ACADEMIA DE STUDII ECONOMICE
MASTER INFORMATICA ECONOMICA

Managementul Bazelor de Date

Tema 2

PyMongo, Flask

Autor: Radu Alice-Mihaela

Grupa: 1130

Cerinta 1

Pe baza colecției *clienti_leasing* din MongoDB, încărcați într-un *df* numele, suma solicitată, suma din depozite și presoringul clienților cu vârsta > 45 de ani care au in depozite o suma mai mare de 35.000$. Verificați în *df* dacă suma din depozit este mai mare decât suma solicitată și pentru acești clienți modificați presoringul în 7.

```python
from pymongo import MongoClient
import pandas as pd

# Conectarea la MongoDB
client = MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_leasing']

# Definirea interogarii
query = {
    'VARSTA': {'$gt': 45},
    'SUMA_DEPOZIT': {'$gt': 35000}
}

# Definirea proiectiei
projection = {
    '_id': 0,
    'NUME_CLIENT': 1,
    'SUMA_SOLICITATA': 1,
    'SUMA_DEPOZIT': 1,
    'PRESCORING': 1
}

# Executarea interogarii
results = collection.find( *args: query, projection)

# Incarcarea datelor intr-un DataFrame Pandas
df = pd.DataFrame(list(results))
```

```python
# Incarcarea datelor intr-un DataFrame Pandas
df = pd.DataFrame(list(results))

# Verificarea structurii DataFrame-ului
print("Structura initiala a DataFrame-ului:")
print(df.head())

# Verificarea si modificarea presoringului
df.loc[df['SUMA_DEPOZIT'] > df['SUMA_SOLICITATA'], 'PRESCORING'] = 7

# Afisarea DataFrame-ului modificat
print("Structura DataFrame-ului dupa modificare:")
print(df)
```

```
Structura inițială a DataFrame-ului:
               NUME_CLIENT  PRESCORING  SUMA_DEPOZIT  SUMA_SOLICITATA
0            Herteg Editha           7      174894.4            25118
1  Horvat Alexandru Iulius           7       50045.2             8881
2            Husariu Greta           7     2555796.2           248944
3            Herascu Mihai           7       74717.4             4384
4            Harbu Viorica           7       67273.8             7588
Structura DataFrame-ului după modificare:
                 NUME_CLIENT  PRESCORING  SUMA_DEPOZIT  SUMA_SOLICITATA
0              Herteg Editha           7      174894.4            25118
1    Horvat Alexandru Iulius           7       50045.2             8881
2              Husariu Greta           7     2555796.2           248944
3              Herascu Mihai           7       74717.4             4384
4              Harbu Viorica           7       67273.8             7588
..                       ...         ...           ...              ...
308               Hoha Maria           7       41007.5             6985
309             Hornet Ionel           7       51480.2            11984
310         Harnagea Neculai           7       92444.3             9319
311           Herman Viorica           7       40823.3            15274
312       Hirlaoanu Anisoara           7       49766.3             5832

[313 rows x 4 columns]
```

Cerinta 2

Utilizand metoda find(), pe baza colecției clienti_daune din MongoDB, într-un df marca, modelul, anul de fabricație, componenta, prețul total și prețul manoperei pentru autoturismele din mărcile TOYOTA, FORD, HONDA. Calculați procentul manoperei din prețul total.

```python
import pandas as pd
from pymongo import MongoClient

# Conectarea la MongoDB
client = MongoClient(
    "mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_daune']

try:
    # Interogarea datelor din colectia clienti_daune pentru marcile TOYOTA, FORD și HONDA
    cursor = collection.find(
        *args: {"MARCA": {"$in": ["TOYOTA", "FORD", "HONDA"]}},
        {"MARCA": 1, "MODEL": 1, "AN_FABRICATIE": 1, "COMPONENTA": 1, "PRET_TOTAL": 1, "PRET_MANOPERA": 1, "_id": 0}
    )

    # Convertirea la DataFrame Pandas
    df = pd.DataFrame(list(cursor))

    # Setarea optiuni afisare randuri si coloane
    pd.set_option('display.max_columns', None)
    pd.set_option('display.max_rows', None)
    pd.set_option('display.width', 1000)
    pd.set_option('display.max_colwidth', None)
```

```python

    # Verificarea dacă DataFrame-ul nu este gol
    if not df.empty:
        # Calcularea procentului manoperei din pretul total
        df['PROCENT_MANOPERA'] = (df['PRET_MANOPERA'] / df['PRET_TOTAL']) * 100

        # Afisarea DataFrame-ului rezultat
        print(df)

        # Verificarea numarului de intrari pentru fiecare marca
        print("Numar de intrari pe marcă:")
        print(df['MARCA'].value_counts())
    else:
        print("Nu s-au gasit date pentru marcile specificate.")
except Exception as e:
    print(f"A aparut o eroare: {e}")

