Big Data Analysis for Chicago Traffic: `Enhancing Urban Mobility and Safety`

In this comprehensive notebook, we embark on an extensive journey through the intricate web of Chicago's traffic data. Utilizing cutting-edge Big Data technologies and methodologies, our primary aim is to unravel the complexities of traffic patterns, identify key factors contributing to congestion, and highlight potential areas for infrastructural improvements and policy intervention.

Overview

Chicago, known for its vibrant culture, iconic skyline, and bustling streets, also grapples with the challenges of urban traffic. From the daily commute to managing the influx of tourists, understanding traffic flow becomes paramount for city planners and policymakers. This analysis leverages vast datasets to paint a detailed picture of the city's traffic dynamics.

Objectives

The analysis focuses on several key areas to address the multifaceted nature of traffic management and safety:

> Traffic Volume and Congestion Analysis: Understanding traffic patterns, identifying peak congestion times, and analyzing traffic flow to suggest improvements.

> Crash Data Insights: Analyzing data on vehicles and people involved in crashes, including fatalities, to identify high-risk areas and factors contributing to accidents.

> Red Light Camera Violations: Examining the effectiveness of red light cameras in reducing violations and improving safety at intersections.

> Traffic Camera Compliance and Safety: Assessing the impact of speed cameras on driving behavior and identifying correlations between camera locations and accident reductions.

> Accident Hotspots Identification: Analyze crash data to pinpoint locations with high incidences of traffic accidents, aiding in targeted safety measures.

> Impact of Weather Conditions: Investigate how various weather conditions affect traffic flow and accident rates, providing insights for weather-adaptive traffic management.

Datasets

Our analysis utilizes several key datasets, including but not limited to:

> Traffic Volume Counts: Data on the volume of vehicles passing through key points in the city.

> Crash Data: Detailed records of traffic accidents, including involved vehicles, people, and resulting fatalities.

> Red Light and Speed Camera Violations: Data on traffic violations captured by automated camera systems.

> Traffic Congestion Data: Metrics and indices representing the levels of traffic congestion in different areas and times.

> Weather Data: Historical weather conditions to correlate with traffic patterns and accident occurrences.

Methodology

Employing PySpark for its robust processing capabilities, we perform a variety of data transformations and aggregations. Our approach includes:

> Transforming data to ensure accuracy and relevance.

> Joining datasets to enrich our analysis with multiple dimensions of traffic data.

> Aggregating data to uncover trends in traffic volume, accident rates, and violations.

> Visualizing findings to present actionable insights in an accessible manner.

Insights and Recommendations

Through our analysis, we aim to uncover actionable insights that can guide efforts to enhance traffic flow, improve safety, and reduce congestion in Chicago. These insights will form the basis for recommendations to city planners, traffic authorities, and policymakers.

Conclusion

Big Data Analysis for Chicago Traffic offers a deep dive into the city's traffic dynamics, providing a data-driven foundation for enhancing urban mobility and safety. By leveraging big data, we can identify patterns, challenges, and opportunities for making Chicago's streets safer and more efficient for everyone.

→ IMPORTING LIBRARIES

In [1]:
```
import pyspark
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import col, sum as spark_sum
from pyspark.sql.functions import count
from pyspark.sql.functions import broadcast
from pyspark.sql.functions import to_date
from pyspark.sql.types import IntegerType
```

→ CREATING SPARK SESSIONS:

Azure_SQL, CSV, Maria_DB, Mongo_DB

In [2]:
```
# AzureSQLServer Session
spark_azuresql = SparkSession.builder\
    .config("spark.jars.packages", "com.microsoft.azure:spark-mssql-connector_2.12:1.2.0")\
    .getOrCreate()
server_name = "jdbc:sqlserver://mansi-server.database.windows.net"
database_name = "traffic-chicago"
url = server_name + ";" + "databaseName=" + database_name + ";"

# CSV Session
spark_csv = SparkSession.builder.getOrCreate()

# MariaDB Session
spark_maria = SparkSession.builder \
    .config("spark.driver.extraClassPath", "mariadb-java-client-3.3.2.jar") \
    .getOrCreate()
server = "localhost"
port = 3306
database = "redlight_violations"
jdbc_url = f"jdbc:mysql://{server}:{port}/{database}?permitMysqlScheme"
jdbc_driver = "org.mariadb.jdbc.Driver"
properties = {
    "user": "root",
    "password": "root",
    "driver": jdbc_driver
}

# MongoDB Session
spark_mongo = SparkSession.builder \
    .appName("MongoDBMflixAnalysis") \
    .config("spark.mongodb.input.uri", f"mongodb+srv://admin:admin@cluster0.xn5enkc.mongodb.net/dailytraffic.avgcount") \
    .config("spark.mongodb.output.uri", f"mongodb+srv://admin:admin@cluster0.xn5enkc.mongodb.net/dailytraffic.avgcount") \
    .config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:3.0.1") \
    .getOrCreate()
```

```
:: loading settings :: url = jar:file:/usr/local/spark/spark-3.5.0-bin-hadoop3/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/azureuser/.ivy2/cache
The jars for the packages stored in: /home/azureuser/.ivy2/jars
com.microsoft.azure#spark-mssql-connector_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-e3d99096-dbcd-4356-9395-956b1f98f27b;1.0
        confs: [default]
        found com.microsoft.azure#spark-mssql-connector_2.12;1.2.0 in central
        found com.microsoft.sqlserver#mssql-jdbc;8.4.1.jre8 in central
:: resolution report :: resolve 120ms :: artifacts dl 5ms
        :: modules in use:
        com.microsoft.azure#spark-mssql-connector_2.12;1.2.0 from central in [default]
        com.microsoft.sqlserver#mssql-jdbc;8.4.1.jre8 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   2   |   0   |   0   |   0   ||   2   |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-e3d99096-dbcd-4356-9395-956b1f98f27b
        confs: [default]
        0 artifacts copied, 2 already retrieved (0kB/4ms)
24/03/07 02:22:02 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/03/07 02:22:05 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
24/03/07 02:22:05 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
```

