# Assignment 2: Yue, Shenghua ¶

In [1]:
```
import pyspark
```

In [2]:
```
# read input data
input_path = "ratings.txt"
records = spark.sparkContext.textFile(input_path)
print(records.count())
records.collect()
```

```
20
```

Out[2]:
```
['U1,M4,4',
 'U1,M4,3',
 'U1,M2,5',
 'U1,M2,0',
 'U1,M3,2',
 'U2,M4,3',
 'U2,M4,4',
 'U2,M4,5',
 'U3,M1,1',
 'U3,M5,6',
 'U3,M4,4',
 'U3,M4,5',
 'U4,M2,3',
 'U4,M1,1',
 'U4,M1,4',
 'U4,M1,5',
 'U5,M1,3',
 'U5,M1,1',
 'U6,M1,3',
 'U6,M9,4']
```

In [3]:
```
records_map = records.map(lambda x: x.split(","))
print(records_map.count())
records_map.collect()
```

```
20
```

Out[3]:
```
[['U1', 'M4', '4'],
 ['U1', 'M4', '3'],
 ['U1', 'M2', '5'],
 ['U1', 'M2', '0'],
 ['U1', 'M3', '2'],
 ['U2', 'M4', '3'],
 ['U2', 'M4', '4'],
 ['U2', 'M4', '5'],
 ['U3', 'M1', '1'],
 ['U3', 'M5', '6'],
 ['U3', 'M4', '4'],
 ['U3', 'M4', '5'],
 ['U4', 'M2', '3'],
 ['U4', 'M1', '1'],
 ['U4', 'M1', '4'],
 ['U4', 'M1', '5'],
 ['U5', 'M1', '3'],
 ['U5', 'M1', '1'],
 ['U6', 'M1', '3'],
 ['U6', 'M9', '4']]
```

```
In  [4]: # filter out the invalid ratings
         records_filter = records_map.filter(lambda x: int(x[2]) > 0 and int(x[2]) < 6)
         print(records_filter.count())
         records_filter.collect()

         18

Out[4]: [['U1', 'M4', '4'],
         ['U1', 'M4', '3'],
         ['U1', 'M2', '5'],
         ['U1', 'M3', '2'],
         ['U2', 'M4', '3'],
         ['U2', 'M4', '4'],
         ['U2', 'M4', '5'],
         ['U3', 'M1', '1'],
         ['U3', 'M4', '4'],
         ['U3', 'M4', '5'],
         ['U4', 'M2', '3'],
         ['U4', 'M1', '1'],
         ['U4', 'M1', '4'],
         ['U4', 'M1', '5'],
         ['U5', 'M1', '3'],
         ['U5', 'M1', '1'],
         ['U6', 'M1', '3'],
         ['U6', 'M9', '4']]

In  [5]: # mapper output as (key, value) pairs
         records_pairs = records_filter.map(lambda x: (x[1], x[0]))
         print(records_pairs.count())
         records_pairs.collect()

         18

Out[5]: [('M4', 'U1'),
         ('M4', 'U1'),
         ('M2', 'U1'),
         ('M3', 'U1'),
         ('M4', 'U2'),
         ('M4', 'U2'),
         ('M4', 'U2'),
         ('M1', 'U3'),
         ('M4', 'U3'),
         ('M4', 'U3'),
         ('M2', 'U4'),
         ('M1', 'U4'),
         ('M1', 'U4'),
         ('M1', 'U4'),
         ('M1', 'U5'),
         ('M1', 'U5'),
         ('M1', 'U6'),
         ('M9', 'U6')]

In  [6]: reduce_input = records_pairs.groupByKey().map(lambda x: (x[0],list(x[1])))
         print(reduce_input.count())
         reduce_input.collect()

         5

Out[6]: [('M4', ['U1', 'U1', 'U2', 'U2', 'U2', 'U3', 'U3']),
         ('M2', ['U1', 'U4']),
         ('M3', ['U1']),
         ('M1', ['U3', 'U4', 'U4', 'U4', 'U5', 'U5', 'U6']),
         ('M9', ['U6'])]

In  [8]: # filter out total number of raters less than 2
         reduce_filter = reduce_input.filter(lambda x: len(x[1])>=2)
         print(reduce_filter.count())
         reduce_filter.collect()

         3

Out[8]: [('M4', ['U1', 'U1', 'U2', 'U2', 'U2', 'U3', 'U3']),
         ('M2', ['U1', 'U4']),
         ('M1', ['U3', 'U4', 'U4', 'U4', 'U5', 'U5', 'U6'])]

In  [9]: # reducer output
         reduce_output = reduce_filter.map(lambda x: (x[0], (len(x[1]), len(set(x[1])))))
         print(reduce_output.count())
         reduce_output.collect()

         3

Out[9]: [('M4', (7, 3)), ('M2', (2, 2)), ('M1', (7, 4))]

In  [ ]:
```