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

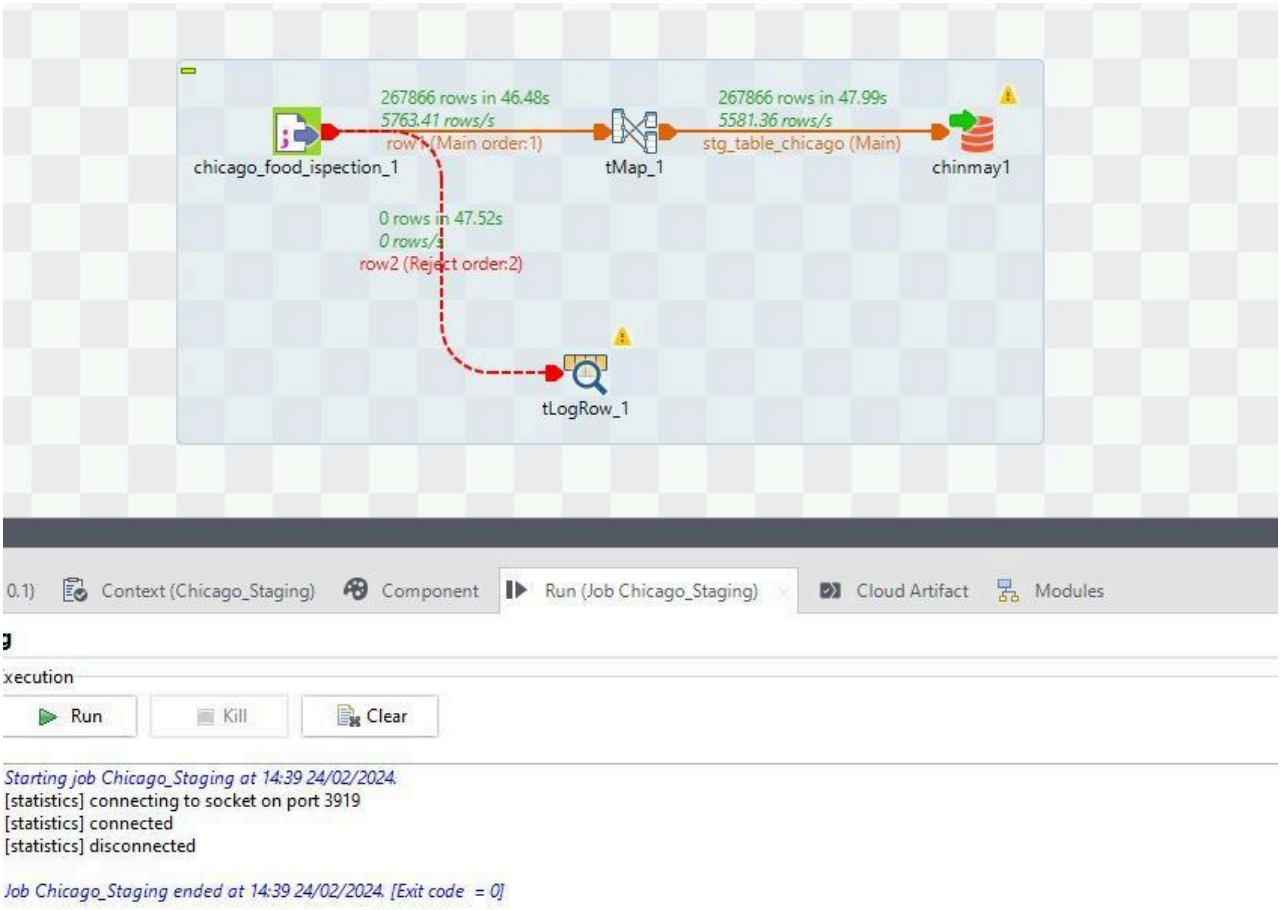
The study that is offered offers a thorough examination of the Dallas dataset that includes information pertinent to the food industry. The report includes a wide range of factors designed to address any discrepancy found in the dataset. With 114 variables spread across 78,400 observations, the dataset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

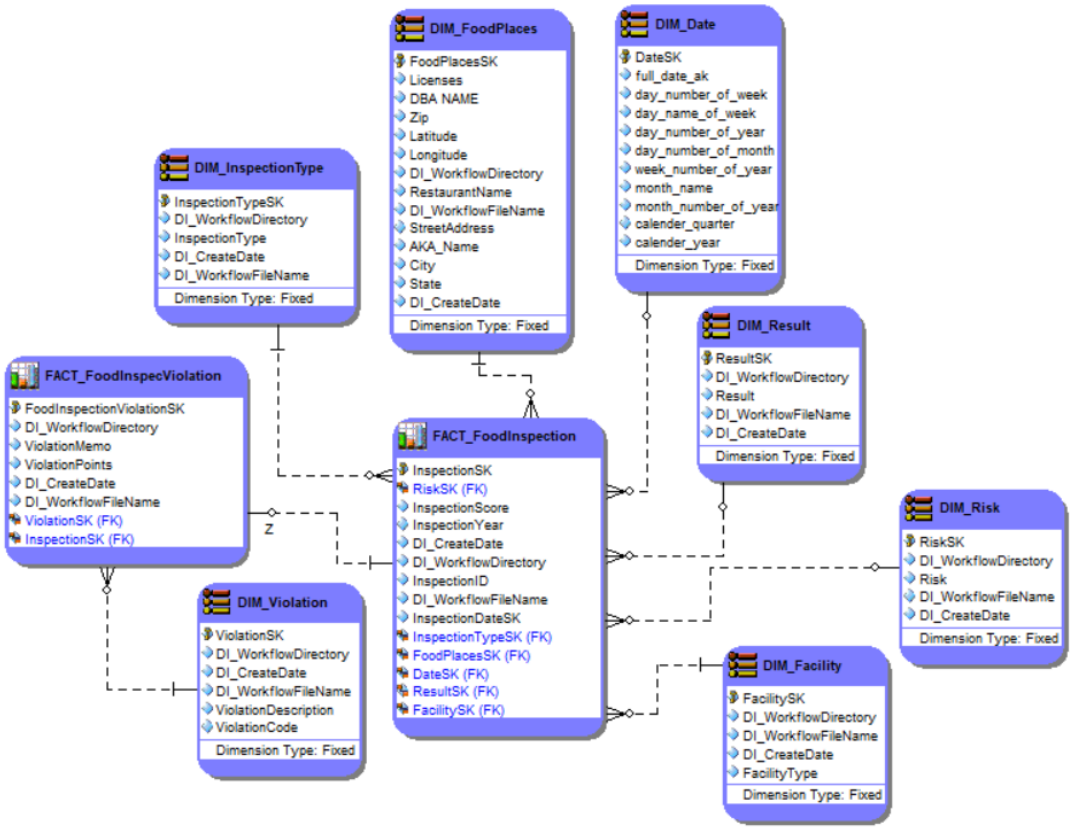
- Input Component (New_Dallas): Data is read from a source at the start of the data flow, which is represented by the input component (New Dallas). According to the statistics, 2,434.56 rows per second were processed out of 78,400 rows.
- Transformation Component (tMap_1): Following extraction, fields 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)
);
```

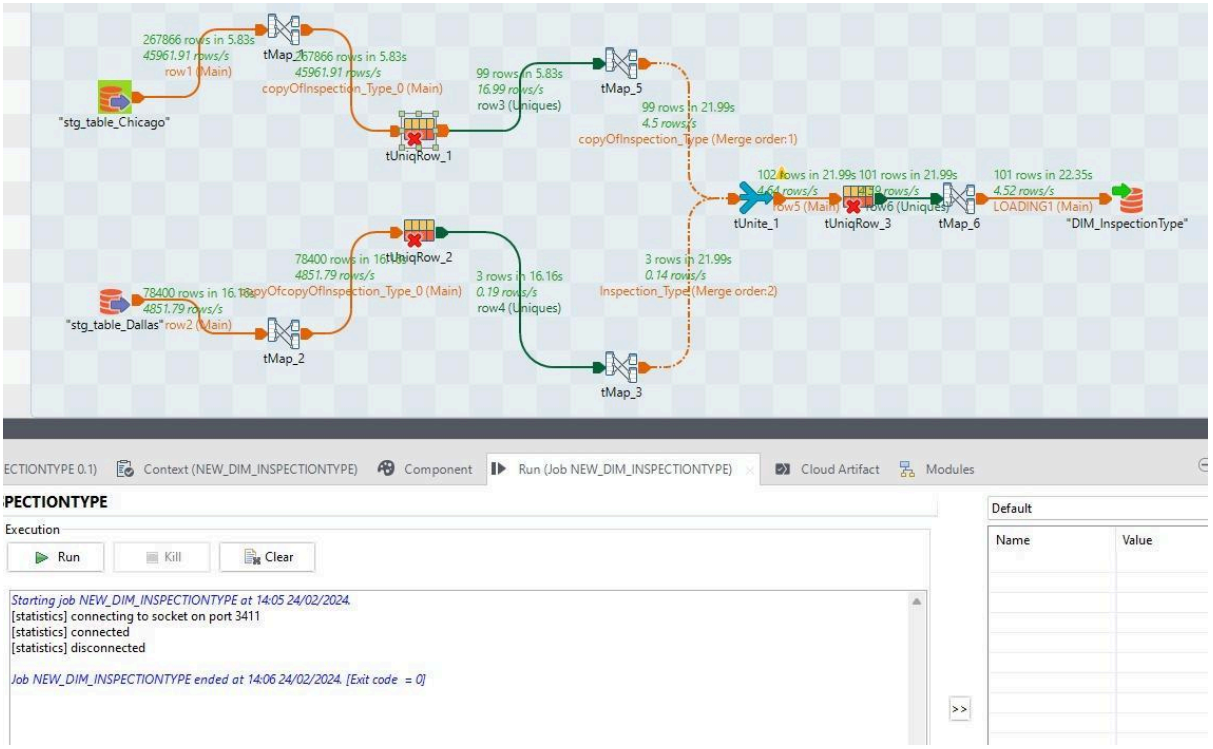
FACT_FoodInspViolation

```
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

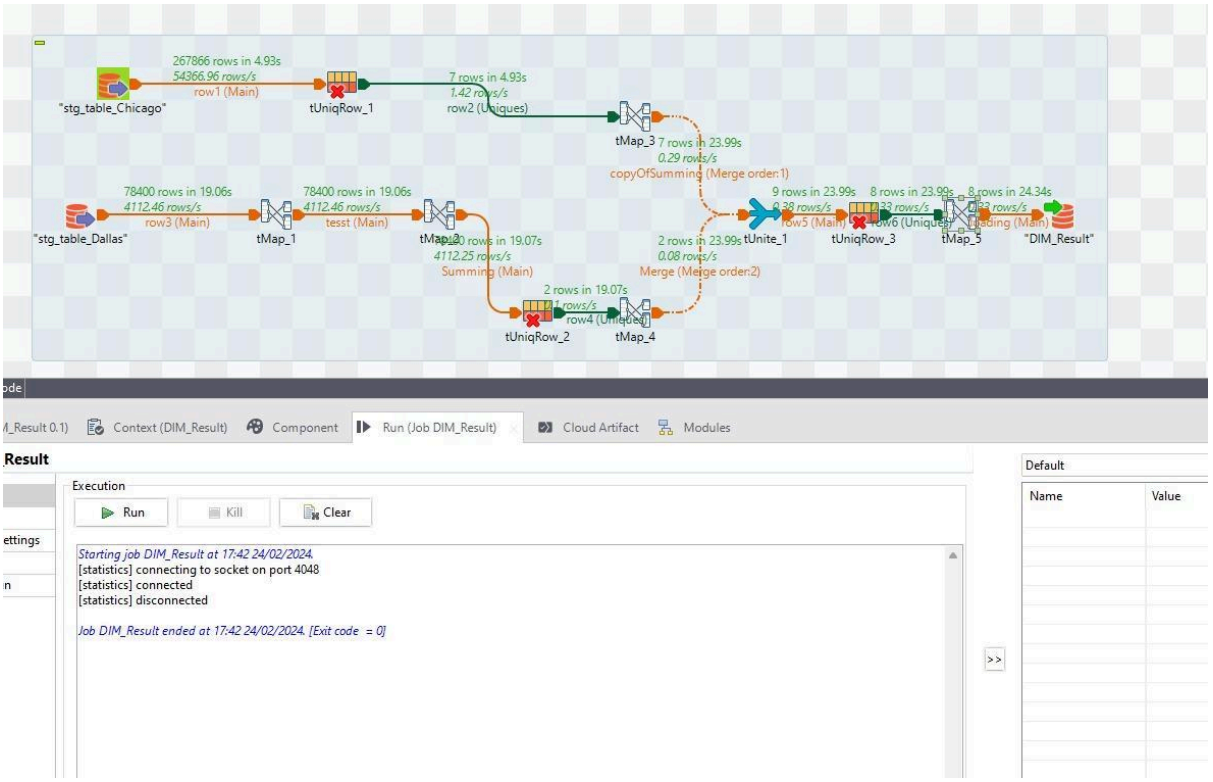


An alert check of the "Inspection Type" column inside both staging tables of the datasets revealed that no null values existed. This discovery was validated by careful data validation procedures, which ensured the integrity and completeness of the data being analysed. As a proactive approach to address 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

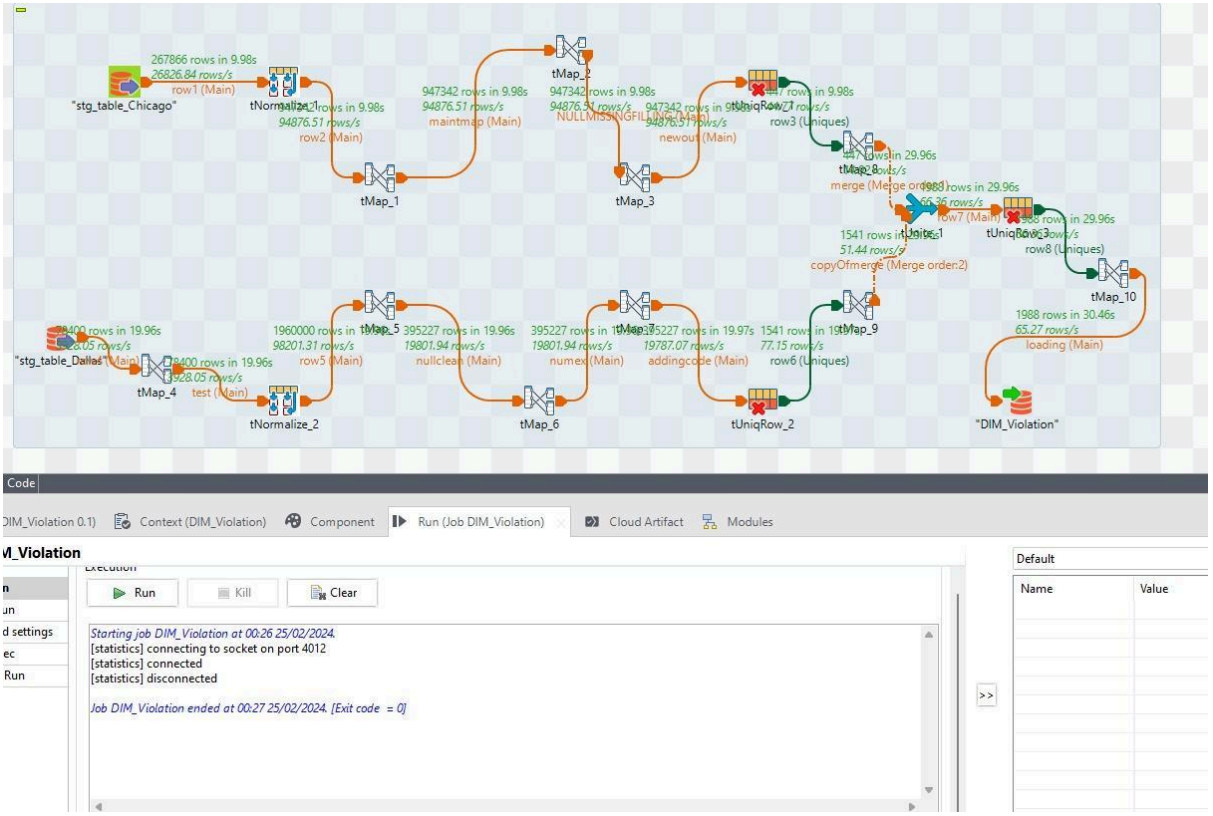
demonstrates our dedication to data integrity and operational efficiencies in data management procedures.

ETL job for loading DIM_Result from Stage tables of Chicago Dataset and Dallas Dataset



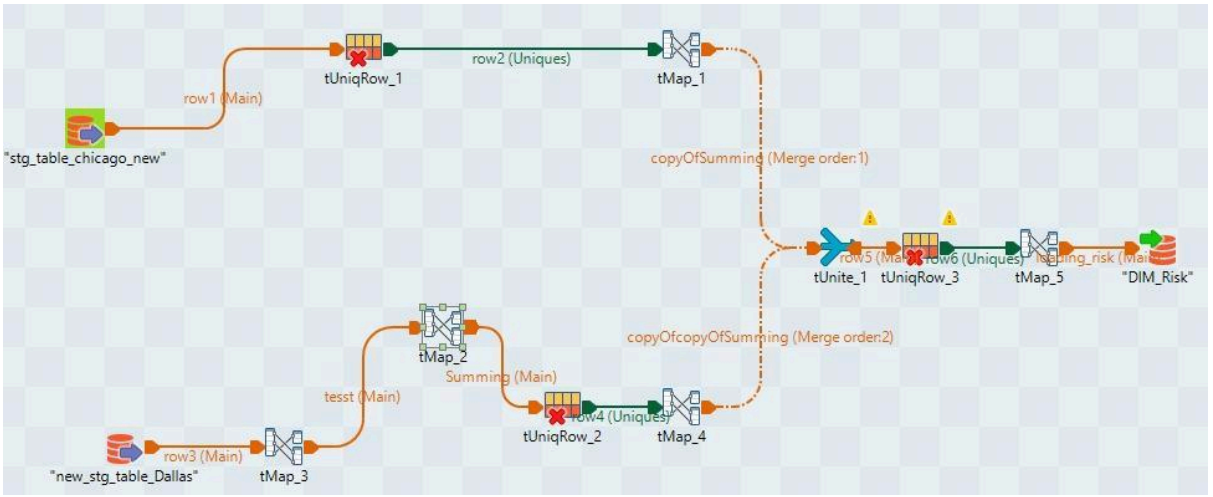
Because there was no direct 'results' column in a dataset about Dallas inspections, a novel technique was required for analysis. To determine the outcome of each inspection, a computation based on the sum of all infraction points was used. The inspection findings were determined using the following criteria: 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

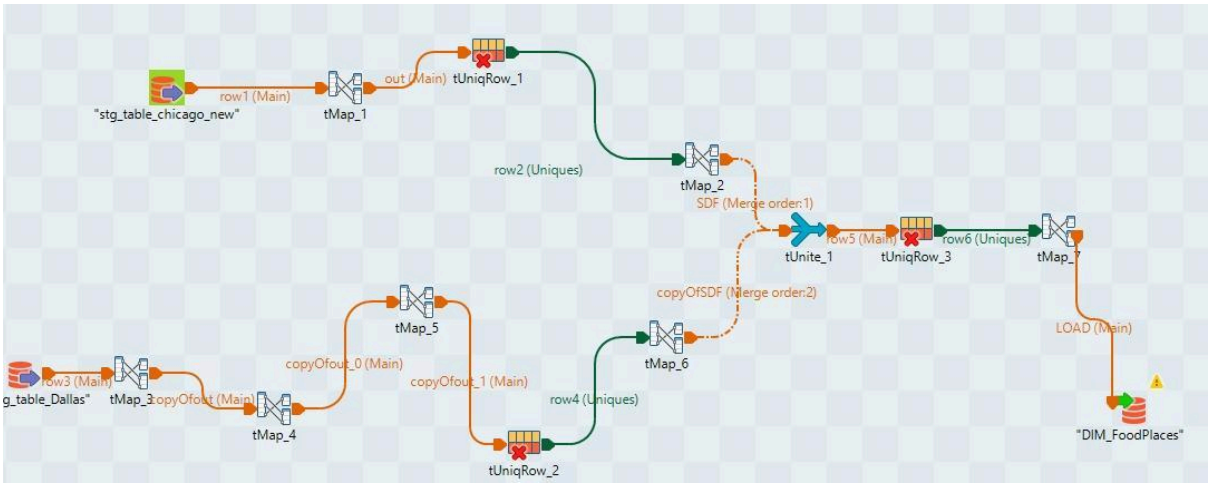


Data is collected from two staging tables, ‘stg_table_chicago_new’ and `new_stg_table_Dallas’ , in the Talend ETL job that is shown. Following a first round of data loading and cleaning, these tables represent the first holding regions. After that, `tUniqRow_1’ is used to undergo a deduplication operation 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.

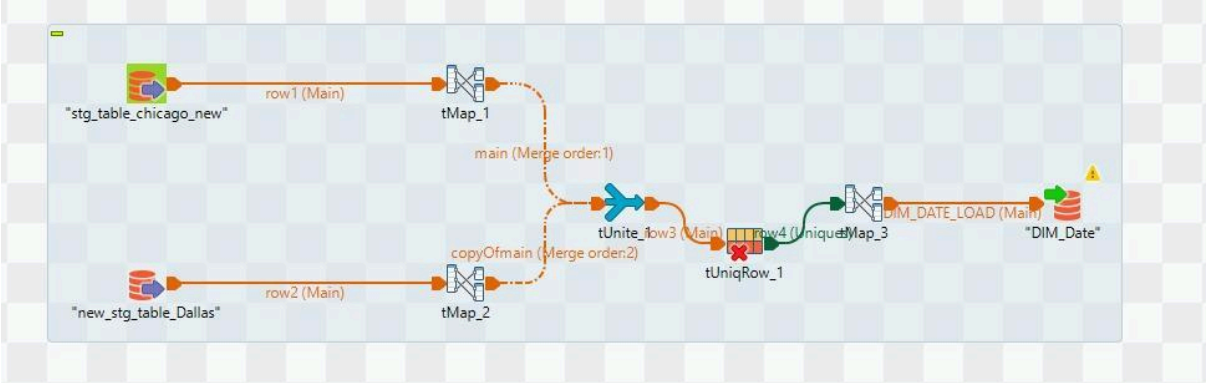
Then, in {tMap_4}, the combined data is transformed one more time before splitting in two. One route, which repeats the previous procedure to confirm data uniqueness, goes to (tUniqRow_2). (tMap_5) is the next path, and it is focused on risk analysis, as shown by the components {mapping_risk} and ‘DIM_Risk’, 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



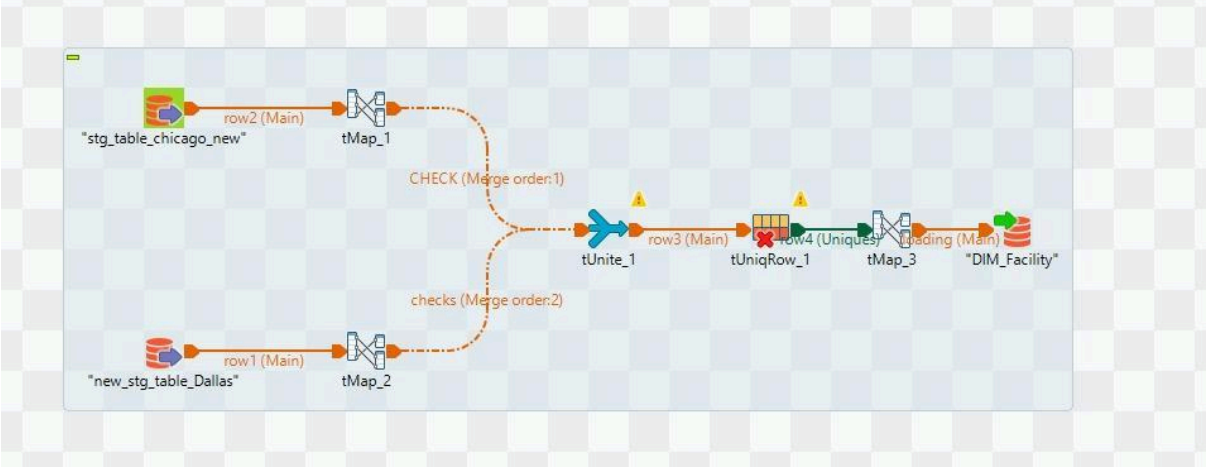
For mapping and deduplication, the Chicago data is processed by tMap_1 and tUniqRow_1, whereas the Dallas data is handled by tMap_4. The two data streams are then combined by tUnite_1 and any leftover duplicates are eliminated by tUniqRow_3, after which they undergo further modification (tMap_2 for Chicago 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.

ETL job for loading DIM_ Date from Stage tables of Chicago Dataset and Dallas Dataset



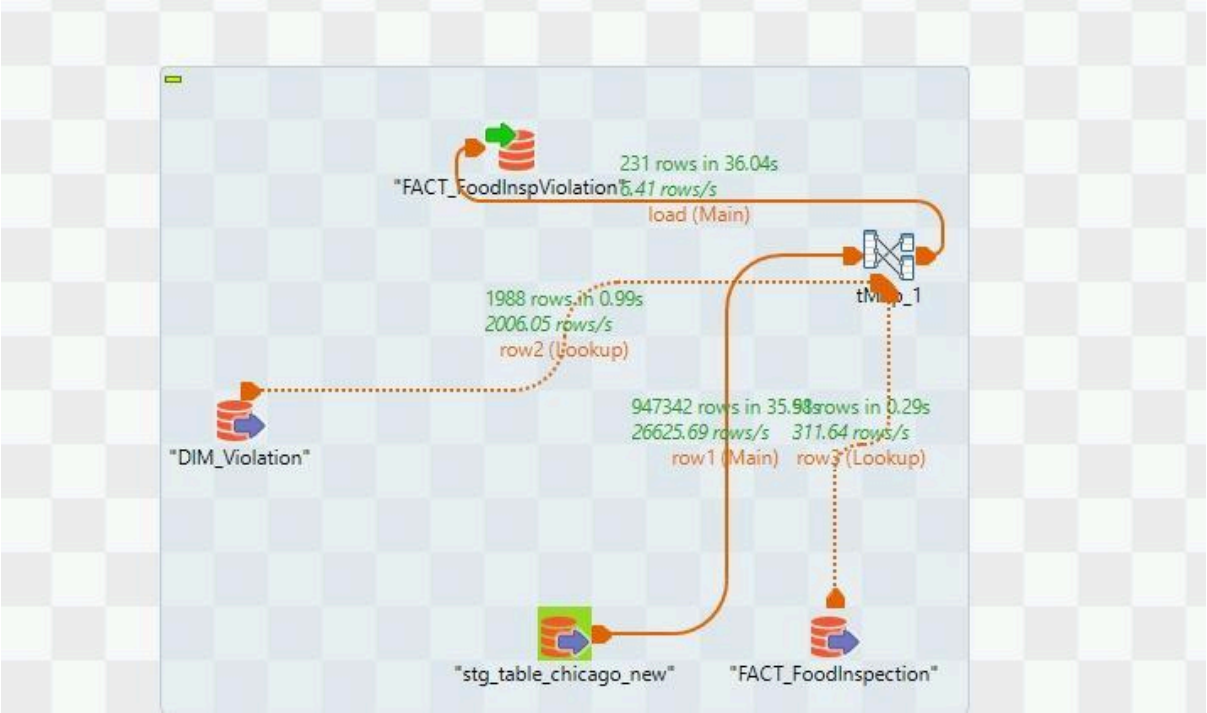
The results of these transformations are then combined in a tUnite component, indicating a consolidation of data from both cities, after being mapped using staging tables. Then, as indicated by the naming convention and the existence of a date dimension load operation (tMap_3), a deduplication step via 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



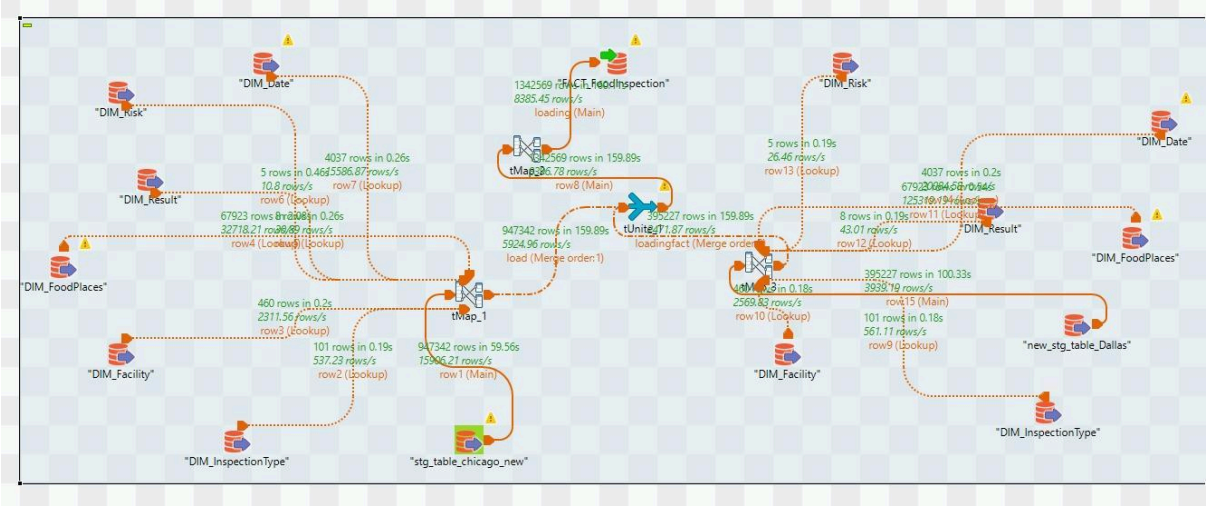
Following these changes and the use of staging tables, the data streams are merged using tUnite_1, signifying the combination of datasets from Chicago and Dallas. After the merging, duplicate entries are removed using tUniqRow_1, guaranteeing data uniqueness before it is sent to tMap_3 for additional 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



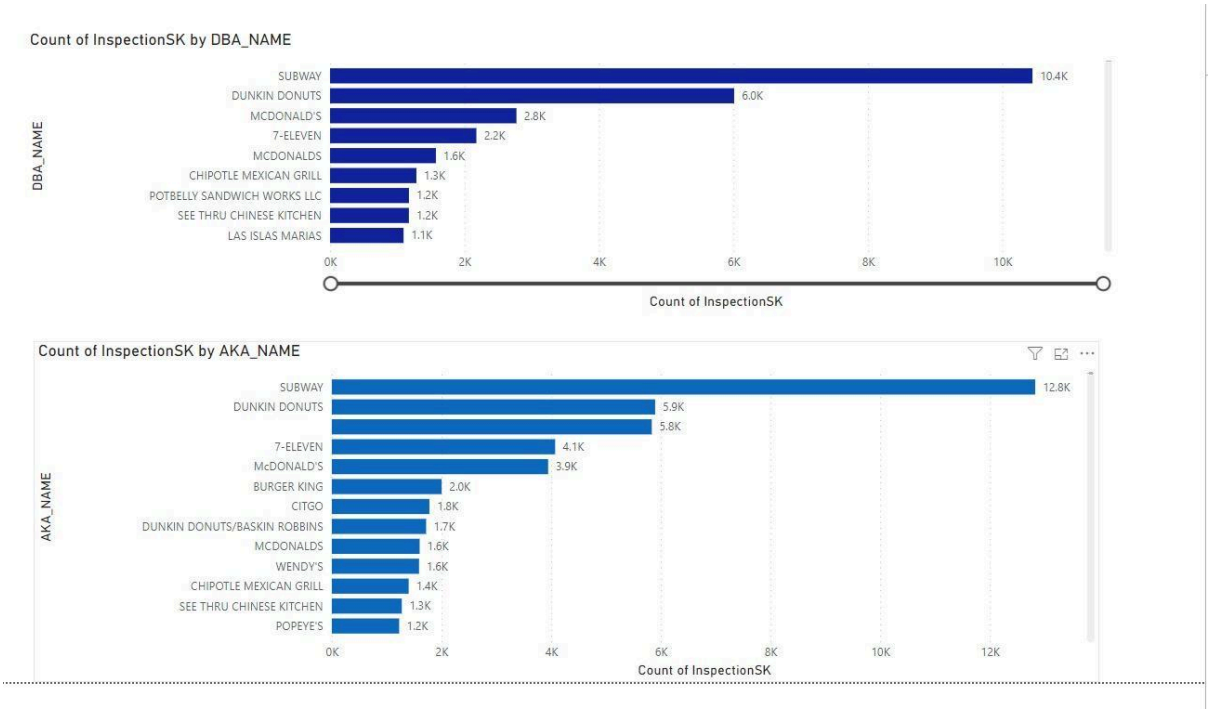
The two primary staging tables, stg_table_chicago_new and new_stg_table_Dallas, are where the data is extracted from, indicating that this task integrates data from two distinct sources. After passing through tMap_1, the Chicago data is probably transformed and made more valuable by connecting it with 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.

