

Milestone 1: Data Collection, Preprocessing, and Exploratory Data Analysis (EDA)

- **Member:**

Yuxi Lyu (67468519)

- **Objective of the project**

The goal of this project is to analyze the global impact of COVID-19 using a data-driven approach. This includes developing an interactive dashboard to visualize patterns in selected datasets, geospatial data, time series forecasts, and key statistics. Enable users to dynamically explore COVID-19 trends through interactive charts, maps, and statistical summaries. Provide real-time insights into the evolution of the outbreak. Understand the spread and trends of COVID-19 across different countries and time periods. Identify key factors influencing the outbreak, such as case growth, recovery rates, and mortality rates. Forecast future COVID-19 trends using time series forecasting techniques.

- **Type of tool**

The project involves:

Data exploration and preprocessing: cleaning, handling missing values, and detecting anomalies.

Statistical analysis: identifying patterns, correlations, and distributions.

Data visualization: creating interactive charts, heat maps, trend analysis, and geographic maps.

Predictive models: implementing machine learning models to predict COVID-19 trends.

Geospatial analysis: mapping the global spread of COVID-19.

Interactive dashboards: providing user-friendly visualization tools for exploring pandemic data in real time. The tool will provide interactive dashboards and statistical insights to visualize and analyze the global impact of the COVID-19 pandemic.

● tech stack

Programming languages and libraries: Python, Pandas, NumPy, Matplotlib, Seaborn, Dash, Scipy, Scikit-learn and so on.

Data processing and storage: Jupyter Notebook, SQLite, CSV.

Visualization and reporting: Matplotlib, Seaborn, Plotly, Dash, etc.

Machine learning and forecasting: ARIMA, SARIMA, LSTM (optional), K-Means clustering, Random Forest/XGBoost (optional).

● Data to be used.

The following datasets have been utilized for this analysis:

Full Grouped Dataset(full_grouped.csv): Time-series data of confirmed cases, recoveries, deaths, and active cases worldwide.

```
In [124]: print(df_full_grouped.head(), "\n")
```

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	\
0	2020-01-22	Afghanistan	0	0	0	0	0	
1	2020-01-22	Albania	0	0	0	0	0	
2	2020-01-22	Algeria	0	0	0	0	0	
3	2020-01-22	Andorra	0	0	0	0	0	
4	2020-01-22	Angola	0	0	0	0	0	

	New deaths	New recovered	WHO Region
0	0	0	Eastern Mediterranean
1	0	0	Europe
2	0	0	Africa
3	0	0	Europe
4	0	0	Africa

```
In [106]: df_full_grouped.describe()
```

Out[106]:

	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered
count	3.515600e+04	35156.000000	3.515600e+04	3.515600e+04	35156.000000	35156.000000	35156.000000
mean	2.356663e+04	1234.068239	1.104813e+04	1.128443e+04	469.36375	18.603339	269.315593
std	1.499818e+05	7437.238354	6.454640e+04	8.997149e+04	3005.86754	115.706351	2068.063852
min	0.000000e+00	0.000000	0.000000e+00	-2.000000e+00	0.000000	-1918.000000	-16298.000000
25%	1.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000
50%	2.500000e+02	4.000000	3.300000e+01	8.500000e+01	2.000000	0.000000	0.000000
75%	3.640250e+03	78.250000	1.286250e+03	1.454000e+03	75.000000	1.000000	20.000000
max	4.290259e+06	148011.000000	1.846641e+06	2.816444e+06	77255.000000	3887.000000	140050.000000

COVID-19 Clean Complete Dataset(covid_19_clean_complete.csv): Comprehensive dataset including country-level statistics with additional geographic information (latitude, longitude).

Worldometer(worldometer_data.csv) Dataset: A snapshot of country-level statistics such as total cases, deaths, recovered cases, active cases, and per capita statistics.

```
: print(df_worldometer_data.head(), "\n")
```

```
Country/Region  Continent  Population  TotalCases  NewCases  \
0      USA      North America  3.311981e+08    5032179      NaN
1    Brazil  South America  2.127107e+08    2917562      NaN
2     India      Asia  1.381345e+09    2025409      NaN
3    Russia      Europe  1.459409e+08     871894      NaN
4  South Africa      Africa  5.938157e+07     538184      NaN

TotalDeaths  NewDeaths  TotalRecovered  NewRecovered  ActiveCases  \
0    162804.0      NaN    2576668.0      NaN    2292707.0
1    98644.0      NaN    2047660.0      NaN    771258.0
2    41638.0      NaN    1377384.0      NaN    606387.0
3    14606.0      NaN    676357.0      NaN    180931.0
4     9604.0      NaN    387316.0      NaN    141264.0

Serious,Critical  Tot Cases/1M pop  Deaths/1M pop  TotalTests  \
0      18296.0      15194.0      492.0    63139605.0
1      8318.0      13716.0      464.0    13206188.0
2      8944.0      1466.0      30.0    22149351.0
3      2300.0      5974.0      100.0    29716907.0
4       539.0      9063.0      162.0    3149807.0

Tests/1M pop  WHO Region
0    190640.0      Americas
1     62085.0      Americas
2    16035.0  South-EastAsia
3    203623.0      Europe
4    53044.0      Africa
```

```
: df_worldometer_data.describe()
```

```
Population  TotalCases  NewCases  TotalDeaths  NewDeaths  TotalRecovered  NewRecovered  ActiveCases  Serious,Critical  Tot
Cases/1M
pop
count  2.080000e+02  2.090000e+02  4.000000  188.000000  3.000000  2.050000e+02  3.000000  2.050000e+02  122.000000  208.000000
mean  3.041549e+07  9.171850e+04  1980.500000  3792.590426  300.000000  5.887898e+04  1706.000000  2.766433e+04  534.393443  3196.024038
std  1.047661e+08  4.325867e+05  3129.611424  15487.184877  451.199512  2.566984e+05  2154.779803  1.746327e+05  2047.518613  5191.986457
min  8.010000e+02  1.000000e+01  20.000000  1.000000  1.000000  7.000000e+00  42.000000  0.000000e+00  1.000000  3.000000
25%  9.663140e+05  7.120000e+02  27.500000  22.000000  40.500000  3.340000e+02  489.000000  8.600000e+01  3.250000  282.000000
50%  7.041972e+06  4.491000e+03  656.000000  113.000000  80.000000  2.178000e+03  936.000000  8.990000e+02  27.500000  1015.000000
75%  2.575614e+07  3.689600e+04  2609.000000  786.000000  449.500000  2.055300e+04  2538.000000  7.124000e+03  160.250000  3841.750000
max  1.381345e+09  5.032179e+06  6590.000000  162804.000000  819.000000  2.576668e+06  4140.000000  2.292707e+06  18296.000000  39922.000000

mean  21.433730  23.528236  1.688490e+04  884.179160  7.915713e+03  8.085012e+03
std  24.950320  70.442740  1.273002e+05  6313.584411  5.480092e+04  7.625890e+04
min  -51.796300  -135.000000  0.000000e+00  0.000000  0.000000e+00  -1.400000e+01
25%  7.873054  -15.310100  4.000000e+00  0.000000  0.000000e+00  0.000000e+00
50%  23.634500  21.745300  1.680000e+02  2.000000  2.900000e+01  2.600000e+01
75%  41.204380  80.771797  1.518250e+03  30.000000  6.660000e+02  6.060000e+02
max  71.706900  178.065000  4.290259e+06  148011.000000  1.846641e+06  2.816444e+06
```

● **Project timeline**

Project Timeline

Milestone 1: Data Preparation (Due by February 21)

Data Collection

Data Preprocessing

Exploratory Data Analysis (EDA)

Milestone 2: Model Development

Feature Engineering (Due by March 5)

Feature Selection (Due by March 10)

Data Modeling (Due by March 15)

Report Writing (Due by March 17)

Uploading to GitHub (Due by March 20)

Milestone 3: Evaluation & Finalization

Evaluation and Interpretation (Due by March 30)

Tool Development (Due by April 10)

Report Writing (Due by April 13)

Uploading to GitHub (Due by April 15)

4-Minute Presentation (Due by April 20)

● **Data Preprocessing**

The data preprocessing stage involved multiple steps to ensure data integrity, consistency, and usability for analysis. First, we examined the structure of each dataset, identifying missing values and data types. Missing categorical values, such as

Province/State, Continent, and WHO Region, were replaced with "Unknown", while missing numerical values, including Population, TotalTests, and NewCases, were set to 0 to maintain uniformity across all records. A pre- and post-cleaning missing value check was conducted to confirm data completeness.

