**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

# О Т Ч Е Т

## по домашнему заданию № 1

**Название:** Дескриптивный анализ данных

**Дисциплина:** Методы машинного обучения

| | | | |
|---|---|---|---|
| Студент | ИУ6-22М | | Д.С. Каткова |
| | (Группа) | (Подпись, дата) | (И.О. Фамилия) |
| Преподаватель | | | С.Ю. Папулин |
| | | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

# Домашнее задание 1. Дескриптивный анализ данных

Каткова Дарья, ИУ6-22М

## Цель работы

Приобрести опыт решения практических задач по анализу данных, таких как загрузка, трансформация, вычисление простых статистик и визуализация данных в виде графиков и диаграмм, посредством языка программирования Python.

## Вариант: 5-1-3

In [2]:
```
surname = "Каткова"  # Фамилия

alp = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'
w = [1, 42, 21, 21, 34,  6, 44, 26, 18, 44, 38, 26, 14, 43,  4, 49,
        7, 42, 29,  4,  9, 36, 34, 31, 29,  5, 30,  4, 19, 28, 25,

d = dict(zip(alp, w))
variant =  sum([d[el] for el in surname.lower()]) % 40 + 1

print("Задача № 1, шаг 5 – вариант: ", variant % 5 + 1)
print("Задача № 1, шаг 11 – вариант: ", variant % 2 + 1 )
print("Задача № 2 – вариант: ", variant % 4 + 1)
```

```
Задача № 1, шаг 5 – вариант:  5
Задача № 1, шаг 11 – вариант:  1
Задача № 2 – вариант:  3
```

# Задание 1. Анализ индикаторов качества государственного управления (WGI)

In [3]:
```
# Подключение библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

## Загрузка данных в DataFrame и сортировка данных по убыванию индекса DataFrame

In [4]:
```
df_corruption = pd.read_excel('wgidataset.xlsx', sheet_name = "Cont

# Сортировка по убыванию индекса
df_corruption = df_corruption.sort_index(ascending = False)

# Вывод десяти строк, начиная с головы
df_corruption.head(10)
```

Out[4]:

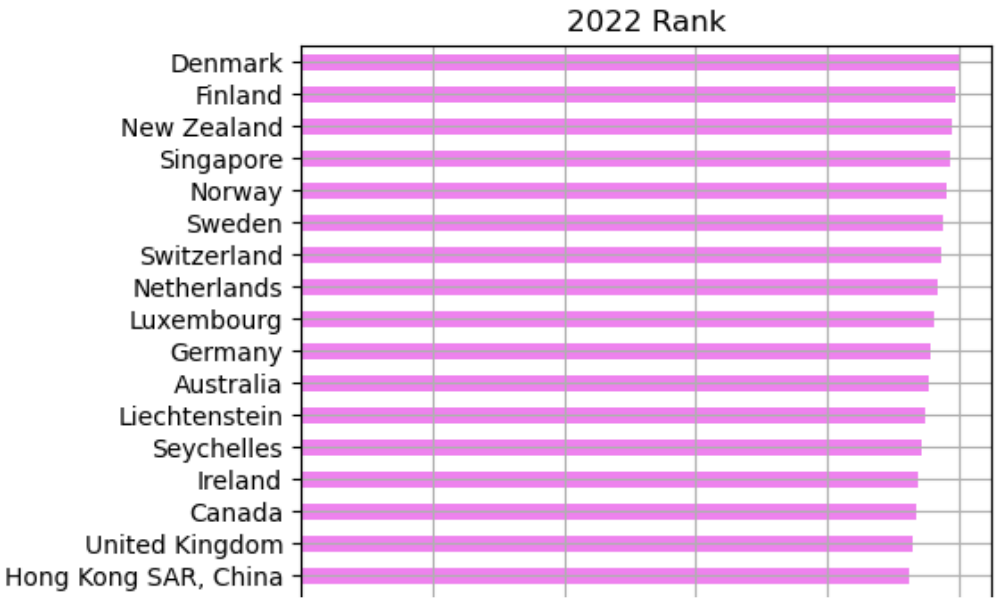
| | Country/Territory | Code | Estimate | StdErr | NumSrc | Rank | Lower | Uppe |
|---|---|---|---|---|---|---|---|---|
| **213** | Zimbabwe | ZWE | -0.278847 | 0.244907 | 5.0 | 47.849461 | 30.645161 | 60.75268 |
| **212** | Zambia | ZMB | -0.840641 | 0.262077 | 4.0 | 24.731182 | 5.913979 | 41.39785 |
| **211** | Congo, Dem. Rep. | ZAR | -1.647852 | 0.315914 | 3.0 | 0.000000 | 0.000000 | 12.36559 |
| **210** | South Africa | ZAF | 0.732927 | 0.210325 | 6.0 | 76.344086 | 66.129036 | 81.18279 |
| **209** | Serbia | SRB | -1.140072 | 0.262077 | 4.0 | 11.827957 | 0.537634 | 29.03225 |
| **208** | Yemen, Rep. | YEM | -0.743732 | 0.262077 | 4.0 | 27.419355 | 8.602151 | 47.31182 |
| **207** | Jersey, Channel Islands | JEY | NaN | NaN | NaN | NaN | NaN | NaN |
| **206** | West Bank and Gaza | WBG | 0.041492 | 0.340507 | 2.0 | 59.139786 | 35.483871 | 73.11827 |
| **205** | Vanuatu | VUT | 0.216309 | 0.439480 | 1.0 | 62.365593 | 36.559139 | 80.64516 |
| **204** | Vietnam | VNM | -0.489799 | 0.212363 | 6.0 | 37.096775 | 24.731182 | 52.15053 |

10 rows × 146 columns

## Отображение данных по индексу WGI за 2022 год в виде горизонтального столбчатого графика (rank)

In [5]:
```
# Установим индекс Country/Territory
df_rank = df_corruption.set_index('Country/Territory')

# Выбираем для отображения Rank за 2022 год, заполняем нулями там,
df_rank = df_rank[["Rank.23"]].fillna(0).sort_values("Rank.23")

