In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
from sklearn.linear_model import LinearRegression
```

In [5]:
```
from sklearn.model_selection import train_test_split
```

In [6]:
```
sal_df=pd.read_csv("Salary_Data.csv")
```

In [7]:
```
sal_df.columns=["Exp","Salary"]
```

In [8]:
```
sal_df
```

Out[8]:

	Exp	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0

	Exp	Salary
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [9]:
```
x=sal_df[["Exp"]]
```

In [10]:
```
y=sal_df[["Salary"]]
```

In [11]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
```

In [12]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(24, 1)
(6, 1)
```

In [13]:
```
print(y_train.shape)
print(y_test.shape)
```

```
(24, 1)
(6, 1)
```

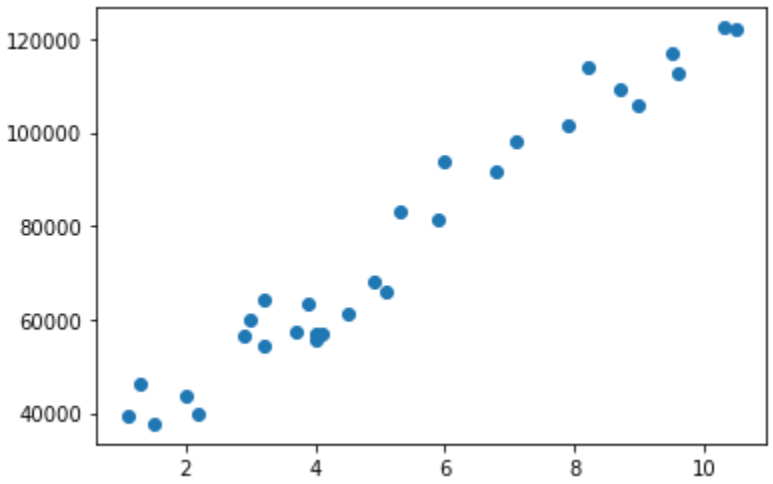
In [14]:
```
lreg=LinearRegression()
```

In [15]:
```
lreg.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
plt.scatter(x,y)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x27357f7e5e0>
```



In [17]:
```
lreg.predict(x_test)
```

Out[17]:
```
array([[ 92330.69987324],
       [ 51983.37215718],
       [116346.96637089],
       [ 80802.89195437],
       [ 54865.3241369 ],
       [ 75038.98799493]])
```

In [18]:
```
y_test
```

Out[18]:

	Salary
21	98273.0
5	56642.0
27	112635.0
18	81363.0
8	64445.0
17	83088.0

In [19]:
```
lreg.score(x_test,y_test)
```

Out[19]:
```
0.8942487177913169
```

In [20]:
```
plt.scatter(x,y)
plt.plot(x_train,lreg.predict(x_train),color="yellow")
plt.xlabel("Experience")
plt.ylabel("Salary")
plt.title("Employe Salary Calculation using experience")
```

Out[20]:
```
Text(0.5, 1.0, 'Employe Salary Calculation using experience')
```

In [21]:
```
lreg.predict(x_train)
```

Out[21]:
```
array([[ 59668.57743643],
       [107701.11043174],
       [ 62550.52941615],
       [102897.8571322 ],
       [ 89448.74789352],
       [ 67353.78271568],
       [ 61589.87875624],
       [ 73117.68667512],
       [ 54865.3241369 ],
       [ 36612.96159869],
       [123071.52099023],
       [ 34691.66027888],
       [ 43337.51621803],
       [ 52944.02281709],
       [ 71196.38535531],
       [100015.90515249],
       [124992.82231004],
       [ 62550.52941615],
       [ 38534.2629185 ],
       [ 81763.54261427],
       [ 3007606],
```

```
       [ 45258.81753784],
       [110583.06241145],
       [115386.31571098]])
```

In [22]:
```
x_train
```

Out[22]:

	Exp
9	3.7
24	8.7
12	4.0
23	8.2
20	6.8
14	4.5
10	3.9
16	5.1
7	3.2
1	1.3
28	10.3
0	1.1
3	2.0
6	3.0
15	4.9
22	7.9
29	10.5
11	4.0
2	1.5
19	6.0
13	4.1
4	2.2
25	9.0
26	9.5

In [23]:
```
#MULTIPLE LINEAR REGRESSION
#multiple independent variable to predict one dependent variable
```

In [51]:
```
cars=pd.read_csv("cars.csv")
```

In [52]:
```
cars
```

Out[52]:

	age	gender	miles	debt	income	sales
0	28	0	23	0	4099	620
1	26	0	27	0	2677	1792
2	30	1	58	41576	6215	27754
3	26	1	25	43172	7626	28256
4	20	1	17	6979	8071	4438

	age	gender	miles	debt	income	sales
...	...	...	...	...	...	...
958	22	0	11	8778	9829	1593
959	19	1	23	4850	3470	4742
960	28	1	28	9312	2720	12771
961	50	0	29	51343	8713	28511
962	47	1	15	3735	6406	6104

963 rows × 6 columns

In [53]:
```
X=cars[["miles","debt","income"]]
```

In [55]:
```
Y=cars[["sales"]]
```

In [56]:
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,random_state=21)
```

