Bailey Thompson
20653683
ECE 356
Project 2
Part 2 - Chess

From Kaggle, we have a CSV, and if we put this into a table with each column of the CSV being an attribute, we would get the following table:

Table 1. Not in NF form.

Main
gameID
rated
createdAt
lastMoveAt
turns
victoryStatus
winner
incrementCode
whiteID
whiteRating
blackID
blackRating
moves
openingEco
openingName
openingPly

For 1NF, each attribute must be atomic. From Table 1, we see that incrementCode is not atomic because it is two attributes combined in a format such as “10+5”. The first attribute is the number of starting minutes for each player, and the second one is the seconds added to the player’s clock each ply. We can then separate this attribute to timePerPlayerMinute and incrementSec. Additionally, the moves attribute is not atomic because it contains every move in the whole game. We can separate this out into a move attribute for a single ply, and a turn attribute specifying which ply that move attributes is associated with. This causes a lot of data duplication but will be sorted out in 2NF. The table then looks like:

Table 2. In 1NF form.

Main
gameID
rated
createdAt
lastMoveAt
turns
victoryStatus
winner
timePerPlayerMin
incrementSec
whiteID
whiteRating
blackID
blackRating
move
turn
openingEco
openingName
openingPly

For 2NF, the data must be in 1NF and also have no partial dependencies. The primary key is gameID, so this means that an attribute cannot depend on only one part of the candidate key. However, a partial dependency would not be possible since there is only one attribute that is the candidate key. Thus, Table 2 is in 2NF.

For 3NF, the data must be in 1NF and also have no transitive dependencies. This means that all attributes must depend entirely and only on the candidate key. When checking the dependency of each attribute, we see that whiteRating and blackRating depends on the whiteID and blackID in addition to the gameID. We also see that turn depends on move as well as gameID. Everything else depends only on gameID. The three opening attribute (openingEco, openingName, and openingPly) also only depend on gameID because the openingEco doesn't tell us what the openingName will be, and vice versa, and the openingPly number doesn't strictly rely on the type of opening. This gives us the following tables:

Table 3. In 3NF form.

Main	Rating	Ply
gameID	**gameID**	**gameID**
rated	**playerID**	**turn**
createdAt	rating	move
lastMoveAt		
turns		
victoryStatus		
winner		
timePerPlayerMinute		
incrementSec		
whiteID		
blackID		
openingEco		
openingName		
openingPly		

For BCNF form, we assure that the data is in 3NF form, and also make sure that it is only possible to derive non-candidate keys from candidate keys and not vice versa. This is the case, so Table 3 is also in 3NF form.

In Table 3, primary keys are in bold, and the foreign keys for each table are the following:

1. The Main table has no foreign keys.
2. The Rating table references the Main table's gameID attribute since it's only possible to have a rating if the game exists.
3. The Ply table references the Main table's gameID attribute since it's only possible to have a rating if the game exists.

I made a Java program to clean up the CSV and create insert statements out of it and have included it in the zipped folder.