1. Estimate how long this assignment will take.
It will take 8 hours to finish homework

2. You will need to remove one of the attributes in the CSV file. Which one should you ***certainly always*** remove?
Attribute ID should be removed from given attributes as ID does not affect the behavior of records.

3. You can keep all the other attributes, or remove more. Which attributes did you finally use? (½)
All the attributes except ID were used

4. Submit code that shows the clustering process. (4 points, including code readability)
Please find attached

5. At each stage of clustering (from stage 1 to 99), what was the size of smaller cluster that was merged in? What does this indicate about the true number of clusters? (½)
The size of smaller cluster was usually 1, ranging to 5, which means the true number of clusters is close to 100.

6. When you have clustered to three clusters, report the guest id's in each of these three clusters. (1)
{1: [1, 61, 70, 91, 99, 2, 3, 38, 87, 19, 8, 68, 80, 25, 34, 75, 85, 74, 39, 35, 53, 42, 66, 90, 83, 11, 37, 49, 92, 36, 77, 51, 65, 86, 15, 24, 64, 88, 40, 96, 89, 16, 58, 5, 62, 81, 82, 6, 46, 55], 4: [4, 14, 44, 21, 54, 29, 33, 12, 73, 43, 23, 84, 60, 76, 71, 22, 45], 7: [7, 10, 100, 50, 63, 56, 28, 95, 32, 17, 52, 20, 94, 79, 18, 57, 59, 41, 47, 97, 30, 48, 13, 78, 31, 98, 69, 72, 26, 27, 93, 9, 67]}

7. What typifies the third cluster? What nick-name should we give these customers? (be polite) (1)
Following are centroid of each of three clusters and its names:
family purchases: 1 [8.46, 4.960000000000001, 7.72, 9.22, 3.88, 6.84, 4.040000000000001, 6.2, 5.28, 2.4000000000000004, 2.98, 4.82] 50

party animals: 4 [6.0, 5.470588235294118, 5.176470588235294, 8.470588235294118, 7.352941176470588, 2.11764705882353, 5.470588235294118, 4.764705882352942, 3.7058823529411766, 6.411764705882353, 8.058823529411764, 1.3529411764705883] 17

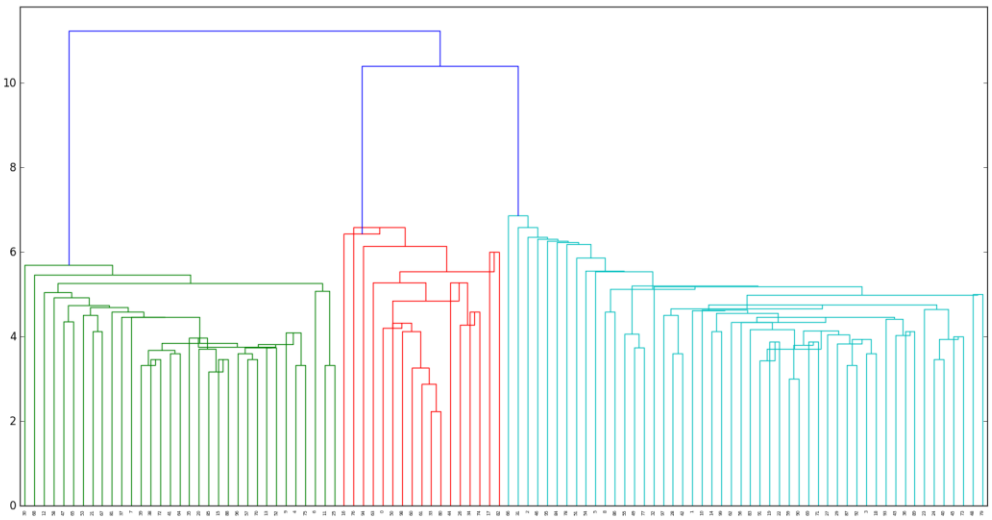
vegetarians: 7 [1.4848484848484849, 4.96969696969697, 9.545454545454545, 6.03030303030303, 6.96969696969697, 8.090909090909092, 0.48484848484848486, 1.7878787878787878, 7.545454545454546, 1.8484848484848484, 1.0909090909090908, 6.848484848484849] 33
Third group can be identified as vegetarians since they hardly have meat or eggs. They hardly have milk but they do have Yogurt so might not be vegan.

8. If we switched from "central link" to a "single link" merge step, what would you need to add to the algorithm when computing the distance between two clusters? (1)
In single linkage clustering distances between all data points are calculated with O(n2) and clusters are formed by merging two clusters whose one-one member are closest to each other. So at each iteration minimum distance is chosen such that points belong to different cluster and the representative clusters are merged. Hence as the clusters are formed, distances between intra-cluster points can be deleted so that minimum distance chosen at next iteration belongs to points from different cluster which will be merged.

9. How long did this assignment take? (½)
10 Hrs

10. Write a short answer question for the next midterm exam. If your question is used, you get the points on the exam. Part of the reason I ask this is to be sure you think about the questions that might be on the
next exam. (½)
Give difference between complete linkage and single linkage clustering

11. Bonus: Generate a dendrogram of the clusters as they are being merged. (1)
Show the code that demonstrates your understanding of this.
For generating the dendrogram it is necessary to preserve information about what 2 clusters were merged to form a new cluster at each iteration of n-1 iterations. New cluster created should be given a new ID in order to trace dendrogram.



I modified the merge method to create new Cluster IDs for merged points as follows need to use self.distances to create dendrogram: Please find attached the dendrogram generated

```
def merge(self, mergePoint):
    number_of_points1 = len(self.cluster_record_ID_map[mergePoint.ID])
    number_of_points2 = len(self.cluster_record_ID_map[mergePoint.record_ID])
    self.distances.append([mergePoint.ID, mergePoint.record_ID, mergePoint.distance, number_of_points1
+number_of_points2 ])
    sum_record1 = [x*number_of_points1 for x in self.center_of_masses[mergePoint.ID]]
    sum_record2 = [x*number_of_points2 for x in self.center_of_masses[mergePoint.record_ID]]
    new_center = self.calculate_center_of_mass(sum_record1, sum_record2)
    new_center = [attribute_sum/(number_of_points1+number_of_points2) for attribute_sum in new_center]
    self.center_of_masses[self.total] = new_center
    self.cluster_record_ID_map[self.total] = [self.total]
    self.clusters_assigned[self.total] = self.total
    for record in self.cluster_record_ID_map[mergePoint.record_ID]:
        self.clusters_assigned[record] = self.total
        self.cluster_record_ID_map[self.total].append(record)
    del [self.cluster_record_ID_map[mergePoint.record_ID]]
    del [self.cluster_record_ID_map[mergePoint.ID]]
    del [self.center_of_masses[mergePoint.record_ID]]
    del [self.center_of_masses[mergePoint.ID]]

    self.count -= 1
    self.total+=1
    pass
```