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

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

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

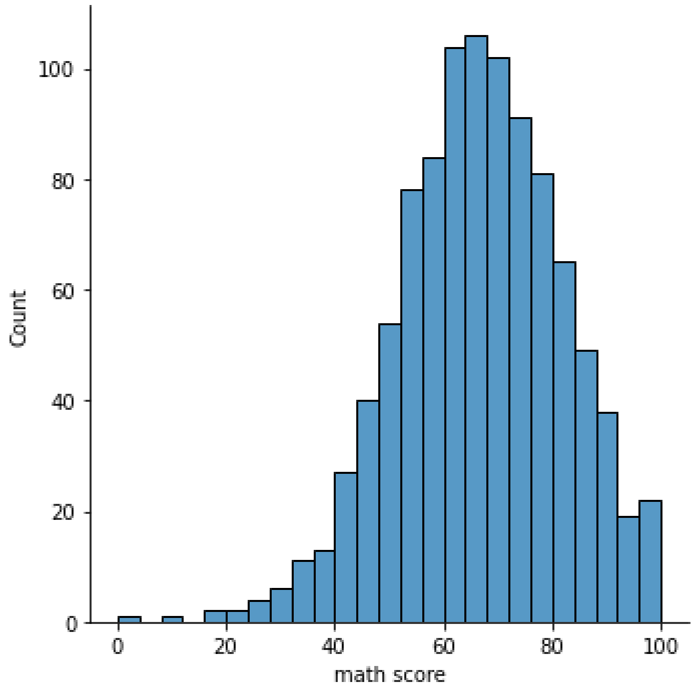
	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

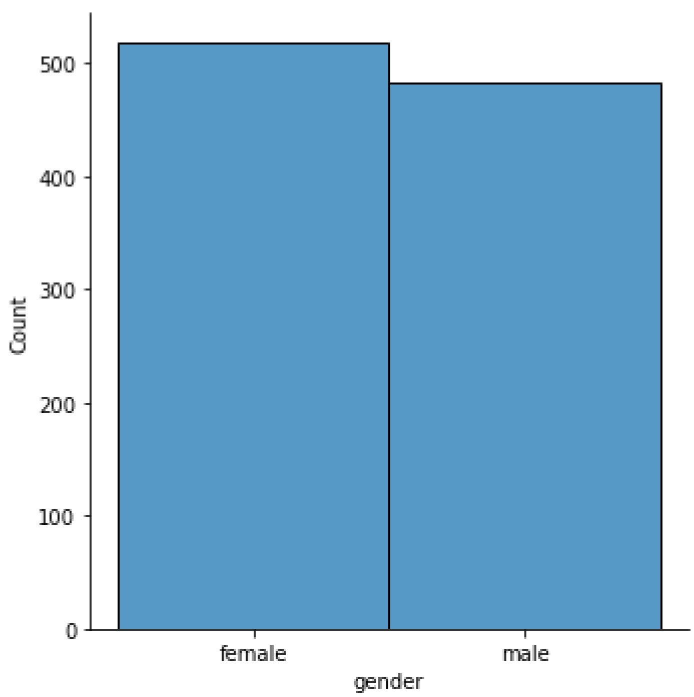
In [5]:
```
sns.displot(df, x='math score')
```

Out[5]:
```
<seaborn.axisgrid.FacetGrid at 0x221f1b109a0>
```



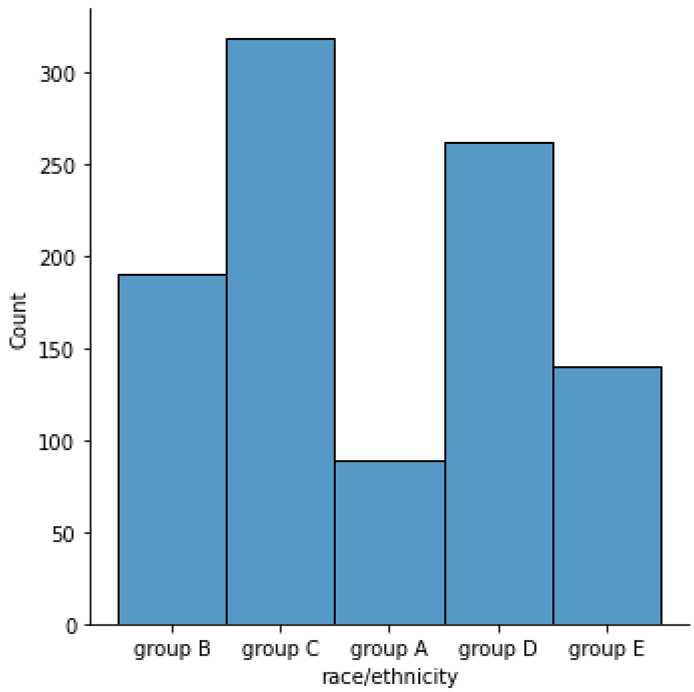
In [6]:
```
sns.displot(df, x='gender')
```

Out[6]:
```
<seaborn.axisgrid.FacetGrid at 0x221f3cbdd60>
```



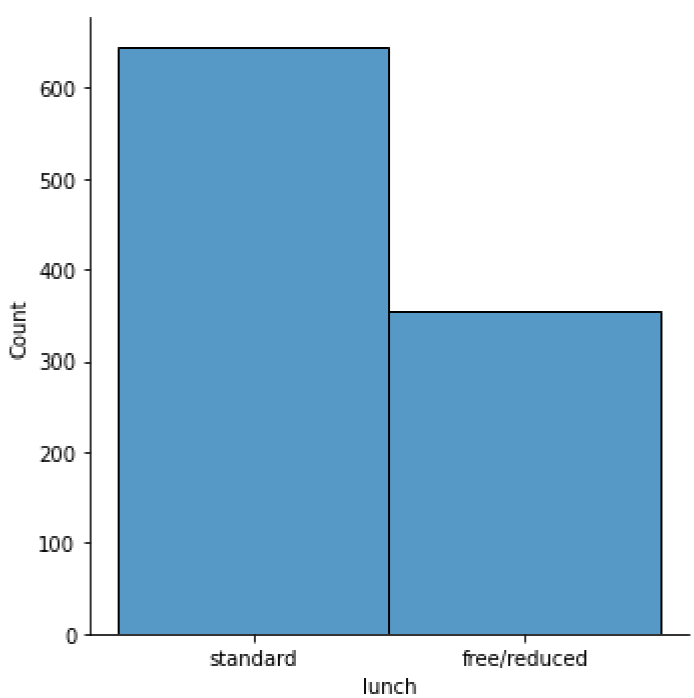
In [7]:
```
sns.displot(df, x='race/ethnicity')
```

Out[7]: <seaborn.axisgrid.FacetGrid at 0x221f3cf8c70>



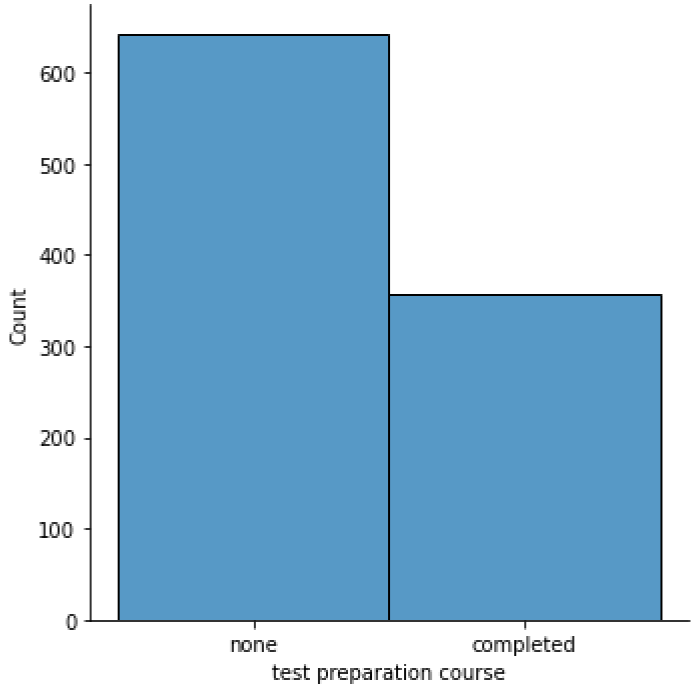
In [8]:
```
sns.displot(df, x='lunch')
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0x221f3d12460>



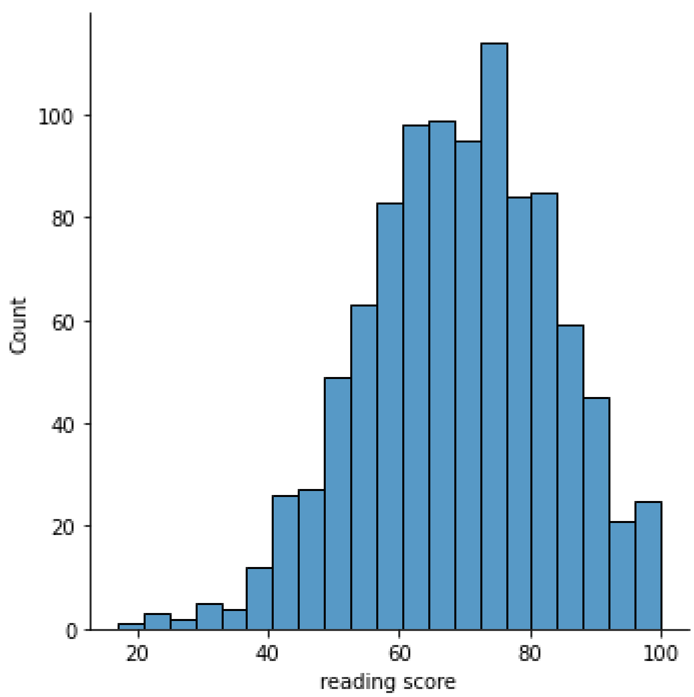
In [9]:
```
sns.displot(df, x='test preparation course')
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x221f3d78d60>



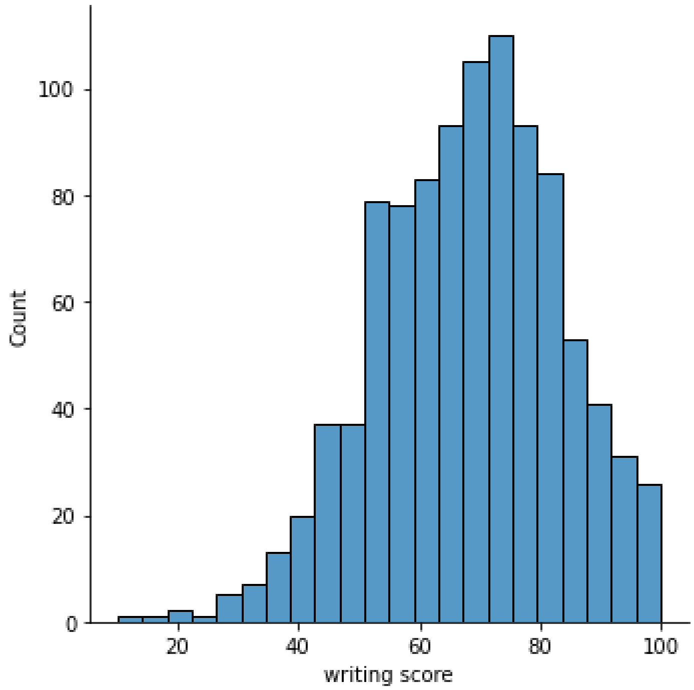
In [10]:
```
sns.displot(df, x='reading score')
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x221f3d78f70>



In [11]:
```
sns.displot(df, x='writing score')
```

Out[11]: <seaborn.axisgrid.FacetGrid at 0x221f3e25520>



In [12]:
```
df_numerical = df[['math score','writing score','reading score']]
df_categorical = df[['gender','lunch','race/ethnicity','parental level of educati
```

In [13]:
```
from sklearn.preprocessing import StandardScaler
```

In [14]:
```
scaler = StandardScaler()
```

In [15]:
```
model = scaler.fit(df_numerical)
```

In [16]:
```
scaled_numerical = model.transform(df_numerical)
```

In [17]:
```
print(scaled_numerical)
```

