|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Avg CPU Used (≈ 100 − %idle) | Avg Disk %util | Notes |
| Bubble Sort | ≈ 1.9 % | ≈ 0.4 % | |  | | --- | |  |   CPU light, minimal disk I/O (in-memory) |
| Insertion Sort | |  | | --- | |  |   ≈ 1.5 % | |  | | --- | |  |   ≈ 0.5 % | Similar pattern, slightly higher peaks |
| Merge Sort | ≈ 3.0 % | |  | | --- | |  |   ≈ 0.6 % | More CPU work due to recursion splits |
| Quick Sort | ≈ 2.0 % | ≈ 0.3 % | Balanced – fastest and efficient overall |

Summary Table

**Analysis :**

During the performance testing, I observed that all four sorting algorithms used very low CPU and disk resources since the data was processed entirely in memory. Among them, Merge Sort showed slightly higher CPU activity due to its recursive nature, while Quick sort performed efficiently with balanced CPU usage. Overall, the results confirm that the “memory database” design minimizes disk operations and keeps system utilization low.