🚗 Traffic Accident Analysis: Complex Questions

1 Accident Severity and Conditions

- How do **weather conditions, light conditions, and road surface conditions** impact **accident severity**?

2 Temporal Accident Patterns

- What are the **most accident-prone months, days, and time slots**?
- Is there a pattern in **weekdays vs. weekends**?

3 Geospatial Accident Hotspots

- Using **latitude and longitude**, can we identify accident hotspots using **clustering algorithms** (e.g., DBSCAN, K-Means)?

4 Speed Limit vs. Severity

- What is the **relationship between speed limits and accident severity**?
- Are higher speed limits linked to **more severe accidents**?

5 Vehicle Type and Casualties

- Which **vehicle types** are involved in the most severe accidents?
- Do certain vehicles correlate with **higher casualty numbers**?

6 Junction Control and Accidents

- Does the **presence or absence of junction control** (e.g., traffic signals, roundabouts) impact accident frequency and severity?

7 Multi-Vehicle Collisions

- What percentage of accidents involve **multiple vehicles**?
- How does **severity** change as the number of vehicles increases?

8 Rural vs. Urban Accident Trends

- Are **urban areas more accident-prone** than rural areas?
- How does accident severity differ between these regions?

9 District-Level Accident Trends

- Which **local authority districts** report the highest number of accidents?
- Are certain regions more accident-prone due to **weather, traffic, or road types**?

Installing PySpark

Method	Ease of Use	Dependencies Managed	Best For	Notes
Using pip	High	Partial (requires Java)	Quick local setups, clusters	Experimental, specify Hadoop version if needed, Python 3.8+ required
Using conda	High	Yes (includes Java)	Anaconda users, data science	Community-maintained, may lag behind releases
Manual Download	Low	No (manual setup)	Full control, custom setups	Requires setting environment variables, more complex
From Source	Very Low	No (manual setup)	Developers contributing to Spark	Involves building, not for general users

!pip install pyspark

importing findspark

- What it does: The findspark.init() function initializes the PySpark environment by adding PySpark to sys.path at runtime, making it importable. According to the GitHub repository findspark GitHub Repository, it uses the SPARK_HOME environment variable by default to locate the Spark installation. If SPARK_HOME is not set, it checks other possible locations, such as /usr/local/opt/apache-spark/libexec for OS X with brew install apache-spark.

In [1]:
```
import findspark
findspark.init()
```

Checking PySpark Version

You can check the installed PySpark version using the following code:

In [2]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("CheckVersion").getOrCreate()
print(spark.version)
```

```
3.5.4
```

In [3]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()
print(spark.version)
```

```
3.5.4
```

In [4]:
```
spark
```

Out[4]:

SparkSession - in-memory

SparkContext

Spark UI

Version	`v3.5.4`
Master	`local[*]`
AppName	`CheckVersion`

Read and Check and Selecting method

- PySpark Dataframe
- Reading the datasets
- Checking the datatypes of the columns(schema)
- Selecting Columns and Indexing
- Check Describe option similer to Pandas
- Adding columns
- Dropping columns

Reading a CSV File in PySpark

- You can read a CSV file using PySpark with the following code:
- SparkSession.builder.appName("ReadCSV").getOrCreate() initializes a Spark session.
- spark.read.csv("file location with file type", header=True, inferSchema=True) reads the CSV file:
- header=True: Treats the first row as column headers.
- inferSchema=True: Automatically detects column data types.

- df_csv.show() displays the first few rows of the dataset.

In [5]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, count, dayofweek, month, hour,when

# Create Spark session
spark = SparkSession.builder.appName("data").getOrCreate()
```

In [103…
```
# Read CSV file
df_data = spark.read.csv(r"D:\Resume_project\pyspark_analysis\AccidentData.csv", he

# Show the first few rows
df_data.show(2)
```

```
+--------------+-------------+-----+----+-----------+--------------------+----------
----------+-----------------+---------+---------------+-------------------------+-
-----------------+---------+--------------------+-----------------+--------------
----+----------------------+------------------+-----------+--------------------+----
--------------+------------------+--------------------+
|Accident_Index|Accident Date|Month|Year|Day_of_Week|    Junction_Control|     Junct
ion_Detail|Accident_Severity| Latitude|Light_Conditions|Local_Authority_(District)|C
arriageway_Hazards|Longitude|Number_of_Casualties|Number_of_Vehicles|       Police_F
orce|Road_Surface_Conditions|         Road_Type|Speed_limit|               Time|Urba
n_or_Rural_Area|Weather_Conditions|        Vehicle_Type|
+--------------+-------------+-----+----+-----------+--------------------+----------
----------+-----------------+---------+---------------+-------------------------+-
-----------------+---------+--------------------+-----------------+--------------
----+----------------------+------------------+-----------+--------------------+----
--------------+------------------+--------------------+
| 200901BS70001|   01-01-2021|  Jan|2021|   Thursday|Give way or uncon...|T or stagg
ered ju...|          Serious|51.512273|        Daylight|      Kensington and Ch...|
None|-0.201349|                  1|                2|Metropolitan Police|
Dry|    One way street|         30|2025-03-20 15:11:00|              Urban|Fine no h
igh winds|                 Car|
| 200901BS70002|   05-01-2021|  Jan|2021|     Monday|Give way or uncon...|
Crossroads|          Serious|51.514399|        Daylight|      Kensington and Ch...|
None|-0.199248|                 11|                2|Metropolitan Police|
Wet or damp|Single carriageway|         30|2025-03-20 10:59:00|              Urban|F
ine no high winds|Taxi/Private hire...|
+--------------+-------------+-----+----+-----------+--------------------+----------
----------+-----------------+---------+---------------+-------------------------+-
-----------------+---------+--------------------+-----------------+--------------
----+----------------------+------------------+-----------+--------------------+----
--------------+------------------+--------------------+
only showing top 2 rows
```

In [7]:
```
spark
```

Out[7]: **SparkSession - in-memory**

SparkContext

Spark UI

Version	`v3.5.4`
Master	`local[*]`
AppName	`CheckVersion`

Reading JSON and Parquet Files in PySpark

Reading a JSON File

- `spark.read.json("file.json")` is used to read JSON files.
- **Schema Inference**: PySpark automatically detects the schema, but specifying a schema is recommended for large files.
- **Nested Data**: JSON supports hierarchical data, which can be accessed using dot notation.

Reading a Parquet File

- `spark.read.parquet("file.parquet")` reads Parquet files efficiently.
- **Optimized for Performance**: Parquet is a columnar storage format, making queries faster and reducing storage space.
- **Schema Preservation**: Unlike CSV, Parquet maintains schema and data types.
- **Partitioning**: Supports partitioning for faster query execution.

🚀 **Tip**: Use Parquet for large datasets due to its compression and query efficiency.

In [8]:
```
df_json = spark.read.json("drivers.json")
```

In [104…
```
df_json.show(2)
```

```
+----+----------+--------+---------+----------------+----------+------+----------
---------+
|code|       dob|driverId|driverRef|            name|nationality|number|
url|
+----+----------+--------+---------+----------------+----------+------+----------
---------+
| HAM|1985-01-07|       1| hamilton|{Lewis, Hamilton}|    British|    44|http://en.w
ikiped...|
| HEI|1977-05-10|       2| heidfeld| {Nick, Heidfeld}|     German|    \N|http://en.w
ikiped...|
+----+----------+--------+---------+----------------+----------+------+----------
---------+
only showing top 2 rows
```

In [10]:
```
df_json.describe
```

Out[10]:
```
<bound method DataFrame.describe of DataFrame[code: string, dob: string, driverId: bigint, driverRef: string, name: struct<forename:string,surname:string>, nationality: string, number: string, url: string]>
```

In [11]:
```
df_data.describe()
```

Out[11]:
```
DataFrame[summary: string, Accident_Index: string, Accident Date: string, Month: string, Year: string, Day_of_Week: string, Junction_Control: string, Junction_Detail: string, Accident_Severity: string, Latitude: string, Light_Conditions: string, Local_Authority_(District): string, Carriageway_Hazards: string, Longitude: string, Number_of_Casualties: string, Number_of_Vehicles: string, Police_Force: string, Road_Surface_Conditions: string, Road_Type: string, Speed_limit: string, Urban_or_Rural_Area: string, Weather_Conditions: string, Vehicle_Type: string]
```

In [12]:
```
df_data.printSchema()
```

```
root
 |-- Accident_Index: string (nullable = true)
 |-- Accident Date: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Year: integer (nullable = true)
 |-- Day_of_Week: string (nullable = true)
 |-- Junction_Control: string (nullable = true)
 |-- Junction_Detail: string (nullable = true)
 |-- Accident_Severity: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Light_Conditions: string (nullable = true)
 |-- Local_Authority_(District): string (nullable = true)
 |-- Carriageway_Hazards: string (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Number_of_Casualties: integer (nullable = true)
 |-- Number_of_Vehicles: integer (nullable = true)
 |-- Police_Force: string (nullable = true)
 |-- Road_Surface_Conditions: string (nullable = true)
 |-- Road_Type: string (nullable = true)
 |-- Speed_limit: integer (nullable = true)
 |-- Time: timestamp (nullable = true)
 |-- Urban_or_Rural_Area: string (nullable = true)
 |-- Weather_Conditions: string (nullable = true)
 |-- Vehicle_Type: string (nullable = true)
```

1️⃣ 🚙 Accident Severity and Conditions

Question

How do **weather conditions, light conditions, and road surface conditions** impact **accident severity**?

Analysis Ideas

- Group accidents by severity and analyze the impact of weather conditions.
- Identify patterns in different lighting conditions (e.g., daytime vs. nighttime).
- Compare accident severity across different road surface conditions.

In [14]:
```
df_selected = df_data.select(
    "Accident_Severity",
    "Weather_Conditions",
    "Light_Conditions",
    "Road_Surface_Conditions"
)
df_selected.show()
```

```
+-----------------+-------------------+-------------------+----------------------
+
|Accident_Severity|  Weather_Conditions|    Light_Conditions|Road_Surface_Conditions
|
+-----------------+-------------------+-------------------+----------------------
+
|          Serious|  Fine no high winds|           Daylight|                   Dry
|
|          Serious|  Fine no high winds|           Daylight|            Wet or damp
|
|           Slight|  Fine no high winds|           Daylight|                   Dry
|
|          Serious|               Other|           Daylight|           Frost or ice
|
|          Serious|  Fine no high winds|Darkness - lights...|                   Dry
|
|           Slight|  Fine no high winds|           Daylight|                   Dry
|
|          Serious|  Fine no high winds|           Daylight|                   Dry
|
|           Slight|  Fine no high winds|           Daylight|                   Dry
|
|           Slight|  Fine no high winds|           Daylight|                   Dry
|
|           Slight|               Other|           Daylight|            Wet or damp
|
|           Slight|  Fine no high winds|Darkness - lights...|                   Dry
|
|           Slight|  Fine no high winds|Darkness - lights...|                   Dry
|
|           Slight|Raining no high w...|           Daylight|            Wet or damp
|
|           Slight|Raining no high w...|           Daylight|            Wet or damp
|
|           Slight|  Fine no high winds|           Daylight|                   Dry
|
|          Serious|  Fine no high winds|Darkness - lights...|                   Dry
|
|           Slight|  Fine no high winds|Darkness - lights...|                   Dry
|
|           Slight|  Fine no high winds|Darkness - lights...|                   Dry
|
|           Slight|  Fine no high winds|           Daylight|            Wet or damp
|
|           Slight|  Fine no high winds|Darkness - lights...|                   Dry
|
+-----------------+-------------------+-------------------+----------------------
+
only showing top 20 rows
```

In [15]:
```
df_weather = df_selected.groupby("Weather_Conditions","Accident_Severity").count().
```

In [16]:
```
df_weather.show()
```

```
+--------------------+----------------+------+
|  Weather_Conditions|Accident_Severity| count|
+--------------------+----------------+------+
|  Fine no high winds|          Slight|207574|
|  Fine no high winds|         Serious| 33654|
|Raining no high w...|          Slight| 30391|
|               Other|          Slight| 13318|
|Snowing no high w...|          Slight|  4390|
|Raining no high w...|         Serious|  4107|
|  Fine no high winds|           Fatal|  3230|
|Raining + high winds|          Slight|  3036|
|   Fine + high winds|          Slight|  2661|
|         Fog or mist|          Slight|  1434|
|               Other|         Serious|  1403|
|Snowing + high winds|          Slight|   476|
|Raining + high winds|         Serious|   440|
|   Fine + high winds|         Serious|   431|
|Snowing no high w...|         Serious|   418|
|Raining no high w...|           Fatal|   370|
|         Fog or mist|         Serious|   226|
|               Other|           Fatal|   137|
|Snowing + high winds|         Serious|    61|
|   Fine + high winds|           Fatal|    56|
+--------------------+----------------+------+
only showing top 20 rows
```

In [17]:
```
df_weather = df_selected.groupby("Light_Conditions","Accident_Severity").count().or
```

In [18]:
```
df_weather.show()
```

```
+--------------------+----------------+------+
|    Light_Conditions|Accident_Severity| count|
+--------------------+----------------+------+
|            Daylight|          Slight|196243|
|Darkness - lights...|          Slight| 50716|
|            Daylight|         Serious| 28644|
|Darkness - no lig...|          Slight| 12801|
|Darkness - lights...|         Serious|  8515|
|Darkness - no lig...|         Serious|  3078|
|Darkness - lighti...|          Slight|  2539|
|            Daylight|           Fatal|  2366|
|Darkness - lights...|          Slight|   981|
|Darkness - lights...|           Fatal|   846|
|Darkness - no lig...|           Fatal|   649|
|Darkness - lighti...|         Serious|   357|
|Darkness - lights...|         Serious|   146|
|            Daylight|           fatal|    33|
|Darkness - lighti...|           Fatal|    28|
|Darkness - lights...|           fatal|    16|
|Darkness - lights...|           Fatal|    15|
+--------------------+----------------+------+
```

In [19]:
```
df_road = df_selected.groupby("Road_Surface_Conditions","Accident_Severity").count(
```

In [20]:
```
df_road.show()
```

```
+----------------------+----------------+------+
|Road_Surface_Conditions|Accident_Severity| count|
+----------------------+----------------+------+
|                   Dry|          Slight|177582|
|           Wet or damp|          Slight| 70103|
|                   Dry|         Serious| 28692|
|          Frost or ice|          Slight| 10703|
|           Wet or damp|         Serious| 10274|
|                  Snow|          Slight|  4287|
|                   Dry|           Fatal|  2658|
|          Frost or ice|         Serious|  1259|
|           Wet or damp|           Fatal|  1088|
|                  Snow|         Serious|   443|
|  Flood over 3cm. deep|          Slight|   308|
|                Normal|          Slight|   297|
|          Frost or ice|           Fatal|   116|
|  Flood over 3cm. deep|         Serious|    53|
|                   Dry|           fatal|    35|
|                  Snow|           Fatal|    28|
|                Normal|         Serious|    19|
|           Wet or damp|           fatal|    14|
|  Flood over 3cm. deep|           Fatal|    13|
|                Normal|           Fatal|     1|
+----------------------+----------------+------+
```

⚡ Speed Limit vs. Severity

Question

- What is the **relationship between speed limits and accident severity**?
- Are higher speed limits linked to **more severe accidents**?

Analysis Ideas

- Compare accident severity at **different speed limits.**
- Analyze trends in **highway vs. city accidents.**
- Use **regression analysis** to measure correlation between speed and accident severity.

In [21]:
```
df_selected2 = df_data.select("Speed_limit","Accident_Severity")
```

In [22]:
```
df_selected2.show()
```

```
+-----------+-----------------+
|Speed_limit|Accident_Severity|
+-----------+-----------------+
|         30|          Serious|
|         30|          Serious|
|         30|           Slight|
|         30|          Serious|
|         30|          Serious|
|         30|           Slight|
|         30|          Serious|
|         30|           Slight|
|         30|           Slight|
|         30|           Slight|
|         30|           Slight|
|         30|           Slight|
|         30|           Slight|
|         30|           Slight|
|         30|          Serious|
|         30|           Slight|
|         30|           Slight|
|         30|           Slight|
|         30|           Slight|
+-----------+-----------------+
only showing top 20 rows
```

In [23]:
```
df_speed = df_selected2.groupBy("Speed_limit", "Accident_Severity").count().orderBy
df_speed.show()
```

```
+-----------+-----------------+------+
|Speed_limit|Accident_Severity| count|
+-----------+-----------------+------+
|         10|           Slight|     3|
|         15|           Slight|     2|
|         20|          Serious|   377|
|         20|            Fatal|    14|
|         20|           Slight|  2508|
|         30|            Fatal|  1488|
|         30|            fatal|    49|
|         30|           Slight|174427|
|         30|          Serious| 24076|
|         40|           Slight| 21987|
|         40|          Serious|  3295|
|         40|            Fatal|   368|
|         50|           Slight|  8507|
|         50|            Fatal|   220|
|         50|          Serious|  1464|
|         60|          Serious|  8817|
|         60|           Slight| 36696|
|         60|            Fatal|  1313|
|         70|            Fatal|   501|
|         70|           Slight| 19150|
+-----------+-----------------+------+
only showing top 20 rows
```

🕒 Temporal Accident Patterns

Question

- What are the **most accident-prone months, days, and time slots**?
- Is there a pattern in **weekdays vs. weekends**?

Analysis Ideas

- Perform **time-series analysis** to detect accident peaks.
- Compare accident rates between **weekdays and weekends.**
- Visualize accidents across **different times of the day** (morning, afternoon, evening, night).

In [44]:
```
from pyspark.sql.functions import to_date, date_format, col, month, dayofweek, hour

# Convert 'Accident Date' to proper date format
df_time = df_data.withColumn("Accident Date", to_date(col("Accident Date"), "dd-MM-
                 .withColumn("Month", date_format(col("Accident Date"), "MMMM")) \
                 .withColumn("Day_of_Week", date_format(col("Accident Date"), "EEEE
                 .withColumn("Hour", hour(col("Time")))

df_time.show(2)
```

```
+--------------+-------------+-------+----+-----------+--------------------+--------
------------+-----------------+---------+---------------+-------------------------
+-------------------+---------+--------------------+-----------------+-------------
------+----------------------+------------------+-----------+--------------------+--
-----------------+-----------------+--------------------+----+
|Accident_Index|Accident Date|  Month|Year|Day_of_Week|    Junction_Control|     Jun
ction_Detail|Accident_Severity| Latitude|Light_Conditions|Local_Authority_(District)
|Carriageway_Hazards|Longitude|Number_of_Casualties|Number_of_Vehicles|       Police
_Force|Road_Surface_Conditions|         Road_Type|Speed_limit|               Time|Ur
ban_or_Rural_Area|Weather_Conditions|        Vehicle_Type|Hour|
+--------------+-------------+-------+----+-----------+--------------------+--------
------------+-----------------+---------+---------------+-------------------------
+-------------------+---------+--------------------+-----------------+-------------
------+----------------------+------------------+-----------+--------------------+--
-----------------+-----------------+--------------------+----+
| 200901BS70001|   2021-01-01|January|2021|     Friday|Give way or uncon...|T or sta
ggered ju...|          Serious|51.512273|        Daylight|      Kensington and Ch...
|              None|-0.201349|                   1|                2|Metropolitan
Police|                    Dry|    One way street|         30|2025-03-20 15:11:00|
Urban|Fine no high winds|                  Car|  15|
| 200901BS70002|   2021-01-05|January|2021|    Tuesday|Give way or uncon...|
Crossroads|          Serious|51.514399|        Daylight|      Kensington and Ch...|
None|-0.199248|                  11|                2|Metropolitan Police|
Wet or damp|Single carriageway|         30|2025-03-20 10:59:00|              Urban|F
ine no high winds|Taxi/Private hire...|  10|
+--------------+-------------+-------+----+-----------+--------------------+--------
------------+-----------------+---------+---------------+-------------------------
+-------------------+---------+--------------------+-----------------+-------------
------+----------------------+------------------+-----------+--------------------+--
-----------------+-----------------+--------------------+----+
only showing top 2 rows
```

In [43]:
```
df_col = df_time.select("Month","Day_of_Week","Hour")
df_col.show(2)
```

```
+-------+-----------+----+
|  Month|Day_of_Week|Hour|
+-------+-----------+----+
|January|     Friday|  15|
|January|    Tuesday|  10|
+-------+-----------+----+
only showing top 2 rows
```

📍 Geospatial Accident Hotspots

Question

Can we identify accident hotspots using **clustering algorithms** (e.g., DBSCAN, K-Means) based on **latitude and longitude**?

Analysis Ideas

- Use **geospatial clustering** to detect high-risk locations.
- Compare accident hotspots between **urban and rural areas**.
- Use **heatmaps** to visualize accident frequency across different regions.

In [49]:
```
from pyspark.sql.functions import when, col

# Define weekend days based on their names
df_weekday_vs_weekend = df_time.withColumn(
    "Weekend", when(col("Day_of_Week").isin("Saturday", "Sunday"), "Weekend").other
)

# Count the number of accidents on weekends vs. weekdays
df_weekday_vs_weekend_count = df_weekday_vs_weekend.groupBy("Weekend").count()

# Show results
df_weekday_vs_weekend_count.show()
```

```
+-------+------+
|Weekend| count|
+-------+------+
|Weekday|215878|
|Weekend| 92095|
+-------+------+
```

In [55]:
```
from pyspark.sql.functions import col, count, dayofweek, month, hour,when,desc
df_time_slots = df_time.withColumn(
    "Time_Slot",
     when((col("Hour") >= 6) & (col("Hour") <12), "Morning")
    .when((col("Hour") >= 12) & (col("Hour") <18), "Afternoon")
    .when((col("Hour") >= 12) & (col("Hour") <18), "Evening")
    .otherwise("Night")

)

df_time_slot_count = df_time_slots.groupby("Time_Slot").count().orderBy(desc("count
df_time_slot_count.show()
```

```
+---------+------+
|Time_Slot| count|
+---------+------+
|Afternoon|132398|
|    Night| 89697|
|  Morning| 85878|
+---------+------+
```

🚙 Vehicle Type and Casualties

Question

- Which **vehicle types** are involved in the most severe accidents?
- Do certain vehicles correlate with **higher casualty numbers**?

Analysis Ideas

- Group accidents by **vehicle type** and compare severity levels.
- Identify whether **motorcycles, trucks, or cars** contribute to more casualties.
- Explore casualty trends for **commercial vs. personal vehicles.**

In [59]:
```
from pyspark.sql.functions import col, count, avg, desc

# Grouping by Vehicle_Type and Accident_Severity,Counts the number of accidents per
df_vehicle_severity = df_data.groupBy("Vehicle_Type", "Accident_Severity").count()

# Sorting by severity count
df_vehicle_severity = df_vehicle_severity.orderBy(desc("count"))

# Show the result
df_vehicle_severity.show()
```

```
+--------------------+-----------------+------+
|        Vehicle_Type|Accident_Severity| count|
+--------------------+-----------------+------+
|                 Car|           Slight|205079|
|                 Car|          Serious| 31583|
|Van / Goods 3.5 t...|           Slight| 13364|
|Motorcycle over 5...|           Slight|  9587|
|Bus or coach (17 ...|           Slight|  7465|
|Motorcycle 125cc ...|           Slight|  5845|
|Goods 7.5 tonnes ...|           Slight|  5587|
|Taxi/Private hire...|           Slight|  4742|
|Motorcycle 50cc a...|           Slight|  3159|
|                 Car|            Fatal|  3096|
|Motorcycle over 1...|           Slight|  2822|
|       Other vehicle|           Slight|  2127|
|Van / Goods 3.5 t...|          Serious|  2121|
|Goods over 3.5t. ...|           Slight|  2119|
|Motorcycle over 5...|          Serious|  1496|
|Bus or coach (17 ...|          Serious|  1135|
|Motorcycle 125cc ...|          Serious|   926|
|Goods 7.5 tonnes ...|          Serious|   878|
|Taxi/Private hire...|          Serious|   742|
|Minibus (8 - 16 p...|           Slight|   693|
+--------------------+-----------------+------+
only showing top 20 rows
```

🚦 Junction Control and Accidents

Question

Does the **presence or absence of junction control** (e.g., traffic signals, roundabouts) impact accident frequency and severity?

Analysis Ideas

- Compare accident rates at **controlled vs. uncontrolled intersections**.
- Identify the **most dangerous types of junctions.**
- Analyze the effect of **traffic signals on accident reduction.**

1 Compare accident rates at controlled vs. uncontrolled intersections

In [64]:
```
from pyspark.sql.functions import count, desc

# Count accidents by Junction_Control and sort by count
df_junction_control = df_data.groupBy("Junction_Control").count().orderBy(desc("cou

# Show the result
df_junction_control.show()
```

```
+--------------------+------+
|    Junction_Control| count|
+--------------------+------+
|Give way or uncon...|150045|
|Data missing or o...| 98056|
| Auto traffic signal| 32349|
|Not at junction o...| 25378|
|           Stop sign|  1685|
|   Authorised person|   460|
+--------------------+------+
```

2 Identify the most dangerous types of junctions

In [66]:
```
df_junction_detail = df_data.groupby("Junction_Detail").count().orderBy(desc("count
df_junction_detail.show()
```

```
+--------------------+------+
|     Junction_Detail| count|
+--------------------+------+
|Not at junction o...|123094|
|T or staggered ju...| 96718|
|          Crossroads| 29948|
|          Roundabout| 27264|
|Private drive or ...| 10875|
|      Other junction|  8315|
|           Slip road|  4265|
|More than 4 arms ...|  4148|
|     Mini-roundabout|  3346|
+--------------------+------+
```

3️⃣ Analyze the effect of traffic signals on accident severity

In [67]:
```
df_severity_junction = df_data.groupby("Junction_Control","Accident_Severity").coun
df_severity_junction.show()
```

```
+--------------------+-----------------+------+
|    Junction_Control|Accident_Severity| count|
+--------------------+-----------------+------+
|Give way or uncon...|           Slight|130585|
|Data missing or o...|           Slight| 81128|
| Auto traffic signal|           Slight| 28763|
|Not at junction o...|           Slight| 20889|
|Give way or uncon...|          Serious| 18237|
|Data missing or o...|          Serious| 15012|
|Not at junction o...|          Serious|  3990|
| Auto traffic signal|          Serious|  3284|
|Data missing or o...|            Fatal|  1908|
|           Stop sign|           Slight|  1499|
|Give way or uncon...|            Fatal|  1200|
|Not at junction o...|            Fatal|   499|
|   Authorised person|           Slight|   416|
| Auto traffic signal|            Fatal|   284|
|           Stop sign|          Serious|   175|
|   Authorised person|          Serious|    42|
|Give way or uncon...|            fatal|    23|
| Auto traffic signal|            fatal|    18|
|           Stop sign|            Fatal|    11|
|Data missing or o...|            fatal|     8|
+--------------------+-----------------+------+
only showing top 20 rows
```

🚗🚕 Multi-Vehicle Collisions

Question

- What percentage of accidents involve **multiple vehicles**?
- How does **severity** change as the number of vehicles increases?

Analysis Ideas

- Group accidents by the **number of vehicles involved**.
- Identify whether **multi-vehicle crashes** are more severe.
- Compare **single-vehicle vs. multi-vehicle** accident trends.

In [73]:
```
from pyspark.sql.functions import col, count, when, avg

# Total number of accidents
total_accidents = df_data.count()
```

```
# Count of accidents involving multiple vehicles (more than 1)
multi_vehicle_accidents = df_data.filter(col("Number_of_Vehicles") > 1).count()

# Percentage of multi-vehicle accidents
multi_vehicle_percentage = (multi_vehicle_accidents / total_accidents) * 100
print(f"Percentage of accidents involving multiple vehicles: {multi_vehicle_percent

# Group by the number of vehicles and get accident severity distribution
df_severity_by_vehicles = (
    df_data.groupBy("Number_of_Vehicles")
    .agg(
        count("*").alias("Total_Accidents"),
        avg(when(col("Accident_Severity") == "Fatal", 1).otherwise(0)).alias("Fatal
        avg(when(col("Accident_Severity") == "Serious", 1).otherwise(0)).alias("Ser
        avg(when(col("Accident_Severity") == "Slight", 1).otherwise(0)).alias("Slig
    )
    .orderBy(col("Number_of_Vehicles"))
)

# Show accident severity distribution by number of vehicles
df_severity_by_vehicles.show()
```

```
Percentage of accidents involving multiple vehicles: 69.69%
+-----------------+--------------+-------------------+------------------+-------
-----------+
|Number_of_Vehicles|Total_Accidents|    Fatal_Percentage| Serious_Percentage| Slight
_Percentage|
+-----------------+--------------+-------------------+------------------+-------
-----------+
|                1|          93349|0.018511178480755015|0.19225701400122122|0.78904
96952297293|
|                2|         183595|0.008943598681881314|0.10510090144067104|0.88578
66499632343|
|                3|          24226|0.015850738875588213|0.10831338231651944|0.87583
58788078924|
|                4|           5077|0.018908804412054364|0.12192239511522553|0.85897
18337600946|
|                5|           1127| 0.02839396628216504|0.13043478260869565|0.84117
12511091393|
|                6|            346| 0.03468208092485549|0.16184971098265896|0.80346
82080924855|
|                7|            127| 0.03937007874015748|0.25984251968503935|0.70078
74015748031|
|                8|             63|0.031746031746031744| 0.1746031746031746|0.79365
07936507936|
|                9|             31|                0.0|0.06451612903225806|0.93548
38709677419|
|               10|             12| 0.16666666666666666|0.16666666666666666|0.66666
66666666666|
|               11|              6|                0.0|                0.0|
1.0|
|               12|              4|                0.0|               0.25|
0.75|
|               13|              5|                0.0|                0.2|
0.8|
|               14|              2|                0.0|                0.0|
1.0|
|               16|              1|                1.0|                0.0|
0.0|
|               19|              1|                0.0|                1.0|
0.0|
|               32|              1|                0.0|                0.0|
1.0|
+-----------------+--------------+-------------------+------------------+-------
-----------+
```

🌆🏞️ Rural vs. Urban Accident Trends

Question

- Are **urban areas more accident-prone** than rural areas?
- How does accident severity differ between these regions?

Analysis Ideas

- Compare accident **frequency** in urban vs. rural settings.
- Identify **rural-specific risk factors** (e.g., lack of lighting, sharp turns).
- Use **spatial analysis** to detect accident concentration.

In [74]:
```
from pyspark.sql.functions import col, count, avg, when

# Count of accidents in Urban vs. Rural areas
df_urban_rural = (
    df_data.groupBy("Urban_or_Rural_Area")
    .agg(
        count("*").alias("Total_Accidents"),
        avg(when(col("Accident_Severity") == "Fatal", 1).otherwise(0)).alias("Fatal
        avg(when(col("Accident_Severity") == "Serious", 1).otherwise(0)).alias("Ser
        avg(when(col("Accident_Severity") == "Slight", 1).otherwise(0)).alias("Slig
    )
)

# Show results
df_urban_rural.show()
```

```
+------------------+--------------+--------------------+------------------+------
------------+
|Urban_or_Rural_Area|Total_Accidents|    Fatal_Percentage| Serious_Percentage| Sligh
t_Percentage|
+------------------+--------------+--------------------+------------------+------
------------+
|              Urban|         198532|0.007832490480124112| 0.1184292708480245|0.8734
914270747285|
|              Rural|         109441|0.021463619667217954|0.15741815224641587|0.8211
182280863661|
+------------------+--------------+--------------------+------------------+------
------------+
```

In [75]:
```
from pyspark.sql.functions import round

# Convert proportions to percentages
df_urban_rural = df_urban_rural.withColumn("Fatal_Percentage", round(col("Fatal_Per
df_urban_rural = df_urban_rural.withColumn("Serious_Percentage", round(col("Serious
df_urban_rural = df_urban_rural.withColumn("Slight_Percentage", round(col("Slight_P

# Show results
df_urban_rural.show()
```

```
+------------------+--------------+---------------+-----------------+-----------
------+
|Urban_or_Rural_Area|Total_Accidents|Fatal_Percentage|Serious_Percentage|Slight_Perc
entage|
+------------------+--------------+---------------+-----------------+-----------
------+
|             Urban|        198532|           0.78|            11.84|
87.35|
|             Rural|        109441|           2.15|            15.74|
82.11|
+------------------+--------------+---------------+-----------------+-----------
------+
```

🏙️ District-Level Accident Trends

Question

- Which **local authority districts** report the highest number of accidents?
- Are certain regions more accident-prone due to **weather, traffic, or road types**?

Analysis Ideas

- Rank districts by **total accident count.**
- Analyze **district-wise weather impact** on accident severity.
- Identify **high-risk road types** in different districts.

In [76]:
```
from pyspark.sql.functions import col, count, desc

# Count accidents per district
df_district_accidents = df_data.groupBy("Local_Authority_(District)").agg(count("*"

# Rank districts by accident count
df_district_accidents = df_district_accidents.orderBy(desc("Total_Accidents"))

# Show top 10 accident-prone districts
df_district_accidents.show(10)
```

```
+-------------------------+---------------+
|Local_Authority_(District)|Total_Accidents|
+-------------------------+---------------+
|                Birmingham|           6165|
|                     Leeds|           4140|
|                Manchester|           3132|
|                  Bradford|           3006|
|               Westminster|           2811|
|                 Sheffield|           2750|
|                 Liverpool|           2611|
|                  Cornwall|           2606|
|                    Barnet|           2302|
|          Bristol, City of|           2270|
+-------------------------+---------------+
only showing top 10 rows
```