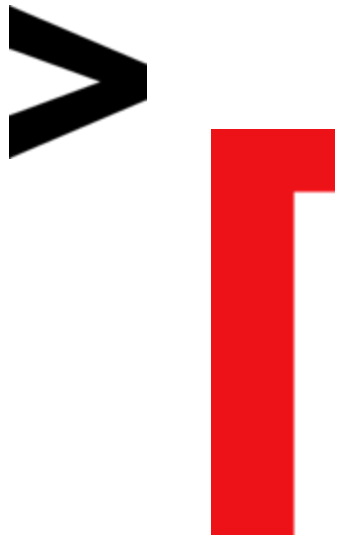
Couchbase

5교시.
Vector Search 개요, 실습

1. AI & Vector Search
2. Hybrid Search
3. Vector Search 실습 : RGB 모델

5-1. AI & Vector Search



Key Milestones in the History of AI | 20th Century

AI is born | **Early Developments** | **Expert Systems** | **Machine Learning**

Can a Machine think?

1950

Alan Turing test - if a machine tricks a human into thinking it's a human, then it's intelligence.

AI term coined

1956

John McCarthy introduced the term 'Artificial Intelligence' at The Dartmouth Conference.

Gen. Problem Solver

1957

General Problem Solver is the first AI algorithm. It can solve formalized problems (e.g. Towers of Hanoi).

The chatbot Eliza

1965

Eliza is the first chatterbot - or "chatbot" modernly. It can simulate a human conversation.

1st AI Winter
1974-1980

R1, an Expert System

1980

R1 uses 2500 rules to ensure that the customer's order is complete, saving the company $25M a year.

2nd AI Winter
1987-1993

Deep Blue

1997

IBM's Deep Blue supercomputer defeats the world chess champion Garry Kasparov.

Kismet

1998

Cynthia Breazeal at MIT creates Kismet, a robot that can detect and respond to people's feeling.

Key Milestones in the History of AI | 21st Century

Machine Learning | Deep Learning | Generative Models

Watson

2006

IBM's Watson question-answering system is created. In 2011, it defeats Jeopardy!'s champion.

Siri & Alexa

2011

Apple released Siri, an AI assistant that can respond to voice. 3 years later, Amazon launched Alexa.

Google AI

2014

Google AI recognizes cats from 10 million Youtube videos with almost 75% accuracy in 3 days.

AlphaGO

2016

AlphaGo defeats top Go player Lee Sedol in Seoul. Go is incredibly difficult given the vast number of positions.

AlphaFold

2020

DeepMind's AlphaFold system can predict protein structure, with implications for drug discovery and biology.

AI Boom

AI Explosion

DALL.E

2021

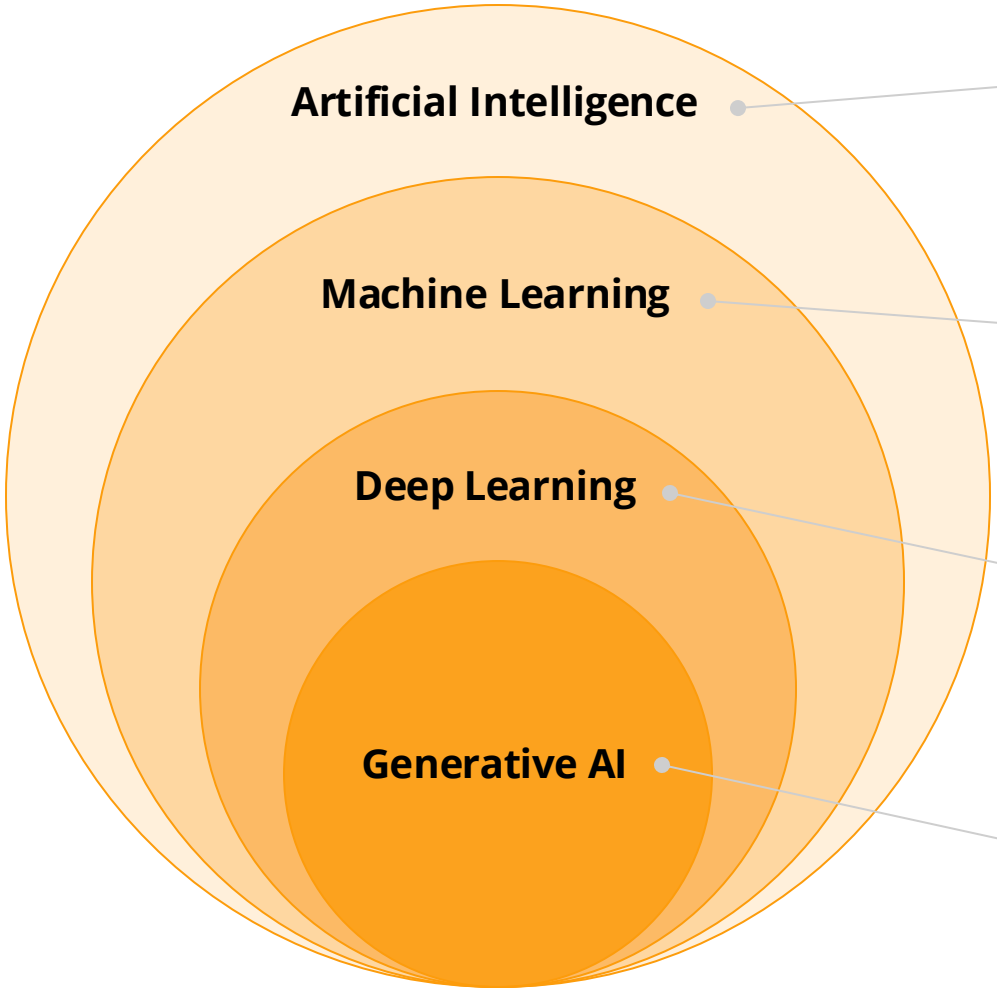
DALL·E, and a year later Midjourney and StableDiffusion, can generate images from textual descriptions.

ChatGPT

2022

OpenAI releases the AI chatbot ChatGPT. It is based on the Large Language Model GPT-3 created in 2020.

The Technology behind AI



Artificial Intelligence (AI)
Techniques that allows computers to emulate human behavior (e.g. learn, recognize patterns, solve complex problems).

Machine Learning (ML)
A subset of AI, using advanced algorithms to detect patterns in large data sets, allowing machines to learn and adapt for prediction or content generation use cases.

Deep Learning (DL)
A subset of ML, using multiple layers of artificial neural networks that simulate human brains for in-depth data processing.

Generative AI (GenAI)
A subset of DL, using models that generate content like text, images, or code based on provided input.

Powering Apps: A Combination of Predictive & Generative AI

Predictive AI

Outcomes and Insights driven by ML

- Predict Outcomes based on historical data
- Utilize ML algorithms for pattern recognition
- Learns patterns and correlations from data
- Drives decision making and Future planning
- High ROI, trained on proprietary data

- Predictive Insights
- Dynamic Pricing
- Fraud Detection
- Inventory Optimization

+

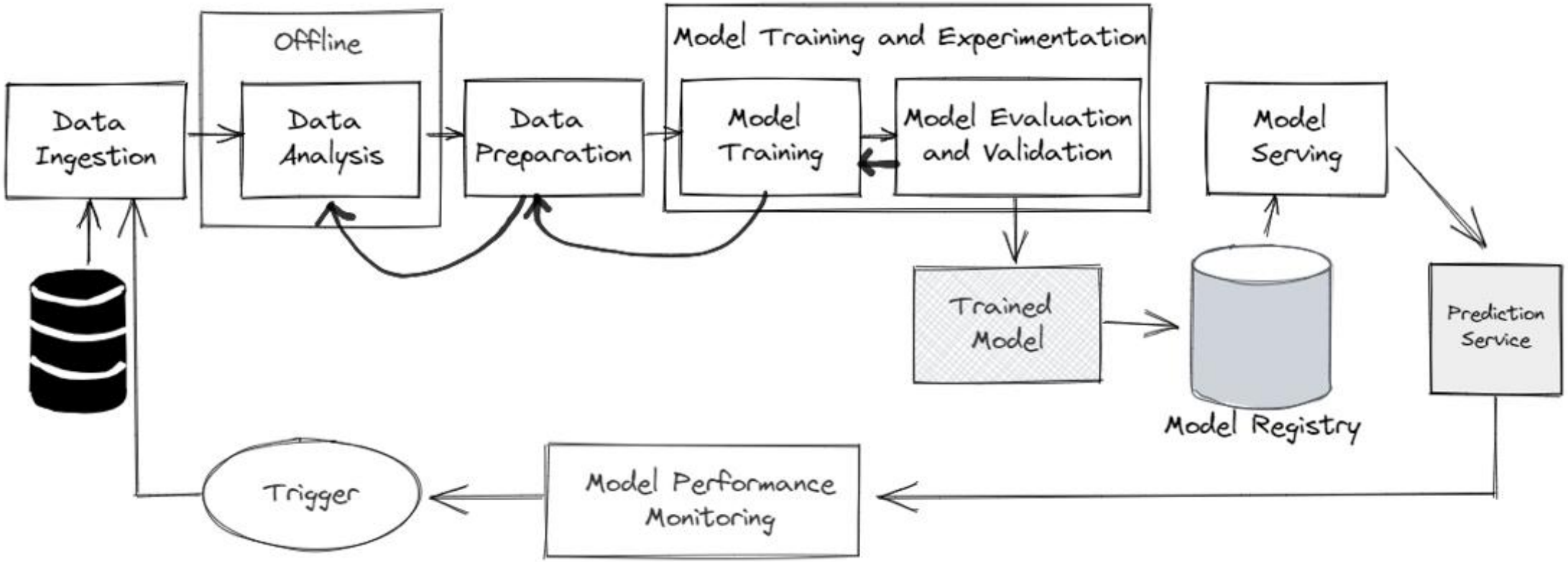
Generative AI

Generate Content and Experiences

- Generate or Synthesize content
- Needs large amounts of unlabeled data for training
- Generates new data probabilistically
- Fosters creativity, innovation
- Accelerates human productivity

- Hyper-personalized experiences
- Contextualized content
- Chatbots and CoPilots
- Synthetic data and Summarization

Model? Machine Learning Workflow



출처 : https://www.iguazio.com/blog/ml-workflows-what-can-you-automate/
https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

What is a Vector

What is a Vector? | Basic RGB Example

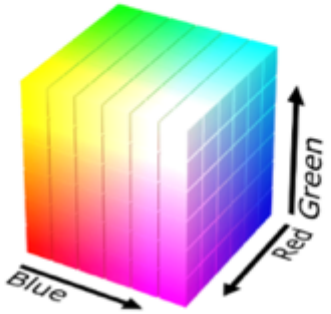
This is a vector

2.6	11.3	-4.2

First value | Second value | Third value

Here, it contains 3 values
=> its dimension is 3

The RGB model example

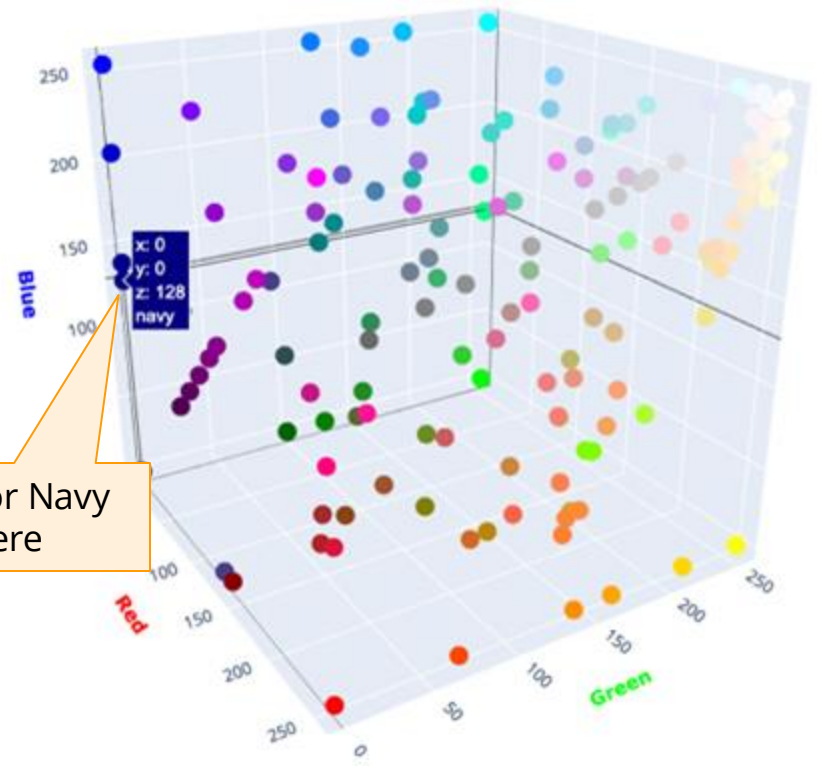


The model used to create the colors you see on TV and computer screens.
Each color is the addition of a **R**ed, **G**reen and **B**lue primary colors.

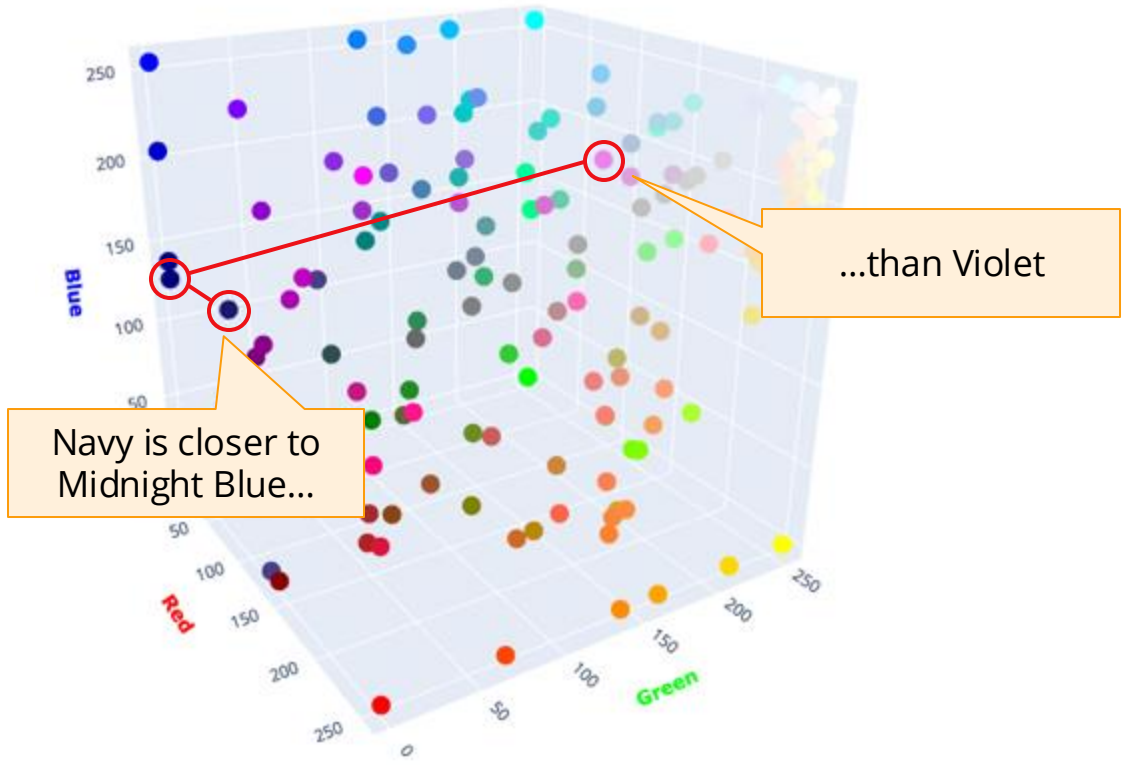
A Vector is a just an **array of numerical values**

Vectors Similarity

Example of 123 vectors of RGB colors



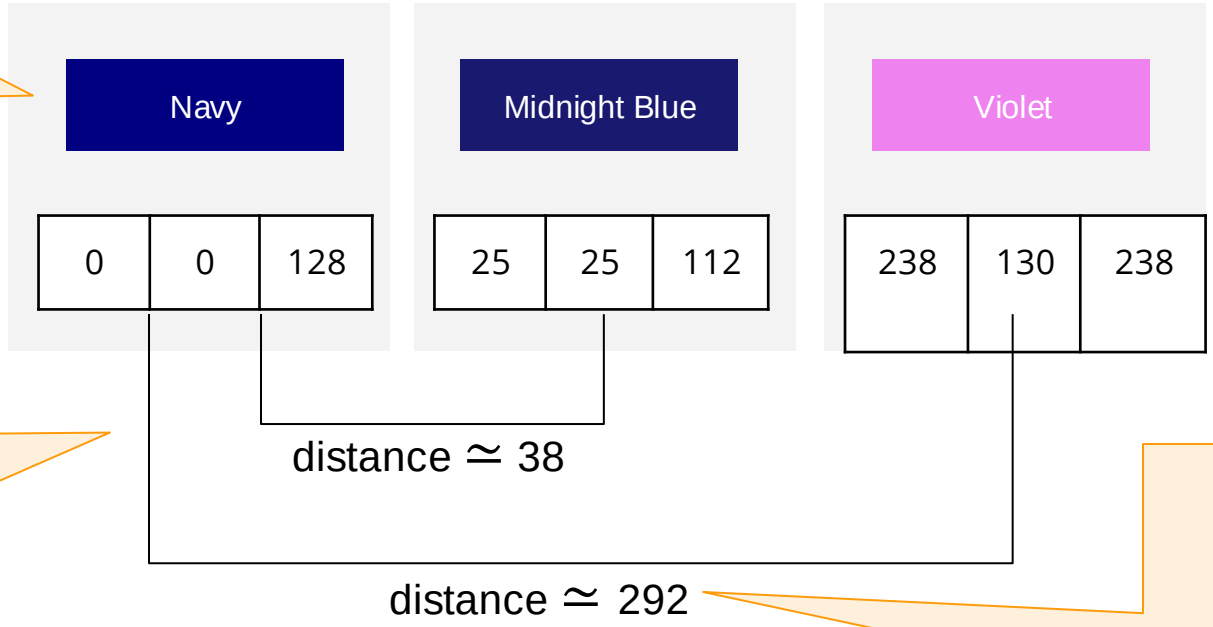
Similar colors are closer to each other



Vectors make it possible to translate **similarity** as perceived by humans to **proximity in a vector space**.

How does Similarity works

To the human eyes,
Navy is closer to Midnight Blue than Violet



Mathematically,
we got the same result by comparing the vectors

Vectors are compared using **a similarity distance.**

Here the *euclidean distance*

$292 \simeq \sqrt{(238-0)^2+(130-0)^2+(238-128)^2}$

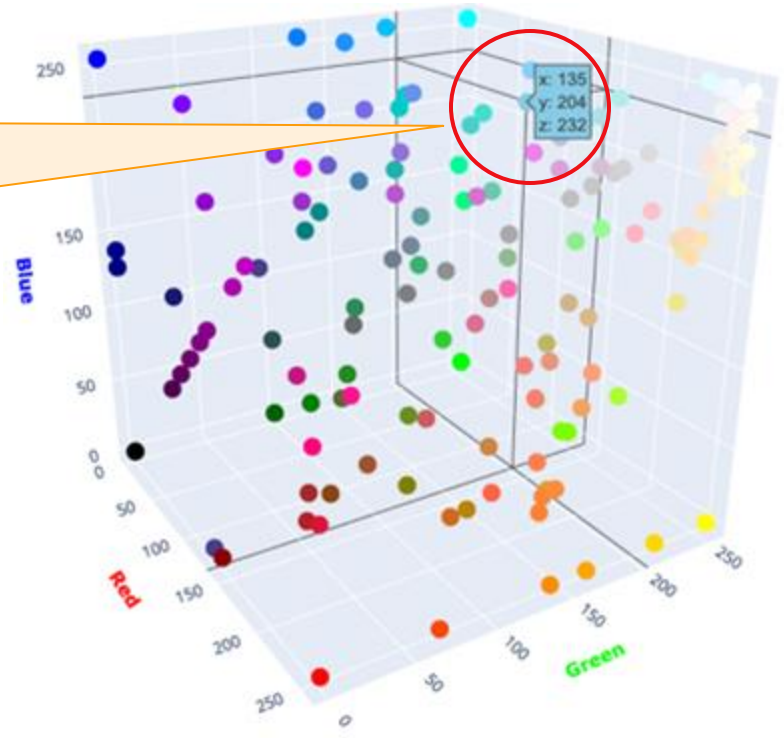
Vectors can easily be compared mathematically using a **similarity distance**

Similarity Search with K-NN (K-Nearest Neighbors)

Which are the top k nearest neighbors to this color?

[135,204,232]

Example of 123 vectors of RGB colors

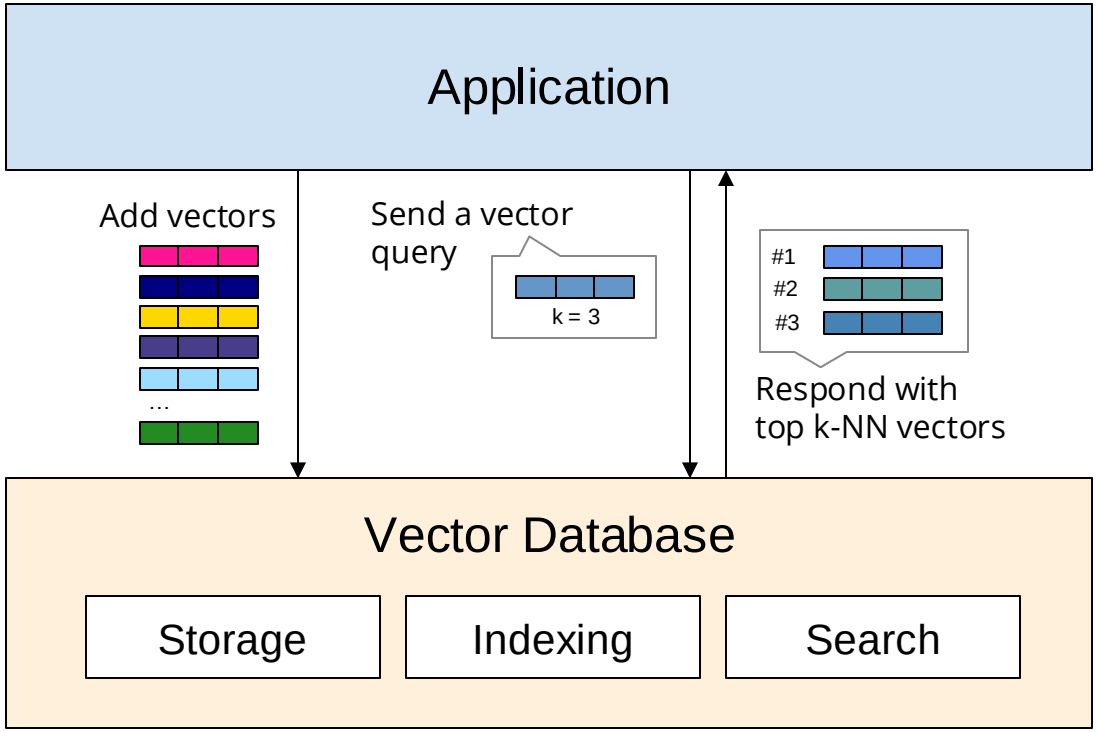


Top k-NN results of the query

	sky blue
#1	[135,206,235]
	light sky blue
#2	[135,206,250]
	light blue
#3	[173,216,230]

A similarity search is a query that **finds the k nearest neighbors to a vector**, as measured by a similarity metric

What is Vector Database



Vector databases provide the ability to **store, index and search vectors** using similarity search

Couchbase Vector Search

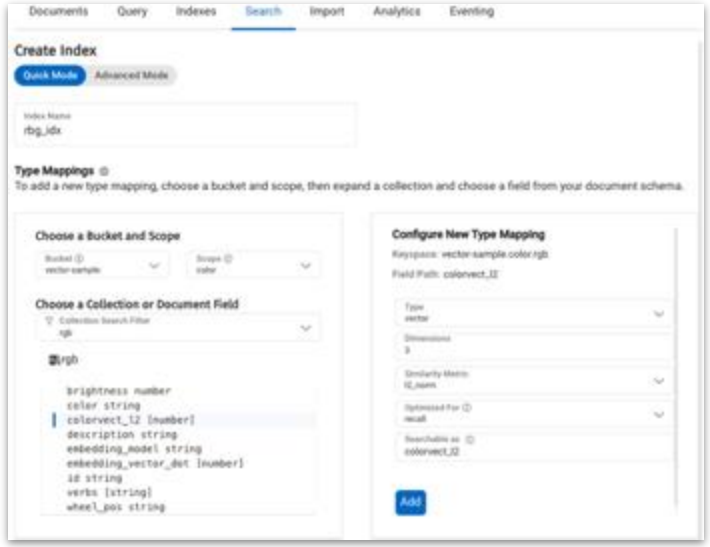
The vectors are stored as **a field in JSON** documents

```
{
 "id": "#000080",
 "color": "navy",
 "brightness": 14.592,
 "colorvect_l2": [0, 0, 128],
 "description": "Navy is a deep, rich color that
     exudes sophistication. It is a dark shade of
     blue that is often associated with authority,
     stability, and elegance. Navy is a versatile
     color that can be both bold and understated,
     making it a popular choice in fashion and
     interior design. It is a timeless color that
     never goes out of style and adds a touch of
     sophistication to any look or space.",
 }
```

JSON Storage

Data Service

A **Vector Index** must created to allow the vectors to be searched



Vector Index

Search Service

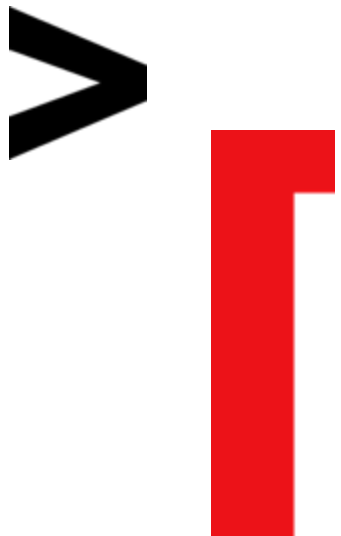
A **Vector Query** can now search for the top k-NN of a color

```
{
  "query": { "match_none": {} },
  "knn": [
    {
      "field": "colorvect_l2",
      "vector": [135, 204, 232],
      "k": 3
    }
  ],
  "fields": ["color"]
}
```

Vector Query

Couchbase uses the **Data Service to store vectors**, and the **Search Service to index and query vectors**

