In [72]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv('datasets/50_Startups.csv')
print(data.head())
```

```
   R&D Spend  Administration  Marketing Spend     Profit
0  165349.20       136897.80        471784.10  192261.83
1  162597.70       151377.59        443898.53  191792.06
2  153441.51       101145.55        407934.54  191050.39
3  144372.41       118671.85        383199.62  182901.99
4  142107.34        91391.77        366168.42  166187.94
```

In [73]:
```
data.isnull().sum()
```

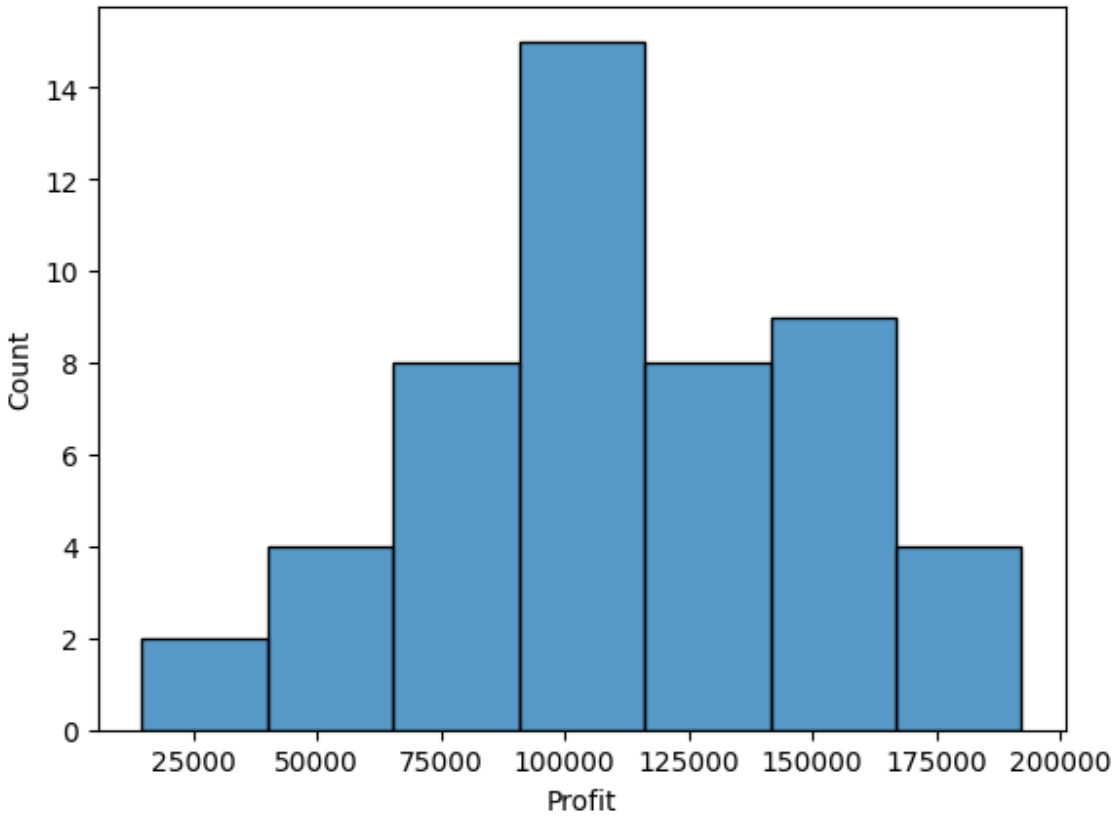
Out[73]:
```
R&D Spend          0
Administration     0
Marketing Spend    0
Profit             0
dtype: int64
```

In [74]:
```
X=data.iloc[:,:-1]
Y=data.Profit
X.head()
Y.head()
```

Out[74]:
```
0    192261.83
1    191792.06
2    191050.39
3    182901.99
4    166187.94
Name: Profit, dtype: float64
```

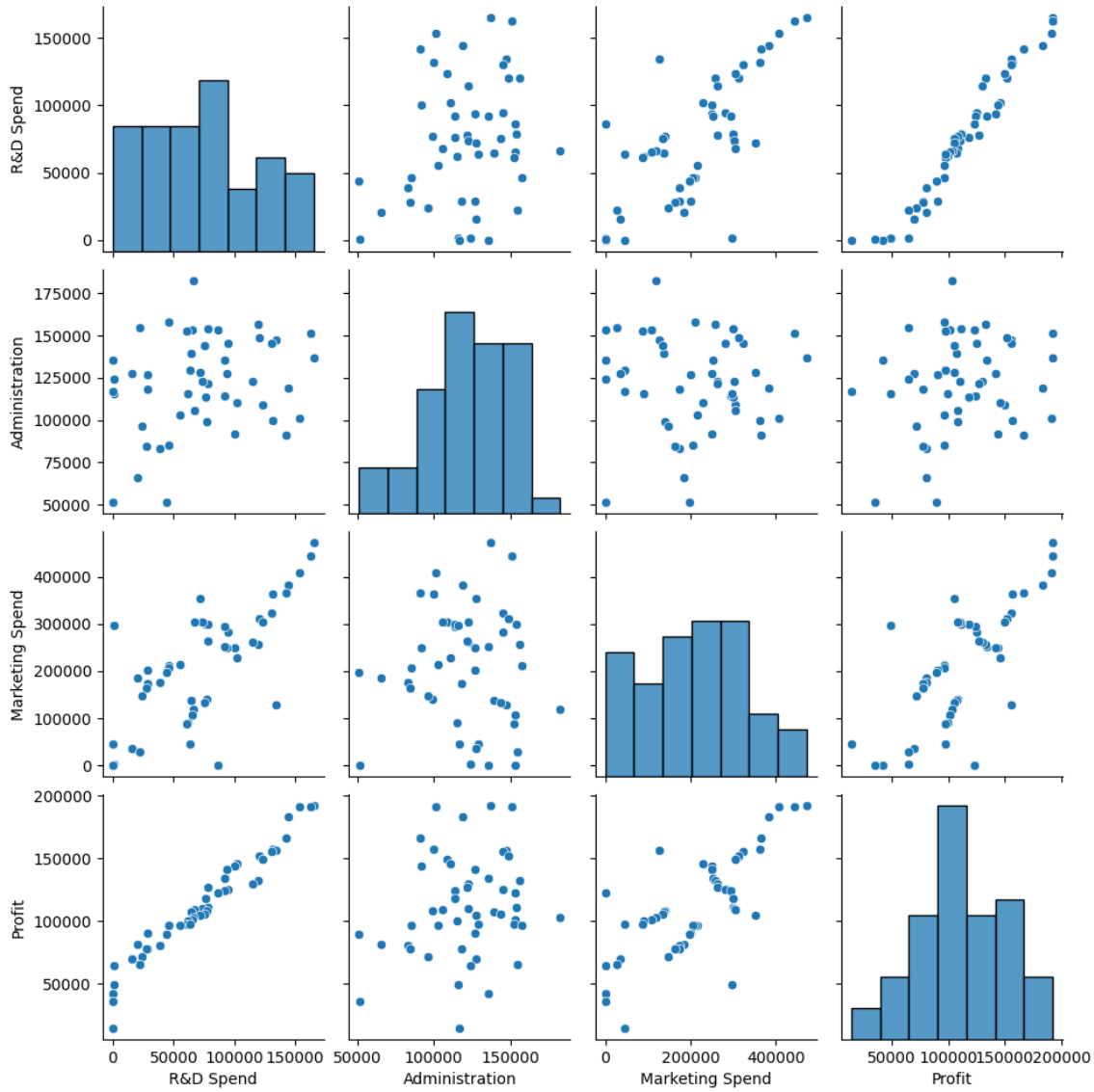
In [115]:
```
# Find relation between profit through histogram
sns.histplot(data.Profit)
```

Out[115]: <Axes: xlabel='Profit', ylabel='Count'>



In [117]:
```
sns.pairplot(data)
```

Out[117]:
```
<seaborn.axisgrid.PairGrid at 0x23cb587ea40>
```



Preparing test and train data

In [75]:
```
from sklearn.model_selection import train_test_split
x1,x2,y1,y2=train_test_split(X,Y,test_size=0.2,random_state=42)
```

In [76]:
```
from sklearn.linear_model import LinearRegression
model_linear=LinearRegression()
model_linear.fit(x1,y1)
```

Out[76]:
```
LinearRegression
LinearRegression()
```

In [77]:
```
# Accuracy of Linear Regression Model
accu=model_linear.score(x2,y2)*100
print("{:.2f}%".format(accu))
```

