Extract Transform Load (ETL) Lab

Estimated time needed: **30** minutes

Objectives

After completing this lab you will be able to:

- Read CSV and JSON file types.
- Extract data from the above file types.
- Transform data.
- Save the transformed data in a ready-to-load format which data engineers can use to load into an RDBMS.

Import the required modules and functions

[1]:
```
import glob                       # this module helps in selecting files.
import pandas as pd               # this module helps in processing CSV files
import xml.etree.ElementTree as ET  # this module helps in processing XML files.
from datetime import datetime
```

Download Files

[2]:
```
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0221EN-SkillsNetwork/labs/module%206/Lab%20-%20Extract%20Transform%20Load/data/source.z
```

```
--2021-08-14 02:55:46--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0221EN-SkillsNetwork/labs/module%206/Lab%20-%20Extract%20Transform%20Load/data/source.zip
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)... 169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)|169.63.118.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2707 (2.6K) [application/zip]
Saving to: 'source.zip'

source.zip          100%[===================>]   2.64K  --.-KB/s    in 0s

2021-08-14 02:55:46 (51.8 MB/s) - 'source.zip' saved [2707/2707]
```

Unzip Files

[3]:
```
!unzip source.zip
```

```
Archive:  source.zip
  inflating: source3.json
  inflating: source1.csv
  inflating: source2.csv
  inflating: source3.csv
  inflating: source1.json
  inflating: source2.json
  inflating: source1.xml
  inflating: source2.xml
  inflating: source3.xml
```

Set Paths

[4]:
```
tmpfile    = "temp.tmp"               # file used to store all extracted data
logfile    = "logfile.txt"            # all event logs will be stored in this file
targetfile = "transformed_data.csv"   # file where transformed data is stored
```

Extract

CSV Extract Function

[5]:
```
def extract_from_csv(file_to_process):
    dataframe = pd.read_csv(file_to_process)
    return dataframe
```

JSON Extract Function

[6]:
```
def extract_from_json(file_to_process):
    dataframe = pd.read_json(file_to_process,lines=True)
    return dataframe
```

XML Extract Function

[7]:
```
def extract_from_xml(file_to_process):
    dataframe = pd.DataFrame(columns=["name", "height", "weight"])
    tree = ET.parse(file_to_process)
```

```
    root = tree.getroot()
    for person in root:
        name = person.find("name").text
        height = float(person.find("height").text)
        weight = float(person.find("weight").text)
        dataframe = dataframe.append({"name":name, "height":height, "weight":weight}, ignore_index=True)
    return dataframe
```

Extract Function

[8]:
```
def extract():
    extracted_data = pd.DataFrame(columns=['name','height','weight']) # create an empty data frame to hold extracted data

-----
    #process all csv files
    for csvfile in glob.glob("*.csv"):
        extracted_data = extracted_data.append(extract_from_csv(csvfile), ignore_index=True)

----------
    #process all json files
    for jsonfile in glob.glob("*.json"):
        extracted_data = extracted_data.append(extract_from_json(jsonfile), ignore_index=True)

------
    #process all xml files
    for xmlfile in glob.glob("*.xml"):
        extracted_data = extracted_data.append(extract_from_xml(xmlfile), ignore_index=True)

----------
    return extracted_data
```

Transform

The transform function does the following tasks.

1. Convert height which is in inches to millimeter
2. Convert weight which is in pounds to kilograms

[9]:
```
def transform(data):
        #Convert height which is in inches to millimeter
        #Convert the datatype of the column into float
        #data.height = data.height.astype(float)
        #Convert inches to meters and round off to two decimals(one inch is 0.0254 meters)
        data['height'] = round(data.height * 0.0254,2)

----------
        #Convert weight which is in pounds to kilograms
        #Convert the datatype of the column into float
        #data.weight = data.weight.astype(float)
        #Convert pounds to kilograms and round off to two decimals(one pound is 0.45359237 kilograms)
        data['weight'] = round(data.weight * 0.45359237,2)
        return data
```

Loading

[10]:
```
def load(targetfile,data_to_load):
    data_to_load.to_csv(targetfile)
```

Logging

[11]:
```
def log(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("logfile.txt","a") as f:
        f.write(timestamp + ',' + message + '\n')
```

Running ETL Process

[12]:
```
log("ETL Job Started")
```

[13]:
```
log("Extract phase Started")
extracted_data = extract()
log("Extract phase Ended")
extracted_data
```

[13]:

	name	height	weight
0	alex	65.78	112.99
1	ajay	71.52	136.49
2	alice	69.40	153.03
3	ravi	68.22	142.34
4	joe	67.79	144.30
5	alex	65.78	112.99
6	ajay	71.52	136.49
7	alice	69.40	153.03
8	ravi	68.22	142.34
9	joe	67.79	144.30
10	alex	65.78	112.99
11	ajay	71.52	136.49
12	alice	69.40	153.03
13	ravi	68.22	142.34
14	joe	67.79	144.30
15	jack	68.70	123.30
16	tom	69.80	141.49
17	tracy	70.01	136.46
18	john	67.90	112.37
19	jack	68.70	123.30
20	tom	69.80	141.49
21	tracy	70.01	136.46
22	john	67.90	112.37
23	jack	68.70	123.30

24	tom	69.80	141.49
25	tracy	70.01	136.46
26	john	67.90	112.37
27	simon	67.90	112.37
28	jacob	66.78	120.67
29	cindy	66.49	127.45
30	ivan	67.62	114.14
31	simon	67.90	112.37
32	jacob	66.78	120.67
33	cindy	66.49	127.45
34	ivan	67.62	114.14
35	simon	67.90	112.37
36	jacob	66.78	120.67
37	cindy	66.49	127.45
38	ivan	67.62	114.14

[14]:
```
log("Transform phase Started")
transformed_data = transform(extracted_data)
log("Transform phase Ended")
transformed_data
```

[14]:

	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
28	jacob	1.70	54.73
29	cindy	1.69	57.81
30	ivan	1.72	51.77
31	simon	1.72	50.97
32	jacob	1.70	54.73
33	cindy	1.69	57.81
34	ivan	1.72	51.77
35	simon	1.72	50.97
36	jacob	1.70	54.73
37	cindy	1.69	57.81
38	ivan	1.72	51.77

[15]:
```
log("Load phase Started")
load(targetfile,transformed_data)
log("Load phase Ended")
```

[16]:
```
log("ETL Job Ended")
```

Exercise

Using the example above complete the exercise below.

Download Files

[17]:
```
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0221EN-SkillsNetwork/labs/module%206/Lab%20-%20Extract%20Transform%20Load/data/datasour
```

```
--2021-08-14 02:56:02--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0221EN-SkillsNetwork/labs/module%206/Lab%20-%20Extract%20Transform%20Load/data/datasource.zip
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)... 169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)|169.63.118.104|:443... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 4249 (4.1K) [application/zip]
Saving to: ‘datasource.zip’

datasource.zip      100%[===================>]   4.15K  --.-KB/s    in 0s

2021-08-14 02:56:02 (97.2 MB/s) - ‘datasource.zip’ saved [4249/4249]
```

Unzip Files

[18]:
```
!unzip datasource.zip -d dealership_data
```

```
Archive:  datasource.zip
  inflating: dealership_data/used_car_prices1.csv
  inflating: dealership_data/used_car_prices2.csv
  inflating: dealership_data/used_car_prices3.csv
  inflating: dealership_data/used_car_prices1.json
  inflating: dealership_data/used_car_prices2.json
  inflating: dealership_data/used_car_prices3.json
  inflating: dealership_data/used_car_prices1.xml
  inflating: dealership_data/used_car_prices2.xml
  inflating: dealership_data/used_car_prices3.xml
```

About the Data

Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in. Learn more here.

The file `dealership_data` contains CSV, JSON, and XML files for used car data which contain features named `car_model`, `year_of_manufacture`, `price`, and `fuel`.

Set Paths

[19]:
```
tmpfile    = "dealership_temp.tmp"               # file used to store all extracted data
logfile    = "dealership_logfile.txt"            # all event logs will be stored in this file
targetfile = "dealership_transformed_data.csv"   # file where transformed data is stored
```

Extract

Question 1: CSV Extract Function

[21]:
```
# Add the CSV extract function below
def extract_from_csv(file_to_process):
    dataframe = pd.read_csv(file_to_process)
    return dataframe
```

▼Click here for the solution

``` def extract_from_csv(file_to_process): dataframe = pd.read_csv(file_to_process) return dataframe ```

Question 2: JSON Extract Function

[22]:
```
# Add the JSON extract function below
def extract_from_json(file_to_process):
    dataframe = pd.read_json(file_to_process,lines=True)
    return dataframe
```

▼Click here for the solution

``` def extract_from_json(file_to_process): dataframe = pd.read_json(file_to_process,lines=True) return dataframe ```

Question 3: XML Extract Function

[23]:
```
# Add the XML extract function below, it is the same as the xml extract function above but the column names need to be renamed.
def extract_from_xml(file_to_process):
    dataframe = pd.DataFrame(columns=['car_model','year_of_manufacture','price', 'fuel'])
    tree = ET.parse(file_to_process)
    root = tree.getroot()
    for person in root:
        car_model = person.find("car_model").text
        year_of_manufacture = int(person.find("year_of_manufacture").text)
        price = float(person.find("price").text)
        fuel = person.find("fuel").text
        dataframe = dataframe.append({"car_model":car_model, "year_of_manufacture":year_of_manufacture, "price":price, "fuel":fuel}, ignore_index=True)
    return dataframe
```

▼Click here for the solution

``` def extract_from_xml(file_to_process): dataframe = pd.DataFrame(columns=['car_model','year_of_manufacture','price', 'fuel']) tree = ET.parse(file_to_process) root = tree.getroot() for person in root: car_model = person.find("car_model").text year_of_manufacture = int(person.find("year_of_manufacture").text) price = float(person.find("price").text) fuel = person.find("fuel").text dataframe = dataframe.append({"car_model":car_model, "year_of_manufacture":year_of_manufacture, "price":price, "fuel":fuel}, ignore_index=True) return dataframe ```

Question 4: Extract Function

Call the specific extract functions you created above by replacing the `ADD_FUNCTION_CALL` with the proper function call.

[24]:
```
def extract():
    extracted_data = pd.DataFrame(columns=['car_model','year_of_manufacture','price', 'fuel']) # create an empty data frame to hold extracted data
------
    #process all csv files
    for csvfile in glob.glob("dealership_data/*.csv"):
        extracted_data = extracted_data.append(extract_from_csv(csvfile), ignore_index=True)
----------
    #process all json files
    for jsonfile in glob.glob("dealership_data/*.json"):
        extracted_data = extracted_data.append(extract_from_json(jsonfile), ignore_index=True)
------
    #process all xml files
    for xmlfile in glob.glob("dealership_data/*.xml"):
        extracted_data = extracted_data.append(extract_from_xml(xmlfile), ignore_index=True)
----------
    return extracted_data
```

▼Click here for the solution

``` def extract(): extracted_data = pd.DataFrame(columns=['car_model','year_of_manufacture','price', 'fuel']) # create an empty data frame to hold extracted data #process all csv files for csvfile in glob.glob("dealership_data/*.csv"): extracted_data = extracted_data.append(extract_from_csv(csvfile), ignore_index=True) #process all json files for jsonfile in glob.glob("dealership_data/*.json"): extracted_data = extracted_data.append(extract_from_json(jsonfile), ignore_index=True) #process all xml files for xmlfile in glob.glob("dealership_data/*.xml"): extracted_data = extracted_data.append(extract_from_xml(xmlfile), ignore_index=True) return extracted_data ```

Transform

Question 5: Transform

Round the `price` columns to 2 decimal places

[25]:
```
# Add the transform function below
def transform(data):
    data['price'] = round(data.price, 2)
    return data
```

▼Click here for the solution
```

def transform(data): data['price'] = round(data.price, 2) return data

</details>
```

Loading

Question 6: Load

[26]:
```
# Add the load function below
def load(targetfile,data_to_load):
    data_to_load.to_csv(targetfile)
```

▼Click here for the solution
```

def load(targetfile,data_to_load): data_to_load.to_csv(targetfile)

</details>
```

Logging

Question 7: Log

Make sure to change the name of the logfile to the one specified in the set paths section. Change the timestamp order to Hour-Minute-Second-Monthname-Day-Year.

[27]:
```
# Add the log function below
def log(message):
    timestamp_format = '%H:%M:%S-%h-%d-%Y' #Hour-Minute-Second-MonthName-Day-Year now = datetime.now()
    # get current timestamp timestamp = now.strftime(timestamp_format) with open("dealership_logfile.txt","a") as f: f.write(timestamp + ',' + message + '\n')
```

▼Click here for the solution
```

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

</details>
```

Running ETL Process

Question 8: ETL Process

Run all functions to extract, transform, and load the data. Make sure to log all events using the `log` function. Place your code under each comment.

[29]:
```
# Log that you have started the ETL process
log("ETL Job Started")

# Log that you have started the Extract step
log("Extract phase Started")

# Call the Extract function
extracted_data = extract()

# Log that you have completed the Extract step
log("Extract phase Ended")

# Log that you have started the Transform step
log("Transform phase Started")

# Call the Transform function
transformed_data = transform(extracted_data)

# Log that you have completed the Transform step
log("Transform phase Ended")

# Log that you have started the Load step
log("Load phase Started")

# Call the Load function
load(targetfile,transformed_data)

# Log that you have completed the Load step
log("Load phase Ended")

# Log that you have completed the ETL process
log("ETL Job Ended")
```

▼Click here for the solution
```

log("ETL Job Started")

log("Extract phase Started") extracted_data = extract() log("Extract phase Ended")

log("Transform phase Started") transformed_data = transform(extracted_data) log("Transform phase Ended")

log("Load phase Started") load(targetfile,transformed_data) log("Load phase Ended")

log("ETL Job Ended")

</details>
```

Authors

Ramesh Sannareddy

Joseph Santarcangelo

Azim Hirjani

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-11-25	0.1	Ramesh Sannareddy	Created initial version of the lab