```
In [130]: # Import pandas and numpy package
          import pandas as pd
          import numpy as np
```

Loading the data set ¶

In [131]:

```
#Reading csv to dataframe
df=pd.read_csv("dirtydata1.csv")
df
```

Out[131]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020/12/01'	110	130	409.1
1	60	2020/12/02'	117	145	479.0
2	60	2020/12/03'	103	135	340.0
3	45	2020/12/04'	109	175	282.4
4	45	2020/12/05'	117	148	406.0
5	60	2020/12/06'	102	127	-300.0
6	60	2020/12/07'	110	136	374.0
7	450	2020/12/08'	104	134	253.3
8	30	2020/12/09'	109	133	195.1
9	60	2020/12/10'	98	124	269.0
10	60	2020/12/11'	103	147	329.3
11	60	2020/12/12'	100	120	250.7
12	60	2020/12/12'	100	120	250.7
13	60	2020/12/13'	106	128	345.3
14	60	2020/12/14'	104	132	379.3
15	60	2020/12/15'	98	123	275.0
16	60	2020/12/16'	98	120	215.2
17	60	2020/12/17'	100	120	300.0
18	45	2020/12/18'	90	112	NaN
19	60	2020/12/19'	103	123	323.0
20	45	2020/12/20'	97	125	243.0
21	60	2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	2020/12/23'	130	101	300.0
24	45	2020/12/24'	105	132	246.0
25	60	2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	2020/12/27'	92	118	241.0
28	60	2020/12/28'	103	132	NaN
29	60	2020/12/29'	100	132	-280.0
30	60	2020/12/30'	102	129	380.3
31	60	2020/12/31'	92	115	243.0

In [9]:

```
#finding Missing Values and their count
nv=df.isnull().sum()
print(nv)
```

```
Duration    0
Date        1
Pulse       0
Maxpulse    0
Calories    2
dtype: int64
```

In [10]:

```
# dataframe shape to get the dimension of the dataset
df.shape
```

Out[10]:

```
(32, 5)
```

In [11]:

```
#description of the data = to give the details about the dataset
df.describe()
```

Out[11]:

	Duration	Pulse	Maxpulse	Calories
count	32.000000	32.000000	32.000000	30.000000
mean	68.437500	103.500000	128.500000	266.013333
std	70.039591	7.832933	12.998759	164.876415
min	30.000000	90.000000	101.000000	-300.000000
25%	60.000000	100.000000	120.000000	247.000000
50%	60.000000	102.500000	127.500000	282.200000
75%	60.000000	106.500000	132.250000	343.975000
max	450.000000	130.000000	175.000000	479.000000

In [12]:

```
#data types in the DataFrame
df.dtypes
```

Out[12]:

```
Duration      int64
Date         object
Pulse         int64
Maxpulse      int64
Calories    float64
dtype: object
```

In [13]:

```
#abs(): to remove the -ve values
df["Calories"]=df["Calories"].abs()
```

In [14]:

```
df
```

Out[14]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020/12/01'	110	130	409.1
1	60	2020/12/02'	117	145	479.0
2	60	2020/12/03'	103	135	340.0
3	45	2020/12/04'	109	175	282.4
4	45	2020/12/05'	117	148	406.0
5	60	2020/12/06'	102	127	300.0
6	60	2020/12/07'	110	136	374.0
7	450	2020/12/08'	104	134	253.3
8	30	2020/12/09'	109	133	195.1
9	60	2020/12/10'	98	124	269.0
10	60	2020/12/11'	103	147	329.3
11	60	2020/12/12'	100	120	250.7
12	60	2020/12/12'	100	120	250.7
13	60	2020/12/13'	106	128	345.3
14	60	2020/12/14'	104	132	379.3
15	60	2020/12/15'	98	123	275.0
16	60	2020/12/16'	98	120	215.2
17	60	2020/12/17'	100	120	300.0
18	45	2020/12/18'	90	112	NaN
19	60	2020/12/19'	103	123	323.0
20	45	2020/12/20'	97	125	243.0
21	60	2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	2020/12/23'	130	101	300.0
24	45	2020/12/24'	105	132	246.0
25	60	2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	2020/12/27'	92	118	241.0
28	60	2020/12/28'	103	132	NaN
29	60	2020/12/29'	100	132	280.0
30	60	2020/12/30'	102	129	380.3
31	60	2020/12/31'	92	115	243.0

In [20]:
```
#Take mean value of calories
x=df['Calories'].mean()
x
```

Out[20]:
```
304.68
```

In [21]:
```
#Fill every null value with value of x
df['Calories'].fillna(x,inplace=True)
```

In [25]:
```
#convert data type of calories from float to integer
df["Calories"]=df["Calories"].astype(int)
df
```

Out[25]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020/12/01'	110	130	409
1	60	2020/12/02'	117	145	479
2	60	2020/12/03'	103	135	340
3	45	2020/12/04'	109	175	282
4	45	2020/12/05'	117	148	406
5	60	2020/12/06'	102	127	300
6	60	2020/12/07'	110	136	374
7	450	2020/12/08'	104	134	253
8	30	2020/12/09'	109	133	195
9	60	2020/12/10'	98	124	269
10	60	2020/12/11'	103	147	329
11	60	2020/12/12'	100	120	250
12	60	2020/12/12'	100	120	250
13	60	2020/12/13'	106	128	345
14	60	2020/12/14'	104	132	379
15	60	2020/12/15'	98	123	275
16	60	2020/12/16'	98	120	215
17	60	2020/12/17'	100	120	300
18	45	2020/12/18'	90	112	304
19	60	2020/12/19'	103	123	323
20	45	2020/12/20'	97	125	243
21	60	2020/12/21'	108	131	364
22	45	NaN	100	119	282
23	60	2020/12/23'	130	101	300
24	45	2020/12/24'	105	132	246
25	60	2020/12/25'	102	126	334
26	60	20201226	100	120	250
27	60	2020/12/27'	92	118	241
28	60	2020/12/28'	103	132	304
29	60	2020/12/29'	100	132	280
30	60	2020/12/30'	102	129	380
31	60	2020/12/31'	92	115	243

In [26]:
```
df
```

Out[26]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020/12/01'	110	130	409
1	60	2020/12/02'	117	145	479
2	60	2020/12/03'	103	135	340
3	45	2020/12/04'	109	175	282
4	45	2020/12/05'	117	148	406
5	60	2020/12/06'	102	127	300
6	60	2020/12/07'	110	136	374
7	450	2020/12/08'	104	134	253
8	30	2020/12/09'	109	133	195
9	60	2020/12/10'	98	124	269
10	60	2020/12/11'	103	147	329
11	60	2020/12/12'	100	120	250
12	60	2020/12/12'	100	120	250
13	60	2020/12/13'	106	128	345
14	60	2020/12/14'	104	132	379
15	60	2020/12/15'	98	123	275
16	60	2020/12/16'	98	120	215
17	60	2020/12/17'	100	120	300
18	45	2020/12/18'	90	112	304
19	60	2020/12/19'	103	123	323
20	45	2020/12/20'	97	125	243
21	60	2020/12/21'	108	131	364
22	45	NaN	100	119	282
23	60	2020/12/23'	130	101	300
24	45	2020/12/24'	105	132	246
25	60	2020/12/25'	102	126	334
26	60	20201226	100	120	250
27	60	2020/12/27'	92	118	241
28	60	2020/12/28'	103	132	304
29	60	2020/12/29'	100	132	280
30	60	2020/12/30'	102	129	380
31	60	2020/12/31'	92	115	243

In [27]:
```
#removes all the rows of Date column that contains NULL values
df.dropna(subset='Date',inplace=True)
```

