This code connects Google Drive to Google Colab, allowing to access and use files stored in your Drive within the notebook.

```
Start coding or generate with AI.
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Listing the files in the google drive

```
!ls /content/drive/MyDrive
```

```
 17126653786084101591701804519246.jpg
'5-CSE3041_PDS-TUPLE_SET-Dr_RAJESH_M-Ref-PPTs (1).gdoc'
'5-CSE3041_PDS-TUPLE_SET-Dr_RAJESH_M-Ref-PPTs (2).gdoc'
'5-CSE3041_PDS-TUPLE_SET-Dr_RAJESH_M-Ref-PPTs (3).gdoc'
'5-CSE3041_PDS-TUPLE_SET-Dr_RAJESH_M-Ref-PPTs (4).gdoc'
 5-CSE3041_PDS-TUPLE_SET-Dr_RAJESH_M-Ref-PPTs.gdoc
 6-CSE3041_PDS-DICT_FUNC-Dr_RAJESH_M-Ref-PPTs.gdoc
 7-CSE3041_Class-Package_File-Dr_RAJESH_M-Ref-PPTs.gdoc
 880cf805-1213-47b7-8f93-264ae9af0e08_payment_confirmation_20250236-081430.png
 980a7924-37a7-43c3-8609-455cac38164f_payment_confirmation_20250241-110511.png
 Classroom
'Colab Notebooks'
 IMG-20231006-WA0040.jpg
 IMG_20231103_232206_006.jpg
 IMG_20240409_200114_280.jpg
 IMG-20240727-WA0037.jpg
 IMG_20240808_220857.jpg
 IMG_20240920_165749.jpg
 IMG-20241111-WA0029.jpg
 IMG_9675.JPG
 IPS-6.gdoc
'Mgt advertisement '
'nelson mandela .docx'
'pat 1 23mia1104.gdoc'
 PHOTO-2025-03-08-15-25-42.jpg
'Rohith T 23MIA1104 - Digital Assignment 1.gdoc'
 Screenshot_2023_0927_092645.jpg
'Screenshot_2023_1105_132953 (1).jpg'
 Screenshot_2023_1105_132953.jpg
'supplychain (1).gsheet'
 supplychain.gsheet
 Untitled6.ipynb
'watson (1).gsheet'
 watson.gsheet
 WIID_28NOV2023.csv
```

Importing all needed libraries

Loading the dataset and also printing its first five rows

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
file_path = "/content/drive/MyDrive/WIID_28NOV2023.csv"
df = pd.read_csv(file_path)

df.head()
```

	id	country	c3	c2	year	gini	ge0	ge1	ge2	a025	...	population	revision	quality	quality_score	source	sou
0	1	Afghanistan	AFG	AF	2008	29.00	NaN	NaN	NaN	NaN	...	\t26,427,200	New 2019	High	12	National statistical authority	
1	2	Afghanistan	AFG	AF	2012	33.00	NaN	NaN	NaN	NaN	...	\t30,466,478	New 2019	High	12	National statistical authority	
2	3	Afghanistan	AFG	AF	2017	31.00	NaN	NaN	NaN	NaN	...	\t35,643,416	New 2019	High	12	National statistical authority	
3	4	Albania	ALB	AL	1996	27.01	NaN	NaN	NaN	NaN	...	\t3,271,331	New 2023	Average	13	World Bank	In
4	5	Albania	ALB	AL	2002	31.74	NaN	NaN	NaN	NaN	...	\t3,123,551	New 2023	Average	13	World Bank	In

5 rows × 68 columns

Double-click (or enter) to edit

Printing the the dataset info and head and general info about the dataset.

```
df.head()
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24367 entries, 0 to 24366
Data columns (total 68 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   24367 non-null  int64
 1   country              24367 non-null  object
 2   c3                   24367 non-null  object
 3   c2                   24357 non-null  object
 4   year                 24367 non-null  int64
 5   gini                 24279 non-null  float64
 6   ge0                  11723 non-null  float64
 7   ge1                  13395 non-null  float64
 8   ge2                  11784 non-null  float64
 9   a025                 10989 non-null  float64
 10  a050                 13325 non-null  float64
 11  a075                 10989 non-null  float64
 12  a1                   13209 non-null  float64
 13  a2                   11714 non-null  float64
 14  palma                19007 non-null  float64
 15  ratio_top20bottom20  19865 non-null  float64
 16  bottom40             18860 non-null  float64
 17  q1                   19030 non-null  float64
 18  q2                   18864 non-null  float64
 19  q3                   18885 non-null  float64
 20  q4                   18891 non-null  float64
 21  q5                   19031 non-null  float64
 22  d1                   18082 non-null  float64
 23  d2                   18074 non-null  float64
 24  d3                   18074 non-null  float64
 25  d4                   18074 non-null  float64
 26  d5                   18094 non-null  float64
 27  d6                   18099 non-null  float64
 28  d7                   18105 non-null  float64
 29  d8                   18102 non-null  float64
 30  d9                   18121 non-null  float64
 31  d10                  18158 non-null  float64
 32  bottom5              11619 non-null  float64
 33  top5                 12270 non-null  float64
 34  resource             24367 non-null  object
 35  resource_detailed    24367 non-null  object
 36  scale                23861 non-null  object
 37  scale_detailed       23861 non-null  object
 38  sharing_unit         23872 non-null  object
 39  reference_unit       24330 non-null  object
 40  areacovr             24365 non-null  object
 41  areacovr_detailed    24365 non-null  object
 42  popcovr              24367 non-null  object
 43  popcovr_detailed     24367 non-null  object
 44  region_un            24230 non-null  object
 45  region_un_sub        24230 non-null  object
 46  region_wb            24367 non-null  object
 47  eu                   24367 non-null  object
 48  oecd                 24367 non-null  object
 49  incomegroup          24213 non-null  object
 50  mean                 17091 non-null  object
 51  median               15478 non-null  object
 52  currency             17474 non-null  object
 53  reference_period     17623 non-null  object
 54  exchangerate         4031 non-null   float64
 58  population           24306 non-null  object
 59  revision             20551 non-null  object
                          24367 non-null  object
                          24367 non-null  int64
 62  source               24367 non-null  object
                          24367 non-null  object
                          19635 non-null  object
 65  survey               19126 non-null  object
 66  link                 8615 non-null   object
                          24367 non-null  object
...t64(3), object(33)
memory usage: 12.6+ MB
```

Finding the null values in the dataset

```
df.isnull().sum()
```

	0
id	0
country	0
c3	0
c2	10
year	0
...	...
source_detailed	0
source_comments	4732
survey	5241
link	15752
wiidcompanion	0

68 rows × 1 columns

	id	year	gini	ge0	ge1	ge2	a025	a050	a07
count	24367.000000	24367.000000	24279.000000	11723.000000	13395.000000	11784.000000	10989.000000	13325.000000	10989.00000
[illegible]	[illegible]	[illegible]3.411622	37.053688	951.307507	814.745016	1566.812687	236.796683	384.168245	681.55702
std	7034.291341	[illegible]14.585084	9.099320	1109.840124	1036.510708	4173.783279	257.855927	485.786143	741.25019
[illegible]	[illegible]	[illegible]67.000000	12.100000	5.530000	5.960000	6.670000	1.440000	2.800000	4.11000
25%	6092.500000	1997.000000	30.420000	22.650000	23.820000	33.407500	5.240000	11.160000	14.99000
50%	12184.000000	2007.000000	35.400000	50.270000	50.400000	136.525000	13.670000	21.380000	34.64000

max	24367.000000	2022.000000	78.600000	5332.580000	5361.630000	159490.990000	1057.520000	2008.490000	3038.0500(

8 rows × 35 columns

df.isnull() Returns `True` for missing values and `False` for non-missing values.
sum() Counts the number of missing values in each column.

```
df.dtypes
```

	0
id	int64
country	object
c3	object
c2	object
year	int64
...	...
source_detailed	object
source_comments	object
survey	object
link	object
wiidcompanion	object

68 rows × 1 columns

```
Start coding or generate with AI.
```

```
df.columns
```

```
Index(['id', 'country', 'c3', 'c2', 'year', 'gini', 'ge0', 'ge1', 'ge2',
       'a025', 'a050', 'a075', 'a1', 'a2', 'palma', 'ratio_top20bottom20',
       'bottom40', 'q1', 'q2', 'q3', 'q4', 'q5', 'd1', 'd2', 'd3', 'd4', 'd5',
       'd6', 'd7', 'd8', 'd9', 'd10', 'bottom5', 'top5', 'resource',
       'resource_detailed', 'scale', 'scale_detailed', 'sharing_unit',
       'reference_unit', 'areacovr', 'areacovr_detailed', 'popcovr',
       'popcovr_detailed', 'region_un', 'region_un_sub', 'region_wb', 'eu',
       'oecd', 'incomegroup', 'mean', 'median', 'currency', 'reference_period',
       'exchangerate', 'mean_usd', 'median_usd', 'gdp', 'population',
       'revision', 'quality', 'quality_score', 'source', 'source_detailed',
       'source_comments', 'survey', 'link', 'wiidcompanion'],
      dtype='object')
```

```
df.isnull().sum()
```

	0
id	0
country	0
c3	0
c2	10
year	0
...	...
source_detailed	0
source_comments	4732
survey	5241
link	15752
wiidcompanion	0

68 rows × 1 columns

```
df.isnull().sum()[df.isnull().sum() > 0]
```

	0
c2	10
gini	88
ge0	12644
ge1	10972
ge2	12583
a025	13378
a050	11042
a075	13378
a1	11158
a2	12653
palma	5360
ratio_top20bottom20	4502
bottom40	5507
q1	5337
q2	5503
q3	5482
q4	5476
q5	5336
d1	6285
d2	6293
d3	6293
d4	6293
d5	6273
d6	6268
d7	6262
d8	6265
d9	6246
d10	6209
bottom5	12748
top5	12097
scale	506

```
df.duplicated().sum()
```

0

sharing_unit	495
reference_unit	37

```
df[df['c2'].isnull()][['country']]
```

areacovr_detailed	2
region_un	137
region_un_sub	137
incomegroup	154
mean	7276
median	8889
currency	6893
reference_period	6744
exchangerate	20336
mean_usd	24303
median_usd	24353

	country
14990	Namibia
14991	Namibia
14992	Namibia
14993	Namibia
14994	Namibia
14995	Namibia
14996	Namibia
14997	Namibia
14998	Namibia

```
df.loc[df['country'] == 'Namibia', 'c2'] = 'NA'
```

```
revision          3816
```

```
df['gini'] = df['gini'].astype(float)
df['gini'] = df.groupby('country')['gini'].transform(lambda group: group.interpolate(method='linear', limit_direction='both
df['gini'] = df.groupby('country')['gini'].transform(lambda group: group.fillna(group.median()))
df['gini'] = df['gini'].fillna(df['gini'].median())
```

```
df.drop(columns=['link', 'survey', 'source_comments'], inplace=True)
```

```
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])  # Show only columns with missing values
```

```
c2                       10
ge0                   12644
ge1                   10972
ge2                   12583
a025                  13378
a050                  11042
a075                  13378
a1                    11158
a2                    12653
palma                  5360
ratio_top20bottom20    4502
bottom40               5507
q1                     5337
q2                     5503
q3                     5482
q4                     5476
q5                     5336
d1                     6285
d2                     6293
d3                     6293
d4                     6293
d5                     6273
d6                     6268
d7                     6262
d8                     6265
d9                     6246
d10                    6209
bottom5               12748
top5                  12097
scale                   506
scale_detailed          506
sharing_unit            495
reference_unit           37
areacovr                  2
areacovr_detailed         2
region_un               137
region_un_sub           137
incomegroup             154
mean                   7276
median                 8889
currency               6893
reference_period       6744
exchangerate          20336
mean_usd              24303
median_usd            24353
gdp                     116
population               61
revision               3816
source_comments        4732
survey                 5241
link                  15752
dtype: int64
```

```
import pandas as pd

# Define numerical and categorical columns
num_cols = ['ge0', 'ge1', 'ge2', 'a025', 'a050', 'a075', 'a1', 'a2', 'palma', 'ratio_top20bottom20',
            'bottom40', 'q1', 'q2', 'q3', 'q4', 'q5', 'd1', 'd2', 'd3', 'd4', 'd5', 'd6', 'd7', 'd8',
            'd9', 'd10', 'bottom5', 'top5', 'mean', 'median', 'gdp', 'population', 'revision']

cat_cols = ['currency', 'region_un', 'region_un_sub', 'incomegroup', 'scale', 'scale_detailed',
            'sharing_unit', 'reference_unit', 'areacovr', 'areacovr_detailed']

# Convert numeric columns to proper format (handling numbers stored as text)
for col in num_cols:
    df.loc[:, col] = pd.to_numeric(df[col].astype(str).str.replace(',', '').str.strip(), errors='coerce')

# Fill missing numerical values with median
for col in num_cols:
    df.loc[:, col] = df[col].fillna(df[col].median())

# Fill missing categorical values with mode
for col in cat_cols:
```

```
    df.loc[:, col] = df[col].astype(str).str.strip()  # Remove extra spaces
    df.loc[:, col] = df[col].fillna(df[col].mode()[0])

# Check for remaining missing values
print(df.isnull().sum()[df.isnull().sum() > 0])
```

```
c2                      10
gini                    88
reference_period      6744
exchangerate         20336
mean_usd             24303
median_usd           24353
revision             24367
source_comments       4732
survey                5241
link                 15752
dtype: int64
<ipython-input-13-34e05eb9e5a5>:17: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprec
  df.loc[:, col] = df[col].fillna(df[col].median())
<ipython-input-13-34e05eb9e5a5>:17: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprec
  df.loc[:, col] = df[col].fillna(df[col].median())
<ipython-input-13-34e05eb9e5a5>:17: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprec
  df.loc[:, col] = df[col].fillna(df[col].median())
<ipython-input-13-34e05eb9e5a5>:17: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprec
  df.loc[:, col] = df[col].fillna(df[col].median())
/usr/local/lib/python3.11/dist-packages/numpy/lib/_nanfunctions_impl.py:1231: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
<ipython-input-13-34e05eb9e5a5>:17: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprec
  df.loc[:, col] = df[col].fillna(df[col].median())
```

```
df.drop(columns=['revision'], inplace=True)
```

```
df.info()
```

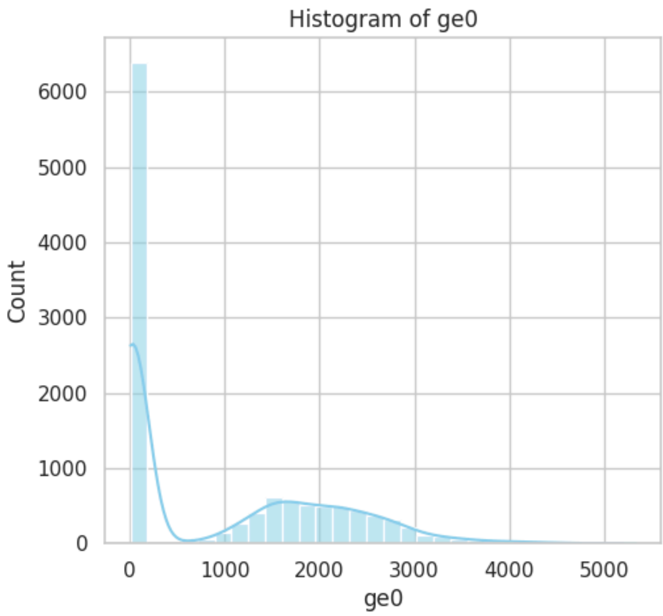
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24367 entries, 0 to 24366
Data columns (total 64 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   id                   24367 non-null  int64  
 1   country              24367 non-null  object 
 2   c3                   24367 non-null  object 
 3   c2                   24367 non-null  object 
 4   year                 24367 non-null  int64  
 5   gini                 24367 non-null  float64
 6   ge0                  24367 non-null  float64
 7   ge1                  24367 non-null  float64
 8   ge2                  24367 non-null  float64
 9   a025                 24367 non-null  float64
 10  a050                 24367 non-null  float64
 11  a075                 24367 non-null  float64
 12  a1                   24367 non-null  float64
 13  a2                   24367 non-null  float64
 14  palma                24367 non-null  float64
 15  ratio_top20bottom20  24367 non-null  float64
 16  bottom40             24367 non-null  float64
 17  q1                   24367 non-null  float64
 18  q2                   24367 non-null  float64
 19  q3                   24367 non-null  float64
 20  q4                   24367 non-null  float64
 21  q5                   24367 non-null  float64
 22  d1                   24367 non-null  float64
 23  d2                   24367 non-null  float64
 24  d3                   24367 non-null  float64
 25  d4                   24367 non-null  float64
 26  d5                   24367 non-null  float64
 27  d6                   24367 non-null  float64
 28  d7                   24367 non-null  float64
 29  d8                   24367 non-null  float64
 30  d9                   24367 non-null  float64
 31  d10                  24367 non-null  float64
 32  bottom5              24367 non-null  float64
 33  top5                 24367 non-null  float64
 34  resource             24367 non-null  object 
 35  resource_detailed    24367 non-null  object 
 36  scale                24367 non-null  object 
 37  scale_detailed       24367 non-null  object 
 38  sharing_unit         24367 non-null  object 
 39  reference_unit       24367 non-null  object 
 40  areacovr             24367 non-null  object 
 41  areacovr_detailed    24367 non-null  object 
 42  popcovr              24367 non-null  object 
 43  popcovr_detailed     24367 non-null  object 
 44  region_un            24367 non-null  object 
 45  region_un_sub        24367 non-null  object 
```

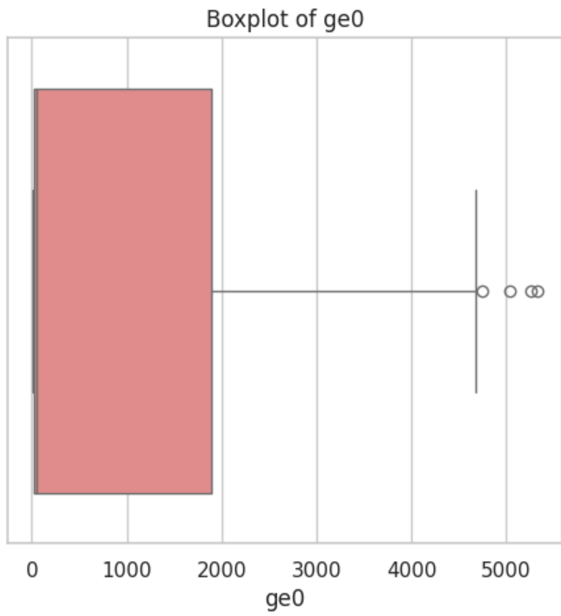
```
 46  region_wb            24367 non-null  object
 47  eu                   24367 non-null  object
 48  oecd                 24367 non-null  object
 49  incomegroup          24367 non-null  object
 50  mean                 24367 non-null  float64
 51  median               24367 non-null  float64
```

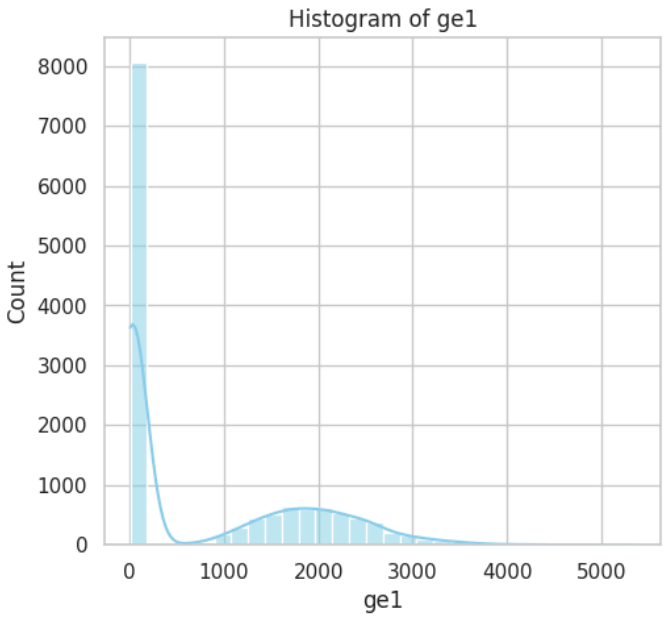
```
# Basic statistical summary for numerical columns
num_cols = ['ge0', 'ge1', 'ge2', 'a025', 'a050', 'a075', 'a1', 'a2', 'palma', 'ratio_top20bottom20',
            'bottom40', 'q1', 'q2', 'q3', 'q4', 'q5', 'd1', 'd2', 'd3', 'd4', 'd5', 'd6', 'd7', 'd8',
            'd9', 'd10', 'bottom5', 'top5', 'mean', 'median', 'gdp', 'population', 'exchangerate',
            'mean_usd', 'median_usd', 'revision']

# Statistical summary for numerical columns
num_summary =  df[num_cols].describe().T
num_summary["skewness"] = df[num_cols].skew(numeric_only=True)

# Basic statistical summary for categorical columns
cat_cols = ['currency', 'region_un', 'region_un_sub', 'incomegroup', 'scale', 'scale_detailed',
            'sharing_unit', 'reference_unit', 'areacovr', 'areacovr_detailed']

cat_summary = df[cat_cols].describe().T

# Display results
print("📌 Numerical Columns Summary:\n", num_summary)
print("\n📌 Categorical Columns Summary:\n", cat_summary)
```

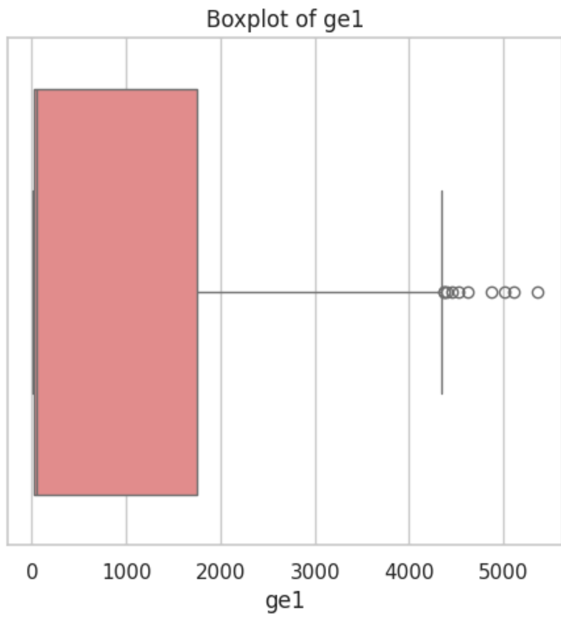
```
📌 Numerical Columns Summary:
                        count         mean          std           min  \
ge0                  11723.0   951.307507  1109.840124  5.530000e+00
ge1                  13395.0   814.745016  1036.510708  5.960000e+00
ge2                  11784.0  1566.812687  4173.783279  6.670000e+00
a025                 10989.0   236.796683   257.855927  1.440000e+00
a050                 13325.0   384.168245   485.786143  2.800000e+00
a075                 10989.0   681.557025   741.250193  4.110000e+00
a1                   13209.0   758.916534   955.202356  5.380000e+00
a2                   11714.0  1985.510212  2295.813624  1.074000e+01
palma                19007.0     1.808576     1.254811  5.200000e-01
ratio_top20bottom20  19865.0     8.145493     6.797569  1.890000e+00
bottom40             18860.0    17.933425     4.302160  1.600000e+00
q1                   19030.0     6.531106     2.074778  2.000000e-01
q2                   18864.0    11.403914     2.306268  1.300000e+00
q3                   18885.0    15.941342     2.119216  4.200000e+00
q4                   18891.0    22.266236     1.597061  1.060000e+01
q5                   19031.0    43.877454     7.354925  2.550000e+01
d1                   18082.0     2.469968     0.950078 -1.000000e-01
d2                   18074.0     4.077017     1.138688  2.000000e-01
d3                   18074.0     5.178075     1.166559  5.000000e-01
d4                   18074.0     6.235170     1.150456  8.000000e-01
d5                   18094.0     7.343845     1.105887  1.000000e-01
d6                   18099.0     8.598824     1.021466  2.700000e+00
d7                   18105.0    10.119569     0.898464  4.240000e+00
d8                   18102.0    12.139420     0.742304  6.250000e+00
d9                   18121.0    15.385310     1.007931  8.000000e+00
d10                  18158.0    28.485110     6.833391  1.612000e+01
bottom5              11619.0     0.981927     0.387118 -1.000000e+00
top5                 12270.0    16.825234     4.560707  9.100000e+00
exchangerate          4031.0     0.694842     6.315205  8.090000e-14
mean_usd                64.0  2793.031250  5251.287207  3.200000e+01
median_usd              14.0  4338.428571  5458.003040  5.710000e+02

                          25%         50%          75%            max  \
ge0                  22.65000   50.270000  1893.650000    5332.580000
ge1                  23.82000   50.400000  1754.240000    5361.630000
ge2                  33.40750  136.525000  2603.402500  159490.990000
a025                  5.24000   13.670000   466.870000    1057.520000
a050                 11.16000   21.380000   826.720000    2008.490000
a075                 14.99000   34.640000  1342.400000    3038.050000
a1                   21.51000   37.610000  1620.230000    4133.090000
a2                   45.19250   70.620000  4110.967500    8397.590000
palma                 1.12000    1.410000     2.030000      39.810000
ratio_top20bottom20   4.76000    6.300000     9.200000     293.000000
bottom40             15.00000   18.355000    21.160000      31.160000
q1                    5.03250    6.580000     8.100000      14.600000
q2                    9.91000   11.790000    13.100000      17.180000
q3                   14.92000   16.560000    17.420000      21.100000
q4                   21.70000   22.570000    23.190000      31.800000
q5                   38.70000   42.010000    47.655000      81.400000
d1                    1.79000    2.450000     3.180000       7.000000
d2                    3.27000    4.150000     4.950000       7.610000
d3                    4.40000    5.330000     6.040000       8.200000
d4                    5.54000    6.470000     7.080000       8.700000
d5                    6.75000    7.630000     8.130000       9.530000
d6                    8.14000    8.920000     9.290000      10.780000
d7                    9.79000   10.390000    10.680000      13.900000
```

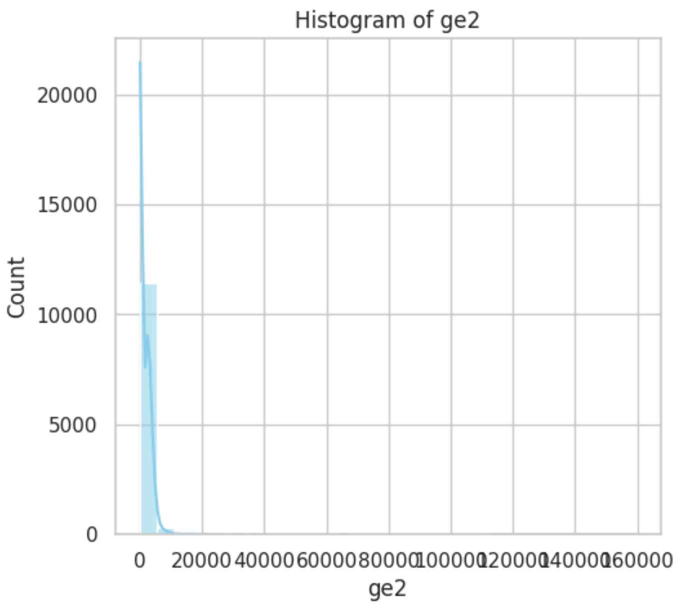
```
# Set plot style
sns.set(style="whitegrid")

# Univariate Analysis for Numerical Columns
for col in num_cols:
    fig, axes = plt.subplots(1, 2, figsize=(12, 5))

    # Histogram
    sns.histplot(df[col], kde=True, bins=30, ax=axes[0], color="skyblue")
    axes[0].set_title(f"Histogram of {col}")

    # Boxplot
    sns.boxplot(x=df[col], ax=axes[1], color="lightcoral")
    axes[1].set_title(f"Boxplot of {col}")

    plt.show()

# Univariate Analysis for Categorical Columns
for col in cat_cols:
    plt.figure(figsize=(10, 5))
    sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
    plt.title(f"Bar Chart of {col}")
    plt.show()
```

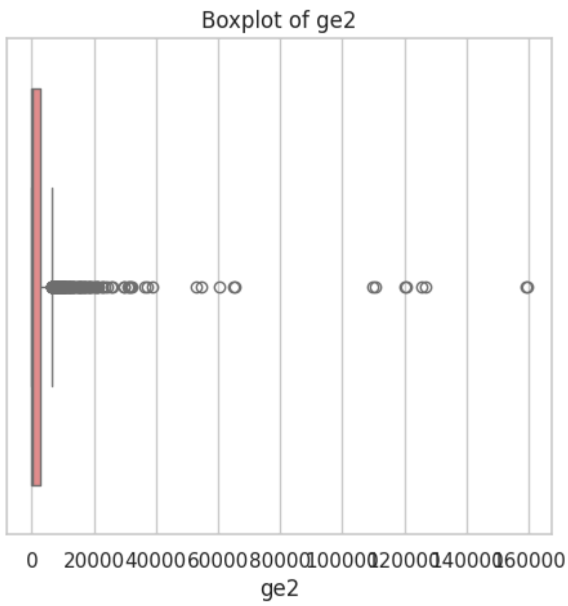

Histogram of ge0
Count
ge0

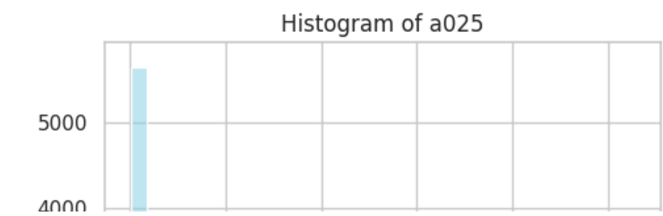

Boxplot of ge0
ge0

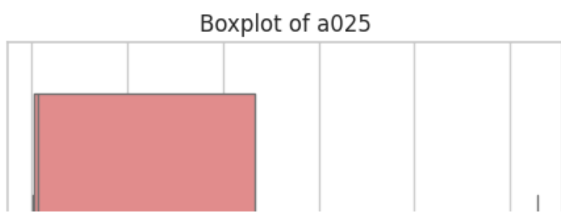

Histogram of ge1
Count
ge1

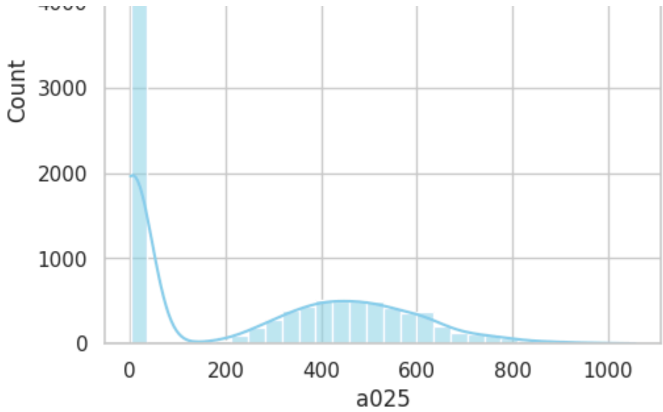

Boxplot of ge1
ge1

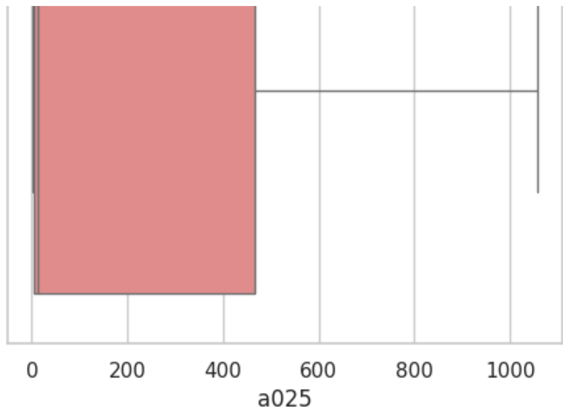

Histogram of ge2
Count
ge2


Boxplot of ge2
ge2

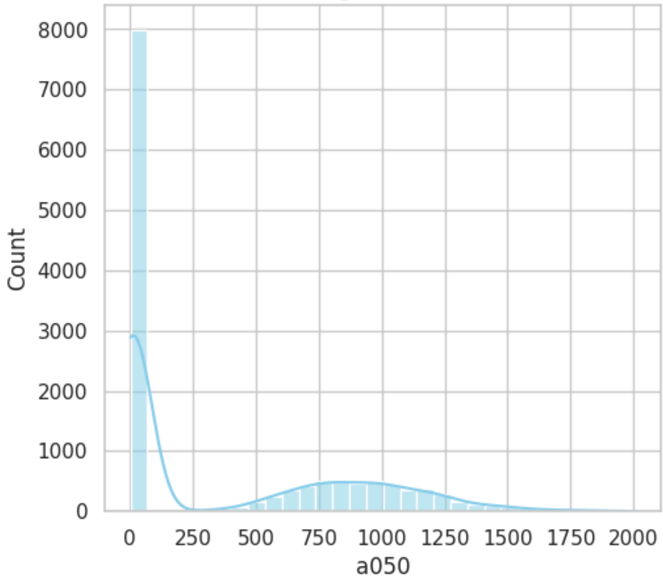

Histogram of a025


Boxplot of a025

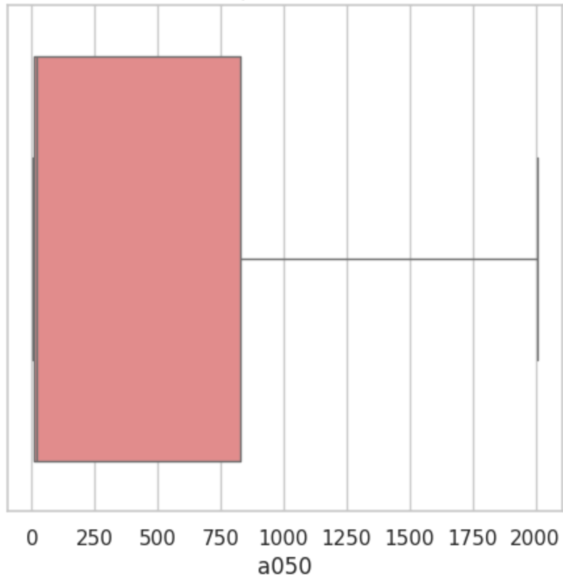




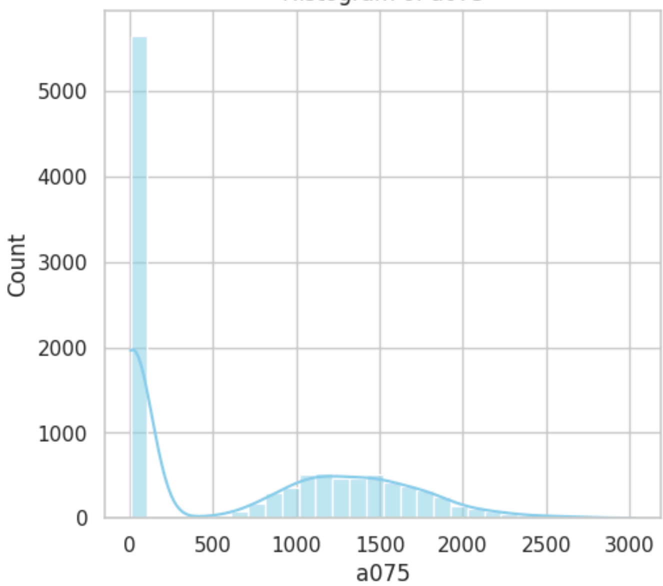
Histogram of a050



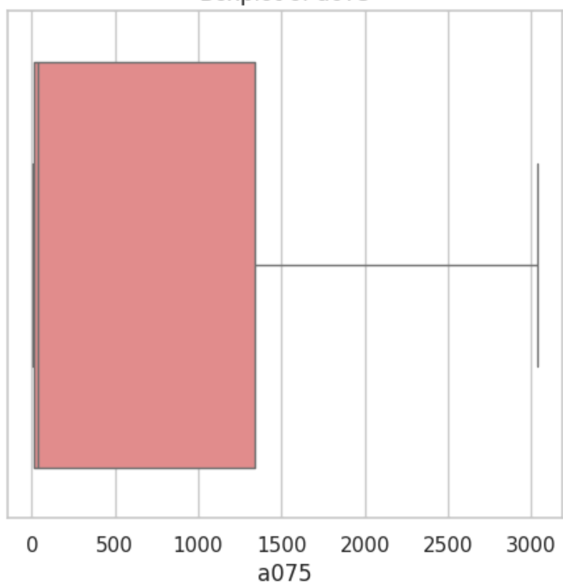
Boxplot of a050



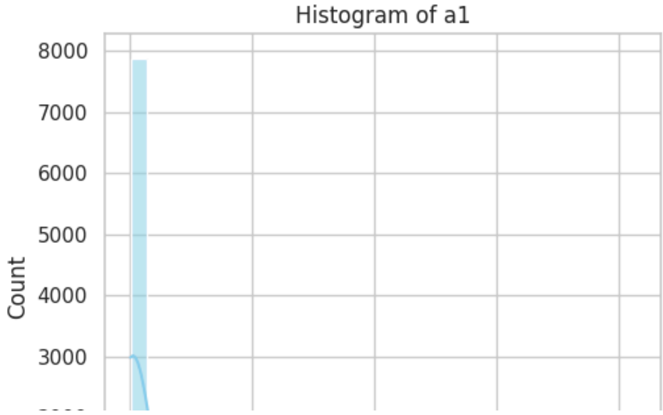
Histogram of a075



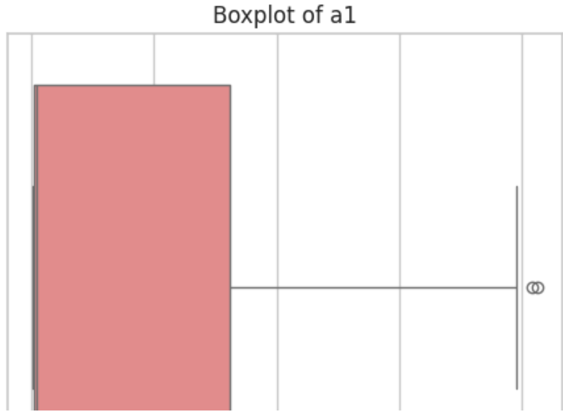
Boxplot of a075

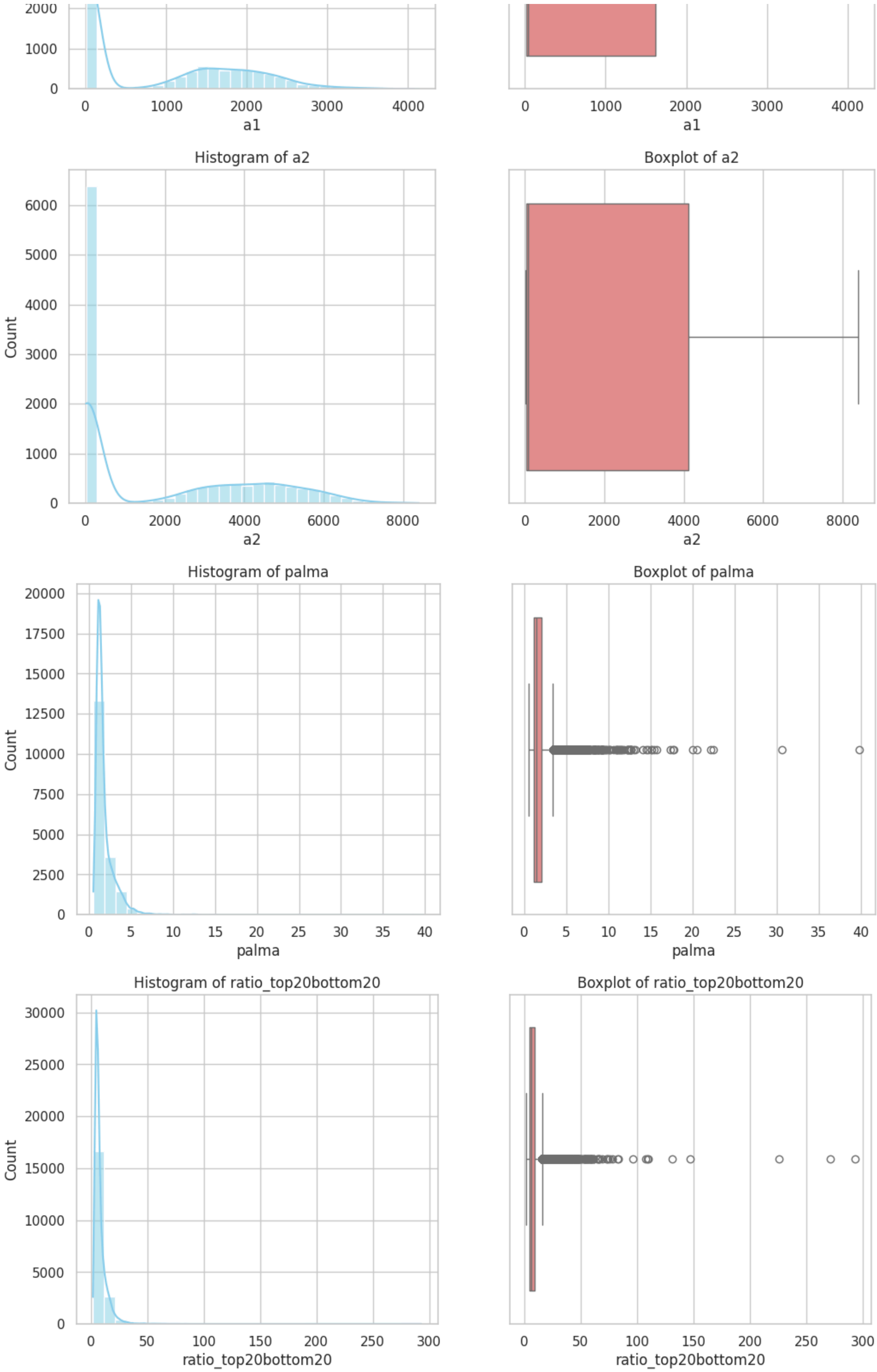


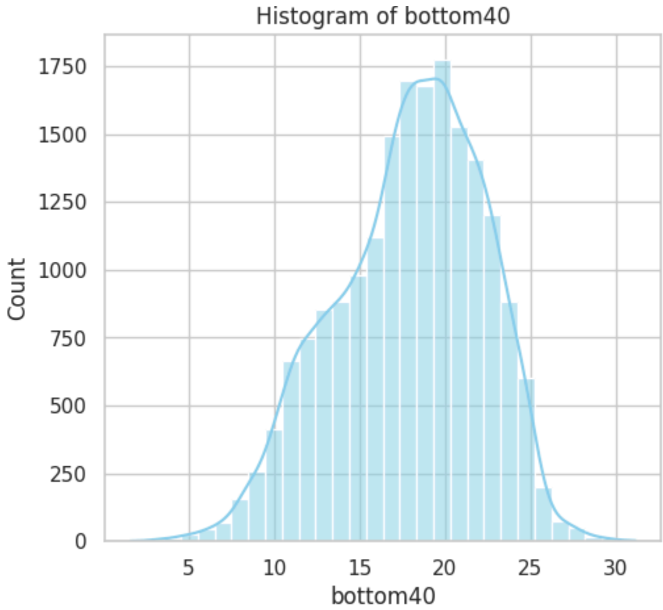
Histogram of a1

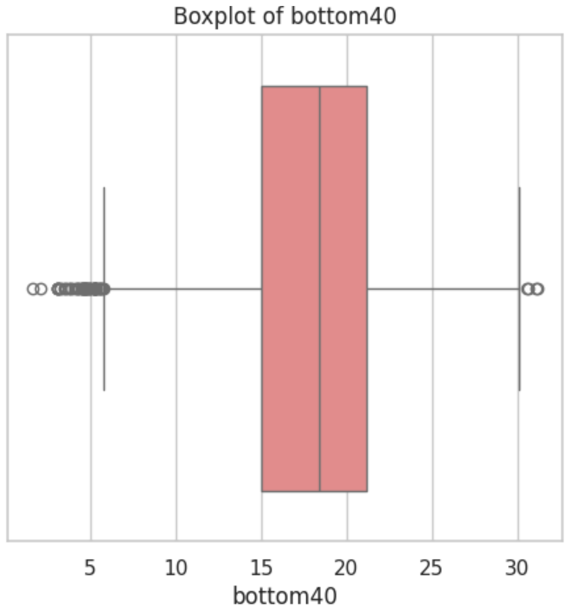


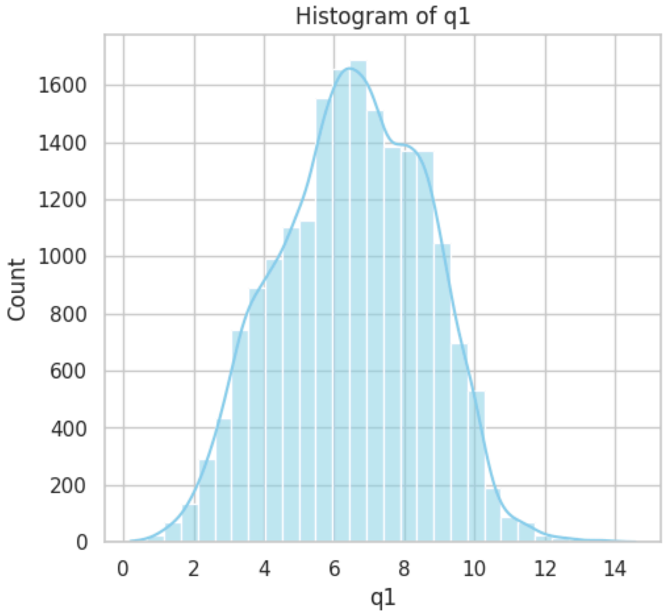
Boxplot of a1

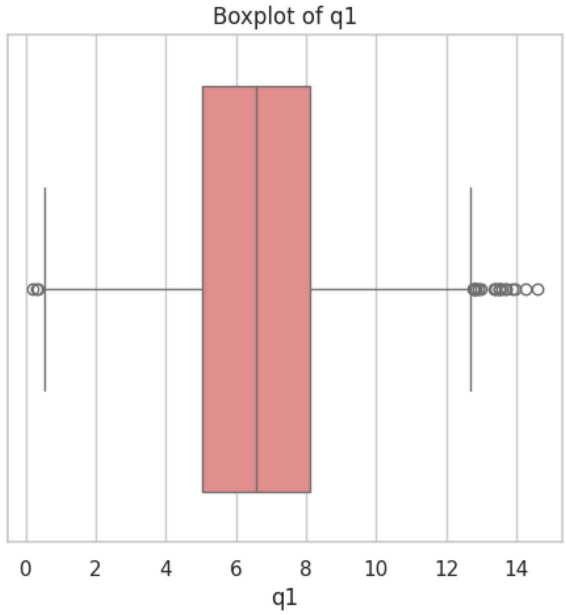


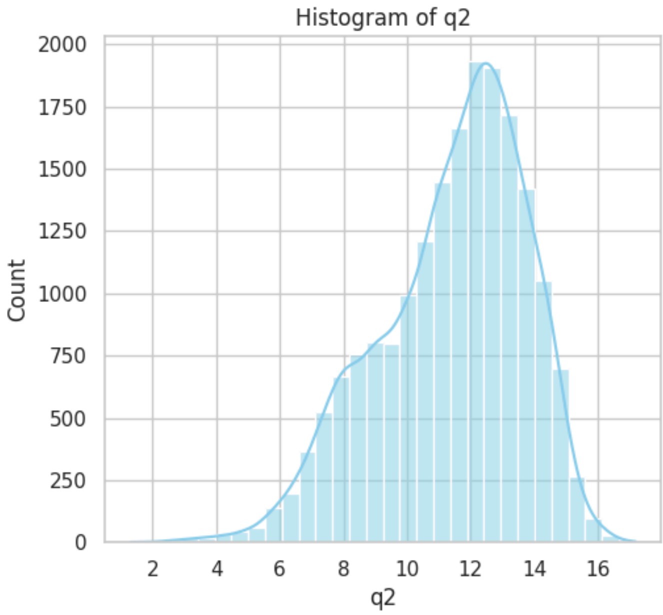

a1
a1
Histogram of a2
Boxplot of a2
Count
a2
a2
Histogram of palma
Boxplot of palma
Count
palma
palma
Histogram of ratio_top20bottom20
Boxplot of ratio_top20bottom20
Count
ratio_top20bottom20
ratio_top20bottom20

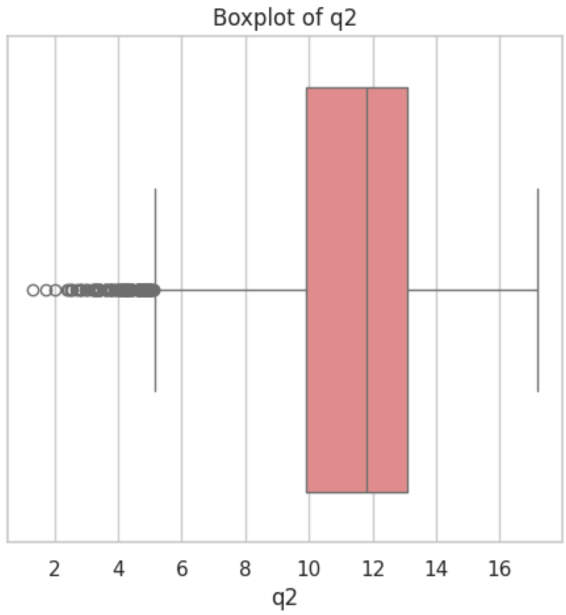

Histogram of bottom40
Count
0
250
500
750
1000
1250
1500
1750
5
10
15
20
25
30
bottom40


Boxplot of bottom40
5
10
15
20
25
30
bottom40

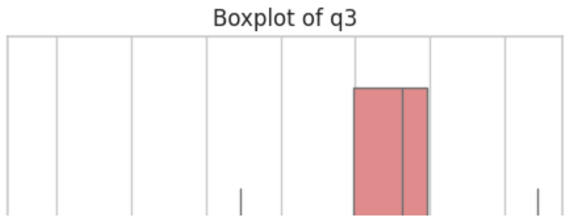

Histogram of q1
Count
0
200
400
600
800
1000
1200
1400
1600
0
2
4
6
8
10
12
14
q1

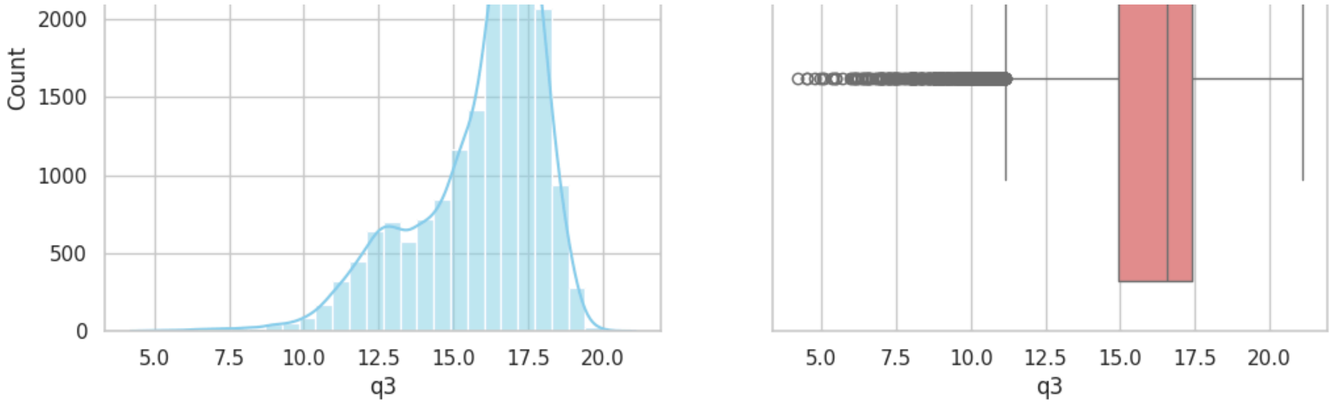

Boxplot of q1
0
2
4
6
8
10
12
14
q1

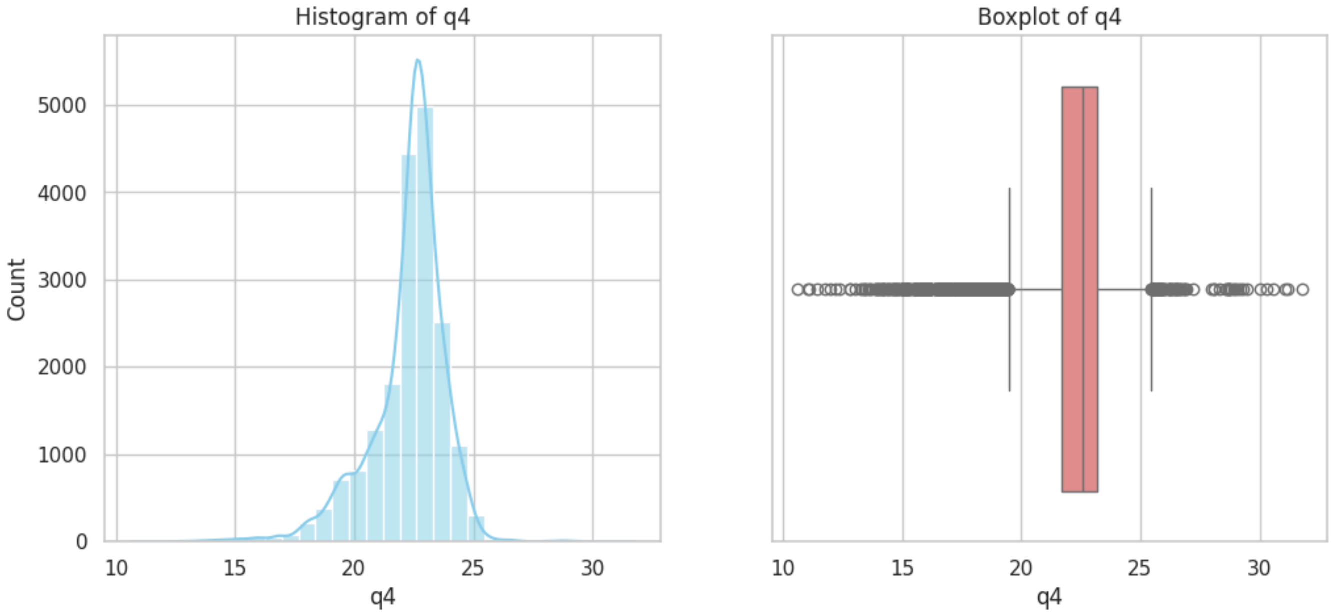

Histogram of q2
Count
0
250
500
750
1000
1250
1500
1750
2000
2
4
6
8
10
12
14
16
q2

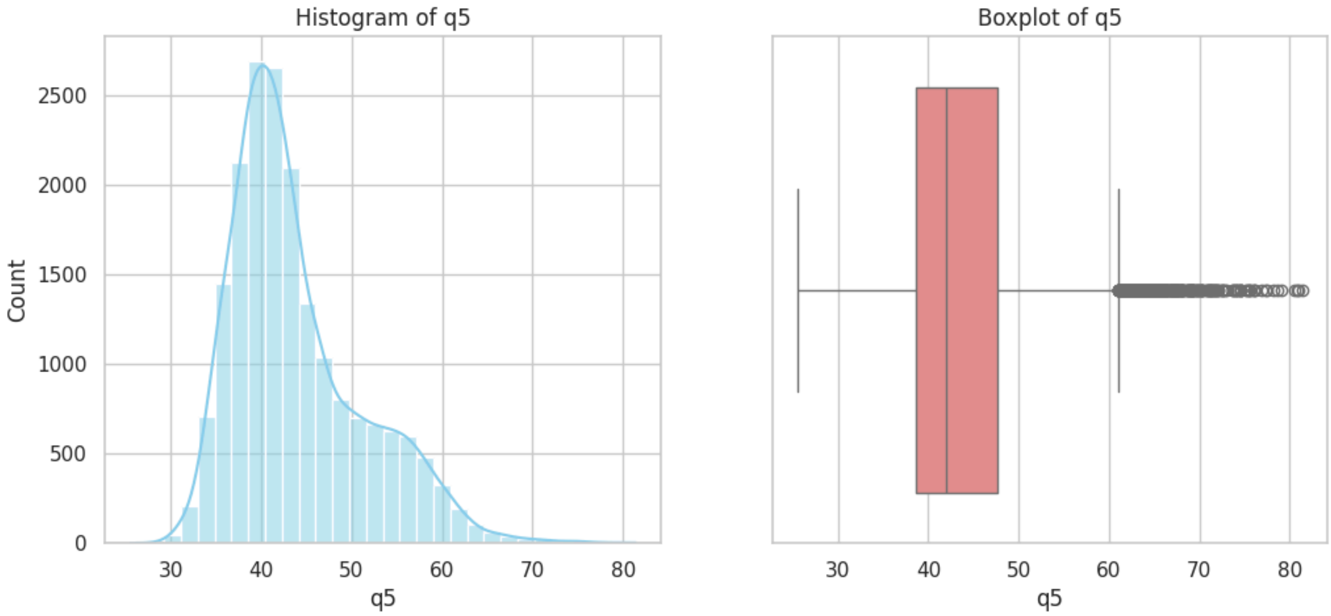

Boxplot of q2
2
4
6
8
10
12
14
16
q2

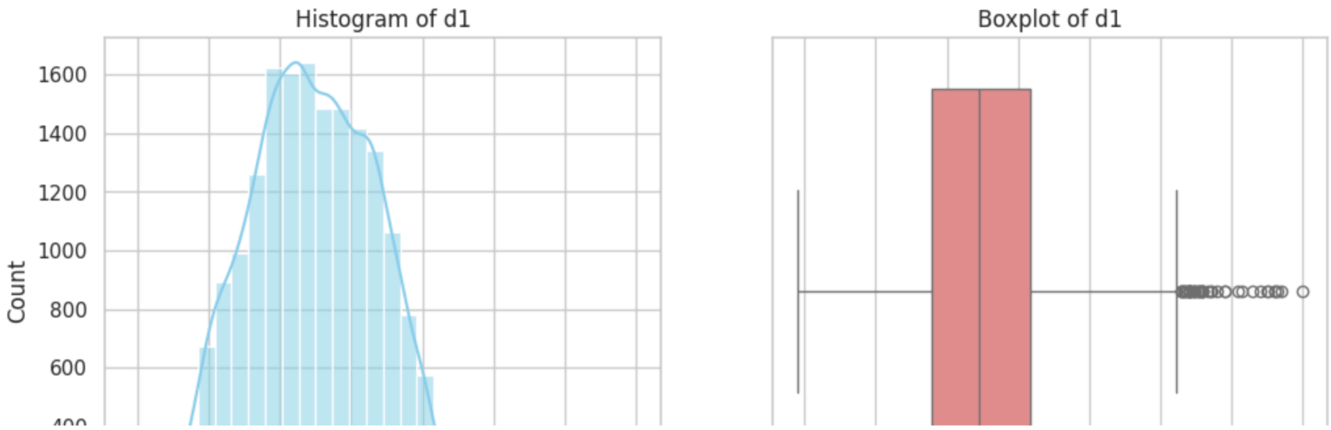

Histogram of q3
3000
2500

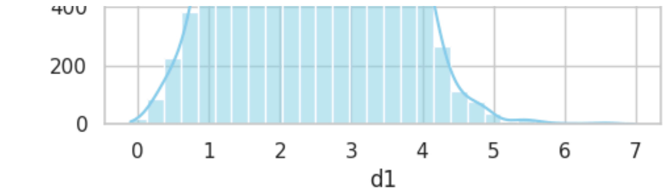

Boxplot of q3

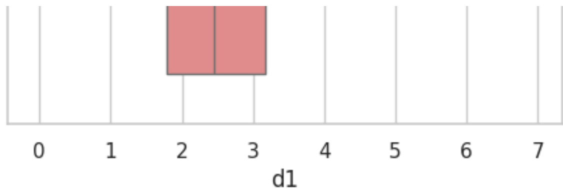

Count
q3
q3

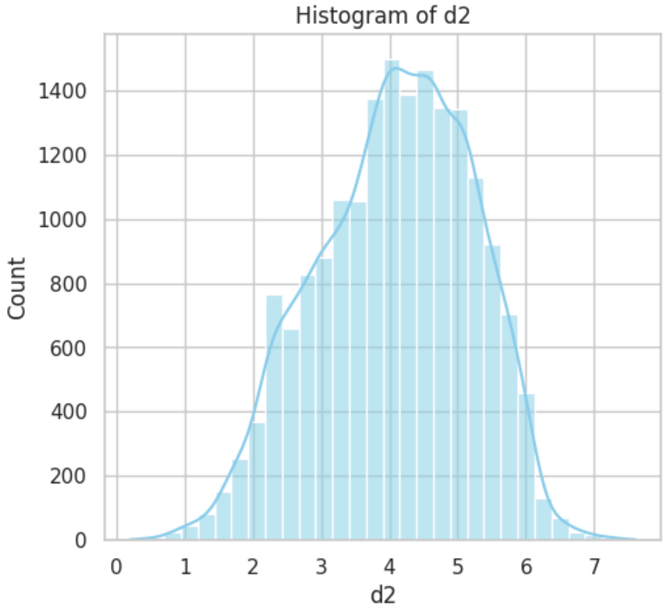

Histogram of q4
Boxplot of q4
Count
q4
q4

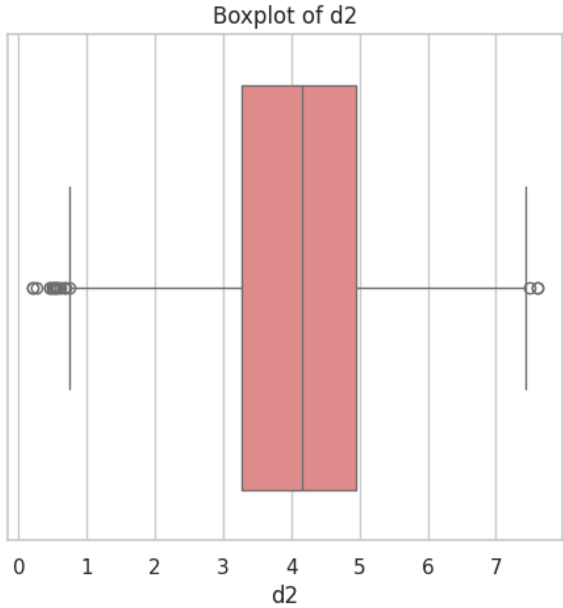

Histogram of q5
Boxplot of q5
Count
q5
q5

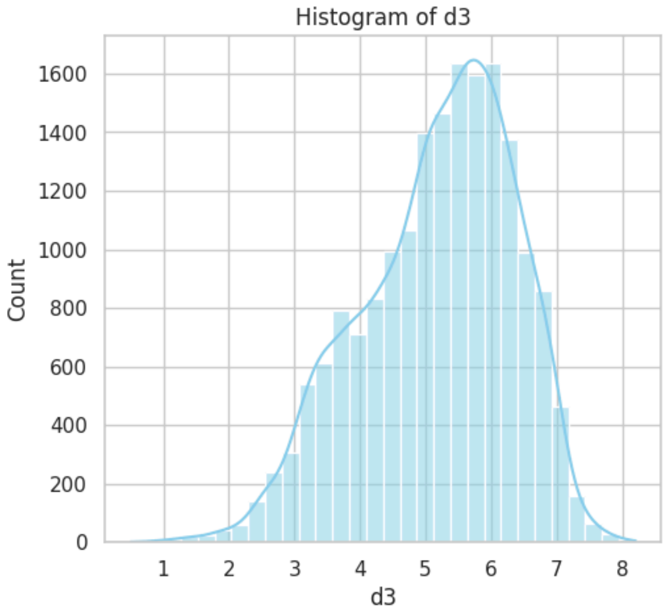

Histogram of d1
Boxplot of d1
Count

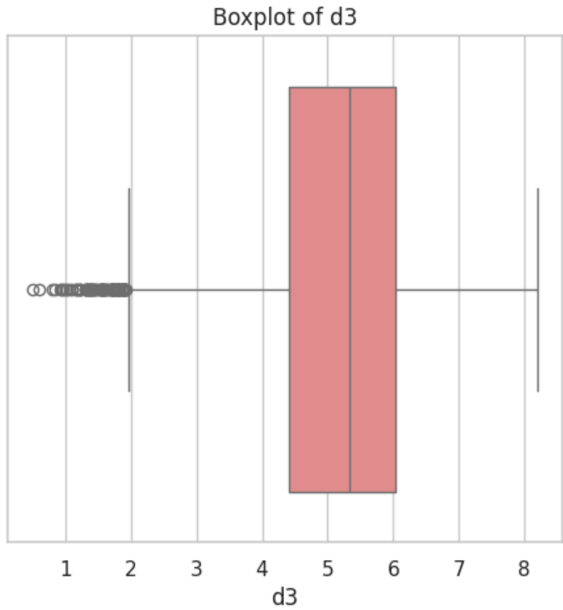

d1

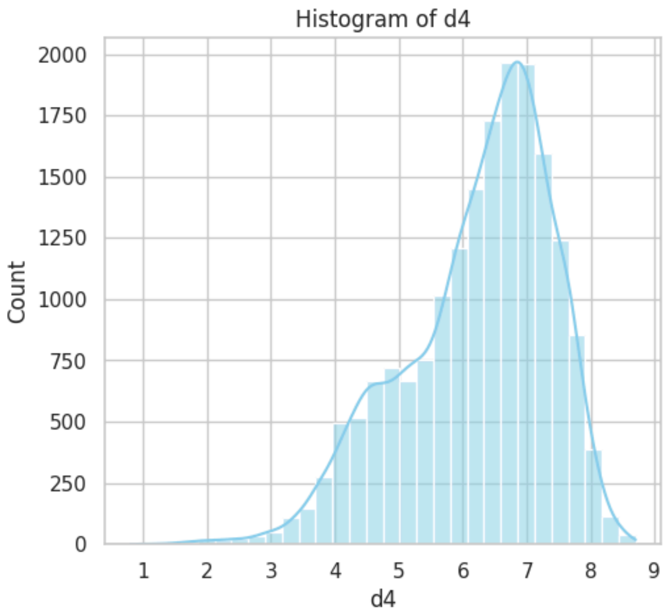

d1

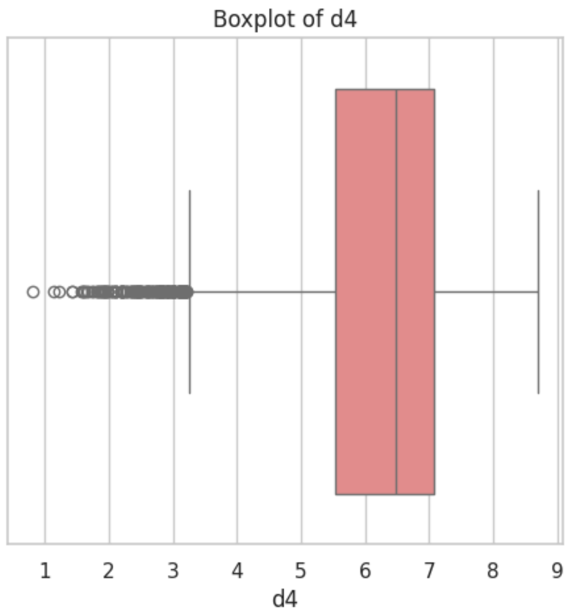

Histogram of d2
Count
d2

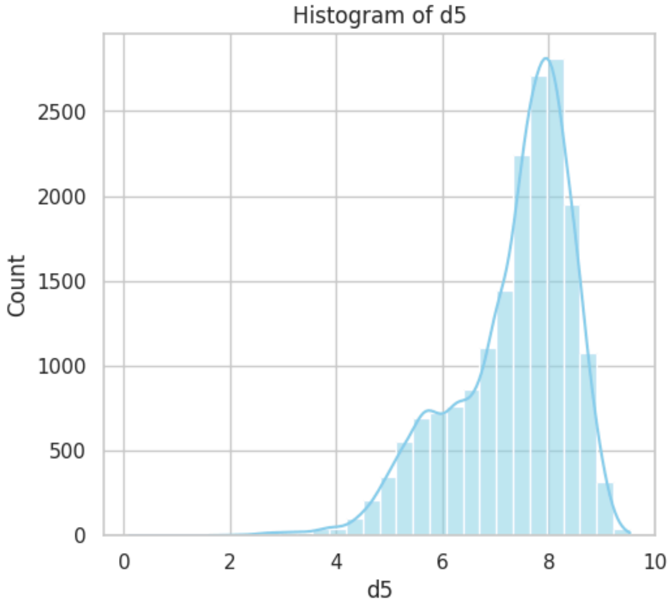

Boxplot of d2
d2

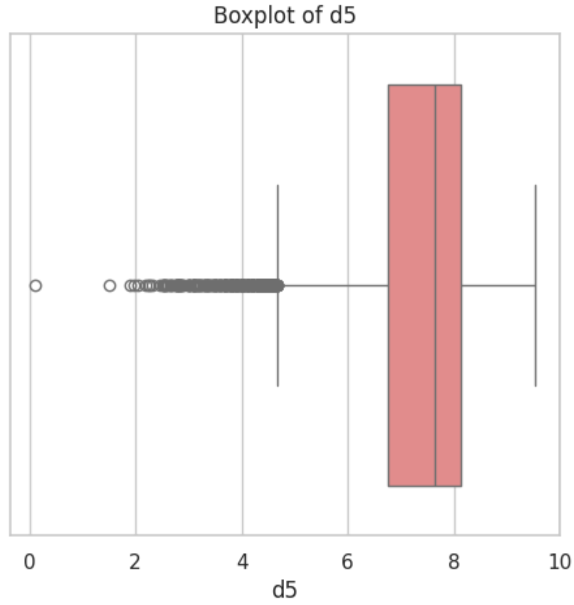

Histogram of d3
Count
d3

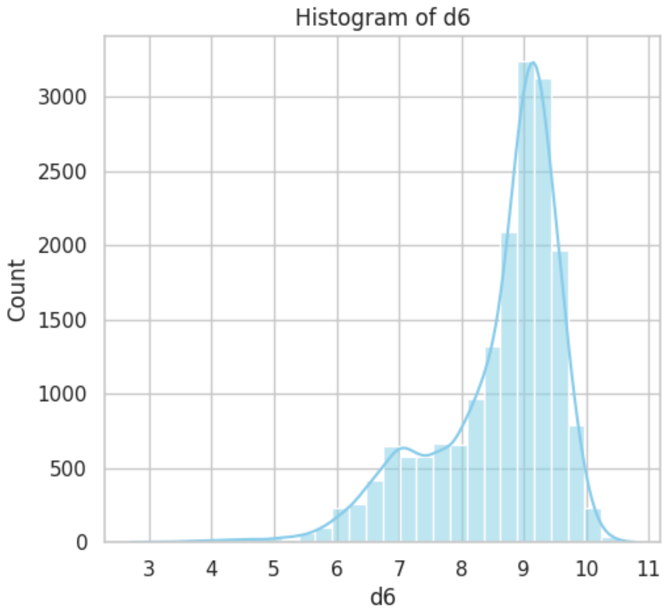

Boxplot of d3
d3

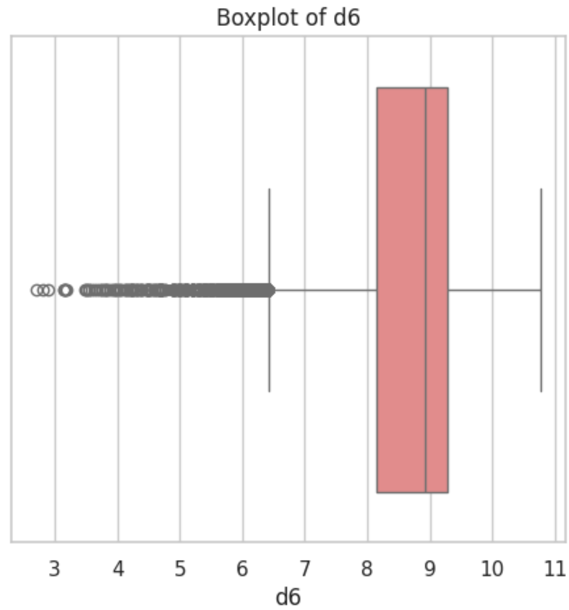

Histogram of d4
Count
d4

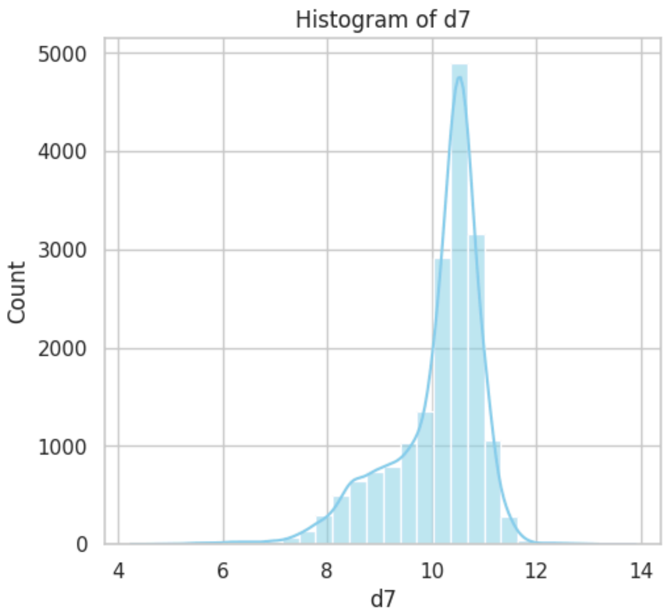

Boxplot of d4
d4

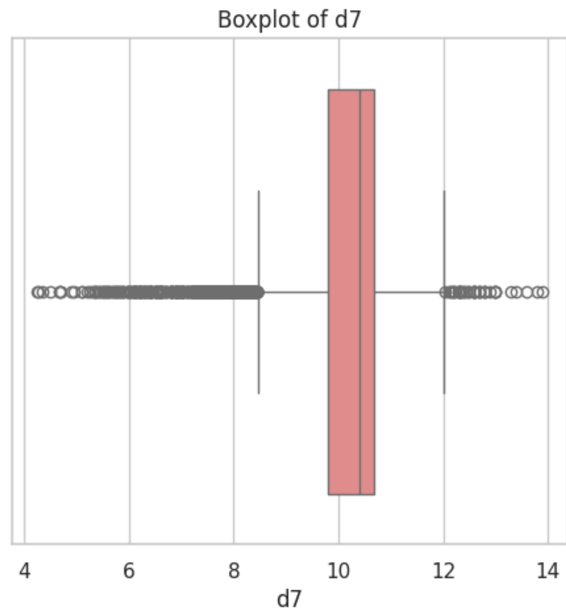

Histogram of d5
Count
d5

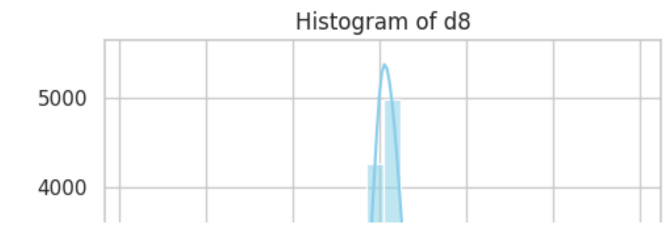

Boxplot of d5
d5

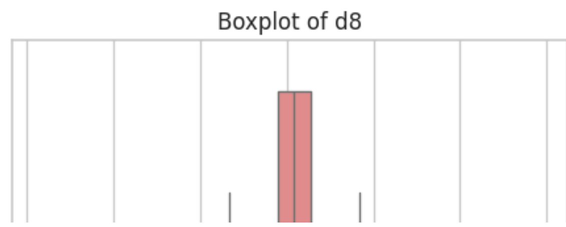

Histogram of d6
Count
d6

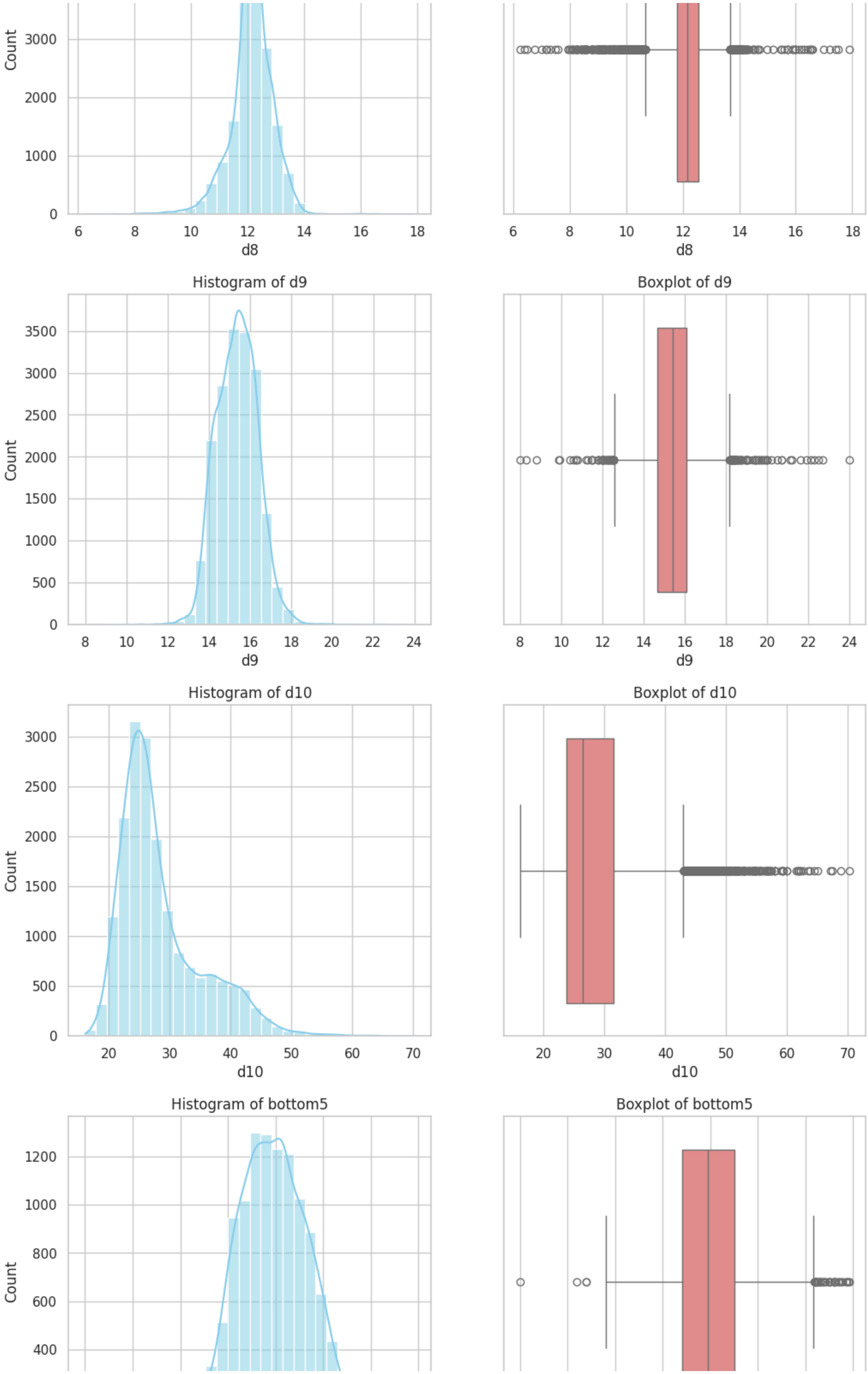

Boxplot of d6
d6

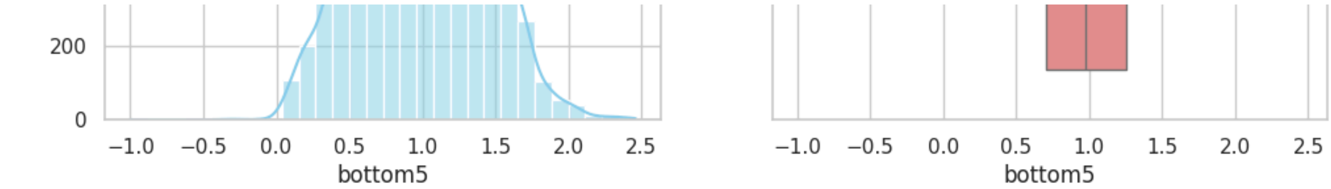

Histogram of d7
Count
d7

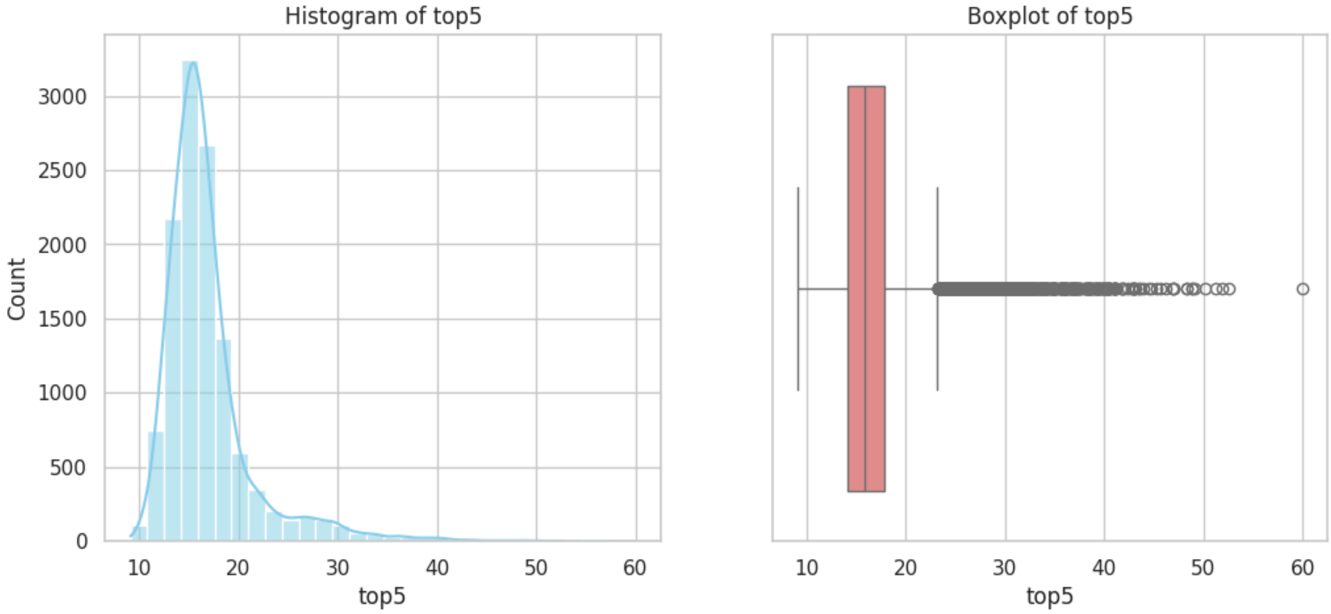

Boxplot of d7
d7


Histogram of d8

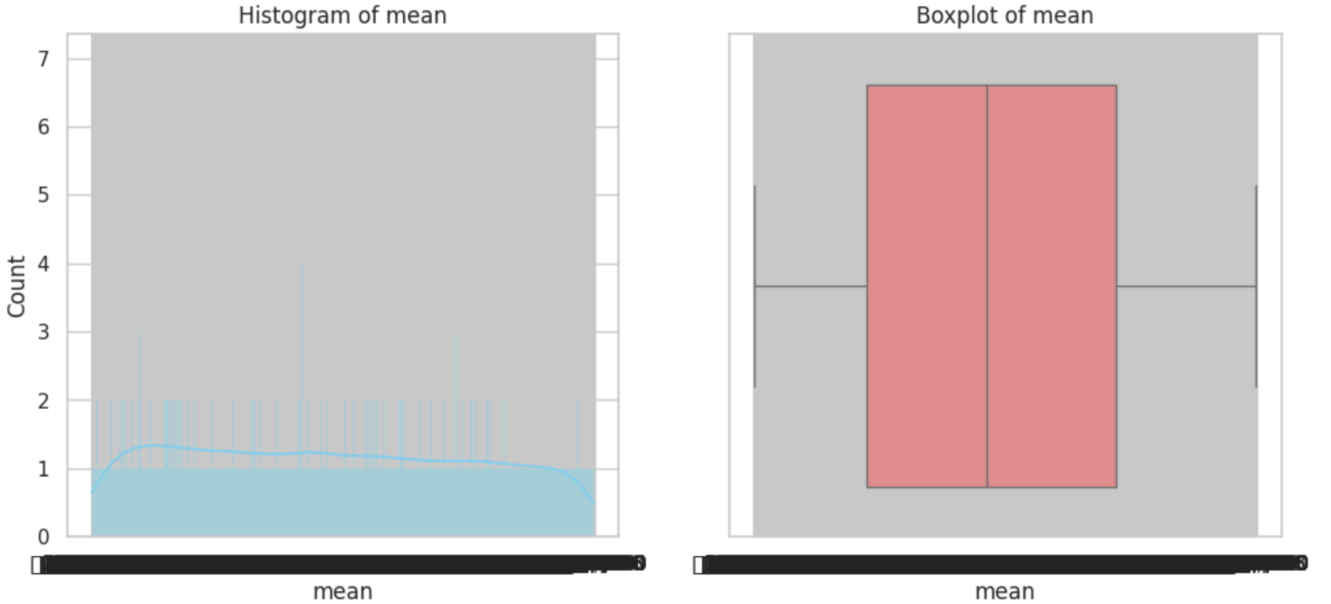

Boxplot of d8


Count
d8
Histogram of d9
Boxplot of d9
d9
Histogram of d10
Boxplot of d10
d10
Histogram of bottom5
Boxplot of bottom5

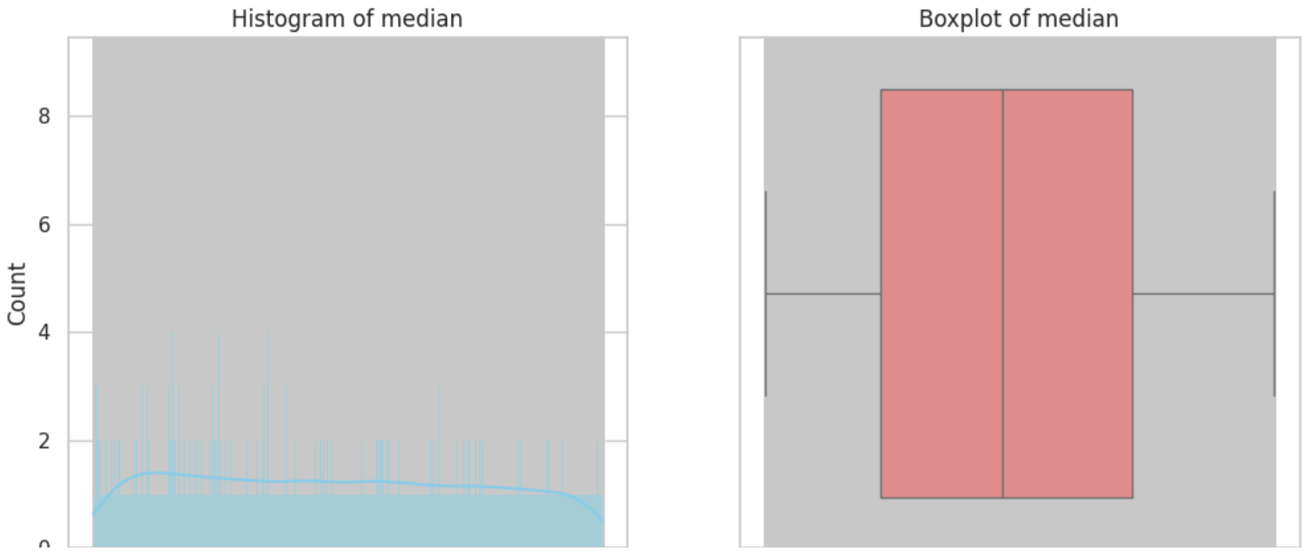




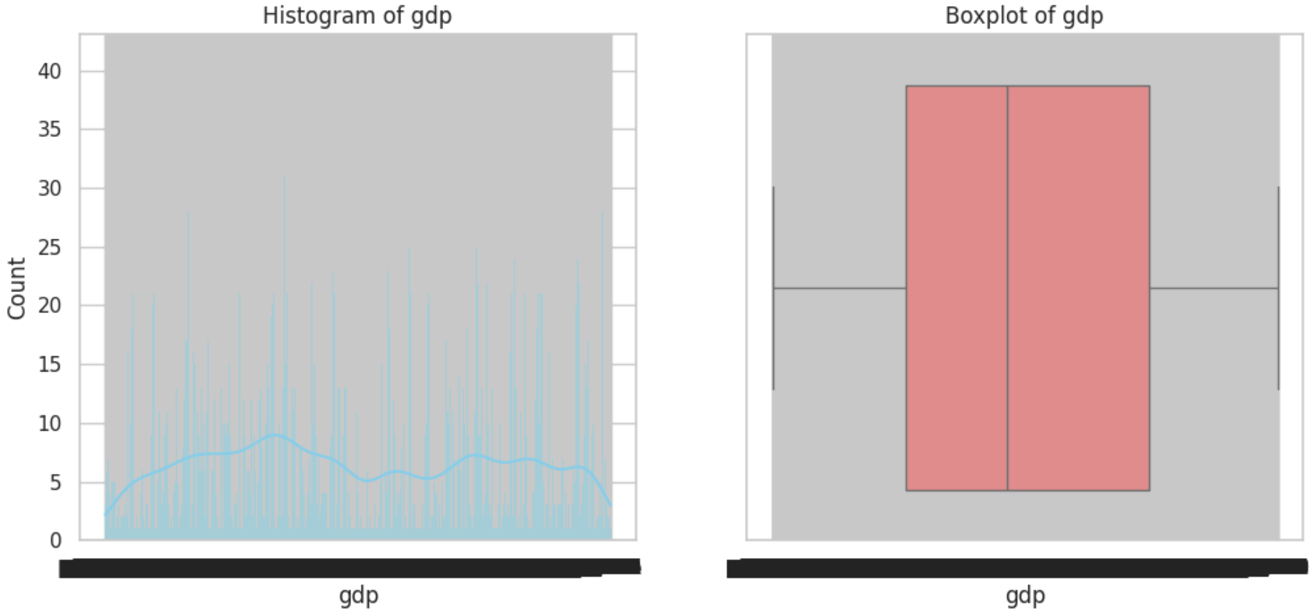

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 9 (	) missing from font(s)
  fig.canvas.print_figure(bytes_io, **kw)
```



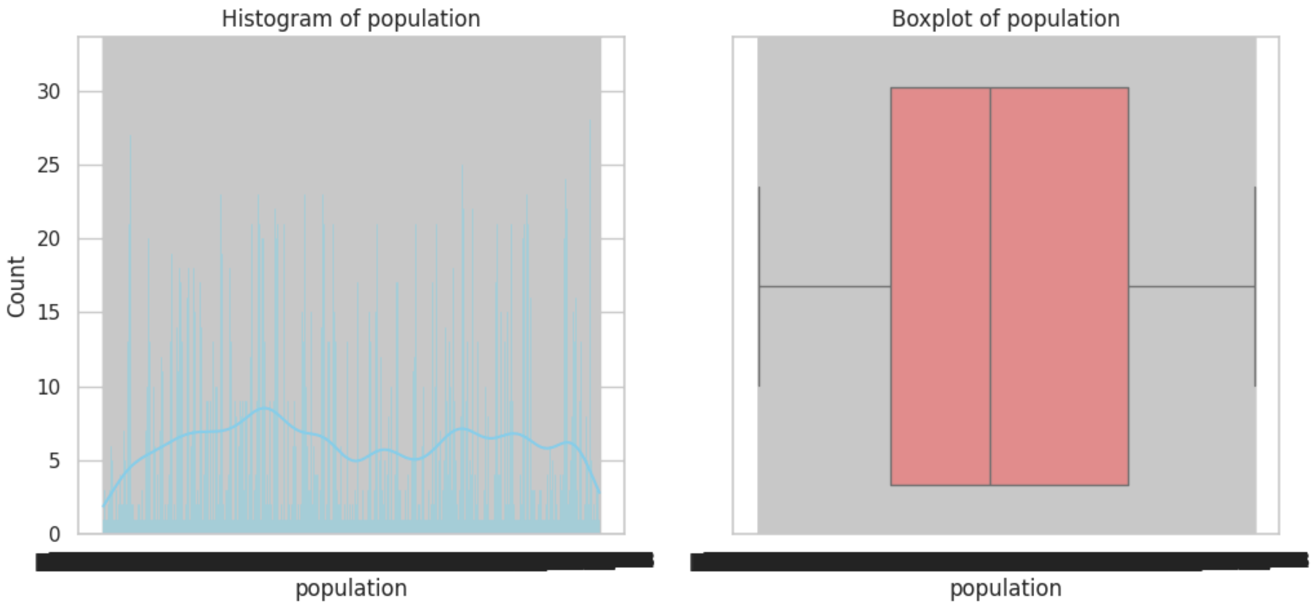

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 9 (	) missing from font(s)
  fig.canvas.print_figure(bytes_io, **kw)
```

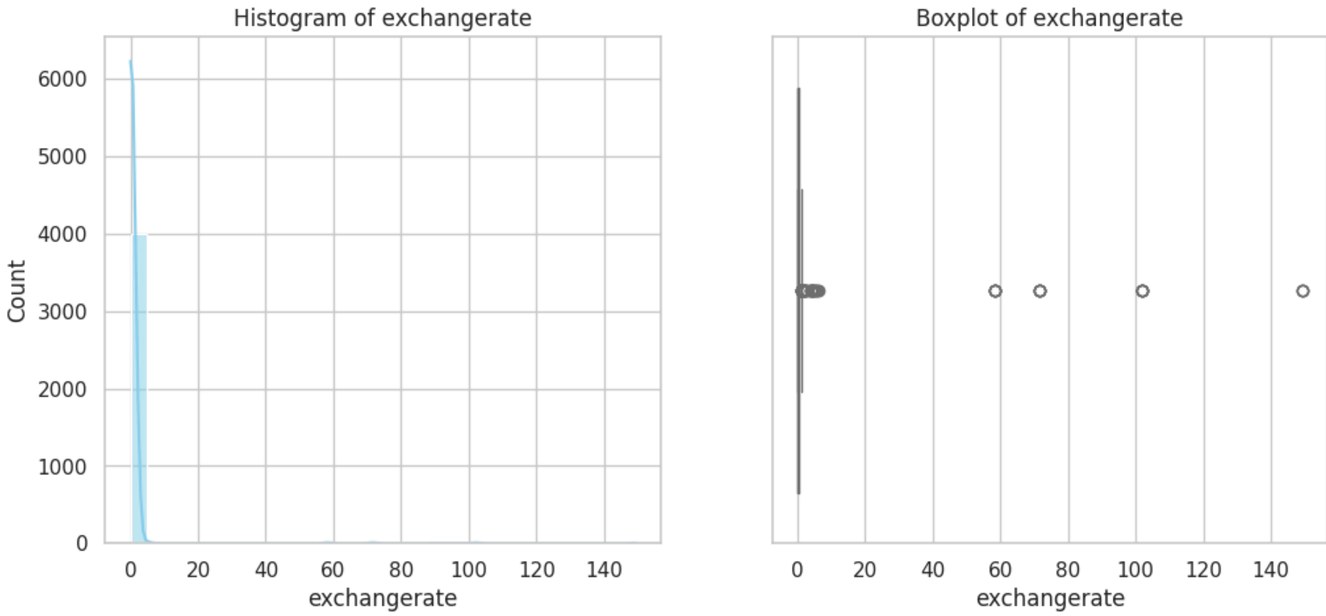


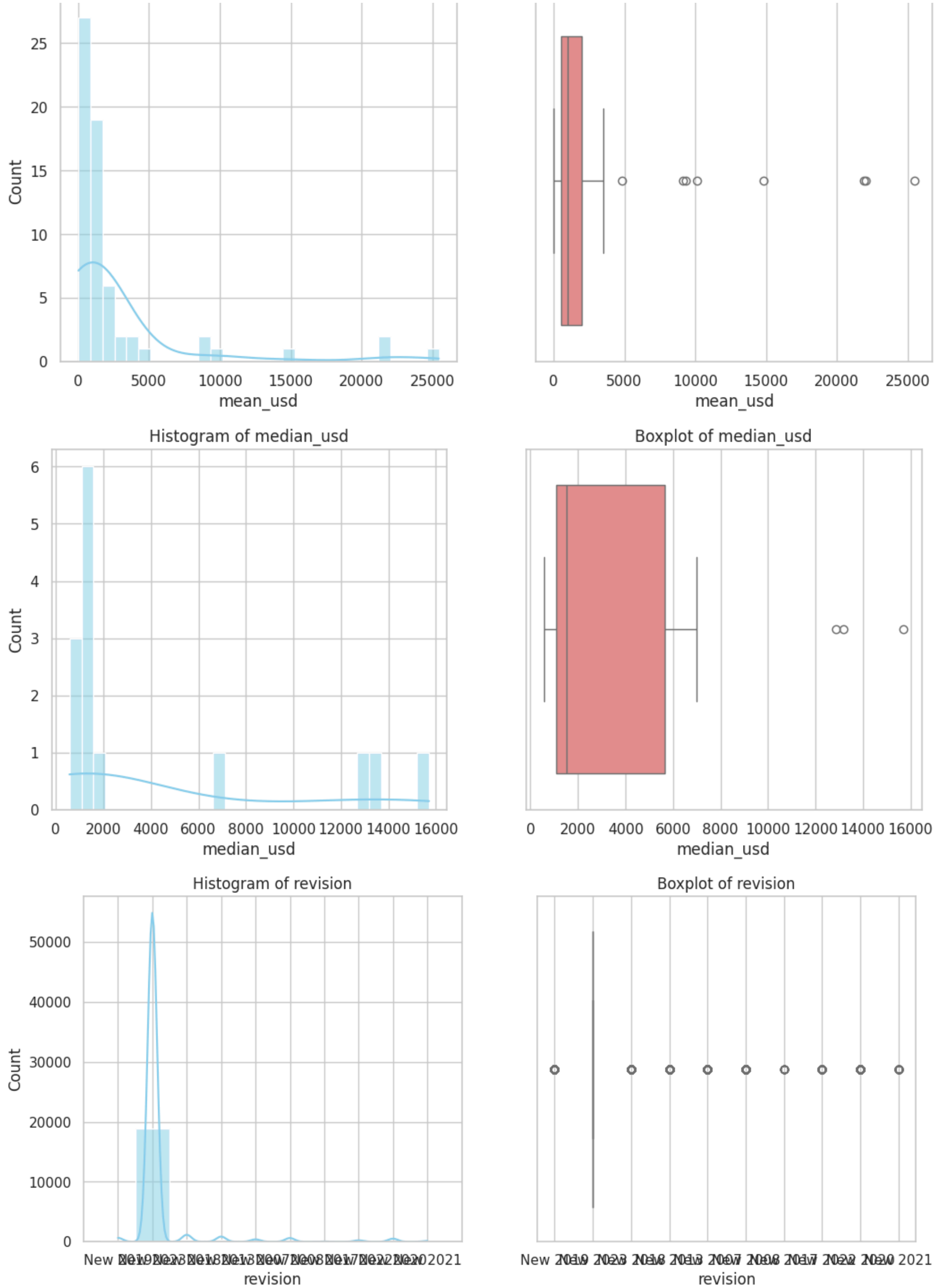
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 9 (	) missing from font(s)
  fig.canvas.print_figure(bytes_io, **kw)



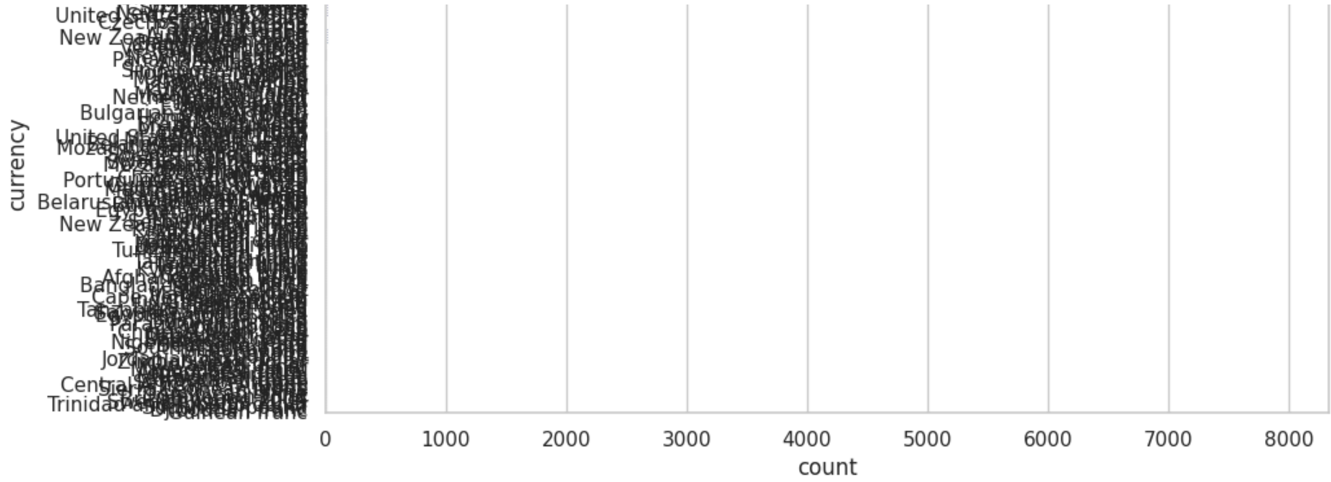
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 9 (	) missing from font(s)
  fig.canvas.print_figure(bytes_io, **kw)



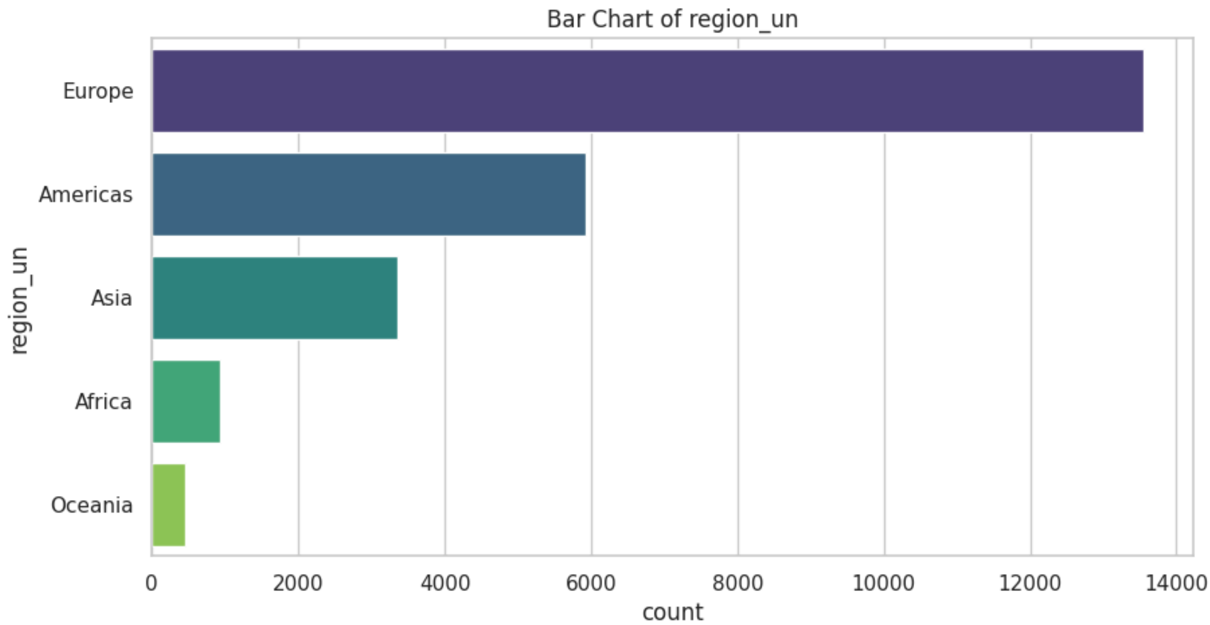




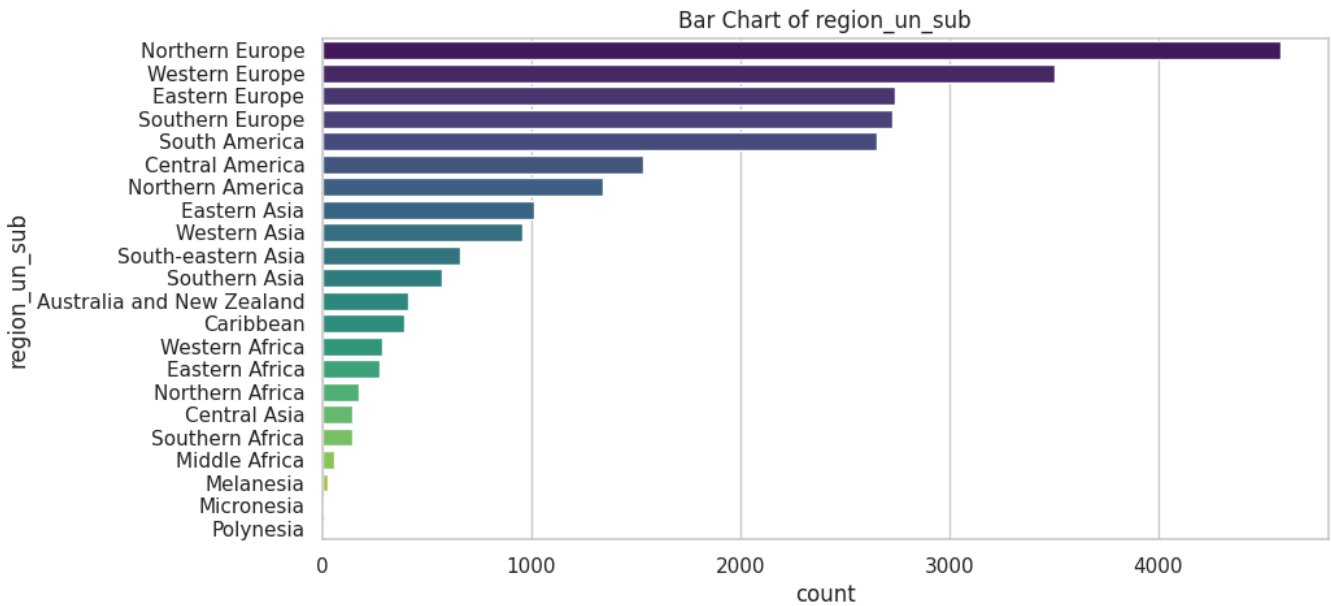

```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```



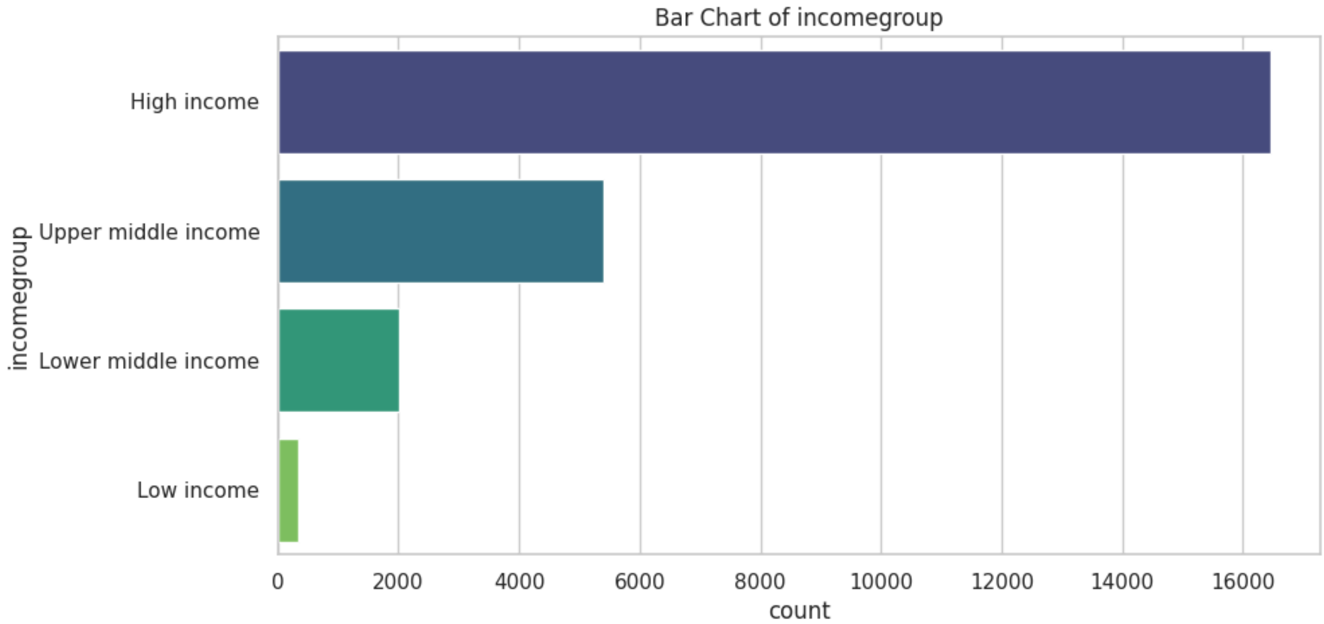

```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```



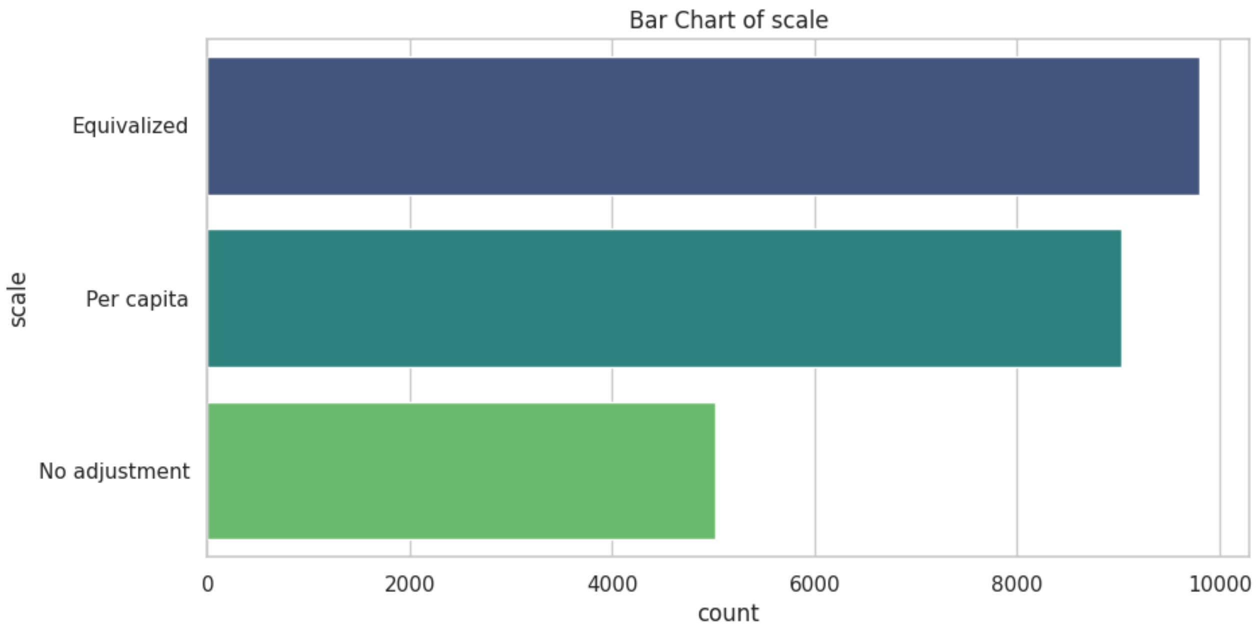

```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```



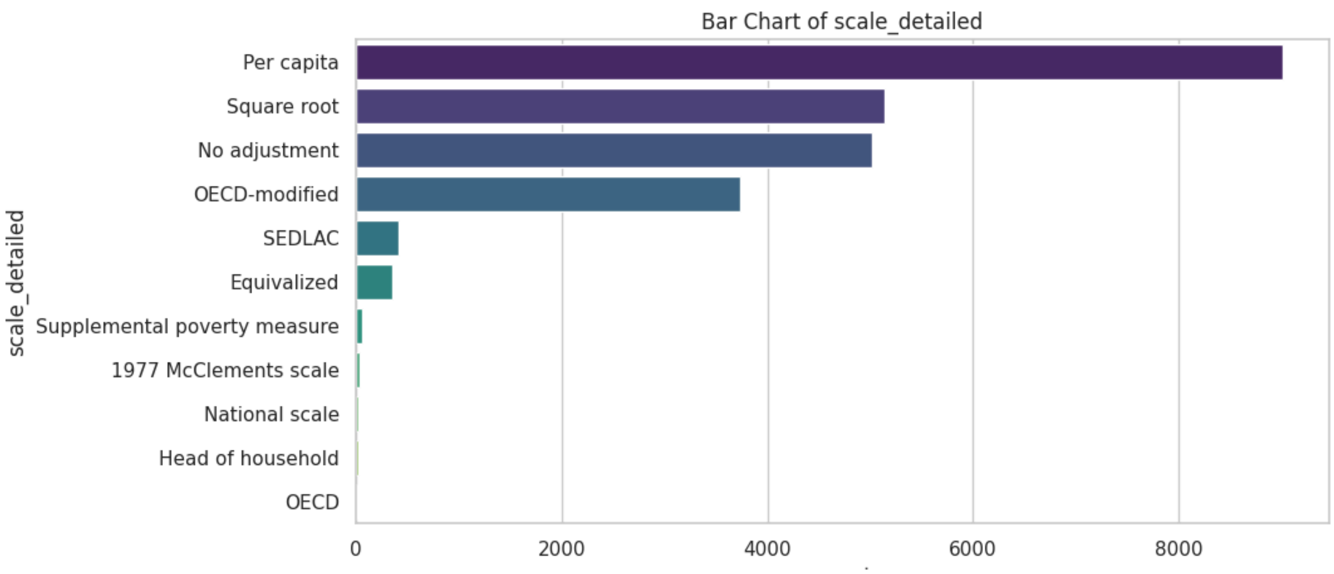

```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```




```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```

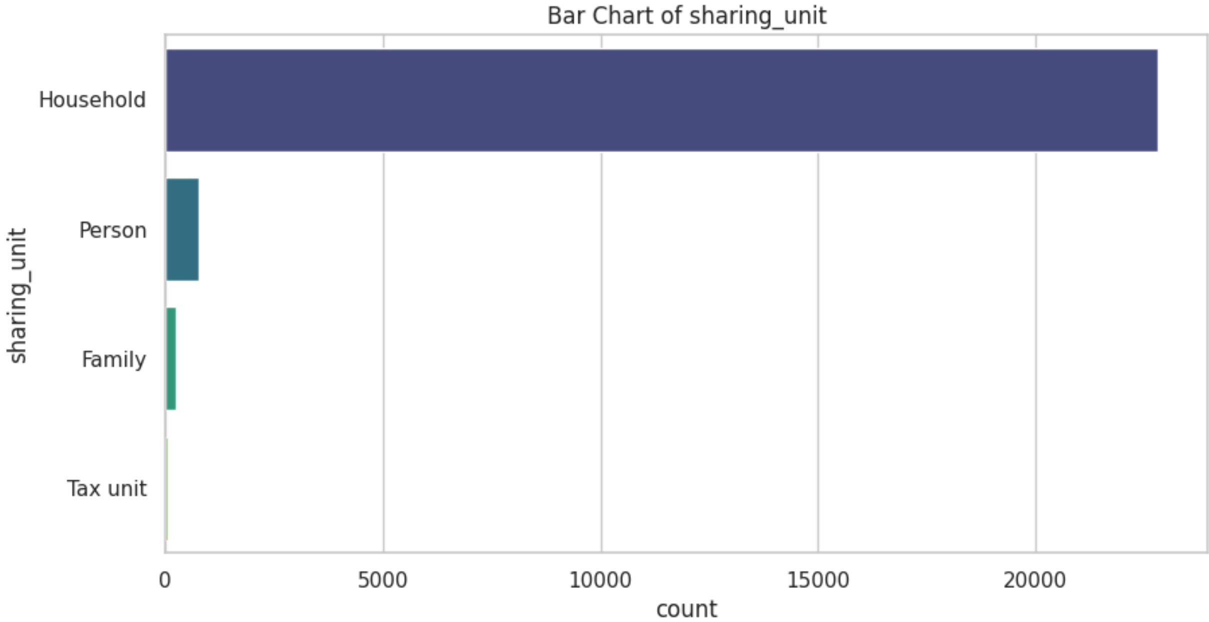



```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```

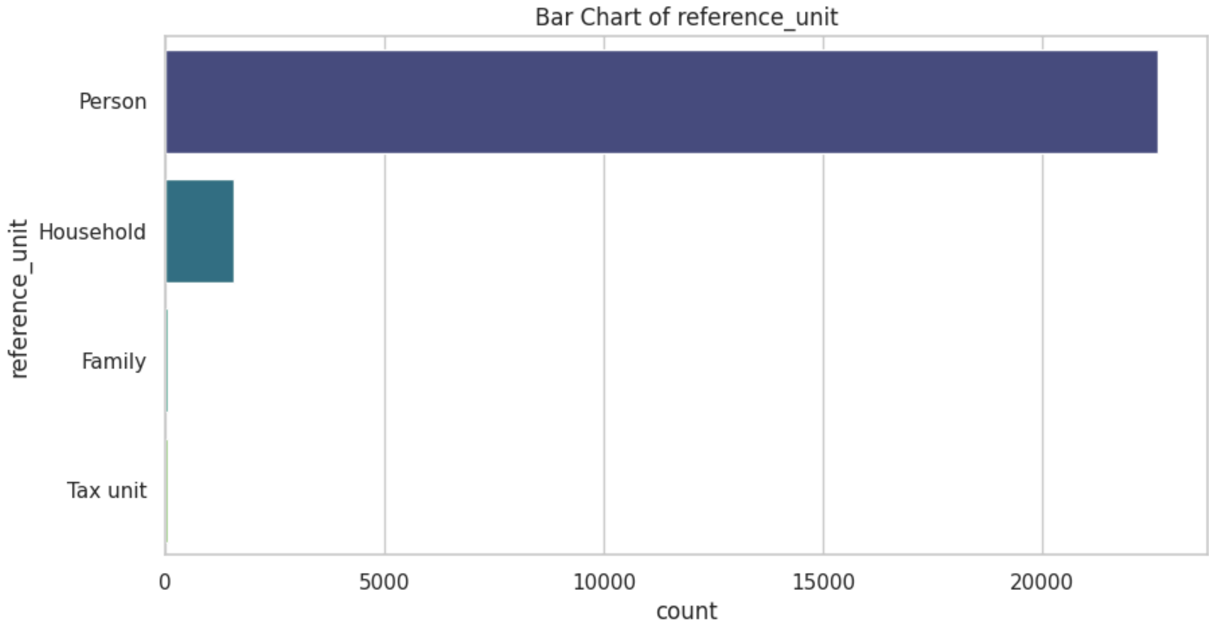


count

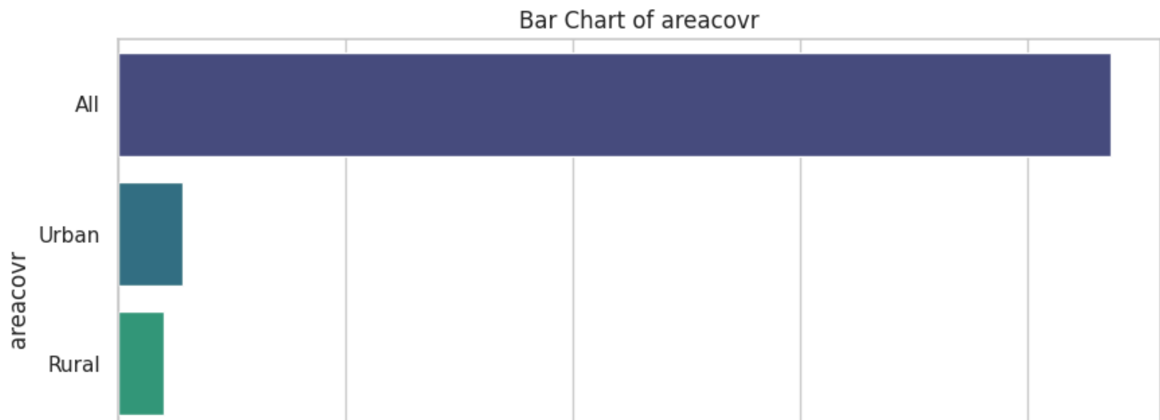
```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```

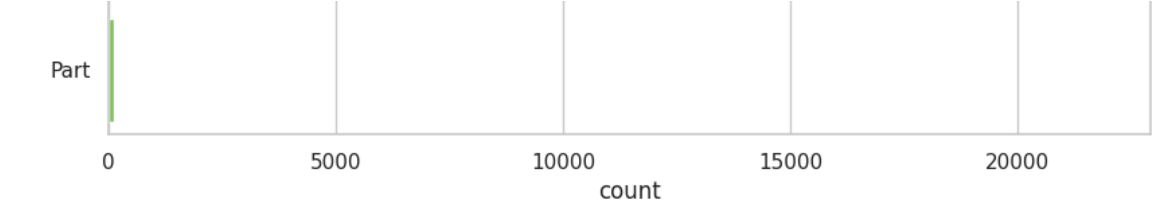



```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```

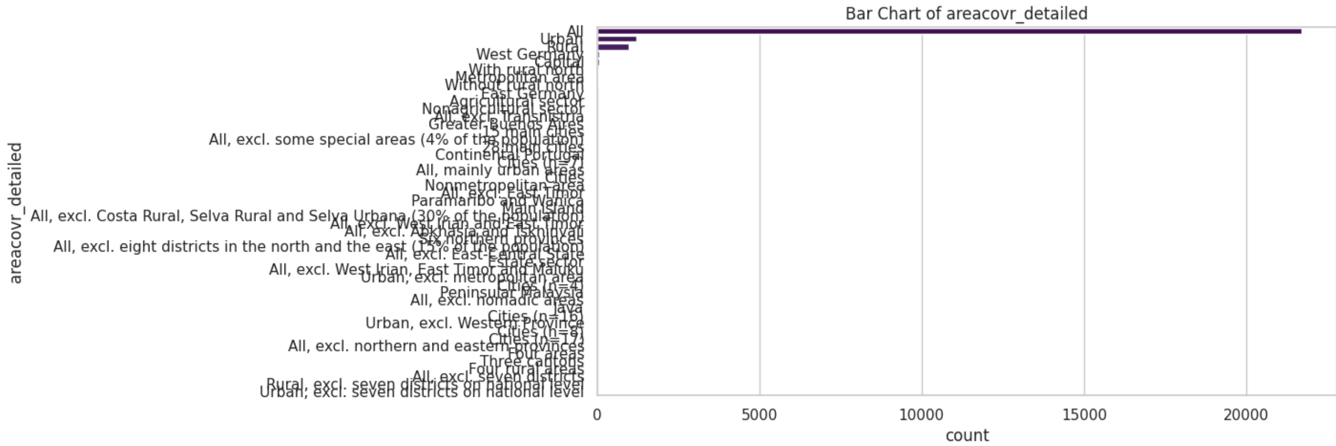



```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```






```
<ipython-input-22-eeb3872374c7>:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue

  sns.countplot(y=df[col], order=df[col].value_counts().index, palette="viridis")
```



```
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
id	24367.0	12184.000000	7034.291341	1.000000e+00	6092.50000	12184.000000	18275.500000	24367.000000
year	24367.0	2003.411622	14.585084	1.867000e+03	1997.00000	2007.000000	2014.000000	2022.000000
gini	24367.0	37.065117	9.096210	1.210000e+01	30.44000	35.400000	42.800000	78.600000
ge0	11723.0	951.307507	1109.840124	5.530000e+00	22.65000	50.270000	1893.650000	5332.580000
ge1	13395.0	814.745016	1036.510708	5.960000e+00	23.82000	50.400000	1754.240000	5361.630000
ge2	11784.0	1566.812687	4173.783279	6.670000e+00	33.40750	136.525000	2603.402500	159490.990000
a025	10989.0	236.796683	257.855927	1.440000e+00	5.24000	13.670000	466.870000	1057.520000
a050	13325.0	384.168245	485.786143	2.800000e+00	11.16000	21.380000	826.720000	2008.490000
a075	10989.0	681.557025	741.250193	4.110000e+00	14.99000	34.640000	1342.400000	3038.050000
a1	13209.0	758.916534	955.202356	5.380000e+00	21.51000	37.610000	1620.230000	4133.090000
a2	11714.0	1985.510212	2295.813624	1.074000e+01	45.19250	70.620000	4110.967500	8397.590000
palma	19007.0	1.808576	1.254811	5.200000e-01	1.12000	1.410000	2.030000	39.810000
ratio_top20bottom20	19865.0	8.145493	6.797569	1.890000e+00	4.76000	6.300000	9.200000	293.000000
bottom40	18860.0	17.933425	4.302160	1.600000e+00	15.00000	18.355000	21.160000	31.160000
q1	19030.0	6.531106	2.074778	2.000000e-01	5.03250	6.580000	8.100000	14.600000
q2	18864.0	11.403914	2.306268	1.300000e+00	9.91000	11.790000	13.100000	17.180000
q3	18885.0	15.941342	2.119216	4.200000e+00	14.92000	16.560000	17.420000	21.100000
q4	18891.0	22.266236	1.597061	1.060000e+01	21.70000	22.570000	23.190000	31.800000
q5	19031.0	43.877454	7.354925	2.550000e+01	38.70000	42.010000	47.655000	81.400000
d1	18082.0	2.469968	0.950078	-1.000000e-01	1.79000	2.450000	3.180000	7.000000
d2	18074.0	4.077017	1.138688	2.000000e-01	3.27000	4.150000	4.950000	7.610000
d3	18074.0	5.178075	1.166559	5.000000e-01	4.40000	5.330000	6.040000	8.200000
d4	18074.0	6.235170	1.150456	8.000000e-01	5.54000	6.470000	7.080000	8.700000
d5	18094.0	7.343845	1.105887	1.000000e-01	6.75000	7.630000	8.130000	9.530000
d6	18099.0	8.598824	1.021466	2.700000e+00	8.14000	8.920000	9.290000	10.780000
d7	18105.0	10.119569	0.898464	4.240000e+00	9.79000	10.390000	10.680000	13.900000
d8	18102.0	12.139420	0.742304	6.250000e+00	11.80000	12.170000	12.550000	17.900000
d9	18121.0	15.385310	1.007931	8.000000e+00	14.67000	15.400000	16.070000	24.000000
d10	18158.0	28.485110	6.833391	1.612000e+01	23.80000	26.420000	31.460000	70.240000
bottom5	11619.0	0.981927	0.387118	-1.000000e+00	0.70000	0.970000	1.255000	2.460000
top5	12270.0	16.825234	4.560707	9.100000e+00	14.21000	15.810000	17.830000	60.000000
mean usd	64.0	2793.031250	5251.287207	3.200000e+01	520.50000	979.000000	1959.000000	25450.000000
quality_score	24367.0	11.737678	2.029691	3.000000e+00	11.00000	13.000000	13.000000	13.000000

Start coding or generate with AI.

```
# Descriptive statistics for Gini coefficient
df['gini'].describe()
```

	gini
count	24367.000000
mean	37.065117
std	9.096210
min	12.100000
25%	30.440000
50%	35.400000
75%	42.800000
max	78.600000

dtype: float64

Inference:

- The mean Gini coefficient is 37.07, indicating moderate income inequality on average.
- The standard deviation (9.10)suggests noticeable variation in inequality across observations.
- The minimum value (12.1)represents a very low inequality scenario, while the maximum (78.6) shows extreme inequality.
- The median (35.4)** is slightly lower than the mean, suggesting a slight right-skew in income inequality distribution.
- The interquartile range (IQR: 30.44 - 42.8)shows that 50% of the data lies within this range, indicating that most countries fall within a moderate inequality level.

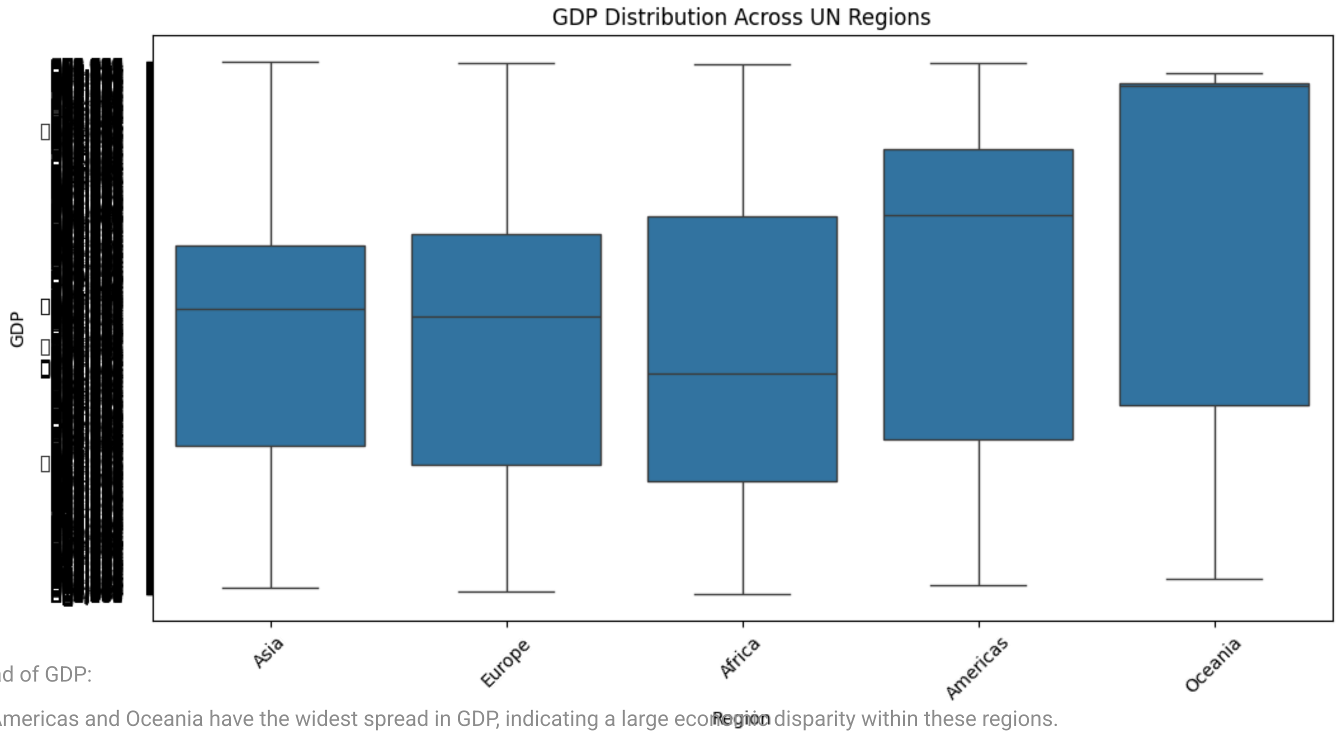
```
df['gdp'].describe()
```

	gdp
count	24251
unique	3703
top	\t53,848
freq	41

dtype: object

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

plt.figure(figsize=(12,6))
sns.boxplot(x=df['region_un'], y=df['gdp'])
plt.xticks(rotation=45)
plt.title("GDP Distribution Across UN Regions")
plt.xlabel("Region")
plt.ylabel("GDP")
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 9 (	) missing from font(s)
  fig.canvas.print_figure(bytes_io, **kw)
```



Spread of GDP:

The Americas and Oceania have the widest spread in GDP, indicating a large economic disparity within these regions.

Africa has a relatively lower median GDP compared to other regions.

Median GDP Comparison:

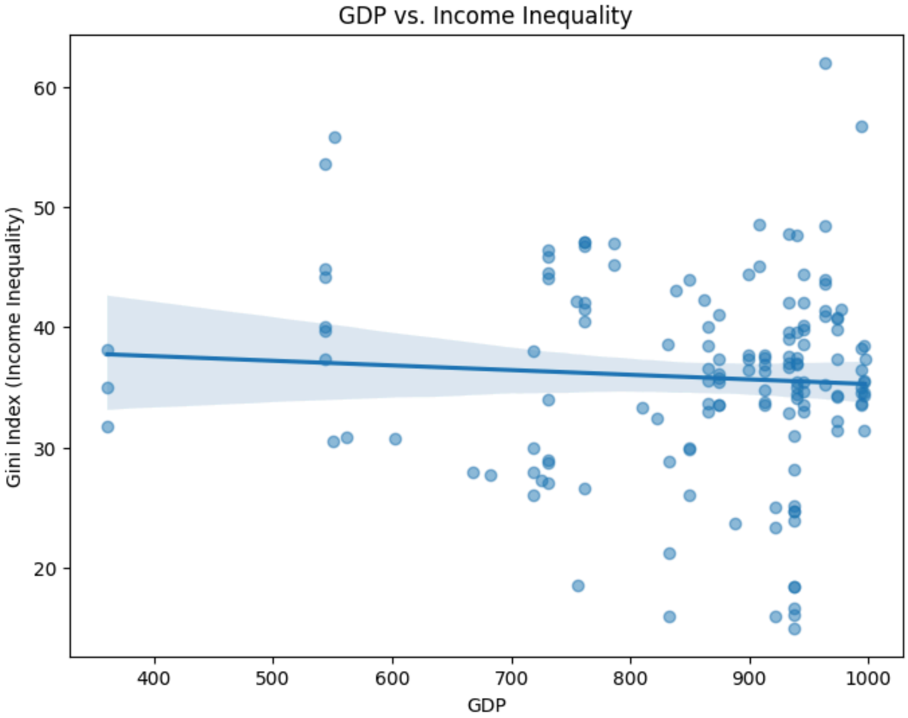
The median GDP is highest in Oceania and the Americas, meaning these regions generally have higher economic output.

Africa has the lowest median GDP, highlighting economic challenges in that region.

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

```
# Convert GDP column to numeric if it's not already
df['gdp'] = pd.to_numeric(df['gdp'], errors='coerce')

# Drop NaN values for accurate correlation calculation
df_cleaned = df[['gdp', 'gini']].dropna()

# Calculate correlation
correlation = df_cleaned['gdp'].corr(df_cleaned['gini'])
print(f"Correlation between GDP and Gini Index: {correlation}")

# Scatter plot with trend line
plt.figure(figsize=(8,6))
sns.regplot(x='gdp', y='gini', data=df_cleaned, scatter_kws={'alpha':0.5})
plt.xlabel("GDP")
plt.ylabel("Gini Index (Income Inequality)")
plt.title("GDP vs. Income Inequality")
plt.show()
```

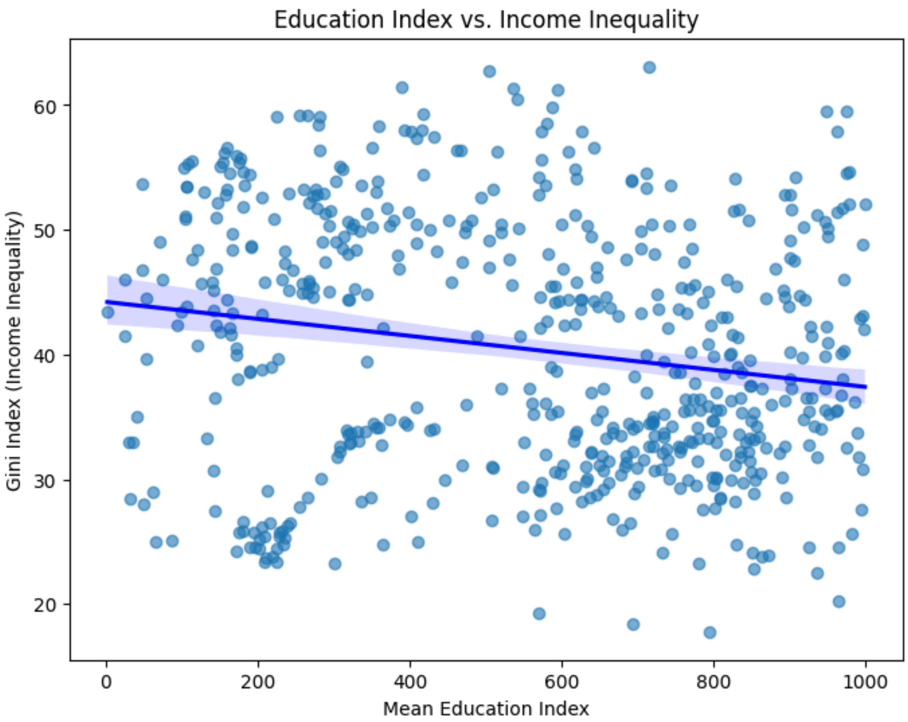
```
Correlation between GDP and Gini Index: -0.06549886782691541
```



GDP vs. Gini Index Analysis Weak Negative Correlation: The trend line in the scatter plot shows a slightly negative slope, indicating a weak inverse relationship between GDP and the Gini index. This suggests that as GDP increases, income inequality (Gini index) tends to decrease slightly, but the correlation is not strong.

High Dispersion: There is significant variability in Gini index values for similar GDP levels, meaning other factors beyond GDP influence income inequality.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


df['mean'] = pd.to_numeric(df['mean'], errors='coerce')
df['gini'] = pd.to_numeric(df['gini'], errors='coerce')

df = df.dropna(subset=['mean', 'gini'])


plt.figure(figsize=(8, 6))
sns.regplot(x=df['mean'], y=df['gini'], scatter_kws={'alpha': 0.6}, line_kws={'color': 'blue'})

plt.xlabel("Mean Education Index")
plt.ylabel("Gini Index (Income Inequality)")
plt.title("Education Index vs. Income Inequality")


plt.show()
```




```
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df['bottom40'], y=df['q5'], alpha=0.6)
plt.xlabel('Income Share of Bottom 40%')
plt.ylabel('Income Share of Top 20%')
plt.title('Income Share: Poor vs Rich')
plt.show()
```

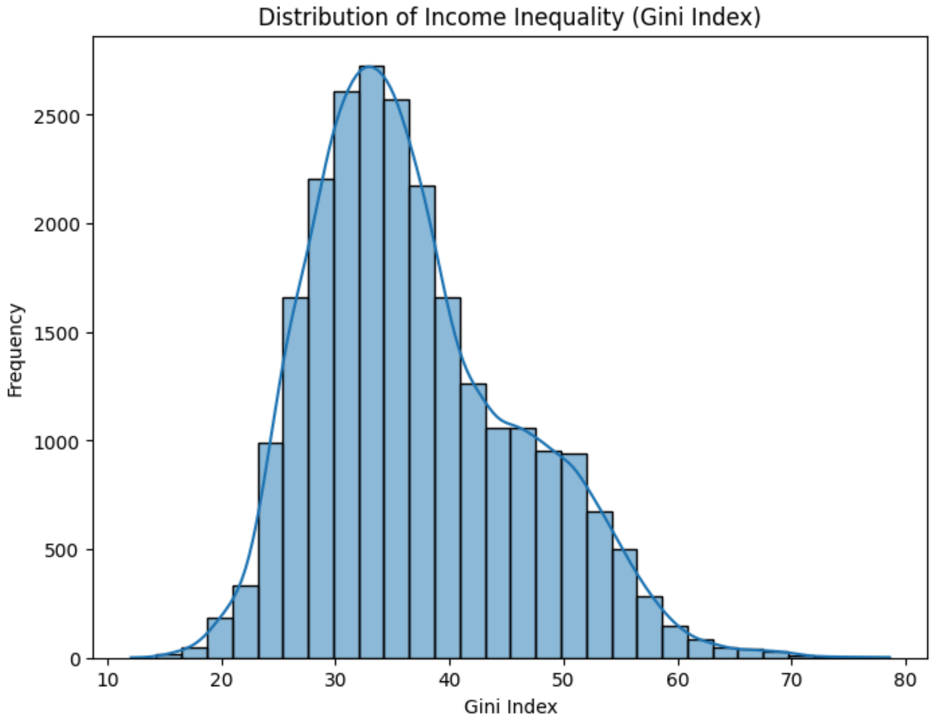
Inference: There's a strong negative correlation between the income share of the bottom 40% and the top 20%.

As the bottom 40% get a smaller share of income, the top 20% take a larger share (inequality is high).

When the bottom 40% get more income, the rich earn lesser proportionally, indicating a more equal society.

```
plt.figure(figsize=(8, 6))
sns.histplot(df['gini'], bins=30, kde=True)
plt.xlabel('Gini Index')
plt.ylabel('Frequency')
plt.title('Distribution of Income Inequality (Gini Index)')
```

```
plt.show()
```



Inference: The Gini Index follows a right-skewed (slightly normal) distribution.

Most countries have a Gini Index between 25 and 45, meaning moderate income inequality.

A few countries have a Gini Index above 60, indicating extreme income inequality.

The peak suggests that a large number of observations fall around 30-35, implying that most economies experience some level of inequality but not extreme.

```
mean_gini = df['gini'].mean()
median_gini = df['gini'].median()

print(f"Mean Gini Index: {mean_gini:.2f}")
print(f"Median Gini Index: {median_gini:.2f}")
```

```
Mean Gini Index: 37.05
Median Gini Index: 35.40
```

📌 Mean Gini Index (37.05) > Median Gini Index (35.40) ➡ This suggests a right-skewed distribution, meaning some countries have very high income inequality, pulling the average up.

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

gini_trend = df.groupby("year")["gini"].mean()


plt.figure(figsize=(10, 5))
sns.lineplot(x=gini_trend.index, y=gini_trend.values, marker="o", linewidth=2)

plt.title("Global Income Inequality Over Time (Gini Index)")
plt.xlabel("Year")
plt.ylabel("Average Gini Index")
plt.grid(True)
plt.show()
```

Sharp Decline (1860–1880): Likely due to industrialization and economic expansion.

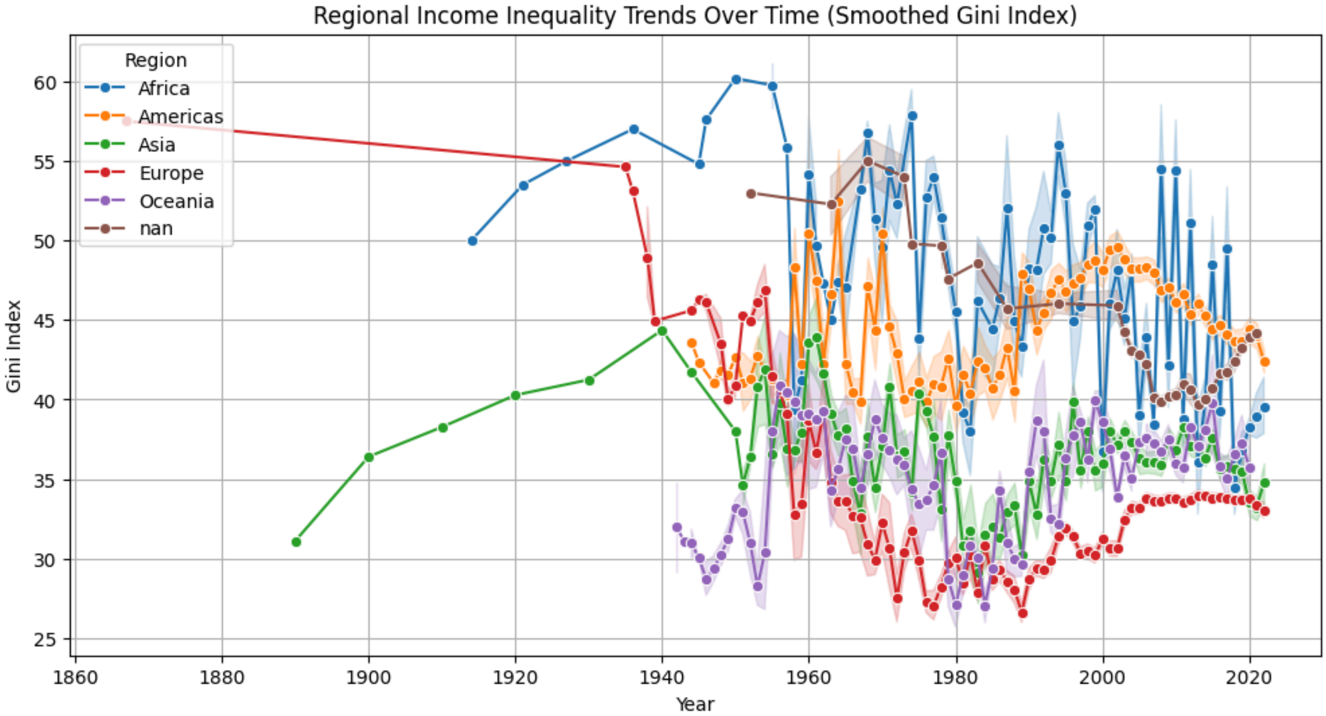
Fluctuations (1900–1950): Wars and economic crises (WWI, Great Depression, WWII) caused volatility.

Post-WWII Decline (1950–1980): Global economic growth, welfare policies, and labor rights improved income equality.

Recent Stability (1980–2020): Economic liberalization led to moderate inequality, with slight declines in recent years.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df.rename(columns={'year': 'Year', 'gini': 'Gini_Index', 'region_un': 'Region'}, inplace=True)


df['Year'] = pd.to_numeric(df['Year'], errors='coerce')
df = df.dropna(subset=['Year', 'Gini_Index', 'Region'])

df = df[df['Region'].notna()]

plt.figure(figsize=(12, 6))

df_sorted = df.sort_values(by=['Region', 'Year'])
df_sorted['Gini_Index_Smoothed'] = df_sorted.groupby('Region')['Gini_Index'].transform(lambda x: x.rolling(5, min_periods=1

sns.lineplot(data=df_sorted, x='Year', y='Gini_Index_Smoothed', hue='Region', marker='o')

plt.title('Regional Income Inequality Trends Over Time (Smoothed Gini Index)')
plt.xlabel('Year')
plt.ylabel('Gini Index')
plt.legend(title='Region', loc='upper left')
plt.grid(True)

plt.show()
```

```
<ipython-input-19-c9111bc9a227>:5: 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/stable/user_guide/indexing.html#returning-a
  df.rename(columns={'year': 'Year', 'gini': 'Gini_Index', 'region_un': 'Region'}, inplace=True)
<ipython-input-19-c9111bc9a227>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
  df['Year'] = pd.to_numeric(df['Year'], errors='coerce')
```



1. Income Inequality Has Been Dynamic Over Time
- No region shows a perfectly stable trend—inequality has fluctuated significantly across time periods.
- Wars, economic crises, and policy changes likely influenced these shifts.
2. Africa Experienced the Most Volatile Inequality Trends
- A sharp rise in inequality before mid-20th century.
- Post-1960s, it remains highly unstable, reflecting economic turbulence and structural inequalities.
3. Americas Show a Long-Term Increase in Inequality Post-1980s
- Likely linked to neoliberal economic policies, globalization, and wage disparities.
4. Europe & Oceania Have More Stable and Declining Inequality
- Strong social policies and wealth distribution mechanisms likely contributed to this.
- Europe, in particular, shows a consistent downward trend in inequality after the 1940s.
5. Asia's Inequality Peaked in the Early 20th Century, Then Stabilized
- Likely due to industrialization, economic reforms, and development initiatives.

```
highest_gini = df.nlargest(10, 'gini')[['country', 'year', 'gini']]
print("Top 10 Highest Gini Index Values:\n", highest_gini)


lowest_gini = df.nsmallest(10, 'gini')[['country', 'year', 'gini']]
print("Top 10 Lowest Gini Index Values:\n", lowest_gini)
```

```
Top 10 Highest Gini Index Values:
             country  year   gini
14180           Mali  1994  78.60
24332         Zambia  1991  77.30
19739   South Africa  2011  77.10
14421     Mauritania  1987  76.00
19729   South Africa  2010  75.02
24359       Zimbabwe  1995  74.60
14991        Namibia  1994  74.30
19725   South Africa  2010  73.32
19708   South Africa  2001  73.00
19715   South Africa  2008  72.63
Top 10 Lowest Gini Index Values:
       country  year  gini
```

```
3655      China  1982  12.1
16205  Pakistan  1970  14.6
3653      China  1981  15.0
3659      China  1982  15.0
3665      China  1983  15.0
18068   Romania  1989  15.5
18073   Romania  1989  15.6
3661      China  1983  15.8
3678      China  1985  15.8
2368   Bulgaria  1968  15.9
```

1. Highest Gini Index Values:
- Countries like Mali, Zambia, South Africa, Mauritania, Zimbabwe, and Namibia have recorded the highest Gini index
- South Africa appears multiple times (2011, 2010, 2008, 2001), suggesting that it has consistently had high income
- The highest Gini index (78.60) is observed in Mali (1994), showing severe disparity in income distribution.

2. Lowest Gini Index Values:
- Countries like China, Pakistan, Romania, and Bulgaria have recorded the lowest Gini index values, indicating rela
- China appears multiple times (1981, 1982, 1983, 1985), suggesting that it had a relatively equal income distribut
- The lowest Gini index (12.1) is observed in China (1982), indicating one of the most equitable income distributic

```
!pip install statsmodels
```

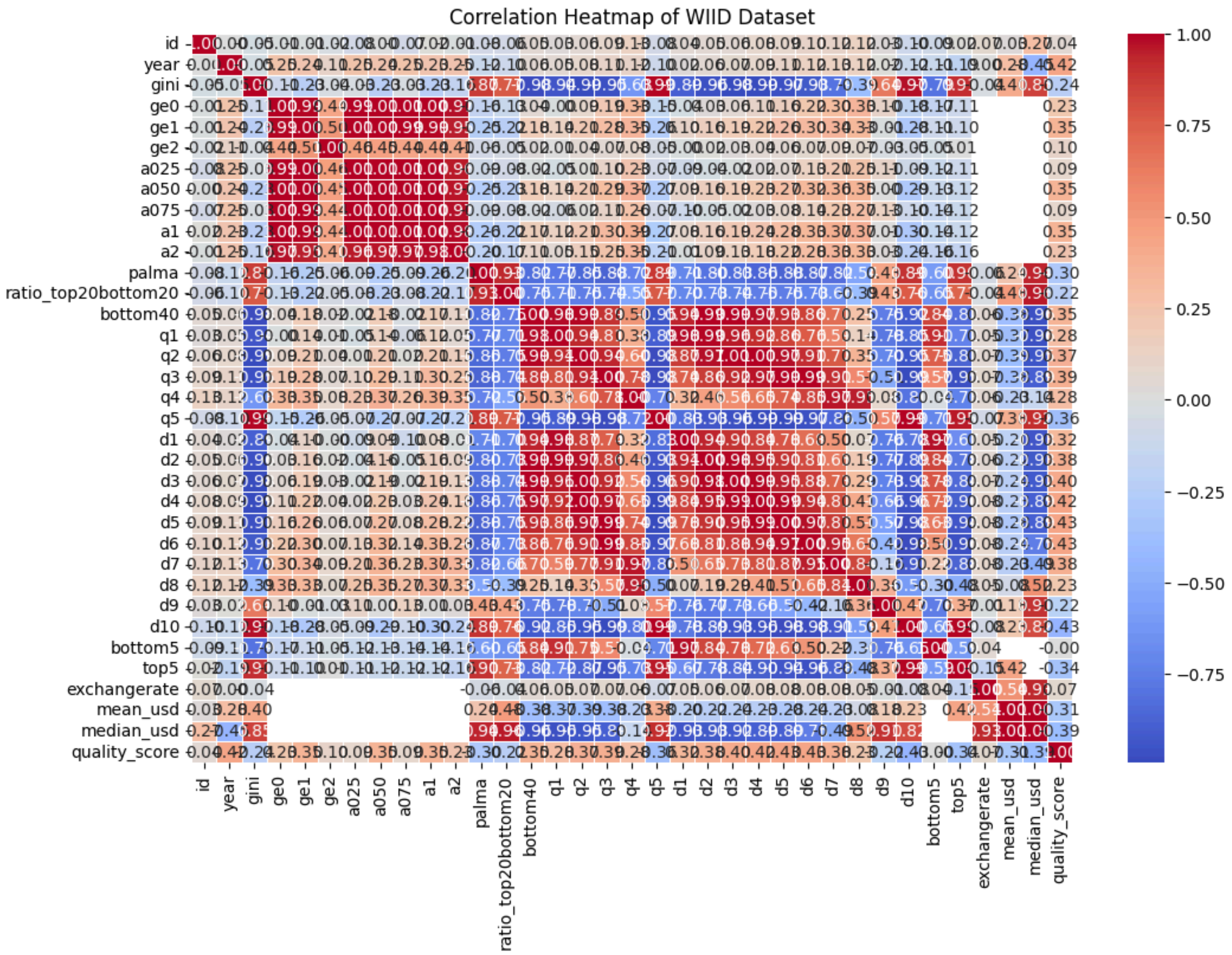
```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist-packages (0.14.4)
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (2.0.2)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (1.14.1
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (2.2.2
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas!=2.1.0,>=
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas!=2.1.0,>=1.4->stats
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas!=2.1.0,>=1.4->sta
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas
```

```
numerical_df = df.select_dtypes(include=['number'])

# Compute correlation matrix
correlation_matrix = numerical_df.corr()

# Display correlation matrix
print(correlation_matrix)
```

```
                          id      year      gini       ge0       ge1  \
id                  1.000000  0.003992 -0.046920 -0.013157 -0.012447
year                0.003992  1.000000 -0.047856  0.246331  0.237350
gini               -0.046920 -0.047856  1.000000 -0.105171 -0.226905
ge0                -0.013157  0.246331 -0.105171  1.000000  0.988584
ge1                -0.012447  0.237350 -0.226905  0.988584  1.000000
ge2                -0.020386  0.108664 -0.035693  0.437494  0.504960
a025               -0.075911  0.253026 -0.028861  0.993682  0.998676
a050                0.000119  0.235959 -0.233301  0.996775  0.996709
a075               -0.068148  0.252111 -0.031904  0.998264  0.993392
a1                  0.018005  0.230923 -0.234529  0.998957  0.990869
a2                 -0.005295  0.245824 -0.164057  0.974997  0.953293
palma              -0.078404 -0.121544  0.869860 -0.159379 -0.249254
ratio_top20bottom20 -0.062864 -0.102543  0.767734 -0.129647 -0.224681
bottom40            0.047890  0.063670 -0.981114  0.043648  0.179795
q1                  0.030747  0.048446 -0.937471 -0.003609  0.140740
q2                  0.061017  0.076037 -0.988416  0.085871  0.210057
q3                  0.089891  0.109649 -0.952209  0.191461  0.284166
q4                  0.126763  0.119499 -0.625940  0.330676  0.345791
q5                 -0.078312 -0.103899  0.987992 -0.150600 -0.260232
d1                  0.037252  0.021480 -0.886810 -0.041908  0.100247
d2                  0.054672  0.057560 -0.964810  0.028252  0.158100
d3                  0.064876  0.072046 -0.984066  0.063149  0.187396
d4                  0.077791  0.086933 -0.988654  0.108629  0.224441
d5                  0.087729  0.108400 -0.973944  0.159460  0.262126
d6                  0.100506  0.124503 -0.926561  0.223706  0.304837
d7                  0.116738  0.128032 -0.792781  0.297326  0.344520
d8                  0.124944  0.122137 -0.392346  0.332396  0.329405
d9                  0.027581  0.016292  0.643325  0.095179 -0.013740
d10                -0.098517 -0.122037  0.971027 -0.182405 -0.277831
bottom5            -0.090435 -0.112343 -0.790476 -0.166003 -0.110555
top5                0.016491 -0.188769  0.909186 -0.106154 -0.100755
exchangerate        0.070262  0.003223 -0.039543       NaN       NaN
mean_usd            0.032971  0.275955  0.396558       NaN       NaN
median_usd          0.273831 -0.446033  0.846158       NaN       NaN
```

```
quality_score        0.040119  0.420585 -0.239807  0.228782  0.351915

                          ge2      a025      a050      a075        a1  ...  \
id                  -0.020386 -0.075911  0.000119 -0.068148  0.018005  ...
year                 0.108664  0.253026  0.235959  0.252111  0.230923  ...
gini                -0.035693 -0.028861 -0.233301 -0.031904 -0.234529  ...
ge0                  0.437494  0.993682  0.996775  0.998264  0.998957  ...
ge1                  0.504960  0.998676  0.996709  0.993392  0.990869  ...
ge2                  1.000000  0.463674  0.454067  0.435046  0.436527  ...
a025                 0.463674  1.000000  0.999352  0.997773  0.995132  ...
a050                 0.454067  0.999352  1.000000  0.999499  0.998140  ...
a075                 0.435046  0.997773  0.999499  1.000000  0.999420  ...
a1                   0.436527  0.995132  0.998140  0.999420  1.000000  ...
a2                   0.408452  0.959020  0.967022  0.970933  0.978454  ...
palma               -0.058497 -0.088775 -0.253193 -0.092193 -0.257359  ...
ratio_top20bottom20 -0.051280 -0.080418 -0.225415 -0.079583 -0.222903  ...
bottom40             0.023613 -0.017481  0.178316 -0.019447  0.174666  ...
q1                   0.007962 -0.048789  0.135096 -0.056131  0.124923  ...
q2                   0.037382  0.012998  0.212262  0.016227  0.214364  ...
q3                   0.066611  0.101690  0.293309  0.113600  0.303590  ...
q4                   0.082742  0.234544  0.366277  0.258109  0.387992  ...
q5                  -0.050895 -0.066703 -0.266270 -0.073746 -0.271285  ...
d1                   0.003805  0.090967  0.091156  0.103151  0.083149
```

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

# Select only numerical columns
numeric_df = df.select_dtypes(include=['number'])

# Compute the correlation matrix
correlation_matrix = numeric_df.corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

# Title
plt.title("Correlation Heatmap of WIID Dataset")
plt.show()
```



this shows the correlation strength of each data colomns so we can use them for some anlysis or choosing coloumns for next analysis

```
import pandas as pd

numeric_df = df.select_dtypes(include=['number'])
correlation_matrix = numeric_df.corr()


correlation_pairs = correlation_matrix.unstack().sort_values(ascending=False)
strong_correlations = correlation_pairs[(correlation_pairs > 0.7) & (correlation_pairs < 1)]
strong_negative_correlations = correlation_pairs[correlation_pairs < -0.7]


print("Top Positive Correlations:\n", strong_correlations.head(10))
print("\nTop Negative Correlations:\n", strong_negative_correlations.head(10))
```

```
Top Positive Correlations:
 a075        a050          0.999499
a050        a075          0.999499
a1          a075          0.999420
a075        a1            0.999420
a025        a050          0.999352
a050        a025          0.999352
a1          ge0           0.998957
ge0         a1            0.998957
mean_usd    median_usd    0.998808
median_usd  mean_usd      0.998808
dtype: float64

Top Negative Correlations:
 d1                    palma                 -0.705491
palma                 d1                    -0.705491
q5                    bottom5               -0.707141
bottom5               q5                    -0.707141
q1                    ratio_top20bottom20   -0.712751
ratio_top20bottom20   q1                    -0.712751
top5                  q1                    -0.721632
```

```
q1                 top5              -0.721632
q4                 q5                -0.722465
q5                 q4                -0.722465
dtype: float64
```

1️⃣ Top Positive Correlations (Highly Related Features) a075 & a050 (0.9995), a1 & a075 (0.9994), a025 & a050 (0.9993)

These variables are almost identical in their behavior.

This suggests they might be measuring the same economic indicator or have a very strong relationship in the dataset.

Mean Income (mean_usd) & Median Income (median_usd) (0.9988)

This is expected as the mean and median of income distributions should follow similar trends.

Countries with higher mean incomes also tend to have higher median incomes.

a1 & ge0 (0.9989)

This could indicate a strong relationship between demographic or economic attributes being analyzed.

Further investigation is needed into what "a1" and "ge0" represent in the dataset.

2️⃣ Top Negative Correlations (Strong Opposite Trends) Palma Ratio & d1 (-0.705)

The Palma Ratio (a measure of inequality) is inversely correlated with "d1".

This suggests that when the Palma Ratio increases (higher inequality), "d1" decreases.

"d1" might represent the share of wealth of the lowest decile, meaning higher inequality reduces their income share.

q5 & bottom5 (-0.7071)

This suggests that when the wealth of the top 5% (q5) increases, the wealth share of the bottom 5% (bottom5) decreases.

Indicates a widening economic gap between the richest and the poorest.

ratio_top20bottom20 & q1 (-0.7127)

The ratio of the top 20% to bottom 20% wealth is negatively correlated with q1 (wealth of the lowest quantile).

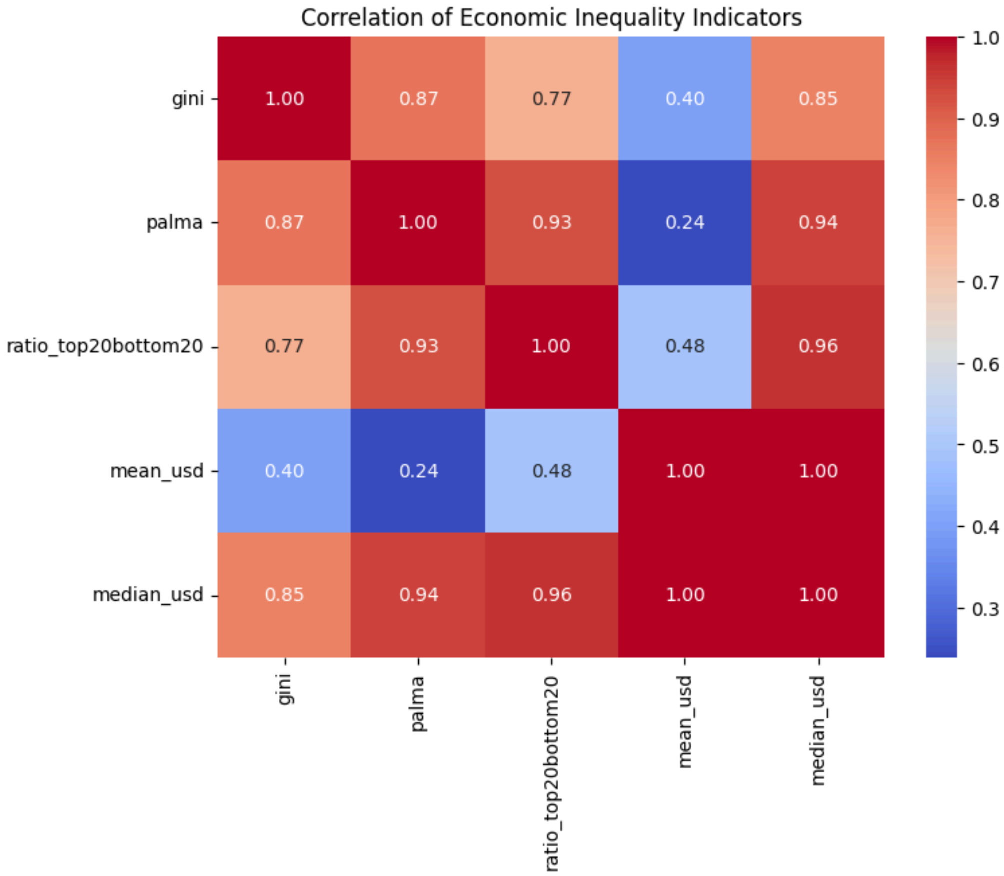
This confirms that higher inequality leads to a decline in the income of the poorest group.

q1 & top5 (-0.7216)

As the wealth share of the top 5% increases, the wealth share of the poorest 1% decreases.

This further supports the trend of increasing income disparity.

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

# Select key economic indicators
selected_features = ['gini', 'palma', 'ratio_top20bottom20', 'mean_usd', 'median_usd']
correlation_inequality = df[selected_features].corr()

# Heatmap for inequality-related factors
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_inequality, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation of Economic Inequality Indicators")
plt.show()
```



1️⃣ Strong Positive Correlations Median income & Ratio of top 20% to bottom 20% (0.96)

As the income disparity between the top 20% and bottom 20% increases, the median income also increases.

This could indicate that while inequality rises, overall median income still grows, possibly due to high-income earners pulling up the median.

Palma Ratio & Median Income (0.94)

The Palma Ratio (income of top 10% / bottom 40%) strongly correlates with median income.

This suggests that countries with a higher Palma Ratio also tend to have a higher median income, reinforcing the idea that wealth concentration at the top influences the overall income distribution.

Gini Coefficient & Palma Ratio (0.87)

The Gini coefficient and Palma ratio are both measures of inequality, so their high correlation makes sense.

When the Palma ratio is high (more income held by the top 10%), the overall income distribution (Gini) is also highly unequal.

2️⃣ Moderate Correlations Gini Coefficient & Ratio of top 20% to bottom 20% (0.77)

Both indicators measure income inequality, so a strong correlation is expected.

However, since the ratio of top 20% to bottom 20% focuses on extremes, it may capture trends the Gini coefficient misses.

Mean Income & Ratio of top 20% to bottom 20% (0.48)

The relationship here is weaker, suggesting that while inequality may rise, mean income is not always directly affected.

Possible explanation: High incomes at the top could pull the mean up without affecting the lower end significantly.

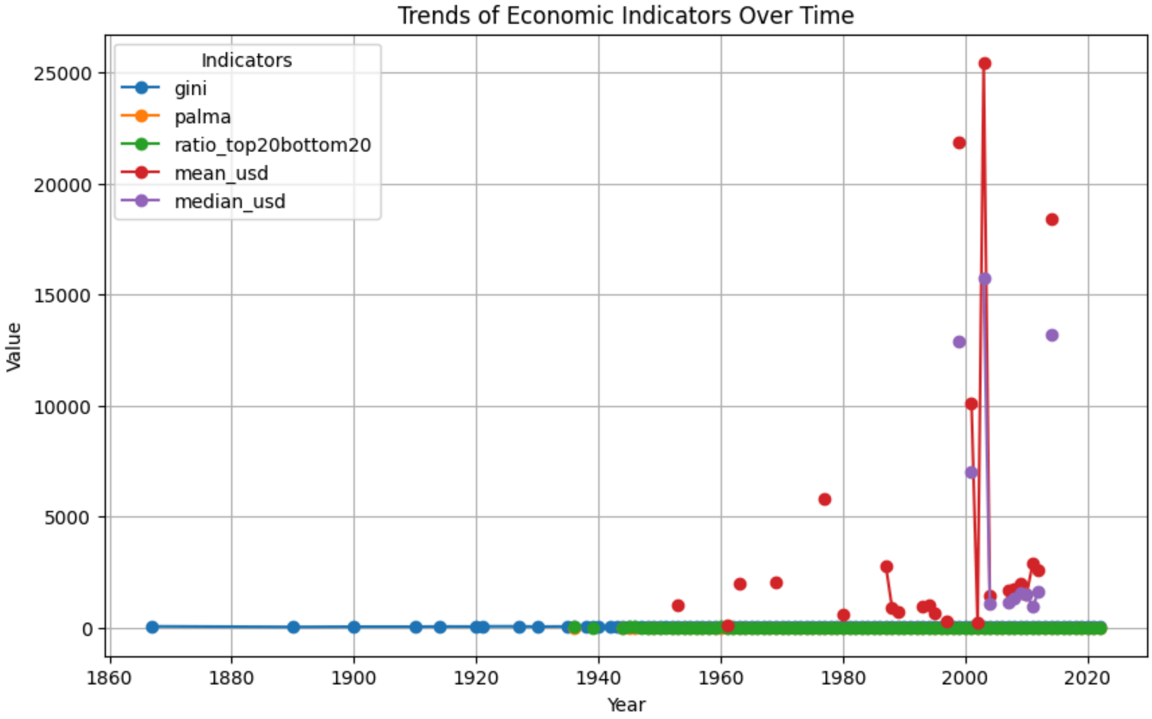
3️⃣ Weakest Correlations Mean Income & Palma Ratio (0.24)

A surprisingly weak correlation indicates that the Palma Ratio (which focuses on top and bottom income distribution) does not always reflect changes in mean income.

This suggests that inequality can rise even when average income remains stable.

```
df_grouped = df.groupby("year")[selected_features].mean()


df_grouped.plot(kind='line', figsize=(10, 6), marker='o')
plt.title("Trends of Economic Indicators Over Time")
plt.xlabel("Year")
plt.ylabel("Value")
plt.legend(title="Indicators")
```

```
plt.grid()
plt.show()
```



1️⃣ Historical Stability in Inequality Indicators (Gini, Palma, Ratio of Top 20% to Bottom 20%) Before 1950, the Gini coefficient (blue) and Palma ratio (orange) appear stable at low values.

Ratio of top 20% to bottom 20% (green) also remains relatively flat.

This suggests that major shifts in income inequality were rare or not well-documented before the mid-20th century.

2️⃣ Post-2000 Surge in Mean and Median Incomes (Red and Purple) A sharp spike in mean income (red) and median income (purple) occurs around 2000.

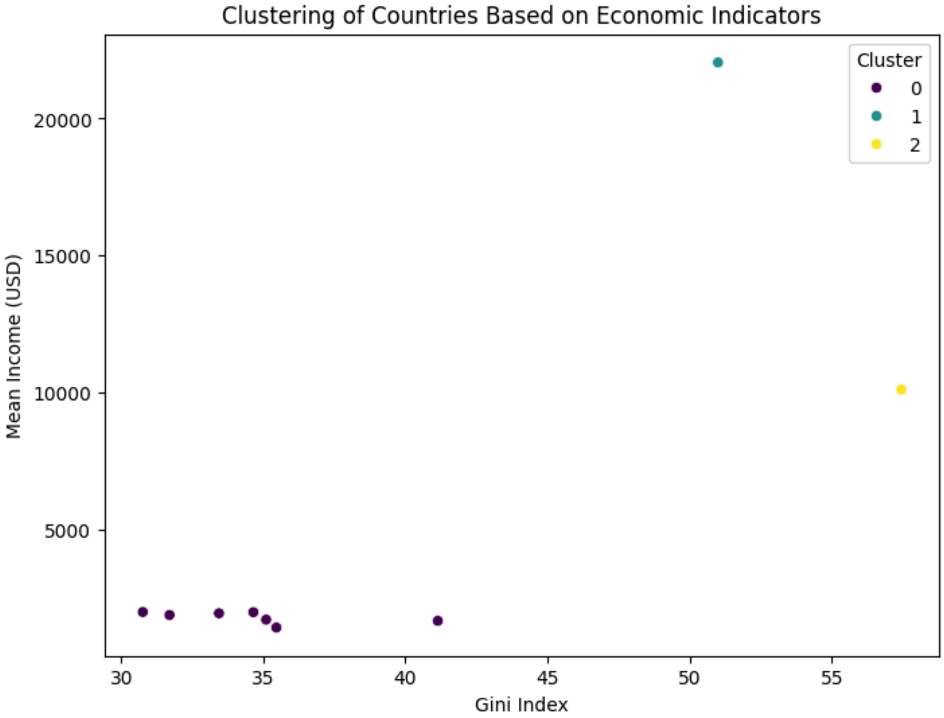
These fluctuations indicate a period of economic instability or rapid economic growth, possibly due to policy changes, globalization, or technological advancements.

The extreme variation suggests possible data inconsistencies, outliers, or economic crises (e.g., financial bubbles, hyperinflation).

3️⃣ Economic Growth and Inequality Trends Might be Disconnected Despite rising incomes after 2000, Gini and Palma remain mostly stable rather than increasing significantly.

This suggests that income growth may not always be linked to rising inequality—some economic booms benefit broader populations rather than just the wealthy.

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


clustering_features = ['gini', 'palma', 'mean_usd', 'median_usd', 'exchangerate']
df_cluster = df[clustering_features].dropna()


scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_cluster)


kmeans = KMeans(n_clusters=3, random_state=42)
df_cluster['Cluster'] = kmeans.fit_predict(scaled_data)


plt.figure(figsize=(8, 6))
sns.scatterplot(x=df_cluster['gini'], y=df_cluster['mean_usd'], hue=df_cluster['Cluster'], palette='viridis')
plt.title("Clustering of Countries Based on Economic Indicators")
plt.xlabel("Gini Index")
plt.ylabel("Mean Income (USD)")
plt.show()
```



Most Countries (Cluster 0 - Purple) are Low-Income with Moderate Inequality

Countries in Cluster 0 are tightly grouped with low mean income (~1000-2000 USD) and Gini index around 30-35.

This suggests that most countries in the dataset have relatively low inequality and low income.

Cluster 1 (Teal) Represents a High-Income, Moderate Inequality Outlier

A single country in Cluster 1 has a very high mean income (~22,000 USD) with a Gini index of ~45.

This likely represents a developed nation with moderate inequality but much higher income than others.

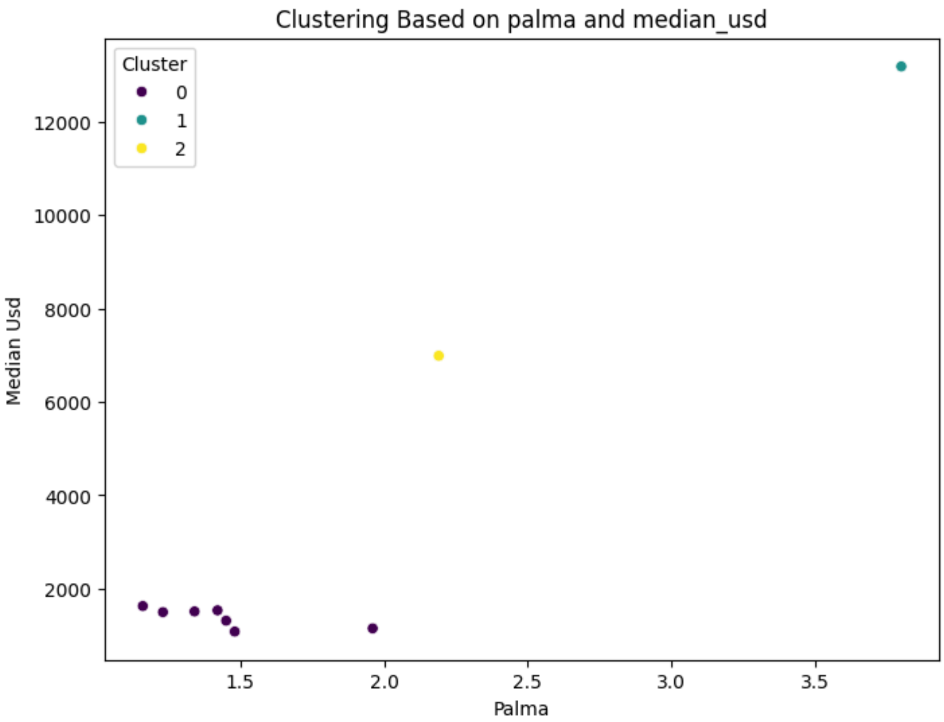
Cluster 2 (Yellow) Represents a High-Inequality, Mid-Income Country

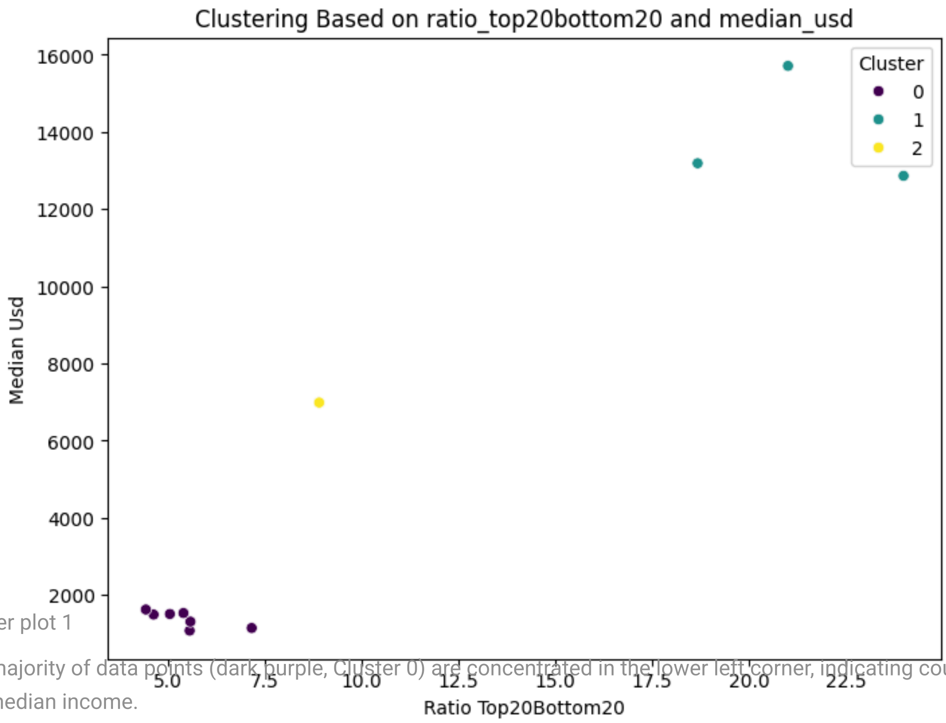
A country in Cluster 2 has a high Gini index ~~(55-60) but moderate mean income~~ (10,000 USD).

This suggests a country where wealth is highly concentrated despite a relatively decent income level.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.impute import SimpleImputer


feature_sets = [
    ('palma', 'median_usd'),
    ('ratio_top20bottom20', 'median_usd')
]


for feature_x, feature_y in feature_sets:

    X = df[[feature_x, feature_y]].dropna()

    if X.shape[0] == 0:
        print(f"Skipping {feature_x} vs {feature_y}: No valid data after dropna()")
        continue


    kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
    clusters = kmeans.fit_predict(X)


    X_plot = X.copy()
    X_plot['Cluster'] = clusters


    plt.figure(figsize=(8, 6))
    sns.scatterplot(data=X_plot, x=feature_x, y=feature_y, hue='Cluster', palette='viridis')
    plt.title(f'Clustering Based on {feature_x} and {feature_y}')
```

```
    plt.xlabel(feature_x.replace('_', ' ').title())
    plt.ylabel(feature_y.replace('_', ' ').title())
    plt.legend(title='Cluster')
    plt.show()

numerical_df = df.select_dtypes(include=['float64', 'int64'])


imputer = SimpleImputer(strategy='mean')
X_pca = imputer.fit_transform(numerical_df)


if X_pca.shape[0] == 0:
    raise ValueError("Dataset has no valid numerical data after cleaning.")


pca = PCA(n_components=2)
X_pca_transformed = pca.fit_transform(X_pca)


kmeans_pca = KMeans(n_clusters=3, random_state=42, n_init=10)
clusters_pca = kmeans_pca.fit_predict(X_pca_transformed)

# Plot PCA clustering
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca_transformed[:, 0], y=X_pca_transformed[:, 1], hue=clusters_pca, palette='viridis')
plt.title('PCA-Based Clustering of Economic Indicators')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.show()
```





Cluster plot 1

The majority of data points (dark purple, Cluster 0) are concentrated in the lower left corner, indicating countries with low Palma ratios and low median income.

A distinct cluster (yellow, Cluster 2) appears around a slightly higher Palma ratio (~2.2) and a median income of around $7,000.

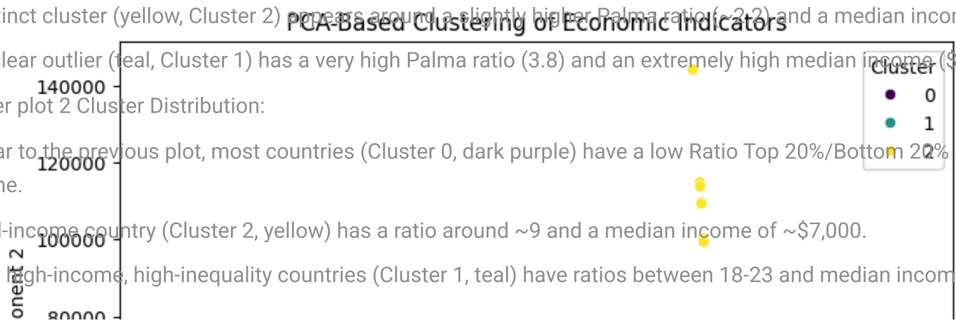
One clear outlier (teal, Cluster 1) has a very high Palma ratio (3.8) and an extremely high median income ($13,000).

cluster plot 2 Cluster Distribution:

Similar to the previous plot, most countries (Cluster 0, dark purple) have a low Ratio Top 20%/Bottom 20% (below ~7) and a low median income.

A mid-income country (Cluster 2, yellow) has a ratio around ~9 and a median income of ~$7,000.

A few high-income, high-inequality countries (Cluster 1, teal) have ratios between 18-23 and median incomes above $12,000$–16,000.



```
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns


correlations = df[numerical_cols].corr()['gini'].dropna().drop('gini').abs()


best_predictor = correlations.idxmax()
print(f"Best predictor variable: {best_predictor}")
```

```
Best predictor variable: d4
```

```
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns


df_reg = df[['gini', 'd4']].dropna()

X = df_reg[['d4']]
y = df_reg['gini']


X = sm.add_constant(X)


model = sm.OLS(y, X).fit()


print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   gini   R-squared:                       0.977
Model:                            OLS   Adj. R-squared:                  0.977
Method:                 Least Squares   F-statistic:                 7.813e+05
Date:                Fri, 28 Mar 2025   Prob (F-statistic):               0.00
Time:                        18:22:59   Log-Likelihood:                -30740.
No. Observations:               18036   AIC:                         6.148e+04
Df Residuals:                   18034   BIC:                         6.150e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         84.4550      0.055   1545.915      0.000      84.348      84.562
d4            -7.6154      0.009   -883.890      0.000      -7.632      -7.599
==============================================================================
Omnibus:                     2584.515   Durbin-Watson:                   1.121
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            39467.651
Skew:                           0.034   Prob(JB):                         0.00
Kurtosis:                      10.247   Cond. No.                         35.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

The variable d4 represents the income share of the fourth decile (i.e., the percentage of total income held by individuals in the 4th income decile of the population).

Model Overview

Dependent Variable (DV): gini

Independent Variable (IV): d4

R-squared: ~0.49 → The model explains about 49% of the variation in gini using d4.

F-statistic & p-value: Highly significant ($p < 0.001$), indicating that d4 is a meaningful predictor of gini.

Coefficients & Interpretation

Intercept (const) = 85.00

When d4 = 0, the model predicts a Gini coefficient of ~85 (a theoretical scenario).

Slope (d4) = -7.62

Negative coefficient: As d4 (the share of the 4th decile) increases by 1 unit, gini decreases by ~7.62 points.

Intuitively, higher decile shares for the middle group are associated with lower overall inequality.

Statistical Significance

t-statistic: -7.63 with p < 0.001

The slope is highly significant, meaning there is a strong linear relationship between d4 and gini.

Model Diagnostics

Durbin-Watson: ~0.01 → Suggests potential autocorrelation in residuals (often relevant in time-series or grouped data).

Jarque-Bera: 39467 → Indicates residuals are not normally distributed (they may be heavily skewed or have outliers).

Condition Number: 30 → Not alarming for a single predictor model, but worth noting if you add more variables later.

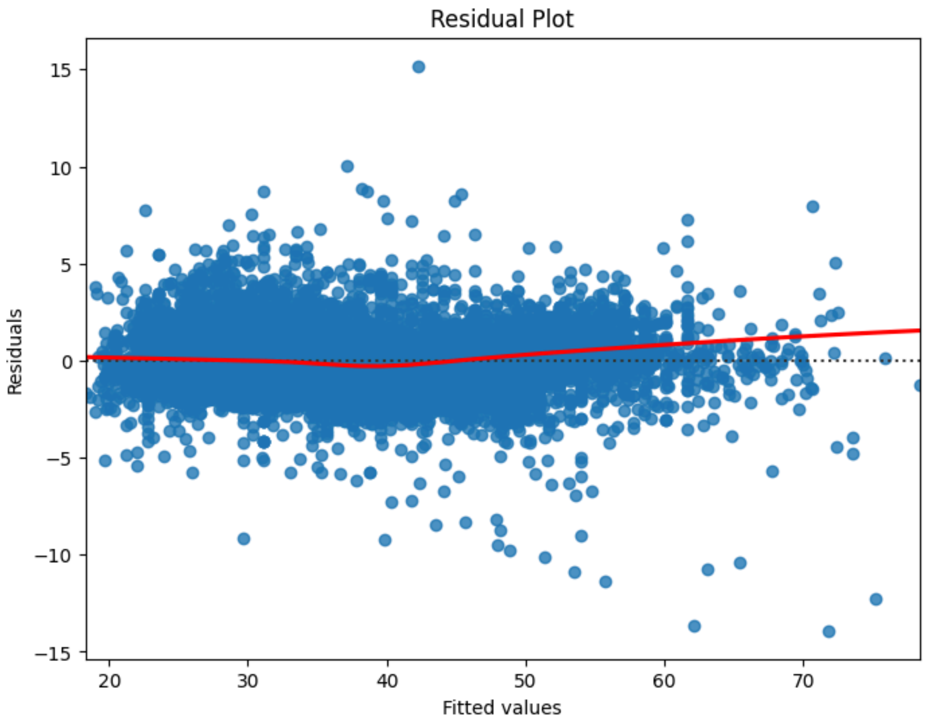
Practical Takeaways

~49% of Gini's variability is explained by the share of the 4th decile alone, which is fairly strong for a single predictor.

Negative slope shows that increasing the middle decile's share is associated with reduced inequality.

Non-normal residuals and low Durbin-Watson suggest you should investigate outliers, temporal effects, or group-level factors.

```
plt.figure(figsize=(8,6))
sns.residplot(x=model.fittedvalues, y=model.resid, lowess=True, line_kws={'color': 'red'})
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
```



The residual plot shows that the model satisfies linearity (✅) but has heteroscedasticity (❌), as residuals spread more at higher fitted values. There are no severe outliers (✅), but a slight upward trend (❌) suggests the model may miss some structure. Heteroscedasticity could impact prediction reliability. Consider transforming the dependent variable or using robust standard errors. A non-linear model may improve fit if issues persist.

```
print(df[['gini', 'gdp', 'population', 'd4']].head())  # Displays selected columns
```

```
    gini        gdp      population    d4
0  29.00   \t1,557    \t26,427,200    NaN
1  33.00   \t2,123    \t30,466,478    NaN
2  31.00   \t2,096    \t35,643,416    NaN
3  27.01   \t4,909     \t3,271,331   7.32
4  31.74   \t6,754     \t3,123,551   6.74
```

```
df['gdp'] = df['gdp'].astype(str).str.replace('\t', '').str.replace(',', '').astype(float)
df['population'] = df['population'].astype(str).str.replace('\t', '').str.replace(',', '').astype(float)
df['d4'] = pd.to_numeric(df['d4'], errors='coerce')  # Ensures d4 is numeric
```

```
print(df[['gini', 'gdp', 'population', 'd4']].head())
```

```
    gini     gdp  population    d4
0  29.00  1557.0  26427200.0   NaN
1  33.00  2123.0  30466478.0   NaN
2  31.00  2096.0  35643416.0   NaN
3  27.01  4909.0   3271331.0  7.32
4  31.74  6754.0   3123551.0  6.74
```

```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split

# Define target and predictor variables
target = 'gini'
predictors = ['gdp', 'population', 'd4']

# Ensure all required columns exist
missing_cols = [col for col in [target] + predictors if col not in df.columns]
if missing_cols:
    raise ValueError(f"Missing columns in DataFrame: {missing_cols}")

# Convert to float and handle errors
for col in [target] + predictors:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Fill missing values with median to avoid losing data
df_filled = df.copy()
df_filled[predictors] = df_filled[predictors].fillna(df_filled[predictors].median())
df_filled[target] = df_filled[target].fillna(df_filled[target].median())

# Check if data is still empty
if df_filled[predictors].isnull().sum().sum() > 0 or df_filled[target].isnull().sum() > 0:
    raise ValueError("There are still missing values after filling. Check data preprocessing.")

# Define X (features) and y (target)
X = df_filled[predictors]
y = df_filled[target]

# Ensure there are enough samples for splitting
if len(X) == 0:
    raise ValueError("No valid data left after preprocessing. Check for missing values.")

# Train-test split (ensure we have enough samples)
test_size = min(0.2, len(X) - 1)  # Adjust test size dynamically
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42)

# Add constant for intercept in regression
X_train_const = sm.add_constant(X_train)
X_test_const = sm.add_constant(X_test)

# Fit OLS Regression Model
model = sm.OLS(y_train, X_train_const).fit()

# Print summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   gini   R-squared:                       0.674
Model:                            OLS   Adj. R-squared:                  0.674
Method:                 Least Squares   F-statistic:                 1.343e+04
Date:                Sat, 29 Mar 2025   Prob (F-statistic):               0.00
Time:                        04:39:46   Log-Likelihood:                -59741.
No. Observations:               19493   AIC:                         1.195e+05
Df Residuals:                   19489   BIC:                         1.195e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         83.6955      0.243    344.397      0.000      83.219      84.172
gdp        -2.668e-05   2.04e-06    -13.061      0.000   -3.07e-05   -2.27e-05
population -1.452e-09   2.18e-10     -6.646      0.000   -1.88e-09   -1.02e-09
d4            -7.2686      0.041   -177.138      0.000      -7.349      -7.188
==============================================================================
Omnibus:                     6368.197   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            43351.720
Skew:                           1.399   Prob(JB):                         0.00
Kurtosis:                       9.749   Cond. No.                     1.21e+09
==============================================================================

Notes:
```