APRIORI FOR ASSOCIATION RULE MINING

a. Implementation of Fk-1 × F1 Method:
   This method requires merging of 1 candidate item set to 1:k other candidate itemsets.

   Implementation of Fk-1 × Fk-1 Method:
   This is an improvised method that generated lower number of candidates when compared with the previous method. It creates a new candidate by merging k-1 same items in both the sets and then appending the kth item of both the sets.
   For eg: A = {beer,bread,diaper}
   B = {beer,bread,eggs}
   New Candidate: {beer, bread, diaper, eggs}

   The basic requirement of such methods is to reduce the generation of unnecessary candidates that might not contribute generation in interesting rules. Therefore it reduces the computational complexity when compared to Brute Force method.

b. Let look at the results obtained after implementation of the above methods on three different datasets.

	Market Basket Dataset : 1361 Records		
	Min Support	No. of Candidates	No. of Frequent Items
Fk-1 × F1	35	29630	448
	40	16914	292
	45	6197	160
Fk-1 × Fk-1	35	13635	388
	40	9949	277
	45	4041	155

	adults.csv : 1000 Records		
	Min Support	No. of Candidates	No. of Frequent Items
Fk-1 × F1	100	6985	1208
	150	3612	506
	200	2100	220
Fk-1 × Fk-1	100	6294	1012
	150	2725	420
	200	1547	216

	plants.csv : 10000 Records		
	Min Support	No. of Candidates	No. of Frequent Items
Fk-1 × F1	800	6745	762

	850	6282	659
	900	4281	400
Fk-1 × Fk-1	800	2209	493
	850	1914	411
	900	1733	334

OBSERVATIONS

Looking at the results obtained we can infer following:

1. **Methods:**
   Fk-1 × Fk-1 Method is better than Fk-1 × F1 as it generated less number of important candidates. This is because the later method may generate infrequent items while merging. For eg:- 1-Itemset = {Bread,milk,coke}
   2-Itemset = {[Bread,milk],[Bread,coke],[milk, Bread]}
   The Fk-1 × F1 will generate {Bread, milk, coke} which may be infrequent and will be pruned later in the course. However Fk-1 × Fk-1 Method essentially doesn't create such candidates and hence creates less number of candidates and hence reduces the computational complexity.
2. **Minimum Support Threshold:**
   Now let us focus on the candidate generations for various levels of support. In the Market Basket dataset, we can observe that when the support count in 35, high number of candidates are generated compared to support count of 45.
   However, running the algorithm for low support count is costlier than larger support count. This is attributed to the fact that when the threshold is low, the potential number of candidates are more due to more items in frequent itemlist. This inturn contributes to more number of rules generated for low support count. These rules may not be interesting if the support count is very low. Hence, we need to make a conscious decision about the selection minimum support threshold.

c.

FREQUENT CLOSED ITEMSETS:
Definitions:
Closed itemsets provides a minimal representation of itemsets without losing their support information. An itemset X is closed if none of its immediate superset has exactly the same support as X.
We are required to find frequent closed itemset.
An itemset is a closed frequent itemset if it is closed and its support is greater then or equal to minsup.

MAXIMAL FREQUENT ITEMSETS
Definitions:
A maximal frequent itemset is defined as a frequent itemset for which none of its immediate supersets are frequent.

No. of maximal frequent itemsets < No of Frequent closed itemsets < no.frequent itemsets

ADVANTAGES:

1. In case of large transactional data, this kind of compact representation sets can help in deriving all other frequent itemsets.
2. Use of such compact representations results in reduction of computational time and memory requirements.

Now lets go ahead and look at the number of itemsets generated via the above two methods.

	Market Basket Dataset: 1361 Records			
	Min Support	No. of Frequent Items	No. of Frequent Closed Items	No. of Maximal Frequent Items
Fk-1 × F1	35	448	139	2
	40	292	115	18
	45	160	102	5
Fk-1 × Fk-1	35	388	208	5
	40	277	156	138
	45	155	112	68

	adults.csv : 1000 Records			
	Min Support	No. of Frequent Items	No. of Frequent Closed Items	No. of Maximal Frequent Items
Fk-1 × F1	100	465	444	41
	150	312	265	219
	200	213	114	79
Fk-1 × Fk-1	100	1012	318	2
	150	420	217	3
	200	98	92	90

	Plants Dataset: 10000 Records			
	Min Support	No. of Frequent Items	No. of Frequent Closed Items	No. of Maximal Frequent Items
Fk-1 × F1	800	512	490	473
	850	497	484	333
	900	466	394	377
Fk-1 × Fk-1	800	493	457	433
	850	411	393	381
	900	334	317	280

OBSERVATION

We can observe in every case that the count of frequents items in maximal frequent items is lowest, frequent closed is intermediate and highest in normal method of frequent candidate selection.

In conclusion, these representations can aid in regeneration of standard frequent itemsets.

d. This part in the question required to actually generate rules using the frequent itemsets obtained using different methods.
   Confidence(A->B) indicates how often do we see A and B together whenever we see A in our data set. That is Count(A and B) divided by Count(A).

	adults.csv : 1000 Records			
	Min Support	Min Confidence	No. of Rules generated	Brute Force
		0.9	293	3213
		0.95	137	3213
	100	0.7	1614	3213
		0.9	71	526
		0.95	18	526
	150	0.7	499	526
		0.9	39	319
		0.95	10	319
3Fk-1 × F1	200	0.7	188	319
		0.9	283	2133
		0.95	131	2133
	100	0.7	1584	2133
		0.9	60	983
		0.95	15	983
	150	0.7	474	983
		0.9	30	317
		0.95	8	317
Fk-1 × Fk-1	200	0.7	185	317

Market Basket Dataset: 1361 Records				
	Min Support	Min Confidence	No. of Rules generated	Brute Force
	35	0.3	91	183
		0.4	22	183
		0.55	5	183
	40	0.2	13	34
		0.3	8	34
		0.4	3	34
	45	0.3	15	33
Fk-1 × F1		0.4	9	33

		0.55	1	33
	35	0.3	14	18
		0.4	8	18
		0.55	5	18
	40	0.3	9	17
		0.4	5	17
		0.55	2	17
	45	0.3	5	16
		0.4	2	16
Fk-1 × Fk-1		0.55	1	16

Plant Dataset: 10000 Records				
	Min Support	Min Confidence	No. of Rules generated	Brute Force
	800	0.7	16	35
		0.8	6	35
		0.9	2	35
	850	0.7	18	23
		0.8	4	23
		0.9	0	23
	900	0.7	10	21
		0.8	3	21
Fk-1 × F1		0.9	0	21
	800	0.7	13	29
		0.8	4	29
		0.9	2	29
	850	0.7	14	25
		0.8	3	25
		0.9	0	25
	900	0.7	9	21
		0.8	4	21
Fk-1 × Fk-1		0.9	0	21

OBSERVATION

Looking at the tables we observe that at low support threshold and low confidence, highest number of rules are generated.
At High Minsup and High MinConf, very low/none rules are generated.
If the MinConf value is very low, high number of rules are generated. This might lead to uninteresting rules showing up in our result. Also, if the Minsup is very low, we might miss interesting rules in our result.
We can observe that due to confidence pruning, significant number of rules are reduced as compared to the brute force method. Brute force might result in uninteresting rules generation. This can be

avoided by using our confidence measure that indicates the likelihood of actual rule occurring in reality.

Hence, we need to build an algorithm with a proper value of support and confidence, that are computationally affordable and also provides insightful rules.

e. Now we will look at the top 5 interesting rules generated by the various combinations.

Minsup	**Minconf**	**conf/lift**	**Top Rules**
35	0.4	0.59322	([' 2pct. Milk', ' Toothpaste'], '->', [' Eggs', ' White Bread'])")
		0.590164	([' 2pct. Milk', ' Potato Chips'], '->', [' Eggs', ' White Bread'])")
		0.57377	([' Toothpaste', ' Eggs'], '->', [' 2pct. Milk', ' White Bread'])
		0.545455	([' Eggs', ' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])")
		0.538462	([' White Bread', ' Toothpaste'], '->', [' 2pct. Milk', ' Eggs'])")
	0.3	0.59322	([' 2pct. Milk', ' Toothpaste'], '->', [' Eggs', ' White Bread'])")
		0.590164	([' 2pct. Milk', ' Potato Chips'], '->', [' Eggs', ' White Bread'])")
		0.57377	([' Toothpaste', ' Eggs'], '->', [' 2pct. Milk', ' White Bread'])
		0.545455	([' Eggs', ' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])")
		0.538462	([' White Bread', ' Toothpaste'], '->', [' 2pct. Milk', ' Eggs'])")
	0.55	0.59322	([' 2pct. Milk', ' Toothpaste'], '->', [' Eggs', ' White Bread'])")
		0.590164	([' 2pct. Milk', ' Potato Chips'], '->', [' Eggs', ' White Bread'])")
		0.57377	([' Toothpaste', ' Eggs'], '->', [' 2pct. Milk', ' White Bread'])

Minsup	**Minconf**	**conf/lift**	**Top Rules**
40	0.2	0.5125	([' Hot Dog Buns'], '->', [' Hot Dogs', ' Sweet Relish'])
		0.40952381	([' Wheat Bread'], '->', [' 2pct. Milk', ' White Bread'])")
		0.39047619	([' Wheat Bread'], '->', [' Eggs', ' White Bread'])")
		0.353448276	([' Sweet Relish'], '->', [' Hot Dog Buns', ' Hot Dogs'])
		0.325396825	([' Hot Dogs'], '->', [' Hot Dog Buns', ' Sweet Relish'])")
	0.3	0.5125	([' Hot Dog Buns'], '->', [' Hot Dogs', ' Sweet Relish'])
		0.40952381	([' Wheat Bread'], '->', [' 2pct. Milk', ' White Bread'])")
		0.39047619	([' Wheat Bread'], '->', [' Eggs', ' White Bread'])")
		0.353448276	([' Sweet Relish'], '->', [' Hot Dog Buns', ' Hot Dogs'])
		0.325396825	([' Hot Dogs'], '->', [' Hot Dog Buns', ' Sweet Relish'])")
	0.4	0.5125	([' Hot Dog Buns'], '->', [' Hot Dogs', ' Sweet Relish'])
		0.40952381	([' Wheat Bread'], '->', [' 2pct. Milk', ' White Bread'])")

Minsup	**Minconf**	**conf/lift**	**Top Rules**

	0.3	0.416667	([' Toothpaste'], '->', [' 2pct. Milk', ' White Bread'])
		0.345865	([' Potato Chips'], '->', [' 2pct. Milk', ' Eggs'])
		0.345865	([' Potato Chips'], '->', [' Eggs', ' White Bread'])
		0.338346	([' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])
		0.328859	([' 2pct. Milk'], '->', [' Eggs', ' White Bread'])
	0.4	0.416667	([' Toothpaste'], '->', [' 2pct. Milk', ' White Bread'])
	0.2	0.416667	([' Toothpaste'], '->', [' 2pct. Milk', ' White Bread'])
		0.345865	([' Potato Chips'], '->', [' 2pct. Milk', ' Eggs'])
		0.345865	([' Potato Chips'], '->', [' Eggs', ' White Bread'])
		0.338346	([' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])
45		0.328859	([' 2pct. Milk'], '->', [' Eggs', ' White Bread'])

Note: In order to avoid this document to be cluttered from excessive tables, I have created **rules_conf.xls** document with top 5 association rules for other two data sets.

EXPLANATION:

Let look at the generated rule:

1. ([' Hot Dog Buns'], '->', [' Hot Dogs', ' Sweet Relish'])

   Logically anyone who would buy a hot dog bun would also buy hot dogs and sweet relish with it make the dish complete. Hence the rule looks correct.
2. ([' Wheat Bread'], '->', [' Eggs', ' White Bread'])")

   People often buy breads at once. So wheat and white bread are bought together. Moreover, bread and eggs are another common things bought together.
3. ([' Toothpaste', ' Eggs'], '->', [' 2pct. Milk', ' White Bread'])

   Here we see toothpaste in the list. Generally, whenever a person goes out to buy grocery like toothpaste with eggs, it indicates that he will also buy other regular consumables like milk, bread etc.

   Note: We can observe new rules to appear for various levels of support. If the confidence is increased, there are potential rules that are eliminated by marking them unimportant.

PART F

In this part we will be using lift as our measure instead of confidence.

$$Lift(A\text{->}B) = Confidence(A\text{->}B)/ Support(A)$$

Confidence has an issue of ignoring the Support of the itemset appearing in the rule consequent.
Lets look at the results obtained:

Minsup	**MinLift**	**conf/lift**	**Top Rules**
	0.004	0.00640625	([' Hot Dog Buns'], '->', [' Hot Dogs', ' Sweet Relish'])
	0.003	0.00640625	([' Hot Dog Buns'], '->', [' Hot Dogs', ' Sweet Relish'])
		0.003900227	([' Toothpaste'], '->', [' 2pct. Milk', ' White Bread'])
40		0.003858025	([' Wheat Bread'], '->', [' 2pct. Milk', ' White Bread'])

		0.003809524	([' Toothpaste'], '->', [' Eggs', ' White Bread'])
		0.003772291	([' Wheat Bread'], '->', [' 2pct. Milk', ' Eggs'])")
	0.0038	0.003858025	([' Wheat Bread'], '->', [' 2pct. Milk', ' White Bread'])
		0.003809524	([' Toothpaste'], '->', [' Eggs', ' White Bread'])
		0.003772291	([' Wheat Bread'], '->', [' 2pct. Milk', ' Eggs'])")
	0.005	0.010054582	([' 2pct. Milk', ' Toothpaste'], '->', [' Eggs', ' White Bread'])")
		0.009674819	([' 2pct. Milk', ' Potato Chips'], '->', [' Eggs', ' White Bread'])")
		0.009406074	([' Toothpaste', ' Eggs'], '->', [' 2pct. Milk', ' White Bread'])
		0.008284024	([' Eggs', ' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])")
		0.008264463	([' White Bread', ' Toothpaste'], '->', [' 2pct. Milk', ' Eggs'])")
	0.008	0.010054582	([' 2pct. Milk', ' Toothpaste'], '->', [' Eggs', ' White Bread'])")
		0.009674819	([' 2pct. Milk', ' Potato Chips'], '->', [' Eggs', ' White Bread'])")
		0.009406074	([' Toothpaste', ' Eggs'], '->', [' 2pct. Milk', ' White Bread'])
		0.008284024	([' Eggs', ' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])")
		0.008264463	([' White Bread', ' Toothpaste'], '->', [' 2pct. Milk', ' Eggs'])")
	0.009	0.010054582	([' 2pct. Milk', ' Toothpaste'], '->', [' Eggs', ' White Bread'])")
		0.009674819	([' 2pct. Milk', ' Potato Chips'], '->', [' Eggs', ' White Bread'])")
35		0.009406074	([' Toothpaste', ' Eggs'], '->', [' 2pct. Milk', ' White Bread'])
	0.003	0.416666667	([' Toothpaste'], '->', [' 2pct. Milk', ' White Bread'])
	0.002	0.416666667	([' Toothpaste'], '->', [' 2pct. Milk', ' White Bread'])
		0.345864662	([' Potato Chips'], '->', [' 2pct. Milk', ' Eggs'])
		0.345864662	([' Potato Chips'], '->', [' Eggs', ' White Bread'])
		0.338345865	([' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])
		0.32885906	([' 2pct. Milk'], '->', [' Eggs', ' White Bread'])
	0.001	0.416666667	([' Toothpaste'], '->', [' 2pct. Milk', ' White Bread'])
		0.345864662	([' Potato Chips'], '->', [' 2pct. Milk', ' Eggs'])
		0.345864662	([' Potato Chips'], '->', [' Eggs', ' White Bread'])
		0.338345865	([' Potato Chips'], '->', [' 2pct. Milk', ' White Bread'])
45		0.32885906	([' 2pct. Milk'], '->', [' Eggs', ' White Bread'])

Note: In order to avoid this document to be cluttered from excessive tables, I have created **rules_lift.xls** document with top 5 association rules generated using lift for other two data sets.