In [31]:
```
df
```

Out[31]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409
1	60	2020-12-02	117	145	479
2	60	2020-12-03	103	135	340
3	45	2020-12-04	109	175	282
4	45	2020-12-05	117	148	406
5	60	2020-12-06	102	127	300
6	60	2020-12-07	110	136	374
7	450	2020-12-08	104	134	253
8	30	2020-12-09	109	133	195
9	60	2020-12-10	98	124	269
10	60	2020-12-11	103	147	329
11	60	2020-12-12	100	120	250
12	60	2020-12-12	100	120	250
13	60	2020-12-13	106	128	345
14	60	2020-12-14	104	132	379
15	60	2020-12-15	98	123	275
16	60	2020-12-16	98	120	215
17	60	2020-12-17	100	120	300
18	45	2020-12-18	90	112	304
19	60	2020-12-19	103	123	323
20	45	2020-12-20	97	125	243
21	60	2020-12-21	108	131	364
23	60	2020-12-23	130	101	300
24	45	2020-12-24	105	132	246
25	60	2020-12-25	102	126	334
26	60	2020-12-26	100	120	250
27	60	2020-12-27	92	118	241
28	60	2020-12-28	103	132	304
29	60	2020-12-29	100	132	280
30	60	2020-12-30	102	129	380
31	60	2020-12-31	92	115	243

In [30]:

```
#Convert string data types Datecolumn to date type
df['Date']=pd.to_datetime(df['Date'])
df
```

Out[30]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409
1	60	2020-12-02	117	145	479
2	60	2020-12-03	103	135	340
3	45	2020-12-04	109	175	282
4	45	2020-12-05	117	148	406
5	60	2020-12-06	102	127	300
6	60	2020-12-07	110	136	374
7	450	2020-12-08	104	134	253
8	30	2020-12-09	109	133	195
9	60	2020-12-10	98	124	269
10	60	2020-12-11	103	147	329
11	60	2020-12-12	100	120	250
12	60	2020-12-12	100	120	250
13	60	2020-12-13	106	128	345
14	60	2020-12-14	104	132	379
15	60	2020-12-15	98	123	275
16	60	2020-12-16	98	120	215
17	60	2020-12-17	100	120	300
18	45	2020-12-18	90	112	304
19	60	2020-12-19	103	123	323
20	45	2020-12-20	97	125	243
21	60	2020-12-21	108	131	364
23	60	2020-12-23	130	101	300
24	45	2020-12-24	105	132	246
25	60	2020-12-25	102	126	334
26	60	2020-12-26	100	120	250
27	60	2020-12-27	92	118	241
28	60	2020-12-28	103	132	304
29	60	2020-12-29	100	132	280
30	60	2020-12-30	102	129	380
31	60	2020-12-31	92	115	243

In [32]:
```
#here we select the location and set the values
#Change 7th index's Duratoin column's value to 45
df.loc[7,'Duration']=45
```

In [33]:
```
df
```

Out[33]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409
1	60	2020-12-02	117	145	479
2	60	2020-12-03	103	135	340
3	45	2020-12-04	109	175	282
4	45	2020-12-05	117	148	406
5	60	2020-12-06	102	127	300
6	60	2020-12-07	110	136	374
7	45	2020-12-08	104	134	253
8	30	2020-12-09	109	133	195
9	60	2020-12-10	98	124	269
10	60	2020-12-11	103	147	329
11	60	2020-12-12	100	120	250
12	60	2020-12-12	100	120	250
13	60	2020-12-13	106	128	345
14	60	2020-12-14	104	132	379
15	60	2020-12-15	98	123	275
16	60	2020-12-16	98	120	215
17	60	2020-12-17	100	120	300
18	45	2020-12-18	90	112	304
19	60	2020-12-19	103	123	323
20	45	2020-12-20	97	125	243
21	60	2020-12-21	108	131	364
23	60	2020-12-23	130	101	300
24	45	2020-12-24	105	132	246
25	60	2020-12-25	102	126	334
26	60	2020-12-26	100	120	250
27	60	2020-12-27	92	118	241
28	60	2020-12-28	103	132	304
29	60	2020-12-29	100	132	280
30	60	2020-12-30	102	129	380
31	60	2020-12-31	92	115	243

In [36]:
```
#check for duplicate value
df.duplicated()
```

Out[36]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12     True
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
dtype: bool
```

In [37]:
```
#total numbers of duplicate rows
df.duplicated().sum()
```

Out[37]:
```
1
```

In [38]:
```
#remove duplicate data
df.drop_duplicates(inplace=True)
df
```

Out[38]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409
1	60	2020-12-02	117	145	479
2	60	2020-12-03	103	135	340
3	45	2020-12-04	109	175	282
4	45	2020-12-05	117	148	406
5	60	2020-12-06	102	127	300
6	60	2020-12-07	110	136	374
7	45	2020-12-08	104	134	253
8	30	2020-12-09	109	133	195
9	60	2020-12-10	98	124	269
10	60	2020-12-11	103	147	329
11	60	2020-12-12	100	120	250
13	60	2020-12-13	106	128	345
14	60	2020-12-14	104	132	379
15	60	2020-12-15	98	123	275
16	60	2020-12-16	98	120	215
17	60	2020-12-17	100	120	300
18	45	2020-12-18	90	112	304
19	60	2020-12-19	103	123	323
20	45	2020-12-20	97	125	243
21	60	2020-12-21	108	131	364
23	60	2020-12-23	130	101	300
24	45	2020-12-24	105	132	246
25	60	2020-12-25	102	126	334
26	60	2020-12-26	100	120	250
27	60	2020-12-27	92	118	241
28	60	2020-12-28	103	132	304
29	60	2020-12-29	100	132	280
30	60	2020-12-30	102	129	380
31	60	2020-12-31	92	115	243

In [93]:
```
#to save file after preprocessing
df.to_csv("dirty_preposseddata.csv")
```

New dataset : nba.csv

In [70]:

```
#Reading csv to dataframe
df_nba=pd.read_csv("nba.csv")
df_nba
```

Out[70]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	2-Jun	180	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6-Jun	235	Marquette	6796117.0
2	John Holland	Boston Celtics	30	SG	27	5-Jun	205	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	5-Jun	185	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	10-Jun	231	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
452	Trey Lyles	Utah Jazz	41	PF	20	10-Jun	234	Kentucky	2239800.0
453	Shelvin Mack	Utah Jazz	8	PG	26	3-Jun	203	Butler	2433333.0
454	Raul Neto	Utah Jazz	25	PG	24	1-Jun	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21	C	26	3-Jul	256	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24	C	26	Jul-00	231	Kansas	947276.0

457 rows × 9 columns

In [61]:

```
#to check the dimension of dataset
df_nba.shape
```

Out[61]:

```
(457, 9)
```

In [62]:

```
df_nba.describe()  #description of the data
```

Out[62]:

	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 [63]:
```
#to check the data types of the dataset
df_nba.dtypes
```

Out[63]:
```
Name         object
Team         object
Number        int64
Position     object
Age           int64
Height       object
Weight        int64
College      object
Salary      float64
dtype: object
```

In [64]:
```
    #null values in the dataset
df_nba.isnull()
```

Out[64]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	True
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...
452	False	False	False	False	False	False	False	False	False
453	False	False	False	False	False	False	False	False	False
454	False	False	False	False	False	False	False	True	False
455	False	False	False	False	False	False	False	True	False
456	False	False	False	False	False	False	False	False	False

457 rows × 9 columns

In [69]:
```
df_nba.isnull().sum()  #to check the null values columnwise and find their count
```

Out[69]:
```
Name         0
Team         0
Number       0
Position     0
Age          0
Height       0
Weight       0
College     84
Salary       0
dtype: int64
```

In [67]:
```
#mean of salary column
y=df_nba['Salary'].mean()
print(y)
```

```
4842684.105381166
```

In [71]:

```
    #filling NaN values in salary with constant value
#The fillna() method replaces the NULL values with a specified value
#inplace=True keyword in a pandas method changes the default behaviour
df_nba['Salary'].fillna(780000,inplace=True)
df_nba
```

Out[71]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	2-Jun	180	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6-Jun	235	Marquette	6796117.0
2	John Holland	Boston Celtics	30	SG	27	5-Jun	205	Boston University	780000.0
3	R.J. Hunter	Boston Celtics	28	SG	22	5-Jun	185	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	10-Jun	231	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
452	Trey Lyles	Utah Jazz	41	PF	20	10-Jun	234	Kentucky	2239800.0
453	Shelvin Mack	Utah Jazz	8	PG	26	3-Jun	203	Butler	2433333.0
454	Raul Neto	Utah Jazz	25	PG	24	1-Jun	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21	C	26	3-Jul	256	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24	C	26	Jul-00	231	Kansas	947276.0

457 rows × 9 columns

In [72]:

```
#converts all the negative values to positive values
df_nba['Salary']=df_nba['Salary'].abs()
```

In [73]:

```
df_nba
```

Out[73]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	2-Jun	180	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6-Jun	235	Marquette	6796117.0
2	John Holland	Boston Celtics	30	SG	27	5-Jun	205	Boston University	780000.0
3	R.J. Hunter	Boston Celtics	28	SG	22	5-Jun	185	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	10-Jun	231	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
452	Trey Lyles	Utah Jazz	41	PF	20	10-Jun	234	Kentucky	2239800.0
453	Shelvin Mack	Utah Jazz	8	PG	26	3-Jun	203	Butler	2433333.0
454	Raul Neto	Utah Jazz	25	PG	24	1-Jun	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21	C	26	3-Jul	256	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24	C	26	Jul-00	231	Kansas	947276.0

457 rows × 9 columns

In [74]:

```
#unique values is found out in position column
df_nba['Position'].unique()
```

Out[74]:

```
array(['PG', 'SF', 'SG', 'PF', 'C'], dtype=object)
```

In [76]:

```
df_nba.duplicated().sum()
```

Out[76]:

```
0
```

In [80]:

```
df_nba['Position'].value_counts()
```

Out[80]:

```
SG    102
PF    100
PG     92
SF     85
C      78
Name: Position, dtype: int64
```

In [77]:

```
#converted to quantitative
df_nba['Pos']=df_nba['Position'].replace(['SG','PF','PG','SF','C'],[1,2,3,4,5])
```

In [78]:
```
df_nba
```

Out[78]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary	Pos
0	Avery Bradley	Boston Celtics	0	PG	25	2-Jun	180	Texas	7730337.0	3
1	Jae Crowder	Boston Celtics	99	SF	25	6-Jun	235	Marquette	6796117.0	4
2	John Holland	Boston Celtics	30	SG	27	5-Jun	205	Boston University	780000.0	1
3	R.J. Hunter	Boston Celtics	28	SG	22	5-Jun	185	Georgia State	1148640.0	1
4	Jonas Jerebko	Boston Celtics	8	PF	29	10-Jun	231	NaN	5000000.0	2
...	...	...	...	...	...	...	...	...	...	...
452	Trey Lyles	Utah Jazz	41	PF	20	10-Jun	234	Kentucky	2239800.0	2
453	Shelvin Mack	Utah Jazz	8	PG	26	3-Jun	203	Butler	2433333.0	3
454	Raul Neto	Utah Jazz	25	PG	24	1-Jun	179	NaN	900000.0	3
455	Tibor Pleiss	Utah Jazz	21	C	26	3-Jul	256	NaN	2900000.0	5
456	Jeff Withey	Utah Jazz	24	C	26	Jul-00	231	Kansas	947276.0	5

457 rows × 10 columns

In [82]:
```
#categarial to quantitative using label encoding
from sklearn import preprocessing
l_en=preprocessing.LabelEncoder()
```

In [85]:
```
df_nba['Position']=l_en.fit_transform(df_nba['Position'])  #Fit label encoder and r
print(df_nba)
```

```
              Name            Team  Number  Position  Age Height  Weight  \
0    Avery Bradley  Boston Celtics       0         2   25  2-Jun     180   
1      Jae Crowder  Boston Celtics      99         3   25  6-Jun     235   
2     John Holland  Boston Celtics      30         4   27  5-Jun     205   
3      R.J. Hunter  Boston Celtics      28         4   22  5-Jun     185   
4    Jonas Jerebko  Boston Celtics       8         1   29 10-Jun     231   
..             ...             ...     ...       ...  ...    ...     ...   
452     Trey Lyles       Utah Jazz      41         1   20 10-Jun     234   
453   Shelvin Mack       Utah Jazz       8         2   26  3-Jun     203   
454      Raul Neto       Utah Jazz      25         2   24  1-Jun     179   
455   Tibor Pleiss       Utah Jazz      21         0   26  3-Jul     256   
456    Jeff Withey       Utah Jazz      24         0   26 Jul-00     231   

               College     Salary  Pos  
0                Texas  7730337.0    3  
1            Marquette  6796117.0    4  
2    Boston University   780000.0    1  
3        Georgia State  1148640.0    1  
4                  NaN  5000000.0    2  
..                 ...        ...  ...  
452           Kentucky  2239800.0    2  
453             Butler  2433333.0    3  
454                NaN   900000.0    3  
455                NaN  2900000.0    5  
456             Kansas   947276.0    5  

[457 rows x 10 columns]
```

In [87]:
```
df_nba['Age'].unique()  #finding unique values of Age column
```

Out[87]:
```
array([25, 27, 22, 29, 21, 24, 20, 26, 28, 32, 23, 30, 33, 34, 37, 36, 31,
       38, 39, 19, 35, 40], dtype=int64)
```

In [88]:
```
#quatitative to categarical in python using pandas
category=pd.cut(df_nba.Age,bins=[19,25,30,35,45],labels=['A','B','C','D'])
print(category)
```

```
0      A
1      A
2      B
3      A
4      B
      ..
452    A
453    B
454    A
455    B
456    B
Name: Age, Length: 457, dtype: category
Categories (4, object): ['A' < 'B' < 'C' < 'D']
```

In [90]:

```
#to insert new column in existing dataset
df_nba.insert(5,"Age_group",category)  #here 5 is the column position
df_nba
```

Out[90]:

	Name	Team	Number	Position	Age	Age_group	Height	Weight	College	Salary	Pos
0	Avery Bradley	Boston Celtics	0	2	25	A	2-Jun	180	Texas	7730337.0	3
1	Jae Crowder	Boston Celtics	99	3	25	A	6-Jun	235	Marquette	6796117.0	4
2	John Holland	Boston Celtics	30	4	27	B	5-Jun	205	Boston University	780000.0	1
3	R.J. Hunter	Boston Celtics	28	4	22	A	5-Jun	185	Georgia State	1148640.0	1
4	Jonas Jerebko	Boston Celtics	8	1	29	B	10-Jun	231	NaN	5000000.0	2
...	...	...	...	...	...	...	...	...	...	...	...
452	Trey Lyles	Utah Jazz	41	1	20	A	10-Jun	234	Kentucky	2239800.0	2
453	Shelvin Mack	Utah Jazz	8	2	26	B	3-Jun	203	Butler	2433333.0	3
454	Raul Neto	Utah Jazz	25	2	24	A	1-Jun	179	NaN	900000.0	3
455	Tibor Pleiss	Utah Jazz	21	0	26	B	3-Jul	256	NaN	2900000.0	5
456	Jeff Withey	Utah Jazz	24	0	26	B	Jul-00	231	Kansas	947276.0	5

457 rows × 11 columns

New dataset : A1_ALCHOHOL

In [108]:

```
#reading csv to dataframe
df_alc=pd.read_csv("A1_ALCHOHOL.csv")
df_alc
```

Out[108]:

	Country	Alcohol	Deaths	Heart	Liver
0	Australia	2.50	785	211.0	15.300000
1	Austria	3.00	863	167.0	45.599998
2	Belg. and Lux.	2.90	883	131.0	20.700001
3	Canada	2.40	793	NaN	16.400000
4	Denmark	2.90	971	220.0	23.900000
5	Finland	0.80	970	297.0	19.000000
6	France	9.10	751	11.0	37.900002
7	Iceland	-0.80	743	211.0	11.200000
8	Ireland	0.70	1000	300.0	6.500000
9	Israel	0.60	-834	183.0	13.700000
10	Italy	27.90	775	107.0	42.200001
11	Japan	1.50	680	36.0	23.200001
12	Netherlands	1.80	773	167.0	9.200000
13	New Zealand	1.90	916	266.0	7.700000
14	Norway	0.08	806	227.0	12.200000
15	Spain	6.50	724	NaN	NaN
16	Sweden	1.60	743	207.0	11.200000
17	Switzerland	5.80	693	115.0	20.299999
18	UK	1.30	941	285.0	10.300000
19	US	1.20	926	199.0	22.100000
20	West Germany	2.70	861	172.0	36.700001
21	India	2.95	750	171.0	20.270000

In [94]:

```
df_alc.shape
```

Out[94]:

```
(22, 5)
```

In [95]:

```
df_alc.describe()
```

Out[95]:

	Alcohol	Deaths	Heart	Liver
count	22.000000	22.000000	20.000000	21.000000
mean	3.605909	750.590909	184.150000	20.265238
std	5.862724	366.636535	77.707464	11.428617
min	-0.800000	-834.000000	11.000000	6.500000
25%	1.225000	744.750000	158.000000	11.200000
50%	2.150000	789.000000	191.000000	19.000000
75%	2.937500	907.750000	221.750000	23.200001
max	27.900000	1000.000000	300.000000	45.599998

In [96]:

```
#datatpes of all the columns
df_alc.dtypes
```

Out[96]:

```
Country     object
Alcohol    float64
Deaths       int64
Heart      float64
Liver      float64
dtype: object
```

In [97]:

```
#sum of null values
df_alc.isnull().sum()
```

Out[97]:

```
Country    0
Alcohol    0
Deaths     0
Heart      2
Liver      1
dtype: int64
```

In [98]:

```
df_alc.dropna(subset='Heart',inplace=True)
```

In [99]:
```
df_alc
```

Out[99]:

	Country	Alcohol	Deaths	Heart	Liver
0	Australia	2.50	785	211.0	15.300000
1	Austria	3.00	863	167.0	45.599998
2	Belg. and Lux.	2.90	883	131.0	20.700001
4	Denmark	2.90	971	220.0	23.900000
5	Finland	0.80	970	297.0	19.000000
6	France	9.10	751	11.0	37.900002
7	Iceland	-0.80	743	211.0	11.200000
8	Ireland	0.70	1000	300.0	6.500000
9	Israel	0.60	-834	183.0	13.700000
10	Italy	27.90	775	107.0	42.200001
11	Japan	1.50	680	36.0	23.200001
12	Netherlands	1.80	773	167.0	9.200000
13	New Zealand	1.90	916	266.0	7.700000
14	Norway	0.08	806	227.0	12.200000
16	Sweden	1.60	743	207.0	11.200000
17	Switzerland	5.80	693	115.0	20.299999
18	UK	1.30	941	285.0	10.300000
19	US	1.20	926	199.0	22.100000
20	West Germany	2.70	861	172.0	36.700001
21	India	2.95	750	171.0	20.270000

In [101]:
```
# rounding of liver column values and storing in same column
df_alc['Liver']=df_alc['Liver'].round(2)
```

In [103]:
```
df_alc
```

Out[103]:

	Country	Alcohol	Deaths	Heart	Liver
0	Australia	2.50	785	211.0	15.30
1	Austria	3.00	863	167.0	45.60
2	Belg. and Lux.	2.90	883	131.0	20.70
4	Denmark	2.90	971	220.0	23.90
5	Finland	0.80	970	297.0	19.00
6	France	9.10	751	11.0	37.90
7	Iceland	-0.80	743	211.0	11.20
8	Ireland	0.70	1000	300.0	6.50
9	Israel	0.60	-834	183.0	13.70
10	Italy	27.90	775	107.0	42.20
11	Japan	1.50	680	36.0	23.20
12	Netherlands	1.80	773	167.0	9.20
13	New Zealand	1.90	916	266.0	7.70
14	Norway	0.08	806	227.0	12.20
16	Sweden	1.60	743	207.0	11.20
17	Switzerland	5.80	693	115.0	20.30
18	UK	1.30	941	285.0	10.30
19	US	1.20	926	199.0	22.10
20	West Germany	2.70	861	172.0	36.70
21	India	2.95	750	171.0	20.27

In [109]:
```
df_alc.columns
```

Out[109]:
```
Index(['Country', 'Alcohol', 'Deaths', 'Heart', 'Liver'], dtype='object')
```

In [110]:
```
#Remove leading and trailing characters spaces.
df_alc.columns=[c.strip() for c in df_alc.columns]
```

In [111]:
```
df_alc.columns
```

Out[111]:
```
Index(['Country', 'Alcohol', 'Deaths', 'Heart', 'Liver'], dtype='object')
```

In [113]:
```
x1=df_alc['Liver'].mean()
x1
```

Out[113]:
```
20.265238149428573
```

In [114]:
```
df_alc['Liver'].fillna(x1,inplace=True)
```

In [115]:
```
df_alc.isnull().sum()
```

Out[115]:
```
Country    0
Alcohol    0
Deaths     0
Heart      2
Liver      0
dtype: int64
```

In [116]:
```
x2=df_alc['Heart'].mean()
x2
```

Out[116]:
```
184.15
```

In [117]:
```
df_alc['Heart'].fillna(x2,inplace=True)
```

In [118]:
```
df_alc.isnull().sum()
```

Out[118]:
```
Country    0
Alcohol    0
Deaths     0
Heart      0
Liver      0
dtype: int64
```

In [128]:
```
df_alc["Alcohol"]=df_alc["Alcohol"].abs()   #getting absolute value
```

In [129]:
```
df_alc["Deaths"]=df_alc["Deaths"].abs()
```

In [122]:
```
df_alc
```

Out[122]:
```
          Country  Alcohol  Deaths   Heart      Liver
 0      Australia     2.50     785  211.00  15.300000
 1        Austria     3.00     863  167.00  45.599998
 2  Belg. and Lux.    2.90     883  131.00  20.700001
 3         Canada     2.40     793  184.15  16.400000
 4        Denmark     2.90     971  220.00  23.900000
 5        Finland     0.80     970  297.00  19.000000
 6         France     9.10     751   11.00  37.900002
 7        Iceland     0.80     743  211.00  11.200000
 8        Ireland     0.70    1000  300.00   6.500000
 9         Israel     0.60     834  183.00  13.700000
10          Italy    27.90     775  107.00  42.200001
11          Japan     1.50     680   36.00  23.200001
12    Netherlands     1.80     773  167.00   9.200000
13    New Zealand     1.90     916  266.00   7.700000
14         Norway     0.08     806  227.00  12.200000
15          Spain     6.50     724  184.15  20.265238
16         Sweden     1.60     743  207.00  11.200000
17    Switzerland     5.80     693  115.00  20.299999
18             UK     1.30     941  285.00  10.300000
19             US     1.20     926  199.00  22.100000
20   West Germany     2.70     861  172.00  36.700001
21          India     2.95     750  171.00  20.270000
```

In [123]:
```
#setting value at specified location
df_alc.loc[10,'Alcohol']=2.90
```

In [124]:

```
df_alc
```

Out[124]:

	Country	Alcohol	Deaths	Heart	Liver
0	Australia	2.50	785	211.00	15.300000
1	Austria	3.00	863	167.00	45.599998
2	Belg. and Lux.	2.90	883	131.00	20.700001
3	Canada	2.40	793	184.15	16.400000
4	Denmark	2.90	971	220.00	23.900000
5	Finland	0.80	970	297.00	19.000000
6	France	9.10	751	11.00	37.900002
7	Iceland	0.80	743	211.00	11.200000
8	Ireland	0.70	1000	300.00	6.500000
9	Israel	0.60	834	183.00	13.700000
10	Italy	2.90	775	107.00	42.200001
11	Japan	1.50	680	36.00	23.200001
12	Netherlands	1.80	773	167.00	9.200000
13	New Zealand	1.90	916	266.00	7.700000
14	Norway	0.08	806	227.00	12.200000
15	Spain	6.50	724	184.15	20.265238
16	Sweden	1.60	743	207.00	11.200000
17	Switzerland	5.80	693	115.00	20.299999
18	UK	1.30	941	285.00	10.300000
19	US	1.20	926	199.00	22.100000
20	West Germany	2.70	861	172.00	36.700001
21	India	2.95	750	171.00	20.270000