DAY 9:

Horse Dataset

Linear Regression

In [1]:

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

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\15_horse.csv")[0:50]
df
```

10	28.03.2018	Happy Valley	8	1800	Gress	1310000	9	M F Poon	53	Sverige	...
11	11.04.2018	Happy Valley	6	1650	Gress	1310000	11	W M Lai	55	Sverige	...
12	25.04.2018	Happy Valley	3	2200	Gress	1310000	2	W M Lai	54	Sverige	...
13	09.05.2018	Happy Valley	7	1650	Gress	1310000	3	W M Lai	54	Sverige	...
14	22.09.2018	Sha Tin	4	1600	Gress	920000	11	C Y Ho	57	Sverige	...
15	07.10.2018	Sha Tin	6	1600	Gress	920000	9	C Y Ho	56	Sverige	...
16	02.12.2018	Sha Tin	3	1800	Dirt	920000	1	C Schofield	57	Sverige	...
17	23.12.2018	Sha Tin	2	2000	Gress	920000	6	Silvestre De Sousa	59	Sverige	...

In [3]:

```
df.head()
```

Out[3]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Country
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	Sverige
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	Sverige
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	Sverige
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	Sverige
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	Sverige

5 rows × 21 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Dato               50 non-null     object
 1   Track              50 non-null     object
 2   Race Number        50 non-null     int64 
 3   Distance           50 non-null     int64 
 4   Surface            50 non-null     object
 5   Prize money        50 non-null     int64 
 6   Starting position  50 non-null     int64 
 7   Jockey             50 non-null     object
 8   Jockey weight      50 non-null     int64 
 9   Country            50 non-null     object
 10  Horse age          50 non-null     int64 
 11  TrainerName        50 non-null     object
 12  Race time          50 non-null     object
 13  Path               50 non-null     int64 
 14  Final place        50 non-null     int64 
 15  FGrating           50 non-null     int64 
 16  Odds               50 non-null     object
 17  RaceType           50 non-null     object
 18  HorseId            50 non-null     int64 
 19  JockeyId           50 non-null     int64 
 20  TrainerID          50 non-null     int64 
dtypes: int64(12), object(9)
memory usage: 8.3+ KB
```

In [5]:

```
#to display summary of statistics
df.describe()
```

Out[5]:

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	Path	I
count	50.000000	50.000000	5.000000e+01	50.000000	50.000000	50.000000	50.000000	50.
mean	6.560000	1438.000000	3.954000e+06	6.460000	56.120000	7.400000	1.460000	6.
std	2.383275	326.165102	4.632386e+06	3.375845	2.512337	0.832993	1.501156	2.
min	1.000000	1000.000000	9.200000e+05	1.000000	51.000000	5.000000	0.000000	1.
25%	5.000000	1200.000000	1.310000e+06	4.000000	54.250000	7.000000	0.000000	4.
50%	7.000000	1400.000000	2.500000e+06	6.000000	56.500000	8.000000	1.000000	6.
75%	8.000000	1637.500000	4.000000e+06	9.000000	57.000000	8.000000	2.000000	8.
max	10.000000	2400.000000	1.850000e+07	14.000000	60.000000	9.000000	5.000000	11.

In [6]:

```
#to display cloumn heading
df.columns
```

Out[6]:

```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize mone
y',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse a
ge',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odd
s',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],
      dtype='object')
```

EDA and VISUALIZATION

In [7]:

```
df1=df[['Race Number', 'Distance','Prize money',
       'Starting position','HorseId', 'JockeyId', 'TrainerID']]
df1
```

Out[7]:

	Race Number	Distance	Prize money	Starting position	HorseId	JockeyId	TrainerID
0	10	1400	1310000	6	1736	8656	6687
1	10	1400	1310000	14	1736	8659	6687
2	10	1400	1310000	8	1736	8659	6687
3	9	1600	1310000	13	1736	8453	6687
4	9	1600	1310000	9	1736	8659	6687
5	1	1800	1310000	4	1736	8659	6687
6	9	1800	1310000	9	1736	8655	6687
7	5	1800	1310000	6	1736	8443	6687
8	8	1800	1310000	3	1736	8659	6687
9	10	1600	1310000	8	1736	8659	6687
10	8	1800	1310000	9	1736	8658	6687
11	6	1650	1310000	11	1736	8674	6687
12	3	2200	1310000	2	1736	8674	6687
13	7	1650	1310000	3	1736	8674	6687
14	4	1600	920000	11	1736	8659	6687
15	6	1600	920000	9	1736	8659	6687
16	3	1800	920000	1	1736	8655	6687
17	2	2000	920000	6	1736	4340	6687
18	1	2000	920000	4	1736	8774	6687
19	9	1800	1860000	5	10394	8650	6685
20	7	1000	3000000	8	19373	8650	6684
21	7	1200	4000000	2	19373	8663	6684
22	7	1200	4000000	8	19373	8663	6684
23	5	1200	18500000	9	19373	8663	6684
24	10	1400	3000000	10	19373	8654	6684
25	7	1200	10000000	3	19373	8453	6684
26	9	1400	10000000	2	19373	8453	6684
27	7	1200	2500000	4	19373	8651	6684
28	7	1200	4000000	6	19373	8651	6684
29	7	1200	16000000	2	19373	8651	6684
30	7	1000	3000000	4	20799	8609	6535
31	7	1200	4000000	4	20799	8609	6535
32	7	1200	4000000	9	20799	8609	6535
33	5	1200	18500000	10	20799	8609	6535
34	7	1000	3000000	6	20799	8609	6535
35	7	1200	10000000	1	20799	8609	6535
36	7	1200	2500000	2	20799	8658	6535

	Race Number	Distance	Prize money	Starting position	HorseId	JockeyId	TrainerID
37	7	1200	4000000	5	20799	8653	6535
38	4	1400	2500000	10	20799	8453	6535
39	7	1000	3250000	7	20799	8659	6535
40	7	1200	4250000	4	20799	8659	6535
41	3	1000	1950000	9	20799	4340	6535
42	7	1200	1950000	8	20799	4340	6535
43	4	2400	18000000	8	20816	8438	6585
44	3	1400	2500000	3	21512	8651	6726
45	10	1400	3000000	12	21512	8655	6726
46	8	1400	2500000	4	21512	8655	6726
47	7	1200	2500000	11	21512	8655	6726
48	7	1200	4000000	3	21512	8655	6726
49	4	1400	2500000	8	21512	8650	6726

In [8]:

```
df1.info()
```

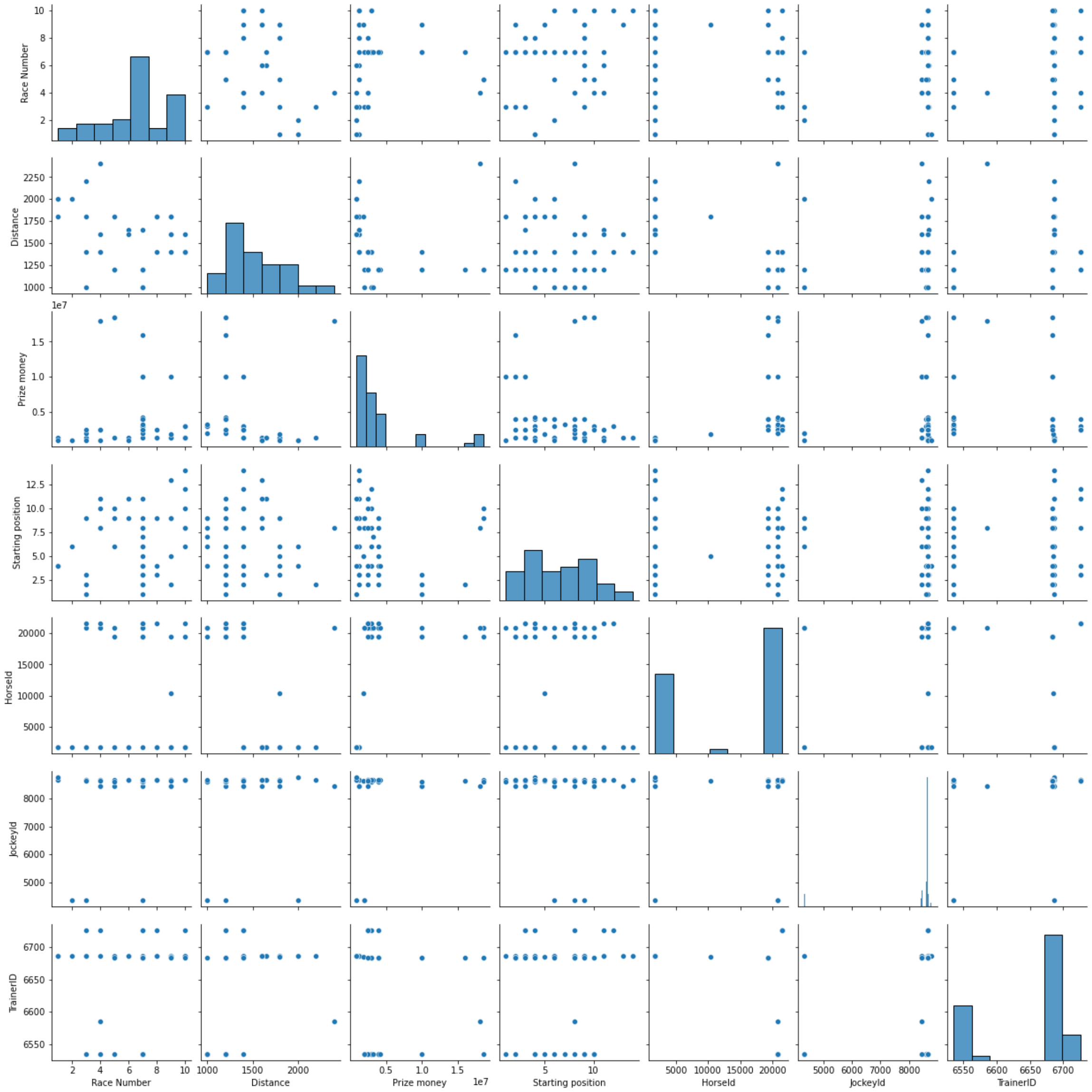
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Race Number        50 non-null     int64
 1   Distance           50 non-null     int64
 2   Prize money        50 non-null     int64
 3   Starting position  50 non-null     int64
 4   HorseId            50 non-null     int64
 5   JockeyId           50 non-null     int64
 6   TrainerID          50 non-null     int64
dtypes: int64(7)
memory usage: 2.9 KB
```

In [9]:

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

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x2323b909a00>
```



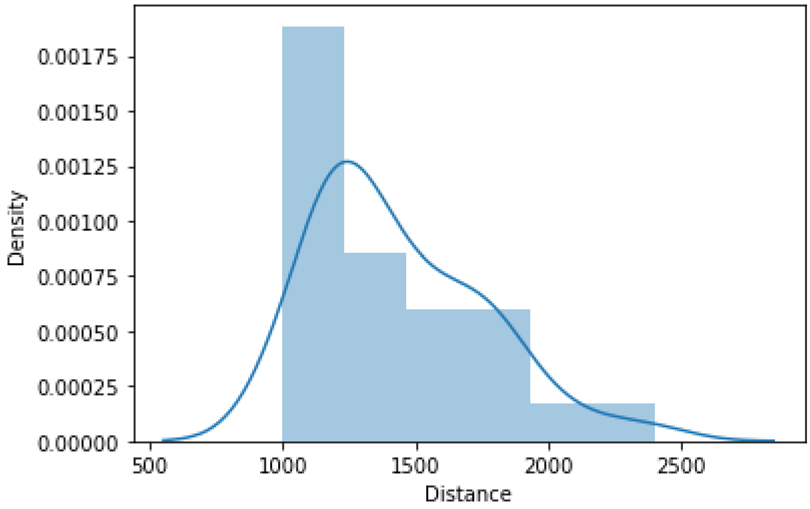
In [10]:

```
sns.distplot(df['Distance'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<AxesSubplot:xlabel='Distance', ylabel='Density'>
```

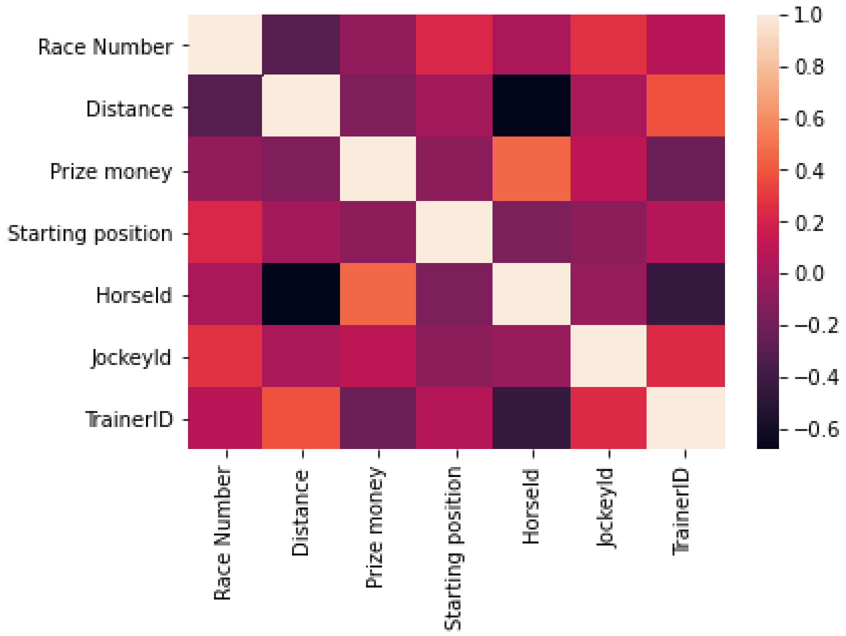


In [11]:

```
data=df1[[
       'Race Number', 'Distance','Prize money',
       'Starting position','HorseId', 'JockeyId', 'TrainerID']]
sns.heatmap(data.corr())
```

Out[11]:

```
<AxesSubplot:>
```



to Train the model-Model buliding

we are going to split our data into two variable where x is a independent and y is dependent on x

In [12]:

```
x=data[['Race Number', 'Distance','Prize money',
       'Starting position','HorseId', 'JockeyId']]
y=data['TrainerID']
```

In [13]:

```
# to split my dataset into test and train data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [14]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:

```
LinearRegression()
```

In [15]:

```
print(lr.intercept_)
```

```
6216.649739683464
```

In [16]:

```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[16]:

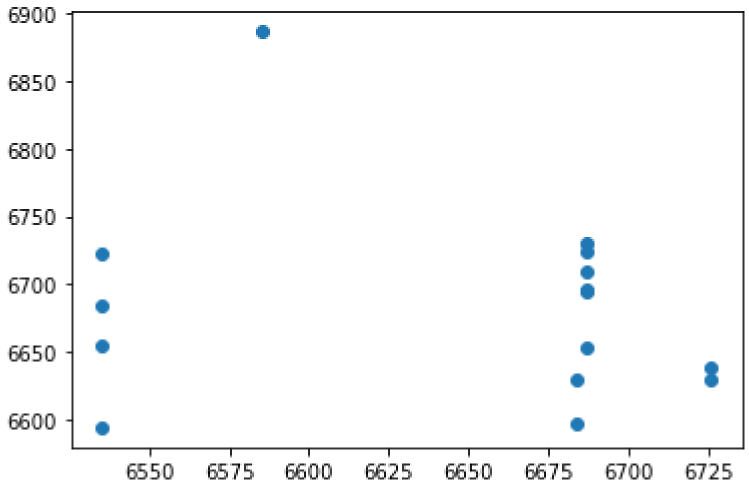
	Co-effecient
Race Number	0.047406
Distance	0.155011
Prize money	0.000004
Starting position	8.276341
HorseId	0.000586
JockeyId	0.017141

In [17]:

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

Out[17]:

```
<matplotlib.collections.PathCollection at 0x2323fcb5fa0>
```



In [18]:

```
print(lr.score(x_test,y_test))
```

```
-1.5235203943057791
```

In [19]:

```
lr.score(x_train,y_train)
```

Out[19]:

```
0.4797777130422708
```

Ridge Regression

In [20]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[21]:

```
-1.4825293527028314
```

Lasso Regression

In [22]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:

```
Lasso(alpha=10)
```

In [23]:

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

Out[23]:

```
-1.3558809738208026
```

In [ ]:

```

```