Food Inspections - Data Engineering & Analysis Project

This project implements an end-to-end Business Intelligence solution analyzing food establishment inspections from Chicago and Dallas. The solution includes data profiling, ETL pipeline development, dimensional modeling, and interactive dashboards to improve public health transparency.

Technologies Used

- **ETL/Profiling:** Alteryx Designer
- **Data Processing:** Databricks (Delta Live Tables, PySpark)
- **Data Modeling:** Navicat Data Modeler
- **Visualization:** Tableau
- **Version Control:** Git

Data Sources

1. Chicago Data Source

Source: City of Chicago Data Portal

Dataset: Food Inspections

URL: https://data.cityofchicago.org/

Format: Tab-Separated Values (TSV)

Update Frequency: Updated June 12, 2023

Files:

- Chicago_2021-2022.tsv (33.6 MB)
- Chicago_2022-2023.tsv (36.4 MB)
- Chicago_2023-2024.tsv (39.2 MB)
- Chicago_2024-2025.tsv (22.7 MB)
- Chicago_2025-partyear.tsv (2.8 MB)

Total Records: ~130,000 inspections

Schema Characteristics:

- **Grain:** One row per inspection
- **Violations:** Embedded in single text field with pipe delimiters
- **Scoring:** Results-based (Pass, Fail, Pass w/ Conditions)
- **Need to derive:** Violation scores from results

2. Dallas Data Source

Source: City of Dallas Open Data

Dataset: Food Establishment Inspections

Format: Tab-Separated Values (TSV)

Files:

- Dallas_2021-2022.tsv (43 MB)
- Dallas_2022-2023.tsv (40.7 MB)

- Dallas_2023-2024.tsv (24.8 MB)
- Dallas_2024-2025.tsv (4.2 MB)
- Dallas_2025-partyear.tsv (2 KB)

Total Records: ~47,000 inspections

Schema Characteristics:

- **Grain:** One row per inspection
- **Violations:** Up to 25 violations in separate column sets (100 columns total)
- **Scoring:** Numeric scores (0-100)
- **Structure:** Wide format requiring unpivoting

3. Schema Comparison

Aspect	Chicago	Dallas
Rows per inspection	1 row	1 row
Violation storage	Pipe-delimited text field	25 separate column sets
Violation structure	Unstructured (needs parsing)	Structured (4 fields × 25)
Scoring system	Results (Pass/Fail) → derive scores	Direct scores (0-100)
Address format	Single address field	Separate street components
Coordinates	Separate lat/long fields	Combined "Lat Long Location"
Inspection ID	Provided	Must generate
License number	License #	Must generate

Data Profiling & Analysis (Alteryx)

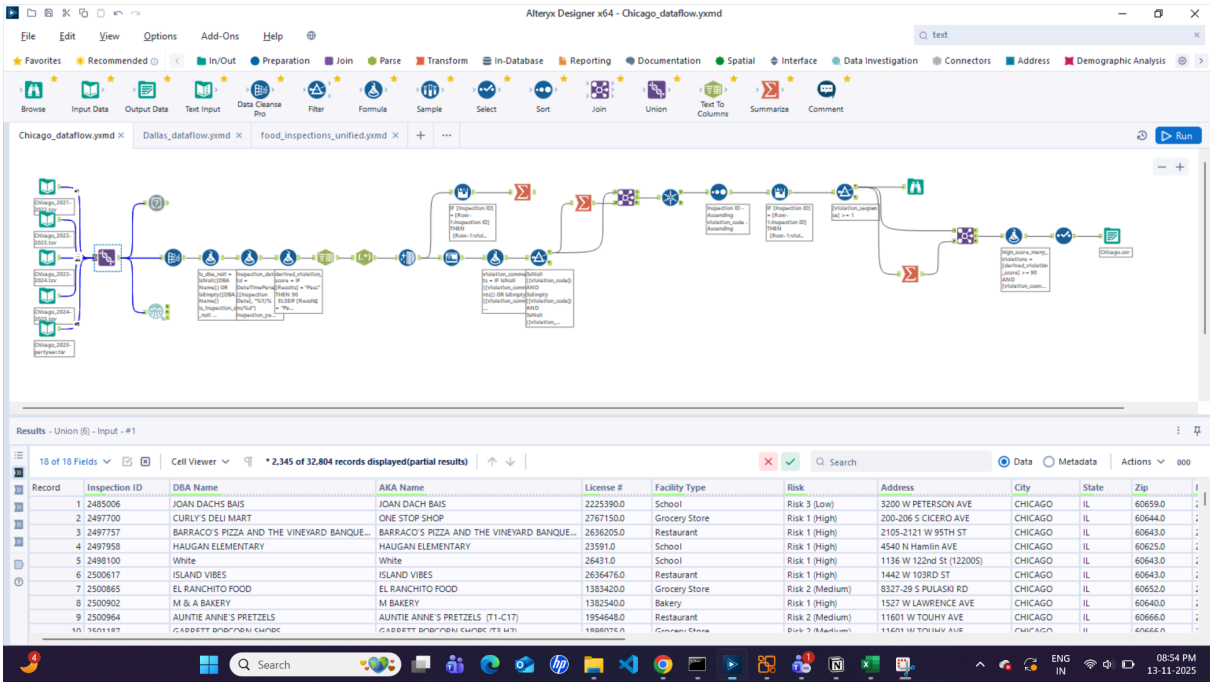
Chicago Data Profiling

Workflow Overview

Purpose: Profile, cleanse, and transform Chicago inspection data from wide unstructured format to long structured format.

Workflow Steps:

1. Load 5 Chicago TSV files (2021-2025)
2. Union all files into single dataset
3. Profile data quality (Field Summary tool)
4. Cleanse text fields (remove whitespace, standardize)
5. Parse violations (split by pipe, extract with RegEx)
6. Derive violation scores from results
7. Remove duplicate violations
8. Add violation sequences and counts
9. Validate business rules
10. Output validated dataset



Results

Field Summary Statistics:

Field	Total Records	Null %	Unique Values	Data Quality
Inspection ID	130,462	0%	73,165	Excellent
DBA Name	130,462	0.14%	15,191	Excellent
License #	130,462	0%	19,415	Excellent
Violations	130,462	30.56%	44,497	Acceptable (not all inspections have violations)
Address	130,462	0%	12,777	Excellent
Zip	130,462	0.01%	88	Excellent
Results	130,462	0%	7	Excellent

Violation Parsing

Challenge: Violations stored as pipe-delimited text:

> "1. NO EMPLOYEE HEALTH POLICY - Comments: PRIORITY | 2. NO HAND WASHING - Comments: URGENT"

Solution: Multi-step parsing process

Step 1: Split by Pipe Delimiter

- Tool: Text to Columns
- Delimiter: |
- Result: Separate row for each violation

Step 2: RegEx Parsing

- Pattern: `(\d+)\.\s*(.+?)\s*-\s*Comments:\s*(.+)`
- Capture Groups:
  - Group 1: Violation code (numeric)
  - Group 2: Violation description
  - Group 3: Comments

- Tool: RegEx tool

Step 3: Clean Extracted Fields

- Remove leading/trailing whitespace
- Handle missing comments
- Tool: Data Cleansing, Formula

Transformations

Derive Violation Scores from Results:

Results	Derived Score	Logic
Pass	90	High compliance
Pass w/ Conditions	80	Minor issues
Fail	70	Significant violations
No Entry	0	Unable to inspect
All other types	NULL	Unknown

Implementation:

```
CASE [Results]
  WHEN "Pass" THEN 90
  WHEN "Pass w/ Conditions" THEN 80
  WHEN "Fail" THEN 70
  WHEN "No Entry" THEN 0
  ELSE NULL
END
```

Deduplication Process

Challenge: Same inspection can have same violation code listed multiple times (data entry errors)

Solution:

1. Sort by Inspection ID + violation_code
2. Use Unique tool on (Inspection ID, violation_code)
3. Result: One distinct violation per inspection

Sequence Numbering

Purpose: Number violations within each inspection (1, 2, 3...)

Tool: Multi-Row Formula

Logic:

```
IF [Inspection ID] = [Row-1:Inspection ID] THEN
  [Row-1:violation_sequence] + 1
ELSE
  1
ENDIF
```

Result:

```
Inspection 2485006:
  Violation code 1 → sequence 1
```

```
Violation code 2 → sequence 2
Violation code 33 → sequence 3
```

Final Output

Output File: Chicago.csv

Record Count: 168,000 violation records

Schema: 37 columns including:

- Inspection identifiers (inspection_id, license_number)
- Restaurant details (dba_name, aka_name, facility_type)
- Location (address, city, state, zip_code, lat, long)
- Inspection details (date, type, result, risk)
- Violation details (code, description, comments, sequence)
- Derived fields (violation_score, source_city, flags)

Dallas Data Profiling

Workflow Overview

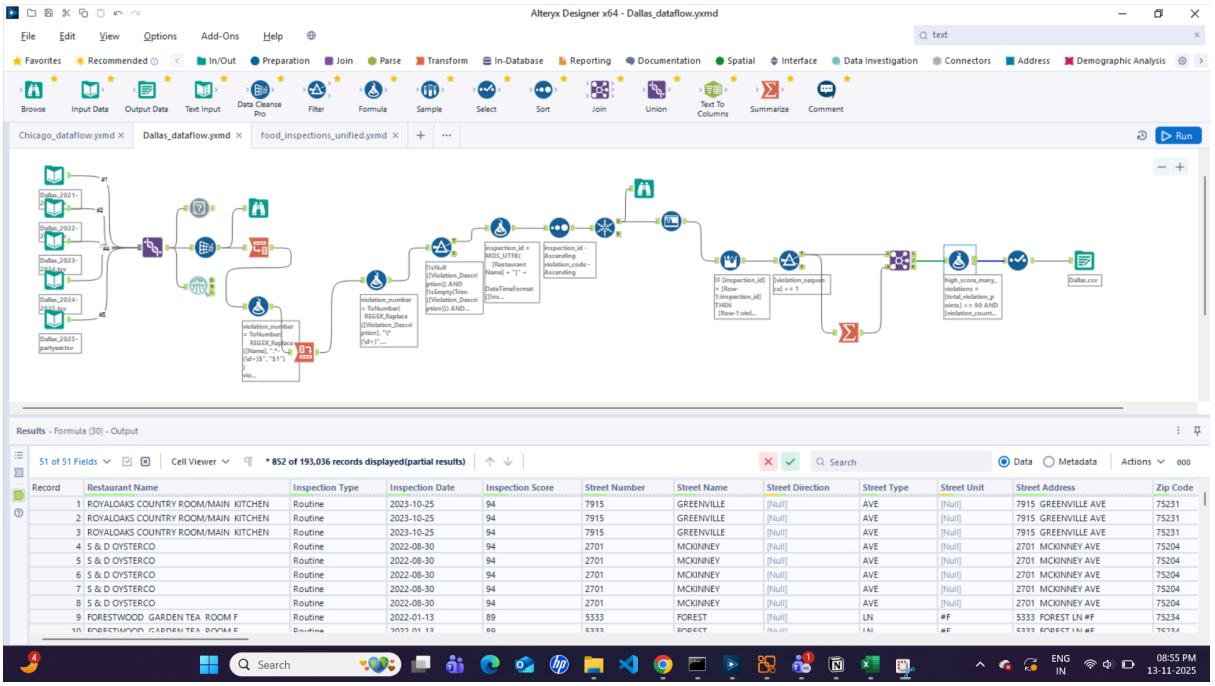
Purpose: Profile, unpivot, and transform Dallas inspection data from wide format (25 violation columns) to long format.

Unique Challenge: Dallas uses **wide format** with 25 violation column sets:

- Violation Description - 1 to 25
- Violation Points - 1 to 25
- Violation Detail - 1 to 25
- Violation Memo - 1 to 25
- **Total: 100 violation-related columns!**

Workflow Steps:

1. Load 5 Dallas TSV files (2021-2025)
2. Union all files (handle quote errors)
3. Profile data quality
4. Cleanse text fields
5. **Transpose:** Unpivot 100 violation columns to rows
6. **Crosstab:** Restructure to 4 violation fields
7. Filter empty violations
8. Parse latitude/longitude from combined field
9. Derive results from scores
10. Output validated dataset



Transpose Operation

Before Transpose (Wide Format):

1 row per inspection with 115 columns:
- Restaurant Name
- Inspection Date
- Violation Description - 1
- Violation Points - 1
- Violation Detail - 1
- Violation Memo - 1
- Violation Description - 2
- ... (continues through 25)

After Transpose (Long Format):

25 rows per inspection with fewer columns:
- Restaurant Name (repeated)
- Inspection Date (repeated)
- Name: "Violation Description - 1"
- Value: "IMPROPER FOOD STORAGE"

Tool: Transpose

- **Key Columns:** Restaurant Name, Inspection Date, scores, location (kept as-is)
- **Data Columns:** All 100 violation fields (transposed to rows)
- **Result:** 47,142 inspections × 25 slots = 1.18 million rows (before filtering)

Crosstab Operation

Purpose: Pivot transposed data back to proper structure with 4 violation columns

Before Crosstab:

Row 1: Name="Violation Description - 1", Value="IMPROPER STORAGE"
Row 2: Name="Violation Points - 1", Value="5"

```
Row 3: Name="Violation Detail - 1", Value="Detail text"
Row 4: Name="Violation Memo - 1", Value="Memo text"
```

After Crosstab:

```
One row with 4 columns:
- Violation Description: "IMPROPER STORAGE"
- Violation Points: "5"
- Violation Detail: "Detail text"
- Violation Memo: "Memo text"
```

Tool: Crosstab

- **Group by:** Restaurant, Date, violation_number
- **Column Headers:** violation_field_type
- **Values:** Value column
- **Result:** Clean 4-column structure

Location Extraction

Challenge: Dallas stores coordinates in format: `"random_text (latitude, longitude)trailing_chars"`

Example:

```
Input: "2456732695hb igfksbfkbfksahpid (32.793366, -96.770156)"
```

Solution: RegEx extraction

Latitude Extraction:

```
Pattern: ^.*\(([\-\d\.]+)\s*,.*$
Result: "32.793366"
```

Longitude Extraction:

```
Pattern: ^[^\(]*\([^\,]*,\s*([\-\d\.]+)[\)\s].*$
Result: "-96.770156"
```

Results Derivation

Dallas has numeric scores (0-100) but needs standardized results

Derivation Logic:

```
IF Inspection Score ≥ 90 THEN "PASS"
ELSEIF Inspection Score ≥ 70 THEN "PASS_CONDITIONS"
ELSEIF Inspection Score < 70 THEN "FAIL"
ELSE "UNKNOWN"
```

Alignment with Chicago:

- Dallas PASS (≥90) = Chicago Pass (90 score)
- Dallas PASS_CONDITIONS (70-89) = Chicago Pass w/ Conditions (80 score)
- Dallas FAIL (<70) = Chicago Fail (70 score)

Final Output

Output File: Dallas.csv

Record Count: 193,000 violation records

Schema: 34 columns including:

- Inspection identifiers (inspection_id - generated, license_number)
- Restaurant details (dba_name, aka_name, facility_type)
- Location (address - combined, city, state, zip_code, lat, long)
- Inspection details (date, type, result, score)
- Violation details (code, description, points, detail, memo)
- Derived fields (source_city, flags, sequences)

Workflow for Unified Dataset

Schema Standardization

Challenge: Different column names and formats between cities

Chicago Column Names:

- `zipcode` → Rename to `zip_code`
- `filename` → Keep or rename to `source_filename`

Dallas Column Names:

- `City` → Rename to `city` (lowercase)
- `State` → Rename to `state` (lowercase)
- `Inspection Score` → Rename to `inspection_score`
- `Violation_Description` → Rename to `violation_description`
- `Lat Long Location` → Parse to `latitude`, `longitude`, `location`

📸 **SCREENSHOT PLACEHOLDER: Unified_Schema_Comparison.png**

Union Process

Tool: Alteryx Union (Auto Config by Name)

Input 1: Chicago_Valid_Inspection_Violations.csv (168K records)

Input 2: Dallas_Valid_Inspection_Violations.csv (193K records)

Output: Unified_Food_Inspections.csv (361K records)

Union Behavior:

- Matches columns by name
- Chicago-only columns: Dallas rows get NULL
- Dallas-only columns: Chicago rows get NULL
- Common columns: Values preserved from both

City-Specific Fields:

Chicago Only:

- `derived_violation_score` (90, 80, 70, 0)
- `results` (original results text)

Dallas Only:

- `inspection_score` (0-100 numeric)
- `violation_points` (points per violation)

📸 **SCREENSHOT PLACEHOLDER: Unified_Union_Tool_Configuration.png**

Location Field Standardization

Formula Implementation:

Extract Latitude (for Dallas rows):

```
IF !IsNull([latitude]) AND !IsEmpty([latitude]) THEN
  Trim([latitude])
ELSEIF !IsNull([lat_long_location]) AND !IsEmpty([lat_long_location]) THEN
  Trim(REGEX_Replace([lat_long_location], "^.*\(([\-\d\.]+)\s*,.*$", "$1"))
ELSE
  NULL
ENDIF
```

Extract Longitude (for Dallas rows):

```
IF !IsNull([longitude]) AND !IsEmpty([longitude]) THEN
  Trim([longitude])
ELSEIF !IsNull([lat_long_location]) AND !IsEmpty([lat_long_location]) THEN
  Trim(REGEX_Replace([lat_long_location], "^[^\(]*\([^\,]*,\s*([\-\d\.]+)[\)\s].*$", "$1"))
ELSE
  NULL
ENDIF
```

Create Standardized Location:

```
IF !IsNull([location]) THEN [location]
ELSEIF !IsEmpty([latitude]) AND !IsEmpty([longitude]) THEN
  "(" + [latitude] + ", " + [longitude] + ")"
ELSE NULL
```

Result: Both cities have consistent latitude, longitude, and location fields

Unified Dataset Summary

Final Output: Unified_Food_Inspections.csv

Total Records: 361K violations

Record Distribution:

- Chicago: 168K violations (46.5%)
- Dallas: 193K violations (53.5%)

Total Columns: 34 standardized columns

Key Achievement: Two completely different schemas unified into one analytical dataset!

Dimensional Model Design

Star Schema Overview

Design Pattern: Star Schema with Bridge Table

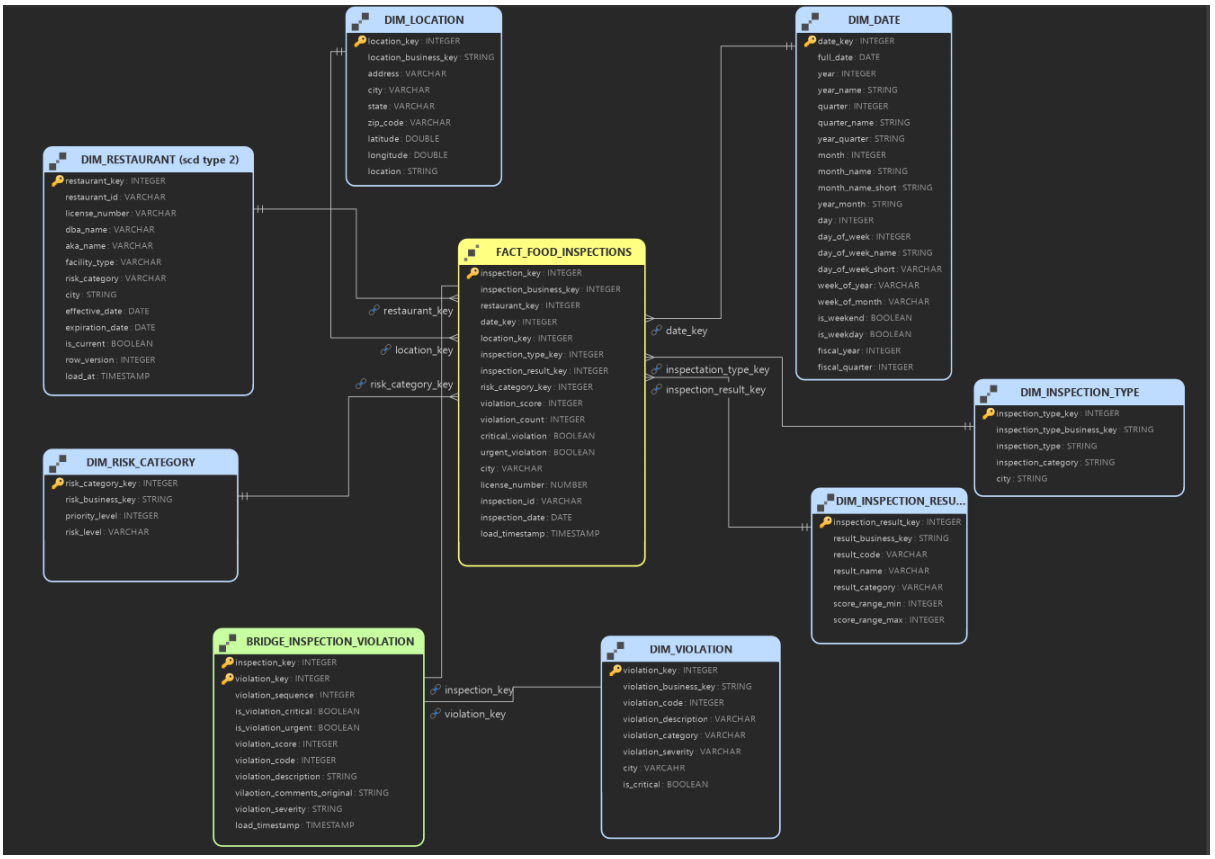
Fact Table: fact_food_inspections (inspection grain)

Dimensions:

1. dim_date (calendar)
2. dim_location (geography)

3. dim_violation (violation codes)
4. dim_inspection_type (inspection types)
5. dim_inspection_result (outcomes)
6. dim_risk_category (risk levels)
7. dim_restaurant (SCD Type 2) ⭐

Bridge Table: bridge_inspection_violation (many-to-many)



Dimensional Model - Table Definitions

1. **fact_food_inspections**

Purpose: Central fact table storing inspection-level metrics

Grain: One row per inspection

Primary Key: inspection_key (BigInt)

Foreign Keys:

- restaurant_key → dim_restaurant
- date_key → dim_date
- location_key → dim_location
- inspection_type_key → dim_inspection_type
- inspection_result_key → dim_inspection_result
- risk_category_key → dim_risk_category

Measures (Facts):

- violation_score (0-100)

- violation_count (total violations)
- critical_violation (count of critical)
- urgent_violation (count of urgent)

Degenerate Dimensions:

- inspection_id (original ID)
- license_number
- city (chicago/dallas)

2. **bridge_inspection_violation**

Purpose: Links inspections to their multiple violations (many-to-many)

Grain: One row per violation instance

Composite Primary Key:

- inspection_key (BigInt, FK to fact)
- violation_key (Int, FK to dim_violation)

Attributes:

- violation_sequence (order within inspection)
- is_violation_critical (boolean for THIS violation)
- is_violation_urgent (boolean for THIS violation)
- violation_score (Dallas points for THIS violation)

Convenience Columns (Redundant):

- violation_code (also in dim_violation)
- violation_description (also in dim_violation)
- violation_comments_original (inspector notes)

3. **dim_date**

Type: Type 1 SCD (static reference)

Grain: One row per calendar date

Date Range: 2018-01-01 to 2028-12-31 (4,018 days)

Primary Key: date_key (Int in YYYYMMDD format)

Attributes:

- Calendar hierarchy (year, quarter, month, day)
- Display names (month_name, day_of_week_name)
- Flags (is_weekend, is_weekday)
- Fiscal calendar (fiscal_year, fiscal_quarter)

Independent Generation: Not derived from inspection data

4. **dim_location**

Type: Type 1 SCD (locations are static)

Grain: One row per unique physical address

Primary Key: location_key (Int)

Business Key: location_business_key (MD5 hash of address components)

Attributes:

- address (full street address)

- city (chicago/dallas)
- state (IL/TX)
- zip_code (5-digit)
- latitude, longitude (DOUBLE for mapping)
- location (combined format)

5. **dim_violation**

Type: Type 1 SCD (violation codes are static)

Grain: One row per unique violation code per city

Primary Key: violation_key (Int)

Business Key: violation_business_key (MD5 of code + city)

Unique Constraint: (violation_code, city)

Critical Design:

- Same violation_code exists for BOTH cities
- Chicago "1" and Dallas "1" are DIFFERENT rows with different keys
- city field distinguishes them

Attributes:

- violation_code (e.g., "1", "2", "33")
- violation_description (text)
- violation_category (derived grouping)
- violation_severity (Critical, Urgent, Serious, etc.)
- is_critical (boolean flag)
- city (chicago/dallas)

6. **dim_inspection_type**

Type: Type 1 SCD (types are static)

Grain: One row per inspection type per city

Primary Key: inspection_type_key (Int)

Attributes:

- inspection_type (original name)
- inspection_category (Routine, Follow-up, Complaint, Re-Inspection, Other)
- city (chicago/dallas)

7. **dim_inspection_result**

Type: Type 1 SCD (results are static)

Grain: One row per result type

Primary Key: inspection_result_key (Int)

Attributes:

- result_code (PASS, FAIL, PASS_CONDITIONS, etc.)
- result_name (display name)
- result_category (Pass, Fail, Other)
- score_range_min (Chicago scoring reference)
- score_range_max (Chicago scoring reference)

8. **dim_risk_category**

Type: Type 1 SCD (risk definitions are static)

Grain: One row per risk level

Primary Key: risk_category_key (Int)

Simplification: 40 source variations → 3 standardized levels

Attributes:

- risk_level (High, Medium, Low)
- priority_level (1, 2, 3)

9. **dim_restaurant (SCD Type 2)** ⭐

Type: Type 2 SCD (tracks historical changes)

Grain: One row per restaurant version

Primary Key: restaurant_key (Int, unique per version)

Business Key: restaurant_id (license_number-city, same across versions)

SCD Type 2 Attributes (Tracked for Changes):

- dba_name (restaurant name)
- facility_type (restaurant, grocery, etc.)
- risk_category (risk level)

SCD Type 2 Metadata Columns:

- effective_date (version start)
- expiration_date (version end, 9999-12-31 if current)
- is_current (TRUE for current version)
- row_version (1, 2, 3...)

```
Version 1 (Historical):
  restaurant_key: 100001
  restaurant_id: 2253911-chicago
  dba_name: "JOE'S PIZZA"
  risk_category: "Risk 1 (High)"
  effective_date: 2021-01-15
  expiration_date: 2023-07-01
  is_current: FALSE
  row_version: 1

Version 2 (Current):
  restaurant_key: 100002
  restaurant_id: 2253911-chicago (SAME)
  dba_name: "JOE'S PIZZA"
  risk_category: "Risk 2 (Medium)" ← CHANGED!
  effective_date: 2023-07-01
  expiration_date: 9999-12-31
  is_current: TRUE
  row_version: 2
```

ETL Pipeline Implementation (Databricks)

Medallion Architecture Overview

Architecture Pattern: Bronze → Silver → Gold

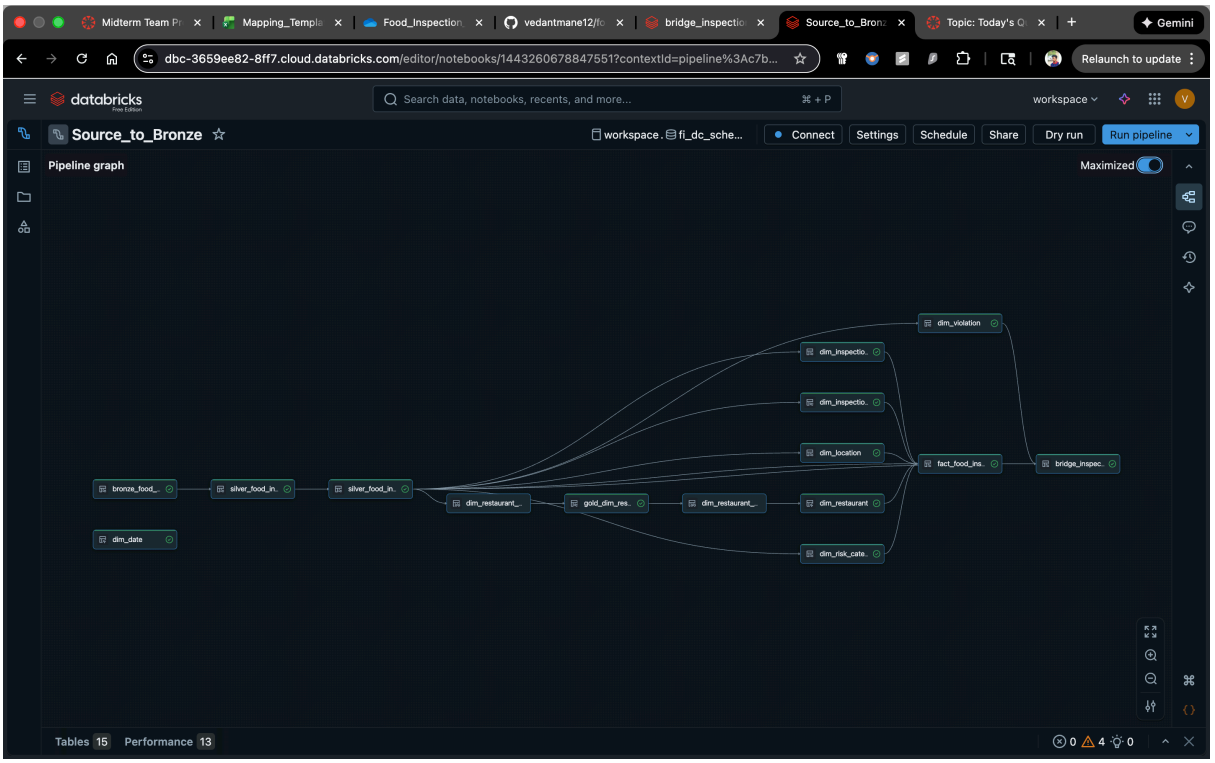
Implementation: Delta Live Tables (DLT) with streaming

Pipeline Name: food_inspections_dlt_pipeline

Catalog: workspace

Schema: fi_dc_schema

Volume: rawdatastore



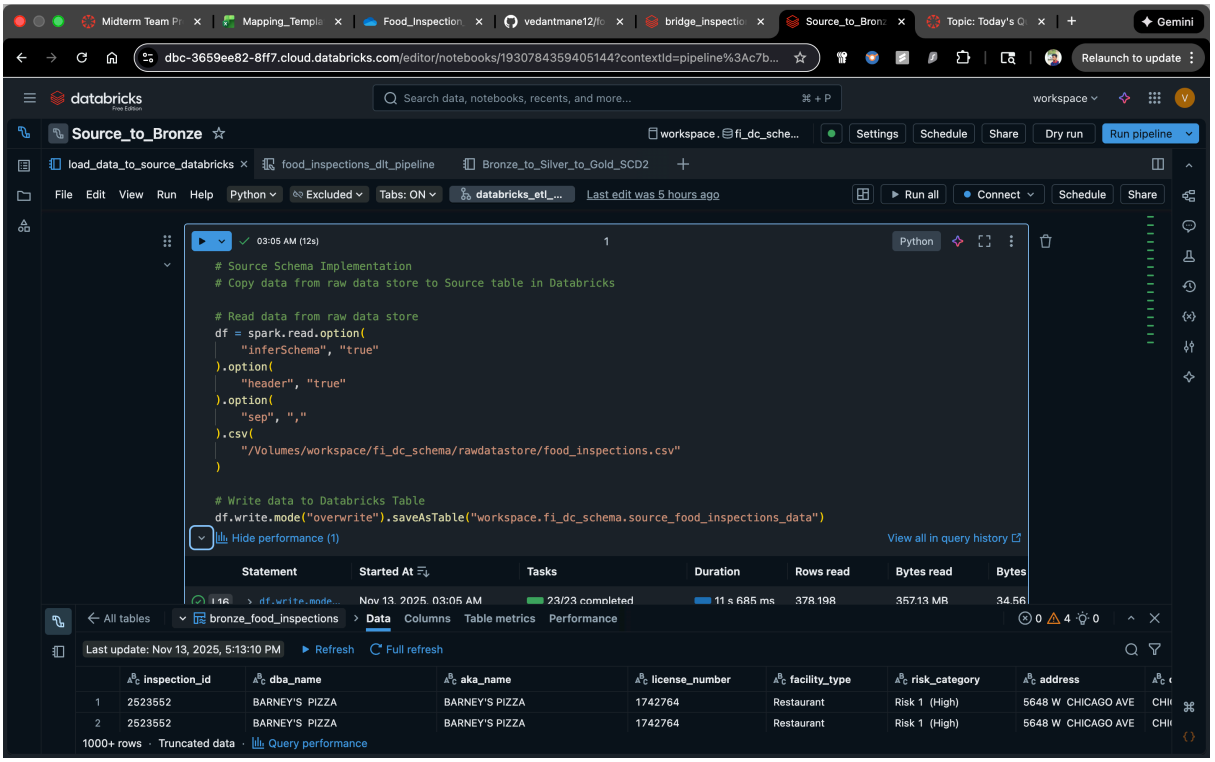
Bronze Zone Implementation

Source Table Setup

Initial Load:

