Data Curation Project Report

The report starts from the perspective of datasets and deals with data quality issues in the process of data integration. Various data quality issues found in several datasets are analyzed, and cleaned in combination with business conditions.

# Chicago Public Schools - School Profile Information SY1617

## Dataset Introduction

**Domain**: School profile information in the Chicago (Education)

School profile information for all schools in the Chicago Public School district for the school year 2016-2017.

Rows: 661 Columns: 91 Each row is a school.

The specific information about the columns in the dataset can be found at the following URL.

[https://data.cityofchicago.org/Education/Chicago-Public-Schools-School-Profile- Information-/8i6r-et8s](https://data.cityofchicago.org/Education/Chicago-Public-Schools-School-Profile-Information-/8i6r-et8s)

## Data Quality Problems

**Problems:**

The main problems with this dataset include null values, missing values, outlier values and function dependency.

**Tools and Methods for Problem Identification:**

All of the above issues are addressed by implementing a set of shared functions to identify and inspect problems in the dataset.

This involves using SQL for data display, Python's Pandas library for detection, and Python Bokeh library for visualizing problematic data.

* **Null values**

null values are determined using the isnull method in Pandas, and the count is calculated using pd.isnull().sum(). Figure 1-1 illustrates the number of null values in each column using Bokeh

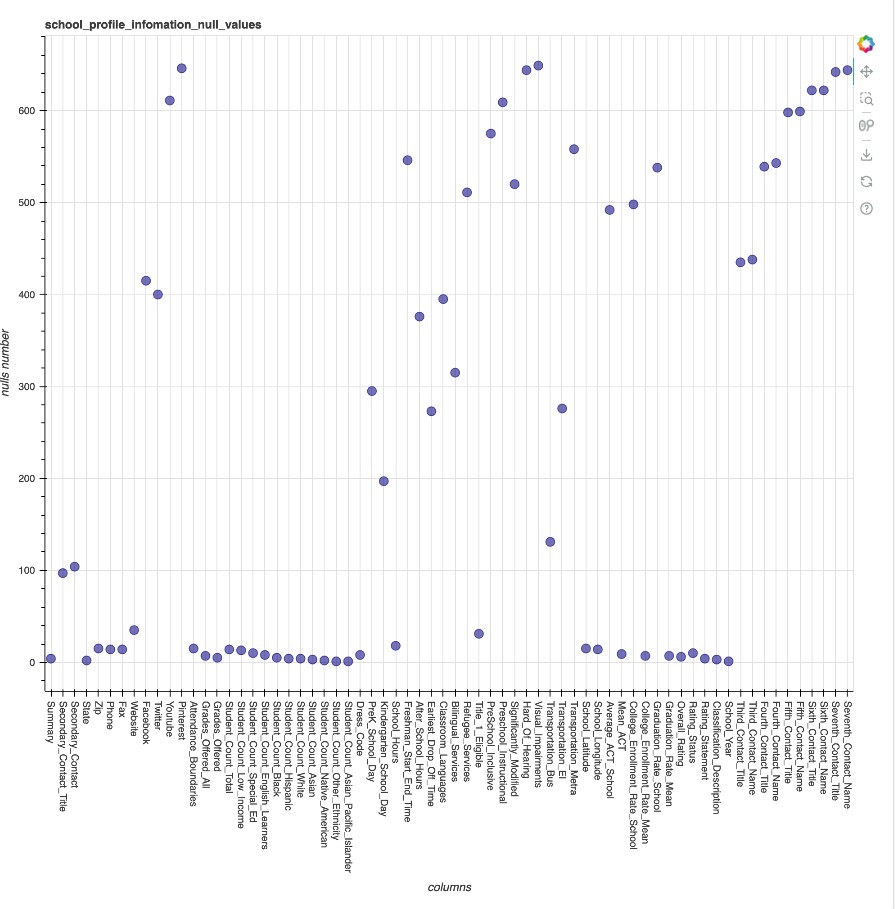


Figure 1-1

* **Missing values**

Missing values are determined for columns of data type 'number' using Pandas' isNull method, and the count is calculated using pd.isNull().sum(). The figure , Figure 1-2 below, illustrates the proportion of missing values in each column using Bokeh.

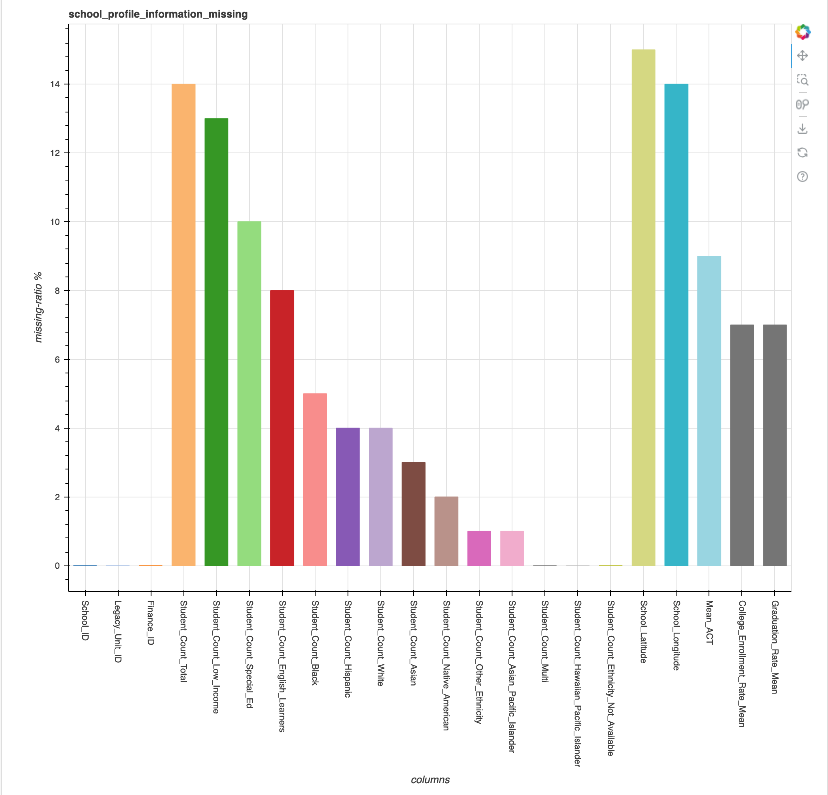


Figure 1-2

* **Outlier values**

During data processing, data beyond three standard deviations (probability less than three in a thousand) are considered as outlier data according to the properties of the normal distribution. Figure 1-3 below, illustrates the number of outlier values in each column using Bokeh.

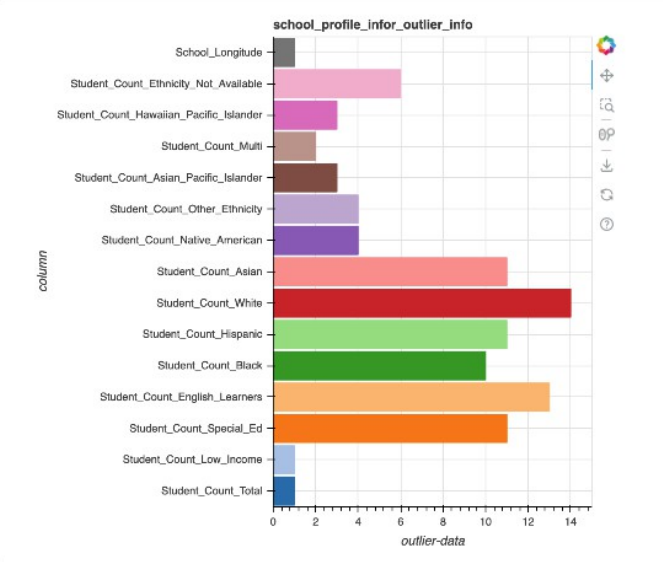


Figure 1-3

* **Function dependency**

Matching the schema between "Chicago Public Schools - Progress Report Cards (2011-2012)" and "Chicago Public Schools - School Profile Information SY1617," the field name\_of\_school from the former corresponds to the long\_name field in the latter,where long\_name -> city,state . To link these, an SQL query was executed to find matching long\_name = school\_of\_name. However, discrepancies were found where city<> city and state <> state in the data.

The figure below, Figure 1-4, illustrates the functional dependency problem where schools in Chicago have the same school name but different city names, presented using Sql.

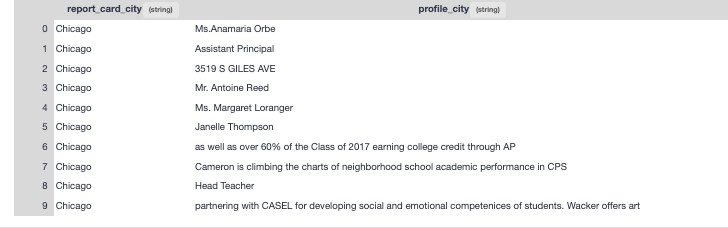


Figure 1-4

The figure 1-5 below, illustrates the functional dependency problem where schools in Chicagohave the same school name but different state names, presented using Sgl.

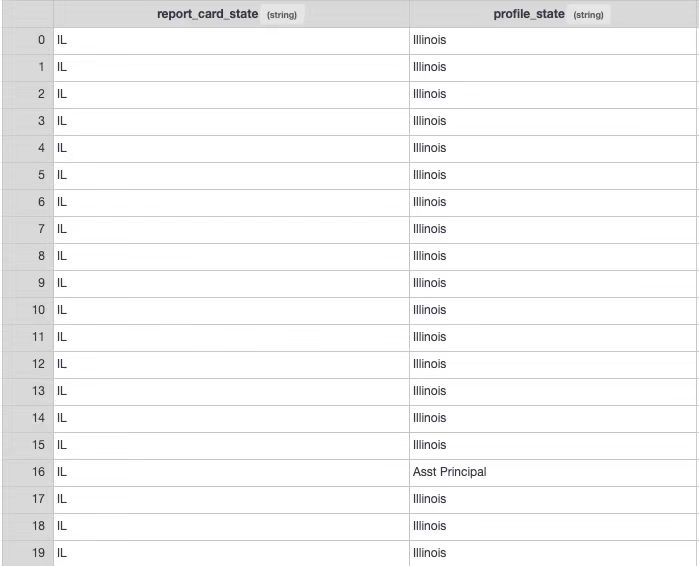


Figure 1-5

* **Diversity of phone formats**

In the data, there are two formats of mobile phone numbers. such as: "7735341215" and "(773) 535-4580". This will also cause them to be parsed into different data types. For example, the image below Figure 1-6.

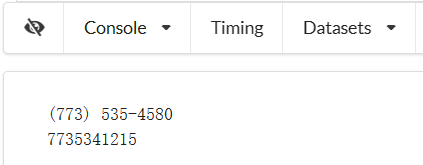


Figure 1-6

## Solution

* **Null values**

Dropping a column if the percentage of null data exceeds a certain threshold. The dropped column will not be involved in our data integration.

The Figure below Figure 1-7, illustrated using Bokeh, displays columns with null values exceeding the set percentage. After observing the data, a drop operation is performed.

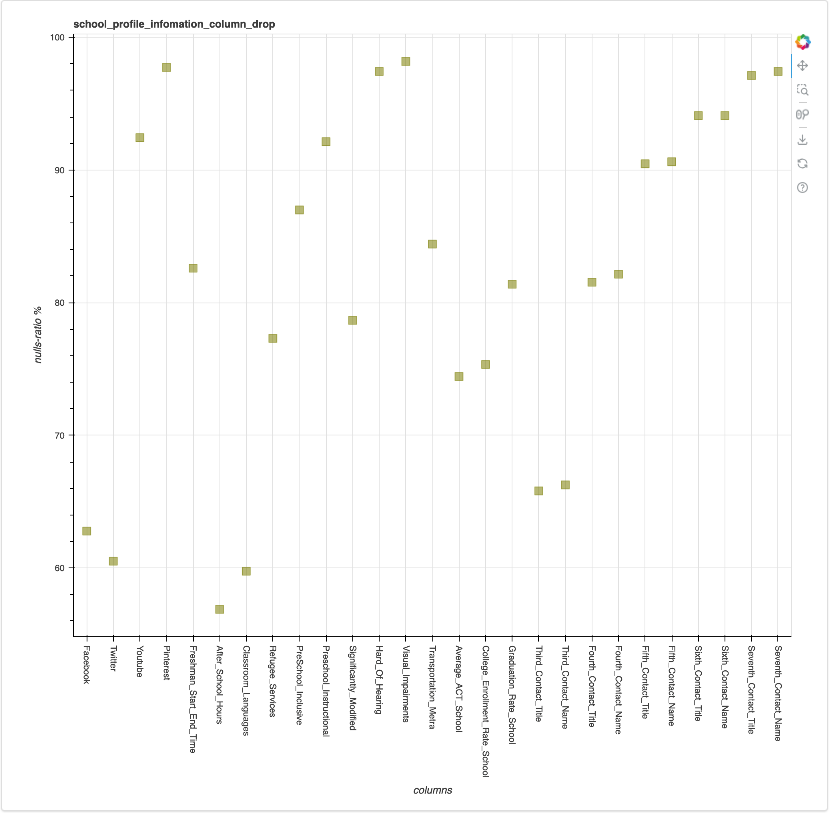


Figure 1-7

* **Missing values**

For numerical columns with missing values, utilize Vizier's "impute missing value" functionality to fill in the mean. Figure 1-8 using Bokeh illustrated the missing values after being processed with mean imputation. It can be observed that, apart from latitude and longitude (which cannot be handled using the mean imputation method and require map data filling), all other missing values have been filled with the mean."

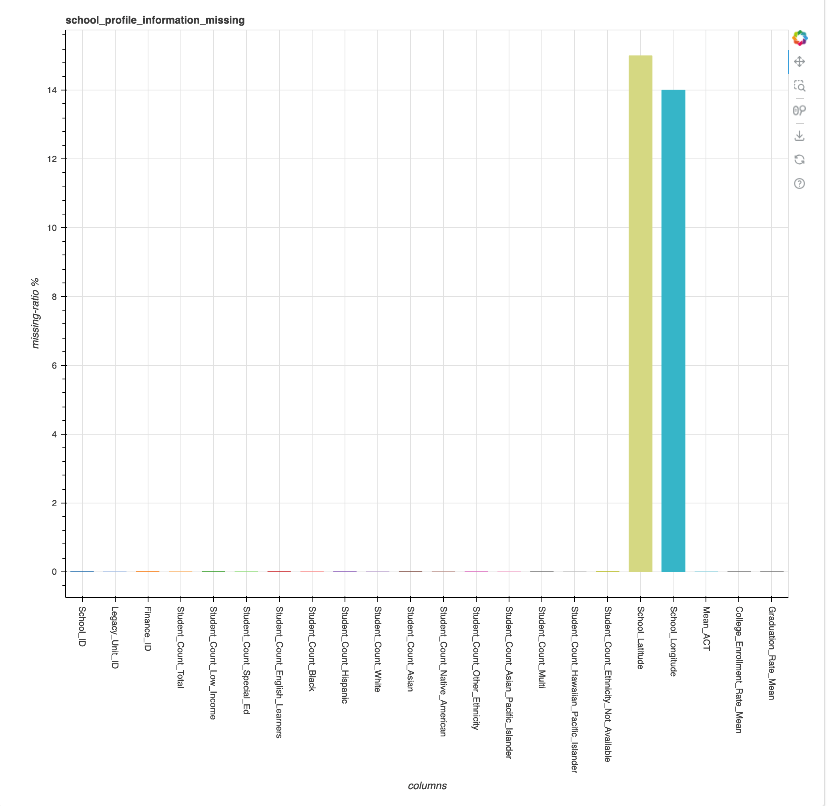


Figure 1-8

* **Outlier values**

Replace the outliers value with the column mean.

Figure 1-9 illustrates the outliers in a specific column 'student\_count\_total’ This column's outlier value is 7735355725 and it is handled by filling with the mean. The same approach is applied to other columns.

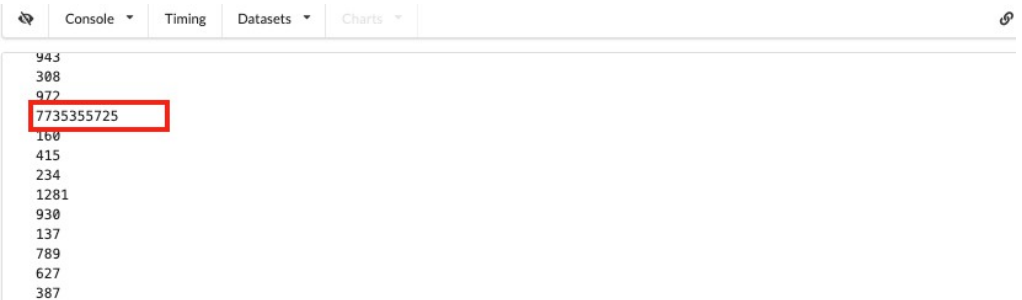


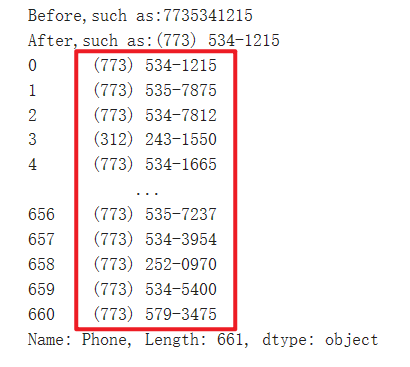
Figure 1-9

* **Function dependency**

After observing the data, update incorrect data with the correct values.

* **Diversity of phone formats**

Unified into a more readable format.



## The encountered issues

While addressing the aforementioned issues, difficulties were encountered in converting a Pandas DataFrame to a dataset, resulting in errors.

The problem with data was resolved by converting the data using SQL statements. However, when executing dataset value updates in Vizier's Python script, the process was exceptionally slow, and a satisfactory solution has not yet been identified.

# Chicago Public Schools - Progress Report Cards

（**2011-2012**）

## Dataset Introduction

**Domain**: Chicago Public Schools - Progress Report Cards(2011-2012) (Education)

This dataset shows all school level performance data used to create CPS School Report Cards for the 2011-2012 school year.

Rows: 566 Columns: 79 Each row is a school

The specific information about the columns in the dataset can be found at the following URL.

[https://data.cityofchicago.org/Education/Chicago-Public-Schools-Progress-Report-Cards- 2011-/9xs2-f89t](https://data.cityofchicago.org/Education/Chicago-Public-Schools-Progress-Report-Cards-2011-/9xs2-f89t)

## Data Quality Problems

**Problems:**

The problems with this dataset include null values, missing values, and outlier values， The problems with individual datasets have been mentioned in the dataset descriptions above and will not be reiterated here.

The purpose of using this dataset is to perform schema matching and data matching with the dataset1 mentioned earlier. These two operations are crucial in the preprocessing stage of schema mapping for the data integration.

Figure 2-1 illustrates the generation of a similarity matrix through the implementation of the Jaccard Similarity algorithm for schema matching.

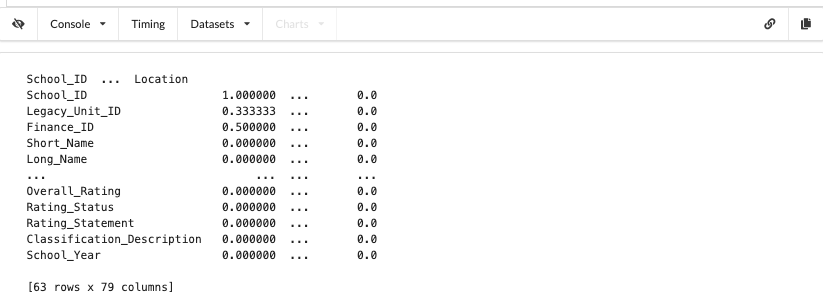


Figure 2-1

Figure 2-2 illustrates that following the schema matching in the previous step, the 'name\_of\_schools' in Dataset 1 matches with the 'long name' in Dataset 2. This graph represents the data matching process for the non-matching data between the two datasets.