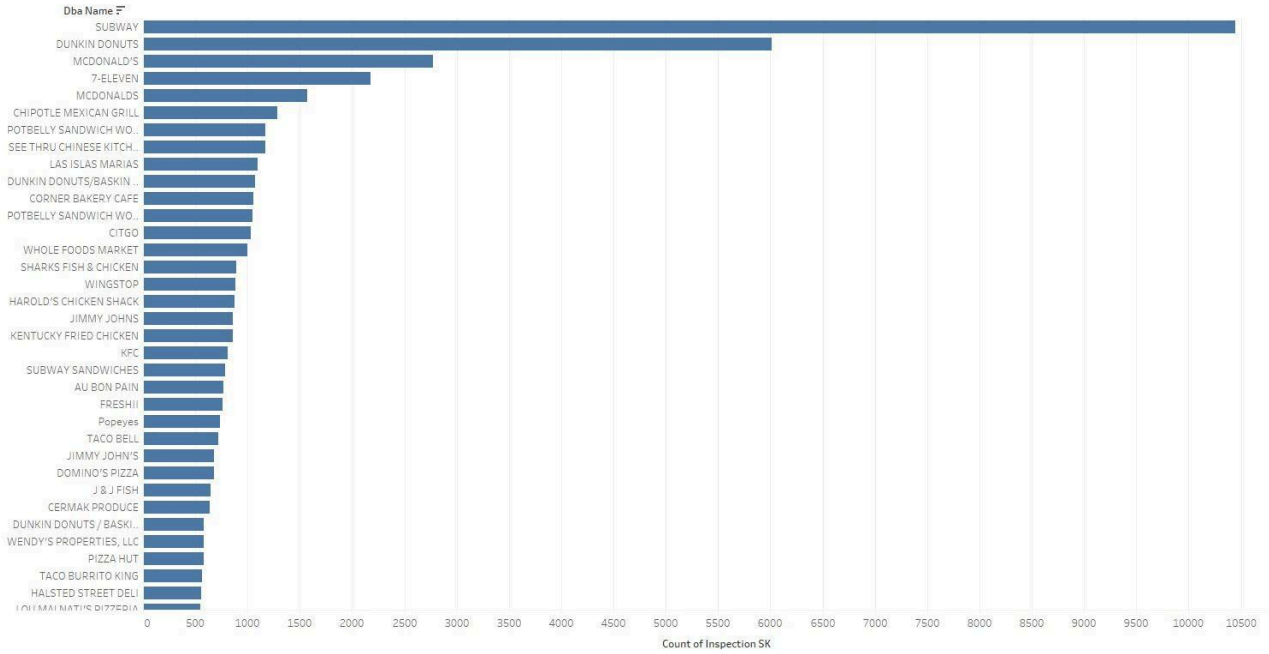
Part 04

Visualisations in Power BI and Tableau

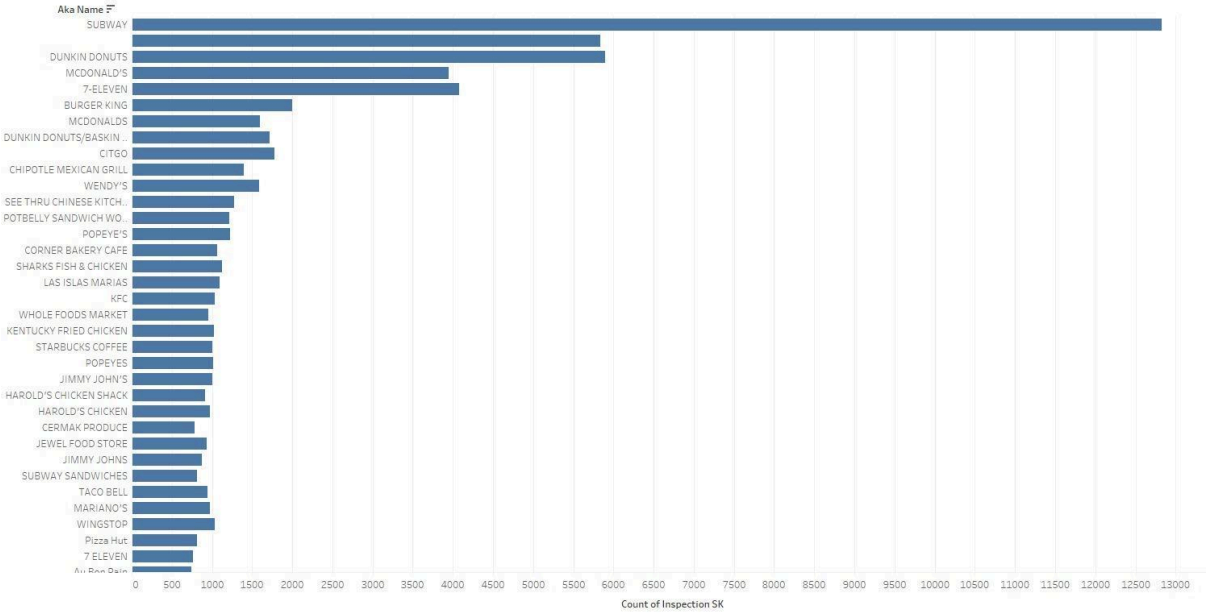
Count of InspectionSK by DBA name and AKA name


Count of InspectionSK by DBA_NAME
DBA_NAME
SUBWAY 10.4K
DUNKIN DONUTS 6.0K
MCDONALD'S 2.8K
7-ELEVEN 2.2K
MCDONALDS 1.6K
CHIPOTLE MEXICAN GRILL 1.3K
POTBELLY SANDWICH WORKS LLC 1.2K
SEE THRU CHINESE KITCHEN 1.2K
LAS ISLAS MARIAS 1.1K
0K 2K 4K 6K 8K 10K
Count of InspectionSK
Count of InspectionSK by AKA_NAME
AKA_NAME
SUBWAY 12.8K
DUNKIN DONUTS 5.9K
5.8K
7-ELEVEN 4.1K
McDONALD'S 3.9K
BURGER KING 2.0K
CITGO 1.8K
DUNKIN DONUTS/BASKIN ROBBINS 1.7K
MCDONALDS 1.6K
WENDY'S 1.6K
CHIPOTLE MEXICAN GRILL 1.4K
SEE THRU CHINESE KITCHEN 1.3K
POPEYE'S 1.2K
0K 2K 4K 6K 8K 10K 12K
Count of InspectionSK

By DBA


Dba Name
SUBWAY
DUNKIN DONUTS
MCDONALD'S
7-ELEVEN
MCDONALDS
CHIPOTLE MEXICAN GRILL
POTBELLY SANDWICH WO..
SEE THRU CHINESE KITCH..
LAS ISLAS MARIAS
DUNKIN DONUTS/BASKIN ..
CORNER BAKERY CAFE
POTBELLY SANDWICH WO..
CITGO
WHOLE FOODS MARKET
SHARKS FISH & CHICKEN
WINGSTOP
HAROLD'S CHICKEN SHACK
JIMMY JOHNS
KENTUCKY FRIED CHICKEN
KFC
SUBWAY SANDWICHES
AU BON PAIN
FRESHII
Popeyes
TACO BELL
JIMMY JOHN'S
DOMINO'S PIZZA
J & J FISH
CERMAK PRODUCE
DUNKIN DONUTS / BASKI..
WENDY'S PROPERTIES, LLC
PIZZA HUT
TACO BURRITO KING
HALSTED STREET DELI
0 500 1000 1500 2000 2500 3000 3500 4000 4500 5000 5500 6000 6500 7000 7500 8000 8500 9000 9500 10000 10500
Count of Inspection SK

