HW1

Due Date: Oct. 26

Submission requirements:

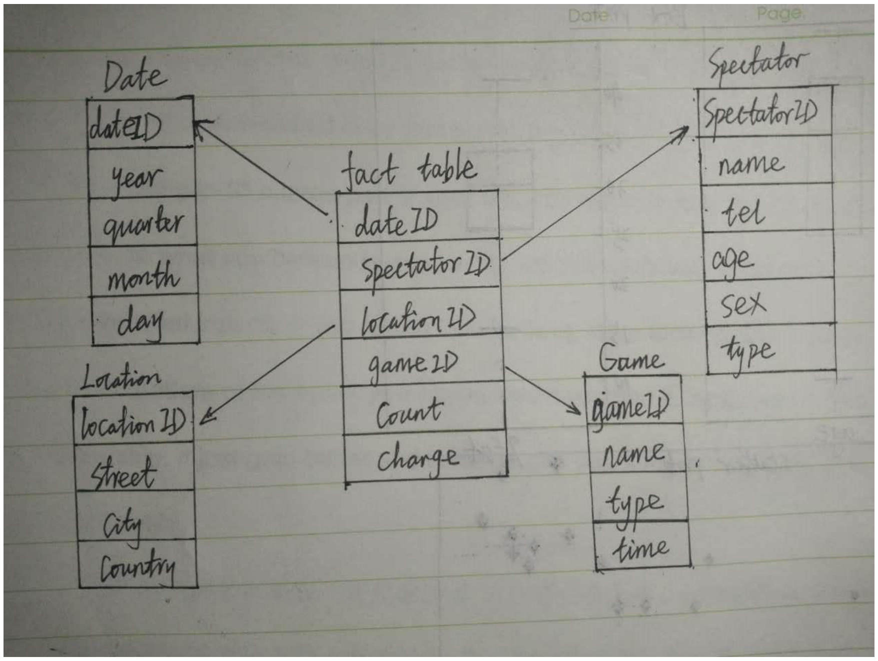
Please submit your solutions to our class website.

Part I: written part:

1. Suppose that a data warehouse consists of four dimensions, *date*, *spectator*, *location*, and *game*, and two measures, *count and charge*, where charge is the fare that a spectator pays when watching a game on a given date. Spectators may be students, adults, or seniors, with each category having its own charge rate.

(a) Draw a *star schema diagram* for the data warehouse.

star schema 如下图所示：



(b) Starting with the base cuboid [*date, spectator, location, game*]， what specific *OLAP operations* should one perform in order to list the total charge paid by student spectators in Los Angeles?

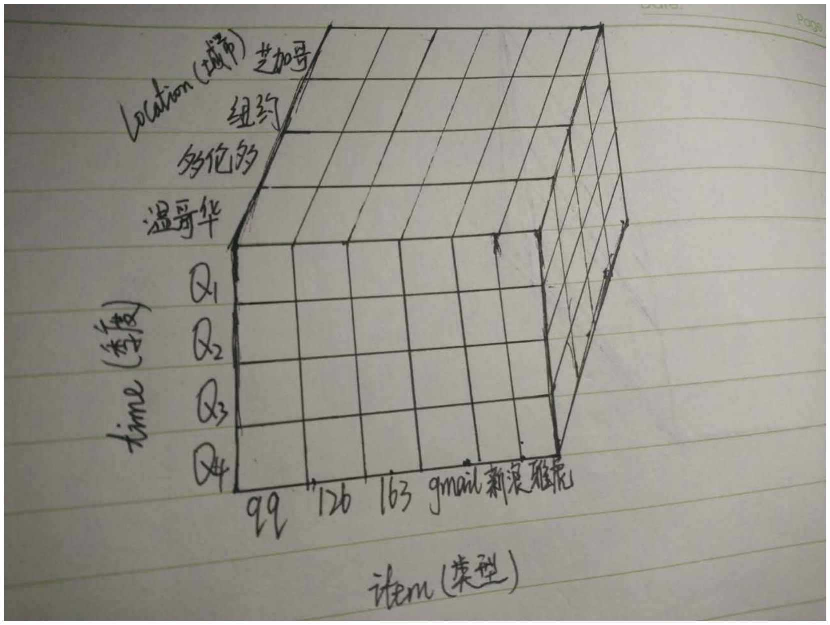
Slice for Spectator=student
Slice for Location=LosAngeles
Roll-up on Game (total games)
Roll-up on Date from day to year

(c) Bitmap indexing is a very useful optimization technique. Please present the pros and cons of using bitmap indexing in this given data warehouse.

