```python
from dimod import ConstrainedQuadraticModel, CQM, SampleSet
from dimod import Binary, quicksum
from dwave.system import LeapHybridCQMSampler
from dwave.cloud.client import Client
from dwave.cloud import config
import numpy as np
import pandas as pd
import ast
import itertools
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import random
import math
from scipy.spatial import distance
import time
import re
import matplotlib.pyplot as plt
import random



# Generate random number of cities and vehicles
nbOfPointToCluster = random.randint(10, 100)  # Random number between 5 and 10
nbOfCluster = random.randint(3, 8)  # Random number between 3 and 6

# Generate random demands for each city, with the depot having a demand of 0
vectorOfVolume = [0] + [random.randint(100, 300) for _ in range(nbOfPointToCluster - 1)]

# Generate random capacities for each vehicle
vectorOfCapacity = [random.randint(10,30) for _ in range(nbOfCluster)]

# Define the binary variables
x = {(i, d): Binary('x{}_{}'.format(i, d)) for i in range(nbOfPointToCluster) for d in
range(nbOfCluster)}

# Generate random costs between each pair of cities
matrixOfCost = [[random.randint(10, 30) if i != j else 0 for j in range(nbOfPointToCluster)] for i in
range(nbOfPointToCluster)]

objective = quicksum(matrixOfCost[i][j] * x[(i,d)] * x[(j,d)]
 for i in range(nbOfPointToCluster)
 for j in range(i+1, nbOfPointToCluster)
 for d in range(nbOfCluster) )
```

```
cqm = ConstrainedQuadraticModel()
cqm.set_objective(objective)

# Define the constraints
for d in range(nbOfCluster):
   cqm.add_constraint(x[(0,d)] == 1)
for d in range(nbOfCluster):
   cqm.add_constraint(quicksum(vectorOfVolume[i] * x[(i,d)] for i in range(nbOfPointToCluster))
<= vectorOfCapacity[d])
for i in range(1,nbOfPointToCluster):
    cqm.add_constraint(quicksum(x[(i,d)] for d in range(nbOfCluster)) == 1)


# Generate random coordinates for each city
city_coordinates = [(random.randint(0, 100), random.randint(0, 100)) for _ in
range(nbOfPointToCluster)]

#We get our solution.
cqm_sampler = LeapHybridCQMSampler(token='insertokenhere')
sampleset = cqm_sampler.sample_cqm(cqm,label='CVRP')
#We transform it in a panda dataframe
 dataFrame = sampleset.to_pandas_dataframe(sample_column=True)
dataFrame = dataFrame[['sample','energy','is_feasible']]
dataFrame = dataFrame.sort_values(by = 'energy')
dataFrame.to_csv('C:/Users/Gebruiker/Desktop/learning/cvrp/clustering4.csv')
print(dataFrame)
```