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

# **Introduction**

The Texas Department of Insurance (TDI) is a government 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 objective of this report is to document the processes used to verify the quality of this data and build the insurance complaints dataset.

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 tables data dictionary is available in Exhibit 1.1.

Exhibit 1.1

Data Dictionary of the Original Table.

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

# **SQL Query Guide**

Throughout this report, I will refer to sections of various SQL queries. To find the SQL queries associated with each sub section, please refer to the guide below. To reproduce the results of this report, you can follow the report in linear order.

|  |  |
| --- | --- |
| Sub Section | SQL Queries |
| Uniqueness Check | duplicate-search.sql |
| Categorical Value Check | Categorical-columns-overview.sql |
| Missing Value Overview | incomplete-record-count.sql,  missing-value-count.sql |
| Column Splitting | how\_resolved\_splitting.sql, reasons\_complaint\_filed\_splitting.sql, |
| Date Type Conversions & Descriptive Complaints Table Creation | complaint\_desc\_creation.sql |

# **Database Upload Process**

I downloaded the data from the website Data.gov on March 21 2025. Then I imported the data into SQL Server, using the Table Data Import Wizard. SQL Server 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 table, I checked if the composite primary key appeared more than once. To achieve this goal, I used the SQL query in Exhibit 1.2. The query returned no records, indicating no “complaint\_no” – “respondent\_id” pairs appear more than once within the table.

Exhibit 1.2 – Duplicate Check SQL Query.

|  |
| --- |
| SELECT  coverage\_type,  COUNT(coverage\_type) AS "count"  FROM insurance\_complaints.dbo.complaints  GROUP BY coverage\_type  ORDER BY COUNT(coverage\_type) DESC; |

## Categorical Value Check

For each categorical column, I created a query (Exhibit 1.3) to return a table of all distinct values, and the number of times these values occurred. This helped me quickly identify if there were any data entry related errors within the dataset.

Exhibit 1.3 – Column Categories Check

|  |
| --- |
| SELECT  categorical\_column,  COUNT(categorical\_column) As “count”  FROM insurance\_complaints.dbo.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 |

Through this column check, I discovered there were NULL values and non-atomic values within multiple columns. In this case, non-atomic values contained multiple strings separated by semicolons (Exhibit 1.4).

Exhibit 1.4 – non-atomic value example.

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

## Missing Value Overview.

To develop a stronger understanding of missing values, I counted the number of incomplete records (Exhibit 1.5), and found the number of missing values for each column (Exhibit 1.6). Due to the size of the query, Exhibit 1.6 has been reduce to its essential parts.

Exhibit 1.5 – Count of Incomplete records.

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

Exhibit 1.6 – Count of Incomplete Records by Column

|  |
| --- |
| 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; |

I found that there were 73,743 incomplete records (27% of records) within the ‘complaints’ table based upon the query in Exhibit 1.5. 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 2 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 |

Exhibit 1.7 – Count of Incomplete Records in the “keywords” and “others\_involved” columns

|  |
| --- |
| SELECT  COUNT(\*)  FROM insurance\_complaints.dbo.complaints  WHERE keywords IS NULL OR others\_involved IS NULL; |

Using the query in Exhibit 1.7, I found that 73092 records had an empty “keywords” or ‘others\_involved’ column. Given that both “keywords” and “others\_involved” columns are optional, I have decided to remove them from the final descriptive analytics table.

# **Data Modeling**

## The Non-Atomic Data Problem

In the *Categorical Column Check* section, I found several columns (“how\_resolved”, “reason\_complaint\_filed”, “keywords”, and “others\_involved”) contained non-atomic values (see Exhibit 1.4). These columns violate first normal form, and obstruct all analysis involving these columns.

For example, imagine an analyst needed to answer the question “What is the most common reason behind consumer complaints?” The current ‘complaints’ dataset would provide the answer seen in Table 3.

Exhibit 1.7 – Top 10 Reasons using the ‘complaints’ table

|  |
| --- |
| SELECT TOP 10  reason\_complaint\_filed,  COUNT(reason\_complaint\_filed) AS "reason\_count"  FROM insurance\_complaints.dbo.complaints  GROUP BY reason\_complaint\_filed  ORDER BY COUNT(reason\_complaint\_filed) DESC; |

Table 3 – Exhibit 1.7 query output.

|  |  |
| --- | --- |
| value | count |
| Denial Of Claim | 26515 |
| Delays (Claims Handling) | 26354 |
| Unsatisfactory Settle/Offer | 21030 |
| Cust Service Claim Handling; Delays (Claims Handling) | 19131 |
| Balance Billing; Unsatisfactory Settle/Offer | 17577 |
| Delays (Claims Handling); Unsatisfactory Settle/Offer | 10829 |
| Cust Service Claim Handling; Delays (Claims Handling); Unsatisfactory Settle/Offer | 8050 |
| Delays (Policyholder Service) | 6934 |
| Denial Of Claim; Unsatisfactory Settle/Offer | 6634 |
| Premium and Rating | 5070 |

Non-atomic values are highlighted in red.

The problem is, the output returned only provides a list of values, not reasons.

Values, in this context, refers to a string. This can range from a single phrase that represents one reason/resolution (i.e. Denial of Claim) to a series of phrases in a single string (i.e. Denial of Claim; Unsatisfactory Settle/Offer). A reason/resolution refers to a single phrase, like ‘Denial of Claim’ or ‘Delays (Claims Handling)’.

Because the values are non-atomic, the query in Exhibit 1.7 cannot provide any information on any specific reason/resolution. To find specific information about the reason ‘Denial of Claims’, each value needs to be split into multiple rows, each representing a reason associated with a specific complaint.

## The Incomplete Record Problem

Analysts cannot use incomplete records, unless the columns with the missing data are not important to their analysis, like the “keywords” and “others\_involved” columns. Therefore, it is important to isolate incomplete records, without entirely removing them from the dataset.

## A New Data Model

To solve the non-atomic data and incomplete record problems, I decided to add new tables that address both issues.

Table 4 – Tables in Data Model.

|  |  |
| --- | --- |
| Name | Description |
| complaints\_desc | Contains all complete records from the original dataset without the columns “how\_resolved”, “reason\_complaint\_filed”, “others\_involved”, and “keywords”. |
| complaint\_reason\_table | Each record represents a distinct complaint and a specific phrase cited amongst the reasons for filing the complaint. |
| complaint\_resolution\_table | Each record represents a distinct complaint and a specific method cited in how the complaint was resolved. |
| complaints\_inc | Similar to ‘complaints\_desc’, but contains incomplete records. |
| complaint\_reason\_table\_inc | Similar to ‘complaints\_reason\_table\_inc’, but stores information about incomplete records. |
| complaint\_resolution\_table\_inc | Similar to ‘complaints\_resolution\_table\_inc’, but stores information about incomplete records. |

Each ‘complaint\_reason\_table’ and ‘complaint\_how\_resolved\_table’ contains a distinct complaint\_no, respondent\_id, and reason/resolution. For example, if a distinct complaint has three reasons, the complaint would appear three times within the ‘complaint\_reason\_table’.

Table 5 – Example of a Complaint with Multiple reasons.

|  |  |  |
| --- | --- | --- |
| complaint\_no | respondent\_id | reason |
| 29 | 6760 | Agent Handling |
| 29 | 6760 | Delays (Policyholder Service) |
| 29 | 6760 | Pre-Existing Condition |

The same logic applies to the ‘complaint\_resolution\_table’, ‘complaint\_reason\_table\_inc’, and ‘complaint\_resolution\_table\_inc’.

# **Complete Dataset Diagram**

Having completed the creation of the descriptive dataset, I have created a diagram of relationships that existed in the database. This will help analysts understand what relationships currently exist in the database.

|  |  |
| --- | --- |
| **dbo.complaints\_desc** | |
| PK | complaint\_no |
| respondent\_id |
|  | complaint\_filed\_against |
|  | complaint\_filed\_by |
|  | … |

|  |  |
| --- | --- |
| **dbo.complaints** | |
| PK | complaint\_no |
| respondent\_id |
|  | complaint\_filed\_against |
|  | complaint\_filed\_by |
|  | … |

|  |  |
| --- | --- |
| **dbo.complaint\_resolution\_table** | |
| PK | complaint\_no |
| respondent\_id |
| resolution |

|  |  |
| --- | --- |
| **dbo.complaint\_reason\_table** | |
| PK | complaint\_no |
| respondent\_id |
| reason |

1

∞

∞

1

∞

1

|  |  |
| --- | --- |
| **dbo.complaints\_inc** | |
| PK | complaint\_no |
| respondent\_id |
|  | complaint\_filed\_against |
|  | complaint\_filed\_by |
|  | … |

|  |  |
| --- | --- |
| **dbo.complaint\_resolution\_table** | |
| PK | complaint\_no |
| respondent\_id |
| resolution |

|  |  |
| --- | --- |
| **dbo.complaint\_reason\_table\_inc** | |
| PK | complaint\_no |
| respondent\_id |
| reason  1  ∞ |