PYTHON MEGA PROJECTS

MEGA-PROJECT-1:

PYTHON FOR COMPUTER SCIENCE AND DATA SCIENCE: AN INTRODUCTORY MEGA PROJECT

PROJECT TITLE:

2020 COVID-19 GLOBAL IMPACT VISUALIZER

PRESENTED BY: PRAMODH NARAIN

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

PROJECT PRESENTED YEAR: 2025

ACKNOWLEDGMENT

I express my deepest gratitude to my beloved parents for their unwavering support, encouragement, and sacrifices throughout my journey in learning and applying Python. Their belief in my abilities has been my greatest motivation, and this project is a testament to their continuous guidance and love. Without their encouragement, this project would not have been possible. I dedicate this work to them with immense respect and appreciation

PROJECT ABSTRACT

Python has emerged as one of the most versatile and powerful programming languages in modern computing. This project, ***"Python for Computer Science and Data Science: An Introductory Mega Project"***, serves as a foundational guide for understanding Python's significance in these domains. The project aims to provide an ***introductory yet comprehensive*** exploration of Python, starting from its core fundamentals to its practical applications in ***Computer Science*** and ***Data Science***.

The first section introduces ***Python as a programming language***, covering its ***history, features, and benefits***. Python's simplicity, readability, and extensive library support make it an ideal language for beginners and professional alike. A comparison with other programming languages highlights Python's advantaged in terms of code efficiency and ease of learning.

The second section explores ***Python's role in Computer Science***, illustrating its relevance in ***software development, artificial intelligence, algorithms, and automation***. Python's object-oriented nature, coupled with its extensive frameworks, makes it a preferred choice for backend development, game programming, and scripting. This section also introduces key concepts such as ***Data Structures, Recursion, and Algorithmic Problem-Solving*** using Python.

The third section focuses on ***Python in Data Science***, explaining the core ***principles of data science*** and how Python facilitates data-driven decision-making. Essential libraries like ***Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn*** are discussed, along with real-world applications such as ***data analysis, machine learning, and predictive modelling***. The project will include demonstrations of ***data preprocessing, exploratory data analysis (EDA), and visualization techniques*** to make concepts more tangible.

To ensure an ***interactive learning experience***, this project will include ***hands-on code implementation*** and ***simulated data sets*** for practical demonstration ***basic Python scripts***, perform ***data manipulations***, and visualize trends using graphical representations. Additionally, sample ***machine-learning models*** will be introduced to showcase Python's predictive capabilities.

By the end of this project, users will have a ***clear understanding of Python's capabilities*** in both ***Computer Science*** and ***Data Science.*** This introductory project serves as a ***stepping stone for further exploration*** into advanced areas such as ***deep learning, artificial intelligence, and big data analytics.*** With a focus on simplicity and clarity, this mega project is designed to be accessible to beginners while providing valuable insights for those looking to deepen their expertise in Python.

TABLE OF CONTENTS

1. INTRODUCTION

Python is one of the most popular programming languages in the world today, widely used in diverse domains such as web-development, artificial intelligence, data science, and automation. Its easy-to-learn syntax extensive libraries, and strong community support make it an excellent choice for both beginners and experienced developers. In this section, we will explore Python's evolution, its unique advantages, and a comparison with other prominent programming languages.

1.1 Python and Its Evolution:

Python was created by Guido van Rossum in the late 1980s and officially released in 1991. His goal was to design a programming language that emphasized code readability and simplicity while maintaining powerful capabilities. The name "Python" was inspired by the British comedy group Monty Python, reflecting van Rossum's goal of making programming more fun and accessible.

Python quickly gained popularity due to its clear syntax and ease of use. Over the years, multiple versions of Python have been released, each introducing new features and improvements. The most significant transition occurred with the release of Python 3 in 2008, which introduced several enhancements such as better Unicode support, improved memory management, and a more consistent syntax. While Python 2 was used for many years, Python 3 has become the standard, with official support for Python 2 ending in 2020.

Today, Python continues to evolve, with frequent updates enhancing its performance, security, and compatibility.

1.2 ***Why Python? Features and Advantages:***

Python is favoured by developers worldwide due to its rich set of features and numerous advantages. Some of the key reasons for Python's widespread adoption include:

- ***Easy-to-Read Syntax:*** Python's syntax is simple and resembles natural language, making it easier to learn and write code efficiently. Unlike languages like C++ or Java, Python does not require extensive boilerplate code, reducing the time required to develop applications.
- ***Extensive Standard Libraries:*** Python comes with a vast collection of built-in libraries that provide pre-written functions for various tasks such as data manipulation, wed development, machine learning, and automation. This eliminates the need for writing complex code from Scratch.
- ***Cross-Platform Support:*** Python is platform-independent language, meaning that a Python program written on one operating system (Windows, macOS, or Linux) can run on another without major modifications.
- ***Strong Community Support:*** Python has a large active community of developers who continuously contribute to its development. This means users have access to extensive documentation, tutorials, and third-party libraries that further enhance Python's capabilities.
- ***Versatility Across Multiple Domains:*** Python is widely used in various fields, including web-development, artificial intelligence, data science, cybersecurity, game development, and automation. Its adaptability makes it a preferred choice for modern software applications.

1.3 Comparison with Other Programming Languages:

Python stands out when compared to other popular programming languages such as Java, C++, and JavaScript. Here are some key differences:

- ***More Concise and Readable Code:*** Python allows developers to write fewer lines of code compared to Java or C++. Its syntax is designed to be intuitive, reducing the complexity of writing and maintaining programs.
- ***Higher-Level Abstraction:*** Python provides built-in functions and dynamic typing, allowing developers to focus on problem-solving rather than dealing with low-level programming details such as memory allocation.
- ***Simplified Memory Management:*** Unlike C++, which requires manual memory management, Python includes automatic memory management with garbage collection. This feature helps developers avoid memory leaks and improve program stability.
- ***Versatility:*** While JavaScript is mainly used for web-development and C++ is known for system programming, Python is a general-purpose language that seamlessly integrates into multiple domains, including AI, machine learning, and scientific computing.

Overall, Python's ease of use, flexibility, and powerful features make it one of the most sought-after programming languages today. Its continuous evolution ensures that it remains in the ever-changing landscape of technology.

2. PYTHON FOR COMPUTER SCIENCE

Python plays a significant role in various areas of computer science, ranging from software development to artificial intelligence and data structures. Due to its simplicity and rich ecosystem, Python has become a preferred choice for solving complex problems efficiently.

2.1 Python in Software Development:

Python is widely used in software development, providing frameworks and libraries that simplify application development. Some of its key applications include:

- ***Web-Development:*** Python offers powerful web frameworks such as Django and Flask that enable developers to build scalable and secure web applications.
- ***Desktop Applications:*** Python supports GUI (Graphical User Interface) Development using frameworks like Tkinter, PyQt, and Kivy making it possible to create cross-platform desktop applications.
- ***Game Development:*** Libraries such as Pygame facilitate game development, allowing developers to create 2D and simple 3D games with ease.

2.2 ***Python in Artificial Intelligence:***

Python is one of the leading languages in artificial intelligence and machine learning due to its extensive ecosystem of libraries and tools. Some of the most popular AI and ML libraries include:

- ***TensorFlow and PyTorch:*** These libraries provide powerful tools for deep learning, enabling the creation of neural networks for image recognition, natural language processing, and other AI-driven applications.
- ***Scikit-Learn:*** A library for classical machine learning algorithms such as regression, clustering, and classification.
- ***NLTK and spaCy:*** Used for natural language processing (NLP) tasks, including text analysis, sentiment analysis, and chatbot development.

2.3 ***Python in Automation:***

Python's simplicity and scripting capabilities make it an excellent choice for automation. It is widely used for:

- ***File Management:*** Automating file operations such as renaming, copying and moving files.
- ***Data Processing:*** Automating repetitive data analysis tasks using libraries like Pandas.
- ***Web Scraping:*** Extracting data from websites using BeautifulSoup and Scrapy, which is useful for research and business intelligence.

2.4 Python for Data Structures and Algorithms:

Python provides built-in support for essential data structures, making it a popular choice for algorithm design and competitive programming. Some of its key features include:

- ***Built-in Data Structures:*** Lists, dictionaries, sets, and tuples provide efficient ways to store and manipulate data.
- ***Algorithm Implementation:*** Python's simplicity allows developers to implement complex algorithms, such as sorting and searching, with minimal effort.
- ***Competitive Programming:*** Python is used in coding competitions due to its expressive syntax and built-in functions that simplify problem-solving.

With its versatility and efficiency, Python continues to be a dominant language in the field of computer science, enabling innovation and problem-solving across various domains.

3. PYTHON FOR DATA SCIENCE

3.1 Introduction to Data Science:

Data Science is an interdisciplinary field that focuses on extracting insights and knowledge from structured and unstructured data. It combines statistical analysis, machine learning, and data visualization techniques to help business and organizations make informed decisions. With the exponential growth of data in recent years, the demand for skilled data scientists has increased significantly. Python has emerged as the leading programming language for data science due to its simplicity, versatility, and a rich ecosystem of libraries that streamline data analysis, preprocessing, and visualization.

Python's extensive range of data science libraries provides powerful tools for handling large datasets, performing numerical computations, creating machine learning models, and generating insightful visualizations. The ease of integrating Python with other technologies makes it an ideal choice for data-driven applications in finance, healthcare, marketing, and more.

3.2 Key Python Libraries for Data Science:

Python offers numerous libraries tailored for data science. Some of the most widely used ones include:

- ***Pandas:*** A powerful library for data manipulation and analysis, allowing users to handle tabular data with ease. Pandas provides functionalities such as reading and writing data from various sources (CSV, Excel, SQL), filtering, and transforming datasets efficiently.
- ***NumPy:*** This library is essential for numerical computing in Python. It supports multi-dimensional array and a collection of mathematical functions, enabling scientific computations.
- ***Matplotlib and Seaborn:*** These libraries help in data visualization by creating various types of charts and graphs, such as histograms, scatter plots, and heatmaps. Seaborn, built on top of Matplotlib, provides enhanced visualization features with aesthetic and informative statistical graphs.
- ***Scikit-Learn:*** A machine learning library that offers simple and efficient tools for predictive modeling, clustering, and regression. It includes various algorithms such as decision trees, support vector machines, and ensemble methods for building and evaluating models.

3.3 Data Preprocessing and Cleaning with Python:

Before analyzing data, it is crucial to clean and preprocess it to ensure accuracy and consistency. Python provides tools for handling missing data, transforming variables, and preparing datasets for analysis. Some key steps in data preprocessing include:

- ***Handling Missing Values:*** Missing data is a common issue in data sets. Pandas provides methods such as fillna() to replace missing values with mean, median, or other statistical measures and dropna() to remove incomplete records.
- ***Normalizing Data:*** Data normalization ensures that numerical values are scaled to a uniform range, improving the performance of machine learning models. NumPy and Scikit-learn offer functions for standardization and normalization.
- ***Encoding Categorical Data:*** Many datasets contain categorial variables that need to be converted into numerical representations using techniques like one-hot encoding or label encoding, which can be done efficiently with Pandas and Scikit-learn.

3.4 ***Data Visualization:***

Data visualization is an essential part of data science, allowing analysis to identify patterns, trends, and anomalies in datasets. Python's Matplotlib and Seaborn libraries provide powerful tools for creating informative and aesthetically pleasing visualizations, including:

- ***Histograms and Bar Charts:*** Used for visualizing the distribution of numerical and categorical data.
- ***Scatter Plots:*** Help identify relationships and correlations between variables.
- ***Heatmaps:*** Used to display the intensity of data values and highlight patterns in large datasets.
- ***Box Plots:*** Useful for detecting outliers and analyzing the spread of data.

With these visualization tools, data scientists can communicate their findings effectively, aiding better decision-making and strategic planning. Python's role in data science continues to grow, making it an invaluable tool for professionals in the field.

4. PROJECT: COVID-19 GLOBAL IMPACT VISUALIZER

In this project, we aim to build a simple yet powerful data science model using Python to analyze and visualize COVID-19 data from 2020. The datasets come from real-time COVID-19 statistics provided by Kaggle, and they cover a wide range of metrics such as confirmed cases, deaths, recoveries, active cases, new cases, and other important indicators across different countries and regions. The primary objective of this project is to demonstrate the entire data science workflow, from data loading and cleaning to analysis and visualization, followed by building simple predictive models.

The datasets include detailed information about:

- **4.1 *Confirmed Cases:*** Total number of confirmed cases in each country/region.
- **4.2 *Deaths:*** Total deaths attributed to COVID-19.
- **4.3 *Recovered Cases:*** Number of patients who have recovered.
- **4.4 *Active Cases:*** The number of currently active cases.
- **4.5 *New Cases:*** Daily new confirmed cases.
- **4.6 *New Deaths:*** Daily new deaths due to COVID-19.
- **4.7 *New Recovered:*** Daily new recoveries.
- **4.8 *Deaths per 100 Cases:*** The proportion of deaths per 100 confirmed cases.
- **4.9 *Recoveries per 100 Cases:*** The proportion of recoveries per 100 confirmed cases.
- **4.10 *Summary of Key COVID-19 Data Across Countries:*** The summary of key COVID-19 data across multiple countries.

This project is designed to serve as a hands-on guide for anyone looking to explore data science techniques such as data cleaning, preprocessing, exploratory data analysis (EDA), and machine learning. We will use popular Python libraries like pandas, matplotlib, seaborn, and scikit-learn for data manipulation, visualization, and modeling. By analyzing these datasets, we aim to uncover trends, patterns, and insights that reflect the global impact of COVID-19 during the year 2020. Furthermore, the project will demonstrate how these data can be used to predict future trends and generate recommendations for policymakers and public health officials. The scope of the analysis is vast, offering valuable information on how the pandemic affected different regions and countries.

❖ ***Python Code for COVID-19 Data Analysis and Visualization: Key Metrics:***

❖ ***Introduction:***

In this section, we provide an in-depth look into the Python code that processes and visualizes COVID-19 data of the year 2020, including confirmed cases, deaths, recoveries, and trends. By utilizing Python libraries like ***matplotlib***, ***seaborn***, and ***pandas***, this code produces visual representations that helps us better understand the pandemic's impact globally. Let's explore the code for each metric in detail.

