**📖 Explicação das Questões**

**Questão 1 – Criação do Banco**

* **O que pede:** Criar o banco de dados megaArena e as coleções jogadores, torneios e partidas.
* **Por que:** Toda aplicação precisa de um banco inicial estruturado. Aqui, temos três entidades principais: jogadores, torneios e partidas.
* **Exemplo:**
* use megaArena
* db.createCollection("jogadores")
* db.createCollection("torneios")
* db.createCollection("partidas")

**Questão 2 – Embedding**

* **O que pede:** Criar uma versão do torneio onde os jogadores ficam **dentro do documento do torneio**.
* **Por que:** Esse modelo (embedding) é útil quando você sempre precisa acessar jogadores junto com o torneio. Evita consultas adicionais.
* **Exemplo:**
* db.torneios.insertOne({
* nome: "Free Fire Challenge",
* premio: 4000,
* jogadores: [
* { nome: "Carlos", nick: "carlim" },
* { nome: "Ana", nick: "aninha" }
* ]
* })

**Questão 3 – Referencing**

* **O que pede:** Criar uma versão onde o torneio guarda **apenas os IDs dos jogadores**.
* **Por que:** Esse modelo (referencing) é útil quando o mesmo jogador participa de vários torneios. Assim, ele fica em uma coleção separada, sem duplicação de dados.
* **Exemplo:**
* db.torneios.insertOne({
* nome: "Valorant Masters",
* premio: 6000,
* jogadores: [
* ObjectId("ID\_DO\_CARLOS"),
* ObjectId("ID\_DA\_ANA")
* ]
* })

**Questão 4 – Justificativa de Modelagem**

* **O que pede:** Explicar quando usar embedding ou referencing.
* **Por que:** Isso mostra que você entende os trade-offs do MongoDB.
* **Resposta:**
  + **Embedding**: quando os dados são sempre usados juntos (ex.: lista de jogadores em um torneio específico).
  + **Referencing**: quando os dados são reutilizados em vários lugares (ex.: jogador que participa de muitos torneios).

**Questão 5 – Consulta (prêmio >= 4000)**

* **O que pede:** Listar torneios que têm prêmio maior ou igual a 4000.
* **Por que:** É uma consulta simples com filtro numérico.
* **Exemplo:**
* db.torneios.find({ premio: { $gte: 4000 } })

**Questão 6 – Consulta avançada com $and**

* **O que pede:** Listar jogadores do **Brasil** com **idade > 21**.
* **Por que:** Treinar o uso do operador $and para múltiplas condições.
* **Exemplo:**
* db.jogadores.find({
* $and: [ { pais: "Brasil" }, { idade: { $gt: 21 } } ]
* })

**Questão 7 – Operador $exists**

* **O que pede:** Mostrar jogadores que já possuem o campo nick.
* **Por que:** No MongoDB, documentos podem ter campos opcionais. O $exists ajuda a identificar quem já tem ou não aquele campo.
* **Exemplo:**
* db.jogadores.find({ nick: { $exists: true } })

**Questão 8 – Atualização com $set**

* **O que pede:** Atualizar o torneio “Copa CS:GO” e adicionar o campo status: "ativo".
* **Por que:** Serve para mostrar como alterar ou acrescentar dados em documentos existentes.
* **Exemplo:**
* db.torneios.updateOne(
* { nome: "Copa CS:GO" },
* { $set: { status: "ativo" } }
* )

**Questão 9 – Exclusão**

* **O que pede:** Remover o jogador chamado Pedro.
* **Por que:** Treinar a remoção de documentos com base em um critério.
* **Exemplo:**
* db.jogadores.deleteOne({ nome: "Pedro" })

**Questão 10 – Paginação**

* **O que pede:** Listar os torneios ordenados pela premiação (do maior para o menor), mas exibir apenas o **segundo**.
* **Por que:** Ensina a usar sort, skip e limit para paginação de resultados.
* **Exemplo:**
* db.torneios.find().sort({ premio: -1 }).skip(1).limit(1)