Name: İrem Tanrıverdi
Batch code: LISP01
Submission date: 11/04/2021
Submitted to: Data Glacier

pandas.py | untitled0.py* | ymal.py* | testutility.py | file.yaml

```python
# I have dataset whose size is 4GB
# First, I try to read the data with pandas

import pandas as pd

df=pd.read_csv('chess_games.csv', sep=',') #data were not read


# Secondly, I try to read with Modin
import modin.pandas as pd
from distributed import Client
client = Client()
data = pd.read_csv('chess_games.csv', sep=',') #taking long time to read
data

# Thirdly, I try to read the data with Dask

import dask.dataframe as dd

df = dd.read_csv('chess_games.csv', sep=',') #dataset readed easily
df
```

Firstly, dataset was tried to read using different methods like pandas, Modin and Dask.

- When I used pandas data were not read. It gives timeout error.
- Then I used Modin, again it took a long time to read the data.
- Finally I used Dask, and it worked. Dataset was read immediately.

Creating YAML file

Console 1/A

```
In [1]: %%writefile testutility.py
   ...: import logging
   ...: import os
   ...: import subprocess
   ...: import yaml
   ...: import datetime
   ...: import gc
   ...: import re
   ...:
   ...: ################
   ...: # File Reading #
   ...: ################
   ...:
   ...: def read_config_file(filepath):
   ...:     with open(filepath, "r") as stream:
   ...:         try:
   ...:             return yaml.safe_load(stream)
   ...:         except yaml.YAMLError as exc:
   ...:             logging.error(exc)
   ...:
   ...: def replacer(string, char):
   ...:     pattern= char + '{2,}'
   ...:     string= re.sub(pattern, char, string)
   ...:     return string
   ...:
   ...: def col_header_val(df, table_config):
   ...:
   ...:     df.columns= df.columns.str.lower()
   ...:     df.columns = df.columns.str.replace('[^\w]', '_', regex= True)
   ...:     df.columns= list(map(lambda x: x.strip('_'), list(df.columns)))
   ...:     df.columns= list(map(lambda x: replacer(x,'_'), list(df.columns)))
   ...:     expected_col= list(map(lambda x: x.lower(), table_config['columns']))
   ...:     expected_col.sort()
   ...:     df.columns= list(map(lambda x: x.lower(), list(df.columns)))
   ...:     df= df.reindex(sorted(df.columns), axis=1)
   ...:     if len(df.columns)==len(expected_col) and list(expected_col)== list(df.columns):
   ...:         print("column validation is passed")
   ...:         return 1
   ...:     else:
   ...:         print("column validation is failed")
   ...:         mismatced_column_file= list(set(df.columns).difference(expected_col))
   ...:         print("following file columns are not in the yaml file", mismatced_column_file)
   ...:         missing_YAML_file = list(set(expected_col).difference(df.columns))
   ...:         print("following yml columns are not in the file uploaded", missing_YAML_file)
   ...:         logging.info(f'df columns: {df.columns}')
   ...:         logging.info(f'expected columns : {expected_col}')
   ...:         return 0
Overwriting testutility.py
```

Created test utility file

/Users/irem/Desktop/week6/testutility.py

pandas.py | untitled0.py | ymal.py | testutility.py | file.yaml

```python
import logging
import os
import subprocess
import yaml
import datetime
import gc
import re

###############
# File Reading #
###############

def read_config_file(filepath):
    with open(filepath, "r") as stream:
        try:
            return yaml.safe_load(stream)
        except yaml.YAMLError as exc:
            logging.error(exc)

def replacer(string, char):
    pattern= char + '{2,}'
    string= re.sub(pattern, char, string)
    return string

def col_header_val(df, table_config):

    df.columns= df.columns.str.lower()
    df.columns = df.columns.str.replace('[^\w]', '_', regex= True)
    df.columns= list(map(lambda x: x.strip('_'), list(df.columns)))
    df.columns= list(map(lambda x: replacer(x,'_'), list(df.columns)))
    expected_col= list(map(lambda x: x.lower(), table_config['columns']))
    expected_col.sort()
    df.columns= list(map(lambda x: x.lower(), list(df.columns)))
    df= df.reindex(sorted(df.columns), axis=1)
    if len(df.columns)==len(expected_col) and list(expected_col)== list(df.columns):
        print("column validation is passed")
        return 1
    else:
        print("column validation is failed")
        mismatced_column_file= list(set(df.columns).difference(expected_col))
        print("following file columns are not in the yaml file", mismatced_column_file)
        missing_YAML_file = list(set(expected_col).difference(df.columns))
        print("following yml columns are not in the file uploaded", missing_YAML_file)
        logging.info(f'df columns: {df.columns}')
        logging.info(f'expected columns : {expected_col}')
        return 0
```

Then YAML file was created:

```
Console 1/A

In [12]: %%writefile file.yaml
    ...: file_type: csv
    ...: dataset_name: chess_games
    ...: file_name: chess_games
    ...: table_name: chess_games
    ...: inbound_delimiter: ","
    ...: outbound_delimiter: ","
    ...: skip_leading_rows: 1
    ...: columns:
    ...:     - Event
    ...:     - White
    ...:     - Black
    ...:     - Result
    ...:     - UTCDate
    ...:     - UTCTime
    ...:     - WhiteElo
    ...:     - BlackElo
    ...:     - WhiteRatingDiff
    ...:     - BlackRatingDiff
    ...:     - Opening
    ...:     - TimeControl
    ...:     - Termination
    ...:     - AN
Overwriting file.yaml
```

Created YAML file

```
pandas.py | untitled0.py | ymal.py* | testutility.py | file.yaml

1  file_type: csv
2  dataset_name: chess_games
3  file_name: chess_games
4  table_name: chess_games
5  inbound_delimiter: ","
6  outbound_delimiter: ","
7  skip_leading_rows: 1
8  columns:
9      - Event
10     - White
11     - Black
12     - Result
13     - UTCDate
14     - UTCTime
15     - WhiteElo
16     - BlackElo
17     - WhiteRatingDiff
18     - BlackRatingDiff
19     - Opening
20     - TimeControl
21     - Termination
22     - AN
```

Console 1/A

```
In [3]: import testutility as util

In [4]: config_data= util.read_config_file("file.yaml")

In [5]: config_data['file_type']
Out[5]: 'csv'

In [6]: config_data
Out[6]:
{'file_type': 'csv',
 'dataset_name': 'chess_games',
 'file_name': 'chess_games',
 'table_name': 'chess_games',
 'inbound_delimiter': ',',
 'outbound_delimiter': ',',
 'skip_leading_rows': 1,
 'columns': ['id',
  'rated',
  'creat_at',
  'last_move_at',
  'turns',
  'victory_status',
  'winner',
  'increment_code',
  'white_id',
  'white_rating',
  'black_id',
  'black_rating',
  'moves',
  'opening_eco',
  'opening_name',
  'opwning_ply']}
```

Console 1/A

```
In [17]: import pandas as pd

In [18]: file_type= config_data['file_type']

In [19]: source_file= "./" + config_data['file_name'] + f'.{file_type}'

In [20]: df= pd.read_csv(source_file, config_data['inbound_delimiter'])

In [21]: util.col_header_val(df, config_data)
column validation is failed
following file columns are not in the yaml file ['eco']
following yml columns are not in the file uploaded []
Out[21]: 0
```

Console 1/A

```
In [22]: print("columns of files are:" ,df.columns)
columns of files are: Index(['event', 'white', 'black', 'result', 'utcdate', 'utctime', 'whiteelo',
       'blackelo', 'whiteratingdiff', 'blackratingdiff', 'eco', 'opening',
       'timecontrol', 'termination', 'an'],
      dtype='object')

In [23]: print("columns of YAML are:", config_data['columns'])
columns of YAML are: ['Event', 'White', 'Black', 'Result', 'UTCDate', 'UTCTime', 'WhiteElo', 'BlackElo',
'WhiteRatingDiff', 'BlackRatingDiff', 'Opening', 'TimeControl', 'Termination', 'AN']
```

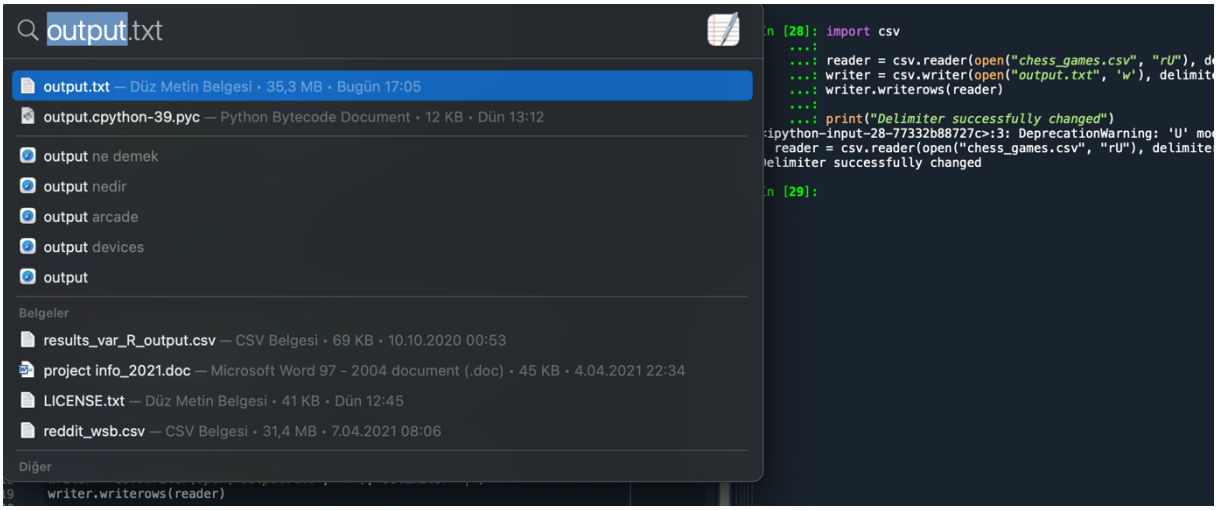
Console 1/A

```
In [24]: if util.col_header_val(df, config_data)==0:
    ...:     print("validation is failed")
    ...:
    ...: else:
    ...:     print("validation is passed")
column validation is failed
following file columns are not in the yaml file ['eco']
following yml columns are not in the file uploaded []
validation is failed
```

```
Console 1/A

In [25]: pd.read_csv("./chess_games.csv")
Out[25]:
                     Event  ...                                                AN
0                Classical  ...  1. d4 d5 2. c4 c6 3. e3 a6 4. Nf3 e5 5. cxd5 e...
1                    Blitz  ...  1. e4 e5 2. b3 Nf6 3. Bb2 Nc6 4. Nf3 d6 5. d3 ...
2          Blitz tournament  ...  1. e4 d5 2. exd5 Qxd5 3. Nf3 Bg4 4. Be2 Nf6 5....
3            Correspondence  ...  1. e3 Nf6 2. Bc4 d6 3. e4 e6 4. Nf3 Nxe4 5. Nd...
4          Blitz tournament  ...  1. e4 c5 2. Nf3 d6 3. d4 cxd4 4. Nxd4 Nf6 5. N...
...                     ...  ...                                                ...
6256179              Blitz  ...  1. e4 c5 2. Nf3 d6 3. d4 Qa5+ 4. Bd2 Qc7 5. Bc...
6256180          Classical  ...  1. e4 e5 2. Nf3 h6 3. Nc3 d6 4. Bc4 Qf6 5. Nd5...
6256181             Bullet  ...  1. e4 { [%eval 0.22] } 1... c5 { [%eval 0.35] ...
6256182             Bullet  ...  1. Nf3 d5 2. c4 dxc4 3. Qa4+ c6 4. Qxc4 Nf6 5....
6256183          Classical  ...  1. e4 e6 2. d4 d6 3. Nc3 c6 4. Nf3 Be7 5. Bc4 ...

[6256184 rows x 15 columns]
```

Then I write the file in pipe separated text file format:

```
Console 1/A

In [28]: import csv
    ...:
    ...: reader = csv.reader(open("chess_games.csv", "rU"), delimiter=',')
    ...: writer = csv.writer(open("output.txt", 'w'), delimiter='|')
    ...: writer.writerows(reader)
    ...:
    ...: print("Delimiter successfully changed")
<ipython-input-28-77332b88727c>:3: DeprecationWarning: 'U' mode is deprecated
  reader = csv.reader(open("chess_games.csv", "rU"), delimiter=',')
Delimiter successfully changed
```



And it is created txt file

output.txt

```
Event|White|Black|Result|UTCDate|UTCTime|WhiteElo|BlackElo|WhiteRatingDiff|BlackRatingDiff|ECO|
Opening|TimeControl|Termination|AN
 Classical |eisaaaa|HAMID449|1-0|2016.06.30|22:00:01|1901|1896|11.0|-11.0|D10|Slav Defense|300+5|Time
forfeit|1. d4 d5 2. c4 c6 3. e3 a6 4. Nf3 e5 5. cxd5 e4 6. Ne5 cxd5 7. Qa4+ Bd7 8. Nxd7 Nxd7 9. Nc3
Nf6 10. Qb3 Be7 11. Nxd5 Qa5+ 12. Nc3 O-O 13. Be2 b5 14. O-O Rad8 15. Bd2 Qc7 16. Rac1 Qd6 17. Qc2
Qe6 18. Nb1 Bd6 19. a3 Nb6 20. Qc6 Nfd5 21. Ba5 Rc8 22. Qb7 Qh6 23. h3 Nc4 24. Bxc4 bxc4 25. Qxd5
Rfd8 26. Qxe4 Rd7 27. Bc3 Re7 28. Qf3 Re6 29. Nd2 Rf6 30. Qg4 Re8 31. Ne4 Rg6 32. Qd7 Rf8 33. Nxd6
Rxd6 34. Qc7 Rg6 35. Qh2 Re8 36. d5 f6 37. d6 Rd8 38. Rfd1 1-0
 Blitz |go4jas|Sergei1973|0-1|2016.06.30|22:00:01|1641|1627|-11.0|12.0|C20|King's Pawn Opening: 2.b3|
300+0|Normal|1. e4 e5 2. b3 Nf6 3. Bb2 Nc6 4. Nf3 d6 5. d3 g6 6. Nbd2 Bg7 7. g3 Be6 8. Bg2 Qd7 9. O-O
O-O 10. c3 b5 11. d4 exd4 12. cxd4 Bg4 13. Rc1 Rfe8 14. Qc2 Nb4 15. Qxc7 Qxc7 16. Rxc7 Nxa2 17. Ra1
Nb4 18. Raxa7 Rxa7 19. Rxa7 Nxe4 20. Nxe4 Rxe4 21. Ng5 Re1+ 22. Bf1 Be2 23. Rxf7 Bxf1 24. Kh1 Bh3#
0-1
 Blitz tournament |Evangelistaizac|kafune|1-0|2016.06.30|22:00:02|1647|1688|13.0|-13.0|B01|
Scandinavian Defense: Mieses-Kotroc Variation|180+0|Time forfeit|1. e4 d5 2. exd5 Qxd5 3. Nf3 Bg4 4.
Be2 Nf6 5. Nc3 Qd8 6. O-O Nc6 7. Bc4 e6 8. h3 Bh5 9. d3 Bd6 10. Bg5 Ne5 11. g4 Nxf3+ 12. Qxf3 Bg6 13.
Bxf6 gxf6 14. h4 h5 15. gxh5 Bxh5 16. Qxb7 Bg4 17. Qc6+ Ke7 18. Rae1 Rxh4 19. Nd5+ Kf8 20. Nxf6 Qxf6
21. Qxa8+ Kg7 22. Qg2 Bh2+ 23. Qxh2 Rxh2 24. Kxh2 Qh4+ 25. Kg2 Qh3+ 26. Kg1 Qf3 27. Re3 Qf5 28. Rg3
Kf8 29. f3 Bh5 30. Rf2 Bg6 31. Rh2 1-0
 Correspondence |Jvayne|Wsjvayne|1-0|2016.06.30|22:00:02|1706|1317|27.0|-25.0|A00|Van't Kruijs
Opening|-|Normal|1. e3 Nf6 2. Bc4 d6 3. e4 e6 4. Nf3 Nxe4 5. Nd4 Nxf2 6. Kxf2 Qh4+ 7. g3 Qxg3+ 8.
Kxg3 d5 9. Bb5+ c6 10. Bf1 e5 11. Nf3 e4 12. Ne5 f6 13. Ng4 h5 14. Nf2 h4+ 15. Kf4 Bh3 16. Nxh3 g5+
17. Nxg5 fxg5+ 18. Kxg5 Rh5+ 19. Qxh5+ Kd7 20. Qf7+ Kd6 21. Qxf8+ Kc7 22. Qf4+ Kb6 23. Qf2+ Kc7 24.
Qf4+ Kc8 25. Bh3+ Kd8 26. Qd6+ Nd7 27. Bxd7 d4 28. d3 exd3 29. cxd3 b5 30. Kxh4 Rb8 31. Bg5# 1-0
 Blitz tournament |kyoday|BrettDale|0-1|2016.06.30|22:00:02|1945|1900|-14.0|13.0|B90|Sicilian
Defense: Najdorf, Lipnitsky Attack|180+0|Time forfeit|1. e4 c5 2. Nf3 d6 3. d4 cxd4 4. Nxd4 Nf6 5.
Nc3 a6 6. Bc4 Qc7 7. Bb3 b5 8. O-O Bb7 9. Re1 Nbd7 10. Be3 g6 11. f3 Bg7 12. Qe2 O-O 13. Rad1 b4 14.
Na4 a5 15. Qf2 Rfc8 16. Ne2 Qd8 17. Bb6 Qe8 18. Be3 Ra6 19. c4 Nh5 20. Ng3 Nxg3 21. hxg3 Bc6 22. Qd2
Be5 23. Kf2 Ba8 24. Bd4 Rac6 25. f4 Bf6 26. g4 Nc5 27. Nxc5 dxc5 28. Be3 e5 29. f5 Be7 30. Kf3 Rd6
31. Qe2 Rcd8 32. Rxd6 Rxd6 33. Rd1 Qd7 34. Rxd6 Qxd6 35. Qd1 Qf6 36. Qd7 Bc6 37. Qc8+ Kg7 38. g5 Qd6
```

Then I obtained the dimention of the data and size of the data.

Console 1/A

```
In [35]: import enum
    ...: # Enum for size units
    ...: class SIZE_UNIT(enum.Enum):
    ...:     BYTES = 1
    ...:     KB = 2
    ...:     MB = 3
    ...:     GB = 4
    ...: def convert_unit(size_in_bytes, unit):
    ...:     """ Convert the size from bytes to other units like KB, MB or GB"""
    ...:     if unit == SIZE_UNIT.KB:
    ...:         return size_in_bytes/1024
    ...:     elif unit == SIZE_UNIT.MB:
    ...:         return size_in_bytes/(1024*1024)
    ...:     elif unit == SIZE_UNIT.GB:
    ...:         return size_in_bytes/(1024*1024*1024)
    ...:     else:
    ...:         return size_in_bytes
    ...: import os
    ...: def get_file_size(file_name, size_type = SIZE_UNIT.BYTES ):
    ...:     """ Get file in size in given unit like KB, MB or GB"""
    ...:     size = os.path.getsize(file_name)
    ...:     return convert_unit(size, size_type)
    ...:
    ...: file_path = 'chess_games.csv'
    ...: # get file size in GB
    ...: size = get_file_size(file_path, SIZE_UNIT.GB)
    ...: print('Size of file is : ', size ,  'GB')
Size of file is :  4.07816391158849 GB
```

Console 1/A

```
In [31]:
    ...: rows = len(df.axes[0])
    ...:
    ...: # computing number of columns
    ...: cols = len(df.axes[1])
    ...:
    ...: print("Number of Rows: ", rows)
    ...: print("Number of Columns: ", cols)
Number of Rows:  6256184
Number of Columns:  15
```