**Scalability Analysis**

The current script is designed to handle a relatively small dataset of five buildings. To scale this script to handle a dataset of 1,000,000 buildings, we need to evaluate its performance in terms of:

1. **Time Complexity:** The current script processes each building in a loop, resulting in a time complexity of 𝑂(𝑛)*O*(*n*), where 𝑛*n* is the number of buildings. While linear time complexity is efficient, the sheer volume of data will make processing time considerable.
2. **Space Complexity:** The script currently stores all building data and results in memory. For a dataset of 1,000,000 buildings, this approach will require significant memory, which might exceed the available RAM.

**Optimization Strategies**

1. **Data Loading and Processing:** Instead of loading the entire dataset into memory, we can process the data in chunks. This approach ensures that only a manageable portion of data is in memory at any given time.
2. **Parallel Processing:** Utilize multi-threading or multi-processing to parallelize the computation. This can significantly reduce the processing time by leveraging multiple CPU cores.
3. **Efficient Libraries:** Use optimized libraries such as NumPy and Pandas for numerical computations. These libraries are implemented in C and are optimized for performance.
4. **Disk-Based Storage:** Instead of storing intermediate results in memory, use disk-based storage solutions such as SQLite, HDF5, or even CSV files for intermediate storage.
5. **Memory Management:** Use generators and iterators to manage memory usage efficiently. This ensures that we process one building at a time rather than loading the entire dataset into memory.

**Resource Management**

1. **Memory Management:** Using generators to read and process data in chunks will prevent memory overflow. Tools like Python's **gc** (garbage collection) module can help manage memory by cleaning up unused objects.
2. **Batch Processing:** Implement batch processing to handle data in chunks. This method ensures that we do not overload the system's memory and can process data continuously in manageable segments.