**Name : Mohammad AsifReg**
**No: 20214096Group: 6C**

1.

a) Code:

```
import numpy as np
import pandas as pd


def compute_distance_matrix(data):
    data = pd.get_dummies(data)  # Convert non-numeric data to numeric if needed
    distance_matrix = np.zeros((len(data), len(data)))



    for i in range(len(data)):
        for j in range(len(data)):
            distance_matrix[i][j] = np.sqrt(np.sum((data.iloc[i] - data.iloc[j])**2))



    return distance_matrix

# Example usage with a DataFrame `df`
# distance_matrix = compute_distance_matrix(df)
```

b) Code:

```
import random


def k_means_clustering(k, data):
    centroids = data.sample(n=k)
    partitions = {}



    while True:
        clusters = {i: [] for i in range(k)}
```

```
    # Assigning clusters
    for index, row in data.iterrows():
        distances = [np.sqrt(np.sum((row - centroids.iloc[i])**2)) for i in range(k)]
        cluster_index = distances.index(min(distances))
        clusters[cluster_index].append(index)



    # Updating centroids
    new_centroids_data = []
    for key, value in clusters.items():
        cluster_data = data.loc[value]
        new_centroids_data.append(cluster_data.mean(axis=0))



    new_centroids = pd.DataFrame(new_centroids_data)



    if new_centroids.equals(centroids):
        partitions = clusters
        break



    centroids = new_centroids



return partitions, centroids
```

```
# Example usage with a DataFrame `df`
# k_value is the number of desired clusters.
# partitions, final_centroids  k_means_clustering(k_value, df)
```

c) Code:

```
import matplotlib.pyplot as plt

def visualize_clusters(partitions, data):
    colors=['orange', 'blue', 'green', 'red', 'purple']  # Add more colors if needed


    plt.figure()


    for i, indices in partitions.items():
        cluster_data=data.loc[indices]


        plt.scatter(cluster_data['x'], cluster_data['y'], c=colors[i], label=f'Cluster {i}')


    plt.legend()


plt.show()

# Example usage with
```