该立方体一共有四个维（或属性），只需要为这四个为分别维护一张位图索引表，当属性的域基数较小时，因为比较、连接和聚集操作都变成了位运算，大大减少了处理时间。由于用来表示具体事务的字符串可以用单个二进位表示，位图索引显著降低了空间和 I/O 开销。但是如果属性的域的基数很大时，可能会浪费存储空间来存储大量的数据。

2．某电子邮件数据库中存储了大量的电子邮件。请设计数据仓库的结构，以便用户从多个维度进行查询和挖掘。

设计的电子邮件的数据仓库分为三个维度 location 维（按城市组织），time 维（按季度组织），item 维（按邮箱邮件类型组织）如下所示：



location 维：温哥华、多伦多、纽约

Time 维：Q1、Q2、Q3、Q4

Item 维：qq 邮箱、126 邮箱、163 邮箱、gmail 邮箱、新浪邮箱、雅虎邮箱

3. Suppose a hospital tested the age and body fat data for 18 random selected adults with the following result:

age	23	23	27	27	39	41	47	49	50	52	54	54	56	57	58	58	60	61
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

(a) Calculate the mean, median, and standard deviation of *age* and *%fat*.

mean of age:

(23+23+27+27+39+41+47+49+50+52+54+54+56+57+58+58+60+61)/18=46.4

median of age:

(50+52)/2=51

standard deviation of *age:*

Ó=12.846

mean of %fat:

(7.8+9.5+17.8+25.9+26.5+27.2+27.4+28.8+30.2+31.2+31.4+32.9+33.4+34.1+34.6+35.7+41.2+42.5)/18=28.78

median of %fat:

(30.2+31.2)/2=30.7

standard deviation of %fat*:*

Ó=8.994

(b) Draw the boxplots for *age* and *%fat*.

Five-number in summary of age:
23,39,51,57,61
Five-number in summary of %fat:
7.8,26.5,30.7,34.1,42.5
Boxplot 如图 3.(b)-(c)所示：

(c) Draw a *scatter plot* based on these two variables.

Scatter plot 如图 3.(b)-(c)所示：

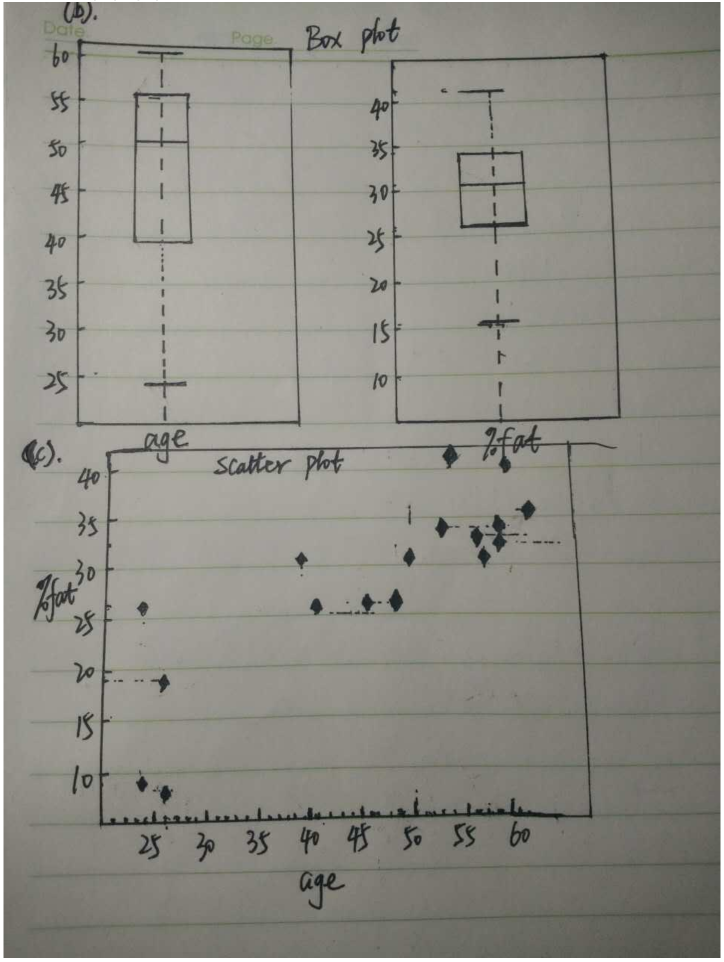


图 3.(b)-(c)

(d) Normalize the two variables based on *min-max normalization*.
Suppose new_min=0,new_max=1
Normalized data:

age	0.00	0.00	0.11	0.11	0.42	0.47	0.63	0.68	0.71	0.76	0.82	0.82	0.87	0.89	0.92	0.92	0.97	1.00
%fat	0.049	0.539	0.000	0.288	0.680	0.522	0.565	0.559	0.647	0.772	1.000	0.605	0.738	0.646	0.758	0.723	0.963	0.804

(e) Calculate the *correlation coefficient* (Pearson's product moment coefficient). Are these two variables positively or negatively correlated?

$$r_{A,B} = \frac{\sum(a_i b_i) - n\overline{A}\overline{B}}{(n-1)\sigma_A \sigma_B} = \frac{25763.2 - 18 \times 51 \times 28.78}{17 \times 12.846 \times 8.994} = -0.3344$$

$r_{A,B}$ <0,negatively correlated

(f) Smooth the fat data by bin means, using a bin depth of 6.
Bin 1: 19.1,19.1,19.1,19.1,19.1,19.1
Bin 2: 30.3,30.3,30.3,30.3,30.3,30.3
Bin 3: 36.9,36.9,36.9,36.9,36.9,36.9

(g) Smooth the fat data by bin boundaries, using a bin depth of 6.
Bin 1: 7.8,7.8,27.2,27.2,27.2,27.2
Bin 2: 27.4,27.4,32.9,32.9,32.9,32.9
Bin 3: 33.4,33.4,33.4,33.4,42.5,42.5

4．Consider the data set shown in Table 1（min_sup = 60%, min_conf=80%）.

(a) Find all frequent itemsets using Apriori by treating each transaction ID as a market basket.

Apriori：最小支持度计数值 4*60%=2.4,所以最小支持度计数为 3

C1=

A	4
B	4
C	3
D	2
E	2

L1=

A	4
B	4
C	3

C2=

AB	4
AC	3
BC	3

L2=

AB	4
AC	3
BC	3

C3=

ABC	3

L3=

ABC	3

频繁子项集：L={{A}{B}{C}{AB}{AC}{BC}{ABC}}

(b) Use the results in part (a) to compute the confidence for the association rules {a, b}→{c} and {c}→{a, b}. Is confidence a symmetric measure?

{a, b}=> {c}, confidence=3/4=75%
{c}=> {a, b}, confidence=3/3=100%
两个关联规则的置信度不相等，因此置信度不是对称规则。

(c) List all of the strong association rules (with support s and confidence c) matching the following metarule, where X is a variable representing customers, and item$_i$ denotes variables representing items (e.g. “A”, “B”, etc.):

$$\forall x \in transaction, buys(X, item_1) \wedge buys(X, item_2) => buys(X, item_3) \qquad [s, c]$$

Table 1. Example of market basket transactions.

TID	Items-bought
T1	{A, D, B, C}
T2	{D, A, C, E, B}
T3	{A, B, E}
T4	{A, B, C}

由(a)中得出的频繁项集 L 可得所有的关联规则如下：

对{AB}有两个子集{A},{B}, 得到的关联规则为：

A=>B, confidence=4/4=100%   B=>A, confidence=4/4=100%

对{AC}有两个子集{A}, {C},得到的关联规则为：

A=>C, confidence=3/4=75%   C=>A, confidence=3/3=100%

对{BC}有两个子集{B}, {C},得到的关联规则为：

B=>C, confidence=3/4=75%   C=>B, confidence=3/3=100%

对{ABC}有 6 个子集{AB}, {BC},{AC}, {C},{A}, {B},得到的关联规则为：

{AB}=>{C}, confidence=3/4=75%   {C}=>{AB}, confidence=3/3=100%

{BC}=>{A}, confidence=3/3=100%   {A}=>{BC}, confidence=3/4=75%

{AC}=>{B}, confidence=3/3=100%   {B}=>{AC}, confidence=3/4=75%

因为 min_conf=80%，得出所有强关联规则为：

buys(X, A)=>buys(X, B) [s=100%, c=100%]