4.1 Python Code for Analyzing Confirmed COVID-19 Cases:

➢ ***Python Code:***

```
# Data Visualization of COVID-19 Confirmed Cases by
Country/Region
# This code visualizes the confirmed COVID-19 cases data for
various countries/regions using a bar chart.

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

# Create a dictionary with the country/region and confirmed
cases
data = {
    'Country/Region': [
        'Afghanistan', 'Albania', 'Algeria', 'Andorra',
'Angola', 'Antigua and Barbuda', 'Argentina', 'Armenia',
'Australia',
        'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin',
        'Bhutan', 'Bolivia', 'Bosnia and Herzegovina',
'Botswana', 'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso',
'Burma',
```

```
        'Burundi', 'Cabo Verde', 'Cambodia', 'Cameroon',
'Canada', 'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia',
        'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)',
'Costa Rica', 'Cote d\'Ivoire', 'Croatia', 'Cuba', 'Cyprus',
'Czechia',
        'Denmark', 'Djibouti', 'Dominica', 'Dominican
Republic', 'Ecuador', 'Egypt', 'El Salvador', 'Equatorial
Guinea', 'Eritrea',
        'Estonia', 'Eswatini', 'Ethiopia', 'Fiji', 'Finland',
'France', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana',
'Greece',
        'Greenland', 'Grenada', 'Guatemala', 'Guinea', 'Guinea-
Bissau', 'Guyana', 'Haiti', 'Holy See', 'Honduras', 'Hungary',
'Iceland',
        'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland',
'Israel', 'Italy', 'Jamaica', 'Japan', 'Jordan', 'Kazakhstan',
'Kenya', 'Kosovo',
        'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
'Lesotho', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania',
'Luxembourg',
        'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali',
'Malta', 'Mauritania', 'Mauritius', 'Mexico', 'Moldova',
'Monaco', 'Mongolia',
        'Montenegro', 'Morocco', 'Mozambique', 'Namibia',
'Nepal', 'Netherlands', 'New Zealand', 'Nicaragua', 'Niger',
'Nigeria', 'North Macedonia',
        'Norway', 'Oman', 'Pakistan', 'Panama', 'Papua New
Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland',
'Portugal', 'Qatar', 'Romania',
        'Russia', 'Rwanda', 'Saint Kitts and Nevis', 'Saint
Lucia', 'Saint Vincent and the Grenadines', 'San Marino', 'Sao
Tome and Principe',
        'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles',
'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia', 'Somalia',
'South Africa',
        'South Korea', 'South Sudan', 'Spain', 'Sri Lanka',
'Sudan', 'Suriname', 'Sweden', 'Switzerland', 'Syria',
'Taiwan*', 'Tajikistan',
        'Tanzania', 'Thailand', 'Timor-Leste', 'Togo',
'Trinidad and Tobago', 'Tunisia', 'Turkey', 'US', 'Uganda',
'Ukraine', 'United Arab Emirates',
        'United Kingdom', 'Uruguay', 'Uzbekistan', 'Venezuela',
'Vietnam', 'West Bank and Gaza', 'Western Sahara', 'Yemen',
'Zambia', 'Zimbabwe'
    ],
    '#Confirmed': [
```

```
        36263, 4880, 27973, 907, 950, 86, 167416, 37390, 15303,
20558, 30446, 382, 39482, 226225, 110, 67251, 66428, 48, 1770,
99, 71181, 10498,
        739, 2442375, 141, 10621, 1100, 350, 378, 2328, 226,
17110, 116458, 4599, 922, 347923, 86783, 257101, 354, 3200,
8844, 15841, 15655, 4881,
        2532, 1060, 15516, 13761, 5059, 18, 64156, 81161,
92482, 15035, 3071, 265, 2034, 2316, 14547, 27, 7398, 220352,
7189, 326, 1137, 207112,
        33624, 4227, 14, 23, 45309, 7055, 1954, 389, 7340, 12,
39741, 4448, 1854, 1480073, 100303, 293606, 112585, 25892,
63985, 246286, 853, 31142,
        1176, 84648, 17975, 7413, 64379, 33296, 20, 1219, 3882,
505, 1167, 2827, 86, 2019, 6321, 9690, 3664, 8904, 3369, 2513,
701, 6208, 344,
        395489, 23154, 116, 289, 2893, 20887, 1701, 1843,
18752, 53413, 1557, 3439, 1132, 41180, 10213, 9132, 77058,
274289, 61442, 62, 4548, 389717,
        82040, 43402, 50299, 109597, 45902, 816680, 1879, 17,
24, 52, 699, 865, 268934, 9764, 24141, 114, 1783, 50838, 2181,
2087, 3196, 452529,
        14203, 2305, 272421, 2805, 11424, 1483, 79395, 34477,
674, 462, 7235, 509, 3297, 24, 874, 148, 1455, 227019, 4290259,
1128, 67096, 59177,
        301708, 1202, 21209, 15988, 431, 10621, 10, 1691, 4552,
2704
    ]
}

# Convert the dictionary into a pandas DataFrame
df = pd.DataFrame(data)

# Create a bar plot using seaborn
plt.figure(figsize=(10, 8))
sns.barplot(x='#Confirmed', y='Country/Region',
data=df.sort_values('#Confirmed', ascending=False).head(20),
palette='viridis')

# Add labels and title
plt.title('Top 20 Countries/Regions by Confirmed Cases',
fontsize=16)
plt.xlabel('Number of Confirmed Cases', fontsize=12)
plt.ylabel('Country/Region', fontsize=12)

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

- ***Code Explanation:***

  - ***'data' Dictionary:*** Contains data about each country/region and its corresponding confirmed cases.
    - ✓ Key: 'Country/Region' (List of countries/regions)
    - ✓ Key: '#Confirmed' (List of confirmed cases in corresponding countries)
  - ***'df' Data Frame:*** A pandas DataFrame created from the dictionary, making the data easier to manipulate and analyze.
  - ***'sns.barplot()':*** This function is used to plot a bar chart of the top 20 countries/regions with the highest number of confirmed cases. It sorts the data in descending order using 'df.sort_values('#Confirmed', ascending=False).head(20))'.

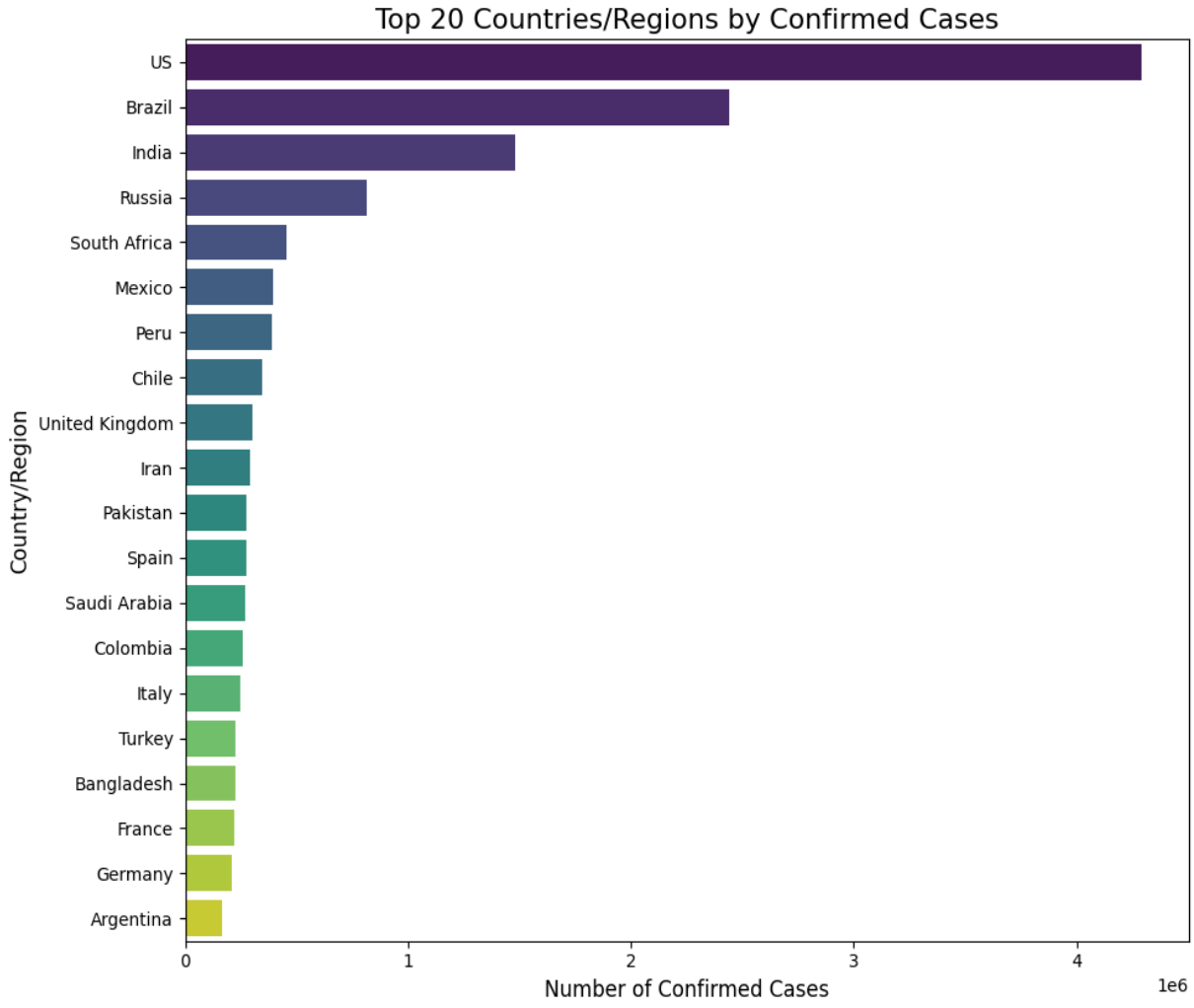
- ***Output: Top 20 Countries/Regions by Confirmed Cases:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Displays the number of confirmed cases.
    - ✓ ***'Y-axis':*** Displays the country/region names.
    - ✓ The top 20 countries with the highest confirmed cases are shown in a descending order.

- ***Interpretation:*** This chart provides a snapshot of the countries most affected by COVID-19 in terms of total confirmed cases.

❖ *OUTPUT:*

```
<ipython-input-2-124da6500809>:48: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and
will be removed in v0.14.0. Assign the `y` variable to `hue`
and set `legend=False` for the same effect.

  sns.barplot(x='#Confirmed', y='Country/Region',
data=df.sort_values('#Confirmed', ascending=False).head(20),
palette='viridis')
```



4.2 Python Code for Mortality Analysis: Total COVID-19 Deaths:

- **Python Code:**

```
# Data Visualization of COVID-19 Deaths by Country/Region
# This code visualizes the death counts of COVID-19 across
different countries/regions using a bar chart.


# Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Creating the data
countries = ["Afghanistan", "Albania", "Algeria", "Andorra",
"Angola", "Antigua and Barbuda", "Argentina",
             "Armenia", "Australia", "Austria", "Azerbaijan",
"Bahamas", "Bahrain", "Bangladesh", "Barbados",
             "Belarus", "Belgium", "Belize", "Benin", "Bhutan",
"Bolivia", "Bosnia and Herzegovina", "Botswana",
             "Brazil", "Brunei", "Bulgaria", "Burkina Faso",
"Burma", "Burundi", "Cabo Verde", "Cambodia",
             "Cameroon", "Canada", "Central African Republic",
"Chad", "Chile", "China", "Colombia", "Comoros",
             "Congo (Brazzaville)", "Congo (Kinshasa)", "Costa
Rica", "Cote d'Ivoire", "Croatia", "Cuba", "Cyprus",
             "Czechia", "Denmark", "Djibouti", "Dominica",
"Dominican Republic", "Ecuador", "Egypt", "El Salvador",
             "Equatorial Guinea", "Eritrea", "Estonia",
"Eswatini", "Ethiopia", "Fiji", "Finland", "France",
             "Gabon", "Gambia", "Georgia", "Germany", "Ghana",
"Greece", "Greenland", "Grenada", "Guatemala",
             "Guinea", "Guinea-Bissau", "Guyana", "Haiti",
"Holy See", "Honduras", "Hungary", "Iceland", "India",
             "Indonesia", "Iran", "Iraq", "Ireland", "Israel",
"Italy", "Jamaica", "Japan", "Jordan", "Kazakhstan",
             "Kenya", "Kosovo", "Kuwait", "Kyrgyzstan", "Laos",
"Latvia", "Lebanon", "Lesotho", "Liberia", "Libya",
             "Liechtenstein", "Lithuania", "Luxembourg",
"Madagascar", "Malawi", "Malaysia", "Maldives", "Mali",
             "Malta", "Mauritania", "Mauritius", "Mexico",
"Moldova", "Monaco", "Mongolia", "Montenegro", "Morocco",
             "Mozambique", "Namibia", "Nepal", "Netherlands",
"New Zealand", "Nicaragua", "Niger", "Nigeria",
```

```
            "North Macedonia", "Norway", "Oman", "Pakistan",
"Panama", "Papua New Guinea", "Paraguay", "Peru",
            "Philippines", "Poland", "Portugal", "Qatar",
"Romania", "Russia", "Rwanda", "Saint Kitts and Nevis",
            "Saint Lucia", "Saint Vincent and the Grenadines",
"San Marino", "Sao Tome and Principe", "Saudi Arabia",
            "Senegal", "Serbia", "Seychelles", "Sierra Leone",
"Singapore", "Slovakia", "Slovenia", "Somalia",
            "South Africa", "South Korea", "South Sudan",
"Spain", "Sri Lanka", "Sudan", "Suriname", "Sweden",
            "Switzerland", "Syria", "Taiwan*", "Tajikistan",
"Tanzania", "Thailand", "Timor-Leste", "Togo",
            "Trinidad and Tobago", "Tunisia", "Turkey", "US",
"Uganda", "Ukraine", "United Arab Emirates",
            "United Kingdom", "Uruguay", "Uzbekistan",
"Venezuela", "Vietnam", "West Bank and Gaza", "Western Sahara",
            "Yemen", "Zambia", "Zimbabwe"]

deaths = [1269, 144, 1163, 52, 41, 3, 3059, 711, 167, 713, 423,
11, 141, 2965, 7, 538, 9822, 2, 35, 0,
          2647, 294, 2, 87618, 3, 347, 53, 6, 1, 22, 0, 391,
8944, 59, 75, 9187, 4656, 8777, 7, 54, 208,
          115, 96, 139, 87, 19, 373, 613, 58, 0, 1083, 5532,
4652, 408, 51, 0, 69, 34, 228, 0, 329, 30212,
          49, 8, 16, 9125, 168, 202, 0, 0, 1761, 45, 26, 20,
158, 0, 1166, 596, 10, 33408, 4838, 15912,
          4458, 1764, 474, 35112, 10, 998, 11, 585, 285, 185,
438, 1301, 0, 31, 51, 12, 72, 64, 1, 80, 112,
          91, 99, 124, 15, 124, 9, 156, 10, 44022, 748, 4, 0,
45, 316, 11, 8, 48, 6160, 22, 108, 69, 860,
          466, 255, 393, 5842, 1322, 0, 43, 18418, 1945, 1676,
1719, 165, 2206, 13334, 5, 0, 0, 0, 42,
          14, 2760, 194, 543, 0, 66, 27, 28, 116, 93, 7067,
300, 46, 28432, 11, 720, 24, 5700, 1978, 40,
          7, 60, 21, 58, 0, 18, 8, 50, 5630, 148011, 2, 1636,
345, 45844, 35, 121, 146, 0, 78, 1, 483,
          140, 36]

# Creating a DataFrame from the data
df = pd.DataFrame({
    'Country/Region': countries,
    '#Deaths': deaths
})

# Sorting the data by #Deaths in descending order
df_sorted = df.sort_values('#Deaths', ascending=False)

# Plotting the data
```

```
plt.figure(figsize=(12, 18))
sns.barplot(x='#Deaths', y='Country/Region',
data=df_sorted.head(20), palette='viridis')

# Adding title and labels
plt.title('Top 20 Countries by Number of Deaths', fontsize=16)
plt.xlabel('Number of Deaths', fontsize=12)
plt.ylabel('Country/Region', fontsize=12)

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

- ***Code Explanation:***

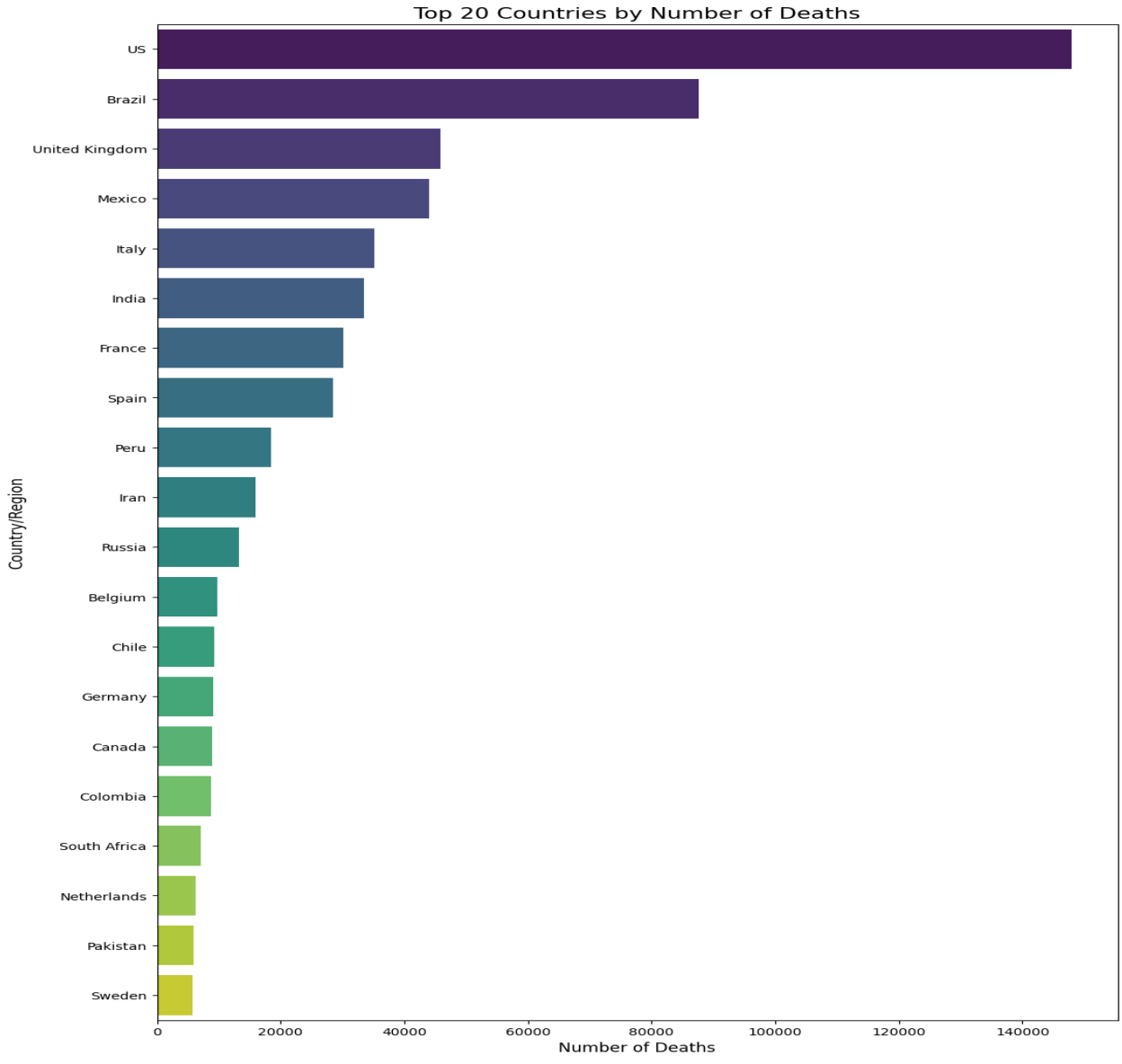
  - ***'data' Dictionary:*** Includes total death data for each country/region/
    - ✓ Key: 'Country/Region' (List of countries/region).
    - ✓ Key: '#Deaths' (List of total deaths in the respective countries).
  - ***'df' DataFrame:*** The pandas DataFrame helps store this information, facilitating further analysis.
  - ***'sns.barplot()':*** A bar chat is plotted to visualize the total deaths for the top 20 affected countries.

- ***Output: Top 20 Countries/Regions by Total Deaths:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Shows the number of deaths.
    - ✓ ***'Y-axis':*** Displays the country/region names.
  - ***Interpretation:*** The chart highlights the countries with the highest COVID-19 death tolls.

❖ ***OUTPUT:***

```
<ipython-input-13-193ffef870b6>:58: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and
will be removed in v0.14.0. Assign the `y` variable to `hue`
and set `legend=False` for the same effect.

  sns.barplot(x='#Deaths', y='Country/Region',
data=df_sorted.head(20), palette='viridis')
```



4.3 Python Code for Recovery Trends: Total Recovered Cases:

- **Python Code:**

```
# Data Visualization of COVID-19 Recovered Cases by
Country/Region
# This code visualizes the number of COVID-19 recovered cases
across different countries/regions using a bar chart.

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# COVID-19 Data for Countries/Regions (Recovered Cases)
data = {
    'Country/Region': ['Afghanistan', 'Albania', 'Algeria',
'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina',
'Armenia', 'Australia', 'Austria',
                       'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin', 'Bhutan', 'Bolivia',
                       'Bosnia and Herzegovina', 'Botswana',
'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma',
'Burundi', 'Cabo Verde',
                       'Cambodia', 'Cameroon', 'Canada',
'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Congo (Brazzaville)',
                       'Congo (Kinshasa)', 'Costa Rica', "Cote
d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
'Djibouti', 'Dominica',
                       'Dominican Republic', 'Ecuador',
'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
'Estonia', 'Eswatini', 'Ethiopia', 'Fiji',
                       'Finland', 'France', 'Gabon', 'Gambia',
'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenland',
'Grenada', 'Guatemala', 'Guinea',
                       'Guinea-Bissau', 'Guyana', 'Haiti',
'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Iran', 'Iraq', 'Ireland',
                       'Israel', 'Italy', 'Jamaica', 'Japan',
'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
                       'Lesotho', 'Liberia', 'Libya',
'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
'Malawi', 'Malaysia', 'Maldives', 'Mali',
```

```
                       'Malta', 'Mauritania', 'Mauritius',
'Mexico', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
'Morocco', 'Mozambique', 'Namibia', 'Nepal',
                       'Netherlands', 'New Zealand',
'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
'Oman', 'Pakistan', 'Panama', 'Papua New Guinea',
                       'Paraguay', 'Peru', 'Philippines',
'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
'Saint Kitts and Nevis', 'Saint Lucia',
                       'Saint Vincent and the Grenadines', 'San
Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
'Serbia', 'Seychelles', 'Sierra Leone',
                       'Singapore', 'Slovakia', 'Slovenia',
'Somalia', 'South Africa', 'South Korea', 'South Sudan',
'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
                       'Sweden', 'Switzerland', 'Syria',
'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',
'Togo', 'Trinidad and Tobago', 'Tunisia',
                       'Turkey', 'US', 'Uganda', 'Ukraine',
'United Arab Emirates', 'United Kingdom', 'Uruguay',
'Uzbekistan', 'Venezuela', 'Vietnam', 'West Bank and Gaza',
                       'Western Sahara', 'Yemen', 'Zambia',
'Zimbabwe'],
    'Recovered': [25198, 2745, 18837, 803, 242, 65, 72575,
26665, 9311, 18246, 23242, 91, 36110, 125683, 94, 60492, 17452,
26, 1036, 86, 21478, 4930, 63,
                   1846641, 138, 5585, 926, 292, 301, 1550, 147,
14539, 107514, 1546, 810, 319954, 78869, 131161, 328, 829,
5700, 3824, 10361, 3936, 2351,
                   852, 11428, 12605, 4977, 18, 30204, 34896,
34838, 7778, 842, 191, 1923, 1025, 6386, 18, 6920, 81212, 4682,
66, 922, 190314, 29801, 1374,
                   13, 23, 32455, 6257, 803, 181, 4365, 12,
5039, 3329, 1823, 951166, 58173, 255144, 77144, 23364, 27133,
198593, 714, 21970, 1041, 54404,
                   7833, 4027, 55057, 21205, 19, 1045, 1709,
128, 646, 577, 81, 1620, 4825, 6260, 1645, 8601, 2547, 1913,
665, 4653, 332, 303810, 16154,
                   104, 222, 809, 16553, 0, 101, 13754, 189,
1514, 2492, 1027, 18203, 5564, 8752, 57028, 241026, 35086, 11,
2905, 272547, 26446, 32856,
                   35375, 106328, 25794, 602249, 975, 15, 22,
39, 657, 734, 222936, 6477, 0, 39, 1317, 45692, 1616, 1733,
1543, 274925, 13007, 1175,
                   150376, 2121, 5939, 925, 0, 30900, 0, 440,
6028, 183, 3111, 0, 607, 128, 1157, 210469, 1325804, 986,
37202, 52510, 1437, 951, 11674,
                   9959, 365, 3752, 8, 833, 2815, 542]
```

```
}

# Creating a DataFrame
df = pd.DataFrame(data)

# Sorting the DataFrame by 'Recovered' in descending order
df_sorted = df.sort_values(by='Recovered', ascending=False)

# Select the top 20 countries
df_top_20 = df_sorted.head(20)

# Plotting Recovered Cases for Top 20 Countries/Regions
plt.figure(figsize=(15, 10))

# Bar Plot for Recovered Cases (Top 20 Countries)
sns.barplot(x='Recovered', y='Country/Region', data=df_top_20,
palette='viridis')

# Adding Titles and Labels
plt.title('Top 20 COVID-19 Recovered Cases by Country/Region',
fontsize=16)
plt.xlabel('Recovered Cases', fontsize=12)
plt.ylabel('Country/Region', fontsize=12)

# Display the Plot
plt.tight_layout()
plt.show()
```

- ***Code Explanation:***

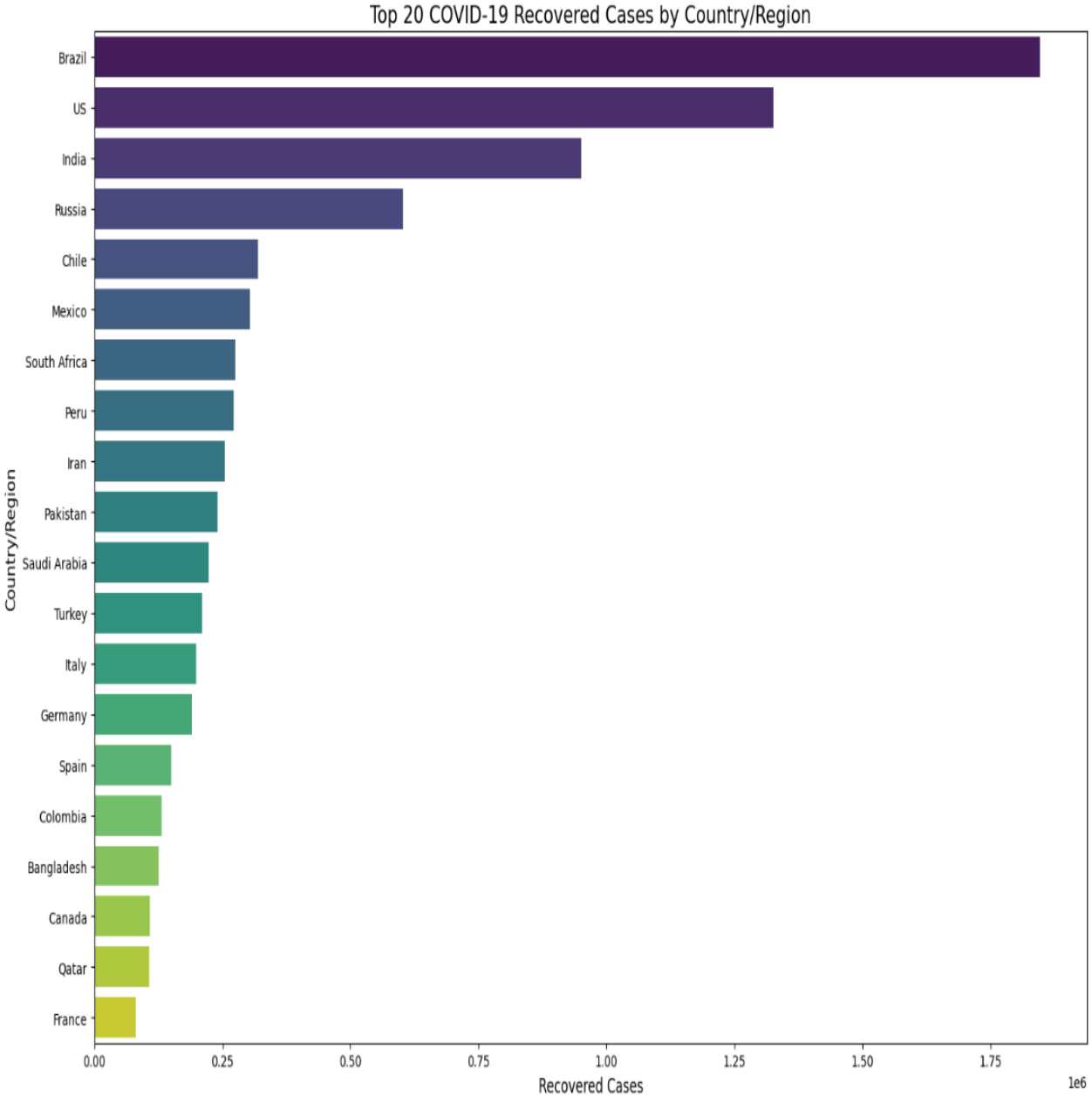
  - ***'data' Dictionary:*** Contains data on recoveries for each country/region.
    - ✓ Key: 'Country/Region' (List of countries/regions).
    - ✓ Key: '#Recovered' (List of total recoveries in each country).
  - ***'df' DataFrame:*** This DataFrame is created to manage the recovery data.
  - ***'sns.barplot()':*** A bar plot visualizes the total recoveries by country/region for the top 20 countries.

