Quiz 1: DADS5001

In [1]:
```
pip install pymongo
```

```
Requirement already satisfied: pymongo in c:\users\pajaya\anaconda3\lib\site-packages (4.3.3)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in c:\users\pajaya\anaconda3\lib\site-packages (from pymongo) (2.2.1)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import os
import pymongo
import json
import tkinter as tk
from tkinter import messagebox
```

In [3]:
```
client = pymongo.MongoClient("mongodb+srv://Lisa:Blackpink@cluster0.cgywexa.mongodb.net/?retryWrites=true&w=majority")
f=open('C:\\Users\\Pajaya\\restaurants.json')
database = client["Restaurants"]
collection = database["restaurants"]
for line in f:
    print(line)
    database = json.loads(line)
    x = collection.insert_one(database)
```

```
{"address": {"building": "1007", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462"}, "boroug
h": "Bronx", "cuisine": "Bakery", "grades": [{"date": {"$date": 1393804800000}, "grade": "A", "score": 2}, {"date": {"$date":
1378857600000}, "grade": "A", "score": 6}, {"date": {"$date": 1358985600000}, "grade": "A", "score": 10}, {"date": {"$date":
1322006400000}, "grade": "A", "score": 9}, {"date": {"$date": 1299715200000}, "grade": "B", "score": 14}], "name": "Morris Pa
rk Bake Shop", "restaurant_id": "30075445"}
```

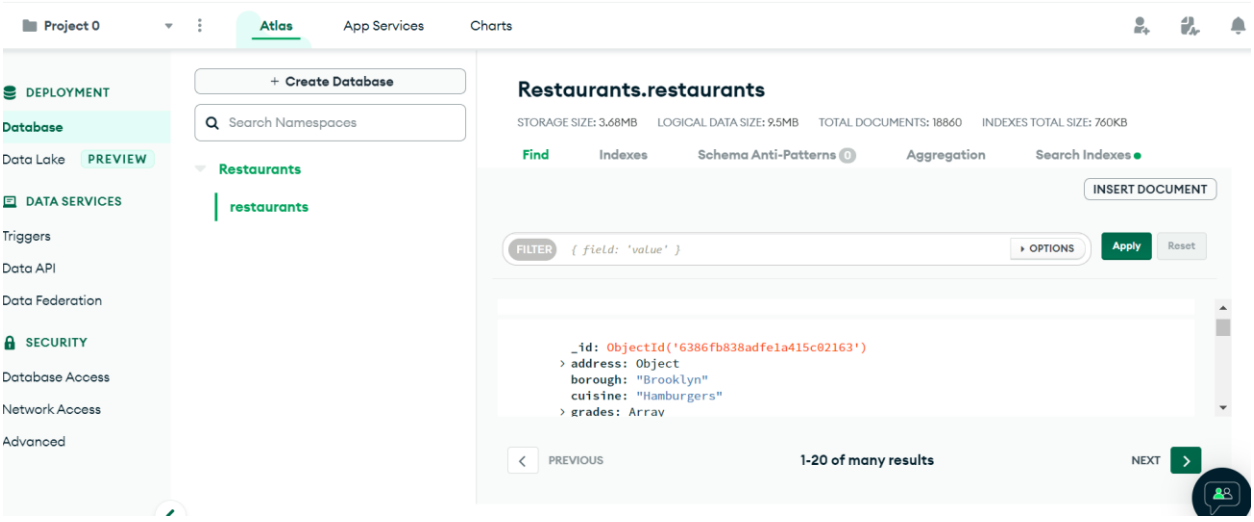
In [*]:
```
lst=[['ID','Address','Borough','Cuisine','Grades','Name']]

def callback(event):
    global lstindex
    li=[]
    li=event.widget._values
    lstindex=li[1]
    cid.set(lst[li[1]][0])
    caddress.set(lst[li[1]][1])
    cborough.set(lst[li[1]][2])
    ccuisine.set(lst[li[1]][3])
    cgrades.set(lst[li[1]][4])
    cname.set(lst[li[1]][5])

def creategrid(n):
    lst.clear()
    lst.append(["ID","Address","Borough","Cuisine","Grades","Name"])
    cursor=collection.find([])
    for text_fromDB in cursor:
            studid=str(text_fromDB['studid'])
            studaddress=str(text_fromDB['studaddress'].encode('utf-8').decode("utf-8"))
            studborough=str(text_fromDB['studborough'].encode('utf-8').decode("utf-8"))
            studcuisine=str(text_fromDB['studcuisine'].encode('utf-8').decode("utf-8"))
            studgrades=str(text_fromDB['studgrades'].encode('utf-8').decode("utf-8"))
            studname=str(text_fromDB['studname'].encode('utf-8').decode("utf-8"))
            lst.append([studid,studaddress,studborough,studcuisine,studgrades,studname])
    for i in range(len(lst)):
            for j in range(len(lst[0])):
                    mgrid=tk.Entry(window,width=10)
                    mgrid.insert(tk.END,lst[i][j])
                    mgrid._values=mgrid.get(), i
                    mgrid.grid(row=i+7,column=j+6)
                    mgrid.bind("<Button-1>",callback)
```

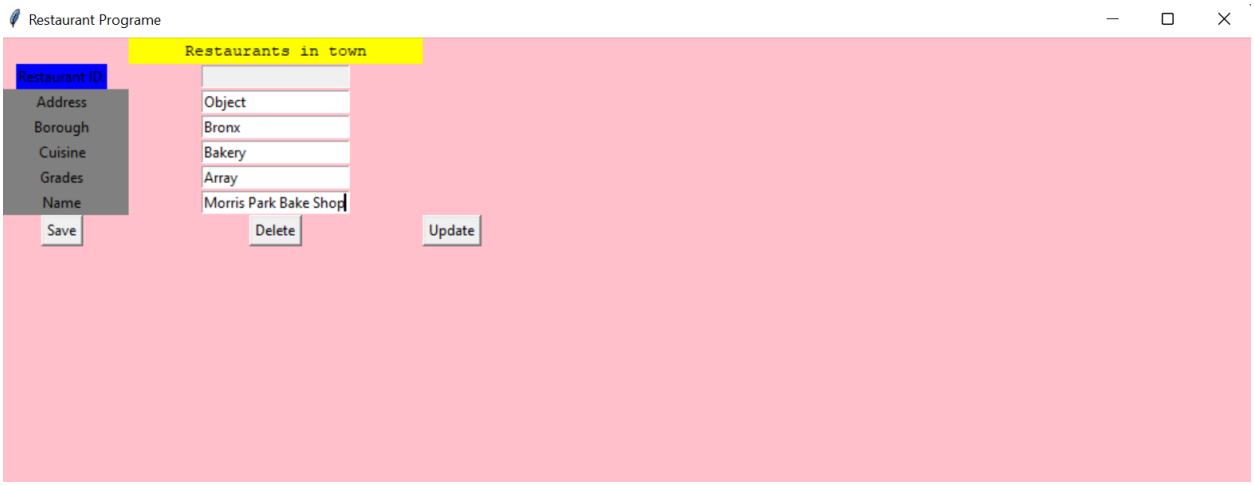
```
    if n==1:
            for label in window.grid_slaves():
                if int(label.grid_info()["row"]) > 6:
                    label.grid_forget()


def msgbox(msg,titlebar):
    result=messagebox.askokcancel(title=titlebar,message=msg)
    return result

def save(): # Create or Insert Data
    r=msgbox ("save record?","record")
    if r==True:
        newid=collection.count_documents({})
        if newid!=0:
            newid=collection.find_one(sort=[("studid",-1)])["studid"] #Retrieve Data
        id=newid+1
        cid.set(id)
        mydict={"studid": int(studid.get()),"studaddress": studaddress.get(),"studborough":studborough.get(),"studcuisine":studcu
        x=collection.insert_one(mydict)
        creategrid(1)
        creategrid(0)

def delete(): # Delete Data
    r=msgbox ("Delete?","record")
    if r==True:
        myquery={"studid":int(studid.get())}
        collection.delete_one(myquery)
        creategrid(1)
        creategrid(0)
```

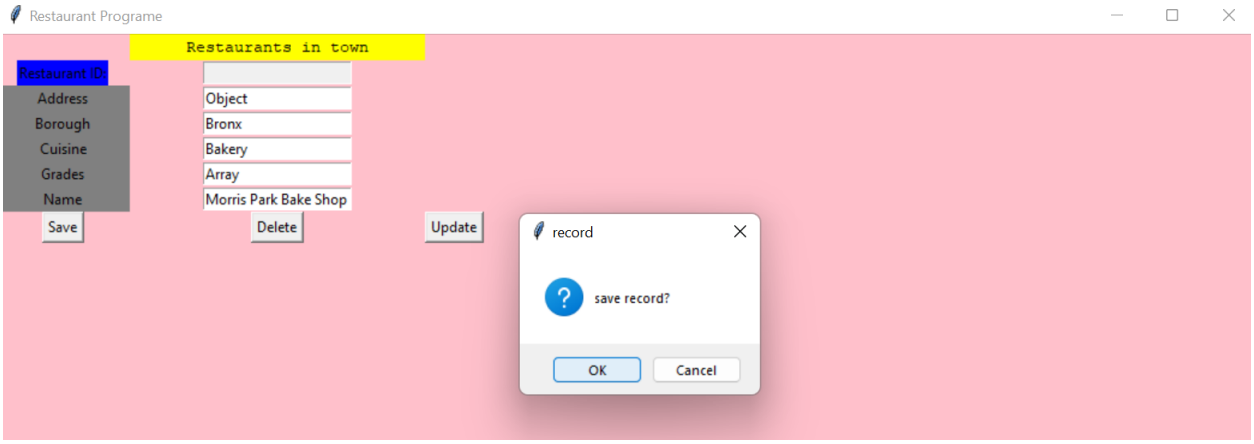
```
def update(): # Update Data
    r=msgbox ("Update?","record")
    if r==True:
        myquery={"studid":int(studid.get())}
        newvalues={"$set":{"studaddress":studaddress.get()}}
        collection.update_one(myquery,newvalues)

        newvalues={"$set":{"studborough":studborough.get()}}
        collection.update_one(myquery,newvalues)

        newvalues={"$set":{"studcuisine":studcuisine.get()}}
        collection.update_one(myquery,newvalues)

        newvalues={"$set":{"studgrades":studgrades.get()}}
        collection.update_one(myquery,newvalues)

        newvalues={"$set":{"studname":studname.get()}}
        collection.update_one(myquery,newvalues)

        creategrid(1)
        creategrid(0)


window = tk.Tk()
window.title("Restaurant Programe")
window.geometry("1050x400")
window.configure(bg="pink")


label = tk.Label(window,text='Restaurants in town',width=30,height=1,bg="yellow",anchor="center")
label.config(font=("Courier",10))
label.grid(column=2,row=1)
```

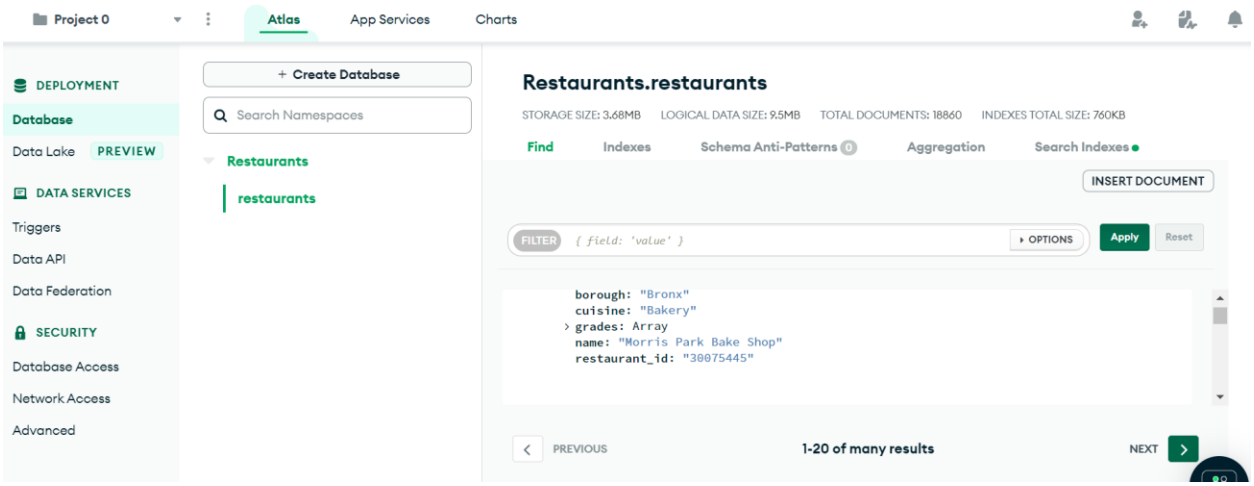
```
label = tk.Label(window,text='Restaurants in town',width=30,height=1,bg="yellow",anchor="center")
label.config(font=("Courier",10))
label.grid(column=2,row=1)

label=tk.Label(window, text="Restaurant ID:",width=10,height=1,bg="blue")
label.grid(column=1,row=2)
cid=tk.StringVar()
studid=tk.Entry(window,textvariable=cid)
studid.grid(column=2,row=2)
studid.configure(state=tk.DISABLED)

label=tk.Label(window, text="Address",width=15,height=1,bg="grey")
label.grid(column=1,row=3)
caddress=tk.StringVar()
studaddress=tk.Entry(window,textvariable=caddress)
studaddress.grid(column=2,row=3)

label=tk.Label(window, text="Borough",width=15,height=1,bg="grey")
label.grid(column=1,row=4)
cborough=tk.StringVar()
studborough=tk.Entry(window,textvariable=cborough)
studborough.grid(column=2,row=4)

label=tk.Label(window, text="Cuisine",width=15,height=1,bg="grey")
label.grid(column=1,row=5)
ccuisine=tk.StringVar()
studcuisine=tk.Entry(window, textvariable=ccuisine)
studcuisine.grid(column=2,row=5)

label=tk.Label(window, text="Grades",width=15,height=1,bg="grey")
label.grid(column=1,row=6)
cgrades=tk.StringVar()
studgrades=tk.Entry(window, textvariable=cgrades)
studgrades.grid(column=2,row=6)
```

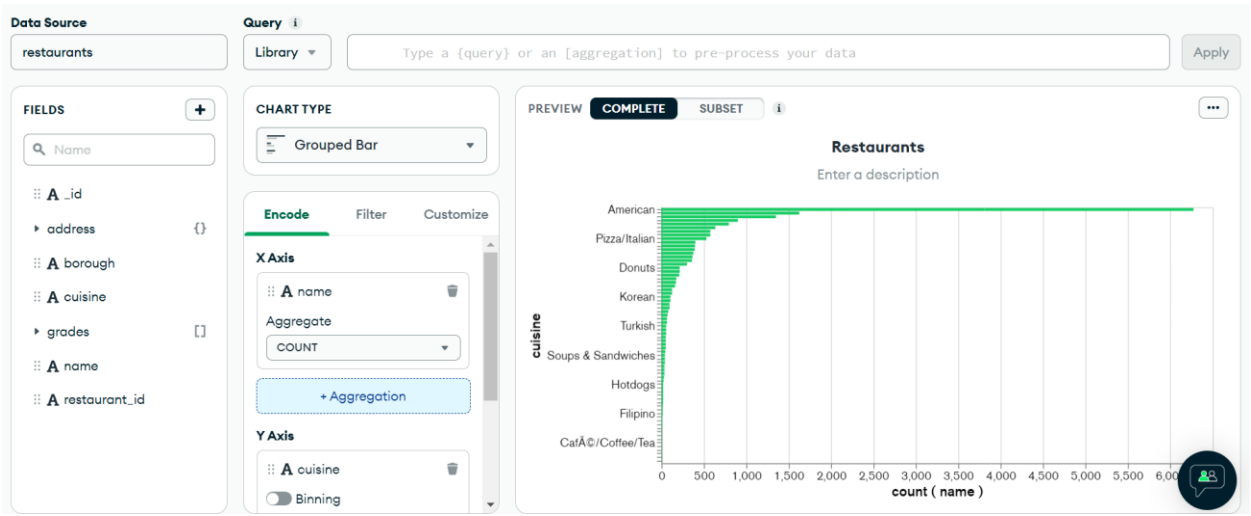
```
label=tk.Label(window, text="Name",width=15,height=1,bg="grey")
label.grid(column=1,row=7)
cname=tk.StringVar()
studname=tk.Entry(window, textvariable=cname)
studname.grid(column=2,row=7)


savebtn=tk.Button(text="Save",command=save)
savebtn.grid(column=1,row=8)
savebtn=tk.Button(text="Delete", command=delete)
savebtn.grid(column=2,row=8)
savebtn=tk.Button(text="Update", command=update)
savebtn.grid(column=3,row=8)

window.mainloop()
```




Restaurant Programe
Restaurants in town
Restaurant ID:
Address
Borough
Cuisine
Grades
Name
Object
Bronx
Bakery
Array
Morris Park Bake Shop
Save
Delete
Update


Restaurant Programe
Restaurants in town
Restaurant ID:
Address
Borough
Cuisine
Grades
Name
Object
Bronx
Bakery
Array
Morris Park Bake Shop
Save
Delete
Update
record
save record?
OK
Cancel


Project 0
Atlas
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
+ Create Database
Search Namespaces
Restaurants
restaurants
Restaurants.restaurants
STORAGE SIZE: 3.68MB
LOGICAL DATA SIZE: 9.5MB
TOTAL DOCUMENTS: 18860
INDEXES TOTAL SIZE: 760KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER
{ field: 'value' }
OPTIONS
Apply
Reset
borough: "Bronx"
cuisine: "Bakery"
grades: Array
name: "Morris Park Bake Shop"
restaurant_id: "30075445"
PREVIOUS
1-20 of many results
NEXT


Data Source
restaurants
Query
Library
Type a {query} or an [aggregation] to pre-process your data
Apply
FIELDS
Name
_id
address
borough
cuisine
grades
name
restaurant_id
CHART TYPE
Grouped Bar
Encode
Filter
Customize
X Axis
name
Aggregate
COUNT
+ Aggregation
Y Axis
cuisine
Binning
PREVIEW
COMPLETE
SUBSET
Restaurants
Enter a description
American
Pizza/Italian
Donuts
Korean
Turkish
Soups & Sandwiches
Hotdogs
Filipino
CafÃ©/Coffee/Tea
cuisine
0
500
1,000
1,500
2,000
2,500
3,000
3,500
4,000
4,500
5,000
5,500
count (name)