buys(X, B)=>buys(X, A) [s=100%, c=100%]

buys(X,C)=>buys(X, A) [s=75%, c=100%]

buys(X, C)=>buys(X, B) [s=75%, c=100%]

buys(X, C)=>buys(X, A)^buys(X, B) [s=75%, c=100%]

buys(X, B)^buys(X, C)=>buys(X, A) [s=75%, c=100%]

buys(X, A)^buys(X, C)=>buys(X, B) [s=75%, c=100%]

5．Consider the data set shown in Table 1（min_sup = 60%）.

(a) Find all frequent itemsets using FP-Growth. Please present all the FP-trees and all the conditional pattern bases.

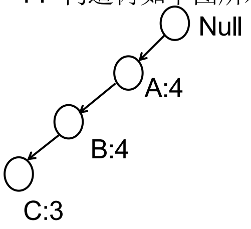
FP-Growth:

数据库的第一次扫描同 Apriori，导出频繁 1 项集的集合和支持度计数（最小支持度计数为 3）。频繁 1 项集按支持度递减序排序，结果集记为：L={{A:4},{B:4},{C:3}}。

L1=

A	4
B	4
C	3

FP 构造树如下图所示：



从底层开始构建条件模式基挖掘条件 FP 树从而找出频繁项集

项	条件模式基	条件 FP 树	产生的频繁模式
C	{{ABC:3}}	<A:3,B:3>	{AB:3}{AC:3}{BC:3}{ABC:3}
B	{AB:4}	<A:4>	{AB:4}

频繁子项集：L={{A}{B}{C}{AB}{AC}{BC}{ABC}}

(b) Compare the efficiency of Apriori and FP-Growth.
Apriori挖掘全部频繁项集时需要产生候选项集，而且需要多次重复地扫描数据库，增加I/O开销。而FP-growth是不用产生候选的方法，它构造一个高度压缩的数据结构—FP树来压缩原先的事务数据库，并且整个FP-growth过程只需2次来扫描数据库，还有就是FP-growth的不是使用Apriori方法的产生-测试策略，而聚焦于频繁模式段增长，避免了高代价的候选产生，此外，它的基本操作是计数频繁项集和建立条件FP树，没有模式搜索和匹配过程，因而获得了更好的效率。但是FP-growth也存在一些缺点，额外建立FP树是要耗内存的，而且经常包含一些冗余信息。

Part II: lab part

Question 1: Learn the use of market basket analysis for the purpose of making product purchase recommendations to the customers.

The data set contains transactions from a large supermarket. Each transaction is made by someone holding the loyalty card. We limited the total number of categories in this supermarket data to 20 categories for simplicity. The field value for a certain product in the transaction basket is 1 if the customer has bought it and 0 if he/she has not. The file named “Transactions” has data for 46243 transactions.

The data are available from the class web page.

Your submission should consist only of those deliverables marked indicated by “Hand-in”.

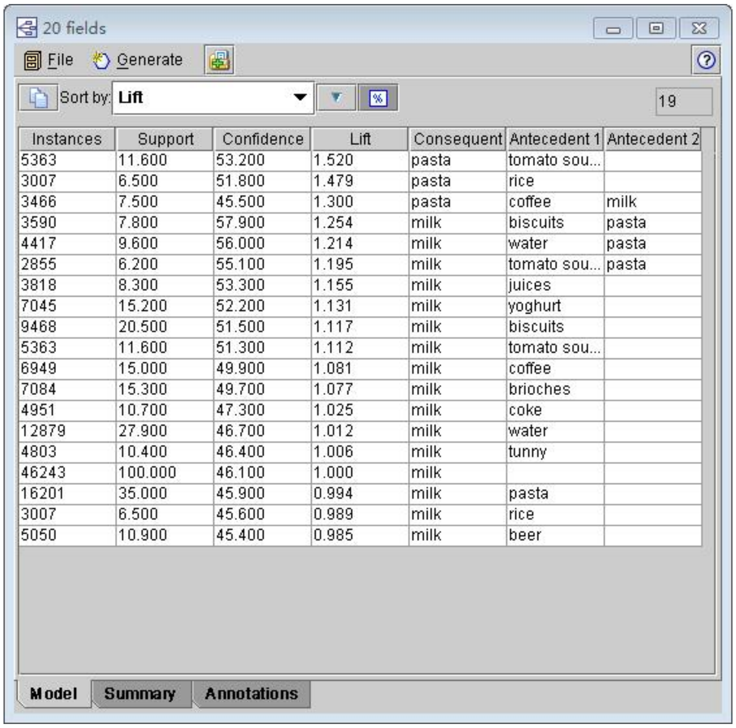
Market basket analysis has the objective to discover individual products, or groups of products that tend to occur together in transactions. The knowledge obtained from a market basket analysis can be employed by a business to recognize products frequently sold together in order to determine recommendations and cross-sell and up-sell opportunities. It can also be used to improve the efficiency of a promotional campaign.

Run Apriori on “transaction” data set. Set the “Type” of “COD” as “Typeless”, set the “direction” of all the other 20 categories as “Both”, set their “Type” as “Flag”. Set “Minimum antecedent support” to be 6%, “Minimum confidence” to be 45%, and “Maximum number of antecedents” to be 4 in the modeling node (Apriori node). In general you should explore by trying different values of these parameters to see what type of rules you get.

- **Hand-in**: The list of association rules generated by the model.

- Sort the rules by lift, support, and confidence, respectively to see the rules identified. **Hand-in**: **For each case**, choose top 5 rules (note: make sure no redundant rules in the 5 rules) and give 2-3 lines comments. Many of the rules will be logically redundant and therefore will have to be eliminated after you think carefully about them.

Sort by lift:



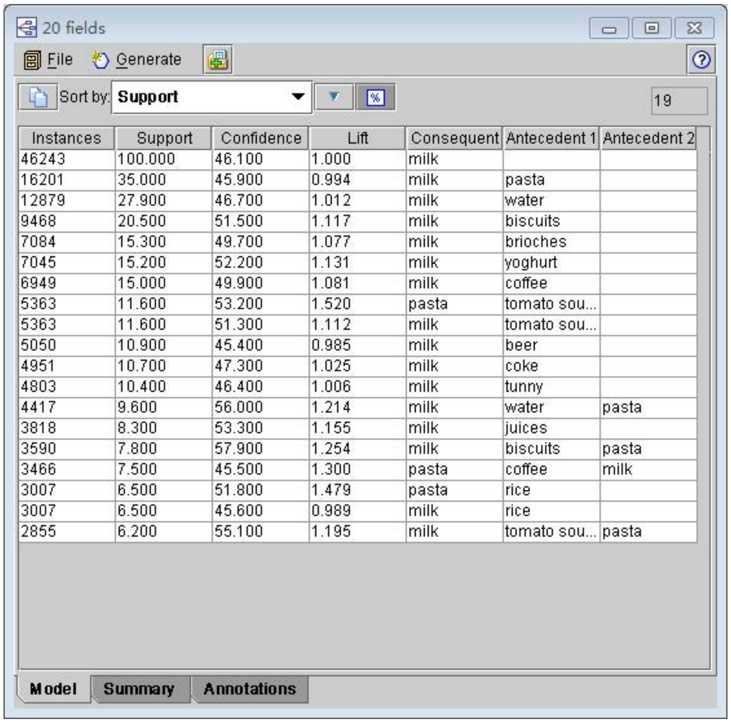
Instances	Support	Confidence	Lift	Consequent	Antecedent 1	Antecedent 2
5363	11.600	53.200	1.520	pasta	tomato sou...	
3007	6.500	51.800	1.479	pasta	rice	
3466	7.500	45.500	1.300	pasta	coffee	milk
3590	7.800	57.900	1.254	milk	biscuits	pasta
4417	9.600	56.000	1.214	milk	water	pasta
2855	6.200	55.100	1.195	milk	tomato sou...	pasta
3818	8.300	53.300	1.155	milk	juices	
7045	15.200	52.200	1.131	milk	yoghurt	
9468	20.500	51.500	1.117	milk	biscuits	
5363	11.600	51.300	1.112	milk	tomato sou...	
6949	15.000	49.900	1.081	milk	coffee	
7084	15.300	49.700	1.077	milk	brioches	
4951	10.700	47.300	1.025	milk	coke	
12879	27.900	46.700	1.012	milk	water	
4803	10.400	46.400	1.006	milk	tunny	
46243	100.000	46.100	1.000	milk		
16201	35.000	45.900	0.994	milk	pasta	
3007	6.500	45.600	0.989	milk	rice	
5050	10.900	45.400	0.985	milk	beer	

(1)符合要求的关联规则有:
Tomato souse =>pasta
rice=> pasta
coffee ^ milk   => pasta
biscuits ^ pasta =>milk
water ^ pasta=>milk
tomato souse ^ pasta=> milk
juice=>milk
yoghurt =>milk
biscuits =>milk
Tomato souse =>milk
coffee =>milk
brioches=>milk
coke =>milk
water =>milk
tunny=>milk
milk
pasta =>milk
rice=> milk
beer=> milk

(2)lift最高的 5 个规则是:
1.Tomato souse =>pasta
2.rice=> pasta
3.coffee ^ milk   => pasta
4.biscuits ^ pasta =>milk
5.water ^ pasta=>milk

按 lift 排序的前 5 个规则没有冗余规则

Sort by support:



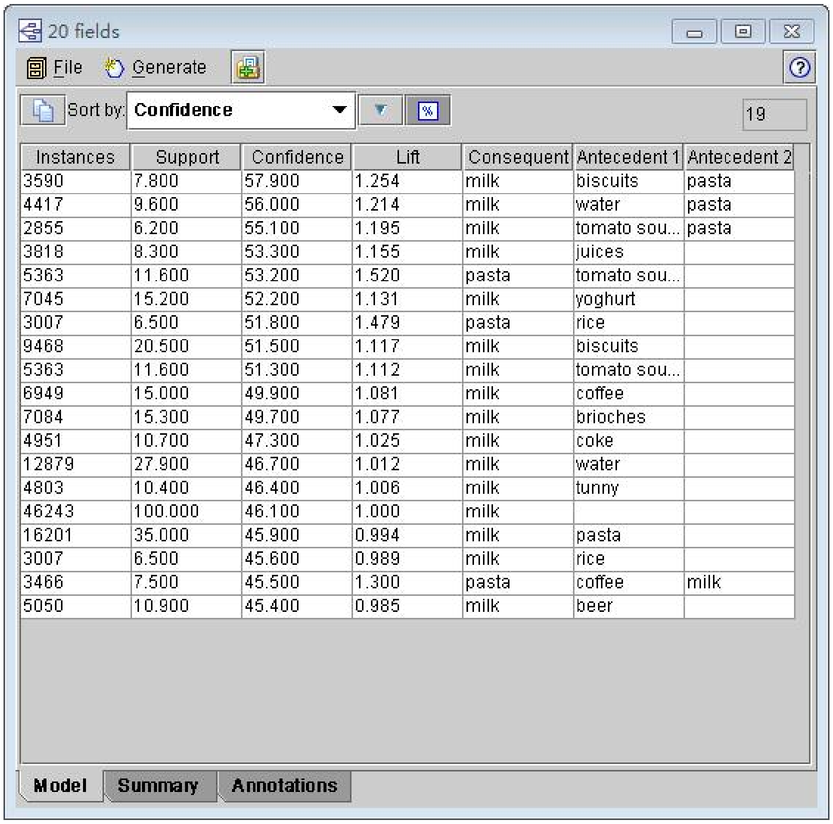
Instances	Support	Confidence	Lift	Consequent	Antecedent 1	Antecedent 2
46243	100.000	46.100	1.000	milk		
16201	35.000	45.900	0.994	milk	pasta	
12879	27.900	46.700	1.012	milk	water	
9468	20.500	51.500	1.117	milk	biscuits	
7084	15.300	49.700	1.077	milk	brioches	
7045	15.200	52.200	1.131	milk	yoghurt	
6949	15.000	49.900	1.081	milk	coffee	
5363	11.600	53.200	1.520	pasta	tomato sou...	
5363	11.600	51.300	1.112	milk	tomato sou...	
5050	10.900	45.400	0.985	milk	beer	
4951	10.700	47.300	1.025	milk	coke	
4803	10.400	46.400	1.006	milk	tunny	
4417	9.600	56.000	1.214	milk	water	pasta
3818	8.300	53.300	1.155	milk	juices	
3590	7.800	57.900	1.254	milk	biscuits	pasta
3466	7.500	45.500	1.300	pasta	coffee	milk
3007	6.500	51.800	1.479	pasta	rice	
3007	6.500	45.600	0.989	milk	rice	
2855	6.200	55.100	1.195	milk	tomato sou...	pasta

(1)符合要求的关联规则有:
milk
pasta =>milk
water =>milk
biscuits =>milk
brioches=>milk
yoghurt =>milk
coffee =>milk
Tomato souse =>pasta
Tomato souse =>milk
beer=> milk
coke =>milk
tunny=>milk
water ^ pasta=>milk
juice=>milk
biscuits ^ pasta =>milk
coffee ^ milk => pasta
rice=> pasta
rice=> milk
tomato souse ^ pasta=> milk

(2)Support 最高的 5 个规则是:
1. milk
2.pasta =>milk
3.water =>milk
4.biscuits =>milk
5.brioches=>milk

按 support 排序的表中的第 2，3，4，5 条规则是冗余的，这是因为既然可以通过仅仅促销 milk 销售，因而没有必要再去促销 pasta 或者 water 或者 water 或者 biscuits 或者 brioches 同时促销 milk。因而规则 2,3,4,5 不是有趣的，它不提供任何附加的信息。

Sort by confidence:



Sort by: Confidence

Instances	Support	Confidence	Lift	Consequent	Antecedent 1	Antecedent 2
3590	7.800	57.900	1.254	milk	biscuits	pasta
4417	9.600	56.000	1.214	milk	water	pasta
2855	6.200	55.100	1.195	milk	tomato sou...	pasta
3818	8.300	53.300	1.155	milk	juices	
5363	11.600	53.200	1.520	pasta	tomato sou...	
7045	15.200	52.200	1.131	milk	yoghurt	
3007	6.500	51.800	1.479	pasta	rice	
9468	20.500	51.500	1.117	milk	biscuits	
5363	11.600	51.300	1.112	milk	tomato sou...	
6949	15.000	49.900	1.081	milk	coffee	
7084	15.300	49.700	1.077	milk	brioches	
4951	10.700	47.300	1.025	milk	coke	
12879	27.900	46.700	1.012	milk	water	
4803	10.400	46.400	1.006	milk	tunny	
46243	100.000	46.100	1.000	milk		
16201	35.000	45.900	0.994	milk	pasta	
3007	6.500	45.600	0.989	milk	rice	
3466	7.500	45.500	1.300	pasta	coffee	milk
5050	10.900	45.400	0.985	milk	beer	

(1)符合要求的关联规则有:
biscuits ^ pasta =>milk
water ^ pasta=>milk
tomato souse ^ pasta=> milk
juice=>milk
Tomato souse =>pasta
yoghurt =>milk
rice=> pasta
biscuits =>milk
Tomato souse =>milk
coffee =>milk
brioches=>milk
coke =>milk
water =>milk
tunny=>milk
milk
pasta =>milk
rice=> milk
coffee ^ milk => pasta
beer=> milk
(2)confidence最高的 5 个规则是:
1.biscuits ^ pasta =>milk
2.water ^ pasta=>milk
3.tomato souse ^ pasta=> milk
4.juice=>milk
5.tomato souse =>pasta
按 confidence 排序的前 5 个规则没有冗余规则