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

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
company_data = pd.read_csv('Company_Data.csv')
company_data
```

Out[3]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education
0	9.50	138	73	11	276	120	Bad	42	17
1	11.22	111	48	16	260	83	Good	65	10
2	10.06	113	35	10	269	80	Medium	59	12
3	7.40	117	100	4	466	97	Medium	55	14
4	4.15	141	64	3	340	128	Bad	38	13
...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	Good	33	14
396	6.14	139	23	3	37	120	Medium	55	11
397	7.41	162	26	12	368	159	Medium	40	18
398	5.94	100	79	7	284	95	Bad	50	12
399	9.71	134	37	0	27	120	Good	49	16

400 rows × 11 columns

In [4]:
```
company_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Sales        400 non-null    float64
 1   CompPrice    400 non-null    int64  
 2   Income       400 non-null    int64  
 3   Advertising  400 non-null    int64  
 4   Population   400 non-null    int64  
 5   Price        400 non-null    int64  
 6   ShelveLoc    400 non-null    object 
 7   Age          400 non-null    int64  
 8   Education    400 non-null    int64  
 9   Urban        400 non-null    object 
 10  US           400 non-null    object 
dtypes: float64(1), int64(7), object(3)
memory usage: 34.5+ KB
```

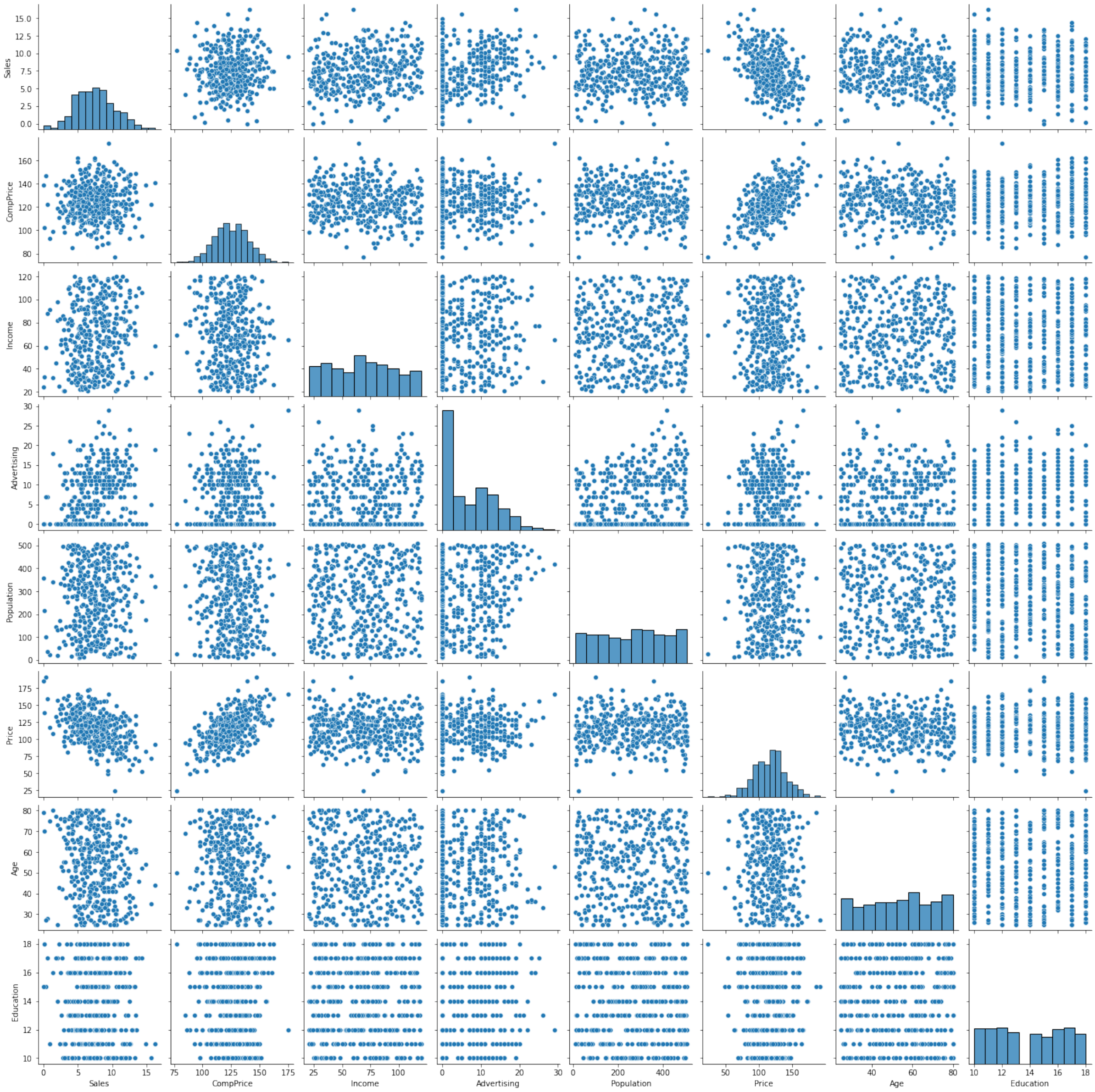
In [5]:
```
company_data.describe()
```

Out[5]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.00000
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.32250
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.00000
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.75000
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.50000
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.00000
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.00000

In [6]:
```python
# pairplot
import seaborn as sns
sns.pairplot(company_data)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x7faaff4437c0>

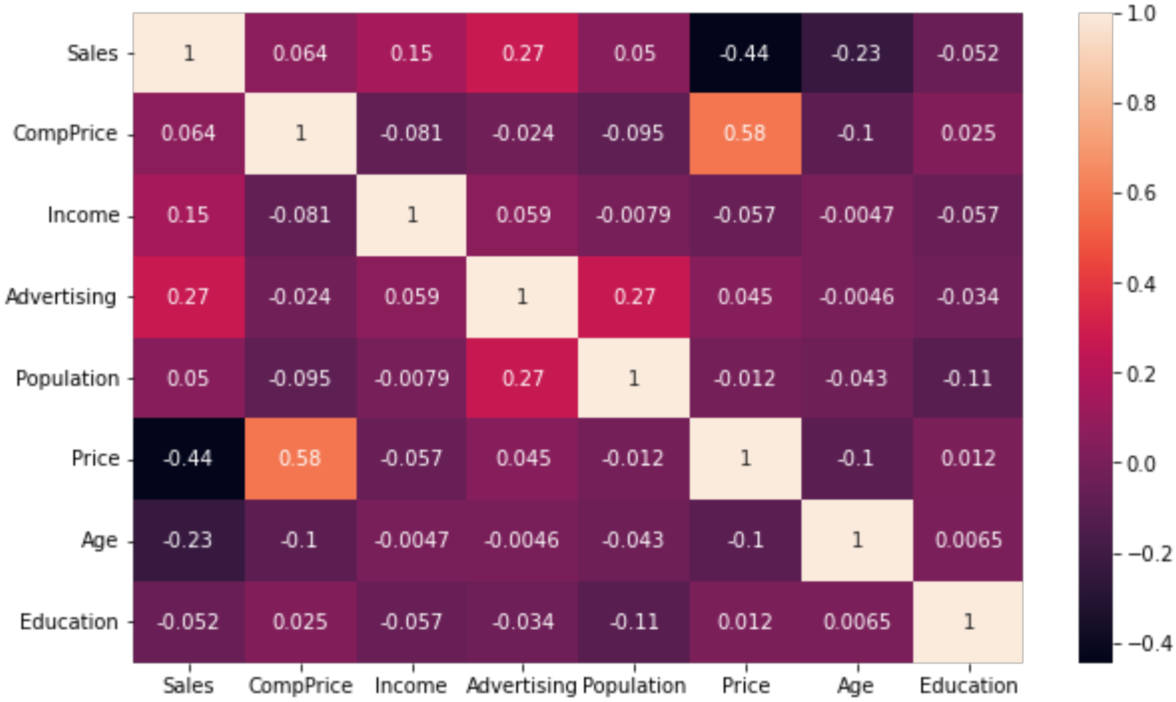


In [7]:
```
# Correlation analysis for company_data
corr = company_data.corr()
corr
```

Out[7]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age
Sales	1.000000	0.064079	0.151951	0.269507	0.050471	-0.444951	-0.231815
CompPrice	0.064079	1.000000	-0.080653	-0.024199	-0.094707	0.584848	-0.100239
Income	0.151951	-0.080653	1.000000	0.058995	-0.007877	-0.056698	-0.004670
Advertising	0.269507	-0.024199	0.058995	1.000000	0.265652	0.044537	-0.004557
Population	0.050471	-0.094707	-0.007877	0.265652	1.000000	-0.012144	-0.042663
Price	-0.444951	0.584848	-0.056698	0.044537	-0.012144	1.000000	-0.102177
Age	-0.231815	-0.100239	-0.004670	-0.004557	-0.042663	-0.102177	1.000000
Education	-0.051955	0.025197	-0.056855	-0.033594	-0.106378	0.011747	0.006488

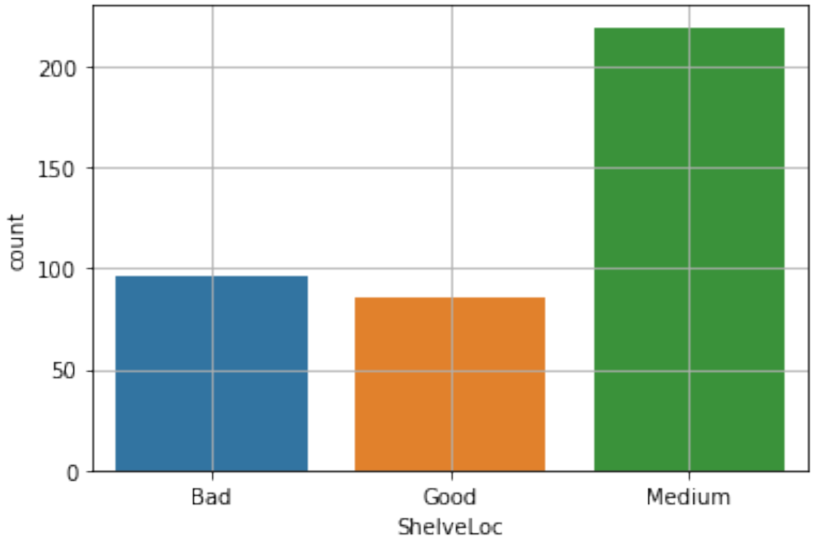
In [8]:
```
plt.figure(figsize=(10,6))
sns.heatmap(corr,annot=True)
plt.show()
```

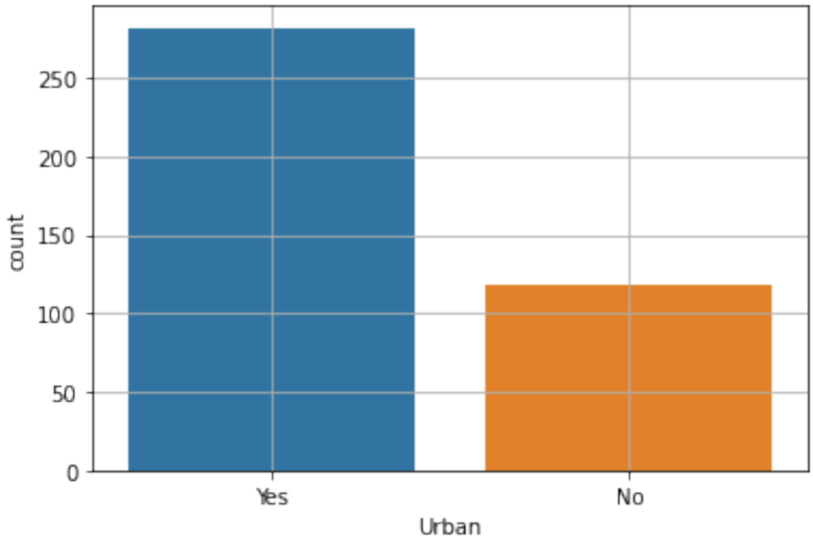


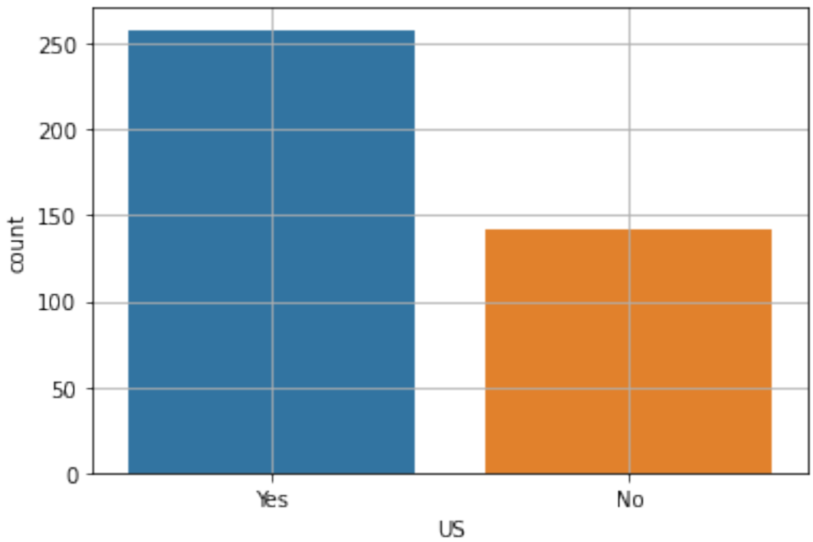
In [9]:
```
#count plot
sns.countplot(company_data['ShelveLoc'])
plt.grid(True)
plt.show()

sns.countplot(company_data['Urban'])
plt.grid(True)
plt.show()
```

```
sns.countplot(company_data['US'])
plt.grid(True)
plt.show()
```



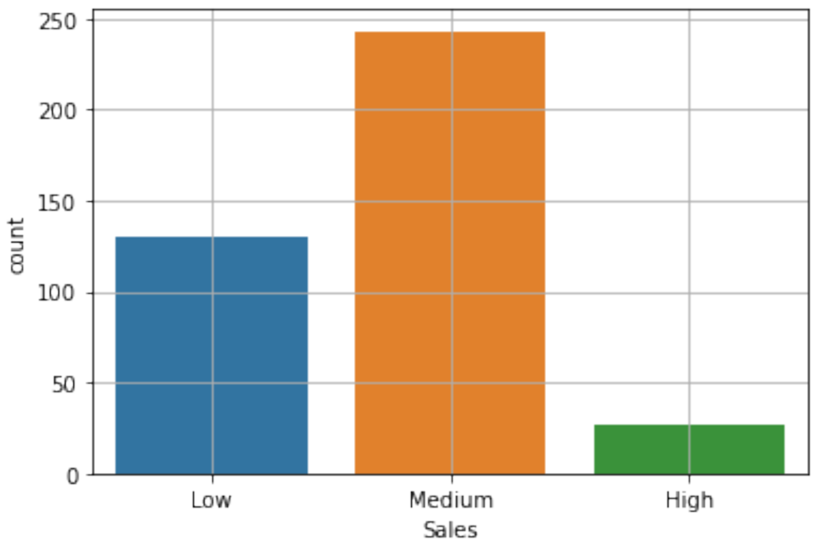




In [10]:
```
company_data['Sales'] = pd.cut(x=company_data['Sales'],bins=[0, 6,
company_data['Sales']
```

Out[10]:
```
0      Medium
1      Medium
2      Medium
3      Medium
4         Low
        ...
395      High
396    Medium
397    Medium
398       Low
399    Medium
Name: Sales, Length: 400, dtype: category
Categories (3, object): ['Low' < 'Medium' < 'High']
```

In [11]:
```
sns.countplot(company_data['Sales'])
plt.grid(True)
plt.show()
```



In [12]:
```
company_data['Sales'].value_counts()
```

Out[12]:
```
Medium    243
Low       130
High       27
Name: Sales, dtype: int64
```

In [13]:
```
company_data.head()
```

Out[13]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education
0	Medium	138	73	11	276	120	Bad	42	17
1	Medium	111	48	16	260	83	Good	65	10
2	Medium	113	35	10	269	80	Medium	59	12
3	Medium	117	100	4	466	97	Medium	55	14
4	Low	141	64	3	340	128	Bad	38	13

In [14]:
```
#encoding categorical company_data
label_encoder = preprocessing.LabelEncoder()

company_data['Sales'] = label_encoder.fit_transform(company_data['S
company_data['ShelveLoc'] = label_encoder.fit_transform(company_dat
company_data['Urban'] = label_encoder.fit_transform(company_data['U
company_data['US'] = label_encoder.fit_transform(company_data['US']

company_data
```

Out[14]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education
0	2	138	73	11	276	120	0	42	17
1	2	111	48	16	260	83	1	65	10
2	2	113	35	10	269	80	2	59	12
3	2	117	100	4	466	97	2	55	14
4	1	141	64	3	340	128	0	38	13
...	...	...	...	...	...	...	...	...	...
395	0	138	108	17	203	128	1	33	14
396	2	139	23	3	37	120	2	55	11
397	2	162	26	12	368	159	2	40	18
398	1	100	79	7	284	95	0	50	12
399	2	134	37	0	27	120	1	49	16

400 rows × 11 columns

In [15]:
```
# Input and Output variables
X = company_data.drop('Sales', axis = 1)
y = company_data[['Sales']]
```

In [16]:
```
y_test = train_test_split(X, y, test_size= 0.33, random_state= 42)
```

In [17]:
```
x_train_shape :',x_train.shape ,'\n y_train_shape :',y_train.shape)
```

```
x_train_shape : (268, 10)
 y_train_shape : (268, 1)
```

In [18]:
```
nt('x_test_shape :',x_test.shape ,'\n y_test_shape :',y_test.shape)
```

```
x_test_shape : (132, 10)
 y_test_shape : (132, 1)
```

In [19]:
```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(x_train, y_train)
```

Out[19]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Grid SearchCv

In [20]:
```
from sklearn.model_selection import GridSearchCV

grid_search = GridSearchCV(estimator = rf_model,
                           param_grid = {'criterion':['entropy','gi
                                         'max_depth':[2,3,4,5,6,7,8
                           cv=5)
grid_search.fit(X,y)
print(grid_search.best_params_)
print(grid_search.best_score_)
```

```
{'criterion': 'gini', 'max_depth': 10}
0.7425
```

In [21]:
```
rf_best_model = RandomForestClassifier(criterion = 'gini',random_st
rf_best_model.fit(x_train, y_train)
```

Out[21]:
```
RandomForestClassifier
RandomForestClassifier(max_depth=10, random_state=42)
```

In [22]:
```
#prediction train data
pred_train_y = rf_best_model.predict(x_train)
```

In [23]:
```
pred_test_y = rf_best_model.predict(x_test)
pred_test_y
```

Out[23]:
```
array([1, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 1,
       2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 1,
2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1,
2, 2,
       2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,
1, 2,
       1, 1, 2, 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2,
2, 2,
       2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2,
1, 2])
```

In [24]:
```
pd.Series(pred_test_y).value_counts()
```

Out[24]:
```
2    99
1    33
dtype: int64
```

In [25]:
```
accuracy_score(y_train,pred_train_y)
```

Out[25]:
```
0.996268656716418
```

In [26]:
```
confusion_matrix(y_train,pred_train_y)
```

Out[26]:
```
array([[ 14,   0,   0],
       [  0,  94,   1],
       [  0,   0, 159]])
```

In [27]:
```
ssification Report:\n',classification_report(y_train,pred_train_y))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      0.99      0.99        95
           2       0.99      1.00      1.00       159

    accuracy                           1.00       268
   macro avg       1.00      1.00      1.00       268
weighted avg       1.00      1.00      1.00       268
```

In [28]:
```
accuracy_score(y_test,pred_test_y)
```

Out[28]:
```
0.696969696969697
```

In [29]:
```
confusion_matrix(y_test,pred_test_y)
```

Out[29]:
```
array([[ 0,  1, 12],
       [ 0, 20, 15],
       [ 0, 12, 72]])
```

In [30]:
```
lassification Report:\n',classification_report(y_test,pred_test_y))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00        13
           1       0.61      0.57      0.59        35
           2       0.73      0.86      0.79        84

    accuracy                           0.70       132
   macro avg       0.44      0.48      0.46       132
weighted avg       0.62      0.70      0.66       132
```

In [ ]: