Data Ingestion and Schema Validation

Name: Nonhlanhla Luphade

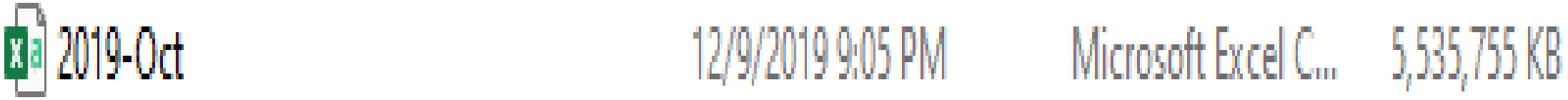
Batch Code: LISP01

Submission Date: 4/11/21

Submitted to: Data Glacier

The data file

- The data file used for this task is:
  - File size: 5.5 GB
  - Obtained from: kaggle



Reading file using Pandas

```
#read file with pandas
import pandas as pd
import time as time
s = time.time()
%time df = pd.read_csv("/content/drive/MyDrive/2019-Oct.csv")
e = time.time()
print("Pandas Loading Time = {}".format(e-s))
```

```
CPU times: user 1min 16s, sys: 22.4 s, total: 1min 39s
Wall time: 2min 14s
Pandas Loading Time = 135.28663802146912
```

- The pandas managed to read the file; however, they were slow and took a while to load the data.

Reading the file with Modin (Ray)

```
#read file with modin
import time as time
import modin.pandas as md
s = time.time()
%time df = md.read_csv("/content/drive/MyDrive/2019-Oct.csv")
e = time.time()
print("Modin ray Loading Time = {}".format(e-s))
```

```
(pid=1699) tcmalloc: large alloc 2834309120 bytes == 0x55fba3ac2000 @  0x7f75da6191e7 0x55fba0be0f48 0x55fba0bab9c7 0x55fba
(pid=1698) tcmalloc: large alloc 2834309120 bytes == 0x561d2d780000 @  0x7fab6042b1e7 0x561d2a87ef48 0x561d2a8499c7 0x561d2
2021-04-11 22:58:49,942 WARNING worker.py:1034 -- A worker died or was killed while executing task bd37d2621480fc7dffffffff
(pid=1834) tcmalloc: large alloc 2834309120 bytes == 0x55a593f68000 @  0x7f589ae9f1e7 0x55a590a7ff48 0x55a590a4a9c7 0x55a59
2021-04-11 23:00:47,150 WARNING worker.py:1034 -- A worker died or was killed while executing task bd37d2621480fc7dffffffff
(pid=1698) tcmalloc: large alloc 2834309120 bytes == 0x561d2d780000 @  0x7fab6042b1e7 0x561d2a87ef48 0x561d2a8499c7 0x561d2
```

- When modin was used to read the big file, it failed and was using up a lot of memory. It took a long time and did not manage to read the file.

Reading with dask

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

```
[ ] import time as time
    s = time.time()
    %time  df = dd.read_csv("/content/drive/MyDrive/2019-Oct.csv")
    e = time.time()
    print("Pandas Loading Time = {}".format(e-s))

    CPU times: user 61.9 ms, sys: 4.02 ms, total: 66 ms
    Wall time: 75.8 ms
    Pandas Loading Time = 0.07826638221740723
```

- Reading the big file with dask was simple and fast. It took less than 10 seconds to read the csv file.

Conclusion

In terms of computer efficiency, I believe dask will be best for this data and most large data sets in general because of its speed and efficiency. It is also mature and has many resources online for those who are new. Unlike Ray and Modin. Additionally, all the other methods make use of pandas in some way or another. Pandas are good for small datasets, however, when it comes to large datasets, they become slow and less efficient.