

Chicago Traffic Crash Analysis

ALY6110 | Data Management and Big Data

CRN: 80320

Group Beta:

Sanchi Gupta, gupta.sanch@northeastern.edu

Sankalp Biswal, biswal.s@northeastern.edu

Siddhant Viswanath, viswanath.si@northeastern.edu

Rhea John Thoppil, thoppil.r@northeastern.edu

Krutikkumar Patel, patel.krutikk@northeastern.edu

Introduction

This report documents the steps and processes involved in analyzing traffic crashes data for Chicago streets. The aim is to track traffic safety and congestion, providing insights into recent traffic crashes, red light and speed camera violations, and traffic patterns. The dataset was sourced from the City of Chicago's open data portal and ingested into Databricks for analysis.

Dataset Description

1. **Traffic Crash Data:**
   * **Source:** [City of Chicago's open data portal](https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if/about_data).
   * **Details:** Includes traffic crash records reported within Chicago city streets as part of the electronic crash reporting system (E-Crash) at the Chicago Police Department (CPD).
   * **Records:** Over 837K records starting from September 2017.
2. **People Involved in Crashes Data:**
   * **Source:** [City of Chicago's open data portal](https://data.cityofchicago.org/Transportation/Traffic-Crashes-People/u6pd-qa9d/about_data).
   * **Details:** Includes details about individuals involved in traffic crashes, such as demographics, injury severity, and their roles (e.g., driver, passenger).
   * **Records:** Over 1 million records.

A screenshot of a computer

Description automatically generated

**Combined Dataset:**

* After joining the two datasets on the common key (CRASH\_ID), the combined dataset has:
  + **Rows:** 1,859,283 rows.
  + **Columns:** 48 columns.

Objective of the Project

The objectives of this project is to analyze and provide insights into traffic safety and congestion on Chicago streets by integrating and examining multiple datasets. Specifically, we aim to:

1. **Analyze Traffic Crash Data:** Identify patterns and trends in traffic crashes, including factors such as weather conditions, lighting conditions, and traffic control devices involved.
2. **Examine People Involved in Crashes:** Integrate and analyze data about people involved in traffic crashes to understand the demographics, injury severity, and contributing factors related to the individuals in these incidents.
3. **Generate Meaningful Insights:** Use the integrated data to generate actionable insights that can help improve traffic safety measures and reduce the number of traffic-related incidents.
4. **Create an Interactive Dashboard:** Develop a user-friendly dashboard that visualizes the key findings and trends from the analysis, providing easy access to the insights for decision-makers and the public.

By combining data about traffic crashes with information about the people involved, we aim to provide a comprehensive view of traffic safety issues and contribute to more effective policy-making and resource allocation to enhance road safety in Chicago.

Process

Databricks was chosen for this project due to its powerful capabilities in handling big data and its seamless integration with various data sources and processing frameworks, including Apache Spark. Databricks provides:

* **Scalability:** Efficiently handle large datasets with distributed computing.
* **Collaboration:** Facilitate collaboration among team members through shared workspaces and notebooks.
* **Integration:** Easy integration with Azure services and other data sources.

**Step 1: Create Azure Storage Account and Ingest Data**

1. **Create Azure Storage Account:**
   * We logged in to the Azure portal and created a new resource, storage account.
   * The storage account was named aly6110finalproject.
   * We created a container within the storage account named traffic-data.
2. **Ingest Data:**
   * We uploaded the dataset Traffic\_Crashes\_-\_Crashes\_20240623.csv to the traffic-data container as a blob storage.

**Step 2: Set Up Databricks Cluster for Spark environment and Connect to Azure Data**

1. **Create a Databricks Cluster:**
   * We logged in to our Databricks workspace.
   * We navigated to "Clusters" and created a new cluster with the necessary configurations such as Runtime version etc.
2. **Connect the Cluster with a Notebook:**
   * We created a new notebook and attached it to the cluster.
3. **Connect to Azure Data:**
   * We used the following code to connect to Azure Blob Storage and read the CSV file into a DataFrame:

For Traffic Data:

A screenshot of a computer program

Description automatically generated

For People Data:

A screenshot of a computer program

Description automatically generated

**Step 3: Clean and process the Data** We cleaned the data to ensure accuracy and consistency. The cleaning process included:

* Removing redundant columns.
* Converting data types to appropriate formats for analysis.
* Dropped duplicate columns.
* Merged two dataframes namely people\_df and df using inner merge on crash\_record\_id.

**Step 4: Merge the Datasets** We merged the traffic crash data with the people involved in crashes data using a common key (e.g., CRASH\_RECORD\_ID) to integrate the datasets. This allows us to analyze the relationship between crashes and the people involved.

A close-up of a computer screen

Description automatically generated

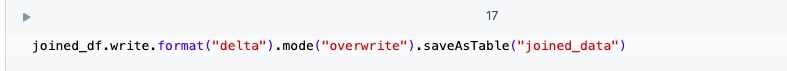
**Step 5: Convert Data into Delta Format and Create Batch and Serving Layers** To leverage the benefits of Delta Lake, such as ACID transactions, scalable metadata handling, and efficient query performance, we saved the cleaned DataFrame in Delta format. This step also established the batch and serving layers:

* **Batch Layer:** This layer manages the master dataset (the source of truth) and pre-computes batch views by ingesting and storing raw data in a structured format (Delta tables).
* **Serving Layer:** This layer indexes the batch views to allow for low-latency and ad-hoc queries, making the data readily accessible for analysis and visualization.

Because we used Delta format, we did not need to use Hive tables. Delta Lake provides the same benefits with added ACID (Atomicity, Consistency, Isolation, and Durability) transaction support and efficient file management.

Generally, Hadoop HDFS is used for storing large datasets, but in this project, we used Azure Blob Storage for its seamless integration with Databricks and also because we faced errors while uploading the dataset to Databricks.

The following code was used to save the data as a Delta table:



Analysis

1. **Count the number of crashes by traffic control device**

***SQL Syntax***

SELECT TRAFFIC\_CONTROL\_DEVICE, COUNT(\*) AS crash\_count

FROM joined\_data

GROUP BY TRAFFIC\_CONTROL\_DEVICE

ORDER BY crash\_count DESC

LIMIT 5

***Output***

A graph with orange bars

Description automatically generated

The SQL query retrieves and orders the crash counts for each traffic control device type, displaying the top five. The graph shows:

* **No Controls:** Highest crash count, indicating significant risks in areas without regulatory measures.
* **Traffic Signal:** Second highest, suggesting issues like signal timing or visibility despite control presence.
* **Stop Sign/Flasher:** Significant crashes, pointing to potential non-compliance or visibility problems.
* **Unknown:** Notable crashes due to unrecorded or unidentified control devices, indicating data gaps.
* **Other:** Includes less common controls, still contributing to crashes.

This highlights that the absence of controls and intersections with signals are major contributors to crashes.

1. **Find the number if crashes that occurred under different weather conditions**

***SQL Syntax***

SELECT WEATHER\_CONDITION, COUNT(\*) AS crash\_count

FROM joined\_data

GROUP BY WEATHER\_CONDITION

ORDER BY crash\_count DESC

LIMIT 4

***Output***

A screenshot of a graph

Description automatically generated

The SQL query counts the number of crashes under different weather conditions and orders them by crash count in descending order, displaying the top four. The graph reveals:

* **Clear Weather:** Highest crash count, indicating that most crashes occur under clear conditions.
* **Rain:** Second highest, suggesting that wet conditions significantly contribute to crashes.
* **Snow:** The fourth highest, showing that snowy conditions also contribute to a significant number of crashes.

This graph emphasizes that most crashes occur in clear weather, but adverse conditions like rain and snow still present substantial risks.

