**Q1**

(1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Location** | **Time** | **Item** | **SUM(Quality)** |
| 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)

SELECT L, T, I, SUM(Quantity) FROM Sales

GROUP BY L, T, I

UNION ALL

SELECT L, T, ALL, SUM(Quantity)

FROM Sales

GROUP BY L, T

UNION ALL

SELECT L, ALL, I, SUM(Quantity)

FROM Sales

GROUP BY L, I

UNION ALL

SELECT ALL, T, I, SUM(Quantity)

FROM Sales

GROUP BY T, I

UNION ALL

SELECT L, ALL, ALL, SUM(Quantity)

FROM Sales

GROUP BY L

UNION ALL

SELECT ALL, T, ALL, SUM(Quantity)

FROM Sales

GROUP BY T

UNION ALL

SELECT ALL, ALL, I, SUM(Quantity)

FROM Sales

GROUP BY I

UNION ALL

SELECT ALL, ALL, ALL, SUM(Quantity)

FROM Sales

(3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **cuboid** | **Location** | **Time** | **Item** | **SUM(Quality)** |
| 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)

***f(x) = 16 · fLocation(x) + 4 · fT ime(x) + fItem(x)*** is more feasible

Because there are three types of *time* and four types of *item*, when doing the injective mapping, the coefficient of location should be at least 12 and the coefficient of time should be 4 to ensure that the offset after mapping will not be repeated, which means that the combination of dimensions mapped to offset is a one-to-one function. Therefore, only f(x) written above is a one-to-one function.

Firstly, we get a cube after mapping:

|  |  |  |  |
| --- | --- | --- | --- |
| **Location** | **Time** | **Item** | **SUM(Quality)** |
| 1 | 1 | 1 | 1400 |
| 1 | 2 | 1 | 1500 |
| 1 | 2 | 3 | 500 |
| 2 | 1 | 2 | 1700 |
| 1 | 1 | 0 | 1400 |
| 1 | 2 | 0 | 2000 |
| 2 | 1 | 0 | 1700 |
| 1 | 0 | 1 | 2900 |
| 1 | 0 | 3 | 500 |
| 2 | 0 | 2 | 1700 |
| 0 | 1 | 1 | 1400 |
| 0 | 2 | 1 | 1500 |
| 0 | 2 | 3 | 500 |
| 0 | 1 | 2 | 1700 |
| 1 | 0 | 0 | 3400 |
| 2 | 0 | 0 | 1700 |
| 0 | 1 | 0 | 3100 |
| 0 | 2 | 0 | 2000 |
| 0 | 0 | 1 | 2900 |
| 0 | 0 | 3 | 500 |
| 0 | 0 | 2 | 1700 |
| 0 | 0 | 0 | 5100 |

Finally, we get the MOLAP cube:

|  |  |
| --- | --- |
| **offset** | **SUM(Quality)** |
| 21 | 1400 |
| 25 | 1500 |
| 27 | 500 |
| 38 | 1700 |
| 20 | 1400 |
| 24 | 2000 |
| 36 | 1700 |
| 17 | 2900 |
| 19 | 500 |
| 34 | 1700 |
| 5 | 1400 |
| 9 | 1500 |
| 11 | 500 |
| 6 | 1700 |
| 16 | 3400 |
| 32 | 1700 |
| 4 | 3100 |
| 8 | 2000 |
| 1 | 2900 |
| 3 | 500 |
| 2 | 1700 |
| 0 | 5100 |

**Q2**

(1)

Firstly, we calculate the gini index for each feature(Gender, Smokes, Chest pain, Cough):

*Gender*

gini(Gender = Male) =

gini(Gender = Female) =

Therefore, ginisplit(Gender) =

*Smokes?*

gini(Smokes = Yes) =

gini(Smokes = No) =

Therefore, ginisplit(Smokes) =

*Chest painr?*

gini(Chest pain = Yes) =

gini(Chest pain = No) =

Therefore, ginisplit(Chest pain) =

*Cough?*

gini(Cough = Yes) =

gini(Cough = No) =

Therefore, ginisplit(Cough) =

*Decide*

|  |  |
| --- | --- |
| **Feature** | **ginisplit index** |
| *Gender* | 0.417 |
| *Smokes?* | 0.222 |
| *Chest pain?* | 0.333 |
| *Cough?* | 0.333 |

We have calculated gini index values for each feature. We will choose the smallest value which is *Smokes*.

Moreover, we find that sub dataset in the *Yes* leaf belonging to feature *Smokes?* has only yes decisions which means that *Yes* leaf is over. So, we will continue to calculate the gini index for remaining features(*Gender, Chest pain, Cough*) from *No* leaf:

*Gender*

gini(Gender = Male) =

gini(Gender = Female)

Therefore, ginisplit(Chest pain) =

*Chest painr?*

gini(Chest pain = Yes)

gini(Chest pain = No)

Therefore, ginisplit(Chest pain)

*Cough?*

gini(Cough = Yes)

gini(Cough = No)

Therefore, ginisplit(Cough)