```

	MARCA	MODEL	AN_FABRICATIE	COMPONENTA	PRET_MANOPERA	PRET_TOTAL	PROCENT_MANOPERA
0	FORD	FOCUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	538.96	260.48	206.910319
1	FORD	WINDSTAR	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	321.00	246.23	130.365918
2	FORD	WINDSTAR	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	603.00	455.00	132.527473
3	FORD	FOCUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	431.73	321.97	134.090133
4	FORD	FOCUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	395.69	318.86	124.095214
5	FORD	MUSTANG	2010	NaN	538.83	404.25	133.291280
6	FORD	TAURUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	435.04	388.64	111.939070
7	FORD	WINDSTAR	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	449.43	237.42	189.297448
8	TOYOTA	CAMRY	2009	VEHICLE SPEED CONTROL	115.42	327.84	35.206198
9	TOYOTA	TACOMA	2009	SERVICE BRAKES, HYDRAULIC:ANTILOCK	78.85	327.84	24.051367
10	TOYOTA	CAMRY	2009	VEHICLE SPEED CONTROL	93.98	327.84	28.666423
11	FORD	RANGER	2009	VEHICLE SPEED CONTROL	135.52	340.95	39.747764
12	FORD	EXPLORER	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	444.87	316.87	140.395115
13	FORD	EXPLORER	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	458.68	430.21	106.617698
14	FORD	TAURUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	236.84	399.21	59.327171
15	FORD	FOCUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	482.75	205.68	234.709257
16	FORD	RANGER	2009	VEHICLE SPEED CONTROL	238.38	340.95	69.916410
17	FORD	WINDSTAR	2009	POWER TRAIN:AUTOMATIC TRANSMISSION	238.38	340.95	69.916410
18	TOYOTA	PRIUS	2009	ELECTRICAL SYSTEM	238.38	340.95	69.916410
19	TOYOTA	COROLLA	2009	ELECTRICAL SYSTEM	70.96	445.35	15.933535
20	TOYOTA	CAMRY	2009	VEHICLE SPEED CONTROL	73.66	340.95	21.604341
21	FORD	FREESTYLE	2009	ELECTRICAL SYSTEM	128.39	555.00	23.133333
22	FORD	ESCAPE	2009	VEHICLE SPEED CONTROL	189.27	528.35	35.822845
23	FORD	EXPLORER	2009	ELECTRICAL SYSTEM	282.21	508.96	55.448365
61	TOYOTA	PRIUS	2009	ELECTRICAL SYSTEM	182.54	406.49	44.906394
62	FORD	TAURUS	2009	POWER TRAIN:AUTOMATIC TRANSMISSION	99.03	340.95	29.045315
63	TOYOTA	CAMRY	2009	VEHICLE SPEED CONTROL	103.32	477.15	21.653568
64	FORD	RANGER	2009	SERVICE BRAKES, HYDRAULIC:ANTILOCK	112.36	445.35	25.229595
65	HONDA	CIVIC	2009	AIR BAGS	0.00	0.00	NaN
66	FORD	ESCAPE	2009	AIR BAGS	130.06	445.35	29.203997
67	FORD	EXPLORER	2009	AIR BAGS	137.29	595.48	23.055350
68	FORD	RANGER	2009	VEHICLE SPEED CONTROL	148.66	340.95	43.601701
69	FORD	WINDSTAR	2009	AIR BAGS	132.42	340.95	38.838539
70	TOYOTA	CAMRY	2008	POWER TRAIN:AUTOMATIC TRANSMISSION	117.12	327.84	35.724744
71	TOYOTA	CAMRY SOLARA	2009	ELECTRICAL SYSTEM	94.77	536.79	17.654949
72	TOYOTA	TUNDRA	2009	SERVICE BRAKES, HYDRAULIC:ANTILOCK	28.47	319.26	8.917497
73	FORD	WINDSTAR	2009	AIR BAGS	101.26	558.77	18.121946
74	FORD	MUSTANG GT	2009	ELECTRICAL SYSTEM	291.21	339.64	85.740784
75	HONDA	CIVIC	2010	AIR BAGS	113.67	319.26	35.604210
76	TOYOTA	TUNDRA	2009	VEHICLE SPEED CONTROL	67.56	445.35	15.170091
77	FORD	TAURUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	405.84	342.86	118.369014
78	FORD	TAURUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	393.67	477.67	82.414638
79	FORD	TAURUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	292.50	320.27	91.329191
80	FORD	TAURUS	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	253.23	392.43	64.528706
81	FORD	ESCAPE	2010	ELECTRICAL SYSTEM:ALTERNATOR/GENERATOR/REGULATOR	367.64	519.65	70.747619
82	FORD	EXPEDITION	2009	ELECTRICAL SYSTEM	111.14	327.84	33.900683
83	FORD	THUNDERBIRD	2010	SERVICE BRAKES, HYDRAULIC:FOUNDATION COMPONENTS:DISC:CALIPER	93.64	340.95	27.464438
84	TOYOTA	CAMRY	2009	VEHICLE SPEED CONTROL	96.29	340.95	28.241678
85	FORD	FORD	2010	POWER TRAIN:AUTOMATIC TRANSMISSION	96.10	340.95	28.185951
86	FORD	TAURUS	2009	AIR BAGS	198.66	327.84	60.596633

```
2509    FORD              TAURUS          2009                                                BATTERY      276.91      198.20      139.712412
2510    FORD              TAURUS          2009                                                BATTERY      159.57      131.13      121.688401
2511    FORD            EXPLORER          2009                                                BATTERY      257.49      243.34      105.814909
2512    FORD            EXPLORER          2009                                                BATTERY      306.99      154.60      198.570505
2513    FORD            EXPLORER          2009                                                BATTERY      176.70      250.91       70.423658
2514    FORD            EXPLORER          2009                                                BATTERY      257.70      115.53      223.058946
2515    FORD              TAURUS          2009                                                BATTERY      340.59      232.92      146.226172
2516    FORD              TAURUS          2009                                                BATTERY      235.30      216.79      108.538217
2517    FORD             MUSTANG          2009                                                BATTERY      279.17      102.36      272.733490
2518    FORD             MUSTANG          2009                                                BATTERY      183.10      174.80      104.748284
2519    FORD              TAURUS          2009                                                BATTERY      148.67      105.87      140.426939
2520    FORD              TAURUS          2009                                                BATTERY      142.81      226.26       63.117652
2521    FORD               FOCUS          2009                                                BATTERY      316.69      185.19      171.008154
2522    FORD            EXPLORER          2009                                                BATTERY      231.09      205.94      112.212295
2523    FORD            EXPLORER          2009                                                BATTERY      274.29      246.63      111.215181
2524    FORD             MUSTANG          2009                                                BATTERY      217.57      170.28      127.771905
2525    FORD             MUSTANG          2009                                                BATTERY      214.14      115.81      184.906312
Numar de intrari pe marcă:
MARCA
FORD      1942
TOYOTA     366
HONDA      218
Name: count, dtype: int64
```

Cerinta 3

Utilizand agregarile de tip pipeline, pe baza colecției clienti_leasing din MongoDB, încărcați într-un df starea civila, venitul anual, suma din depozite și suma solicitată pe fiecare profesie.

