TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Report IT-300

Business Intelligence and Database Management Systems

Comprehensive Analysis of Accident Data Using Business Intelligence Tools

---

Authors:

Maram Abdallah

Tasnim Chagetmi

Submitted to:

Dr Manel Abdelkader

Table of Contents

1. Introduction

This mini-project focuses on leveraging modern business intelligence (BI) tools to analyze **traffic collision data** with the goal of addressing critical road safety concerns. By exploring patterns and trends in traffic accidents, the project aims to provide valuable insights that can support stakeholders in designing interventions to reduce the frequency and severity of collisions.

The scope of this project involves the analysis of datasets related to traffic collisions to identify **key performance indicators (KPIs)** and derive meaningful insights. These insights are presented through an integrated and visually compelling **BI dashboard**.
By combining reports, graphs, and trends from multiple data sources, the dashboard ensures that decision-makers can easily access, interpret, and act upon data.

2. Problem Statement

Road traffic collisions remain a significant global challenge, resulting in substantial human and economic losses each year. These incidents are influenced by multiple factors, such as environmental conditions, road infrastructure, and driver behavior. Despite the availability of extensive datasets on road collisions, much of this information is underutilized due to its raw, unstructured nature. This project seeks to address this issue by organizing and analyzing traffic collision data to generate actionable insights. The ultimate goal is to support road safety initiatives by providing stakeholders with evidence-based insights into the factors contributing to collisions.

3. Objective

The primary objectives of this project are as follows:

1. **Clean and Transform Raw Data:** Prepare raw collision data for analysis by addressing inconsistencies, missing values, and redundant information to ensure data quality and reliability.
2. **Develop a Data Warehouse:** Design and implement a star schema to structure data efficiently for querying and reporting.
3. **Analyze Collision Data:** Extract insights into trends, severity, and contributing factors of traffic collisions, uncovering key patterns.
4. **Visualize Findings:** Build interactive dashboards that communicate key insights in an accessible and visually engaging manner for stakeholders.

4. Methodology

The methodology of this project revolves around implementing a robust **ETL** pipeline to prepare raw traffic collision data for analysis and visualization.
The project integrates **Python, SQLite,** and **Power BI** to clean, structure, analyze, and visualize the data. Each stage of the methodology is designed to ensure the reliability, accuracy, and usability of the data for generating actionable insights.

4.1. ETL Process Overview

1. **Extract:**
    - Data was sourced from two raw datasets:
        - A **CSV dataset** containing detailed traffic collision records for 2019.
        - A **JSON dataset** representing collision data for 2020.
    - The data was loaded into Python using libraries like pandas and json.
2. **Transform:**
    - **Data Cleaning:**
        - Addressed missing values through imputation or removal of irrelevant records.
        - Standardized categorical attributes (e.g., Road_Surface and Lighting_Conditions).
        - Normalized time and date formats for consistency.
    - **Schema Design:**
        - A **star schema** was implemented with a central fact table (Collisions_Fact) linked to six dimension tables:
            - ➔ Casualties_Dimension
            - ➔ Conditions_Dimension
            - ➔ Date_Dimension
            - ➔ Location_Dimension
            - ➔ Road_Dimension
            - ➔ Vehicles_Dimension
    - Data was processed in Python, structured into these tables, and stored in SQLite for efficient querying.
3. **Load:**
    - Cleaned and transformed data was exported as CSV files and loaded into **Power BI** for visualization and reporting.

4.2. Tools and Technologies

1. **Python:**
    - Used for data extraction, cleaning, transformation, and preparing data for the database.
    - Libraries: pandas, sqlite3, numpy, and json.
2. **SQLite:**
    - Chosen as the database for its lightweight nature and ease of integration.
    - Enabled efficient storage and querying of data using the star schema model.
3. **Power BI:**
    - Used to build dynamic and interactive dashboards.

