[DAN.IT]

FINAL PROJECT

ГАННА САФОНОВА


SQL

БАЗА ДАНИХ ПРАЦІВНИКІВ

1. Покажіть середню зарплату за кожен рік.

```
2 •   USE employees;
3
4 •   SELECT YEAR(s.from_date) AS `Year`,
5           ROUND(AVG(s.salary),2) AS `Average salary`
6       FROM salaries s
7     GROUP BY 1
8     ORDER BY 1;
```

Year	Average salary
1985	53182.36
1986	54084.78
1987	54959.63
1988	55862.45
1989	56840.67
1990	57839.46
1991	58803.87
1992	59758.74
1993	60753.66
1994	61727.76
1995	62681.04
1996	63618.94
1997	64565.43
1998	65540.27
1999	66525.36
2000	68556.28
2001	70694.92

2. Покажіть середню зарплату співробітників у кожному відділу. Примітка: візьміть поточні відділи та поточну зарплату.


```sql
-- option 1
SELECT d.dept_name AS Department,
       ROUND(AVG(s.salary),2) AS 'The average salary of employees'
  FROM salaries s
    INNER JOIN dept_emp de ON (s.emp_no=de.emp_no
                           AND curdate() BETWEEN de.from_date AND de.to_date)
    INNER JOIN departments d ON (de.dept_no=d.dept_no)
  WHERE curdate() BETWEEN s.from_date AND s.to_date
GROUP BY 1
ORDER BY 1;
```

Department	The average salary of employees
Customer Service	67285.23
Development	67657.57
Finance	78555.35
Human Resources	63921.90
Marketing	80058.85
Production	67843.30
Quality Management	65441.99
Research	67913.38
Sales	88853.40

```sql
-- option 2
SELECT DISTINCT q.dept_name AS Department,
                q.avg_salary AS 'The average salary of employees'
FROM
-- finding the average salaries by department for employees
  (SELECT d.dept_name,
     ROUND(AVG(s.salary) OVER(partition BY de.dept_no),2) AS avg_salary
     FROM salaries s
         INNER JOIN dept_emp de ON (s.emp_no=de.emp_no
                                AND curdate() BETWEEN de.from_date AND de.to_date)
         INNER JOIN departments d ON (de.dept_no=d.dept_no)
    WHERE curdate() BETWEEN s.from_date AND s.to_date)q
ORDER BY 1;
```

3. Покажіть середню зарплату працівників у кожному відділі за кожен рік.

Примітка: для середньої зарплати відділу X року Y нам потрібно взяти середнє
значення всіх зарплат співробітників у році Y, які були у відділі X в році Y.


```sql
-- option 1
SELECT d.dept_name AS Department,
       YEAR(s.from_date) AS `Year`,
       ROUND(AVG(s.salary),2) AS 'The average salary of employees'
  FROM salaries s
  INNER JOIN dept_emp de ON (s.emp_no=de.emp_no
                        AND de.to_date>=s.from_date
                        AND de.from_date<=s.from_date)
  INNER JOIN departments d ON (de.dept_no=d.dept_no)
GROUP BY 1, 2
ORDER BY 1, 2;
```

Department	Year	The average salary of employees
Customer Service	1994	54545.30
Customer Service	1995	55781.57
Customer Service	1996	56994.77
Customer Service	1997	58198.64
Customer Service	1998	59475.02
Customer Service	1999	60783.59
Customer Service	2000	63152.06
Customer Service	2001	65601.70
Customer Service	2002	67923.45
Development	1985	48754.84
Development	1986	49736.63
Development	1987	50665.92
Development	1988	51653.61
Development	1989	52596.45

```sql
-- option 2
SELECT DISTINCT q.dept_name AS Department,
                YEAR(q.from_date) AS `YEAR`,
                q.avg_salary AS 'The average salary of employees'
FROM
  (SELECT d.dept_name, s.from_date,
          ROUND(AVG(s.salary) OVER(partition BY de.dept_no,
                                   YEAR(s.from_date)),2) AS avg_salary
     FROM salaries s
       INNER JOIN dept_emp de ON (s.emp_no=de.emp_no
                             AND de.to_date>=s.from_date
                             AND de.from_date<=s.from_date)
       INNER JOIN departments d ON (de.dept_no=d.dept_no)
       )q
;
```

```sql
SELECT s.*,de.*
FROM salaries s
INNER JOIN dept_emp de ON s.emp_no=de.emp_no
WHERE s.emp_no=10010;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

emp_no	salary	from_date	to_date	emp_no	dept_no	from_date	to_date
10010	72488	1996-11-24	1997-11-24	10010	d004	1996-11-24	2000-06-26
10010	74347	1997-11-24	1998-11-24	10010	d004	1996-11-24	2000-06-26
10010	75405	1998-11-24	1999-11-24	10010	d004	1996-11-24	2000-06-26
10010	78194	1999-11-24	2000-11-23	10010	d004	1996-11-24	2000-06-26
10010	79580	2000-11-23	2001-11-23	10010	d004	1996-11-24	2000-06-26
10010	80324	2001-11-23	9999-01-01	10010	d004	1996-11-24	2000-06-26

4. Покажіть для кожного року найбільший відділ цього року та його середню зарплату.


```
91 •  WITH `count_emp_no` AS
92    -- finding for each year, each department the number of employees
93    (SELECT YEAR(from_date) AS `year`,
94            dept_name, COUNT(emp_no)AS count_empl
95       FROM dept_emp de
96       INNER JOIN departments d ON (de.dept_no=d.dept_no)
97     GROUP BY 1, 2)

100   SELECT DISTINCT q.`year`, q.dept_name AS department,  q1.average_salary
101     FROM
102     -- finding for each year, each department max number of employees
103       (SELECT *, MAX(count_empl)OVER(PARTITION BY `year`) AS max
104           FROM `count_emp_no`)q
105
106   INNER JOIN
107
108   -- finding for each year, each department, average salary
109   (SELECT YEAR(s.from_date) AS `year`, d.dept_name,
110           ROUND(AVG(s.salary),2) AS average_salary
111     FROM salaries s
112     INNER JOIN dept_emp de ON (s.emp_no=de.emp_no
113                           AND de.to_date>=s.from_date
114                           AND de.from_date<=s.from_date)
115     INNER JOIN departments d ON (de.dept_no=d.dept_no)
116    GROUP BY 1, 2) q1
117                                       ON (q.dept_name=q1.dept_name
118                                       AND q.`year`=q1.`year`)
119     WHERE count_empl=max
120   ORDER BY 1;
```

	year	department	average_salary
▶	1985	Development	48754.84
	1986	Development	49736.63
	1987	Development	50665.92
	1988	Development	51653.61
	1989	Development	52596.45
	1990	Development	53573.60
	1991	Development	54534.45
	1992	Development	55438.41
	1993	Development	56402.30
	1994	Development	57361.48
	1995	Development	58311.05
	1996	Development	59250.78
	1997	Development	60179.83
	1998	Development	61135.43
	1999	Development	62093.12
	2000	Production	64404.01
	2001	Production	66534.84
	2002	Production	68422.63
	2023	Finance	50001.00

5. Покажіть детальну інформацію про поточного менеджера, який найдовше виконує свої обов'язки.


```
125  SELECT q1.emp_no, q1.birth_date, q1.first_name,
126         q1.last_name, q1.gender, q1.hire_date,
127         q1.dept_no, q1.dept_name, q1.title,
128         q1.from_date, q1.to_date, q1.salary,
129         TIMESTAMPDIFF(Month,q1.from_date,curdate()) AS 'months, was employed'
130    FROM
131        (SELECT e.*,d.dept_no, d.dept_name, q.from_date,
132             q.to_date, s.salary, t.title, q.diff,
133             MAX(q.diff) OVER() AS  max
134         FROM  employees e
135        INNER JOIN salaries s ON (e.emp_no=s.emp_no
136                              AND curdate() BETWEEN s.from_date AND s.to_date)
137
138        INNER JOIN titles t ON (s.emp_no=t.emp_no
139                            AND curdate() BETWEEN t.from_date AND t.to_date)
140
141        INNER JOIN (SELECT *, DATEDIFF(curdate(),from_date) AS diff
142                      FROM dept_manager
143                     WHERE curdate() BETWEEN from_date AND to_date
144                                                          )q
145                                                    ON (t.emp_no=q.emp_no)
146        INNER JOIN departments d ON (q.dept_no=d.dept_no)
147                                                      )q1
148    WHERE q1.diff=q1.max
149    ;
```

emp_no	birth_date	first_name	last_name	gender	hire_date	dept_no	dept_name	title	from_date	to_date	salary	months was employed
110114	1957-03-28	Isamu	Legleitner	F	1985-01-14	d002	Finance	Manager	1989-12-17	9999-01-01	83457	403

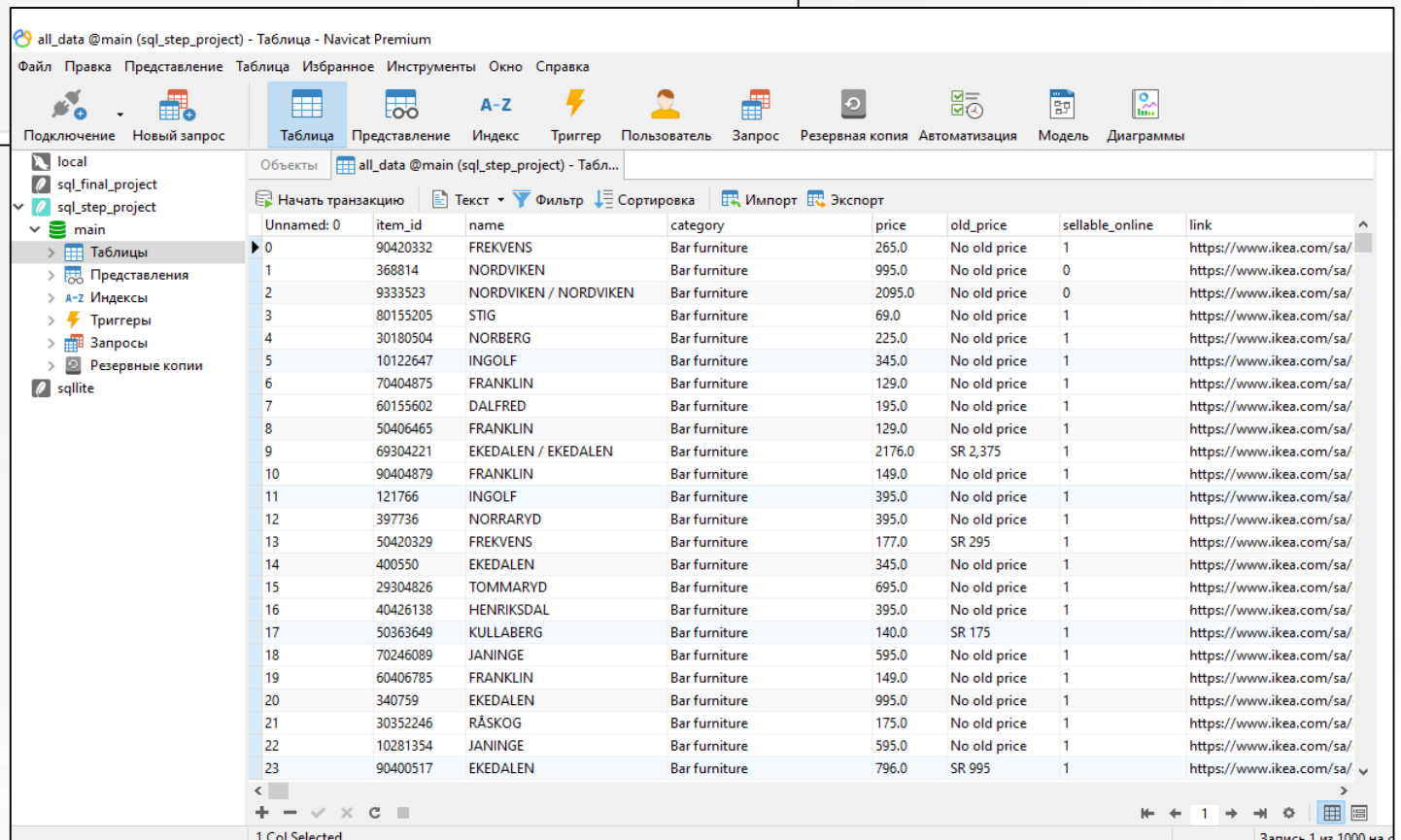

python

ЗАВАНТАЖЕННЯ БАЗИ ДАНИХ МЕБЛІВ ІКЕА

[1]:
```
import requests
import pandas as pd

data = requests.get(
    'https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-11-03/ikea.csv').text

with open('dataset.csv', 'w', encoding="utf-8") as f:#'w' означає відкривається для запису
    f.write(data)
data = pd.read_csv('dataset.csv', sep=',')
# Показує скільки рядків та колонок має набір даних
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3694 entries, 0 to 3693
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          3694 non-null   int64
 1   item_id             3694 non-null   int64
 2   name                3694 non-null   object
 3   category            3694 non-null   object
 4   price               3694 non-null   float64
 5   old_price           3694 non-null   object
 6   sellable_online     3694 non-null   bool
 7   link                3694 non-null   object
 8   other_colors        3694 non-null   object
 9   short_description   3694 non-null   object
 10  designer            3694 non-null   object
 11  depth               2231 non-null   float64
 12  height              2706 non-null   float64
 13  width               3105 non-null   float64
dtypes: bool(1), float64(4), int64(2), object(7)
memory usage: 378.9+ KB
```

[4]:
```python
import sqlite3
conn = sqlite3.connect('sql_step_project.db', check_same_thread=False, )
cursor = conn.cursor()

def delete_table():
    cursor.execute("DROP TABLE IF EXISTS all_data")
delete_table()

columns = ','.join(['"' + col + '"' + ' ' + 'TEXT' for col in data.columns])

def creationandfilingDB():
    cursor.execute(f"""CREATE TABLE IF NOT EXISTS all_data ({columns})""")

    for x in data.values:
        cursor.execute("""INSERT INTO all_data VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)""",(x))
        conn.commit()

# Створення та наповнення бази даних
creationandfilingDB()
```



Unnamed: 0	item_id	name	category	price	old_price	sellable_online	link
0	90420332	FREKVENS	Bar furniture	265.0	No old price	1	https://www.ikea.com/sa/
1	368814	NORDVIKEN	Bar furniture	995.0	No old price	0	https://www.ikea.com/sa/
2	9333523	NORDVIKEN / NORDVIKEN	Bar furniture	2095.0	No old price	0	https://www.ikea.com/sa/
3	80155205	STIG	Bar furniture	69.0	No old price	1	https://www.ikea.com/sa/
4	30180504	NORBERG	Bar furniture	225.0	No old price	1	https://www.ikea.com/sa/
5	10122647	INGOLF	Bar furniture	345.0	No old price	1	https://www.ikea.com/sa/
6	70404875	FRANKLIN	Bar furniture	129.0	No old price	1	https://www.ikea.com/sa/
7	60155602	DALFRED	Bar furniture	195.0	No old price	1	https://www.ikea.com/sa/
8	50406465	FRANKLIN	Bar furniture	129.0	No old price	1	https://www.ikea.com/sa/
9	69304221	EKEDALEN / EKEDALEN	Bar furniture	2176.0	SR 2,375	1	https://www.ikea.com/sa/
10	90404879	FRANKLIN	Bar furniture	149.0	No old price	1	https://www.ikea.com/sa/
11	121766	INGOLF	Bar furniture	395.0	No old price	1	https://www.ikea.com/sa/
12	397736	NORRARYD	Bar furniture	395.0	No old price	1	https://www.ikea.com/sa/
13	50420329	FREKVENS	Bar furniture	177.0	SR 295	1	https://www.ikea.com/sa/
14	400550	EKEDALEN	Bar furniture	345.0	No old price	1	https://www.ikea.com/sa/
15	29304826	TOMMARYD	Bar furniture	695.0	No old price	1	https://www.ikea.com/sa/
16	40426138	HENRIKSDAL	Bar furniture	395.0	No old price	1	https://www.ikea.com/sa/
17	50363649	KULLABERG	Bar furniture	140.0	SR 175	1	https://www.ikea.com/sa/
18	70246089	JANINGE	Bar furniture	595.0	No old price	1	https://www.ikea.com/sa/
19	60406785	FRANKLIN	Bar furniture	149.0	No old price	1	https://www.ikea.com/sa/
20	340759	EKEDALEN	Bar furniture	995.0	No old price	1	https://www.ikea.com/sa/
21	30352246	RÅSKOG	Bar furniture	175.0	No old price	1	https://www.ikea.com/sa/
22	10281354	JANINGE	Bar furniture	595.0	No old price	1	https://www.ikea.com/sa/
23	90400517	EKEDALEN	Bar furniture	796.0	SR 995	1	https://www.ikea.com/sa/

кількість_записів @main (sql_step_project) - Запрос - Navicat Premium

Файл Правка Представление Запрос Формат Избранное Инструменты Окно Справка

Подключение | Новый запрос | Таблица | Представление | Индекс | Триггер | Пользователь | Запрос | Резервная копия | Автоматизация | Модель | Диаграммы

local
sql_final_project
sql_step_project
main
Таблицы
Представления
Индексы
Триггеры
Запросы
ROUND
Вилучення_No_old_pr
Змінити_тип_на_float_
кількість_записів
копіювання існуючих
Почистити_disigner
Прибрати_кому_та_р
Створення таблиці бе
Створення таблиці з д
Створення таблиці_ п
Створення таблиця д
Резервные копии
sqllite

Объекты | all_data @main (sql_step_project) - Табл... | кількість_записів @main (sql_step_proj...

Сохранить | Конструктор запросов | Форматировать SQL | Сниппет | Результат экспорта

sql_step_project | main | Запуск | Остановить | Объяснить

```sql
SELECT count(DISTINCT item_id)
FROM all_data;  --2962

SELECT count(DISTINCT name)
FROM all_data;--607

SELECT count(DISTINCT category)
FROM all_data;--17

SELECT count(DISTINCT price)
FROM all_data;--979

SELECT COUNT(*)
FROM all_data
WHERE old_price='No old price'    -- 3040

SELECT count(*)
FROM all_data
WHERE sellable_online='0';--28

SELECT count(DISTINCT link)
FROM all_data;--2962

SELECT count(*)
FROM all_data
WHERE other_colors='Yes';--1512

SELECT count(DISTINCT short_description)
FROM all_data;--1706

SELECT count(*)
FROM all_data--381
WHERE designer like '%0%' OR designer like '%9%';--143

```

Все ярлыки

COMMENTS Комментар
Create a comment

INSERT Syntax DML
Insert new rows into an existi table

SELECT Syntax DML
Retrieve rows selected from o more tables

UPDATE Syntax DML
Updates columns of existing r in the named table with new values

Поиск

Завершение кода готово. (Последнее обновление: 2023-07-24 10:56) Update Now

ОЦІНКА ЯКОСТІ НАБОРУ ДАНИХ

Data Quality Criteria	*Data Quality Metrics*	*Data Quality Score*
Accuracy	Error rate	0.3%
Completeness	Field completeness	12.96%
Consistency	Field consistency	wasn't found
Validity	Format validation	0.03%
Uniqueness	Duplication rate	19.82%

Error rate: відсоток записів з неправильними даними. Неправильні дані є у стовпчику '**designer**' -143

Field completeness: відсоток записів, які містять неповні дані для всіх полів.
Такі дані містять стовпчики '**old_price**'- 3040, '**designer**'-143 (ті, що з помилкою), **'depth'** – 1463, **'height'** – 988, **'width'** – 589

Field consistency: відсоток записів, які містять суперечливі, непослідовні дані для всіх полів.
Такі дані не були виявлені.

Format validation: відсоток записів, які містять неочікуваний формат даних для всіх полів.
Такі дані містить стовпчик '**old_price**' – 10

Duplication rate: відсоток записів, які містять дублікати. Таблиця містить 732 дублікати.

ДОСЛІДНИЦЬКИЙ АНАЛІЗ ДАНИХ

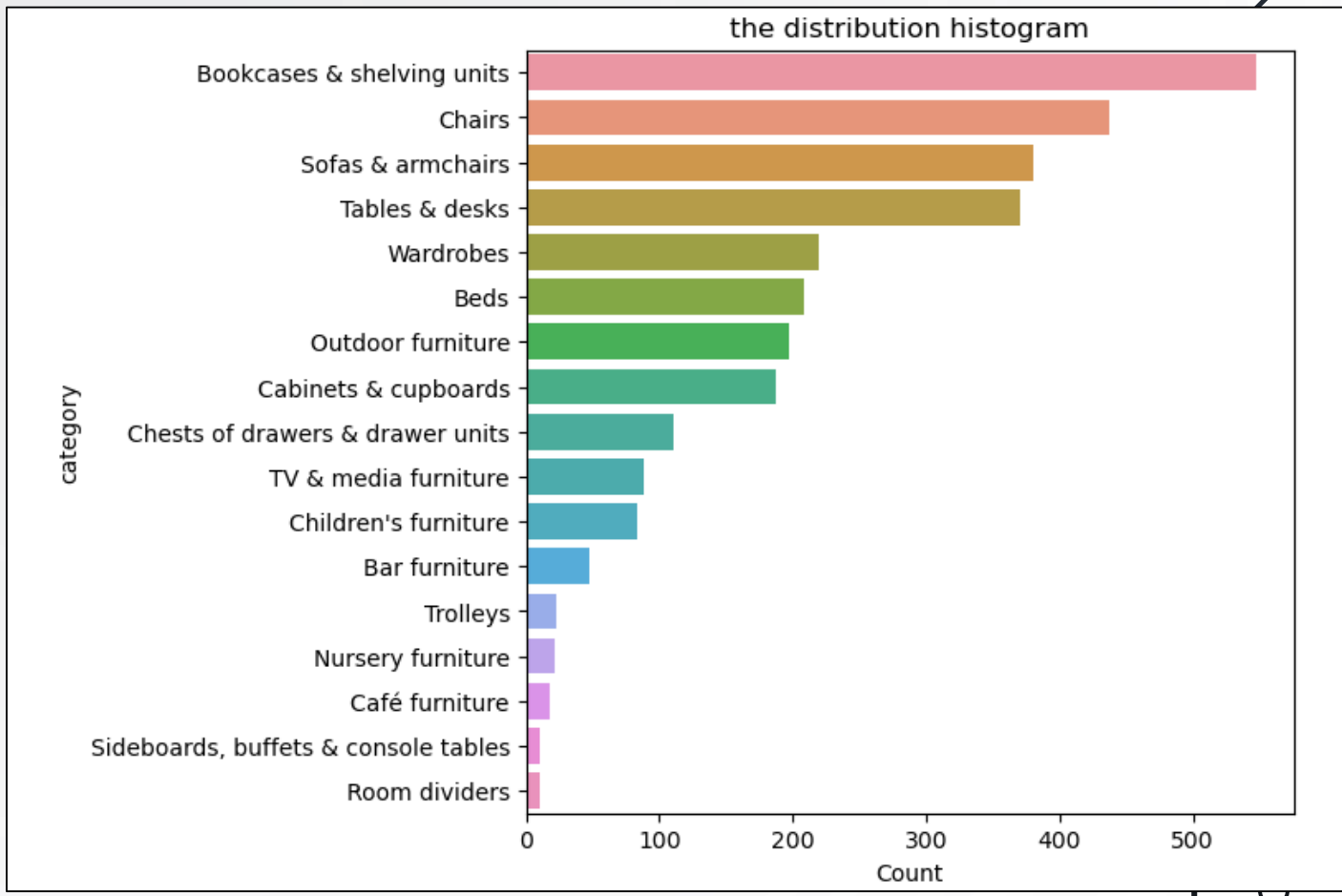
Category

```
# запит вибору колонки category
cursor.execute("""SELECT category
                  FROM all_data_without_duplicate""")
exer1= cursor.fetchall()
exer1=pd.DataFrame(exer1,columns=['category'])
# описові статистики множини category
print(exer1.describe())
```

```
                          category
count                         2962
unique                          17
top     Bookcases & shelving units
freq                           548
```

```
# кількість товарів в кожній category
count_category = exer1['category'].value_counts()
# сортування за спаданням
count_category = count_category.sort_values(ascending=False)
```

```
#побудова гістограми category
import matplotlib.pyplot as plt
import seaborn as sns
fig,ax=plt.subplots(figsize=(6,6))
sns.barplot(x=count_category.values,y=count_category.index)
ax.set_ylabel('category')
ax.set_xlabel('Count')
ax.set_title('the distribution histogram')
```



Price

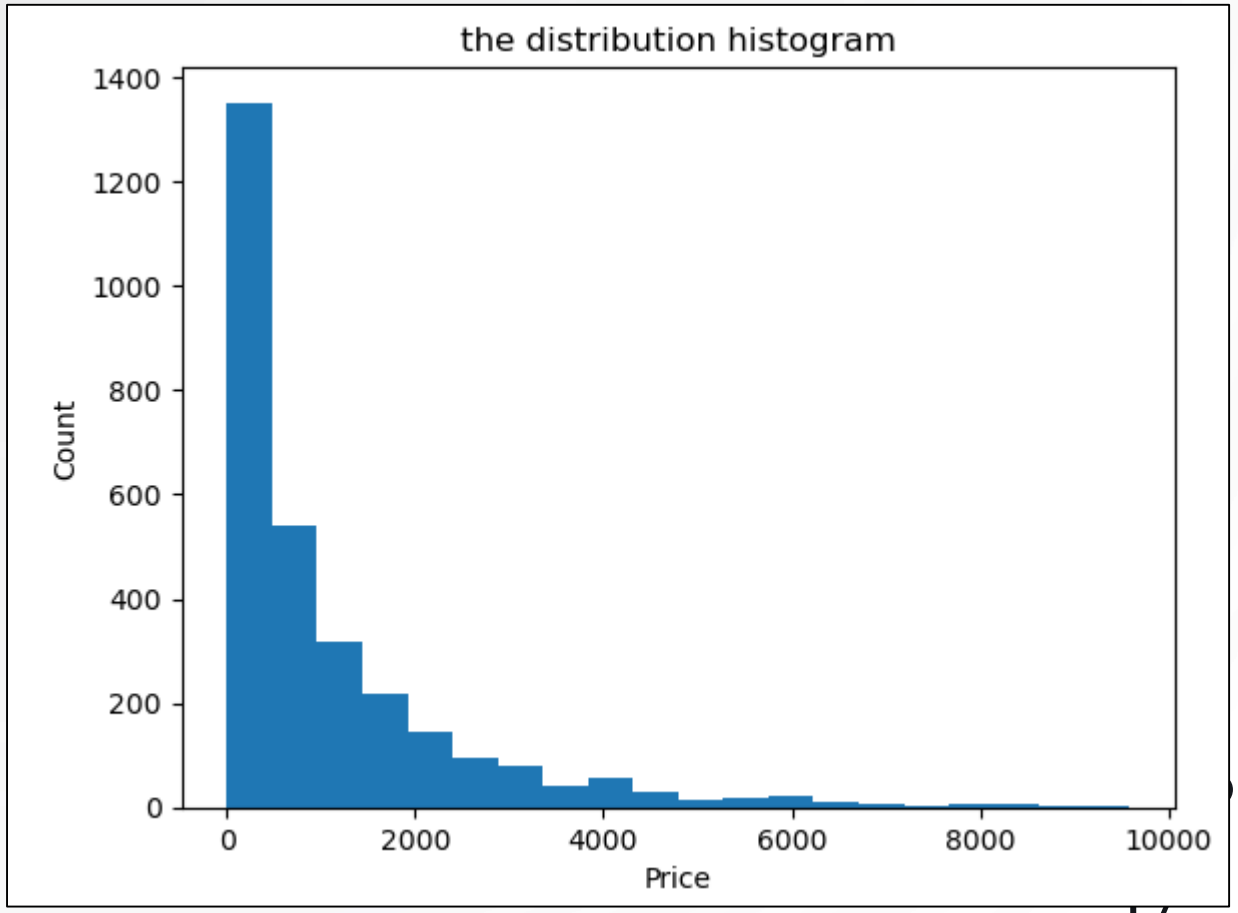
```
# запит вибору колонки price
cursor.execute("""SELECT price
                  FROM all_data_without_duplicate""")
exer2= cursor.fetchall()
exer2=pd.DataFrame(exer2,columns=['price'])
exer2['price'] = exer2['price'].astype(float)
# описові статистики множини price
print(exer2.describe().round(2))
```

```
          price
count  2962.00
mean   1108.72
std    1393.58
min       3.00
25%     200.00
50%     570.00
75%    1475.00
max    9585.00
```

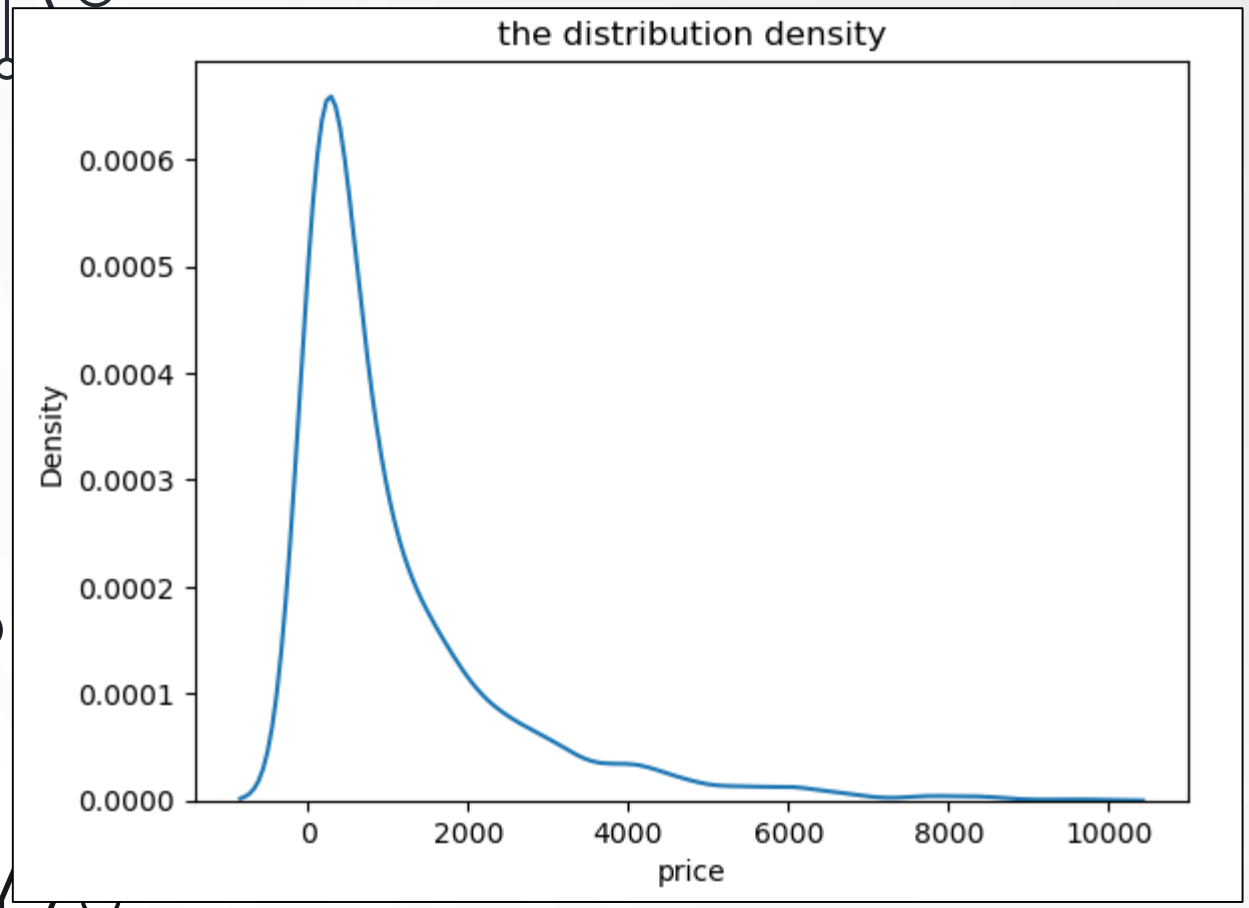
```
# знайдемо mode для price
from statistics import mode

mode = mode(exer2['price'])
print('mode ',mode)
```

```
mode  395.0
```

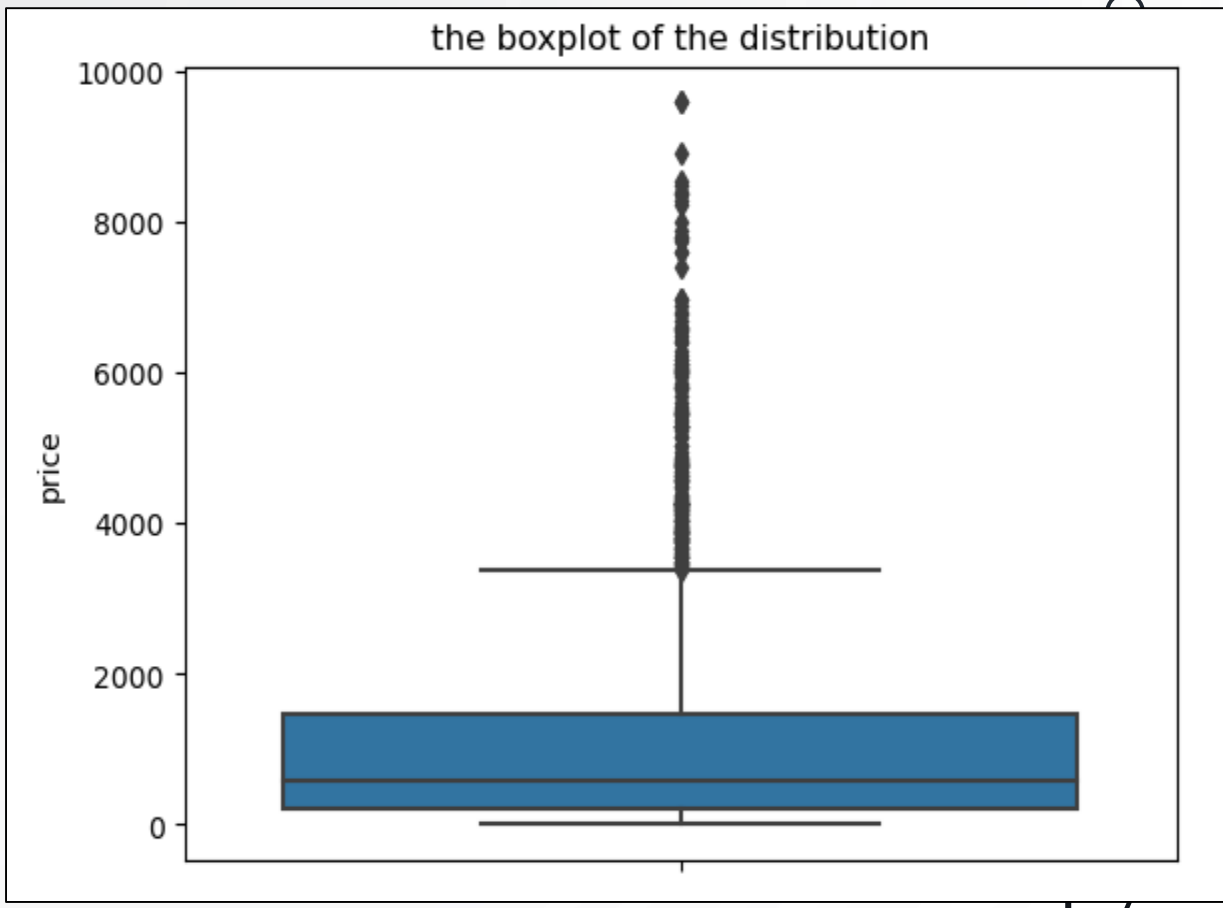
```
fig,ax=plt.subplots()
ax.hist(exer2['price'], bins=20)
ax.set_xlabel('Price')
ax.set_ylabel('Count')
ax.set_title('the distribution histogram')
```



```python
#графік щільності price
fig,ax=plt.subplots()
sns.kdeplot(exer2['price'])
ax.set_title('the distribution density')
plt.show()
```




```python
#побудова boxplot price
fig,ax=plt.subplots()
sns.boxplot(y ='price', data=exer2)
ax.set_title('the boxplot of the distribution')
plt.show()
```



ВЗАЄМОДІЯ ЗМІННИХ

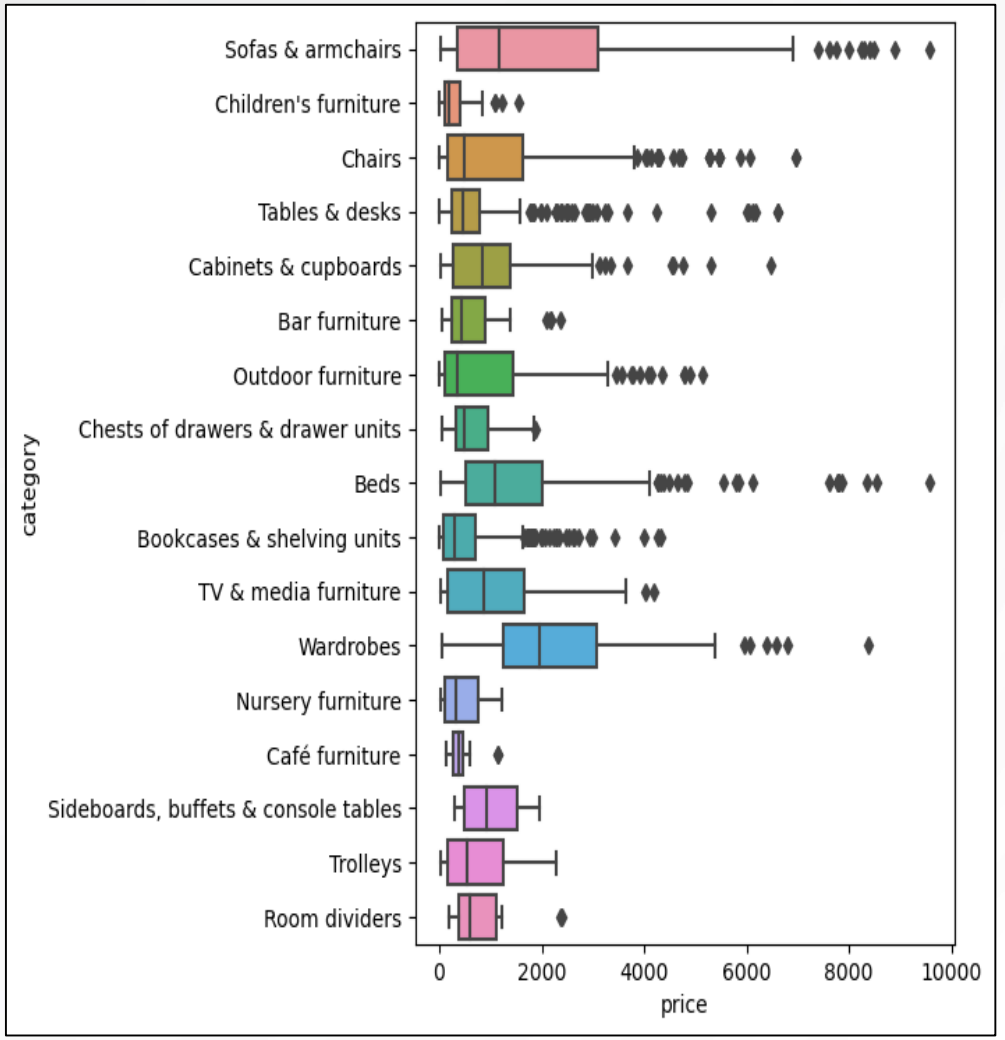
Category vs price

```
# запит вибору колонок category, price
cursor.execute("""SELECT category, price
                  FROM all_data_without_duplicate""")
exer3= cursor.fetchall()
exer3=pd.DataFrame(exer3,columns=['category','price'])
exer3['price'] = exer3['price'].astype(float)
# опис колонки price за category
print(exer3.groupby(['category'])['price'].mean().round(2))
```

category	
Bar furniture	679.55
Beds	1647.43
Bookcases & shelving units	519.42
Cabinets & cupboards	1044.82
Café furniture	426.72
Chairs	1097.12
Chests of drawers & drawer units	657.49
Children's furniture	286.18
Nursery furniture	431.77
Outdoor furniture	919.76
Room dividers	912.60
Sideboards, buffets & console tables	1013.00
Sofas & armchairs	1968.16
TV & media furniture	1045.65
Tables & desks	760.13
Trolleys	748.87
Wardrobes	2249.02

Name: price, dtype: float64

```
#побудова boxplot price за category
plt.subplots(figsize=(5,7))
sns.boxplot(x='price',y ='category', data=exer3)
plt.show()
```



ПРИПУЩЕННЯ ПРО ЗНАЧНИЙ ВПЛИВ ФАКТОРНИХ ЗМІННИХ НА ЦІНОУТВОРЕННЯ

Нульова гіпотеза: немає суттєвої впливу змінних ‘category’ та ‘designer’ на змінну ‘price’

Альтернативна гіпотеза: є суттєвий вплив факторних змінних на ціноутворення

тест ANOVA залежності ціни від двох факторів 'category','designer'

```
# середнє значення ціни за кожним дизайнером по кожній категорії
mean_des_in_categor=exer7.groupby(['category','designer'])['price'].mean().round(2)
print(mean_des_in_categor)
```

```
category       designer
Bar furniture  Carina Bengs                                370.00
               Ehlén Johansson                            1149.89
               Francis Cayouette                          1121.67
               Henrik Preutz                                69.00
               IKEA of Sweden                              257.50
                                                             ...
Wardrobes      L Hilland/J Karlsson                        945.00
               Ola Wihlborg                                648.67
               Ola Wihlborg/Ehlén Johansson/IKEA of Sweden 3935.00
               Ola Wihlborg/IKEA of Sweden                1397.95
               T Winkel/T Jacobsen                        1270.00
Name: price, Length: 558, dtype: float64
```

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Задаємо формулу для моделі
formula = 'price ~ C(category) + C(designer) + C(category):C(designer)'

# Створюємо модель та обчислюємо ANOVA
model = ols(formula, data=exer6).fit()
table = sm.stats.anova_lm(model, typ=2)

# Виводимо результати ANOVA
print(table)
```

	sum_sq	df	F	PR(>F)
C(category)	-1.459431e+01	16.0	-9.353310e-07	1.000000e+00
C(designer)	-6.537808e+09	278.0	-2.411510e+01	1.000000e+00
C(category):C(designer)	2.232818e+10	4448.0	5.147428e+00	1.266065e-168
Residual	2.244935e+09	2302.0	NaN	NaN

Category vs designer

```
# запит вибору колонок category, COUNT_empty_value, де пусті значеня designer
cursor.execute("""SELECT category , COUNT(designer) AS COUNT_empty_value
                   FROM all_data_without_duplicate
                  WHERE designer=''
                  GROUP BY category
                  ORDER BY COUNT_empty_value DESC""")
exer6= cursor.fetchall()
exer6=pd.DataFrame(exer6,columns=['category','COUNT_empty_value'])
```

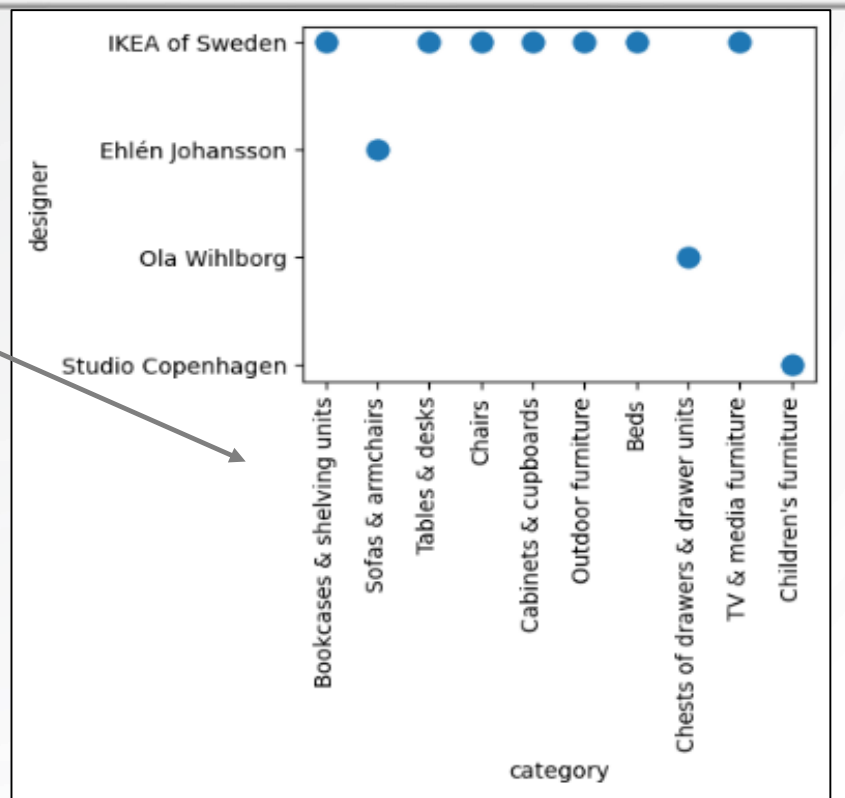
	category	COUNT_empty_value
0	Bookcases & shelving units	26
1	Chairs	23
2	Sofas & armchairs	20
3	Tables & desks	14
4	Cabinets & cupboards	5
5	Beds	5
6	Children's furniture	4
7	TV & media furniture	3
8	Outdoor furniture	1
9	Chests of drawers & drawer units	1

```
# кількість пустих комірок
print(exer6['COUNT_empty_value'].sum())
```

```
102
```

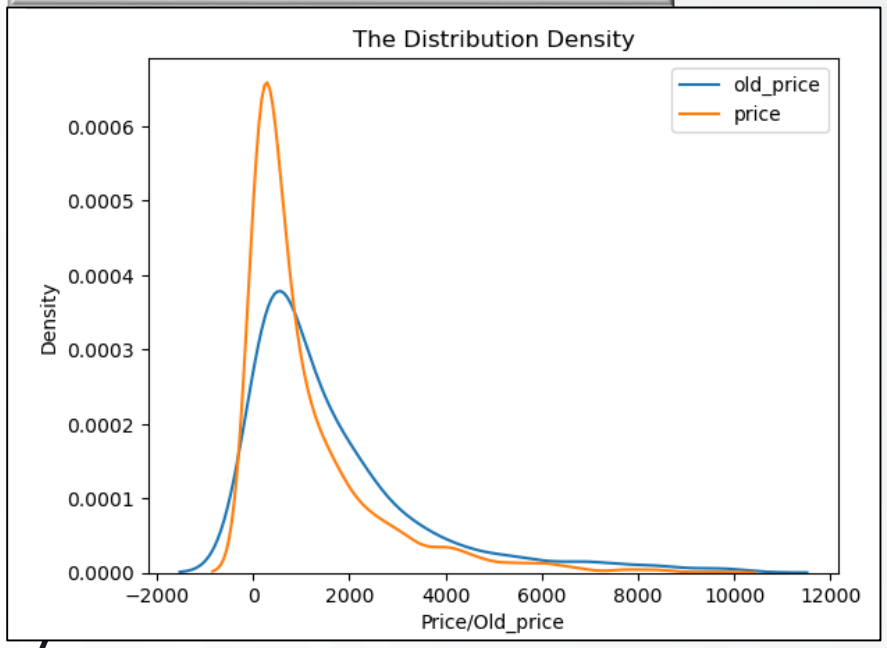
```
# запит вибору стовпчика max  кількість (designer який найчастіше зустрічається),
# також стовпчики category, designer
cursor.execute("""SELECT category, designer, max
                    FROM designer_for_empty""")
exer6_2= cursor.fetchall()
exer6_2=pd.DataFrame(exer6_2,columns=['category','designer','max'])
```

```
# графік розподілу кількості дизайнерів за категоріями
plt.subplots(figsize=(4,3))
sns.scatterplot(x='category',y ='designer',data=exer6_2, s=125)
plt.xticks(rotation=90)
plt.show()
```



ПРИПУЩЕННЯ ПРО ЛІНІЙНУ ЗАЛЕЖНІСТЬ ЗМІННИХ ‘PRICE’ ТА ‘OLD PRICE’

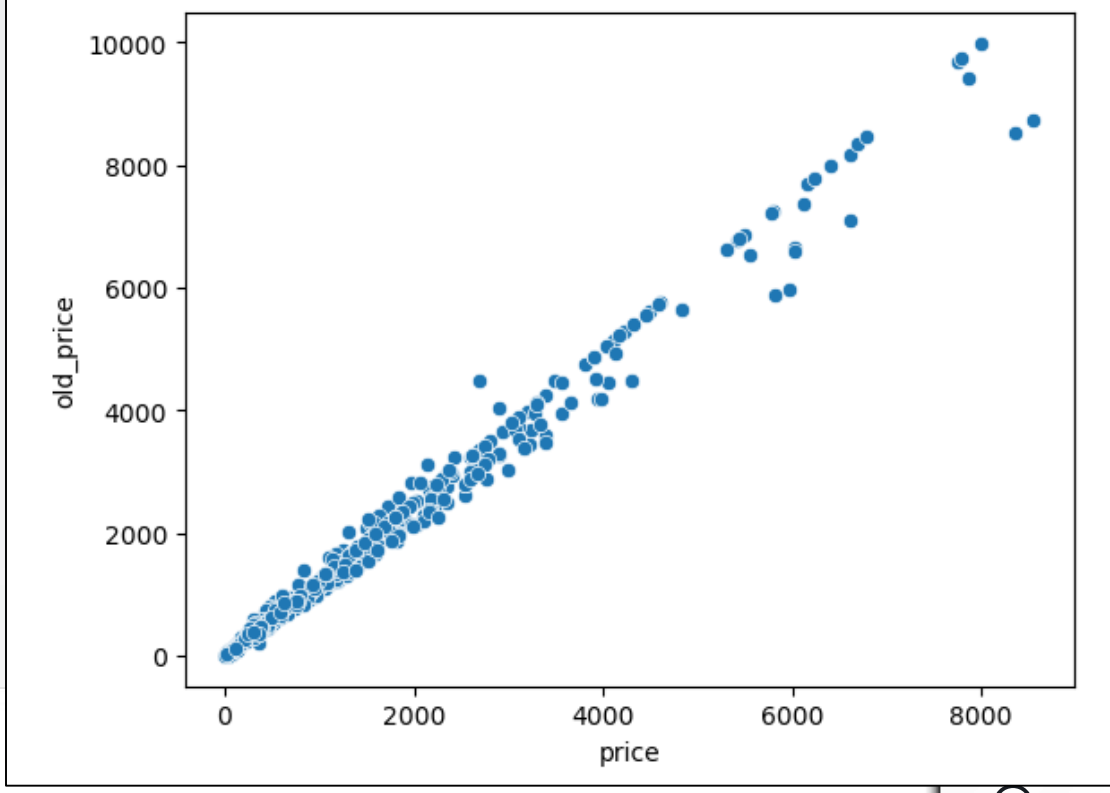
```python
import matplotlib.pyplot as plt
import seaborn as sns

fig, ax = plt.subplots()
sns.kdeplot(exer2_1['old_price'], label='old_price')
sns.kdeplot(exer2['price'], label='price')
ax.set_title('The Distribution Density')
ax.legend()
ax.set_xlabel('Price/Old_price')
plt.show()
```



Нульова гіпотеза: між змінними ‘price’ та ‘old_price’ немає лінійної залежності

Альтернативна гіпотеза: між змінними ‘price’ та ‘old_price’ є лінійної залежність

```python
#побудова scatterplot price за old_price
sns.scatterplot(x='price',y ='old_price', data=exer5_1)
plt.show()
```




```python
import numpy as np
from scipy.stats import pearsonr

 # Розрахунок коефіцієнта кореляції Пірсона та p-значення (для перевірки статистичної значущості)
correlation_coefficient, p_value = pearsonr(exer5_1['old_price'], exer5_1['price'])

print("Коефіцієнт кореляції:", correlation_coefficient)
print("p-значення:", p_value)

 # Розрахунок коефіцієнта кореляції Пірсона та p-значення (для перевірки статистичної значущості)
if p_value < 0.05:  #підходящий рівень значущості
    print("Залежність статистично значуща.")
else:
    print("Залежність не є статистично значущою.")
```

```
Коефіцієнт кореляції: 0.9937052030377374
p-значення: 0.0
Залежність статистично значуща.
```

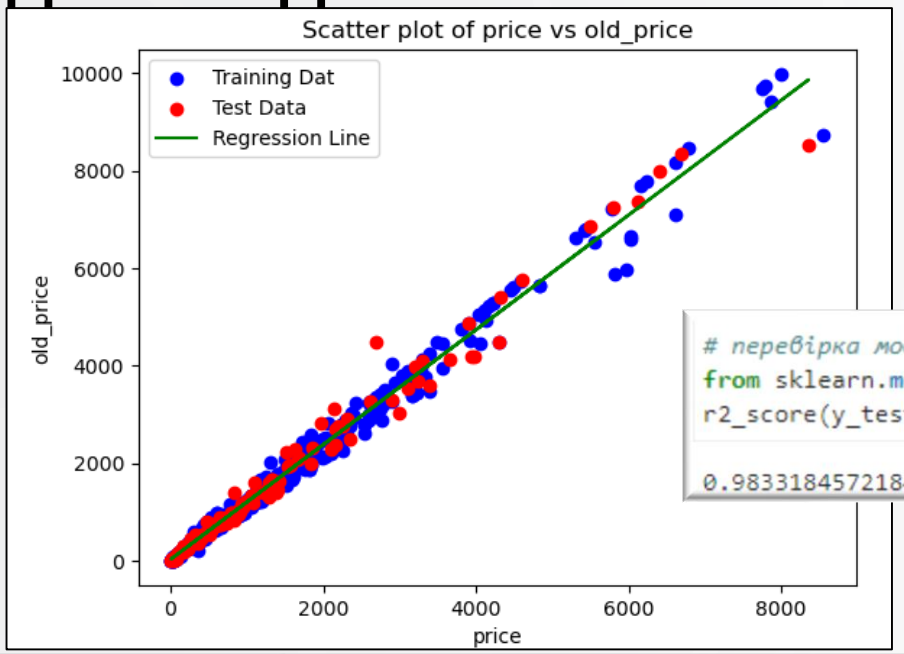
ВІДНОВЛЕННЯ ВІДСУТНІХ ДАНИХ

Missing old price

```
#запам'ятовування окремо даних з 'old_price',
які не дорівнюють і які дорівнюють 0
X=exer5_1['price'].values.reshape(-1, 1)
y=exer5_1['old_price']

#розподілення даних на тренувальні та тестові
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,random_state=42)
```

```
# побудова моделі лінійної регресії
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

```
# графічне зображення моделі
import matplotlib.pyplot as plt

plt.scatter(X_train, y_train, color='blue', label='Training Dat')
plt.scatter(X_test, y_test, color='red', label='Test Data')

# Line of best fit for the test data
plt.plot(X_test, y_pred, color='green', label='Regression Line')

# Add axis labels and title
plt.xlabel('price')
plt.ylabel('old_price')
plt.title('Scatter plot of price vs old_price')

# Add legend
plt.legend()

# Show the plot
plt.show()
```




```
# перевірка моделі
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
0.98331845721843
```

```
plot_y=model.predict(plot_x)
plot_y
```

```
array([350.04409999, 142.74841553, 114.4808222 , ..., 743.43477392,
       479.60390278, 338.26593611])
```

```
# запишимо передбачені значення до датафрейму exer5
exer5.loc[exer5['old_price'] == 0.0, 'old_price'] = plot_y.reshape(-1, 1)
```

```
# запишемо передбачені значення стовпчика old_price до таблиці NEW_data
conn = sqlite3.connect('sql_step_project.db')
conn.execute('BEGIN TRANSACTION')
for index, row in exer5.iterrows():
    item_id=row['item_id']
    category = row['category']
    old_price = row['old_price']
    conn.execute('UPDATE NEW_data SET old_price_predict = ? WHERE item_id = ? AND category = ?',
                 (old_price, item_id, category))
conn.commit()
```

МОДЕЛЬ ПЕРЕДБАЧЕННЯ ЦІНИ НА ОСНОВІ DECISIONTREEREGRESSOR

```
# вибір даних для створення моделі
X=exer8[['category','old_price_predict', 'depth_new','height_new', 'width_new']]
y=exer8['price']

# розбиття даних на тренувальні та тестувальні
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,random_state=42)
```

```
# створення моделі передбачення на основі DecisionTreeRegressor
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeRegressor
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
numeric_transformer = Pipeline(steps=[
        ('scaler', StandardScaler())])
categorical_transformer = Pipeline(steps=[
       ('onehot', OneHotEncoder())])
column_preprocessing = ColumnTransformer(transformers=[
    ('numeric', numeric_transformer,['old_price_predict', 'depth_new','height_new', 'width_new']),
    ('categorical', categorical_transformer,['category'])])
clf = Pipeline(steps=[
    ('preprocessing', column_preprocessing),
    ('clf', DecisionTreeRegressor())])
```

```
# створення моделі передбачення на основі DecisionTreeRegressor з найкращою стратегією, max_depth=7
numeric_transformer = Pipeline(steps=[
        ('scaler', StandardScaler())])
categorical_transformer = Pipeline(steps=[
       ('onehot', OneHotEncoder())])
column_preprocessing = ColumnTransformer(transformers=[
    ('numeric', numeric_transformer,['old_price_predict', 'depth_new','height_new', 'width_new']),
    ('categorical', categorical_transformer,['category'])])
clf = Pipeline(steps=[
    ('preprocessing', column_preprocessing),
    ('clf', DecisionTreeRegressor(max_depth=7))])
```

```
# перехресна перевірка
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, X, y, cv=5)
print(scores)
print(scores.mean())
clf.fit(X_train,y_train)
mean_squared_error(y_test,clf.predict(X_test))
```

```
[0.99622499 0.99616386 0.9959229  0.99364417 0.99370714]
0.995132608488553
6986.589563521508
```

```
# перевірка та підбір найкращої стратегії (max_depth)
from sklearn.model_selection import GridSearchCV
gridsearch=GridSearchCV(estimator=clf,
                        param_grid = {'clf__max_depth': [None,1,2,3,4,5,6,7,8,9,10,20,30,40,50,100]
})
gridsearch.fit(X_train, y_train)
print(gridsearch.best_params_)
print(gridsearch.best_score_)
clf.fit(X_train, y_train)
mean_squared_error(y_test,clf.predict(X_test))
```

```
{'clf__max_depth': 7}
0.9940961746981408
8292.885735492579
```

МОДЕЛЬ ПЕРЕДБАЧЕННЯ ЦІНИ НА ОСНОВІ KNEIGHBORSREGRESSOR

```python
# створення моделі передбачення на основі KNeighborsRegressor
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
numeric_transformer = Pipeline(steps=[
        ('scaler', StandardScaler())])
categorical_transformer = Pipeline(steps=[
       ('onehot', OneHotEncoder())])
column_preprocessing = ColumnTransformer(transformers=[
    ('numeric', numeric_transformer,['old_price_predict', 'depth_new','height_new', 'width_new']),
    ('categorical', categorical_transformer,['category'])])
model = Pipeline(steps=[
    ('preprocessing', column_preprocessing),
    ('reg', KNeighborsRegressor())])
```

```python
# створення моделі передбачення на основі KNeighborsRegressor з найкращою стратегією, n_neighbors=2
numeric_transformer = Pipeline(steps=[
        ('scaler', StandardScaler())])
categorical_transformer = Pipeline(steps=[
       ('onehot', OneHotEncoder())])
column_preprocessing = ColumnTransformer(transformers=[
    ('numeric', numeric_transformer,['old_price_predict', 'depth_new','height_new', 'width_new']),
    ('categorical', categorical_transformer,['category'])])
model = Pipeline(steps=[
    ('preprocessing', column_preprocessing),
    ('reg', KNeighborsRegressor(n_neighbors=2))])
```

```python
# перевірка та підбір найкращої стратегії
# Find the best value of "k"
from sklearn.metrics import accuracy_score
for k in range(1, 21):
    numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])
    categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder())])
    column_preprocessing = ColumnTransformer(transformers=[
    ('numeric', numeric_transformer,['old_price_predict', 'depth_new','height_new', 'width_new']),
    ('categorical', categorical_transformer,['category'])])
    model = Pipeline(steps=[
    ('preprocessing', column_preprocessing),
    ('reg', KNeighborsRegressor(n_neighbors=k))])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    msr = mean_squared_error(y_test,model.predict(X_test))
    print("k:", k, "mean_squared_error:", msr)
```

```
k: 1 mean_squared_error: 48750.41018893388
k: 2 mean_squared_error: 44387.504298245614
k: 3 mean_squared_error: 47037.34362273205
k: 4 mean_squared_error: 50628.56895748987
k: 5 mean_squared_error: 51702.30704831309
k: 6 mean_squared_error: 56414.93213337831
k: 7 mean_squared_error: 60971.98071910546
k: 8 mean_squared_error: 64781.97206224696
k: 9 mean_squared_error: 67683.93469652288
k: 10 mean_squared_error: 68919.45061538462
```

```python
# перехресна перевірка
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5)
print(scores)
print(scores.mean())
model.fit(X_train,y_train)
mean_squared_error(y_test,model.predict(X_test))
```

```
[0.963896   0.97760489 0.97834837 0.97331722 0.97305159]
0.9732436125082495

44387.504298245614
```

ВИСНОВКИ

Приклад передбачення

```
# дані для передбачення
new_data = pd.DataFrame({
    'category': ['Chairs'],
    'old_price_predict': [320],
    'depth_new': [50],
    'height_new': [80],
    'width_new': [40]
})
# передбачення за допомого моделі DecisionTreeRegressor
Price_new=clf.predict(new_data)
Price_new.round(1)
```

```
array([250.4])
```

```
# передбачення за допомого моделі KNeighborsRegressor
Price_new_1=model.predict(new_data)
Price_new_1.round(1)
```

```
array([198.5])
```

DecisionTreeRegressor модель є точнішою та має меншу середньоквадратичну помилку передбачення ціни товару, ніж KNeighborsRegressor модель.

Щоб покращити якість передбачення моделі варто б було навчити її враховувати такі факторні змінні як **‘designer’, ‘name’.**

В самому наборі вхідних даних також хотілось би мати деякі додаткові характеристики товарів, які впливають на його ціну, наприклад собівартість товарів, вартість матеріалів, роботи, тощо.


+ableau

ВХІДНІ ДАНІ

Tableau Public - Safonova_step_project_final1

File Data Window Help

Output

Connections Add

Output
Text file

Files

Use Data Interpreter

Data Interpreter might be able to clean your Text file workbook.

2018.csv

Output.csv

New Union

New Table Extension

Output.csv

Drag ta

Output.csv 17 fields 156 rows

Table Details

Output.csv Overall rank	Output.csv Country or region	Output.csv Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life ch...
1	Finland	7,63200	1,30500	1,592000	0,874000	0,68
2	Norway	7,59400	1,45600	1,582000	0,861000	0,69
3	Denmark	7,55500	1,35100	1,590000	0,868000	0,68

Tableau Public - Safonova_step_project_final1

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Data Analytics

Output

Search

Folders

- Calculated_
- Country/continent
  - Continent_only
  - Continent/Country
    - Continent
    - Country
  - Country or region
- Measure
  - Freedom to make lif...
  - GDP per capita
  - Generosity
  - Healthy life expecta...
  - Overall rank
  - Perceptions of corru...
  - Score
  - Social support

Abc *Measure Names*

Pages

Columns Continent_onl

Rows AVG(Changec

Create Group [Continent]

Field Name: Continent (group)

Groups: Add to:

Africa
Asia
Australia & New Zealand
Central America
Europe
North America
South America

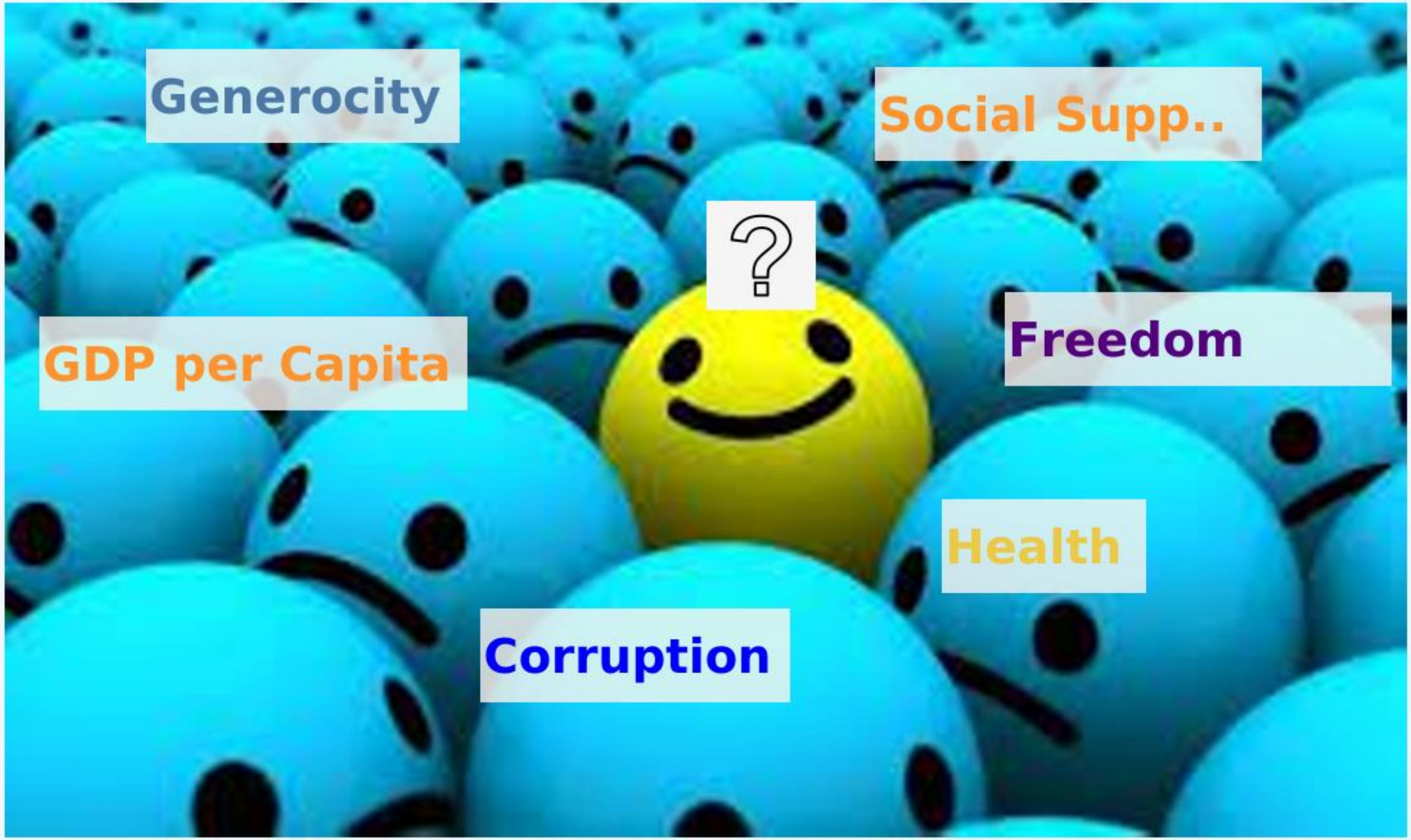
Group Rename Ungroup Show Add Location

Include 'Other' Find >>

Reset OK Cancel Apply

Qualities of countries happy life

Top N
Fall
Line
Map



Qualities of countries happy life

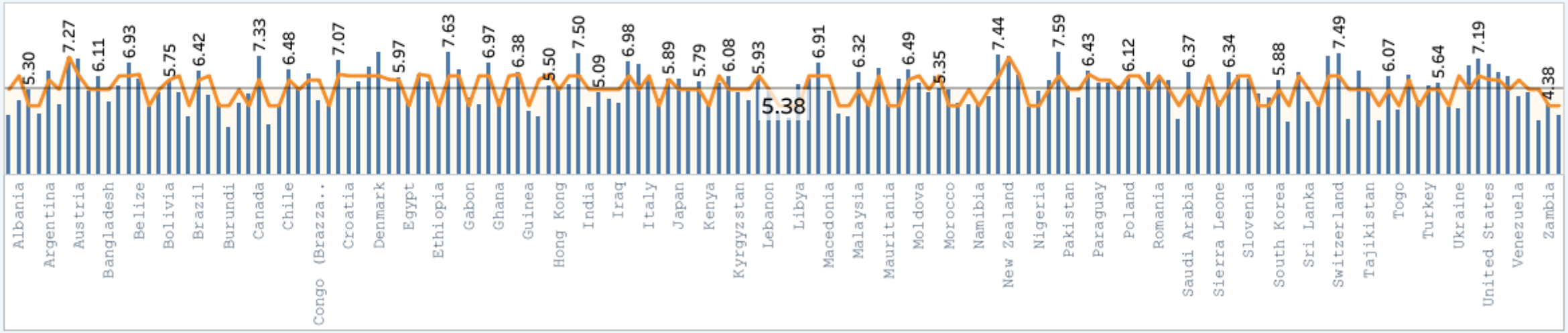
Top N

Fall

Line

Map

Score by country & continent



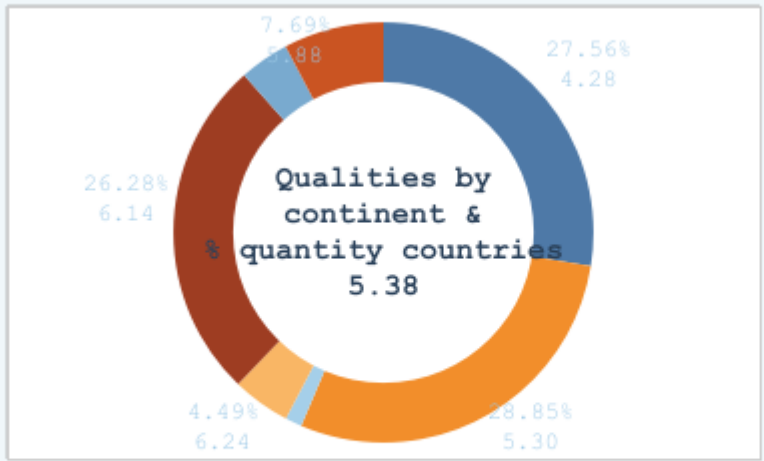
Country: All

Continent: All

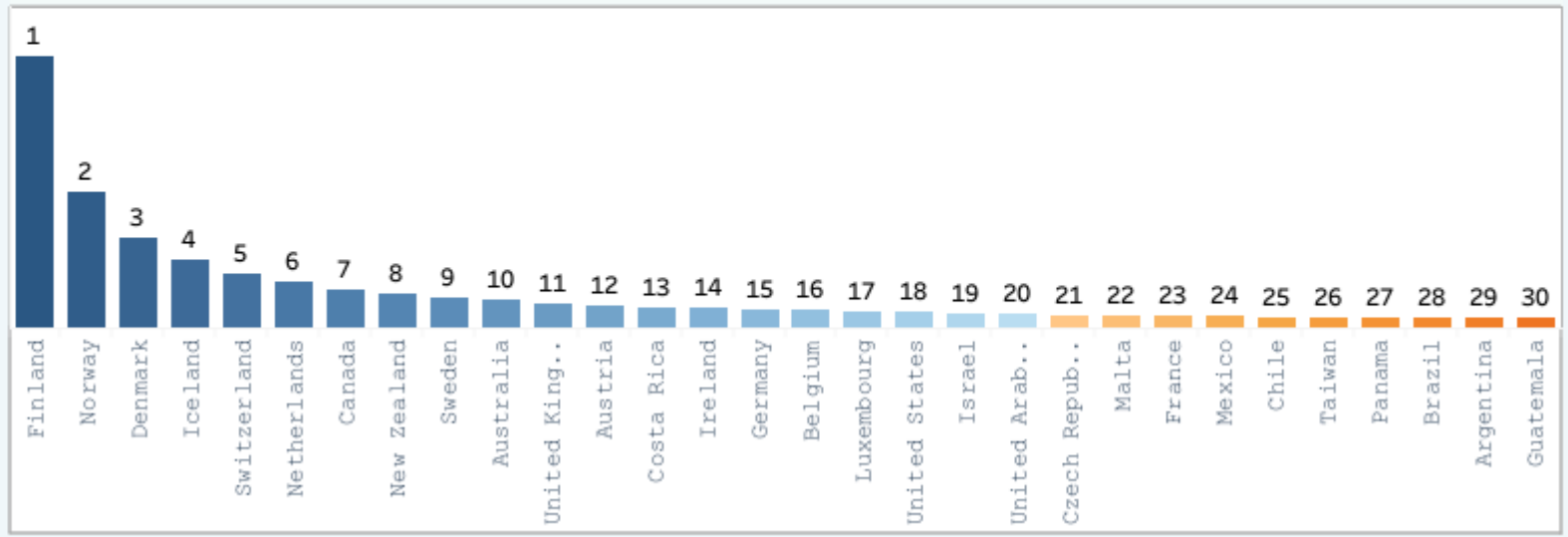
Score

Score by continent

Parameter_top_N: 30



30 top countries



Qualities of countries happy life

Top N

Fall

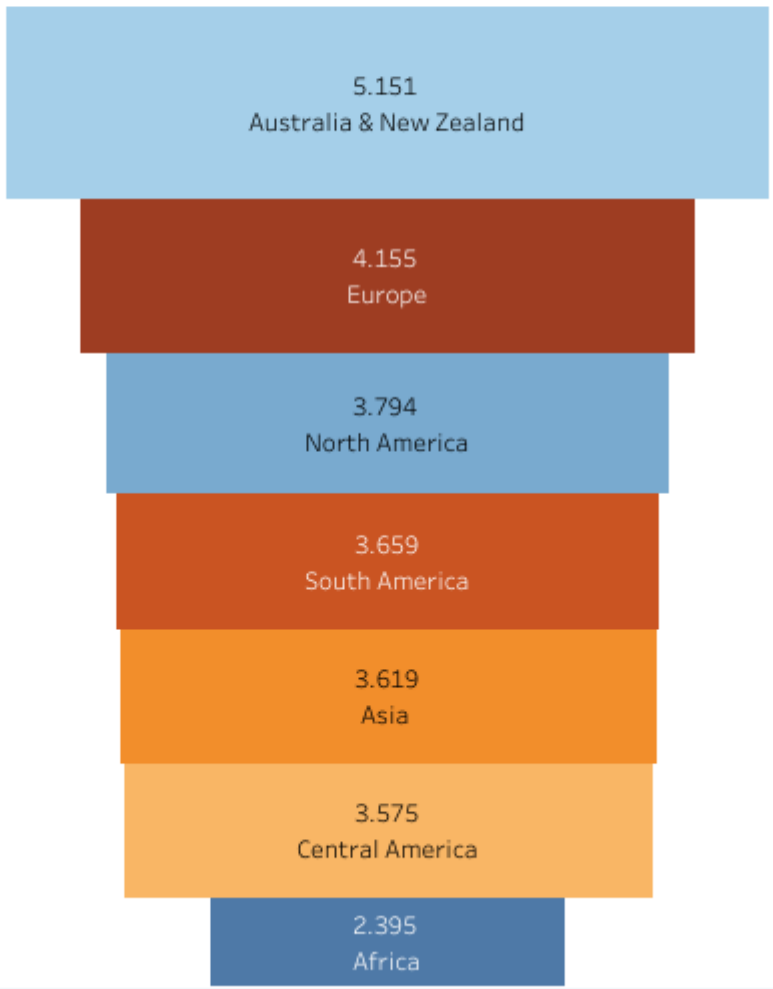
Line

Map

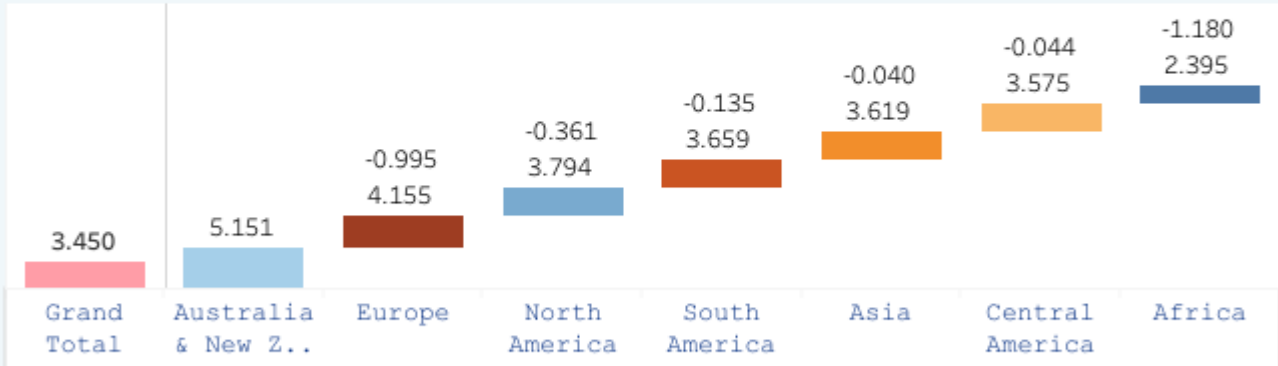
Choose parameter
all parameters

Continent
All

all parameters by continents



Difference all parameters by Continents



Difference Qualities by Continents

	% Difference Freedom from the Previous	% Difference GDP from the Previous	% Difference Generosity from the Previous	% Difference Healthy from the Previous	% Difference Corruption from the Previous	% Difference Social support from the..	% Difference Quality of life from the Prev..
Australia & New Zealand							
Europe	-28.7%	-9.6%	-52.8%	-9.8%	-62.1%	-11.8%	-19.3%
North America	1.6%	-16.8%	25.5%	-15.7%	-3.4%	-5.9%	-8.7%
South America	6.8%	-1.8%	-44.7%	-5.1%	-45.6%	2.9%	-3.6%
Asia	-8.1%	1.0%	80.8%	0.0%	70.6%	-11.3%	-1.1%
Central America	18.3%	-14.8%	-26.8%	2.4%	-26.1%	7.2%	-1.2%
Africa	-30.7%	-40.8%	3.8%	-54.3%	6.9%	-25.3%	-33.0%

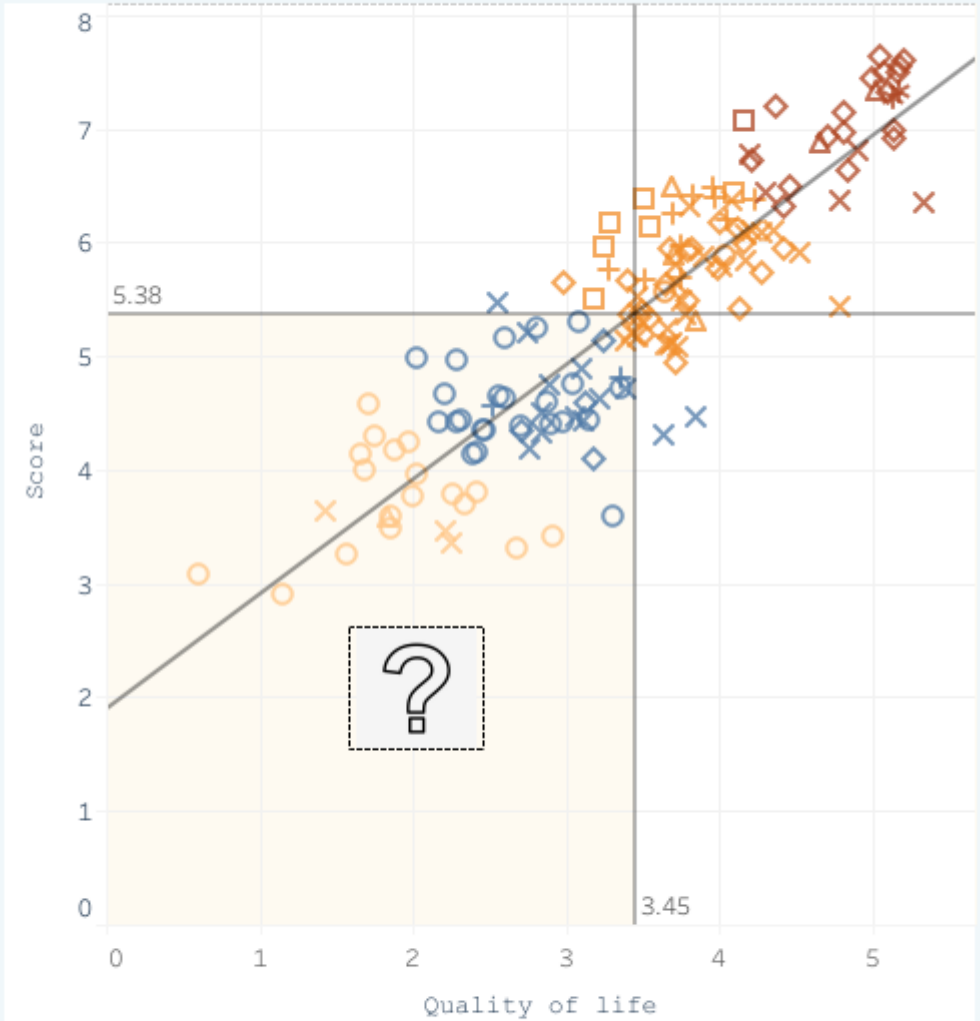
Qualities of countries happy life

Top N

Fall

Line

Map



Continent

- Africa
- Asia
- Australia &..
- Central Ame..
- Europe
- North America
- South America

Continent

All

Country

All

Qualities by countries

	Freedom to make life ch..	GDP per capita	Generosity	Healthy life expectancy	Perceptions of corruption	Social support	Quality of life
Europe	0.47	1.18	0.17	0.81	0.13	1.40	4.16
Australia & New Zealand	0.66	1.30	0.36	0.89	0.35	1.59	5.15
North America	0.48	0.98	0.21	0.68	0.13	1.32	3.79
Asia	0.47	0.97	0.21	0.64	0.12	1.20	3.62
South America	0.51	0.96	0.12	0.64	0.07	1.36	3.66
Central America	0.55	0.83	0.16	0.66	0.09	1.29	3.58
Africa	0.38	0.49	0.16	0.30	0.09	0.96	2.39
Grand Total	0.50	0.96	0.20	0.66	0.14	1.30	3.76

0.07 — 5.15

Assumtion about changing qualities

Top N

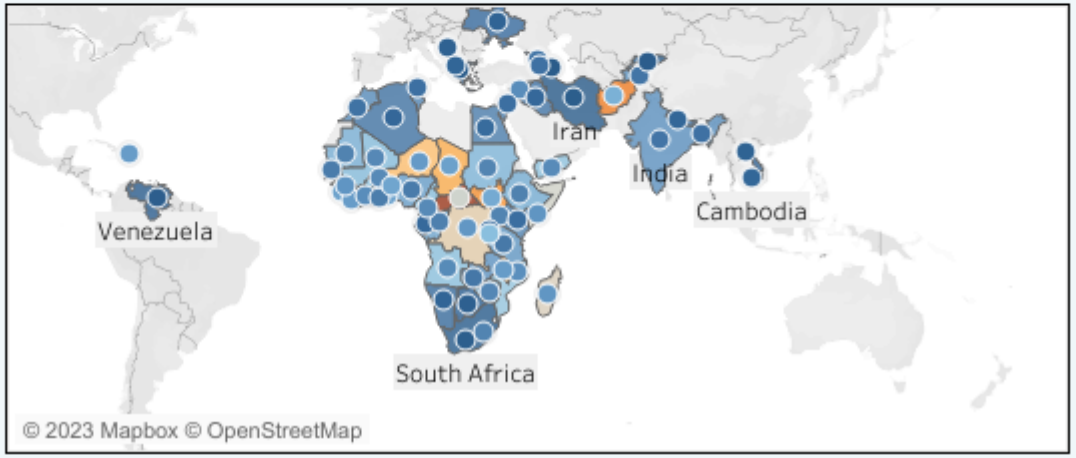
Fall

Line

Map

Africa
Namibia

AVG changed quality of life 3.154
AVG quality of life 3.154
Score 4.441



Country of less points
True

Choose parameter
all parameters

Freedom 0.29

Quality/ Changed quality of life
0.595 6.345

Generoc.. 0.75

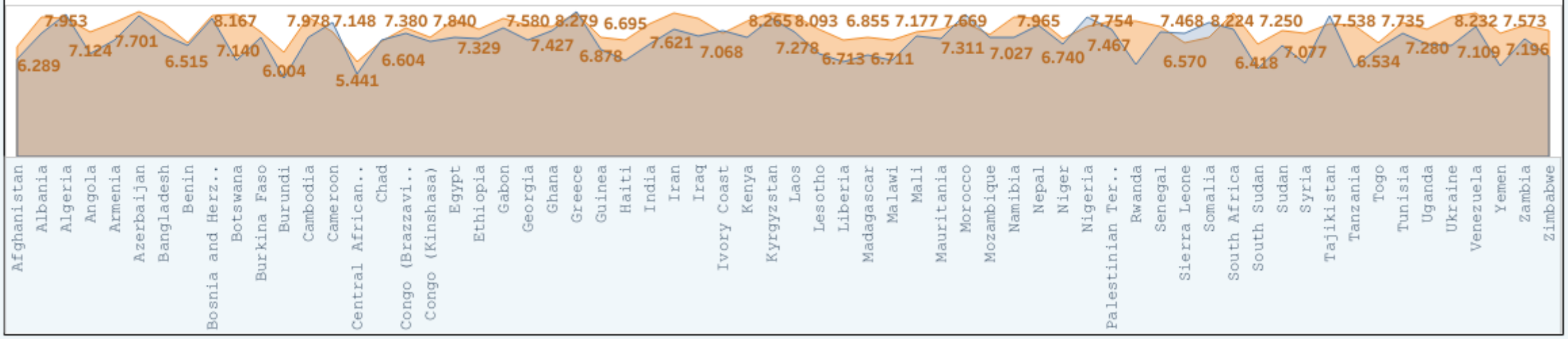
GDP per capita
0.55

Health 0.72

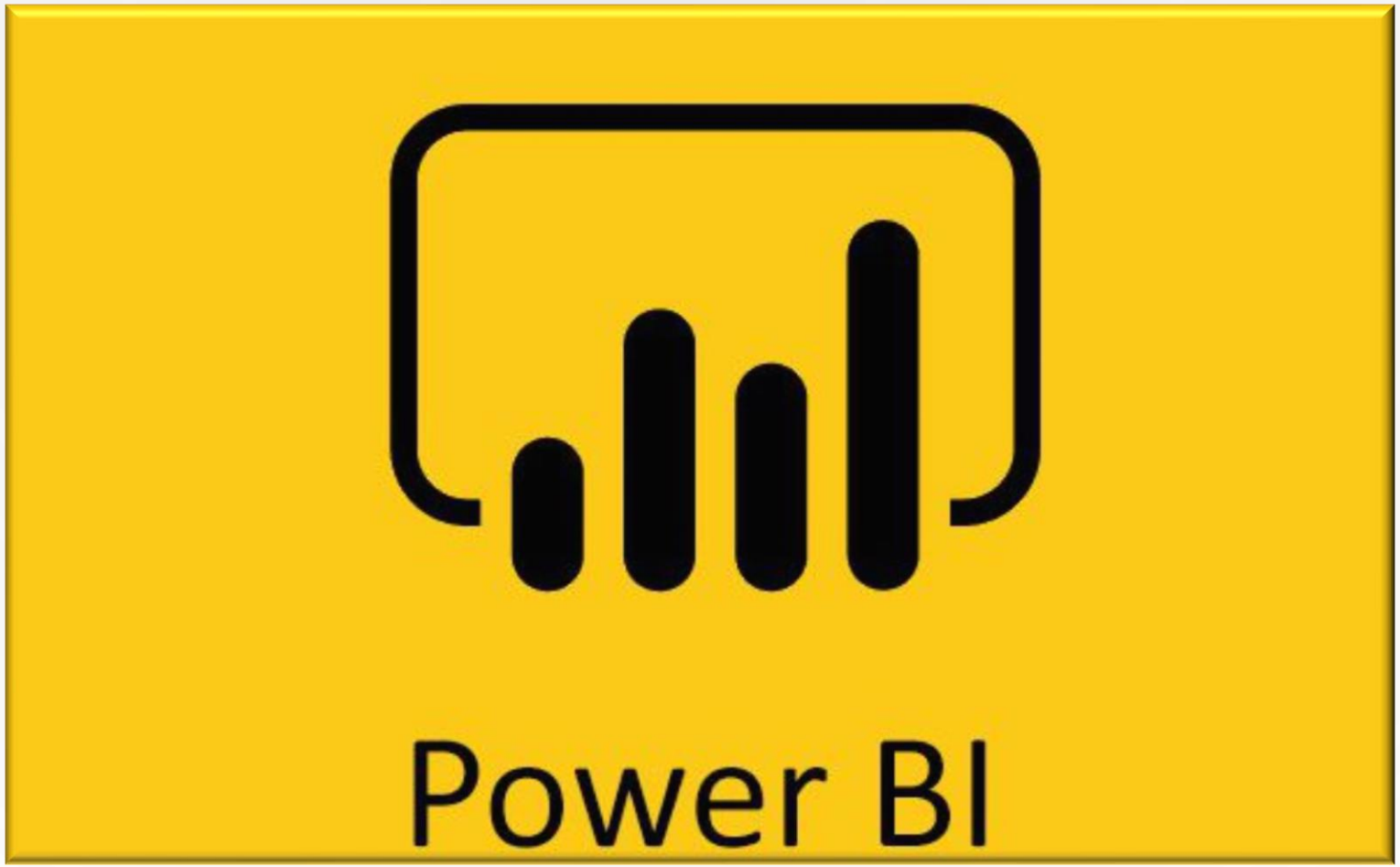
Support 0.7

Corrupt.. 0.09

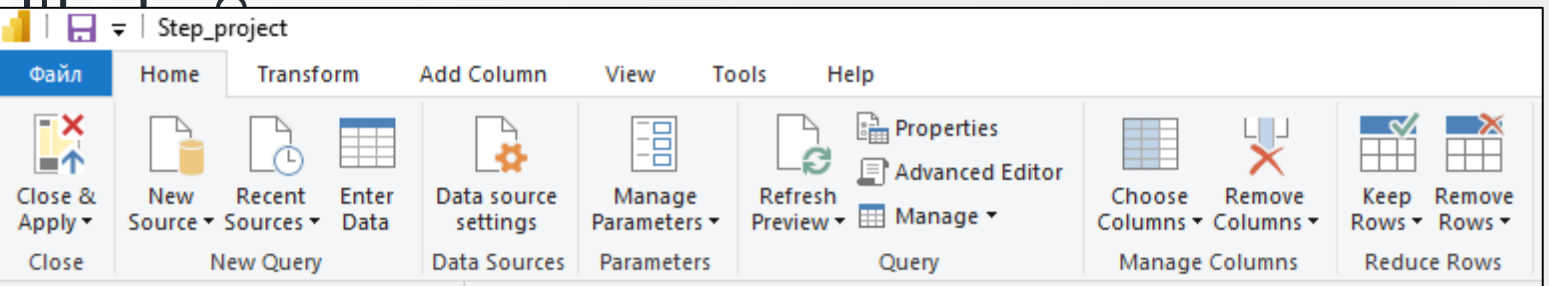
AVG Scores before/after changing

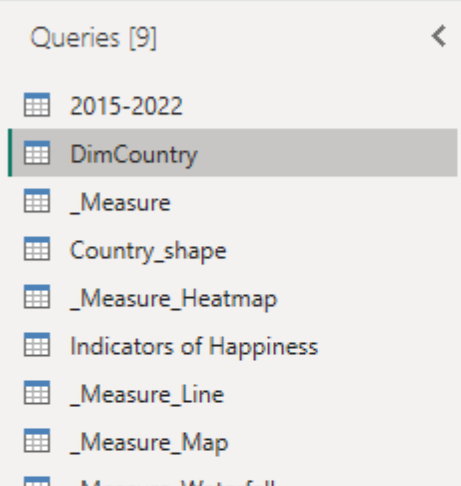


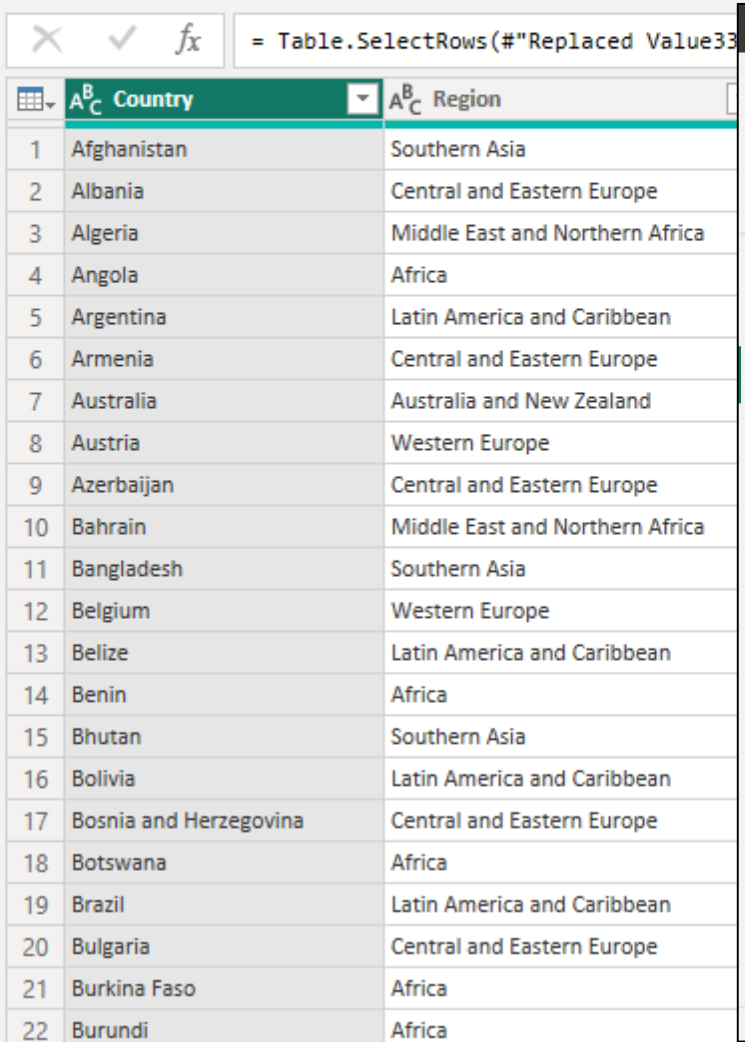
Score Score_calculate


Power BI

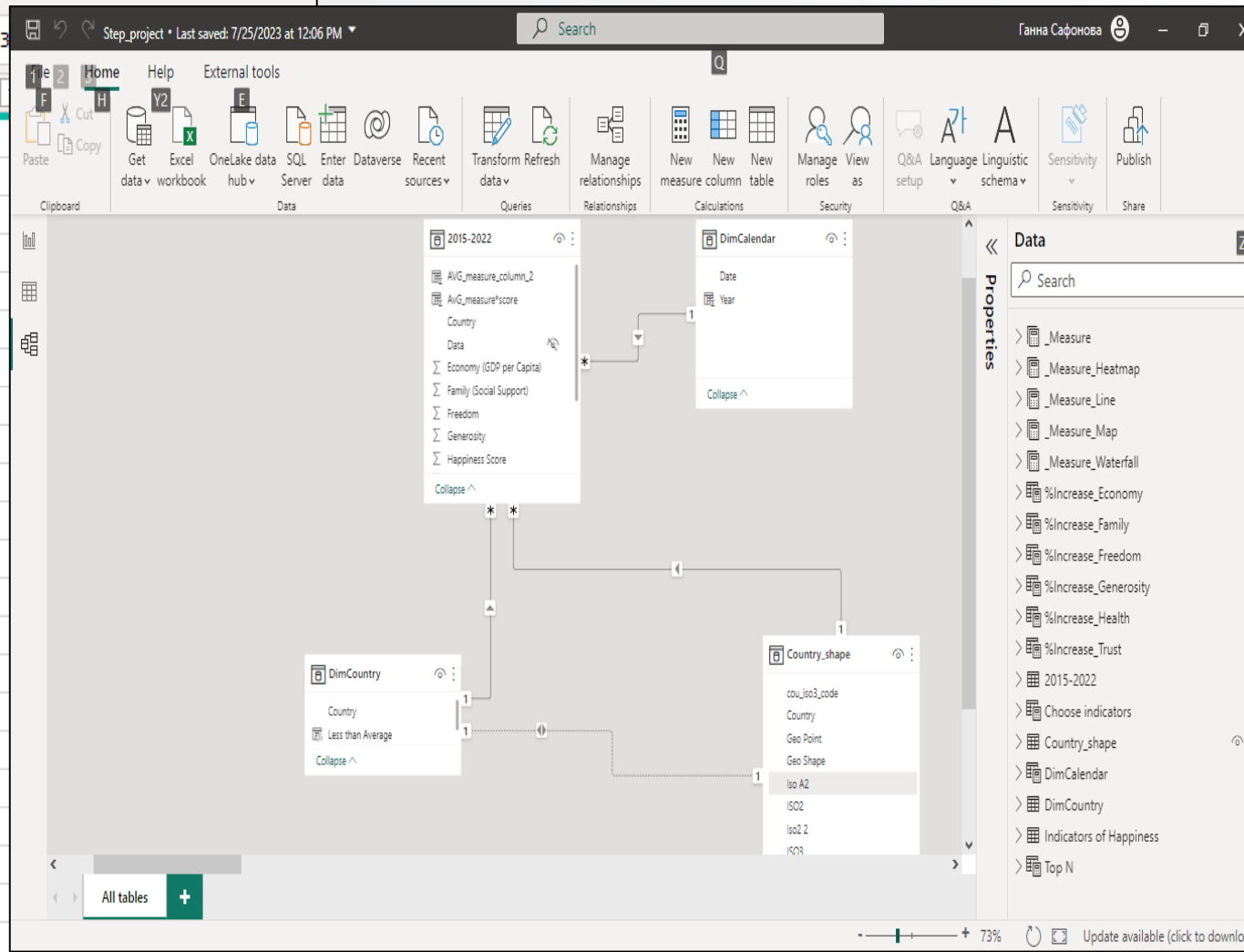
ВХІДНІ ДАНІ







	Country	Region
1	Afghanistan	Southern Asia
2	Albania	Central and Eastern Europe
3	Algeria	Middle East and Northern Africa
4	Angola	Africa
5	Argentina	Latin America and Caribbean
6	Armenia	Central and Eastern Europe
7	Australia	Australia and New Zealand
8	Austria	Western Europe
9	Azerbaijan	Central and Eastern Europe
10	Bahrain	Middle East and Northern Africa
11	Bangladesh	Southern Asia
12	Belgium	Western Europe
13	Belize	Latin America and Caribbean
14	Benin	Africa
15	Bhutan	Southern Asia
16	Bolivia	Latin America and Caribbean
17	Bosnia and Herzegovina	Central and Eastern Europe
18	Botswana	Africa
19	Brazil	Latin America and Caribbean
20	Bulgaria	Central and Eastern Europe
21	Burkina Faso	Africa
22	Burundi	Africa



TopN

Heatmap

Waterfall

Line

Map

Top N

Region: All

Top N Country: 6

Clear all slicers

2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022

TopN

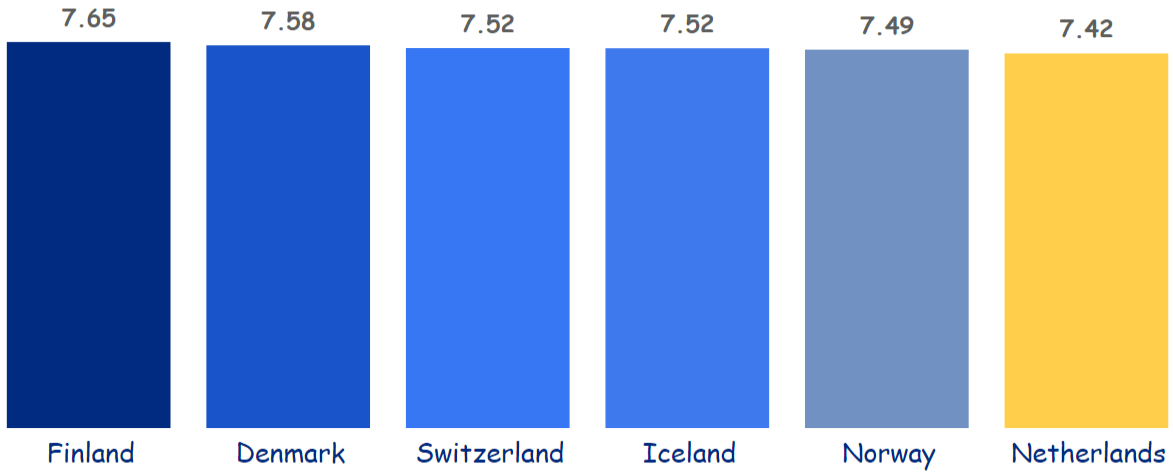
Heatmap

Waterfall

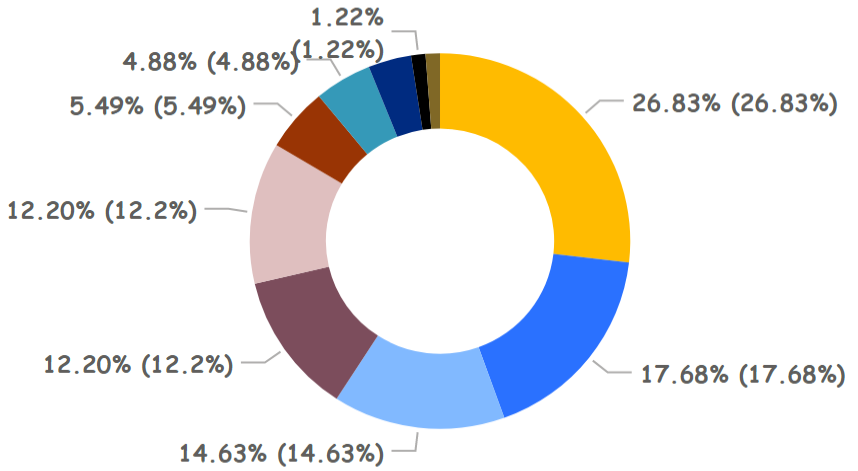
Line

Map

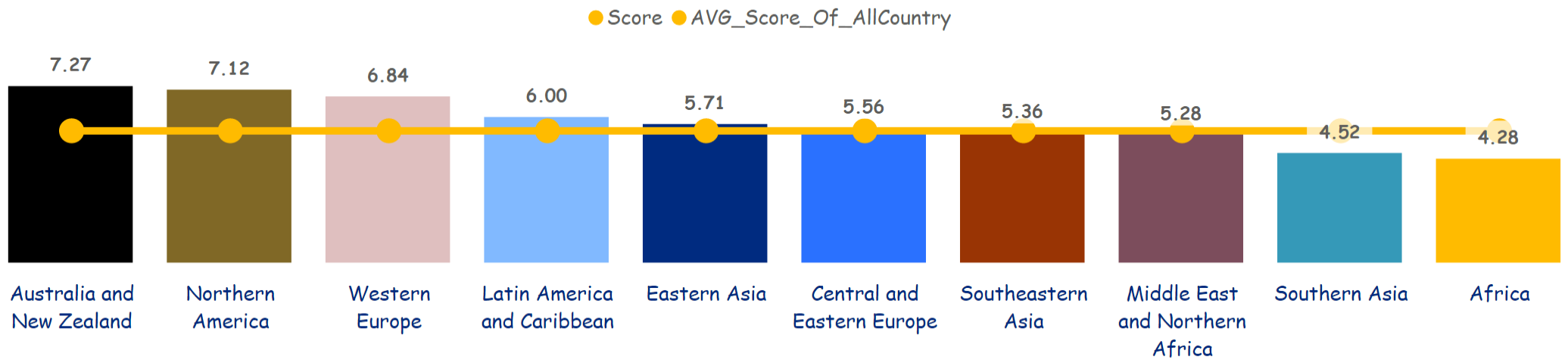
Top 6 AVG_Score by Countries



Quantity Countries by Region



AVG_Score by Regions comparatively to AVG_Score by All Counrties



Heatmap

Country: All

Clear all slicers

Economy | Family (Social Support) | Freedom | Generosity | Health | Trust to Government

TopN

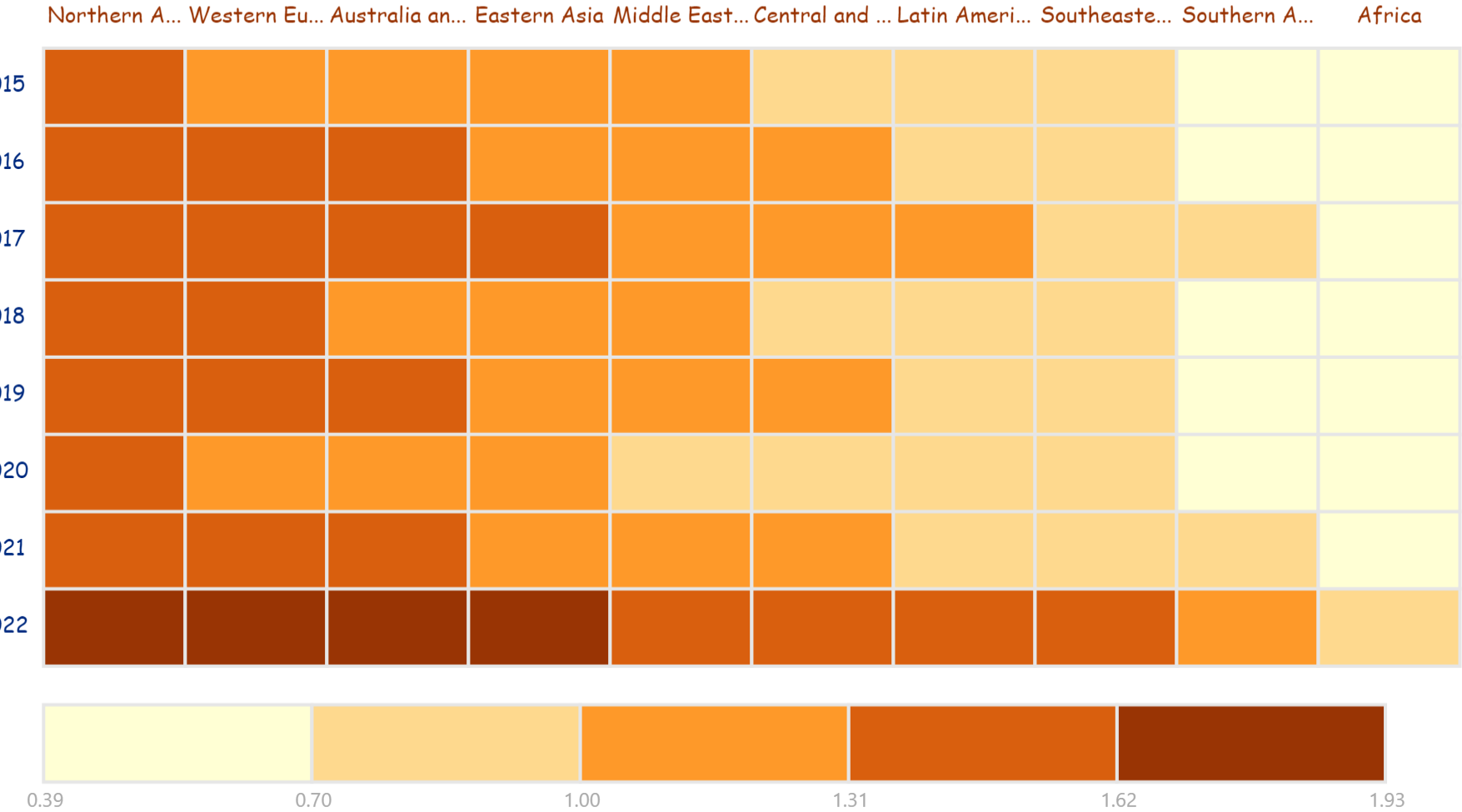
Heatmap

Waterfall

Line

Map

Economy by Regions and Years



More than one Country

Economy by Regions and Years

AVGValue_Indicator
0.97

MaxValue_Indicator
2.21

MinValue_Indicator
0.00

Waterfall

Clear all slicers

Economy | Family (Social Support) | Freedom | Generosity | Health | Trust to Government

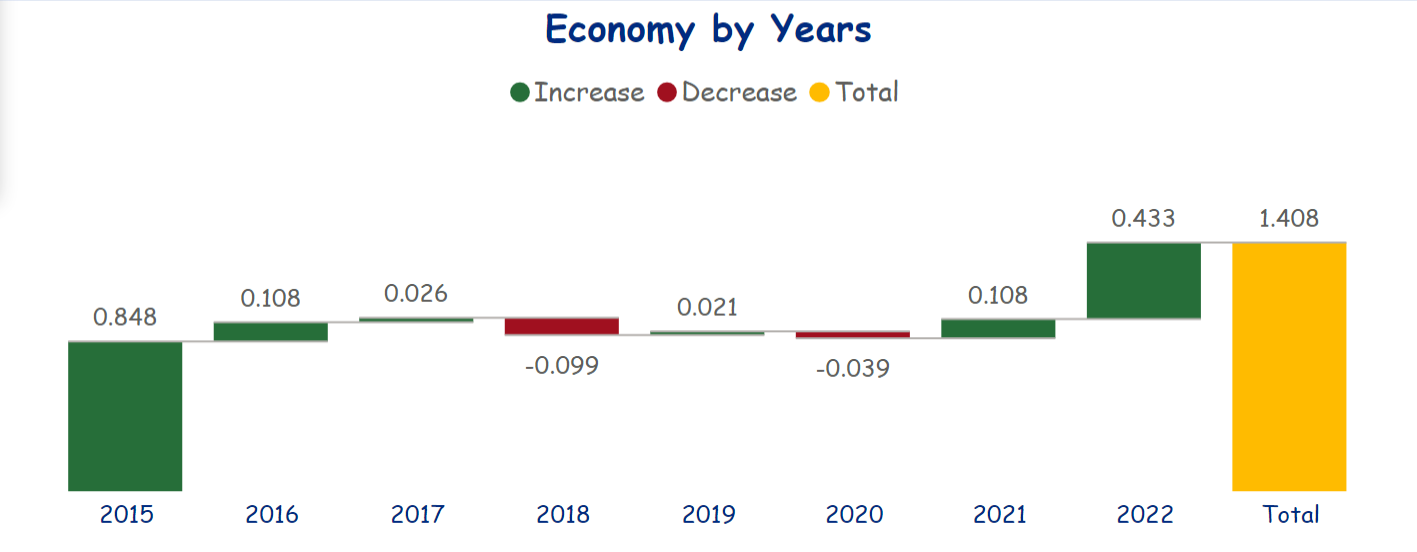
TopN

Heatmap

Waterfall

Line

Map



Region/Country

All

More than one Country

More than one Region

Year	Economy	%Diff EconomyTo LastYear	Family	%Diff FamilyTo LastYear	Freedom	%Diff FreedomTo LastYear	Generosity	%Diff GenerosityTo LastYear	Health	%Diff HealthTo LastYear	Trust	%Diff TrustTo LastYear
2015	0.848		0.991		0.428		0.235		0.630		0.142	
2016	0.956	12.7%	0.794	-19.9%	0.370	-13.4%	0.241	2.3%	0.557	-11.5%	0.136	-4.1%
2017	0.982	2.7%	1.189	49.8%	0.408	10.2%	0.247	2.5%	0.550	-1.4%	0.123	-9.6%
2018	0.884	-10.0%	1.216	2.3%	0.456	11.6%	0.181	-26.7%	0.597	8.6%	0.112	-8.9%
2019	0.905	2.4%	1.209	-0.6%	0.393	-13.8%	0.185	2.1%	0.725	21.5%	0.111	-1.2%
2020	0.867	-4.3%	1.155	-4.4%	0.463	18.1%	0.189	2.4%	0.691	-4.8%	0.130	17.8%
2021	0.974	12.5%	0.793	-31.3%	0.499	7.6%	0.178	-6.0%	0.518	-25.0%	0.135	3.4%
2022	1.408	44.5%	0.906	14.2%	0.517	3.7%	0.196	10.0%	0.585	12.8%	0.272	101.6%

Line

Region/Country: All

Clear all slicers

Choose indicators: All

Less than Average: Select all | False | True

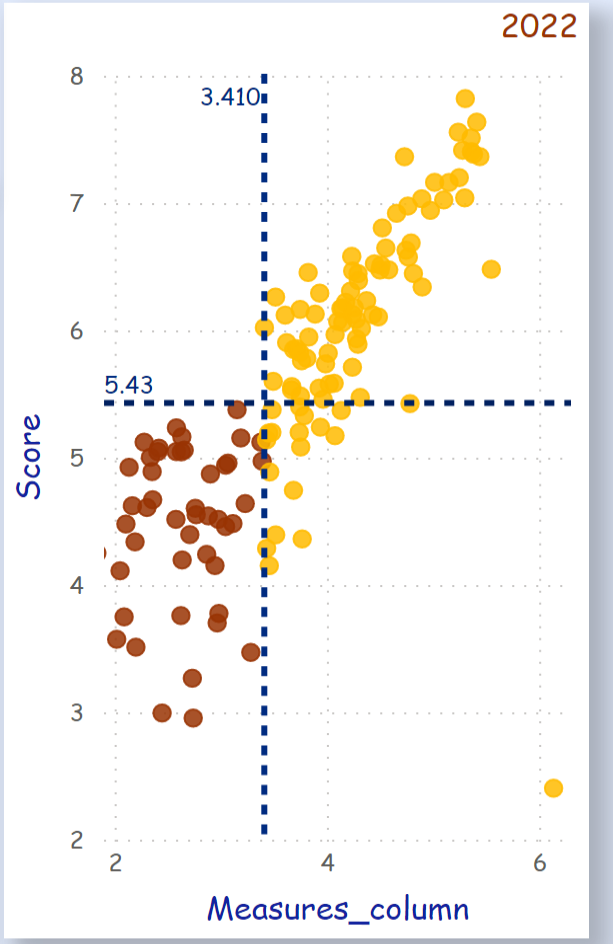
TopN

Heatmap

Waterfall

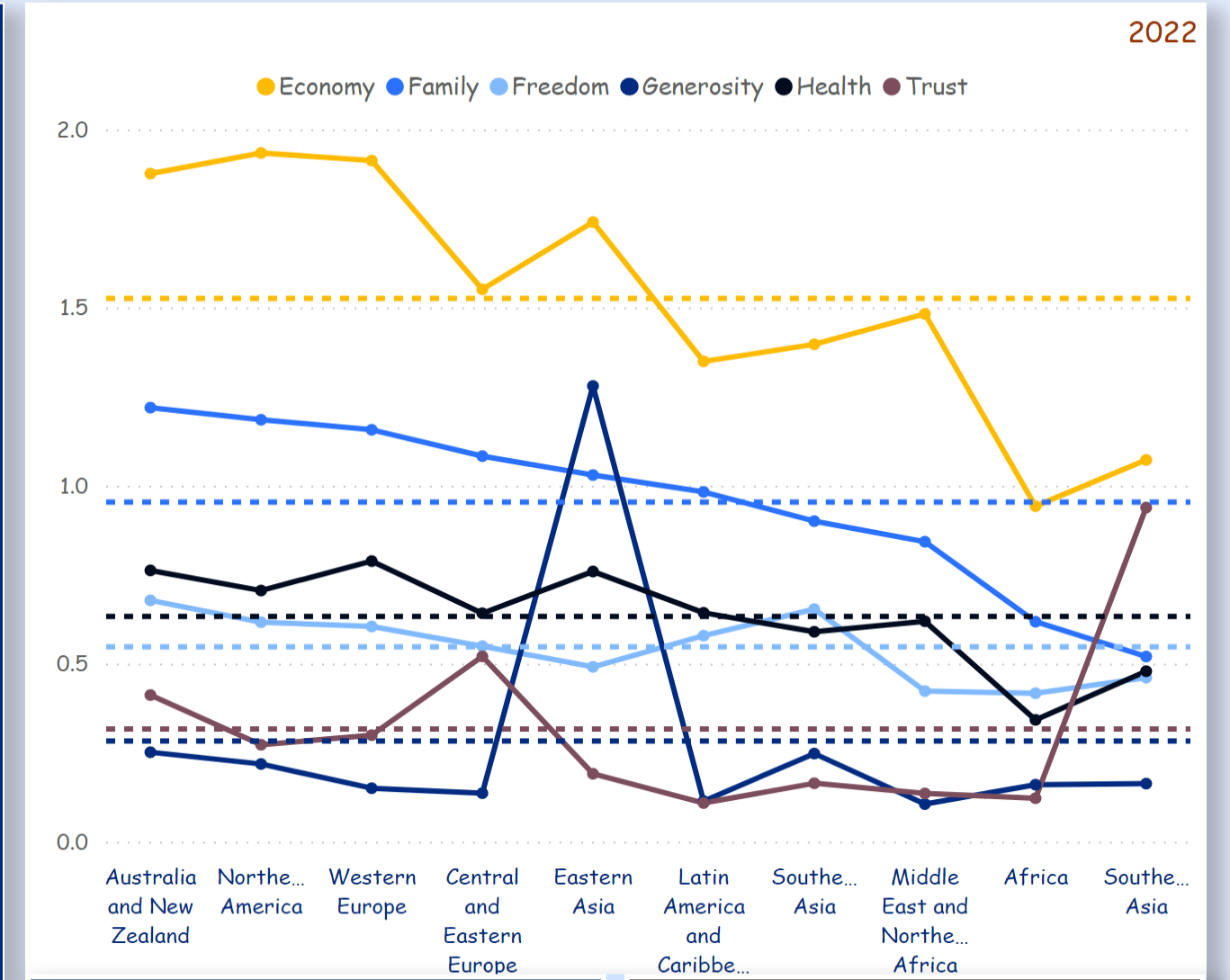
Line

Map



AVG_AllScore	AVG_AllMeasure
5.43	3.41

Economy2022	Freedom2022	Generosity2022	Health2022	Family2022	Trust2022
1.41	0.52	0.20	0.58	0.91	0.27



Graph | Matrix

Line

Region/Country: All

Clear all slicers

Choose indicators: All

Less than Average: Select all | False | True

TopN

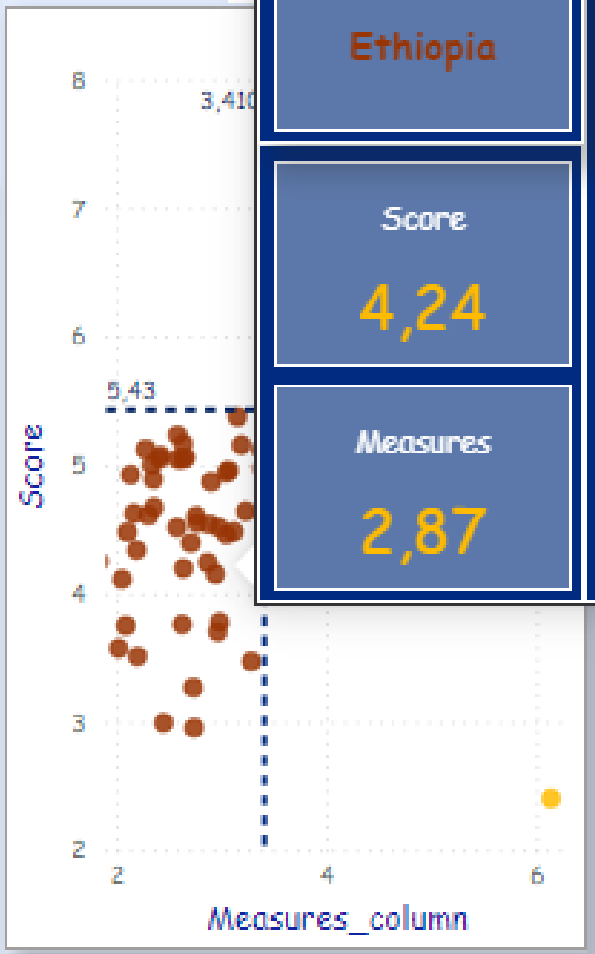
Heatmap

Waterfall

Line

Map

	Economy	Family	Freedom	Generosity	Health	Trust
ope	1,913	1,157	0,605	0,150	0,788	0,299
a	1,072	0,520	0,461	0,163	0,479	0,939
Asia	1,397	0,900	0,653	0,248	0,590	0,164
erica	1,934	1,185	0,617	0,219	0,706	0,273
nd	1,482	0,843	0,423	0,106	0,619	0,136
rica						
and	1,349	0,982	0,579	0,114	0,643	0,109
	1,740	1,030	0,491	1,280	0,759	0,191
astern	1,551	1,083	0,549	0,137	0,642	0,521
New	1,876	1,219	0,678	0,252	0,762	0,412
Africa	0,942	0,618	0,417	0,160	0,342	0,122



Ethiopia

Economy	Health
0,788	0,457
Score	**Freedom**	**Trust**
4,24	0,472	0,136
Measures	**Generosity**	**Family**
2,87	0,205	0,809

Health2022	0,58
Family2022	0,91
Trust2022	0,27

AVG_AllScore	AVG_AllMeasure
5,43	3,41

Graph | Matrix

Map

Region/Country

All

Clear all slicers

Less than Average

Select all | False | True

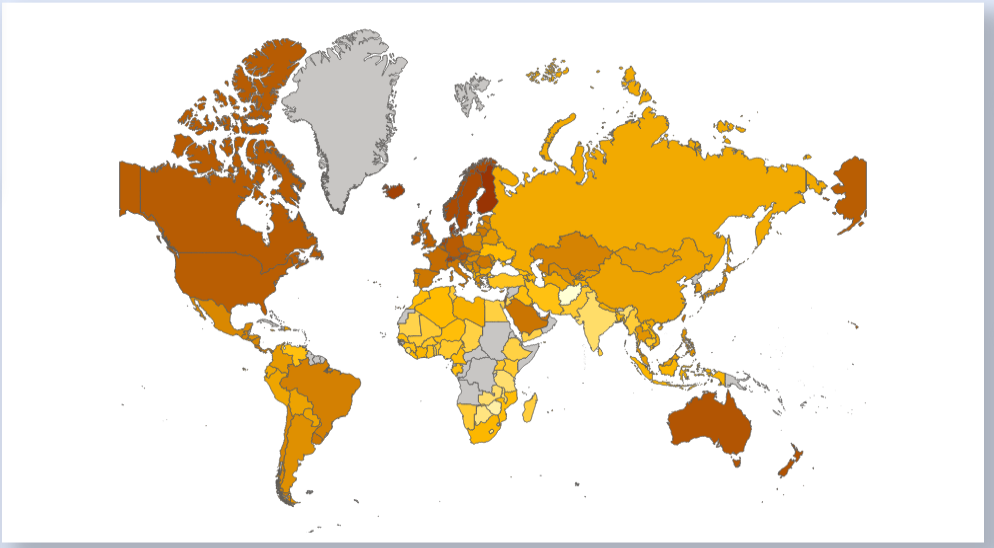
TopN

Heatmap

Waterfall

Line

Map

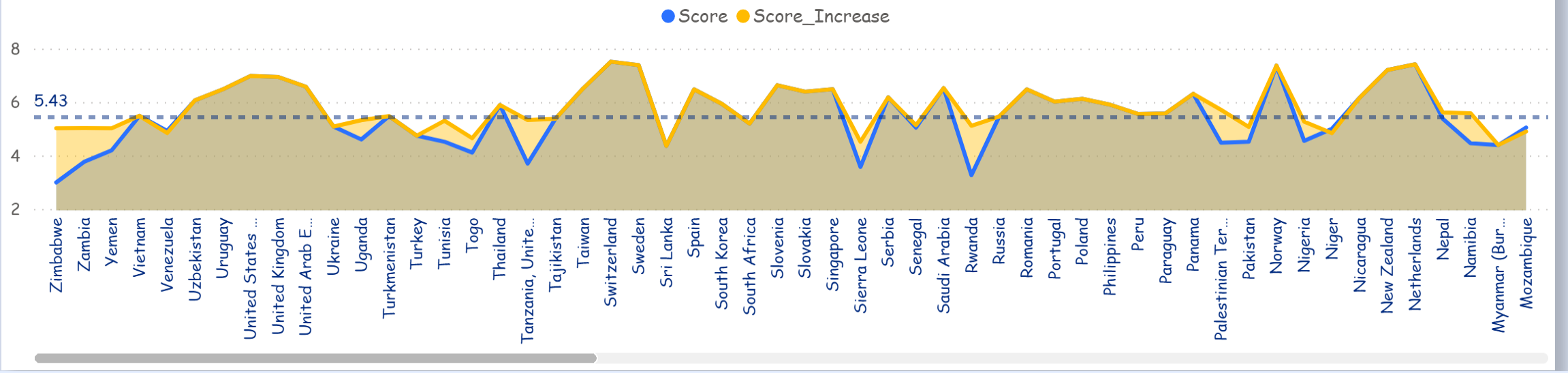


Choose new values for Indicators

%_Increase Economy	%_Increase Family	%_Increase Freedom
46	31	21

%_Increase Generosity	%_Increase Trust	%_Increase Health
62	47	33

Score and Score_Increase by Country



Map

Region/Country

All

Clear all slicers

Less than Average

Select all | False | True

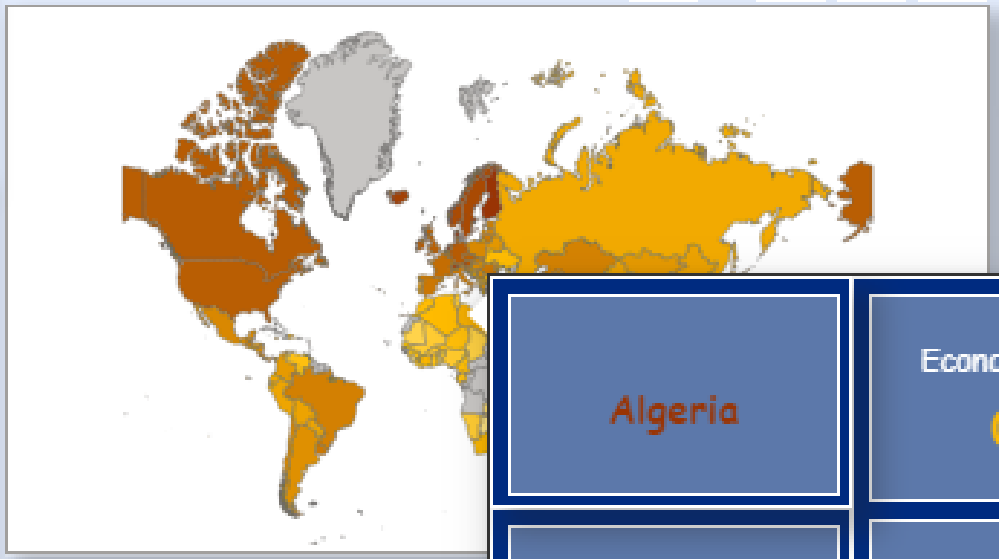
TopN

Heatmap

Waterfall

Line

Map



Choose new values for Indicators

%_Increase Economy: 34

%_Increase Family: 20

%_Increase Freedom: 21

%_Increase Trust: 97

%_Increase Health: 33

Algeria	EconomyDifShow 0,39	FamilyDifShow -0.06 !
MeasuresDifShow -0.03 !	FreedomDifShow -0.29 !	HealthDifShow 0,04
ScoreDifShow -0.31 !	GenerosityDifShow -0.1 !	TrustDifShow 0,01