etl

April 9, 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/')
```

```
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  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 = df[['song_id', 'title', 'artist_id', 'year', 'duration']].values[0]
        song_data = song_data.tolist()
        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 [ ]: artist_data = df[['artist_id','artist_name','artist_location','artist_latitude','artist_
        artist_data = artist_data.tolist()
        artist_data
```

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

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

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

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

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

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

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

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 = (row.ts, row.userId, row.level, songid, artistid, row.sessionId, row
            cur.execute(songplay_table_insert, songplay_data)
            conn.commit()
            row.head()
```

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 [ ]:
```