etl

October 1, 2023

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/')

In [5]: filepath = song_files[0]

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

```
Out[6]:             artist_id  artist_latitude artist_location  artist_longitude  \
        0  AR558FS1187FB45658              NaN                               NaN

          artist_name  duration  num_songs             song_id  title  year
        0     40 Grit  75.67628          1  SOGDBUF12A8C140FAA  Intro  2003
```

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]: ['SOGDBUF12A8C140FAA', 'Intro', 'AR558FS1187FB45658', 2003, 75.67628]
```

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]: ['AR558FS1187FB45658', '40 Grit', '', 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/')

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	None	Logged In	Kevin	M	0	Arellano	
1	Fu	Logged In	Kevin	M	1	Arellano	
2	None	Logged In	Maia	F	0	Burke	
3	All Time Low	Logged In	Maia	F	1	Burke	
4	Nik & Jay	Logged In	Wyatt	M	0	Scott	

	length	level	location	method	page	\
0	NaN	free	Harrisburg-Carlisle, PA	GET	Home	
1	280.05832	free	Harrisburg-Carlisle, PA	PUT	NextSong	
2	NaN	free	Houston-The Woodlands-Sugar Land, TX	GET	Home	
3	177.84118	free	Houston-The Woodlands-Sugar Land, TX	PUT	NextSong	
4	196.51873	free	Eureka-Arcata-Fortuna, CA	PUT	NextSong	

	registration	sessionId	song	status	\
0	1.540007e+12	514	None	200	
1	1.540007e+12	514	Ja I Ty	200	
2	1.540677e+12	510	None	200	
3	1.540677e+12	510	A Party Song (The Walk of Shame)	200	
4	1.540872e+12	379	Pop-Pop!	200	

	ts	userAgent	userId
0	1542069417796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	66
1	1542069637796	"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...	66
2	1542071524796	"Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...	51
3	1542071549796	"Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...	51
4	1542079142796	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7...	9

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.loc[df['page'] == 'NextSong']
         df.head()
```

```
Out[14]:            artist       auth firstName gender  itemInSession  lastName  \
         1              Fu  Logged In     Kevin      M              1  Arellano
         3    All Time Low  Logged In      Maia      F              1     Burke
         4       Nik & Jay  Logged In     Wyatt      M              0     Scott
         5  Quad City DJ's  Logged In     Chloe      F              0    Cuevas
         8      Hoobastank  Logged In      Noah      M              1    Chavez

               length level                                location method      page  \
         1  280.05832  free                     Harrisburg-Carlisle, PA    PUT  NextSong
         3  177.84118  free  Houston-The Woodlands-Sugar Land, TX    PUT  NextSong
         4  196.51873  free                   Eureka-Arcata-Fortuna, CA    PUT  NextSong
         5  451.44771  free     San Francisco-Oakland-Hayward, CA    PUT  NextSong
         8  232.17587  free                       Ogden-Clearfield, UT    PUT  NextSong

            registration  sessionId                                        song  status  \
         1  1.540007e+12        514                                      Ja I Ty     200
         3  1.540677e+12        510           A Party Song (The Walk of Shame)     200
         4  1.540872e+12        379                                    Pop-Pop!     200
         5  1.540941e+12        506  C'mon N' Ride It (The Train) (LP Version)     200
         8  1.540971e+12        492                                 Born To Lead     200

                       ts                                          userAgent userId
         1  1542069637796  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4...     66
         3  1542071549796  "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebK...     51
         4  1542079142796  Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7...      9
         5  1542081112796  Mozilla/5.0 (Windows NT 5.1; rv:31.0) Gecko/20...     49
         8  1542085206796  Mozilla/5.0 (Windows NT 6.1; WOW64; rv:32.0) G...     94
```

```
In [15]: t = pd.to_datetime(df['ts'], unit='ms')
         t.head()
```

```
Out[15]: 1   2018-11-13 00:40:37.796
         3   2018-11-13 01:12:29.796
         4   2018-11-13 03:19:02.796
         5   2018-11-13 03:51:52.796
         8   2018-11-13 05:00:06.796
         Name: ts, dtype: datetime64[ns]
```

```
In [16]: time_data = [df.ts.values, t.dt.hour.values, t.dt.day.values, t.dt.weekofyear.values, t
         column_labels = ['start_time', 'hour', 'day', 'week', 'month', 'year', 'weekday']

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

Out[17]:

	start_time	hour	day	week	month	year	weekday
0	1542069637796	0	13	46	11	2018	1
1	1542071549796	1	13	46	11	2018	1
2	1542079142796	3	13	46	11	2018	1
3	1542081112796	3	13	46	11	2018	1
4	1542085206796	5	13	46	11	2018	1

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 [ ]: 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 [ ]: user_df = df[['userId', 'firstName', 'lastName', 'gender', 'level']]
        user_df.head()
```

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 [ ]: 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 [ ]: 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 = ()
            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 [ ]:
```