Chicago Traffic Crashes - Data Processing

Name: Kirtoria Ward

Batch Code: LISUM38

Submission Date: 12/27/2024

Submitted to: GitHub

Project Explanation:

This project focuses on processing the Chicago Traffic Crashes dataset (2+ GB) by reading the file using different methods (Pandas, Dask, Modin, Ray), validating data against a YAML schema, and saving the cleaned dataset in a pipe-separated gzip format. Performance was compared to evaluate computational efficiency.

File Reading Performance:

Method	Time (Seconds)
Pandas	17.25
Dask	0.22
Modin	26.12
Ray	1.05

Pandas Readtime Code and Output

```
[3] import pandas as pd
    import time

    # Measure time taken to read with Pandas
    start = time.time()
    df_pandas = pd.read_csv(file_path)
    pandas_time = time.time() - start

    print(f"Pandas read time: {pandas_time:.2f} seconds")
```

```
Pandas read time: 17.25 seconds
```

Dask Readtime Code and Ouput:

```
[4] import dask.dataframe as dd

    # Measure time taken to read with Dask
    start = time.time()
    df_dask = dd.read_csv(file_path)
    dask_time = time.time() - start

    print(f"Dask read time: {dask_time:.2f} seconds")
```

```
/usr/local/lib/python3.10/dist-packages/dask/dataframe/__init__.py:42: FutureWarning:
Dask dataframe query planning is disabled because dask-expr is not installed.

You can install it with `pip install dask[dataframe]` or `conda install dask`.
This will raise in a future version.

  warnings.warn(msg, FutureWarning)
Dask read time: 0.22 seconds
```

Modin Readtime Code and Ouput:

```
[12] !pip install modin
     import modin.pandas as mpd
     import time

     # Measure time taken to read with Modin
     start = time.time()
     df_modin = mpd.read_csv(file_path)
     modin_time = time.time() - start

     print(f"Modin read time: {modin_time:.2f} seconds")
```

```
Requirement already satisfied: modin in /usr/local/lib/python3.10/dist-packages (0.32.0)
Requirement already satisfied: pandas<2.3,>=2.2 in /usr/local/lib/python3.10/dist-packages (from modin) (2.2.2)
Requirement already satisfied: packaging>=21.0 in /usr/local/lib/python3.10/dist-packages (from modin) (24.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from modin) (1.26.4)
Requirement already satisfied: fsspec>=2022.11.0 in /usr/local/lib/python3.10/dist-packages (from modin) (2024.10.0)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-packages (from modin) (5.9.5)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas<2.3,>=2.2->modin) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<2.3,>=2.2->modin) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas<2.3,>=2.2->modin) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas<2.3,>=2.2->modin) (1.17.0)
UserWarning: The size of /dev/shm is too small (6133121024 bytes). The required size at least half of RAM (6804715520 bytes). Please, delete files in /dev
2024-12-27 23:55:01,762 INFO worker.py:1821 -- Started a local Ray instance.
Modin read time: 26.12 seconds
```

Ray Readtime Code and Ouput:

```
[9] !pip install ray  # Install Ray
    import ray
    import ray.data

    ray.init(ignore_reinit_error=True)

    # Measure time taken to read with Ray
    start = time.time()
    df_ray = ray.data.read_csv(file_path)
    ray_time = time.time() - start

    print(f"Ray read time: {ray_time:.2f} seconds")

    ray.shutdown()
```

```
Collecting ray
  Downloading ray-2.40.0-cp310-cp310-manylinux2014_x86_64.whl.metadata (17 kB)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.10/dist-packages (from ray) (8.1.7)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from ray) (3.16.1)
Requirement already satisfied: jsonschema in /usr/local/lib/python3.10/dist-packages (from ray) (4.23.0)
Requirement already satisfied: msgpack<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ray) (1.1.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from ray) (24.2)
Requirement already satisfied: protobuf!=3.19.5,>=3.15.3 in /usr/local/lib/python3.10/dist-packages (from ray) (4.25.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from ray) (6.0.2)
Requirement already satisfied: aiosignal in /usr/local/lib/python3.10/dist-packages (from ray) (1.3.2)
Requirement already satisfied: frozenlist in /usr/local/lib/python3.10/dist-packages (from ray) (1.5.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from ray) (2.32.3)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema->ray) (24.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema->ray) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema->ray) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema->ray) (0.22.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->ray) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->ray) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->ray) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->ray) (2024.12.14)
Downloading ray-2.40.0-cp310-cp310-manylinux2014_x86_64.whl (66.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.8/66.8 MB 10.9 MB/s eta 0:00:00
Installing collected packages: ray
Successfully installed ray-2.40.0
2024-12-27 23:53:16,282 INFO worker.py:1821 -- Started a local Ray instance.
Ray read time: 1.05 seconds
```

The summary ouput of all readtimes:

```
[13] print("Summary of File Reading Performance:")
     print("==================================")
     print(f"Pandas read time: {pandas_time:.2f} seconds")
     print(f"Dask read time: {dask_time:.2f} seconds")
     print(f"Modin read time: {modin_time:.2f} seconds")
     print(f"Ray read time: {ray_time:.2f} seconds")
```

```
Summary of File Reading Performance:
==================================
Pandas read time: 17.25 seconds
Dask read time: 0.22 seconds
Modin read time: 26.12 seconds
Ray read time: 1.05 seconds
```

Data Validation and Cleaning:

Column names were cleaned to remove special characters and whitespace. A YAML schema was created to define the dataset's structure and validate the number of columns and their names.

Cleaned Column Names:

```
[14] # Clean column names: remove special characters, replace spaces with underscores
     df_pandas.columns = df_pandas.columns.str.replace('[^A-Za-z0-9]+', '_').str.strip()
     print("Cleaned Column Names:")
     print(df_pandas.columns)

Cleaned Column Names:
Index(['CRASH_RECORD_ID', 'CRASH_DATE_EST_I', 'CRASH_DATE',
       'POSTED_SPEED_LIMIT', 'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION',
       'WEATHER_CONDITION', 'LIGHTING_CONDITION', 'FIRST_CRASH_TYPE',
       'TRAFFICWAY_TYPE', 'LANE_CNT', 'ALIGNMENT', 'ROADWAY_SURFACE_COND',
       'ROAD_DEFECT', 'REPORT_TYPE', 'CRASH_TYPE', 'INTERSECTION_RELATED_I',
       'NOT_RIGHT_OF_WAY_I', 'HIT_AND_RUN_I', 'DAMAGE', 'DATE_POLICE_NOTIFIED',
       'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'STREET_NO',
       'STREET_DIRECTION', 'STREET_NAME', 'BEAT_OF_OCCURRENCE',
       'PHOTOS_TAKEN_I', 'STATEMENTS_TAKEN_I', 'DOORING_I', 'WORK_ZONE_I',
       'WORK_ZONE_TYPE', 'WORKERS_PRESENT_I', 'NUM_UNITS',
       'MOST_SEVERE_INJURY', 'INJURIES_TOTAL', 'INJURIES_FATAL',
       'INJURIES_INCAPACITATING', 'INJURIES_NON_INCAPACITATING',
       'INJURIES_REPORTED_NOT_EVIDENT', 'INJURIES_NO_INDICATION',
       'INJURIES_UNKNOWN', 'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MONTH',
       'LATITUDE', 'LONGITUDE', 'LOCATION'],
      dtype='object')
```

Yaml Schema File Creation:

```
[15] !pip install pyyaml
     import yaml

     # Define schema
     schema = {
         'separator': '|',
         'columns': list(df_pandas.columns)
     }

     # Save schema to a YAML file
     with open('schema.yaml', 'w') as file:
         yaml.dump(schema, file)

     print("Schema file 'schema.yaml' created successfully.")

Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (6.0.2)
Schema file 'schema.yaml' created successfully.
```

Validation Code and Ouput:

```
[16] # Load the schema
     with open('schema.yaml') as file:
         schema = yaml.load(file, Loader=yaml.FullLoader)

     # Validate column names
     assert list(df_pandas.columns) == schema['columns'], "Column names do not match!"
     print("Column names match the schema.")

Column names match the schema.
```

File Saving:

The cleaned dataset was saved as a pipe-separated gzip file, reducing file size while maintaining data integrity

The gzip file creation code:

```
[17] # Save the cleaned dataset in pipe-separated gzip format
     df_pandas.to_csv('Chicago_Traffic_Crashes_cleaned.txt.gz', sep='|', index=False, compression='gzip')

     print("Dataset saved successfully in pipe-separated gzip format.")

Dataset saved successfully in pipe-separated gzip format.
```

Dataset Summary:

Total Rows	Total Columns	File Size
904032	48	130.55MB

Code for Summary of Dataset:

```
[18] import os

     # Total rows and columns
     rows, columns = df_pandas.shape

     # File size
     file_size = os.path.getsize('Chicago_Traffic_Crashes_cleaned.txt.gz')

     # Print summary
     print("Summary of the Cleaned Dataset:")
     print(f"Total Rows: {rows}")
     print(f"Total Columns: {columns}")
     print(f"File Size: {file_size / (1024 * 1024):.2f} MB")

Summary of the Cleaned Dataset:
Total Rows: 904032
Total Columns: 48
File Size: 130.55 MB
```

Project Summary

This project involved processing the Chicago Traffic Crashes dataset (2+ GB) to demonstrate efficient data handling and validation techniques. The dataset was read using four methods: **Pandas**, **Dask**, **Modin**, and **Ray**, with computational times compared to evaluate performance. **Dask** proved to be the fastest with a read time of 0.22 seconds, followed by **Ray** (1.05 seconds), **Pandas** (17.25 seconds), and **Modin** (26.12 seconds). After reading the data, column names were cleaned, and a YAML schema was created to validate the dataset structure. The cleaned dataset was saved in a pipe-separated gzip format, resulting in a file of **130.55MB**. The processed dataset contains **904032** rows and **48** columns, offering a compact and structured output for further analysis.