Police Death Portfolio Project

Nestor Torrech

2022-06-21

Introduction

Hello, welcome to my exploratory analysis of a dataset compiling registered police deaths in the United States. The main focus of this project shall be to perform some beginner and intermediate queries to answer interesting questions about the data and to demonstrate my proficiency with SQL. Some additional details are the following.

This document was prepared in Rmarkdown, utilzing its particular features which allow me to use SQL syntax within the R framework. This proves immensely useful as Rmarkdown can generate this pdf.

Preparing the Data

As we stated before, the Dataset is sourced from Kaggle, but originally comes from FiveThirtyEight. Links to both sources shall be included in the *.txt* file that is in this project's repository. In this same text file, other relevant information, such as the data dictionary are available for those interested.

Now, on to the subject of interest. First things first, we'll load some libraries that will facilitate us working with SQL in Rmarkdown.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(DBI)
library(odbc)
police_death <- read_csv("C:\\SQL Datasets\\Datasets for Portfolio Project\\police_deaths_in_america.csv
```

```
## Rows: 26269 Columns: 10
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): Rank, Name, Cause_of_Death, Date, Month, Day, Department, State
## dbl (2): Year, K9_Unit
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(police_death)
```

```
## # A tibble: 6 x 10
##   Rank     Name  Cause_of_Death Date   Year Month Day   Department State K9_Unit
##   <chr>    <chr> <chr>          <chr> <dbl> <chr> <chr> <chr>      <chr>   <dbl>
## 1 Constab~ Dari~ Stabbed        Mond~  1791 Janu~ Mond~ Albany Co~ New ~       0
## 2 Sheriff  Corn~ Gunfire        Satu~  1791 Octo~ Satu~ Columbia ~ New ~       0
## 3 Deputy ~ Isaa~ Gunfire        Thur~  1792 May   Thur~ Westchest~ New ~       0
## 4 Marshal  Robe~ Gunfire        Satu~  1794 Janu~ Satu~ United St~ Unit~       0
## 5 Deputy ~ Robe~ Gunfire        Thur~  1797 June  Thur~ New York ~ New ~       0
## 6 Sheriff  Robe~ Gunfire        Sund~  1797 Nove~ Sund~ Washingto~ Sout~       0
```

The data stretches back all the way to the late 18th century! These means we've got plenty to parse through ahead of us. It's important to note that at this point in the process, the table is being handled purely through R code. So, if we want to manipulate it using SQL, we'll need to set up a Database in R.

So let's get to it.

NOTE: Bear in mind that in the pdf version of this document, due to the formating in Rmarkdown, the text will look somewhat squished once we view all the columns in the dataset. This only happens on this occasion. If you wish to have a clearer view of the Data, please refer to the html version of this report or perhaps consider using the rmd file in which this document was drafted. Please forgive any inconvenience

Readying the Database

```
#Readying a Database

pol_DB <- dbConnect(drv = RSQLite::SQLite(),
                    dbname = ":memory:")

#Storing Sample Data

dbWriteTable(conn = pol_DB,
             name = "police_death",
             value = police_death)

# Remove table from environment

rm(police_death)
```

```
tbl(src = pol_DB,
    "police_death")
```

```
## # Source:   table<police_death> [?? x 10]
## # Database: sqlite 3.38.5 [:memory:]
##    Rank    Name  Cause_of_Death Date   Year Month Day   Department State K9_Unit
##    <chr>   <chr> <chr>          <chr> <dbl> <chr> <chr> <chr>      <chr>   <dbl>
##  1 Consta~ Dari~ Stabbed        Mond~  1791 Janu~ Mond~ Albany Co~ New ~       0
##  2 Sheriff Corn~ Gunfire        Satu~  1791 Octo~ Satu~ Columbia ~ New ~       0
##  3 Deputy~ Isaa~ Gunfire        Thur~  1792 May   Thur~ Westchest~ New ~       0
##  4 Marshal Robe~ Gunfire        Satu~  1794 Janu~ Satu~ United St~ Unit~       0
##  5 Deputy~ Robe~ Gunfire        Thur~  1797 June  Thur~ New York ~ New ~       0
##  6 Sheriff Robe~ Gunfire        Sund~  1797 Nove~ Sund~ Washingto~ Sout~       0
##  7 Superi~ Hila~ Duty related ~ Tues~  1798 Sept~ Tues~ Philadelp~ Penn~       0
##  8 High S~ John~ Gunfire        Tues~  1804 Octo~ Tues~ Mecklenbu~ Nort~       0
##  9 Watchm~ Chri~ Stabbed        Thur~  1806 Dece~ Thur~ New York ~ New ~       0
## 10 Deputy~ John~ Gunfire        Satu~  1807 March Satu~ Livingsto~ Kent~       0
## # ... with more rows
```

