Designing Advanced Data Architecture for Business Intelligence

Final Project Report

Team 6

Bhagyashri Pagar	NUID: 002310690
Abisha Vadukoot	NUID: 002371769
Rahul Bothra	NUID: 002378790

Table of Contents

1. Introduction

Food safety inspections represent a critical public health function in urban environments, helping to prevent foodborne illness and ensure compliance with health standards across thousands of food establishments. This project undertakes a comprehensive analysis of food establishment inspection data from two major American cities Chicago and Dallas to extract meaningful insights, identify patterns, and develop a unified data model that accommodates different municipal reporting systems.

The analysis spans multiple years of inspection data (2021-present) from both cities, representing tens of thousands of inspections across diverse establishment types. Chicago's dataset comprises five TSV files totaling approximately, featuring 17 core fields with consolidated violation information and a risk classification system that includes both numeric and descriptive components. Dallas's dataset similarly consists of five TSV files but structures data differently, with up to 25 separate violation entries per inspection, detailed address components, and a score-based evaluation system.

Despite their differences in structure, coding systems, and evaluation methodologies, both datasets capture essential information about food safety compliance. Chicago uses a categorical pass/fail/conditional system with risk levels, while Dallas employs a numerical scoring approach. Chicago records violations in consolidated text fields that require extensive parsing, whereas Dallas distributes violation information across multiple dedicated fields. Chicago's address information appears in a unified format, while Dallas breaks address into separate components with varying levels of completeness.

By transforming these complex datasets through the medallion architecture progressing from raw TSV files (Bronze) through cleaned and standardized Parquet files (Silver) to a unified dimensional model (Gold) we create a valuable analytical resource that supports data-driven public health decision-making while offering unprecedented transparency into food safety compliance across major metropolitan areas.

2. Data Sources

2.1 Chicago Food Inspections

Data Origin: City of Chicago Department of Public Health's Food Protection Program

Source: City of Chicago Data Portal (https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5)

Coverage Period: 2021 to present

File Structure:

- Chicago_2021-2022.tsv
- Chicago_2022-2023.tsv
- Chicago_2023-2024.tsv
- Chicago_2024-2025.tsv
- Chicago_2025-present.tsv

Chicago Data Schema: The Chicago inspection data includes the following fields:

1. Inspection_ID - Unique identifier for each inspection
2. DBA_Name - Doing Business As (restaurant or business name)
3. AKA_Name - Also Known As (alternative business name)
4. Facility_Type - Type of establishment (restaurant, grocery, bakery, etc.)
5. Address - Street address of establishment
6. City - City location (primarily Chicago)
7. State - State location (primarily IL)
8. Zip - ZIP code
9. Inspection_Date - Date of inspection
10. Inspection_Type - Type of inspection (canvass, complaint, etc.)
11. Inspection_Results - Inspection result (pass, pass with conditions, fail)
12. Latitude - Geographic coordinate
13. Longitude - Geographic coordinate
14. Violation_Category_ID - Identifier for violation category

15. Violation_Category - Description of violation category
16. Violation_Comment - Detailed violation descriptions and comments
17. Risk_Level - Risk category of the establishment

Description of Chicago Inspection Process: According to the Chicago Data Portal, inspections are performed by staff from the Chicago Department of Public Health's Food Protection Program using a standardized procedure. The results are inputted into a database, then reviewed and uploaded to the data portal.

Chicago has approximately 15,000 food establishments across the city subject to food safety inspections. Inspectors are responsible for checking these establishments to ensure compliance with health codes designed to prevent foodborne illness. The inspection frequency is based on risk level:

- Risk 1 (High-Risk): Inspected twice yearly
- Risk 2 (Medium-Risk): Inspected once yearly
- Risk 3 (Low-Risk): Inspected once every two years

The city also conducts inspections in response to complaints from the public regarding potential food safety issues.

2.2 Dallas Food Inspections

Data Origin: City of Dallas Department of Health, Food Safety Division

Coverage Period: 2021 to present

File Structure:

- Dallas_2021-2022.tsv
- Dallas_2022-2023.tsv
- Dallas_2023-2024.tsv
- Dallas_2024-2025.tsv
- Dallas_2025-present.tsv

Dallas Data Schema: The Dallas food inspection data includes the following key fields:

1. Restaurant Name - Name of the establishment
2. Inspection Type - Category of inspection conducted
3. Inspection Date - Date when inspection was performed

4. Inspection Score - Numerical score assigned during inspection
5. Street Number - Building number of the establishment
6. Street Name - Name of the street where establishment is located
7. Street Direction - Directional prefix (N, S, E, W) for the street
8. Street Type - Type of street (Ave, St, Blvd, etc.)
9. Street Unit - Unit or suite number
10. Street Address - Complete address of the establishment
11. Zip Code - Postal code of the establishment location
12. Violation Description (1-25) - Description of the violation
13. Violation Points (1-25) - Points assessed for the violation
14. Violation Detail (1-25) - Specific details about the violation
15. Violation Memo (1-25) - Inspector's notes about the violation
16. Inspection Month - Month when inspection was conducted
17. Inspection Year - Year when inspection was conducted
18. Lat Long Location - Geographical coordinates of the establishment

Note on Data Differences: The Dallas and Chicago datasets have significantly different schemas and structures:

- **Violation Recording**: Chicago records violations in a single field, while Dallas uses a structured approach with up to 25 separate violation entries per inspection, each with description, points, details, and inspector notes
- **Address Formatting**: Dallas breaks down addresses into multiple components (number, name, direction, type, unit), while Chicago uses a single address field
- **Scoring System**: Dallas uses a numerical scoring system, whereas Chicago primarily uses categorical results (pass/fail/conditional)
- **Data Quality Challenges**: Dallas data shows high null percentages in many fields, particularly in the violation records beyond the first 11 entries
- **Field Standardization**: Both datasets require extensive data cleaning, with the Dallas data needing particular attention to case standardization, special character removal, and type conversion

These differences create significant data integration challenges, requiring careful mapping and transformation to create a unified dimensional model that preserves the information from both cities.

3. Business Objectives

The primary business objectives are:

- Analyzing food inspection results by facility type and identify establishments with recurring violations.
- Provide insights on inspection outcomes by location, risk category, and time period across both Chicago and Dallas.
- Track inspection trends over time and generate metrics to identify seasonal patterns or improvements.
- Identify high-risk vs. low-risk establishments, and analyze violation patterns by inspector, facility type, and geographical area.
- Generate reports on the most common violations, including breakdowns by severity and establishment type.
- Analyze violation data comparatively between Chicago and Dallas, including identifying regional differences in compliance rates.
- Enable geographical analysis to identify "hot spots" of food safety concerns within each city.
- Support resource allocation decisions by highlighting areas and establishment types with the highest failure rates.
- Provide transparency to the public through accessible visualizations of food safety compliance in their communities.
- Enable drill-down analysis from summary metrics to specific inspection details, including individual violation records.

4. Technologies and Tools Used

This project employs a robust suite of modern data technologies and tools to process, analyze, and visualize food inspection data from Chicago and Dallas:

Data Modeling and Design

- **ER Studio**: Used for comprehensive dimensional modeling, including the design of the star schema for the gold layer. This tool facilitated the creation of dimension and fact table designs, supporting the documentation of entity relationships, primary and foreign key constraints, and data lineage. ER Studio was instrumental in developing the unified data model that accommodates both Chicago and Dallas inspection systems despite their structural differences.

Data Integration and ETL

- **Azure Data Factory (ADF)**: Implemented as the orchestration layer for all data movement and transformation processes. ADF pipelines manage the scheduled extraction of new inspection data, coordinate the execution of transformation workflows, and handle the loading of processed data into the Snowflake data warehouse. ADF's monitoring capabilities provide visibility into data pipeline health and execution status, ensuring reliable data processing.

Data Transformation and Cleaning

- **Alteryx**: Served as the primary tool for in-depth data profiling, cleansing, and transformation. Separate workflows were developed for Chicago and Dallas data to address their unique quality issues and structural characteristics. Alteryx's visual workflow interface enabled complex data manipulation including:
    - Automated data type detection and conversion
    - Multi-stage text parsing for violation extraction
    - NULL value standardization and handling
    - Regular expression-based text cleaning
    - Conditional transformations for risk categorization
    - Geographic coordinate standardization
    - Data validation and quality profiling

Data Storage and Management

- **Snowflake**: Functions as the cloud-based data warehouse for the entire solution, providing a scalable, high-performance environment for both the silver and gold layers of the medallion architecture. Snowflake's multi-cluster architecture allows for concurrent analytical queries without performance degradation. The platform's native support for semi-structured data facilitates the storage of complex violation information while maintaining query efficiency.

Data Visualization and Reporting

- **Power BI**: Selected as the exclusive visualization tool for this project, Power BI connects directly to the Snowflake data warehouse to create interactive dashboards and reports that fulfill the business objectives. Power BI enables:
    - Interactive maps showing geographical distribution of inspection results using the built-in mapping capabilities
    - Time-series visualizations of compliance trends with dynamic filtering
    - Comparative analysis between Chicago and Dallas through multi-page reports
    - Drill-down capabilities from summary metrics to detailed inspection records
    - Role-based dashboards for different stakeholder groups
    - DAX measures for calculating key performance indicators
    - Custom visuals for specialized analytical needs
    - Automated refresh schedules to incorporate new inspection data

This integrated technology stack ensures efficient data processing from source to insight, with each tool optimized for its specific role in the data pipeline, creating a maintainable and scalable solution for food inspection analytics.

5. Data Cleaning

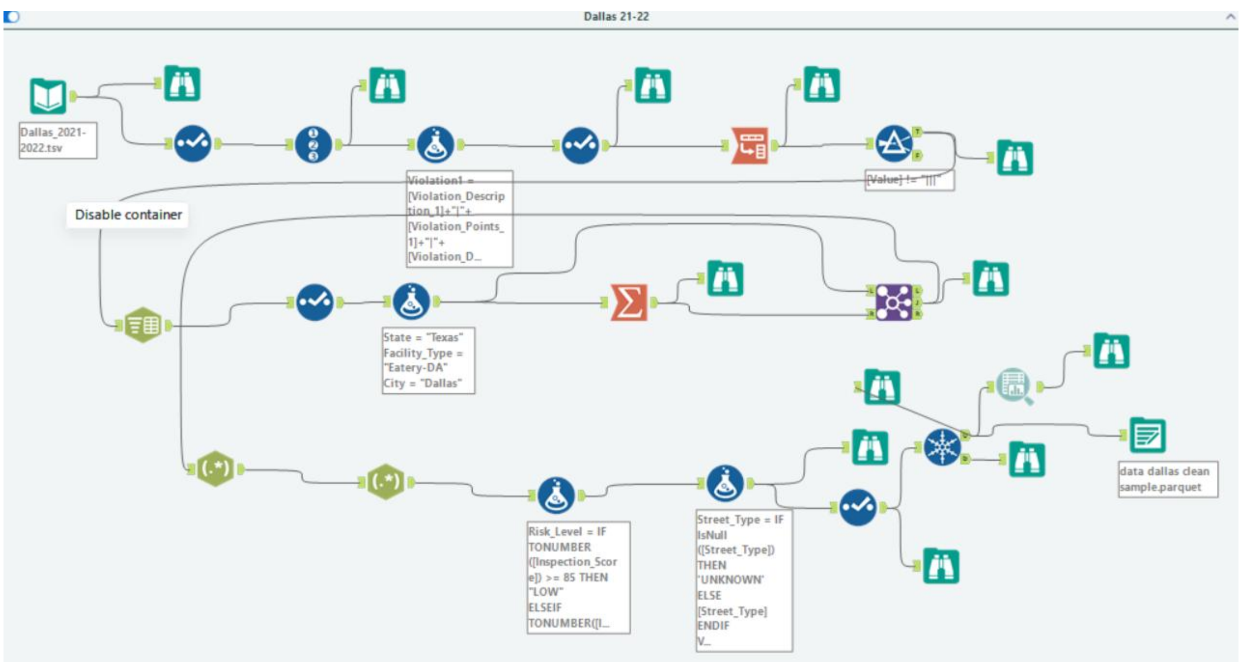
5.1 Data Profiling Approach

The workflow for both Dallas and Chicago datasets incorporates systematic data profiling techniques to:

- Identify key metrics including unique values, nulls, and potential outliers
- Detect and resolve data anomalies through pattern matching and standardization
- Apply formula-based data cleansing to ensure consistency
- Validate data types and formats for each field
- Handle multilingual content and special characters

5.2 Dallas Food Inspection Data

5.2.1 Detailed Workflow Sequence



Step 1: Initial Data Selection and Profiling (Select Tool)

The workflow begins with selecting and profiling the raw data:

- Inputs the Dallas_2021-2022.tsv file
- Evaluates all fields, which are initially imported as V_String with size 254
- Examine inspection fields, location components, and the 25 sets of violation fields

- Identifies key metrics including null percentages and data patterns
- Provides a foundation for subsequent transformations

Step 2: Record ID Generation (Record ID Tool)

A unique identifier is created for each inspection record:

- Generates Inspection_ID with a starting value of 1
- Configures as String type with size 7
- Positions as the first column for tracking through the workflow
- Establishes a consistent reference point for all records

Step 3: Violation Consolidation (Formula Tool)

Multiple violation fields are consolidated using a formula:

- Creates 25 new fields (Violation1 through Violation25)
- Combines related violation data with pipe delimiters:

[Violation_Description_1]+"|"+[Violation_Points_1]+"|"+[Violation_Detail_1]+"|"+[Violation_Memo_1]

- Sets output as V_WString with size 1073741823
- Streamlines the 100+ violation fields into a more manageable structure

Step 4: Field Selection (Select Tool)

Relevant fields are selected for further processing:

- Maintains the consolidated violation fields
- Retains core inspection data fields
- Excludes redundant or low-value fields
- Prepares data for subsequent transformations

Step 5: Transpose Operation (Transpose Tool)

The data structure is optimized for dimensional modeling:

- Restructures key inspection fields
- Transforms fields including Inspection_ID, Restaurant_Name, Inspection_Type
- Prepares data for more efficient processing

Step 6: Data Filtering (Filter Tool)

Records with empty violation data are filtered out:

- Applies condition: [Value] != "|||"
- Removes records without meaningful violation information
- Improves data quality for downstream analysis
- Similar to addressing empty string issues mentioned in the reference

Step 7: Text to Columns Transformation (Text to Columns Tool)

Consolidated violation fields are split into components:

- Configures to split the "Value" field using pipe (|) delimiter
- Creates 3 columns from the split
- Sets "Leave extra in last column" for remaining data
- Names the output root as "Violations"
- Transforms concatenated violation data back into analytical components

Step 8: Enhanced Field Selection and Type Conversion (Select Tool)

Field selection and type conversion are performed:

- Converts Inspection_Score from string to Int64 (size 8)
- Converts Zip_Code to Int64 (size 8)
- Includes derived violation fields:
    - Violations1 (V_WString) - Violation description
    - Violations2 (Int64) - Renamed to Violation_Point
    - Violations3 (V_WString) - Renamed to Violation_Other
- Adds missing coordinate fields as Double type
- Prepares the data for standardization and analytics

Step 9: Location and Facility Standardization (Formula Tool)

Standardized location fields are created:

- Sets State = "Texas" (V_WString, size 1073741823)
- Sets Facility_Type = "Eatery-DA" (V_WString, size 1073741823)

- Sets City = "Dallas" (V_WString, size 1073741823)
- Ensures consistency in location data across all records
- Similar to standardization approaches in the reference document

Step 10: Data Summarization (Summarize Tool)

Data is summarized by inspection:

- Groups by Inspection_ID
- Applies Sum action to Violation_Point, creating Sum_Violation_Point
- Aggregates violation metrics at the inspection level
- Creates a consolidated view for violation analysis

Step 11: Data Joining (Join Tool)

Datasets are joined:

- Connects by specific fields using Inspection_ID as the key
- Includes all fields from the left input
- Selectively includes fields from the right input
- Handles unknown fields with NA designations
- Consolidates data streams from different workflow branches

Step 12: Regular Expression Processing (RegEx Tool)

RegEx is applied to extract violation categories:

- Parses the Violations1 field
- Uses regular expression \^(\d+)\$\(.*\) with case insensitivity
- Extracts violation category IDs and descriptions
- Creates two output columns:
    - Violations_Category_Id (Int64, size 8) - numeric violation ID
    - Violations_Category (V_WString, size 1073741823) - violation description
- Separates structured violation data from raw text
- Aligns with the RegEx techniques mentioned in the reference document

Step 13: Geographic Coordinate Extraction (RegEx Tool)

Geographic coordinates are extracted:

