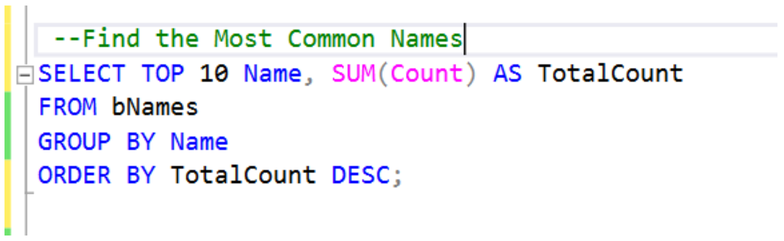
SQL : Analyzing American Baby Names for Trends and Timelessness

Introduction

Embarking on an exciting journey into the world of SQL, I recently completed a dynamic project focused on American baby names, which allowed me to put my SQL skills to the test. By working with the extensive dataset provided by the United States Social Security Administration, I explored trends in baby name popularity, identified timeless classics, and examined the rise and fall of names over the years. This project was not only a fun challenge but also a valuable learning experience, as it deepened my understanding of SQL and its ability to extract meaningful insights from large datasets. The process of analyzing name trends over the past century truly showcased SQL's power in revealing patterns and offering a data-driven perspective on cultural shifts.

1. Find the Most Common Names

The goal was to identify the most common baby names across the dataset. To achieve this, a query was written to retrieve the top 10 names with the highest total occurrences, providing insights into the most frequently chosen names over time.

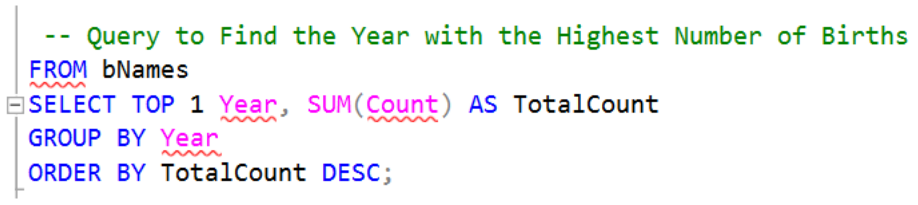


```sql
--Find the Most Common Names
SELECT TOP 10 Name, SUM(Count) AS TotalCount
FROM bNames
GROUP BY Name
ORDER BY TotalCount DESC;
```

The query works by grouping the data based on the Name column and summing the Count for each name. This allows us to capture the cumulative number of times a name has appeared across all years, rather than just its frequency in a single year. Once the names are grouped and their total occurrences are calculated, the results are ordered in descending order of the total count. The top 10 names with the highest total counts are then selected for display.

2. Query to Find the Year with the Highest Number of Births

The objective was to determine the year with the highest number of total births, based on the dataset provided. This query calculates the cumulative number of births for each year by summing the Count for all names within a given year, and then it identifies the year with the highest total.

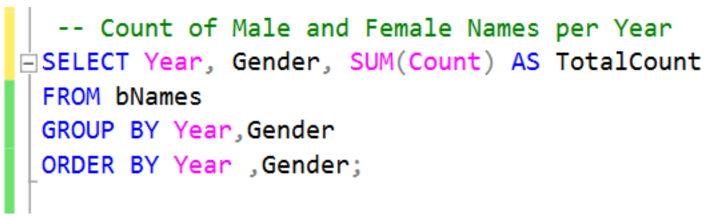


```sql
-- Query to Find the Year with the Highest Number of Births
FROM bNames
SELECT TOP 1 Year, SUM(Count) AS TotalCount
GROUP BY Year
ORDER BY TotalCount DESC;
```

The query groups the data by Year and sums the Count values, which represent the number of occurrences for each name in a particular year. By doing this, the total number of births for each year is calculated. The results are then ordered in descending order by the summed TotalCount, ensuring that the year with the highest number of births appears at the top.

3. Count of Male and Female Names per Year

The aim was to examine the distribution of male and female names across different years by calculating the total number of occurrences for each gender per year. This query aggregates the data based on Year and Gender, providing a comprehensive view of the total count of male and female names for each year.

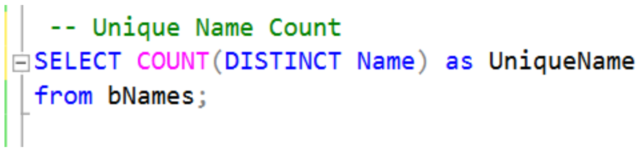


```sql
 -- Count of Male and Female Names per Year
SELECT Year, Gender, SUM(Count) AS TotalCount
FROM bNames
GROUP BY Year,Gender
ORDER BY Year ,Gender;
```

The query works by grouping the data by both Year and Gender, and then it sums the Count values, which represent the number of occurrences of each name. By summing these counts for each gender within each year, the query calculates the total number of male and female names recorded for each year in the dataset.

4. **Unique Name Count**

The goal was to determine the total number of unique baby names in the dataset. The query uses the COUNT(DISTINCT Name) function to count the number of distinct names, ensuring that each name is only counted once, regardless of how many times it appears in the dataset across different years or with different gender entries.

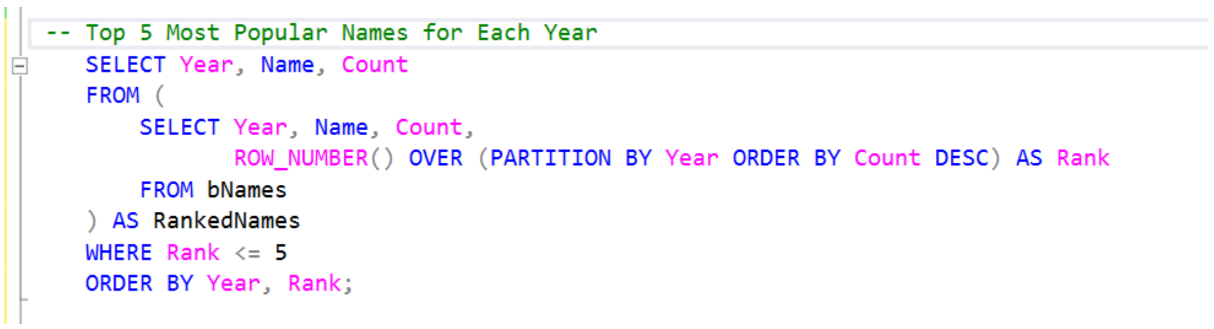


```sql
 -- Unique Name Count
SELECT COUNT(DISTINCT Name) as UniqueName
from bNames;
```

The query retrieves the count of all distinct names from the bNames table by applying the DISTINCT keyword to the Name column. This ensures that only unique names are considered, effectively eliminating duplicates and providing an accurate count of all the different names in the dataset.

5 .**Top 5 Most Popular Names for Each Year**

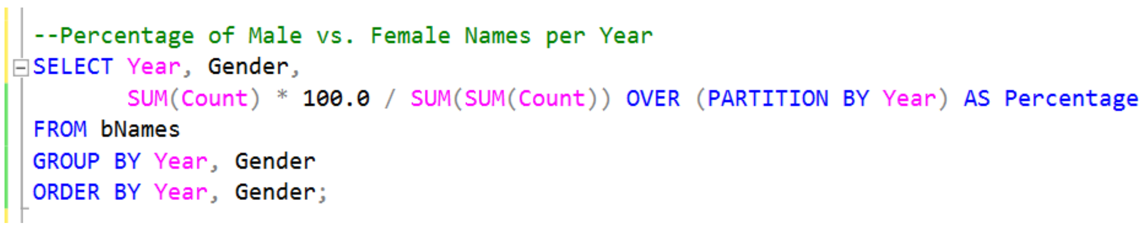
In this section of the project, the goal was to identify the top 5 most popular baby names for each year, based on the total number of occurrences (Count) in the dataset. This query uses the ROW_NUMBER() window function to rank the names by their popularity within each year, and then filters the results to display only the top 5 names for each year.



```sql
-- Top 5 Most Popular Names for Each Year
   SELECT Year, Name, Count
   FROM (
       SELECT Year, Name, Count,
              ROW_NUMBER() OVER (PARTITION BY Year ORDER BY Count DESC) AS Rank
       FROM bNames
   ) AS RankedNames
   WHERE Rank <= 5
   ORDER BY Year, Rank;
```

6. **Percentage of Male vs. Female Names per Year**

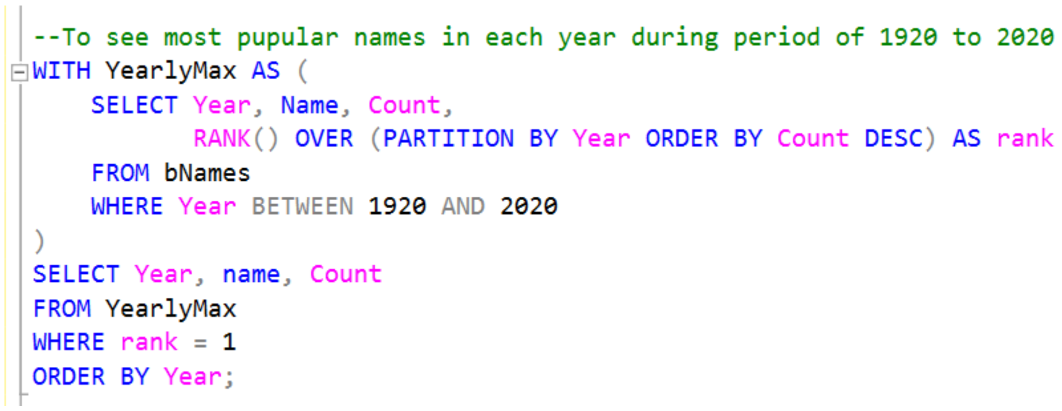
The goal was to analyze the distribution of male and female names over the years and calculate the percentage of total names that were male versus female in each year. This query helps to understand the gender breakdown of baby names on a yearly basis, providing insights into trends and shifts in gender preferences for names.



```sql
--Percentage of Male vs. Female Names per Year
SELECT Year, Gender,
       SUM(Count) * 100.0 / SUM(SUM(Count)) OVER (PARTITION BY Year) AS Percentage
FROM bNames
GROUP BY Year, Gender
ORDER BY Year, Gender;
```

7. To See the Most Popular Names in Each Year from 1920 to 2020

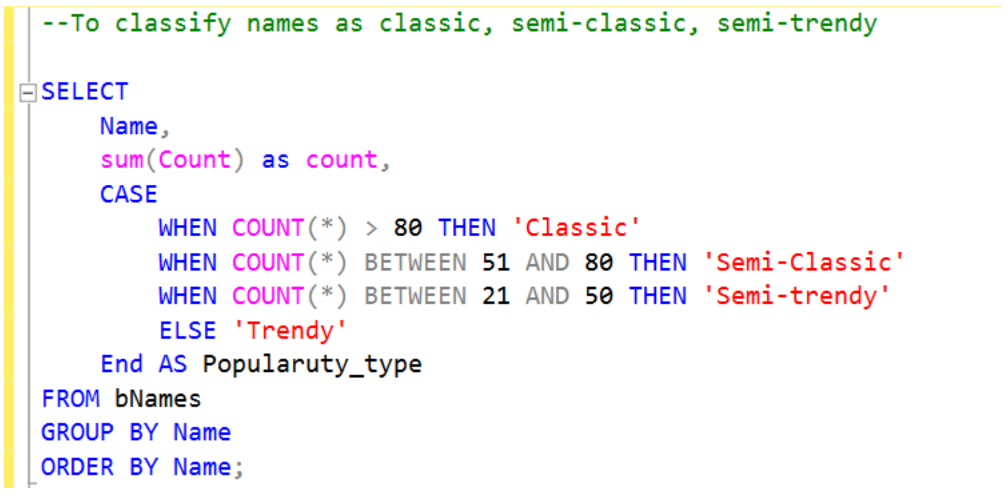
In this section of the project, the objective was to identify the most popular baby name for each year between 1920 and 2020. This query leverages a Common Table Expression (CTE) and the RANK() window function to rank the names within each year based on their total occurrences and then filters to retrieve only the most popular name for each year.



```sql
--To see most pupular names in each year during period of 1920 to 2020
WITH YearlyMax AS (
    SELECT Year, Name, Count,
           RANK() OVER (PARTITION BY Year ORDER BY Count DESC) AS rank
    FROM bNames
    WHERE Year BETWEEN 1920 AND 2020
)
SELECT Year, name, Count
FROM YearlyMax
WHERE rank = 1
ORDER BY Year;
```