- ***Output: Top 20 Countries/Regions by Total Recovered Cases:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Represents the number of recoveries.
    - ✓ ***'Y-axis':*** Displays the country/region names.
  - ***Interpretation:*** This chart shows the countries with the highest number of recoveries, reflecting the effectiveness of recovery treatments.

❖ ***OUTPUT:***

```
<ipython-input-22-6064e2551c32>:53: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and
will be removed in v0.14.0. Assign the `y` variable to `hue`
and set `legend=False` for the same effect.

  sns.barplot(x='Recovered', y='Country/Region',
data=df_top_20, palette='viridis')
```



4.4 Python Code for Tracking Active COVID-19 Cases:

- **Python Code:**

```
# Data Visualization of COVID-19 Active Cases by Country/Region
# This code visualizes the number of COVID-19 active cases
across different countries/regions using a bar chart.


# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# COVID-19 Data for Countries/Regions (Active Cases)
data_active = {
    'Country/Region': ['Afghanistan', 'Albania', 'Algeria',
'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina',
'Armenia', 'Australia', 'Austria',
                       'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin', 'Bhutan', 'Bolivia',
                       'Bosnia and Herzegovina', 'Botswana',
'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma',
'Burundi', 'Cabo Verde',
                       'Cambodia', 'Cameroon', 'Canada',
'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Congo (Brazzaville)',
                       'Congo (Kinshasa)', 'Costa Rica', "Cote
d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
'Djibouti', 'Dominica',
                       'Dominican Republic', 'Ecuador',
'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
'Estonia', 'Eswatini', 'Ethiopia', 'Fiji',
                       'Finland', 'France', 'Gabon', 'Gambia',
'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenland',
'Grenada', 'Guatemala', 'Guinea',
                       'Guinea-Bissau', 'Guyana', 'Haiti',
'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Iran', 'Iraq', 'Ireland',
                       'Israel', 'Italy', 'Jamaica', 'Japan',
'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
                       'Lesotho', 'Liberia', 'Libya',
'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
'Malawi', 'Malaysia', 'Maldives', 'Mali',
```

```
                        'Malta', 'Mauritania', 'Mauritius',
'Mexico', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
'Morocco', 'Mozambique', 'Namibia', 'Nepal',
                        'Netherlands', 'New Zealand',
'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
'Oman', 'Pakistan', 'Panama', 'Papua New Guinea',
                        'Paraguay', 'Peru', 'Philippines',
'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
'Saint Kitts and Nevis', 'Saint Lucia',
                        'Saint Vincent and the Grenadines', 'San
Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
'Serbia', 'Seychelles', 'Sierra Leone',
                        'Singapore', 'Slovakia', 'Slovenia',
'Somalia', 'South Africa', 'South Korea', 'South Sudan',
'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
                        'Sweden', 'Switzerland', 'Syria',
'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',
'Togo', 'Trinidad and Tobago', 'Tunisia',
                        'Turkey', 'US', 'Uganda', 'Ukraine',
'United Arab Emirates', 'United Kingdom', 'Uruguay',
'Uzbekistan', 'Venezuela', 'Vietnam', 'West Bank and Gaza',
                        'Western Sahara', 'Yemen', 'Zambia',
'Zimbabwe'],
    'Active': [9796, 1991, 7973, 52, 667, 18, 91782, 10014,
5825, 1599, 6781, 280, 3231, 97577, 9, 6221, 39154, 20, 699,
13, 47056, 5274, 674, 508116, 0,
                4689, 121, 52, 76, 756, 79, 2180, 682, 2994, 37,
18782, 3258, 117163, 19, 2317, 2936, 11902, 5198, 806, 94, 189,
3715, 543, 24, 0, 32869,
                40733, 52992, 6849, 2178, 74, 42, 1257, 7933, 9,
149, 108928, 2458, 252, 199, 7673, 3655, 2651, 1, 0, 11093,
753, 1125, 188, 2817, 0, 33536,
                523, 21, 495499, 37292, 22550, 30983, 764,
36378, 12581, 129, 8174, 124, 29659, 9857, 3201, 8884, 10790,
1, 143, 2122, 365, 449, 2186, 4,
                319, 1384, 3339, 1920, 179, 807, 476, 27, 1399,
2, 47657, 6252, 8, 67, 2039, 4018, 1690, 1734, 4950, 47064, 21,
839, 36, 22117, 4183,
                125, 19637, 27421, 25034, 51, 1600, 98752,
53649, 8870, 13205, 3104, 17902, 201097, 899, 2, 2, 13, 0, 117,
43238, 3093, 23598, 75, 400,
                5119, 537, 238, 1560, 170537, 896, 1084, 93613,
673, 4765, 534, 73695, 1599, 634, 15, 1147, 305, 128, 24, 249,
12, 248, 10920, 2816444,
                140, 28258, 6322, 254427, 216, 9414, 5883, 66,
6791, 1, 375, 1597, 2126]
}
```

```
# Creating a DataFrame
df_active = pd.DataFrame(data_active)

# Sorting the DataFrame by 'Active' in descending order
df_active_sorted = df_active.sort_values(by='Active',
ascending=False)

# Select the top 20 countries
df_active_top_20 = df_active_sorted.head(20)

# Plotting Active Cases for Top 20 Countries/Regions
plt.figure(figsize=(15, 10))

# Bar Plot for Active Cases (Top 20 Countries)
sns.barplot(x='Active', y='Country/Region',
data=df_active_top_20, palette='viridis')

# Adding Titles and Labels
plt.title('Top 20 COVID-19 Active Cases by Country/Region',
fontsize=16)
plt.xlabel('Active Cases', fontsize=12)
plt.ylabel('Country/Region', fontsize=12)

# Display the Plot
plt.tight_layout()
plt.show()
```

- ***Code Explanation:***

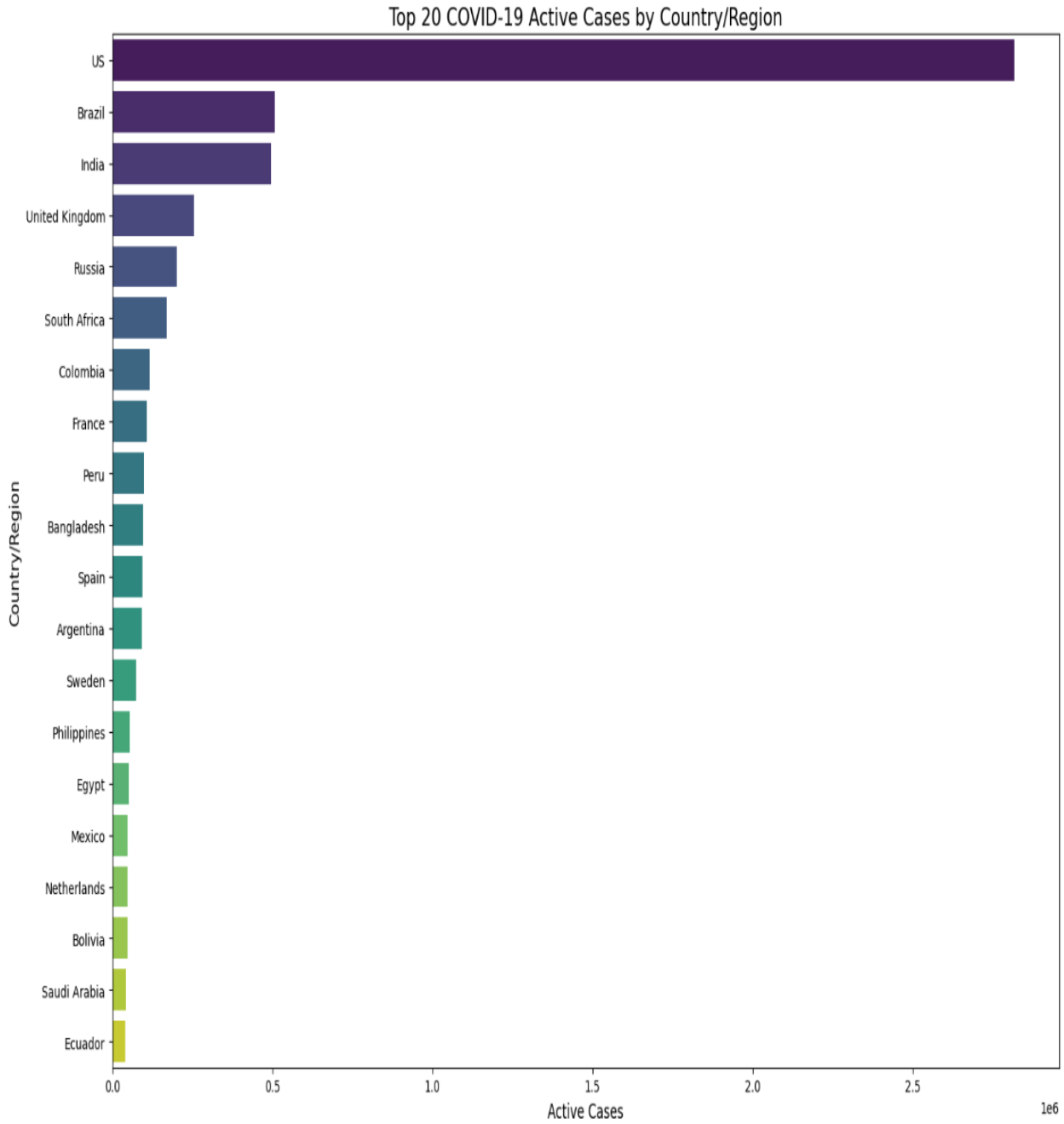
  - ***'data' Dictionary:*** Includes data on the current active cases.
    - ✓ Key: 'Country/Region' (List of countries/regions).
    - ✓ Key: '#Active' (List of active cases for each country/region).
  - ***'df' DataFrame:*** The active case data is stored here for easy manipulation.
  - ***'sns.barplot()':*** A bar plot is generated to visualize the number of active cases for the top 20 countries/regions.

- ***Output: Top 20 Countries/Regions by Total Recovered Cases:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Displays the number of active cases
    - ✓ ***'Y-axis':*** Lists the country/region names.
  - ***Interpretation:*** This chart helps identify regions still dealing with active cases showing the current burden of the virus.

❖ ***OUTPUT:***

```
<ipython-input-25-44adf99d3407>:49: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and
will be removed in v0.14.0. Assign the `y` variable to `hue`
and set `legend=False` for the same effect.

  sns.barplot(x='Active', y='Country/Region',
data=df_active_top_20, palette='viridis')
```



4.5 Python Code for Identifying Daily New COVID-19 Cases:

- **Python Code:**

```
# Data Visualization of COVID-19 New Cases by Country/Region
# This code visualizes the number of new COVID-19 cases across
different countries/regions using a bar chart.


# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# COVID-19 Data for Countries/Regions (New Cases)
data_new_cases = {
    'Country/Region': ['Afghanistan', 'Albania', 'Algeria',
'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina',
'Armenia', 'Australia', 'Austria',
                       'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin', 'Bhutan', 'Bolivia',
                       'Bosnia and Herzegovina', 'Botswana',
'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma',
'Burundi', 'Cabo Verde',
                       'Cambodia', 'Cameroon', 'Canada',
'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Congo (Brazzaville)',
                       'Congo (Kinshasa)', 'Costa Rica', "Cote
d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
'Djibouti', 'Dominica',
                       'Dominican Republic', 'Ecuador',
'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
'Estonia', 'Eswatini', 'Ethiopia', 'Fiji',
                       'Finland', 'France', 'Gabon', 'Gambia',
'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenland',
'Grenada', 'Guatemala', 'Guinea',
                       'Guinea-Bissau', 'Guyana', 'Haiti',
'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Iran', 'Iraq', 'Ireland',
                       'Israel', 'Italy', 'Jamaica', 'Japan',
'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
                       'Lesotho', 'Liberia', 'Libya',
'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
'Malawi', 'Malaysia', 'Maldives', 'Mali',
```

```
                        'Malta', 'Mauritania', 'Mauritius',
'Mexico', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
'Morocco', 'Mozambique', 'Namibia', 'Nepal',
                        'Netherlands', 'New Zealand',
'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
'Oman', 'Pakistan', 'Panama', 'Papua New Guinea',
                        'Paraguay', 'Peru', 'Philippines',
'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
'Saint Kitts and Nevis', 'Saint Lucia',
                        'Saint Vincent and the Grenadines', 'San
Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
'Serbia', 'Seychelles', 'Sierra Leone',
                        'Singapore', 'Slovakia', 'Slovenia',
'Somalia', 'South Africa', 'South Korea', 'South Sudan',
'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
                        'Sweden', 'Switzerland', 'Syria',
'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',
'Togo', 'Trinidad and Tobago', 'Tunisia',
                        'Turkey', 'US', 'Uganda', 'Ukraine',
'United Arab Emirates', 'United Kingdom', 'Uruguay',
'Uzbekistan', 'Venezuela', 'Vietnam', 'West Bank and Gaza',
                        'Western Sahara', 'Yemen', 'Zambia',
'Zimbabwe'],
    'New Cases': [106, 117, 616, 10, 18, 4, 4890, 73, 368, 86,
396, 40, 351, 2772, 0, 119, 402, 0, 0, 4, 1752, 731, 53, 23284,
0, 194, 14, 0, 17, 21,
                    1, 402, 11, 0, 7, 2133, 213, 16306, 0, 162,
13, 612, 59, 24, 37, 3, 192, 109, 9, 0, 1248, 467, 420, 405, 0,
2, 0, 109, 579, 0, 5,
                    2551, 205, 49, 6, 445, 655, 34, 1, 0, 256,
47, 0, 19, 25, 0, 465, 13, 7, 44457, 1525, 2434, 2553, 11,
2029, 168, 11, 594, 8, 1526,
                    372, 496, 606, 483, 0, 0, 132, 0, 5, 158, 0,
11, 49, 395, 24, 7, 67, 3, 1, 37, 0, 4973, 120, 0, 1, 94, 609,
32, 68, 139, 419, 1,
                    0, 0, 648, 127, 15, 1053, 1176, 1146, 0, 104,
13756, 1592, 337, 135, 292, 1104, 5607, 58, 0, 0, 0, 0, 2,
1993, 83, 411, 0, 0, 469,
                    2, 5, 18, 7096, 28, 43, 0, 23, 39, 44, 398,
65, 24, 4, 43, 0, 6, 0, 6, 1, 3, 919, 56336, 13, 835, 264, 688,
10, 678, 525, 11, 152,
                    0, 10, 71, 192]
}

# Creating a DataFrame
df_new_cases = pd.DataFrame(data_new_cases)

# Sorting the DataFrame by 'New Cases' in descending order
```

```
df_new_cases_sorted = df_new_cases.sort_values(by='New Cases',
ascending=False)

# Select the top 20 countries
df_new_cases_top_20 = df_new_cases_sorted.head(20)

# Plotting New Cases for Top 20 Countries/Regions
plt.figure(figsize=(15, 10))

# Bar Plot for New Cases (Top 20 Countries)
sns.barplot(x='New Cases', y='Country/Region',
data=df_new_cases_top_20, palette='magma')

# Adding Titles and Labels
plt.title('Top 20 Countries with Most New COVID-19 Cases',
fontsize=16)
plt.xlabel('New Cases', fontsize=12)
plt.ylabel('Country/Region', fontsize=12)

# Display the Plot
plt.tight_layout()
plt.show()
```

