APRIORI ALGORITHM

Apriori is an algorithm used to identify frequent item sets (in our case, item pairs). It does so using a "bottom up" approach, first identifying individual items that satisfy a minimum occurence threshold. It then extends the item set, adding one item at a time and checking if the resulting item set still satisfies the specified threshold. The algorithm stops when there are no more items to add that meet the minimum occurrence requirement. Here's an example of apriori in action, assuming a minimum occurence threshold of 3:

order 1: apple, egg, milk

order 2: carrot, milk

order 3: apple, egg, carrot

order 4: apple, egg

order 5: apple, carrot

Iteration 1:

Count the number of times each item occurs

item set	occurrence count
{apple}	4
{egg}	3
{milk}	2
{carrot}	2

{milk} and {carrot} are eliminated because they do not meet the minimum occurrence threshold.

Iteration 2:

Build item sets of size 2 using the remaining items from Iteration 1

(ie: apple, egg)

item set	occurence count
{apple, egg}	3

Only {apple, egg} remains and the algorithm stops since there are no more items to add.

If we had more orders and items, we could continue to iterate, building item sets consisting of more than 2 elements. For the problem we are trying to solve (ie: finding relationships between pairs of items), it suffices to implement apriori to get to item sets of size 2.

Association Rules Mining

Once the item sets have been generated using apriori, we can start mining association rules. Given that we are only looking at item sets of size 2, the association rules we will generate will be of the form {A} -

> {B}. One common application of these rules is in the domain of recommender systems, where customers who purchased item A are recommended item B.

Here are 3 key metrics to consider when evaluating association rules:

1. Support

This is the percentage of orders that contains the item set. In the example above, there are 5 orders in total and {apple,egg} occurs in 3 of them, so:

```
support{apple,egg} = 3/5 or 60%
```

The minimum support threshold required by apriori can be set based on knowledge of your domain. In this grocery dataset for example, since there could be thousands of distinct items and an order can contain only a small fraction of these items, setting the support threshold to 0.01% may be reasonable.

2. Confidence

Given two items, A and B, confidence measures the percentage of times that item B is purchased, given that item A was purchased. This is expressed as:

```
confidence{A->B} = support{A,B} / support{A}
```

Confidence values range from 0 to 1, where 0 indicates that B is never purchased when A is purchased, and 1 indicates that B is always purchased whenever A is purchased. Note that the confidence measure is directional. This means that we can also compute the percentage of times that item A is purchased, given that item B was purchased:

```
confidence{B->A} = support{A,B} / support{B}
```

In our example, the percentage of times that egg is purchased, given that apple was purchased is:

```
confidence{apple->egg} = support{apple,egg} / support{apple}

                       = (3/5) / (4/5)

                       = 0.75 or 75%
```

A confidence value of 0.75 implies that out of all orders that contain apple, 75% of them also contain egg. Now, we look at the confidence measure in the opposite direction (ie: egg->apple):

```
confidence{egg->apple} = support{apple,egg} / support{egg}

                       = (3/5) / (3/5)

                       = 1 or 100%
```

Here we see that all of the orders that contain egg also contain apple. But, does this mean that there is a relationship between these two items, or are they occurring together in the same orders simply by chance? To answer this question, we look at another measure which takes into account the popularity of *both* items.

3. Lift

Given two items, A and B, lift indicates whether there is a relationship between A and B, or whether the two items are occuring together in the same orders simply by chance (ie: at random). Unlike the confidence metric whose value may vary depending on direction (eg: confidence{A->B} may be different from confidence{B->A}), lift has no direction. This means that the lift{A,B} is always equal to the lift{B,A}:

```
lift{A,B} = lift{B,A} = support{A,B} / (support{A} * support{B})
```

In our example, we compute lift as follows:

```
lift{apple,egg} = lift{egg,apple} = support{apple,egg} / (support{apple} * support{egg})

                                  = (3/5) / (4/5 * 3/5)

                                  = 1.25
```

One way to understand lift is to think of the denominator as the likelihood that A and B will appear in the same order if there was *no* relationship between them. In the example above, if apple occurred in 80% of the orders and egg occurred in 60% of the orders, then if there was no relationship between them, we would *expect* both of them to show up together in the same order 48% of the time (ie: 80% * 60%). The numerator, on the other hand, represents how often apple and egg *actually* appear together in the same order. In this example, that is 60% of the time. Taking the numerator and dividing it by the denominator, we get to how many more times apple and egg actually appear in the same order, compared to if there was no relationship between them (ie: that they are occurring together simply at random).

In summary, lift can take on the following values:

```
 * lift = 1 implies no relationship between A and B.

   (ie: A and B occur together only by chance)


 * lift > 1 implies that there is a positive relationship between A and B.

   (ie:  A and B occur together more often than random)


 * lift < 1 implies that there is a negative relationship between A and B.

   (ie:  A and B occur together less often than random)
```

In our example, apple and egg occur together 1.25 times *more* than random, so we conclude that there exists a positive relationship between them.