Partitional Clustering vs Hierarchical Clustering

	Partitioned Clustering	**Hierarchical Clustering**
Goal	Group data into a fixed number of clusters, maximizing similarity within clusters and dissimilarity between clusters	Create a hierarchy of nested clusters without requiring a predetermined number of clusters
Algorithm	K-means, K-medoids, Fuzzy C-means	Agglomerative, Divisive
Scalability	More scalable for large datasets	Can handle datasets of any size
Interpretability	Easier to interpret and understand	More difficult to interpret
Flexibility	Less flexible, requires a predetermined number of clusters	More flexible, can handle unknown data structure and provide more informative visualization
Advantages	Speed, scalability, interpretability	Flexibility, informative visualization
Disadvantages	Limited flexibility, requires a predetermined number of clusters	Computationally expensive for large datasets, less interpretable

Method	General Characteristics
Partitioning methods	– Find mutually exclusive clusters of spherical shape – Distance-based – May use mean or medoid (etc.) to represent cluster center – Effective for small- to medium-size data sets
Hierarchical methods	– Clustering is a hierarchical decomposition (i.e., multiple levels) – Cannot correct erroneous merges or splits – May incorporate other techniques like microclustering or consider object "linkages"
Density-based methods	– Can find arbitrarily shaped clusters – Clusters are dense regions of objects in space that are separated by low-density regions – Cluster density: Each point must have a minimum number of points within its "neighborhood" – May filter out outliers
Grid-based methods	– Use a multiresolution grid data structure – Fast processing time (typically independent of the number of data objects, yet dependent on grid size)

PCY vs Apriori Algorithm

	Apriori Algorithm	**PCY Algorithm**
Goal	Find frequent itemsets in a dataset	Find frequent itemsets in a dataset
Algorithm	Iterative, generates all possible combinations of items	Hash-based, estimates frequency of pairs of items
Efficiency	Can be computationally expensive for large datasets with many items	Faster and more memory-efficient

Result	Guarantees all frequent itemsets are found	May not find all frequent itemsets depending on dataset sparsity and minimum support threshold
Memory Usage	Memory-intensive, requires storing all candidate itemsets	Memory-efficient, only stores hash table
Advantages	Reliable, works well with many datasets	Efficient, suitable for very large datasets
Disadvantages	Slower for large datasets, memory-intensive	May not find all frequent itemsets depending on dataset characteristics