SPRAWOZDANIE

Zajęcia: Nauka o danych I

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 1 Data 28.09.2024 Temat: Wariant 8	Imię Nazwisko Hubert Mentel Informatyka II stopień, nie stacjonarne, 1 semestr, gr.1a

1. Zadanie dotyczy pobrania danych z pliku, tworzenia ramki danych, wykonania poszczególnych zadań poniżej na podstawie odpowiedniego zbioru danych: wariant 8 zadania

ORB General Population COVID-19 Health Services Disruption Survey 2020
http://ghdx.healthdata.org/record/ihme-data/orb-general-population-covid-19-health-services-disruption-survey-2020

Pliki dostępne są na GitHubie pod linkiem: https://github.com/HubiPX/NOD

2. Opis programu opracowanego (kody źródłowe, rzuty ekranu)

[1]:
```
#ladowanie biblioteki Pandas
import pandas as pd
```

[2]:
```
#tworzenie ramki danych ze slownika

data = {'kolumna_1': [1, 2, 3, 4], 'kolumna_2': ['a', 'b', 'c', 'd'], 'kolumna_3': ['1a', '2b', '3c', '4d']}

pd.DataFrame.from_dict(data)
```

[2]:

	kolumna_1	kolumna_2	kolumna_3
0	1	a	1a
1	2	b	2b
2	3	c	3c
3	4	d	4d

[5]:
```
#zachowanie ramki danych pobranych z pliku w formacie csv (xlsx)

df = pd.read_csv('IHME_ORB_C19HSDS_2020_Y2020M12D03.CSV')
print(df)
```

```
         SbjNum NetDuration InterviewTimeVStart InterviewTimeVEnd  \
0     133476254     0:10:14     7/17/2020 13:53   7/17/2020 14:26
1     133281846     0:22:16     7/10/2020 12:53   7/10/2020 14:47
2     133280780     0:19:23     7/10/2020 12:35   7/10/2020 12:54
3     133281834     0:10:11     7/10/2020 10:21   7/10/2020 10:32
4     133491249     0:09:59      7/18/2020 8:27    7/18/2020 8:39
...         ...         ...                ...               ...
3053  133323839     0:09:03     7/11/2020 12:44   7/11/2020 12:53
3054  133305818     0:06:57     7/11/2020 16:18   7/11/2020 16:25
3055  133260048     0:21:46      7/9/2020 11:49    7/9/2020 12:12
3056  133305807     0:06:50      7/11/2020 9:05    7/11/2020 9:12
3057  133352713     0:09:20      7/13/2020 9:56   7/13/2020 14:44

                Date  Srvyr  Country  LANG  R1  R1_5  ... G11_Other G11_99  \
0     7/17/2020 8:53   3232        2     1   9  15.0  ...       NaN    NaN
1     7/10/2020 7:53   3206        2     4  12  22.0  ...       NaN    NaN
2     7/10/2020 7:35   3202        2     3  10  13.0  ...       NaN    NaN
3     7/10/2020 5:21   3212        2     1  12   9.0  ...       NaN    NaN
4     7/18/2020 3:27   3225        2     3  11  28.0  ...       NaN    NaN
...              ...    ...      ...   ...  ..   ...  ...       ...    ...
3053  7/11/2020 5:44   3012        1     7   8   NaN  ...       NaN    NaN
3054  7/11/2020 9:18   3008        1     1   3   NaN  ...       NaN    NaN
3055   7/9/2020 4:49   3004        1     1   7   NaN  ...       NaN    NaN
3056  7/11/2020 2:05   3008        1     1   3   NaN  ...       NaN    NaN
3057  7/13/2020 2:56   3003        1     1   2   NaN  ...       NaN    NaN

      FinalOutcome  NumOfVisits  weight_combined kenya_weight  nigeria_weight  \
0                1            1         0.829860          NaN        0.829860
1                1            1         1.416946          NaN        1.416946
2                1            1         0.883601          NaN        0.883601
3                1            1         1.416946          NaN        1.416946
4                1            1         0.829860          NaN        0.829860
...            ...          ...              ...          ...             ...
3053             1            1         3.791351     3.791351             NaN
3054             1            1         1.157689     1.157689             NaN
3055             1            1         0.799916     0.799916             NaN
3056             1            1         0.799916     0.799916             NaN
3057             1            3         1.157689     1.157689             NaN

     southafrica_weight  agegroup gk_weight
0                   NaN         1  1.555754
1                   NaN         2  1.949579
2                   NaN         2  2.151458
3                   NaN         2  2.325065
4                   NaN         1  1.640484
...                 ...       ...       ...
3053                NaN         3  2.354356
3054                NaN         2  1.869021
3055                NaN         1  1.907830
3056                NaN         1  1.753344
3057                NaN         2  1.869021

[3058 rows x 247 columns]
```

[7]:
```
#tworzenie ramki danych z listy list

list_data = [["Pies", "Kot", "Kura"], [12,   4,  2]]

pd.DataFrame(list_data)
```

[7]:

	0	1	2
0	Pies	Kot	Kura
1	12	4	2

[9]:
```
#transponowanie (wymieniamy kolumny a wierszy)

df_1 = pd.DataFrame.transpose(pd.DataFrame(list_data))
print(df_1)
```

```
      0   1
0  Pies  12
1   Kot   4
2  Kura   2
```

[11]:
```
#wyswietlic pierwsze 10 wierszy ramki danych
df.head(10)
```

[11]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R1_5	...	G11_Other	G11_99	FinalOutcome	NumOfVisits
0	133476254	0:10:14	7/17/2020 13:53	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	15.0	...	NaN	NaN	1	1
1	133281846	0:22:16	7/10/2020 12:53	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	22.0	...	NaN	NaN	1	1
2	133280780	0:19:23	7/10/2020 12:35	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	13.0	...	NaN	NaN	1	1
3	133281834	0:10:11	7/10/2020 10:21	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	9.0	...	NaN	NaN	1	1
4	133491249	0:09:59	7/18/2020 8:27	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	28.0	...	NaN	NaN	1	1
5	133309774	0:17:14	7/11/2020 11:36	7/11/2020 11:54	7/11/2020 6:36	3233	2	1	11	26.0	...	NaN	NaN	1	1
6	133520640	0:09:09	7/19/2020 13:20	7/19/2020 13:29	7/19/2020 8:20	3240	2	1	9	35.0	...	NaN	NaN	1	1
7	133219300	0:12:21	7/8/2020 11:48	7/8/2020 12:05	7/8/2020 6:48	3214	2	1	14	33.0	...	NaN	NaN	1	1
8	133325892	0:18:05	7/12/2020 9:42	7/12/2020 10:03	7/12/2020 4:42	3204	2	3	11	29.0	...	NaN	NaN	1	1
9	133496489	0:11:29	7/18/2020 17:24	7/18/2020 17:36	7/18/2020 12:24	3202	2	3	10	10.0	...	NaN	NaN	1	1

10 rows × 247 columns

[13]:
```
#wyswietlic ostatnie 10 wierszy ramki danych

df.tail(10)
```

[13]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R1_5	...	G11_Other	G11_99	FinalOutcome	NumOfV
3048	133210782	0:08:02	7/8/2020 10:57	7/8/2020 11:23	7/8/2020 3:57	3008	1	1	4	NaN	...	NaN	NaN	1	
3049	133323835	0:07:24	7/11/2020 12:11	7/11/2020 12:19	7/11/2020 5:11	3012	1	1	8	NaN	...	NaN	NaN	1	
3050	133495603	0:08:51	7/18/2020 15:10	7/18/2020 15:48	7/18/2020 8:10	3006	1	1	8	NaN	...	NaN	NaN	1	
3051	133259534	0:10:46	7/9/2020 13:39	7/9/2020 15:03	7/9/2020 6:39	3003	1	1	1	NaN	...	NaN	NaN	1	
3052	133192430	0:14:41	7/7/2020 18:00	7/7/2020 18:15	7/7/2020 11:00	3005	1	1	2	NaN	...	NaN	NaN	1	
3053	133323839	0:09:03	7/11/2020 12:44	7/11/2020 12:53	7/11/2020 5:44	3012	1	7	8	NaN	...	NaN	NaN	1	
3054	133305818	0:06:57	7/11/2020 16:18	7/11/2020 16:25	7/11/2020 9:18	3008	1	1	3	NaN	...	NaN	NaN	1	
3055	133260048	0:21:46	7/9/2020 11:49	7/9/2020 12:12	7/9/2020 4:49	3004	1	1	7	NaN	...	NaN	NaN	1	
3056	133305807	0:06:50	7/11/2020 9:05	7/11/2020 9:12	7/11/2020 2:05	3008	1	1	3	NaN	...	NaN	NaN	1	
3057	133352713	0:09:20	7/13/2020 9:56	7/13/2020 14:44	7/13/2020 2:56	3003	1	1	2	NaN	...	NaN	NaN	1	

10 rows × 247 columns

[15]:
```
#wyswietlic informacje, o ramce danych

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3058 entries, 0 to 3057
Columns: 247 entries, SbjNum to gk_weight
dtypes: float64(208), int64(18), object(21)
memory usage: 5.8+ MB
```

[17]:
```
#wyswietlic, ile wierszy i kolumn znajduje sie w ramce danych

df.shape
```

[17]:
```
(3058, 247)
```

[19]:
```
#wyswietlic informacje statystyczna o kolumnach liczbowych (wartosci
#niepowtarzalne, srednia, odchylenie standardowe, minimum, kwartyle,
#maksimum)

df.describe()
```

[19]:

	SbjNum	Srvyr	Country	LANG	R1	R1_5	R4	R5	R6	R7	...	G11_96	G11_99	Fina
count	3.058000e+03	3058.000000	3058.000000	3058.000000	3058.000000	1016.000000	3058.000000	3058.000000	3058.000000	3058.000000	...	32.000000	2.0	
mean	1.333905e+08	3084.743623	2.012426	2.180510	22.521583	27.378937	1.503270	34.038914	1.771419	24.448986	...	0.718750	1.0	
std	1.256690e+05	97.219107	0.817203	2.364438	20.294923	10.088041	0.500071	11.386285	3.130841	26.377909	...	0.456803	0.0	
min	1.331715e+08	3001.000000	1.000000	1.000000	1.000000	9.000000	1.000000	18.000000	1.000000	1.000000	...	0.000000	1.0	
25%	1.332825e+08	3010.000000	1.000000	1.000000	7.000000	18.750000	1.000000	25.000000	1.000000	8.000000	...	0.000000	1.0	
50%	1.333755e+08	3026.000000	2.000000	1.000000	11.000000	28.000000	2.000000	31.500000	2.000000	21.000000	...	1.000000	1.0	
75%	1.334985e+08	3212.000000	3.000000	1.000000	49.000000	35.000000	2.000000	40.000000	2.000000	24.000000	...	1.000000	1.0	
max	1.336450e+08	3264.000000	3.000000	11.000000	54.000000	45.000000	2.000000	99.000000	99.000000	99.000000	...	1.000000	1.0	

8 rows × 226 columns

[20]:
```
#wyswietlic informacje statystyczna o kolumnach kategoryzowanych (ile
#unikalnych wartosci, top - jaka jest najpopularniejsza wartosc, freq -
#jak czesto najpopularniejsza


df.describe(include = 'all')
```

[20]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R1_5	...	G11_Oth
count	3.058000e+03	3058	3058	3058	3058	3058.000000	3058.000000	3058.000000	3058.000000	1016.000000	...	2
unique	NaN	983	2611	2607	2611	NaN	NaN	NaN	NaN	NaN	...	
top	NaN	0:07:39	7/9/2020 13:11	7/10/2020 12:31	7/18/2020 8:50	NaN	NaN	NaN	NaN	NaN	...	XXX
freq	NaN	11	5	4	4	NaN	NaN	NaN	NaN	NaN	...	2
mean	1.333905e+08	NaN	NaN	NaN	NaN	3084.743623	2.012426	2.180510	22.521583	27.378937	...	Na
std	1.256690e+05	NaN	NaN	NaN	NaN	97.219107	0.817203	2.364438	20.294923	10.088041	...	Na
min	1.331715e+08	NaN	NaN	NaN	NaN	3001.000000	1.000000	1.000000	1.000000	9.000000	...	Na
25%	1.332825e+08	NaN	NaN	NaN	NaN	3010.000000	1.000000	1.000000	7.000000	18.750000	...	Na
50%	1.333755e+08	NaN	NaN	NaN	NaN	3026.000000	2.000000	1.000000	11.000000	28.000000	...	Na
75%	1.334985e+08	NaN	NaN	NaN	NaN	3212.000000	3.000000	1.000000	49.000000	35.000000	...	Na
max	1.336450e+08	NaN	NaN	NaN	NaN	3264.000000	3.000000	11.000000	54.000000	45.000000	...	Na

11 rows × 247 columns

[22]:
```
df = df.dropna(axis=1)
df
```

[22]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	...	R11	R12	H1	H6	H11	FinalOutcome
0	133476254	0:10:14	7/17/2020 13:53	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	XXXX	...	80000.0	1	1	1	1	1
1	133281846	0:22:16	7/10/2020 12:53	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	XXXX	...	2000.0	1	2	2	1	1
2	133280780	0:19:23	7/10/2020 12:35	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	XXXX	...	43000.0	1	1	2	1	1
3	133281834	0:10:11	7/10/2020 10:21	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	XXXX	...	10000.0	1	2	1	1	1
4	133491249	0:09:59	7/18/2020 8:27	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	XXXX	...	15000.0	1	2	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3053	133323839	0:09:03	7/11/2020 12:44	7/11/2020 12:53	7/11/2020 5:44	3012	1	7	8	XXXX	...	8000.0	1	1	1	2	1
3054	133305818	0:06:57	7/11/2020 16:18	7/11/2020 16:25	7/11/2020 9:18	3008	1	1	3	XXXX	...	12000.0	1	1	2	1	1
3055	133260048	0:21:46	7/9/2020 11:49	7/9/2020 12:12	7/9/2020 4:49	3004	1	1	7	XXXX	...	5000.0	1	1	2	1	1
3056	133305807	0:06:50	7/11/2020 9:05	7/11/2020 9:12	7/11/2020 2:05	3008	1	1	3	XXXX	...	0.0	1	2	2	2	1
3057	133352713	0:09:20	7/13/2020 9:56	7/13/2020 14:44	7/13/2020 2:56	3003	1	1	2	XXXX	...	99.0	2	1	2	1	1

3058 rows × 27 columns

[25]:
```
#przedstawic wybor wierszy i kolumny uzywajac nazw oraz indeksow na
#rozne sposoby

df["SbjNum"]
```

[25]:
```
0       133476254
1       133281846
2       133280780
3       133281834
4       133491249
          ...
3053    133323839
3054    133305818
3055    133260048
3056    133305807
3057    133352713
Name: SbjNum, Length: 3058, dtype: int64
```

[27]:
```
# wszystkie wiersze z kolumny index 1
column_id = df.iloc[:,1]
print(column_id)
```

```
0       0:10:14
1       0:22:16
2       0:19:23
3       0:10:11
4       0:09:59
          ...
3053    0:09:03
3054    0:06:57
3055    0:21:46
3056    0:06:50
3057    0:09:20
Name: NetDuration, Length: 3058, dtype: object
```

[29]:
```
df.SbjNum
```

[29]:
```
0       133476254
1       133281846
2       133280780
3       133281834
4       133491249
          ...
3053    133323839
3054    133305818
3055    133260048
3056    133305807
3057    133352713
Name: SbjNum, Length: 3058, dtype: int64
```

[31]:
```
df[["SbjNum","NetDuration"]] # wybor wielu kolumn
```

[31]:

	SbjNum	NetDuration
0	133476254	0:10:14
1	133281846	0:22:16
2	133280780	0:19:23
3	133281834	0:10:11
4	133491249	0:09:59
...	...	...
3053	133323839	0:09:03
3054	133305818	0:06:57
3055	133260048	0:21:46
3056	133305807	0:06:50
3057	133352713	0:09:20

3058 rows × 2 columns

[33]:
```
#wybor kolumn od SbjNum do InterviewTimeVEnd
df.loc[:, "SbjNum":"InterviewTimeVEnd"]
```

