Advanced Tutorial on Implementing PPP_ECLAT Algorithm

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

Step 1: Import the PPP_ECLAT algorithm and pandas data frame

In [1]:
```
from PAMI.partialPeriodicPattern.basic import PPP_ECLAT  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 PPP_ECLAT algorithm
```

Step 3: Execute the PPP_ECLAT algorithm using a for loop

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

```
Partial Periodic Frequent patterns were generated successfully using 3PEclat algorit
hm
   algorithm  minSup  period  patterns    runtime     memory
0  PPP_ECLAT     100     500     16157  24.899616  225271808
Partial Periodic Frequent patterns were generated successfully using 3PEclat algorit
hm
   algorithm  minSup  period  patterns    runtime     memory
0  PPP_ECLAT     100     500     16157  24.899616  225271808
1  PPP_ECLAT     150     500     10227  23.130321  225390592
Partial Periodic Frequent patterns were generated successfully using 3PEclat algorit
hm
   algorithm  minSup  period  patterns    runtime     memory
0  PPP_ECLAT     100     500     16157  24.899616  225271808
1  PPP_ECLAT     150     500     10227  23.130321  225390592
2  PPP_ECLAT     200     500      6498  21.629186  225374208
Partial Periodic Frequent patterns were generated successfully using 3PEclat algorit
hm
   algorithm  minSup  period  patterns    runtime     memory
0  PPP_ECLAT     100     500     16157  24.899616  225271808
1  PPP_ECLAT     150     500     10227  23.130321  225390592
2  PPP_ECLAT     200     500      6498  21.629186  225374208
3  PPP_ECLAT     250     500      3810  20.336407  225333248
Partial Periodic Frequent patterns were generated successfully using 3PEclat algorit
hm
   algorithm  minSup  period  patterns    runtime     memory
0  PPP_ECLAT     100     500     16157  24.899616  225271808
1  PPP_ECLAT     150     500     10227  23.130321  225390592
2  PPP_ECLAT     200     500      6498  21.629186  225374208
3  PPP_ECLAT     250     500      3810  20.336407  225333248
4  PPP_ECLAT     300     500      2554  19.138694  225083392
```

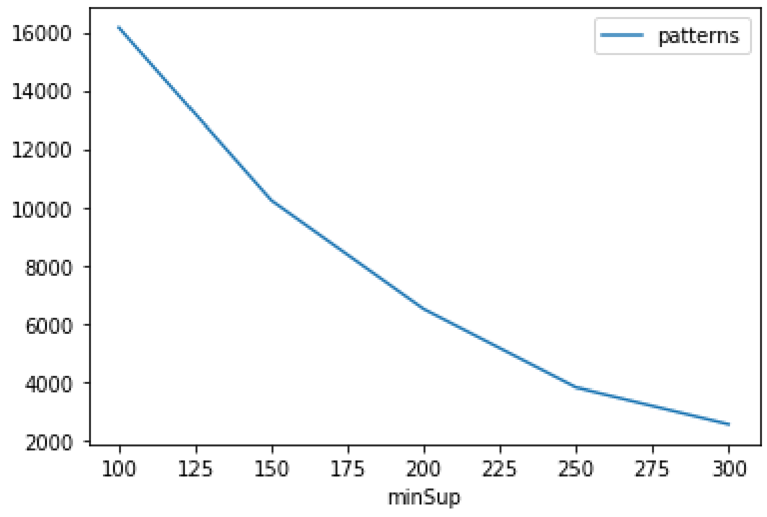
Step 5: Visualizing the results

Step 5.1 Importing the plot library

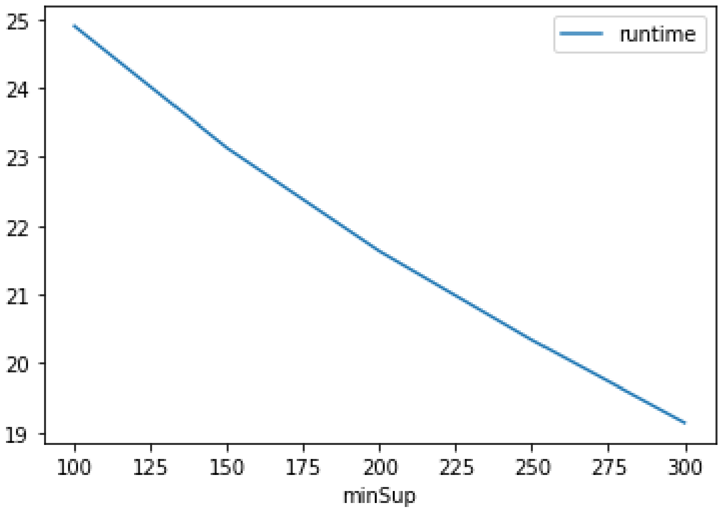
In [4]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

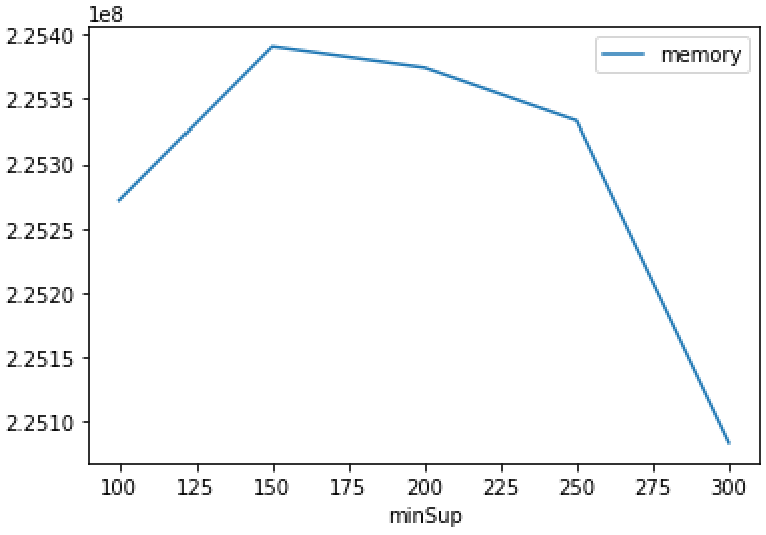
In [5]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



```
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
```