

[15 hrs] BUILD: Practicum II (Spring 2024) / Mine a Database

Start Assignment

- Due Wednesday by 11:59pm
- Points 100
- Submitting a file upload
- File Types zip
- Available until Apr 20 at 11:59pm

Nota Bene

Read this entire page twice before you get started.

The due date is **April 17 at 11:59pm ET**. Late submissions are accepted (with the usual penalty of 2.5% per day late) until April 20 at 11:59pm ET. No submissions are accepted after that as it would not give us enough time to do a viva voce and grade your work.

Work on the practicum for at least three hours every day and use the time during the week prior to the practicum to start working on it, especially the configuration of the MySQL Server and the loading of the data.

A gentle reminder that the average of both practicums must be above 70% to pass this course, so be sure to complete on time and seek help right away. Do not procrastinate -- things that appear simple often take more time than expected and, of course, programming is fraught with potholes on the road to success. So plan accordingly.

The average time to complete the practicum is 15-25 hours. Do not wait to start. Seek help early. Submit often and as soon as you have enough code that works. We will only grade the last submission. Check your submission before you submit and after.

You can earn "bonus points" on this assignment. That means that the maximum score is above 100 but we still grade out of 100, which means you can get additional points towards your final grade.

Motivation

In this practicum you will extract data from an XML document and then store the data relationally in a SQLite database. That database represents a "transactional" database. Then you will extract data from the transactional database and create an "analytical" database using a star schema in MySQL. Finally, you will query facts from the MySQL analytical database. This will require that you connect to two different databases simultaneously -- a common occurrence in practice.


Format


Method: Individual (no collaboration)

Materials: RStudio or posit.cloud; SQLite; MySQL on cloud service such as *freemysqlhosting.net* or *db4free.net*

In Part 1 you create a normalized relational OLTP database and populate it with data from an XML document. In Part 2 you will add to the normalized schema fact tables and turn the normalized schema into a denormalized schema suitable for OLAP. In Part 3 you'll use the OLAP star/snowflake schema to do some (simple) data mining. The parenthesis contain the maximum number of points that can be earned for that question and an estimate of the time in hours.

Part 1 (55 pts) Load XML Data

1. (0 pts / 0.1 hrs) Create an R Project named `CS5200.PraciticumII.LastNameF` (where *LastName* is your last name and *F* is your first initial, e.g., `CS5200.PraciticumII.WuX` and then within that project create an R Script (.R file, not a .Rmd Notebook) called `LoadXML2DB.LastNameF.R` for Part 1; Parts 2 and 3 will be done in different programs.
2. (0 pts / 0.5 hrs) Download and then uncompress the zip file ([pharma-sales-v2.zip](https://s3.us-east-2.amazonaws.com/artificium.us/datasets/pharma-sales-v2-F23.zip) ) (<https://s3.us-east-2.amazonaws.com/artificium.us/datasets/pharma-sales-v2-F23.zip>) containing several XML files:

Save the XML files locally in a subfolder named `txn-xml` inside your project folder, and then inspect the files to familiarize yourself with their content and structure. The file names contain the number of "records" in them; you may wish to only load one or two of the small files for development, e.g., [pharmaSalesTxn-20-A.xml](https://s3.console.aws.amazon.com/s3/object/artificium.us?region=us-east-2&prefix=datasets/pharmaSalesTxn-20-A.xml) ) (<https://s3.console.aws.amazon.com/s3/object/artificium.us?region=us-east-2&prefix=datasets/pharmaSalesTxn-20-A.xml>).

The XML *pharmaResp.xml* is a list of sales representatives for a pharmaceutical company. The other XML files contain sales transactions.

To download and save a zip file, open the context menu by (generally) right-clicking on the link and selecting "Save As..." or "Save link as..." or a similar choice depending on your browser. Do not click on the link as that would cause the browser to attempt to open the file which would fail.

3. (5 pts / 1 hr) Create a normalized relational schema that contains the following entities/tables:

products: the products being sold by the pharma company, e.g., Alaraphosol

reps: the sales reps and their info and territory, e.g., EMEA

customers: the customer buying products with their country, e.g., USA

sales: sales transactions of products

Use the XML document to determine the appropriate attributes (fields/columns) for the tables. Create appropriate primary and foreign keys. Where necessary, add synthetic surrogate keys. Store all information necessary for Parts 2 and 3. The choice of table structure is up to you. You may wish to visualize the schema using an ERD, although you do not have to submit the ERD.


4. (10 pts / 1 hr) Realize the relational schema in SQLite (use CREATE TABLE statements using R functions; you cannot use {sql} chunks in a R Script). You may **not use MySQL** for this step.
5. (5 pts / 2 hrs) Load (without validation as there are no DTDs) all XML files from the *txn-xml* folder into R. You may want to start with one file for testing, but eventually you must load all files in the *txn-xml* folder keeping in mind that files can be added or removed so you cannot rely on fixed files names, but you can rely on the data for the sales reps to be in the file with the pattern "pharmaReps*.xml" and all transactions to be in files following the pattern "pharmaSalesTxn*.xml". Rather than loading all files at once, you may wish to load them one by one and add them to the database one at a time. So read the next question before doing this question as you may want to combine the design of the two questions.
6. (30 pts / 5 hrs) Extract and transform the data from the XML files in the folder and then save the data into the appropriate tables in the database. You may use any strategy of your choice and any combination of functions of your choice, i.e., `xmlToDataFrame`, node-by-node tree traversal, and

XPath. Efficiency becomes a consideration when loading large amounts of data, so pay attention to your choice of algorithm and functions.

For dates, you need to devise a conversion scheme, document your decision, and convert all dates to your chosen encoding scheme.

7. (5 pts / 0 hrs) Practice good coding: write R functions as needed; structure your code to be readable and maintainable; use proper programming practices, thoroughly comment your code; add your name, course information, date to a header in your script.

Part 2 (25 pts) Create Star/Snowflake Schema

1. (0 pts / 0.1 hrs) Create a new R Script for Part 2 named *LoadOLAP.LastNameF.R*.
2. (5 pts / 1 hr) Create a **new MySQL** database on *freemysqlhosting.net* (recommended), *db4free.net*, or any other cloud MySQL provider. Be mindful that *freemysqlhosting.net* imposes a 5MB limit, and exceeding that limit may cause your account to be locked. So, pay attention to the volume of data and only load what can be accommodated. You may use the database you created for Practicum I. Connect to the database. If you use SQLite for this step, no credit will be awarded for this question.
3. (20 pts / 5 hrs) Create and populate an analytical database based on a star schema with fact tables as described below. Load the data from the SQLite Database created in Part 1 and populate the fact tables via R. Do not load the entire tables from SQLite into R as that does not scale. Use SQL to get the data you need. Note that there is not a single way to create the fact tables -- you may use dimension tables or you may collapse the dimensions into the fact table. Remember that the goal of fact tables is to make interactive analytical queries fast through pre-computation and storage -- more storage but better performance. This requires thinking and creativity -- there is not a single best solution. Make sure your code scales -- imagine that your transactional database (the SQLite database) contains hundreds of millions of rows per table -- would your code still work and still performance well? ETL code is not always the fastest but it must scale. This [chalk talk](https://youtu.be/H6no78vdNUM)  (<https://youtu.be/H6no78vdNUM>) on how to build fact tables and why might help...
 - A. (10 pts / 2 hrs) Create and populate a fact table called "*product_facts*" that contains the

product names, the total amount sold of each product, the total amount sold of each product each quarter and year for which there is data, and the total units for each product per region.

- B. (10 pts / 2 hrs) Create and populate a fact table called "*rep_facts*" that contains the name of the sales rep, the total and average amount sold for each quarter and each year for which there is data.

Your star schema must support analytical queries such as these:

What is the total sold for each quarter of 2021 for 'Alaraphosol'?

Which sales rep sold the most in 2022?

How many units were sold in EMEA in 2022 for 'Alaraphosol'?

This are the fact tables you will need for Part 3.2. Of course, the years in the examples above are arbitrary and other analytical queries are possible, but these are examples to help you define your fact tables.

Part 3 (40 pts) Explore and Mine Data

1. (0 pts / 0.5 hrs) Create an R Notebook named *AnalyzeData.LastNameF.Rmd* within your R Project (use same naming as before if you are in a group).
2. (40 pts / 4 hrs) In the notebook, use markdown to write a "report" which shows the results of the following analytical queries against your MySQL data warehouse from Part 2 (which might go to some manager as part of a weekly report):

Analytical Query I: *Top five sales reps with the most sales broken down by year. So, for each year for which there is data, the top five reps for that year. Think about how you would best present this information so it is easily usable by an analyst or manager.*

Analytical Query II: *Total sold per product per quarter. Think about how to best display this information.*



Analytical Query III: *Number of units sold per product per region. Show the result in a line graph visualization.*

Analytical Query IV: Average sales per sales rep over the years. Show the result in a line graph visualization.

Of course, be sure that your report uses the data from the fact tables.

If you use SQLite for this step, maximum half the credit will be awarded for this question. Each of the analytics use cases is equally weighted. Thoughtfulness and creativity do count and we want you to think about your presentation and report. Apply yourself a bit.

Hints

- see the hints from Practicum I -- many apply here as well
- if you cannot connect to MySQL, be sure that you do not have firewall software running that blocks port 3306
- remember that *db4free* cannot write data with `dbWriteTable()`; see Practicum I on tutorials for bulk loading or use *freemysqlhosting.net* instead
- hashmaps can speed up look-ups (they are a kind of index); R doesn't directly support them, but they can be "faked": <https://www.r-bloggers.com/2015/03/hash-table-performance-in-r-part-i>  [_ \(https://www.r-bloggers.com/2015/03/hash-table-performance-in-r-part-i\)](https://www.r-bloggers.com/2015/03/hash-table-performance-in-r-part-i) or investigate the functions from the package **r2r** which implements hashmaps
- saving large data frames and other objects can save time... look up how to save objects in binary: <https://rstudio-education.github.io/hopr/dataio.html>  [_ \(https://rstudio-education.github.io/hopr/dataio.html\)](https://rstudio-education.github.io/hopr/dataio.html)
- use a smaller XML file for development; it'll speed things up during dev
- you may use AI assistants as needed but let us know which ones you used and for what; add acknowledgments in your code and you must be able to explain any code you copied or borrowed

Submission

- Submit all files: *.R*, *.Rmd* in a *zip* file.
- Do not submit the XML files; we will have our own set for testing.
- Your code must run.
- Do not use absolute paths for your XML files or the database; make all paths relative to your

project folder.

- Do not do any work or data shaping outside of R.
 - Score is out of 100, so one can get over 100%.
-

Your code has to run, obviously, but it also has to run somewhat efficiently... if everyone else's code runs in 10-30 minutes but yours takes several hours then clearly is due to poor programming and not due to the inherent complexity of the problem... follow common coding strategies for writing efficient code such as factoring out invariants from loops, not calling functions repeatedly, pre-allocating memory, not copying objects needlessly, not calling expensive functions when simpler ones will do (e.g., call `substring()` instead of doing regular expressions), use `which()` when searching and don't use `sqldf`, and so forth. These practices are not specific to R, although there are R specific performance issues, but those are less likely to be a concern here.