- Processes the Lat_Long_Loc field
- Uses regular expression \((-+)?[0-9]\.[0-9]+\),?\s*\((-+)?[0-9]\.[0-9]+\)
- Extracts latitude and longitude values
- Outputs coordinates as V_String with size 10000000
- Prepares geospatial data for mapping and analysis
- Similar to RegEx techniques in the reference document

Step 14: Risk Level Calculation (Formula Tool)

Risk classification system is implemented:

- Creates a Risk_Level field based on inspection scores:

  IF TONUMBER([Inspection_Score]) >= 85 THEN "LOW"

  ELSEIF TONUMBER([Inspection_Score])>= 70 AND TONUMBER([Inspection_Score]) <= 84 THEN "MEDIUM"

  ELSE "HIGH"

ENDIF

- Sets output as V_WString with size 1073741823
- Standardizes risk categorization for analysis
- Similar to formula implementations mentioned in the reference

Step 15: Data Cleansing and Standardization (Formula Tool)

Multiple data cleansing operations are performed:

- Handles missing Street_Type:

  IF IsNull([Street_Type]) THEN 'UNKNOWN' ELSE [Street_Type] ENDIF

- Cleans Violations1: REGEX_Replace([Violations1], '\r?\n|\r', ' ')
- Cleans Lat_Long_Loc: REGEX_Replace([Lat_Long_Loc], '\r?\n|\r', ' ')
- Handles missing Violations_Category_Id: IF IsNull([Violations_Category_Id]) THEN -9999 ELSE [Violations_Category_Id] ENDIF

- Handles missing Violations_Category: IF IsNull([Violations_Category]) THEN 'UNKNOWN' ELSE [Violations_Category] ENDIF
- Handles missing coordinates with default values (-9999)
- Creates standardized Inspection_Result categories
- Sets AKA_Name to Restaurant_Name
- Trims Violation_Other using Trim() function
- These approaches align with the "Fixing Anomalies" and "Formula Implementation" techniques in the reference document

Step 16: Final Field Selection and Renaming (Select Tool)

The final field selection and renaming is performed:

- Selects key fields for the output dataset
- Renames fields to match the dimensional model:
    - Restaurant_Name to DBA_Name
    - Zip_Code to Zip
    - Lat_Long_Loc to Address
    - Violation_Other to Violation_Category
    - Violations_Category_Id to Violation_Category_ID
    - Violations_Category to Violation_Comments
- Sets appropriate data types for all fields
- Prepares the dataset for dimensional modeling

Step 17: Duplicate Removal (Unique Tool)

Duplicate records are removed:

- Configures unique record selection based on key fields
- Ensures data integrity by eliminating redundant records
- Maintains a clean dataset for analytical purposes
- Aligns with the "Unique Values and Duplicates" checks mentioned in the reference

Step 18: Data Profiling (Basic Data Profile Tool)

As visible in image, a comprehensive data profile is generated:

- Sets limit for exact count to 10000

- Sets size limit to return all unique values to 1000 characters
- Provides statistical summaries and value distributions
- Identifies potential data quality issues
- Documents on the characteristics of the transformed dataset
- Similar to the "Identifying and Checking Key Metrics" approach in the reference

Step 19: Output to Parquet (Output Data Tool)

The final dataset is output to Parquet format:

- Specifies output location: C:\Users\v2lbo\OneDrive\Desktop\DALLAS 21-22.parquet
- Sets Parquet as the file format (.parquet/.pqt)
- Completes the transformation process
- Creates the finalized dataset for the silver layer of the medallion architecture

5.2.2 Data Quality Issues and Resolutions

Missing Value Handling

- Empty strings and nulls were replaced with standardized values:
    - Missing Street_Type values replaced with "UNKNOWN"
    - Missing category identifiers replaced with -9999
    - Missing geographic coordinates handled with default values
    - This approach aligns with the "Nulls and Missing Data" handling in the reference

Text Standardization

- Regular expressions were used to clean text fields:
    - Removal of newlines and special characters
    - Trimming of whitespace
    - Standardization of text values
    - These techniques align with the "White Space Issues" and "REGEX_Replace" formulas in the reference

Data Type Conversion

- String fields converted to appropriate types:
    - Numeric scores converted to integers

    - Dates properly formatted
    - Violation points converted to numeric types
    - These conversions support data integrity and analysis

Character Handling

- Special character handling in text fields:
    - Newlines and carriage returns removed
    - Standardized text formatting
    - Consistent approach to special characters
    - Similar to the "Character Encoding Verification" mentioned in the reference

5.2.3 Data Transformation Techniques

Regular Expression Processing

- Multiple RegEx patterns for data extraction and cleaning:
    - Coordinate extraction from composite fields
    - Category ID and description separation
    - Character replacement for standardization
    - These align with the RegEx approaches in the reference

Conditional Logic

- IF-THEN-ELSE formulas for data standardization:
    - Risk level categorization
    - Null value replacement
    - Standard value assignment
    - Similar to the conditional logic examples in the reference

Text Processing

- Text manipulation for consistency:
    - Trimming whitespace
    - Replacing newlines
    - Standardizing formats

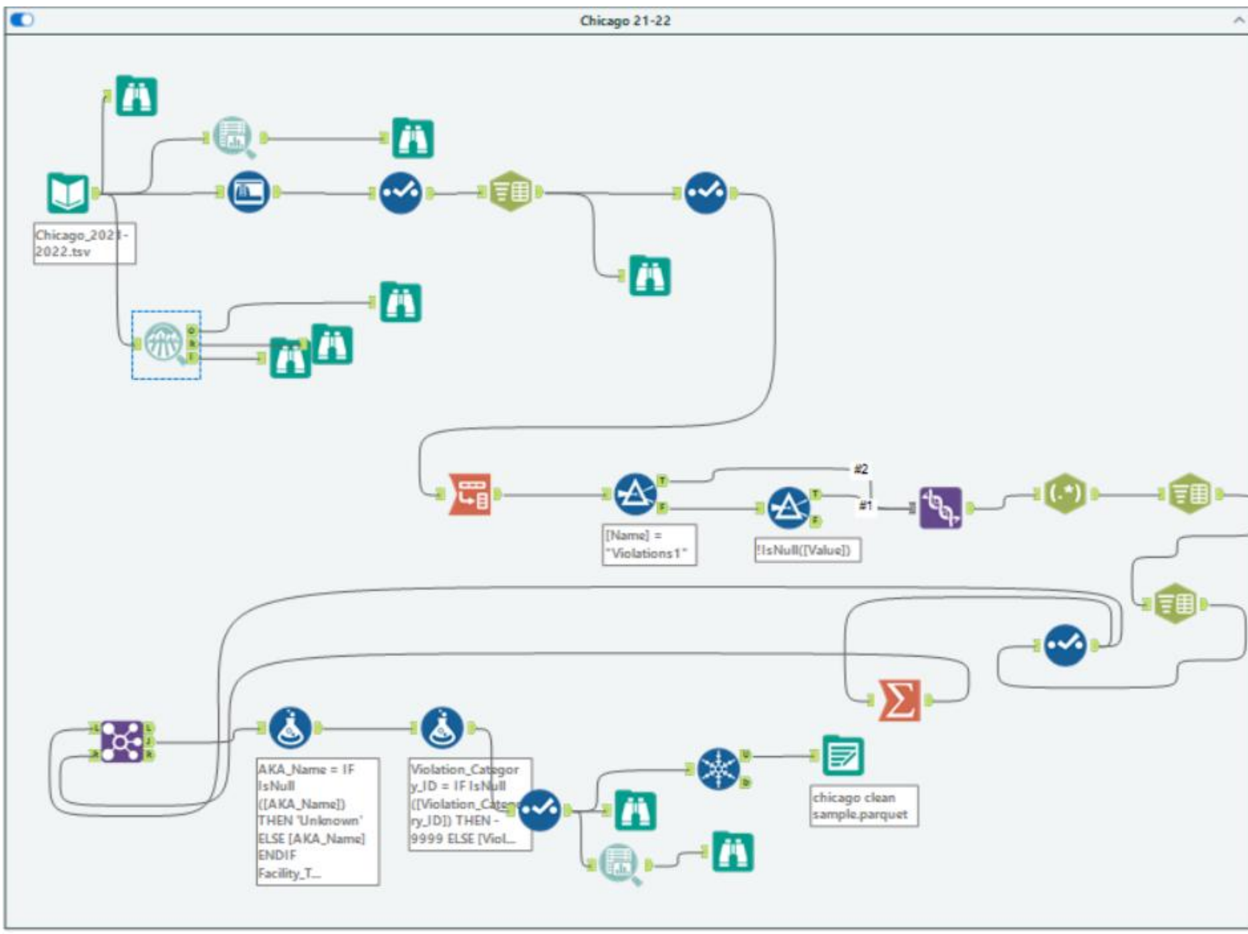
- These align with the "Trim" and text processing in the reference

5.2.4 Other Dallas Files

Similarly, the same workflow process and techniques were applied to other Dallas food inspection data files, ensuring consistent data quality and standardization across all datasets.

5.3 Chicago Food Inspection Data

5.3.1 Detailed Workflow Sequence



Step 1: Initial Data Profiling (Basic Data Profile Tool)

The workflow begins with a comprehensive data profiling operation to understand the structure and quality of the source data:

- Sets limit for exact count to 10000
- Sets size limit for unique values to 1000 characters

- Analyzes the Chicago_2021-2022.tsv source file
- Provides statistical summaries of all fields
- Identifies data quality issues for targeted cleansing

Step 2: Automated Data Type Detection (Auto Field Tool)

The Auto Field tool intelligently detects and converts appropriate field types:

- Automatically identifies optimal data types for all fields
- Processes key fields including:
    - Inspection ID
    - DBA Name
    - AKA Name
    - License #
    - Facility Type
    - Risk
    - Address fields (Address, City, State, Zip)
    - Inspection Date and Type
    - Results
    - Violations
    - Geographic coordinates
- Improve performance by optimizing data types

Step 3: Field Selection and Type Conversion (Select Tool)

A Select tool provides precise control over field types and metadata:

- Converts Inspection_ID to Int32 (size 4)
- Maintains DBA_Name and AKA_Name as V_String (size 57)
- Converts License # to Double (size 8)
- Maintains Facility_Type as V_String (size 38)
- Maintains Risk as String (size 15)
- Maintains Address as V_String (size 51)
- Maintains City as V_String (size 14) and State as String (size 2)
- Converts Zip to Double (size 8)
- Converts Inspection_Date to Date (size 10)

- Maintains Inspection_Type as V_String (size 38)
- Maintains Results as V_String (size 20)
- Maintains Violations as V_String (size 8703)
- Maintains Latitude and Longitude as String (size 18)
- Maintains Location as String (size 40)

Step 4: Violation Field Parsing (Text to Columns Tool)

The Text to Columns tool divides the complex violation text field:

- Splits the "Violations" field using pipe (|) delimiter
- Creates 35 separate columns to capture all components
- Configures "Leave extra in last column" for any remaining data
- Names the output root as "Violations"
- Transforms monolithic violation text into structured analytical components

Step 5: Enhanced Field Selection (Select Tool)

A second Select tool refines the field selection after parsing:

- Maintains core inspection fields
- Includes all newly split violation fields (Violations1-35)
- Converts Inspection_Date to V_WString (size 214)
- Maintains the parsed Violations fields as V_String (size 8703)
- Positions fields for subsequent transformations

Step 6: Data Restructuring (Transpose Tool)

The Transpose tool optimizes the data structure for dimensional modeling:

- Selects key columns to transpose:
    - Inspection_ID
    - DBA_Name
    - AKA_Name
    - Facility_Type
    - Risk

    - Address
    - City
- Configures warning for missing columns
- Ensures all selected fields are included in output
- Creates a more analysis-friendly data structure

Step 7: First Filter Operation (Filter Tool)

The first Filter tool targets specific violation data:

- Uses custom filter: [Name] = "Violations1"
- Isolates primary violation records
- Focuses processing on the most significant violation information
- Prepares data for downstream violation analysis

Step 8: Non-Null Value Filter (Filter Tool)

A second Filter tool removes records without violation data:

- Applies basic filter: "Value is not null"
- Eliminates records without meaningful violation information
- Improves data quality by ensuring completeness
- Streamlines subsequent processing

Step 9: Data Union (Union Tool)

The Union tool combines multiple data streams:

- Configures union by field names (automatic matching)
- Sets "Warning - Continue Processing Records" for when fields differ
- Outputs all fields from input streams
- Creates a unified dataset from multiple processing branches

Step 10: Comments Extraction (RegEx Tool)

The RegEx tool processes violation comments:

- Parses the "Value" field
- Uses regular expression pattern "- Comments:" with case insensitivity

- Output method set to "Replace"
- Replacement text set to "%"
- Copies unmatched text to output
- Standardizes the format of violation comments

Step 11: Further Parsing of Violation Data (Text to Columns Tool)

Second Text to Columns tool is used:

- Configures to split the "Value" field using "%" delimiter
- Creates 2 columns from the split
- Sets "Leave extra in last column"
- Names the output root as "Value"
- Further refines the violation information structure

Step 12: Violation Detail Extraction (Text to Columns Tool)

Third Text to Columns tool processes violation details:

- Splits "Value1" field using "." (period) delimiter
- Creates 2 columns from the split
- Sets "Leave extra in last column"
- Names output root as "ViolationID"
- Isolates violation category identifiers from descriptions

Step 13: Final Field Selection (Select Tool)

As visible in image, the Select tool prepares fields for the dimensional model:

- Maintains core fields (Inspection_ID, DBA_Name, AKA_Name)
- Excludes missing or unnecessary fields
- Renames violation fields:
    - ViolationID1 to Violation_Category_ID
    - ViolationID2 to Violation_Category
    - Value2 to Violation_Comments
- Converts data types appropriately

Step 14: Final Data Joining (Join Tool)

Join tool combines processed data streams:

- Joins by specific fields using Inspection_ID as the key
- Includes all fields from the left input
- Selectively includes fields from the right input
- Consolidates violation information with core inspection data

Step 15: Missing Value Handling (Formula Tool)

Formula tool handles missing values:

- Processes multiple fields with NULL handling logic:
  - AKA_Name: IF IsNull([AKA_Name]) THEN 'Unknown' ELSE [AKA_Name] ENDIF
  - Facility_Type: IF IsNull([Facility_Type]) THEN 'Unknown' ELSE [Facility_Type] ENDIF
  - State: IF IsNull([State]) THEN 'Unknown' ELSE [State] ENDIF
  - Risk_Category: IF IsNull([Risk]) THEN 'Unknown' ELSE [Risk] ENDIF
  - City: IF IsNull([City]) THEN 'Unknown' ELSE [City] ENDIF
  - Latitude: IF IsNull([Latitude]) THEN -9999 ELSE [Latitude] ENDIF
  - Longitude: IF IsNull([Longitude]) THEN -9999 ELSE [Longitude] ENDIF
  - Location: IF IsNull([Location]) THEN -9999 ELSE [Location] ENDIF
- Ensures no null values remain in critical fields

Step 16: Advanced Data Transformation (Formula Tool)

Formula tool applies complex transformations:

- Handles violation category data:

  IF IsNull([Violation_Category_ID]) THEN -9999 ELSE [Violation_Category_ID] ENDIF

- Creates standardized Risk information:

IF CONTAINS([Risk], "Risk") THEN REGEX_Replace([Risk], "Risk (\d+).*", "$1") ELSE "NA" ENDIF

- Creates Risk level categories:

  IF CONTAINS([Risk], "(") THEN REGEX_Replace([Risk], ".*\((.*)\)", "$1") ELSE "None" ENDIF

- Formats Location field:

  "(" + ToString([Latitude]) + ", " + ToString([Longitude]) + ")"

- Handles missing Zip values:

  IF IsNull([Zip]) THEN 0 ELSE [Zip] ENDIF

- Processes Violation fields:
  - Trims whitespace from Violation_Category_ID and Violation_Category
  - Provides standardized handling for empty values:

  IF IsEmpty([Violation_Category_ID]) THEN 'Unknown' ELSE [Violation_Category_ID] ENDIF

  IF IsEmpty([Violation_Category]) THEN 'Unknown' ELSE [Violation_Category] ENDIF

  IF IsEmpty([Violation_Comments]) THEN 'Unknown' ELSE [Violation_Comments] ENDIF

Step 17: Final Field Selection and Renaming (Select Tool)

Final Select tool prepares the output dataset:

- Selects key fields for the final output
- Renames Results to Inspection_Result
- Renames Count to Violation_Point
- Ensures appropriate data types for all fields
- Positions fields in logical order for dimension modeling

Step 18: Final Data Profiling (Basic Data Profile Tool)

Final profiling operation validates the processed data:

- Sets limit for exact count to 10000

- Sets size limit for unique values to 1000 characters
- Provides statistical summaries of transformed data
- Verifies data quality improvements
- Confirms readiness for dimensional model

Step 19: Duplicate Removal (Unique Tool)

The Unique tool eliminates redundant records:

- Configures unique record selection based on key fields:
    - Inspection_ID
    - DBA_Name
    - AKA_Name
    - Facility_Type
    - Address
    - City
    - State
    - Zip
- Ensures data integrity by removing duplicates

Step 20: Output to Parquet (Output Data Tool)

Dataset is output to Parquet format:

- Specifies output location: C:\Users\v2lbo\OneDrive\Desktop\CHICAGO 21-22.parquet
- Sets Parquet as the file format (.parquet/.pqt)
- Completes the transformation process
- Creates the finalized dataset for the silver layer

5.3.2 Data Quality Issues and Resolutions

Text Parsing and Extraction

The Chicago workflow employs sophisticated multi-stage text parsing:

- Initial Text to Columns splitting of the monolithic Violations field
- Secondary splitting using RegEx to extract comments
- Tertiary parsing to isolate violation categories and IDs
- Standardized formatting through formula-based transformations

Missing Value Treatment

The workflow implements a comprehensive approach to missing values:

- Field-specific default values ('Unknown' for categorical fields)
- Numeric placeholders (-9999) for missing coordinates
- Empty string handling with IsEmpty() checks
- Distinct treatment based on field purpose and usage

Data Type Management

Strategic data type handling ensures optimal performance:

- Auto Field detection for initial optimization
- Manual overrides for specific business requirements
- Appropriate sizing to balance storage efficiency with data preservation
- Consistence with dimensional model expectations

Multi-stage Processing

The workflow uses a progressive refinement approach:

- Initial profiling to understand data characteristics
- Structural transformations to normalize data format
- Content standardization through formula tools
- Final refinements and validations before output

5.3.3 Data Transformation Techniques

Chicago-Specific Data Characteristics

Chicago's violation data has unique characteristics:

- Violations stored in a single field with complex formatting
- Multiple delimiter patterns requiring multi-stage parsing
- Comment sections separated by specific text patterns
- Violation identifiers embedded within text

Risk Classification System

Chicago uses a structured risk classification system:

- Risk levels encoded in text (e.g., "Risk 1")
- Additional risk information in parentheses
- Requires RegEx extraction to standardize
- Converted to numeric and categorical values

Location Data Handling

Chicago's location information includes:

- Separate Latitude and Longitude fields
- Combined Location field
- Special handling for missing coordinate data
- Standardized output format for geo-analysis

5.3.4 Other Chicago Files

Similarly, the same workflow process and techniques were applied to other Chicago inspection data files, ensuring consistent data quality and standardization across all datasets.

6. Data Modeling and Design

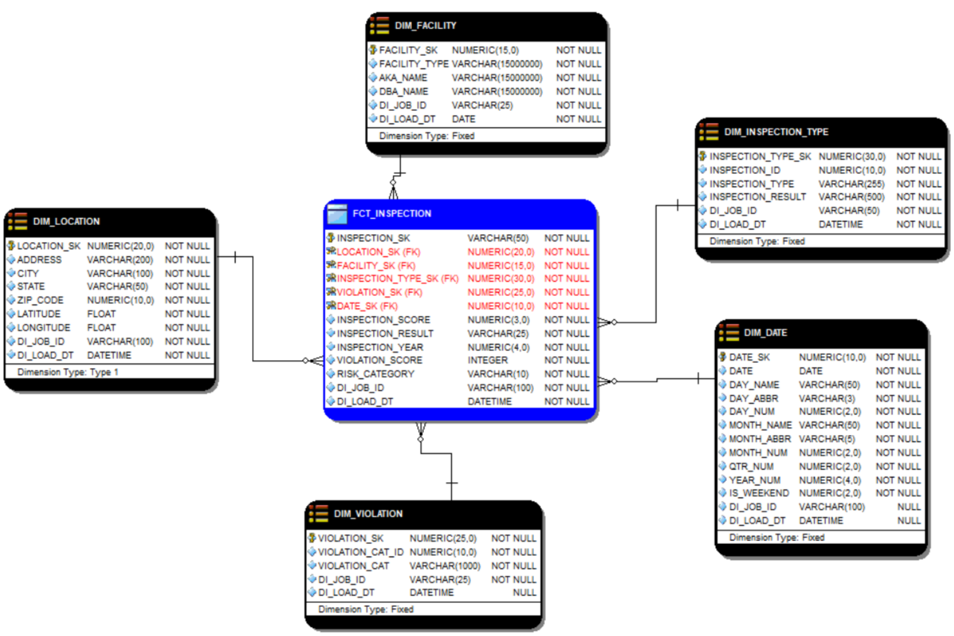
6.1 Data Modeling Process

The data modeling process for the Food Establishment Inspections project involved designing a schema that could effectively support the business requirements while enabling efficient analysis of food inspection data from Chicago and Dallas. After careful consideration of analytical needs and data structures, a Star Schema was selected as the optimal design pattern, balancing simplicity, query performance, and analytical capability.

The modeling process entailed several key stages:

- Analysis of source data structures from both Chicago and Dallas systems
- Identification of common dimensions and metrics across both datasets
- Design of a unified dimensional model that could accommodate both cities' data
- Implementation of appropriate surrogate keys, constraints, and relationships
- Validation of the model against business requirements and analytical needs

6.2 Dimensional Model



At the core of our data warehouse architecture is a thoughtfully designed star schema with **FCT_INSPECTION** as the central fact table, surrounded by five carefully constructed dimension tables: **DIM_FACILITY**, **DIM_LOCATION**, **DIM_INSPECTION_TYPE**, **DIM_DATE**, and

DIM_VIOLATION. This architecture enables high-performance analytical queries while maintaining data consistency across both cities' inspection systems.

Fact Table Structure

FCT_INSPECTION - The central entity capturing all inspection events:

- **Primary Key**: INSPECTION_SK (VARCHAR(50))
- **Foreign Keys**:
  - LOCATION_SK (NUMERIC(20,0)) - Links to DIM_LOCATION
  - FACILITY_SK (NUMERIC(15,0)) - Links to DIM_FACILITY
  - INSPECTION_TYPE_SK (NUMERIC(30,0)) - Links to DIM_INSPECTION_TYPE
  - VIOLATION_SK (NUMERIC(25,0)) - Links to DIM_VIOLATION
  - DATE_SK (NUMERIC(10,0)) - Links to DIM_DATE
- **Measures**:
  - INSPECTION_SCORE (NUMERIC(3,0)) - Numerical rating of inspection
  - INSPECTION_RESULT (VARCHAR(25)) - Outcome designation (pass, conditional, fail)
  - INSPECTION_YEAR (NUMERIC(4,0)) - Year when inspection occurred
  - VIOLATION_SCORE (INTEGER) - Quantified violation severity
  - RISK_CATEGORY (VARCHAR(10)) - Risk classification of the establishment
- **Administrative Fields**:
  - DI_JOB_ID (VARCHAR(100)) - Data integration job identifier
  - DI_LOAD_DT (DATETIME) - Data load timestamp

The fact table consolidates inspection data from both Chicago and Dallas, harmonizing different scoring systems and result categorizations into a consistent framework while preserving the unique attributes of each city's inspection approach.

Dimension Tables

The star schema includes five key dimension tables, each providing important contextual information for food inspection analysis:

1. DIM_FACILITY

Represents the food establishments being inspected:

- **Primary Key**: FACILITY_SK (NUMERIC(15,0)) - NOT NULL
- **Descriptive Attributes**:
  - FACILITY_TYPE (VARCHAR(15000000)) - NOT NULL - Categorization of establishment (restaurant, grocery, bakery, etc.)
  - AKA_NAME (VARCHAR(15000000)) - NOT NULL - Alternative business name
  - DBA_NAME (VARCHAR(15000000)) - NOT NULL - Primary business name (Doing Business As)
- **Administrative Fields**:
  - DI_JOB_ID (VARCHAR(25)) - NOT NULL - ETL job identifier
  - DI_LOAD_DT (DATE) - NOT NULL - Data load date
- **Dimension Type**: Fixed dimension

This dimension consolidates establishment information from both cities, standardizing facility types and maintaining both official (DBA) and alternative (AKA) business names.

2. DIM_LOCATION

Contains geographical and address details:

- **Primary Key**: LOCATION_SK (NUMERIC(20,0)) - NOT NULL
- **Descriptive Attributes**:
  - ADDRESS (VARCHAR(200)) - NOT NULL - Street address
  - CITY (VARCHAR(100)) - NOT NULL - City name
  - STATE (VARCHAR(50)) - NOT NULL - State code
  - ZIP_CODE (NUMERIC(10,0)) - NOT NULL - Postal code
  - LATITUDE (FLOAT) - NOT NULL - Geographic coordinate

    - LONGITUDE (FLOAT) - NOT NULL - Geographic coordinate
- **Administrative Fields**:
    - DI_JOB_ID (VARCHAR(100)) - NOT NULL - ETL job identifier
    - DI_LOAD_DT (DATETIME) - NOT NULL - Data load timestamp
- **Dimension Type**: Type 1 dimension (maintains current values only)

The location dimension handles the differing address formats between Chicago and Dallas, creating a standardized structure that supports both detailed address analysis and geospatial visualization.

3. DIM_INSPECTION_TYPE

Classifies the types of inspections performed:

- **Primary Key**: INSPECTION_TYPE_SK (NUMERIC(30,0)) - NOT NULL
- **Business Keys**:
    - INSPECTION_ID (NUMERIC(10,0)) - NOT NULL - Source system identifier
- **Descriptive Attributes**:
    - INSPECTION_TYPE (VARCHAR(255)) - NOT NULL - Type of inspection performed (routine, complaint, follow-up)
    - INSPECTION_RESULT (VARCHAR(200)) - NOT NULL - Standardized result values
- **Administrative Fields**:
    - DI_JOB_ID (VARCHAR(50)) - NOT NULL - ETL job identifier
    - DI_LOAD_DT (DATETIME) - NOT NULL - Data load timestamp
- **Dimension Type**: Fixed dimension

This dimension normalizes the different inspection classification systems used by Chicago and Dallas into a consistent framework while preserving the original categorizations for city-specific analysis.

4. DIM_DATE

Time dimension with rich calendar attributes:

- **Primary Key**: DATE_SK (NUMERIC(10,0)) - NOT NULL
- **Core Date Field**:

- DATE (DATE) - NOT NULL - Calendar date

- **Date Hierarchy Attributes**:
  - DAY_NAME (VARCHAR(50)) - NOT NULL - Full day name
  - DAY_ABBR (VARCHAR(3)) - NOT NULL - Abbreviated day
  - DAY_NUM (NUMERIC(2,0)) - NOT NULL - Day of month
  - MONTH_NAME (VARCHAR(50)) - NOT NULL - Full month name
  - MONTH_ABBR (VARCHAR(5)) - NOT NULL - Abbreviated month
  - MONTH_NUM (NUMERIC(2,0)) - NOT NULL - Month number (1-12)
  - QTR_NUM (NUMERIC(2,0)) - NOT NULL - Quarter (1-4)
  - YEAR_NUM (NUMERIC(4,0)) - NOT NULL - Year value
  - IS_WEEKEND (NUMERIC(2,0)) - NOT NULL - Weekend indicator
- **Administrative Fields**:
  - DI_JOB_ID (VARCHAR(100)) - NULL - ETL job identifier
  - DI_LOAD_DT (DATETIME) - NULL - Data load timestamp
- **Dimension Type**: Fixed dimension

The date dimension provides powerful temporal analysis capabilities, enabling time-based patterns and trends to be identified across inspection data from both cities.

5. DIM_VIOLATION

Contains standardized violation categories:

- **Primary Key**: VIOLATION_SK (NUMERIC(25,0)) - NOT NULL
- **Business Keys**:
  - VIOLATION_CAT_ID (NUMERIC(10,0)) - NOT NULL - Violation category identifier
- **Descriptive Attributes**:
  - VIOLATION_CAT (VARCHAR(1000)) - NOT NULL - Violation category description
- **Administrative Fields**:
  - DI_JOB_ID (VARCHAR(25)) - NOT NULL - ETL job identifier

    - DI_LOAD_DT (DATETIME) - NULL - Data load timestamp

- **Dimension Type**: Fixed dimension

This dimension reconciles the different violation coding and categorization systems employed by Chicago and Dallas, creating a unified framework for violation analysis while maintaining the ability to analyze city-specific patterns.

6.3 Relationship Mapping & Data Integrity

The star schema implements a robust system of primary and foreign key relationships that ensure data integrity while enabling efficient query performance:

Primary-to-Foreign Key Relationships

- FCT_INSPECTION.LOCATION_SK → DIM_LOCATION.LOCATION_SK
- FCT_INSPECTION.FACILITY_SK → DIM_FACILITY.FACILITY_SK
- FCT_INSPECTION.INSPECTION_TYPE_SK→ DIM_INSPECTION_TYPE.INSPECTION_TYPE_SK
- FCT_INSPECTION.DATE_SK → DIM_DATE.DATE_SK
- FCT_INSPECTION.VIOLATION_SK → DIM_VIOLATION.VIOLATION_SK

Data Integrity Constraints

The model implements strict NOT NULL constraints on all dimension table primary keys and most descriptive attributes, ensuring data completeness and referential integrity. This approach guarantees that every inspection event is properly contextualized with complete dimensional information.

6.4 Analytical Capabilities

The dimensional model supports a comprehensive range of analytical capabilities that directly address the project's business requirements:

Multi-dimensional Analysis

- **Facility Performance Analysis**:
    - Analyze inspection results by facility type
    - Identify establishments with recurring violations
    - Compare compliance rates across different establishment categories
- **Geographical Analysis**:

    - Map inspection results by city, zip code, or custom geographical areas
    - Identify violation "hot spots" for targeted intervention
    - Compare regional compliance patterns between Chicago and Dallas
- **Temporal Trends**:
    - Track inspection outcomes over time (daily, monthly, quarterly, annually)
    - Identify seasonal patterns in violations or inspection results
    - Analyze weekend vs. weekday inspection differences
    - Measure year-over-year improvements in compliance rates
- **Violation Pattern Analysis**:
    - Identify most common violation categories
    - Correlate violations with establishment types or locations
    - Analyze severity distribution of violations
    - Compare violation patterns between Chicago and Dallas
- **Risk-Based Analysis**:
    - Profile establishments by risk category
    - Analyze inspection frequency relative to risk levels
    - Track risk level changes over time for specific establishments

Business Intelligence Support

Star schema design enables effective business intelligence capabilities through:

- Easy-to-understand dimensional structure that business users can navigate
- Consistent naming conventions that align with business terminology
- Optimized query performance for interactive dashboards
- Flexible filtering and drill-down capabilities across multiple dimensions

This dimensional model provides a comprehensive framework for analyzing food inspection data from both Chicago and Dallas, enabling stakeholders to gain valuable insights into food safety patterns, compliance trends, and areas requiring targeted intervention. The model successfully balances analytical flexibility, query performance, and data governance requirements in a cohesive design that harmonizes disparate municipal inspection systems into a unified analytical framework.