→ ETL Speed_Camera_Violations Data From CSV File

In [3]:
```
speed_camera_df = spark_csv.read.csv("/home/azureuser/Desktop/Datasources/Speed_Camera_Violations_20240226.csv",
                                     header=True)
print("Speed Camera Violations Data:")
speed_camera_df.show()
```

Speed Camera Violations Data:

ADDRESS	CAMERA ID	VIOLATION DATE	VIOLATIONS	X COORDINATE	Y COORDINATE	LATITUDE	LONGITUDE	LOCATION
1111 N HUMBOLDT	CHI010	04/10/2015	67	NULL	NULL	NULL	NULL	NULL
1111 N HUMBOLDT	CHI010	04/25/2015	71	NULL	NULL	NULL	NULL	NULL
1111 N HUMBOLDT	CHI010	04/14/2015	38	NULL	NULL	NULL	NULL	NULL
1111 N HUMBOLDT	CHI010	04/16/2015	55	NULL	NULL	NULL	NULL	NULL
1111 N HUMBOLDT	CHI010	04/24/2015	54	NULL	NULL	NULL	NULL	NULL
5520 S WESTERN	CHI069	04/26/2015	35	NULL	NULL	NULL	NULL	NULL
5520 S WESTERN	CHI069	08/07/2014	42	NULL	NULL	NULL	NULL	NULL
5529 S WESTERN	CHI068	08/07/2014	21	NULL	NULL	NULL	NULL	NULL
5529 S WESTERN	CHI068	07/07/2014	14	NULL	NULL	NULL	NULL	NULL
5520 S WESTERN	CHI069	08/04/2014	26	NULL	NULL	NULL	NULL	NULL
5520 S WESTERN	CHI069	08/09/2014	57	NULL	NULL	NULL	NULL	NULL
5520 S WESTERN	CHI069	08/05/2014	38	NULL	NULL	NULL	NULL	NULL
1111 N HUMBOLDT	CHI010	07/05/2014	90	NULL	NULL	NULL	NULL	NULL
1111 N HUMBOLDT	CHI010	07/01/2014	77	NULL	NULL	NULL	NULL	NULL
5529 S WESTERN	CHI068	07/31/2014	5	NULL	NULL	NULL	NULL	NULL
5520 S WESTERN	CHI069	07/11/2014	1	NULL	NULL	NULL	NULL	NULL
7738 S WESTERN	CHI065	07/18/2014	39	NULL	NULL	NULL	NULL	NULL
5520 S WESTERN	CHI069	07/14/2014	21	NULL	NULL	NULL	NULL	NULL
7739 S WESTERN	CHI064	07/07/2014	62	NULL	NULL	NULL	NULL	NULL
1111 N HUMBOLDT	CHI010	07/03/2014	108	NULL	NULL	NULL	NULL	NULL

only showing top 20 rows

In [4]:
```
# Check the number of rows
num_rows = speed_camera_df.count()
print("Number of rows in Speed Camera Violations Data:", num_rows)

# Check the number of columns
num_columns = len(speed_camera_df.columns)
print("Number of columns in Speed Camera Violations Data:", num_columns)
```

```
Number of rows in Speed Camera Violations Data: 372708
Number of columns in Speed Camera Violations Data: 9
```

In [5]:
```
speed_camera_df.dtypes
```

Out[5]:
```
[('ADDRESS', 'string'),
 ('CAMERA ID', 'string'),
 ('VIOLATION DATE', 'string'),
 ('VIOLATIONS', 'string'),
 ('X COORDINATE', 'string'),
 ('Y COORDINATE', 'string'),
 ('LATITUDE', 'string'),
 ('LONGITUDE', 'string'),
 ('LOCATION', 'string')]
```

In [6]:
```
speed_camera_df = speed_camera_df.withColumn("VIOLATION DATE", to_date("VIOLATION DATE", "MM/dd/yyyy"))
speed_camera_df = speed_camera_df.withColumn("VIOLATIONS", speed_camera_df["VIOLATIONS"].cast(IntegerType()))
speed_camera_df = speed_camera_df.withColumn("X COORDINATE", col("X COORDINATE").cast("double"))
speed_camera_df = speed_camera_df.withColumn("Y COORDINATE", col("Y COORDINATE").cast("double"))
speed_camera_df = speed_camera_df.withColumn("LATITUDE", col("LATITUDE").cast("double"))
speed_camera_df = speed_camera_df.withColumn("LONGITUDE", col("LONGITUDE").cast("double"))
speed_camera_df.printSchema()
```

```
root
 |-- ADDRESS: string (nullable = true)
 |-- CAMERA ID: string (nullable = true)
 |-- VIOLATION DATE: date (nullable = true)
 |-- VIOLATIONS: integer (nullable = true)
 |-- X COORDINATE: double (nullable = true)
 |-- Y COORDINATE: double (nullable = true)
 |-- LATITUDE: double (nullable = true)
 |-- LONGITUDE: double (nullable = true)
 |-- LOCATION: string (nullable = true)
```

In [7]:
```
# Display the number of NA values in each column
speedcamera_na_counts = speed_camera_df.select(*(spark_sum(col(c).isNull().cast("int")).alias(c)
                                                 for c in speed_camera_df.columns))
print("Number of NA values in each column in Speed Camera Violations:")
speedcamera_na_counts.show()
```

```
Number of NA values in each column in Speed Camera Violations:
[Stage 5:==============>                                          (1 + 3) / 4]
```

ADDRESS	CAMERA ID	VIOLATION DATE	VIOLATIONS	X COORDINATE	Y COORDINATE	LATITUDE	LONGITUDE	LOCATION
0	5	0	0	13932	13932	13932	13932	13932

