Module 5 Assignment 3

Public Housing Inspections Star Schema

Chuanzhe Hu

College of Professional Studies, Northeastern University

ALY6030 Data Warehousing & SQL

Prof. Adam Jones

Nov 30th, 2024

Introduction

In this project we conducted an analysis of Public Housing Agency (PHA) inspection data to identify changes in inspection costs over time. Our primary goal working for Federal Housing & Urban Development (HUD) department in this project, was to understand the dataset and how to construct the star schema database for them given the business context. By leveraging SQL window functions and a series of filtering and transformation steps, we created a sub dataset that provides insights into inspection cost changes while ensuring data accuracy and relevance.

Analysis

Part 1:

1. *How many facts are there in this dataset? Which facts do you identify? For the facts that you identify, what type of facts are they?*

In the dataset, there are two facts. One is COST_OF_INSPECTION_IN_DOLLARS, it measures the cost of performing that inspection in dollar amount, which is an additive fact. Another one is INSPECTION_SCORE, which is a ratio from 0 to 100, this is non-additive fact.

2. *How many dimensions are there in this dataset? Which dimensions do you identify?*

There are six dimensions in the dataset. They are listed below:
PUBLIC_HOUSING_AGENCY_NAME, INSPECTED_DEVELOPMENT_NAME, INSPECTED_DEVELOPMENT_ADDRESS, INSPECTED_DEVELOPMENT_CITY, INSPECTED_DEVELOPMENT_STATE, and INSPECTION_DATE.

3. *Senior management is interested in viewing the facts identified above, at both the inspection level, as well as a periodic summary of inspection costs for each month. Based on this context, if you were to store these data in a set of fact tables, which type (or types) of fact tables would you use and why?*

For the purpose to view the facts individually, we'd create a **transaction fact table**, where records individual inspections event, including the inspection cost in dollars and inspection score.

For the periodic summary of inspection costs for each month, we'd create a **periodic snapshot fact table** to present the summary over a defined period, like a month. In this table, we do not store individual records, but provide a summary, for example the total inspection fee adds up amounts to the management.

4. *Senior Management is also concerned with changes in the names and addresses of the public housing agency names since they tend to get merged with other agencies on a frequent basis. Based on this, how should we handle this slowly changing dimension? Select from types 0, 1, 2, or 3 from the Kimball reading. Justify your answer.*

In our case, we recommend using Type 2 SCD to handle the frequent changes for agencies. For example, if an agency changes its name due to a merger, we will create a new surrogate key for the agency with the new name, as well as include a column to indicate whether a record is current. This approach allows us to maintain a complete history of agency changes. While Type 2 changes can be resource-intensive, given our dataset size of 6,464 records, it is manageable and applicable.

5. *Finally, Senior Management is interested in a subset of this data, for only those PHAs that saw an increase in the cost of performing an inspection in their jurisdiction. Since none of them are SQL programmers, they've asked your help in performing this analysis by providing a file as your final deliverable with the following columns (note that MR stands for "most recent"):*

a. PHA_NAME

b. MR_INSPECTION_DATE

c. MR_INSPECTION_COST

d. SECOND_MR_INSPECTION_DATE

e. SECOND_MR_INSPECTION_COST

f. CHANGE_IN_COST

g. PERCENT_CHANGE_IN_COST

Management has asked that you perform this function using lead or lag functions in SQL. However, they're concerned that the files when imported into MySQL Workbench may not properly refer to dates using the correct format. If that is the case, they've asked you to investigate how best to convert dates from TEXT to Date format so that the lead/lag functions work as expected.

They've also asked that you filter your dataset to only those PHAs that saw an increase in cost (in dollars), and that you only list the PHA once with no duplicates to avoid noisy data. Naturally, this would also require you to filter out PHAs that only performed one inspection, so they've asked you to remove those as well.

To address the requirements by senior management, we imported the csv file into MySQL workbench and performed analysis. We created a database named "Housing", then used the import wizard uploaded the file. Table was named as "inspection".

First, we addressed the data type transformation from string to date format.

```
-- after import csv file through wizard,
-- Edit date format in table

SET SQL_SAFE_UPDATES = 0;
-- change all delimiters from / to -
UPDATE inspection
SET INSPECTION_DATE = REPLACE(INSPECTION_DATE, '/', '-')
WHERE INSPECTION_DATE LIKE '%/%';

-- change the string to date using STR_TO_DATE
UPDATE inspection
SET INSPECTION_DATE = CASE
    WHEN INSPECTION_DATE LIKE '%-%-____' THEN STR_TO_DATE(INSPECTION_DATE, '%c-%e-%Y')
    ELSE INSPECTION_DATE
END
WHERE INSPECTION_DATE LIKE '%-%-____';

-- they now have a silimar format, transform the column format to DATE
ALTER TABLE inspection
MODIFY COLUMN INSPECTION_DATE DATE;
-- view any distinct date left out
SELECT DISTINCT INSPECTION_DATE
FROM inspection;
-- validate the date form transformation if successful
describe inspection;
SET SQL_SAFE_UPDATES = 1;

```

The SQL queries shown above work to transform the INSPECTION_DATE column in the inspection table from a text string format to a DATE data type. To start with, the INSPECTION_DATE values like “12/10/2014” contain inconsistent delimiters (/), which are replaced with dashes (-) for uniformity. Then, the STR_TO_DATE function is used to convert the string representation of dates into the DATE type. The conversion is performed conditionally using a CASE statement to ensure values that match the format (%c-%e-%Y, where %c represents numeric month 0..12, %e represents numeric day of the month 0..31, and %Y represents four digits numeric year from MySQL, 2024.) are transformed, while other may exist values remain unchanged to avoid errors. Finally, the column is altered to the DATE type, and the distinct dates are verified to confirm the successful transformation. This process ensures that 6464 records INSPECTION_DATE column is stored correctly as a DATE, enabling consistent and later date-based operations.

To generate the final output for senior management, we took a series steps.

We started by creating a new table named pha_inspections to present the most recent and the second most recent inspection dates and costs for each PHAs. We used the LEAD window function to create two new columns: SECOND_MR_INSPECTION_DATE and SECOND_MR_INSPECTION_COST. The LEAD function was set to get the next inspection date and cost for each PHA, ordered by INSPECTION_DATE in descending order, thereby providing the information for the second most recent inspection.

```
37    -- Q5
38    -- step1: Create a new table to retrieve the most recent and second most recent inspections for each PHA
39 •  DROP TABLE IF EXISTS pha_inspections;
40 •  CREATE TABLE pha_inspections AS
41    SELECT
42        PUBLIC_HOUSING_AGENCY_NAME AS PHA_NAME,
43        INSPECTION_DATE AS MR_INSPECTION_DATE,
44        COST_OF_INSPECTION_IN_DOLLARS AS MR_INSPECTION_COST,
45        Lead(INSPECTION_DATE, 1) OVER (PARTITION BY PUBLIC_HOUSING_AGENCY_NAME ORDER BY INSPECTION_DATE DESC) AS SECOND_MR_INSPECTION_DATE,
46        Lead(COST_OF_INSPECTION_IN_DOLLARS, 1) OVER (PARTITION BY PUBLIC_HOUSING_AGENCY_NAME ORDER BY INSPECTION_DATE DESC) AS SECOND_MR_INSPECTION_COST
47    FROM inspection;
48
49 •  SELECT * FROM pha_inspections;-- view the table
```

From the above summary tables, the top 2 hospitals for license beds are Phoenix

In step 2, we filtered out PHAs that did not have a second most recent inspection by removing any rows where SECOND_MR_INSPECTION_DATE or SECOND_MR_INSPECTION_COST was NULL. This helped in ensuring that only PHAs with at least two inspections were retained for further analysis.

```
51    -- Step2: Filter out PHAs that do not have a second most recent inspection
52    DROP TABLE IF EXISTS t2;
53    CREATE TABLE t2 AS
54    SELECT *
55    FROM pha_inspections
56    WHERE SECOND_MR_INSPECTION_DATE IS NOT NULL AND SECOND_MR_INSPECTION_COST IS NOT NULL;
57
58    SELECT * FROM t2;
```

In step 3, we added columns for the change in inspection cost and the percentage change between the most recent and second most recent inspections. We used a calculation to find the difference (MR_INSPECTION_COST - SECOND_MR_INSPECTION_COST) as CHANGE_IN_COST and cast the percentage calculation to a decimal value to ensure appropriate formatting. Additionally, we applied a filter to retain only positive cost differences.

```
60    -- step3: Add on cost change (positive), and percent change to t2
61    DROP TABLE IF EXISTS t3;
62    CREATE TABLE t3 AS
63    SELECT
64        PHA_NAME,
65        MR_INSPECTION_DATE,
66        MR_INSPECTION_COST,
67        SECOND_MR_INSPECTION_DATE,
68        SECOND_MR_INSPECTION_COST,
69        -- new columns, change in cost
70        (MR_INSPECTION_COST - SECOND_MR_INSPECTION_COST) AS CHANGE_IN_COST,
71        CAST(((MR_INSPECTION_COST - SECOND_MR_INSPECTION_COST) / SECOND_MR_INSPECTION_COST) * 100 AS DECIMAL(10, 2)) AS PERCENT_CHANGE_IN_COST
72    FROM pha_inspections
73    WHERE (MR_INSPECTION_COST - SECOND_MR_INSPECTION_COST) > 0; -- only store the positive costs diffs
```

Step 4 involved limiting each PHA to a single display in the output. To achieve this, we used the RANK() function with a partition on PHA_NAME and ordered by MR_INSPECTION_DATE in descending order. This ranked the records for each PHA based on inspection dates. Finally, in step 5, we generated the **final_output** table by selecting only the rows where RANKING = 1. This ensured that only the most recent

inspection record for each PHA was included in the final output, 442 rows were returned.

```sql
-- step4: Limit PHA name to only one time display, create ranking as a new column
DROP TABLE IF EXISTS t4;
CREATE TABLE t4 AS
SELECT
    *,
    RANK() OVER (PARTITION BY PHA_NAME ORDER BY MR_INSPECTION_DATE DESC) AS RANKING
FROM t3;
SELECT * FROM t4;

-- step5: final output
DROP TABLE IF EXISTS final_output;
CREATE TABLE final_output AS
SELECT
    PHA_NAME,
    MR_INSPECTION_DATE,
    MR_INSPECTION_COST,
    SECOND_MR_INSPECTION_DATE,
    SECOND_MR_INSPECTION_COST,
    CHANGE_IN_COST,
    PERCENT_CHANGE_IN_COST
FROM t4
WHERE RANKING = 1; -- filter out the first mr record of each PHA
```

Each step was carefully executed to meet the requirements of the analysis while filtering out unnecessary or redundant data.

Table 1 Heading rows for final PHAs inspection output.

PHA_NAME	MR_INSPECTION_DATE	MR_INSPECTION_COST	SECOND_MR_INSPECTION_DATE	SECOND_MR_INSPECTION_COST	CHANGE_IN_COST	PERCENT_CHANGE_IN_COST
Akron Metropolitan Housing Autho	2014-10-09	25593	2014-10-08	15626	9967	63.78
Alachua County	2015-01-22	37345	2014-05-01	17019	20326	119.43
Alaska Housing Finance Corporati	2014-11-14	26342	2014-11-13	21366	4976	23.29
Albany Housing Authority	2015-01-12	31115	2015-01-09	30247	868	2.87
Alexander County Housing Authori	2014-11-18	31272	2014-04-24	18855	12417	65.86
Alexandria Redevelopment & Housi	2014-05-09	29123	2014-04-18	14767	14356	97.22
ALLEGHENY COUNTY HOUSING AUTHORI	2015-02-02	37108	2015-02-02	36454	654	1.79
Allentown Housing Authority	2014-11-17	34040	2014-11-14	18989	15051	79.26
ALTOONA HOUSING AUTHORITY	2014-11-24	25750	2014-09-15	24813	937	3.78
ANNISTON HA	2014-12-30	31506	2014-08-21	10785	20721	192.13
Area Housing Commission	2013-06-25	28713	2013-06-24	19114	9599	50.22
Asbury Park Housing Authority	2014-06-03	35723	2014-05-21	14987	20736	138.36
Ashland Housing Authority	2014-04-29	29106	2014-04-29	17510	11596	66.23
ASHTABULA METROPOLITAN HOUSING A	2014-06-03	37948	2014-04-24	13920	24028	172.61
Athens Metropolitan Housing Auth	2014-05-22	21816	2014-05-21	10996	10820	98.40

We then created a csv file from wizard to export the final output for senior management to view the cost changes examples for each Public Housing Agency. Table 1 above is the heading rows of the output.

Conclusion

Through this project, we effectively utilized SQL window functions, specifically the LEAD() and RANK() functions, to extract relevant inspection data and calculate changes in cost. We filtered out unnecessary records to retain only PHAs with multiple inspections and focused on those with positive cost differences. The final

output dataset provides a clear view of which PHAs experienced an increase in inspection costs, offering valuable insights for senior management to evaluate cost trends and efficiency within PHA operations. Strategic decision-making can generated from our output work.

References

1. Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley.
2. MySQL. (2024). *Date and time functions*. MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_date-format