w6

July 11, 2022

[1]:
```
import pandas as pd
from datetime import datetime
import os
import dask.dataframe as dd
import modin.pandas as mp
import ray
import logging
import subprocess
import yaml
import gc
import re
import testutility as util
import gzip
import csv
```

```
/Applications/python-anaconda/install/anaconda3/lib/python3.8/site-
packages/dask/dataframe/utils.py:369: FutureWarning: pandas.Int64Index is
deprecated and will be removed from pandas in a future version. Use pandas.Index
with the appropriate dtype instead.
  _numeric_index_types = (pd.Int64Index, pd.Float64Index, pd.UInt64Index)
/Applications/python-anaconda/install/anaconda3/lib/python3.8/site-
packages/dask/dataframe/utils.py:369: FutureWarning: pandas.Float64Index is
deprecated and will be removed from pandas in a future version. Use pandas.Index
with the appropriate dtype instead.
  _numeric_index_types = (pd.Int64Index, pd.Float64Index, pd.UInt64Index)
/Applications/python-anaconda/install/anaconda3/lib/python3.8/site-
packages/dask/dataframe/utils.py:369: FutureWarning: pandas.UInt64Index is
deprecated and will be removed from pandas in a future version. Use pandas.Index
with the appropriate dtype instead.
  _numeric_index_types = (pd.Int64Index, pd.Float64Index, pd.UInt64Index)
```

Different approaches to read the file

[2]:
```
# use pandas to read the file
start_time = datetime.now()
df_pd=pd.read_csv('credit_card_transactions-ibm_v2.csv')
end_time = datetime.now()
result = end_time - start_time
print(result)
```

```
0:00:33.137170
```

[3]:
```python
# use dask to read the file
start_time = datetime.now()
df_dd=dd.read_csv('credit_card_transactions-ibm_v2.csv')
end_time = datetime.now()
result = end_time - start_time
print(result)
```

```
0:00:00.030113
```

[4]:
```python
# use modin to read the file
start_time = datetime.now()
df_md=mp.read_csv('credit_card_transactions-ibm_v2.csv')
end_time = datetime.now()
result = end_time - start_time
print(result)
```

```
UserWarning: Ray execution environment not yet initialized. Initializing…
To remove this warning, run the following python code before doing dataframe
operations:

    import ray
    ray.init()

UserWarning: On Macs, Ray's performance is known to degrade with object store
size greater than 2.0 GiB. Ray by default does not allow setting an object store
size greater than that. Modin is overriding that default limit because it would
rather have a larger, slower object store than spill to disk more often. To
override Modin's behavior, you can initialize Ray yourself.
```

```
0:00:37.121915
```

[5]:
```python
# use ray to read the file
start_time = datetime.now()
df_ray=ray.data.read_csv('credit_card_transactions-ibm_v2.csv')
end_time = datetime.now()
result = end_time - start_time
print(result)
```

```
(raylet) Spilled 2429 MiB, 8 objects, write throughput 219 MiB/s.
Set RAY_verbose_spill_logs=0 to disable this message.
```

```
0:00:42.642238
```

Basic Validation and write the YAML file

[6]:
```python
df_pd.columns
```

[6]:
```
Index(['User', 'Card', 'Year', 'Month', 'Day', 'Time', 'Amount', 'Use Chip',
       'Merchant Name', 'Merchant City', 'Merchant State', 'Zip', 'MCC',
       'Errors?', 'Is Fraud?'],
      dtype='object')
```

[7]:
```
%%writefile file.yaml
file_type: csv
dataset_name: testfile
file_name: credit_card_transactions-ibm_v2
inbound_delimiter: ","
outbound_delimiter: "|"
skip_leading_rows: 1
columns:
    - user
    - card
    - year
    - month
    - day
    - time
    - amount
    - use_chip
    - merchant_name
    - merchant_city
    - merchant_state
    - zip
    - mcc
    - errors
    - is_fraud
```

```
Overwriting file.yaml
```

[8]:
```
%%writefile testutility.py
import logging
import os
import subprocess
import yaml
import pandas as pd
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):
    '''
    replace whitespaces in the column
    and standardized column names
    '''
    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 name and column length validation passed")
        return 1
    else:
        print("column name and column length validation failed")
        mismatched_columns_file = list(set(df.columns).difference(expected_col))
        print("Following File columns are not in the YAML␣
 ↪file",mismatched_columns_file)
        missing_YAML_file = list(set(expected_col).difference(df.columns))
        print("Following YAML 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
```

