

```
from scipy.stats import binom
from scipy.stats import bernoulli
from scipy.stats import poisson
import numpy as np
```

```
k,n,p=2,50,0.3
print(binom.pmf(k,n,p))
```

4.046546345956635e-06

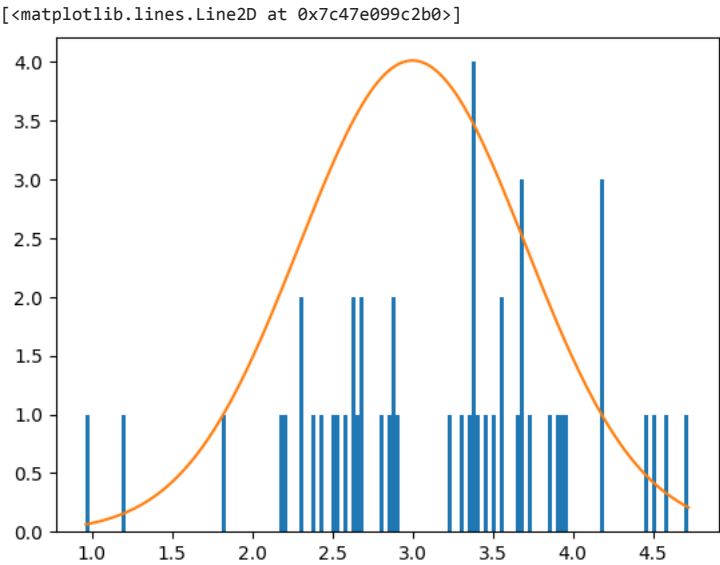
```
l=bernoulli(p)
print(l.pmf(k))
```

0.0

```
mu=3
print(poisson(mu).pmf(k))
```

0.22404180765538775

```
mu1,sigma=0.7,0.8
s=np.random.normal(mu,sigma,50)
count,bins,ignored=plt.hist(s,150)
plt.plot(bins,1/sigma*np.sqrt(2*np.pi)*np.exp(-(bins-mu)**2)*(2*sigma**2))
```



```
exp=np.random.exponential(2.0,200)
plt.hist(exp)
```



```
(array([81., 53., 27., 17., 9., 7., 1., 3., 1., 1.]),
array([8.81581120e-03, 1.16521308e+00, 2.32161034e+00, 3.47800761e+00,
4.63440487e+00, 5.79080214e+00, 6.94719941e+00, 8.10359667e+00,
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from scipy.stats import pearsonr
from scipy.stats import spearmanr
from numpy import cov
```

```
df=pd.read_csv("/content/2_2015 - 2_2015.csv")
df
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176
...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22628	0.67042
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1.63328
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179	0.32858
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1.83302
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16681	1.56726

158 rows × 12 columns



```
print("First 10 Rows:\n",df.head(10))
print("Last 7 Rows:\n",df.tail(7))
```

9 0.04083 1.33358 1.30923

	Country	Region	Happiness Rank	\
151	Burkina Faso	Sub-Saharan Africa	152	
152	Afghanistan	Southern Asia	153	
153	Rwanda	Sub-Saharan Africa	154	
154	Benin	Sub-Saharan Africa	155	
155	Syria	Middle East and Northern Africa	156	
156	Burundi	Sub-Saharan Africa	157	
157	Togo	Sub-Saharan Africa	158	

	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	\
151	3.587	0.04324	0.25812	0.85188	
152	3.575	0.03084	0.31982	0.30285	
153	3.465	0.03464	0.22208	0.77370	
154	3.340	0.03656	0.28665	0.35386	
155	3.006	0.05015	0.66320	0.47489	
156	2.905	0.08658	0.01530	0.41587	
157	2.839	0.06727	0.20868	0.13995	

	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	\
151	0.27125	0.39493	0.12832	
152	0.30335	0.23414	0.09719	
153	0.42864	0.59201	0.55191	
154	0.31910	0.48450	0.08010	
155	0.72193	0.15684	0.18906	
156	0.22396	0.11850	0.10062	
157	0.28443	0.36453	0.10731	

	Generosity	Dystopia Residual
151	0.21747	1.46494
152	0.36510	1.95210
153	0.22628	0.67042
154	0.18260	1.63328

df.isna().sum()

Country	0
Region	0
Happiness Rank	0
Happiness Score	0
Standard Error	0
Economy (GDP per Capita)	0
Family	0
Health (Life Expectancy)	0
Freedom	0
Trust (Government Corruption)	0
Generosity	0
Dystopia Residual	0
dtype: int64	

```
data=df[["Happiness Rank","Happiness Score"]]
print(data.sum())
print(data.median())
print("Mode:\n",df.mode().iloc[0])
```

Happiness Rank	12560.000
Happiness Score	849.366
dtype: float64	
Happiness Rank	79.5000
Happiness Score	5.2325
dtype: float64	
Mode:	
Country	Afghanistan
Region	Sub-Saharan Africa
Happiness Rank	82.0
Happiness Score	5.192
Standard Error	0.03751
Economy (GDP per Capita)	0.0
Family	0.0
Health (Life Expectancy)	0.92356
Freedom	0.0
Trust (Government Corruption)	0.32524
Generosity	0.0
Dystopia Residual	0.32858
Name: 0, dtype: object	

```
print("Shape:",df.shape)
print("Dimension:",df.ndim)
print("Size:",df.size)
print("Description:\n",df.describe())
```

Shape: (158, 12)
Dimension: 2
Size: 1896
Description:

	Happiness Rank	Happiness Score	Standard Error	\
count	158.000000	158.000000	158.000000	
mean	79.493671	5.375734	0.047885	
std	45.754363	1.145010	0.017146	
min	1.000000	2.839000	0.018480	

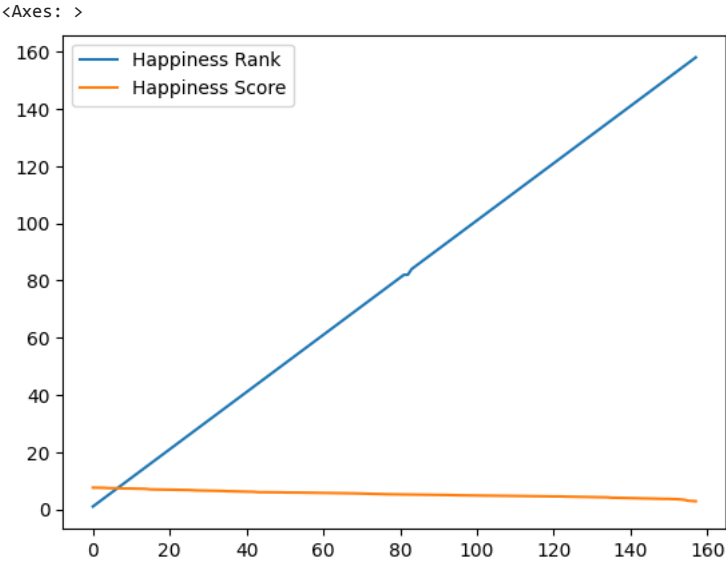
25%	40.250000	4.526000	0.037268
50%	79.500000	5.232500	0.043940
75%	118.750000	6.243750	0.052300
max	158.000000	7.587000	0.136930

	Economy (GDP per Capita)	Family	Health (Life Expectancy) \
count	158.000000	158.000000	158.000000
mean	0.846137	0.991046	0.630259
std	0.403121	0.272369	0.247078
min	0.000000	0.000000	0.000000
25%	0.545808	0.856823	0.439185
50%	0.910245	1.029510	0.696705
75%	1.158448	1.214405	0.811013
max	1.690420	1.402230	1.025250

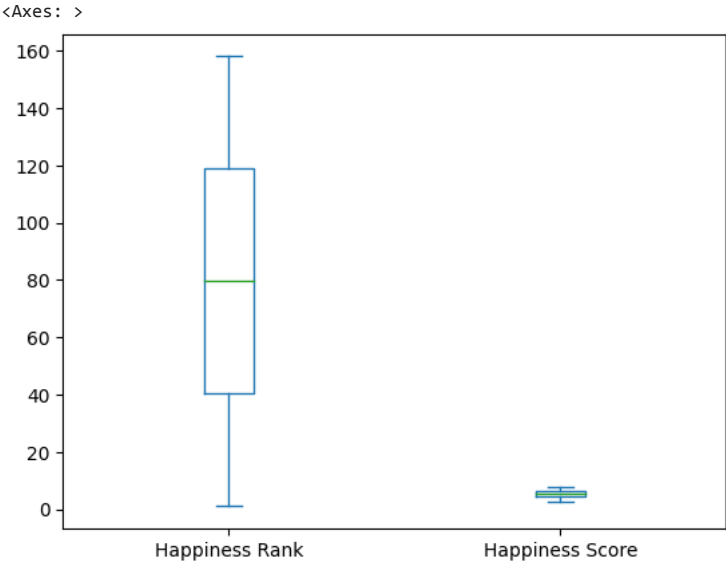
	Freedom	Trust (Government Corruption)	Generosity \
count	158.000000	158.000000	158.000000
mean	0.428615	0.143422	0.237296
std	0.150693	0.120034	0.126685
min	0.000000	0.000000	0.000000
25%	0.328330	0.061675	0.150553
50%	0.435515	0.107220	0.216130
75%	0.549092	0.180255	0.309883
max	0.669730	0.551910	0.795880

	Dystopia Residual
count	158.000000
mean	2.098977
std	0.553550
min	0.328580
25%	1.759410
50%	2.095415
75%	2.462415
max	3.602140

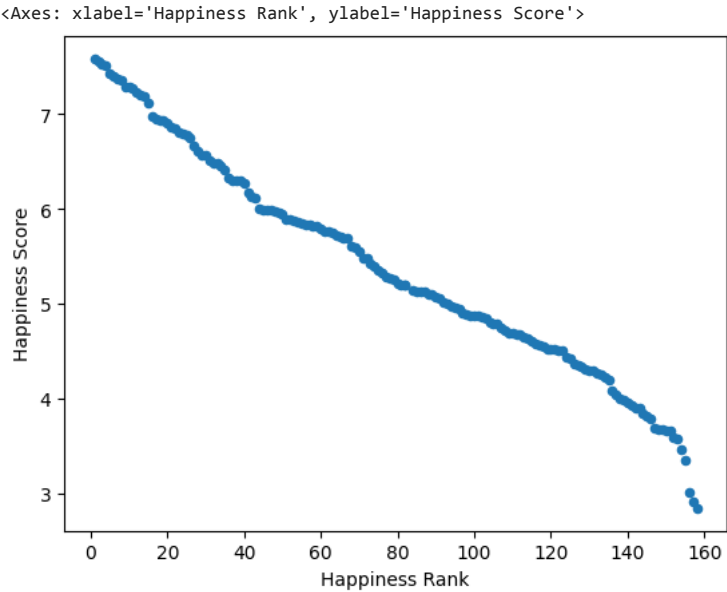
data.plot.line()



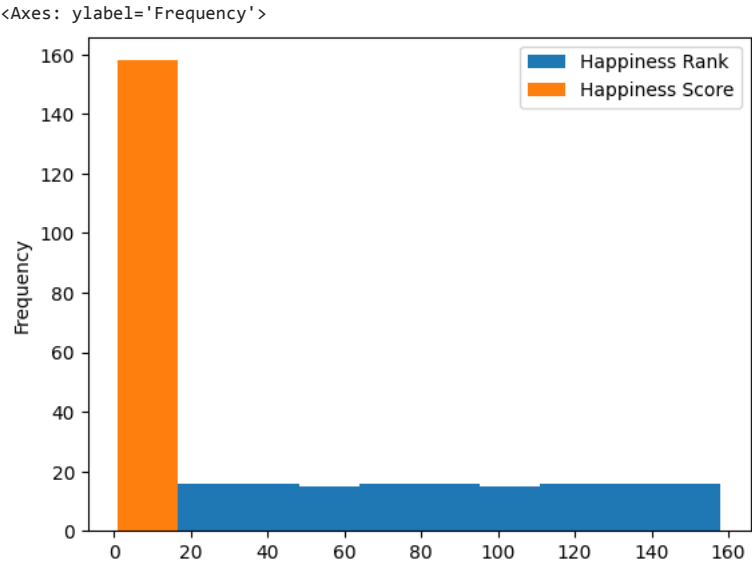
data.plot.box()



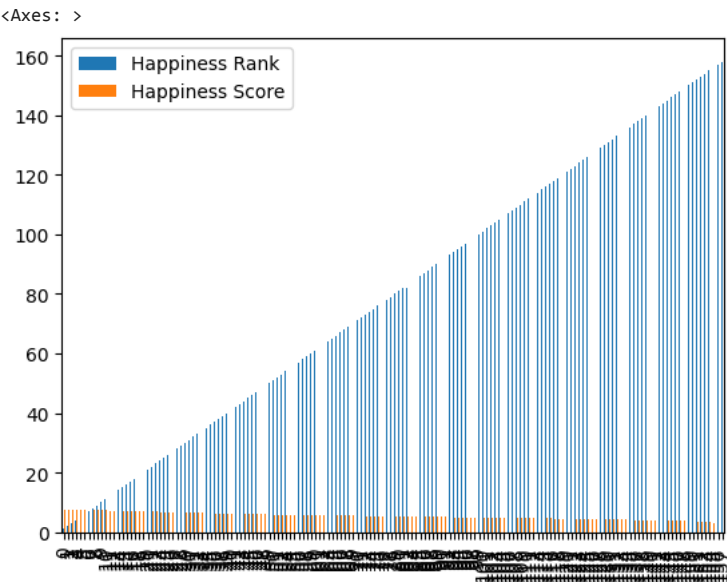
```
data.plot.scatter(x="Happiness Rank",y="Happiness Score")
```



```
data.plot.hist()
```

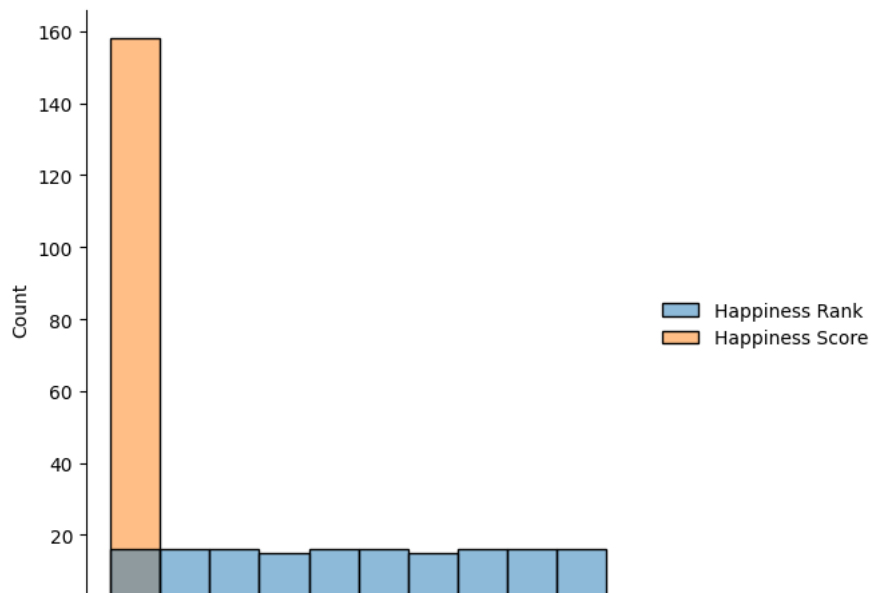


```
data.plot.bar()
```



```
sns.displot(data)
```

<seaborn.axisgrid.FacetGrid at 0x7c47f2294c10>



```
print(df.cov())
print("Pearson Correlation :\n",pearsonr(df["Happiness Rank"],df["Happiness Score"]))
print("Spearman Correlation :\n",spearmanr(df["Freedom"],df["Family"]))
```

	Happiness Rank	Happiness Score \
Happiness Rank	2093.461743	-51.975613
Happiness Score	-51.975613	1.311048
Standard Error	0.124358	-0.003480
Economy (GDP per Capita)	-14.483883	0.360476
Family	-9.142720	0.230969
Health (Life Expectancy)	-8.316021	0.204881
Freedom	-3.839647	0.098042
Trust (Government Corruption)	-2.044785	0.054316
Generosity	-0.928243	0.026156
Dystopia Residual	-13.220847	0.336225

	Standard Error	Economy (GDP per Capita) \
Happiness Rank	0.124358	-14.483883
Happiness Score	-0.003480	0.360476
Standard Error	0.000294	-0.001504
Economy (GDP per Capita)	-0.001504	0.162506
Family	-0.000564	0.070852
Health (Life Expectancy)	-0.001315	0.081323
Freedom	-0.000335	0.022495
Trust (Government Corruption)	-0.000367	0.014898
Generosity	-0.000192	-0.000534
Dystopia Residual	0.000797	0.008939

	Family	Health (Life Expectancy)	Freedom \
Happiness Rank	-9.142720	-8.316021	-3.839647
Happiness Score	0.230969	0.204881	0.098042
Standard Error	-0.000564	-0.001315	-0.000335
Economy (GDP per Capita)	0.070852	0.081323	0.022495
Family	0.074185	0.035741	0.018122
Health (Life Expectancy)	0.035741	0.061047	0.013422
Freedom	0.018122	0.013422	0.022708
Trust (Government Corruption)	0.006722	0.007365	0.008927
Generosity	0.003020	0.003391	0.007138
Dystopia Residual	0.022332	0.002596	0.005237

	Trust (Government Corruption)	Generosity \
Happiness Rank	-2.044785	-0.928243
Happiness Score	0.054316	0.026156
Standard Error	-0.000367	-0.000192
Economy (GDP per Capita)	0.014898	-0.000534
Family	0.006722	0.003020
Health (Life Expectancy)	0.007365	0.003391
Freedom	0.008927	0.007138
Trust (Government Corruption)	0.014408	0.004199
Generosity	0.004199	0.016049
Dystopia Residual	-0.002200	-0.007104

	Dystopia Residual
Happiness Rank	-13.220847
Happiness Score	0.336225
Standard Error	0.000797
Economy (GDP per Capita)	0.008939
Family	0.022332
Health (Life Expectancy)	0.002596
Freedom	0.005237
Trust (Government Corruption)	-0.002200

```
df2=pd.read_csv("/content/1_fiat500_VehicleSelection_Dataset - 1_fiat500_VehicleSelection_Dataset.csv")
df2
```

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Unnamed: 9	Unnamed: 10		
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900	NaN	NaN		
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800	NaN	NaN		
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200	NaN	NaN		
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000	NaN	NaN		
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700	NaN	NaN		
...		
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	5	NaN	NaN		
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonprice	NaN	NaN		
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	NO	NaN	NaN		
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	1	NaN	NaN		
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	1	NaN	NaN		

1549 rows x 11 columns

```
df2.isna().sum()
```

```
ID          11
model       11
engine_power 11
age_in_days 11
km          11
previous_owners 11
lat         11
lon         0
price       0
Unnamed: 9  15499
Unnamed: 10 1548
dtype: int64
```

```
df2=df2.drop(df2.index[1537:1549],axis=0)
df2=df2.drop(["Unnamed: 9","Unnamed: 10"],axis=1)
df2
```

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700
...
1532	1533.0	pop	51.0	1917.0	52008.0	1.0	45.548000	11.54946995	9900
1533	1534.0	sport	51.0	3712.0	115280.0	1.0	45.069679	7.704919815	5200
1534	1535.0	lounge	74.0	3835.0	112000.0	1.0	45.845692	8.666870117	4600
1535	1536.0	pop	51.0	2223.0	60457.0	1.0	45.481541	9.413479805	7500
1536	1537.0	lounge	51.0	2557.0	80750.0	1.0	45.000702	7.68227005	5990

1537 rows x 9 columns

```
print("First 10 Rows:\n",df2.head(10))
print("Last 7 Rows:\n",df2.tail(7))
```

```

First 10 Rows:
  ID  model  engine_power  age_in_days  km  previous_owners
0  1.0  lounge      51.0         882.0    25000.0      1.0
1  2.0  pop        51.0        1186.0   32500.0      1.0
2  3.0  sport       74.0        4658.0  142228.0      1.0
3  4.0  lounge      51.0        2739.0  160000.0      1.0
4  5.0  pop         73.0        3074.0  106880.0      1.0
5  6.0  pop         74.0        3623.0   70225.0      1.0
6  7.0  lounge      51.0         731.0   11600.0      1.0
7  8.0  lounge      51.0        1521.0   49076.0      1.0
8  9.0  sport       73.0        4049.0   76000.0      1.0
9 10.0  sport       51.0        3653.0   89000.0      1.0

      lat      lon  price
0  44.907242  8.611559  8868

```

1	45.666359	12.24188995	8800
2	45.503300	11.41784	4200
3	40.633171	17.63460922	6000
4	41.903221	12.49565029	5700
5	45.000702	7.68227005	7900
6	44.907242	8.611559868	10750
7	41.903221	12.49565029	9190
8	45.548000	11.54946995	5600
9	45.438301	10.99170017	6000

Last 7 Rows:

	ID	model	engine_power	age_in_days	km	previous_owners	\
1530	1531.0	lounge	51.0	670.0	29000.0	1.0	
1531	1532.0	sport	73.0	4505.0	127000.0	1.0	
1532	1533.0	pop	51.0	1917.0	52008.0	1.0	
1533	1534.0	sport	51.0	3712.0	115280.0	1.0	
1534	1535.0	lounge	74.0	3835.0	112000.0	1.0	
1535	1536.0	pop	51.0	2223.0	60457.0	1.0	
1536	1537.0	lounge	51.0	2557.0	80750.0	1.0	

	lat	lon	price
1530	45.764648	8.99450016	10800
1531	45.528511	9.593230247	4750
1532	45.548000	11.54946995	9900
1533	45.069679	7.704919815	5200
1534	45.845692	8.666870117	4600
1535	45.481541	9.413479805	7500
1536	45.000702	7.68227005	5990

```
data3=df2[["age_in_days","km"]]
print(data3.sum())
print(data3.median())
print("Mode:\n",df2.mode().iloc[0])
```

```
age_in_days    2537442.0
km             82068790.0
dtype: float64
age_in_days    1035.0
km             39024.0
dtype: float64
Mode:
ID              1.0
model           lounge
engine_power    51.0
age_in_days     366.0
km             17000.0
previous_owners 1.0
lat            41.903221
lon            12.49565029
price          10500
Name: 0, dtype: object
```

```
print("Shape:",df2.shape)
print("Dimension:",df2.ndim)
print("Size:",df2.size)
print("Description:\n",df2.describe())
```

```
Shape: (1537, 9)
Dimension: 2
Size: 13833
Description:
```

	ID	engine_power	age_in_days	km	previous_owners	\
count	1537.000000	1537.000000	1537.000000	1537.000000	1537.000000	
mean	769.000000	51.905010	1650.905660	53395.439167	1.123617	
std	443.837996	3.989254	1289.938635	40059.858383	0.416546	
min	1.000000	51.000000	366.000000	1232.000000	1.000000	
25%	385.000000	51.000000	670.000000	20000.000000	1.000000	
50%	769.000000	51.000000	1035.000000	39024.000000	1.000000	
75%	1153.000000	51.000000	2616.000000	79800.000000	1.000000	
max	1537.000000	77.000000	4658.000000	235000.000000	4.000000	

	lat
count	1537.000000
mean	43.543455
std	2.132631
min	36.855839
25%	41.802990
50%	44.399971
75%	45.467960
max	46.795612

```
sns.displot(data3)
```

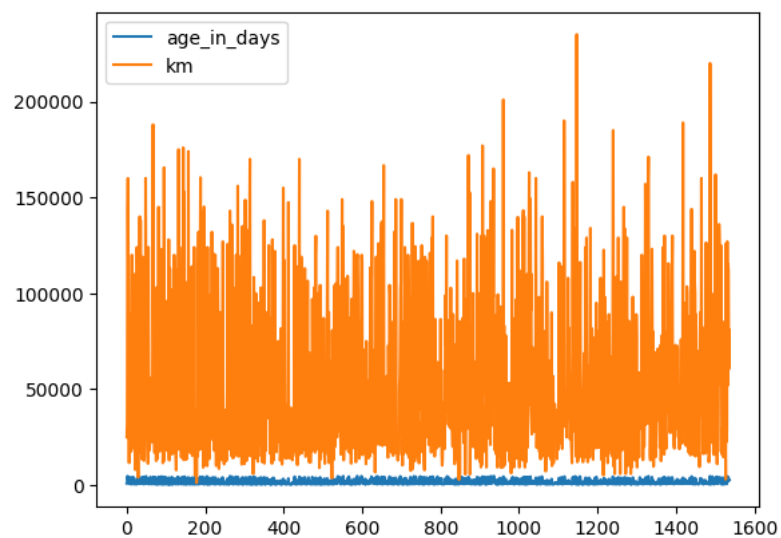


```
<seaborn.axisgrid.FacetGrid at 0x7c47e71d2020>
```



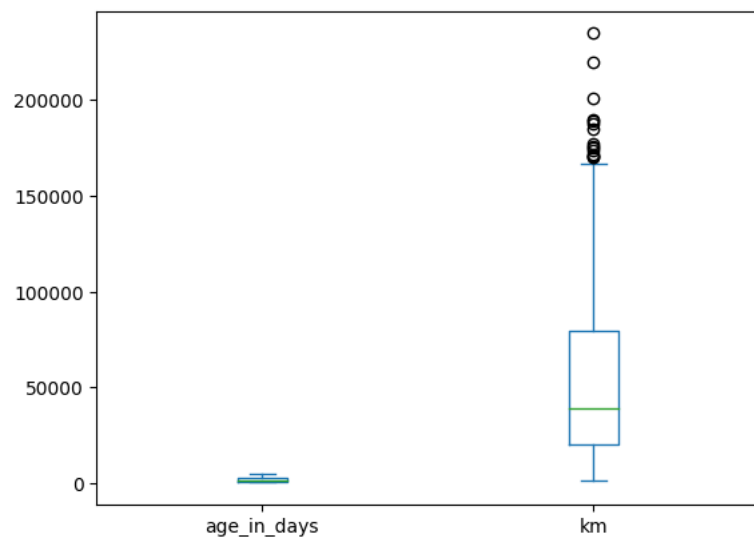
```
data3.plot.line()
```

<Axes: >



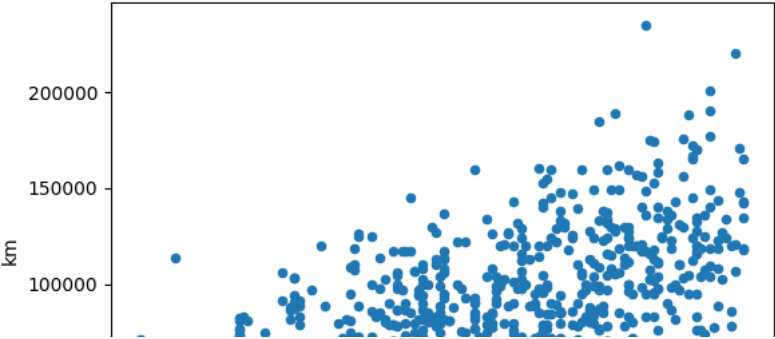
```
data3.plot.box()
```

<Axes: >



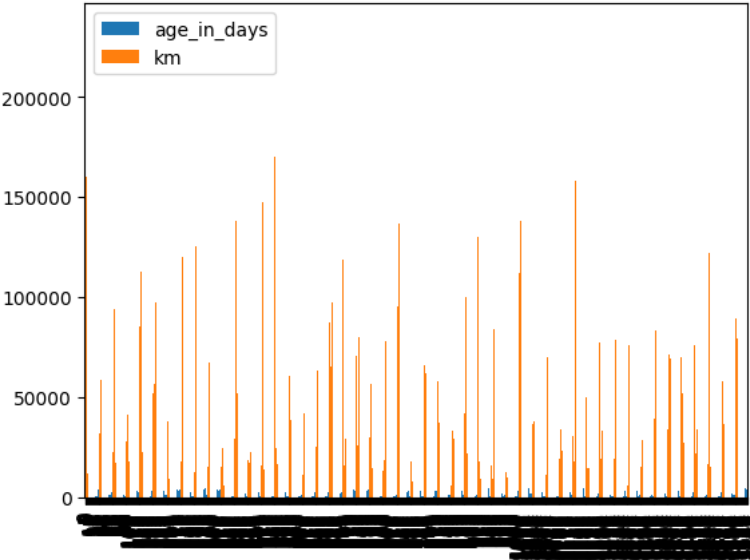
```
data3.plot.scatter("age_in_days", "km")
```

<Axes: xlabel='age_in_days', ylabel='km'>



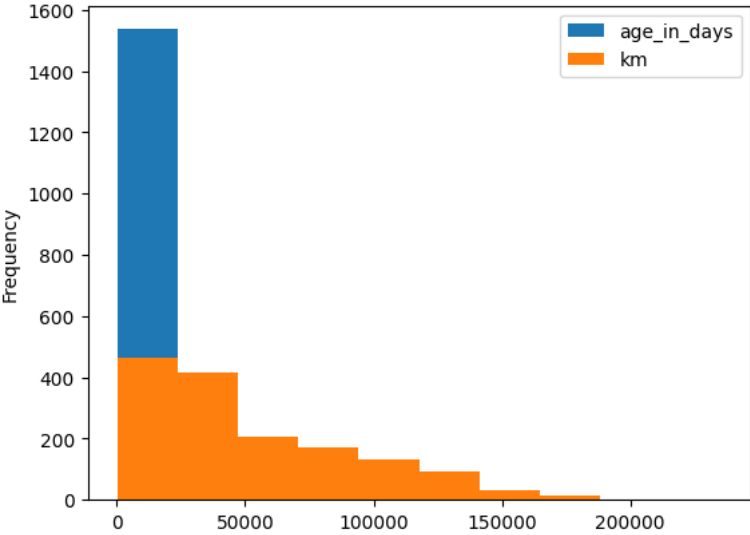
data3.plot.bar()

<Axes: >



data3.plot.hist()

<Axes: ylabel='Frequency'>



```
print(df2.cov())
print("Pearson Correlation :\n",pearsonr(df2["engine_power"],df2["km"]))
print("Spearman Correlation :\n",spearmanr(df2["engine_power"],df2["km"]))
```

	ID	engine_power	age_in_days	km	\
ID	196992.166667	-59.912109	-3.487395e+04	-1.167776e+05	
engine_power	-59.912109	15.914148	1.642617e+03	4.562600e+04	
age_in_days	-34873.952474	1642.617335	1.663942e+06	4.309110e+07	
km	-116777.595703	45626.001379	4.309110e+07	1.604792e+09	
previous_owners	1.505859	-0.008432	4.072586e+01	1.627724e+03	
lat	-53.578661	0.046811	1.736320e+02	3.038624e+03	
	previous_owners	lat			
ID	1.505859	-53.578661			
engine_power	-0.008432	0.046811			
age_in_days	40.725862	173.631987			
km	1627.723671	3038.623575			

previous_owners 0.173511 0.001250
lat 0.001250 4.548116

Pearson Correlation :
PearsonRResult(statistic=0.2855034117553602, pvalue=3.227790125120896e-30)
Spearman Correlation :
SignificanceResult(statistic=0.23693784711375426, pvalue=4.691193244222526e-21)

<ipython-input-192-f44d629ea5d1>:1: FutureWarning: The default value of numeric_only in DataFrame.cov is deprecated. In a future version, it will
print(df2.cov())

```
df3=pd.read_csv("/content/3_Fitness-1 - 3_Fitness-1.csv")  
df3
```

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

```
print("First 10 Rows:\n",df3.head(10))  
print("Last 7 Rows:\n",df3.tail(7))
```

First 10 Rows:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

Last 7 Rows:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

```
da=df3[["Sum of Mar","Sum of Total Sales"]]  
da
```

	Sum of Mar	Sum of Total Sales
0	6.16%	75
1	19.21%	160
2	5.17%	101
3	7.88%	127
4	11.82%	179
5	18.47%	167
6	17.49%	171
7	13.79%	170
8	100.00%	1150

```
print(da.sum())  
print(da.median())  
print("Mode:\n",df3.mode().iloc[0])
```

Sum of Mar 6.16%19.21%5.17%7.88%11.82%18.47%17.49%13.79%1...
Sum of Total Sales 2300
dtype: object
Sum of Total Sales 167.0
dtype: float64
Mode:
Row Labels A

```
Sum of Jan      100.00%
Sum of Feb      10.57%
Sum of Mar      100.00%
Sum of Total Sales    75
```

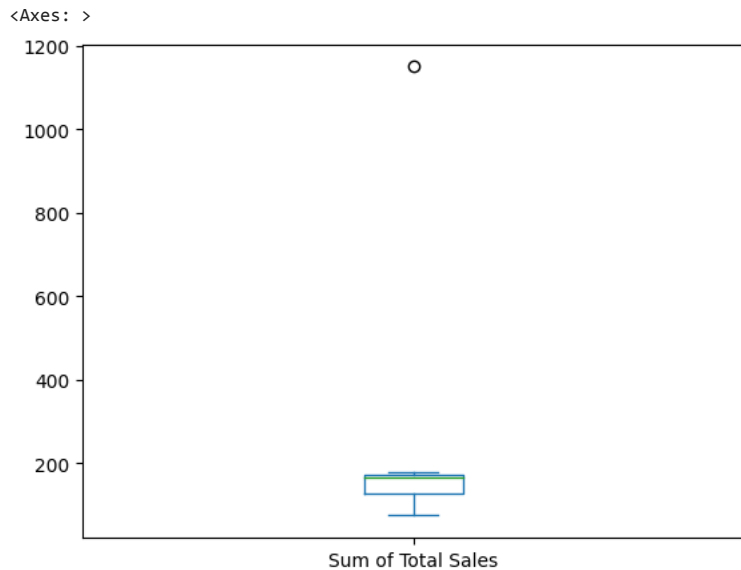
```
Name: 0, dtype: object
```

```
<ipython-input-182-bed56b5da75d>:2: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future version, it w
print(da.median())
```

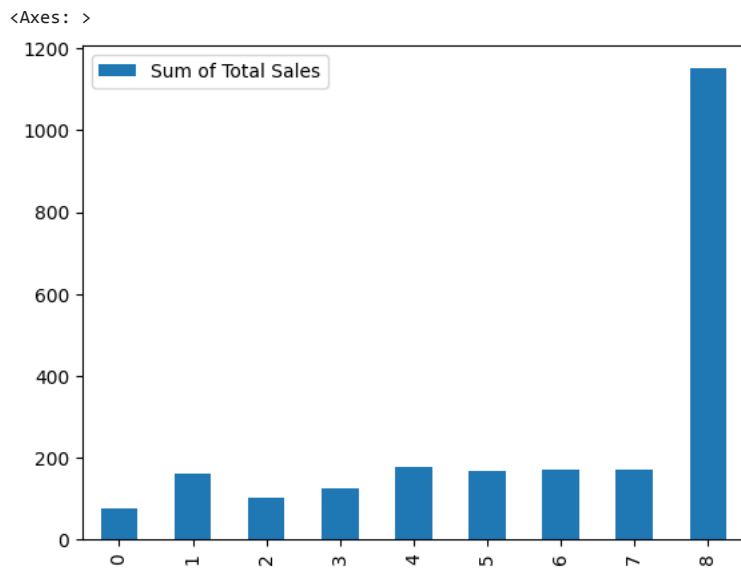
```
print("Shape:",df3.shape)
print("Dimension:",df3.ndim)
print("Size:",df3.size)
print("Description:\n",df3.describe())
```

```
Shape: (9, 5)
Dimension: 2
Size: 45
Description:
      Sum of Total Sales
count      9.000000
mean      255.555556
std       337.332963
min        75.000000
25%       127.000000
50%       167.000000
75%       171.000000
max       1150.000000
```

```
da.plot.box()
```

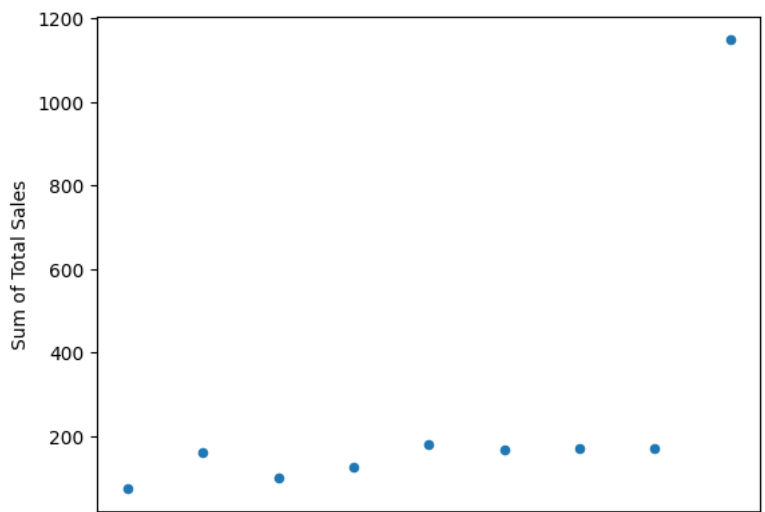


```
da.plot.bar()
```



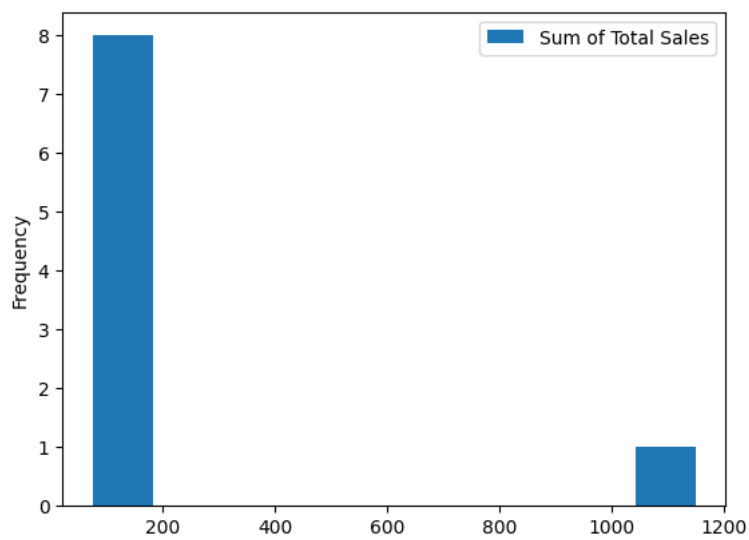
```
da.plot.scatter("Sum of Mar","Sum of Total Sales")
```

<Axes: xlabel='Sum of Mar', ylabel='Sum of Total Sales'>



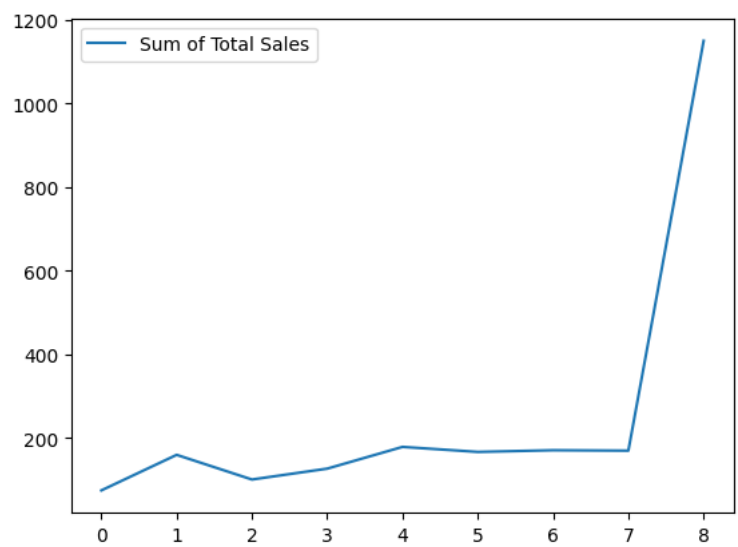
```
da.plot.hist()
```

<Axes: ylabel='Frequency'>



```
da.plot.line()
```

<Axes: >



```
df3["Sum of Mar"]=df3["Sum of Mar"].str.rstrip("%")
df3["Sum of Mar"]=df3["Sum of Mar"].astype(float)
```

```
print(df3.cov())
print("Pearson Correlation :\n",pearsonr(df3["Sum of Mar"],df3["Sum of Total Sales"]))
print("Spearman Correlation :\n",spearmanr(df3["Sum of Mar"],df3["Sum of Total Sales"]))
```

	Sum of Mar	Sum of Total Sales
Sum of Mar	878.588811	9935.666806

```
Sum of Total Sales  9935.666806      113793.527778
Pearson Correlation :
PearsonRResult(statistic=0.9936773809789188, pvalue=6.576638324757487e-08)
Spearman Correlation :
SignificanceResult(statistic=0.6666666666666667, pvalue=0.04986723056888511)
<ipython-input-191-b19954d893e0>:1: FutureWarning: The default value of numeric_only in DataFrame.cov is deprecated. In a future version, it will
print(df3.cov())
```

