Python Project for Data engineering

Thursday, February 8, 2024 4:09 PM

Code Results

from the ETL Pipeline for Multi-format Data Integration Python code basics

```
  dataframe = pd.concat([dataframe, pd.DataFrame([{"name":name, "height":height, "weight":we
ight}])], ignore_index=True)
Transformed Data
      name  height  weight
0     alex    1.67   51.25
1     ajay    1.82   61.91
2    alice    1.76   69.41
3     ravi    1.73   64.56
4      joe    1.72   65.45
5     alex    1.67   51.25
6     ajay    1.82   61.91
7    alice    1.76   69.41
8     ravi    1.73   64.56
9      joe    1.72   65.45
10    alex    1.67   51.25
11    ajay    1.82   61.91
12   alice    1.76   69.41
13    ravi    1.73   64.56
14     joe    1.72   65.45
15    jack    1.74   55.93
16     tom    1.77   64.18
17   tracy    1.78   61.90
18    john    1.72   50.97
19    jack    1.74   55.93
20     tom    1.77   64.18
21   tracy    1.78   61.90
22    john    1.72   50.97
23    jack    1.74   55.93
24     tom    1.77   64.18
25   tracy    1.78   61.90
26    john    1.72   50.97
27   simon    1.72   50.97
```

Project to extracts data from multiple sources and different file formats using Python

ETL Pipeline Documentation an Extract, Transform, Load (ETL) pipeline implemented in Python. The ETL pipeline is designed to process various types of data files (CSV, JSON, XML), extract relevant information, transform the data, and load it into a CSV file. The pipeline consists of several functions organized into logical steps: Extraction, Transformation, and Loading.

A **data engineer** extracts data from multiple sources and different file formats, transforms the extracted data to predefined settings and then loads the data to a database for further processing.

Libraries needed for data Exctration

```
import glob                          #specifically for data extraction
import pandas as pd                  #for reading csv and JSON
import xml.etree.ElementTree as ET   #parsing information from an .XML file format
from datetime import datetime        # for date and time information from the point of logging
```

[72] ✓ 0.0s Python

```
log_file = "log_file.txt"               #to store the loggs
target_file = "transformed_data.csv"    #to store the final output the will be loaded in to the database
```

[73] ✓ 0.0s Python

Components **The ETL pipeline consists of the following components:** • Extraction Functions: ○ extract_from_csv: Extracts data from CSV files using Pandas. ○ extract_from_json: Extracts data from JSON files using Pandas. ○ extract_from_xml: Extracts data from XML files using ElementTree.

```python
#CSv files
def exctract_from_csv(file_to_process):
    dataframe = pd.read_csv(file_to_process)
    return datetime
```

[74] ✓ 0.0s Python

```python
#JASON files
def extract_from_jason(file_to_process):
    dataframe = pd.read_jason(file_to_process)
    return dataframe
```

[75] ✓ 0.0s Python

```python
#XML files
def extract_from_xml(file_to_process):
    tree = ET.parse(file_to_process)                    #reading XML as a tree
    root = tree.getroot()
    dataframe = pd.DataFrame(columns=["name","height","weight"]) #creating a dataframe to store the extrac
    for i in root():
        n = i.find("name").text
        h = float(i.find("height").text)
        w = float(i.find("weight").text)
        dataframe = pd.concat([dataframe, pd.DataFrame([{"name":n,"height":h,"weight":w}])])
    return dataframe
```

[82] ✓ 0.0s Python

Function to identify which function to call based the filetype of the data file In order to start to extract

• **Extraction:** ○ The extract function iterates through all available CSV, JSON, and XML files in the current directory. ○ It utilizes specific extraction functions for each file type to extract data. Extracted data is stored in a Pandas DataFrame.

```
def extract():
    extracted_data = pd.DataFrame(columns=['name','height','weight']) # create an empty data frame to hold

    # process all csv files
    for csvfile in glob.glob("*.csv"):
        extracted_data = pd.concat([extracted_data, pd.DataFrame(extract_from_csv(csvfile))], ignore_index

    # process all json files
    for jsonfile in glob.glob("*.json"):
        extracted_data = pd.concat([extracted_data, pd.DataFrame(extract_from_json(jsonfile))], ignore_ind

    # process all xml files
    for xmlfile in glob.glob("*.xml"):
        extracted_data = pd.concat([extracted_data, pd.DataFrame(extract_from_xml(xmlfile))], ignore_index

    return extracted_data
```

[77] ✓ 0.0s Python

• **Transformation:** ○ The transform function converts height from inches to meters and weight from pounds to kilograms. The transformations are applied to the entire dataset

```
def transform(data):
    data['height'] = round(data.height * 0.0254,2)
    data['weight'] = round(data.weight * 0.45359237,2)
    return data
```

[78] ✓ 0.0s Python

• **Loading:** The load_data function saves the transformed data into a CSV file specified by the user.

```
def load_data(target_file,transformed_data):
    transformed_data.to_csv(target_file)
    return target_file
```

[79] ✓ 0.0s Python

• **Logging:** O Progress of the ETL process is logged using the log_progress function. Logs include timestamps and messages indicating the start and end of each phase (Extract, Transform, Load) and the overall ETL job.

```
def log_progress(message):
    timestamp_format = '%Y-%h-%d-%H:%M:%S' # Year-Monthname-Day-Hour-Minute-Second
    now = datetime.now() # get current timestamp
    timestamp = now.strftime(timestamp_format)
    with open(log_file,"a") as f:
        f.write(timestamp + ',' + message + '\n')
```

[80] ✓ 0.0s Python

```
# Log the initialization of the ETL process
log_progress("ETL Job Started")

# Log the beginning of the Extraction process
log_progress("Extract phase Started")
extracted_data = extract()

# Log the completion of the Extraction process
log_progress("Extract phase Ended")

# Log the beginning of the Transformation process
log_progress("Transform phase Started")
transformed_data = transform(extracted_data)
print("Transformed Data")
print(transformed_data)

# Log the completion of the Transformation process
log_progress("Transform phase Ended")

# Log the beginning of the Loading process
log_progress("Load phase Started")
load_data(target_file,transformed_data)

# Log the completion of the Loading process
log_progress("Load phase Ended")

# Log the completion of the ETL process
log_progress("ETL Job Ended")
```