Project_1B_ Project_Template

January 8, 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 [111]: # 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 [112]: # checking your current working directory
          print(os.getcwd())

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

          # 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
```

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

In [113]:
```
# 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 da
#print(full_data_rows_list)

# creating a smaller event data csv file called event_datafile_full csv that will be u
# 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 [114]:
```
# 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))
```

```
6821
```

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

          from cassandra.cluster import Cluster
          try:
              cluster = Cluster(['127.0.0.1'])
              # To establish connection and begin executing queries, need a session
              session = cluster.connect()
          except Exception as e:
              print(e)
```

Create Keyspace

```
In [116]: # TO-DO: Create a Keyspace
          try:
              session.execute("""
              CREATE KEYSPACE IF NOT EXISTS udacity
              WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication_factor' : 1}
              """)
          except Exception as e:
              print(e)
```

Set Keyspace

```
In [117]: # TO-DO: Set KEYSPACE to the keyspace specified above
          try:
              session.set_keyspace('udacity')
          except Exception as e:
              print(e)
```

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 [118]: ## TO-DO: Query 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
          try:
              session.execute("CREATE TABLE IF NOT EXISTS playtime_history(artist text, firstNam
                                              gender text, itemInSession int, \
                                              lastName text, length float, level text, \
                                              location text, sessionId int, song text, \
                                              userId int, PRIMARY KEY((sessionId), itemInSession,
                              )
          except Exception as e:
              print(e)
```

```
In [119]: # We have provided part of the code to set up the CSV file. Please complete the Apache
          file = 'event_datafile_new.csv'

          with open(file, encoding = 'utf8') as f:
              csvreader = csv.reader(f)
              next(csvreader) # skip header
              for line in csvreader:
          ## TO-DO: Assign the INSERT statements into the `query` variable
                  try:
                      query = "INSERT INTO playtime_history(artist, firstName, gender, itemInSes
                      query = query + "VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
                      ## TO-DO: Assign which column element should be assigned for each column i
                      ## For e.g., to INSERT artist_name and user first_name, you would change t
                      session.execute(query, (line[0], line[1], line[2], int(line[3]), line[4],
                  except Exception as e:
                      print(e)
```

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

```
In [120]: ## TO-DO: Add in the SELECT statement to verify the data was entered into the table
          try:
              rows = session.execute("SELECT artist, song, length FROM playtime_history WHERE se
              for row in rows:
                  print(row.artist, row.song, row.length)
          except Exception as e:
              print(e)
```

```
Faithless Music Matters (Mark Knight Dub) 495.30731201171875
```

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

```
In [121]: ## TO-DO: Query 2: Give me only the following: name of artist, song (sorted by itemInS
          ## for userid = 10, sessionid = 182
          try:
              session.execute("CREATE TABLE IF NOT EXISTS session_history(artist text, firstName
                                              gender text, itemInSession int, \
                                              lastName text, length float, level text, \
                                              location text, sessionId int, song text, \
                                              userId int, PRIMARY KEY((userId, sessionId), itemInS
                              )
          except Exception as e:
              print(e)
```

```
In [122]: file = 'event_datafile_new.csv'
          with open(file, encoding = 'utf8') as f:
              csvreader = csv.reader(f)
              next(csvreader) # skip header
              for line in csvreader:
                  ## TO-DO: Assign the INSERT statements into the `query` variable
                  try:
                      query = "INSERT INTO session_history(artist, firstName, gender, itemInSess
                      query = query + "VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
                      ## TO-DO: Assign which column element should be assigned for each column i
                      ## For e.g., to INSERT artist_name and user first_name, you would change t
                      session.execute(query, (line[0], line[1], line[2], int(line[3]), line[4],
                  except Exception as e:
                      print(e)
```

```
In [123]: try:
              rows = session.execute("SELECT artist, song, firstName, lastName FROM session_hist
              for row in rows:
                  print(row.artist, row.song, row.firstname, row.lastname)
          except Exception as e:
              print(e)
```

```
Down To The Bone Keep On Keepin' On Sylvie Cruz
Three Drives Greece 2000 Sylvie Cruz
Sebastien Tellier Kilometer Sylvie Cruz
Lonnie Gordon Catch You Baby (Steve Pitron & Max Sanna Radio Edit) Sylvie Cruz
```

```
In [124]: ## TO-DO: Query 3: Give me every user name (first and last) in my music app history wh
          try:
              session.execute("CREATE TABLE IF NOT EXISTS song_viewers(artist text, firstName te
                                              gender text, itemInSession int, \
                                              lastName text, length float, level text, \
                                              location text, sessionId int, song text, \
                                              userId int, PRIMARY KEY((song), userId, sessionId, i
                              )
          except Exception as e:
              print(e)
```

```
In [125]: file = 'event_datafile_new.csv'
          with open(file, encoding = 'utf8') as f:
              csvreader = csv.reader(f)
              next(csvreader) # skip header
              for line in csvreader:
                  ## TO-DO: Assign the INSERT statements into the `query` variable
                  try:
                      query = "INSERT INTO song_viewers(artist, firstName, gender, itemInSession
                      query = query + "VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
                      ## TO-DO: Assign which column element should be assigned for each column i
                      ## For e.g., to INSERT artist_name and user first_name, you would change t
                      session.execute(query, (line[0], line[1], line[2], int(line[3]), line[4],
                  except Exception as e:
                      print(e)
```

```
In [126]: try:
              rows = session.execute("SELECT firstName, lastName FROM song_viewers WHERE song =
              for row in rows:
                  print(row.firstname, row.lastname)
          except Exception as e:
              print(e)
```

```
Jacqueline Lynch
Tegan Levine
Sara Johnson
```

2.3.5 Drop the tables before closing out the sessions

```
In [4]: ## TO-DO: Drop the table before closing out the sessions
```

```
In [127]: try:
              session.execute("DROP TABLE IF EXISTS playtime_history")
```

```
except Exception as e:
    print(e)

try:
    session.execute("DROP TABLE IF EXISTS session_history")
except Exception as e:
    print(e)

try:
    session.execute("DROP TABLE IF EXISTS song_viewers")
except Exception as e:
    print(e)
```

2.3.6 Close the session and cluster connectionű

In [128]:
```
session.shutdown()
cluster.shutdown()
```

In [ ]:

In [ ]: