Lab 01 - Techniques for Handling Large Datasets in Clustering

During this lab we will continue our previous study, this time focusing on clustering tasks.

1. The Dataset

We are going to use the same dataset as for the previous lab.

During this lab, we will use the NYC Yellow Taxi Trip Data from Kaggle, which is a subset of the original dataset provided by the NYC Taxi & Limousine Commission (TLC).

2. Preprocess the Dataset

Clean and preprocess the dataset, e.g., handle missing values, engineer new features (e.g., a day of the week from time-related columns), perform discretization if needed, etc. You may approach this task in an iterative manner, i.e., you may implement a first version and then back to it later. You can reuse some of the code from the previous lab.

Clustering task

Prepare a solution for a clustering task. Wherever possible, introduce hyperparameters so that you can easily obtain results for different settings.

What evaluation approach will you use to compare the clustering results?

In [1]:
```
# write your code here
```

3. Implement Different Techniques for Handling Large Datasets

Clustering task

Implement an algorithm from the family of k-means, k-medoids, or k-prototypes, depending on the types of features you have engineered. Remember that the data should be properly preprocessed before applying an algorithm of your choice, e.g., what is the risk of using k-means on features with very different scales? Use hyperparameters such as the number of clusters `k`, the distance measure, the maximum number of iterations, etc.

Techniques for Handling Large Datasets

Use the same techniques for handling large datasets as in the previous lab:

- Process the full dataset in parts - process the dataset in parts, e.g., by processing one column at a time for Naive Bayes (compute the necessary distributions for each column) or by processing batches of rows and creating an ensemble of models.
- Sampling techniques - sample a subset of the dataset, e.g., using random sampling, stratified sampling, - introduce a parameter `FRAC` that defines the fraction of the dataset to be sampled.
- Summarization techniques - apply clustering/gridding/binning to group similar instances and aggregate the values within each cluster/bin. E.g., you can use location-related features (pickup and dropoff coordinates) to create bins and aggregate (you can take the mean for numerical features and the majority value "*mode*" for categorical features) the rides which have pickup and dropoff locations within the same bin.
- Quantization techniques - reduce the number of distinct values for numerical features, e.g., by applying rounding, flooring, ceiling, or other techniques. Store values with reduced precision - it can reduce I/O time and memory usage and consequently speed up the computations.

In [2]:
```
# write your code here
```