Q1. (6 points)

(1) (3 points)

Suppose that a transaction database $DB$ is partitioned into $DB_1, ..., DB_p$. The outline of a distributed algorithm is as follows.

1. Find sets $S_1, ..., S_p$ of local frequent patterns from $DB_1, ..., DB_p$ using the Apriori or FP-growth algorithms.
2. At the server, collect $S_1, ..., S_p$ together with their local counts, and then, distribute $S_1, ..., S_p$ to the $p$-1 other locations if the site has not reported the count of the pattern.
3. At $i$-th location ($1 \leq i \leq p$), get the local count of each pattern in $S_1, S_{i-1}, S_{i+1}, ..., S_p$, (if the site has not reported so) and then, pass these local counts to the server.
4. At the server, aggregate these local counts in order to get the global support for each pattern in $S_1 \cup ... \cup S_p$.
5. Check each pattern whether its global support exceeds a given threshold.

Note: A more efficient algorithm can be found in the literature: for example, Cheung, D. W., Han, J., Ng, V. T. Y., Fu, A. W., and Fu, Y., “A Fast Distributed Algorithm for Mining Association Rules,” In Proc. Int’l Conf. on Parallel and Distributed Information Systems, Miami Beach, Florida, pp. 31~42, Dec. 1996.
(2) (3 points)

Suppose that we keep local counts for every global frequent pattern in the server, and the transaction data have been incrementally updated at \(i\)-th location.

1. At \(i\)-th location, find a set \(S_i\) of local frequent patterns.
   - Here, we can run an algorithm designed for updated databases. Such algorithm obviates the need for rescanning \(DB_i\) exhaustively.

2. At the server, collect \(S_i\) together with their local counts, and then, check whether each pattern is new or not.
   2.1. For new patterns, distribute them to the \(p-1\) other locations in order to obtain the global support just as above.
   2.2. For existing patterns, update their local counts at \(i\)-th location, which are maintained in the server. Then, the global support can be calculated.

3. Check each pattern in only \(S_i\) whether its global support exceeds a given threshold.

Reference: Below is an example of incremental mining algorithms.

Note: You can get full marks for Q1 if your extensions are reasonable.
Q2. (6 points)

(1) (3 points)

The largest $k$ is 3, and the frequent 3-itemset is {Milk, Bread, Cheese}. The strong association rules are as follows.

$\{\text{Bread, Cheese}\} \rightarrow \{\text{Milk}\} \ [0.75, 1.0]$  
$\{\text{Milk, Cheese}\} \rightarrow \{\text{Bread}\} \ [0.75, 1.0]$  
$\{\text{Cheese}\} \rightarrow \{\text{Milk, Bread}\} \ [0.75, 1.0]$  

(2) (3 points)

The largest $k$ is 3, and the frequent 3-itemsets are as follows.

$\{\text{Dairyland-Milk, Wonder-Bread, Tasty-Pie}\}$  
$\{\text{Wonder-Bread, Sunset-Milk, Dairyland-Cheese}\}$
Q3. (6 points)

(1) (3 points)

Suppose that $ab$ represents the number of transactions containing both items A and B; $a\overline{b}$ the number of the ones containing only A; $\overline{a}b$ the number of the ones containing only B; and $\overline{a}\overline{b}$ the number of the ones containing neither A nor B. Here, $\overline{a}\overline{b}$ is the number of null transactions.

The coherence, all-confidence, cosine, and $\beta$ are calculated using the following formulas.

$$
\text{coherence} = \frac{ab}{ab + ab + \overline{a}b + \overline{b}a - ab} = \frac{ab}{ab + \overline{a}\overline{b} + \overline{a}b + \overline{b}a} = \frac{ab}{ab + \overline{a}\overline{b} + \overline{b}a + \overline{a}b},
$$

$$
\text{all-confidence} = \frac{ab}{\max(ab + \overline{a}\overline{b}, ab + \overline{b}a)}
$$

$$
\text{cosine} = \frac{ab}{\sqrt{(ab + \overline{a}\overline{b})(ab + \overline{b}a)}}
$$

$$
\beta = \frac{1}{2} \left( \frac{ab}{ab + \overline{a}\overline{b}} + \frac{ab}{ab + \overline{b}a} \right)
$$

We note that $\overline{a}\overline{b}$ is not used in the formulas above. Since these measures are free from the influence of null-transactions, they are all null-invariant.

On the other hand, the lift and $\chi^2$ are calculated using the following formulas.

$$
lift = \frac{ab}{\frac{ab + \overline{a}b + \overline{b}a + \overline{a}\overline{b}}{ab + ab} \times \frac{ab + \overline{b}a}{ab + \overline{a}\overline{b} + \overline{b}a + \overline{a}b}} = \frac{ab}{(ab + \overline{a}\overline{b})(ab + \overline{b}a)}
$$

$$
\chi^2 = \frac{(ab - E_{ab})^2}{E_{ab}} + \frac{(a\overline{b} - E_{\overline{a}b})^2}{E_{\overline{a}b}} + \frac{(\overline{a}b - E_{\overline{b}a})^2}{E_{\overline{b}a}} + \frac{(ab - E_{ab})^2}{E_{ab}}
$$

Since $\overline{a}\overline{b}$ is used in the lift and $\chi^2$, it is obvious that they are not null-invariant.
(2) (3 points)

The measure $\beta$ is the most desirable. $\beta$ treats neutral points fairly and is a balanced measure. Below is an example borrowed from Prof. Han’s slide.

<table>
<thead>
<tr>
<th>Data set</th>
<th>mc</th>
<th>mce</th>
<th>me</th>
<th>mce</th>
<th>$\chi^2$</th>
<th>Lift</th>
<th>AllConf</th>
<th>Coherence</th>
<th>Cosine</th>
<th>$\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>10,000</td>
<td>1,000</td>
<td>1,000</td>
<td>100,000</td>
<td>90557</td>
<td>9.26</td>
<td>0.91</td>
<td>0.83</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>$D_2$</td>
<td>10,000</td>
<td>1,000</td>
<td>1,000</td>
<td>100</td>
<td>0</td>
<td>1</td>
<td>0.91</td>
<td>0.83</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>$D_3$</td>
<td>100</td>
<td>1,000</td>
<td>1,000</td>
<td>100,000</td>
<td>870</td>
<td>8.44</td>
<td>0.09</td>
<td>0.05</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>$D_4$</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
<td>100,000</td>
<td>24740</td>
<td>25.75</td>
<td>0.5</td>
<td>0.33</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>$D_5$</td>
<td>1,000</td>
<td>100</td>
<td>10,000</td>
<td>100,000</td>
<td>8173</td>
<td>9.18</td>
<td>0.09</td>
<td>0.09</td>
<td>0.29</td>
<td>0.5</td>
</tr>
<tr>
<td>$D_6$</td>
<td>1,000</td>
<td>10</td>
<td>100,000</td>
<td>100,000</td>
<td>965</td>
<td>1.97</td>
<td>0.01</td>
<td>0.01</td>
<td>0.10</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Note: You can get full marks for Q3.2 if your example is reasonable.
Q4. (6 points)

1. **k-means**
   - **Cluster shapes**: spherical or convex
   - **Input parameters**: \( k \): the number of clusters
   - **Limitations**: Can be applied only when the mean of a cluster is defined; requires users to specify \( k \); not suitable for discovering clusters with nonconvex shapes or those of very different size; sensitive to noise and outlier data points; ...

2. **CLARANS**
   - **Cluster shapes**: spherical or convex
   - **Input parameters**: *maxneighbor*: the maximum number of neighbors examined, *numlocal*: the number of local minima obtained
   - **Limitations**: Dependent on the sampling method used; ...

3. **BIRCH**
   - **Cluster shapes**: spherical
   - **Input parameters**: *branching factor* \( B \): the maximum number of children per nonleaf node in the CF tree, *threshold* \( T \): the maximum diameter of subclusters stored at the leaf nodes of the CF tree
   - **Limitations**: Might not produce a natural cluster; does not perform very well if the clusters are not spherical in shape; ...

4. **ROCK**
   - **Cluster shapes**: arbitrary
   - **Input parameters**: \( \theta \): used to control how close a pair of points must be in order to be considered neighbors
   - **Limitations**: Ignores information regarding cluster proximity; ...

5. **CHAMELEON**
   - **Cluster shapes**: arbitrary
   - **Input parameters**: \( k \): the number of \( k \)-nearest neighbors
   - **Limitations**: May require \( O(n^2) \) time for high-dimensional data; ...

6. **DBSCAN**
   - **Cluster shapes**: arbitrary
   - **Input parameters**: \( \varepsilon \): the radius of an \( \varepsilon \)-neighborhood, *MinPts*: the minimum number of points in an \( \varepsilon \)-neighborhood
   - **Limitations**: Sensitive to parameter values; ...
Q5. (6 points)

(1) (3 points)

We had better choose $k$-medoids rather than $k$-means. In $k$-means, the center of points in a cluster may be at an obstacle (e.g., at the center of a river). On the other hand, in $k$-medoids, such problem does not occur since this algorithm selects an object within the cluster as a center.

We modify the distance function so as to consider obstacles. For example, the obstacle distance can be defined as the length of the shortest Euclidean path from a starting point and to an ending point without cutting any obstacles. Then, we run the $k$-medoids algorithm using the obstacle distance.

Reference:
We introduce a modification of $k$-means so as to handle constraints. This algorithm transforms constrained $k$-means clustering into linear network optimization (more specifically, the minimum cost flow problem). Here, the constraint is that each cluster should have at least $\tau^h$ points.

The figure below shows the equivalent minimum cost flow formulation. A data point $x^i$ corresponds to a supply node with supply = 1. A cluster $C^h$ corresponds to a demand node with demand = $-\tau^h$. The cost on an arc is the distance between a data point and the center of its cluster. Then, we find the optimal solution using a simplex algorithm.

Reference:

Note: You can get full marks for Q5 if your modifications are reasonable.