Unsupervised

In [25]:

```
import numpy as np
import pandas as pd

#sklearn imports
from sklearn.decomposition import PCA #Principal Component Analysis
from sklearn.manifold import TSNE #T-Distributed Stochastic Neighbor Embedding
from sklearn.cluster import KMeans #K-Means Clustering
from sklearn.preprocessing import StandardScaler #used for 'Feature Scaling'
from sklearn.datasets import make_blobs


#plotly imports
import plotly as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

import matplotlib.pyplot as plt
```

In [26]:

```
df = pd.read_csv("dataSet.csv")
df = df.drop(['DATABASE_NAME','Unkown','USER_NAME','SESSION_ID','PLAN_ID','SQL_TEXT
df.head()
```

Out[26]:

	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_SIZE
0	0	1	0	0
1	4	4	0	0
2	0	1	0	0
3	0	1	0	0
4	0	1	0	0

In [27]:

```
#Initialize our model
kmeans = KMeans(n_clusters=5)
```

In [28]:

```
#Fit our model
kmeans.fit(df)
```

Out[28]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='au
to',
       random_state=None, tol=0.0001, verbose=0)
```

In [29]:

```
#Find which cluster each data-point belongs to
clusters = kmeans.predict(df)
clusters
```

Out[29]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int32)
```

In [43]:

```
#Add the cluster vector to our DataFrame, X
df["Cluster"] = clusters
df.to_csv('data1.csv')
df.head()
```

Out[43]:

	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_SIZE	Cluster
0	0	1	0	0	0
1	4	4	0	0	0
2	0	1	0	0	0
3	0	1	0	0	0
4	0	1	0	0	0

In [31]:

```
#plotX is a DataFrame containing 3000 values sampled randomly from X
plotX = pd.DataFrame(np.array(df.sample(3000)))

#Rename plotX's columns since it was briefly converted to an np.array above
plotX.columns = df.columns
```

In [32]:

```
#PCA with one principal component
pca_1d = PCA(n_components=1)

#PCA with two principal components
pca_2d = PCA(n_components=2)

#PCA with three principal components
pca_3d = PCA(n_components=3)
```

In [33]:

```
#This DataFrame holds that single principal component mentioned above
PCs_1d = pd.DataFrame(pca_1d.fit_transform(plotX.drop(["Cluster"], axis=1)))

#This DataFrame contains the two principal components that will be used
#for the 2-D visualization mentioned above
PCs_2d = pd.DataFrame(pca_2d.fit_transform(plotX.drop(["Cluster"], axis=1)))

#And this DataFrame contains three principal components that will aid us
#in visualizing our clusters in 3-D
PCs_3d = pd.DataFrame(pca_3d.fit_transform(plotX.drop(["Cluster"], axis=1)))
```

In [34]:

```


PCs_1d.columns = ["PC1_1d"]

#"PC1_2d" means: 'The first principal component of the components created for 2-D v
#And "PC2_2d" means: 'The second principal component of the components created for
PCs_2d.columns = ["PC1_2d", "PC2_2d"]

PCs_3d.columns = ["PC1_3d", "PC2_3d", "PC3_3d"]
```

In [35]:

```
plotX = pd.concat([plotX,PCs_1d,PCs_2d,PCs_3d], axis=1, join='inner')
```

In [36]:

```
plotX["dummy"] = 0
```

In [37]:

```
#Note that all of the DataFrames below are sub-DataFrames of 'plotX'.
#This is because we intend to plot the values contained within each of these DataFr

cluster0 = plotX[plotX["Cluster"] == 0]
cluster1 = plotX[plotX["Cluster"] == 1]
cluster2 = plotX[plotX["Cluster"] == 2]
cluster3 = plotX[plotX["Cluster"] == 3]
cluster4 = plotX[plotX["Cluster"] == 4]
```