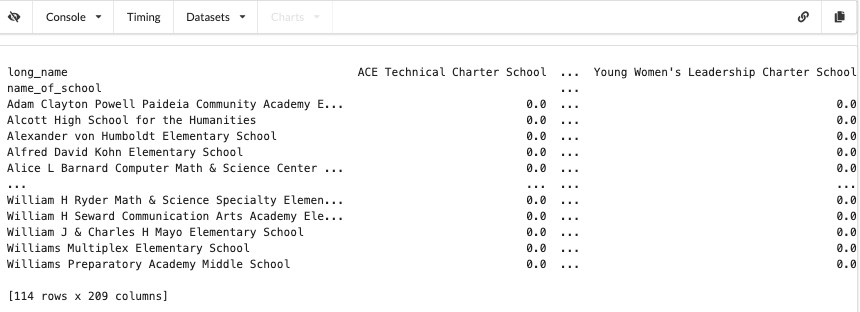


Figure 2-2

**Tools and Methods for Problem Identification：**

We've implemented the Jaccard Similarity algorithm in Python to detect schema mapping between two datasets and generated the Similarity Matrix for these two schemas using Pandas. the Jaccard Similarity algorithm in Python for detecting schema mapping between two datasets and creating the Similarity Matrix for these two schemas using Pandas.

## Solution

* **Schema Matching:**

After obtaining the Similarity Matrix mentioned in the previous issues, we need to identify a match selector. We choose a value between 0 and 1 as the threshold value. In our schema matching,

we set it to 0.2. Before performing the matching, we utilize regular expressions for tokenization and handle stop words to exclude interference and enhance the accuracy of matching.

This preprocessing step is aimed at improving the precision of the matching process and then obtaining the match selector.

Figure 2-3 illustrated the match selector after performing schema matching

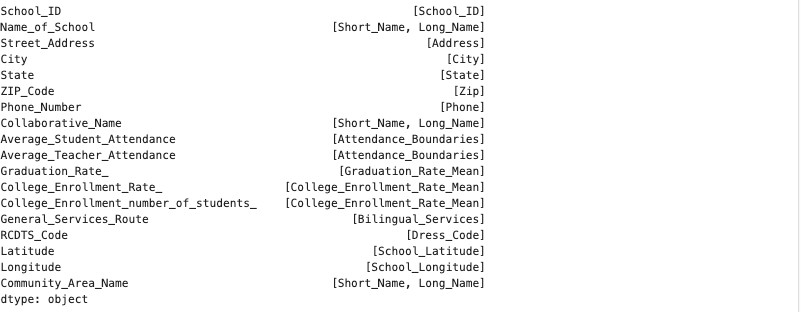


Figure 2-3

* **Data Matching**

After completing schema matching, it was observed that the 'name\_of\_school' in the School Progress Report dataset matches with the 'long\_name' in the School Profile Information dataset.

Upon connecting the two tables using SQL, it was noticed that the number of matching schools on both sides was not consistent. Subsequently, through left and right joins, the schools that didn't match were identified.

Using the Jaccard Similarity algorithm in Pandas, efforts were made to discover as many matching data points as possible.

Figure 2-4 displays the school names after matching the non-matching data between the two datasets.

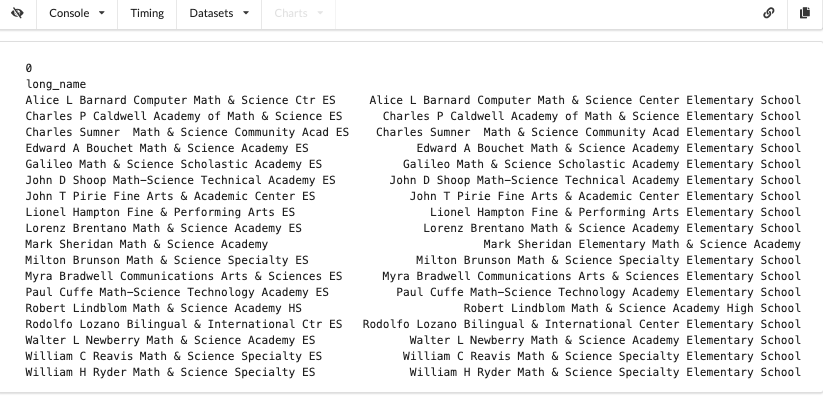


Figure 2-4

# CPS-School\_Progress\_Reports\_SY1617

## 3.1 Dataset Introduction

**Domain:**

2016 school progress report ratings for all Chicago Public Schools.

Locations of educational units in the Chicago Public School District for school year 2016-2017. To view or use these shapefiles, compression software, such as 7-Zip, and special GIS software, such as Google Earth or ArcGIS, are required.

**Data scale:**

CPS\_School\_Progress\_Reports\_SY1617: Rows: 661 Columns: 161 Each row is a school.

CPS\_School\_Locations\_1617: Rows: 670 Columns: 16 Each row is a school.

**Download:**

CPS\_School\_Progress\_Reports\_SY1617: <https://data.cityofchicago.org/Education/Chicago-Public-Schools-School-Progress-Reports-SY1/cp7s-7gxg>

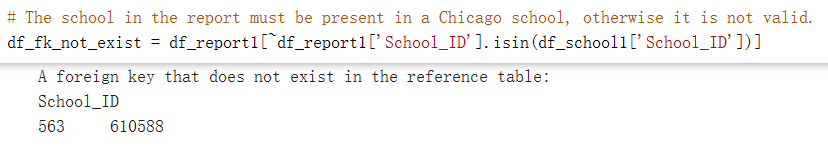
CPS\_School\_Locations\_1617: <https://data.cityofchicago.org/Education/Chicago-Public-Schools-School-Locations-SY1617/9zky-nrsy>

## 3.2 Data Quality Problems

**Problems:**

* **Data dependency relation error**

In the referenced table, there is data that does not exist. We hope to be able to link and query school information on the map.



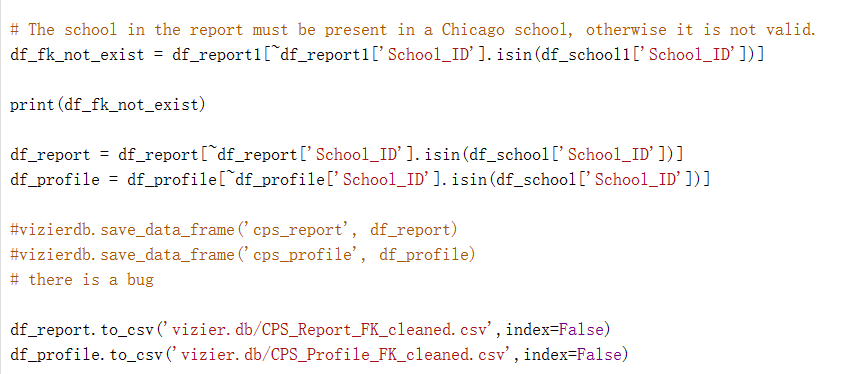
**Tools and Methods for Problem Identification:**

According to the business meaning of the field, we found that no data was returned when querying two tables through the foreign key field join, and the data of the two tables was sampled, and the official data of the collection was introduced, and we identified that there may be foreign key constraint integrity problems in the data set.

## 3.3 Solution

* **Data dependency relation error**

Remove invalid data. Save new data.



# Traffic Crashes - Vehicles and People

## Dataset Introduction

**Domain**: Traffic Crashes - Vehicles and People in the Chicago (Transportation)

This dataset contains information about vehicles and People (or units as they are identified in crash reports) involved in a traffic crash,Last Updated November 18, 2023.

**Vehicles table**: Rows: 1.59M Columns: 72 Each row is a Vehicle.

**People table**: Rows: 1.72M Columns: 30 Each row is a Person.

The specific information about the columns in the dataset can be found at the following URL.

**Vehicles**: <https://data.cityofchicago.org/Transportation/Traffic-Crashes-Vehicles/68nd-jvt3>

**People**: <https://data.cityofchicago.org/Transportation/Traffic-Crashes-People/u6pd-qa9d>

## Data Quality Problems

**Problems：**

After a column with a unique primary key is defined, the data appears to have a unique primary key duplicate

**Tools and Methods for problems identification:**

By grouping data and calculating the number of data rows to find the problem of duplicate data, using SQL functions can help us solve the problem of duplicate data.

According to the official introduction of data meaning, naming rules, and analysis of query date-related fields and sampled values, it is known that date types need to be processed in a unified manner.

* **Duplicate the unique primary key**

Solve the problem by using SQL's distinct functions and some data association queries

Figure 3.1 , Is the result of statistics when the data repeats the primary key (Red box marks statistical quantity)

Figure 3.2 , A unique primary key duplication problem was discovered using SQL.

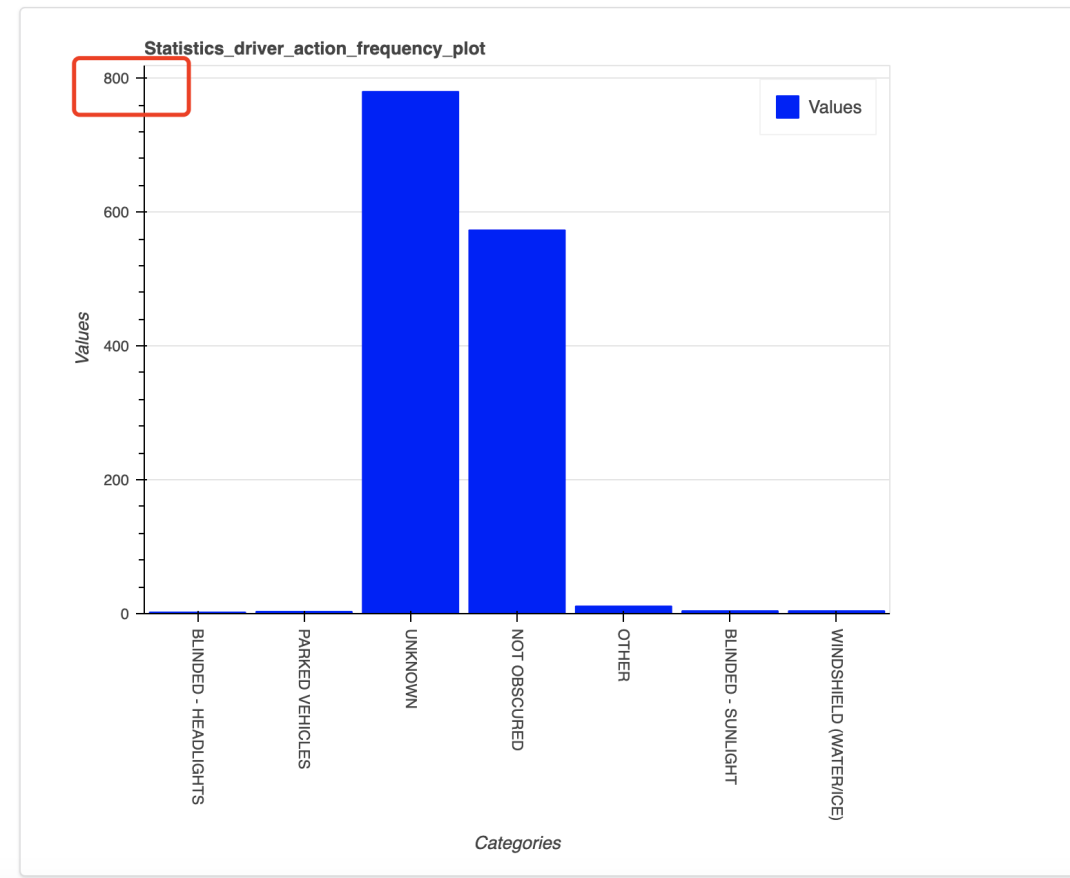


Figure 3.1



Figure 3.2

* **The date format is recognized**

If we look at the date format, it is not recognized as a date in Excel, although pandas can recognize it, such as the field crash\_date, but we still want to unify the date into a common format in order to avoid problems.

And because this field does not have a time zone, we may face a time zone issue when querying Chicago data in the Beijing time zone of China.

There are also year values that are recognized as floating-point numbers, such as the field vehicle\_year, which can cause problems with the data not being queried when querying.

For example, Figure 3.3 compare with Figure 3.4, Figure 3.3 shows the dataset read by zivier, Figure 3.4 shows the DataFrame read by Python Pandas. They present different date values.

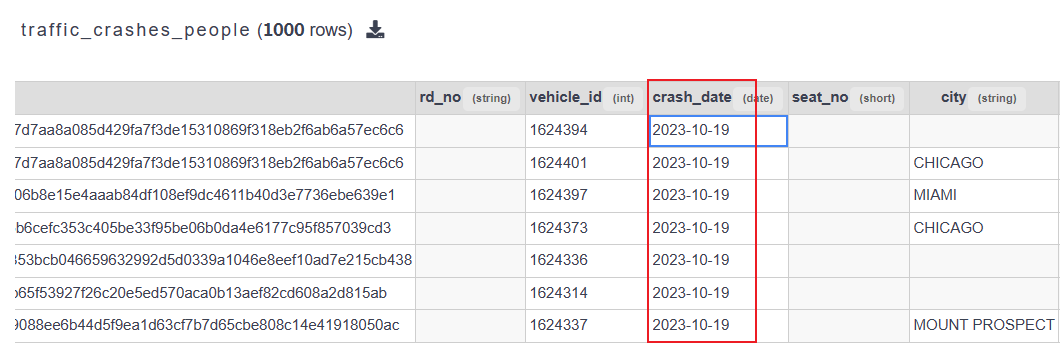
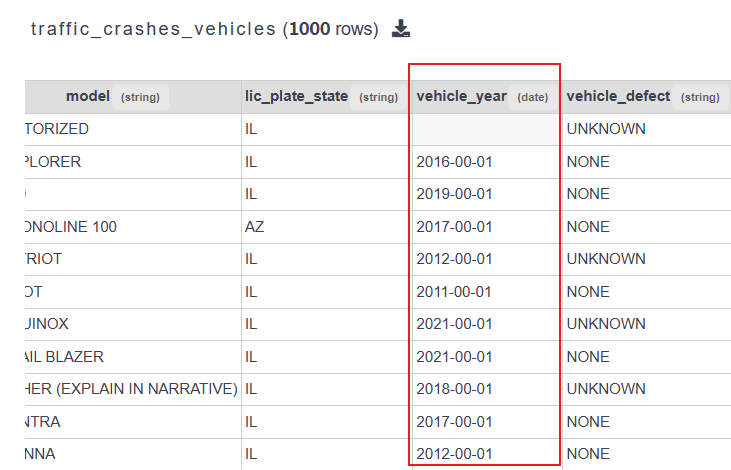


Figure 3.3

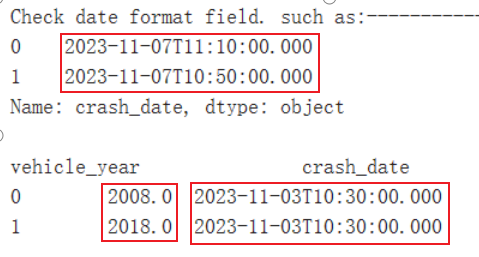


Figure 3.4

* **The data structure is abnormal**

A column person\_id in the table Traffic\_Crashes\_People contains multiple columns of information from other tables, making it impossible to join queries directly. Figure 3.5 shows the columns description, Figure3.6 shows the person\_id values.

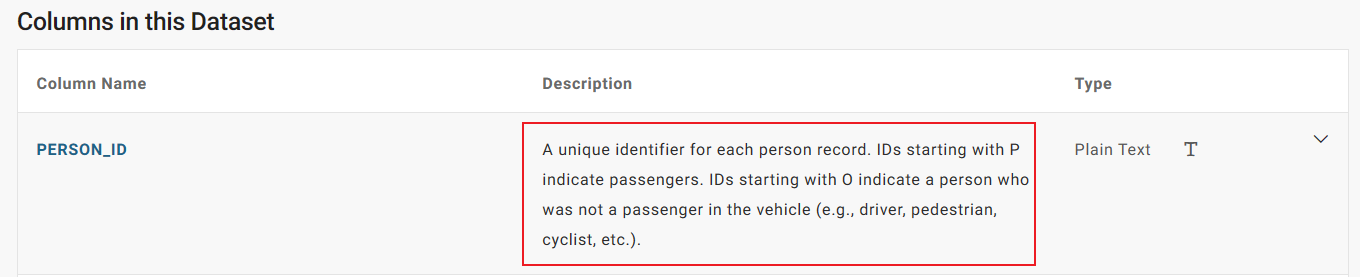


Figure 3.5

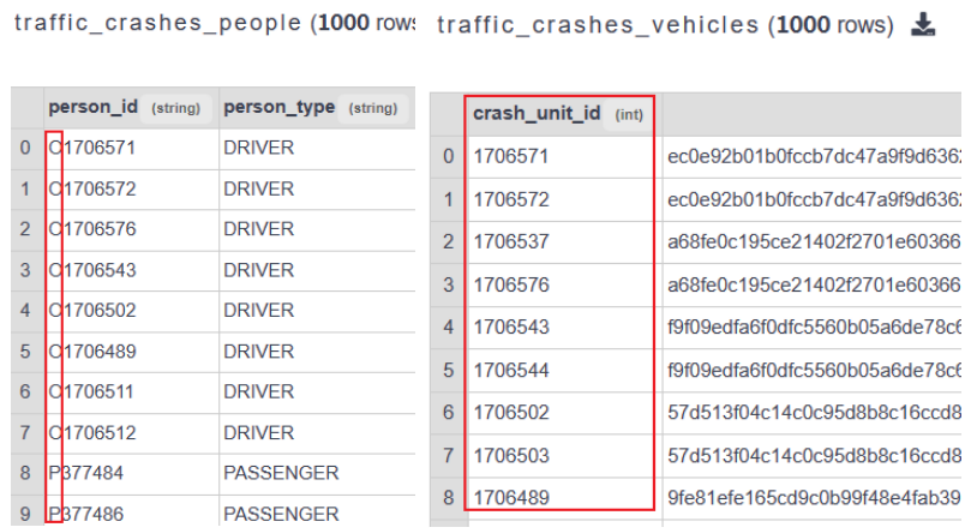


Figure 3.6

## Solution

* **Duplicate the unique primary key**

Using the distinct function to filter duplicate data and then using table association can get the correct data result set.

Figure 3.7 The result data is collected after removing the duplicate unique primary key.

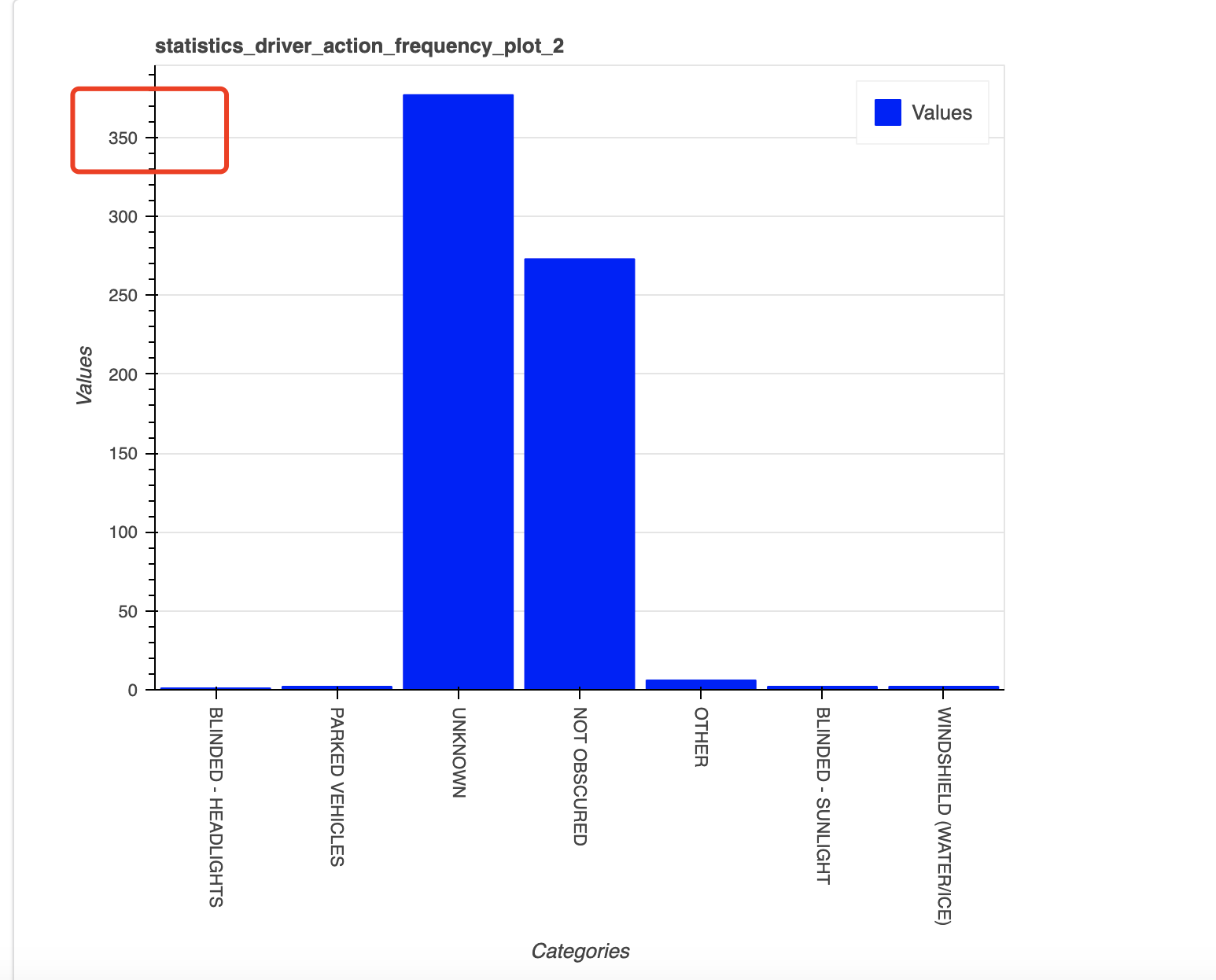


Figure 3.7

* **The date format is not recognized.**

For date format issues in data, the date format is converted uniformly. Figure 3.8 shows the corrected data format.

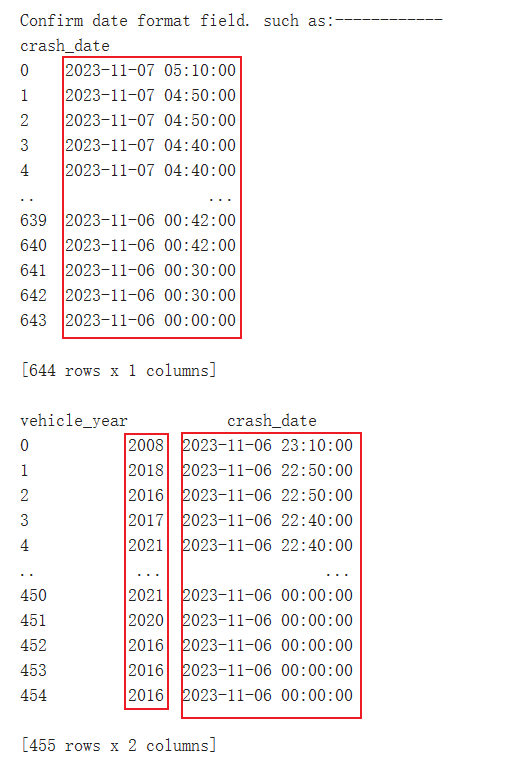


Figure 3.8

* **The data structure is abnormal**

Extract the split values according to the rules, splitting one column into two columns. Figure 3.9 shows the changes before and after the column split.

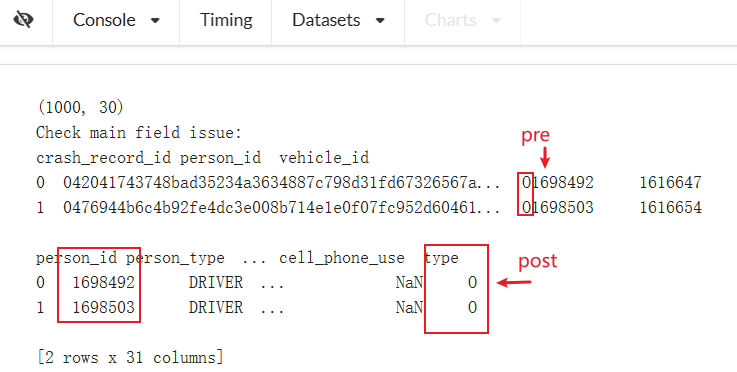


Figure 3.9

## The encountered issues.

By comparing the data, we find that only one data in the traffic accident-vehicle data table will be recorded, while more than one data may be recorded in the traffic accident-personnel table. This makes the unique primary key we defined (CRASH\_RECORD\_ID) duplicate when combining data, so a filtering operation is required.

# Business Licenses and Business Licenses - Current Active

## Dataset Introduction

**Domain**: Business Licenses and Business Licenses - Current Active (Community & Economic Development).

**Business Licenses - Current Active:** This dataset contains all current and active business licenses issued by the Department of Business Affairs and Consumer Protection.

**Business Licenses:** Business licenses issued by the Department of Business Affairs and Consumer Protection in the City of Chicago from 2002 to the present.

**Business Licenses:** Rows: 1.12M Columns: 34

**Business Licenses - Current Active:** Rows: 53.6K Columns: 34

The specific information about the columns in the dataset can be found at the following URL.

**Business Licenses - Current Active:** <https://data.cityofchicago.org/Community-Economic-Development/Business-Licenses-Current-Active/uupf-x98q>

**Business Licenses:** <https://data.cityofchicago.org/Community-Economic-Development/Business-Licenses/r5kz-chrr>

## Data Quality Problems

**Problems:**

A data error occurs during data redundancy.

**Tools and Methods for problems identification:**

* **Redundant data error**

Redundant data errors can be found by comparing the data table with the wrong data table with the correct data. Correlate and populate the right data with the right data tables.

Figure 5.1 , Check the status of the business license with area number 18.

Figure 5.2 , Problem data is found. Comparing the redundant data on the left (red box is divided around) with the latest data on the right, it is found that many business license information under the same address does not correspond.



Figure 5.1

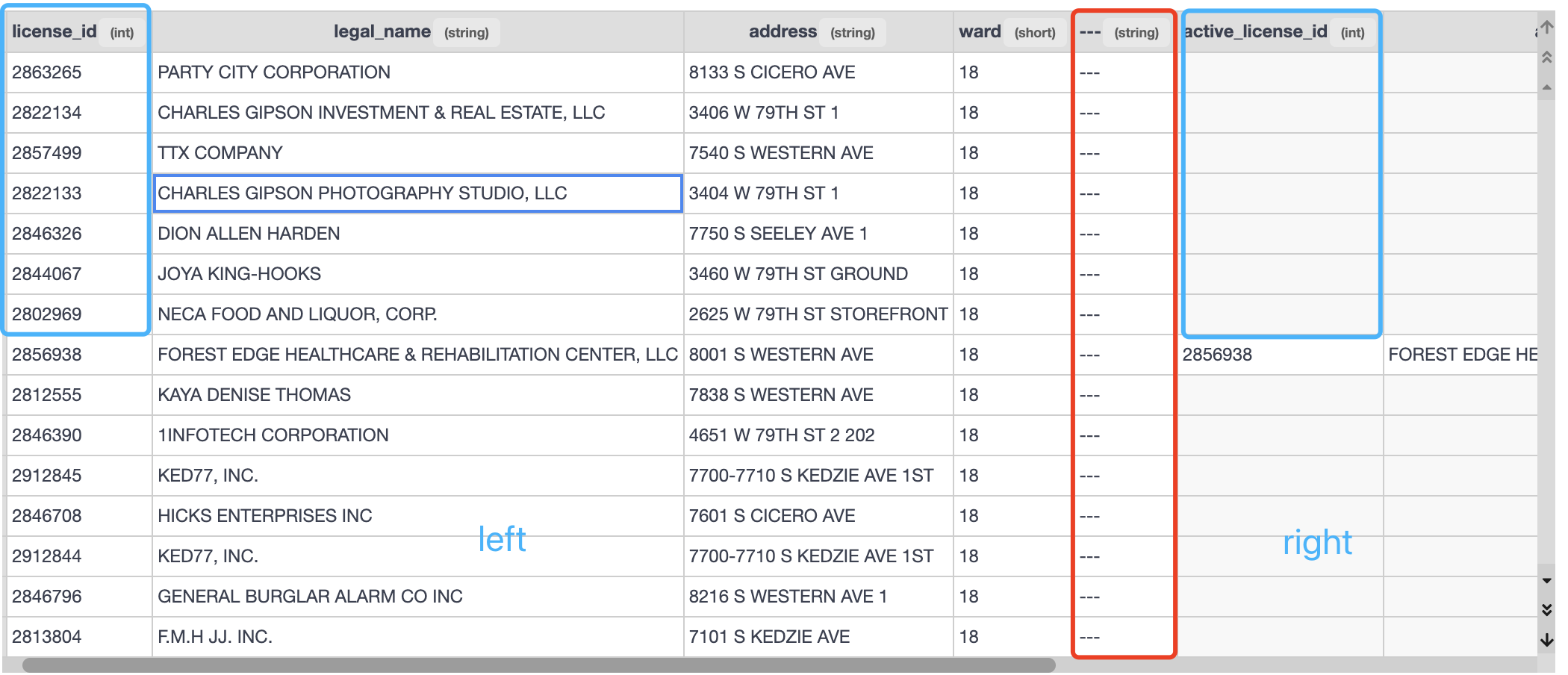


Figure 5.2

## Solution

* **Redundant data error**

Populate the data with a data table with the correct data.

Figure 5.3, The result of data cleaning.

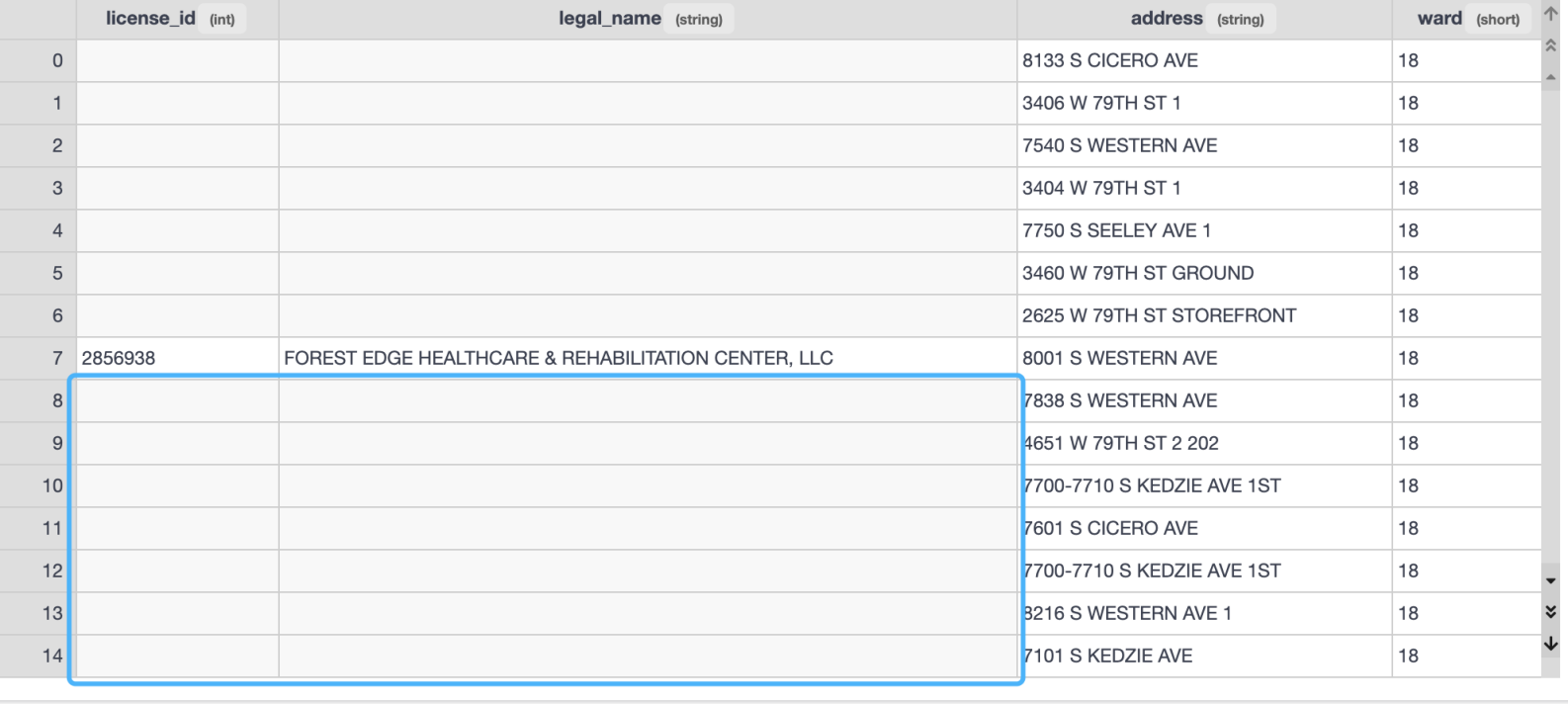


Figure 5.3

## The encountered issues

Because the business license data corresponding to the address in a region will change, the business license corresponding to the address will also change. Because the redundant data in the business license table is not updated, the corresponding address may be wrong. In the merged data, it is necessary to use the correct data table for data filling.