Cuprins

Programare SAS *2*

Crearea unui set de date SAS din fișiere externe **2**

Crearea și folosirea de formate definite de utilizator **2**

Procesarea iterativă și condițională a datelor **3**

Crearea de subseturi de date **4**

Utilizarea de funcții SAS **5**

Combinarea seturilor de date prin proceduri specifice SAS si SQL **6**

Raportare, Analiză statistică, Vizualizare grafică **7**

Programare Python *9*

Steamlit **9**

Utilizarea pachetului GeoPandas **10**

Tratarea valorilor lipsă **10**

Metode de codificare a datelor **11**

Metode de scalare **11**

Predicții statistice, gruparea și agregarea datelor în pachetul pandas **12**

Utilizarea funcțiilor de grup **12**

Clusterizare prin analiza factoriala **13**

Corelograma **14**

Regresie multipla **14**

Programare SAS

Crearea unui set de date SAS din fișiere externe

Datele SGI

Obs	SGI 2022 Values	Sustainable Policies	Inflation	Gross Fixed Capital Formation	Real Interest Rates	Potential Output Growth Rate	Real GDP Growth Rate	Unemployment	Long-term Unemployment	Youth Unemployment	Low-skilled Unemployment	Employment	Low Pay Incidence	Employment Rates by Gender	Involuntary Part-time Employmen	Tax System Complexity	Structural Balance	Statutory Corporate Tax Rate	Redistribution Effect	Statutory Maximum Personal Inco	Debt to GDP	Primary Balance	Gross Interest Payments by Gene	Budget Consolidation	Debt per Child	External Debt to GDP	Public R&D Spending	Private R&D Spending
1	Australia	5.7307037526	2.82	22.94	-3.87	4.01	4.69	5.26	1.03	11.27	7.58	74.97	15.46	0.9	29.53	85.67	-7.68	30	28.41	47	59.81	-6.66	0.49	1	174.38	0.19	0.78	1.47
2	Austria	6.3375536205	2.75	27.57	-1.83	5.64	4.48	6.3	1.95	11	13.6	72.4	14.74	0.89	9.2	83.5	-4.94	25	43.62	50	83.06	-4.98	1.11	1	345.21	0.54	1.06	2.14
3	Belgium	6.0168262273	3.22	26.32	-4.23	8.49	6.28	6.3	2.85	18.2	12.9	65.3	11.5	0.9	21.4	78.38	-5.8	25	46.42	53.07	108.25	-4.67	1.66	1	369.2	0.64	0.56	2.6
4	Bulgaria	5.2691130093	2.84	19.56	-5.68	12.08	4.18	5.3	2.6	15.8	15.6	68.1	20.42	0.89	49	72.32	-2.24	10	23.14	10	23.83	-2.87	0.5	1	42.5	0.12	0.2	0.63
5	Canada	6.5282213352	3.4	23.69	-6.3	11.01	4.56	7.44	1.22	13.5	12.31	73.23	20.68	0.92	23.72	88.05	-3.62	26.21	28.5	53.53	112.06	-5.3	2.67	1	376.9	0.26	0.56	1.14
6	Chile	5.6368418707	4.52	25.27	-3.06	10.58	11.69	9.1	2.39	19.96	6.33	58.47	11.81	0.72	31.42	75.28	-11.83	10	7.07	40	36.28	-6.9	0.89	1	52.31	0.14	0.15	0.19
7	Croatia	5.5423458494	2.61	20.22	-2.71	8.4	10.45	7.6	2.81	21.9	10.2	63.4	18.33	0.86	35	81.85	-3.72	18	39.45	35.4	80.86	-2.38	2.13	1	186.9	0.26	0.46	0.79
8	Cyprus	5.0419296743	2.25	16.89	-1.97	6.97	5.51	7.7	2.55	17.1	8.1	70.8	16.54	0.85	46.1	85.48	-1.37	12.5	38.21	35	103.93	-0.01	2.2	1	291.7	0.88	0.25	0.46
9	Czechia	5.7791868266	3.84	30.1	-2.21	6.03	3.26	2.9	0.79	8.2	11.7	74.4	17.77	0.83	17.6	81.35	-6.21	19	42.59	23	41.85	-5.43	0.74	1	114.16	0.11	0.68	1.31
10	Denmark	7.8652891953	1.94	23.18	-2.56	4.5	4.13	5.1	0.97	10.8	6.8	75.5	8.73	0.93	9.3	91.14	-0.92	22	40.63	55.9	37.31	-0.5	0.54	1	146.86	0.1	0.84	2.09
11	Estonia	6.959760603	4.49	30.52	-5.11	9.26	8.35	6.5	1.56	16.7	12	74	21.95	0.96	15.8	89.88	-3.61	20	34.41	20	18.05	-2.36	0.03	1	46.23	0.16	0.61	1.02
12	Finland	7.4131514844	2.07	24.54	-2.74	3.8	3.3	7.8	1.84	17.1	13.6	72.7	8.63	0.97	31.6	90.89	-2.38	20	47.15	51.26	66.69	-2.75	0.49	1	230.43	0.35	0.78	2.02
13	France	6.9059951422	2.06	24.99	-0.81	4.58	6.98	7.9	2.32	18.9	12	67.2	7.74	0.92	28.3	79.18	-5.87	25.8	43.74	51.46	112.33	-5.82	1.39	1	315.91	0.57	0.69	1.5
14	Germany	7.2955185609	3.21	22.65	-3.42	5.48	2.79	3.6	1.16	6.9	7	75.8	17.64	0.91	7.1	82.16	-2.61	29.8	41.5	47.48	70.21	-3.28	0.59	1	297.15	0.32	0.88	2.29
15	Greece	4.6966339751	0.57	17.84	-1.17	0.65	8.34	14.9	9.26	35.5	16.6	57.2	16.43	0.73	55.7	77.14	-6.66	22	41.33	54	198.93	-5.91	2.46	1	453.39	1.49	0.64	0.86
16	Hungary	5.1143595167	5.12	30.16	-3.75	5.88	7.12	4.1	1.26	13.5	9.6	73.1	20.26	0.88	23	80.57	-6.08	10.8	38.23	15	78.11	-4.88	2.34	1	199.12	0.23	0.49	0.99
17	Iceland	6.0912769287	4.45	23	-2.28	8.02	4.35	6.2	1.28	12	6.8	79.8	7.6	0.93	15.4	83.75	-5.01	20	32.25	46.25	75.02	-6.58	4.09	1	239.78	0.12	0.75	1.72
18	Ireland	6.7428657526	2.41	24.2	0.44	9.74	13.48	6.3	1.61	14.5	8.6	69.8	14.9	0.88	12.6	94.59	0.25	12.5	43.85	40	55.27	-1.24	0.78	2	312.98	0.41	0.28	0.95
19	Israel	5.8376773735	1.49	22.79	-1.45	7.89	8.19	5.09	0.47	7.76	4.84	66.6	22.41	0.96	5.9	89.61	-4.19	23	23.83	50	68.93	-1.97	2.16	1	113.82	0.13	0.49	4.65

Crearea și folosirea de formate definite de utilizator

```
proc format;
  value somaj_fmt
    low - <4 = 'Mic'
    4 - <8 = 'Mediu'
    8 - high = 'Mare';

  value tineri_fmt
    low - <10 = 'Redus'
    10 - <20 = 'Mediu'
    20 - high = 'Ridicat';

  value lung_fmt
    low - <1 = 'Scăzut'
    1 - <3 = 'Mediu'
    3 - high = 'Ridicat';

  value ocupare_fmt
    low - <60 = 'Scăzută'
    60 - <75 = 'Mediu'
    75 - high = 'Ridicată';

  value pib_fmt
    low - <2 = 'Lentă'
    2 - <5 = 'Moderată'
    5 - high = 'Rapidă';
run;

data SGIDatas.SGI_formatat;
  set SGIDatas.SGI;

  label
    'SGI 2022 Values'n = 'Scor SGI'
    ' Unemployment'n = 'Rata Șomajului Total'
    ' Youth Unemployment'n = 'Șomajul Tinerilor'
    ' Long-term Unemployment'n = 'Șomaj pe Termen Lung'
    ' Employment'n = 'Rata de Ocupare'
    ' Real GDP Growth Rate'n = 'Creșterea PIB Real';

  format
    ' Unemployment'n somaj_fmt.
    ' Youth Unemployment'n tineri_fmt.
    ' Long-term Unemployment'n lung_fmt.
    ' Employment'n ocupare_fmt.
    ' Real GDP Growth Rate'n pib_fmt.;
run;

title "Clasificare Economico-Socială SGI";
proc print data=SGIDatas.SGI_formatat label;
  var 'SGI 2022 Values'n
      ' Unemployment'n
      ' Youth Unemployment'n
      ' Long-term Unemployment'n
      ' Employment'n
      ' Real GDP Growth Rate'n;
run;
```

Clasificare Economico-Socială SGI

Obs	Scor SGI	Rata Șomajului Total	Șomajul Tinerilor	Șomaj pe Termen Lung	Rata de Ocupare	Creșterea PIB Real
1	Australia	Mediu	Mediu	Mediu	Mediu	Moderată
2	Austria	Mediu	Mediu	Mediu	Mediu	Moderată
3	Belgium	Mediu	Mediu	Mediu	Mediu	Rapidă
4	Bulgaria	Mediu	Mediu	Mediu	Mediu	Moderată
5	Canada	Mediu	Mediu	Mediu	Mediu	Moderată
6	Chile	Mare	Mediu	Mediu	Scăzută	Rapidă
7	Croatia	Mediu	Ridicat	Mediu	Mediu	Rapidă
8	Cyprus	Mediu	Mediu	Mediu	Mediu	Rapidă
9	Czechia	Mic	Redus	Scăzut	Mediu	Moderată
10	Denmark	Mediu	Mediu	Scăzut	Ridicată	Moderată
11	Estonia	Mediu	Mediu	Mediu	Mediu	Rapidă
12	Finland	Mediu	Mediu	Mediu	Mediu	Moderată
13	France	Mediu	Mediu	Mediu	Mediu	Rapidă
14	Germany	Mic	Redus	Mediu	Ridicată	Moderată
15	Greece	Mare	Ridicat	Ridicat	Scăzută	Rapidă
16	Hungary	Mediu	Mediu	Mediu	Mediu	Rapidă
17	Iceland	Mediu	Mediu	Mediu	Ridicată	Moderată
18	Ireland	Mediu	Mediu	Mediu	Mediu	Rapidă
19	Israel	Mediu	Redus	Scăzut	Mediu	Rapidă
20	Italy	Mare	Ridicat	Ridicat	Scăzută	Rapidă
21	Japan	Mic	Redus	Mediu	Ridicată	Lentă
22	Latvia	Mediu	Mediu	Mediu	Mediu	Moderată
23	Lithuania	Mediu	Mediu	Mediu	Mediu	Moderată
24	Luxembour	Mediu	Mediu	Mediu	Mediu	Rapidă
25	Malta	Mic	Redus	Scăzut	Ridicată	Rapidă
26	Mexico	Mediu	Redus	Scăzut	Mediu	Moderată
27	Netherlan	Mediu	Redus	Scăzut	Ridicată	Rapidă
28	New Zeala	Mic	Mediu	Scăzut	Ridicată	Rapidă

Procesarea iterativă și condițională a datelor

```
proc import datafile="/home/u64210699/SGI/SGI-2022-Final-clean.csv"
    out=sgi
    dbms=csv
    replace;
    guessingrows=max;
run;

data sgi_clasificare;
    set sgi;
    length ClasaSomaj $10;

    /* Verificare lipsă */
    if missing(Unemployment) then ClasaSomaj = "Necunoscut";
    else if Unemployment < 5 then ClasaSomaj = "Mic";
    else if Unemployment < 10 then ClasaSomaj = "Mediu";
    else ClasaSomaj = "Ridicat";
run;

proc freq data=sgi_clasificare;
    tables ClasaSomaj;
    title "Clasificarea țărilor după nivelul șomajului";
run;
```

Clasificarea țărilor după nivelul șomajului

The FREQ Procedure

ClasaSomaj	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Mediu	25	60.98	25	60.98
Mic	13	31.71	38	92.68
Ridicat	3	7.32	41	100.00

```
data sgi_combinate;
    set sgi;
    length Clasa $24;
    if Unemployment < 5 and Inflation < 3 then Clasa = "Stabilitate mare";
    else if Unemployment < 10 and Inflation < 5 then Clasa = "Stabilitate moderata";
    else Clasa = "Instabilitate economica";
run;

proc freq data=sgi_combinate;
    tables Clasa;
run;
```

The FREQ Procedure

Clasa	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Instabilitate economica	7	17.07	7	17.07
Stabilitate mare	6	14.63	13	31.71
Stabilitate moderata	28	68.29	41	100.00

```
proc print data=sgi;
    where Unemployment < 5 and Inflation < 3;
    var "SGI 2022 Values"n Unemployment Inflation;
    title "Țări cu șomaj < 5% și inflație < 3% ";
run;
```

Țări cu șomaj < 5% și inflație < 3%

Obs	SGI 2022 Values	Unemployment	Inflation
21	Japan	2.97	-0.26
25	Malta	3.6	0.71
27	Netherlands	4.2	2.83
34	Slovenia	4.8	1.91
35	South Korea	3.64	2.5
40	United Kingdom	4.51	2.59

```
data sgi_economic;
    set sgi;
    array indicatori {*}
        Inflation
        Unemployment
        "Real GDP Growth Rate"n
        "Gross Fixed Capital Formation"n
        "Real Interest Rates"n;

    do i = 1 to dim(indicatori);
        if indicatori[i] <= 0 then indicatori[i] = .;
    end;

    MedieEco = mean(of indicatori[*]);

    drop i;
run;
```

Obs	SGI 2022 Values	Inflation	Unemployment	Real GDP Growth Rate	MedieEco
1	Australia	2.82	5.26	4.69	8.9275
2	Austria	2.75	6.3	4.48	10.2750
3	Belgium	3.22	6.3	6.28	10.5300
4	Bulgaria	2.84	5.3	4.18	7.9700
5	Canada	3.4	7.44	4.56	9.7725
6	Chile	4.52	9.1	11.69	12.6450
7	Croatia	2.61	7.6	10.45	10.2200
8	Cyprus	2.25	7.7	5.51	8.0875
9	Czechia	3.84	2.9	3.26	10.0250
10	Denmark	1.94	5.1	4.13	8.5875
11	Estonia	4.49	6.5	8.35	12.4650
12	Finland	2.07	7.8	3.3	9.4275
13	France	2.06	7.9	6.98	10.4825
14	Germany	3.21	3.6	2.79	8.0625
15	Greece	0.57	14.9	8.34	10.4125
16	Hungary	5.12	4.1	7.12	11.6250
17	Iceland	4.45	6.2	4.35	9.5000
18	Ireland	2.41	6.3	13.48	9.3660
19	Israel	1.49	5.09	8.19	9.3900
20	Italy	1.94	9.7	6.64	7.7240
21	Japan	.	2.97	1.62	7.7000
22	Latvia	3.24	7.9	4.67	10.7725
23	Lithuania	4.63	7.4	4.94	9.0575
24	Luxembourg	3.47	5.3	6.89	8.4600
25	Malta	0.71	3.6	9.41	9.2400
26	Mexico	5.69	4.26	4.8	8.7425
27	Netherlands	2.83	4.2	5.04	8.3200
28	New Zealand	3.94	3.97	5.63	9.5150
29	Norway	3.48	4.5	3.92	9.2775
30	Poland	5.1	3.4	5.67	8.7400
31	Portugal	0.94	6.7	4.88	8.0500
32	Romania	5.05	5.6	5.88	10.6925

```
proc print data=sgi_economic;
    var "SGI 2022 Values"n Inflation Unemployment "Real GDP Growth Rate"n MedieEco;
run;
```

Crearea de subseturi de date

```
data clasificare_emisiuni;
    set sgi;
    length ClasaCO2 $12;

    if "Gross Greenhouse Gas Emissions"n < 10 then ClasaCO2 = "Mică";
    else if "Gross Greenhouse Gas Emissions"n < 20 then ClasaCO2 = "Mediu";
    else ClasaCO2 = "Ridicată";
run;

proc freq data=clasificare_emisiuni;
    tables ClasaCO2;
run;
```

The FREQ Procedure

ClasaCO2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Mediu	8	19.51	8	19.51
Mică	32	78.05	40	97.56
Ridicată	1	2.44	41	100.00

```
proc print data=sgi;
    where "SGI 2022 Values"n in ("Finland", "Iceland", "Sweden", "Norway", "Denmark")
        and Unemployment between 4 and 10
        and Inflation <5;
    var "SGI 2022 Values"n Unemployment Inflation;
    title "Țări nordice cu șomaj 4–10% și inflație mică";
run;
```

Țări nordice cu șomaj 4–10% și inflație mică

Obs	SGI 2022 Values	Unemployment	Inflation
10	Denmark	5.1	1.94
12	Finland	7.8	2.07
17	Iceland	6.2	4.45
29	Norway	4.5	3.48
37	Sweden	9	2.65

Utilizarea de funcții SAS

```
12 data sgi_rotunjit;
13     set sgi;
14     InflatieRotunjita = round(Inflation, 0.5); /* rotunjire la 0.5 */
15     ScorEconomie = round("Real GDP Growth Rate"n - Inflation, 0.1);
16 run;
17
18 proc print data=sgi_rotunjit;
19     var "SGI 2022 Values"n Inflation InflatieRotunjita "Real GDP Growth Rate"n ScorEconomie;
20     title "Rotunjiri și scoruri economice";
21 run;
22
```

Rotunjiri și scoruri economice

Obs	SGI 2022 Values	Inflation	InflatieRotunjita	Real GDP Growth Rate	ScorEconomie
1	Australia	2.82	3.0	4.69	1.9
2	Austria	2.75	3.0	4.48	1.7
3	Belgium	3.22	3.0	6.28	3.1
4	Bulgaria	2.84	3.0	4.18	1.3
5	Canada	3.4	3.5	4.56	1.2
6	Chile	4.52	4.5	11.69	7.2
7	Croatia	2.61	2.5	10.45	7.8
8	Cyprus	2.25	2.5	5.51	3.3
9	Czechia	3.84	4.0	3.26	-0.6
10	Denmark	1.94	2.0	4.13	2.2
11	Estonia	4.49	4.5	8.35	3.9
12	Finland	2.07	2.0	3.3	1.2
13	France	2.06	2.0	6.98	4.9
14	Germany	3.21	3.0	2.79	-0.4
15	Greece	0.57	0.5	8.34	7.8
16	Hungary	5.12	5.0	7.12	2.0
17	Iceland	4.45	4.5	4.35	-0.1
18	Ireland	2.41	2.5	13.48	11.1
19	Israel	1.49	1.5	8.19	6.7
20	Italy	1.94	2.0	6.64	4.7
21	Japan	-0.26	-0.5	1.62	1.9
22	Latvia	3.24	3.0	4.67	1.4
23	Lithuania	4.63	4.5	4.94	0.3
24	Luxembourg	3.47	3.5	6.89	3.4

```
24 data sgi_prefixe;
25     set sgi;
26     PrefixTara = substr("SGI 2022 Values"n, 1, 3);
27 run;
28
29 proc freq data=sgi_prefixe;
30     tables PrefixTara;
31     title "Prefixe din numele țărilor (primele 3 litere)";
32 run;
33
```

Prefixe din numele țărilor (primele 3 litere)

The FREQ Procedure

PrefixTara	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Aus	2	4.88	2	4.88
Bel	1	2.44	3	7.32
Bul	1	2.44	4	9.76
Can	1	2.44	5	12.20
Chi	1	2.44	6	14.63
Cro	1	2.44	7	17.07
Cyp	1	2.44	8	19.51
Cze	1	2.44	9	21.95
Den	1	2.44	10	24.39
Est	1	2.44	11	26.83
Fin	1	2.44	12	29.27
Fra	1	2.44	13	31.71
Ger	1	2.44	14	34.15
Gre	1	2.44	15	36.59
Hun	1	2.44	16	39.02
Ice	1	2.44	17	41.46
Ire	1	2.44	18	43.90
Isr	1	2.44	19	46.34
Ita	1	2.44	20	48.78
Jap	1	2.44	21	51.22
Lat	1	2.44	22	53.66
Lit	1	2.44	23	56.10
Lux	1	2.44	24	58.54
Mal	1	2.44	25	60.98
Mex	1	2.44	26	63.41
Net	1	2.44	27	65.85
New	1	2.44	28	68.29
Nor	1	2.44	29	70.73
Pol	1	2.44	30	73.17

```sas
data sgi_continent_check;
    set sgi;
    length Este_European $3;
    if find(Continent, "Europe") > 0 then Este_European = "Da";
    else Este_European = "Nu";
run;

proc freq data=sgi_continent_check;
    tables Este_European;
    title "Distribuția țărilor europene";
run;
```

Distribuția țărilor europene

The FREQ Procedure

Este_European	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Da	32	78.05	32	78.05
Nu	9	21.95	41	100.00

Combinarea seturilor de date prin proceduri specifice SAS si SQL

```sas
data sgi_economic;
    set sgi(keep="SGI 2022 Values"n Unemployment Inflation);
run;

data sgi_social;
    set sgi(keep="SGI 2022 Values"n "Life Satisfaction"n "Poverty Rate"n);
run;
```

MERGE necesită ca seturile de date să fie sortate după variabila comună specificată în BY.

```sas
proc sort data=sgi_economic;
    by "SGI 2022 Values"n;
run;

proc sort data=sgi_social;
    by "SGI 2022 Values"n;
run;
```

```sas
data sgi_combinate_sas;
    merge sgi_economic sgi_social;
    by "SGI 2022 Values"n;
run;

proc print data=sgi_combinate_sas(obs=5);
    title "Fuziune prin SAS MERGE";
run;
```

Fuziune prin SAS MERGE

Obs	SGI 2022 Values	Inflation	Unemployment	Poverty Rate	Life Satisfaction
1	Australia	2.82	5.26	12.4	7.16
2	Austria	2.75	6.3	9.1	7.16
3	Belgium	3.22	6.3	6.3	6.8
4	Bulgaria	2.84	5.3	14.6	5.37
5	Canada	3.4	7.44	11.6	7.03

```sql
proc sql;
    create table medii_pe_continent as
    select
        e.Continent,
        mean(e.Unemployment) as SomajMediu format=6.2,
        mean(e.Inflation) as InflatieMedie format=6.2,
        mean(s."Poverty Rate"n) as SaracieMedie format=6.2,
        mean(s."Life Satisfaction"n) as SatisfactieMedie format=6.2
    from sgi_economic as e
    left join sgi_social as s
    on e."SGI 2022 Values"n = s."SGI 2022 Values"n
    group by e.Continent;
quit;

proc print data=medii_pe_continent;
    title "Medii economice și sociale pe continent";
run;
```

Medii economice și sociale pe continent

Obs	Continent	SomajMediu	InflatieMedie	SaracieMedie	SatisfactieMedie
1	Asia	3.90	1.24	16.10	6.45
2	Europe	6.70	3.39	9.74	6.66
3	North America	5.70	4.59	15.40	6.71
4	Oceania	4.62	3.38	12.40	7.18
5	South America	9.10	4.52	16.50	6.17

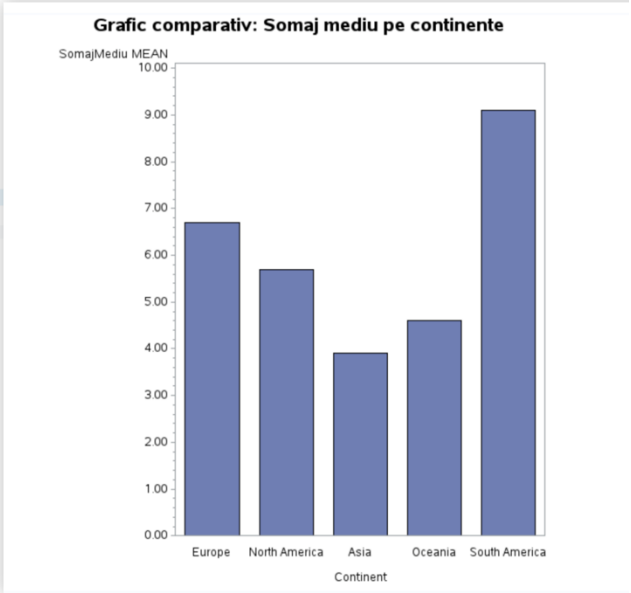
Raportare, Analiză statistică, Vizualizare grafică

```sql
proc sql;
    create table somaj_pe_continent as
    select Continent,
        mean(Unemployment) as SomajMediu format=6.2
    from sgi
    group by Continent;
quit;
```

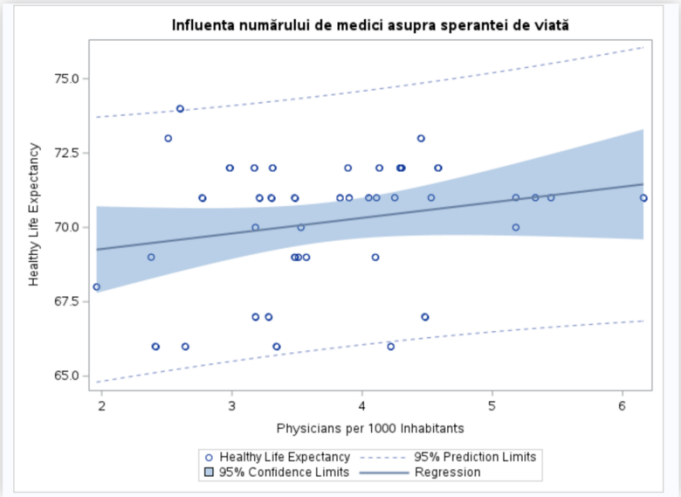
	Continent	SomajMediu
1	Asia	3.90
2	Europe	6.70
3	North America	5.70
4	Oceania	4.62
5	South America	9.10

```
pattern1 value=solid;
axis1 label=('Continent') order=('Europe' 'North America' 'Asia' 'Oceania' 'South America');

proc gchart data=somaj_pe_continent;
    vbar Continent /
        sumvar=SomajMediu
        type=mean
        width=8
        space=3
        maxis=axis1
        coutline=black;
    title "Grafic comparativ: Somaj mediu pe continente";
    subtitle "Date agregate din SGI";
run;
quit;
```




```
proc sgplot data=sgi;
    scatter x="Physicians per 1000 Inhabitants"n y="Healthy Life Expectancy"n;
    reg x="Physicians per 1000 Inhabitants"n y="Healthy Life Expectancy"n / clm cli;
    title "Influenta numărului de medici asupra sperantei de viată";
run;
```



Programare Python

Steamlit

```python
st.sidebar.header("Country Comparison")
country1 = st.sidebar.selectbox("Select first country", sorted(df.iloc[:, 0].unique()), key="country1")
country2 = st.sidebar.selectbox("Select second country", sorted(df.iloc[:, 0].unique()), key="country2")
comparison_indicator = st.sidebar.selectbox("Select an Indicator", df.columns[1:], key="comparison_indicator")
st.sidebar.header("Top/Bottom Rankings")
ranking_indicator = st.sidebar.selectbox("Select indicator for ranking", numeric_columns, key="ranking_indicator")
ranking_count = st.sidebar.slider("Number of countries to show", 3, 15, 5, key="ranking_count")
ranking_order = st.sidebar.radio("Order", ["Top (Highest)", "Bottom (Lowest)"], key="ranking_order")
tab1, tab2, tab3, tab4, tab5, tab6 = st.tabs(["Dataset Overview", "Data Analysis", "Country Details", "Comparisons", "Factor Analysis", "Geographic Analysis"])

with tab1:
    col1, col2 = st.columns([3, 1])

    with col1:
        st.subheader("Dataset Preview")
        sort_col, sort_order = st.columns(2)
        with sort_col:
            sort_by = st.selectbox("Sort by column", df.columns, key="sort_column")
        with sort_order:
            ascending = st.radio("Sort order", ["Ascending", "Descending"], key="sort_order") == "Ascending"
        st.dataframe(df.sort_values(by=sort_by, ascending=ascending))

    with col2:
        st.subheader("Summary Statistics")
        st.dataframe(df.describe())

        st.subheader("Dataset Info")
        st.write(f"Total countries: {len(df)}")
        st.write(f"Total indicators: {len(df.columns) - 1}")
```

Utilizarea pachetului GeoPandas

```python
with tab6:
    st.header("Geographic Analysis")

    st.info("World maps for easier geographic comparison.")

    @st.cache_data
    def load_world_data():
        url = "https://naciscdn.org/naturalearth/110m/cultural/ne_110m_admin_0_countries.zip"
        try:
            world = gpd.read_file(url)
            country_name_map = {
                'United States of America': 'United States',
                'Czech Rep.': 'Czechia',
                'Korea': 'South Korea',
                'United Kingdom': 'United Kingdom'
            }
            country_col = 'NAME' if 'NAME' in world.columns else 'ADMIN'
            world[country_col] = world[country_col].replace(country_name_map)

            return world, country_col
        except Exception as e:
            st.error(f"Error loading world data: {str(e)}")
            return None, None

    try:
        world_data = load_world_data()
        if world_data is not None:
            world, country_col = world_data

            col1, col2 = st.columns([3, 1])

            with col2:
                st.subheader("Map Settings")

                geo_indicator = st.selectbox("Select Indicator", df.columns[1:], key="geo_indicator")

                use_binning = st.checkbox("Enable Binning", value=False, key="use_binning")

                if use_binning:
                    bin_method = st.radio(
                        "Binning Method",
                        ["Equal Width", "Equal Frequency", "Custom"],
                        key="bin_method"
                    )

                    if bin_method in ["Equal Width", "Equal Frequency"]:
                        n_bins = st.slider("Number of Bins", 2, 10, 5, key="n_bins")
                    else:
                        custom_bins_text = st.text_area(
                            "Enter bin edges (comma-separated)",
                            value="0,25,50,75,100",
                            key="custom_bins"
                        )
                        try:
                            custom_bins = [float(x.strip()) for x in custom_bins_text.split(",")]
                        except ValueError:
                            st.error("Please enter valid numeric values")
                            custom_bins = [0, 25, 50, 75, 100]

```

Tratarea valorilor lipsă

```python
def fill_na(x):
    assert isinstance(x, pd.DataFrame)
    for col in x.columns:
        if is_numeric_dtype(x[col]):
            x[col] = x[col].fillna(x[col].mean())
        else:
            x[col] = x[col].fillna(x[col].mode()[0])
    return x
```

Metode de codificare a datelor

```python
def normalize_data(data, method='zscore'):
    norm_df = data.copy()

    for col in numeric_columns:
        if method == 'zscore':
            norm_df[col] = stats.zscore(data[col].values)
        elif method == 'log':
            min_val = data[col].min()
            if min_val <= 0:
                offset = abs(min_val) + 1
                norm_df[col] = np.log(data[col] + offset)
            else:
                norm_df[col] = np.log(data[col])
        elif method == 'minmax':
            norm_df[col] = (data[col] - data[col].min()) / (data[col].max() - data[col].min())
    return norm_df


std_df = normalize_data(df, method='zscore')
zscore_df = normalize_data(df, method='zscore')
log_df = normalize_data(df, method='log')
minmax_df = normalize_data(df, method='minmax')
```

Metode de scalare

```python
from sklearn.preprocessing import StandardScaler
```

```python
        else:
            x_std = StandardScaler().fit_transform(x)
            kmo_all, kmo_model = calculate_kmo(fa_data)
            chi_square_value, p_value = calculate_bartlett_sphericity(fa_data)
```

Predicții statistice, gruparea și agregarea datelor în pachetul pandas

```
    df_filled = df.copy()
    df_filled = fill_na(df_filled)


    category_stats = {}
    for category, indicators in indicator_categories.items():
        category_data = df_filled[indicators]
        category_stats[category] = {
            "Mean": category_data.mean().mean(),
            "Median": category_data.median().median(),
            "Std": category_data.std().mean(),
            "Min": category_data.min().min(),
            "Max": category_data.max().max()
        }

    stats_df = pd.DataFrame(category_stats).T
    st.write("**Statistics by Category:**")
    st.dataframe(stats_df)


    plt.figure(figsize=(10, 6))
    sns.barplot(x=stats_df.index, y=stats_df['Mean'])
    plt.xticks(rotation=45)
    plt.title("Mean Values by Category")
    st.pyplot(plt)
```

Utilizarea funcțiilor de grup

```
with col1:
    fig, ax = plt.subplots(figsize=(10, 5))
    sns.barplot(x=top_countries.iloc[:, 0], y=top_countries[ranking_indicator], ax=ax)
    ax.set_title(f"{'Top' if ranking_order == 'Top (Highest)' else 'Bottom'} {ranking_count} Countries")
    ax.set_ylabel(ranking_indicator)
    plt.xticks(rotation=45)
    plt.tight_layout()
    st.pyplot(fig)

with col2:
    st.dataframe(top_countries[[df.columns[0], ranking_indicator]])

st.header("Indicator Correlations")
with st.expander("Correlation Analysis"):
    corr_norm_method = st.selectbox(
        "Select normalization method for correlation analysis",
        ["Raw Values", "Z-Score Normalization", "Log Transformation", "Min-Max Scaling"],
        key="corr_norm_method"
    )
```

Clusterizare prin analiza factoriala

```
575     st.header("Factor Analysis")
576
577     fa_category = st.selectbox("Select Category for Factor Analysis",
578                                list(indicator_categories.keys()),
579                                key="fa_category")
580
581
582     fa_indicators = indicator_categories[fa_category]
583
584     if len(fa_indicators) < 3:
585         st.error(f"Not enough indicators in the {fa_category} category. Need at least 3 indicators for factor analysis.")
586     else:
587         fa_data = df[fa_indicators].copy()
588         fa_data = fill_na(fa_data)
589         x = fa_data.values
590         n, m = x.shape
591         if n <= 1:
592             st.error("Not enough data for factor analysis.")
593         else:
594             x_std = StandardScaler().fit_transform(x)
595             kmo_all, kmo_model = calculate_kmo(fa_data)
596             chi_square_value, p_value = calculate_bartlett_sphericity(fa_data)
597
598             col1, col2 = st.columns(2)
599             with col1:
600                 st.subheader("Kaiser-Meyer-Olkin (KMO) Test")
601                 st.write(f"KMO Score: {kmo_model:.3f}")
602                 if kmo_model < 0.5:
603                     st.warning("KMO < 0.5: Sample may not be adequate for factor analysis")
604                 elif kmo_model < 0.7:
605                     st.info("0.5 ≤ KMO < 0.7: Sample is mediocre but acceptable")
606                 else:
607                     st.success("KMO ≥ 0.7: Sample is adequate for factor analysis")
608
609             with col2:
610                 st.subheader("Bartlett's Test of Sphericity")
611                 st.write(f"Chi-square: {chi_square_value:.3f}")
612                 st.write(f"p-value: {p_value:.6f}")
613                 if p_value < 0.05:
614                     st.success("p < 0.05: Correlation matrix is not an identity matrix")
615                 else:
616                     st.warning("p ≥ 0.05: Correlation matrix might be an identity matrix")
617
618
619             eigenvalues, _ = fact.factor_analyzer.FactorAnalyzer().fit(x_std).get_eigenvalues()
620
621             st.subheader("Scree Plot")
622             fig, ax = plt.subplots(figsize=(10, 6))
623             ax.plot(range(1, len(eigenvalues) + 1), eigenvalues, marker='o')
624             ax.axhline(y=1, color='r', linestyle='--')
625             ax.set_xlabel('Factor Number')
626             ax.set_ylabel('Eigenvalue')
627             ax.set_title('Scree Plot')
628             ax.grid(True)
629             st.pyplot(fig)
630
631             n_factors = st.slider("Select number of factors",
632                                   min_value=1,
633                                   max_value=min(m, 10),
634                                   value=min(3, m))
635
636
637             model_af = fact.FactorAnalyzer(n_factors=n_factors, rotation="varimax")
638             model_af.fit(x_std)
639
640
641             varianta = model_af.get_factor_variance()
642             etichete = [f"F{i+1}" for i in range(n_factors)]
643             varianta_df = tabelare_varianta(varianta, etichete)
644
645             st.subheader("Factor Analysis Results")
646             st.write("**Factor Variance**")
647             st.dataframe(varianta_df)
```

Corelograma

```python
def corelograma(r, title):
    fig, ax = plt.subplots(figsize=(12, 8))
    sns.heatmap(r, vmin=-1, vmax=1, cmap="RdYlBu", center=0, annot=True, ax=ax)
    plt.title(title)
    return fig
```

Regresie multipla

```python
with tab7:
    st.header("Multiple Regression with 3D Visualization")



    col1, col2 = st.columns([1, 2])

    with col1:
        st.subheader("Variable Selection")


        dependent_var = st.selectbox(
            "Select Dependent Variable (Y)",
            numeric_columns,
            key="dependent_var"
        )

        available_vars = [col for col in numeric_columns if col != dependent_var]

        independent_var1 = st.selectbox(
            "Select First Independent Variable (X1)",
            available_vars,
            key="independent_var1"
        )

        available_vars2 = [col for col in available_vars if col != independent_var1]
        independent_var2 = st.selectbox(
            "Select Second Independent Variable (X2)",
            available_vars2,
            key="independent_var2"
        )


        normalize_regression = st.checkbox("Normalize data before regression", value=True)

        if normalize_regression:
            regression_norm_method = st.selectbox(
                "Normalization Method",
                ["Z-Score", "Min-Max", "Log"],
                key="regression_norm_method"
            )

    with col2:
        st.subheader("Regression Results")


        regression_data = df[[dependent_var, independent_var1, independent_var2]].copy()
        regression_data = fill_na(regression_data)

        if normalize_regression:
            if regression_norm_method == "Z-Score":
                regression_data = normalize_data(regression_data, method='zscore')
            elif regression_norm_method == "Min-Max":
                regression_data = normalize_data(regression_data, method='minmax')
            else:
                regression_data = normalize_data(regression_data, method='log')


        X = regression_data[[independent_var1, independent_var2]].values
        y = regression_data[dependent_var].values


        mask = ~(np.isnan(X).any(axis=1) | np.isnan(y))
        X = X[mask]
        y = y[mask]
        countries_filtered = df[df.columns[0]][mask].values

        if len(X) > 3:

            model = LinearRegression()
            model.fit(X, y)

            y_pred = model.predict(X)
```