Code Walkthrough

Summary:

I tried to use as much Pandas in this code as possible. Seeing as I would be working with multiple CSVs, and because of Pandas speed and vectorization abilities, it seemed like an appropriate choice. I also used SQLAlchemy for database interaction, because of how well it fits with Pandas dataframes.

For the output, I decided to go with a long skinny file, unique by event_id, play_id, and player_id. While not very pretty to look at and using lots of rows, it feels very intuitive for leveraging with other datasets to do calculations. For example, calculating Steph Curry's plus/minus would be very easy.

-From my dataset, pull all rows where player_id = Steph's player_id
-Use that list of plays , or everything but those plays to pull net changes while Steph is on/off the court

Improvements:
-Less looping, more vectorization
-Investigate TJ Warren edge case and find a more definitive solution (see best_guess_starters() at the end for details.)
-Currently outputs every game to one table. Due to nature of output, this could take up a lot of s pace very quickly. Might make sense to output one table per game.

Walkthrough:

Main():
140-146:
-Set up SQL database connection using SQLAlchemy, passing in a credentials file for database connection credentials

- I chose to use SQLAlchemy due to its compatibility with Pandas Datafarmes. At the end of the coude, you will see a simple one-liner to add the entire dataframe of players on the court

162-169:
-Use Pandas to read in CSV inputs and convert to dataframes.

167-174:
-Sort dataframes to ensure correct play_id order, and take only necessary columns.

174:
-Make empty dataframe to iteratively append to

176-177:

-Split general play-by-play data in to groups according to game (event_id) and begin looping over games.

181-182:

-Within each game present, split up according to quarter and begin looping.

-Quarterly processing was necessary due to the fact that, in the NBA, lineups can completely change from the end of one quarter to the beginning of another, without any substitution events showing up.

184:

-Also make subset of player specific play-by-play data.

-Initially, I was just looping through the player-specific data. However, there are instances where plays were repeated due to two or more players being involved. Instead of coding up exceptions to handle that, I decided to loop through the general play-by-play, and access the player-specific file when needed.

186:

-Make a dataframe listing the starers for that quarter and game.

187:

-Pass the starters dataframe to get_active_players() function, returning which players were on the court during that quarter of that game.

189:

- Add that quarter/game's information to the dataframe created on line174.

192:

-Use to_sql() method of Pandas and SQLAlchemy engine to export dataframe to our database

192-195:

-Example of reading in the pbp_players_on_court table and exporting to a CSV file

get_starting_lineup():

30-31:

-This gets the starting lineup for the first quarter. This was made very easy by starting lineups corresponding to play_event = 0.

32-49:

-Getting starting lineups for other quarters was trickier, but here was my logic:

-Make two empty checker lists: starters, non-starters
-Take all of the players that logged any plays in that quarterly

        - Take all of the substitution events in the quarterly
        - Looping sequentially through all of the substitutions by play_id:
            - Check our two checker lists to see if current player was already accounted for
            - If that player was not accounted for, check the play sequence to find out what list they should be added to
                - Whoever was being subbed in could not have started
                - Player being subbed out must have started
                - Add to corresponding list, and remove from list of all players who logged plays in the quarter

52-68:

- There are cases where players play an entire quarter without being subbed out. In that case, the starters list would be less than 10. The player would also not show up in either the starters list or the non-starters list.
- Because we removed players as we found them in the step above, we can figure out that anyone who is still on that list must have played the entire quarter. We can add them to the starters list

NOTE:

- There was one edge case: T.J Warren in the 4th quarter of the Suns vs Rockets game:
    - T.J Warren was subbed out in the 3rd quarter, and never subbed back in. However, he did receive a technical foul which showed up in the play-by-play. Thus, T.J Warren made a play without being on the court.
    - This made adding unaccounted for players incorrect. I made a function to make the best guess possible on which player should be added, but also included warnings that the resulting lineup is not definitive, and there are rare cases in which it would be incorrect.

- Return a dataframe containing all starters for that quarterly

get_active_players():

84-108:

- Take in starters dataframe
- Append starters to players on court dataframe
- Take all plays after the starting lineup for Q1, and after the starting period play for other quarters
- If the play was not a substitution, update play_ID and append
- If it was a substitution, take out/append corresponding players to active player dataframe
- Append that dataframe to players on court dataframe
- Return players on court dataframe for full quarterly

best_guess_starters():

- This is for rare edge case (which I will refer to as the TJ Warren effect from now on)

130-138:

- Take all players who logged plays in the quarter, and were not subbed in or out.
- Take how many plays they were involved in

-The player with the least amount of plays is removed from the starting lineup.
-EX: TJ Warren only showed up in one play, the other unaccounted for player showed up in around 22 plays. It’s an educated guess as to who was actually on the court