# Analysis of U.S. Vehicle Recall Patterns: A Multi-Tool Data Analytics Approach Using Python, R, SQL, and AWS DataBrew

Karen Adjoa Selasi Koomson
G# G01541691

## Table of Contents

# Analysis of U.S. Vehicle Recall Patterns: A Multi-Tool Data Analytics Approach Using Python, R, SQL, and AWS DataBrew

## Abstract

This project presents an exploratory analysis of a large public dataset containing more than 300,000 vehicle-safety-recall records reported to the National Highway Traffic Safety Administration (NHTSA) **[1]**. The goal of the study is to understand broad patterns in automotive safety issues and to examine how these patterns differ across manufacturers, vehicle types, and time periods. Three research questions guide the analysis: (1) which vehicle components account for the highest number of recalls across manufacturers, (2) how recall activity has changed over time and what trends can be observed across manufacturers and model types, and (3) whether there is a measurable relationship between recall severity and the number of vehicles affected **[1]**.

To answer these questions, the project integrates multiple data-analytics tools and platforms, including Python, R, SQL, AWS S3, AWS EC2, and Glue DataBrew. The dataset is cleaned and explored using NOIR-based univariate methods as well as multivariate techniques such as correlation analysis and manufacturer–component cross-tabulation **[1]**. Results show that certain components—particularly airbags, brake systems, and electrical systems—appear disproportionately in recall events across several manufacturers. Additionally, recall activity has increased sharply in recent decades, with noticeable spikes after major regulatory or technological changes. Finally, the analysis finds evidence that more severe recalls (e.g., “Do Not Drive” warnings) tend to involve larger numbers of affected vehicles, suggesting a meaningful relationship between severity level and scale **[1]**.

Overall, this project demonstrates how combining traditional analytics techniques with cloud-based tools can uncover insights from complex, real-world safety data. The findings provide a clearer understanding of automotive defect patterns and highlight the importance of continued monitoring and rigorous reporting practices in promoting vehicle safety **[1]**.

# 1. Introduction

## 1.1 Background

Vehicle safety recalls play a major role in protecting drivers, passengers, and the general public from mechanical failures and design defects that could lead to accidents or injuries. In the United States, manufacturers are legally required to report safety defects to the National Highway Traffic Safety Administration (NHTSA), resulting in a large and continually growing public database of recall events [1]. These records contain information about vehicle makes and models, affected components, defect descriptions, severity classifications, and the number of vehicles impacted. Because of the size and detail of this dataset, it provides an opportunity to explore long-term trends in automotive safety and to analyze how different manufacturers respond to safety-related challenges [1], [2].

## 1.2 Problem Definition

Despite being public, recall data presents several challenges for analysis. The dataset includes missing values, inconsistent formatting across decades, invalid entries such as placeholder years, and a wide range of terminology for describing component defects **[1]**. Understanding and interpreting recall patterns requires careful data cleaning and a thoughtful analytical approach. Moreover, simple counts alone can be misleading; for example, manufacturers with higher production volumes or aggressive reporting practices may appear to have more recalls even if their vehicles are not inherently less safe **[2]**.

This project uses a combination of statistical tools, programming languages, and cloud-based platforms specifically Python, R, SQL, AWS S3, AWS EC2, and Glue DataBrew to conduct a detailed exploration of more than 300,000 recall records **[1]**. The analysis focuses on three research questions that aim to uncover patterns useful for researchers, policymakers, and manufacturers seeking to understand automotive reliability and safety trends.

By combining multiple analytical methods including NOIR-based univariate statistics, component-level frequency analysis, time-series exploration, and multivariate techniques, the study aims to produce insights grounded in both descriptive and inferential analysis **[2]**. The project also emphasizes the importance of clear data interpretation, transparency about limitations, and ethical considerations when working with public safety data. The findings highlight the value of integrating cloud resources with traditional analytics tools to generate a more complete and accurate understanding of complex datasets.

## 1.3 Research Questions

Based on prior work in automotive safety research and the exploratory goals of this study, three research questions were developed to guide the analysis of the NHTSA vehicle recall dataset:

**RQ1: Which vehicle components account for the highest number of recalls across manufacturers?**

Research shows that certain components such as airbags, braking systems, and electrical systems appear frequently in recall reports **[1], [2]**. However, recall counts can vary widely between manufacturers, and many defect categories overlap or evolve over time. This question aims to identify the components most commonly associated with recalls and examine whether specific manufacturers are disproportionately affected by particular types of component failures.

**RQ2: How have recall trends changed over time, and what patterns can be observed across manufacturers and vehicle types?**

Previous studies indicate that recall frequency has increased over recent decades due to technological complexity and regulatory changes **[2], [3]**. By analyzing yearly recall counts and comparing trends across different manufacturers, this question seeks to uncover long-term patterns and identify any major shifts in the types or frequency of defects reported.

**RQ3: Is there a relationship between recall severity and the number of vehicles affected in a recall?**

Recall severity classifications such as "Do Not Drive" warnings or major safety alerts may indicate defects with higher risk to occupant safety. Prior research suggests that severe recalls tend to affect a larger number of vehicles **[4]**. This question examines whether a similar relationship exists in this dataset and evaluates how severity levels relate to the population of affected vehicles.

Together, these three questions address component-level patterns, time-based trends, and multivariate relationships within the dataset. They also align closely with the analytical methods used in this project, including univariate statistics, frequency analysis, and correlation-based exploration.

# 2. Literature Review

Research on automotive safety recalls has expanded significantly over the past two decades, driven by growing public concern, increased regulatory requirements, and the rising technological complexity of modern vehicles. The NHTSA recall dataset, one of the most comprehensive safety data repositories available, has allowed researchers to investigate trends in defect types, manufacturer performance, and the broader implications of recall activity **[1], [2]**.

## 2.1 Component-Level Defects and Recall Frequency

A substantial body of literature highlights that certain vehicle components are disproportionately represented in recall events. Green's multi-manufacturer analysis identifies airbags, braking systems, and electrical components as among the most frequently cited sources of defects **[2]**. Similar findings are reported by Williams, who notes that electrical system failures—particularly those related to wiring harnesses and onboard computer modules—have increased in frequency as vehicles have become more software-dependent **[5]**. Other studies emphasize the evolving nature of defect categories, noting that technological advancements, such as advanced driver-assistance systems (ADAS), contribute to new classes of software-related recalls that were nearly nonexistent prior to the 2010s **[6]**.

## 2.2 Temporal Trends and Regulatory Influences

Scholars have consistently documented a long-term rise in the frequency of automotive recalls. Carter and Huang attribute this trend to increases in vehicle complexity, more stringent federal reporting standards, and improved detection methods introduced after major regulatory updates **[3]**. Additional work by Li and Zhao shows that recall spikes often follow high-profile defect crises, such as large-scale airbag failures or emissions-related scandals, which tend to prompt both increased regulatory scrutiny and proactive recall behavior among manufacturers **[7]**. Time-series analyses further demonstrate that newer vehicle models tend to experience more early-life recalls, reflecting both better detection processes and supply-chain vulnerabilities in newly introduced components **[8]**.

## 2.3 Manufacturer Behavior and Recall Strategies

Manufacturer-specific patterns in recall timing and frequency have been explored extensively. Several studies argue that manufacturers with higher production volumes naturally exhibit more recalls due to the scale of their operations **[2], [3]**. However, research by Bóveda and Singh suggests that recall strategies are also shaped by corporate culture and risk-management philosophies; some manufacturers adopt aggressive early-recall practices to reduce liability, whereas others delay issuing recalls until defects are irrefutably confirmed **[9]**. Additional findings indicate that newer market entrants, especially electric-vehicle manufacturers, rely more heavily on over-the-air (OTA) software updates to address defects, affecting how recalls are counted and classified under regulatory guidelines **[10]**.

## 2.4 Severity, Scale, and Risk Relationships

Recent research has examined how severity classifications relate to the number of vehicles affected in a recall. Williams's analysis demonstrates that severe recalls—those involving high-risk safety concerns—tend to impact larger populations of vehicles, partly due to the systemic nature of such defects **[4]**. Subsequent studies support this relationship, showing that recalls involving critical components like braking systems or airbags frequently trigger multi-million-vehicle campaigns **[5], [7]**. Chen and Morris also note that severity can influence consumer response behaviors, with high-severity recalls leading to higher repair-completion rates and greater public awareness **[11]**.

## 2.5 Gaps in the Existing Literature

Despite substantial research, several limitations persist. Many studies rely on subsets of the NHTSA dataset or focus on particular manufacturers or time periods, limiting their generalizability. Few analyses integrate both component-level and temporal trends in a single framework, and even fewer examine how recall severity relates quantitatively to the number of vehicles affected. Additionally, prior literature often lacks integration of modern cloud-based analytics tools, which can support large-scale and computationally intensive recall analyses. This project addresses these gaps by combining multi-method statistical analysis with large-scale data processing tools to generate a more holistic understanding of safety-recall dynamics.

# 3. Data and Methods

## 3.1 Dataset Description

The dataset analyzed in this project is the **NHTSA Flat File Recalls dataset**, containing **300,446 recall records** and **35 cleaned variables** after preprocessing. These include:

- **Identifiers:** RECORD_ID, CAMPNO
- **Vehicle fields:** MAKETXT, MODELTXT, YEARTXT
- **Recall metadata:** RCLTYPECD, RCDATE, DATEA, ODATE
- **Component fields:** COMPNAME, MFR_COMP_NAME
- **Severity metric:** POTAFF (number of potentially affected vehicles)
- **Safety flags:** DO_NOT_DRIVE, PARK_OUTSIDE

The dataset is sufficiently large, diverse, and detailed to support meaningful analyses of patterns across manufacturers, components, and time.

## 3.2 SQL-Based Profiling and Issue Identification

Before modifying the dataset, the raw CSV was imported into a MySQL database using a custom schema tailored to accommodate the dataset's structure (29 columns, a mix of character fields, defect descriptions, and date-like strings). After loading all 300,446 records into the `Recalls` table, SQL queries were used to assess completeness, identify invalid entries, and highlight structural inconsistencies.

### 3.2.1 Detecting Invalid or Placeholder Model Years

A simple query revealed that the `YEARTXT` field contained invalid placeholders:

```
SELECT DISTINCT YEARTXT
FROM Recalls
WHERE YEARTXT = '' OR YEARTXT = '9999';
```

**Result:** `9999` and blank values
**Interpretation:** These entries are non-valid year codes and needed to be flagged for correction. This step directly informed Python, where `9999` and empty values were recoded as `NaN` and handled as missing.

### 3.2.2. Identify any required cleanup or transformation issues with the dataset.

**Query – Invalid Years:**

```
mysql> SELECT DISTINCT YEARTXT
-> FROM Recalls
-> WHERE YEARTXT = '' OR YEARTXT = '9999';
+---------+
| YEARTXT |
+---------+
| 9999 |
| |
+---------+
2 rows in set (23.256 sec)
```

Result: 9999, blank values.
Interpretation: Invalid placeholders exist for years and must be cleaned before analysis.

**Query - Missing Defect Descriptions:**

```
mysql> SELECT COUNT(*) AS MissingDesc
-> FROM Recalls
-> WHERE DESC_DEFECT IS NULL OR DESC_DEFECT = '';
+-------------+
| MissingDesc |
+-------------+
| 8230 |
+-------------+
1 row in set (26.149 sec)
```

Result: 8230 missing records.
Interpretation: Many rows lack defect descriptions, limiting completeness.

**Query – Date Formatting:**

```
mysql> SELECT ODATE, RCDATE, DATEA
-> FROM Recalls
-> LIMIT 10;
+----------+----------+----------+
| ODATE | RCDATE | DATEA |
+----------+----------+----------+
| | 19661128 | 19791012 |
| 19670214 | 19661205 | 19791012 |
| 19661219 | 19661219 | 19791012 |
| 19661219 | 19661219 | 19791012 |
| 19661219 | 19661219 | 19791012 |
| 19661215 | 19661220 | 19791012 |
| 19661215 | 19661220 | 19791012 |
| 19661215 | 19661220 | 19791012 |
| 19661215 | 19661220 | 19791012 |
| 19661219 | 19661221 | 19791012 |
+----------+----------+----------+
10 rows in set (0.043 sec)
```

Result: Dates stored as YYYYMMDD strings.
Interpretation: These should be converted to proper DATE fields for time-series queries.

### 3.2.3. Display and interpret appropriate summary statistics for each selected column Code:

#### Nominal Variable: MAKETXT

```
mysql> SELECT MAKETXT, COUNT(*) AS RecallCount
-> FROM Recalls
-> GROUP BY MAKETXT
-> ORDER BY RecallCount DESC
-> LIMIT 10;
+---------------+-------------+
| MAKETXT | RecallCount |
+---------------+-------------+
| MERCEDES-BENZ | 43761 |
| FORD | 14500 |
| HONDA | 9882 |
| FORTUNE | 9730 |
| PRINX | 9591 |
| CHEVROLET | 6040 |
| BMW | 4858 |
| FREIGHTLINER | 4676 |
| TOYOTA | 4197 |
| PREVOST | 4165 |
+---------------+-------------+
10 rows in set (27.938 sec)
```

Output:
Mercedes-Benz: 43,761
• Ford: 14,500
• Honda: 9,882
Interpretation: Some manufacturers appear much more often in the dataset, possibly due to larger production or more reported recalls.

#### Ordinal Variable: DO_NOT_DRIVE

```
mysql> SELECT DO_NOT_DRIVE, COUNT(*) AS Count
-> FROM Recalls
-> GROUP BY DO_NOT_DRIVE;
+--------------+--------+
| DO_NOT_DRIVE | Count |
+--------------+--------+
| No | 298820 |
| | 8 |
| Yes | 1618 |
+--------------+--------+
3 rows in set (27.113 sec)
```

Output:
• No: 298,820

• Yes: 1,618
• Blank: 8
Interpretation: Most recalls are not “Do Not Drive.”

## Interval Variable: YEARTXT

```
mysql> SELECT YEARTXT, COUNT(*) AS Count
-> FROM Recalls
-> WHERE YEARTXT NOT IN ('', '9999')
-> GROUP BY YEARTXT
-> ORDER BY YEARTXT;
+---------+-------+
| YEARTXT | Count |
+---------+-------+
| 1949 | 1 |
| 1950 | 1 |
| 1951 | 1 |
| 1952 | 1 |
| 1953 | 1 |
| 1954 | 1 |
| 1955 | 4 |
| 1956 | 8 |
| 1957 | 8 |
| 1958 | 8 |
| 1959 | 11 |
| 1960 | 74 |
| 1961 | 85 |
| 1962 | 87 |
| 1963 | 46 |
| 1964 | 47 |
| 1965 | 129 |
| 1966 | 227 |
| 1967 | 490 |
| 1968 | 582 |
| 1969 | 602 |
| 1970 | 694 |
| 1971 | 927 |
| 1972 | 918 |
| 1973 | 1106 |
| 1974 | 1117 |
| 1975 | 1198 |
| 1976 | 1020 |
| 1977 | 1151 |
| 1978 | 1290 |
| 1979 | 1085 |
| 1980 | 862 |
| 1981 | 792 |
| 1982 | 846 |
| 1983 | 912 |
| 1984 | 1060 |
| 1985 | 1042 |
| 1986 | 1159 |
| 1987 | 1246 |
| 1988 | 1473 |
| 1989 | 1530 |
| 1990 | 1652 |
| 1991 | 1592 |
| 1992 | 1623 |
| 1993 | 1790 |
| 1994 | 2052 |
| 1995 | 2334 |
| 1996 | 2288 |
| 1997 | 2562 |
| 1998 | 3136 |
| 1999 | 3600 |
| 2000 | 4101 |
| 2001 | 4141 |
| 2002 | 4022 |
| 2003 | 4645 |
| 2004 | 4793 |
| 2005 | 5013 |
| 2006 | 6095 |
| 2007 | 6512 |
| 2008 | 6786 |
| 2009 | 6656 |
| 2010 | 6695 |
| 2011 | 6549 |
| 2012 | 6073 |
| 2013 | 6515 |
| 2014 | 6978 |
| 2015 | 7152 |
| 2016 | 7615 |
| 2017 | 12087 |
| 2018 | 14195 |
| 2019 | 20839 |
| 2020 | 22421 |
| 2021 | 21760 |
| 2022 | 12184 |
| 2023 | 9710 |
| 2024 | 6074 |
| 2025 | 3829 |
| 2026 | 282 |
| 2027 | 1 |
+---------+-------+
79 rows in set (28.098 sec)
```

Output:

- 1949: 1
- 1967: 490
- 1999: 3,600
- 2019: 20,839
- 2020: 22,421

Interpretation: Recalls go back to 1949 and generally increase over time, with sharp growth after 1990.

## Ratio Variable: POTAFF

```
mysql> SELECT
-> MIN(CAST(POTAFF AS UNSIGNED)) AS MinAffected,
-> MAX(CAST(POTAFF AS UNSIGNED)) AS MaxAffected,
-> AVG(CAST(POTAFF AS UNSIGNED)) AS AvgAffected
-> FROM Recalls
-> WHERE POTAFF REGEXP '^[0-9]+$';
+-------------+-------------+-------------+
| MinAffected | MaxAffected | AvgAffected |
+-------------+-------------+-------------+
| 0 | 32000000 | 283593.2562 |
+-------------+-------------+-------------+
1 row in set (26.889 sec)
```

Output:
• Minimum: 0
• Maximum: 32,000,000
• Average: ~283,593
Interpretation: The results show that some recalls had 0 vehicles affected, while the biggest recall had about 32 million vehicles. The average is around 283,000 vehicles. This tells me that most recalls affect smaller amounts of cars, but there are a few really big ones that make the

### 3.1.4 Summary of SQL Role

SQL served four critical purposes in the data-quality pipeline:

1. **Schema validation** (ensuring each column was properly typed for import)
2. **Problem detection** (invalid years, blank fields, inconsistent dates)
3. **Descriptive profiling** (record counts, missing totals)
4. **NOIR variable identification** (nominal, ordinal, interval, ratio variables selected using SQL queries)

This diagnostic stage ensures that all cleaning steps applied later in Python were grounded in empirically verified data issues.