By AKA


Aka Name
SUBWAY
DUNKIN DONUTS
MCDONALD'S
7-ELEVEN
BURGER KING
MCDONALDS
DUNKIN DONUTS/BASKIN ...
CITGO
CHIPOTLE MEXICAN GRILL
WENDY'S
SEE THRU CHINESE KITCH...
POTBELLY SANDWICH WO...
POPEYE'S
CORNER BAKERY CAFE
SHARKS FISH & CHICKEN
LAS ISLAS MARIAS
KFC
WHOLE FOODS MARKET
KENTUCKY FRIED CHICKEN
STARBUCKS COFFEE
POPEYES
JIMMY JOHN'S
HAROLD'S CHICKEN SHACK
HAROLD'S CHICKEN
CERMAK PRODUCE
JEWEL FOOD STORE
JIMMY JOHNS
SUBWAY SANDWICHES
TACO BELL
MARIANO'S
WINGSTOP
Pizza Hut
7 ELEVEN
0 500 1000 1500 2000 2500 3000 3500 4000 4500 5000 5500 6000 6500 7000 7500 8000 8500 9000 9500 10000 10500 11000 11500 12000 12500 13000
Count of Inspection SK

```sql
    SELECT
    fp.AKA_NAME,
    COUNT(fi.InspectionSK) AS InspectionCount
FROM
    DIM_FoodPlaces AS fp
JOIN
    FACT_FoodInspection AS fi
    ON fp.FoodPlaceSK = fi.FoodPlaceSK
GROUP BY
    fp.AKA_NAME
ORDER BY
    InspectionCount DESC;
```

% ▾

Results | Messages

AKA_NAME	InspectionCount
SUBWAY	12827
DUNKIN DONUTS	5896
	5835
7-ELEVEN	4073
MCDONALD'S	3945
BURGER KING	2003
CITGO	1780
DUNKIN DONUTS/BASKIN ROBBINS	1720
McDONALDS	1602
WENDY'S	1591
CHIPOTLE MEXICAN GRILL	1401
SEE THRU CHINESE KITCHEN	1276

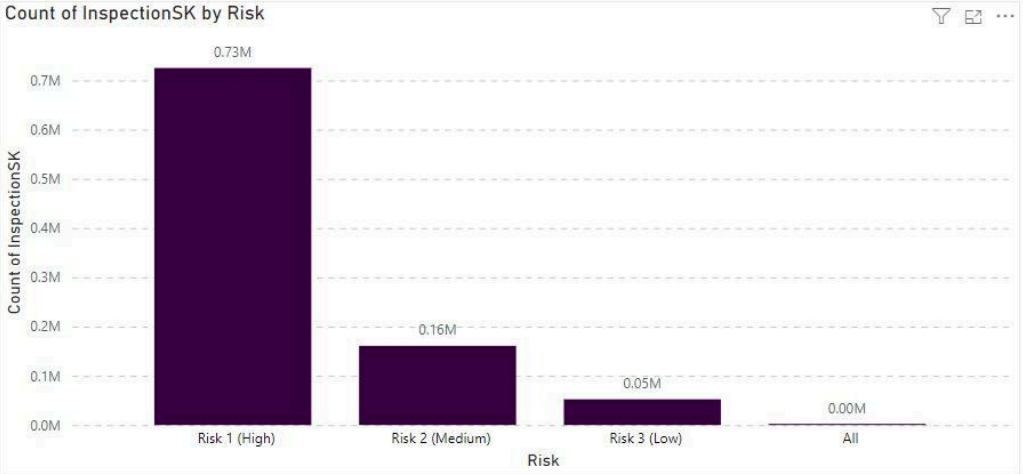
```sql
select  count(*) from FACT_FoodInspection

SELECT
    fp.DBA_NAME,
    COUNT(fi.InspectionSK) AS InspectionCount
FROM
    DIM_FoodPlaces AS fp
JOIN
    FACT_FoodInspection AS fi
    ON fp.FoodPlaceSK = fi.FoodPlaceSK
GROUP BY
    fp.DBA_NAME
ORDER BY
    InspectionCount DESC;
```

00 %

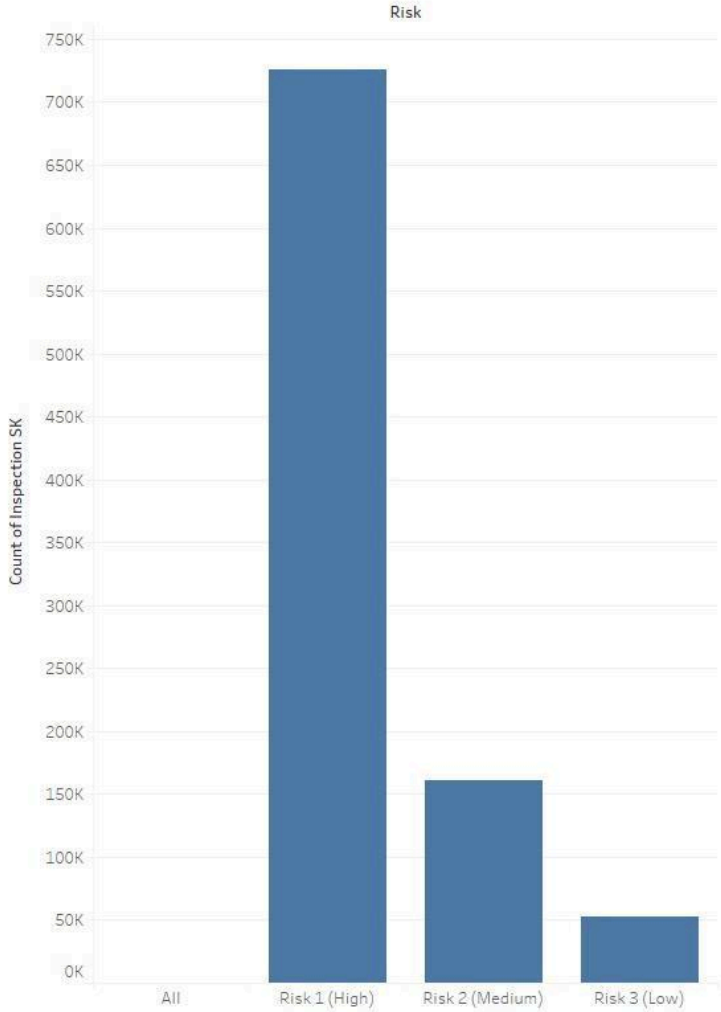
Results | Messages

	DBA_NAME	InspectionCount
1	Subway	10447
2	DUNKIN DONUTS	6010
3	McDonald's	2772
4	7-Eleven	2176
5	McDONALDS	1574
6	CHIPOTLE MEXICAN GRILL	1284
7	POTBELLY SANDWICH WORKS LLC	1173
8	See Thru Chinese Kitchen	1172
9	LAS ISLAS MARIAS	1093
10	DUNKIN DONUTS/BASKIN ROBBINS	1070
11	CORNER BAKERY CAFE	1055
12	POTBELLY SANDWICH WORKS	1046
13	CITGO	1031
14	WHOLE FOODS MARKET	999
15	SHARKS FISH & CHICKEN	890
16	WINGSTOP	881
17	HAROLD'S CHICKEN SHACK	872
18	Jimmy Johns	857
19	KENTUCKY FRIED CHICKEN	856
20	KFC	809

Count Of Inspection SK by Risk



By Risk Category

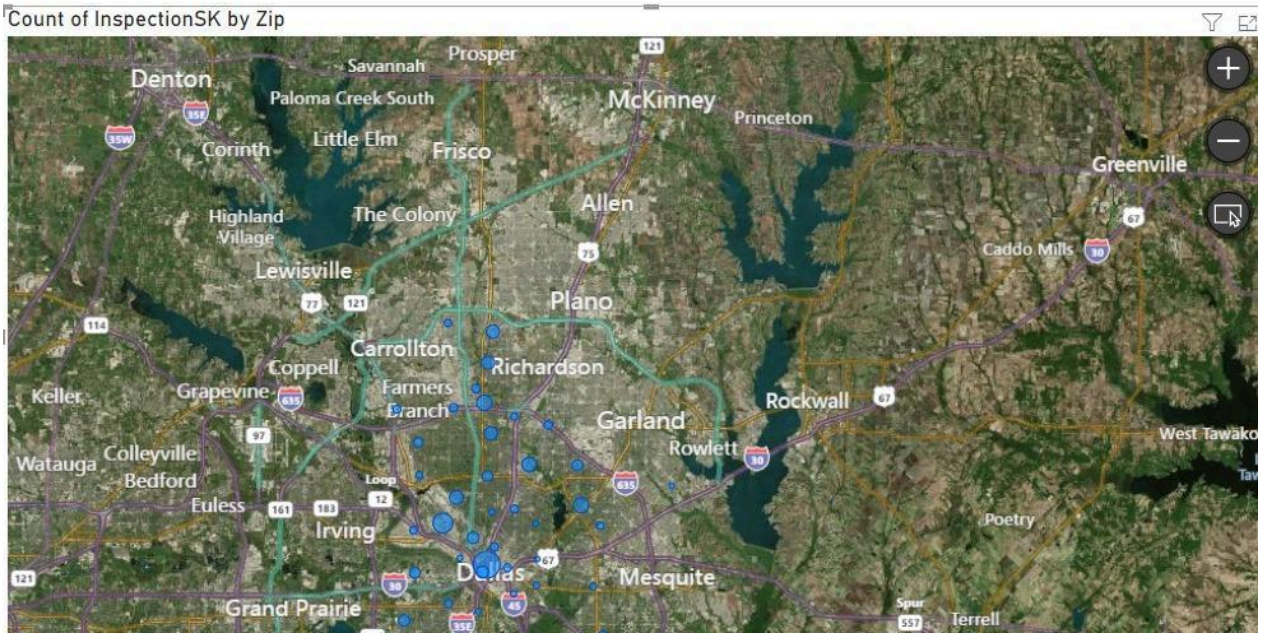


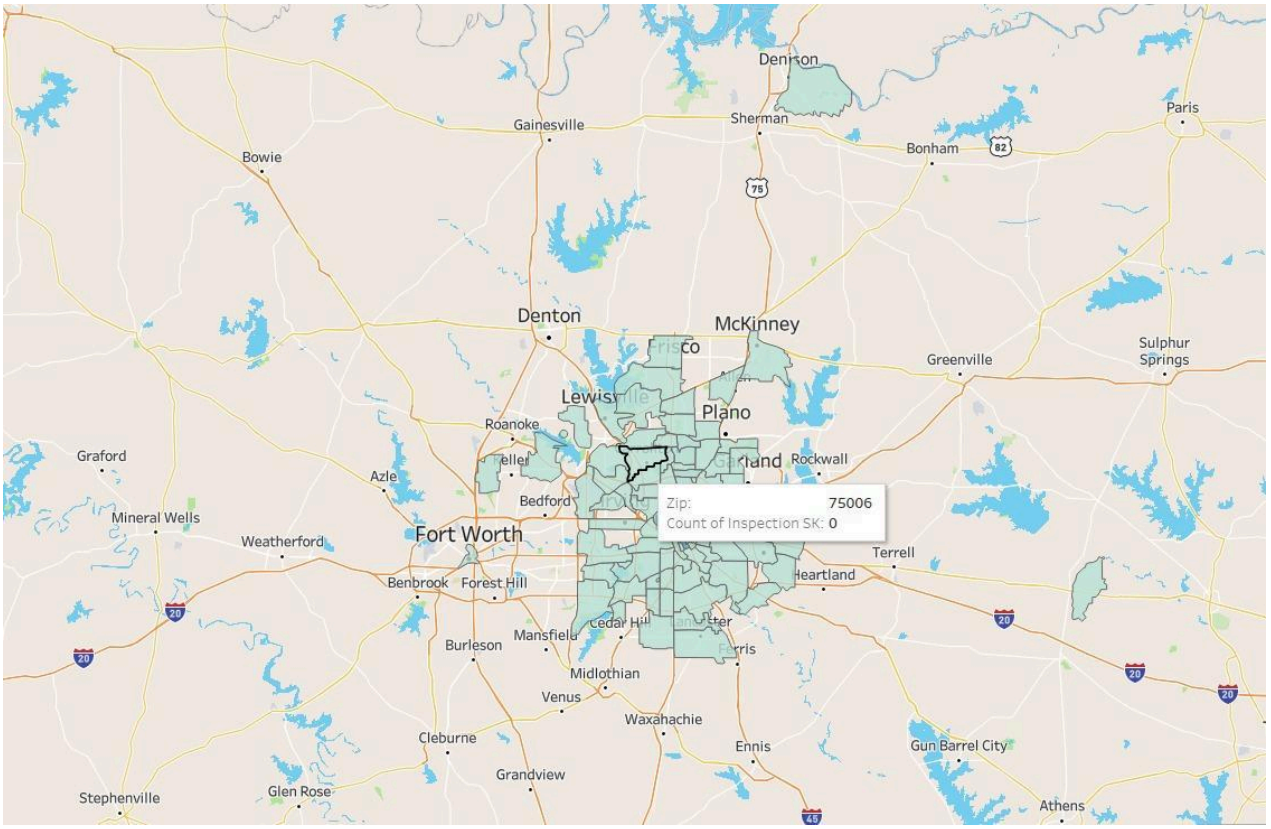
```sql
SELECT
    r.Risk,
    COUNT(fi.InspectionSK) AS InspectionCount
FROM
    DIM_Risk AS r
JOIN
    FACT_FoodInspection AS fi
    ON r.RiskSK = fi.RiskSK
GROUP BY
    r.Risk
ORDER BY
    InspectionCount DESC;
```

Results | Messages

Risk	InspectionCount
Risk 1 (High)	725209
Risk 2 (Medium)	160840
Risk 3 (Low)	52253
All	57

Count of InspectionSK by zipcode

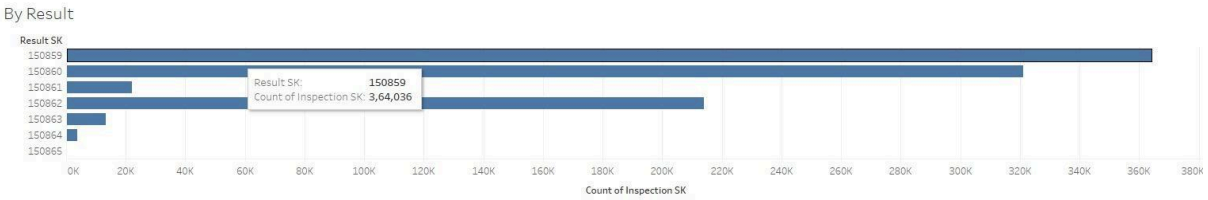




Count of InspectionSK by Result

938.36K

Count of InspectionSK

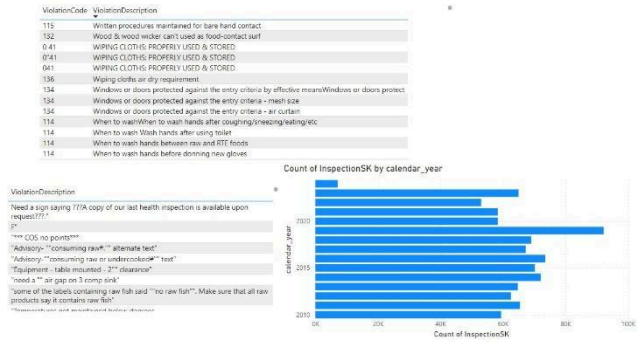



```sql
--by result
    SELECT
    dr.Result,
    COUNT(fi.InspectionSK) AS InspectionCount
FROM
    DIM_Result AS dr
JOIN
    FACT_FoodInspection AS fi
    ON dr.ResultSK = fi.ResultSK
GROUP BY
    dr.Result
```

Results | Messages

Result	InspectionCount
Pass	364036
Fail	321156
Pass w/ Conditions	213922
Out of Business	22169
No Entry	13378
Not Ready	3620
Business Not Located	78

Count by InspectionSk by Violation

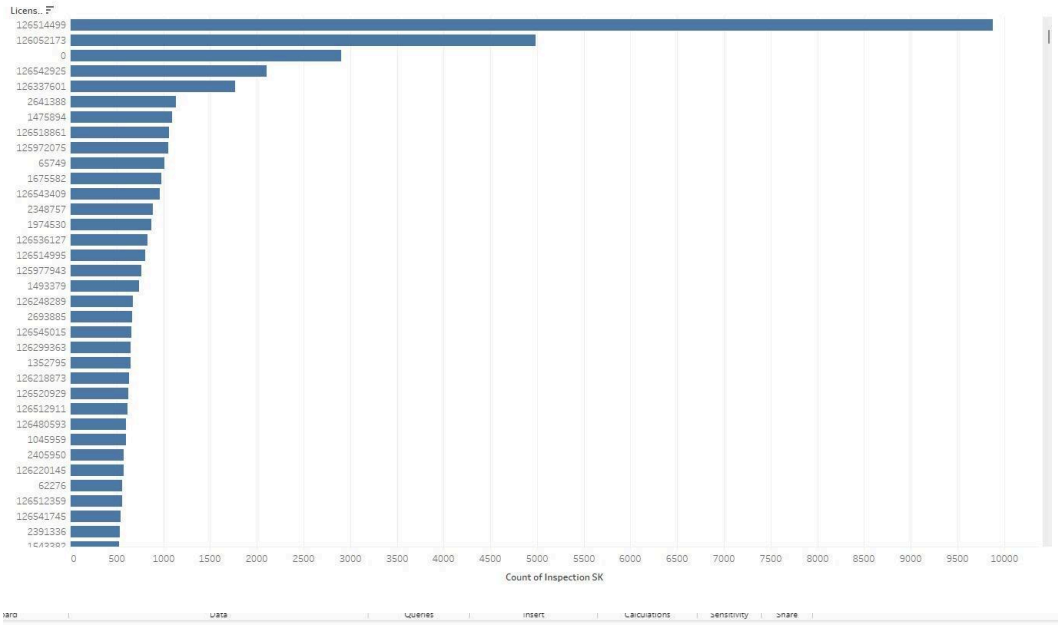


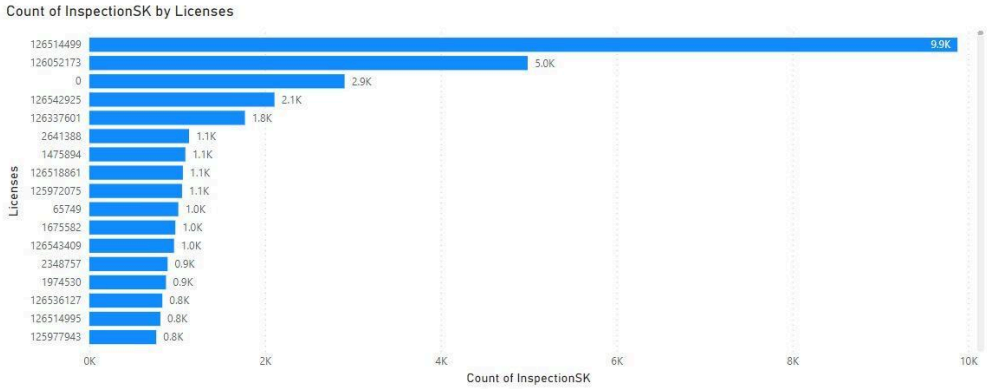
By Violation Code and Description

Violation Code	Violation Description
1Sanitize all Equipment	Sanitize all Equipment
1Prevent contamination b..	Prevent contamination by..
1Outdoor bars single serv..	Outdoor bars single servi..
1Outdoor bars dispense ic..	Outdoor bars dispense ice..
1Hair restraints/exer	1Outdoor bars dispense ice thr
1HACCP-Plan submitte	
1Cleaning floors;spills, dri..	Cleaning floors;spills, drip..
19	Outfitter Operations
	Reduced Oxygen HACCP P..
	Reduces oxygen package ..
	Reduced Oxygen HACCP Pl..
17901	YOUR BUSSINESS NAME ..
17	DATE ALL FOOD PRODUC..
	Eggs and milk products, p..
	Private homes/ living or s..
	Self-Service Food Market
	Eggs and milk products, p..
	Private homes/ living or sl..
	Self-Service Food Market
160	CITATION will be issued i..
16	Discard PHF is exceeds ti..
	must have grease trap re..
	Self-Service Food Market
	Utensils must be stored ?..
	Discard PHF is exceeds ti..
	Self-Service Food Market
150	KEEP SANITIZER IN BUCK..
	Restrict community gath..
	sanitation level weak bel..
147	Central Preparation Facili..
	Central Preparation Facili..
	Conditions of Permit-in u..
	Constructing / Remodelin..
	Food prep/utensil wash/..
	Handler-Certificate Not o..
	Handwashing signage
	Health permit posted

Count of InspectionSK by Licenses

By License



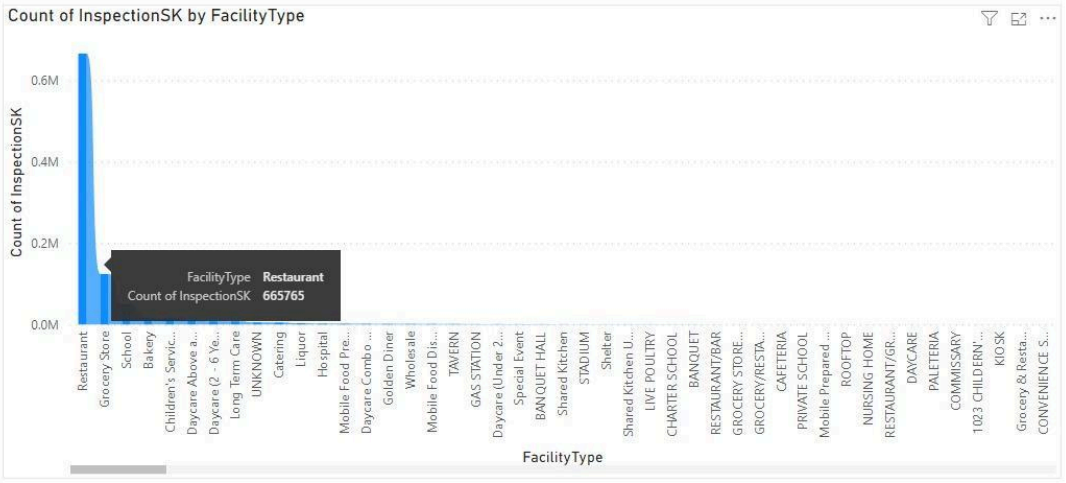



```sql
    --by facility type
    SELECT
    f.FacilityType,
    COUNT(fi.InspectionSK) AS InspectionCount
FROM
    DIM_Facility AS f
JOIN
    FACT_FoodInspection AS fi
    ON f.FacilitySK = fi.FacilitySK
GROUP BY
    f.FacilityType
ORDER BY
    InspectionCount DESC;
```

Results | Messages

Licenses	InspectionCount
126514499	9878
126052173	4988
0	2905
126542925	2107
126337601	1771
2641388	1135
1475894	1093
126518861	1065
125972075	1055
65749	1013
1675582	979
126543409	964
2348757	891
1974530	873
126536127	831
126514995	809
125977943	762
1493379	741
126248289	678
2693885	668

Count of Inspection SK by Facility Type



938.36K

Count of InspectionSK

By Facility Type

```sql
--by facility type
SELECT
    f.FacilityType,
    COUNT(fi.InspectionSK) AS InspectionCount
FROM
    DIM_Facility AS f
JOIN
    FACT_FoodInspection AS fi
    ON f.FacilitySK = fi.FacilitySK
GROUP BY
    f.FacilityType
ORDER BY
    InspectionCount DESC;
```

Results | Messages

	FacilityType	InspectionCount
	Restaurant	665765
	Grocery Store	124173
	School	50231
	Bakery	15407
	Children's Services Facility	14914
	Daycare Above and Under 2 Years	10183
	Daycare (2 - 6 Years)	8058
	Long Term Care	7893
	UNKNOWN	5222
0	Catering	5034
1	Liquor	3115
2	Hospital	2384
3	Mobile Food Preparer	2144
4	Daycare Combo 1586	1844
5	Golden Diner	1729
6	Wholesale	1596
7	Mobile Food Dispenser	1433
8	TAVERN	1331
9	GAS STATION	880
0	Daycare (Under 2 Years)	802

```sql
--by inspection type
SELECT
    dit.InspectionType,
    COUNT(fi.InspectionSK) AS InspectionCount
FROM
    DIM_InspectionType AS dit
JOIN
    FACT_FoodInspection AS fi
    ON dit.InspectionTypeSK = fi.InspectionTypeSK
GROUP BY
    dit.InspectionType
ORDER BY
    InspectionCount DESC;
```

Results | Messages

InspectionType	InspectionCount
Canvass	512099
Complaint	133040
License	105313
Canvass Re-Inspection	86890
Complaint Re-Inspection	34531
Short Form Complaint	24834
License Re-Inspection	23041
Suspected Food Poisoning	4951
Non-Inspection	3073
Tag Removal	2197
License-Task Force	1952
Consultation	1937
Recent Inspection	1563
Suspected Food Poisoning Re-inspection	697
Complaint-Fire	634
Task Force Liquor 1475	583
Short Form Fire-Complaint	280
Special Events (Festivals)	182
Complaint-Fire Re-inspection	143