Advanced Tutorial on Implementing ECLAT Algorithm

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

Step 1: Import the ECLAT algorithm and pandas data frame

In [1]:
```
from PAMI.frequentPattern.basic import ECLAT  as alg
import pandas as pd
```

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'transactional_T10I4D100K.csv'
seperator='¥t'
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', 'patterns', 'runtime', 'memory
#initialize a data frame to store the results of ECLAT algorithm
```

Step 3: Execute the ECLAT algorithm using a for loop

In [3]:
```
algorithm = 'ECLAT'  #specify the algorithm name
for minSupCount in minimumSupportCountList:
    obj = alg.ECLAT('transactional_T10I4D100K.csv', minSup=minSupCount, sep=seperato
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, minSupCount, len(obj.getPatterns()), o
```

```
Frequent patterns were generated successfully using ECLAT algorithm
Frequent patterns were generated successfully using ECLAT algorithm
Frequent patterns were generated successfully using ECLAT algorithm
Frequent patterns were generated successfully using ECLAT algorithm
Frequent patterns were generated successfully using ECLAT algorithm
```

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

```
  algorithm  minSup  patterns    runtime     memory
0     ECLAT     100     27532  8.705271  582139904
1     ECLAT     150     19126  7.865161  509947904
2     ECLAT     200     13255  7.432579  459800576
3     ECLAT     250      7703  7.307075  411705344
4     ECLAT     300      4552  6.954253  384507904
```

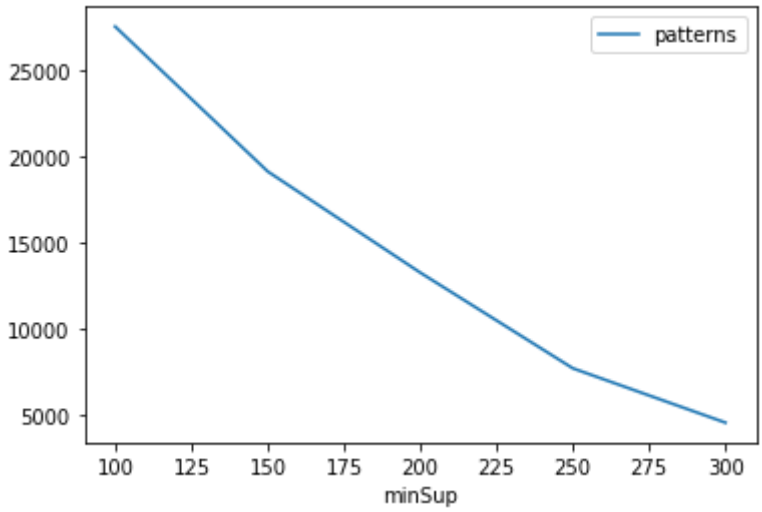
Step 5: Visualizing the results

Step 5.1 Importing the plot library

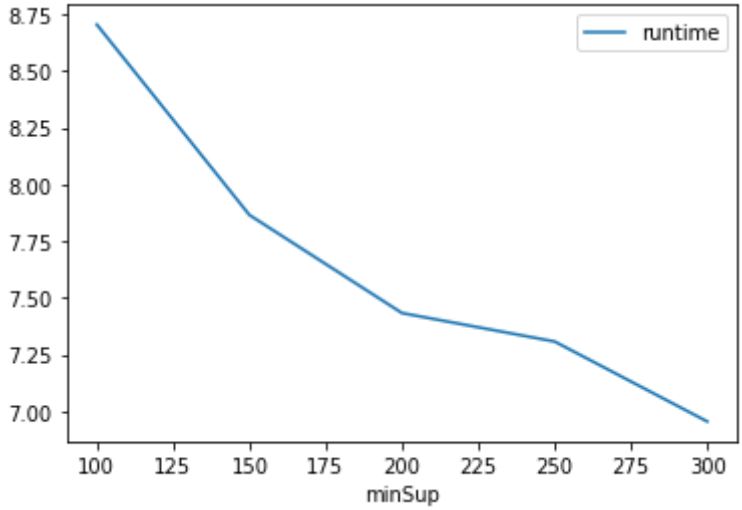
In [5]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

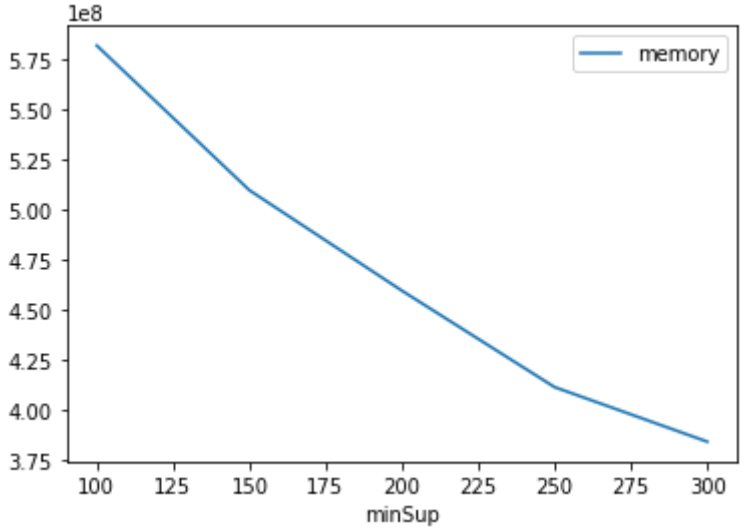
In [6]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



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

In [ ]: