Solution to COMP9318 Assignment 1

Q1

1. See the following table.

cuboid	*Location*	*Time*	*Item*	$SUM(Quantity)$
LTI	Sydney	2005	PS2	1400
LTI	Sydney	2006	PS2	1500
LTI	Sydney	2006	Wii	500
LTI	Melbourne	2005	XBox 360	1700
LT	Sydney	2005	ALL	1400
LT	Sydney	2006	ALL	2000
LT	Melbourne	2005	ALL	1700
LI	Sydney	ALL	PS2	2900
LI	Sydney	ALL	Wii	500
LI	Melbourne	ALL	XBox 360	1700
TI	ALL	2005	PS2	1400
TI	ALL	2006	PS2	1500
TI	ALL	2006	Wii	500
TI	ALL	2005	XBox 360	1700
L	Sydney	ALL	ALL	3400
L	Melbourne	ALL	ALL	1700
T	ALL	2005	ALL	3100
T	ALL	2006	ALL	2000
I	ALL	ALL	PS2	2900
I	ALL	ALL	Wii	500
I	ALL	ALL	XBox 360	1700
	ALL	ALL	ALL	5100

2. See below. Note that we

   (a) need to ensure all `SELECT` has the same schema (called "Union compatible"), and

   (b) should use `UNION ALL` as there is no duplicate across different `SELECT` queries.

```
SELECT   L, T, I, SUM(M)
FROM     R
GROUP BY L, T, I
UNION ALL
SELECT   L, T, ALL, SUM(M)
FROM     R
GROUP BY L, T
UNION ALL
SELECT   L, ALL, I, SUM(M)
FROM     R
GROUP BY L, I
UNION ALL
SELECT   ALL, T, I, SUM(M)
FROM     R
GROUP BY T, I
UNION ALL
SELECT   L, ALL, ALL, SUM(M)
FROM     R
GROUP BY L
UNION ALL
SELECT   ALL, T, ALL, SUM(M)
FROM     R
GROUP BY T
UNION ALL
SELECT   ALL, ALL, I, SUM(M)
FROM     R
GROUP BY I
UNION ALL
SELECT   ALL, ALL, ALL, SUM(M)
FROM     R
```

3. The iceberg cube is

cuboid	*Location*	*Time*	*Item*	$SUM(Quantity)$
LT	Sydney	2006	ALL	2000
LI	Sydney	ALL	PS2	2900
L	Sydney	ALL	ALL	3400
T	ALL	2005	ALL	3100
T	ALL	2006	ALL	2000
I	ALL	ALL	PS2	2900
	ALL	ALL	ALL	5100

4. The mapping function we choose should satisfy the property that it is a one-to-one function (such that we can always recover the original

value even after the mapping). The simplest form is $h(L,T,I) = 12L + 4T + I$. Hence,

Location	*Time*	*Item*	$SUM(Quantity)$	$h(L,T,I)$
1	1	1	1400	17
1	2	1	1500	21
1	2	3	500	23
2	1	2	1700	30
1	1	0	1400	16
1	2	0	2000	20
2	1	0	1700	28
1	0	1	2900	13
1	0	3	500	15
2	0	2	1700	26
0	1	1	1400	5
0	2	1	1500	9
0	2	3	500	11
0	1	2	1700	6
1	0	0	3400	12
2	0	0	1700	24
0	1	0	3100	4
0	2	0	2000	8
0	0	1	2900	1
0	0	3	500	3
0	0	2	1700	2
0	0	0	5100	0

So the final result is:

index	*value*
17	1400
21	1500
23	500
30	1700
16	1400
20	2000
28	1700
13	2900
15	500
26	1700
5	1400
9	1500
11	500
6	1700
12	3400
24	1700
4	3100
8	2000
1	2900
3	500
2	1700
0	5100

Q2

1. Original similarity matrix can be rewritten into

	p_1	p_2	p_3	p_4	p_5
p_1		0.10	0.41	0.55	0.35
p_2			0.64	0.47	0.98
p_3				0.44	0.85
p_4					0.76
p_5					

2. The largest similarity is 0.98. Thus we merge p_2 and p_5 into p_{25}. Update the matrix as

	p_1	p_{25}	p_3	p_4
p_1		0.225	0.410	0.550
p_{25}			0.745	0.615
p_3				0.440
p_4				

3. The largest similarity is 0.745. Thus we merge p_{25} and p_3 into p_{235}. Update the matrix as

	p_1	p_{235}	p_4
p_1		0.287	0.550
p_{235}			0.557
p_4			

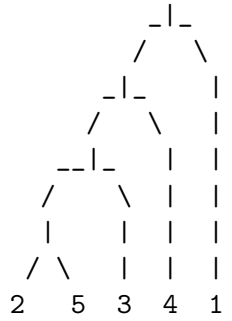
4. The largest similarity is 0.557. Thus we merge p_{235} and p_4 into p_{2345}. Update the matrix as

	p_1	p_{2345}
p_1		0.352
p_{2345}		

(the matrix is optional here)

5. Finally, we merge all into one cluster.

Therefore the final result is:



Q3

(1) See Algorithm 1.

(2) It is obvious that we only need to prove the same conclusion for a cost function y that sums up square of the $dist$, i.e.,

$$f = cost'(g_1, \ldots, g_k) = \sum_{i=1}^{k} cost'(g_i)$$
$$= \sum_{i=1}^{k} \left(\sum_{p \in g_i} (dist(p, c_i))^2 \right)$$

Algorithm 1: k-means(D, k)

Data: D is a dataset of n d-dimensional points; k is the number of clusters.

```
Initialize k centers C = [c_1, c_2, ..., c_k];
canStop ← false;
while canStop = false do
    Initialize k empty clusters G = [g_1, g_2, ..., g_k];
    for each data point p ∈ D do
        c_x ← NearestCenter(p, C);
        g_{c_x}.append(p);
    canStop ← true;
    for each group g ∈ G do
        old_c_i ← c_i;
        c_i ← ComputeCenter(g);
        if old_c_i ≠ c_i then
            canStop ← false;
return G;
```

Within each iteration, we first assign each p to its nearest center, and then update the centers. It is easy to see first step always reduces f. For the second step, we study the extreme value of f. It is obvious that we can study $cost'(g_i)$ individually. Assume in two dimensional space, c_i is represented as (x, y), then we take partial derivatives

$$\begin{aligned}\frac{\partial cost'(g_i)}{\partial x} &= \frac{\sum_{p \in g_i}(p.x - x)^2 + (p.y - y)^2}{\partial x} \\ &= -\sum_{p \in g_i} 2(p.x - x)\end{aligned}$$

We can obtain similar results on y.

Hence $x = \frac{\sum_{p \in g_i} p.x}{|g_i|}$ (similar for y) achieve the minimum value, which is exactly the new centers chosen at the end of each iteration.

(3) Assume the conclusion in (2). This says the cost f at the end of each iteration is monotonically decreasing. Obviously f has a lower bound of 0. Therefore, according to the "monotone convergence theorem", the cost will converge.