1. Lucene
Utwórz indeks. Dodaj do niego dokumenty:
{“isbn”: “123”, “author” : “Anna Nowak“, “text” : “Ala ma kota”}
{“isbn”: “456”, “author” : “Maria Nowak“, “text” : “Ola ma psa”}
{“isbn”: “789”, “author” : “Anna Kowalska“, “text” : “Ela ma psa i kota”}
{“isbn”: “321”, “author” : “Anna Nowak“, “text” : “Tomek ma chomika”}
{“isbn”: “654”, “author” : “Anna Nowak“, “text” : “Koty jedzą myszy”}

Wyszukaj dokumenty, których autorem jest Anna Nowak.
Wyszukaj dokumenty o psach lub kotach.
Wyszukaj dokumenty Anny Nowak o psach lub kotach.
Zamodeluj dodawanie książki, która ma 2 autorów.

2. Elasticsearch:
Uruchom elasticsearcha: bin/elasticsearch
Uruchom kibanę: bin/kibana

Utwórz indeks. Dodaj do niego dokumenty z poprzedniego zadania.

Wyszukaj dokumenty, których autorem jest Anna Nowak.
Wyszukaj dokumenty o psach lub kotach.
Wyszukaj dokumenty Anny Nowak o psach lub kotach.

3. Logstash:
Załaduj plik Posts.json.gz do indeksu ‘posts’. Przykładowa konfiguracja znajduje się w pliku logstash.conf

bin/logstash -f logstash.conf

załaduj plik Comments.json.gz do indeksu ‘comments’.

4. Mapping:
Popraw mapowanie indeksu z postami tak, żeby wartości numeryczne były traktowane jako numeryczne, a nie stringi. Skorzystaj z typu number.

Wypróbuj czy teraz poprawnie działa range query.
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-range-query.html

5. Analizatory:
Przykładowy customowy analizator znajduje się w pliku custom_analyzer.json

Utwórz analizator, który skorzysta z angielskiego stemmera, żeby sprowadzić słowa do formy podstawowej i zobacz czy możesz wyszukać termy w formie podstawowej.
https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-stemmer-tokenfilter.html

Utwórz analizator, który pozwoli wyszukiwać dokumenty suffixowo bez konieczności przeglądu całego słownika.

Czy możemy przyspieszyć wyszukiwanie prefixowe kosztem wzrostu rozmiaru indeksu?

zadanie dodatkowe:
zainstaluj plugin stempel
https://www.elastic.co/guide/en/elasticsearch/plugins/current/analysis-stempel.html
wypróbuj działanie ‘polish’ analyzera i filtra tokenów ‘polish_stem‘

6. Agregacje:
Znajdź maksymalny Score dla 10 najczęściej występujących tagów związanych z piłką nożną.
Utwórz kubełki od -20 do 100 ze skokiem 20 i sprawdź w którym kubełku było najwięcej dokumentów.

7. Dashboardy kibana:
Utwórz wizualizację na kibanie wyświetlającą najczęściej używane tagi.
Utwórz wizualizację na kibanie wyświetlającą histogram po dacie utworzenia postu.
Utwórz wyizalizację na kibanie wyświetlającą histogram po liczbie punktów jakie zdobył post.
Utwórz dashboard z tych wizualizacji.

8. Pola nested:
Zaindeksuj dokumenty z autorami I książkami. Imię I nazwisko autora potraktuj jako pole typu nested.
Zadaj zapytanie typu nested o książki dla danego autora.

9. Relacje parent-child
Utwórz relację parent child pomiędzy autorami I książkami. Poszukaj książek napisanych przez autora. Użyj zapytania has_child.

Zaindeksuj za pomocą logstasha posty I komentarze w 1 indeksie. Połącz je relacją parent-child. Użyj filtra mutate, żeby dodać pola relacji do dokumentów. Może być konieczne logstasha 2 razy z różną konfiguracją.
Znajdź użytkowników, którzy dodali najwięcje postów w tagu nba.

10. Klastrowanie
Dobierzcie się w trójki. Połączcie swoje instancje elasticsearcha w klaster. Może być konieczne wyczyszczenie katalogu ‘data’.
Ustawcie ilość replik indeksu na 1. Zatrzymajcie 1 instancję. Spróbujcie zadać jakieś zapytania.
Ustawcie ilość replik na 2. Poczekajcie aż indeksy się zreplikują. Zatrzymajcie 1 instancję. Spróbujcie zadać jakieś zapytania.
Zmieńcie rolę 2 nodów na access node’y. Czy repliki będą zaalokowane?

11. Zadanie dodatkowe
Napisz aplikację korzystającą z elasticsearcha w dowolnym języku programowania. Skorzystaj z oficjalnego api, np:

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>elasticsearch-rest-high-level-client</artifactId>
    <version>6.5.4</version>
</dependency>
```