Week 6

Task Description

> Take any csv file of 2+ GB .

> Read the file methods of file reading eg: Dask, pandas and present findings in term of computational efficiency

> Perform basic validation on data columns : eg: remove special character , white spaces from the col name

> As already known the schema hence create a YAML file and write the column name in YAML file.

> Validate number of columns and column name of ingested file with YAML.

> Write the file in pipe separated text file (|) in gz format.

> Create a summary of the file:

> Total number of rows

> Total number of columns

> File size

In [ ]:
```
# Importing Libraries
```

In [ ]:
```
import csv
import re
import yaml
import time
import os
```

Creating Large CSV file

In [4]:
```
file= r'D:\EDU GERMANY\Data Glacier\WEEK 6\statements.csv\statements.csv'
os.path.getsize(file)
```

Out[4]: 2632582969

Reading file using different methods

In [6]:
```
import pandas as pd
start = time.time()
df_pandas = pd.read_csv(file)
```

```
end = time.time()
print("Read csv with pandas: ",(end-start),"sec")
```

```
Read csv with pandas:  87.97031211853027 sec
```

In [7]:
```
# Method 2: Dask
import dask.dataframe as dd
from dask import dataframe as dd
start = time.time()
df_dask = dd.read_csv(file)
end = time.time()
print("Read csv with dask: ",(end-start),"sec")
```

```
Read csv with dask:  0.06254696846008301 sec
```

Perform Basic Validation and Processing

In [9]:
```
def clean_column_names(df):
    df.columns = df.columns.str.lower()
    df.columns = df.columns.str.replace(r'[^\w]', '_', regex=True)
    df.columns = df.columns.str.strip('_')
    return df

# Clean columns for each dataframe
df_pandas = clean_column_names(df_pandas)
df_pandas= clean_column_names(df_dask)
```

Create and Read YAML Configuration

In [11]:
```
config = {
    'file_type': 'csv',
    'dataset_name': 'statements',
    'file_name': 'statements',
    'table_name': 'table',
    'inbound_delimiter': ',',
    'outbound_delimiter': '|',
    'skip_leading_rows': 0,
    'columns': ['source_item_id', 'edge_property_id', 'target_item_id']  # Add a
}

with open('config.yaml', 'w') as file:
     yaml.dump(config, file)
```

Read the YAML configuration

In [13]:
```
with open('config.yaml', 'r') as file:
    config_data = yaml.safe_load(file)
```

Validate Columns

In [15]:
```
def validate_columns(df, config):
    expected_columns = [col.lower() for col in config['columns']]
```

```
    df_columns = df.columns.tolist()
    if sorted(df_columns) == sorted(expected_columns):
        print("Validation passed")
        return True
    else:
        print("Validation failed")
        print("Expected columns:", expected_columns)
        print("DataFrame columns:", df_columns)
        return False

# Validate each dataframe
validate_columns(df_pandas, config_data)
validate_columns(df_dask, config_data)
#validate_columns(df_modin, config_data)
```

```
Validation passed
Validation passed
```

Out[15]: True

Write the File in Pipe-Separated Format and Compress

In [20]:
```
output_file = 'large_file_processed.txt.gz'
df_pandas.to_csv(output_file, sep='|', index=False, compression='gzip')
```

Create a Summary

In [22]:
```
def create_summary(df, file):
    total_rows = len(df)
    total_columns = len(df.columns)
    file_size = os.path.getsize(file)

    summary = {
        'total_rows': total_rows,
        'total_columns': total_columns,
        'file_size': file_size
    }
    return summary

summary = create_summary(df_pandas, output_file)
print(summary)
```

```
{'total_rows': 141206853, 'total_columns': 3, 'file_size': 597772282}
```

In [ ]: