**Texas Department of Insurance: Insurance Complaints – Data Cleaning Report**

Introduction:

The Texas Department of Insurance (TDI) is a government regulatory body responsible for regulating the Texas insurance industry. As part of their regulatory responsibilities, TDI is responsible for investigating and resolving complaints submitted by insureds against insurance companies.

AS TDI collects more information about complaints, the potential for deriving insights from TDI’s complaints data only grows. However, before any analysis can begin, we must clean the data to remove any errors/inconsistencies that may affect the trustworthiness of further analysis.

The objectives of this report are to:

1. assess the cleanliness and readiness of the dataset for analysis,
2. walkthrough data cleaning processes, and
3. create a descriptive analytics-ready version of the data.

About the data:

The data used for this report was extracted from Data.gov ([Insurance complaints: All data - Catalog](https://catalog.data.gov/dataset/insurance-complaints-all-data)) on March 21 2025, as a comma separated value (csv) file. The data contains 266,551 individual records, each representing one complaint, with 17 columns. Each record represents a unique combination of the complaint\_no and the respondent\_id column. The Data dictionary for the raw, uncleaned and unedited datasets data dictionary is available in Exhibit 1.1.

Exhibit 1.1

Data Dictionary of the Original Dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Data Type | Example | Description |
| complaint\_no | Integer | 1 | The number assigned to a specific complaint. |
| complaint\_filed\_against | Text | “MEDCO HEALTH, LLC” | The name of the person or organization the complaint was filed against. |
| complaint\_filed\_by | Text | “Insured” | The role of individual who filed the complaint. |
| reason\_complaint\_filed | Text | “Other; Refusal to Insure” | List of reasons why a complaint was filed. |
| confirmed\_complaint | Text | “Yes” | Whether or not the licensed person or organization was in error. |
| how\_resolved | Text | “Claim Settled; Question of Fact” | The actions taken to resolve a claim. |
| received\_date | Text | “06/07/17” | The date TDI received the complaint. |
| closed\_date | Text | “09/07/12” | The date TDI closed the complaint. |
| complaint\_type | Text | “Life, Accident and Health” |  |
| coverage\_type | Text | “Accident and Health” | The type of insurance coverage related to the complaint. |
| coverage\_level | Text | “Individual Life” |  |
| others\_involved | Text | “Associated Agent” | A list of the other individuals involved in a complaint. |
| respondent\_id | Integer | 132 | The number assigned to the person or organization the complaint was filed against. |
| respondent\_role | Text | “Ins Co – License Active” | The role of the individual or organization that complaint was filed against. |
| respondent\_type | Text | “Organization” | Whether or not the complaint was filed against a person or organization. |
| complainant\_type | Text | “INDV” | Whether or not an individual or organization filed the complaint. |
| keywords | Text | “CLAIM EVALUATION” | Additional information about the complaint. |

Database Upload Process

We downloaded the data from Data.gov on March 21 2025. After downloading the data, we created a schema named ‘insurance\_complaints’ and a table named ‘complaints’ in MySQL Workbench.

Then we imported the data into our complaints’ table, using MySQL’s Table Data Import Wizard. MySQL successfully imported all records from the original csv file into the ‘complaints’ table.

Data Exploration

Uniqueness Check

To ensure there were no duplicates within the dataset, we checked if there were any complaint\_no – respondent\_id pairs that appeared more than once. To achieve this goal, we used the SQL query in Exhibit 1.2. No records were returned, indicating no complaint\_no – respondent\_id pairs appear more than once within the dataset.

Exhibit 1.2 – Duplicate Check SQL Query.

|  |
| --- |
| SELECT  complaint\_no,  respondent\_id,  COUNT(\*) AS `no\_of\_occurences`  FROM insurance\_complaints.complaints  GROUP BY complaint\_no, respondent\_id  HAVING COUNT(\*) > 1  ORDER BY COUNT(\*) DESC; |

Categorical Value Check

For each categorical column, we created a query (Exhibit 1.3) to return a table of all distinct values, and the number of times these values occurred.

Exhibit 1.3 – Column Categories Check

|  |
| --- |
| SELECT  categorical\_column,  COUNT(categorical\_column) As `count`  FROM insurance\_complaints.complaints  GROUP BY categorical\_column  ORDER BY COUNT(categorical\_column) DESC; |

Table 1 – Example Output for the `coverage\_type` column.

|  |  |
| --- | --- |
| coverage\_type | count |
| Accident and Health | 111689 |
| Automobile | 86953 |
| Homeowners | 37147 |
| Life & Annuity | 15563 |
| Miscellaneous | 9107 |
| Liability | 3554 |
| Fire, Allied Lines & CMP | 2487 |
|  | 51 |

From these queries, we were able to discover the following issues:

In multiple instances, columns had an empty string as a category. This indicates a significant number of missing values, not limited to one column. Some columns, like `keywords` and `others\_involved` have a large number of missing values. We will expand upon missing values in our Missing Values Analysis.

The columns `reason\_complaint\_filed`, `how\_resolved`, `keywords` and `others\_involved` contain multiple string (text) split by semicolons. For example, an entry into the `reason\_complaint\_filed` column could resemble Exhibit 1.4

Exhibit 1.4 - `reason\_complaint\_filed` example.

|  |
| --- |
| Agent Handling; Delays (Policyholder Service); Refund Of Premium |

Missing Value Overview.

To develop a stronger understanding of missing values we counted the number of incomplete records (Exhibit 1.4), and produced a table that returns the missing value count of all columns (Exhibit 1.5). Due to the size of the query, Exhibit 1.5 has been reduce to its essential parts.

Exhibit 1.5

|  |
| --- |
| SELECT COUNT(\*) AS `incomplete record count` FROM insurance\_complaints.complaints  WHERE  complaint\_filed\_against IN ("",NULL," ") OR  complaint\_filed\_by IN ("", NULL, " ") OR  reason\_complaint\_filed IN ("",NULL," ") OR  confirmed\_complaint in ("", NULL, " ") OR  how\_resolved IN ("", NULL, " ") OR  received\_date IN ("",NULL," ") OR  closed\_date IN ("",NULL," ") OR  complaint\_type IN ("",NULL, " ") OR  coverage\_type IN ("", NULL, " ") OR  coverage\_level IN ("", NULL, " ") OR  others\_involved IN ("", NULL," ") OR  respondent\_role IN ("",NULL, " ") OR  respondent\_type IN ("", NULL, " ") OR  complainant\_type IN ("", NULL, " ") OR  keywords IN ("", NULL, " ")  ; |

Exhibit 1.6

|  |
| --- |
| WITH empty\_table AS (  SELECT  SUM(CASE WHEN categorical\_column IN ("",NULL, " ") THEN 1 ELSE 0 END) AS `empty value count – categorical\_column`,  …  FROM insurance\_complaints.complaints)  SELECT "categorical\_column" AS `column`, `empty value count – categorical\_column` AS `missing\_value\_count` FROM empty\_table  UNION ALL  …  ORDER BY `missing\_value\_count` DESC; |

We found that there 73,743 incomplete records within our dataset – about 27% of the original dataset at this point contains incomplete records. However, a closer look into the missing value data reveals that most of the missing values are found within two optional columns: keywords and others involved. See Table 1 below.

Table 2 – Missing Value Table

|  |  |
| --- | --- |
| column | missing\_value\_count |
| keywords | 55013 |
| others\_involved | 28501 |
| how\_resolved | 1104 |
| confirmed\_complaint | 116 |
| coverage\_type | 51 |
| coverage\_level | 51 |
| reason\_complaint\_filed | 18 |
| complaint\_filed\_by | 4 |
| complainant\_type | 4 |
| respondent\_role | 2 |
| complaint\_type | 1 |
| complaint\_filed\_against | 0 |
| received\_date | 0 |
| closed\_date | 0 |
| respondent\_type | 0 |

Given that both `keywords` and `others\_involved` columns are optional, we have decided to remove them from the final descriptive analytics dataset.

Data Cleaning

Column Splitting

For the columns `how\_resolved` and `reason\_complaint\_filed`, we decided to create new tables to split up the results into multiple records.

The tables created are named `complaint\_reason\_table` and `complaint\_resolution\_table`. The `complaint\_reason\_table` contains unique pairs of complaints and reasons. Each record contains a `complaint\_id` and `respondent\_id` to represent a single complaint, and a single reason (Table 2). A complaint\_id and respondent\_id pair can appear multiple times within the table. However, each time they appear, they appear with a new unique reason. The `complaint\_resolution\_table` works in a similar fashion, only with a `resolution` column to represent the various methods used to resolve a complaint. For more information, see “Diagram 1”.

Table 3 – Preview of `complaint\_reason\_table`

|  |  |  |
| --- | --- | --- |
| complaint\_no | respondent\_id | reason |
| 1 | 13191 | Cust Service Claim Handling |
| 2 | 245 | Delays (Claims Handling) |
| 3 | 10047 | Denial Of Claim |
| 4 | 10047 | Denial Of Claim |
| 5 | 2918 | Unsatisfactory Settle/Offer |

Table 4 – Preview of `complaint\_resolution\_table`

|  |  |  |
| --- | --- | --- |
| complaint\_no | respondent\_id | resolution |
| 1 | 13191 | Other |
| 2 | 245 | Information Furnished |
| 3 | 10047 | Other |
| 4 | 10047 | Other |
| 5 | 2918 | Question of Fact |

Data Type Conversions & `complaint\_desc` table creation

Our final step to make the data ready for descriptive analysis was to build a separate table named `complaint\_desc`. This table will contain all columns from the original complaints table, excluding the columns `reason\_complaint\_filed`, `how\_resolved`, `others\_involved` and `kerywords`. The `complaint\_desc` table will also only contain complete records.

We also changed the data type of the columns `received\_date` and `closed\_date` from TEXT to DATE (Exhibit 1.6).

Exhibit 1.7

|  |
| --- |
| CAST(REPLACE(CONCAT(SUBSTRING\_INDEX(received\_date,"/",-1), SUBSTRING\_INDEX(received\_date,"/",2)),"/","") AS DATE) AS received\_date,  CAST(REPLACE(CONCAT(SUBSTRING\_INDEX(closed\_date,"/",-1), SUBSTRING\_INDEX(closed\_date,"/",2)),"/","") AS DATE) AS closed\_date, |

Diagram 1 – Complaints Descriptive Analysis Relationship Diagram