```
SELECT *
FROM police_death
```

Table 1: Displaying records 1 - 10

Rank	Name	Cause_of_Death	Date	Year	Month	Day	Department	State	K9_Unit
Constable	Darius Quimby	Stabbed	Monday, January 3, 1791	1791	January	Monday	Albany County Constable's Office, NY	New York	0
Sheriff	Cornelius Hogeboom	Gunfire	Saturday, October 22, 1791	1791	October	Saturday	Columbia County Sheriff's Office, NY	New York	0
Deputy Sheriff	Isaac Smith	Gunfire	Thursday, May 17, 1792	1792	May	Thursday	Westchester County Sheriff's Department, NY	New York	0
Marshal	Robert Forsyth	Gunfire	Saturday, January 11, 1794	1794	January	Saturday	United States Department of Justice - United States Marshals Service, US	United States	0
Deputy Sheriff	Robert Berwick	Gunfire	Thursday, June 29, 1797	1797	June	Thursday	New York County Sheriff's Office, NY	New York	0
Sheriff	Robert Maxwell	Gunfire	Sunday, November 12, 1797	1797	November	Sunday	Washington District Sheriff's Office, SC	South Carolina	0
Superintendent	Hilary Baker	Duty related illness	Tuesday, September 25, 1798	1798	September	Tuesday	Philadelphia Police Department, PA	Pennsylvania	0
High Sheriff	John Caldwell Cook	Gunfire	Tuesday, October 16, 1804	1804	October	Tuesday	Mecklenburg County Sheriff's Office, NC	North Carolina	0

Rank	Name	Cause_of_Death	Date	Year	Month	Day	Department	State	K9_Unit
Watchman	Christian Luswanger	Stabbed	Thursday, December 25, 1806	1806	December	Thursday	New York City Night Watch, NY	New York	0
Deputy Sheriff	John A. Gooch	Gunfire	Saturday, March 7, 1807	1807	March	Saturday	Livingston County Sheriff's Department, KY	Kentucky	0

Now we're talking! Now that we have *police_death* as an SQL table, we can apply SQL commands like we just did. Here we merely applied a basic command to open up all the information available to us. Now that everything's up and running we can start asking ourselves some interesting questions about the data.

What caused the most deaths?

We see here that the column *Cause_of_Death* relates to us in what manner the police officer died. While this in and of itself is useful, perhaps we could make it even more useful by applying an aggregate function to discern what caused the most police deaths.

```
-- Total Deaths by Types
SELECT Cause_of_Death, COUNT(*) AS Deaths
FROM police_death
GROUP BY Cause_of_Death
ORDER BY Deaths DESC
```

Table 2: Displaying records 1 - 10

Cause_of_Death	Deaths
Gunfire	13105
Automobile crash	2567
Heart attack	1216
Motorcycle crash	1167
Vehicular assault	1000
Struck by vehicle	986
COVID19	771
Assault	712
Gunfire (Inadvertent)	693
Vehicle pursuit	692

Perhaps predictably, the most common cause of death among officers was gunfire in the line of action. This followed by Automobile crashes, Heart Attacks, Motorcycle crashes, and vehicular assault.

Let's apply this same query to other categories

Which Ranks died the most?

```
-- Total Deaths by Rank
SELECT Rank, COUNT(*) AS Deaths
FROM police_death
GROUP BY Rank
ORDER BY Deaths DESC
```

Table 3: Displaying records 1 - 10

Rank	Deaths
Patrolman	3876
Police Officer	3549
Deputy Sheriff	3224
Officer	1746
Sergeant	1423
Detective	861
Trooper	857
Sheriff	775
Chief of Police	691
Constable	589

Here the top 5 ranks of those that died are more evenly spread by the look of it. Patrolman clocks in at the most deaths, which makes sense considering that these would be the policemen more out on the field most out on the field. There less deaths the ranks has, the more likely it seems to be a senior or administrative role. Another possibility could also be that there are some roles whose duties are similar, yet for whatever reason function under different titles or might be more obscure.

Number of Deaths by Year

```
-- Number of Deaths by Year
SELECT Year, COUNT(*) AS Deaths
FROM police_death
WHERE Year >= 1930
GROUP BY Year
ORDER BY Deaths DESC
```

Table 4: Displaying records 1 - 10

Year	Deaths
2021	643
2020	436
1930	347
1932	327
1931	318
1974	289
1973	279
1934	274
1933	266
1935	257

So, we changed things up a little for measuring the deaths by year. We restricted the analysis to the year 1930 onwards since reports prior to that tend to be more sparse. So for the sake of slimming the analysis, we went with that restriction.

Shockingly, *2021* and *2020* seem to be some of the most violent years in terms of police death in memory. Not only that, but going on the basis of the dataset, they are the two most violent years for police officers in US history. Both *1930* and *1932* trail behind it.

Segmenting Number of Deaths by Cause and Rank

```
-- Number of Deaths broken down by Rank and Cause
SELECT Rank, Cause_of_Death AS Cause, COUNT(*) AS Deaths
FROM police_death
WHERE Year >= 1930
GROUP BY Rank, Cause
HAVING Deaths >= 200
ORDER BY Deaths DESC
```

Table 5: Displaying records 1 - 10

Rank	Cause	Deaths
Police Officer	Gunfire	1191
Patrolman	Gunfire	1058
Deputy Sheriff	Gunfire	991
Officer	Gunfire	532
Sergeant	Gunfire	462
Deputy Sheriff	Automobile crash	419
Detective	Gunfire	328
Police Officer	Automobile crash	282
Patrolman	Motorcycle crash	259
Patrolman	Automobile crash	247

Here we segmented deaths in accordance to police officer Rank and the Cause. Again, to streamline our analysis, I limited the rows to show only those situations in which deaths exceeded 200. I did this utilizing a *HAVING* statement. The top 5 deaths are related to Gunfire, which is consistent with the results we found earlier.

But let's make that ranking official using the *RANK* window function.

Ranking of Death Causes

```
-- Ranking the Causes of Death
SELECT Rank, Cause_of_Death AS Cause, COUNT(*) AS Deaths,
RANK() OVER(ORDER BY COUNT(*) DESC) AS Rank
FROM police_death
WHERE Year >= 1930
GROUP BY Cause, Rank
HAVING Deaths >= 200
LIMIT 5
```

Table 6: 5 records

Rank	Cause	Deaths	Rank
Police Officer	Gunfire	1191	1
Patrolman	Gunfire	1058	2
Deputy Sheriff	Gunfire	991	3

Rank	Cause	Deaths	Rank
Officer	Gunfire	532	4
Sergeant	Gunfire	462	5

Here are our ranks. A pretty nifty feature if I might say so myself.

Peering through Window Functions

We toyed around a bit with window functions earlier by using the *RANK* function. However, with this nice feature, we can do a fair bit more things. But before we get to that, it would be prudent to create a couple Views in SQL to facilitate the analysis. The Views will be useful in that, they store the result of a query as though it were a table.

What does this mean?

Well, for our purposes, the primary aim in this is so we can apply Aggregate Functions to Aliased column 'Deaths'. Deaths being the column we created in our prior queries in order to aggregate the counts of individually reported deaths. If we can perform aggregate functions on 'Deaths' it will allow us to query interesting results we were not able to query earlier

Creating the Yearly Death View

Our first view shall be for looking at the total police deaths reported per year

```
-- Number of Deaths by Year
CREATE VIEW [Year Deaths] AS
SELECT Year, COUNT(*) AS Deaths
FROM police_death
GROUP BY Year;

SELECT * FROM [Year Deaths];

SELECT AVG(Deaths) FROM [Year Deaths];
```

Creating the Yearly Death View by Rank

Next we'll create another view that shall serve our purposes more generally.

```
-- Yearly Ranked Deaths by Year
CREATE VIEW [Yearly Rank Deaths] AS
SELECT Rank,
      Cause_of_Death,
      Year,
      COUNT(*) AS Deaths
FROM police_death
GROUP BY Year, Rank;
```

```
SELECT * FROM [Yearly Rank Deaths]
ORDER BY Year DESC
```

Table 7: Displaying records 1 - 10

Rank	Cause_of_Death	Year	Deaths
Agent	COVID19	2022	1
Border Patrol Agent	Automobile crash	2022	1
Captain	COVID19	2022	2
Chief Deputy Sheriff	Gunfire	2022	1
Chief of Police	COVID19	2022	3
Corporal	COVID19	2022	6
Correctional Lieutenant	COVID19	2022	1
Correctional Officer	COVID19	2022	1
Correctional Officer III	Heart attack	2022	1
Correctional Officer IV	COVID19	2022	1

Window Functions to find Cumulative Sum of Deaths Now, if we wanted to find see how deaths accumulated over time, we need only employ a window function for our purposes. Using an **OVER** and **PARTITION BY** statement, we can create a new column called **Accumulated_Deaths** which will show how each death increases per year by cause of death. I also included a **CASE** statement that shall evaluate when the Death count for a particular year (in case of gunfire) exceeds three deaths in a year

It should be noted that I shall be restricting the analysis to Gunfire Deaths, as such deaths are the most common, and it's easier to read what's going on with the accumulated deaths by restricting them to one cause. Similarly, I filtered the date to only those police deaths registered on or after 1930. The query itself can easily be tweaked so as to accomodate for other causes of deaths or years.

```
SELECT Year, Cause_of_Death, Deaths,
SUM(Deaths) OVER(PARTITION BY Cause_of_Death ORDER BY Year ASC) AS Accumulated_Deaths,
CASE WHEN
  Deaths > 3 THEN 1
  ELSE 0
  END AS "High_Death_Year"
FROM [Yearly Rank Deaths]
WHERE Cause_of_Death = "Gunfire" AND Year >= 1930
GROUP BY Cause_of_Death, Year
ORDER BY Year ASC
```

Table 8: Displaying records 1 - 10

Year	Cause_of_Death	Deaths	Accumulated_Deaths	High_Death_Year
1930	Gunfire	1	1	0
1931	Gunfire	4	5	1
1932	Gunfire	1	6	0
1933	Gunfire	1	7	0
1934	Gunfire	3	10	0
1935	Gunfire	1	11	0
1936	Gunfire	1	12	0
1937	Gunfire	1	13	0
1938	Gunfire	1	14	0
1939	Gunfire	3	17	0

Subqueries to Determine High Death Years Here, we'll put to work the first view we created, which focused on the deaths grouped by year. What I'm interested in doing is comparing years to each other in

the dataset, and to this end, I figured that finding out the severity in terms of death for a given year would be of interest. Notice that to do this I used a subquery to find the average deaths from the dataset filtered for 1930. Then, I used a **CASE** statement where I compared the total sum of deaths in a given year to the average deaths of said year times 2 or divided by 2–using either metric to determine whether the year had high or lower deaths than usual. This too was accomplished with a subquery as can be seen in the denominator.

```
-- Determining High Death Years
SELECT Year, Deaths,
(SELECT AVG(DEATHS) FROM [Year Deaths] WHERE Year >= 1930) AS Average_Deaths,
CASE
  WHEN Deaths > (SELECT AVG(DEATHS) FROM [Year Deaths]) * 2 THEN 'High Death Year'
  WHEN Deaths < (SELECT AVG(DEATHS) FROM [Year Deaths]) /2 THEN 'Low Death Year'
  ELSE 'Normal Year'
  END AS Year_Class
FROM [Year Deaths]
WHERE Year >= 1930
GROUP BY Year
ORDER BY Year DESC
```

Table 9: Displaying records 1 - 10

Year	Deaths	Average_Deaths	Year_Class
2022	121	194.3656	Normal Year
2021	643	194.3656	High Death Year
2020	436	194.3656	High Death Year
2019	186	194.3656	Normal Year
2018	216	194.3656	Normal Year
2017	210	194.3656	Normal Year
2016	218	194.3656	Normal Year
2015	194	194.3656	Normal Year
2014	183	194.3656	Normal Year
2013	157	194.3656	Normal Year

Conclusion

This was a perfect occasion and dataset to test out various functions in SQL. As you can see, there was plenty of useful information which we may not have been able to see otherwise if we'd used another program.

As for me, I'm happy that you took the time to read through this. Have a good one!