Importing the Data

In [1]:
```
import pandas as pd
custdata = pd.read_csv("custdata.csv")
```

In [2]:
```
custdata
```

Out[2]:

	id	PartyName	PartyCode	DistrChl	Location	PinCode
0	1	xx	ECIGGALUR, BANGALORE562106KA219.533	EC	IGGALUR, BANGALORE	562106
1	2	xx	MTBANGALORE560100KA1474.7033	MT	BANGALORE	560100
2	3	xx	ECJIGANI, ANEKAL, BANGALORE560099KA668.7275003...	EC	JIGANI, ANEKAL, BANGALORE	560099
3	4	xx	GTBANGALORE560105KA1469.8638	GT	BANGALORE	560105
4	5	xx	ECBANGALORE562106KA668.727500369458	EC	BANGALORE	562106
5	6	xx	GTANEKAL562106KA3657.75045454545	GT	ANEKAL	562106
6	7	xx	ISBANGALORE560076KA3326.74418181818	IS	BANGALORE	560076
7	8	xx	ISBANGALORE560076KA1021.05744444444	IS	BANGALORE	560076
8	9	xx	MTANEKAL562106KA1495.78935013509	MT	ANEKAL	562106
9	10	xx	GTBENGALURU560102KA2993.31148118806	GT	BENGALURU	560102
10	11	xx	ISBANGALORE560102KA76.17	IS	BANGALORE	560102
11	12	xx	GTBANGALORE560029KA6290.7166	GT	BANGALORE	560029
12	13	xx	ISBANGALORE560095KA105.4875	IS	BANGALORE	560095
13	14	xx	MTBANGALORE560076KA1021.05744444444	MT	BANGALORE	560076
14	15	xx	MTBANGALORE560011KA5958.5932	MT	BANGALORE	560011
15	16	xx	MTBANGALORE560027KA2759.4412	MT	BANGALORE	560027
16	17	xx	CCBANGALORE560007KA1456.55266666667	CC	BANGALORE	560007
17	18	xx	MTBANGALORE URBAN560027KA3099.5304	MT	BANGALORE URBAN	560027
18	19	xx	GTBENGALURU560078KA6346.128	GT	BENGALURU	560078
19	20	xx	MTBANGALORE560083KA13789.7764285714	MT	BANGALORE	560083

In [3]:
```
custdata = custdata.fillna(0)
custdata
```

Out[3]:

	id	PartyName	PartyCode	DistrChl	Location	PinCode
0	1	xx	ECIGGALUR, BANGALORE562106KA219.533	EC	IGGALUR, BANGALORE	562106
1	2	xx	MTBANGALORE560100KA1474.7033	MT	BANGALORE	560100
2	3	xx	ECJIGANI, ANEKAL, BANGALORE560099KA668.7275003...	EC	JIGANI, ANEKAL, BANGALORE	560099
3	4	xx	GTBANGALORE560105KA1469.8638	GT	BANGALORE	560105
4	5	xx	ECBANGALORE562106KA668.727500369458	EC	BANGALORE	562106
5	6	xx	GTANEKAL562106KA3657.75045454545	GT	ANEKAL	562106
6	7	xx	ISBANGALORE560076KA3326.74418181818	IS	BANGALORE	560076
7	8	xx	ISBANGALORE560076KA1021.05744444444	IS	BANGALORE	560076
8	9	xx	MTANEKAL562106KA1495.78935013509	MT	ANEKAL	562106
9	10	xx	GTBENGALURU560102KA2993.31148118806	GT	BENGALURU	560102
10	11	xx	ISBANGALORE560102KA76.17	IS	BANGALORE	560102
11	12	xx	GTBANGALORE560029KA6290.7166	GT	BANGALORE	560029
12	13	xx	ISBANGALORE560095KA105.4875	IS	BANGALORE	560095
13	14	xx	MTBANGALORE560076KA1021.05744444444	MT	BANGALORE	560076
14	15	xx	MTBANGALORE560011KA5958.5932	MT	BANGALORE	560011
15	16	xx	MTBANGALORE560027KA2759.4412	MT	BANGALORE	560027
16	17	xx	CCBANGALORE560007KA1456.55266666667	CC	BANGALORE	560007
17	18	xx	MTBANGALORE URBAN560027KA3099.5304	MT	BANGALORE URBAN	560027
18	19	xx	GTBENGALURU560078KA6346.128	GT	BENGALURU	560078
19	20	xx	MTBANGALORE560083KA13789.7764285714	MT	BANGALORE	560083

In [4]:
```
len(custdata)
```

Out[4]:
```
20
```

In [5]:
```
warehouse = [[0, 12.819304, 77.688005]]
li = warehouse.copy()

rows = pd.DataFrame(custdata, columns=["id", "Latitude", "Longitude"])
li.extend(rows.values.tolist())

list_length = len(li)
```

In [6]:
```
li
```

Out[6]:
```
[[0, 12.819304, 77.688005],
 [1.0, 12.78315, 77.70227],
 [2.0, 12.8498, 77.6545],
 [3.0, 12.77918, 77.64354],
 [4.0, 12.77745, 77.64208],
 [5.0, 12.72299, 77.67642],
 [6.0, 12.72547, 77.67747],
 [7.0, 12.88451, 77.60355],
 [8.0, 12.88456, 77.60344],
 [9.0, 12.72192, 77.6763],
 [10.0, 12.9126, 77.64485],
 [11.0, 12.91227, 77.64432],
 [12.0, 12.93341, 77.60426],
 [13.0, 12.93769, 77.64042],
 [14.0, 12.88483, 77.60423],
 [15.0, 12.93366, 77.58942],
 [16.0, 12.95609, 77.59214],
 [17.0, 12.95773, 77.63047],
 [18.0, 12.95621, 77.5922],
 [19.0, 12.89852, 77.57647],
 [20.0, 12.82672, 77.55494]]
```

In [7]:

```
from geopy.distance import geodesic
import numpy as np

num_points = len(li)
dist_matrix = np.zeros((num_points, num_points))

for i in range(num_points):
    lat1, lon1 = li[i][1], li[i][2]
    for j in range(num_points):
        lat2, lon2 = li[j][1], li[j][2]

        coord1 = (lat1, lon1)
        coord2 = (lat2, lon2)
        print(coord1)
        print(coord2)
        distance = geodesic(coord1, coord2).kilometers
        print(distance)
        distance = round(distance,2)
        dist_matrix[i, j] = round(distance, 2)

dist =dist_matrix[0].tolist()
print(dist)

#print(dist_matrix)

# IT PRINTS THE DISTANCE FROM WAREHOUSE TO OTHER LOCATIONS ONLY
```

```
(12.819304, 77.688005)
(12.819304, 77.688005)
0.0
(12.819304, 77.688005)
(12.78315, 77.70227)
4.289060603629483
(12.819304, 77.688005)
(12.8498, 77.6545)
4.960966682381569
(12.819304, 77.688005)
(12.77918, 77.64354)
6.558162984242644
(12.819304, 77.688005)
(12.77745, 77.64208)
6.804484375750543
(12.819304, 77.688005)
(12.72299, 77.67642)
10.729077797573837
(12.819304, 77.688005)
```

In [8]:
```
import pandas as pd

# Assuming you have an existing DataFrame called 'df'

# Define the values for the new row
new_row = ['0', 'xx', 'Warehouse-Bommasandra','xx','Bomassandra','560099','KA'

# Add the new row to the DataFrame
#custdata.loc[len(custdata)] = new_row

new_df = pd.DataFrame([new_row], columns=custdata.columns)

## Concatenate the new DataFrame with the existing DataFrame
custdata = pd.concat([new_df, custdata]).reset_index(drop=True)
```

In [9]:
```
print(new_df)
```

```
  id PartyName              PartyCode DistrChl     Location PinCode State  \
0  0        xx  Warehouse-Bommasandra       xx  Bomassandra  560099    KA

   AvgMonthlyVolume  AvgDailyVolume  CustomerPreferedDay   Latitude  Longitud
e
0                 0               0                    0  12.819304  77.68800
5
```

In [10]:
```
custdata
```

Out[10]:

	id	PartyName	PartyCode	DistrChl	Location	PinCode
0	0	xx	Warehouse-Bommasandra	xx	Bomassandra	560099
1	1	xx	ECIGGALUR, BANGALORE562106KA219.533	EC	IGGALUR, BANGALORE	562106
2	2	xx	MTBANGALORE560100KA1474.7033	MT	BANGALORE	560100
3	3	xx	ECJIGANI, ANEKAL, BANGALORE560099KA668.7275003...	EC	JIGANI, ANEKAL, BANGALORE	560099
4	4	xx	GTBANGALORE560105KA1469.8638	GT	BANGALORE	560105
5	5	xx	ECBANGALORE562106KA668.727500369458	EC	BANGALORE	562106
6	6	xx	GTANEKAL562106KA3657.75045454545	GT	ANEKAL	562106
7	7	xx	ISBANGALORE560076KA3326.74418181818	IS	BANGALORE	560076
8	8	xx	ISBANGALORE560076KA1021.05744444444	IS	BANGALORE	560076
9	9	xx	MTANEKAL562106KA1495.78935013509	MT	ANEKAL	562106
10	10	xx	GTBENGALURU560102KA2993.31148118806	GT	BENGALURU	560102
11	11	xx	ISBANGALORE560102KA76.17	IS	BANGALORE	560102
12	12	xx	GTBANGALORE560029KA6290.7166	GT	BANGALORE	560029
13	13	xx	ISBANGALORE560095KA105.4875	IS	BANGALORE	560095
14	14	xx	MTBANGALORE560076KA1021.05744444444	MT	BANGALORE	560076
15	15	xx	MTBANGALORE560011KA5958.5932	MT	BANGALORE	560011
16	16	xx	MTBANGALORE560027KA2759.4412	MT	BANGALORE	560027
17	17	xx	CCBANGALORE560007KA1456.55266666667	CC	BANGALORE	560007
18	18	xx	MTBANGALORE URBAN560027KA3099.5304	MT	BANGALORE URBAN	560027
19	19	xx	GTBENGALURU560078KA6346.128	GT	BENGALURU	560078
20	20	xx	MTBANGALORE560083KA13789.7764285714	MT	BANGALORE	560083

In [13]:
```
len(dist)
```

Out[13]: 21

In [14]:
```
len(custdata)
```

Out[14]: 21

In [15]:
```
custdata["Distance"] = dist
```

In [16]:
```
# Create a new DataFrame containing only the rows where distance is less than
custdata_less_than_100= custdata[custdata['Distance'] < 100]


custdata_less_than_100
```

Out[16]:

	id	PartyName	PartyCode	DistrChl	Location	PinCode
0	0	xx	Warehouse-Bommasandra	xx	Bomassandra	560099
1	1	xx	ECIGGALUR, BANGALORE562106KA219.533	EC	IGGALUR, BANGALORE	562106
2	2	xx	MTBANGALORE560100KA1474.7033	MT	BANGALORE	560100
3	3	xx	ECJIGANI, ANEKAL, BANGALORE560099KA668.7275003...	EC	JIGANI, ANEKAL, BANGALORE	560099
4	4	xx	GTBANGALORE560105KA1469.8638	GT	BANGALORE	560105
5	5	xx	ECBANGALORE562106KA668.727500369458	EC	BANGALORE	562106
6	6	xx	GTANEKAL562106KA3657.75045454545	GT	ANEKAL	562106
7	7	xx	ISBANGALORE560076KA3326.74418181818	IS	BANGALORE	560076
8	8	xx	ISBANGALORE560076KA1021.05744444444	IS	BANGALORE	560076
9	9	xx	MTANEKAL562106KA1495.78935013509	MT	ANEKAL	562106
10	10	xx	GTBENGALURU560102KA2993.31148118806	GT	BENGALURU	560102
11	11	xx	ISBANGALORE560102KA76.17	IS	BANGALORE	560102
12	12	xx	GTBANGALORE560029KA6290.7166	GT	BANGALORE	560029
13	13	xx	ISBANGALORE560095KA105.4875	IS	BANGALORE	560095
14	14	xx	MTBANGALORE560076KA1021.05744444444	MT	BANGALORE	560076
15	15	xx	MTBANGALORE560011KA5958.5932	MT	BANGALORE	560011
16	16	xx	MTBANGALORE560027KA2759.4412	MT	BANGALORE	560027
17	17	xx	CCBANGALORE560007KA1456.55266666667	CC	BANGALORE	560007
18	18	xx	MTBANGALORE URBAN560027KA3099.5304	MT	BANGALORE URBAN	560027
19	19	xx	GTBENGALURU560078KA6346.128	GT	BENGALURU	560078
20	20	xx	MTBANGALORE560083KA13789.7764285714	MT	BANGALORE	560083

