column called 'Discounted_Price' after applying a 10% discount on the existing
'price' column.(try using lambda function)
f- Convert the column type of "Date" to datetime format.
g- Create a column rank which ranks the products based on the price (one with the highest price
will be rank 1).
'''
```

Out[6]:
```
"Q3- Create a Pandas Data frame from the given data -\nraw_Data = {'Date':['10/2/2011', '11/2/2011', '12/2/2011
', '13/2/2011'],\n'Product':['Umbrella', 'Matress', 'Badminton','Shuttle'],\n'Price':[1250, 1450, 1550, 400],\n
'Expense': [ 21525220.653, 31125840.875, 23135428.768, 56245263.942]}\na- Add Index as Item1, Item2, Item3, Ite
m4\nb- Find the index labels of all items whose 'Price' is greater than 1000.\nc- Replace products using Map()
with respective codes- Umbrella : 'U', Matress : 'M', Badminton\n: 'B', Shuttle: 'S'\nd- Round off the Expense
column values to two decimal places.\ne- Create a new column called 'Discounted_Price' after applying a 10% dis
count on the existing\n'price' column.(try using lambda function)\nf- Convert the column type of "Date" to date
time format.\ng- Create a column rank which ranks the products based on the price (one with the highest price\n
will be rank 1).\n"
```

In [7]:
```
# Ans3
# Creating a dataframe from given dataset
raw_Data = {'Date':['10/2/2011', '11/2/2011', '12/2/2011', '13/2/2011'],
            'Product':['Umbrella', 'Matress', 'Badminton','Shuttle'],
            'Price':[1250, 1450, 1550, 400],
            'Expense': [ 21525220.653, 31125840.875, 23135428.768, 56245263.942]}
df = pd.DataFrame(raw_Data)
print(df)

# a- Add Index as Item1, Item2, Item3, Item4
df.index = ['Item1','Item2','Item3','Item4']
print(df)

#b- Find the index labels of all items whose 'Price' is greater than 1000.
X = list(df[df['Price']>1000].index)
print('items whose 'Price' is greater than 1000: ',X)

#c- Replace products using Map() with respective codes- Umbrella : 'U', Matress : 'M', Badminton: 'B', Shuttle:
Product_Code = {'Umbrella':'U','Matress':'M','Badminton':'B','Shuttle':'S'}
df['Product'] = df['Product'].map(Product_Code)
print(df)

#d- Round off the Expense column values to two decimal places.
df['Expense'] = df['Expense'].round(2)
print(df)

#e- Create a new column called 'Discounted_Price' after applying a 10% discount on the existing
#   'price' column.(try using lambda function)
x = list(df['Price'])
a = lambda i:i-(i*0.1)
df['Dicounted_Price'] = list(map(a,x))
print(df)

#f- Convert the column type of "Date" to datetime format.
```

```
from datetime import datetime
df['Date'] = pd.to_datetime(df['Date'],format="%d/%m/%Y")

#g- Create a column rank which ranks the products based on the price (one with the highest price
# will be rank 1).
df['Rank'] = df['Price'].rank(ascending=False).astype(int)
print(df)
```

```
       Date    Product  Price       Expense
0  10/2/2011   Umbrella   1250  2.152522e+07
1  11/2/2011    Matress   1450  3.112584e+07
2  12/2/2011  Badminton   1550  2.313543e+07
3  13/2/2011    Shuttle    400  5.624526e+07
            Date    Product  Price       Expense
Item1  10/2/2011   Umbrella   1250  2.152522e+07
Item2  11/2/2011    Matress   1450  3.112584e+07
Item3  12/2/2011  Badminton   1550  2.313543e+07
Item4  13/2/2011    Shuttle    400  5.624526e+07
items whose ‘Price’ is greater than 1000:  ['Item1', 'Item2', 'Item3']
            Date Product  Price       Expense
Item1  10/2/2011       U   1250  2.152522e+07
Item2  11/2/2011       M   1450  3.112584e+07
Item3  12/2/2011       B   1550  2.313543e+07
Item4  13/2/2011       S    400  5.624526e+07
            Date Product  Price      Expense
Item1  10/2/2011       U   1250  21525220.65
Item2  11/2/2011       M   1450  31125840.88
Item3  12/2/2011       B   1550  23135428.77
Item4  13/2/2011       S    400  56245263.94
            Date Product  Price      Expense  Dicounted_Price
Item1  10/2/2011       U   1250  21525220.65           1125.0
Item2  11/2/2011       M   1450  31125840.88           1305.0
Item3  12/2/2011       B   1550  23135428.77           1395.0
Item4  13/2/2011       S    400  56245263.94            360.0
            Date Product  Price      Expense  Dicounted_Price  Rank
Item1 2011-02-10       U   1250  21525220.65           1125.0     3
Item2 2011-02-11       M   1450  31125840.88           1305.0     2
Item3 2011-02-12       B   1550  23135428.77           1395.0     1
Item4 2011-02-13       S    400  56245263.94            360.0     4
```

Assignment: Exploring NBA Player Data

'''Download the nba.csv file containing NBA player data Complete the following tasks using Python, Pandas, and data visualization libraries:

1. Load Data:

● Load the nba.csv data into a Pandas DataFrame. ● Display basic information about the DataFrame. 2. Data Cleaning: ● Handle missing values by either removing or imputing them. ● Remove duplicate rows. 3. Data Transformation: ● Create a new column 'BMI' (Body Mass Index) using the formula: BMI = (weight in pounds / (height in inches)^2) * 703.(Assuming a fixed height value of 70 inches (5 feet 10 inches) 4. Exploratory Data Analysis (EDA): ● Display summary statistics of the 'age', 'weight', and 'salary' columns. ● Calculate the average age, weight, and salary of players in each 'position' category. 5. Data Visualization: ● Create a histogram of player ages. ● Create a boxplot of player salaries for each 'position'. ● Plot a scatter plot of 'age' vs. 'salary' with a different color for each 'position'. 6. Top Players: ● Display the top 10 players with the highest salaries. 7. College Analysis: ● Determine the top 5 colleges with the most represented players. 8. Position Distribution: Grow Data Skil s ● Plot a pie chart to show the distribution of players across different 'positions'. 9. Team Analysis: ● Display the average salary of players for each 'team'. ● Plot a bar chart to visualize the average salary of players for each 'team'. 10. Extras ● Get the index at which the minimum weight value is present. ● Sort values based on name in alphabetical order for the rows (the original Dataframe sorting should not change) ● Create a series from given dataframe on “name” column and display top and last 10

Guidelines:

1. Write Python code to complete each task.
2. Provide comments explaining your code.
3. Use meaningful variable names.
4. Include necessary library imports.
5. Present your findings in a clear and organized manner.
6. Feel free to use additional code cells for each task.'''

Loading Data

In [8]:
```
df = pd.read_csv(r"C:\Users\hites\OneDrive\Desktop\Data analyst Bootcamp\Python\Class 4\user_course_file_648da6
```

Basic info about Data

In [9]:
```
df.head(10)
```

Out[9]:

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

Out[10]:

	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 [11]:
```
df.shape
```

Out[11]:
```
(457, 8)
```

In [12]:
```
df.describe()
```

Out[12]:

	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 [13]:
```
df.columns
```

Out[13]:
```
Index(['Name', 'Team', 'Number', 'Position', 'Age', 'Weight', 'College',
       'Salary'],
      dtype='object')
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 457 entries, 0 to 456
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      457 non-null    object 
 1   Team      457 non-null    object 
 2   Number    457 non-null    int64  
 3   Position  457 non-null    object 
 4   Age       457 non-null    int64  
 5   Weight    457 non-null    int64  
 6   College   373 non-null    object 
 7   Salary    446 non-null    float64
dtypes: float64(1), int64(3), object(4)
memory usage: 28.7+ KB
```

Column name 'College' and 'Salary' has Null values

Data Cleaning

Finding column name and null values

In [15]:
```
pd.isnull(df).sum()
```

Out[15]:
```
Name         0
Team         0
Number       0
Position     0
Age          0
Weight       0
College     84
Salary      11
dtype: int64
```

Column Name 'Collage' and 'Salary' has null values (84,11)

In [16]:
```
df[df['Salary'].isnull()]
```

Out[16]:

	Name	Team	Number	Position	Age	Weight	College	Salary
2	John Holland	Boston Celtics	30	SG	27	205	Boston University	NaN
46	Elton Brand	Philadelphia 76ers	42	PF	37	254	Duke	NaN
171	Dahntay Jones	Cleveland Cavaliers	30	SG	35	225	Duke	NaN
264	Jordan Farmar	Memphis Grizzlies	4	PG	29	180	UCLA	NaN
269	Ray McCallum	Memphis Grizzlies	5	PG	24	190	Detroit	NaN
270	Xavier Munford	Memphis Grizzlies	14	PG	24	180	Rhode Island	NaN
273	Alex Stepheson	Memphis Grizzlies	35	PF	28	270	USC	NaN
350	Briante Weber	Miami Heat	12	PG	23	165	Virginia Commonwealth	NaN
353	Dorell Wright	Miami Heat	11	SF	30	205	NaN	NaN
397	Axel Toupane	Denver Nuggets	6	SG	23	210	NaN	NaN
409	Greg Smith	Minnesota Timberwolves	4	PF	25	250	Fresno State	NaN

In [17]:
```
df[df['College'].isnull()]
```

Out[17]:

	Name	Team	Number	Position	Age	Weight	College	Salary
4	Jonas Jerebko	Boston Celtics	8	PF	29	231	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90	PF	29	240	NaN	12000000.0
15	Bojan Bogdanovic	Brooklyn Nets	44	SG	27	216	NaN	3425510.0
20	Sergey Karasev	Brooklyn Nets	10	SG	22	208	NaN	1599840.0
32	Thanasis Antetokounmpo	New York Knicks	43	SF	23	205	NaN	30888.0
...	...	...	...	...	...	...	...	...
445	Dante Exum	Utah Jazz	11	PG	20	190	NaN	3777720.0
447	Rudy Gobert	Utah Jazz	27	C	23	245	NaN	1175880.0
450	Joe Ingles	Utah Jazz	2	SF	28	226	NaN	2050000.0
454	Raul Neto	Utah Jazz	25	PG	24	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21	C	26	256	NaN	2900000.0

84 rows × 8 columns

Handling Null Values:

1. I am not going to remove null value rows because it can remove some important player profile which we need later for our analysis.
2. I am going to replace 'College' null values with Other.
3. And replace 'Salary' null values with mean salary.

In [18]:
```
df['College'].fillna('Other',inplace=True)
```

In [19]:
```
df['College'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 457 entries, 0 to 456
Series name: College
Non-Null Count  Dtype
--------------  -----
457 non-null    object
dtypes: object(1)
memory usage: 3.7+ KB
```

In [20]:
```
df.loc[454,'College']
```

Out[20]:
```
'Other'
```

In [21]:
```
df['Salary'].fillna(df['Salary'].mean(),inplace=True)
```

In [22]:
```
df['Salary'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 457 entries, 0 to 456
Series name: Salary
Non-Null Count  Dtype
--------------  -----
457 non-null    float64
dtypes: float64(1)
memory usage: 3.7 KB
```

In [23]:
```
df.loc[2,'Salary']
```

Out[23]:
```
4842684.105381166
```

In [24]:
```
df.isnull().sum()
```

Out[24]:
```
Name        0
Team        0
Number      0
Position    0
Age         0
Weight      0
College     0
Salary      0
dtype: int64
```

In [36]:
```
df.duplicated().sum()
```

Out[36]:
```
0
```

No Duplicate rows available in dataset

Data Transformation

In [38]:
```
# BMI = (weight in pounds / (height in inches)^2) * 703
df['BMI'] = (df['Weight']/70**2)*703
df
```

Out[38]:

	Name	Team	Number	Position	Age	Weight	College	Salary	BMI
0	Avery Bradley	Boston Celtics	0	PG	25	180	Texas	7.730337e+06	25.824490
1	Jae Crowder	Boston Celtics	99	SF	25	235	Marquette	6.796117e+06	33.715306
2	John Holland	Boston Celtics	30	SG	27	205	Boston University	4.842684e+06	29.411224
3	R.J. Hunter	Boston Celtics	28	SG	22	185	Georgia State	1.148640e+06	26.541837
4	Jonas Jerebko	Boston Celtics	8	PF	29	231	Other	5.000000e+06	33.141429
...	...	...	...	...	...	...	...	...	...
452	Trey Lyles	Utah Jazz	41	PF	20	234	Kentucky	2.239800e+06	33.571837
453	Shelvin Mack	Utah Jazz	8	PG	26	203	Butler	2.433333e+06	29.124286
454	Raul Neto	Utah Jazz	25	PG	24	179	Other	9.000000e+05	25.681020
455	Tibor Pleiss	Utah Jazz	21	C	26	256	Other	2.900000e+06	36.728163
456	Jeff Withey	Utah Jazz	24	C	26	231	Kansas	9.472760e+05	33.141429

457 rows × 9 columns

Exploratory Data Analysis (EDA)

- Summary statistics display of the 'age', 'weight', and 'salary' columns.

In [47]:
```
df[['Age','Weight','Salary']].describe()
```

Out[47]:

	Age	Weight	Salary
count	457.000000	457.000000	4.570000e+02
mean	26.938731	221.522976	4.842684e+06
std	4.404016	26.368343	5.165781e+06
min	19.000000	161.000000	3.088800e+04
25%	24.000000	200.000000	1.100602e+06
50%	26.000000	220.000000	2.869440e+06
75%	30.000000	240.000000	6.331404e+06
max	40.000000	307.000000	2.500000e+07

- Average age, weight, and salary calculation of players in each 'position' category.

In [58]:
```
Pos_grp = df.groupby('Position')
Avg_age = Pos_grp['Age'].mean()
print(Avg_age)
```

```
Position
C     27.371795
PF    27.160000
PG    26.847826
SF    26.858824
SG    26.539216
Name: Age, dtype: float64
```

In [60]:
```
Avg_Weight = Pos_grp['Weight'].mean()
print(Avg_Weight)
```

```
Position
C     254.205128
PF    240.430000
PG    189.478261
SF    221.776471
SG    206.686275
Name: Weight, dtype: float64
```

In [61]:
```
Avg_Salary = Pos_grp['Salary'].mean()
print(Avg_Salary)
```

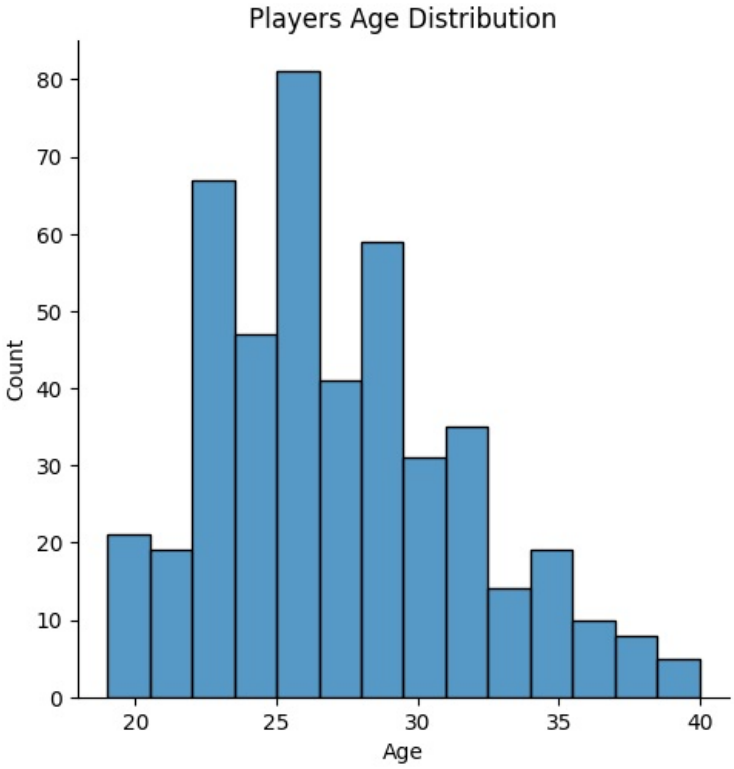
```
Position
C     5.967052e+06
PF    4.570889e+06
PG    5.067606e+06
SF    4.857220e+06
SG    4.034356e+06
Name: Salary, dtype: float64
```

Data Visualization

- Histogram of player ages.

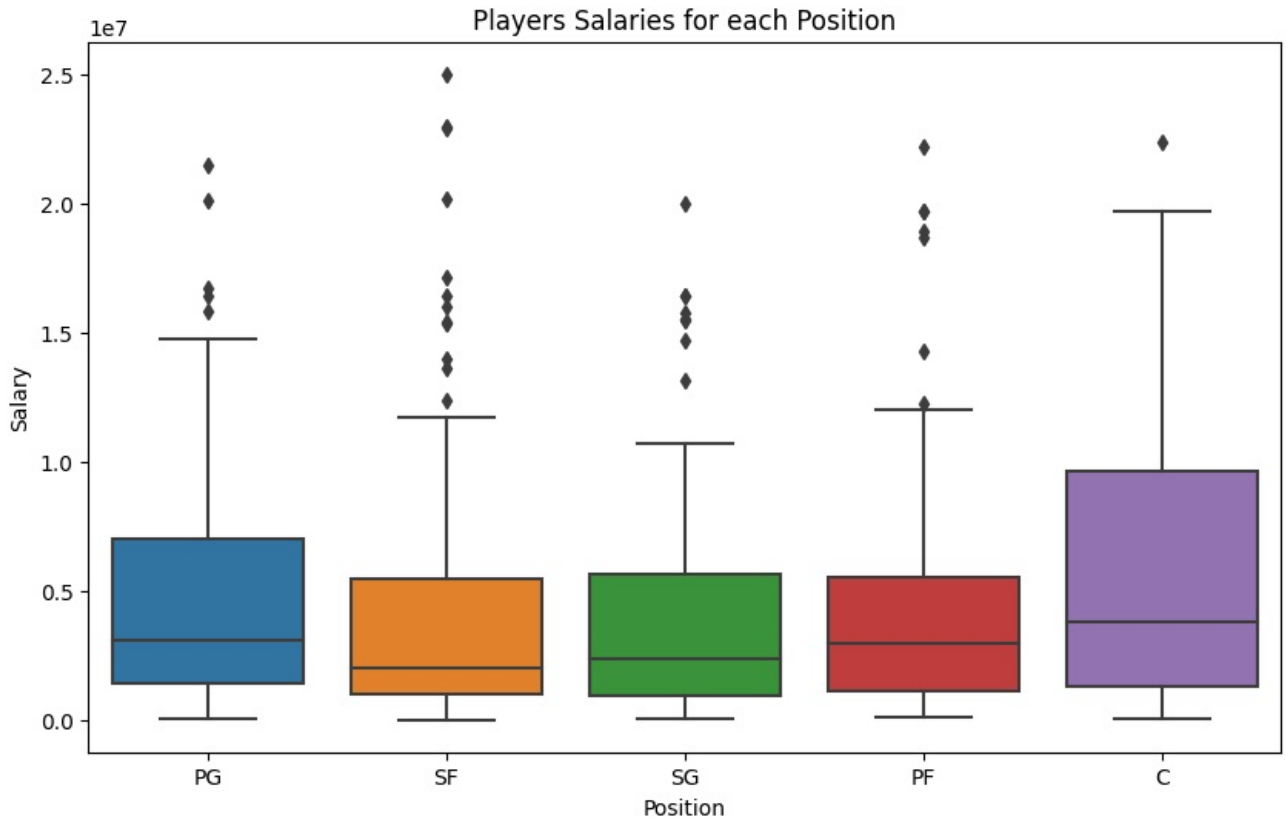
In [89]:
```
sns.displot(df['Age'])
plt.title('Players Age Distribution')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



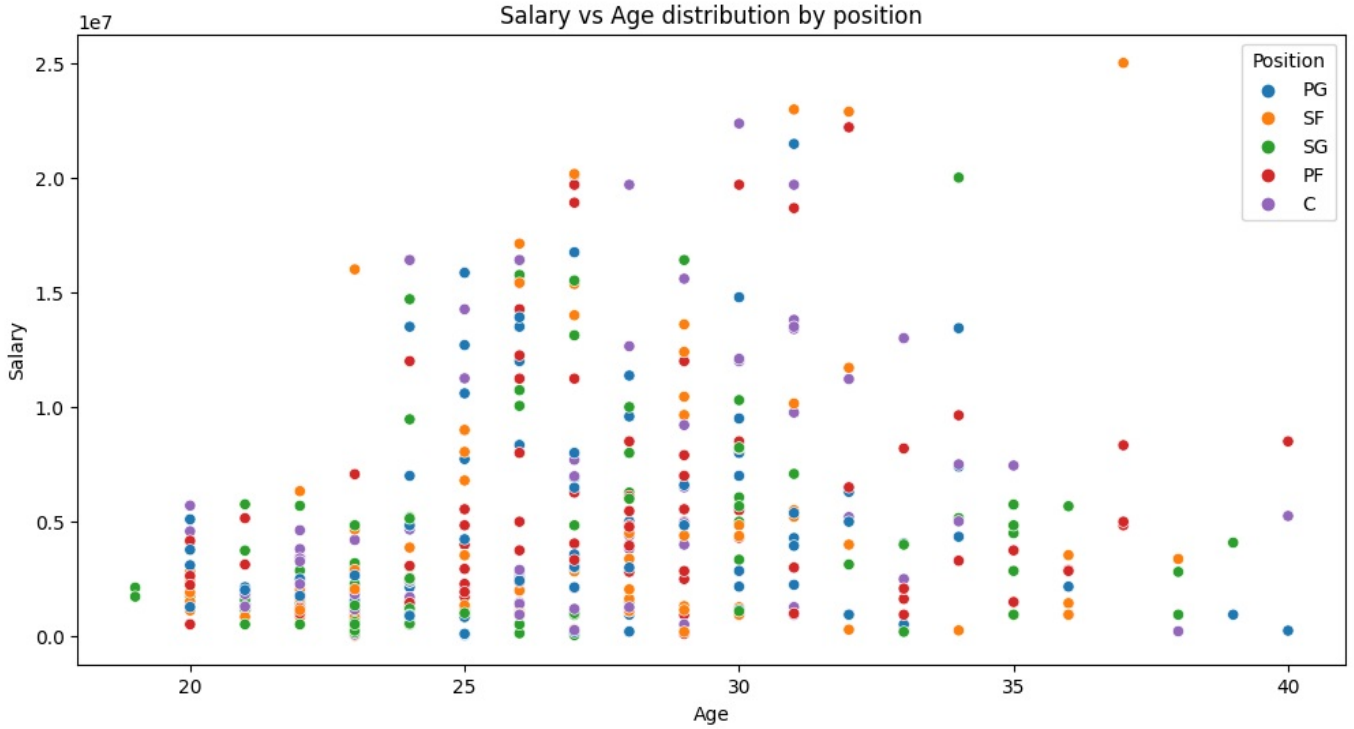
- Boxplot of player salaries for each 'position'

In [90]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Position'],y=df['Salary'],data=df)
plt.title('Players Salaries for each Position')
plt.show()
```



- Scatter plot of 'age' vs. 'salary' with a different color for each 'position'.

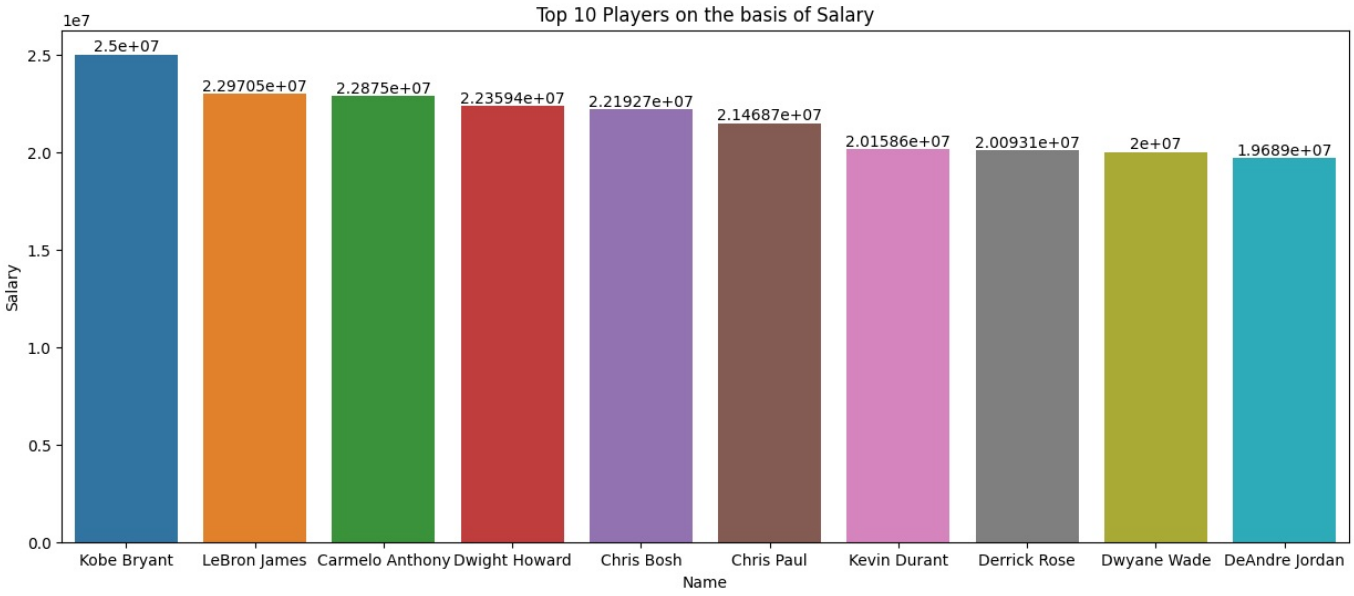
In [88]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=df['Age'],y=df['Salary'],data=df,hue=df['Position'])
plt.title('Salary vs Age distribution by position')
plt.show()
```



Top Players

- Top 10 players with the highest salaries.

In [104…
```
Top_10_Salaries_df = df.sort_values(by='Salary',ascending=False).head(10)
plt.figure(figsize=(15,6))
x = sns.barplot(x='Name',y='Salary',data=Top_10_Salaries_df)
for i in x.containers:
    x.bar_label(i)
plt.title('Top 10 Players on the basis of Salary')
plt.show()
```



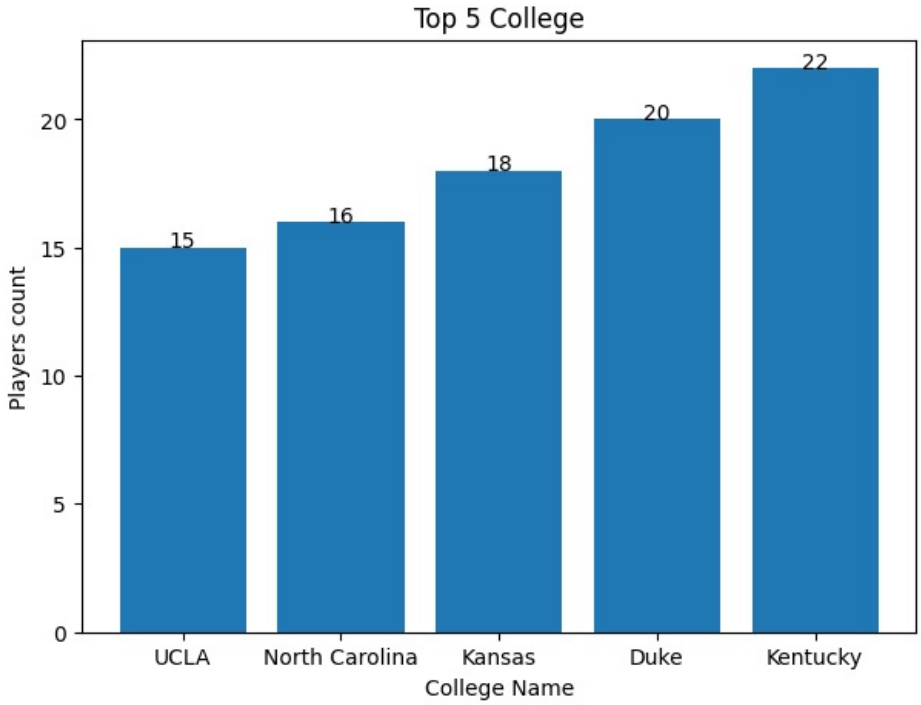
Top Colleges

- Top 5 colleges with the most represented players.

In [138…

```
Clg_grp = df.groupby('College')
Top_clg = Clg_grp['College'].count().sort_values().tail(6)
# Removing College name: 'Other'
Top_clg.drop(['Other'],axis=0,inplace=True)
y = list(Top_clg.values)
x = list(Top_clg.keys())
plt.figure(figsize=(7,5))
plt.bar(x,y)
plt.xlabel('College Name')
plt.ylabel('Players count')
plt.title('Top 5 College')
for i in range(len(x)):
    plt.text(i,y[i],y[i],ha='center')

plt.show()
```



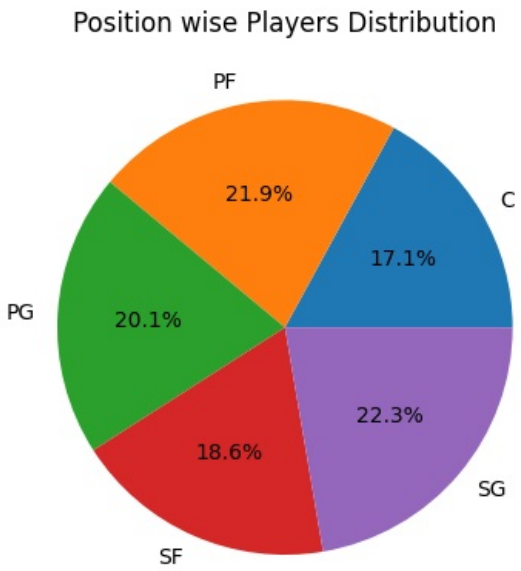
Position Distribution

- Pie chart to showing the distribution of players across different 'positions'.

In [156…

```
Pos_grp = df.groupby('Position')['Name'].count()
Label = list(Pos_grp.keys())
Value = list(Pos_grp.values)
plt.pie(Value,labels=Label,autopct='%1.1f%%')
plt.title('Position wise Players Distribution')
```

```
plt.show()
```



Team Analysis:

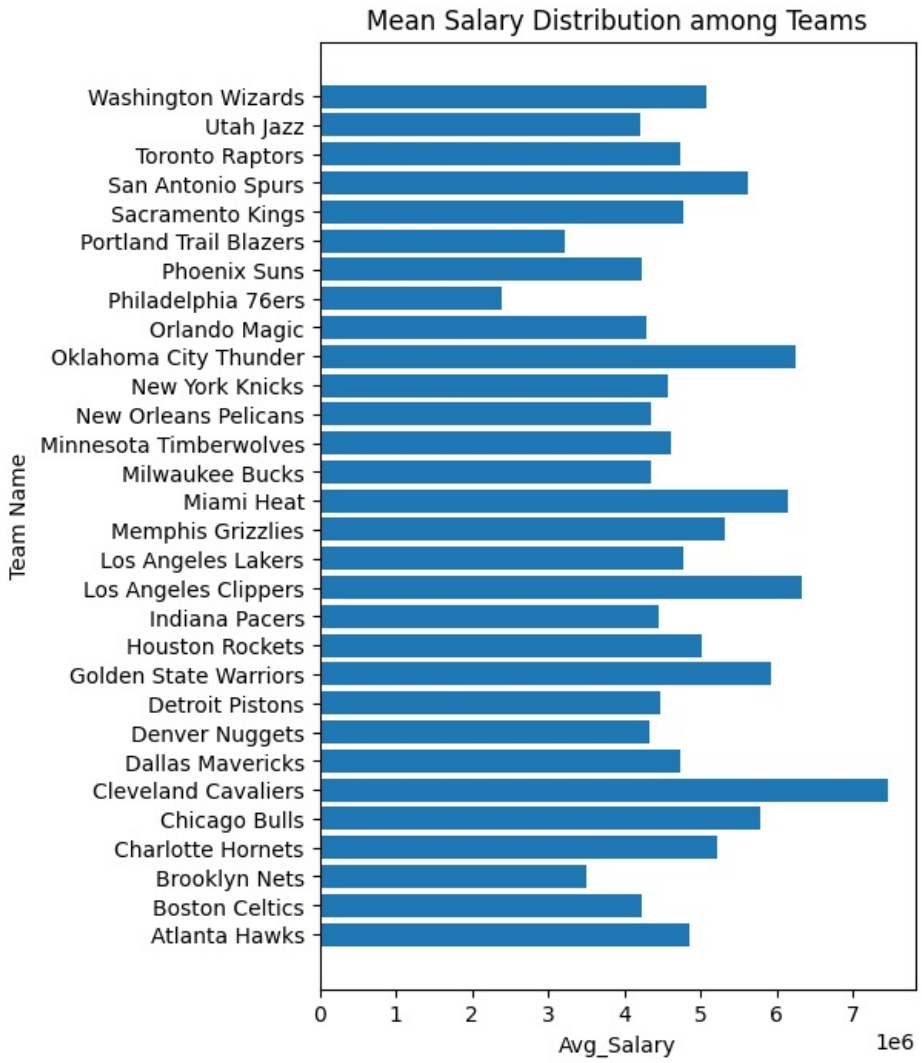
- Displaying average salary of players for each 'team'

