Analysis of Traffic Stop Receipts in Buffalo, NY

Ryan Glasser
rtglasse

Rian Casimir
riancasi

Vatsalya Anand
vatsalya

I. Project Overview

This project and supplemental report serves to investigate, analyze, and communicate findings about records of traffic stop receipts issued in the City of Buffalo and its thirty-five neighborhoods from October 1, 2022 to October 17, 2023.

II. Problem Statement

A. Problem Discussion and Questions

It is paramount for the City of Buffalo to keep receipts of the traffic stops they issue to individuals on a daily basis so there is a clear record of law violations for citizens in an effort to sustain public safety. At the same time, it is often the case and trend that high frequency issues of traffic stop receipts in certain regions can correspond to lower standards of living, socioeconomic status, and greater levels of poverty for those in and around that region. Analyzing a database that contains a number of features related to a traffic stop, including, but not limited to, demographic information of individuals such as age, race, and ethnicity, and location information, such as the neighborhood, council district, and police district that a traffic stop receipt is issued can give vital information as to how city officials may allocate resources and implement programs to not only improve public safety, but ensure a more sustainable future for the neighborhoods of Buffalo.

There are a multitude of reasons that require the analysis of such a database as traffic stop receipts in the City of Buffalo in an effort to solve problems across several domains. The City of Buffalo is one of the most impoverished metropolitan areas in the United States, and oftentimes government officials only truly analyze what are deemed the most contributing factors to poverty, such as crime rates, household income, and educational experience, while not focusing on factors that may not be as obvious, such as traffic stop issues. The analysis of this database brings in an additional perspective to the much broader issue of poverty that the city faces. This database's analysis intends to uncover information that can be used to address some less obvious issues that contribute to poverty in the city, and aid in curbing such trends for the better and moving towards a safer and healthier Buffalo.

Some of the questions we wish to answer are as follows:

1) What neighborhoods in the City of Buffalo over the analyzed time period have had the highest rate of traffic stop violations? How can this be used to implement programs that focus on public safety sustainability in those specific areas?
2) Is there a higher frequency of traffic violations at specific times? Should there be more focus on better monitoring during those specific times in an attempt to curb those higher traffic issue rates?
3) How do personal demographics play a role in traffic stop receipts issued? Is there a disparity in how traffic stop receipts are issued based on certain demographics?
4) Is there a difference in traffic stop details based on whether a police officer has a body camera on their person or not?
5) What kinds of violations are most prevalent? Is there a specific street-level location that sees many traffic stop receipts issued that needs the attention of city officials?

B. Reason for Database Management

A database is needed instead of an Excel file due to the fact that insertions are needed for this database. Every day, new traffic stop receipts are issued, and an efficient mechanism is needed for the insertion of this information. In addition, as more traffic receipts are issued, trends amongst the data may change over time. A database is necessary to effectively capture and analyze such changes over time, as new data streams in daily.

C. Background and Significance of Problem

Poverty has had a grip on the City of Buffalo for decades, characterized by lower standards of living, high crime rates, food insecurity, inadequate access to healthcare, and significant economic and education deficiencies and gaps for a substantial proportion of those who live in the region's neighborhoods. Rates of poverty are greater among children and minority individuals. This is a significant problem because such issues revolving around poverty greatly hinder individuals' abilities to navigate their daily lives and provide for themselves and families. People are living paycheck to paycheck and frankly, are working just to live. Those individuals deserve equal access to services that the other percentage of the population does not have to worry about not having access to, and understanding underlying issues contributing to this issue, would allow for government officials to pinpoint regions and/or sub-populations of need.

D. Potential For Project Contribution to Problem Domain

This project has the potential to catalyze government and city officials to focus on areas or sub-populations that may not have been of primary concern in the past as it relates to curbing poverty rates in the City of Buffalo. While it is important to address crime and access to education, healthcare, and food, there is also an importance in addressing other smaller scale issues that over time may have added up and also contributed to the problem as a whole. There is potential to make significant improvements for the sub-populations of interest and supporting those individuals in an effort to make sure they rise above the poverty line, and do not fall below it in the future. Government response to such dire issues can be the difference between life and death for some. Thus, stating that the analysis of such data is crucial is an understatement. Strides can be made in addressing public safety by deploying resources to areas that need aid, by means of welfare programs, and as it directly relates to the traffic stop receipt database, educating individuals through driver education programs, or fixing erroneous road conditions that lead to traffic stops as a result of insufficient city infrastructure and failure to meet safety standards.

III. Target User

A. Database Users

The database can be utilized by several groups of people. City and government officials that wish to analyze various demographic and regional trends can use this database to pinpoint sub-populations and areas of interest to provide better programs to curb violations, which in turn can curb poverty rates. Also, databases of this nature are publicly available, as the name of the person that a traffic stop receipt is issued to is not accessible. This database may be used to examine demographic trends, rather than individual and personal information for a specific individual. Thus, politicians and those running for government office may find this information useful in campaigning to voters as to how they can tackle outstanding issues as it relates to traffic violations and poverty, and how they intend on solving those problems, whether it be by implementing welfare programs or improving city infrastructure. In addition, those who simply wish to educate themselves on community demographic trends in the City of Buffalo can use this database. People who live in the City of Buffalo that are not issued traffic stop receipts, but still may be concerned about the safety of the area they live in, or activists that wish to lobby city officials and politicians based on public safety and its potential effect on city poverty levels may also intend to use this database.

B. Database Administers

The database will be administered by a select group of city officials and clerks. Typically, the New York State Attorney General's office would be in charge of matters such as traffic stop receipts, and updating records on a daily basis. Beyond this, the Buffalo Police Department may administer the database, as officers directly issue receipts to individuals. Thus, as it directly relates to this specific database regarding traffic stop receipts issued in the City of Buffalo, city officials and clerks, specifically those in the New York State Attorney General's office downtown, as well as the Buffalo Police Department will administer the database.

C. Scenarios of Database Management

Scenario 1: Traffic Stop Incident and Insertion of Records

A police officer in the City of Buffalo is monitoring the University Heights neighborhood and encounters an individual blowing through a stop sign. The officer pulls the individual over, and gathers the individual's information from their driver's license and vehicle registration. After the traffic stop is completed, that information is relayed to the Attorney General's office, where a clerk or other official inserts records into the database on a daily basis. The fields inserted into the database include, but are not limited to, a traffic stop identification number, reason for the stop, date and time of issuance, location of traffic stop, and the ace, ethnicity, and age of the individual issued the receipt.

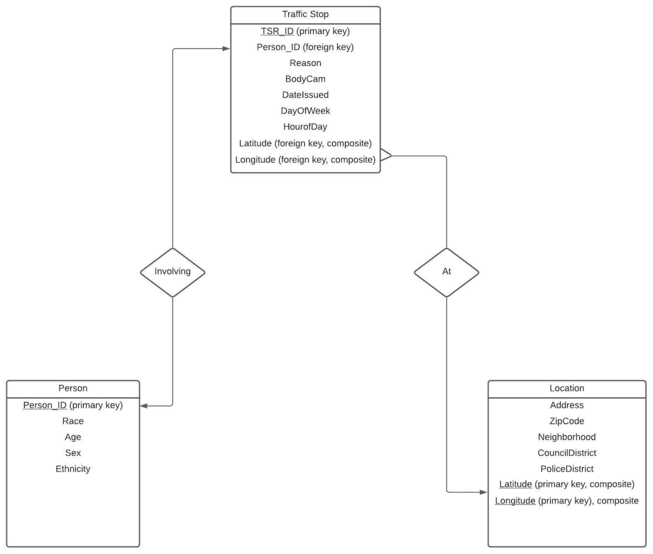
Scenario 2: Use of Database to Analyze Demographic and Location Data

A political campaign for a candidate running for city mayor wishes to understand the frequency of traffic stop receipts issued over the past 5 years based on race, age bins, and which neighborhoods traffic stop receipt issues are greatest. After querying data for the time period of interest, it is discovered that there is a greater proportion of traffic stop receipts issued to individuals of minority races, individuals 16-25 years of age make up the most speeding violations, and the most receipts are issued in University Heights and Broadway-Fillmore of the thirty-five neighborhoods that make up the city. As a result, that government hopeful focuses their campaign on how they would solve those problems by ensuring public safety and curbing poverty if elected, in an effort to appeal to voters.

IV. Entity-Relationship Diagram

A. Figure

The figure below depicts the entity-relationship diagram for the traffic stop database.



B. Relationships Between Tables

The “TrafficStop” table relates to the “Person” table by means of an attribute “Involving”. That is, the “TrafficStop” table that contains the main information regarding the incident, including, but not limited to, a foreign key “Person_Id”, reason for the traffic stop, and date and time details, relates to the “Person” table, where the “Person_Id” is the primary key, and contains tuples pertaining to the individual involved in that traffic incident. This is a one-to-one relationship, as one traffic stop pertains to exactly one person, and conversely, one individual on record (each entry is a unique incident with no identifiable information for the individual recorded) pertains to exactly one traffic incident. While a “TSR_ID” uniquely identifies a person and record, a separate table for “Person” is created for organizational purposes.

The “TrafficStop” table relates to the “Location” table by means of an attribute “At”. That is, the “TrafficStop” table that contains the main information regarding the incident, including, but not limited to, a foreign key that is a composite key consisting of “Latitude” and “Longitude”, reason for the traffic stop, and date and time details, relates to the “Location” table, where the “Latitude” and “Longitude” is the primary key, and contains tuples pertaining to the location at which a traffic receipt was issued. This is a many-to-one relationship, as many traffic stops can occur at one location (in the event that latitude and longitude are the same for different receipts issued), and conversely, one location can relate to many traffic stop incidents.

C. Functional Dependencies, and Assurance of Boyce-Codd Normal Form For Each Relation

All relations of the database are in Boyce-Codd Normal Form. This is justified for each of the relations as follows:

“TrafficStop” functional dependencies:

- TSR_ID $\rightarrow$ Person_ID, Latitude, Longitude, Reason, BodyCam, DateIssued, DayOfWeek, HourOfDay
- Person_ID $\rightarrow$ TSR_ID, Latitude, Longitude, Reason, BodyCam, DateIssued, DayOfWeek, HourOfDay

For all non trivial functional dependencies holding in ”TrafficStop”, that is the one above, *TSR_ID* and Person_ID are candidate keys. This is true as both *TSR_ID* and *Person_ID* on their can own uniquely identify all details of a traffic stop, namely all other attributes in the relation.

“Person” functional dependencies:

- Person_ID $\rightarrow$ Race, Age, Sex, Ethnicity

For all non trivial functional dependencies holding in ”Person”, that is the one above, *Person_ID* is a candidate key. This is true as *Person_ID* uniquely identifies all details related to a person involved in the traffic stop.

“Location” function dependencies:

- Latitude, Longitude $\rightarrow$ Address, ZipCode, Neighborhood, CouncilDistrict, PoliceDistrict

For all non trivial functional dependencies holding in ”Location”, that is the one above, *Latitude*, *Longitude* is a candidate key. This is true as *Latitude* and *Longitude* uniquely identify all location-based details related to a traffic stop.

D. Transformation from Initial Schema to Boyce-Codd Normal Form

The initial schema was taken from Open Data Buffalo, a public repository of data sets related to the City of Buffalo (https://data.buffalony.gov/Public-Safety/Traffic-Stop-Receipts/8mqs-6g9h). The data set originally contained 14.3K rows and 32 columns. For the purposes of this project, three relations were created by taking subsets of columns from the data set to create an organized and coherent database. For each of the three relations, the primary key uniquely determines all other attributes, with no other functional dependencies present, as each record is a unique instance. Thus, for each of the relations, there is a functional dependency that is nontrivial, in which the primary key determines all other attributes. That is the primary key is a candidate key. Thus, all relations are Boyce-Codd Normal Form.

V. Database Specifics: Relations and Their Attributes

Relation 1: TrafficStop

TrafficStop(*TSR_ID*, *Person_ID*, *Reason*, *BodyCam*, *DateIssued*, *DayOfWeek*, *HourOfDay*, *Latitude*, *Longitude*)

Primary Key:

- *TSR_ID* - unique identifier for a traffic stop

Foreign Keys:

- *Person_ID* references *Person*: ON DELETE CASCADE and ON UPDATE CASCADE are employed where all rows referenced in the child table (*Person*) are deleted/updated when the corresponding row is deleted/updated from the parent table (*TrafficStop*).
- *Latitude*, *Longitude* references *Location*: No action is taken with regards to the deletion of rows in the child table (*Location*) when a corresponding row in the parent table (*TrafficStop*) is deleted, as although rare, a specific location still may be relevant for other traffic stops in the database

Attributes:

- *TSR_ID* (integer, not null): unique number for a traffic stop
- *Person_ID* (integer, not null): unique number for a person involved at a traffic stop
- *Reason* (varchar, not null): brief description of why a traffic stop receipt was issued
- *BodyCam* (boolean, default null): true or false value as to whether the officer issuing a traffic stop receipt had a body camera on person
- *DateIssued* (timestamp, not null): date and time the traffic stop receipt was issued
- *DayOfWeek* (varchar, default null): day of the week the traffic stop receipt was issued

- *HourOfDay* (varchar, default null): hour of the day the traffic stop receipt was issued
- *Latitude* (double precision, not null): latitude where the traffic stop receipt was issued
- *Longitude* (double precision, not null): longitude where the traffic stop receipt was issued

Relation 2: Person
Person(*Person_ID*, *Race*, *Age*, *Sex*, *Ethnicity*
Primary Key:

- *Person_ID* - unique identifier for a person involved in a traffic stop

Foreign Keys:

- None

Attributes:

- *Person_ID* (integer, not null) - unique number for a person involved in a traffic stop
- *Race* (varchar, default 'UNKNOWN') - race of the person who received the traffic stop receipt
- *Age* (integer, not null) - age of the person at the time of receiving the traffic stop receipt
- *Sex* (char(1), not null) - sex of the person who received the traffic stop receipt
- *Ethnicity* (varchar, default 'UNKNOWN') - ethnicity of the person who received the traffic stop receipt

Relation 3: Location
Location(*Address*, *ZipCode*, *Neighborhood*, *CouncilDistrict*, *PoliceDistrict*, *Latitude*, *Longitude*)
Primary Key:

- *Latitude, Longitude* - unique identifier for a specific place at which a traffic stop receipt is issued

Foreign Keys:

- None

Attributes:

- *Address* (varchar, not null) - approximate address where the traffic stop receipt was issued
- *ZipCode* (varchar, default 'UNKNOWN') - zip code where the traffic stop receipt was issued
- *Neighborhood* (varchar, default 'UNKNOWN') - neighborhood where the traffic stop receipt was issued
- *CouncilDistrict* (varchar, default 'UNKNOWN') - council district where the traffic stop receipt was issued
- *PoliceDistrict* (varchar, default 'UNKNOWN') - police district where the traffic stop receipt was issued
- *Latitude* (double precision, not null) - latitude where the traffic stop receipt was issued
- *Longitude* (double precision, not null) - longitude where the traffic stop receipt was issued

VI. Dataset Imports Into PostgreSQL

In terms of handling the copy and importing of data sets from our local machine upon downloading from the Open Data Buffalo public repository, problems arose when it came to granting permissions to PostgreSQL, the database management system used for this project, to read the CSV files of data. In an attempt to solve this issue, proper permissions were given by navigating to the local machine's system preferences, and adjusting the security settings appropriately. Beyond this, a few separate terminal commands were needed to enable proper reading and importing of data into the three relations. In terms of adopting indexing concepts, such was not necessary. No lazy loading of data was needed, and imports of data from the CSV files into the relations were quick and efficient as the amount of data being copied into the table was not large enough to amount to any issues. In terms of the number of rows in each relation specifically, the TrafficStop and Person relations contain 4,323 rows, while the Location relation contains 2,719 rows.

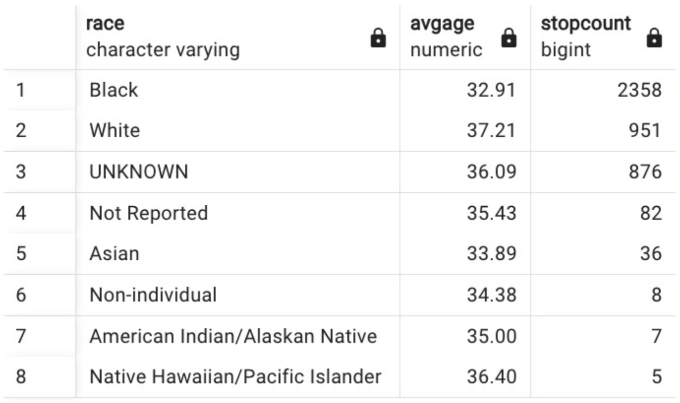
One other problem that came up upon copying data from the three CSV files into the relations of the database dealt with the Location relation and its primary key of (Latitude, Longitude). Since the latitude and longitude coordinates uniquely identify a row in that relation, when two rows with the same primary key appear, namely the same latitude and longitude coordinates, the other values in the row should match. But, even after adjusting the CSV file to remove duplicate rows for the purposes of our database, copying errors arose due to the fact that some rows, while having the same primary key, had values in other columns that did not match. For instance, two rows may have matched on all columns, except for their address, where one entry's address was inputted as "Corner of Street A Street B", and the other's was inputted as "Corner of Street B Street A". Thus, while the location of the traffic stop receipt issuance was the same, import errors arose due to the way information was inputted. There were very few instances in which such an error occurred when importing the data into the relations (less than five cases), and in each case PostgreSQL helpfully provided the row number of the discrepancy, and that corresponding row was deleted from the CSV file, and thus was not imported into the database.

VII. Testing Database with SQL Queries

This section intends to test the database and ensure its workings by executing SQL queries. A total of twelve queries are carried out (6 'SELECT', 2 'INSERT', 2 'UPDATE', and 2 'DELETE'), and their results before and after execution are displayed when applicable.

Query 1: Find the average age and the number of traffic stops grouped by race

```
SELECT Person.Race,
    ROUND(AVG(Person.Age), 2) AS AvgAge,
    COUNT(*) AS StopCount
FROM Person
JOIN TrafficStop ON
    Person.Person_ID = TrafficStop.Person_ID
GROUP BY Person.Race
ORDER BY StopCount DESC;
```

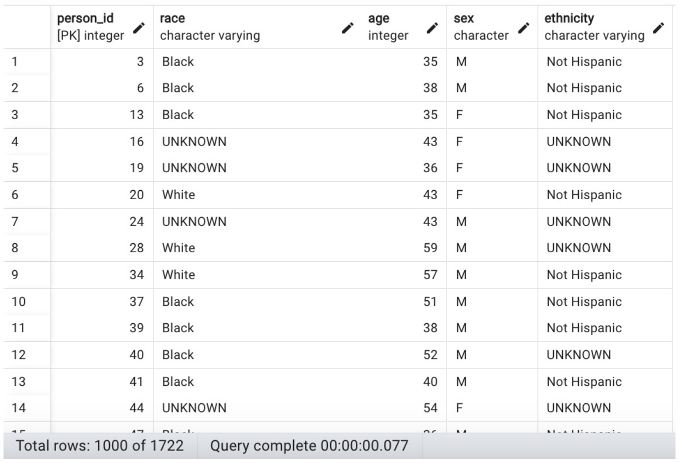


	race character varying	avgage numeric	stopcount bigint
1	Black	32.91	2358
2	White	37.21	951
3	UNKNOWN	36.09	876
4	Not Reported	35.43	82
5	Asian	33.89	36
6	Non-individual	34.38	8
7	American Indian/Alaskan Native	35.00	7
8	Native Hawaiian/Pacific Islander	36.40	5

Fig. 1. Execution results for Query 1

Query 2: Find persons involved in traffic stops who are older than the average age

```
SELECT Person_ID, Race, Age, Sex, Ethnicity
FROM Person
WHERE Age > (SELECT AVG(Age) FROM PERSON);
```



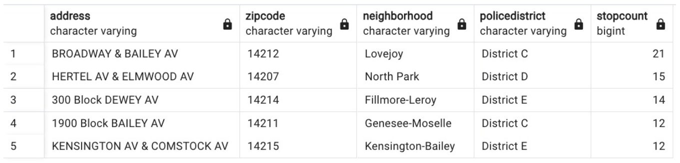
	person_id [PK] integer	race character varying	age integer	sex character	ethnicity character varying
1	3	Black	35	M	Not Hispanic
2	6	Black	38	M	Not Hispanic
3	13	Black	35	F	Not Hispanic
4	16	UNKNOWN	43	F	UNKNOWN
5	19	UNKNOWN	36	F	UNKNOWN
6	20	White	43	F	Not Hispanic
7	24	UNKNOWN	43	M	UNKNOWN
8	28	White	59	M	UNKNOWN
9	34	White	57	M	Not Hispanic
10	37	Black	51	M	Not Hispanic
11	39	Black	38	M	Not Hispanic
12	40	Black	52	M	UNKNOWN
13	41	Black	40	M	Not Hispanic
14	44	UNKNOWN	54	F	UNKNOWN

Total rows: 1000 of 1722 Query complete 00:00:00.077

Fig. 2. Execution results for Query 2

Query 3: Identify the 5 locations (addresses) with the highest number of traffic stops

```
SELECT Location.Address, Location.ZipCode,
    Location.Neighborhood,
    Location.PoliceDistrict,
    COUNT(*) as StopCount
FROM TrafficStop
JOIN Location ON
    TrafficStop.Latitude =
        Location.Latitude
    AND TrafficStop.Longitude =
        Location.Longitude
GROUP BY Location.Address,
    Location.ZipCode,
    Location.Neighborhood,
    Location.PoliceDistrict
ORDER BY StopCount DESC
LIMIT 5;
```

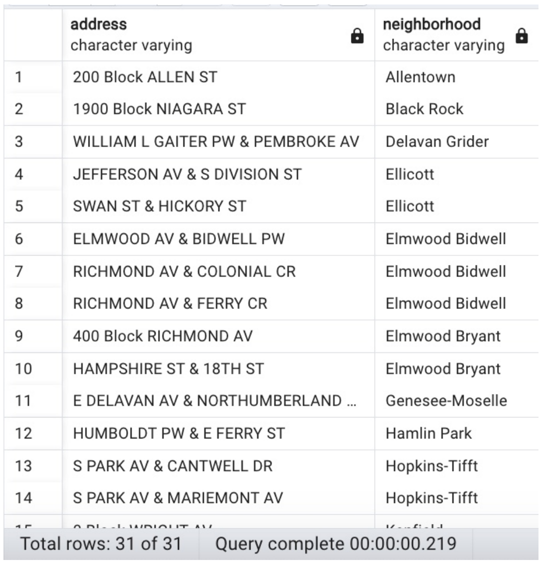


	address character varying	zipcode character varying	neighborhood character varying	policedistrict character varying	stopcount bigint
1	BROADWAY & BAILEY AV	14212	Lovejoy	District C	21
2	HERTEL AV & ELMWOOD AV	14207	North Park	District D	15
3	300 Block DEWEY AV	14214	Fillmore-Leroy	District E	14
4	1900 Block BAILEY AV	14211	Genesee-Moselle	District C	12
5	KENSINGTON AV & COMSTOCK AV	14215	Kensington-Bailey	District E	12

Fig. 3. Execution results for Query 3

Query 4: Find the locations (addresses and neighborhoods) where traffic stops without body cameras were issued on weekends

```
SELECT Location.Address, Location.Neighborhood
FROM TrafficStop
JOIN Location ON
    TrafficStop.Latitude
        = Location.Latitude
    AND TrafficStop.Longitude
        = Location.Longitude
WHERE TrafficStop.BodyCam = FALSE
    AND (TrafficStop.DayOfWeek = 'SATURDAY'
    OR TrafficStop.DayOfWeek = 'SUNDAY')
GROUP BY Location.Neighborhood, Location.Address;
```



	address character varying	neighborhood character varying
1	200 Block ALLEN ST	Allentown
2	1900 Block NIAGARA ST	Black Rock
3	WILLIAM L GAITER PW & PEMBROKE AV	Delavan Grider
4	JEFFERSON AV & S DIVISION ST	Ellicott
5	SWAN ST & HICKORY ST	Ellicott
6	ELMWOOD AV & BIDWELL PW	Elmwood Bidwell
7	RICHMOND AV & COLONIAL CR	Elmwood Bidwell
8	RICHMOND AV & FERRY CR	Elmwood Bidwell
9	400 Block RICHMOND AV	Elmwood Bryant
10	HAMPSHIRE ST & 18TH ST	Elmwood Bryant
11	E DELAVAN AV & NORTHUMBERLAND ...	Genesee-Moselle
12	HUMBOLDT PW & E FERRY ST	Hamlin Park
13	S PARK AV & CANTWELL DR	Hopkins-Tifft
14	S PARK AV & MARIEMONT AV	Hopkins-Tifft

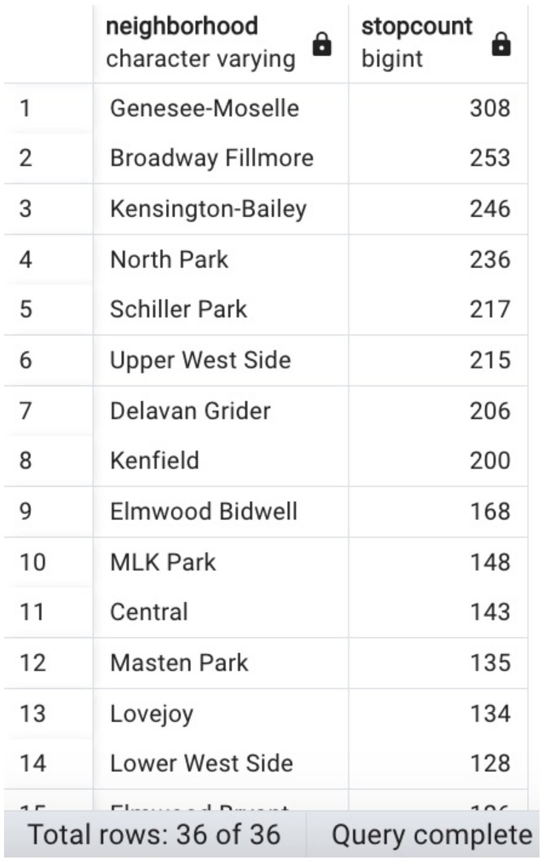
Total rows: 31 of 31 Query complete 00:00:00.219

Fig. 4. Execution results for Query 4

Query 5: Find the number of traffic stops grouped by neighborhood in descending order

```
SELECT Location.Neighborhood,
    COUNT(*) AS StopCount
FROM TrafficStop
JOIN Location
```

```
    ON TrafficStop.Latitude =
        Location.Latitude
    AND TrafficStop.Longitude =
        Location.Longitude
GROUP BY Location.Neighborhood
ORDER BY StopCount DESC;
```



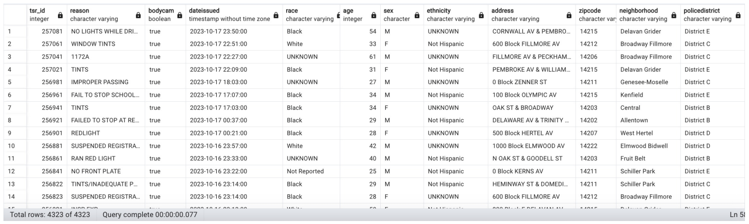
	neighborhood character varying	stopcount bigint
1	Genesee-Moselle	308
2	Broadway Fillmore	253
3	Kensington-Bailey	246
4	North Park	236
5	Schiller Park	217
6	Upper West Side	215
7	Delavan Grider	206
8	Kenfield	200
9	Elmwood Bidwell	168
10	MLK Park	148
11	Central	143
12	Masten Park	135
13	Lovejoy	134
14	Lower West Side	128

Total rows: 36 of 36 Query complete

Fig. 5. Execution results for Query 5

Query 6: Retrieve the details of each traffic stop, along with the person who was stopped, and location of the traffic stop, ordered by date issued in descending order

```
SELECT TrafficStop.TSR_ID,
    TrafficStop.Reason,
    TrafficStop.BodyCam,
    TrafficStop.DateIssued,
    Person.Race, Person.Age, Person.Sex,
    Person.Ethnicity, Location.Address,
    Location.ZipCode, Location.Neighborhood,
    Location.PoliceDistrict
FROM TrafficStop
JOIN Person
    ON TrafficStop.Person_ID =
        Person.Person_ID
JOIN Location
    ON TrafficStop.Latitude =
        Location.Latitude
    AND TrafficStop.Longitude =
        Location.Longitude
ORDER BY TrafficStop.DateIssued DESC;
```



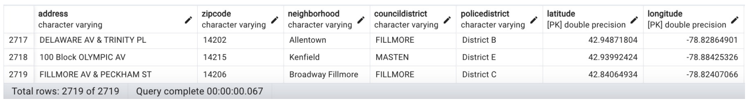
	tsr_id integer	reason character varying	bodycam boolean	dateissued timestamp without time zone	race character varying	age integer	sex character	ethnicity character varying	address character varying	zipcode character vary	neighborhood character varying	policedistrict character varying
1	257081	NO LIGHTS WHILE DRI...	true	2023-10-17 23:50:00	Black	54	M	UNKNOWN	CORNWALL AV & PEMBRO...	14215	Delavan Grider	District E
2	257061	WINDOW TINTS	true	2023-10-17 22:51:00	White	33	F	Not Hispanic	600 Block FILLMORE AV	14212	Broadway Fillmore	District C
3	257041	1172A	true	2023-10-17 22:27:00	UNKNOWN	61	M	UNKNOWN	FILLMORE AV & PECKHAM...	14206	Broadway Fillmore	District C
4	257021	TINTS	true	2023-10-17 22:09:00	Black	31	F	Not Hispanic	PEMBROKE AV & WILLIAM...	14215	Delavan Grider	District E
5	256981	IMPROPER PASSING	true	2023-10-17 18:03:00	UNKNOWN	27	M	UNKNOWN	0 Block ZENNER ST	14211	Genesee-Moselle	District C
6	256961	FAIL TO STOP SCHOOL...	true	2023-10-17 17:07:00	Black	34	M	Not Hispanic	100 Block OLYMPIC AV	14215	Kenfield	District E
7	256941	TINTS	true	2023-10-17 17:03:00	Black	34	F	UNKNOWN	OAK ST & BROADWAY	14203	Central	District B
8	256921	FAILED TO STOP AT RE...	true	2023-10-17 00:37:00	Black	29	M	Not Hispanic	DELAWARE AV & TRINITY ...	14202	Allentown	District B
9	256901	REDLIGHT	true	2023-10-17 00:21:00	Black	28	F	UNKNOWN	500 Block HERTEL AV	14207	West Hertel	District D
10	256881	SUSPENDED REGISTRA...	true	2023-10-16 23:57:00	White	42	M	UNKNOWN	1000 Block ELMWOOD AV	14222	Elmwood Bidwell	District D
11	256861	RAN RED LIGHT	true	2023-10-16 23:33:00	UNKNOWN	40	M	Not Hispanic	N OAK ST & GOODELL ST	14203	Fruit Belt	District B
12	256841	NO FRONT PLATE	true	2023-10-16 23:22:00	Not Reported	25	M	Not Hispanic	0 Block KERNS AV	14211	Schiller Park	District E
13	256822	TINTS/INADEQUATE P...	true	2023-10-16 23:14:00	Black	29	M	Not Hispanic	HEMINWAY ST & DOMEDI...	14211	Schiller Park	District C
14	256823	SUSPENDED REGISTRA...	true	2023-10-16 23:14:00	Black	28	F	UNKNOWN	600 Block FILLMORE AV	14212	Broadway Fillmore	District C

Total rows: 4323 of 4323 Query complete 00:00:00.077 Ln 58

Fig. 6. Execution results for Query 6

Query 7: Insert new location into Location table

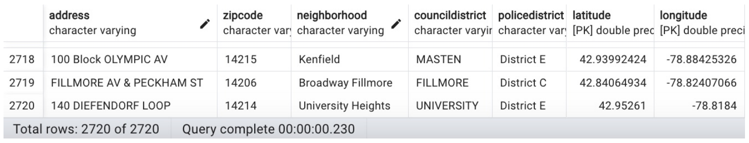
```
INSERT INTO Location
VALUES (’140 DIEFENDORF LOOP’, ’14214’,
    ’University Heights’, ’UNIVERSITY’,
    ’District E’, 42.952610, −78.818400);
```



	address character varying	zipcode character varying	neighborhood character varying	councildistrict character varying	policedistrict character varying	latitude [PK] double precision	longitude [PK] double precision
2717	DELAWARE AV & TRINITY PL	14202	Allentown	FILLMORE	District B	42.94871804	-78.82864901
2718	100 Block OLYMPIC AV	14215	Kenfield	MASTEN	District E	42.93992424	-78.88425326
2719	FILLMORE AV & PECKHAM ST	14206	Broadway Fillmore	FILLMORE	District C	42.84064934	-78.82407066

Total rows: 2719 of 2719 Query complete 00:00:00.067

Fig. 7. Execution results prior to the execution of Query 7



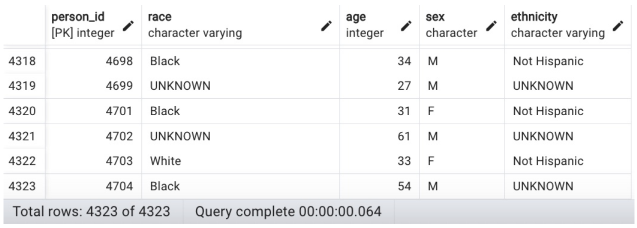
	address character varying	zipcode character var	neighborhood character varying	councildistrict character varyir	policedistrict character vary	latitude [PK] double prec	longitude [PK] double preci
2718	100 Block OLYMPIC AV	14215	Kenfield	MASTEN	District E	42.93992424	-78.88425326
2719	FILLMORE AV & PECKHAM ST	14206	Broadway Fillmore	FILLMORE	District C	42.84064934	-78.82407066
2720	140 DIEFENDORF LOOP	14214	University Heights	UNIVERSITY	District E	42.95261	-78.8184

Total rows: 2720 of 2720 Query complete 00:00:00.230

Fig. 8. Execution results following the execution of Query 7

Query 8: Insert new records into Person (perhaps from a certain day)

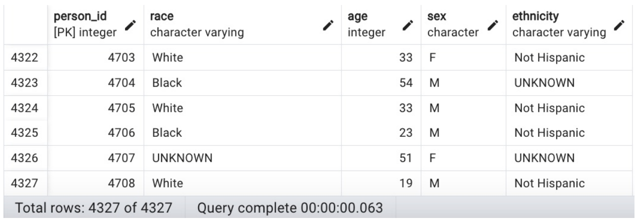
```
INSERT INTO Person
VALUES (4705, ’White’, 33, ’M’, ’Not Hispanic’),
    (4706, ’Black’, 23, ’M’, ’Not Hispanic’),
    (4707, DEFAULT, 51, ’F’, DEFAULT),
    (4708, ’White’, 19, ’M’, ’Not Hispanic’);
```



	person_id [PK] integer	race character varying	age integer	sex character	ethnicity character varying
4318	4698	Black	34	M	Not Hispanic
4319	4699	UNKNOWN	27	M	UNKNOWN
4320	4701	Black	31	F	Not Hispanic
4321	4702	UNKNOWN	61	M	UNKNOWN
4322	4703	White	33	F	Not Hispanic
4323	4704	Black	54	M	UNKNOWN

Total rows: 4323 of 4323 Query complete 00:00:00.064

Fig. 9. Execution results prior to the execution of Query 8



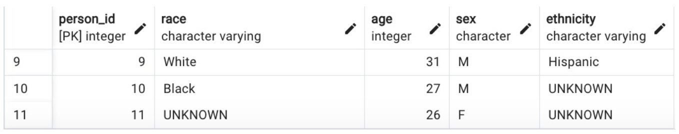
	person_id [PK] integer	race character varying	age integer	sex character	ethnicity character varying
4322	4703	White	33	F	Not Hispanic
4323	4704	Black	54	M	UNKNOWN
4324	4705	White	33	M	Not Hispanic
4325	4706	Black	23	M	Not Hispanic
4326	4707	UNKNOWN	51	F	UNKNOWN
4327	4708	White	19	M	Not Hispanic

Total rows: 4327 of 4327 Query complete 00:00:00.063

Fig. 10. Execution results following the execution of Query 8

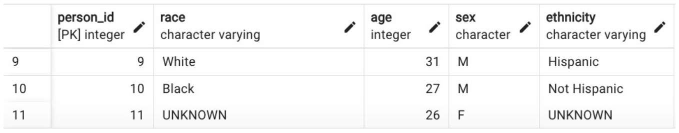
Query 9: Update the Ethnicity of person with Person_ID = 10, whose Ethnicity was previously set to 'UNKNOWN'

```
UPDATE Person
SET Ethnicity = 'Not Hispanic'
WHERE Person_ID = 10;
```



	person_id [PK] integer	race character varying	age integer	sex character	ethnicity character varying
9	9	White	31	M	Hispanic
10	10	Black	27	M	UNKNOWN
11	11	UNKNOWN	26	F	UNKNOWN

Fig. 11. Execution results prior to the execution of Query 9

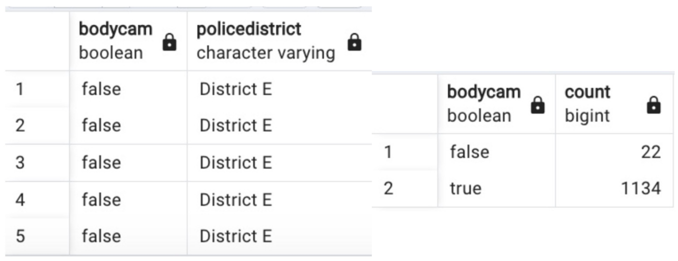


	person_id [PK] integer	race character varying	age integer	sex character	ethnicity character varying
9	9	White	31	M	Hispanic
10	10	Black	27	M	Not Hispanic
11	11	UNKNOWN	26	F	UNKNOWN

Fig. 12. Execution results following the execution of Query 9

Query 10: Update BodyCam to TRUE for all traffic stops in Police District E, as there is certainty all officers in that district wear body cameras

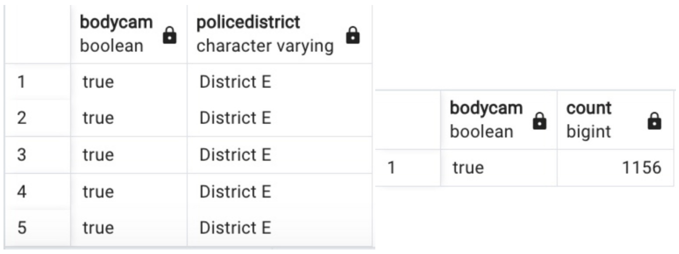
```
UPDATE TrafficStop
SET BodyCam = 'TRUE'
FROM Location
WHERE (TrafficStop.Latitude =
    Location.Latitude
    AND TrafficStop.Longitude =
        Location.Longitude)
AND PoliceDistrict = 'District E';
```



	bodycam boolean	policedistrict character varying
1	false	District E
2	false	District E
3	false	District E
4	false	District E
5	false	District E

	bodycam boolean	count bigint
1	false	22
2	true	1134

Fig. 13. Execution results prior to the execution of Query 10



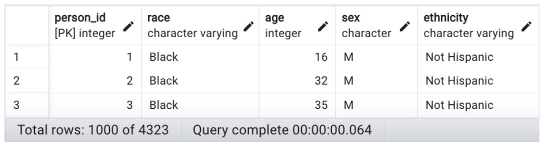
	bodycam boolean	policedistrict character varying
1	true	District E
2	true	District E
3	true	District E
4	true	District E
5	true	District E

	bodycam boolean	count bigint
1	true	1156

Fig. 14. Execution results following the execution of Query 10

Query 11: Delete the individual whose Person_ID = 2 from the Person table

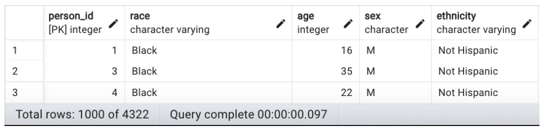
```
DELETE FROM Person
WHERE Person_ID = 2;
```



	person_id [PK] integer	race character varying	age integer	sex character	ethnicity character varying
1	1	Black	16	M	Not Hispanic
2	2	Black	32	M	Not Hispanic
3	3	Black	35	M	Not Hispanic

Total rows: 1000 of 4323 Query complete 00:00:00.064

Fig. 15. Execution results prior to the execution of Query 11



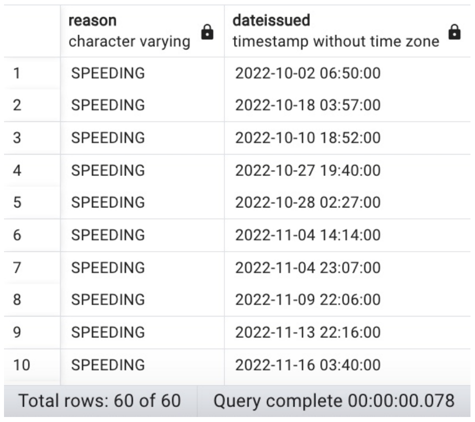
	person_id [PK] integer	race character varying	age integer	sex character	ethnicity character varying
1	1	Black	16	M	Not Hispanic
2	3	Black	35	M	Not Hispanic
3	4	Black	22	M	Not Hispanic

Total rows: 1000 of 4322 Query complete 00:00:00.097

Fig. 16. Execution results following the execution of Query 11

Query 12: Delete all traffic stops that occurred between 10/1/2022 and 11/4/2022 in which the reason for the issuance was 'SPEEDING'

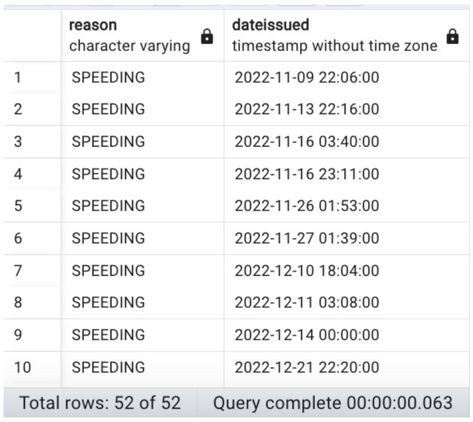
```
DELETE FROM TrafficStop
WHERE Reason = 'SPEEDING'
AND DateIssued >= '2022-10-01 00:00:00'
AND DateIssued <= '2022-11-04 23:59:59';
```



	reason character varying	dateissued timestamp without time zone
1	SPEEDING	2022-10-02 06:50:00
2	SPEEDING	2022-10-18 03:57:00
3	SPEEDING	2022-10-10 18:52:00
4	SPEEDING	2022-10-27 19:40:00
5	SPEEDING	2022-10-28 02:27:00
6	SPEEDING	2022-11-04 14:14:00
7	SPEEDING	2022-11-04 23:07:00
8	SPEEDING	2022-11-09 22:06:00
9	SPEEDING	2022-11-13 22:16:00
10	SPEEDING	2022-11-16 03:40:00

Total rows: 60 of 60 Query complete 00:00:00.078

Fig. 17. Execution results prior to the execution of Query 12



	reason character varying	dateissued timestamp without time zone
1	SPEEDING	2022-11-09 22:06:00
2	SPEEDING	2022-11-13 22:16:00
3	SPEEDING	2022-11-16 03:40:00
4	SPEEDING	2022-11-16 23:11:00
5	SPEEDING	2022-11-26 01:53:00
6	SPEEDING	2022-11-27 01:39:00
7	SPEEDING	2022-12-10 18:04:00
8	SPEEDING	2022-12-11 03:08:00
9	SPEEDING	2022-12-14 00:00:00
10	SPEEDING	2022-12-21 22:20:00

Total rows: 52 of 52 Query complete 00:00:00.063

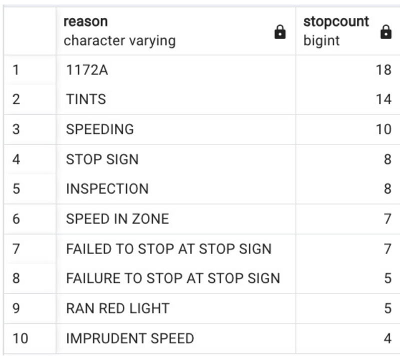
Fig. 18. Execution results following the execution of Query 12

VIII. Query Execution Analysis

When modifying or querying from a database, some actions taken may be costly, depending on the operations performed or the size of the relations being acted on. Thus, it is of interest to find a more optimal way of querying so that the operations taken are less costly. This section serves to display examples relevant to the Traffic Stop Receipt database in which problematic queries are identified, and then modified to improve performance. The 'EXPLAIN' tool in PostgreSQL is used to analyze the costs of each query.

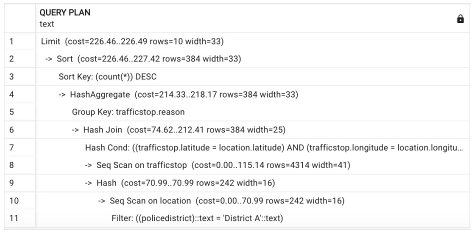
Problematic Query 1: Find the top 10 reasons for traffic stop receipts being issued in police district A)

```
SELECT Reason, COUNT(*) AS StopCount
FROM TrafficStop
JOIN Location ON TrafficStop.Latitude
    = Location.Latitude
    AND TrafficStop.Longitude =
    Location.Longitude
WHERE Location.PoliceDistrict =
    'District A'
GROUP BY Reason
ORDER BY StopCount DESC
LIMIT 10;
```



	reason character varying	stopcount bigint
1	1172A	18
2	TINTS	14
3	SPEEDING	10
4	STOP SIGN	8
5	INSPECTION	8
6	SPEED IN ZONE	7
7	FAILED TO STOP AT STOP SIGN	7
8	FAILURE TO STOP AT STOP SIGN	5
9	RAN RED LIGHT	5
10	IMPRUDENT SPEED	4

Fig. 19. Execution results of Problematic Query 1

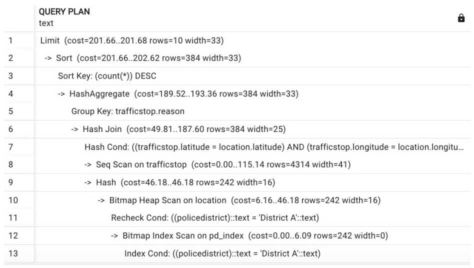


	QUERY PLAN text
1	Limit (cost=226.46..226.49 rows=10 width=33)
2	-> Sort (cost=226.46..227.42 rows=384 width=33)
3	Sort Key: (count(*)) DESC
4	-> HashAggregate (cost=214.33..218.17 rows=384 width=33)
5	Group Key: trafficstop.reason
6	-> Hash Join (cost=74.62..212.41 rows=384 width=25)
7	Hash Cond: ((trafficstop.latitude = location.latitude) AND (trafficstop.longitude = location.longitu...
8	-> Seq Scan on trafficstop (cost=0.00..115.14 rows=4314 width=41)
9	-> Hash (cost=70.99..70.99 rows=242 width=16)
10	-> Seq Scan on location (cost=0.00..70.99 rows=242 width=16)
11	Filter: ((policedistrict)::text = 'District A'::text)

Fig. 20. Cost breakdown of Problematic Query 1 using 'EXPLAIN'

In an effort to improve performance and lower the cost, an indexing technique is implemented on the PoliceDistrict column in the Location relation as follows:

```
CREATE INDEX PD_INDEX ON Location(PoliceDistrict);
```



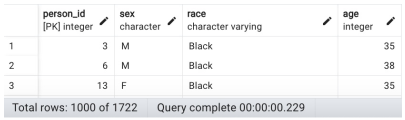
	QUERY PLAN text
1	Limit (cost=201.66..201.68 rows=10 width=33)
2	-> Sort (cost=201.66..202.62 rows=384 width=33)
3	Sort Key: (count(*)) DESC
4	-> HashAggregate (cost=189.52..193.36 rows=384 width=33)
5	Group Key: trafficstop.reason
6	-> Hash Join (cost=49.81..187.60 rows=384 width=25)
7	Hash Cond: ((trafficstop.latitude = location.latitude) AND (trafficstop.longitude = location.longitu...
8	-> Seq Scan on trafficstop (cost=0.00..115.14 rows=4314 width=41)
9	-> Hash (cost=46.18..46.18 rows=242 width=16)
10	-> Bitmap Heap Scan on location (cost=6.16..46.18 rows=242 width=16)
11	Recheck Cond: ((policedistrict)::text = 'District A'::text)
12	-> Bitmap Index Scan on pd_index (cost=0.00..6.09 rows=242 width=0)
13	Index Cond: ((policedistrict)::text = 'District A'::text)

Fig. 21. Cost breakdown following execution of the query after indexing PoliceDistrict and using 'EXPLAIN'

Upon rerunning the query after indexing, the cost is much lower across all steps of the query plan.

Problematic Query 2: Find those individuals who are older than the average age in the Person table)

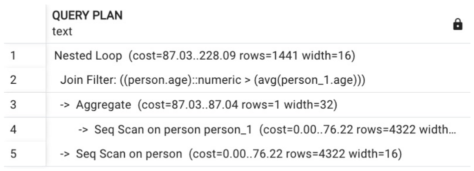
```
SELECT Person.Person_ID, Person.Sex,
    Person.Race, Person.Age
FROM Person
JOIN (SELECT AVG(Age) as AverageAge FROM Person)
    AS AvgSub
ON Person.Age > AvgSub.AverageAge;
```



	person_id [PK] integer	sex character	race character varying	age integer
1	3	M	Black	35
2	6	M	Black	38
3	13	F	Black	35

Total rows: 1000 of 1722 Query complete 00:00:00.229

Fig. 22. Execution results of Problematic Query 2

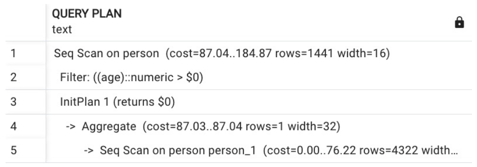


	QUERY PLAN text
1	Nested Loop (cost=87.03..228.09 rows=1441 width=16)
2	Join Filter: ((person.age)::numeric > (avg(person_1.age)))
3	-> Aggregate (cost=87.03..87.04 rows=1 width=32)
4	-> Seq Scan on person person_1 (cost=0.00..76.22 rows=4322 width...
5	-> Seq Scan on person (cost=0.00..76.22 rows=4322 width=16)

Fig. 23. Cost breakdown of Problematic Query 2 using 'EXPLAIN'

In an effort to improve performance and lower the cost, a new and more efficient way to query the same information is carried out without using 'JOIN' or a subquery and is written as follows:

```
SELECT Person_ID, Sex, Race, Age
FROM Person
WHERE Age > (SELECT AVG(Age) FROM Person);
```



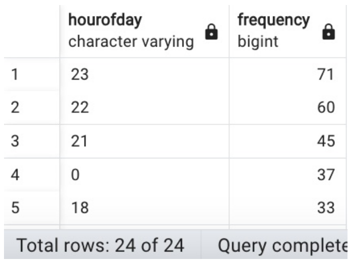
	QUERY PLAN text
1	Seq Scan on person (cost=87.04..184.87 rows=1441 width=16)
2	Filter: ((age)::numeric > $0)
3	InitPlan 1 (returns $0)
4	-> Aggregate (cost=87.03..87.04 rows=1 width=32)
5	-> Seq Scan on person person_1 (cost=0.00..76.22 rows=4322 width...

Fig. 24. Cost breakdown following execution of the query without using 'JOIN' or a subquery and using 'EXPLAIN'

Upon rerunning the query after indexing, the cost is lower when it comes to the sequential scan of the Person relation.

Problematic Query 3: Find the count of traffic stop receipts issued on Fridays and Saturdays for each hour of the day and order by frequency in descending order)

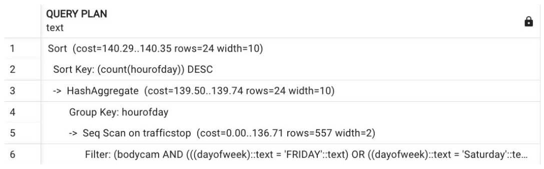
```
SELECT HourOfDay,
    COUNT(HourOfDay) AS Frequency
FROM TrafficStop
WHERE BodyCam = true
AND (DayOfWeek = 'FRIDAY'
    OR DayOfWeek = 'Saturday')
GROUP BY HourOfDay
ORDER BY Frequency DESC;
```



	hourofday character varying	frequency bigint
1	23	71
2	22	60
3	21	45
4	0	37
5	18	33

Total rows: 24 of 24 Query complete

Fig. 25. Execution results of Problematic Query 3

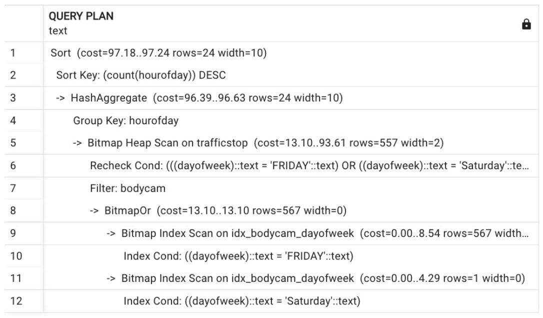


	QUERY PLAN text
1	Sort (cost=140.29..140.35 rows=24 width=10)
2	Sort Key: (count(hourofday)) DESC
3	-> HashAggregate (cost=139.50..139.74 rows=24 width=10)
4	Group Key: hourofday
5	-> Seq Scan on trafficstop (cost=0.00..136.71 rows=557 width=2)
6	Filter: (bodycam AND (((dayofweek)::text = 'FRIDAY'::text) OR ((dayofweek)::text = 'Saturday'::te...

Fig. 26. Cost breakdown of Problematic Query 3 using 'EXPLAIN'

In an effort to improve performance and lower the cost, an indexing technique is implemented on the DayOfWeek column in the TrafficStop relation as follows:

```
CREATE INDEX BC_INDEX ON TrafficStop(DayOfWeek);
```



	QUERY PLAN text
1	Sort (cost=97.18..97.24 rows=24 width=10)
2	Sort Key: (count(hourofday)) DESC
3	-> HashAggregate (cost=96.39..96.63 rows=24 width=10)
4	Group Key: hourofday
5	-> Bitmap Heap Scan on trafficstop (cost=13.10..93.61 rows=557 width=2)
6	Recheck Cond: (((dayofweek)::text = 'FRIDAY'::text) OR ((dayofweek)::text = 'Saturday'::te...
7	Filter: bodycam
8	-> BitmapOr (cost=13.10..13.10 rows=567 width=0)
9	-> Bitmap Index Scan on idx_bodycam_dayofweek (cost=0.00..8.54 rows=567 width...
10	Index Cond: ((dayofweek)::text = 'FRIDAY'::text)
11	-> Bitmap Index Scan on idx_bodycam_dayofweek (cost=0.00..4.29 rows=1 width=0)
12	Index Cond: ((dayofweek)::text = 'Saturday'::text)

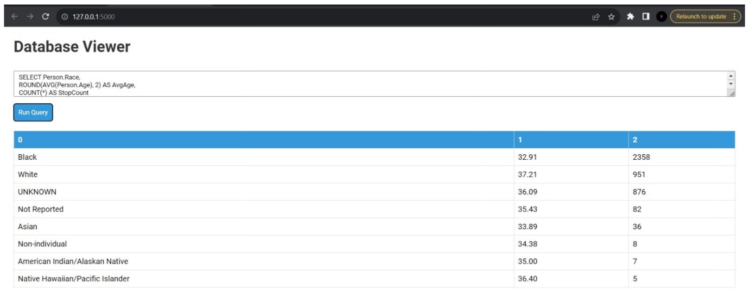
Fig. 27. Cost breakdown following execution of the query after indexing DayOfWeek and using 'EXPLAIN'

Upon rerunning the query after indexing, the total cost of the query is lower.

IX. Web Application

A web application was built using Python and Flask, and then connected to the PostgreSQL database. The application allows users to input SQL queries through a web interface, and the Flask backend processes those queries, interacts with the PostgreSQL database, and returns the desired results. Below the results of query 1 from Section VII are displayed in the web application, in which the average age and the number of traffic stops are grouped by race. The SQL query is also repeated below for reference.

```
SELECT Person.Race,
    ROUND(AVG(Person.Age), 2) AS AvgAge,
    COUNT(*) AS StopCount
FROM Person
JOIN TrafficStop ON
    Person.Person_ID = TrafficStop.Person_ID
GROUP BY Person.Race
ORDER BY StopCount DESC;
```



Database Viewer

```
SELECT Person.Race,
ROUND(AVG(Person.Age), 2) AS AvgAge,
COUNT(*) AS StopCount
```

0	1	2
Black	32.91	2358
White	37.21	951
UNKNOWN	36.09	876
Not Reported	35.43	82
Asian	33.89	36
Non-individual	34.38	8
American Indian/Alaskan Native	35.00	7
Native Hawaiian/Pacific Islander	36.40	5

Fig. 28. Execution results for Query 1, queried via a web interface

X. Conclusion

To recap the analysis of traffic stop receipts issued in the City of Buffalo between October 1, 2022 and October 17, 2023, the initial schema was taken from Open Data Buffalo, which features numerous data sets for the city. The data set of over 14,000 rows and 32 columns was narrowed down to just under 5,000 instances, and the appropriate relations ”Person”, ”Location”, and ”TrafficStop” were created in an organized manner. From there, tables were created in PostgreSQL, a database management system, and the data itself was imported into those relations. The database was tested with twelve sample queries, utilizing ’SELECT’, ’INSERT’, ’UPDATE’, and ’DELETE’ keywords, and such analysis attempted to uncover information and trends that can be used to address some overlooked features that contribute to poverty in the City of Buffalo. Finally, a sample of three problematic queries were devised and analyzed as it relates to their overall costs to query. In an effort to improve performance and drive down cost for those queries, techniques such as indexing and eliminating the use of ’JOIN’ and subqueries were used.

When it comes to answering the questions posed in the problem statement at the start of this report, the analyses conducting allowed for a greater understanding of the domain and the at least partial answering of some of the questions. In terms of the frequency of traffic stops receipts issued by neighborhood, Genesee-Moselle, Broadway Fillmore, Kensington-Bailey, North Park, and Upper West Side were among the top five neighborhoods with the most issued, all having more than 200 receipts issued in the one year span of analyses. Knowing this may allow city officials to pinpoint such areas and have a greater focus on making sure that those neighborhoods have the proper resources and infrastructure to ensure the safety of all citizens. It is typical for more traffic stop receipts to be issued during rush hours and on Friday and Saturday evenings, particularly between the hours of 10PM and 1AM. This allows for government officials to understand the times of day in which more monitoring may be needed in order to curb violations in the future and keep Buffalo safer. In terms of the breakdown of traffic stop receipts issued by race, the plurality were issued to Black people during the time period analyzed, which is more than double the amount of receipts issued to White people, who came in second on the list. This shows a sort of disparity in how traffic stop receipts are issued, and should urgently be examined further. Finally, it was discovered that running stop signs, having illegal tints, and speeding are the top reasons for traffic stop receipts being issued. In all, there is no doubt that the analysis of traffic stop receipts issued in the City of Buffalo uncovered vital information that city officials may be interested in considering in curbing the poverty problems in the area.

References

[1] Open Data Buffalo, “Traffic Stop Receipts,” [Online]. Available: (https://data.buffalony.gov/Public-Safety/Traffic-Stop-Receipts/8mqs-6g9h).

[2] PostgreSQL, [Online]. Available: (https://www.postgresql.org).

[3] Lucidchart, [Online]. Available: (https://www.lucidchart.com).

[4] Overleaf, [Online]. Available: (https://www.overleaf.com).