Large File Processing & Validation
Leena Ganta
LISUM45
06/12/2025
Data Glacier

Data Set Information:

Dataset Used	NYC Yellow Taxi Trip Data - January 2019
File Size	~1.8 GB raw CSV
Rows	7.696,617
Columns	19

```
import pandas as pd

df = pd.read_parquet("yellow_tripdata_2019-01.parquet")
df.to_csv("yellow_tripdata_2019-01.csv", index=False)

print(f"Saved CSV: {df.shape[0]:,} rows × {df.shape[1]} columns")
```

```
Saved CSV: 7,696,617 rows × 19 columns
```

File Reading Methods Comparison:

Method	Time(s)	Rows	Columns	Notes
PANDAS	26.16	7,696,617	19	Baseline CSV loader
Dask	26.04	7,696,617	19	Parallelized equally as fast as PANDAS
Modin	74.15	7,696,617	19	The Ray overhead slowed it down significantly compared to the other two methods

Large File Processing & Validation
Leena Ganta
LISUM45
06/12/2025
Data Glacier

```python
import pandas as pd
import time

start = time.time()

df_pandas = pd.read_csv("yellow_tripdata_2019-01.csv")

end = time.time()
print(f"Pandas read time: {round(end - start, 2)} seconds")
print(f"Shape: {df_pandas.shape}")
df_pandas.head()
```

```
Pandas read time: 26.16 seconds
Shape: (7696617, 19)
```

```python
import dask.dataframe as dd
import time

start = time.time()

df_dask = dd.read_csv("yellow_tripdata_2019-01.csv")
df_dask_result = df_dask.compute()

end = time.time()
print(f"Dask read time: {round(end - start, 2)} seconds")
print(f"Shape: {df_dask_result.shape}")
df_dask_result.head()
```

```
Dask read time: 26.04 seconds
Shape: (7696617, 19)
```

```
[16] !pip install -q modin[ray]
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 68.9/68.9 MB 11.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 47.4 MB/s eta 0:00:00
```

```python
import modin.pandas as mpd
import ray
import time

ray.shutdown()
ray.init(ignore_reinit_error=True)

start = time.time()
df_modin = mpd.read_csv("yellow_tripdata_2019-01.csv")
end = time.time()

print(f"Modin (Ray) read time: {round(end - start, 2)} seconds")
print(f"Shape: {df_modin.shape}")
df_modin.head()
```

```
2025-06-18 18:47:05,919 INFO worker.py:1917 -- Started a local Ray instance.
(raylet) [2025-06-18 18:48:05,840 E 8714 8714] (raylet) node_manager.cc:3193: 2 Workers (tasks / actors) killed d
(raylet)
(raylet) Refer to the documentation on how to address the out of memory issue: https://docs.ray.io/en/latest/ray-
Modin (Ray) read time: 74.15 seconds
Shape: (7696617, 19)
```

Large File Processing & Validation
Leena Ganta
LISUM45
06/12/2025
Data Glacier

Data Cleaning:
• Removed special characters from column names
• Converted all spaces to underscores to make it clean
• Lowercased all column names to maintain consistency

```python
import re

df_cleaned = df_pandas.copy()

def clean_col(col):
    col = col.strip()
    col = re.sub(r'[^\w\s]', '', col)
    col = re.sub(r'\s+', '_', col)
    return col.lower()

df_cleaned.columns = [clean_col(c) for c in df_cleaned.columns]

df_cleaned.columns.tolist()
```

```
['vendorid',
 'tpep_pickup_datetime',
 'tpep_dropoff_datetime',
 'passenger_count',
 'trip_distance',
 'ratecodeid',
 'store_and_fwd_flag',
 'pulocationid',
 'dolocationid',
 'payment_type',
 'fare_amount',
 'extra',
 'mta_tax',
 'tip_amount',
 'tip_amount',
 'tolls_amount',
 'improvement_surcharge',
 'total_amount',
 'congestion_surcharge',
 'airport_fee']
```

YAML Schema:
separator_read: ","
separator_write: "|"
columns: [vendorid, tpep_pickup_datetime, ..., airport_fee]
(saved as yellow_tripdata_schema.yaml)

Schema Validation:
• Column Count: Matches
• Column Names: Match exactly

Large File Processing & Validation
Leena Ganta
LISUM45
06/12/2025
Data Glacier

```yaml
import yaml

schema = {
    "separator_read": ",",
    "separator_write": "|",
    "columns": df_cleaned.columns.tolist()
}

with open("yellow_tripdata_schema.yaml", "w") as f:
    yaml.dump(schema, f)

print("The YAML schema saved as 'yellow_tripdata_schema.yaml'")
```

```
The YAML schema saved as 'yellow_tripdata_schema.yaml'
```

```yaml
with open("yellow_tripdata_schema.yaml", "r") as f:
    schema_loaded = yaml.safe_load(f)

expected_cols = schema_loaded["columns"]
actual_cols = df_cleaned.columns.tolist()

if len(expected_cols) != len(actual_cols):
    print(f"There is a mismatch in number of columns: Expected {len(expected_cols)}, Found {len(actual_cols)}")
else:
    print("The number of columns matches!")

if expected_cols == actual_cols:
    print("The Column names match the schema exactly!")
else:
    print("The Column names do not match:")
    for i, (exp, act) in enumerate(zip(expected_cols, actual_cols)):
        if exp != act:
            print(f"  → Mismatch at position {i}: Expected '{exp}' vs Found '{act}'")
```

```
The number of columns matches!
The Column names match the schema exactly!
```

File Export & Summary:
• Saved cleaned file in .psv.gz format
• Separator: | (pipe)
• Compression: gzip
• Output filename: yellow_tripdata_cleaned.psv.gz

• Final File Size: 123.79 MB
• Total Rows: 7,696,617
• Total Columns: 19

Large File Processing & Validation
Leena Ganta
LISUM45
06/12/2025
Data Glacier

```
output_path = "yellow_tripdata_cleaned.psv.gz"

df_cleaned.to_csv(
    output_path,
    sep="|",
    index=False,
    compression="gzip"
)

print(f"File saved as: {output_path}")
```

```
File saved as: yellow_tripdata_cleaned.psv.gz
```

```
import os

# Get basic summary
num_rows, num_cols = df_cleaned.shape
file_size_bytes = os.path.getsize("yellow_tripdata_cleaned.psv.gz")
file_size_mb = round(file_size_bytes / (1024 * 1024), 2)

print("📄 File Summary:")
print(f"• Total Rows    : {num_rows}")
print(f"• Total Columns : {num_cols}")
print(f"• File Size     : {file_size_mb} MB")
```

```
📄 File Summary:
• Total Rows    : 7696617
• Total Columns : 19
• File Size     : 123.79 MB
```