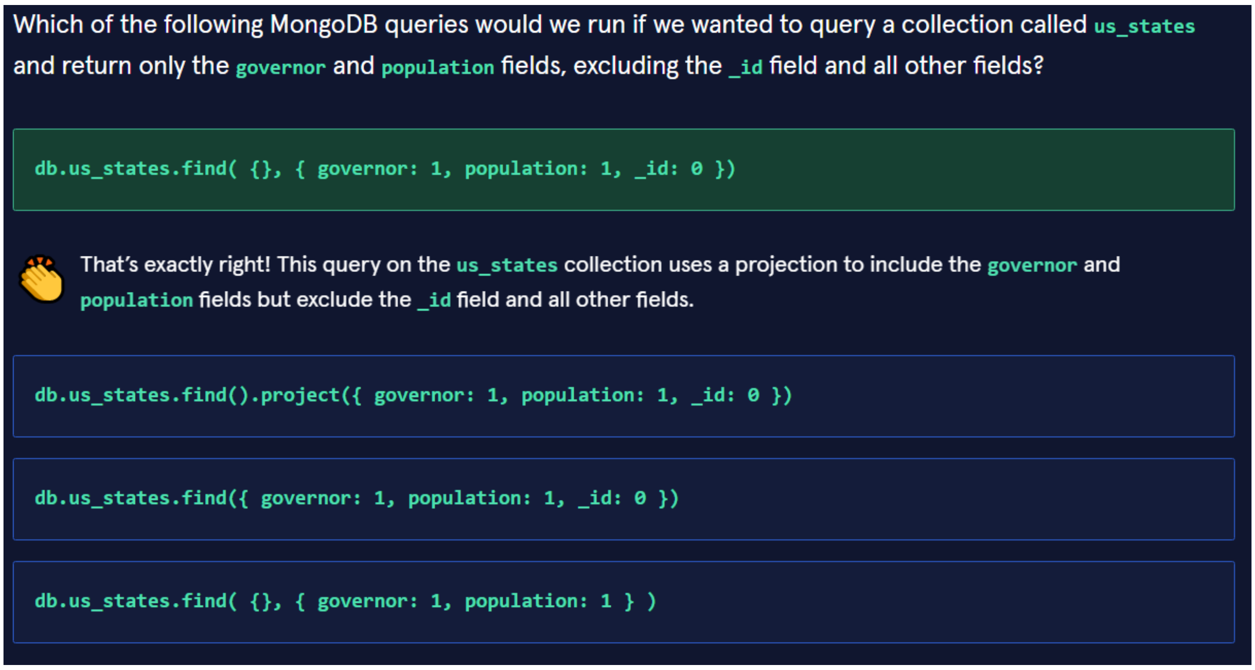
QUIZ I



Which of the following MongoDB queries would we run if we wanted to query a collection called `us_states` and return only the `governor` and `population` fields, excluding the `_id` field and all other fields?

