C. ID: MIPDA2M2420
Date: 11 Mar 2024

CERTIFICATE

OF ACHIEVEMENT

MENTORNESS

IN **Data Analyst Internship**

This certificate is awarded as recognition to:

Darshan Kamleshbhai Viramgama

for successfully completing the 1 month Virtual Internship Program at MENTORNESS. He/She has showcased skills, professionalism, and valuable contributions during the internship. We believe that he/she has the potential to excel in future career endeavors.

Aashish
C.T.O
Mentorness

Rakesh
C.E.O
Mentorness

MENTORNESS INTERNSHIP PROGRAM
CERTIFIED
www.mentorness.com

Task Breakdown

Article Writing

In the first phase of the internship, I have showcased my ability to articulate and share knowledge by writing a comprehensive article on any one topic related to SQL or Power BI.

SQL Project

In this project, I had worked with a dataset related to a game. The dataset includes two tables: `Player Details` and `Level Details`. There are 15 questions for which you have to find the answers by writing SQL queries

Power BI Project

In this project aim was to conduct a comprehensive analysis of Chicago traffic incidents using Power BI, leveraging a dataset containing detailed information about crashes. The goal is to visualize and analyze traffic and crash patterns and also provide solutions.

Quiz on SQL & Power BI

To assess my grasp of SQL and Power BI concepts and my ability to apply them, a comprehensive quiz was conducted one 30th day of the internship

Article Writing

Data Preparation with Power Query

ARTICLE

on

DATA PREPARATION WITH POWER QUERY

Submitted To

MENTORNESS

IT Services and IT Consulting

Ahmedabad, Gujarat

In Partial Fulfilment –

Of Data Analyst Internship

Submitted by-

VIRAMGAMA DARSHAN KAMLESHBHAI

Pursuing MBA in Business Intelligence

BKBI, Gujarat University

Ahmedabad, Gujarat

DATA PREPARATION WITH POWER QUERY

What is a Power Query?

Power Query is a powerful data transformation and data prep engine. It's an essential part to do data acquisition regardless of the use case of your data. It allows users to connect to many different data sources, whether it be a database, an excel file, a web service, or cloud storage it can connect you to your data.

Here are the primary pieces that encompass Power Query.

1. **Data Connectivity:**

Power Query allows you to seamlessly connect to a massive amount of data sources, no matter if it's an extremely large data set, or a data source that isn't shaped how you need in order to be used Power Query can connect to your data.

2. **Transformations and Shaping:**

Power Query editor allows users to apply many different transformations to their data. This GUI is built in a way that it allows users to interactively build their queries, apply filters to their data, merge tables, and shape their data in other important ways.

The operations you define in a Power Query are repeatable, so that the data can be refreshed as long as needed.

3. **The ETL (Extract, Transform, Load) process:**

Power Query performs the whole ETL process: it extracts data from the source, transforms it to meet your requirements, and loads the data into your destination (such as Power BI).

Why Is Power Query Essential for Power BI?

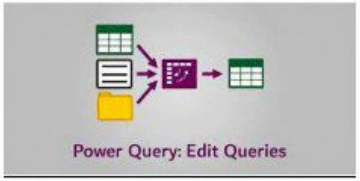
Clean Data for Visualizations:

For visualizations, Power Query helps you avoid misleading errors in data.

Before any visualization appears: With Power Query your data are cleaned up and transformed. You'll no longer have to worry about graphs that 'lie' or misleading tables in Power BI. There's too much 'tell' in 'showing'.

Simply put, Power Query is at the intersection of raw data and insightful visualizations. It flexes the muscles of those using it to more efficiently manage their data and thereby makes Power BI an even better tool for understanding and analyzing data.

How Power Query Helps :



1. **Simplifying Data Acquisition:**

   Business users often spend a lot of time on data preparation. Power Query combines data of varying data formats to get consistent connectivity experiences from data sources.

   Whatever databases or irregular files you're handling, Power Query provides you a unified experience.

2. **Interactive and Intuitive Shaping:**

   Power Query 's interface is interactive, and allows users to build queries over just about any data source. You can easily modify the data structure, filter rows and create calculated columns.

   Changes you make during the reshaping process will always be reflected in the future, keeping the same

3. **Handling Data Volume and Variety:**

   Power Query lets you work against a subset of the entire dataset. You can define the data you need and perform the necessary transformations on it. At the same time, Power Query won't overload your system.

   Power Query adapts, whether your data amounts to only one email or 10 trillion bytes, and whether it arrives from Hive at 1 msg/day or AMPLab at 100,000 messages/millisecond.

4. **Scheduled Refresh and Programmability:**

   Power Query queries can be manually refreshed, or automatically scheduled (e.g., in Power BI).

   To refresh your data now, you can use the Excel object model or other tools programmatically.

COMBINING QUERIES

Appending Queries in Power Query

Appending queries is a prominent feature in **Power BI** that allows you to create a joint table by combining similar data tables. Imagine it as stacking one table on top of another.

Here's how it works:

1. **What does Appending?**

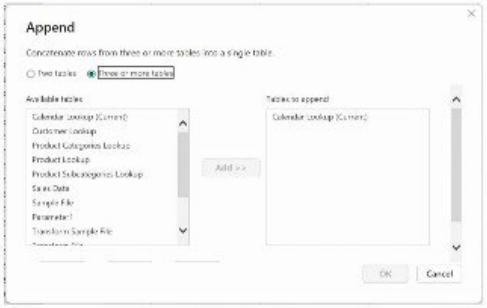
   Into one query combine the results for two or more queries, each represented by a table.

   Adding one table to another top-down is akin to appending in full result set of queries on a single table.

2. **How does Appending Work?**

   When you have multiple data sources with similar structures (e.g., daily sales data for different months).

   With the inclusion of these all of these queries into one concatenated dataset, you have an unified data one.



3.How to Append Queries:

Suppose you have two datasets Sales Data 2020 & Sales Data 2021.

- Append Queries from the Home tab.
- Choose Queries to Append.
- If you want to append only 2 table than choose another table.
- But if you want to append 3 or more than 3 than choose '3 and more table' option next to it and than bring all those table which you want to append to the right.(make sure all selected table has the similar column)
- Than press "Ok"
- You will be getting a new table.

Merging Queries in Power Query

Another important skill is merging queries into some kind of a useful form. Continuing with this introduction to Merging:

1. **What is Merging?**

   By merging data or tables along their shared column(s), merging operations bind various datasets or tables together horizontally.

   Imagine plugging a series of different tables into each other to obtain a third table.

2. **Types of Merge:**

Inner Merge: Rows which match in both tables are retained.

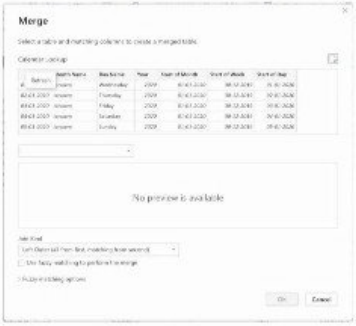
Left Outer Merge: All rows from the first table are preserved and only the rows from the second table which match are taken.

Right Outer Merge: Takes all of the rows from the second table and only the rows from the first table which match.

Full Outer Merge: Every row in both tables remains. Some missing values are filled in where a value is needed.

Left Anti Join: Left Anti Joins in Power Query only include rows in the left table where there are no matching rows in the right table.

Right Anti Join: Alternatively, in Power Query a right anti join returns only rows from the right table that do not match any rows in the left table.



3. How to Merge Queries:

Suppose we have Sales Data and Product Data.

Steps:

- In the Power Query Editor, select the Sales Data table as your base query.
- Click on the Home tab then click Merge.
- Specify (e.g.,Product Key) is the shared column between the two tables.

