Advanced Tutorial on Implementing CHARM Algorithm

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

Step 1: Import the CHARM algorithm and pandas data frame

In [1]:

```
from PAMI.frequentPattern.closed import CHARM  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 CHARM algorithm
```

Step 3: Execute the CHARM algorithm using a for loop

In [3]:

```
algorithm = 'CHARM'  #specify the algorithm name
for minSupCount in minimumSupportCountList:
    obj = alg.CHARM('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()), c
```

```
Closed Frequent patterns were generated successfully using CHARM algorithm
Closed Frequent patterns were generated successfully using CHARM algorithm
Closed Frequent patterns were generated successfully using CHARM algorithm
Closed Frequent patterns were generated successfully using CHARM algorithm
Closed Frequent patterns were generated successfully using CHARM algorithm
```

In [4]:

```
print(result)
```

```
  algorithm  minSup  patterns    runtime     memory
0     CHARM     100     17145  18.463059  140075008
1     CHARM     150     12356  17.562555  140505088
2     CHARM     200      8713  16.763453  140963840
3     CHARM     250      4969  15.939386  140640256
4     CHARM     300      2865  15.101360  140718080
```

Step 5: Visualizing the results

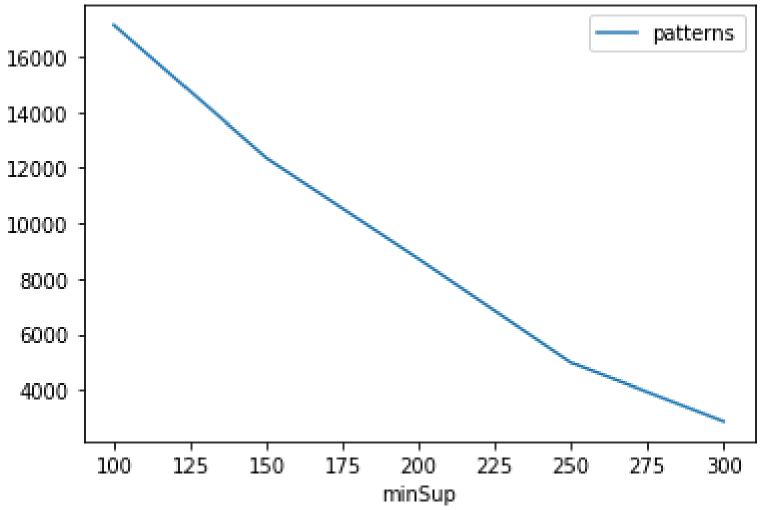
Step 5.1 Importing the plot library

In [5]:

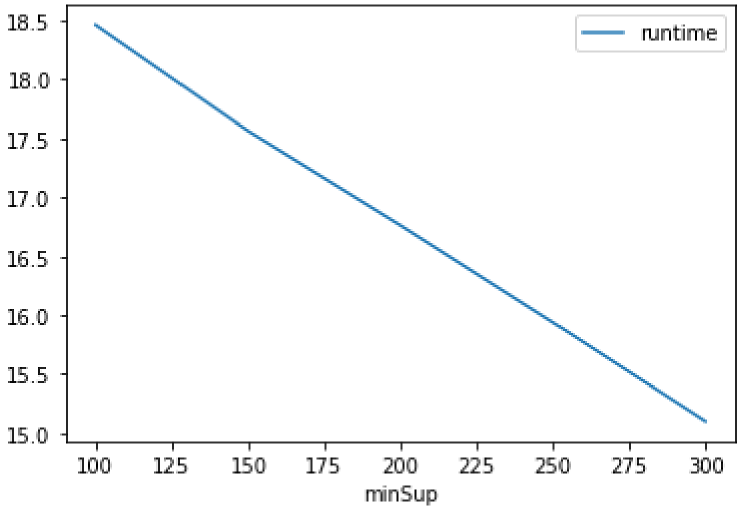
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

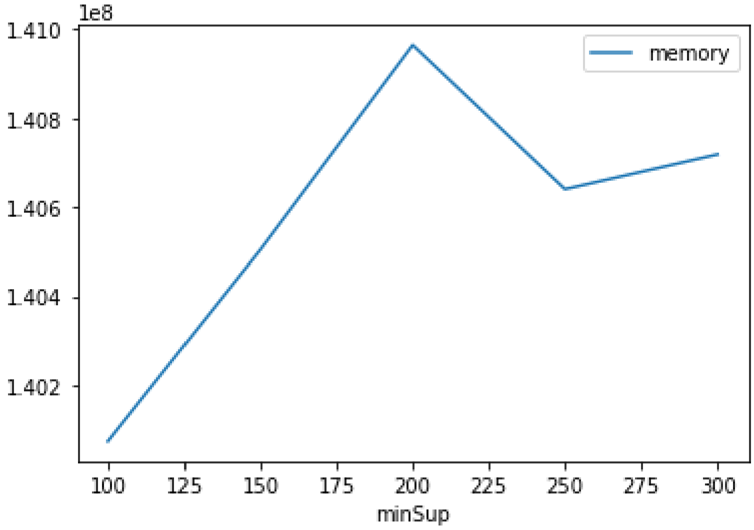
In [6]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



```
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 [ ]:
```