**TY B.Tech. (CSE) – II [ 2022-23 ]**

**5CS372 : Advanced Database System Lab.**

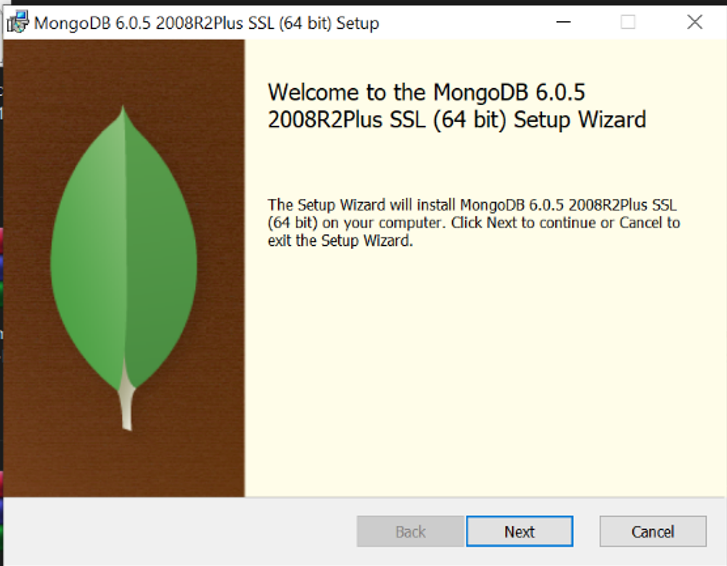
**Assignment No. 9**

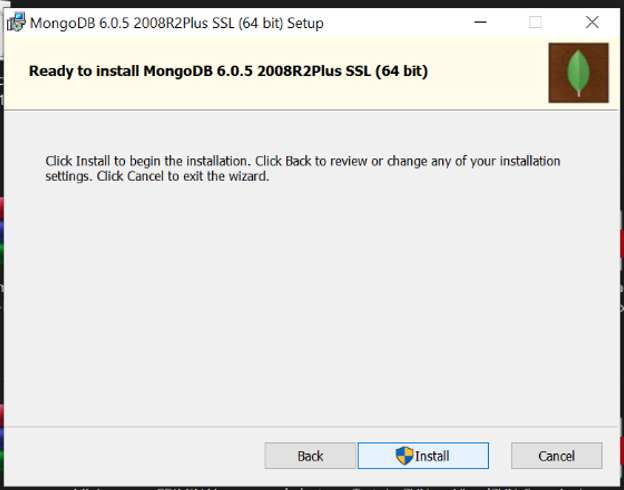
Q1. Install & deploy the following cloud databases on windows platform :

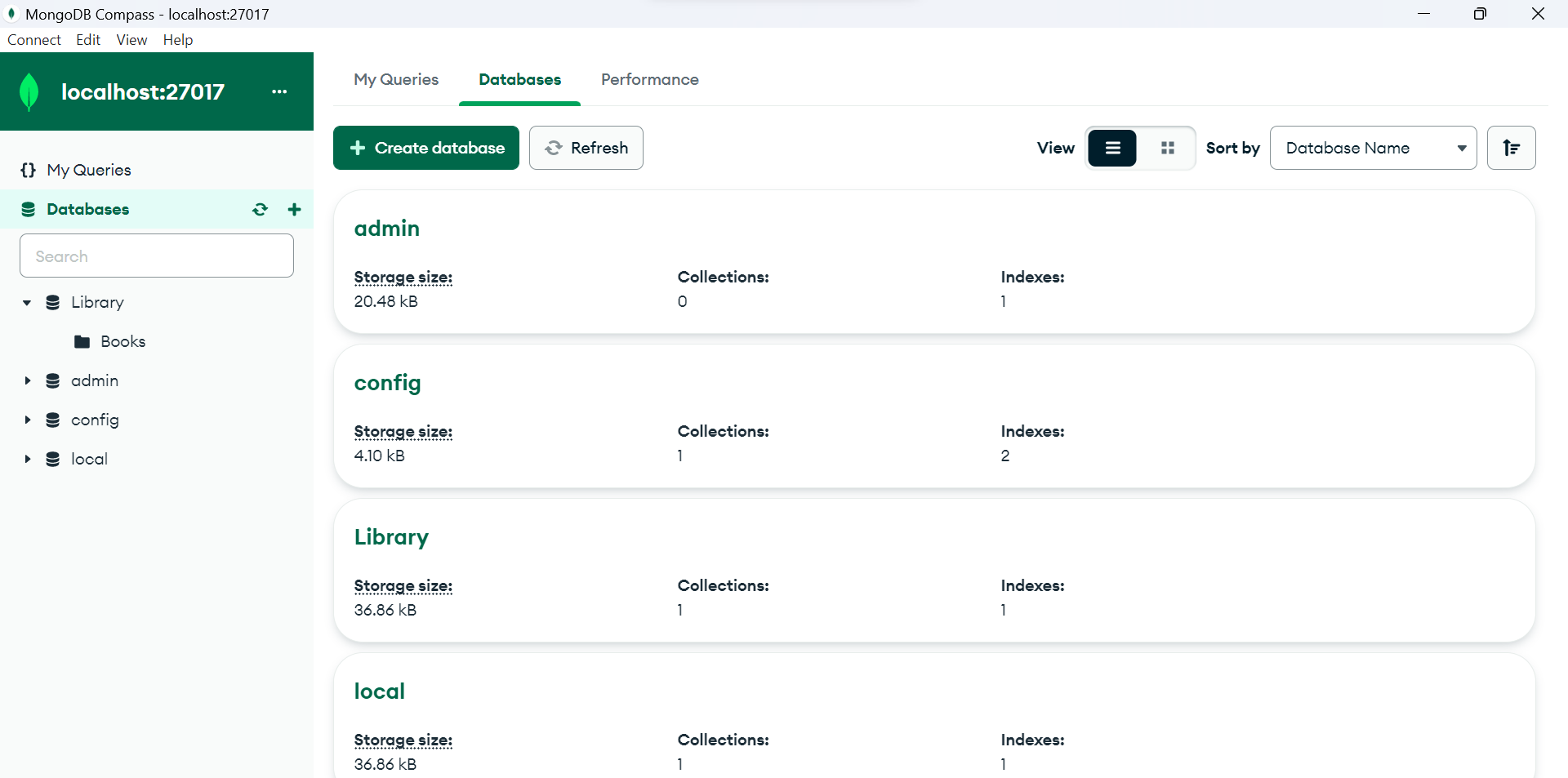
1. MongoDB
2. CassandraDB

**Install MongoDB**

* Download the latest version of MongoDB for Windows from the official website: https://www.mongodb.com/try/download/community
* Run the installer and choose the "Complete" setup type
* Leave the default installation directory and click "Next"
* Choose "Install MongoDB Compass" if you want to install the MongoDB GUI tool
* Click "Install" and wait for the installation to complete
* Once the installation is complete, MongoDB is ready to use



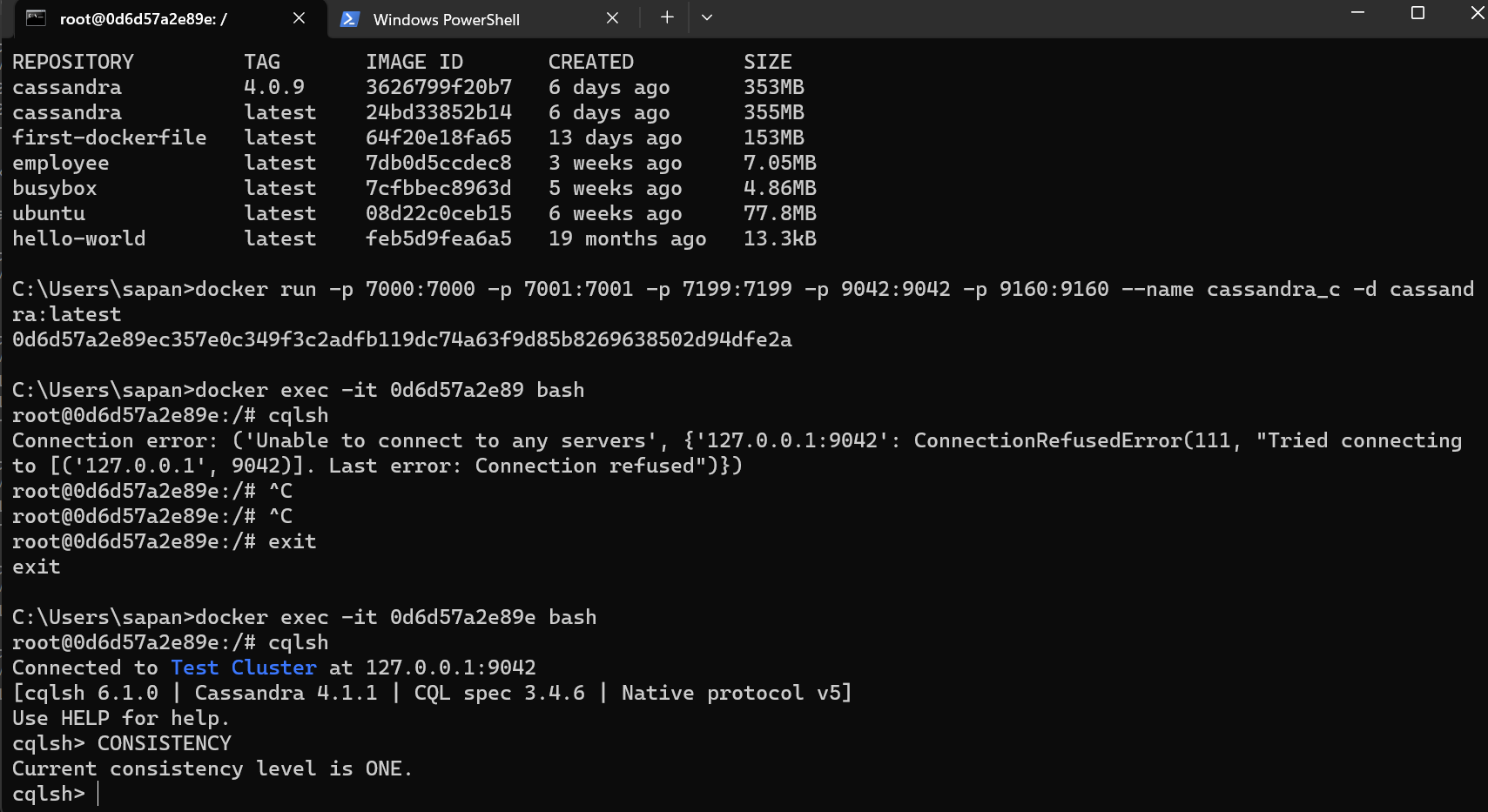




**Install CassandraDB**

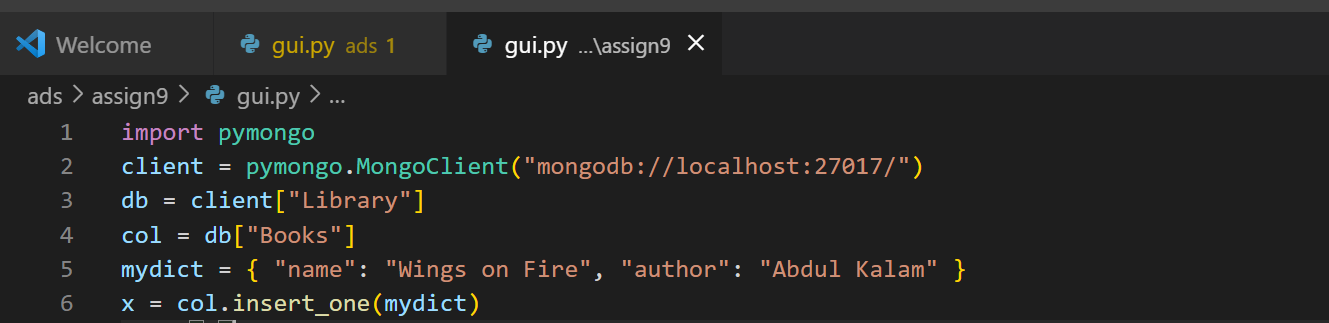
* -Download the latest version of CassandraDB for Windows from the official website: https://cassandra.apache.org/download/
* Extract the downloaded file to a directory of your choice
  + Open Start on Windows 10.
  + Search for Command Prompt, right-click the top result, and select the Run as administrator option.
  + Type the following command to use tar to extract the files and press Enter:
  + tar -xvzf C:\PATH\TO\FILE\FILE-NAME.tar.gz -C C:\PATH\TO\FOLDER\EXTRACTION
* Rename the extracted directory to "cassandra"
* Add the Cassandra bin directory to your PATH environment variable by following these steps:
* Open the Start menu and search for "Environment Variables"
* Click on "Edit the system environment variables"
* Click on the "Environment Variables" button
* Under "System variables", find the "Path" variable and click "Edit"
* Click "New" and add the path to the Cassandra bin directory (e.g. C:\cassandra\bin)
* Click "OK" to close all windows

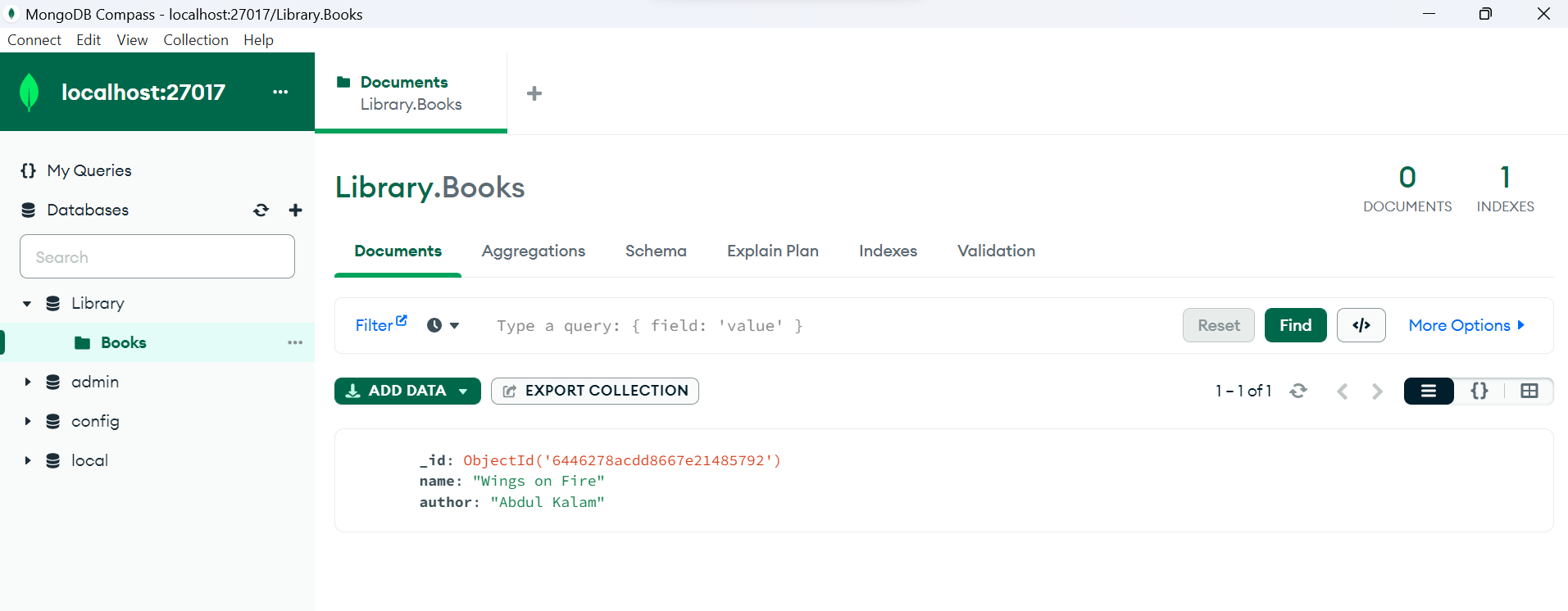
Open a command prompt and type "cassandra" to start the Cassandra server. Once the server is running, CassandraDB is ready to use



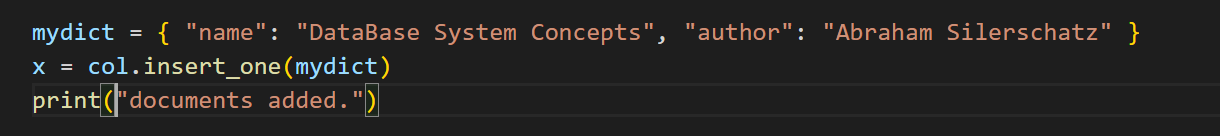
Q2. Write Python desktop Application to demonstrate the CRUD operation with above backend cloud databases. ***Assume any database***.

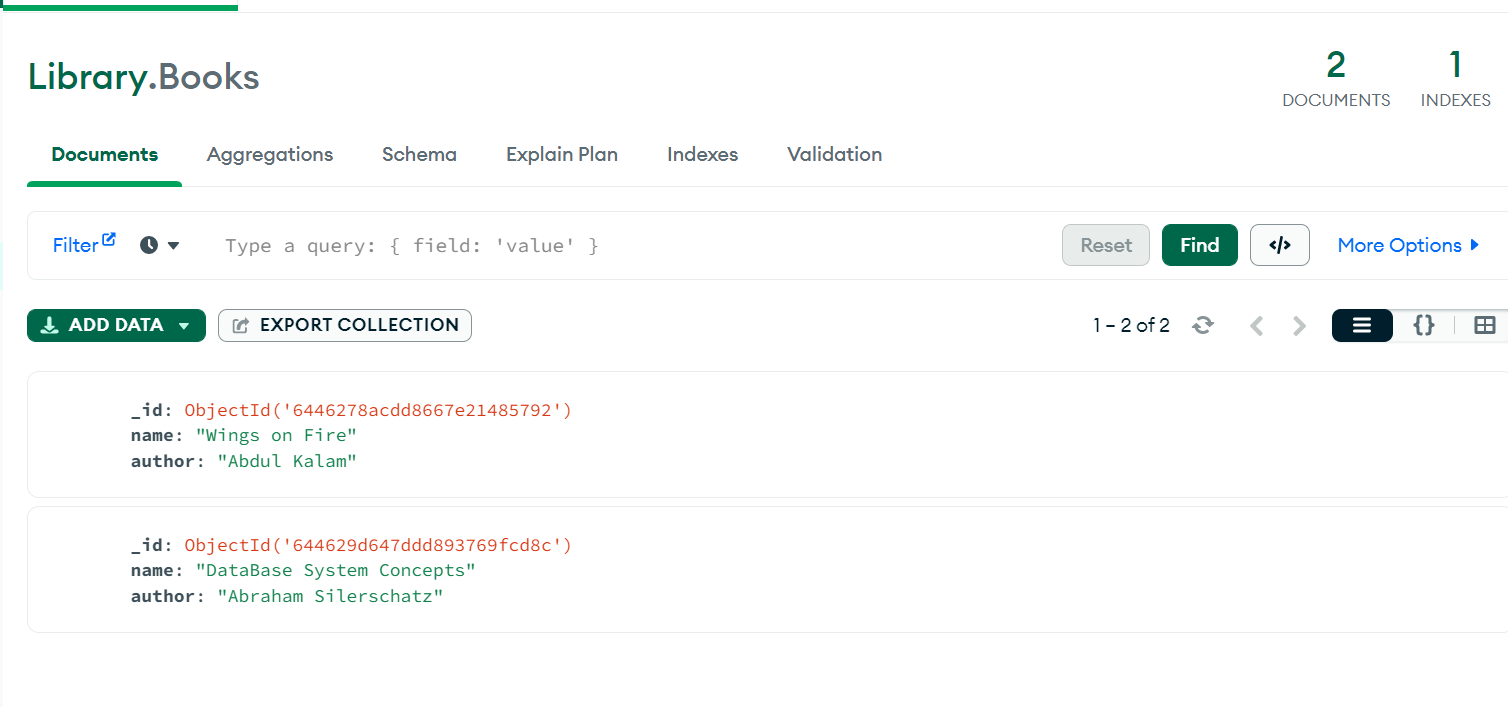
**CREATE / INSERT –**

****

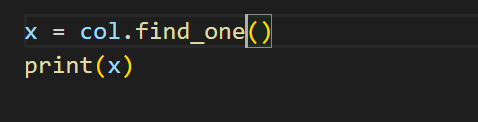
****

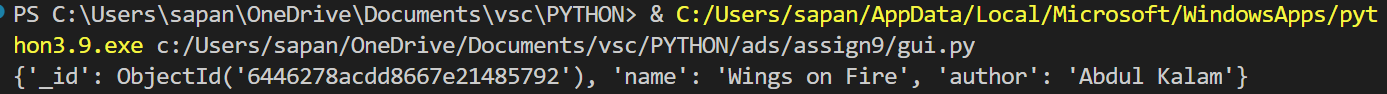
One more document –



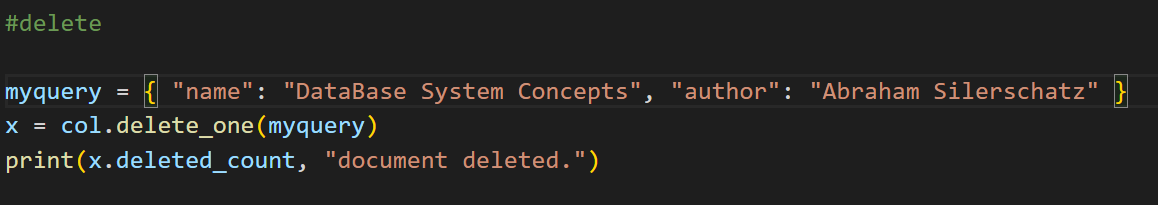
****

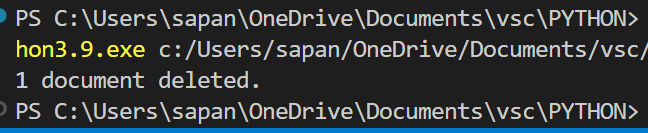
**READ –**

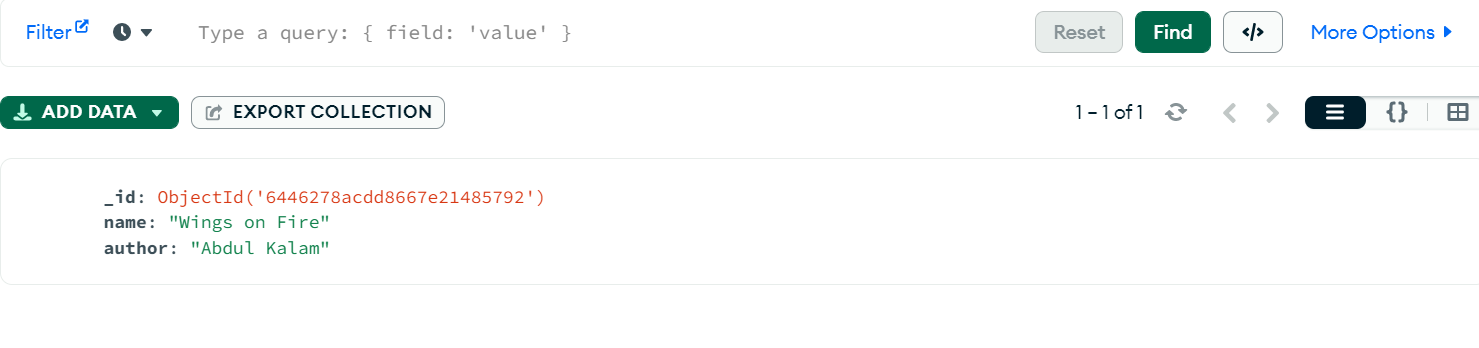
****

****

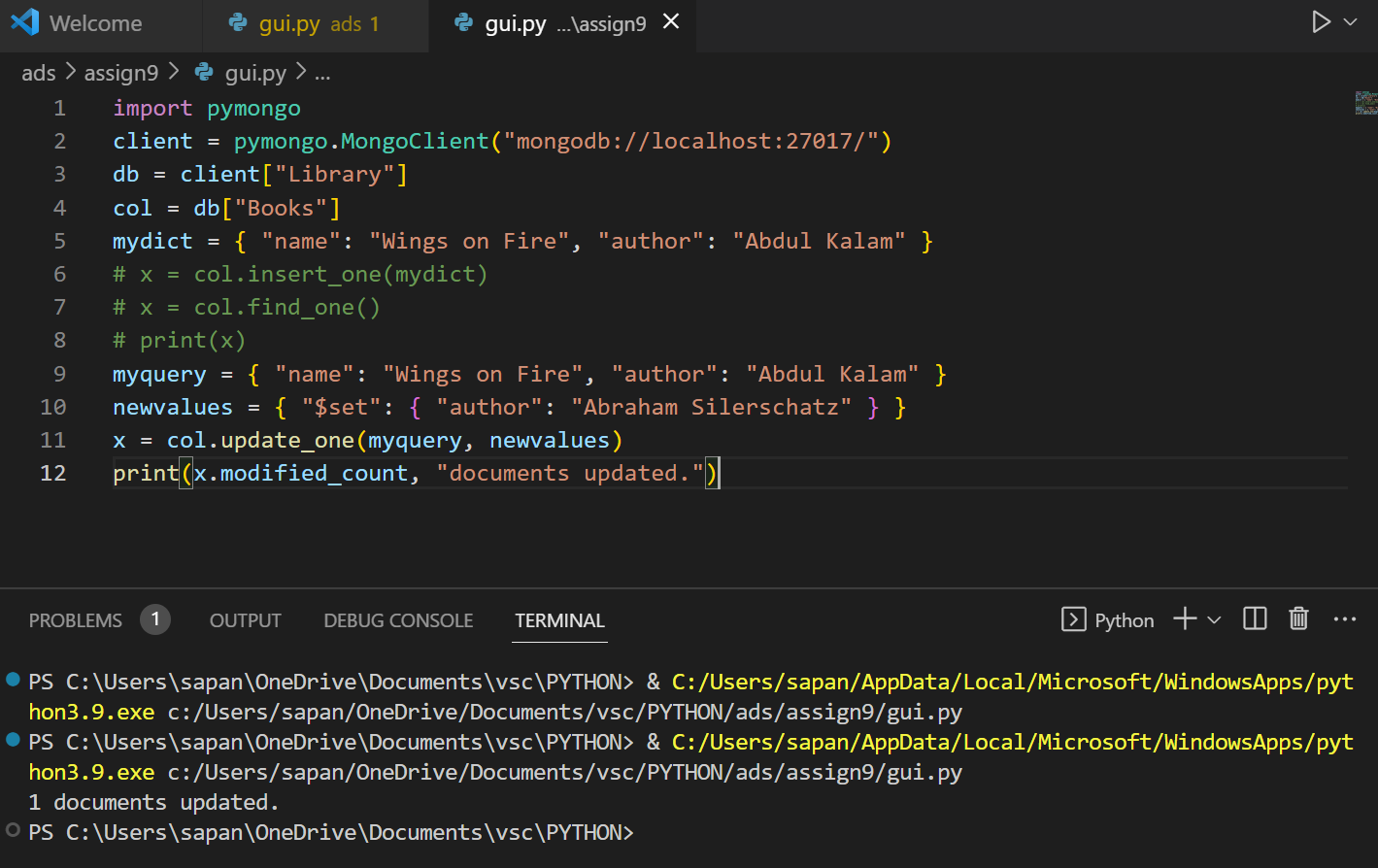
**DELETE –**

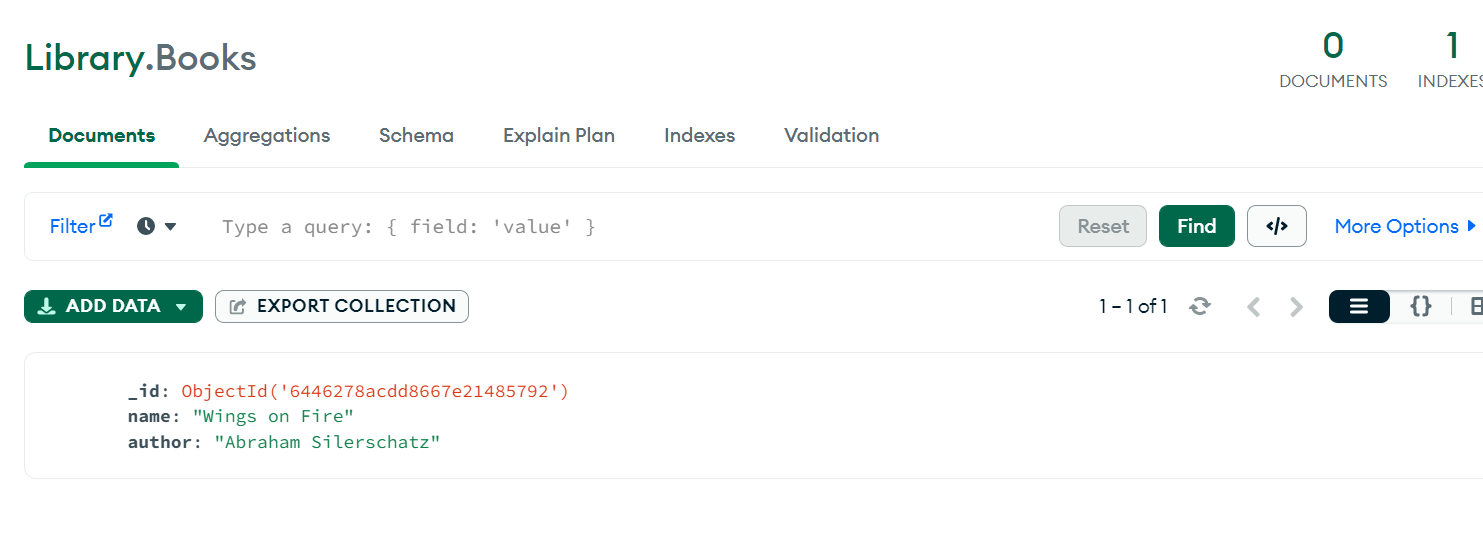
****

****

****

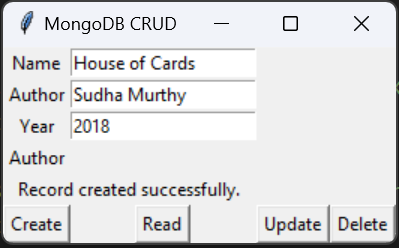
**UPDATE -**

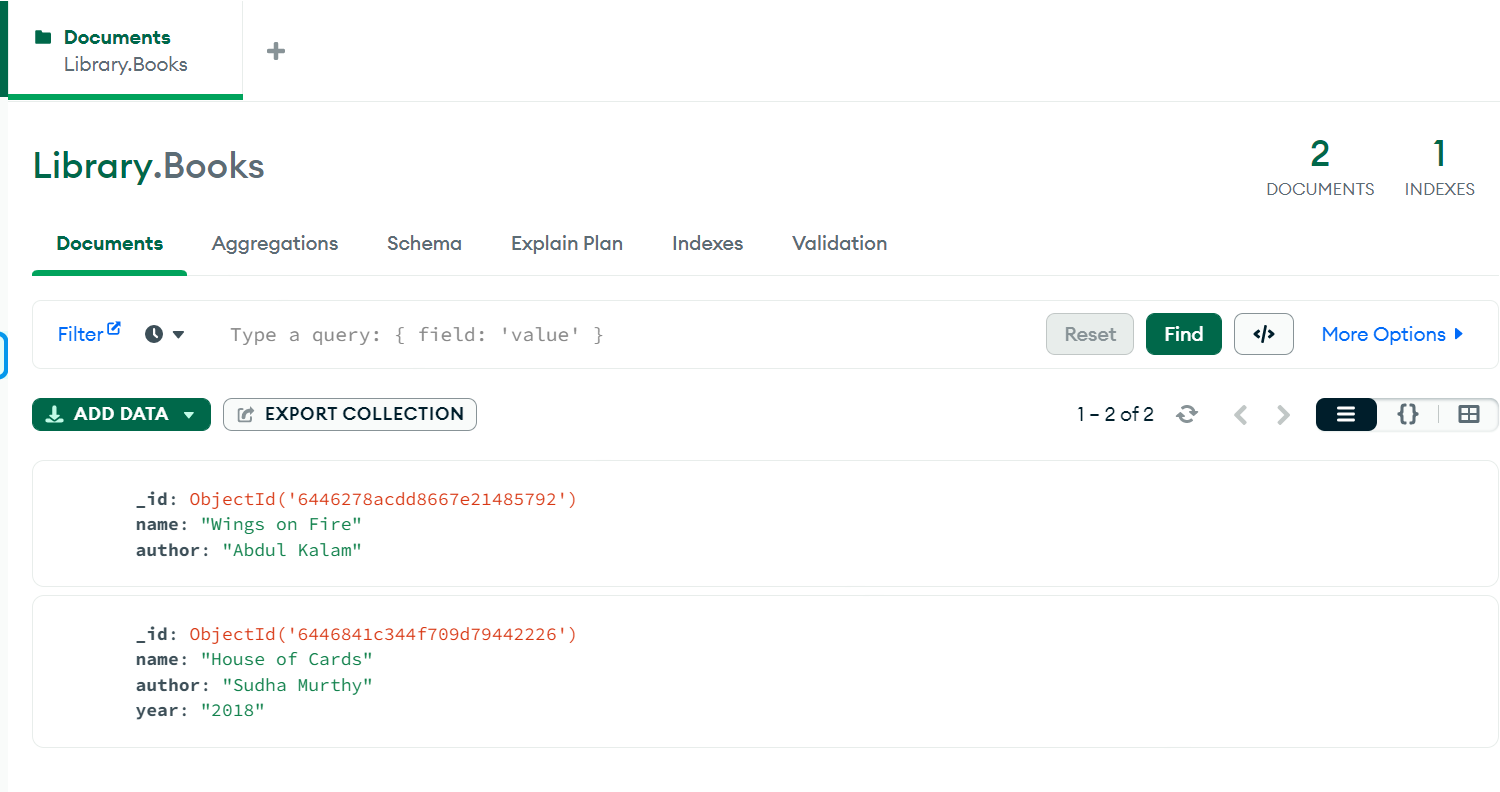
****



USING GUI –

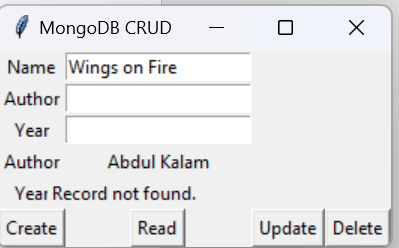
**CREATE -**



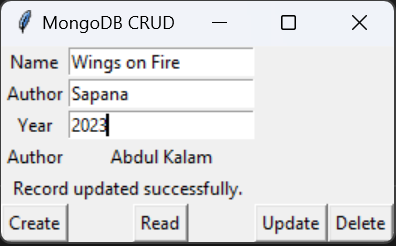


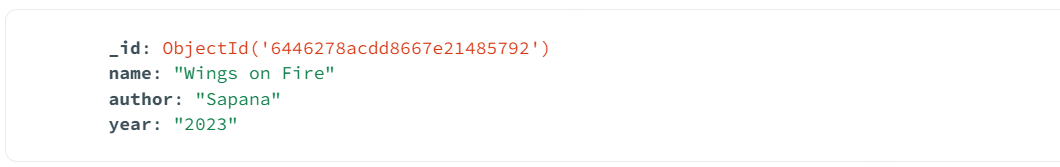
**READ –**

**Will give details of a book when name is typed in**

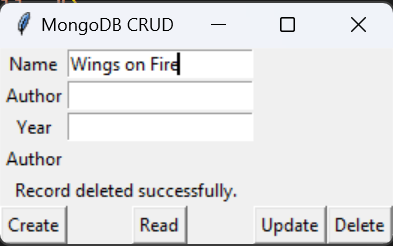
****

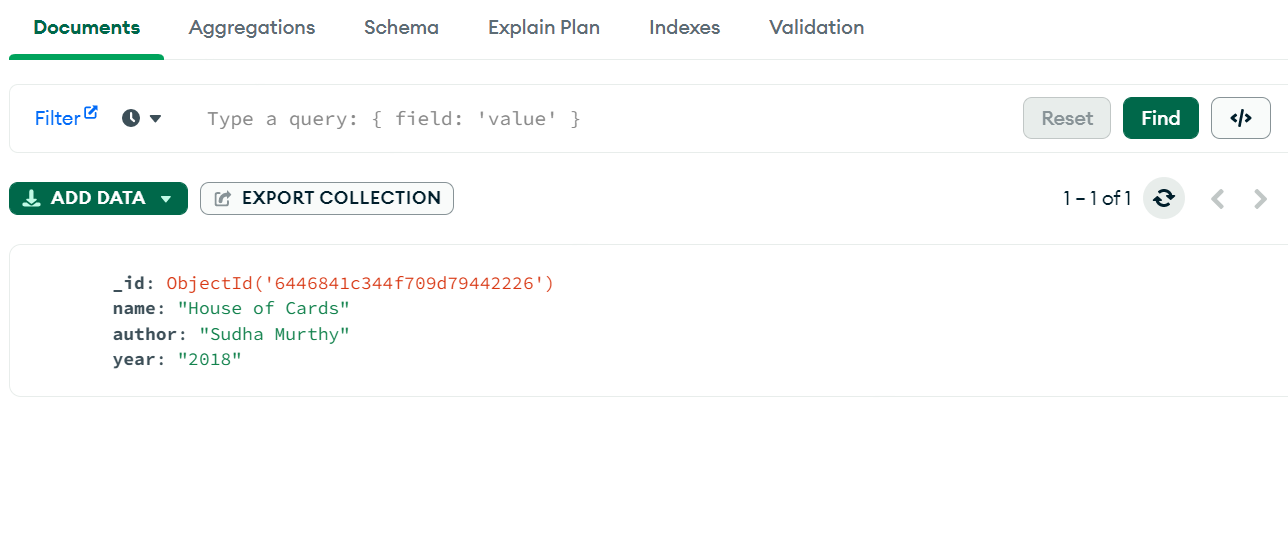
**UPDATE –**

****

****

**DELETE –**

****

****

**GUI CODE –**

import pymongo

import tkinter as tk

from tkinter import \*

client = pymongo.MongoClient("mongodb://localhost:27017/")

db = client["Library"]

col = db["Books"]

#create

def create\_record():

    # Get data from input fields

    name = name\_entry.get()

    author = author\_entry.get()

    year = year\_entry.get()

    # Insert data into MongoDB

    record = {"name": name, "author": author, "year": year}

    result = col.insert\_one(record)

    # Show success message

    status\_label.config(text="Record created successfully.")

#read

def read\_record():

    # Get data from input fields

    name = name\_entry.get()

    # Find data in MongoDB

    record = col.find\_one({"name": name})

    # Show data in output field

    if record:

        author\_output.config(text=record['author'])

        year\_output.config(text=record['year'])

    else:

        author\_output.config(text="")

        year\_output.config(text="")

        status\_label.config(text="Record not found.")

#update

def update\_record():

    # Get data from input fields

    name = name\_entry.get()

    author = author\_entry.get()

    year = year\_entry.get()

    # Update data in MongoDB

    result = col.update\_one({"name": name}, {"$set": {"author": author, "year": year}})

    # Show success message

    if result.modified\_count > 0:

        status\_label.config(text="Record updated successfully.")

    else:

        status\_label.config(text="Record not found.")

#delete

def delete\_record():

    # Get data from input fields

    name = name\_entry.get()

    # Delete data from MongoDB

    result = col.delete\_one({"name": name})

    # Show success message

    if result.deleted\_count > 0:

        author\_output.config(text="")

        year\_output.config(text="")

        status\_label.config(text="Record deleted successfully.")

    else:

        status\_label.config(text="Record not found.")

# Create GUI window

root = Tk()

root.title("MongoDB CRUD")

# Create input fields

name\_label = Label(root, text="Name")

name\_label.grid(row=0, column=0)

name\_entry = Entry(root)

name\_entry.grid(row=0, column=1)

author\_label = Label(root, text="Author")

author\_label.grid(row=1, column=0)

author\_entry = Entry(root)

author\_entry.grid(row=1, column=1)

year\_label = Label(root, text="Year")

year\_label.grid(row=2, column=0)

year\_entry = Entry(root)

year\_entry.grid(row=2, column=1)

# Create output fields

author\_output\_label = Label(root, text="Author")

author\_output\_label.grid(row=3, column=0)

author\_output = Label(root, text="")

author\_output.grid(row=3, column=1)

year\_output\_label = Label(root, text="Year")

year\_output\_label.grid(row=4, column=0)

year\_output = Label(root, text="")

year\_output.grid(row=4, column=1)

status\_label = tk.Label(root, text='')

status\_label.grid(row=4, columnspan=2)

# Create buttons for CRUD operations

create\_button = Button(root, text="Create", command=create\_record)

create\_button.grid(row=5, column=0)

read\_button = Button(root, text="Read", command=read\_record)

read\_button.grid(row=5, column=1)

update\_button = Button(root, text="Update", command=update\_record)

update\_button.grid(row=5,column=2)

delete\_button = Button(root, text="Delete", command=delete\_record)

delete\_button.grid(row=5,column=3)

root.mainloop()