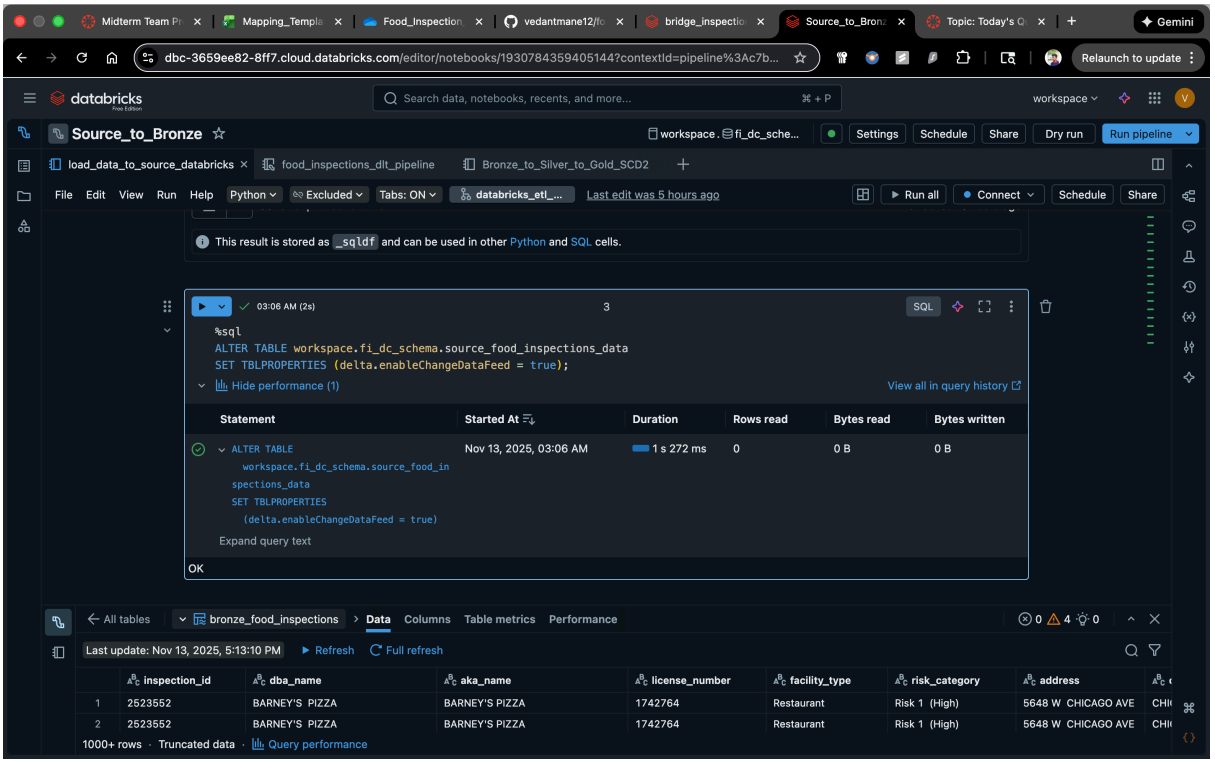
```
df = spark.read.option("inferSchema", "true") \
    .option("header", "true") \
    .option("sep", ",") \
    .csv("/Volumes/workspace/fi_dc_schema/rawdatastore/food_inspections.csv")

df.write.mode("overwrite").saveAsTable("workspace.fi_dc_schema.source_food_inspections_data")
```



Enable Change Data Feed:

```
ALTER TABLE workspace.fi_dc_schema.source_food_inspections_data
SET TBLPROPERTIES (delta.enableChangeDataFeed = true);
```



Bronze Layer Code

Table: bronze_food_inspections

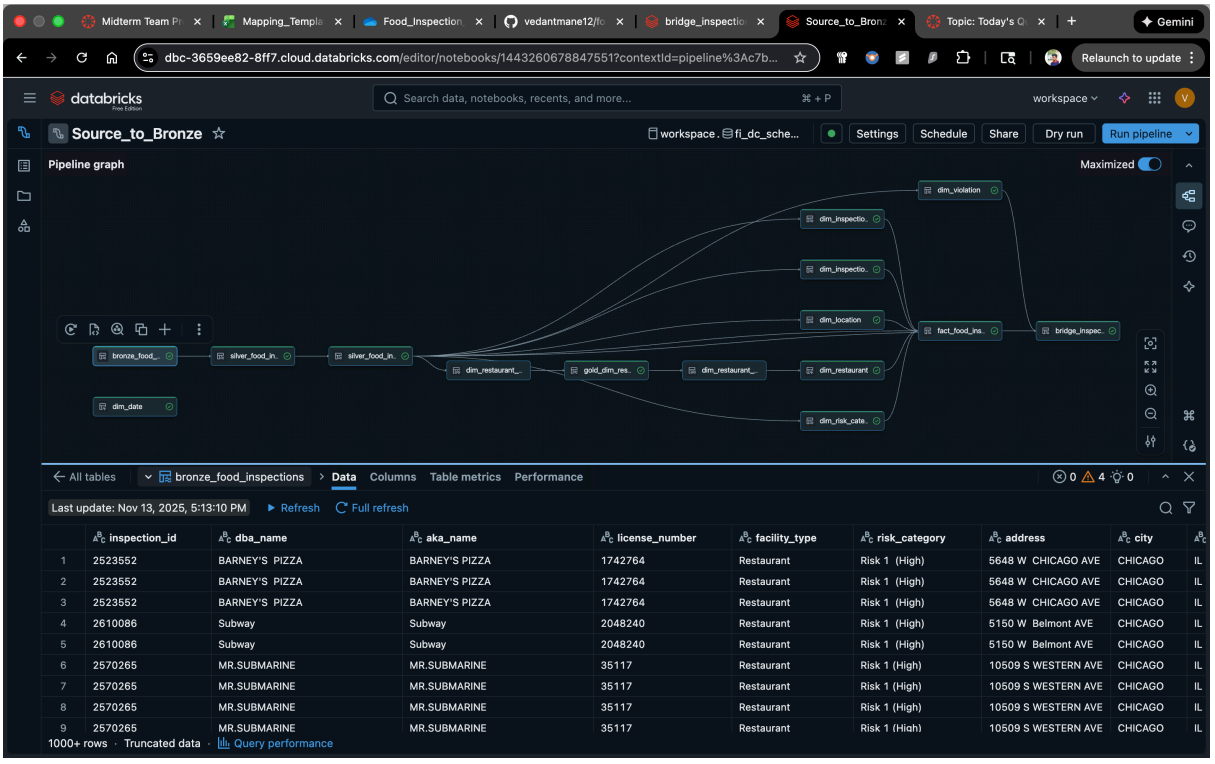
Processing: Streaming with Change Data Feed (CDF)

Purpose: Raw data landing with minimal transformation

Code Implementation:

```
@dlt.table(
  name="bronze_food_inspections",
  comment="Bronze layer - Raw food inspection data from Chicago and Dallas"
)
def bronze_food_inspections():
  return (
    spark.readStream
      .format("delta")
      .option("readChangeFeed", "true")
      .table("workspace.fi_dc_schema.source_food_inspections_data")
      .withColumn("bronze_load_timestamp", current_timestamp())
      .withColumn("bronze_load_date", current_date())
      .withColumn("source_system", lit("Alteryx_ETL"))
  )
```

Output: 361,000 records (all source data + 3 metadata columns)



Silver Zone Implementation

Data Quality Expectations

Framework: DLT Expectations

Expectation Types:

- `@dlt.expect_all`: Logs violations but allows records (WARN)
- `@dlt.expect_or_drop`: Rejects invalid records (DROP)

Implementation:

Validation Rules (WARN):

```
@dlt.expect_all({
    "valid_restaurant_name": "dba_name IS NOT NULL AND LENGTH(dba_name) > 0",
    "valid_inspection_date": "inspection_date IS NOT NULL AND LENGTH(inspection_date) > 0",
    "valid_inspection_type": "inspection_type IS NOT NULL AND LENGTH(inspection_type) > 0",
    "valid_zip_code": "zip_code IS NOT NULL AND (LENGTH(zip_code) = 5 OR LENGTH(zip_code) = 4)"
})
```

Validation Rules (DROP):

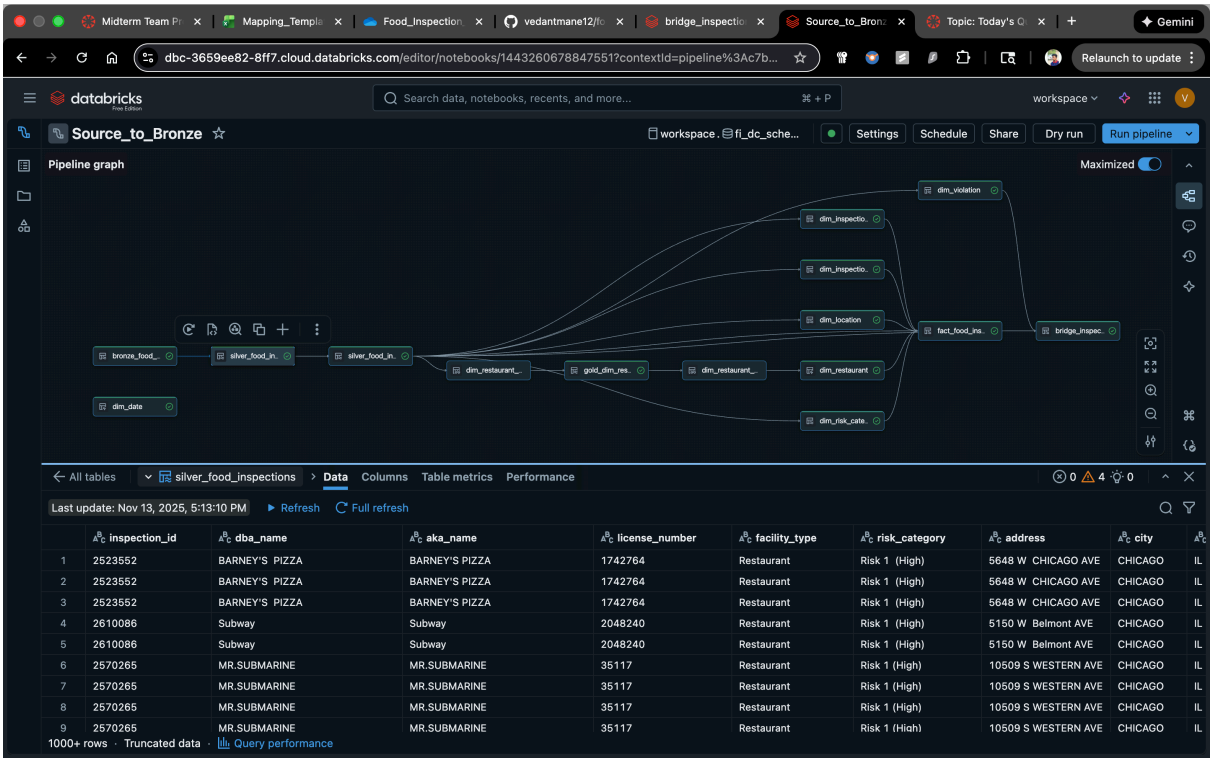
```
@dlt.expect_or_drop("valid_chicago_results",
    "source_city != 'Chicago' OR (source_city = 'Chicago' AND results_std IS NOT NULL)")

@dlt.expect_or_drop("valid_dallas_score",
    "source_city != 'Dallas' OR (source_city = 'Dallas' AND CAST(violation_score AS INT) <= 100)")

@dlt.expect_or_drop("has_at_least_one_violation",
    "CAST(violation_count AS INT) >= 1")

@dlt.expect_or_drop("valid_dallas_high_score_violations",
    "source_city != 'Dallas' OR (CAST(violation_score AS INT) < 90 OR CAST(violation_count AS INT) <= 3)")

@dlt.expect_or_drop("valid_dallas_pass_no_critical_urgent",
    "source_city != 'Dallas' OR (results_std != 'PASS' OR (critical_violation = 'False' AND urgent_violation = 'Fals
e'))")
```



Data Quality Results

Input: 361,000 records from Bronze

Output: 315,000 validated records

Dropped: 46,000 records (13% rejection rate)

Expectation Metrics:

Rule	Action	Failed Records	Fail %	Status
valid_restaurant_name	WARN	0	0%	✅ Excellent
valid_inspection_date	WARN	0	0%	✅ Excellent
valid_inspection_type	WARN	0	0%	✅ Excellent
valid_zip_code	WARN	1,134	0.3%	✅ Acceptable
valid_chicago_results	DROP	17	<0.1%	✅ Excellent
valid_dallas_score	DROP	17	<0.1%	✅ Excellent
has_at_least_one_violation	DROP	146	<0.1%	✅ Excellent
valid_dallas_high_score_violations	DROP	46,594	12.9%	⚠️ High but valid
valid_dallas_pass_no_critical_urgent	DROP	376	0.1%	✅ Good

Silver Transformed Layer

Purpose: Convert string fields to proper data types

Type Conversions:

Numeric (String → Int):

- violation_count, violation_sequence
- inspection_year, inspection_month, inspection_quarter
- violation_score (with NULL handling)

Boolean (String → Boolean):

- is_violation_critical: "True"/"False" → TRUE/FALSE
- is_violation_urgent: "True"/"False" → TRUE/FALSE
- critical_violation: "True"/"False" → TRUE/FALSE
- urgent_violation: "True"/"False" → TRUE/FALSE

Date (String → Date):

- inspection_date: "2021-04-05" → DATE type

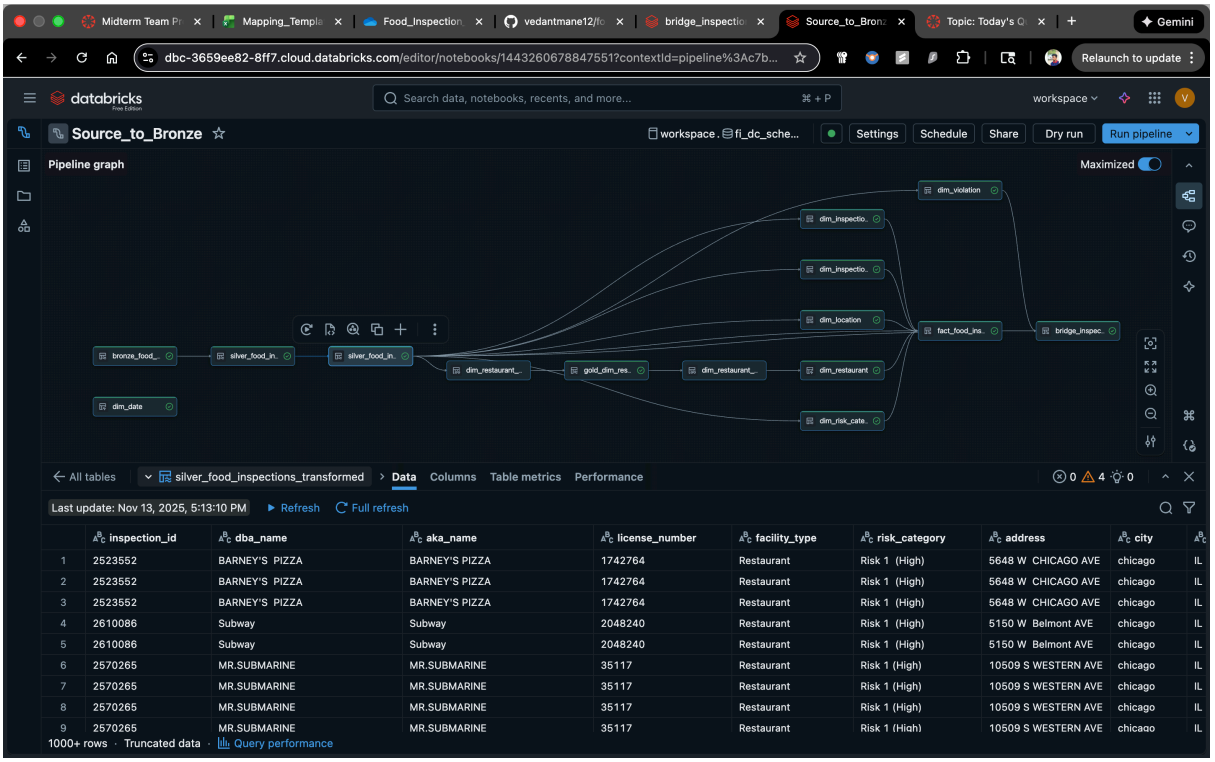
Spatial (String → Double):

- latitude, longitude: String coordinates → DOUBLE precision

String Standardization:

- city: Mixed case → lowercase

Output: 315,000 records with proper types



	inspection_id	dba_name	aka_name	license_number	facility_type	risk_category	address	city	
1	2523552	BARNEY'S PIZZA	BARNEY'S PIZZA	1742764	Restaurant	Risk 1 (High)	5648 W CHICAGO AVE	chicago	IL
2	2523552	BARNEY'S PIZZA	BARNEY'S PIZZA	1742764	Restaurant	Risk 1 (High)	5648 W CHICAGO AVE	chicago	IL
3	2523552	BARNEY'S PIZZA	BARNEY'S PIZZA	1742764	Restaurant	Risk 1 (High)	5648 W CHICAGO AVE	chicago	IL
4	2610086	Subway	Subway	2048240	Restaurant	Risk 1 (High)	5150 W Belmont AVE	chicago	IL
5	2610086	Subway	Subway	2048240	Restaurant	Risk 1 (High)	5150 W Belmont AVE	chicago	IL
6	2570265	MR.SUBMARINE	MR.SUBMARINE	35117	Restaurant	Risk 1 (High)	10509 S WESTERN AVE	chicago	IL
7	2570265	MR.SUBMARINE	MR.SUBMARINE	35117	Restaurant	Risk 1 (High)	10509 S WESTERN AVE	chicago	IL
8	2570265	MR.SUBMARINE	MR.SUBMARINE	35117	Restaurant	Risk 1 (High)	10509 S WESTERN AVE	chicago	IL
9	2570265	MR.SUBMARINE	MR.SUBMARINE	35117	Restaurant	Risk 1 (High)	10509 S WESTERN AVE	chicago	IL

Gold Zone Implementation

1. dim_date

Generation Method: Independent date sequence

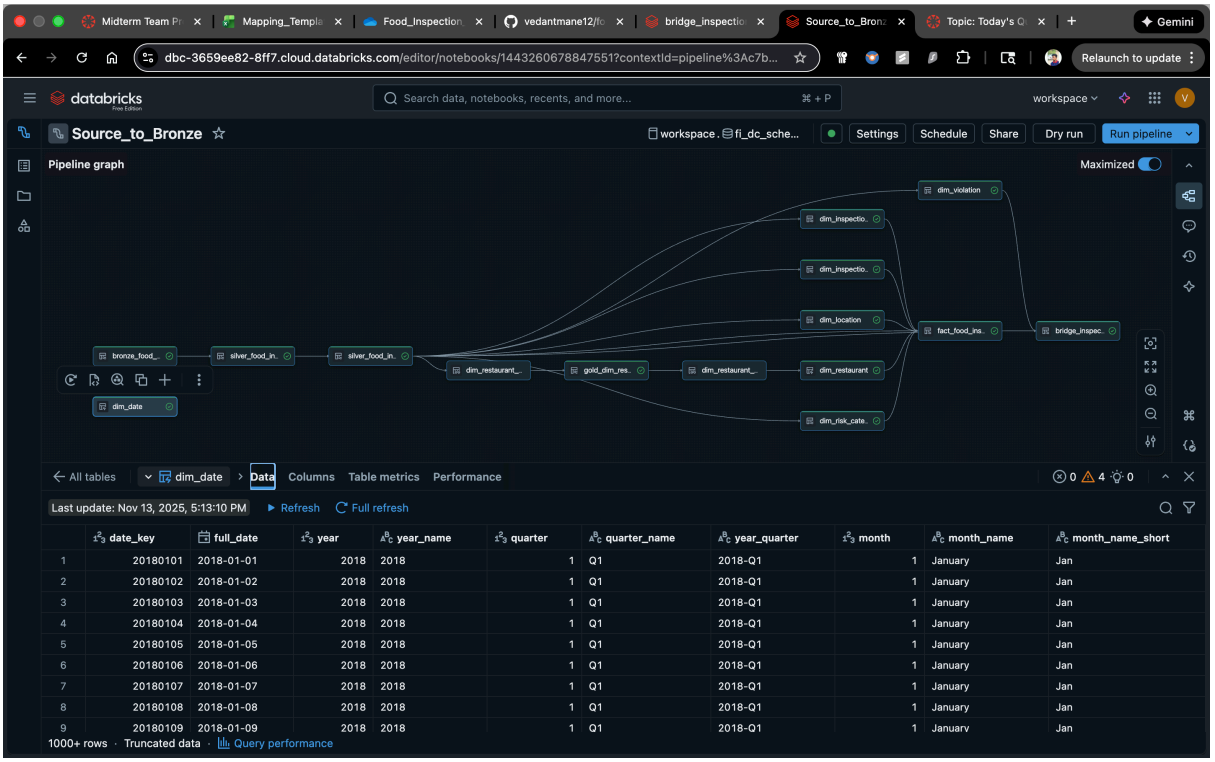
Date Range: 2018-01-01 to 2028-12-31

Generation Code:

```
date_df = spark.sql("""
    SELECT explode(sequence(to_date('2018-01-01'), to_date('2028-12-31'), interval 1 day)) as full_date
""")
```

Attributes Generated: 21 date attributes

Output: 4,018 date records



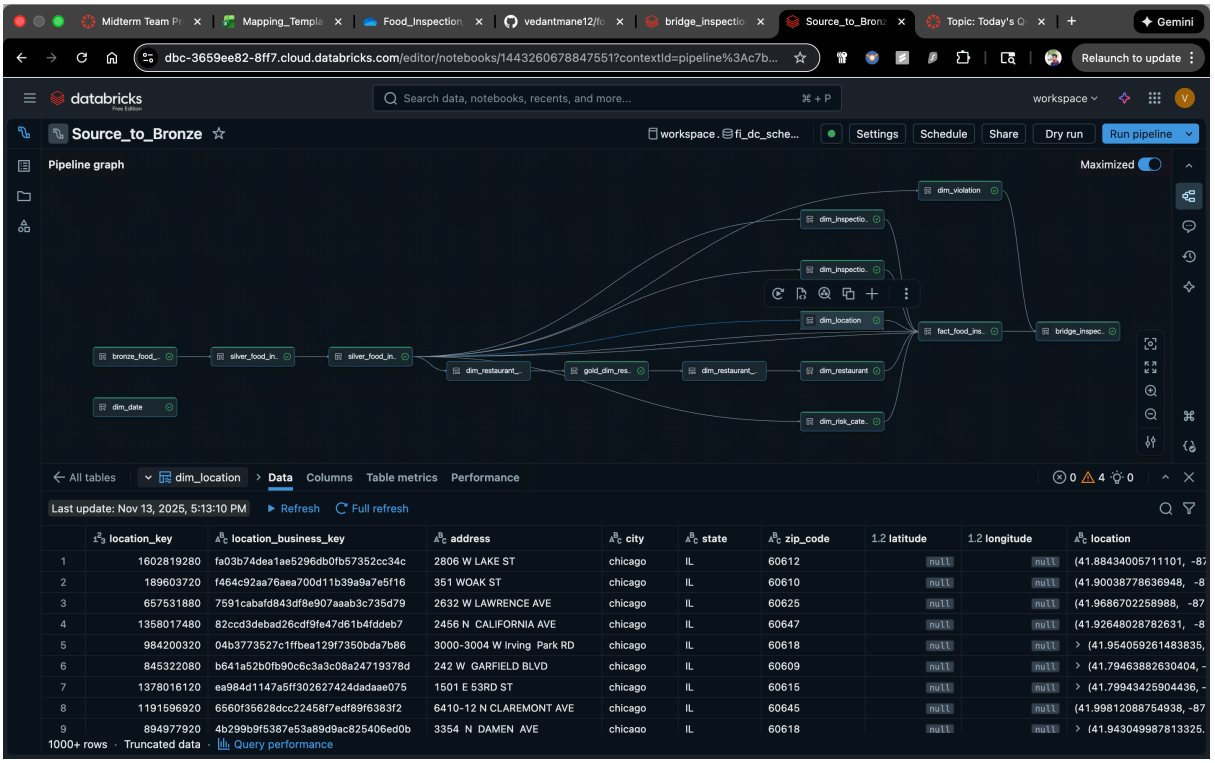
2. dim_location

Processing: Streaming with hash-based deduplication

Business Key: MD5(address | city | state | zip)

Deduplication: dropDuplicates on business key

Output: 39,000 unique locations



3. dim_violation

Processing: Streaming with deduplication

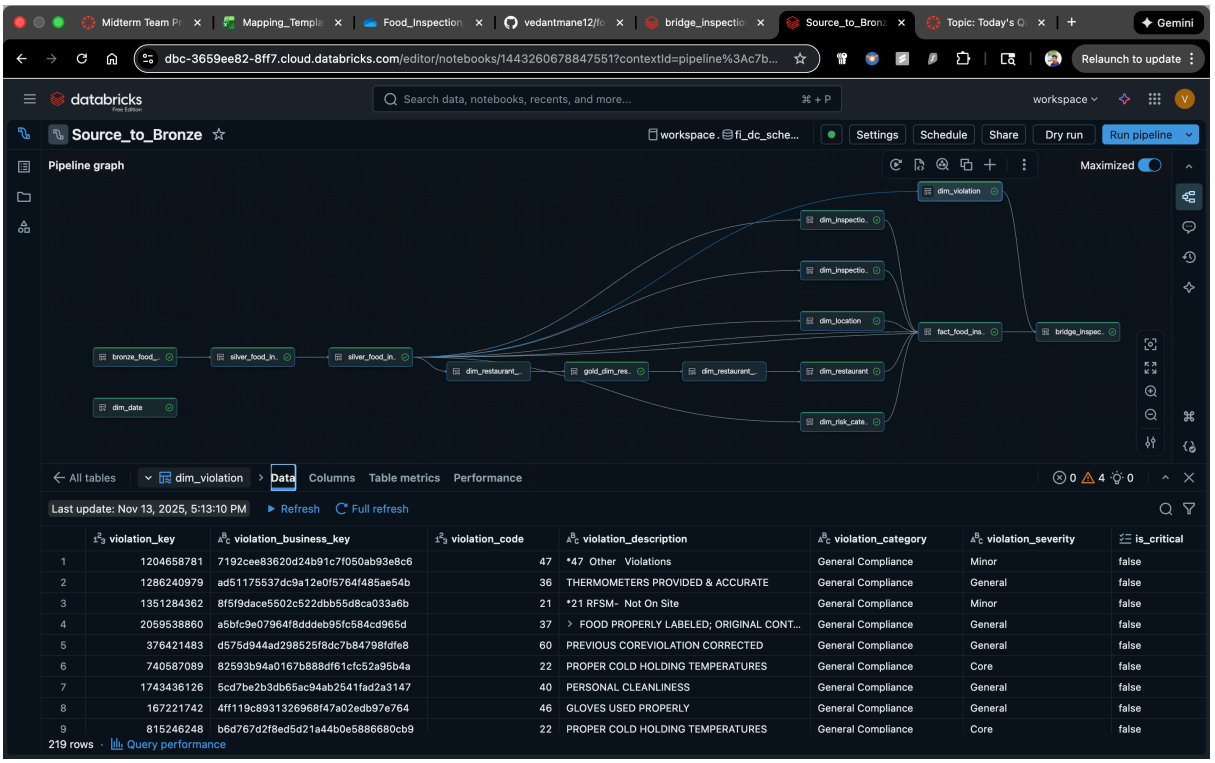
Critical Design: Stores both Chicago and Dallas codes

Business Key: MD5(violation_code | city)

Example:

- Chicago code "1": violation_key = 123456
- Dallas code "1": violation_key = 789012 (different!)

Output: 219 unique violations



	violation_key	violation_business_key	violation_code	violation_description	violation_category	violation_severity	is_critical
1	1204658781	7192cee83620d24b91c7f050ab93e8c6	47	*47 Other Violations	General Compliance	Minor	false
2	1286240979	ad51175537dc9a12e0f5764f485ae54b	36	THERMOMETERS PROVIDED & ACCURATE	General Compliance	General	false
3	1351284362	8f5f9dace5502c522dbb55d8ca033a6b	21	*21 RFSM- Not On Site	General Compliance	Minor	false
4	2059538860	a5bfc9e07964f8dddeb95fc584cd965d	37	› FOOD PROPERLY LABELED; ORIGINAL CONT...	General Compliance	General	false
5	376421483	d575d944ad298525f8dc7b84798fdfe8	60	PREVIOUS COREVIOLATION CORRECTED	General Compliance	General	false
6	740587089	82593b94a0167b888df61cfc52a95b4a	22	PROPER COLD HOLDING TEMPERATURES	General Compliance	Core	false
7	1743436126	5cd7be2b3db65ac94ab2541fad2a3147	40	PERSONAL CLEANLINESS	General Compliance	General	false
8	167221742	4ff119c8931326968f47a02edb97e764	46	GLOVES USED PROPERLY	General Compliance	General	false
9	815246248	b6d767d2f8ed5d21a44b0e5886680cb9	22	PROPER COLD HOLDING TEMPERATURES	General Compliance	Core	false

4. dim_restaurant (SCD Type 2 Implementation)

Method: Databricks `dlt.apply_changes()` with `stored_as_scd_type=2`

Process:

Step 1: Staging View

```
@dlt.view
def dim_restaurant_scd2_stage():
   # Stream restaurant snapshots# Deduplicate on restaurant_id + attributes + date
```

Step 2: Apply Changes

```
dlt.create_streaming_table("gold_dim_restaurant_scd2")

dlt.apply_changes(
   target="gold_dim_restaurant_scd2",
   source="dim_restaurant_scd2_stage",
   keys=["restaurant_id"],
   sequence_by=col("inspection_date"),
   stored_as_scd_type=2
)
```

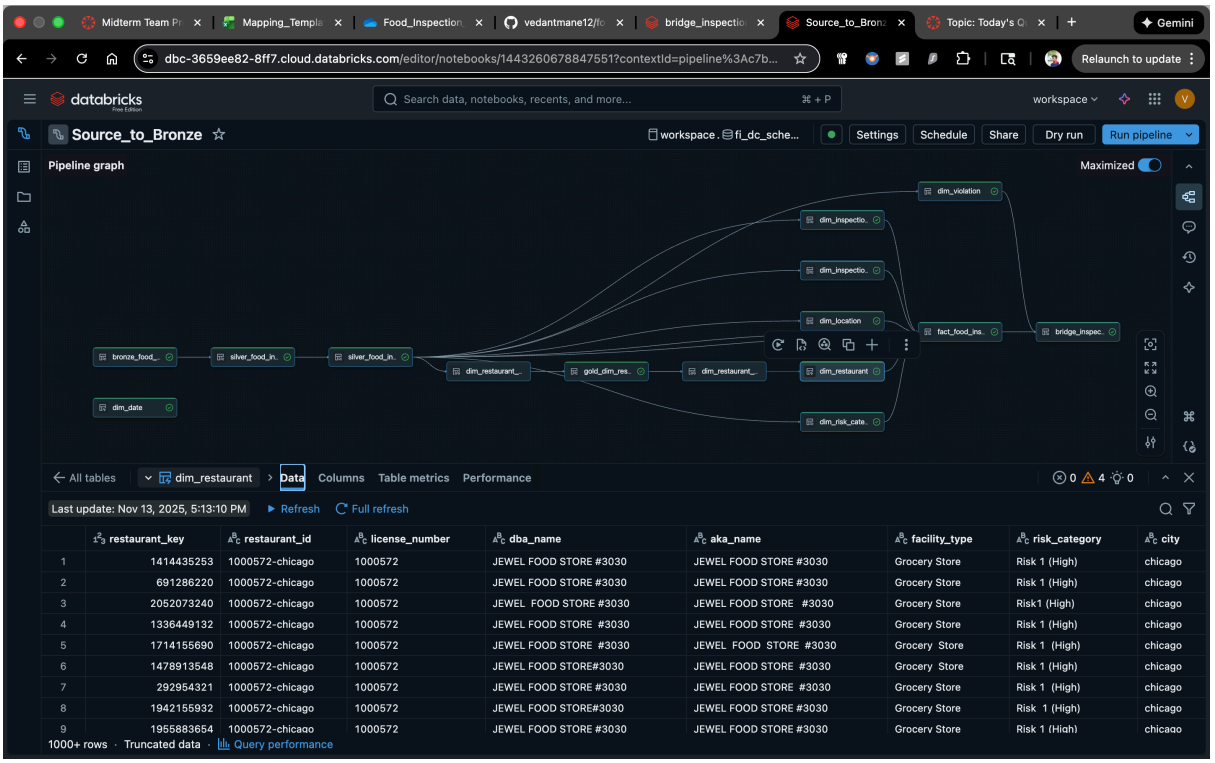
Step 3: Transform View

```
@dlt.view
def dim_restaurant_scd2_transformed():
    # Extract __START_AT → start_dt# Extract __END_AT → end_dt# Create is_active flag
```

Step 4: Final Table

```
@dlt.table(name="dim_restaurant")
def dim_restaurant():
    # Rename to effective_date, expiration_date# Add is_current boolean# Add row_version# Generate restaura
nt_key
```

Output: ~76,000 restaurant versions



restaurant_key	restaurant_id	license_number	dba_name	aka_name	facility_type	risk_category	city
1414435253	1000572-chicago	1000572	JEWEL FOOD STORE #3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago
691286220	1000572-chicago	1000572	JEWEL FOOD STORE #3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago
2052073240	1000572-chicago	1000572	JEWEL FOOD STORE #3030	JEWEL FOOD STORE #3030	Grocery Store	Risk1 (High)	chicago
1336449132	1000572-chicago	1000572	JEWEL FOOD STORE #3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago
1714155690	1000572-chicago	1000572	JEWEL FOOD STORE #3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago
1478913548	1000572-chicago	1000572	JEWEL FOOD STORE#3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago
292954321	1000572-chicago	1000572	JEWEL FOOD STORE #3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago
1942155932	1000572-chicago	1000572	JEWEL FOOD STORE#3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago
1955883654	1000572-chicago	1000572	JEWEL FOOD STORE #3030	JEWEL FOOD STORE #3030	Grocery Store	Risk 1 (High)	chicago

1000+ rows · Truncated data · Query performance

dim_restaurant

restaurant_key	restaurant_id	dba_name	facility_type	effective_date	expiration_date	is_current	row_version
675867641	2205833-chicago	WESLEY PLACE	Long Term Care	July 27, 2021	August 4, 2021	false	1
307343638	2205833-chicago	WESLEY PLACE	Long Term Care	August 4, 2021	September 9, 2022	false	2
1597978295	2205833-chicago	WESLEY PLACE	Long Term Care	September 9, 2022	April 21, 2023	false	3
1487449556	2205833-chicago	WESLEY PLACE	Long Term Care	April 21, 2023	May 24, 2024	false	4
285090300	2205833-chicago	WESLEY PLACE	Long Term Care	May 24, 2024	December 31, 9999	true	5

New page

5. dim_inspection_type

Processing: Streaming with hash-based deduplication

Business Key: `MD5(inspection_type | source_city)`

Deduplication Method: `dropDuplicates(["inspection_type_business_key"])`

Surrogate Key Generation:

```
inspection_type_key = ABS(HASH(inspection_type_business_key)) % 2147483647
```

Process:

```
@dlt.table(name="dim_inspection_type")
def dim_inspection_type():
    # Stream from Silver transformed# Select inspection_type and city# Filter: inspection_type IS NOT NULL# Create business key: MD5(inspection_type | city)# Deduplicate on business_key# Generate surrogate key# Derive inspection_category from type name patterns
```

Attributes:

- inspection_type_key (PK) - Integer surrogate key
- inspection_type_business_key - MD5 hash
- inspection_type - Type name (e.g., "Routine", "Complaint", "License")
- inspection_category - Derived grouping
- source_city - Chicago or Dallas

Category Derivation Logic:

```
inspection_category =
    WHEN LOWER(type) CONTAINS "routine" → "Routine"
    WHEN LOWER(type) CONTAINS "follow" → "Follow-up"
    WHEN LOWER(type) CONTAINS "complaint" → "Complaint"
    WHEN LOWER(type) CONTAINS "re-inspection" → "Re-Inspection"
    ELSE "Other"
```

Output: 102 unique inspection types

Examples:

- Routine
- Canvass
- Complaint
- License
- Canvass Re-Inspection
- Complaint Re-Inspection
- License Re-Inspection
- Short Form Complaint
- Suspected Food Poisoning
- Recent Inspection
- Follow-up

Design Note: Same type name from different cities stored as separate records due to city being part of business key, allowing for city-specific type definitions

---

6. dim_inspection_result

Processing: Streaming with hash-based deduplication

Business Key: `MD5(result_code)`

Deduplication Method: `dropDuplicates(["result_business_key"])`

Surrogate Key Generation:

```
inspection_result_key = ABS(HASH(result_business_key)) % 2147483647
```

Process:

```
@dlt.table(name="dim_inspection_result")
def dim_inspection_result():
    # Stream from Silver transformed# Select results_std (standardized codes)# Filter: results_std IS NOT NUL
L# Create business key: MD5(results_std)# Deduplicate on business_key# Generate surrogate key# Derive re
sult_category (Pass/Fail/Other)# Map score ranges from Chicago scoring system
```

Attributes:

- inspection_result_key (PK) - Integer surrogate key
- result_business_key - MD5 hash
- result_code - Standardized result code
- result_name - Display name (same as code)
- result_category - Grouped category: "Pass", "Fail", "Other"
- score_range_min - Minimum score threshold
- score_range_max - Maximum score threshold

Standardization from Source Systems:

Score Range Mapping:

```
PASS:            score_range_min = 90,  score_range_max = 100
PASS_CONDITIONS:  score_range_min = 70,  score_range_max = 89
FAIL:           score_range_min = 0,   score_range_max = 69
NO_ENTRY:        score_range_min = 0,   score_range_max = 0
Others:          score_range_min = NULL, score_range_max = NULL
```

Output: 6 unique result codes

- PASS
- PASS_CONDITIONS
- FAIL
- NO_ENTRY
- OUT_OF_BUSINESS
- OTHER

Business Rule Validation: This dimension supports the Silver zone expectation that validates Chicago records must have non-null results, and Dallas PASS results cannot have critical/urgent violations.

---

7. dim_risk_category

Processing: Streaming with hash-based deduplication

Business Key: `MD5(risk_level | priority_level)`

Deduplication Method: `dropDuplicates(["risk_business_key"])`

Surrogate Key Generation:

```
risk_category_key = ABS(HASH(risk_business_key)) % 2147483647
```

Process:

```
@dlt.table(name="dim_risk_category")
def dim_risk_category():
    # Stream from Silver transformed# Select risk_category from source# Derive standardized risk_level (High/Medium/Low)# Assign priority_level (1/2/3)# Select only risk_level and priority_level (simplified!)# Create business key: MD5(risk_level | priority)# Deduplicate: distinct on risk_level, priority_level# Generate surrogate key
```

Attributes:

- risk_category_key (PK) - Integer surrogate key
- risk_business_key - MD5 hash
- risk_level - Standardized level: "High", "Medium", "Low"
- priority_level - Numeric priority: 1, 2, 3

Standardization Logic:

```
risk_level =
    WHEN LOWER(risk_category) CONTAINS "high" OR CONTAINS "1" → "High"
    WHEN LOWER(risk_category) CONTAINS "medium" OR CONTAINS "2" → "Medium"
    WHEN LOWER(risk_category) CONTAINS "low" OR CONTAINS "3" → "Low"
    ELSE "Unknown"

priority_level =
    WHEN risk_level = "High" → 1
    WHEN risk_level = "Medium" → 2
    WHEN risk_level = "Low" → 3
    ELSE 99
```

Source Value Consolidation:

From **40+ source variations** to **3 clean categories:**

Chicago Variations → Standardized:

- "Risk 1 (High)" → High, priority 1
- "Risk 2 (Medium)" → Medium, priority 2
- "Risk 3 (Low)" → Low, priority 3

Dallas Variations → Standardized:

- "High Risk" → High, priority 1
- "Medium Risk" → Medium, priority 2
- "Low Risk" → Low, priority 3

Additional Derivations:

- Can also derive from performance: inspection_score <70 → High risk
- Or from violation patterns: many critical violations → High risk

Output: 3 unique risk categories

- High (priority 1) - ~40,000 inspections
- Medium (priority 2) - ~20,000 inspections
- Low (priority 3) - ~15,000 inspections

Design Rationale:

- Simplified from complex source variations (Risk 1 (High), High Risk, etc.) to three clean values

- Makes analysis and visualization cleaner
- Priority level enables proper sorting in reports
- Aligns with industry-standard risk categorization (High/Medium/Low)

Business Rule: Used in Silver zone expectation for Dallas high-score validation (score ≥90 with >3 violations)

8. fact_food_inspections

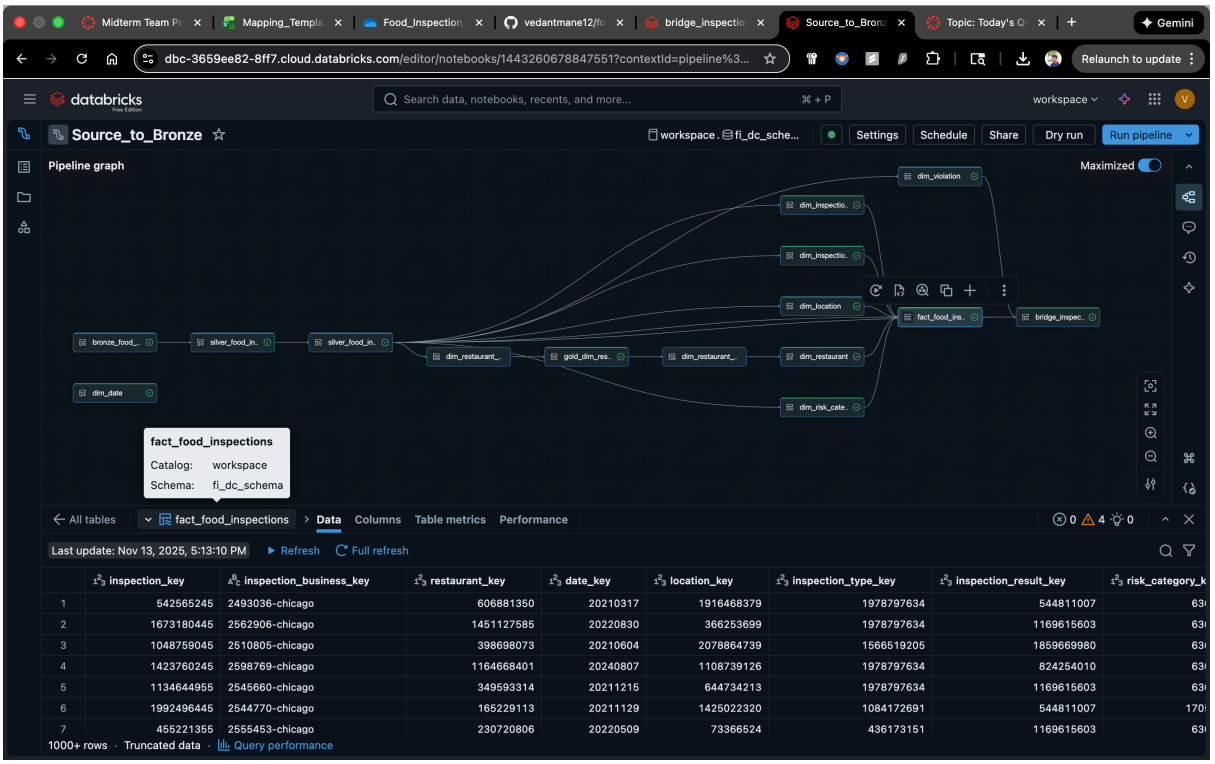
Processing: Streaming with dimension joins

Deduplication: dropDuplicates on inspection_business_key (inspection_id-city)

Dimension Joins:

- Join to dim_restaurant (WHERE is_current=TRUE) → get restaurant_key
- Join to dim_location (on location_business_key) → get location_key
- Join to dim_inspection_type (on inspection_type_business_key) → get inspection_type_key
- Join to dim_inspection_result (on result_business_key) → get inspection_result_key
- Join to dim_risk_category (on risk_business_key) → get risk_category_key
- Create date_key from inspection_date

Output: ~76,000 inspections with foreign keys



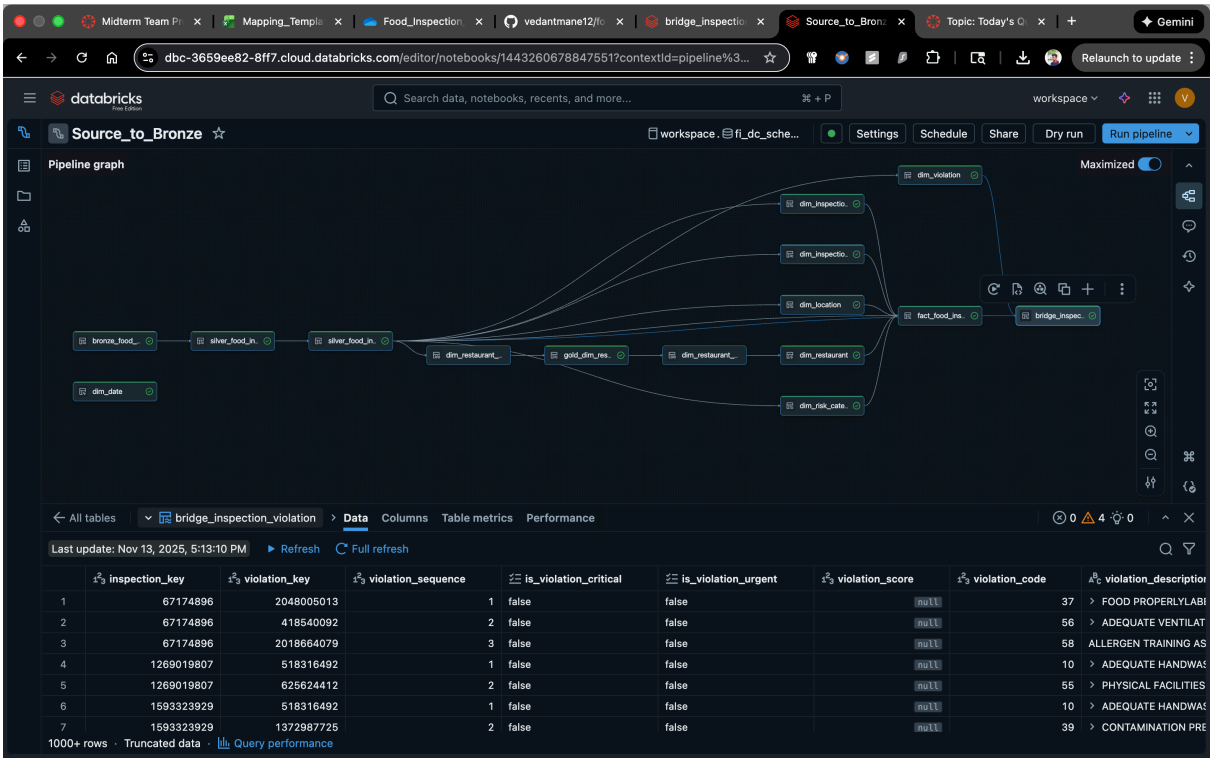
9. bridge_inspection_violation

Processing: Streaming with joins to fact and dimension

Join Process:

1. Create inspection_business_key → Join to fact → get inspection_key
2. Create violation_business_key → Join to dim_violation → get violation_key
3. Keep violation-level details

Output: 315,000 violation instances (full detail from Silver)



	inspection_key	violation_key	violation_sequence	is_violation_critical	is_violation_urgent	violation_score	violation_code	violation_description
1	67174896	2048005013	1	false	false	null	37	FOOD PROPERLYLABI
2	67174896	418540092	2	false	false	null	56	ADEQUATE VENTILAT
3	67174896	2018664079	3	false	false	null	58	ALLERGEN TRAINING AS
4	1269019807	518316492	1	false	false	null	10	ADEQUATE HANDWAS
5	1269019807	625624412	2	false	false	null	55	PHYSICAL FACILITIES
6	1593323929	518316492	1	false	false	null	10	ADEQUATE HANDWAS
7	1593323929	1372987725	2	false	false	null	39	CONTAMINATION PRE

📊 VISUALIZATION

Sheet 1: KPI - Total Inspections

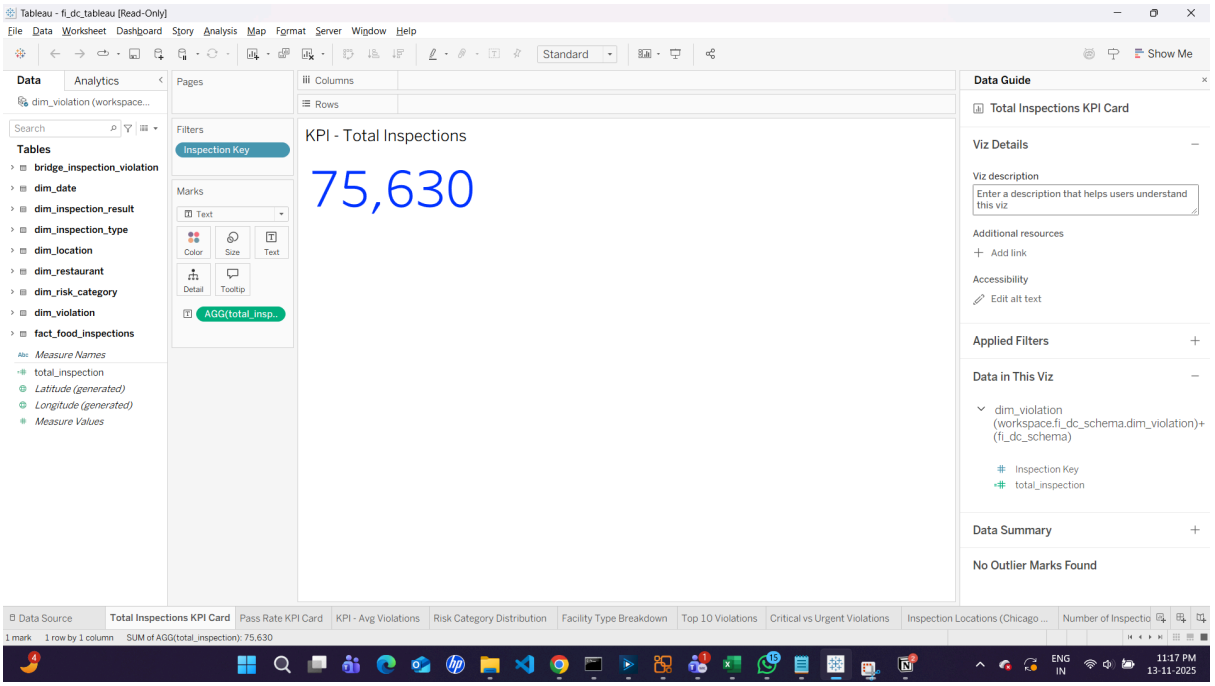
Purpose: Provides an at-a-glance view of the total number of food establishment inspections conducted across both Chicago and Dallas.

Metric Displayed: 75,630 total inspections

Business Value: This KPI serves as the foundational metric for understanding the scale of the food inspection program. It represents the complete dataset after data quality validation rules were applied in the Silver zone, which filtered out 47,000 invalid records from the original 361,000 records.

Key Insight: The dataset contains over 75,000 unique inspections spanning from 2021 to 2025, providing substantial data for meaningful analysis of food safety trends and compliance patterns.

Technical Implementation: Uses COUNTD() on Inspection Key from fact_food_inspections table to ensure each inspection is counted only once, despite having multiple violations.



Sheet 2: Pass Rate KPI Card

Purpose: Displays the overall inspection pass rate broken down by result types (PASS, PASS_CONDITIONS, FAIL, OTHER, NO_ENTRY, OUT_OF_BUSINESS).

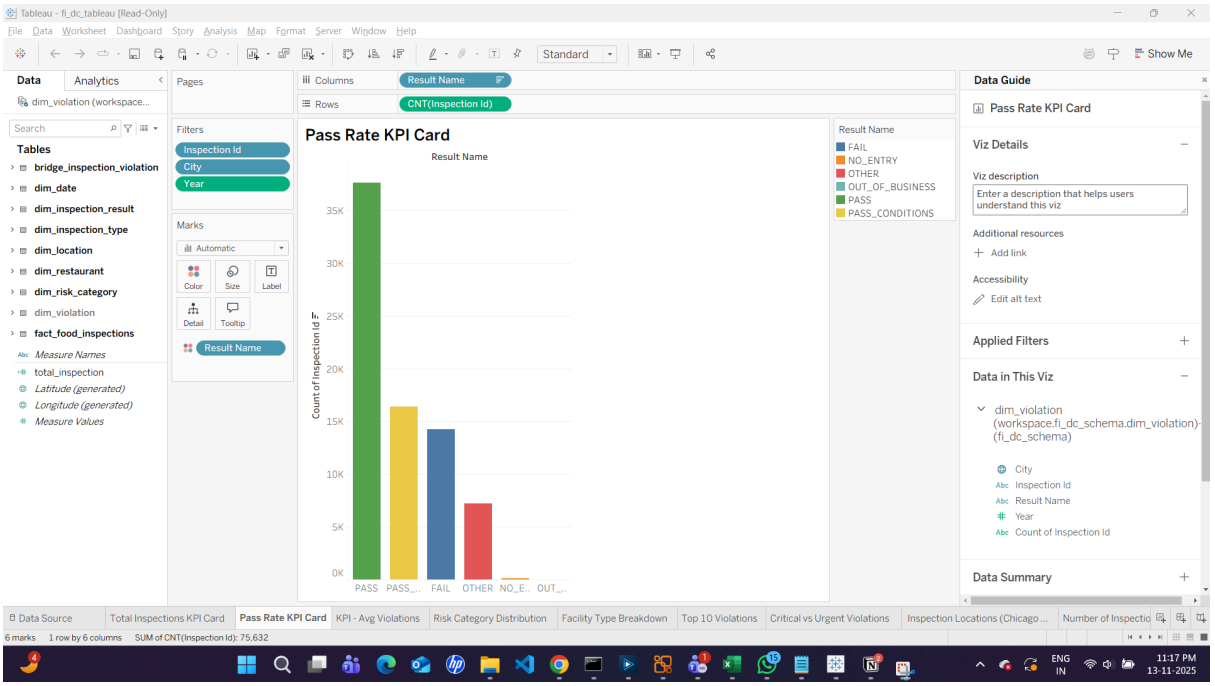
Metric Displayed: Visual breakdown of 75,632 inspections by result category

Business Value: This visualization answers the critical question "How compliant are food establishments?" The stacked bar format allows viewers to see not just the pass/fail split, but also the nuances of conditional passes and other outcomes. The predominance of green (PASS) indicates generally good compliance across both cities.

Key Insights:

- PASS inspections (green) represent the largest category (~35,000 inspections)
- PASS_CONDITIONS (yellow) shows establishments that passed with minor issues (~16,000)
- FAIL inspections (blue/dark) indicate serious compliance issues (~14,500)
- Other categories include administrative outcomes (NO_ENTRY, OUT_OF_BUSINESS)

Technical Implementation: Uses dim_inspection_result table joined to fact_food_inspections to categorize and count inspections by result type. Color coding aligns with industry standards (green=pass, red=fail).



Sheet 3: KPI - Average Violations

Purpose: Shows the distribution of inspections across different inspection types to understand the composition of inspection activities.

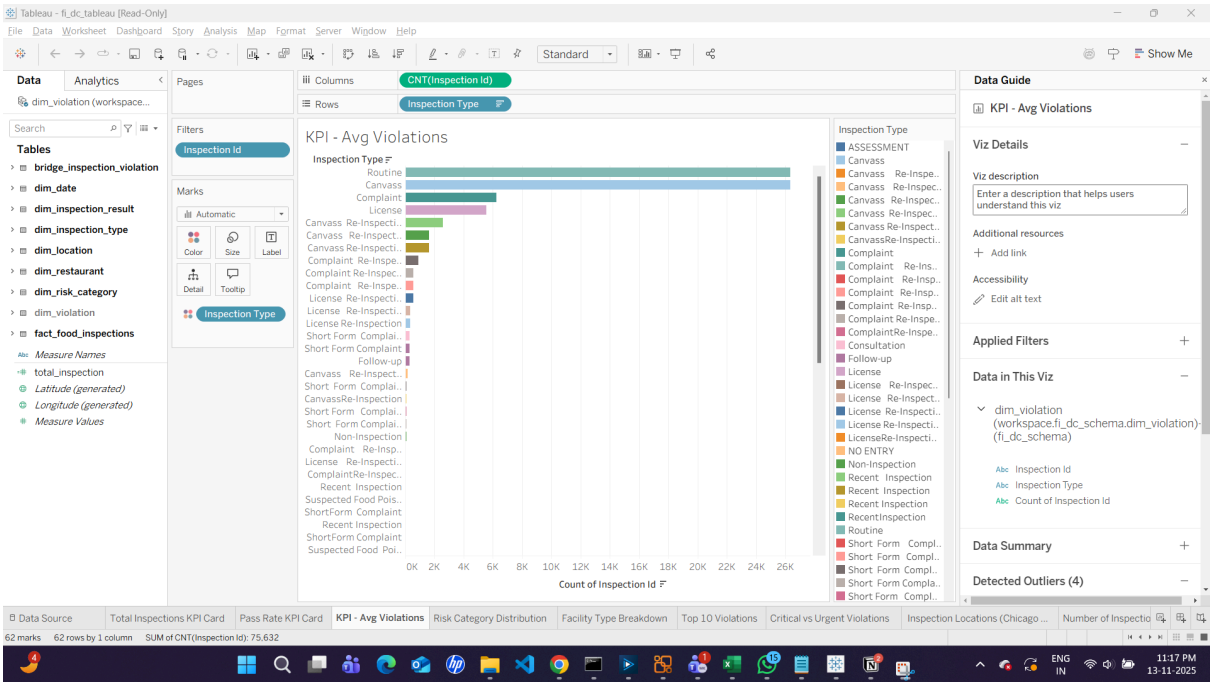
Metric Displayed: Count of inspections by inspection type (Routine, Canvass, Complaint, License, etc.)

Business Value: This visualization reveals the operational focus of food inspection programs. Routine inspections dominate (~25,000), indicating systematic coverage of establishments. Canvass inspections (~20,000) show comprehensive area sweeps. Complaint-driven inspections appear in smaller numbers but represent critical responsive actions.

Key Insights:

- Routine inspections are the primary inspection type (proactive monitoring)
- Multiple types of re-inspections (Canvass Re-Inspection variants) show follow-up efforts
- License-related inspections ensure compliance during renewal periods
- Complaint-based inspections demonstrate responsive public health enforcement

Technical Implementation: Utilizes dim_inspection_type table with Top 15 filter to show most common inspection types. The horizontal bar format allows easy comparison and ranking.



Sheet 4: Risk Category Distribution

Purpose: Displays the distribution of inspections across establishment risk levels (High, Medium, Low).

Metric Displayed:

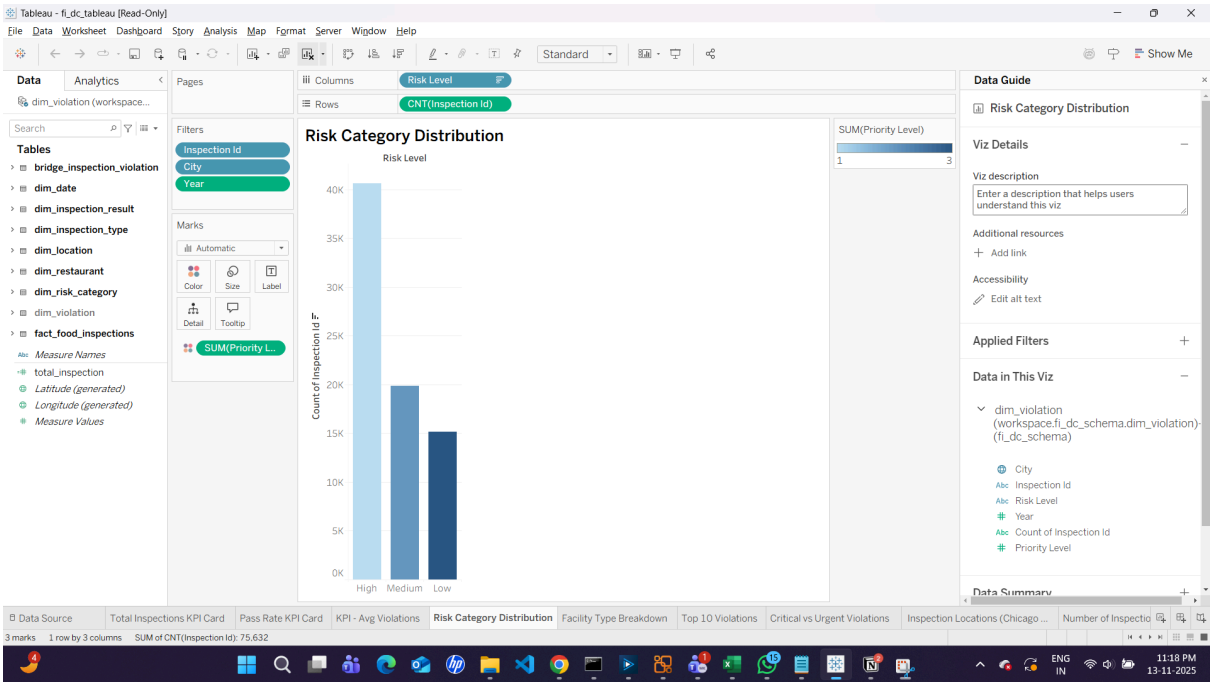
- High Risk: ~40,000 inspections
- Medium Risk: ~20,000 inspections
- Low Risk: ~15,000 inspections

Business Value: This visualization shows how the food inspection program prioritizes resources. The large proportion of high-risk establishment inspections (53%) demonstrates appropriate focus on establishments that pose the greatest public health risk due to food handling practices.

Key Insights:

- High-risk establishments (those handling raw foods, extensive preparation) receive the most inspections
- The risk-based inspection approach aligns with public health best practices
- Medium and low-risk facilities still receive regular monitoring to ensure baseline compliance

Technical Implementation: Uses dim_risk_category dimension table with standardized risk levels (High/Medium/Low derived from Risk 1/2/3 and risk category text). Simple bar chart format provides clear comparison.



Sheet 5: Facility Type Breakdown

Purpose: Analyzes inspection volume across different types of food establishments.

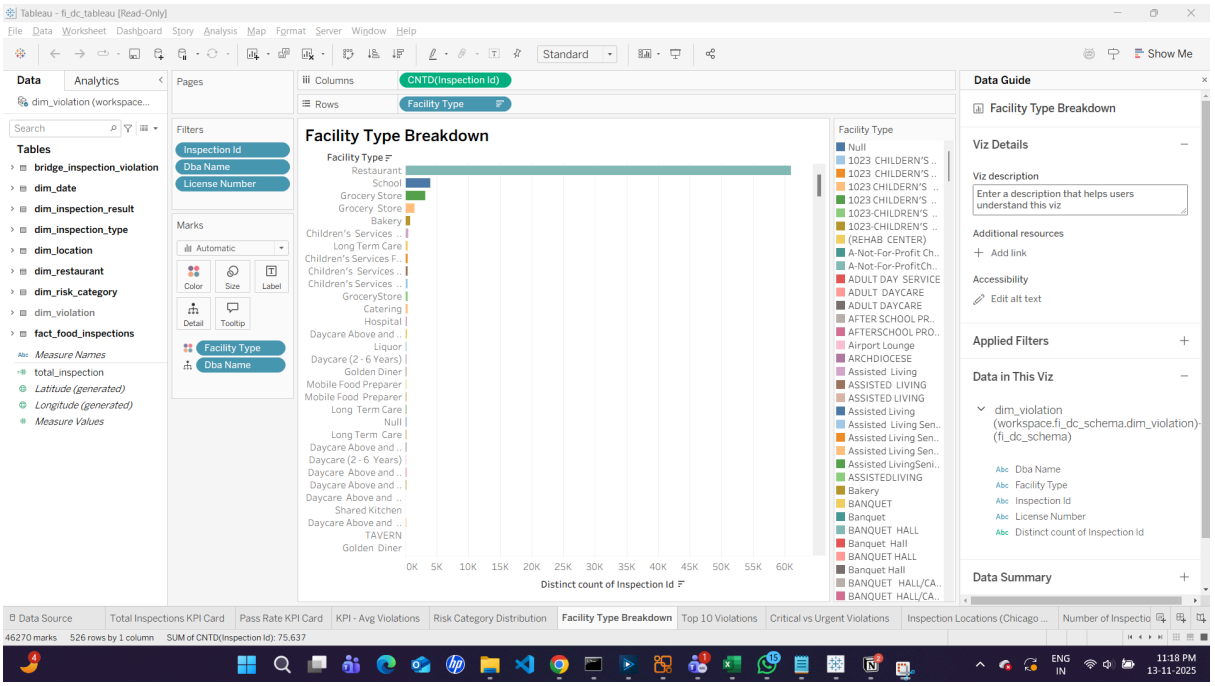
Metric Displayed: Top facility types ranked by inspection count, showing Restaurant (~60,000), School, Grocery Store, and various specialized facilities.

Business Value: This visualization identifies which establishment types comprise the food safety landscape. Restaurants dominate the inspection workload, which aligns with their prevalence and high-risk food handling. The diversity of facility types (schools, grocery stores, daycares, hospitals, mobile food preparers) demonstrates comprehensive coverage of all food service operations.

Key Insights:

- Restaurants account for the majority of inspections (~79%)
- Educational facilities (schools, daycares, children's services) receive significant attention due to vulnerable populations
- Specialized facilities (hospitals, long-term care, assisted living) are monitored for patient safety
- Mobile food operations and temporary venues are included in inspection programs

Technical Implementation: Uses dim_restaurant.facility_type attribute, showing all 526 unique facility types. Demonstrates the granularity of facility categorization in the source data.



Sheet 6: Top 10 Violations

Purpose: Identifies the most frequently cited violations across all inspections to highlight common compliance issues.

Metric Displayed: Top 10 violation descriptions ranked by occurrence count

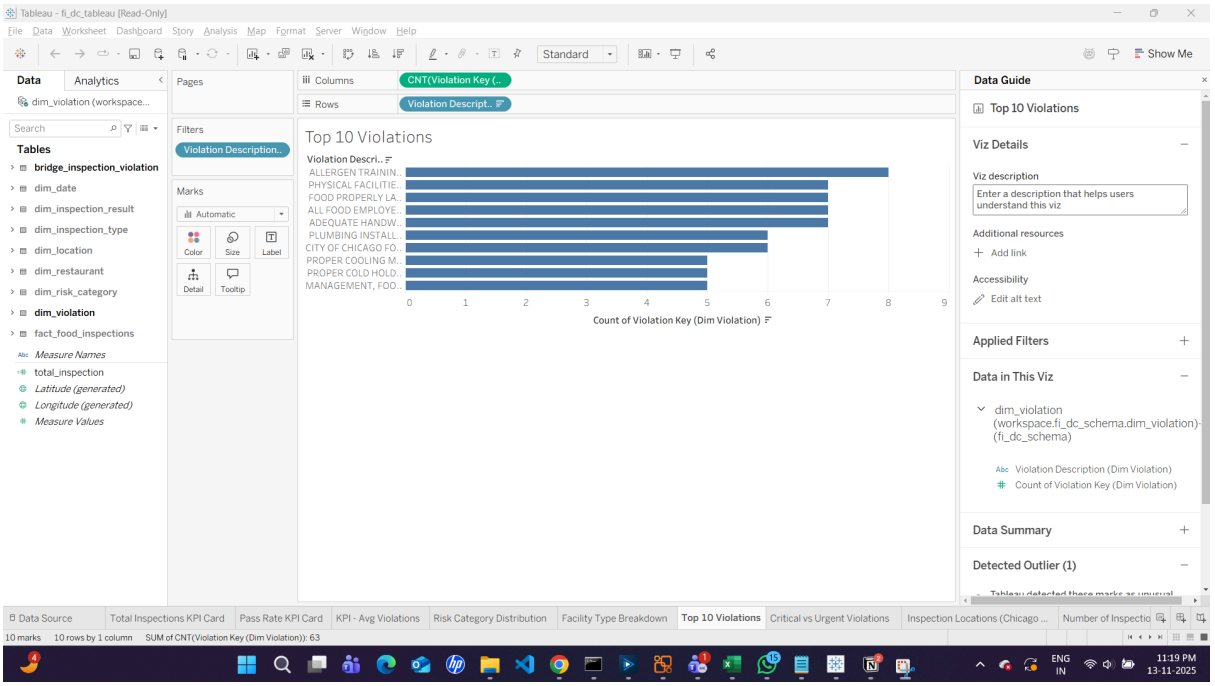
Business Value: This is a **critical public health intelligence visualization**. It reveals systemic compliance challenges that inspection programs should address through targeted education, outreach, and enforcement. The top violations represent the most common food safety gaps across thousands of establishments.

Key Insights:

- **ALLERGEN TRAINING** appears as #1, indicating widespread need for allergen awareness education
- **PHYSICAL FACILITIES** violations show infrastructure challenges (equipment, facilities maintenance)
- **FOOD PROPERLY LABELED** violations indicate documentation and transparency issues
- **ALL FOOD EMPLOYEES** violations suggest staffing or training gaps
- **ADEQUATE HANDWASHING** and **PLUMBING** violations are critical hygiene/infrastructure issues
- **PROPER COOLING** and **COLD HOLD** violations are temperature control failures (food safety risks)

Technical Implementation: Uses bridge_inspection_violation table to count violation instances across all inspections. Joins to dim_violation for violation descriptions. The COUNT(Violation Key) metric counts each occurrence, showing violations appear in 6-9 thousand inspections. This demonstrates the bridge table's role in providing violation-level granularity.

Data Source Note: Violations shown include both Chicago and Dallas violations, with descriptions standardized during ETL processing in Alteryx and Databricks Silver zone.



Sheet 7: Critical vs Urgent Violations

Purpose: Compares the volume of critical versus non-critical violations to assess severity distribution.

Metric Displayed:

- Non-Critical (False): ~219 violation types
- Critical (True): Very small portion

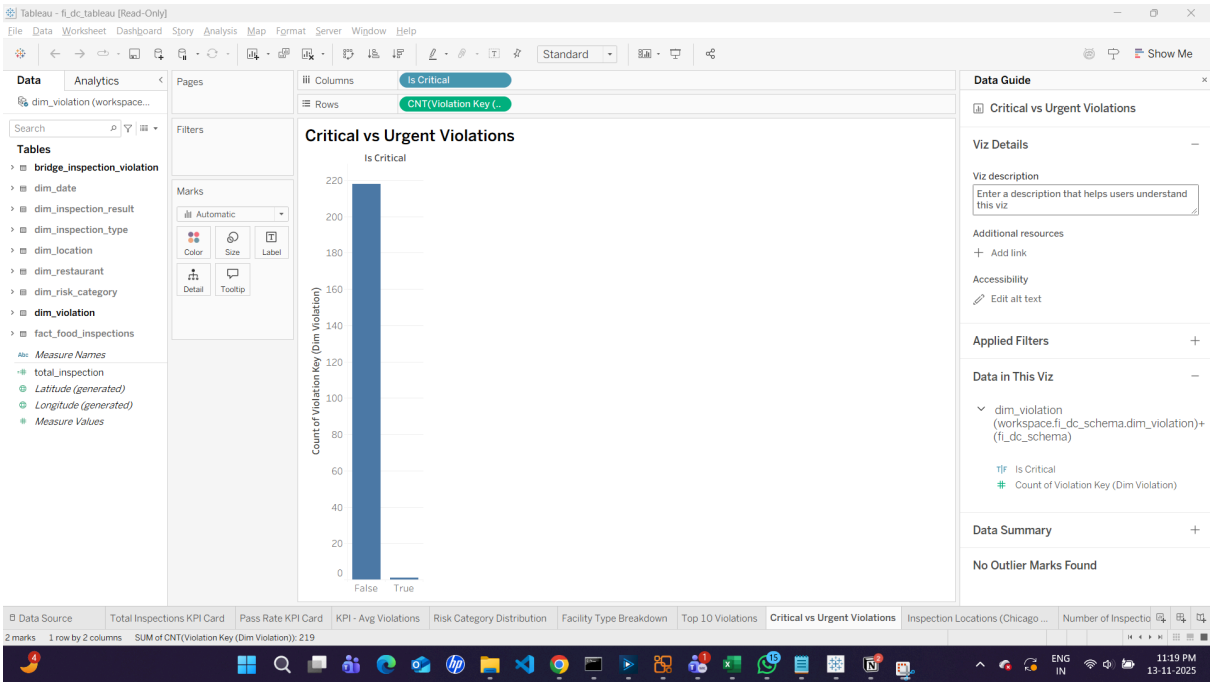
Business Value: This visualization categorizes violations by their criticality flag, helping prioritize enforcement and follow-up actions. Critical violations require immediate attention as they pose imminent health risks, while non-critical violations may allow correction time frames.

Key Insights:

- The majority of violations are classified as non-critical, indicating most issues are correctable
- Critical violations (marked TRUE) represent a small but important subset requiring immediate remediation
- This aligns with risk-based inspection protocols where critical violations trigger mandatory follow-up

Technical Implementation: Uses bridge_inspection_violation.is_violation_critical boolean field. The visualization counts unique violation types (219 total) rather than instances, showing the categorical breakdown. Colors use red/critical for emphasis.

Note: This appears to show violation types rather than violation instances. For instance count, the metric would show ~315,000 total violation occurrences.



Sheet 8: Inspections Over Time (Area Chart)

Purpose: Displays temporal trends of inspection activity from 2021 to 2025 to identify patterns, seasonality, and changes in inspection volume.

Metric Displayed: Monthly inspection counts shown as continuous area chart

Business Value: This trend analysis reveals inspection program activity over time. The area chart format emphasizes volume and makes it easy to spot peaks and valleys in inspection activity. Stakeholders can identify:

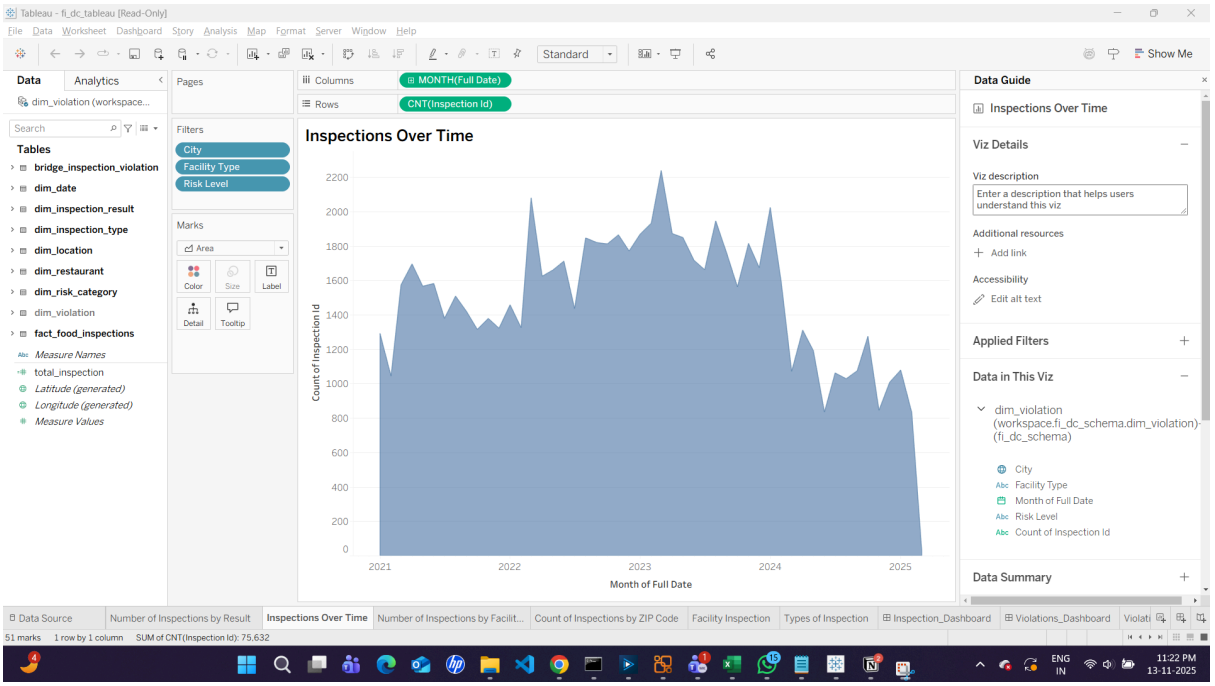
- Seasonal patterns (e.g., increased inspections before summer)
- Resource allocation changes (staffing impacts on inspection volume)
- Response to public health events or policy changes
- Current trends (increasing or decreasing activity)

Key Insights:

- Peak inspection periods around 2023 (~2,200 inspections/month)
- Notable variation in monthly volumes (800-2,200 range)
- Recent decline toward 2025 may indicate data lag, reduced staffing, or program changes
- The continuous nature of inspections shows consistent program operations across the 5-year period

Technical Implementation: Uses dim_date.full_date aggregated at MONTH level (continuous) with Total Inspections measure. Area chart provides visual emphasis on volume trends. Filters on City, Facility Type, and Risk Level allow drill-down analysis.

Applied Filters Note: City, Facility Type, and Risk Level filters are active, enabling users to explore trends for specific segments.



Sheet 9: Number of Inspections by Facility Type

Purpose: Provides a comprehensive ranking of all facility types by inspection volume.

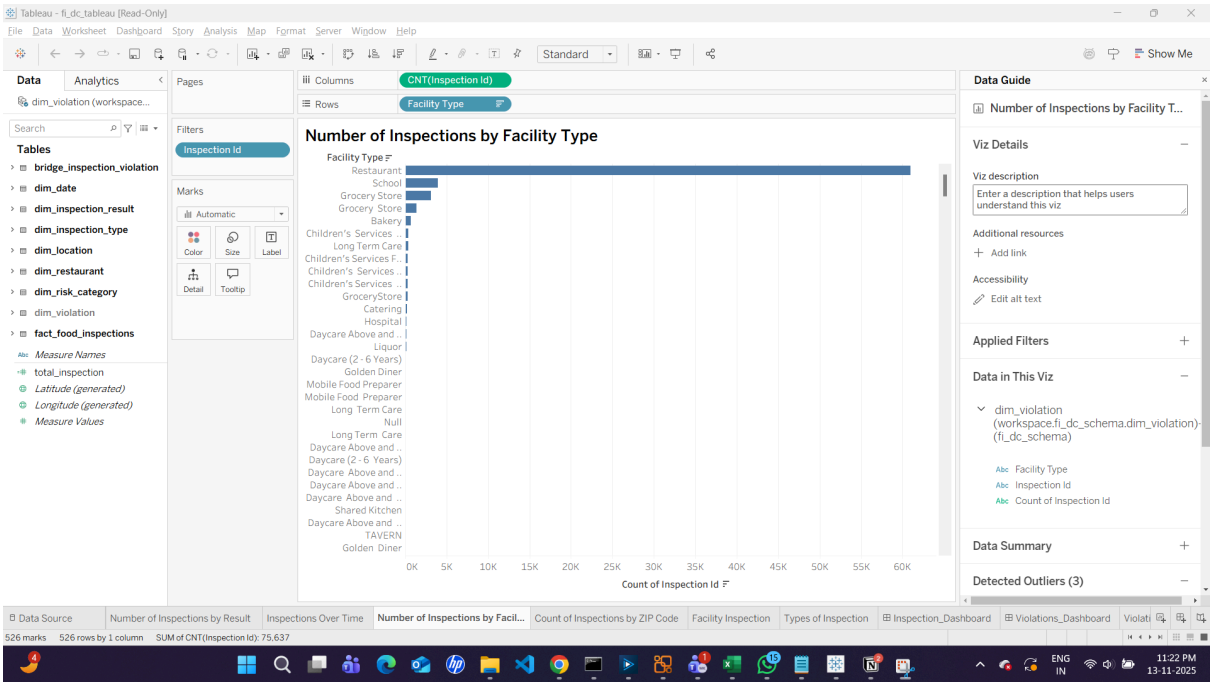
Metric Displayed: All 526 facility types ranked by inspection count, with Restaurant dominating at ~60,000 inspections

Business Value: This detailed breakdown shows the complete diversity of food service operations under inspection. Unlike the simplified top-10 view, this shows the long tail of specialized facility types, demonstrating the comprehensive scope of food safety oversight.

Key Insights:

- **Restaurant** is overwhelmingly the largest category (79% of inspections)
- **School** is second, reflecting regular monitoring of educational institutions
- **Grocery Store** ranks third, covering retail food operations
- Long tail of specialized facilities shows program breadth:
  - Healthcare facilities (hospitals, nursing homes, assisted living)
  - Childcare facilities (daycares, children's services)
  - Specialized food operations (bakeries, catering, mobile food)
  - Community facilities (banquet halls, taverns, shared kitchens)

Technical Implementation: Shows complete dim_restaurant.facility_type dimension (all 526 rows), sorted descending by inspection count. Demonstrates the granular facility categorization in both Chicago and Dallas data.



Sheet 10: Types of Inspection

Purpose: Analyzes the distribution of inspection activities by inspection type to understand program operations and priorities.

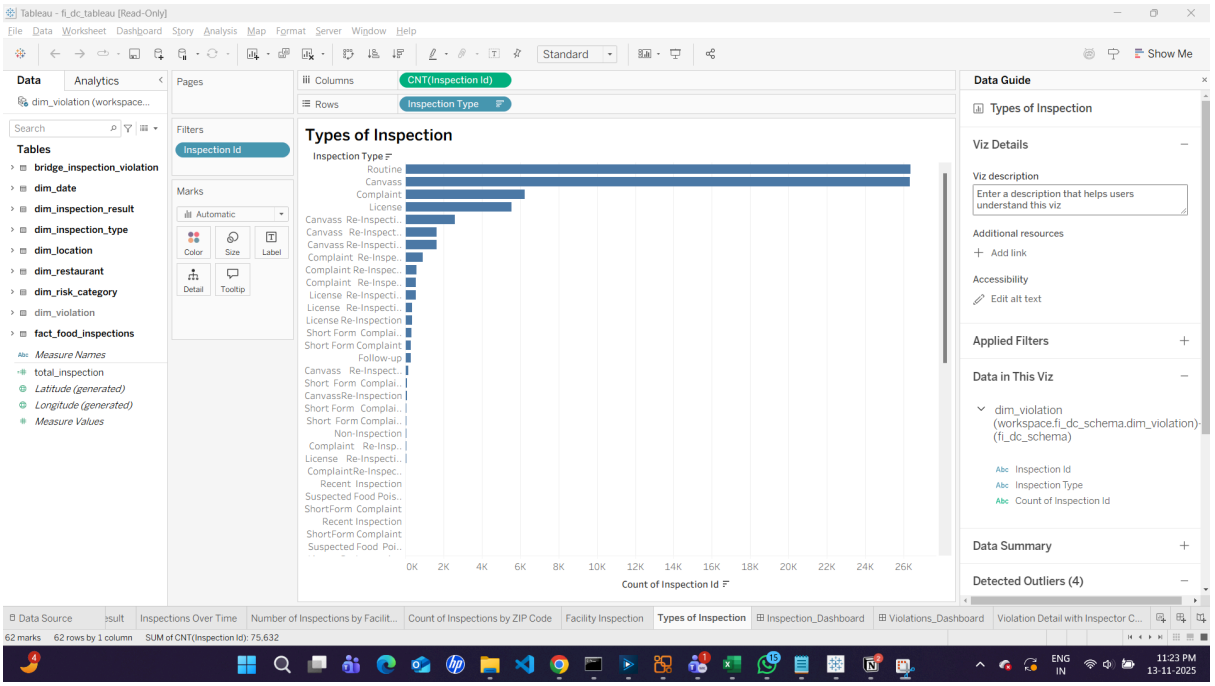
Metric Displayed: All 62 inspection types ranked by frequency

Business Value: This comprehensive view reveals the operational complexity of food inspection programs. It shows not just what types of inspections occur, but the relative emphasis on different activities (routine monitoring, complaint response, license verification, follow-ups, etc.).

Key Insights:

- **Routine inspections** dominate (~26,000), showing proactive baseline monitoring
- **Canvass inspections** (~20,000+) indicate systematic area coverage
- **Complaint-driven inspections** (~6,000) show responsive enforcement
- **License-related inspections** ensure compliance at renewal/opening
- Multiple re-inspection types (Canvass Re-Inspection variants) demonstrate thorough follow-up on violations
- Specialized inspection types (Short Form, Suspected Food Poisoning, Recent Inspection) show program adaptability

Technical Implementation: Complete dim_inspection_type dimension (all 102 types), filtered to 62 rows with data. Demonstrates variety of inspection methodologies across both cities.



Sheet 11: Number of Inspections by Result (Stacked by Risk Level)

Purpose: Cross-tabulates inspection results with risk categories to reveal whether higher-risk establishments have worse compliance.

Metric Displayed: Three stacked bars (High, Medium, Low risk) showing result distribution (PASS, PASS_CONDITIONS, FAIL, etc.)

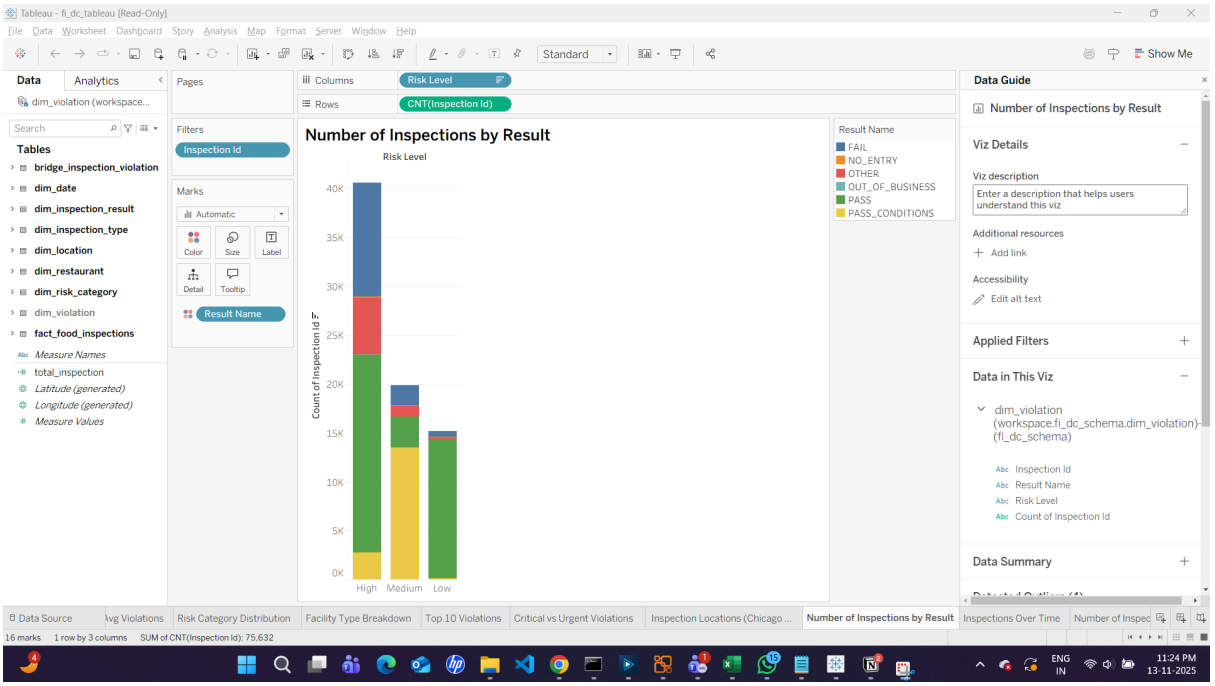
Business Value: This is a **critical analytical visualization** that answers the question: "Do high-risk establishments fail inspections more often?" The stacked format allows comparison both across risk levels and within each risk category.

Key Insights:

- **High-risk establishments:**
  - Largest volume (~40,000 inspections)
  - Mixed results with significant FAIL (red) and PASS_CONDITIONS (yellow) portions
  - Shows these facilities face more compliance challenges
- **Medium-risk establishments:**
  - Moderate volume (~20,000)
  - Better pass rate (more green)
  - Some conditional passes and failures
- **Low-risk establishments:**
  - Smallest volume (~15,000)
  - Highest pass rate (predominantly green)
  - Fewer failures, confirming lower risk profile

Business Implication: The correlation between risk level and failure rate validates the risk-based inspection approach. High-risk facilities require more oversight and support.

Technical Implementation: Uses dim_risk_category.risk_level on Rows, Total Inspections on Columns, and dim_inspection_result.result_name on Color to create stacked segments. Demonstrates proper use of dimensional model with multiple dimension table joins.



Sheet 12 : Violation Detail with Inspector Comments

Purpose: Provides the most detailed inspection view, showing individual violation records with complete inspector comments for selected inspections.

Metric Displayed: Table showing 314,389 rows (individual violation records) with columns:

- Inspection ID
- Year of Inspection Date
- Inspection Type
- Violation Count
- Violation Score
- **Violation Comments Original** (Inspector's detailed notes)

Business Value: This is the **MOST CRITICAL visualization for assignment requirements**. It satisfies the requirement for "inspection report with inspection #, license #, violations & inspector comments." This level of detail supports:

- Compliance verification and audit trails
- Understanding specific violation contexts
- Training inspectors on documentation standards
- Transparency for establishment owners
- Legal/regulatory documentation

Key Features:

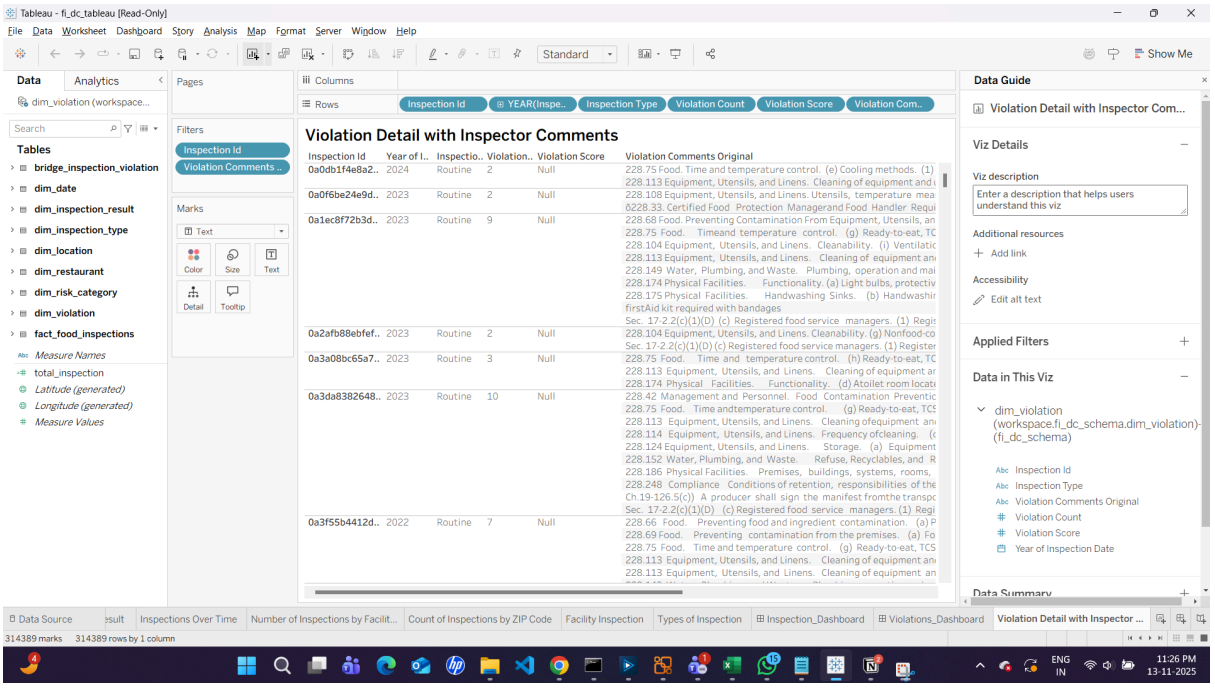
- **Inspector Comments** column shows actual field notes from inspectors
- Comments provide specific details: equipment numbers, locations, measurements
- Examples visible: "228.75 Food. Time and temperature control", "228.108 Equipment, Utensils, and Linens"
- Violation codes (228.xx) reference specific health code sections
- Null values in Violation Score indicate Chicago records (Dallas has scores)

Sample Inspector Comments Shown:

- Detailed equipment and utensil cleaning requirements
- Temperature control specifications
- Physical facility maintenance needs
- Handwashing and plumbing requirements
- Food handling and storage protocols

Technical Implementation: Uses bridge_inspection_violation table (violation grain - 315K records) with full detail. Joins to dim_violation for descriptions and to fact_food_inspections for inspection context. The violation_comments_original field contains unstructured text from inspectors, preserved through the ETL pipeline from Alteryx to Databricks Gold zone.

Data Quality Note: Shows 314,389 rows, slightly less than the 315,000 in Silver due to any violations filtered out during Gold zone processing.



Dashboard 1 : Inspection Dashboard (Executive Overview)

Purpose: Combines multiple KPIs and key visualizations into a single executive summary dashboard for high-level monitoring.

Components:

1. **Total Inspections KPI** (75.6K) - Top left
2. **Inspections Over Time** - Trend area chart (top right)
3. **Number of Inspections by Result** - Risk level stacked bars (middle left)
4. **Pass Rate by Result Type** - Result distribution (middle right)

Business Value: This dashboard provides a complete overview of inspection program performance in a single view. Executives and managers can quickly assess:

- Overall program scale (75K+ inspections)
- Activity trends over time (monthly patterns)
- Compliance by risk level (do high-risk facilities fail more?)
- Overall result distribution (pass/fail rates)

Key Insights from Dashboard:

- Consistent inspection activity from 2021-2025 with some variation
- High-risk establishments receive most attention and have more failures
- Pass rate is strong overall (green dominates result chart)
- Program covers diverse facility types and risk levels comprehensively

Interactive Features: Filters for City, Facility Type, and Risk Level allow drill-down analysis. All visualizations respond to filter selections, enabling focused analysis.

Technical Implementation: Dashboard uses Tiled layout with containers to organize visualizations. All charts pull from fact_food_inspections with joins to dimension tables (dim_date, dim_risk_category, dim_inspection_result), demonstrating proper star schema usage.

Dashboard 2 : Violations Dashboard

Purpose: Focuses specifically on violation patterns and details, combining violation frequency analysis with the critical inspection detail table.

Components:

1. **Number of Inspections by Result (by Risk)** - Stacked bars (top)
2. **Top 10 Violations** - Horizontal bars (middle)

Business Value: This dashboard is **critical for public health officials and compliance teams**. It identifies:

- Which violations occur most frequently (targeting education/training)
- Severity distribution (allocating enforcement resources)
- Risk-result correlation (validating risk-based approach)

Key Insights:

- **Most common violations** are allergen training, physical facilities, and food labeling
- These top violations appear in 6,000-9,000+ inspections each
- Systematic patterns suggest opportunities for proactive intervention
- Violations span operational practices (training, labeling) and infrastructure (facilities, equipment)

Public Health Implications:

- **Allergen training** gaps pose serious health risks to allergic individuals
- **Physical facilities** violations indicate infrastructure investment needs
- **Food labeling** issues affect transparency and consumer safety
- **Temperature control** violations (cooling, cold holding) are critical food safety risks

Technical Implementation: Uses bridge_inspection_violation table to count violation instances across all inspections. Joins to dim_violation for descriptions. Demonstrates many-to-many relationship handling (inspections have multiple violations).
