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 → 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 → Neighborhood
- Neighborhood ID → Council District ID
- Neighborhood → 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 → 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 → Investigation Status, police District ID, incident Type
- Incident ID → Investigation Status, police District ID, incident Type
- Case Number → Incident ID
- Incident ID → 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 → 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 → 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 → Incident Type, Parent Incident Type
- Incident Type → Parent Incident Type

3.2 ER Diagram

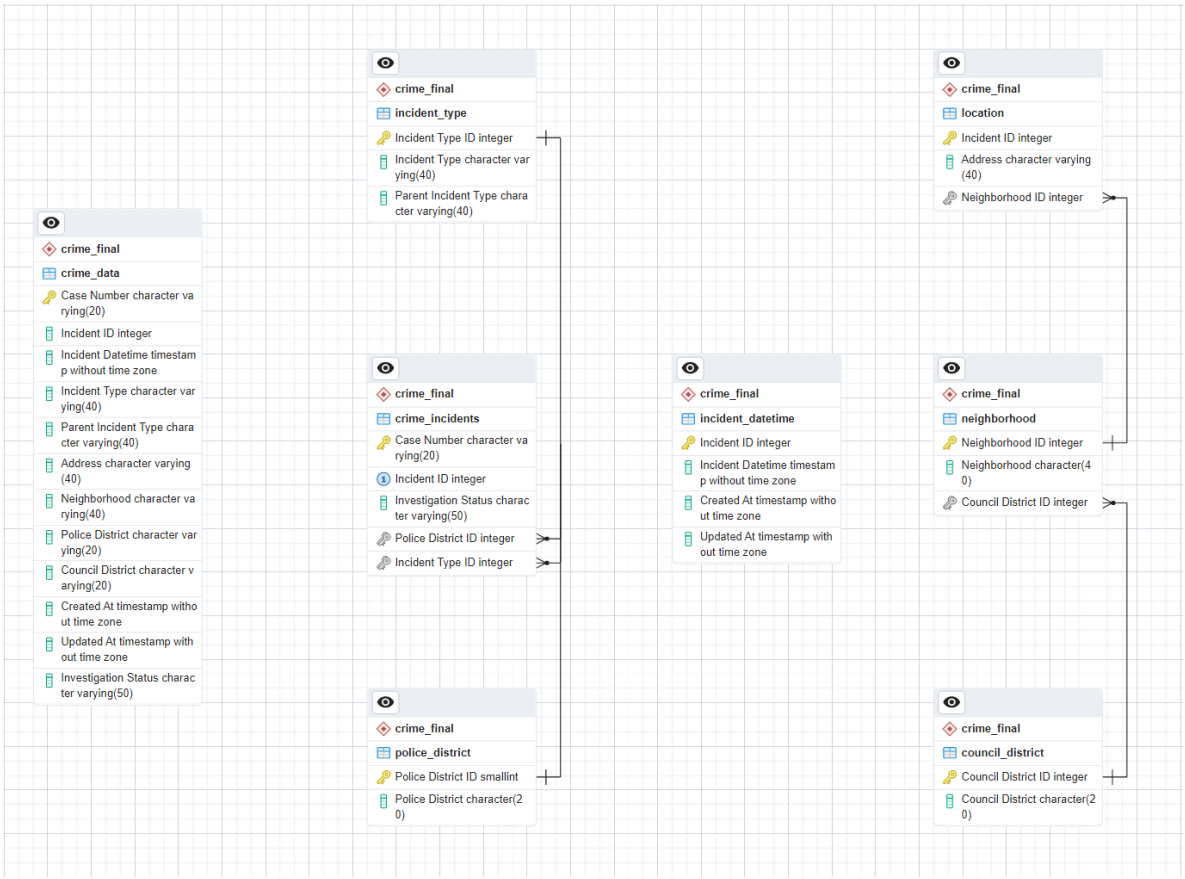

Figure 1: ER Diagram for the database

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.

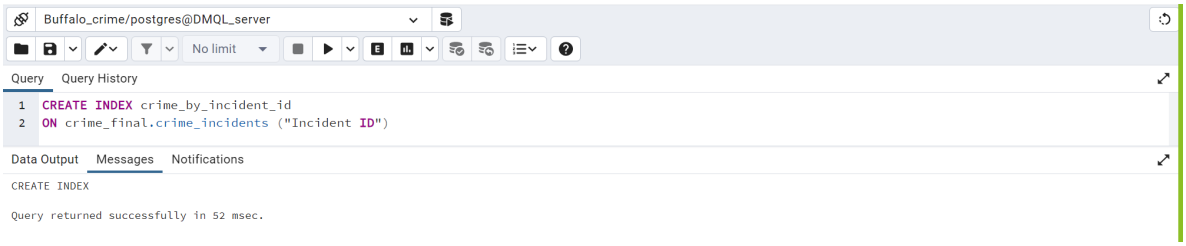

Figure 2: Index creation on crime_incidents table

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

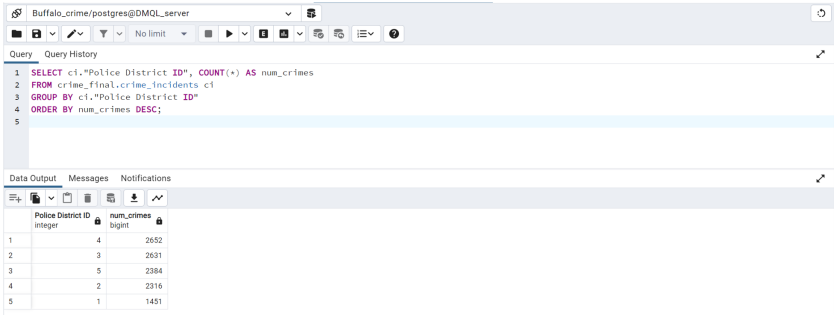

Figure 3: Get number of crimes for every police district

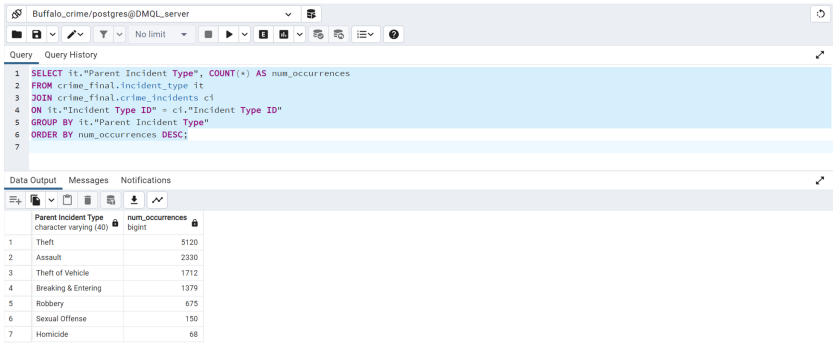

Figure 4: Get number of occurrences for every incident type

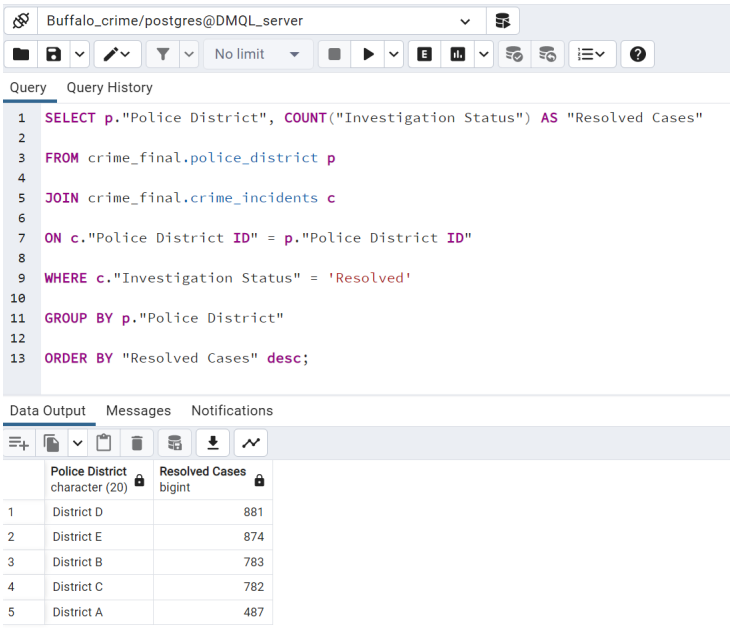

Figure 5: Get number of resolved cases for every police district

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.

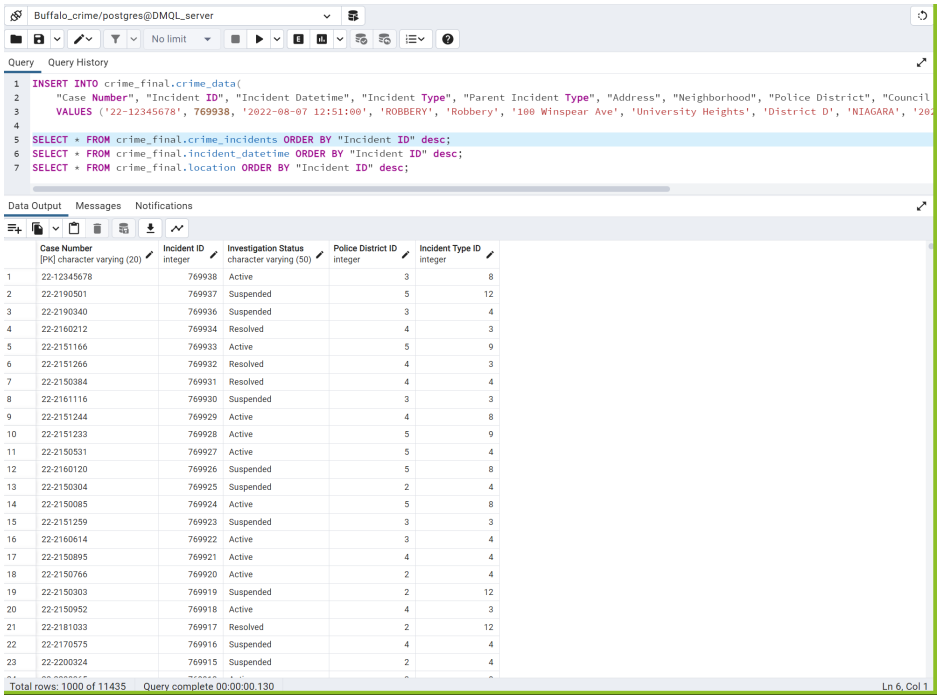

Figure 6: Verifying new data in the crime_incidents table

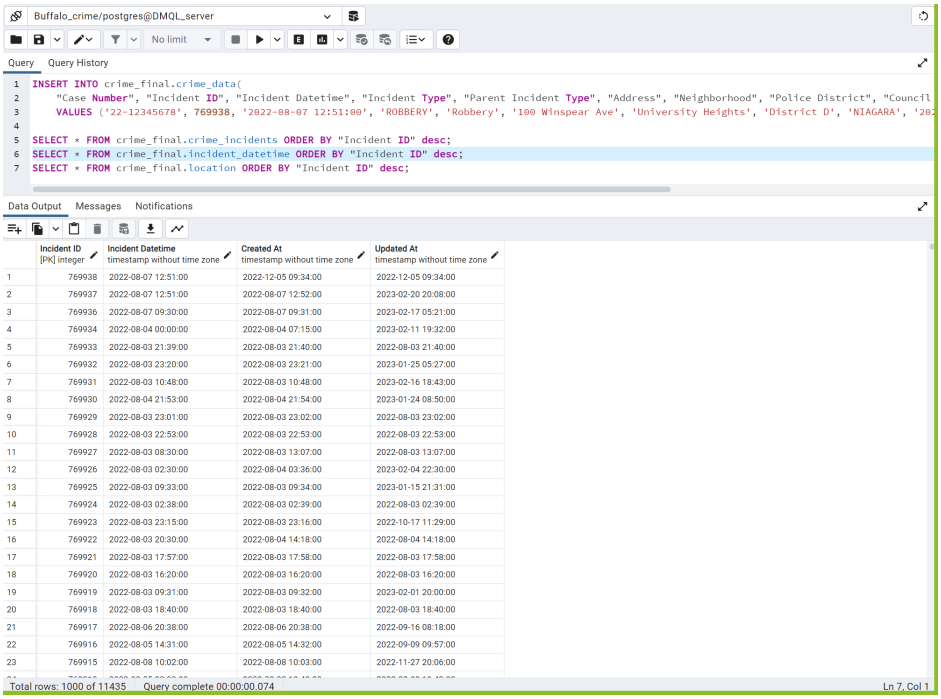

Figure 7: Verifying new data in the incident_datetime table

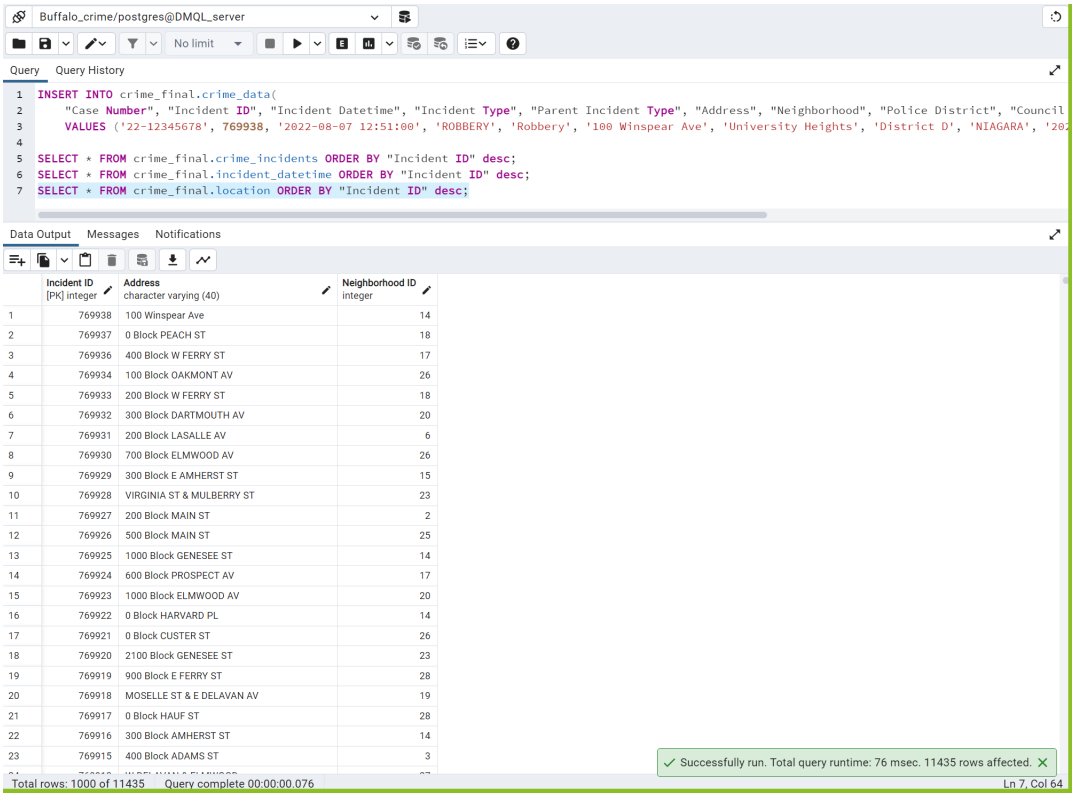

Figure 8: Verifying new data in the location table

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.

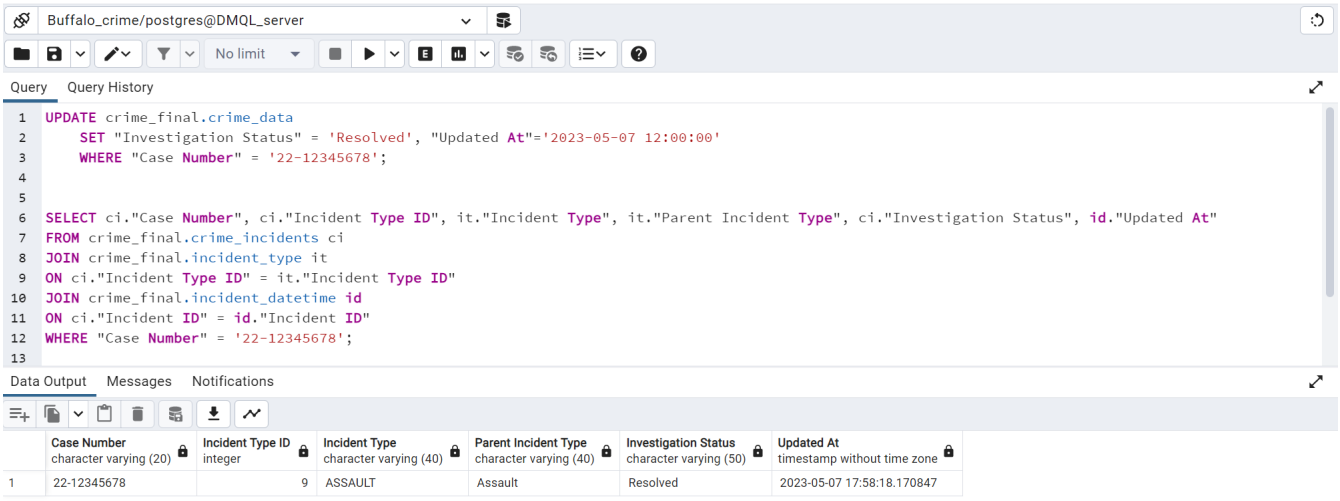

Figure 9: Verifying updated data

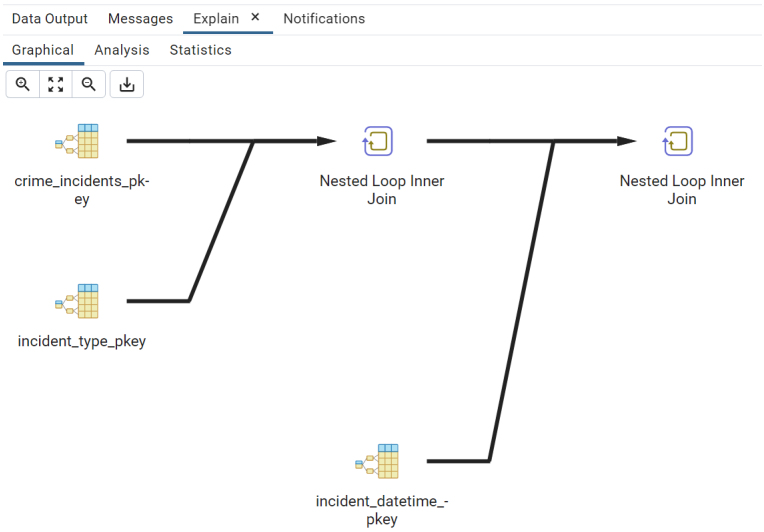

Figure 10: EXPLAIN tool on the above query

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.

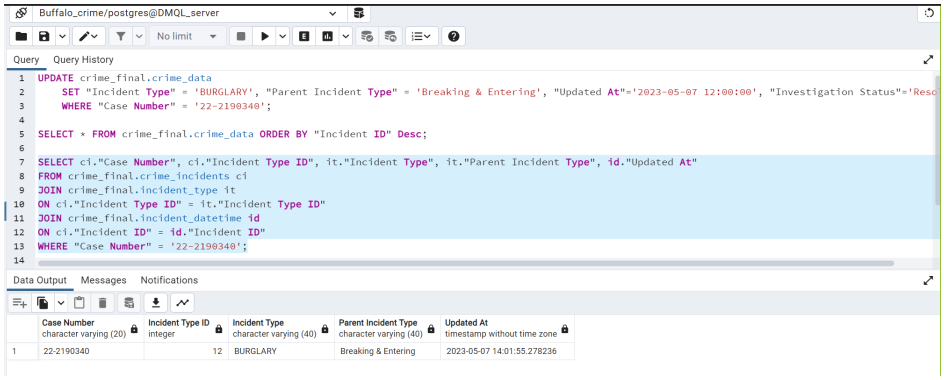

Figure 11: Verifying updated data

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.

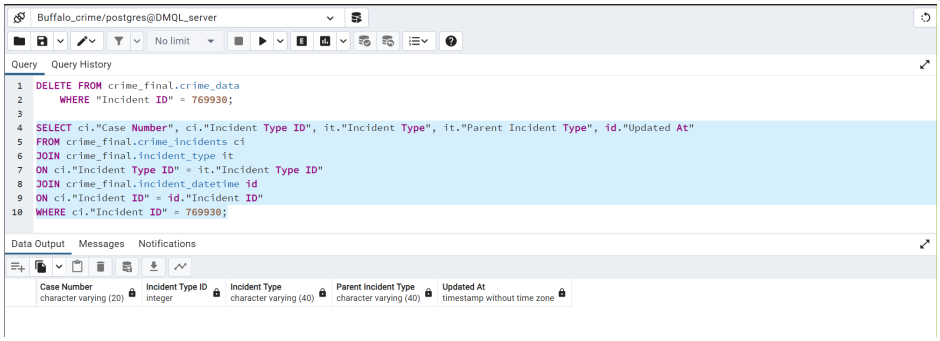

Figure 12: Deleting a case

4.5 Improving Performance

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

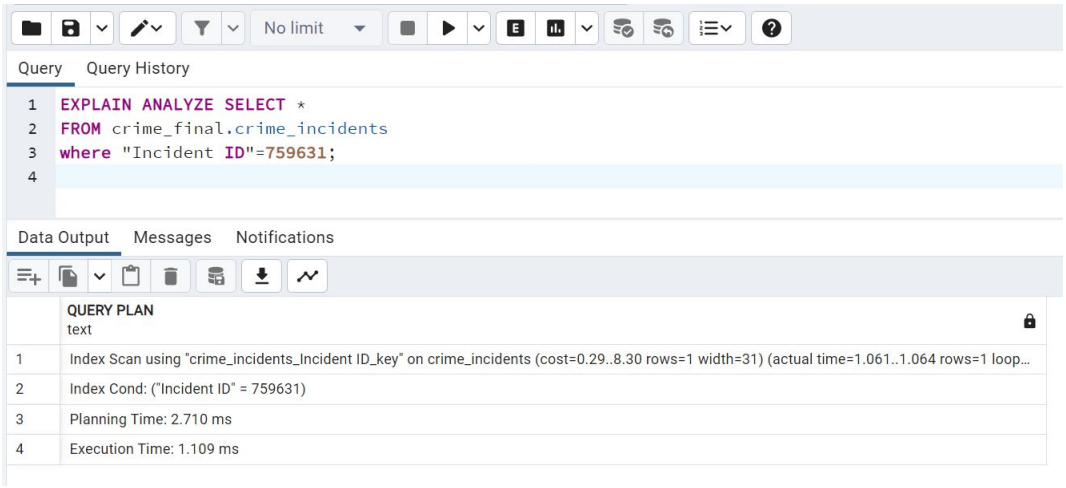

Figure 13: Fetching data without any index

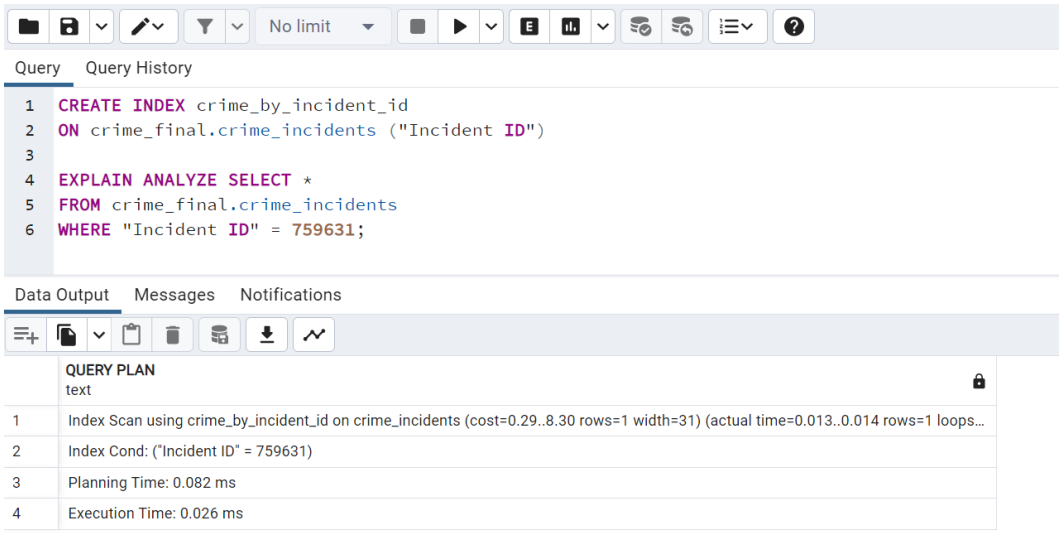

Figure 14: Fetching data without an index

Evidently, Indexing significantly boosts performance and reduces the Planning and Execution times for a SELECT query.

5 Bonus Task

Running the bonus_task_project_ui.py file starts a Streamlit server and opens a web browser with a user interface for the SQL query tool. This code uses the Streamlit library to create a simple web-based user-interface with a text area for an SQL query input and a button to execute the query. When the user clicks the button, the query is executed using the psycopg2 library to connect to a PostgreSQL database. The results of the query are displayed in a table on the same page.

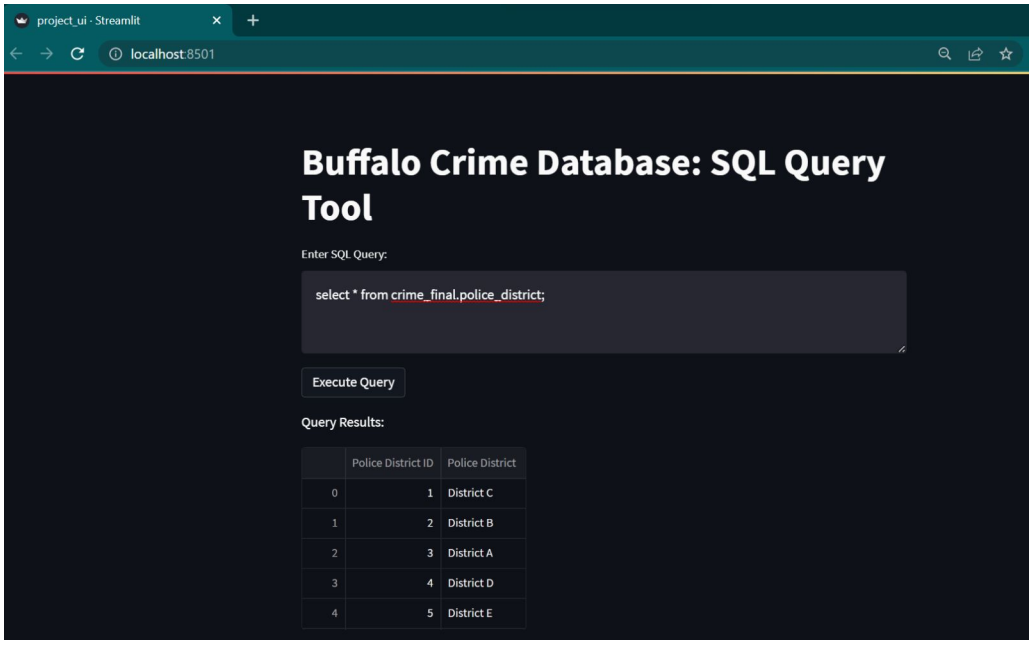

Figure 15: Running a query on the web app

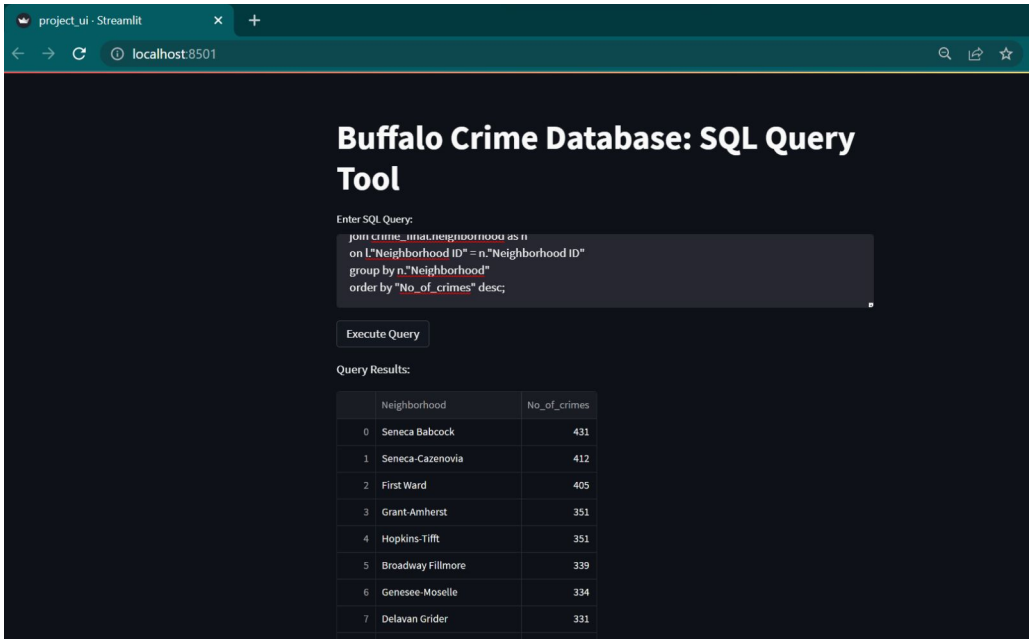

Figure 16: Running a query on the web app

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/