Date: 14-06-2024

Topic: Elastic search

Elastic Search:

Elasticsearch is a powerful open-source search and analytics engine built on the Apache Lucene library. It is designed to be scalable, distributed, and capable of efficiently handling large amounts of data. Here are some key features and concepts related to Elasticsearch:

1. Full-Text Search: Elasticsearch provides robust full-text search capabilities, enabling complex querying and filtering of text-based data.

2. Distributed Architecture: It is designed to operate in a distributed environment, allowing horizontal scaling across multiple nodes and clusters.

3. Real-Time Search and Analytics: Elasticsearch supports near real-time search and analytics, making it suitable for applications requiring up-to-date information.

4. RESTful API: Interactions with Elasticsearch are performed through a RESTful API, making it accessible via standard HTTP methods (GET, POST, PUT, DELETE).

5. Schema-Free JSON Documents: Data is stored in a schema-free format using JSON documents, providing flexibility in data representation and structure.

6. Indexing and Sharding: Data is indexed for efficient querying. Elasticsearch automatically shards indices across multiple nodes to balance the load and ensure high availability.

7. Aggregation: It provides powerful aggregation capabilities, allowing users to perform complex analytics and create detailed reports from large datasets.

8. Plugins and Extensions: Elasticsearch has a rich ecosystem of plugins and extensions, enabling additional functionalities such as security, monitoring, and language-specific analysis.

9. Kibana Integration: Often used in conjunction with Kibana, a data visualization and exploration tool that allows users to create interactive dashboards and visualizations based on data stored in Elasticsearch.

10. Use Cases: Common use cases for Elasticsearch include log and event data analysis, search engines for applications and websites, metrics analysis, and more.

Elasticsearch is commonly used in various industries for its speed, scalability, and flexibility in managing and analysing large volumes of data.

Steps to create a index:



For creating an index

**es.indices.create(index = ‘index\_name’)**

If you want to check, if the index exists or not

**es.indices.exists(index = ‘index\_name’)**

If you want to delete the index

**es.indices.delete(index = ‘index\_name’)**

If you want to allocate the ID to the documents

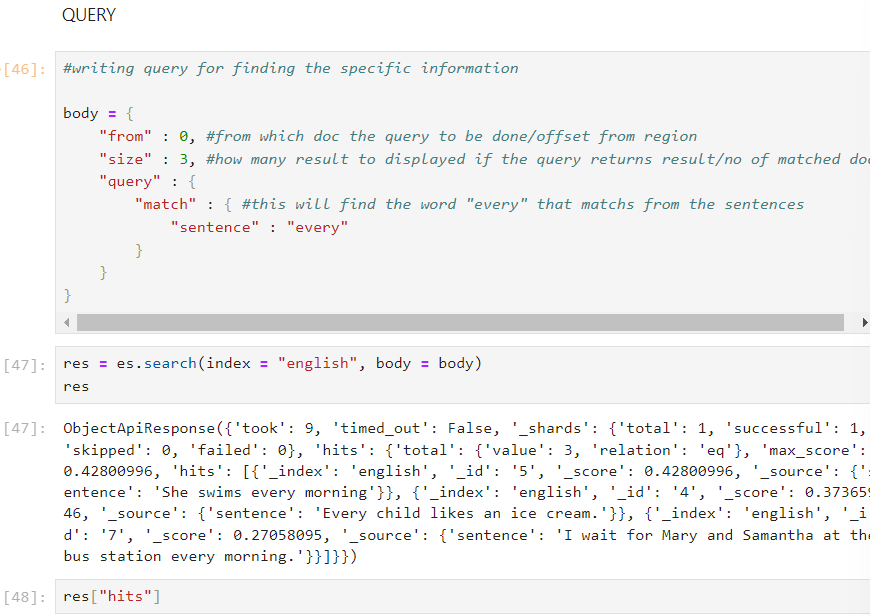
**es.index(index="person", id = 1, body= Data\_1)**

If you want to get the data using the id

**res = es.get(index = "person", id = 2)**

**Search required data**

es.search(index = , body = )



We can get how many documents there are in a single index using count()

**count\_doc = es.count(index = "english")**

**Combined Query:**



#regexp - regular expression, Regular expressions allow for pattern matching within strings, which is useful for complex string searches.

“ #^start.” - matches any sentence that starts with the word "start". The caret ^ indicates the beginning of the string.

Mapping:

Mapping is the process of defining how a document and the fields it contains are stored and indexed.

It enables faster search retrieval and aggregations. Hence, your mapping defines how effectively you can handle your data. A bad mapping can have severe consequences on the performance of your system.

We can map the datatype to their data in existing index



**Bulk Indexing:**

Bulk indexing refers to the process of inserting or updating multiple documents into an Elasticsearch index in a single API call, rather than making individual calls for each document. This method is designed to improve efficiency and performance when handling large volumes of data.

1. Import necessary packages like Elasticsearch, helpers, pandas, json.
2. Load a file in the environment.
3. Convert into a json file format.

# Save it as a JSON file

**df.to\_json('nyc\_traffic\_incidents.json', orient='records', date\_format = 'iso', lines=True)**

1. Covert the date column into proper datetime format.

**df['crash\_date'] = pd.to\_datetime(df['crash\_date'])**

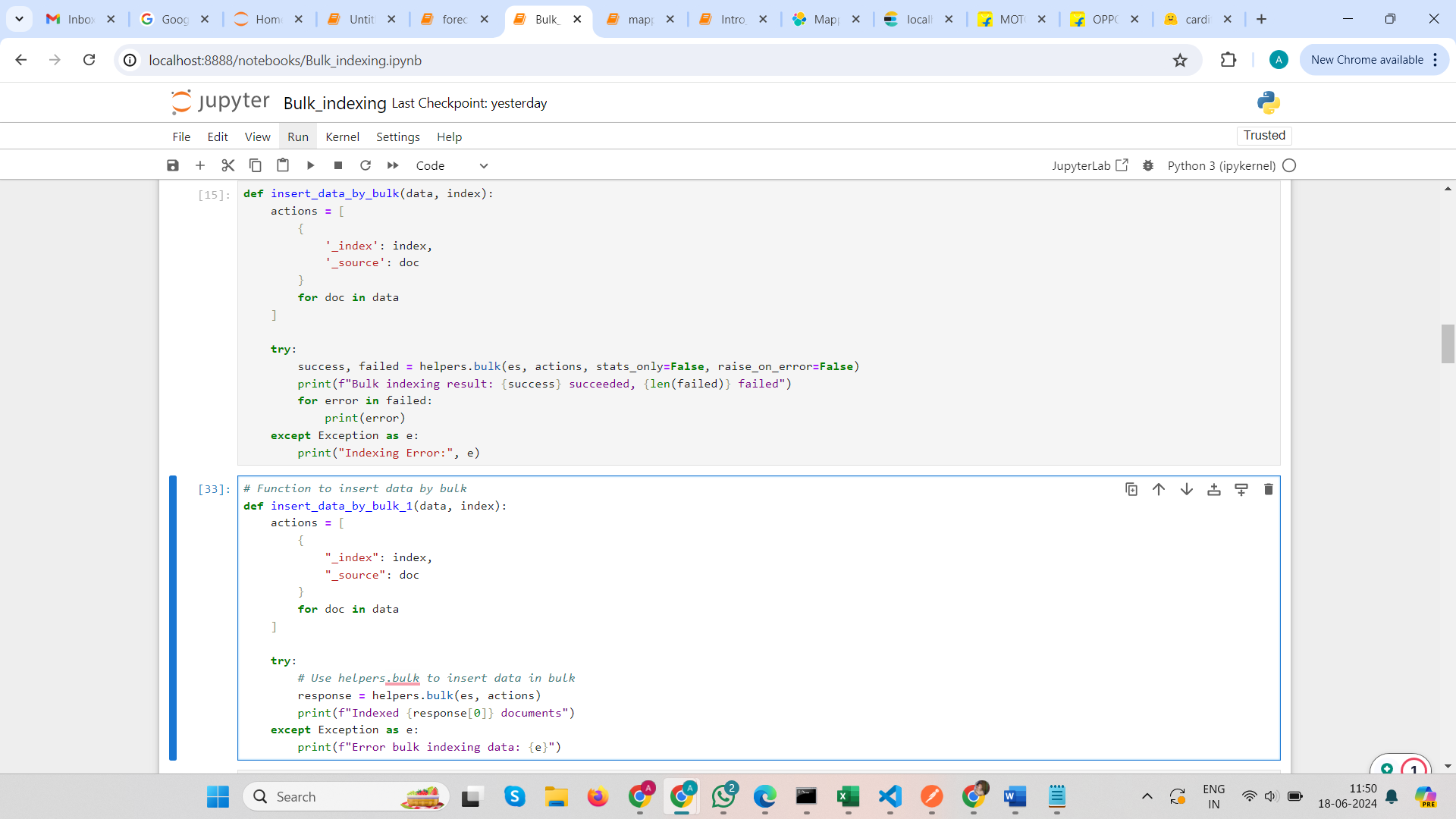
1. Define the body which means the mapping



1. Then, create an index

**es.indices.create(index=index, body= INDEX\_BODY)**

1. Insert the data into an index at a single call



can use these two functions also but the main action is or the step is

**helpers.bulk(es, actions)**

**actions** refer to a list of dictionaries where each dictionary represents a single action to be performed in the bulk indexing operation for Elasticsearch.