Handle missing values:

```
# Define a function to check for missing values
def check_missing_values(df, name):
    print(f"\nChecking Missing Values for {name} dataset:")
    missing_data = df.isnull().sum()
    missing_data = missing_data[missing_data > 0]

    if not missing_data.empty:
        print(missing_data)
    else:
        print("No Missing Values Found.")
    print("\n" + "-" * 50)

# Reload the dataset to ensure integrity before cleaning
df_full_grouped_clean = df_full_grouped.copy()
df_covid_19_clean_clean = df_covid_19_clean.copy()
df_worldometer_data_clean = df_worldometer_data.copy()

# Check for initial missing values
check_missing_values(df_full_grouped_clean, "Full Grouped")
check_missing_values(df_covid_19_clean_clean, "COVID-19 Clean Complete")
check_missing_values(df_worldometer_data_clean, "Worldometer Data")
```

Checking Missing Values for Full Grouped dataset:
No Missing Values Found.

Checking Missing Values for COVID-19 Clean Complete dataset:
Province/State 34404
dtype: int64

Checking Missing Values for Worldometer Data dataset:

Continent	1
Population	1
NewCases	205
TotalDeaths	21
NewDeaths	206
TotalRecovered	4
NewRecovered	206
ActiveCases	4
Serious,Critical	87
Tot Cases/1M pop	1
Deaths/1M pop	22
TotalTests	18
Tests/1M pop	18
WHO Region	25

dtype: int64

```

: # Handling missing values **in the covid_19_clean_complete dataset
df_covid_19_clean_clean['Province/State'] = df_covid_19_clean_clean['Province/State'].fillna('Unknown')

# Handling missing values **in the worldometer_data dataset
df_worldometer_data_clean['Continent'] = df_worldometer_data_clean['Continent'].fillna('Unknown')
df_worldometer_data_clean['Population'] = df_worldometer_data_clean['Population'].fillna(0)

# Fill missing numeric columns with 0
df_worldometer_data_clean.fillna({
    'NewCases': 0,
    'TotalDeaths': 0,
    'NewDeaths': 0,
    'TotalRecovered': 0,
    'NewRecovered': 0,
    'ActiveCases': 0,
    'Serious,Critical': 0,
    'Tot Cases/1M pop': 0,
    'Deaths/1M pop': 0,
    'TotalTests': 0,
    'Tests/1M pop': 0
}, inplace=True)

# Fill missing WHO Regions with "Unknown"
df_worldometer_data_clean['WHO Region'] = df_worldometer_data_clean['WHO Region'].fillna('Unknown')

# Recheck the missing values **after cleaning
print("\nPost-Cleaning Missing Value Check:")
check_missing_values(df_full_grouped_clean, "Full Grouped")
check_missing_values(df_covid_19_clean_clean, "COVID-19 Clean Complete")
check_missing_values(df_worldometer_data_clean, "Worldometer Data")

```

Post-Cleaning Missing Value Check:

Checking Missing Values for Full Grouped dataset:
No Missing Values Found.

Checking Missing Values for COVID-19 Clean Complete dataset:
No Missing Values Found.

Checking Missing Values for Worldometer Data dataset:
No Missing Values Found.

To handle outliers, different methods were applied based on dataset characteristics. The Interquartile Range (IQR) method was used for Worldometer to remove extreme values from TotalCases, TotalDeaths, TotalRecovered, and related features. Following outlier removal, numerical features were processed using Z-score standardization, ensuring all variables had a mean of 0 and a standard deviation of 1. Additionally, certain features in Worldometer, such as Tot Cases/1M pop, Deaths/1M pop, and Tests/1M pop, were Min-Max normalized to scale values between 0 and 1, making them more interpretable and comparable across different countries. These preprocessing steps ensure the dataset is clean, properly scaled, and ready for further statistical analysis, visualization, and modeling.

Handle outliers:

```
# Use IQR method to handle outliers
def remove_outliers_iqr(df, columns):
    Q1 = df[columns].quantile(0.25)
    Q3 = df[columns].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[~((df[columns] < lower_bound) | (df[columns] > upper_bound)).any(axis=1)]

iqr_columns_worldometer = ["TotalCases", "TotalDeaths", "TotalRecovered", "ActiveCases",
                           "Tot Cases/1M pop", "Deaths/1M pop", "TotalTests", "Tests/1M pop"]

# Reload Worldometer data to avoid accidental deletion of data in previous cleanup
df_worldometer_data_clean = df_worldometer_data.copy()

# use IQR method to handle outliers
df_worldometer_data_clean = remove_outliers_iqr(df_worldometer_data_clean, iqr_columns_worldometer)

print(f"Worldometer Data dataset IQR processed size: {df_worldometer_data_clean.shape}")

Worldometer Data dataset IQR processed size: (140, 16)
```

