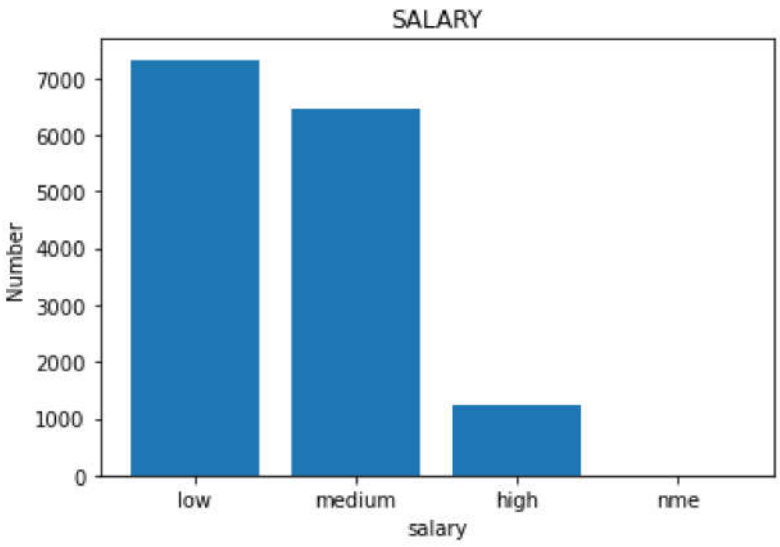
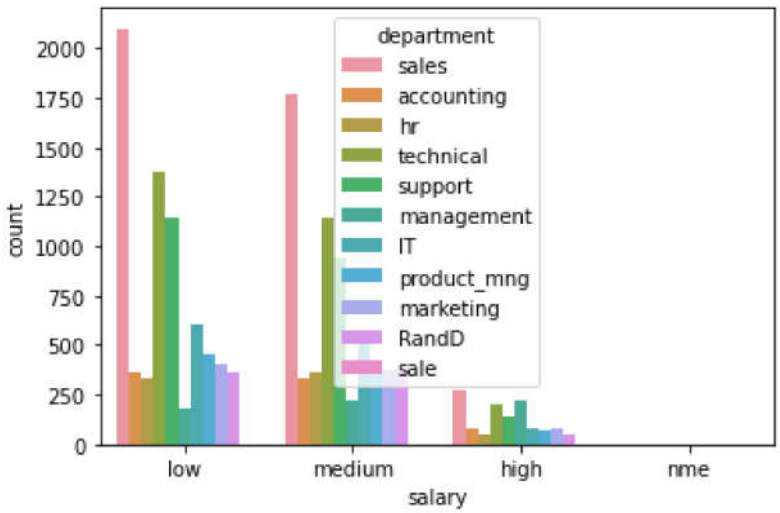
In [98]:

```
import numpy as np
import pandas as pd
# import keras
# import matplotlib as mpl
import scipy.stats as ss

import seaborn as sns
import matplotlib.pyplot as plt
```

In [99]:

```
df = pd.read_csv("./dataset/HR.csv")
plt.title("SALARY")
plt.xlabel("salary")
plt.ylabel("Number")
plt.xticks(np.arange(len(df["salary"].value_counts())),df["salary"].value_counts().index)
plt.bar(np.arange(len(df["salary"].value_counts())),df["salary"].value_counts())
plt.show()

sns.countplot(x="salary", hue="department", data=df)  ## 多层分割绘图
plt.show()
```





In [6]:

```
## 生成一个符合标准正态分布的数组
```

In [7]:

```
norm_dist = ss.norm.rvs(size=20)
```

In [8]:

```
norm_dist
```

Out[8]:

```
array([-0.02295821,  0.44001665, -0.64008911,  1.16467676,  2.36596981,
        0.24677982, -0.40780714,  0.40054582,  0.25623135,  0.38030889,
        0.07794774, -1.34435049,  1.00800101, -0.74548174, -0.59672363,
       -0.85147454, -0.31272703,  0.64052648, -0.79551208, -0.29350523])
```

In [9]:

```
## 检测一组数是否符合正态分布
```

In [10]:

```
ss.normaltest(norm_dist) #使用了基于偏度和峰度的检验方法
```

Out[10]:

```
NormaltestResult(statistic=5.415961873407559, pvalue=0.06667128445315099)
```

In [11]:

```
#(若显著性水平定为0.05 则可以确定该分布符合正态分布 pvalue>0.05)
```

In [ ]:

```

```

In [12]:

```
## 卡方检验
```

In [15]:

```
ss.chi2_contingency([[15,95],[85,5]]) # 两条数据（行）(得到 检验统计量 p值 自由度 理论分布)
```

Out[15]:

```
(126.08080808080808,
 2.9521414005078985e-29,
 1,
 array([[55., 55.],
        [45., 45.]]))
```