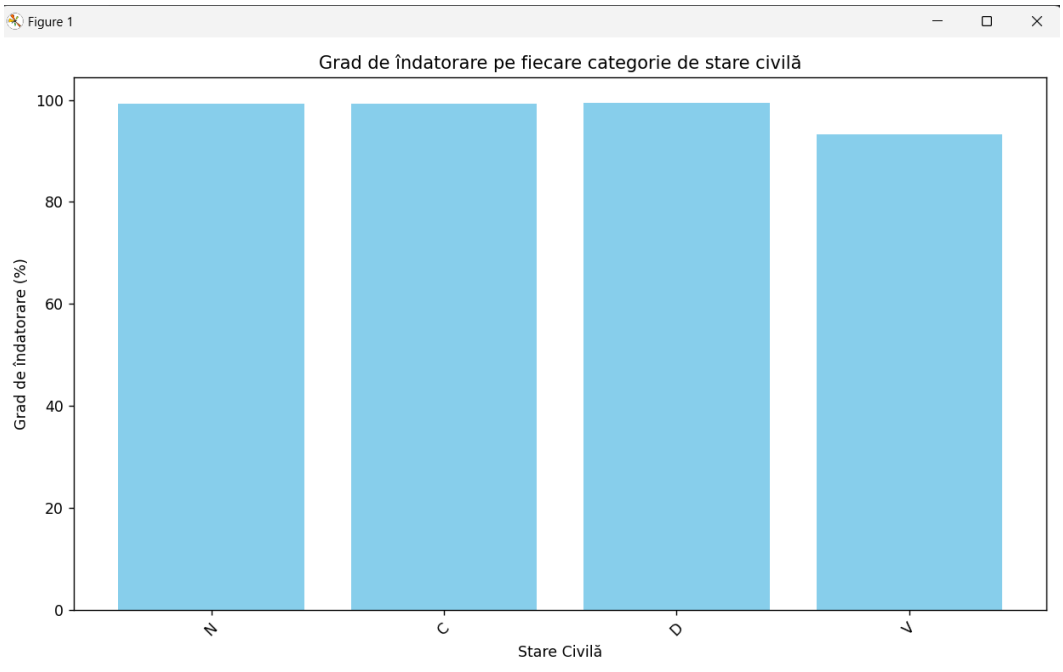
```python
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt

# Conectare la MongoDB
client = MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_leasing']

# Definirea pipeline-ului pentru agregare
pipeline = [
    {
        "$group": {
            "_id": "$PROFESIA",
            "stare_civila": {"$first": "$STARE_CIVILA"},
            "venit_anual": {"$sum": "$VENIT_ANUAL_RON"},
            "suma_depozite": {"$sum": "$SUMA_DEPOZIT"},
            "suma_solicitata": {"$sum": "$SUMA_SOLICITATA"}
        }
    },
    {
        "$addFields": {
            "profesia_clean": {
                "$function": {
                    "body": r"""
                        function(profesia) {
                            // Definim o functie pentru a elimina codul numeric
                            var match = /^([^\d]+)/.exec(profesia);
```

```
                            var match = /^([^\d]+)/.exec(profesia);
                            if (match) {
                                return match[1].trim();
                            }
                            return profesia.trim();
                        }
                    """,
                    "args": ["$_id"],
                    "lang": "js"
                }
            }
        }
    },
    {
        "$project": {
            "_id": 0,  # Exclude _id from final result
            "profesia": "$profesia_clean",
            "stare_civila": 1,
            "venit_anual": 1,
            "suma_depozite": 1,
            "suma_solicitata": 1
        }
    }
]
```

```
# Executarea agregării
cursor = collection.aggregate(pipeline)

# Convertirea rezultatelor într-un DataFrame Pandas
df = pd.DataFrame(list(cursor))

# Calculul gradului de îndatorare
df['grad_indatorare'] = (df['suma_solicitata'] / (df['venit_anual'] + df['suma_depozite'])) * 100

# Setarea opțiunilor pentru a afișa toate coloanele și rândurile
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', None)

# Afișarea întregului DataFrame
print(df)

# Reprezentarea grafică a gradului de îndatorare pe fiecare categorie de stare civilă
plt.figure(figsize=(10, 6))
plt.bar(df['stare_civila'], df['grad_indatorare'], color='skyblue')
plt.xlabel('Stare Civilă')
plt.ylabel('Grad de îndatorare (%)')
plt.title('Grad de îndatorare pe fiecare categorie de stare civilă')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

	stare_civila	venit_anual	suma_depozite	suma_solicitata	profesia	grad_indatorare
0	N	3.071913e+04	122.9	24054	CONDUCATORI AUTO	77.990975
1	C	6.000000e+03	0.0	5719	Muncitor necalificat in agricultura	95.316667
2	D	7.500000e+03	648.2	6761	Operator programare	82.975381
3	N	2.570090e+03	0.0	2553	Alti muncitori necalificati in servicii publice	99.335043
4	N	5.695920e+03	694.1	5055	Muncitor necalificat la intretinerea de drumuri, sosele,poduri, baraje	79.107734
5	D	1.500000e+04	951.3	12352	Masinist pentru utilaje specifice la extractie si executia	77.435695
6	D	1.000000e+04	114.8	9339	Instalator instalatii tehnico-sanitare si de gaze	92.330051
7	C	2.330000e+01	162.7	11	Operator chimist la fabricarea altor produse organice	5.913978
8	D	5.000000e+03	1.5	4106	Economist in industrie	82.095371
9	D	5.000000e+03	49.0	4862	Electrician pentru utilizarea energiei electrice	96.296296
10	N	8.580067e+04	27685.0	73244	Sef depozit	64.540307
11	C	2.850000e+03	217.5	2152	Instalator incalzire centrala si gaze	70.154849
12	C	2.900000e+04	1.1	26711	Lacatus mecanica fina	92.103403
13	D	2.267451e+07	23101764.9	16254949	Medic	35.509549
14	D	1.150000e+04	11504.7	11225	Director vanzari	48.794377
15	C	3.330990e+04	316.7	28614	Preot	85.093349
16	D	8.500000e+03	99.1	8021	Primitor distribuitor materiale si scule	93.277203
17	D	1.600000e+04	30.1	15570	Asistent maternal	97.129775
18	V	6.212064e+06	5580592.0	5122115	Pensionar	43.434787
19	N	5.381174e+04	39865.8	51753	Manipulant marfuri	55.245900
20	D	8.800000e+03	189.6	8280	ALTI FUNCTIONARI DE BIROU	92.106434
21	D	8.304050e+04	26424.8	69160	DIRECTORI GENERALI, DIRECTORI DIN UNITATI ECONOMICO-SOCIALE MARI SI ASIMILATI	63.179839
22	C	5.635997e+06	4400681.5	4751727	Agricultor	47.343618
23	C	1.650000e+04	2010.0	15940	Secretara	86.115613
24	N	7.079439e+04	501.5	44779	Analisti, programatori, informaticieni designeri	62.807267

101	C	1.003101e+04	0.0	6628	Inginer automatist	66.075101
102	D	5.366595e+04	13460.9	21899	ZUGRAVI, VOPSITORI, CURATITORI DE FATADE SI ASIMILATI	32.623309
103	D	9.900000e+03	452.0	9829	Fochist la caldari pentru incalziri centrale	94.947836
104	C	9.512170e+03	133.8	3811	Liftier	39.508727
105	C	2.466542e+04	214.1	23799	MONTATORI SI REPARATORI DE APARATE SI ECHIPAMENTE ELECTRONICE SI ELECTROTEHNICE	95.656990
106	N	4.672890e+03	45.7	4631	Secretar administrativ	98.143725
107	C	2.350000e+04	20075.0	22961	Administrator cont	52.693058
108	D	8.656900e+02	11.7	663	MESERIASI SI MUNCITORI CALIFICATI IN METALURGIE, CONSTRUCTII METALICE SI ASIMILATI	75.565028
109	D	3.145935e+04	843.2	19543	Invatatori in invatamantul primar	60.499868
110	D	7.009340e+03	13445.6	6903	Dulgher pentru constructii	33.747349
111	D	5.384392e+04	80.6	52047	INVATATORI	96.518244
112	D	1.500000e+04	304.9	3690	Medic veterinar	24.109926
113	C	7.961993e+04	573.7	65276	Contabil	81.397986
114	C	1.600000e+03	90.3	1279	Ospatar (chelner)	75.667041
115	C	1.120000e+04	239.7	10714	Functionar economic	93.656302
116	C	2.300000e+03	620.5	2221	Barman	76.048622
117	V	8.750000e+03	20.8	8178	Referent resurse umane	93.241209
118	C	5.373830e+03	42.5	5352	Agent paza in incinte (hoteluri, magazine, etc)	98.812295
119	D	1.729485e+04	166.0	9014	Pompagiu	51.624062
120	D	5.000000e+03	25.5	4314	INGINERI IN INDUSTRIA TEXTILA-PIELARIE SI INDUSTRIA ALIMENTARA	85.842205
121	C	1.146885e+08	57763202.2	83393891	Inginer	48.357829
122	N	4.542750e+03	24.7	4405	Zugravi, tapetari, lacuitori si vopsitori	96.443311
123	C	1.032110e+05	13174.3	84056	Lucrator comercial	72.222179
124	N	5.000000e+03	0.0	4656	Paznic	93.120000
125	D	8.000000e+03	0.0	6852	Galvanizator	85.650000



Cerinta 4

Utilizand agregarile de tip *pipeline*, pe baza colecției *clienti_daune* din MongoDB, încărcați într-un *df* marca, modelul, valoarea totală și numărul de daune pe fiecare model și marcă. Afișați numărul de autoturisme pentru care valoarea totală depășește 50.000$. Reprezentați grafic modelele care au înregistrat mai mult de 150 de daune.

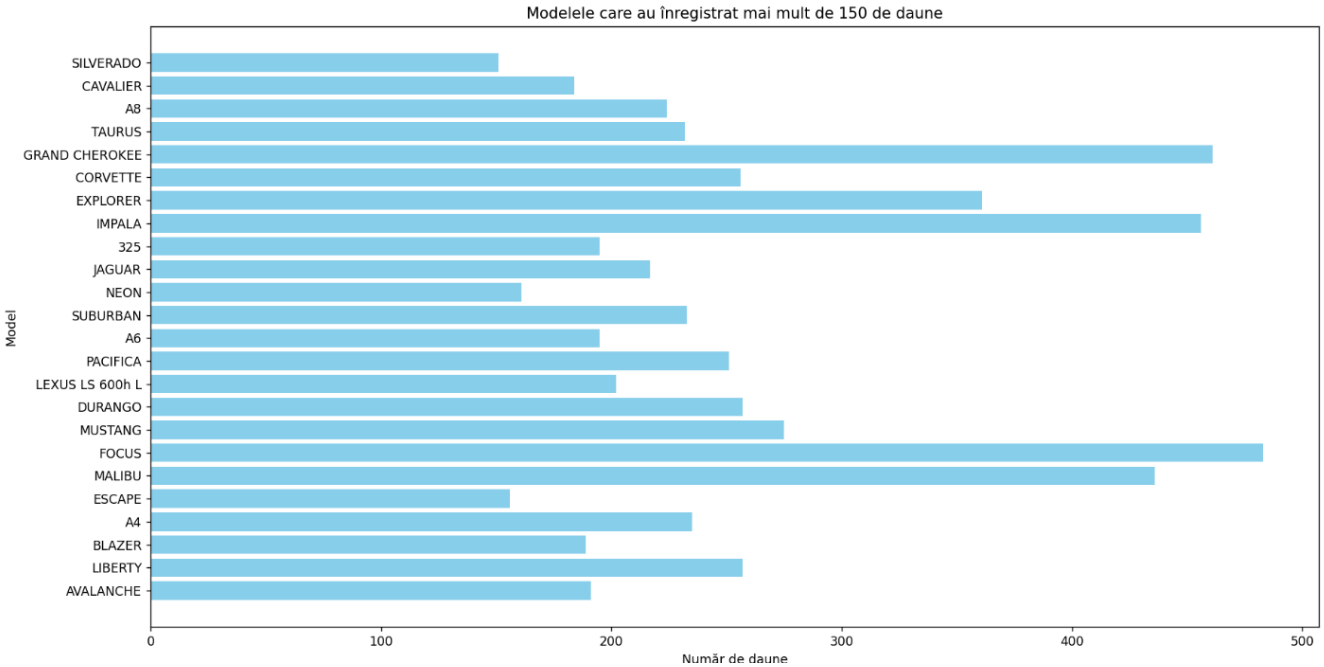
```
import pandas as pd
import matplotlib.pyplot as plt

# Conectare la MongoDB
client = MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_daune']

# Definirea pipeline-ului pentru agregare
pipeline = [
    {
        "$group": {
            "_id": {
                "marca": "$MARCA",
                "model": "$MODEL"
            },
            "valoare_totala": {"$sum": "$VALOARE_DAUNA"},
            "numar_daune": {"$sum": 1}
        }
    },
    {
        "$project": {
            "_id": 0,
            "marca": "$_id.marca",
            "model": "$_id.model",
            "valoare_totala": 1,
            "numar_daune": 1
        }
    }
]
```

```
# Executarea agregarii
cursor = collection.aggregate(pipeline)

# Convertirea rezultatelor intr-un DataFrame Pandas
df = pd.DataFrame(list(cursor))

# Asigurarea că coloana 'model' este tratată ca sir de caractere
df['model'] = df['model'].astype(str)

# Filtrarea autoturismelor pentru care valoarea totala a daunelor depaseste 50.000$
autoturisme_valoare_mare = df[df['valoare_totala'] > 50000]
numar_autoturisme_valoare_mare = autoturisme_valoare_mare.shape[0]

# Afisarea numarului de autoturisme
print(f"Numarul de autoturisme pentru care valoarea totala a daunelor depaseste 50.000$: {numar_autoturisme_valoare_mare}")

# Afisarea DataFrame-ului cu autoturismele care au valoarea totala a daunelor mai mare de 50.000$
print("Autoturismele cu valoarea totala a daunelor mai mare de 50.000$: \n", autoturisme_valoare_mare)

# Filtrarea si reprezentarea grafica a modelelor care au inregistrat mai mult de 150 de daune
modele_daune_multe = df[df['numar_daune'] > 150]

# Afisarea DataFrame-ului cu modelele care au inregistrat mai mult de 150 de daune
print("Modelele care au inregistrat mai mult de 150 de daune: \n", modele_daune_multe)
```

```
57
58  # Reprezentarea grafica
59  plt.figure(figsize=(12, 8))
60  plt.barh(modele_daune_multe['model'], modele_daune_multe['numar_daune'], color='skyblue')
61  plt.xlabel('Număr de daune')
62  plt.ylabel('Model')
63  plt.title('Modelele care au inregistrat mai mult de 150 de daune')
64  plt.tight_layout()
65  plt.show()
66
67  # Salvare DataFrame in fisier CSV pentru analiza ulterioara
68  df.to_csv( path_or_buf: 'rezultate_daune.csv', index=False)
69
```

```
Autoturismele cu valoarea totală a daunelor mai mare de 50.000$:
      valoare_totala  numar_daune      marca            model
17         74043.92          191  CHEVROLET        AVALANCHE
51        111987.17          257       JEEP          LIBERTY
57         71270.49          189  CHEVROLET           BLAZER
65         97012.82          235       AUDI               A4
113        69036.99          156       FORD           ESCAPE
114       147364.91          436  CHEVROLET           MALIBU
129       175275.01          483       FORD            FOCUS
136       112991.93          275       FORD          MUSTANG
148       115180.82          257      DODGE          DURANGO
172        83467.01          202      LEXUS  LEXUS LS 600h L
203       108425.16          251   CHRYSLER         PACIFICA
217        82780.98          195       AUDI               A6
218        88494.15          233  CHEVROLET         SUBURBAN
297        67521.49          161      DODGE             NEON
311        60003.10          134       FORD         WINDSTAR
324        73557.98          136     TOYOTA            CAMRY
327        88458.32          217     JAGUAR           JAGUAR
365        79025.17          195        BMW              325
367       154508.43          456  CHEVROLET           IMPALA
371       149815.20          361       FORD         EXPLORER
375       104398.85          256  CHEVROLET         CORVETTE
382       168762.62          461       JEEP   GRAND CHEROKEE
390        93698.28          232       FORD           TAURUS
415        52212.96          116  CHEVROLET      TRAILBLAZER
416        93102.87          224       AUDI               A8
496        68505.97          184  CHEVROLET         CAVALIER
```

```
496        68505.97          184  CHEVROLET         CAVALIER
519        67227.97          151  CHEVROLET        SILVERADO
Modelele care au înregistrat mai mult de 150 de daune:
      valoare_totala  numar_daune      marca            model
17         74043.92          191  CHEVROLET        AVALANCHE
51        111987.17          257       JEEP          LIBERTY
57         71270.49          189  CHEVROLET           BLAZER
65         97012.82          235       AUDI               A4
113        69036.99          156       FORD           ESCAPE
114       147364.91          436  CHEVROLET           MALIBU
129       175275.01          483       FORD            FOCUS
136       112991.93          275       FORD          MUSTANG
148       115180.82          257      DODGE          DURANGO
172        83467.01          202      LEXUS  LEXUS LS 600h L
203       108425.16          251   CHRYSLER         PACIFICA
217        82780.98          195       AUDI               A6
218        88494.15          233  CHEVROLET         SUBURBAN
297        67521.49          161      DODGE             NEON
327        88458.32          217     JAGUAR           JAGUAR
365        79025.17          195        BMW              325
367       154508.43          456  CHEVROLET           IMPALA
371       149815.20          361       FORD         EXPLORER
375       104398.85          256  CHEVROLET         CORVETTE
382       168762.62          461       JEEP   GRAND CHEROKEE
390        93698.28          232       FORD           TAURUS
416        93102.87          224       AUDI               A8
496        68505.97          184  CHEVROLET         CAVALIER
519        67227.97          151  CHEVROLET        SILVERADO
```



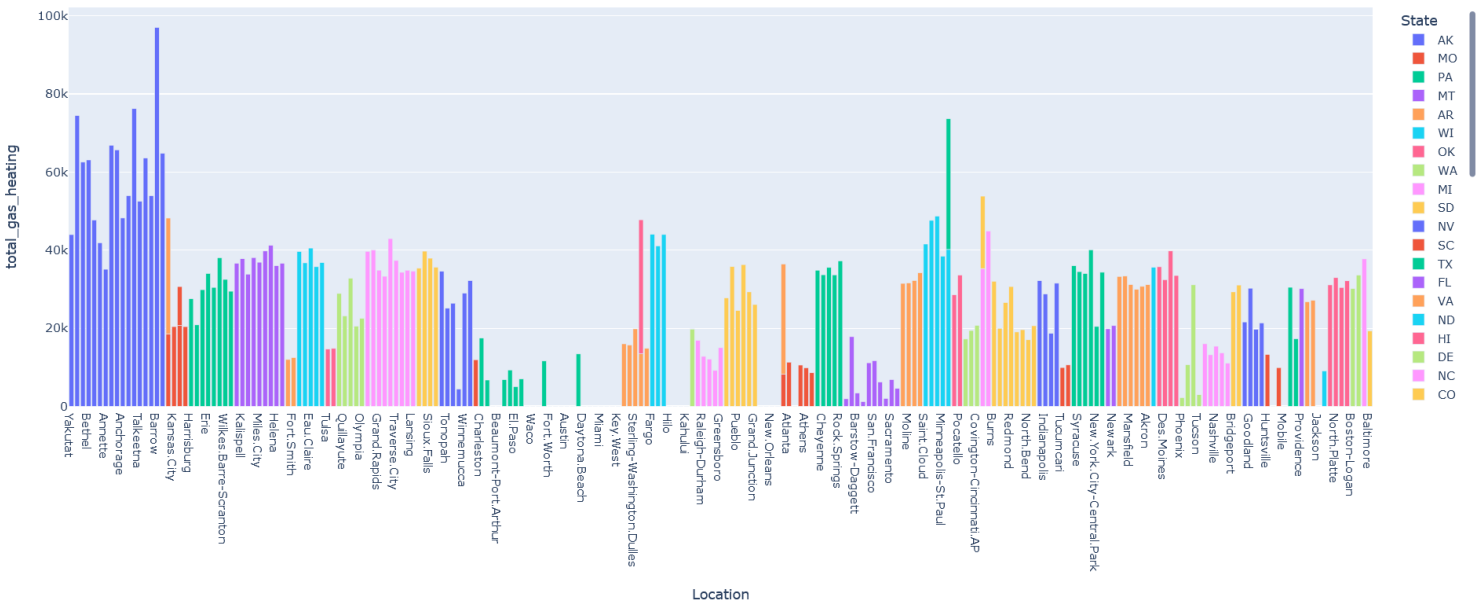
Cerinta 5

Utilizand agregarile de tip *pipeline,* pe baza colecției *USA_TMY* din MongoDB, încărcați într-un *df* consumul total de gaz utilizat pentru incalzire (*GasHeating*) si de electriciate (*ElectricHeating*) pe locatii (Location) si state (State). Utilizand pachetul *plotly* realizati 2 grafice prin care sa reprezentati consumurile.

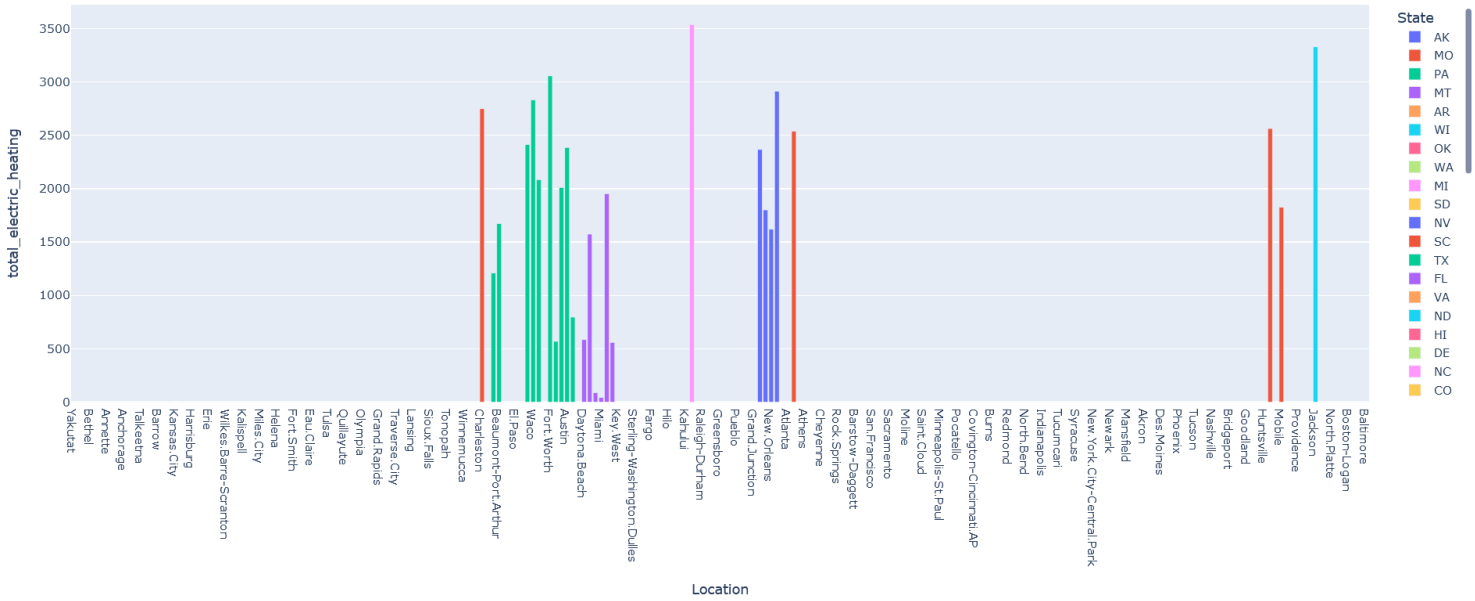
```python
from pymongo import MongoClient
import pandas as pd
import plotly.express as px

# Conectare la MongoDB
client = MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing&authMechanism=SCRAM-SHA-256"
db = client['daune_leasing']
collection = db['USA_TMY']

# Definirea pipeline-ului pentru agregare
pipeline = [
    {
        "$group": {
            "_id": {
                "Location": "$Location",
                "State": "$State"
            },
            "total_gas_heating": {"$sum": "$GasHeating"},
            "total_electric_heating": {"$sum": "$ElectricHeating"}
        }
    },
    {
        "$project": {
            "_id": 0,
            "Location": "$_id.Location",
            "State": "$_id.State",
            "total_gas_heating": 1,
            "total_electric_heating": 1
```

```python
            "total_electric_heating": 1
        }
    }
]

# Executarea agregarii
cursor = collection.aggregate(pipeline)

# Convertirea rezultatelor intr-un DataFrame Pandas
df = pd.DataFrame(list(cursor))

# Afisarea DataFrame-ului
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', None)
print(df)

# Graficul consumului de gaz pentru incalzire pe locatii si state
fig_gas = px.bar(df, x='Location', y='total_gas_heating', color='State', title='Consumul total de gaz pentru incalzire pe locatii si state')
fig_gas.show()

# Graficul consumului de electricitate pentru incalzire pe locatii si state
fig_electric = px.bar(df, x='Location', y='total_electric_heating', color='State', title='Consumul total de electricitate pentru incalzire pe locatii
fig_electric.show()
```

Consumul total de gaz pentru incalzire pe locatii si state



Consumul total de electricitate pentru incalzire pe locatii si state

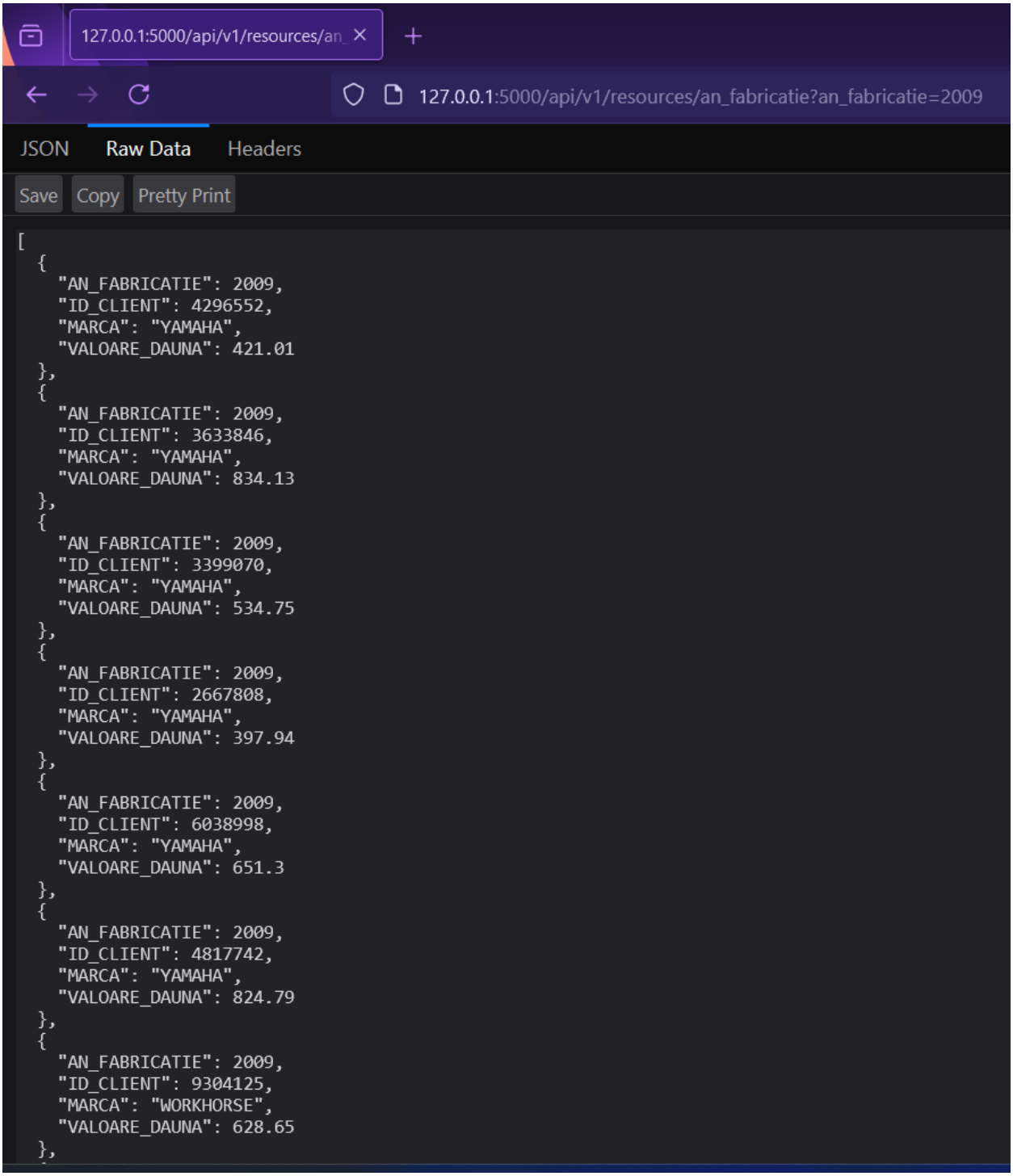


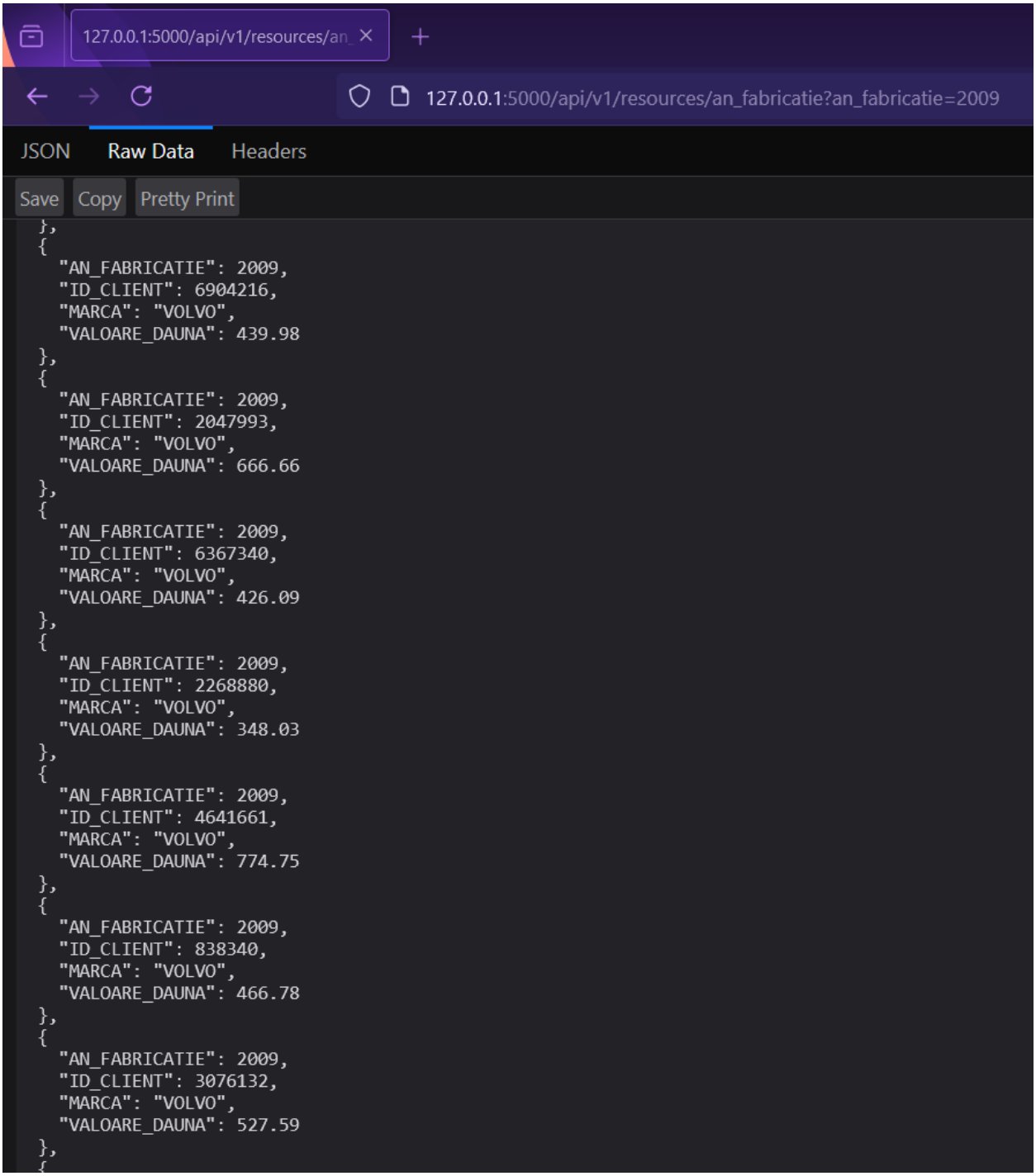
Cerinta 6

Pornind de la "Exemplu REST API - integrarea dintre Oracle și MongoDB" din "S5 - Python_MongoDB_Flask" definiti o functie care sa mapeze la calea /api/v1/resources/an_fabricatie o functie Python care sa returneze din colectia MongoDB daunele produse la autoturismele fabricate in anul trimis ca parametru.

