Data Collected

Processes	1	2	4	8
Number of Data Items: 1 Million	0.003880	0.003290	0.001766	0.001040
Number of Data Items: 10 Million	0.036866	0.024531	0.020218	0.019473

The formula for calculation speedup is given below where, T(n) is time taken with n process and T(1) is time with process 1:

Speedup(n) = T(1) / T(n)

Processes	1	2	4	8
Number of Data Items: 1 Million	1	1.18	2.20	3.73
Number of Data Items: 10 Million	1	1.50	1.82	1.89

The formula for calculation efficiency is given below where, n is number of processes:

Efficiency(n) = Speedup(n) / n

Processes	1	2	4	8
Number of Data Items: 1 Million	1	0.59	0.55	0.47
Number of Data Items: 10 Million	1	0.75	.46	0.24

a) As the number of processes increases, speedup improves but does not scale perfectly. For 1 million items, speedup is reasonable, reaching 3.73 at 8 processes. However, for 10 million items, speedup is significantly lower at 1.89 at 8 processes, indicating that the program faces overhead challenges as data size increases.
b) Our results show that speedup improves as the number of processes increases, but it does not reach the result expected (Amdahl's Law). MPI overhead increases as we add more processes, which I think led to making parallel execution less useful.
c) Efficiency decreases as more processes are present, which is expected. For 1 million items, efficiency is around 50% at 4 processes and drops below 50% at 8 processes. For 10 million items, efficiency is even lower, showing that the parallel implementation is not having issues for larger problem sizes.
d) Efficiency decreases as the number of processes increases, which is expected in parallel computing. This is because MPI overhead increases with more processes. At lower processes like at 2 and 4, efficiency remains reasonable, but at 8 processes, it drops significantly. This suggests that beyond a certain point, adding more processes does not provide better performance.

Data Collected

Processes	1	2	4	8
Number of Bins: 4	0.004107	0.002963	0.001718	0.001042
Number of Bins: 10	0.003871	0.003009	0.001696	0.000998

We know parallelization distributes the workload effectively. Increasing the number of bins from 4 to 10 has a negligible effect on execution time. In some cases, performance is slightly better for 10 bins, likely due to memory optimizations. The reason for minimal performance impact is that increasing bins affects histogram calculation, as evident in the results above.