Assignment 1

Group 1

1. Display each beer's name and style name. A beer should be displayed regardless of whether a style name exists or not.

```
SELECT
   beer_name,
   style_name
FROM
   beers
   LEFT OUTER JOIN styles
      ON ( beers.style_id = styles.style_id );
```

2. Display each beer's name, category name, color example, and style name, for all beers that have values for category name, color example, and style name.

```
SELECT
   beer_name,
   category_name,
   examples,
   style_name
FROM
   beers
   INNER JOIN categories
      ON ( beers.cat_id = categories.category_id )
   INNER JOIN beerdb.colors
      ON ( beers.srm = colors.lovibond_srm )
   INNER JOIN beerdb.styles
      ON ( beers.style_id = styles.style_id );
```

3. Display each brewer's name along with the minimum, maximum, and average alcohol by volume (ABV) of its beers. Exclude any beers with an ABV of zero. Show the brewers with the highest average ABV first.

```
SELECT
   brewer.name AS name,
   MIN(beer.abv) AS min_abv,
   MAX(beer.abv) AS max_abv,
   round(AVG(beer.abv), 2) AS avg_abv
FROM
   breweries brewer
   INNER JOIN beers beer
```

```
      ON ( brewer.brewery_id = beer.brewery_id )
WHERE
   beer.abv > 0
GROUP BY
   brewer.name
ORDER BY
   avg_abv DESC;
```

4. Find which cities would be good for hosting microbrewery tours. A city must have at least 10 breweries to be considered. Display the city's name as well as how many breweries are in the city. Show cities with the most breweries first.

```
SELECT
   city,
   COUNT(name)
FROM
   breweries
WHERE
   city IS NOT NULL
GROUP BY
   city
HAVING
   COUNT(name) >= 10
ORDER BY
   COUNT(name) DESC;
```

5. Display all beer names that (1) belong to a category with a name containing “Lager” somewhere in the name and (2) have an alcohol by volume (ABV) of eight or greater. Show the beer names in alphabetical order.

```
SELECT
   beer_name
FROM
   beerdb.beers be
   INNER JOIN db2slate.beerdb_categories ca
      ON be.cat_id = ca.category_id
WHERE
   ( abv >= 8.0 )
   AND ca.category_name LIKE '%Lager%'
ORDER BY
   be.beer_name;
```

6. Display the name of all movies that have an IMDB rating of at least 8.0, with more than 100,000 IMDB votes, and were released from 2007 to 2013. Show the movies with the highest IMDB ratings first.

```
SELECT
   film_title
FROM
   relmdb.movies
WHERE
   imdb_rating >= 8.0
   AND imdb_votes > 100000
   AND film_year BETWEEN 2007 AND 2013
ORDER BY
   imdb_rating;
```

7. Display each movie's title and total gross, where total gross is USA gross and worldwide gross combined. Exclude any movies that do not have values for either USA gross or worldwide gross. Show the highest grossing movies first.

```
SELECT
   film_title,
   to_number(usa_gross + worldwide_gross) AS total_gross
FROM
   movies
WHERE
   ( usa_gross IS NOT NULL
     AND worldwide_gross IS NOT NULL )
ORDER BY
   total_gross DESC;
```

8. Display the titles of any movies where Tom Hanks or Tim Allen were cast members. Each movie title should be shown only once.

```
SELECT UNIQUE
   film_title
FROM
   relmdb.casts
   INNER JOIN movies
      ON CASTS.film_id = movies.film_id
WHERE
   cast_member IN (
      'Tom Hanks',
      'Tim Allen'
```

);

Group 2

10. Label the strength of a beer based on its ABV. For each beer display the beer's name, ABV, and a textual label describing the strength of the beer. The label should be "Very High" for an ABV more than 10, "High" for an ABV of 6 to 10, "Average" for an ABV of 3 to 6, and "Low" for an ABV less than 3. Show the records by beer name.

```
SELECT
   beer_name,
   abv,
   CASE
      WHEN abv > 10 THEN
         'Very High'
      WHEN abv BETWEEN 6 AND 10 THEN
         'High'
      WHEN abv BETWEEN 3 AND 6 THEN
         'Average'
      ELSE
         'Low'
   END AS "Strength"
FROM
   beerdb.beers
ORDER BY
   beer_name;
```

11. Find all breweries that specialize in a particular beer style. A brewer is considered specialized if they produce at least 10 beers from the same style. Show the brewer's name, style name, and how many beers the brewer makes of that style. Display the records by style name first and then by breweries with the most beers within that style.

```
SELECT
   br.name,
   st.style_name,
   COUNT(st.style_name)
FROM
   beerdb.beers be
   INNER JOIN beerdb.breweries br
      ON be.brewery_id = br.brewery_id
   INNER JOIN beerdb.styles st
      ON be.style_id = st.style_id
GROUP BY
   br.name,
   st.style_name
HAVING
```

```
   COUNT(st.style_name) >= 10
ORDER BY
   st.style_name,
   COUNT(st.style_name);
```

12. Display each brewer's name and how many beers they have associated with their brewery. Only include brewers that are located outside the United States and have more than the average number of beers from all breweries (excluding itself when calculating the average). Show the brewers with the most beers first. If there is a tie in number of beers, then sort by the brewers' names.

```
SELECT
        brewer.name AS brewery_name,
        COUNT(beer.beer_id) AS num_beers
FROM
        breweries   brewer
        INNER JOIN beers     beer ON ( brewer.brewery_id = beer.brewery_id )
WHERE
        brewer.country != 'United States'
GROUP BY
        brewer.name
HAVING
        COUNT(beer.beer_id) > (
        SELECT
        AVG(snum_beers) AS avg_beers
        FROM
        (
                SELECT
                COUNT(sbeer.beer_id) AS snum_beers
                FROM
                breweries   sbrewer
                INNER JOIN beers     sbeer ON ( sbrewer.brewery_id = sbeer.brewery_id )
                WHERE
                sbrewer.name != brewer.name
                GROUP BY
                sbrewer.name
        )
        )
ORDER BY
        num_beers DESC,
        brewery_name ASC;
```

13. For each movie display its movie title, year, and how many cast members were a part of the movie. Exclude movies with five or fewer cast members. Display movies with the most cast members first, followed by movie year and title.

```
SELECT
   film_title,
   film_year,
   COUNT(cast_member)
FROM
   relmdb.movies mv
   INNER JOIN relmdb.casts cs
      ON cs.film_id = mv.film_id
GROUP BY
   film_year,
   film_title
HAVING
   COUNT(cs.cast_member) >= 6
ORDER BY
   COUNT(cs.cast_member) DESC,
   film_year,
   film_title;
```

14. For each genre display the total number of films, average fan rating, and average USA gross. A genre should only be shown if it has at least five films. Any film without a USA gross should be excluded. A film should be included regardless of whether any fans have rated the film. Show the results by genre.

```
SELECT DISTINCT
   genre,
   COUNT(*),
   AVG(IMDB_RATING),
   AVG(USA_GROSS)
FROM
   movies
   INNER JOIN genres
      ON (MOVIES.FILM_ID = genres.film_id )
WHERE
   usa_gross IS NOT NULL
GROUP BY
   genre
HAVING
   COUNT(genre) >= 5
ORDER BY
```

```
   genre;
```

15. Find the average budget for all films from a director with at least one movie in the top 25 IMDB ranked films. Show the director with the highest average budget first.

```
SELECT
        directors_table.director AS director_name,
        round(AVG(movies_table.budget), 0) AS avg_budget
FROM
        relmdb.movies   movies_table
        INNER JOIN directors directors_table
           ON ( movies_table.film_id = directors_table.film_id )
WHERE
        directors_table.director IN (
        SELECT
        sub_directors_table.director AS sub_director_name
        FROM
        directors sub_directors_table
        INNER JOIN relmdb.movies sub_movies_table
           ON ( sub_directors_table.film_id = sub_movies_table.film_id )
        WHERE
        sub_movies_table.imdb_rank <= 25
        GROUP BY
        sub_directors_table.director
        )
        AND movies_table.budget IS NOT NULL
GROUP BY
        directors_table.director
ORDER BY
   avg_budget DESC;
```

16. Find all duplicate fans. A fan is considered duplicate if they have the same first name, last name, city, state, zip, and birth date

```
SELECT
   fname,
   lname,
   city,
   state,
   zip,
   birth_day,
   COUNT(*)
FROM
   fans
```

```
GROUP BY
   fname,
   lname,
   city,
   state,
   zip,
   birth_day
HAVING
   COUNT(*) > 1;
```

17. We believe there may be erroneous data in the movie database. To help uncover unusual records for manual review, write a query that finds all actors/actresses with a career spanning 60 years or more. Display each actor's name, how many films they worked on, the year of the earliest and latest film they worked on, and the number of years the actor was active in the film industry (assume all years between the first and last film were active years). Display actors with the longest career first.

```
SELECT
   cast_member,
   COUNT(film_title),
   MAX(RELMDB.MOVIES.film_year),
   MIN(film_year),
   MAX(RELMDB.MOVIES.FILM_YEAR) - MIN(film_year)
FROM
   relmdb.casts
   INNER JOIN relmdb.movies
      ON casts.film_id = movies.film_id
GROUP BY
   cast_member
HAVING
   MAX(relmdb.movies.film_year) - MIN(film_year) >= 60
ORDER BY
   MAX(relmdb.movies.film_year) - MIN(film_year) DESC;
```

18. The movies database has two tables that contain data on fans (FANS_OLD and FANS). Due to a bug in our application, fans may have been entered into the old fans table rather then the new table. Find all fans that exist in the old fans table but not the new table. Use only the first and last name when comparing fans between the two tables.

```
( SELECT
   fname,
   lname
FROM
```

```
    relmdb.fans_old
)
MINUS
( SELECT
    fname,
    lname
FROM
    relmdb.fans
);
```

Group 3

19. Assign breweries to groups based on the number of beers they brew. Display the brewery ID, name, number of beers they brew, and group number for each brewery. The group number should range from 1 to 4, with group 1 representing the top 25% of breweries (in terms of number of beers), group 2 representing the next 25% of breweries, group 3 the next 25%, and group 4 for the last 25%. Breweries with the most beers should be shown first. In the case of a tie, show breweries by brewery ID (lowest to highest).

```
SELECT
   breweries.brewery_id,
   name,
   COUNT(beer_name),
   NTILE(4) OVER(
      ORDER BY
         COUNT(beer_name) DESC,
         breweries.brewery_id
   ) rating_quartile
FROM
   breweries
   INNER JOIN beers
      ON ( breweries.brewery_id = beers.brewery_id )
GROUP BY
   name,
   breweries.brewery_id;
```

20. Rank beers in descending order by their alcohol by volume (ABV) content. Only consider beers with an ABV greater than zero. Display the rank number, beer name, and ABV for all beers ranked 1-10. Do not leave any gaps in the ranking sequence when there are ties (e.g., 1, 2, 2, 2, 3, 4, 4, 5). (Hint: derived tables may help with this query.)

```
SELECT
   rank_number,
   beer_name,
   abv
FROM
   (
      SELECT
         DENSE_RANK() OVER(
            ORDER BY
               abv DESC
         ) rank_number,
         beer_name,
         abv
```

```
      FROM
         beers
      WHERE
         abv > 0
   )
WHERE
   rank_number <= 10;
```

21. Display the film title, film year and worldwide gross for all movies directed by Christopher Nolan that have a worldwide gross greater than zero. In addition, each row should contain the cumulative worldwide gross (current row's worldwide gross plus the sum of all previous rows' worldwide gross). Records should be sorted in ascending order by film year.

```
SELECT
   relmdb.movies.film_title,
   relmdb.movies.film_year,
   relmdb.movies.worldwide_gross,
   SUM(worldwide_gross) OVER(
      ORDER BY
         relmdb.movies.film_year
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
   ) cum_sum
FROM
   relmdb.directors
   INNER JOIN relmdb.movies
      ON directors.film_id = movies.film_id
WHERE
   director = 'Christopher Nolan'
   AND relmdb.movies.worldwide_gross >= 0
ORDER BY
   relmdb.movies.film_year;
```

Interesting Queries

1. Selected fan_id and movie details from the database of those fans who are most frequent raters and criteria is those who have rated more than 20 movies.

```
SELECT
   fan_id,
   film_title,
   film_year,
   mpaa_rating,
   imdb_rank,
   runtime
FROM
   fan_ratings
   INNER JOIN movies
      ON ( fan_ratings.film_id = movies.film_id )
WHERE
   fan_id IN (
      SELECT
         fan_id
      FROM
         (
            SELECT DISTINCT
               fan_id,
               COUNT(fan_id)
            FROM
               fan_ratings
            GROUP BY
               fan_id
            HAVING
               COUNT(fan_id) > 20
         )
   )
ORDER BY
   fan_id ASC;
```

2. Query to select fans with their favourite genres. A fan has a favourite genres if they have watched movies with same genres 3 or more times.

```
SELECT DISTINCT
   genre,
   COUNT(genre),
   fan_id,
```

```
    fname,
    lname
FROM
    (
        SELECT
            fans.fan_id,
            fname,
            lname,
            fan_ratings.film_id,
            genres.genre
        FROM
            fans
            INNER JOIN fan_ratings
                ON ( fans.fan_id = fan_ratings.fan_id )
            INNER JOIN genres
                ON ( fan_ratings.film_id = genres.film_id )
        WHERE
            fans.fan_id IN (
                SELECT
                    fans.fan_id
                FROM
                    fans
            )
    )
GROUP BY
    genre,
    fan_id,
    fname,
    lname
HAVING
    COUNT(genre) >= 3
ORDER BY
    fan_id asc;
```

3. We want to know which movies to watch next weekend and who is in them. Select movies in the top 10 IMDB ranking along with each cast member.

```
SELECT
    imdb_rank,
    film_title,
    cast_member
FROM
    relmdb.movies mv
```

```
    INNER JOIN relmdb.casts cs
        ON cs.film_id = mv.film_id
WHERE
    mv.imdb_rank <= 10;
```

** Bonus - Interesting Query From Cheat Sheet - List all the breweries in Colorado, along with the number of beers they produce. Sort the results in descending order by the number of beers, so the most prolific breweries appear first.

```
SELECT
    brewer.name AS brewery_name,
    COUNT(beer.beer_id) AS num_beers
FROM
    breweries brewer
    INNER JOIN beers beer
        ON ( brewer.brewery_id = beer.brewery_id )
WHERE
    state = 'Colorado'
GROUP BY
    brewer.name
ORDER BY
    num_beers DESC;
```