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

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

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12
...	...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
df1=df.drop(df.index[1537:],axis=0)
df1=df1.drop(["Unnamed: 9","Unnamed: 10","model"],axis=1)
df1
```

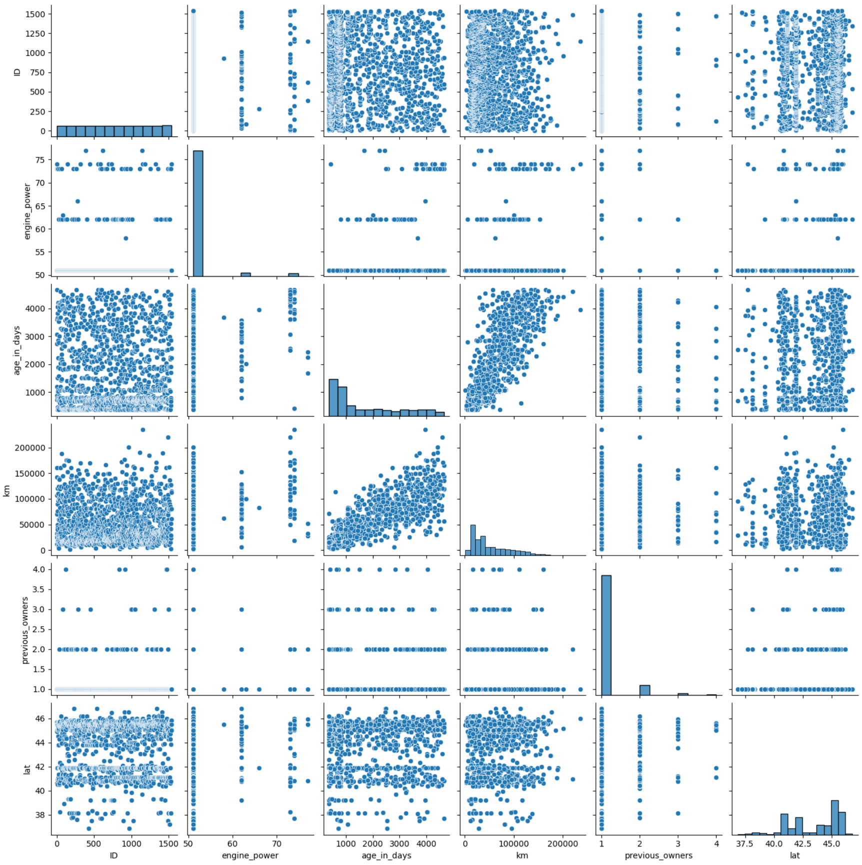
	ID	engine_power	age_in_days	km	previous_owners	l
0	1.0	51.0	882.0	25000.0	1.0	44.9072
1	2.0	51.0	1186.0	32500.0	1.0	45.6663
2	3.0	74.0	4658.0	142228.0	1.0	45.5033
3	4.0	51.0	2739.0	160000.0	1.0	40.6331
4	5.0	73.0	3074.0	106880.0	1.0	41.9032
...	...	...	...	...	...	
1532	1533.0	51.0	1917.0	52008.0	1.0	45.5480
1533	1534.0	51.0	3712.0	115280.0	1.0	45.0696
1534	1535.0	74.0	3835.0	112000.0	1.0	45.8456
1535	1536.0	51.0	2223.0	60457.0	1.0	45.4815

```
df1.describe()
```

	ID	engine_power	age_in_days	km	previous_own
count	1537.000000	1537.000000	1537.000000	1537.000000	1537.000(
mean	769.000000	51.905010	1650.905660	53395.439167	1.123(
std	443.837996	3.989254	1289.938635	40059.858383	0.416!
min	1.000000	51.000000	366.000000	1232.000000	1.000(
25%	385.000000	51.000000	670.000000	20000.000000	1.000(
50%	769.000000	51.000000	1035.000000	39024.000000	1.000(
75%	1153.000000	51.000000	2616.000000	79800.000000	1.000(

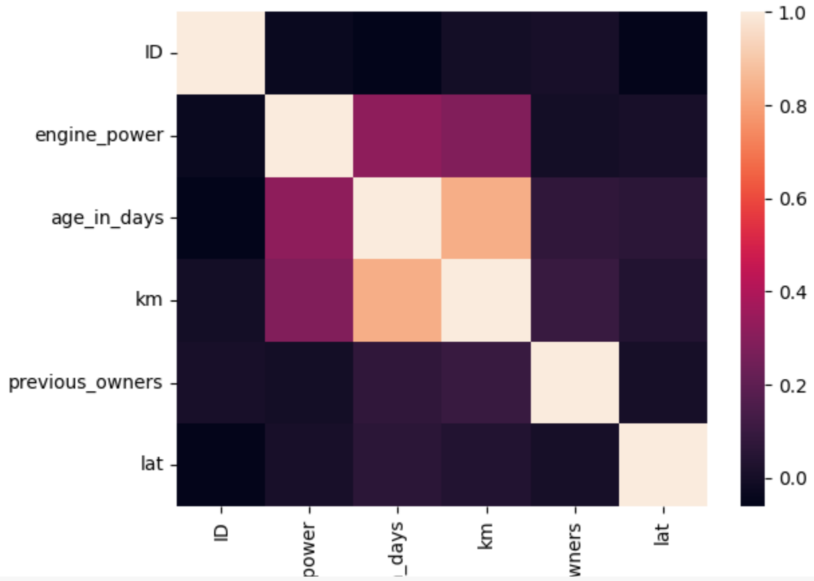
```
sns.pairplot(df1)
```

```
<seaborn.axisgrid.PairGrid at 0x7daebdc06980>
```



```
sns.heatmap(df1.corr())
```

```
<ipython-input-6-3ed1a1a51dc0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  sns.heatmap(df1.corr())
<Axes: >
```



```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
y=df1['age_in_days']
x=df1.drop(['age_in_days'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

```
2832.7802274686583
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
ID	-0.112496
engine_power	19.996885
km	0.009130
previous_owners	-13.112851
lat	24.000397
lon	-5.359399
price	-0.418384

```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7daeb728dc60>
```

```
model.score(x_test,y_test)
```

```
0.8120188273167547
```

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

	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

```
daf1=daf.drop(["Country","Region"],axis=1)
daf1.isna().sum()
```

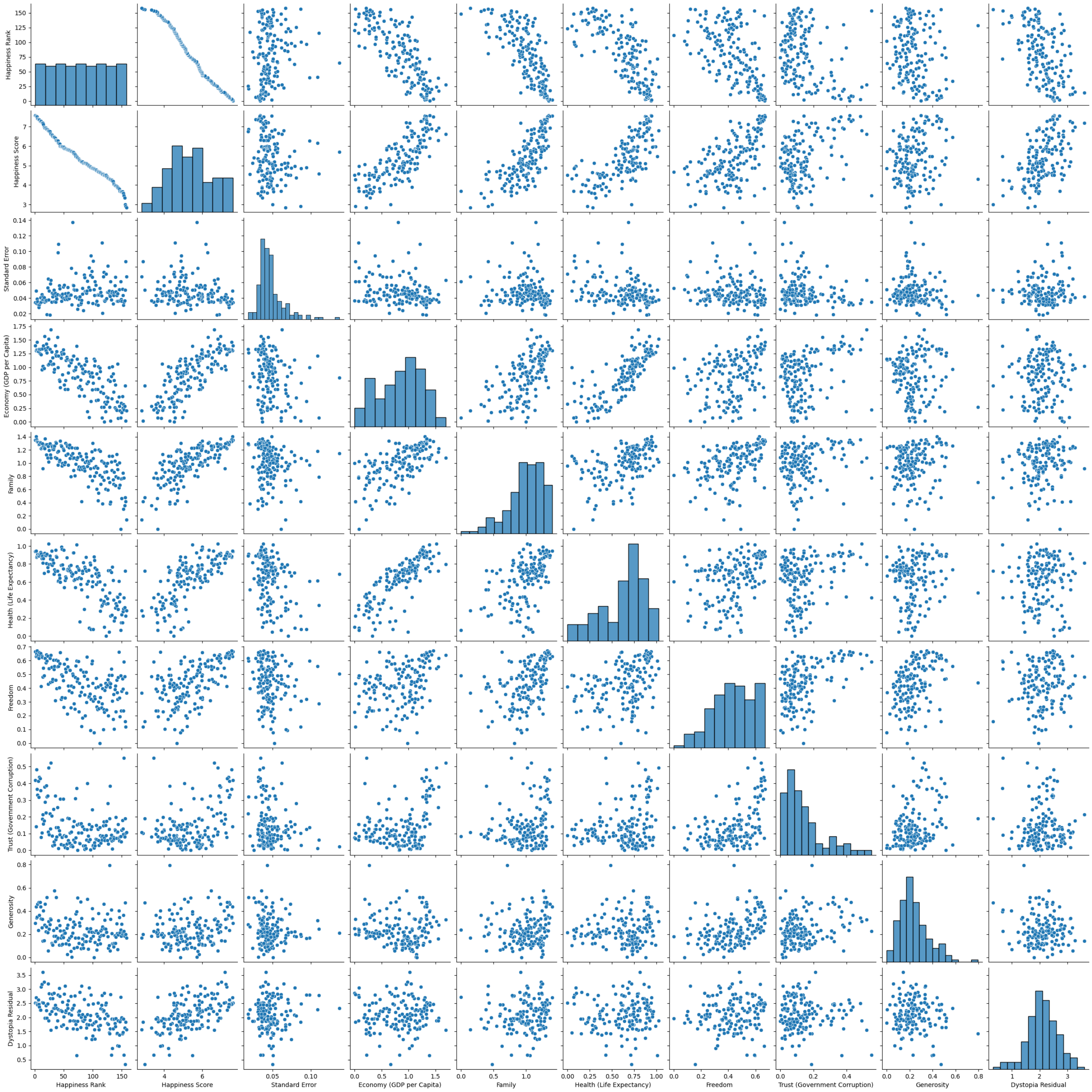
```
Happiness Rank                   0
Happiness Score                  0
Standard Error                   0
Economy (GDP per Capita)         0
Family                           0
Health (Life Expectancy)         0
Freedom                          0
Trust (Government Corruption)    0
Generosity      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

```
daf1.describe()
```

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.237296	2.098977
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.126685	0.553550
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.328580
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.150553	1.759410
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.216130	2.095415
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.309883	2.462415
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.795880	3.602140

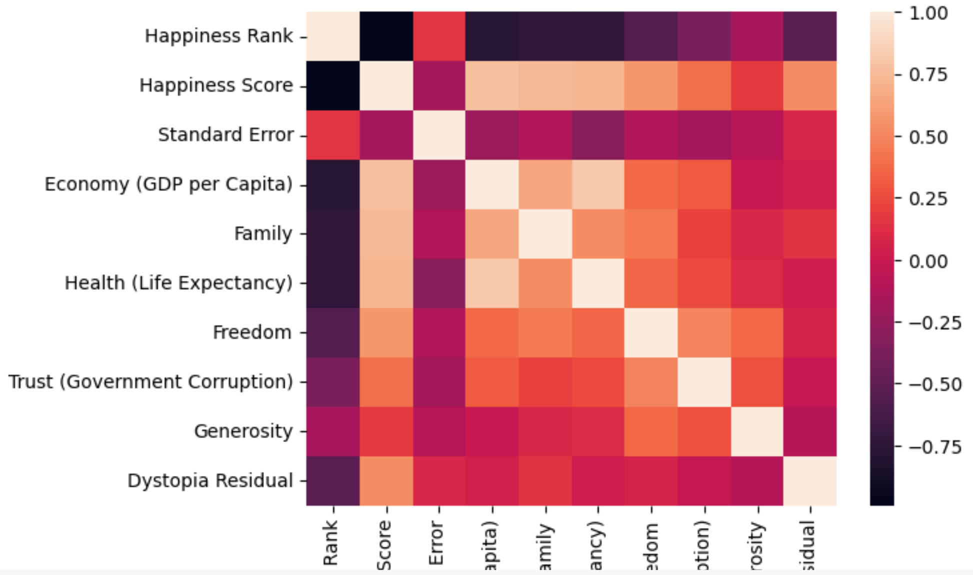
```
sns.pairplot(daf1)
```

```
<seaborn.axisgrid.PairGrid at 0x7daeb4ea8e80>
```



```
sns.heatmap(daf1.corr())
```

```
<Axes: >
```



```
y=daf1['Standard Error']
x=daf1.drop(['Standard Error'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()

model.fit(x_train,y_train)

model.intercept_
```

```
2832.7802274686583
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

	Coefficient
ID	-0.112496
engine_power	19.996885
km	0.009130
previous_owners	-13.112851
lat	24.000397
lon	-5.359399
price	-0.418384

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7daeaedadb70>
```

```
model.score(x_test,y_test)
```

```
0.8120188273167547
```

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

	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

```
df=df.drop(['Row Labels'],axis=1)
```

```
df["Sum of Jan"]=df["Sum of Jan"].replace("%","",regex=True).astype(float)
df["Sum of Feb"]=df["Sum of Feb"].replace("%","",regex=True).astype(float)
df["Sum of Mar"]=df["Sum of Mar"].replace("%","",regex=True).astype(float)
df
```

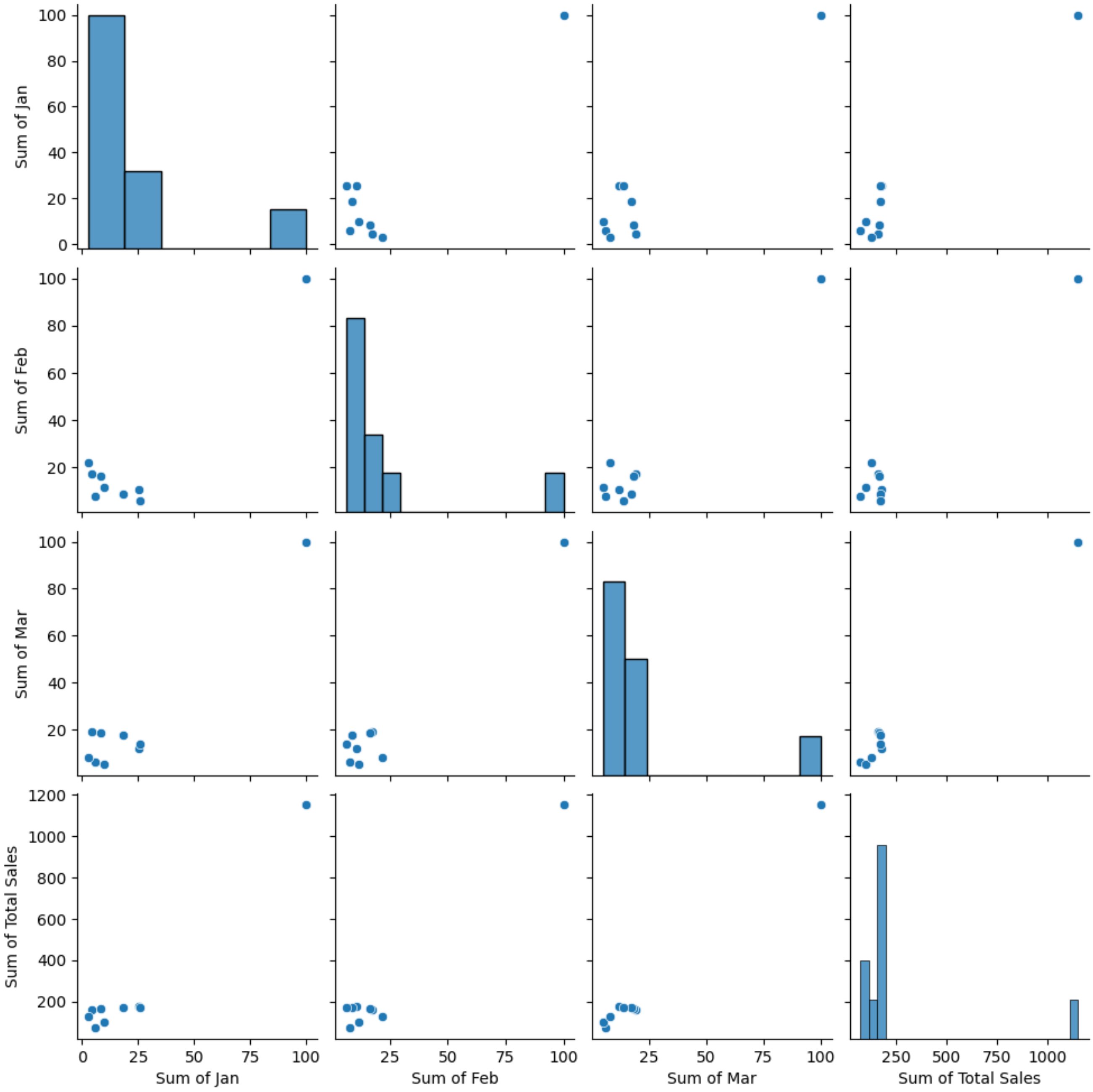
	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	5.62	7.73	6.16	75
1	4.21	17.27	19.21	160
2	9.83	11.60	5.17	101
3	2.81	21.91	7.88	127
4	25.28	10.57	11.82	179
5	8.15	16.24	18.47	167
6	18.54	8.76	17.49	171
7	25.56	5.93	13.79	170
8	100.00	100.00	100.00	1150

```
df.describe()
```

	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
count	9.000000	9.000000	9.000000	9.000000
mean	22.222222	22.223333	22.221111	255.555556
std	30.438329	29.612265	29.640999	337.332963
min	2.810000	5.930000	5.170000	75.000000
25%	5.620000	8.760000	7.880000	127.000000
50%	9.830000	11.600000	13.790000	167.000000
75%	25.280000	17.270000	18.470000	171.000000
max	100.000000	100.000000	100.000000	1150.000000

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7daeaa41f0d0>
```



```
sns.heatmap(df.corr())
```

```
<Axes: >
```

```
y=df['Sum of Feb']
x=df.drop(['Sum of Feb'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

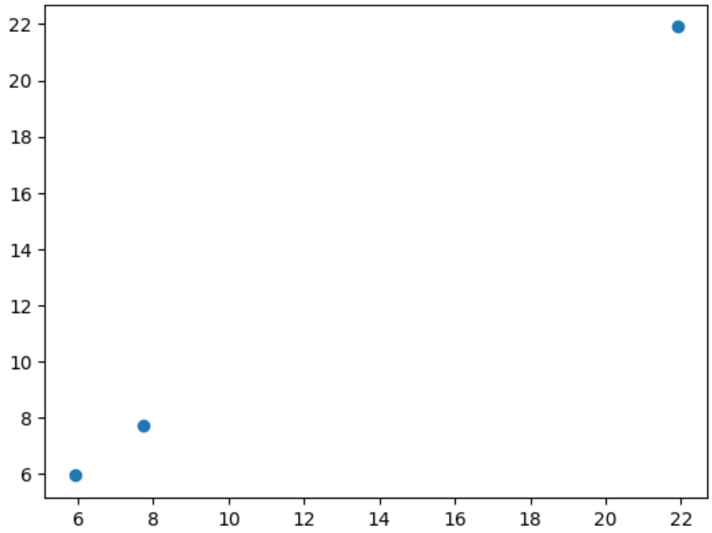
```
-0.0007206429545334458
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])

coeff
```

	Coefficient
Sum of Jan	-0.917397
Sum of Mar	-1.046100
Sum of Total Sales	0.257696

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7daea9b6bfd0>
```



```
model.score(x_test,y_test)
```

```
0.9999997321796058
```

```
df=pd.read_csv("/content/6_Salesworkload1.csv")
df
```

1 to 25 of 7658 entries Filter

index	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sales units	Turnover	Customer	Area (m2
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398560.0	1226244.0	NaN	953.04
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725.0	387810.0	NaN	720.48
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438400.0	654657.0	NaN	966.72
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309425.0	499434.0	NaN	1053.36
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165515.0	329397.0	NaN	1053.36
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	1713310.0	5617137.0	NaN	11735.16
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0	3107935.0	8714679.0	NaN	19865.64
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0	213680.0	1615341.0	NaN	8513.52
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.0	54915.0	290400.0	NaN	4842.72
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	59260.0	450015.0	NaN	5608.8

```
df.isna().sum()
```

```
MonthYear           0
Time index          8
Country             8
StoreID             8
City                8
Dept_ID             8
Dept. Name          8
HoursOwn            8
HoursLease          8
Sales units         8
Turnover            8
Customer         7658
Area (m2)           8
Opening hours       8
dtype: int64
```

```
df1=df.drop(["Customer","Country","Dept. Name","Opening hours","City"],axis=1)
df1=df1.dropna()
```

```
df1.isin(val).sum()
```

```
MonthYear        0
Time index       0
StoreID          0
Dept_ID          0
HoursOwn         0
HoursLease       0
Sales units      0
Turnover         0
Area (m2)      850
dtype: int64
```

```
val=df["HoursOwn"]=="?"
print(df.index[val])
```

```
Int64Index([2966, 5889], dtype='int64')
```

```
val=["#NV"]
df1["Area (m2)"].isin(val).sum()
df1=df1.drop([2966,5889],axis=0)
```

```
df1=df1.drop(["Area (m2)"],axis=1)
df1
```

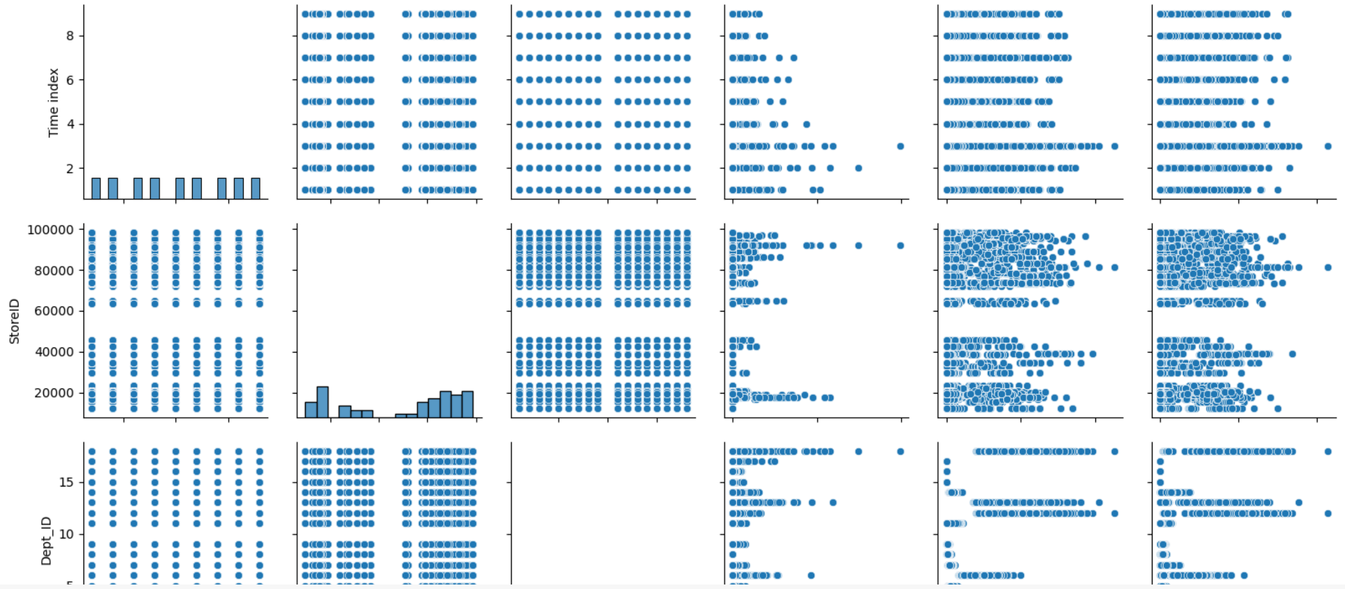
```
   MonthYear  Time index  StoreID  Dept_ID  HoursOwn  HoursLease  Sales units  Turnover
0    10.2016         1.0  88253.0      1.0  3184.764         0.0     398560.0  1226244.0
1    10.2016         1.0  88253.0      2.0  1582.941         0.0      82725.0   387810.0
2    10.2016         1.0  88253.0      3.0    47.205         0.0     438400.0   654657.0
3    10.2016         1.0  88253.0      4.0  1623.852         0.0     309425.0   499434.0
4    10.2016         1.0  88253.0      5.0  1759.173         0.0     165515.0   329397.0
...      ...         ...      ...      ...       ...         ...          ...        ...
```

```
df1.describe()
```

```
       Time index       StoreID      Dept_ID   HoursLease   Sales units      Turnover
count 7648.000000   7648.000000  7648.000000  7648.000000  7.648000e+03  7.648000e+03
mean     4.999869  61999.574268     9.472019    22.041841  1.076492e+06  3.721465e+06
std      2.582369  29923.753974     5.337296   133.316467  1.728290e+06  6.004067e+06
min      1.000000  12227.000000     1.000000     0.000000  0.000000e+00  0.000000e+00
25%      3.000000  29650.000000     5.000000     0.000000  5.455375e+04  2.724480e+05
50%      5.000000  76852.000000     9.000000     0.000000  2.932300e+05  9.315390e+05
75%      7.000000  87703.000000    14.000000     0.000000  9.164325e+05  3.259014e+06
max      9.000000  98422.000000    18.000000  3984.000000  1.124296e+07  4.271739e+07
```

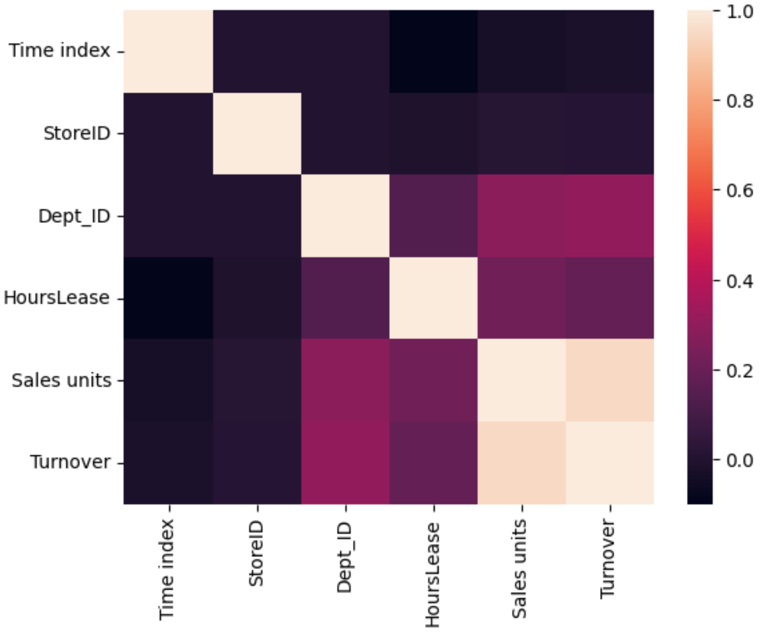
```
sns.pairplot(df1)
```

```
<seaborn.axisgrid.PairGrid at 0x7daea73c8c40>
```



```
sns.heatmap(df1.corr())
```

```
<ipython-input-127-3ed1a1a51dc0>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future v
  sns.heatmap(df1.corr())
<Axes: >
```



```
y=df1['Sales units']
x=df1.drop(['Sales units'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
model=LinearRegression()

model.fit(x_train,y_train)

model.intercept_
```

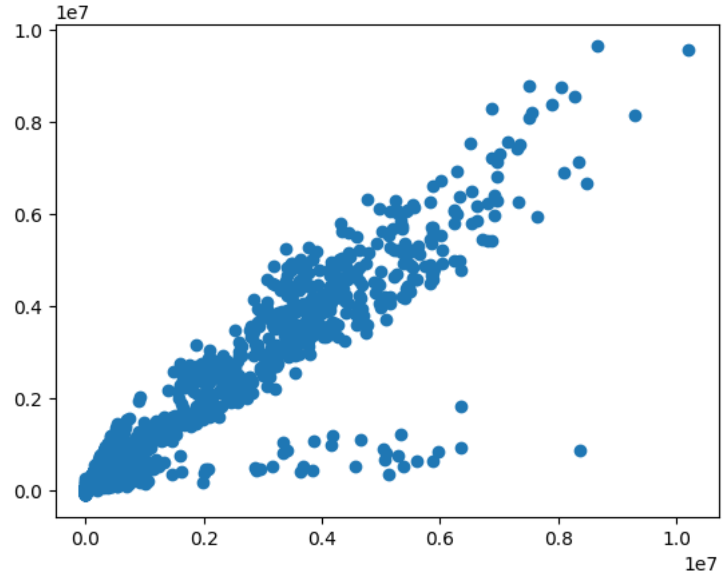
```
85049.7310071832
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])

coeff
```

	Coefficient
MonthYear	4262.112939
Time index	-4080.289435
StoreID	0.187693
Dept_ID	-10025.975367
HoursOwn	16.058768
HoursLease	286.756891
Turnover	0.251824

```
prediction=model.predict(x_test)

plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7daea61af7c0>
```



```
model.score(x_test,y_test)
```

```
0.8835992990450046
```

```
df=pd.read_csv("/content/5_Instagram data.csv")
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-134-9b19e53c1253> in <cell line: 1>()
----> 1 df=pd.read_csv("/content/5_Instagram data.csv")

                                  10 frames
/usr/local/lib/python3.10/dist-packages/pandas/_libs/parsers.pyx in pandas._libs.parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa0 in position 288: invalid start byte
```

SEARCH STACK OVERFLOW

0s completed at 10:54 PM