# Отображаем данные в виде горизонтального столбчатого графика
df_rank.plot.barh(color = 'violet', figsize = (5, 50), title = "202
```

Out[5]:
```
<Axes: title={'center': '2022 Rank'}, ylabel='Country/Territory'
>
```




## Сформируем DataFrame из исходного для региона в соответствии с вариантом 5: Sub Saharan Africa

In [6]:
```
df_regions = pd.read_excel("regions.xlsx")

# Объединим regions.xlsx и ControlofCorruption по Code
df_merged = pd.merge(df_regions, df_corruption, on='Code')

# Сформируем DataFrame для региона Sub Saharan Africa (SSA) и отсор
df_SSA = df_merged[df_merged.Region=='SSA'].sort_index(ascending =
df_SSA.head(10)
```

Out[6]:

| | Country | Code | Region | Country/Territory | Estimate | StdErr | NumSrc | Rank |
|---|---|---|---|---|---|---|---|---|
| **176** | Zimbabwe | ZWE | SSA | Zimbabwe | -0.278847 | 0.244907 | 5.0 | 47.849461 |
| **175** | Zambia | ZMB | SSA | Zambia | -0.840641 | 0.262077 | 4.0 | 24.731182 |
| **164** | Uganda | UGA | SSA | Uganda | -0.723757 | 0.262077 | 4.0 | 27.956989 |
| **159** | Togo | TGO | SSA | Togo | -0.842621 | 0.315914 | 3.0 | 24.193548 |
| **157** | Tanzania | TZA | SSA | Tanzania | -0.702762 | 0.262077 | 4.0 | 29.569893 |
| **150** | Sudan | SDN | SSA | Sudan | -1.240006 | 0.262077 | 4.0 | 6.989247 |
| **147** | South Sudan | SSD | SSA | South Sudan | NaN | NaN | NaN | NaN |
| **146** | South Africa | ZAF | SSA | South Africa | 0.732927 | 0.210325 | 6.0 | 76.344086 |
| **145** | Somalia | SOM | SSA | Somalia | -1.273832 | 0.315914 | 3.0 | 5.376344 |
| **140** | Sierra Leone | SLE | SSA | Sierra Leone | -0.756474 | 0.315914 | 3.0 | 26.344086 |

10 rows × 148 columns

## Построение графиков индекса WGI за 1996-2022 для стран своего региона (estimate).

In [7]:
```
# Установим индекс Country и отфильтруем по Estimate
df_estimate = df_SSA.set_index('Country').filter(regex='Estimate')
df_estimate.head(10)
```

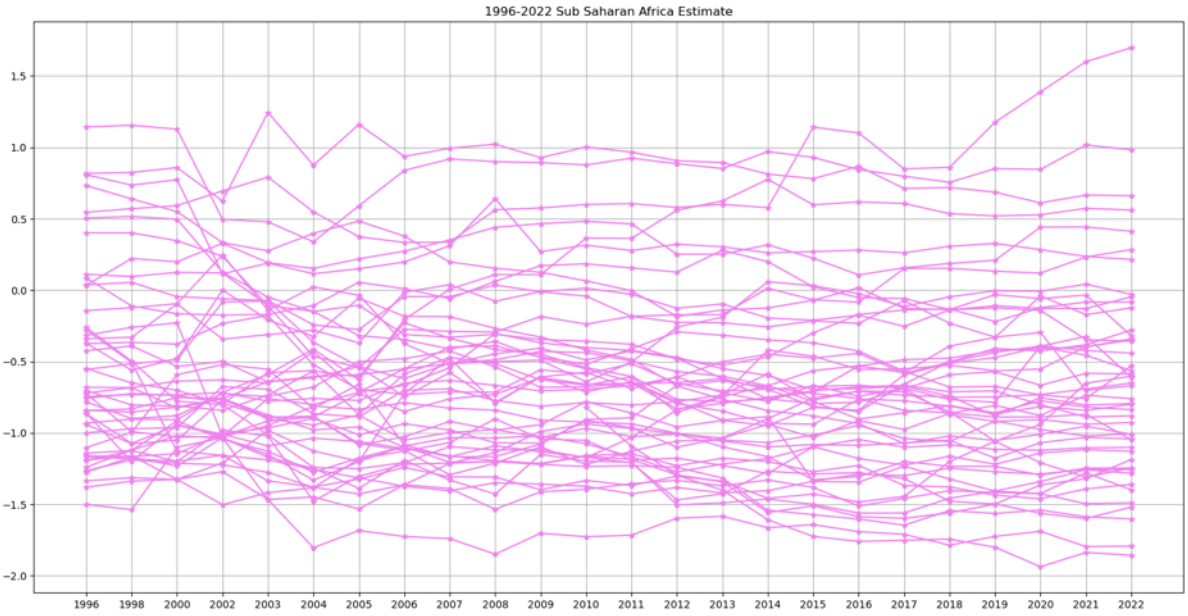
Out[7]:

| Country | Estimate | Estimate.1 | Estimate.2 | Estimate.3 | Estimate.4 | Estimate.5 | Estimate.6 |
|---|---|---|---|---|---|---|---|
| **Zimbabwe** | -0.278847 | -0.504802 | -1.127275 | -1.156760 | -1.188868 | -1.253563 | -1.314617 |
| **Zambia** | -0.840641 | -0.853156 | -0.818261 | -0.758519 | -0.641859 | -0.604488 | -0.592440 |
| **Uganda** | -0.723757 | -0.989971 | -1.028190 | -1.025739 | -0.948509 | -0.811879 | -0.833649 |
| **Togo** | -0.842621 | -0.831448 | -0.751899 | -0.758070 | -0.902239 | -0.943994 | -0.873771 |
| **Tanzania** | -0.702762 | -0.803638 | -0.810984 | -0.840932 | -0.734542 | -0.565570 | -0.607606 |
| **Sudan** | -1.240006 | -1.119795 | -1.018885 | -1.035823 | -1.170058 | -1.246903 | -1.387641 |
| **South Sudan** | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **South Africa** | 0.732927 | 0.638809 | 0.550270 | 0.332902 | 0.275541 | 0.397753 | 0.483857 |
| **Somalia** | -1.273832 | -1.169290 | -1.231536 | -1.032931 | -1.465910 | -1.803035 | -1.683112 |
| **Sierra Leone** | -0.756474 | -0.724984 | -0.736802 | -0.789459 | -0.892783 | -0.881413 | -1.085492 |

10 rows × 24 columns

In [8]:
```
# Переименуем названия столбцов
years = np.arange(1996, 2023, 1)

# Удалим года, информации о которых нет (1997, 1999, 2001)
years = np.delete(years, 1)
years = np.delete(years, 2)
years = np.delete(years, 3)

# Для корректного отображения преобразуем years в строку
df_estimate.columns = map(str, years)

# Отобразим данные в виде графиков
df_estimate.T.plot(color='violet', figsize=(20,10), title='1996-202
```

Out[8]:
```
<Axes: title={'center': '1996-2022 Sub Saharan Africa Estimate'}>
```




## Найдем страны с наибольшим и наименьшим значением WGI региона SSA за 2022 год (estimate)

In [9]:
```
# Наибольшее значение WGI региона
max_estimate = df_estimate['2022'].idxmax()
max_estimate
```

Out[9]:
```
'Seychelles'
```

In [10]:
```
# Наименьшее значение WGI региона
min_estimate = df_estimate['2022'].idxmin()
min_estimate
```

Out[10]:
```
'South Sudan'
```

## Определим средние значения региона SSA за каждый год в период с 1996 по 2022 (estimate)

In [11]:
```
# Средние значения WGI региона
mean = df_estimate.mean()
mean
```

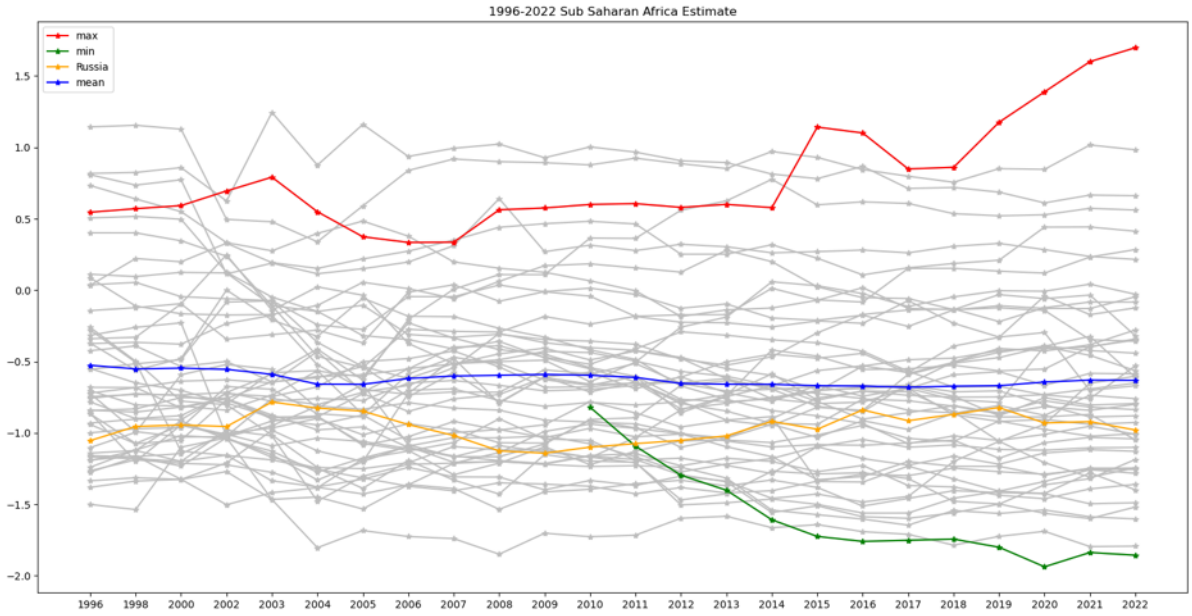
Out[11]:
```
1996   -0.526003
1998   -0.551466
2000   -0.545919
2002   -0.554189
2003   -0.589383
2004   -0.658061
2005   -0.659048
2006   -0.618499
2007   -0.600693
2008   -0.595974
2009   -0.590122
2010   -0.595736
2011   -0.610929
2012   -0.652935
2013   -0.658446
2014   -0.659135
2015   -0.668719
2016   -0.671172
2017   -0.679782
2018   -0.671723
2019   -0.668655
2020   -0.643094
2021   -0.629466
2022   -0.631573
dtype: float64
```

## Построим графики индекса WGI за 1996-2022 для стран SSA и выделите страны с наибольшим и наименьшим значением WGI за 2022 год, а также отобразите среднее значение по региону и РФ.

In [12]:
```
# Транспонируем
df_estimate = df_estimate.T
```

In [14]:
```
# Отобразим данные в виде графиков
df_estimate.plot(color='#bdbdbd', figsize=(20,10), title='1996-2022

# Наибольшее значение WGI региона
df_estimate[max_estimate].plot(label="max", color='red', marker='*'

# Наименьшее значение WGI региона
df_estimate[min_estimate].plot(label="min", color='green', marker='

# Среднее значение по РФ
RF = df_merged.set_index('Country/Territory').T.get("Russian Federa
RF.plot(label="Russia", color='orange', marker='*', legend=True)

# Среднее значение по региону
mean.T.plot(color='blue', marker='*', legend=True, label="mean", xt
```

Out[14]:
```
<Axes: title={'center': '1996-2022 Sub Saharan Africa Estimate'}>
```




## Определим, как изменилось значение показателя rank с 1996 по 2022 для Europe and Central Asia

In [15]:
```
# Фильтруем по региону согласно варианту ECA и установим индекс Cou
df_ECA = df_merged[df_merged.Region=='ECA'].set_index('Country/Terr

# Выбираем из таблицы столбцы Rank (1996 год) и Rank.23 (2022 год)
df_ECA = df_ECA.filter(items = ['Rank', 'Rank.23'])

# Находим разницу rank между 2022 и 1996 годом
df_ECA['Diff'] = df_ECA['Rank'] - df_ECA['Rank.23']

# Меняем заголовки столбца
df_ECA.columns = ['1996', '2022', 'Difference']

df_ECA
```

Out[15]:

| Country/Territory | 1996 | 2022 | Difference |
|---|---|---|---|
| **Albania** | 19.354839 | 38.679245 | -19.324406 |
| **Armenia** | 38.172043 | 56.132076 | -17.960033 |
| **Azerbaijan** | 2.688172 | 16.981133 | -14.292960 |
| **Belarus** | 42.473118 | 31.603773 | 10.869345 |
| **Bosnia and Herzegovina** | 48.924732 | 25.943396 | 22.981337 |
| **Georgia** | 1.075269 | 72.169815 | -71.094546 |
| **Kazakhstan** | 12.365591 | 48.584908 | -36.219316 |
| **Kosovo** | NaN | 47.169811 | NaN |
| **Kyrgyz Republic** | 17.204302 | 10.377358 | 6.826943 |
| **Moldova** | 39.784946 | 42.924530 | -3.139584 |
| **Montenegro** | NaN | 50.943398 | NaN |
| **North Macedonia** | 32.258064 | 43.867924 | -11.609859 |
| **Russian Federation** | 15.053763 | 19.339622 | -4.285859 |
| **Serbia** | 11.827957 | 35.377357 | -23.549400 |
| **Tajikistan** | 5.913979 | 6.132075 | -0.218097 |
| **Türkiye** | 51.612904 | 34.905659 | 16.707245 |
| **Turkmenistan** | 15.591398 | 5.660378 | 9.931021 |
| **Ukraine** | 13.440860 | 29.245283 | -15.804423 |
| **Uzbekistan** | 12.903226 | 24.528301 | -11.625075 |

## Выведем таблицу для варианта 3 - Europe and Central Asia (WGI - rank)

In [16]:
```
# Переменные, хранящие названия стран с максимальным и минимальным
max_eca = df_ECA['2022'].idxmax()
min_eca = df_ECA['2022'].idxmin()

# Данные, содержащиеся в таблице
data = [
    ["ECA", "-", df_ECA['1996'].mean(), df_ECA['2022'].mean(), df_E
    ["ECA", max_eca, df_ECA['1996'][max_eca], df_ECA['2022'][max_ec
    ["ECA", min_eca, df_ECA['1996'][min_eca], df_ECA['2022'][min_ec
    ["ECA", "Russian Federation", df_ECA['1996']["Russian Federatio
    ]

# Названия строк и столбцов
df_table = pd.DataFrame(data, columns=["Регион", "Страна", "WGI 199
                  index=["mean_2022", "max_2022", "min_2022", "Russ
df_table
```

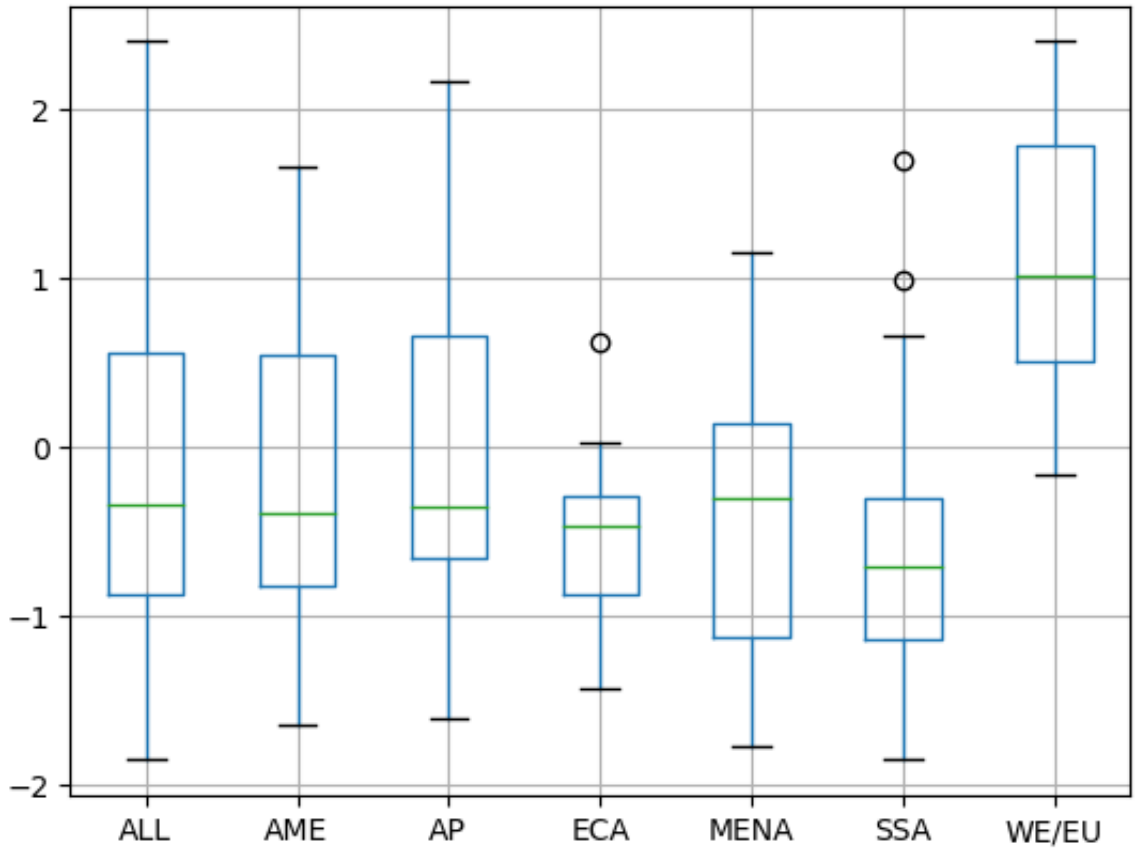
Out[16]:

| | Регион | Страна | WGI 1996 | WGI 2022 | Изменение |
|---|---|---|---|---|---|
| **mean_2022** | ECA | - | 22.390892 | 33.714002 | 11.323110 |
| **max_2022** | ECA | Georgia | 1.075269 | 72.169815 | 71.094546 |
| **min_2022** | ECA | Turkmenistan | 15.591398 | 5.660378 | -9.931021 |
| **Russia_2022** | ECA | Russian Federation | 15.053763 | 19.339622 | 4.285859 |

## Отобразим диаграмму размаха (boxplot) индекса WGI за 2022 для всех стран и для каждого региона в отдельности (на одном графике) (estimate)

In [18]:
```
# Группируем по региону и выбираем столбцы estimate за 2022 год и с
regions = df_merged.groupby('Region')[['Estimate.23', 'Country']]
df_box = []

# Формируем массив DataFrame'ов по каждому региону
for region in regions:
    df_box.append(region[1].set_index('Country').set_axis([region[0

# Создаем DataFrame для всех стран и добавляем его к массиву
df_all_regions = df_merged.set_index('Country/Territory').filter(re
df_box.append(df_all_regions)

# Объединяем массив DataFrame'ов
df_box = pd.concat(df_box, sort = True)
df_box.boxplot()
```

Out[18]: <Axes: >




## Задание 2.

**Загрузить данные в один dataframe из всех файлов в папке /data/stock. Все файлы имеют одинаковую структуру, в том числе наименование столбцов. В качестве значений индекса dataframe'а необходимо указать значения столбца "Date". Название столбцов должные соответствовать названию акций (имя файла без .csv), а их значения - значениям цены закрытия (столбец "Close" в файлах .csv)**

In [19]:
```
# Подключение всех файлов
import glob
from pathlib import Path

glob_file = glob.glob('./stock/*.csv')

# Создаем переменную для хранения значений
df_file = {}

# Цикл прохождения по всем файлам
for file in glob_file:
    data = pd.read_csv(file, index_col='Date') # Считываем данные
    data_name = Path(file).stem # Получаем название файла
    df_file[data_name] = data['Close']

# Объединение всех файлов
df_all_files = pd.concat(df_file, axis=1, sort = True)
df_all_files.head(10)
```

Out[19]:

| Date | CSCO | GTLB | ADBE | TCOM | DBX | ABNB | ORCL | |
|---|---|---|---|---|---|---|---|---|
| 2022-01-01 | 55.669998 | 64.010002 | 534.299988 | 26.610001 | 24.750000 | 153.970001 | 81.160004 | 60.0 |
| 2022-02-01 | 55.770000 | 58.270000 | 467.679993 | 25.820000 | 22.690001 | 151.490005 | 75.970001 | 54.5 |
| 2022-03-01 | 55.759998 | 54.450001 | 455.619995 | 23.120001 | 23.250000 | 171.759995 | 82.730003 | 57.2 |
| 2022-04-01 | 48.980000 | 47.930000 | 395.950012 | 23.650000 | 21.750000 | 153.210007 | 73.400002 | 51.9 |
| 2022-05-01 | 45.049999 | 38.939999 | 416.480011 | 22.059999 | 20.840000 | 120.870003 | 71.919998 | 48.6 |
| 2022-06-01 | 42.639999 | 53.139999 | 366.059998 | 27.450001 | 20.990000 | 89.080002 | 69.870003 | 41.6 |
| 2022-07-01 | 45.369999 | 57.400002 | 410.119995 | 25.780001 | 22.740000 | 110.980003 | 77.839996 | 48.6 |
| 2022-08-01 | 44.720001 | 59.869999 | 373.440002 | 25.719999 | 21.389999 | 113.120003 | 74.150002 | 44.1 |
| 2022-09-01 | 40.000000 | 51.220001 | 275.200012 | 27.309999 | 20.719999 | 105.040001 | 61.070000 | 36.8 |
| 2022-10-01 | 45.430000 | 48.459999 | 318.500000 | 22.629999 | 21.750000 | 106.910004 | 78.070000 | 39.8 |

10 rows × 25 columns

## Рассчитаем корреляционную матрицу для всех акций

In [20]:
```
# Корреляционная матрица для всех акций
df_all_files.corr().head(10)
```

Out[20]:

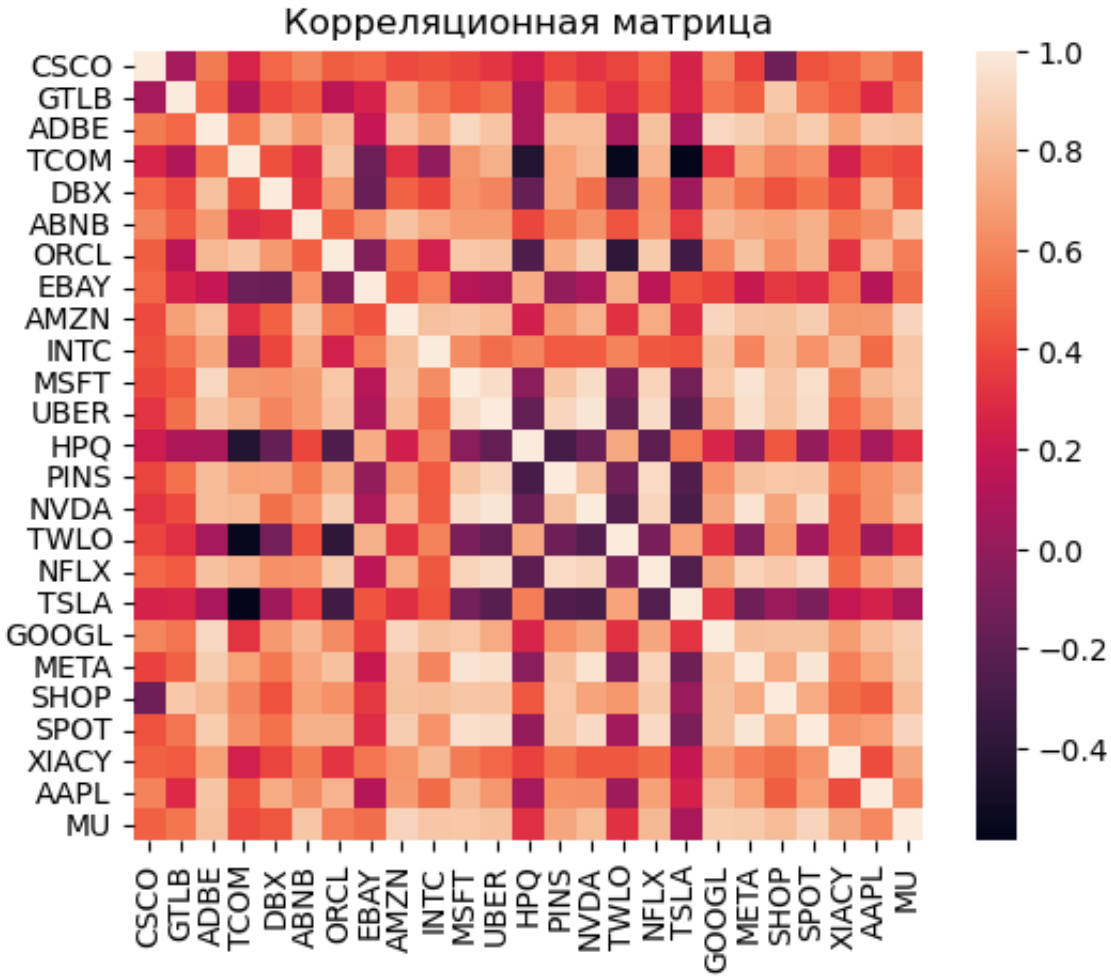
| | CSCO | GTLB | ADBE | TCOM | DBX | ABNB | ORCL | EBAY |
|---|---|---|---|---|---|---|---|---|
| **CSCO** | 1.000000 | 0.068856 | 0.554172 | 0.257188 | 0.496982 | 0.594365 | 0.463955 | 0.494938 |
| **GTLB** | 0.068856 | 1.000000 | 0.496556 | 0.103614 | 0.402517 | 0.460602 | 0.138574 | 0.251066 |
| **ADBE** | 0.554172 | 0.496556 | 1.000000 | 0.533298 | 0.816359 | 0.670509 | 0.785432 | 0.180354 |
| **TCOM** | 0.257188 | 0.103614 | 0.533298 | 1.000000 | 0.423136 | 0.294269 | 0.836340 | -0.149330 |
| **DBX** | 0.496982 | 0.402517 | 0.816359 | 0.423136 | 1.000000 | 0.332740 | 0.667833 | -0.157363 |
| **ABNB** | 0.594365 | 0.460602 | 0.670509 | 0.294269 | 0.332740 | 1.000000 | 0.471504 | 0.644140 |
| **ORCL** | 0.463955 | 0.138574 | 0.785432 | 0.836340 | 0.667833 | 0.471504 | 1.000000 | -0.070414 |
| **EBAY** | 0.494938 | 0.251066 | 0.180354 | -0.149330 | -0.157363 | 0.644140 | -0.070414 | 1.000000 |
| **AMZN** | 0.404820 | 0.690644 | 0.819614 | 0.309545 | 0.478171 | 0.830690 | 0.534556 | 0.434078 |
| **INTC** | 0.420854 | 0.535441 | 0.713875 | -0.014994 | 0.390625 | 0.738241 | 0.239485 | 0.580047 |

10 rows × 25 columns

## Отобразим корреляционную матрицу в виде диаграммы.

In [21]:
```
# Подключим библиотеку для диаграмм heatmap
import seaborn as sns

# Вывод диаграммы
plt.title("Корреляционная матрица")
sns.heatmap(df_all_files.corr(), square = True)
```

Out[21]:
```
<Axes: title={'center': 'Корреляционная матрица'}>
```




## В соответствии с вариантом №3 Uber (UBER) определить:

- акцию с максимальной положительной корреляцией (max)
- акцию с максимальной отрицательной корреляцией (min)
- акцию с минимальной корреляцией (которая больше всего соответствует отсутствию какой-либо корреляции (none)

In [22]:
```
# Найдем корреляции акций, удалим лишнюю строчку с Uber
df_uber = df_all_files.corr().drop(["UBER"], axis = 1)
df_uber = df_uber.loc[["UBER"]].T
df_uber
```

Out[22]:

| | UBER |
|---|---|
| **CSCO** | 0.326346 |
| **GTLB** | 0.521399 |
| **ADBE** | 0.834611 |
| **TCOM** | 0.754442 |
| **DBX** | 0.595928 |
| **ABNB** | 0.680764 |
| **ORCL** | 0.832075 |
| **EBAY** | 0.085736 |
| **AMZN** | 0.796897 |
| **INTC** | 0.512572 |
| **MSFT** | 0.939538 |
| **HPQ** | -0.180970 |
| **PINS** | 0.907751 |
| **NVDA** | 0.969790 |
| **TWLO** | -0.186828 |
| **NFLX** | 0.937042 |
| **TSLA** | -0.221155 |
| **GOOGL** | 0.737311 |
| **META** | 0.954444 |
| **SHOP** | 0.836565 |
| **SPOT** | 0.933308 |
| **XIACY** | 0.495835 |
| **AAPL** | 0.661323 |
| **MU** | 0.820809 |

In [23]:
```
# Определим акцию с максимальной положительной корреляцией (max)
max_corr = df_uber.filter(regex='UBER').idxmax()
max_corr[0]
```

Out[23]: 'NVDA'

In [24]:
```
# Определим акцию с максимальной отрицательной корреляцией (min)
min_corr = df_uber.filter(regex='UBER').idxmin()
min_corr[0]
```

Out[24]: 'TSLA'

In [26]:
```
# Определим акцию с корреляцией, ближайшей к none
closest_to_zero_corr = df_uber.filter(regex='UBER').abs().idxmin()
closest_to_zero_corr[0]
```

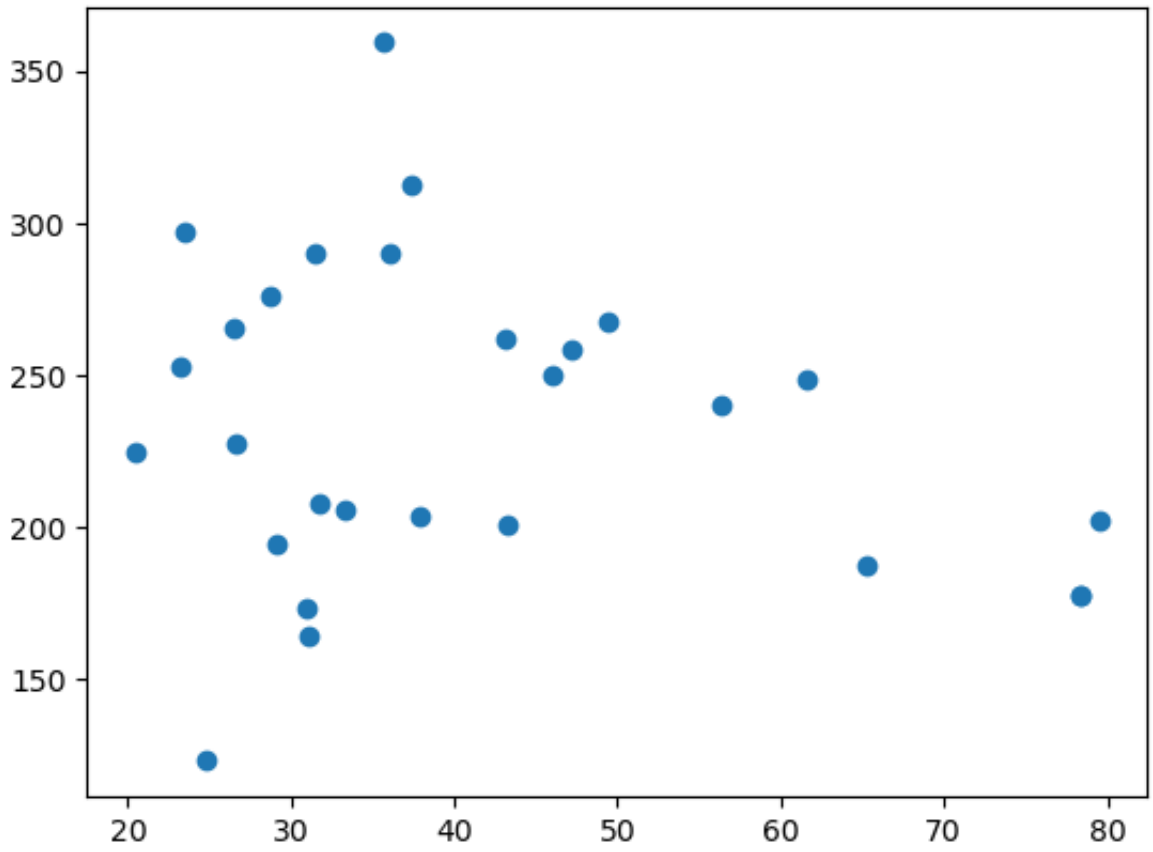
Out[26]: 'EBAY'

### Построим диаграммы разброса

- (Ваша компания - Компания с min),
- (Ваша компания - Компания с max),
- (Ваша компания - Компания с none)

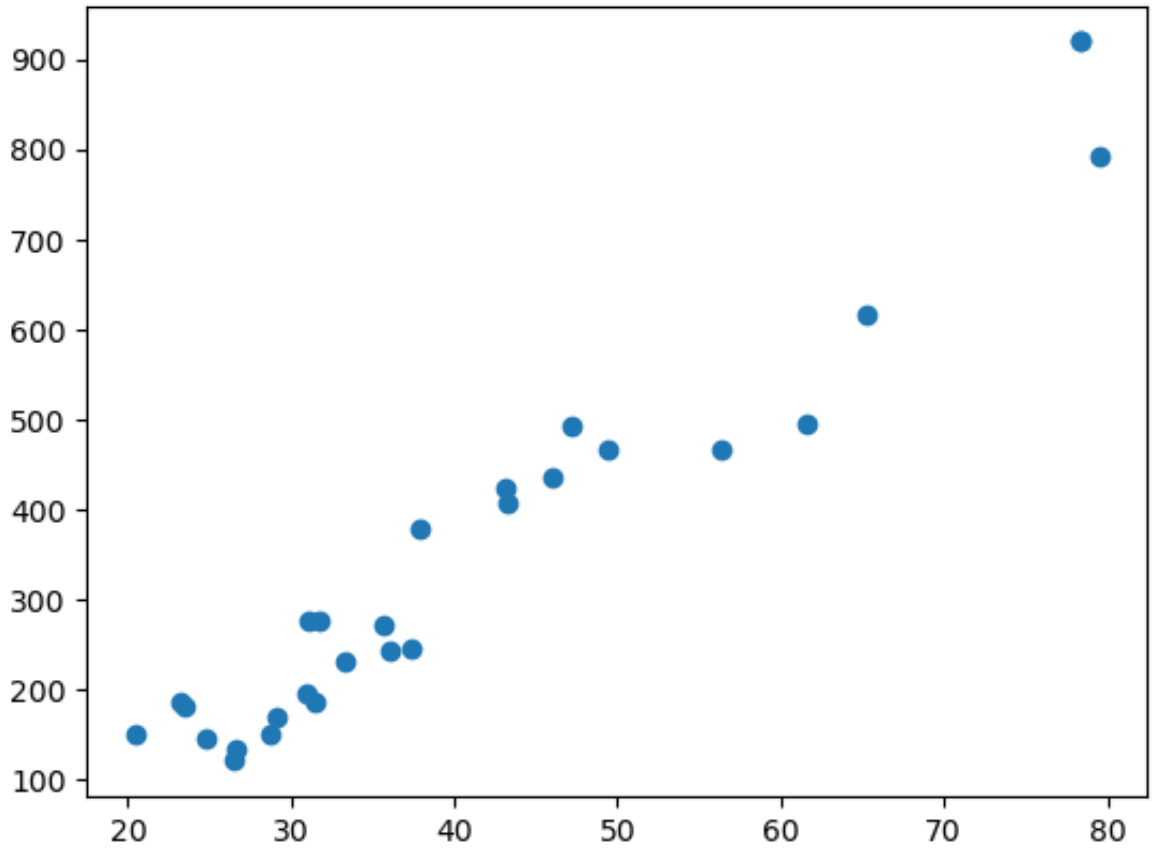
In [27]:
```
# Диаграмма разброса UBER – Компания с min (TSLA)
plt.scatter(df_all_files["UBER"], df_all_files[min_corr[0]])
```

Out[27]: <matplotlib.collections.PathCollection at 0x15b64ff50>



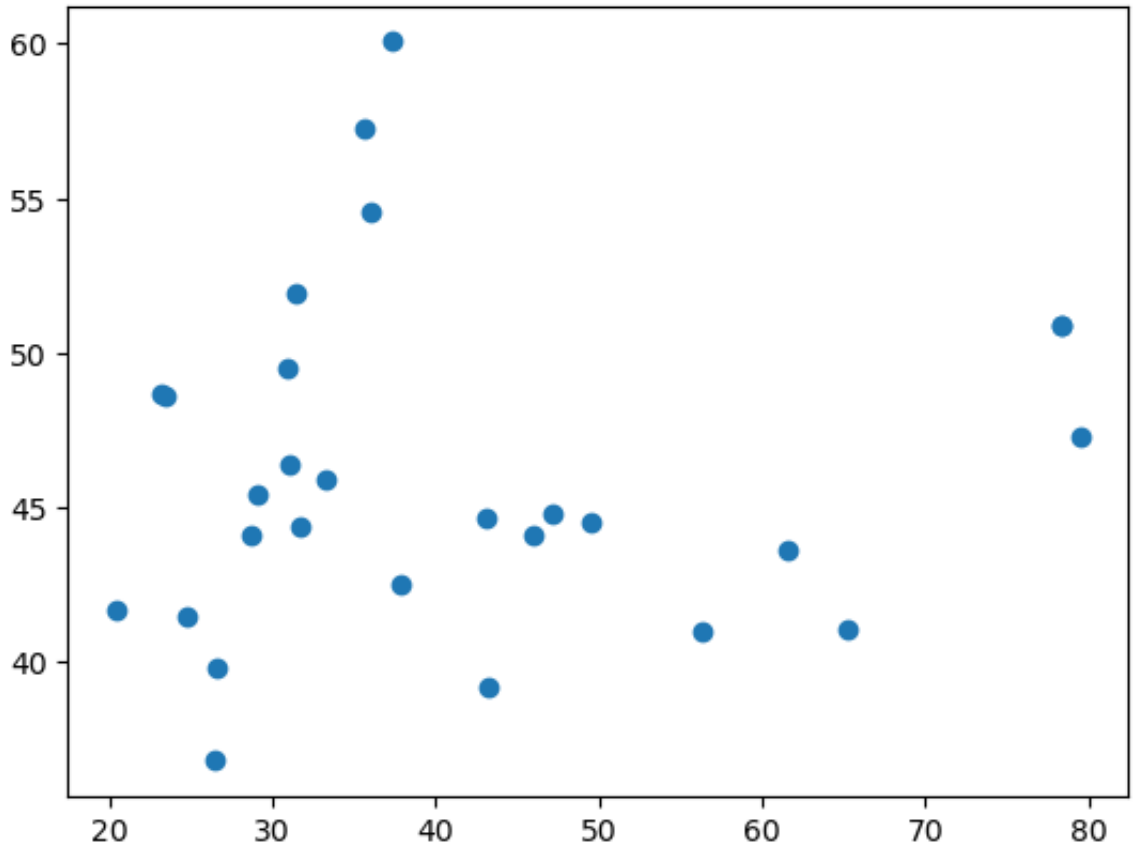
In [32]:
```
# Диаграмма разброса с UBER – Компания с max (NVDA)
plt.scatter(df_all_files["UBER"], df_all_files[max_corr[0]])
```

Out[32]:
```
<matplotlib.collections.PathCollection at 0x15c0d8ed0>
```



In [29]:
```
# Диаграмма разброса с UBER – Компания с none (EBAY)
plt.scatter(df_all_files["UBER"], df_all_files[closest_to_zero_corr
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x15b6586d0>
```




**Рассчет средней цены акций для каждого месяца (исходные данные взяты с интервалом в месяц)**

In [30]:
```
# Средняя цена акций для каждого месяца
mean_per_month = df_all_files.T.mean()
mean_per_month
```

Out[30]:
```
Date
2022-01-01    154.857167
2022-02-01    140.774723
2022-03-01    145.272287
2022-04-01    115.763514
2022-05-01    112.316034
2022-06-01     99.256929
2022-07-01    114.014999
2022-08-01    107.380833
2022-09-01     94.437083
2022-10-01     97.227501
2022-11-01    100.671666
2022-12-01     92.028958
2023-01-01    108.279540
2023-02-01    108.613126
2023-03-01    120.210832
2023-04-01    115.778799
2023-05-01    131.258401
2023-06-01    145.426799
2023-07-01    153.207200
2023-08-01    152.016000
2023-09-01    141.760400
2023-10-01    140.454598
2023-11-01    159.367601
2023-12-01    164.859599
2024-01-01    174.886801
2024-02-01    189.609962
2024-03-01    196.083201
2024-03-12    196.083201
dtype: float64
```

**Постройте графики для акций из пункта 4 и средней из пункта 6.**

In [31]:
```
# Построение графиков для акций
df_all_files["UBER"].plot(color='orange', marker='*', legend=True)
df_all_files[max_corr[0]].plot(color='r', marker='*', legend=True)
df_all_files[min_corr[0]].plot(color='g', marker='*', legend=True)
df_all_files[closest_to_zero_corr[0]].plot(color='blue', marker='*'
mean.plot(title="Графики акций", figsize=(10,5), grid=1, color='gre
```

Out[31]:
```
<Axes: title={'center': 'Графики акций'}, xlabel='Date'>
```


***Вывод:***

В ходе выполнения домашнего задания был приобретен опыт решения практических задач по анализу данных, таких как загрузка, трансформация, вычисление простых статистик и визуализация данных в виде графиков и диаграмм, посредством языка программирования Python.