CS W186 Introduction to Database Systems
Spring 2020 Josh Hug, Michael Ball

DIS 1

1 Single-Table SQL

Write SQL queries to accomplish each task below. You will not need to join any tables. Assume you have access to tables with the following schemas, where each primary key is in `ALL_CAPS`:

- `Songs(SONG_ID, song_name, album_id, weeks_in_top_40)`
- `Artists(ARTIST_ID, artist_name, first_yr_active)`
- `Albums(ALBUM_ID, album_name, artist_id, yr_released, genre)`

Solution:

(a) Find the 5 songs that spent the fewest weeks in the top 40, ordered from least to most. Break ties by song name in alphabetical order.

```
SELECT song_name
FROM Songs
ORDER BY weeks_in_top_40 ASC, song_name ASC
LIMIT 5;
```

(b) Find the name and the first year active for every artist whose name starts with the letter 'B'.

```
SELECT artist_name, first_yr_active
FROM Artists
WHERE artist_name LIKE 'B%';
```

(c) Find the total number of albums released per genre.

```
SELECT genre, COUNT(album_id)
FROM Albums
GROUP BY genre;
```

(d) Find the total number of albums released per genre. Don't include genres with a count less than 10.


```
SELECT genre, COUNT(*)
FROM Albums
GROUP BY genre
HAVING COUNT(*) >= 10;
```

(e) Find the genre for which the most albums were released in the year 2000. Assume there are no ties.

```
SELECT genre
FROM albums
WHERE yr_released = 2000
GROUP BY genre
ORDER BY COUNT(*) DESC
LIMIT 1;
```

2 Multi-Table SQL

Write SQL queries to accomplish each task below. Use the same tables from the previous question (copied from the front page). You will need to use joins.

- `Songs(SONG_ID, song_name, album_id, weeks_in_top_40)`
- `Artists(ARTIST_ID, artist_name, first_yr_active)`
- `Albums(ALBUM_ID, album_name, artist_id, yr_released, genre)`

Solution:

(a) Find the names of all artists who released a 'country' genre album in 2020.

```
SELECT artist_name
FROM Artists INNER JOIN Albums
    ON Artists.ARTIST_ID = Albums.artist_id
WHERE genre = 'country' AND yr_released = 2020
GROUP BY Artists.artist_id, artist_name;
```

or

```
SELECT DISTINCT artist_name
FROM Artists, Albums
WHERE Artists.ARTIST_ID = Albums.artist_id AND genre = 'country';
    AND yr_released = 2020
GROUP BY Artists.artist_id, artist_name;
```


(b) Find the name of the album with the song that spend the most weeks in the top 40. Assume there is only one such song.

```
SELECT album_name
FROM Songs INNER JOIN Albums
    ON Songs.album_id = Albums.album_id
ORDER BY weeks_in_top_40 DESC
LIMIT 1;
```

or

```
SELECT album_name
FROM Songs, Albums
WHERE Songs.album_id = Albums.album_id
ORDER BY weeks_in_top_40 DESC
LIMIT 1;
```

(c) Find the the artist name and the most weeks one of their songs spent in the top 40 for each artist. Include artists that have not released an album.

```
SELECT artist_name, MAX(weeks_in_top_40)
FROM Artists LEFT JOIN
    (Songs INNER JOIN Albums ON Songs.album_id = Albums.album_id)
ON Artists.artist_id = Albums.artist_id
GROUP BY Artists.artist_id, artist_name
```

We include artist_name in the GROUP BY clause because if a GROUP BY clause is used, as a general rule, all columns in a SELECT clause should either be in a GROUP BY clause or an aggregation function. Otherwise, this may lead to indeterminate behavior since the number of rows may not be the same for each column. (You can technically omit from the GROUP BY columns that are functionally dependent on other columns in the GROUP BY clause, but this exception rarely shows up and you should follow the rule as much as possible.)