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

In [81]:
```
data1 = pd.read_csv (r"C:\Users\vishal gajarmal\Desktop\Salary_data.CSV")
```

In [82]:
```
data1.head()
```

Out[82]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

In [4]:
```
data1.isnull () . sum()
```

Out[4]:
```
YearsExperience    0
Salary             0
dtype: int64
```

In [5]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 608.0 bytes
```

In [6]:
```
data1.corr()
```

Out[6]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

In [7]:
```
data1.describe()
```

Out[7]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [8]:
```
data1.shape
```

Out[8]:
```
(30, 2)
```

In [9]:
```
data1.var
```

Out[9]:
```
<bound method NDFrame._add_numeric_operations.<locals>.var of     YearsExperience  Salary
0               1.1   39343
1               1.3   46205
2               1.5   37731
3               2.0   43525
4               2.2   39891
5               2.9   56642
6               3.0   60150
7               3.2   54445
8               3.2   64445
9               3.7   57189
10              3.9   63218
11              4.0   55794
12              4.0   56957
13              4.1   57081
14              4.5   61111
15              4.9   67938
16              5.1   66029
17              5.3   83088
18              5.9   81363
19              6.0   93940
20              6.8   91738
21              7.1   98273
22              7.9  101302
23              8.2  113812
24              8.7  109431
25              9.0  105582
26              9.5  116969
27              9.6  112635
28             10.3  122391
29             10.5  121872>
```

In [13]:
```
data1[data1.duplicated()] . shape
```

Out[13]:
```
(0, 2)
```

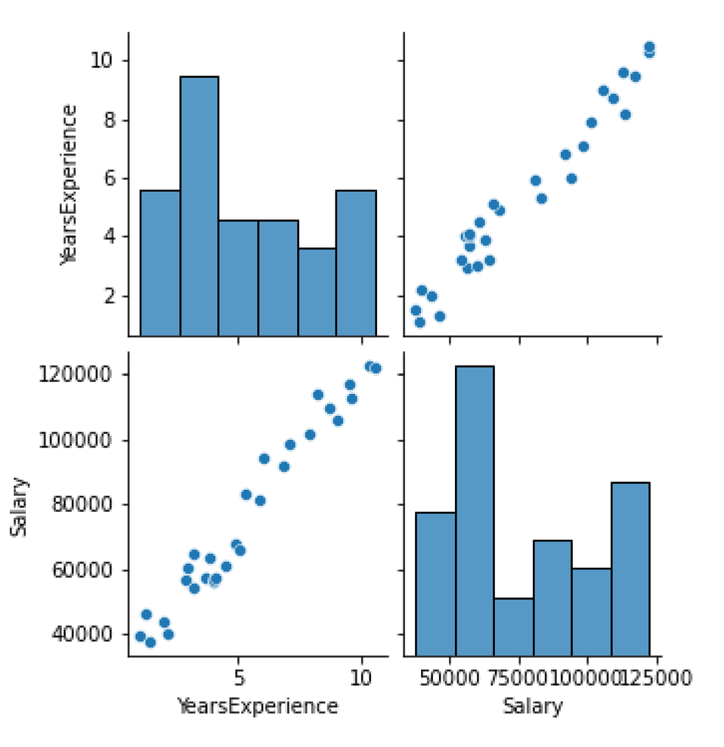
In [14]:
```
data1.dtypes
```

Out[14]:
```
YearsExperience    float64
Salary               int64
dtype: object
```

In [15]:
```
import seaborn as sea
```

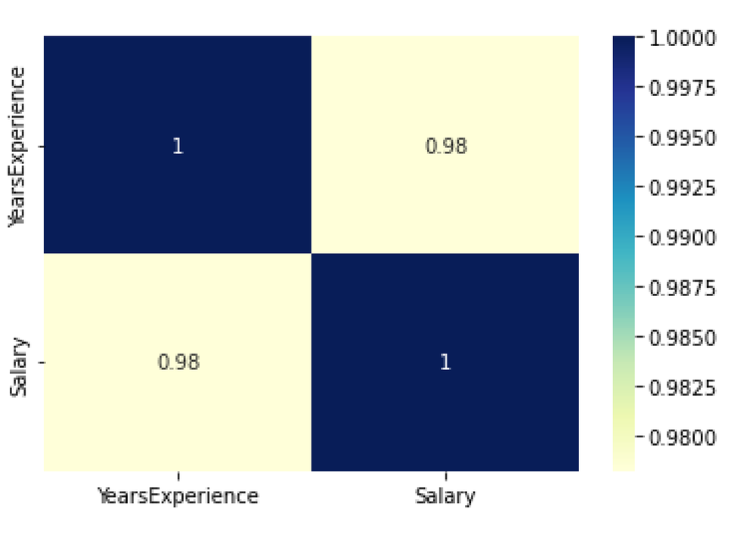
In [16]:
```
sea.pairplot(data1)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x199ae4ecf40>
```

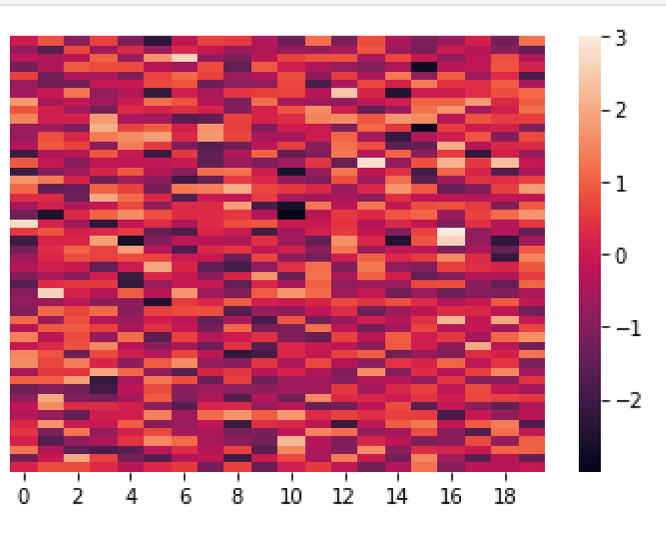


In [25]:
```
sea.heatmap (data1.corr() , cmap="YlGnBu", annot=True)
```

Out[25]:
```
<AxesSubplot:>
```



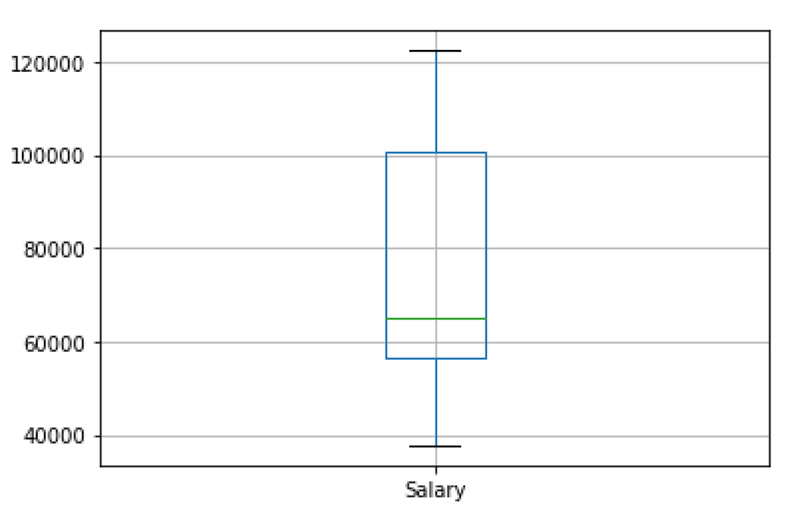
In [30]:
```
data1 = np.random.randn(50, 20)
ax = sea.heatmap(data1, xticklabels=2, yticklabels=False)
```



Outlier Detection

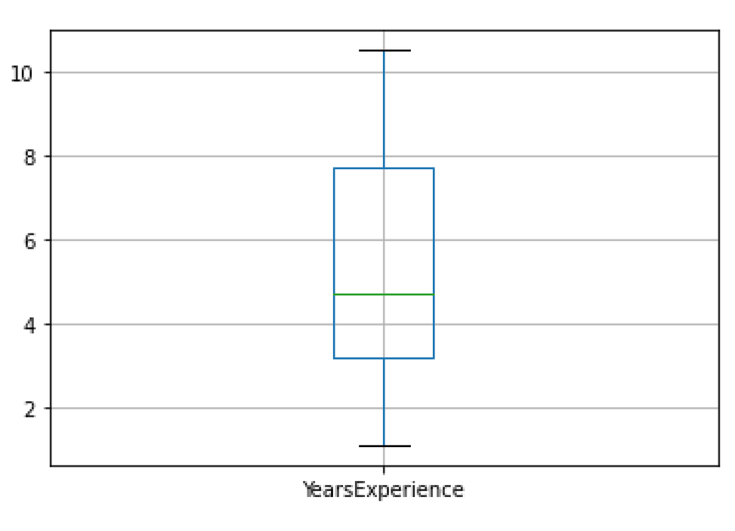
In [84]:
```
data1.boxplot("Salary")
```

Out[84]:
```
<AxesSubplot:>
```



In [86]:
```
data1.boxplot("YearsExperience")
```

Out[86]:
```
<AxesSubplot:>
```



In [87]:
```
data1.hist()
```

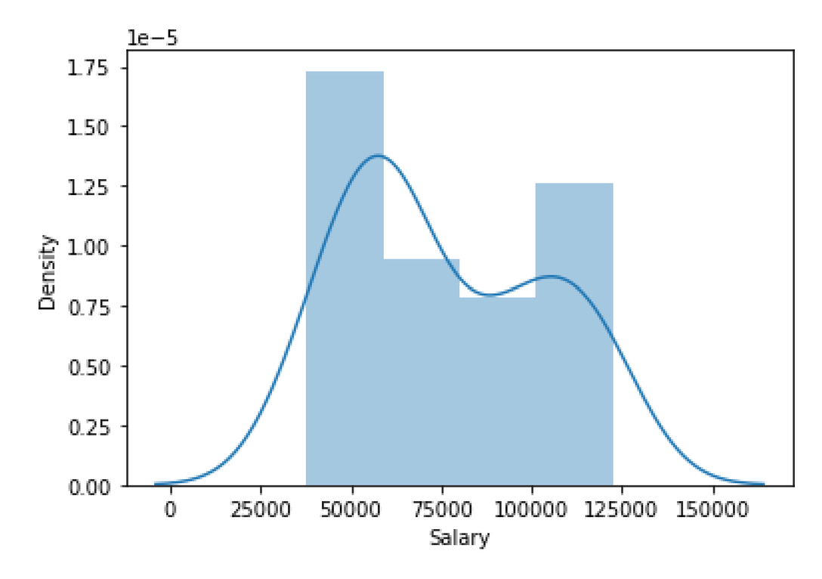
Out[87]:
```
array([[<AxesSubplot:title={'center':'YearsExperience'}>,
        <AxesSubplot:title={'center':'Salary'}>]], dtype=object)
```

In [88]:
```
import statsmodels.formula.api as smf
```

In [89]:
```
sea.distplot(data1['Salary'])
```