5. Implementation

5.1 Data Gathering

The data gathering phase involved identifying, documenting, and exploring the datasets required for analysis. This step ensured a clear understanding of the data's structure, quality, and relevance to the project's goals. Two datasets were utilized for this project:

1. **Accidents-2019.csv (CSV Format)**
   Description: This dataset provides detailed records of accidents, including information about vehicles involved, road surface conditions, lighting, weather, and casualty details.
   Purpose: Crucial for analyzing accident trends, road conditions, and casualty patterns.
   Metadata:
   File Format: CSV
   Source: kaggle
   Number of Rows: 271
   Number of Columns: 45
   Data Types:
   - Categorical: 1st Road Class, Road Surface, Weather Conditions.
   - Numerical: Grid Ref: Easting, Grid Ref: Northing, Accident Date.
2. **CalderdaleCollisions2020.json (JSON Format)**
   Description: This dataset documents collisions within the Calderdale region for the year 2020, capturing similar attributes as the CSV dataset, such as road conditions, weather, lighting, and casualties.
   Purpose: Valuable for understanding collision trends and regional accident characteristics.
   Metadata:
   File Format: JSON
   Source: kaggle
   Number of Rows: 202
   Number of Columns: 40
   Data Types:
   - Categorical: 1st Road Class, Road Surface, Weather Conditions.
   - Numerical: Grid Ref: Easting, Grid Ref: Northing, Accident Date.

Data Exploration and Quality Assessment

Loading the Data

The datasets were loaded into Python using the Pandas library and documented in a Jupyter Notebook to facilitate systematic exploration. Code snippets below illustrate the loading process:

[19]:
```python
import pandas as pd
import json

# Load the datasets
csv_data = pd.read_csv('data/raw data /Accidents-2019.csv')

with open('data/raw data /CalderdaleCollisions2020.json', 'r') as file:
    json_data = pd.json_normalize(json.load(file))

# Preview the datasets
print("CSV Data Preview:")
display(csv_data.head())

print("\nJSON Data Preview:")
display(json_data.head())
```

Dataset Inspection

Shape of Datasets:

- CSV Dataset: 271 rows and 45 columns.
- JSON Dataset: 202 rows and 40 columns.

Attributes in Common: Both datasets shared attributes such as Road Surface, Lighting Conditions, Weather Conditions, and casualty details, which facilitates cross-referencing during analysis.

Missing Data Analysis

CSV Dataset: Significant missing values were observed in columns related to secondary vehicles and casualties:

- Type of Vehicle 2: 80 missing values.
- Type of Vehicle 3: 232 missing values.
- Higher-order casualty details (Type of Vehicle 4-7) had significant gaps.

JSON Dataset: Minimal missing values, with only 2 gaps in the Reference Number column.

[15]:
```python
print("CSV Missing Data Summary:")
print(csv_data.isnull().sum())

print("\nJSON Missing Data Summary:")
print(json_data.isnull().sum())
```

Unique Value Inspection

Road Surface Conditions:

- CSV Dataset: Categories included ['Dry', 'Wet / Damp', 'Frost / Ice', 'Flood (surface water over 3cm deep)', 'Snow'].
- JSON Dataset: Duplicate formatting issues were identified in ['Wet / damp', 'Wet / damp', 'Wet / damp'], necessitating standardization during the cleaning phase.

```
[16]: print("Unique Road Surface Conditions in CSV:")
      print(csv_data['Road Surface'].unique())

      print("\nUnique Road Surface Conditions in JSON:")
      print(json_data['Road Surface'].unique())
```

Structured Documentation in Jupyter Notebook

All steps and results from the data exploration phase were meticulously documented in a Jupyter Notebook to ensure transparency and reproducibility. The notebook includes:

- **Code Execution**: Scripts for loading data using pandas.read_csv() and pandas.json_normalize().
- **Data Previews**: Snapshots of the first few rows of each dataset to validate their structure and attributes.
- **Missing Data Analysis**: Count of missing values in each column, providing insight into potential challenges.
- **Unique Value Inspection**: Summaries of distinct values in key columns, highlighting formatting inconsistencies and areas for standardization.

The complete notebook will be made available on GitHub as part of this project's repository, offering readers and collaborators full visibility into the exploration process.

Conclusion

The data gathering and exploration phase provided a comprehensive understanding of the datasets' structure, quality, and potential challenges. This phase ensured readiness for the next steps of data cleaning and transformation, where identified issues, such as missing values and formatting inconsistencies, will be addressed to prepare the data for integration into a data warehouse.

5.2 . Data Cleaning and Preparation

Data Cleaning

The data cleaning phase is a critical component in ensuring the reliability and usability of datasets for analysis. This phase involved addressing key data quality issues such as missing values, formatting inconsistencies, and potential duplicates to prepare the data for subsequent transformations and analyses. The outputs from this phase now provide a robust and accurate foundation for the project.

[21]:
```
# Check for missing values
print("Missing Values in CSV Dataset:")
print(csv_data.isnull().sum())

print("\nMissing Values in JSON Dataset:")
print(json_data.isnull().sum())

# Check for duplicates
print("\nDuplicate Rows in CSV Dataset:", csv_data.duplicated().sum())
print("Duplicate Rows in JSON Dataset:", json_data.duplicated().sum())

# Check unique values for key columns
print("\nUnique Road Surface Conditions in CSV:")
print(csv_data['Road Surface'].unique())

print("\nUnique Road Surface Conditions in JSON:")
print(json_data['Road Surface'].unique())
```

Objectives of Data Cleaning

The cleaning process aimed to:

- Address missing values to avoid introducing biases or gaps in the analysis.
- Standardize categorical data for consistency across datasets and ensure compatibility for cross-referencing.
- Validate the structure and quality of the data by identifying and resolving duplicates and formatting issues.
- Prepare and save cleaned datasets in structured formats, ensuring readiness for integration and visualization.

Key Actions and Outcomes

Handling Missing Values

[25]:
```
# Handle missing values in CSV dataset
# Drop columns with excessive missing data
csv_data_cleaned = csv_data.drop(columns=[
    'Type of Vehicle 4', 'Casualty Class 4', 'Casualty Severity 4', 'Sex of Casualty 4', 'Age of Casualty 4',
    'Type of Vehicle 5', 'Casualty Class 5', 'Casualty Severity 5', 'Sex of Casualty 5', 'Age of Casualty 5',
    'Type of Vehicle 6', 'Casualty Class 6', 'Casualty Severity 6', 'Sex of Casualty 6', 'Age of Casualty 6',
    'Type of Vehicle 7', 'Casualty Class 7', 'Casualty Severity 7', 'Sex of Casualty 7', 'Age of Casualty 7'
])
```

➔ **CSV Dataset**:

- Missing values were primarily concentrated in columns related to secondary vehicles and casualties. Columns with excessive missing data (e.g., Type of Vehicle 4-7 and associated casualty details) were removed to maintain data integrity without overloading the dataset with incomplete information.
- Missing values in critical columns (Type of Vehicle 2 and Type of Vehicle 3) were imputed with the placeholder Unknown, preserving the completeness of the dataset while avoiding analytical bias.

➔ **JSON Dataset**:

- Minimal missing values were observed, with only 2 missing entries in the Reference Number column. These rows were removed to ensure the dataset contained only complete records.

Standardizing Formats

[27]:
```
# Standardize Road Surface in JSON dataset
json_data_cleaned['Road Surface'] = json_data_cleaned['Road Surface'].str.strip().replace({
    'Wet  / damp': 'Wet / Damp',
    'Wet /  damp': 'Wet / Damp',
    'Wet / damp': 'Wet / Damp'
})

# Preview unique values in Road Surface after standardization
print("Unique Road Surface Conditions in JSON (after cleaning):")
print(json_data_cleaned['Road Surface'].unique())
```

➔ **Road Surface**:

- The Road Surface column in the JSON dataset exhibited inconsistent formatting (e.g., variations of Wet / Damp). These entries were standardized to a uniform format to ensure consistency across datasets and facilitate accurate aggregation and analysis.

Validation of Data

- Both datasets were examined for duplicate rows, and no duplicates were identified, confirming the uniqueness of the data entries.
- Data types were validated to align with expected formats. For example, numerical fields such as Grid Ref: Easting and Accident Date were confirmed to be integers, while categorical fields such as Road Surface and Lighting Conditions were validated as strings.

Final Outputs

- The cleaned datasets were saved, ready for subsequent phases of the project:
    - **CSV Dataset**: Accidents-2019-cleaned.csv
    - **JSON Dataset**: CalderdaleCollisions2020-cleaned.json
- These outputs serve as a reliable foundation for data transformation, integration, and visualization.

Data Preparation (ETL/ELT)

This phase also involved laying the groundwork for the ETL/ELT process, ensuring that the cleaned datasets are prepared for efficient extraction, transformation, and loading into the data warehouse.

[32]:
```
# Save the cleaned CSV dataset
csv_data_cleaned.to_csv('data/cleaned data /Accidents-2019-cleaned.csv', index=False)

# Save the cleaned JSON dataset
json_data_cleaned.to_json('data/cleaned data /CalderdaleCollisions2020-cleaned.json', orient='records')

print("Cleaned datasets saved successfully!")
```

```
Cleaned datasets saved successfully!
```

Data Extraction

- The datasets were successfully extracted from their raw formats (CSV and JSON) and loaded into Pandas DataFrames for processing.
- This ensured compatibility and accessibility for cleaning and transformation tasks.

Data Transformation

- The cleaning process itself constituted a significant portion of the transformation phase, where:
    - Missing values were addressed.
    - Formats were standardized.
    - Non-essential columns were removed, reducing noise and improving data focus.

Data Loading

- The cleaned datasets were saved in structured formats:
    - **CSV** and **JSON** outputs were prepared to facilitate further loading into analytical tools or data warehouses.
- These structured outputs ensure that the datasets can be seamlessly integrated into the data warehouse or used directly for analysis.

Conclusion

The completion of the data cleaning and preparation phase has ensured that the datasets are now structured, reliable, and ready for integration and analysis. By addressing missing values, standardizing formats, and validating the data structure, this phase has provided a strong foundation for deriving meaningful insights and advancing to the next stages of the project.

5.3. Data Storage and Modeling

The primary goal of this phase was to design and build a robust Data Warehouse (DWH) to store the cleaned and transformed data. This phase ensures that the data is structured and optimized for querying and analysis, supporting efficient reporting and visualization.

Database Setup:

[64]:
```
import sqlite3

# Create the database file
conn = sqlite3.connect('accidents_dwh.sqlite')
print("Database created successfully!")
conn.close()
```

```
Database created successfully!
```

- The **SQLite** database named **accidents_dwh.sqlite** was created to serve as the central repository for the data warehouse.
- SQLite was chosen for its simplicity, lightweight nature, and compatibility with Python for development and testing.

[65]:
```
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect("accidents_dwh.sqlite")
cursor = conn.cursor()

# SQL script to create the tables
create_tables_script = """
CREATE TABLE IF NOT EXISTS Casualties_Dimension (
    CasualtiesID INTEGER PRIMARY KEY,
    Casualty_Class TEXT,
    Casualty_Severity TEXT,
    Casualty_Sex TEXT,
    Age_of_Casualty_1 INTEGER,
    Age_Group TEXT
);

CREATE TABLE IF NOT EXISTS Conditions_Dimension (
    ConditionsID INTEGER PRIMARY KEY,
    Lighting_Conditions TEXT,
    Weather_Conditions TEXT
);

CREATE TABLE IF NOT EXISTS Date_Dimension (
    DateID INTEGER PRIMARY KEY,
    Accident_Date TEXT,
    Time__24hr TEXT,
    Year INTEGER,
    Month INTEGER,
    MonthName TEXT,
    Day_of_Week TEXT,
    Is_Holiday BOOLEAN
);
```

```
Tables created successfully.
Tables in the database: [('Casualties_Dimension',), ('Conditions_Dimension',), ('Date_Dimensio
n',), ('Location_Dimension',), ('Road_Dimension',), ('Vehicles_Dimension',), ('Collisions_Fact',)]
```

- Schema Design:
  - The schema was designed using the Star Schema approach to facilitate Online Analytical Processing (OLAP).
  - The schema consists of one fact table and six dimension tables, each designed to hold specific attributes related to traffic collisions.

Fact Table: Collisions_Fact

Stores measurable data related to traffic collisions and links to dimension tables.

Columns:

- Reference_Number (Primary Key)
- Number_of_Vehicles
- LocationID (Foreign Key → Location_Dimension)
- DateID (Foreign Key → Date_Dimension)
- ConditionsID (Foreign Key → Conditions_Dimension)
- RoadID (Foreign Key → Road_Dimension)
- VehicleID (Foreign Key → Vehicles_Dimension)
- CasualtiesID (Foreign Key → Casualties_Dimension)

Dimension Tables

Casualties_Dimension: Describes the casualties involved in collisions.

Attributes:

- CasualtiesID (Primary Key)
- Casualty_Class
- Casualty_Severity
- Casualty_Sex
- Age_of_Casualty_1
- Age_Group

Conditions_Dimension: Captures environmental conditions during collisions.

Attributes:

- ConditionsID (Primary Key)
- Lighting_Conditions
- Weather_Conditions

Location_Dimension: Specifies geographic details of collision locations.

Attributes:

- LocationID (Primary Key)
- Grid_Ref_Easting
- Grid_Ref_Northing

Date_Dimension: Provides temporal details of collisions.

Attributes:

- DateID (Primary Key)
- Accident_Date
- Time__24hr
- Year
- Month
- MonthName
- Day_of_Week
- Is_Holiday

Road_Dimension: Contains details about road classifications and conditions.

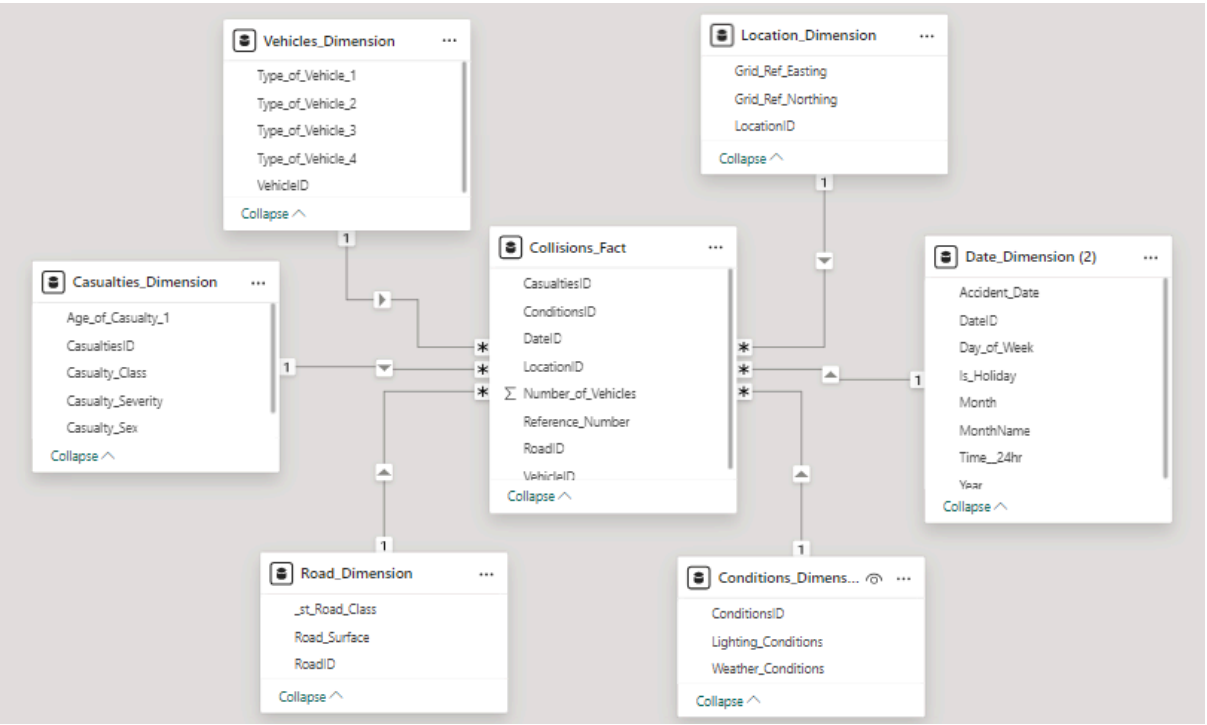
Attributes:

- RoadID (Primary Key)
- _st_Road_Class
- Road_Surface

Vehicles_Dimension: Describes the types of vehicles involved in collisions.

Attributes:

- VehicleID (Primary Key)
- Type_of_Vehicle_1
- Type_of_Vehicle_2
- Type_of_Vehicle_3
- Type_of_Vehicle_4



4. Table Creation:
    - SQL scripts were written and executed to create the fact and dimension tables in the SQLite database.
    - Primary keys and foreign key relationships were defined to ensure data integrity.

Create the Dimension Tables

[41]:
```
# Prepare Location Dimension
location_dimension = csv_data[['Grid Ref: Easting', 'Grid Ref: Northing']].drop_duplicates().reset_index(drop=True)
location_dimension.columns = ['Grid_Ref_Easting', 'Grid_Ref_Northing']  # Rename columns with underscores
location_dimension['LocationID'] = location_dimension.index + 1

# Save to CSV
location_dimension.to_csv('/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Location_dimension.csv', index=False)
print("Location Dimension saved to CSV successfully.")
```

```
Location Dimension saved to CSV successfully.
```

[42]:
```
# Prepare Conditions Dimension
conditions_dimension = csv_data[['Lighting Conditions', 'Weather Conditions']].drop_duplicates().reset_index(drop=True)
conditions_dimension.columns = ['Lighting_Conditions', 'Weather_Conditions']  # Rename columns with underscores
conditions_dimension['ConditionsID'] = conditions_dimension.index + 1

# Save to CSV
conditions_dimension.to_csv('/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Conditions_dimension.csv', index=False
print("Conditions Dimension saved to CSV successfully.")
```

```
Conditions Dimension saved to CSV successfully.
```

5. Data Population:
    - Data was imported from the cleaned CSV files into the respective tables using Python and the pandas library.
    - The data was successfully loaded into the database tables without errors or inconsistencies.

[68]:
```
import pandas as pd

# File paths for the CSV files
files = {
    "Casualties_Dimension": "/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Casualties_dimension.csv",
    "Conditions_Dimension": "/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Conditions_dimension.csv",
    "Date_Dimension": "/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Date_dimension.csv",
    "Location_Dimension": "/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Location_dimension.csv",
    "Road_Dimension": "/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Road_dimension.csv",
    "Vehicles_Dimension": "/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Vehicles_dimension.csv",
    "Collisions_Fact": "/Users/mac/Desktop/AccidentDataWarehouse/data/warehouse data/Collisions_Fact.csv"
}

# Connect to the SQLite database
conn = sqlite3.connect("accidents_dwh.sqlite")

# Load each CSV into its corresponding table
for table, file in files.items():
    data = pd.read_csv(file)
    data.to_sql(table, conn, if_exists="replace", index=False)
    print(f"Table {table} populated successfully!")

# Close the connection
conn.close()
```

```
Table Casualties_Dimension populated successfully!
Table Conditions_Dimension populated successfully!
Table Date_Dimension populated successfully!
Table Location_Dimension populated successfully!
Table Road_Dimension populated successfully!
Table Vehicles_Dimension populated successfully!
Table Collisions_Fact populated successfully!
```

6. Relationship Validation:
    - Queries were executed to validate the relationships between the fact table and dimension tables.
    - Examples of validation queries:
        - Verified that all LocationID values in Collisions_Fact exist in Location_Dimension.
        - Ensured that DateID, ConditionsID, RoadID, VehicleID, and CasualtiesID in Collisions_Fact map correctly to their corresponding dimension tables.
    - All relationships were confirmed to be consistent.

Tools and Technologies Used

- SQLite: A lightweight relational database for creating and managing the data warehouse.
- Python: Used for data transformation, table creation, and populating data.
    - Libraries: pandas, sqlite3.
- SQL Queries: Executed for validation and ensuring data consistency.

6. Data Visualization

The **Data Visualization** phase was an essential step in transforming the cleaned and structured data into actionable insights. Using Power BI as the primary tool, we designed a comprehensive dashboard to present key metrics, trends, and patterns effectively. This phase aimed to enhance the interpretability of the data, support decision-making, and provide an intuitive user interface for exploring critical information.

Objectives

The objectives of the Data Visualization phase were:

1. To provide an overview of the data through high-level metrics and trends.
2. To enable detailed analysis across temporal, geographical, environmental, and casualty-related dimensions.
3. To create an interactive dashboard that allows users to filter and drill down into specific insights.
4. To ensure all visualizations align with the project's analysis goals.

Dashboard Design

The dashboard was structured into the following key sections, each addressing a specific analysis goal:

1. **Overview Section**:
    - Displays high-level metrics such as total collisions, total casualties by severity (slight, serious, fatal), and the average number of vehicles involved in collisions.
    - Provides a snapshot of the dataset's key insights in a visually engaging manner.
2. **Environmental Section**:
    - Explores the impact of lighting, weather, and road surface conditions on collisions.
    - Uses stacked bar charts, pie charts, and column charts to analyze these factors.
3. **Casualty Section**:
    - Analyzes casualty severity distribution, demographics (age group, gender), and vehicle-casualty relationships.
    - Provides insights into the factors contributing to casualty severity through stacked bar charts and clustered column charts.
4. **Vehicle Section**:
    - Explores the types of vehicles involved in collisions and their frequency.
    - Analyzes multi-vehicle collisions and their relationship to casualty severity.
    - Examines the time of day when specific vehicle types are most involved in collisions.
5. **Combined Analysis**:
    - Explores relationships between multiple dimensions, such as:
        - Casualty severity by road class.
        - Casualty severity by weather conditions.
        - Collision patterns by location and time.

Visualizations

The dashboard includes a variety of visuals tailored to different insights, including:

- **Line Charts**: To show temporal trends, such as yearly or monthly collision counts.
- **Bar Charts**: To compare categories like casualty demographics and road conditions.
- **Pie Charts**: To display proportions, such as casualty severity distribution.
- **Stacked Charts**: To analyze relationships between multiple dimensions, such as road surface and casualty severity.

Interactivity : The dashboard was designed with user interactivity in mind filters and slicers.

Challenges and Mitigations

1. **Coordinate System Conversion**:
    - Initially, geographic data was in a local coordinate system and required conversion to latitude/longitude for map visualization. This was resolved using a Python-based transformation script.
2. **Handling Missing or Irregular Data**:
    - Adjustments were made to ensure consistent formatting and removal of incomplete records, ensuring accurate visualizations.
3. **Dashboard Layout**:
    - Balancing the number of visuals while maintaining clarity required iterative design and user feedback.

The Data Visualization phase successfully translated the data into meaningful and actionable insights through a combination of well-designed visuals and interactive elements. The Power BI dashboard not only meets the project's analysis goals but also provides a robust foundation for exploring and understanding collision data. This phase highlights the power of data visualization in bridging the gap between raw data and informed decision-making.

Conclusion

This project represents a comprehensive journey through the end-to-end data analytics pipeline, from initial data collection to the final delivery of actionable insights via an interactive dashboard. By leveraging modern tools such as Python for data preparation, SQLite for data storage, and Power BI for data visualization, the project successfully transformed raw collision data into meaningful and impactful knowledge.

Key Achievements

1. **Data Preparation and Cleaning**: The project tackled diverse challenges, including handling missing values, standardizing data formats, and ensuring the structural integrity of the datasets. This phase laid a strong foundation for the subsequent phases, ensuring the reliability and accuracy of the analysis.
2. **Data Warehouse Development**: Using a star schema design, the project established a robust data warehouse with fact and dimension tables that optimized querying and analytical efficiency. The use of SQLite provided an accessible yet powerful platform for implementing the data warehouse.
3. **Comprehensive Analysis**: The analysis plan explored multiple dimensions of the collision data, including temporal trends, geographic hotspots, environmental conditions, casualty patterns, and vehicle involvement. This multi-faceted approach ensured that the project addressed key questions and uncovered valuable insights.
4. **Interactive Dashboard**: The final dashboard in Power BI combined simplicity and depth, offering stakeholders a dynamic tool to explore the data. Through intuitive visuals and interactivity, the dashboard effectively communicated complex patterns and trends.