```
90.01%
```

In [78]:
```
new_arr=[170450,140900,850000]
new_arr=np.array(new_arr).reshape(1,-1)
```

In [79]:
```
profit=model_linear.predict(new_arr)[0]
print("Profit would be:{:.2f}/-".format(profit))
```

```
Profit would be:208060.15/-
```

Using Random Forest for same prediction

In [80]:
```
data=pd.read_csv('datasets/50_Startups.csv')
print(data.head())
```

```
   R&D Spend  Administration  Marketing Spend     Profit
0  165349.20       136897.80        471784.10  192261.83
1  162597.70       151377.59        443898.53  191792.06
2  153441.51       101145.55        407934.54  191050.39
3  144372.41       118671.85        383199.62  182901.99
4  142107.34        91391.77        366168.42  166187.94
```

Define function to reshape inputs to pass to prediction model

In [81]:
```
def reshape(arr):
    arr=np.array(arr).reshape(1,-1)
    return arr
```

In [82]:
```
data.head()
```

Out[82]:

	R&D Spend	Administration	Marketing Spend	Profit
0	165349.20	136897.80	471784.10	192261.83
1	162597.70	151377.59	443898.53	191792.06
2	153441.51	101145.55	407934.54	191050.39
3	144372.41	118671.85	383199.62	182901.99
4	142107.34	91391.77	366168.42	166187.94

Split columns which are input & output

In [83]:
```
from sklearn.model_selection import train_test_split
X=data.drop(['Profit'],axis=1)
y=data['Profit']
```

In [84]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

Join the data to perform any manipulations further

This acts as duplicate of 'data' DataFrame

In [99]:
```
train_data = X_train.join(y_train)
```

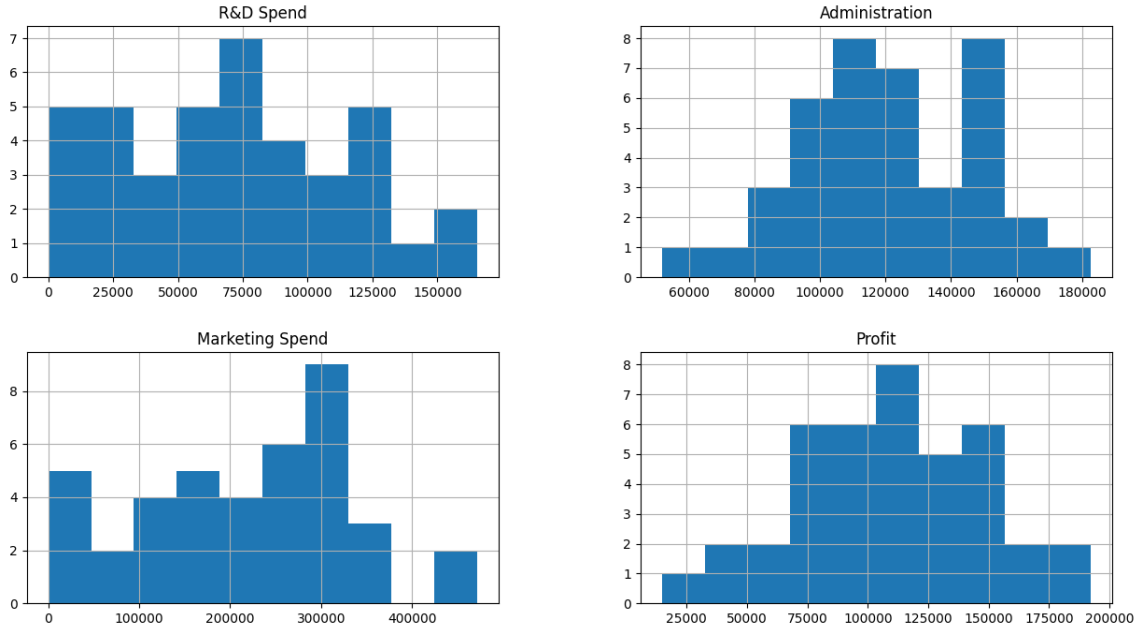
In [98]:
```
train_data.head()
```

Out[98]:

	R&D Spend	Administration	Marketing Spend	Profit
44	10.006889	11.949935	10.251878	11.085235
25	11.076986	11.846208	11.834745	11.584365
8	11.699766	11.909820	12.649521	11.933035
33	10.924047	11.543052	12.276698	11.480195
10	11.531885	11.613631	12.342184	11.892204

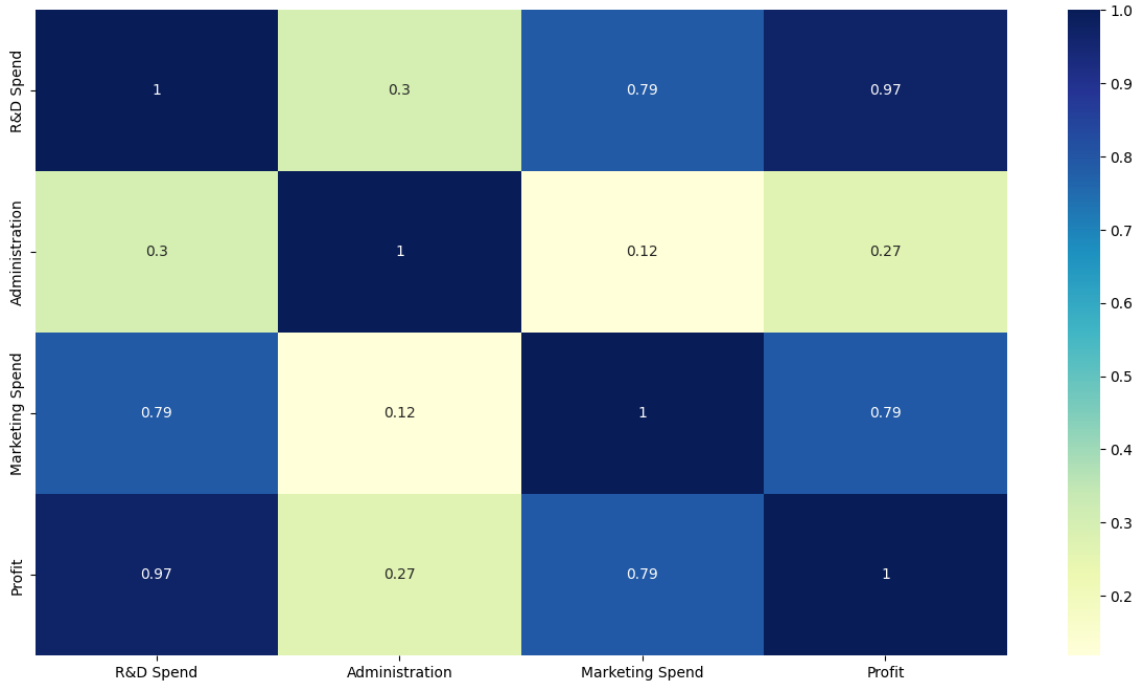
In [87]:
```
# Draw Histogram to figure out the data
train_data.hist(figsize=(15,8))
```

Out[87]:
```
array([[<Axes: title={'center': 'R&D Spend'}>,
        <Axes: title={'center': 'Administration'}>],
       [<Axes: title={'center': 'Marketing Spend'}>,
        <Axes: title={'center': 'Profit'}>]], dtype=object)
```



In [88]:
```
# Heat map to find correlation
import matplotlib.pyplot as plt
plt.figure(figsize=(15,8))
sns.heatmap(train_data.corr(),annot=True,cmap="YlGnBu")
```

Out[88]:
```
<Axes: >
```



In [89]:
```
train_data.head()
```

Out[89]:

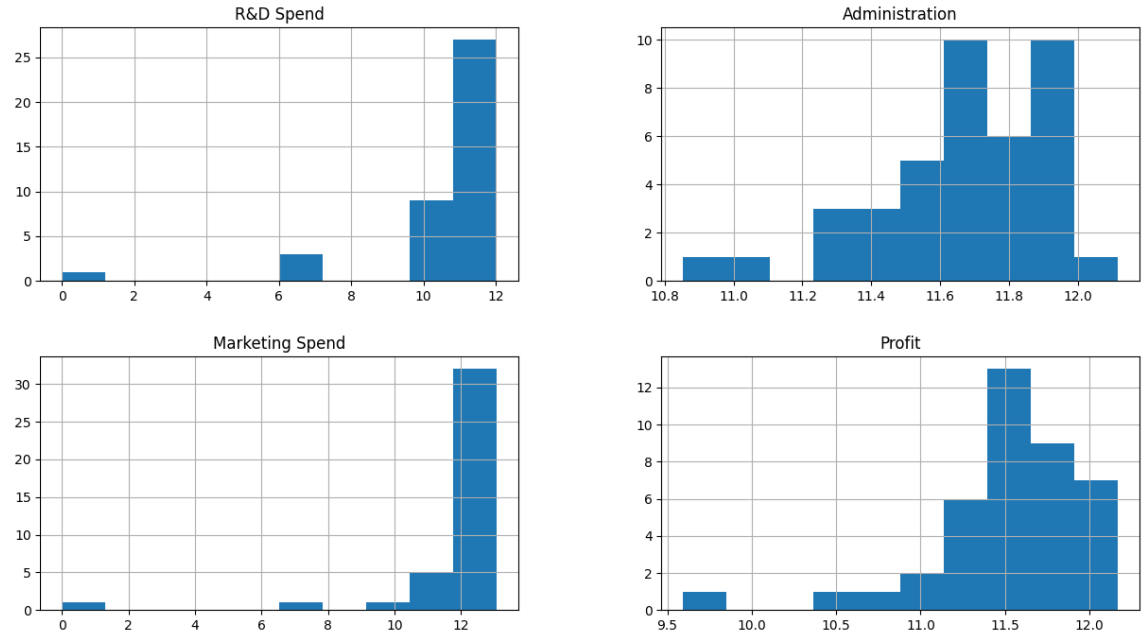
	R&D Spend	Administration	Marketing Spend	Profit
44	22177.74	154806.14	28334.72	65200.33
25	64664.71	139553.16	137962.62	107404.34
8	120542.52	148718.95	311613.29	152211.77
33	55493.95	103057.49	214634.81	96778.92
10	101913.08	110594.11	229160.95	146121.95

Transform the train_data to have logarithmic bell curve

In [90]:
```
train_data['R&D Spend']=np.log(train_data['R&D Spend']+1)
train_data['Administration']=np.log(train_data['Administration']+1)
train_data['Marketing Spend']=np.log(train_data['Marketing Spend']+1)
train_data['Profit']=np.log(train_data['Profit']+1)
```

In [91]:
```
train_data.hist(figsize=(15,8))
```

Out[91]:
```
array([[<Axes: title={'center': 'R&D Spend'}>,
        <Axes: title={'center': 'Administration'}>],
       [<Axes: title={'center': 'Marketing Spend'}>,
        <Axes: title={'center': 'Profit'}>]], dtype=object)
```



Train the Random Forest Regerssor

In [92]:
```
from sklearn.ensemble import RandomForestRegressor
forest=RandomForestRegressor()
forest.fit(X_train,y_train)
```

Out[92]:
```
RandomForestRegressor
RandomForestRegressor()
```

In [93]:
```
# Find Accuracy of the model
forest.score(X_test,y_test)
```

Out[93]:
```
0.9274556994530677
```

In [94]:
```
vals=[250000,350000,150000]
vals=reshape(vals)
forest.predict(vals)[0]
```

Out[94]:
```
160153.96929999994
```

Further more imporve the Random Forest by tweaking properties and training again

In [95]:
```
from sklearn.model_selection import GridSearchCV
param_grid={
    'n_estimators':[30,50,100],
    'max_features':[8,12,20],
    'min_samples_split':[2,4,6,8]
}

grid_search=GridSearchCV(forest,param_grid,cv=5,
                         scoring='neg_mean_squared_error',
                         return_train_score=True)
grid_search.fit(X_train,y_train)
```

Out[95]:
```
GridSearchCV
estimator: RandomForestRegressor
RandomForestRegressor
```

In [96]:
```
best_forest=grid_search.best_estimator_
```

Calculate Accuracy of imporved random forest

In [97]:
```
best_forest.score(X_test,y_test)
```

Out[97]:
```
0.934648260560111
```

In [111]:
```
linear_accuracy=model_linear.score(X_test,y_test)*100
forest_accuracy=forest.score(X_test,y_test)*100
best_forest_accuracy=best_forest.score(X_test,y_test)*100

print("Linear Regression Accuracy is:{:.2f}%".format(linear_accuracy))
print("Simple random forest Accuracy is:{:.2f}%".format(forest_accuracy))
print("Optimized random forest Accuracy is:{:.2f}%".format(best_forest_accu
```

```
Linear Regression Accuracy is:96.82%
Simple random forest Accuracy is:92.75%
Optimized random forest Accuracy is:93.46%
```

Linear Regression is most accurate, therefore will use that

1. Save all models into disk

In [118]:
```
import pickle as pkl

linear_name="linear_model.sav"
forest_name='forest_model.sav'
best_forest_name='best_forest_model.sav'

pkl.dump(model_linear,open(linear_name,'bw'))
pkl.dump(forest,open(forest_name,'bw'))
pkl.dump(best_forest,open(best_forest_name,'bw'))

print("Linear model saved as:",linear_name)
print("Basic Forest model saved as:",forest_name)
print("Best Forest model saved as:",best_forest_name)
```

```
Linear model saved as: linear_model.sav
Basic Forest model saved as: forest_model.sav
Best Forest model saved as: best_forest_model.sav
```

In [ ]:
```

```