PROJECT:1

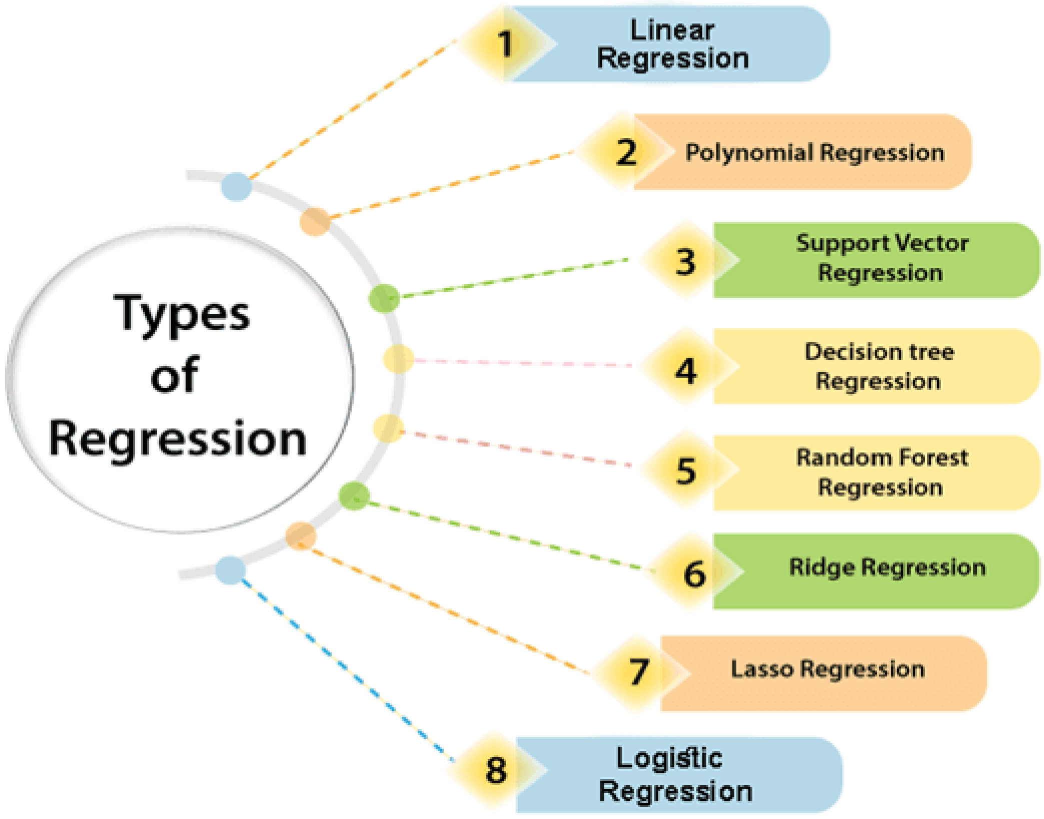
Estimating Movie Box Office Revenue :

```
Estimating Movie Box Office Revenue : Problem Statement: Estimating Movie Box Office Revenue
Project Description: Build a regression model to estimate the box office revenue of movies based on factors such as
genre, production budget, release date, and marketing efforts.
Domain: Film Industry
Dataset Link: https://www.kaggle.com/datasets/kalilurrahman/top-box-office-revenue-data-english-movies?
select=bomojobrandindices.csv
```

- Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables.
- More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed.
- It predicts continuous/real values such as temperature, age, salary, price, etc.

REGRESSION PROBLEM:

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression
- Ridge Regression
- Lasso Regression:



LINEAR REGRESSION:

In [1]:
```
import pandas as pd
```

In [66]:
```
data = pd.read_csv("bomojobrandindices.csv")
```

In [67]:
```
data
```

Out[67]:

	Brand\tTotal\tReleases\t#1 Release\tLifetime Gross
0	Marvel Comics\t15806336901\t69\tAvengers: Endg...
1	Legendary Pictures\t7018798067\t56\tJurassic W...
2	Lucasfilm\t6325022918\t39\tStar Wars: Episode ...
3	Pixar\t6078217662\t28\tIncredibles 2\t608581744
4	DC Comics\t5815645953\t46\tThe Dark Knight\t53...
5	DreamWorks Animation\t5792217737\t41\tShrek 2\...
6	Vertigo Entertainment\t3154664176\t41\tIt\t327...
7	Bad Robot\t3077078931\t15\tStar Wars: Episode ...
8	Walt Disney Animation Studios\t2774912904\t15\...
9	Illumination Entertainment\t2759505881\t13\tTh...
10	Blumhouse Productions\t2451844676\t49\tGet Out...
11	Hasbro\t2079485824\t17\tTransformers: Revenge ...
12	Nickelodeon\t1930746182\t26\tTeenage Mutant Ni...
13	Sony Pictures Animation\t1918763863\t21\tSpide...
14	Walden Media\t1842852183\t38\tThe Chronicles o...
15	Blue Sky\t1740428763\t13\tIce Age 3: Dawn of t...
16	Stephen King\t1729648950\t49\tIt\t327481748
17	MTV\t1520456359\t36\tThe Longest Yard\t158119460
18	Platinum Dunes\t1422911399\t19\tTeenage Mutant...
19	Saturday Night Live - Alumni Debuts\t102359603...
20	Dark Horse Comics\t947671338\t16\t300\t210614939
21	Tim Burton-Johnny Depp\t889955830\t8\tAlice in...
22	Warner Animation Group\t786497181\t8\tThe Lego...
23	Tyler Perry\t765635362\t16\tMadea Goes to Jail...
24	CBS Films\t652824187\t29\tScary Stories to Tel...
25	John Grisham\t645661825\t10\tThe Firm\t158348367

	Brand\tTotal\tReleases\t#1 Release\tLifetime Gross
26	Robert Ludlum\t645459186\t6\tThe Bourne Ultima...
27	MonsterVerse\t580145113\t4\tGodzilla\t200676069
28	Hanna-Barbera\t578586146\t10\tScooby-Doo\t1532...
29	Nicholas Sparks\t574728259\t11\tThe Notebook\t...
30	Philip K. Dick\t495237720\t14\tMinority Report...
31	Dark Castle\t454625110\t16\tUnknown\t63686397
32	National Lampoon\t436669213\t21\tNational Lamp...
33	Disney Channel\t359952780\t8\tHigh School Musi...
34	Roald Dahl\t356238494\t7\tCharlie and the Choc...
35	Saturday Night Live\t346533876\t11\tWayne's Wo...
36	DisneyToon Studios\t337576791\t8\tPlanes\t9028...
37	Aardman\t333777829\t16\tChicken Run\t106834564
38	Laika\t300158323\t6\tCoraline\t75286229
39	Alan Moore\t276088604\t4\tWatchmen\t107509799
40	Amazon Studios\t204376048\t43\tManchester by t...
41	Clive Barker\t171686009\t10\tCandyman\t61186570
42	Disneynature\t151620585\t8\tEarth\t32011576
43	Broken Lizard\t73338237\t5\tSuper Troopers 2\t...
44	Studio Ghibli\t70666453\t26\tThe Secret World ...

In [68]:
```
data = pd.read_csv("bomojobrandindices.csv",sep = '\t')
```

In [28]:
```
data
```

Out[28]:

	Brand	Total	Releases	#1 Release	Lifetime Gross
0	Marvel Comics	15806336901	69	Avengers: Endgame	858373000
1	Legendary Pictures	7018798067	56	Jurassic World	652270625
2	Lucasfilm	6325022918	39	Star Wars: Episode VII - The Force Awakens	936662225
3	Pixar	6078217662	28	Incredibles 2	608581744
4	DC Comics	5815645953	46	The Dark Knight	533345358
5	DreamWorks Animation	5792217737	41	Shrek 2	441226247
6	Vertigo Entertainment	3154664176	41	It	327481748
7	Bad Robot	3077078931	15	Star Wars: Episode VII - The Force Awakens	936662225
8	Walt Disney Animation Studios	2774912904	15	Frozen II	477373578
9	Illumination Entertainment	2759505881	13	The Secret Life of Pets	368384330
10	Blumhouse Productions	2451844676	49	Get Out	176040665
11	Hasbro	2079485824	17	Transformers: Revenge of the Fallen	402111870
12	Nickelodeon	1930746182	26	Teenage Mutant Ninja Turtles	191204754
13	Sony Pictures Animation	1918763863	21	Spider-Man: Into the Spider-Verse	190241310
14	Walden Media	1842852183	38	The Chronicles of Narnia: The Lion the Witch a...	291710957
15	Blue Sky	1740428763	13	Ice Age 3: Dawn of the Dinosaurs	196573705
16	Stephen King	1729648950	49	It	327481748
17	MTV	1520456359	36	The Longest Yard	158119460
18	Platinum Dunes	1422911399	19	Teenage Mutant Ninja Turtles	191204754
19	Saturday Night Live - Alumni Debuts	1023596031	30	Bridesmaids	169106725
20	Dark Horse Comics	947671338	16	300	210614939
21	Tim Burton-Johnny Depp	889955830	8	Alice in Wonderland	334191110
22	Warner Animation Group	786497181	8	The Lego Movie	257760692
23	Tyler Perry	765635362	16	Madea Goes to Jail	90508336
24	CBS Films	652824187	29	Scary Stories to Tell in the Dark	68947075
25	John Grisham	645661825	10	The Firm	158348367

	Brand	Total	Releases	#1 Release	Lifetime Gross
26	Robert Ludlum	645459186	6	The Bourne Ultimatum	227471070
27	MonsterVerse	580145113	4	Godzilla	200676069
28	Hanna-Barbera	578586146	10	Scooby-Doo	153294164
29	Nicholas Sparks	574728259	11	The Notebook	81001787
30	Philip K. Dick	495237720	14	Minority Report	132072926
31	Dark Castle	454625110	16	Unknown	63686397
32	National Lampoon	436669213	21	National Lampoon's Animal House	120091123
33	Disney Channel	359952780	8	High School Musical 3: Senior Year	90559416
34	Roald Dahl	356238494	7	Charlie and the Chocolate Factory	206459076
35	Saturday Night Live	346533876	11	Wayne's World	121697323
36	DisneyToon Studios	337576791	8	Planes	90288712
37	Aardman	333777829	16	Chicken Run	106834564
38	Laika	300158323	6	Coraline	75286229
39	Alan Moore	276088604	4	Watchmen	107509799
40	Amazon Studios	204376048	43	Manchester by the Sea	47695371
41	Clive Barker	171686009	10	Candyman	61186570
42	Disneynature	151620585	8	Earth	32011576
43	Broken Lizard	73338237	5	Super Troopers 2	30617396
44	Studio Ghibli	70666453	26	The Secret World of Arrietty	19202743

In [69]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Brand           45 non-null     object
 1   Total           45 non-null     int64 
 2   Releases        45 non-null     int64 
 3   #1 Release      45 non-null     object
 4   Lifetime Gross  45 non-null     int64 
dtypes: int64(3), object(2)
memory usage: 1.9+ KB
```

In [30]:
```python
data.isna().sum()
```

Out[30]:
```
Brand             0
Total             0
Releases          0
#1 Release        0
Lifetime Gross    0
dtype: int64
```

In [31]:
```python
data.dropna(inplace = True)
```

In [32]:
```python
data.isna().sum()
```

Out[32]:
```
Brand             0
Total             0
Releases          0
#1 Release        0
Lifetime Gross    0
dtype: int64
```

In [33]:
```
data.describe()
```

Out[33]:

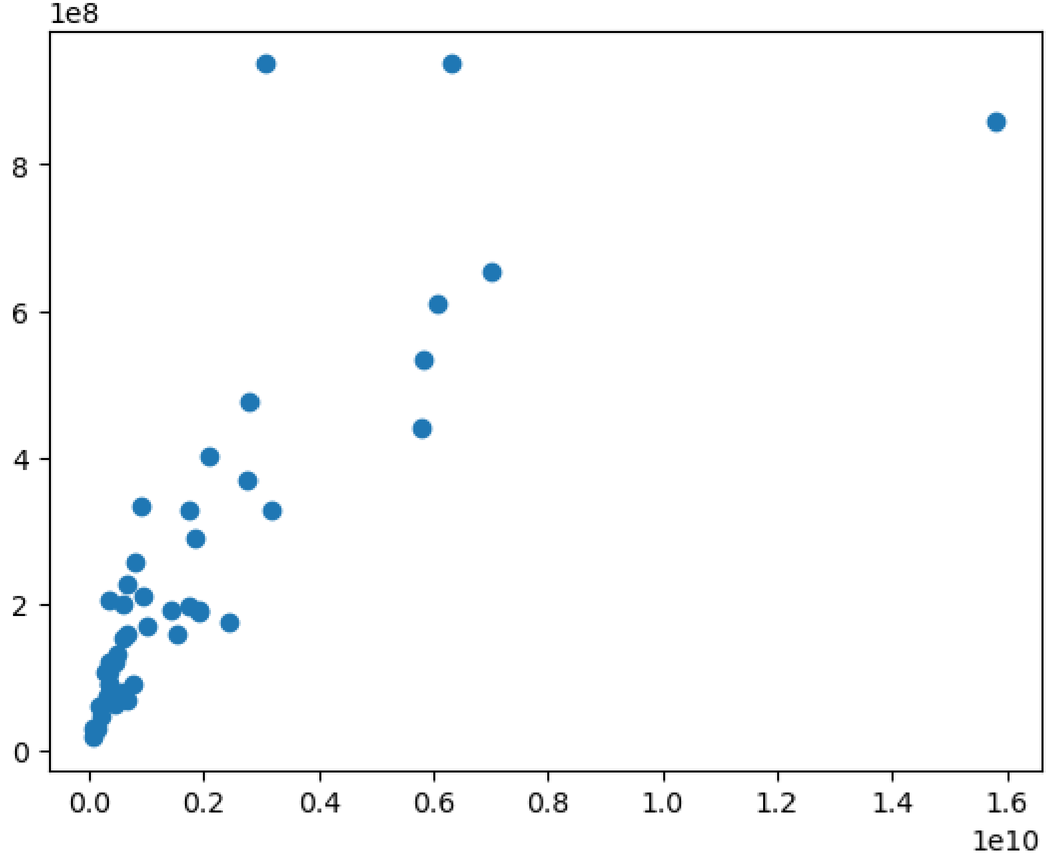
	Total	Releases	Lifetime Gross
count	4.500000e+01	45.000000	4.500000e+01
mean	1.948863e+09	21.822222	2.560482e+08
std	2.806106e+09	16.035881	2.336907e+08
min	7.066645e+07	4.000000	1.920274e+07
25%	3.599528e+08	10.000000	9.055942e+07
50%	7.864972e+08	16.000000	1.902413e+08
75%	2.079486e+09	30.000000	3.274817e+08
max	1.580634e+10	69.000000	9.366622e+08

In [36]:
```
import matplotlib.pyplot as plt
```

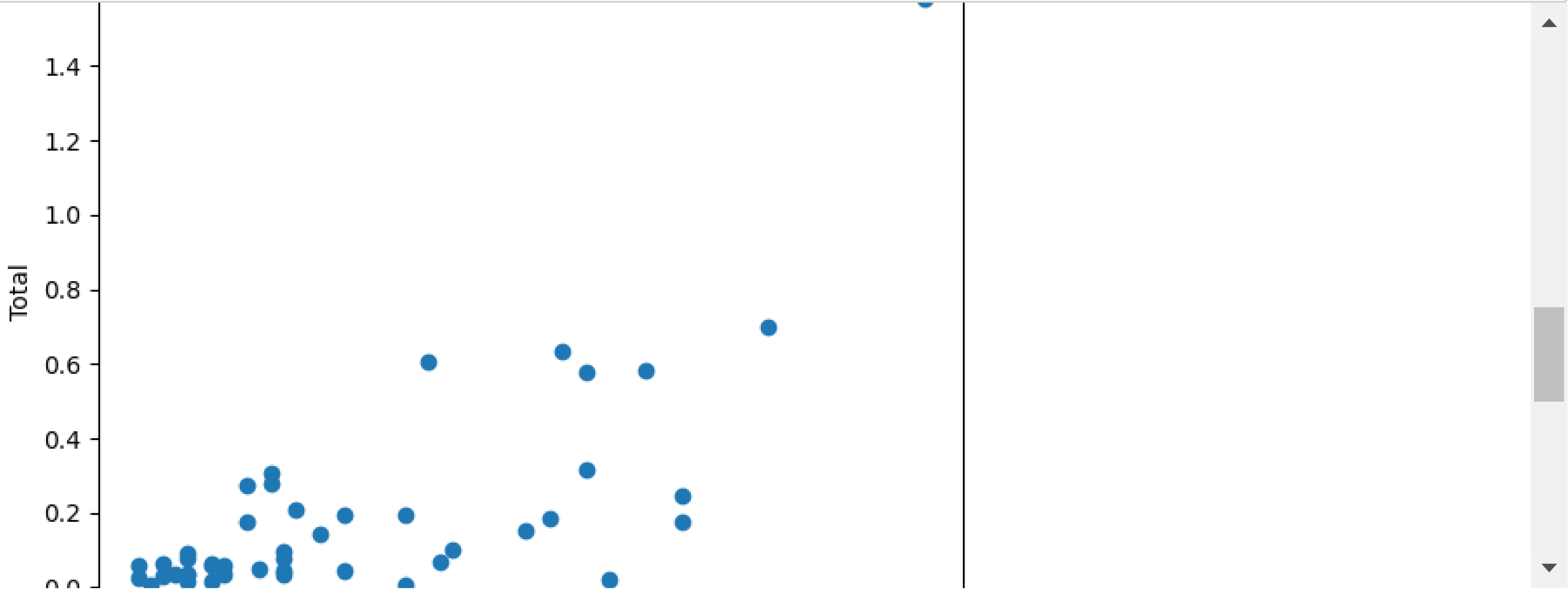
In [37]:
```
data.columns
```

Out[37]:
```
Index(['Brand', 'Total', 'Releases', '#1 Release', 'Lifetime Gross'], dtype='object')
```

In [40]:
```
plt.scatter((data["Total"]),data["Lifetime Gross"])
plt.show()
```



In [45]:
```
for i in data.columns[:-1]:
    plt.xlabel(i)
    plt.ylabel("Total")
    plt.scatter(data[i],data["Total"])
    plt.show()
```



In [47]:
```
import numpy as np
x = np.array(data["Releases"]).reshape(-1,1)
y = np.array(data["Total"]).reshape(-1,1)
```

In [49]:
```
from sklearn.linear_model import LinearRegression
```

In [50]:
```
linear = LinearRegression()
```

In [51]:
```
linear.fit(x,y)
```

Out[51]: LinearRegression()

In [52]:
```
linear.predict([[34]])
```
Out[52]:
```
array([[3.47997776e+09]])
```

multiple Linear Regression

In [70]:
```
len(data.columns)
```
Out[70]:
```
5
```

In [71]:
```
data.columns
```
Out[71]:
```
Index(['Brand', 'Total', 'Releases', '#1 Release', 'Lifetime Gross'], dtype='object')
```

In [73]:
```
x.head()
```
Out[73]:

	Total	Releases	#1 Release	Lifetime Gross
0	15806336901	69	Avengers: Endgame	858373000
1	7018798067	56	Jurassic World	652270625
2	6325022918	39	Star Wars: Episode VII - The Force Awakens	936662225
3	6078217662	28	Incredibles 2	608581744
4	5815645953	46	The Dark Knight	533345358

In [74]:
```
y.head()
```
Out[74]:
```
0          Marvel Comics
1     Legendary Pictures
2              Lucasfilm
3                  Pixar
4              DC Comics
Name: Brand, dtype: object
```

In [75]:

```
model = LinearRegression()
```

In [92]:

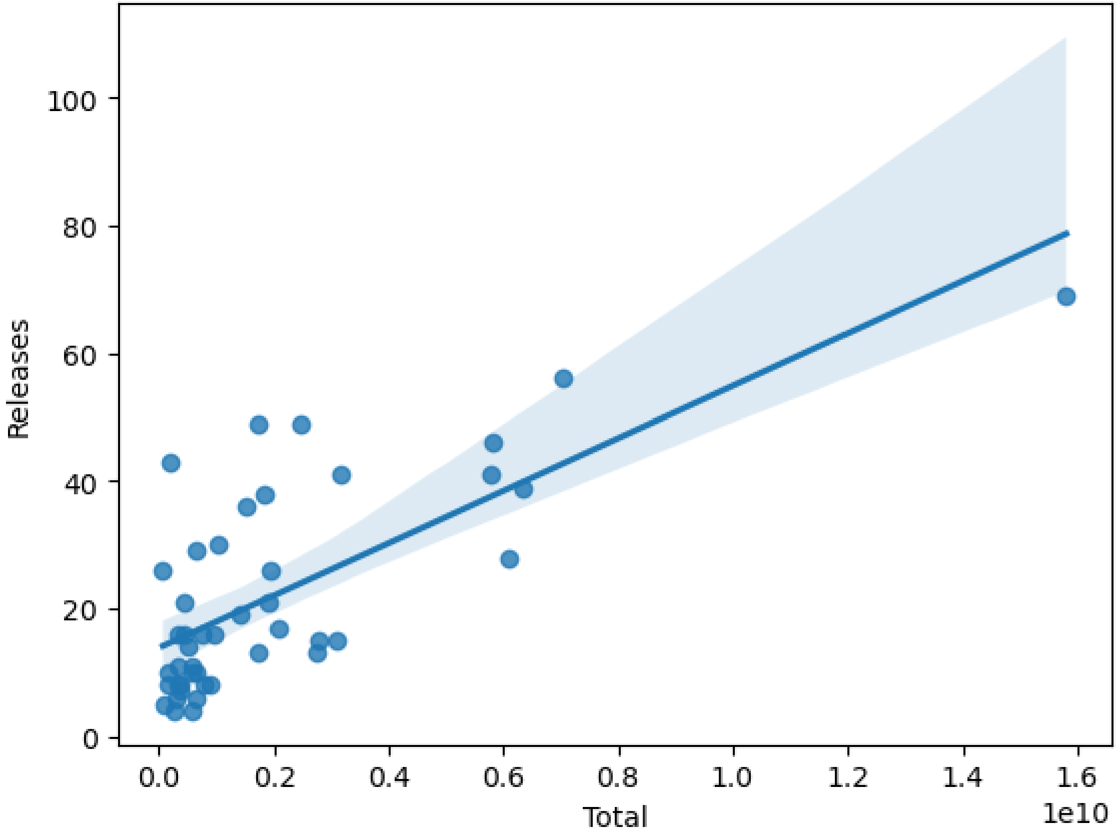
```
import seaborn as sns
data = pd.read_csv("bomojobrandindices.csv",sep = '\t')
```

In [99]:

```
sns.regplot(x = "Total",
            y = "Releases",
            data = data)
```

Out[99]:

```
<AxesSubplot:xlabel='Total', ylabel='Releases'>
```



split the data into testing and training

In [121]:
```
from sklearn.model_selection import train_test_split
```

In [124]:
```
xtrain,xtest,ytrain,ytest = train_test_split(data.drop("Total",axis=1),data["Total"],train_size = .75)
```

In [125]:
```
xtrain.shape
```

Out[125]:
```
(33, 4)
```

In [126]:
```
ytrain.shape
```

Out[126]:
```
(33,)
```

In [127]:
```
xtest.shape
```

Out[127]:
```
(12, 4)
```

In [128]:
```
ytest.shape
```

Out[128]:
```
(12,)
```

In [129]:
```
model.fit(xtrain,ytrain)
```

Out[129]:
```
LinearRegression()
```

In [113]:
```

# One hot encoding for the columns
data_onehot = pd.get_dummies(data, columns=['Brand', '#1 Release'])
```

In [114]:

```
# Import LabelEncoder
from sklearn.preprocessing import LabelEncoder

# Instantiate LabelEncoder
le = LabelEncoder()

# Apply le on categorical feature columns
data[['Brand', '#1 Release']] = data[['Brand', '#1 Release']].apply(lambda col: le.fit_transform(col))
```

In [116]:

```
data.sketch.howto("ValueError: could not convert string to float: 'Vertigo Entertainment'")
```

```
# Replace the string value with a numeric value
data['Brand'] = data['Brand'].replace('Vertigo Entertainment', 0)
```

Copy

In [117]:

```
# Replace the string value with a numeric value
data['Brand'] = data['Brand'].replace('Vertigo Entertainment', 0)
```

In [120]:
```
data
```

Out[120]:

	Brand	Total	Releases	#1 Release	Lifetime Gross
0	24	15806336901	69	2	858373000
1	21	7018798067	56	16	652270625
2	22	6325022918	39	26	936662225
3	30	6078217662	28	14	608581744
4	9	5815645953	46	31	533345358
5	15	5792217737	41	24	441226247
6	41	3154664176	41	15	327481748
7	3	3077078931	15	26	936662225
8	43	2774912904	15	9	477373578
9	18	2759505881	13	36	368384330
10	5	2451844676	49	10	176040665
11	17	2079485824	17	38	402111870
12	28	1930746182	26	28	191204754
13	36	1918763863	21	25	190241310
14	42	1842852183	38	30	291710957
15	4	1740428763	13	13	196573705
16	37	1729648950	49	15	327481748
17	23	1520456359	36	34	158119460
18	31	1422911399	19	28	191204754
19	35	1023596031	30	3	169106725
20	11	947671338	16	0	210614939
21	39	889955830	8	1	334191110
22	44	786497181	8	33	257760692
23	40	765635362	16	17	90508336
24	7	652824187	29	22	68947075
25	19	645661825	10	32	158348367

	Brand	Total	Releases	#1 Release	Lifetime Gross
26	33	645459186	6	29	227471070
27	25	580145113	4	11	200676069
28	16	578586146	10	23	153294164
29	27	574728259	11	35	81001787
30	29	495237720	14	19	132072926
31	10	454625110	16	39	63686397
32	26	436669213	21	20	120091123
33	12	359952780	8	12	90559416
34	32	356238494	7	5	206459076
35	34	346533876	11	41	121697323
36	13	337576791	8	21	90288712
37	0	333777829	16	6	106834564
38	20	300158323	6	7	75286229
39	1	276088604	4	40	107509799
40	2	204376048	43	18	47695371
41	8	171686009	10	4	61186570
42	14	151620585	8	8	32011576
43	6	73338237	5	27	30617396
44	38	70666453	26	37	19202743

In [130]:
```
model.fit(xtrain,ytrain)
```

Out[130]:
```
LinearRegression()
```

In [131]:
```
y_pred = model.predict(xtest)
```

In [132]:
```
ytest.head(10)
```

Out[132]:
```
3      6078217662
44       70666453
13     1918763863
41      171686009
15     1740428763
4      5815645953
19     1023596031
33      359952780
0     15806336901
32      436669213
Name: Total, dtype: int64
```

In [133]:
```
xtest.head()
```

Out[133]:

	Brand	Releases	#1 Release	Lifetime Gross
3	30	28	14	608581744
44	38	26	37	19202743
13	36	21	25	190241310
41	8	10	4	61186570
15	4	13	13	196573705

In [134]:
```
for i in y_pred[:10]:
    print(i)
```

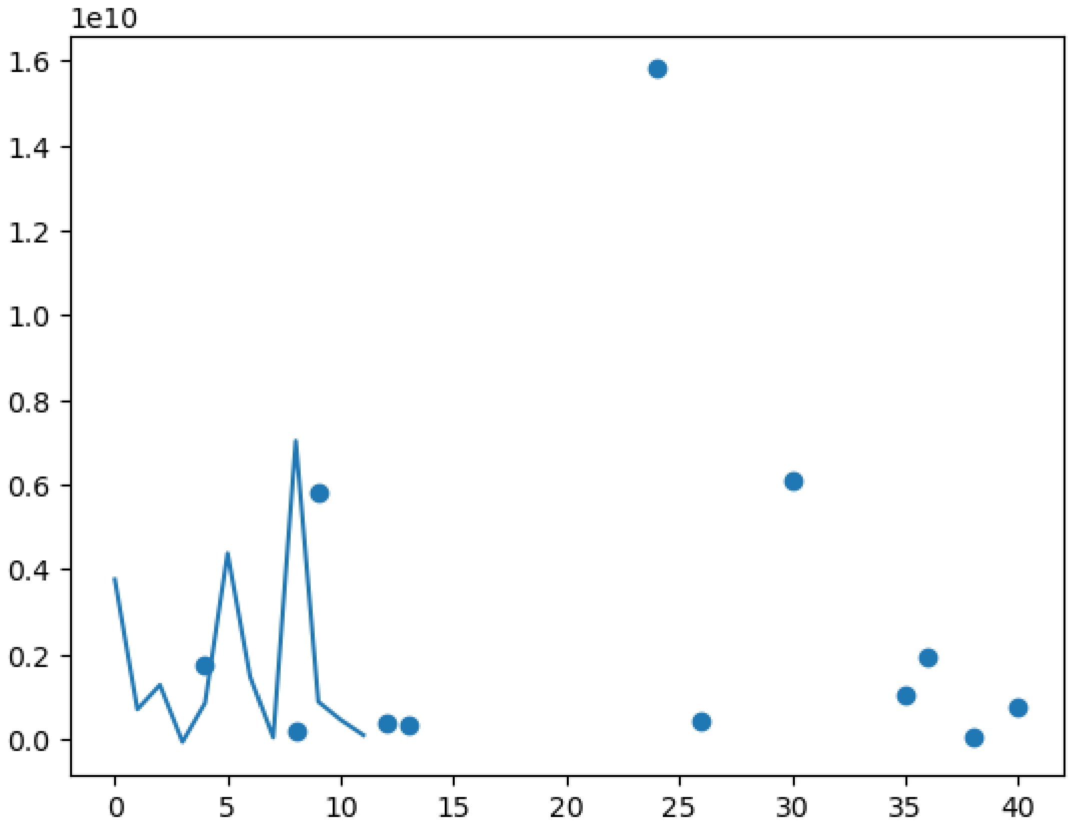
```
3759981527.9100475
703903546.8336082
1279417825.7516363
-66796223.29696846
854936214.8594639
4377559329.549724
1472205887.3786857
38449329.33133459
7032576530.735122
882981429.5428896
```

In [135]:
```
model.score(xtrain, ytrain )
```

Out[135]:
```
0.8177496414203069
```

In [141]:
```
plt.scatter(xtest["Brand"],ytest)
plt.plot(y_pred)
```

Out[141]: [<matplotlib.lines.Line2D at 0x12e0b00cb80>]



In [142]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error
```

In [143]:
```
mean_absolute_error(ytest, y_pred)
```

Out[143]: 1391561105.1962876

In [144]:
```
mean_squared_error(ytest, y_pred)
```
Out[144]:
```
7.227713975573045e+18
```
In [145]:
```
model.score(xtrain,ytrain) #r2 score
```
Out[145]:
```
0.8177496414203069
```
Polynomial Regression

In [146]:
```
import numpy as np
import matplotlib.pyplot as plt
```
In [147]:
```
from sklearn.preprocessing import PolynomialFeatures
```
In [150]:
```
lin = LinearRegression()
```
In [152]:
```
x_train = poly.fit_transform(x)
```
In [153]:
```
poly.fit(x_train,y)
```
Out[153]:
```
PolynomialFeatures()
```
In [155]:
```
lin = LinearRegression()
```
logistic regression:

In [159]:
```
from sklearn.linear_model import LogisticRegression
```

In [160]:
```python
log = LogisticRegression()
```

In [162]:
```python
import pandas as pd
```

In [163]:
```python
data = pd.read_csv("bomojobrandindices.csv")
```

In [164]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 1 columns):
 #   Column                                                          Non-Null Count  Dtype
---  ------                                                          --------------  -----
 0   Brand      Total   Releases        #1 Release      Lifetime Gross  45 non-null     object
dtypes: object(1)
memory usage: 488.0+ bytes
```

In [165]:
```python
data = pd.read_csv("bomojobrandindices.csv",sep = "\t")
```

In [166]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Brand           45 non-null     object
 1   Total           45 non-null     int64
 2   Releases        45 non-null     int64
 3   #1 Release      45 non-null     object
 4   Lifetime Gross  45 non-null     int64
dtypes: int64(3), object(2)
memory usage: 1.9+ KB
```

In [173]:
```python
data["Lifetime Gross"].unique()
```

Out[173]:
```
array([858373000, 652270625, 936662225, 608581744, 533345358, 441226247,
       327481748, 477373578, 368384330, 176040665, 402111870, 191204754,
       190241310, 291710957, 196573705, 158119460, 169106725, 210614939,
       334191110, 257760692,  90508336,  68947075, 158348367, 227471070,
       200676069, 153294164,  81001787, 132072926,  63686397, 120091123,
        90559416, 206459076, 121697323,  90288712, 106834564,  75286229,
       107509799,  47695371,  61186570,  32011576,  30617396,  19202743],
      dtype=int64)
```

In [174]:
```python
for i in data.columns:
    print(f"{i}        {data[i].dtype}")
```

```
Brand        object
Total        int64
Releases        int64
#1 Release        object
Lifetime Gross        int64
```

In [175]:
```python
data.isna().sum()
```

Out[175]:
```
Brand             0
Total             0
Releases          0
#1 Release        0
Lifetime Gross    0
dtype: int64
```

In [176]:
```python
for column in data.columns:
    if data[column].dtype == "object":
        print(column)
```

```
Brand
#1 Release
```

In [177]:
```
le = LabelEncoder()
for column in data.columns:
    if data[column].dtype == "object":
        data[column] = le.fit_transform(data[column])
```

In [178]:
```
data.head()
```

Out[178]:

	Brand	Total	Releases	#1 Release	Lifetime Gross
0	24	15806336901	69	2	858373000
1	21	7018798067	56	16	652270625
2	22	6325022918	39	26	936662225
3	30	6078217662	28	14	608581744
4	9	5815645953	46	31	533345358

In [179]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Brand           45 non-null     int32
 1   Total           45 non-null     int64
 2   Releases        45 non-null     int64
 3   #1 Release      45 non-null     int32
 4   Lifetime Gross  45 non-null     int64
dtypes: int32(2), int64(3)
memory usage: 1.5 KB
```

In [180]:
```
from sklearn.linear_model import LogisticRegression
```

In [181]:
```
log = LogisticRegression()
```

In [182]:

```
x = data.drop("#1 Release",axis=1)
y = data["#1 Release"]
```

In [183]:

```
x
```

Out[183]:

	Brand	Total	Releases	Lifetime Gross
0	24	15806336901	69	858373000
1	21	7018798067	56	652270625
2	22	6325022918	39	936662225
3	30	6078217662	28	608581744
4	9	5815645953	46	533345358
5	15	5792217737	41	441226247
6	41	3154664176	41	327481748
7	3	3077078931	15	936662225
8	43	2774912904	15	477373578
9	18	2759505881	13	368384330
10	5	2451844676	49	176040665
11	17	2079485824	17	402111870
12	28	1930746182	26	191204754
13	36	1918763863	21	190241310
14	42	1842852183	38	291710957
15	4	1740428763	13	196573705
16	37	1729648950	49	327481748
17	23	1520456359	36	158119460
18	31	1422911399	19	191204754
19	35	1023596031	30	169106725
20	11	947671338	16	210614939
21	39	889955830	8	334191110
22	44	786497181	8	257760692
23	40	765635362	16	90508336
24	7	652824187	29	68947075
25	19	645661825	10	158348367

	Brand	Total	Releases	Lifetime Gross
26	33	645459186	6	227471070
27	25	580145113	4	200676069
28	16	578586146	10	153294164
29	27	574728259	11	81001787
30	29	495237720	14	132072926
31	10	454625110	16	63686397
32	26	436669213	21	120091123
33	12	359952780	8	90559416
34	32	356238494	7	206459076
35	34	346533876	11	121697323
36	13	337576791	8	90288712
37	0	333777829	16	106834564
38	20	300158323	6	75286229
39	1	276088604	4	107509799
40	2	204376048	43	47695371
41	8	171686009	10	61186570
42	14	151620585	8	32011576
43	6	73338237	5	30617396
44	38	70666453	26	19202743

In [184]:

```
y
```

Out[184]:
```
0      2
1     16
2     26
3     14
4     31
5     24
6     15
7     26
8      9
9     36
10    10
11    38
12    28
13    25
14    30
15    13
16    15
17    34
18    28
19     3
20     0
21     1
22    33
23    17
24    22
25    32
26    29
27    11
28    23
29    35
30    19
31    39
32    20
33    12
34     5
35    41
36    21
37     6
38     7
39    40
40    18
```

```
41     4
42     8
43    27
44    37
Name: #1 Release, dtype: int32
```

In [185]:
```
log.fit(x,y)
```

```
C:\Users\munep\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed t
o converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/m
odules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[185]:
```
LogisticRegression()
```

In [186]:
```
log.intercept_
```

Out[186]:
```
array([ 2.86285554e-17, -1.85914091e-16, -1.39181778e-15,  8.32407369e-17,
        8.58631258e-16,  4.45867324e-16,  5.76586802e-16,  6.56153836e-16,
        9.19305777e-16, -9.44390716e-16, -2.41915921e-16,  2.34424915e-16,
        5.73761593e-16, -1.50758715e-16, -1.32956516e-15, -1.58356233e-16,
       -1.44823536e-15,  3.04579637e-16,  8.16519469e-16,  4.09619835e-16,
        4.82188874e-16,  5.93281141e-16,  3.77671048e-16,  3.01878487e-16,
       -9.43093054e-16, -1.73660675e-16, -3.05455765e-15,  1.05629418e-15,
        5.73632089e-16,  1.49254454e-16, -3.78566823e-16, -1.15632182e-15,
        2.48736620e-16,  2.37871516e-17, -3.32495489e-17,  4.11783452e-16,
       -6.55909867e-16,  1.06091036e-15, -6.67713766e-16,  5.36489387e-16,
        6.45832090e-16,  5.44968108e-16])
```

In [187]:

```
log.coef_
```

```
Out[187]: array([[-1.37451387e-14, -4.27565812e-09, -1.22503283e-15,
          4.77516077e-08],
        [ 1.54403560e-14, -4.22568594e-08, -1.10156011e-14,
          1.58297380e-07],
        [-2.77565405e-14,  2.68044808e-08, -2.82443164e-14,
         -2.59151363e-07],
        [ 1.84203299e-14,  4.26109300e-09,  1.62829232e-14,
          1.58170000e-09],
        [ 2.11955779e-15, -1.23251079e-09,  6.50137180e-15,
          2.89592603e-08],
        [ 2.31560063e-14, -3.86034575e-08, -2.57614029e-15,
          1.43904027e-07],
        [-1.44204536e-14, -4.41500921e-09,  9.83508234e-15,
          4.38612941e-08],
        [ 1.31374216e-14,  1.09575809e-09, -2.02122511e-15,
          1.62488557e-08],
        [ 1.10700274e-14,  2.25647880e-09,  4.70400558e-15,
         -3.74222515e-10],
        [ 3.73928229e-15, -5.41147036e-10, -2.56453549e-14,
          3.42568848e-08],
        [-2.65134460e-14,  1.96890880e-08,  2.52135266e-14,
         -1.46119819e-07],
        [ 9.32996985e-15, -1.83039926e-08, -1.07697920e-14,
          9.23072693e-08],
        [ 9.10462009e-16,  6.88237880e-10, -8.68060303e-16,
          2.14492248e-08],
        [-2.69961390e-14,  1.10324343e-08, -1.41743138e-14,
         -5.33939481e-08],
        [-1.97572627e-14,  1.31786969e-08, -3.82723649e-14,
         -7.21600381e-08],
        [ 3.61439698e-14,  4.98082200e-09,  4.99283010e-14,
          2.62437452e-10],
        [-3.29912443e-14,  1.56806940e-08, -9.87434156e-15,
         -9.84728320e-08],
        [ 3.03047126e-14,  9.74167937e-09,  2.62299326e-15,
         -4.30630174e-08],
        [-6.52688628e-15,  2.20784317e-09,  4.82064518e-14,
          5.65056270e-09],
        [ 1.88440988e-14, -4.72136905e-09,  3.99009336e-15,
          4.13181108e-08],
        [ 1.67281468e-14, -3.36708113e-09,  1.43095932e-14,
```

```
  4.00211425e-08],
[ 2.68161848e-15, -1.48679304e-10, -4.36179740e-16,
  2.51823806e-08],
[-1.04299433e-14,  9.84359155e-09,  2.10906461e-14,
 -5.10945678e-08],
[-4.81455733e-16, -4.51794847e-09, -2.99196969e-15,
  4.78797014e-08],
[-2.78516663e-14,  2.08819125e-08, -1.50468659e-14,
 -1.59284754e-07],
[ 1.39350677e-14,  1.35185002e-08, -5.66203867e-15,
 -7.84753691e-08],
[-9.55712233e-14, -1.80970647e-10, -3.96928108e-14,
  3.49634789e-08],
[ 3.40154561e-15, -1.08132358e-08,  3.01142586e-15,
  1.50809019e-08],
[ 3.06047297e-14,  1.00631858e-08,  9.22893142e-15,
 -4.22710883e-08],
[ 1.73654805e-14, -2.32637457e-08, -9.37872508e-15,
  1.05854718e-07],
[ 1.59062932e-14,  2.73617599e-09,  1.52154479e-14,
  1.36807108e-08],
[-4.18169197e-14,  1.62778925e-08, -1.17579186e-14,
 -1.05314706e-07],
[ 2.05754506e-15, -3.29656670e-09, -4.22695212e-15,
  4.20375773e-08],
[ 2.82957545e-14, -2.43227845e-08, -9.05288864e-15,
  1.10082436e-07],
[ 3.60059418e-16,  1.23262042e-08,  1.84170910e-14,
 -6.63084428e-08],
[ 1.61946236e-14,  7.68558725e-09, -9.20136980e-16,
 -2.59275846e-08],
[-2.15006522e-14,  6.29201452e-09, -2.68411440e-14,
 -1.00143741e-08],
[ 4.47037283e-14, -1.27800771e-08,  3.01288104e-14,
  7.88374997e-10],
[-2.28561422e-14, -5.06110869e-09, -1.47843719e-14,
  5.07064474e-08],
[-2.74441810e-15,  7.14421063e-09,  8.02931210e-15,
 -2.42334692e-08],
[-1.15303316e-14, -8.27673868e-09, -4.28102685e-15,
  5.63297131e-08],
```

```
[ 2.86390763e-14, -8.00764112e-09,  3.04356538e-15,
  5.72033971e-08]])
```