Analyzing American Baby Name Trends

This project uses techniques learned in **Intermediate SQL**, including `CASE WHEN` statements, pattern matching using the `LIKE` operator, subqueries, common table expressions, and window functions. You'll also be expected to know concepts from **Introduction to SQL** and **Joining Data with SQL**, such as how to select columns from a table, filter rows where they meet a criterion, use aggregation functions, perform calculations on groups of rows, filter grouped data, and join data.

The project is solved in mySQL. You may look at the tasks and try to solve using any other SQL language. The last 2 tasks were a bit advance topics so it is advised to look for solution in notebook if not able to solve as even I had to look at hint for the same.

Below are the tasks to be performed:

Task 1: Instructions

Find names that have been given to over 5,000 babies of either sex every year for the 101 years from 1920 through 2020; recall that names only show up in our dataset when at least 5,000 babies have been given that name in a year.

- Select `first_name` and the total number of babies that have ever been given that name.
- Group by `first_name` and filter for those names that appear in all 101 years.
- Order the results by the total number of babies that have ever been given that name, descending.
- *The line `postgresql:///names` is used to connect to the database; don't remove it.*

Task 2: Instructions

Classify each name's popularity according to the number of years that the name appears in the dataset.

- Select `first_name`, the sum of babies who've ever been given that name, and `popularity_type`.
- Classify all names in the dataset as 'Classic,' 'Semi-classic,' 'Semi-trendy,' or 'Trendy' based on whether the name appears in the dataset more than 80, 50, 20, or 0 times, respectively.
- Alias the new classification column as `popularity_type`.

- Order the results alphabetically by `first_name`.

Task 3: Instructions

Let's take a look at the ten highest-ranked American female names in our dataset.

- Select `name_rank`, `first_name`, and the sum of babies who've ever had that name.
- `RANK` the `first_name` by the sum of babies who've ever had that name, aliasing as `name_rank` and showing the names in descending order by `name_rank`.
- Filter the data to include only results where `sex` equals 'F'.
- Limit to ten results.

Task 4: Instructions

Return a list of first names which meet this friend's baby name criteria.

- Select only the `first_name` column.
- Filter the data for results where `sex` equals 'F', the `year` is greater than 2015, and the `first_name` ends in an 'a.'
- Group the data by `first_name` and order by the total number of babies ever given that `first_name`, descending.

Task 5: Instructions

Find the cumulative number of babies named Olivia over the years since the name first appeared in our dataset.

- Select `year`, `first_name`, `num` of Olivias in that year, and `cumulative_olivias`.
- Using a window function, sum the cumulative number of babies who have ever been named Olivia up to that `year`; alias as `cumulative_olivias`.
- Filter the results so that only data for the name Olivia is returned.
- Order the results by `year` from the earliest year Olivia appeared in the dataset to the most recent.

Task 6: Instructions

Write a query that selects the `year` and the maximum `num` of babies given any male name in that year.

- Select the `year` and the maximum `num` of babies given any one male name in that year; alias the maximum as `max_num`.
- Filter the data to include only results where `sex` equals 'M'.

Task 7: Instructions

Using the previous task's code as a subquery, look up the `first_name` that corresponds to the maximum number of babies given a specific male name in a year.

- Select `year`, the `first_name` given to the largest number of male babies, and `num` of babies given the `first_name` that year.
- Join `baby_names` to the code in the last task as a subquery, using whatever alias you like and joining on *both* columns in the subquery.
- Order the results by year, starting with the most recent year.

Task 8: Instructions

Return a list of first names that have been the top male first name in any year along with a count of the number of years that name has been the top name.

- Select `first_name` and a count of the number of years that the `first_name` appeared as a year's top name in the last task; alias this count as `count_top_name`.
- To do this, use the code from the previous task as a common table expression.
- Group by `first_name` and order the results from the name with the most years at the top to the name with the fewest.