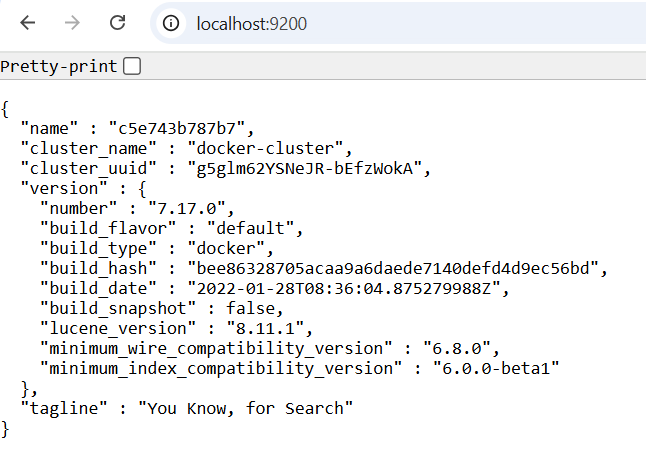
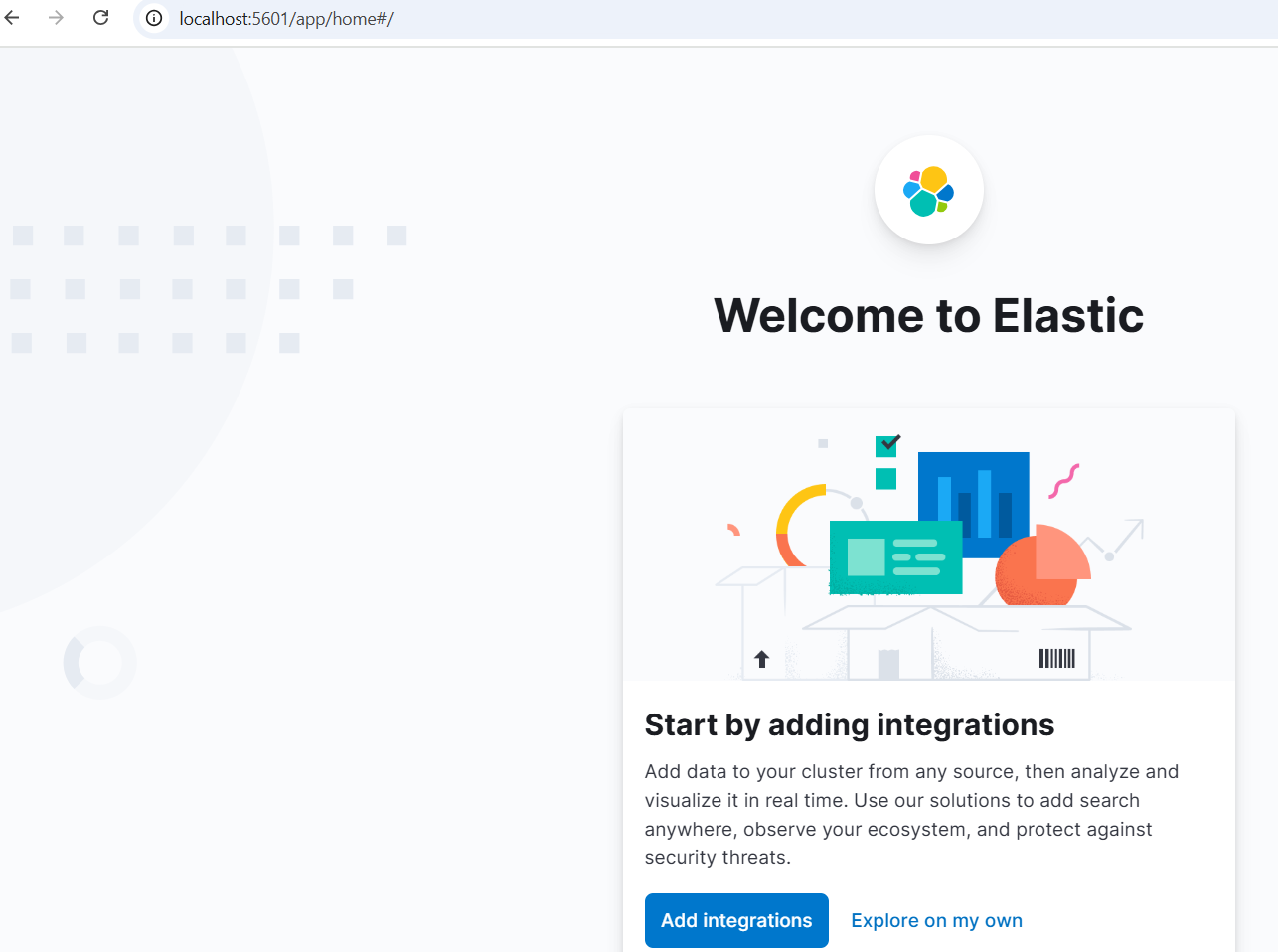
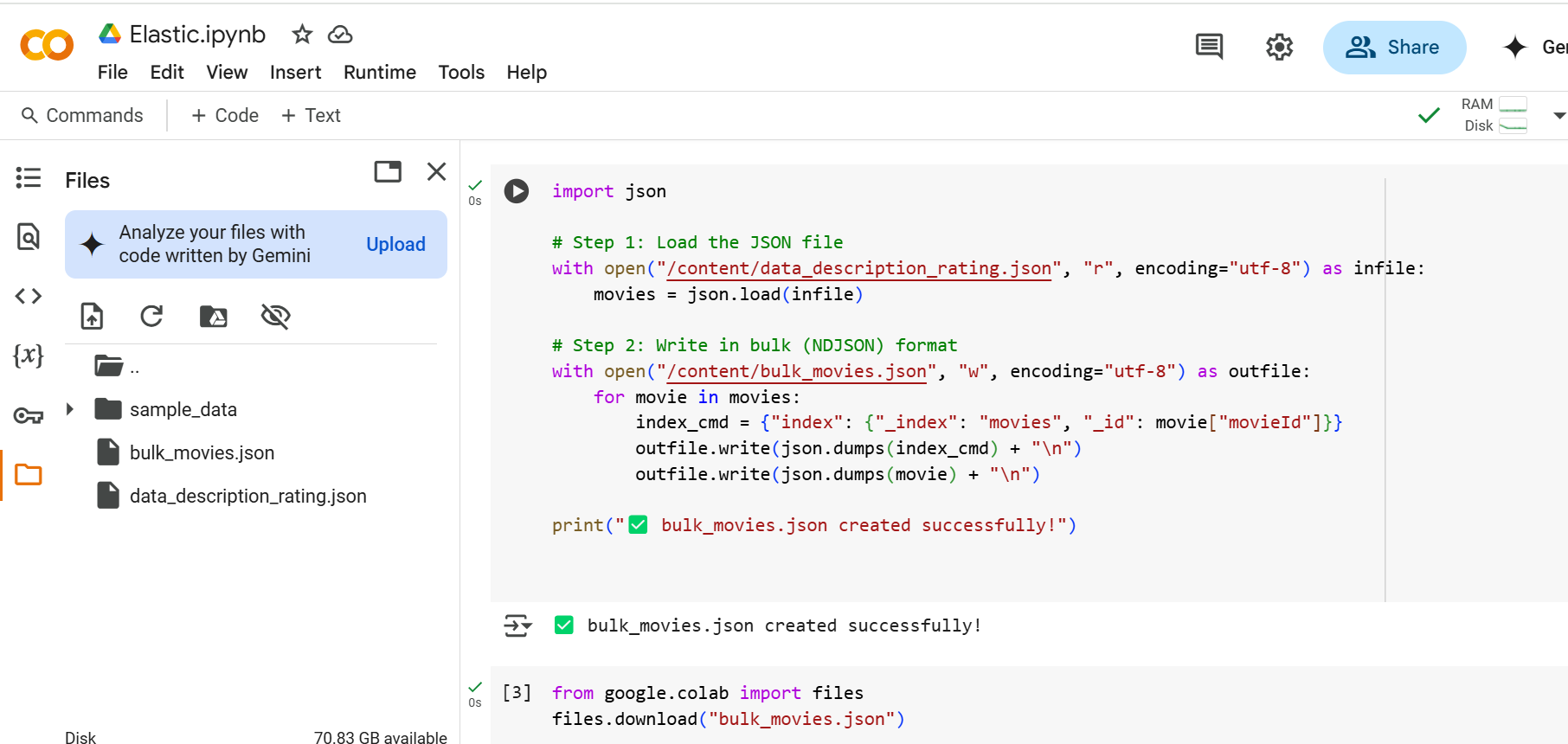


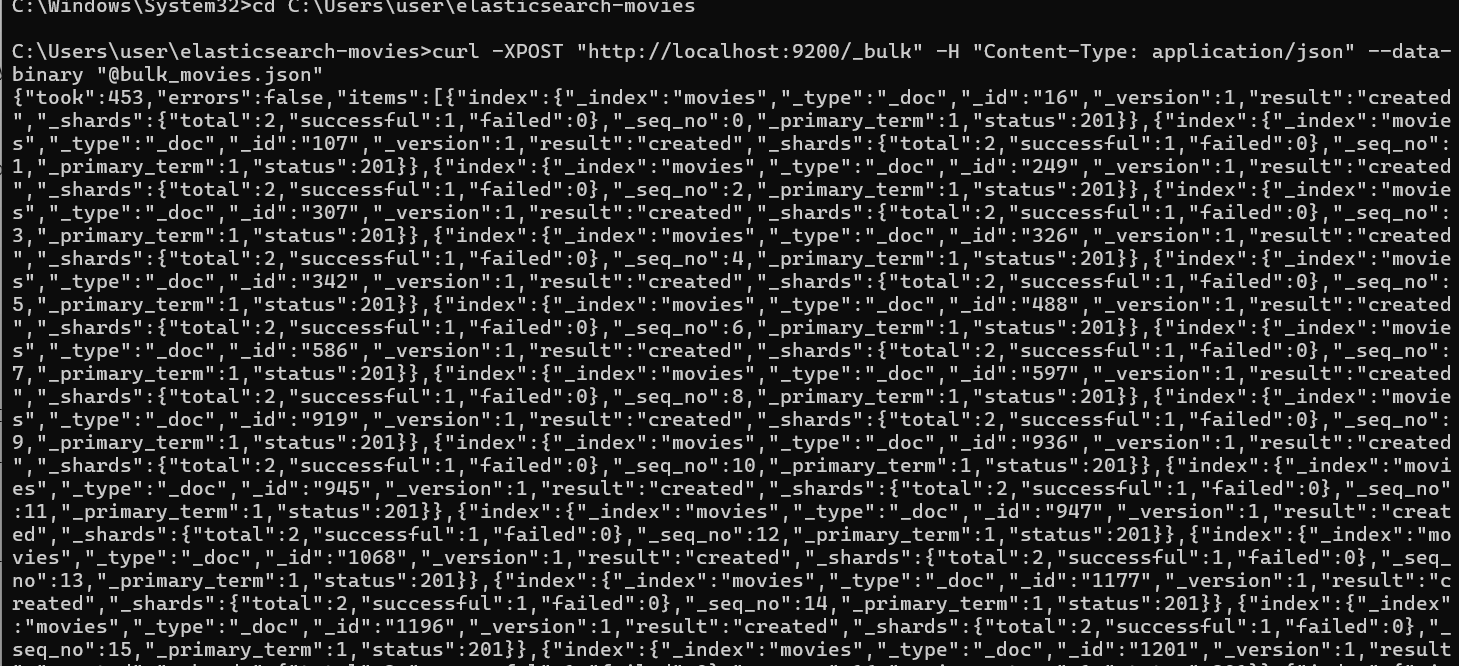
Elasticsearch:

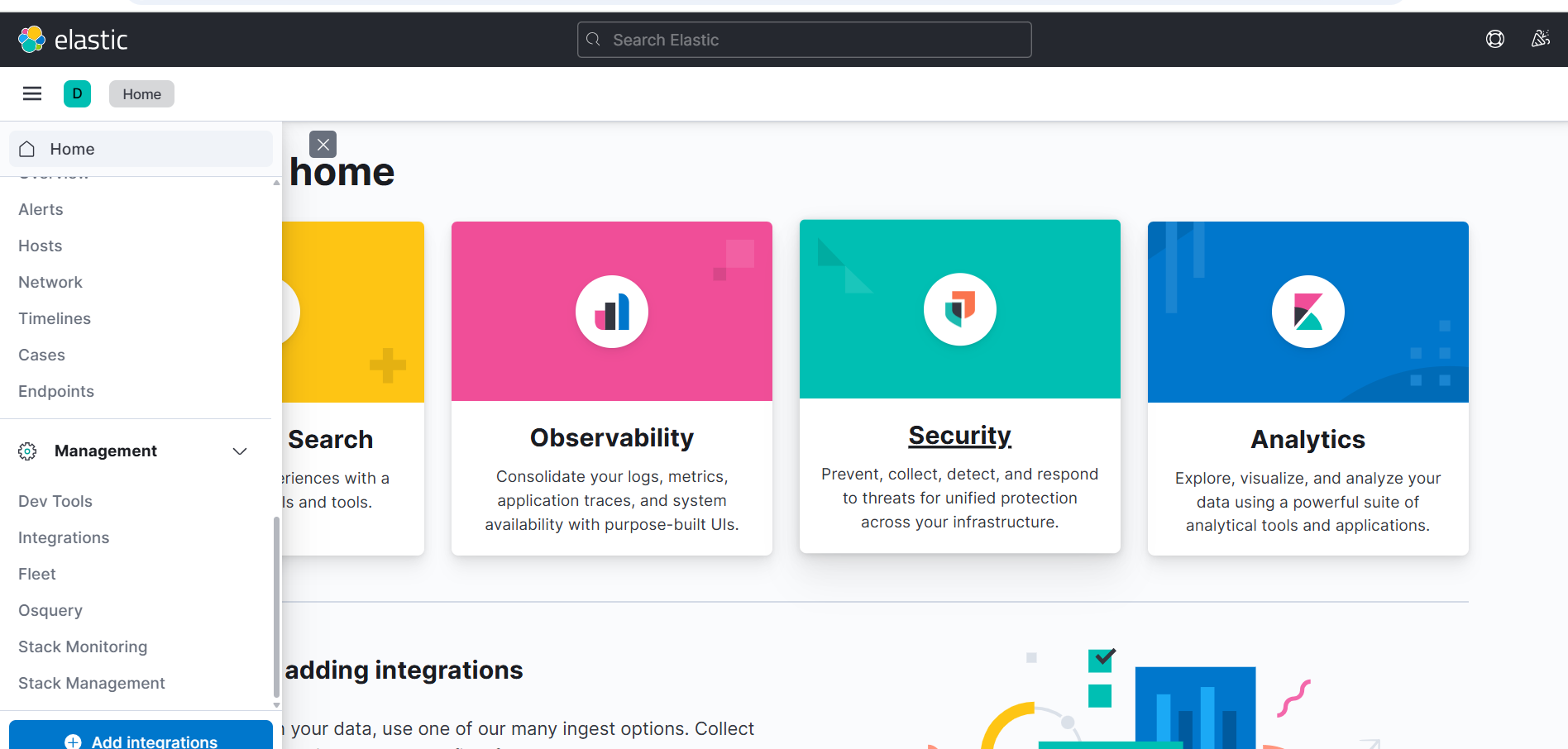


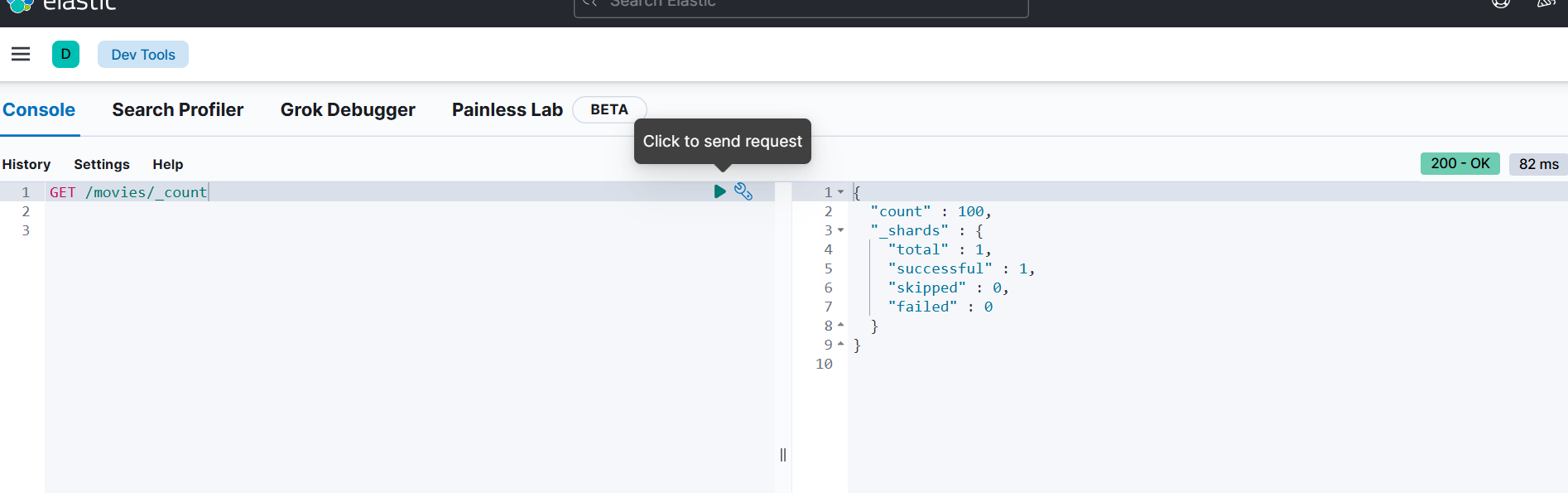
Kibana:







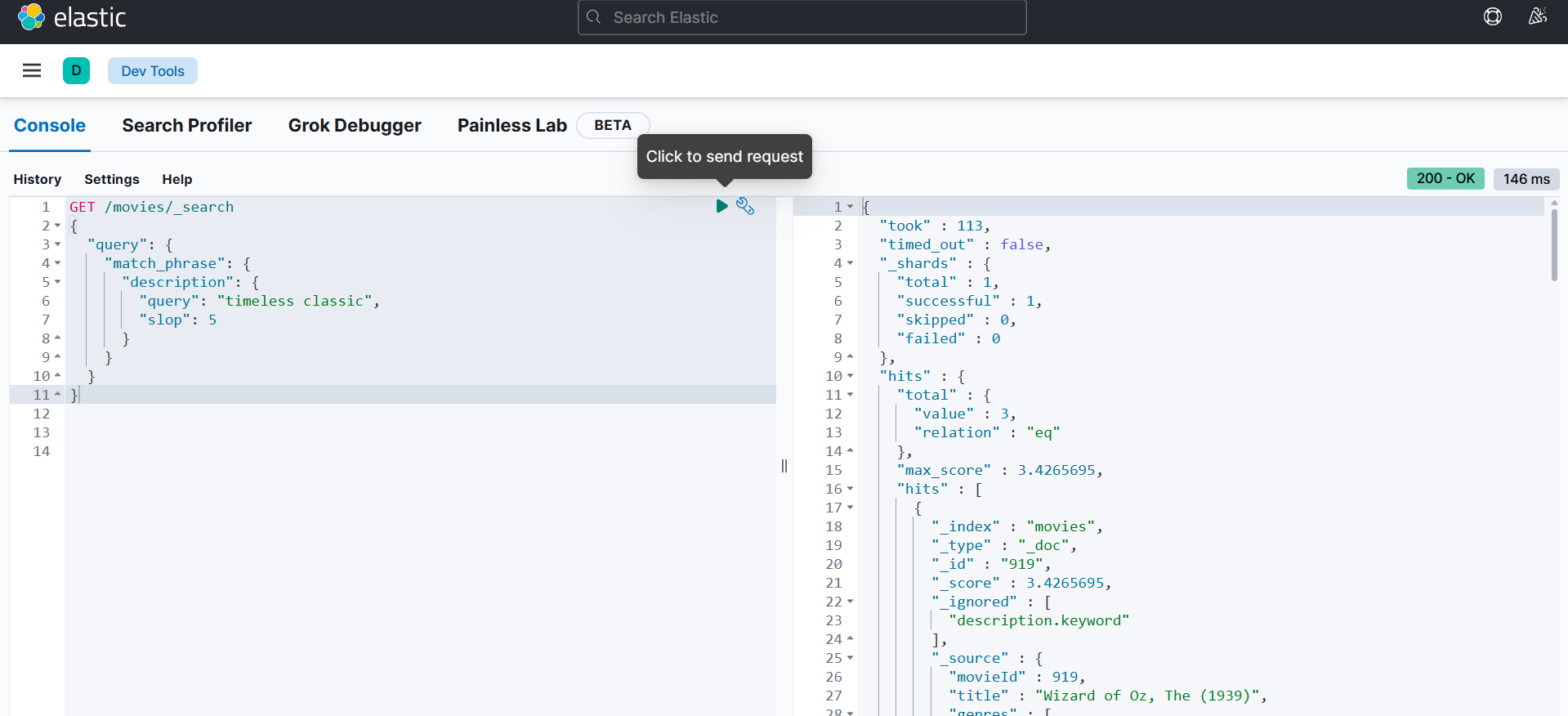


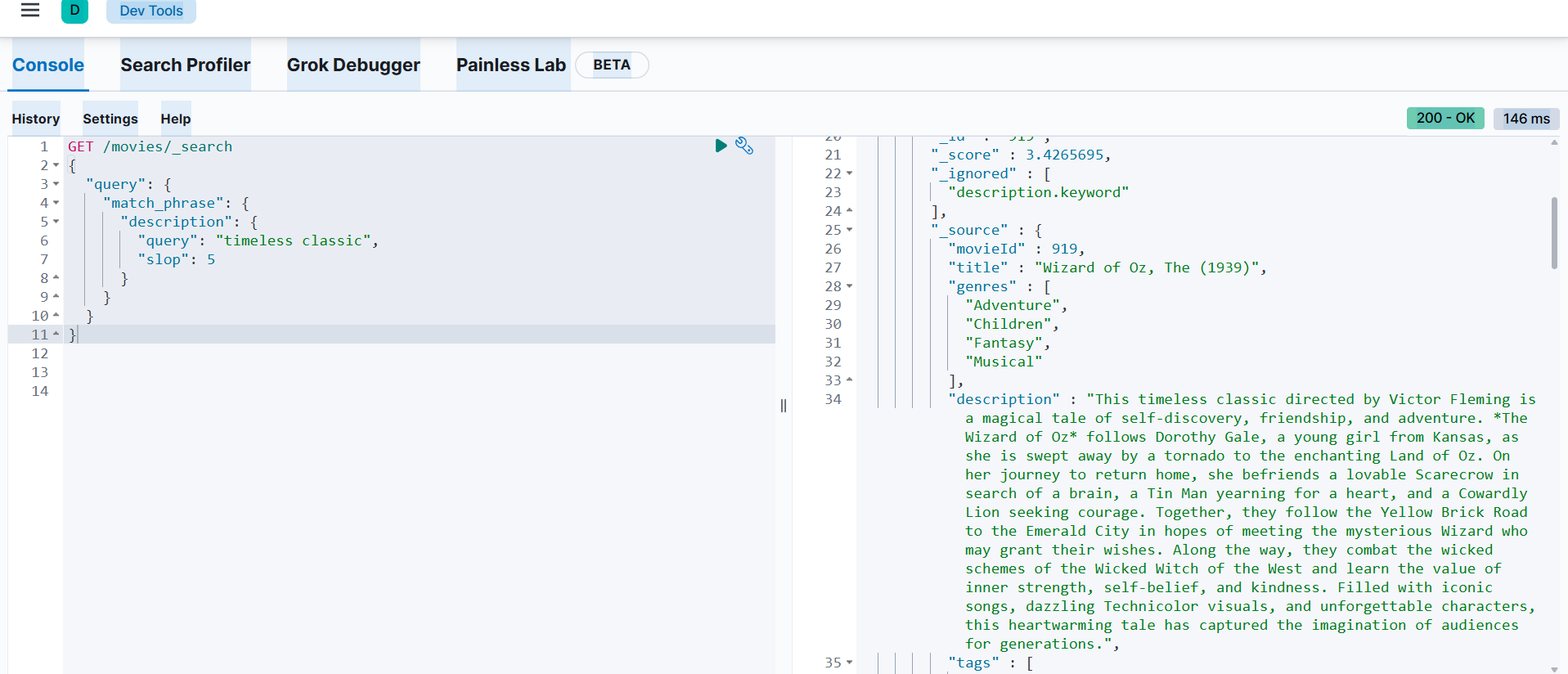


| **Task** | **Done?** |
| --- | --- |
| Converted JSON to bulk format | ✅ |
| Created index in Kibana | ✅ |
| Uploaded data with curl | ✅ |
| Verified the data with \_count | ✅ |



Query 1:

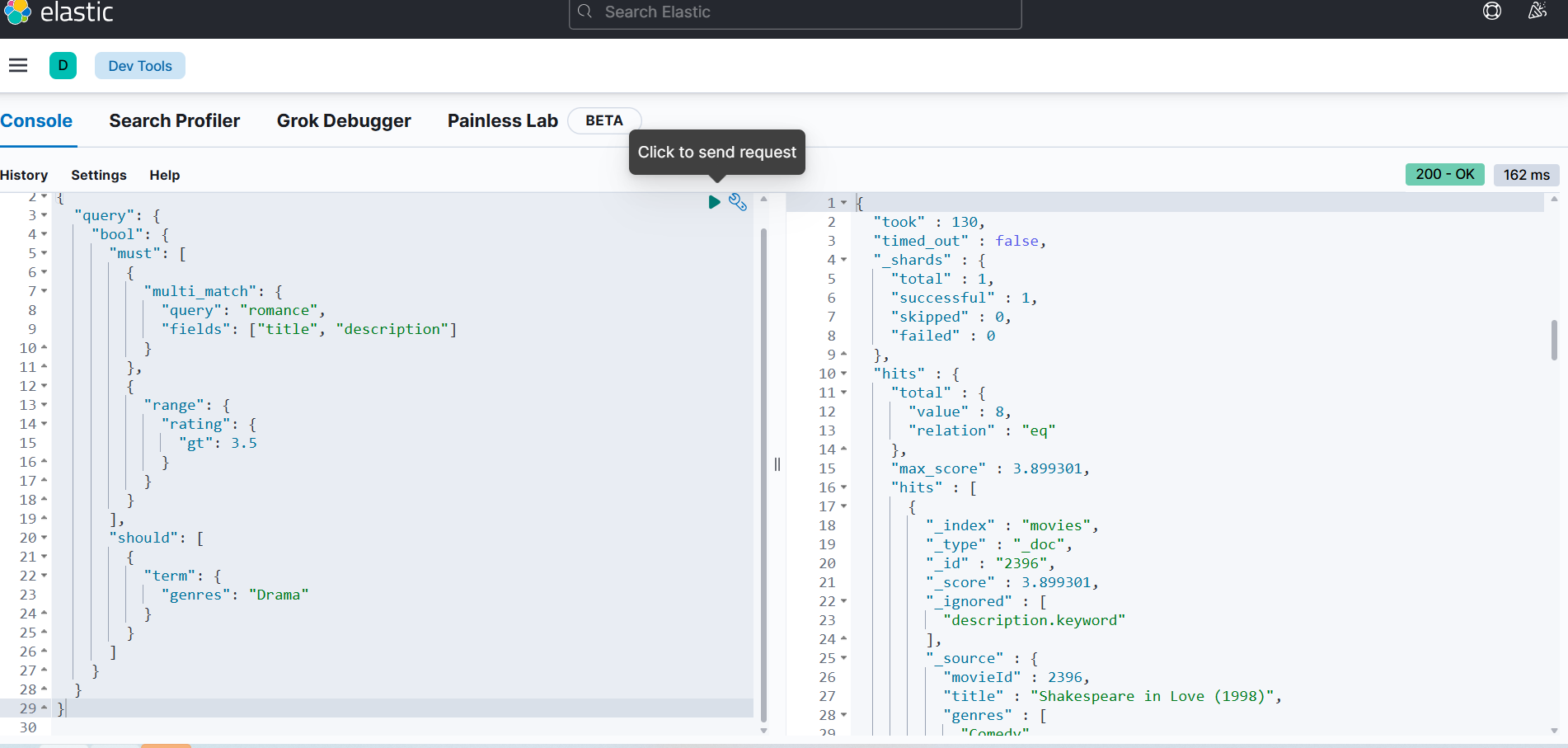




**Explanation:**

* match\_phrase: looks for exact phrase
* slop: 5: allows up to 5 other words between them, but **order is preserved**

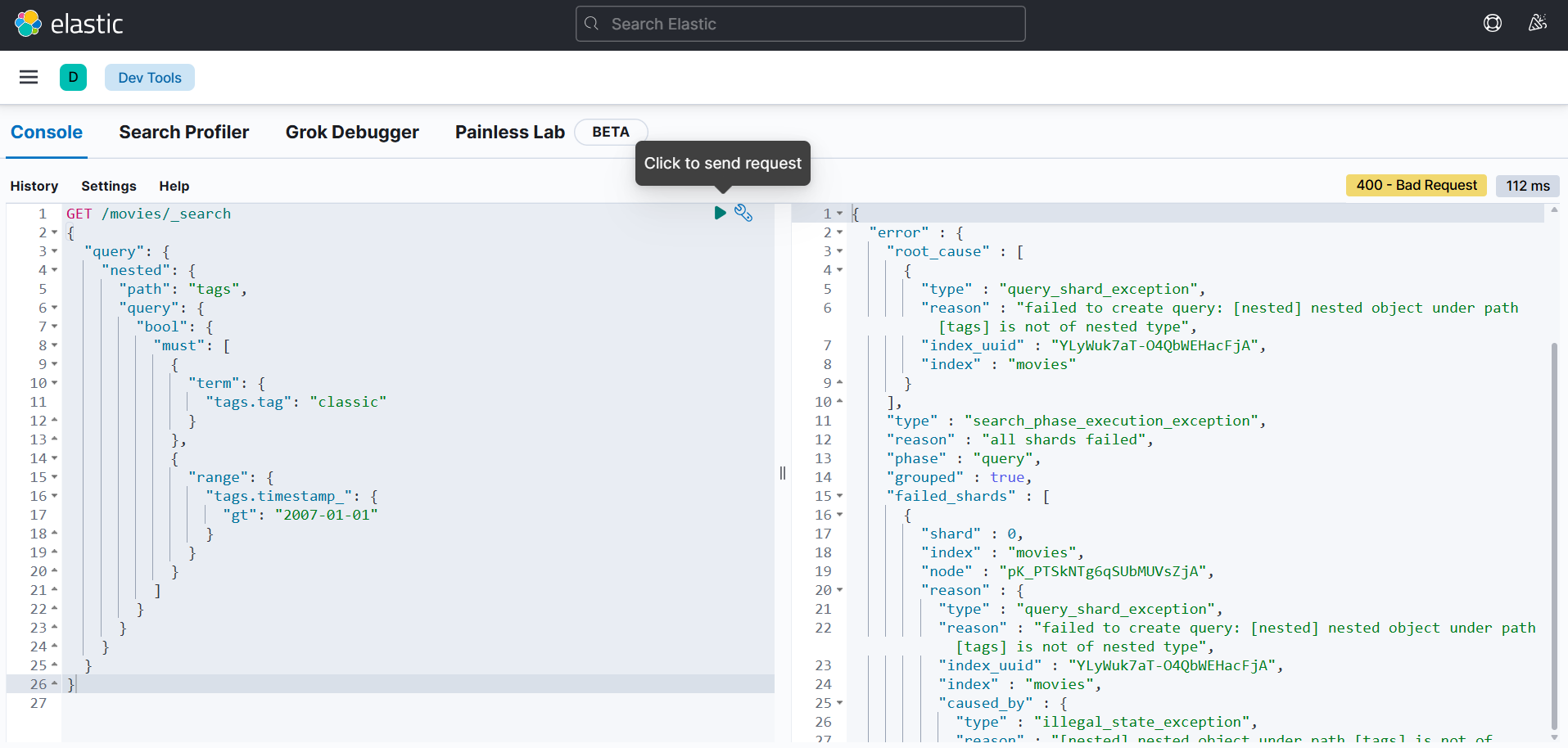
Query 2:



**Explanation:**

* must: both **romance** and **rating > 3.5** are required
* should: if the genre is **Drama**, boost it higher in the results

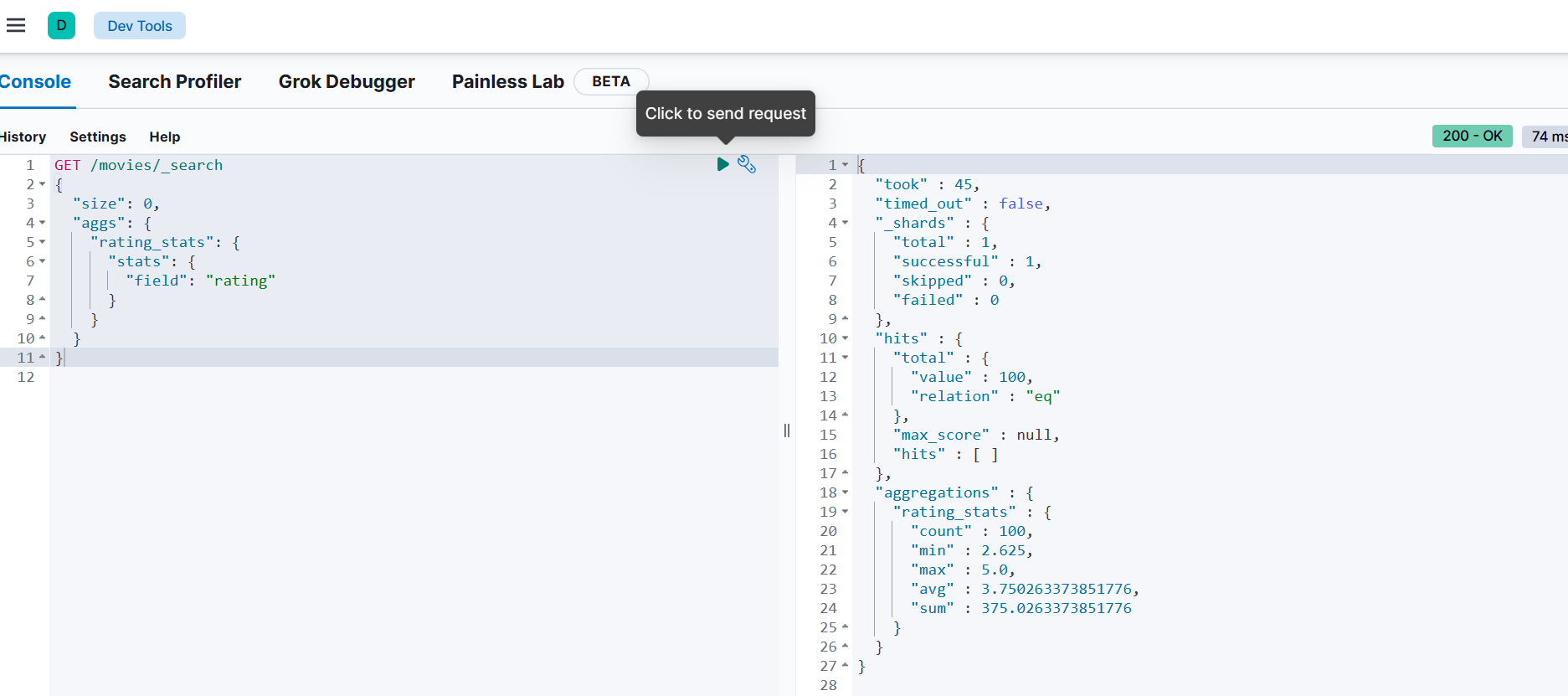
Query 3:



**Explanation:**

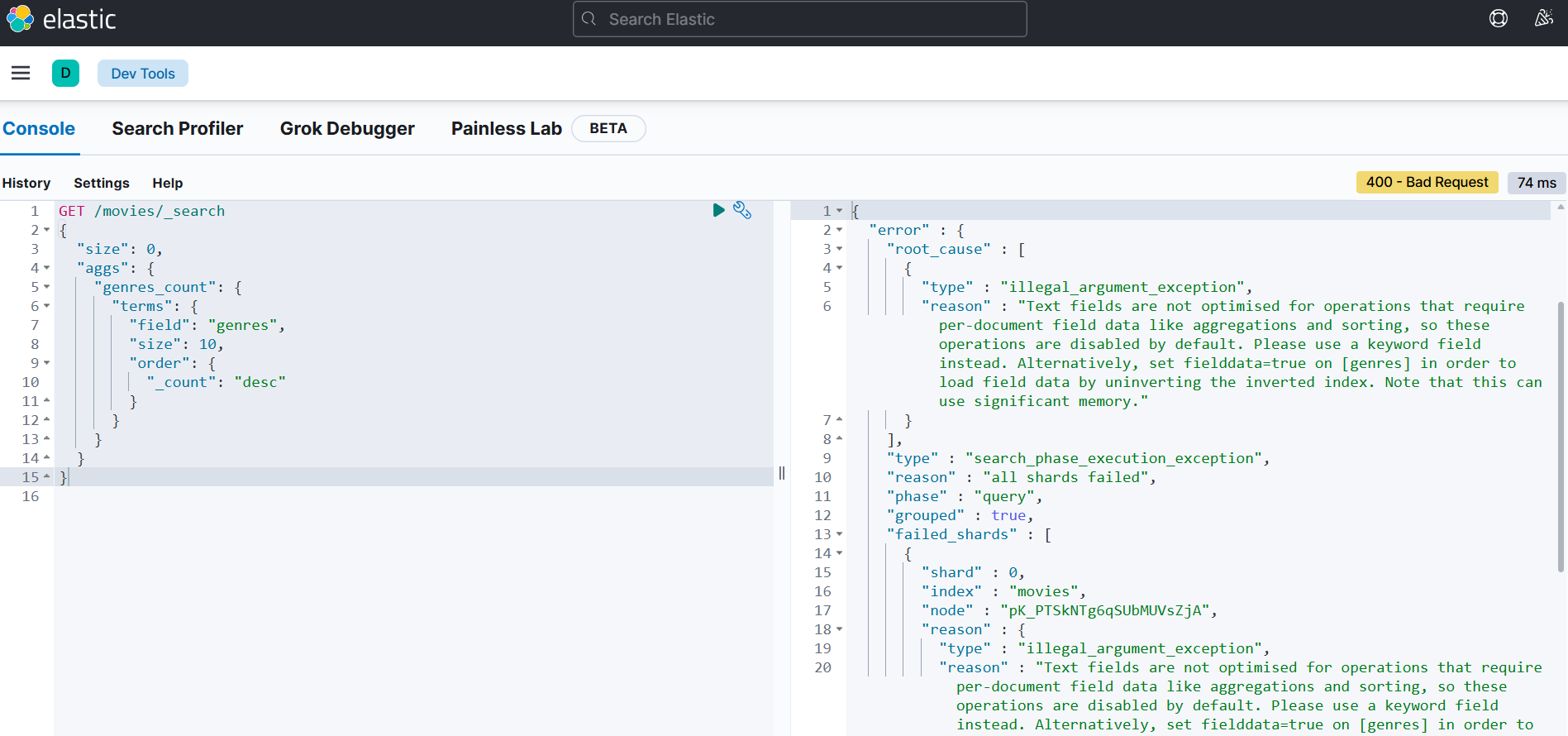
* nested: because tags is a nested array of objects
* term: matches exact value of "classic"
* range: makes sure timestamp\_ is after Jan 1, 2007
* The query ensures both **tag and timestamp** are matched **within the same object** in tags array

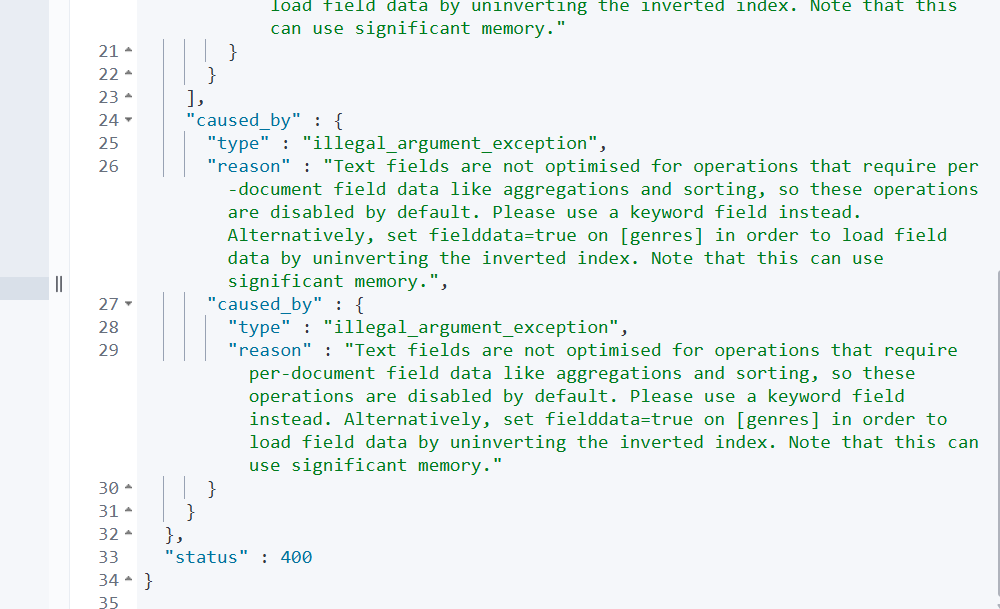




**Explanation:**

* "size": 0: we don’t need actual movie data, just the aggregation
* aggs: keyword for aggregations
* rating\_stats: just a name for this aggregation
* "stats": built-in aggregation type for numeric stats
* "field": "rating": use the rating field





**Explanation:**

* terms: groups by **unique values** in genres
* size: 10: show top 10 genres
* order.\_count: desc: most common genres first

Github:

https://github.com/dilrabonu/Real-Projects/tree/main/elasticsearch