Assignment: Clustering — Part A

Q1. Hierarchical Clustering (Average and MIN)

Given points

Point	X	Y
P1	0.40	0.50
P2	0.20	0.30
P3	0.10	0.08
P4	0.21	0.12
P5	0.60	0.16
P6	0.33	0.28
P7	0.11	0.15

(A) Euclidean distance matrix (rounded to 6 decimal places)

	P1	P2	P3	P4	P5	P6	P7
P1	0.000000	0.282843	0.516140	0.424853	0.394462	0.230868	0.454533
P2	0.282843	0.000000	0.241661	0.180278	0.423792	0.131529	0.174929
P3	0.516140	0.241661	0.000000	0.117047	0.506360	0.304795	0.070711
P4	0.424853	0.180278	0.117047	0.000000	0.392046	0.200000	0.104403
P5	0.394462	0.423792	0.506360	0.392046	0.000000	0.295466	0.490102

P6	0.230868	0.131529	0.304795	0.200000	0.295466	0.000000	0.255539
P7	0.454533	0.174929	0.070711	0.104403	0.490102	0.255539	0.000000

(B) Clustering — MIN (single linkage)

Single-linkage merge sequence (merge order, cluster composition after merge, and linkage distance):

1. Merge **P3 & P7**. Distance = **0.070711**. Cluster: (P3, P7).
2. Merge **(P3,P7) & P4**. Distance = **0.104403**. Cluster: (P3, P7, P4).
3. Merge **P2 & P6**. Distance = **0.131529**. Cluster: (P2, P6).
4. Merge **(P2,P6) & (P3,P4,P7)**. Distance = **0.174929** (minimum connecting distance between the two clusters). Cluster: (P2, P6, P3, P4, P7).
5. Merge **P1** with the existing cluster. Distance = **0.230868**. Cluster: (P1, P2, P3, P4, P6, P7).
6. Merge **P5** last. Distance = **0.295466** (final merge). Final cluster: (P1, P2, P3, P4, P5, P6, P7).

Interpretation (single linkage): The algorithm first links closest pairs (P3–P7), then agglomerates nearby points into a chain; P5 is the most distant and merges last.

(C) Clustering — AVERAGE (average linkage)

Average-linkage merge sequence (merge order, cluster composition after merge, and linkage distance):

1. Merge **P3 & P7**. Distance = **0.070711**. Cluster: (P3, P7).
2. Merge **(P3,P7) & P4**. Distance = **0.110725**. Cluster: (P3, P7, P4).
3. Merge **P2 & P6**. Distance = **0.131529**. Cluster: (P2, P6).

4. Merge **(P2,P6) & (P3,P4,P7)**. Distance = **0.226200**. Cluster: (P2, P6, P3, P4, P7).

5. Merge **P1** with the existing cluster. Distance = **0.381847**. Cluster: (P1, P2, P3, P4, P6, P7).

6. Merge **P5** last. Distance = **0.417038** (final merge). Final cluster: (P1, P2, P3, P4, P5, P6, P7).

Interpretation (average linkage): Average linkage delays joining some groups compared to single-linkage — P5 still remains the most separated point and merges last.

(D) Dendrograms

- The dendrograms should reflect the merge orders and distances given above.

- You can reproduce the dendrograms in any plotting tool (for example: `scipy.cluster.hierarchy.dendrogram` in Python). The merge heights correspond to the linkage distances listed in each merge step.

Q2. K = 3 (Given centroids) — Distance table and assignments

Given data points

Points: (2,1), (3,1), (3,3), (4,1), (5,1), (6,7), (1,3), (2,5)

Given initial centroids:

- Centroid 1 = (2, 1)

- Centroid 2 = (4, 1)

- Centroid 3 = (5, 1)

Compute Euclidean distance from each point to each centroid (rounded to 6 decimals).

Point (index)	To C1 (2,1)	To C2 (4,1)	To C3 (5,1)	Assigned centroid (nearest)

(2,1) — p1	0.000000	2.000000	3.000000	C1
(3,1) — p2	1.000000	1.000000	2.000000	C1 (tie-breaker — nearest listed first)
(3,3) — p3	2.236068	2.236068	2.828427	C1 (tie-breaker: C1 chosen)
(4,1) — p4	2.000000	0.000000	1.000000	C2
(5,1) — p5	3.000000	1.000000	0.000000	C3
(6,7) — p6	7.211103	6.324555	6.082763	C3
(1,3) — p7	2.236068	3.605551	4.472136	C1
(2,5) — p8	4.000000	4.472136	5.000000	C1

Final assignments (after this distance computation)

- **Cluster 1 (C1 = (2,1))**: (2,1), (3,1), (3,3), (1,3), (2,5)
- **Cluster 2 (C2 = (4,1))**: (4,1)
- **Cluster 3 (C3 = (5,1))**: (5,1), (6,7)

  Note: Because some points are tied in distance (e.g., (3,1) and (3,3) to C1/C2), I used the standard tie-breaking by selecting the earliest centroid listed (C1 over C2) so assignments are deterministic.

Assignment: Clustering — Part B (Short Answers)

Q1.

(a) *Agglomerative Hierarchical Clustering*:
It is a bottom-up clustering approach where each data point starts as its own cluster. Pairs of clusters are successively merged based on a similarity (or distance) measure until only one large cluster remains or a desired number of clusters is formed.

(b) *Divisive Hierarchical Clustering*:
It is a top-down approach that starts with all data points in one cluster and repeatedly splits the clusters into smaller sub-clusters until each data point becomes its own cluster.

(c) *Commonly Used Method*:
Agglomerative clustering is more commonly used because it is computationally simpler, requires fewer assumptions, and works efficiently with distance matrices compared to divisive methods, which are more computationally intensive.

Q2.

(a) To improve clustering quality, **inter-cluster distance should be maximized**.
Maximizing the distance between clusters ensures that clusters are well separated and distinct from one another.

(b) Intra-cluster distance should be minimized.
Minimizing the distance within a cluster ensures that points inside each cluster are closely related or similar, increasing cohesion.

Q3.

(a) *Definitions*:

- **Single Link (Minimum Link)**: The distance between two clusters is defined as the shortest distance between any two points (one from each cluster).

- **Complete Link (Maximum Link)**: The distance between two clusters is defined as the farthest distance between any two points (one from each cluster).
- **Average Link**: The distance between two clusters is the average of all pairwise distances between points in the two clusters.

(b) *Strength and Weakness of Single Link*:

- **Strength**: Good at finding elongated or irregularly shaped clusters.
- **Weakness**: Sensitive to noise and chaining effects, which can cause dissimilar clusters to be linked through intermediate points.

Q4.

(a) *Role of Tokenization*:
Tokenization splits text into smaller units called tokens (words, subwords, or sentences). This step is essential for further NLP processing.
Example: The sentence "ChatGPT helps students." becomes tokens ["ChatGPT", "helps", "students", "."].

(b) *Stemming vs. Lemmatization*:

- **Stemming** is faster because it uses simple heuristic rules to chop off word endings.
- **Lemmatization** is slower but more accurate, as it uses vocabulary and morphological analysis to return the valid base form (lemma) of a word.

Q5.

(a) *Word Sense Ambiguity*:
Occurs when a word has multiple meanings depending on context.
Example: "Bank" can mean a financial institution or the side of a river.

(b) *Pronoun Reference Ambiguity*:
When a pronoun like "he", "she", or "it" can refer to multiple possible nouns, the model may

become confused.
Example: “John met David after he arrived.” — “he” could refer to John or David.

Q6.

(a) *Why POS tagging can’t predict independently*:
Part-of-Speech (POS) tagging depends on context. The correct tag for a word often depends on surrounding words (e.g., “book” can be a noun or a verb depending on context). Predicting independently would ignore this dependency.

(b) *Example of mutual dependency*:
In the sentence “She will book a ticket,” the word “book” is a **verb** because it follows “will,” a modal verb — showing dependency between tokens.