- Choose the second query column (e.g., Product Data) and corresponding to choose the first.
- Click OK to merge the queries.

Remember:

Merging queries will change the existing table, and merging queries as a new makes such a table out of the combined two or more. (Likewise, Append Queries)

You can choose several columns to join together into a dataset.

To sum up, no matter whether it is appending or merging that we are doing here, both are necessary operations for preparing your data within Power BI. No matter if you are stacking like tables on top of one another or taking data from different columns for orderly assembly; mastering these methods is sure to help make lighter your journey upward through data analysis!

TRANSFORM & ADD COLUMN

In Power Query, the "Transform" and "Add Column" features are both essential to effectively manipulating data. Even though both share many features, there is considerable difference between them.

Two features are for directly modifying data within your dataset, namely: renaming columns, changing data types, filtering rows, sorting data, and applications of other transformations.

Their primary feature is really about managing new data or existing columns in the dataset.

But "Add Column" extends the dataset by creating new columns through calculations or transformations based on existing data.

In short, "Transform" changes existing data, while "Add Column" takes the dataset and adds new information that is extracted from existing pieces. These two functions provide essential tools for preparing and analyzing data from power query. You will likewise have more flexibility with your dataset's structure as well as its content, providing you with the means to adapt it at will.

So lets discuss some important feature of Transform and Add Column:

Replace Value:

With this operation, you can swap out specific data in your dataset with others. To illustrate: Null values or incorrect data entries, or any other values that you want to get rid of shouldn't be hard to deal with using Power Query's "Replace Values" function.

Pivot and Unpivot Columns:

Pivoting changes unique values in a given column to separate columns. Typically this is done to summarize data. Unpivoting, on the other hand, goes in the opposite direction--it takes many columns and turns them into two: one for attribute names and another with their respective values. Actually, you may want to respin your data for analysis or reporting purposes.

Split Columns:

Doing this operation will let you take a single column and chop it up into many columns based on a delimiter(such as a comma or space). It's handy when you have data stored in a single column that needs to be separated into distinct categories.

Merge Columns:

It can also work in the opposite direction, taking data from two or more columns and putting it all into one. This can be handy in cases where you want to sort things out or try joining strings from different columns.

Standard and Scientific Numerical Operation:

Power Query offers up a whole selection of basic plus scientific calculations like addition, subtraction, multiplication and division. What's more, it even supports scientific operations too, including exponentiation, logarithms, trigonometric functions as well as statistical functions such as mean, median, mode, standard deviation, etc.

Statistical Tools:

Power Query now has a variety of statistical functions. They range from measures of central tendency (mean, median, mode), to measures of dispersion (standard deviation, variance), measures of association (correlation, covariance) and so on. These tools tell you where data stands in terms of distribution and relationship.

Date and Time Columns:

Power Query offers various functions for modifying date and time columns. Moreover, you can extract components like year, month, day, hour, minute, and second from datetime values. Moreover, you can compute operations such as the difference between two dates by using an expression; add or subtract days, months, years; and reformat dates in various styles.

SQL PROJECT

Decoding Gaming Behaviour

SQL PROJECT

Decode Gaming Behavior

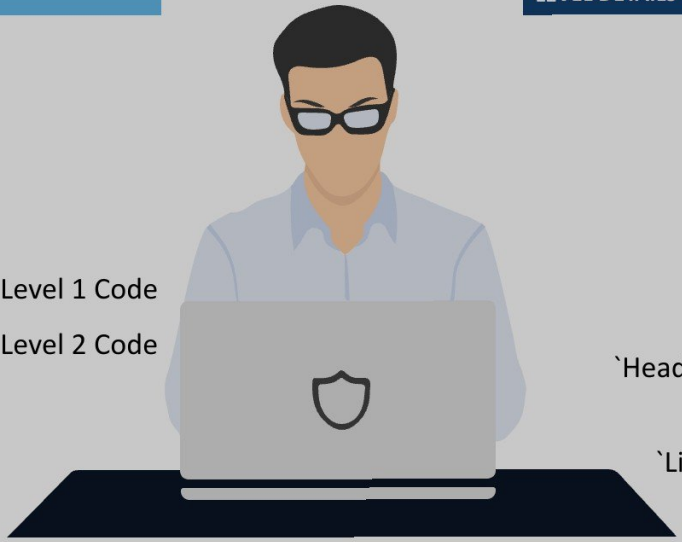
-Darshan Viramgama

DECODE GAMING BEHAVIOR

In this project we have two table Player Details and Level Details where we have to work on Loading Data, Data Manipulation and Data Extraction with using SQL. SQL a query language helps us to perform all this task easily. Below are the details of both the table:

PLAYERS DETAILS TABLE

- `P_ID`: Player ID
- `PName`: Player Name
- `L1_status`: Level 1 Status
- `L2_status`: Level 2 Status
- `L1_code`: Systemgenerated Level 1 Code
- `L2_code`: Systemgenerated Level 2 Code



LEVEL DETAILS TAB

`P_ID`: Player ID

`Dev_ID`: Device ID

`TimeStamp`: Start Time

`Stages_Cross`: Stages Crossed

`Level`: Game Level

`Difficulty`: Difficulty Level

`Kill_count`: Kill Count

`Headshots_count`: Headshots Count

`Score`: Player Score

`Lives_earned`: Extra Lives Earned

FUNCTIONS & CLAUSES

SELECT

SELECT statement: Used to retrieve data from one or more tables.

FROM Clause

FROM clause: Specifies the table(s) from which to retrieve the data..

ROW_NUMBER( )

ROW_NUMBER() function: Assigns a unique number to each row in the result set.

WHERE Clause

WHERE clause: Filters the rows based on specified conditions.

ORDER BY

ORDER BY clause: Sorts the result set based on specified columns.

GROUP BY

GROUP BY clause: Groups the rows based on specified columns..

JOIN Clause

JOIN clause: Combines rows from two or more tables based on related columns.

AVG(), COUNT(), DISTINCT, SUM()

This are all mathematical tools which are use to perform on table as per the problems.

FUNCTIONS & CLAUSES

RANK( )

RANK() function: Assigns a unique rank to each row based on specified criteria.

MIN( ) & MAX( )

MIN() & MAX() function is use to get the minimum and maximum value from the column

CREATE PROCEDURE

CREATE PROCEDURE allows you to define reusable blocks of SQL code with parameters that can be executed as needed to perform specific tasks within the database.

PARTITION BY

PARTITION BY clause is used with window functions to divide the result set into partitions or groups based on the values of one or more columns.

WITH Clause

WITH clause provides a way to create named subqueries that can be used like tables or views within the scope of a single query

OVER Clause

OVER clause is used with window functions to define the window. It allows you to specify how to partition, order, and frame the rows for the window function's calculations.

HAVING Clause

HAVING clause in SQL is used to filter rows in a result set based on a specified condition, similar to the WHERE clause, but we you this clause after grouping.

UPDATE & SET

'UPDATE' allows you to modify data in a table by specifying new values for one or more columns
SET function allows you to change or add new value to the column of the table.

STORED PROCEDURE

Stored Procedure is a precompiled collection of SQL statements and procedural logic that performs a specific task or set of tasks. Once created, you can execute the stored procedure multiple times without needing to recompile or rewrite the code.

Steps for Creating S.P.:

Definition: The S.P. statement starts with the keyword "**CREATE PROCEDURE**" followed by the name of the procedure you want to create.

Parameters: You can optionally specify input parameters that the procedure will accept. Parameters are enclosed in parentheses after the procedure name.

Body: The body of the procedure contains the SQL statements and procedural logic that define the task(s) to be performed. It is enclosed within the "**BEGIN"** and "**END"** keywords.

Execution: After creating the procedure, you can execute it using the "**EXECUTE**" or "**CALL**" statement, followed by the procedure name and any required parameter values.

```sql
-- Q1) Extract P_ID,Dev_ID,PName and Difficulty_level of all players
--     at level 0
SELECT level_details2.P_ID, level_details2.Dev_ID,
    player_details.PName, level_details2.Difficulty, level_details2.Level
FROM level_details2
JOIN player_details ON level_details2.P_ID=player_details.P_ID
WHERE level_details2.Level=0
GROUP BY player_details.P_ID,player_details.PName;

-- Q2) Find Level1_code wise Avg_Kill_Count where lives_earned is 2 and atleast
--    3 stages are crossed
SELECT player_details.L1_Code, AVG(level_details2.Kill_Count)
FROM level_details2
JOIN player_details ON player_details.P_ID=level_details2.P_ID
WHERE level_details2.Lives_Earned=2 AND level_details2.Stages_crossed>=3
group by player_details.L1_Code;

-- Q3) Find the total number of stages crossed at each diffuculty level
--     where for Level2 with players use zm_series devices. Arrange the result
--     in decsreasing order of total number of stages crossed.
SELECT Stages_crossed, Difficulty
FROM level_details2
WHERE Level=2 AND Dev_ID LIKE 'zm_%'
ORDER BY Stages_crossed DESC;
```

```sql
-- Q4) Extract P_ID and the total number of unique dates for those players
-- who have played games on multiple days.

-- Step 1: Add a date column
ALTER TABLE level_details2
ADD Datee VARCHAR(50)

-- Step 2: Populate the datee with split values
UPDATE level_details2
SET Datee = SUBSTRING(TimeStamp, 1, 10);

-- Step 3: Extract what we want
SELECT P_ID, COUNT(DISTINCT Datee) as 'Unique Date'
FROM level_details2
GROUP BY P_ID
HAVING COUNT(DISTINCT Datee) > 1;

-- Q5) Find P_ID and level wise sum of kill_counts where kill_count
-- is greater than avg kill count for the Medium difficulty.
WITH Medium_Avg AS (
SELECT AVG(Kill_Count) AS avg_kill_count
FROM level_details2
WHERE Difficulty = 'Medium' )
SELECT P_ID, level, SUM(Kill_Count) AS total_kill_count
FROM level_details2
WHERE Kill_Count > (SELECT avg_kill_count FROM Medium_Avg)
GROUP BY P_ID, Level;
```

```sql
-- Q6)  Find Level and its corresponding Level code wise sum of lives earned
-- excluding level 0. Arrange in asecending order of level.
SELECT player_details.L1_Code, level_details2.Level,
SUM(level_details2.Lives_Earned) as "Total_Lives_Earned"
FROM player_details
JOIN level_details2 on level_details2.P_ID=player_details.P_ID
WHERE level_details2.Level<>0
GROUP by player_details.L1_Code, level_details2.Level
ORDER BY level_details2.Level;

-- Q7) Find Top 3 score based on each dev_id and Rank them in increasing order
-- using Row_Number. Display difficulty as well.
WITH RankedScores AS (
 SELECT Dev_ID, score, difficulty, ROW_NUMBER() OVER (
PARTITION BY Dev_ID ORDER BY score) AS rank
FROM level_details2 )
SELECT Dev_ID, score, difficulty, rank
FROM RankedScores
WHERE rank <= 3
ORDER BY Dev_ID, rank;
```

```sql
-- Q8) Find first_login datetime for each device id
SELECT Dev_ID, MIN(TimeStamp) AS first_timestamp
FROM level_details2
GROUP BY Dev_ID;

-- Q9) Find Top 5 score based on each difficulty level and Rank them in
-- increasing order using Rank. Display dev_id as well.
WITH RankedScores AS (
SELECT Dev_ID, Score, Difficulty,
RANK() OVER (PARTITION BY difficulty
ORDER BY score DESC) AS score_rank
FROM level_details2 )
SELECT Dev_ID, Score, Difficulty
FROM RankedScores
WHERE score_rank <= 5
ORDER BY difficulty, Score;

-- Q10) Find the device ID that is first logged in(based on start_datetime)
-- for each player(p_id). Output should contain player id, device id and
-- first login datetime.
SELECT DISTINCT P_ID,Dev_ID, MIN(TimeStamp) AS first_timestamp
FROM level_details2
GROUP BY P_ID;
```

```sql
-- Q11) For each player and date, how many kill_count played so far by the player.
--That is, the total number of games played -- by the player until that date.
-- a) window function
 SELECT DISTINCT P_ID, MAX(Datee), SUM(Kill_Count) OVER (
PARTITION BY P_ID ORDER BY TimeStamp)
AS total_kills_so_far
FROM level_details2
GROUP by P_ID
ORDER BY P_ID;

-- b) without window function
Select P_ID, Datee, sum(Kill_Count)
FROM level_details
GROUP BY P_ID, Datee
ORDER BY P_ID, Datee

-- Q12) Find the cumulative sum of an stages crossed over a start_datetime
-- for each player id but exclude the most recent start_datetime
SELECT P_ID, SUM(Stages_crossed) OVER (
PARTITION BY P_ID ORDER BY TimeStamp ROWS
BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING)
AS cumulative_sum
FROM level_details2;
```

```sql
-- Q13) Extract top 3 highest sum of score for each device id and the corresponding player_id
WITH Sum_Score3 AS (
SELECT Dev_ID, P_ID, SUM(Score) AS total_score, ROW_NUMBER()
OVER (PARTITION BY Dev_ID ORDER BY SUM(score) DESC)
AS row_num
FROM level_details2
GROUP BY Dev_ID, P_ID)
SELECT row_num,Dev_ID, P_ID, total_score
FROM Sum_Score3
WHERE row_num<=3
order by Dev_ID,row_num;

-- Q14) Find players who scored more than 50% of the avg score scored by sum of
--      scores for each player_id
SELECT P_ID, SUM(Score) AS total_score
FROM level_details2
GROUP BY P_ID
HAVING SUM(Score) > (SELECT AVG(sum_score) * 0.5
FROM (SELECT SUM(Score) AS sum_score
FROM level_details2
GROUP BY P_ID) subquery);
```

```sql
-- Q15) Create a stored procedure to find top n headshots_count based on each dev_id
--      and Rank them in increasing orderusing Row_Number. Display difficulty as well.

CREATE PROCEDURE GetTopHeadshotsCountByDevID
@n INT AS
BEGIN
SELECT Dev_ID, Headshots_Count, Difficulty,
ROW_NUMBER() OVER (PARTITION BY Dev_ID
ORDER BY Headshots_Count ASC) AS rank
FROM level_details2
ORDER BY Dev_ID, rank
OFFSET 0 ROWS FETCH NEXT @n ROWS ONLY;
END;

EXEC GetTopHeadshotsCountByDevID @n = 5;
```

POWER BI PROJECT

Car Crash Analysis

DATA VISUALIZATION WITH POWER BI

CAR CRASH ANALYSIS OF CHICAGO

-Darshan Viramgama

AGENDA

01 Introduction

02 Objective

03 Data Overview

04 Dashboard

05 Interpretation

06 Solution and Awarness

INTRODUCTION

I had made this project on analyzing Traffic Crash Data of Chicago, the major concern of the given project is to provide a profound investigation into the multiple causes of car crashes. By examining both geographical and human aspects and by using the developed dashboard, the goal is to bring out the crucial patterns and tendencies that will lead to the discovery of the factors that underlie car crashes. By incorporating both the exploratory power of data as the primary source of information and the active intervention as an opportunity to work out strategies for raising public awareness, the given project aims to contribute to the development and evolution of new strategies for securing the safety of all road users in the area of Chicago.

OBJECTIVE

The objective is to conduct a comprehensive analysis of car crashes in Chicago, focusing on identifying the primary factors contributing to accidents, including both geographical and human factors. By examining geographical factors such as road conditions, intersections, and traffic patterns, as well as human factors such as rule braking, distraction, and impairment, the aim is to pinpoint the root causes of crashes. Also we will analysis proportion and type of injury cause by the crashes

Through this analysis, the goal is to raise awareness among the public about the key factors contributing to crashes and their implications for road safety. Additionally, the objective is to propose effective strategies and interventions to mitigate these factors and reduce the frequency and severity of accidents in Chicago.

DATA OVERVIEW

In the current project we are using 16-19 columns from the original dataset and also made some of measure columns by manipulating the data. Total number of rows in the project dataset is clean and filtered up and brought around 247K . Using this columns we have made a dashboard which helps use to drive various insights out of it.

1. CRASH_RECORD_ID: Unique identifier for each crash record.
2. CRASH_DATE: Actual date and time of the crash.
3. WEATHER_CONDITION: Weather conditions at the time of the crash.
4. LIGHTING_CONDITION: Lighting conditions at the crash location.
5. PRIM_CONTRIBUTORY_CAUSE: Primary contributing cause of the crash.
6. SEC_CONTRIBUTORY_CAUSE: Secondary contributing cause of the crash.
7. STREET_NAME: Street name at the crash location.
8. NUM_UNITS: Number of units involved in the crash.
9. MOST_SEVERE_INJURY: Most severe injury reported.
10. INJURIES_TOTAL: Total number of injuries reported.
11. INJURIES_FATAL: Number of fatal injuries reported.
12. INJURIES_INCAPACITATING: Number of incapacitating injuries reported.
13. CRASH_DAY_OF_WEEK: Day of the week when the crash occurred.
14. ROADWAY_SURFACE_COND: Condition of the roadway surface.
15. LATITUDE: Latitude of the crash location.
16. LONGITUDE: Longitude of the crash location.
17. LOCATION: Geographical coordinates of the crash location.

CAR CRASH ANALYSIS OF CHICAGO

2013 2024

247.55K
Count of CRASH_RECORD_ID

513K
Sum of NUM_UNITS

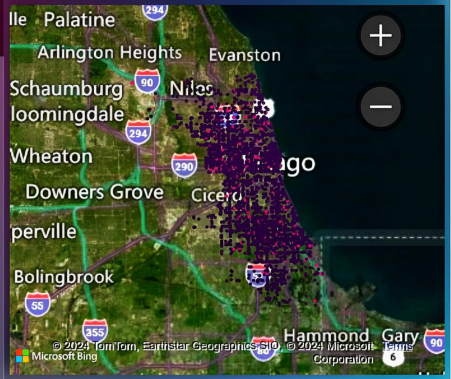
32K
Sum of INJURIES_TOTAL

No Injury 192K

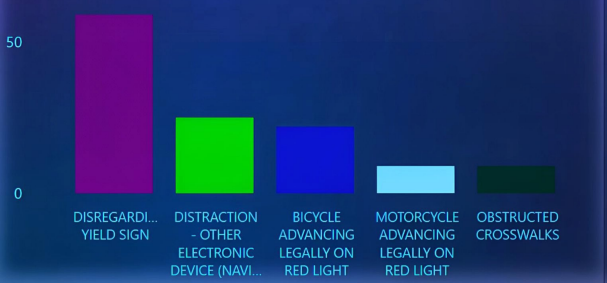
Injury 55K

Most Severe Injury

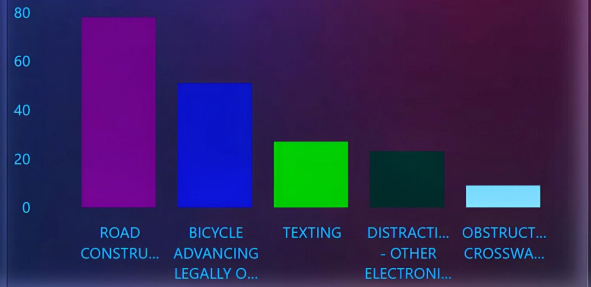
FATAL INCAPACITATING INJURY



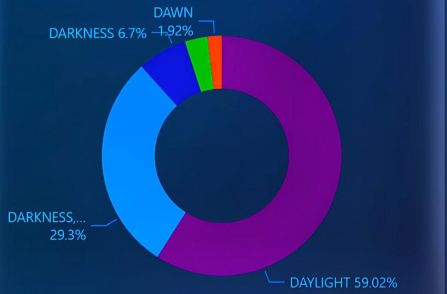
Top 5 Primary Cause



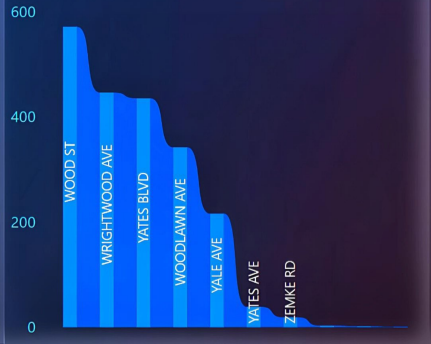
Top 5 Secondary Cause



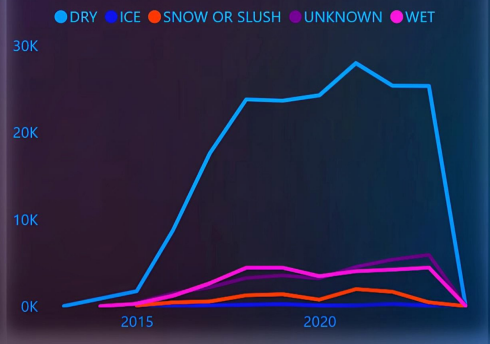
Lightening Condition



Top 10 Streets Causes Crash



Road Surface condition



Days of Week

INTERPRETATION

- Chicago witnessed 247.55k crash records between 2013 and 2024, with 55k being injurious crashes, emphasizing the urgent need for improved road safety measures.
- Key causes of crashes include failure to yield, distractions from electronic devices, road construction, and texting, highlighting areas for targeted interventions.
- Despite the majority of crashes occurring in daylight (59.02%), a significant proportion still happen in low-light conditions or darkness, emphasizing the importance of visibility and awareness at all times.
- Specific streets like Wood St and Wrightwood Ave are associated with higher crash rates, indicating the need for focused infrastructure upgrades and traffic management measures.
- Peak crash days are Saturdays, Fridays, and Sundays, suggesting the necessity for increased enforcement and awareness efforts during these times.
- High accidents are causing at Cityfront Place , West Garfield park and West Englewood
- Most crashes occurred on dry roads, followed by wet roads.Crash incidents on dry roads have steadily increased yearly, with a slight decline in 2020 which was due to covid.

SOLUTION AND AWARNESS

- Strengthen law enforcement and penalties for violations like failure to yield and distracted driving.
- Launch targeted public awareness campaigns to educate drivers about safe driving practices, especially regarding distractions and yielding right-of-way.
- Invest in infrastructure upgrades such as improved signage and lighting to enhance visibility and safety.
- Explore technology integration for automated enforcement systems and driver assistance technologies.
- Foster community engagement to identify local concerns and implement tailored solutions.
- Road Maintenance: Regular upkeep to ensure safe road conditions.
- Driver Education: Raise awareness about safe driving practices in various weather conditions.
- Implement targeted safety measures such as traffic calming measures, improved signage, and enhanced law enforcement in high-risk areas like Cityfront Place, West Garfield Park, and West Englewood to reduce accidents and improve road safety


THANK
YOU