- ***Code Explanation:***

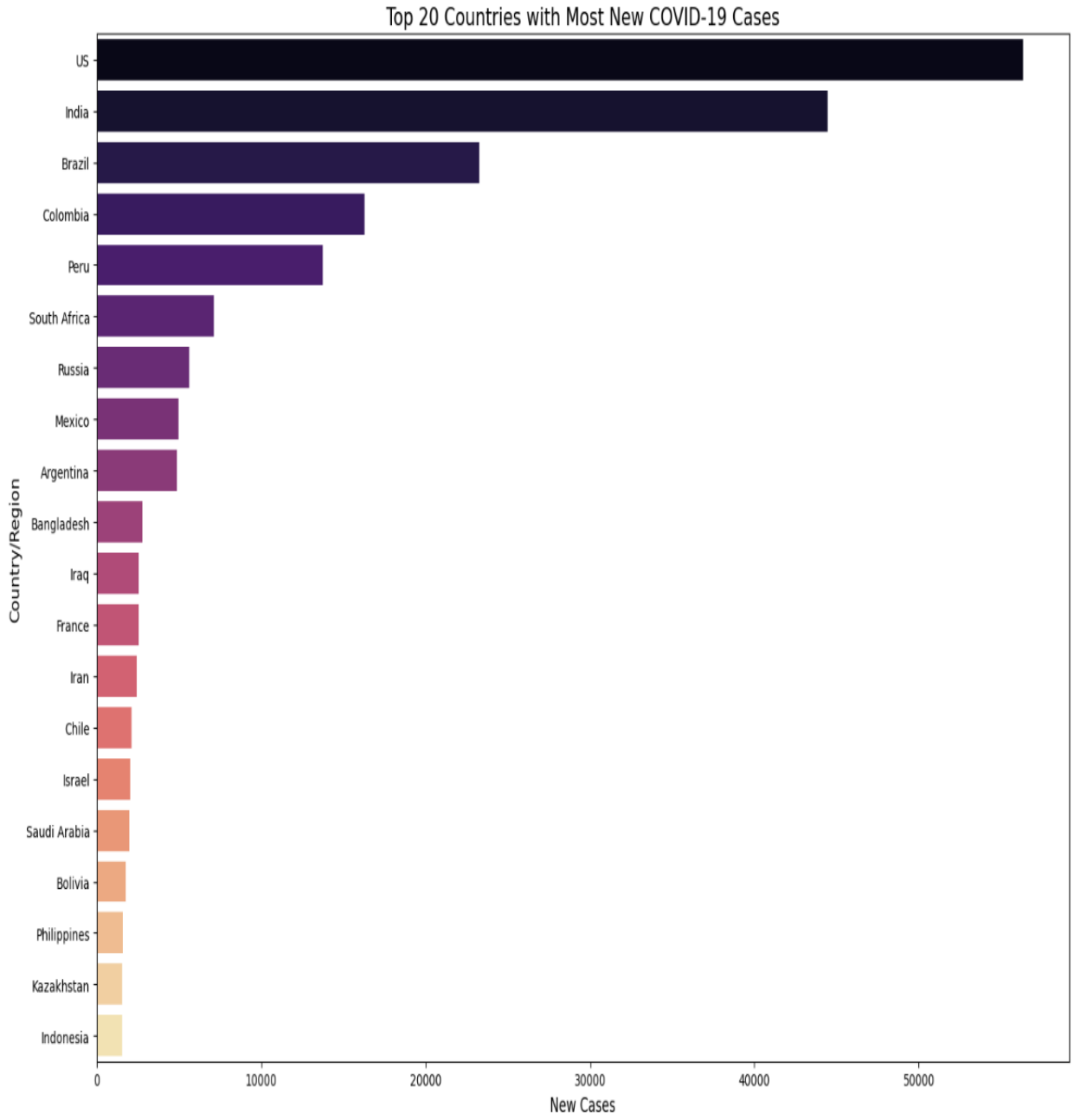
  - ***'data' Dictionary:*** Contains data about new confirmed cases.
    - ✓ Key: 'Country/Region' (List of countries/regions).
    - ✓ Key: '#New Cases' (List of daily new confirmed cases).
  - ***'df' DataFrame:*** Stores the new cases data.
  - ***'sns.barplot()':*** A bar plot visualizes the daily new cases for the top 20 countries.

- ***Output: Top 20 Countries/Regions by Daily New Cases:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Represents the number of new cases
    - ✓ ***'Y-axis':*** Displays the country/region names.
  - ***Interpretation:*** This chart shows the countries with the highest daily new cases, highlighting the current rate of infection spread.

❖ ***OUTPUT:***

```
<ipython-input-2-d44d5e6540b8>:52: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and
will be removed in v0.14.0. Assign the `y` variable to `hue`
and set `legend=False` for the same effect.

  sns.barplot(x='New Cases', y='Country/Region',
data=df_new_cases_top_20, palette='magma')
```



4.6 Python Code for Tracking New Daily Deaths:

- **Python Code:**

```
# Data Visualization of COVID-19 New Deaths by Country/Region
# This code visualizes the number of new COVID-19 deaths across
different countries/regions using a bar chart.


# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# COVID-19 Data for Countries/Regions (New Deaths)
data_new_deaths = {
    'Country/Region': ['Afghanistan', 'Albania', 'Algeria',
'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina',
'Armenia', 'Australia', 'Austria',
                       'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin', 'Bhutan', 'Bolivia',
                       'Bosnia and Herzegovina', 'Botswana',
'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma',
'Burundi', 'Cabo Verde',
                       'Cambodia', 'Cameroon', 'Canada',
'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Congo (Brazzaville)',
                       'Congo (Kinshasa)', 'Costa Rica', "Cote
d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
'Djibouti', 'Dominica',
                       'Dominican Republic', 'Ecuador',
'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
'Estonia', 'Eswatini', 'Ethiopia', 'Fiji',
                       'Finland', 'France', 'Gabon', 'Gambia',
'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenland',
'Grenada', 'Guatemala', 'Guinea',
                       'Guinea-Bissau', 'Guyana', 'Haiti',
'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Iran', 'Iraq', 'Ireland',
                       'Israel', 'Italy', 'Jamaica', 'Japan',
'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
                       'Lesotho', 'Liberia', 'Libya',
'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
'Malawi', 'Malaysia', 'Maldives', 'Mali',
```

```
                        'Malta', 'Mauritania', 'Mauritius',
'Mexico', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
'Morocco', 'Mozambique', 'Namibia', 'Nepal',
                        'Netherlands', 'New Zealand',
'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
'Oman', 'Pakistan', 'Panama', 'Papua New Guinea',
                        'Paraguay', 'Peru', 'Philippines',
'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
'Saint Kitts and Nevis', 'Saint Lucia',
                        'Saint Vincent and the Grenadines', 'San
Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
'Serbia', 'Seychelles', 'Sierra Leone',
                        'Singapore', 'Slovakia', 'Slovenia',
'Somalia', 'South Africa', 'South Korea', 'South Sudan',
'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
                        'Sweden', 'Switzerland', 'Syria',
'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',
'Togo', 'Trinidad and Tobago', 'Tunisia',
                        'Turkey', 'US', 'Uganda', 'Ukraine',
'United Arab Emirates', 'United Kingdom', 'Uruguay',
'Uzbekistan', 'Venezuela', 'Vietnam', 'West Bank and Gaza',
                        'Western Sahara', 'Yemen', 'Zambia',
'Zimbabwe'],
    'New Deaths': [10, 6, 8, 0, 1, 0, 120, 6, 6, 1, 6, 0, 1,
37, 0, 4, 1, 0, 0, 0, 64, 14, 1, 614, 0, 7, 0, 0, 0, 0,
                    0, 6, 0, 0, 0, 75, 4, 508, 0, 3, 4, 11, 0,
3, 0, 0, 2, 0, 0, 0, 20, 17, 46, 8, 0, 0, 0, 2, 5, 0,
                    0, 17, 0, 2, 0, 1, 0, 0, 0, 0, 27, 2, 0, 0,
1, 0, 50, 0, 0, 637, 57, 212, 96, 0, 4, 5, 0, 0, 0,
                    0, 5, 16, 5, 24, 0, 0, 0, 0, 0, 4, 0, 0, 0,
6, 0, 0, 0, 1, 0, 0, 0, 342, 13, 0, 0, 2, 3, 0, 0,
                    3, 1, 0, 0, 0, 2, 6, 0, 9, 20, 28, 0, 2,
575, 13, 5, 2, 0, 19, 85, 0, 0, 0, 0, 0, 0, 27, 3, 9, 0,
                    0, 0, 0, 0, 0, 298, 1, 1, 0, 0, 3, 1, 3, 1,
2, 0, 1, 0, 0, 0, 0, 0, 0, 17, 1076, 0, 11, 1, 7, 1, 5, 4,
                    0, 2, 0, 4, 1, 2]
}

# Creating a DataFrame from the given data
df_new_deaths = pd.DataFrame(data_new_deaths)

# Sorting the DataFrame by 'New Deaths' in descending order
df_new_deaths_sorted = df_new_deaths.sort_values(by='New
Deaths', ascending=False)

# Select the top 20 countries with the most new deaths
df_new_deaths_top_20 = df_new_deaths_sorted.head(20)
```

```
# Setting figure size for better visualization
plt.figure(figsize=(15, 10))

# Creating a bar plot to visualize new COVID-19 deaths in the
top 20 countries
sns.barplot(x='New Deaths', y='Country/Region',
data=df_new_deaths_top_20, palette='magma')

# Adding a title to the plot
plt.title('Top 20 Countries with Most New COVID-19 Deaths',
fontsize=16)

# Labeling the x-axis
plt.xlabel('New Deaths', fontsize=12)

# Labeling the y-axis
plt.ylabel('Country/Region', fontsize=12)

# Adjust layout for better spacing
plt.tight_layout()

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

- ***Code Explanation:***

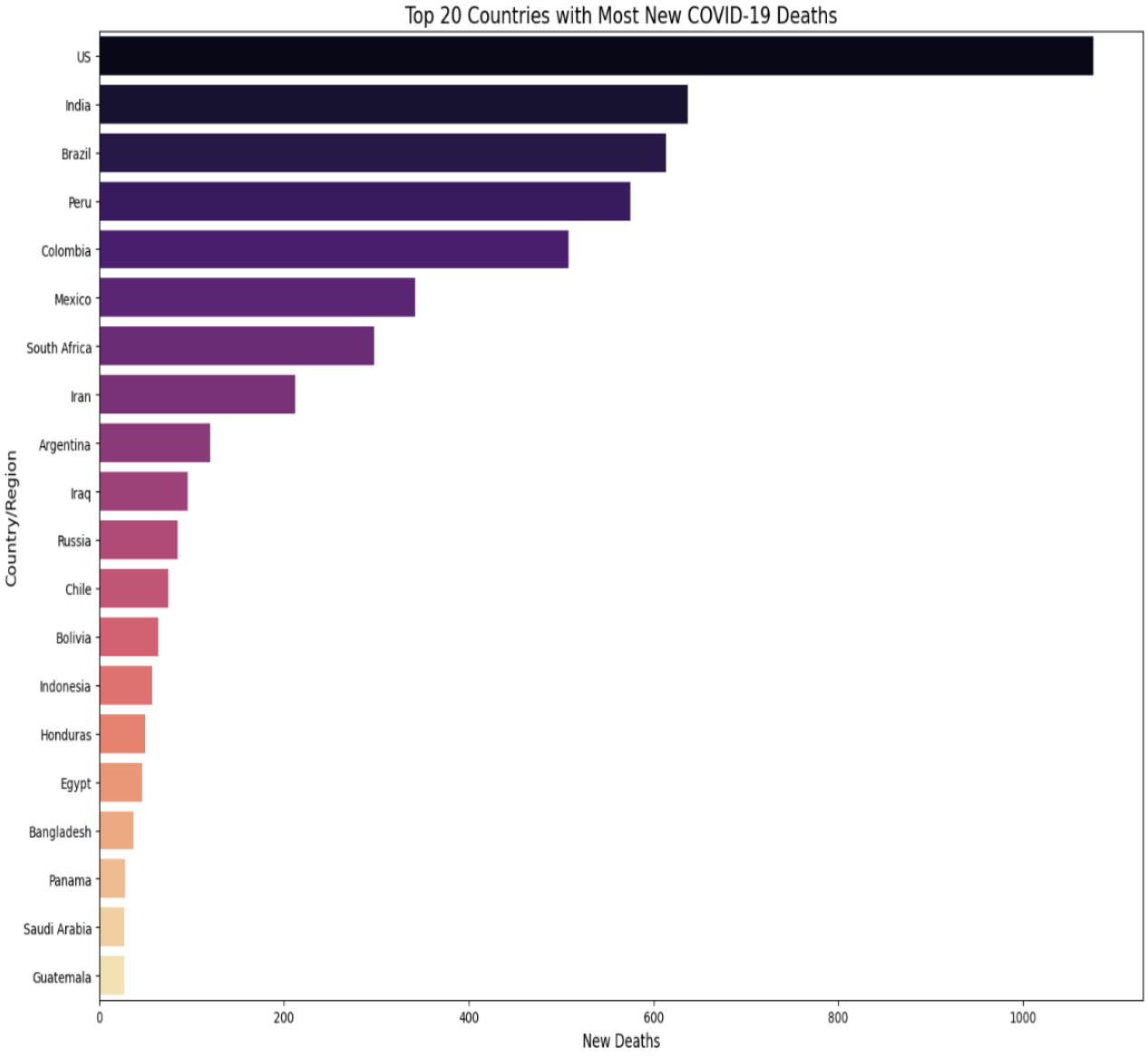
  - ***'data' Dictionary:*** Includes the data for new deaths each day.
    - ✓ Key: 'Country/Region' (List of countries/regions).
    - ✓ Key: '#New Deaths' (List of daily new deaths).
  - ***'df' DataFrame:*** The new deaths data is stored here for analysis..
  - ***'sns.barplot()':*** A bar plot is created to show the daily new deaths for the top 20 countries.

- ***Output: Top 20 Countries/Regions by Daily New Deaths:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Represents the number of daily new deaths.
    - ✓ ***'Y-axis':*** Lists the country/region names.
  - ***Interpretation:*** This chart shows the countries with the highest daily death toll, helping track the severity of the pandemic.

❖ ***OUTPUT:***

```
<ipython-input-3-d35de57e32f4>:52: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and
will be removed in v0.14.0. Assign the `y` variable to `hue`
and set `legend=False` for the same effect.

  sns.barplot(x='New Deaths', y='Country/Region',
data=df_new_deaths_top_20, palette='magma')
```



4.7 Python Code for Analyzing New Daily Recoveries:

- **Python Code:**

```
# Data Visualization of COVID-19 New Recovered by
Country/Region
# This code visualizes the number of new COVID-19 recoveries
across different countries/regions using a bar chart.

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# COVID-19 Data for Countries/Regions (New Recovered)
data_new_recovered = {
    'Country/Region': ['Afghanistan', 'Albania', 'Algeria',
'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina',
'Armenia', 'Australia', 'Austria',
                       'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin', 'Bhutan', 'Bolivia',
                       'Bosnia and Herzegovina', 'Botswana',
'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma',
'Burundi', 'Cabo Verde',
                       'Cambodia', 'Cameroon', 'Canada',
'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Congo (Brazzaville)',
                       'Congo (Kinshasa)', 'Costa Rica', "Cote
d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
'Djibouti', 'Dominica',
                       'Dominican Republic', 'Ecuador',
'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
'Estonia', 'Eswatini', 'Ethiopia', 'Fiji',
                       'Finland', 'France', 'Gabon', 'Gambia',
'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenland',
'Grenada', 'Guatemala', 'Guinea',
                       'Guinea-Bissau', 'Guyana', 'Haiti',
'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Iran', 'Iraq', 'Ireland',
                       'Israel', 'Italy', 'Jamaica', 'Japan',
'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
                       'Lesotho', 'Liberia', 'Libya',
'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
'Malawi', 'Malaysia', 'Maldives', 'Mali',
```

```
                        'Malta', 'Mauritania', 'Mauritius',
'Mexico', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
'Morocco', 'Mozambique', 'Namibia', 'Nepal',
                        'Netherlands', 'New Zealand',
'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
'Oman', 'Pakistan', 'Panama', 'Papua New Guinea',
                        'Paraguay', 'Peru', 'Philippines',
'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
'Saint Kitts and Nevis', 'Saint Lucia',
                        'Saint Vincent and the Grenadines', 'San
Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
'Serbia', 'Seychelles', 'Sierra Leone',
                        'Singapore', 'Slovakia', 'Slovenia',
'Somalia', 'South Africa', 'South Korea', 'South Sudan',
'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
                        'Sweden', 'Switzerland', 'Syria',
'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',
'Togo', 'Trinidad and Tobago', 'Tunisia',
                        'Turkey', 'US', 'Uganda', 'Ukraine',
'United Arab Emirates', 'United Kingdom', 'Uruguay',
'Uzbekistan', 'Venezuela', 'Vietnam', 'West Bank and Gaza',
                        'Western Sahara', 'Yemen', 'Zambia',
'Zimbabwe'],
    'New Recovered': [18, 63, 749, 0, 0, 5, 2057, 187, 137, 37,
558, 0, 421, 1801, 0, 67, 14, 0, 0, 1, 309, 375, 11, 33728, 0,
230, 6, 2, 22, 103,
                      4, 0, 7.68, 1.28, 8.13, 1859, 7, 11494,
1.98, 73, 190, 88, 183, 70, 2, 0, 0, 77, 11, 0, 1601, 0, 1007,
130, 0, 2, 1, 39, 170, 0,
                      0, 267, 219, 6, 2, 259, 307, 0, 0, 0,
843, 105, 0, 0, 0, 0, 117, 0, 0, 33598, 1518, 1931, 1927, 0,
108, 147, 1.17, 3.2, 0.94, 0.69,
                      1.59, 2.5, 0.68, 3.91, 0, 2.54, 1.31,
2.38, 6.17, 2.26, 1.16, 3.96, 1.77, 0.94, 2.7, 1.39, 0.45,
4.93, 1.28, 2.51, 2.91, 11.13, 3.23,
                      3.45, 0, 1.56, 1.51, 0.65, 26, 626, 0, 1,
0, 0, 829, 137, 0, 1729, 3592, 955, 0, 111, 4697, 336, 103,
158, 304, 151, 3077, 57, 0, 0, 0,
                      0, 38, 2613, 68, 0, 0, 4, 171, 39, 55,
22, 9848, 102, 0, 0, 15, 49, 35, 0, 200, 0, 0, 58, 0, 2, 0, 8,
0, 15, 982, 27941, 4, 317,
                      328, 3, 3, 569, 213, 0, 0, 0, 36, 465,
24]
}

# Creating a DataFrame from the given data
df_new_recovered = pd.DataFrame(data_new_recovered)
```

```
# Sorting the DataFrame by 'New Recovered' in descending order
df_new_recovered_sorted = df_new_recovered.sort_values(by='New
Recovered', ascending=False)

# Select the top 20 countries with the most new recoveries
df_new_recovered_top_20 = df_new_recovered_sorted.head(20)

# Setting figure size for better visualization
plt.figure(figsize=(15, 10))

# Creating a bar plot to visualize new COVID-19 recoveries in
the top 20 countries
sns.barplot(x='New Recovered', y='Country/Region',
data=df_new_recovered_top_20, palette='magma')

# Adding a title to the plot
plt.title('Top 20 Countries with Most New COVID-19 Recoveries',
fontsize=16)

# Labeling the x-axis
plt.xlabel('New Recovered', fontsize=12)

# Labeling the y-axis
plt.ylabel('Country/Region', fontsize=12)

# Adjust layout for better spacing
plt.tight_layout()

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

- ***Code Explanation:***

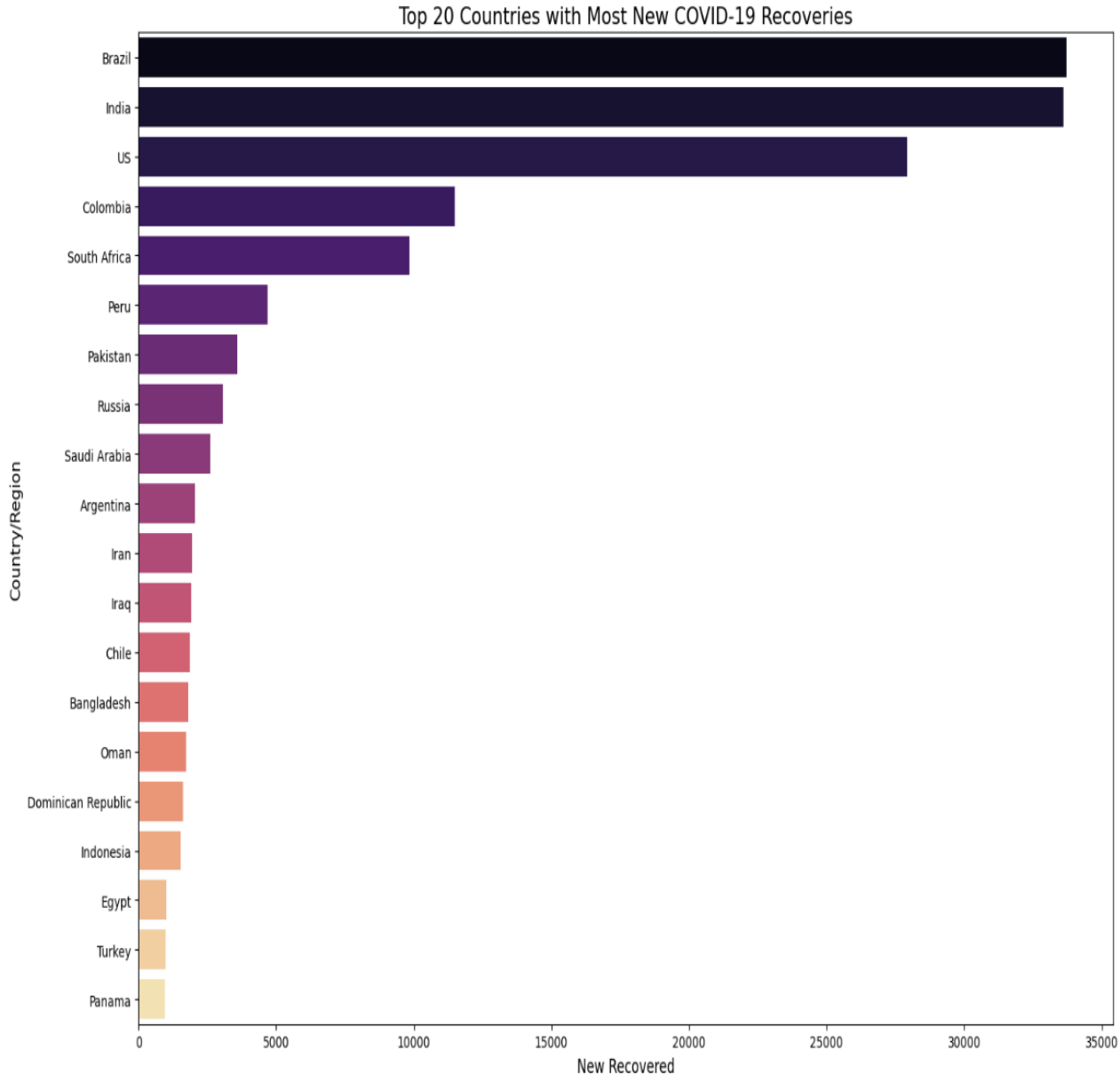
  - ***'data' Dictionary:*** Contains data on new daily recoveries.
    - ✓ Key: 'Country/Region' (List of countries/regions).
    - ✓ Key: '#New Recovered' (List of new daily recoveries).
  - ***'df' DataFrame:*** Manages the new recovery data.
  - ***'sns.barplot()':*** A bar chart visualizes the new recoveries for the top 20 countries.

- ***Output: Top 20 Countries/Regions by Daily New Deaths:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Shows the number of daily new recoveries.
    - ✓ ***'Y-axis':*** Displays the country/region names.
  - ***Interpretation:*** This chart shows which countries are seeing the highest rates of recovery on a daily basis, indicating the effectiveness of treatment methods.

❖ ***OUTPUT:***

```
<ipython-input-4-0fb7a6fa0ffb>:51: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and
will be removed in v0.14.0. Assign the `y` variable to `hue`
and set `legend=False` for the same effect.

  sns.barplot(x='New Recovered', y='Country/Region',
data=df_new_recovered_top_20, palette='magma')
```



4.8 Python Code for Mortality Rate Analysis: Deaths per 100 Confirmed Cases:

- **Python Code:**

```
# Data Visualization of COVID-19 Deaths per 100 Cases by
Country/Region
# This code visualizes the number of deaths per 100 cases
across different countries/regions using a bar chart.

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# COVID-19 Data for Countries/Regions (Deaths per 100 Cases)
data_deaths_per_100_cases = {
    'Country/Region': ['Afghanistan', 'Albania', 'Algeria',
'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina',
'Armenia', 'Australia', 'Austria',
                       'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin', 'Bhutan', 'Bolivia',
                       'Bosnia and Herzegovina', 'Botswana',
'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma',
'Burundi', 'Cabo Verde',
                       'Cambodia', 'Cameroon', 'Canada',
'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Congo (Brazzaville)',
                       'Congo (Kinshasa)', 'Costa Rica', "Cote
d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
'Djibouti', 'Dominica',
                       'Dominican Republic', 'Ecuador',
'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
'Estonia', 'Eswatini', 'Ethiopia', 'Fiji',
                       'Finland', 'France', 'Gabon', 'Gambia',
'Georgia', 'Germany', 'Ghana', 'Greece', 'Greenland',
'Grenada', 'Guatemala', 'Guinea',
                       'Guinea-Bissau', 'Guyana', 'Haiti',
'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Iran', 'Iraq', 'Ireland',
                       'Israel', 'Italy', 'Jamaica', 'Japan',
'Jordan', 'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait',
'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
                       'Lesotho', 'Liberia', 'Libya',
'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
'Malawi', 'Malaysia', 'Maldives', 'Mali',
```

```
                        'Malta', 'Mauritania', 'Mauritius',
'Mexico', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
'Morocco', 'Mozambique', 'Namibia', 'Nepal',
                        'Netherlands', 'New Zealand',
'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
'Oman', 'Pakistan', 'Panama', 'Papua New Guinea',
                        'Paraguay', 'Peru', 'Philippines',
'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
'Saint Kitts and Nevis', 'Saint Lucia',
                        'Saint Vincent and the Grenadines', 'San
Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
'Serbia', 'Seychelles', 'Sierra Leone',
                        'Singapore', 'Slovakia', 'Slovenia',
'Somalia', 'South Africa', 'South Korea', 'South Sudan',
'Spain', 'Sri Lanka', 'Sudan', 'Suriname',
                        'Sweden', 'Switzerland', 'Syria',
'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',
'Togo', 'Trinidad and Tobago', 'Tunisia',
                        'Turkey', 'US', 'Uganda', 'Ukraine',
'United Arab Emirates', 'United Kingdom', 'Uruguay',
'Uzbekistan', 'Venezuela', 'Vietnam', 'West Bank and Gaza',
                        'Western Sahara', 'Yemen', 'Zambia',
'Zimbabwe'],
    'Deaths per 100 Cases': [3.5, 2.95, 4.16, 5.73, 4.32, 3.49,
1.83, 1.9, 1.09, 3.47, 1.39, 2.88, 0.36, 1.31, 6.36, 0.8,
14.79, 4.17, 1.98, 0, 3.72,
                             2.8, 0.27, 3.59, 2.13, 3.27, 4.82,
1.71, 0.26, 0.95, 0, 2.29, 0, 33.62, 87.85, 2.64, 5.37, 3.41,
92.66, 1.69, 2.35, 0.73, 0.61,
                             2.85, 3.44, 1.79, 2.4, 4.45, 1.15,
0, 1.69, 6.82, 5.03, 2.71, 1.66, 0, 3.39, 1.47, 1.57, 0, 4.45,
13.71, 0.68, 2.45, 1.41,
                             4.41, 0.5, 4.78, 0, 0, 3.89, 0.64,
1.33, 5.14, 2.15, 0, 2.93, 13.4, 0.54, 2.26, 4.82, 5.42, 3.96,
6.81, 0.74, 14.26, 83.7,
                             70.55, 88.52, 64.27, 43.58, 54.32,
85.52, 63.69, 95, 85.73, 44.02, 25.35, 55.36, 20.41, 94.19,
80.24, 76.33, 64.6, 44.9, 96.6,
                             75.6, 76.12, 94.86, 74.95, 96.51,
76.82, 69.77, 89.66, 76.82, 27.96, 79.25, 0, 0.43, 0.26, 11.53,
1.41, 3.14, 6.1, 2.09, 4.56,
                             2.79, 0.51, 2.13, 2.15, 0, 0.95,
4.73, 2.37, 3.86, 3.42, 0.15, 4.81, 1.63, 0.27, 0, 0, 0, 6.01,
1.62, 1.03, 1.99, 2.25, 0,
                             3.7, 0.05, 1.28, 5.56, 2.91, 1.56,
2.11, 2, 10.44, 0.39, 6.3, 1.62, 7.18, 5.74, 5.93, 1.52, 0.83,
4.13, 1.76, 0, 2.06, 5.41,
```

```
                                3.44, 2.48, 3.45, 0.18, 2.44,
0.58, 15.19, 2.91, 0.57, 0.91, 0, 0.73, 10, 28.56, 3.08, 1.33]
}

# Creating a DataFrame from the given data
df_deaths_per_100_cases =
pd.DataFrame(data_deaths_per_100_cases)

# Sorting the DataFrame by 'Deaths per 100 Cases' in descending
order
df_deaths_per_100_cases_sorted =
df_deaths_per_100_cases.sort_values(by='Deaths per 100 Cases',
ascending=False)

# Select the top 20 countries with the most deaths per 100
cases
df_deaths_per_100_cases_top_20 =
df_deaths_per_100_cases_sorted.head(20)

# Setting figure size for better visualization
plt.figure(figsize=(15, 10))

# Creating a bar plot to visualize deaths per 100 cases in the
top 20 countries
sns.barplot(x='Deaths per 100 Cases', y='Country/Region',
data=df_deaths_per_100_cases_top_20, palette='magma')

# Adding a title to the plot
plt.title('Top 20 Countries with Most Deaths per 100 COVID-19
Cases', fontsize=16)

# Labeling the x-axis
plt.xlabel('Deaths per 100 Cases', fontsize=12)

# Labeling the y-axis
plt.ylabel('Country/Region', fontsize=12)

# Adjust layout for better spacing
plt.tight_layout()

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

- ***Code Explanation:***

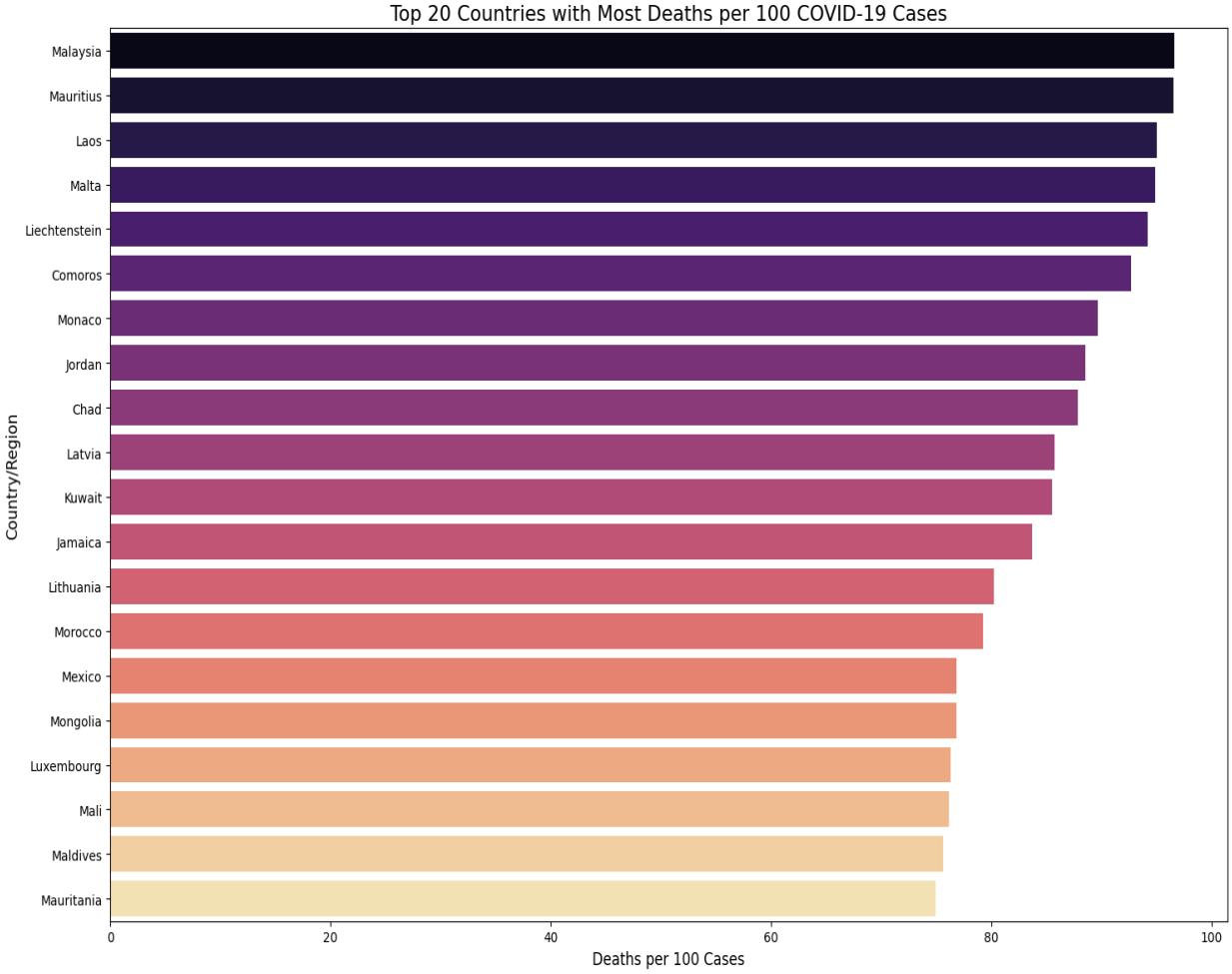
  - ***'data' Dictionary:*** Contains information on the number of deaths per 100 confirmed cases.
    - ✓ Key: 'Country/Region' (List of countries/regions).
    - ✓ Key: '#Deaths per 100 Cases' (List of death rates per 100 confirmed cases).
  - ***'df' DataFrame:*** This is used to store and analyze the mortality rate data.
  - ***'sns.barplot()':*** A bar plot visualizes the deaths per 100 confirmed cases for the top 20 countries.

- ***Output: Top 20 Countries/Regions by Daily New Deaths:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Shows the death rate per 100 confirmed cases.
    - ✓ ***'Y-axis':*** Displays the country/region names.
  - ***Interpretation:*** This chart helps us understand the fatality rate in relation to the confirmed cases, highlighting countries with high mortality rates.

❖ ***OUTPUT:***

```
<ipython-input-8-bb2082b7e502>:53: FutureWarning:

Passing `palette` without assigning `hue` is
deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the
same effect.

  sns.barplot(x='Deaths per 100 Cases',
y='Country/Region',
data=df_deaths_per_100_cases_top_20, palette='magma')
```



4.9 Python Code for Recovery Rate Analysis: Recoveries per 100 Confirmed Cases:

- **Python Code:**

```
# Data Visualization of COVID-19 Recoveries per 100 Cases by
Country/Region
# This code visualizes the number of recoveries per 100 cases
across different countries/regions using a bar chart.

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Updated COVID-19 Data for Countries/Regions (Recoveries per
100 Cases)
data_recoveries_per_100_cases = {
    'Country/Region': [
        'Afghanistan', 'Albania', 'Algeria', 'Andorra',
'Angola', 'Antigua and Barbuda', 'Argentina', 'Armenia',
'Australia', 'Austria',
        'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
'Bolivia',
        'Bosnia and Herzegovina', 'Botswana', 'Brazil',
'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma', 'Burundi', 'Cabo
Verde',
        'Cambodia', 'Cameroon', 'Canada', 'Central African
Republic', 'Chad', 'Chile', 'China', 'Colombia', 'Comoros',
'Congo (Brazzaville)',
        'Congo (Kinshasa)', 'Costa Rica', "Cote d'Ivoire",
'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark', 'Djibouti',
'Dominica',
        'Dominican Republic', 'Ecuador', 'Egypt', 'El
Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia',
'Eswatini', 'Ethiopia', 'Fiji',
        'Finland', 'France', 'Gabon', 'Gambia', 'Georgia',
'Germany', 'Ghana', 'Greece', 'Greenland', 'Grenada',
'Guatemala', 'Guinea',
        'Guinea-Bissau', 'Guyana', 'Haiti', 'Holy See',
'Honduras', 'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran',
'Iraq', 'Ireland',
        'Israel', 'Italy', 'Jamaica', 'Japan', 'Jordan',
'Kazakhstan', 'Kenya', 'Kosovo', 'Kuwait', 'Kyrgyzstan',
'Laos', 'Latvia', 'Lebanon',
```

```
        'Lesotho', 'Liberia', 'Libya', 'Liechtenstein',
'Lithuania', 'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia',
'Maldives', 'Mali',
        'Malta', 'Mauritania', 'Mauritius', 'Mexico',
'Moldova', 'Monaco', 'Mongolia', 'Montenegro', 'Morocco',
'Mozambique', 'Namibia', 'Nepal',
        'Netherlands', 'New Zealand', 'Nicaragua', 'Niger',
'Nigeria', 'North Macedonia', 'Norway', 'Oman', 'Pakistan',
'Panama', 'Papua New Guinea',
        'Paraguay', 'Peru', 'Philippines', 'Poland',
'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda', 'Saint
Kitts and Nevis', 'Saint Lucia',
        'Saint Vincent and the Grenadines', 'San Marino', 'Sao
Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
'Seychelles', 'Sierra Leone',
        'Singapore', 'Slovakia', 'Slovenia', 'Somalia', 'South
Africa', 'South Korea', 'South Sudan', 'Spain', 'Sri Lanka',
'Sudan', 'Suriname',
        'Sweden', 'Switzerland', 'Syria', 'Taiwan*',
'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste', 'Togo',
'Trinidad and Tobago', 'Tunisia',
        'Turkey', 'US', 'Uganda', 'Ukraine', 'United Arab
Emirates', 'United Kingdom', 'Uruguay', 'Uzbekistan',
'Venezuela', 'Vietnam', 'West Bank and Gaza',
        'Western Sahara', 'Yemen', 'Zambia', 'Zimbabwe'
    ],
    'Recoveries per 100 Cases': [
        69.49, 56.25, 67.34, 88.53, 25.47, 75.58, 43.35, 71.32,
60.84, 88.75, 76.34, 23.82, 91.46, 55.56, 85.45, 89.95, 26.27,
54.17,
        58.53, 86.87, 30.17, 46.96, 8.53, 75.61, 97.87, 52.58,
84.18, 83.43, 79.63, 66.58, 65.04, 84.97, 0, 0, 0, 91.96,
90.88,
        51.02, 0, 25.91, 64.45, 24.14, 66.18, 80.64, 92.85,
80.38, 73.65, 91.6, 98.38, 100, 47.08, 43, 37.67, 51.73, 27.42,
72.08,
        94.54, 44.26, 43.9, 66.67, 93.54, 36.86, 65.13, 20.25,
81.09, 91.89, 88.63, 32.51, 92.86, 100, 71.63, 88.69, 41.1,
46.53,
        59.47, 100, 12.68, 74.84, 98.33, 64.26, 58, 86.9,
68.52, 90.24, 42.41, 80.64, 83.7, 70.55, 88.52, 64.27, 43.58,
54.32, 85.52,
        63.69, 95, 85.73, 44.02, 25.35, 55.36, 20.41, 94.19,
80.24, 76.33, 64.6, 44.9, 96.6, 75.6, 76.12, 94.86, 74.95,
96.51,
        76.82, 69.77, 89.66, 76.82, 27.96, 79.25, 0, 5.48,
73.35, 0.35, 97.24, 72.46, 90.72, 44.2, 54.48, 95.84, 74.01,
87.87,
```

```
        57.1, 17.74, 63.87, 69.93, 32.24, 75.7, 70.33, 97.02,
56.19, 73.74, 51.89, 88.24, 91.67, 75, 93.99, 84.86, 82.9,
66.34,
        0, 34.21, 73.86, 89.88, 74.09, 83.04, 48.28, 60.75,
91.58, 50.98, 55.2, 75.61, 51.99, 62.37, 0, 89.62, 0, 95.24,
83.32,
        35.95, 94.36, 0, 69.45, 86.49, 79.52, 92.71, 30.9,
87.41, 55.45, 88.73, 0.48, 79.12, 55.04, 62.29, 84.69, 35.33,
80,
        49.26, 61.84, 20.04
    ]
}

# Creating a DataFrame from the updated data
df_recoveries_per_100_cases =
pd.DataFrame(data_recoveries_per_100_cases)

# Sorting the DataFrame by 'Recoveries per 100 Cases' in
descending order
df_recoveries_per_100_cases_sorted =
df_recoveries_per_100_cases.sort_values(by='Recoveries per 100
Cases', ascending=False)

