Python Task:
create a script in python to split the input file (input.txt or input.csv provided below) into smaller ones grouped by "deal_id" column.
Zip the individual files after.

Your solution in the end should generate one .zip file, that will contain multiple *deal_id*.txt files.
Each *deal_id* file should contain only entries associated with that deal.
for example, in the 1.txt will be like
1,col_2_value_1047,col_3_value_1047,col_4_value_1047,col_5_value_1047..
1,col_2_value_1552,col_3_value_1552,col_4_value_1552,col_5_value_1552..

Note that the input file must not be loaded entirely into memory.

Assume the file can be as big as 20 GB and there can be max of 10000 deals (10000 files)

Provide an efficient solution taking care of processing time and memory.

Processing logic could be just in python or mix of python and external libraries. You can provide alternative solutions to the problem.

Additionally:
- Check how long does it take to process the whole file, and the peak memory usage.

Use this script below to generate **input.txt/input.csv** (input to your solution) with the following format
deal_id,col_2,col_3,col_4,col_5..
93,col_2_value_0,col_3_value_0,col_4_value_0,col_5_value_0..
..

```
import random

if __name__ == "__main__":
    filename = "./input.txt"
    total_lines = 1_000_000
    total_columns = 50
    min_deal_id = 1
    max_deal_id = 1000  # this can be upto 10000

    header_values = ["deal_id"]
    for i in range(2, total_columns + 1):
        header_values.append(f"col_{i}")

    header = ",".join(header_values)

    with open(filename, "w") as f:
        f.write(f"{header}\n")

        for i in range(total_lines):
            deal_id = str(random.randint(min_deal_id, max_deal_id))
            entry_values = [deal_id]
            for j in range(2, total_columns + 1):
                entry_values.append(f"col_{j}_value_{i}")

            entry = ",".join(entry_values)
            f.write(f"{entry}\n")
```