

DuckDB Tutorial: Building AI Projects

This tutorial guides you through DuckDB's key features and practical applications, including building tables, performing data analysis, building an RAG application, and using an SQL query engine with LLM.

Jul 7, 2024 · 13 min read



Abid Ali Awan

Certified data scientist, passionate about building ML apps, blogging on data science, and editing.

TOPICS

Artificial Intelligence

Recently, DuckDB came out of beta and released its stable version, gaining popularity rapidly as various data frameworks integrate it into their ecosystems. This makes it a prime time to learn DuckDB so you can keep up with the ever-changing world of data and AI.

In this tutorial, we will learn about DuckDB and its key features with code examples. Our primary focus will be on how we can integrate it with current AI frameworks. For that, we will work on two projects. First, we'll build a Retrieval-Augmented Generation (RAG) application using DuckDB as a vector database. Then, we'll use DuckDB as an AI query engine to analyze data using natural language instead of SQL.

What is DuckDB?

[DuckDB](#) is a modern, high-performance, in-memory analytical database management system (DBMS) designed to support complex analytical queries. It is a relational (table-oriented) DBMS that supports the Structured Query Language (SQL).

DuckDB combines the simplicity and ease of use of SQLite with the high-performance capabilities required for analytical workloads, making it an excellent choice for data scientists and analysts.

Key features

1. **Simple operation:** DuckDB is serverless, has no external dependencies, and is embedded within a host process. This makes it easy to install and deploy, requiring only a C++11 compiler for building.
2. **Feature-rich:** It supports extensive SQL data management features. DuckDB also offers deep integration with Python and R, making it suitable for data science and interactive data analysis.
3. **Fast analytical queries:** DuckDB uses a columnar-vectorized query execution engine optimized for analytics, enabling parallel query processing and efficient handling of large datasets.
4. **Free and open source:** It is released under the permissive MIT License, making it free to use and open-source.
5. **Portability:** With no external dependencies, DuckDB is highly portable and can run on various operating systems (Linux, macOS, Windows) and CPU architectures (x86, ARM). It can even run in web browsers using DuckDB-Wasm.
6. **Extensibility:** DuckDB supports a flexible extension mechanism, allowing the addition of new data types, functions, file formats, and SQL syntax.
7. **Thorough testing:** It undergoes intensive testing using Continuous Integration, with a test suite containing millions of queries. This ensures stability and reliability across different platforms and compilers.

Getting Started with DuckDB

In this section, we will learn to set up DuckDB, load CSV files, perform data analysis, and learn about relations and query functions.

We will start by installing the DuckDB Python package.

```
pip install duckdb --upgrade
```

 Explain code

POWERED BY  datalab

Creating the DuckDB database

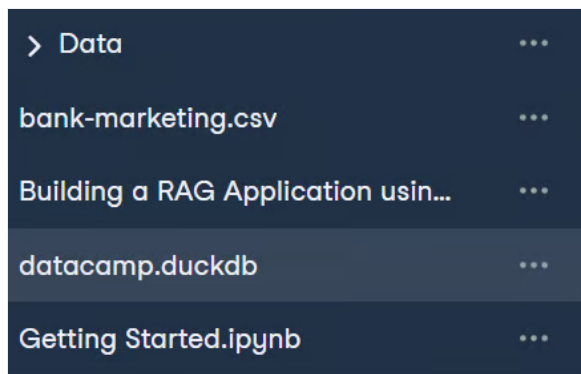
To create the persistent database, you just have to use the `connect` function and provide it with the database name.

```
import duckdb
con = duckdb.connect("datacamp.duckdb")
```

 Explain code

POWERED BY  datalab

It will create a database base file in your local directory.



We will load a CSV file and create a "bank" table. The dataset we are using is available on DataLab and is called [Bank Marketing](#). It consists of direct marketing campaigns by a Portuguese banking institution using phone calls.

To load the CSV file, you have to create a Table first using SQL and then use the `read_csv()` function within the SQL script to load the file. It is that simple.

We will then validate our table by executing the SQL script that shows all of the tables within the database and using the `fetchdf` function to display the result as a pandas DataFrame.

Note: We are using DataCamp's DataLab as a code editor. DataLab is a cloud Jupyter Notebook that you can access for free if you have a DataCamp account.

```
con.execute("""
    CREATE TABLE IF NOT EXISTS bank AS
    SELECT * FROM read_csv('bank-marketing.csv')
""")
con.execute("SHOW ALL TABLES").fetchdf()
```

 Explain code

POWERED BY  datalab

	database ▾	schema ▾	name ▾	column_names ▾	column_types ▾	temporary ▾
0	datacamp	main	bank	["age","job","marital","educati...	["BIGINT","VARCHAR","VARCH...	False

Now that we have successfully created our first table, we will run a beginner-level query to analyze the data and display the result as a DataFrame.

```
con.execute("SELECT * FROM bank WHERE duration < 100 LIMIT 5").fetchdf()
```

 Explain code

POWERED BY  datalab

education	default	housing	loan	contact	month	day_of_week	duration
high.school	no	yes	no	telephone	may	mon	50
unknown	unknown	no	no	telephone	may	mon	55
high.school	no	no	no	telephone	may	mon	38
university.degree	no	no	yes	telephone	may	mon	99
unknown	no	yes	no	telephone	may	mon	93


DuckDB is natively integrated into the new DataLab by DataCamp. Learn more about it by reading the blog "[DuckDB Makes SQL a First-Class Citizen on DataLab](#)" and using the interactive SQL cell to analyze data.

DuckDB Relations

DuckDB relations are essentially tables that can be queried using the Relational API. This API allows for the chaining of various query operations on data sources like Pandas DataFrames. Instead of using SQL queries, you will be chaining together various Python functions to analyze the data.

For example, we will load a CSV file to create the DuckDB relation. To analyze the table, you can chain the filter and limit functions.

```
bank_duck = duckdb.read_csv("bank-marketing.csv", sep=";")
bank_duck.filter("duration < 100").limit(3).df()
```

 Explain code

POWERED BY  datalab

education	default	housing	loan	contact	month	day_of_week	duration
high.school	no	yes	no	telephone	may	mon	50
unknown	unknown	no	no	telephone	may	mon	55
high.school	no	no	no	telephone	may	mon	38

We can also create relations by loading the table from the DuckDB database.

```
rel = con.table("bank")
rel.columns
```

 Explain code

POWERED BY  datalab

```
['age',
 'job',
 'marital',
 'education',
 'default',
 'housing',
 'loan',
 'contact',
 'month',
 'day_of_week',
 'duration',
 'campaign',
 'pdays',
 'previous',
 'poutcome',
 'emp.var.rate',
 'cons.price.idx',
 'cons.conf.idx',
 'euribor3m',
 'nr.employed',
 'y']
```

 Explain code

POWERED BY  datalab

Let's write a relation that uses multiple functions to analyze the data.

```
filter("duration < 100").project("job,education,loan").order("job").limit(3).df()
```

 Explain code

POWERED BY  datalab

We have three rows and columns sorted by job and filtered by duration column.

	job	education	loan
0	admin.	university.degree	no
1	admin.	high.school	yes
2	admin.	high.school	no

DuckDB Query Function

The DuckDB query function allows SQL queries to be executed within the database, returning results that can be converted into various formats for further analysis.

In the code example, we are running the SQL query to find out the job titles of clients over the age of 30, count the number of clients contacted for each job, and calculate the average duration of the campaign.

Take the [SQL Fundamentals](#) skill track to learn how to manage a relational database and execute queries for simple data analysis.

```
res = duckdb.query("""SELECT
                        job,
                        COUNT(*) AS total_clients_contacted,
                        AVG(duration) AS avg_campaign_duration,
                    FROM
                        'bank-marketing.csv'
                    WHERE
                        age > 30
                    GROUP BY
                        job
                    ORDER BY
                        total_clients_contacted DESC;""")

res.df()
```

 Explain code

POWERED BY  datalab

	job	total_clients_contacted	avg_campaign_duration
0	admin.	8276	262.9604881585
1	blue-collar	7763	263.795955172
2	technician	5578	249.0123700251
3	services	3054	256.7423051735
4	management	2658	257.0127915726
5	retired	1715	273.8915451895
6	entrepreneur	1318	256.9188163885
7	self-employed	1161	267.5004306632
8	housemaid	1001	247.5194805195
9	unemployed	845	251.9621301775

We will now close the connection to the database and release any resources associated with that connection, preventing potential memory and file handle leaks.

```
con.close()
```

 Explain code

POWERED BY  datalab

If you are facing issues running the above code, please have a look at the [Getting Started with DuckDB](#) workspace.

Building a RAG Application with DuckDB

In the first project, we will learn to build an RAG application with LlamaIndex and use DuckDB as a Vector database and retriever.

Setting up

Install all the necessary Python packages that will be used to create and retrieve the index.

```
%%capture
%pip install duckdb
%pip install llama-index
%pip install llama-index-vector-stores-duckdb
```

 Explain code

POWERED BY  datalab

Import the necessary Python package with the functions.

```
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader
from llama_index.vector_stores.duckdb import DuckDBVectorStore
from llama_index.core import StorageContext

from IPython.display import Markdown, display
```

 Explain code

POWERED BY  datalab

Setting up GPT-4o and Embedding Model

For a language model, we will use the latest GPT4o model and the OpenAI API. To create the large language model (LLM) client, you just have to provide a model name and [API key](#).

```
import os
from llama_index.llms.openai import OpenAI

llm = OpenAI(model="gpt-4o", api_key=os.environ["OPENAI_API_KEY"])
```


 Explain code

POWERED BY  datalab

Then, we will create the embed model client using the OpenAI `text-embedding-3-small` model.

Note: Providing an OpenAI API key is optional if the environment variable is set with the name “OPENAI_API_KEY” on your development environment.

```
from llama_index.embeddings.openai import OpenAIEmbedding
embed_model = OpenAIEmbedding(
    model="text-embedding-3-small",
)
```

 Explain code

POWERED BY  datalab

We will make OpenAI LLM and Embedding models global for all LlamaIndex functions to use. In short, these models will be set as default.

```
from llama_index.core import Settings

Settings.llm = llm
Settings.embed_model = embed_model
```

 Explain code

POWERED BY  datalab

Using DuckDB as a vector database

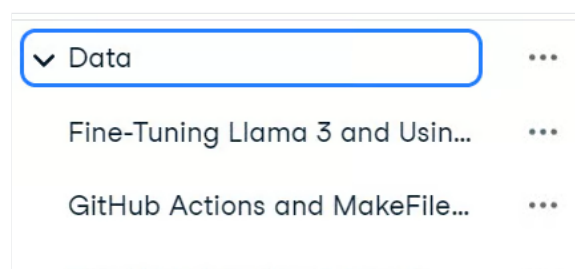
For our project, we will load the PDF files from the data folder. These PDF files are tutorials from DataCamp that are saved as PDF files using the browser’s print function.

Provide the folder directory to the `SimpleDirectoryReader` function and load the data.

```
documents = SimpleDirectoryReader("Data").load_data()
```

 Explain code

POWERED BY  datalab



Then, create the vector store called “blog” using an existing database called “datacamp.duckdb.” After that, convert the PDF’s data into embeddings and store them in the vector store.

```
vector_store = DuckDBVectorStore(database_name = "datacamp.duckdb", table_name="embeddings")
storage_context = StorageContext.from_defaults(vector_store=vector_store)

index = VectorStoreIndex.from_documents(
    documents, storage_context=storage_context
)
```

 Explain code

POWERED BY  datalab

To check if our vector store was successfully created, we will connect the database using the DuckDB Python API and run the SQL query to display all the tables in the database.

```
import duckdb
con = duckdb.connect("datacamp.duckdb")

con.execute("SHOW ALL TABLES").fetchdf()
```

 Explain code

POWERED BY  datalab

We have two tables: a “bank” promotional table and a “blog” table, which is a vector store. The “blog” table has an “embedding” column where all the embeddings are stored.

	database	schema	name	column_names	column_types	temporary
0	datacamp	main	bank	["age", "job", "marital", "educati...	["BIGINT", "VARCHAR", "VARCH...	False
1	datacamp	main	blog	["node_id", "text", "embedding", ...]	["VARCHAR", "VARCHAR", "FLO...	False

Creating a simple RAG application

Convert the index into the query engine, which will automatically first search the vector database for similar documents and use the additional context to generate the response.

To test the RAG query engine, we will ask the question about the tutorial.

```
query_engine = index.as_query_engine()
response = query_engine.query("Who wrote 'GitHub Actions and MakeFile: A Hands-on Introduction'")
display(Markdown(f"<b>{response}</b>"))
```

 Explain code

POWERED BY  datalab

And the answer is correct.

```
The author of "GitHub Actions and MakeFile: A Hands-on Introduction" is Abir i
```

 Explain code

POWERED BY  datalab

Creating a RAG chatbot with memory

Now, let’s create an advanced RAG application that uses the conversation history to generate the response. For that, we have to create a chat memory buffer and then a chat engine with memory, LLM, and vector store retriever.

```
from llama_index.core.memory import ChatMemoryBuffer
from llama_index.core.chat_engine import CondensePlusContextChatEngine

memory = ChatMemoryBuffer.from_defaults(token_limit=3900)

chat_engine = CondensePlusContextChatEngine.from_defaults(
    index.as_retriever(),
    memory=memory,
    llm=llm
)

response = chat_engine.chat(
    "What is the easiest way of finetuning the Llama 3 model? Please provide step by step instructions."
)

display(Markdown(response.response))
```

We asked the chat engine how to fine-tune the Llama 3 model, and it used the vector store to give a highly accurate answer.

The easiest way to fine-tune the Llama 3 model involves using the Kaggle Notebook and following a series of steps. Here's a detailed step-by-step guide based on the provided documents:

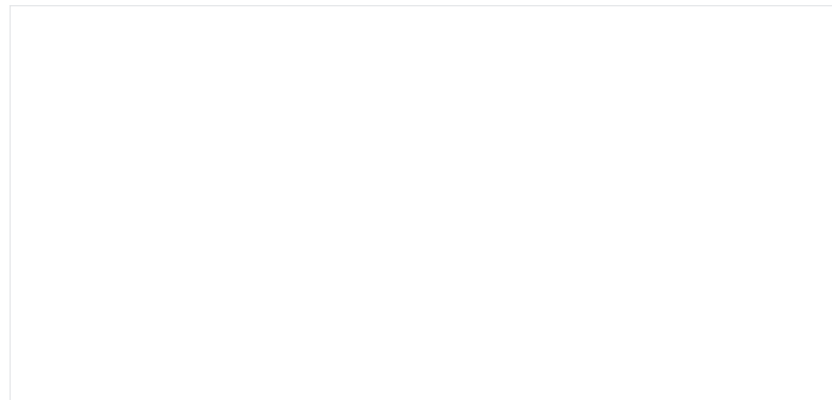
Step-by-Step Instructions for Fine-Tuning Llama 3

- 1. Fill Out the Meta Download Form:**
 - Before you start, you need to fill out the Meta download form with your Kaggle email address. This is necessary to access the Llama 3 model.
- 2. Accept the Agreement on Kaggle:**
 - Go to the Llama 3 model page on Kaggle and accept the agreement. The approval process may take one to two days.
- 3. Launch a New Notebook on Kaggle:**
 - Once you have access, launch a new Notebook on Kaggle.
- 4. Add the Llama 3 Model:**
 - In the Notebook, click the `+ Add Input` button.
 - Select the `Models` option.

To check if the memory buffer is working correctly, we will ask a follow-up question.

```
response = chat_engine.chat(
    "Could you please provide more details about the Post Fine-Tuning Steps?"
)
display(Markdown(response.response))
```

The chat engine remembered the previous conversation and responded accordingly.



If you are facing issues running the above code, please have a look at the [Building a RAG application with DuckDB](#) workspace.

Building a DuckDB SQL Query Engine Using an LLM

In the second project, we will use DuckDB as an SQL query engine. This involves integrating the database engine with the GPT-4o model to generate natural language responses to questions about the database.

Install `duckdb-engine` to create a database engine using SQLAlchemy.

```
%pip install duckdb-engine -q
```

Loading the DuckDB database

We will load the DuckDB database using the `create_engine` function and then write a simple SQL query to check whether it is successfully loaded.

```
from sqlalchemy import create_engine

engine = create_engine("duckdb:///datacamp.duckdb")
with engine.connect() as connection:
    cursor = connection.exec_driver_sql("SELECT * FROM bank LIMIT 3")
    print(cursor.fetchall())
```

[⚡ Explain code](#)POWERED BY  datalab

Prefect. Our DuckDB database engine is ready to be used.

```
[(56, 'housemaid', 'married', 'basic.4y', 'no', 'no', 'no', 'telephone', 'mr', 'hi
```

[⚡ Explain code](#)POWERED BY  datalab

Now, we have to create a database Tool using the `SQLDatabase` function. Provide it with an engine object and table name.

```
from llama_index.core import SQLDatabase
sql_database = SQLDatabase(engine, include_tables=["bank"])
```

[⚡ Explain code](#)POWERED BY  datalab

Building the SQL query engine

Create the SQL query engine using the `NLSQLTableQueryEngine` function by providing it with the LlamaIndex SQL database object.

```
from llama_index.core.query_engine import NLSQLTableQueryEngine

query_engine = NLSQLTableQueryEngine(sql_database)
```

[⚡ Explain code](#)POWERED BY  datalab

Ask the question from the query engine about the “bank” table in the natural language.

```
response = query_engine.query("Which is the longest running campaign?")

print(response.response)
```

[⚡ Explain code](#)POWERED BY  datalab

In response, we will get the answer to your query in natural languages. This is awesome, don't you think?

```
The longest running campaign in the database has a duration of 4918 days.
```

[⚡ Explain code](#)POWERED BY  datalab

Let's ask a complex question.

```
response = query_engine.query("Which type of job has the most housing loan?")

print(response.response)
```

[⚡ Explain code](#)POWERED BY  datalab

The answer is precise, with additional information.

```
The job type with the most housing loans is 'admin.' with 5559 housing loans. This
```

[⚡ Explain code](#)POWERED BY  datalab

To check what is going on on the back end, we will print the metadata.

```
print(response.metadata)
```

[⚡ Explain code](#)POWERED BY  datalab

As we can see, GPT-4o first generates the SQL query, runs the query to get the result, and uses the result to generate the response. This multi-step process is achieved through two lines of code.


```
{'d4ddf03c-337e-4ee6-957a-5fd2cfaa4b1c': {}, 'sql_query': "SELECT job, COUNT" us
```

 Explain code

POWERED BY  datalab

Close the engine when you are done with the project.

```
engine.close()
```

 Explain code

POWERED BY  datalab

If you are facing issues running the above code, please have a look at the [DuckDB SQL Query Engine](#) workspace.

Conclusion

DuckDB is fast, easy to use, and integrates seamlessly with numerous data and AI frameworks. As a data scientist, you will find that it takes only a few minutes to get accustomed to its API and start using it like any other Python package. One of the best features of DuckDB is that it has no dependencies, meaning you can use it virtually anywhere without worrying about hosting or additional setup.

In this tutorial, we have learned about DuckDB and its key features. We have also explored the DuckDB Python API, using it to create a table and perform simple data analysis. The second half of the tutorial covered two projects: one involving a Retrieval-Augmented Generation (RAG) application with DuckDB as a vector database and the other demonstrating DuckDB as an SQL query engine.

Before jumping into using a SQL query engine or integrating a database with AI, you need a basic understanding of SQL and data analysis. You can write the query, but how would you know what question to ask? This is where a basic knowledge of data analysis and SQL comes in. You can gain this knowledge by completing the [Associate Data Analyst in SQL](#) career track.



AUTHOR

Abid Ali Awan



As a certified data scientist, I am passionate about leveraging cutting-edge technology to create innovative machine learning applications. With a strong background in speech recognition, data analysis and reporting, MLOps, conversational AI, and NLP, I have honed my skills in developing intelligent systems that can make a real impact. In addition to my technical expertise, I am also a skilled communicator with a talent for distilling complex concepts into clear and concise language. As a result, I have become a sought-after blogger on data science, sharing my insights and experiences with a growing community of fellow data professionals. Currently, I am focusing on content creation and editing, working with large language models to develop powerful and engaging content that can help businesses and individuals alike make the most of their data.

TUTORIALS ▾

Category ▾ 🔍



🌐 EN 🍽

Top DataCamp Courses

📖 COURSE

Vector Databases for Embeddings with Pinecone

🕒 3 hr 🧑 464

Discover how the Pinecone vector database is revolutionizing AI application development!

[See Details →](#)

[Start Course](#)

[See More →](#)

Related

BLOG

An Introduction to DuckDB:
What is It and Why Should You...

BLOG

DuckDB makes SQL a first-class
citizen on DataLab



TUTORIAL

A Comprehensive Guide to
Databricks Lakehouse AI For...

[See More →](#)

Grow your data skills with DataCamp for Mobile

Make progress on the go with our mobile courses and daily 5-minute coding challenges.



LEARN

[Learn Python](#)

[Learn R](#)

[Learn AI](#)

[Learn SQL](#)

[Learn Power BI](#)

[Learn Tableau](#)

[Learn Data Engineering](#)

[Assessments](#)

[Career Tracks](#)

[Skill Tracks](#)

[Courses](#)

[Data Science Roadmap](#)

DATA COURSES

[Python Courses](#)

[R Courses](#)

[SQL Courses](#)

[Power BI Courses](#)

[Tableau Courses](#)

[Alteryx Courses](#)

[Azure Courses](#)

[Google Sheets Courses](#)

[AI Courses](#)

[Data Analysis Courses](#)

[Data Visualization Courses](#)

[Machine Learning Courses](#)

[Data Engineering Courses](#)

[Probability & Statistics Courses](#)

DATALAB

[Get Started](#)

[Pricing](#)

[Security](#)

[Documentation](#)

CERTIFICATION

[Certifications](#)

[Data Scientist](#)

[Data Analyst](#)

[Data Engineer](#)

[SQL Associate](#)

[Power BI Data Analyst](#)

[Tableau Certified Data Analyst](#)

[Azure Fundamentals](#)

[AI Fundamentals](#)

RESOURCES

[Resource Center](#)

[Upcoming Events](#)

[Blog](#)

[Code-Alongs](#)

[Tutorials](#)

[Docs](#)

[Open Source](#)

[RDocumentation](#)

[Course Editor](#)

[Book a Demo with DataCamp for Business](#)

[Data Portfolio](#)

[Portfolio Leaderboard](#)

PLANS

[Pricing](#)

[For Business](#)

[For Universities](#)

[Discounts, Promos & Sales](#)

[DataCamp Donates](#)

FOR BUSINESS

[Business Pricing](#)

[Teams Plan](#)

[Data & AI Unlimited Plan](#)

[Customer Stories](#)

[Partner Program](#)

ABOUT

[About Us](#)

[Learner Stories](#)

[Careers](#)

[Become an Instructor](#)

[Press](#)

[Leadership](#)

[Contact Us](#)

[DataCamp Español](#)

[DataCamp Português](#)

[DataCamp Deutsch](#)

[DataCamp Français](#)

SUPPORT

[Help Center](#)

[Become an Affiliate](#)



[Privacy Policy](#) [Cookie Notice](#) [Do Not Sell My Personal Information](#) [Accessibility](#) [Security](#) [Terms of Use](#)

© 2024 DataCamp, Inc. All Rights Reserved.