EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced SQL
Winter Term 2025/2026
Prof. Torsten Grust, Louisa Flüchter
WSI — Database Systems Research Group

Assignment 1

Hand in this assignment until **Friday, 31 October 2025** at the latest.

🤔 Running out of ideas?
Are you hitting a roadblock? Are some of the exercises unclear? Do you just need that one hint to get the ball rolling? Refer to the `#forum` 🌐 channel on our Discord server—maybe you'll find just the help you need.

💡 Rules for this and all future assignments
- In general, the only acceptable file format is plain text (`*.md`, `*.sql` for SQL code). Files in other formats are not graded, unless explicitly stated differently.
- All code you submit must run on an empty database or with the task-specific data pre-loaded. Code that does not run without errors might not be graded.
- Please submit code that is nicely and consistently formatted and well-documented[1].

Ⓔxam-style Exercises
Exercises marked with Ⓔ are similar in style to those you will find in the exam. You can use these to hone your expectations and gauge your skills.

Task 1: The Knight's moves Ⓔ

Formulate a SQL query `Q` that computes all possible `x`/`y` chess board positions for the knight pieces in table `knights`.

```sql
CREATE TABLE knights (
  piece text,
  x     int,
  y     int
);
```

```sql
INSERT INTO knights(piece, x, y) VALUES
  ('♘', 2, 3),
  ('♞', 4, 4);
```

In chess, knights move in a certain pattern[2]: a knight may move two squares vertically and one square horizontally, or vice-versa.

```
 |1|2|3|4|5|6|7|8
8| | | | | | | |
7| | | | | | | |
6| | |0| |0| | |
5| |0| | | |0| |
4| | | |♞| | | |
3| |0| | | |0| |
2| | |0| |0| | |
1| | | | | | | |
```

Example: Positions reachable by ♞ with one move are marked with 0

For a board position to be valid, `x` and `y` both need to be in the range `1,...,8`. `Q` should return rows with row type `row(piece text, x int, y int)`. Your solution should comprise 14 rows.

💡 Note
In case you run into problems with the unicode chess piece characters in `chess.sql`, simply replace them: ♔ with `'k'`, ♚ with `'K'`, ♘ with `'n'` and ♞ with `'N'`.

[1] To have an idea of "nicely formatted code", you can find a short style guide here: https://www.sqlstyle.guide 🌐.
[2] For details about chess, see https://en.wikipedia.org/wiki/Chess 🌐.

Task 2: Get to know the DuckDB CLI

The DuckDB and its Command Line Interface (CLI) offer a large variety of functionality. Explore the documentation[3] to fulfill the following tasks.

A Write SQL DDL statements (`CREATE TABLE` and `INSERT INTO`) to represent the chess board from Task 1. Change the output mode[4] of DuckDB such that the full output (including table header) of query `TABLE chess_board;` looks exactly like the example.

B You might have noticed that DuckDB opens with the message:

```
Connected to a transient in-memory database.
```

If you exit DuckDB (`.quit` or `.q` for short) and re-open it, you will see that your `chess_board` and `knights` tables have disappeared:

```
Error: Catalog Error: Table with name chess_board does not exist!
```

Create a database file to persist your data and store the `chess_board` and `knights` tables inside. Check the dot command `.open`[5] and/or the for overview page[6] of the CLI to find out more about opening persistent databases.

[3] DuckDB documentation: https://duckdb.org/docs/index
[4] output formats: https://duckdb.org/docs/api/cli/output_formats
[5] dot commands: https://duckdb.org/docs/api/cli/dot_commands
[6] CLI overview: https://duckdb.org/docs/api/cli/overview