2.

a. Not exactly following here… Is this just asking for the pseudo code effectively in each loop?

M = RDD of (row, (column, value))

v = RDD of (row, value)

mmult = join v to M

flat Map to get (row, M\_value \* v\_value)

reduceByKey (add up values)

x = M.join(v).flatMap(lambda (row,((column,M\_value),v\_value)): (row, (M\_value\*v\_value)).reduceByKey()

b. A stage is a unit of execution. It consists of a set of parallel tasks with 1 task per partition.

Stages are operations that can run on the same data partitioning in parallel across executors/nodes

Tasks within a stage are operations executed by one executor/node that are pipelined together

Stage1: join, flatMap Stage2: reduceByKey

c.

M = (1, (1, 16)), (1, (2, 2)), (1, (3, 3)), (1, (4, 13));

(2, (1, 5)), (2, (2, 11)), (2, (3, 10)), (2, (4, 8));

(3, (1, 9)), (3, (2, 7)), (3, (3, 6)), (3, (4, 12));

(4, (1, 4)), (4, (2, 14)), (4, (3, 15)), (4, (4, 1));

v = (1, 1), (2, 2), (3, 3), (4, 4)

d.

join: (1, [(1,16),1]), (1,[(2,2),1]), (1,[(3,3),1]),(1,[(4,13),1]);

(2,[(1,5),2]), (2,[(2,11),2]), (2,[(3,10),2]), (2,[(4,8),2]);

(3,[(1,9),3]), (3,[(2,7),3]), (3,[(3,6),3]), (3,[(4,12),3]);

(4,[(1,4),4]), (4,[(2,14),4]), (4,[(3,15),4]), (4,[(4,1),4]);

flatMap: (1,16),(1,2),(1,3),(1,13);

(2,10),(2,22),(2,20),(2,16);

(3,27),(3,21),(3,18),(3,36);

(4,16),(4,56),(4,60),(4,4);

reduceByKey: (1,34);

(2,68);

(3,102);

(4,136);

e. (m \* n) + n

Input = (m \* n) + n

Join = (m\*2n)

flatMap = (m\*n)

reduceByKey = (n)

Total = 2(m\*n)+ (m\*2n)+2n

f. A potential memory bottlekneck could occur during the repartition between the flatMap and the reduceByKey.