etl

October 20, 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 [1]: import os
        import glob
        import psycopg2
        import pandas as pd
        from sql_queries import *
```

```
In [2]: conn = psycopg2.connect("host=127.0.0.1 dbname=sparkifydb user=student password=student"
        cur = conn.cursor()
```

```
In [3]: def get_files(filepath):
            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
```

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 [4]: song_files = get_files('data/song_data')
        song_files[0]
```

```
Out[4]: '/home/workspace/data/song_data/A/A/A/TRAAAAW128F429D538.json'
```

```
In [5]: filepath = song_files[0]
```

```
In [6]: df = pd.read_json(song_files[0],lines=True)
        df.head()
```

```
Out[6]:             artist_id  artist_latitude  artist_location  artist_longitude  \
        0  ARD7TVE1187B99BFB1              NaN  California - LA               NaN

          artist_name   duration  num_songs             song_id             title  \
        0      Casual  218.93179          1  SOMZWCG12A8C13C480  I Didn't Mean To

           year
        0     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 [7]: song_data = list(df[['song_id','title','artist_id','year','duration']].values[0])
        song_data
```

```
Out[7]: ['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 [8]: cur.execute(song_table_insert, song_data)
        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 [9]: artist_data = list(df[['artist_id', 'artist_name', 'artist_location', 'artist_latitude',
        artist_data
```

```
Out[9]: ['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 [10]: cur.execute(artist_table_insert, artist_data)
         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 [11]: log_files = get_files('data/log_data')
         log_files[0]
```

```
Out[11]: '/home/workspace/data/log_data/2018/11/2018-11-30-events.json'
```

```
In [12]: filepath = log_files[0]
```

```
In [13]: df = pd.read_json(filepath,lines=True)
         df.head()
```

```
Out[13]:           artist       auth firstName gender  itemInSession lastName  \
         0  Stephen Lynch  Logged In    Jayden      M              0     Bell
         1        Manowar  Logged In     Jacob      M              0    Klein
         2      Morcheeba  Logged In     Jacob      M              1    Klein
         3       Maroon 5  Logged In     Jacob      M              2    Klein
         4          Train  Logged In     Jacob      M              3    Klein

               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 userId
0  1543537327796  Mozilla/5.0 (compatible; MSIE 10.0; Windows NT...     91
1  1543540121796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...     73
2  1543540368796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...     73
3  1543540625796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...     73
4  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 [14]: df = df[df['page']=='NextSong']
         df.head(1)
```

```
Out[14]:           artist       auth firstName gender  itemInSession lastName  \
         0  Stephen Lynch  Logged In    Jayden      M              0     Bell

               length level                          location method      page  \
         0  182.85669  free  Dallas-Fort Worth-Arlington, TX    PUT  NextSong

            registration  sessionId                song  status             ts  \
         0  1.540992e+12        829  Jim Henson's Dead     200  1543537327796

                                                 userAgent userId
         0  Mozilla/5.0 (compatible; MSIE 10.0; Windows NT...     91
```

```
In [15]: ts = pd.to_datetime(df['ts'],unit='ms')
         ts.head()
```

```
Out[15]: 0   2018-11-30 00:22:07.796
         1   2018-11-30 01:08:41.796
         2   2018-11-30 01:12:48.796
         3   2018-11-30 01:17:05.796
         4   2018-11-30 01:20:56.796
         Name: ts, dtype: datetime64[ns]
```

```
In [16]: time_data = (ts, ts.dt.hour, ts.dt.day, ts.dt.week, ts.dt.month, ts.dt.year, ts.dt.week
         column_labels = ('timestamp','hour','day','week','month','year','weekday')
```

```
In [17]: time_df = pd.DataFrame.from_dict(dict(zip(column_labels,time_data)))
         time_df.head()
```

Out[17]:

	timestamp	hour	day	week	month	year	weekday
0	2018-11-30 00:22:07.796	0	30	48	11	2018	4
1	2018-11-30 01:08:41.796	1	30	48	11	2018	4
2	2018-11-30 01:12:48.796	1	30	48	11	2018	4
3	2018-11-30 01:17:05.796	1	30	48	11	2018	4
4	2018-11-30 01:20:56.796	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 [18]: for i, row in time_df.iterrows():
             cur.execute(time_table_insert, list(row))
             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 [19]: user_df = df[['userId', 'firstName','lastName','gender','level']]

         user_df
```

Out[19]:

	userId	firstName	lastName	gender	level
0	91	Jayden	Bell	M	free
1	73	Jacob	Klein	M	paid
2	73	Jacob	Klein	M	paid
3	73	Jacob	Klein	M	paid
4	73	Jacob	Klein	M	paid
5	73	Jacob	Klein	M	paid
6	73	Jacob	Klein	M	paid
7	73	Jacob	Klein	M	paid
9	73	Jacob	Klein	M	paid
10	73	Jacob	Klein	M	paid
11	73	Jacob	Klein	M	paid
12	73	Jacob	Klein	M	paid
13	73	Jacob	Klein	M	paid
14	73	Jacob	Klein	M	paid
15	73	Jacob	Klein	M	paid
16	73	Jacob	Klein	M	paid
17	73	Jacob	Klein	M	paid

18	73	Jacob	Klein	M	paid
19	73	Jacob	Klein	M	paid
23	86	Aiden	Hess	M	free
24	86	Aiden	Hess	M	free
25	86	Aiden	Hess	M	free
26	86	Aiden	Hess	M	free
27	86	Aiden	Hess	M	free
30	24	Layla	Griffin	F	paid
31	24	Layla	Griffin	F	paid
33	24	Layla	Griffin	F	paid
35	24	Layla	Griffin	F	paid
36	24	Layla	Griffin	F	paid
38	24	Layla	Griffin	F	paid
..	...	...	...	...	...
356	16	Rylan	George	M	paid
357	16	Rylan	George	M	paid
358	49	Chloe	Cuevas	F	paid
359	16	Rylan	George	M	paid
360	49	Chloe	Cuevas	F	paid
361	49	Chloe	Cuevas	F	paid
362	16	Rylan	George	M	paid
363	49	Chloe	Cuevas	F	paid
364	16	Rylan	George	M	paid
365	49	Chloe	Cuevas	F	paid
366	16	Rylan	George	M	paid
367	91	Jayden	Bell	M	free
368	16	Rylan	George	M	paid
369	49	Chloe	Cuevas	F	paid
370	91	Jayden	Bell	M	free
371	16	Rylan	George	M	paid
372	49	Chloe	Cuevas	F	paid
373	16	Rylan	George	M	paid
374	49	Chloe	Cuevas	F	paid
375	16	Rylan	George	M	paid
376	16	Rylan	George	M	paid
377	49	Chloe	Cuevas	F	paid
378	16	Rylan	George	M	paid
380	49	Chloe	Cuevas	F	paid
381	49	Chloe	Cuevas	F	paid
382	16	Rylan	George	M	paid
383	16	Rylan	George	M	paid
384	16	Rylan	George	M	paid
385	16	Rylan	George	M	paid
387	5	Elijah	Davis	M	free

[330 rows x 5 columns]

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 [20]: for i, row in user_df.iterrows():
             cur.execute(user_table_insert, row)
             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 [21]: 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 = (pd.to_datetime(row.ts, unit='ms'), int(row.userId), row.level, son
                              row.location, row.userAgent)
             cur.execute(songplay_table_insert, songplay_data)
             conn.commit()
```

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

4 Close Connection to Sparkify Database

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

5 Implement `etl.py`

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

In [ ]:

In [ ]:

In [ ]: