etl

October 16, 2022

1 ETL Processes

Use this notebook to develop the ETL process for each of your tables before completing the `etl.py` file to load the whole datasets.

```
In [25]: import os
         import glob
         import psycopg2
         import pandas as pd
         from sql_queries import *
         from psycopg2 import Error
```

```
In [26]: try:
             conn = psycopg2.connect("host=127.0.0.1 dbname=sparkifydb user=student password=stu
             cur = conn.cursor()
             print('done connecting')
         except Error as e:
             print("Error while connecting to Postgress", e)
```

```
done connecting
```

```
In [27]: def get_files(filepath):
             # rcount = 0
             # while rcount < 1:
             all_files = []
             for root, dirs, files in os.walk(filepath):
                 files = glob.glob(os.path.join(root,'*.json'))
                 for f in files:
                     all_files.append(os.path.abspath(f))
             return all_files
         #below code I was experimenting on having one sourcefile and running it instead of
         # using notebook. I was abit confortable working with one source file.
         #             for row in df:
         # #                 print('fgg')
         #                 num_songs  = df['num_songs']
         #                 artist_id = df['artist_id']
         #                 artist_latitude=df['artist_latitude']
```

```
#                 artist_longitude=df['artist_longitude']
#                 artist_name=df['artist_name']
#                 song_id = df['song_id']
#                 title = df['title']
#                 duration=df['duration']
#                 year=df['year']
#                 # print(song_id)
#                 song_data =  [song_id,title,artist_id,year,duration]
#                 # song_data = [('SOQOTLQ12AB01868D0', 'Clementina Santafè', 'ARGCY1Y1
# #                   print(song_data)
#                 try:
#                     cur.execute("INSERT INTO songs (song_id,title,artist_id,year,dura

#                 except Error as e:
#                     print('failed coz of ifala fulani',song_id)
#             conn.commit()
#                 # rows.append(song_data)
```

2 Process `song_data`

In this first part, you'll perform ETL on the first dataset, `song_data`, to create the `songs` and `artists` dimensional tables.

Let's perform ETL on a single song file and load a single record into each table to start. - Use the `get_files` function provided above to get a list of all song JSON files in `data/song_data` - Select the first song in this list - Read the song file and view the data

```
In [28]: song_files = get_files('data/song_data')

In [29]: filepath = song_files[0]

In [30]: df = pd.read_json(filepath, lines=True)
         df.head()
```

Out[30]:

	artist_id	artist_latitude	artist_location	artist_longitude	artist_name	duration	num_songs	song_id	title	year
0	ARD7TVE1187B99BFB1	NaN	California - LA	NaN	Casual	218.93179	1	SOMZWCG12A8C13C480	I Didn't Mean To	0

2.1 #1: `songs` Table

Extract Data for Songs Table

- Select columns for song ID, title, artist ID, year, and duration

- Use `df.values` to select just the values from the dataframe
- Index to select the first (only) record in the dataframe
- Convert the array to a list and set it to `song_data`

```
In [31]: song_data = df[['song_id','title','artist_id','year','duration']].values.tolist()
         song_data = [item for sublist in song_data for item in sublist]
         song_data

Out[31]: ['SOMZWCG12A8C13C480', "I Didn't Mean To", 'ARD7TVE1187B99BFB1', 0, 218.93179]
```

Insert Record into Song Table Implement the `song_table_insert` query in `sql_queries.py` and run the cell below to insert a record for this song into the `songs` table. Remember to run `create_tables.py` before running the cell below to ensure you've created/resetted the `songs` table in the sparkify database.

```
In [32]: try:
             cur.execute(song_table_insert, song_data)
         except Error as e:
             print("Error while inserting to songs table", e)
         conn.commit()
```

Run `test.ipynb` to see if you've successfully added a record to this table.

2.2 #2: `artists` Table

Extract Data for Artists Table

- Select columns for artist ID, name, location, latitude, and longitude
- Use `df.values` to select just the values from the dataframe
- Index to select the first (only) record in the dataframe
- Convert the array to a list and set it to `artist_data`

```
In [33]: artist_data = df[['artist_id','artist_name','artist_location','artist_latitude','artist
         artist_data = [item for sublist in artist_data for item in sublist]
         artist_data

Out[33]: ['ARD7TVE1187B99BFB1', 'Casual', 'California - LA', nan, nan]
```

Insert Record into Artist Table Implement the `artist_table_insert` query in `sql_queries.py` and run the cell below to insert a record for this song's artist into the `artists` table. Remember to run `create_tables.py` before running the cell below to ensure you've created/resetted the `artists` table in the sparkify database.

```
In [34]: try:
             cur.execute(artist_table_insert, artist_data)
         except Error as e:
             print("Error while inserting to artist table", e)
         conn.commit()
```

Run `test.ipynb` to see if you've successfully added a record to this table.

3 Process `log_data`

In this part, you'll perform ETL on the second dataset, `log_data`, to create the `time` and `users` dimensional tables, as well as the `songplays` fact table.

Let's perform ETL on a single log file and load a single record into each table. - Use the `get_files` function provided above to get a list of all log JSON files in `data/log_data` - Select the first log file in this list - Read the log file and view the data

```
In [35]: log_files = get_files("data/log_data")

In [36]: filepath = log_files[0]

In [37]: df = pd.read_json(filepath, lines=True)
         df.head()
```

Out[37]:

	artist	auth	firstName	gender	itemInSession	lastName	length	level	location	method	page	registration	sessionId	song	status	ts	userAgent	userId
0	Stephen Lynch	Logged In	Jayden	M	0	Bell	182.85669	free	Dallas-Fort Worth-Arlington, TX	PUT	NextSong	1.540992e+12	829	Jim Henson's Dead	200	1543537327796	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT...	91
1	Manowar	Logged In	Jacob	M	0	Klein	247.56200	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Shell Shock	200	1543540121796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
2	Morcheeba	Logged In	Jacob	M	1	Klein	257.41016	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Women Lose Weight (Feat: Slick Rick)	200	1543540368796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
3	Maroon 5	Logged In	Jacob	M	2	Klein	231.23546	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Won't Go Home Without You	200	1543540625796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
4	Train	Logged In	Jacob	M	3	Klein	216.76363	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Hey_ Soul Sister	200	1543540856796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73

3.1 #3: `time` Table

Extract Data for Time Table

- Filter records by `NextSong` action
- Convert the `ts` timestamp column to datetime

- Hint: the current timestamp is in milliseconds
- Extract the timestamp, hour, day, week of year, month, year, and weekday from the `ts` column and set `time_data` to a list containing these values in order
- Hint: use pandas' `dt` attribute to access easily datetimelike properties.
- Specify labels for these columns and set to `column_labels`
- Create a dataframe, `time_df`, containing the time data for this file by combining `column_labels` and `time_data` into a dictionary and converting this into a dataframe

```
In [38]: df = df[df['page']=='NextSong']
         df.head()
```

Out[38]:

	artist	auth	firstName	gender	itemInSession	lastName	length	level	location	method	page	registration	sessionId	song	status	ts	userAgent	userId
0	Stephen Lynch	Logged In	Jayden	M	0	Bell	182.85669	free	Dallas-Fort Worth-Arlington, TX	PUT	NextSong	1.540992e+12	829	Jim Henson's Dead	200	1543537327796	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT...	91
1	Manowar	Logged In	Jacob	M	0	Klein	247.56200	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Shell Shock	200	1543540121796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
2	Morcheeba	Logged In	Jacob	M	1	Klein	257.41016	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Women Lose Weight (Feat: Slick Rick)	200	1543540368796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
3	Maroon 5	Logged In	Jacob	M	2	Klein	231.23546	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Won't Go Home Without You	200	1543540625796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
4	Train	Logged In	Jacob	M	3	Klein	216.76363	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Hey_ Soul Sister	200	1543540856796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73

```
In [39]: t = df.copy()
         t['ts'] = pd.to_datetime(t['ts'],unit='ms')
         t.head()
```

Out[39]:

	artist	auth	firstName	gender	itemInSession	lastName	length	level	location	method	page	registration	sessionId	song	status	ts	userAgent	userId
0	Stephen Lynch	Logged In	Jayden	M	0	Bell	182.85669	free	Dallas-Fort Worth-Arlington, TX	PUT	NextSong	1.540992e+12	829	Jim Henson's Dead	200	1543537327796	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT...	91
1	Manowar	Logged In	Jacob	M	0	Klein	247.56200	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Shell Shock	200	1543540121796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
2	Morcheeba	Logged In	Jacob	M	1	Klein	257.41016	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Women Lose Weight (Feat: Slick Rick)	200	1543540368796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
3	Maroon 5	Logged In	Jacob	M	2	Klein	231.23546	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Won't Go Home Without You	200	1543540625796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73
4	Train	Logged In	Jacob	M	3	Klein	216.76363	paid	Tampa-St. Petersburg-Clearwater, FL	PUT	NextSong	1.540558e+12	1049	Hey_ Soul Sister	200	1543540856796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	73

```
      length level                              location method      page  \
0  182.85669  free      Dallas-Fort Worth-Arlington, TX    PUT  NextSong
1  247.56200  paid  Tampa-St. Petersburg-Clearwater, FL    PUT  NextSong
2  257.41016  paid  Tampa-St. Petersburg-Clearwater, FL    PUT  NextSong
3  231.23546  paid  Tampa-St. Petersburg-Clearwater, FL    PUT  NextSong
4  216.76363  paid  Tampa-St. Petersburg-Clearwater, FL    PUT  NextSong

   registration  sessionId                                     song  status  \
0  1.540992e+12        829                      Jim Henson's Dead     200
1  1.540558e+12       1049                            Shell Shock     200
2  1.540558e+12       1049  Women Lose Weight (Feat: Slick Rick)     200
3  1.540558e+12       1049              Won't Go Home Without You     200
4  1.540558e+12       1049                       Hey_ Soul Sister     200

                        ts                                          userAgent  \
0 2018-11-30 00:22:07.796  Mozilla/5.0 (compatible; MSIE 10.0; Windows NT...
1 2018-11-30 01:08:41.796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...
2 2018-11-30 01:12:48.796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...
3 2018-11-30 01:17:05.796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...
4 2018-11-30 01:20:56.796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...

  userId
0     91
1     73
2     73
3     73
4     73
```

```
In [40]: time_data = [df['ts'],t['ts'].dt.hour, t['ts'].dt.day,t['ts'].dt.weekofyear,t['ts'].dt.
         column_labels = ['ts','hour','day','week of year','month','year','weekday']

In [41]: time_df = dict(zip(column_labels, time_data))
         time_df = pd.DataFrame.from_dict(time_df)
         time_df.head()
```

Out[41]:

	ts	hour	day	week of year	month	year	weekday
0	1543537327796	0	30	48	11	2018	4
1	1543540121796	1	30	48	11	2018	4
2	1543540368796	1	30	48	11	2018	4
3	1543540625796	1	30	48	11	2018	4
4	1543540856796	1	30	48	11	2018	4

Insert Records into Time Table Implement the `time_table_insert` query in `sql_queries.py` and run the cell below to insert records for the timestamps in this log file into the `time` table. Remember to run `create_tables.py` before running the cell below to ensure you've created/resetted the `time` table in the sparkify database.

```
In [42]: for i, row in time_df.iterrows():
             try:
                 cur.execute(time_table_insert, list(row))
             except Error as e:
                 print("Error while inserting to time table", e)
             conn.commit()
```

Run `test.ipynb` to see if you've successfully added records to this table.

3.2 #4: `users` Table

Extract Data for Users Table

- Select columns for user ID, first name, last name, gender and level and set to `user_df`

```
In [43]: user_df = df[['userId','firstName','lastName','gender','level']]
```

Insert Records into Users Table Implement the `user_table_insert` query in `sql_queries.py` and run the cell below to insert records for the users in this log file into the `users` table. Remember to run `create_tables.py` before running the cell below to ensure you've created/resetted the `users` table in the sparkify database.

```
In [44]: for i, row in user_df.iterrows():
             try:
                 cur.execute(user_table_insert, row)
             except Error as e:
                 print("Error while inserting to user table", e)
             conn.commit()
```

Run `test.ipynb` to see if you've successfully added records to this table.

3.3 #5: `songplays` Table

Extract Data and Songplays Table This one is a little more complicated since information from the songs table, artists table, and original log file are all needed for the `songplays` table. Since the log file does not specify an ID for either the song or the artist, you'll need to get the song ID and artist ID by querying the songs and artists tables to find matches based on song title, artist name, and song duration time. - Implement the `song_select` query in `sql_queries.py` to find the song ID and artist ID based on the title, artist name, and duration of a song. - Select the timestamp, user ID, level, song ID, artist ID, session ID, location, and user agent and set to `songplay_data`

Insert Records into Songplays Table

- Implement the `songplay_table_insert` query and run the cell below to insert records for the songplay actions in this log file into the `songplays` table. Remember to run `create_tables.py` before running the cell below to ensure you've created/resetted the `songplays` table in the sparkify database.

```
In [45]: for index, row in df.iterrows():

             # get songid and artistid from song and artist tables
             cur.execute(song_select, (row.song, row.artist, row.length))
             results = cur.fetchone()

             if results:
                 songid, artistid = results
             else:
                 songid, artistid = None, None

             # insert songplay record
             songplay_data = (row.ts,row.userId,row.level,songid,artistid,row.sessionId,row.loca
             try:
                 cur.execute(songplay_table_insert, songplay_data)
             except Error as e:
                 print("Error while inserting to songplay table", e)
             conn.commit()
```

Run `test.ipynb` to see if you've successfully added records to this table.

4 Close Connection to Sparkify Database

```
In [46]: conn.close()
```

5 Implement `etl.py`

Use what you've completed in this notebook to implement `etl.py`.

```
In [ ]:
```