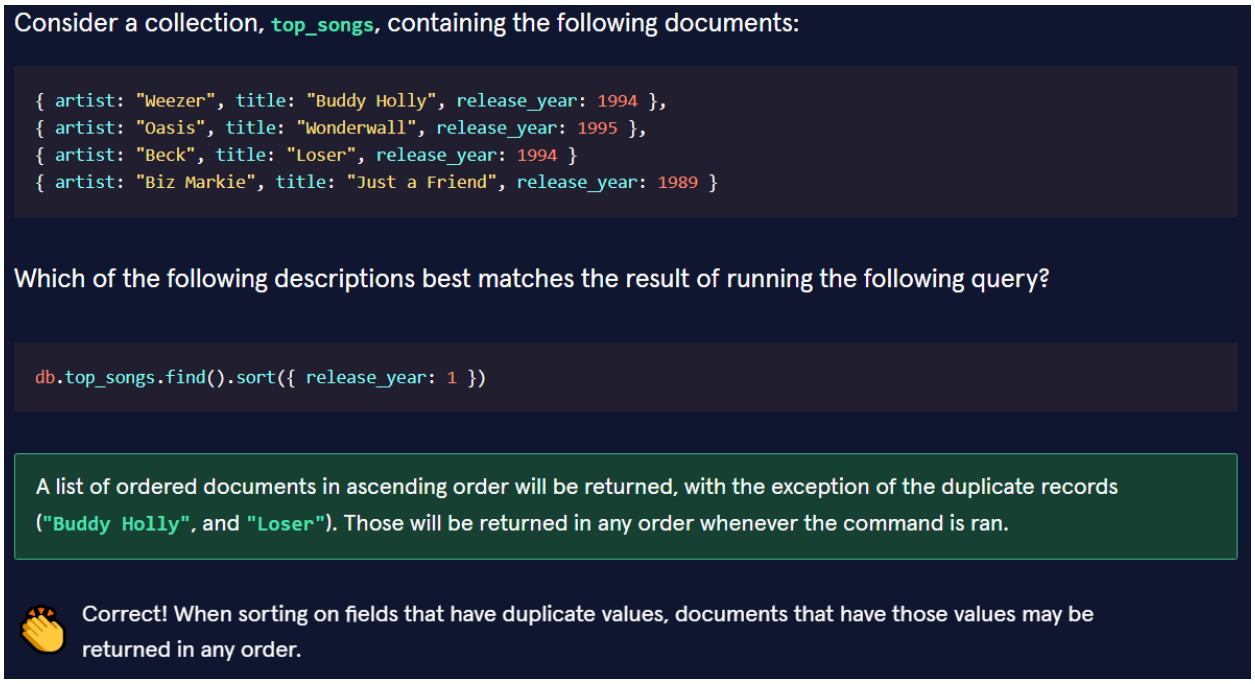
```
db.us_states.find( {}, { governor: 1, population: 1, _id: 0 })
```

👏 That's exactly right! This query on the `us_states` collection uses a projection to include the `governor` and `population` fields but exclude the `_id` field and all other fields.

```
db.us_states.find().project({ governor: 1, population: 1, _id: 0 })
```

```
db.us_states.find({ governor: 1, population: 1, _id: 0 })
```

```
db.us_states.find( {}, { governor: 1, population: 1 } )
```



Consider a collection, `top_songs`, containing the following documents:

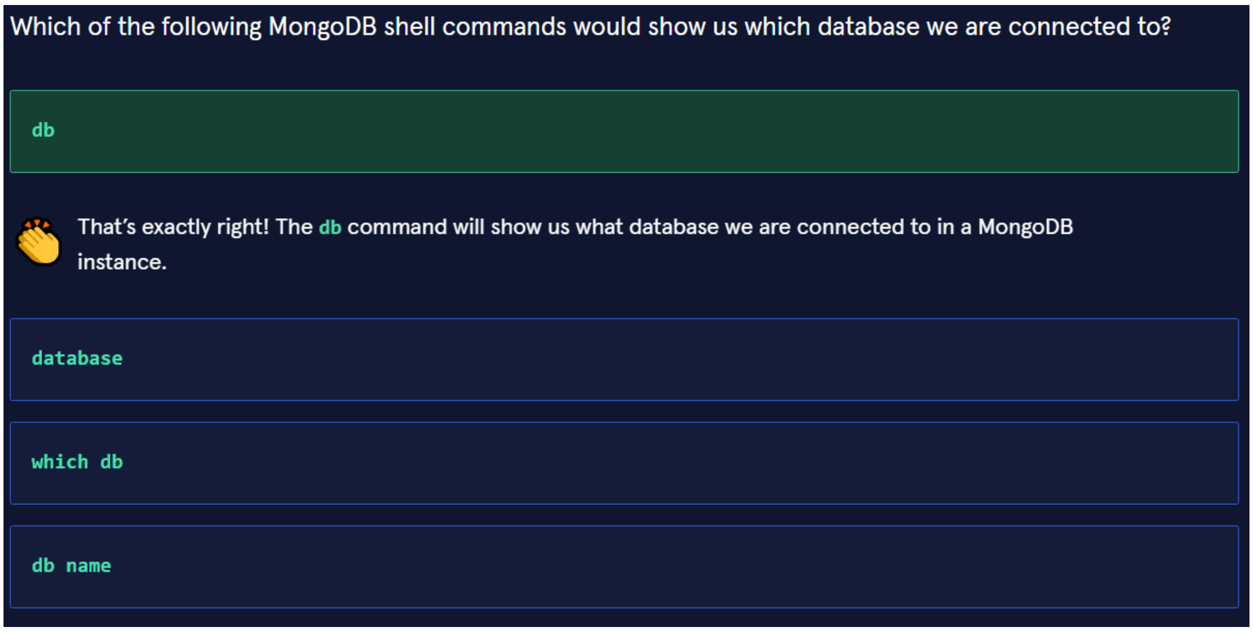
```
{ artist: "Weezer", title: "Buddy Holly", release_year: 1994 },
{ artist: "Oasis", title: "Wonderwall", release_year: 1995 },
{ artist: "Beck", title: "Loser", release_year: 1994 }
{ artist: "Biz Markie", title: "Just a Friend", release_year: 1989 }
```

Which of the following descriptions best matches the result of running the following query?

```
db.top_songs.find().sort({ release_year: 1 })
```

A list of ordered documents in ascending order will be returned, with the exception of the duplicate records (`"Buddy Holly"`, and `"Loser"`). Those will be returned in any order whenever the command is ran.

👏 Correct! When sorting on fields that have duplicate values, documents that have those values may be returned in any order.



Which of the following MongoDB shell commands would show us which database we are connected to?

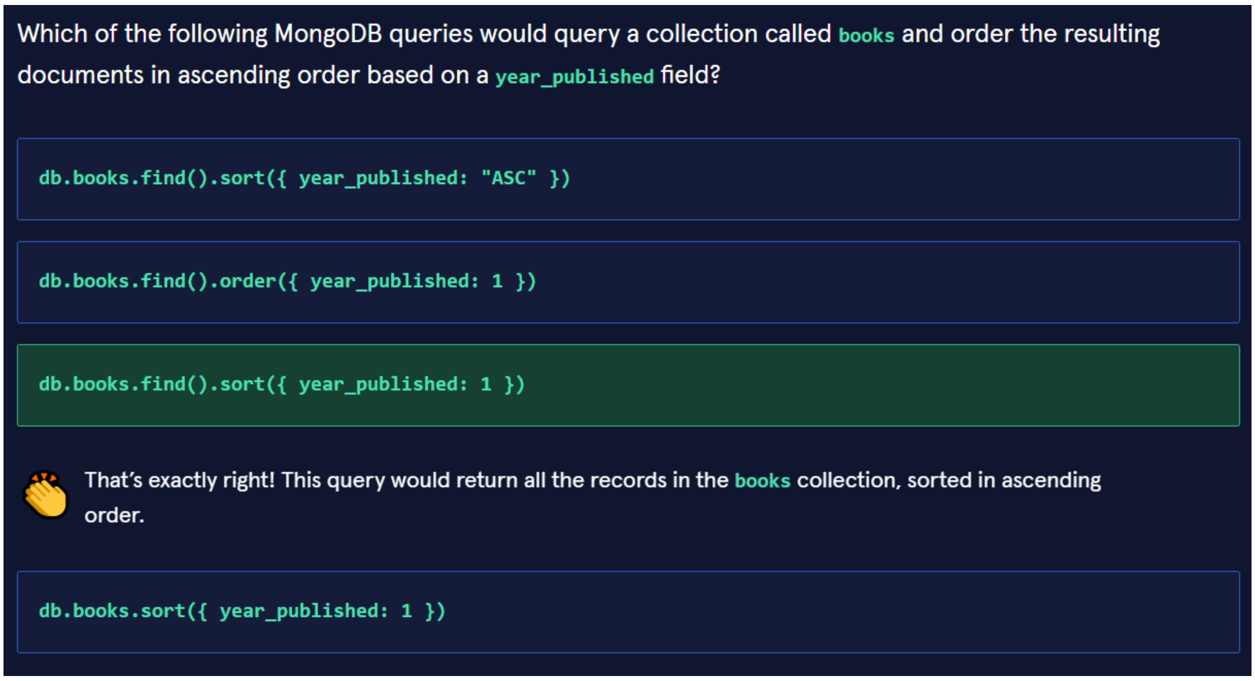
[✓ `db`]

👏 That's exactly right! The `db` command will show us what database we are connected to in a MongoDB instance.

`database`

`which db`

`db name`



Which of the following MongoDB queries would query a collection called `books` and order the resulting documents in ascending order based on a `year_published` field?

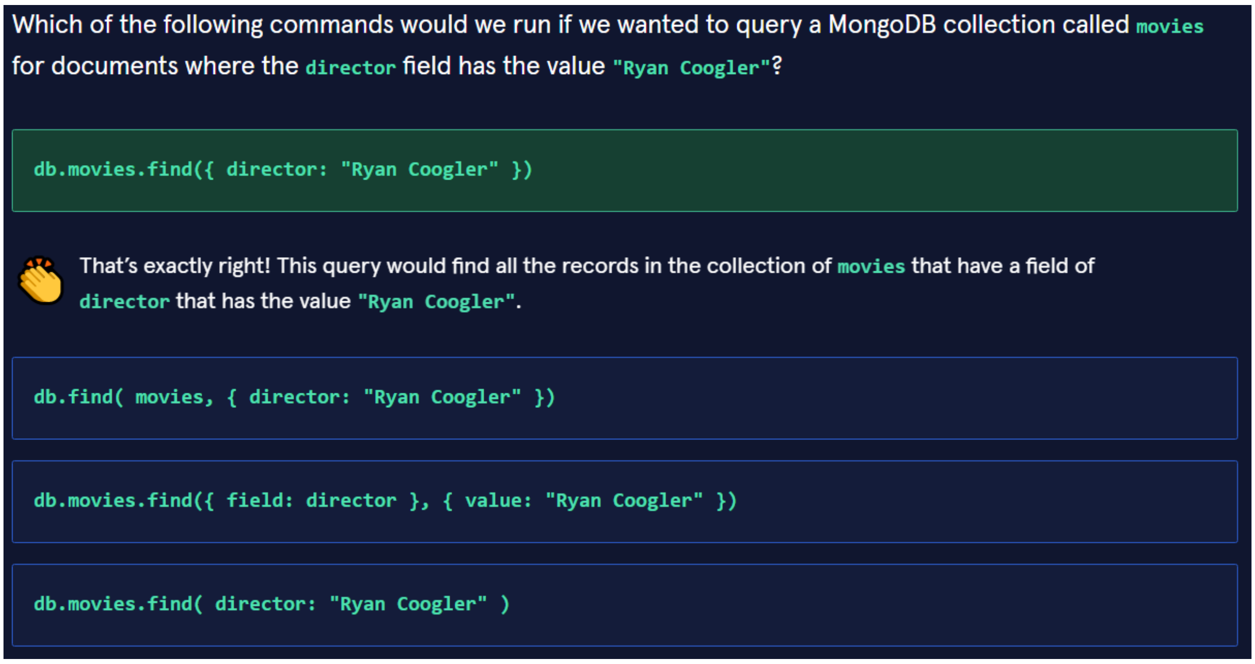
`db.books.find().sort({ year_published: "ASC" })`

`db.books.find().order({ year_published: 1 })`

[✓ `db.books.find().sort({ year_published: 1 })`]

👏 That's exactly right! This query would return all the records in the `books` collection, sorted in ascending order.

`db.books.sort({ year_published: 1 })`



Which of the following commands would we run if we wanted to query a MongoDB collection called `movies` for documents where the `director` field has the value `"Ryan Coogler"`?

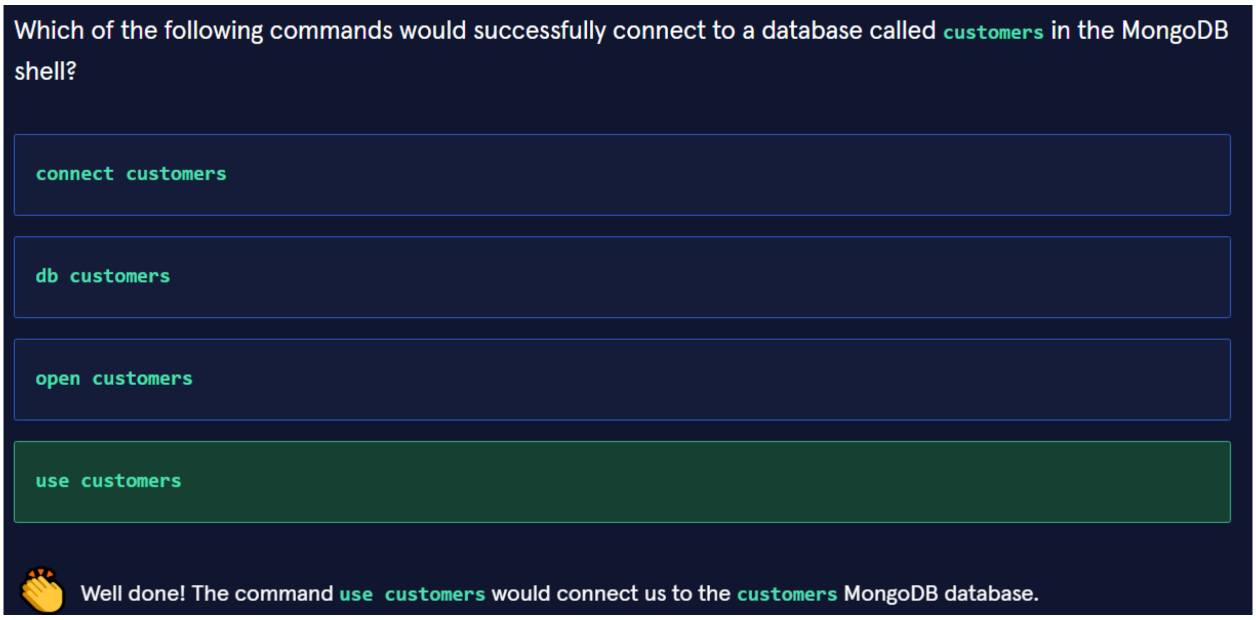
[✓ `db.movies.find({ director: "Ryan Coogler" })`]

👏 That's exactly right! This query would find all the records in the collection of `movies` that have a field of `director` that has the value `"Ryan Coogler"`.

`db.find( movies, { director: "Ryan Coogler" })`

`db.movies.find({ field: director }, { value: "Ryan Coogler" })`

`db.movies.find( director: "Ryan Coogler" )`



Which of the following commands would successfully connect to a database called `customers` in the MongoDB shell?

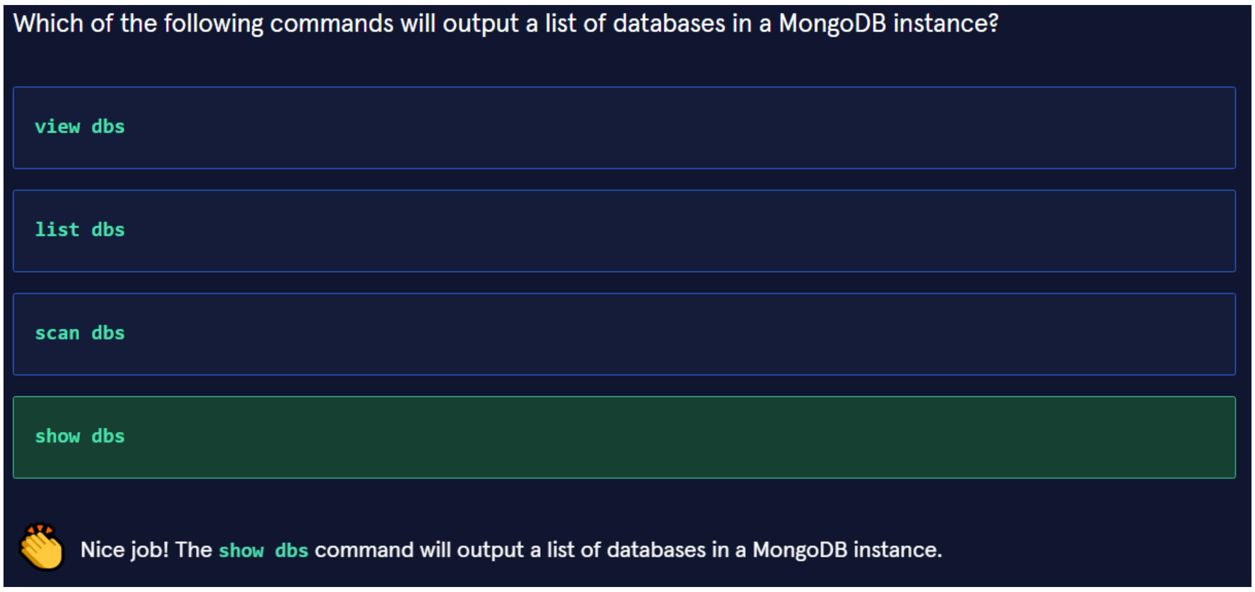
`connect customers`

`db customers`

`open customers`

[✓ `use customers`]

👏 Well done! The command `use customers` would connect us to the `customers` MongoDB database.



Which of the following commands will output a list of databases in a MongoDB instance?

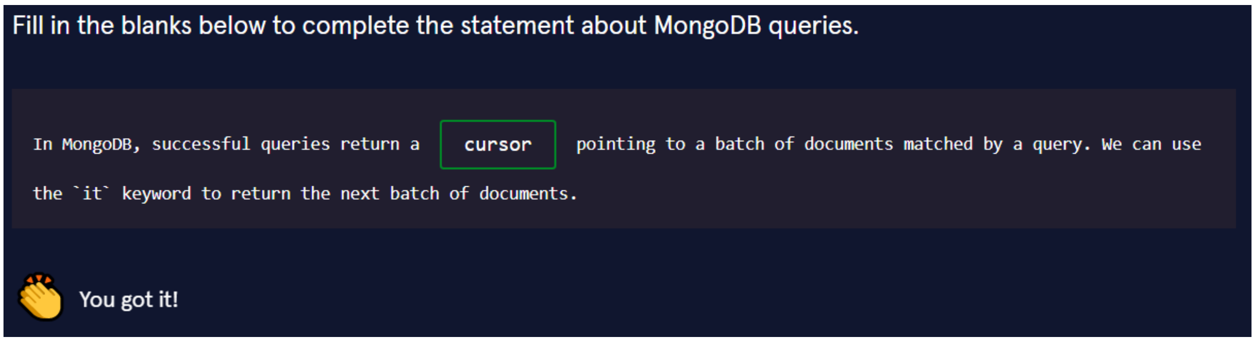
`view dbs`

`list dbs`

`scan dbs`

[✓ `show dbs`]

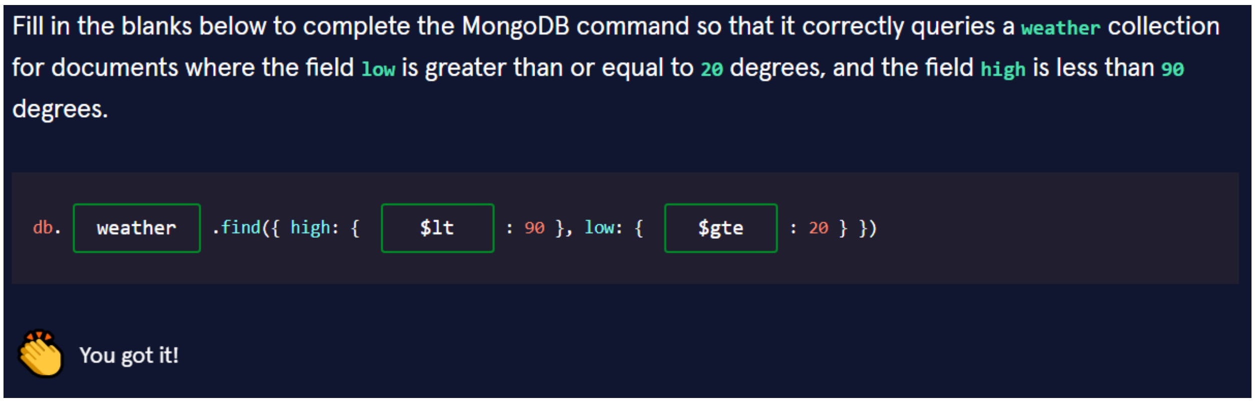
👏 Nice job! The `show dbs` command will output a list of databases in a MongoDB instance.



Fill in the blanks below to complete the statement about MongoDB queries.

```
In MongoDB, successful queries return a [cursor] pointing to a batch of documents matched by a query. We can use
the `it` keyword to return the next batch of documents.
```

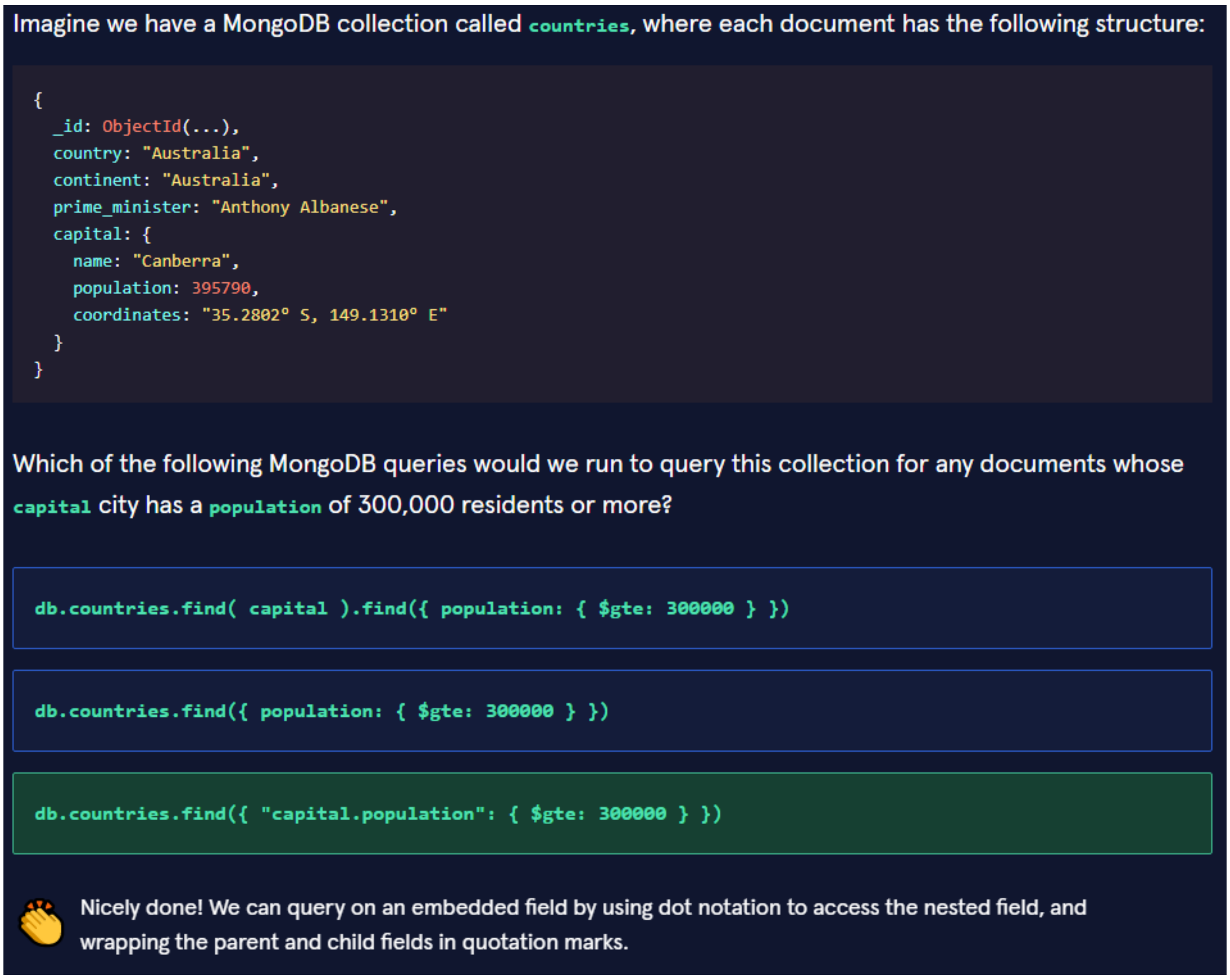
👏 You got it!



Fill in the blanks below to complete the MongoDB command so that it correctly queries a `weather` collection for documents where the field `low` is greater than or equal to `20` degrees, and the field `high` is less than `90` degrees.

```
db.[weather].find({ high: { [$lt] : 90 }, low: { [$gte] : 20 } })
```

👏 You got it!



Imagine we have a MongoDB collection called `countries`, where each document has the following structure:

```
{
  _id: ObjectId(...),
  country: "Australia",
  continent: "Australia",
  prime_minister: "Anthony Albanese",
  capital: {
    name: "Canberra",
    population: 395790,
    coordinates: "35.2802° S, 149.1310° E"
  }
}
```

Which of the following MongoDB queries would we run to query this collection for any documents whose `capital` city has a `population` of 300,000 residents or more?

- `db.countries.find( capital ).find({ population: { $gte: 300000 } })`
- `db.countries.find({ population: { $gte: 300000 } })`
- [✓ `db.countries.find({ "capital.population": { $gte: 300000 } })`]

👏 Nicely done! We can query on an embedded field by using dot notation to access the nested field, and wrapping the parent and child fields in quotation marks.