p2

March 28, 2024

[ ]:
```
import pandas as pd

df = pd.read_csv(r"dataSet.csv")
```

[ ]:
```
print("Printing the information of the dataset\n")
df.info()

print("\n\n Printing the head of the dataset\n")
print(df.head())

print("\n\n Printing the tail of the dataset\n")
print(df.tail())

print("\n\n Printing the count of the dataset\n")
print(df.count())

print("\n\n Printing the info of the dataset\n")
print(df.info())

print("\n\n Printing the isNull values of the dataset\n")
print(df.isnull())

print("\n\n Printing the null values sum of the dataset\n")
df.isnull().sum()
```

```
Printing the information of the dataset

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   roll    10 non-null     int64
 1   name    10 non-null     object
 2   class   9 non-null      object
 3   marks   8 non-null      float64
 4   age     9 non-null      float64
dtypes: float64(2), int64(1), object(2)
```

```
memory usage: 528.0+ bytes


 Printing the head of the dataset

   roll     name class  marks   age
0     1     anil    TE  56.77  22.0
1     2     amit    TE  59.77  21.0
2     3   aniket    BE  76.88  19.0
3     4  ajinkya    TE  69.66   NaN
4     5     asha   NaN  63.28  20.0


 Printing the tail of the dataset

   roll    name class  marks   age
5     6  ayesha    BE  49.55  20.0
6     7    amar    BE    NaN  19.0
7     8   amita    BE    NaN  23.0
8     9    amol    TE  56.75  20.0
9    10   anmol    BE  78.66  21.0


 Printing the count of the dataset

roll     10
name     10
class     9
marks     8
age       9
dtype: int64


 Printing the info of the dataset

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   roll    10 non-null     int64
 1   name    10 non-null     object
 2   class   9 non-null      object
 3   marks   8 non-null      float64
 4   age     9 non-null      float64
dtypes: float64(2), int64(1), object(2)
memory usage: 528.0+ bytes
None
```

```
 Printing the isNull values of the dataset

    roll   name  class  marks    age
0  False  False  False  False  False
1  False  False  False  False  False
2  False  False  False  False  False
3  False  False  False  False   True
4  False  False   True  False  False
5  False  False  False  False  False
6  False  False  False   True  False
7  False  False  False   True  False
8  False  False  False  False  False
9  False  False  False  False  False


 Printing the null values sum of the dataset
```

[ ]:
```
roll     0
name     0
class    1
marks    2
age      1
dtype: int64
```

[ ]:
```python
breakline = "\n\n\n\n"

#drop all rows that having no value
print("\n Drop all rows that having no value \n \n")
print(df.dropna())


#identify missing value using domain knowldege....fill absed n domain knowlwedge
print(breakline,"Identify missing value using domain knowldege....fill absed n␣
 ↪domain knowlwedge: \n \n");
print(df.fillna(0))

#only using class column
print(breakline,"Only using class column \n \n")
print(df['class'].fillna('TE'))


# filling the missing values with mean of marks
print(breakline,"Filling the missing values with mean of marks \n \n")
print(df['marks'].fillna(df['marks'].mean()))
```

```python
#find the count of distinct values
print(breakline,"Find the count of distinct values \n \n")
print(df['class'].value_counts())


#filling the missing values with median values of age
print(breakline,"Filling the missing values with median values of age \n \n");
print(df['age'].fillna(df['age'].median()))
```

```
 Drop all rows that having no value


   roll    name class  marks   age
0     1    anil    TE  56.77  22.0
1     2    amit    TE  59.77  21.0
2     3  aniket    BE  76.88  19.0
5     6  ayesha    BE  49.55  20.0
8     9    amol    TE  56.75  20.0
9    10   anmol    BE  78.66  21.0




 Identify missing value using domain knowldege…fill absed n domain
knowlwedge:


   roll     name class  marks   age
0     1     anil    TE  56.77  22.0
1     2     amit    TE  59.77  21.0
2     3   aniket    BE  76.88  19.0
3     4  ajinkya    TE  69.66   0.0
4     5     asha     0  63.28  20.0
5     6   ayesha    BE  49.55  20.0
6     7     amar    BE   0.00  19.0
7     8    amita    BE   0.00  23.0
8     9     amol    TE  56.75  20.0
9    10    anmol    BE  78.66  21.0



 Only using class column
```

```
0    TE
1    TE
2    BE
3    TE
4    TE
5    BE
6    BE
7    BE
8    TE
9    BE
Name: class, dtype: object



 Filling the missing values with mean of marks


0    56.770
1    59.770
2    76.880
3    69.660
4    63.280
5    49.550
6    63.915
7    63.915
8    56.750
9    78.660
Name: marks, dtype: float64



 Find the count of distinct values


BE    5
TE    4
Name: class, dtype: int64



 Filling the missing values with median values of age
```

```
0    22.0
1    21.0
2    19.0
3    20.0
4    20.0
5    20.0
6    19.0
7    23.0
8    20.0
9    21.0
Name: age, dtype: float64
```

[ ]:
```
# Propogates the values previous to the element with NaN field
print("Propogates the values previous to the element with NaN field \n \n")
print(df.fillna(method='pad'))

# Same as above just fills it with next value
print(breakline,"Same as above just fills it with next value \n \n")
print(df.fillna(method='backfill'))
```

```
Propogates the values previous to the element with NaN field


   roll     name class  marks   age
0     1     anil    TE  56.77  22.0
1     2     amit    TE  59.77  21.0
2     3   aniket    BE  76.88  19.0
3     4  ajinkya    TE  69.66  19.0
4     5     asha    TE  63.28  20.0
5     6   ayesha    BE  49.55  20.0
6     7     amar    BE  49.55  19.0
7     8    amita    BE  49.55  23.0
8     9     amol    TE  56.75  20.0
9    10   anmol    BE  78.66  21.0




 Same as above just fills it with next value


   roll     name class  marks   age
0     1     anil    TE  56.77  22.0
1     2     amit    TE  59.77  21.0
2     3   aniket    BE  76.88  19.0
3     4  ajinkya    TE  69.66  20.0
4     5     asha    BE  63.28  20.0
5     6   ayesha    BE  49.55  20.0
```

```
6     7     amar    BE   56.75   19.0
7     8    amita    BE   56.75   23.0
8     9     amol    TE   56.75   20.0
9    10    anmol    BE   78.66   21.0
```

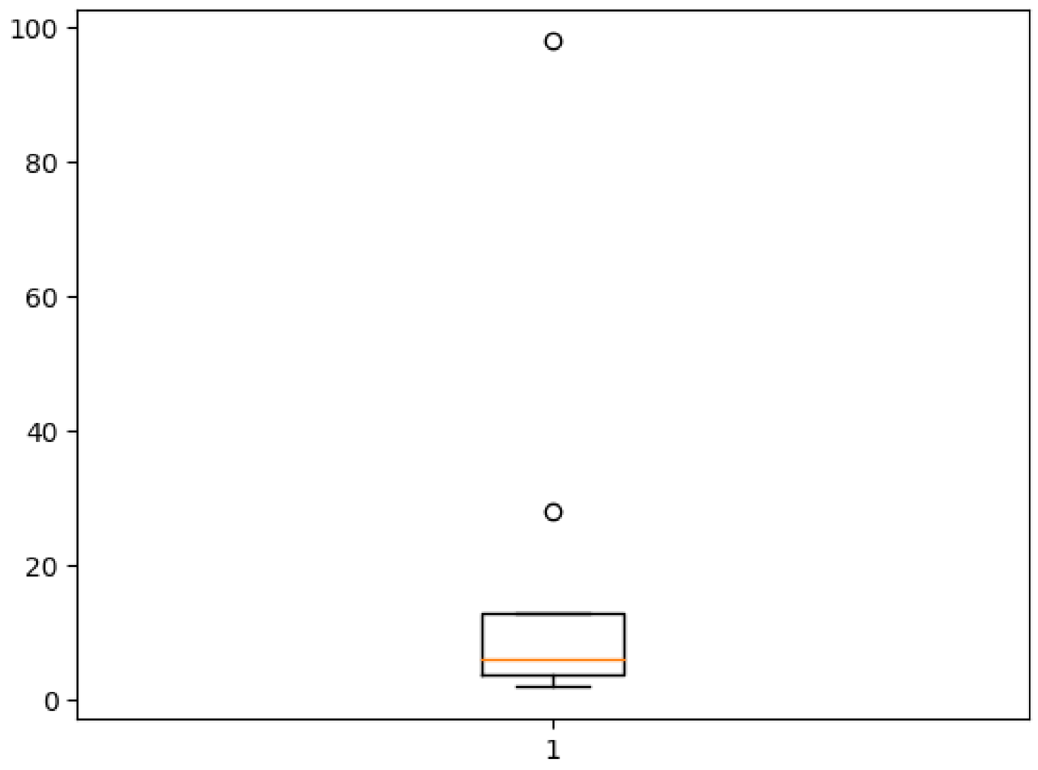
[ ]:
```
# Fills the values with the maximum freq value in that column for values with␣
  ↪NaN
print("Fills the values with the maximum freq value in that column for values␣
  ↪with NaN \n \n");
print(df['class'].fillna(df['class'].mode()[0]))
```

```
Fills the values with the maximum freq value in that column for values with NaN


0    TE
1    TE
2    BE
3    TE
4    BE
5    BE
6    BE
7    BE
8    TE
9    BE
Name: class, dtype: object
```

[ ]:
```
import numpy as np
x = np.array([5,4,3,2,7,8,98,28])
print("Mean is : ", np.mean(x))
print("Median is: ", np.median(x))
```

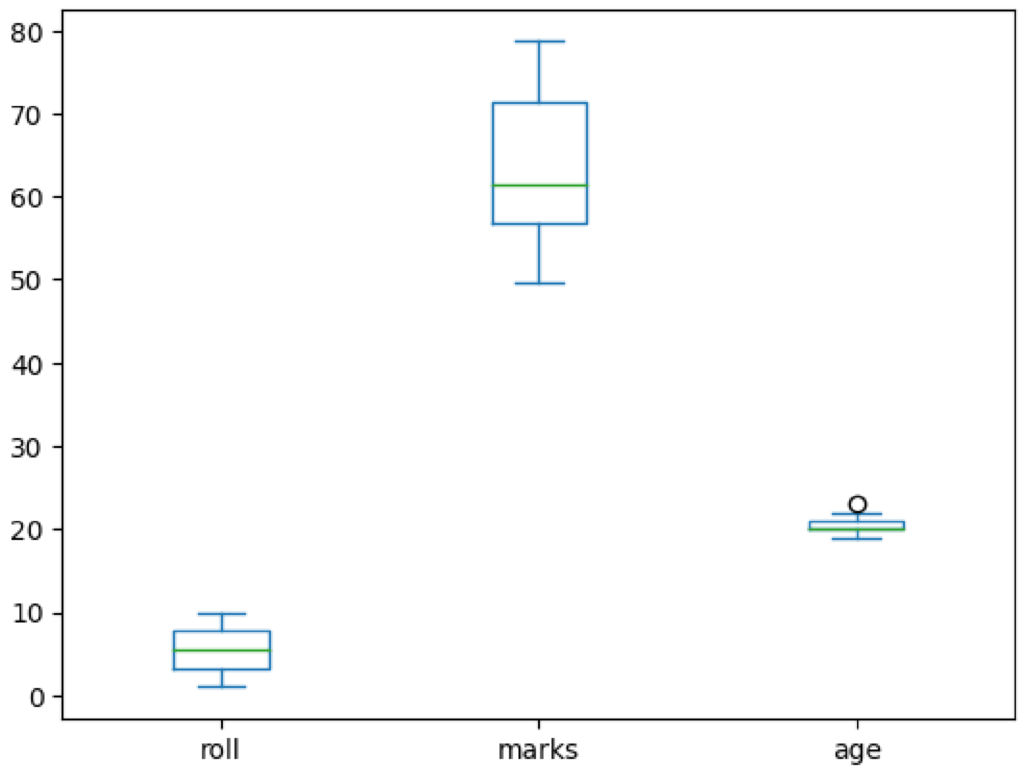
```
Mean is :  19.375
Median is:  6.0
```

[ ]:
```
import matplotlib.pyplot as plt
plt.boxplot(x);
```



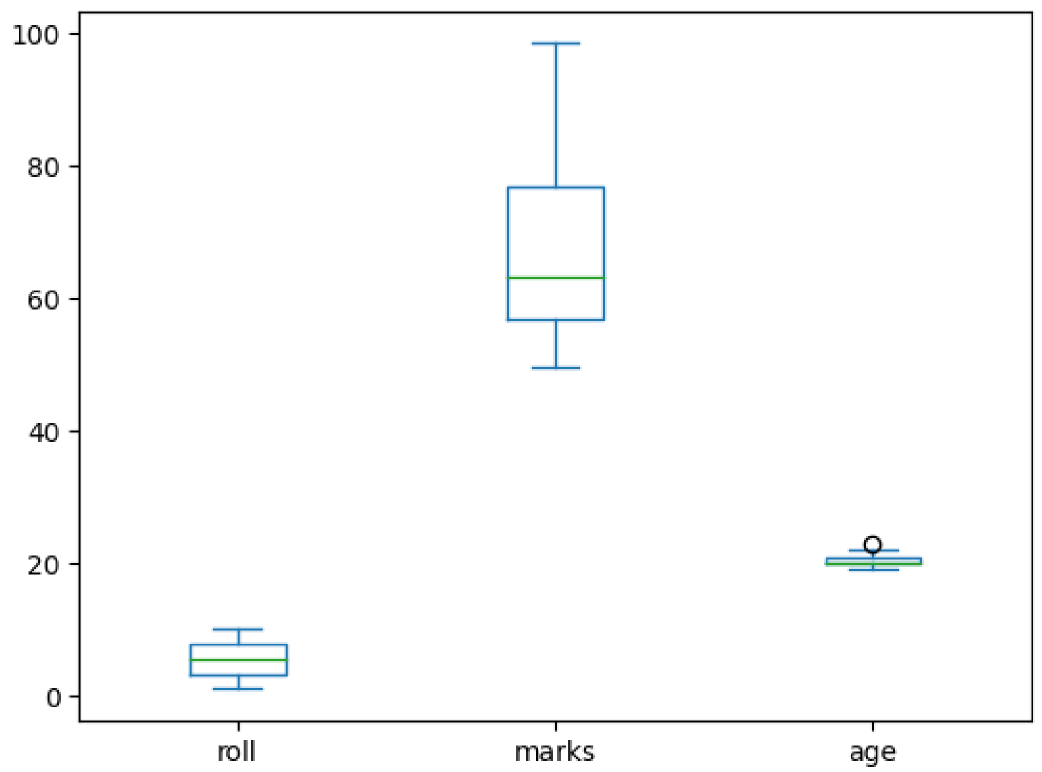
```
[ ]: df.plot.box()
```

```
[ ]: <Axes: >
```



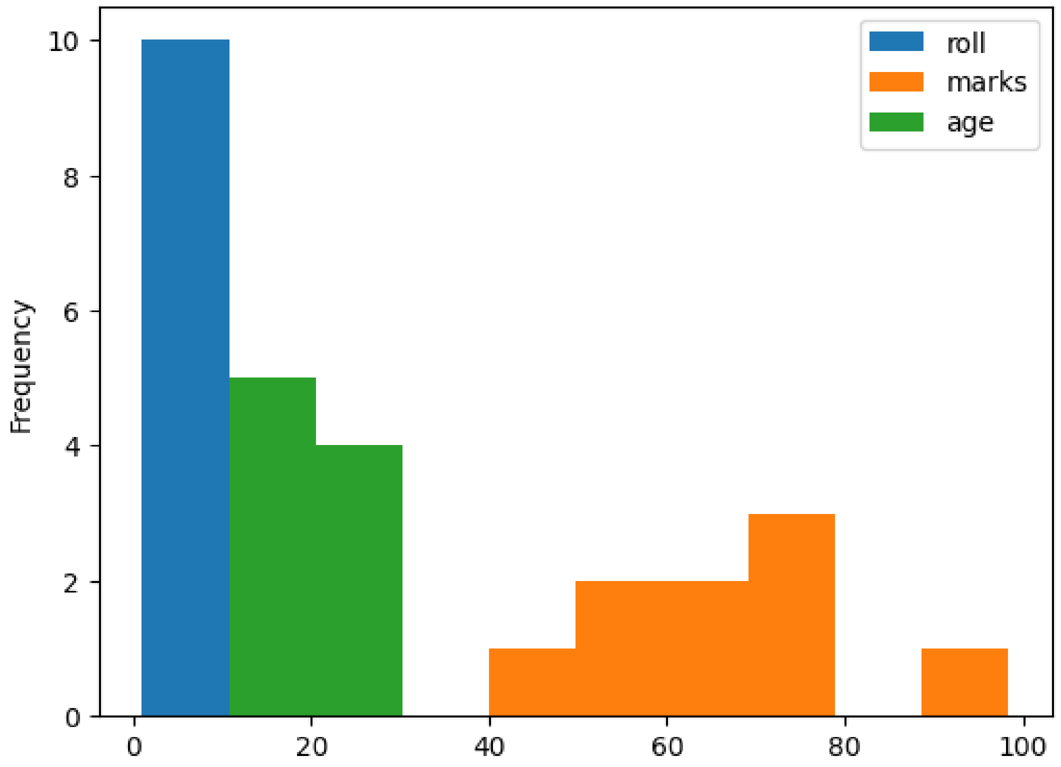
```
[ ]: df.loc[6,'marks']=98.45

     df.plot.box()
```

```
[ ]: <Axes: >
```



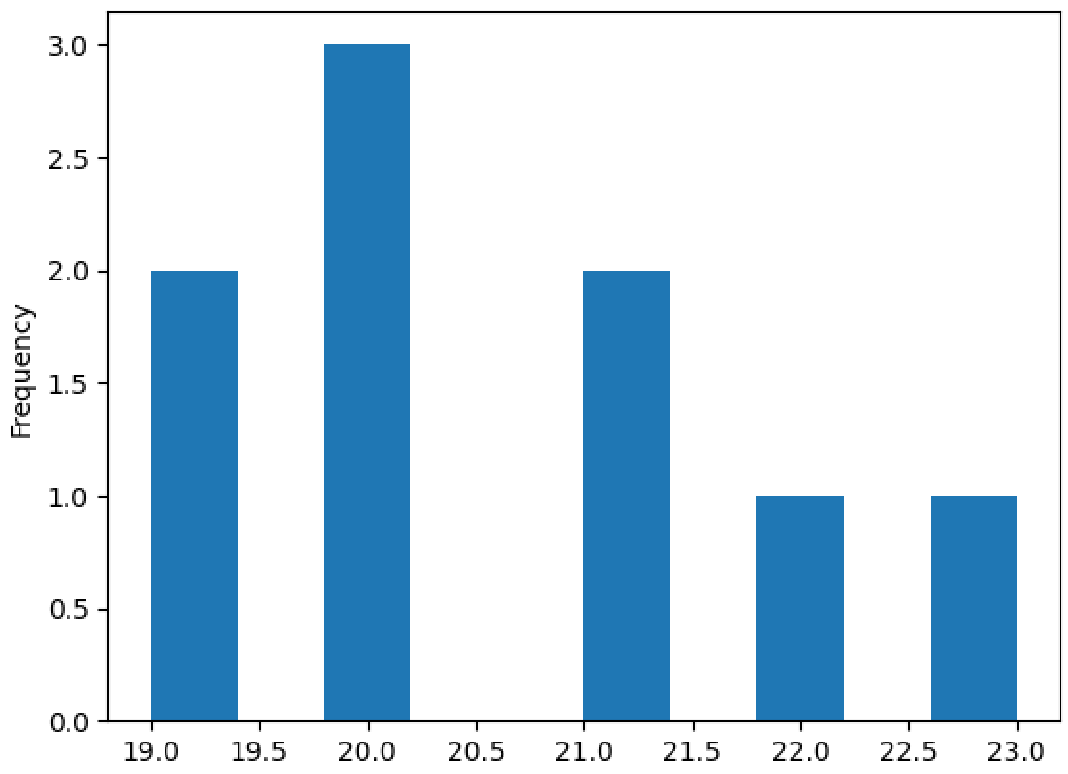
```
[ ]: df.plot.hist()
```

```
[ ]: <Axes: ylabel='Frequency'>
```



```
[ ]: df['age'].plot.hist()
```

```
[ ]: <Axes: ylabel='Frequency'>
```



[24]:
```
x= df[['age','marks']]
x.describe()
```

[24]:
```
             age      marks
count   9.000000   9.000000
mean   20.555556  67.752222
std     1.333333  15.020755
min    19.000000  49.550000
25%    20.000000  56.770000
50%    20.000000  63.280000
75%    21.000000  76.880000
max    23.000000  98.450000
```

[26]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
pd.DataFrame(x_scaled).describe()
```

[26]:
```
              0         1
count  9.000000  9.000000
mean   0.388889  0.372234
std    0.333333  0.307173
```

```
min    0.000000  0.000000
25%    0.250000  0.147648
50%    0.250000  0.280777
75%    0.500000  0.558896
max    1.000000  1.000000
```

[27]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
pd.DataFrame(x_scaled).describe()
```

[27]:
```
                  0             1
count  9.000000e+00  9.000000e+00
mean  -1.276756e-15  4.317534e-16
std    1.060660e+00  1.060660e+00
min   -1.237437e+00 -1.285313e+00
25%   -4.419417e-01 -7.754873e-01
50%   -4.419417e-01 -3.157969e-01
75%    3.535534e-01  6.445395e-01
max    1.944544e+00  2.167661e+00
```