In [159…
```
Team_grp = df.groupby('Team')
Avg_salary = Team_grp['Salary'].mean()
# Note: We used mean salary in place of null salary.
```

```
Team
Atlanta Hawks             4.860197e+06
Boston Celtics            4.225583e+06
Brooklyn Nets             3.501898e+06
Charlotte Hornets         5.222728e+06
Chicago Bulls             5.785559e+06
Cleveland Cavaliers       7.455425e+06
Dallas Mavericks          4.746582e+06
Denver Nuggets            4.330974e+06
Detroit Pistons           4.477884e+06
Golden State Warriors     5.924600e+06
Houston Rockets           5.018868e+06
Indiana Pacers            4.450122e+06
Los Angeles Clippers      6.323643e+06
Los Angeles Lakers        4.784695e+06
Memphis Grizzlies         5.328979e+06
Miami Heat                6.146736e+06
Milwaukee Bucks           4.350220e+06
Minnesota Timberwolves    4.610884e+06
New Orleans Pelicans      4.355304e+06
New York Knicks           4.581494e+06
Oklahoma City Thunder     6.251020e+06
Orlando Magic             4.297248e+06
Philadelphia 76ers        2.389039e+06
Phoenix Suns              4.229676e+06
Portland Trail Blazers    3.220121e+06
Sacramento Kings          4.778911e+06
San Antonio Spurs         5.629516e+06
Toronto Raptors           4.741174e+06
Utah Jazz                 4.204006e+06
Washington Wizards        5.088576e+06
Name: Salary, dtype: float64
```

- Bar chart showing average salary of players for each 'team'.

In [163…
```
x = list(Avg_salary.keys())
y = list(Avg_salary.values)
plt.figure(figsize=(5,8))
plt.barh(x,y)
plt.xlabel('Avg_Salary')
plt.ylabel('Team Name')
plt.title('Mean Salary Distribution among Teams')
plt.show()
```



Other Analysis

- Finding index at which the minimum weight value is present.

In [179…
```
print('Minimum Weight in pounds:',df['Weight'].min())
index_no = 0
for i in df['Weight']:
    if i == df['Weight'].min():
        break
    index_no +=1
print('Min weight index no.',index_no)
df[152:153]
```

```
Minimum Weight in pounds: 161
Min weight index no. 152
```

Out[179]:

	Name	Team	Number	Position	Age	Weight	College	Salary	BMI
152	Aaron Brooks	Chicago Bulls	0	PG	31	161	Oregon	2250000.0	23.098571

- Sorting values based on name in alphabetical order for the rows (the original Dataframe sorting not going to change)

In [184…
```
df1 = df.sort_values(by='Name',ascending=True)
print(df.head())
print('-------------------------------------------------')
print(df1.head())
```

```
            Name            Team  Number Position  Age  Weight
0  Avery Bradley  Boston Celtics       0       PG   25     180  \
1    Jae Crowder  Boston Celtics      99       SF   25     235
2   John Holland  Boston Celtics      30       SG   27     205
3    R.J. Hunter  Boston Celtics      28       SG   22     185
4  Jonas Jerebko  Boston Celtics       8       PF   29     231

             College        Salary        BMI
0              Texas  7.730337e+06  25.824490
1          Marquette  6.796117e+06  33.715306
2  Boston University  4.842684e+06  29.411224
3      Georgia State  1.148640e+06  26.541837
4              Other  5.000000e+06  33.141429
------------------------------------------------
              Name                    Team  Number Position  Age  Weight
152    Aaron Brooks           Chicago Bulls       0       PG   31     161  \
356    Aaron Gordon           Orlando Magic       0       PF   20     220
328  Aaron Harrison       Charlotte Hornets       9       SG   21     210
404   Adreian Payne  Minnesota Timberwolves      33       PF   25     237
312      Al Horford           Atlanta Hawks      15        C   30     245

            College      Salary        BMI
152          Oregon   2250000.0  23.098571
356         Arizona   4171680.0  31.563265
328        Kentucky    525093.0  30.128571
404  Michigan State   1938840.0  34.002245
312         Florida  12000000.0  35.150000
```

- Creating a series from given dataframe on “name” column and display top and last 10

In [194…

```
Name = df['Name']
print('Top 10 Names')
print(Name.head(10))
print('---------------------------')
print('Last 10 Names')
print(Name.tail(10))
```

```
Top 10 Names
0    Avery Bradley
1      Jae Crowder
2     John Holland
3      R.J. Hunter
4    Jonas Jerebko
5     Amir Johnson
6    Jordan Mickey
7     Kelly Olynyk
8     Terry Rozier
9     Marcus Smart
Name: Name, dtype: object
---------------------------
Last 10 Names
447       Rudy Gobert
448    Gordon Hayward
449       Rodney Hood
450        Joe Ingles
451     Chris Johnson
452        Trey Lyles
453      Shelvin Mack
454         Raul Neto
455      Tibor Pleiss
456       Jeff Withey
Name: Name, dtype: object
```