[33]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd
0	133476254	0:10:14	7/17/2020 13:53	7/17/2020 14:26
1	133281846	0:22:16	7/10/2020 12:53	7/10/2020 14:47
2	133280780	0:19:23	7/10/2020 12:35	7/10/2020 12:54
3	133281834	0:10:11	7/10/2020 10:21	7/10/2020 10:32
4	133491249	0:09:59	7/18/2020 8:27	7/18/2020 8:39
...	...	...	...	...
3053	133323839	0:09:03	7/11/2020 12:44	7/11/2020 12:53
3054	133305818	0:06:57	7/11/2020 16:18	7/11/2020 16:25
3055	133260048	0:21:46	7/9/2020 11:49	7/9/2020 12:12
3056	133305807	0:06:50	7/11/2020 9:05	7/11/2020 9:12
3057	133352713	0:09:20	7/13/2020 9:56	7/13/2020 14:44

3058 rows × 4 columns

[35]:
```
#wybor kolumn od SbjNum do InterviewTimeVEnd oraz ograniczenie wiersze od 10 do 15
df.loc[10:15, "SbjNum":"InterviewTimeVEnd"]
```

[35]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd
10	133282746	0:15:03	7/10/2020 11:09	7/10/2020 11:26
11	133508499	0:07:16	7/18/2020 16:20	7/18/2020 16:27
12	133367630	0:15:19	7/14/2020 11:56	7/14/2020 12:12
13	133555121	0:12:55	7/20/2020 12:15	7/20/2020 12:58
14	133337639	0:10:22	7/12/2020 20:57	7/12/2020 21:15
15	133521046	0:24:57	7/19/2020 11:25	7/19/2020 12:38

[37]:
```
#wybor kolumn od SbjNum do InterviewTimeVEnd oraz ograniczenie wiersze od 10 do 15 indexami
df.iloc[10:15, 0:4]
```

[37]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd
10	133282746	0:15:03	7/10/2020 11:09	7/10/2020 11:26
11	133508499	0:07:16	7/18/2020 16:20	7/18/2020 16:27
12	133367630	0:15:19	7/14/2020 11:56	7/14/2020 12:12
13	133555121	0:12:55	7/20/2020 12:15	7/20/2020 12:58
14	133337639	0:10:22	7/12/2020 20:57	7/12/2020 21:15

[39]:
```
#przedstawic wybor wierszy z ramki danych pod warunkiem odnosnie
#okreslonej wartosci kolumny

df[df["Country"] == 2]
```

[39]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	...	R11	R12	H1	H6	H11	FinalOutcome
0	133476254	0:10:14	7/17/2020 13:53	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	XXXX	...	80000.0	1	1	1	1	1
1	133281846	0:22:16	7/10/2020 12:53	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	XXXX	...	2000.0	1	2	2	1	1
2	133280780	0:19:23	7/10/2020 12:35	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	XXXX	...	43000.0	1	1	2	1	1
3	133281834	0:10:11	7/10/2020 10:21	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	XXXX	...	10000.0	1	2	1	1	1
4	133491249	0:09:59	7/18/2020 8:27	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	XXXX	...	15000.0	1	2	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1011	133350222	0:18:10	7/13/2020 11:03	7/13/2020 11:24	7/13/2020 6:03	3204	2	3	11	XXXX	...	3000.0	1	1	2	2	1
1012	133325893	0:18:36	7/12/2020 10:08	7/12/2020 10:30	7/12/2020 5:08	3204	2	3	11	XXXX	...	15000.0	1	1	2	2	1
1013	133325270	0:20:53	7/12/2020 12:02	7/12/2020 12:40	7/12/2020 7:02	3205	2	3	11	XXXX	...	27000.0	1	1	2	2	1
1014	133521045	0:21:34	7/19/2020 11:01	7/19/2020 12:57	7/19/2020 6:01	3205	2	1	11	XXXX	...	7000.0	1	2	2	2	1
1015	133490330	0:41:25	7/18/2020 13:09	7/18/2020 14:01	7/18/2020 8:09	3202	2	1	10	XXXX	...	10000.0	1	2	2	2	1

1016 rows × 27 columns

[41]:
```
#przedstawi´c wyb´or wierszy z ramki danych pod warunkiem spełnienia
#kilku warunk´ow jednocze´snie

final_df = df[(df["Country"] == 2)&(df["R1"] == 11)&(df["SbjNum"] > 133521043)]
final_df
```

[41]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	...	R11	R12	H1	H6	H11	FinalOutcome
15	133521046	0:24:57	7/19/2020 11:25	7/19/2020 12:38	7/19/2020 6:25	3205	2	3	11	XXXX	...	15000.0	1	1	2	1	1
44	133617209	0:14:32	7/23/2020 9:25	7/23/2020 9:43	7/23/2020 4:25	3204	2	3	11	XXXX	...	5000.0	1	2	1	1	1
79	133521684	0:11:55	7/19/2020 17:34	7/19/2020 17:48	7/19/2020 12:34	3225	2	3	11	XXXX	...	0.0	1	2	2	1	1
140	133521683	0:08:25	7/19/2020 17:05	7/19/2020 17:22	7/19/2020 12:05	3225	2	3	11	XXXX	...	0.0	1	2	2	2	1
143	133521054	0:10:43	7/19/2020 14:59	7/19/2020 15:09	7/19/2020 9:59	3205	2	1	11	XXXX	...	10000.0	1	2	99	2	1
145	133521050	0:11:29	7/19/2020 13:21	7/19/2020 13:32	7/19/2020 8:21	3205	2	3	11	XXXX	...	28000.0	1	2	2	2	1
155	133559779	0:12:42	7/21/2020 10:47	7/21/2020 10:59	7/21/2020 5:47	3205	2	3	11	XXXX	...	5000.0	1	2	2	2	1
237	133521048	0:10:09	7/19/2020 12:42	7/19/2020 12:52	7/19/2020 7:42	3205	2	3	11	XXXX	...	10000.0	1	2	2	2	1
310	133528919	0:07:03	7/19/2020 20:08	7/19/2020 20:19	7/19/2020 15:08	3225	2	3	11	XXXX	...	20000.0	1	2	2	2	1
312	133615526	0:08:31	7/21/2020 12:38	7/21/2020 12:54	7/21/2020 7:38	3233	2	1	11	XXXX	...	20000.0	1	1	2	1	1
370	133528916	0:13:41	7/19/2020 19:21	7/19/2020 19:37	7/19/2020 14:21	3225	2	3	11	XXXX	...	0.0	1	2	2	1	1
443	133617208	0:14:02	7/23/2020 9:02	7/23/2020 9:20	7/23/2020 4:02	3204	2	3	11	XXXX	...	10000.0	1	2	2	1	1
465	133556251	0:16:22	7/21/2020 10:10	7/21/2020 10:27	7/21/2020 5:10	3205	2	3	11	XXXX	...	3000.0	1	1	2	2	1
477	133559778	0:14:39	7/21/2020 10:28	7/21/2020 10:42	7/21/2020 5:28	3205	2	3	11	XXXX	...	9000.0	1	1	2	2	1
631	133617210	0:13:33	7/23/2020 9:50	7/23/2020 10:07	7/23/2020 4:50	3204	2	3	11	XXXX	...	11000.0	1	1	1	1	1
637	133627182	0:14:45	7/23/2020 16:11	7/23/2020 16:33	7/23/2020 11:11	3223	2	3	11	XXXX	...	99.0	1	2	1	2	1
638	133532122	0:15:23	7/20/2020 11:42	7/20/2020 12:00	7/20/2020 6:42	3204	2	3	11	XXXX	...	70000.0	1	2	1	1	1
645	133521053	0:16:21	7/19/2020 14:37	7/19/2020 14:56	7/19/2020 9:37	3205	2	1	11	XXXX	...	2000.0	1	2	2	2	1
653	133521044	0:09:47	7/19/2020 10:32	7/19/2020 11:00	7/19/2020 5:32	3205	2	3	11	XXXX	...	5000.0	1	1	2	1	1
672	133528915	0:10:34	7/19/2020 18:57	7/19/2020 19:15	7/19/2020 13:57	3225	2	3	11	XXXX	...	5000.0	1	2	1	1	1
736	133528917	0:06:42	7/19/2020 19:45	7/19/2020 19:55	7/19/2020 14:45	3225	2	3	11	XXXX	...	40000.0	1	2	1	1	1
964	133533796	0:05:49	7/19/2020 18:37	7/19/2020 18:48	7/19/2020 13:37	3225	2	3	11	XXXX	...	0.0	1	2	2	2	1
965	133528918	0:05:59	7/19/2020 19:53	7/19/2020 20:02	7/19/2020 14:53	3225	2	3	11	XXXX	...	25000.0	1	2	2	2	1
969	133521049	0:08:24	7/19/2020 12:58	7/19/2020 13:06	7/19/2020 7:58	3205	2	3	11	XXXX	...	2000.0	1	2	2	2	1
991	133627181	0:11:08	7/23/2020 15:42	7/23/2020 15:58	7/23/2020 10:42	3223	2	3	11	XXXX	...	0.0	1	2	2	2	1
1002	133627180	0:15:44	7/23/2020 10:55	7/23/2020 11:14	7/23/2020 5:55	3223	2	3	11	XXXX	...	0.0	1	1	1	1	1
1014	133521045	0:21:34	7/19/2020 11:01	7/19/2020 12:57	7/19/2020 6:01	3205	2	1	11	XXXX	...	7000.0	1	2	2	2	1

27 rows × 27 columns

[43]:
```
# wybrac wiersze ktore zawieraja w kolumnie kategoryzowanej okreslone slowo

df[df["InterviewTimeVStart"].str.contains("7/10/2020")]
```

[43]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	...	R11	R12	H1	H6	H11	FinalOutcome
1	133281846	0:22:16	7/10/2020 12:53	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	XXXX	...	2000.0	1	2	2	1	1
2	133280780	0:19:23	7/10/2020 12:35	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	XXXX	...	43000.0	1	1	2	1	1
3	133281834	0:10:11	7/10/2020 10:21	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	XXXX	...	10000.0	1	2	1	1	1
10	133282746	0:15:03	7/10/2020 11:09	7/10/2020 11:26	7/10/2020 6:09	3214	2	1	13	XXXX	...	99.0	1	2	1	1	1
58	133282912	0:21:46	7/10/2020 9:39	7/10/2020 10:03	7/10/2020 4:39	3216	2	1	13	XXXX	...	99.0	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3010	133283791	0:10:23	7/10/2020 16:22	7/10/2020 18:08	7/10/2020 9:22	3012	1	7	8	XXXX	...	37000.0	1	1	2	1	1
3018	133275037	0:13:01	7/10/2020 12:05	7/10/2020 12:18	7/10/2020 5:05	3002	1	1	6	XXXX	...	60000.0	2	2	2	2	1
3033	133302364	0:13:49	7/10/2020 12:32	7/11/2020 10:08	7/10/2020 5:32	3002	1	1	6	XXXX	...	150000.0	1	1	2	2	1
3034	133296327	0:14:52	7/10/2020 10:41	7/10/2020 10:56	7/10/2020 3:41	3006	1	1	1	XXXX	...	200.0	2	1	2	2	1
3045	133290062	0:07:22	7/10/2020 12:18	7/10/2020 12:26	7/10/2020 5:18	3008	1	1	2	XXXX	...	8000.0	1	2	2	1	1

264 rows × 27 columns

[45]:
```
# wybrac wiersze ktore nie zawieraja w kolumnie kategoryzowanej okreslone slowo

df[df["InterviewTimeVStart"].str.contains("7/10/2020") == False]
```

[45]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	...	R11	R12	H1	H6	H11	FinalOutcome
0	133476254	0:10:14	7/17/2020 13:53	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	XXXX	...	80000.0	1	1	1	1	1
4	133491249	0:09:59	7/18/2020 8:27	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	XXXX	...	15000.0	1	2	1	1	1
5	133309774	0:17:14	7/11/2020 11:36	7/11/2020 11:54	7/11/2020 6:36	3233	2	1	11	XXXX	...	15000.0	1	2	2	1	1
6	133520640	0:09:09	7/19/2020 13:20	7/19/2020 13:29	7/19/2020 8:20	3240	2	1	9	XXXX	...	10000.0	1	1	2	1	1
7	133219300	0:12:21	7/8/2020 11:48	7/8/2020 12:05	7/8/2020 6:48	3214	2	1	14	XXXX	...	70000.0	1	2	2	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3053	133323839	0:09:03	7/11/2020 12:44	7/11/2020 12:53	7/11/2020 5:44	3012	1	7	8	XXXX	...	8000.0	1	1	1	2	1
3054	133305818	0:06:57	7/11/2020 16:18	7/11/2020 16:25	7/11/2020 9:18	3008	1	1	3	XXXX	...	12000.0	1	1	2	1	1
3055	133260048	0:21:46	7/9/2020 11:49	7/9/2020 12:12	7/9/2020 4:49	3004	1	1	7	XXXX	...	5000.0	1	1	2	1	1
3056	133305807	0:06:50	7/11/2020 9:05	7/11/2020 9:12	7/11/2020 2:05	3008	1	1	3	XXXX	...	0.0	1	2	2	2	1
3057	133352713	0:09:20	7/13/2020 9:56	7/13/2020 14:44	7/13/2020 2:56	3003	1	1	2	XXXX	...	99.0	2	1	2	1	1

2794 rows × 27 columns

[47]:
```
#utworz kolumne na podstawie istniejacej

df["new_column"] = df["Country"] - df["LANG"]
print(df[["Country", "LANG", "new_column"]])
```

```
      Country  LANG  new_column
0           2     1           1
1           2     4          -2
2           2     3          -1
3           2     1           1
4           2     3          -1
...       ...   ...         ...
3053        1     7          -6
3054        1     1           0
3055        1     1           0
3056        1     1           0
3057        1     1           0

[3058 rows x 3 columns]
```

[49]:
```
#usun kolumne

df = df.drop("InterviewTimeVStart", axis = 1)
df
```

[49]:

	SbjNum	NetDuration	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	R4	...	R12	H1	H6	H11	FinalOutcome	NumOfVisits	weight_com
0	133476254	0:10:14	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	XXXX	1	...	1	1	1	1	1	1	0.8
1	133281846	0:22:16	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	XXXX	2	...	1	2	2	1	1	1	1.4
2	133280780	0:19:23	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	XXXX	1	...	1	1	2	1	1	1	0.8
3	133281834	0:10:11	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	XXXX	2	...	1	2	1	1	1	1	1.4
4	133491249	0:09:59	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	XXXX	1	...	1	2	1	1	1	1	0.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3053	133323839	0:09:03	7/11/2020 12:53	7/11/2020 5:44	3012	1	7	8	XXXX	1	...	1	1	1	2	1	1	3.7
3054	133305818	0:06:57	7/11/2020 16:25	7/11/2020 9:18	3008	1	1	3	XXXX	1	...	1	1	2	1	1	1	1.1
3055	133260048	0:21:46	7/9/2020 12:12	7/9/2020 4:49	3004	1	1	7	XXXX	2	...	1	1	2	1	1	1	0.7
3056	133305807	0:06:50	7/11/2020 9:12	7/11/2020 2:05	3008	1	1	3	XXXX	1	...	1	2	2	2	1	1	0.7
3057	133352713	0:09:20	7/13/2020 14:44	7/13/2020 2:56	3003	1	1	2	XXXX	1	...	2	1	2	1	1	3	1.1

3058 rows × 27 columns

[51]:
```
#zmien nazwe kolumny

df = df.rename(columns = {"SbjNum":"new_name"})
df
```

[51]:

	new_name	NetDuration	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	R4	...	R12	H1	H6	H11	FinalOutcome	NumOfVisits	weight_con
0	133476254	0:10:14	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	XXXX	1	...	1	1	1	1	1	1	0.
1	133281846	0:22:16	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	XXXX	2	...	1	2	2	1	1	1	1.
2	133280780	0:19:23	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	XXXX	1	...	1	1	2	1	1	1	0.
3	133281834	0:10:11	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	XXXX	2	...	1	2	1	1	1	1	1.
4	133491249	0:09:59	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	XXXX	1	...	1	2	1	1	1	1	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3053	133323839	0:09:03	7/11/2020 12:53	7/11/2020 5:44	3012	1	7	8	XXXX	1	...	1	1	1	2	1	1	3.
3054	133305818	0:06:57	7/11/2020 16:25	7/11/2020 9:18	3008	1	1	3	XXXX	1	...	1	1	2	1	1	1	1.
3055	133260048	0:21:46	7/9/2020 12:12	7/9/2020 4:49	3004	1	1	7	XXXX	2	...	1	1	2	1	1	1	0.
3056	133305807	0:06:50	7/11/2020 9:12	7/11/2020 2:05	3008	1	1	3	XXXX	1	...	1	2	2	2	1	1	0.
3057	133352713	0:09:20	7/13/2020 14:44	7/13/2020 2:56	3003	1	1	2	XXXX	1	...	2	1	2	1	1	3	1.

3058 rows × 27 columns

[53]:
```
#zachowaj ramke danych jako plik csv na komputerze

df.to_csv("F.csv")
```

[55]:
```
#wyswietlic srednia (maksymalna, minimalna) wartosc z jednej kolumny

print(df["Country"].mean())
print(df["Country"].max())
print(df["Country"].min())
```

```
2.012426422498365
3
1
```

[57]:
```
#wyswietlic liczbe wierszy

rows = len(df.axes[0])
rows
```

[57]:
```
3058
```

[59]:
```
#wyswietlic wartosci unikatowe w kolumnie

df['Country'].unique()
```

[59]:
```
array([2, 1, 3], dtype=int64)
```

[61]:
```
#wyswietlic liczby rekordow odpowiadajacych do wartosci

df['Country'].value_counts()
```

[61]:
```
Country
3    1040
2    1016
1    1002
Name: count, dtype: int64
```

[63]:
```
#sortowanie wierszy ramki danych wedlug wartosci okreslonej kolumny
#(malejaco, rosnaco)

df.sort_values(['R1'], ascending = True) # sortowanie rosnaco
```

[63]:

	new_name	NetDuration	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	R4	...	R12	H1	H6	H11	FinalOutcome	NumOfVisits	weight_com
2646	133216690	0:16:22	7/8/2020 14:24	7/8/2020 5:03	3007	1	1	1	XXXX	2	...	1	1	2	2	1	2	0.7
2768	133352717	0:11:51	7/13/2020 11:22	7/13/2020 3:42	3003	1	7	1	XXXX	1	...	1	1	1	1	1	1	0.7
2949	133380015	0:06:55	7/14/2020 15:23	7/14/2020 8:16	3001	1	1	1	XXXX	2	...	1	2	2	2	1	1	0.7
2770	133198153	0:11:11	7/7/2020 19:33	7/7/2020 12:18	3007	1	7	1	XXXX	2	...	2	1	1	1	1	1	0.7
2772	133198457	0:13:46	7/7/2020 17:10	7/7/2020 9:26	3001	1	1	1	XXXX	2	...	1	2	2	2	1	1	0.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1379	133238505	0:07:31	7/9/2020 11:24	7/9/2020 5:17	3024	3	1	54	XXXX	2	...	1	1	2	2	1	1	1.0
1717	133464158	0:10:34	7/17/2020 14:15	7/17/2020 8:04	3029	3	1	54	XXXX	2	...	2	2	2	1	1	1	1.0
1348	133518560	0:05:56	7/19/2020 19:08	7/19/2020 13:02	3021	3	1	54	XXXX	2	...	1	1	2	1	1	1	1.0
2035	133244583	0:05:46	7/9/2020 15:01	7/9/2020 8:56	3022	3	1	54	XXXX	1	...	2	2	2	2	1	1	1.0
1716	133464159	0:09:54	7/17/2020 14:27	7/17/2020 8:17	3029	3	1	54	XXXX	2	...	2	2	1	2	1	1	1.0

3058 rows × 27 columns

[65]:
```
df.sort_values(['R1'], ascending =  False) # sortowanie malejaco
```

[65]:

	new_name	NetDuration	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	R4	...	R12	H1	H6	H11	FinalOutcome	NumOfVisits	weight_com
2063	133238732	0:06:12	7/9/2020 11:31	7/9/2020 5:25	3024	3	1	54	XXXX	1	...	2	2	2	2	1	1	1.0
1556	133568903	0:05:55	7/21/2020 19:15	7/21/2020 13:09	3029	3	1	54	XXXX	2	...	2	2	1	2	1	1	1.0
1813	133422463	0:07:03	7/16/2020 11:35	7/16/2020 5:28	3029	3	1	54	XXXX	2	...	1	2	2	1	1	1	1.0
1346	133568620	0:07:19	7/21/2020 19:03	7/21/2020 12:56	3029	3	1	54	XXXX	2	...	2	1	2	2	1	1	1.0
1342	133300438	0:09:59	7/11/2020 12:04	7/11/2020 5:54	3021	3	1	54	XXXX	1	...	1	1	1	2	1	1	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2829	133193703	0:11:36	7/7/2020 19:16	7/7/2020 11:57	3011	1	7	1	XXXX	2	...	1	1	2	2	1	1	0.7
2827	133259522	0:17:00	7/9/2020 10:23	7/8/2020 6:36	3003	1	1	1	XXXX	1	...	1	1	1	1	1	3	0.7
2957	133379605	0:09:19	7/14/2020 12:17	7/14/2020 5:07	3003	1	7	1	XXXX	1	...	1	2	2	2	1	1	0.7
2562	133347600	0:18:46	7/13/2020 15:41	7/13/2020 5:32	3001	1	1	1	XXXX	2	...	1	1	1	1	1	1	0.7
2743	133326526	0:09:20	7/12/2020 14:29	7/12/2020 7:19	3010	1	7	1	XXXX	1	...	2	1	2	2	1	1	0.7

3058 rows × 27 columns

[67]:
```
#wyswietlic wierszy dla 10 najwiekszych (najmniejszych) wartosci okreslonej
#kolumny
df.sort_values(['Srvyr'], ascending =  True).head(10)
```

[67]:

	new_name	NetDuration	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	R4	...	R12	H1	H6	H11	FinalOutcome	NumOfVisits	weight_com
2772	133198457	0:13:46	7/7/2020 17:10	7/7/2020 9:26	3001	1	1	1	XXXX	2	...	1	2	2	2	1	1	0.7
2403	133332015	0:07:24	7/12/2020 12:12	7/12/2020 5:04	3001	1	1	1	XXXX	1	...	1	2	2	2	1	1	0.7
2615	133442400	0:09:51	7/16/2020 16:02	7/16/2020 8:52	3001	1	1	7	XXXX	1	...	1	2	2	2	1	1	0.7
2617	133256525	0:11:39	7/9/2020 9:38	7/9/2020 2:27	3001	1	1	7	XXXX	2	...	1	2	2	2	1	1	0.7
2399	133279344	0:16:54	7/10/2020 14:46	7/10/2020 4:54	3001	1	1	7	XXXX	1	...	1	1	2	2	1	1	0.7
2393	133279345	0:10:08	7/10/2020 13:21	7/10/2020 5:03	3001	1	1	3	XXXX	1	...	1	2	2	1	1	1	0.7
2919	133595179	0:11:24	7/22/2020 16:10	7/22/2020 8:58	3001	1	1	1	XXXX	2	...	1	2	2	2	1	1	0.7
2392	133492671	0:09:25	7/17/2020 13:40	7/17/2020 6:31	3001	1	1	7	XXXX	1	...	1	2	2	2	1	1	0.7
2609	133558507	0:11:43	7/21/2020 14:20	7/21/2020 5:46	3001	1	1	1	XXXX	2	...	1	1	2	1	1	1	1.1
2917	133332016	0:09:26	7/12/2020 15:20	7/12/2020 5:17	3001	1	1	6	XXXX	2	...	1	2	2	2	1	1	0.7

10 rows × 27 columns

[69]:
```
df.sort_values(['Srvyr'], ascending =  False).head(10)
```

[69]:

	new_name	NetDuration	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	R4	...	R12	H1	H6	H11	FinalOutcome	NumOfVisits	weight_comb
460	133603974	0:14:42	7/22/2020 20:12	7/22/2020 14:57	3264	2	1	13	XXXX	2	...	1	2	1	2	1	1	0.87
530	133251451	0:11:49	7/9/2020 14:13	7/9/2020 9:01	3264	2	1	13	XXXX	2	...	1	2	1	1	1	1	0.87
531	133603975	0:12:01	7/22/2020 20:28	7/22/2020 15:16	3264	2	1	13	XXXX	2	...	1	2	2	2	1	1	0.87
51	133262795	0:22:17	7/9/2020 21:08	7/9/2020 15:45	3264	2	1	13	XXXX	2	...	2	2	1	2	1	1	0.87
888	133617615	0:13:28	7/23/2020 9:17	7/23/2020 4:03	3264	2	1	14	XXXX	2	...	1	2	2	1	1	1	0.87
884	133309082	0:12:21	7/11/2020 12:37	7/11/2020 7:11	3263	2	1	14	XXXX	1	...	1	2	2	2	1	1	0.82
275	133309079	0:24:18	7/11/2020 9:53	7/11/2020 4:29	3263	2	1	14	XXXX	2	...	1	1	2	2	1	1	0.87
340	133282813	0:17:57	7/10/2020 9:49	7/9/2020 8:11	3263	2	1	13	XXXX	1	...	1	1	2	1	1	2	0.82
134	133254937	0:17:03	7/9/2020 14:23	7/9/2020 8:57	3263	2	1	12	XXXX	1	...	1	2	2	1	1	2	0.82
754	133282820	0:15:52	7/10/2020 11:45	7/10/2020 6:28	3263	2	1	13	XXXX	1	...	1	1	1	1	1	1	0.82

10 rows × 27 columns

[71]:
```
#wyswietlic wierszy dla 10 najwiekszych wartosci okreslonej kolumny
#pod warunkiem okreslonych wartosci innej kolumny
df[(df['Srvyr'].isin([3264]))].nlargest(10,'new_name')
```

[71]:

	new_name	NetDuration	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R3	R4	...	R12	H1	H6	H11	FinalOutcome	NumOfVisits	weight_com
888	133617615	0:13:28	7/23/2020 9:17	7/23/2020 4:03	3264	2	1	14	XXXX	2	...	1	2	2	1	1	1	0.8
531	133603975	0:12:01	7/22/2020 20:28	7/22/2020 15:16	3264	2	1	13	XXXX	2	...	1	2	2	2	1	1	0.8
460	133603974	0:14:42	7/22/2020 20:12	7/22/2020 14:57	3264	2	1	13	XXXX	2	...	1	2	1	2	1	1	0.8
51	133262795	0:22:17	7/9/2020 21:08	7/9/2020 15:45	3264	2	1	13	XXXX	2	...	2	2	1	2	1	1	0.8
530	133251451	0:11:49	7/9/2020 14:13	7/9/2020 9:01	3264	2	1	13	XXXX	2	...	1	2	1	1	1	1	0.8

5 rows × 27 columns

[73]:
```
#grupowanie wierszy wedlug wartosci kolumny kategoryzowanej, potem
#- usrednienie wartosci wszystkich kolumn w grupie - MultiIndex

df_3 = df.groupby(['new_name','Country']).agg({'weight_combined': 'mean',
                                                'agegroup': 'mean',
                                                'gk_weight': 'mean'})
df_3
```

[73]:

		weight_combined	agegroup	gk_weight
new_name	**Country**			
133171538	**2**	0.829860	1.0	2.054230
133172154	**2**	1.416946	2.0	1.949579
133172253	**2**	2.722043	3.0	2.479905
133183877	**3**	1.000000	1.0	2.987726
133184260	**3**	1.000000	1.0	3.655632
...	**...**	...	...	...
133645011	**3**	1.000000	1.0	2.325065
133645012	**3**	1.000000	1.0	2.912190
133645013	**3**	1.000000	1.0	2.912190
133645016	**3**	1.000000	1.0	2.325065
133645018	**3**	1.000000	1.0	2.912190

3058 rows × 3 columns

[75]:
```
#grupowanie wierszy wedlug wartosci kolumny kategoryzowanej, potem
#- usrednienie wartosci dla pewnych kolumn, liczba wartosci i mediana
#dla pozostalych kolumn w grupach

df_4 = df.groupby(['new_name','Country']).agg({
                        'weight_combined': 'mean',
                        'agegroup': ['median','count'],
                        'gk_weight' : ['median','count']})
df_4
```

[75]:

		weight_combined	agegroup		gk_weight	
		mean	median	count	median	count
new_name	Country					
133171538	2	0.829860	1.0	1	2.054230	1
133172154	2	1.416946	2.0	1	1.949579	1
133172253	2	2.722043	3.0	1	2.479905	1
133183877	3	1.000000	1.0	1	2.987726	1
133184260	3	1.000000	1.0	1	3.655632	1
...	...	...	...	...	...	...
133645011	3	1.000000	1.0	1	2.325065	1
133645012	3	1.000000	1.0	1	2.912190	1
133645013	3	1.000000	1.0	1	2.912190	1
133645016	3	1.000000	1.0	1	2.325065	1
133645018	3	1.000000	1.0	1	2.912190	1

3058 rows × 5 columns

[77]:
```
#wyswietlic nazwy kolumn indeksu zlozonego

df_4.columns
```

[77]:
```
MultiIndex([('weight_combined',   'mean'),
            (       'agegroup', 'median'),
            (       'agegroup',  'count'),
            (      'gk_weight', 'median'),
            (      'gk_weight',  'count')],
           )
```

[79]:
```
#sortowac kolumne indeksu zlozonego

df_4['gk_weight']['median'].sort_values(ascending = True)
```

[79]:
```
new_name   Country
133476254  2          1.555754
133402438  2          1.555754
133301837  2          1.555754
133264434  2          1.555754
133224163  2          1.555754
                        ...
133525201  2          7.110619
133438178  2          7.110619
133365835  2          7.110619
133621876  2          7.110619
133350222  2          7.110619
Name: median, Length: 3058, dtype: float64
```

[81]:
```
#stworzyc tabele przystawna (pivot table) na podstawie ramki danych

df_pivot = df.pivot_table(values='weight_combined', index='new_name', columns='Country', aggfunc='mean',
                          margins=False, dropna=True, fill_value=None) # tabela podsumowujaca
df_pivot
```

[81]:

Country	1	2	3
new_name			
133171538	NaN	0.829860	NaN
133172154	NaN	1.416946	NaN
133172253	NaN	2.722043	NaN
133183877	NaN	NaN	1.0
133184260	NaN	NaN	1.0
...	...	...	...
133645011	NaN	NaN	1.0
133645012	NaN	NaN	1.0
133645013	NaN	NaN	1.0
133645016	NaN	NaN	1.0
133645018	NaN	NaN	1.0

3058 rows × 3 columns

[83]:
```
#wyswietlic indeksy i kolumny tabeli przystawnej

print(df_pivot.index)
print(df_pivot.columns)
```

```
Index([133171538, 133172154, 133172253, 133183877, 133184260, 133184356,
       133184416, 133184425, 133184636, 133184864,
       ...
       133645006, 133645007, 133645008, 133645009, 133645010, 133645011,
       133645012, 133645013, 133645016, 133645018],
      dtype='int64', name='new_name', length=3058)
Index([1, 2, 3], dtype='int64', name='Country')
```

[85]:
```
#utworz indeks zlozony tabeli przystawnej i wyswieti go

df_pivot = df.pivot_table(values='new_name', index=['LANG','R1'], columns='Country', aggfunc='mean',
                          margins=False, dropna=True, fill_value=None)
df_pivot
```

[85]:

	Country	1	2	3
LANG	R1			
1	1	1.333413e+08	NaN	NaN
	2	1.333515e+08	NaN	NaN
	3	1.333371e+08	NaN	NaN
	4	1.333337e+08	NaN	NaN
	5	1.333749e+08	NaN	NaN
...	...	...	...	...
10	52	NaN	NaN	133274171.5
	54	NaN	NaN	133626309.0
11	49	NaN	NaN	133318968.0
	53	NaN	NaN	133563096.0
	54	NaN	NaN	133588302.0

64 rows × 3 columns

[87]:
```
#zaimportuj modul pyplot z biblioteki matplotlib
import matplotlib.pyplot as plt
```

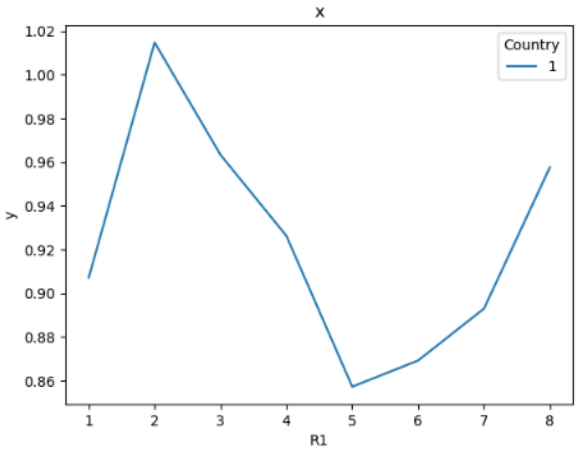
[88]:
```
#wskazac, ze wykresy nalezy rysowac bezposrednio w zeszycie a nie w
#osobnej zakladce
%matplotlib inline
```

[91]:
```
#wyswietlic wykres na podstawie tabeli przystawnej

df[(df['LANG'] == 1) & (df['Country'] == 1)].pivot_table(values='weight_combined', index='R1', columns='Country', aggfunc='mean',
                                        fill_value=None, margins=False, dropna=True).plot(kind = 'line')

plt.ylabel('y')
plt.title('x')
```

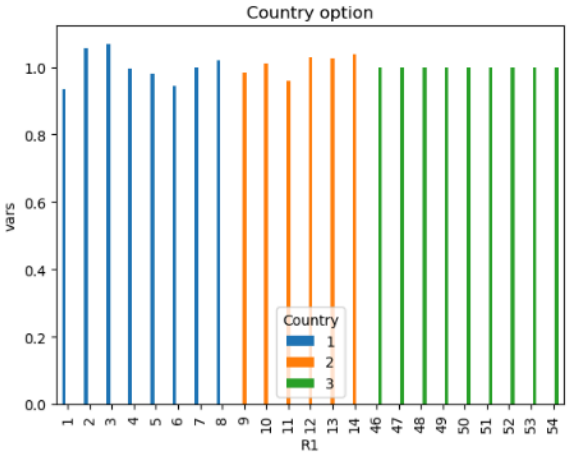
[91]:
```
Text(0.5, 1.0, 'x')
```



[93]:
```
#narysowac histogram na podstawie wartosci kolumny

df_bar = df[(df['Country'].isin([1,2,3]))].pivot_table(values='weight_combined',
               index='R1', columns='Country', aggfunc='mean',
                     fill_value=None, margins=False, dropna=True)
df_bar.plot(kind = 'bar')
plt.ylabel('vars')
plt.title('Country option')
```

[93]:
```
Text(0.5, 1.0, 'Country option')
```



[95]:
```
#przedstawic sposoby laczenia ramek danych za pomoca metod merge i
#concat

df2 = pd.read_csv('IHME_ORB_C19HSDS_2020_Y2020M12D03.CSV')
df2
```

[95]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R1_5	...	G11_Other	G11_99	FinalOutcome	NumOf
0	133476254	0:10:14	7/17/2020 13:53	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	15.0	...	NaN	NaN	1	
1	133281846	0:22:16	7/10/2020 12:53	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	22.0	...	NaN	NaN	1	
2	133280780	0:19:23	7/10/2020 12:35	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	13.0	...	NaN	NaN	1	
3	133281834	0:10:11	7/10/2020 10:21	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	9.0	...	NaN	NaN	1	
4	133491249	0:09:59	7/18/2020 8:27	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	28.0	...	NaN	NaN	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3053	133323839	0:09:03	7/11/2020 12:44	7/11/2020 12:53	7/11/2020 5:44	3012	1	7	8	NaN	...	NaN	NaN	1	
3054	133305818	0:06:57	7/11/2020 16:18	7/11/2020 16:25	7/11/2020 9:18	3008	1	1	3	NaN	...	NaN	NaN	1	
3055	133260048	0:21:46	7/9/2020 11:49	7/9/2020 12:12	7/9/2020 4:49	3004	1	1	7	NaN	...	NaN	NaN	1	
3056	133305807	0:06:50	7/11/2020 9:05	7/11/2020 9:12	7/11/2020 2:05	3008	1	1	3	NaN	...	NaN	NaN	1	
3057	133352713	0:09:20	7/13/2020 9:56	7/13/2020 14:44	7/13/2020 2:56	3003	1	1	2	NaN	...	NaN	NaN	1	

3058 rows × 247 columns

[97]:
```
#przedstawic sposoby laczenia ramek danych za pomoca metod merge i
#concat

dfm_1 = df2[['SbjNum', 'NetDuration', 'Date', 'Srvyr']]
dfm_2 = df2[['SbjNum', 'Country', 'LANG', 'R1']]
merged_df = pd.merge(dfm_1, dfm_2, on='SbjNum', how='inner')
print(merged_df)
```

```
         SbjNum NetDuration            Date  Srvyr  Country  LANG  R1
0     133476254     0:10:14  7/17/2020 8:53   3232        2     1   9
1     133281846     0:22:16  7/10/2020 7:53   3206        2     4  12
2     133280780     0:19:23  7/10/2020 7:35   3202        2     3  10
3     133281834     0:10:11  7/10/2020 5:21   3212        2     1  12
4     133491249     0:09:59  7/18/2020 3:27   3225        2     3  11
...         ...         ...             ...    ...      ...   ...  ..
3053  133323839     0:09:03  7/11/2020 5:44   3012        1     7   8
3054  133305818     0:06:57  7/11/2020 9:18   3008        1     1   3
3055  133260048     0:21:46   7/9/2020 4:49   3004        1     1   7
3056  133305807     0:06:50  7/11/2020 2:05   3008        1     1   3
3057  133352713     0:09:20  7/13/2020 2:56   3003        1     1   2

[3058 rows x 7 columns]
```

[97]:
```
#przedstawic sposoby laczenia ramek danych za pomoca metod merge i
#concat

dfm_1 = df2[['SbjNum', 'NetDuration', 'Date', 'Srvyr']]
dfm_2 = df2[['SbjNum', 'Country', 'LANG', 'R1']]
merged_df = pd.merge(dfm_1, dfm_2, on='SbjNum', how='inner')
print(merged_df)
```

```
      SbjNum NetDuration            Date  Srvyr  Country  LANG  R1
0  133476254     0:10:14  7/17/2020 8:53   3232        2     1   9
1  133281846     0:22:16  7/10/2020 7:53   3206        2     4  12
2  133280780     0:19:23  7/10/2020 7:35   3202        2     3  10
3  133281834     0:10:11  7/10/2020 5:21   3212        2     1  12
4  133491249     0:09:59  7/18/2020 3:27   3225        2     3  11
...      ...         ...             ...    ...      ...   ...  ..
3053  133323839     0:09:03  7/11/2020 5:44   3012        1     7   8
3054  133305818     0:06:57  7/11/2020 9:18   3008        1     1   3
3055  133260048     0:21:46   7/9/2020 4:49   3004        1     1   7
3056  133305807     0:06:50  7/11/2020 2:05   3008        1     1   3
3057  133352713     0:09:20  7/13/2020 2:56   3003        1     1   2

[3058 rows x 7 columns]
```