Normalize or scale features as needed for analysis (e.g., Min-Max scaling, standardization):

```

# Data after handling outliers (Z-score standardization & IQR method)
df_full_grouped_clean = remove_outliers_zscore(df_full_grouped_clean, numerical_cols_full_grouped, threshold=4)
df_covid_19_clean_clean = remove_outliers_zscore(df_covid_19_clean_clean, numerical_cols_covid_19, threshold=4)
df_worldometer_data_clean = remove_outliers_iqr(df_worldometer_data_clean, iqr_columns_worldometer)

# Define the standardization (Z-score) and normalization (Min-Max) scalers
standard_scaler = StandardScaler()
min_max_scaler = MinMaxScaler()

# Columns to be standardized (Z-score)
zscore_columns_full_grouped = ["Confirmed", "Deaths", "Recovered", "Active", "New cases", "New deaths", "New recover"]
zscore_columns_covid_19 = ["Confirmed", "Deaths", "Recovered", "Active"]
zscore_columns_worldometer = ["TotalCases", "TotalDeaths", "TotalRecovered", "ActiveCases"]

# Columns to be normalized (Min-Max)
minmax_columns_worldometer = ["Tot Cases/1M pop", "Deaths/1M pop", "TotalTests", "Tests/1M pop"]

# Z-score normalization for selected columns
df_full_grouped_clean[zscore_columns_full_grouped] = standard_scaler.fit_transform(df_full_grouped_clean[zscore_columns_full_grouped])
df_covid_19_clean_clean[zscore_columns_covid_19] = standard_scaler.fit_transform(df_covid_19_clean_clean[zscore_columns_covid_19])
df_worldometer_data_clean[zscore_columns_worldometer] = standard_scaler.fit_transform(df_worldometer_data_clean[zscore_columns_worldometer])

# Min-Max normalization for selected columns
df_worldometer_data_clean[minmax_columns_worldometer] = min_max_scaler.fit_transform(df_worldometer_data_clean[minmax_columns_worldometer])

print("The complete grouped dataset after normalization (first 5 rows): ")
print(df_full_grouped_clean.head(), "\n")

print("Normalized COVID-19 Clean Complete dataset (first 5 rows): ")
print(df_covid_19_clean_clean.head(), "\n")

print("Standardized and normalized Worldometer dataset (first 5 rows): ")
print(df_worldometer_data_clean.head(), "\n")

```

The complete grouped dataset after normalization (first 5 rows):

	Date	Country/Region	Confirmed	Deaths	Recovered	Active
0	2020-01-22	Afghanistan	-0.373005	-0.241166	-0.299827	-0.344399
1	2020-01-22	Albania	-0.373005	-0.241166	-0.299827	-0.344399
2	2020-01-22	Algeria	-0.373005	-0.241166	-0.299827	-0.344399
3	2020-01-22	Andorra	-0.373005	-0.241166	-0.299827	-0.344399
4	2020-01-22	Angola	-0.373005	-0.241166	-0.299827	-0.344399

	New cases	New deaths	New recovered	WHO Region
0	-0.345265	-0.267631	-0.285039	Eastern Mediterranean
1	-0.345265	-0.267631	-0.285039	Europe
2	-0.345265	-0.267631	-0.285039	Africa
3	-0.345265	-0.267631	-0.285039	Europe
4	-0.345265	-0.267631	-0.285039	Africa

Normalized COVID-19 Clean Complete dataset (first 5 rows):

	Province/State	Country/Region	Lat	Long	Date	Confirmed
0	NaN	Afghanistan	33.93911	67.709953	2020-01-22	-0.335405
1	NaN	Albania	41.15330	20.168300	2020-01-22	-0.335405
2	NaN	Algeria	28.03390	1.659600	2020-01-22	-0.335405
3	NaN	Andorra	42.50630	1.521800	2020-01-22	-0.335405
4	NaN	Angola	-11.20270	17.873900	2020-01-22	-0.335405

	Deaths	Recovered	Active	WHO Region
0	-0.238463	-0.274485	-0.29683	Eastern Mediterranean
1	-0.238463	-0.274485	-0.29683	Europe
2	-0.238463	-0.274485	-0.29683	Africa
3	-0.238463	-0.274485	-0.29683	Europe
4	-0.238463	-0.274485	-0.29683	Africa

Standardized and normalized Worldometer dataset (first 5 rows):

	Country/Region	Continent	Population	TotalCases	NewCases	TotalDeaths
71	Ivory Coast	Africa	26437950.0	4.231974	NaN	0.451130
76	Bulgaria	Europe	6942854.0	3.190801	NaN	4.528782
77	Madagascar	Africa	27755708.0	3.042798	NaN	0.831875
80	Senegal	Africa	16783877.0	2.493551	NaN	1.924980
81	Norway	Europe	5425471.0	2.115356	NaN	2.330289

	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical
71	NaN	4.279582	NaN	2.808452	NaN
76	NaN	2.253235	NaN	4.021607	47.0
77	NaN	3.353252	NaN	1.350861	88.0
80	NaN	2.144978	NaN	2.385425	33.0
81	NaN	2.841312	NaN	-0.352970	3.0

	Tot Cases/1M pop	Deaths/1M pop	TotalTests	Tests/1M pop	WHO Region
71	0.242460	0.050962	0.165775	0.036948	Africa
76	0.732863	0.817993	0.467377	0.432947	Europe
77	0.175480	0.063963	0.073015	0.013354	Africa
80	0.248727	0.167967	0.181972	0.066667	Africa
81	0.682335	0.609984	0.751872	0.894859	Europe

● Merging dataset

```
1: # Merge `Full Grouped` and `COVID-19 Clean Complete` by `Country/Region` + `Date`
# Select the appropriate columns to merge
merge_columns = ["Date", "Country/Region", "Confirmed", "Deaths", "Recovered", "Active"]

# Ensure that the Date column format is consistent
df_full_grouped_clean["Date"] = pd.to_datetime(df_full_grouped_clean["Date"])
df_covid_19_clean_clean["Date"] = pd.to_datetime(df_covid_19_clean_clean["Date"])

# Merge by `Country/Region` + `Date`, using `outer join` to keep all data
df_merged = pd.merge(df_full_grouped_clean[merge_columns],
                    df_covid_19_clean_clean[merge_columns],
                    on=["Date", "Country/Region"],
                    how="outer",
                    suffixes=("_full", "_clean"))

# Handling missing values ** (some countries may only exist in one dataset)
df_merged.fillna(0, inplace=True)
df_worldometer_data_clean = df_worldometer_data_clean.drop_duplicates(subset=["Country/Region"])
df_worldometer_data_clean.drop(columns=["WHO Region"], inplace=True, errors="ignore")
df_final = pd.merge(df_merged, df_worldometer_data_clean, on="Country/Region", how="left")
df_final.fillna(0, inplace=True)
print(f"The size of the combined dataset: {df_final.shape}")
print(df_final.head())
```

The size of the combined dataset: (47436, 24)

	Date	Country/Region	Confirmed_full	Deaths_full	Recovered_full	\
0	2020-01-22	Afghanistan	-0.373005	-0.241166	-0.299827	
1	2020-01-22	Albania	-0.373005	-0.241166	-0.299827	
2	2020-01-22	Algeria	-0.373005	-0.241166	-0.299827	
3	2020-01-22	Andorra	-0.373005	-0.241166	-0.299827	
4	2020-01-22	Angola	-0.373005	-0.241166	-0.299827	

	Active_full	Confirmed_clean	Deaths_clean	Recovered_clean	Active_clean	\
0	-0.344399	-0.335405	-0.238463	-0.274485	-0.29683	
1	-0.344399	-0.335405	-0.238463	-0.274485	-0.29683	
2	-0.344399	-0.335405	-0.238463	-0.274485	-0.29683	
3	-0.344399	-0.335405	-0.238463	-0.274485	-0.29683	
4	-0.344399	-0.335405	-0.238463	-0.274485	-0.29683	

	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	\
0	0.000000	0.0	0.000000	0.0	0.000000	
1	1.495108	0.0	0.580209	0.0	1.737808	
2	0.000000	0.0	0.000000	0.0	0.000000	
3	0.000000	0.0	0.000000	0.0	0.000000	
4	-0.027871	0.0	-0.464688	0.0	0.137705	

	Serious,Critical	Tot Cases/1M pop	Deaths/1M pop	TotalTests	Tests/1M pop
0	0.0	0.000000	0.000000	0.000000	0.000000
1	23.0	0.817861	0.843994	0.061391	0.135911
2	0.0	0.000000	0.000000	0.000000	0.000000
3	0.0	0.000000	0.000000	0.000000	0.000000
4	20.0	0.016451	0.024961	0.102373	0.016417

```
df_merged.describe()
```

	Date	Confirmed_full	Deaths_full	Recovered_full	Active_full	Confirmed_clean	Deaths_clean	Recovered_clean	Active_clean
count	47436	47436.000000	47436.000000	47436.000000	47436.000000	4.743600e+04	4.743600e+04	4.743600e+04	4.743600e+04
mean	2020-04-22 20:23:58.907159040	0.638691	0.543689	0.833929	0.134637	4.793273e-18	9.586545e-17	-3.834618e-17	-1.917309e-17
min	2020-01-22 00:00:00	-0.373005	-0.241166	-0.299827	-0.344725	-3.354046e-01	-2.384631e-01	-2.744853e-01	-2.992062e-01
25%	2020-03-07 00:00:00	-0.372152	-0.241166	-0.299827	-0.343746	-3.351311e-01	-2.384631e-01	-2.744853e-01	-2.968301e-01
50%	2020-04-22 00:00:00	-0.304563	-0.222113	-0.282044	-0.286628	-3.220911e-01	-2.350432e-01	-2.709579e-01	-2.934358e-01
75%	2020-06-08 00:00:00	0.131045	0.000000	0.000000	0.000000	-2.251348e-01	-2.008449e-01	-1.889064e-01	-2.197780e-01
max	2020-07-27 00:00:00	9.285026	12.234827	11.309992	10.035665	9.859118e+00	1.046904e+01	1.104264e+01	9.032246e+00
std	NaN	2.034856	1.835322	2.718387	1.234747	9.984810e-01	9.984810e-01	9.984810e-01	9.984810e-01

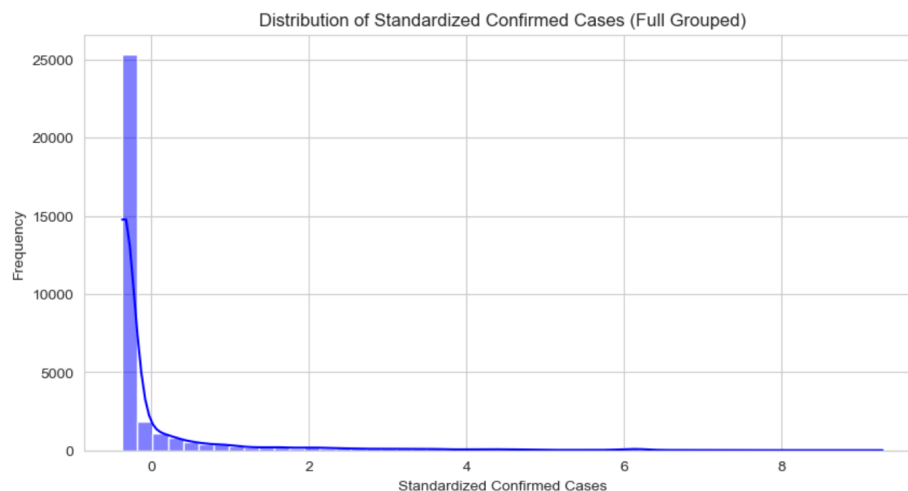
- **EDA report with key insights (charts, tables, visualizations, and observations).**

The histogram of standardized confirmed COVID-19 cases shows a highly right-skewed distribution, indicating that most countries have relatively low case counts, while a few have extremely high numbers. This long-tail effect suggests that the pandemic's impact is unevenly distributed globally, with certain countries experiencing significantly higher outbreaks. The Kernel Density Estimation (KDE) curve further highlights this pattern, reinforcing the presence of outliers. Given the severe skewness, applying a log transformation or analyzing high-case and low-case countries separately could provide deeper insights into the spread and trends of COVID-19.

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

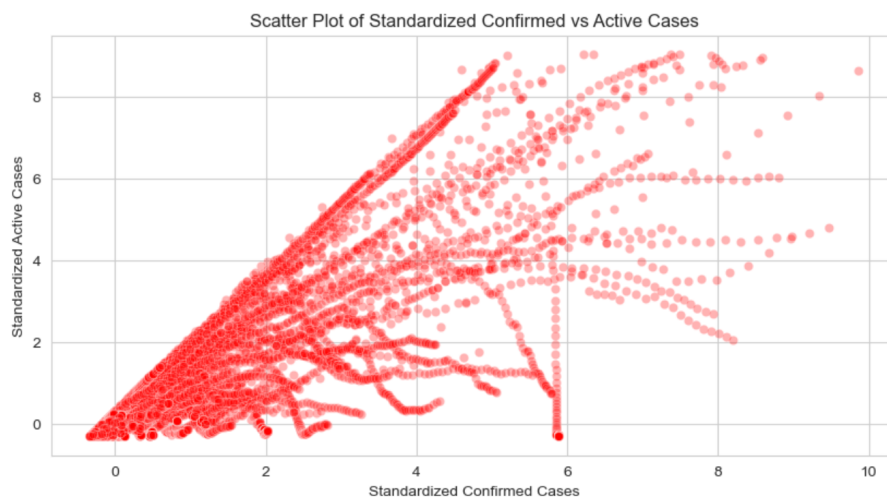
# Set drawing style
sns.set_style("whitegrid")

# 1. Histogram: Distribution of confirmed cases (Full Grouped dataset)
plt.figure(figsize=(10, 5))
sns.histplot(df_full_grouped_clean["Confirmed"], bins=50, kde=True, color="blue")
plt.xlabel("Standardized Confirmed Cases")
plt.ylabel("Frequency")
plt.title("Distribution of Standardized Confirmed Cases (Full Grouped)")
plt.show()
```



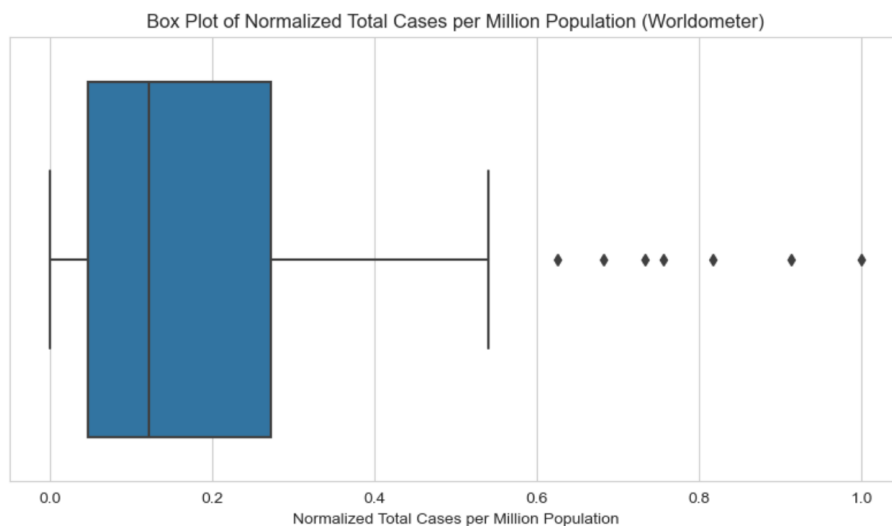
This scatter plot visualizes the relationship between standardized confirmed cases and standardized active cases using the COVID-19 Clean Complete dataset. Each red dot represents a data point corresponding to a specific country and date. The positive correlation observed indicates that as the number of confirmed cases increases, the number of active cases also tends to rise. However, the spread of points suggests variability, where some regions may have a higher proportion of recoveries or deaths, leading to fewer active cases despite high confirmed cases. The fanning-out effect seen in the plot may indicate differences in outbreak severity, government responses, or healthcare system effectiveness across different regions. Further analysis could involve segmentation by country or time-based trends to better understand pandemic dynamics.

```
# 2. Scatter plot: Active cases vs. confirmed cases (COVID-19 Clean Complete dataset)
plt.figure(figsize=(10, 5))
sns.scatterplot(x=df_covid_19_clean_clean["Confirmed"], y=df_covid_19_clean_clean["Active"], alpha=0.3, color="red")
plt.xlabel("Standardized Confirmed Cases")
plt.ylabel("Standardized Active Cases")
plt.title("Scatter Plot of Standardized Confirmed vs Active Cases")
plt.show()
```



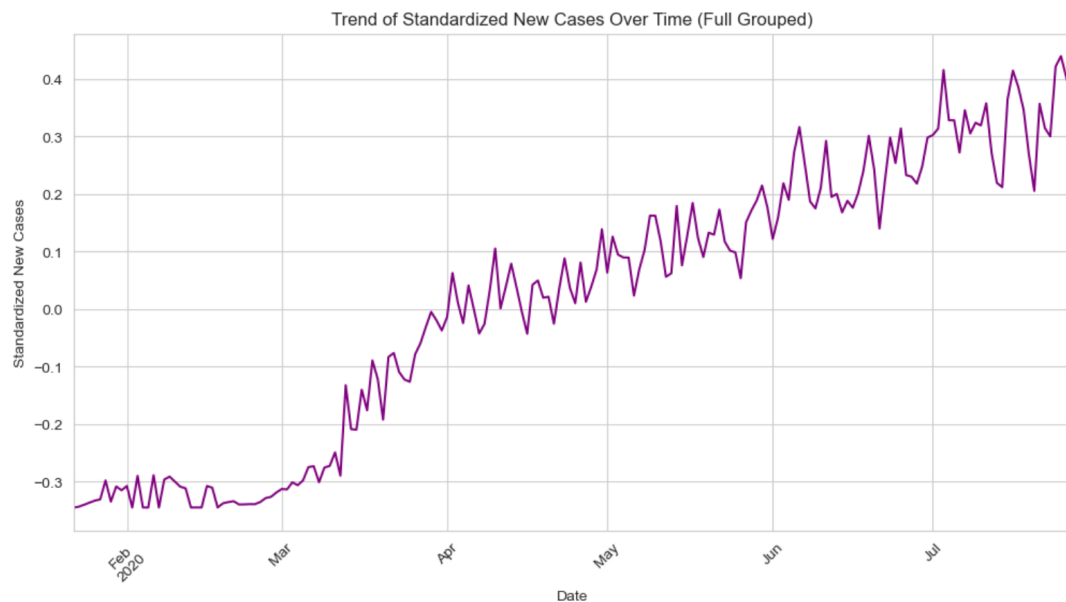
The box plot of normalized total COVID-19 cases per million population reveals a right-skewed distribution, indicating that most countries have relatively low case rates, while a few experience significantly higher infection levels. The presence of multiple outliers suggests that certain countries faced severe outbreaks or had extensive testing policies leading to higher reported cases per million. The wide interquartile range (IQR) reflects substantial variation in case distribution, likely influenced by population density, healthcare infrastructure, and government interventions. Further analysis of outlier countries could provide insights into the factors contributing to these discrepancies.

```
: # 3. Box plot: Distribution of cases per million population in each country (Worldometer dataset)
plt.figure(figsize=(10, 5))
sns.boxplot(x=df_worldometer_data_clean["Tot Cases/1M pop"])
plt.xlabel("Normalized Total Cases per Million Population")
plt.title("Box Plot of Normalized Total Cases per Million Population (Worldometer)")
plt.show()
```



The trend chart illustrates the standardized daily new COVID-19 cases over time, based on the Full Grouped dataset. The upward trajectory indicates a consistent increase in reported new cases, suggesting a worsening outbreak over the observed period. The initial rise in March and April 2020 aligns with the global spread of the pandemic, followed by fluctuations but an overall increasing trend until July. The short-term variations suggest periodic surges, potentially due to waves of infections, policy changes, or testing expansions. This trend highlights the need for further investigation into regional variations, intervention measures, and forecasting models to anticipate future outbreaks.

```
# 4. Trend chart: Daily new case trends (Full Grouped dataset)
plt.figure(figsize=(12, 6))
df_full_grouped_clean.groupby("Date")["New cases"].mean().plot(color="purple")
plt.xlabel("Date")
plt.ylabel("Standardized New Cases")
plt.title("Trend of Standardized New Cases Over Time (Full Grouped)")
plt.xticks(rotation=45)
plt.show()
```



When the three datasets are combined, the global COVID-19 trend graph illustrates the cumulative confirmed, death, and recovered cases over time. The confirmed cases (blue) show a continuous upward trajectory, indicating the sustained spread of the virus. Death cases (red) follow a slower but steady rise, while recovered cases (green) increase at a higher rate, suggesting effective recovery efforts. However, data fluctuations in early 2020 and sharp drops in May indicate potential reporting inconsistencies, data corrections, or delayed case adjustments. Further analysis is needed to examine daily new cases, country-specific trends, and data anomalies for a clearer understanding of the pandemic's evolution.

```
]: import matplotlib.pyplot as plt
```

```
# Aggregate data by date
df_global_trend = df_final.groupby("Date")[["Confirmed_full", "Deaths_full", "Recovered_full"]].sum()

plt.figure(figsize=(12, 6))
plt.plot(df_global_trend.index, df_global_trend["Confirmed_full"], label="Confirmed Cases", color="blue")
plt.plot(df_global_trend.index, df_global_trend["Deaths_full"], label="Deaths", color="red")
plt.plot(df_global_trend.index, df_global_trend["Recovered_full"], label="Recovered", color="green")

plt.xlabel("Date")
plt.ylabel("Number of Cases")
plt.title("Global Trend of COVID-19 Cases Over Time")
plt.legend()
plt.xticks(rotation=45)
plt.grid(True)

plt.show()
```