6.5 Business Queries

1. Recurring Violations by Establishment

Requirement: Identify establishments with patterns of recurring violations to target for enhanced monitoring or intervention.

```
SELECT
    f.DBA_NAME,
    f.FACILITY_TYPE,
    v.VIOLATION_CAT,
    COUNT(DISTINCT i.INSPECTION_SK) AS inspection_count,
    MIN(d.DATE) AS first_occurrence,
    MAX(d.DATE) AS most_recent_occurrence
FROM FCT_INSPECTION i
JOIN DIM_FACILITY f ON i.FACILITY_SK = f.FACILITY_SK
JOIN DIM_VIOLATION v ON i.VIOLATION_SK = v.VIOLATION_SK
JOIN DIM_DATE d ON i.DATE_SK = d.DATE_SK
GROUP BY f.DBA_NAME, f.FACILITY_TYPE, v.VIOLATION_CAT
HAVING COUNT(DISTINCT i.INSPECTION_SK) > 1
ORDER BY inspection_count DESC, f.DBA_NAME;
```

2. High-Risk Establishment Tracking

Requirement: Track establishments classified as high-risk that have recently failed inspections to prioritize follow-up activities.

```
SELECT
    f.DBA_NAME,
    f.FACILITY_TYPE,
    l.ADDRESS,
    l.CITY,
    COUNT(i.INSPECTION_SK) AS failed_inspection_count,
```

```
    MAX(d.DATE) AS most_recent_failed_inspection
FROM FCT_INSPECTION i
JOIN DIM_FACILITY f ON i.FACILITY_SK = f.FACILITY_SK
JOIN DIM_LOCATION l ON i.LOCATION_SK = l.LOCATION_SK
JOIN DIM_DATE d ON i.DATE_SK = d.DATE_SK
WHERE i.RISK_CATEGORY = 'HIGH'
  AND i.INSPECTION_RESULT = 'FAIL'
  AND d.YEAR_NUM = YEAR(CURRENT_DATE())
  AND d.QTR_NUM = FLOOR((MONTH(CURRENT_DATE())-1)/3)+1
GROUP BY f.DBA_NAME, f.FACILITY_TYPE, l.ADDRESS, l.CITY
ORDER BY failed_inspection_count DESC;
```

3. Violation Hot Spots

Requirement: Identify geographic areas with high concentrations of inspection failures to target for community-level interventions.

```
SELECT
    l.ZIP_CODE,
    l.CITY,
    COUNT(i.INSPECTION_SK) AS total_inspections,
    SUM(CASE WHEN i.INSPECTION_RESULT = 'FAIL' THEN 1 ELSE 0 END) AS failed_inspections,
    ROUND(100.0 * SUM(CASE WHEN i.INSPECTION_RESULT = 'FAIL' THEN 1 ELSE 0 END) / COUNT(i.INSPECTION_SK), 2) AS failure_rate
FROM FCT_INSPECTION i
JOIN DIM_LOCATION l ON i.LOCATION_SK = l.LOCATION_SK
JOIN DIM_DATE d ON i.DATE_SK = d.DATE_SK
WHERE d.YEAR_NUM >= YEAR(CURRENT_DATE()) - 1
GROUP BY l.ZIP_CODE, l.CITY
```

```
HAVING COUNT(i.INSPECTION_SK) > 10
ORDER BY failure_rate DESC;
```

4. Cross-City Comparison

Requirement: Compare inspection outcomes between Chicago and Dallas to identify regional differences in compliance rates.

```
WITH city_stats AS (
    SELECT
        l.CITY,
        f.FACILITY_TYPE,
        COUNT(i.INSPECTION_SK) AS total_inspections,
        SUM(CASE WHEN i.INSPECTION_RESULT = 'PASS' THEN 1 ELSE 0 END) AS passing_inspections
    FROM FCT_INSPECTION i
    JOIN DIM_FACILITY f ON i.FACILITY_SK = f.FACILITY_SK
    JOIN DIM_LOCATION l ON i.LOCATION_SK = l.LOCATION_SK
    WHERE l.CITY IN ('Chicago', 'Dallas')
    GROUP BY l.CITY, f.FACILITY_TYPE
    HAVING COUNT(i.INSPECTION_SK) > 5
)
SELECT
    cs.FACILITY_TYPE,
    MAX(CASE WHEN cs.CITY = 'Chicago' THEN cs.total_inspections ELSE 0 END) AS chicago_inspections,
    MAX(CASE WHEN cs.CITY = 'Chicago' THEN ROUND(100.0 * cs.passing_inspections / cs.total_inspections, 2) ELSE 0 END) AS chicago_pass_rate,
    MAX(CASE WHEN cs.CITY = 'Dallas' THEN cs.total_inspections ELSE 0 END) AS dallas_inspections,
```

```
    MAX(CASE WHEN cs.CITY = 'Dallas' THEN ROUND(100.0 * cs.passing_inspections / cs.total_inspections, 2) ELSE 0 END) AS dallas_pass_rate
FROM city_stats cs
GROUP BY cs.FACILITY_TYPE
HAVING chicago_inspections > 0 AND dallas_inspections > 0
ORDER BY ABS(chicago_pass_rate - dallas_pass_rate) DESC;
```

5. Seasonal Violation Patterns

Requirement: Analyze violation data by month to identify seasonal patterns that may inform inspection scheduling.

```
SELECT
    v.VIOLATION_CAT,
    d.MONTH_NAME,
    COUNT(*) AS violation_count,
    ROUND(COUNT(*)*100.0/COUNT(DISTINCT i.INSPECTION_SK), 2) AS violations_per_100_inspections
FROM FCT_INSPECTION i
JOIN DIM_VIOLATION v ON i.VIOLATION_SK = v.VIOLATION_SK
JOIN DIM_DATE d ON i.DATE_SK = d.DATE_SK
WHERE d.YEAR_NUM >= YEAR(CURRENT_DATE()) - 2
GROUP BY v.VIOLATION_CAT, d.MONTH_NAME, d.MONTH_NUM
ORDER BY v.VIOLATION_CAT, d.MONTH_NUM;
```

6. Establishment Improvement Tracking

Requirement: Track year-over-year improvement in compliance rates to identify successful interventions.

```
WITH yearly_performance AS (
    SELECT
```

```
        f.DBA_NAME,
        d.YEAR_NUM,
        COUNT(i.INSPECTION_SK) AS total_inspections,
        SUM(CASE WHEN i.INSPECTION_RESULT = 'PASS' THEN 1 ELSE 0 END) AS passing_inspections,
        ROUND(100.0 * SUM(CASE WHEN i.INSPECTION_RESULT = 'PASS' THEN 1 ELSE 0 END) / COUNT(i.INSPECTION_SK), 2) AS pass_rate
    FROM FCT_INSPECTION i
    JOIN DIM_FACILITY f ON i.FACILITY_SK = f.FACILITY_SK
    JOIN DIM_DATE d ON i.DATE_SK = d.DATE_SK
    WHERE d.YEAR_NUM >= YEAR(CURRENT_DATE()) - 3
    GROUP BY f.DBA_NAME, d.YEAR_NUM
    HAVING COUNT(i.INSPECTION_SK) >= 2
)
SELECT
    yp1.DBA_NAME,
    yp1.YEAR_NUM AS previous_year,
    yp1.pass_rate AS previous_pass_rate,
    yp2.YEAR_NUM AS current_year,
    yp2.pass_rate AS current_pass_rate,
    yp2.pass_rate - yp1.pass_rate AS pass_rate_change
FROM yearly_performance yp1
JOIN yearly_performance yp2 ON yp1.DBA_NAME = yp2.DBA_NAME AND yp1.YEAR_NUM = yp2.YEAR_NUM - 1
ORDER BY pass_rate_change DESC;
```

7. Common Violation Types

Requirement: Identify the most common violation categories to focus training and education efforts.

```
SELECT
    v.VIOLATION_CAT,
    COUNT(*) AS violation_count,
    COUNT(DISTINCT f.DBA_NAME) AS unique_establishments,
    ROUND(100.0 * COUNT(*) / (SELECT COUNT(*) FROM FCT_INSPECTION), 4) AS percent_of_all_violations
FROM FCT_INSPECTION i
JOIN DIM_VIOLATION v ON i.VIOLATION_SK = v.VIOLATION_SK
JOIN DIM_FACILITY f ON i.FACILITY_SK = f.FACILITY_SK
GROUP BY v.VIOLATION_CAT
ORDER BY violation_count DESC;
```

8. Violation Severity by Facility Type

Requirement: Analyze which types of facilities tend to have the most severe violations to inform risk-based inspection planning.

```
SELECT
    f.FACILITY_TYPE,
    AVG(i.VIOLATION_SCORE) AS avg_violation_score,
    APPROX_PERCENTILE(i.VIOLATION_SCORE, 0.5) AS median_violation_score,
    COUNT(*) AS inspection_count,
    ROUND(100.0 * SUM(CASE WHEN i.INSPECTION_RESULT = 'FAIL' THEN 1 ELSE 0 END) / COUNT(*), 2) AS failure_rate
FROM FCT_INSPECTION i
JOIN DIM_FACILITY f ON i.FACILITY_SK = f.FACILITY_SK
WHERE i.VIOLATION_SCORE IS NOT NULL
```

GROUP BY f.FACILITY_TYPE

HAVING COUNT(*) > 10

ORDER BY avg_violation_score DESC;

9. Inspection Type Effectiveness

Requirement: Evaluate which inspection types are most effective at detecting violations to optimize inspection protocols.

SELECT

it.INSPECTION_TYPE,

COUNT(i.INSPECTION_SK) AS total_inspections,

SUM(CASE WHEN i.VIOLATION_SCORE > 0 THEN 1 ELSE 0 END) AS inspections_with_violations,

ROUND(100.0 * SUM(CASE WHEN i.VIOLATION_SCORE > 0 THEN 1 ELSE 0 END) / COUNT(i.INSPECTION_SK), 2) AS violation_detection_rate

FROM FCT_INSPECTION i

JOIN DIM_INSPECTION_TYPE it ON i.INSPECTION_TYPE_SK = it.INSPECTION_TYPE_SK

GROUP BY it.INSPECTION_TYPE

HAVING total_inspections > 20

ORDER BY violation_detection_rate DESC;

10. Risk-Based Inspection Planning

Requirement: Generate a prioritized list of establishments for inspection based on risk level, past performance, and time since last inspection.

WITH last_inspection AS (

SELECT

f.FACILITY_SK,

MAX(d.DATE) AS last_inspection_date

FROM FCT_INSPECTION i

JOIN DIM_FACILITY f ON i.FACILITY_SK = f.FACILITY_SK

JOIN DIM_DATE d ON i.DATE_SK = d.DATE_SK

```sql
    GROUP BY f.FACILITY_SK
)
SELECT
    f.DBA_NAME,
    f.FACILITY_TYPE,
    l.ADDRESS,
    l.CITY,
    li.last_inspection_date,
    DATEDIFF(day, li.last_inspection_date, CURRENT_DATE()) AS days_since_last_inspection,
    i.RISK_CATEGORY,
    i.INSPECTION_RESULT
FROM last_inspection li
JOIN DIM_FACILITY f ON li.FACILITY_SK = f.FACILITY_SK
JOIN FCT_INSPECTION i ON i.FACILITY_SK = f.FACILITY_SK
JOIN DIM_DATE d ON i.DATE_SK = d.DATE_SK AND d.DATE = li.last_inspection_date
JOIN DIM_LOCATION l ON i.LOCATION_SK = l.LOCATION_SK
WHERE (i.RISK_CATEGORY = 'HIGH' AND DATEDIFF(day, li.last_inspection_date, CURRENT_DATE()) > 90) OR
      (i.RISK_CATEGORY = 'MEDIUM' AND DATEDIFF(day, li.last_inspection_date, CURRENT_DATE()) > 180) OR
      (i.RISK_CATEGORY = 'LOW' AND DATEDIFF(day, li.last_inspection_date, CURRENT_DATE()) > 365)
ORDER BY
    CASE i.RISK_CATEGORY
        WHEN 'HIGH' THEN 1
        WHEN 'MEDIUM' THEN 2
        ELSE 3
    END,
```

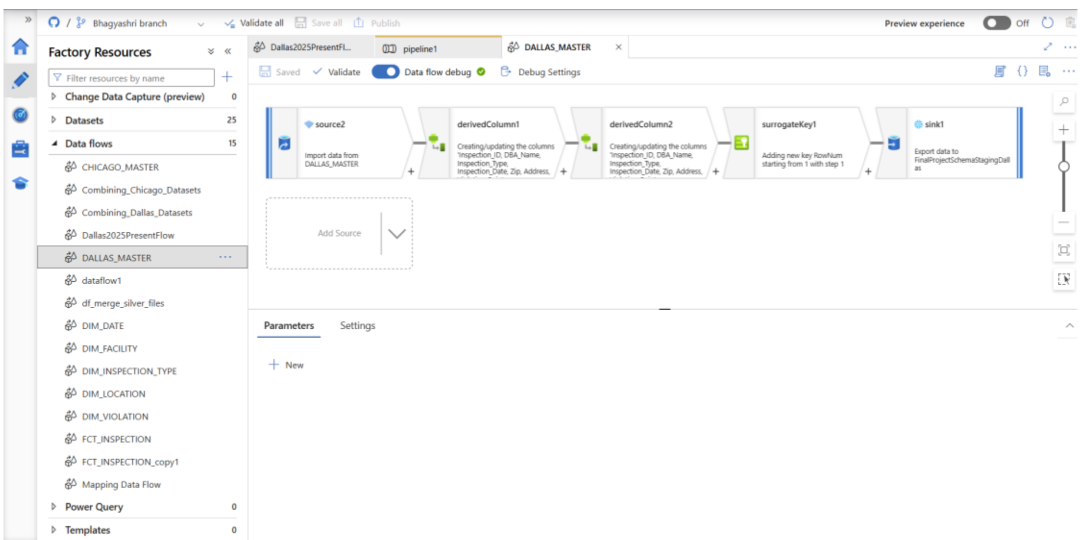
days_since_last_inspection DESC;

7. ETL Process Using Azure Data Factory (ADF)

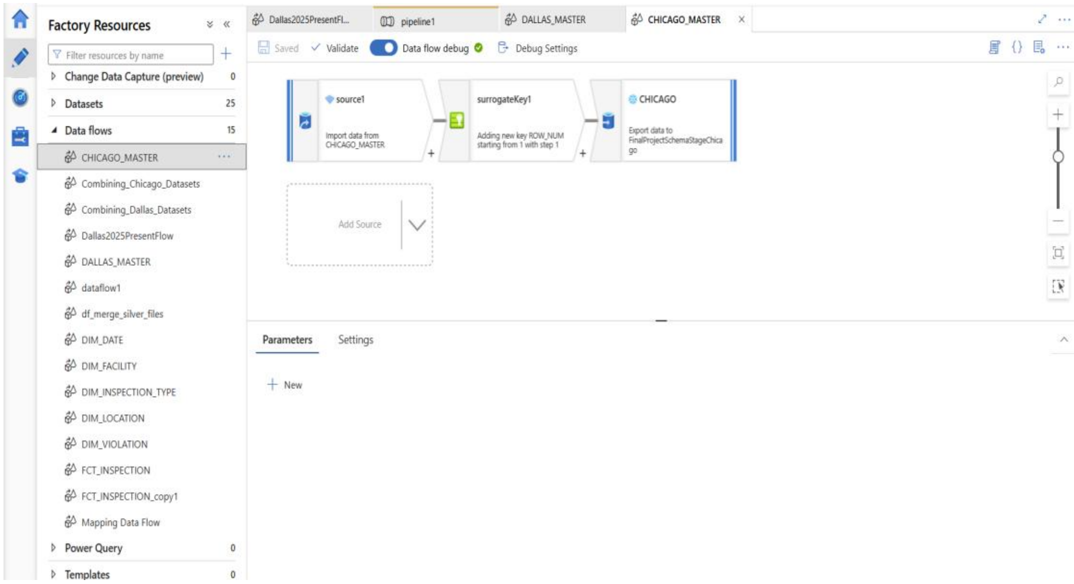
7.1 Staging Data Pipeline

The staging data pipeline in Azure Data Factory orchestrates the movement and initial processing of food inspection data from both Chicago and Dallas sources. As shown in the images, the pipeline consists of two main data flows:

1. **DALLAS_MASTER Data Flow**: Imports raw data from the Dallas food inspection source files, applies initial transformations including column derivation, and adds surrogate keys before exporting to the staging environment.

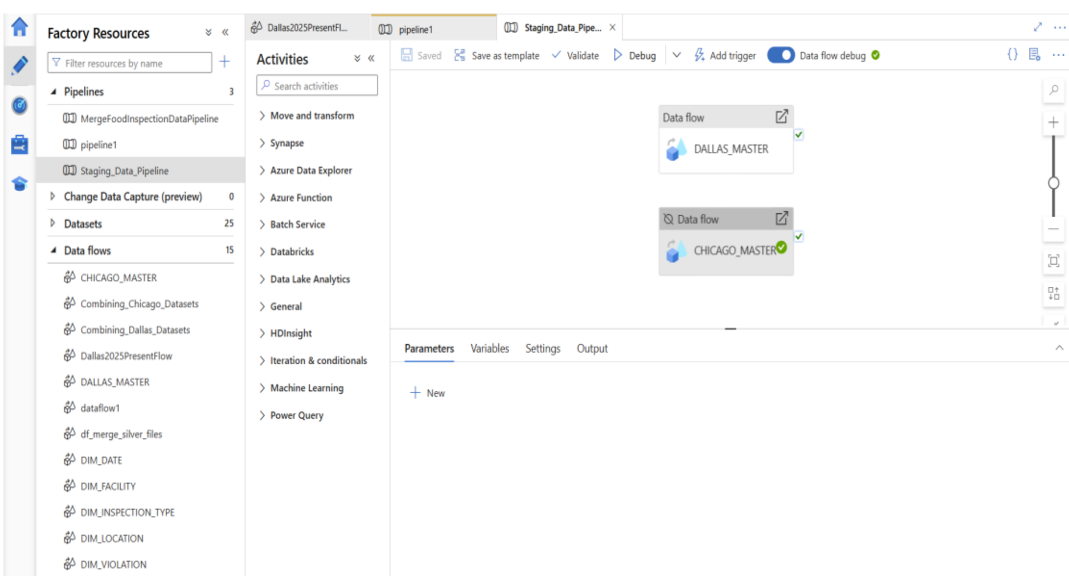


2. **CHICAGO_MASTER Data Flow**: Similarly processes the Chicago inspection data, importing from source files, applying transformations, and exporting to the staging layer.

