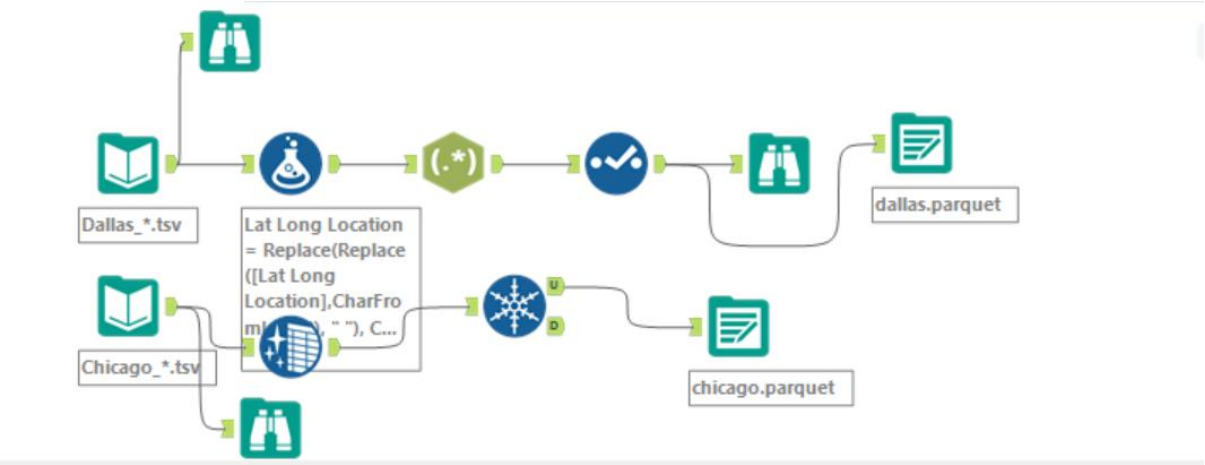
DAMG 7370 Designing Advanced Data Architectures for Business Intelligence

Mid-Term

Contents

Data Profiling Observations



1. Record Volume
    - Chicago: 130,467 raw records
    - Dallas: 47,226 raw records
    - Significant data quality issues requiring validation
2. ZIP Code Data Quality
    - Chicago: All 130,458 records had ZIP stored as float (7-character length: "60659.0")
    - Only 9 true NULL values
    - Dallas: Mixed format - some 5-digit, some with +4 extension
3. Violation Patterns
    - Chicago: Violations stored as pipe-delimited text in single field
    - Dallas: Up to 25 separate violation columns, many sparse (lots of NULLs)
    - Every valid inspection had at least 1 violation (enforced business rule)
4. Scoring Systems
    - Chicago: No numeric scores - only text results (Pass/Fail/Pass w/ Conditions)
    - Dallas: Numeric scores 0-100, with most between 70-100
    - Required derived scoring for Chicago to enable cross-city comparison
5. Location Data Format

- Chicago: Separate latitude/longitude fields as strings
- Dallas: Combined coordinates with embedded newlines causing row splits
- Coordinate precision: Up to 15 decimal places (e.g., 32.662836999)

6. Missing Data Patterns
   - Restaurant names: Very few nulls (high quality)
   - Risk categories: Only present for Chicago
   - License numbers: Only available for Chicago
   - AKA names: Sporadic availability
7. Data Type Inconsistencies
   - Dates stored as strings requiring parsing
   - Numeric fields (scores, coordinates) stored as strings
   - Mixed NULL representations (NULL, empty strings)
8. Text Field Issues
   - Embedded special characters in violation descriptions
   - Inconsistent spacing and formatting
   - Line breaks within data fields (not just row delimiters)

Alteryx transformations before loading into databricks

1. Embedded Newline Characters in Location Field

- **Issue**: Dallas "Lat Long Location" field contained embedded newlines, causing coordinates to split into separate rows

"125 W CAMP WISDOM RD

(32.662836999, -96.824488024)"

- **Solution**: Used Formula tool with Replace(Replace([Lat Long Location], CharFromInt(10), " "), CharFromInt(13), " ") to remove line feed and carriage return characters

2. ZIP Code Format Issues

- **Issue**: Chicago ZIP codes appeared as floats (60659.0) instead of strings
- **Solution**: Applied RegEx tool to remove ".0" suffix: regexp_replace([zip], "\.0$", "")

3. Input Data Tool Configuration

- Set delimiter to \t (tab) for TSV files
- Checked "First Row Contains Field Names"

- Important: Enabled "Allow Newlines in Fields" option to prevent row splitting

4. Column Name Cleanup

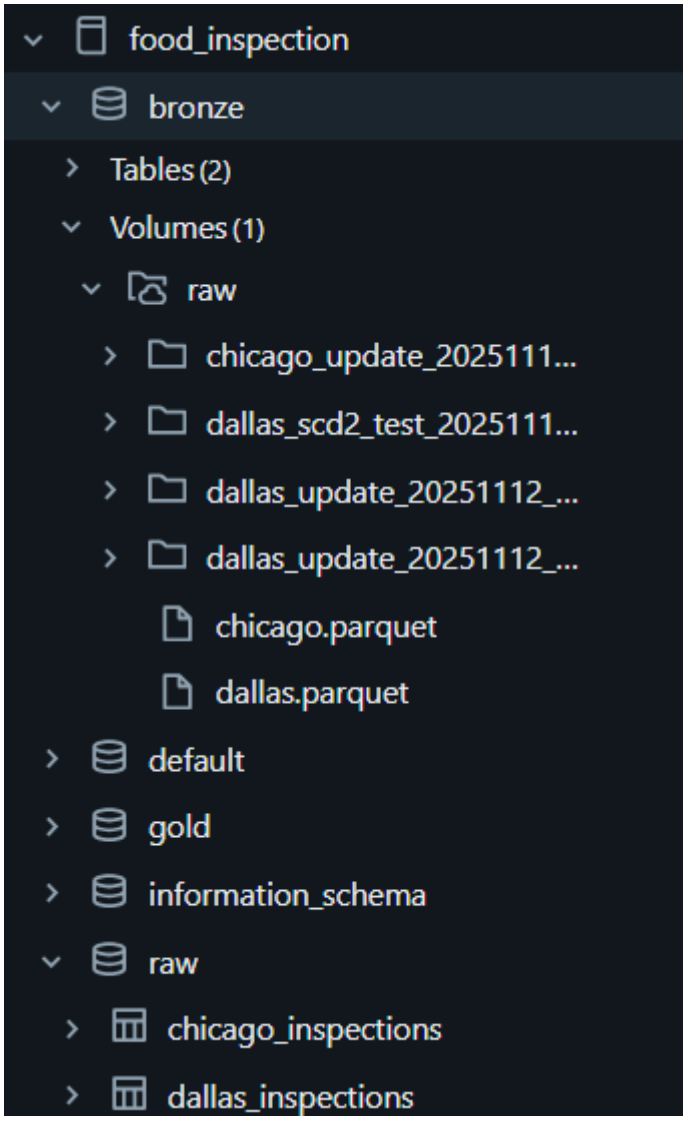
- Used Dynamic Rename or Formula tool to replace spaces with underscores
- Converted special characters in column names (e.g., "License #" to "license")

5. Data Cleansing Tool Usage

- Applied "Remove Line Breaks" option specifically for the Location field
- Used to clean embedded newlines without complex formulas

These Alteryx-specific transformations ensured the tab-delimited data was properly parsed and ready for downstream processing.

Data Loading: From Alteryx to Databricks



Overview

The data pipeline leveraged Alteryx for initial data preparation and cleansing of the restaurant inspection TSV files, with the processed output being written as Parquet files. These files were then loaded into Databricks volumes as raw tables, establishing the foundation for subsequent medallion architecture transformations.

Parquet File Generation in Alteryx

After completing the data cleansing steps including handling embedded newlines, standardizing column names, and addressing the ZIP code formatting issues which Alteryx was configured to output the processed data as Parquet files. The columnar storage format was selected for its compression efficiency and compatibility with Databricks' distributed processing engine. Both Chicago and Dallas datasets were output as separate Parquet files, maintaining their distinct schemas while ensuring clean, database-friendly column names.

Volume Storage and Raw Schema Design

The Parquet files were loaded directly into Databricks volumes without any hierarchical directory structure, following a straightforward raw table approach. The data was ingested into a simple schema structure:

/Volumes/main/raw/

This flat structure aligned with the raw layer philosophy of minimal transformation, preserving the data as close to its Alteryx-processed state as possible.

Data Modelling and Medallion Architecture

Bronze Layer: Raw Data Ingestion with Streaming

Purpose

The Bronze layer serves as the initial ingestion point for raw restaurant inspection data, preserving the original data while adding minimal metadata for tracking and deduplication. This layer uses streaming tables with Change Data Feed (CDF) enabled for real-time processing capabilities.

Architecture Design

Data Sources

- **Chicago**: food_inspection.raw.chicago_inspections
- **Dallas**: food_inspection.raw.dallas_inspections

Both sources contain pre-cleaned data from Alteryx, stored as raw tables in Databricks with database-friendly column names.

Key Features

1. **Streaming Ingestion**
    - Utilizes Spark Structured Streaming with readChangeFeed option
    - Enables real-time processing of new inspection records
    - Supports incremental updates and late-arriving data
2. **Deduplication Strategy**
    - Creates SHA-256 hash of all non-metadata columns

- Uses 1-hour watermark for streaming deduplication
- Prevents duplicate records from downstream processing

3. **Metadata Enrichment**
    - _ingestion_timestamp: Tracks when records entered the pipeline
    - _source_file: Identifies data origin (set to "raw_table")
    - Preserves data lineage for auditing
4. **Change Data Feed Handling**
    - Removes reserved CDF columns before processing
    - Ensures clean data flow to Silver layer
    - Maintains compatibility with streaming operations

Silver Layer: Data Validation and Business Rules

Purpose

The Silver layer transforms and validates the Bronze data, applying business rules and data quality expectations to create trusted datasets ready for analytical processing. This layer implements streaming tables with comprehensive data quality checks using Delta Live Tables expectations.

Data Quality Framework

The Silver layer leverages DLT's expectation framework with @dlt.expect_all_or_drop decorators, ensuring only high-quality records proceed to downstream processing. Failed records are automatically dropped, maintaining data integrity.

Chicago Silver Processing

Validation Rules Applied

1. **Required Fields Validation**
    - dba_name cannot be null
    - inspection_date and inspection_type must exist
    - results field must be populated
2. **ZIP Code Standardization**
    - Removes .0 suffix from float representation
    - Validates exactly 5-digit format
    - Handles the Chicago-specific float storage issue
3. **Violation Processing**
    - Ensures violations field is not empty
    - Counts violations by splitting pipe-delimited text
    - Enforces minimum of 1 violation per inspection

4. **Score Derivation**
    - Implements Chicago-specific scoring logic:
        - Pass → 90
        - Pass w/ Conditions → 80
        - Fail → 70
        - No Entry → 0
        - Others → NULL

Dallas Silver Processing

Validation Rules

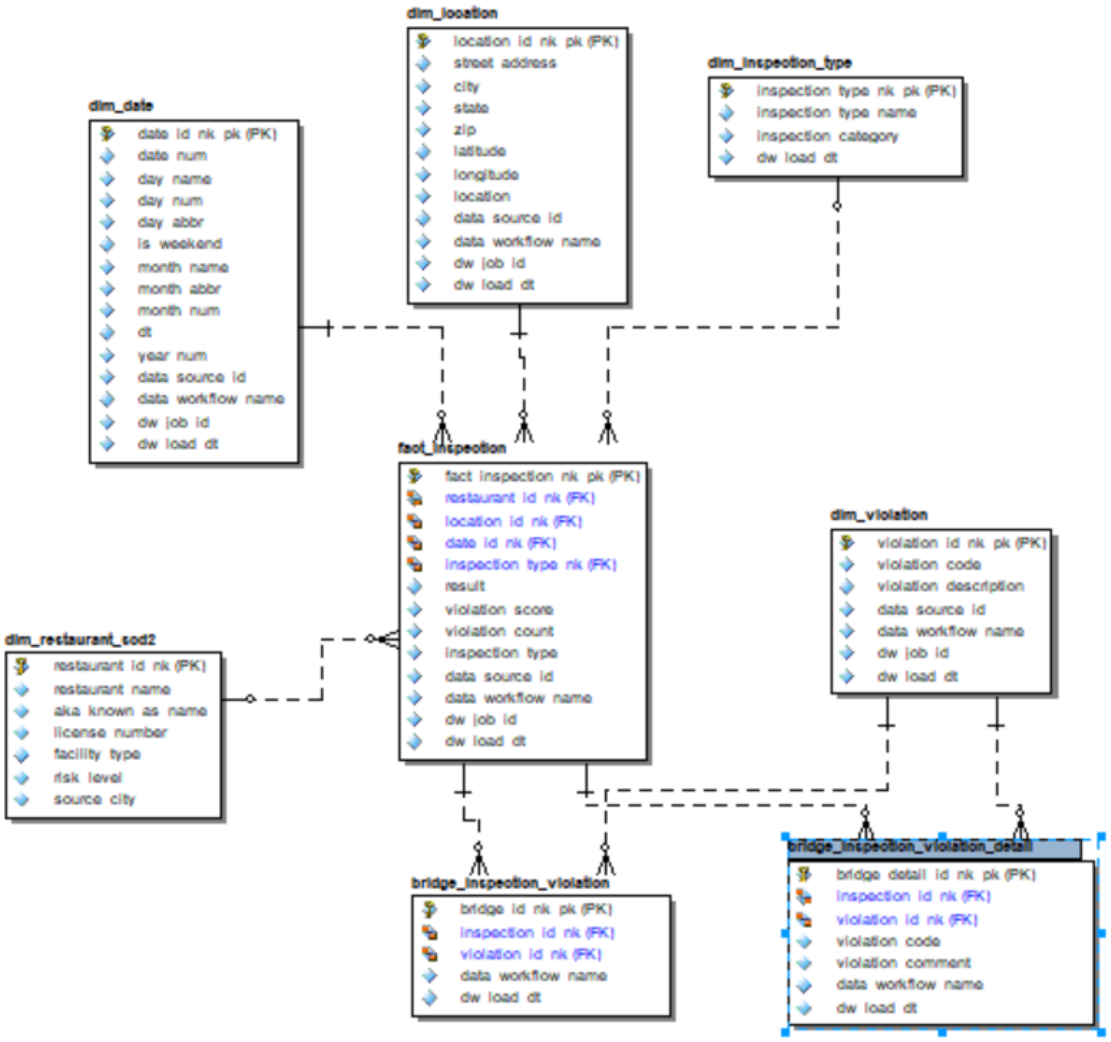
1. **Basic Data Quality**
    - Restaurant name, date, and type validation
    - ZIP code must match 5-digit pattern
    - Inspection scores bounded between 0-100
2. **Violation Aggregation**
    - Creates 25 binary flags for violation presence
    - Aggregates total violation count across all columns
    - Handles sparse violation matrix efficiently
3. **Business Rule Enforcement**
    - **High Score Limit**: Scores ≥90 limited to maximum 3 violations
    - **Critical Violation Check**: Searches all memo fields for "Urgent" or "Critical" keywords
    - **PASS Validation**: Prevents PASS result (score ≥70) with critical violations
4. **Metadata Enhancement**
    - Adds has_critical_violation boolean flag
    - Creates derived_score (same as inspection_score for Dallas)
    - Maintains source tracking with source_city

