proj0

October 9, 2024

```
[6]: # Initialize Otter
     import otter
     grader = otter.Notebook("proj0.ipynb")
```

1 Project 0 - SQL Review – Music Querying

1.1 Due Date: Wednesday, September 4th, 5:00pm

In this project, we will be working with SQL on a Billboard Music database.

1.2 Collaboration Policy

Please read out course collaboration policy. Do not share code. If you discuss the assignments with others, please include their names below.

Collaborators: *list collaborators here*

1.3 Objectives

- Review SQL syntax from prerequisite courses

Note: If at any point during the project, the internal state of the database or its tables have been modified in an undesirable way (i.e. a modification not resulting from the instructions of a question), restart your kernel, clear output, and simply re-run the notebook as normal. This will shutdown your current connection to the database, which will prevent the issue of multiple connections to the database at any given point. When re-running the notebook, you will create a fresh database based on the provided Postgres dump.

If you face slow kernel times or are unable to open your notebook, restart your server by going to File -> Hub Control Panel -> Stop My Server and then clicking Start My Server.

1.4 Logistics & Scoring Breakdown

Please read the submission instructions carefully and double check that your submission is not throwing any errors. Please ensure that public tests pass upon submission. It is your responsibility to wait until the autograder finishes running. We will not be accepting regrade requests for submission issues.

Each coding question has **both public tests and hidden tests**. Roughly 50% of your coding grade will be made up of your score on the public tests released to you, while the remaining 50%

will be made up of unreleased hidden tests. If there are free-response questions (marked 'm' in the table below), they are manually graded.

This is an **individual project**. However, you're welcome to collaborate with any other student in the class as long as it's within the academic honesty guidelines.

Question	Points
0	3
1	3
2	3
3	3
4	3
5a	3
5b	3
6	0
Total	21

Summary: 21 points (autograded: 21, manual: 0)

[7]:
```python
# Run this cell to set up imports
import numpy as np
import pandas as pd
```

2 Before You Start

2.1 Assignment Tips

We recommend you read through Fall 2024 Assignment Tips—a handy resource that has many other tips that we **highly recommend** taking a moment to read through to save you time and improve your workflow.

- PostgreSQL documentation
- Jupyter cell magic
- JupySQL and magic commands in Jupyter
- JupyterHub keyboard shortcuts
- psql and common meta-commands (not required for Proj 0)
- Debugging:
  - Where to create new cells to play nice with the autograder
  - Opening/closing connections, deleting databases if all else fails
- Local installation (not supported by staff officially, but for your reference)

2.2 About PostgreSQL

In prior courses, you've used the databases SQLite or DuckDB which live "in memory". These have had their own versions of For DATA 101, we'll be connecting to PostgreSQL (also called "postgres"), an open-source object-relational database with an eponymous SQL variant.

2.3 Table of Contents

Jump to different parts of this notebook: * Before You Start: This section * Setup: Database Setup, Grader Setup (make sure to run these cells) * The billboard Database: Information about this project's database schema, plus optional features for you to explore * Question 0: The first assignment question

3 Setup

3.1 Database Setup

We are going to be using the `JupySQL` library to connect our notebook to a PostgreSQL database server on your JupyterHub account. Running the next cell will do so; you should not see any error messages after it executes.

[8]:
```
# The first time you are running this cell, you may need to run the following␣
↪line as: %load_ext sql
%reload_ext sql
```

In the next cell, we will unzip the data. This only needs to be done once.

[9]:
```
!unzip -u data/billboard.zip -d data/
```

```
Archive:  data/billboard.zip
```

Create the `billboard` database: We will use PostgreSQL commands to create a database and import our data into it. Run the following cell to do this. It may take a few seconds.

- You can also run these cells in the command-line via `psql`.
- If you run into the **role does not exist** error, feel free to ignore it. It does not affect data import.

[10]:
```
!psql postgresql://localhost/billboard -c 'SELECT␣
↪pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity WHERE␣
↪datname = current_database()  AND pid <> pg_backend_pid();'
!psql postgresql://localhost/postgres -c 'DROP DATABASE IF EXISTS billboard'
!psql postgresql://localhost/postgres -c 'CREATE DATABASE billboard'
!psql postgresql://localhost/billboard -f data/billboard.sql -q
```

```
 pg_terminate_backend
----------------------
(0 rows)

DROP DATABASE
CREATE DATABASE
psql:data/billboard.sql:16: ERROR:  role "michael" does not exist
psql:data/billboard.sql:33: ERROR:  role "michael" does not exist
```

Connect to `billboard` database in the Notebook: Now let's connect to the new database we just created! There should be no errors after running the following cell.

[11]:
```
%sql postgresql://localhost/billboard
```

Quick check: To make sure things are working, let's fetch 10 rows from one of our tables `tiktok_top_50`. Just run the following cells. If there are no errors, no further action is needed.

[12]:
```
%%sql
SELECT * FROM tiktok_top_50 LIMIT 10;
```

```
Running query in 'postgresql://localhost/billboard'

10 rows affected.
```

[12]:
```
+-------------+------+----------------------+--------------------------------
------------------------------------+-------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+---------------+----------------+
| week_ending | rank |        title         |
artist                                |
image_url
| peak_position | weeks_on_chart |
+-------------+------+----------------------+--------------------------------
------------------------------------+-------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+---------------+----------------+
|  2023-09-16 |  1   |        SkeeYee       |
Sexyy Red                              |              <a href=https://charts-
static.billboard.com/img/2023/08/sexyyred-8hq-skeeyee-
jvl-180x180.jpg>https://charts-static.billboard.com/img/2023/08/sexyyred-8hq-
skeeyee-jvl-180x180.jpg</a>                |       1       |       1        |
|  2023-09-16 |  2   |   Paint The Town Red |
Doja Cat                              |       <a href=https://charts-
static.billboard.com/img/2023/08/dojacat-cqv-paintthetownred-
unh-180x180.jpg>https://charts-static.billboard.com/img/2023/08/dojacat-cqv-
paintthetownred-unh-180x180.jpg</a>       |       2       |       1        |
|  2023-09-16 |  3   |         August       |
Taylor Swift                            |                      <a
href=https://charts-static.billboard.com/img/2006/07/taylor-
swift-9sy-180x180.jpg>https://charts-static.billboard.com/img/2006/07/taylor-
swift-9sy-180x180.jpg</a>                    |       3       |       1
|
|  2023-09-16 |  4   |          Go!         | Greg Cipes, Scott Menville, Khary
Payton, Tara Strong & Hynden Walch |                  <a href=https://charts-
static.billboard.com/img/2018/08/gregcipes-000-go-
gll-180x180.jpg>https://charts-
```

```
static.billboard.com/img/2018/08/gregcipes-000-go-gll-180x180.jpg</a>
|       4       |       1        |
|  2023-09-16 |  5   | I Remember Everything |                 Zach Bryan
Featuring Kacey Musgraves                 | <a href=https://charts-
static.billboard.com/img/2023/09/zachbryan-de1-irememberereverything-
to3-180x180.jpg>https://charts-static.billboard.com/img/2023/09/zachbryan-
de1-irememberereverything-to3-180x180.jpg</a> |       5       |       1        |
|  2023-09-16 |  6   |         Deli          |                              Ice
Spice                               |                           <a
href=https://charts-static.billboard.com/img/2022/09/icespice-
ouz-180x180.jpg>https://charts-static.billboard.com/img/2022/09/icespice-
ouz-180x180.jpg</a>                           |       6       |       1        |
|  2023-09-16 |  7   |     I Love You Hoe    |                           Odetari
& 9lives                           |           <a href=https://charts-static.bil
lboard.com/img/2023/08/odetari-31a-iloveyouhoe-p6l-180x180.jpg>https://charts-
static.billboard.com/img/2023/08/odetari-31a-iloveyouhoe-p6l-180x180.jpg</a>
|       7       |       1        |
|  2023-09-16 |  8   |    It's Getting Hot   |                              NLE
Choppa                              |      <a href=https://charts-static.billbo
ard.com/img/2023/08/nlechoppa-000-itsgettinghot-k6f-180x180.jpg>https://charts-
static.billboard.com/img/2023/08/nlechoppa-000-itsgettinghot-k6f-180x180.jpg</a>
|       8       |       1        |
|  2023-09-16 |  9   |      I'm Blessed      |                   Charlie Wilson
Featuring T.I.                     |      <a href=https://charts-
static.billboard.com/img/2017/01/charlie-wilson-4cp-
imblessed-p3c-180x180.jpg>https://charts-
static.billboard.com/img/2017/01/charlie-wilson-4cp-
imblessed-p3c-180x180.jpg</a>      |       9       |       1        |
|  2023-09-16 |  10  |      Let It Whip      |                              Dazz
Band                               |           <a href=https://charts-
static.billboard.com/img/1982/03/dazz-band-s10-letitwhip-
dkw-180x180.jpg>https://charts-static.billboard.com/img/1982/03/dazz-
band-s10-letitwhip-dkw-180x180.jpg</a>           |       10      |       1
|
+-------------+------+-----------------------+---------------------------------
------------------------------------+---------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+---------------+----------------+
Truncated to displaylimit of 10.
```

[13]:
```
%%sql
SELECT *
  FROM tiktok_top_50
LIMIT 10
```

```
Running query in 'postgresql://localhost/billboard'
```

```
10 rows affected.
```

[13]:
```
+-------------+------+------------------------+---------------------------------
-------------------------------------+------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+---------------+----------------+
| week_ending | rank |         title          |
artist                                |
image_url
| peak_position | weeks_on_chart |
+-------------+------+------------------------+---------------------------------
-------------------------------------+------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+---------------+----------------+
|  2023-09-16 |  1   |        SkeeYee         |
Sexyy Red                             |              <a href=https://charts-
static.billboard.com/img/2023/08/sexyyred-8hq-skeeyee-
jvl-180x180.jpg>https://charts-static.billboard.com/img/2023/08/sexyyred-8hq-
skeeyee-jvl-180x180.jpg</a>              |       1       |       1        |
|  2023-09-16 |  2   |   Paint The Town Red   |
Doja Cat                             |       <a href=https://charts-
static.billboard.com/img/2023/08/dojacat-cqv-paintthetownred-
unh-180x180.jpg>https://charts-static.billboard.com/img/2023/08/dojacat-cqv-
paintthetownred-unh-180x180.jpg</a>       |       2       |       1        |
|  2023-09-16 |  3   |         August         |
Taylor Swift                          |                      <a
href=https://charts-static.billboard.com/img/2006/07/taylor-
swift-9sy-180x180.jpg>https://charts-static.billboard.com/img/2006/07/taylor-
swift-9sy-180x180.jpg</a>                      |       3       |       1
|
|  2023-09-16 |  4   |          Go!           | Greg Cipes, Scott Menville, Khary
Payton, Tara Strong & Hynden Walch |                  <a href=https://charts-
static.billboard.com/img/2018/08/gregcipes-000-go-
gll-180x180.jpg>https://charts-
static.billboard.com/img/2018/08/gregcipes-000-go-gll-180x180.jpg</a>
|       4       |       1        |
|  2023-09-16 |  5   | I Remember Everything |                 Zach Bryan
Featuring Kacey Musgraves                 | <a href=https://charts-
static.billboard.com/img/2023/09/zachbryan-de1-irememberevery thing-
to3-180x180.jpg>https://charts-static.billboard.com/img/2023/09/zachbryan-
de1-irememberevery thing-to3-180x180.jpg</a> |       5       |       1        |
|  2023-09-16 |  6   |          Deli          |                              Ice
Spice                                 |                          <a
href=https://charts-static.billboard.com/img/2022/09/icespice-
ouz-180x180.jpg>https://charts-static.billboard.com/img/2022/09/icespice-
```

```
ouz-180x180.jpg</a>                               |       6        |       1        |
|  2023-09-16 |  7   |      I Love You Hoe     |                             Odetari
& 9lives                           |            <a href=https://charts-static.bil
lboard.com/img/2023/08/odetari-31a-iloveyouhoe-p6l-180x180.jpg>https://charts-
static.billboard.com/img/2023/08/odetari-31a-iloveyouhoe-p6l-180x180.jpg</a>
|       7        |       1        |
|  2023-09-16 |  8   |    It's Getting Hot   |                                NLE
Choppa                              |       <a href=https://charts-static.billbo
ard.com/img/2023/08/nlechoppa-000-itsgettinghot-k6f-180x180.jpg>https://charts-
static.billboard.com/img/2023/08/nlechoppa-000-itsgettinghot-k6f-180x180.jpg</a>
|       8        |       1        |
|  2023-09-16 |  9   |      I'm Blessed      |                       Charlie Wilson
Featuring T.I.                     |      <a href=https://charts-
static.billboard.com/img/2017/01/charlie-wilson-4cp-
imblessed-p3c-180x180.jpg>https://charts-
static.billboard.com/img/2017/01/charlie-wilson-4cp-
imblessed-p3c-180x180.jpg</a>      |       9        |       1        |
|  2023-09-16 |  10  |      Let It Whip      |                                Dazz
Band                               |            <a href=https://charts-
static.billboard.com/img/1982/03/dazz-band-s10-letitwhip-
dkw-180x180.jpg>https://charts-static.billboard.com/img/1982/03/dazz-
band-s10-letitwhip-dkw-180x180.jpg</a>           |       10       |       1
|
+-------------+------+-----------------------+--------------------------------
------------------------------------+-----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+----------------+----------------+
Truncated to displaylimit of 10.
```

3.2 Grader Setup

[14]:
```python
# Connecting the grader
# Just run the following cell, no further action is needed.
from data101_utils import GradingUtil
grading_util = GradingUtil("proj0")
grading_util.prepare_autograder()
```

4 The billboard Database

Billboard is a music industry publication which tracks the sales and distribution of music across many genres. Each week, they publish “charts” which aggregate the total sales, streams (e.g. via Spotify, Apple Music, or YouTube) as well as radio airplay in the US.

The most (in)famous chart is the Billboard Hot 100, widely regarded as the list of most popular songs in the US. For this assignment, you’ll be looking at data from the Hot 100 list, primarily in recent years — though the database we’ve provided to you goes all the way back to 1958!

Later questions will involve `JOIN`ing with data from the Billboard TikTok Top 50, a list of the songs that are the most popular on TikTok, factoring in views and user engagement.

Why might we make our own SQL database of something that's already available on a webpage? Well, we write queries and learn things about music or pop history that might be harder from just looking at a chart. But perhaps an alternate reason is that the Billboard website loads hundreds of data trackers, and dozens of megabytes of data on every webpage. Sometimes, it's nicer to just view data… in a database.

Below is a list of the relations (tables) in our database. - **hot_100**: Metadata for songs in the Billboard Hot 100 - **tiktok_top_50**: Metadata for songs in the Billboard TikTok Top 50

4.1 Data Schema

When approaching a new database, one of the most important things to do is understand the database schema. Remember that a database is a set of tables, which each contain their own schema. For this assignment, both the `hot_100` and `tiktok_top_50` tables contain the following schema.

```
CREATE TABLE hot_100 (
    week_ending TEXT,
    rank INTEGER,
    title TEXT,
    artist TEXT,
    image_url TEXT,
    peak_position INTEGER,
    weeks_on_chart INTEGER,
    UNIQUE (week_ending, rank)
);
```

- `week_ending`: The week that "chart" was published, as a `YYYY-MM-DD` string
- `rank`: The track's current position on the chart. (A rank of 1 is better than a rank of 10.)
- `title`: Track's title
- `artist`: Track's artist
- `image_url`: Image URL of the track's album cover
- `peak_position`: The track's peak position on the chart as of the chart date
- `weeks_on_chart`: The number of weeks the track has been or was on the chart up until that point

Note: In the case of Billboard charts, a song (or album's) `peak_position` is based on the idea that being #1 is the 'best'. Therefore, a *smaller* value for `peak_position` is better.

An aside: We've decided to use a `TEXT` data type for the `week_ending` date. Later we'll explore the `date` type. However, we can operate ordered comparisions on these strings, since '2024-01-01' comes before '2024-01-02' lexicographically.

4.2 Optional Fun

Call the `display_query` function passing in your SQL query result to view the results with a table that links to Apple Music and Spotify. An example usage of the `display_query` function can be found at the end of Question 0.

[15]:
```
# Some utilities to make exploration easier.
# NOTE: Make sure to write display_query(...);
# Use the ; to supress the notebook's default output.

import urllib.parse
from IPython.display import display, HTML

# Adjust pandas display options to better handle large DataFrames
pd.set_option('display.max_rows', 1000) # Adjust as needed
pd.set_option('display.max_seq_items', 1000)
pd.set_option('display.max_columns', None)  # Show all columns
pd.set_option('display.colheader_justify', 'left')

spotify_url = 'https://play.spotify.com/search/'
apple_music_url = 'https://music.apple.com/us/search?term='

# You probably don't need this function but you can use it make "fancy"
# datatables which are sortable and searchable. (But the can be a little slow␣
 ↪to load.)
def html_datatable(html):
    return display(HTML(f"""
    <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.
 ↪10.21/css/jquery.dataTables.css">
    <script type="text/javascript" charset="utf8" src="https://code.jquery.com/
 ↪jquery-3.5.1.js"></script>
    <script type="text/javascript" charset="utf8" src="https://cdn.datatables.
 ↪net/1.10.21/js/jquery.dataTables.js"></script>
    <script>
    $(document).ready(function() {{
        $('table.display').DataTable();
    }});
    </script>
    <div>{html}</div>
    """))

def basic_table(html):
    return display(HTML(f"""
    <div style="height: 300px; overflow-y: scroll; border: 1px solid #ccc;">
        {html}
    </div>
    """))

def display_query(result, show_search=True):
    """
    A fairly basic df to HTML-table display utilitiy.
    Includes the ability to give links to Spotify/Apple Music for easy␣
 ↪exploring and maximal time wastage.
```

```
    """
    # Display the DataFrame as a nicely formatted table with scroll bars
    df = result.DataFrame()
    if show_search and ('artist' in df.columns or 'title' in df.columns):
        song_query = lambda row: urllib.parse.quote(row.get("title", '') + " "␣
 ↪+ row.get("artist", ''))
        df['spotify'] = df.apply(lambda row: f'<a␣
 ↪href="{spotify_url}{song_query(row)}" target="_blank">Spotify</a>', axis=1)
        df['apple'] = df.apply(lambda row: f'<a␣
 ↪href="{apple_music_url}{song_query(row)}" target="_blank">Apple Music</a>',␣
 ↪axis=1)

    if 'image_url' in df.columns:
        # You could update each row to render the image_url as HTML...
        # e.g. <img src="{image_url}" alt="Song Cover Art" height=20>
        df = df.drop(columns='image_url')
    # Convert DataFrame to HTML;
    html_table = df.to_html(index=False, escape=False, classes="display")

    # Swap these two lines to see the fancy version of the output.
    # html_datatable(html_table)
    basic_table(html_table)

    print(f"DataFrame: {df.shape[0]} rows x {df.shape[1]} columns")
    return df;
```

5 Question 0 - Warmup

This question is given to you! Verify that your table loads correctly. Run each of the following queries, and you should see the right output.

Type your answer here, replacing this text.

[16]:
```
%%sql --save query_0_0 result_0_0 <<
SELECT COUNT(*) FROM hot_100
```

```
Running query in 'postgresql://localhost/billboard'
```

```
1 rows affected.
```

[17]:
```
# Do not delete/edit this cell!
# You must run this cell before running the autograder.
query_0_0 = %sqlcmd snippets query_0_0
grading_util.save_results("result_0_0", query_0_0, result_0_0)
result_0_0
```

[17]:
```
+---------+
| count  |
+---------+
| 344458 |
+---------+
```

[18]:
```
%%sql --save query_0_1 result_0_1 <<
SELECT * FROM hot_100 LIMIT 25
```

```
Running query in 'postgresql://localhost/billboard'
```

```
25 rows affected.
```

[19]:
```
# Do not delete/edit this cell!
# You must run this cell before running the autograder.
query_0_1 = %sqlcmd snippets query_0_1
grading_util.save_results("result_0_1", query_0_1, result_0_1)
result_0_1
```

[19]:
```
+-------------+------+--------------------------+-------------------------------
--------------+-----------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------------------------------------------------+-----
----------+----------------+
| week_ending | rank |          title           |                    artist
|
image_url
| peak_position | weeks_on_chart |
+-------------+------+--------------------------+-------------------------------
--------------+-----------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------------------------------------------------+-----
----------+----------------+
|  1958-08-04 |  1   |     Poor Little Fool     |                  Ricky Nelson
| <a href=https://www.billboard.com/wp-content/themes/vip/pmc-
billboard-2021/assets/public/lazyload-fallback.gif>https://www.billboard.com/wp-
content/themes/vip/pmc-billboard-2021/assets/public/lazyload-fallback.gif</a> |
1       |       1        |
|  1958-08-04 |  2   |         Patricia         |         Perez Prado And His
Orchestra        | <a href=https://www.billboard.com/wp-content/themes/vip/pmc-
billboard-2021/assets/public/lazyload-fallback.gif>https://www.billboard.com/wp-
content/themes/vip/pmc-billboard-2021/assets/public/lazyload-fallback.gif</a> |
2       |       1        |
|  1958-08-04 |  3   |      Splish Splash       |                  Bobby Darin
|                            <a href=https://charts-
static.billboard.com/img/1958/08/bobby-darin-hm6-180x180.jpg>https://charts-
static.billboard.com/img/1958/08/bobby-darin-hm6-180x180.jpg</a>
|       3       |       1        |
|  1958-08-04 |  4   |    Hard Headed Woman     |      Elvis Presley With The
Jordanaires     |                          <a href=https://charts-
static.billboard.com/img/1958/08/elvis-presley-0r3-180x180.jpg>https://charts-
static.billboard.com/img/1958/08/elvis-presley-0r3-180x180.jpg</a>
|       4       |       1        |
|  1958-08-04 |  5   |           When           |                 Kalin Twins
| <a href=https://www.billboard.com/wp-content/themes/vip/pmc-
billboard-2021/assets/public/lazyload-fallback.gif>https://www.billboard.com/wp-
content/themes/vip/pmc-billboard-2021/assets/public/lazyload-fallback.gif</a> |
5       |       1        |
|  1958-08-04 |  6   |      Rebel-'rouser       | Duane Eddy His Twangy Guitar
And The Rebels |                             <a href=https://charts-
static.billboard.com/img/1958/08/duane-eddy-vt9-180x180.jpg>https://charts-
static.billboard.com/img/1958/08/duane-eddy-vt9-180x180.jpg</a>
|       6       |       1        |
|  1958-08-04 |  7   |        Yakety Yak        |                 The Coasters
|                           <a href=https://charts-
static.billboard.com/img/1958/08/the-coasters-jiw-180x180.jpg>https://charts-
static.billboard.com/img/1958/08/the-coasters-jiw-180x180.jpg</a>
|       7       |       1        |
|  1958-08-04 |  8   |       My True Love       |                  Jack Scott
| <a href=https://www.billboard.com/wp-content/themes/vip/pmc-
billboard-2021/assets/public/lazyload-fallback.gif>https://www.billboard.com/wp-
content/themes/vip/pmc-billboard-2021/assets/public/lazyload-fallback.gif</a> |
8       |       1        |
|  1958-08-04 |  9   | Willie And The Hand Jive |             The Johnny Otis
Show            | <a href=https://www.billboard.com/wp-content/themes/vip/pmc-
billboard-2021/assets/public/lazyload-fallback.gif>https://www.billboard.com/wp-
content/themes/vip/pmc-billboard-2021/assets/public/lazyload-fallback.gif</a> |
9       |       1        |
|  1958-08-04 |  10  |          Fever           |                  Peggy Lee
|                              <a href=https://charts-
static.billboard.com/img/1958/08/peggy-lee-55r-180x180.jpg>https://charts-
static.billboard.com/img/1958/08/peggy-lee-55r-180x180.jpg</a>
|       10      |       1        |
+-------------+------+--------------------------+------------------------------
--------------+----------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------------------------------------------------+-----
----------+----------------+
```

[20]:
```
%%sql --save query_0_2 result_0_2 <<
SELECT * FROM hot_100 ORDER BY week_ending DESC LIMIT 30
```

```
Running query in 'postgresql://localhost/billboard'
```

```
30 rows affected.
```

```
[21]: # Do not delete/edit this cell!
      # You must run this cell before running the autograder.
      query_0_2 = %sqlcmd snippets query_0_2
      grading_util.save_results("result_0_2", query_0_2, result_0_2)
      result_0_2
```

```
[21]: +-------------+------+-------------------+-----------------------------------+--
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------------------+---------------+----------------
+
| week_ending | rank |       title       |               artist              |
image_url
| peak_position | weeks_on_chart |
+-------------+------+-------------------+-----------------------------------+--
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------------------+---------------+----------------
+
|  2024-08-10 | 100  |   Wanna Be Loved  |         The Red Clay Strays        |
<a href=https://charts-static.billboard.com/img/2024/06/theredclaystrays-xuz-
wannabeloved-eg6-180x180.jpg>https://charts-
static.billboard.com/img/2024/06/theredclaystrays-xuz-wannabeloved-
eg6-180x180.jpg</a> |      100      |       1        |
|  2024-08-10 |  99  |       Misses      |             Dominic Fike             |
<a href=https://charts-static.billboard.com/img/2019/03/dominicfike-
cvy-180x180.jpg>https://charts-static.billboard.com/img/2019/03/dominicfike-
cvy-180x180.jpg</a>                          |       99      |       2        |
|  2024-08-10 |  98  |       Alibi       | Sevdaliza, Pabllo Vittar & Yseult |
<a href=https://charts-static.billboard.com/img/2024/07/sevdaliza-jrq-alibi-
bsr-180x180.jpg>https://charts-static.billboard.com/img/2024/07/sevdaliza-jrq-
alibi-bsr-180x180.jpg</a>                    |       95      |       2        |
|  2024-08-10 |  97  |    Better Days    |  Zach Bryan Featuring John Mayer  |
<a href=https://charts-static.billboard.com/img/2019/09/zachbryan-
de1-180x180.jpg>https://charts-static.billboard.com/img/2019/09/zachbryan-
de1-180x180.jpg</a>                              |       46      |       4        |
|  2024-08-10 |  96  |        Okay       |                 JT                 |
<a href=https://charts-static.billboard.com/img/2024/05/jt-000-okay-
ny4-180x180.jpg>https://charts-static.billboard.com/img/2024/05/jt-000-okay-
ny4-180x180.jpg</a>                          |       72      |       5        |
|  2024-08-10 |  95  | Wine Into Whiskey |           Tucker Wetmore           |
<a href=https://charts-static.billboard.com/img/2024/03/tuckerwetmore-cvz-
wineintowhiskey-n1n-180x180.jpg>https://charts-
static.billboard.com/img/2024/03/tuckerwetmore-cvz-
wineintowhiskey-n1n-180x180.jpg</a> |       68      |       18       |
|  2024-08-10 |  94  |       Linger      |             Royel Otis             |
<a href=https://charts-static.billboard.com/img/2024/05/royelotis-000-linger-
```

```
3qn-180x180.jpg>https://charts-
static.billboard.com/img/2024/05/royelotis-000-linger-3qn-180x180.jpg</a>
|       94      |       1        |
|  2024-08-10 |  93  |       Girls       |           The Kid LAROI           |
<a href=https://charts-static.billboard.com/img/2024/07/thekidlaroi-qev-girls-
eaj-180x180.jpg>https://charts-static.billboard.com/img/2024/07/thekidlaroi-qev-
girls-eaj-180x180.jpg</a>             |       51      |       5        |
|  2024-08-10 |  92  |      We Ride      |            Bryan Martin           |
<a href=https://charts-static.billboard.com/img/2022/11/bryanmartin-a12-weride-
asg-180x180.jpg>https://charts-
static.billboard.com/img/2022/11/bryanmartin-a12-weride-asg-180x180.jpg</a>
|       56      |       17       |
|  2024-08-10 |  91  |        Nel        |           Fuerza Regida           |
<a href=https://charts-
static.billboard.com/img/2018/06/fuerzaregida-o2t-180x180.jpg>https://charts-
static.billboard.com/img/2018/06/fuerzaregida-o2t-180x180.jpg</a>
|       91      |       1        |
+-------------+------+-------------------+-----------------------------------+--
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------------------+---------------+----------------
+
```

[22]:
```
display_query(result_0_2);
```

```
<IPython.core.display.HTML object>

DataFrame: 30 rows x 8 columns
```

[23]:
```
grader.check("q0")
```

[23]:
```
q0 results: All test cases passed!
```

6 Question 1

Select all song titles and artists on the Billboard Hot 100 chart in August 2024 which have ranking of 10 or better, sorted alphabetically by artist name.

Notes: - The column `week_ending` is a **string**. it is represented `YYYY-MM-DD`. Think back to string comparisons in Data 100. (Later in this class we will explore Postgres date types.) - Rank 1 is the best, so make sure you are using the right comparison operator! - Remember that a unique pair of song and artist can appear multiple times on the chart (for example, if the song charts for multiple weeks). Make sure your output only includes each song once.

[24]:
```
%%sql  --save query_1 result_1 <<
SELECT DISTINCT title, artist
FROM hot_100
WHERE rank <= 10
```

```
AND week_ending LIKE '2024-08-%'
GROUP BY title, artist
ORDER BY artist ASC;
```

```
Running query in 'postgresql://localhost/billboard'
```

```
11 rows affected.
```

[25]:
```
# Do not delete/edit this cell!
# You must run this cell before running the autograder.
query_1 = %sqlcmd snippets query_1
grading_util.save_results("result_1", query_1, result_1)
result_1
```

[25]:
```
+----------------------+-------------------------------------+
|        title         |                artist               |
+----------------------+-------------------------------------+
|   Beautiful Things   |             Benson Boone            |
|  Birds Of A Feather  |            Billie Eilish            |
|   Good Luck, Babe!   |            Chappell Roan            |
|      Too Sweet       |                Hozier               |
|     Not Like Us      |            Kendrick Lamar           |
|   I Had Some Help    | Post Malone Featuring Morgan Wallen |
|       Espresso       |          Sabrina Carpenter          |
| Please Please Please |          Sabrina Carpenter          |
|  A Bar Song (Tipsy)  |              Shaboozey              |
|     Lose Control     |             Teddy Swims             |
+----------------------+-------------------------------------+
```

[26]:
```
display_query(result_1);
```

```
<IPython.core.display.HTML object>
```

```
DataFrame: 11 rows x 4 columns
```

[27]:
```
grader.check("q1")
```

[27]:
```
q1 results: All test cases passed!
```

7 Question 2 - Longest Time Spent on the Charts

Find all song titles, their artist, and the total number of weeks the song spent on the Billboard Hot 100.

Only include songs that have spent >5 weeks on the chart (call this column `total_weeks_on_chart`).

Sort by the total number of weeks the song spent on the chart, descending order.

Notes: - Similar to the previous question, make sure each song appears once - The `weeks_on_chart` column is the number of weeks on the chart **as of that week**. How can we get the total number of weeks?

```
[28]: %%sql --save query_2 result_2 <<
      SELECT DISTINCT title, artist, MAX(weeks_on_chart) AS total_weeks_on_chart
      FROM hot_100
      WHERE weeks_on_chart > 5
      GROUP BY title, artist
      ORDER BY total_weeks_on_chart DESC;
```

```
Running query in 'postgresql://localhost/billboard'

21679 rows affected.
```

```
[29]: # Do not delete/edit this cell!
      # You must run this cell before running the autograder.
      query_2 = %sqlcmd snippets query_2
      grading_util.save_results("result_2", query_2, result_2)
      result_2
```

```
[29]: +-----------------+----------------------------+----------------------+
      |      title      |           artist           | total_weeks_on_chart |
      +-----------------+----------------------------+----------------------+
      |    Heat Waves   |       Glass Animals        |          91          |
      | Blinding Lights |         The Weeknd         |          90          |
      |   Radioactive   |      Imagine Dragons       |          87          |
      |       Sail      |         AWOLNATION         |          79          |
      |    Levitating   |          Dua Lipa          |          77          |
      |     I'm Yours   |         Jason Mraz         |          76          |
      |      Snooze     |            SZA             |          70          |
      |   How Do I Live |        LeAnn Rimes         |          69          |
      | Save Your Tears | The Weeknd & Ariana Grande |          69          |
      |  Counting Stars |        OneRepublic         |          68          |
      +-----------------+----------------------------+----------------------+
```

```
[30]: display_query(result_2);
```

```
<IPython.core.display.HTML object>

DataFrame: 21679 rows x 5 columns
```

```
[31]: grader.check("q2")
```

```
[31]: q2 results: All test cases passed!
```

8 Question 3 - Longest Top 10 Hits of the 1970's

Find all song titles, their corresponding artist, the total number of weeks the song spent on the Billboard Hot 100 (call this column `total_weeks_on_chart`), and its (highest) peak position (call this column `highest_peak_position`).

Only include songs… - with a highest peak position in the top 10 (e.g. position 1-10, inclusive) AND - that have spent >10 weeks on the chart AND - that charted from Jan. 1, 1970 - Dec. 31 1979, inclusive

Sort by `total_weeks_on_chart`, descending order.

Notes: - The same notes from Question 2 apply to this question. - Remember "highest" position refers to lower numbers (e.g. position of 5 is higher than position of 10)

[32]:
```
%%sql  --save query_3 result_3  <<
SELECT DISTINCT title, artist, MAX(weeks_on_chart) AS total_weeks_on_chart,␣
  ↪MIN(peak_position) AS highest_peak_position
FROM hot_100
WHERE peak_position BETWEEN 1 AND 10
AND weeks_on_chart > 10
AND week_ending >= '1970-01-01'
AND week_ending <= '1979-12-31'
GROUP BY title, artist
ORDER BY total_weeks_on_chart DESC;
```

```
Running query in 'postgresql://localhost/billboard'

933 rows affected.
```

[33]:
```
# Do not delete/edit this cell!
# You must run this cell before running the autograder.
query_3 = %sqlcmd snippets query_3
grading_util.save_results("result_3", query_3, result_3)
result_3
```

[33]:
```
+--------------------------------------+-------------------------------------+----
------------------+-----------------------+
|                title                 |               artist                |
total_weeks_on_chart | highest_peak_position |
+--------------------------------------+-------------------------------------+----
------------------+-----------------------+
|              I Go Crazy              |              Paul Davis             |
40          |           7           |
|         How Deep Is Your Love        |               Bee Gees              |
33          |           1           |
|            Baby Come Back            |                Player               |
32          |           1           |
|               Feelings               |             Morris Albert           |
32          |           6           |
|        Hot Child In The City        |           Nick Gilder              |
31          |           1            |
|  I Just Want To Be Your Everything  |            Andy Gibb               |
31          |           1            |
| I Love The Nightlife (Disco 'round) |          Alicia Bridges            |
31          |           5            |
|              Dream On               |            Aerosmith               |
29          |           6            |
|     (Love Is) Thicker Than Water    |            Andy Gibb               |
29          |           1            |
|         A Fifth Of Beethoven        | Walter Murphy & The Big Apple Band |
28          |           1            |
+-------------------------------------+------------------------------------+----
------------------+----------------------+
```

[34]:
```
display_query(result_3);
```

```
<IPython.core.display.HTML object>

DataFrame: 933 rows x 6 columns
```

[35]:
```
grader.check("q3")
```

[35]:
```
q3 results: All test cases passed!
```

9 Question 4 – Top Artist's By Number 1 Hits

Find the top 50 *artists* by total number of weeks any of their songs spent at rank 1 (call this column `total_weeks_at_number_one`).

Sort by `total_weeks_at_number_one`, descending order.

Example: Suppose artist Jane Doe has 3 songs A, B, and C that have spent 10, 20, and 30 weeks at rank 1, respectively. Then the expected output would be: Jane Doe, 60 because 10 + 20 + 30 = 60 weeks total.

Notes: - Remember that an artist can have multiple songs that have reached rank 1 at some point - Remember that every row in the `hot_100` table represents 1 week on the chart, so be careful about how you aggregate the rows. Which aggregation function would work best here?

[36]:
```
%%sql  --save query_4 result_4  <<
SELECT  artist, COUNT(weeks_on_chart) AS total_weeks_at_number_one
FROM hot_100
WHERE rank = 1
GROUP BY artist
ORDER BY total_weeks_at_number_one DESC
LIMIT 50;
```

```
Running query in 'postgresql://localhost/billboard'
```

```
50 rows affected.
```

[37]:
```
# Do not delete/edit this cell!
# You must run this cell before running the autograder.
query_4 = %sqlcmd snippets query_4
grading_util.save_results("result_4", query_4, result_4)
result_4
```

[37]:
```
+----------------------+---------------------------+
|        artist        | total_weeks_at_number_one |
+----------------------+---------------------------+
|     Mariah Carey     |             74            |
|     The Beatles      |             54            |
|     Boyz II Men      |             34            |
|        Adele         |             34            |
|     Taylor Swift     |             33            |
|       Madonna        |             32            |
|   Whitney Houston    |             31            |
|        Drake         |             31            |
|   Michael Jackson    |             30            |
| The Black Eyed Peas  |             28            |
+----------------------+---------------------------+
```

[38]:
```
display_query(result_4);
```

```
<IPython.core.display.HTML object>

DataFrame: 50 rows x 4 columns
```

[39]:
```
grader.check("q4")
```

[39]:
```
q4 results: All test cases passed!
```

10 Question 5 – What's Trending on TikTok?

10.1 Question 5a – Popular Everywhere

Find the songs that were on both the Billboard Hot 100 and TikTok Top 50 charts in the same week.

Sort alphabetically by song title.

Notes: - Remember that the same song can appear in multiple rows of the chart (for example, if the song charts for multiple weeks) - A song is considered the "same" as another song if the song title and artist match

[40]:
```
%%sql --save query_5_0 result_5_0 <<

SELECT DISTINCT hot_100.title, hot_100.artist
```

```
FROM hot_100
JOIN tiktok_top_50
ON hot_100.title= tiktok_top_50.title
AND hot_100.artist= tiktok_top_50.artist
AND hot_100.week_ending= tiktok_top_50.week_ending
ORDER BY hot_100.title;
```

```
Running query in 'postgresql://localhost/billboard'

128 rows affected.
```

```
[41]: # Do not delete/edit this cell!
      # You must run this cell before running the autograder.
      query_5_0 = %sqlcmd snippets query_5_0
      grading_util.save_results("result_5_0", query_5_0, result_5_0)
      result_5_0
```

```
[41]: +----------------------------------+------------------------------------+
      |              title               |               artist               |
      +----------------------------------+------------------------------------+
      |                28                |             Zach Bryan             |
      |         A Bar Song (Tipsy)       |             Shaboozey              |
      |          Act II: Date @ 8        |       4Batz Featuring Drake        |
      |            Agora Hills           |              Doja Cat              |
      |      Ain't No Love In Oklahoma   |             Luke Combs             |
      |               Alibi              | Sevdaliza, Pabllo Vittar & Yseult  |
      |         All-American Bitch       |           Olivia Rodrigo           |
      | All I Want For Christmas Is You  |            Mariah Carey            |
      |             Anti-Hero            |            Taylor Swift            |
      |               Apple              |             Charli xcx             |
      +----------------------------------+------------------------------------+
```

```
[42]: display_query(result_5_0);
```

```
<IPython.core.display.HTML object>

DataFrame: 128 rows x 4 columns
```

```
[43]: grader.check("q5a")
```

```
[43]: q5a results: All test cases passed!
```

10.2 Question 5b – Brat Summer

Write a query that selects:

- `week_ending`
- Billboard Hot 100 peak position for that week (call this column `hot_100_peak_pos`)
- TikTok Top 50 peak position for that week (call this column `tiktok_50_peak_pos`)

- Song title

Only include songs where the artist is `'Charli xcx'` (case sensitive).

Sort by `week_ending`, ascending order.

[44]:
```
%%sql --save query_5_1 result_5_1 <<
SELECT DISTINCT hot_100.week_ending, hot_100.peak_position AS  hot_100_peak_pos,
tiktok_top_50.peak_position AS tiktok_50_peak_pos, hot_100.title
FROM hot_100
JOIN tiktok_top_50
ON hot_100.title=  tiktok_top_50.title
AND hot_100.week_ending=  tiktok_top_50.week_ending
WHERE hot_100.artist = 'Charli xcx'
ORDER BY week_ending ASC;
```

```
Running query in 'postgresql://localhost/billboard'

2 rows affected.
```

[45]:
```
# Do not delete/edit this cell!
# You must run this cell before running the autograder.
query_5_1 = %sqlcmd snippets query_5_1
grading_util.save_results("result_5_1", query_5_1, result_5_1)
result_5_1
```

[45]:
```
+-------------+-----------------+--------------------+-------+
| week_ending | hot_100_peak_pos | tiktok_50_peak_pos | title |
+-------------+-----------------+--------------------+-------+
|  2024-08-03 |        81       |         4          | Apple |
|  2024-08-10 |        66       |         3          | Apple |
+-------------+-----------------+--------------------+-------+
```

[46]:
```
display_query(result_5_1);
```

```
<IPython.core.display.HTML object>

DataFrame: 2 rows x 6 columns
```

[47]:
```
grader.check("q5b")
```

[47]:
```
q5b results: All test cases passed!
```

11 Question 6 – Reminders!

Fill out the pre-semester form. Then in the code cell below, set `secret_word` to the secret word at the end of the form.

[48]:
```
secret_word = "schema"
```

12 Congratulations! You have finished Project 0.

We hope you found something interesting to listen to :D

Here is an optional but interesting video by Vox related to viral TikTok songs: We tracked what happens after TikTok songs go viral

Final Question What's something fun/weird/interesting you discovered while exploring some data in this assignment? (Totall feel free to leave this blank…)

> …put you answer in this cell.

12.1 Acknowledgements

This assignment was inspired by Chris Molanphy's podcast "Hit Parade" which dives into the history of popular music in the US. Former TA Allen Guo's billboard.py library made it possible to easily extract the data from Billboard's ad-riddled website.

If you'd like to explore any of the other charts, or perhaps query more recent data, you should give it a try.

12.2 Submission

Run the following cell to zip and download the results of your queries. You will also need to run the export cell at the end of the notebook.

Please save your notebook before exporting (this is a good time to do it!) Otherwise, we may not be able to register your written responses.

For your submission on Gradescope, you will need to submit the `proj0.zip` file generated by the export cell. Please ensure that your submission includes `proj0.pdf`, `proj0.ipynb`, and `results.zip`.

Please ensure that public tests pass upon submission. It is your responsibility to wait until the autograder finishes running. We will not be accepting regrade requests for submission issues.

Common submission issues: You MUST submit the generated zip file to the autograder. However, Safari is known to automatically unzip files upon downloading. You can fix this by going into Safari preferences, and deselect the box with the text "Open safe files after downloading" under the "General" tab. If you experience issues with downloading via clicking on the link, you can also navigate to the project 0 directory within JupyterHub (remove `proj0.ipynb` from the url), and manually download the generated zip files. Please post on Ed if you encounter any other submission issues.

Run the following cell to zip and download the results of your queries. You will also need to run the export cell at the end of the notebook.

[49]:
```
grading_util.prepare_submission_and_cleanup()
```

To double-check your work, the cell below will rerun all of the autograder tests.

```
[50]: grader.check_all()
```

```
[50]: q0 results: All test cases passed!

      q1 results: All test cases passed!

      q2 results: All test cases passed!

      q3 results: All test cases passed!

      q4 results: All test cases passed!

      q5a results: All test cases passed!

      q5b results: All test cases passed!

      q6 results: All test cases passed!
```

12.3 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

```
[51]: # Save your notebook first, then run this cell to export your submission.
      grader.export(pdf=False, files=['results.zip'])
```

```
<IPython.core.display.HTML object>
```

```
[ ]:
```