

Midterm Project Report

Data Profiling: Chicago	3
Data Profiling: Dallas	5
Data Staging	7
Dimensional Model	9
Talend Workflow	10
Data Architecture	11
Dimensions	11
1. AddressDim	11
2. DimDate	12
3. DimInspectionType	12
4. DimViolations	13
5. FacilityDim	14
Fact	15
1. InspectionFact	15
6. BI Visualization	16
Tableau Dashboards:	16
PowerBI Dashboard	17
7. Testing	19
1. Inspection count vs years	19
2. Inspection Results vs year	20
3. Inspection Result VS Location (Dallas)	21
4. Inspection Result VS Location (Chicago)	22
5. Top 10 Restaurant Name vs TVS	23
6. Facility Type vs Inspection Result	24

Data Profiling: Chicago

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Chicago_YData.ipynb
- File List:** Chicago_YData.ipynb, Dallas_YData.ipynb, Python
- Toolbar:** Search, Select Kernel, etc.
- Code Cells:**
 - Cell 4:

```
import pandas as pd
from ydata_profiling import ProfileReport
```

 (Output: DeprecationWarning about pandas 3.0)
 - Cell 5:

```
df = pd.read_csv("/Users/ritwikkirgi/Downloads/Food_Inspections_20240221.csv")
```
 - Cell 6:

```
profile = ProfileReport(df, title="Profiling Report_Chicago")
```
 - Cell 7:

```
profile.to_file("Chicago_Report.html")
```

 (Output: Summarize dataset warning)
- Status Bar:** Summarizing dataset, 0% | 17/57 [01:07:00:01] > 224+ / Missing diagram.html

Handling Multi-Valued Attributes and Data Type Conversion:

Multi-Valued Attributes Handling:

Identification: Through dataset examination, we can pinpoint columns containing multi-valued attributes. These are usually in columns listing items separated by delimiters, like violation codes or service categories.

Evaluation: Understand the importance of each multi-valued attribute within your analysis framework. Determine if it's necessary to disaggregate these attributes for detailed analysis or if they can be analyzed as a whole.

Transformation Strategies: For columns requiring disaggregation, select a suitable method. Options include:

Column Splitting: If values are few and uniform, splitting into distinct columns may work.

Row Expansion: To analyze values individually, transform each value into a separate row, replicating other column data for context.

Aggregation: Summarizing these attributes could be useful, particularly for compliance or violation data, through methods like counting occurrences or statistical analysis.

Dataset Structure: Our dataset doesn't have traditional multi-valued attributes but rather distributes violation information across various columns in a 'wide' format.

Violation Data Handling: To analyze violations, restructure the data so each violation has its own row, maintaining a link to its inspection.

Data Type Conversion:

- **Inspection Date:** Convert string-based dates to datetime format for easier chronological analysis.
- **Numeric Data:** Inspection scores and street numbers are already integers. For columns like inspection year, convert to integers or maintain as is, based on analysis needs. Latitude and longitude should be split into numeric columns for geospatial analysis. Convert any numeric values stored as strings to their appropriate numeric types.
- **Categorical Data:** Convert columns representing categories into Pandas' categorical type for efficiency and faster category operations.

Conversion Processes:

Date and Time: Change any date or time columns from strings to Python's datetime format to enhance time-series analysis and manipulation.

Numerical Conversion: Adjust columns with numeric values stored as strings to the correct numeric data types for accurate analysis.

Categorical Conversion: Change columns with categorical data to Pandas' categorical type to save on memory and simplify category-based analysis.

Post-Conversion Validation: Conduct consistency checks post-transformation to ensure data integrity and correctness, using data validation techniques or summary statistics to verify the success of the conversions.

Data Profiling: Dallas

The screenshot shows a Jupyter Notebook interface with two open files: 'Chicago_YData.ipynb' and 'Dallas_YData.ipynb'. The current cell is running 'Dallas_YData.ipynb'. The code in the cell is as follows:

```
import pandas as pd
from ydata_profiling import ProfileReport
```

Cell [4] output:

```
✓ 2.4s
```

Output:

```
[var/folders/0d/g9bv54cj2hbcc0/v9f8067f00000gn/T/ipykernel_3447/2829373970.py:1: DeprecationWarning: Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0), (to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries) but was not found to be installed on your system.
If this would cause problems for you, please provide us feedback at https://github.com/pandas-dev/pandas/issues/5446
import pandas as pd
/opt/homebrew/lib/python3.11/site-packages/tqdm/auto.py:21: TqdmWarning: IPProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user\_install.html
from .autonotebook import tqdm as notebook_tqdm
```

Cell [8] output:

```
✓ 3.5s
```

Cell [10] output:

```
✓ 0.0s
```

Cell [11] output:

```
✓ 1m 15.6s
```

Output:

```
[...]
Summarize dataset: 79% | 19/24 [01:01<01:12, 14.51s/it, Calculate auto correlation] /opt/homebrew/lib/python3.11/site-packages/ydata_profiling/model/correlations.py:66: UserWarning: To hide this warning, disable the calculation (using `df.profile_report(correlations='auto': {'calculate': False}))` If this is problematic for your use case, please report this as an issue: https://github.com/ydataai/ydata-profiling/issues (include the error message: 'could not convert string to float: 'Pass'')
warnings.warn(
Summarize dataset: 90% | 47/52 [01:07<00:01, 3.33it/s, Missing diagram heatmap] /opt/homebrew/lib/python3.11/site-packages/seaborn/matrix.py:260: FutureWarning: Feature annotation = ('{:.' + self fmt + '}').format(val)
annotation = ('{:.' + self fmt + '}').format(val)
/opt/homebrew/lib/python3.11/site-packages/ydata_profiling/model/missing.py:78: UserWarning: There was an attempt to generate the Heatmap missing values diagrams, but this failed.
```

Bottom status bar:

```
Ln 1, Col 1 Spaces: 4 LF Cell 1 of 5 ⌂ ↵
```

Handling Multi-Valued Attributes:

For columns with potential multi-valued attributes, such as violation lists or service offerings, the handling strategy will vary based on the analysis objectives:

Column Division: If the data within these attributes is uniform and limited, dividing it into distinct columns may facilitate easier analysis.

Row Expansion: When each value needs individual consideration, expand this data across multiple rows. Ensure to replicate other relevant column data for comprehensive context.

Data Summarization: Sometimes, summarizing the data (e.g., counting occurrences or detecting specific keywords) might offer more analytical value than detailed individual analysis.

Data Type Adjustments:

Adapting data types is crucial for enhancing data usability and accuracy in analysis, particularly for typical columns in inspection datasets:

Inspection Date:

Initial Type: Often stored as a string.

Adjustment: Convert to datetime to enable detailed time-series investigations.

Inspection Score:

Initial Type: May appear as a string due to data inconsistencies.

Adjustment: Convert to a numerical format (integer or float) to normalize data formatting.

Address Information (e.g., Street Number, Name):

Initial Type: Stored as strings.

Adjustment: Maintain as strings but standardize formatting and casing for uniformity.

Violation Codes/Points:

Initial Type: Possibly mixed or strings, influenced by missing data or textual annotations.

Adjustment: Convert points to numerical formats when feasible. For violation codes with categorical or multi-valued characteristics, strategize their representation (e.g., dividing into columns or binary encoding) tailored to your analytical needs.

Categorical Columns (e.g., Inspection Type):

Initial Type: String format.

Adjustment: Change to categorical data type in pandas to boost efficiency and support category-based analyses.

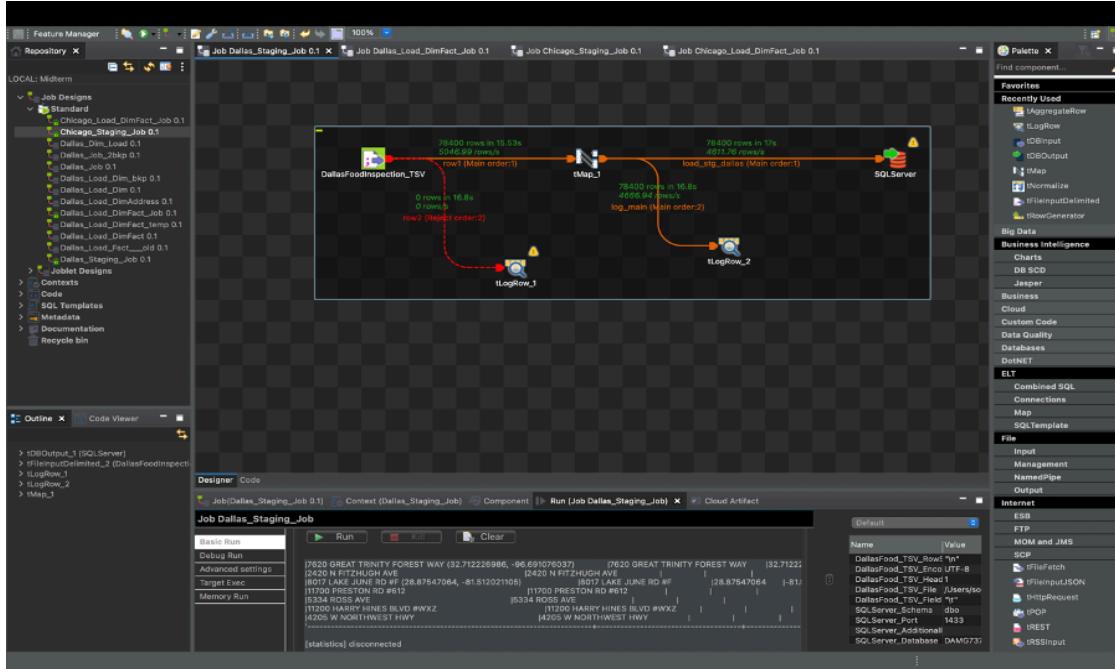
Geolocation Data (Latitude, Longitude):

Initial Type: Strings or a mixed format.

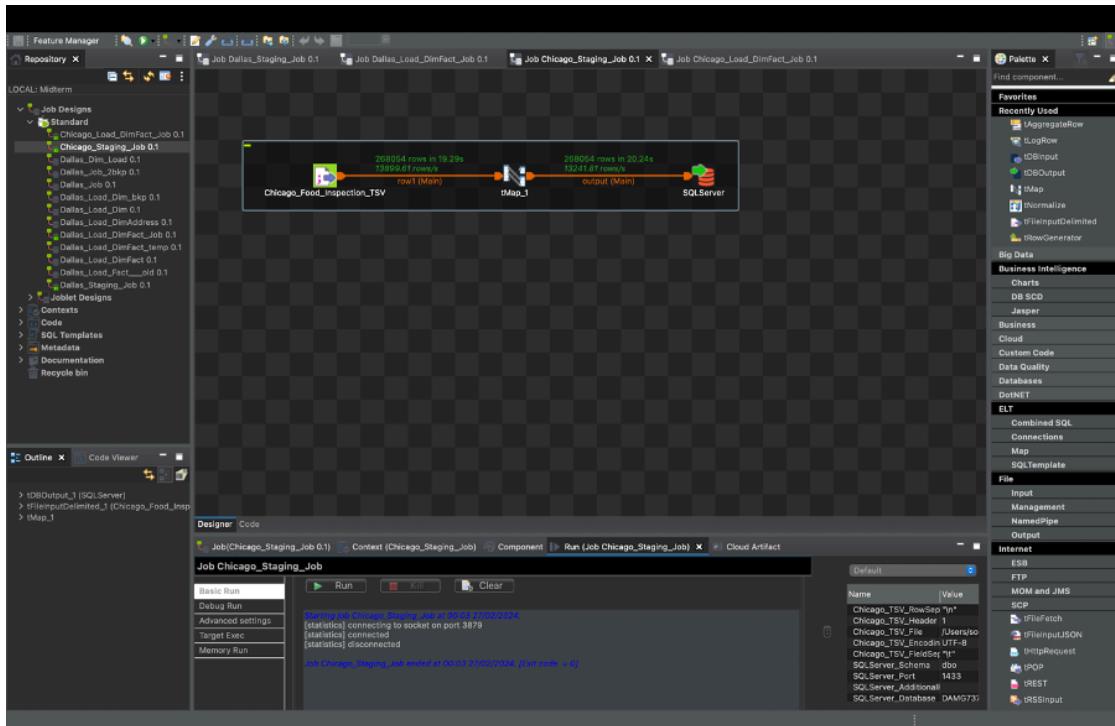
Adjustment: Convert to a double precision format to ensure precision for mapping or spatial analysis purposes.

Data Staging

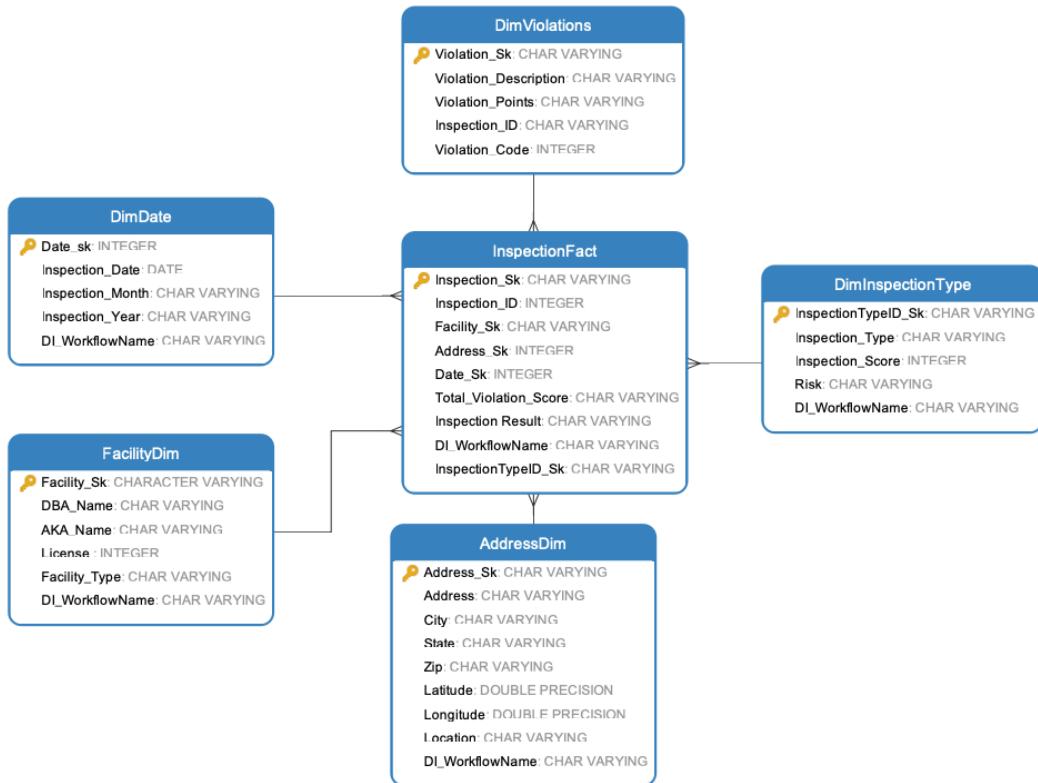
Dallas data staging:



Chicago data staging:

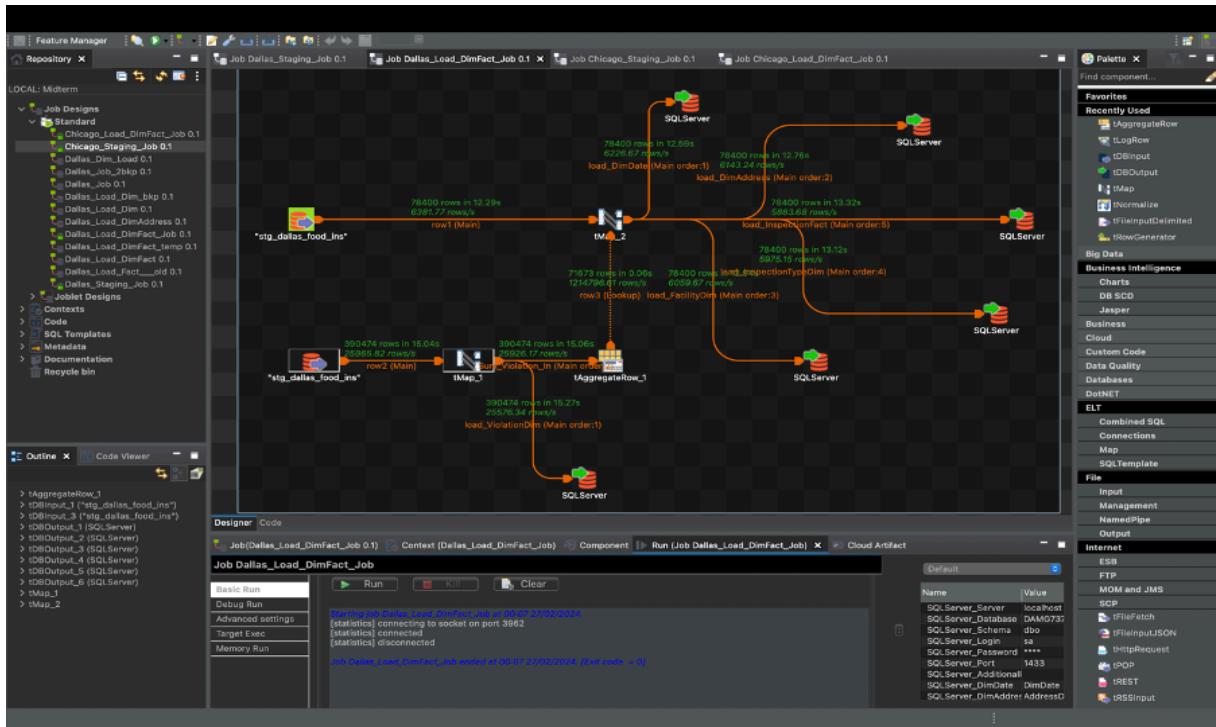


Dimensional Model

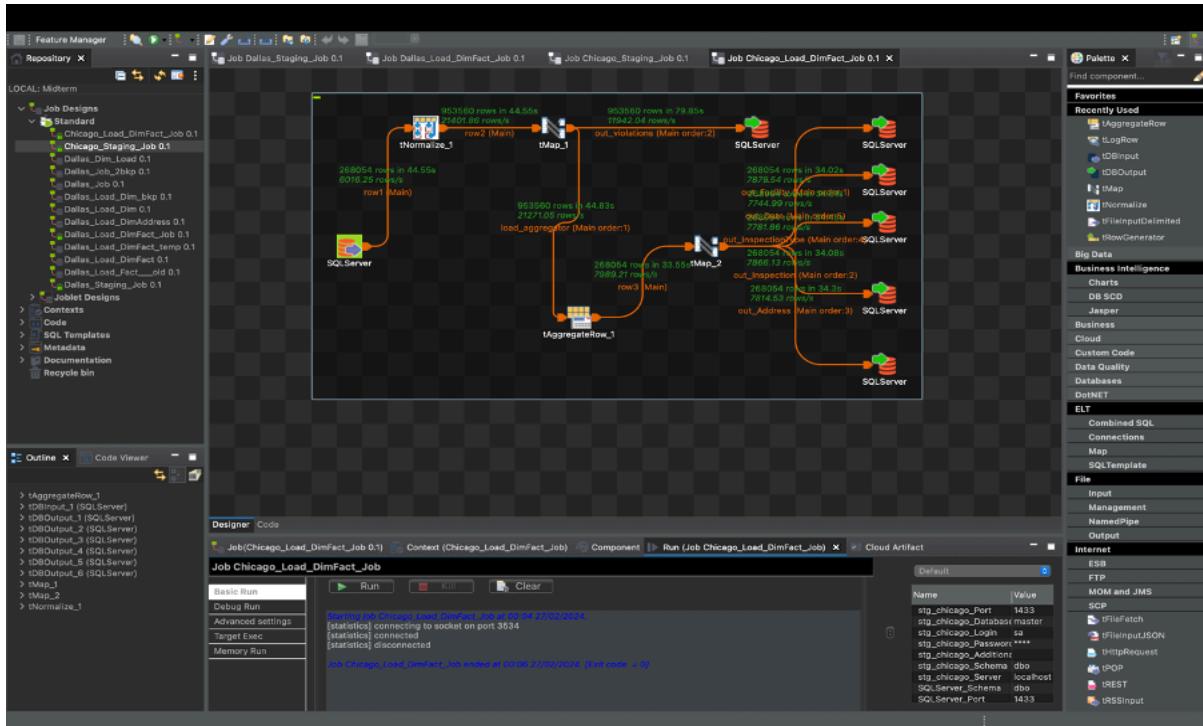


Talend Workflow

Dallas data loading:



Chicago data loading:



Data Architecture

Dimensions

1. AddressDim

```
CREATE TABLE [dbo].[AddressDim] (
    [Address_Sk]           VARCHAR (10) NOT NULL,
    [Address]               VARCHAR (200) NULL,
    [City]                  VARCHAR (100) NULL,
    [State]                 VARCHAR (10) NULL,
    [Zip]                   VARCHAR (10) NULL,
    [Latitude]              VARCHAR (80) NULL,
    [Longitude]             VARCHAR (80) NULL,
    [Location]              VARCHAR (200) NULL,
    [DI_WorkflowName]       VARCHAR (100) NULL,
    PRIMARY KEY CLUSTERED ([Address_Sk] ASC)
);
```

The screenshot shows the SSMS interface with the following details:

- Connections:** Docker, SQLQuery_7 - SQL Server Docker, SQLQuery_3 - disconnected, SQLQuery_12 - disconnected 1, SQLQuery_13 - SQL Server Docker.
- Servers:** DAMG7370
- Tables:** dbo.AddressDim
- Results Grid:** Displays 14 rows of address data. The columns are: Address_Sk, Address, City, State, Zip, Latitude, Longitude, and Location. The last row shows a count of 346454.

Address_Sk	Address	City	State	Zip	Latitude	Longitude	Location
c10	1335 W WRIGHTWOOD AVE	CHICAGO	IL	60614	41.928757	-87.66237	(41.92875605419546, -87.66237147194119)
c100	3847 S MORGAN ST	CHICAGO	IL	60609	41.82415	-87.65066	(41.82415036365103, -87.6506580321786)
c1000	1104-1106 W ARGYLE ST	CHICAGO	IL	60640	41.973377	-87.658	(41.9733789982834, -87.6579985526953)
c10000	5752 N CALIFORNIA AVE	CHICAGO	IL	60659	41.98653	-87.69942	(41.9865305618974, -87.69941716278377)
c100000	1309 E 53RD ST	CHICAGO	IL	60615	41.79937	-87.59462	(41.79936856037866, -87.59461606317524)
c100001	3533 N WESTERN AVE	CHICAGO	IL	60618	41.94591	-87.68809	(41.94591210546262, -87.6880803317516)
c100002	5419 W DEVON AVE	CHICAGO	IL	60646	41.99722	-87.76456	(41.99721018706988, -87.7645546496093)
c100003	108 N STATE ST	CHICAGO	IL	60602	41.883423	-87.62802	(41.88342263701488, -87.62802165207536)
c100004	4434 S COTTAGE GROVE AVE	CHICAGO	IL	60653	41.813942	-87.66681	(41.8139405139736, -87.6668064252573)
c100005	3151 W WILNU St (234N)	CHICAGO	IL	60612	41.885296	-87.70582	(41.88529401072774, -87.70581859311523)
c100006	735 S CALIFORNIA AVE	CHICAGO	IL	60612	41.87187	-87.6959	(41.87186697013989, -87.69589834683019)
c100007	401 N STATE ST	CHICAGO	IL	60654	41.889267	-87.62787	(41.88926541596091, -87.62786765786387)
c100008	3142 S MORGAN ST	CHICAGO	IL	60608	41.83673	-87.65126	(41.836731631091354, -87.651276122364)
	count						
		346454					

Bottom status bar: Ln 2, Col 51 Spaces: 4 UTF-8 LF 346,465 rows MSSQL 00:00:01 localhost : DAMG7370

2. DimDate

```
CREATE TABLE [dbo].[DimDate] (
    [Date_Sk]           VARCHAR (10) NOT NULL,
    [Inspection_Date]  DATETIME      NULL,
    [Inspection_Month] VARCHAR (23)  NULL,
    [Inspection_Year]   VARCHAR (6)   NULL,
    [Inspection_Day]    VARCHAR (15)  NULL,
    [DI_WorkflowName]   VARCHAR (100) NULL,
    PRIMARY KEY CLUSTERED ([Date_Sk] ASC)
);
```

The screenshot shows the SSMS interface with the following details:

- Connections:** Docker, SQLQuery_7 - SQL Server Docker, SQLQuery_3 - disconnected, SQLQuery_12 - disconnected, SQLQuery_13 - SQL Server Docker.
- Servers:** SQL Server Edge Docker, DAMG6210, DAMG7370.
- Databases:** System Databases, AdultLiteracy.
- Tables:** dbo.AddressDim, dbo.DimDate.
- Columns:** Date_Sk (PK, varchar(10), not null), Inspection_Date (datetime, null), Inspection_Month (varchar(23), null), Inspection_Year (varchar(6), null), Inspection_Day (varchar(15), null), DI_WorkflowName (varchar(100), null).
- Script:** The script pane contains two queries:

```
1 SELECT * FROM [dbo].[DimDate];
2 SELECT COUNT(*) as count FROM [dbo].[DimDate];
```
- Results:** The results pane displays the data from the DimDate table, showing 346454 rows. The columns are Date_Sk, Inspection_Date, Inspection_Month, Inspection_Year, Inspection_Day, and DI_WorkflowName. The data spans from 2018-06-03 to 2021-12-09.
- Messages:** No messages are present.
- Status Bar:** Ln 2, Col 47, Spaces: 4, UTF-8, LF, 346,455 rows, MSSQL, 00:00:01, localhost:DAMG7370.

3. DimInspectionType

```
CREATE TABLE [dbo].[DimInspectionType] (
    [InspectionTypeID_Sk] VARCHAR (10) NOT NULL,
    [Inspection_Type]     VARCHAR (50)  NULL,
    [Inspection_Score]    INT          NULL,
    [Risk]                VARCHAR (15)  NULL,
    [DI_WorkflowName]     VARCHAR (100) NULL,
    PRIMARY KEY CLUSTERED ([InspectionTypeID_Sk] ASC)
```

```
) ;
```

The screenshot shows the SSMS interface. The Object Explorer on the left lists databases, tables, and other objects under the 'Dim' database. The 'Tables' node for 'Dim' is expanded, showing 'DimAddress' and 'DimDate'. The 'DimDate' table is selected. The 'Columns' node for 'DimDate' is expanded, listing columns like 'Date_Sk', 'Inspection_Type', 'Inspection_Score', 'Risk', and 'DI_WorkflowName'. A context menu is open over one of the columns. The 'Results' tab is active in the center pane, displaying a grid of data from the 'DimDate' table. The grid has columns: 'Inspec...', 'Inspection_Type', 'Inspection_Score', 'Risk', and 'DI_WorkflowName'. The data shows various routine entries with scores ranging from 81 to 100 and workflow names like 'Dallas_Load_Data...'. A total count of 346454 is shown at the bottom of the grid. The status bar at the bottom right shows 'Ln 2, Col 57' and 'localhost : DAMG7370'.

Inspec...	Inspection_Type	Inspection_Score	Risk	DI_WorkflowName
2.. d15423	Routine	92	NA	Dallas_Load_Data...
2.. d15424	Routine	87	NA	Dallas_Load_Data...
2.. d15425	Routine	98	NA	Dallas_Load_Data...
2.. d15426	Routine	100	NA	Dallas_Load_Data...
2.. d15427	Routine	97	NA	Dallas_Load_Data...
2.. d15428	Routine	81	NA	Dallas_Load_Data...
2.. d15429	Routine	90	NA	Dallas_Load_Data...
2.. d1543	Routine	90	NA	Dallas_Load_Data...
2.. d15430	Routine	89	NA	Dallas_Load_Data...
2.. d15431	Routine	91	NA	Dallas_Load_Data...
2.. d15432	Routine	89	NA	Dallas_Load_Data...
2.. d15433	Routine	99	NA	Dallas_Load_Data...
2.. d15434	Routine	83	NA	Dallas_Load_Data...
2.. d15435	Routine	94	NA	Dallas_Load_Data...

4. DimViolations

```
CREATE TABLE [dbo].[DimViolations] (
    [Violations_Sk]           VARCHAR (10) NOT NULL,
    [Inspection_ID]          VARCHAR (10) NULL,
    [Violations_Code]         VARCHAR (MAX) NULL,
    [Violations_Description]  VARCHAR (MAX) NULL,
    [Violation_Points]        INT        NULL,
    [DI_CreateDate]           DATE      NULL,
    [DI_WorkflowName]         VARCHAR (100) NULL,
    PRIMARY KEY CLUSTERED ([Violations_Sk] ASC)
);
```

Violations_Sk	Inspection_ID	Violations_Code	Violations_Description	Viol
1	2579234	10907100738030	ADEQUATE HANDWASHING SINKS PROPERLY SUPPLIED AND ACC...	0
2	2579234	44	UTENSILS, EQUIPMENT & LINENS: PROPERLY STORED, DRIED...	0
3	2578857	556501114	PHYSICAL FACILITIES INSTALLED, MAINTAINED & CLEAN -...	0
4	2575938	49	NON-FOOD/FOOD CONTACT SURFACES CLEAN – Comments: MUS...	0
5	2484562	45	SINGLE-USE/SINGLE-SERVICE ARTICLES: PROPERLY STORED ...	0
6	2581150	56630311	ADEQUATE VENTILATION & LIGHTING; DESIGNATED AREAS US...	0
7	2579111	4846031533	WAREWASHING FACILITIES: INSTALLED, MAINTAINED & USED...	0
8	1532790	43	FOOD (ICE) DISPENSING UTENSILS, WASH CLOTHS PROPERLY...	0
9	1482607	NULL	NULL	0
10	1522697	NULL	NULL	0
11	1480568	NULL	NULL	0
12	2580733	2253254148655664879484491475468100541738005	PROPER COLD HOLDING TEMPERATURES – Comments: OBSERVE...	0
13	2580733	3352941738005	PROPER COOLING METHODS USED; ADEQUATE EQUIPMENT FOR...	0

5. FacilityDim

```

CREATE TABLE [dbo].[FacilityDim] (
    [Facility_Sk]      VARCHAR (10) NOT NULL,
    [DBA_Name]         VARCHAR (80) NULL,
    [AKA_Name]         VARCHAR (80) NULL,
    [License]          INT        NULL,
    [Facility_Type]    VARCHAR (50) NULL,
    [DI_WorkflowName]  VARCHAR (100) NULL,
    PRIMARY KEY CLUSTERED ([Facility_Sk] ASC)
);

```

The screenshot shows the SSMS interface with the following details:

- Connections:** LQuery_3 - disconnected, SQLQuery_12 - disconnected 1, SQLQuery_13 - SQL Se... Docker
- Database:** DAMG7370
- Query:**

```

1 SELECT * FROM [dbo].[FacilityDim];
2 SELECT COUNT(*) as count FROM [dbo].[FacilityDim];

```
- Results Grid:**

	Facility_Sk	DBA_Name	AKA_Name	License	Facility_Type	DI_WorkflowName
1	c1	RYHANAS CUISINE	RYHANAS CUISINE	2423338	Restaurant	Chicago_Job
2	c10	AVLI TAVERNA	AVLI TAVERNA	2616777	Restaurant	Chicago_Job
3	c100	VIENNA SAUSAGE	VIENNA SAUSAGE	2463873	Restaurant	Chicago_Job
4	c1000	PHO 777 RESTAURANT	PH 777 RESTAURANT	2929008	Restaurant	Chicago_Job
5	c10000	MARIE GOLD BAKE SHOPPE	MARIE GOLD BAKE SHOPPE	38776	Restaurant	Chicago_Job
6	c100000	POCKETS HYDE PARK	POCKETS	2153361	Restaurant	Chicago_Job
7	c100001	TASTY HOUSE ll	TASTY HOUSE ll	1991101	Restaurant	Chicago_Job
8	c100002	RUK SUSHI & THAI	RUK SUSHI & THAI	2583177	Restaurant	Chicago_Job
9	c100003	LIFEWAY KEFIR SHOP	LIFEWAY KEFIR SHOP	2003356	Restaurant	Chicago_Job
10	c100004	REGGIO'S PIZZA EXPRESS II	REGGIO'S PIZZA EXPRESS II	2856813	Restaurant	Chicago_Job
11	c100005	JACOB BEIDLER ELEMENTARY SCHOOL	Beidler Elementary School	22211	School	Chicago_Job
12	c100006	CALIFORNIA FOOD MART	CALIFORNIA FOOD MART	1928274	Grocery Store	Chicago_Job
13	c100007	GUS'S WORLD FAMOUS FRIED CHICKEN	GUS'S WORLD FAMOUS FRIED CHICKEN	2600522	Restaurant	Chicago_Job
14	c100008	NONNA SOLURI'S ITALIAN DELI, INC.	NONNA SOLURI'S ITALIAN DELI	2528417	Restaurant	Chicago_Job
1	346454					
- Statistics:** Ln 2, Col 61, Spaces: 4, UTF-8, LF, 346,455 rows, MSSQL, 00:00:01, localhost : DAMG7370

Fact

1. InspectionFact

```

CREATE TABLE [dbo].[InspectionFact] (
    [Inspection_Sk]           VARCHAR (10) NOT NULL,
    [Inspection_ID]           VARCHAR (10) NULL,
    [Facility_Sk]              VARCHAR (10) NOT NULL,
    [Address_Sk]               VARCHAR (10) NOT NULL,
    [date_sk]                  VARCHAR (10) NULL,
    [InspectionTypeID_Sk]     VARCHAR (10) NOT NULL,
    [Total_Violation_Score]   INT          NULL,
    [Inspection_Result]       VARCHAR (100) NULL,
    [DI_WorkflowName]          VARCHAR (100) NULL,
    PRIMARY KEY CLUSTERED ([Inspection_Sk] ASC)
);

```

Servers

```

1 SELECT * FROM [dbo].[InspectionFact];
2 SELECT COUNT(*) AS count FROM [dbo].[InspectionFact];

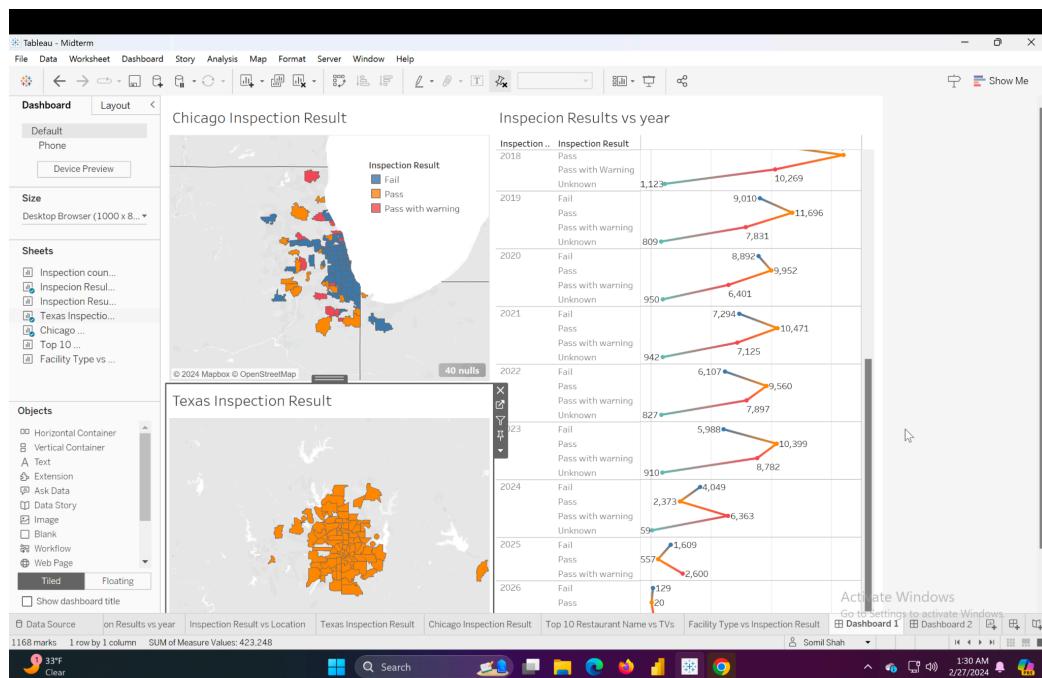
```

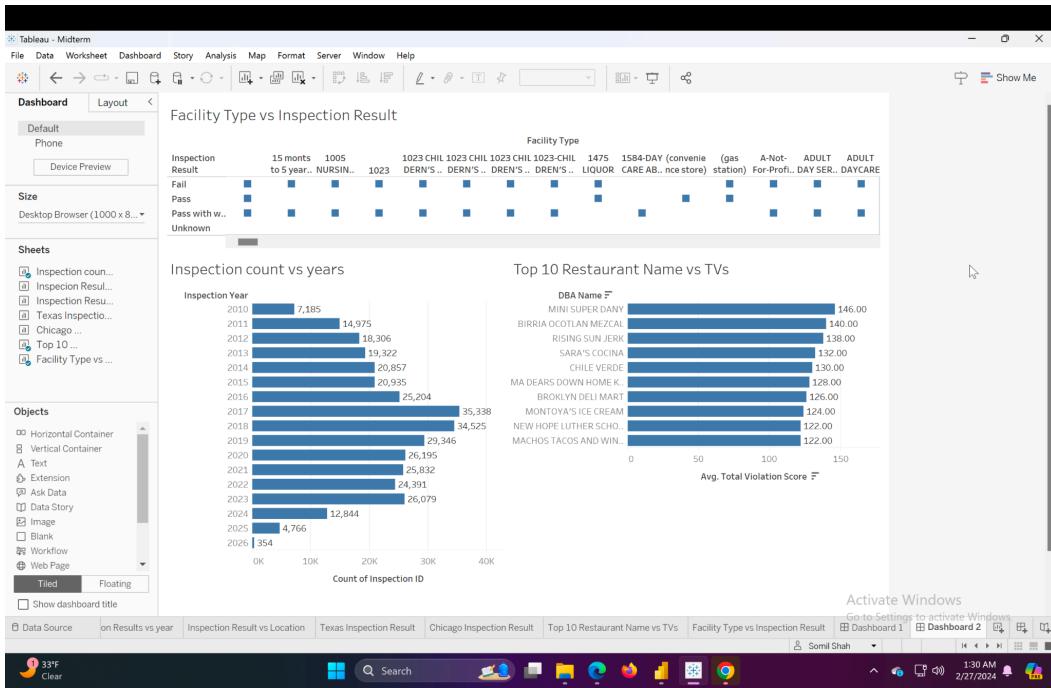
Inspection_Sk	Inspection_ID	Facility_Sk	Address_Sk	date_sk	InspectionTypeID_sk	Total_Violation_Score	Inspect
c1	2145968	c1	c1	c1	c1	32	Pass
c10	2570734	c10	c10	c10	c10	40	Pass
c100	1746253	c100	c100	c100	c100	32	Pass
c1000	2580409	c1000	c1000	c1000	c1000	38	Pass
c10000	1418654	c10000	c10000	c10000	c10000	40	Pass
c100000	1386535	c100000	c100000	c100000	c100000	26	Pass
c100001	521932	c100001	c100001	c100001	c100001	32	Pass
c100002	2565272	c100002	c100002	c100002	c100002	32	Pass
c100003	2384974	c100003	c100003	c100003	c100003	82	Fail
c100004	2587483	c100004	c100004	c100004	c100004	12	Pass
c100005	1445308	c100005	c100005	c100005	c100005	108	Fail
c100006	1324930	c100006	c100006	c100006	c100006	110	Fail
c100007	2555986	c100007	c100007	c100007	c100007	32	Pass
		count					
		1	346454				

Ln 2, Col 54 Spaces: 4 UTF-8 LF 346,455 rows MSSQL 00:00:01 localhost : DAMG7370

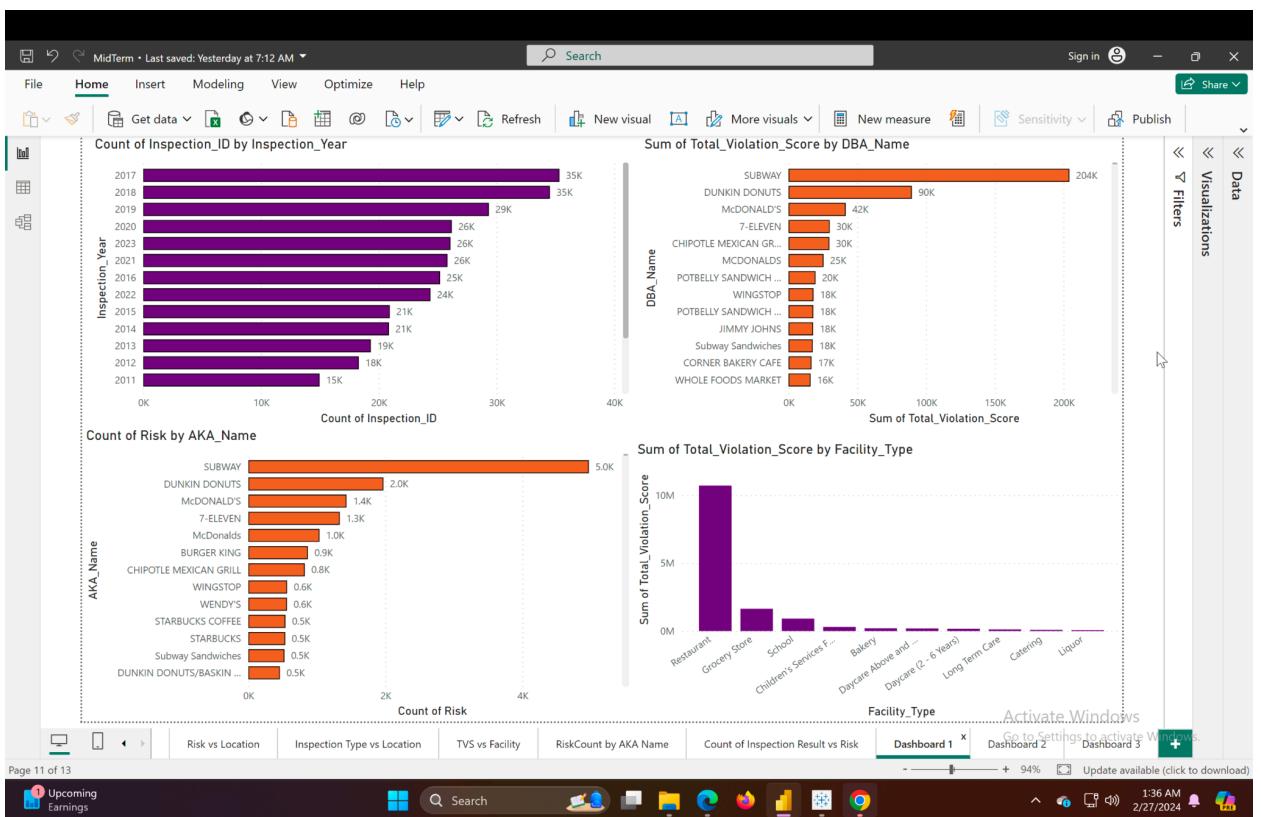
6. BI Visualization

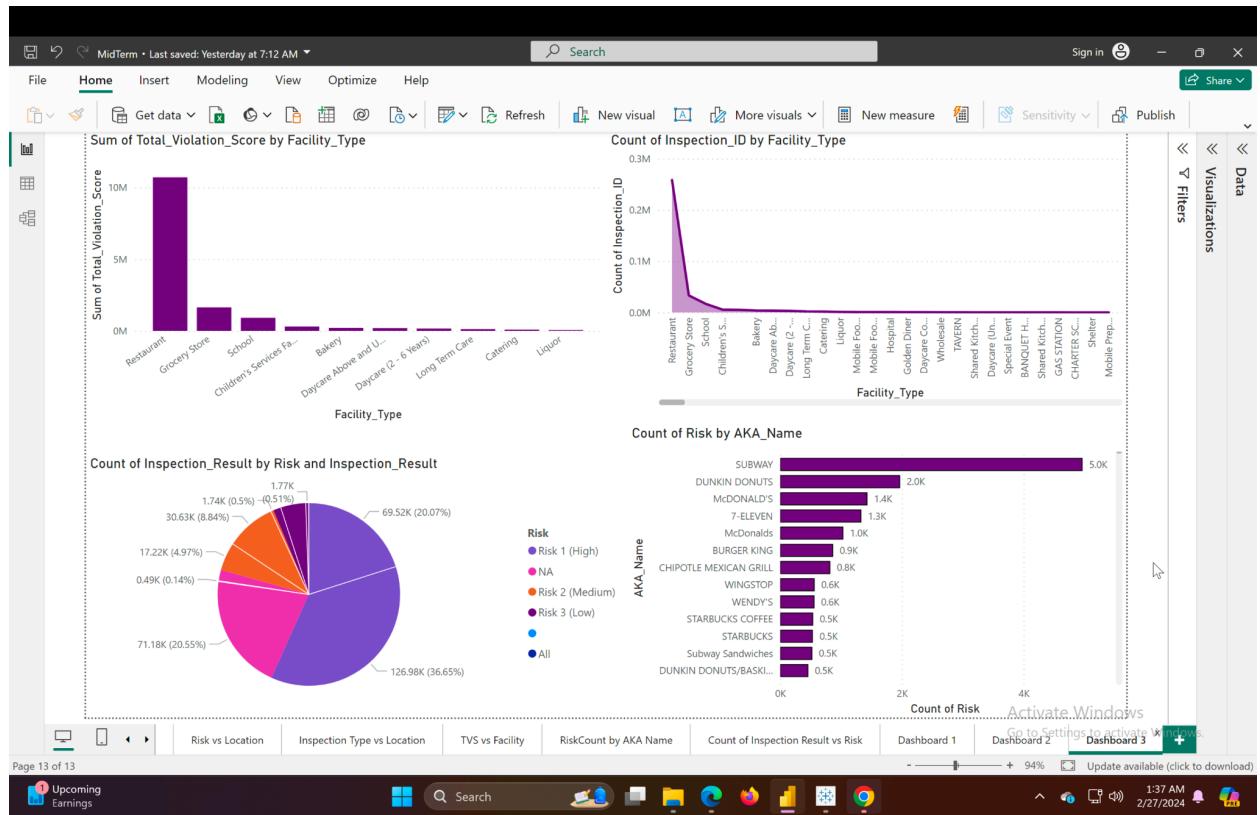
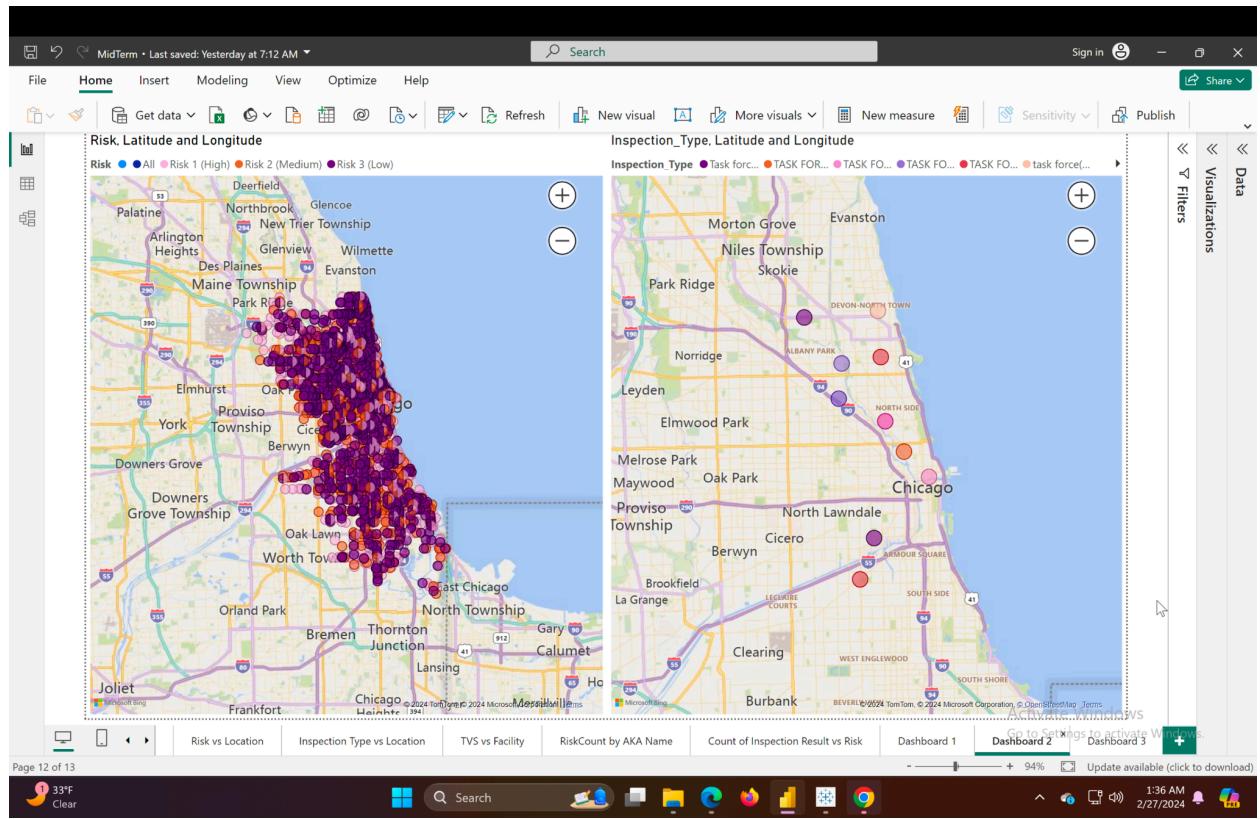
Tableau Dashboards:





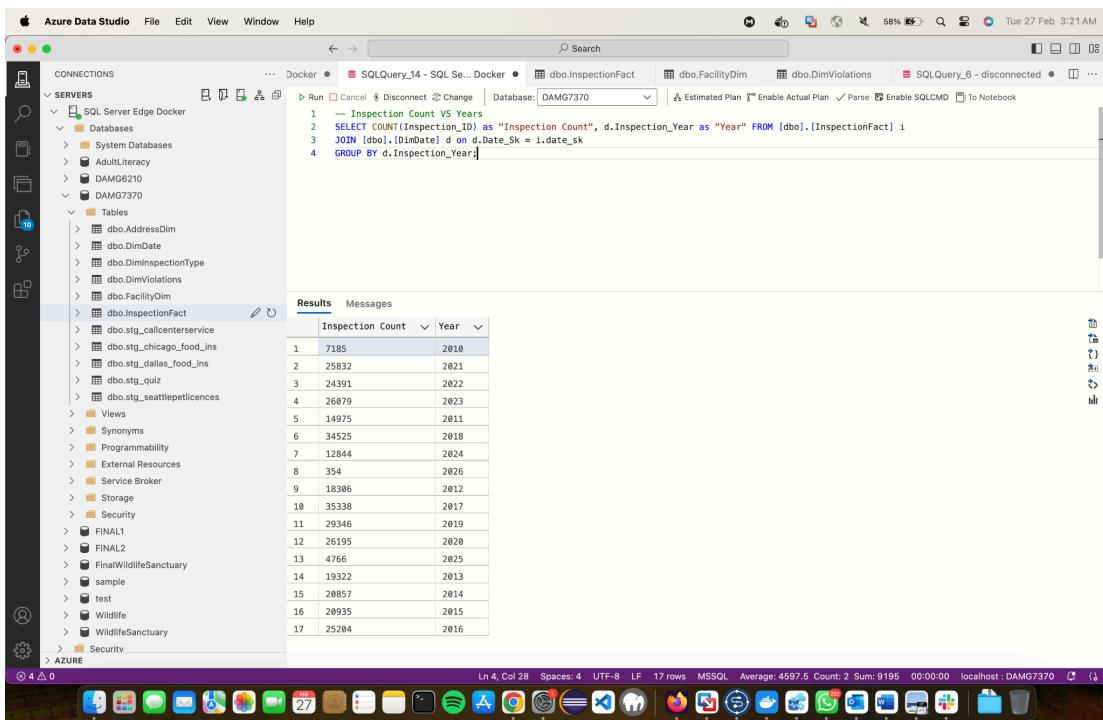
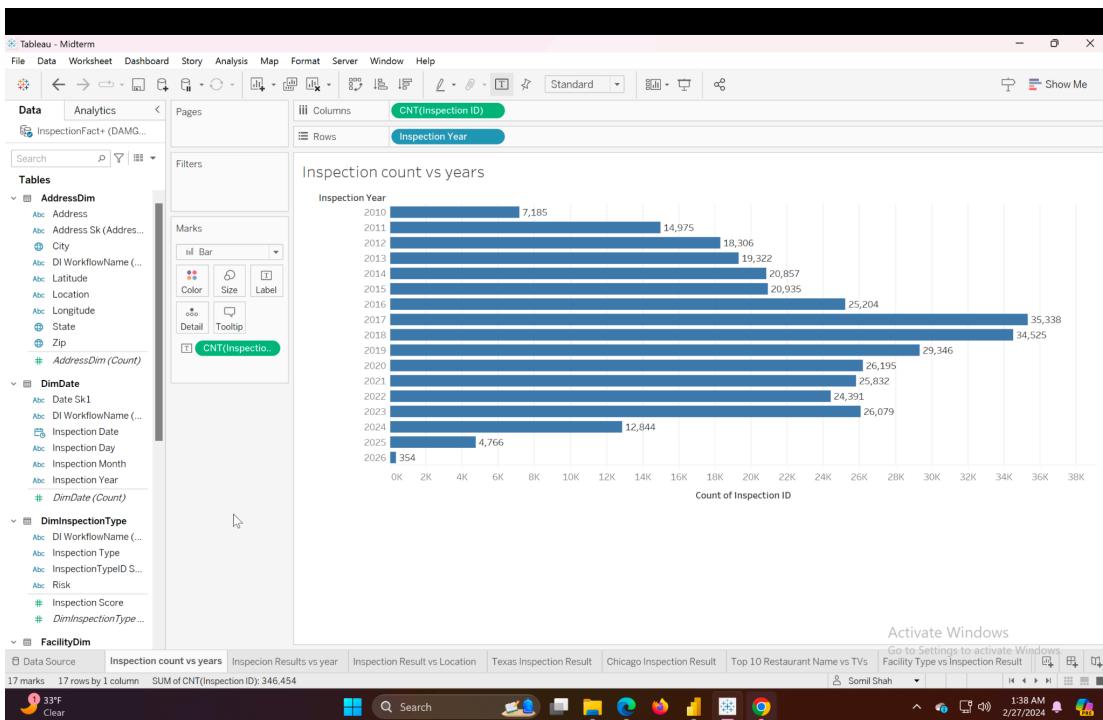
PowerBI Dashboard



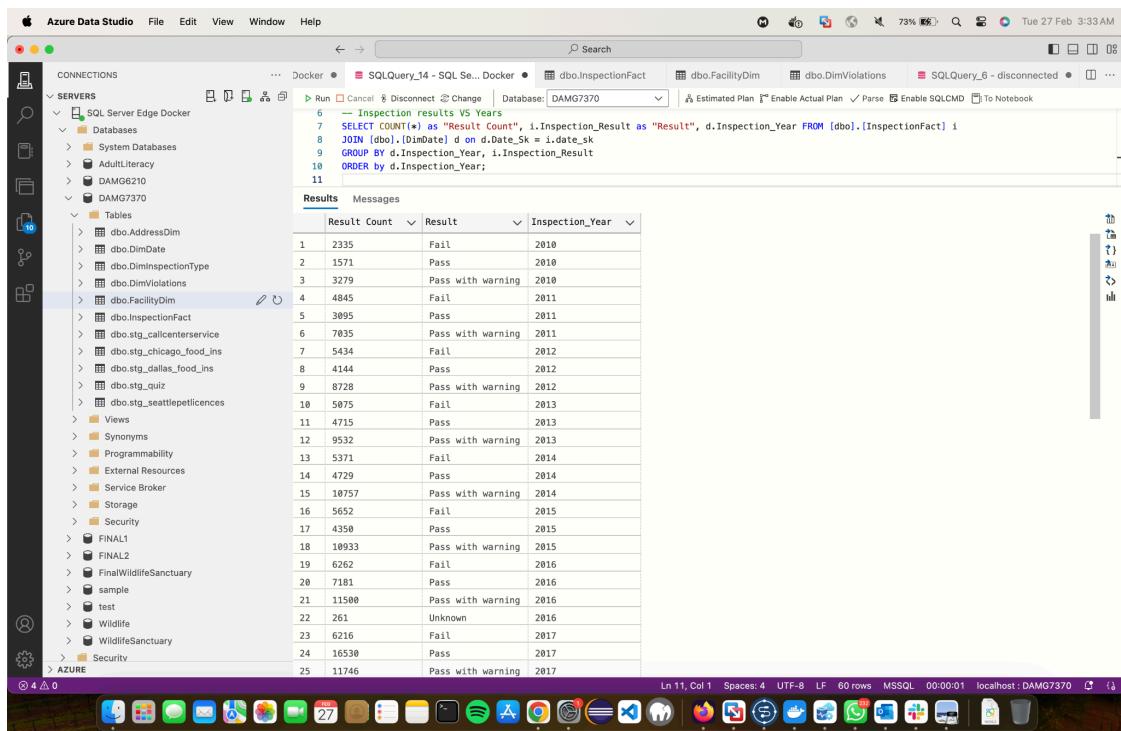
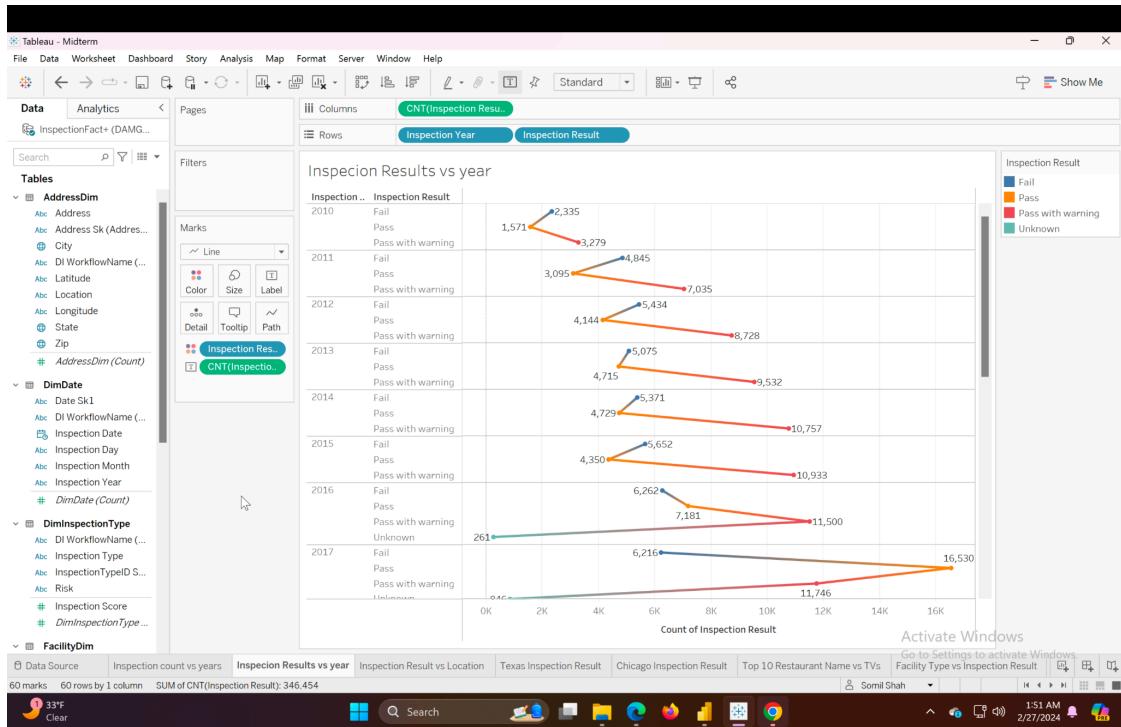


7. Testing

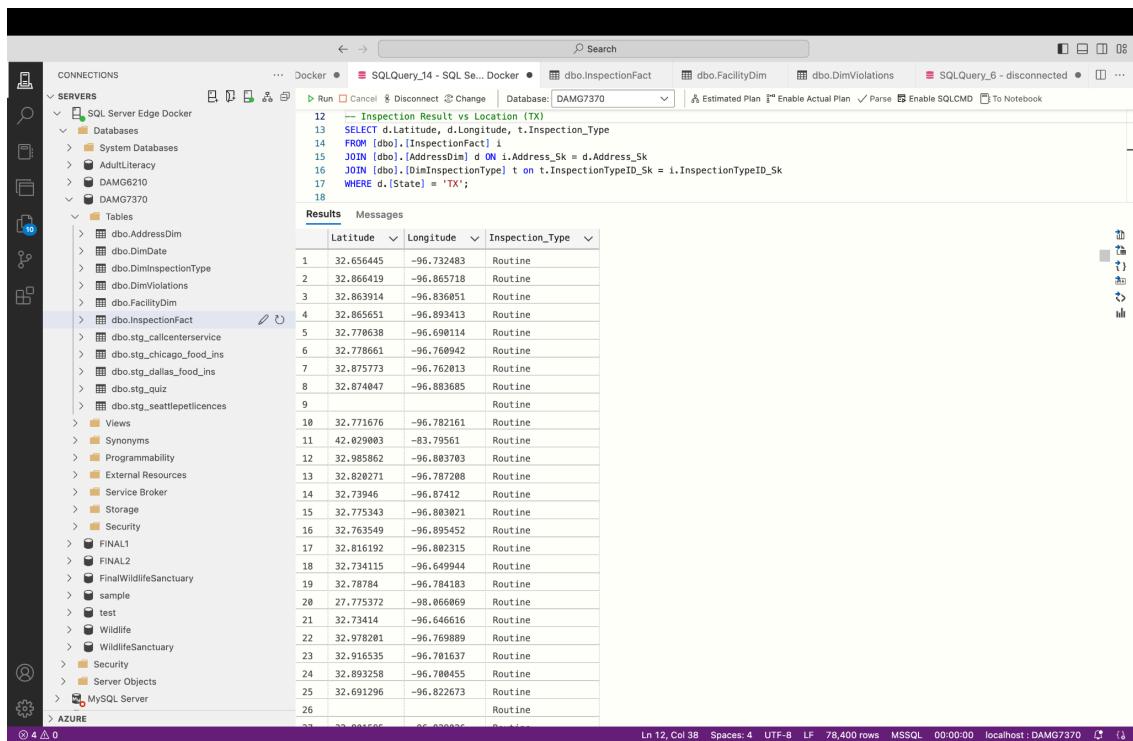
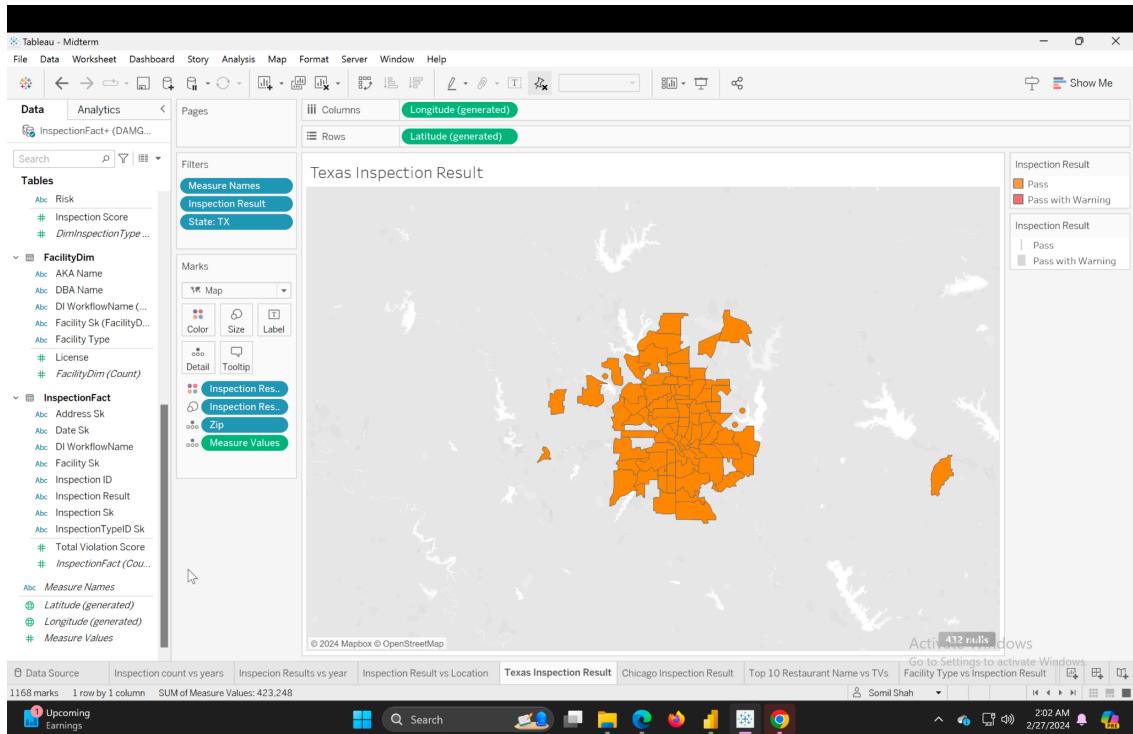
1. Inspection count vs years



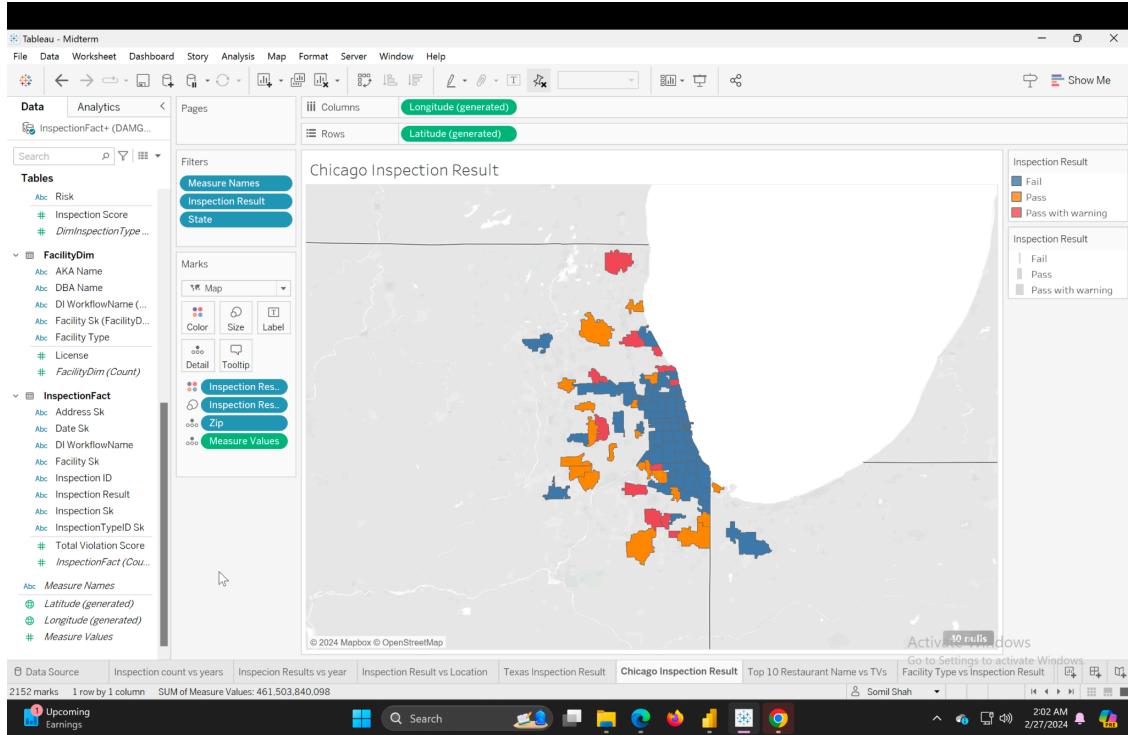
2. Inspection Results vs year



3. Inspection Result VS Location (Dallas)



4. Inspection Result VS Location (Chicago)



The screenshot shows a Microsoft SQL Server Management Studio (SSMS) window. The left sidebar displays the database structure, including servers, databases, tables, and other objects. The main pane shows a T-SQL query being run against the "dbo.InspectionFact" table:

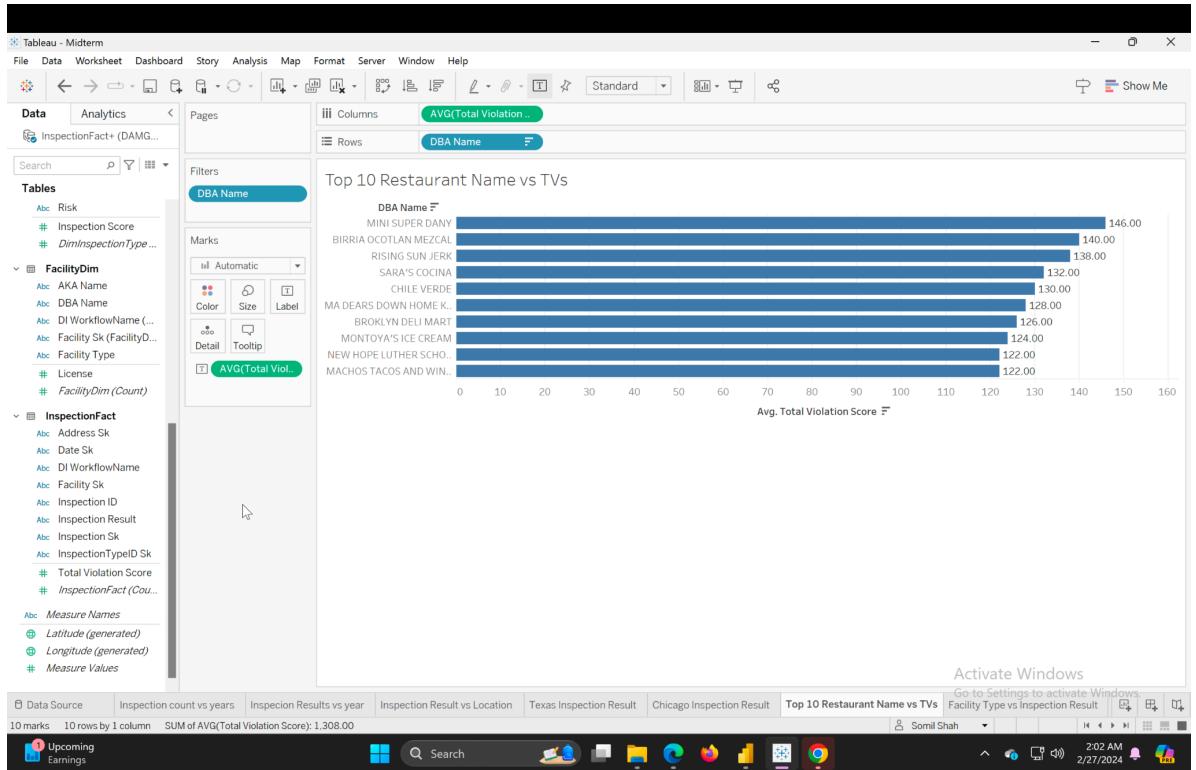
```

19 --> Inspection Result vs Location (IL)
20 SELECT d.Latitude, d.Longitude, t.Inspection_Type
21 FROM [dbo].[AddressDim] d ON i.Address_Sk = d.Address_Sk
22 JOIN [dbo].[DimInspectionType] t ON i.InspectionTypeID_Sk = t.InspectionTypeID_Sk
23 WHERE d.[State] = 'IL';
24
25

```

The results pane displays a table with columns: Latitude, Longitude, and Inspection_Type. The data consists of approximately 25 rows of inspection records for Illinois locations. The inspection types listed include "License Re-Inspection", "Canvass", "License", "Out of Business", "Complaint", and "Complaint Re-Inspection".

5. Top 10 Restaurant Name vs TVs

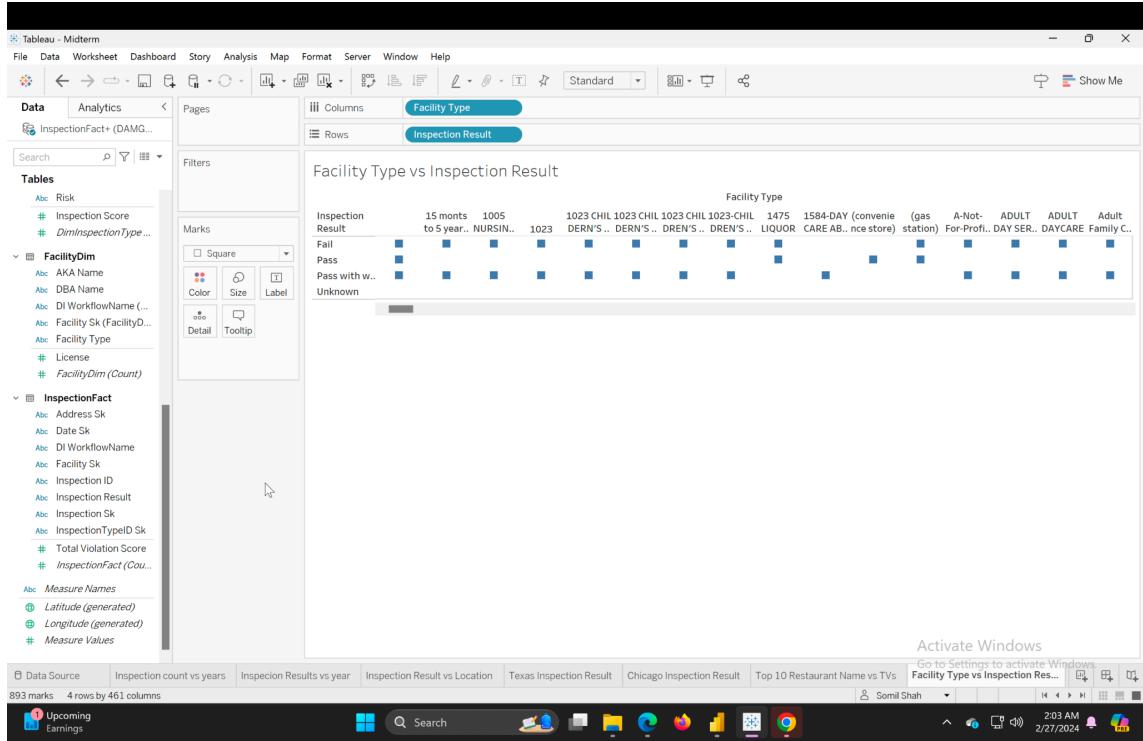


```

SELECT TOP 10 f.DBA_Name, i.Total_Violation_Score
FROM [dbo].[InspectionFact] i
JOIN [dbo].[FacilityDim] f ON f.Facility_Sk = i.Facility_Sk
ORDER BY i.Total_Violation_Score DESC
  
```

DBA_Name	Total_Violation_Score
ABOOD FOOD MARKET INC.	168
KARACHI CHAT HOUSE	168
HOTEL LINCOLN	166
MACIAS PRODUCE INC.	166
JAI YEN	166
SOUL VEG CITY	166
TUMI	166
TAQUERIA EL ARCO #3 EL POLLO FELIZ, INC.	166
NEW STAR GYROS	166
FIRST CHOICE MARKET	166

6. Facility Type vs Inspection Result



The screenshot shows an SSMS session titled "SQLQuery_14 - SQL Server Docker". The left pane shows the "CONNECTIONS" tree with "Docker" selected. The right pane has a "Results" tab displaying a table of inspection results:

	Result Count	Result	Facility_Type
6	12	Fail	1023 CHILDREN'S SERVICES FACILITY
7	1	Fail	1023 CHILDREN'S SERVICES FACILITY
8	7	Fail	1023-CHILDREN'S SERVICES FACILITY
9	1	Fail	1475 LIQUOR
10	2	Fail	15 monts to 5 years old
11	3	Fail	A-Not-For-Profit Chef Training Pr...
12	1	Fail	ADULT DAY SERVICE
13	3	Fail	ADULT DAYCARE
14	1	Fail	Adult Family Care Center
15	7	Fail	AFTER SCHOOL PROGRAM
16	3	Fail	Airport Lounge
17	2	Fail	Animal Shelter Cafe Permit
18	1	Fail	ARCHDIOCESE
19	1	Fail	ART GALLERY
20	1	Fail	ART GALLERY W/WINE AND BEER
21	12	Fail	Assisted Living
22	4	Fail	Assisted Living Senior Care
23	1375	Fail	Bakery
24	1	Fail	BAKERY/DELI
25	1	Fail	BAKERY/GROCERY
26	32	Fail	BANQUET
27	1	Fail	Banquet Dining
28	5	Fail	BANQUET FACILITY
29	66	Fail	BANQUET HALL
30	6	Fail	BANQUET HALL/CATERING
31	2	Fail	BANQUET ROOM

The query in the editor is:

```

SELECT COUNT(*) as "Result Count", i.Inspection_Result as "Result", d.Facility_Type FROM [dbo].[InspectionFact] i
JOIN [dbo].[FacilityDim] d on d.Facility_Sk = i.Facility_Sk
GROUP BY d.Facility_Type, i.Inspection_Result

```