```
[[ 0.39002351  0.39149181  0.19399858]
 [ 0.19207553  1.31326868  1.42747598]
 [ 1.57771141  1.64247471  1.77010859]
 ...
 [-0.46775108 -0.20107904  0.12547206]
 [ 0.12609287  0.58901542  0.60515772]
 [ 0.71993682  1.18158627  1.15336989]]
```

In [18]:
```
df_numerical = pd.DataFrame(scaled_numerical, columns = ['math score','writing sc
```

In [19]:
```
df_numerical
```

Out[19]:

	math score	writing score	reading score
0	0.390024	0.391492	0.193999
1	0.192076	1.313269	1.427476
2	1.577711	1.642475	1.770109
3	-1.259543	-1.583744	-0.833899
4	0.653954	0.457333	0.605158
...	...	...	...
995	1.445746	1.774157	2.044215
996	-0.269803	-0.859491	-0.970952
997	-0.467751	-0.201079	0.125472
998	0.126093	0.589015	0.605158
999	0.719937	1.181586	1.153370

1000 rows × 3 columns

In [20]:
```
df_categorical = pd.get_dummies(df_categorical, drop_first=True)
```

In [21]:
```
df_categorical
```

Out[21]:

	gender_male	lunch_standard	race/ethnicity_group B	race/ethnicity_group C	race/ethnicity_group D
0	0	1	1	0	0
1	0	1	0	1	0
2	0	1	1	0	0
3	1	0	0	0	0
4	1	1	0	1	0
...	...	...	...	...	...
995	0	1	0	0	0
996	1	0	0	1	0
997	0	0	0	1	0
998	0	1	0	0	1
999	0	0	0	0	1

1000 rows × 12 columns

In [22]:
```
df_multreg2 = pd.concat([df_numerical, df_categorical], axis=1, ignore_index=True
```

In [23]:
```
df_multreg2
```

Out[23]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.390024	0.391492	0.193999	0	1	1	0	0	0	1	0	0	0	0	1
1	0.192076	1.313269	1.427476	0	1	0	1	0	0	0	0	0	1	0	0
2	1.577711	1.642475	1.770109	0	1	1	0	0	0	0	0	1	0	0	1
3	-1.259543	-1.583744	-0.833899	1	0	0	0	0	0	0	0	0	0	0	1
4	0.653954	0.457333	0.605158	1	1	0	1	0	0	0	0	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	1.445746	1.774157	2.044215	0	1	0	0	0	1	0	0	1	0	0	0
996	-0.269803	-0.859491	-0.970952	1	0	0	1	0	0	0	1	0	0	0	1
997	-0.467751	-0.201079	0.125472	0	0	0	1	0	0	0	1	0	0	0	0
998	0.126093	0.589015	0.605158	0	1	0	0	1	0	0	0	0	1	0	0
999	0.719937	1.181586	1.153370	0	0	0	0	1	0	0	0	0	1	0	1

1000 rows × 15 columns

In [24]:
```
df_multreg2.columns = ['math score','writing score','reading score','gender_male'
```

In [25]:
```
df_multreg2
```

Out[25]:

	math score	writing score	reading score	gender_male	lunch_standard	race/ethnicity_group B	race/ethni
0	0.390024	0.391492	0.193999	0	1	1	
1	0.192076	1.313269	1.427476	0	1	0	
2	1.577711	1.642475	1.770109	0	1	1	
3	-1.259543	-1.583744	-0.833899	1	0	0	
4	0.653954	0.457333	0.605158	1	1	0	
...	...	...	...	...	...	...	
995	1.445746	1.774157	2.044215	0	1	0	
996	-0.269803	-0.859491	-0.970952	1	0	0	
997	-0.467751	-0.201079	0.125472	0	0	0	
998	0.126093	0.589015	0.605158	0	1	0	
999	0.719937	1.181586	1.153370	0	0	0	

1000 rows × 15 columns

In [32]:
```
df_multreg2 = df_multreg2.rename(columns = {"race/ethnicity_group B":"Race_B"})
```

In [49]:
```
df_multreg2 = df_multreg2.rename(columns = {"math score":"math_score"})
```

In [50]:
```
df_multreg2 = df_multreg2.rename(columns = {"writing score":"writing_score"})
```

In [51]:
```
df_multreg2 = df_multreg2.rename(columns = {"reading score":"reading_score"})
```

In [52]:
```
df_multreg2 = df_multreg2.rename(columns = {"race/ethnicity_group C":"Race_C"})
```

In [53]:
```
df_multreg2 = df_multreg2.rename(columns = {"race/ethnicity_group D":"Race_D"})
```

In [54]:
```
df_multreg2 = df_multreg2.rename(columns = {"race/ethnicity_group E":"Race_E"})
```

In [55]:
```
df_multreg2 = df_multreg2.rename(columns = {"race/ethnicity_group B":"Race_B"})
```

In [56]:
```
df_multreg2 = df_multreg2.rename(columns = {"parental level of education_bachelor
```

In [57]:
```
df_multreg2 = df_multreg2.rename(columns = {"parental level of education_high sch
```

In [58]:
```
df_multreg2 = df_multreg2.rename(columns = {"parental level of education_masters
```

In [59]:
```
df_multreg2 = df_multreg2.rename(columns = {"parental level of education_some col
```

In [60]:
```
df_multreg2 = df_multreg2.rename(columns = {"parental level of education_some hig
```

In [61]:
```
df_multreg2
```

Out[61]:

	math_score	writing_score	reading_score	gender_male	lunch_standard	Race_B	Race_C	Ra
0	0.390024	0.391492	0.193999	0	1	1	0	
1	0.192076	1.313269	1.427476	0	1	0	1	
2	1.577711	1.642475	1.770109	0	1	1	0	
3	-1.259543	-1.583744	-0.833899	1	0	0	0	
4	0.653954	0.457333	0.605158	1	1	0	1	
...	...	...	...	...	...	...	...	
995	1.445746	1.774157	2.044215	0	1	0	0	
996	-0.269803	-0.859491	-0.970952	1	0	0	1	
997	-0.467751	-0.201079	0.125472	0	0	0	1	
998	0.126093	0.589015	0.605158	0	1	0	0	
999	0.719937	1.181586	1.153370	0	0	0	0	

1000 rows × 16 columns

In [62]:
```
import statsmodels.api as sm
```

In [63]:
```
df_multreg2['intercept']=1
```

In [64]:
```
lm = sm.OLS(df_multreg2['math_score'],df_multreg2[['intercept','writing_score','r
results = lm.fit()
results.summary()
```

Out[64]:

OLS Regression Results

Dep. Variable:	math_score	**R-squared:**	0.877
Model:	OLS	**Adj. R-squared:**	0.875
Method:	Least Squares	**F-statistic:**	500.3
Date:	Mon, 31 Jan 2022	**Prob (F-statistic):**	0.00
Time:	19:15:19	**Log-Likelihood:**	-372.34
No. Observations:	1000	**AIC:**	774.7
Df Residuals:	985	**BIC:**	848.3
Df Model:	14		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
intercept	-0.7735	0.051	-15.277	0.000	-0.873	-0.674
writing_score	0.7031	0.044	16.120	0.000	0.617	0.789
reading_score	0.2537	0.040	6.266	0.000	0.174	0.333
gender_male	0.8736	0.025	35.599	0.000	0.825	0.922
lunch_standard	0.2120	0.025	8.585	0.000	0.164	0.260
Race_B	0.0551	0.046	1.207	0.228	-0.035	0.145
Race_C	0.0118	0.043	0.275	0.784	-0.072	0.096
Race_D	0.0065	0.044	0.147	0.883	-0.080	0.093
Race_E	0.3350	0.049	6.888	0.000	0.240	0.430
bachelors	-0.0691	0.041	-1.700	0.089	-0.149	0.011
high school	0.0375	0.035	1.061	0.289	-0.032	0.107
masters	-0.1225	0.052	-2.340	0.019	-0.225	-0.020
some college	0.0264	0.034	0.788	0.431	-0.039	0.092
some_high_school	0.0364	0.036	1.004	0.316	-0.035	0.108
test preparation course_none	0.2311	0.026	8.831	0.000	0.180	0.282

Omnibus:	0.330	**Durbin-Watson:**	1.986
Prob(Omnibus):	0.848	**Jarque-Bera (JB):**	0.402
Skew:	-0.034	**Prob(JB):**	0.818
Kurtosis:	2.930	**Cond. No.**	12.9

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [65]:
```
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [76]:
```
X = df_multreg2[['intercept','writing_score','reading_score','math_score','gende
```

In [77]:
```
vif = pd.DataFrame()
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['Predictor Variables'] = X.columns
```

In [78]:
```
vif.sort_values(by=['VIF'], ascending=False)
```

Out[78]:

	VIF	Predictor Variables
0	25.332213	intercept
1	19.206435	writing_score
2	13.620604	reading_score
3	8.110733	math_score
7	3.182725	Race_C
8	3.020288	Race_D
4	2.746935	gender_male
6	2.569326	Race_B
9	2.385137	Race_E
12	1.571950	some college
11	1.571663	high school
13	1.547186	some_high_school
10	1.376347	bachelors
14	1.356828	test preparation course_none
15	1.221831	masters
5	1.198670	lunch_standard

Need to look at original model and remove any variables that have a p-value greater than 0.05. When looking at the VIF or variance inflation factor you typically want to remove anything that is over a ten. Following these two guidelines we are going to be taking away Race_C, writing score, reading score, and race d. We will re-run the model with the new set of varialbes to try and increase our r-squared and lower the p values in the model.

In [79]:
```
df_multreg3 = df_multreg2[['intercept','math_score','gender_male','Race_B','Race_
```

In [80]:
```
lm = sm.OLS(df_multreg2['math_score'],df_multreg2[['intercept','gender_male','lur
results = lm.fit()
results.summary()
```

Out[80]:

OLS Regression Results

Dep. Variable:	math_score	**R-squared:**	0.245
Model:	OLS	**Adj. R-squared:**	0.237
Method:	Least Squares	**F-statistic:**	32.08
Date:	Mon, 31 Jan 2022	**Prob (F-statistic):**	4.32e-54
Time:	19:29:49	**Log-Likelihood:**	-1278.5
No. Observations:	1000	**AIC:**	2579.
Df Residuals:	989	**BIC:**	2633.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
intercept	-0.3500	0.086	-4.092	0.000	-0.518	-0.182
gender_male	0.3284	0.055	5.924	0.000	0.220	0.437
lunch_standard	0.7210	0.058	12.457	0.000	0.607	0.835
Race_B	-0.0791	0.072	-1.097	0.273	-0.221	0.062
Race_E	0.4534	0.082	5.551	0.000	0.293	0.614
bachelors	0.1239	0.100	1.245	0.214	-0.071	0.319
high school	-0.3225	0.086	-3.749	0.000	-0.491	-0.154
masters	0.2173	0.128	1.694	0.091	-0.035	0.469
some college	-0.0300	0.083	-0.363	0.717	-0.192	0.132
some_high_school	-0.2838	0.088	-3.222	0.001	-0.457	-0.111
test preparation course_none	-0.3560	0.058	-6.134	0.000	-0.470	-0.242

Omnibus:	9.258	**Durbin-Watson:**	2.051
Prob(Omnibus):	0.010	**Jarque-Bera (JB):**	9.443
Skew:	-0.232	**Prob(JB):**	0.00890
Kurtosis:	2.890	**Cond. No.**	9.27

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

As you can see from the results of the model the p values did drop but the r-squared value dropped so significantly that using this reduced model would not be a good idea. To make business decisions I would use the original model with an r-squared of 0.877.

In [ ]: