Problem 1

Processes	1	2	4	8
1 million data items	0.004185	0.003205	0.002589	0.002278
10 millions data items	0.039432	0.024126	0.019019	0.016882

Problem 2

Processes	1	2	4	8
1 million data items	1	0.004185 / 0.003205 = 1.306	0.004185 / 0.002589 = 1.616	0.004185 / 0.002278 = 1.837
10 millions data items	1	0.039432 / 0.024126 = 1.634	0.039432 / 0.019019 = 2.073	0.039432 / 0.016882 = 2.336

Problem 3

Processes	1	2	4	8
1 million data items	1	1.306 / 2 = 0.653	1.616 / 4 = 0.404	1.837 / 8 = 0.230
10 millions data items	1	1.634 / 2 = 0.817	2.073 / 4 = 0.518	2.336 / 8 = 0.292

Problem 4

a. Both problem sizes seem to show a similar sub-linear speedup pattern that increases with more processes, but the gain decreases as the number of processes increases. Furthermore, the 10 million dataset achieves better speedup across all process counts compared to the smaller dataset.
b. The sub-linear pattern is due to the communication overhead, which explains why the speedup gain decreases as the number of processes increases. The 10 million data items problem size achieves better speedup because each process performs 10x more computation, so it better utilizes parallelism and offsets the impact of overhead.
c. Both problem sizes have a decrease in efficiency as processes increase. However, the 10 million data items problem size consistently has a higher efficiency.

d. The efficiency declines because adding processes increases communication overhead, thus reducing work per process. Similar to part b, the 10 million data items have an overall better efficiency than 1 million data items because the larger per-process workload better utilizes parallelism, thus offsetting the impact of communication overhead.

Problem 5

Processes	1	2	4	8
4 bins	0.003907	0.003198	0.002484	0.002229
10 bins	0.004057	0.003232	0.002520	0.002268

Problem 6

Based on the table, there's a very minimal effect on the performance, as we can see that there's only a small difference in time between 4 bins and 10 bins, no matter what the number of processes is. That said, this can be explained by how the performance for the histogram generation is mostly dominated by communication overhead from distributing the floats and combining results, in which both are independent of the number of bins.