1) The following contingency table summarizes the survey data of a student population, where ski refers to students who ski, $\overline{ski}$ refers to students who do not ski, $football$ refers to students who play football, and $\overline{football}$ refers to students who do not play football.

	$foot$	$\overline{foot}$	$\sum_{ro}$
ski	1500	1000	2500
$\overline{ski}$	500	1000	1500
$\sum_{co}$	2000	2000	4000

(a) Based on the given data, determine the correlation relationship between ski and playing football using the *lift* measure.

(b) Suppose that the association rule "*ski ⇒ football* " is mined. Given a minimum support threshold of 25% and a minimum confidence threshold of 50%, is this association rule strong (i.e., meet the thresholds)?

1) a)
Correlation between events will be determined by lift measure using the below equation.

lift(A,B) = P(AUB) / P(A)P(B)

From the given data,
P(ski) = 2500/4000
P(fooftball) = 2000/4000
P(ski U football) = 1500/4000

lift(ski, football) = (1500/4000) / (2500/4000)(2000/4000)
lift(ski, football) = 1.2
lift(ski, football) > 1 so ski and football are positively dependent.

1) b)
Support for the given association rule A ⇒ B is Sup(A ⇒ B) = P(AUB)
Confidence for the given association rule A ⇒ B is Conf(A ⇒ B) = Sup(AUB) / Sup(A)
We are given the association rule ski ⇒ football.

Let's find support and confidence for the given association rule.
Sup(ski ⇒ football) = P(ski ⇒ football)
Sup(ski ⇒ football) = 1500/4000
Sup(ski ⇒ football) = 0.375
Sup(ski ⇒ football) = 37.5%

Conf(ski ⇒ football) = Sup(ski ⇒ football) / Sup(ski)
Conf(ski ⇒ football) = P(ski ⇒ football) / P(ski)
Conf(ski ⇒ football) = (1500/4000) / (2500/4000)
Conf(ski ⇒ football) = 1500/2500
Conf(ski ⇒ football) = 0.6
Conf(ski ⇒ football) = 60%

Support and confidence for the association rule (ski ⇒ football) is
ski ⇒ football [37.5%, 60%]

We are given minimum support threshold = 25% and minimum confidence threshold = 50%
As we can see association rule, ski ⇒ football satisfied both minimum support and minimum confidence thresholds hence it is **strong association rule**.

2) Given a data set with five transactions, each containing five items, as shown in the table. Let *min_support* = 60%.

TID	***items_bought***
T1	{E, G, S, F, Z}
T2	{B, E, D, I, N}
T3	{B, E, I, N, O}
T4	{B, G, I, N, Z}
T5	{B, G, N, T, Z}

(a) What is the maximum number of possible frequent itemsets?
(b) Find all frequent itemsets using the Apriori algorithm. Your answer should include the key steps of the computation process.

(c) In the computation above, how many rounds of database scan are needed? What is the total number of candidates?

(d) Let *n* be the total number of transactions, *b* be the number of items in each transaction, *m* be the number of *k*-itemset candidates. Consider the following two different approaches for counting the support values of the candidates. For each transaction, the first approach checks if a candidate occurred in the transaction or not; the second approach enumerates all the possible *k*-itemsets of the transaction and checks if the itemset is one of the candidates. What is the computation complexity for each approach? Is one always better than the other?

2) a) For a given dataset with k items, there are 2^k -1 potentially possible frequent itemsets.
so potentially possible frequent itemsets are = 2^{11} -1= 2047.
Out of all these possible itemsets we are interested in itemsets with minimum support count.
Here, we are given 11 items,
We can find all frequent itemsets with minimum support count using enumeration.

item	support count
{B}	4
{E}	3
{G}	3
{I}	3
{N}	4
{Z}	3
{B,I}	3
{B,N}	4
{G,Z}	3
{I,N}	3
{B,I,N}	3

Out of 2047 possible number of frequent itemsets, there are only 11 such frequent itemsets

2)b)

We are given transactions with TID and item sets. Let's consider a database D which has all these transactions.
We are given min_support = 60% means, total support count = 60% of 5 = 3
We need to find frequent itemsets with minimum support count >= 3 using Apriori algorithm.

From the given items bought, we can build candidate set of 1-itemsets C_1 as below.
In the first iteration of the algorithm, each item is a member of the set of candidates 1-itemsets, C_1. The algorithm simply scans all the transactions in database D in order to count the number of occurrences of each item.

C_1 Itemset	Support Count
{B}	4
{D}	1

{E}	3
{F}	1
{G}	3
{I}	3
{N}	4
{O}	1
{S}	1
{T}	1
{Z}	3

Frequent 1-itemsets L_1 consists of the candidate itemsets satisfying the minimum support count of 3. So all the candidates in C_1, with support count>=3 are in L_1

L_1 Itemset	Support Count
{B}	4
{E}	3
{G}	3
{I}	3
{N}	4
{Z}	3

Candidate set of 2-itemsets C_2 will be generated by joining L_1 with itself. Here no candidates are removed from C_2 during the pruning step since each subset of the candidates is also frequent. At the same time scan all the transactions in database D to calculate support count for each candidate itemset C_2.

C_2 Itemset	Support Count
{B,E}	2
{B,G}	2
{B,I}	3
{B,N}	4

{B,Z}	2
{E,G}	1
{E,I}	2
{E,N}	2
{E,Z}	1
{G,I}	1
{G,N}	2
{G,Z}	3
{I,N}	3
{I,Z}	1
{N.Z}	2

Frequent 2-itemsets L_2 consists of the candidate itemsets satisfying the minimum support count of 3. So all the candidates in C_2, with support count>=3 are in L_2

L_2 Itemset	Support Count
{B,I}	3
{B,N}	4
{G,Z}	3
{I,N}	3

Candidate set of 3-itemsets C_3 will be generated by joining L_2 with itself. At the same time scan all the transactions in database D to calculate support count for each candidate itemset C_3.

C_3 Itemset	Support Count
{B,I,N}	3

As C_3 Itemset has minimum support count, it will be present in frequent 3-itemsets L_2.

L_3 Itemset	Support Count
{B,I,N}	3

Since there are no large itemsets are present, Apriori algorithm will terminate here.
Frequent itemsets found using Apriori algorithm are listed below.

Frequent 1-itemsets L_1 = {B}, {E}, {G}, {I}, {N}, {Z}
Frequent 2-itemsets L_2 = {B,I}, {B,N}, {G,Z}, {I,N}
Frequent 3-itemsets L_3 = {B,I,N}
All Frequent itemsets L = {{B}, {E}, {G}, {I}, {N}, {Z}, {B,I}, {B,N}, {G,Z}, {I,N}, {B,I,N}}

2) c) In the process above to find all frequent items using Apriori algorithm, we scanned all the transactions in the database for 3 times to determine support count for each candidate itemsets. So **3 rounds of database scan** are needed.
In the above process we generated,
Set of candidates 1-itemsets, C_1= 11
Set of candidates 2-itemsets, C_2= 15
Set of candidates 3-itemsets, C_3= 1
So, **total number of candidates = 27**

2) d) Let's say n be the total number of transactions, b be the number of items in each transaction and m be the number of k-itemset candidates.
We are given that first approach checks if a candidate occurred in the transaction or not. So it scans all the items in the database for each transaction and it will be repeated for all candidate itemsets..
Then, computation complexity for the first approach is O(n*b*m)

Second approach enumerates all the possible k-itemsets of the transaction and checks if the itemset is one of the candidates
Computation complexity for the second approach is O(n*b*(${}^{b}C_{k}$))

3) Given a data set with four transactions. Let *min_support* = 60%, and *min_confidence* = 80%.

cust_ID	***TID***	***items_bought* (in the form of brand-item category)**
01	T100	{Sunny-Cherry, Dairyland-Milk, Wonder-Bread, Sweet-Pie}
02	T200	{Best-Cheese, Dairyland-Milk, Goldenfarm-Cherry, Sweet-Pie, Wonder-Bread}
01	T300	{King's-Cereal, Sunset-Milk, Dairyland-Cheese, Best-Bread}
03	T400	{Wonder-Bread, Sunset-Milk, Best-Cereal, Sweet-Pie, Dairyland-Cheese}

(a) At the granularity of *item_category* (e.g., $item_i$ could be "*Milk*" and ignore brand name), for the following rule template,

$\forall$X $\in$ **transaction**, *buys*(X, $item_1$) $\wedge$ *buys*(X, $item_2$) $\Rightarrow$ *buys*(X, $item_3$) [s, c]

list the frequent *k*-itemset for the largest *k*, and all of the strong association rules (with their support *s* and confidence *c*) containing the frequent *k*-itemset for the largest *k*.

(b) At the granularity of *brand−item_category* (e.g., $item_i$ could be "*Sunset − Milk*"), for the following rule template,

$\forall$X $\in$ **customer**, *buys*(X, $item_1$) $\wedge$ *buys*(X, $item_2$) $\Rightarrow$ *buys*(X, $item_3$)

list the frequent k-itemset for the largest k (but do not print any rules).

3) a)
I have found frequent itemsets for the largest k (i.e, k = 3) using Apriori algorithm in the 3) b) question below and please check below 3)b) for the explanation.

Frequent 3-itemsets got from 3) b) are {{Bread, Milk, Cheese}, {Bread, Milk, Pie}, {Pie, Milk, cheese}}

Bread∧ Milk => Cheese, [100%, 100%]
Bread ∧ Cheese => Milk, [100%, 100%]
Cheese ∧ Milk => Bread, [100%, 100%]

Bread ∧ Milk => Pie, [100%, 100%]
Pie∧ Milk => Bread, [100%, 100%]
Pie ∧ Bread => Milk, [100%, 100%]

Pie ∧ Milk => Cheese, [100%, 100%]
Cheese ∧ Milk => Pie, [100%, 100%]
Cheese ∧ Pie => Milk, [100%, 100%]

3) b)
Before going ahead, we need to merge transactions with same cust_id.

cust_ID	***TID***	***items_bought*** **(in the form of brand-item category)**
01	T100	{Sunny-Cherry, Dairyland-Milk, Wonder-Bread, Sweet-Pie, King's-Cereal, Sunset-Milk, Dairyland-Cheese, Best-Bread}

02	T200	{Best-Cheese, Dairyland-Milk, Goldenfarm-Cherry, Sweet-Pie, Wonder-Bread}
03	T400	{Wonder-Bread, Sunset-Milk, Best-Cereal, Sweet-Pie, Dairyland-Cheese}

We are given min_support = 60% means, total support count = 60% of 4 = 2
We need to find frequent itemsets with minimum support count >= 2 using Apriori algorithm.

From the given items bought, we can build candidate set of 1-itemsets C_1 as below.
In the first iteration of the algorithm, each item is a member of the set of candidates 1-itemsets, C_1. The algorithm simply scans all the transactions in database D in order to count the number of occurrences of each item.

C_1 Itemset	Support Count
{Sunny-Cherry}	1
{Dairyland-Milk}	2
{Wonder-Bread}	3
{Sweet-Pie}	3
{Best-Cheese}	1
{Goldenfarm-Cherry}	1
{King's-Cereal}	1
{Sunset-Milk}	2
{Dairyland-Cheese}	2
{Best-Bread}	1
{Best-Cereal}	1

Frequent 1-itemsets L_1 consists of the candidate itemsets satisfying the minimum support count of 2. So all the candidates in C_1, with support count>=2 are in L_1

L_1 Itemset	Support Count
{Dairyland-Milk}	2
{Wonder-Bread}	3
{Sweet-Pie}	3
{Sunset-Milk}	2

{Dairyland-Cheese}	2

Candidate set of 2-itemsets C_2 will be generated by joining L_1 with itself. At the same time scan all the transactions in database D to calculate support count for each candidate itemset C_2.

C_2 Itemset	Support Count
{Dairyland-Milk, Wonder-Bread}	2
{Dairyland-Milk,Sweet-Pie}	2
{Dairyland-Milk,Sunset-Milk}	0
{Dairyland-Milk,Dairyland-Cheese}	1
{Wonder-Bread,Sweet-Pie}	3
{Wonder-Bread,Sunset-Milk}	2
{Wonder-Bread,Dairyland-Cheese}	2
{Sweet-Pie,Sunset-Milk}	2
{Sweet-Pie,Dairyland-Cheese}	2
{Sunset-Milk,Dairyland-Cheese}	2

Frequent 2-itemsets L_2 consists of the candidate itemsets satisfying the minimum support count of 2. So all the candidates in C_2, with support count>=2 are in L_2

L_2 Itemset	Support Count
{Dairyland-Milk, Wonder-Bread}	2
{Dairyland-Milk,Sweet-Pie}	2
{Wonder-Bread,Sweet-Pie}	3
{Wonder-Bread,Sunset-Milk}	2
{Wonder-Bread, Dairyland-Cheese}	2
{Sweet-Pie,Sunset-Milk}	2
{Sweet-Pie,Dairyland-Cheese}	2
{Sunset-Milk,Dairyland-Cheese}	2

Candidate set of 3-itemsets C_3 will be generated by joining L_2 with itself. At the same time scan all the transactions in database D to calculate support count for each candidate itemset C_3.

C_3 Itemset	Support Count
{Dairyland-Milk, Wonder-Bread, Sweet-Pie}	2
{Wonder-Bread, Sweet-Pie, Sunset-Milk}	2
{Wonder-Bread, Sweet-Pie, Dairyland-Cheese}	2
{Wonder-Bread,Sunset-Milk, Dairyland-Cheese}	2
{Sweet-Pie, Sunset-Milk, Dairyland-Cheese}	2

Frequent 3-itemsets L_3 consists of the candidate itemsets satisfying the minimum support count of 2. So all the candidates in C_3, with support count>=2 are in L_3

L_3 Itemset	Support Count
{Dairyland-Milk, Wonder-Bread, Sweet-Pie}	2
{Wonder-Bread, Sweet-Pie, Sunset-Milk}	2
{Wonder-Bread, Sweet-Pie, Dairyland-Cheese}	2
{Wonder-Bread,Sunset-Milk, Dairyland-Cheese}	2
{Sweet-Pie, Sunset-Milk, Dairyland-Cheese}	2

From the above sets, we need to consider minimum confidence as well. Means, we need to find out sets which has confidence >= 80%.
Dairyland-Milk $\wedge$ Wonder-Bread => Sweet-Pie [66.7%, 100%]
Wonder-Bread $\wedge$ Sunset-Milk => Dairyland-Cheese [66.7%, 100%]
Sweet-Pie $\wedge$ Sunset-Milk => Dairyland-Cheese [66.7%, 100%]

Frequent 3-itemsets satisfies minimum support and confidence given are,
L_3 = {{Dairyland-Milk, Wonder-Bread, Sweet-Pie}, {Wonder-Bread,Sunset-Milk, Dairyland-Cheese}, {Sweet-Pie, Sunset-Milk, Dairyland-Cheese}}