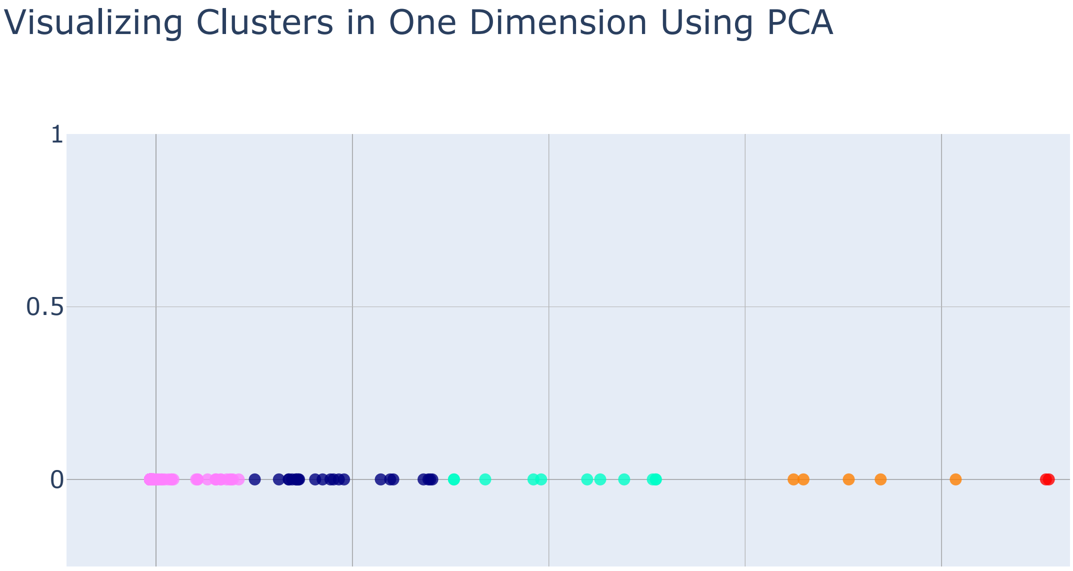
In [38]:

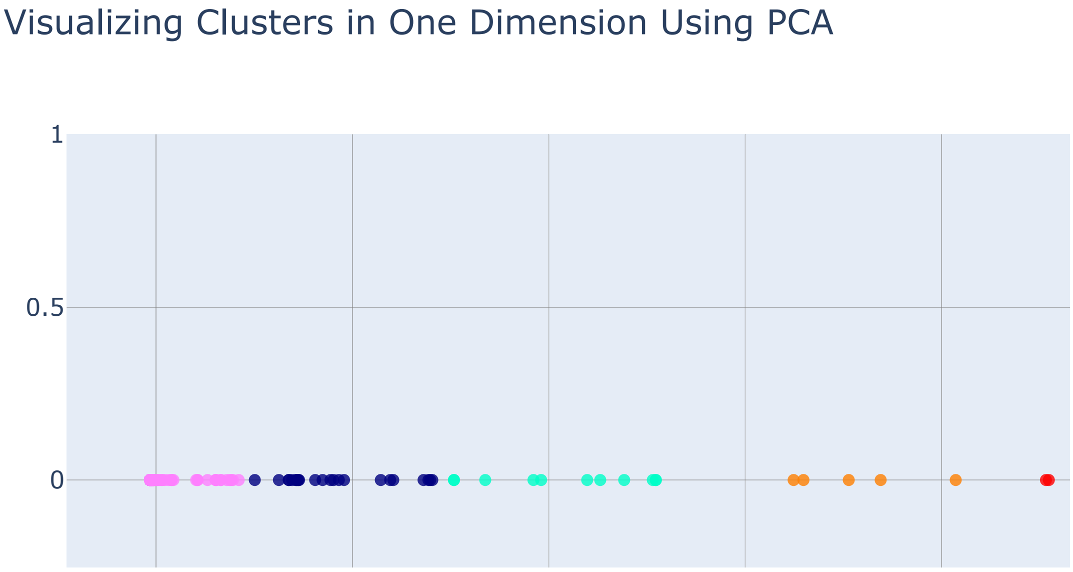
```
#This is needed so we can display plotly plots properly
init_notebook_mode(connected=True)
```

In [39]:

```
#Instructions for building the 1-D plot

#trace1 is for 'Cluster 0'
trace1 = go.Scatter(
                    x = cluster0["PC1_1d"],
                    y = cluster0["dummy"],
                    mode = "markers",
                    name = "Cluster 0",
                    marker = dict(color = 'rgba(255, 128, 255, 0.8)'),
                    text = None)

#trace2 is for 'Cluster 1'
trace2 = go.Scatter(
                    x = cluster1["PC1_1d"],
                    y = cluster1["dummy"],
                    mode = "markers",
                    name = "Cluster 1",
                    marker = dict(color = 'rgba(255, 128, 2, 0.8)'),
                    text = None)

#trace3 is for 'Cluster 2'
trace3 = go.Scatter(
                    x = cluster2["PC1_1d"],
                    y = cluster2["dummy"],
                    mode = "markers",
                    name = "Cluster 2",
                    marker = dict(color = 'rgba(0, 255, 200, 0.8)'),
                    text = None)

#trace4 is for 'Cluster 3'
trace4 = go.Scatter(
                    x = cluster3["PC1_1d"],
                    y = cluster3["dummy"],
                    mode = "markers",
                    name = "Cluster 3",
                    marker = dict(color = 'rgba(255, 0, 0, 0.8)'),
                    text = None)

#trace4 is for 'Cluster 4'
trace5 = go.Scatter(
                    x = cluster4["PC1_1d"],
                    y = cluster4["dummy"],
                    mode = "markers",
                    name = "Cluster 4",
                    marker = dict(color = 'rgba(0, 0, 128, 0.8)'),
                    text = None)

data = [trace1, trace2, trace3, trace4, trace5]

title = "Visualizing Clusters in One Dimension Using PCA"

layout = dict(title = title,
              xaxis= dict(title= 'PC1',ticklen= 5,zeroline= False),
              yaxis= dict(title= '',ticklen= 5,zeroline= False)
             )

fig = dict(data = data, layout = layout)

iplot(fig)
```

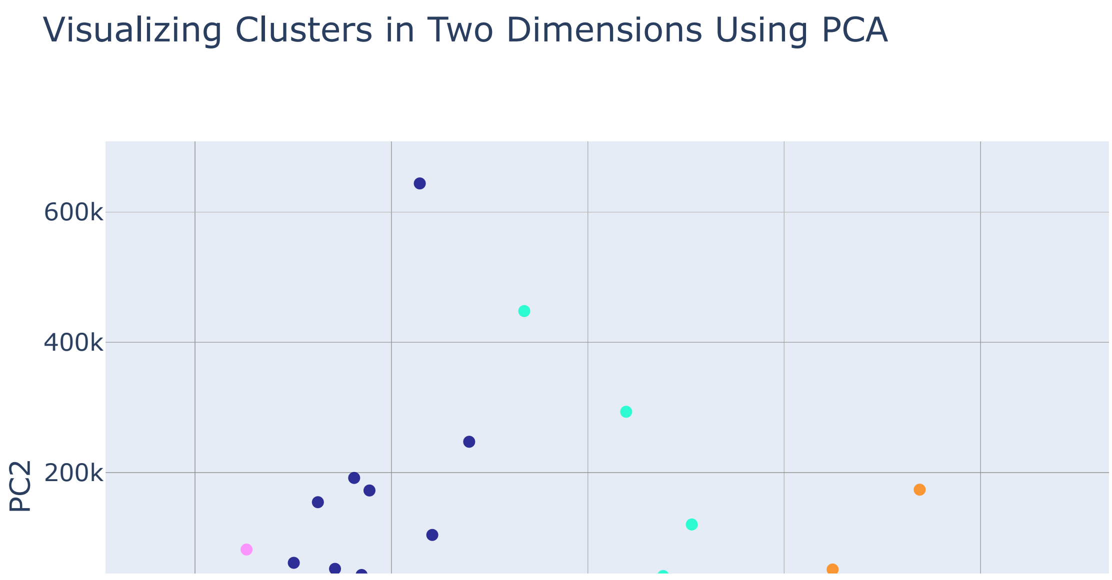
```
plt.show(iplot(fig))
```





In [40]:

```
#Instructions for building the 2-D plot

#trace1 is for 'Cluster 0'
trace1 = go.Scatter(
                    x = cluster0["PC1_2d"],
                    y = cluster0["PC2_2d"],
                    mode = "markers",
                    name = "Cluster 0",
                    marker = dict(color = 'rgba(255, 128, 255, 0.8)'),
                    text = None)

#trace2 is for 'Cluster 1'
trace2 = go.Scatter(
                    x = cluster1["PC1_2d"],
                    y = cluster1["PC2_2d"],
                    mode = "markers",
                    name = "Cluster 1",
                    marker = dict(color = 'rgba(255, 128, 2, 0.8)'),
                    text = None)

#trace3 is for 'Cluster 2'
trace3 = go.Scatter(
                    x = cluster2["PC1_2d"],
                    y = cluster2["PC2_2d"],
                    mode = "markers",
                    name = "Cluster 2",
                    marker = dict(color = 'rgba(0, 255, 200, 0.8)'),
                    text = None)

#trace4 is for 'Cluster 3'
trace4 = go.Scatter(
                    x = cluster3["PC1_2d"],
                    y = cluster3["PC2_2d"],
                    mode = "markers",
                    name = "Cluster 3",
                    marker = dict(color = 'rgba(255, 0, 0, 0.8)'),
                    text = None)

#trace4 is for 'Cluster 4'
trace5 = go.Scatter(
                    x = cluster4["PC1_2d"],
                    y = cluster4["PC2_2d"],
                    mode = "markers",
                    name = "Cluster 4",
                    marker = dict(color = 'rgba(0, 0, 128, 0.8)'),
                    text = None)

data = [trace1, trace2, trace3, trace4, trace5]

title = "Visualizing Clusters in Two Dimensions Using PCA"

layout = dict(title = title,
              xaxis= dict(title= 'PC1',ticklen= 5,zeroline= False),
              yaxis= dict(title= 'PC2',ticklen= 5,zeroline= False)
             )

fig = dict(data = data, layout = layout)

iplot(fig)
```

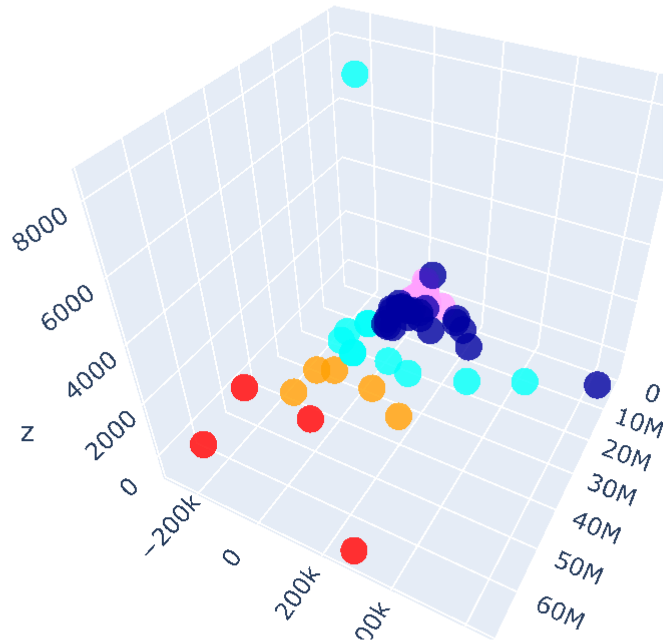

Visualizing Clusters in Two Dimensions Using PCA
600k
400k
200k
PC2

In [41]:

```
#Instructions for building the 3-D plot

#trace1 is for 'Cluster 0'
trace1 = go.Scatter3d(
                    x = cluster0["PC1_3d"],
                    y = cluster0["PC2_3d"],
                    z = cluster0["PC3_3d"],
                    mode = "markers",
                    name = "Cluster 0",
                    marker = dict(color = 'rgba(255, 128, 255, 0.8)'),
                    text = None)

#trace2 is for 'Cluster 1'
trace2 = go.Scatter3d(
                    x = cluster1["PC1_3d"],
                    y = cluster1["PC2_3d"],
                    z = cluster1["PC3_3d"],
                    mode = "markers",
                    name = "Cluster 1",
                    marker = dict(color = 'rgba(255, 128, 2, 0.8)'),
                    text = None)

#trace3 is for 'Cluster 2'
trace3 = go.Scatter3d(
                    x = cluster2["PC1_3d"],
                    y = cluster2["PC2_3d"],
                    z = cluster2["PC3_3d"],
                    mode = "markers",
                    name = "Cluster 2",
                    marker = dict(color = 'rgba(0, 255, 200, 0.8)'),
                    text = None)

#trace4 is for 'Cluster 3'
trace4 = go.Scatter3d(
                    x = cluster3["PC1_3d"],
                    y = cluster3["PC2_3d"],
                    z = cluster3["PC3_3d"],
                    mode = "markers",
                    name = "Cluster 3",
                    marker = dict(color = 'rgba(255, 0, 0, 0.8)'),
                    text = None)

#trace5 is for 'Cluster 4'
trace5 = go.Scatter3d(
                    x = cluster4["PC1_3d"],
                    y = cluster4["PC2_3d"],
                    z = cluster4["PC3_3d"],
                    mode = "markers",
                    name = "Cluster 4",
                    marker = dict(color = 'rgba(0, 0, 128, 0.8)'),
                    text = None)

data = [trace1, trace2, trace3, trace4, trace5]

title = "Visualizing Clusters in Three Dimensions Using PCA"

layout = dict(title = title,
              xaxis= dict(title= 'PC1',ticklen= 5,zeroline= False),
              yaxis= dict(title= 'PC2',ticklen= 5,zeroline= False)
```

```
            )

fig = dict(data = data, layout = layout)

iplot(fig)
```

Visualizing Clusters in Three Dimensions Using PCA



In [47]:

```
kMeansOut = pd.read_csv("data1.csv")
kMeansOut.head()
```

Out[47]:

	Unnamed: 0	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_SIZE	Clu
0	0	0	1	0	0	
1	1	4	4	0	0	
2	2	0	1	0	0	
3	3	0	1	0	0	
4	4	0	1	0	0	

In [63]:

```
kMeansOut.describe()
```

Out[63]:

	Unnamed: 0	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_
count	49028.000000	49028.000000	49028.000000	4.902800e+04	4.90280(
mean	24513.500000	4.170066	1.673758	3.559961e+03	3.10461
std	14153.308836	103.266710	4.314721	4.796966e+04	2.87304(
min	0.000000	0.000000	1.000000	0.000000e+00	0.00000(
25%	12256.750000	0.000000	1.000000	0.000000e+00	0.00000(
50%	24513.500000	0.000000	1.000000	0.000000e+00	0.00000(
75%	36770.250000	1.000000	1.000000	0.000000e+00	0.00000(
max	49027.000000	11688.000000	258.000000	3.171818e+06	6.91276

In [50]:

```
clust0=kMeansOut[kMeansOut.Cluster==0]
clust1=kMeansOut[kMeansOut.Cluster==1]
clust2=kMeansOut[kMeansOut.Cluster==2]
clust3=kMeansOut[kMeansOut.Cluster==3]
clust4=kMeansOut[kMeansOut.Cluster==4]
```

In [51]:

```
clust0.describe()
```

Out[51]:

	Unnamed: 0	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_
count	48273.00000	48273.000000	48273.000000	48273.000000	4.827300
mean	24485.85004	2.393947	1.647256	231.444327	1.723108
std	14132.00369	78.388010	4.262306	3922.850105	2.006875
min	0.00000	0.000000	1.000000	0.000000	0.000000
25%	12254.00000	0.000000	1.000000	0.000000	0.000000
50%	24489.00000	0.000000	1.000000	0.000000	0.000000
75%	36719.00000	1.000000	1.000000	0.000000	0.000000
max	49027.00000	11688.000000	258.000000	341681.000000	4.581685

In [52]:

```
clust1.describe()
```

Out[52]:

	Unnamed: 0	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_SI
count	118.000000	118.000000	118.000000	1.180000e+02	1.180000e-
mean	22495.889831	61.245763	2.169492	3.027804e+05	3.340483e-
std	13487.638587	147.590675	2.740345	2.149194e+05	4.454310e-
min	998.000000	1.000000	1.000000	1.316100e+04	2.700535e-
25%	11362.000000	2.000000	1.000000	1.434230e+05	2.953597e-
50%	21718.000000	6.000000	1.000000	2.741810e+05	3.302461e-
75%	31179.250000	29.000000	2.000000	4.076610e+05	3.668609e-
max	48441.000000	725.000000	17.000000	1.097003e+06	4.352997e-

In [53]:

```
clust2.describe()
```

Out[53]:

	Unnamed: 0	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_S
count	209.000000	209.000000	209.000000	2.090000e+02	2.090000e·
mean	25883.937799	78.191388	2.703349	2.199458e+05	2.030581e·
std	14859.967557	602.385619	5.251194	2.263232e+05	3.663402e·
min	200.000000	1.000000	1.000000	7.573000e+03	1.497392e·
25%	12776.000000	2.000000	1.000000	7.559200e+04	1.694257e·
50%	25635.000000	7.000000	1.000000	1.817060e+05	2.012699e·
75%	38977.000000	22.000000	2.000000	2.859770e+05	2.384556e·
max	49022.000000	8668.000000	55.000000	1.606517e+06	2.665077e·

In [54]:

```
clust3.describe()
```

Out[54]:

	Unnamed: 0	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_SI
count	49.000000	49.000000	49.000000	4.900000e+01	4.900000e+
mean	26669.265306	33.530612	1.571429	7.703410e+05	5.463251e+
std	15178.278370	60.090939	0.957427	8.423874e+05	7.111579e+
min	889.000000	1.000000	1.000000	6.196700e+04	4.419489e+
25%	13698.000000	2.000000	1.000000	1.867670e+05	4.885906e+
50%	26251.000000	12.000000	1.000000	4.850400e+05	5.264178e+
75%	41047.000000	21.000000	2.000000	1.158851e+06	6.089890e+
max	48719.000000	206.000000	6.000000	3.171818e+06	6.912768e+

In [55]:

```
clust4.describe()
```

Out[55]:

	Unnamed: 0	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_S
count	379.000000	379.000000	379.000000	379.000000	3.790000e
mean	27628.989446	168.007916	4.340369	115888.583113	9.305656e
std	16038.939295	600.071678	8.312528	140624.455591	2.814440e
min	106.000000	1.000000	1.000000	1698.000000	4.701254e
25%	12442.500000	3.000000	1.000000	29813.500000	7.271504e
50%	28599.000000	18.000000	2.000000	69469.000000	8.996196e
75%	43944.500000	68.500000	4.000000	136107.000000	1.157377e
max	49011.000000	5722.000000	75.000000	789117.000000	1.475504e

In [ ]: