The data source and questions for this project was provided by Datacamp.com, this is part of their paid subscription service located in the Project portion of the website. PostgreSQL was written by Robert Marston.

The purpose of this project is to demonstrate critical thinking, problem solving and SQL knowledge. Follow along in Tableau on the right!

1. Classic American names

How have American baby name tastes changed since 1920? Which names have remained popular for over 100 years, and how do those names compare to more recent top baby names?

We'll be working with data provided by the United States Social Security Administration, which lists first names along with the number and sex of babies they were given to in each year. For processing speed purposes, we've limited the dataset to first names which were given to over 5,000 American babies in a given year. Our data spans 101 years, from 1920 through 2020.

baby_names database layout

column	type	meaning
year	int	year
first_name	varchar	first name
sex	varchar	sex of babies given first_name
num	int	number of babies of sex given first_name in that year

Let's get oriented to American baby name tastes by looking at the names that have stood the test of time! **Find names that have been given to babies of either sex every year from 1920-2020**

In:

```
%%sql
postgresql:///names

SELECT first_name, SUM(num)
FROM baby_names
   GROUP BY first_name
      HAVING COUNT(year) = 101
ORDER BY SUM(num) DESC;
```

Out:

8 rows affected. View these results with Slide #1 “Baby names that have withstood the test of time”

first_name	sum
James	4748138
John	4510721
William	3614424
David	3571498
Joseph	2361382
Thomas	2166802
Charles	2112352
Elizabeth	1436286

2. Timeless or trendy?

Now, let's broaden our understanding of the dataset by looking at all names. We'll attempt to capture the type of popularity that each name in the dataset enjoyed. Was the name classic and popular across many years or trendy, only popular for a few years? Let's find out. **Classify each name’s popularity according to the number of years the name appears in the dataset.**

In:

```
%%sql

SELECT first_name, SUM(num),
   CASE WHEN COUNT(year) > 80 THEN 'Classic'
      WHEN COUNT(year) > 50 THEN 'Semi-classic'
      WHEN COUNT(year) > 20 THEN 'Semi-trendy'
      ELSE 'Trendy' END AS popularity_type
FROM baby_names
GROUP BY first_name
ORDER BY first_name;
```

```
 * postgresql:///names
```

547 rows affected. ****A SAMPLE OF THE OUTPUT IS BELOW; THE FULL OUTPUT CAN BE VIEWED IN THE TABLEAU SCREEN. Slide #2 Timeless or Trendy Make sure you scroll to the right!**

Out:

first_name	sum	popularity_type
Aaliyah	15870	Trendy
Aaron	530592	Semi-classic
Abigail	338485	Semi-trendy
Adam	497293	Semi-trendy

Addison	107433	Trendy
Bailey	10219	Trendy
Barbara	1343901	Semi-classic
Beatrice	27983	Trendy
Bella	5127	Trendy
Caitlin	38501	Trendy
Caleb	259439	Semi-trendy
Cameron	253711	Semi-trendy
Charles	2112352	Classic
Dale	130919	Trendy
Dana	51558	Trendy
Daniel	1824274	Classic
Danielle	299683	Semi-trendy
Easton	21820	Trendy
Edna	63698	Trendy
Faith	27204	Trendy
Florence	99171	Trendy
Frances	348520	Semi-trendy
Greg	15849	Trendy
Gregory	644286	Semi-trendy
Hailey	111571	Trendy
Haley	106119	Trendy
Jocelyn	5292	Trendy
Joe	302127	Semi-trendy
Kyle	394877	Semi-trendy
Kylie	16309	Trendy
Landon	129558	Trendy
Mackenzie	46972	Trendy
Madeline	26888	Trendy
Olivia	429118	Semi-trendy
Nicole	533803	Semi-trendy
Peter	388795	Semi-classic
Peyton	5315	Trendy
Rebecca	638458	Semi-classic
Regina	10003	Trendy
Samantha	514826	Semi-trendy
Samuel	539556	Semi-classic
Travis	218731	Semi-trendy
Virginia	441418	Semi-trendy
Walter	378194	Semi-trendy
Wanda	125458	Trendy
Xavier	51892	Trendy
Zachary	483955	Semi-trendy

3. Top-ranked female names since 1920

Since we didn't get many traditionally female names in our classic American names search in the first task, let's limit our search to names which were given to female babies. **What are the top 10 ranked female names since 1920?**

In:

```
%%sql

SELECT
   RANK() OVER(ORDER BY SUM(num) DESC) AS name_rank,
   first_name,
   SUM(num)
FROM baby_names
   WHERE sex = 'F'
   GROUP BY first_name
LIMIT 10;
```

```
 * postgresql:///names
10 rows affected. View results with slide 3 “Top Ten Ranked Female Names Since 1920”
```

Out:

name_rank	first_name	sum
1	Mary	3215850
2	Patricia	1479802
3	Elizabeth	1436286
4	Jennifer	1404743
5	Linda	1361021
6	Barbara	1343901
7	Susan	1025728
8	Jessica	994210
9	Lisa	920119
10	Betty	893396

4. Picking a baby name

A friend has heard of our work analyzing baby names and would like help choosing a name for her baby, a girl! She doesn't like any of the top-ranked names we found in the previous task. She's set on a traditionally female name ending in the letter 'a' since she's heard that vowels in baby names

are trendy. She's also looking for a name that has been popular in the years since 2015. **What are the top Female names popular since the year 2015 that end in “A”?**

In [73]:

```
%%sql

SELECT first_name
FROM baby_names
   WHERE sex = 'F' AND year > 2015 AND first_name LIKE '%a'
   GROUP BY first_name
   ORDER BY SUM(num) DESC;
```

```
 * postgresql:///names
19 rows affected. View results with slide #4 “Picking a Baby Name”
```

Out:

first_name
Olivia
Emma
Ava
Sophia
Isabella
Mia
Amelia
Ella
Sofia
Camila
Aria
Victoria
Layla
Nora
Mila
Luna
Stella
Gianna
Aurora

5. The Olivia expansion

Based on the results in the previous task, we can see that Olivia is the most popular female name ending in 'A' since 2015. When did the name Olivia become so popular? **Find the cumulative number of babies named Olivia over the years since the name first appeared in our Dataset.**

In:

```
%%sql

SELECT year, first_name, num,
   SUM(num) OVER (ORDER BY year) AS cumulative_olivias
FROM baby_names
   WHERE first_name = 'Olivia'
ORDER BY year;
```

```
 * postgresql:///names
30 rows affected. View results with slide #5 “The Olivia Expansion”
```

Out:

year	first_name	num	cumulative_olivias
1991	Olivia	5601	5601
1992	Olivia	5809	11410
1993	Olivia	6340	17750
1994	Olivia	6434	24184
1995	Olivia	7624	31808
1996	Olivia	8124	39932
1997	Olivia	9477	49409
1998	Olivia	10610	60019
1999	Olivia	11255	71274
2000	Olivia	12852	84126
2001	Olivia	13977	98103
2002	Olivia	14630	112733
2003	Olivia	16152	128885
2004	Olivia	16106	144991
2005	Olivia	15694	160685
2006	Olivia	15501	176186

year	first_name	num	cumulative_olivias
2007	Olivia	16584	192770
2008	Olivia	17084	209854
2009	Olivia	17438	227292
2010	Olivia	17029	244321
2011	Olivia	17327	261648
2012	Olivia	17320	278968
2013	Olivia	18439	297407
2014	Olivia	19823	317230
2015	Olivia	19710	336940
2016	Olivia	19380	356320
2017	Olivia	18744	375064
2018	Olivia	18011	393075
2019	Olivia	18508	411583
2020	Olivia	17535	429118

6. Top male names over the years

In the previous task, we found the maximum number of babies given any one male name in each year. Incredibly, the most popular name each year varied from being given to less than 20,000 babies to being given to more than 90,000! **Find out what that top male name is for each year in our dataset.**

In:

```
SELECT baby_names.year, first_name, num
FROM baby_names
   INNER JOIN (SELECT year, MAX(num) AS max_num
               FROM baby_names
                  WHERE sex = 'M'
               GROUP BY year
               ORDER BY year) AS subquery
      ON baby_names.year = subquery.year
         AND num = max_num
   WHERE sex = 'M'
   ORDER BY year DESC;
 * postgresql:///names
```

101 rows affected.

Out:

year	first_name	num	year	first_name	num	year	first_name	num	year	first_name	num
2020	Liam	19659	1995	Michael	41399	1970	Michael	85291	1945	James	74460
2019	Liam	20555	1994	Michael	44472	1969	Michael	85201	1944	James	76954
2018	Liam	19924	1993	Michael	49554	1968	Michael	81995	1943	James	80274
2017	Liam	18824	1992	Michael	54397	1967	Michael	82440	1942	James	77174
2016	Noah	19154	1991	Michael	60793	1966	Michael	79990	1941	James	66743
2015	Noah	19650	1990	Michael	65302	1965	Michael	81021	1940	James	62476
2014	Noah	19319	1989	Michael	65399	1964	Michael	82642	1939	Robert	59653
2013	Noah	18266	1988	Michael	64150	1963	Michael	83778	1938	Robert	62269
2012	Jacob	19088	1987	Michael	63654	1962	Michael	85041	1937	Robert	61842
2011	Jacob	20378	1986	Michael	64224	1961	Michael	86917	1936	Robert	58499
2010	Jacob	22139	1985	Michael	64924	1960	David	85933	1935	Robert	56522
2009	Jacob	21184	1984	Michael	67745	1959	Michael	85224	1934	Robert	55834
2008	Jacob	22603	1983	Michael	68010	1958	Michael	90564	1933	Robert	54223
2007	Jacob	24292	1982	Michael	68244	1957	Michael	92718	1932	Robert	59265
2006	Jacob	24850	1981	Michael	68776	1956	Michael	90665	1931	Robert	60518
2005	Jacob	25837	1980	Michael	68704	1955	Michael	88372	1930	Robert	62149
2004	Jacob	27886	1979	Michael	67742	1954	Michael	88576	1929	Robert	59804
2003	Jacob	29643	1978	Michael	67157	1953	Robert	86247	1928	Robert	60703
2002	Jacob	30579	1977	Michael	67609	1952	James	87063	1927	Robert	61671
2001	Jacob	32554	1976	Michael	66947	1951	James	87261	1926	Robert	61130
2000	Jacob	34483	1975	Michael	68451	1950	James	86229	1925	Robert	60897
1999	Jacob	35367	1974	Michael	67580	1949	James	86865	1924	Robert	60801
1998	Michael	36616	1973	Michael	67842	1948	James	88589	1923	John	57469
1997	Michael	37549	1972	Michael	71401	1947	James	94764	1922	John	57280
1996	Michael	38365	1971	Michael	77599	1946	James	87439	1921	John	58215
									1920	John	56914

8. The most years at number one

Noah and Liam have ruled the roost in the last few years, but if we scroll down in the results, it looks like Michael and Jacob have also spent a good number of years as the top name! **Which name has been number one for the largest number of years?**

In:

```
%%sql

WITH subquery AS (
SELECT baby_names.year, first_name, num
```

```
FROM baby_names
   INNER JOIN (SELECT year, MAX(num) AS max_num
               FROM baby_names
                  WHERE sex = 'M'
               GROUP BY year
               ORDER BY year) AS subquery
      ON baby_names.year = subquery.year
         AND num = max_num
   WHERE sex = 'M'
   ORDER BY year DESC
               )
                SELECT first_name, COUNT(*) AS count_top_name
                FROM subquery
                   GROUP BY first_name
                   ORDER BY count_top_name DESC;
```

```
 * postgresql:///names
8 rows affected.
```

Out:

first_name	count_top_name
Michael	44
Robert	17
Jacob	14
James	13
Noah	4
John	4
Liam	4
David	1