```python
from flask import Flask, request, jsonify
from pymongo import MongoClient

app = Flask(__name__)

@app.route( rule: '/api/v1/resources/an_fabricatie', methods=['GET'])
def get_daune_by_an_fabricatie():
    # Preluarea parametrului an_fabricatie din cerere
    an_fabricatie = request.args.get('an_fabricatie')
    if not an_fabricatie:
        return jsonify({"error": "Missing parameter: an_fabricatie"}), 400

    # Verificarea ca an_fabricatie este un numar intreg
    try:
        an_fabricatie = int(an_fabricatie)
    except ValueError:
        return jsonify({"error": "Invalid parameter: an_fabricatie must be an integer"}), 400

    # Conectarea la MongoDB
    client = MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing&authMechanism=SCRAM-SHA-256")
    db = client['daune_leasing']
    collection = db['clienti_daune']
```

```python

    # Definirea proiectiei si sortarii
    projection = {
        "_id": 0,
        "AN_FABRICATIE": 1,
        "MARCA": 1,
        "VALOARE_DAUNA": 1,
        "ID_CLIENT": 1
    }
    sort = [("MARCA", -1)]

    # Interogarea bazei de date
    cursor = collection.find( *args: {"AN_FABRICATIE": an_fabricatie}, projection).sort(sort)
    daune_list = list(cursor)

    return jsonify(daune_list), 200

if __name__ == '__main__':
    app.run(debug=True)

```


127.0.0.1:5000/api/v1/resources/an_
127.0.0.1:5000/api/v1/resources/an_fabricatie?an_fabricatie=2009
JSON
Raw Data
Headers
Save
Copy
Pretty Print
[
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 4296552,
"MARCA": "YAMAHA",
"VALOARE_DAUNA": 421.01
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 3633846,
"MARCA": "YAMAHA",
"VALOARE_DAUNA": 834.13
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 3399070,
"MARCA": "YAMAHA",
"VALOARE_DAUNA": 534.75
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 2667808,
"MARCA": "YAMAHA",
"VALOARE_DAUNA": 397.94
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 6038998,
"MARCA": "YAMAHA",
"VALOARE_DAUNA": 651.3
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 4817742,
"MARCA": "YAMAHA",
"VALOARE_DAUNA": 824.79
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 9304125,
"MARCA": "WORKHORSE",
"VALOARE_DAUNA": 628.65
},


127.0.0.1:5000/api/v1/resources/an_
127.0.0.1:5000/api/v1/resources/an_fabricatie?an_fabricatie=2009
JSON
Raw Data
Headers
Save
Copy
Pretty Print
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 6904216,
"MARCA": "VOLVO",
"VALOARE_DAUNA": 439.98
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 2047993,
"MARCA": "VOLVO",
"VALOARE_DAUNA": 666.66
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 6367340,
"MARCA": "VOLVO",
"VALOARE_DAUNA": 426.09
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 2268880,
"MARCA": "VOLVO",
"VALOARE_DAUNA": 348.03
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 4641661,
"MARCA": "VOLVO",
"VALOARE_DAUNA": 774.75
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 838340,
"MARCA": "VOLVO",
"VALOARE_DAUNA": 466.78
},
{
"AN_FABRICATIE": 2009,
"ID_CLIENT": 3076132,
"MARCA": "VOLVO",
"VALOARE_DAUNA": 527.59
},

Script

--Cerinta 1

```
from pymongo import MongoClient
import pandas as pd

# Conectarea la MongoDB
client =
MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing
&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_leasing']

# Definirea interogarii
query = {
   'VARSTA': {'$gt': 45},
   'SUMA_DEPOZIT': {'$gt': 35000}
}

# Definirea proiectiei
projection = {
   '_id': 0,
   'NUME_CLIENT': 1,
   'SUMA_SOLICITATA': 1,
   'SUMA_DEPOZIT': 1,
   'PRESCORING': 1
}

# Executarea interogarii
results = collection.find(query, projection)

# Incarcarea datelor intr-un DataFrame Pandas
df = pd.DataFrame(list(results))
pd.set_option('display.max_rows', None)

# Verificarea structurii DataFrame-ului
print("Structura initiala a DataFrame-ului:")
print(df.head())

# Verificarea si modificarea presoringului
df.loc[df['SUMA_DEPOZIT'] > df['SUMA_SOLICITATA'], 'PRESCORING'] = 7
# Afisarea DataFrame-ului modificat
print("Structura DataFrame-ului dupa modificare:")
```

```
print(df)
```

--Cerinta 2

```
import pandas as pd
from pymongo import MongoClient

# Conectarea la MongoDB
client = MongoClient(

"mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing&authMecha
nism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_daune']

try:
   # Interogarea datelor din colecția clienti_daune pentru marcile TOYOTA, FORD și HONDA
   cursor = collection.find(
      {"MARCA": {"$in": ["TOYOTA", "FORD", "HONDA"]}},
      {"MARCA": 1, "MODEL": 1, "AN_FABRICATIE": 1, "COMPONENTA": 1, "PRET_TOTAL": 1,
"PRET_MANOPERA": 1, "_id": 0}
   )

   # Convertirea la DataFrame Pandas
   df = pd.DataFrame(list(cursor))

   # Setarea opțiuni afisare randuri si coloane
   pd.set_option('display.max_columns', None)
   pd.set_option('display.max_rows', None)
   pd.set_option('display.width', 1000)
   pd.set_option('display.max_colwidth', None)

   # Verificarea dacă DataFrame-ul nu este gol
   if not df.empty:
      # Calcularea procentului manoperei din prețul total
      df['PROCENT_MANOPERA'] = (df['PRET_MANOPERA'] / df['PRET_TOTAL']) * 100

      # Afișarea DataFrame-ului rezultat
      print(df)

      # Verificarea numarului de intrari pentru fiecare marca
      print("Numar de intrari pe marcă:")
      print(df['MARCA'].value_counts())
   else:
      print("Nu s-au gasit date pentru marcile specificate.")
```

```
except Exception as e:
    print(f"A aparut o eroare: {e}")
```

--Cerinta 3

```
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt

# Conectare la MongoDB
client =
MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing
&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_leasing']

# Definirea pipeline-ului pentru agregare
pipeline = [
    {
        "$group": {
            "_id": "$PROFESIA",
            "stare_civila": {"$first": "$STARE_CIVILA"},
            "venit_anual": {"$sum": "$VENIT_ANUAL_RON"},
            "suma_depozite": {"$sum": "$SUMA_DEPOZIT"},
            "suma_solicitata": {"$sum": "$SUMA_SOLICITATA"}
        }
    },
    {
        "$addFields": {
            "profesia_clean": {
                "$function": {
                    "body": r"""
                        function(profesia) {
                            // Definim o functie pentru a elimina codul numeric
                            var match = /^([^\d]+)/.exec(profesia);
                            if (match) {
                                return match[1].trim();
                            }
                            return profesia.trim();
                        }
                    """,
                    "args": ["$_id"],
                    "lang": "js"
                }
            }
```

```
      }
    },
    {
      "$project": {
        "_id": 0,  # Exclude _id from final result
        "profesia": "$profesia_clean",
        "stare_civila": 1,
        "venit_anual": 1,
        "suma_depozite": 1,
        "suma_solicitata": 1
      }
    }
]

# Executarea agregarii
cursor = collection.aggregate(pipeline)

# Convertirea rezultatelor intr-un DataFrame Pandas
df = pd.DataFrame(list(cursor))

# Calculul gradului de indatorare
df['grad_indatorare'] = (df['suma_solicitata'] / (df['venit_anual'] + df['suma_depozite'])) * 100

# Setarea optiunilor pentru a afisa toate coloanele si randurile
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', None)

# Afisarea intregului DataFrame
print(df)

# Reprezentarea grafica a gradului de indatorare pe fiecare categorie de stare civila
plt.figure(figsize=(10, 6))
plt.bar(df['stare_civila'], df['grad_indatorare'], color='skyblue')
plt.xlabel('Stare Civilă')
plt.ylabel('Grad de îndatorare (%)')
plt.title('Grad de îndatorare pe fiecare categorie de stare civilă')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Salvare DataFrame intr-un fisier CSV pentru analiza ulterioara
df.to_csv('rezultate_agregare.csv', index=False)
```

--Cerinta 4

```
from pymongo import MongoClient
import pandas as pd
import matplotlib.pyplot as plt

# Conectare la MongoDB
client =
MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing
&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['clienti_daune']

# Definirea pipeline-ului pentru agregare
pipeline = [
  {
    "$group": {
      "_id": {
        "marca": "$MARCA",
        "model": "$MODEL"
      },
      "valoare_totala": {"$sum": "$VALOARE_DAUNA"},
      "numar_daune": {"$sum": 1}
    }
  },
  {
    "$project": {
      "_id": 0,
      "marca": "$_id.marca",
      "model": "$_id.model",
      "valoare_totala": 1,
      "numar_daune": 1
    }
  }
]

# Executarea agregarii
cursor = collection.aggregate(pipeline)

# Convertirea rezultatelor intr-un DataFrame Pandas
df = pd.DataFrame(list(cursor))

# Asigurarea că coloana 'model' este tratată ca sir de caractere
df['model'] = df['model'].astype(str)
```

```
# Filtrarea autoturismelor pentru care valoarea totala a daunelor depaseste 50.000$
autoturisme_valoare_mare = df[df['valoare_totala'] > 50000]
numar_autoturisme_valoare_mare = autoturisme_valoare_mare.shape[0]

# Afișarea numarului de autoturisme
print(f"Numarul de autoturisme pentru care valoarea totala a daunelor depaseste 50.000$:
{numar_autoturisme_valoare_mare}")

# Afișarea DataFrame-ului cu autoturismele care au valoarea totala a daunelor mai mare de
50.000$
print("Autoturismele cu valoarea totala a daunelor mai mare de 50.000$: \n",
autoturisme_valoare_mare)

# Filtrarea si reprezentarea grafica a modelelor care au inregistrat mai mult de 150 de daune
modele_daune_multe = df[df['numar_daune'] > 150]

# Afișarea DataFrame-ului cu modelele care au inregistrat mai mult de 150 de daune
print("Modelele care au inregistrat mai mult de 150 de daune: \n", modele_daune_multe)

# Reprezentarea grafica
plt.figure(figsize=(12, 8))
plt.barh(modele_daune_multe['model'], modele_daune_multe['numar_daune'], color='skyblue')
plt.xlabel('Număr de daune')
plt.ylabel('Model')
plt.title('Modelele care au inregistrat mai mult de 150 de daune')
plt.tight_layout()
plt.show()

# Salvare DataFrame in fisier CSV pentru analiza ulterioara
#df.to_csv('rezultate_daune.csv', index=False)
```

--Cerinta 5

```
from pymongo import MongoClient
import pandas as pd
import plotly.express as px

# Conectare la MongoDB
client =
MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing
&authMechanism=SCRAM-SHA-256")
db = client['daune_leasing']
collection = db['USA_TMY']
```

```
# Definirea pipeline-ului pentru agregare
pipeline = [
  {
    "$group": {
      "_id": {
        "Location": "$Location",
        "State": "$State"
      },
      "total_gas_heating": {"$sum": "$GasHeating"},
      "total_electric_heating": {"$sum": "$ElectricHeating"}
    }
  },
  {
    "$project": {
      "_id": 0,
      "Location": "$_id.Location",
      "State": "$_id.State",
      "total_gas_heating": 1,
      "total_electric_heating": 1
    }
  }
]

# Executarea agregarii
cursor = collection.aggregate(pipeline)

# Convertirea rezultatelor intr-un DataFrame Pandas
df = pd.DataFrame(list(cursor))

# Afisarea DataFrame-ului
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', None)
print(df)

# Graficul consumului de gaz pentru incalzire pe locatii si state
fig_gas = px.bar(df, x='Location', y='total_gas_heating', color='State', title='Consumul total de gaz pentru incalzire pe locatii si state')
fig_gas.show()

# Graficul consumului de electricitate pentru incalzire pe locatii si state
fig_electric = px.bar(df, x='Location', y='total_electric_heating', color='State', title='Consumul total de electricitate pentru incalzire pe locatii si state')
```

```
fig_electric.show()
```

--Cerinta 6

```
from flask import Flask, request, jsonify
from pymongo import MongoClient

app = Flask(__name__)

@app.route('/api/v1/resources/an_fabricatie', methods=['GET'])
def get_daune_by_an_fabricatie():
    # Preluarea parametrului an_fabricatie din cerere
    an_fabricatie = request.args.get('an_fabricatie')
    if not an_fabricatie:
        return jsonify({"error": "Missing parameter: an_fabricatie"}), 400

    # Verificarea ca an_fabricatie este un numar intreg
    try:
        an_fabricatie = int(an_fabricatie)
    except ValueError:
        return jsonify({"error": "Invalid parameter: an_fabricatie must be an integer"}), 400

    # Conectarea la MongoDB
    client =
MongoClient("mongodb://master:stud1234@193.226.34.57:27017/?authSource=daune_leasing
&authMechanism=SCRAM-SHA-256")
    db = client['daune_leasing']
    collection = db['clienti_daune']

    # Definirea proiectiei si sortarii
    projection = {
        "_id": 0,
        "AN_FABRICATIE": 1,
        "MARCA": 1,
        "VALOARE_DAUNA": 1,
        "ID_CLIENT": 1
    }
    sort = [("MARCA", -1)]

    # Interogarea bazei de date
    cursor = collection.find({"AN_FABRICATIE": an_fabricatie}, projection).sort(sort)
    daune_list = list(cursor)

    return jsonify(daune_list), 200
```

```
if __name__ == '__main__':
    app.run(debug=True)
```