

Return to "Data Engineering Nanodegree" in the classroom

## Data Modeling with Postgres

```
REVIEW
                                                       CODE REVIEW 3
                                                           HISTORY
▼ sql_queries.py 3
    1 # DROP TABLES
    3 songplay_table_drop = "DROP TABLE IF EXISTS songplays;"
    4 user_table_drop = "DROP TABLE IF EXISTS users;"
     5 song table drop = "DROP TABLE IF EXISTS songs;"
     6 artist_table_drop = "DROP TABLE IF EXISTS artists;"
     7 time table drop = "DROP TABLE IF EXISTS time;"
    9 # CREATE TABLES
    10
   11 songplay_table_create = ("""
    12 CREATE TABLE IF NOT EXISTS songplays \
   13 (songplay_id SERIAL PRIMARY KEY, start_time BIGINT, user_id INT DEFAULT NULL, level CHAR(4), song_id VARCHAR, artist_id V
```



## REQUIRED

• Kindly add start\_time and user\_id as NOT NULLs here. This is to avoid any entry where there is either no user or session (no start time) in the fact table.

```
15
16 user_table_create = ("""
17 CREATE TABLE IF NOT EXISTS users \
18 (user_id INT PRIMARY KEY , first_name VARCHAR (255), last_name VARCHAR (255), gender CHAR(1), level CHAR(4));
19 """)
20
21 song_table_create = ("""
22 CREATE TABLE IF NOT EXISTS songs \
23 (song_id VARCHAR (255) PRIMARY KEY, title VARCHAR (255), artist_id VARCHAR (255), year INT, duration NUMERIC(8,5));
24 """)
25
26 artist_table_create = ("""
27 CREATE TABLE IF NOT EXISTS artists \
28 (artist_id VARCHAR (255) PRIMARY KEY, name VARCHAR (255), location VARCHAR (255), latitude DECIMAL, longitude DECIMAL);
29 """)
30
31 time_table_create = ("""
32 CREATE TABLE IF NOT EXISTS time \
33 (start_time BIGINT PRIMARY KEY, hour INT, day INT, week INT, month INT, year INT, weekday INT);
REOUIRED
   • start_time should be of type timestamp
34
```

```
35
36
37
38
39 # INSERT RECORDS
40
41 songplay_table_insert = ("""
INSERT INTO songplays (start_time, user_id, level, song_id, artist_id, session_id, location, user_agent) \
```

```
VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

44 """)

45

46 user_table_insert = ("""

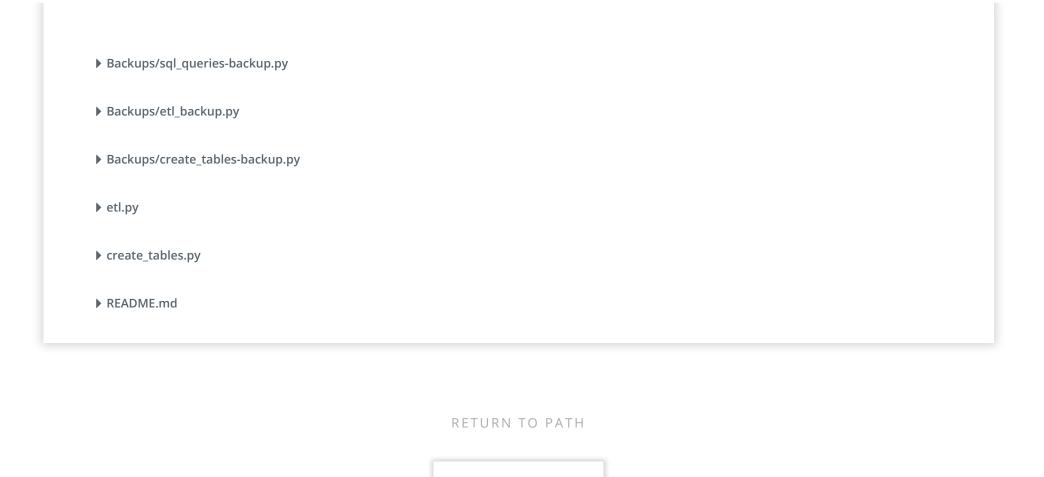
47 INSERT INTO users (user_id, first_name, last_name, gender, level) \

48 VALUES (%s, %s, %s, %s, %s) ON CONFLICT (user_id) DO NOTHING;
```

## REQUIRED

• Make the changes here as suggested.

```
49
50
51 song_table_insert = ("""
52 INSERT INTO songs (song_id, title, artist_id, year, duration) \
                    VALUES (%s, %s, %s, %s, %s) ON CONFLICT (song_id) DO NOTHING;
53
54
55
56 artist_table_insert = ("""
57 INSERT INTO artists (artist id, name, location, latitude, longitude) \
                    VALUES (%s, %s, %s, %s, %s) ON CONFLICT (artist_id) DO NOTHING;
58
59
60
61
62 time_table_insert = ("""
63 INSERT INTO time (start time, hour, day, week, month, year, weekday) \
                    VALUES (%s, %s, %s, %s, %s, %s, %s) ON CONFLICT (start time) DO NOTHING;
64
65
66
67 # FIND SONGS
68
69 song_select = ("""
70 select songs.song_id, artists.artist_id \
71 from songs JOIN artists ON songs.artist id = artists.artist id \
72 where songs.title = %s AND artists.name = %s AND songs.duration = %s;
73 """)
74
75 # QUERY LISTS
76
77 create_table_queries = [songplay_table_create, user_table_create, song_table_create, artist_table_create, time_table_crea
78 drop table queries = [songplay table drop, user table drop, song table drop, artist table drop, time table drop]
```



Rate this review