Advanced API:

getMostAttractiveStadiums:

Sources: Stadiums, goalsPerStadium (view)

We joined goals per stadium on stadiums to get goaled scored per stadium including the 0 for unreported stadiums. We ordered as expected and returned the list.

mostGoalsForTeam:

Sources: goalsPerPlayer(view)

We filtered the players by the required team, sorted, limited and returned.

getClosePlayers:

Sources: friends (view), Players, black magic.

This one was tricky.

We couldn't use too many views because the parameter (playerId) needed to be taking care of soon to have an efficient query.

We used 3 nested queries.

N1: Use friends, to filter by pid2=playerId and count by pid1. This way, we've got a relation that holds for every player that scored with playerId, how many matches they scored together. Another interesting row we've got, is how many matches playerId scored with itself (with no duplications) => How many matches playerId scored in.

N2: We joined N1 on Players to get (pid, scoredTogether) for every player existed. scoredTogether will be 0 if the player didn't score in any mutual match.

N3: We used our wild row on N2 – the self PlayerId row. We used it and the fact that the player scored with itself the most. Then, using N2 and MAX, we had a relation with a single constant col – the number of matches playerId scored in.

The query: We used N2 and N3 to get a relation with on col – pid. We filtered with "WHERE n2.pid <> {playerId} AND n2.playedTogether * 2 >= n3.max" sorted, limited, and returned.