Final Exam

Final: Question 5

Back to the Question

The correct answer is the following:

```
var pipeline = [
    {"$match": { "country": "USA"}},
    {"$addFields": { "mean": {"$avg": "$sunnydays"}}},
    {"$match": { "mean": {"$gte": 220}, "sunnydays": {"$not":
{"$lt": 200 }}}},
    {"$sort": {"city": 1}}
]
```

In this case, we try to remove as much data as possible upfront, all cities not matching the right country, using the available index.

We then calculate the mean number of sunny days.

The `$match` stage then filters out documents where the mean isn't greater than or equal to 220, and there are no entries in the **sunnydays** vector less than 200.

We are left with a sort in memory, however the number should be small enough to not take much resources. There are 285 cities with 100,000 habitants in the USA, and some are likely not to match the number of sunny days criteria.

Another answer provides the desired results, but will not improve the performance as much:

```
var pipeline = [
  {"$sort": {"city": 1}},
  {"$addFields": { "min": {"$min": "$sunnydays"}}},
  {"$addFields": { "mean": {"$avg": "$sunnydays" }}},
  {"$match": { "country": "USA", "min": {"$gte": 200}, "mean":
{"$gte": 220}}}
]
```

The above approach uses the index to sort, however it performs an unnecessary calculation to get the minimum value within **sunnydays**. Because the `$match` stage did not come prior to these `$addFields` stages, all source documents will pass through them, a wasteful computation.

The pipeline:

```
var pipeline = [
    {"$sort": {"city": 1}},
    {"$addFields": { "min": {"$min": "$sunnydays"}}},
    {"$match": { "country": "USA", "min": {"$gte": 200}}}
]
```

does not satisfy the query requirements.

The last 2 queries are doing a `$match` on `mean` before it is calculated, making them also invalid.

Proceed to next section