Project_1B_ Project_Template

March 24, 2023

1 Part I. ETL Pipeline for Pre-Processing the Files

1.1 PLEASE RUN THE FOLLOWING CODE FOR PRE-PROCESSING THE FILES

Import Python packages

```
In [69]: # Import Python packages
         import pandas as pd
         import cassandra
         import re
         import os
         import glob
         import numpy as np
         import json
         import csv
```

Creating list of filepaths to process original event csv data files

```
In [70]: # checking your current working directory
         print(os.getcwd())

         # Get your current folder and subfolder event data
         filepath = os.getcwd() + '/event_data'

         # Create a for loop to create a list of files and collect each filepath
         for root, dirs, files in os.walk(filepath):

         # join the file path and roots with the subdirectories using glob
             file_path_list = glob.glob(os.path.join(root,'*'))
             print(file_path_list)
```

```
/home/workspace
['/home/workspace/event_data/2018-11-27-events.csv', '/home/workspace/event_data/2018-11-04-even
```

Processing the files to create the data file csv that will be used for Apache Casssandra tables

In [71]:
```
# initiating an empty list of rows that will be generated from each file
full_data_rows_list = []

# for every filepath in the file path list
for f in file_path_list:

# reading csv file
    with open(f, 'r', encoding = 'utf8', newline='') as csvfile:
        # creating a csv reader object
        csvreader = csv.reader(csvfile)
        next(csvreader)

 # extracting each data row one by one and append it
        for line in csvreader:
            #print(line)
            full_data_rows_list.append(line)

# uncomment the code below if you would like to get total number of rows
print(len(full_data_rows_list))
# uncomment the code below if you would like to check to see what the list of event dat
print(full_data_rows_list)

# creating a smaller event data csv file called event_datafile_full csv that will be us
# Apache Cassandra tables
csv.register_dialect('myDialect', quoting=csv.QUOTE_ALL, skipinitialspace=True)

with open('event_datafile_new.csv', 'w', encoding = 'utf8', newline='') as f:
    writer = csv.writer(f, dialect='myDialect')
    writer.writerow(['artist','firstName','gender','itemInSession','lastName','length',
                'level','location','sessionId','song','userId'])
    for row in full_data_rows_list:
        if (row[0] == ''):
            continue
        writer.writerow((row[0], row[2], row[3], row[4], row[5], row[6], row[7], row[8]
```

In [ ]:
```
# check the number of rows in your csv file
with open('event_datafile_new.csv', 'r', encoding = 'utf8') as f:
    print(sum(1 for line in f))
```

2 Part II. Complete the Apache Cassandra coding portion of your project.

2.1 Now you are ready to work with the CSV file titled event_datafile_new.csv, located within the Workspace directory. The event_datafile_new.csv contains the following columns:

- artist

- firstName of user
- gender of user
- item number in session
- last name of user
- length of the song
- level (paid or free song)
- location of the user
- sessionId
- song title
- userId

The image below is a screenshot of what the denormalized data should appear like in the **event_datafile_new.csv** after the code above is run:

2.2 Begin writing your Apache Cassandra code in the cells below

Creating a Cluster

```
In [ ]: # This should make a connection to a Cassandra instance your local machine
        # (127.0.0.1)

        from cassandra.cluster import Cluster
        cluster = Cluster(['127.0.0.1'])

        # To establish connection and begin executing queries, need a session
        session = cluster.connect()
```

Create Keyspace

```
In [ ]: try:
            session.execute("""
            CREATE KEYSPACE IF NOT EXISTS my_key
            WITH REPLICATION =
            { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }"""
            )
            session.set_keyspace('my_key')
        except Exception as e:
            print(e)
```

Set Keyspace

```
In [ ]: session.set_keyspace("my_key")
```

2.2.1 Now we need to create tables to run the following queries. Remember, with Apache Cassandra you model the database tables on the queries you want to run.

2.3 Create queries to ask the following three questions of the data

2.3.1 1. Give me the artist, song title and song's length in the music app history that was heard during sessionId = 338, and itemInSession = 4

2.3.2 2. Give me only the following: name of artist, song (sorted by itemInSession) and user (first and last name) for userid = 10, sessionid = 182

2.3.3 3. Give me every user name (first and last) in my music app history who listened to the song 'All Hands Against His Own'

```
In [ ]: try:
            session.execute("""
                CREATE TABLE IF NOT EXISTS Song101 (
                    sessionId INT,
                    itemInSession INT,
                    artist TEXT,
                    song TEXT,
                    length FLOAT,
                    PRIMARY KEY (sessionId, itemInSession));
            """)
        except Exception as e:
            print(e)
```

```
In [ ]: file = 'event_datafile_new.csv'

        with open(file, encoding = 'utf8') as f:
            csvreader = csv.reader(f)
            next(csvreader)

            for line in csvreader:
                query = "INSERT INTO Song101 (sessionId, itemInSession, artist, song, length)"
                query = query + " VALUES (%s, %s, %s, %s, %s);"
                session.execute(query, (int(line[8]), int(line[3]), line[0], line[9], float(line
```

Do a SELECT to verify that the data have been inserted into each table

```
In [ ]: ## TO-DO: Add in the SELECT statement to verify the data was ensession.row_factory = pan


        query = "SELECT * FROM Song101 WHERE sessionId = 338 AND itemInSession = 4"
        rows = session.execute(query)
        for row in rows:
            print(row.artist, row.song, row.length, row.sessionid, row.iteminsession)
```

2.3.4 COPY AND REPEAT THE ABOVE THREE CELLS FOR EACH OF THE THREE QUESTIONS

```
In [ ]: try:
            session.execute("""
                CREATE TABLE IF NOT EXISTS Song102 (
                    userId INT,
                    sessionId INT,
                    itemInSession INT,
                    artist TEXT,
                    song TEXT,
                    firstName TEXT,
                    lastName TEXT,
                    PRIMARY KEY ((userId, sessionId), itemInSession));
            """)
        except Exception as e:
            print(e)
```

```
In [ ]: file = 'event_datafile_new.csv'

        with open(file, encoding = 'utf8') as f:
            csvreader = csv.reader(f)
            next(csvreader)

            for line in csvreader:
                query = "INSERT INTO Song102 (userId, sessionId, itemInSession, artist, song, fi
                query = query + " VALUES (%s, %s, %s, %s, %s, %s, %s);"
                session.execute(query, (int(line[10]), int(line[8]), int(line[3]), line[0], line
```

```
In [ ]: query = "SELECT * FROM Song102 WHERE userId = 10 AND sessionId = 182"
        rows = session.execute(query)
        for row in rows:
            print(row.artist, row.song, row.firstname, row.lastname, row.iteminsession)
```

```
In [ ]: try:
            session.execute("""
                CREATE TABLE IF NOT EXISTS Song103 (
                    song TEXT,
                    firstName TEXT,
                    lastName TEXT,
                    userId INT,
                    PRIMARY KEY (song, firstName, lastName, userId));
            """)
        except Exception as e:
            print(e)
```

```
In [ ]: file = 'event_datafile_new.csv'
```

```
        with open(file, encoding = 'utf8') as f:
            csvreader = csv.reader(f)
            next(csvreader)

            for line in csvreader:
                query = "INSERT INTO Song103 (song, firstName, lastName, userId)"
                query = query + " VALUES (%s, %s, %s, %s);"
                session.execute(query, (line[9], line[1], line[4], int(line[10])))
```

```
In [ ]: query = "SELECT firstName, lastName FROM Song103 WHERE song = 'All Hands Against His Own
        rows = session.execute(query)
        for row in rows:
            print(row.firstname, row.lastname)
```

2.3.5 Drop the tables before closing out the sessions

```
In [ ]: ## TO-DO: Drop the table before closing out the sessions

        try:
            session.execute("DROP TABLE Song101")
            session.execute("DROP TABLE Song102")
            session.execute("DROP TABLE Song103")
        except Exception as e:
            print(e)
```

2.3.6 Close the session and cluster connectionű

```
In [ ]: session.shutdown()
        cluster.shutdown()
```