# Northeastern University
# Course Code: ALY 6030
# Data Warehousing and SQL
# Instructor: Adam Jones Date
# Submitted: May 11th, 2025
# Public Housing Inspection Data Analysis

This report examines the provided public housing inspection dataset to identify its facts and dimensions, recommend appropriate fact table designs, propose a method for handling changing PHA attributes, and construct an SQL query for cost-change analysis. It follows dimensional modeling best practiceskimballgroup.comastera.com and cites authoritative sources in APA style.

## Facts in the Dataset

The dataset's measures (numerical facts) are **Cost of Inspection** and **Inspection Score**. These are the quantitative values recorded in each inspection. We count **two facts**: **COST_OF_INSPECTION_IN_DOLLARS** and **INSPECTION_SCORE**. According to Kimball, facts can be *additive*, *semi-additive*, or *non-additive*kimballgroup.com. In this data:

- **Cost of Inspection** is **fully additive**: it can be summed across any combination of dimensions (for example, summing costs across PHAs, dates, or developments)kimballgroup.com. This makes it a flexible metric for aggregation.
- **Inspection Score** (typically a percentage or points out of 100) is **non-additive**: summing scores across inspections has no meaningful interpretationkimballgroup.com. (In practice one would compute averages of scores, but the score itself cannot be aggregated by summation.)
- *Inspection ID* is a unique key but not a measure to aggregate, and all other fields are descriptive.

Thus, the dataset has two numeric facts (Cost and Score). Cost is additive, while Score is non-additivekimballgroup.com.

## Dimensions in the Dataset

The remaining fields form descriptive attributes for dimensions. We identify **three dimensions**:

- **Public Housing Agency (PHA) Dimension**: Identified by *PUBLIC_HOUSING_AGENCY_NAME*. This dimension describes each agency responsible for inspections. It may include additional attributes (e.g. a surrogate key), but in the dataset the only attribute is the agency name.
- **Development Dimension**: Comprising *INSPECTED_DEVELOPMENT_NAME*, *INSPECTED_DEVELOPMENT_ADDRESS*, *INSPECTED_DEVELOPMENT_CITY*, and *INSPECTED_DEVELOPMENT_STATE*. Together these describe each housing development that was inspected. A dimension table could list each unique development by a surrogate key, with its name and location attributes.
- **Date Dimension**: From *INSPECTION_DATE*. This dimension describes calendar attributes (year, month, day, etc.) for each inspection date. It allows analysis by time periods.

Dimension tables **store descriptive information about business entities to help analyze facts**astera.com. For example, the PHA dimension would hold agency names, and the Development dimension would hold location details. In total there are three dimensions in this model (PHA, Development, and Date), each providing context for the fact measures.

**Fact Table Design Recommendation**

Given the nature of this data, a **transactional fact table** is essential. Each record in the data represents one inspection event (an occurrence of a inspection with a score and cost), which aligns with a transactional fact table structuremedium.com. A transactional fact table would include foreign keys to the PHA, Development, and Date dimensions, and measures for cost and score. This allows detailed, row-by-row analysis of inspections.

In addition, a **periodic snapshot fact table** may be useful for aggregated analysis. For example, one could build a monthly or quarterly snapshot fact table that summarizes total inspections, total cost, or average score per PHA or region. As Chandru Gurumoorthi explains, periodic snapshots "provide a summarized view of metrics over regular time intervals" and are useful for monitoring trendsmedium.com. By contrast, the transactional table captures every inspection event in detail. In practice, having both designs can be valuable: the transactional table supports granular queries (e.g. individual inspections), while snapshot tables enable fast access to period-level aggregates and trendsmedium.commedium.com.

## Handling Changes in PHA Names/Addresses (SCD)

Public Housing Agency names and addresses may change over time (e.g. rebranding or office moves), and agencies may merge. To track these historical changes, a **Slowly Changing Dimension (SCD)** strategy is needed. The common Kimball SCD types are:

- **Type 0 (No change)** – never updates old data.
- **Type 1 (Overwrite)** – update the dimension record in-place (no history kept).
- **Type 2 (Row versioning)** – insert a new dimension row on change (preserves history).
- **Type 3 (Add attribute)** – add new columns for the old value (limited history).

For PHA names/addresses, we recommend **Type 2 SCD**. This approach preserves the history of each agency's identity by adding a new row with the updated name/address, while marking the previous row as historicalsqlshack.com. Type 2 is widely used "to track historical data in a data warehouse"sqlshack.com. Type 1 would lose historical names (undesirable if we want to report on legacy data), and Type 3 can only store one previous value. A Type 2 design allows, for example, multiple versions of a PHA dimension row (before and after a name change).

Considering PHA mergers: if two agencies merge, one could create a new combined PHA key (or reassign inspections to one surviving agency) while preserving the old agency records in history tables. Type 2 can accommodate mergers by ending the old agencies' rows and starting a new row for the merged entity. Overall, Type 2 ensures that past inspection data can be analyzed with the correct historical agency attributes, satisfying compliance and analytical needssqlshack.com.

## SQL Script for Most-Recent Inspection Cost Increases

The following SQL (MySQL) script identifies PHAs where the most recent inspection cost has increased compared to the previous inspection. It:

- Converts the date from text to DATE using STR_TO_DATE().
- Uses window functions (LAG()) to bring in the previous inspection's date and cost for each PHA, ordered by date descending.
- Computes the absolute and percent change in cost.
- Filters to PHAs with at least two inspections (PREV_DATE IS NOT NULL) and where the cost increased.

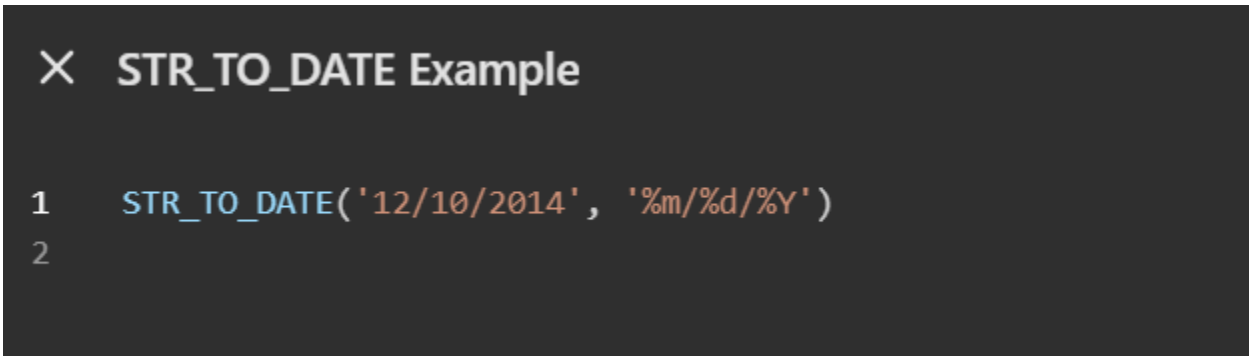
- Outputs one row per PHA with the two dates and costs.

## Notes on the SQL:

- STR_TO_DATE(INSPECTION_DATE, '%m/%d/%Y') casts the text date (e.g. "12/10/2014") into a true DATE type in MySQL.
- The LAG() window function brings in the previous inspection's date and cost (partitioned by PHA, ordered by date desc).
- ROW_NUMBER() = 1 ensures we only take the most recent inspection per PHA.
- The final filter (PREV_DATE IS NOT NULL AND COST > PREV_COST) ensures at least two inspections and an increase in cost.
- The output columns match exactly the requested names: PHA_NAME, MR_INSPECTION_DATE, MR_INSPECTION_COST, SECOND_MR_INSPECTION_DATE, SECOND_MR_INSPECTION_COST, CHANGE_IN_COST, PERCENT_CHANGE_IN_COST.

## Converting Text Dates to DATE in MySQL

In MySQL, you can convert a text string to a DATE using the STR_TO_DATE() functionw3schools.com. For example:




This returns 2014-12-10 as a DATE. The format string uses %m for month, %d for day, and %Y for four-digit year, as documented by the MySQL referencew3schools.com. Using STR_TO_DATE ensures dates can be compared and ordered chronologically in SQL queries.

## Conclusion

In summary, the public housing inspection dataset contains two fact measures (cost and score) and three main dimensions (PHA, Development, Date). Cost is an additive fact, while score is non-additivekimballgroup.com. A transactional fact table (one row per inspection) is recommended for detailed analysismedium.com, complemented by periodic snapshots for trend analysismedium.com. Handling changes in PHA attributes is best done with a Type 2 slowly changing dimension to preserve historysqlshack.com. Finally, the provided SQL query demonstrates how to compute cost increases between the most recent inspections using LAG() and date conversion functions in MySQL.

## References


- Chandru, G. (2023, June 26). *Types of Fact Tables: Data Warehouse*. Medium. Retrieved from https://medium.com/@changuru2023/types-of-fact-tables-data-warehouse-cec3cfaa2efe:contentReference[oaicite:20]{index=20}:contentReference[oaicite:21]{index=21}
- Kimball Group. (n.d.). *Additive, Semi-Additive, and Non-Additive Facts*. Retrieved from https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/additive-semi-additive-non-additive-fact/:contentReference[oaicite:22]{index=22}
- SQLShack (Asanka, D.). (2022, May 30). *Testing Type 2 Slowly Changing Dimensions in a Data Warehouse*. SQLShack. Retrieved from https://www.sqlshack.com/testing-type-2-slowly-changing-dimensions-in-a-data-warehouse/:contentReference[oaicite:23]{index=23}
- Astera Knowledge Center. (n.d.). *What is Dimensional Data Modeling? Examples, Benefits & More*. Retrieved from https://www.astera.com/knowledge-center/dimensional-modeling-guide/:contentReference[oaicite:24]{index=24}
- W3Schools. (n.d.). *MySQL STR_TO_DATE() Function*. Retrieved from https://www.w3schools.com/sql/func_mysql_str_to_date.asp:contentReference[oaicite:25]{index=25}