Question 1

We import the file using read_csv() and utilize pandas.DataFrame library to do the computation. Note that pandas.DataFrame.var() has default N-1 freedom.

In [3]:

```
import pandas as pd
import numpy as np
import math
from scipy.spatial.distance import minkowski
from sklearn.metrics.pairwise import cosine_similarity as cos
header = ['id', 'mid_scores', 'final_scores']
data = pd.read_csv('data.online.scores.txt', delimiter = '\t', names = header)
print (data['mid_scores'].describe())
modes = data['mid_scores'].mode().values
emp_var = data['mid_scores'].var()

print ('empirical variance is {0}'.format(emp_var))
print ('modes are {0}'.format(modes))
```

```
count    1000.00000
mean       76.71500
std        13.16355
min        37.00000
25%        68.00000
50%        77.00000
75%        87.00000
max       100.00000
Name: mid_scores, dtype: float64
empirical variance is 173.27905405405397
modes are [77 83]
```

We get the following answer from the output:

(a)max = 100, min = 37

(b)first quantile = 68, median = 77, third quantile = 87

(c)mean score = 76.715

(d)mode scores = 77, 83

(e)Empirical Variance = 173.27905405405397

Question 2

(a) The empirical variance before normalization is what we get in the previous question, which equals to 173.27905405405397.
Using the formula for z-score normalization :
$z = \frac{x-\mu}{\sigma}$
and computing it in python, we get the **empirical variance after normalization = 1.0000000000000002**.

In [4]:

```
data['z_score'] = (data['mid_scores'] - data['mid_scores'].mean())/(data['mid_sc
ores'].std())

print ('the variance after normalization is {0}'.format(data['z_score'].var(ddof
= 1)))
```

```
the variance after normalization is 1.0000000000000002
```

(b) By using pandas.DataFrame.query funciton we extract the rows with midterm scores = 90

In [5]:

```
df_90 = data.query('mid_scores == 90')
print (df_90)
```

```
      id  mid_scores  final_scores   z_score
11    11          90            99  1.009226
42    42          90            79  1.009226
62    62          90            83  1.009226
82    82          90            99  1.009226
90    90          90           100  1.009226
101  101          90            92  1.009226
154  154          90            95  1.009226
157  157          90            93  1.009226
223  223          90            90  1.009226
247  247          90            77  1.009226
345  345          90            88  1.009226
351  351          90           100  1.009226
494  494          90            87  1.009226
564  564          90            76  1.009226
565  565          90           100  1.009226
574  574          90            96  1.009226
591  591          90            89  1.009226
598  598          90            79  1.009226
637  637          90            96  1.009226
803  803          90            86  1.009226
836  836          90            96  1.009226
885  885          90           100  1.009226
911  911          90           100  1.009226
927  927          90            92  1.009226
939  939          90            93  1.009226
```

We see that the corresponding score of 90 after normalization is 1.009226.

(c) In order to calculate Pearson's Correlation Coefficient between midterm scores and final scores

- we first need to find the covariance between midterm scores and final scores. Using pandas.DataFrame.cov(which has N-1 freedom by default) function, we get a table of all the correlations between the data.

In [6]:

```
print (data.cov())
```

```
                        id  mid_scores  final_scores    z_score
id            83416.666667  -51.146647     11.645646  -3.885475
mid_scores      -51.146647  173.279054     78.254194  13.163550
final_scores     11.645646   78.254194    119.232176   5.944764
z_score          -3.885475   13.163550      5.944764   1.000000
```

So we find that the covariance between midterm scores and final scores is 78.254194
Formula for PCC is:

- $\rho_{(X,Y)} = \frac{cov(X,Y)}{(\sigma_X \sigma_Y)}$

In [7]:

```
print ("Pearson's correlation coefficient between midterm scores and final score
s is {0}".format(78.254194 / (data['mid_scores'].std() * data['final_scores'].st
d())))
```

```
Pearson's correlation coefficient between midterm scores and final s
cores is 0.5444247409613782
```

(d) solved in **(c)**

Question 3

(a) The formula for Jaccard coefficient is:

- $sim(i, j) = \frac{q}{q+r+s}$
  In this case, q is 58, r is 2, s is 120. Thus, Jaccard Coefficient for CBL and CML is 0.322.

(b) The formula for minkowski distance is:

- $d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + x_{ip} - x_{jp}|^h}$
  The following code uses the scipy library which has minkowski function to compute it.

In [8]:

```
import math
from scipy.spatial.distance import minkowski
from sklearn.metrics.pairwise import cosine_similarity as cos

data = pd.read_csv('data.libraries.inventories.txt', delimiter = '\t')
CML = data.iloc[0][1:]
CBL = data.iloc[1][1:]

h_1 = minkowski(CBL, CML, 1)
h_2 = minkowski(CBL, CML, 2)
h_inf = minkowski(CBL, CML, math.inf)
print ('minkowski distance for h = 1 is {0}'.format(h_1))
print ('minkowski distance for h = 2 is {0}'.format(h_2))
print ('minkowski distance for h approaches infinity is {0}'.format(h_inf))
```

```
minkowski distance for h = 1 is 6152
minkowski distance for h = 2 is 715.3278968417211
minkowski distance for h approaches infinity is 170
```

We get:

1. **h = 1, Minkowski Distance = 6152**
2. **h = 2, Minkowski Distance = 715.3278968417211**
3. **h = infinite, Minkowski Distance =170**

(c) The formula for cosine similarity is:

- $sim(x, y) = \frac{x \cdot y}{||x||||y||}$

In [9]:

```
cos_sim = np.dot(CML, CBL) / (np.linalg.norm(CML) * np.linalg.norm(CBL))
print (cos_sim)
```

```
0.841404025662
```

We get **Cosine similarity = 0.841404025662**

(d) The formula for KL Divergence is :

- $D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) ln \frac{p(x)}{q(x)}$

We compute it as follows:

In [10]:

```
length = len(CML)
CML_sum = CML.sum()
CBL_sum = CBL.sum()
book_sum = CBL_sum + CML_sum
CML_prob = np.zeros(length,)
CBL_prob = np.zeros(length,)
for i in range(len(CML)):
    CML_prob[i] = CML[i] / CML_sum
    CBL_prob[i] = CBL[i] / CBL_sum
#calculating KL divergence:
KLD = np.sum(CML_prob*np.log(CML_prob/CBL_prob))
print (KLD)
```

```
0.207080937332
```

We get Kullback–Leibler divergence of these two libraries**P(CML || CBL) = 0.207080937332**

Question 4

The formula for chi_square test is:

- $\chi^2 = \sum_k^n \frac{(O_k - E_k)^2}{E_k}$

Using this formula and compute it in python:

In [11]:

```
buy_beer_sum = 190
no_beer_sum = 3315
buy_diaper_sum = 165
no_diaper_sum = 3340

tol_sum = buy_beer_sum  + no_beer_sum
buy_beer = np.array([150, 40])
no_beer = np.array([15, 3300])
buy_beer_exp = np.array([buy_diaper_sum * (buy_beer_sum/tol_sum), no_diaper_sum
* (buy_beer_sum/tol_sum)])
no_beer_exp = np.array([buy_diaper_sum * (no_beer_sum/tol_sum), no_diaper_sum *
(no_beer_sum/tol_sum)])
chi_square = np.sum((buy_beer - buy_beer_exp)**2 / buy_beer_exp) + np.sum((no_be
er - no_beer_exp)**2 / no_beer_exp)


print ('chi-square correlation value is {0}'.format(chi_square))
```

```
chi-square correlation value is 2468.183255909104
```

we get **chi-square correlation value** $\approx$ **2468.183**