Assignment 07

Sajal Shrestha, Due Date: 04/03/2022,11:59pm

In [ ]:
```
# imports
import pandas as pd
import sqlite3
import pymongo
```

1. Write a python code to read the employees.csv file into a data-frame. Loop through each row in the data-frame, and insert/store the records into a table of sqlite3 database.

In [ ]:
```
df = pd.read_csv("Employees.csv")
df.head()
```

Out[ ]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services

In [ ]:
```
con = sqlite3.connect("employees.db")

cursor = con.cursor()

# Create table
cursor.execute(
    """
    CREATE TABLE IF NOT EXISTS employees (
        first_name CHAR(50) NOT NULL,
        gender CHAR(10) NOT NULL,
        start_date CHAR(20) NOT NULL,
        last_login_time CHAR(20) NOT NULL,
        salary INTEGER,
        bonus REAL,
        is_senior_management CHAR(5) NOT NULL,
        team CHAR(30)
    )
    """
)

# Loop through each row in the data-frame, and insert/store the records into a table of
for index, row in df.iterrows():
    cursor.execute(
        (
            "INSERT INTO employees VALUES("
            f"'{row['First Name']}', '{row['Gender']}',"
            f"'{row['Start Date']}', '{row['Last Login Time']}',"
            f"'{row['Salary']}', '{row['Bonus %']}',"
            f"'{row['Senior Management']}', '{row['Team']}'"
```

```
            ")"
        )
    )
    con.commit()

cursor.execute("SELECT count(*) FROM employees").fetchone()
```

Out[ ]:
```
(1000,)
```

2. In the table (from step-1), drop at-least 5 employees, update (some) information of at-least 5 employees, and then add at-least 5 new employees. Perform at-least one analysis task using group-by clause.

In [ ]:
```
con = sqlite3.connect("employees.db")
cursor = con.cursor()

# Drop at-least 5 employees
query = (
    "DELETE FROM employees where first_name in (NULL, 'nan') or gender in (NULL, 'nan')
)
cursor.execute(query)
con.commit()

print(cursor.rowcount, "employees removed")
```

```
236 employees removed
```

In [ ]:
```
# update some information of at-least 5 emplyees
cursor.execute("UPDATE employees set salary = salary + 10000 where is_senior_management
con.commit()
print(cursor.rowcount, "employees salary increased")
```

```
381 employees salary increased
```

In [ ]:
```
# add at-least 5 new employees
cursor.execute("INSERT INTO employees (first_name, gender, start_date, last_login_time,
    VALUES ('Sam', 'Male', '3/4/2005', '1:00 PM', 138705, 10.5, 'False', 'Legal' )")

cursor.execute("INSERT INTO employees (first_name, gender, start_date, last_login_time,
    VALUES ('Pamela', 'Female', '12/2/1993', '11:00 PM', 125000, 6.7, 'True', 'Finance'

cursor.execute("INSERT INTO employees (first_name, gender, start_date, last_login_time,
    VALUES ('Sita', 'Female', '6/9/2000', '12:00 AM', 110234, 11.2, 'False', 'Human Reso

cursor.execute("INSERT INTO employees (first_name, gender, start_date, last_login_time,
    VALUES ('John', 'Male', '5/12/2001', '1:45 PM', 95325, 8.9, 'False', 'Marketing' )")

cursor.execute("INSERT INTO employees (first_name, gender, start_date, last_login_time,
    VALUES ('Ram', 'Male', '4/14/1991', '5:30 PM', 140892, 9.8, 'True', 'Engineering' )"

con.commit()

cursor.execute("SELECT count(*) FROM employees").fetchone()
```

Out[ ]:
```
(769,)
```

In [ ]:
```
# Perform at-least one analysis task using group-by clause.
query = "SELECT team, count(first_name) count FROM employees GROUP BY team"

cursor.execute(query)

for item in cursor:
```

```
    team, count = item
    print("Team", team, "has", count, "employees")
```

```
Team Business Development has 88 employees
Team Client Services has 85 employees
Team Distribution has 60 employees
Team Engineering has 80 employees
Team Finance has 81 employees
Team Human Resources has 77 employees
Team Legal has 68 employees
Team Marketing has 75 employees
Team Product has 83 employees
Team Sales has 72 employees
```

3. Read the Employees-data of sqlite3 table (from step-2 above) into a data-frame. Loop through each row in the data-frame, and convert the rows into documents. Store each of these documents into a mongodb-collection.

In [ ]:
```
# Read the Employees-data of sqlite3 table (from step-2 above) into a data-frame.
cursor.execute("SELECT * from employees")
columns = ["first_name", "gender", "start_date", "last_login_time", "salary", "bonus", "
employees_df = pd.DataFrame(data=cursor.fetchall(), columns=columns)
employees_df.head()
```

Out[ ]:

	first_name	gender	start_date	last_login_time	salary	bonus	is_senior_management	team
0	Douglas	Male	8/6/1993	12:42 PM	107308	6.945	True	Marketing
1	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
2	Jerry	Male	3/4/2005	1:00 PM	148705	9.340	True	Finance
3	Larry	Male	1/24/1998	4:47 PM	111004	1.389	True	Client Services
4	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal

In [ ]:
```
# Loop through each row in the data-frame, and convert the rows into documents.
rows = []
for _, row in employees_df.iterrows():
    rows.append(row.to_dict())
```

In [ ]:
```
# Store each of these documents into a mongodb-collection.
client = pymongo.MongoClient("mongodb://127.0.0.1:27017/")

db = client["employeesDb"]
collection = db["employees"]
collection.drop()

collection.insert_many(rows)
```

Out[ ]:
```
<pymongo.results.InsertManyResult at 0x127597b40>
```

In [ ]:
```
collection.count_documents({})
```

Out[ ]:
```
769
```

4. In the collection (from step-3), drop at-least 5 employees, update (some) information of at- least 5 employees, and add at-least 5 new employees. Perform at-least one analysis task using group-by clause.

In [ ]:
```
# drop at-least 5 employees
filter = {"first_name": {"$regex": "^A"}}
res = collection.delete_many(filter)
res.deleted_count
```

Out[ ]:
```
67
```

In [ ]:
```
# update at-least 5 employees
filter = {"is_senior_management": "False"}
new_value = {"$inc": { "bonus": 5 }}

res = collection.update_many(filter, new_value)
res.modified_count
```

Out[ ]:
```
363
```

In [ ]:
```
# add at-least 5 new employees
new_employees = [
    {
        "first_name": "Sam",
        "gender": "Male",
        "start_date": "8/6/1993",
        "last_login_time": "12:42 PM",
        "salary": 107308,
        "bonus": 6.945,
        "is_senior_management": "True",
        "team": "Marketing",
    },
    {
        "first_name": "Ram",
        "gender": "Male",
        "start_date": "8/6/1991",
        "last_login_time": "12:00 AM",
        "salary": 11200,
        "bonus": 7.5,
        "is_senior_management": "True",
        "team": "Research",
    },
    {
        "first_name": "Gita",
        "gender": "Female",
        "start_date": "7/6/1996",
        "last_login_time": "1:42 PM",
        "salary": 93500,
        "bonus": 12.5,
        "is_senior_management": "False",
        "team": "Humar Resources",
    },
    {
        "first_name": "Micheal",
        "gender": "Male",
        "start_date": "8/6/1996",
        "last_login_time": "5:55 PM",
        "salary": 120000,
        "bonus": 6.5,
        "is_senior_management": "False",
        "team": "Legal",
    },
    {
        "first_name": "Jasmine",
        "gender": "Female",
        "start_date": "8/8/1990",
        "last_login_time": "5:42 PM",
```

```
        "salary": 140000,
        "bonus": 12.1,
        "is_senior_management": "True",
        "team": "Engineering",
    },
]

res = collection.insert_many(new_employees)
res.inserted_ids
```

Out[ ]:
```
[ObjectId('625761396d95681211b076a3'),
 ObjectId('625761396d95681211b076a4'),
 ObjectId('625761396d95681211b076a5'),
 ObjectId('625761396d95681211b076a6'),
 ObjectId('625761396d95681211b076a7')]
```

In [ ]:
```
# perform at-least one analysis task using group-by cloause
# https://pymongo.readthedocs.io/en/stable/examples/aggregation.html

pipeline = [{"$group": {"_id": "$gender", "salary": {"$avg": "$salary"}}}]

result = list(collection.aggregate(pipeline))

print(result)

for item in result:
    print("Gender", item["_id"], "has average salary of", item["salary"])
```

```
[{'_id': 'Male', 'salary': 96201.67806267807}, {'_id': 'Female', 'salary': 93908.0786516
854}]
Gender Male has average salary of 96201.67806267807
Gender Female has average salary of 93908.0786516854
```

5. Read the Employees-data of sqlite3 table (from step-2 above), and write it to a csv file (comma- delimited). Similarly, read each document of the mongodb-collection (from step-4 above), and write it to a text file in JSON (key-value) format.

In [ ]:
```
# Read the Employees-data of sqlite3 table (from step-2 above), and write it to a csv fi
# https://docs.python.org/3/library/csv.html

import csv

columns = ["first_name", "gender", "start_date", "last_login_time", "salary", "bonus", "

with open("employees_sqlite.csv", "w") as file:
    writer = csv.writer(file, delimiter=",")

    # write header
    writer.writerow(columns)

    # fetch data
    data = cursor.execute("SELECT * from employees")

    # write rows
    for item in data:
        writer.writerow(item)
```

In [ ]:
```
# Read each document of the mongodb-collection (from step-4 above), and write it to a te
import json

# remove _id from data
data = list(collection.find({}, {"_id": 0}))
```

```
with open("employees.json", "w") as file:
    json.dump(data,file, indent=4)
```