

# CSE 4/560 Data Models and Query Language Semester Project

Deepthi D'Souza (deepthid), Sankalp Mehani (sankalpm), Pritish Kamble (pritishk)

May 7, 2023

**Project Title:** CrimeTrack: A Database Management System for Law Enforcement Agencies

## 1 Problem Statement

Law enforcement agencies face challenges in effectively managing criminal cases due to the lack of a centralized and integrated crime database system. Generally, criminal data is scattered across multiple systems and in various formats, making it difficult for investigators to access and analyze the information they need to solve cases.

The absence of a comprehensive crime database system also leads to inefficiencies in data entry, storage, and retrieval, resulting in delays and errors in investigations. This can lead to the loss of valuable evidence and the inability to identify suspects, ultimately hindering the administration of justice.

To address these challenges, a web-based crime database system needs to be developed, which can provide a centralized repository to store, manage, and analyze crime data. The system should be accessible to authorized personnel and should implement robust security measures to ensure the confidentiality and integrity of the data.

The development of a comprehensive crime database system will enable law enforcement agencies to streamline their operations, improve their investigative capabilities, and ultimately, enhance public safety.

## 2 Target User

The target users for this project would primarily be law enforcement agencies, such as police departments, detective bureaus, and other agencies responsible for investigating criminal cases. The system would be used by investigators, detectives, analysts, and other personnel involved in managing and analyzing crime data.

Additionally, the system could also be used by other agencies, such as prosecutors' offices and courts, to access and analyze crime data for legal proceedings.

### 3 Database Design

The Buffalo Crime Dataset was downloaded from [here](#). The dataset is then divided into the following tables.

**Table Name: Crime\_data**

Field Name	Data Type	Constraints
Case Number	character varying (20)	NOT NULL, PKEY
Incident Datetime	timestamp without time zone	
Incident ID	integer	NOT NULL, UNIQUE
Incident Type	character varying (40)	
Parent Incident Type	character varying (40)	
Address	character varying (40)	
State	character varying (40)	
Created At	timestamp without time zone	NOT NULL
Updated At	timestamp without time zone	
Neighborhood	character (40)	
Police District	character (20)	
Council District	character (20)	
Investigation Status	character varying (50)	

#### 3.1 Normalization

The 'Crime\_data' table is not normalized. To achieve normalization, it is decomposed into 7 subsequent tables. The said tables have been created in a way to ensure that all conform to the Boyce-Codd Normalization norms. The functional dependencies below each table are evident of the same.

**Table Name: Council\_district**

Field Name	Data Type	Constraints
Council District ID	integer	NOT NULL, PKEY
Council District	character varying (20)	

Functional Dependencies:

- Council District ID  $\rightarrow$  Council District

**Table Name: Neighborhood**

Field Name	Data Type	Constraints
Neighborhood ID	integer	NOT NULL, PKEY
Neighborhood	character varying (40)	
Council District ID	integer	FKEY

Functional Dependencies:

- Neighborhood ID  $\rightarrow$  Neighborhood
- Neighborhood ID  $\rightarrow$  Council District ID
- Neighborhood  $\rightarrow$  Council District ID

**Table Name: police\_district**

Field Name	Data Type	Constraints
Police District ID	integer	NOT NULL, PKEY
Police District	character varying (20)	

Functional Dependencies:

- Police District ID  $\rightarrow$  Police District

**Table Name: crime\_incidents**

Field Name	Data Type	Constraints
Case Number	character varying(20)	NOT NULL, PKEY
Incident ID	integer	NOT NULL, PKEY
Investigation Status	character varying(50)	
Police District ID	integer	FKEY
Incident Type ID	integer	FKEY

Functional Dependencies:

- Case Number  $\rightarrow$  Investigation Status, police District ID, incident Type
- Incident ID  $\rightarrow$  Investigation Status, police District ID, incident Type
- Case Number  $\rightarrow$  Incident ID
- Incident ID  $\rightarrow$  Case Number

**Table Name: Incident\_datetime**

Field Name	Data Type	Constraints
Incident ID	integer	NOT NULL, PKEY
Incident Datetime	timestamp without timezone	
Created At	timestamp without timezone	
Updated At	timestamp without timezone	

Functional Dependencies:

- Incident ID  $\rightarrow$  Incident Datetime, Created At, Updated At

**Table Name: Location**

Field Name	Data Type	Constraints
Incident ID	integer	NOT NULL, PKEY
Address	character varying (40)	
Neighborhood ID	integer	FKEY

Functional Dependencies:

- Incident ID  $\rightarrow$  Address, Neighborhood ID

Table Name: Incident Type

Field Name	Data Type	Constraints
Incident Type ID	integer	NOT NULL, PKEY
Incident Type	character varying(40)	
Parent Incident Type	character varying(40)	

### Functional Dependencies:

- Incident Type ID  $\rightarrow$  Incident Type, Parent Incident Type
- Incident Type  $\rightarrow$  Parent Incident Type

### 3.2 ER Diagram

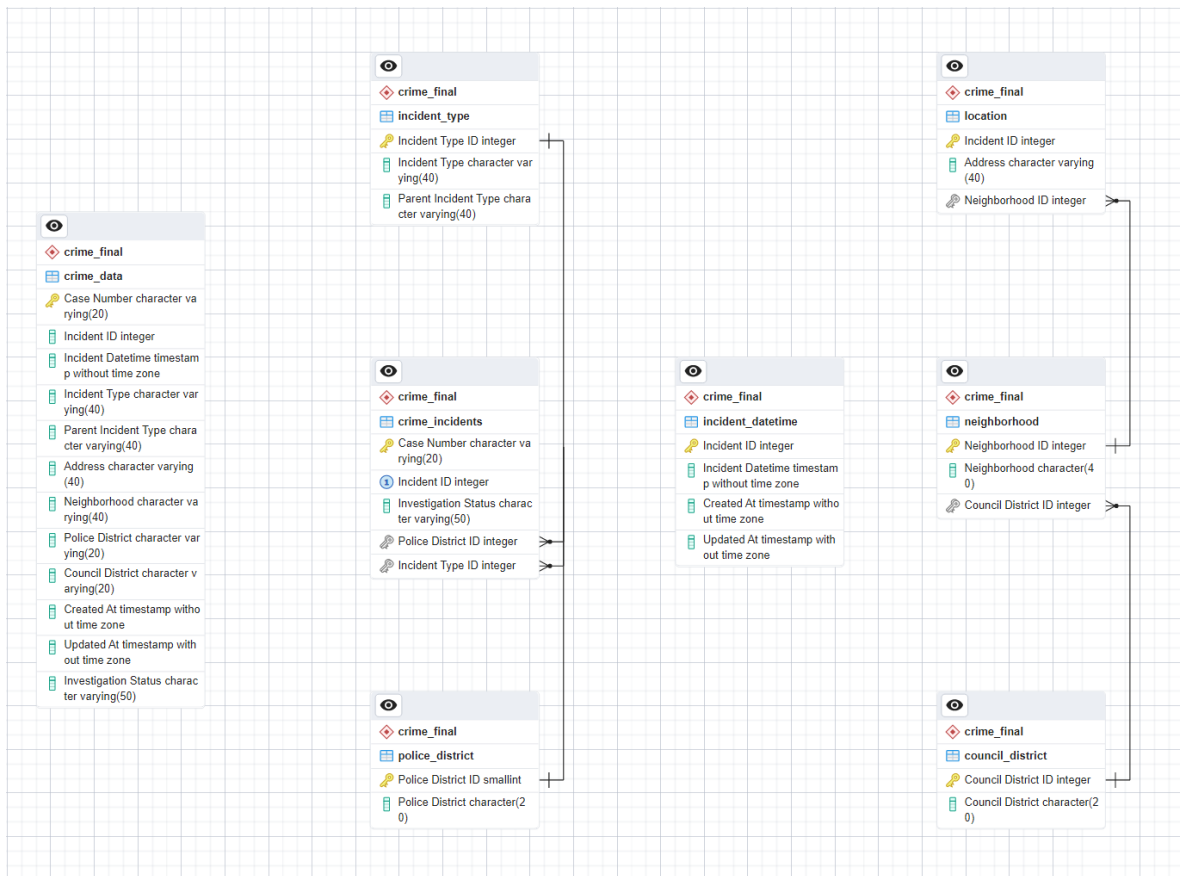


Figure 1: ER Diagram for the database

### 3.3 Indexing

An index is added for the crime\_incidents table on the "Incident ID" column. This clustered index significantly reduces the fetch/access time on this table.

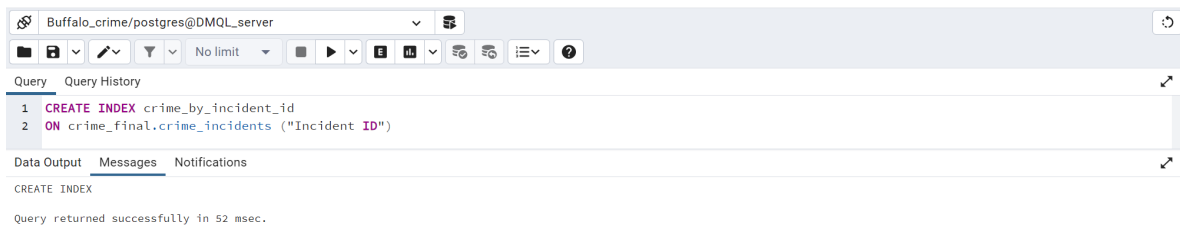


Figure 2: Index creation on crime\_incidents table

### 3.4 Triggers

Three triggers: Insert, Update, and Delete are added to the database to enforce the design rules and maintain integrity of every table. These triggers are fired whenever new data is inserted, or existing data is updated or deleted.

## 4 Queries

### 4.1 SELECT Queries

The screenshot shows a PostgreSQL query editor interface. The query window contains the following SQL code:

```
1 SELECT c1."Police District ID", COUNT(*) AS num_crimes
2 FROM crime_final.crime_incidents c1
3 GROUP BY c1."Police District ID"
4 ORDER BY num_crimes DESC;
```

The Data Output tab shows the following table:

	Police District ID integer	num_crimes bigint
1	4	2652
2	3	2631
3	5	2384
4	2	2316
5	1	1451

Figure 3: Get number of crimes for every police district

The screenshot shows a PostgreSQL query editor interface. The query window contains the following SQL code:

```
1 SELECT it."Parent Incident Type", COUNT(*) AS num_occurrences
2 FROM crime_final.incident_type it
3 JOIN crime_final.crime_incidents c1
4 ON it."Incident Type ID" = c1."Incident Type ID"
5 GROUP BY it."Parent Incident Type"
6 ORDER BY num_occurrences DESC;
```

The Data Output tab shows the following table:

	Parent Incident Type character varying (40)	num_occurrences bigint
1	Theft	5120
2	Assault	2330
3	Theft of Vehicle	1712
4	Breaking & Entering	1379
5	Robbery	675
6	Sexual Offense	150
7	Homicide	68

Figure 4: Get number of occurrences for every incident type

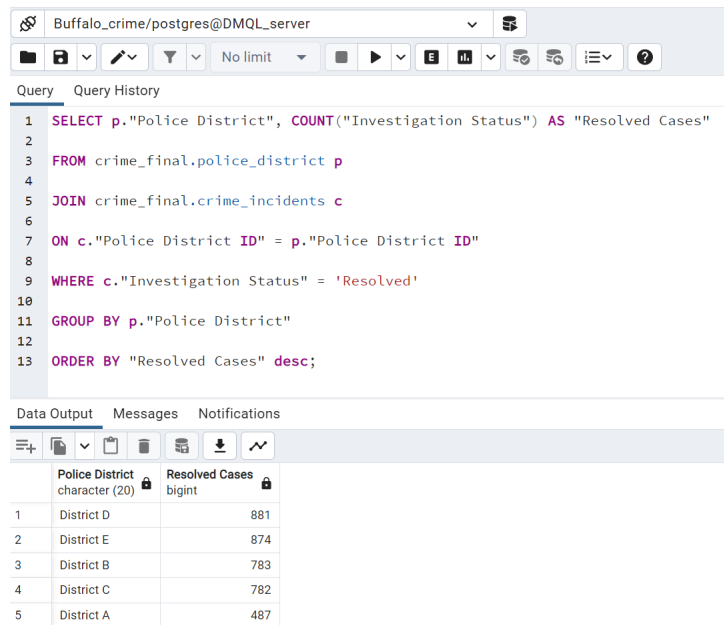


Figure 5: Get number of resolved cases for every police district

## 4.2 INSERT queries

A new crime case is inserted into the crime\_data table. The 'insert' trigger automatically fires and inserts corresponding data into the relevant tables.

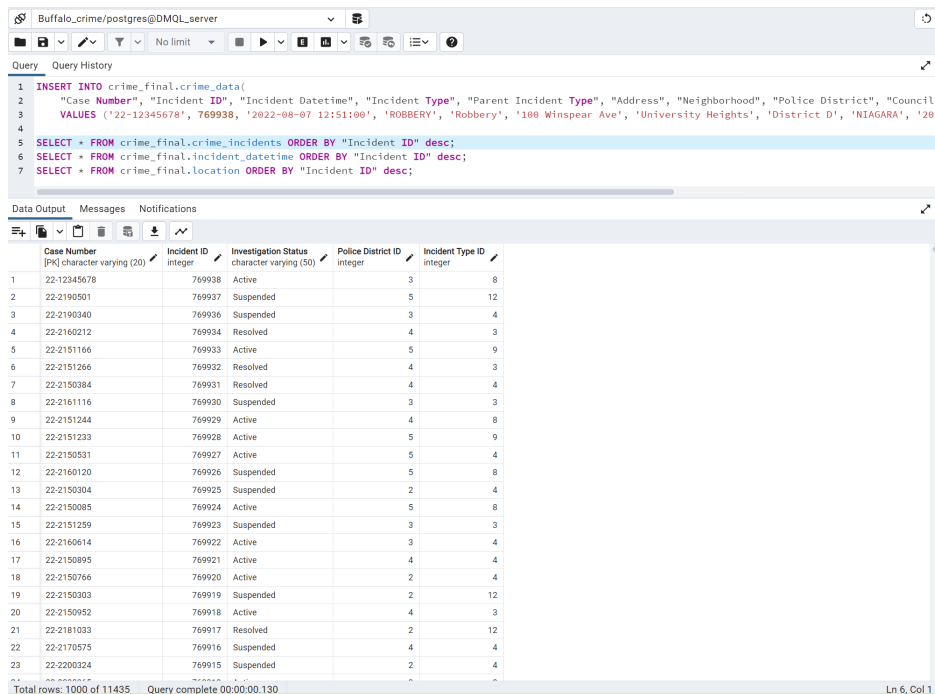


Figure 6: Verifying new data in the crime\_incidents table

Buffalo\_crime/postgres@DMQL\_server

Query Query History

```

1 INSERT INTO crime_final.crime_data(
2   "Case Number", "Incident ID", "Incident Datetime", "Incident Type", "Parent Incident Type", "Address", "Neighborhood", "Police District", "Council
3   VALUES ('22-12345678', 769938, '2022-08-07 12:51:00', 'ROBBERY', 'Robbery', '100 Winspear Ave', 'University Heights', 'District D', 'NIAGARA', '20
4
5 SELECT * FROM crime_final.crime_incidents ORDER BY "Incident ID" desc;
6 SELECT * FROM crime_final.incident_datetime ORDER BY "Incident ID" desc;
7 SELECT * FROM crime_final.location ORDER BY "Incident ID" desc;

```

Data Output Messages Notifications

	Incident ID [PK] integer	Incident Datetime timestamp without time zone	Created At timestamp without time zone	Updated At timestamp without time zone
1	769938	2022-08-07 12:51:00	2022-12-05 09:34:00	2022-12-05 09:34:00
2	769937	2022-08-07 12:51:00	2022-08-07 12:52:00	2023-02-20 20:08:00
3	769936	2022-08-07 09:30:00	2022-08-07 09:31:00	2023-02-17 05:21:00
4	769934	2022-08-04 00:00:00	2022-08-04 07:15:00	2023-02-11 19:32:00
5	769933	2022-08-03 21:39:00	2022-08-03 21:40:00	2022-08-03 21:40:00
6	769932	2022-08-03 23:20:00	2022-08-03 23:21:00	2023-01-25 05:27:00
7	769931	2022-08-03 10:48:00	2022-08-03 10:48:00	2023-02-16 18:43:00
8	769930	2022-08-04 21:53:00	2022-08-04 21:54:00	2023-01-24 08:50:00
9	769929	2022-08-03 23:01:00	2022-08-03 23:02:00	2022-08-03 23:02:00
10	769928	2022-08-03 22:53:00	2022-08-03 22:53:00	2022-08-03 22:53:00
11	769927	2022-08-03 08:30:00	2022-08-03 13:07:00	2022-08-03 13:07:00
12	769926	2022-08-03 02:30:00	2022-08-04 03:36:00	2023-02-04 22:30:00
13	769925	2022-08-03 09:33:00	2022-08-03 09:34:00	2023-01-15 21:31:00
14	769924	2022-08-03 02:38:00	2022-08-03 02:39:00	2022-08-03 02:39:00
15	769923	2022-08-03 23:15:00	2022-08-03 23:16:00	2022-10-17 11:29:00
16	769922	2022-08-03 20:30:00	2022-08-04 14:18:00	2022-08-04 14:18:00
17	769921	2022-08-03 17:57:00	2022-08-03 17:58:00	2022-08-03 17:58:00
18	769920	2022-08-03 16:20:00	2022-08-03 16:20:00	2022-08-03 16:20:00
19	769919	2022-08-03 09:31:00	2022-08-03 09:32:00	2023-02-01 20:00:00
20	769918	2022-08-03 18:40:00	2022-08-03 18:40:00	2022-08-03 18:40:00
21	769917	2022-08-06 20:38:00	2022-08-06 20:38:00	2022-09-16 08:18:00
22	769916	2022-08-05 14:31:00	2022-08-05 14:32:00	2022-09-09 09:57:00
23	769915	2022-08-08 10:02:00	2022-08-08 10:03:00	2022-11-27 20:06:00

Total rows: 1000 of 11435 Query complete 00:00:00.074 Ln 7, Col 1

Figure 7: Verifying new data in the incident\_datetime table

Buffalo\_crime/postgres@DMQL\_server

Query Query History

```

1 INSERT INTO crime_final.crime_data(
2   "Case Number", "Incident ID", "Incident Datetime", "Incident Type", "Parent Incident Type", "Address", "Neighborhood", "Police District", "Council
3   VALUES ('22-12345678', 769938, '2022-08-07 12:51:00', 'ROBBERY', 'Robbery', '100 Winspear Ave', 'University Heights', 'District D', 'NIAGARA', '20
4
5 SELECT * FROM crime_final.crime_incidents ORDER BY "Incident ID" desc;
6 SELECT * FROM crime_final.incident_datetime ORDER BY "Incident ID" desc;
7 SELECT * FROM crime_final.location ORDER BY "Incident ID" desc;

```

Data Output Messages Notifications

	Incident ID [PK] integer	Address character varying (40)	Neighborhood ID integer
1	769938	100 Winspear Ave	14
2	769937	0 Block PEACH ST	18
3	769936	400 Block W FERRY ST	17
4	769934	100 Block OAKMONT AV	26
5	769933	200 Block W FERRY ST	18
6	769932	300 Block DARTMOUTH AV	20
7	769931	200 Block LASALLE AV	6
8	769930	700 Block ELMWOOD AV	26
9	769929	300 Block E AMHERST ST	15
10	769928	VIRGINIA ST & MULBERRY ST	23
11	769927	200 Block MAIN ST	2
12	769926	500 Block MAIN ST	25
13	769925	1000 Block GENESEE ST	14
14	769924	600 Block PROSPECT AV	17
15	769923	1000 Block ELMWOOD AV	20
16	769922	0 Block HARVARD PL	14
17	769921	0 Block CUSTER ST	26
18	769920	2100 Block GENESEE ST	23
19	769919	900 Block E FERRY ST	28
20	769918	MOSELLE ST & E DELAVAN AV	19
21	769917	0 Block HAUF ST	28
22	769916	300 Block AMHERST ST	14
23	769915	400 Block ADAMS ST	3

Successfully run. Total query runtime: 76 msec. 11435 rows affected. X

Total rows: 1000 of 11435 Query complete 00:00:00.076 Ln 7, Col 64

Figure 8: Verifying new data in the location table

### 4.3 UPDATE Queries

A case is marked as 'Resolved'. This change is updated into the crime\_data table. The 'update' trigger fires up and updates corresponding data into the relevant tables.

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

```
1 UPDATE crime_final.crime_data
2 SET "Investigation Status" = 'Resolved', "Updated At"='2023-05-07 12:00:00'
3 WHERE "Case Number" = '22-12345678';
4
5
6 SELECT ci."Case Number", ci."Incident Type ID", it."Incident Type", it."Parent Incident Type", ci."Investigation Status", id."Updated At"
7 FROM crime_final.crime_incidents ci
8 JOIN crime_final.incident_type it
9 ON ci."Incident Type ID" = it."Incident Type ID"
10 JOIN crime_final.incident_datetime id
11 ON ci."Incident ID" = id."Incident ID"
12 WHERE "Case Number" = '22-12345678';
13
```

Below the query, the 'Data Output' tab shows the results of the SELECT query:

Case Number	Incident Type ID	Incident Type	Parent Incident Type	Investigation Status	Updated At
22-12345678	9	ASSAULT	Assault	Resolved	2023-05-07 17:58:18.170847

Figure 9: Verifying updated data

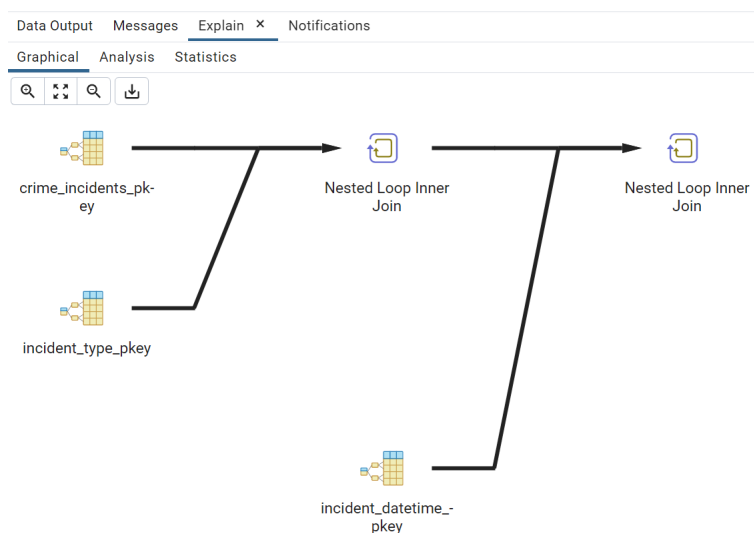


Figure 10: EXPLAIN tool on the above query

A case's incident type is changed to 'Burglary'. This change is updated into the crime\_data table. The 'update' trigger fires up and updates corresponding data into the relevant tables.



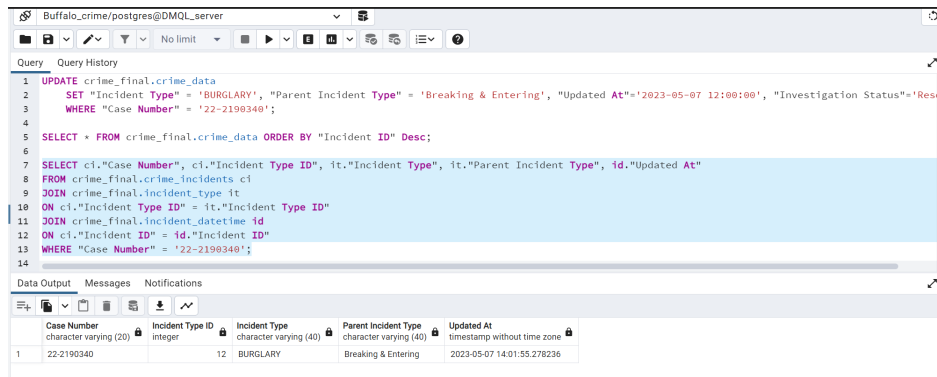


Figure 11: Verifying updated data

## 4.4 DELETE Queries

A case is deleted using its Incident ID. The 'delete' trigger fires up and deletes the linked entries in other tables.

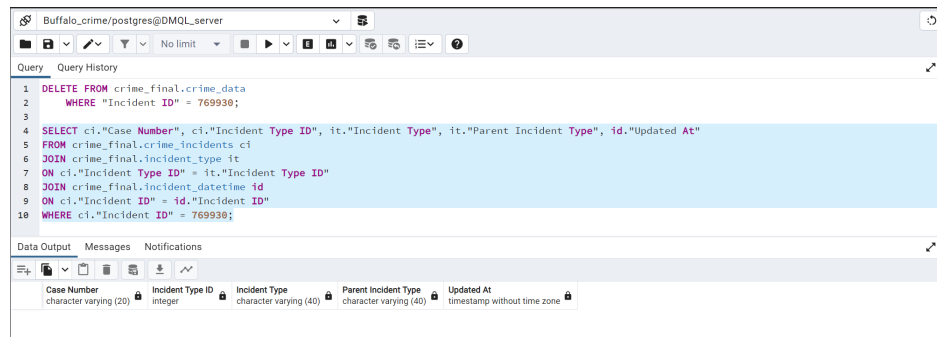


Figure 12: Deleting a case

## 4.5 Improving Performance

Planning and Execution times for a query with and without indexing are compared.

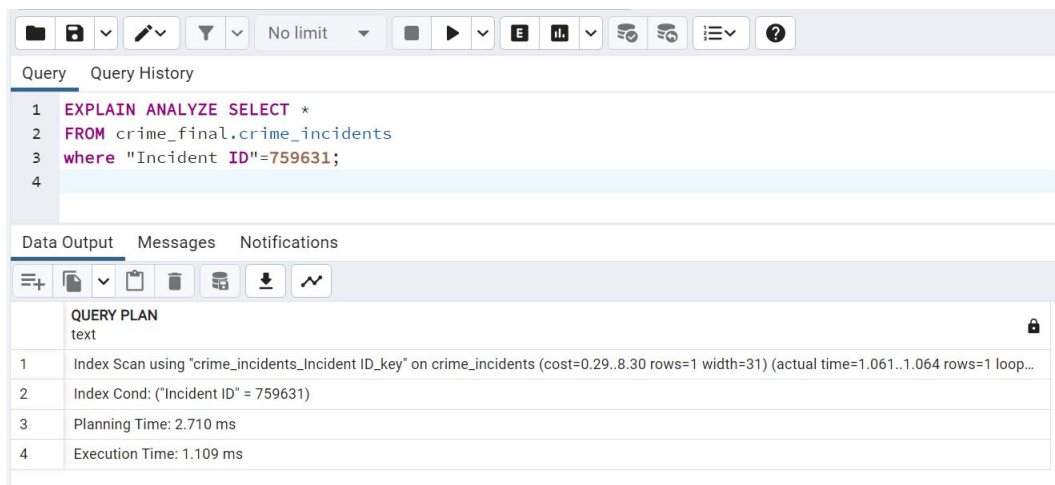


Figure 13: Fetching data without any index



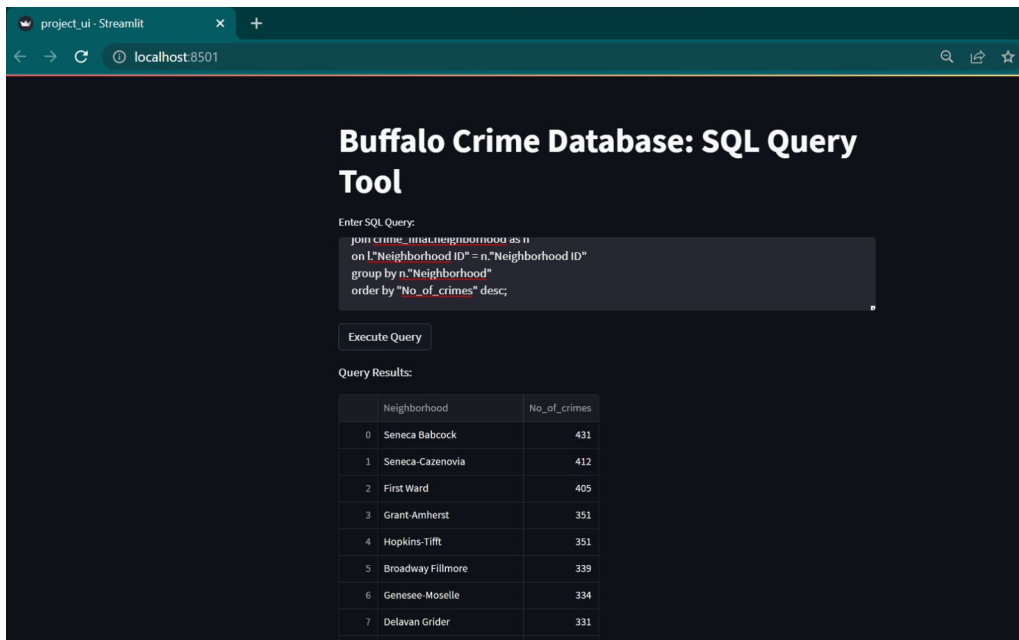


Figure 16: Running a query on the web app

## 6 Contribution

Each member made an equal contribution to the project. Every member played an important role in the project and worked collaboratively to achieve its goals. As a result, the project was a true team effort and all team members can take equal credit for its success.

## 7 References

- <https://www.javatpoint.com/dbms-normalization>
- <https://www.postgresqltutorial.com/postgresql-python/connect/>
- <https://www.sqlshack.com/designing-effective-sql-server-clustered-indexes/>
- <https://www.tutorialgateway.org/after-insert-triggers-in-sql-server/>