Practical tasks: SQL Querying

Submission format

The submission for this whole assignment should be a **link** to either a PDF (pre-uploaded to OneDrive or Google Drive and ***shared*** *for anybody having the link to be able to view*), or a link to a ***public*** GitHub repository containing SQL scripts for code and markdown files for solution descriptions. The second option is recommended, yet not required.

AI Usage policy

In 2025, it is clear that to succeed in IT it is needed to *both* have the hard skills in an IT area *and* be able to apply AI effectively to solve the problems within that area, doubling human productivity. This implies that you should not neglect standard training and learning to do stuff on your own if needed, but at the same time you should gradually let AI into what you are building, and learn to do it in a consistent, reliable manner.
Hence, the practical assignments in the current course do not require you to abandon AI, but rather use it wisely. I suggest you do the things below.

Learning to prompt

If you are using a chatbot to come up with a solution rather than coding a solution from scratch on your own, use this as an opportunity to level up your prompting skills. In particular, strive to make the chatbot produce a working solution from the first try. While achieving this goal, you will gain the intuition of how much context you need to expose to the chatbot so it could produce a syntactically correct query for your database and SQL dialect. Too much context = waste of your typing time and environmental resources (although energy-wise per conversation things are getting better). Too little context and LLM has to guess some important things that it should not.
Also, by improving your prompting skills you are likely to improve your general technical communication skills as well.

Socratic questioning

One of the recommended LLM-powered learning techniques is to ask the chatbot to be your Socratic tutor (see this and this). Give the LLM the necessary context about your database and SQL dialect and the problem, but do not ask for a complete solution right away. Instead, use this template for your prompt: `Act as an expert mentor and database analyst. Ask me a series of questions in such a way that by answering these questions I will come up with a solution myself. Do not be overly supportive by revealing the pieces of the solution too early on, but if you find me to be on the wrong track, provide some hints on SQL keywords or database functions that could help me achieve the goal, giving away as little information as possible at first.`

Follow-up questions

Even if you strive to solve each and every problem on your own, after you do so, submit your code to an advanced chatbot asking for *critique* or *alternative solutions*. You will often be rewarded with cool insights and alternative pathways. Clearly, asking follow-up questions is encouraged even for the solutions that LLM itself has produced.

Databases

The tasks concern World and AdventureWorks educational databases that we've already worked with. All the tasks can be solved in SQLite, MySQL and PostgreSQL; so use the version of the database for your favorite DBMS. When using LLMs it is recommended to provide them from the very start with the SQL dialect and the DDL definitions for the table(s) relevant for the partitular task.

Tasks

Task 1

Database: world
Task description:

- determine which Latin letter does NOT appear in the first place of the three-letter country code in the Country table;
- same for letter not appearing in the second place;
- same for the third place.

Deliverables:

- query (or multiple queries) that you've used to answer the task question(s);
- answer(s) to the question(s).

Task 2

Database: world
Task description:

- determine the ratio of population per square mile for each country;
- using CTE (common table expression) and the result of the previous step, determine the maximum, minimum and median of the ratio mentioned in the previous step;
- challenge yourself: using UNION or otherwise write a query such that the result set for this query has two columns "Metric" and "Value" and contains exactly three lines with maximum, minimum and median values in the "Value" column, and the metic names (i.e. literally 'Minimum', 'Maximum' and 'Median') in the "Metric" column.

Deliverables:

- query (or multiple queries) that you've used to answer the task question(s);
- answer(s) to the question(s).

Task 3

Database: world
Task description:

- For each country determine the percentage of its population living in its capital. Which 10 countries have this percentage being the smallest?

Deliverables:

- query (or multiple queries) that you've used to answer the task question(s);
- answer(s) to the question(s).

Task 4

Database: world

Task description:

- Determine the average life epectancy for each language in a simple (and not really a correct) way: for each language just average out the life expectancy of all countries where this language is spoken.
- What would be a more correct way to compute average life expectancies per language? Challenge yourself by coming up with the solution that is as accurate as possible given the data available in the database.

Deliverables:

- query (or multiple queries) that you've used to answer the task question(s);
- answer(s) to the question(s) with plain text description of the strategy for the second question.

Task 5

Database: AdventureWorks
Task description:

- How many people were employed in each department on May 01, 1999?
- Use *employeedepartmenthistory* table StartDate and EndDate.

Deliverables:

- query (or multiple queries) that you've used to answer the task question(s);
- answer(s) to the question(s).

Task 6

Database: AdventureWorks
Task description:

- At what date have the USD<->CAD exchanged rate has changed the most (compared to the previous day)?
- Use *currencyrate* table only. You can use the *lag* window function, CTE and subquerying, or anything else if needed.

Deliverables:

- query (or multiple queries) that you've used to answer the task question(s);
- answer(s) to the question(s).

Task 7

Database: AdventureWorks
Task description:

- Use an LLM to craft a *recursive query* (or *recursive CTE*) over the *employee* table to obtain a table storing subordination chains. A subordination chain is a string like "A -> B -> C -> …" where A,B,C etc are IDs of employees, A is a manager of B, B is a manager of C and so on. It is fine if for every chain its subchains are also in the result set, but you can also think on how to deduplicate that. Note that we have not covered recursive queries in out practice sessions, so feel free to look this up in the DBMS documentation and/or ask an LLM about it.

Deliverables:

- SQL query with commentary.

Task 8

Database: AdventureWorks
Task description:

- Investigate if and how *vendor*'s CreditRating correlates with the total monetary amount of transactions (see *PurchaseOrders*) with that vendor.

Deliverables:

- One-two paragraph analysis with related SQL queries included.

Task 9

Database: AdventureWorks
Task description:

- Investigate if and how employee's pay rate correlates with age, gender and marital status.

Deliverables:

- One-two paragraph analysis with related SQL queries included.

Task 10 [optional, ungraded]

If you've used LLMs for any parts of the assignment, share your findings:

- What typical mistakes did LLM make related to SQL development, were there hallucinations?
- How much context did LLM need to reliably come up with a solution?
- Was LLM as effective in solving analytical type of questions (tasks 8 and 9) as for more technical questions (tasks 1-7)?
- What advice to your future self or your colleagues would you give regarding using LLMs for SQL analytics?