In [1]:
```python
from IPython.display import Image
Image(filename='logo.PNG', height=340, width=900)
```

Out[1]:

KMEANS ALGORITHM

STEPS:

1. Step 1: Specify desired number of clusters k.
2. Step 2: Select at Random the Centroids in each of the clusters
3. Step 3: Assign each data point to the closest Cluster
4. Step 4: Now the new mean(Centroid) is the average of the data points in a cluster.
5. Step 5: Reassign the datapoint to the closest centroid. If there is no reassignment - finish. else keep repeating

Optimal Clusters

We can also find the optimal cluster using the Elbow Method. We shall see this below

In [1]:

```
# Importing Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
# Import Dataset
df = pd.read_csv('creditcard.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Cust ID	Gender	Age	Monthly Income in 1000s	CreditScore (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [4]:

```
df.tail()
```

Out[4]:

	Cust ID	Gender	Age	Monthly Income in 1000s	CreditScore (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
Cust ID                    200 non-null int64
Gender                     200 non-null object
Age                        200 non-null int64
Monthly Income in 1000s    200 non-null int64
CreditScore (1-100)        200 non-null int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [6]:
```
df.describe()
```

Out[6]:

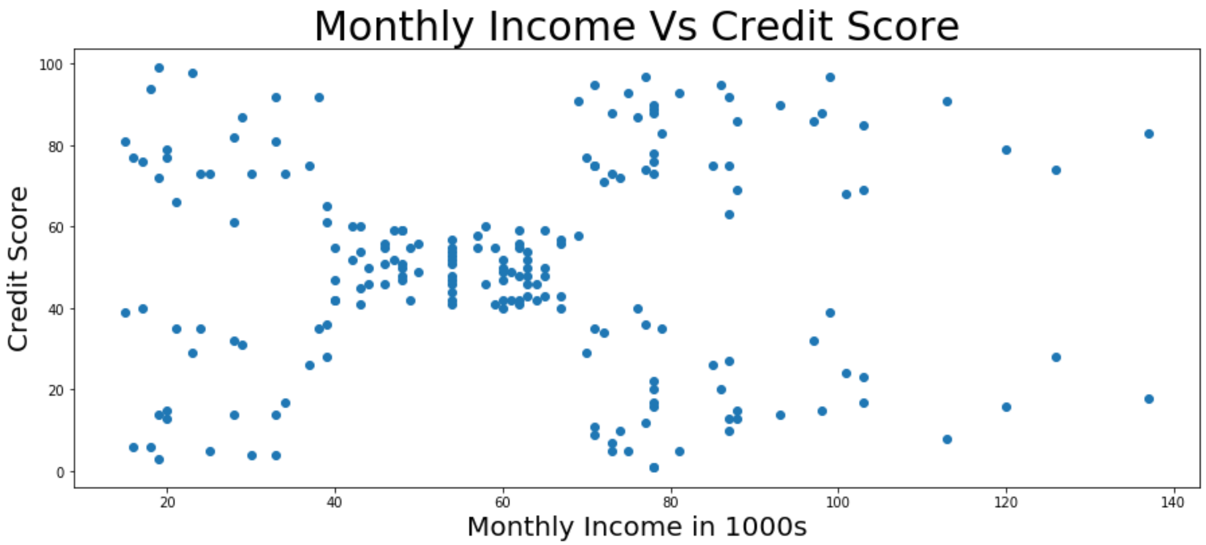
	Cust ID	Age	Monthly Income in 1000s	CreditScore (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

In [4]:
```
X = df.iloc[:,[3,4]].values
```

In [5]:
```
# Initial View of the Data
plt.figure(figsize=(15,6))
plt.scatter(X[:,0], X[:,1])
plt.xlabel('Monthly Income in 1000s', fontsize=20)
plt.ylabel('Credit Score', fontsize=20)
plt.title('Monthly Income Vs Credit Score', fontsize=30)
```

Out[5]:

```
Text(0.5, 1.0, 'Monthly Income Vs Credit Score')
```



ELBOW METHOD - OPTIMAL CLUSTERS

Within Clusters Sum of Squares

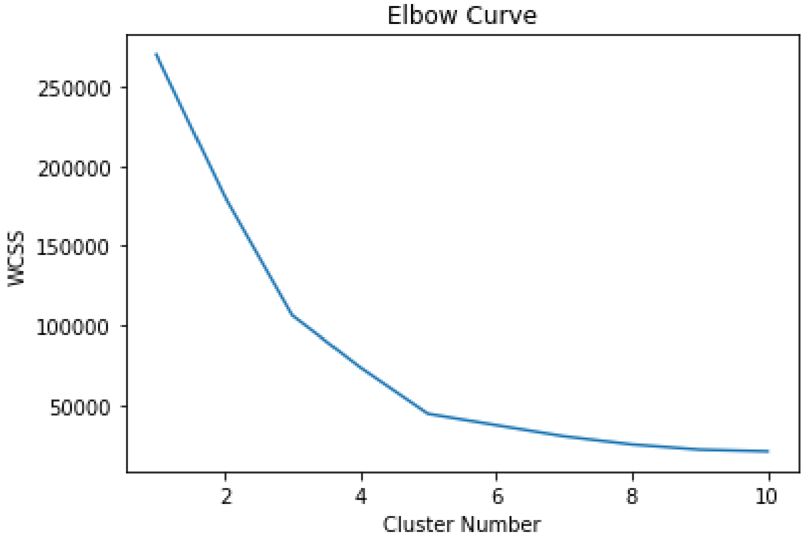
wcss = SUMMATION euclidean distance of (Pi, C1)^2 + SUMMATION euclidean distance of (Pi, C2)^2

In [6]:

```
# Finding the optimum clusters using the ELBOW curve
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmean = KMeans(n_clusters=i, random_state=0)

    kmean.fit(X)
    wcss.append(kmean.inertia_)
```

```
plt.plot(range(1,11), wcss)
plt.title('Elbow Curve')
plt.xlabel('Cluster Number')
plt.ylabel('WCSS')
plt.show()
```



In [7]:

```
# Fitting the model
kmean = KMeans(n_clusters=5, random_state=0)
y_kmean_clustering = kmean.fit_predict(X)
```

In [16]:

```
y_kmean_clustering
```

Out[16]:

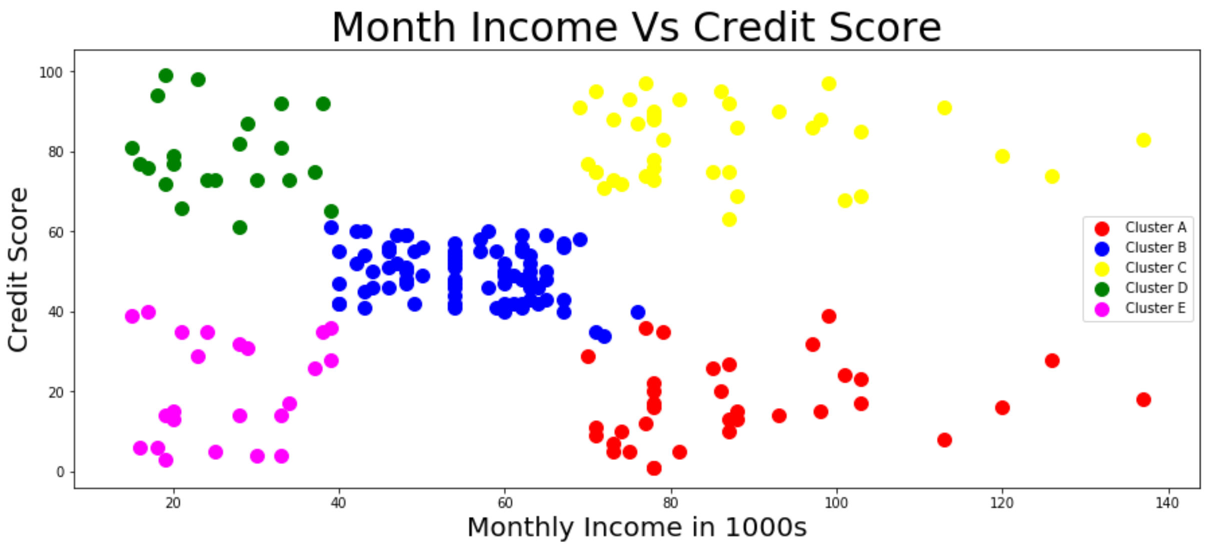
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4,
3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4,
1,
       4, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 2, 1, 2, 0, 2, 0,
2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0,
2,
       0, 2])
```

In [25]:

```
# Visualizing Results
plt.figure(figsize=(15,6))
plt.scatter(X[y_kmean_clustering==0, 0], X[y_kmean_clustering==0, 1], s
=100, c='red', label = 'Cluster A')
plt.scatter(X[y_kmean_clustering==1, 0], X[y_kmean_clustering==1, 1], s
=100, c='blue', label = 'Cluster B')
plt.scatter(X[y_kmean_clustering==2, 0], X[y_kmean_clustering==2, 1], s
=100, c='yellow', label = 'Cluster C')
plt.scatter(X[y_kmean_clustering==3, 0], X[y_kmean_clustering==3, 1], s
=100, c='green', label = 'Cluster D')
plt.scatter(X[y_kmean_clustering==4, 0], X[y_kmean_clustering==4, 1], s
=100, c='magenta', label = 'Cluster E')
#plt.scatter(kmean.cluster_centers_[:,0], kmean.cluster_centers_[:,1],
 s = 200, c = 'black', label = 'Centroid')
plt.legend()
plt.xlabel('Monthly Income in 1000s', fontsize=20)
plt.ylabel('Credit Score', fontsize=20)
plt.title('Month Income Vs Credit Score', fontsize=30)
```

Out[25]:

```
Text(0.5, 1.0, 'Month Income Vs Credit Score')
```


Month Income Vs Credit Score
Credit Score
100
80
60
40
20
0
20
40
60
80
100
120
140
Monthly Income in 1000s
● Cluster A
● Cluster B
● Cluster C
● Cluster D
● Cluster E

In [ ]:

In [ ]:

In [ ]:

In [ ]: