etl

November 25, 2019

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 [31]: import os
         import glob
         import psycopg2
         import pandas as pd
         from sql_queries import *
```

```
In [32]: conn = psycopg2.connect("host=127.0.0.1 dbname=sparkifydb user=student password=student
         cur = conn.cursor()
```

```
In [33]: 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 [34]: song_files = get_files('data/song_data')
```

```
In [35]: filepath = 'data/song_data/A/B/B/TRABBTA128F933D304.json'
```

```
In [36]: df = pd.read_json(filepath, lines=True)
         df.head()
```

```
Out[36]:             artist_id  artist_latitude artist_location  artist_longitude  \
         0  ARAGB201187FB3A161              NaN                             NaN

                           artist_name   duration  num_songs             song_id  \
         0  Pucho & His Latin Soul Brothers  338.23302          1  SOLEYHO12AB0188A85

                        title  year
         0  Got My Mojo Workin     0
```

```
In [37]: df['year'] = df['year'].apply(lambda x: x if x != 0 else None)
         df = df.replace({pd.np.nan: None, "": None})

         df.head()
```

```
Out[37]:             artist_id artist_latitude artist_location artist_longitude  \
         0  ARAGB201187FB3A161            None            None             None

                           artist_name   duration  num_songs             song_id  \
         0  Pucho & His Latin Soul Brothers  338.23302          1  SOLEYHO12AB0188A85

                        title  year
         0  Got My Mojo Workin  None
```

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 [38]: song_data = df[['song_id', 'title', 'artist_id', 'year', 'duration']].values[0]
         song_data
```

```
Out[38]: array(['SOLEYHO12AB0188A85', 'Got My Mojo Workin', 'ARAGB201187FB3A161',
                None, 338.23302], dtype=object)
```

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 [39]: 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 [40]: artist_data = df[['artist_id', 'artist_name', 'artist_location', 'artist_latitude', 'ar
         artist_data
```

```
Out[40]: array(['ARAGB201187FB3A161', 'Pucho & His Latin Soul Brothers', None, None,
                None], dtype=object)
```

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 [41]: 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 [42]: log_files = get_files('data/log_data')
```

```
In [45]: filepath = 'data/log_data/2018/11/2018-11-09-events.json'
```

```
In [47]: df = pd.read_json(filepath, lines=True)
         df.head()
```

```
Out[47]:                  artist       auth firstName gender  itemInSession lastName  \
         0                  Muse  Logged In    Harper      M              1  Barrett
         1          Beastie Boys  Logged In    Harper      M              2  Barrett
         2               Shakira  Logged In    Harper      M              3  Barrett
         3                Selena  Logged In    Harper      M              4  Barrett
         4  Kid Cudi Vs Crookers  Logged In    Harper      M              5  Barrett

               length level                              location method      page  \
         0  209.50159  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
```

```
1  161.56689  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
2  145.84118  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
3  172.66893  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
4  162.97751  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong

   registration  sessionId                                               song  \
0  1.540685e+12        275  Supermassive Black Hole (Twilight Soundtrack V...
1  1.540685e+12        275                                         Lighten Up
2  1.540685e+12        275                                       Pienso En Ti
3  1.540685e+12        275                                     Amor Prohibido
4  1.540685e+12        275                                       Day 'N' Nite

   status             ts                                          userAgent  \
0     200  1541721977796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
1     200  1541722186796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
2     200  1541722347796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
3     200  1541722492796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
4     200  1541722664796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...

  userId
0     42
1     42
2     42
3     42
4     42
```

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 [48]: # Filter records by NextSong action
         df = df[df['page']=='NextSong']
         df.head()
```

```
Out[48]:
                  artist       auth firstName gender  itemInSession lastName  \
0                   Muse  Logged In    Harper      M              1  Barrett
1           Beastie Boys  Logged In    Harper      M              2  Barrett
2                Shakira  Logged In    Harper      M              3  Barrett
3                 Selena  Logged In    Harper      M              4  Barrett
```

```
        4  Kid Cudi Vs Crookers  Logged In    Harper      M             5  Barrett

              length level                               location method      page  \
        0  209.50159  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
        1  161.56689  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
        2  145.84118  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
        3  172.66893  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong
        4  162.97751  paid  New York-Newark-Jersey City, NY-NJ-PA    PUT  NextSong

           registration  sessionId                                               song  \
        0  1.540685e+12        275  Supermassive Black Hole (Twilight Soundtrack V...
        1  1.540685e+12        275                                         Lighten Up
        2  1.540685e+12        275                                       Pienso En Ti
        3  1.540685e+12        275                                     Amor Prohibido
        4  1.540685e+12        275                                     Day 'N' Nite

           status             ts                                          userAgent  \
        0     200  1541721977796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
        1     200  1541722186796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
        2     200  1541722347796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
        3     200  1541722492796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...
        4     200  1541722664796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...

          userId
        0     42
        1     42
        2     42
        3     42
        4     42
```

```
In [50]: #Convert the ts timestamp column to datetime. Current timestamp is in ms
         t = pd.to_datetime(df['ts'], unit='ms')
         t.head()
```

```
Out[50]: 0   2018-11-09 00:06:17.796
         1   2018-11-09 00:09:46.796
         2   2018-11-09 00:12:27.796
         3   2018-11-09 00:14:52.796
         4   2018-11-09 00:17:44.796
         Name: ts, dtype: datetime64[ns]
```

```
In [51]: time_data = pd.concat([t, t.dt.hour, t.dt.day, t.dt.week, t.dt.month, t.dt.year, t.dt.w
         column_labels = ['start_time', 'hour', 'day', 'week', 'month', 'year', 'weekday']
```

```
In [52]: time_df = pd.DataFrame(data=time_data.values, columns=column_labels)
         time_df.head()
```

```
Out[52]:                   start_time hour day week month  year weekday
         0  2018-11-09 00:06:17.796000    0   9   45    11  2018       4
```

```
1  2018-11-09 00:09:46.796000    0    9    45    11   2018        4
2  2018-11-09 00:12:27.796000    0    9    45    11   2018        4
3  2018-11-09 00:14:52.796000    0    9    45    11   2018        4
4  2018-11-09 00:17:44.796000    0    9    45    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 [53]: 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 [56]: user_df = df[['userId', 'firstName', 'lastName', 'gender', 'level']]
         user_df.head()
```

```
Out[56]:   userId firstName lastName gender level
         0     42    Harper  Barrett      M  paid
         1     42    Harper  Barrett      M  paid
         2     42    Harper  Barrett      M  paid
         3     42    Harper  Barrett      M  paid
         4     42    Harper  Barrett      M  paid
```

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 [58]: 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 [59]: 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
             # INSERT INTO songplays(start_time, user_id, level, song_id, artist_id, session_id,
             songplay_data = (pd.to_datetime(row.ts, unit='ms'), row.userId, row.level, songid,
             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 [60]: conn.close()
```

5 Implement `etl.py`

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

```
In [ ]:
```