5-2. Hybrid Search



Hybrid SQL++ and Vector Search with Couchbase

This is a SQL++ query

Combining Vector Search query

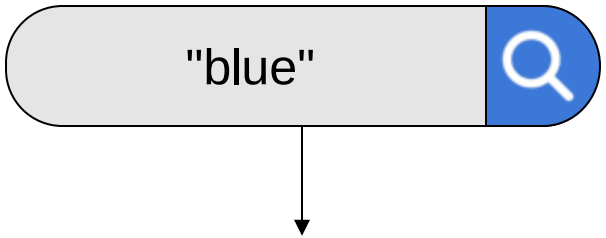
And standard SQL++ criteria

```
SELECT color, brightness
FROM `vector-sample`.color.rgb AS t1
WHERE
    SEARCH(t1,
     {
       "query": {  "match_none": {} },
       "knn": [{
         "field": "colorvect_l2",
         "vector": [135,204,232],
         "k": 3 }]
     }
AND
    brightness >= 180 AND brightness <= 190
    )
```

- ✓ SQL++ is easy and familiar to developers
- ✓ You can filter vector search results with other criteria
- ✓ **You don't have to run 2 separate databases, one for Documents and one for Vector Search!**

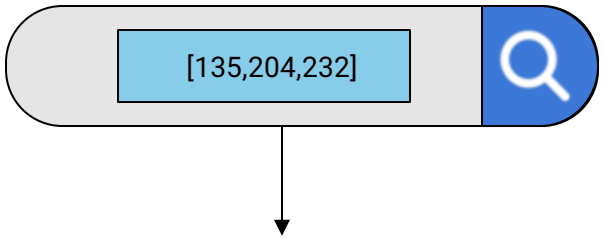
Couchbase can run hybrid SQL++ and Vector Search queries to **facilitate application development**

Comparison between Keyword Search and Vector Search



Keyword Search on the
description of the colors

- Dodger blue
- Blue violet
- Powder blue
- Sky blue



Vector Search on the
RGB vectors of the colors

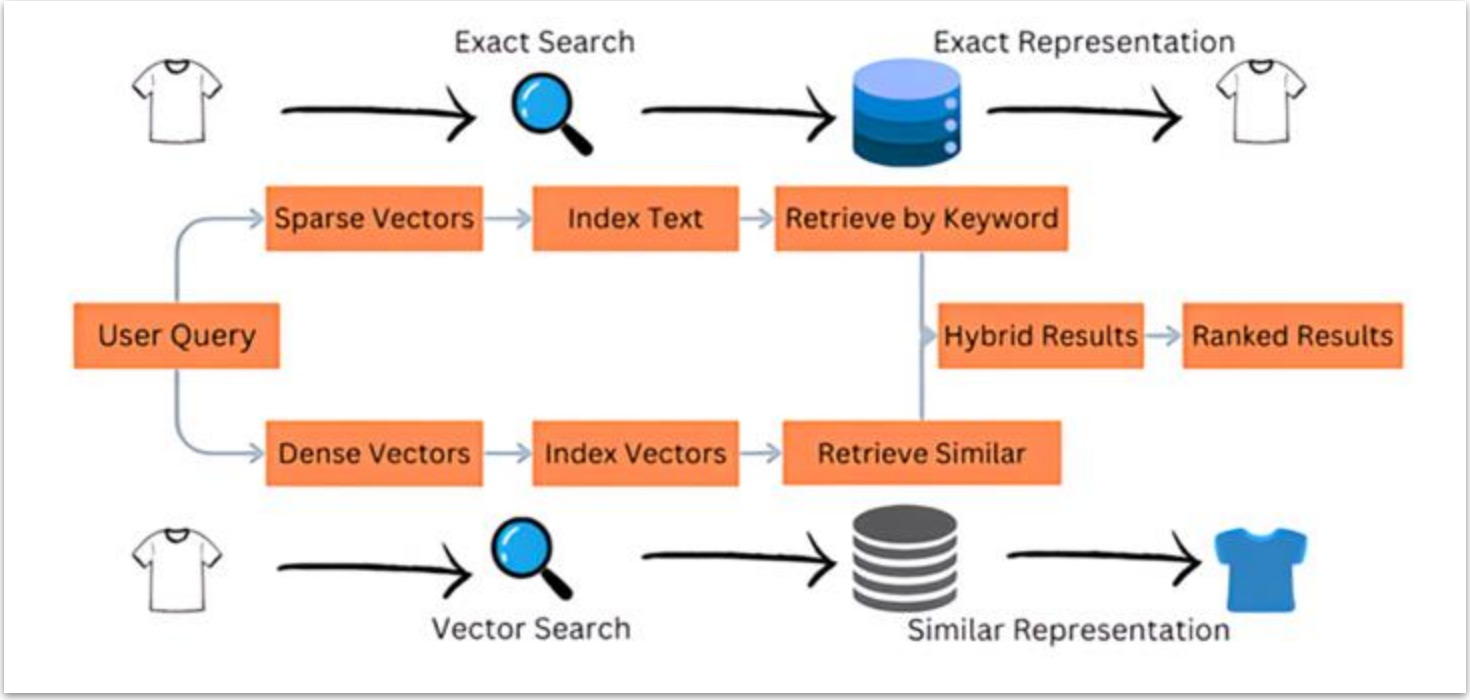
- Sky blue
- Light sky blue
- Light blue
- Light steel blue

A Keyword search looks for **terms** that match

A Vector search looks for **similarity**

Hybrid Search to get the best of both worlds

Hybrid Search Architecture



Hybrid Search with Couchbase

```
{
  "query": {
     "match": "blue",
     "field": "description"
  },
  "knn": [
    {
   "field": "colorvect_l2",
    "vector": [135,204,232],
    "k": 4
    }
  ],
  "fields": ["color","description"],
  "size": 4
}
```

Keyword search

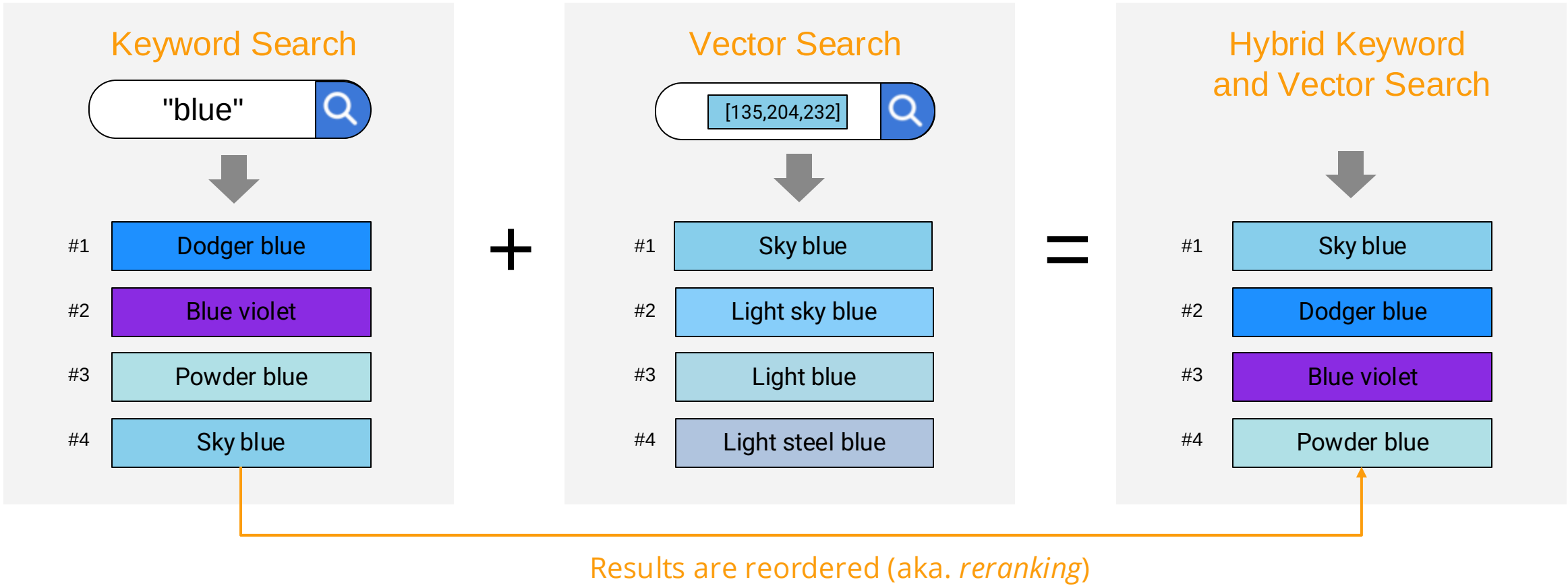
Vector search

Results to return

Vector search in conjunction with traditional Keyword search delivers the most complete and relevant results

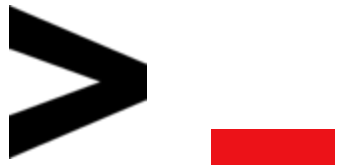
Source: Couchbase

Hybrid Keyword and Vector Search Example

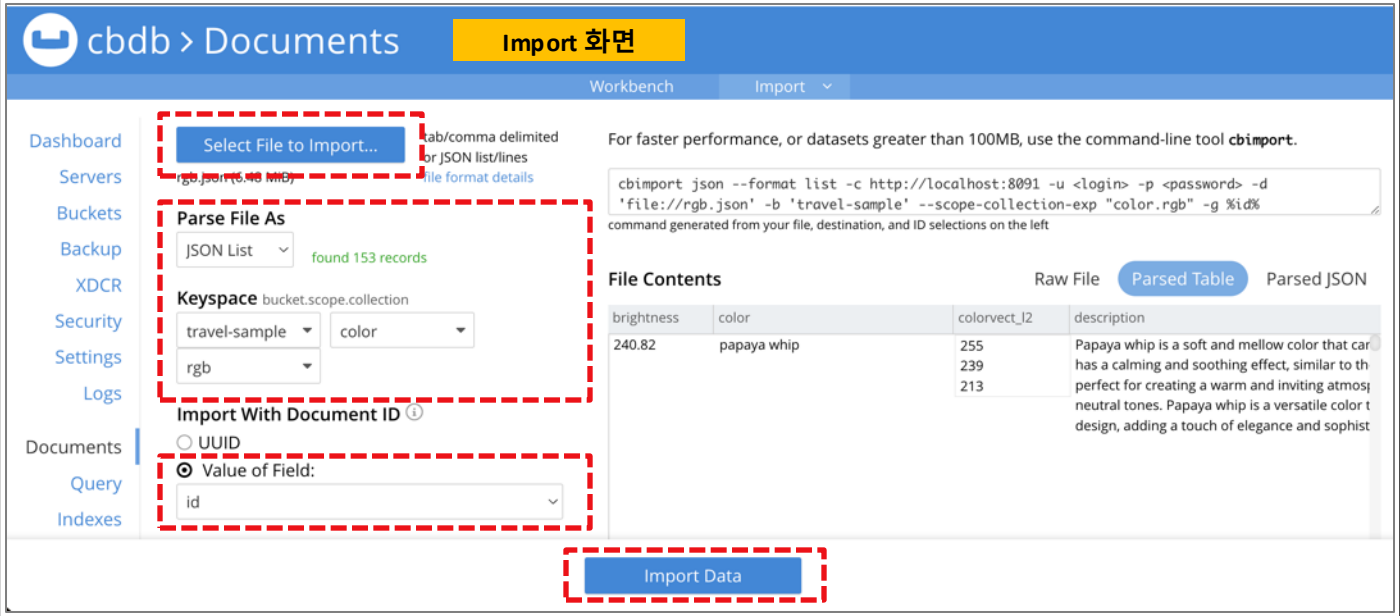


Results from the Keyword search are **boosted** if they appear in the Vector Search results

5-3. Vector Search 실습 : RGB

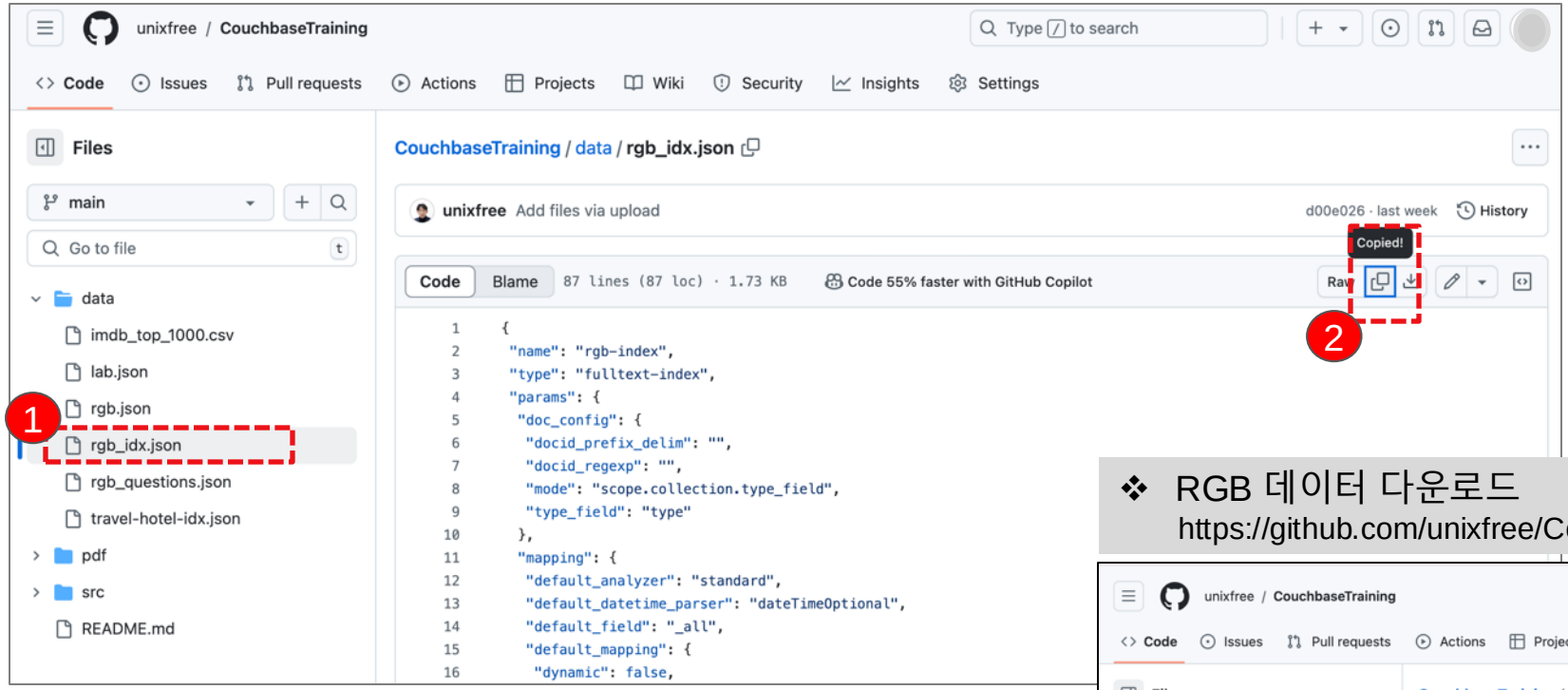


실습

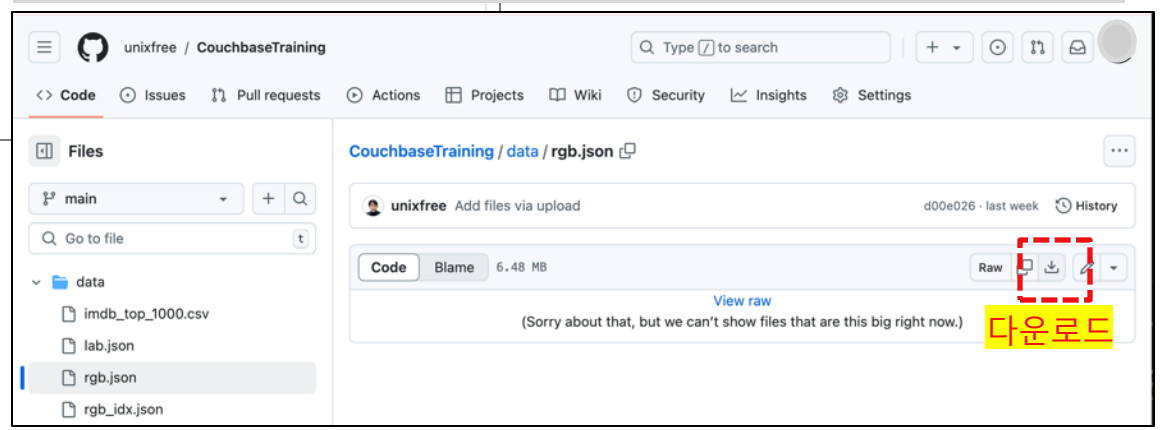
	실습 항목	상세 실습 내용	기타
1	Scope, Collection 생성	1. travel-sample 버킷에 2. color Scope 생성 3. rgb Collection 생성	travel-sample > color > rgb
2	Data Import	1. Github에 있는 json 파일을 내 노트북으로 다운로드 https://github.com/unixfree/CouchbaseTraining/blob/main/data/rgb.json 2. Couchbase Document 페이지 이동. 3. Import 선택 4. Import 수행, Document ID 는 id 로 지정	
3	**검색 Index 생성**	• 21페이지에서 24페이지 참고하여 인덱스 생성 https://github.com/unixfree/CouchbaseTraining/blob/main/data/rgb_idx.json	
4	Vector Search 수행	• 26페이지에서 31페이지 참고하여 벡터 검색 수행	
5	**검색 Index 수정**	• 32페이지에서 33페이지 참고하여 벡터 검색 수행	
6	Vector Search 수행	• 34페이지에서 38페이지 참고하여 벡터 검색 수행	

실습 > Inverted Index 생성

1. 인덱스 설정 파일 복사
   https://github.com/unixfree/CouchbaseTraining

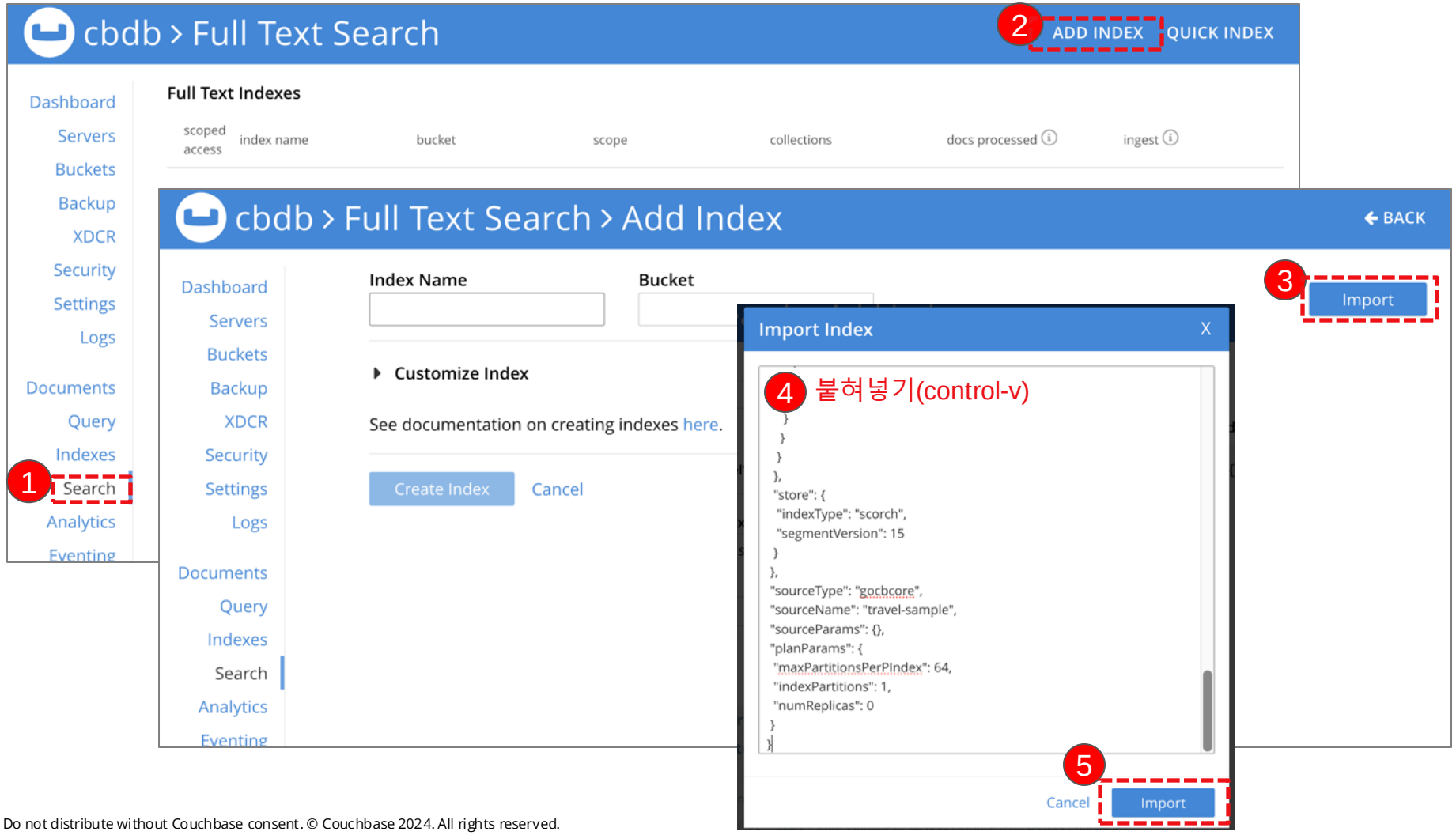


❖ RGB 데이터 다운로드
https://github.com/unixfree/CouchbaseTraining/blob/main/data/rgb.json



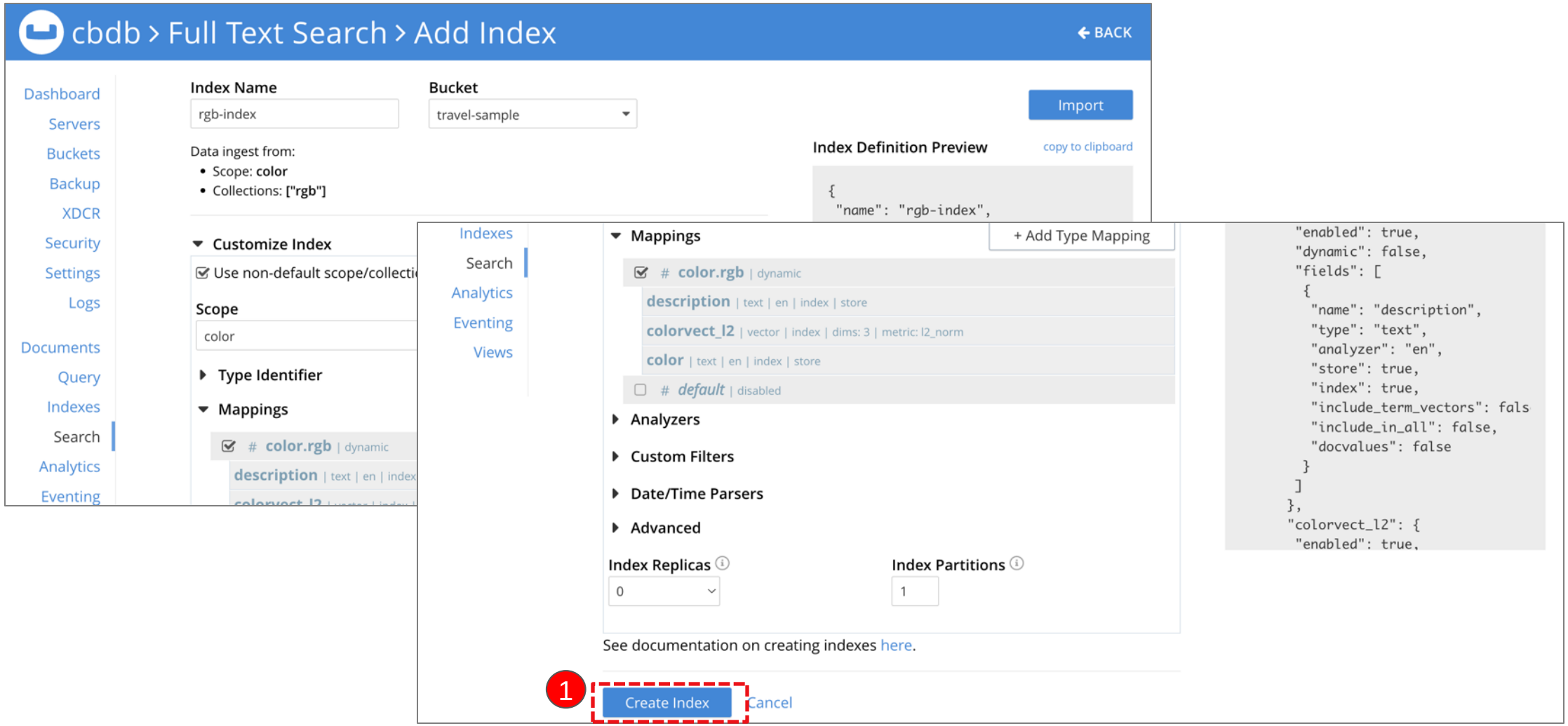
실습 > Inverted Index 생성 (계속)

2. 인덱스 설정 파일 Import로 인덱스 생성
   1) Search > 2) ADD INDEX > 3) Import > 4) 붙혀넣기(control-v) > 5) Import



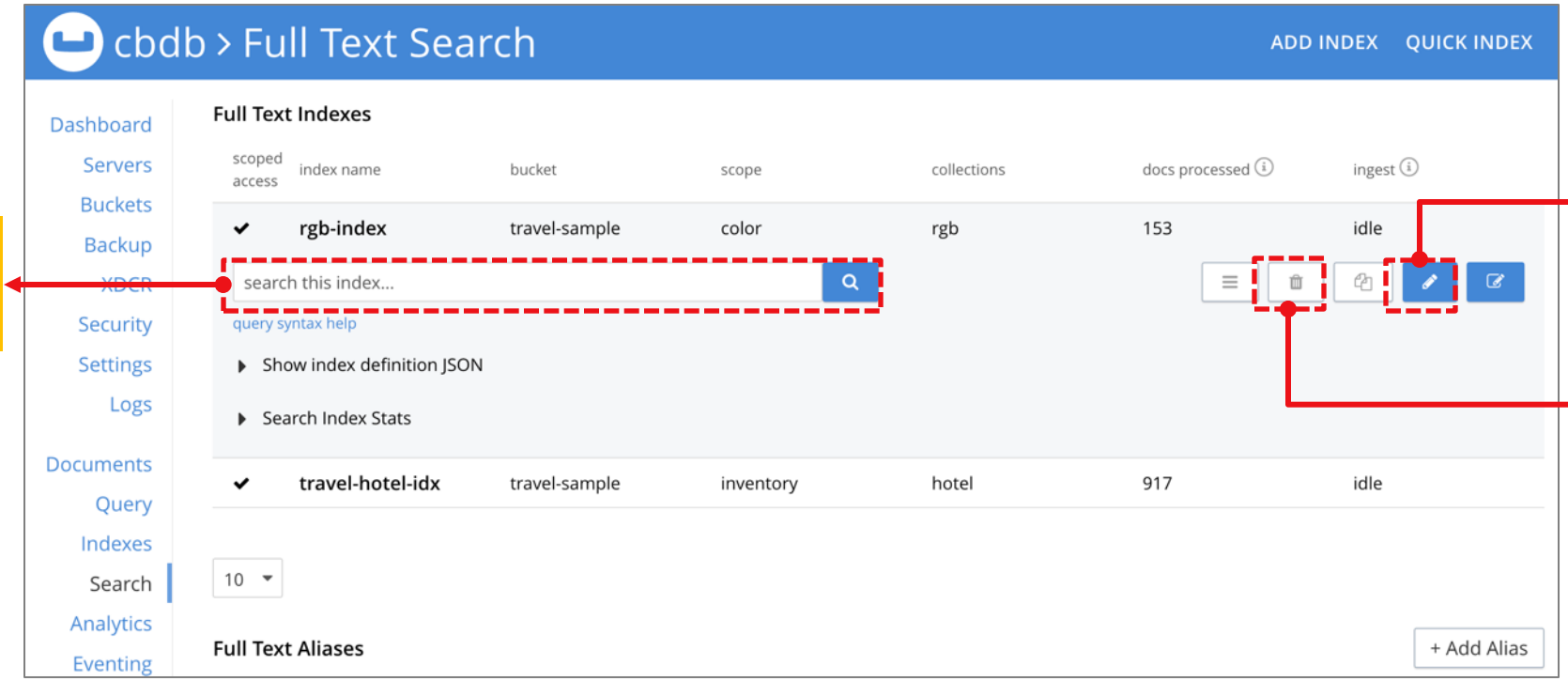
실습 > Inverted Index 생성 (계속)

2. 인덱스 설정 파일 Import로 인덱스 생성
   1) 아래와 같이 각 항목이 채워짐 > 2) 화면 스크롤 다운 후, 3) Create Index 클릭



실습 > Inverted Index 생성 (계속)

2. 인덱스 설정이 완료.
   인덱스가 만들어 지면 아래와 같이 화면(UI)가 보임.



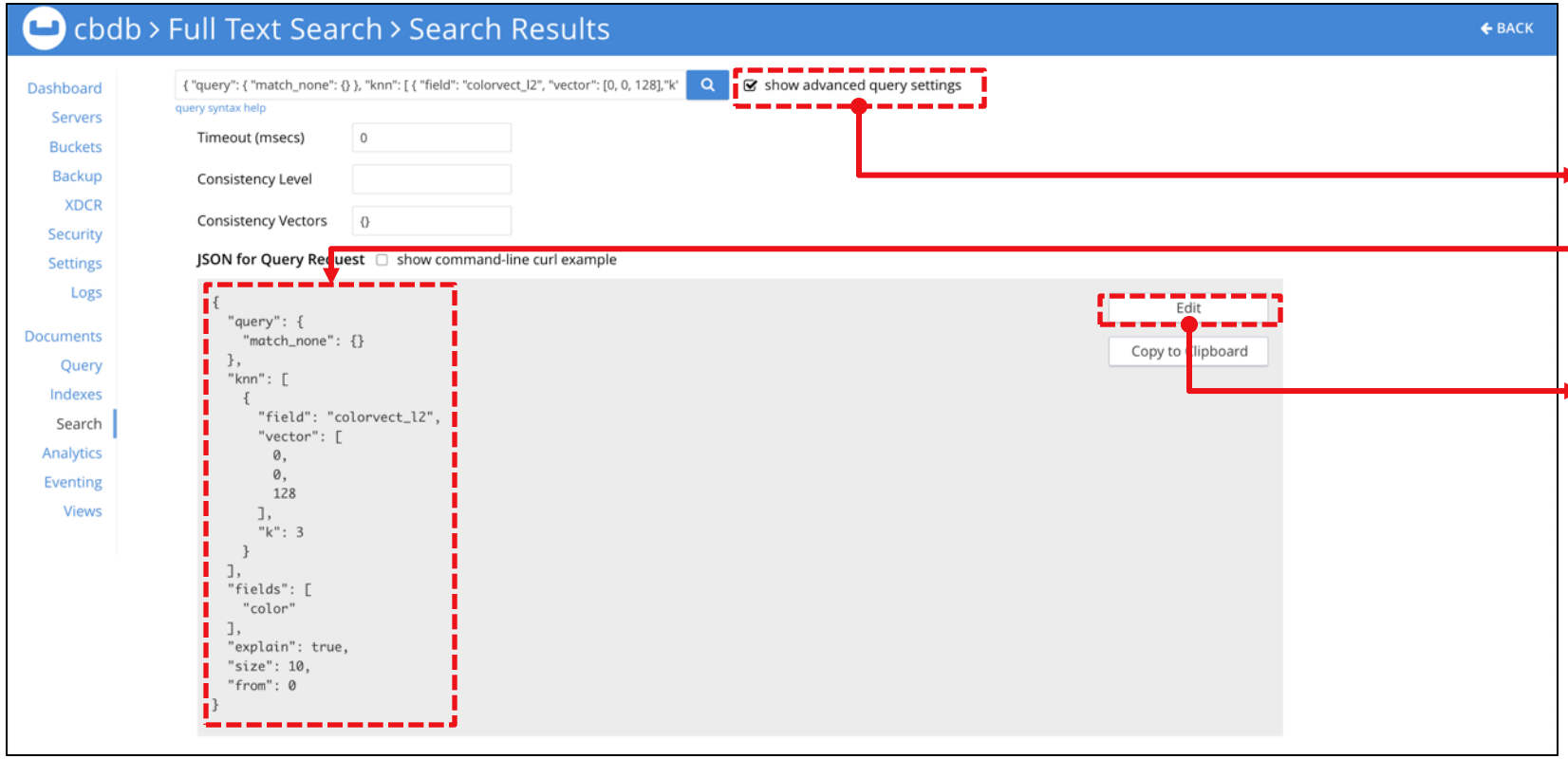
실습 > color.rgb 데이터 설명


```
{
 "id": "#000080",
 "color": "navy",
 "brightness": 14.592,
 "verbs": ["deep", "rich", "sophisticated"],
 "colorvect_l2": [0, 0, 128],
 "description": "Navy is a deep, rich color that exudes sophistication. It is
    a dark shade of blue that is often associated with authority, stability,
    and elegance. Navy is a versatile color that can be both bold and
    understated, making it a popular choice in fashion and interior design. It
    is a timeless color that never goes out of style and adds a touch of
    sophistication to any look or space.",
 "embedding_model": "text-embedding-ada-002-v2",
 "embedding_vector_dot": [
   0.0021118249278515577,
   -0.005944395903497934,
   … 1533 omitted for space
   -0.018224267289042473
 ],
 "wheel_pos": "other"
}
```


- **id**: the hex code of the color
- **color**: the name of the color
- **brightness**: a calculation of the brightness to the human eye
- **colorvect_l2**: vector based on the RGB color
- **description**: a text describing the color
- **embedding_model**: the model used to encode the embedding_vector_dot vector
- **embedding_vector_dot** : vector based on the field “description” encoded via the text-embedding-ada-002 OpenAI model
- **verbs**: list of qualifiers

실습 > Vector Search (RGB)

```
{
"query": { "match_none": {} },
"knn": [
{ "field": "colorvect_l2", "vector": [0, 0, 128],"k": 3 }
],
"fields": ["color"]
}
```

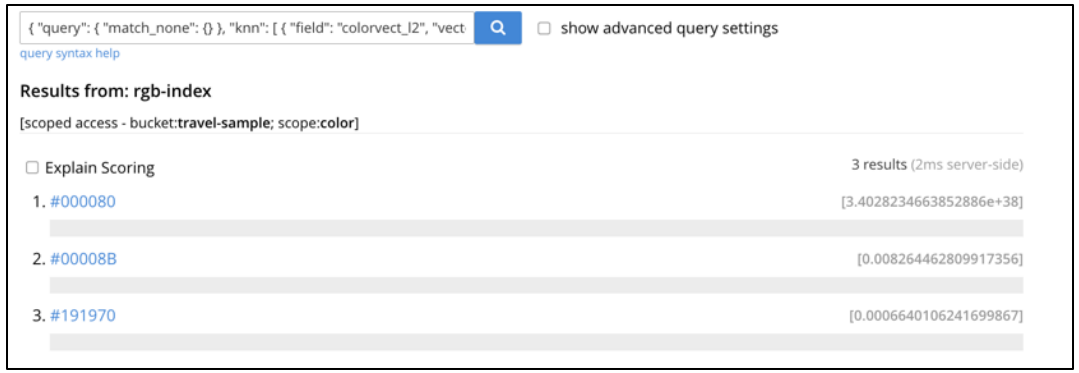


완전한 검색 Query 확인

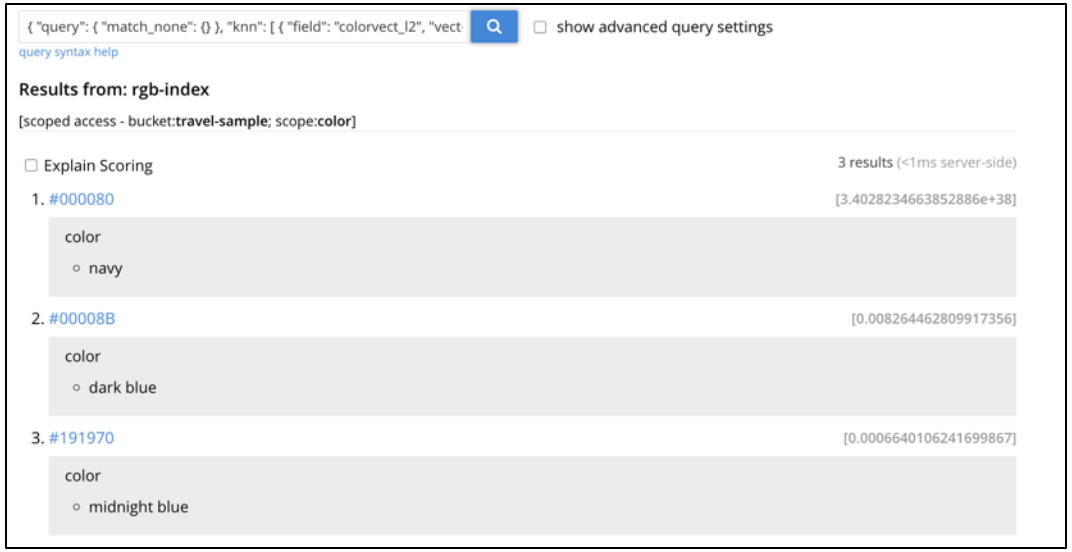
완전한 검색 Query 수정

실습 > Vector Search (RGB)

```
{
"query": { "match_none": {} },
"knn": [
{ "field": "colorvect_l2", "vector": [0, 0, 128],"k": 3 }
]
}
```



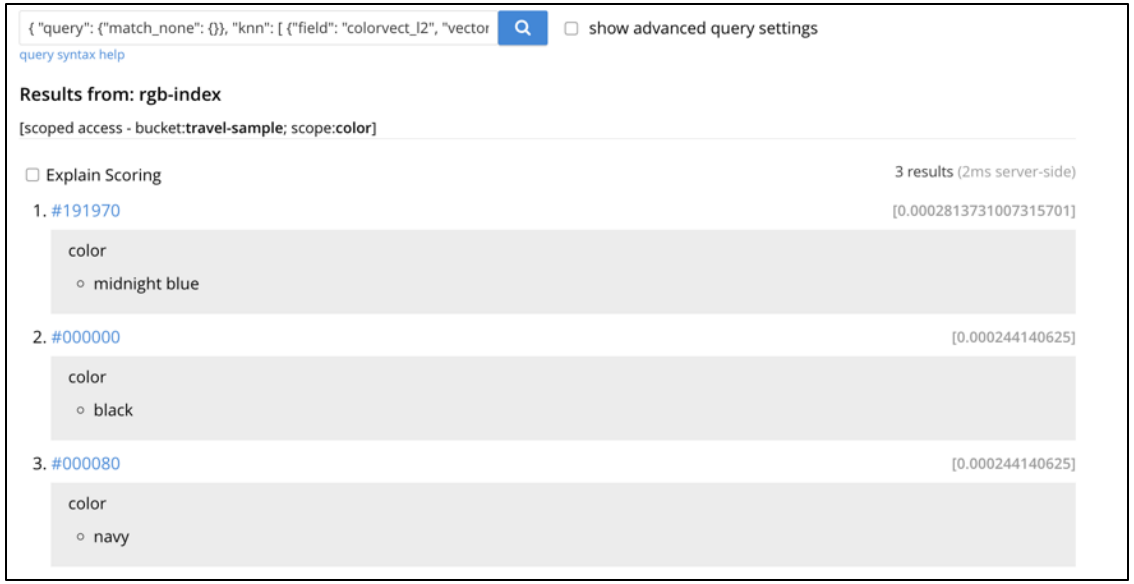

```
{
"query": { "match_none": {} },
"knn": [
{ "field": "colorvect_l2", "vector": [0, 0, 128],"k": 3 }
],
"fields": ["color"]
}
```



실습 > Vector Search (RGB)

- 유사도에 따른 결과 수 지정 : 3

```
{
"query": {"match_none": {}},
"knn": [ {"field": "colorvect_l2", "vector": [0,0,64], "k": 3}],
"fields": ["color"]
}
```



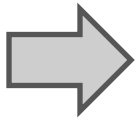
- 유사도에 따른 결과 수 지정 : 153

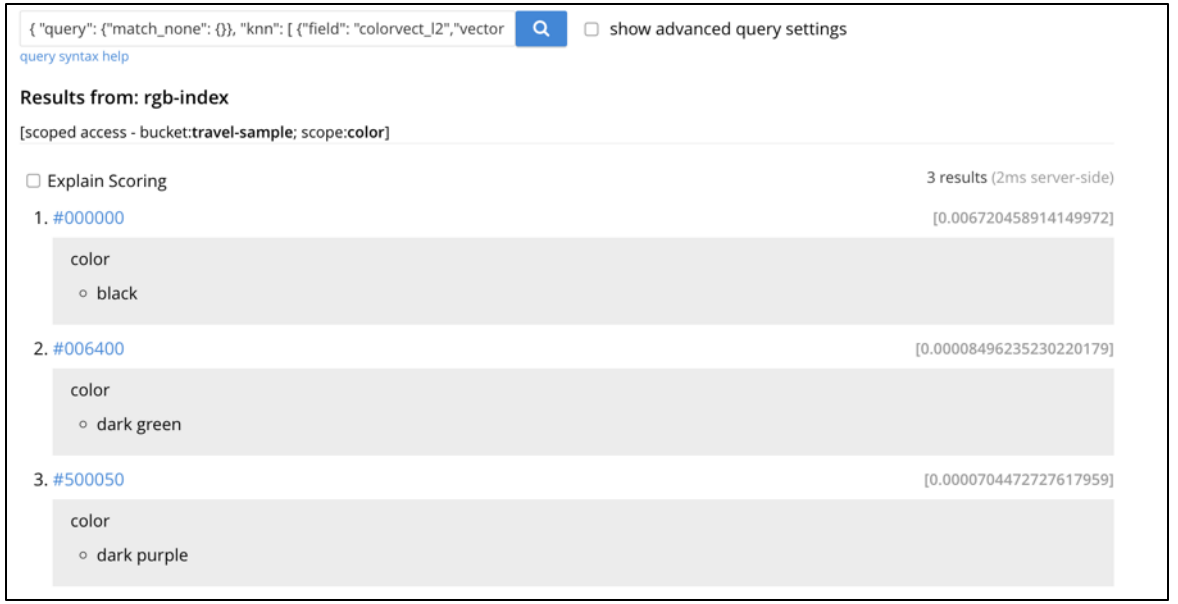
```
{
"query": {"match_none": {}},
"knn": [ {"field": "colorvect_l2", "vector": [0,0,64], "k": 153}],
"fields": ["color"]
}
```

실습 > Vector Search (RGB)

- 없는 벡터 값

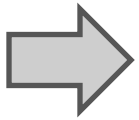
```
{
"query": {"match_none": {}},
"knn": [
{"field": "colorvect_l2","vector": [1.241999836687228,-8.105597595174153,-9.030840590925482],
"k": 3} ],
"fields": ["color"]
}
```

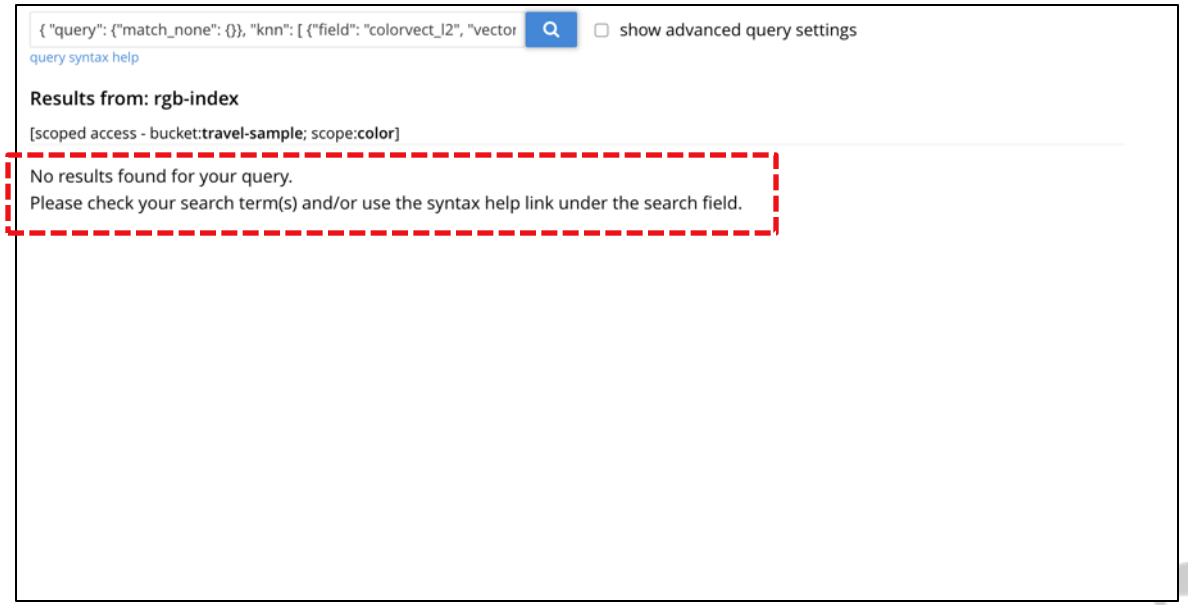




- 벡터 차원(dimension)이 다를 때

```
{
"query": {"match_none": {}},
"knn": [
{"field": "colorvect_l2",
"vector": [0,64],
"k": 3} ],
"fields": ["color"]
}
```

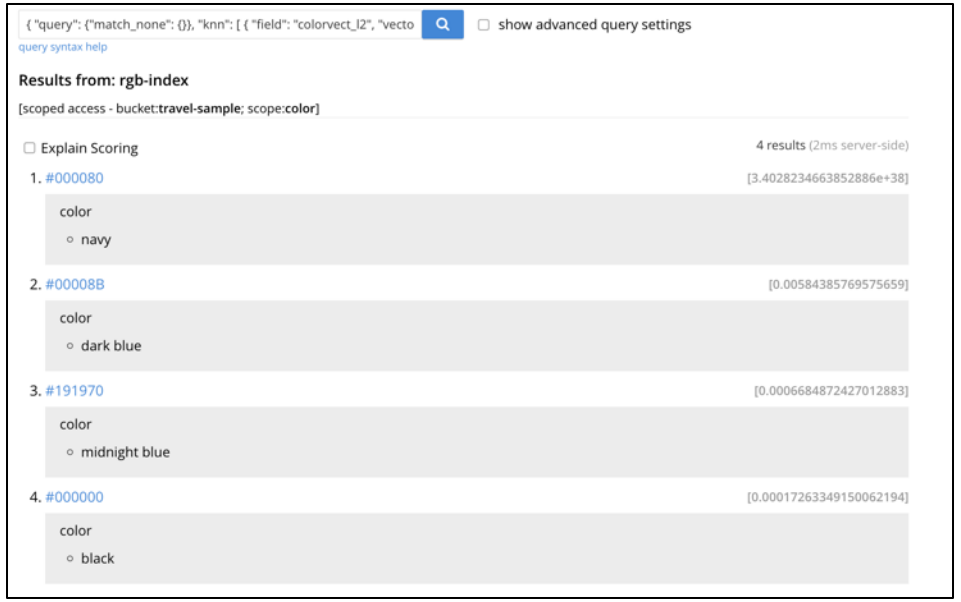




실습 > Vector Search (RGB)

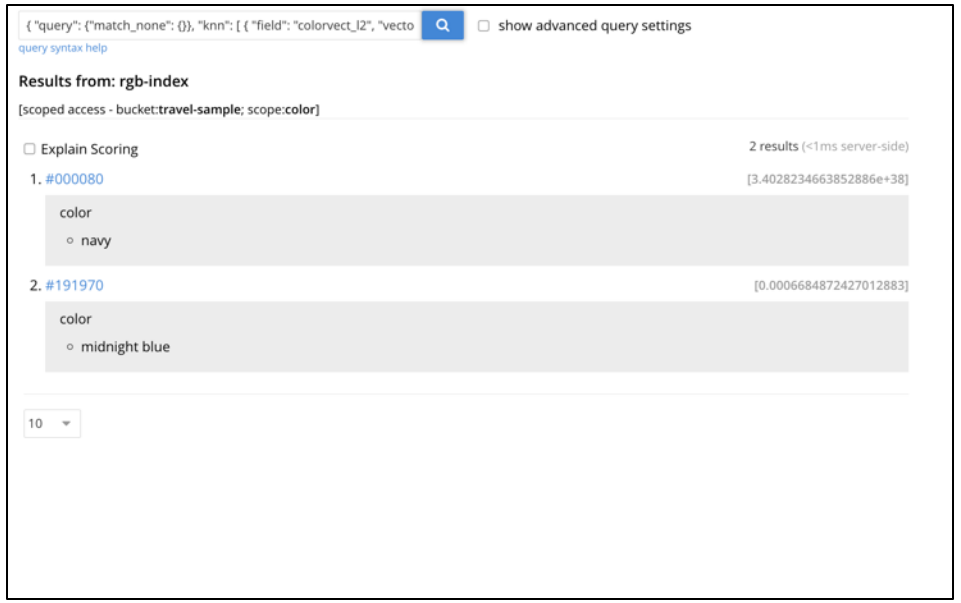
- OR 조건

```
{
"query": {"match_none": {}},
"knn": [
{ "field": "colorvect_l2", "vector": [0, 0, 128], "k": 3 },
{ "field": "colorvect_l2", "vector": [0, 0, 64], "k": 3 }
],
"fields": ["color"]
}
```



- AND 조건

```
{
"query": {"match_none": {}},
"knn": [
{ "field": "colorvect_l2", "vector": [0, 0, 128], "k": 3},
{ "field": "colorvect_l2", "vector": [0, 0, 64], "k": 3}],
"knn_operator": "and",
"fields": ["color"]
}
```

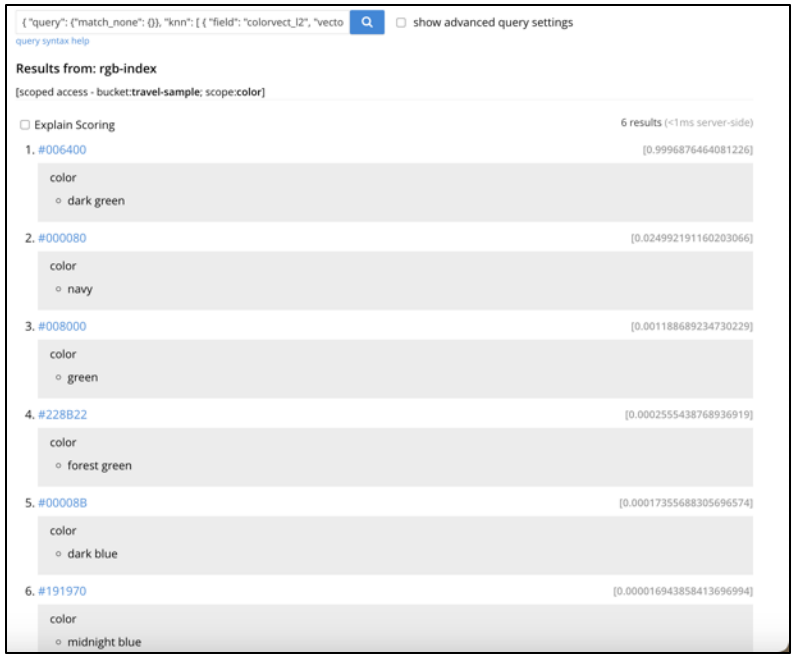


실습 > Vector Search (RGB)

- Boost 조건

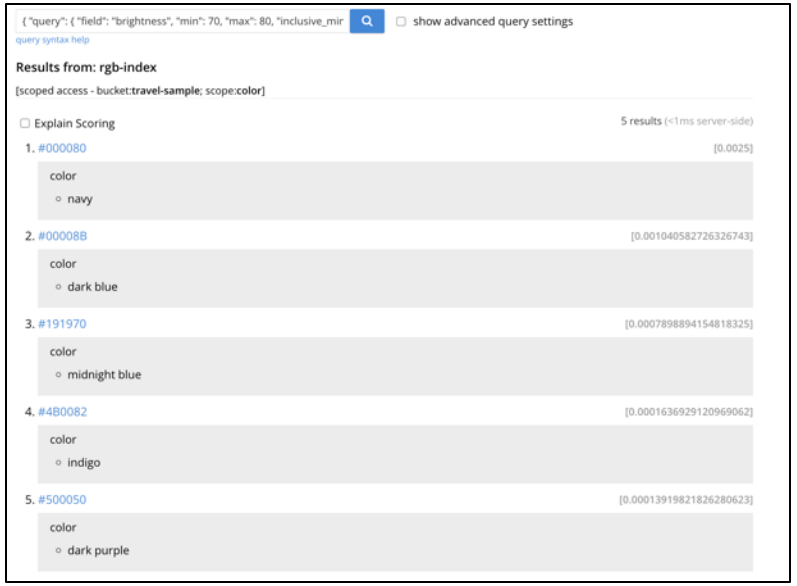
❖ 1보다 작으면, 가중치 감소시키고, 1보다 크면 가중치 증가시킴

```
{
"query": {"match_none": {}},
"knn": [
{ "field": "colorvect_l2", "vector": [0, 0, 127], "k":3, "boost": 0.1},
{ "field": "colorvect_l2", "vector": [0, 99, 0], "k":3, "boost": 4.0}
],
"fields": ["color"]
}
```



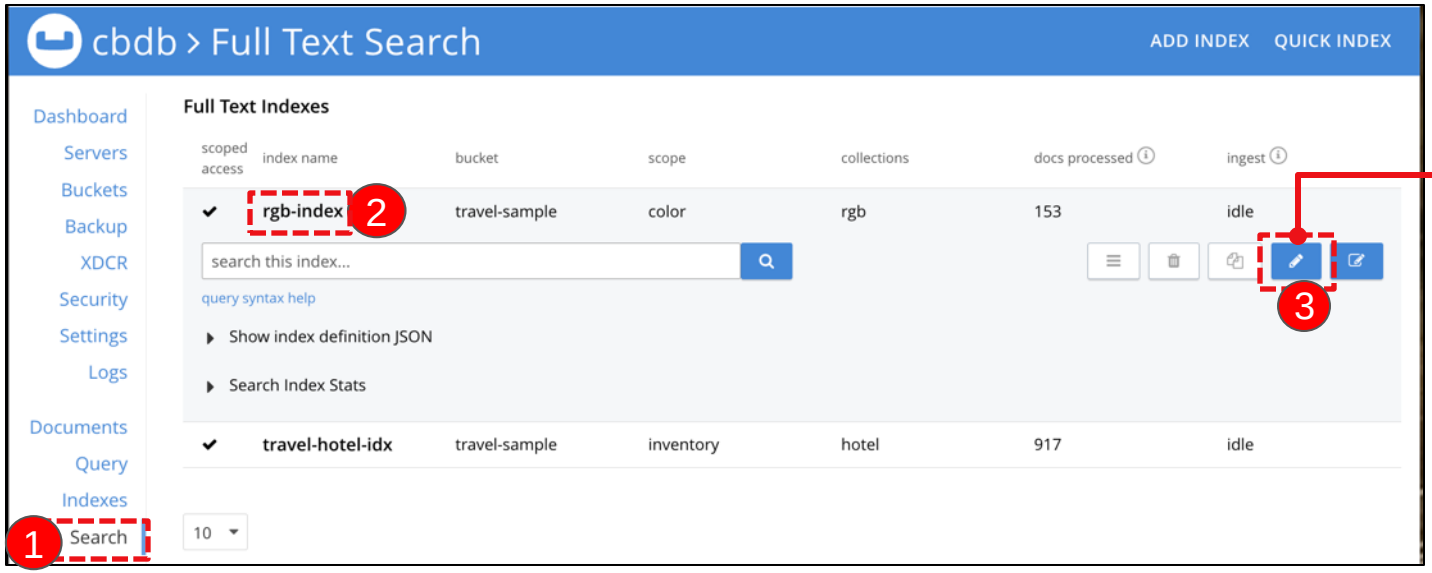
- Hybrid Search 조건

```
{
"query": {
    "field": "brightness", "min": 70, "max": 80,
    "inclusive_min": false, "inclusive_max": true },
"knn": [ {"field": "colorvect_l2", "vector": [0.0, 0.0, 108.0], "k": 5} ],
"fields": ["color","brightness"],
"size": 5
}
```

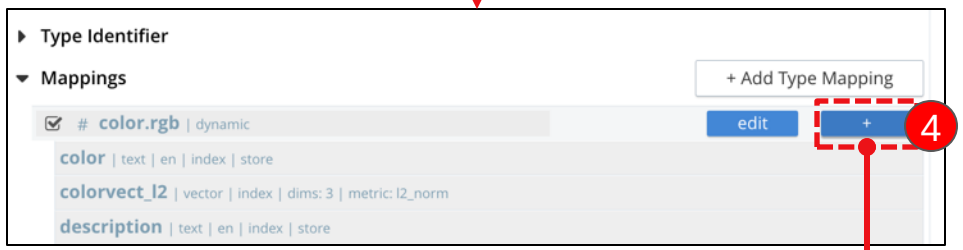


실습 > 인덱스 변경

1. 인덱스 설정 파일 Import로 인덱스 생성
   1) Search > 2) rgb-index > 3) 수정(연필) 클릭 > 마우스를 color.rgb로 이동하면 edit 와 + 가 나타남, 4) +를 선택, 5) insert child field 선택

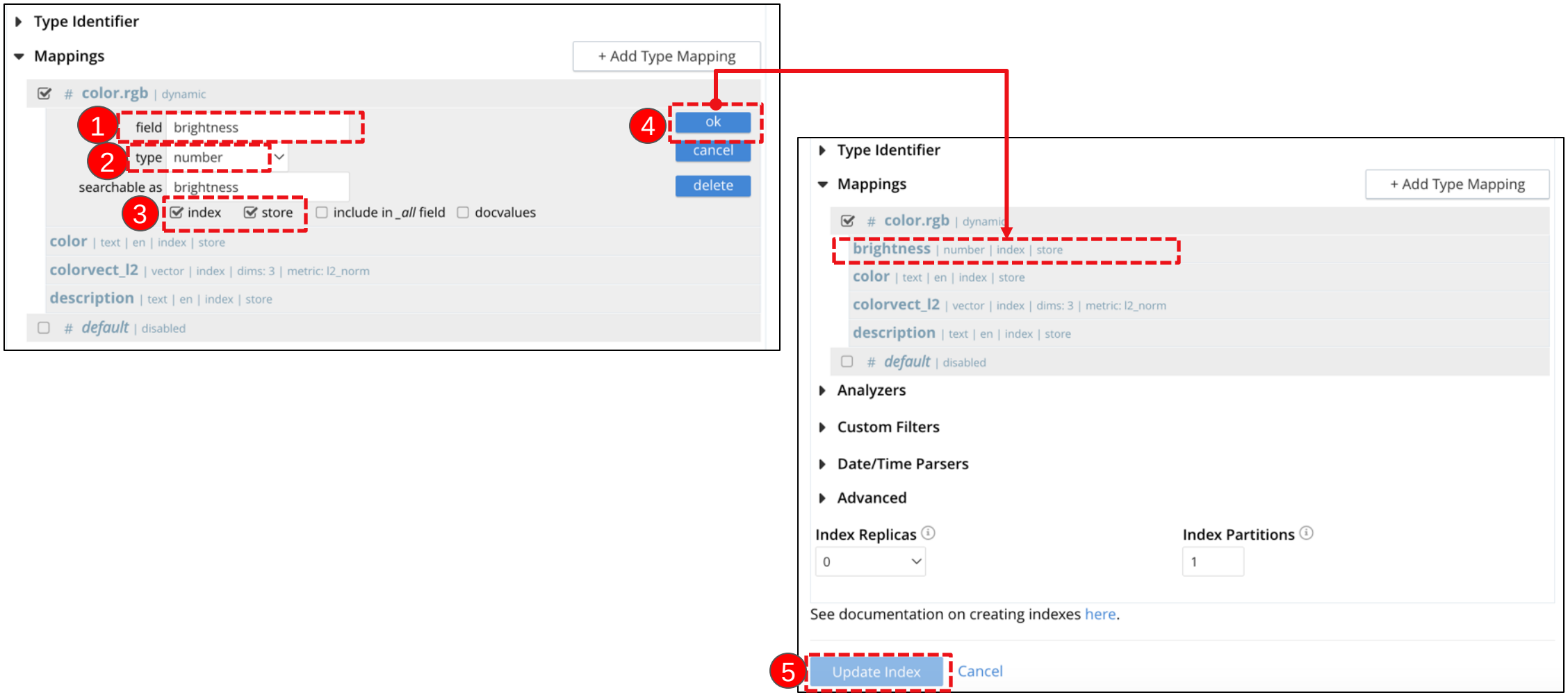


인덱스수정



실습 > 인덱스 변경

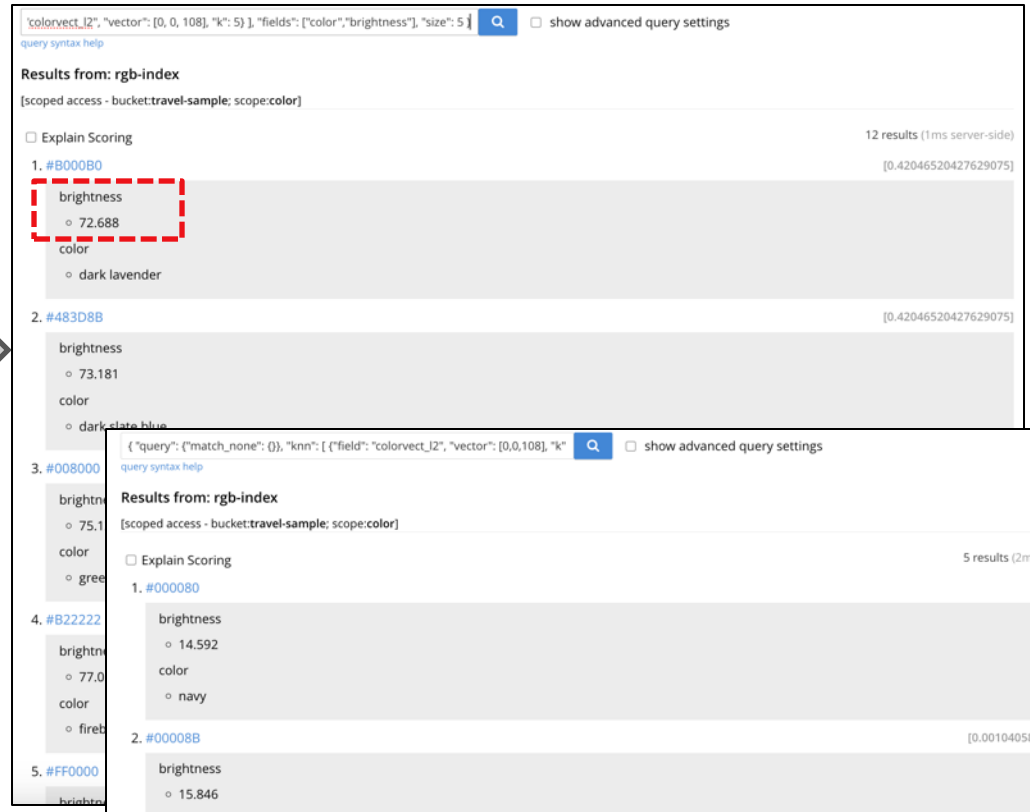
2. 새로운 필드에 대한 인덱스 추가
   1) field 에 brightness 입력 > 2) type을 *number* 선택 *3) index, store* 선택, 4) ok 를 선택, 5) Update index 를 클릭



실습 > Vector Search (RGB)

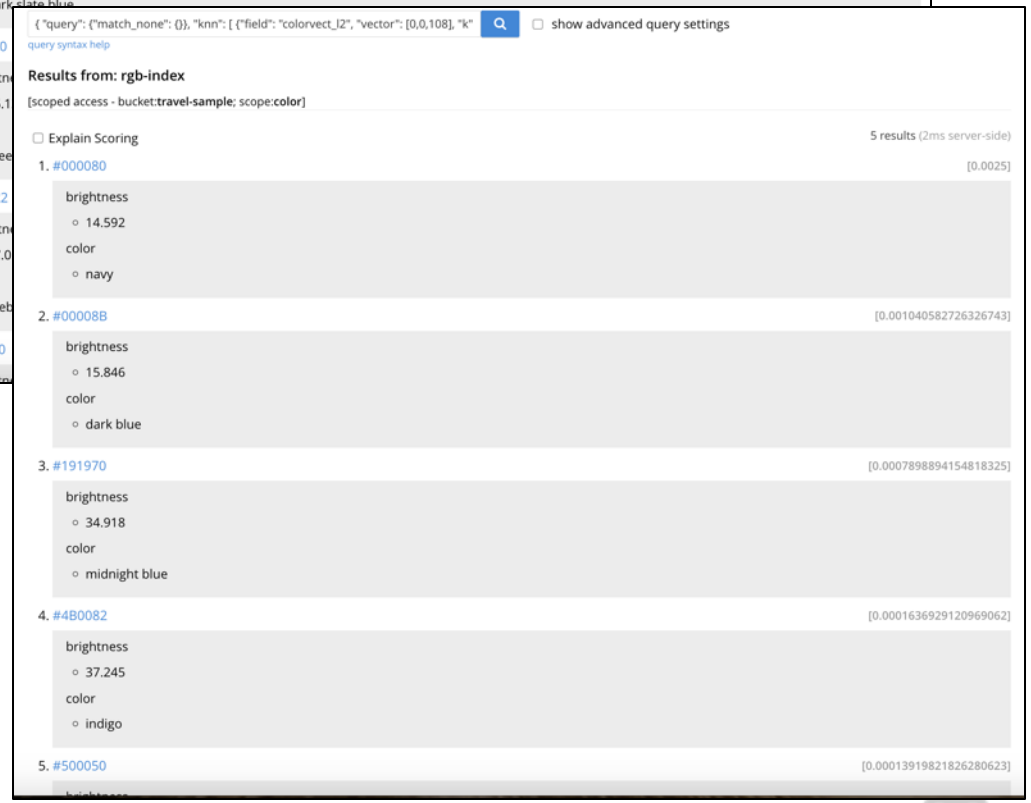
- Hybrid Search, 결과 출력 변경

```
{
"query": {
   "field": "brightness", "min": 70, "max": 80,
   "inclusive_min": false, "inclusive_max": true },
"knn": [ {"field": "colorvect_l2", "vector": [0, 0, 108], "k": 5} ],
"fields": ["color","brightness"],
"size": 5
}
```



- Vector Search, 결과 출력 변경

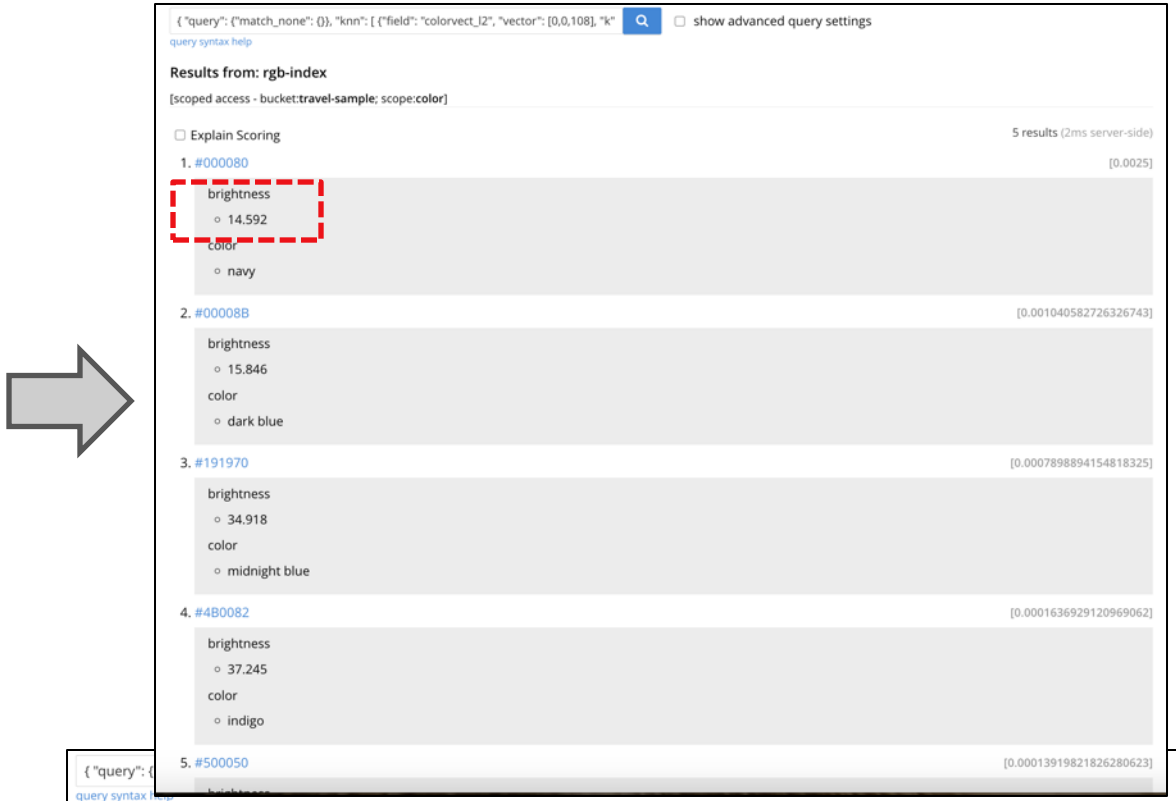
```
{
"query": {"match_none": {}},
"knn": [ {"field": "colorvect_l2", "vector": [0,0,108], "k": 5} ],
"fields": ["color","brightness"]
}
```



실습 > Vector Search (RGB)

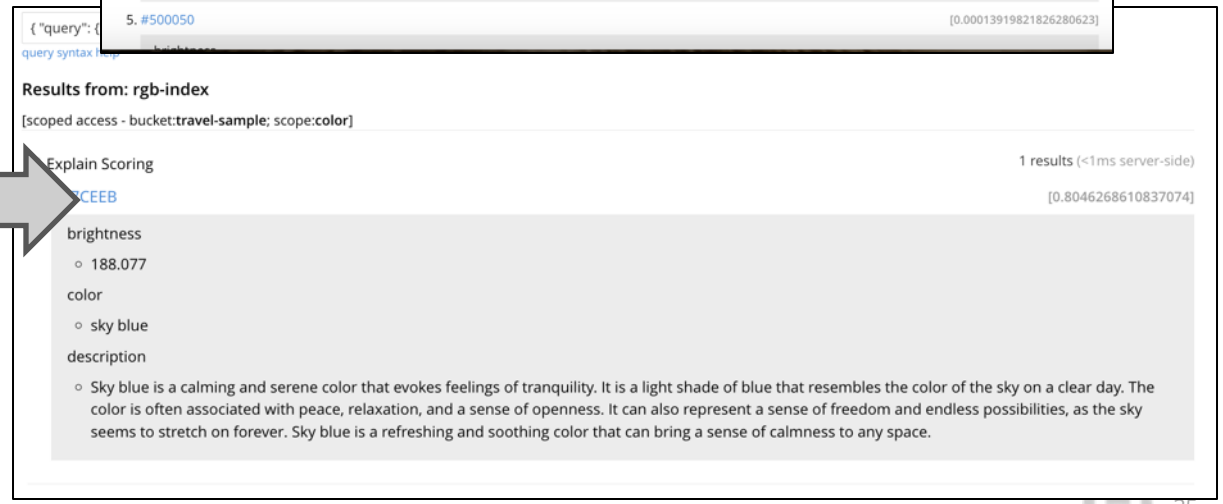
- Hybrid Search, 결과 출력 변경

```
{
"query": {
"field": "brightness", "min": 10, "max": 20,
"inclusive_min": false, "inclusive_max": true },
"knn": [ {"field": "colorvect_l2", "vector": [0.0, 0.0, 108.0], "k": 5} ],
"fields": ["color","brightness"],
"size": 5
}
```



- Hybrid Search, 결과 출력 변경

```
{
"query": {"match": "freedom","field": "description"} ,
"fields": ["color","brightness","description"],
"size": 4
}
```



실습 > SQL++ Hybrid Query (RGB)

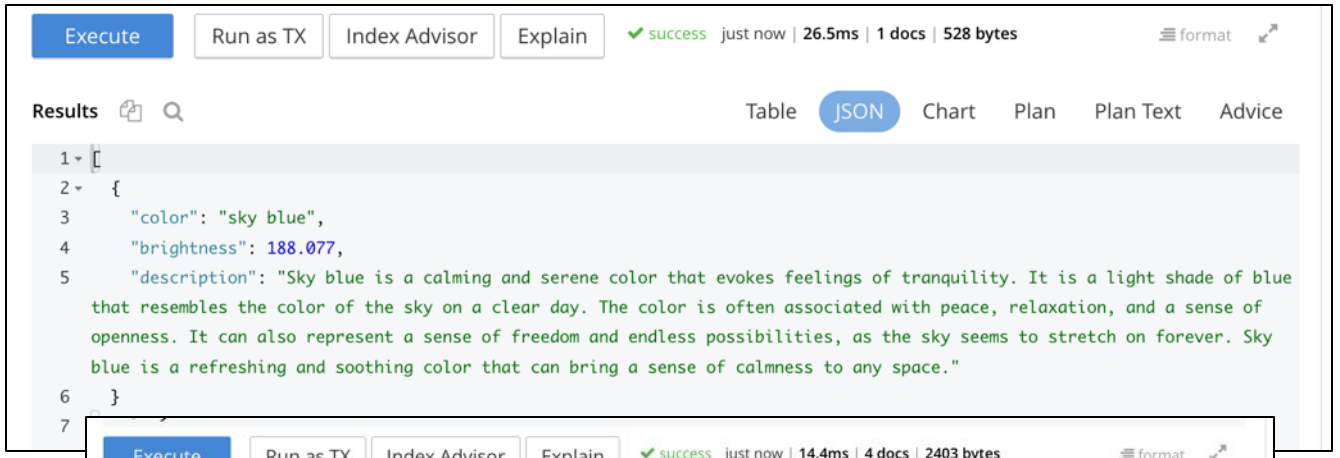
- Hybrid Search, 결과 출력 변경

```
SELECT color, brightness
FROM `travel-sample`.color.rgb AS t1
WHERE brightness <= 20 AND brightness>=10
AND
SEARCH(t1, {
  "query": { "match_none": {} },
  "knn": [{ "field": "colorvect_l2", "vector": [0.0, 0.0, 108.0],"k": 3 }]
 }
)
```

실습 > SQL++ Hybrid Query (RGB)

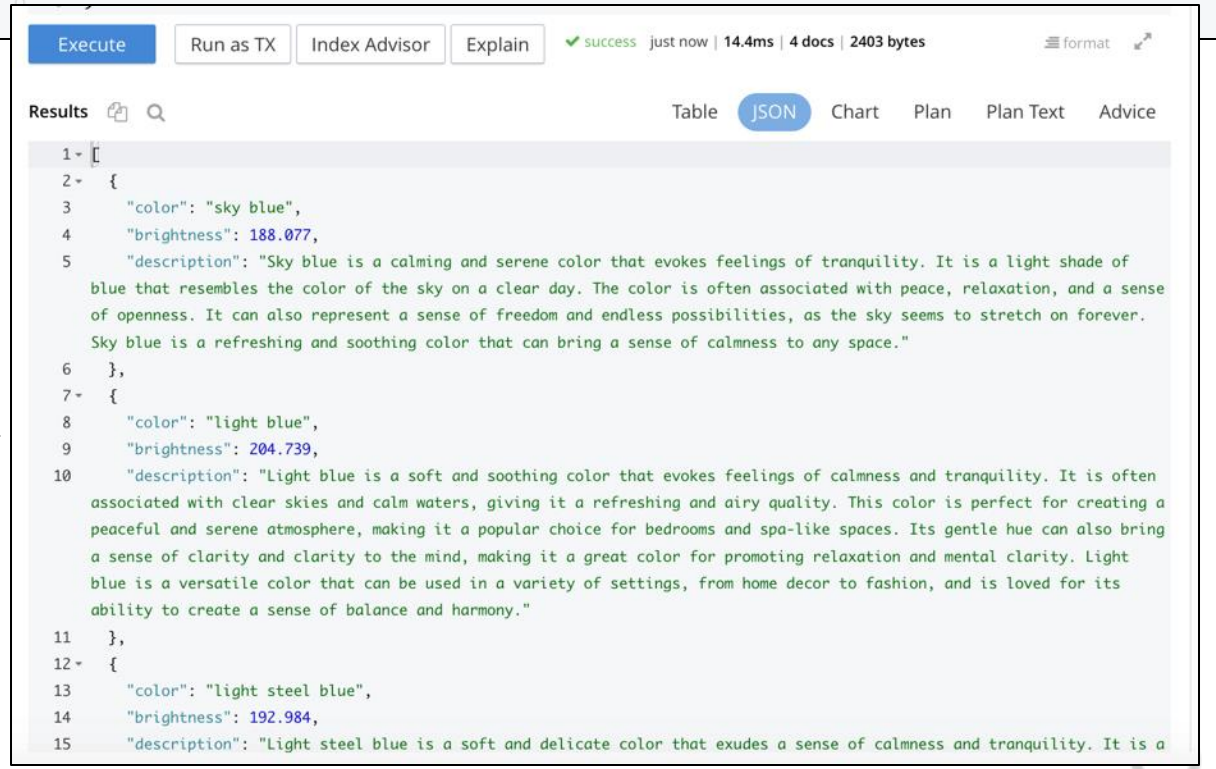
- Hybrid Search, 결과 출력 변경

```
SELECT color, brightness, description
FROM `travel-sample`.color.rgb AS t1
WHERE
SEARCH(t1, {
"query": {"match": "freedom","field": "description"} ,
"fields": ["color","description"],"size": 4}
)
```



- Hybrid Search, 결과 출력 변경

```
SELECT color, brightness, description
FROM `travel-sample`.color.rgb AS t1
WHERE
SEARCH(t1, {
"query": {"match": "freedom","field": "description"} ,
"knn": [ { "field": "colorvect_l2", "vector": [135,204,232], "k": 4} ],
"fields": ["color","description"],"size": 4}
)
```



실습 > SQL++ Hybrid Query (RGB)

- Hybrid Search, 결과 출력 변경

```
SELECT color, brightness, description
FROM `travel-sample`.color.rgb AS t1
WHERE brightness >= 180 AND brightness <= 190
AND
SEARCH(t1, {
  "query": {"match": "freedom","field": "description"} ,
  "knn": [ { "field": "colorvect_l2", "vector": [135,204,232], "k": 4} ],
  "fields": ["color","description"],"size": 4}
)
```

수고하셨습니다.

paul.son@couchbase.com

www.couchbase.com

cloud.couchbase.com

Couchbase