A. I’ve designed a Database for NBA and the first thing I like to say is that I don’t know much about the NBA so I tried my best. You can find the NBA database structure in the NBA.sql file at the end I wrote some queries for searching results using date and team name and search game statistics using the player name.

I’ve also use indexing in some fields to make it fast.

Table games →

id → game id

home\_team\_id → Id of the home team

away\_team\_id → Id of the away team

score\_home → Score of the home team

score\_away → Score of away team

date → Game date

Table teams →

id→ team id

team\_name → team name

Table players →

id→player id

first\_name →

last\_name →

team\_id → id of the team this user belongs to

Table game\_leaders → This table is will hold the data of top scorer (Optional)

id →

game\_id →

team\_id →

player\_id →

score → highest score by the player of this specific game

B. For solving this problem I use recursion. You can find the solution in the file\_remover.php this file has 3 methods. removeFiles($prefix, $dir) this is the main method for deleting the files. It requires two methods $prefix the prefix of the files which needs to be deleted and $dir name of the main directory in which we have dir and sub dirs and files. The other two methods randomString() and createSampleDirs($prefix,$main\_dir) are for creating sample directories and files.

C. The solution of this question is located in the sort.php. Here we a method quick\_sort($array) Which receives an array of unsorted numbers. I’ve choose quick sort because I think this is the best solution for this problem. And it will take 10000 to 13600 min to run this function 10 billion times on a normal machine. Time complexity (n log(n)) and Space complexity is O(n log(n))

D. I don’t understand the question I also asked for clarification but got no answer.

## Advanced/Practical

Setup

Run→ composer install

Run→ php artisan migrate

Run→ php db:seed

Routes:

users:

/api/users → get all users

/api/users/USER-ID → get user by id

/api/users/USER-EMAIL → get by email

Posts:

/api/posts → get all posts

/api/posts/USER-ID → get post by user id

/api/posts/USER-EMAIL → get post by user email

Comments:

/api/comments → get all comments

/api/comments/POST-ID → get comments by post id

Search Post:

/api/searchPost/POST-TITLE → search post by title

Structure

In this project, we have 3 tables users, posts, comments. For inserting jsonplaceholder data I’am using 3 json files (users.json , posts.json, and comments.json) you can find those files into the project root dir. I’ve created database seeders for inserting the data.

And as for the api results, I added pagination so if we have a large dataset in the future our app can handle data without crashing or taking too much to load data.

Future Optimization:

I’ve added indexing on certain fields to make database fetching faster and for making it faster we could use caching techniques like redis and can also use ElasticSearch. I'm using redis and ElasticSearch with kibana and the results are impressive.