```
import warnings
warnings.filterwarnings('ignore')


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


df = pd.read_csv('lego_sets.csv')


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12261 entries, 0 to 12260
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   ages               12261 non-null  object 
 1   list_price         12261 non-null  float64
 2   num_reviews        10641 non-null  float64
 3   piece_count        12261 non-null  float64
 4   play_star_rating   10486 non-null  float64
 5   prod_desc          11884 non-null  object 
 6   prod_id            12261 non-null  float64
 7   prod_long_desc     12261 non-null  object 
 8   review_difficulty  10206 non-null  object 
 9   set_name           12261 non-null  object 
 10  star_rating        10641 non-null  float64
 11  theme_name         12258 non-null  object 
 12  val_star_rating    10466 non-null  float64
 13  country            12261 non-null  object 
dtypes: float64(7), object(7)
memory usage: 1.3+ MB
```

```
df.head(20)
```

	ages	list_price	num_reviews	piece_count	play_star_rating	prod_desc	prod_id
0	6-12	29.99	2.0	277.0	4.0	Catapult into action and take back the eggs fr...	75823.0
1	6-12	19.99	2.0	168.0	4.0	Launch a flying attack and rescue the eggs fro...	75822.0
2	6-12	12.99	11.0	74.0	4.3	Chase the piggy with lightning-fast Chuck and ...	75821.0
3	12+	99.99	23.0	1032.0	3.6	Explore the architecture of the United States ...	21030.0
4	12+	79.99	14.0	744.0	3.2	Recreate the Solomon R. Guggenheim Museum® wit...	21035.0
5	12+	59.99	7.0	597.0	3.7	Celebrate Shanghai with this LEGO® Architectur...	21039.0
6	12+	59.99	37.0	598.0	3.7	Celebrate New York City with this LEGO® Archit...	21028.0
7	12+	49.99	24.0	780.0	4.4	Recreate Buckingham Palace with LEGO® Architec...	21029.0
8	12+	39.99	23.0	468.0	3.6	Celebrate London with this LEGO® Architecture ...	21034.0
9	12+	39.99	11.0	444.0	3.6	Celebrate Chicago with this LEGO® Architecture...	21033.0

Experience

10	12+	39.99	14.0	386.0	4.1	the grandeur of the Arc de Triomphe!	21036.0
11	12+	34.99	53.0	321.0	3.2	Build your own LEGO® interpretation of the ico...	21019.0
12	12+	29.99	7.0	361.0	4.2	Celebrate Sydney with this LEGO® Architecture ...	21032.0
13	7-12	159.99	63.0	847.0	3.8	Bring your LEGO® creations to life!	17101.0
14	10+	29.99	13.0	708.0	4.7	Build a LEGO® BrickHeadz version of yourself!	41597.0
15	10+	19.99	1.0	234.0	3.0	Train a raptor with LEGO® BrickHeadz™ Owen and...	41614.0
16	10+	19.99	1.0	160.0	5.0	Join Mr. Incredible and Frozone for LEGO® Bric...	41613.0
17	10+	9.99	1.0	149.0	2.0	Growl like a Wookiee with a LEGO® BrickHeadz™	41609.0

```
df['review_difficulty'].unique()
```

```
array(['Average', 'Easy', 'Challenging', 'Very Easy', nan,
       'Very Challenging'], dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['ages'] = le.fit_transform(df['ages'])
```

```
df['ages'] = df['ages'].astype(float)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['country'] = le.fit_transform(df['country'])
```

```
df['country'] = le.fit_transform(df['country'])
```

```
df['country'] = df['country'].astype(float)
```

```
df['review_difficulty'].value_counts()
```

```
Easy               4236
Average            3765
Very Easy          1139
Challenging        1058
Very Challenging      8
Name: review_difficulty, dtype: int64
```

```
df['review_difficulty'].replace(np.nan,'Easy',inplace=True)
```

```
df['review_difficulty'].unique()
```

```
array(['Average', 'Easy', 'Challenging', 'Very Easy', 'Very Challenging'],
      dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['review_difficulty'] = le.fit_transform(df['review_difficulty'])
```

```
df['review_difficulty'] = df['review_difficulty'].astype(float)
```

```
df['theme_name'].unique()
```

```
array(['Angry Birds™', 'Architecture', 'BOOST', 'BrickHeadz', 'City',
       'Juniors', 'Classic', 'Creator 3-in-1', 'Creator Expert',
       'THE LEGO® BATMAN MOVIE', 'DC Comics™ Super Heroes', 'DIMENSIONS™',
       'DC Super Hero Girls', 'Disney™', 'DUPLO®', 'Elves', 'Friends',
       'Ghostbusters™', 'Ideas', 'Indoraptor Rampage at Lockwood Estate',
       'Carnotaurus Gyrosphere Escape', 'T. rex Transport',
       'Jurassic Park Velociraptor Chase', 'Dilophosaurus Outpost Attack',
       "Blue's Helicopter Pursuit", 'Stygimoloch Breakout',
       'Pteranodon Chase', 'Marvel Super Heroes', 'MINDSTORMS®',
       'Minecraft™', 'Minifigures', 'NEXO KNIGHTS™',
       'THE LEGO® NINJAGO® MOVIE™', 'NINJAGO®', 'SERIOUS PLAY®',
       'Speed Champions', 'Star Wars™', 'Technic', 'Power Functions',
       'LEGO® Creator 3-in-1', nan], dtype=object)
```

```
df['theme_name'].replace(np.nan,"Star Wars™",inplace=True)
```

```
df['theme_name'].unique()
```

```
array(['Angry Birds™', 'Architecture', 'BOOST', 'BrickHeadz', 'City',
       'Juniors', 'Classic', 'Creator 3-in-1', 'Creator Expert',
```

```
       'THE LEGO® BATMAN MOVIE', 'DC Comics™ Super Heroes', 'DIMENSIONS™',
       'DC Super Hero Girls', 'Disney™', 'DUPLO®', 'Elves', 'Friends',
       'Ghostbusters™', 'Ideas', 'Indoraptor Rampage at Lockwood Estate',
       'Carnotaurus Gyrosphere Escape', 'T. rex Transport',
       'Jurassic Park Velociraptor Chase', 'Dilophosaurus Outpost Attack',
       "Blue's Helicopter Pursuit", 'Stygimoloch Breakout',
       'Pteranodon Chase', 'Marvel Super Heroes', 'MINDSTORMS®',
       'Minecraft™', 'Minifigures', 'NEXO KNIGHTS™',
       'THE LEGO® NINJAGO® MOVIE™', 'NINJAGO®', 'SERIOUS PLAY®',
       'Speed Champions', 'Star Wars™', 'Technic', 'Power Functions',
       'LEGO® Creator 3-in-1'], dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['theme_name'] = le.fit_transform(df['theme_name'])
df['theme_name'] = df['theme_name'].astype(float)
```

```
df.info()
```

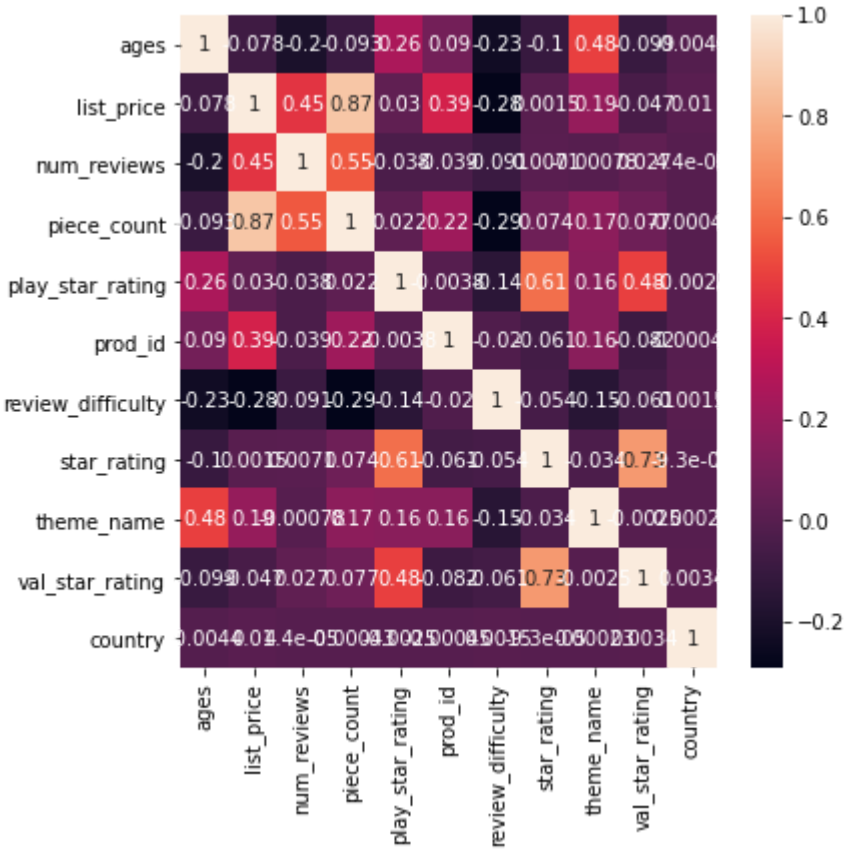
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12261 entries, 0 to 12260
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   ages               12261 non-null  float64
 1   list_price         12261 non-null  float64
 2   num_reviews        10641 non-null  float64
 3   piece_count        12261 non-null  float64
 4   play_star_rating   10486 non-null  float64
 5   prod_desc          11884 non-null  object 
 6   prod_id            12261 non-null  float64
 7   prod_long_desc     12261 non-null  object 
 8   review_difficulty  12261 non-null  float64
 9   set_name           12261 non-null  object 
 10  star_rating        10641 non-null  float64
 11  theme_name         12261 non-null  float64
 12  val_star_rating    10466 non-null  float64
 13  country            12261 non-null  float64
dtypes: float64(11), object(3)
memory usage: 1.3+ MB
```

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

```
df.corr()
```

	ages	list_price	num_reviews	piece_count	play_star_rating	prod
ages	1.000000	-0.077782	-0.195753	-0.092585	0.257059	0.089
list_price	-0.077782	1.000000	0.450785	0.869630	0.030027	0.388
num_reviews	-0.195753	0.450785	1.000000	0.546618	-0.037705	-0.039
piece_count	-0.092585	0.869630	0.546618	1.000000	0.022386	0.217
play_star_rating	0.257059	0.030027	-0.037705	0.022386	1.000000	-0.003
prod_id	0.089696	0.388633	-0.039141	0.217716	-0.003823	1.000
review_difficulty	-0.232085	-0.283644	-0.091069	-0.291459	-0.141154	-0.019
star_rating	-0.102985	0.001544	0.007111	0.073903	0.608193	-0.060

```
plt.figure(figsize=(6,6))
sns.heatmap(df.corr(), annot=True)
plt.show()
```




```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
x = df[['ages', 'theme_name', 'country', 'review_difficulty','piece_count']]
x_train,x_test,y_train,y_test=train_test_split(x,df[['list_price']],test_size=0.3)
```

```
Lr=LinearRegression()
Lr.fit(x_train,y_train)
y_hat=Lr.predict(x_test)
```

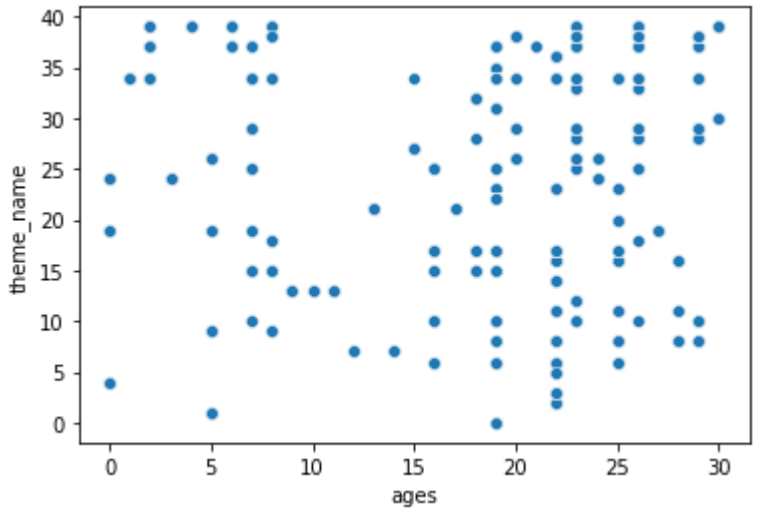
```
y_hat
```

```
array([[60.74714852],
       [23.27575593],
       [99.52928573],
       ...,
       [56.41569416],
       [52.52220425],
       [55.65490491]])
```

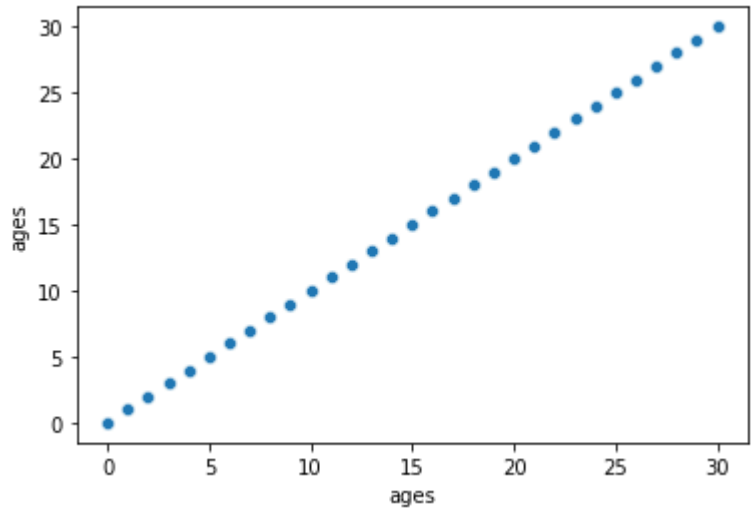
```
r2_score(y_test,y_hat)
```

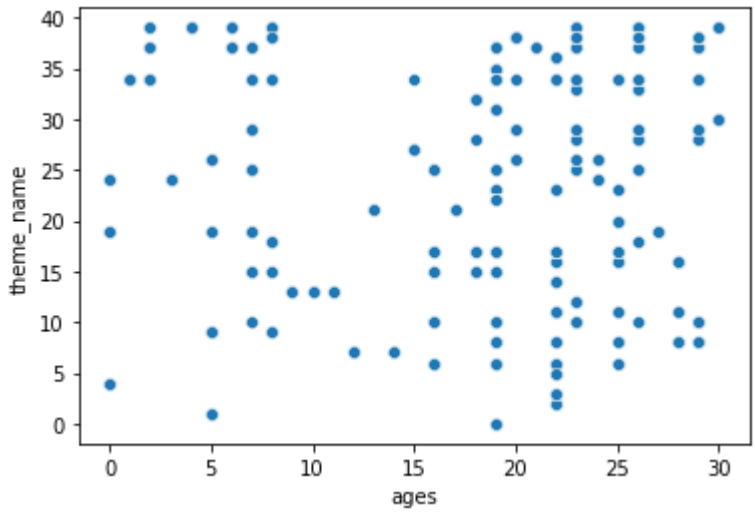
```
0.7811803081396985
```

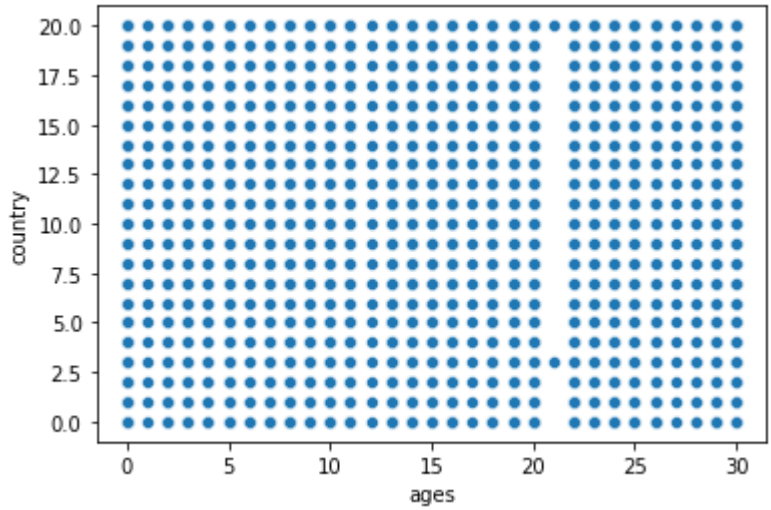
```
#condition 2
sns.scatterplot(data = df , x = "ages", y = "theme_name")
plt.show()
```



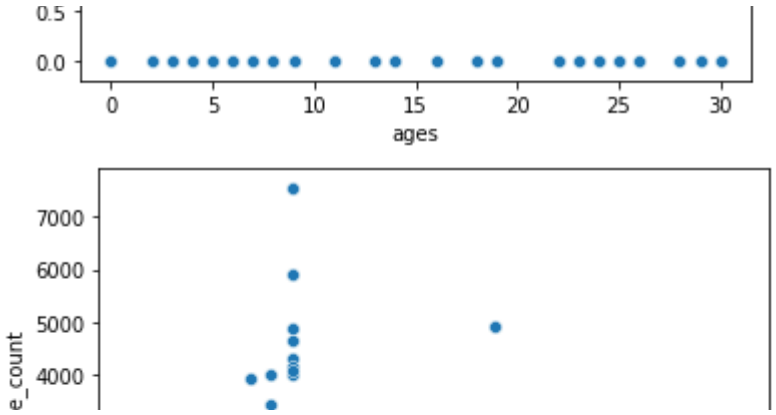

```
for i in x:
  sns.scatterplot(data = df , x = "ages", y = i)
  plt.show()
```


ages
ages
0
5
10
15
20
25
30

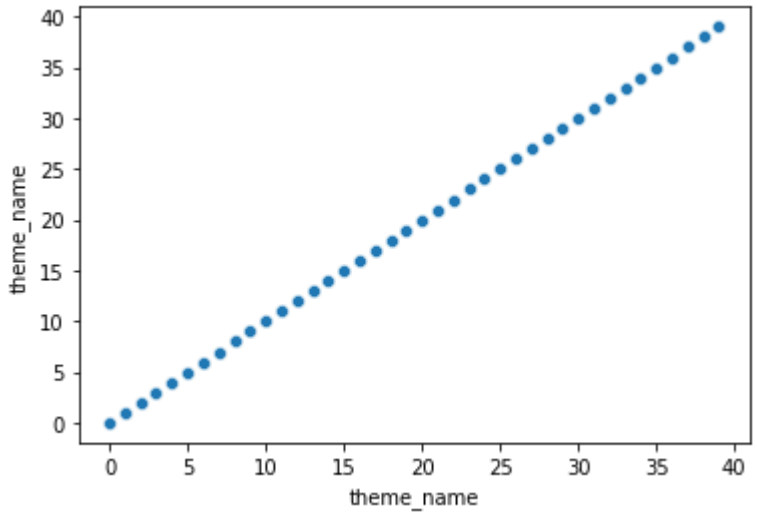

theme_name
ages
0
5
10
15
20
25
30
35
40

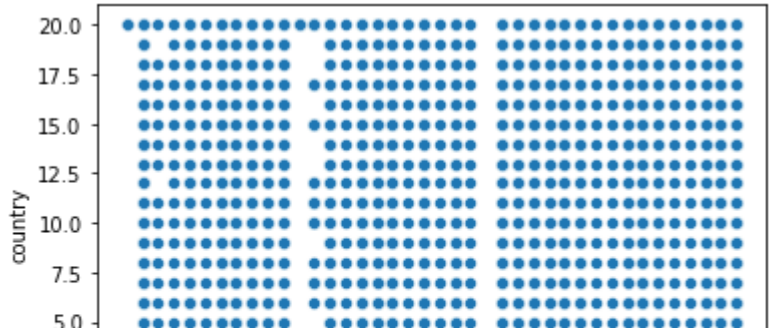

country
ages
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
5
10
15
20
25
30


review_difficulty
1.0
1.5
2.0
2.5
3.0
3.5
4.0

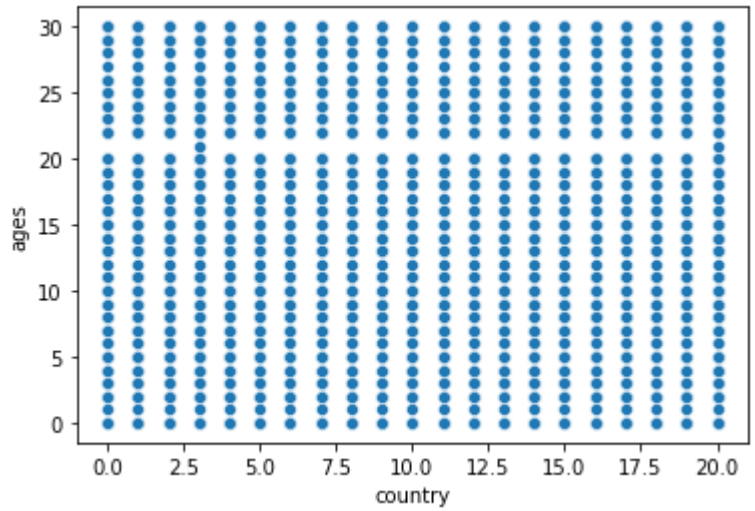


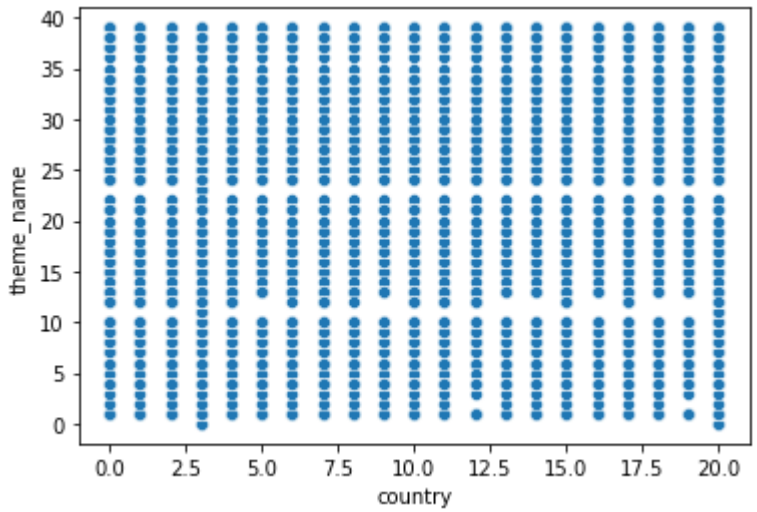

```
for i in x:
  if i == "ages":
    pass
  else:
    sns.scatterplot(data = df , x = "theme_name", y = i)
    plt.show()
```

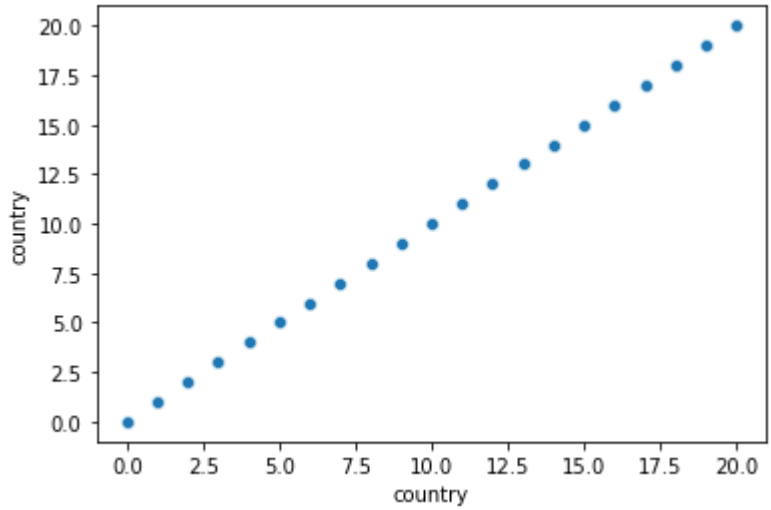




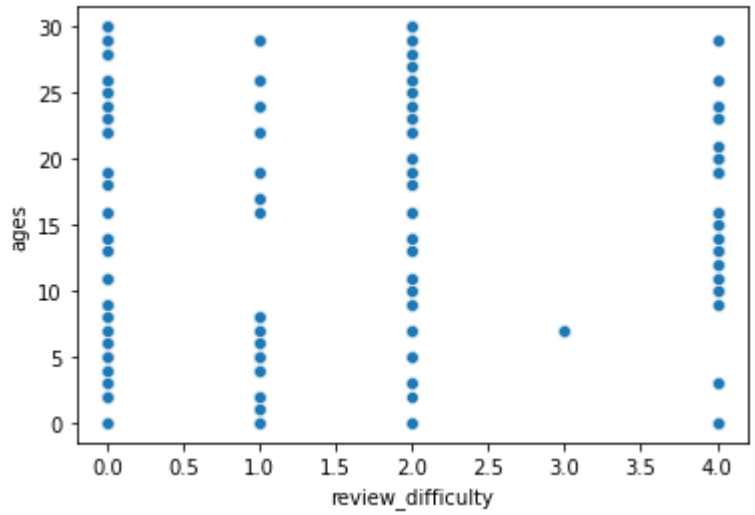

```
for i in x:
  sns.scatterplot(data = df , x = "country", y = i)
  plt.show()
```

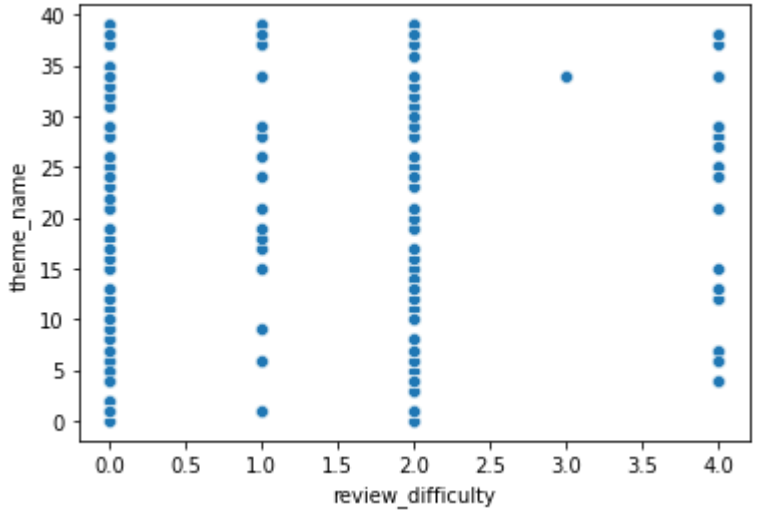


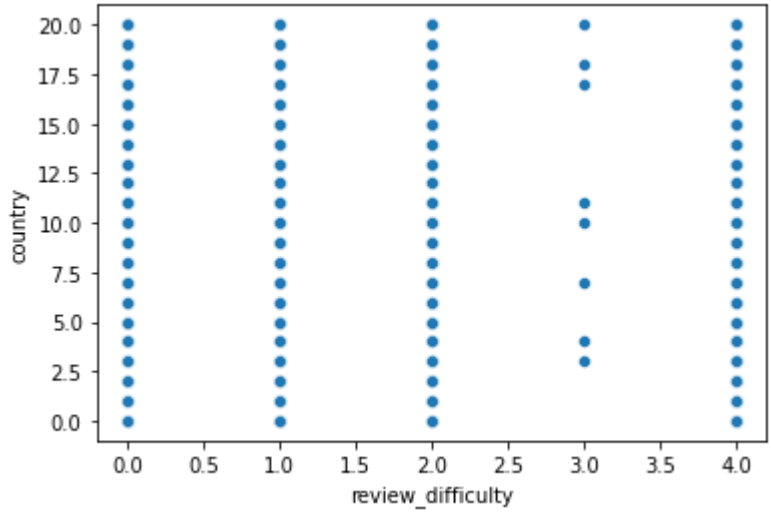




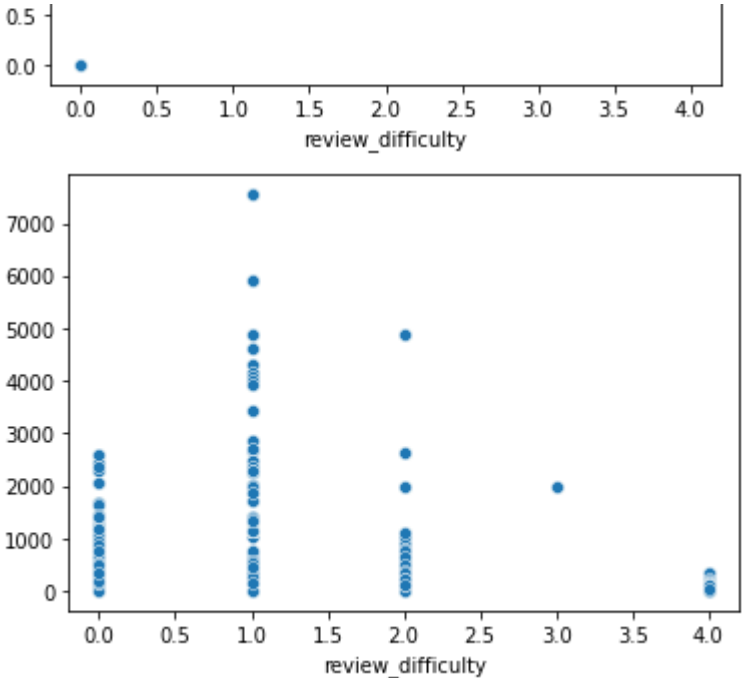
```
for i in x:
  sns.scatterplot(data = df , x = "review_difficulty", y = i)
  plt.show()
```


ages
review_difficulty

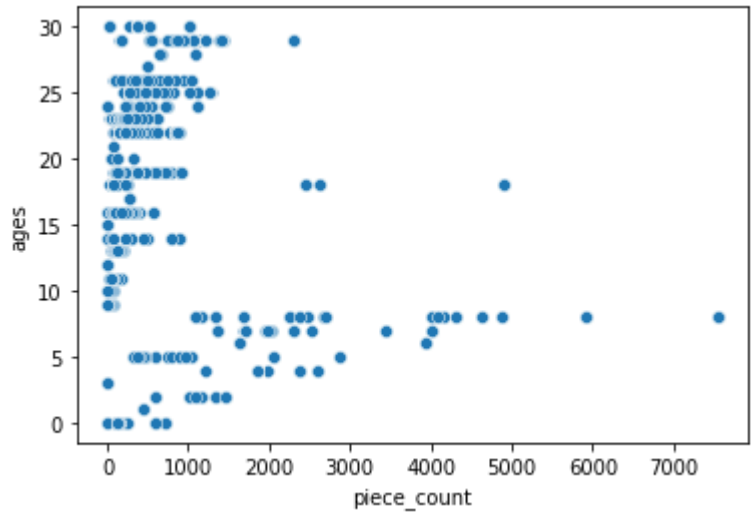

theme_name
review_difficulty

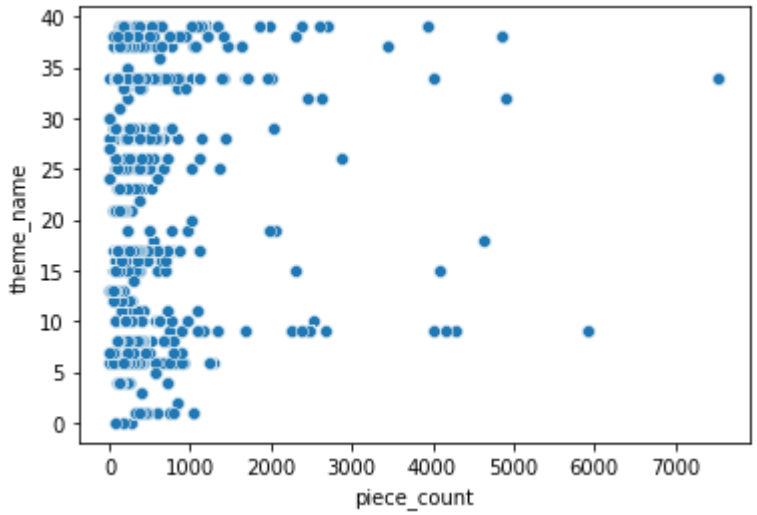

country
review_difficulty

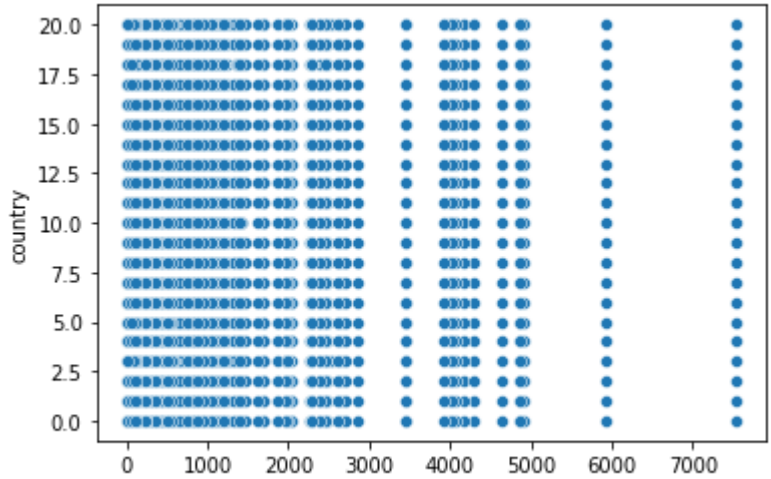

review_difficulty




```
for i in x:
  sns.scatterplot(data = df , x = "piece_count", y = i)
  plt.show()
```





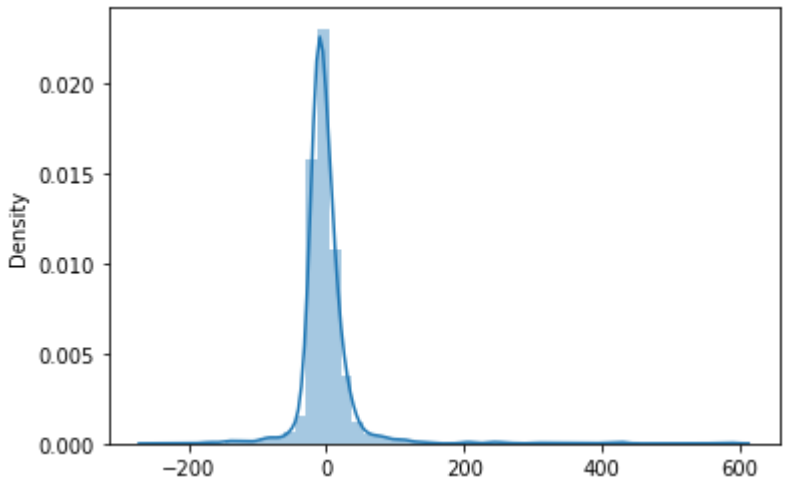



```
#by checking all the graphs for multicolinearity we can conclude that
#for feautres ages,theme_name, country, review_difficulty,piece_count have no multi colineari
```

```
#condition 3
residuals = y_test-y_hat
```

```
from scipy.stats import skew
sns.distplot(residuals)
plt.show()
```

```
print("Skew = ", skew(residuals))
```




```
Skew =  [5.64914979]
```

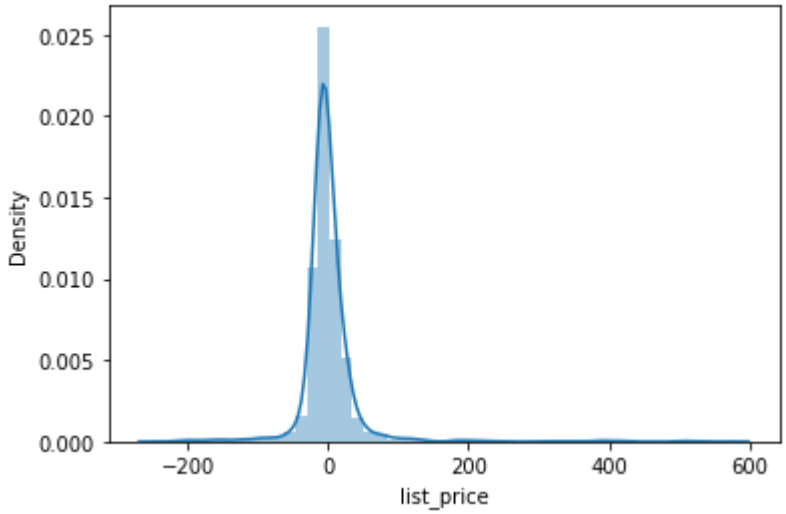
```
from sklearn.preprocessing import PolynomialFeatures

x = df[['ages', 'theme_name', 'country', 'review_difficulty','piece_count']]


pf = PolynomialFeatures()
x_poly = pf.fit_transform(x)



x_poly
```

```
array([[1.0000e+00, 1.9000e+01, 0.0000e+00, ..., 0.0000e+00, 0.0000e+00,
        7.6729e+04],
       [1.0000e+00, 1.9000e+01, 0.0000e+00, ..., 4.0000e+00, 3.3600e+02,
        2.8224e+04],
       [1.0000e+00, 1.9000e+01, 0.0000e+00, ..., 4.0000e+00, 1.4800e+02,
        5.4760e+03],
       ...,
       [1.0000e+00, 2.3000e+01, 3.8000e+01, ..., 4.0000e+00, 4.6600e+02,
        5.4289e+04],
       [1.0000e+00, 2.0000e+01, 3.8000e+01, ..., 1.6000e+01, 1.9200e+02,
        2.3040e+03],
       [1.0000e+00, 2.0000e+01, 3.8000e+01, ..., 4.0000e+00, 2.1800e+02,
        1.1881e+04]])
```

```
x_train , x_test , y_train , y_test = train_test_split(x_poly, df['list_price'],random_state
lr = LinearRegression()
lr.fit(x_train, y_train)
print('coef' , lr.coef_)
print('intercept' , lr.intercept_)
y_hat = lr.predict(x_test)
print("r2 = ", r2_score(y_test, y_hat))
print(lr.score(x_test, y_test))
```

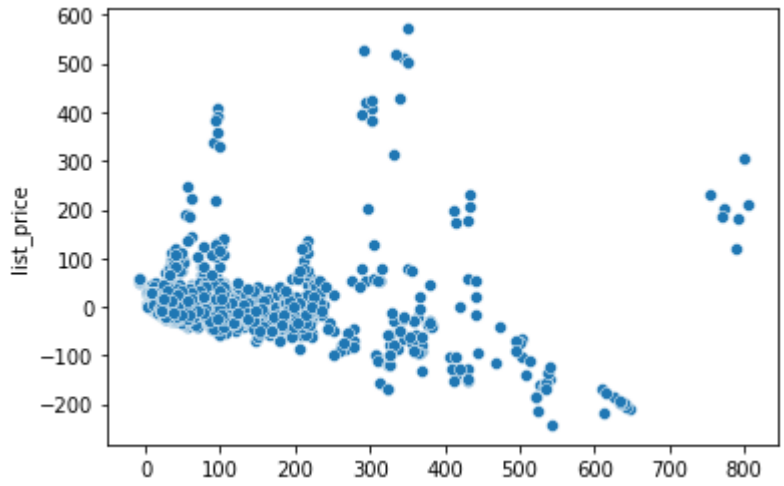
```
coef [ 0.00000000e+00 -6.47163174e-01  2.73705853e+00  1.18979105e+00
 -1.58784471e+01  3.92966780e-02 -9.15869493e-03 -5.32787761e-02
  5.72999519e-03  2.43433387e-01  2.30763326e-03 -4.53387601e-02
 -4.89935409e-03  1.43872540e-01  8.41549068e-04 -6.14275309e-02
  2.48453794e-02  3.99333342e-04  1.55209243e+00  6.69052819e-03
  2.53201779e-07]
intercept 21.849728565241513
r2 =  0.768101398555185
0.768101398555185
```

```
residuals = y_test-y_hat
sns.distplot(residuals)
plt.show()
print("Skew = ", skew(residuals))
```




```
Skew =  4.7576483520527235
```

```
sns.scatterplot(x = y_hat, y = residuals)
plt.show()
```




```
print("Bias = ",lr.score(x_train,y_train))
print("Variance = ", lr.score(x_test,y_test))
```

```
Bias =  0.8000298342200844
Variance =  0.768101398555185
```

```
from sklearn.linear_model import Ridge, Lasso


x = df[['ages', 'theme_name', 'country', 'review_difficulty','piece_count']]
x_train,x_test,y_train,y_test=train_test_split(x,df['list_price'],test_size=0.3)
l2 = Ridge(alpha=0.2)
l2.fit(x_train,y_train)
print("Bias = ",l2.score(x_train,y_train))
print("Variance = ", l2.score(x_test,y_test))
```

```
Bias =  0.740207250776957
Variance =  0.8088424993109439
```

```
x_train,x_test,y_train,y_test=train_test_split(x,df['list_price'],test_size=0.3)
for i in range(50):
  print(f"{i}")
  r = Ridge(i)
  r.fit(x_train,y_train)
  print("Bias = ",r.score(x_train,y_train))
  print("Variance = ", r.score(x_test,y_test))
  print("**************************************")
```

```
**************************************
9
Bias =  0.7445941643963018
Variance =  0.7897019176083682
**************************************
10
Bias =  0.7445941640452549
Variance =  0.7897021577071291
**************************************
11
Bias =  0.744594163657369
Variance =  0.7897023977276757
**************************************
12
Bias =  0.7445941632326656
Variance =  0.7897026376700403
**************************************
13
Bias =  0.7445941627711663
Variance =  0.7897028775342555
**************************************
14
Bias =  0.7445941622728928
Variance =  0.7897031173203531
**************************************
15
Bias =  0.7445941617378664
Variance =  0.7897033570283657
```

```
**************************************
16
Bias =  0.7445941611661087
Variance =  0.7897035966583255
**************************************
17
Bias =  0.744594160557641
Variance =  0.7897038362102646
**************************************
18
Bias =  0.7445941599124852
Variance =  0.7897040756842154
**************************************
19
Bias =  0.7445941592306624
Variance =  0.7897043150802099
**************************************
20
Bias =  0.7445941585121941
Variance =  0.7897045543982806
**************************************
21
Bias =  0.7445941577571019
Variance =  0.7897047936384592
**************************************
22
Bias =  0.7445941569654071
Variance =  0.7897050328007784
**************************************
23
Bias =  0.7445941561371312
```

```
l1=Lasso(100)
l1.fit(x_train, y_train)
l1.score(x_train, y_train)
```

```
0.7389357451610967
```

```
l1.score(x_test,y_test)
```

```
0.7895366773076686
```

```
for j in range(100, 1000,100):
  l1 = Lasso(j)
  l1.fit(x_train,y_train)
  print(j,l1.score(x_test,y_test))
```

```
100 0.7895366773076686
200 0.7893771259457174
300 0.7892140538383403
400 0.7890474609855376
500 0.7888773473873089
600 0.7887037130436545
700 0.7885265579545744
```

```
800 0.7883458821200684
900 0.7881616855401365
```

```
#This case study are Linear Regression and Regularisation r2 Score are 0.75, in that used the
```