In [17]:
```
row = custdata_less_than_100[["id", "Latitude", "Longitude"]]
xi= row.values.tolist()
len(xi)
```

Out[17]:
```
21
```

In [18]:
```
xi
```

Out[18]:
```
[['0', 12.819304, 77.688005],
 [1, 12.78315, 77.70227],
 [2, 12.8498, 77.6545],
 [3, 12.77918, 77.64354],
 [4, 12.77745, 77.64208],
 [5, 12.72299, 77.67642],
 [6, 12.72547, 77.67747],
 [7, 12.88451, 77.60355],
 [8, 12.88456, 77.60344],
 [9, 12.72192, 77.6763],
 [10, 12.9126, 77.64485],
 [11, 12.91227, 77.64432],
 [12, 12.93341, 77.60426],
 [13, 12.93769, 77.64042],
 [14, 12.88483, 77.60423],
 [15, 12.93366, 77.58942],
 [16, 12.95609, 77.59214],
 [17, 12.95773, 77.63047],
 [18, 12.95621, 77.5922],
 [19, 12.89852, 77.57647],
 [20, 12.82672, 77.55494]]
```

In [19]:

```
from geopy.distance import geodesic
import numpy as np

num_points = len(xi)
distance_matrix = np.zeros((num_points, num_points))

for i in range(num_points):
    lat1, lon1 = xi[i][1], xi[i][2]
    for j in range(num_points):
        lat2, lon2 = xi[j][1], xi[j][2]

        coord1 = (lat1, lon1)
        coord2 = (lat2, lon2)
        #print(coord1)
        #print(coord2)
        distances = geodesic(coord1, coord2).kilometers
        distance_matrix[i, j] = round(distances,1)

print(distance_matrix)
```

```
[[ 0.   4.3  5.   6.6  6.8 10.7 10.4 11.7 11.7 10.8 11.3 11.3 15.6 14.1
  11.6 16.6 18.4 16.5 18.4 14.9 14.5]
 [ 4.3  0.   9.   6.4  6.6  7.2  6.9 15.5 15.5  7.3 15.6 15.6 19.7 18.4
  15.5 20.7 22.6 20.8 22.6 18.7 16.7]
 [ 5.   9.   0.   7.9  8.1 14.2 14.   6.7  6.7 14.3  7.   7.  10.7  9.8
   6.7 11.7 13.6 12.2 13.6 10.  11.1]
 [ 6.6  6.4  7.9  0.   0.2  7.2  7.  12.4 12.4  7.3 14.8 14.7 17.6 17.5
  12.4 18.1 20.4 19.8 20.4 15.1 11. ]
 [ 6.8  6.6  8.1  0.2  0.   7.1  6.9 12.6 12.6  7.2 15.  14.9 17.7 17.7
  12.6 18.2 20.5 20.  20.5 15.2 10.9]
 [10.7  7.2 14.2  7.2  7.1  0.   0.3 19.5 19.6  0.1 21.3 21.2 24.6 24.1
  19.5 25.1 27.4 26.4 27.4 22.2 17.5]
 [10.4  6.9 14.   7.   6.9  0.3  0.  19.3 19.3  0.4 21.  21.  24.3 23.8
  19.3 24.9 27.1 26.2 27.2 22.1 17.4]
 [11.7 15.5  6.7 12.4 12.6 19.5 19.3  0.   0.  19.6  5.5  5.4  5.4  7.1
   0.1  5.6  8.   8.6  8.   3.3  8.3]
 [11.7 15.5  6.7 12.4 12.6 19.6 19.3  0.   0.  19.7  5.5  5.4  5.4  7.1
   0.1  5.6  8.   8.6  8.   3.3  8.3]
 [10.8  7.3 14.3  7.3  7.2  0.1  0.4 19.6 19.7  0.  21.4 21.3 24.7 24.2
  19.6 25.3 27.5 26.6 27.5 22.3 17.6]
 [11.3 15.6  7.  14.8 15.  21.3 21.   5.5  5.5 21.4  0.   0.1  5.   2.8
   5.4  6.5  7.5  5.2  7.5  7.6 13.6]
 [11.3 15.6  7.  14.7 14.9 21.2 21.   5.4  5.4 21.3  0.1  0.   4.9  2.8
   5.3  6.4  7.5  5.2  7.5  7.5 13.6]
 [15.6 19.7 10.7 17.6 17.7 24.6 24.3  5.4  5.4 24.7  5.   4.9  0.   4.
   5.4  1.6  2.8  3.9  2.8  4.9 13. ]
 [14.1 18.4  9.8 17.5 17.7 24.1 23.8  7.1  7.1 24.2  2.8  2.8  4.   0.
   7.   5.6  5.6  2.5  5.6  8.2 15.4]
 [11.6 15.5  6.7 12.4 12.6 19.5 19.3  0.1  0.1 19.6  5.4  5.3  5.4  7.
   0.   5.6  8.   8.6  8.   3.4  8.4]
 [16.6 20.7 11.7 18.1 18.2 25.1 24.9  5.6  5.6 25.3  6.5  6.4  1.6  5.6
   5.6  0.   2.5  5.2  2.5  4.1 12.4]
 [18.4 22.6 13.6 20.4 20.5 27.4 27.1  8.   8.  27.5  7.5  7.5  2.8  5.6
   8.   2.5  0.   4.2  0.   6.6 14.9]
 [16.5 20.8 12.2 19.8 20.  26.4 26.2  8.6  8.6 26.6  5.2  5.2  3.9  2.5
   8.6  5.2  4.2  0.   4.2  8.8 16.7]
 [18.4 22.6 13.6 20.4 20.5 27.4 27.2  8.   8.  27.5  7.5  7.5  2.8  5.6
   8.   2.5  0.   4.2  0.   6.6 14.9]
 [14.9 18.7 10.  15.1 15.2 22.2 22.1  3.3  3.3 22.3  7.6  7.5  4.9  8.2
   3.4  4.1  6.6  8.8  6.6  0.   8.3]
 [14.5 16.7 11.1 11.  10.9 17.5 17.4  8.3  8.3 17.6 13.6 13.6 13.  15.4
   8.4 12.4 14.9 16.7 14.9  8.3  0. ]]
```

In [20]:
```
distance_matrix[2]
```

Out[20]:
```
array([ 5. ,  9. ,  0. ,  7.9,  8.1, 14.2, 14. ,  6.7,  6.7, 14.3,  7. ,
        7. , 10.7,  9.8,  6.7, 11.7, 13.6, 12.2, 13.6, 10. , 11.1])
```

In [21]:
```
cust_demand = custdata['AvgDailyVolume']
#cust_demand = cust_demand.round(2)
cust_demand= list(round(cust_demand,2))
# Print the column value
print(cust_demand)
```

```
[0.0, 25.72, 115.13, 56.53, 8.44, 25.72, 140.68, 56.72, 57.53, 4.06, 52.02, 2
44.08, 175.2, 483.98, 241.95, 115.13, 2.93, 127.95, 39.27, 229.18, 39.27]
```

In [ ]:

In [24]:

```
"""Capacited Vehicles Routing Problem (CVRP)."""

from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp


def create_data_model():
    """Stores the data for the problem."""
    data = {}
    data['distance_matrix'] = distance_matrix
    data['demands'] = cust_demand
    data['vehicle_capacities'] = [700, 700, 700, 700]
    data['num_vehicles'] = 4
    data['depot'] = 0
    return data

def print_solution(data, manager, routing, solution):
    """Prints solution on console."""
    print(f'Objective: {solution.ObjectiveValue()}')
    total_distance = 0
    total_load = 0
    for vehicle_id in range(data['num_vehicles']):
        index = routing.Start(vehicle_id)
        plan_output = 'Route for vehicle {}:\n'.format(vehicle_id)
        route_distance = 0
        route_load = 0
        while not routing.IsEnd(index):
            node_index = manager.IndexToNode(index)
            route_load += data['demands'][node_index]
            plan_output += ' {0} Load({1}) -> '.format(node_index, route_load)
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            route_distance += routing.GetArcCostForVehicle(
                previous_index, index, vehicle_id)
        plan_output += ' {0} Load({1})\n'.format(manager.IndexToNode(index),
                                                 route_load)
        plan_output += 'Distance of the route: {}Km\n'.format(route_distance)
        plan_output += 'Load of the route: {}\n'.format(route_load)
        print(plan_output)
        total_distance += route_distance
        total_load += route_load
    print('Total distance of all routes: {}Km'.format(total_distance))
    print('Total load of all routes: {}'.format(total_load))


def main():
    """Solve the CVRP problem."""
    # Instantiate the data problem.
    data = create_data_model()

    # Create the routing index manager.
    manager = pywrapcp.RoutingIndexManager(len(data['distance_matrix']),
                                           data['num_vehicles'], data['depot'])

    # Create Routing Model.
    routing = pywrapcp.RoutingModel(manager)
```

```
    # Create and register a transit callback.
    def distance_callback(from_index, to_index):
        """Returns the distance between the two nodes."""
        # Convert from routing variable Index to distance matrix NodeIndex.
        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        return data['distance_matrix'][from_node][to_node]

    transit_callback_index = routing.RegisterTransitCallback(distance_callback

    # Define cost of each arc.
    routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)


    # Add Capacity constraint.
    def demand_callback(from_index):
        """Returns the demand of the node."""
        # Convert from routing variable Index to demands NodeIndex.
        from_node = manager.IndexToNode(from_index)
        return data['demands'][from_node]

    demand_callback_index = routing.RegisterUnaryTransitCallback(
        demand_callback)
    routing.AddDimensionWithVehicleCapacity(
        demand_callback_index,
        0,  # null capacity slack
        data['vehicle_capacities'],  # vehicle maximum capacities
        True,  # start cumul to zero
        'Capacity')

    # Setting first solution heuristic.
    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC)
    search_parameters.local_search_metaheuristic = (
        routing_enums_pb2.LocalSearchMetaheuristic.GUIDED_LOCAL_SEARCH)
    search_parameters.time_limit.FromSeconds(1)

    # Solve the problem.
    solution = routing.SolveWithParameters(search_parameters)

    # Print solution on console.
    if solution:
        print_solution(data, manager, routing, solution)


if __name__ == '__main__':
    main()
```

```
Objective: 124
Route for vehicle 0:
 0 Load(0.0) ->  10 Load(52.02) ->  13 Load(536.0) ->  17 Load(663.95) ->  0
Load(663.95)
Distance of the route: 31Km
Load of the route: 663.95

Route for vehicle 1:
 0 Load(0.0) ->  2 Load(115.13) ->  15 Load(230.26) ->  18 Load(269.53) ->  1
6 Load(272.46) ->  12 Load(447.65999999999997) ->  11 Load(691.74) ->  0 Load
(691.74)
Distance of the route: 35Km
Load of the route: 691.74

Route for vehicle 2:
 0 Load(0.0) ->  7 Load(56.72) ->  14 Load(298.66999999999996) ->  8 Load(35
6.19999999999993) ->  19 Load(585.3799999999999) ->  20 Load(624.649999999999
9) ->  4 Load(633.0899999999999) ->  3 Load(689.6199999999999) ->  0 Load(68
9.6199999999999)
Distance of the route: 38Km
Load of the route: 689.6199999999999

Route for vehicle 3:
 0 Load(0.0) ->  1 Load(25.72) ->  6 Load(166.4) ->  9 Load(170.46) ->  5 Loa
d(196.18) ->  0 Load(196.18)
Distance of the route: 20Km
Load of the route: 196.18

Total distance of all routes: 124Km
Total load of all routes: 2241.49
```

In [ ]: