**Assignment-2: NoSQL mongoDB**

**Udiptaman Das**

**16340325**

This assignment has 7 major steps. First two steps include creating mongoDB Atlas account and mongoDB free cluster. Database named “Library” is created in step 3 and in it, we add “Books” collection.

A screenshot of a computer

Description automatically generated with medium confidence

Collection

Database

Figure 1: mongoDB cloud environment

This is what mongoDB cloud looks like. “Library” is the name of the database and “Books” is the name of collection, which we will create later.

!python -m pip install pymongo

from pymongo.mongo\_client import MongoClient

from pymongo.server\_api import ServerApi

uri = "mongodb+srv://udidas:mongo@assignment-2.ulpb8p7.mongodb.net/?retryWrites=true&w=majority"

# Create a new client and connect to the server

client = MongoClient(uri, server\_api=ServerApi('1'))

# Send a ping to confirm a successful connection

try:

    client.admin.command('ping')

    print("Pinged your deployment. You successfully connected to MongoDB!")

except Exception as e:

    print(e)

Google colab is used for coding and above is a code snippet that is used to establish a connection with the mongoDB via cloud.

A screen shot of a computer program

Description automatically generated with low confidence

Figure 2: Connecting to mongoDB from colab

Above is the output for the code snippet above. Successful connection displays the following message “*Pinged your deployment. You successfully connected to MongoDB!*”. Once the connection is established, the database can be populated right from my colab file.

collection.insert\_many(

    [

        {

            'ISBN': '9780316015844',

            'Title': 'The Catcher in the Rye',

            'Author': 'J.D. Salinger',

            'Date': '1951',

            'Publisher': 'Little, Brown and Company',

            'Category': 'Fiction',

            'Quantity': 10,

            'Price': 12.99

        },

Similar to the above lines, information of a total of 12 books have been added to the “Books” collection using the “*insert\_many*” function. 12 such dictionaries are added in the code.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 3: Data added to Library database from colab

Above picture shows the data in collection named “Books”, that we injected into our database.

Once we have all the data in the database, we can proceed to creating a search function that gives the system user the ability to search for any books.

def search(attribute,value,condition=None):

  if condition == None:

    search\_result = collection.find({attribute:value})

  else:

    search\_result = collection.find({attribute:{condition:value}})

  return pd.DataFrame(search\_result)

This is how my search function works:

1. It takes in “attribute”, “value” and “condition” as input.
2. “attribute” can be any of the properties of a book. It could be anything like “Date”, “Category”, and so on. “value” will be the parameter with which conditions will be matched and accordingly display outputs.
3. “condition” is set to None by default.
4. The user may or may not set any conditions for the search. If so, the search query will consider “=” condition by default. Else, whatever condition is provided, it will be used by the search function.
5. The “search\_result” parameter in the code stores all the books that satisfy the specified condition.
6. Finally display information regarding all the available books that fulfil the search condition as a dataframe.

A screenshot of a computer program

Description automatically generated with low confidence

Figure 4: Search by year of publishing

Above are search queries based on the attribute “Date”. The first code searches for all the books that were published in the year ***1951***. Second code displays all the books that were published ***before*** the year ***1960***.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5: Search by category

*Figure 5* shows all the books belonging to a category that has string ***‘fict’*** in it. All the books that are shown above belong to category “*Fiction*”, “*Science Fiction*” or “*Non-Fiction*”.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 6:Search by Price

Figure 6 shows all the books that cost less than $11.

After this we design a checkout function so that users can select desired books by their choice of attribute like ‘***ISBN***’, ‘***\_id***’ or ‘***Title***’. Once user inputs desired search criteria, the code will check whether that particular book is in stock and once that is confirmed, the user can go on and decide whether it wants to add the book to cart and checkout or maybe go back and look for some other book.

def check\_out(collection):

  cart = []

  total = 0

  search\_by = input("Enter attribute: ")

  while True:

    value = input(f"Enter {search\_by}: ")

    if value == 'done':

      break

    book = collection.find\_one({search\_by:value})

    if book['Quantity'] == 0:

      print('This book is out of stock')

    else:

      cart.append(book)

      total +=book['Price']

  confirmation = input(f"Total = {total}\nProceed to checkout: (y/n)")

  if confirmation == 'y':

    for book in cart:

      quantity = book["Quantity"]

      if quantity > 0:

        collection.update\_one({f"{search\_by}": book[f"{search\_by}"]},{"$set": {"Quantity": quantity - 1}})

    print("Checkout complete!!")

  else:

    print("checkout aborted!!")

Users can add as many books as they like to the cart and confirm checkout after looking at the total amount to be paid. Once purchases are confirmed, the “*Quantity*” parameter will be updated, which is reflected in the mongoDB dataset in real time.

Following is an execution of the checkout function

* First the user searches for a book by “*Title*”. A screenshot of a computer

  Description automatically generated with medium confidence
* Next step is to enter the desired titles and once all the desired books are selected, the user types “**done**” to proceed to confirmation. A screenshot of a computer

  Description automatically generated
* Once all the books are selected, total price is displayed for the user to confirm purchase by entering ***y*** meaning yes and ***n*** meaning no. A screenshot of a computer

  Description automatically generated
* On entering ***y***, the purchase will be confirmed and database will be updated accordingly A screenshot of a computer

  Description automatically generated with medium confidence

Following images show changes in the database before and after executing

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer code

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generatedA screenshot of a computer code

Description automatically generated with medium confidenceA screenshot of a computer code

Description automatically generated with low confidence

*Figure 6:Before and after checkout*