**Midterm Report**

Chinmay Deshpande 002859266

Sushil Girish 002817666

Shreya Thakur 002818444

**Part 01**

**Data Profiling with y-data Profiling and Staging the datasets in Talend**

Chicago Dataset Statistics

The Chicago Dataset is analysed and visualised using a variety of data points, all of which are included in this report's analytical summary. It is vital to comprehend the structure and quality of the dataset in order to utilise it efficiently inside the business model.   
Over 26,791 observations, 17 variables are included in the dataset. This size suggests a rather extensive set of data points appropriate for in-depth examination.

**Data Quality:** Data Quality: Approximately 1.8% of the dataset is comprised of 8,399 missing cells. This indicates a high-quality dataset in terms of completeness, as there is a very small amount of missing data. Interestingly, there aren't any duplicate rows, indicating that each observation's uniqueness was preserved during the data collecting procedure.

**Memory Usage:** Memory Usage: The dataset has an average record size of 144.0 B and a total size of 3.7 MiB in memory. These numbers imply that most contemporary computer systems should be able to handle the dataset without the requirement for specialised data processing methods. Five of the variables are numeric, six are text, five are categorical, and one is datetime. Due to the diversity of data formats, several data pretreatment techniques will be required to guarantee that the dataset is prepared appropriately for any processing or analysis.  
**Possible Consequences:** It is unlikely that the small proportion of missing data will have a major effect on the reliability of machine learning models or statistical analysis. Nevertheless, depending on the analysis's needs, attention should still be made to deal with these missing values effectively, either via imputation or exclusion. It is advantageous because there are no duplicate rows, which lessens the requirement for initial data cleansing.  
It is advised to look into the pattern of the missing data before moving on with the analysis to see if it is a random omission or if there is a systematic problem that needs to be fixed. If appropriate, think about filling in the missing data with imputation techniques. Furthermore, confirm that each variable's data type corresponds to the format needed for the planned analysis.

**Conclusion:** In summary, the dataset seems to be of excellent quality, with few missing values and no duplicate entries. Because of its moderate size, analysis on standard computing systems shouldn't encounter any major difficulties. The dataset is ready for a comprehensive examination or usage in machine learning applications after the few missing values have been handled correctly.

Dallas Dataset Statistics

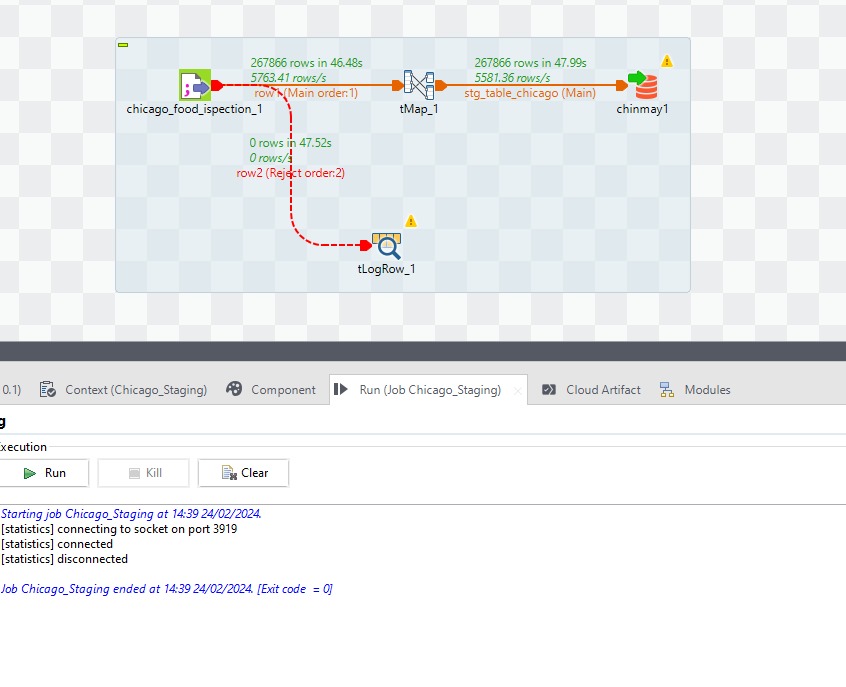
The study that is offered offers a thorough examination of the Dallas dataset that includes information pertinent to the food industry. The report includes a wide range of factors designed to address any discrepancy found in the dataset. With 114 variables spread across 78,400 observations, the dataset offers a sizable amount of data for analysis.

**Data Quality:** The existence of 64,543,27 missing cells, or 72.2% of the dataset, is a serious problem for data quality. Given the significant percentage of missing data, urgent action is needed. 42 duplicate rows are also found in the dataset, however they only make up 0.1% of the total.

**Memory Usage:** The dataset has an average record size of 912.0 B and takes up 68.2 MiB in memory. Understanding the memory needs for processing and analysis depends on this information.  
**Possible Consequences:** The significant proportion of missing data makes statistical analysis difficult and could necessitate the adoption of imputation techniques or a review of data gathering strategies. The dataset appears to be rather huge based on the memory utilisation, and working with it properly may need the use of appropriate data handling techniques.  
Our recommendation is to look into the reasons behind the missing data and utilise suitable imputation techniques in order to enhance the dataset's integrity. To make sure that every observation is distinct, the duplicates should be looked at and, if needed, eliminated.

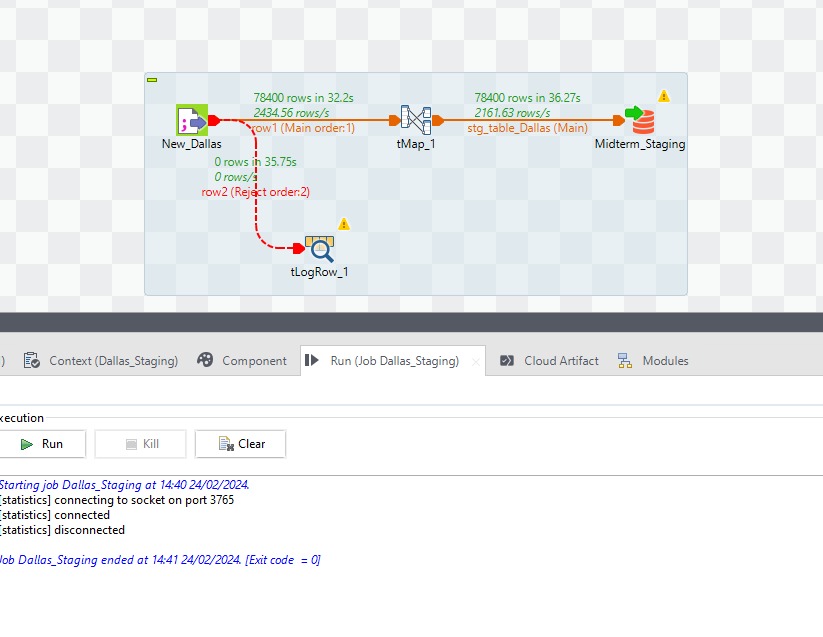
**Conclusion:** There are many potential and difficulties with the dataset. Before conducting a trustworthy analysis, it is imperative to address the alarming amount of missing data. The amount of text variables in the dataset is very small, but it can require specific text analysis methods due to its high percentage. The dataset has great insights if it is cleaned and prepared properly.

Staging Chicago Dataset in Talend



* Input Component (chicago\_food\_inspection\_1): This is the beginning of the data flow, when data is read from a source. The figures show that 267,866 rows were processed in 46.48 seconds.
* Transformation Component (tMap\_1): A tmap is been used to transform data and to be stored in thedatabase. It entails applying business logic, combining data from many sources, filtering, and transforming data kinds.
* Output Component (stg\_table\_chicago): The converted data is loaded into a staging table, which is frequently used in databases as a temporary storage space. The procedure loaded 267,866 rows in 47.99 seconds. The staging table is frequently used for further modifications or as an intermediary step before putting data into the ultimate destination.
* Logging Component (tLogRow\_1): This component is used for logging and debugging. It has the ability to write data to a file for review or display it on the console.

Staging Dallas Dataset

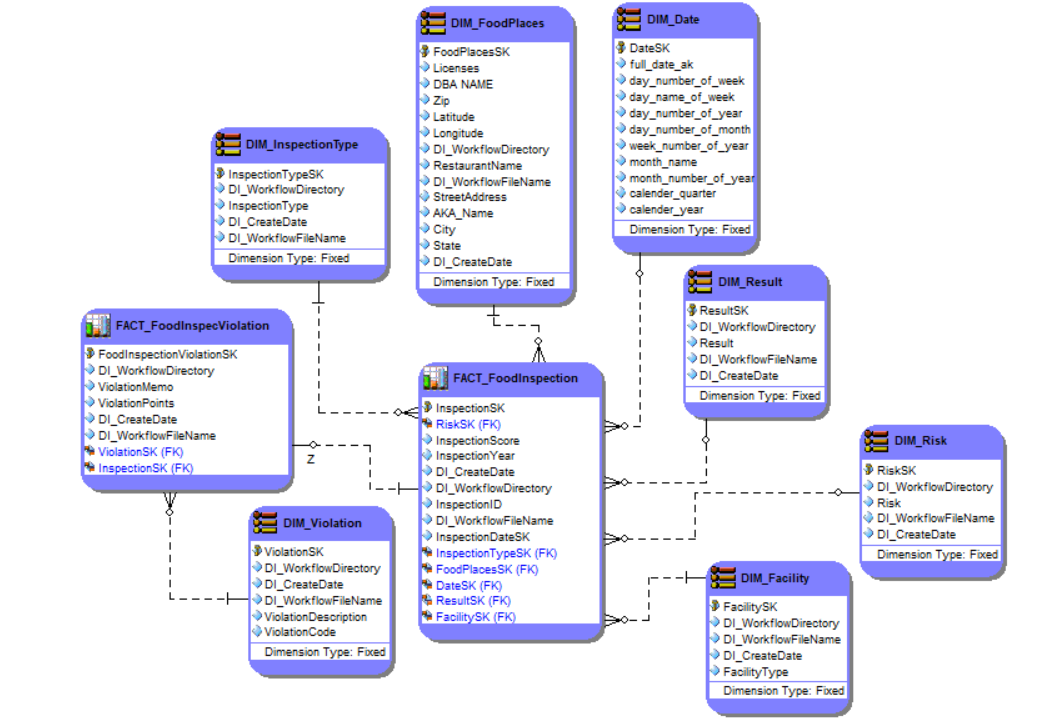


* Input Component (New\_Dallas): Data is read from a source at the start of the data flow, which is represented by the input component (New Dallas). According to the statistics, 2,434.56 rows per second were processed out of 78,400 rows.
* Transformation Component (tMap\_1): Following extraction, fields are usually mapped from the source to the destination, transformations are applied, and data flow conditions are specified using the tMap component. The data flows through the tMap at a pace of 2,161.63 rows per second, processing the same number of rows (78,400) in 36.27 seconds.
* Output Component (stg\_table\_Dallas): The staging table'stg\_table\_Dallas' receives the altered data. In a database, staging tables are frequently used as a temporary intermediate store space for data.
* Logging Component (tLogRow\_1): The dotted line connecting the tLogRow\_1 component often indicates a supplemental or optional data flow that is frequently used for logging or error handling. It means that any route you've put up for data that doesn't match the requirements in the tMap will be flagged for evaluation. But since "0 rows" are logged, it seems that every entry has complied with the tMap's requirements and hasn't been rejected.

**Part 02**

**Dimension Model, SQL queries and DDL Commands**

Dimension table

****

DDL Queries

**DIM\_FoodPlaces**

CREATE TABLE DIM\_FoodPlaces (

FoodPlaceSK INT PRIMARY KEY,

DBA\_NAME VARCHAR(1000),

Zip FLOAT(50),

Latitude FLOAT(50),

Longitude FLOAT(50),

Licenses VARCHAR(1000),

DL\_WorkflowDirectory VARCHAR(1000),

RestaurantName VARCHAR(1000),

DL\_WorkflowFileName VARCHAR(1000),

StreetAddress VARCHAR(1000),

CityName VARCHAR(1000),

State VARCHAR(1000),

DL\_CreateDate DATE,

AKA\_NAME VARCHAR(1000)

);

**DIM\_Facility**

CREATE TABLE DIM\_Facility (

FacilitySK INT PRIMARY KEY,

DI\_WorkflowDirectory VARCHAR(1000),

DI\_WorkflowFileName VARCHAR(1000),

DI\_CreateDate DATE,

FacilityType VARCHAR(1000)

);

**DIM\_InspectionType**

CREATE TABLE DIM\_InspectionType (

InspectionTypeSK INT PRIMARY KEY,

DL\_WorkflowDirectory VARCHAR(1000),

InspectionType VARCHAR(1000),

DL\_CreateDate DATE,

DL\_WorkflowFileName VARCHAR(1000)

);

**DIM\_Date**

CREATE TABLE DIM\_Date (

DateSK INT PRIMARY KEY,

full\_date\_at DATE,

day\_name\_of\_week VARCHAR(1000),

day\_number\_of\_week INT,

day\_number\_of\_year INT,

day\_number\_of\_month INT,

week\_number\_of\_year INT,

month\_name VARCHAR(1000),

month\_number\_of\_year INT,

calendar\_quarter INT,

calendar\_year INT

);

**DIM\_Result**

CREATE TABLE DIM\_Result (

ResultSK INT PRIMARY KEY,

DL\_WorkflowDirectory VARCHAR(1000),

Result VARCHAR(1000),

DL\_WorkflowFileName VARCHAR(1000),

DL\_CreateDate DATE

);

**DIM\_Risk**

CREATE TABLE DIM\_Risk (

RiskSK INT PRIMARY KEY,

DL\_WorkflowDirectory VARCHAR(1000),

Risk VARCHAR(1000),

DL\_CreateDate DATE,

DL\_WorkflowFileName VARCHAR(1000)

);

**DIM\_Violation**

CREATE TABLE DIM\_Violation (

ViolationSK INT PRIMARY KEY,

DL\_WorkflowDirectory VARCHAR(1000),

DL\_CreateDate DATE,

DL\_WorkflowFileName VARCHAR(1000),

ViolationCode VARCHAR(1000),

ViolationDescription VARCHAR(1000)

);

**FACT\_FoodInspection**

CREATE TABLE FACT\_FoodInspection (

InspectionSK INT PRIMARY KEY,

RiskSK INT,

InspectionScore INT,

InspectionYear INT,

DL\_CreateDate DATE,

DL\_WorkflowDirectory VARCHAR(1000),

InspectionID INT,

DL\_WorkflowFileName VARCHAR(1000),

InspectionTypeSK INT,

FoodPlaceSK INT,

DateSK INT,

ResultSK INT,

FacilitySK INT,

FOREIGN KEY (RiskSK) REFERENCES DIM\_Risk(RiskSK),

FOREIGN KEY (InspectionTypeSK) REFERENCES DIM\_InspectionType(InspectionTypeSK),

FOREIGN KEY (FoodPlaceSK) REFERENCES DIM\_FoodPlaces(FoodPlaceSK),

FOREIGN KEY (DateSK) REFERENCES DIM\_Date(DateSK),

FOREIGN KEY (ResultSK) REFERENCES DIM\_Result(ResultSK),

FOREIGN KEY (FacilitySK) REFERENCES DIM\_Facility(FacilitySK)

);

**FACT\_FoodInspViolation**

CREATE TABLE FACT\_FoodInspViolation (

FoodInspectionViolationSK INT PRIMARY KEY,

DL\_WorkflowDirectory VARCHAR(1000),

ViolationMemo VARCHAR(1000),

ViolationPoints INT,

DL\_CreateDate DATE,

DL\_WorkflowFileName VARCHAR(1000),

ViolationSK INT,

InspectionSK INT,

FOREIGN KEY (ViolationSK) REFERENCES DIM\_Violation(ViolationSK),

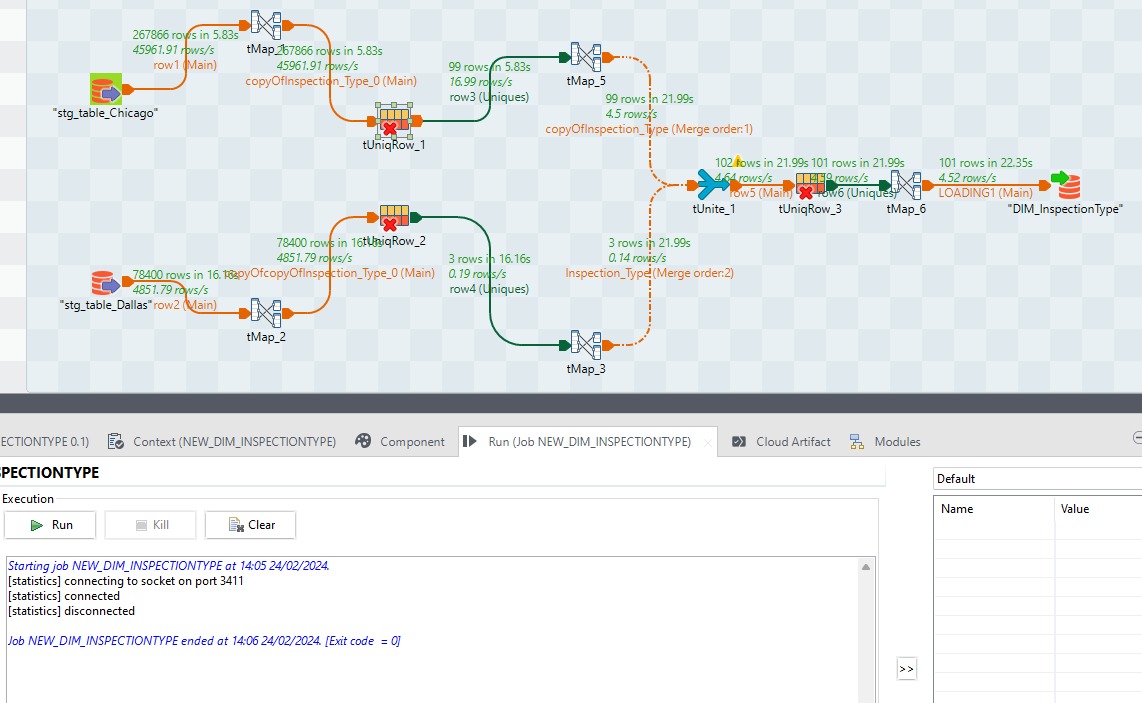
FOREIGN KEY (InspectionSK) REFERENCES FACT\_FoodInspection(InspectionSK)

);

**Part 03**

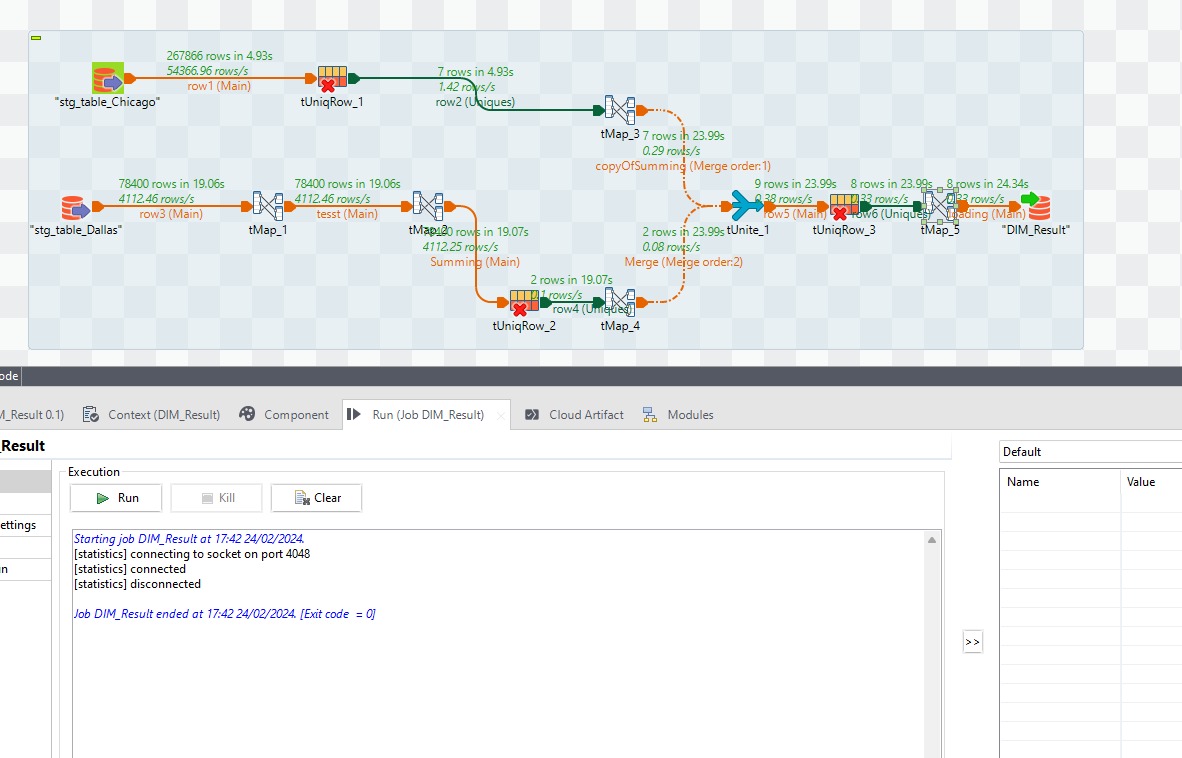
**ETL Jobs for all Dimensions and Facts**

ETL job for loading DIM\_InspectionType from Stage tables of Chicago Dataset and Dallas Dataset



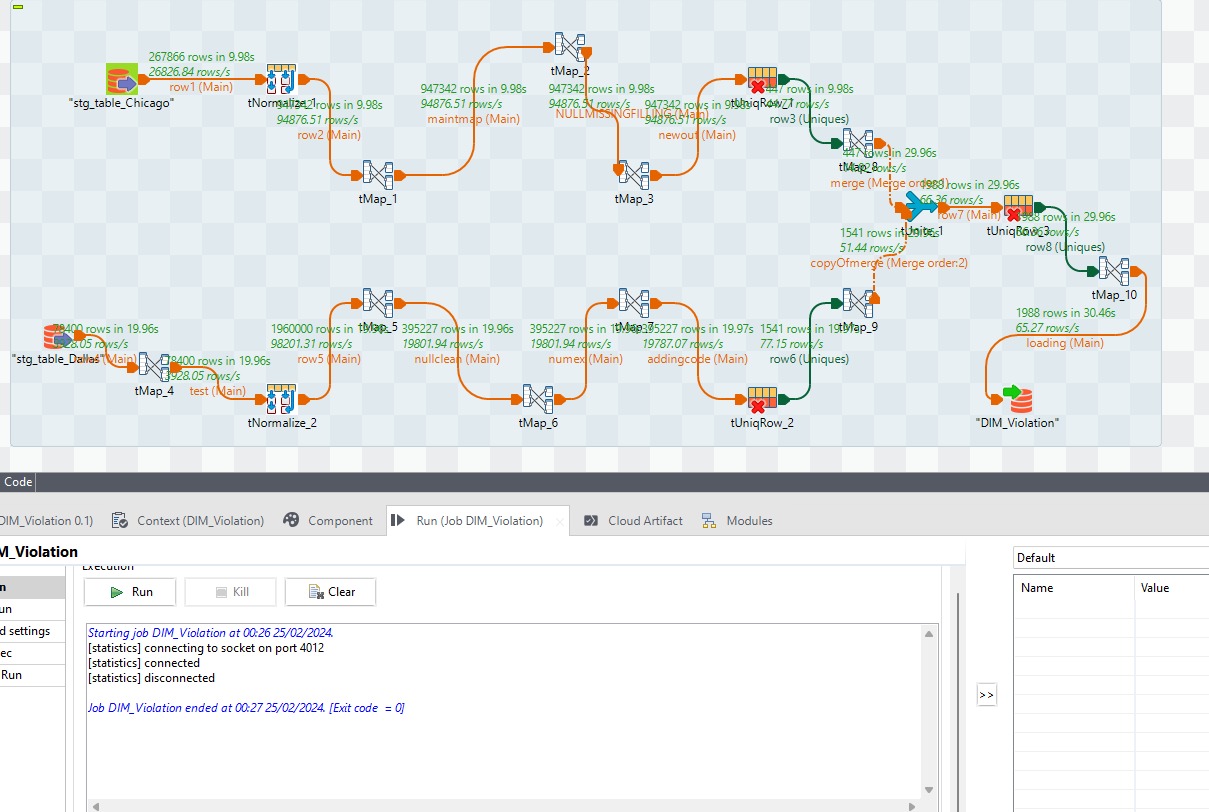
An alert check of the "Inspection Type" column inside both staging tables of the datasets revealed that no null values existed. This discovery was validated by careful data validation procedures, which ensured the integrity and completeness of the data being analysed. As a proactive approach to address possible data quality concerns, a TMAP code was added to replace any future occurrences of null values in this field with the phrase "Unknown." This procedure was performed to ensure that the dataset remained consistent and reliable, allowing for more efficient data processing and analysis.   
  
Furthermore, during the creation of the data warehouse, a sequence for Surrogate Key (SK) generation was created expressly for usage in the Dallas environment. This sequence was created to offer a unique identifier for each record, improving the data structure and allowing for more efficient data retrieval and maintenance. This sequence reflects a systematic approach to data warehousing, ensuring that each piece of data is correctly tracked and maintained throughout its existence in the system. This innovation demonstrates our dedication to data integrity and operational efficiencies in data management procedures.

ETL job for loading DIM\_Result from Stage tables of Chicago Dataset and Dallas Dataset



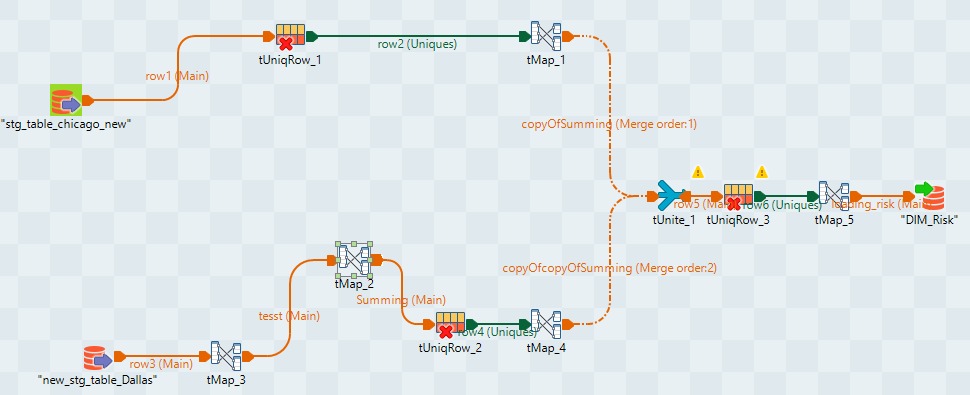
Because there was no direct 'results' column in a dataset about Dallas inspections, a novel technique was required for analysis. To determine the outcome of each inspection, a computation based on the sum of all infraction points was used. The inspection findings were determined using the following criteria: If the amount of violation points exceeded 60, the inspection was categorised as 'Fail'; for sums larger than 30 but less than or equal to 60, the result was classified as 'Pass with Warning'; and for sums less than or equal to 30, the inspection was judged 'Pass'. Given that the 'pass' characteristic was shared by datasets from both Dallas and Chicago, the 'tuniqrow' command was used to verify that only one record per unique inspection was transmitted for further processing, successfully reducing duplicate entries and speeding the data analysis process.

ETL job for loading DIM\_Violation from Stage tables of Chicago Dataset and Dallas Dataset



When analysing the datasets for Chicago and Dallas to extract violation codes hidden inside descriptions, unique issues arose, notably with the Dallas dataset. Unlike the easy extraction in Chicago, the Dallas data needed a more sophisticated technique since the infraction numbers were intertwined with descriptions over 25 columns. To remedy this, these columns were initially combined into a single field, followed by the 'tnormalise' function, which separated the violation codes. The following stages entailed using a function to remove null values, thus separating the programmes. Given the similarities in violation codes between Chicago and Dallas, a unique identification was required to ensure distinction. This was accomplished by prefixing '0' to Chicago codes and '1' to Dallas codes, which ensured the dataset's violation codes were clear and distinct, allowing for a more organised and efficient analysis.

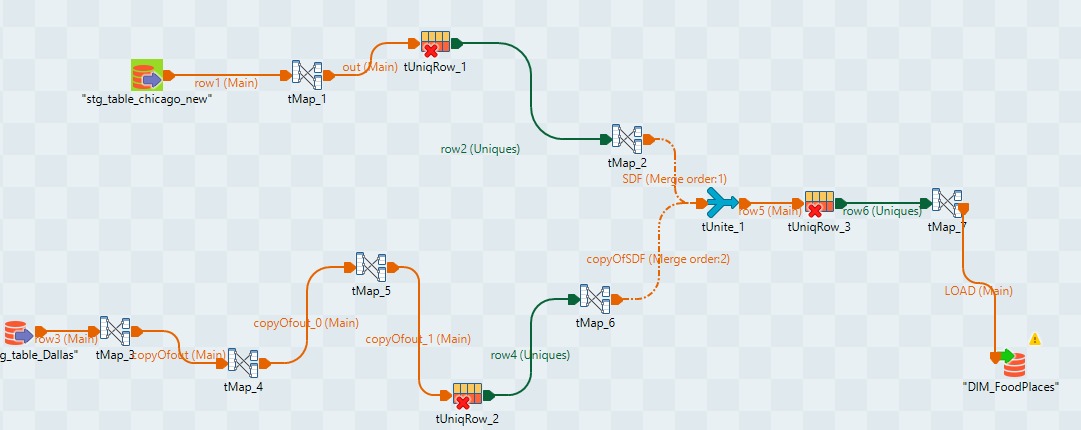
ETL job for loading DIM\_Risk from Stage tables of Chicago Dataset and Dallas Dataset



Data is collected from two staging tables, ‘stg\_table\_chicago\_new’ and `new\_stg\_table\_Dallas’ , in the Talend ETL job that is shown. Following a first round of data loading and cleaning, these tables represent the first holding regions. After that, `tUniqRow\_1’ is used to undergo a deduplication operation on the Chicago data, guaranteeing the dataset's uniqueness. Concurrently, (tMap\_3) is used to change the Dallas data, suggesting that a customised mapping step or transformation is being performed on this specific data set.

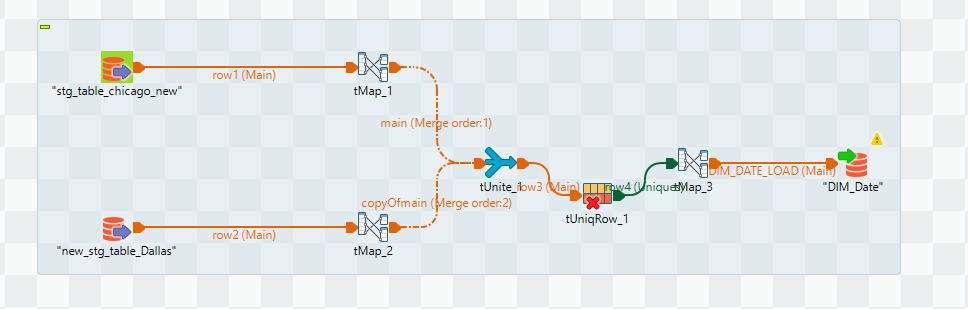
Following the first treatments, (tMap\_1) is used to further alter the Chicago stream, which is now reduced to unique entries. After going through (tMap\_), the Dallas data goes via (tMap\_2), which is another layer of aggregation or modification. Then, these two data streams are joined in a component called `Summing`, which suggests a summative or consolidation process, maybe in advance of a combined analysis or reporting.   
  
Then, in {tMap\_4}, the combined data is transformed one more time before splitting in two. One route, which repeats the previous procedure to confirm data uniqueness, goes to (tUniqRow\_2). (tMap\_5) is the next path, and it is focused on risk analysis, as shown by the components {mapping\_risk} and ‘DIM\_Risk’, which highlight the importance of data integrity and risk assessment in this work. Finally, the outputs are directed to (tUnite\_1) and (tUniqRow\_3), indicating that the processed streams have been combined into a single, deduplicated dataset that is ready for further analysis or downstream use.

ETL job for loading DIM\_FoodPlaces from Stage tables of Chicago Dataset and Dallas Dataset



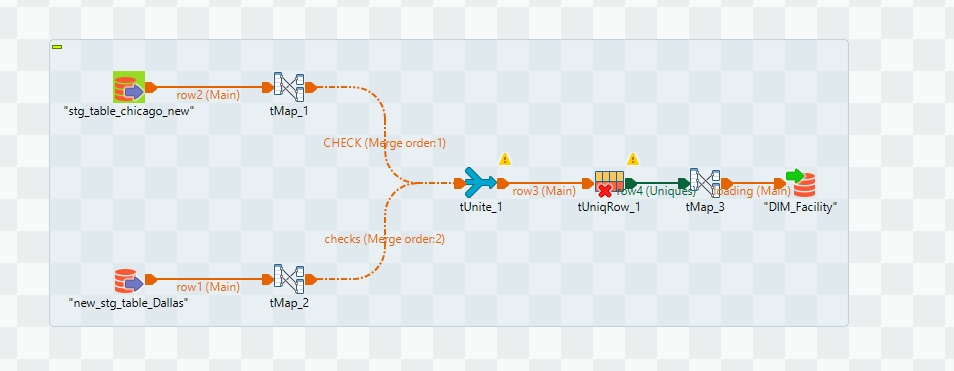
For mapping and deduplication, the Chicago data is processed by tMap\_1 and tUniqRow\_1, whereas the Dallas data is handled by tMap\_4. The two data streams are then combined by tUnite\_1 and any leftover duplicates are eliminated by tUniqRow\_3, after which they undergo further modification (tMap\_2 for Chicago and tMap\_5 followed by tMap\_6 for Dallas) and deduplication (tUniqRow\_2 for Dallas data). The final step of the procedure involves loading the cleaned and converted data into a dimension table called DIM\_FoodPlaces.Top of Form

ETL job for loading DIM\_Date from Stage tables of Chicago Dataset and Dallas Dataset



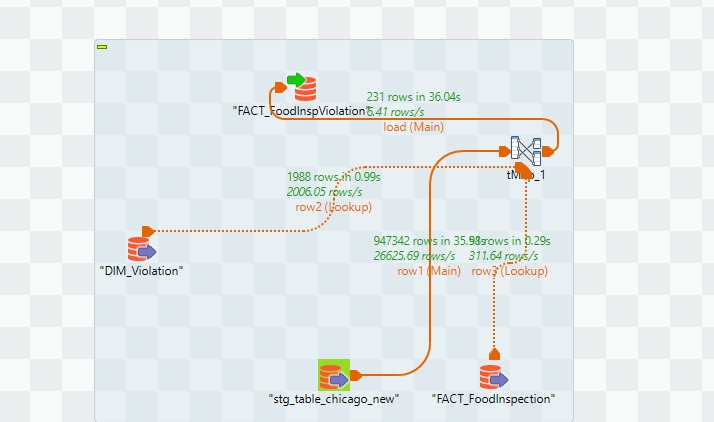
The results of these transformations are then combined in a tUnite component, indicating a consolidation of data from both cities, after being mapped using staging tables. Then, as indicated by the naming convention and the existence of a date dimension load operation (tMap\_3), a deduplication step via tUniqRow\_1 guarantees the uniqueness of the combined data set, which is essential before the data is finally loaded into the DIM\_Date dimension table, a section of a data warehouse intended to facilitate time-based reporting or analytics.

ETL job for loading DIM\_Facility from Stage tables of Chicago Dataset and Dallas Dataset



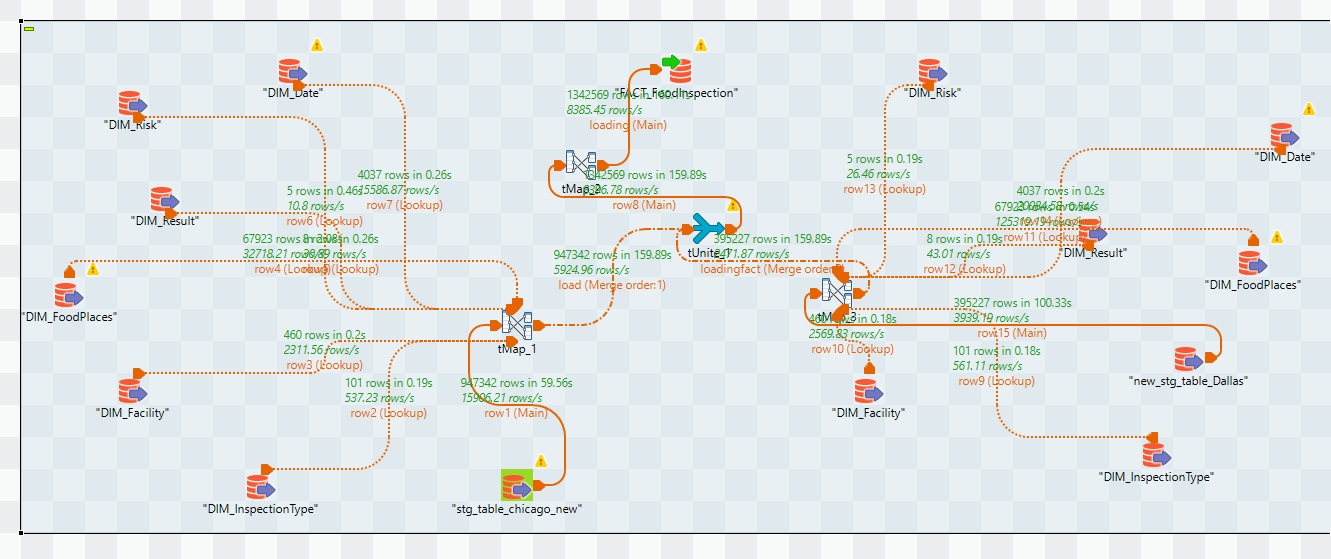
Following these changes and the use of staging tables, the data streams are merged using tUnite\_1, signifying the combination of datasets from Chicago and Dallas. After the merging, duplicate entries are removed using tUniqRow\_1, guaranteeing data uniqueness before it is sent to tMap\_3 for additional processing or business logic application. The final result is sent to a dimension table called DIM\_Facility, which is commonly used for analytical reasons in data warehousing. The form of this work indicates that the goal is to combine data from two sources into a single, clean dataset for analytics or reporting, with an emphasis on facilities data.

ETL job for loading FACT\_FoodInspecViolation from Stage tables of Chicago Dataset and Dallas Dataset



The data in this case was taken from the staging table stg\_table\_chicago\_new, processed using tMap\_1, and joined with DIM\_Violation. By adding details from the dimension table to the staging data, the transformation improves and verifies it. Following the transformation, the information is imported into the FACT\_FoodInspection and FACT\_FoodInspViolation tables, which are separate entities. The flow from FACT\_FoodInspection to FACT\_FoodInspViolation is smaller than that of FACT\_FoodInspection, at 6.41 rows/s, indicating that this may require more intricate reasoning or a conditional filter that only permits specific rows to pass through. The procedure is intended to add enhanced and verified data from the Chicago staging area to these fact tables, which are commonly used in data warehousing to hold quantitative measures for analysis.

ETL job for loading FACT\_FoodInspection from Stage tables of Chicago Dataset and Dallas Dataset



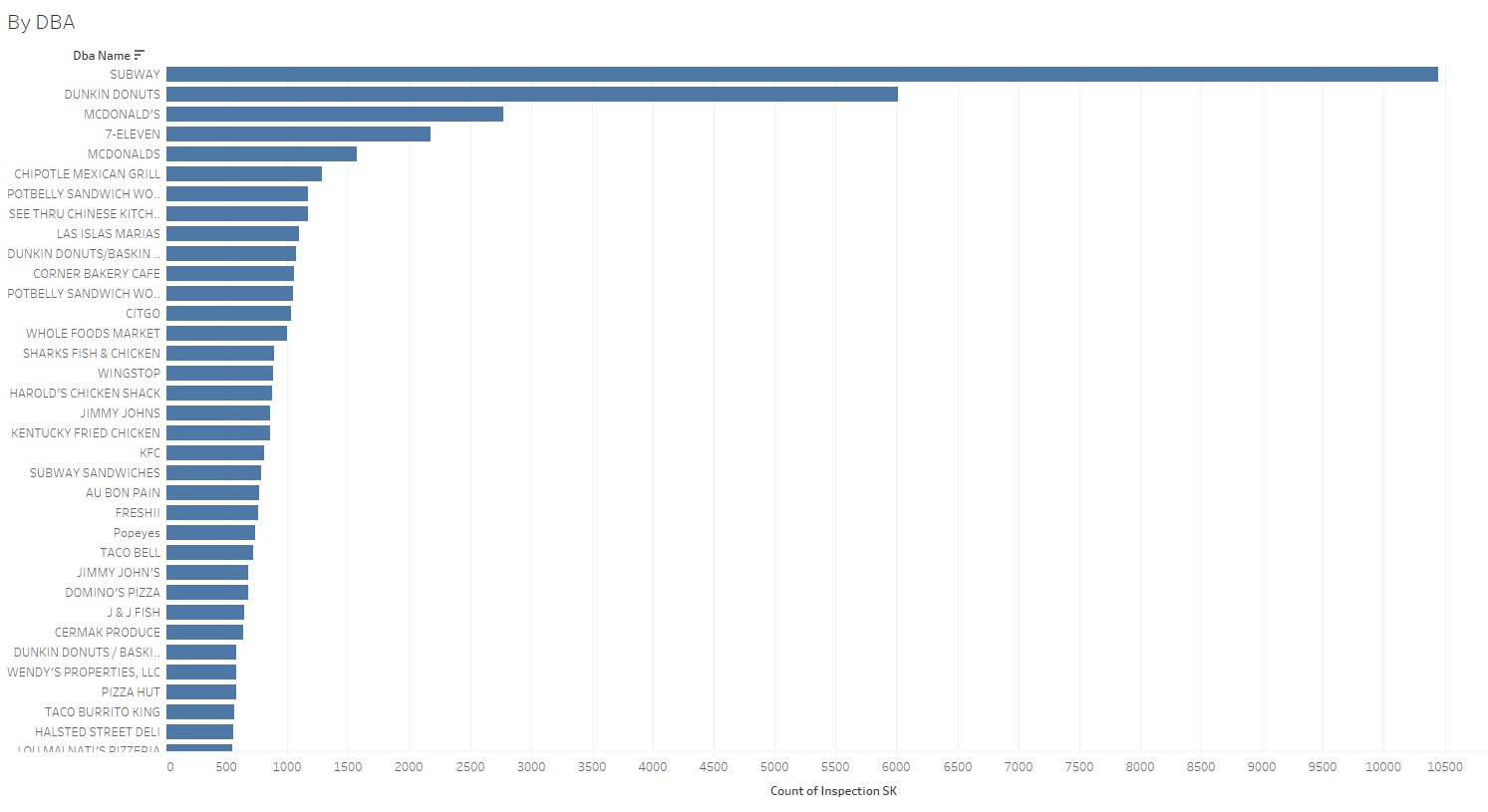
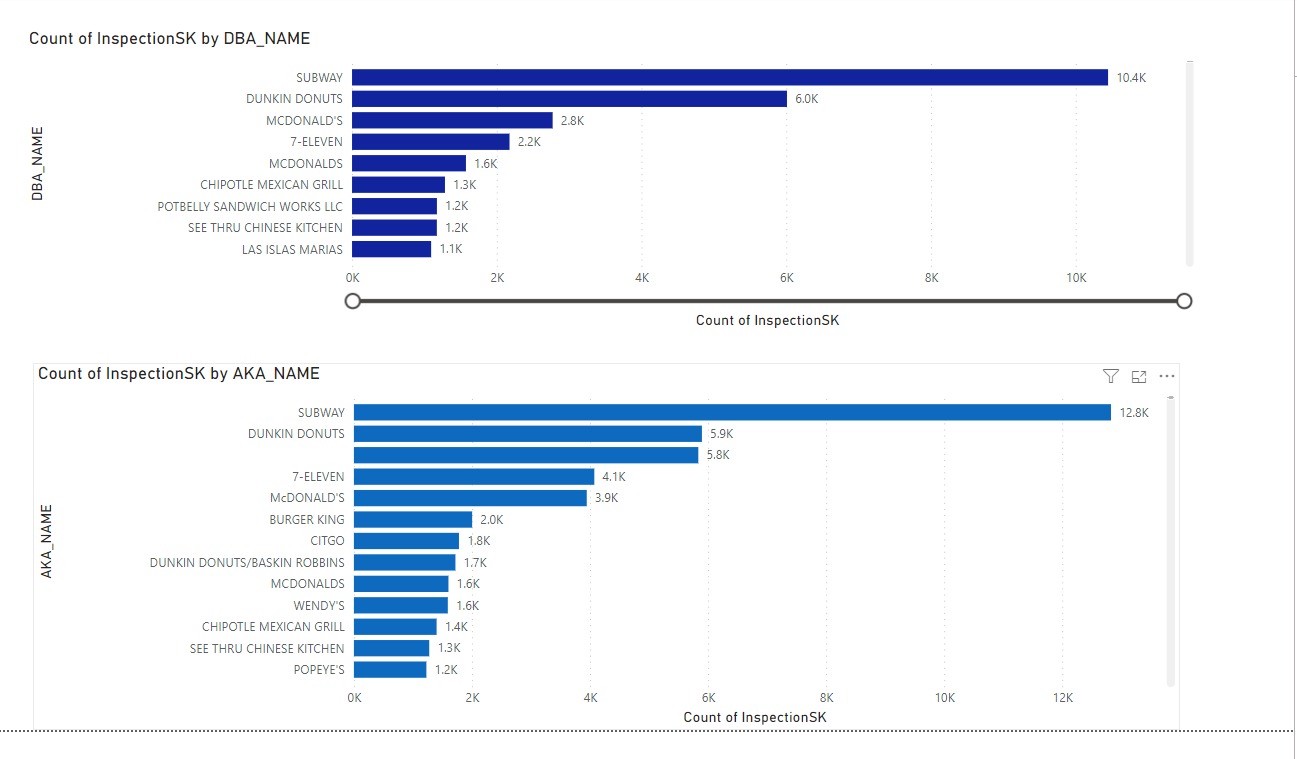
The two primary staging tables, stg\_table\_chicago\_new and new\_stg\_table\_Dallas, are where the data is extracted from, indicating that this task integrates data from two distinct sources. After passing through tMap\_1, the Chicago data is probably transformed and made more valuable by connecting it with several dimension tables, like DIM\_Facility, DIM\_InspectionType, and others. The existence of several lookup processes to these dimension tables, which are characteristic of a dimensional modelling technique in data warehousing, indicates that the Chicago data is being enriched with extra qualities including risk, outcome, and inspection kinds.

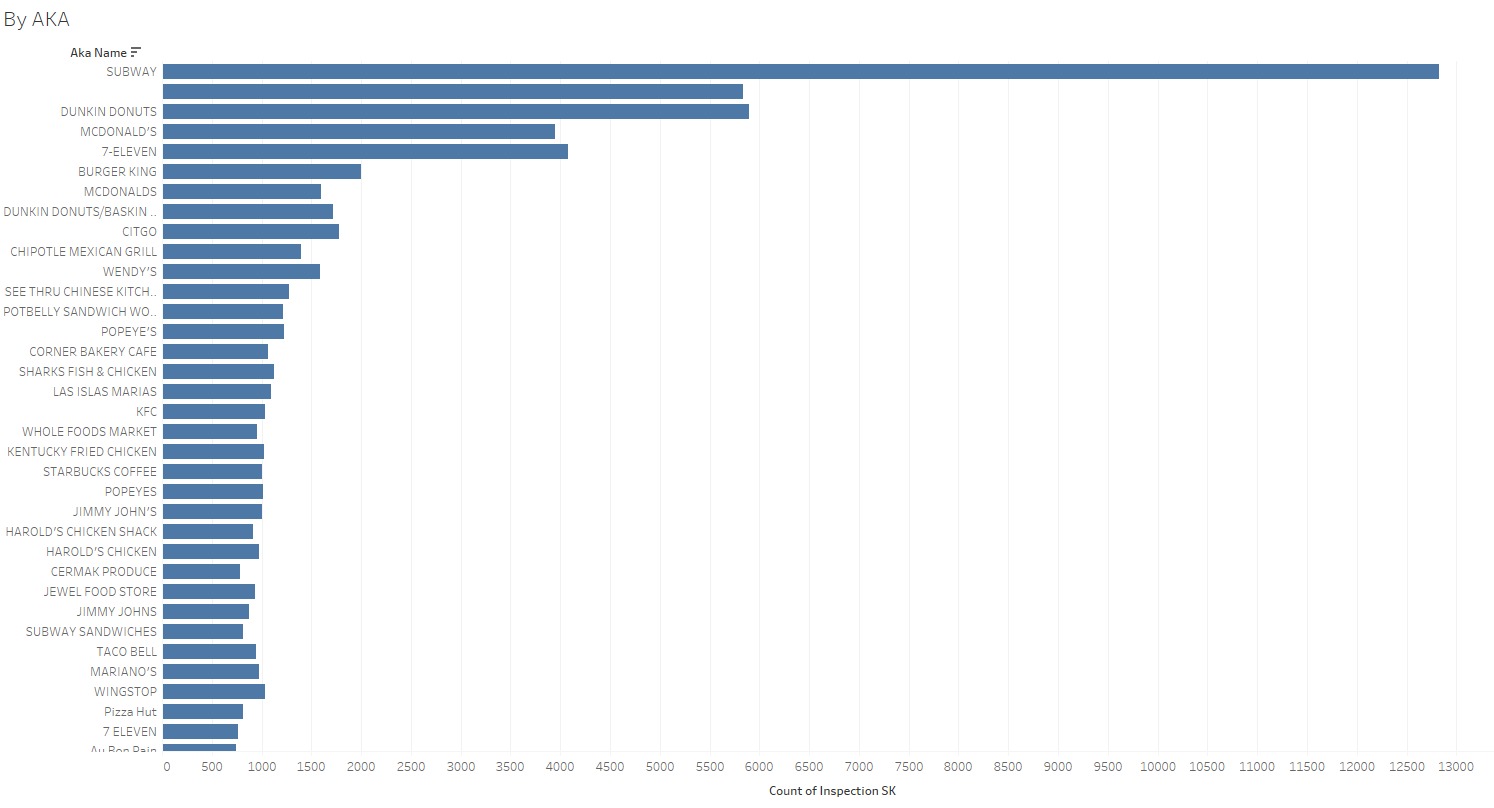
Following the transformation phase, both the Chicago and Dallas data streams are merged using a tUnite component and then split again, with one stream loading into DIM\_Risk and the other undergoing additional transformation through tMap\_3 before loading into DIM\_Result. The enriched Chicago data is then loaded into the FACT\_FoodInspection table. This suggests an emphasis on breaking out various data aspects for analytical reasons. Before being combined with the Chicago data, the Dallas data seems to go through a similar enrichment procedure that includes lookup connects to dimension tables like DIM\_Date, DIM\_FoodPlaces, and DIM\_InspectionType. The use of several lookups and transformations helps to give a thorough picture of the data.

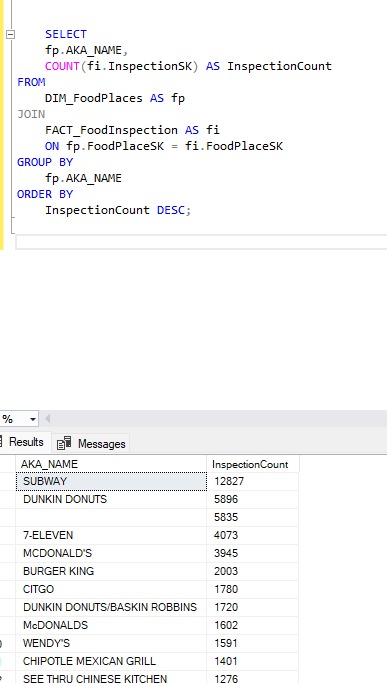
**Part 04**

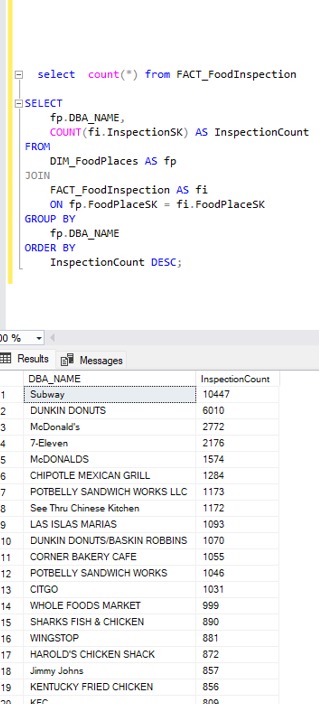
**Visualisations in Power BI and Tableau**

Count of InspectionSK by DBA name and AKA name

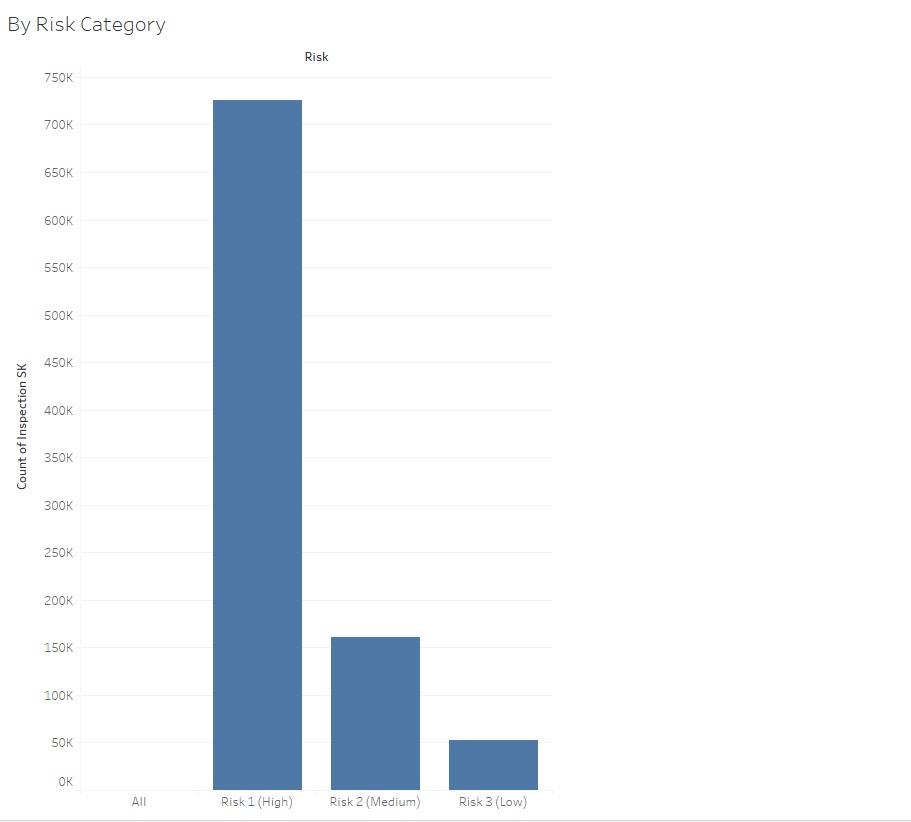
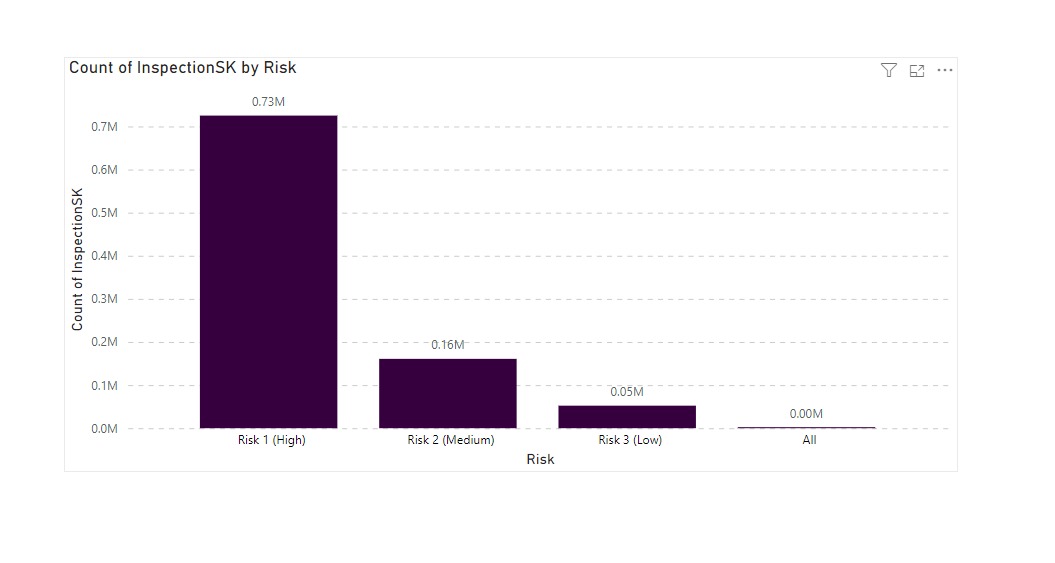


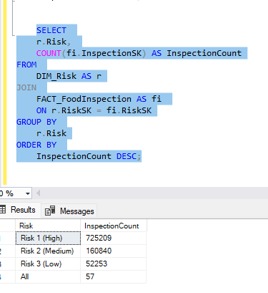




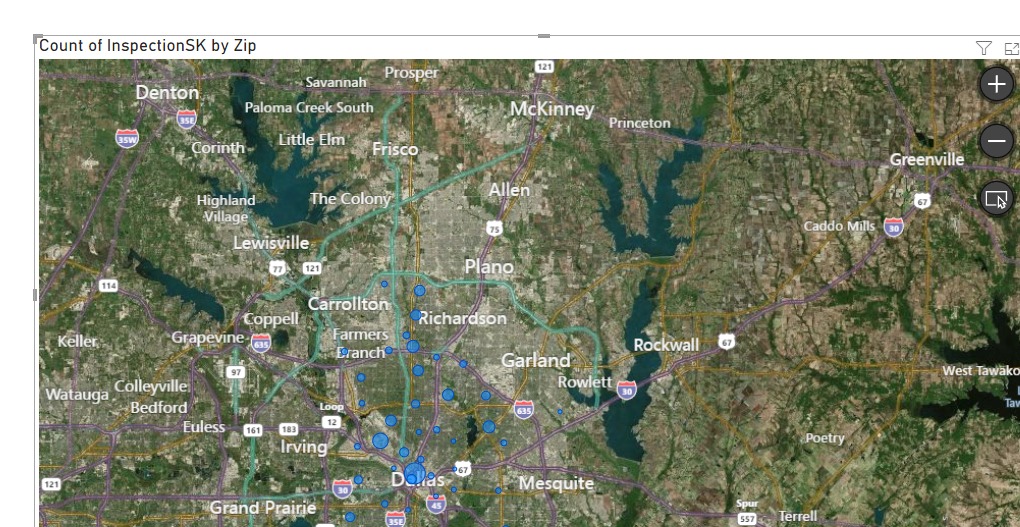


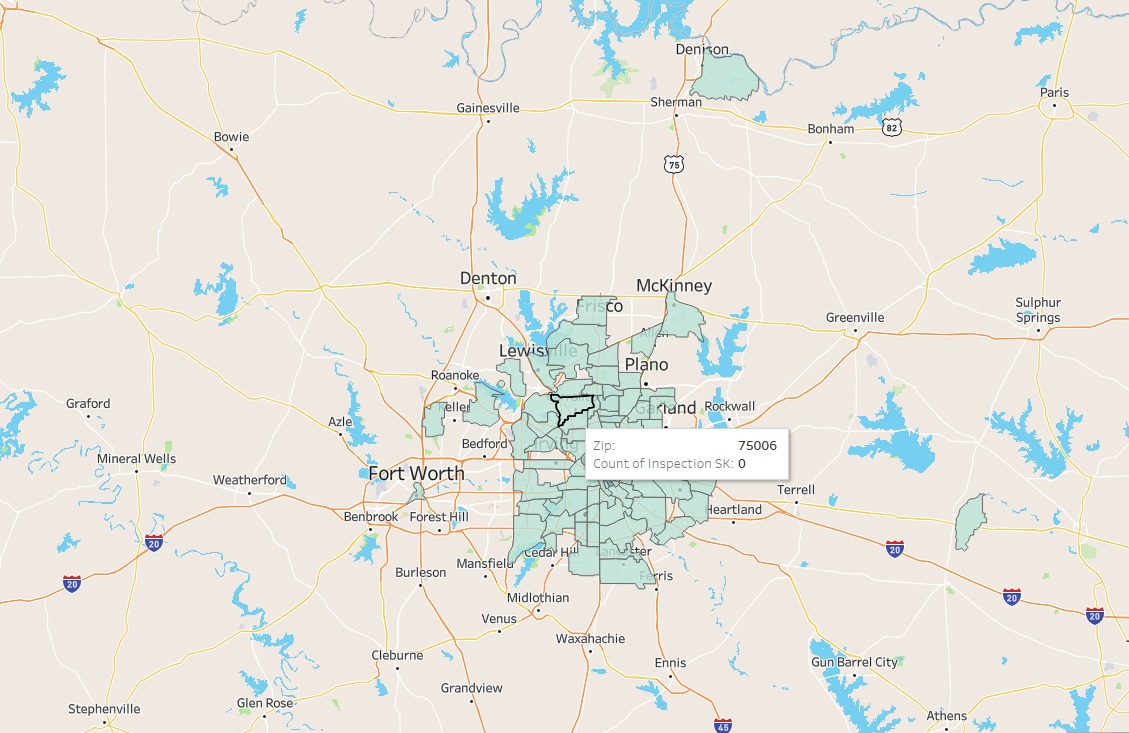
Count Of Inspection SK by Risk



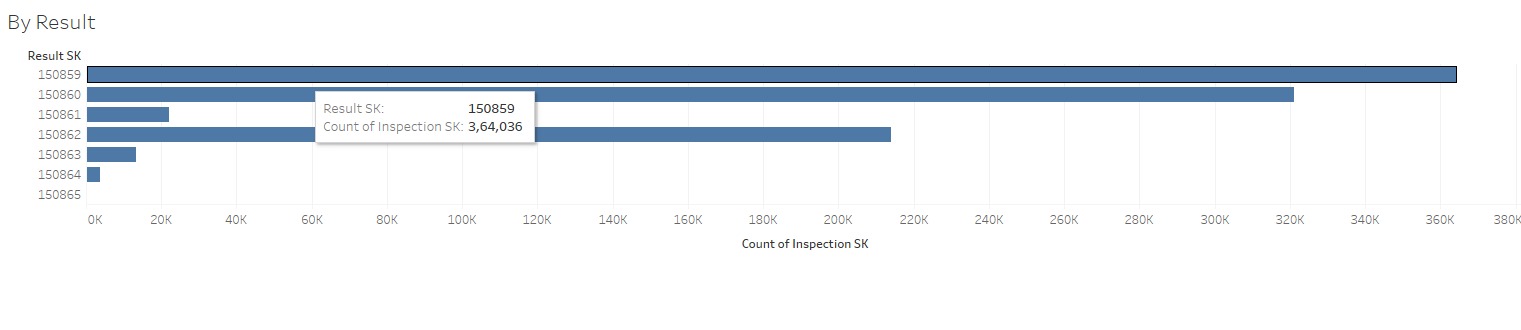
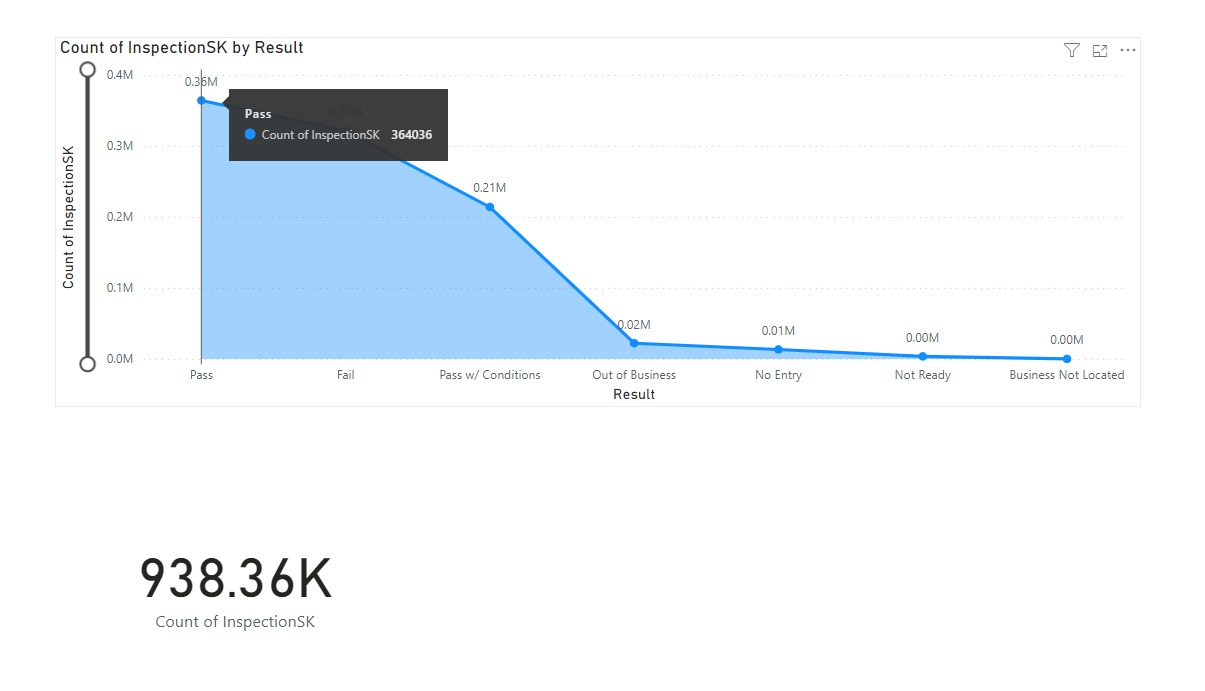


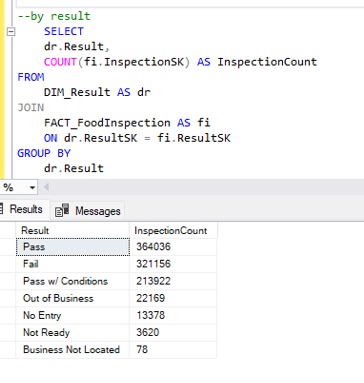
Count of InspectionSK by zipcode



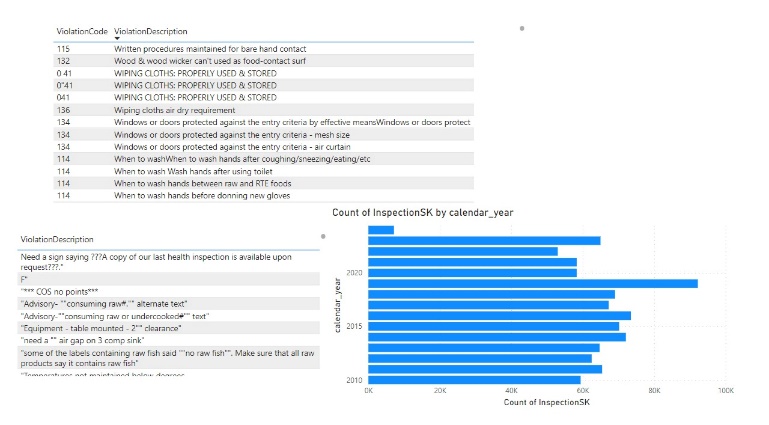


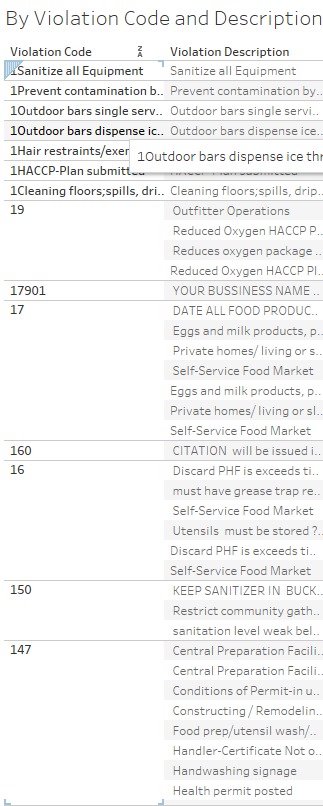
Count of InspectionSK by Result



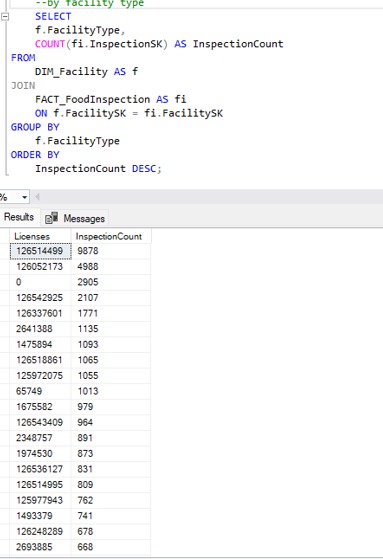
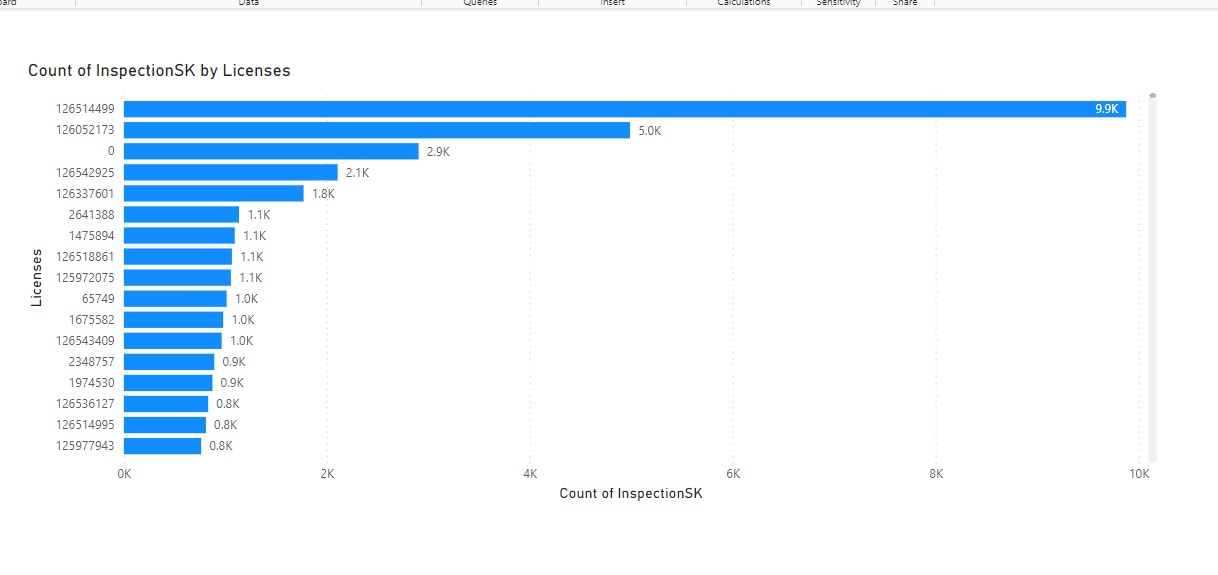
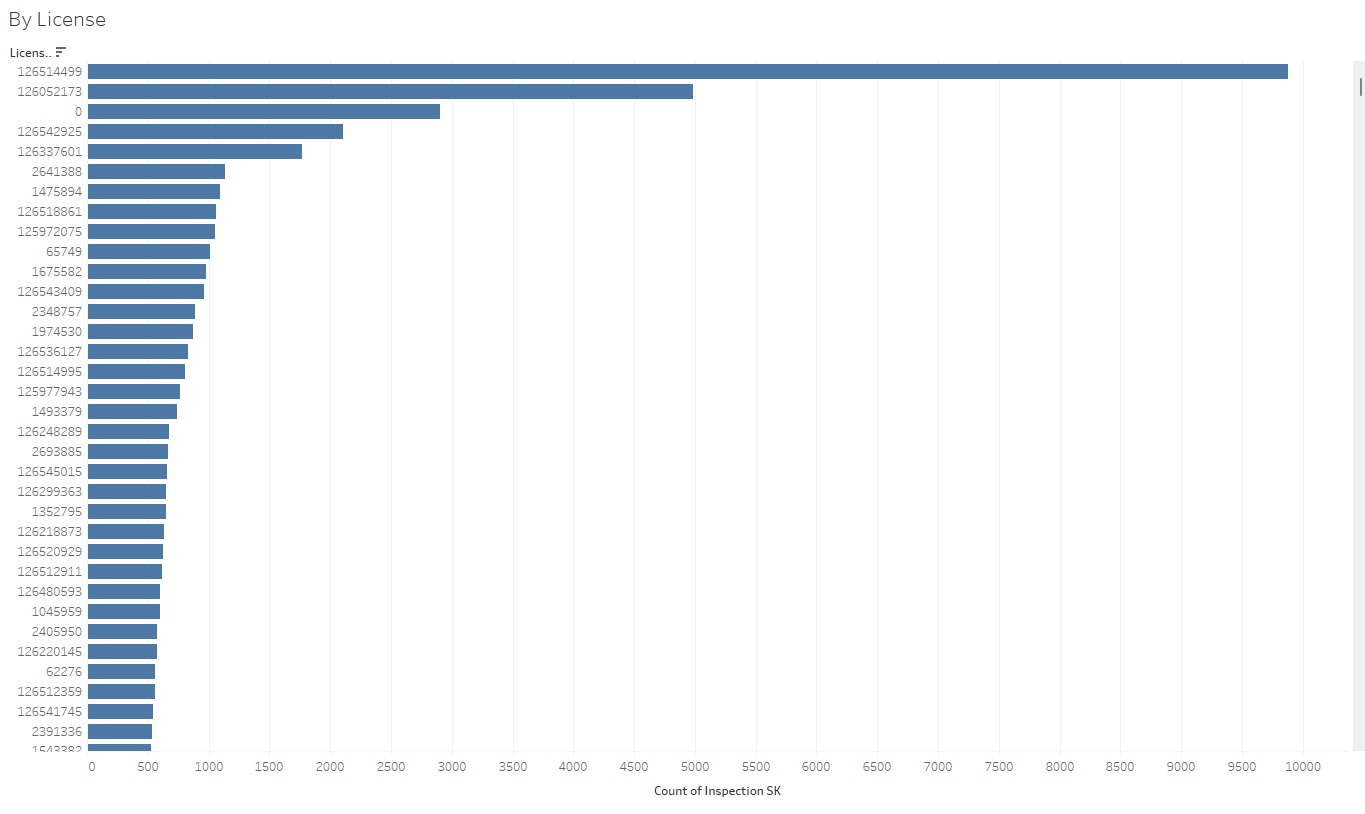


Count by InspectionSk by Violation

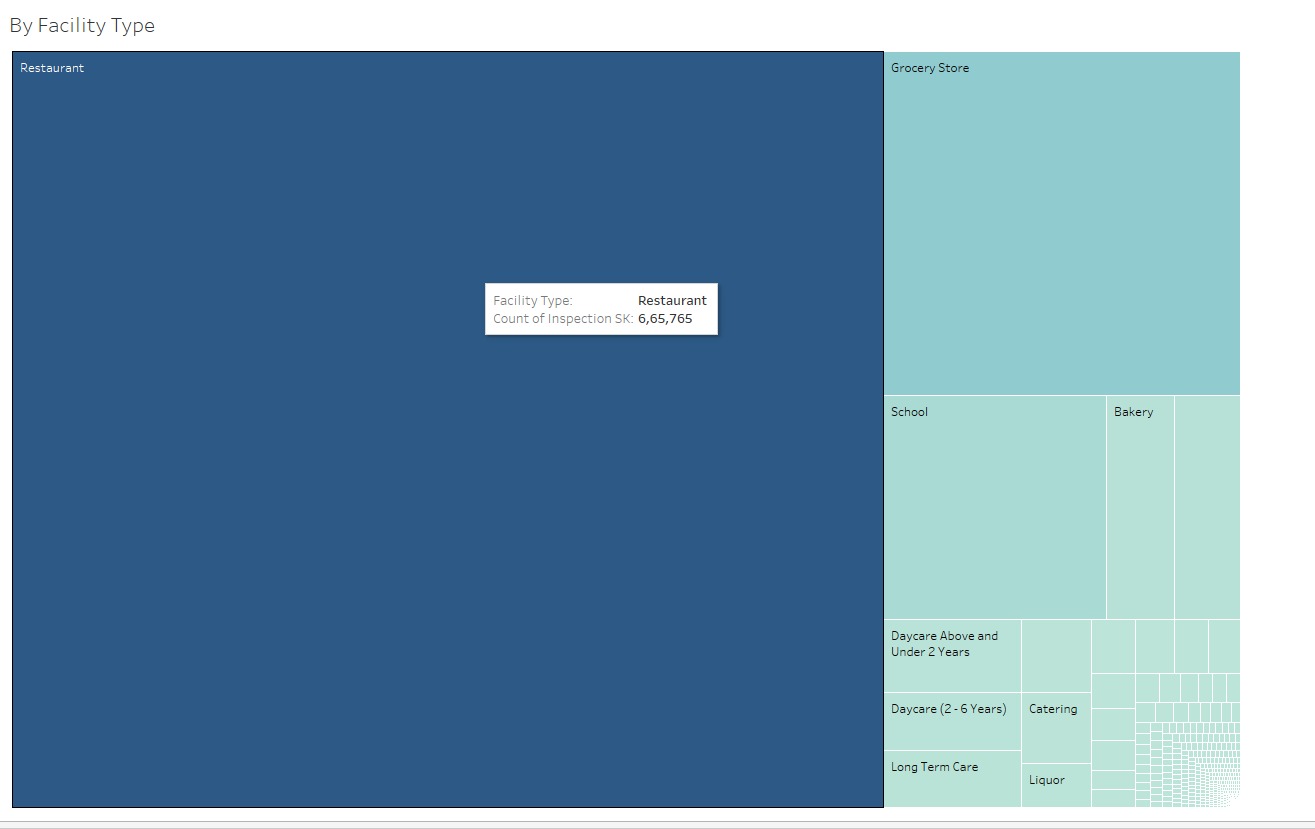
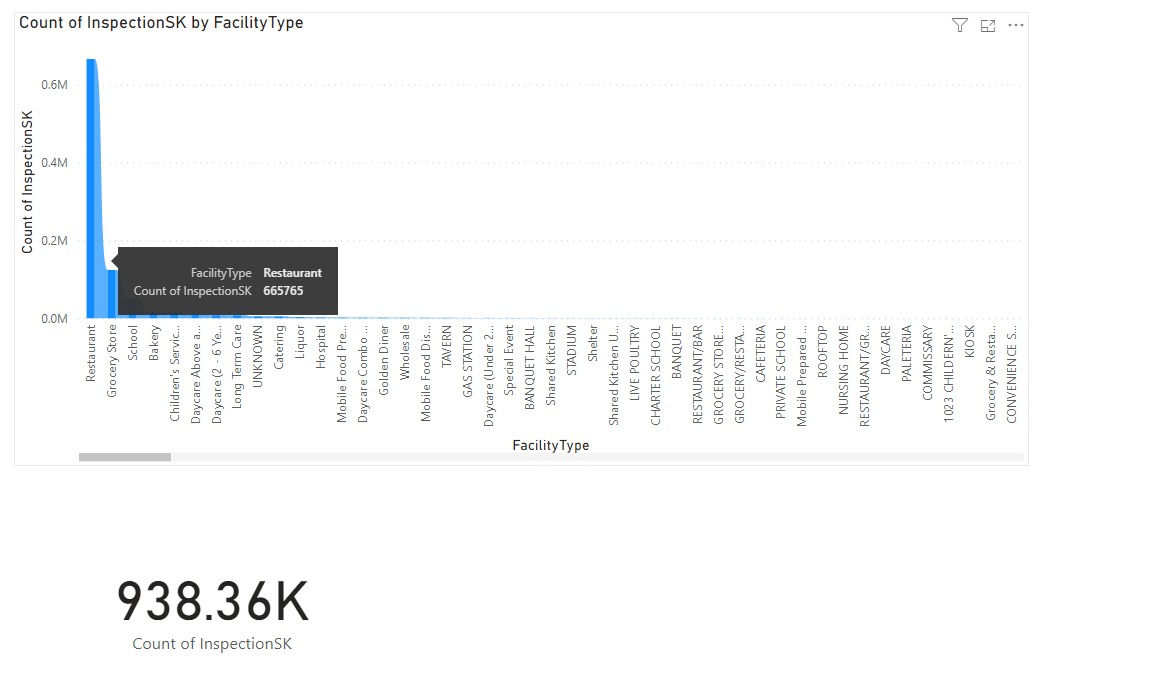


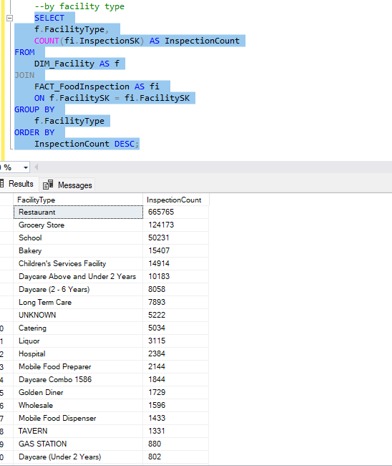


Count of InspectionSK by Licenses



Count of Inspection SK by Facility Type



****