Technical Implementation Details

Streaming Capabilities

- Both tables use dlt.read_stream() for incremental processing
- CDC columns automatically dropped to prevent conflicts
- Maintains real-time data freshness

Gold Layer



Overview

The Gold layer implements a **dimensional data warehouse** using a **star schema** design pattern with **natural keys** instead of surrogate keys. This layer consolidates data from both Chicago and Dallas restaurant inspection systems into a unified analytical model optimized for business intelligence and reporting.

Architecture Pattern

- **Schema Type**: Star Schema with Bridge Tables
- **Key Strategy**: Natural Keys (business keys) instead of surrogate keys
- **Processing Mode**: Streaming tables using Delta Live Tables (DLT)
- **SCD Implementation**: Type 2 (Slowly Changing Dimensions) for restaurant dimension
- **Change Tracking**: Delta Change Data Feed (CDF) enabled on most tables

Dimension Tables

1. dim_date

Purpose: Calendar dimension providing time-based attributes for analysis

Natural Key: date_id_nk_pk (format: YYYYMMDD as integer)

Key Attributes:

- date_num: Actual date value
- day_name: Full day name (e.g., "Monday")
- day_abbr: Abbreviated day name (e.g., "Mon")
- is_weekend: Boolean flag for weekend dates
- month_name, month_abbr, month_num: Month attributes
- year_num: Year value

Data Source: Extracted from inspection dates across both Chicago and Dallas datasets

Update Pattern: Streaming with deduplication on date values

---

2. dim_location

Purpose: Geographic dimension containing address and coordinate information

Natural Key: location_id_nk_pk (format: CITY_ADDRESS_ZIP)

- Chicago: CHI_[address]_[zip]
- Dallas: DAL_[street_address]_[zip_code]

Key Attributes:

- street_address: Full street address
- city, state, zip: Location identifiers
- latitude, longitude: Geographic coordinates (float)
- source_city: Source system identifier (CHI/DAL)

Data Sources:

- Chicago: address, city, state, zip fields
- Dallas: street_address, zip_code with hardcoded "Dallas, TX"

Update Pattern: Streaming with deduplication on natural key

---

3. dim_restaurant_scd2

Purpose: Restaurant master dimension tracking historical changes

Natural Key: restaurant_id_nk

- Chicago: CHI_[license]_[latitude]_[longitude]
- Dallas: DAL_[restaurant_name]_[lat]_[long]

Key Attributes:

- restaurant_name: Primary business name (DBA name for Chicago)
- aka_known_as_name: Alternative name (Chicago only)

- license_number: Business license number (Chicago only)
- facility_type: Type of food establishment
- risk_level: Risk classification
- source_city: Source system identifier

SCD Type 2 Implementation:

- **Tracked Changes**: restaurant_name, aka_known_as_name, risk_level, facility_type, license_number
- **Sequence Column**: source_timestamp (ingestion timestamp)
- **System Columns**: __START_AT, __END_AT, __CURRENT (auto-generated by DLT)

Risk Level Mapping (Dallas):

- Score < 80: "Risk 1 (High)"
- Score < 90: "Risk 2 (Medium)"
- Score ≥ 90: "Risk 3 (Low)"

Update Pattern: DLT's apply_changes() with SCD Type 2

Testing SCD 2 on restaurant_dim

Updating name of a restaurant in raw Chicago table

Running the pipeline to capture changes

DIM_RESTAURANT Scd2 captures the updated record and shows history

---

4. dim_violation

Purpose: Violation code master dimension

Natural Key: violation_id_nk_pk (format: CITY_[violation_code])

Key Attributes:

- violation_code: Numeric violation code
- violation_description: Full violation description/title
- data_workflow_name: Source city (CHI/DAL)

Data Extraction:

- **Chicago**: Parsed from pipe-delimited violations field using regex pattern ^(\d+)\.\s*(.+?)\s*-\s*Comments:
- **Dallas**: Extracted from 25 violation_description columns (violation_description_1 through violation_description_25)

Update Pattern: Streaming with deduplication on composite key (violation_code + source_city)

---

5. dim_inspection_type

Purpose: Classification dimension for inspection types

Natural Key: inspection_type_nk_pk (inspection type name)

Key Attributes:

