```
In [25]: from mpi4py import MPI
         import numpy as np

         def parallel_quicksort(data):
             comm = MPI.COMM_WORLD
             rank = comm.Get_rank()
             size = comm.Get_size()

             local_size = len(data) // size

             local_data = np.empty(local_size, dtype=int)
             comm.Scatter(data, local_data, root=0)

             print(f"Rank {rank}: Local Data (First 5 elements): {local_data[:5]}")

             local_data.sort()

             sorted_data = comm.gather(local_data, root=0)

             if rank == 0:

                 sorted_data = np.concatenate(sorted_data)
                 sorted_data.sort()
                 return sorted_data
             else:
                 return None

         if __name__ == "__main__":
             comm = MPI.COMM_WORLD
             rank = comm.Get_rank()

             np.random.seed(51)
             data = np.random.randint(0, 100, size=100)

             print(f"Rank {rank}: Original Data (First 10 elements): {data[:10]}")

             data = comm.bcast(data, root=0)

             sorted_data = parallel_quicksort(data)

             if rank == 0:
```

```
        print("Sorted Data (First 30 elements):", sorted_data[:10])
        print("Unique Sorted Data (First 10 elements):", set(sorted_data[:10]))
```

```
Rank 0: Original Data (First 10 elements): [57 96 73 69 16 21 94 52 37 37]
Rank 0: Local Data (First 5 elements): [57 96 73 69 16]
Sorted Data (First 30 elements): [ 0  1  1  3  5  6  7  8  9 10]
Unique Sorted Data (First 10 elements): {0, 1, 3, 5, 6, 7, 8, 9, 10}
```

In [ ]:

In [ ]: