Analytical Data Cleaning and Preprocessing Framework for Large-Scale Dataset Using Python

`About the Dataset`

The `COVID-19` dataset contains `166,326 rows` and `67 columns`, representing detailed records of `COVID-19` cases collected from various regions and time periods.

It includes information such as patient demographics, testing details, hospitalization status, vaccination details, symptoms, comorbidities, outcomes, and regional distribution.

The dataset provides a comprehensive view of how `COVID-19` has impacted different populations, allowing for in-depth analysis of infection patterns, recovery rates, and mortality trends.

🎯 Objectives of the Analysis

Data Understanding & Cleaning:

- Explore the dataset structure and identify inconsistencies, missing values, and outliers.
- Standardize column names, formats (dates, categorical values), and remove duplicate or irrelevant records.

Descriptive Analysis:

- Analyze the distribution of cases across age groups, genders, and regions.
- Study trends in confirmed, recovered, and death cases over time.
- Assess the impact of vaccination and comorbidities on recovery outcomes.

Data Visualization:

- Build clear and interactive visualizations using `Power BI / Python (Matplotlib, Seaborn, Plotly)` to represent trends and comparisons effectively.

Insights Generation:

- Identify key factors affecting fatality and recovery rates.
- Correlate infection severity with demographics and pre-existing conditions.
- Provide actionable insights to support healthcare planning and awareness strategies.

1. Import Libraries

In [ ]:
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

2. Import and Load the data

In [ ]:
```
df1= pd.read_csv('covid.csv')
print(df1.head())
```

```
  iso_code continent     location        date  total_cases  new_cases  \
0      AFG      Asia  Afghanistan  2020-02-24          5.0        5.0
1      AFG      Asia  Afghanistan  2020-02-25          5.0        0.0
2      AFG      Asia  Afghanistan  2020-02-26          5.0        0.0
3      AFG      Asia  Afghanistan  2020-02-27          5.0        0.0
4      AFG      Asia  Afghanistan  2020-02-28          5.0        0.0

   new_cases_smoothed  total_deaths  new_deaths  new_deaths_smoothed  ...  \
0                 NaN           NaN         NaN                  NaN  ...
1                 NaN           NaN         NaN                  NaN  ...
2                 NaN           NaN         NaN                  NaN  ...
3                 NaN           NaN         NaN                  NaN  ...
4                 NaN           NaN         NaN                  NaN  ...

   female_smokers  male_smokers  handwashing_facilities  \
0             NaN           NaN                  37.746
1             NaN           NaN                  37.746
2             NaN           NaN                  37.746
3             NaN           NaN                  37.746
4             NaN           NaN                  37.746

   hospital_beds_per_thousand  life_expectancy  human_development_index  \
0                         0.5            64.83                    0.511
1                         0.5            64.83                    0.511
2                         0.5            64.83                    0.511
3                         0.5            64.83                    0.511
4                         0.5            64.83                    0.511

   excess_mortality_cumulative_absolute  excess_mortality_cumulative  \
0                                   NaN                          NaN
1                                   NaN                          NaN
2                                   NaN                          NaN
3                                   NaN                          NaN
4                                   NaN                          NaN

   excess_mortality  excess_mortality_cumulative_per_million
0               NaN                                     NaN
1               NaN                                     NaN
2               NaN                                     NaN
3               NaN                                     NaN
4               NaN                                     NaN
```

```
[5 rows x 67 columns]
```

In [ ]:

3. Data Inspection

📊 Check basic structure

In [ ]:
```
#shape (Rows & Columns)
print(df1.shape)
```

```
(166326, 67)
```

Check Data Info

In [ ]:
```
#info(data type & null)
print(df1.info)
```

```
<bound method DataFrame.info of        iso_code continent     location        date  total_cases  new_cases  \
0           AFG      Asia  Afghanistan  2020-02-24          5.0        5.0
1           AFG      Asia  Afghanistan  2020-02-25          5.0        0.0
2           AFG      Asia  Afghanistan  2020-02-26          5.0        0.0
3           AFG      Asia  Afghanistan  2020-02-27          5.0        0.0
4           AFG      Asia  Afghanistan  2020-02-28          5.0        0.0
...         ...       ...          ...         ...          ...        ...
166321      ZWE    Africa     Zimbabwe  2022-03-01     236871.0      491.0
166322      ZWE    Africa     Zimbabwe  2022-03-02     237503.0      632.0
166323      ZWE    Africa     Zimbabwe  2022-03-03     237503.0        0.0
166324      ZWE    Africa     Zimbabwe  2022-03-04     238739.0     1236.0
166325      ZWE    Africa     Zimbabwe  2022-03-05     239019.0      280.0

        new_cases_smoothed  total_deaths  new_deaths  new_deaths_smoothed  \
0                      NaN           NaN         NaN                  NaN
1                      NaN           NaN         NaN                  NaN
2                      NaN           NaN         NaN                  NaN
3                      NaN           NaN         NaN                  NaN
4                      NaN           NaN         NaN                  NaN
...                    ...           ...         ...                  ...
166321             413.000        5395.0         0.0                1.000
166322             416.286        5396.0         1.0                1.143
166323             362.286        5396.0         0.0                0.857
166324             467.429        5397.0         1.0                0.714
166325             459.429        5397.0         0.0                0.571

        ...  female_smokers  male_smokers  handwashing_facilities  \
0       ...             NaN           NaN                  37.746
1       ...             NaN           NaN                  37.746
2       ...             NaN           NaN                  37.746
3       ...             NaN           NaN                  37.746
4       ...             NaN           NaN                  37.746
...     ...             ...           ...                     ...
166321  ...             1.6          30.7                  36.791
166322  ...             1.6          30.7                  36.791
166323  ...             1.6          30.7                  36.791
166324  ...             1.6          30.7                  36.791
166325  ...             1.6          30.7                  36.791

        hospital_beds_per_thousand  life_expectancy  human_development_index  \
0                              0.5            64.83                    0.511
```

```
1                            0.5           64.83                    0.511
2                            0.5           64.83                    0.511
3                            0.5           64.83                    0.511
4                            0.5           64.83                    0.511
...                          ...             ...                      ...
166321                       1.7           61.49                    0.571
166322                       1.7           61.49                    0.571
166323                       1.7           61.49                    0.571
166324                       1.7           61.49                    0.571
166325                       1.7           61.49                    0.571

        excess_mortality_cumulative_absolute  excess_mortality_cumulative  \
0                                        NaN                          NaN
1                                        NaN                          NaN
2                                        NaN                          NaN
3                                        NaN                          NaN
4                                        NaN                          NaN
...                                      ...                          ...
166321                                   NaN                          NaN
166322                                   NaN                          NaN
166323                                   NaN                          NaN
166324                                   NaN                          NaN
166325                                   NaN                          NaN

        excess_mortality  excess_mortality_cumulative_per_million
0                    NaN                                     NaN
1                    NaN                                     NaN
2                    NaN                                     NaN
3                    NaN                                     NaN
4                    NaN                                     NaN
...                  ...                                     ...
166321               NaN                                     NaN
166322               NaN                                     NaN
166323               NaN                                     NaN
166324               NaN                                     NaN
166325               NaN                                     NaN

[166326 rows x 67 columns]>
```

In [ ]:

Check All Columns Name

In [ ]:
```
# All columns
print(df1.columns)
```

```
Index(['iso_code', 'continent', 'location', 'date', 'total_cases', 'new_cases',
       'new_cases_smoothed', 'total_deaths', 'new_deaths',
       'new_deaths_smoothed', 'total_cases_per_million',
       'new_cases_per_million', 'new_cases_smoothed_per_million',
       'total_deaths_per_million', 'new_deaths_per_million',
       'new_deaths_smoothed_per_million', 'reproduction_rate', 'icu_patients',
       'icu_patients_per_million', 'hosp_patients',
       'hosp_patients_per_million', 'weekly_icu_admissions',
       'weekly_icu_admissions_per_million', 'weekly_hosp_admissions',
       'weekly_hosp_admissions_per_million', 'new_tests', 'total_tests',
       'total_tests_per_thousand', 'new_tests_per_thousand',
       'new_tests_smoothed', 'new_tests_smoothed_per_thousand',
       'positive_rate', 'tests_per_case', 'tests_units', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated', 'total_boosters',
       'new_vaccinations', 'new_vaccinations_smoothed',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred', 'total_boosters_per_hundred',
       'new_vaccinations_smoothed_per_million',
       'new_people_vaccinated_smoothed',
       'new_people_vaccinated_smoothed_per_hundred', 'stringency_index',
       'population', 'population_density', 'median_age', 'aged_65_older',
       'aged_70_older', 'gdp_per_capita', 'extreme_poverty',
       'cardiovasc_death_rate', 'diabetes_prevalence', 'female_smokers',
       'male_smokers', 'handwashing_facilities', 'hospital_beds_per_thousand',
       'life_expectancy', 'human_development_index',
       'excess_mortality_cumulative_absolute', 'excess_mortality_cumulative',
       'excess_mortality', 'excess_mortality_cumulative_per_million'],
      dtype='object')
```

Check Column Data Types

In [ ]:
```
# column data types
print(df1.dtypes)
```

```
iso_code                                  object
continent                                 object
location                                  object
date                                      object
total_cases                              float64
                                          ...
human_development_index                  float64
excess_mortality_cumulative_absolute     float64
excess_mortality_cumulative              float64
excess_mortality                         float64
excess_mortality_cumulative_per_million  float64
Length: 67, dtype: object
```

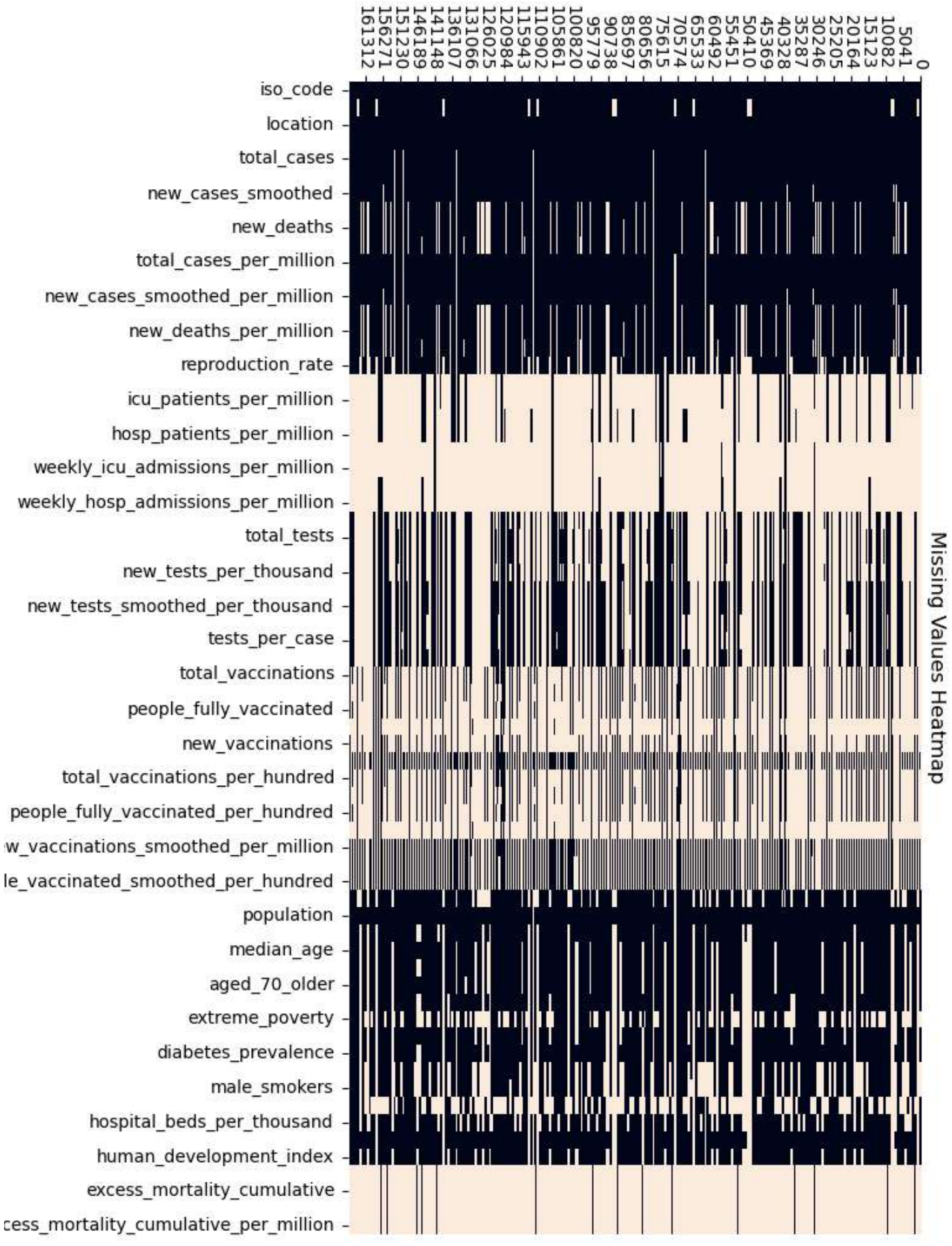
Null Value Check

In [ ]:
```
print(df1.isnull().sum())
```

```
iso_code                                     0
continent                                 9956
location                                     0
date                                         0
total_cases                               3033
                                          ...
human_development_index                  30073
excess_mortality_cumulative_absolute    160630
excess_mortality_cumulative             160630
excess_mortality                        160630
excess_mortality_cumulative_per_million 160630
Length: 67, dtype: int64
```

Missing Values Heatmap

In [ ]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df1.isnull(), cbar=False)
plt.title("Missing Values Heatmap")
plt.show()
```


161312
156271
151230
146189
141148
136107
131066
126025
120984
115943
110902
105861
100820
95779
90738
85697
80656
75615
70574
65533
60492
55451
50410
45369
40328
35287
30246
25205
20164
15123
10082
5041
0
iso_code
location
total_cases
new_cases_smoothed
new_deaths
total_cases_per_million
new_cases_smoothed_per_million
new_deaths_per_million
reproduction_rate
icu_patients_per_million
hosp_patients_per_million
weekly_icu_admissions_per_million
weekly_hosp_admissions_per_million
total_tests
new_tests_per_thousand
new_tests_smoothed_per_thousand
tests_per_case
total_vaccinations
people_fully_vaccinated
new_vaccinations
total_vaccinations_per_hundred
people_fully_vaccinated_per_hundred
w_vaccinations_smoothed_per_million
le_vaccinated_smoothed_per_hundred
population
median_age
aged_70_older
extreme_poverty
diabetes_prevalence
male_smokers
hospital_beds_per_thousand
human_development_index
excess_mortality_cumulative
cess_mortality_cumulative_per_million
Missing Values Heatmap

ne new_peopl ex

In [ ]:

1234 Check Summary Statistics For Numerical Columns

In [ ]:
```
print(df1.describe())
#describe(include ='all') for all
```

```
        total_cases     new_cases  new_cases_smoothed  total_deaths  \
count  1.632930e+05  1.631330e+05        1.611500e+05  1.454510e+05
mean   2.536044e+06  1.157084e+04        1.156560e+04  5.766407e+04
std    1.543441e+07  8.442598e+04        8.257830e+04  3.021145e+05
min    1.000000e+00  0.000000e+00        0.000000e+00  1.000000e+00
25%    2.001000e+03  1.000000e+00        7.000000e+00  7.900000e+01
50%    2.611700e+04  7.900000e+01        1.071430e+02  7.830000e+02
75%    2.987020e+05  1.063000e+03        1.146000e+03  7.307000e+03
max    4.451295e+08  4.206334e+06        3.444237e+06  5.995245e+06

          new_deaths  new_deaths_smoothed  total_cases_per_million  \
count  145487.000000        143390.000000            162535.000000
mean      171.137304           172.673031             29447.756785
std       832.251328           817.024076             51852.379656
min         0.000000             0.000000                 0.001000
25%         0.000000             0.143000               623.579000
50%         2.000000             2.429000              4731.521000
75%        20.000000            21.286000             37724.466000
max     18020.000000         14689.143000            706541.904000

       new_cases_per_million  new_cases_smoothed_per_million  \
count          162375.000000                   160398.000000
mean              166.431538                      165.507110
std               683.021740                      532.174029
min                 0.000000                        0.000000
25%                 0.042000                        1.630000
50%                11.439000                       18.829000
75%               101.289500                      120.859750
max             51427.491000                    16052.608000

       total_deaths_per_million  ...  female_smokers   male_smokers  \
count             144706.000000  ...   106050.000000  104595.000000
mean                 509.384956  ...       10.627229      32.778221
std                  784.551311  ...       10.558306      13.523688
min                    0.000000  ...        0.100000       7.700000
25%                   18.580250  ...        1.900000      21.600000
50%                  127.737500  ...        6.300000      31.400000
75%                  711.954750  ...       19.300000      41.300000
max                 6322.263000  ...       44.000000      78.100000

       handwashing_facilities  hospital_beds_per_thousand  life_expectancy  \
```

```
count            68569.000000               123664.000000    155268.000000
mean                50.788710                    3.027816        73.576309
std                 31.811788                    2.450110         7.491615
min                  1.188000                    0.100000        53.280000
25%                 19.351000                    1.300000        69.500000
50%                 49.839000                    2.400000        75.050000
75%                 83.241000                    4.000000        78.930000
max                100.000000                   13.800000        86.750000

       human_development_index  excess_mortality_cumulative_absolute  \
count            136253.000000                          5.696000e+03
mean                  0.725587                          3.761302e+04
std                   0.149964                          1.043065e+05
min                   0.394000                         -3.772610e+04
25%                   0.602000                         -7.515000e+01
50%                   0.743000                          3.424600e+03
75%                   0.845000                          2.478462e+04
max                   0.957000                          1.080748e+06

       excess_mortality_cumulative  excess_mortality  \
count                  5696.000000       5696.000000
mean                      9.404336         15.967077
std                      16.439173         30.092830
min                     -28.450000        -95.920000
25%                      -0.722500         -0.752500
50%                       6.065000          7.195000
75%                      14.520000         22.997500
max                     111.010000        374.930000

       excess_mortality_cumulative_per_million
count                              5696.000000
mean                                972.197816
std                                1420.342295
min                               -1826.595723
25%                                 -29.788442
50%                                 473.393182
75%                                1656.361067
max                                9153.060433

[8 rows x 62 columns]
```

In [ ]:

Before Data Cleaning Make A Copy Of The Data

In [ ]:
```
df=df1.copy()
```

4. Data Cleaning

- Correcting data types
- Removing duplicates
- Handling `missing/null` values
- Handling inconsistent entries
- `Outlier detection` and `capping`
- Skewness treatment `(log/box-cox transformations)`

🧠Correction Date Column Data Type

In [ ]:
```
df1['date']= pd.to_datetime(df1['date'])
print(df['date'].dtype)
```

```
datetime64[ns]
```

🔍 Check for Duplicates and Drop if required

- `df1.duplicated().sum()`
- If > 0, you can remove them:

`df1.drop_duplicates(inplace=True)`

🔍Cleaning Categorical Data

- `df1['col']=df1['col'].str.strip().str.title()` `for standardization`
- `df1['col'].str.lower()` `for lower text`

- `df1['col'].str.upper()` for upper text
- `df1['col'].str.title()` for cap each word

In [ ]:
```
df1.duplicated().sum()
```

Out[ ]:
```
0
```

Check Missing Values & Nulls

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
iso_code                                    0
continent                                9956
location                                    0
date                                        0
total_cases                              3033
                                           ...
human_development_index                 30073
excess_mortality_cumulative_absolute   160630
excess_mortality_cumulative            160630
excess_mortality                       160630
excess_mortality_cumulative_per_million    160630
Length: 67, dtype: int64
```

In [ ]:
```

```

Check which columns are completely empty

In [ ]:
```
empty_cols = df.columns[df.isna().sum() == len(df)]

print("🗑️ Columns that are completely empty:")
print(empty_cols)
```

```
🗑️ Columns that are completely empty:
Index([], dtype='object')
```

Drop those empty columns

```
df.drop(columns=empty_cols, inplace=True)

print(f"✅ Removed {len(empty_cols)} completely empty columns.")
```

In [ ]:

Calculate Missing Percentage of Every Column

In [ ]:
```
# Calculate missing values (count)
missing_count = df.isna().sum()


# Calculate missing percentage
missing_percent = (missing_count / len(df)) * 100


# Combine into a clean summary DataFrame
missing_summary = pd.DataFrame({
    'Missing_Count': missing_count,
    'Missing_%': missing_percent.round(2)

})


# Sort from most missing to least missing
missing_summary = missing_summary.sort_values(by='Missing_%', ascending=True)


# Display top 15 columns with missing data
print("📊 Missing Values Summary (Top 20):")
print(missing_summary.head(20))

print("📊 Missing Values Summary (Bottom 20):")
print(missing_summary.tail(20))
```

📊 Missing Values Summary (Top 20):

	Missing_Count	Missing_%
iso_code	0	0.00
location	0	0.00
date	0	0.00
population	1075	0.65
total_cases	3033	1.82
new_cases	3193	1.92
total_cases_per_million	3791	2.28
new_cases_per_million	3951	2.38
new_cases_smoothed	5176	3.11
new_cases_smoothed_per_million	5928	3.56
continent	9956	5.99
life_expectancy	11058	6.65
population_density	18398	11.06
new_deaths	20839	12.53
total_deaths	20875	12.55
new_deaths_per_million	21584	12.98
total_deaths_per_million	21620	13.00
diabetes_prevalence	22377	13.45
new_deaths_smoothed	22936	13.79
new_deaths_smoothed_per_million	23675	14.23

📊 Missing Values Summary (Bottom 20):

	Missing_Count	Missing_%
total_vaccinations_per_hundred	121132	72.83
people_vaccinated	123339	74.15
people_vaccinated_per_hundred	123339	74.15
people_fully_vaccinated	126085	75.81
people_fully_vaccinated_per_hundred	126085	75.81
new_vaccinations	128879	77.49
hosp_patients_per_million	141709	85.20
hosp_patients	141709	85.20
icu_patients_per_million	142863	85.89
icu_patients	142863	85.89
total_boosters	148787	89.46
total_boosters_per_hundred	148787	89.46
weekly_hosp_admissions_per_million	155403	93.43
weekly_hosp_admissions	155403	93.43
excess_mortality_cumulative	160630	96.58
excess_mortality_cumulative_absolute	160630	96.58
excess_mortality	160630	96.58

```
excess_mortality_cumulative_per_million          160630      96.58
weekly_icu_admissions_per_million                160893      96.73
weekly_icu_admissions                            160893      96.73
```

In [ ]:

Calculate Missing % Of Total Of The Data

In [ ]:
```
total_missing_percent = (df.isnull().sum().sum() / (df.shape[0] * df.shape[1])) * 100
print(f"Total missing percentage in dataset: {total_missing_percent:.2f}%")
```

```
Total missing percentage in dataset: 44.54%
```

In [ ]:

Visualize missing values per column (bar chart)

In [ ]:
```
missing_percent = df.isnull().mean() * 100

# Sort descending for clarity
missing_percent = missing_percent.sort_values(ascending=False)

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x=missing_percent.values, y=missing_percent.index)
plt.title("Missing Value Percentage per Column", fontsize=14)
plt.xlabel("Percentage (%)")
plt.ylabel("Columns")
plt.show()
```

In [ ]:

Separate Numeric & Categorical Columns

In [ ]:
```
num_cols = df.select_dtypes(include=['float64', 'int64']).columns
cat_cols = df.select_dtypes(include=['object']).columns
```

In [ ]:
```
print(cat_cols)
```

```
Index(['iso_code', 'continent', 'location', 'tests_units'], dtype='object')
```

In [ ]:
```
print(len(num_cols))
```

```
41
```

Check Nulls & Missing Values In Categorical Columns

In [ ]:
```
df[cat_cols].isnull().sum()
```

Out[ ]:
```
iso_code          0
continent       9956
location           0
tests_units    79940
dtype: int64
```

In [ ]:
```
df['continent']
```

Out[ ]:
```
0           Asia
1           Asia
2           Asia
3           Asia
4           Asia
           ...
166321    Africa
166322    Africa
166323    Africa
166324    Africa
166325    Africa
Name: continent, Length: 166326, dtype: object
```

Fill Categorical Columns

1. Fill with the Most Frequent Value(Mean/Median or Mode)

`most_frequent = df['column_name'].mode()[0]` find the most frequent

`df['column_name'] = df['column_name'].fillna(most_frequent)`

2. Fill with a New Category or Default

`df['column_name'] = df['column_name'].fillna('unknown')`

3. Fill Using Random Sampling from Existing Values

---

filling categorical columns * `df['tests_units']= df['tests_units'].fillna('unknown')` * `Print(df['tests_units'])` * `df['continent']= df['continent'].fillna('unknown')` * `print(df['continent'])`

Fill Categorical Columns (if large number of columns present)

```
for col in cat_cols:
   mode_value = df[col].mode()[0]
   df[col].fillna(mode_value)
```

In [ ]:
```
#print(cat_cols)
#print(cat_cols.isna().sum())    #check null
```

`print(cat_cols)`

`print(cat_cols.isna().sum())` `-check nulls`

📜 Data Cleaning & Preprocessing — Numeric Columns

Handling Missing Values

Objective:

- `Ensure no null or NaN values remain before further processing.`

Steps:

- `Check percentage of missing values per column`
- `Drop columns with >70% missing data if column is not important`
- `Fill remaining missing values (usually with median or mean)`

🧭 Correct Sequence for Numeric Data Cleaning

1️⃣ Handle missing values: `Needed before stats like skewness/outliers`

`Method : Median/mean/imputation`

2️⃣ Handle outliers: `Outliers distort distribution and skewness`

`Method : IQR, Z-score capping`

3️⃣ Fix skewness: `Apply transformation only after data is clean`

`Method : Log, sqrt, Box-Cox, Yeo-Johnson`

In [ ]:
```
num_cols = df.select_dtypes(include=['float64', 'int64']).columns
```

Calculate missing % of each column

In [ ]:
```
# Calculate missing percentage for each column
missing_percent = df.isnull().mean()
print(missing_percent)
```

```
iso_code                                    0.000000
continent                                   0.000000
location                                    0.000000
date                                        0.000000
total_cases                                 0.018235
                                              ...
human_development_index                     0.180808
excess_mortality_cumulative_absolute        0.965754
excess_mortality_cumulative                 0.965754
excess_mortality                            0.965754
excess_mortality_cumulative_per_million     0.965754
Length: 67, dtype: float64
```

In [ ]:
```

```

Keeps only columns with ≤70% missing

In [ ]:
```
# Assuming df is your DataFrame
threshold = 0.7  # 70%
# Drop columns with more than 70% missing data
df = df.loc[:, missing_percent <= threshold]

# (Optional) See what was removed
dropped_cols = missing_percent[missing_percent > threshold].index.tolist()
print("Dropped columns:", dropped_cols)
```

```
Dropped columns: ['icu_patients', 'icu_patients_per_million', 'hosp_patients', 'hosp_patients_per_million', 'weekly_icu_admissi
ons', 'weekly_icu_admissions_per_million', 'weekly_hosp_admissions', 'weekly_hosp_admissions_per_million', 'total_vaccination
s', 'people_vaccinated', 'people_fully_vaccinated', 'total_boosters', 'new_vaccinations', 'total_vaccinations_per_hundred', 'pe
ople_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred', 'total_boosters_per_hundred', 'excess_mortality_cumulative
_absolute', 'excess_mortality_cumulative', 'excess_mortality', 'excess_mortality_cumulative_per_million']
```

In [ ]:
```
print(df.shape)
```

```
(166326, 46)
```

In [ ]:
```

```

If you want to apply the same rule only to numeric columns

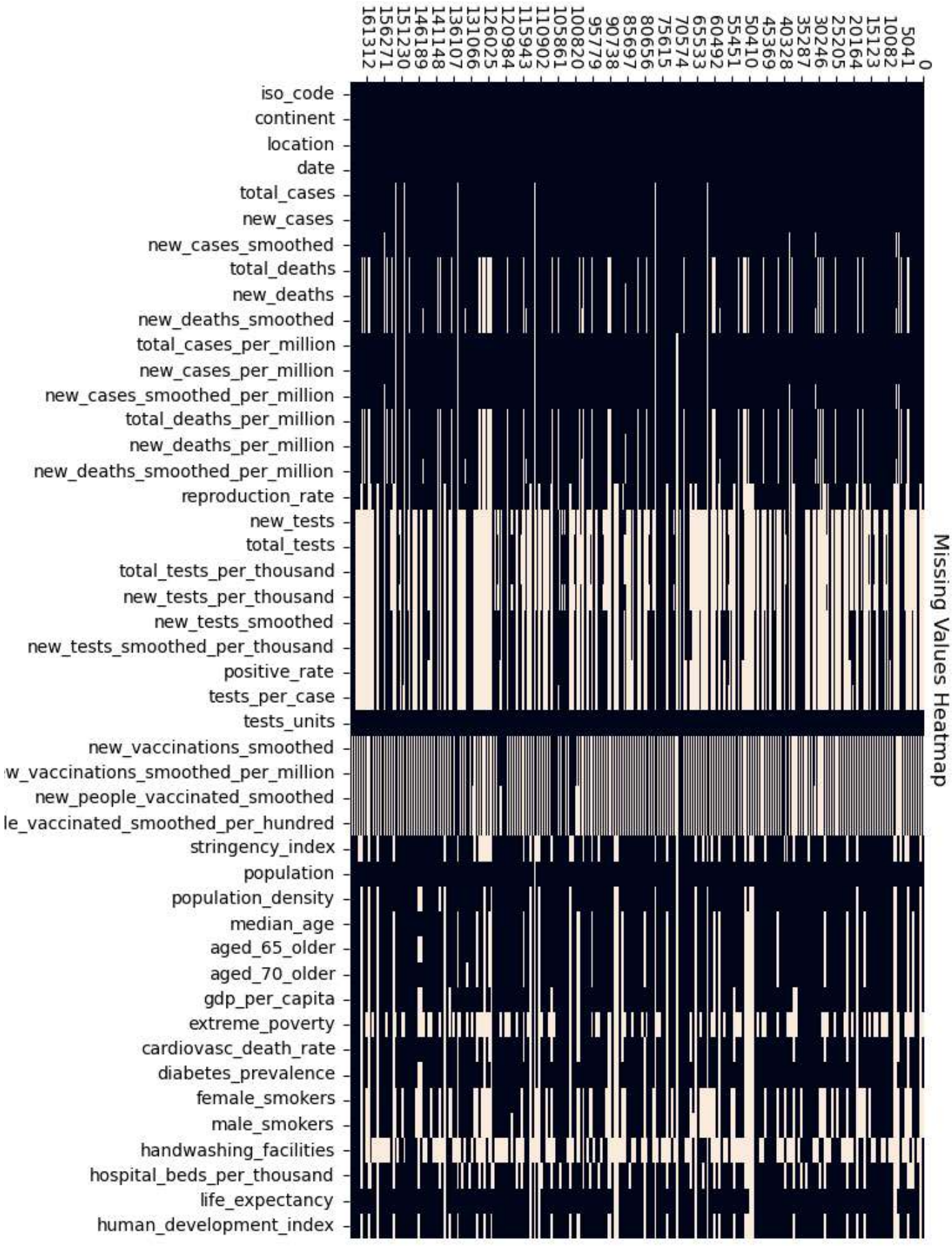
```
num_cols = df.select_dtypes(include=['number']).columns
```

```
missing_percent = df[num_cols].isnull().mean()
```

```
df = df.drop(columns=missing_percent[missing_percent > 0.7].index)
```

Again Check Heatmap for missing data

In [ ]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cbar=False)
plt.title("Missing Values Heatmap")
plt.show()
```


161312
156271
151230
146189
141148
136107
131066
126025
120984
115943
110902
105861
100820
95779
90738
85697
80656
75615
70574
65533
60492
55451
50410
45369
40328
35287
30246
25205
20164
15123
10082
5041
0
iso_code
continent
location
date
total_cases
new_cases
new_cases_smoothed
total_deaths
new_deaths
new_deaths_smoothed
total_cases_per_million
new_cases_per_million
new_cases_smoothed_per_million
total_deaths_per_million
new_deaths_per_million
new_deaths_smoothed_per_million
reproduction_rate
new_tests
total_tests
total_tests_per_thousand
new_tests_per_thousand
new_tests_smoothed
new_tests_smoothed_per_thousand
positive_rate
tests_per_case
tests_units
new_vaccinations_smoothed
w_vaccinations_smoothed_per_million
new_people_vaccinated_smoothed
le_vaccinated_smoothed_per_hundred
stringency_index
population
population_density
median_age
aged_65_older
aged_70_older
gdp_per_capita
extreme_poverty
cardiovasc_death_rate
diabetes_prevalence
female_smokers
male_smokers
handwashing_facilities
hospital_beds_per_thousand
life_expectancy
human_development_index
Missing Values Heatmap

ne

new_peopl

Best Practice to handle missing values

1️⃣ Handle missing values using skewness-based method (it adapts per column)

2️⃣ Then handle outliers (capping via IQR or z-score)

3️⃣ Then fix skewness again if needed (log, yeo-johnson)

Check skewness for only numeric column

In [ ]:
```
# check skewness of the columns
skew_values = df[num_cols].skew().sort_values(ascending=False)

# Display the top 15 skewed columns
print("📊 Skewness of Numerical Columns (Top 25):")
print(skew_values.tail(25))
```

```
📊 Skewness of Numerical Columns (Top 25):
total_tests                              7.650397
new_tests_smoothed_per_thousand          7.613275
new_deaths_smoothed_per_million          7.461201
total_tests_per_thousand                 6.455212
new_tests_smoothed                       6.305285
new_tests                                6.243854
new_vaccinations_smoothed_per_million    4.865356
total_cases_per_million                  3.195068
total_deaths_per_million                 2.386192
positive_rate                            1.974944
gdp_per_capita                           1.842146
hospital_beds_per_thousand               1.753916
extreme_poverty                          1.621607
diabetes_prevalence                      1.386603
female_smokers                           0.995240
aged_70_older                            0.929393
cardiovasc_death_rate                    0.861898
reproduction_rate                        0.815390
aged_65_older                            0.788319
male_smokers                             0.523345
median_age                               0.095010
handwashing_facilities                  -0.044368
stringency_index                        -0.178710
human_development_index                 -0.353307
life_expectancy                         -0.659925
dtype: float64
```

In [ ]:

Check non skew value

In [ ]:
```
nan_skew_cols = skew_values[skew_values.isna()].index
print("Columns with NaN skew:", list(nan_skew_cols))
```

```
Columns with NaN skew: []
```

Fill in Python (Even With High Skew)

In [ ]:
```
num_cols = df.select_dtypes(include=['int64', 'float64']).columns

for col in num_cols:
    if df[col].isnull().sum() > 0:
        skewness = df[col].skew()

        # Decide fill method
        if abs(skewness) < 0.5:
            fill_value = df[col].mean()
            method = "mean"
        elif abs(skewness) < 1:
            fill_value = df[col].median()
            method = "median"
        else:
            fill_value = df[col].mode()[0]
            method = "mode"

        df[col].fillna(fill_value)

        print(f"{col}: missing values filled using {method} (skewness = {skewness:.3f})")
```

Check The Null values Again

In [ ]:
```
df.isnull().sum()
```

Out[ ]:

iso_code	0
continent	0
location	0
date	0
total_cases	0
new_cases	0
new_cases_smoothed	0
total_deaths	0
new_deaths	0
new_deaths_smoothed	0
total_cases_per_million	0
new_cases_per_million	0
new_cases_smoothed_per_million	0
total_deaths_per_million	0
new_deaths_per_million	0
new_deaths_smoothed_per_million	0
reproduction_rate	0
new_tests	0
total_tests	0
total_tests_per_thousand	0
new_tests_per_thousand	0
new_tests_smoothed	0
new_tests_smoothed_per_thousand	0
positive_rate	0
tests_per_case	0
tests_units	0
new_vaccinations_smoothed	0
new_vaccinations_smoothed_per_million	0
new_people_vaccinated_smoothed	0
new_people_vaccinated_smoothed_per_hundred	0
stringency_index	0
population	0
population_density	0
median_age	0
aged_65_older	0
aged_70_older	0
gdp_per_capita	0
extreme_poverty	0
cardiovasc_death_rate	0
diabetes_prevalence	0

```
female_smokers                    0
male_smokers                      0
handwashing_facilities            0
hospital_beds_per_thousand        0
life_expectancy                   0
human_development_index           0
dtype: int64
```

Perfect No Null Values Found

In [ ]:

Check Skewness Again After Cleaning The Data

In [ ]:
```
skew_values = df[num_cols].skew().sort_values(ascending=True)

# Display the top 15 skewed columns
print("📊 Skewness of Numerical Columns (Top 25):")
print(skew_values.tail(25))
```

```
📊 Skewness of Numerical Columns (Top 25):
positive_rate                                 2.943986
total_cases_per_million                       3.226455
new_vaccinations_smoothed_per_million         5.446572
new_deaths_smoothed_per_million               7.891926
population                                    8.125707
population_density                            8.577806
new_tests_smoothed                            8.877099
new_deaths_smoothed                           9.193840
new_cases_smoothed_per_million                9.539779
new_deaths                                    9.573634
total_tests_per_thousand                      9.659277
new_tests                                     9.818973
new_tests_smoothed_per_thousand              10.770022
total_deaths                                 11.440344
total_tests                                  11.827990
new_vaccinations_smoothed                    12.359942
total_cases                                  13.113016
new_people_vaccinated_smoothed               13.755047
new_people_vaccinated_smoothed_per_hundred   13.857364
new_tests_per_thousand                       17.857501
new_cases_per_million                        17.906697
new_deaths_per_million                       18.712848
new_cases_smoothed                           19.985897
new_cases                                    20.715828
tests_per_case                               95.214415
dtype: float64
```

Skewness Range Meaning What To Do

- `[-0.5 to +0.5]` Fairly symmetrical Leave as is ✅
- `[±0.5 to ±1]` Moderate skew Maybe adjust (optional)
- `[> +1 or < -1]` Strong skew Should transform (log, sqrt, etc.)
- `[> +5]` Very high skew Definitely transform (log/Box-Cox)

Detect Outliers Using IQR (Standard Method)

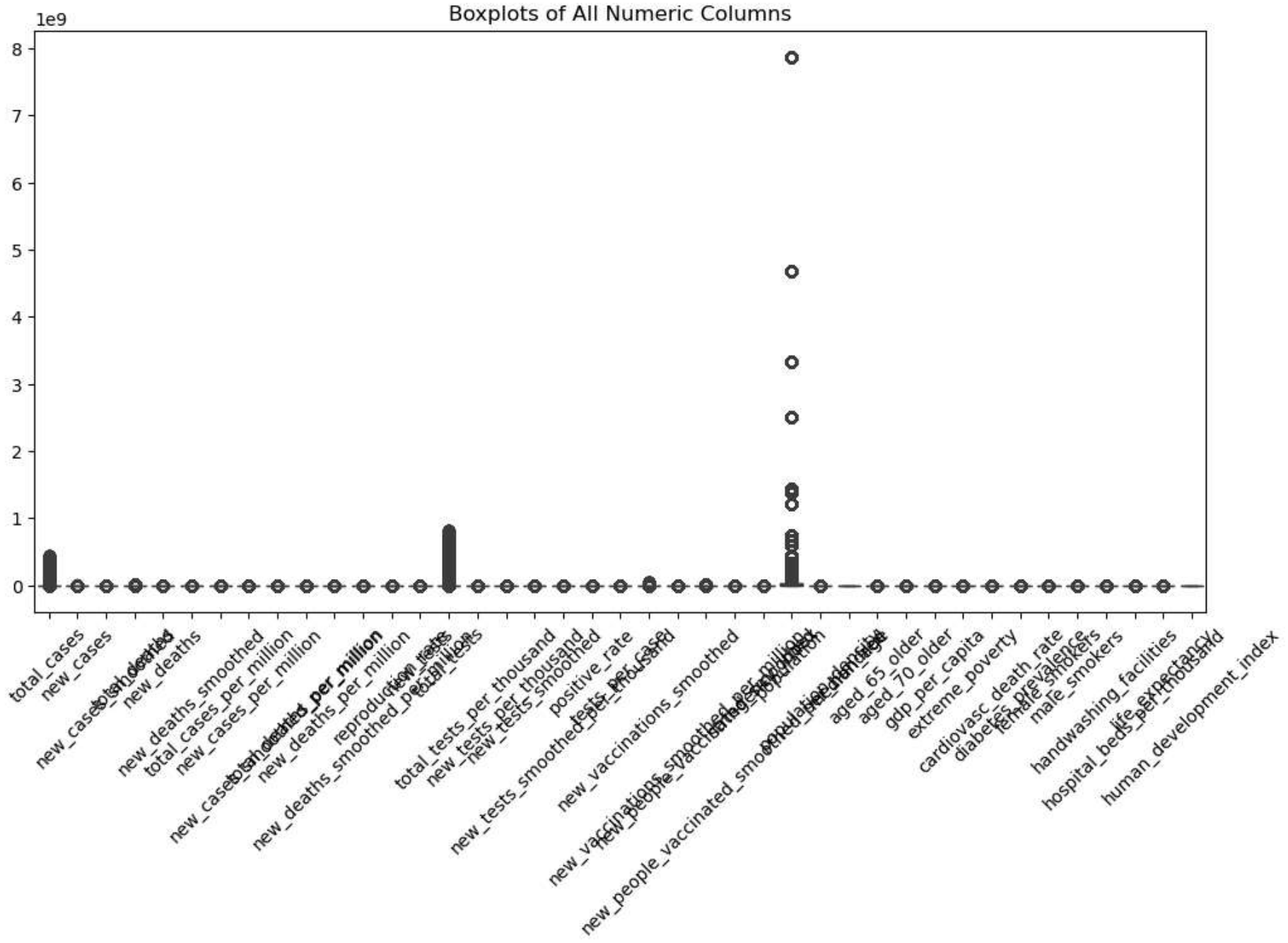
In [ ]:
```
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_limit = Q1 - 1.5 * IQR
    upper_limit = Q3 + 1.5 * IQR

    outliers = df[(df[col] < lower_limit) | (df[col] > upper_limit)]
    print(f"{col}  :  {len(outliers)}   ** outliers detected")
```

```
total_cases  :  27066   ** outliers detected
new_cases  :  27895   ** outliers detected
new_cases_smoothed  :  27613   ** outliers detected
total_deaths  :  28137   ** outliers detected
new_deaths  :  28972   ** outliers detected
new_deaths_smoothed  :  28910   ** outliers detected
total_cases_per_million  :  16850   ** outliers detected
new_cases_per_million  :  22567   ** outliers detected
new_cases_smoothed_per_million  :  21194   ** outliers detected
total_deaths_per_million  :  18698   ** outliers detected
new_deaths_per_million  :  24956   ** outliers detected
new_deaths_smoothed_per_million  :  22237   ** outliers detected
reproduction_rate  :  19799   ** outliers detected
new_tests  :  30412   ** outliers detected
total_tests  :  30543   ** outliers detected
total_tests_per_thousand  :  30453   ** outliers detected
new_tests_per_thousand  :  27736   ** outliers detected
new_tests_smoothed  :  28494   ** outliers detected
new_tests_smoothed_per_thousand  :  57939   ** outliers detected
positive_rate  :  21484   ** outliers detected
tests_per_case  :  27348   ** outliers detected
new_vaccinations_smoothed  :  31464   ** outliers detected
new_vaccinations_smoothed_per_million  :  16426   ** outliers detected
new_people_vaccinated_smoothed  :  31292   ** outliers detected
new_people_vaccinated_smoothed_per_hundred  :  21753   ** outliers detected
stringency_index  :  3283   ** outliers detected
population  :  22736   ** outliers detected
population_density  :  14624   ** outliers detected
median_age  :  0   ** outliers detected
aged_65_older  :  1539   ** outliers detected
aged_70_older  :  3785   ** outliers detected
gdp_per_capita  :  9758   ** outliers detected
extreme_poverty  :  32422   ** outliers detected
cardiovasc_death_rate  :  6989   ** outliers detected
diabetes_prevalence  :  3409   ** outliers detected
female_smokers  :  28905   ** outliers detected
male_smokers  :  25097   ** outliers detected
handwashing_facilities  :  68569   ** outliers detected
hospital_beds_per_thousand  :  6939   ** outliers detected
life_expectancy  :  3542   ** outliers detected
human_development_index  :  0   ** outliers detected
```

In [ ]:

Check Outliers Visuals(Boxplot)

In [ ]:
```
num_cols = df.select_dtypes(include=['int64', 'float64']).columns

plt.figure(figsize=(12, 6))
sns.boxplot(data=df[num_cols])
plt.xticks(rotation=45)
plt.title('Boxplots of All Numeric Columns')
plt.show()
```


Boxplots of All Numeric Columns
1e9
8
7
6
5
4
3
2
1
0
total_cases
new_cases
new_deaths
new_deaths_smoothed
total_cases_per_million
new_cases_per_million
new_deaths_per_million
total_tests_per_thousand
new_tests_per_thousand
new_tests_smoothed
positive_rate
new_vaccinations_smoothed
aged_65_older
aged_70_older
gdp_per_capita
extreme_poverty
cardiovasc_death_rate
male_smokers
handwashing_facilities
hospital_beds_per_thousand
human_development_index

Handle Outliers

Option A: Remove Outliers

```
for col in num_cols:

        Q1 = df[col].quantile(0.25)

        Q3 = df[col].quantile(0.75)

        IQR = Q3 - Q1

        lower = Q1 - 1.5*IQR

        upper = Q3 + 1.5*IQR
```

`df = df[(df[col] >= lower) & (df[col] <= upper)]`

`⚠ Caution: Doing this on multiple columns may remove a lot of rows.(not safer option)`

Option B: Cap Outliers (Winsorization)

- Replace outliers with nearest limit (upper or lower):
- `df[col] = np.where(df[col] < lower_limit, lower_limit,`

  `np.where(df[col] > upper_limit, upper_limit, df[col]))`
- This is safer for datasets with many numeric columns:

Option C: Flag outliers instead of removing — keep them for later analysis:

Cap Outliers (Winsorization)

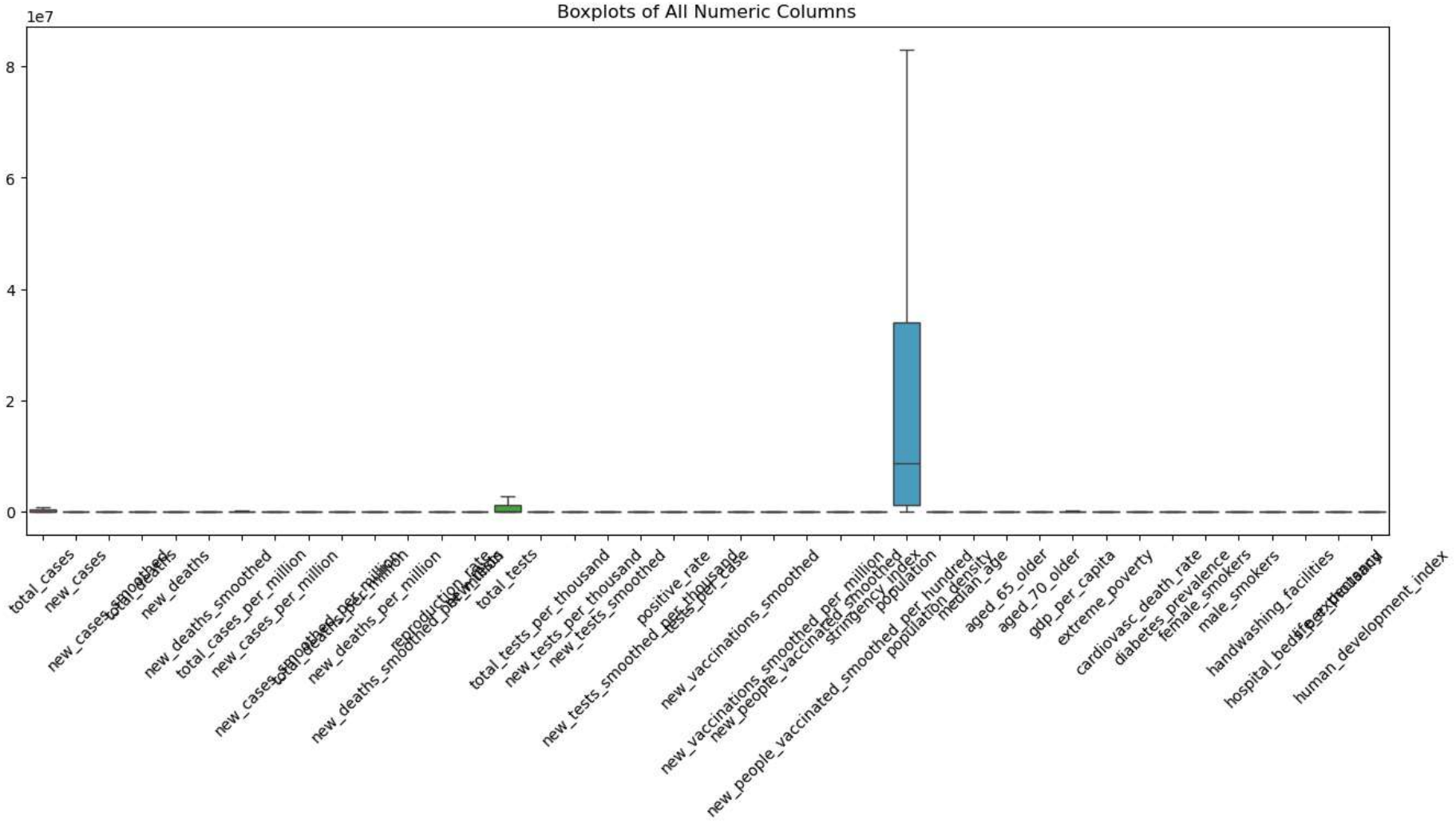
In [ ]:
```
# Cap the outliers in-place

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

for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_limit = Q1 - 1.5 * IQR
    upper_limit = Q3 + 1.5 * IQR

    # Cap outliers safely with .loc
    df.loc[:, col] = np.where(df[col] < lower_limit, lower_limit,
                              np.where(df[col] > upper_limit, upper_limit, df[col]))

    print(f"{col}: Outliers capped at {lower_limit:.2f} (lower) and {upper_limit:.2f} (upper)")
```

```
total_cases: Outliers capped at -422691.00 (lower) and 708725.00 (upper)
new_cases: Outliers capped at -1519.50 (lower) and 2532.50 (upper)
new_cases_smoothed: Outliers capped at -1572.93 (lower) and 2634.50 (upper)
total_deaths: Outliers capped at -7848.50 (lower) and 13139.50 (upper)
new_deaths: Outliers capped at -21.00 (lower) and 35.00 (upper)
new_deaths_smoothed: Outliers capped at -21.86 (lower) and 36.43 (upper)
total_cases_per_million: Outliers capped at -52369.80 (lower) and 89033.07 (upper)
new_cases_per_million: Outliers capped at -144.77 (lower) and 241.28 (upper)
new_cases_smoothed_per_million: Outliers capped at -166.42 (lower) and 280.49 (upper)
total_deaths_per_million: Outliers capped at -800.69 (lower) and 1402.77 (upper)
new_deaths_per_million: Outliers capped at -1.52 (lower) and 2.53 (upper)
new_deaths_smoothed_per_million: Outliers capped at -2.01 (lower) and 3.34 (upper)
reproduction_rate: Outliers capped at 0.53 (lower) and 1.46 (upper)
new_tests: Outliers capped at -7625.00 (lower) and 12711.00 (upper)
total_tests: Outliers capped at -1639123.38 (lower) and 2731877.62 (upper)
total_tests_per_thousand: Outliers capped at -147.94 (lower) and 246.57 (upper)
new_tests_per_thousand: Outliers capped at -0.83 (lower) and 1.38 (upper)
new_tests_smoothed: Outliers capped at -11073.12 (lower) and 21001.88 (upper)
new_tests_smoothed_per_thousand: Outliers capped at 0.39 (lower) and 1.70 (upper)
positive_rate: Outliers capped at -0.08 (lower) and 0.13 (upper)
tests_per_case: Outliers capped at -12.70 (lower) and 32.10 (upper)
new_vaccinations_smoothed: Outliers capped at -14874.00 (lower) and 24790.00 (upper)
new_vaccinations_smoothed_per_million: Outliers capped at -3385.50 (lower) and 5642.50 (upper)
new_people_vaccinated_smoothed: Outliers capped at -6009.00 (lower) and 10015.00 (upper)
new_people_vaccinated_smoothed_per_hundred: Outliers capped at -0.11 (lower) and 0.18 (upper)
stringency_index: Outliers capped at 12.49 (lower) and 97.69 (upper)
population: Outliers capped at -47972725.00 (lower) and 83080859.00 (upper)
population_density: Outliers capped at -259.41 (lower) and 485.02 (upper)
median_age: Outliers capped at 4.65 (lower) and 57.05 (upper)
aged_65_older: Outliers capped at -7.00 (lower) and 22.17 (upper)
aged_70_older: Outliers capped at -4.77 (lower) and 14.14 (upper)
gdp_per_capita: Outliers capped at -21807.68 (lower) and 52403.62 (upper)
extreme_poverty: Outliers capped at -4.60 (lower) and 8.20 (upper)
cardiovasc_death_rate: Outliers capped at 21.81 (lower) and 472.90 (upper)
diabetes_prevalence: Outliers capped at -5.27 (lower) and 19.29 (upper)
female_smokers: Outliers capped at -3.55 (lower) and 17.65 (upper)
male_smokers: Outliers capped at 15.95 (lower) and 47.55 (upper)
handwashing_facilities: Outliers capped at 50.79 (lower) and 50.79 (upper)
hospital_beds_per_thousand: Outliers capped at -3.23 (lower) and 7.25 (upper)
life_expectancy: Outliers capped at 56.26 (lower) and 92.42 (upper)
human_development_index: Outliers capped at 0.39 (lower) and 1.07 (upper)
```

In [ ]:

Visualization After Capping

In [ ]:
```
plt.figure(figsize=(16, 6))
sns.boxplot(data=df[num_cols])
plt.xticks(rotation=45)
plt.title('Boxplots of All Numeric Columns')
plt.show()
```



In [ ]:

Check Skewness Again After Capping(Winsorization) The Data

In [ ]:
```
for col in num_cols:
    skewness = df[col].skew()
    print(f"{col}: skewness = {skewness:.2f}")
```

```
total_cases: skewness = 1.20
new_cases: skewness = 1.20
new_cases_smoothed: skewness = 1.20
total_deaths: skewness = 1.21
new_deaths: skewness = 1.20
new_deaths_smoothed: skewness = 1.19
total_cases_per_million: skewness = 1.18
new_cases_per_million: skewness = 1.22
new_cases_smoothed_per_million: skewness = 1.23
total_deaths_per_million: skewness = 1.25
new_deaths_per_million: skewness = 1.22
new_deaths_smoothed_per_million: skewness = 1.23
reproduction_rate: skewness = -0.02
new_tests: skewness = 1.19
total_tests: skewness = 1.21
total_tests_per_thousand: skewness = 1.20
new_tests_per_thousand: skewness = 1.22
new_tests_smoothed: skewness = 1.21
new_tests_smoothed_per_thousand: skewness = -0.48
positive_rate: skewness = 1.23
tests_per_case: skewness = 1.21
new_vaccinations_smoothed: skewness = 1.19
new_vaccinations_smoothed_per_million: skewness = 1.26
new_people_vaccinated_smoothed: skewness = 1.19
new_people_vaccinated_smoothed_per_hundred: skewness = 1.22
stringency_index: skewness = -0.13
population: skewness = 1.23
population_density: skewness = 1.37
median_age: skewness = 0.10
aged_65_older: skewness = 0.98
aged_70_older: skewness = 1.07
gdp_per_capita: skewness = 0.98
extreme_poverty: skewness = 1.20
cardiovasc_death_rate: skewness = 0.53
diabetes_prevalence: skewness = 0.86
female_smokers: skewness = 0.78
male_smokers: skewness = 0.08
handwashing_facilities: skewness = 0.00
hospital_beds_per_thousand: skewness = 1.14
life_expectancy: skewness = -0.65
human_development_index: skewness = -0.39
```

“After capping outliers, the numeric distribution became symmetric — improving model stability.”

Transformation According To Skewness Range

* `-0.5 to 0.5` No action needed

* `0.5 to 1 / -0.5 to -1` Optional mild transformation

* `>1 / <-1` Recommended transformation (log, sqrt, Yeo-Johnson)

1️⃣ When to use log transformation

- Column has high positive skew `(long tail to the right)`
- Values are strictly positive `(log cannot take zero or negative numbers)`
- Useful for normalizing distributions for:
- Linear regression
- Correlation analysis
- Parametric tests

Log Transformation Of The Data

In [ ]:
```
# Select numeric columns
num_cols = df.select_dtypes(include=['int64', 'float64']).columns

# Transform columns with high skew (>1) using .loc
```

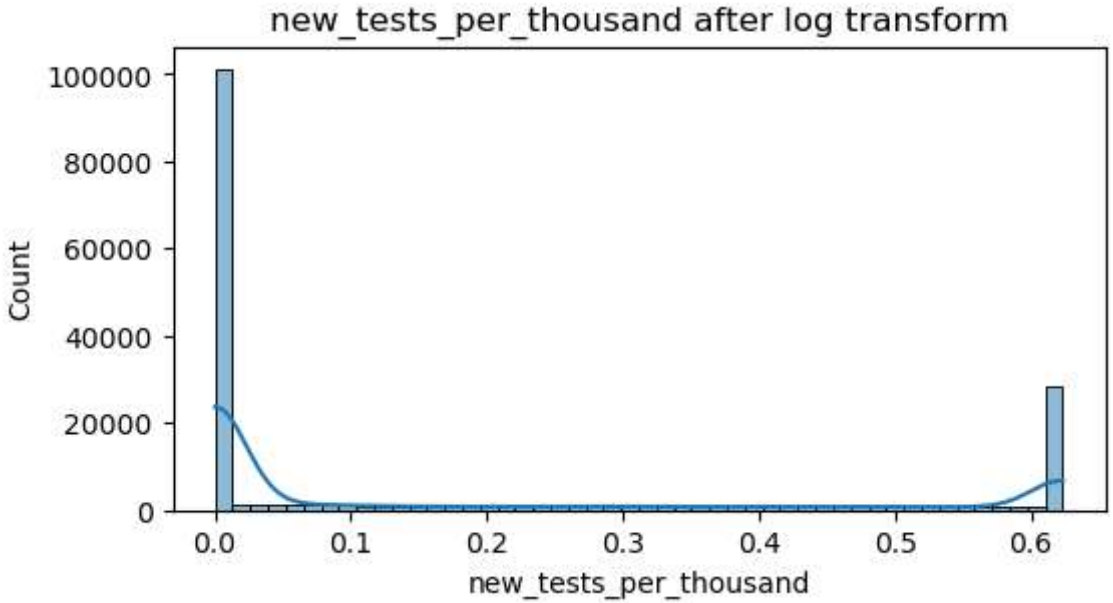
```
for col in num_cols:
    if df[col].skew() > 1:
        # Use .loc to overwrite original column safely
        df.loc[:, col] = np.log(df[col] + 1)
        print(f"{col} transformed in-place using log")
```

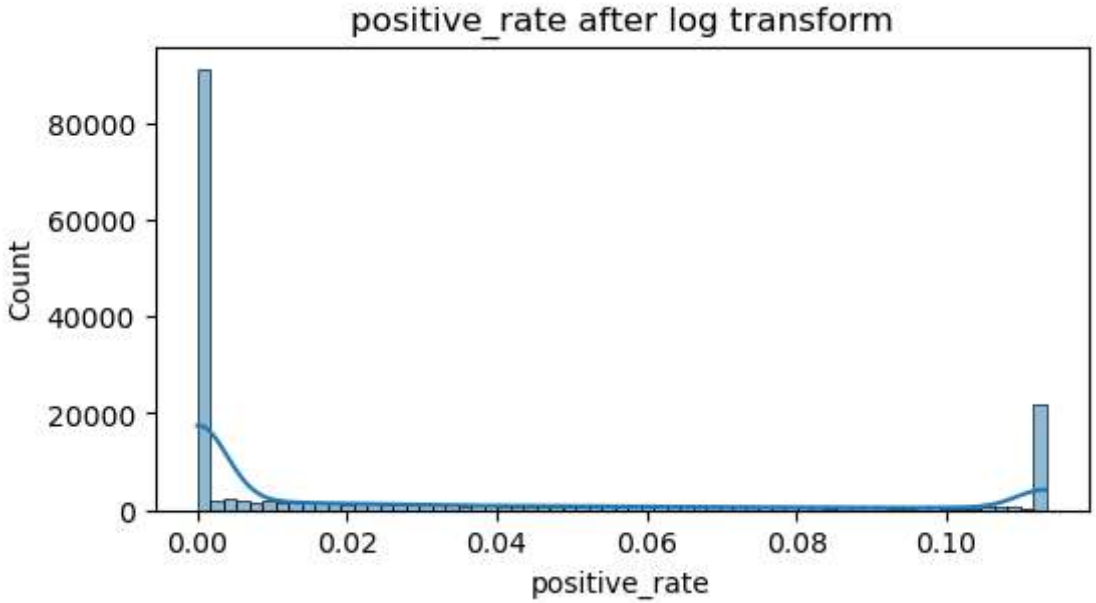
```
new_tests_per_thousand transformed in-place using log
positive_rate transformed in-place using log
new_people_vaccinated_smoothed_per_hundred transformed in-place using log
```

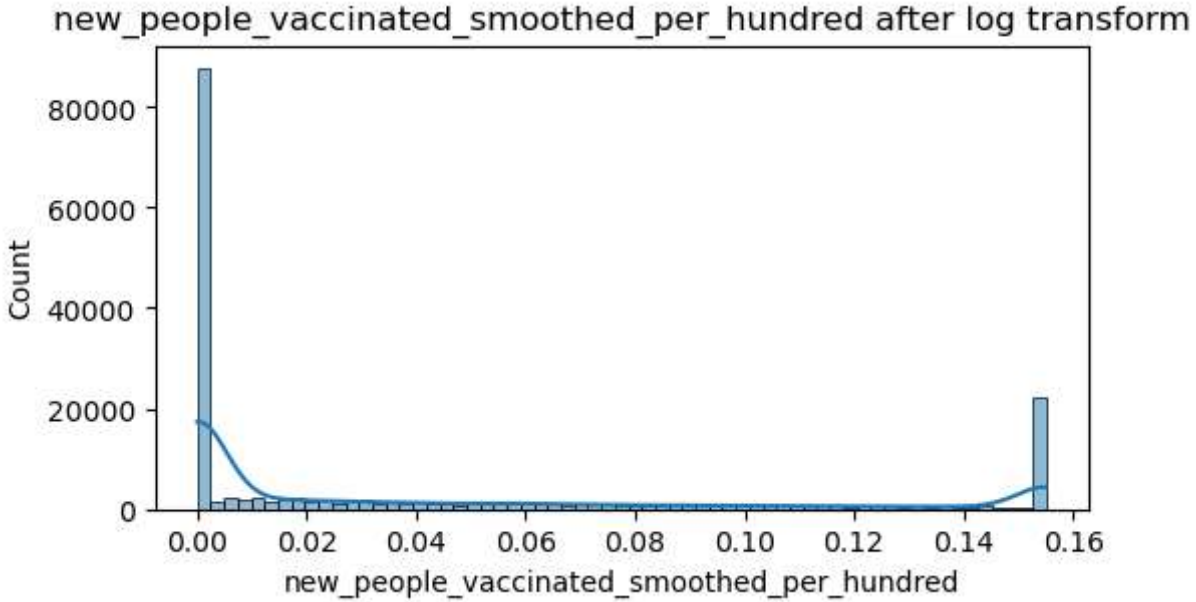
In [ ]:

Visualization After Log Transformation

In [ ]:
```
for col in num_cols:
    if df[col].skew() > 1:
        plt.figure(figsize=(6,3))
        sns.histplot(df[col], kde=True)
        plt.title(f"{col} after log transform")
        plt.show()
```




positive_rate after log transform
Count
80000
60000
40000
20000
0
0.00
0.02
0.04
0.06
0.08
0.10
positive_rate


new_people_vaccinated_smoothed_per_hundred after log transform
Count
80000
60000
40000
20000
0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
new_people_vaccinated_smoothed_per_hundred

In [ ]:

Check Skewness Again After Log Transformation (Compulsory)

In [ ]:
```
skewness = df[num_cols].skew()
print(skewness)
```

```
total_cases                                -0.788955
new_cases                                  -0.103399
new_cases_smoothed                         -0.194819
total_deaths                               -0.295031
new_deaths                                  0.596080
new_deaths_smoothed                         0.543660
total_cases_per_million                    -0.828341
new_cases_per_million                       0.123631
new_cases_smoothed_per_million             -0.003391
total_deaths_per_million                   -0.561445
new_deaths_per_million                      0.992087
new_deaths_smoothed_per_million             0.906948
reproduction_rate                          -0.024224
new_tests                                   0.508201
total_tests                                 0.428812
total_tests_per_thousand                    0.666956
new_tests_per_thousand                      1.029648
new_tests_smoothed                          0.112310
new_tests_smoothed_per_thousand            -0.483048
positive_rate                               1.187155
tests_per_case                              0.896031
new_vaccinations_smoothed                   0.240230
new_vaccinations_smoothed_per_million       0.161460
new_people_vaccinated_smoothed              0.333938
new_people_vaccinated_smoothed_per_hundred  1.155331
stringency_index                           -0.127122
population                                 -0.994514
population_density                         -0.296038
median_age                                  0.104370
aged_65_older                               0.977531
aged_70_older                               0.331730
gdp_per_capita                              0.978632
extreme_poverty                             0.921089
cardiovasc_death_rate                       0.532900
diabetes_prevalence                         0.862620
female_smokers                              0.780304
male_smokers                                0.075645
handwashing_facilities                      0.000000
hospital_beds_per_thousand                  0.480423
life_expectancy                            -0.651232
```

```
human_development_index                      -0.390354
dtype: float64
```

- **Confirm that no column has extreme skew (>1).**
- **Small skew (-0.5 to 0.5) is fine; slight skew (0.5–1) can usually be left as-is.**

Final Checks

> After All The Transformation

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 166326 entries, 0 to 166325
Data columns (total 46 columns):
 #   Column                                      Non-Null Count   Dtype
---  ------                                      --------------   -----
 0   iso_code                                    166326 non-null  object
 1   continent                                   166326 non-null  object
 2   location                                    166326 non-null  object
 3   date                                        166326 non-null  datetime64[ns]
 4   total_cases                                 166326 non-null  float64
 5   new_cases                                   166326 non-null  float64
 6   new_cases_smoothed                          166326 non-null  float64
 7   total_deaths                                166326 non-null  float64
 8   new_deaths                                  166326 non-null  float64
 9   new_deaths_smoothed                         166326 non-null  float64
 10  total_cases_per_million                     166326 non-null  float64
 11  new_cases_per_million                       166326 non-null  float64
 12  new_cases_smoothed_per_million              166326 non-null  float64
 13  total_deaths_per_million                    166326 non-null  float64
 14  new_deaths_per_million                      166326 non-null  float64
 15  new_deaths_smoothed_per_million             166326 non-null  float64
 16  reproduction_rate                           166326 non-null  float64
 17  new_tests                                   166326 non-null  float64
 18  total_tests                                 166326 non-null  float64
 19  total_tests_per_thousand                    166326 non-null  float64
 20  new_tests_per_thousand                      166326 non-null  float64
 21  new_tests_smoothed                          166326 non-null  float64
 22  new_tests_smoothed_per_thousand             166326 non-null  float64
 23  positive_rate                               166326 non-null  float64
 24  tests_per_case                              166326 non-null  float64
 25  tests_units                                 166326 non-null  object
 26  new_vaccinations_smoothed                   166326 non-null  float64
 27  new_vaccinations_smoothed_per_million       166326 non-null  float64
 28  new_people_vaccinated_smoothed              166326 non-null  float64
 29  new_people_vaccinated_smoothed_per_hundred  166326 non-null  float64
 30  stringency_index                            166326 non-null  float64
 31  population                                  166326 non-null  float64
 32  population_density                          166326 non-null  float64
 33  median_age                                  166326 non-null  float64
 34  aged_65_older                               166326 non-null  float64
 35  aged_70_older                               166326 non-null  float64
```

```
 36  gdp_per_capita                          166326 non-null  float64
 37  extreme_poverty                         166326 non-null  float64
 38  cardiovasc_death_rate                   166326 non-null  float64
 39  diabetes_prevalence                     166326 non-null  float64
 40  female_smokers                          166326 non-null  float64
 41  male_smokers                            166326 non-null  float64
 42  handwashing_facilities                  166326 non-null  float64
 43  hospital_beds_per_thousand              166326 non-null  float64
 44  life_expectancy                         166326 non-null  float64
 45  human_development_index                 166326 non-null  float64
dtypes: datetime64[ns](1), float64(41), object(4)
memory usage: 58.4+ MB
```

In [ ]:

In [ ]:
```
print(df.head())
```

```
  iso_code continent     location       date  total_cases  new_cases  \
0      AFG      Asia  Afghanistan 2020-02-24     1.791759   1.791759
1      AFG      Asia  Afghanistan 2020-02-25     1.791759   0.000000
2      AFG      Asia  Afghanistan 2020-02-26     1.791759   0.000000
3      AFG      Asia  Afghanistan 2020-02-27     1.791759   0.000000
4      AFG      Asia  Afghanistan 2020-02-28     1.791759   0.000000

   new_cases_smoothed  total_deaths  new_deaths  new_deaths_smoothed  ...  \
0                 0.0      0.693147         0.0                  0.0  ...
1                 0.0      0.693147         0.0                  0.0  ...
2                 0.0      0.693147         0.0                  0.0  ...
3                 0.0      0.693147         0.0                  0.0  ...
4                 0.0      0.693147         0.0                  0.0  ...

   gdp_per_capita  extreme_poverty  cardiovasc_death_rate  \
0        1803.987         0.182322                472.898
1        1803.987         0.182322                472.898
2        1803.987         0.182322                472.898
3        1803.987         0.182322                472.898
4        1803.987         0.182322                472.898

   diabetes_prevalence  female_smokers  male_smokers  handwashing_facilities  \
0                 9.59             6.3          31.4                50.78871
1                 9.59             6.3          31.4                50.78871
2                 9.59             6.3          31.4                50.78871
3                 9.59             6.3          31.4                50.78871
4                 9.59             6.3          31.4                50.78871

   hospital_beds_per_thousand  life_expectancy  human_development_index
0                    0.405465            64.83                    0.511
1                    0.405465            64.83                    0.511
2                    0.405465            64.83                    0.511
3                    0.405465            64.83                    0.511
4                    0.405465            64.83                    0.511

[5 rows x 46 columns]
```

In [ ]:

In [ ]:
```
df.shape
```

Out[ ]: (166326, 46)

In [ ]:
```
df.isnull().sum()
```

Out[ ]:

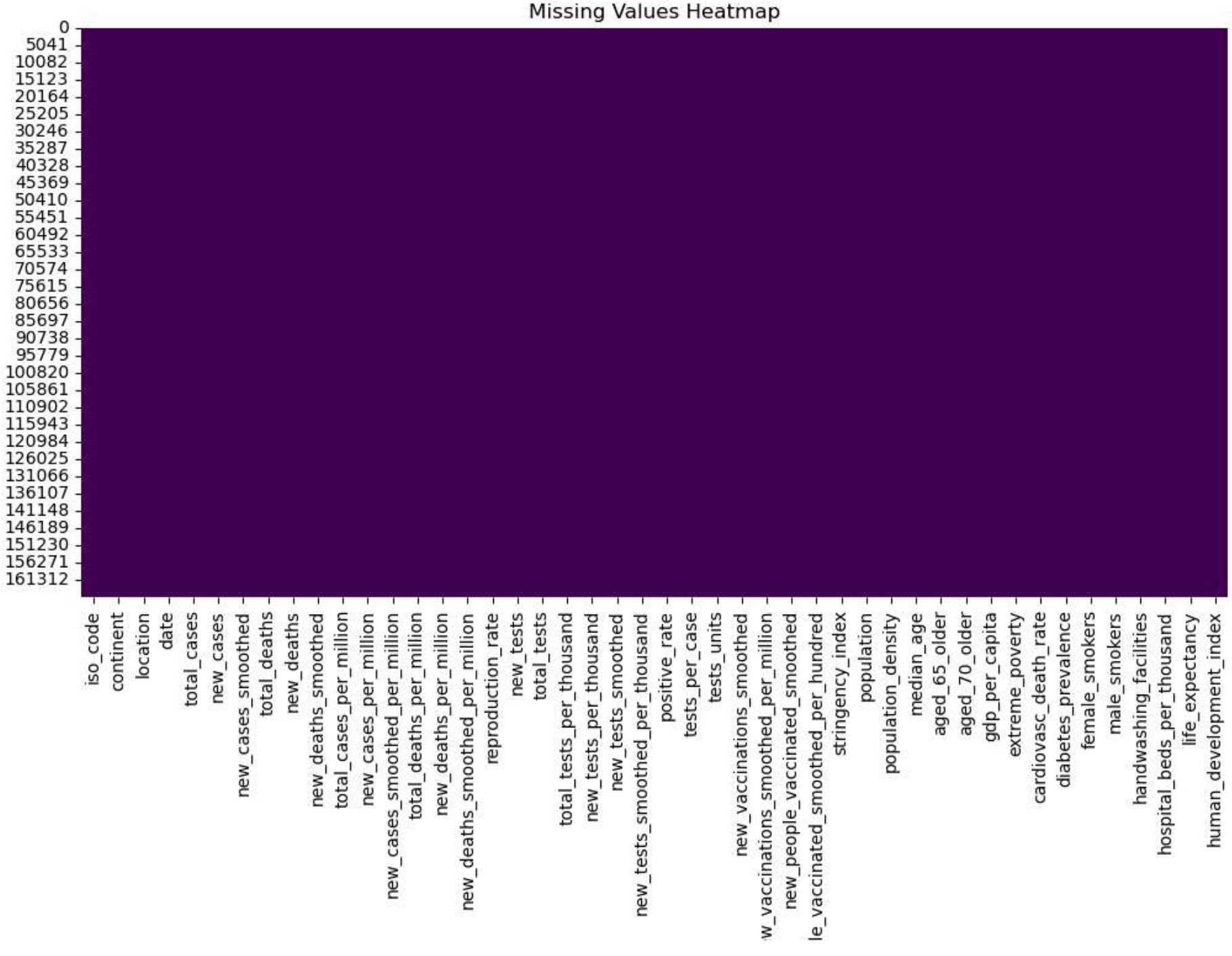
iso_code	0
continent	0
location	0
date	0
total_cases	0
new_cases	0
new_cases_smoothed	0
total_deaths	0
new_deaths	0
new_deaths_smoothed	0
total_cases_per_million	0
new_cases_per_million	0
new_cases_smoothed_per_million	0
total_deaths_per_million	0
new_deaths_per_million	0
new_deaths_smoothed_per_million	0
reproduction_rate	0
new_tests	0
total_tests	0
total_tests_per_thousand	0
new_tests_per_thousand	0
new_tests_smoothed	0
new_tests_smoothed_per_thousand	0
positive_rate	0
tests_per_case	0
tests_units	0
new_vaccinations_smoothed	0
new_vaccinations_smoothed_per_million	0
new_people_vaccinated_smoothed	0
new_people_vaccinated_smoothed_per_hundred	0
stringency_index	0
population	0
population_density	0
median_age	0
aged_65_older	0
aged_70_older	0
gdp_per_capita	0
extreme_poverty	0
cardiovasc_death_rate	0
diabetes_prevalence	0

```
female_smokers                  0
male_smokers                    0
handwashing_facilities          0
hospital_beds_per_thousand      0
life_expectancy                 0
human_development_index         0
dtype: int64
```

In [ ]:

Cleaned Missing Value Heatmap

In [ ]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Values Heatmap")
plt.show()
```


Missing Values Heatmap
0
5041
10082
15123
20164
25205
30246
35287
40328
45369
50410
55451
60492
65533
70574
75615
80656
85697
90738
95779
100820
105861
110902
115943
120984
126025
131066
136107
141148
146189
151230
156271
161312
iso_code
continent
location
date
total_cases
new_cases
new_cases_smoothed
total_deaths
new_deaths
new_deaths_smoothed
total_cases_per_million
new_cases_per_million
new_cases_smoothed_per_million
total_deaths_per_million
new_deaths_per_million
new_deaths_smoothed_per_million
reproduction_rate
new_tests
total_tests
total_tests_per_thousand
new_tests_per_thousand
new_tests_smoothed
new_tests_smoothed_per_thousand
positive_rate
tests_per_case
tests_units
new_vaccinations_smoothed
w_vaccinations_smoothed_per_million
new_people_vaccinated_smoothed
le_vaccinated_smoothed_per_hundred
stringency_index
population
population_density
median_age
aged_65_older
aged_70_older
gdp_per_capita
extreme_poverty
cardiovasc_death_rate
diabetes_prevalence
female_smokers
male_smokers
handwashing_facilities
hospital_beds_per_thousand
life_expectancy
human_development_index

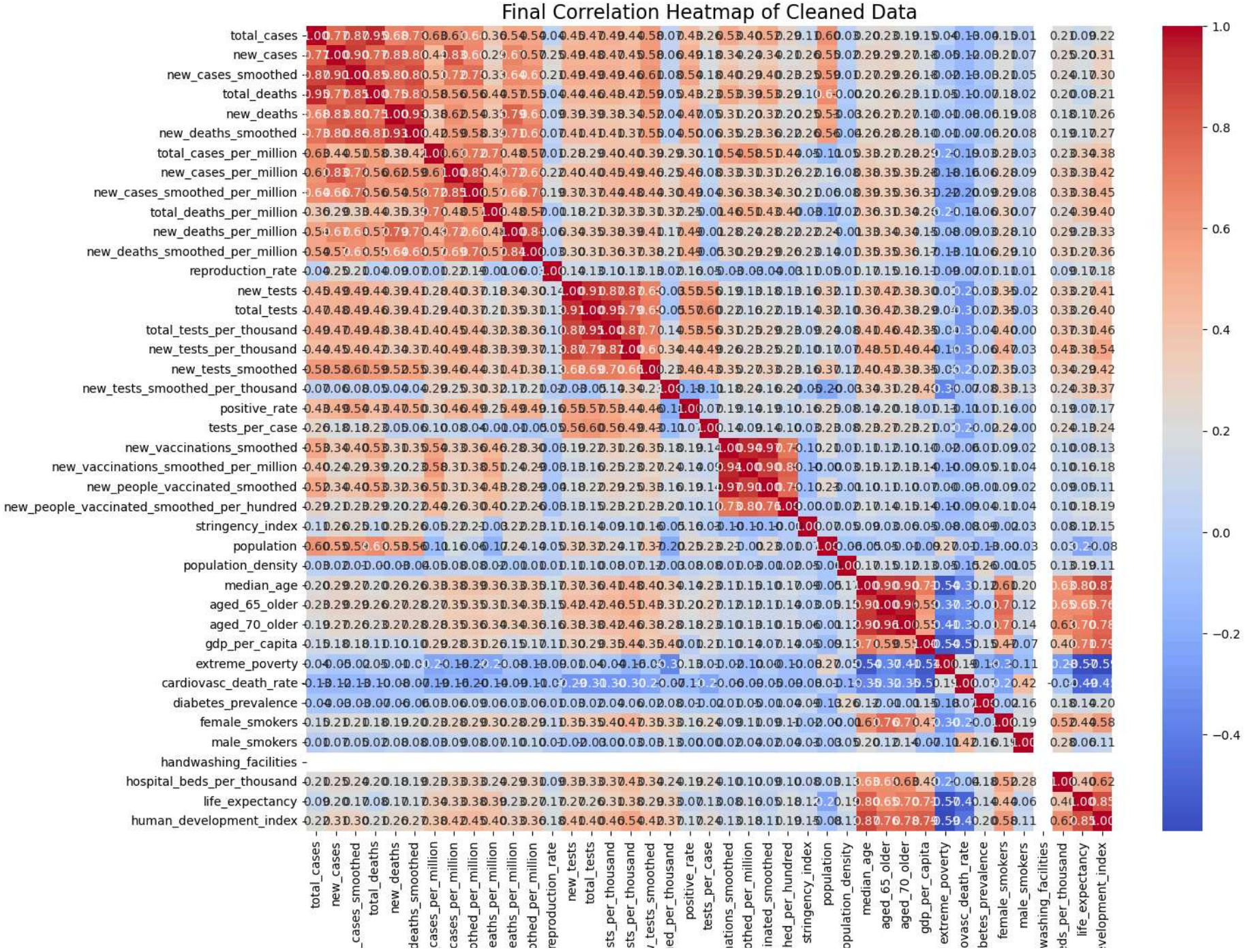
ne

new_peopl

In [ ]:

Final Correlation Heatmap of Cleaned Data

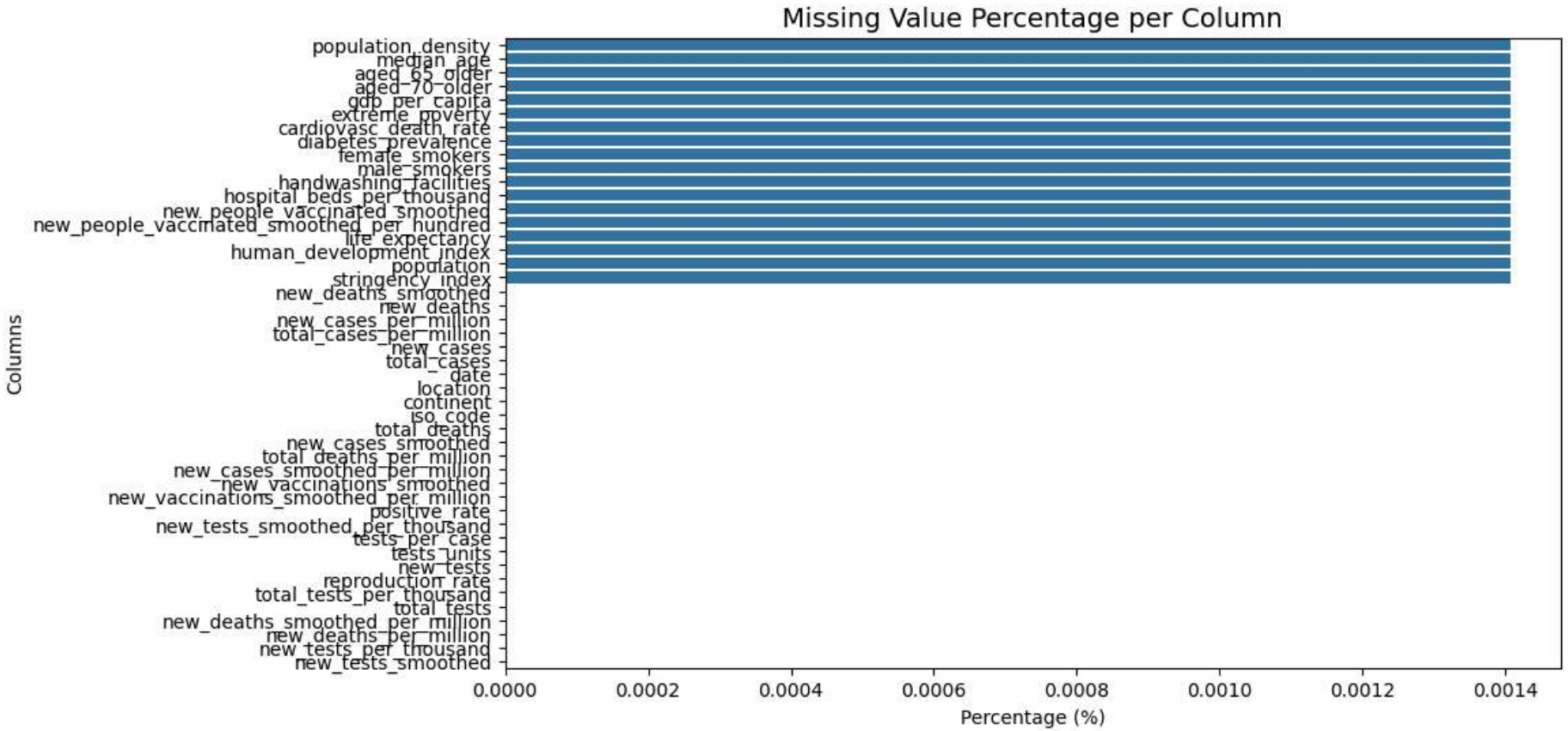
In [ ]:
```
corr_matrix = df[num_cols].corr()

# Plot heatmap
plt.figure(figsize=(15,12))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True, square=True)
plt.title("Final Correlation Heatmap of Cleaned Data", fontsize=16)
plt.show()
```


Final Correlation Heatmap of Cleaned Data
total_cases
new_cases
new_cases_smoothed
total_deaths
new_deaths
new_deaths_smoothed
total_cases_per_million
new_cases_per_million
new_cases_smoothed_per_million
total_deaths_per_million
new_deaths_per_million
new_deaths_smoothed_per_million
reproduction_rate
new_tests
total_tests
total_tests_per_thousand
new_tests_per_thousand
new_tests_smoothed
new_tests_smoothed_per_thousand
positive_rate
tests_per_case
new_vaccinations_smoothed
new_vaccinations_smoothed_per_million
new_people_vaccinated_smoothed
new_people_vaccinated_smoothed_per_hundred
stringency_index
population
population_density
median_age
aged_65_older
aged_70_older
gdp_per_capita
extreme_poverty
cardiovasc_death_rate
diabetes_prevalence
female_smokers
male_smokers
handwashing_facilities
hospital_beds_per_thousand
life_expectancy
human_development_index
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

In [ ]:

Check Columns Missing % After Cleaning The Data

In [ ]:
```
missing_percent = df.isnull().mean() * 100

# Sort descending for clarity
missing_percent = missing_percent.sort_values(ascending=False)

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x=missing_percent.values, y=missing_percent.index)
plt.title("Missing Value Percentage per Column", fontsize=14)
plt.xlabel("Percentage (%)")
plt.ylabel("Columns")
plt.show()
```



`Very Minor (0.0014%)`

> "The dataset has less than `0.001%` missing values, which are statistically insignificant and do not impact overall analysis."

Save the cleaned DataFrame to CSV

- Use pandas `.to_csv()` to save your DataFrame

In [ ]:
```
df.to_csv('cleaned_covid_data.csv', index=False)
```

Cleaned CSV File Ready To Export In SQL For Business Queries & POWER BI For Visualization.

----------The End----------