- inspection_type_name: Full inspection type description
- inspection_category: Categorized type
    - "Complaint-Based": Contains "Complaint"
    - "Follow-up": Contains "Re-Inspection"
    - "Licensing": Contains "License"

    - "Routine": All others

Data Sources: Unique inspection_type values from both Chicago and Dallas

Update Pattern: Streaming with deduplication on inspection_type

---

Fact Table

fact_inspection

Purpose: Central fact table containing inspection measurements and events

Natural Key: fact_inspection_nk_pk

- Chicago: inspection_id from source
- Dallas: [restaurant_name]_[inspection_date]

Foreign Keys (Natural Keys):

- restaurant_id_nk → dim_restaurant_scd2
- location_id_nk → dim_location
- date_id_nk → dim_date
- inspection_type_nk → dim_inspection_type

Measures:

- violation_score: Numeric score/count
- violation_count: Total number of violations
- result: Pass/Fail/Other outcomes

Degenerate Dimensions:

- inspection_type: Stored directly in fact table

Audit Columns:

- data_source_id: Source inspection ID
- data_workflow_name: Source city (CHI/DAL)
- dw_job_id: Job identifier
- dw_load_dt: Load timestamp
- streaming_timestamp: Ingestion timestamp (used for watermarking)

Result Mapping (Dallas):

- Score ≥ 70: "Pass"
- Score < 70: "Fail"

Update Pattern: Streaming with 2-hour watermark and deduplication on natural key

---

Bridge Tables

1. bridge_inspection_violation

Purpose: Many-to-many relationship between inspections and violations (normalized)

Natural Key: bridge_id_nk_pk (format: [inspection_id_nk]_[violation_id_nk])

Key Attributes:

- inspection_id_nk: Links to fact_inspection
- violation_id_nk: Links to dim_violation
- data_workflow_name: Source city

Data Extraction:

- **Chicago**: Exploded from pipe-delimited violations field
- **Dallas**: Flattened from 25 violation_description columns

Update Pattern: Streaming with 2-hour watermark and deduplication on composite key

---

2. bridge_inspection_violation_detail

Purpose: Many-to-many relationship with additional violation context

Natural Key: bridge_detail_id_nk_pk (format: [inspection_id_nk]_[violation_id_nk])

Key Attributes:

- inspection_id_nk: Links to fact_inspection
- violation_id_nk: Links to dim_violation
- violation_code: Denormalized code for convenience
- violation_comment: Inspector comments/notes
- data_workflow_name: Source city

Data Extraction:

- **Chicago**: Comments extracted using regex pattern -\s*Comments:\s*(.+)$
- **Dallas**: Sourced from violation_memo columns (violation_memo_1 through violation_memo_25)

Update Pattern: Streaming with 2-hour watermark and deduplication on composite key

---

Design Decisions

Natural Keys vs Surrogate Keys

Rationale: Natural keys are used throughout to:

- Maintain referential integrity in streaming scenarios
- Enable idempotent processing and reprocessing
- Simplify joins without key lookup overhead
- Preserve business meaning in keys

Visualization

The connection between Databricks gold layer tables and Power BI was established using Power BI's native Databricks connector. After completing the medallion architecture (Bronze → Silver → Gold) in Databricks using Delta Live Tables (DLT), the dimensional model tables (dim_date, dim_location, dim_restaurant_scd2, dim_violation, dim_inspection_type) and fact tables (fact_inspection) along with bridge tables were exposed in the Unity Catalog. In Power BI Desktop, we connected directly to the Databricks SQL Warehouse endpoint, authenticated using Azure Active Directory, and imported the gold layer tables

1. Summary Page

Key Performance Indicators (KPI Cards)

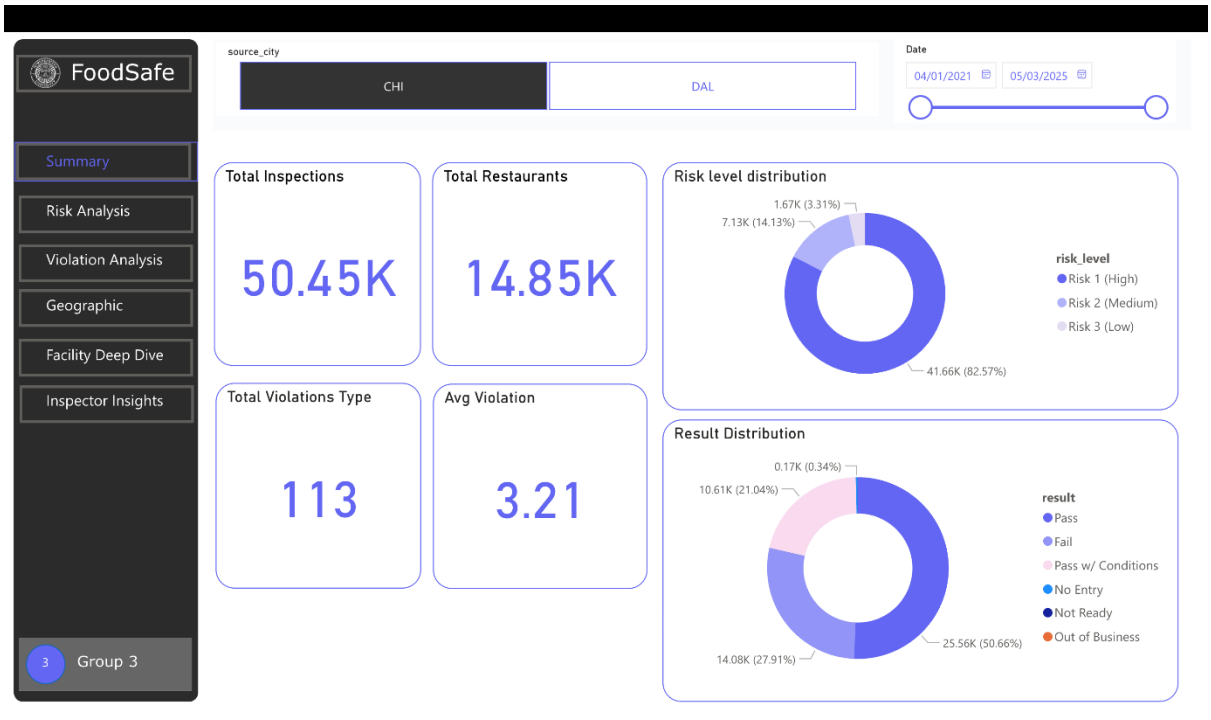
- Total Inspections: Displays the cumulative count of all inspections (69.19K)
- Total Restaurants: Shows the unique count of restaurants inspected (25.98K)
- Total Violations Type: Number of distinct violation categories identified (113)
- Avg Violation: Average violations per inspection (3.84)

Risk Distribution (Donut Chart)

- Purpose: Shows the percentage breakdown of restaurants by risk level
- Insights: Risk 1 (High) comprises 65.41%, Risk 2 (Medium) 21.02%, Risk 3 (Low) 13.58%
- Visual: Color-coded donut chart with percentages and counts

Inspection Results Distribution (Donut Chart)

- Purpose: Displays the outcome distribution of all inspections
- Insights: Pass rate of 63.73%, with Fail at 20.6% and Pass w/ Conditions at 15.39%
- Visual: Multi-colored donut showing all result types including rare outcomes



2. Risk Analysis Page

Root Causes of Failed Inspections (Sankey Diagram)

- Purpose: Traces the flow from failure results through risk levels to specific violations
- Key Flow: Shows how 84,508 failed inspections distribute across risk levels and violation types
- Top Violations:
    - Insects, Rodents & Animals (14,178)

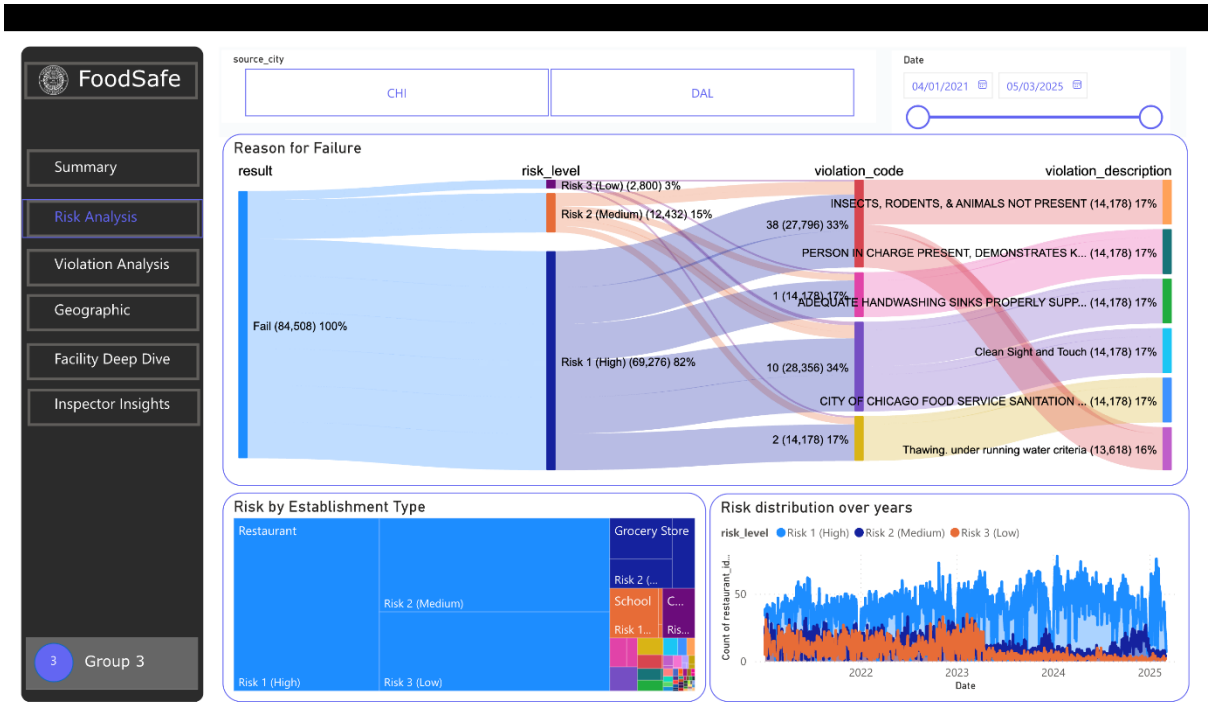
- Person in Charge demonstrations (14,178)
- Handwashing/Sink compliance (14,178)

Risk by Establishment Type (Treemap)

- Purpose: Visualizes risk distribution across different facility types
- Insights: Restaurants dominate, followed by Grocery Stores and Schools
- Color Coding: Risk levels shown through color intensity within each facility type

Risk Distribution Over Years (Area Chart)

- Purpose: Shows temporal trends in risk levels from 2021-2025
- Insights: Tracks the changing proportion of high, medium, and low risk establishments
- Visual: Stacked area chart showing risk level evolution



3. Violation Analysis Page

Top 10 Violation Comments (Bar Chart)

- Purpose: Identifies most frequent violation descriptions
- Top Issues:
  - CLEAN ALL (400 occurrences)
  - PROVIDE FOR ALL (350)
  - Sign about inspection results (240)

- Format: Horizontal bar chart with truncated descriptions

Inspections Distributed by Violation Code (Treemap)

- Purpose: Shows the relative frequency of different violation codes
- Visual: Size represents frequency, with codes like 10, 47, 21, and 38 being prominent
- Insight: Reveals which specific violation codes occur most frequently

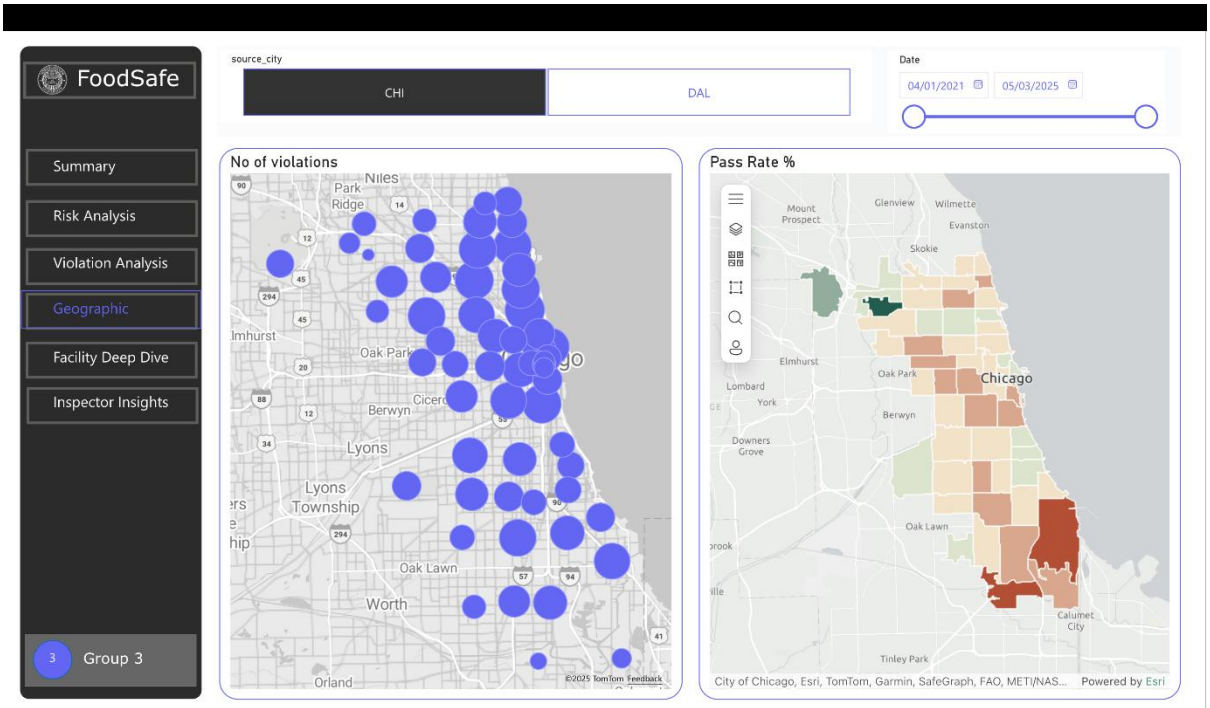
4. Geographic Page

Number of Violations Map (Bubble Map)

- Purpose: Geographic distribution of inspection violations
- Visual: Bubble size indicates violation density in each area
- Coverage: Shows concentration in Chicago area with visible clustering
- Note: System warning about visual being retired

Pass Rate % (Not visible in screenshot)

- Purpose: Would show pass rates by geographic area
- Expected Visual: Choropleth or heat map showing performance by location



5. Facility Deep Dive Page

Restaurant Search Dropdown

- Purpose: Allows users to search and select specific restaurants
- Example: "#1 CHOP SUEY RESTAURANT, INC" selected

Performance Cards

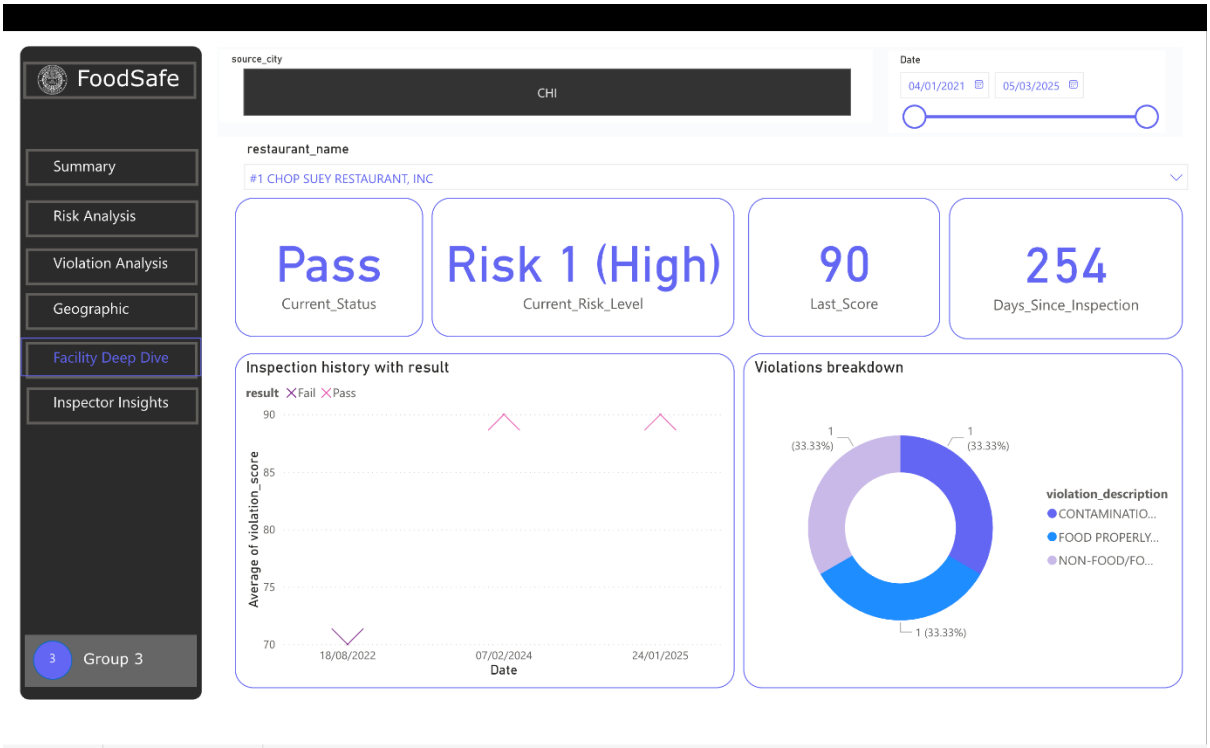
- Current Status: Shows latest inspection result (Pass)
- Current Risk Level: Displays risk category (Risk 1...)
- Last Score: Most recent inspection score (90)
- Days Since Inspection: Time elapsed since last inspection (253 days)

Inspection History with Result (Line Chart)

- Purpose: Tracks inspection scores over time
- Visual: Shows performance trend with Pass/Fail indicators
- Y-axis: Average violation score (70-90 range)

Violations Breakdown (Donut Chart)

- Purpose: Distribution of violation types for selected restaurant
- Categories: Shows CONTAMINATION, FOOD PROPERLY, and NON-FOOD/FOOD each at 33.33%



6. Inspector Insights Page

Inspections by Inspection Type (Donut Chart)

- Purpose: Breakdown of inspection types performed
- Distribution:
    - Canvass: 39.32%
    - Routine: 26.53%
    - License Re-Inspection: 18.18%
- Total Types: Shows 7 different inspection categories

Violation Comments Word Cloud

- Purpose: Visual representation of common terms in violation comments
- Note: Placeholder shown - would display frequently used words in comments

Inspection Results Over the Years (Stacked Bar Chart)

- Purpose: Annual trend of inspection outcomes from 2021-2025
- Visual: 100% stacked bars showing proportion of Pass/Fail/Other results
- Trend: Shows consistency in result distribution over time

Weekend vs Weekday Outcome (Grouped Bar Chart)

- Purpose: Compares inspection results between weekdays and weekends
- Key Insight: Weekend inspections show 98.83% pass rate vs 63.02% on weekdays
- Visual: Grouped bars showing False (weekday) vs True (weekend) performance

Technical Notes

- City Filter: Toggle between CHI (Chicago) and DAL (Dallas) data
- Date Range: Slider allowing selection from 1/4/2021 to 3/5/2025
- Color Scheme: Consistent use of blues/purples for primary metrics, red/yellow/green for risk levels
- Navigation: Left sidebar provides easy access to all dashboard sections