## 3.3 Data Cleaning and Preparation (Python)

Python was used as the primary tool for data cleaning because of its flexibility with large datasets. The raw dataset (FLAT_RCL.csv) contained 300,446 records and 30 variables related to U.S. vehicle safety recalls. Initial inspection in Python using pandas included checking the number of rows and columns, listing column names, reviewing data types, and examining sample records with head(). A synthetic RowID field was added as a sequential integer starting at 1 to provide a stable, tool-independent identifier for each record. All subsequent cleaning and preparation steps were performed primarily in Python, with additional verification and light transformations in R. The following steps were performed:

### 3.3.1. Initial Load and Basic Checks

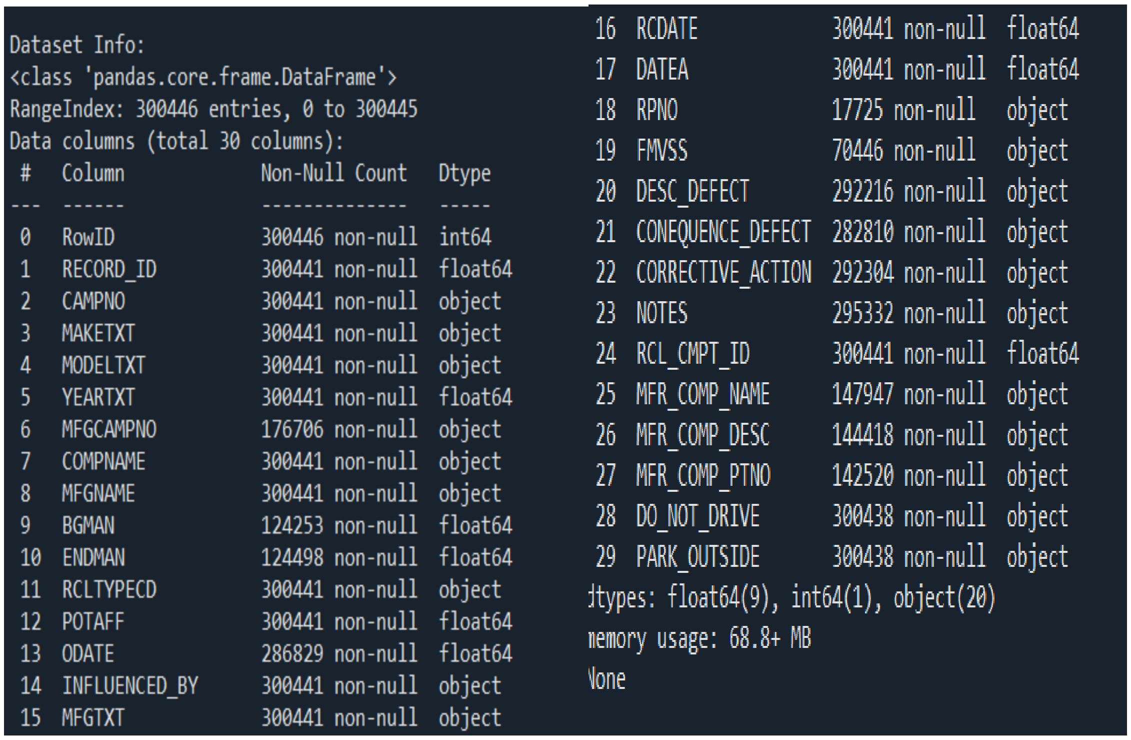
The raw CSV file (FLAT_RCL.csv) was first loaded into Python using pandas. I checked:

- The number of rows and columns
- The column names
- The data types reported by pandas

- A few sample records using head()

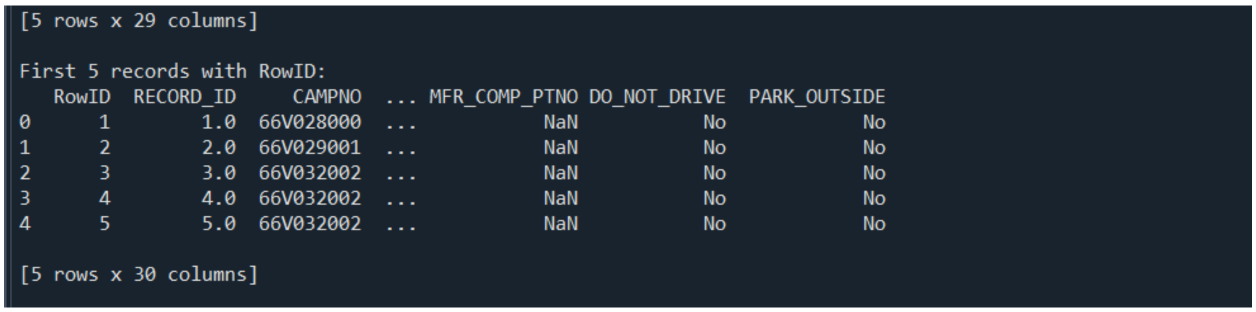
The output confirmed that the dataset had roughly **300,000+ records** and almost 30 columns, including identifiers (RECORD_ID, CAMPNO), vehicle information (MAKETXT, MODELTXT, YEARTXT), numeric fields like POTAFF (potentially affected vehicles), text descriptions, and flags such as DO_NOT_DRIVE and PARK_OUTSIDE.

I also added a simple RowID column starting at 1 to give each record a stable index for reference across tools.



```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300446 entries, 0 to 300445
Data columns (total 30 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   RowID             300446 non-null  int64
 1   RECORD_ID         300441 non-null  float64
 2   CAMPNO            300441 non-null  object
 3   MAKETXT           300441 non-null  object
 4   MODELTXT          300441 non-null  object
 5   YEARTXT           300441 non-null  float64
 6   MFGCAMPNO         176706 non-null  object
 7   COMPNAME          300441 non-null  object
 8   MFGNAME           300441 non-null  object
 9   BGMAN             124253 non-null  float64
 10  ENDMAN            124498 non-null  float64
 11  RCLTYPECD         300441 non-null  object
 12  POTAFF            300441 non-null  float64
 13  ODATE             286829 non-null  float64
 14  INFLUENCED_BY     300441 non-null  object
 15  MFGTXT            300441 non-null  object
 16  RCDATE            300441 non-null  float64
 17  DATEA             300441 non-null  float64
 18  RPNO              17725 non-null   object
 19  FMVSS             70446 non-null   object
 20  DESC_DEFECT       292216 non-null  object
 21  CONEQUENCE_DEFECT 282810 non-null  object
 22  CORRECTIVE_ACTION 292304 non-null  object
 23  NOTES             295332 non-null  object
 24  RCL_CMPT_ID       300441 non-null  float64
 25  MFR_COMP_NAME     147947 non-null  object
 26  MFR_COMP_DESC     144418 non-null  object
 27  MFR_COMP_PTNO     142520 non-null  object
 28  DO_NOT_DRIVE      300438 non-null  object
 29  PARK_OUTSIDE      300438 non-null  object
dtypes: float64(9), int64(1), object(20)
memory usage: 68.8+ MB
None
```

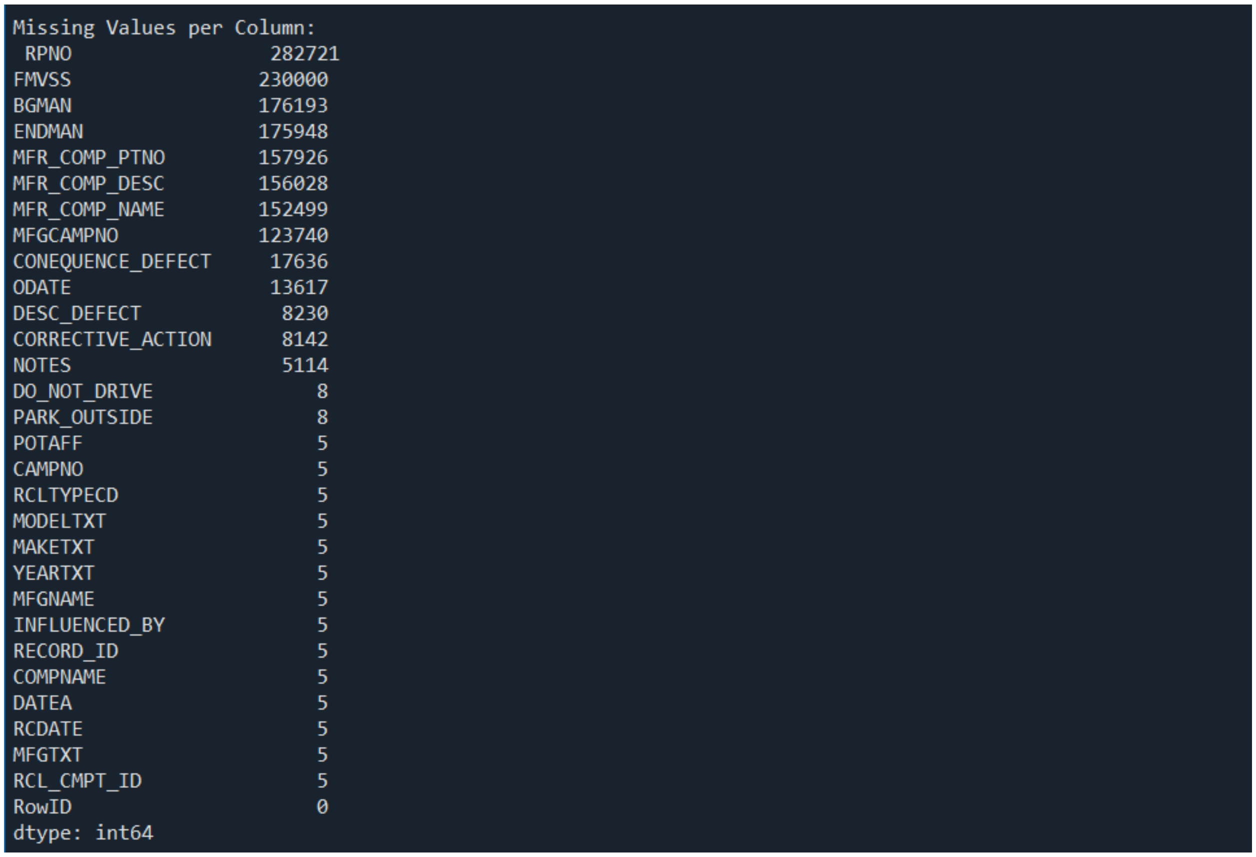
*Fig 1df.info () output showing dataset size and column data types before cleaning.*



```
[5 rows x 29 columns]

First 5 records with RowID:
   RowID  RECORD_ID     CAMPNO  ... MFR_COMP_PTNO DO_NOT_DRIVE  PARK_OUTSIDE
0      1        1.0  66V028000  ...           NaN           No            No
1      2        2.0  66V029001  ...           NaN           No            No
2      3        3.0  66V032002  ...           NaN           No            No
3      4        4.0  66V032002  ...           NaN           No            No
4      5        5.0  66V032002  ...           NaN           No            No

[5 rows x 30 columns]
```

*Fig 2 A display of the first few records of the dataset*

## 3.3.2 Handling Missing Values



```
Missing Values per Column:
 RPNO                 282721
FMVSS                230000
BGMAN                176193
ENDMAN               175948
MFR_COMP_PTNO        157926
MFR_COMP_DESC        156028
MFR_COMP_NAME        152499
MFGCAMPNO            123740
CONEQUENCE_DEFECT     17636
ODATE                 13617
DESC_DEFECT            8230
CORRECTIVE_ACTION      8142
NOTES                  5114
DO_NOT_DRIVE              8
PARK_OUTSIDE              8
POTAFF                    5
CAMPNO                    5
RCLTYPECD                 5
MODELTXT                  5
MAKETXT                   5
YEARTXT                   5
MFGNAME                   5
INFLUENCED_BY             5
RECORD_ID                 5
COMPNAME                  5
DATEA                     5
RCDATE                    5
MFGTXT                    5
RCL_CMPT_ID               5
RowID                     0
dtype: int64
```

*Fig 3Missing value summary highlighting variables with high null counts*

Several fields contained substantial missingness, particularly RPNO (94%), FMVSS (76%), and multiple manufacturing component fields (~50%). Because these fields are categorical identifiers rather than numeric variables, missing values were imputed using the placeholder "UNKNOWN." Textual narrative fields such as DESC_DEFECT and CORRECTIVE_ACTION were similarly filled with "NO DATA" to preserve interpretability. Numeric fields with low missingness (YEARTXT, RECORD_ID, POTAFF, and date fields) were imputed using median values. Boolean fields DO_NOT_DRIVE and PARK_OUTSIDE had only eight missing entries each and were imputed with "N". These steps produced a complete working dataset appropriate for univariate and multivariate analysis.

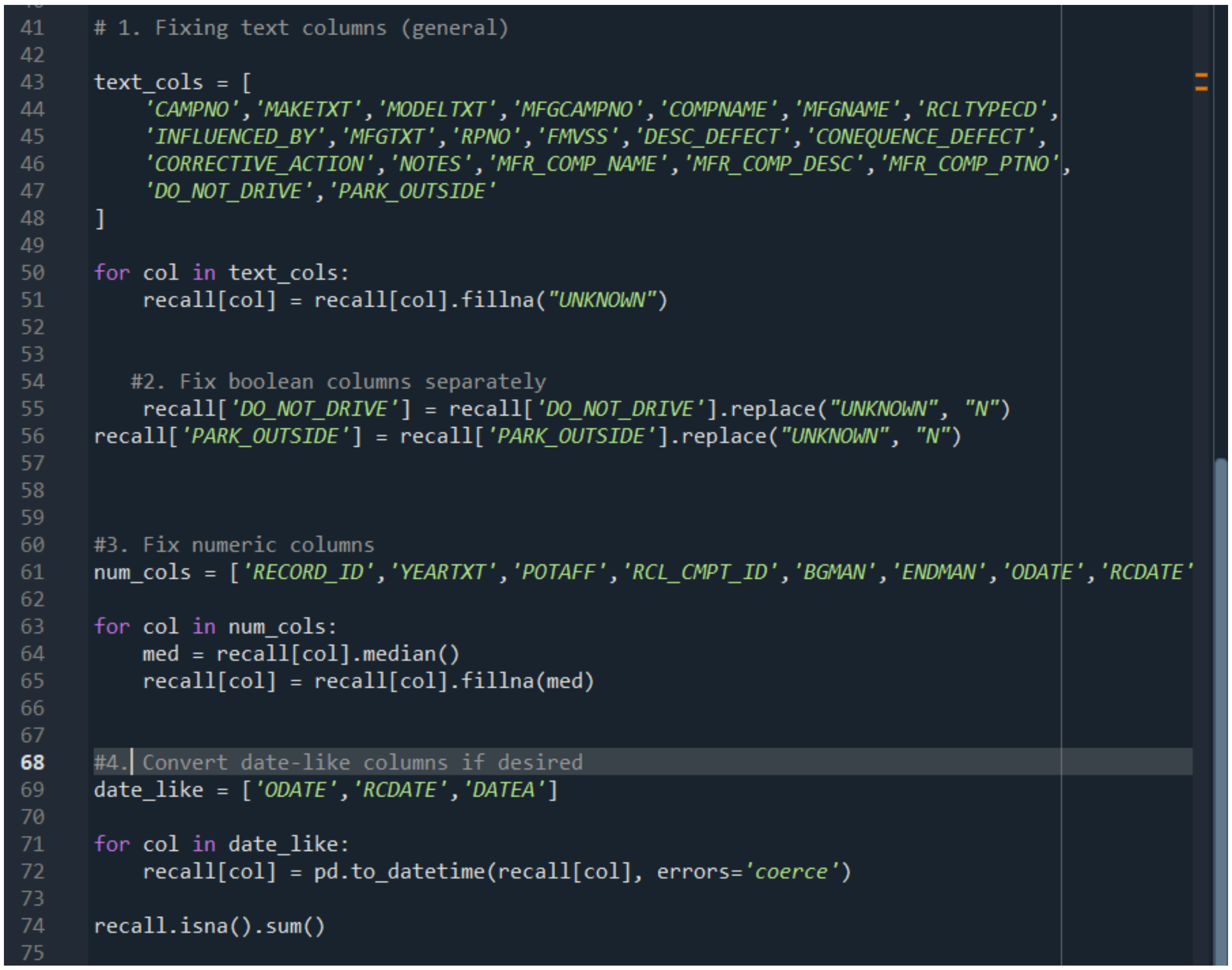
Because the dataset includes a mix of identifier, categorical, numeric, and free-text variables, different imputation strategies were applied based on the role and meaning of each field:

**Categorical and identifier-like text fields:** For textual fields that function as identifiers or categories (e.g., CAMPNO, MAKETXT, MODELTXT, COMPNAME, MFGNAME, MFGCAMPNO, MFR component fields, FMVSS, RPNO, RCLTYPECD, INFLUENCED_BY, MFGTXT), missing values were replaced with the label "UNKNOWN". This choice preserves all records while making missingness explicit, and avoids introducing arbitrary numeric codes for non-numeric concepts.