```
C:\pooja\Anaconda3-2022.05-Windows-x8\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

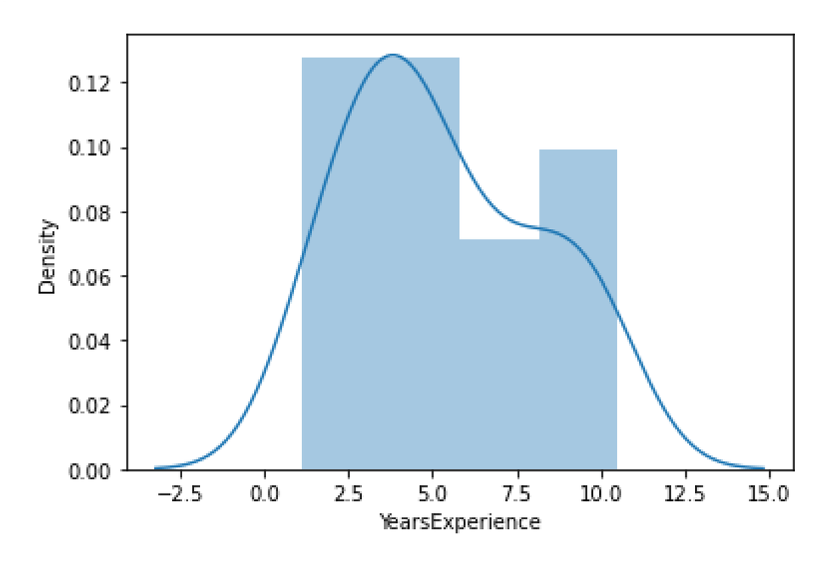
Out[89]:
```
<AxesSubplot:xlabel='Salary', ylabel='Density'>
```



In [90]:
```
sea.distplot(data1['YearsExperience'])
```

```
C:\pooja\Anaconda3-2022.05-Windows-x8\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

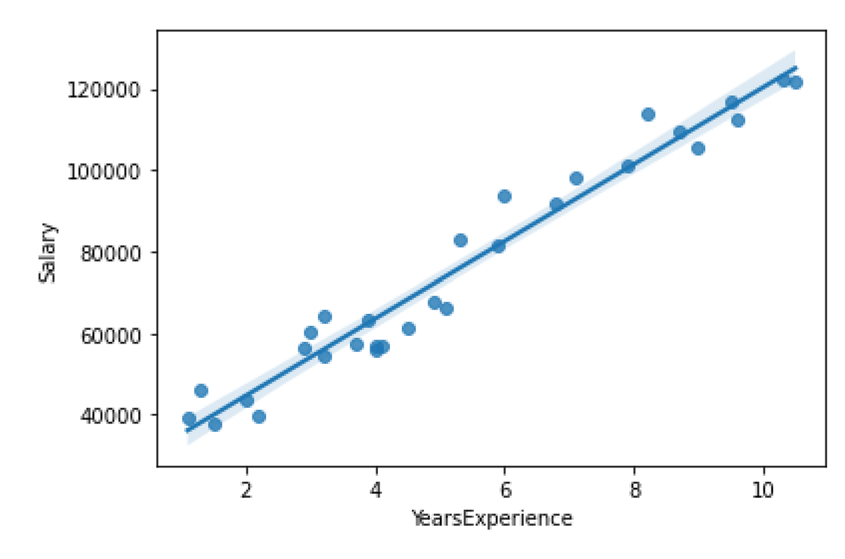
Out[90]:
```
<AxesSubplot:xlabel='YearsExperience', ylabel='Density'>
```



In [91]:
```
model = smf.ols("Salary~YearsExperience",data =data1).fit()
```

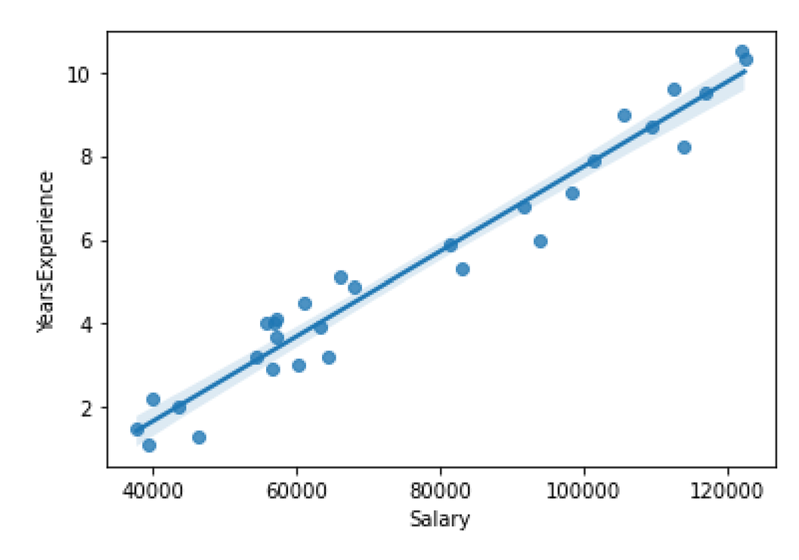
In [92]:
```
sea.regplot(x="YearsExperience",y="Salary",data=data1)
```

Out[92]:
```
<AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>
```



In [93]:
```
sea.regplot(x="Salary",y="YearsExperience",data=data1)
```

Out[93]:
```
<AxesSubplot:xlabel='Salary', ylabel='YearsExperience'>
```



In [95]:
```
(model.rsquared,model.rsquared_adj)
```

Out[95]:
```
(0.9569566641435086, 0.9554194021486339)
```

rsquared value is high but lets also use log,sqrt and sq transformation

Log

In [85]:
```
data1log = np.log (data1)
data1log. head ()
```

Out[85]:

	YearsExperience	Salary
0	0.095310	10.580073
1	0.262364	10.740843
2	0.405465	10.538237
3	0.693147	10.681091
4	0.788457	10.593906

In [96]:
```
modellog = smf.ols("Salary~YearsExperience",data =data1log).fit()
```

In [97]:
```
(modellog.rsquared,modellog.rsquared_adj)
```

Out[97]:
```
(0.9052150725817149, 0.9018298966024904)
```

Sqrt

In [98]:
```
data1sqrt= np.log(data1)
```

In [99]:
```
data1sqrt.head()
```

Out[99]:

	YearsExperience	Salary
0	0.095310	10.580073
1	0.262364	10.740843
2	0.405465	10.538237
3	0.693147	10.681091
4	0.788457	10.593906

In [100…
```
modelsqrt = smf.ols("Salary~YearsExperience",data =data1sqrt).fit()
```

In [102…
```
(modelsqrt.rsquared,modelsqrt.rsquared_adj)
```

Out[102]:
```
(0.9052150725817149, 0.9018298966024904)
```

Sq

In [103…
```
data1sq = np.square(data1)
```

In [104…
```
data1sq.head()
```

Out[104]:

	YearsExperience	Salary
0	1.21	1547871649
1	1.69	2134902025
2	2.25	1423628361
3	4.00	1894425625
4	4.84	1591291881

In [105…
```
modelsq = smf.ols("Salary~YearsExperience",data =data1sq).fit()
```

In [106…
```
(modelsq.rsquared,modelsq.rsquared_adj)
```

Out[106]:
```
(0.9540880842110778, 0.9524483729329021)
```

Predict New Data

In [107…
```
newPoint = pd.Series([12,13])
```

In [108…
```
prediction = pd.DataFrame(newPoint,columns=['YearsExperience'])
```

In [109…
```
model.predict(prediction)
```

Out[109]:
```
0    139191.748056
1    148641.710378
dtype: float64
```

In [110…
```
newPoint2 = pd.Series([32,43])
```

In [111…
```
prediction2 = pd.DataFrame(newPoint2,columns=['YearsExperience'])
```

In [112…
```
model.predict(prediction)
```

Out[112]:
```
0    139191.748056
1    148641.710378
dtype: float64
```

In [ ]: