Выберите вариант

In [17]:
```
surname = "Марчук"

alp = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'
w = [1, 42, 21, 21, 34,  6, 44, 26, 18, 44, 38, 26, 14, 43,  4, 49, 45,
       7, 42, 29,  4,  9, 36, 34, 31, 29,  5, 30,  4, 19, 28, 25, 33]

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:  4
Задача № 1, шаг 11 - вариант:  2
задача № 2 - вариант:  2
```

Задание 1. Анализ индикаторов качества государственного управления (The Worldwide Governance Indicators, WGI)

Загружаем набор данных

In [72]:
```
from google.colab import files

# Загружаем файл данных
uploaded = files.upload()

# Сохраняем загруженный файл на диск
for filename in uploaded.keys():
    with open(filename, 'wb') as f:
        f.write(uploaded[filename])
```

Выбрать файлы Файл не выбран Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving wgidataset.xlsx to wgidataset.xlsx
```

1. Загружаем данные в DataFrame

In [73]:
```
# Теперь мы можем загрузить данные в DataFrame
import pandas as pd

# загруженный файл в формате Excel (xlsx)
regions = pd.read_excel('/content/regions.xlsx')
#загруженный файл имеет формат sta
#wgidataset = pd.read_stata('/content/wgidataset.dta')
df = pd.read_excel("/content/wgidataset.xlsx", 'ControlofCorruption', skiprows=14,)
# Split the DataFrame
part1 = df.iloc[:, :2]  # First two columns
part2 = df.iloc[:, 2:]  # Rest of the columns
```

```
column_groups = [part2.columns[i:i+6] for i in range(0, len(part2.columns), 6)]
separate_datasets = []
for group in column_groups:
    separate_datasets.append(df[group])
# Initialize an empty list to store concatenated datasets
concatenated_datasets = []

# Concatenate part1 with each dataset separately
for dataset in separate_datasets:
    concatenated_dataset = pd.concat([part1, dataset], axis=1)
    concatenated_datasets.append(concatenated_dataset)
new_column_names = ['Estimate', 'StdErr', 'NumSrc', 'Rank', 'Lower', 'Upper']
# Loop through each dataset
for dataset in concatenated_datasets:
    # Identify the last 5 columns
    dataset.columns = list(dataset.columns[:-6]) + new_column_names
```

In [84]:
```
import pandas as pd

# Example data
years = [1996, 1998, 2000, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 20

# Create DataFrames with unique identifiers (e.g., index)
dfs = [df.assign(Year=year) for year, df in zip(years, concatenated_datasets)]

# Concatenate DataFrames
wgidataset = pd.concat(dfs, ignore_index=True)

wgidataset.head()
```

Out[84]:

	Country/Territory	Code	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Year
0	Aruba	ABW	NaN	NaN	NaN	NaN	NaN	NaN	1996
1	Andorra	ADO	1.318143	0.480889	1.0	87.096771	72.043015	96.774193	1996
2	Afghanistan	AFG	-1.291705	0.340507	2.0	4.301075	0.000000	27.419355	1996
3	Angola	AGO	-1.167702	0.262077	4.0	9.677420	0.537634	27.419355	1996
4	Anguilla	AIA	NaN	NaN	NaN	NaN	NaN	NaN	1996

In [85]:
```
# Теперь выводим регионы DataFrame Regions dataset
regions
```

Out[85]:

	Country	Code	Region
0	Afghanistan	AFG	AP
1	Albania	ALB	ECA
2	Algeria	DZA	MENA
3	Angola	AGO	SSA
4	Argentina	ARG	AME
...	...	...	...
175	Venezuela	VEN	AME
176	Vietnam	VNM	AP
177	Yemen	YEM	MENA
178	Zambia	ZMB	SSA
179	Zimbabwe	ZWE	SSA

180 rows × 3 columns

2. Отсортируем данные по убыванию индекса Data Frame

In [86]:
```
# Сортируем данные по показателю индекса WGI в порядке возрастания, чтобы получить
sorted_wgidataset = wgidataset.sort_values(by='Rank', ascending=False)
```

In [87]:
```
# Отображаем отсортированный фрейм данных wgidataset
sorted_wgidataset
```

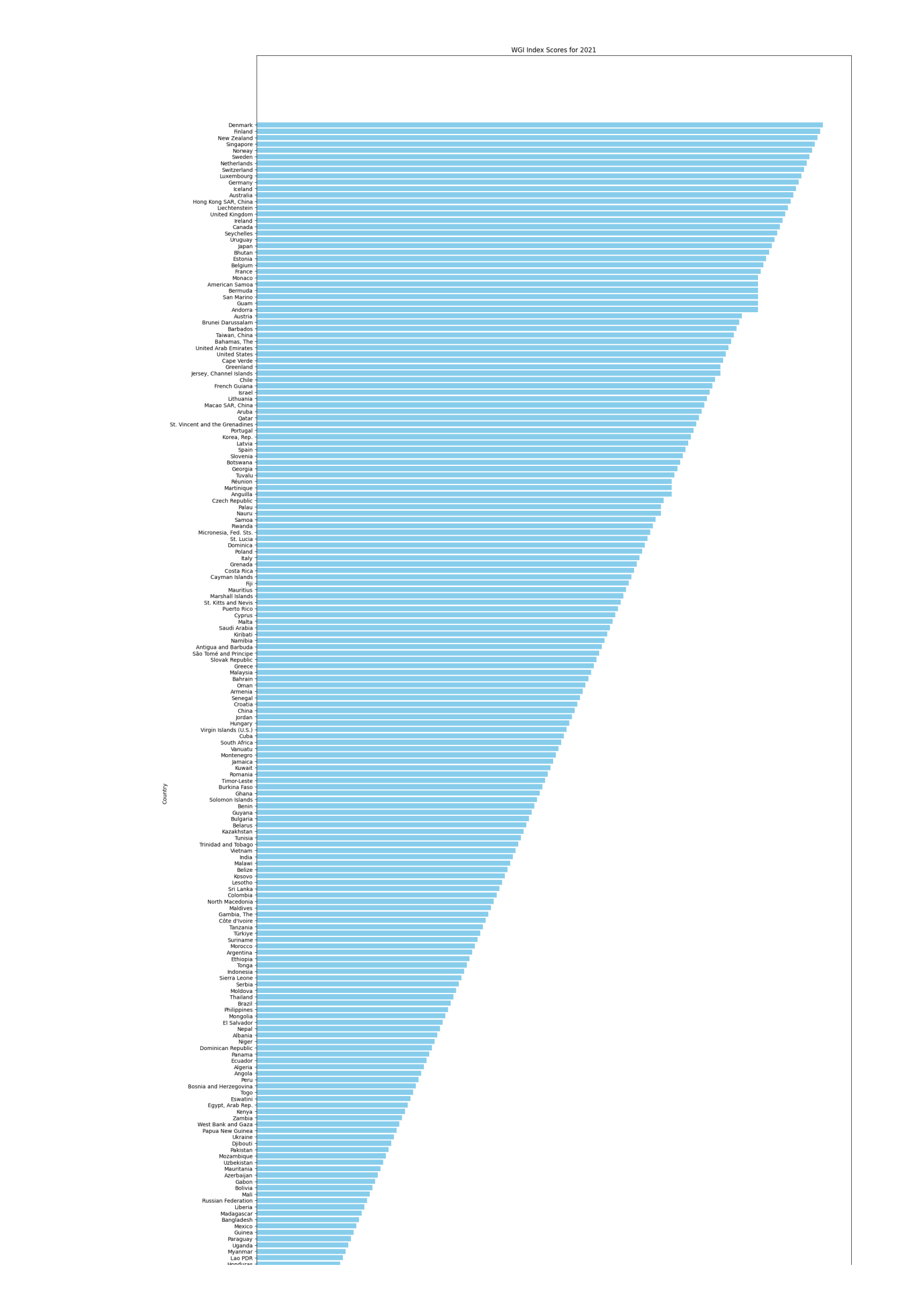
Out[87]:

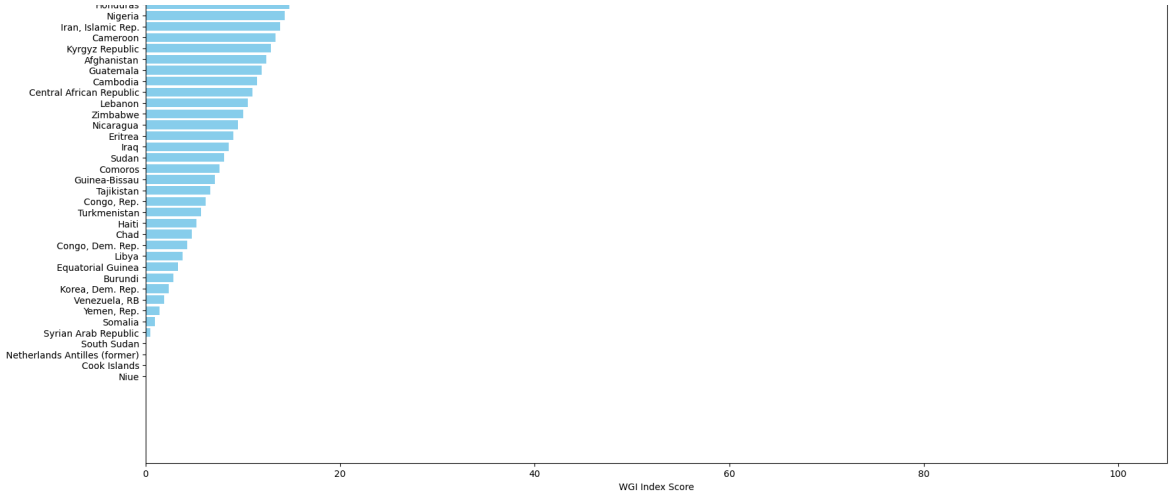
	Country/Territory	Code	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Year
4548	Denmark	DNK	2.236469	0.149797	10.0	100.0	95.714287	100.0	2020
1980	Denmark	DNK	2.376409	0.182742	8.0	100.0	96.116508	100.0	2008
2194	Denmark	DNK	2.435494	0.167893	9.0	100.0	97.607658	100.0	2009
2408	Denmark	DNK	2.352359	0.153291	10.0	100.0	97.619049	100.0	2010
4762	Denmark	DNK	2.333753	0.162209	10.0	100.0	97.142860	100.0	2021
...	...	...	...	...	...	...	...	...	...
4636	Niue	NIU	NaN	NaN	NaN	NaN	NaN	NaN	2020
4714	Netherlands Antilles (former)	ANT	NaN	NaN	NaN	NaN	NaN	NaN	2021
4750	Cook Islands	COK	NaN	NaN	NaN	NaN	NaN	NaN	2021
4850	Niue	NIU	NaN	NaN	NaN	NaN	NaN	NaN	2021
4928	Netherlands Antilles (former)	ANT	NaN	NaN	NaN	NaN	NaN	NaN	2022

5136 rows × 9 columns

3. Отображаем данные по индексу WGI за 2021 год в виде горизонтального столбчатого графика (rank).

In [88]:
```
import matplotlib.pyplot as plt

# Отфильтруйте набор данных, чтобы включить в него данные только за 2021 год
df_2021 = sorted_wgidataset[sorted_wgidataset['Year'] == 2021]


# Построение
plt.figure(figsize=(20, 50))
plt.barh(df_2021['Country/Territory'], df_2021['Rank'], color='skyblue')
plt.xlabel('WGI Index Score')
plt.ylabel('Country')
plt.title('WGI Index Scores for 2021')
plt.gca().invert_yaxis()  # Инвертируйте ось y, чтобы получить самый высокий ранг с
plt.show()
```


WGI Index Scores for 2021
Country
Denmark
Finland
New Zealand
Singapore
Norway
Sweden
Netherlands
Switzerland
Luxembourg
Germany
Iceland
Australia
Hong Kong SAR, China
Liechtenstein
United Kingdom
Ireland
Canada
Seychelles
Uruguay
Japan
Bhutan
Estonia
Belgium
France
Monaco
American Samoa
Bermuda
San Marino
Guam
Andorra
Austria
Brunei Darussalam
Barbados
Taiwan, China
Bahamas, The
United Arab Emirates
United States
Cape Verde
Greenland
Jersey, Channel Islands
Chile
French Guiana
Israel
Lithuania
Macao SAR, China
Aruba
Qatar
St. Vincent and the Grenadines
Portugal
Korea, Rep.
Latvia
Spain
Slovenia
Botswana
Georgia
Tuvalu
Réunion
Martinique
Anguilla
Czech Republic
Palau
Nauru
Samoa
Rwanda
Micronesia, Fed. Sts.
St. Lucia
Dominica
Poland
Italy
Grenada
Costa Rica
Cayman Islands
Fiji
Mauritius
Marshall Islands
St. Kitts and Nevis
Puerto Rico
Cyprus
Malta
Saudi Arabia
Kiribati
Namibia
Antigua and Barbuda
São Tomé and Principe
Slovak Republic
Greece
Malaysia
Bahrain
Oman
Armenia
Senegal
Croatia
China
Jordan
Hungary
Virgin Islands (U.S.)
Cuba
South Africa
Vanuatu
Montenegro
Jamaica
Kuwait
Romania
Timor-Leste
Burkina Faso
Ghana
Solomon Islands
Benin
Guyana
Bulgaria
Belarus
Kazakhstan
Tunisia
Trinidad and Tobago
Vietnam
India
Malawi
Belize
Kosovo
Lesotho
Sri Lanka
Colombia
North Macedonia
Maldives
Gambia, The
Côte d'Ivoire
Tanzania
Türkiye
Suriname
Morocco
Argentina
Ethiopia
Tonga
Indonesia
Sierra Leone
Serbia
Moldova
Thailand
Brazil
Philippines
Mongolia
El Salvador
Nepal
Albania
Niger
Dominican Republic
Panama
Ecuador
Algeria
Angola
Peru
Bosnia and Herzegovina
Togo
Eswatini
Egypt, Arab Rep.
Kenya
Zambia
West Bank and Gaza
Papua New Guinea
Ukraine
Djibouti
Pakistan
Mozambique
Uzbekistan
Mauritania
Azerbaijan
Gabon
Bolivia
Mali
Russian Federation
Liberia
Madagascar
Bangladesh
Mexico
Guinea
Paraguay
Uganda
Myanmar
Lao PDR
Honduras



4. Создание фрейма данных из исходного фрейма для Middle East and North Africa региона

In [91]:
```
# 1 Отфильтруйте фрейм данных, чтобы сохранить только те строки,
# в которых значение в 'column_name' соответствует выбранному значению
merged_df = pd.merge(regions, sorted_wgidataset, on='Code')
df_ap_region = merged_df[merged_df['Region'] == 'AP']
# # Теперь наш набор данных содержит только строки из набора данных wgi,
# которые относятся только к странам Азиатско-Тихоокеанского региона
df_ap_region.head()
```

Out[91]:

	Country	Code	Region	Country/Territory	Estimate	StdErr	NumSrc	Rank	Lowe
0	Afghanistan	AFG	AP	Afghanistan	-1.152327	0.182761	8.0	12.380953	4.76190
1	Afghanistan	AFG	AP	Afghanistan	-1.183776	0.173320	8.0	12.264151	4.71698
2	Afghanistan	AFG	AP	Afghanistan	-1.176012	0.324013	2.0	8.021390	0.00000
3	Afghanistan	AFG	AP	Afghanistan	-1.350647	0.213201	5.0	6.403941	0.00000
4	Afghanistan	AFG	AP	Afghanistan	-1.419499	0.154593	10.0	5.714286	1.42857

5. Выводим данные DataFrame'a

In [95]:
```
# # Теперь наш набор данных содержит только строки из набора данных wgi,
# которые относятся только к странам Азиатско-Тихоокеанского региона
df_ap_region
```

Out[95]:

	Country	Code	Region	Country/Territory	Estimate	StdErr	NumSrc	Rank	L
0	Afghanistan	AFG	AP	Afghanistan	-1.152327	0.182761	8.0	12.380953	4.7
1	Afghanistan	AFG	AP	Afghanistan	-1.183776	0.173320	8.0	12.264151	4.7
2	Afghanistan	AFG	AP	Afghanistan	-1.176012	0.324013	2.0	8.021390	0.0
3	Afghanistan	AFG	AP	Afghanistan	-1.350647	0.213201	5.0	6.403941	0.0
4	Afghanistan	AFG	AP	Afghanistan	-1.419499	0.154593	10.0	5.714286	1.4
...	...	...	...	...	...	...	...	...	
4171	Vietnam	VNM	AP	Vietnam	-0.628040	0.128209	12.0	30.476191	20.0
4172	Vietnam	VNM	AP	Vietnam	-0.711533	0.145354	14.0	28.640778	17.9
4173	Vietnam	VNM	AP	Vietnam	-0.756660	0.161213	9.0	26.600985	15.2
4174	Vietnam	VNM	AP	Vietnam	-0.729582	0.147620	11.0	25.853659	17.0
4175	Vietnam	VNM	AP	Vietnam	-0.747617	0.141613	14.0	25.365854	18.0

720 rows × 11 columns

6. Построим графики индекса WGI за 1996-2021 годы для стран Азиатско-Тихоокеанского региона (rank).

In [94]:
```
# создать DF с 1996 по 2021 год
df_1996_2021 = df_ap_region[['Country/Territory',"Year", "Estimate","Rank"]]
df_1996_2021 = df_1996_2021[df_1996_2021["Year"] != 2022]
df_1996_2021
```

Out[94]:

	Country/Territory	Year	Estimate	Rank
0	Afghanistan	2021	-1.152327	12.380953
2	Afghanistan	1998	-1.176012	8.021390
3	Afghanistan	2004	-1.350647	6.403941
4	Afghanistan	2019	-1.419499	5.714286
5	Afghanistan	2015	-1.354240	5.714286
...	...	...	...	...
4171	Vietnam	2017	-0.628040	30.476191
4172	Vietnam	2008	-0.711533	28.640778
4173	Vietnam	2004	-0.756660	26.600985
4174	Vietnam	2005	-0.729582	25.853659
4175	Vietnam	2006	-0.747617	25.365854

690 rows × 4 columns

In [96]:
```
# Pivot the data frame
pivoted_df = df_1996_2021.pivot_table(columns='Country/Territory', index='Year', va

# Rename the columns
pivoted_df.columns = [f'Estimate.{col}' for col in pivoted_df.columns]

pivoted_df.head()
```

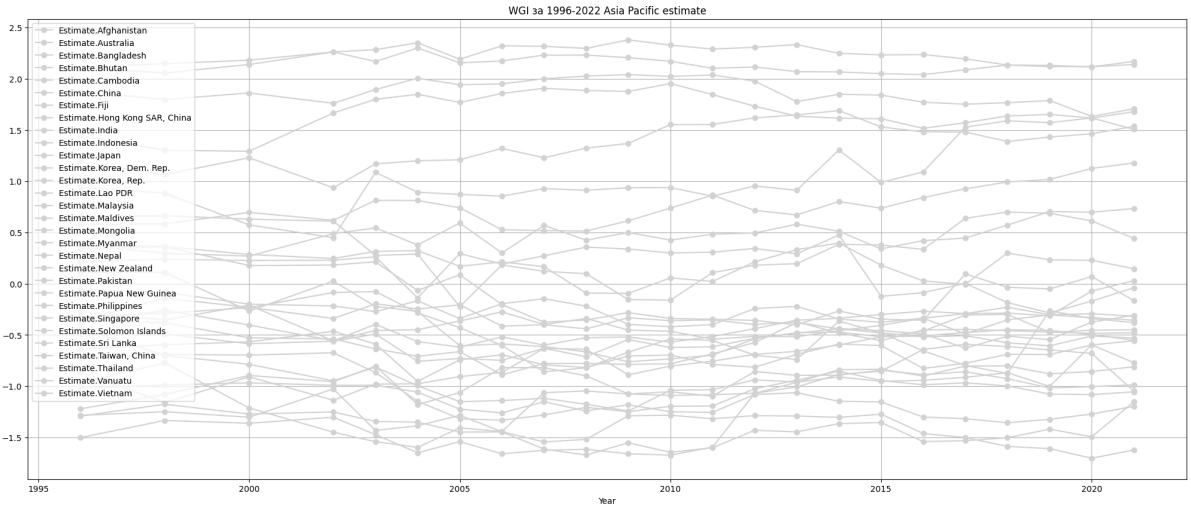
Out[96]:

Year	Estimate.Afghanistan	Estimate.Australia	Estimate.Bangladesh	Estimate.Bhutan	Estimate.Cam
1996	-1.291705	1.877356	-0.969682	0.942838	-1.(
1998	-1.176012	1.798130	-0.773011	0.883641	-0.9
2000	-1.271724	1.862088	-1.212083	0.574340	-0.9
2002	-1.251137	1.761436	-1.449087	0.449922	-0.9
2003	-1.344180	1.895287	-1.541721	1.087011	-0.9

5 rows × 30 columns

In [98]:
```
pivoted_df.plot(grid=1,figsize=(25,10),title='WGI за 1996-2022 Asia Pacific estimat
```

Out[98]:
```
<Axes: title={'center': 'WGI за 1996-2022 Asia Pacific estimate'}, xlabel='Year'>
```



7. Страны с самыми высокими и самыми низкими значениями WGI для варианта "Middle East and North Africa" на 2021 год (rank)

In [102…
```
# Сохранем только строки с 2021 годом
df_2021 = df_1996_2021[df_1996_2021['Year'] == 2021]
# Находим страну с максимальным PVE
max_pve_country = df_2021.loc[df_2021['Estimate'].idxmax()]['Country/Territory']
# Находим страну с минимальным PVE
min_pve_country = df_2021.loc[df_2021['Estimate'].idxmin()]['Country/Territory']
```

In [103…
```
#страна с максимальным PVE
max_pve_country
```

Out[103]:
```
'New Zealand'
```

In [104…
```
#страна с минимальным PVE
min_pve_country
```

Out[104]:
```
'Korea, Dem. Rep.'
```

8. Средние значения по региону за каждый год с 1996 по 2021 год (rank)

In [105…
```
print(df_1996_2021.head())
```

```
  Country/Territory  Year  Estimate       Rank
0       Afghanistan  2021 -1.152327  12.380953
2       Afghanistan  1998 -1.176012   8.021390
3       Afghanistan  2004 -1.350647   6.403941
4       Afghanistan  2019 -1.419499   5.714286
5       Afghanistan  2015 -1.354240   5.714286
```

In [107…
```
# группируем данные по годам и рассчитываем среднее значение для каждого года
average_values_by_year = df_1996_2021.groupby('Year')['Estimate'].mean().reset_inde

print(average_values_by_year)

# average_values_by_year = df_1996_2021.groupby('year').mean().reset_index()
```

```
    Year  Estimate
0   1996  0.005390
1   1998  0.004409
2   2000 -0.035114
3   2002 -0.043584
4   2003 -0.024761
5   2004 -0.117873
6   2005 -0.141014
7   2006 -0.154418
8   2007 -0.139037
9   2008 -0.156720
10  2009 -0.154840
11  2010 -0.140798
12  2011 -0.131607
13  2012 -0.063054
14  2013 -0.046824
15  2014  0.012366
16  2015 -0.049325
17  2016 -0.073050
18  2017 -0.025685
19  2018 -0.012606
20  2019 -0.013031
21  2020  0.016584
22  2021  0.016733
```

In [108…
```
# среднее значение по региону за год
average_values_by_year
```

Out[108]:

	Year	Estimate
0	1996	0.005390
1	1998	0.004409
2	2000	-0.035114
3	2002	-0.043584
4	2003	-0.024761
5	2004	-0.117873
6	2005	-0.141014
7	2006	-0.154418
8	2007	-0.139037
9	2008	-0.156720
10	2009	-0.154840
11	2010	-0.140798
12	2011	-0.131607
13	2012	-0.063054
14	2013	-0.046824
15	2014	0.012366
16	2015	-0.049325
17	2016	-0.073050
18	2017	-0.025685
19	2018	-0.012606
20	2019	-0.013031
21	2020	0.016584
22	2021	0.016733

9. Постройте график индекса WGI за 1996-2021 годы для стран вашего региона и выберите страны с самыми высокими и самыми низкими значениями WGI на 2021 год, а также отобразите среднее значение для региона и Российской Федерации.

In [111…

```
# Российская Федерация
df_russia = wgidataset[wgidataset['Country/Territory'] == "Russian Federation"]
df_russia = df_russia[df_russia['Year'] != 2022]
df_russia = df_russia [['Country/Territory',"Year", "Estimate"]]

# DF страны с максимальным PVE
df_max = df_1996_2021[df_1996_2021['Country/Territory'] == max_pve_country]
# DF страны с минимальным PVE
df_min = df_1996_2021[df_1996_2021['Country/Territory'] == min_pve_country]
```

In [112…

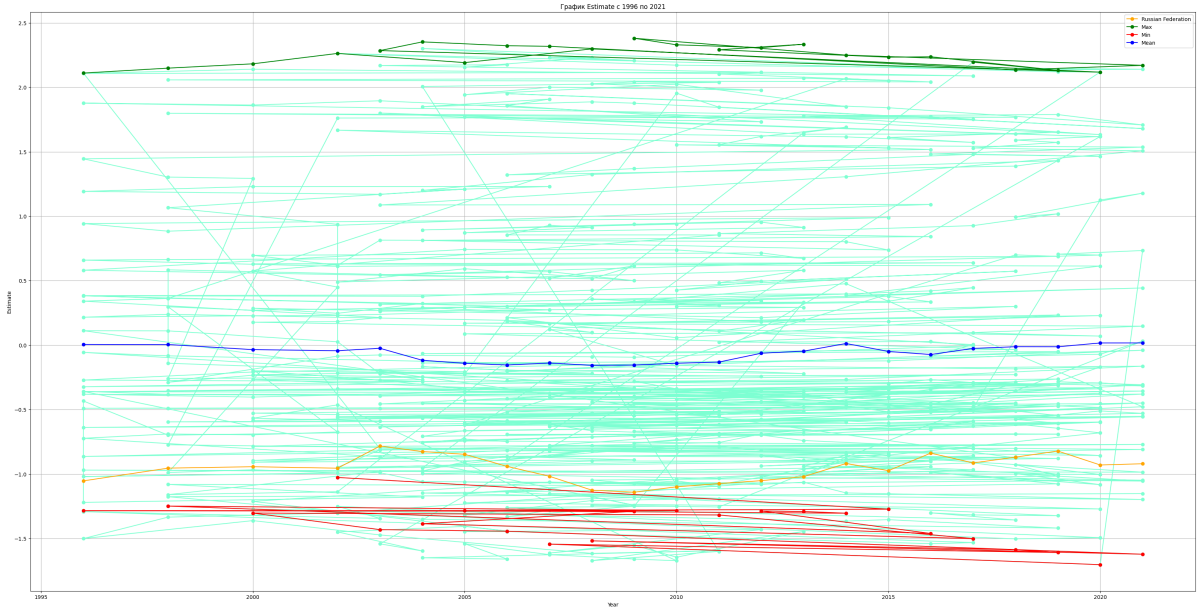
```
df_min
```

Out[112]:

	Country/Territory	Year	Estimate	Rank
2016	Korea, Dem. Rep.	2002	-1.026663	13.756614
2017	Korea, Dem. Rep.	2015	-1.272866	9.047619
2018	Korea, Dem. Rep.	2005	-1.284464	8.292683
2019	Korea, Dem. Rep.	2014	-1.304090	7.211538
2020	Korea, Dem. Rep.	2013	-1.291033	6.635071
2021	Korea, Dem. Rep.	2012	-1.287029	6.161138
2022	Korea, Dem. Rep.	2016	-1.461490	5.714286
2023	Korea, Dem. Rep.	1998	-1.248695	5.347594
2024	Korea, Dem. Rep.	2010	-1.283197	5.238095
2025	Korea, Dem. Rep.	1996	-1.284347	4.838710
2026	Korea, Dem. Rep.	2009	-1.289081	4.784689
2027	Korea, Dem. Rep.	2004	-1.385845	4.433497
2028	Korea, Dem. Rep.	2017	-1.501400	4.285714
2029	Korea, Dem. Rep.	2011	-1.317664	4.265403
2030	Korea, Dem. Rep.	2000	-1.302907	3.723404
2031	Korea, Dem. Rep.	2003	-1.431558	3.703704
2032	Korea, Dem. Rep.	2006	-1.443517	2.439024
2033	Korea, Dem. Rep.	2021	-1.622234	2.380952
2035	Korea, Dem. Rep.	2008	-1.518466	1.941748
2036	Korea, Dem. Rep.	2018	-1.587845	1.904762
2037	Korea, Dem. Rep.	2019	-1.609051	1.904762
2038	Korea, Dem. Rep.	2007	-1.543960	1.456311
2039	Korea, Dem. Rep.	2020	-1.703108	1.428571

In [113…

```
# Рисуем график
plt.figure(figsize=(40, 20))
plt.plot(df_1996_2021['Year'], df_1996_2021['Estimate'], marker='o', linestyle='-',
plt.plot(df_russia['Year'], df_russia['Estimate'], marker='o', linestyle='-',color=
plt.plot(df_max['Year'], df_max['Estimate'], marker='o', linestyle='-',color='green
plt.plot(df_min['Year'], df_min['Estimate'], marker='o', linestyle='-',color='red',
plt.plot(average_values_by_year['Year'], average_values_by_year['Estimate'], marker
plt.xlabel('Year')
plt.ylabel('Estimate')
plt.title('График Estimate с 1996 по 2021')
plt.grid(True)
plt.legend()
plt.show()
```



11. Изменения значения показателя rank с 1996 по 2022 год

In [117…

```
# Создаем DF(Датафрейм) с 1996 по 2021 основанном на rank
df_1996_2021 = wgidataset[["Country/Territory","Year", "Rank"]]
df_1996_2021_rank = df_1996_2021[df_1996_2021["Year"] != 2022]
df_1996_2021_rank
```

Out[117]:

	Country/Territory	Year	Rank
0	Aruba	1996	NaN
1	Andorra	1996	87.096771
2	Afghanistan	1996	4.301075
3	Angola	1996	9.677420
4	Anguilla	1996	NaN
...	...	...	...
4917	Serbia	2021	35.714287
4918	South Africa	2021	53.809525
4919	Congo, Dem. Rep.	2021	4.285714
4920	Zambia	2021	25.714285
4921	Zimbabwe	2021	10.000000

4922 rows × 3 columns

In [119…

```
# Название страны' - это имя столбца, содержащего названия стран в наборе данных
unique_countries = df_1996_2021_rank['Country/Territory'].unique().tolist()
# Инициализируем пустые списки для хранения элементов
percentage_change = []

for country in unique_countries:
  # Рассчитаем процентное изменение
  final_wgi = df_1996_2021_rank.loc[(df_1996_2021_rank['Country/Territory'] == cour
  initial_wgi =  df_1996_2021_rank.loc[(df_1996_2021_rank['Country/Territory'] == c
```

```
    percentage_change.append(final_wgi - initial_wgi)

# Создаём фрейм данных из списков
Changes = pd.DataFrame({'Country': unique_countries, 'Changes': percentage_change})
```

In [120…

```
Changes
```

Out[120]:

	Country	Changes
0	Aruba	[nan]
1	Andorra	[1.4746551513671875]
2	Afghanistan	[8.079877376556396]
3	Angola	[19.37019920349121]
4	Anguilla	[nan]
...	...	...
209	Serbia	[23.886329650878906]
210	South Africa	[-22.534561157226562]
211	Congo, Dem. Rep.	[4.285714149475098]
212	Zambia	[0.9831027984619141]
213	Zimbabwe	[-37.84946060180664]

214 rows × 2 columns

Таблица для (WGI - rank)

In [122…

```
# Сохраняем только 2021 год
df_2021_rank = df_1996_2021[df_1996_2021["Year"] == 2021]

# Расчитываем среднее значение для WGI в 2021 году
df_2021_rank_mean = df_2021_rank['Rank'].mean()

# Сохраняем только 1996 год
df_1996_rank = df_1996_2021[df_1996_2021["Year"] == 1996]

# Расчитываем среднее значение для WGI за 1996 год
df_1996_rank_mean = df_1996_rank['Rank'].mean()

# Расчитываем процентное изменение среднего значения в период с 2021 по 1996 год
changes_mean = (df_2021_rank_mean - df_1996_rank_mean)

# Расчитываем минимум и максимум для WGI в 2021 году
# Находим страну с максимальным значением Rank
max_value_country = df_2021_rank.loc[df_2021_rank['Rank'] == df_2021_rank['Rank'].m
max_value = df_2021_rank.loc[df_2021_rank['Rank'] == df_2021_rank['Rank'].max(), 'R

# Находим страну с минимальным значением var
min_value_country = df_2021_rank.loc[df_2021_rank['Rank'] == df_2021_rank['Rank'].m
min_value = df_2021_rank.loc[df_2021_rank['Rank'] == df_2021_rank['Rank'].min(), 'R

# Российская Федерация
df_russia = wgidataset[wgidataset["Country/Territory"] == "Russian Federation"]
df_russia_2021 = df_russia[df_russia['Year'] == 2021]
```

In [124…
```
# Расчитываем минимум и максимум для WGI в 2021 году
# Находим страну с максимальным значением var
max_value_1996 = df_1996_rank.loc[df_1996_rank['Country/Territory'] == max_value_cc

# Находим страну с минимальным значением var
min_value_1996 = df_1996_rank.loc[df_1996_rank['Country/Territory'] == min_value_cc
# Российская Федерация, 1996
df_russia_1996 = df_russia[df_russia['Year'] == 1996]
```

In [125…
```
# Извлкаем максимальную регион
max_region = regions.loc[regions['Country'] == max_value_country, 'Region'].values[
# Извлкаем минимальный регион
min_region = regions.loc[regions['Country'] == "Korea, North", 'Region'].values[0]
# Извлкаем российский регион
russian_region = regions.loc[regions['Country'] == "Russia", 'Region'].values[0]
```

In [126…
```
# Извлкаем изменения максимального региона
max_region_changes = Changes.loc[Changes['Country'] == max_value_country, 'Changes'
# Извлкаем изменения максимального региона
min_region_changes = Changes.loc[Changes['Country'] == "Korea, Dem. Rep.", 'Changes
# Извлкаем изменения региона России
russian_region_changes = Changes.loc[Changes['Country'] == "Russian Federation", 'C
```

In [128…
```
import pandas as pd

# Создаем списки словарей для каждого столбца
data = {
           '' : ['Mean', 'Max', 'Min', 'Russia'],
    'Region' : ['-', max_region, min_region, russian_region],
    'Country': ['-',max_value_country ,min_value_country , "Russian Federation"],
    'Rank 2021': [df_2021_rank_mean, max_value, min_value,df_russia_2021['Rank'].va
    'Rank 1996': [df_1996_rank_mean,max_value_1996 , min_value_1996, df_russia_1996
    'Changes': [changes_mean, max_region_changes, min_region_changes,russian_regior
}

# # Создаем фрейм данных из словаря
table = pd.DataFrame(data)
# Заполняем таблицу результатами
table
```

Out[128]:

		Region	Country	Rank 2021	Rank 1996	Changes
0	Mean	-	-	50.045137	50.054626	-0.009489
1	Max	WE/EU	Denmark	100.000000	100.000000	[0.0]
2	Min	AP	South Sudan	0.000000	NaN	[-2.4577574729919434]
3	Russia	ECA	Russian Federation	19.523809	15.053763	[4.470046043395996]

13. Отображаем диаграмму размаха (boxplot) индекса WGI за 2021 для всех стран и для каждого региона в отдельности (на одном графике) (estimate)

In [129…
```
# Получаем уникальные регионы
unique_regions = regions['Region'].unique()

# Создаем словарь для хранения фреймов данных для каждого региона
```

```
region_dfs = {}

# Разделяем фрейм данных по регионам
for region in unique_regions:
    region_dfs[region] = regions[regions['Region'] == region].reset_index(drop=True
```

In [130…

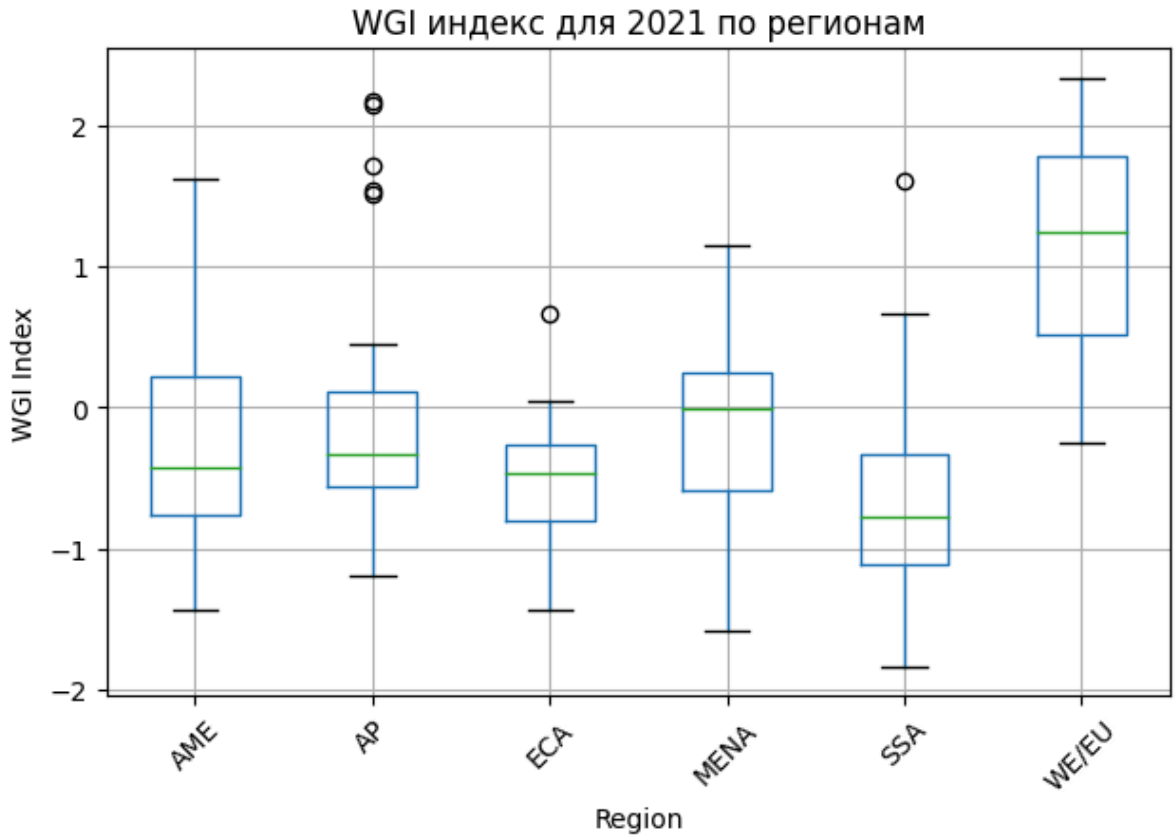
```
# Создаём пустой словарь для хранения фреймов данных каждого региона
DataFrames = {}

df_2021 = wgidataset[wgidataset["Year"] == 2021]
df_2021_estimate = df_2021[['Country/Territory','Estimate']]
df_2021_estimate.rename(columns={'Country/Territory': 'Country'}, inplace=True)

for region, region_df in region_dfs.items():
# Создаём имя переменной на основе индекса
  DataFrames[region] = pd.merge(region_dfs[region], df_2021_estimate, on='Country',
```

```
<ipython-input-130-1911cb9a3ce3>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_2021_estimate.rename(columns={'Country/Territory': 'Country'}, inplace=True)
```

In [132…

```
# Совмещаем все DataFrame-ы в один
df = pd.concat(DataFrames.values())
df
# Чертим boxplot
plt.figure(figsize=(10, 6))
boxplot = df.boxplot(column='Estimate', by='Region')
plt.title('WGI индекс для 2021 по регионам')
plt.suptitle('')
plt.xlabel('Region')
plt.ylabel('WGI Index')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



Задача 2. Анализ рынка акций

Загружаем данные в один dataframe

In [133…

```
import os
import pandas as pd

# Путь к загруженному каталогу, содержащему CSV-файлы
folder_path = '/content/stock'

# Инициализирем пустой фрейм данных для хранения данных
combined_df = pd.DataFrame()

# Выполняем итерацию по каждому файлу в каталоге
for file_name in os.listdir(folder_path):
    # Проверяем, является ли файл CSV-файлом
    if file_name.endswith('.csv'):
        # Считываем CSV-файл во фрейм данных
        file_path = os.path.join(folder_path, file_name)
        df = pd.read_csv(file_path)

        promotion_name = os.path.splitext(file_name)[0]

        # Извлекаем столбцы "Date" и "Close" и устанавливаем "Date" в качестве инде
        df = df[['Date', 'Close']].set_index('Date')

        # Переименуем столбец "Close" в promotion_name
        df.rename(columns={'Close': promotion_name}, inplace=True)

        # Объединяем текущий фрейм данных с объединенным фреймом данных
        combined_df = pd.concat([combined_df, df], axis=1)
```

```
# Отображаем объединенный фрейм данных
combined_df
```

Out[133]:

Date	ADBE	AMZN	NFLX	TCOM	MSFT	GTLB	DBX	NVDA
2022-01-01	534.299988	149.573502	427.140015	26.610001	310.980011	64.010002	24.750000	244.86000
2022-02-01	467.679993	153.563004	394.519989	25.820000	298.790009	58.270000	22.690001	243.85000
2022-03-01	455.619995	162.997498	374.589996	23.120001	308.309998	54.450001	23.250000	272.85998
2022-04-01	395.950012	124.281502	190.360001	23.650000	277.519989	47.930000	21.750000	185.47000
2022-05-01	416.480011	120.209503	197.440002	22.059999	271.869995	38.939999	20.840000	186.72000
2022-06-01	366.059998	106.209999	174.869995	27.450001	256.829987	53.139999	20.990000	151.58999
2022-07-01	410.119995	134.949997	224.899994	25.780001	280.739990	57.400002	22.740000	181.63000
2022-08-01	373.440002	126.769997	223.559998	25.719999	261.470001	59.869999	21.389999	150.94000
2022-09-01	275.200012	113.000000	235.440002	27.309999	232.899994	51.220001	20.719999	121.38999
2022-10-01	318.500000	102.440002	291.880005	22.629999	232.130005	48.459999	21.750000	134.97000
2022-11-01	344.929993	96.540001	305.529999	31.950001	255.139999	39.549999	23.559999	169.22999
2022-12-01	336.529999	84.000000	294.880005	34.400002	239.820007	45.439999	22.379999	146.13999
2023-01-01	370.339996	103.129997	353.859985	36.759998	247.809998	49.410000	23.230000	195.36999
2023-02-01	323.950012	94.230003	322.130005	35.549999	249.419998	44.040001	20.400000	232.16000
2023-03-01	385.369995	103.290001	345.480011	37.669998	288.299988	34.290001	21.620001	277.76998
2023-04-01	377.559998	105.449997	329.929993	35.509998	307.260010	30.360001	20.340000	277.48999
2023-05-01	417.790009	120.580002	395.230011	31.580000	328.390015	36.959999	23.020000	378.33999
2023-06-01	488.989990	130.360001	440.489990	35.000000	340.540009	51.110001	26.670000	423.01998
2023-07-01	546.169983	133.679993	438.970001	41.040001	335.920013	49.630001	26.950001	467.29000
2023-08-01	559.340027	138.009995	433.679993	39.310001	327.760010	47.369999	27.790001	493.54998
2023-09-01	509.899994	127.120003	377.600006	34.970001	315.750000	45.220001	27.230000	434.98999
2023-10-01	532.059998	133.089996	411.690002	34.000000	338.109985	43.279999	26.299999	407.79998

Date	ADBE	AMZN	NFLX	TCOM	MSFT	GTLB	DBX	NVDA
2023-11-01	611.010010	146.089996	473.970001	35.180000	378.910004	48.340000	28.180000	467.70001
2023-12-01	596.599976	151.940002	486.880005	36.009998	376.040009	62.959999	29.480000	495.22000
2024-01-01	617.780029	155.199997	564.109985	36.560001	397.579987	71.110001	31.680000	615.27002
2024-02-01	560.280029	176.759995	602.919983	44.470001	413.640015	72.120003	23.950001	791.11999
2024-03-01	579.140015	175.389999	611.080017	44.279999	415.279999	57.240002	23.840000	919.13000
2024-03-12	579.140015	175.389999	611.080017	44.279999	415.279999	57.240002	23.840000	919.13000

28 rows × 25 columns

Вычисление корреляционной матрицы для всех акций

In [134…

```
# Вычисляем корреляционную матрицу
correlation_matrix = combined_df.corr()

# Отображаем корреляционную матрицу
print(correlation_matrix)
```

	ADBE	AMZN	NFLX	TCOM	MSFT	GTLB	DBX	\
ADBE	1.000000	0.819614	0.821314	0.533298	0.913842	0.496556	0.816359	
AMZN	0.819614	1.000000	0.735466	0.309545	0.838702	0.690644	0.478171	
NFLX	0.821314	0.735466	1.000000	0.766681	0.900263	0.452625	0.635239	
TCOM	0.533298	0.309545	0.766681	1.000000	0.662193	0.103614	0.423136	
MSFT	0.913842	0.838702	0.900263	0.662193	1.000000	0.451366	0.648164	
GTLB	0.496556	0.690644	0.452625	0.103614	0.451366	1.000000	0.402517	
DBX	0.816359	0.478171	0.635239	0.423136	0.648164	0.402517	1.000000	
NVDA	0.802739	0.765294	0.910910	0.787859	0.935386	0.404702	0.519374	
SPOT	0.863827	0.875779	0.920771	0.640120	0.949380	0.540113	0.525305	
GOOGL	0.915440	0.912332	0.717756	0.322718	0.845993	0.535473	0.669228	
META	0.873388	0.830910	0.897908	0.707029	0.966868	0.467641	0.552874	
PINS	0.804657	0.666996	0.930638	0.705551	0.837576	0.525458	0.710191	
XIACY	0.697612	0.654564	0.505430	0.237659	0.565831	0.453669	0.382992	
EBAY	0.180354	0.434078	0.138580	-0.149330	0.127010	0.251066	-0.157363	
ORCL	0.785432	0.534556	0.859397	0.836340	0.847046	0.138574	0.667833	
UBER	0.834611	0.796897	0.937042	0.754442	0.939538	0.521399	0.595928	
AAPL	0.833129	0.665715	0.701937	0.439363	0.790691	0.282373	0.740429	
TWLO	0.067604	0.314869	-0.102302	-0.562073	-0.094023	0.310273	-0.113102	
MU	0.817961	0.906932	0.789551	0.402677	0.849930	0.543109	0.440043	
ABNB	0.670509	0.830690	0.646901	0.294269	0.679204	0.460602	0.332740	
INTC	0.713875	0.816519	0.447049	-0.014994	0.627531	0.535441	0.390625	
TSLA	0.071508	0.302321	-0.251616	-0.586854	-0.117639	0.260908	0.037233	
CSCO	0.554172	0.404820	0.497727	0.257188	0.391476	0.068856	0.496982	
HPQ	0.081518	0.235247	-0.203337	-0.443806	-0.034581	0.094128	-0.177013	
SHOP	0.783919	0.824934	0.852517	0.592950	0.842193	0.855342	0.424923	

	NVDA	SPOT	GOOGL	...	UBER	AAPL	TWLO	\
ADBE	0.802739	0.863827	0.915440	...	0.834611	0.833129	0.067604	
AMZN	0.765294	0.875779	0.912332	...	0.796897	0.665715	0.314869	
NFLX	0.910910	0.920771	0.717756	...	0.937042	0.701937	-0.102302	
TCOM	0.787859	0.640120	0.322718	...	0.754442	0.439363	-0.562073	
MSFT	0.935386	0.949380	0.845993	...	0.939538	0.790691	-0.094023	
GTLB	0.404702	0.540113	0.535473	...	0.521399	0.282373	0.310273	
DBX	0.519374	0.525305	0.669228	...	0.595928	0.740429	-0.113102	
NVDA	1.000000	0.925270	0.715287	...	0.969790	0.633114	-0.244797	
SPOT	0.925270	1.000000	0.821587	...	0.933308	0.687415	0.059969	
GOOGL	0.715287	0.821587	1.000000	...	0.737311	0.806847	0.315410	
META	0.961389	0.973401	0.808784	...	0.954444	0.705358	-0.072886	
PINS	0.815629	0.842858	0.640675	...	0.907751	0.640294	-0.141953	
XIACY	0.445645	0.647331	0.680658	...	0.495835	0.408747	0.447846	
EBAY	0.087027	0.296858	0.375794	...	0.085736	0.115591	0.753732	
ORCL	0.875089	0.763100	0.618983	...	0.832075	0.769309	-0.393536	
UBER	0.969790	0.933308	0.737311	...	1.000000	0.661323	-0.186828	
AAPL	0.633114	0.687415	0.806847	...	0.661323	1.000000	0.042914	
TWLO	-0.244797	0.059969	0.315410	...	-0.186828	0.042914	1.000000	
MU	0.796707	0.902439	0.867191	...	0.820809	0.606787	0.315313	
ABNB	0.649664	0.753797	0.780440	...	0.680764	0.617430	0.429915	
INTC	0.458281	0.645555	0.826042	...	0.512572	0.507251	0.585988	
TSLA	-0.277600	-0.092332	0.326662	...	-0.221155	0.248385	0.703872	
CSCO	0.320159	0.424007	0.600025	...	0.326346	0.589552	0.383777	
HPQ	-0.160502	0.005774	0.263251	...	-0.180970	0.067074	0.728572	
SHOP	0.713391	0.737909	0.824313	...	0.836565	0.465147	0.657843	

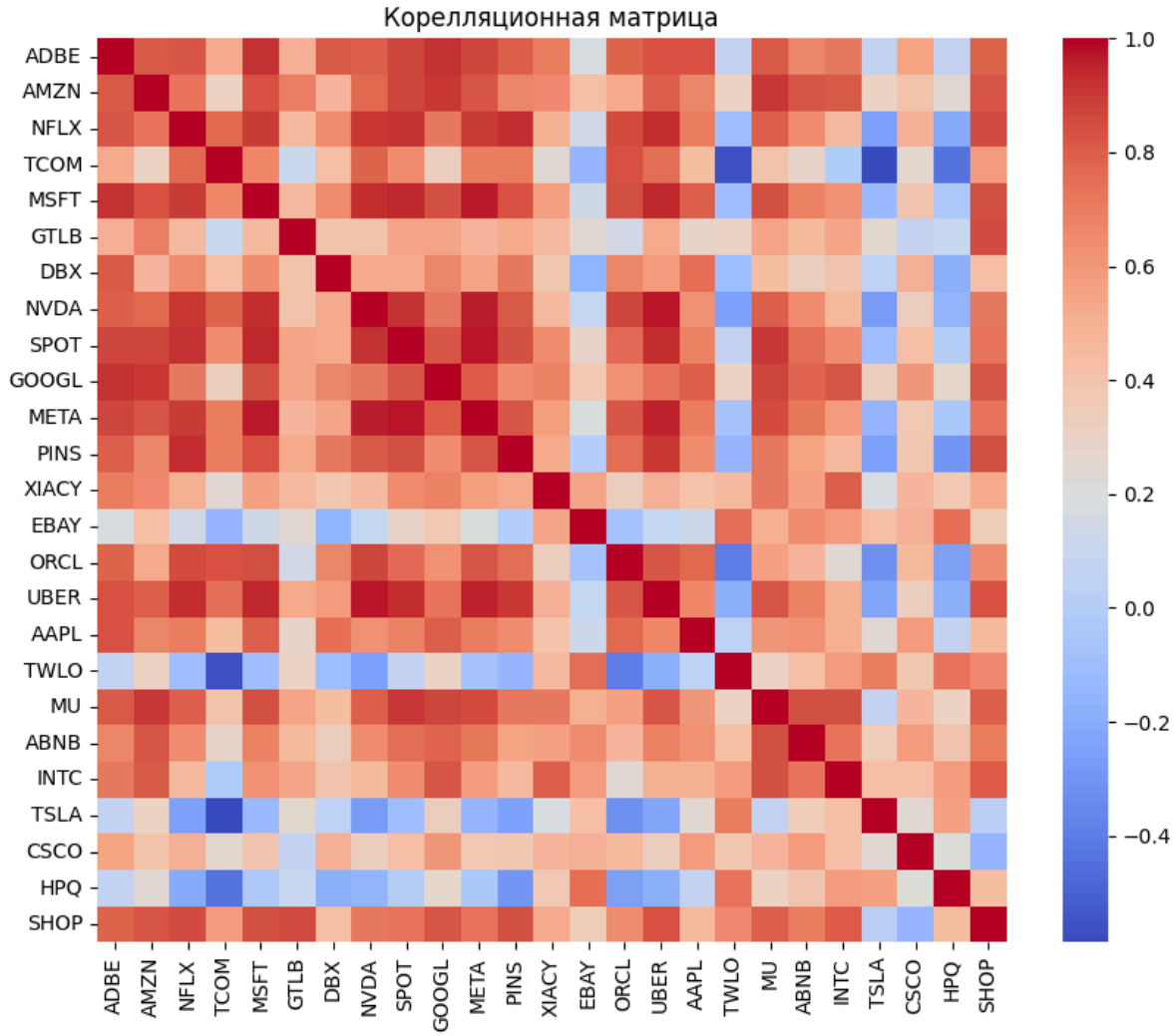
	MU	ABNB	INTC	TSLA	CSCO	HPQ	SHOP
ADBE	0.817961	0.670509	0.713875	0.071508	0.554172	0.081518	0.783919
AMZN	0.906932	0.830690	0.816519	0.302321	0.404820	0.235247	0.824934
NFLX	0.789551	0.646901	0.447049	-0.251616	0.497727	-0.203337	0.852517
TCOM	0.402677	0.294269	-0.014994	-0.586854	0.257188	-0.443806	0.592950
MSFT	0.849930	0.679204	0.627531	-0.117639	0.391476	-0.034581	0.842193
GTLB	0.543109	0.460602	0.535441	0.260908	0.068856	0.094128	0.855342
DBX	0.440043	0.332740	0.390625	0.037233	0.496982	-0.177013	0.424923
NVDA	0.796707	0.649664	0.458281	-0.277600	0.320159	-0.160502	0.713391
SPOT	0.902439	0.753797	0.645555	-0.092332	0.424007	0.005774	0.737909

```
GOOGL  0.867191  0.780440  0.826042  0.326662  0.600025  0.263251  0.824313
META   0.858401  0.723419  0.594611 -0.144519  0.374998 -0.035611  0.735282
PINS   0.717881  0.554616  0.452144 -0.253055  0.384233 -0.285950  0.846115
XIACY  0.720374  0.564475  0.791377  0.184629  0.474311  0.378627  0.519367
EBAY   0.512637  0.644140  0.580047  0.434899  0.494938  0.744560  0.338672
ORCL   0.570765  0.471504  0.239485 -0.310021  0.463955 -0.260316  0.635736
UBER   0.820809  0.680764  0.512572 -0.221155  0.326346 -0.180970  0.836565
AAPL   0.606787  0.617430  0.507251  0.248385  0.589552  0.067074  0.465147
TWLO   0.315313  0.429915  0.585988  0.703872  0.383777  0.728572  0.657843
MU     1.000000  0.842928  0.839860  0.079944  0.472688  0.308473  0.804352
ABNB   0.842928  1.000000  0.738241  0.353807  0.594365  0.390153  0.696599
INTC   0.839860  0.738241  1.000000  0.425236  0.420854  0.591406  0.809582
TSLA   0.079944  0.353807  0.425236  1.000000  0.253808  0.568231  0.025575
CSCO   0.472688  0.594365  0.420854  0.253808  1.000000  0.214262 -0.144612
HPQ    0.308473  0.390153  0.591406  0.568231  0.214262  1.000000  0.436406
SHOP   0.804352  0.696599  0.809582  0.025575 -0.144612  0.436406  1.000000

[25 rows x 25 columns]
```

Отобразите корреляционную матрицу в виде диаграммы.

In [135…

```python
import seaborn as sns
import matplotlib.pyplot as plt


# Рисуем корелляционную матрицу как heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, cmap='coolwarm')
plt.title('Корелляционная матрица')
plt.show()
```



Опроеделение акции с максимальной положительной корреляцией (max)

In [136…

```
# Отфильтруем корреляционную матрицу, чтобы включить только корреляции с Netflix (N
NFLX_correlation = correlation_matrix['NFLX'].drop('NFLX')  # Убираем самокорреляц
# Ищем долю с максимальной положительной корреляцией
max_positive_correlation_stock = NFLX_correlation.idxmax()

print("Самую большую положительную коррелляцию с Netflix (NFLX) имеет:", max_positi
```

```
Самую большую положительную коррелляцию с Netflix (NFLX) имеет: UBER
```

Опроеделение акции с максимальной отрицательной корреляцией (min)

In [137…

```
# Находим часть с максимальной отрицательной корреляцией
max_negative_correlation_stock = NFLX_correlation.idxmin()

print("Самую большую Отрицательную коррелляцию с Netflix (NFLX) имеет:", max_negati
```

```
Самую большую Отрицательную коррелляцию с Netflix (NFLX) имеет: TSLA
```

Опроеделение акции с минимальной корреляцией (которая больше всего

соответствует отсутствию какой-либо корреляции (none)

In [138…

```
# Найдите долю с минимальной корреляцией (ближайшей к нулю)
min_correlation_stock = NFLX_correlation.abs().idxmin()

print("Ищем самую близкую к 0 корелляцию с Netflix:", min_correlation_stock)
```

```
Ищем самую близкую к 0 корелляцию с Netflix: TWLO
```

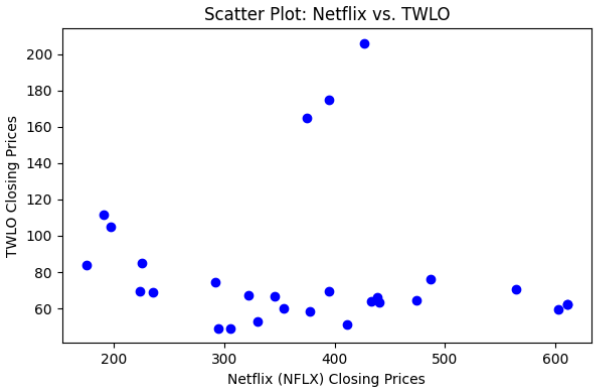
Постройте диаграммы разброса (Ваша компания - Компания с min), (Ваша компания - Компания с max), (Ваша компания - Компания с none)

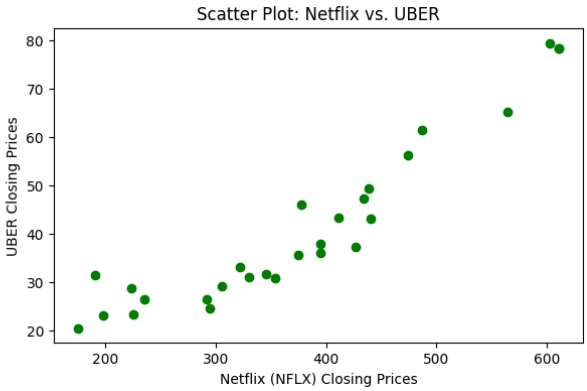
In [139…

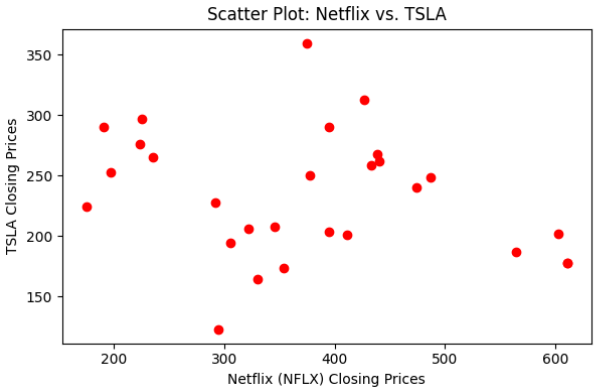
```
# Извлекаем близость Netflix (NFLX) и выбранных компаний
baba_prices = combined_df['NFLX']
max_positive_correlation_prices = combined_df[max_positive_correlation_stock]
max_negative_correlation_prices = combined_df[max_negative_correlation_stock]
min_correlation_prices = combined_df[min_correlation_stock]
```

In [140…

```
# Создание точечных диаграмм
plt.figure(figsize=(12, 8))

# Точечный график для компании с минимальной корреляцией (ближайшей к нулю)
plt.subplot(2, 2, 1)
plt.scatter(baba_prices, min_correlation_prices, color='blue')
plt.title('Scatter Plot: Netflix vs. ' + min_correlation_stock)
plt.xlabel('Netflix (NFLX) Closing Prices')
plt.ylabel(min_correlation_stock + ' Closing Prices')

# Точечный график для компании с максимальной положительной корреляцией
plt.subplot(2, 2, 2)
plt.scatter(baba_prices, max_positive_correlation_prices, color='green')
plt.title('Scatter Plot: Netflix vs. ' + max_positive_correlation_stock)
plt.xlabel('Netflix (NFLX) Closing Prices')
plt.ylabel(max_positive_correlation_stock + ' Closing Prices')

# Точечный график для компании с максимальной отрицательной корреляцией
plt.subplot(2, 2, 3)
plt.scatter(baba_prices, max_negative_correlation_prices, color='red')
plt.title('Scatter Plot: Netflix vs. ' + max_negative_correlation_stock)
plt.xlabel('Netflix (NFLX) Closing Prices')
plt.ylabel(max_negative_correlation_stock + ' Closing Prices')
plt.tight_layout()
plt.show()
```







Рассчитайте среднюю цену акций для каждого месяца (исходные данные взяты с интервалом в месяц)

In [141…

```
# Преобразуем индекс в datetime, если он еще не в формате datetime
combined_df.index = pd.to_datetime(combined_df.index)

# # Повторим выборку фрейма данных с месячными интервалами и вычислим среднее значе
monthly_avg_prices = combined_df.resample('M').mean()

# Отобразим средние цены на акции за каждый месяц
print(monthly_avg_prices)
```

Date	ADBE	AMZN	NFLX	TCOM	MSFT	\
2022-01-31	534.299988	149.573502	427.140015	26.610001	310.980011	
2022-02-28	467.679993	153.563004	394.519989	25.820000	298.790009	
2022-03-31	455.619995	162.997498	374.589996	23.120001	308.309998	
2022-04-30	395.950012	124.281502	190.360001	23.650000	277.519989	
2022-05-31	416.480011	120.209503	197.440002	22.059999	271.869995	
2022-06-30	366.059998	106.209999	174.869995	27.450001	256.829987	
2022-07-31	410.119995	134.949997	224.899994	25.780001	280.739990	
2022-08-31	373.440002	126.769997	223.559998	25.719999	261.470001	
2022-09-30	275.200012	113.000000	235.440002	27.309999	232.899994	
2022-10-31	318.500000	102.440002	291.880005	22.629999	232.130005	
2022-11-30	344.929993	96.540001	305.529999	31.950001	255.139999	
2022-12-31	336.529999	84.000000	294.880005	34.400002	239.820007	
2023-01-31	370.339996	103.129997	353.859985	36.759998	247.809998	
2023-02-28	323.950012	94.230003	322.130005	35.549999	249.419998	
2023-03-31	385.369995	103.290001	345.480011	37.669998	288.299988	
2023-04-30	377.559998	105.449997	329.929993	35.509998	307.260010	
2023-05-31	417.790009	120.580002	395.230011	31.580000	328.390015	
2023-06-30	488.989990	130.360001	440.489990	35.000000	340.540009	
2023-07-31	546.169983	133.679993	438.970001	41.040001	335.920013	
2023-08-31	559.340027	138.009995	433.679993	39.310001	327.760010	
2023-09-30	509.899994	127.120003	377.600006	34.970001	315.750000	
2023-10-31	532.059998	133.089996	411.690002	34.000000	338.109985	
2023-11-30	611.010010	146.089996	473.970001	35.180000	378.910004	
2023-12-31	596.599976	151.940002	486.880005	36.009998	376.040009	
2024-01-31	617.780029	155.199997	564.109985	36.560001	397.579987	
2024-02-29	560.280029	176.759995	602.919983	44.470001	413.640015	
2024-03-31	579.140015	175.389999	611.080017	44.279999	415.279999	

Date	GTLB	DBX	NVDA	SPOT	GOOGL	...	\
2022-01-31	64.010002	24.750000	244.860001	196.259995	135.303497	...	
2022-02-28	58.270000	22.690001	243.850006	156.190002	135.057007	...	
2022-03-31	54.450001	23.250000	272.859985	151.020004	139.067505	...	
2022-04-30	47.930000	21.750000	185.470001	101.650002	114.109497	...	
2022-05-31	38.939999	20.840000	186.720001	112.769997	113.762001	...	
2022-06-30	53.139999	20.990000	151.589996	93.830002	108.962997	...	
2022-07-31	57.400002	22.740000	181.630005	113.019997	116.320000	...	
2022-08-31	59.869999	21.389999	150.940002	108.150002	108.220001	...	
2022-09-30	51.220001	20.719999	121.389999	86.300003	95.650002	...	
2022-10-31	48.459999	21.750000	134.970001	80.580002	94.510002	...	
2022-11-30	39.549999	23.559999	169.229996	79.419998	100.989998	...	
2022-12-31	45.439999	22.379999	146.139999	78.949997	88.230003	...	
2023-01-31	49.410000	23.230000	195.369995	112.720001	98.839996	...	
2023-02-28	44.040001	20.400000	232.160004	116.300003	90.059998	...	
2023-03-31	34.290001	21.620001	277.769989	133.619995	103.730003	...	
2023-04-30	30.360001	20.340000	277.489990	133.600006	107.339996	...	
2023-05-31	36.959999	23.020000	378.339996	148.899994	122.870003	...	
2023-06-30	51.110001	26.670000	423.019989	160.550003	119.699997	...	
2023-07-31	49.630001	26.950001	467.290009	149.410004	132.720001	...	
2023-08-31	47.369999	27.790001	493.549988	153.970001	136.169998	...	
2023-09-30	45.220001	27.230000	434.989990	154.639999	130.860001	...	
2023-10-31	43.279999	26.299999	407.799988	164.759995	124.080002	...	
2023-11-30	48.340000	28.180000	467.700012	185.110001	132.529999	...	
2023-12-31	62.959999	29.480000	495.220001	187.910004	139.690002	...	
2024-01-31	71.110001	31.680000	615.270020	215.350006	140.100006	...	
2024-02-29	72.120003	23.950001	791.119995	256.410004	138.460007	...	
2024-03-31	57.240002	23.840000	919.130005	258.089996	138.500000	...	

Date	UBER	AAPL	TWLO	MU	ABNB	\
2022-01-31	37.400002	174.779999	206.119995	82.269997	153.970001	
2022-02-28	36.029999	165.119995	174.800003	88.860001	151.490005	

2022-03-31	35.680000	174.610001	164.809998	77.889999	171.759995
2022-04-30	31.480000	157.649994	111.820000	68.190002	153.210007
2022-05-31	23.200001	148.839996	105.169998	73.839996	120.870003
2022-06-30	20.459999	136.720001	83.809998	55.279999	89.080002
2022-07-31	23.450001	162.509995	84.800003	61.860001	110.980003
2022-08-31	28.760000	157.220001	69.580002	56.529999	113.120003
2022-09-30	26.500000	138.199997	69.139999	50.099998	105.040001
2022-10-31	26.570000	153.339996	74.370003	54.099998	106.910004
2022-11-30	29.139999	148.029999	49.020000	57.650002	102.139999
2022-12-31	24.730000	129.929993	48.959999	49.980000	85.500000
2023-01-31	30.930000	144.289993	59.840000	60.299999	111.110001
2023-02-28	33.259998	147.410004	67.209999	57.820000	123.279999
2023-03-31	31.700001	164.899994	66.629997	60.340000	124.400002
2023-04-30	31.049999	169.679993	52.610001	64.360001	119.669998
2023-05-31	37.930000	177.250000	69.620003	68.199997	109.769997
2023-06-30	43.169998	193.970001	63.619999	63.110001	128.160004
2023-07-31	49.459999	196.449997	66.029999	71.389999	152.190002
2023-08-31	47.230000	187.869995	63.709999	69.940002	131.550003
2023-09-30	45.990002	171.210007	58.529999	68.029999	137.210007
2023-10-31	43.279999	170.770004	51.259998	66.870003	118.290001
2023-11-30	56.380001	189.949997	64.680000	76.120003	126.339996
2023-12-31	61.570000	192.529999	75.870003	85.339996	136.139999
2024-01-31	65.269997	184.399994	70.330002	85.750000	144.139999
2024-02-29	79.500000	180.750000	59.590000	90.610001	157.470001
2024-03-31	78.320000	173.229996	62.209999	97.419998	166.669998

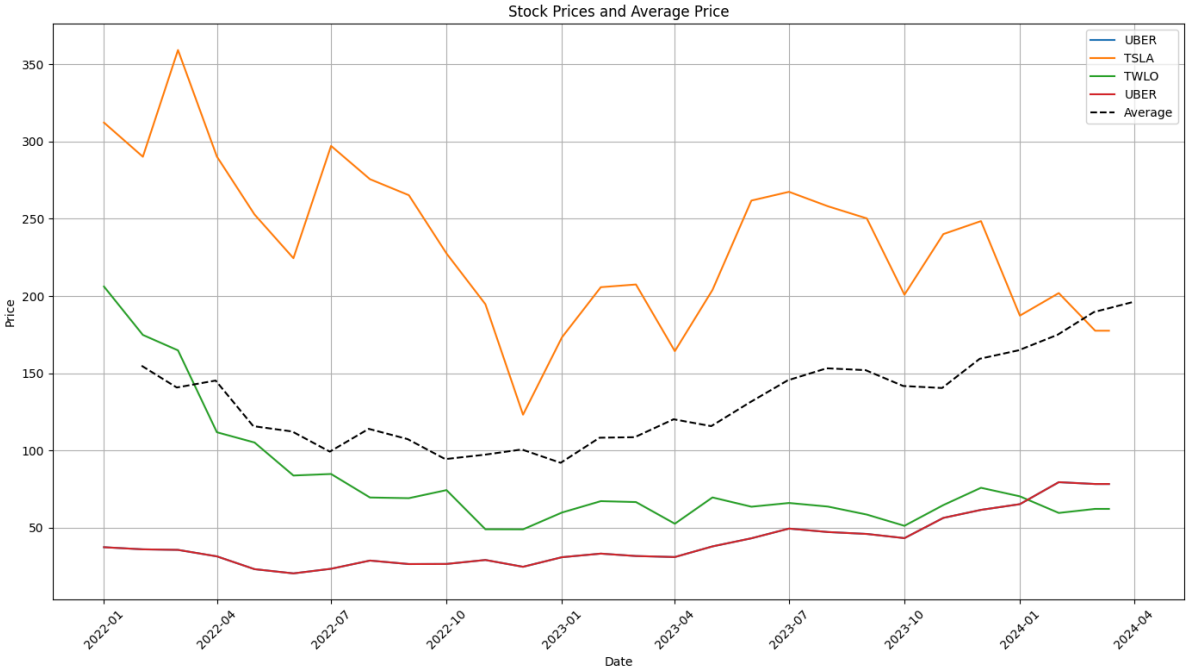
Date	INTC	TSLA	CSCO	HPQ	SHOP
2022-01-31	48.820000	312.239990	55.669998	36.730000	NaN
2022-02-28	47.700001	290.143341	55.770000	34.360001	NaN
2022-03-31	49.560001	359.200012	55.759998	36.299999	NaN
2022-04-30	43.590000	290.253326	48.980000	36.630001	NaN
2022-05-31	44.419998	252.753326	45.049999	38.840000	NaN
2022-06-30	37.410000	224.473328	42.639999	32.779999	NaN
2022-07-31	36.310001	297.149994	45.369999	33.389999	NaN
2022-08-31	31.920000	275.609985	44.720001	28.709999	NaN
2022-09-30	25.770000	265.250000	40.000000	24.920000	NaN
2022-10-31	28.430000	227.539993	45.430000	27.620001	NaN
2022-11-30	30.070000	194.699997	49.720001	30.040001	NaN
2022-12-31	26.430000	123.180000	47.639999	26.870001	NaN
2023-01-31	28.260000	173.220001	48.669998	29.139999	NaN
2023-02-28	24.930000	205.710007	48.419998	29.520000	NaN
2023-03-31	32.669998	207.460007	52.279999	29.350000	NaN
2023-04-30	31.059999	164.309998	47.250000	29.709999	48.450001
2023-05-31	31.440001	203.929993	49.669998	29.059999	57.189999
2023-06-30	33.439999	261.769989	51.740002	30.709999	64.599998
2023-07-31	35.770000	267.429993	52.040001	32.830002	67.580002
2023-08-31	35.139999	258.079987	57.349998	29.709999	66.489998
2023-09-30	35.549999	250.220001	53.759998	25.700001	54.570000
2023-10-31	36.500000	200.839996	52.130001	26.330000	47.189999
2023-11-30	44.700001	240.080002	48.380001	29.340000	72.820000
2023-12-31	50.250000	248.479996	50.520000	30.090000	77.900002
2024-01-31	43.080002	187.289993	50.180000	28.709999	80.070000
2024-02-29	43.049999	201.880005	48.369999	28.330000	76.370003
2024-03-31	45.240002	177.539993	50.070000	30.500000	76.360001

[27 rows x 25 columns]

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

In [142…

```
import matplotlib.pyplot as plt

# Построение графика цен на отдельные акции и средних цен на акции за каждый месяц
plt.figure(figsize=(14, 8))

# Построение графиков цен на отдельные акции
for stock in [max_positive_correlation_stock, max_negative_correlation_stock, min_c
    plt.plot(combined_df.index, combined_df[stock], label=stock)

# Построение графика средних цен на акции за каждый месяц
plt.plot(monthly_avg_prices.index, monthly_avg_prices.mean(axis=1), color='black',

# Добавление метки и заголовок
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Stock Prices and Average Price')
plt.legend()
plt.grid(True)

# рисуем график
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Вывод

В процессе выполнения домашнего задания были изучены библиотеки pandas и matplotlib и их применение в практических задачах на примере задач дескриптивного анализа данных.

Дополнительное задание

- выведите круговую диаграмму (pie) с количеством стран для каждого региона

- постройте гистограмму по estimate за последний год с количеством бинов = 15. добавьте к ней вертикальные линии для среднего и РФ

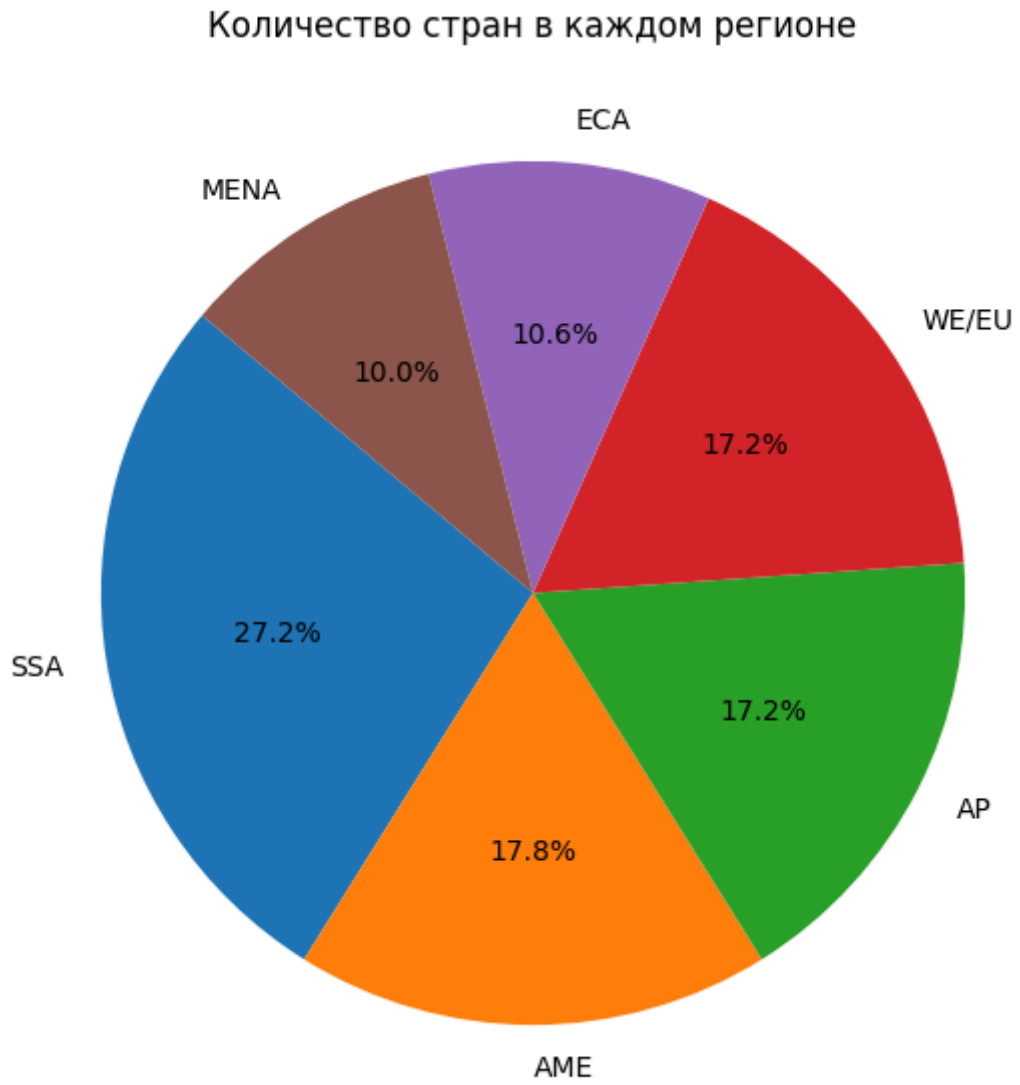
In [143…
```
regions
```

Out[143]:

	Country	Code	Region
0	Afghanistan	AFG	AP
1	Albania	ALB	ECA
2	Algeria	DZA	MENA
3	Angola	AGO	SSA
4	Argentina	ARG	AME
...	...	...	...
175	Venezuela	VEN	AME
176	Vietnam	VNM	AP
177	Yemen	YEM	MENA
178	Zambia	ZMB	SSA
179	Zimbabwe	ZWE	SSA

180 rows × 3 columns

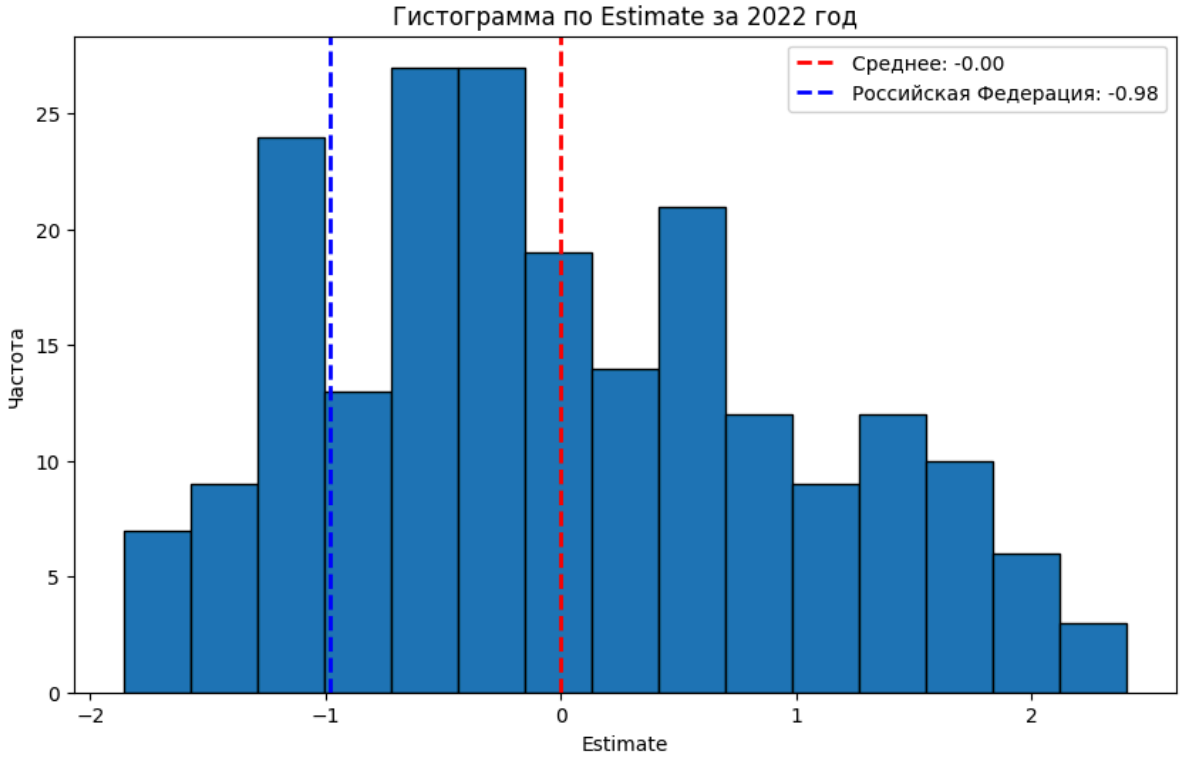
In [144…
```
# Подсчет количества стран в каждом регионе
region_counts = regions['Region'].value_counts()

# Построение круговой диаграммы
plt.figure(figsize=(10, 7))
plt.pie(region_counts, labels=region_counts.index, autopct='%1.1f%%', startangle=14
plt.title('Количество стран в каждом регионе')
plt.show()
```



постройте гистограмму по estimate за последний год с количеством бинов = 15.
добавьте к ней вертикальные линии для среднего и РФ

In [146…

```
# Отфильтровать данные за последний год
last_year = wgidataset['Year'].max()
df_last_year = wgidataset[wgidataset['Year'] == last_year]

# Построить гистограмму по Estimate с количеством бинов = 15
plt.figure(figsize=(10, 6))
n, bins, patches = plt.hist(df_last_year['Estimate'].dropna(), bins=15, edgecolor='

# Добавить вертикальные линии для среднего значения и для значения Russian Federati
mean_estimate = df_last_year['Estimate'].mean()
russian_estimate = df_last_year[df_last_year['Country/Territory'] == 'Russian Feder

plt.axvline(mean_estimate, color='red', linestyle='dashed', linewidth=2, label=f'Cp
plt.axvline(russian_estimate, color='blue', linestyle='dashed', linewidth=2, label=

# Настройка легенды и заголовка
plt.legend()
plt.title(f'Гистограмма по Estimate за {last_year} год')
plt.xlabel('Estimate')
plt.ylabel('Частота')
plt.show()
```


Гистограмма по Estimate за 2022 год
Среднее: -0.00
Российская Федерация: -0.98
Частота
Estimate
0
5
10
15
20
25
−2
−1
0
1
2