8. To Classify Names as Classic, Semi-Classic, Semi-Trendy, or Trendy

In this part of the project, the goal was to classify names based on their popularity patterns into four categories: **Classic**, **Semi-Classic**, **Semi-Trendy**, and **Trendy**. This classification helps to identify how certain names have sustained their popularity over time or how they have emerged and faded. The query categorizes each name by calculating the total number of occurrences across all years and assigning a popularity type based on its frequency.

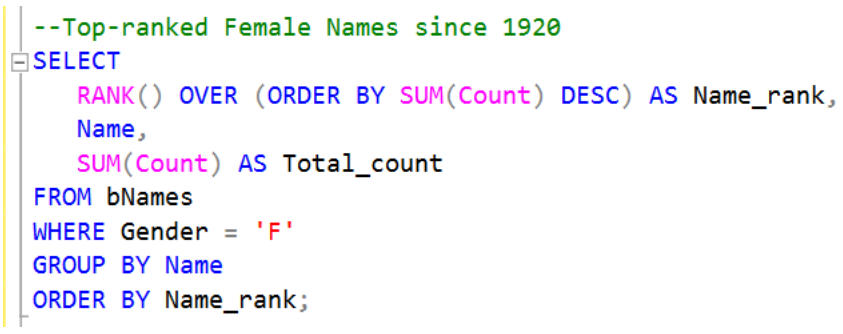


```
--To classify names as classic, semi-classic, semi-trendy

SELECT
    Name,
    sum(Count) as count,
    CASE
        WHEN COUNT(*) > 80 THEN 'Classic'
        WHEN COUNT(*) BETWEEN 51 AND 80 THEN 'Semi-Classic'
        WHEN COUNT(*) BETWEEN 21 AND 50 THEN 'Semi-trendy'
        ELSE 'Trendy'
    End AS Popularuty_type
FROM bNames
GROUP BY Name
ORDER BY Name;
```

9. Top-ranked Female Names Since 1920

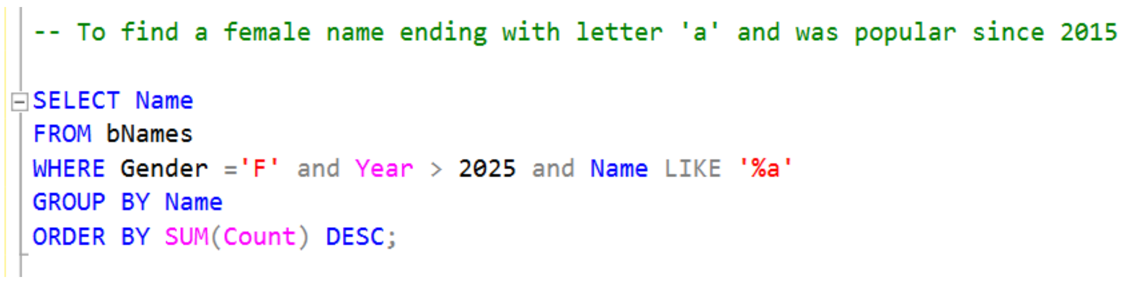
In this part of the project, the goal was to identify the top-ranked female names based on their total occurrences across all years since 1920. The query uses the RANK() window function to rank the female names by their total count, allowing us to understand which names have been the most popular for girls over the years.



```
--Top-ranked Female Names since 1920
SELECT
    RANK() OVER (ORDER BY SUM(Count) DESC) AS Name_rank,
    Name,
    SUM(Count) AS Total_count
FROM bNames
WHERE Gender = 'F'
GROUP BY Name
ORDER BY Name_rank;
```

10. To Find a Female Name Ending with the Letter 'A' and Was Popular Since 2015

The aim of this query is to find female names that end with the letter "a" and have been popular in recent years, specifically since 2015. The query uses the LIKE operator to filter names that end with "a" and includes a condition to only consider names from 2015 onward. The results are grouped by name, and the total count of occurrences for each name is summed, with the most popular names appearing first.

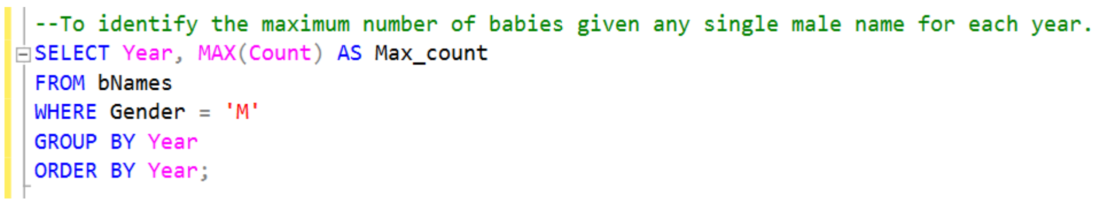


```sql
-- To find a female name ending with letter 'a' and was popular since 2015

SELECT Name
FROM bNames
WHERE Gender ='F' and Year > 2025 and Name LIKE '%a'
GROUP BY Name
ORDER BY SUM(Count) DESC;
```

11. To Identify the Maximum Number of Babies Given Any Single Male Name for Each Year

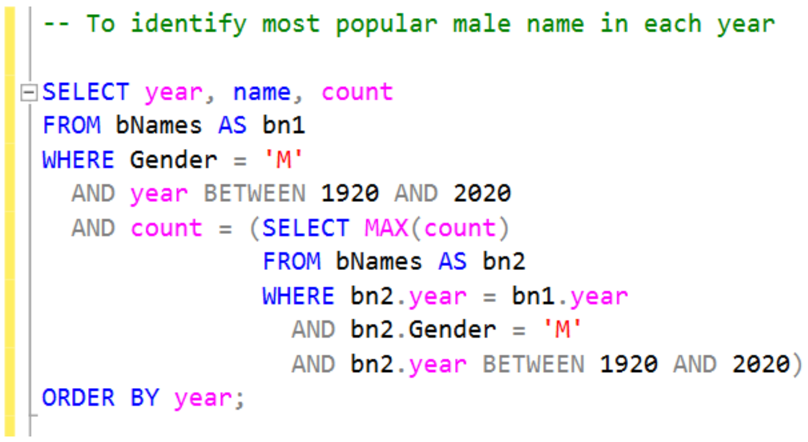
This query is designed to identify the highest number of babies given a single male name in each year. By using the MAX() function, the query determines the most popular male name (in terms of the number of occurrences) for every year within the dataset.



```sql
--To identify the maximum number of babies given any single male name for each year.
SELECT Year, MAX(Count) AS Max_count
FROM bNames
WHERE Gender = 'M'
GROUP BY Year
ORDER BY Year;
```

12. To Identify the Most Popular Male Name in Each Year (1920–2020)

The goal of this query is to identify the most popular male name for each year between 1920 and 2020. It does this by comparing the count of occurrences for each male name in a given year and selecting the name with the highest count.

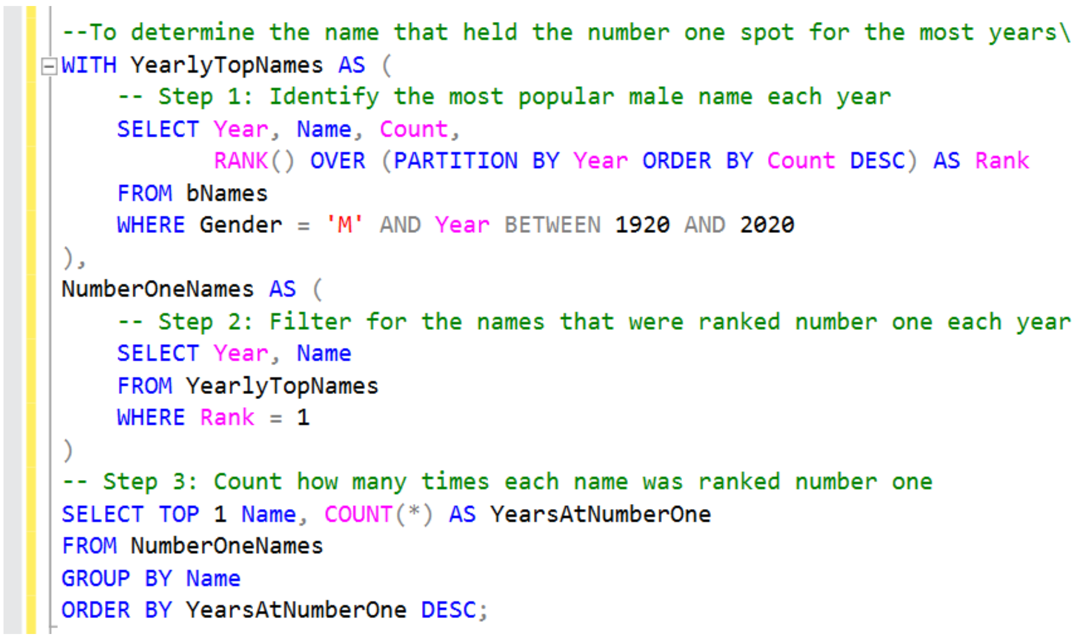


```sql
-- To identify most popular male name in each year

SELECT year, name, count
FROM bNames AS bn1
WHERE Gender = 'M'
  AND year BETWEEN 1920 AND 2020
  AND count = (SELECT MAX(count)
               FROM bNames AS bn2
               WHERE bn2.year = bn1.year
                 AND bn2.Gender = 'M'
                 AND bn2.year BETWEEN 1920 AND 2020)
ORDER BY year;
```

13 .To Determine the Name That Held the Number One Spot for the Most Years

This query aims to identify the name that held the number one spot for the most years from 1920 to 2020. It uses a combination of **Common Table Expressions (CTEs)** and the RANK() window function to achieve this.



```sql
--To determine the name that held the number one spot for the most years\
WITH YearlyTopNames AS (
    -- Step 1: Identify the most popular male name each year
    SELECT Year, Name, Count,
           RANK() OVER (PARTITION BY Year ORDER BY Count DESC) AS Rank
    FROM bNames
    WHERE Gender = 'M' AND Year BETWEEN 1920 AND 2020
),
NumberOneNames AS (
    -- Step 2: Filter for the names that were ranked number one each year
    SELECT Year, Name
    FROM YearlyTopNames
    WHERE Rank = 1
)
-- Step 3: Count how many times each name was ranked number one
SELECT TOP 1 Name, COUNT(*) AS YearsAtNumberOne
FROM NumberOneNames
GROUP BY Name
ORDER BY YearsAtNumberOne DESC;
```

Conclusion :

Working on this baby names project provided an incredibly insightful experience with SQL. Analyzing a comprehensive dataset covering a century of American naming trends, I applied a range of SQL functions and techniques to extract meaningful insights. This project highlighted SQL's versatility as a data analysis tool, enabling me to identify patterns, track name popularity, and observe the emergence of specific names over time. The experience has not only strengthened my SQL skills but also broadened my understanding of its powerful capacity to reveal fascinating narratives within complex datasets.