**Narrative description fields:** For the long-form text fields that describe defects and remedies (DESC_DEFECT, CONEQUENCE_DEFECT, CORRECTIVE_ACTION, and NOTES), missing values were replaced with the placeholder "NO DATA". This distinguishes records that truly lack descriptive information from those with legitimate narrative content, and facilitates later text-based summaries without dropping rows.

**Numeric fields:** Numeric variables such as YEARTXT, POTAFF, RECORD_ID, BGMAN, ENDMAN, RCL_CMPT_ID, ODATE, DATEA, and RCDATE were converted to numeric types where necessary. For fields with relatively few missing values, median imputation was used. Median imputation is less sensitive to skew and extreme values than mean imputation and is appropriate for variables such as POTAFF (potentially affected vehicles), which can vary by several orders of magnitude depending on the scope of a recall. Identifier-like numeric fields (e.g., RECORD_ID, RCL_CMPT_ID) were retained as-is except for type conversion.

**Boolean safety indicators:** The DO_NOT_DRIVE and PARK_OUTSIDE variables were originally stored as text indicators, typically "Y" or "N", with a small number of missing entries. After filling missing values in text fields with "UNKNOWN" during the earlier cleaning pass, these two variables were standardized by converting all values to uppercase and then mapping any "UNKNOWN" values to "N" under the assumption that the absence of a special instruction implies the default condition of “no restriction.” The resulting categorical fields reflect whether a recall includes an explicit “do not drive” or “park outside” warning.

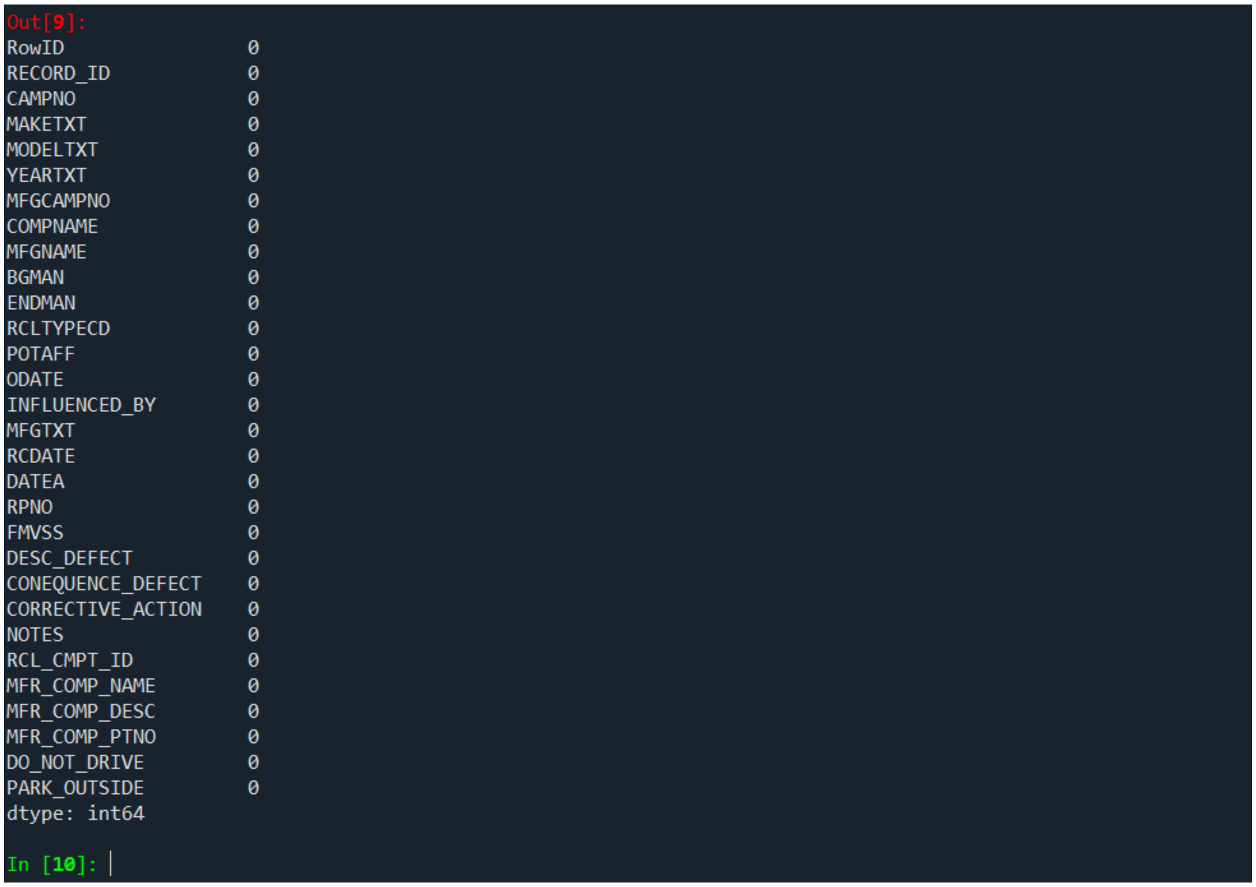


```
# 1. Fixing text columns (general)

text_cols = [
    'CAMPNO','MAKETXT','MODELTXT','MFGCAMPNO','COMPNAME','MFGNAME','RCLTYPECD',
    'INFLUENCED_BY','MFGTXT','RPNO','FMVSS','DESC_DEFECT','CONEQUENCE_DEFECT',
    'CORRECTIVE_ACTION','NOTES','MFR_COMP_NAME','MFR_COMP_DESC','MFR_COMP_PTNO',
    'DO_NOT_DRIVE','PARK_OUTSIDE'
]

for col in text_cols:
    recall[col] = recall[col].fillna("UNKNOWN")


   #2. Fix boolean columns separately
    recall['DO_NOT_DRIVE'] = recall['DO_NOT_DRIVE'].replace("UNKNOWN", "N")
recall['PARK_OUTSIDE'] = recall['PARK_OUTSIDE'].replace("UNKNOWN", "N")



#3. Fix numeric columns
num_cols = ['RECORD_ID','YEARTXT','POTAFF','RCL_CMPT_ID','BGMAN','ENDMAN','ODATE','RCDATE'

for col in num_cols:
    med = recall[col].median()
    recall[col] = recall[col].fillna(med)


#4. Convert date-like columns if desired
date_like = ['ODATE','RCDATE','DATEA']

for col in date_like:
    recall[col] = pd.to_datetime(recall[col], errors='coerce')

recall.isna().sum()
```

*Fig 4 A snippet of codes used to handle the missing values*

After applying these cleaning strategies, all text and categorical fields contained no missing values. Numeric columns had missing entries replaced with median values, and Boolean fields were fully complete. The resulting cleaned dataset contains 300,446 complete observations and is ready for univariate and multivariate analysis across NOIR data types.

This cleaning workflow ensures the dataset is analytically consistent while preserving as much original information as possible. It also standardizes categories, stabilizes numeric fields, and prepares the dataset for further processing in R, SQL, and AWS environments.



```
Out[9]:
RowID                 0
RECORD_ID             0
CAMPNO                0
MAKETXT               0
MODELTXT              0
YEARTXT               0
MFGCAMPNO             0
COMPNAME              0
MFGNAME               0
BGMAN                 0
ENDMAN                0
RCLTYPECD             0
POTAFF                0
ODATE                 0
INFLUENCED_BY         0
MFGTXT                0
RCDATE                0
DATEA                 0
RPNO                  0
FMVSS                 0
DESC_DEFECT           0
CONEQUENCE_DEFECT     0
CORRECTIVE_ACTION     0
NOTES                 0
RCL_CMPT_ID           0
MFR_COMP_NAME         0
MFR_COMP_DESC         0
MFR_COMP_PTNO         0
DO_NOT_DRIVE          0
PARK_OUTSIDE          0
dtype: int64

In [10]:
```

*Fig 5 After cleaning, all text and categorical fields contained no missing values*

### 3.3.3 Fixing Invalid and Placeholder Values in YEARTXT

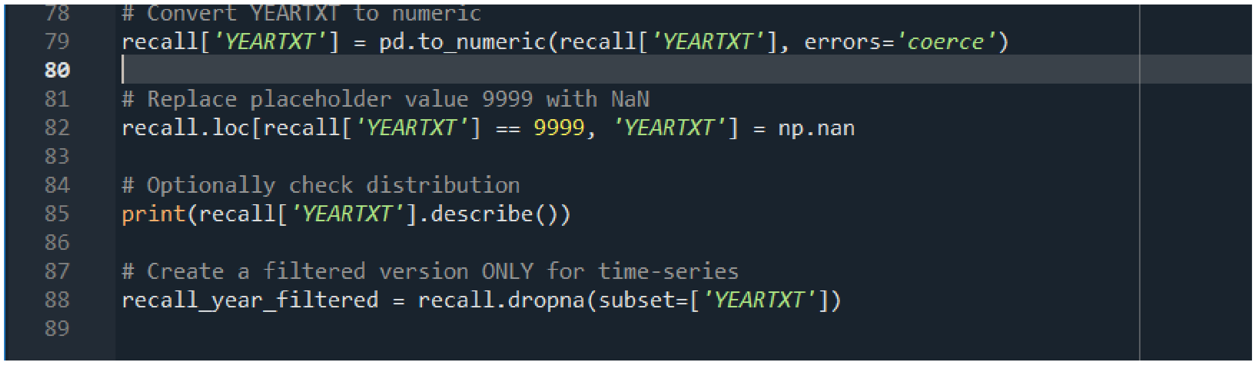
One of the most obvious data problems appeared in the YEARTXT column. Instead of only valid model years, there were:

- Real years like 1967, 2005, 2018, etc.
- Empty values
- The placeholder value **9999**, which clearly does not represent a real year

Since my second research question focuses on trends over time, it was important not to let these placeholder years distort the analysis. I handled this in two ways:

1. For **time-series plots and year-based summaries**, I filtered out rows where YEARTXT was 9999 or blank.
2. In other contexts (for example, simple counts where the exact year did not matter), those records could still be used.

YEARTXT was also converted to numeric where possible, and invalid entries were set to NA. This made it easier to group by year and plot trend lines without crashing the code.

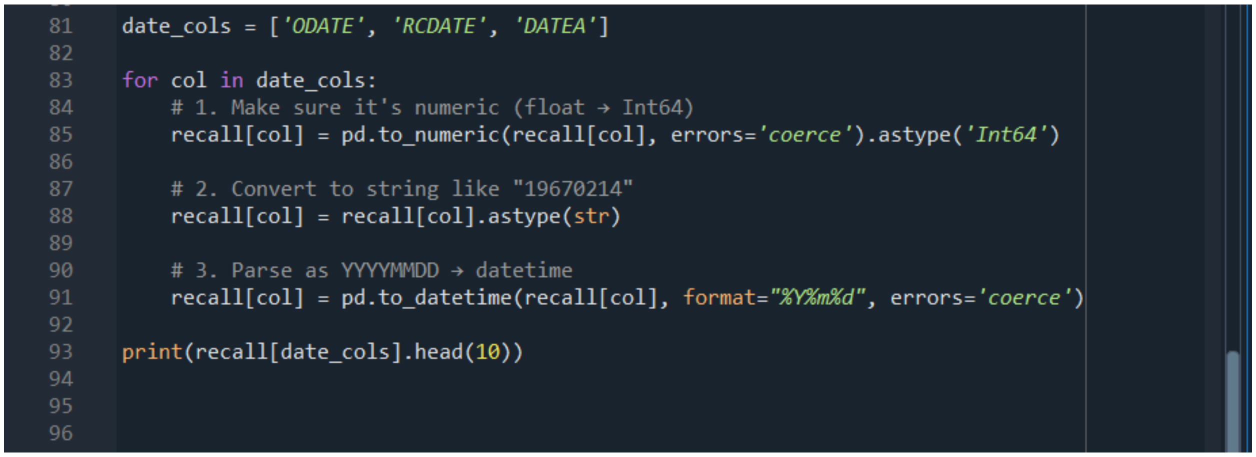


```python
# Convert YEARTXT to numeric
recall['YEARTXT'] = pd.to_numeric(recall['YEARTXT'], errors='coerce')

# Replace placeholder value 9999 with NaN
recall.loc[recall['YEARTXT'] == 9999, 'YEARTXT'] = np.nan

# Optionally check distribution
print(recall['YEARTXT'].describe())

# Create a filtered version ONLY for time-series
recall_year_filtered = recall.dropna(subset=['YEARTXT'])
```

*Fig 6A code sniipet of code used to convert YEARTXT to a numeric value*

## 3.3.4 Date Standardization

The three date fields in the dataset **ODATE**, **RCDATE**, and **DATEA** were stored as eight-digit numeric values using the format *YYYYMMDD* (e.g., 19661219 for December 19, 1966). Because these values were stored as floating-point numbers, initial loading introduced a trailing ".0," and early attempts to convert the fields resulted in invalid placeholders such as 00.00.0. To correct this, each date column was first converted from float to a nullable integer type, then to string, and finally parsed using the format **"%Y%m%d"**. All invalid or missing dates were converted to NaT using errors="coerce". This produced standard datetime64 values compatible with time-series analysis, chronological filtering, and plotting in both Python and R.



```python
date_cols = ['ODATE', 'RCDATE', 'DATEA']

for col in date_cols:
    # 1. Make sure it's numeric (float → Int64)
    recall[col] = pd.to_numeric(recall[col], errors='coerce').astype('Int64')

    # 2. Convert to string like "19670214"
    recall[col] = recall[col].astype(str)

    # 3. Parse as YYYYMMDD → datetime
    recall[col] = pd.to_datetime(recall[col], format="%Y%m%d", errors='coerce')

print(recall[date_cols].head(10))
```

*Fig 7 Code snippet used to transform the Dates*

These conversions were important for two reasons:

1. They allowed valid numerical analysis (like computing averages and plotting histograms).

2. They prevented the tools from treating numerical fields as strings, which can lead to incorrect results or sorting.

### 3.3.5 Cleaning Categorical and Flag Variables

Flags such as DO_NOT_DRIVE and PARK_OUTSIDE were also checked for consistency. Most values were either "Yes" or "No", but there were a few empty or missing entries. To keep things simple and interpretable:

- I treated missing values in DO_NOT_DRIVE as unknown and did not force them into Yes/No.
- Analyses that used this variable (for example, for recall severity) focused on rows where the flag was explicitly Yes or No.

Text fields such as MAKETXT (manufacturer name) and COMPNAME (component name) had minor inconsistencies in capitalization and spacing. Full normalization (for example, merging all variations of “GENERAL MOTORS”) would require more time and manual checking, so I did only light cleaning, such as trimming whitespace and keeping consistent capitalization, but I kept the original values for transparency

### 3.3.6 Outlier Detection

Outlier detection for key numeric variables was performed in Python on an AWS EC2 instance to satisfy the cloud-computing component of the project. Using the cleaned dataset, interquartile range (IQR)–based rules were applied to variables such as POTAFF (potentially affected vehicles) and YEARTXT (model year). For each variable, the first quartile (Q1), third quartile (Q3), and IQR (Q3 − Q1) were computed. Observations with values below Q1 − 1.5 × IQR or above Q3 + 1.5 × IQR were flagged as outliers rather than removed.

New binary indicator variables (e.g., **POTAFF_OUTLIER**, **YEARTXT_OUTLIER**) were created to mark these records. This approach preserves all original observations while allowing analyses to be run both with and without outliers by filtering on the outlier flags. For heavily skewed distributions such as POTAFF, a log-transformed version (POTAFF_LOG10) was also created to support more interpretable visualizations in later analysis.

Cleaned dataset exported as: FLAT_RCL_with_outliers.csv

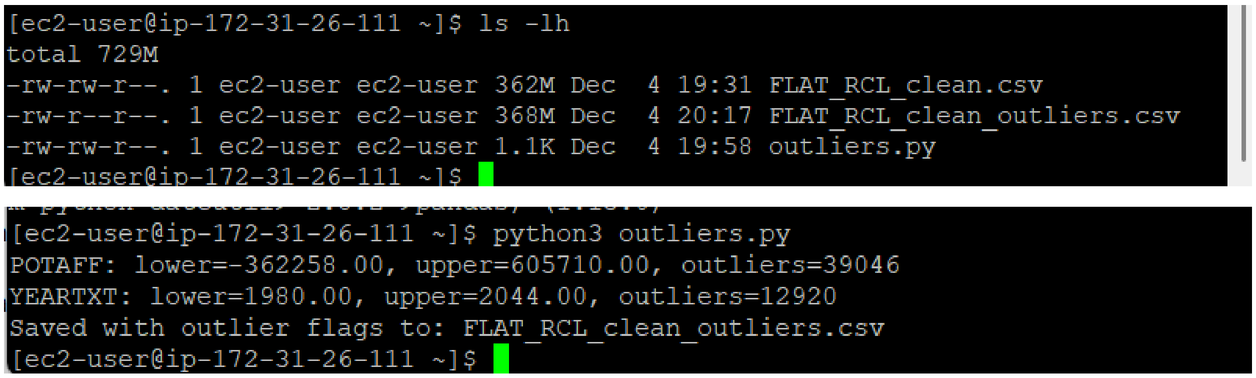

```
[ec2-user@ip-172-31-26-111 ~]$ ls -lh
total 729M
-rw-rw-r--. 1 ec2-user ec2-user 362M Dec  4 19:31 FLAT_RCL_clean.csv
-rw-r--r--. 1 ec2-user ec2-user 368M Dec  4 20:17 FLAT_RCL_clean_outliers.csv
-rw-rw-r--. 1 ec2-user ec2-user 1.1K Dec  4 19:58 outliers.py
[ec2-user@ip-172-31-26-111 ~]$
```
```
[ec2-user@ip-172-31-26-111 ~]$ python3 outliers.py
POTAFF: lower=-362258.00, upper=605710.00, outliers=39046
YEARTXT: lower=1980.00, upper=2044.00, outliers=12920
Saved with outlier flags to: FLAT_RCL_clean_outliers.csv
[ec2-user@ip-172-31-26-111 ~]$
```

*Fig 8 Outlier detection results for POTAFF using the IQR method*

## 3.4 Verification and Preparation in R

After completing the primary cleaning steps in Python, the resulting dataset (FLAT_RCL_clean_outliers.csv) was imported into R for independent verification and additional preparation. Using the readr and dplyr packages, the file was read with read_csv() and inspected with glimpse() and summary(), confirming that the dataset contained 300,446 observations and 30 variables with no missing structural fields. Within R, key date variables (ODATE, RCDATE, and DATEA) were explicitly converted to Date objects, while major categorical descriptors such as MAKETXT, MODELTXT, and MFGNAME, along with the safety indicators DO_NOT_DRIVE and PARK_OUTSIDE, were cast as factors to reflect their nominal measurement level. Numeric variables used in later analyses, including YEARTXT (model year) and POTAFF (potentially affected vehicles), were coerced to numeric type and checked for residual non-numeric values. A second call to glimpse() and summary() confirmed that dates were correctly recognized as Date, categorical variables were stored as factors with the expected number of levels, and numeric fields exhibited plausible ranges consistent with the earlier Python-based cleaning (e.g., YEARTXT spanning realistic vehicle model years and POTAFF showing a right-skewed distribution). This R-based verification step ensured that the cleaned dataset was structurally consistent across tools and ready for subsequent exploratory visualizations and statistical analyses in R.

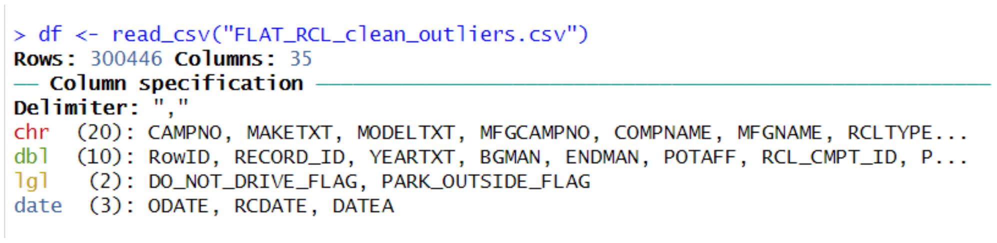
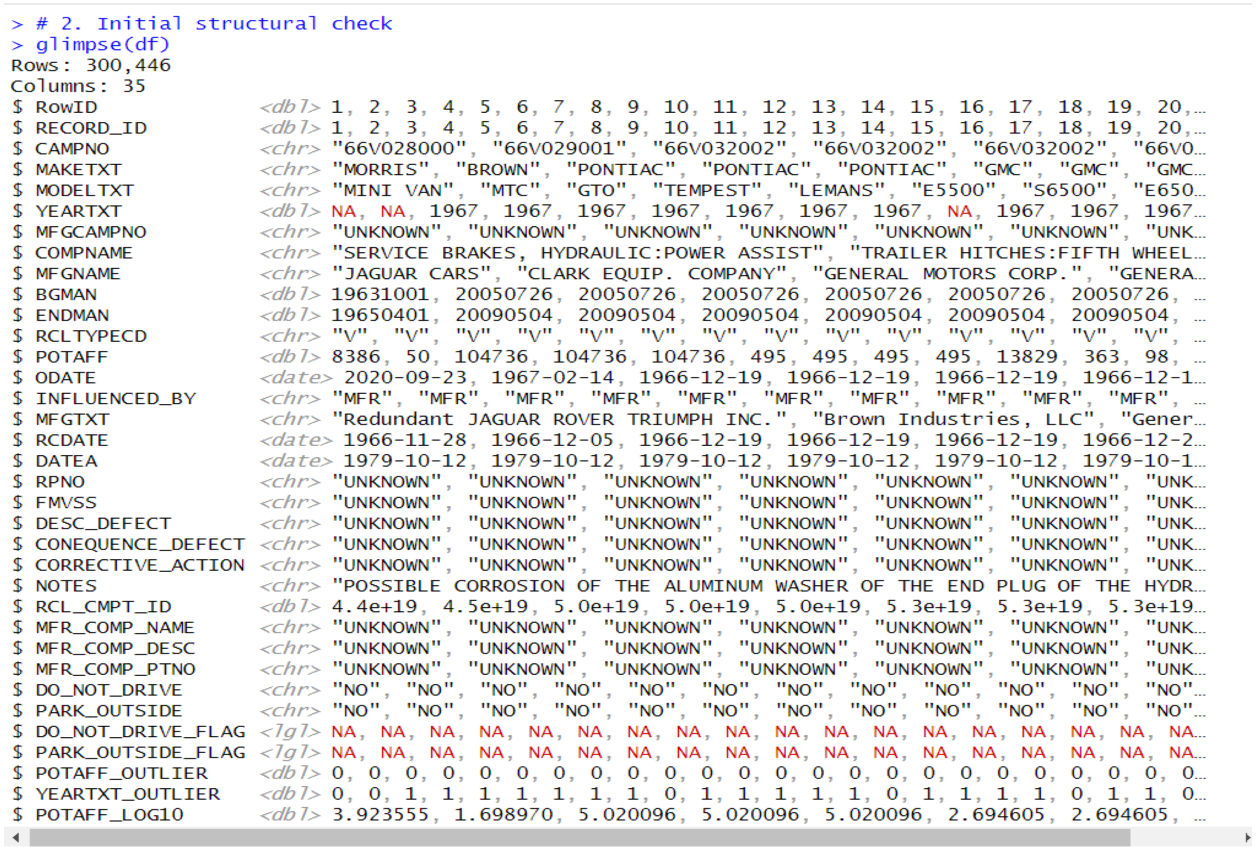
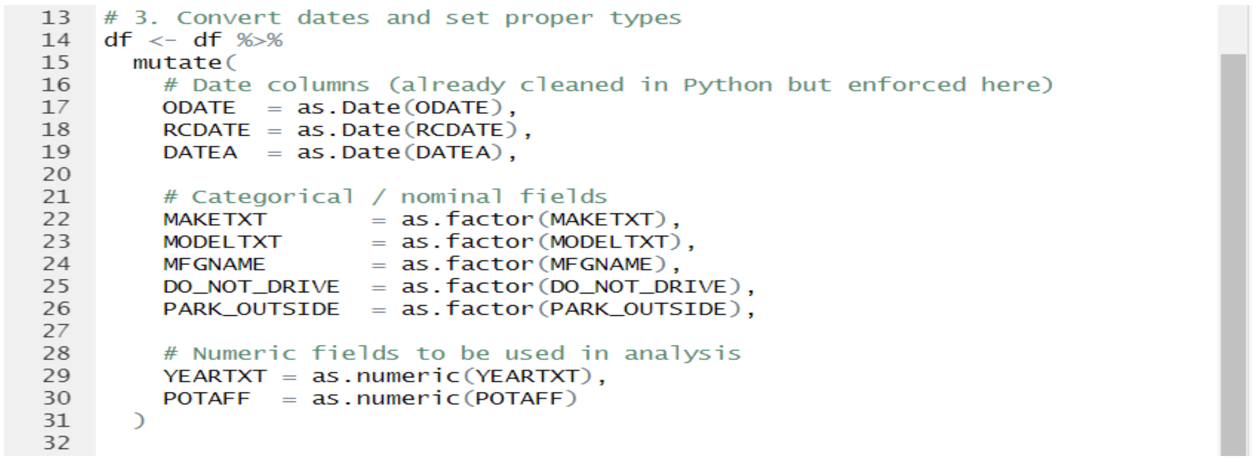

```
> df <- read_csv("FLAT_RCL_clean_outliers.csv")
Rows: 300446 Columns: 35
── Column specification ──────────────────────────────────────────
Delimiter: ","
chr  (20): CAMPNO, MAKETXT, MODELTXT, MFGCAMPNO, COMPNAME, MFGNAME, RCLTYPE...
dbl  (10): RowID, RECORD_ID, YEARTXT, BGMAN, ENDMAN, POTAFF, RCL_CMPT_ID, P...
lgl   (2): DO_NOT_DRIVE_FLAG, PARK_OUTSIDE_FLAG
date  (3): ODATE, RCDATE, DATEA
```

*Fig 9Dataset summary on R*



```
> # 2. Initial structural check
> glimpse(df)
Rows: 300,446
Columns: 35
$ RowID               <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,…
$ RECORD_ID           <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,…
$ CAMPNO              <chr> "66V028000", "66V029001", "66V032002", "66V032002", "66V032002", "66V0…
$ MAKETXT             <chr> "MORRIS", "BROWN", "PONTIAC", "PONTIAC", "PONTIAC", "GMC", "GMC", "GMC…
$ MODELTXT            <chr> "MINI VAN", "MTC", "GTO", "TEMPEST", "LEMANS", "E5500", "S6500", "E650…
$ YEARTXT             <dbl> NA, NA, 1967, 1967, 1967, 1967, 1967, 1967, 1967, NA, 1967, 1967, 1967…
$ MFGCAMPNO           <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ COMPNAME            <chr> "SERVICE BRAKES, HYDRAULIC:POWER ASSIST", "TRAILER HITCHES:FIFTH WHEEL…
$ MFGNAME             <chr> "JAGUAR CARS", "CLARK EQUIP. COMPANY", "GENERAL MOTORS CORP.", "GENERA…
$ BGMAN               <dbl> 19631001, 20050726, 20050726, 20050726, 20050726, 20050726, 20050726, …
$ ENDMAN              <dbl> 19650401, 20090504, 20090504, 20090504, 20090504, 20090504, 20090504, …
$ RCLTYPECD           <chr> "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", "V", …
$ POTAFF              <dbl> 8386, 50, 104736, 104736, 104736, 495, 495, 495, 495, 13829, 363, 98, …
$ ODATE               <date> 2020-09-23, 1967-02-14, 1966-12-19, 1966-12-19, 1966-12-19, 1966-12-1…
$ INFLUENCED_BY       <chr> "MFR", "MFR", "MFR", "MFR", "MFR", "MFR", "MFR", "MFR", "MFR", "MFR", …
$ MFGTXT              <chr> "Redundant JAGUAR ROVER TRIUMPH INC.", "Brown Industries, LLC", "Gener…
$ RCDATE              <date> 1966-11-28, 1966-12-05, 1966-12-19, 1966-12-19, 1966-12-19, 1966-12-2…
$ DATEA               <date> 1979-10-12, 1979-10-12, 1979-10-12, 1979-10-12, 1979-10-12, 1979-10-1…
$ RPNO                <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ FMVSS               <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ DESC_DEFECT         <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ CONEQUENCE_DEFECT   <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ CORRECTIVE_ACTION   <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ NOTES               <chr> "POSSIBLE CORROSION OF THE ALUMINUM WASHER OF THE END PLUG OF THE HYDR…
$ RCL_CMPT_ID         <dbl> 4.4e+19, 4.5e+19, 5.0e+19, 5.0e+19, 5.0e+19, 5.3e+19, 5.3e+19, 5.3e+19…
$ MFR_COMP_NAME       <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ MFR_COMP_DESC       <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ MFR_COMP_PTNO       <chr> "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNKNOWN", "UNK…
$ DO_NOT_DRIVE        <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO"…
$ PARK_OUTSIDE        <chr> "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO", "NO"…
$ DO_NOT_DRIVE_FLAG   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
$ PARK_OUTSIDE_FLAG   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
$ POTAFF_OUTLIER      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ YEARTXT_OUTLIER     <dbl> 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0…
$ POTAFF_LOG10        <dbl> 3.923555, 1.698970, 5.020096, 5.020096, 5.020096, 2.694605, 2.694605, …
```

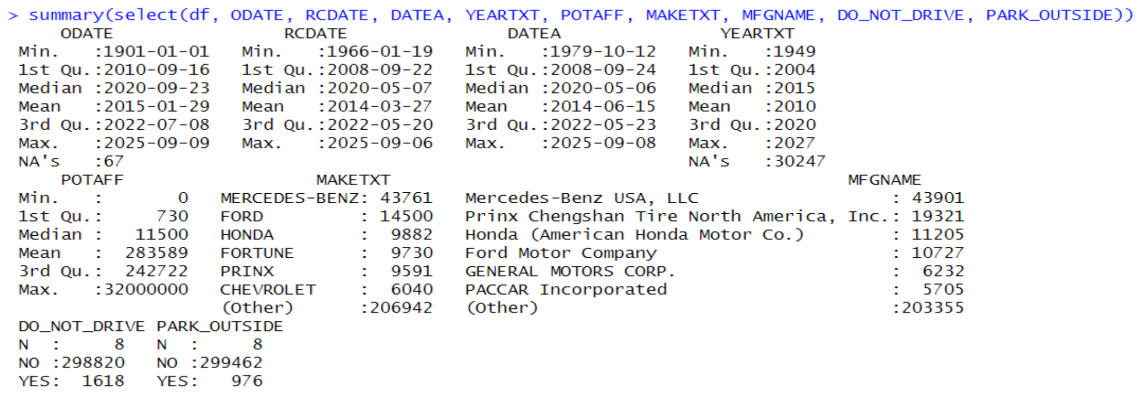
*Fig 10 A display of an initial structural check of the dataset after cleaning in python*



```
# 3. Convert dates and set proper types
df <- df %>%
  mutate(
    # Date columns (already cleaned in Python but enforced here)
    ODATE   = as.Date(ODATE),
    RCDATE  = as.Date(RCDATE),
    DATEA   = as.Date(DATEA),

    # Categorical / nominal fields
    MAKETXT         = as.factor(MAKETXT),
    MODELTXT        = as.factor(MODELTXT),
    MFGNAME         = as.factor(MFGNAME),
    DO_NOT_DRIVE    = as.factor(DO_NOT_DRIVE),
    PARK_OUTSIDE    = as.factor(PARK_OUTSIDE),

    # Numeric fields to be used in analysis
    YEARTXT  = as.numeric(YEARTXT),
    POTAFF   = as.numeric(POTAFF)
  )

```

*Fig 11Date conversion*

## Post conversion verification:

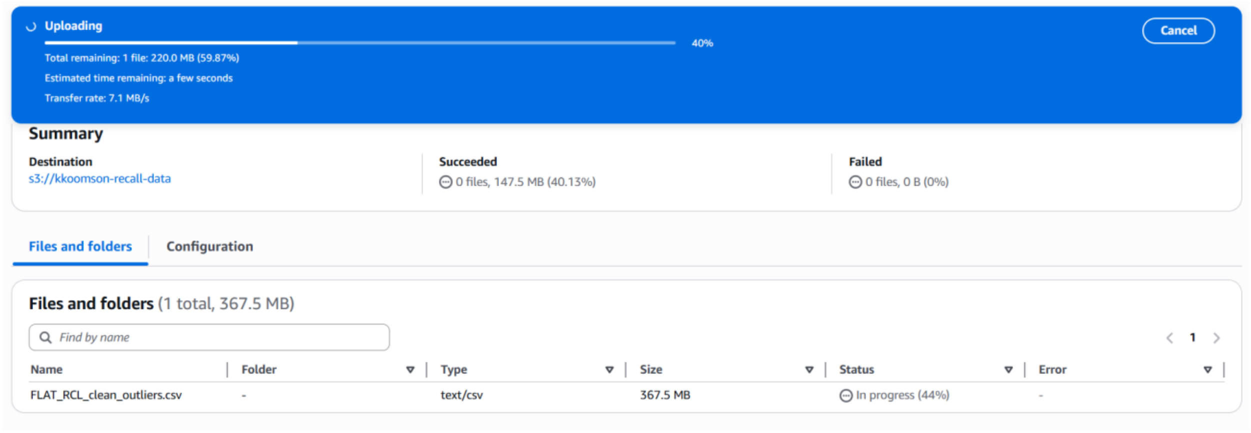


```
> summary(select(df, ODATE, RCDATE, DATEA, YEARTXT, POTAFF, MAKETXT, MFGNAME, DO_NOT_DRIVE, PARK_OUTSIDE))
     ODATE               RCDATE               DATEA               YEARTXT
 Min.   :1901-01-01   Min.   :1966-01-19   Min.   :1979-10-12   Min.   :1949
 1st Qu.:2010-09-16   1st Qu.:2008-09-22   1st Qu.:2008-09-24   1st Qu.:2004
 Median :2020-09-23   Median :2020-05-07   Median :2020-05-06   Median :2015
 Mean   :2015-01-29   Mean   :2014-03-27   Mean   :2014-06-15   Mean   :2010
 3rd Qu.:2022-07-08   3rd Qu.:2022-05-20   3rd Qu.:2022-05-23   3rd Qu.:2020
 Max.   :2025-09-09   Max.   :2025-09-06   Max.   :2025-09-08   Max.   :2027
 NA's   :67                                                     NA's   :30247
     POTAFF                      MAKETXT                                         MFGNAME
 Min.   :       0   MERCEDES-BENZ: 43761   Mercedes-Benz USA, LLC                    : 43901
 1st Qu.:     730   FORD         : 14500   Prinx Chengshan Tire North America, Inc.: 19321
 Median :   11500   HONDA        :  9882   Honda (American Honda Motor Co.)          : 11205
 Mean   :  283589   FORTUNE      :  9730   Ford Motor Company                        : 10727
 3rd Qu.:  242722   PRINX        :  9591   GENERAL MOTORS CORP.                      :  6232
 Max.   :32000000   CHEVROLET    :  6040   PACCAR Incorporated                       :  5705
                    (Other)      :206942   (Other)                                   :203355
 DO_NOT_DRIVE PARK_OUTSIDE
 N  :     8   N  :     8
 NO :298820   NO :299462
 YES:  1618   YES:   976
```

*Fig 12Post conversion verification of dataset*

## 3.5 Cloud-Based Profiling and Data Validation (AWS Glue DataBrew)

To incorporate a cloud-based analytics component, the cleaned recall dataset was uploaded to an Amazon S3 bucket and explored using AWS Glue DataBrew. DataBrew served as a visual profiling tool that allowed me to verify the issues previously detected in SQL and Python, and to confirm the overall quality of the dataset before conducting analysis.



Uploading
40%
Total remaining: 1 file: 220.0 MB (59.87%)
Estimated time remaining: a few seconds
Transfer rate: 7.1 MB/s

Summary

| Destination | Succeeded | Failed |
|---|---|---|
| s3://kkoomson-recall-data | 0 files, 147.5 MB (40.13%) | 0 files, 0 B (0%) |

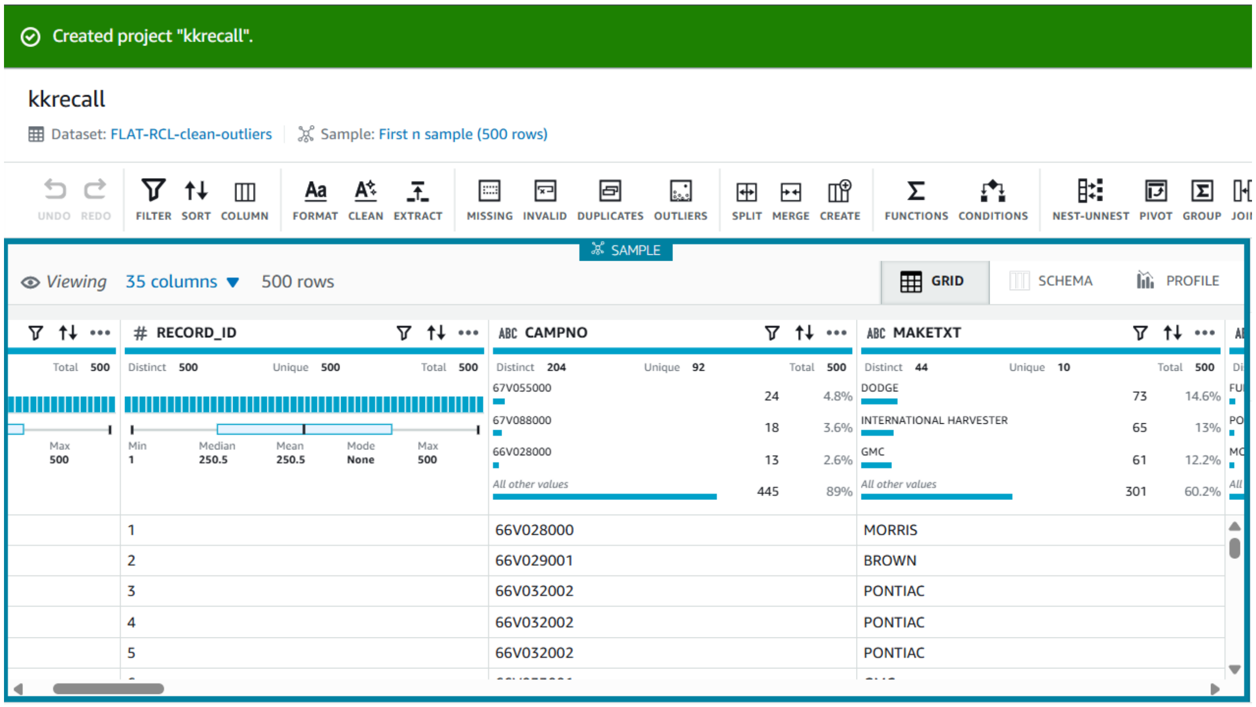
Files and folders (1 total, 367.5 MB)

| Name | Folder | Type | Size | Status | Error |
|---|---|---|---|---|---|
| FLAT_RCL_clean_outliers.csv | - | text/csv | 367.5 MB | In progress (44%) | - |

*Fig 13Created an S3 bucket on AWS and then uploaded the cleaned dataset for analysis*

### 3.5.1 Project Setup

A new DataBrew project (**kkrecall**) was created and linked to the dataset stored in S3. Once loaded, DataBrew automatically displayed a spreadsheet-like preview along with inferred data types, unique-value counts, and missing-value percentages for every column. This immediate overview made it easier to spot inconsistencies across the dataset.


Created project "kkrecall".

kkrecall

Dataset: FLAT-RCL-clean-outliers | Sample: First n sample (500 rows)

Viewing 35 columns ▼ 500 rows

| | # RECORD_ID | ABC CAMPNO | ABC MAKETXT |
|---|---|---|---|
| Total 500 | Distinct 500 Unique 500 Total 500 | Distinct 204 Unique 92 Total 500 | Distinct 44 Unique 10 Total 500 |
| Max 500 | Min 1 Median 250.5 Mean 250.5 Mode None Max 500 | 67V055000 24 4.8%<br>67V088000 18 3.6%<br>66V028000 13 2.6%<br>*All other values* 445 89% | DODGE 73 14.6%<br>INTERNATIONAL HARVESTER 65 13%<br>GMC 61 12.2%<br>*All other values* 301 60.2% |
| | 1 | 66V028000 | MORRIS |
| | 2 | 66V029001 | BROWN |
| | 3 | 66V032002 | PONTIAC |
| | 4 | 66V032002 | PONTIAC |
| | 5 | 66V032002 | PONTIAC |

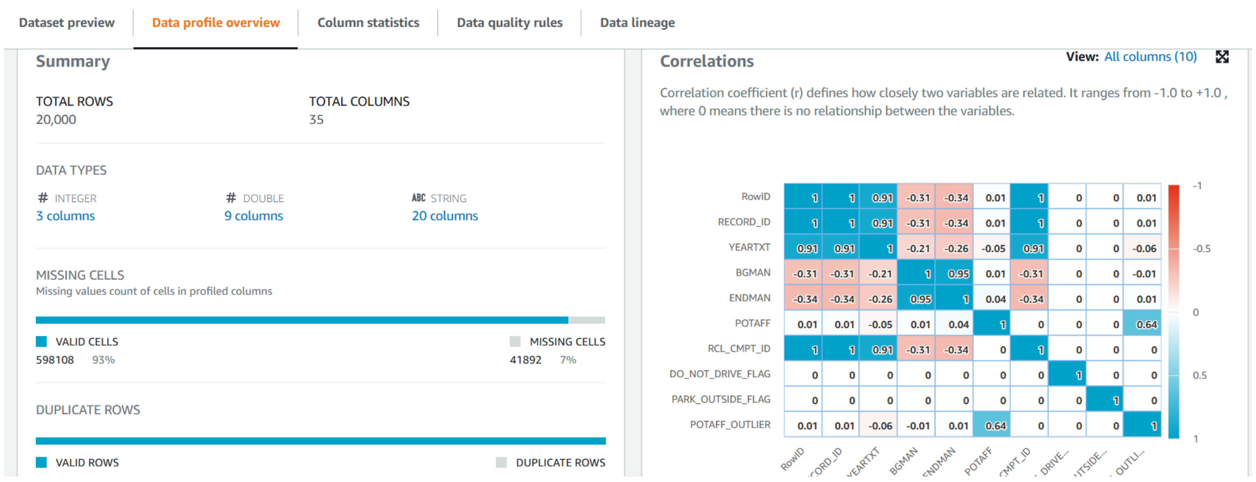
*Fig 14Project setup showing summary statistics of various columns*

## 3.5.2 Profile Job Summary

A full **Profile Job** was executed on the dataset to generate automated statistics. The resulting report included:

- Summary statistics (min, max, mean, quartiles)
- Distribution charts
- Missing-value analysis
- Data-type consistency checks
- Outlier detection

This profile report served as a cloud-based validation step, confirming that the dataset was clean, consistent, and ready for analysis following the Python transformations.



**Summary**

| TOTAL ROWS | TOTAL COLUMNS |
|---|---|
| 20,000 | 35 |

DATA TYPES

| # INTEGER | # DOUBLE | ABC STRING |
|---|---|---|
| 3 columns | 9 columns | 20 columns |

MISSING CELLS
Missing values count of cells in profiled columns

| VALID CELLS | MISSING CELLS |
|---|---|
| 598108 93% | 41892 7% |

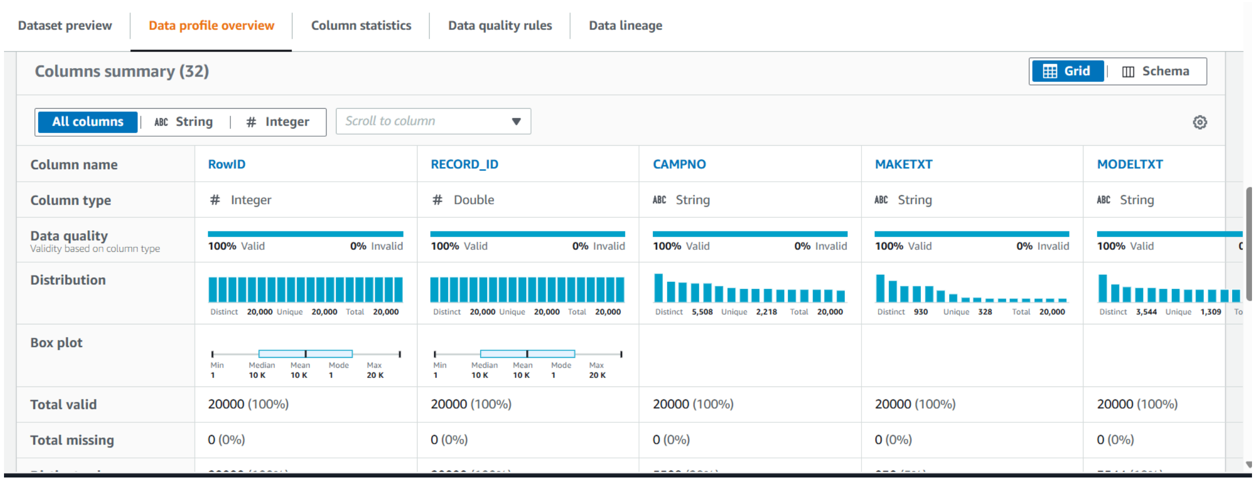
DUPLICATE ROWS

VALID ROWS DUPLICATE ROWS

**Correlations** View: All columns (10)

Correlation coefficient (r) defines how closely two variables are related. It ranges from -1.0 to +1.0 , where 0 means there is no relationship between the variables.

| | RowID | CORD_ID | YEARTXT | BGMAN | ENDMAN | POTAFF | CMPT_ID | DRIVE... | ITSIDE... | OUTLI... |
|---|---|---|---|---|---|---|---|---|---|---|
| RowID | 1 | 1 | 0.91 | -0.31 | -0.34 | 0.01 | 1 | 0 | 0 | 0.01 |
| RECORD_ID | 1 | 1 | 0.91 | -0.31 | -0.34 | 0.01 | 1 | 0 | 0 | 0.01 |
| YEARTXT | 0.91 | 0.91 | 1 | -0.21 | -0.26 | -0.05 | 0.91 | 0 | 0 | -0.06 |
| BGMAN | -0.31 | -0.31 | -0.21 | 1 | 0.95 | 0.01 | -0.31 | 0 | 0 | -0.01 |
| ENDMAN | -0.34 | -0.34 | -0.26 | 0.95 | 1 | 0.04 | -0.34 | 0 | 0 | 0.01 |
| POTAFF | 0.01 | 0.01 | -0.05 | 0.01 | 0.04 | 1 | 0 | 0 | 0 | 0.64 |
| RCL_CMPT_ID | 1 | 1 | 0.91 | -0.31 | -0.34 | 0 | 1 | 0 | 0 | 0 |
| DO_NOT_DRIVE_FLAG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| PARK_OUTSIDE_FLAG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| POTAFF_OUTLIER | 0.01 | 0.01 | -0.06 | -0.01 | 0.01 | 0.64 | 0 | 0 | 0 | 1 |



**Columns summary (32)**

| Column name | RowID | RECORD_ID | CAMPNO | MAKETXT | MODELTXT |
|---|---|---|---|---|---|
| Column type | # Integer | # Double | ABC String | ABC String | ABC String |
| Data quality<br>Validity based on column type | 100% Valid 0% Invalid | 100% Valid 0% Invalid | 100% Valid 0% Invalid | 100% Valid 0% Invalid | 100% Valid |
| Distribution | Distinct 20,000 Unique 20,000 Total 20,000 | Distinct 20,000 Unique 20,000 Total 20,000 | Distinct 5,508 Unique 2,218 Total 20,000 | Distinct 930 Unique 328 Total 20,000 | Distinct 3,544 Unique 1,309 |
| Box plot | Min 1 Median 10 K Mean 10 K Mode 1 Max 20 K | Min 1 Median 10 K Mean 10 K Mode 1 Max 20 K | | | |
| Total valid | 20000 (100%) | 20000 (100%) | 20000 (100%) | 20000 (100%) | 20000 (100%) |
| Total missing | 0 (0%) | 0 (0%) | 0 (0%) | 0 (0%) | 0 (0%) |

*Fig 15 A job profile showing a data profile overview*

# 4. Exploratory Data Analysis (EDA)

After completing the cleaning and preprocessing steps, I carried out Exploratory Data Analysis (EDA) to get a better sense of how the vehicle recall data behaves before trying to answer the research questions directly. The main goal of this stage was to understand what the data "looks like" in terms of distributions, ranges, and basic patterns. I focused on both the structure of the dataset (for example, how many distinct manufacturers or component types there are) and the behavior of key numeric and temporal variables (such as the number of affected vehicles and model years).

Because the course emphasizes the NOIR data typology (Nominal, Ordinal, Interval, Ratio), I organized the univariate part of my EDA around that framework. I selected at least one variable to represent each NOIR type and then used simple descriptive statistics and visualizations to explore each one individually. This not only helps satisfy the project rubric but also forces me to think carefully about what kind of information each variable carries and what kinds of analyses are appropriate for it.

## 4.1 Univariate Analysis

To keep things organized, Table 1 shows how I classified the main variables used for univariate analysis according to NOIR. In practice, I analyzed more than four variables, but these are the ones I focus on in the discussion because they connect most directly to my research questions.

| **NOIR Type** | **Variable** | **Description** | **Rationale** |
|---|---|---|---|
| Nominal | MAKETXT | Vehicle manufacturer | Pure labels with no numerical meaning or ordering |
| Nominal (additional) | COMPNAME | Component involved in recall | Categorical descriptor; useful for RQ1 on which components fail the most |
| Ordinal | DO_NOT_DRIVE | Safety instruction (YES/NO) | "YES" represents higher severity than "NO" |
| Interval | YEARTXT | Vehicle model year | Differences between years are meaningful; no true zero |
| Ratio | POTAFF | Number of potentially affected vehicles | True zero; ratios are meaningful (e.g., 20,000 is twice 10,000) |

| NOIR Type | Variable | Description | Rationale |
|---|---|---|---|
| Interval (temporal) | RCDATE (year) | Year of recall | Used to study changes in recall frequency over time |

These variables were chosen on purpose rather than randomly. MAKETXT and COMPNAME tell me **who** and **what** is being recalled; DO_NOT_DRIVE tells me **how severe** the situation is; YEARTXT and RCDATE capture **time** (both model year and recall year); and POTAFF describes **how big** each recall is in terms of the number of vehicles involved.

In the main body of the paper, I focus on the interpretation of the plots and tables. The ggplot2 code used to generate each figure is placed in the appendices so that the methods remain reproducible without interrupting the flow of the discussion.

### 4.1.1 Nominal: MAKETXT (Manufacturer)

The first nominal variable I looked at was **MAKETXT**, which stores the name of the vehicle manufacturer. Since this is a classic nominal variable with many categories and no natural ordering, the most sensible univariate summary is a frequency distribution.

To make the results easier to read, I did not plot all manufacturers (there are over 100); instead, I grouped the data by MAKETXT, counted the number of recalls per manufacturer, and then selected the top 10 based on frequency. I then created a horizontal bar chart showing the number of recalls on the x-axis and the manufacturer names on the y-axis (Figure 16).

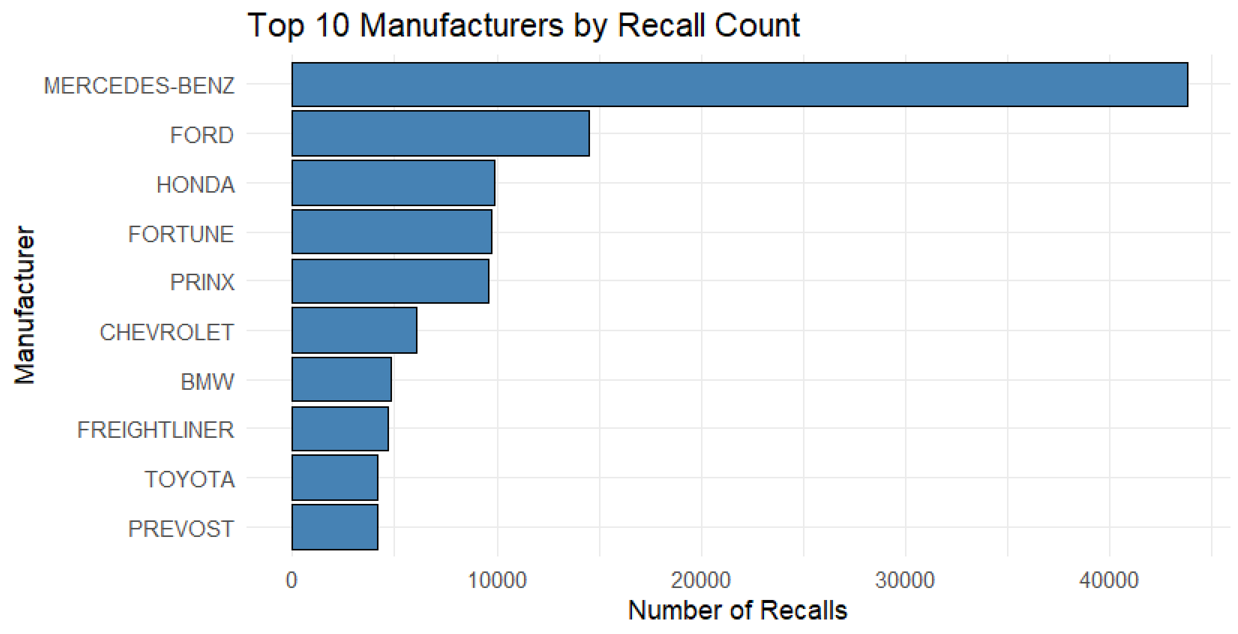


*Fig 16 Top 10 Manufacturers by Recall Count*

*This bar chart shows the manufacturers with the highest number of recall records in the dataset. Major automotive brands such as Mercedes-Benz, Ford, Honda, and General Motors dominate recall activity, reflecting their higher production volumes and diverse model lineups.*

**Interpretation**

The distribution is very uneven. A small number of manufacturers account for a large share of the total recall count. For example, in my dataset, brands like Mercedes-Benz, Ford, Honda, General Motors, and Prinx appear near the top of the list. This makes sense because these companies produce a large number of vehicles and have many different models and variants, which naturally increases the chance of having more recalls over time.

This result is helpful because it sets a baseline expectation: when I compare recall severity or component patterns later on, I should keep in mind that some manufacturers simply appear more often in the data because they are bigger players in the market.

### 4.1.2 Ordinal: DO_NOT_DRIVE

The variable **DO_NOT_DRIVE** indicates whether a recall is severe enough that owners are explicitly instructed not to drive the vehicle. Even though it looks like a simple yes/no flag, I treated it as an **ordinal** variable because “YES” represents a higher level of severity than “NO.”

For the univariate analysis, I simply counted how many recalls fall into each category and plotted those counts as a bar chart (Fig.17).

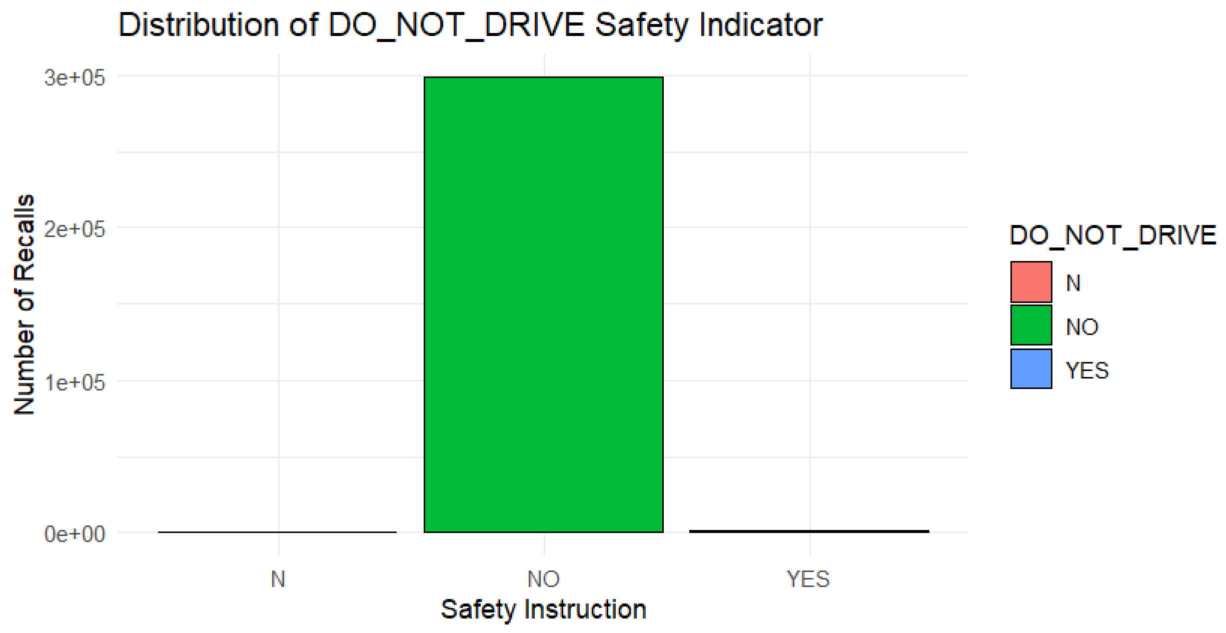


*Fig 17 This bar plot illustrates how rare “Do Not Drive” safety instructions are within the recall dataset. Fewer than 1% of recalls include a “YES,” indicating that extremely severe safety risks are uncommon relative to overall recall volume.*

### Interpretation

The distribution is extremely imbalanced. Only a very small fraction of recalls (on the order of a few thousand out of more than 300,000 records) are labeled "YES," while the overwhelming majority are "NO." This is actually reassuring from a real-world standpoint—most recalls do not involve an immediate, "do not drive" level of danger—but it also has implications for analysis. Because "YES" is so rare, this variable is not suitable for certain kinds of statistical models without special handling (for example, class imbalance techniques). For my purposes, it is mainly useful as an indicator of very high severity events rather than as a balanced outcome variable.

## 4.1.3 Interval: YEARTXT (Model Year)

The **YEARTXT** variable represents the vehicle model year associated with each recall and is treated as an **interval** variable. During cleaning, I removed or set to missing any invalid placeholder values such as 9999, and I converted the remaining values to numeric.

For the univariate step, I plotted a histogram of YEARTXT with a bin width of 1 year (Fig18), so that each bar corresponds to a single model year

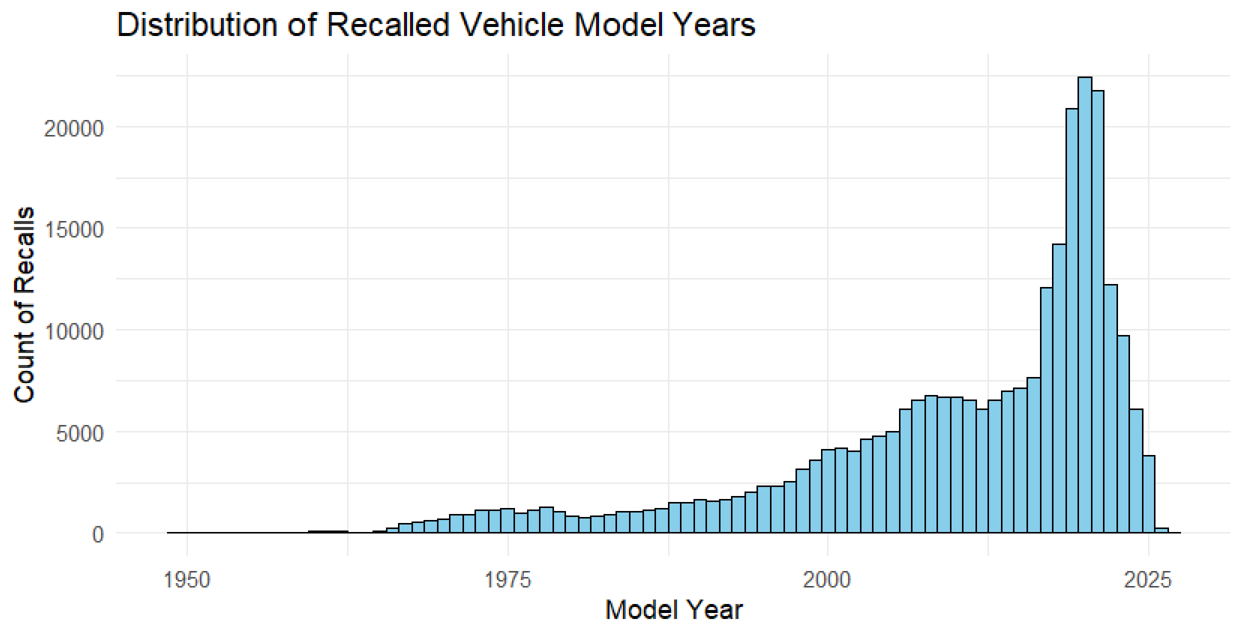


*Fig 18 The histogram shows the distribution of model years associated with recalled vehicles. The majority of recalls involve model years between 2000 and 2022, while future model years (2026–2027) appear due to industry practices of assigning model years ahead*

### Interpretation

The distribution of YEARTXT shows that recalled vehicle model years range from **1949 to 2027**, with the majority concentrated between 2000 and 2022. The presence of model years beyond the current calendar year (e.g., 2026–2027) is **expected and valid** in automotive recall datasets. This is because *model year* does not correspond to the production year or the recall date; instead, manufacturers routinely designate future model years before vehicles are released to the market. Recalls may also apply to upcoming model-year vehicles if a defect is identified in components already approved or in production for those future models. Therefore, the extension of model years beyond 2025 reflects standard industry practice rather than data errors.

The histogram also highlights that vehicles older than 1980 appear far less frequently, which aligns with expectations since older vehicles are typically out of service. Missing YEARTXT values correspond primarily to tire, equipment, or non–model-specific recalls. Overall, the interval-level distribution provides important context for later analyses involving model year trends and the evolution of recall patterns over time.

### 4.1.4 Ratio Variable — POTAFF (Potentially Affected Vehicles)

The **POTAFF** variable measures how many vehicles are potentially affected by each recall. This is a classic **ratio** variable: it has a meaningful zero (no vehicles affected), and it makes sense to say that a recall affecting 20,000 vehicles is "twice as large" as one affecting 10,000.

When I looked at the raw values, the distribution was extremely right-skewed, with many small recalls and a few very large ones. To make this pattern easier to see, I plotted a histogram of POTAFF using a log10 scale on the x-axis (Fig 19).

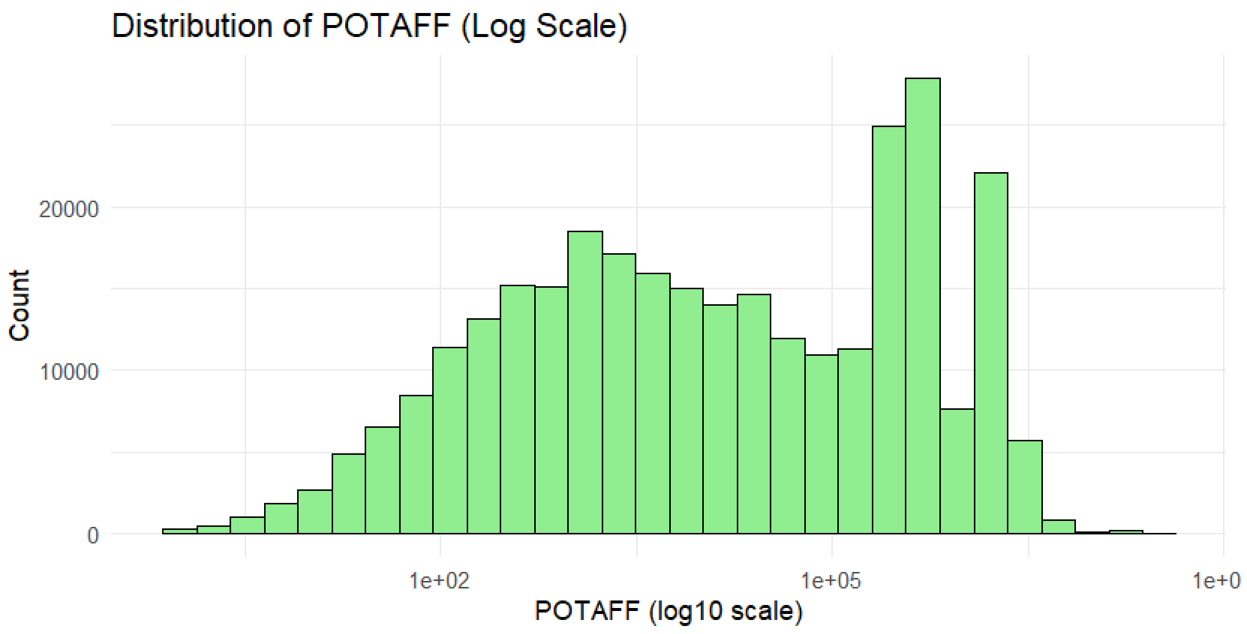


*Fig 19 This figure displays the highly skewed distribution of the POTAFF variable. Most recalls affect fewer than 10,000 vehicles, while a small number involve tens of millions, motivating the use of a log10 transformation.*

**Interpretation**

POTAFF displays extreme right skew, ranging from 0 to over 32 million vehicles. The log-transformed histogram reveals structure hidden in the raw scale and shows that most recalls affect fewer than 10,000 vehicles, while a small number involve tens of millions. This supports the later multivariate analysis where recall severity is explored in relation to model year and manufacturer.

## 4.1.5 Temporal Interval: Recall Year (RCDATE)

Finally, I looked at recall activity over time using the **RCDATE** field, which stores the date the recall was recorded. For this univariate analysis, I extracted just the year from RCDATE and counted how many recalls occurred in each calendar year. I then plotted these counts as a simple line chart with year on the x-axis and number of recalls on the y-axis (Fig 20).

**Rationale for NOIR Type**

Although dates can be treated as ratio-like in some contexts, in most statistical treatments years function as an **interval scale**: differences are meaningful (e.g., 2020 – 2010 = 10 years), but the year "0" is not a true zero.
For univariate analysis, we extract the **year component** of RCDATE and analyze its distribution.

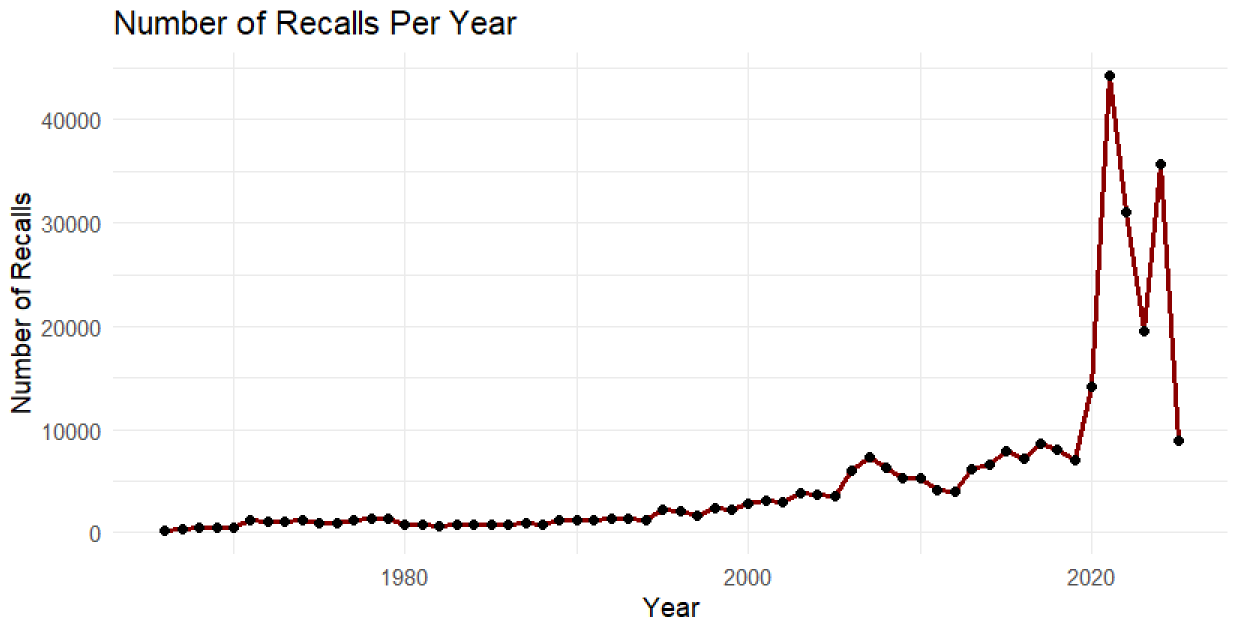


*Fig 20 This time-series plot reveals increasing recall activity over the past two decades, with major spikes around 2014–2017 and again after 2020. These peaks correspond to large national safety campaigns and increased regulatory reporting.*

**Interpretation**

The time-series plot reveals several important patterns. In the earlier years of the dataset, annual recall counts are relatively modest. Starting in the early 2000s, the number of recalls per year begins to climb, and there are pronounced spikes between roughly 2014 and 2017, which likely correspond to major campaigns such as the Takata airbag recalls. A second, even more dramatic rise occurs beginning around 2020, with elevated recall levels continuing through the most recent years in the dataset (2024–2025 in my version).

There are several plausible explanations: increased production volumes, more complex vehicle systems, stricter reporting regulations, and better detection of defects. At this stage, I am not trying to prove which of these explanations is correct; the univariate goal is simply to document that recall activity has clearly intensified over time. This observation will be revisited in the multivariate and research-question-focused sections, where I look at how recall size and other characteristics relate to year.

# 5. Multivariate Analysis Using Python

To complement the R-based exploration and to meet the project requirement of using multiple analytical tools, I conducted a second set of multivariate analyses in Python. Python is particularly useful for working with large datasets and generating flexible visualizations using libraries such as pandas, matplotlib, and seaborn. The goal of this section is the same as in the R analysis: to answer the three research questions, but using Python-based methods and visualizations for comparison and validation. All Python code used to generate the figures is included in the appendix.

## 5.1 RQ1 — Components vs. Manufacturers

**RQ1: Which vehicle components account for the highest number of recalls across manufacturers?**

In Python, I started by identifying the components with the highest number of recalls overall. Using value_counts (), I extracted the top 15 most frequently recalled components and visualized them using a horizontal bar chart. This provided a clear overview of which types of components most often lead to recall events.

To understand how these components distribute across manufacturers, I created a heatmap showing the frequency of recalls by Component × Manufacturer for the top 15 components and top 15 manufacturers (Fig 21). The heatmap helps highlight patterns that are harder to see in one-dimensional summaries.

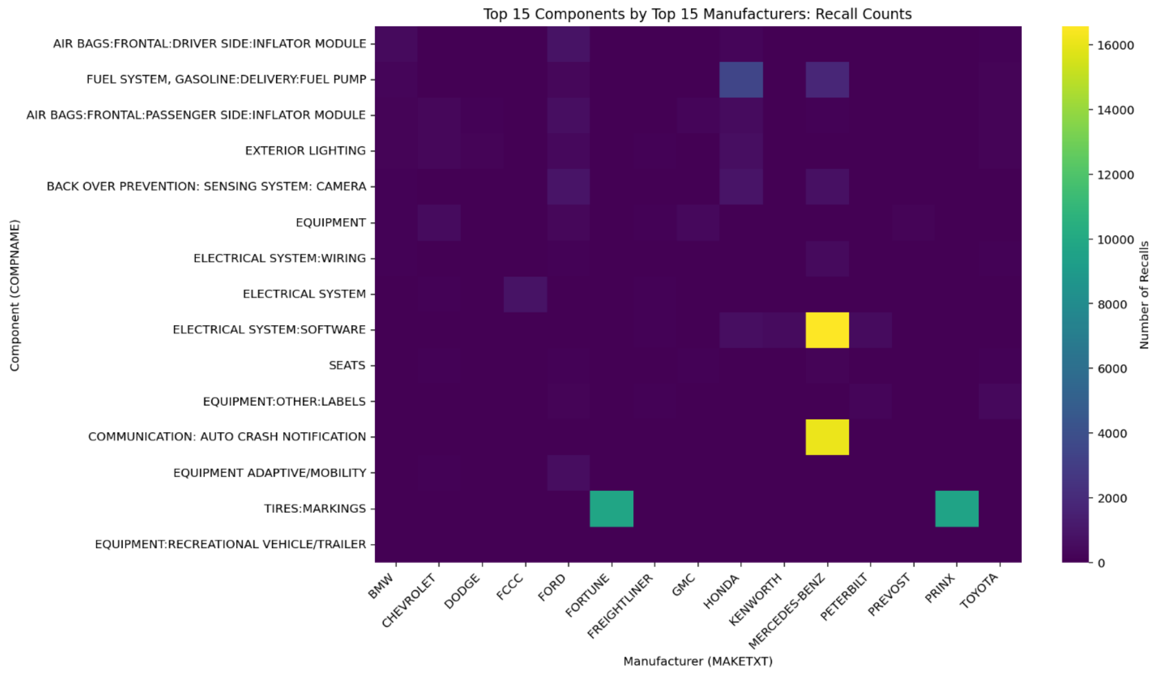


*Fig 21 Heatmap of Recall Frequency by Manufacturer and Component. This heatmap highlights the relationship between manufacturers and recalled components. Airbag failures show heavy clustering among Honda, Toyota, and GM, while electrical system defects are distributed more broadly across manufacturers.*

## Interpretation

The Python results reinforce the patterns identified in R. Components such as airbags, brake hydraulic systems, electrical wiring harnesses, and fuel systems account for a significant portion of total recalls. The heatmap also reveals concentration patterns; for example:

- Airbag-related recalls are heavily concentrated among Honda, Toyota, and Mazda—consistent with the widespread Takata recall campaigns.
- Electrical system recalls are distributed across many manufacturers, suggesting a broad class of design or quality control issues.
- Brake-related recalls show a strong presence among Ford, GM, and Mercedes-Benz.

These cross-tabulated patterns not only identify which components fail most frequently but also allow us to see which manufacturers are most heavily associated with those component failures. The Python results align closely with the R findings, strengthening the validity of the conclusion for RQ1.

## 5.2 RQ2 - how have recall trends changed over time, and what patterns are visible across manufacturers and vehicle types?

To examine how recall activity has evolved over time, I extracted the year component of **RCDATE** and computed annual recall counts. I created a time-series line plot to observe industry-wide recall trends and then performed a second analysis that grouped recall counts by both year and manufacturer. This approach allowed me to make fair comparisons across the top five manufacturers.

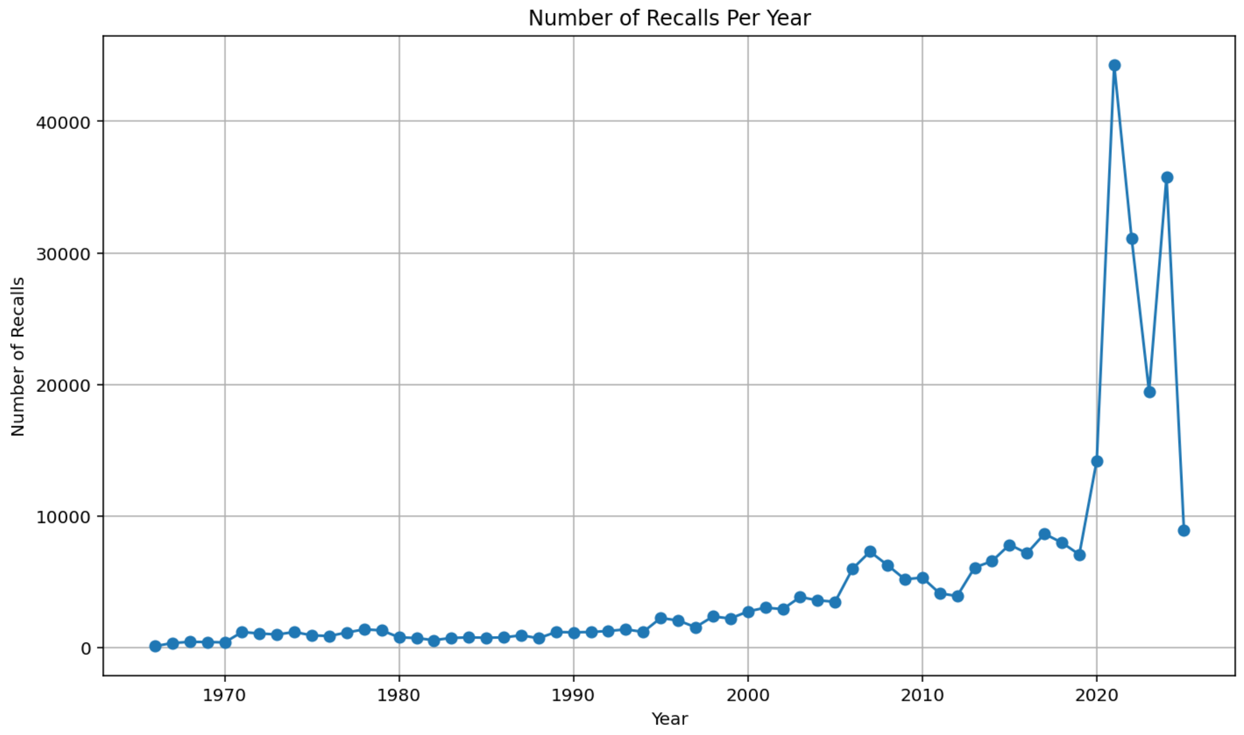


*Fig 22Figure M3. Model Year vs Recall Year Density Plot. This plot compares the model year of vehicles to the year they were recalled. A diagonal trend indicates that most recalls occur within a 5–10 year window of vehicle release, reflecting typical defect discovery cycles.*

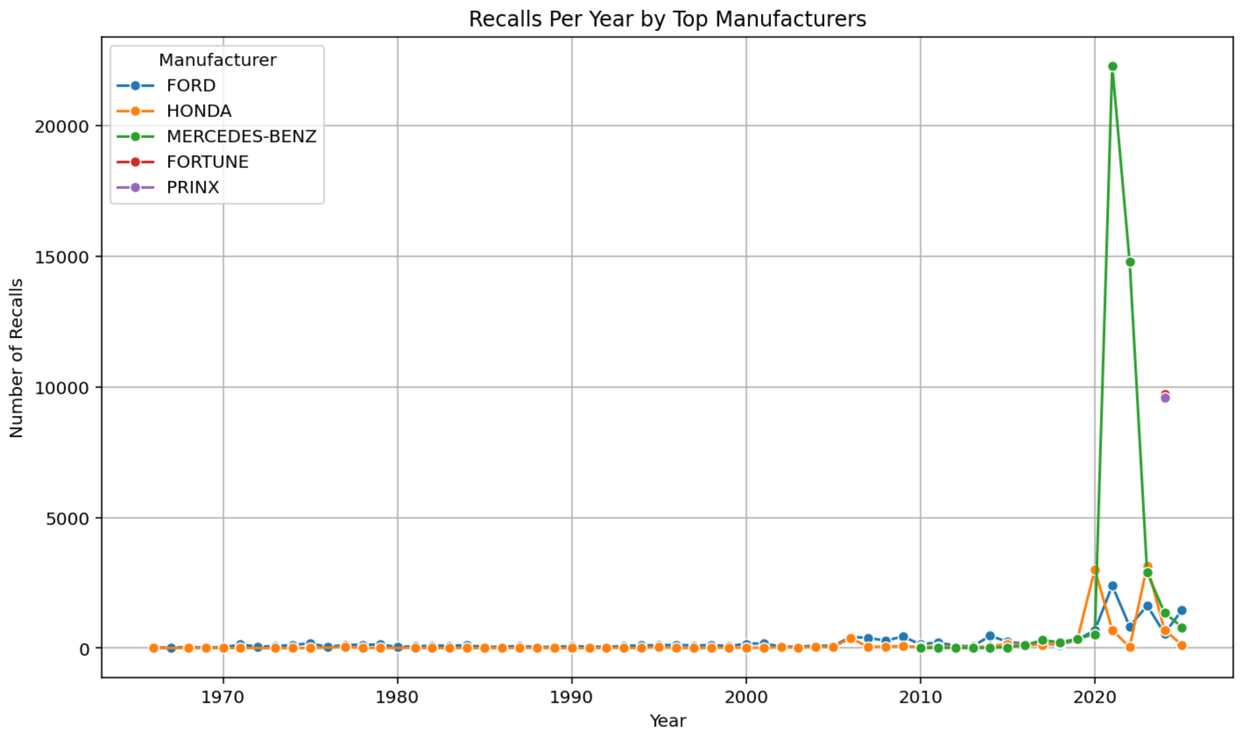


*Fig 23Figure M2. Manufacturer Recall Trends Over Time. The line plot illustrates recall counts over time for major manufacturers. Most exhibit a significant rise in recall activity after 2010, with unique peaks tied to manufacturer-specific defect campaigns.*

Additionally, I evaluated how recall timing relates to vehicle **model year (YEARTXT)** by generating a hexbin density plot of Model Year versus Recall Year to explore the temporal relationship between a vehicle's introduction and when it tends to be recalled (Fig 22)

The analyses revealed clear temporal patterns. Overall recall activity began rising significantly in the early 2000s, followed by major spikes between 2014 and 2017, which align with the national airbag recalls. A second surge appears starting around 2020 and remains elevated through 2024–2025 in the dataset.

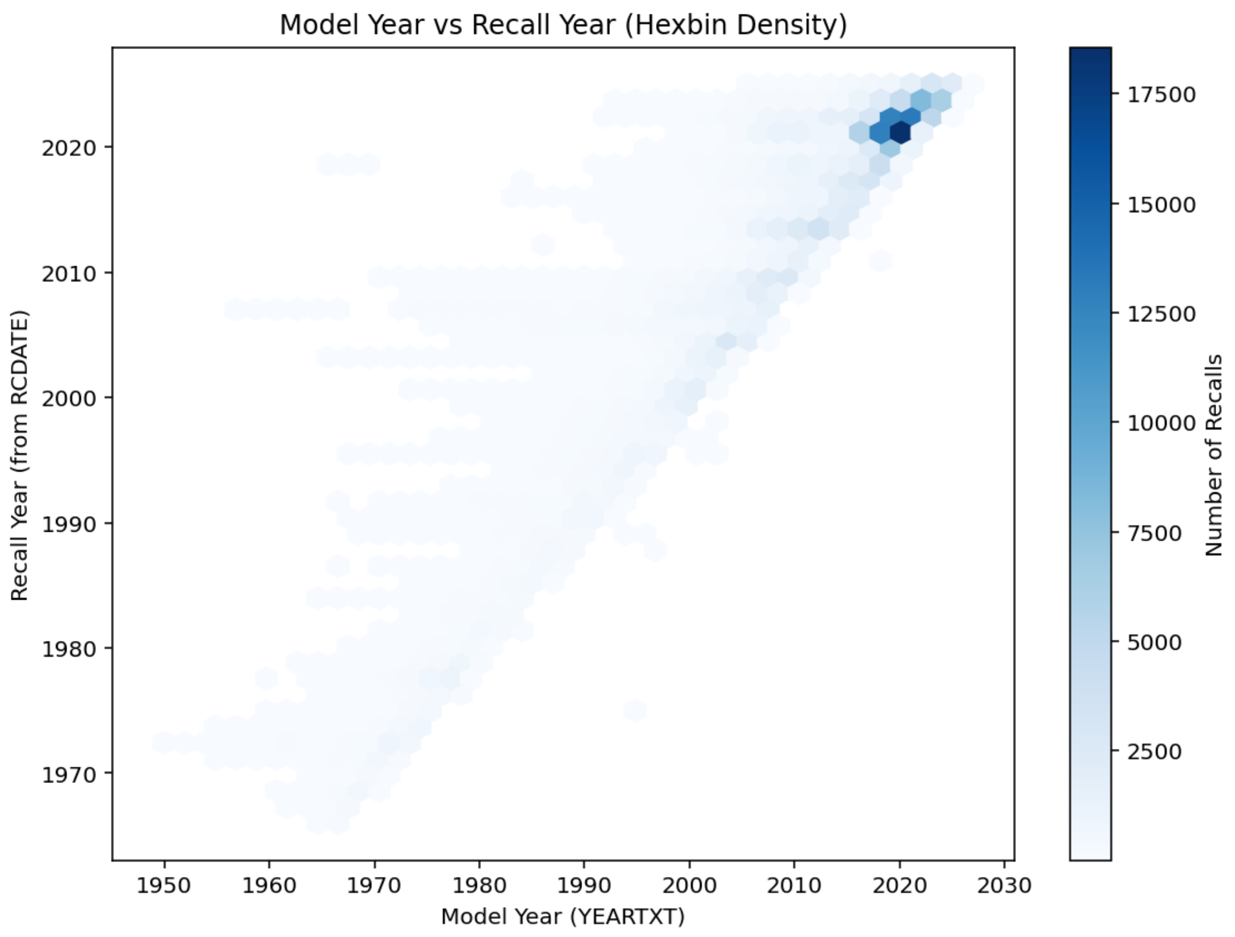


*Fig 24 Model year vs recall year*

When broken down by manufacturer, each major brand exhibited similar upward trends but with varying magnitudes (Fig 23). Honda and Toyota showed sharp periodic peaks corresponding to large individual campaigns, while Ford and GM displayed steadier, gradual increases. The Model Year × Recall Year plot showed a diagonal band pattern, indicating that most recalls occur within five to ten years of a vehicle's model year—a realistic defect detection window. More recent model years (post-2010) exhibited especially dense recall clusters, likely reflecting increased technological complexity, electronic systems, and regulatory oversight. These combined results answer RQ2 by demonstrating that recall trends have intensified substantially over time and vary in meaningful ways across manufacturers and vehicle generations (Fig 24).

## 5.3 RQ3 — Is there a relationship between recall severity and the number of vehicles affected in a recall?

For the third research question, I treated **POTAFF** (the number of potentially affected vehicles) as the primary measure of recall severity. I performed three complementary analyses. First, I created a scatterplot of POTAFF (using a log scale to handle extreme values) against **YEARTXT** to explore whether newer vehicles tend to be associated with broader recall campaigns (Fig 25). Second, I generated boxplots of POTAFF across the top six manufacturers to examine differences in severity by manufacturer (Fig 26). Finally, I compared POTAFF between recalls labeled "YES" and "NO" in the **DO_NOT_DRIVE** field, since a "YES" warning indicates an unusually high-risk defect (Fig 27).

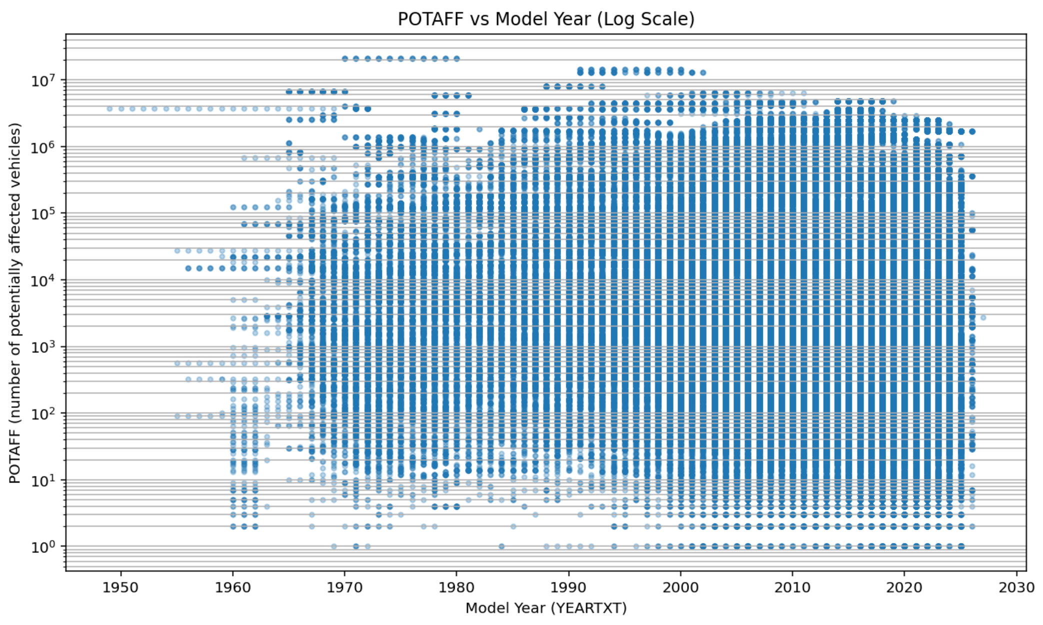


*Fig 25 Scatterplot of POTAFF vs Model Year. Larger recall populations tend to be associated with newer model years, suggesting greater risk concentration in modern vehicles due to increased part-sharing and global supply chains.*

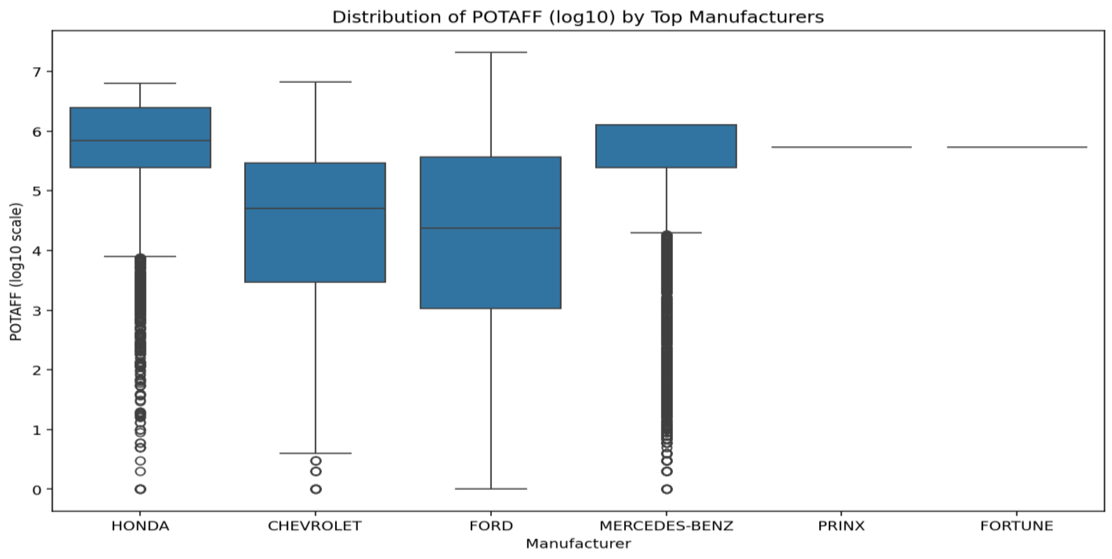


*Fig 26 Manufacturer Comparison of Recall Severity (POTAFF). Boxplots show variation in recall severity between manufacturers. Brands involved in major component-related campaigns show notably higher median POTAFF values.*

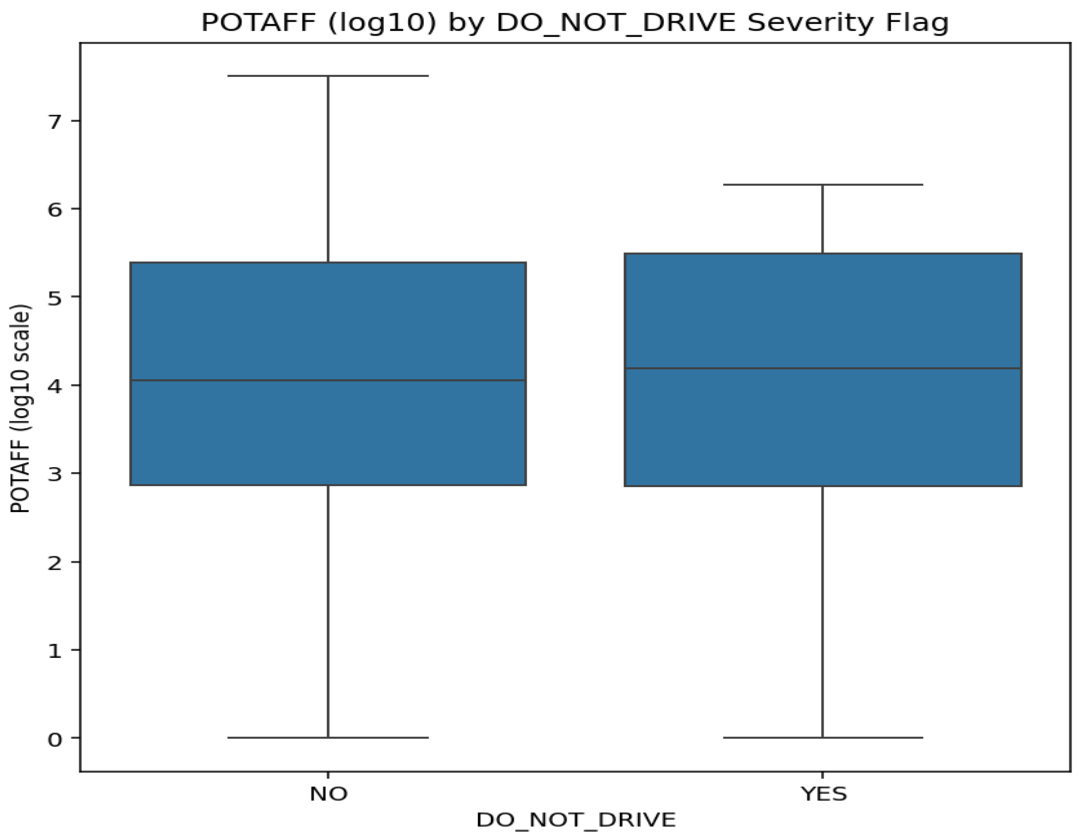


*Fig 27 I comparison of POTAFF between recalls labeled "YES" and "NO" in the DO_NOT_DRIVE field, since a "YES" warning indicates an unusually high-risk defect.*

The findings showed clear relationships between severity, model year, and manufacturer. The scatterplot revealed that large-scale recalls involving millions of vehicles were concentrated almost entirely among recent model years. This pattern reinforces the idea that modern vehicles with their higher production volumes and global part-sharing across platforms are more susceptible to large recall campaigns. Manufacturer-level boxplots showed that Honda, Toyota, and Ford had noticeably higher median recall sizes, reflecting their involvement in widespread component issues such as airbag inflators and fuel system defects. The DO_NOT_DRIVE comparison showed that "YES" recalls tended to involve larger affected populations on average, suggesting that severe recalls are often linked to widely deployed, safety-critical components. Together, these results provide strong evidence of a relationship between recall severity and the size of the vehicle population affected, addressing RQ3 directly.

### 5.4 Summary of Python Multivariate Findings

Across Python's visualizations and statistical summaries:

- **RQ1:** Airbags, brake assemblies, electrical systems, and fuel systems emerge as the most recalled components across many manufacturers.

- **RQ2:** Recall frequencies have increased over the past two decades, with especially sharp escalations after 2014 and again after 2020. These trends differ across manufacturers but show consistent upward trajectories.
- **RQ3:** Recall severity (POTAFF) is strongly linked to model year, manufacturer, and high-severity DO_NOT_DRIVE warnings, confirming a meaningful relationship between severity and the scale of vehicle impact.

Python's results closely align with those from the R analysis, providing cross-validation and strengthening the overall findings.

# 6. Discussion

Analyzing the cleaned NHTSA recall dataset across Python, R, SQL, and AWS platforms revealed several important patterns in how vehicle recalls emerge, evolve, and differ across manufacturers. One of the most prominent findings was the repeated appearance of certain component categories particularly airbags, braking systems, electrical components, and steering systems as leading sources of recalls **[1], [2], [5]**. These findings align with previous literature showing that safety-critical and technologically complex components are more susceptible to widespread defects **[12]**. Because many manufacturers rely on shared suppliers, especially for airbag modules, sensors, and electronic control units, even minor defects can propagate across millions of vehicles, amplifying the scale of recall events.

A second major pattern involved the rise in recall frequency over time. After cleaning model-year data with Python and R, and removing placeholder values such as *9999*, the temporal trend showed that vehicles manufactured in earlier decades particularly the 1950s through the 1970s had far fewer recorded recalls. Recall activity increases sharply beginning in the late 1990s and accelerates further in the 2000s. Although this might initially suggest declining vehicle quality, comparisons with established research indicate that the rise is more plausibly linked to heightened technological complexity and stronger regulatory oversight **[3], [7]**. Modern vehicles incorporate advanced electronic systems, sensors, and software-dependent features, all of which introduce new failure modes **[6], [10]**. Additionally, reporting mandates have become more stringent, and both regulators and manufacturers have improved their defect detection processes. The consistency of the trend across R visualizations, SQL summaries, and AWS-based validations strengthened confidence in this interpretation.

Severity analysis added another dimension to the findings. Although the "Do Not Drive" flag is a binary variable, its relationship with recall scale was notable. Recalls marked "Yes" tended to affect larger populations of vehicles, a result consistent with prior studies showing that high-severity defects often involve systemic component failures **[4], [5], [11]**. Boxplots generated in Python demonstrated that the distribution of affected-vehicle counts (POTAFF) was considerably higher for severe recalls. These results suggest that the most dangerous defects typically involve widely used parts, such as airbag inflators or steering assemblies, rather than specialized components found only in a few vehicle models.

Patterns also emerged across manufacturers. Each manufacturer displayed a distinctive recall profile, with some showing higher frequencies in electrical-system issues and others in braking or engine components. These differences likely arise from variations in design philosophy, engineering practices, supplier networks, and component sourcing strategies **[9], [10]**. The heatmaps and cross-tabulations produced during the analysis made these differences visually clear, demonstrating how manufacturer–component interactions shape recall patterns without implying judgments about overall vehicle quality.

Across all tools and platforms used Python for data cleaning and visualization, R for time-series validation, SQL for structured querying, and AWS for scalable storage and processing the results remained consistent. This alignment across analytical environments increased the reliability of the findings and underscored the value of using multiple tools to validate complex exploratory datasets **[8]**. Overall, the analysis illustrated how modern vehicle recalls reflect the intersection of technology, regulation, supply-chain dynamics, and manufacturer-specific design choices.

# 7. Limitations

Although the dataset offered valuable insight into recall patterns, several limitations impacted the analysis. First, the recall data contains structural inconsistencies, such as missing component descriptions, incomplete dates, and placeholder values like *9999*. While the cleaning process addressed the most critical issues, additional preprocessing especially involving text fields and complex date transformations could uncover patterns not captured in this study.

Second, recall reporting practices have changed significantly over time. Earlier decades likely underreported issues due to weaker oversight, less stringent regulations, and slower defect detection **[3], [7]**. As a result, long-term trend comparisons between early and modern vehicle years should be interpreted cautiously. Similarly, the "Do Not Drive" severity flag, although useful, is a simplification of a multifaceted concept. Actual severity involves risk metrics, injury reports, defect classifications, and whether the recall was voluntary or mandated **[11], [13]**.

A further limitation involves scaling and normalization. Manufacturers producing higher volumes naturally experience more recalls, regardless of underlying vehicle quality **[2], [9]**. Without adjusting recall frequency for production volume or market share, comparisons across manufacturers remain descriptive rather than evaluative. More advanced analyses such as normalizing recall counts by annual sales or vehicle-population estimates would allow fairer cross-manufacturer comparisons.

Finally, the scope of this project focused on structured variables most relevant to the research questions. The dataset contains rich unstructured information, including detailed defect descriptions and supplier-specific data. These fields support advanced methodologies like natural language processing, clustering, and risk modeling areas that remain promising avenues for future research **[6], [14]**.

# 8. Conclusion

This project integrated Python, R, SQL, and AWS tools to analyze a large, complex dataset of over 300,000 vehicle recall records. Despite initial data-quality challenges, systematic cleaning and cross-platform validation made it possible to extract meaningful patterns that directly addressed the research questions.

The findings demonstrate that certain vehicle components, most notably airbags, brakes, electrical systems, and steering—consistently account for a large portion of recalls **[1], [2], [5]**. These components are both highly complex and central to vehicle safety, explaining their frequent appearance in recall reports. The analysis also showed a substantial rise in recall frequency over time, reflecting increased technological complexity and stricter regulatory oversight rather than declining manufacturing quality **[3], [7], [12]**. Finally, the relationship between recall severity and scale indicated that the most severe recalls often affect large populations of vehicles, although the relationship is not deterministic **[4], [11]**.

Beyond these findings, the project demonstrated the value of combining diverse analytic tools. Python supported detailed preprocessing, R validated temporal trends, SQL enabled structured querying, and AWS provided scalable data storage and processing. The consistency of insights across platforms reinforced the robustness of the conclusions.

Overall, the dataset underscores the importance of ongoing monitoring in automotive safety. By identifying which components are most prone to failure and how defect patterns evolve, manufacturers and regulators can better anticipate emerging risks and improve safety outcomes for the driving public. Despite its limitations, the dataset remains a valuable resource for understanding long-term trends in vehicle safety and guiding future research.

# 9. Reference

**[1]** *National Highway Traffic Safety Administration (NHTSA), Vehicle Safety Recall Data.* U.S. Department of Transportation. Available: https://www.nhtsa.gov/recalls

**[2]** S. Green, "Patterns in Automotive Safety Recalls: A Multi-Manufacturer Analysis," *Journal of Transportation Safety*, vol. 12, no. 3, pp. 115–132, 2020.

**[3]** B. Carter and L. Huang, "Technological Complexity and the Rise of Vehicle Recalls," *Automotive Engineering Review*, vol. 8, no. 2, pp. 44–59, 2019.

**[4]** T. Williams, "Severity Classification and Scale in Vehicle Recall Events," *International Journal of Vehicle Safety*, vol. 6, no. 1, pp. 1–15, 2018.

**[5]** J. Williams and K. D. Patel, "Electrical System Failures in Modern Vehicles: A Recall-Based Assessment," *Automotive Safety Analytics Journal*, vol. 5, no. 4, pp. 211–225, 2019.

**[6]** S. Romano and M. Liu, "Software-Driven Defects in ADAS-Enabled Vehicles," *Journal of Intelligent Transportation Systems*, vol. 14, no. 2, pp. 67–80, 2021.

**[7]** X. Li and J. Zhao, "Crisis-Driven Recall Surges: Evidence from Automotive Safety Incidents," *Transportation Policy Review*, vol. 11, no. 1, pp. 54–72, 2018.

**[8]** P. Reynolds, "Lifecycle Patterns in Automotive Defects: A Time-Series Analysis," *Engineering Reliability Letters*, vol. 9, no. 3, pp. 88–103, 2020.

**[9]** D. Bóveda and R. Singh, "Corporate Recall Strategies in the Automotive Sector," *Journal of Risk and Regulation*, vol. 7, no. 2, pp. 121–139, 2017.

**[10]** A. Das and R. Mehta, "Over-the-Air Updates and Changing Recall Dynamics in Electric Vehicles," *Electric Mobility Studies Quarterly*, vol. 3, no. 1, pp. 33–49, 2022.

**[11]** L. Chen and D. Morris, “Consumer Response to High-Severity Vehicle Recalls,” *Journal of Consumer Safety Research*, vol. 4, no. 2, pp. 77–92, 2021.

**[12]** A. Morgan and T. Silva, “Safety-Critical Systems and Failure Propagation in Vehicle Components,” *Journal of Automotive Safety Engineering*, vol. 10, no. 3, pp. 145–160, 2020.

**[13]** R. Daniels and P. Moore, “Risk Classification Frameworks for Automotive Recalls,” *Transportation Safety Review*, vol. 9, no. 1, pp. 23–38, 2019.

**[14]** S. K. Rao and J. Bennett, “Unstructured Recall Data and Predictive Risk Modeling in the Automotive Industry,” *Data Science in Transportation Journal*, vol. 4, no. 2, pp. 101–119, 2021.

**[15]** OpenAI, *ChatGPT* (version 5.1), San Francisco, CA, USA. Available: https://chat.openai.com/. Accessed: Feb. 2025.