1. **Count the number of crashes by lighting condition**

***SQL Syntax***

SELECT LIGHTING\_CONDITION, COUNT(\*) AS crash\_count

FROM joined\_data

GROUP BY LIGHTING\_CONDITION

ORDER BY crash\_count DESC

LIMIT 4

***Output***

A screenshot of a computer

Description automatically generated

The SQL query counts the number of crashes under different lighting conditions and orders them by crash count in descending order. The graph shows:

* **Daylight:** Highest crash count, with 643,339 crashes, indicating that most accidents happen during the day.
* **Darkness, Lighted Road:** Second highest, with 185,831 crashes, highlighting the risks even on lighted roads at night.
* **Darkness:** Third highest, with 40,078 crashes, showing significant risk in unlit areas at night.
* **Unknown:** Notable number of crashes, 39,123, with unknown lighting conditions, suggesting possible data recording issues.

This graph underscores that while most crashes occur in daylight, nighttime conditions, even on lighted roads, present significant risks.

1. **Find the average number of crashes per month**

***SQL Syntax***

WITH month\_mapping AS (

SELECT 1 AS month\_num, 'January' AS month\_name

UNION ALL SELECT 2, 'February'

UNION ALL SELECT 3, 'March'

UNION ALL SELECT 4, 'April'

UNION ALL SELECT 5, 'May'

UNION ALL SELECT 6, 'June'

UNION ALL SELECT 7, 'July'

UNION ALL SELECT 8, 'August'

UNION ALL SELECT 9, 'September'

UNION ALL SELECT 10, 'October'

UNION ALL SELECT 11, 'November'

UNION ALL SELECT 12, 'December'

),

monthly\_crashes AS (

SELECT CRASH\_MONTH, COUNT(\*) AS crash\_count

FROM joined\_data

GROUP BY CRASH\_MONTH

)

SELECT m.month\_name AS crash\_month, AVG(mc.crash\_count) AS avg\_crashes

FROM month\_mapping m

LEFT JOIN monthly\_crashes mc

ON m.month\_num = mc.CRASH\_MONTH

GROUP BY m.month\_name, m.month\_num

ORDER BY m.month\_num;

***Output***

A screenshot of a computer

Description automatically generated

A graph with red lines

Description automatically generated

The SQL query calculates the average number of crashes per month. The line graph displays the average number of crashes for each month throughout the year.

**Graph Interpretation:**

* **April and October:** These months show peaks in the average number of crashes, indicating higher incidents during these times.
* **February and June:** These months have lower average crash counts, suggesting fewer accidents compared to other months.
* **General Trend:** There is variability in crash numbers across the year, with noticeable increases and decreases, reflecting potential seasonal factors affecting road safety.

This graph highlights the need to investigate the causes behind the higher crash rates in specific months, such as April and October, and consider implementing targeted safety measures during these periods.

1. **Determine the most common primary contributory causes of crashes.**

***SQL Syntax***

SELECT PRIM\_CONTRIBUTORY\_CAUSE, COUNT(\*) AS crash\_count

FROM joined\_data

GROUP BY PRIM\_CONTRIBUTORY\_CAUSE

ORDER BY crash\_count DESC

limit(5)

***Output***

A screenshot of a graph

Description automatically generated

The SQL query counts the number of crashes for each primary contributory cause and orders them by crash count in descending order, displaying the top five causes. The bar graph reveals:

* **Unable to Determine:** Highest crash count, indicating that a significant number of crashes could not have their primary cause identified.
* **Failing to Yield Right-of-Way:** Second highest, highlighting a common cause of accidents due to drivers not yielding.
* **Following Too Closely:** Third highest, showing that tailgating is a frequent cause of crashes.
* **Not Applicable:** Indicates cases where a primary cause was not applicable or not recorded.
* **Improper Overtaking/Passing:** Also a notable cause, though less frequent compared to the others listed.

This graph emphasizes that many crashes result from identifiable driver behaviors like failing to yield and following too closely, though a large portion remains undetermined, pointing to potential gaps in data collection or reporting.

1. **Generate a map visualization of crash locations using latitude and longitude**

***SQL Syntax***

SELECT

LATITUDE,

LONGITUDE,

COUNT(\*) AS crash\_count

FROM joined\_data

WHERE LATITUDE IS NOT NULL AND LONGITUDE IS NOT NULL

GROUP BY LATITUDE, LONGITUDE

LIMIT 50

***Output***

A map with dots on it

Description automatically generated

The SQL query counts the number of crashes at specific latitude and longitude coordinates, filtering out null values, and groups the results by these coordinates. It then limits the output to the top 50 locations. The map visualization plots these top 50 crash locations.