[9]:
```
config_data = util.read_config_file("file.yaml")
config_data
```

[9]:
```
{'file_type': 'csv',
 'dataset_name': 'testfile',
 'file_name': 'credit_card_transactions-ibm_v2',
 'inbound_delimiter': ',',
 'outbound_delimiter': '|',
 'skip_leading_rows': 1,
 'columns': ['user',
  'card',
  'year',
  'month',
  'day',
  'time',
  'amount',
  'use_chip',
  'merchant_name',
  'merchant_city',
  'merchant_state',
  'zip',
  'mcc',
  'errors',
  'is_fraud']}
```

[10]:
```
file_type = config_data['file_type']
source_file = "./" + config_data['file_name'] + f'.{file_type}'
df = pd.read_csv(source_file,config_data['inbound_delimiter'])
df.head()
```

```
FutureWarning: In a future version of pandas all arguments of read_csv except
for the argument 'filepath_or_buffer' will be keyword-only.
```

[10]:
```
   User  Card  Year  Month  Day   Time   Amount           Use Chip  \
0     0     0  2002      9    1  06:21  $134.09  Swipe Transaction
1     0     0  2002      9    1  06:42   $38.48  Swipe Transaction
2     0     0  2002      9    2  06:22  $120.34  Swipe Transaction
3     0     0  2002      9    2  17:45  $128.95  Swipe Transaction
4     0     0  2002      9    3  06:23  $104.71  Swipe Transaction

         Merchant Name  Merchant City Merchant State      Zip   MCC Errors?  \
0  3527213246127876953       La Verne             CA  91750.0  5300     NaN
1  -727612092139916043  Monterey Park             CA  91754.0  5411     NaN
2  -727612092139916043  Monterey Park             CA  91754.0  5411     NaN
3  3414527459579106770  Monterey Park             CA  91754.0  5651     NaN
4  5817218446178736267       La Verne             CA  91750.0  5912     NaN

  Is Fraud?
0        No
1        No
2        No
3        No
4        No
```

[11]:
```
util.col_header_val(df,config_data)
```

```
column name and column length validation passed
```

[11]:
```
1
```

[12]:
```
print("columns of files are:" ,df.columns)
print("columns of YAML are:" ,config_data['columns'])
```

```
columns of files are: Index(['user', 'card', 'year', 'month', 'day', 'time',
'amount', 'use_chip',
       'merchant_name', 'merchant_city', 'merchant_state', 'zip', 'mcc',
       'errors', 'is_fraud'],
      dtype='object')
columns of YAML are: ['user', 'card', 'year', 'month', 'day', 'time', 'amount',
'use_chip', 'merchant_name', 'merchant_city', 'merchant_state', 'zip', 'mcc',
'errors', 'is_fraud']
```

Write the file in gz format

[13]:
```
with open('credit_card_transactions-ibm_v2.csv') as fin:
    with open('OutputFile.txt', 'w', newline='') as fout:
        reader = csv.DictReader(fin, delimiter=',')
        writer = csv.DictWriter(fout, reader.fieldnames, delimiter='|')
        writer.writeheader()
        writer.writerows(reader)
```

[14]:
```
# write the file in gz format
f_in = open('OutputFile.txt','rb')
f_out = gzip.open('OutputFile.txt.gz', 'wb')
f_out.writelines(f_in)
f_out.close()
f_in.close()
```

Summary of the file

[15]:
```
if util.col_header_val(df,config_data)==0:
    print("validation failed")
else:
    print("col validation passed")
```

```
column name and column length validation passed
col validation passed
```

[16]:
```
length_of_col = len(df_pd.columns)
length_of_row = df_pd.count()[0]
file_size = os.path.getsize('credit_card_transactions-ibm_v2.csv')/
 ↪(1024*1024*1024)
```

[17]:
```python
# summarize
print("Total number of rows :", length_of_row)
print("Total number of columns :", length_of_col)
print("File Size :", round(file_size,2), "GB")
```

```
Total number of rows : 24386900
Total number of columns : 15
File Size : 2.19 GB
```