In [57]:
```
lin_reg=LinearRegression()
```

In [58]:
```
lin_reg.fit(x_train,y_train)
```

Out[58]:
```
LinearRegression()
```

In [34]:
```
lin_reg.predict(x_test)
```

Out[34]:
```
array([ 6374.47080365, 25770.93492397,  5900.88219627,  5172.35091147,
        6559.21901668, 11745.26967607,  2029.82195813,  5187.44302776,
       14851.0623798 ,  2675.11960469, 11631.21847606, 23490.42989488,
        9485.98449145,  8966.29976434, 22644.48102562, 23415.14527812,
       10479.23428053,  6271.21363319,  3903.40497186,  2523.09405949,
        8236.12861707,  4392.52141066, 29507.79115517, 26429.51797671,
        8275.09098141,  7008.65386709,  5858.87520098,  8774.74015667,
       12015.43673334, 10391.98500916,  2095.20769517,  3883.41181981,
        7214.6417917 , 18806.65950053, 20429.27219053,  5999.48628362,
       22681.57486136, 13107.44864025, 14674.0495577 ,  4460.7900043 ,
        6062.21537343, 13592.50160828,  4698.40334264,  5060.47153311,
        5308.02200697, 24861.90055235, 10652.81628488, 13437.20341003,
        3304.44438433,  6742.18366845, 18013.14337962, 12625.527802  ,
       14102.85083496,  5257.36075421,  4249.48702605,  1902.76954792,
        2915.00001846,  8273.69143149,  5271.29893123,  4367.12139183,
        8029.42930681,  3571.210485  ,  2167.07759764,  8412.16631606,
        2103.3770134 ,  8325.16160911,  3693.0505674 , 18439.27813958,
       26783.68325733, 17645.02405719,  2710.82706037,  2760.06291482,
       10163.76150175,  4777.44008363, 19170.32466714,  3595.07488792,
       21040.33516743, 11829.59351063,  7415.70934794,  3016.37450024,
        4038.77571342,  3878.68114192, 13665.028293  , 11194.43182439,
       22759.46495088, 22443.64656149, 25036.25416773,  6077.41050071,
        4117.59627139,  5708.20423746,  3176.07545145,  9524.09333959,
       10293.99855101,  3346.32638639,  4760.73021104,  9355.48952335,
        9789.56386966,  4446.99594215,  4021.76017985,  1279.19663323,
        3081.79358588,  5992.44833992,  1869.2557312 ,  8294.2638403 ,
        6330.23906345, 22003.48109797,  6111.17892967, 14100.03648321,
        7526.85621175,  9034.70372811,  7936.77345724,  4664.72708597,
       10963.22781465,  1785.80804474,  5630.57376777, 17130.96227396,
        5110.47267182, 29629.4575674 ,  6785.43697178, 26576.47445776,
        8588.02433515, 23401.11631321, 14853.57972875,  4533.29217186,
        6248.60820209,  6810.95400734,  3142.78034211, 10315.69004926,
       16929.81513474, 10200.2770011 ,  8716.68856269, 11936.87949868,
       11880.20868887, 12033.72778478, 28000.75989925, 11808.25458004,
        4144.06865875,  7476.08291111, 10109.74539714,  5434.04530847,
        [illegible]5993, 11518.2334884 ,  7413.28829589,  7216.89508797,
```

```
24324.9122422 ,  8865.30811866, 10295.21740324,  3411.6014385 ,
23761.97232845,  2208.61670085,  3532.61122253,  8428.9929788 ,
 2942.19508724, 20647.02121636,  8188.36689969,  7651.31943693,
23797.02525018,  9292.94573196,  1659.15519186, 13409.57681585,
 8723.418661  ,  4739.00254068, 11195.89112891,  5751.57194482,
 7190.09562008,  5691.95981242, 31808.30301983,  7319.66708216,
  645.93236884,  6041.11504175, 28106.71687603,  3357.12935167,
 9000.12854797,  4606.58535722,  8123.36841894,  3979.51854373,
14315.54541899, 13437.62146064,  5301.82051308,  3842.11680846,
11229.41905582, 30829.59506831,  3523.70834333, 12491.07371056,
 9042.3843829 , 20616.53139731, 13544.44451654, 13482.96577939,
 3934.01907266, 25763.3721659 ,  7438.14306684,  9854.96776359,
 4600.72190637, 10403.55160733,  3075.69548898,  7109.20831238,
10681.26963858,  7830.51745953, 25492.17727631,  1754.39290143,
 4813.51170336, 28981.21774134,  1906.3042285 , 12250.92467003,
22977.45035981,  8615.62868864,  6528.47832239,  2419.07230913,
 8028.21050404, 10984.67472362, 25055.87470687,  8351.25775454,
 7960.61805355, 22209.23533591, 27390.52043527, 10454.96262699,
24938.37115453, 22309.67015101,  6833.04055424, 11903.95519701,
 8546.51974373,  7523.81822249, 12605.89913264, 32353.59656343,
 4984.82160473,  8390.25674415, 12910.51642107, 12455.70239493,
18726.72124647, 21864.13576323,  7394.6648624 , 13920.28590216,
27325.97895826,  9959.82935364,  6160.26245724,  6951.65459442,
28568.74217217,  5756.22622106,  4144.38384523, 13174.32261738,
23852.84505727])
```

In [59]:
```
lin_reg.score(x_test,y_test)
```

Out[59]:
```
0.8149378031364339
```

In [39]:
```
from sklearn.preprocessing import StandardScaler
```

In [40]:
```
sc=StandardScaler()
```

In [41]:
```
x_train_sc=sc.fit_transform(x_train)
x_test_sc=sc.transform(x_test)
```

In [42]:
```
lin_reg.fit(x_train_sc,y_train)
```

Out[42]:
```
LinearRegression()
```

In [67]:
```
plt.scatter(X["debt"],Y)
plt.plot(x_train,lin_reg.predict(x_train),color="red")
```

Out[67]:
```
[<matplotlib.lines.Line2D at 0x273599cd3a0>,
 <matplotlib.lines.Line2D at 0x273599cd4f0>,
 <matplotlib.lines.Line2D at 0x273599cd5b0>]
```