**Graph Interpretation:**

* **Geographical Distribution:** The map shows the geographical distribution of crashes, pinpointing the top 50 locations with the highest crash counts.
* **High-Frequency Areas:** The concentration of markers indicates areas with a higher frequency of crashes, highlighting potential hotspots for traffic incidents.
* **Data Accuracy:** The presence of specific latitude and longitude coordinates allows for precise identification of crash locations, which is essential for targeted interventions.

This map visualization provides a clear spatial understanding of where the most crashes occur, which can be crucial for traffic authorities to prioritize safety measures and allocate resources effectively.

1. **Distribution of crash type.**

***SQL Syntax***

SELECT CRASH\_TYPE, COUNT(\*) AS crash\_count

FROM joined\_data

GROUP BY CRASH\_TYPE

***Output***

A screenshot of a graph

Description automatically generated

The SQL query counts the number of crashes for each crash type and groups the results accordingly. The pie chart visualizes the distribution of crash types.

**Graph Interpretation:**

* **No Injury / Drive Away (73.2%):** The majority of crashes fall into this category, indicating that most crashes result in no injuries and the vehicles involved can drive away.
* **Injury and/or Tow Due to Crash (26.8%):** A significant portion of crashes result in injuries or require the vehicle to be towed.

This distribution highlights that while most crashes are minor with no injuries, a considerable percentage still result in injuries or significant vehicle damage requiring towing. This information can be useful for emergency response planning and resource allocation.

1. **Distribution of Crashes by Injury Classification.**

***SQL Syntax***

SELECT

INJURY\_CLASSIFICATION,

COUNT(\*) AS crash\_count

FROM joined\_data

GROUP BY INJURY\_CLASSIFICATION

ORDER BY crash\_count DESC

***Output***

**A screenshot of a graph

Description automatically generated**

The SQL query counts the number of crashes for each injury classification and orders them by crash count in descending order. The bar graph visualizes this data, showing the distribution of crashes by different injury classifications.

**Graph Interpretation:**

* **No Indication of Injury:** The majority of crashes fall into this category, indicating that most traffic incidents do not result in injuries.
* **Non-Incapacitating Injury:** This category has a notable number of crashes, reflecting incidents where injuries are present but not severe enough to incapacitate the individuals involved.
* **Reported, Not Evident:** A smaller number of crashes fall into this category, where injuries are reported but not immediately evident.
* **Incapacitating Injury:** This category shows fewer crashes compared to the others, indicating severe injuries that incapacitate the individuals involved.
* **Fatal:** The smallest number of crashes fall into this category, representing incidents resulting in fatalities.

This graph highlights that while most crashes do not result in injuries, a significant portion still leads to non-incapacitating injuries, and a smaller yet critical number result in severe injuries or fatalities. This information can be vital for traffic safety initiatives and prioritizing medical response resources.

1. **Patterns in crash occurrences with respect to time (hour of the day, day of the week, month) and location (latitude, longitude)?**

***SQL Syntax***

SELECT

CRASH\_HOUR,

CRASH\_DAY\_OF\_WEEK,

CRASH\_MONTH,

LATITUDE,

LONGITUDE,

COUNT(\*) AS crash\_count

FROM joined\_data

WHERE CRASH\_HOUR IS NOT NULL

AND CRASH\_DAY\_OF\_WEEK IS NOT NULL

AND CRASH\_MONTH IS NOT NULL

AND LATITUDE IS NOT NULL

AND LONGITUDE IS NOT NULL

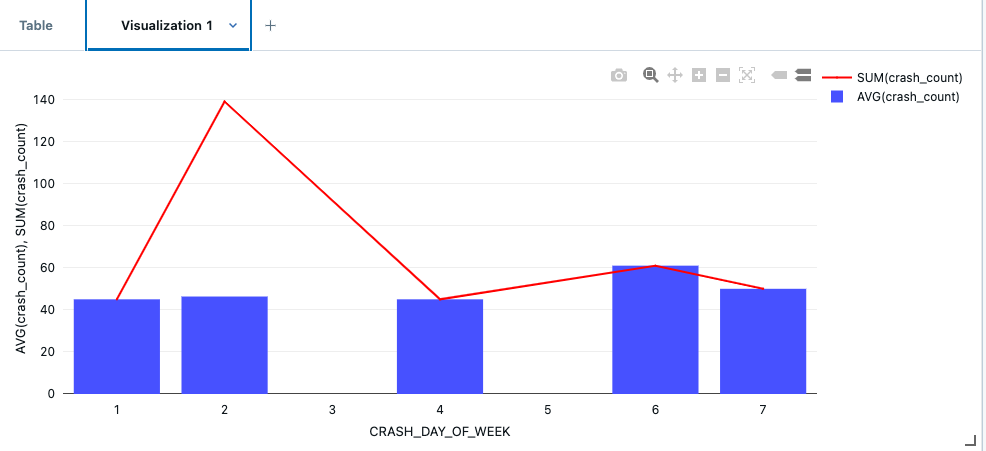
GROUP BY CRASH\_HOUR, CRASH\_DAY\_OF\_WEEK, CRASH\_MONTH, LATITUDE, LONGITUDE

ORDER BY crash\_count DESC

LIMIT 7

***Output***





The SQL query counts crashes grouped by the hour, day of the week, month, and location. The graph shows:

* **Day 4 (Wednesday):** Highest total crash count, indicating most crashes occur on Wednesdays.
* **Days 6 and 7 (Saturday and Sunday):** Higher average crash counts, suggesting weekends have more crashes.
* **Days 1 and 2 (Sunday and Monday):** Lower crash counts compared to other days.

This highlights that Wednesdays and weekends are more prone to crashes, suggesting a focus for traffic safety measures on these days.Top of Form

Dashboard

We created an interactive dashboard to visualize the key findings and trends from our analysis. This dashboard provides a user-friendly interface for exploring the data and gaining insights into traffic crashes in Chicago. The dashboard includes various charts, graphs, and maps to help users understand the distribution of crashes by different factors such as time, location, weather conditions, and injury severity.

#### Public Dashboard Link

You can access the public-facing dashboard using the following link:

<https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3190617666491842/2373282331786799/6555990026723924/latest.html>

A screenshot of a traffic crash analysis

Description automatically generated

A screenshot of a graph

Description automatically generated

This dashboard allows users to:

* View the distribution of crashes by day of the week and time of day.
* Analyze the impact of weather and lighting conditions on crash occurrences.
* Explore the geographical distribution of crashes across Chicago.
* Examine the severity of injuries and identify common causes of crashes.

By providing this interactive tool, we aim to make the insights from our analysis easily accessible to traffic authorities, policymakers, and the general public, helping to improve traffic safety in Chicago.

Conclusion

This project successfully analyzed traffic crash data to uncover important patterns and trends. By leveraging Databricks and Delta Lake, we efficiently processed and analyzed the data, generating valuable insights that can help improve traffic safety in Chicago. The interactive dashboard provides an accessible way to explore and understand the data, aiding decision-making and policy formulation.

References:

Ymericson. (n.d.). *GitHub - ymericson/big-data-streets: Big Data webapp using Chicago street congestion, crashes, red light violations, and speed camera violations*. GitHub. <https://github.com/ymericson/big-data-streets?tab=readme-ov-file>

*Traffic Crashes - Crashes | City of Chicago | Data Portal*. (2024, June 24). <https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if/about_data>

*Traffic Crashes - People | City of Chicago | Data Portal*. (2024, June 24). <https://data.cityofchicago.org/Transportation/Traffic-Crashes-People/u6pd-qa9d/about_data>

Data Science Basics. (2023, September 11). *Day 11: Create Dashboard From Databricks Notebook | 30 Days of Databricks* [Video]. YouTube. <https://www.youtube.com/watch?v=uZtzARQ2bw8>

Codebasics. (2022, December 16). *End To End Cricket Data Analytics Project Using Web Scraping, Python, Pandas and Power BI* [Video]. YouTube. <https://www.youtube.com/watch?v=4QkYy1wANXA>