Advanced Tutorial on Implementing PPPClose Algorithm

In this tutorial, we explain how the PPPClose algorithm can be implemented by varying the minimum support values

Step 1: Import the PPPClose algorithm and pandas data frame

In [1]:
```
from PAMI.partialPeriodicPattern.closed import PPPClose  as alg
import pandas as pd
```

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'temporal_T10I4D100K.csv'
seperator='¥t'
periodCount=500
periodicSupportCountList = [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', 'period','patterns', 'runtime'
#initialize a data frame to store the results of PPPClose algorithm
```

Step 3: Execute the PPPClose algorithm using a for loop

In [3]:
```
algorithm = 'PPPClose'  #specify the algorithm name
for periodicSupportCount in periodicSupportCountList:
    obj = alg.PPPClose('temporal_T10I4D100K.csv', periodicSupport=periodicSupportCou
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, periodicSupportCount,periodCount, len(
    print(result)
```

```
Closed periodic frequent patterns were generated successfully using PPPClose algorit
hm
  algorithm  minSup  period  patterns   runtime     memory
0  PPPClose     100     500     15949  19.58574  228777984
Closed periodic frequent patterns were generated successfully using PPPClose algorit
hm
  algorithm  minSup  period  patterns    runtime     memory
0  PPPClose     100     500     15949  19.585740  228777984
1  PPPClose     150     500     10093  18.424128  227999744
Closed periodic frequent patterns were generated successfully using PPPClose algorit
hm
  algorithm  minSup  period  patterns    runtime     memory
0  PPPClose     100     500     15949  19.585740  228777984
1  PPPClose     150     500     10093  18.424128  227999744
2  PPPClose     200     500      6447  17.333220  227495936
Closed periodic frequent patterns were generated successfully using PPPClose algorit
hm
  algorithm  minSup  period  patterns    runtime     memory
0  PPPClose     100     500     15949  19.585740  228777984
1  PPPClose     150     500     10093  18.424128  227999744
2  PPPClose     200     500      6447  17.333220  227495936
3  PPPClose     250     500      3801  16.050803  226426880
Closed periodic frequent patterns were generated successfully using PPPClose algorit
hm
  algorithm  minSup  period  patterns    runtime     memory
0  PPPClose     100     500     15949  19.585740  228777984
1  PPPClose     150     500     10093  18.424128  227999744
2  PPPClose     200     500      6447  17.333220  227495936
3  PPPClose     250     500      3801  16.050803  226426880
4  PPPClose     300     500      2545  15.293038  225890304
```

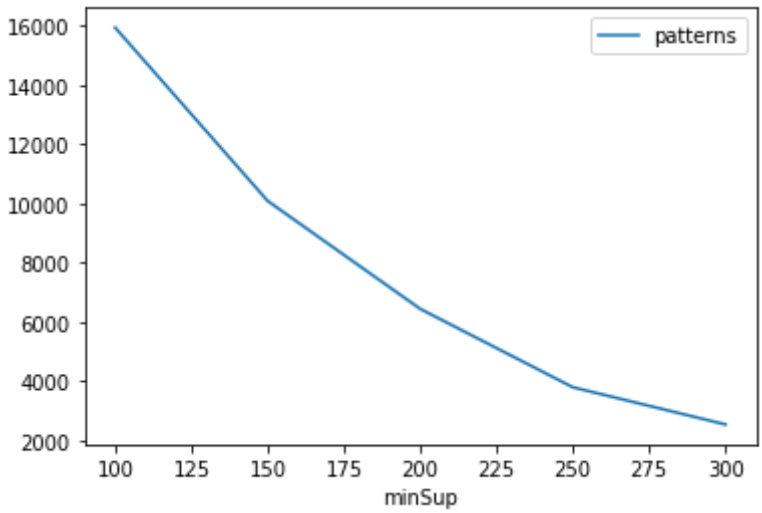
Step 5: Visualizing the results

Step 5.1 Importing the plot library

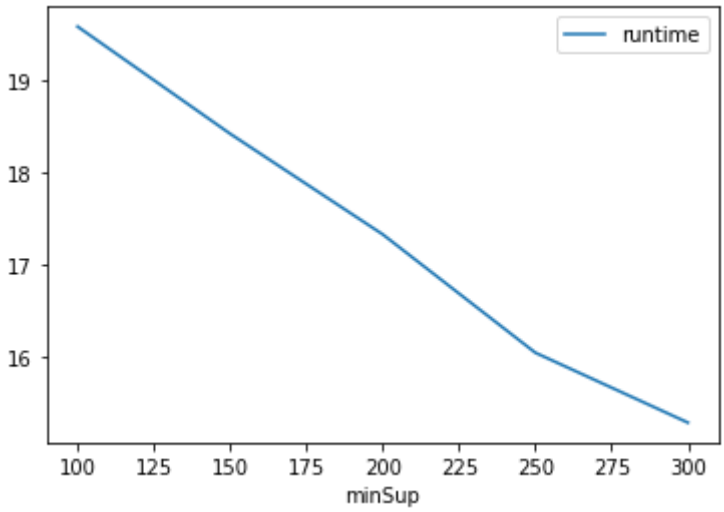
In [4]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

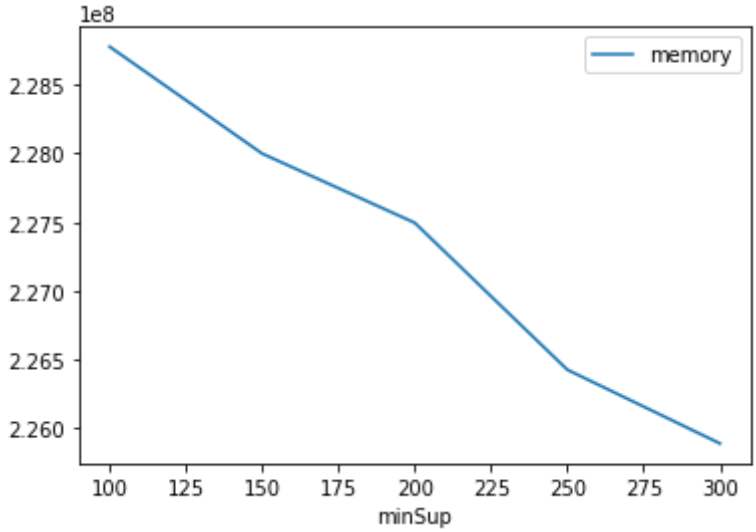
In [5]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```




```
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 [6]:
```
from PAMI.extras.graph import generateLatexFileFromDataFrame as gdf
gdf.generateLatexCode(result)
```

Latex files generated successfully