Advanced Tutorial on Implementing PFECLAT Algorithm

In this tutorial, we explain how the Periodic Frequent ECLAT (PFECLAT) algorithm can be implemented by varying the minimum support values

Step 1: Import the PFECLAT algorithm and pandas data frame

In [1]:
```
from PAMI.periodicFrequentPattern.basic import PFECLAT  as alg
import pandas as pd
```

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'temporal_T10I4D100K.csv'
seperator='¥t'
maxmunPeriodCount=5000
minimumSupportCountList = [100, 150, 200, 250, 300]
#minimumSupport can also specified between 0 to 1. E.g., minSupList = [0.005, 0.006,

result = pd.DataFrame(columns=['algorithm', 'minSup', 'maxPer','patterns', 'runtime'
#initialize a data frame to store the results of PFECLAT algorithm
```

Step 3: Execute the PFECLAT algorithm using a for loop

In [3]:
```
algorithm = 'PFECLAT'  #specify the algorithm name
for minSupCount in minimumSupportCountList:
    obj = alg.PFECLAT('temporal_T10I4D100K.csv', minSup=minSupCount,maxPer=maxmunPer
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, minSupCount,maxmunPeriodCount, len(obj
```

```
Periodic-Frequent patterns were generated successfully using PFECLAT algorithm
Periodic-Frequent patterns were generated successfully using PFECLAT algorithm
Periodic-Frequent patterns were generated successfully using PFECLAT algorithm
Periodic-Frequent patterns were generated successfully using PFECLAT algorithm
Periodic-Frequent patterns were generated successfully using PFECLAT algorithm
```

In [4]:
```
print(result)
```

```
  algorithm  minSup  maxPer  patterns    runtime     memory
0   PFECLAT     100    5000     25462  33.832275  487723008
1   PFECLAT     150    5000     18982  22.235379  485105664
2   PFECLAT     200    5000     13251  14.308821  487538688
3   PFECLAT     250    5000      7702  10.037130  487669760
4   PFECLAT     300    5000      4552   8.163026  487636992
```

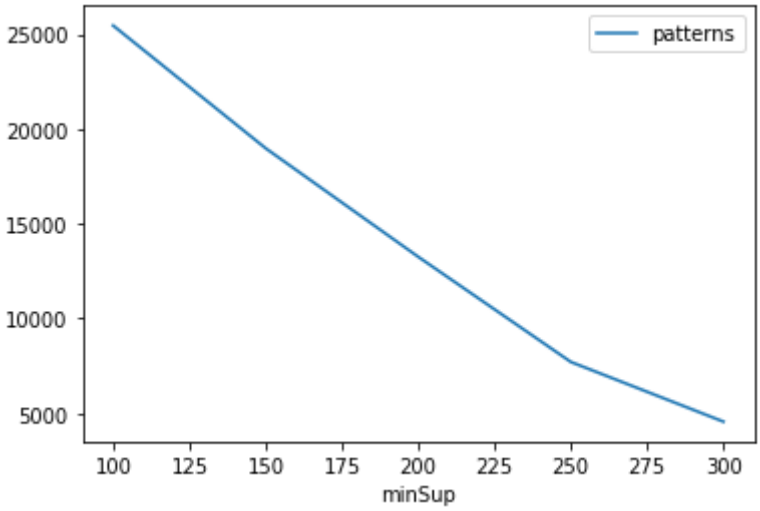
Step 5: Visualizing the results

Step 5.1 Importing the plot library

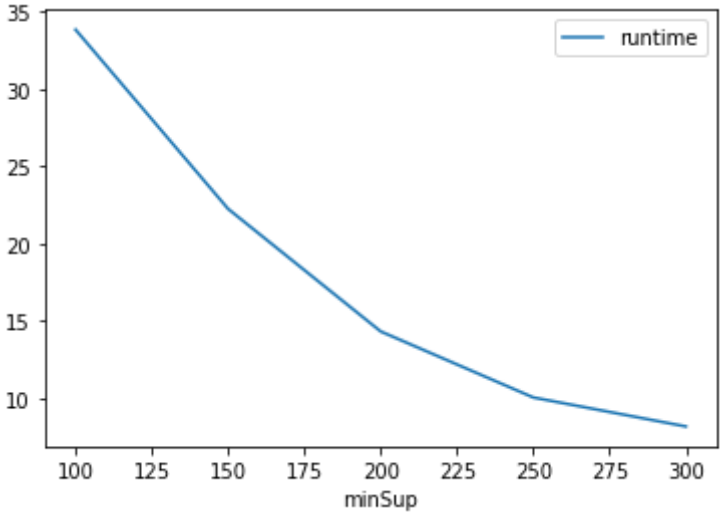
In [5]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

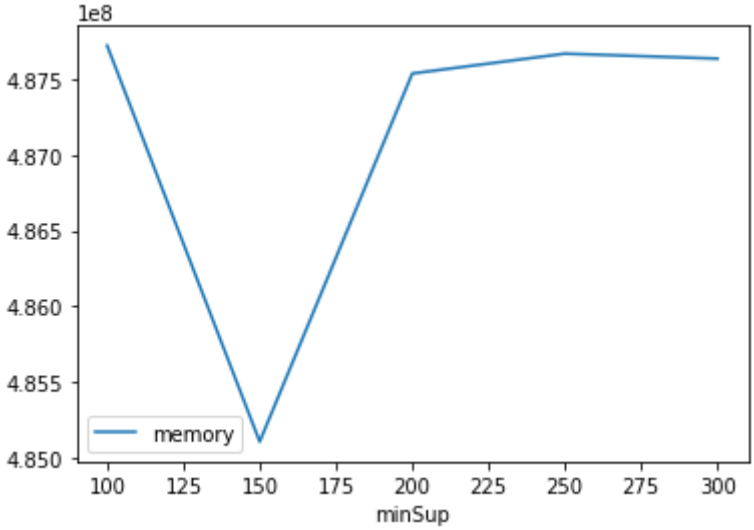
In [6]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



Graph for No Of Patterns is successfully generated!



Graph for Runtime taken is successfully generated!



Graph for memory consumption is successfully generated!

Step 6: Saving the results as latex files

In [7]:
```
from PAMI.extras.graph import generateLatexFileFromDataFrame as gdf
gdf.generateLatexCode(result)
```

```
Latex files generated successfully
```