Data And Beyond

Featured

Vector Databases: A Beginner’s Guide!

Pavan Belagatti Follow 9 min read · Aug 25, 2023

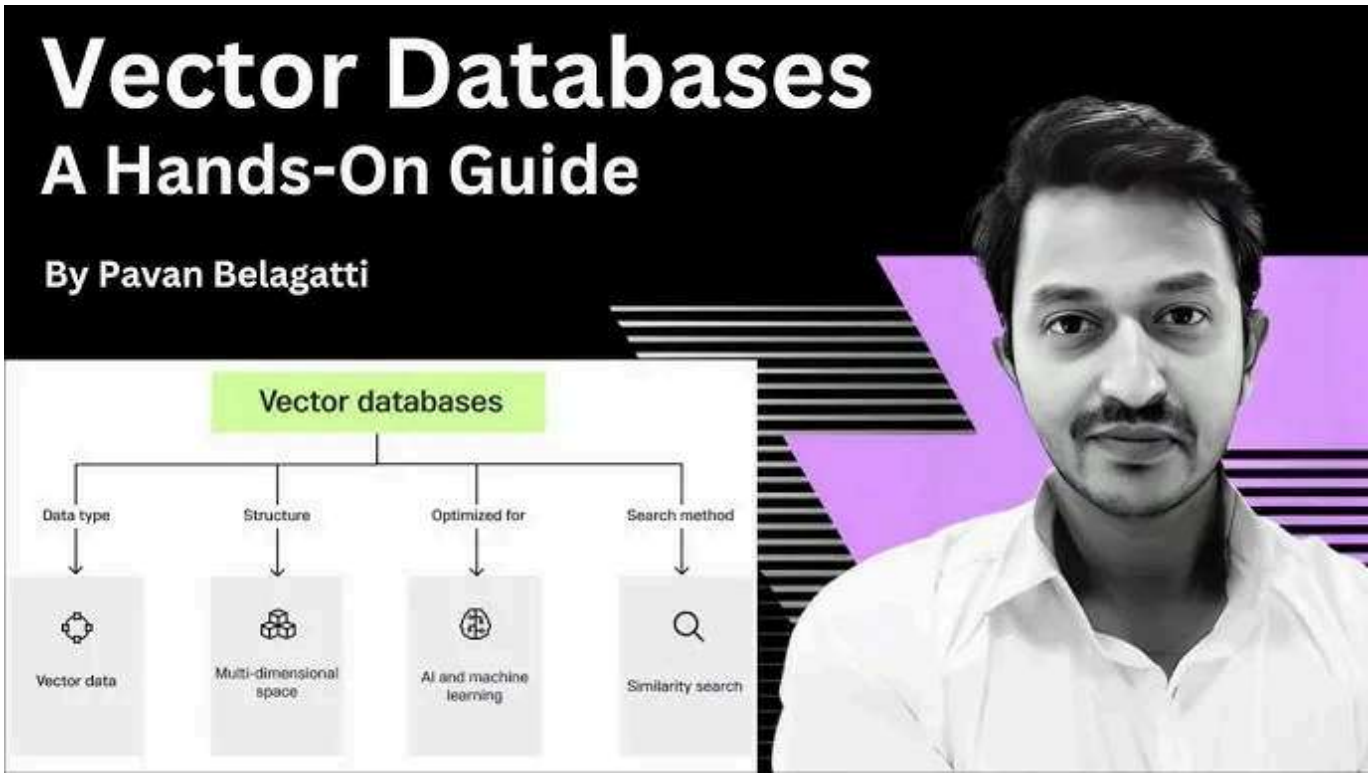


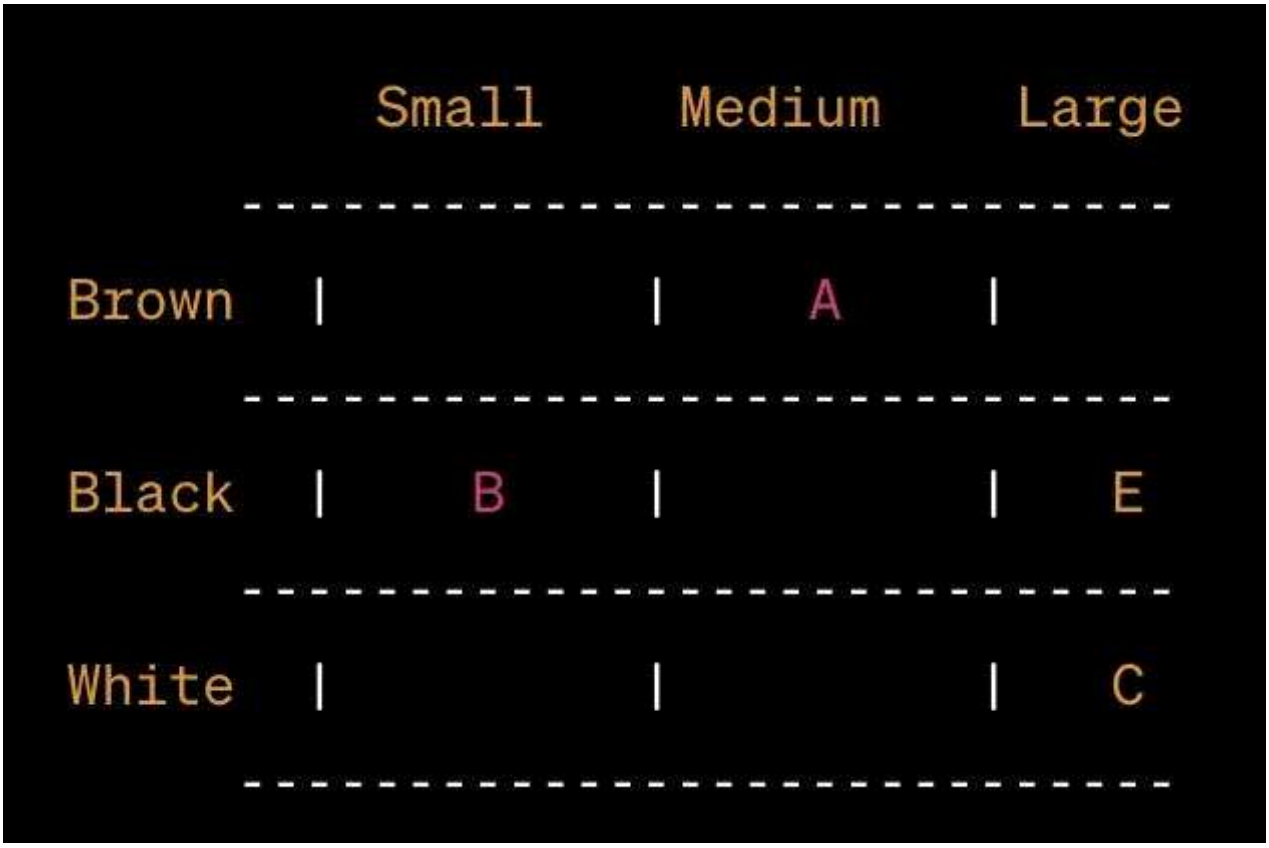
Image by author: Pavan Belagatti

efficiently handling and extracting meaning from intricate datasets. Enter vector databases, a technological innovation that has emerged as a solution to the challenges posed by the ever-expanding landscape of data.

Understanding Vector Databases

Vector databases have gained significant importance in various fields due to their unique ability to efficiently store, index, and search high-dimensional data points, often referred to as vectors. These databases are designed to handle data where each entry is represented as a vector in a multi-dimensional space. The vectors can represent a wide range of information, such as numerical features, embeddings from text or images, and even complex data like molecular structures.

Let's represent the vector database using a 2D grid where one axis represents the color of the animal (brown, black, white) and the other axis represents the size (small, medium, large).



In this representation:

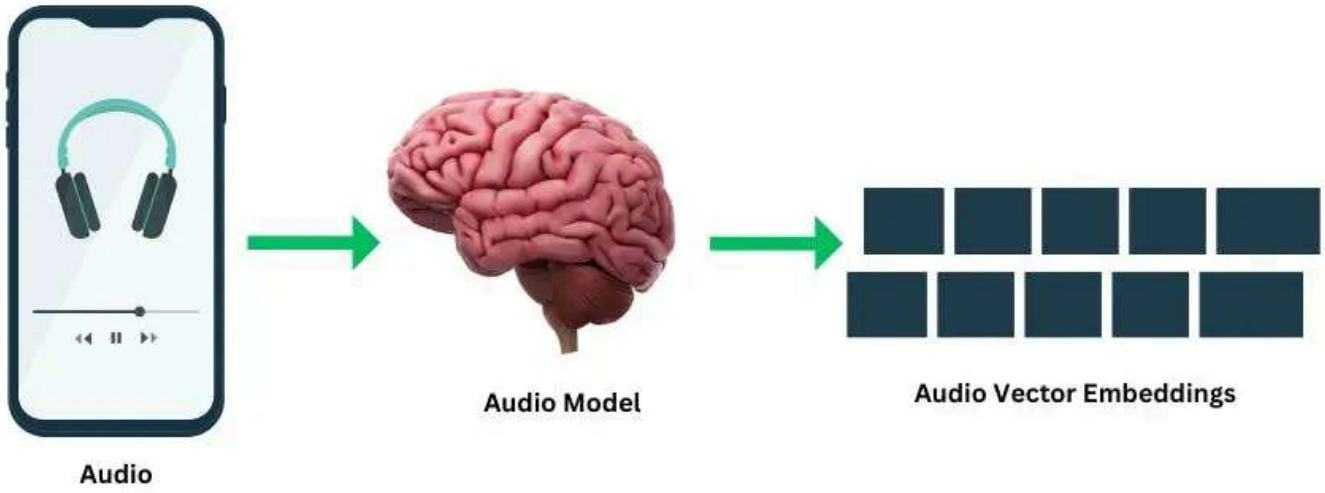
- Image A: Brown color, Medium size
- Image B: Black color, Small size
- Image C: White color, Large size
- Image E: Black color, Large size

You can imagine each image as a point plotted on this grid based on its color and size attributes. This simplified grid captures the essence of how a vector database could be represented visually, even though the actual vector spaces might have many more dimensions and use sophisticated techniques for search and retrieval.

Explain Vector Databases Like I'm 5

Imagine you have a big box of colorful crayons, and each crayon is a different color. A vector database is like a magical sorting machine that helps you find crayons that are similar in color really fast. When you want a crayon that looks like your favorite blue one, you put in a picture of it, and the machine quickly looks through all the crayons. It finds the ones that are closest in color to your blue crayon and shows them to you. This way, you can easily pick out the crayons you want without searching through the whole box!

How Do Vector Databases Store Data?



Vector databases store data as high-dimensional vector embeddings, capturing semantic meaning and relationships. They utilize specialized indexing techniques like hashing, quantization, and graph-based methods to enable fast querying and similarity searches.

Vector databases excel at retrieving semantically similar data points, making them ideal for managing unstructured data like text, images, and audio. While computationally intensive, vector databases are designed to scale efficiently, accommodating the growing demands of AI applications. Despite integration challenges, their ability to manage complex data relationships positions them as a critical component in modern data management strategies.

How Do Vector Databases Work?

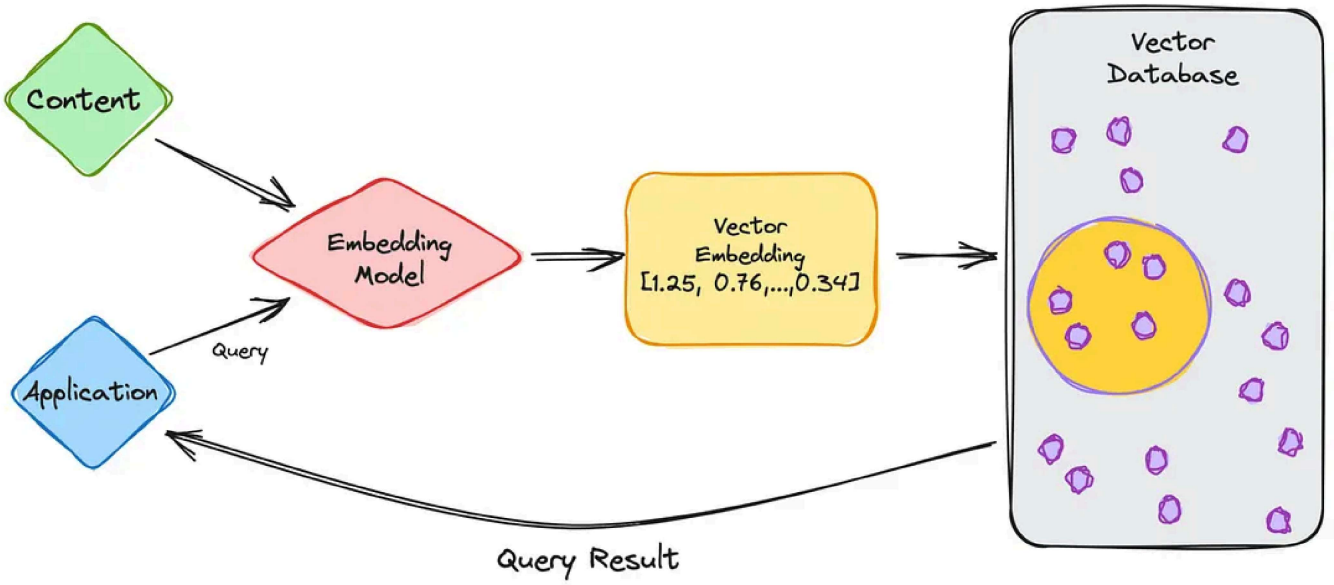


Image credits: KDnuggets

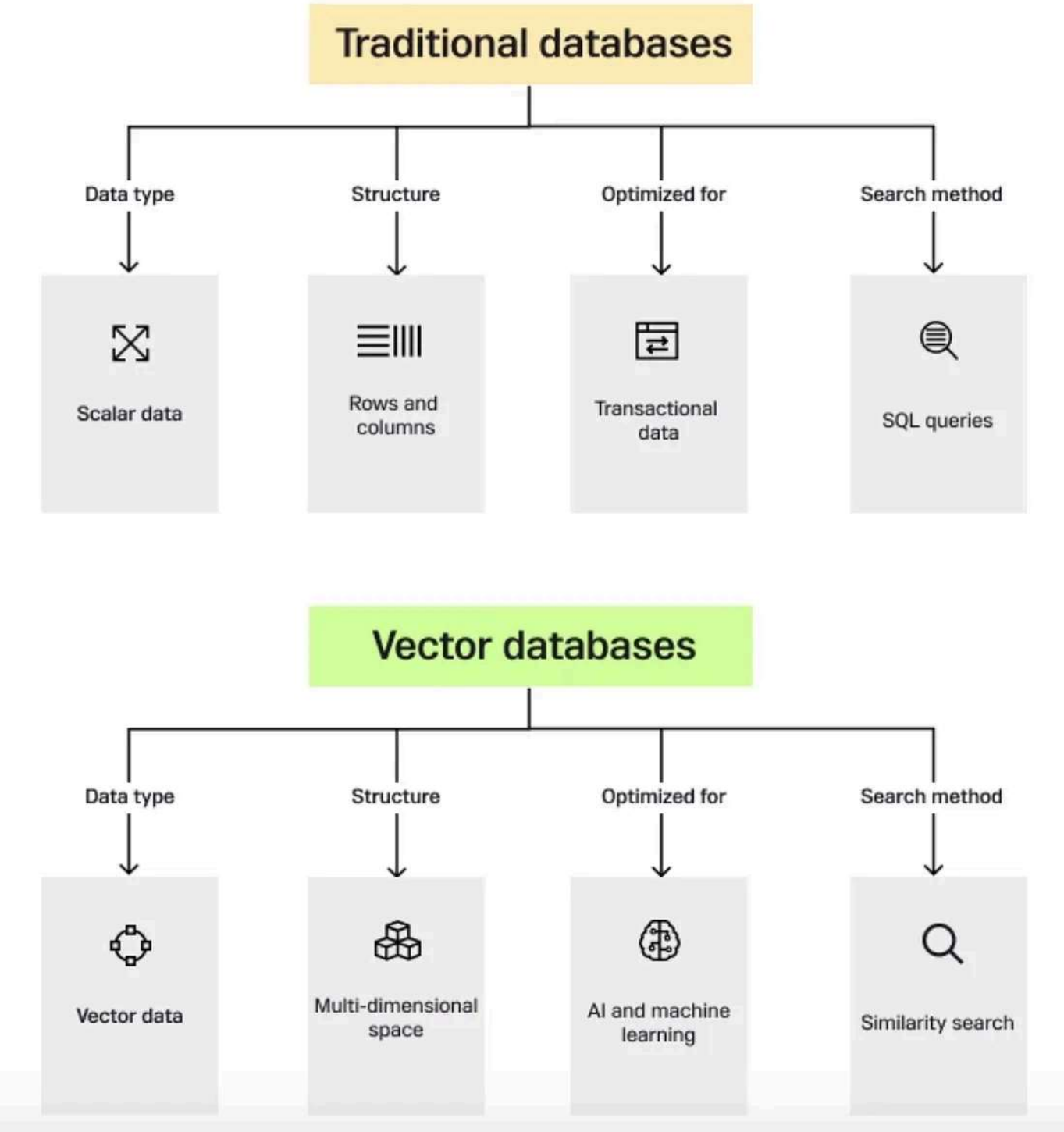
When a user query is initiated, various types of raw data including images, documents, videos and audio. All of this, which can be either unstructured or structured, are first processed through an embedding model. This model is often a complex neural network, translating data into high-dimensional numerical vectors and effectively encoding the data's characteristics into vector embeddings — which are then stored into a a vector database like SingleStoreDB.

When retrieval is required, the vector database performs operations (like similarity searches) to find and retrieve the vectors most similar to the query, efficiently handling complex queries and delivering relevant results to the user. This entire process enables the rapid and accurate management of vast and varied data types in applications that require high-speed search and retrieval functions.

Here is my in-depth hands-on video on vector databases.

How does a vector database differ from a traditional database?

Let’s explore the difference between a vector database and a traditional database.



Vector databases represent a significant departure from traditional databases in their approach to data organization and retrieval. Traditional databases are structured to handle discrete, scalar data types like numbers and strings, organizing them in rows and columns.

This structure is ideal for transactional data but less efficient for the complex, high-dimensional data typically used in AI and machine learning. In contrast, vector databases are designed to store and manage vector data — arrays of numbers that represent points in a multi-dimensional space.

This makes them inherently suited for tasks involving similarity search where the goal is to find the closest data points in a high-dimensional space, a common requirement in AI applications like image and voice recognition, recommendation systems and natural language processing. By leveraging indexing and search algorithms optimized for high-dimensional vector spaces, vector databases offer a more efficient and effective way to handle the kind of data that is increasingly prevalent in the age of advanced AI and machine learning.

Vector Database Capabilities

The significance of vector databases lies in their capabilities and applications:

- Efficient Similarity Search:

Vector databases excel at performing similarity searches, where you can retrieve vectors that are most similar to a given query vector.

- High-Dimensional Data:

Vector databases are designed to handle high-dimensional data more efficiently, making them suitable for applications like natural language processing, computer vision, and genomics.

- Machine Learning and AI:

Vector databases are often used to store embeddings generated by machine learning models. These embeddings capture the essential features of the

data and can be used for various tasks, such as clustering, classification, and anomaly detection.

- Real-time Applications:

Many vector databases are optimized for real-time or near-real-time querying, making them suitable for applications that require quick responses, such as recommendation systems in e-commerce, fraud detection, and monitoring IoT sensor data.

- Personalization and User Profiling:

Vector databases enable personalized experiences by allowing systems to understand and predict user preferences. This is crucial in platforms like streaming services, social media, and online marketplaces.

- Spatial and Geographic Data:

Vector databases can handle geographic data, such as points, lines, and polygons, efficiently. This is essential in applications like geographical information systems (GIS), location-based services, and navigation applications.

- Healthcare and Life Sciences:

In genomics and molecular biology, vector databases are used to store and analyze genetic sequences, protein structures, and other molecular data.

- Data Fusion and Integration:

Vector databases can integrate data from various sources and types, enabling more comprehensive analysis and insights. This is valuable in scenarios where data comes from multiple modalities, such as combining text, image, and numerical data.

- Multilingual Search:

Vector databases can be used to create powerful multilingual search engines by representing text documents as vectors in a common space, enabling cross-lingual similarity searches.

Vector Database Use Cases

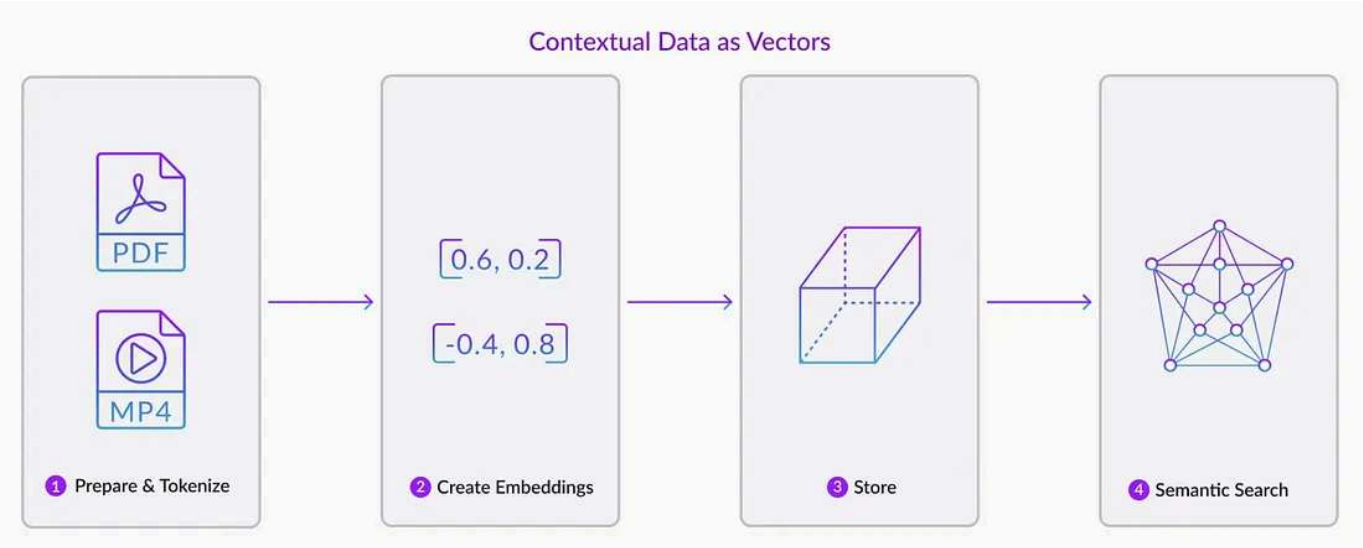
Vector databases play a vital role in recommendation systems for businesses. For example, they can recommend items to a user depending on their browsing or buying behavior. They shine well even in fraud detection systems where they can detect anomalous patterns by comparing transaction embeddings against known profiles of fraudulent activity, thus enabling real-time fraud detection. Face recognition is an additional use case where vector databases store facial feature embeddings and help in security and surveillance.

They can even help organizations with customer support by responding to the similar queries with pre-determined or little varied responses. Market research is another area where vector databases do well by analyzing customer feedback and social media posts, converting them into text embeddings to do sentiment analysis and trend spotting — gaining even more business insights.

Vector Database Tutorial

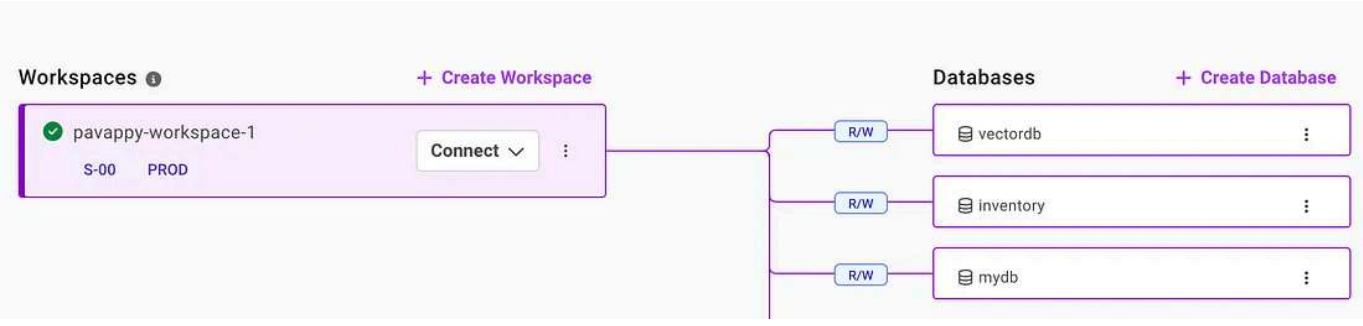
Harness the robust vector database capabilities of SingleStoreDB, tailored to seamlessly serve AI-driven applications, chatbots, image recognition

systems, and more. SingleStore has supported vector capabilities since 2017, enabling efficient storage and retrieval of high-dimensional vector data. This functionality allows for advanced applications such as semantic search, recommendation systems, and real-time analytics, making it a versatile choice for modern data-driven solutions.

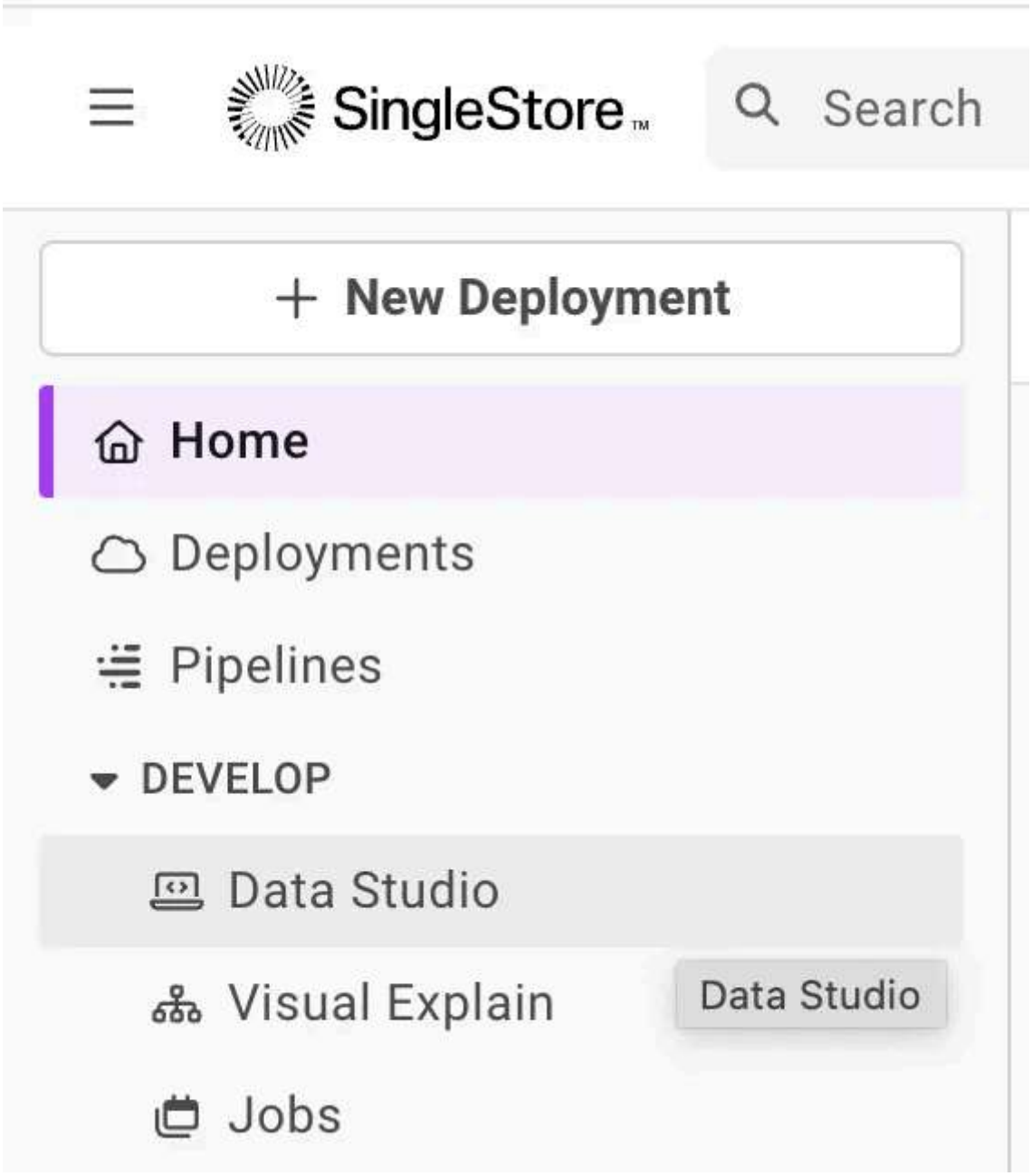


Sign up to SingleStore to start using it as a vector database. You when you sign up, you will receive free credits.

Once you sign up to SingleStore, and sign in, this is where you will land. You will have the Workspace creayed by default (if you don't have a workspace, create one). Under your workspace, create a database just by clicking the '+ Create Database' tab as shown below, it's free.

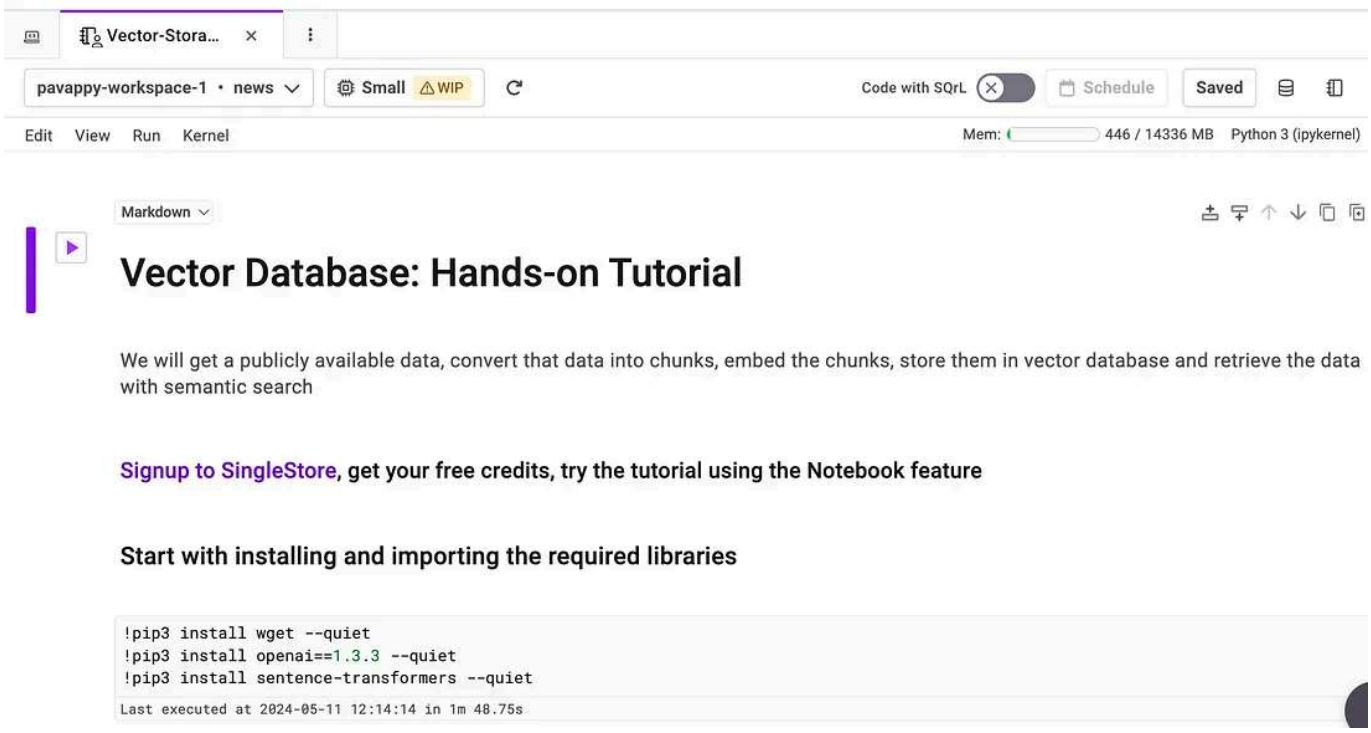


Use SingleStore's Notebooks feature (just like Jupyter Notebooks or Google Colab). That is where we are going to add our code to experience the robust vector database capabilities.



Create a new Notebook and start adding the code.

Make sure to select your respective workspace and database you created.



Start with installing and importing the required libraries and dependencies.

```
!pip3 install wget — quiet
!pip3 install openai==1.3.3 — quiet
!pip3 install sentence-transformers — quiet
```

```
import json
import os
import pandas as pd
import wget
```

Download the model

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('flax-sentence-embeddings/all_datasets_v3_mpnet-base
```

Import data from the csv file (AG News is a subdataset of AG's corpus of news articles)

```
cvs_file_path = 'https://raw.githubusercontent.com/openai/openai-cookbook/main/e
file_path = 'AG_news_samples.csv'

if not os.path.exists(file_path):
    wget.download(cvs_file_path, file_path)
    print('File downloaded successfully.')
else:
    print('File already exists in the local file system.')

df = pd.read_csv('AG_news_samples.csv')
df
```

You can see the data here

```
data = df.to_dict(orient='records')
data[0]
```

The next step is set up the database to store our data

```
%%sql
```

```
DROP TABLE IF EXISTS news_articles;
CREATE TABLE IF NOT EXISTS news_articles (
    title TEXT,
    description TEXT,
    genre TEXT,
    embedding BLOB,
    FULLTEXT(title, description)
);
```

Get embeddings for every row based on the description column

```
descriptions = [row['description'] for row in data]
all_embeddings = model.encode(descriptions)
all_embeddings.shape
```

Merge embedding values into data rows

```
for row, embedding in zip(data, all_embeddings):
    row['embedding'] = embedding
```

Here is an example of one row of the combined data

```
data[0]
```

You should see the response as below,

```
Out[37]: {'title': 'World Briefings',
          'description': 'BRITAIN: BLAIR WARNS OF CLIMATE THREAT Prime Minister Tony Blair urged the international communi
         ty to consider global warming a dire threat and agree on a plan of action to curb the  quot;alarming quot; growth
         of greenhouse gases.',
          'label_int': 1,
          'label': 'World',
          'embedding': array([-1.42552713e-02, -1.03357071e-02,  1.25946105e-02,  8.40715785e-03,
                 -6.92264410e-03, -8.77237227e-03, -5.38323671e-02,  1.95311196e-02,
                  9.50564742e-02,  1.60899572e-02,  4.72200625e-02,  2.30231155e-02,
                 -6.69442937e-02,  2.82599987e-03,  2.79738400e-02, -6.46088347e-02,
                  5.52451760e-02, -4.02353071e-02, -2.22880822e-02, -1.65119395e-02,
                  3.61824557e-02,  3.32110142e-03,  1.18329516e-02,  7.70277716e-03,
                 -4.18954827e-02, -2.76368838e-02,  3.64982933e-02,  3.69321145e-02,
                  5.97776957e-02,  8.05662386e-03,  3.38091105e-02, -1.52911590e-02,
                  1.38111366e-02, -4.00905032e-03,  3.15332080e-08,  2.20769504e-03,
                  3.78836691e-03, -1.83615256e-02,  1.49200745e-02, -1.62021983e-02,
                 -5.16570453e-03, -8.89025070e-03,  6.39182806e-04,  2.90938653e-02,
                 -6.70327619e-02,  8.33853893e-03, -7.37016380e-04,  3.79642798e-03,
                  2.53367070e-02,  4.11500176e-03, -7.02746958e-03,  8.54118988e-02,
                  1.05591035e-02, -8.18551611e-03, -8.49048868e-02, -2.57210108e-03,
                  3.90590318e-02, -6.48996532e-02, -3.32521796e-02,  2.81049777e-02,
                 -6.10832423e-02,  6.69943467e-02, -4.27179411e-02,  1.76054183e-02,
                 -2.59039514e-02,  3.61620728e-03, -3.28656584e-02,  3.43255475e-02,
```

Now, let's populate the database with our data

```
%sql TRUNCATE TABLE news_articles;

import sqlalchemy as sa
from singlestoredb import create_engine

# Use create_table from singlestoredb since it uses the notebook connection URL
conn = create_engine().connect()

statement = sa.text('''
    INSERT INTO news_articles (
        title,
        description,
        genre,
        embedding
    )
    VALUES (
        :title,
        :description,
        :label,
        :embedding
    )
''')

conn.execute(statement, data)
```

Let’s run semantic search, and get scores for the search term ‘India’

```
search_query = 'India'
search_embedding = model.encode(search_query)

query_statement = sa.text('''
    SELECT
        title,
        description,
        genre,
        DOT_PRODUCT(embedding, :embedding) AS score
    FROM news_articles
    ORDER BY score DESC
    LIMIT 10
''')

# Execute the SQL statement.
results = pd.DataFrame(conn.execute(query_statement, dict(embedding=search_embed
print(results)
```

You should see the results as below,

```
                                                title  \
0          Militants beat man thought to be from US
1         Bomb at India Independence Parade Kills 15
2        Microsoft Unveils Windows XP for India (AP)
3  4 killed, 54 wounded in three separate attacks...
4   Northeast Indian State Votes Amid Tight Security
5  Why The Open-Source Model Can Work In India (T...
6          Microsoft to Hire Hundreds More in India
7  Bhopal victims commemorate 20th anniversary of...
8  Follow-on shy Aussies lead India by 355 runs (...
9                           No channel for series

                                          description     genre     score
0  HENDALA, Sri Lanka -- Day after day, locked in...     World  0.402497
1  NEW DELHI - A bomb exploded during an Independ...     World  0.346617
2  AP - Microsoft Corp. announced Wednesday that ...  Sci/Tech  0.344941
3  Canadian Press - GAUHATI, India (AP) - Residen...     World  0.337517
4   GUWAHATI, India (Reuters) - People braved a s...     World  0.323705
5  TechWeb - An Indian Institute of Technology pr...  Sci/Tech  0.316088
6   HYDERABAD, India (Reuters) - Microsoft Corp. ...  Business  0.274268
7  AFP - A series of torchlight rallies and vigil...     World  0.267720
8  AFP - Hosts India braced themselves for a harr...     World  0.264600
9  INDIA #39;S cricket chiefs began a frenetic se...    Sports  0.260308
```

Now, let's run a hybrid search to find articles about India.

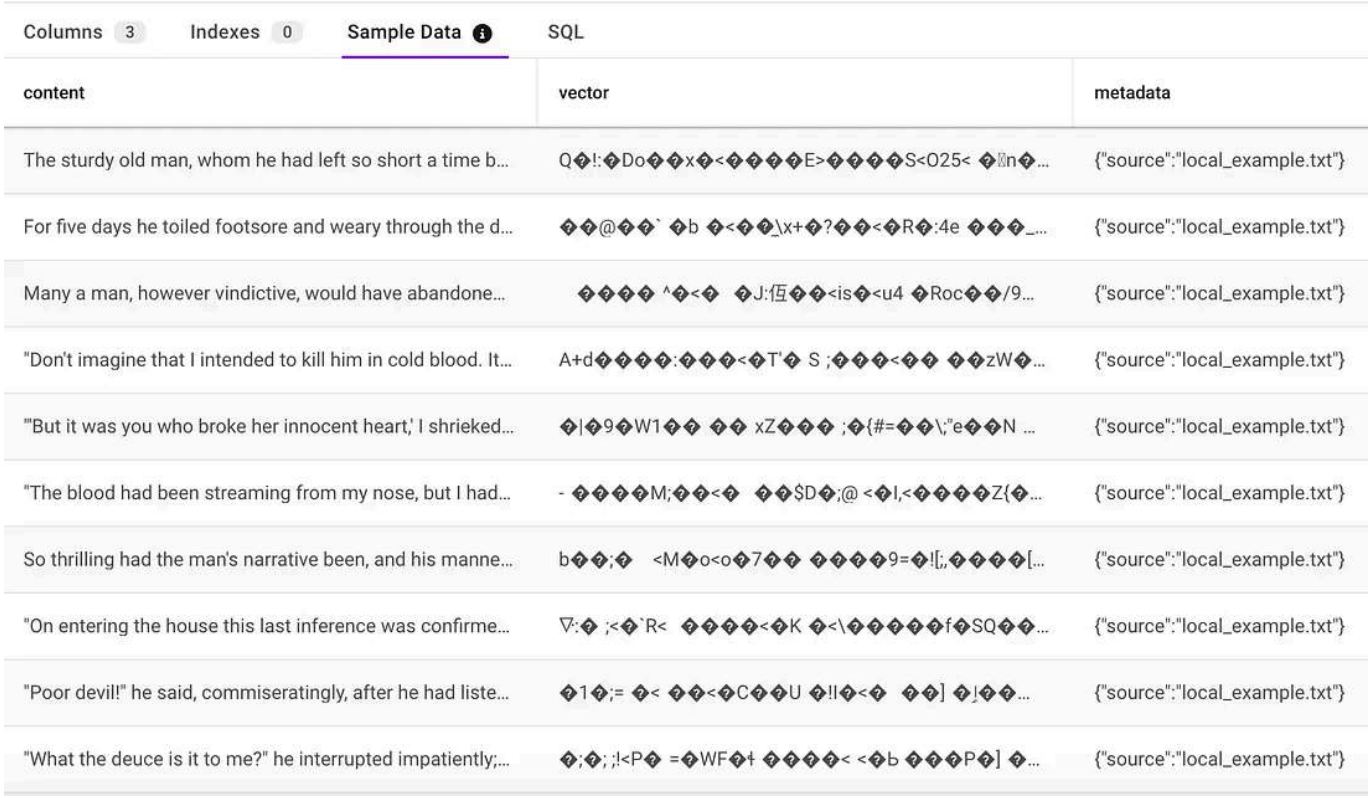
```
hyb_query = 'Articles about India'
hyb_embedding = model.encode(hyb_query)

# Create the SQL statement.
hyb_statement = sa.text('''
    SELECT
        title,
        description,
        genre,
        DOT_PRODUCT(embedding, :embedding) AS semantic_score,
        MATCH(title, description) AGAINST (:query) AS keyword_score,
        (semantic_score + keyword_score) / 2 AS combined_score
    FROM news_articles
    ORDER BY combined_score DESC
    LIMIT 10
''')

# Execute the SQL statement.
```

```
hyb_results = pd.DataFrame(conn.execute(hyb_statement, dict(embedding=hyb_embedd
hyb_results
```

You should see the results as below,

	title	description	genre	semantic_score	keyword_score	combined_score
0	Why The Open-Source Model Can Work In India (T...	TechWeb - An Indian Institute of Technology pr...	Sci/Tech	0.326729	0.380632	0.353681
1	Microsoft to Hire Hundreds More in India	HYDERABAD, India (Reuters) - Microsoft Corp. ...	Business	0.229690	0.410682	0.320186
2	Home series defeats for India	Australia, by winning the third Test at Nagpur...	Sports	0.165084	0.458400	0.311742
3	Microsoft Unveils Windows XP for India (AP)	AP - Microsoft Corp. announced Wednesday that ...	Sci/Tech	0.228409	0.382128	0.305269
4	Putin favors veto right for India as permanent...	In an apparent damage control exercise, Russia...	World	0.242365	0.353575	0.297970
5	Cricket: Aussies dominate India	Australia tighten their grip on the third Test...	World	0.066914	0.511894	0.289404
6	Southern Africa countries pledge enhanced trad...	PORT LOUIS, Aug. 17 (Xinhuanet) -- Southern A...	World	0.181786	0.353575	0.267681
7	Follow-on shy Aussies lead India by 355 runs (...	AFP - Hosts India braced themselves for a harr...	World	0.150445	0.360399	0.255422
8	54 dead, million flee homes as rains lash nort...	AFP - At least 54 people have died and more th...	World	0.159268	0.316645	0.237957
9	Munabao-Khokhropar: Pakistan, India agree to r...	ISLAMABAD: Pakistan and India agreed on Friday...	World	0.154328	0.316645	0.235487

You can go to your database that you created and check how the vector data has been stored.



Columns 3 Indexes 0 Sample Data SQL

content	vector	metadata
The sturdy old man, whom he had left so short a time b...	Q...	{"source":"local_example.txt"}
For five days he toiled footsore and weary through the d...	@ ` b <\x+?<R:4e _...	{"source":"local_example.txt"}
Many a man, however vindictive, would have abandone...	^ < J:<is<u4 Roc/9...	{"source":"local_example.txt"}
"Don't imagine that I intended to kill him in cold blood. It...	A+d:<T' S ;<< zW...	{"source":"local_example.txt"}
"'But it was you who broke her innocent heart,' I shrieked...	|9W1 xZ ;{#=\;"eN ...	{"source":"local_example.txt"}
"The blood had been streaming from my nose, but I had...	- M;< $D;@ <I,<Z{...	{"source":"local_example.txt"}
So thrilling had the man's narrative been, and his manne...	b; <Mo<o7 9=![;,[...	{"source":"local_example.txt"}
"On entering the house this last inference was confirme...	: ;<`R< <K <\f SQ...	{"source":"local_example.txt"}
"Poor devil!" he said, commiseratingly, after he had liste...	1;= < <C U <] ...	{"source":"local_example.txt"}
"What the deuce is it to me?" he interrupted impatiently;...	;; ;I<P =WF< < P] ...	{"source":"local_example.txt"}

The complete code can be found here below,

GitHub - pavanbelagatti/vector-db-tutorial-AV

Contribute to pavanbelagatti/vector-db-tutorial-AV development by creating an account on GitHub.

github.com

Now, it is time for you to play around with SingleStore and build robust AI applications.

Sign up to SingleStore & claim your free credits & get started with building robust AI/ML applications.

The article is originally published on dev.to