etl

September 28, 2021

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 *
        import json
```

```
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')
```

```
In [5]: filepath = song_files[0]
        print(filepath)
```

```
/home/workspace/data/song_data/A/A/A/TRAAAAW128F429D538.json
```

```
In [6]: with open(filepath, 'r') as f: # getting data as dict (load_json not working due to scal
            data = json.load(f)

        df = pd.DataFrame({0: data}) # manually indexing since data is scalar
        df = df.transpose() # transposing for column view
        df.head()
```

```
Out[6]:             artist_id artist_latitude  artist_location artist_longitude  \
        0  ARD7TVE1187B99BFB1            None  California - LA             None

          artist_name duration num_songs             song_id             title year
        0      Casual  218.932         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 [7]: song_list = [] # list of dictionaries to host song data to be converted to pandas df

        for i in range(len(song_files)): # iterating through list of song files
            path = song_files[i]
            with open(path, 'r') as f: # appending each to the list created above
                song_list.append(json.load(f))

        print(len(song_list))
        song_df = pd.DataFrame(song_list) # converting list of dictionaries to pandas df
        print(song_df.dtypes)
        print(song_df.shape)
        song_df.head()
```

```
81
artist_id            object
artist_latitude     float64
artist_location      object
artist_longitude    float64
artist_name          object
duration            float64
num_songs             int64
song_id              object
title                object
year                  int64
```

```
dtype: object
(81, 10)
```

```
Out[7]:             artist_id  artist_latitude              artist_location  \
        0  ARD7TVE1187B99BFB1              NaN              California - LA
        1  ARNTLGG11E2835DDB9              NaN
        2  AR8ZCNI1187B9A069B              NaN
        3  AR10USD1187B99F3F1              NaN  Burlington, Ontario, Canada
        4  ARMJAGH1187FB546F3         35.14968                  Memphis, TN

           artist_longitude            artist_name   duration  num_songs  \
        0               NaN                 Casual  218.93179          1
        1               NaN                    Clp  266.39628          1
        2               NaN       Planet P Project  269.81832          1
        3               NaN  Tweeterfriendly Music  189.57016          1
        4         -90.04892           The Box Tops  148.03546          1

                      song_id                             title  year
        0  SOMZWCG12A8C13C480                    I Didn't Mean To     0
        1  SOUDSGM12AC9618304  Insatiable (Instrumental Version)     0
        2  SOIAZJW12AB01853F1                        Pink World  1984
        3  SOHKNRJ12A6701D1F8                      Drop of Rain     0
        4  SOCIWDW12A8C13D406                         Soul Deep  1969
```

```
In [8]: song_data_subset = song_df[['song_id', 'title', 'artist_id', 'year', 'duration']]
        song_data_array = song_data_subset.values
        song_data_list = song_data_array.tolist()
        first_song_array = song_data_array[0]
        song_data = first_song_array.tolist()
```

```
In [9]: #for i in range(len(song_data_list)):
        #    print(song_data_list[i])
        # for test purposes only
```

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 [10]: #cur.execute(song_table_insert, first_song_tuple) for SINGLE row as requested above
         #cur.execute(song_table_insert, song_data)

         #conn.commit()

         # insert all data (which is needed for the final implementation)

         for i in range(len(song_data_list)):
```

```
        current_song = song_data_list[i]
        cur.execute(song_table_insert, current_song)
        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 [11]: artist_data = song_df[['artist_id', 'artist_name', 'artist_location', 'artist_latitude'
         artist_array = artist_data.values
         artist_list = artist_array.tolist()
         first_artist_array = artist_array[0]
         artist_data = first_artist_array.tolist()
         artist_data
```

```
Out[11]: ['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 [12]: # insert single (as requested in the question...)
         #cur.execute(artist_table_insert, artist_data)
         #conn.commit()

         # insert all data (which is needed for the final implementation)

         for i in range(len(artist_list)):
             current_artist = artist_list[i]
             cur.execute(artist_table_insert, current_artist)
             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 [13]: log_files = get_files('data/log_data')

In [14]: filepath = log_files[0]
         print(filepath)
```

```
/home/workspace/data/log_data/2018/11/2018-11-30-events.json
```

```
In [15]: df = pd.read_json (filepath, lines = True)
         print(df.shape)
         df.head()
```

```
(388, 18)
```

```
Out[15]:           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 [16]: df_ns = df[df['page'] == 'NextSong']
         print(df_ns.shape)
         df_ns.head()
```

```
(330, 18)
```

```
Out[16]:          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
```

```
In [17]: t = df_ns.copy()
         t['ts'] = pd.to_datetime(df_ns['ts'], unit='ms')
         t.head()
```

```
Out[17]:          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  \
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 [18]: t['year'] = pd.DatetimeIndex(t['ts']).year
         t['month'] = pd.DatetimeIndex(t['ts']).month
         t['week'] = pd.DatetimeIndex(t['ts']).week
         t['day'] = pd.DatetimeIndex(t['ts']).day
         t['weekday'] = pd.DatetimeIndex(t['ts']).weekday
         t['hour'] = pd.DatetimeIndex(t['ts']).hour


         # converting df to dict
         df_dict = t.T.to_dict()

         list_of_time_data = []

         # only want dates
         for key, value in df_dict.items():
```

```
        entry = value
        dates_dict = {k: v for k, v in entry.items() if k in ['year', 'month', 'week', 'day
        list_of_time_data.append(dates_dict)

    for i in range(len(list_of_time_data)):
        print(list_of_time_data[i])
        if i > 5: break # keeping the view clean
```

```
{'ts': Timestamp('2018-11-30 00:22:07.796000'), 'year': 2018, 'month': 11, 'week': 48, 'day': 30
{'ts': Timestamp('2018-11-30 01:08:41.796000'), 'year': 2018, 'month': 11, 'week': 48, 'day': 30
{'ts': Timestamp('2018-11-30 01:12:48.796000'), 'year': 2018, 'month': 11, 'week': 48, 'day': 30
{'ts': Timestamp('2018-11-30 01:17:05.796000'), 'year': 2018, 'month': 11, 'week': 48, 'day': 30
{'ts': Timestamp('2018-11-30 01:20:56.796000'), 'year': 2018, 'month': 11, 'week': 48, 'day': 30
{'ts': Timestamp('2018-11-30 01:24:32.796000'), 'year': 2018, 'month': 11, 'week': 48, 'day': 30
{'ts': Timestamp('2018-11-30 01:28:19.796000'), 'year': 2018, 'month': 11, 'week': 48, 'day': 30
```

```
In [19]: time_df = pd.DataFrame(list_of_time_data)

         # want order consistent with SQL INSERT query
         time_df = time_df[['ts', 'hour', 'day', 'week', 'month', 'year', 'weekday']]

         time_df.head()
```

```
Out[19]:                        ts  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 [20]: 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 [21]: user_df = df[['userId', 'firstName', 'lastName', 'gender', 'level']]
         user_df.head()
```

```
Out[21]:   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
```

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 [22]: 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 [23]: print(df.dtypes)
         print(df.shape)
         df.head()
```

```
artist            object
auth              object
firstName         object
gender            object
itemInSession      int64
lastName          object
length           float64
level             object
location          object
```

```
method           object
page             object
registration    float64
sessionId         int64
song             object
status            int64
ts                int64
userAgent        object
userId           object
dtype: object
(388, 18)
```

```
Out[23]:           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
```

```
In [24]: for index, row in df.iterrows():
             cur.execute(song_select, (row.song, row.artist, row.length)) # running the songsele
             results = cur.fetchone()
             if results:
                 print(results)
                 songid, artistid = results
```

```
            else:
                songid, artistid = None, None
```

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

4 Close Connection to Sparkify Database

```
In [25]: conn.close()
```

5 Implement `etl.py`

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

```
In [26]: %run etl.py
```

```
81 files found in data/song_data
1/81 files processed.
2/81 files processed.
3/81 files processed.
4/81 files processed.
5/81 files processed.
6/81 files processed.
7/81 files processed.
8/81 files processed.
9/81 files processed.
10/81 files processed.
11/81 files processed.
12/81 files processed.
13/81 files processed.
14/81 files processed.
15/81 files processed.
16/81 files processed.
17/81 files processed.
18/81 files processed.
19/81 files processed.
20/81 files processed.
21/81 files processed.
22/81 files processed.
23/81 files processed.
24/81 files processed.
25/81 files processed.
26/81 files processed.
27/81 files processed.
28/81 files processed.
29/81 files processed.
30/81 files processed.
31/81 files processed.
```

```
32/81 files processed.
33/81 files processed.
34/81 files processed.
35/81 files processed.
36/81 files processed.
37/81 files processed.
38/81 files processed.
39/81 files processed.
40/81 files processed.
41/81 files processed.
42/81 files processed.
43/81 files processed.
44/81 files processed.
45/81 files processed.
46/81 files processed.
47/81 files processed.
48/81 files processed.
49/81 files processed.
50/81 files processed.
51/81 files processed.
52/81 files processed.
53/81 files processed.
54/81 files processed.
55/81 files processed.
56/81 files processed.
57/81 files processed.
58/81 files processed.
59/81 files processed.
60/81 files processed.
61/81 files processed.
62/81 files processed.
63/81 files processed.
64/81 files processed.
65/81 files processed.
66/81 files processed.
67/81 files processed.
68/81 files processed.
69/81 files processed.
70/81 files processed.
71/81 files processed.
72/81 files processed.
73/81 files processed.
74/81 files processed.
75/81 files processed.
76/81 files processed.
77/81 files processed.
78/81 files processed.
79/81 files processed.
```

```
80/81 files processed.
81/81 files processed.
30 files found in data/log_data
1/30 files processed.
2/30 files processed.
3/30 files processed.
4/30 files processed.
5/30 files processed.
6/30 files processed.
7/30 files processed.
8/30 files processed.
9/30 files processed.
10/30 files processed.
11/30 files processed.
12/30 files processed.
13/30 files processed.
14/30 files processed.
15/30 files processed.
16/30 files processed.
17/30 files processed.
18/30 files processed.
19/30 files processed.
20/30 files processed.
21/30 files processed.
22/30 files processed.
23/30 files processed.
24/30 files processed.
25/30 files processed.
26/30 files processed.
27/30 files processed.
28/30 files processed.
29/30 files processed.
30/30 files processed.
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```