In [20]:

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

In [21]:

```python
df=pd.read_csv("/content/15_Horse Racing Results.csv")
df
```

		Tin						Prebble		
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	S
...	...	...	...	...	...	...	...	...	...	
27003	14.06.2020	Sha Tin	11	1200	Gress	1450000	6	A Hamelin	59	Au
27004	21.06.2020	Sha Tin	2	1200	Gress	967000	7	K C Leung	57	Au
27005	21.06.2020	Sha Tin	4	1200	Gress	967000	6	Blake Shinn	57	Au
27006	21.06.2020	Sha Tin	5	1200	Gress	967000	14	Joao Moreira	57	Z
27007	21.06.2020	Sha Tin	11	1200	Gress	1450000	7	C Schofield	55	Z

27008 rows × 21 columns

In [22]:

```python
df.head()
```

Out[22]:

	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

DATA CLEANING AND DATA PREPROCESSING

In [23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27008 entries, 0 to 27007
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Dato               27008 non-null  object
 1   Track              27008 non-null  object
 2   Race Number        27008 non-null  int64 
 3   Distance           27008 non-null  int64 
 4   Surface            27008 non-null  object
 5   Prize money        27008 non-null  int64 
 6   Starting position  27008 non-null  int64 
 7   Jockey             27008 non-null  object
 8   Jockey weight      27008 non-null  int64 
 9   Country            27008 non-null  object
 10  Horse age          27008 non-null  int64 
 11  TrainerName        27008 non-null  object
 12  Race time          27008 non-null  object
 13  Path               27008 non-null  int64 
 14  Final place        27008 non-null  int64 
 15  FGrating           27008 non-null  int64 
 16  Odds               27008 non-null  object
 17  RaceType           27008 non-null  object
 18  HorseId            27008 non-null  int64 
 19  JockeyId           27008 non-null  int64 
 20  TrainerID          27008 non-null  int64 
dtypes: int64(12), object(9)
memory usage: 4.3+ MB
```

In [24]:
```
df.describe()
```

Out[24]:

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	
count	27008.000000	27008.000000	2.700800e+04	27008.000000	27008.000000	27008.000000	270
mean	5.268624	1401.666173	1.479445e+06	6.741447	55.867373	5.246408	
std	2.780088	276.065045	2.162109e+06	3.691071	2.737006	1.519880	
min	1.000000	1000.000000	6.600000e+05	1.000000	47.000000	2.000000	
25%	3.000000	1200.000000	9.200000e+05	4.000000	54.000000	4.000000	
50%	5.000000	1400.000000	9.670000e+05	7.000000	56.000000	5.000000	
75%	8.000000	1650.000000	1.450000e+06	10.000000	58.000000	6.000000	
max	11.000000	2400.000000	2.800000e+07	14.000000	63.000000	12.000000	

In [25]:
```
df.columns
```

Out[25]:
```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse ag
e',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],
      dtype='object')
```

In [26]:
```
df1=df.dropna(axis=1)
df1
```

		Tin						Frebbie		
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	S
...	...	...	...	...	...	...	...	...	...	
27003	14.06.2020	Sha Tin	11	1200	Gress	1450000	6	A Hamelin	59	Au
27004	21.06.2020	Sha Tin	2	1200	Gress	967000	7	K C Leung	57	Au
27005	21.06.2020	Sha Tin	4	1200	Gress	967000	6	Blake Shinn	57	Au
27006	21.06.2020	Sha Tin	5	1200	Gress	967000	14	Joao Moreira	57	Z
27007	21.06.2020	Sha Tin	11	1200	Gress	1450000	7	C Schofield	55	Z

27008 rows × 21 columns

In [27]:
```
df1.columns
```

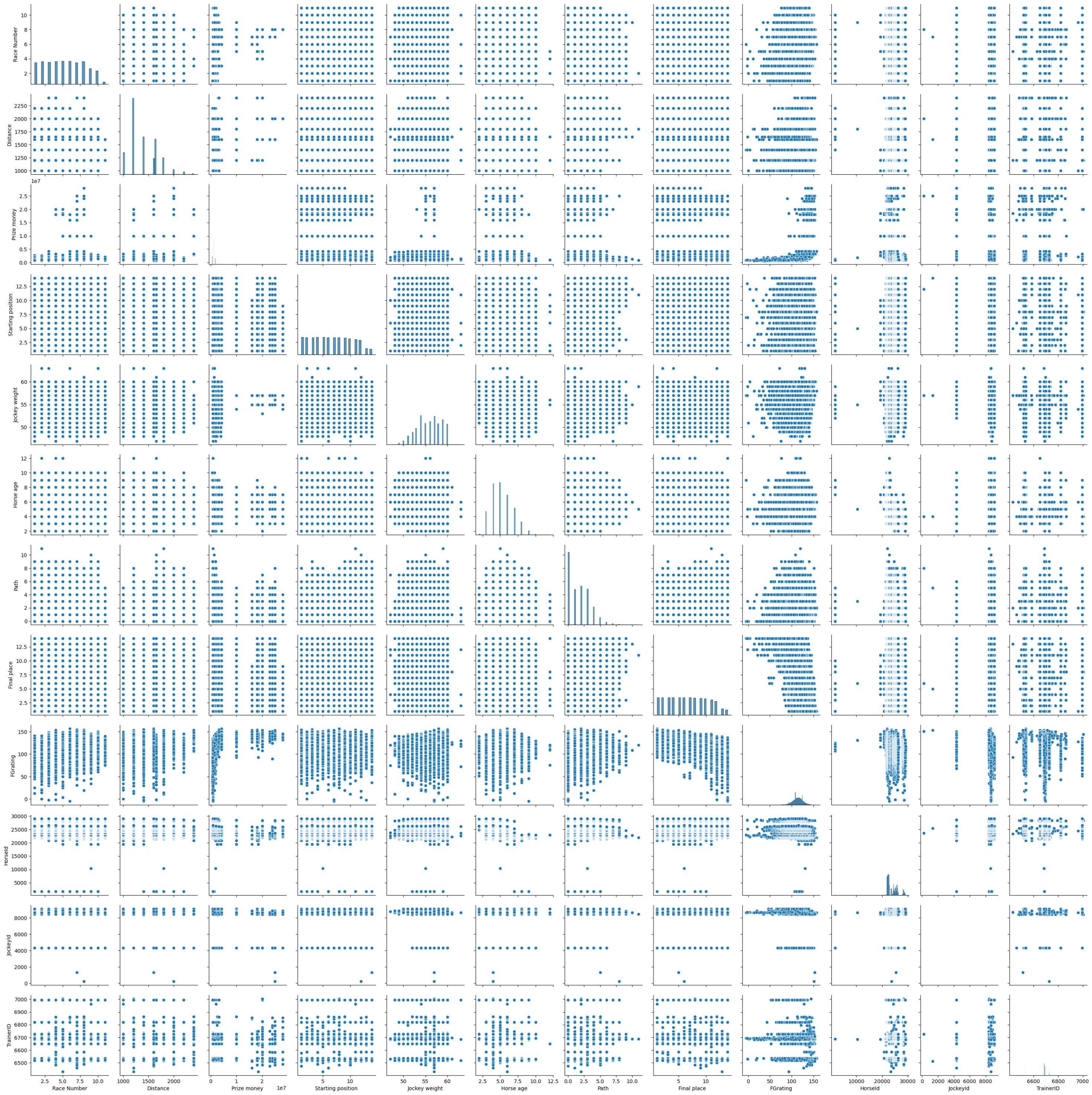
Out[27]:
```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse ag
e',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],
      dtype='object')
```

In [28]:
```
df1=df1[['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse age',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID']]
```

EDA AND VISUALIZATION

In [29]:
```
sns.pairplot(df1)
```

Out[29]:
```
<seaborn.axisgrid.PairGrid at 0x7ea8e39bbb80>
```



In [30]:
```
sns.distplot(df1['Distance'])
```

```
<ipython-input-30-55baf5480dab>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function wit
h
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://g
ist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df1['Distance'])
```

Out[30]:
```
<Axes: xlabel='Distance', ylabel='Density'>
```

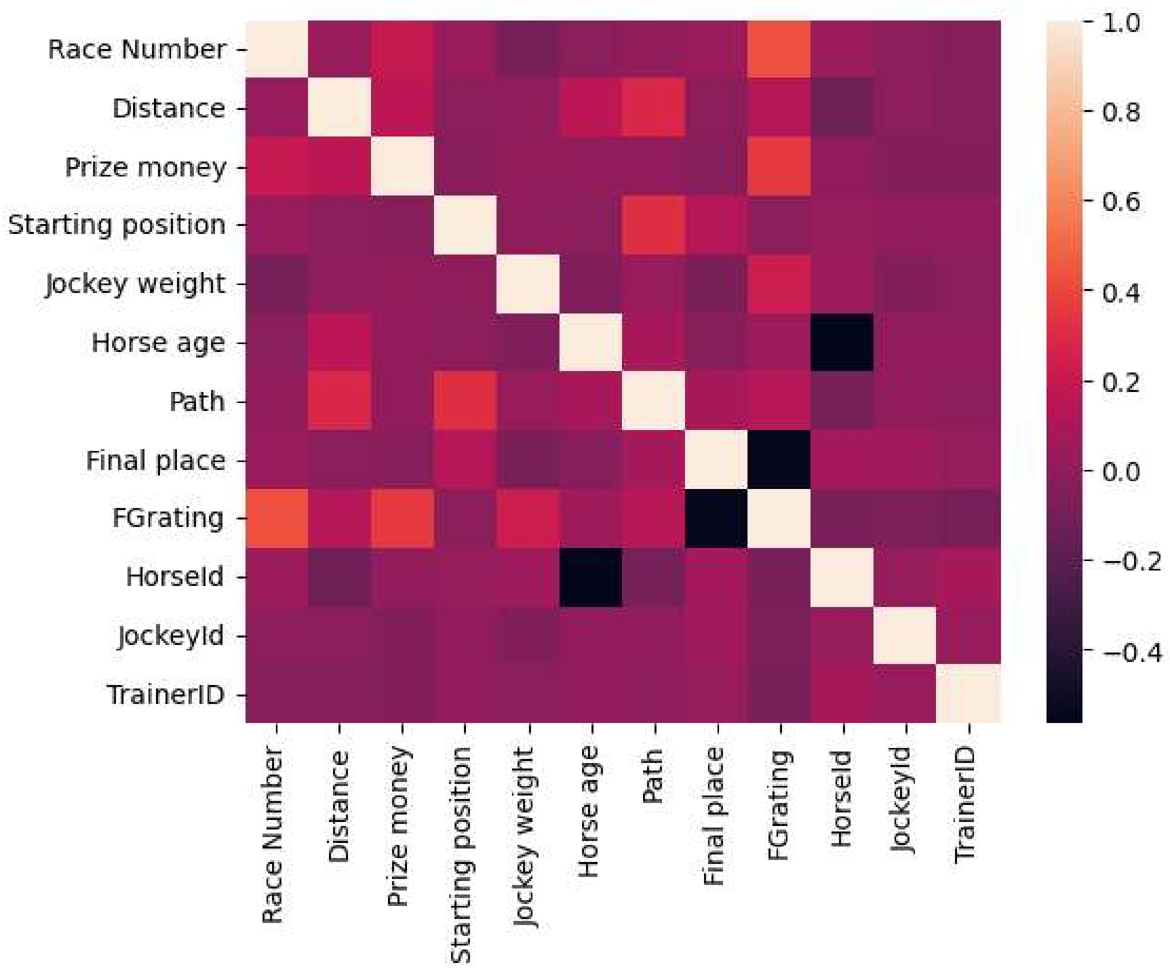
In [31]:

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

```
<ipython-input-31-3ed1a1a51dc0>:1: FutureWarning: The default value of numeri
c_only in DataFrame.corr is deprecated. In a future version, it will default
to False. Select only valid columns or specify the value of numeric_only to s
ilence this warning.
  sns.heatmap(df1.corr())
```

Out[31]:

```
<Axes: >
```



TO TRAIN THE MODEL AND MODEL BULDING

In [32]:

```
x=df[['Race Number', 'Prize money',
       'Starting position',  'Jockey weight',  'Horse age', 'Final place', 'FG
y=df['Distance']
```

In [33]:

```
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 [34]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[34]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [35]:
```
lr.intercept_
```

Out[35]:
```
1702.2246084652907
```

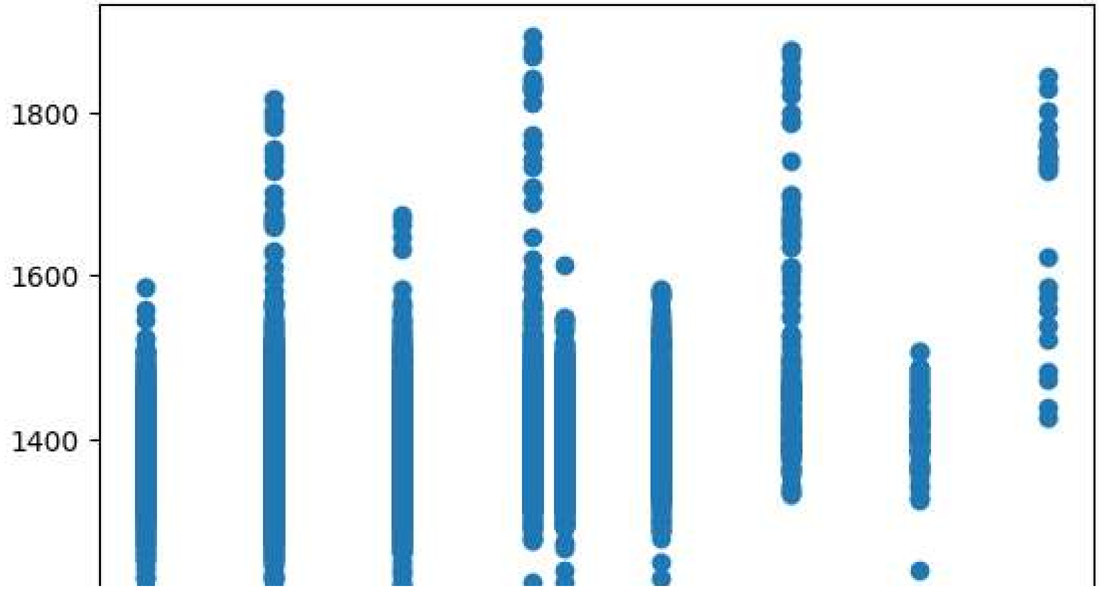
In [36]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[36]:

	Co-efficient
Race Number	-6.066791
Prize money	0.000015
Starting position	-0.576600
Jockey weight	-1.939342
Horse age	22.775448
Final place	6.388438
FGrating	3.067410
HorseId	-0.006693
JockeyId	-0.006062
TrainerID	-0.071689

In [37]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[37]:
```
<matplotlib.collections.PathCollection at 0x7ea8d2eb3ee0>
```



ACCURACY

In [38]:
```
lr.score(x_test,y_test)
```

Out[38]:
```
0.06099752116422741
```

In [39]:
```
lr.score(x_train,y_train)
```

Out[39]:
```
0.06411986650423529
```

In [40]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [41]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[41]:
```
Ridge(alpha=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [42]:
```
rr.score(x_test,y_test)
```

Out[42]:
```
0.06099829608026752
```

In [43]:
```
rr.score(x_train,y_train)
```

Out[43]:
```
0.06411986523298252
```

In [44]:

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

Out[44]:

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [45]:

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

Out[45]:

```
0.05982733108440974
```

In [46]:

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

Out[46]:

```
0.06224170620823566
```