In [8]:
```
speed_camera_df = speed_camera_df.withColumnRenamed("VIOLATIONS", "SPEED CAMERA VIOLATIONS")
speed_camera_df.show()
```

```
+--------------+---------+--------------+----------------------+------------+------------+--------+---------+--------+
|       ADDRESS|CAMERA ID|VIOLATION DATE|SPEED CAMERA VIOLATIONS|X COORDINATE|Y COORDINATE|LATITUDE|LONGITUDE|LOCATION|
+--------------+---------+--------------+----------------------+------------+------------+--------+---------+--------+
|1111 N HUMBOLDT|   CHI010|    2015-04-10|                    67|        NULL|        NULL|    NULL|     NULL|    NULL|
|1111 N HUMBOLDT|   CHI010|    2015-04-25|                    71|        NULL|        NULL|    NULL|     NULL|    NULL|
|1111 N HUMBOLDT|   CHI010|    2015-04-14|                    38|        NULL|        NULL|    NULL|     NULL|    NULL|
|1111 N HUMBOLDT|   CHI010|    2015-04-16|                    55|        NULL|        NULL|    NULL|     NULL|    NULL|
|1111 N HUMBOLDT|   CHI010|    2015-04-24|                    54|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5520 S WESTERN|   CHI069|    2015-04-26|                    35|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5520 S WESTERN|   CHI069|    2014-08-07|                    42|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5529 S WESTERN|   CHI068|    2014-08-07|                    21|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5529 S WESTERN|   CHI068|    2014-07-07|                    14|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5520 S WESTERN|   CHI069|    2014-08-04|                    26|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5520 S WESTERN|   CHI069|    2014-08-09|                    57|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5520 S WESTERN|   CHI069|    2014-08-05|                    38|        NULL|        NULL|    NULL|     NULL|    NULL|
|1111 N HUMBOLDT|   CHI010|    2014-07-05|                    90|        NULL|        NULL|    NULL|     NULL|    NULL|
|1111 N HUMBOLDT|   CHI010|    2014-07-01|                    77|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5529 S WESTERN|   CHI068|    2014-07-31|                     5|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5520 S WESTERN|   CHI069|    2014-07-11|                     1|        NULL|        NULL|    NULL|     NULL|    NULL|
| 7738 S WESTERN|   CHI065|    2014-07-18|                    39|        NULL|        NULL|    NULL|     NULL|    NULL|
| 5520 S WESTERN|   CHI069|    2014-07-14|                    21|        NULL|        NULL|    NULL|     NULL|    NULL|
| 7739 S WESTERN|   CHI064|    2014-07-07|                    62|        NULL|        NULL|    NULL|     NULL|    NULL|
|1111 N HUMBOLDT|   CHI010|    2014-07-03|                   108|        NULL|        NULL|    NULL|     NULL|    NULL|
+--------------+---------+--------------+----------------------+------------+------------+--------+---------+--------+
only showing top 20 rows
```

In [9]:
```
speedcamera_aggregated_df = speed_camera_df.groupBy("VIOLATION DATE").agg(count("*").alias("SPEED_CAMERA_VIOLATION_COUNT"))
print("Aggregated Speed Camera Violations:")
speedcamera_aggregated_df.show()

num_rows = speedcamera_aggregated_df.count()
print("Number of rows in Aggregated Speed Camera Violations Data:", num_rows)
```

```
Aggregated Speed Camera Violations:
+--------------+--------------------------+
|VIOLATION DATE|SPEED_CAMERA_VIOLATION_COUNT|
+--------------+--------------------------+
|    2014-11-12|                       111|
|    2014-09-26|                       117|
|    2015-03-09|                       135|
|    2021-06-22|                       141|
|    2017-08-11|                        71|
|    2019-06-04|                       145|
|    2017-09-11|                       144|
|    2020-08-24|                        70|
|    2018-05-28|                        75|
|    2021-10-11|                        83|
|    2015-05-19|                       142|
|    2021-01-27|                        73|
|    2016-03-01|                       132|
|    2019-05-08|                       154|
|    2018-08-10|                        72|
|    2021-11-13|                        77|
|    2021-08-27|                        93|
|    2015-03-06|                       136|
|    2016-04-25|                       135|
|    2018-03-17|                        76|
+--------------+--------------------------+
only showing top 20 rows

Number of rows in Aggregated Speed Camera Violations Data: 3514
```

→ ETL Red_Light_Violations Data From MariaDB

In [10]:
```
red_light_df = spark_maria.read.jdbc(jdbc_url, "(select * from redlight_views) tab", properties=properties)
print("RedLight Camera Violations Data:")
red_light_df.show()
```

```
RedLight Camera Violations Data:
[Stage 18:>                                                         (0 + 1) / 1]
+--------------------+---------+--------------+----------+----------------+----------------+------------+------------+--------------------+
|             ADDRESS|CAMERA ID|VIOLATION DATE|VIOLATIONS|     X COORDINATE|     Y COORDINATE|    LATITUDE|    LONGITUDE|            LOCATION|
+--------------------+---------+--------------+----------+----------------+----------------+------------+------------+--------------------+
|4700 W IRVING PAR...|     2763|    04/09/2015|         4|            NULL|            NULL|        NULL|        NULL|                    |
|2400 W VAN BUREN ...|     2054|    04/14/2015|         5|            NULL|            NULL|        NULL|        NULL|                    |
|11500 S HALSTED S...|     2552|    04/08/2015|         5|            NULL|            NULL|        NULL|        NULL|                    |
|4700 W IRVING PAR...|     2764|    04/19/2015|         4|            NULL|            NULL|        NULL|        NULL|                    |
|3700 W IRVING PAR...|     1503|    04/23/2015|         3|            NULL|            NULL|        NULL|        NULL|                    |
|         2800 W 31ST|     2064|    09/14/2014|         3|            NULL|            NULL|        NULL|        NULL|                    |
|         2800 W 31ST|     2064|    12/16/2014|         1|            NULL|            NULL|        NULL|        NULL|                    |
|4700 S WESTERN AV...|     2141|    06/05/2019|         3|1161120.438879703|1873431.066474726|41.808378408|-87.684570717|(41.8083784079467...|
|         2800 W 31ST|     2064|    01/30/2015|         4|            NULL|            NULL|        NULL|        NULL|                    |
|6400 N WESTERN AV...|     1211|    06/05/2019|         2|1159113.585759434|1942451.622367891|41.997817807|-87.690033035|(41.997817807026,...|
|11500 S HALSTED S...|     2552|    03/28/2015|        14|            NULL|            NULL|        NULL|        NULL|                    |
|5500 S WENTWORTH ...|     2261|    04/06/2015|        11|            NULL|            NULL|        NULL|        NULL|                    |
|1600 W IRVING PAR...|     1153|    04/08/2015|         1|            NULL|            NULL|        NULL|        NULL|                    |
|3600 N CICERO AVENUE|     1612|    06/05/2019|         7|1143698.904601495|1923517.005821035|41.946163524|-87.747215169|(41.9461635236255...|
|2800 N KIMBALL AV...|     1581|    04/08/2015|         1|            NULL|            NULL|        NULL|        NULL|                    |
|   400 W BELMONT AVE|     1413|    06/05/2019|        75|1172981.744869382|1921577.376128802|41.940241193|-87.639639114|(41.9402411932557...|
|7900 S JEFFERY BO...|     2662|    08/04/2014|         1|            NULL|            NULL|        NULL|        NULL|                    |
|         2800 W 31ST|     2064|    07/31/2014|         3|            NULL|            NULL|        NULL|        NULL|                    |
|3000 W CHICAGO AV...|     1814|    06/05/2019|         8|1156075.266375505|1905216.604770718|41.895704525|-87.702218524|(41.8957045249462...|
|2800 N KIMBALL AV...|     1581|    07/13/2014|         4|            NULL|            NULL|        NULL|        NULL|                    |
+--------------------+---------+--------------+----------+----------------+----------------+------------+------------+--------------------+
only showing top 20 rows
```

In [11]:
```
# Check the number of rows
num_rows = red_light_df.count()
print("Number of rows in RedLight Camera Violations Data:", num_rows)

# Check the number of columns
num_columns = len(red_light_df.columns)
print("Number of columns in RedLight Camera Violations Data:", num_columns)
```

```
Number of rows in RedLight Camera Violations Data: 922510
Number of columns in RedLight Camera Violations Data: 9
```

In [12]:
```
red_light_df.dtypes
```

Out[12]:
```
[('ADDRESS', 'string'),
 ('CAMERA ID', 'int'),
 ('VIOLATION DATE', 'string'),
 ('VIOLATIONS', 'int'),
 ('X COORDINATE', 'double'),
 ('Y COORDINATE', 'double'),
 ('LATITUDE', 'double'),
 ('LONGITUDE', 'double'),
 ('LOCATION', 'string')]
```

In [13]:
```
red_light_df = red_light_df.withColumn("VIOLATION DATE", to_date("VIOLATION DATE", "MM/dd/yyyy"))
red_light_df.printSchema()
```

```
root
 |-- ADDRESS: string (nullable = true)
 |-- CAMERA ID: integer (nullable = true)
 |-- VIOLATION DATE: date (nullable = true)
 |-- VIOLATIONS: integer (nullable = true)
 |-- X COORDINATE: double (nullable = true)
 |-- Y COORDINATE: double (nullable = true)
 |-- LATITUDE: double (nullable = true)
 |-- LONGITUDE: double (nullable = true)
 |-- LOCATION: string (nullable = true)
```

In [14]:
```
# Display the number of NA values in each column
redlight_na_counts = red_light_df.select(*(spark_sum(col(c).isNull().cast("int")).alias(c)
                                           for c in red_light_df.columns))
print("Number of NA values in each column in RedLight Camera Violations:")
redlight_na_counts.show()
```

```
Number of NA values in each column in RedLight Camera Violations:
[Stage 22:>                                                         (0 + 1) / 1]
+-------+---------+--------------+----------+------------+------------+--------+---------+--------+
|ADDRESS|CAMERA ID|VIOLATION DATE|VIOLATIONS|X COORDINATE|Y COORDINATE|LATITUDE|LONGITUDE|LOCATION|
+-------+---------+--------------+----------+------------+------------+--------+---------+--------+
|      0|      289|             0|         0|       48102|       48102|   48102|    48102|       0|
+-------+---------+--------------+----------+------------+------------+--------+---------+--------+
```

In [15]:
```
red_light_df = red_light_df.withColumnRenamed("VIOLATIONS", "REDLIGHT CAMERA VIOLATIONS")
red_light_df.show()
```

```
[Stage 25:>                                                         (0 + 1) / 1]
```

```
+--------------------+---------+--------------+--------------------------+-----------------+-----------------+------------+-------------+--------------------+
|             ADDRESS|CAMERA ID|VIOLATION DATE|REDLIGHT CAMERA VIOLATIONS|     X COORDINATE|     Y COORDINATE|    LATITUDE|    LONGITUDE|            LOCATION|
+--------------------+---------+--------------+--------------------------+-----------------+-----------------+------------+-------------+--------------------+
|4700 W IRVING PAR...|     2763|    2015-04-09|                         4|             NULL|             NULL|        NULL|         NULL|                |
|2400 W VAN BUREN ...|     2054|    2015-04-14|                         5|             NULL|             NULL|        NULL|         NULL|                |
|11500 S HALSTED S...|     2552|    2015-04-08|                         5|             NULL|             NULL|        NULL|         NULL|                |
|4700 W IRVING PAR...|     2764|    2015-04-19|                         4|             NULL|             NULL|        NULL|         NULL|                |
|3700 W IRVING PAR...|     1503|    2015-04-23|                         3|             NULL|             NULL|        NULL|         NULL|                |
|         2800 W 31ST|     2064|    2014-09-14|                         3|             NULL|             NULL|        NULL|         NULL|                |
|         2800 W 31ST|     2064|    2014-12-16|                         1|             NULL|             NULL|        NULL|         NULL|                |
|4700 S WESTERN AV...|     2141|    2019-06-05|                         3|1161120.438879703|1873431.066474726|41.808378408|-87.684570717|(41.8083784079467...|
|         2800 W 31ST|     2064|    2015-01-30|                         4|             NULL|             NULL|        NULL|         NULL|                |
|6400 N WESTERN AV...|     1211|    2019-06-05|                         2|1159113.585759434|1942451.622367891|41.997817807|-87.690033035|(41.9978178070026,...|
|11500 S HALSTED S...|     2552|    2015-03-28|                        14|             NULL|             NULL|        NULL|         NULL|                |
|5500 S WENTWORTH ...|     2261|    2015-04-06|                        11|             NULL|             NULL|        NULL|         NULL|                |
|1600 W IRVING PAR...|     1153|    2015-04-08|                         1|             NULL|             NULL|        NULL|         NULL|                |
|3600 N CICERO AVENUE|     1612|    2019-06-05|                         7|1143698.904601495|1923517.005821035|41.946163524|-87.747215169|(41.9461635236255...|
|2800 N KIMBALL AV...|     1581|    2015-04-08|                         1|             NULL|             NULL|        NULL|         NULL|                |
|   400 W BELMONT AVE|     1413|    2019-06-05|                        75|1172981.744869382|1921577.376128802|41.940241193|-87.639639114|(41.9402411932557...|
|7900 S JEFFERY BO...|     2662|    2014-08-04|                         1|             NULL|             NULL|        NULL|         NULL|                |
|         2800 W 31ST|     2064|    2014-07-31|                         3|             NULL|             NULL|        NULL|         NULL|                |
|3000 W CHICAGO AV...|     1814|    2019-06-05|                         8|1156075.266375505|1905216.604770718|41.895704525|-87.702218524|(41.8957045249462...|
|2800 N KIMBALL AV...|     1581|    2014-07-13|                         4|             NULL|             NULL|        NULL|         NULL|                |
+--------------------+---------+--------------+--------------------------+-----------------+-----------------+------------+-------------+--------------------+
only showing top 20 rows
```

In [16]:
```
redlightcamera_aggregated_df = red_light_df.groupBy("VIOLATION DATE").agg(count("*").alias("REDLIGHT_CAMERA_VIOLATION_COUNT"))
print("Aggregated Speed Camera Violations:")
redlightcamera_aggregated_df.show()

num_rows = redlightcamera_aggregated_df.count()
print("Number of rows in Aggregated RedLight Camera Violations Data:", num_rows)
```

```
Aggregated Speed Camera Violations:
+--------------+-----------------------------+
|VIOLATION DATE|REDLIGHT_CAMERA_VIOLATION_COUNT|
+--------------+-----------------------------+
|    2014-11-12|                          270|
|    2014-09-26|                          315|
|    2019-05-08|                          260|
|    2017-08-11|                          260|
|    2018-05-28|                          262|
|    2015-05-19|                          261|
|    2016-03-01|                          246|
|    2015-03-09|                          254|
|    2019-06-04|                          262|
|    2018-08-10|                          280|
|    2020-08-24|                          259|
|    2017-09-11|                          255|
|    2021-06-22|                          273|
|    2021-11-13|                          264|
|    2021-01-27|                          231|
|    2021-10-11|                          247|
|    2021-08-27|                          276|
|    2021-12-18|                          248|
|    2022-03-28|                          261|
|    2022-07-31|                          254|
+--------------+-----------------------------+
only showing top 20 rows

[Stage 29:>                                                         (0 + 1) / 1]
Number of rows in Aggregated RedLight Camera Violations Data: 3513
```

→ Aggregating & Joining Red_Light and Speed_Camera Violations Data by Date

In [17]:
```
# Joined Violation Count DataFrame
violations_count_df = redlightcamera_aggregated_df.join(speedcamera_aggregated_df, "VIOLATION DATE", "inner")
print("Joined Violations Count DataFrame:")
violations_count_df = violations_count_df.orderBy("VIOLATION DATE")
violations_count_df.show()
```

```
num_rows = violations_count_df.count()
print("Number of rows in Total Aggregated Violations Count Data:", num_rows)
```

```
Joined Violations Count DataFrame:
```

```
+--------------+------------------------------+---------------------------+
|VIOLATION DATE|REDLIGHT_CAMERA_VIOLATION_COUNT|SPEED_CAMERA_VIOLATION_COUNT|
+--------------+------------------------------+---------------------------+
|    2014-07-01|                           286|                         95|
|    2014-07-02|                           279|                         92|
|    2014-07-03|                           300|                         94|
|    2014-07-04|                           293|                         62|
|    2014-07-05|                           287|                         63|
|    2014-07-06|                           284|                         60|
|    2014-07-07|                           285|                         97|
|    2014-07-08|                           289|                         98|
|    2014-07-09|                           284|                         97|
|    2014-07-10|                           298|                         97|
|    2014-07-11|                           301|                         83|
|    2014-07-12|                           282|                         59|
|    2014-07-13|                           291|                         62|
|    2014-07-14|                           293|                         96|
|    2014-07-15|                           282|                         98|
|    2014-07-16|                           301|                         99|
|    2014-07-17|                           293|                         97|
|    2014-07-18|                           293|                         85|
|    2014-07-19|                           296|                         62|
|    2014-07-20|                           294|                         63|
+--------------+------------------------------+---------------------------+
only showing top 20 rows
```

```
[Stage 41:>                                                         (0 + 1) / 1]
Number of rows in Total Aggregated Violations Count Data: 3513
```

→ ETL Avg_Traffic_Count Data From MongoDB

In [18]:
```
avg_traffic_count_df = spark_mongo.read.format("com.mongodb.spark.sql.DefaultSource") \
    .option("uri", "mongodb+srv://admin:admin@cluster0.xn5enkc.mongodb.net/dailytraffic.avgcount") \
    .load()
print("Avg Daily Traffic Counts Data:")
avg_traffic_count_df.select("ID", "Date of Count", "Street", "Total Passing Vehicle Volume", "Traffic Volume Count Location  Address").show()
```

```
Avg Daily Traffic Counts Data:
+----+-------------+----------------+---------------------------+-----------------------------------+
|  ID|Date of Count|          Street|Total Passing Vehicle Volume|Traffic Volume Count Location  Address|
+----+-------------+----------------+---------------------------+-----------------------------------+
| 414|   11/14/2006|          Lake St|                       7100|                          5838 West|
| 176|   03/28/2006|          76th St|                       8600|                           320 East|
|1367|   08/24/2006|          57th Dr|                      53500|                          1730 East|
| 316|   03/30/2006|          24th St|                        700|                           125 East|
|1294|   08/29/2006|         130th St|                       4200|                          2924 East|
| 507|   10/19/2006|          Lake St|                       6900|                          1931 West|
| 691|   08/15/2006|      Kimball Ave|                      15600|                          6067 North|
| 960|   08/22/2006|      Ashland Ave|                      26700|                          5116 North|
|  85|   05/02/2006|         State St|                      19300|                          6416 South|
|1116|   09/21/2006|      La Salle St|                      32300|                           435 North|
| 871|   08/16/2006|     Diversey Ave|                      16600|                          6891 West|
| 674|   08/29/2006|      Ashland Ave|                      33400|                          4034 North|
|1104|   09/27/2006|        Homan Ave|                      10700|                           626 North|
| 397|   10/11/2006|        Damen Ave|                      21500|                          6559 South|
| 160|   05/04/2006|       Harbor Ave|                       2500|                          9150 South|
|1286|   08/31/2006|       Halsted St|                      31800|                         10621 South|
|1260|   10/11/2006|Cottage Grove Ave|                      25000|                          5834 South|
| 819|   09/26/2006|      Pershing Rd|                      11000|                          2641 West|
| 119|   03/21/2006|       Halsted St|                      13500|                          6722 South|
| 518|   10/03/2006|      Chicago Ave|                      18100|                           161 East|
+----+-------------+----------------+---------------------------+-----------------------------------+
only showing top 20 rows
```

In [19]:
```
avg_traffic_count_df.printSchema()
```

```
root
 |-- Date of Count: string (nullable = true)
 |-- ID: long (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Location: string (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Street: string (nullable = true)
 |-- Total Passing Vehicle Volume: long (nullable = true)
 |-- Traffic Volume Count Location  Address: string (nullable = true)
 |-- Vehicle Volume By Each Direction of Traffic: string (nullable = true)
 |-- _id: struct (nullable = true)
 |    |-- oid: string (nullable = true)
```

In [20]:
```
# Display the number of NA values in each column
avg_traffic_na_counts = avg_traffic_count_df.select(*(spark_sum(col(c).isNull().cast("int")).alias(c)
                                                      for c in avg_traffic_count_df.columns))
print("Number of NA values in each column of Avg Daily Traffic Count Data:")
avg_traffic_na_counts.show()
```

```
Number of NA values in each column of Avg Daily Traffic Count Data:
+-------------+---+--------+--------+---------+------+---------------------------+-------------------------------------+---------------------
----------------------+---+
|Date of Count| ID|Latitude|Location|Longitude|Street|Total Passing Vehicle Volume|Traffic Volume Count Location  Address|Vehicle Volume By Eac
h Direction of Traffic|_id|
+-------------+---+--------+--------+---------+------+---------------------------+-------------------------------------+---------------------
----------------------+---+
|            0|  0|       0|       0|        0|     0|                          0|                                    0|
0|  0|
+-------------+---+--------+--------+---------+------+---------------------------+-------------------------------------+---------------------
----------------------+---+
```

→ ETL People_Fatalities Data From Azure_SQL

In [21]:
```
people_fatalities_df = spark_azuresql.read\
    .format("com.microsoft.sqlserver.jdbc.spark")\
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver")\
    .option("url", url)\
    .option("dbtable", "people_fatilities")\
    .option("user", "azureuser")\
    .option("password", "Mansi1234!") \
    .load()
```

In [22]:
```
# Display the number of NA values in each column
people_fatalities_na_counts = people_fatalities_df.select(*(spark_sum(col(c).isNull().cast("int")).alias(c)
                                          for c in people_fatalities_df.columns))
print("Number of NA values in each column in People Fatalities:")
people_fatalities_na_counts.show()
```

```
Number of NA values in each column in People Fatalities:
+---------+----------+------+--------------+-----------+---------------+----------+----+-----+-------+---+---+----------------+---------------+
--------+--------------------+
|PERSON_ID|CRASH_DATE|VICTIM|CRASH_LOCATION|PERSON_TYPE|CRASH_RECORD_ID|VEHICLE_ID|CITY|STATE|ZIPCODE|SEX|AGE|SAFETY_EQUIPMENT|AIRBAG_DEPLOYED|
EJECTION|INJURY_CLASSIFICATION|
+---------+----------+------+--------------+-----------+---------------+----------+----+-----+-------+---+---+----------------+---------------+
--------+--------------------+
|        0|       296|     0|             0|          0|              0|       193|   0|    2|     69|  1| 13|              39|            193|
167|                   0|
+---------+----------+------+--------------+-----------+---------------+----------+----+-----+-------+---+---+----------------+---------------+
--------+--------------------+
```

In [23]:
```
people_fatalities_df.printSchema()
```

```
root
 |-- PERSON_ID: string (nullable = true)
 |-- CRASH_DATE: timestamp (nullable = true)
 |-- VICTIM: string (nullable = true)
 |-- CRASH_LOCATION: string (nullable = true)
 |-- PERSON_TYPE: string (nullable = true)
 |-- CRASH_RECORD_ID: string (nullable = true)
 |-- VEHICLE_ID: string (nullable = true)
 |-- CITY: string (nullable = true)
 |-- STATE: string (nullable = true)
 |-- ZIPCODE: string (nullable = true)
 |-- SEX: string (nullable = true)
 |-- AGE: string (nullable = true)
 |-- SAFETY_EQUIPMENT: string (nullable = true)
 |-- AIRBAG_DEPLOYED: string (nullable = true)
 |-- EJECTION: string (nullable = true)
 |-- INJURY_CLASSIFICATION: string (nullable = true)
```

→ ETL Crash_Vehicle Data From Azure_SQL

In [24]:
```
crash_vehicle_df = spark_azuresql.read\
    .format("com.microsoft.sqlserver.jdbc.spark")\
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver")\
    .option("url", url)\
    .option("dbtable", "crash_vehicle_sum")\
    .option("user", "azureuser")\
    .option("password", "Mansi1234!") \
    .load()
```

In [25]:
```
crash_vehicle_df.printSchema()
```

```
root
 |-- CRASH_RECORD_ID: string (nullable = true)
 |-- POSTED_SPEED_LIMIT: string (nullable = true)
 |-- WEATHER_CONDITION: string (nullable = true)
 |-- LIGHTING_CONDITION: string (nullable = true)
 |-- FIRST_CRASH_TYPE: string (nullable = true)
 |-- TRAFFICWAY_TYPE: string (nullable = true)
 |-- ALIGNMENT: string (nullable = true)
 |-- ROADWAY_SURFACE_COND: string (nullable = true)
 |-- ROAD_DEFECT: string (nullable = true)
 |-- HIT_AND_RUN_I: string (nullable = true)
 |-- REPORT_TYPE: string (nullable = true)
 |-- DAMAGE: string (nullable = true)
 |-- PRIM_CONTRIBUTORY_CAUSE: string (nullable = true)
 |-- SEC_CONTRIBUTORY_CAUSE: string (nullable = true)
 |-- INJURIES_TOTAL: string (nullable = true)
 |-- INJURIES_FATAL: string (nullable = true)
 |-- UNIT_TYPE: string (nullable = true)
 |-- VEHICLE_ID: string (nullable = true)
 |-- MAKE: string (nullable = true)
 |-- MODEL: string (nullable = true)
 |-- VEHICLE_YEAR: string (nullable = true)
 |-- VEHICLE_DEFECT: string (nullable = true)
 |-- VEHICLE_TYPE: string (nullable = true)
 |-- VEHICLE_USE: string (nullable = true)
 |-- MANEUVER: string (nullable = true)
 |-- FIRST_CONTACT_POINT: string (nullable = true)
 |-- VEHICLE_COUNT: integer (nullable = true)
```

In [26]:
```
crash_vehicle_selected = crash_vehicle_df.select("CRASH_RECORD_ID", "POSTED_SPEED_LIMIT", "WEATHER_CONDITION", "LIGHTING_CONDITION", "PRIM_CON
                                                 "SEC_CONTRIBUTORY_CAUSE", "INJURIES_TOTAL", "INJURIES_FATAL", "VEHICLE_ID", "MAKE", "MODEL",

people_fatalities_selected = people_fatalities_df.select("CRASH_RECORD_ID", "PERSON_ID", "CRASH_DATE", "VICTIM", "CRASH_LOCATION", "ZIPCODE",
                                                         "SAFETY_EQUIPMENT", "AIRBAG_DEPLOYED", "EJECTION", "INJURY_CLASSIFICATION")

crashes_result_df = crash_vehicle_selected.join(people_fatalities_selected, "CRASH_RECORD_ID", how="left")
```

In [27]:
```
crashes_result_df.show(5)
```

```
[Stage 58:>                                                         (0 + 1) / 1]
+--------------------+------------------+-----------------+--------------------+----------------------+----------------------+--------------+-
-------------+----------+----+-------------+------------+---------+----------+------+--------------+-------+-----------+----+---------------+-
--------------+--------+--------------------+
|     CRASH_RECORD_ID|POSTED_SPEED_LIMIT|WEATHER_CONDITION|  LIGHTING_CONDITION|PRIM_CONTRIBUTORY_CAUSE|SEC_CONTRIBUTORY_CAUSE|INJURIES_TOTAL|I
NJURIES_FATAL|VEHICLE_ID|MAKE|        MODEL|VEHICLE_YEAR|PERSON_ID|CRASH_DATE|VICTIM|CRASH_LOCATION|ZIPCODE|PERSON_TYPE| AGE|SAFETY_EQUIPMENT|A
IRBAG_DEPLOYED|EJECTION|INJURY_CLASSIFICATION|
+--------------------+------------------+-----------------+--------------------+----------------------+----------------------+--------------+-
-------------+----------+----+-------------+------------+---------+----------+------+--------------+-------+-----------+----+---------------+-
--------------+--------+--------------------+
|000043c6564ec4d54...|                30|            CLEAR|DARKNESS, LIGHTED...|    UNABLE TO DETERMINE|   UNABLE TO DETERMINE|             0|
0|      NULL|NULL|         NULL|        NULL|     NULL|      NULL|  NULL|          NULL|   NULL|       NULL|NULL|            NULL|           NU
LL|    NULL|                NULL|
|000070ed7a6357c32...|                35|            CLEAR|DARKNESS, LIGHTED...|   FAILING TO YIELD ...|  DRIVING SKILLS/KN...|             0|
0|      NULL|NULL|         NULL|        NULL|     NULL|      NULL|  NULL|          NULL|   NULL|       NULL|NULL|            NULL|           NU
LL|    NULL|                NULL|
|00002c0771fb6f2c7...|                30|            CLEAR|               DAWN|    IMPROPER LANE USAGE|   UNABLE TO DETERMINE|             0|
0|      NULL|NULL|         NULL|        NULL|     NULL|      NULL|  NULL|          NULL|   NULL|       NULL|NULL|            NULL|           NU
LL|    NULL|                NULL|
|0000b70a00c8809f7...|                20|            CLEAR|            DAYLIGHT|    UNABLE TO DETERMINE|   UNABLE TO DETERMINE|             0|
0|   1004772|FORD|ECONOLINE 100|        1999|     NULL|      NULL|  NULL|          NULL|   NULL|       NULL|NULL|            NULL|           NU
LL|    NULL|                NULL|
|000013b0123279411...|                35|            CLEAR|            DAYLIGHT|   IMPROPER OVERTAKI...|  IMPROPER OVERTAKI...|             0|
0|      NULL|NULL|         NULL|        NULL|     NULL|      NULL|  NULL|          NULL|   NULL|       NULL|NULL|            NULL|           NU
LL|    NULL|                NULL|
+--------------------+------------------+-----------------+--------------------+----------------------+----------------------+--------------+-
-------------+----------+----+-------------+------------+---------+----------+------+--------------+-------+-----------+----+---------------+-
--------------+--------+--------------------+
only showing top 5 rows
```

→ Aggregating Crash Data by Make, Model, and Weather Condition

In [28]:
```
crashes_statistics_df = crashes_result_df.groupBy("MAKE", "MODEL", "WEATHER_CONDITION") \
    .agg(count("*").alias("CRASH_COUNT"))

# Show the aggregated data
crashes_statistics_df.show()
```

```
[Stage 64:>                                                         (0 + 1) / 1]
```

```
+--------------------+--------------------+--------------------+-----------+
|                MAKE|               MODEL|   WEATHER_CONDITION|CRASH_COUNT|
+--------------------+--------------------+--------------------+-----------+
|U-HAUL COMPANY (D...|OTHER (EXPLAIN IN...|               CLEAR|          4|
|FREIGHTLINER CORP...|  FREIGHTLINER CORP.|                RAIN|         88|
|GENERAL MOTORS CORP.|               Envoy|               CLEAR|         30|
|                JEEP|             UNKNOWN|                RAIN|         16|
|GENERAL MOTORS CO...|               ENVOY|               CLEAR|        328|
|           CHEVROLET|           SILVERADO|             UNKNOWN|         71|
|             GENERAL|             GENERAL|               CLEAR|         28|
|            CADILLAC|                 SRX|                RAIN|         27|
|               DODGE|         RAM 1500 PU|     CLOUDY/OVERCAST|         42|
|              NISSAN|                350Z|                RAIN|          5|
|           CHEVROLET|                 LUV|               CLEAR|          8|
|             LINCOLN|                 MKS|             UNKNOWN|          7|
|              SUBARU|OTHER (EXPLAIN IN...|                RAIN|         24|
|KENWORTH MOTOR TR...|KENWORTH MOTOR TR...|     CLOUDY/OVERCAST|          1|
|GENERAL MOTORS CO...|               ENVOY|FREEZING RAIN/DRI...|          4|
|                AUDI|                  A3|     CLOUDY/OVERCAST|          3|
|       MERCEDES-BENZ|            CL55 AMG|               CLEAR|          5|
|TOYOTA MOTOR COMP...|              4RUNNER|             UNKNOWN|          5|
|           CHEVROLET|           SILVERADO|               OTHER|         10|
|               SCION|                  XD|               CLEAR|         60|
+--------------------+--------------------+--------------------+-----------+
only showing top 20 rows
```

→ Writing DATA To AzureSQL_Database

In [29]:
```
server_name = "mansi-server.database.windows.net"
database_name = "traffic-chicago"
url = f"jdbc:sqlserver://{server_name};database={database_name}"
user = "azureuser"
password = "Mansi1234!"
```

In [30]:
```
speed_camera_df.write \
    .format("jdbc") \
    .option("url", url) \
    .option("dbtable", "speed_camera_violations") \
    .option("user", user) \
    .option("password", password) \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite").save()
```

In [31]:
```
red_light_df.write \
    .format("jdbc") \
    .option("url", url) \
    .option("dbtable", "red_light_violations") \
    .option("user", user) \
    .option("password", password) \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite").save()
```

In [32]:
```
violations_count_df.write \
    .format("jdbc") \
    .option("url", url) \
    .option("dbtable", "violations_count") \
    .option("user", user) \
    .option("password", password) \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite").save()
```

In [33]:
```
avg_traffic_count_df = avg_traffic_count_df.withColumn("oid", col("_id.oid"))
avg_traffic_count_df = avg_traffic_count_df.drop("_id")

avg_traffic_count_df.write \
    .format("jdbc") \
    .option("url", url) \
    .option("dbtable", "avg_traffic_count") \
    .option("user", user) \
    .option("password", password) \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite").save()
```

In [34]:
```
crashes_result_df.write \
    .format("jdbc") \
    .option("url", url) \
    .option("dbtable", "crashes_result") \
    .option("user", user) \
    .option("password", password) \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite").save()
```

```
24/03/07 02:23:47 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
```

In [35]:
```
crashes_statistics_df.write \
    .format("jdbc") \
    .option("url", url) \
    .option("dbtable", "crashes_statistics") \
```

```
    .option("user", user) \
    .option("password", password) \
    .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
    .mode("overwrite").save()
```

In [ ]: