PART 3: Simple Retrieval Queries

1. Get the Titles of all books by <Author>

select Title from Titles where Author= "<Author>";

2. Get Serial numbers of all books by <Author>

select Serial from Inventory natural join Titles where Author="<Author>";

3. Get the Titles of all books checked out by <Patron's name>

select Title from Titles natural join Inventory natural join CheckedOut natural join Patrons where Name="<Patron's name>";

4. Get phone number(s) of anyone with <Title> checked out

select Phone from Phones natural join Patrons natural join CheckedOut natural join Inventory natural join Titles where Title="<Title> ";

Part 4 - Intermediate Retrieval Queries

1. Find the Titles of the library's oldest <N> books. Assume the lowest serial number is the oldest book.

select Title from Titles natural join Inventory order by Serial limit <N>;

2. Find the name of the person who has checked out the most recent book. Assume higher serial numbers are newer. Note that this query is not concerned with the absolute highest serial number, it is concerned with the highest one that has been checked out.

select Name from Patrons natural join (select CardNum from CheckedOut order by Serial desc limit 1) as max;

3. Find the phone number(s) of anyone who has not checked out any books. If a phone number belongs to two Patrons where one of them could have checked out a book, then that phone number should not be included in the output.

select Phone from Phones except (select h.Phone from Phones h where exists (select p.CardNum from Patrons p where exists (select CardNum from CheckedOut c where c.CardNum = p.CardNum and h.CardNum = c.CardNum)));

4. The library wants to expand the number of unique selections in its inventory, thus, it must know the ISBN and Title of all books that it owns at least one copy of. Create a query that will return the ISBN and Title of every book in the library, but will not return the same book twice

select distinct ISBN, Title from Titles;

<u>Part 5 - Chess Queries</u>

Be careful with your queries in this database This one has a lot of data, and a poorly formed query could easily overwhelm the server. If you repeatedly run very expensive queries, your account will be locked.

Note that the schemas in this database have some additional columns that were not shown in Lab 3, and the Result column in the Games table is encoded as 'W', 'B', or 'D', for white winning, black winning, or draw. Also note that names are given as "last, first", so Magnus Carlsen is listed as "Carlsen, Magnus". Feel free to run "describe" commands on the tables in the Chess database.

Provide SQL queries for the six relational queries in Lab 3, Part 2.

Killing Queries

If properly formed, these queries should all run in a few seconds, including the network transfer time. The computational parts should all run in a fraction of a second, assuming nobody else is overwhelming the server. To kill a query that is taking too long, the simplest approach is to press ctrl-c, which will disconnect you. A better approach is to open another mysql client in another terminal and run "show processlist;". Find the ID of your process, and run "kill query <id>;"

Hints

Be careful with natural join - both Players and Events have a column called "Name", which have different meanings (but SQL doesn't know this). Trying to natural join any temporary relation with those two columns will not have the desired result, even if the relations do both have "eID". To be safe, don't use natural join.
Be careful when mixing "and" and "or" in boolean conditions. It's always safest to be explicit with parenthesis. For example, (false and false or true) is not the same as (false and (false or true)). Getting this wrong can result in very expensive queries.

1. Find the names of any player with an Elo rating of 2850 or higher.

select Name from Players where Elo >= 2850;

2. Find the names of any player who has ever played a game as white.

```
select distinct Name from Players join Games where Games.WhitePlayer = Players.pID;
```

3. Find the names of any player who has ever won a game as white.

```
select distinct Name from Players join Games where (Games.WhitePlayer = Players.pID)
and (Games.Result = 'W');
```

4. Find the names of any player who played any games in 2018.

```
select distinct Name from (select BlackPlayer,WhitePlayer from (select eID from Events
where Date >= '2018-01-01' and Date <= '2018-12-31') as e join Games on e.eID =
Games.eID) as g join Players on g.BlackPlayer = Players.pID or g.WhitePlayer =
Players.pID;
```

5. Find the names and dates of any event in which Magnus Carlsen lost a game.

```
select distinct Name, Date from (select pID from Players where Name="Carlsen,
Magnus") as p join Games on ((p.pID = Games.BlackPlayer and Games.Result = 'W') or
(p.pID = Games.WhitePlayer and Games.Result = 'B')) join Events on Events.eID =
Games.eID;
```

6. Find the names of all opponents of Magnus Carlsen. An opponent is someone who he has played a game against. Hint: Both Magnus and his opponents could play as white or black.

```
select Name from (select pID from Players where Name="Carlsen, Magnus") as p join
Games on (p.pID = Games.WhitePlayer or p.pID = Games.WhitePlayer) join Players where
((Players.pID = Games.WhitePlayer or Players.pID = Games.BlackPlayer) and
Players.Name != 'Carlsen, Magnus')
```