Handing In

Upload your homework to gradescope. Please write your Banner ID on your submission. Do not write your name on the submission.

Grading Information

Grading for the homeworks works as follows:

- The set of warm-up problems will be graded as one of (✓+, ✓, ✓-)
- All other problems will be graded in detail and will be given a score.

Solutions for the warmup problems will be provided along with your graded work.

Warmup #1

Consider the following schema and information:

\[
\text{person}(id, \text{first}\_\text{name}, \text{last}\_\text{name}, \text{city}, \text{age}) \\
\text{farmer}(id, \text{work}\_\text{city}, \text{income}) \\
\text{eats}(id, \text{vegetable}\_\text{name})
\]

- Person IDs are integers
- All names, cities, and colors are strings
- Ages and income are non-negative integers

1. Please write out the DDL (Data Definition Language) for creating the above schema in SQL. Make sure to include primary keys, foreign keys, and constraints.
**Problem 2 (To Be Graded)**

Consider the following set of functional dependencies $F$ for schema $R = (A, B, C, D, E)$:

\[
F = \{ \\
AD \rightarrow BC \\
AC \rightarrow E \\
B \rightarrow D \\
E \rightarrow D \\
\}
\]

1. List the set of functional dependencies in $F_c$, the canonical cover.

2. Identify and list all candidate keys.

3. Write a schema in BCNF for relation $R$ if one such exists. State whether your schema is dependency preserving. If not, explain why it is not.

**Problem 3 (To Be Graded)**

Consider the following schema that models chess.

For more info on chess check out this link [link]

\[\text{player}(\text{player_id, age, wins, losses})\]\n\[\text{match}(\text{match_id, winner_id, loser_id, date})\]\n\[\text{move}(\text{match_id, move_number, moving_piece, captured_piece, is_check_mate, new_location_x, new_location_y})\]

Column Definitions:

<table>
<thead>
<tr>
<th>player_id</th>
<th>Unique ID for a player</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>Player age</td>
</tr>
<tr>
<td>wins</td>
<td>Number of player lifetime wins</td>
</tr>
<tr>
<td>losses</td>
<td>Number of player lifetime losses</td>
</tr>
<tr>
<td>match_id</td>
<td>Unique ID for a match</td>
</tr>
<tr>
<td>winner_id</td>
<td>Player ID of Winner</td>
</tr>
<tr>
<td>loser_id</td>
<td>Player ID of Winner</td>
</tr>
<tr>
<td>date</td>
<td>Match Date</td>
</tr>
<tr>
<td>move_number</td>
<td>Monotonically increasing move number beginning at 0</td>
</tr>
<tr>
<td>moving_piece</td>
<td>name of piece being moved</td>
</tr>
<tr>
<td>captured_piece</td>
<td>name of piece being captured OR NULL if no captured piece</td>
</tr>
<tr>
<td>is_check_mate</td>
<td>Boolean, TRUE if check mate, FALSE if not</td>
</tr>
<tr>
<td>new_location_x</td>
<td>x location of new piece (indexed 0)</td>
</tr>
<tr>
<td>new_location_y</td>
<td>y location of new piece (indexed 0)</td>
</tr>
</tbody>
</table>

1. Write a SQL query that returns player_ids for players who have won at least 1 game by moving their own king to force check_mate and who have never lost to the same move.