In [16]:

```
## 独立t分布检验  检验两组数据的均值有无较大差异性
```

In [18]:

```
ss.ttest_ind(ss.norm.rvs(size=10),ss.norm.rvs(size=20))
```

Out[18]:

```
Ttest_indResult(statistic=-0.7966541855366688, pvalue=0.432355312343095)
```

In [19]:

```
ss.ttest_ind(ss.norm.rvs(size=100),ss.norm.rvs(size=200))
```

Out[19]:

```
Ttest_indResult(statistic=-0.5462410288340069, pvalue=0.5853089823067392)
```

In [ ]:

```

```

In [20]:

```
## 方差检验
```

In [21]:

```
ss.f_oneway([49,50,39,40,43],[28,32,30,26,34],[38,40,45,42,48])
```

Out[21]:

```
F_onewayResult(statistic=17.619417475728156, pvalue=0.0002687153079821641)
```

In [ ]:

```

```

In [25]:

```
## qq图：通过比较两个概率分布的分位数对这两个概率分布进行比较的概率图方法。
```

In [ ]:

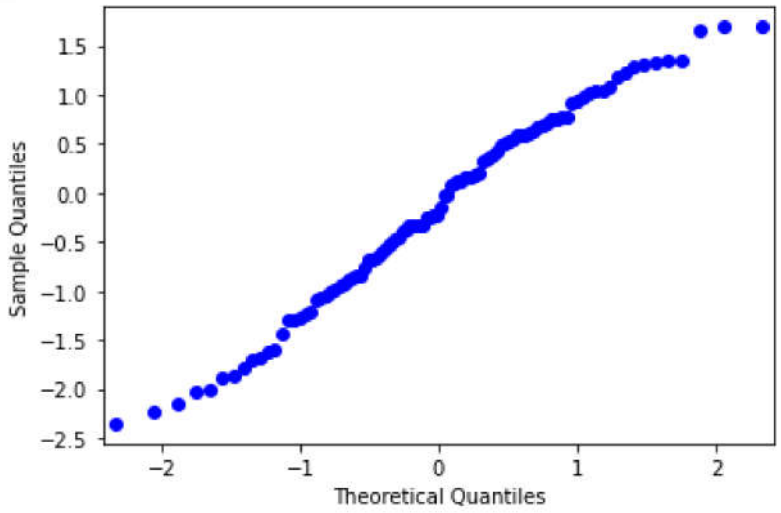
```
## 参考：https://zhuanlan.zhihu.com/p/232271603
```

In [24]:

```
from statsmodels.graphics.api import qqplot
```

In [27]:

```
plt.show(qqplot(ss.norm.rvs(size=100)))  ## qqplot默认判断一个分布是否符合正态分布
```



In [ ]:

```

```

In [28]:

```
## 相关系数
```

In [29]:

```
s1=pd.Series([0.1,0.2,1.1,2.4,1.3,0.3,0.5])
s2=pd.Series([0.5,0.4,1.2,2.5,1.1,0.7,0.1])
```

In [30]:

```
s1.corr(s2)
```

Out[30]:

```
0.9333729600465923
```

In [31]:

```
s1.corr(s2,method="spearman")
```

Out[31]:

```
0.7142857142857144
```

In [32]:

```
df=pd.DataFrame([s1,s2])
```

In [33]:

```
df.corr()  ## df的相关系数是按列计算的
```

Out[33]:

	0	1	2	3	4	5	6
0	1.0	1.0	1.0	1.0	-1.0	1.0	-1.0
1	1.0	1.0	1.0	1.0	-1.0	1.0	-1.0
2	1.0	1.0	1.0	1.0	-1.0	1.0	-1.0
3	1.0	1.0	1.0	1.0	-1.0	1.0	-1.0
4	-1.0	-1.0	-1.0	-1.0	1.0	-1.0	1.0
5	1.0	1.0	1.0	1.0	-1.0	1.0	-1.0
6	-1.0	-1.0	-1.0	-1.0	1.0	-1.0	1.0

In [35]:

```
df2 = pd.DataFrame(np.array([s1,s2]).T)
```

In [36]:

```
df2
```

Out[36]:

	0	1
0	0.1	0.5
1	0.2	0.4
2	1.1	1.2
3	2.4	2.5
4	1.3	1.1
5	0.3	0.7
6	0.5	0.1

In [37]:

```
df2.corr()
```

Out[37]:

	0	1
0	1.000000	0.933373
1	0.933373	1.000000

In [39]:

```
df2.corr(method="spearman")
```

Out[39]:

	0	1
0	1.000000	0.714286
1	0.714286	1.000000

In [ ]:

```

```

In [ ]:

```
## 回归分析
```

In [48]:

```
x=np.arange(10).astype(np.float).reshape((10,1))
x
```

Out[48]:

```
array([[0.],
       [1.],
       [2.],
       [3.],
       [4.],
       [5.],
       [6.],
       [7.],
       [8.],
       [9.]])
```

In [49]:

```
y=x*3+4+np.random.random((10,1))
y
```

Out[49]:

```
array([[ 4.53896836],
       [ 7.36634611],
       [10.20455137],
       [13.55567796],
       [16.45466897],
       [19.70599467],
       [22.44788171],
       [25.0700279 ],
       [28.98703389],
       [31.64758791]])
```

In [50]:

```
from sklearn.linear_model import LinearRegression
```

In [53]:

```
reg = LinearRegression()
```

In [54]:

```
res = reg.fit(x,y)  ## 拟合过程
```

In [58]:

```
y_pred=reg.predict(x) ## 预测值
y_pred
```

Out[58]:

```
array([[ 4.37302725],
       [ 7.40077095],
       [10.42851464],
       [13.45625834],
       [16.48400204],
       [19.51174573],
       [22.53948943],
       [25.56723313],
       [28.59497682],
       [31.62272052]])
```

In [59]:

```
reg.coef_  ## 参数
```

Out[59]:

```
array([[3.0277437]])
```

In [60]:

```
reg.intercept_  ## 截距
```

Out[60]:

```
array([4.37302725])
```

In [ ]:

```

```

In [61]:

```
## PCA变换
```

In [63]:

```
([2.5,.5,2.2,1.9,3.1,2.3,2,1,1.5,1.1]),np.array([2.4,0.7,2.9,2.2,3,2.7,1.6,1.1,1.6,0.9])]).T
```

In [64]:

```
data
```

Out[64]:

```
array([[2.5, 2.4],
       [0.5, 0.7],
       [2.2, 2.9],
       [1.9, 2.2],
       [3.1, 3. ],
       [2.3, 2.7],
       [2. , 1.6],
       [1. , 1.1],
       [1.5, 1.6],
       [1.1, 0.9]])
```

In [81]:

```
from sklearn.decomposition import PCA   ## 用的是SVD奇异值分解的算法
```

In [66]:

```
low_dim=PCA(n_components=1)
```

In [67]:

```
low_dim.fit(data)
```

Out[67]:

```
PCA(n_components=1)
```

In [68]:

```
low_dim.explained_variance_ratio_
```

Out[68]:

```
array([0.96318131])
```

In [69]:

```
low_dim.fit_transform(data)
```

Out[69]:

```
array([[-0.82797019],
       [ 1.77758033],
       [-0.99219749],
       [-0.27421042],
       [-1.67580142],
       [-0.9129491 ],
       [ 0.09910944],
       [ 1.14457216],
       [ 0.43804614],
       [ 1.22382056]])
```

In [91]:

```
## 自定义PCA方法
```

In [92]:

```
def myPCA(data, n_components=10000): ## 默认很大，若没有这么多就会全取
    mean_vals=np.mean(data,axis=0) ##按列(属性)求取均值
    mid=data-mean_vals
    cov_mat=np.cov(mid,rowvar=False)  ##按列求取协方差
    from scipy import linalg ## 引入线性计算的包
    eig_vals,eig_vects=linalg.eig(np.mat(cov_mat)) ## 求特征值和特征向量
    ## 取出最大特征值对应的特征向量
    eig_val_index = np.argsort(eig_vals)  ## argsort排序后得到的是索引
    eig_val_index=eig_val_index[:-(n_components+1):-1]
    eig_vects=eig_vects[:,eig_val_index]
    ## 矩阵乘法
    low_dim_mat = np.dot(mid,eig_vects)
    return low_dim_mat,eig_vals ## 返回降维后的数组和特征值
```

In [93]:

```
## 测试函数
```

In [94]:

```
data=np.array([np.array([2.5,.5,2.2,1.9,3.1,2.3,2,1,1.5,1.1]),np.array([2.4,0.7,2.9,2.2,3,2.7,1.6,1.
```

In [96]:

```
myPCA(data,n_components=1)
```

Out[96]:

```
(array([[-0.82797019],
        [ 1.77758033],
        [-0.99219749],
        [-0.27421042],
        [-1.67580142],
        [-0.9129491 ],
        [ 0.09910944],
        [ 1.14457216],
        [ 0.43804614],
        [ 1.22382056]]),
 array([0.0490834 +0.j, 1.28402771+0.j]))
```

In [ ]:

```

```

In [100]:

```
#Left与部门属性的t独立分布检验
dp_indices=df.groupby(by="department").indices
sales_values=df["left"].iloc[dp_indices["sales"]].values
print(sales_values)
technical_values=df["left"].iloc[dp_indices["technical"]].values
print(technical_values)
```

```
[1 1 1 ... 1 1 1]
[1 1 1 ... 1 1 1]
```

In [ ]: