GROUP 2 - ASSIGNMENT 1

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

```
SELECT
B.BEER_NAME,
S.STYLE_NAME
FROM
BEERDB.BEERS B
LEFT OUTER JOIN BEERDB.STYLES S
        ON B.STYLE_ID = S.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
   B.BEER_NAME,
   CA.CATEGORY_NAME,
   CO.EXAMPLES,
   S.STYLE_NAME
FROM
   BEERDB.BEERS B
   INNER JOIN BEERDB.COLORS CO
      ON B.SRM = CO.LOVIBOND_SRM
   INNER JOIN BEERDB.CATEGORIES CA
      ON B.CAT_ID = CA.CATEGORY_ID
   INNER JOIN BEERDB.STYLES S
           ON B.STYLE_ID = S.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
   BR.NAME,
```

```
   ROUND(AVG(B.ABV), 1) AS AVERAGE_ABV,
   MIN(B.ABV) AS MIN_ABV,
   MAX(B.ABV) AS MAX_ABV
FROM
   BEERDB.BEERS B
   INNER JOIN BEERDB.BREWERIES BR
      ON B.BREWERY_ID = BR.BREWERY_ID
WHERE
ABV > 0
GROUP BY
BR.NAME
ORDER BY
 AVERAGE_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(BREWERY_ID) AS TOTAL
FROM
   BEERDB.BREWERIES
WHERE
         CITY IS NOT NULL
GROUP BY
         CITY
HAVING
   COUNT(BREWERY_ID) >= 10
ORDER BY
   TOTAL 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
   B.BEER_NAME
FROM
   BEERDB.BEERS B
   INNER JOIN BEERDB.CATEGORIES C
      ON B.CAT_ID = C.CATEGORY_ID
```

```
WHERE
   C.CATEGORY_NAME LIKE '%Lager%'
         AND B.ABV >= 8;
```

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 RELEASE_DATE between '01-JAN-07' and '01-JAN-14'
ORDER BY
   IMDB_RATING DESC;
```

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,
   (USA_GROSS + WORLDWIDE_GROSS) AS TOTAL_GROSS
FROM RELMDB.MOVIES
WHERE
   WORLDWIDE_GROSS IS NOT NULL
         ANd USA_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
         DISTINCT (M.FILM_TITLE)
FROM RELMDB.MOVIES M
         INNER JOIN RELMDB.CASTS C ON M.FILM_ID = C.FILM_ID
```

```
WHERE
        CAST_MEMBER IN ('Tom Hanks', 'Tim Allen');
```

9. Display the number of movies with an MPAA rating of G, PG, PG-13, and R.
Show the results in alphabetical order by MPAA rating.

```
SELECT
        MPAA_RATING,
        COUNT(1) AS MOVIE_COUNT
FROM
        RELMDB.MOVIES
WHERE
        MPAA_RATING IN ('G', 'PG', 'PG-13', 'R')
GROUP BY
        MPAA_RATING
ORDER BY
        MPAA_RATING;
```

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 >= 6
        THEN 'HIGH'
        WHEN ABV >= 3
        THEN 'AVERAGE'
        WHEN ABV >= 0
        THEN 'LOW'
        END AS STRENGTH
FROM BEERDB.BEERS
ORDER BY BEER_NAME ASC;
```

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
        S.STYLE_NAME,
        BR.NAME,
        COUNT(2) AS TOTAL_BEERS
FROM
        BEERDB.BEERS B
        INNER JOIN BEERDB.BREWERIES BR
        ON BR.BREWERY_ID = B.BREWERY_ID
        INNER JOIN BEERDB.STYLES S
        ON B.STYLE_ID = S.STYLE_ID
WHERE
        B.BREWERY_ID IS NOT NULL
        AND S.STYLE_ID IS NOT NULL
        AND S.STYLE_NAME IS NOT NULL
GROUP BY
```

```
        S.STYLE_NAME,
        BR.NAME
HAVING
        COUNT(2) >= 10
ORDER BY
        S.STYLE_NAME ASC,
        COUNT(2) DESC
```

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
        brew.BREWERY_ID,
        brew.NAME,
        COUNT(BEER_ID) AS NUM
FROM
        BEERDB.BEERS beer
        INNER JOIN BEERDB.BREWERIES brew ON beer.BREWERY_ID =
brew.BREWERY_ID
WHERE
        COUNTRY NOT IN ('United States')
 GROUP BY
        brew.NAME,
        brew.BREWERY_ID
HAVING
        COUNT(*) > (
        SELECT
                AVG(COUNT(*))
        FROM
                BEERDB.BEERS be1          WHERE          be1.BREWERY_ID <>
brew.BREWERY_ID   GROUP BY              be1.BREWERY_ID,              brew.BREWERY_ID
        )
ORDER BY
        COUNT(*) DESC,
        brew.NAME;
```

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
        movie.FILM_TITLE AS MOVIE_TITLE,
        movie.FILM_YEAR AS YEAR,
        COUNT(cast.FILM_ID) AS Number_of_Cast_Members
FROM
        RELMDB.MOVIES movie
        INNER JOIN RELMDB.CASTS cast ON (movie.FILM_ID = cast.FILM_ID)
        GROUP BY
        movie.FILM_TITLE, movie.FILM_YEAR
HAVING
        COUNT(cast.FILM_ID) > 5
ORDER BY
        COUNT(cast.FILM_ID) DESC,
        movie.FILM_YEAR DESC,
        movie.FILM_TITLE DESC;
```

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.

(Hint: use the TRIM function to only show a single record from the same genre.)

```
SELECT
        TRIM(genre.GENRE) AS GENRE,
        COUNT(genre.FILM_ID) Total_number_of_films,
        ROUND(AVG(movie.IMDB_RATING), 1) AS Average_Rating,
        ROUND(AVG(movie.USA_GROSS), 1) AS Average_Gross_Rev_USA
FROM
        RELMDB.MOVIES movie
        INNER JOIN RELMDB.GENRES genre ON movie.FILM_ID = genre.FILM_ID
WHERE
        movie.USA_GROSS IS NOT NULL
GROUP BY
        TRIM(genre.GENRE)
HAVING
```

```
        COUNT(genre.FILM_ID) >= 5
ORDER BY
        TRIM(genre.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
        ROUND(AVG(movie.BUDGET)) AS
BUDGET, dr.DIRECTOR
FROM
        RELMDB.MOVIES movie
    INNER JOIN RELMDB.DIRECTORS dr ON movie.FILM_ID =
dr.FILM_ID WHERE
        movie.FILM_TITLE IN (
        SELECT
        FILM_TITLE
        FROM
        RELMDB.MOVIES
        WHERE
        IMDB_RANK <= 25
        )
GROUP BY
        dr.DIRECTOR
ORDER BY
        AVG(movie.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 AS First_Name,
        LNAME AS Last_Name,
        CITY,
        STATE,
        ZIP AS Zip_Code,
        BIRTH_DAY AS DOB_DAY,
        BIRTH_MONTH AS DOB_Month,
        BIRTH_YEAR AS DOB_Year
FROM
        RELMDB.FANS
GROUP BY
        FNAME,
        LNAME,
        CITY,
        STATE,
        ZIP,
        BIRTH_DAY,
        BIRTH_MONTH,
        BIRTH_YEAR
HAVING (COUNT(3) > 1) ;
```

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 than the new table. Find all fans that exist in the old fans table but not the new table.

```
SELECT
 old.FNAME AS First_Name,
 old.LNAME AS Last_Name
FROM
        RELMDB.FANS_OLD old
WHERE
        NOT EXISTS (SELECT fan.FNAME,fan.LNAME FROM
        RELMDB.FANS fan WHERE
        old.FNAME = fan.FNAME
        AND
old.LNAME = fan.LNAME);
```

Use only the first and last name when comparing fans between the two tables.

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
        BREWERY_ID,
        NAME AS Brewery_Name,
        NUM_BEERS AS "Number of Beers",
        CASE
        WHEN PERCENT_RANKING <= 0.25 THEN 1
        WHEN PERCENT_RANKING > 0.25 AND PERCENT_RANKING <= 0.5 THEN 2
        WHEN PERCENT_RANKING > 0.5 AND PERCENT_RANKING <= 0.75 THEN 3
```

```
        WHEN PERCENT_RANKING > 0.75 AND PERCENT_RANKING <= 1 THEN 4 END AS
"Percentile Group"
FROM (
        SELECT
        BREWERY_ID,
        NAME,
        NUM_BEERS,
        PERCENT_RANK() OVER (ORDER BY NUM_BEERS DESC, BREWERY_ID DESC)
AS PERCENT_RANKING
        FROM (
        SELECT
        BREW.BREWERY_ID,
        NAME,
        COUNT(*) AS NUM_BEERS
        FROM BEERDB.BREWERIES BREW
        INNER JOIN BEERDB.BEERS BEER
        ON BREW.BREWERY_ID = BEER.BREWERY_ID
        GROUP BY BREW.BREWERY_ID,NAME
        ORDER BY NUM_BEERS DESC, BREW.BREWERY_ID DESC
        )
        );
```

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
```

```
    RANKS.BEER_NAME,
    RANKS.ABV,
    RANKS.RANK
FROM (
    SELECT
        BEER_NAME,
        ABV,
        DENSE_RANK() OVER (ORDER BY ABV DESC) AS RANK
    FROM
        BEERDB.BEERS
        WHERE ABV > 0
        ORDER BY ABV DESC
    ) RANKS
    WHERE RANKS.RANK <= 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
        FILM_TITLE,
        FILM_YEAR,
        WORLDWIDE_GROSS as WW_Gross_Revenue,
        SUM(WORLDWIDE_GROSS) OVER (ORDER BY FILM_YEAR) AS
        Cumulative_WW_Gross_Revenue
FROM
        RELMDB.MOVIES movie
INNER JOIN
```

```
        RELMDB.DIRECTORS direct
ON
        movie.FILM_ID = direct.FILM_ID
WHERE
        DIRECTOR='Christopher Nolan'
AND
        WORLDWIDE_GROSS > 0
ORDER BY
        FILM_YEAR;
```

22. Display the following information using a single SQL statement: (a) total budget and USA gross for

each combination of genre and MPAA rating;

--(b) total budget and USA gross for each genre from (a); and (c) Total budget and USA gross

for all genres and MPAA ratings shown in (a).

--Only movies with non null values of budget and USA gross should be included. Sort the

records by genre and then MPAA rating.

--(Hint: use the TRIM function to only show a single record from the same genre.)

CUSTOM QUERIES:

1. Created a virtual table to test it out using the view command. The sql query is for checking the metascore between 75 and 100. It also includes the movie title, release year, country, IMDB rating as well as the metascore.

   I used the right join to combine the data from the critic reviews and the movies table. The data was arranged by highest metascore value

```
CREATE VIEW 75_100_metascore AS
select MOVIE_TITLE,release_year,COUNTRY,IMDB_RATING,metascore FROM
RMDB.MOVIES
RIGHT JOIN RMDB.CRITIC_REVIEWS on CRITIC_REVIEWS.movie_guid =
movies.movie_guid
WHERE metascore BETWEEN 75 AND 100
```

```
ORDER BY metascore DESC
```

2. Created a sql query to show beer names starting with the letter "H" and that has an Alcohol By Volume (ABV) above 5% that is located in the state of Florida.

```
SELECT *

FROM BEERS

RIGHT JOIN BREWERIES on BREWERIES.BREWERY_ID = BEERS.BREWERY_ID

WHERE BEER_NAME LIKE 'H%' AND ABV > 5 AND COUNTRY = 'United States'

AND breweries.state = 'Florida'
```

3. Created a query to list all the Yuengling beers, where they are brewed, the type (style) of beer and the associated ABV

```
SELECT BEER_NAME,styles.style_id,styles.style_name, brewery_id,ABV

FROM beers

RIGHT JOIN styles on styles.style_id = beers.style_id

WHERE BEER_NAME LIKE 'Yuengling%'
```