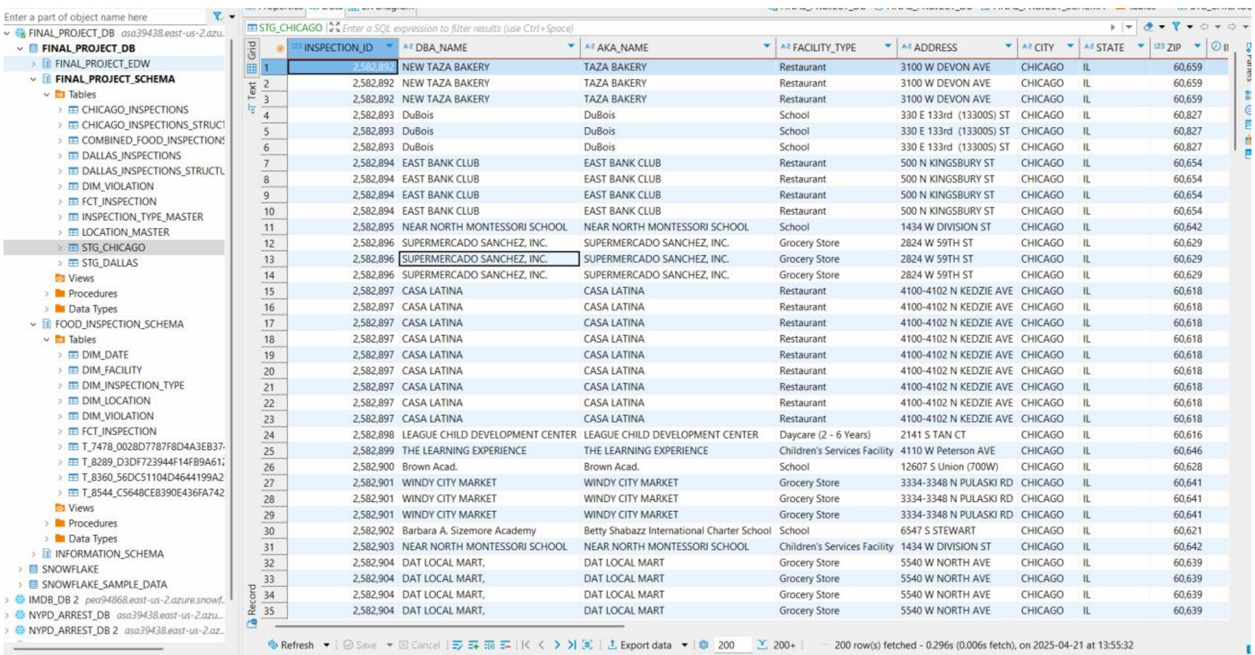


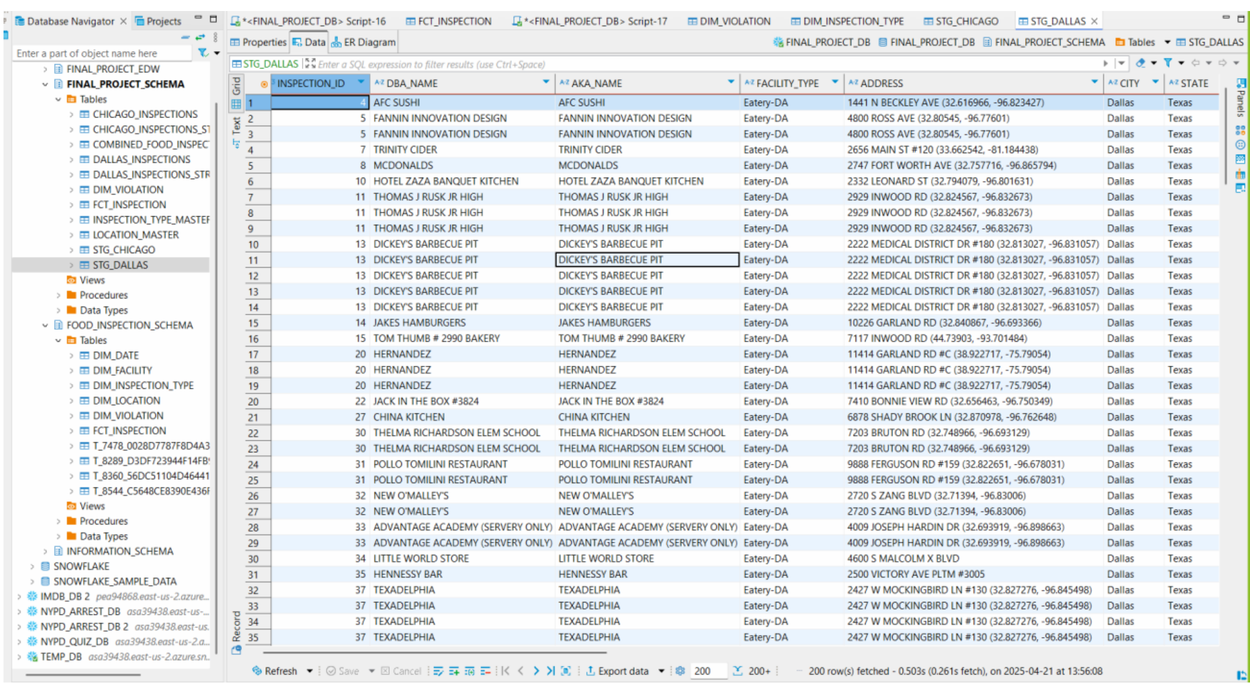
The pipeline structure demonstrates a parallel processing approach for the two city datasets, with each following similar transformation patterns while accommodating their unique data structures. The data flows include essential steps for:

- Source data import
- Column standardization and derivation
- Surrogate key generation
- Export to the appropriate staging tables



This staging pipeline represents the transition from Bronze to Silver in the medallion architecture, preparing the data for subsequent dimensional modeling in the fact and dimension load pipelines.



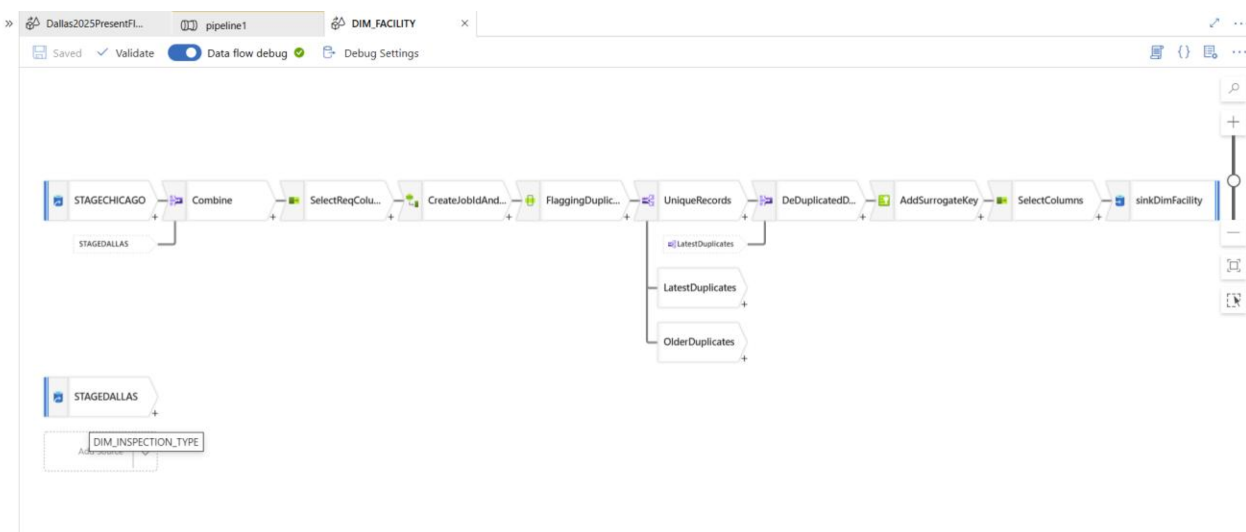


7.2 Dimensional Model ETL Pipelines

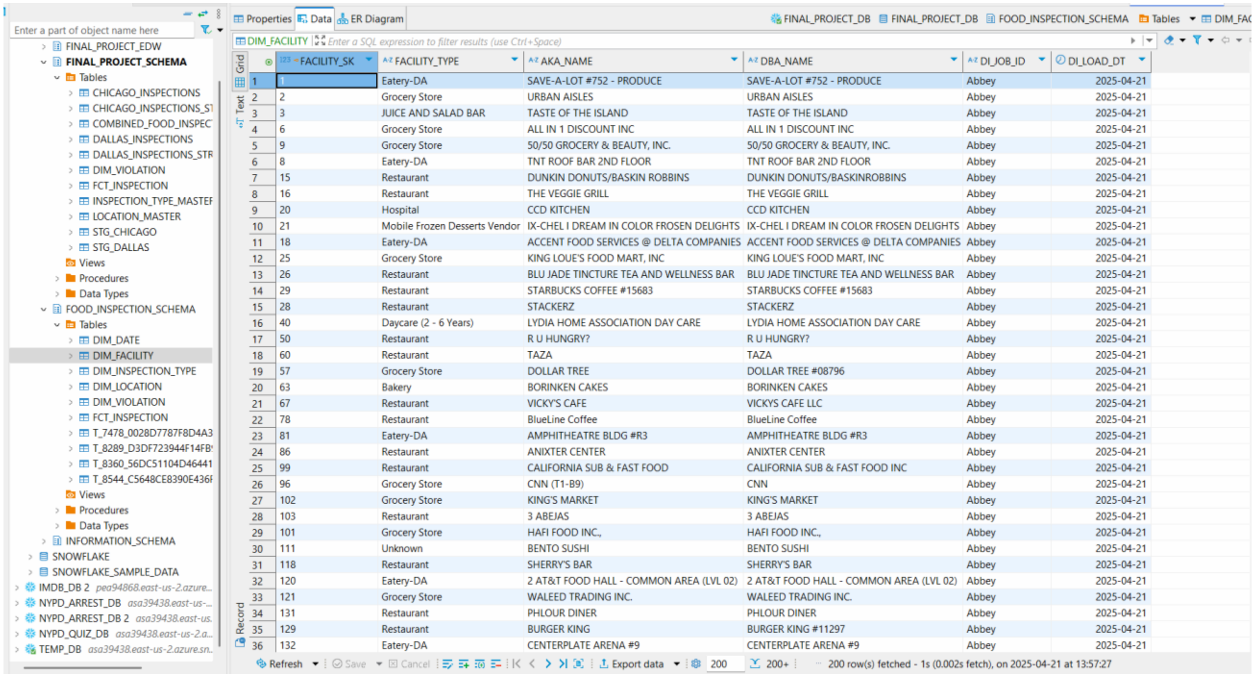
7.2.1 DIM_FACILITY Pipeline

This data flow implements the ETL process for populating the DIM_FACILITY dimension table:

- **STAGECHICAGO/STAGEDALLAS Sources**: Extracts facility data from both Chicago and Dallas staging tables
- **Combine**: Merges the facility data from both cities into a unified stream
- **SelectReqColumns**: Selects and standardizes required columns from the combined dataset
- **CreateJobIdAndDate**: Adds data lineage fields for tracking ETL processing
- **FlaggingDuplicates**: Identifies duplicate facility records based on business keys
- **UniqueRecords**: Separates records into "LatestDuplicates" and "OlderDuplicates" streams
- **DeDuplicatedData**: Removes duplicate records, keeping only the most recent version
- **AddSurrogateKey**: Generates the FACILITY_SK surrogate key for the dimension table
- **SelectColumns**: Finalizes the column selection and ordering
- **sinkDimFacility**: Loads the processed data into the DIM_FACILITY table



This pipeline follows dimensional modeling best practices by properly handling slowly changing dimensions, maintaining data lineage, and implementing surrogate key generation. The process ensures that the facility dimension contains clean, deduplicated records from both Chicago and Dallas food inspection systems.

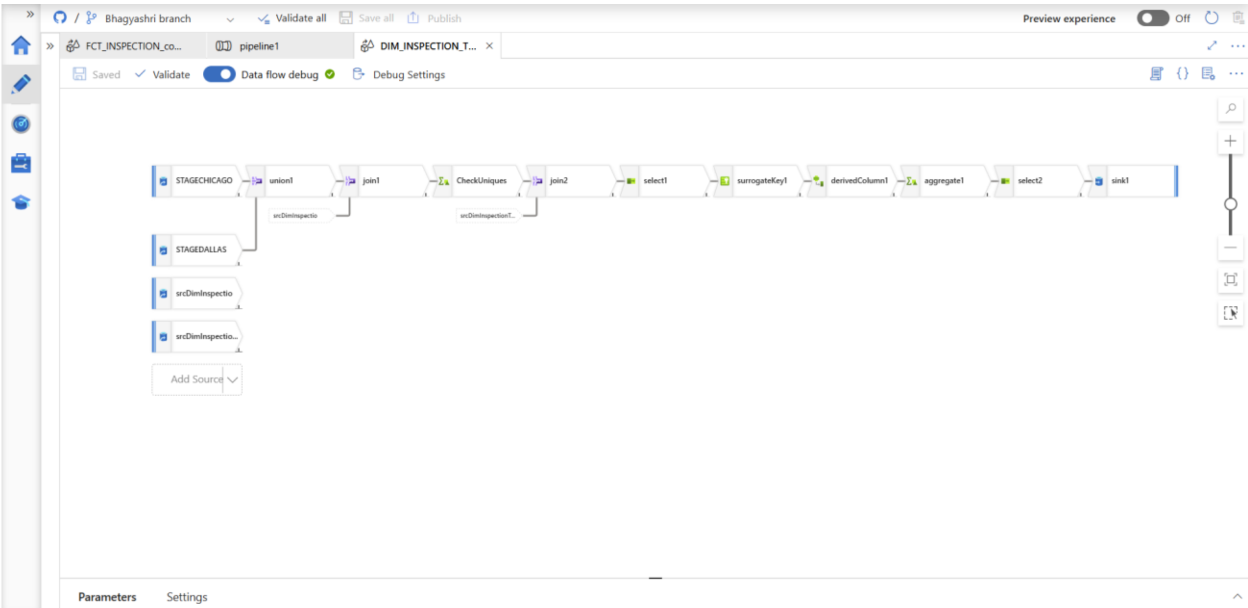


	FACILITY_SK	FACILITY_TYPE	AKA_NAME	DBA_NAME	DI_JOB_ID	DI_LOAD_DT
1	1	Eatery-DA	SAVE-A-LOT #752 - PRODUCE	SAVE-A-LOT #752 - PRODUCE	Abbey	2025-04-21
2	2	Grocery Store	URBAN AISLES	URBAN AISLES	Abbey	2025-04-21
3	3	JUICE AND SALAD BAR	TASTE OF THE ISLAND	TASTE OF THE ISLAND	Abbey	2025-04-21
4	6	Grocery Store	ALL IN 1 DISCOUNT INC	ALL IN 1 DISCOUNT INC	Abbey	2025-04-21
5	9	Grocery Store	50/50 GROCERY & BEAUTY, INC.	50/50 GROCERY & BEAUTY, INC.	Abbey	2025-04-21
6	8	Eatery-DA	TNT ROOF BAR 2ND FLOOR	TNT ROOF BAR 2ND FLOOR	Abbey	2025-04-21
7	15	Restaurant	DUNKIN DONUTS/BASKIN ROBBINS	DUNKIN DONUTS/BASKINROBBINS	Abbey	2025-04-21
8	16	Restaurant	THE VEGGIE GRILL	THE VEGGIE GRILL	Abbey	2025-04-21
9	20	Hospital	CCD KITCHEN	CCD KITCHEN	Abbey	2025-04-21
10	21	Mobile Frozen Desserts Vendor	IX-CHEL I DREAM IN COLOR FROSEN DELIGHTS	IX-CHEL I DREAM IN COLOR FROSEN DELIGHTS	Abbey	2025-04-21
11	18	Eatery-DA	ACCENT FOOD SERVICES @ DELTA COMPANIES	ACCENT FOOD SERVICES @ DELTA COMPANIES	Abbey	2025-04-21
12	25	Grocery Store	KING LOUE'S FOOD MART, INC	KING LOUE'S FOOD MART, INC	Abbey	2025-04-21
13	26	Restaurant	BLU JADE TINCTURE TEA AND WELLNESS BAR	BLU JADE TINCTURE TEA AND WELLNESS BAR	Abbey	2025-04-21
14	29	Restaurant	STARBUCKS COFFEE #15683	STARBUCKS COFFEE #15683	Abbey	2025-04-21
15	28	Restaurant	STACKERZ	STACKERZ	Abbey	2025-04-21
16	40	Daycare (2 - 6 Years)	LYDIA HOME ASSOCIATION DAY CARE	LYDIA HOME ASSOCIATION DAY CARE	Abbey	2025-04-21
17	50	Restaurant	R U HUNGRY?	R U HUNGRY?	Abbey	2025-04-21
18	60	Restaurant	TAZA	TAZA	Abbey	2025-04-21
19	57	Grocery Store	DOLLAR TREE	DOLLAR TREE #08796	Abbey	2025-04-21
20	63	Bakery	BORINKEN CAKES	BORINKEN CAKES	Abbey	2025-04-21
21	67	Restaurant	VICKY'S CAFE	VICKYS CAFE LLC	Abbey	2025-04-21
22	78	Restaurant	BlueLine Coffee	BlueLine Coffee	Abbey	2025-04-21
23	81	Eatery-DA	AMPHITHEATRE BLDG #R3	AMPHITHEATRE BLDG #R3	Abbey	2025-04-21
24	86	Restaurant	ANIXTER CENTER	ANIXTER CENTER	Abbey	2025-04-21
25	99	Restaurant	CALIFORNIA SUB & FAST FOOD	CALIFORNIA SUB & FAST FOOD INC	Abbey	2025-04-21
26	96	Grocery Store	CNN (T1-B9)	CNN	Abbey	2025-04-21
27	102	Grocery Store	KING'S MARKET	KING'S MARKET	Abbey	2025-04-21
28	103	Restaurant	3 ABEJAS	3 ABEJAS	Abbey	2025-04-21
29	101	Grocery Store	HAFI FOOD INC.	HAFI FOOD INC.	Abbey	2025-04-21
30	111	Unknown	BENTO SUSHI	BENTO SUSHI	Abbey	2025-04-21
31	118	Restaurant	SHERRY'S BAR	SHERRY'S BAR	Abbey	2025-04-21
32	120	Eatery-DA	2 AT&T FOOD HALL - COMMON AREA (LVL 02)	2 AT&T FOOD HALL - COMMON AREA (LVL 02)	Abbey	2025-04-21
33	121	Grocery Store	WALEED TRADING INC.	WALEED TRADING INC.	Abbey	2025-04-21
34	131	Restaurant	PHLOUR DINER	PHLOUR DINER	Abbey	2025-04-21
35	129	Restaurant	BURGER KING	BURGER KING #11297	Abbey	2025-04-21
36	132	Eatery-DA	CENTERPLATE ARENA #9	CENTERPLATE ARENA #9	Abbey	2025-04-21

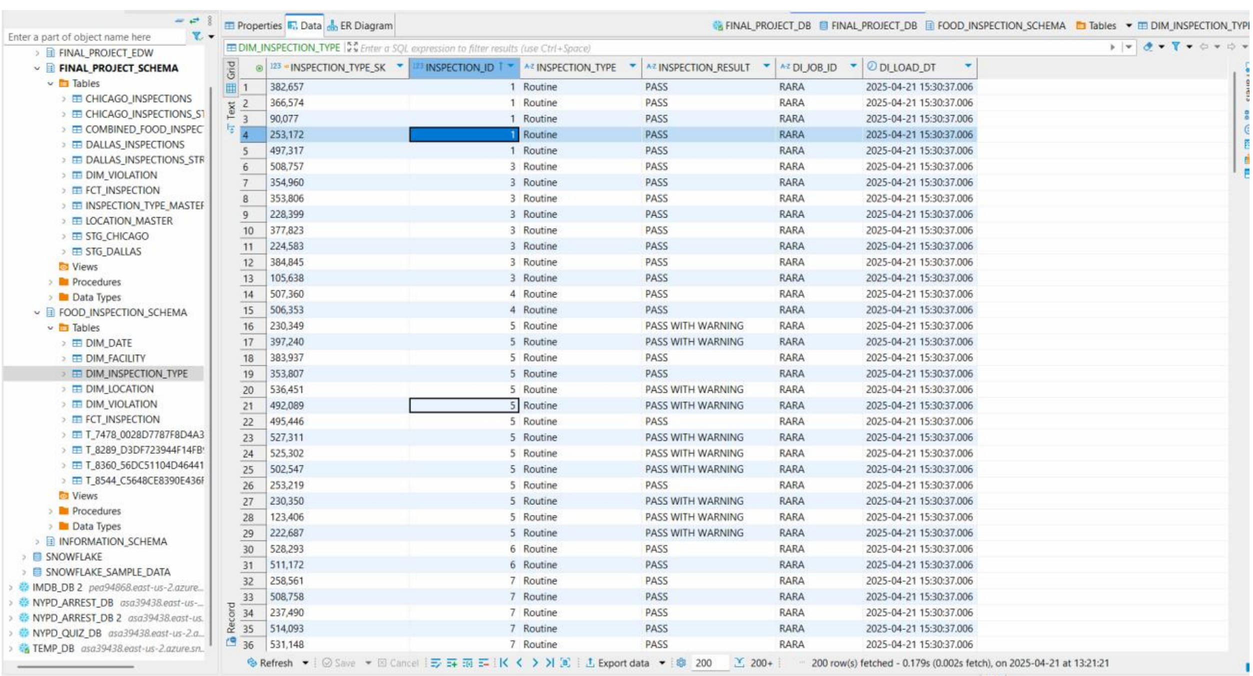
7.2.2 DIM_INSPECTION_TYPE Pipeline

This data flow implements the ETL process for populating the DIM_INSPECTION_TYPE dimension table:

- **STAGE CHICAGO Source**: Extracts inspection type data from the Chicago staging tables
- **union1**: Combines inspection type data from multiple sources
- **join1**: Joins with source inspection type data to ensure completeness and consistency
- **CheckUniques**: Identifies and flags duplicate inspection type records
- **join2**: Joins with additional reference data for inspection type standardization
- **select1**: Selects and standardizes required columns for the dimension table
- **surrogateKey1**: Generates the INSPECTION_TYPE_SK surrogate key for unique identification
- **derivedColumn1**: Creates additional attributes and standardizes formats
- **aggregate1**: Performs data aggregation if required
- **select2**: Finalizes the column selection and ordering for the target dimension
- **sink1**: Loads the processed data into the DIM_INSPECTION_TYPE table



This pipeline standardizes inspection types across both cities, creating a unified reference table that supports cross-city analysis. The process handles differences in inspection type categorization between Chicago and Dallas, ensuring that similar inspection activities can be compared despite different naming conventions in the source systems.



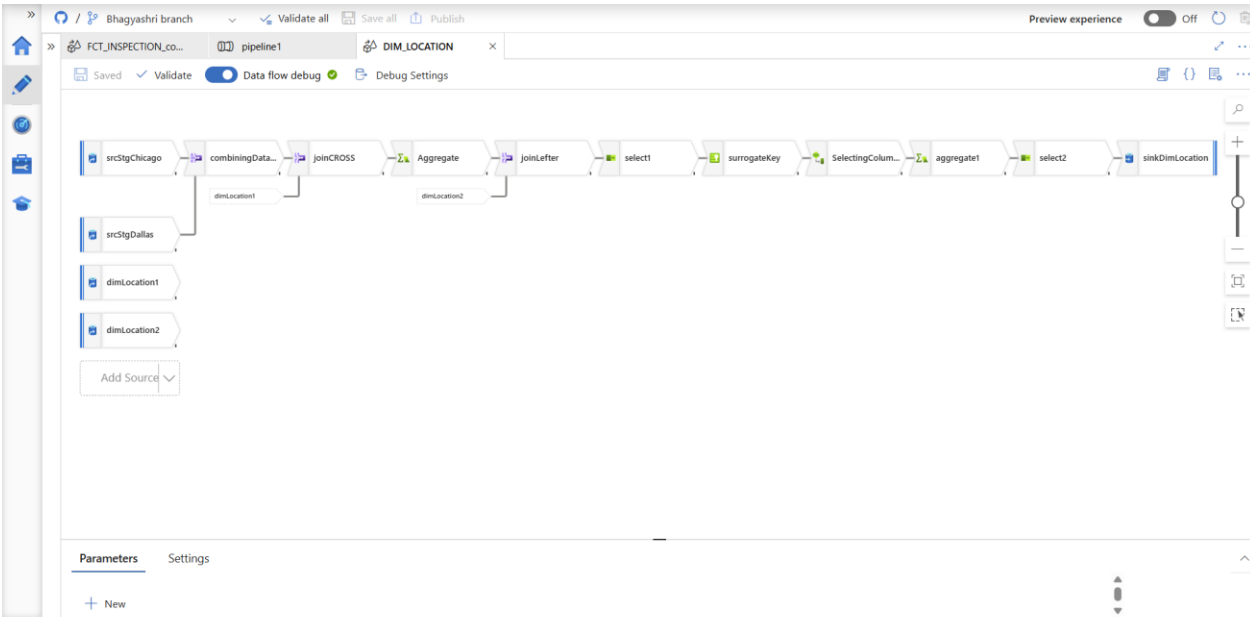
	INSPECTION_TYPE_SK	INSPECTION_ID	INSPECTION_TYPE	INSPECTION_RESULT	DI_JOB_ID	DI_LOAD_DT
1	382,657	1	Routine	PASS	RARA	2025-04-21 15:30:37.006
2	366,574	1	Routine	PASS	RARA	2025-04-21 15:30:37.006
3	90,077	1	Routine	PASS	RARA	2025-04-21 15:30:37.006
4	253,172	1	Routine	PASS	RARA	2025-04-21 15:30:37.006
5	497,317	1	Routine	PASS	RARA	2025-04-21 15:30:37.006
6	508,757	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
7	354,960	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
8	353,806	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
9	228,399	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
10	377,823	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
11	224,583	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
12	384,845	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
13	105,638	3	Routine	PASS	RARA	2025-04-21 15:30:37.006
14	507,360	4	Routine	PASS	RARA	2025-04-21 15:30:37.006
15	506,353	4	Routine	PASS	RARA	2025-04-21 15:30:37.006
16	230,349	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
17	397,240	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
18	383,937	5	Routine	PASS	RARA	2025-04-21 15:30:37.006
19	353,807	5	Routine	PASS	RARA	2025-04-21 15:30:37.006
20	536,451	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
21	492,089	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
22	495,446	5	Routine	PASS	RARA	2025-04-21 15:30:37.006
23	527,311	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
24	525,302	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
25	502,547	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
26	253,219	5	Routine	PASS	RARA	2025-04-21 15:30:37.006
27	230,350	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
28	123,406	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
29	222,687	5	Routine	PASS WITH WARNING	RARA	2025-04-21 15:30:37.006
30	528,293	6	Routine	PASS	RARA	2025-04-21 15:30:37.006
31	511,172	6	Routine	PASS	RARA	2025-04-21 15:30:37.006
32	258,561	7	Routine	PASS	RARA	2025-04-21 15:30:37.006
33	508,758	7	Routine	PASS	RARA	2025-04-21 15:30:37.006
34	237,490	7	Routine	PASS	RARA	2025-04-21 15:30:37.006
35	514,093	7	Routine	PASS	RARA	2025-04-21 15:30:37.006
36	531,148	7	Routine	PASS	RARA	2025-04-21 15:30:37.006

200 row(s) fetched - 0.179s (0.002s fetch), on 2025-04-21 at 13:21:21

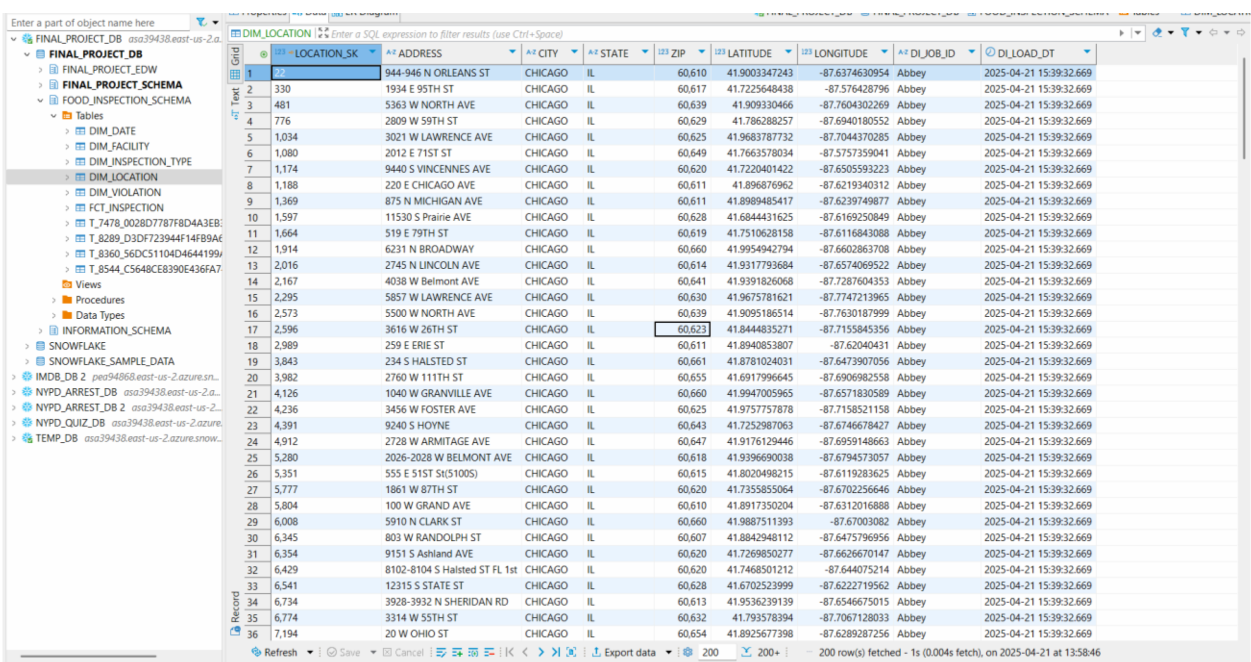
7.2.3 DIM_LOCATION Pipeline

This data flow implements the ETL process for populating the DIM_LOCATION dimension table:

- **srcStgChicago**: Extracts location data from the Chicago staging tables
- **srcStgDallas**: Extracts location data from the Dallas staging tables
- **combiningData**: Merges location data from both Chicago and Dallas sources
- **joinCROSS**: Performs cross-joins necessary for location data consolidation
- **Aggregate**: Groups location records to eliminate duplicates while preserving unique address information
- **joinLefter**: Adds any missing location attributes by left joining with reference data
- **select1**: Selects and standardizes required columns for the dimension table
- **surrogateKey**: Generates the LOCATION_SK surrogate key for unique location identification
- **SelectingColumns**: Refines the column selection and standardizes data formats
- **aggregate1**: Performs final aggregation to ensure data quality
- **select2**: Finalizes the column selection and ordering for the target dimension
- **sinkDimLocation**: Loads the processed data into the DIM_LOCATION table



This pipeline addresses the challenge of integrating location data from two cities with different address formats and components. The Chicago data has unified address fields, while Dallas breaks address into components (street number, name, direction, etc.). The process harmonizes these differences to create a standardized location dimension that supports geographical analysis across both cities.

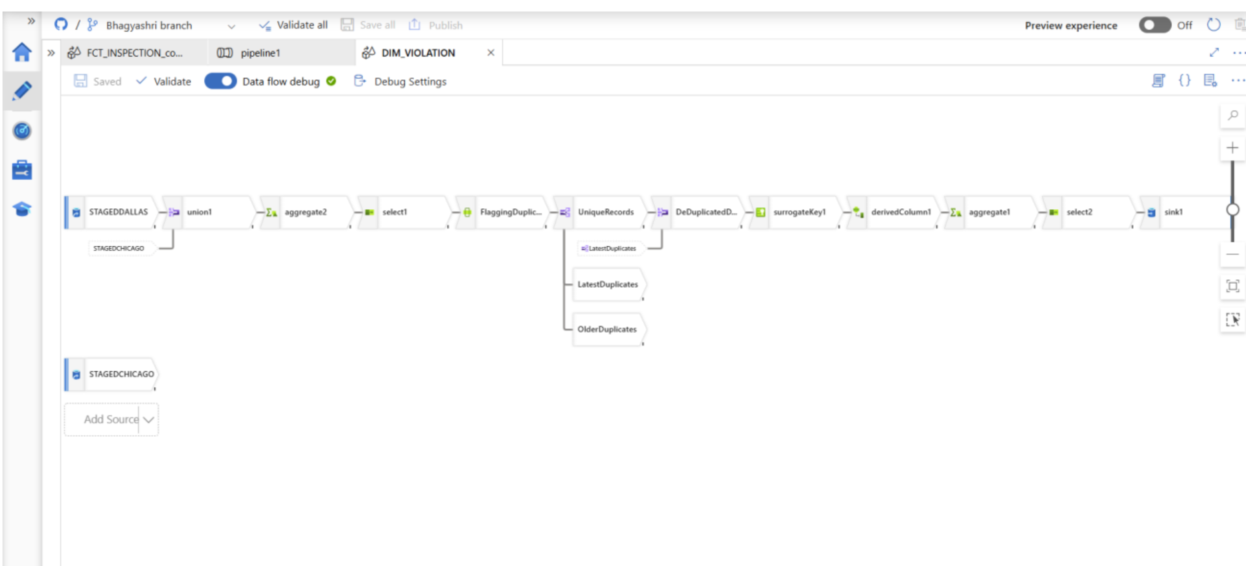


	LOCATION_SK	ADDRESS	CITY	STATE	ZIP	LATITUDE	LONGITUDE	DI_JOB_ID	DI_LOAD_DT
1	22	944-946 N ORLEANS ST	CHICAGO	IL	60,610	41.9003347243	-87.6374630954	Abbey	2025-04-21 15:39:32.669
2	330	1934 E 95TH ST	CHICAGO	IL	60,617	41.7225648438	-87.576428796	Abbey	2025-04-21 15:39:32.669
3	481	5363 W NORTH AVE	CHICAGO	IL	60,639	41.909330466	-87.7604302269	Abbey	2025-04-21 15:39:32.669
4	776	2809 W 59TH ST	CHICAGO	IL	60,629	41.786288257	-87.6940180552	Abbey	2025-04-21 15:39:32.669
5	1,034	3021 W LAWRENCE AVE	CHICAGO	IL	60,625	41.9683787732	-87.7044370285	Abbey	2025-04-21 15:39:32.669
6	1,080	2012 E 71ST ST	CHICAGO	IL	60,649	41.7663578034	-87.5757359041	Abbey	2025-04-21 15:39:32.669
7	1,174	9440 S VINCENNES AVE	CHICAGO	IL	60,620	41.7220401422	-87.6505593223	Abbey	2025-04-21 15:39:32.669
8	1,188	220 E CHICAGO AVE	CHICAGO	IL	60,611	41.896876962	-87.6219340312	Abbey	2025-04-21 15:39:32.669
9	1,369	875 N MICHIGAN AVE	CHICAGO	IL	60,611	41.8989485417	-87.6239749877	Abbey	2025-04-21 15:39:32.669
10	1,597	11530 S Prairie AVE	CHICAGO	IL	60,628	41.6844431625	-87.6169250849	Abbey	2025-04-21 15:39:32.669
11	1,664	519 E 79TH ST	CHICAGO	IL	60,619	41.7510628158	-87.6116843088	Abbey	2025-04-21 15:39:32.669
12	1,914	6231 N BROADWAY	CHICAGO	IL	60,660	41.9954942794	-87.6602863708	Abbey	2025-04-21 15:39:32.669
13	2,016	2745 N LINCOLN AVE	CHICAGO	IL	60,614	41.9317793684	-87.6574069522	Abbey	2025-04-21 15:39:32.669
14	2,167	4038 W Belmont AVE	CHICAGO	IL	60,641	41.9391826068	-87.7287604353	Abbey	2025-04-21 15:39:32.669
15	2,295	5857 W LAWRENCE AVE	CHICAGO	IL	60,630	41.9675781621	-87.7747213965	Abbey	2025-04-21 15:39:32.669
16	2,573	5500 W NORTH AVE	CHICAGO	IL	60,639	41.9095186514	-87.7630187999	Abbey	2025-04-21 15:39:32.669
17	2,596	3616 W 26TH ST	CHICAGO	IL	60,623	41.8444835271	-87.7155845356	Abbey	2025-04-21 15:39:32.669
18	2,989	259 E ERIE ST	CHICAGO	IL	60,611	41.8940853807	-87.62040431	Abbey	2025-04-21 15:39:32.669
19	3,843	234 S HALSTED ST	CHICAGO	IL	60,661	41.8781024031	-87.6473907056	Abbey	2025-04-21 15:39:32.669
20	3,982	2760 W 111TH ST	CHICAGO	IL	60,655	41.6917996645	-87.6906982558	Abbey	2025-04-21 15:39:32.669
21	4,126	1040 W GRANVILLE AVE	CHICAGO	IL	60,660	41.9947005965	-87.6571830589	Abbey	2025-04-21 15:39:32.669
22	4,236	3456 W FOSTER AVE	CHICAGO	IL	60,625	41.9757757878	-87.7158521158	Abbey	2025-04-21 15:39:32.669
23	4,391	9240 S HOYNE	CHICAGO	IL	60,643	41.7252987063	-87.6746678427	Abbey	2025-04-21 15:39:32.669
24	4,912	2728 W ARMITAGE AVE	CHICAGO	IL	60,647	41.9176129446	-87.6959148663	Abbey	2025-04-21 15:39:32.669
25	5,280	2026-2028 W BELMONT AVE	CHICAGO	IL	60,618	41.9396690038	-87.6794573057	Abbey	2025-04-21 15:39:32.669
26	5,351	555 E 51ST St(5100S)	CHICAGO	IL	60,615	41.8020498215	-87.6119283625	Abbey	2025-04-21 15:39:32.669
27	5,777	1861 W 87TH ST	CHICAGO	IL	60,620	41.7355855064	-87.6702256646	Abbey	2025-04-21 15:39:32.669
28	5,804	100 W GRAND AVE	CHICAGO	IL	60,610	41.8917350204	-87.6312016888	Abbey	2025-04-21 15:39:32.669
29	6,008	5910 N CLARK ST	CHICAGO	IL	60,660	41.9887511393	-87.67003082	Abbey	2025-04-21 15:39:32.669
30	6,345	803 W RANDOLPH ST	CHICAGO	IL	60,607	41.8842948112	-87.6475796956	Abbey	2025-04-21 15:39:32.669
31	6,354	9151 S Ashland AVE	CHICAGO	IL	60,620	41.7269850277	-87.6626670147	Abbey	2025-04-21 15:39:32.669
32	6,429	8102-8104 S Halsted ST FL 1st	CHICAGO	IL	60,620	41.7468501212	-87.644075214	Abbey	2025-04-21 15:39:32.669
33	6,541	12315 S STATE ST	CHICAGO	IL	60,628	41.6702523999	-87.6222719562	Abbey	2025-04-21 15:39:32.669
34	6,734	3928-3932 N SHERIDAN RD	CHICAGO	IL	60,613	41.9536239139	-87.6546675015	Abbey	2025-04-21 15:39:32.669
35	6,774	3314 W 55TH ST	CHICAGO	IL	60,632	41.793578394	-87.7067128033	Abbey	2025-04-21 15:39:32.669
36	7,194	20 W OHIO ST	CHICAGO	IL	60,654	41.8925677398	-87.6289287256	Abbey	2025-04-21 15:39:32.669

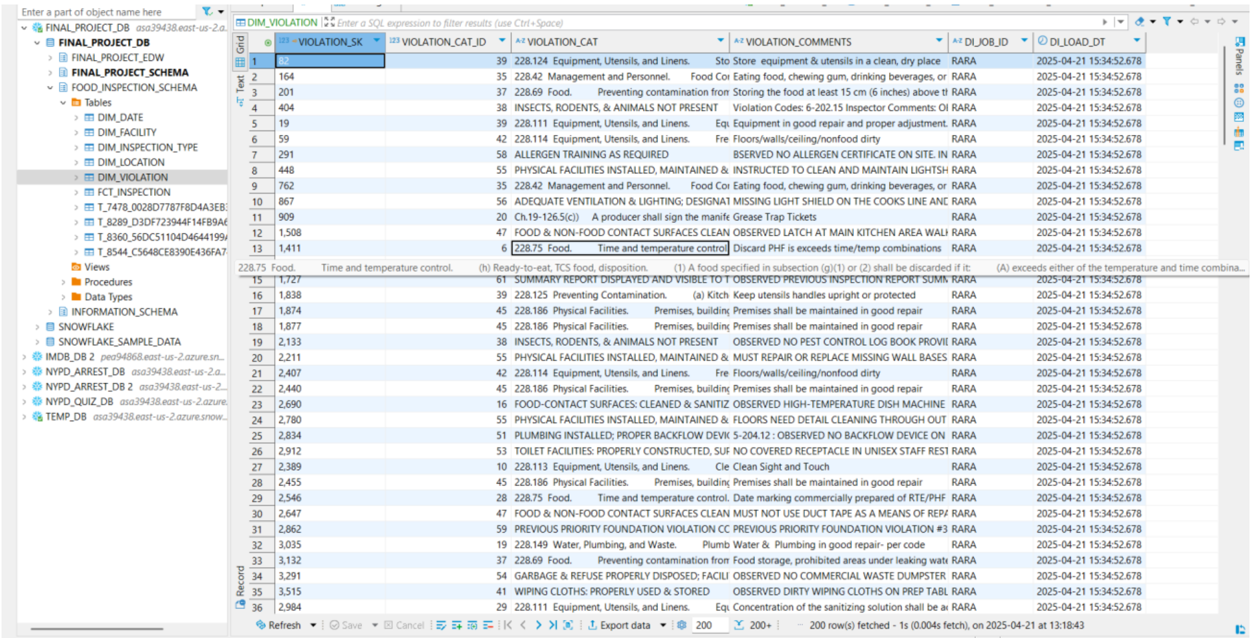
7.2.4 DIM_VIOLATION Pipeline

This data flow implements the ETL process for populating the DIM_VIOLATION dimension table:

- **STAGEDALLAS/STAGEDCHICAGO**: Extracts violation data from both Dallas and Chicago staging tables
- **union1**: Combines violation records from both cities into a unified data stream
- **aggregate2**: Groups similar violation types to identify common categories across cities
- **select1**: Selects and standardizes required columns for the dimension table
- **FlaggingDuplicates**: Identifies duplicate violation records based on violation codes and descriptions
- **UniqueRecords**: Separates records into "LatestDuplicates" and "OlderDuplicates" streams
- **DeDuplicatedData**: Removes duplicate records while preserving the most current information
- **surrogateKey1**: Generates the VIOLATION_SK surrogate key for unique identification
- **derivedColumn1**: Creates additional attributes and standardizes violation descriptions
- **aggregate1**: Performs final aggregation to consolidate similar violation types
- **select2**: Finalizes the column selection and ordering for the target dimension
- **sink1**: Loads the processed data into the DIM_VIOLATION table



This pipeline addresses one of the most challenging aspects of the integration - harmonizing the different violation categorization systems used by Chicago and Dallas. Chicago uses a text-based violation system with categories and comments, while Dallas separates violations into multiple fields with points and descriptions. The pipeline creates a standardized violation dimension that enables cross-city analysis of compliance issues despite the significant differences in the source data structures.

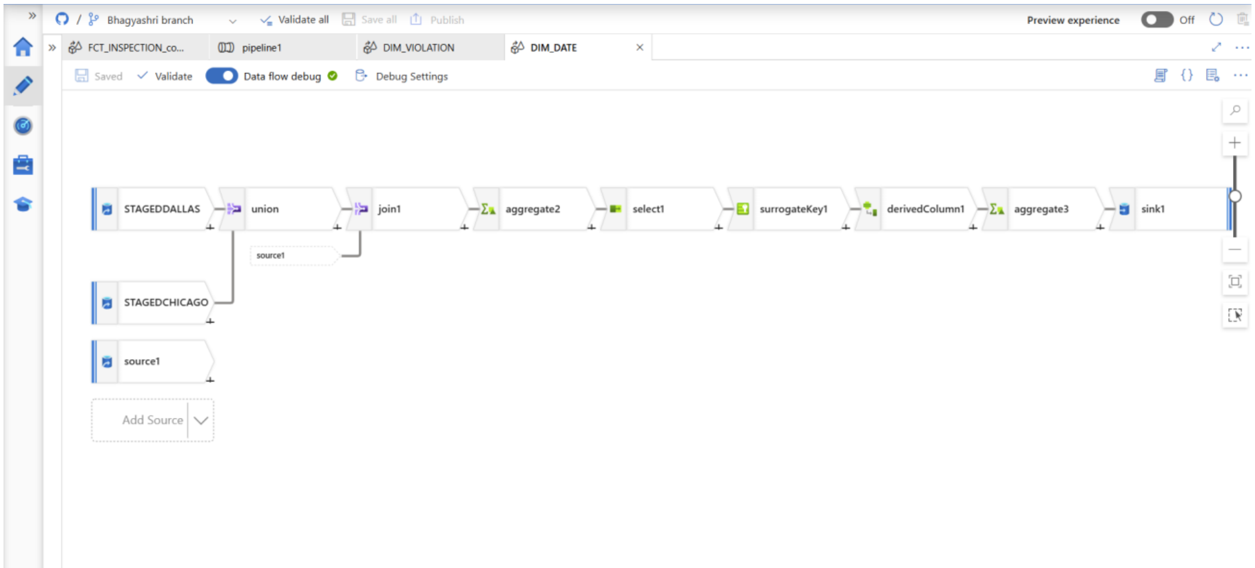


7.2.5 DIM_DATE Pipeline

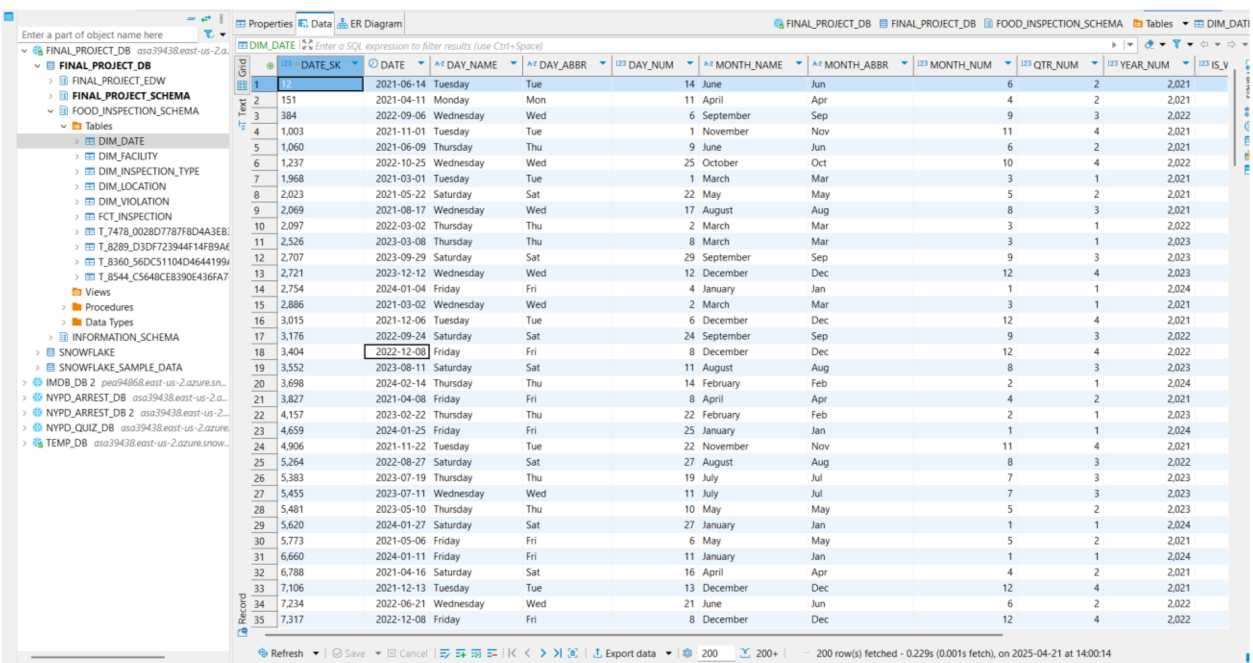
This data flow implements the ETL process for populating the DIM_DATE dimension table:

- **STAGEDALLAS/STAGECHICAGO**: Extracts date information from both Dallas and Chicago staging tables
- **union**: Combines date records from both cities into a unified data stream
- **join1**: Joins with source1 (likely a date generation source) to ensure complete date coverage
- **aggregate2**: Groups date records to eliminate duplicates while ensuring all required dates are present
- **select1**: Selects and standardizes required columns for the dimension table
- **surrogateKey1**: Generates the DATE_SK surrogate key for unique identification
- **derivedColumn1**: Creates additional calendar attributes like day names, month names, quarters, etc.
- **aggregate3**: Performs final aggregation to ensure complete date hierarchy

- **sink1**: Loads the processed data into the DIM_DATE dimension table



This pipeline creates a comprehensive date dimension that supports time-based analysis across both cities' inspection data. The date dimension is designed as a fixed reference table with all calendar attributes needed for temporal analysis, including day/month/year components, weekday flags, and quarter information. This enables consistent time-based reporting regardless of source system differences.



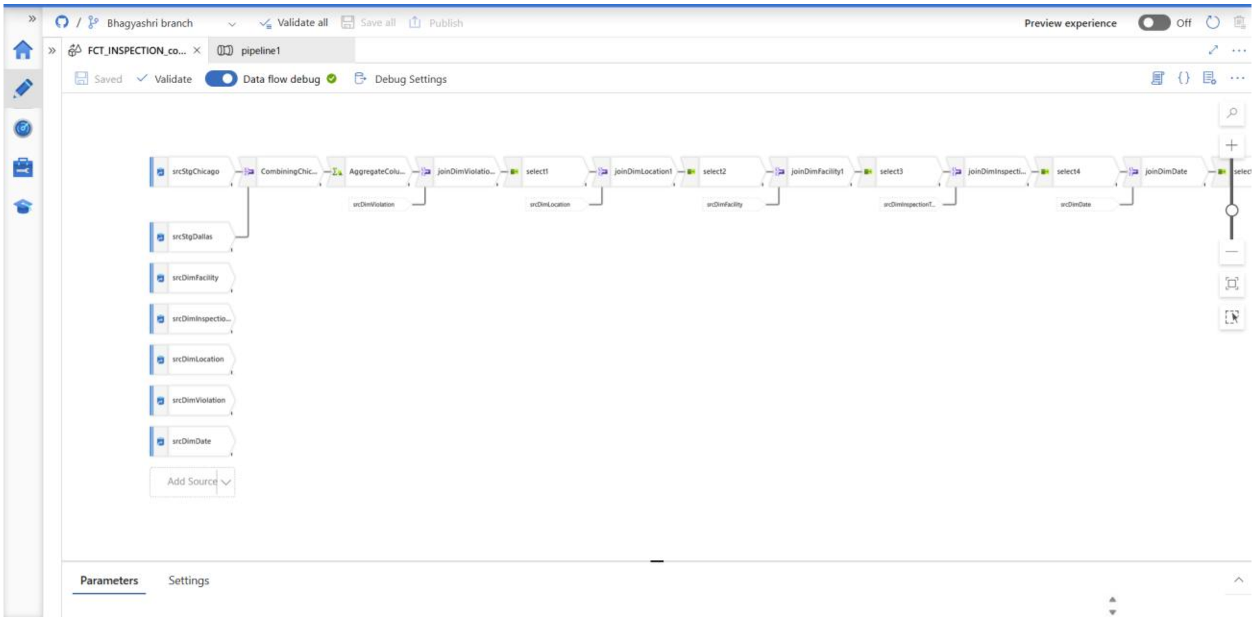
	DATE_SK	DATE	DAY_NAME	DAY_ABBR	DAY_NUM	MONTH_NAME	MONTH_ABBR	MONTH_NUM	QTR_NUM	YEAR_NUM
1	12	2021-06-14	Tuesday	Tue	14	June	Jun	6	2	2,021
2	151	2021-04-11	Monday	Mon	11	April	Apr	4	2	2,021
3	384	2022-09-06	Wednesday	Wed	6	September	Sep	9	3	2,022
4	1,003	2021-11-01	Tuesday	Tue	1	November	Nov	11	4	2,021
5	1,060	2021-06-09	Thursday	Thu	9	June	Jun	6	2	2,021
6	1,237	2022-10-25	Wednesday	Wed	25	October	Oct	10	4	2,022
7	1,968	2021-03-01	Tuesday	Tue	1	March	Mar	3	1	2,021
8	2,023	2021-05-22	Saturday	Sat	22	May	May	5	2	2,021
9	2,069	2021-08-17	Wednesday	Wed	17	August	Aug	8	3	2,021
10	2,097	2022-03-02	Thursday	Thu	2	March	Mar	3	1	2,022
11	2,526	2023-03-08	Thursday	Thu	8	March	Mar	3	1	2,023
12	2,707	2023-09-29	Saturday	Sat	29	September	Sep	9	3	2,023
13	2,721	2023-12-12	Wednesday	Wed	12	December	Dec	12	4	2,023
14	2,754	2024-01-04	Friday	Fri	4	January	Jan	1	1	2,024
15	2,886	2021-03-02	Wednesday	Wed	2	March	Mar	3	1	2,021
16	3,015	2021-12-06	Tuesday	Tue	6	December	Dec	12	4	2,021
17	3,176	2022-09-24	Saturday	Sat	24	September	Sep	9	3	2,022
18	3,404	2022-12-08	Friday	Fri	8	December	Dec	12	4	2,022
19	3,552	2023-08-11	Saturday	Sat	11	August	Aug	8	3	2,023
20	3,698	2024-02-14	Thursday	Thu	14	February	Feb	2	1	2,024
21	3,827	2021-04-08	Friday	Fri	8	April	Apr	4	2	2,021
22	4,157	2023-02-22	Thursday	Thu	22	February	Feb	2	1	2,023
23	4,659	2024-01-25	Friday	Fri	25	January	Jan	1	1	2,024
24	4,906	2021-11-22	Tuesday	Tue	22	November	Nov	11	4	2,021
25	5,264	2022-08-27	Saturday	Sat	27	August	Aug	8	3	2,022
26	5,383	2023-07-19	Thursday	Thu	19	July	Jul	7	3	2,023
27	5,455	2023-07-11	Wednesday	Wed	11	July	Jul	7	3	2,023
28	5,481	2023-05-10	Thursday	Thu	10	May	May	5	2	2,023
29	5,620	2024-01-27	Saturday	Sat	27	January	Jan	1	1	2,024
30	5,773	2021-05-06	Friday	Fri	6	May	May	5	2	2,021
31	6,660	2024-01-11	Friday	Fri	11	January	Jan	1	1	2,024
32	6,788	2021-04-16	Saturday	Sat	16	April	Apr	4	2	2,021
33	7,106	2021-12-13	Tuesday	Tue	13	December	Dec	12	4	2,021
34	7,234	2022-06-21	Wednesday	Wed	21	June	Jun	6	2	2,022
35	7,317	2022-12-08	Friday	Fri	8	December	Dec	12	4	2,022

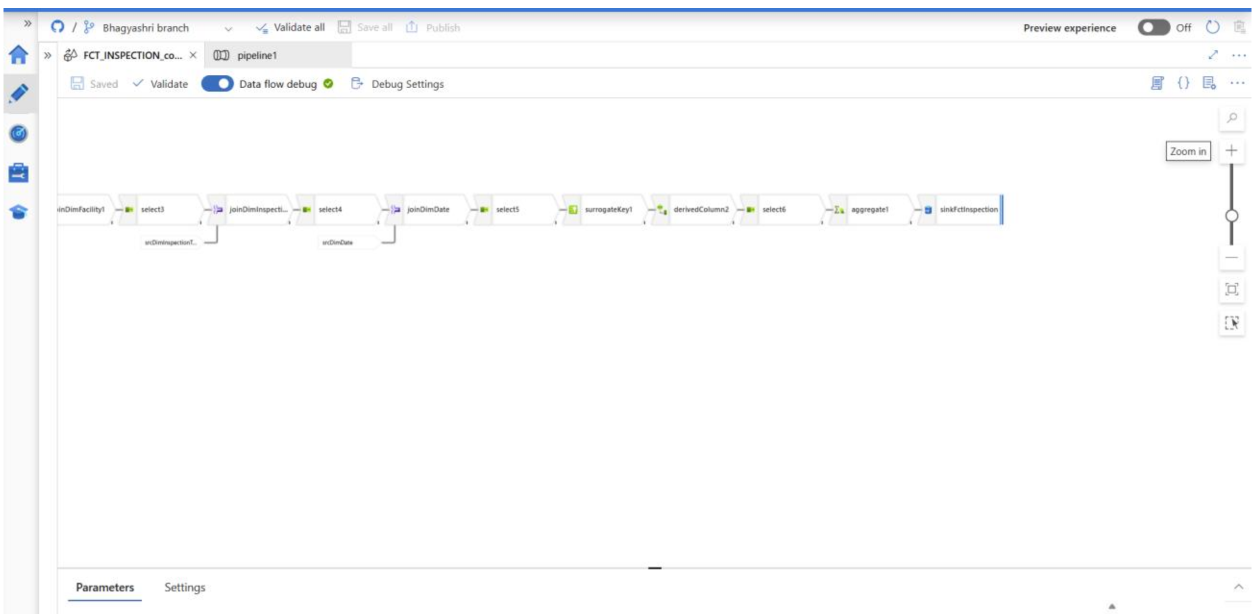
200 row(s) fetched - 0.229s (0.001s fetch), on 2025-04-21 at 14:00:14

7.2.6 FCT_INSPECTION Pipeline

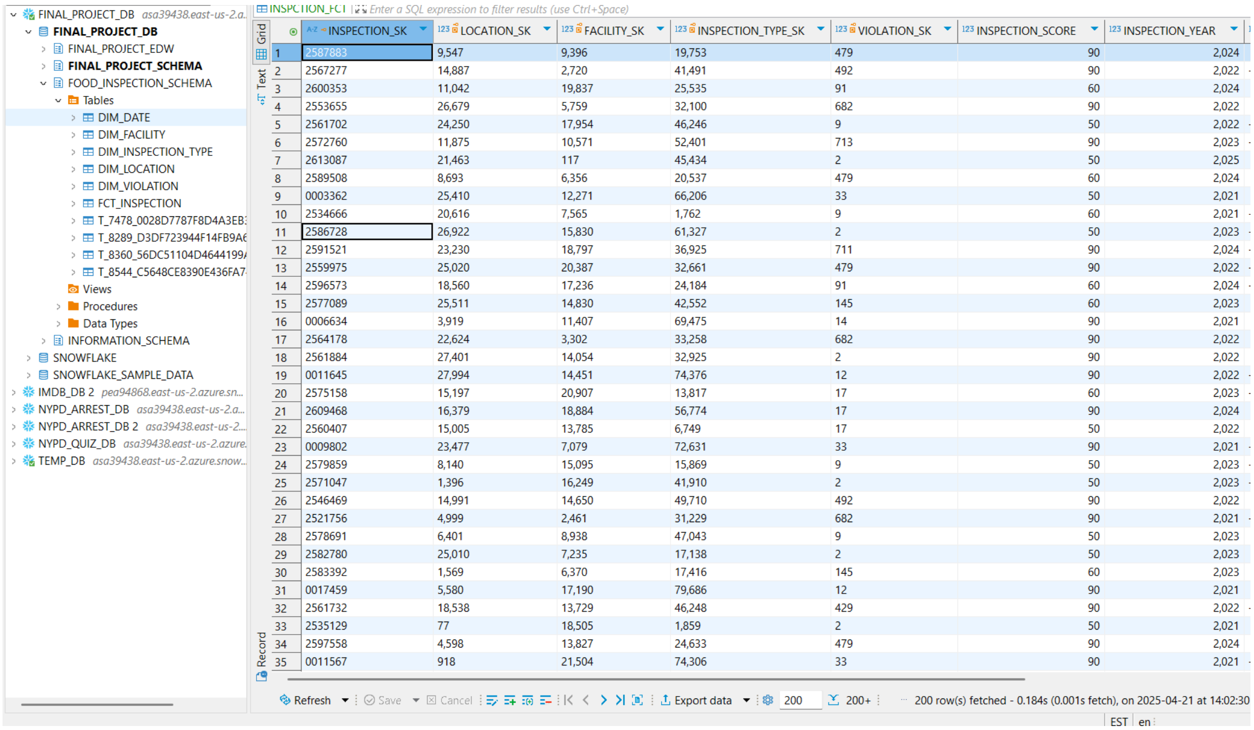
This data flow implements the ETL process for populating the central FCT_INSPECTION fact table:

- **srcStgChicago/srcStgDallas**: Extracts inspection data from Chicago and Dallas staging tables
- **CombiningChicago...**: Combines inspection records from both cities into a unified stream
- **AggregateColumns...**: Performs initial aggregation of inspection metrics
- **joinDimViolation**: Joins with DIM_VIOLATION to connect to standardized violation categories
- **select1**: Performs initial column selection after violation dimension join
- **joinDimLocation**: Joins with DIM_LOCATION to connect geographic information
- **select2**: Refines column selection after location dimension join
- **joinDimFacility**: Joins with DIM_FACILITY to link establishment information
- **select3**: Updates column selection after facility dimension join
- **joinDimInspectionType**: Joins with DIM_INSPECTION_TYPE to standardize inspection classifications
- **select4**: Refines columns after inspection type join
- **joinDimDate**: Joins with DIM_DATE to connect temporal information
- **select5**: Performs further column selection after all dimension joins
- **surrogateKey1**: Generates the INSPECTION_SK primary key for the fact table
- **derivedColumn2**: Creates calculated columns and metrics
- **select6**: Finalizes column selection for the fact table
- **aggregate1**: Performs final aggregation of inspection metrics
- **sinkFctInspection**: Loads the processed data into the FCT_INSPECTION table





This pipeline represents the culmination of the dimensional model implementation, bringing together all dimensions to create the central fact table. It resolves differences between Chicago and Dallas data formats, standardizes metrics, and creates surrogate key relationships with all dimension tables to support comprehensive analysis of food inspection data across both cities.



	INSPECTION_SK	LOCATION_SK	FACILITY_SK	INSPECTION_TYPE_SK	VIOLATION_SK	INSPECTION_SCORE	INSPECTION_YEAR
1	2587883	9,547	9,396	19,753	479	90	2,024
2	2567277	14,887	2,720	41,491	492	90	2,022
3	2600353	11,042	19,837	25,535	91	60	2,024
4	2553655	26,679	5,759	32,100	682	90	2,022
5	2561702	24,250	17,954	46,246	9	50	2,022
6	2572760	11,875	10,571	52,401	713	90	2,023
7	2613087	21,463	117	45,434	2	50	2,025
8	2589508	8,693	6,356	20,537	479	60	2,024
9	0003362	25,410	12,271	66,206	33	50	2,021
10	2534666	20,616	7,565	1,762	9	60	2,021
11	2586728	26,922	15,830	61,327	2	50	2,023
12	2591521	23,230	18,797	36,925	711	90	2,024
13	2559975	25,020	20,387	32,661	479	90	2,022
14	2596573	18,560	17,236	24,184	91	60	2,024
15	2577089	25,511	14,830	42,552	145	60	2,023
16	0006634	3,919	11,407	69,475	14	90	2,021
17	2564178	22,624	3,302	33,258	682	90	2,022
18	2561884	27,401	14,054	32,925	2	90	2,022
19	0011645	27,994	14,451	74,376	12	90	2,022
20	2575158	15,197	20,907	13,817	17	60	2,023
21	2609468	16,379	18,884	56,774	17	90	2,024
22	2560407	15,005	13,785	6,749	17	50	2,022
23	0009802	23,477	7,079	72,631	33	90	2,021
24	2579859	8,140	15,095	15,869	9	50	2,023
25	2571047	1,396	16,249	41,910	2	50	2,023
26	2546469	14,991	14,650	49,710	492	90	2,022
27	2521756	4,999	2,461	31,229	682	90	2,021
28	2578691	6,401	8,938	47,043	9	50	2,023
29	2582780	25,010	7,235	17,138	2	50	2,023
30	2583392	1,569	6,370	17,416	145	60	2,023
31	0017459	5,580	17,190	79,686	12	90	2,021
32	2561732	18,538	13,729	46,248	429	90	2,022
33	2535129	77	18,505	1,859	2	50	2,021
34	2597558	4,598	13,827	24,633	479	90	2,024
35	0011567	918	21,504	74,306	33	90	2,021

8.Conclusion

The Food Establishment Inspections Data Analysis project successfully transforms raw inspection data from Chicago and Dallas into a comprehensive analytical solution that addresses key business requirements while overcoming significant data integration challenges.

Through a well-structured approach using the medallion architecture, the project progresses from raw TSV files (Bronze layer) through standardized Parquet format (Silver layer) to a dimensional model (Gold layer) that supports sophisticated analysis. The Alteryx workflows demonstrate meticulous attention to data quality issues, implementing advanced cleansing techniques for each city's unique data characteristics.

The Azure Data Factory implementation creates a robust ETL framework with separate pipelines for staging data and building the dimensional model. Each dimension table (Facility, Location, Inspection Type, Violation, and Date) is carefully constructed to harmonize differences between the cities' data structures, while the fact table brings everything together with proper relationships and metrics.

The dimensional model enables powerful analytical capabilities as demonstrated by the business requirements queries. These queries showcase the model's ability to support establishment-level analysis, geographic insights, temporal trends, and comparative analytics between cities. The solution successfully transforms disparate municipal datasets into a unified framework for food safety analysis.

By implementing this solution, health departments gain valuable tools for risk-based resource allocation, establishments benefit from consistent compliance measurement, and the public receive greater transparency into food safety in their communities. The project demonstrates the power of modern data transformation techniques to create actionable insights from complex, inconsistent source data.