Report of Data Storage:

Here is a brief analysis of the data loading times you've reported:

Loading Method	Part 1 Time (sec)	Part 2 Time (sec)
Simple Inserts	46.65	48.22
Drop Indexes and Constraints	42.23	43.48
Copy_from	5.32	6.52

Observations:

The **copy_from** method is significantly faster than the other two methods. This is because **copy_from** is a bulk operation that loads the data directly into the table with a single command, minimizing the overhead of executing individual insert commands.
Dropping indexes and constraints before inserting data improves performance. This is because indexes and constraints require additional checks and updates for every inserted row, which can significantly slow down the insertion process. After the data is loaded, the indexes and constraints can be re-created.
Despite the performance improvement from dropping indexes and constraints, this method is still much slower than **copy_from**. This is likely because the overhead of individual insert commands is still significant, even without the additional overhead of maintaining indexes and constraints.
Using simple inserts without any optimization is the slowest method. This is expected, as each insert command has a significant overhead, and indexes and constraints add additional overhead for each inserted row.
From these observations, we can conclude that for bulk loading of data, using a bulk operation method like **copy_from** provides the best performance. However, it's important to ensure the data is clean and well-formatted before using **copy_from**, as this method may not handle data inconsistencies as gracefully as individual insert commands. In situations where the data is not clean or well-formatted, or where more control over the insertion process is needed, using insert commands may still be the best option, despite the slower performance.