Question 1

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('popularity.csv')
```

1.1

In [3]:
```
df.drop('Unnamed: 0', axis=1, inplace=True)
df
```

Out[3]:

	avg_shares	avg_comments	avg_expert	popularity_score
0	147.3	23.9	19.1	14.6
1	28.6	1.5	33.0	7.3
2	17.9	37.6	21.6	8.0
3	94.2	4.9	8.1	9.7
4	293.6	27.7	1.8	20.7
...	...	...	...	...
195	4.1	11.6	5.7	3.2
196	76.4	26.7	22.3	11.8
197	218.5	5.4	27.4	12.2
198	140.3	1.9	9.0	10.3
199	266.9	43.8	5.0	25.4

200 rows × 4 columns

1.2

In [4]:
```
null_filter = df['avg_shares'].isnull()
df[null_filter]
mean = df['avg_shares'].mean()
df['avg_shares'].fillna(mean, inplace=True)
df[19:20]
```

Out[4]:

	avg_shares	avg_comments	avg_expert	popularity_score
19	147.291457	7.6	7.2	9.7

In [5]:
```
null_filter = df['avg_comments'].isnull()
df[null_filter]
mean = df['avg_comments'].mean()
df['avg_comments'].fillna(mean, inplace=True)
df[7:46]
```

Out[5]:

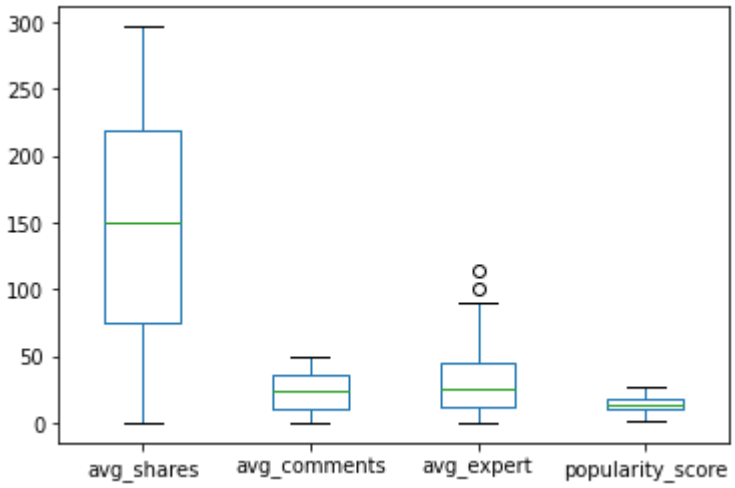
	avg_shares	avg_comments	avg_expert	popularity_score
7	168.400000	23.319388	12.8	11.7
8	280.200000	10.100000	21.4	14.8
9	19.400000	16.000000	22.3	6.6
10	107.400000	14.000000	10.9	11.5
11	177.000000	9.300000	6.4	12.8
12	296.400000	36.300000	100.9	23.8
13	237.400000	27.500000	11.0	18.9
14	232.100000	8.600000	8.7	13.4
15	206.900000	8.400000	26.4	12.9
16	131.100000	42.800000	28.9	18.0
17	191.100000	28.700000	18.2	17.3
18	151.500000	41.300000	58.5	18.5
19	147.291457	7.600000	7.2	9.7
20	120.200000	19.600000	11.6	13.2
21	43.100000	26.700000	35.1	10.1
22	197.600000	3.500000	5.9	11.7
23	239.300000	15.500000	27.3	15.7
24	74.700000	49.400000	45.7	14.7
25	109.800000	14.300000	31.7	12.4
26	202.500000	23.319388	31.6	16.6
27	141.300000	26.800000	46.2	15.5
28	27.500000	1.600000	20.7	6.9
29	38.200000	3.700000	13.8	7.6
30	95.700000	1.400000	7.4	9.5
31	248.400000	30.200000	20.3	20.2
32	205.000000	45.100000	19.6	22.6
33	67.800000	36.600000	114.0	12.5
34	261.300000	42.700000	54.7	24.2
35	117.200000	14.700000	5.4	11.9
36	171.300000	39.700000	37.7	19.0
37	163.500000	23.319388	7.4	18.0
38	240.100000	7.300000	8.7	13.2
39	240.100000	16.700000	22.9	15.9
40	239.900000	41.500000	18.5	23.2
41	292.900000	28.300000	43.2	21.4
42	104.600000	5.700000	34.4	10.4
43	109.800000	47.800000	51.4	16.7

	avg_shares	avg_comments	avg_expert	popularity_score
44	289.700000	42.300000	51.2	25.4
45	70.600000	23.319388	40.8	10.5

1.3

In [6]:
```
# Yes there is an outlier for avg_expert
df.boxplot(column =['avg_shares','avg_comments','avg_expert','popularity_score'], grid =
```

Out[6]: <AxesSubplot:>



1.4

In [7]:
```
from sklearn import preprocessing
normalized_x = preprocessing.normalize(df)
np_array = normalized_x
df = pd.DataFrame(np_array, columns=['avg_shares','avg_comments','avg_expert','popularit
df
```

Out[7]:

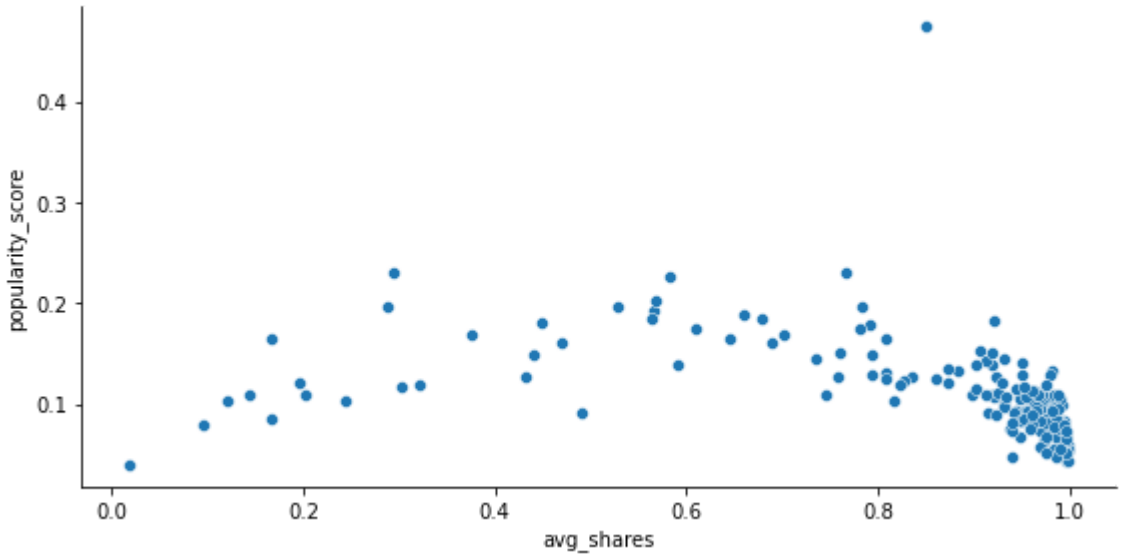
	avg_shares	avg_comments	avg_expert	popularity_score
0	0.974525	0.158121	0.126364	0.096592
1	0.645597	0.033860	0.744919	0.164785
2	0.376136	0.790096	0.453885	0.168105
3	0.989807	0.051487	0.085111	0.101923
4	0.993117	0.093697	0.006089	0.070019
...	...	...	...	...
195	0.294287	0.832617	0.409131	0.229687
196	0.901235	0.314961	0.263057	0.139196
197	0.990413	0.024477	0.124198	0.055300
198	0.995191	0.013477	0.063840	0.073061
199	0.982311	0.161204	0.018402	0.093483

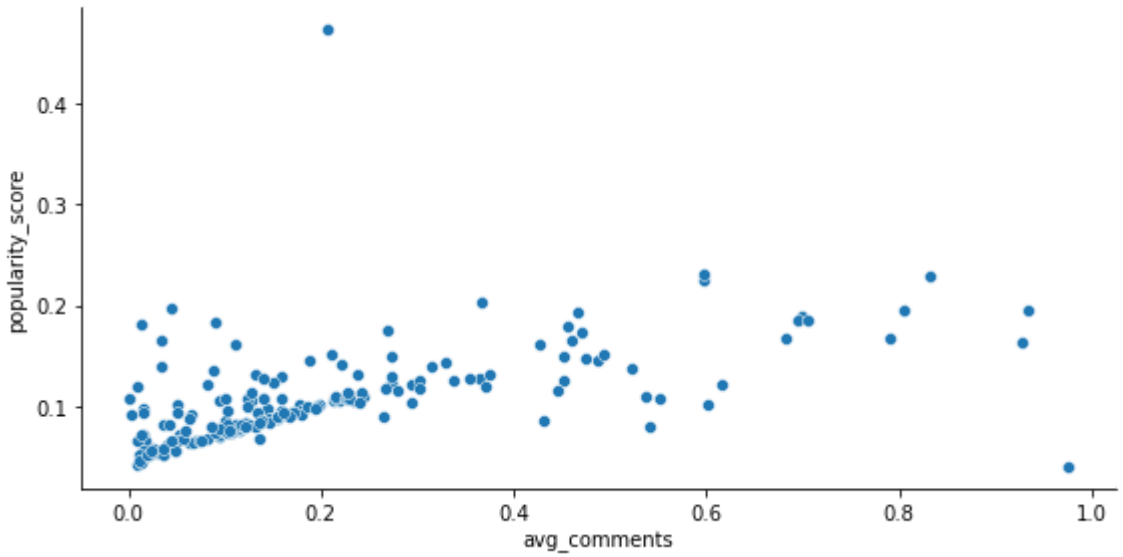
200 rows × 4 columns

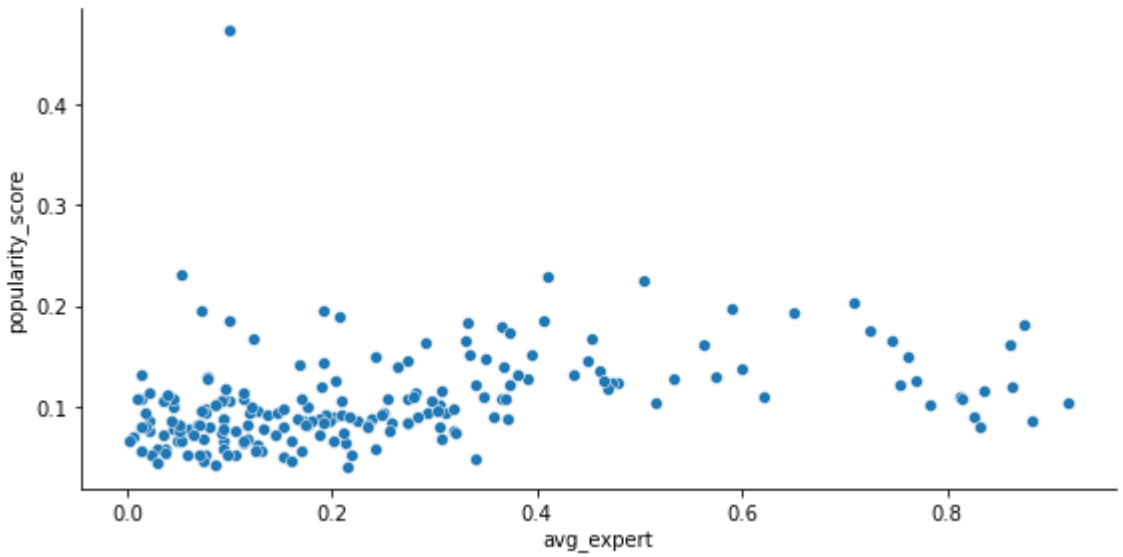
Question 2

In [8]:
```
sns.pairplot(df, x_vars=['avg_shares'],y_vars='popularity_score', height=4, aspect=2)
sns.pairplot(df, x_vars='avg_comments',y_vars='popularity_score', height=4, aspect=2)
sns.pairplot(df, x_vars='avg_expert',y_vars='popularity_score', height=4, aspect=2)
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x21d81b5a130>







Question 3

In [9]:
```
from sklearn.model_selection import train_test_split
X = df[['avg_shares','avg_comments','avg_expert']]
y = df[['popularity_score']]
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)
```

Question 4

In [12]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
```

In [13]:
```
print(X_train.isnull().sum())
print(y_train.isnull().sum())
```

```
avg_shares      0
avg_comments    0
avg_expert      0
dtype: int64
popularity_score    0
dtype: int64
```

In [14]:
```
linreg.fit(X_train, y_train)
print(linreg.intercept_)
print(linreg.coef_)
```

```
[-0.0007503]
[[0.06481305 0.18047553 0.06373593]]
```

Question 5

In [15]:
```
import statsmodels.api as sm
model = sm.OLS(y, X).fit()
```

In [16]:
```
model.summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	popularity_score	**R-squared (uncentered):**	0.893
Model:	OLS	**Adj. R-squared (uncentered):**	0.891
Method:	Least Squares	**F-statistic:**	546.6
Date:	Wed, 16 Jun 2021	**Prob (F-statistic):**	3.35e-95
Time:	00:19:08	**Log-Likelihood:**	371.42
No. Observations:	200	**AIC:**	-736.8
Df Residuals:	197	**BIC:**	-727.0
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
avg_shares	0.0668	0.004	17.150	0.000	0.059	0.075
avg_comments	0.1574	0.014	11.159	0.000	0.130	0.185
avg_expert	0.0634	0.012	5.107	0.000	0.039	0.088

Omnibus:	256.619	**Durbin-Watson:**	2.004
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	21260.439
Skew:	5.208	**Prob(JB):**	0.00
Kurtosis:	52.424	**Cond. No.**	5.81

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [17]:
```
model.pvalues
```

Out[17]:
```
avg_shares      6.124916e-41
avg_comments    1.006212e-22
avg_expert      7.707054e-07
dtype: float64
```

Question 6

In [18]:
```
y_predict = linreg.predict(X_test)
y_predict
```

Out[18]:
```
array([[0.0720563 ],
       [0.08660471],
       [0.07265318],
       [0.10227488],
       [0.13127037],
       [0.12342024],
       [0.10754271],
       [0.08927573],
       [0.09422479],
       [0.08460007],
       [0.08843221],
       [0.08194054],
       [0.1911945 ],
       [0.18705953],
       [0.09731802],
       [0.09622519],
       [0.08089296],
       [0.07105745],
       [0.08423066],
       [0.09585158],
       [0.15872331],
       [0.09274191],
       [0.14211806],
       [0.16104081],
       [0.10678353],
       [0.09401294],
       [0.09194249],
       [0.11239848],
       [0.09883387],
       [0.08762061],
       [0.12382069],
       [0.1951499 ],
       [0.15785695],
       [0.18115173],
       [0.09060002],
       [0.19013974],
       [0.11602044],
       [0.11719355],
       [0.08843043],
       [0.10198118],
       [0.10040286],
       [0.12825199],
       [0.10509876],
       [0.07506071],
       [0.07733673],
       [0.07661564],
       [0.08213683],
       [0.11976968],
       [0.10098521],
       [0.10650292],
       [0.0990025 ],
       [0.10109411],
       [0.107234  ],
       [0.11144135],
       [0.16557923],
       [0.0706933 ],
       [0.10287663],
       [0.1633025 ],
       [0.08375977],
       [0.17101848]])
```

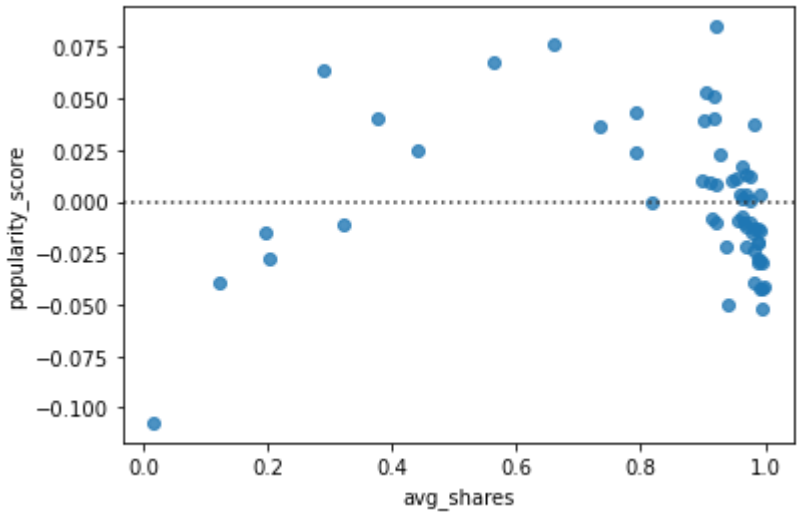
In [20]:
```
from sklearn import metrics
print ("root_mean_squared_error :",np.sqrt(metrics.mean_squared_error(y_test, y_predict)
print ("mean_absolute_error :",metrics.mean_absolute_error(y_test, y_predict))
print ("mean_squared_error :",metrics.mean_squared_error(y_test, y_predict))
```

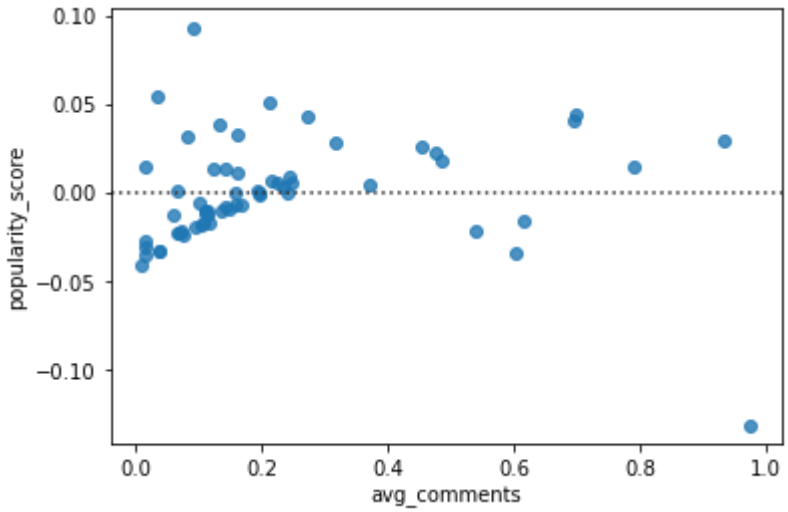
```
root_mean_squared_error : 0.03132526505928333
mean_absolute_error : 0.0209820029840246
mean_squared_error : 0.000981272231034357
```

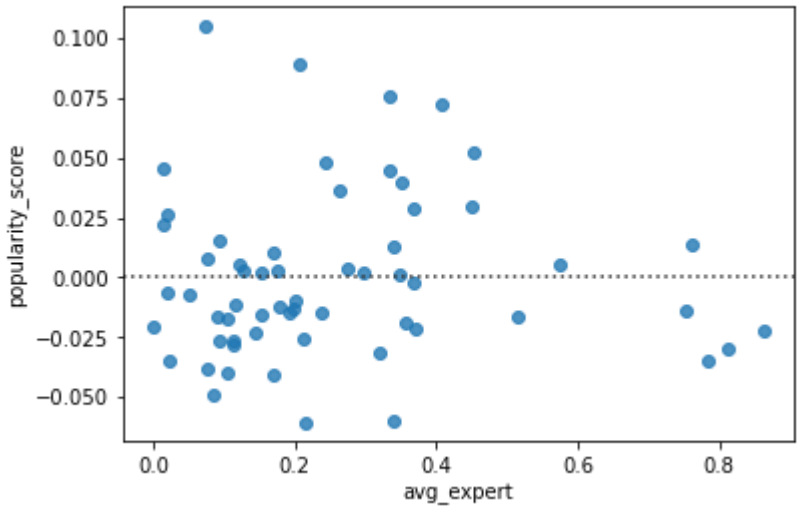
Question 7

In [21]:
```
sns.residplot(X_test['avg_shares'], y_test, df)
plt.show()
sns.residplot(X_test['avg_comments'], y_test, df)
plt.show()
sns.residplot(X_test['avg_expert'], y_test, df)
plt.show()
```

```
C:\Users\Nupur goel\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variables as keyword args: x, y, data. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```







In [22]:
```
fig = sm.qqplot(y_predict, line='45')
plt.show()
```