Answer [CM4]

PCA,t-SNe, DBSCAN, K-Means on encodings that we got after last fully connected layers of CNN.

- Here we are getting the encoding from the 11th layer of our new model(last fully connected layer), which will return the 128 size of array(encodings) per image.
- We will run the model on test data and get the test dataset encodings and perform above analysis on this dataset.
- Test data has 10000 instances after the operation we will have 10000 encodings of 128 array size.

In [43]:
```
keras_function = keras.backend.function([model.input], [model.layers[10].ou
train_encodes = keras_function(X_train)
train_encodes_array = train_encodes[0]
df_train = pd.DataFrame(train_encodes_array)
print("Training data encodings :",df_train.shape)
```

```
Training data encodings : (48000, 128)
```

In [44]:
```
test_encodes = keras_function(X_test)
test_encode_array = test_encodes[0]
print("size of test data encodings :",len(test_encode_array))
```

```
size of test data encodings : 10000
```

In [62]:
```
plot_sample_images(train_sample_images,train_sample_labels, "Greys")
```

Figure 1 : Labelled Images

Above is to show that how labelled are assigned to images and we will compare this output to the PCA and t-SNe graphs to get the better idea of the data labelling.

PCA

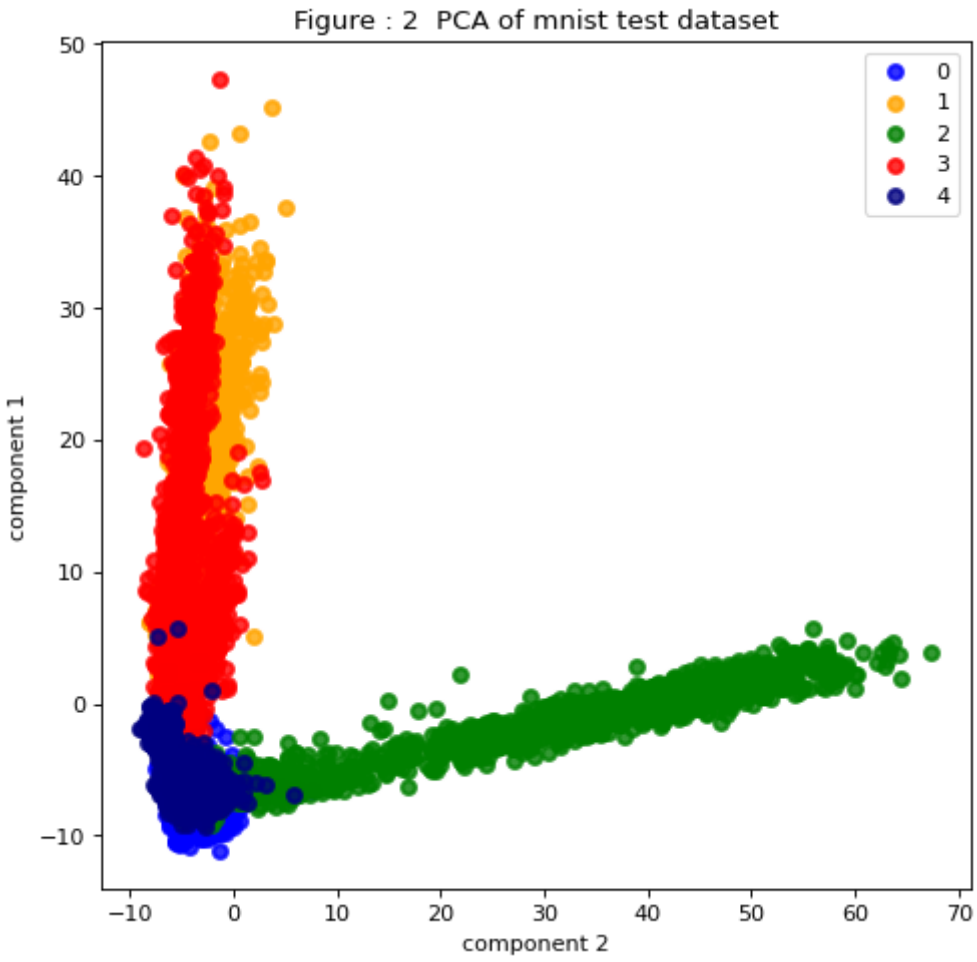
- We will perform PCA on test data and try to get two principal components out of 128 features and plot a graph of assigned labels

In [50]:
```
pca = PCA(n_components= 2)  # we will reduce the data set from 11 columns t
mnist_pca = pca.fit_transform(df_test) # fit means train, fit_transform mea
print(mnist_pca.shape)
```

```
(10000, 2)
```

In [61]:

Out[61]: Text(0.5, 1.0, 'Figure : 2 PCA of mnist test dataset')



From above plotting, we can infer following things,

- class 1 could be labelled more definitively and it is easy to train as is can be easily separable by other classes while we keep only two feature components.
- class 1 and 4 are highly overlaping in 2D space.
- class 1 and 3 are highly co-related as well and most of them are overlapping also.

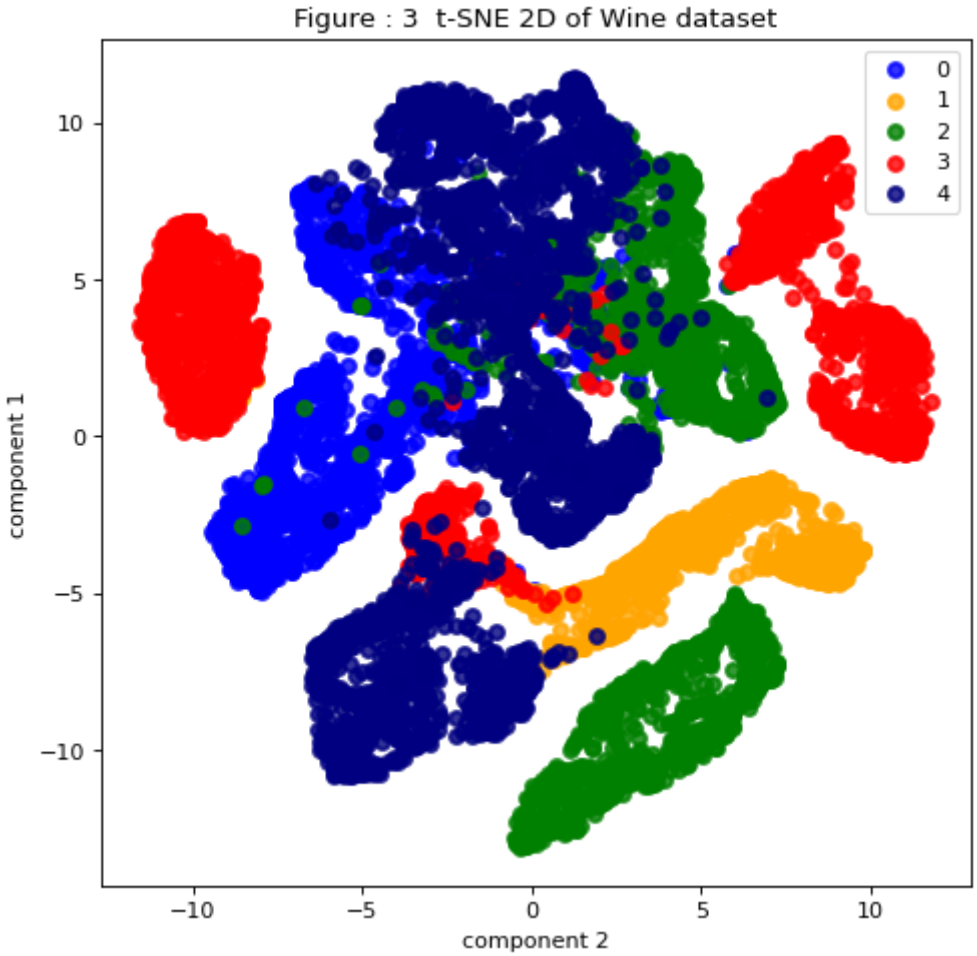
Overall, applying PCA and getting two componet might not help us to give good accuracy as classes(labels) or datapoints are highly overlapping.

t-SNE

In [ ]:
```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=300)
mnist_tsne = tsne.fit_transform(df_test)
```

In [60]:
```
```

Out[60]:
```
Text(0.5, 1.0, 'Figure : 3  t-SNE 2D of Wine dataset')
```



DBSCAN

We will apply DBSCAN clustering algorithm on our training dataset and then try to predict the clsters on testing data.

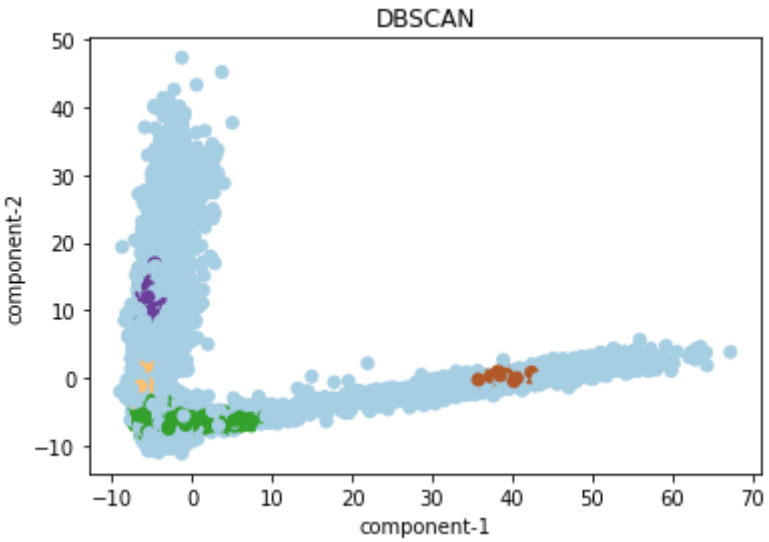
In [75]:
```
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps = 6, min_samples = 80).fit(df_test)
cluster = clustering.labels_
```

I have run different combination of min_sample and eps values on test dataset to find the optimum values so that it will return 5 clusters as we know our data has 5 classes.

In [84]:
```
def dbscan(X_train,X_test,pca_x , eps, min_samples):
    ss = StandardScaler()
    X = ss.fit_transform(X_train)
    db = DBSCAN(eps=eps, min_samples=min_samples)
    db.fit(X)
    y_pred = db.fit_predict(X_test)
    plt.scatter(pca_x[:,0], pca_x[:,1],c=y_pred, cmap='Paired')
    plt.xlabel("component-1")
    plt.ylabel("component-2")
    plt.title("DBSCAN")
```

In [87]:
```
print("         Figure 4         ")
dbscan(df_train,df_test,mnist_pca,6,80)
```

```
         Figure 4
```



Above graph shows that most of the datapoints are in one class only and others are overlapping that class which is not a good classification.

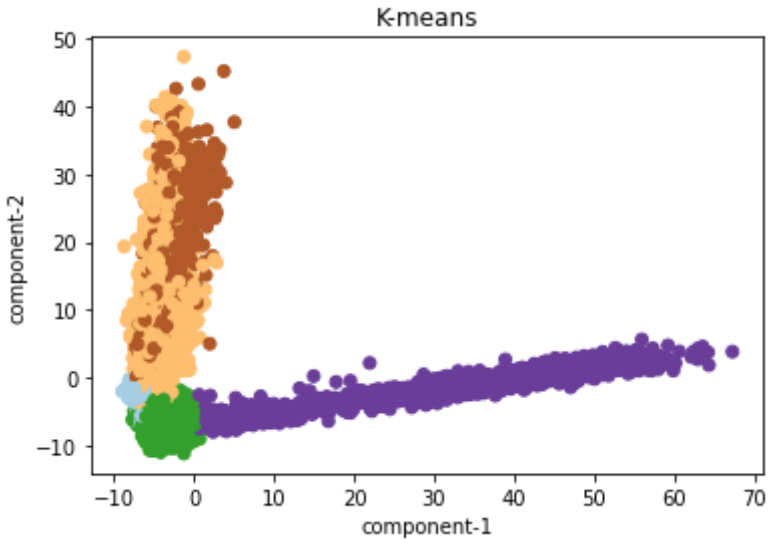
K-means clustering algorithm

This is another clustering algorithm to cluster the datset. Here we will take 5 cluster size and try to train our model on training set and than predict it on testing set.

In [91]:
```
def kmeans(X_train,X_test,pca_x, n_clusters):
    ss = StandardScaler()
    X = ss.fit_transform(X_train)
    km = KMeans(n_clusters=n_clusters)
    km.fit(X)
    y_pred = km.predict(X_test)
    plt.scatter(pca_x[:,0], pca_x[:,1],c=y_pred, cmap='Paired')
    plt.xlabel("component-1")
    plt.ylabel("component-2")
    plt.title("K-means")
```

In [92]:
```
print("           Figure 5              ")
kmeans(df_train,df_test,mnist_pca,5)
```

```
           Figure 5
```



From above graph, we can tell that k-means is performing better than DBSCAN at some extent as it is able to define purple and green clusetes very well while other datapoints are merging into each other.

Final analysis

If we compare figure 3(t-SNE) with figure 1(lablled Images), we can observe following things:

- **Label 1(yellow)** is well seperated from other clusters and the same pattern you can see in **figure 1** (all female sandals) are assigned **label 1**
- Same way label 3(red) is producing two red clusters, which if we will refer to figure 1, we can say that those could be for sneakers and handbags(so these two categories have one label 3)
- label 2(green) is also generating two clusters one of them is close to label 0 and label 4, which expaines the mixied data (sweaters or male top) if you refer figure 1 for this labels.
- while label 0 has three clusters and most of the datapoints are overlaping with label 2 and 4.

So after this analysis we cna say that,

- label 0 could be all the top (female + male)
- label 1 can be female sandals
- label 2 some to (male sweaters + female plus pants)
- label 3 are possibly sneakers and handbags.
- label 4 colud be boots plus clothes.

In [ ]: