CS 6400 Database Project:
LEOFURN Sales Reporting System (LSRS)

Spring 2021

Project Overview

The purpose of this project is to analyze, specify, design, and implement a reporting system for the US branch of the Danish furniture company, LEOFURN. The project will proceed in three phases as outlined in the methodology for database development: Analysis & Specification; Design; and Implementation & Testing. The system will be implemented using a Database Management System (DBMS) that supports standard SQL queries.

LEOFURN Sales Reporting System

LEOFURN is a furniture manufacturer from Denmark, specializing in Scandinavian-style furniture, whose subsidiary has stores throughout the United States. Your team has been asked to design and build a prototype sales reporting system for LEOFURN USA. This section describes in detail the requirements for the LEOFURN Sales Reporting System (LSRS, pronounced like "lasers").

What makes LSRS different is that it is meant to import and aggregate sales data. Unlike *transactional* databases which are generally designed to record repetitive day-to-day business transactions (e.g., point of sale, buy and sell stock orders, online shopping carts, etc.), LSRS is meant for reporting and analysis over millions of records to support enterprise-wide decision making. As an example, a large online merchant like amazon.com or bestbuy.com relies on a transactional (also called *operational*) database system for recording customer orders and payments in real time. A data analyst tasked with generating a report that compares sales of a certain product among the different regions of the United States will typically query a reporting database for the report instead of accessing the transactional databases directly. There are several reasons for this: the system can store data from multiple transactional databases in a consolidated form, its schema is designed to support complex queries aggregating millions of rows, and queries do not impact the performance of the transactional database which must support high transaction throughput and availability.

For this project, you will design the database schema for LSRS and attach it to a rudimentary user interface. ***For this project you should create a normalized schema with as little redundancy as possible***. You also need not be concerned with the details of the transactional databases that we assume exist to support the point-of-sale system at each LEOFURN store. Instead, you will design the schema to support a consolidated view of the products offered and sold in all LEOFURN stores across the country. What follows is a description of the requirements for the system in terms of what information must be stored to support a set of reports defined by the LEOFURN executive team.

Data Requirements

LSRS maintains information about each *store*, including a unique *store number*, the store's *phone number*, and the store's *street address*. (You do not need to store individual components of the address, such as street number, prefix, etc., as the addresses will be unparsed.) Stores may also provide *childcare* while customers are shopping, have a *restaurant* on site, a *snack bar*, or any combination of these. You will not need to load any data regarding restaurant or snack bar sales. Stores offering childcare have an individually determined *limit* as to the maximum number of minutes of childcare offered per customer.

The limit is chosen by each store from predetermined values, and all limits may need to be updated if it changes (such as from 45 minutes to 60) or a new limit requires manual updating outside of a data load.

LSRS should also maintain information about each store's *city*, including the *city name*, the *state* in which the city is located, and the *population* of the city. A city may have multiple stores.

LSRS contains information about every *product* for sale at LEOFURN stores. Products have a numeric unique identifier (*PID*), similar to a UPC barcode, as well as the *name* of the product. Assume that all products are available and sold at all stores—that is, there is no need to specify that a certain product is only available at a certain store.

A product must be assigned to one or more *categories*. Each category has a *name*, which we assume to be unique. A category may or may not have products associated with it.

Every product has a retail *price*. The retail price is in effect unless there is a *discount price*. LSRS maintains the *discount date* and *discount price* of any product that goes on discount. If a product is discounted for multiple days in a row, then a record is stored for each day. It is possible that the same product is discounted multiple times (i.e., different days) with different prices. **If a product is discounted, it is for the same price in all stores—i.e., stores are not allowed to discount items independently or have store-specific discount prices.**

The LEOFURN executive team would like the ability to compare sales data on *holidays* versus non-holidays, so LSRS should maintain information about which dates contain holidays. The *name(s)* of the holiday should be stored. If a day has multiple holidays, their names can be combined, such as "Halloween, Harry Potter Day" as we only need to know that the date had holidays associated to it.

LSRS stores information about which products are *sold*, including the *store* where it is sold, the *date* of the sale, and the *quantity* of the product purchased. The total amount of the sale is not stored explicitly but can be derived based on the date purchased and the quantity, as individual item prices can be determined from either the retail price or the discount price if one is in effect. For reporting purposes sales tax values are ignored. Also, the system is not required to store which products were purchased together during a single sales transaction.

Advertising campaigns occur and are active for a specific set of dates. LEOFURN's executives want to have a rudimentary understanding on the impact of these advertising campaigns, so your system should store that information as well – this will include the *description* of the campaign (such as "Daily Prophet July Print Ad", "Triwizard Tournament Promotion Postcard", "Summer Radio Ads", and is unique) and each specific *date* it was active. Campaign dates may overlap.

The LEOFURN executive team understands that some details provided here may not be utilized in the reports they initially request but would like to have that data stored in the event they decided to enhance the system with more reports, or for any ad-hoc querying after you have built the system.

An important item to consider is that despite a "date" being a relatively simple piece of data, it may be optimal to treat dates as a set of values distinct from discount information, sale information, and campaign information, as discounts could occur on dates where there are no sales, sales could occur on dates where there are no campaigns, etc. Many of the reports requested can be simplified by referencing an established common set of dates instead of individual date information stored on each

record. (What this does not mean, though, is that you need to consider storing all dates that have ever occurred in your design – you will know what subset are needed based on the input data.)

LEOFURN's DBAs are working on an extract of sample data for you to test in your system, however, to avoid revealing confidential information, LEOFURN's data security team has directed them to use data from almost twenty years ago with only certain categories of data and refuse to allow newer data to be used. The data in question is currently housed in a cold storage vault in Greenland, so retrieving the data from tape backup and sanitizing it will not be possible until the spring thaw has occurred. This means it will be approximately two to three months before it can be made available to you. You will need to ensure that your schema design matches the data as described here so that any transformation prior to loading is kept to a minimum. Since this is a proof-of-concept for LSRS, you do not need to worry about any additional data loads beyond this sample dataset.

LSRS User Interface

All of your reports will be accessible from a "dashboard" UI that must be developed. Since this is a prototype version of the system, you do not need to concern yourself with configuring usernames or passwords to control access to the system.

There should be a main menu screen which can be used to access all functionality of the system that has been described in this specification. On this main menu, the following statistics should be displayed along with any buttons/links to reports or functionalities: the count of stores, count of stores that offer food (have a restaurant, a snack bar, or both), count of stores offering childcare, count of products, and count of distinct advertising campaigns. These statistics will be used to determine if the data loaded in LSRS is generally accurate before viewing reports.

In addition to the reports, there are some relatively simple interfaces you should design and provide as part of maintaining additional data for the system. First, you must provide an interface for holidays to be maintained by the user. This interface must allow for viewing and adding holiday information directly within the user interface. Second, your UI must allow for updating the population of any cities in the system, should a city's population change after data for it has been loaded.

LSRS Reports

LEOFURN's management has put your team in charge of developing the queries necessary to produce the following reports. Many of the reports have derived and/or aggregate data. These reports will be accessed with the user interface that you will create.

Some of the report queries are expensive to run given the large number of rows in the LSRS. Therefore, whenever possible you should include the filter conditions specified. For example, some reports ask for data from only a certain time period. If you leave off this filtering condition, the query will likely take a long time to return any results.

Report 1 – Category Report

For each category, including those without products, return the category name, total number of products in that category, the minimum regular retail price, the average regular retail price, and the maximum regular retail price of all the products in that category, sorted by category name ascending.

Report 2 – Actual versus Predicted Revenue for Couches and Sofas

LEOFURN executives want to predict whether offering items at a discount actually helps to increase revenue by encouraging a higher volume of sales. This report compares how much revenue was actually generated from a product's sales versus if the product were never discounted. After speaking with some marketing consultants, LEOFURN executives have learned that product discounts introduce on average a 25% increase in volume (quantity sold). Therefore, we assume that if an item that was offered at a discount were instead offered at the retail price, the quantity of items sold would be reduced by 25%. However, it is still possible that the predicted revenue would be higher since the reduced volume of products would be sold at a higher price per product. Initially, the executives are only interested in seeing the report for products in the "Couches and Sofas" category.

Here is a simple example:
Assume that Product Z has a retail price of $10. Assume that it was offered at a discount for on 6/1/2012 and 6/2/2012. Also assume the following transaction data for Product Z:

Date	Price	Quantity	Actual Revenue
5/1/2012	10.00	5	50.00
6/1/2012	8.00	10	80.00
6/2/2012	7.00	5	35.00
TOTALS		**20**	**$165.00**

Table 1 - Actual Revenue

The predicted revenue is calculated by assuming that the product is never offered at a discount and only 75% of the original quantity was actually sold on discounted days. Note that because this is just a predicted average, we assume that it is possible to sell a fraction of a product (e.g., 7.5 couches).

Date	Price	Quantity	Predicted Revenue
5/1/2012	10.00	5	50.00
6/1/2012	~~8.00~~ 10.00	10 * .75 = 7.5	75.00
6/2/2012	~~7.00~~ 10.00	5 * .75 = 3.75	37.50
TOTALS		**16.25**	**$162.50**

Table 2 - Predicted Revenue

In this example, the discounted prices resulted in slightly more revenue due to the higher volume of sales ($2.50 more).

Generate the following report: For each product in the Couches and Sofas category, return the product ID, the name of the product, the product's retail price, the total number of units ever sold, the total number of units sold at a discount (i.e., during discount days), the total number of units sold at retail price, the actual revenue collected from all the sales of the product, the predicted revenue had the product never been discounted (based on 75% volume selling at retail price), and the difference between the actual revenue and the predicted revenue. If the difference is a positive number, it means that the discounts worked in favor of LEOFURN because the predicted revenue is less than the actual revenue collected. If it is a negative number, it indicates that LEOFURN would have been better off not offering the product discounts. Only predicted revenue differences greater than $5000 (positive or negative) should be displayed and sorted in descending order.

Report 3 – Store Revenue by Year by State

This report shows the revenue collected by stores per state grouped by year. The states available for querying should be presented in a drop-down box. For example, the user would select "New York" and the system would show each store in New York state, show the store ID, store address, city name, sales year, and total revenue. Be sure the revenue calculation takes into account items that were sold at a discount. Sort the report first by year in ascending order and then by revenue in descending order.

Report 4 – Outdoor Furniture on Groundhog Day?

Some of the sales staff have noticed that outdoor furniture sales appear to spike on Groundhog Day (which falls on February 2 each year). They surmise that this is because customers begin thinking about the warm spring weather ahead. The LEOFURN marketing team would like to prove this, so they have requested the following report.

For each year, return the year, the total number of items sold that year in the outdoor furniture category, the average number of units sold per day (assume a year is exactly 365 days), and the total number of units sold on Groundhog Day (which is always February 2) of that year. Sort the report on the year in ascending order. The report will show if the total number of units sold on Groundhog Day each year is significantly higher than the average number of units sold per day. (This report does not imply storing Groundhog Day as a holiday, but to explicitly query for February 2.)

Report 5 – State with Highest Volume for each Category

LEOFURN management is planning to recognize all stores in the states that sell the greatest number of units for each category. They want to view this monthly, so the user interface must allow choosing a year and month from the available dates in the database before running the report. The report will return for each category: the category name, the state that sold the highest number of units in that category (i.e., include items sold by all stores in the state), and the number of units that were sold by stores in that state. This output shall be sorted by category name ascending. Note that each category will only be listed once unless two or more states tied for selling the highest number of units in that category. You may exclude from this report products which do not have a category.
This report can take a significant time to run, which may require tuned indices for the final implementation, but do not focus on their creation until the final phase.

Report 6 – Revenue by Population

To help forecast expansions into other cities, LEOFURN management would like to see what the total revenue is for specific population categories, and to see if there is a trend for growth, the revenue should be broken down on an annual basis. The categories for city size are: Small (population <3,700,000), Medium (population >=3,700,000 and <6,700,000), Large (population >=6,700,000 and <9,000,000) and Extra Large (population >=9,000,000). (While it may make sense to set a city's size category as part of the city's data, you'll need to update the category when the population is updated in the system as well, or your data will be inconsistent.) The team would like this in a tabular form, with city sizes as previously ordered as the columns on the report. Ensure that both rows and columns are arranged in ascending order (oldest to newest for years, smallest to largest for city size categories) so that no matter how it is formatted it is properly organized and understandable. (It would be confusing to have city sizes in the order of Large, Small, Medium, etc. or years ordered as 1999, 2004, 2002, etc.)

Report 7 – Childcare Sales Volume

LEOFURN's team has been trying to understand how offering childcare has an impact on sales. Some think that offering more childcare allows parents to spend more time browsing and purchase more

items, while others believe that the cost of providing childcare is not offset by the additional sales it may generate and should be discontinued. As current stores provide different levels of childcare (or none at all), they believe this report can help them determine its value.

This report should be based on the last 12 months' worth of available sales in the system, by each month, and presented in a tabular format, with row values for each month, and column values based on the grouped childcare time limit values for stores, showing the total sales by month and by childcare "category". (Stores that do not offer childcare should be included on this report and grouped as "No childcare".)

Report 8 – Restaurant Impact on Category Sales

At most of LEOFURN's stores, the restaurant is located before customers enter the main sales floor. Recent psychometric studies show that customers with a full stomach are less interested in purchasing certain kinds of items (such as dining room furniture, kitchen utensils, etc.) but show stronger interest in others (such as beds, couches, sofas, or reclining chairs). While understanding that the data loaded into LSRS does not allow for correlating individual customer restaurant purchases with other sales, the planning team would still like to know if the presence of a restaurant causes this behavioral trend. (This behavior does not apply to snack bars because those are near the exit of the store.)

In this report, for all sales data, for each category, take the total quantity of all products sold in the category and display separate totals for stores with a restaurant, and stores without. Order the report by category name ascending, and with non-restaurant store data listed first. You may exclude any categories that are not assigned products as their information would not be useful here. It should be presented as "grouped rows", grouped by category, like this:

Category	**Store Type**	**Quantity Sold**
Couches	Non-restaurant	14
	Restaurant	35
Pots and Pans	Non-restaurant	24
	Restaurant	4
Outdoor furniture	Non-restaurant	20
	Restaurant	21

Report 9 – Advertising Campaign Analysis

While there are numerous possibilities to correlate advertising campaigns and sales data, the LEOFURN team would like to start out with a relatively simple analysis: if a product is discounted, does an advertising campaign affect its sales volume?

For this report, you should query for all products, and when a discount price is in effect, and for each product, display the product ID, the product name, the total sold when no advertising campaign was active, and the total sold when any advertising campaign was active. Compute and also display the difference between the two totals and sort the results by difference in descending (highest to lowest) order. Only the top 10, followed by the bottom 10 from these results should be in the final report output. A sample of what the report should look like, with the top 1 and bottom 1 result is below.

Product ID	Product Name	Sold During Campaign	Sold Outside Campaign	Difference
3	Bokustinlar	5	2	3
252	Agleepoyo	6	25	-19

Document History

Version	Notes	Date
1.0	New version for Spring 2021	2/1/2021