[99]:
```
dfm_1 = dfm_1.reindex(columns=['SbjNum', 'NetDuration', 'Date', 'Srvyr', 'Country', 'LANG', 'R1'])
combined_df_rows = pd.concat([dfm_1, dfm_2], axis=0, ignore_index=True)
print(combined_df_rows)
```

```
      SbjNum NetDuration            Date   Srvyr  Country  LANG   R1
0  133476254     0:10:14  7/17/2020 8:53  3232.0      NaN   NaN  NaN
1  133281846     0:22:16  7/10/2020 7:53  3206.0      NaN   NaN  NaN
2  133280780     0:19:23  7/10/2020 7:35  3202.0      NaN   NaN  NaN
3  133281834     0:10:11  7/10/2020 5:21  3212.0      NaN   NaN  NaN
4  133491249     0:09:59  7/18/2020 3:27  3225.0      NaN   NaN  NaN
...      ...         ...             ...     ...      ...   ...  ...
6111  133323839         NaN             NaN     NaN      1.0   7.0  8.0
6112  133305818         NaN             NaN     NaN      1.0   1.0  3.0
6113  133260048         NaN             NaN     NaN      1.0   1.0  7.0
6114  133305807         NaN             NaN     NaN      1.0   1.0  3.0
6115  133352713         NaN             NaN     NaN      1.0   1.0  2.0

[6116 rows x 7 columns]
```

[101]:
```
#pokazac dodawanie nowych kolumn za pomoca operacji matematycznych

df2['R1_plus_R1_5'] = df2['R1'] + df2['R1_5']
# Wyświetlenie DataFrame z nowymi kolumnami
print(df)
```

```
      new_name NetDuration InterviewTimeVEnd            Date  Srvyr  Country  \
0  133476254     0:10:14   7/17/2020 14:26  7/17/2020 8:53   3232        2
1  133281846     0:22:16   7/10/2020 14:47  7/10/2020 7:53   3206        2
2  133280780     0:19:23   7/10/2020 12:54  7/10/2020 7:35   3202        2
3  133281834     0:10:11   7/10/2020 10:32  7/10/2020 5:21   3212        2
4  133491249     0:09:59    7/18/2020 8:39  7/18/2020 3:27   3225        2
...      ...         ...               ...             ...    ...      ...
3053  133323839     0:09:03   7/11/2020 12:53  7/11/2020 5:44   3012        1
3054  133305818     0:06:57   7/11/2020 16:25  7/11/2020 9:18   3008        1
3055  133260048     0:21:46    7/9/2020 12:12   7/9/2020 4:49   3004        1
3056  133305807     0:06:50    7/11/2020 9:12  7/11/2020 2:05   3008        1
3057  133352713     0:09:20   7/13/2020 14:44  7/13/2020 2:56   3003        1

      LANG  R1    R3  R4  ...  R12  H1  H6  H11  FinalOutcome  NumOfVisits  \
0        1   9  XXXX   1  ...    1   1   1    1             1            1
1        4  12  XXXX   2  ...    1   2   2    1             1            1
2        3  10  XXXX   1  ...    1   1   2    1             1            1
3        1  12  XXXX   2  ...    1   2   1    1             1            1
4        3  11  XXXX   1  ...    1   2   1    1             1            1
...    ...  ..   ...  ..  ...  ...  ..  ..  ...           ...          ...
3053     7   8  XXXX   1  ...    1   1   1    2             1            1
3054     1   3  XXXX   1  ...    1   1   2    1             1            1
3055     1   7  XXXX   2  ...    1   1   2    1             1            1
3056     1   3  XXXX   1  ...    1   2   2    2             1            1
3057     1   2  XXXX   1  ...    2   1   2    1             1            3

      weight_combined  agegroup  gk_weight  new_column
0            0.829860         1   1.555754           1
1            1.416946         2   1.949579          -2
2            0.883601         2   2.151458          -1
3            1.416946         2   2.325065           1
4            0.829860         1   1.640484          -1
...               ...       ...        ...         ...
3053         3.791351         3   2.354356          -6
3054         1.157689         2   1.869021           0
3055         0.799916         1   1.907830           0
3056         0.799916         1   1.753344           0
3057         1.157689         2   1.869021           0

[3058 rows x 27 columns]
```

[103]:
```
#przedstawic na przyk ladzie dodawanie nowych kolumn z pomoca funkcji
#Lambda
df2['TotalScore'] = df2.apply(lambda x: x['R1'] + x['R1_5'], axis=1)
df2
```

[103]:

	SbjNum	NetDuration	InterviewTimeVStart	InterviewTimeVEnd	Date	Srvyr	Country	LANG	R1	R1_5	...	FinalOutcome	NumOfVisits	weight_combin
0	133476254	0:10:14	7/17/2020 13:53	7/17/2020 14:26	7/17/2020 8:53	3232	2	1	9	15.0	...	1	1	0.8296
1	133281846	0:22:16	7/10/2020 12:53	7/10/2020 14:47	7/10/2020 7:53	3206	2	4	12	22.0	...	1	1	1.4169
2	133280780	0:19:23	7/10/2020 12:35	7/10/2020 12:54	7/10/2020 7:35	3202	2	3	10	13.0	...	1	1	0.8836
3	133281834	0:10:11	7/10/2020 10:21	7/10/2020 10:32	7/10/2020 5:21	3212	2	1	12	9.0	...	1	1	1.4169
4	133491249	0:09:59	7/18/2020 8:27	7/18/2020 8:39	7/18/2020 3:27	3225	2	3	11	28.0	...	1	1	0.8296
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3053	133323839	0:09:03	7/11/2020 12:44	7/11/2020 12:53	7/11/2020 5:44	3012	1	7	8	NaN	...	1	1	3.7913
3054	133305818	0:06:57	7/11/2020 16:18	7/11/2020 16:25	7/11/2020 9:18	3008	1	1	3	NaN	...	1	1	1.1576
3055	133260048	0:21:46	7/9/2020 11:49	7/9/2020 12:12	7/9/2020 4:49	3004	1	1	7	NaN	...	1	1	0.7998
3056	133305807	0:06:50	7/11/2020 9:05	7/11/2020 9:12	7/11/2020 2:05	3008	1	1	3	NaN	...	1	1	0.7998
3057	133352713	0:09:20	7/13/2020 9:56	7/13/2020 14:44	7/13/2020 2:56	3003	1	1	2	NaN	...	1	3	1.1576

3058 rows × 249 columns

[105]:

```
#przedstawic mozliwosci pracy z duzymi plikami przy uzyciu argumentu
#chunksize


chunksize = 1000  # Liczba wierszy w każdym kawałku
results = []

for chunk in pd.read_csv('IHME_ORB_C19HSDS_2020_Y2020M12D03.CSV', chunksize=chunksize):
    chunk['Total'] = chunk['R1'] + chunk['R1_5']
    results.append(chunk)

final_result = pd.concat(results)

print(final_result.head())
```

```
      SbjNum NetDuration InterviewTimeVStart InterviewTimeVEnd  \
0  133476254     0:10:14     7/17/2020 13:53   7/17/2020 14:26
1  133281846     0:22:16     7/10/2020 12:53   7/10/2020 14:47
2  133280780     0:19:23     7/10/2020 12:35   7/10/2020 12:54
3  133281834     0:10:11     7/10/2020 10:21   7/10/2020 10:32
4  133491249     0:09:59      7/18/2020 8:27    7/18/2020 8:39

            Date  Srvyr  Country  LANG  R1  R1_5  ... G11_99 FinalOutcome  \
0  7/17/2020 8:53   3232        2     1   9  15.0  ...    NaN            1
1  7/10/2020 7:53   3206        2     4  12  22.0  ...    NaN            1
2  7/10/2020 7:35   3202        2     3  10  13.0  ...    NaN            1
3  7/10/2020 5:21   3212        2     1  12   9.0  ...    NaN            1
4  7/18/2020 3:27   3225        2     3  11  28.0  ...    NaN            1

   NumOfVisits  weight_combined  kenya_weight nigeria_weight  \
0            1         0.829860           NaN       0.829860
1            1         1.416946           NaN       1.416946
2            1         0.883601           NaN       0.883601
3            1         1.416946           NaN       1.416946
4            1         0.829860           NaN       0.829860

   southafrica_weight agegroup  gk_weight Total
0                 NaN        1   1.555754  24.0
1                 NaN        2   1.949579  34.0
2                 NaN        2   2.151458  23.0
3                 NaN        2   2.325065  21.0
4                 NaN        1   1.640484  39.0

[5 rows x 248 columns]
```

3. Wnioski

Praca z biblioteką pandas umożliwia łatwe i efektywne zarządzanie danymi. Analizowanie danych dzięki filtrowaniu, grupowaniu, transponowaniu, łączeniu kolumn itp. jest proste nawet na dużej ilości danych.