# Select the top 20 countries with the most recoveries per 100
cases
df_recoveries_per_100_cases_top_20 =
df_recoveries_per_100_cases_sorted.head(20)

# Setting figure size for better visualization
plt.figure(figsize=(15, 10))

# Creating a bar plot to visualize recoveries per 100 cases in
the top 20 countries
sns.barplot(x='Recoveries per 100 Cases', y='Country/Region',
data=df_recoveries_per_100_cases_top_20, palette='viridis')

# Adding a title to the plot
plt.title('Top 20 Countries with Most Recoveries per 100
Cases', fontsize=16)

# Labeling the axes
plt.xlabel('Recoveries per 100 Cases (%)', fontsize=14)
plt.ylabel('Country/Region', fontsize=14)

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

- ***Code Explanation:***

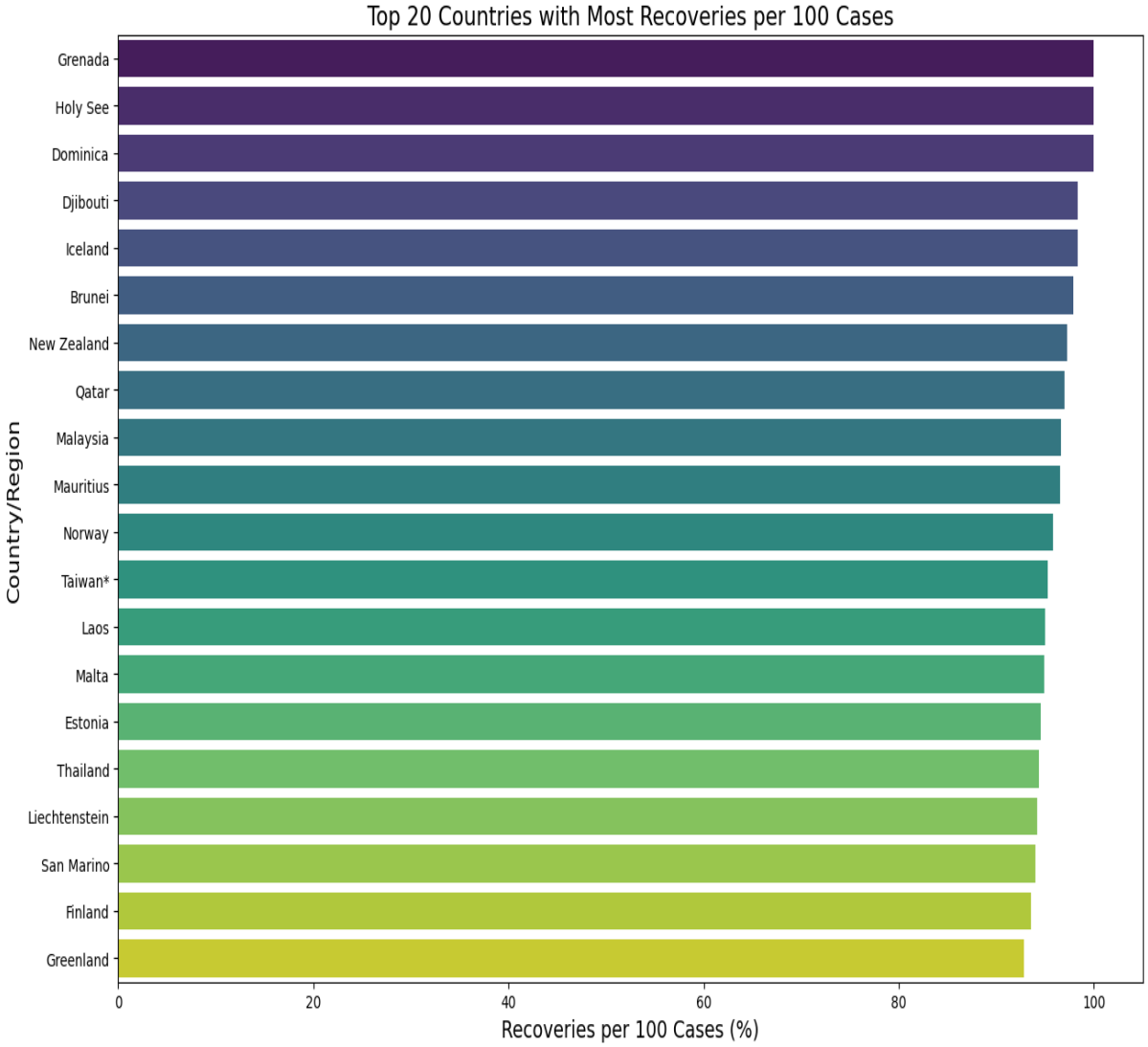
  - ***'data' Dictionary:*** Contains data for recoveries per 100 confirmed cases.
    - ✓ Key: 'Country/Region' (List of countries/regions).
    - ✓ Key: '#Recoveries per 100 Cases' (List of recovery rates per 100 confirmed cases).
  - ***'df' DataFrame:*** Manages the recovery rate data.
  - ***'sns.barplot()':*** A bar plot visualizes the recovery rate per 100 confirmed cases for the top 20 countries.

- ***Output: Top 20 Countries/Regions by Daily New Deaths:***

  - The output is a ***'bar chart'*** where:
    - ✓ ***'X-axis':*** Displays the recovery rate per 100 confirmed cases.
    - ✓ ***'Y-axis':*** Displays the country/region names.
  - ***Interpretation:*** This chart shows how many recoveries are made per 100 confirmed cases, which is useful in accessing the recovery efficiency of different regions.

❖ ***OUTPUT:***

```
<ipython-input-15-fde1bb882e9e>:59: FutureWarning:

Passing `palette` without assigning `hue` is
deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the
same effect.

  sns.barplot(x='Recoveries per 100 Cases',
y='Country/Region',
data=df_recoveries_per_100_cases_top_20,
palette='viridis')
```



4.10 Python Code for Summary of Key COVID-19 Data Across Countries:

This section of the code generates a summary of key COVID-19 data across multiple countries. It computes the total confirmed cases, deaths, recoveries, and the average death and recovery rates, then visualizes this data in bar charts. The code also highlights the countries with the highest confirmed cases and the highest recovery rate.

4.10.1 Python Code Explanation for Summary of Key COVID-19 Data:

- **Importing Libraries:**
  - **'import pandas as pd':** This imports the Pandas library, which is used for data manipulation and analysis, especially handling the data in a tabular format.
  - **'import matplotlib.pyplot as plt':** This imports the Matplotlib library, which is used for data visualization, allowing the generation of charts.

- **Sample Data:**
  - The data is provided as a string containing CSV-like data, with columns for country/region, confirmed cases, deaths, recoveries, active cases, new cases, new deaths, new recoveries,

deaths per 100 cases, and recoveries per 100 cases.

- ***Reading Data into a DataFrame:***

  - ***'df = pd.read_csv(StringIO(data)):*** The raw data is converted into a Pandas DaatFrame, which organizes the data into a table-like structure for easier analysis.

- ***Summary Calculations:***

  - ***'total_confirmed = df['#CONFIRMED'].sum()':*** This calculated the total number of confirmed cases by summing th values in the ***'#CONFIRMED'*** column.
  - ***'total_deaths = df['#DEATHS'].sum()':*** This calculates the total number of deaths by summing the values in the ***'#DEATHS'*** column.
  - ***'total_recoveries = df['#RECOVERED'].sum()':*** This calculates the total number of recoveries by summing the values in the ***'#RECOVERED'*** column.
  - ***'avg_death_rate = df['#DEATHS/100 CASES'].mean()':*** This calculates the average death rate across all countries by computing the mean of the ***'#DEATHS/100 CASES'*** column.
  - ***'avg_recovery_rate = df['#RECOVERED/100 CASES'].mean()':*** This calculates the average recovery rate across all countries by computing the mean of the ***'#RECOVERED/100 CASES'*** column.

- ***Identifying Countries with Maximum Values:***

  - ***'max_confirmed_country = df.loc[df['#CONFIMED].idxmax(), 'COUNTRY/REGION']:*** This identifies the country with the highest confirmed cases by finding the index of the maximum value in the ***'#CONFIRMED'*** column and returning the corresponding country name.
  - ***'max_recovery_country = df.loc[df['#RECOVERED/100 CASES'].idxmax(), 'COUNTRY/REGION']':*** This identifies the country with the highest recovery rate by finding the index of the maximum value in the ***'#RECOVERED/100 CASES'*** column and returning the corresponding country name.

- ***Storing Results:***

  - The calculated summary results, including the total values and countries with the highest numbers, are stored in a dictionary named ***'summary'***.

- ***Bar Chart for Total Counts:***

  - A horizontal bar chart is plotted for the total confirmed cases, total deaths, and total recoveries using ***'matplotlib'***. The counts are displayed on the x-axis, and the categories

(confirmed cases, deaths, recoveries) are shown on the y-axis.

- ***Bar Chart for Average Rates:***

  - Another horizontal bar chart is plotted for the average death rate and average recovery rate, with the percentage displayed on the x-axis.

4.10.2 Output Visualization and Explanation:

- ***Summary of Key COVID-19 Data Across Countries:***

This section visualizes the key statistics for COVID-19 across countries, helping to provide an overview of the current global situation in terms of total cases, deaths, recoveries, and average rates.

- ***Output-1: Total Counts of COVID-19 Data:***
  - The first bar chart visualizes the ***Total Confirmed Cases***, ***Total Deaths***, and ***Total Recoveries*** across all countries.
    I. ***Total Confirmed Cases:*** The cumulative total of all confirmed COVID-19 cases across the countries in the dataset.
    II. ***Total Deaths:*** The total number of COVID-19 related deaths.
    III. ***Total Recoveries:*** The total number of people who have recovered from COVID-19.

- ***Output-2: Average Death Rate and Recovery Rate:***
  - The second bar chart visualizes the ***Average Death Rate*** and ***Average Recovery Rate*** across all countries in the dataset.
    - ***I. Average Death Rate (%):*** The mean percentage of deaths relative to confirmed cases across the countries.
    - ***II. Average Recovery Rate (%):*** The mean percentage of recoveries relative to confirmed cases across the countries.

These visualizations offer a concise, easy-to-understand overview of key metric related to COVID-19 across various countries. By presenting the data in graphical form, we can quickly interpret complex figures, which might be challenging to understand if presented as raw number.

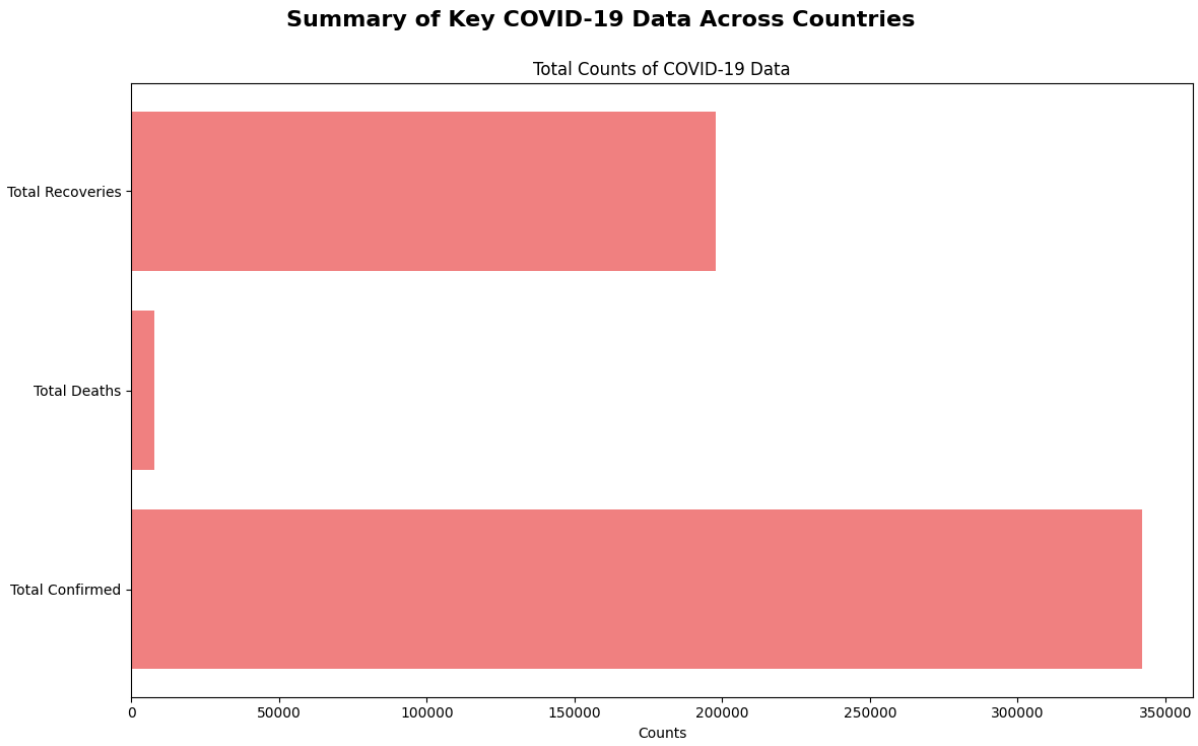
- ***Python Code:***

```
#SUMMARY OF KEY COVID-19 DATA ACROSS COUNTRIES
#THIS IS A SUMMARIZED VISUAL WHICH DISPLAYS ALL THE PREVIOUS
VISUALS IN A COMBINED MANNER

import pandas as pd
import matplotlib.pyplot as plt

# Sample data
data = """
COUNTRY/REGION,#CONFIRMED,#DEATHS,#RECOVERED,#ACTIVE,#NEW
CASES,#NEW DEATHS,#NEW RECOVERED,#DEATHS/100
CASES,#RECOVERED/100 CASES
Afghanistan,36263,1269,25198,9796,106,10,18,3.5,69.49
Albania,4880,144,2745,1991,117,6,63,2.95,56.25
Algeria,27973,1163,18837,7973,616,8,749,4.16,67.34
Andorra,907,52,803,52,10,0,0,5.73,88.53
Angola,950,41,242,667,18,1,0,4.32,25.47
Antigua and Barbuda,86,3,65,18,4,0,5,3.49,75.58
Argentina,167416,3059,72575,91782,4890,120,2057,1.83,43.35
Armenia,37390,711,26665,10014,73,6,187,1.9,71.32
Australia,15303,167,9311,5825,368,6,137,1.09,60.84
Austria,20558,713,18246,1599,86,1,37,3.47,88.75
Azerbaijan,30446,423,23242,6781,396,6,558,1.39,76.34
"""

# Reading the data into a DataFrame
from io import StringIO

df = pd.read_csv(StringIO(data))

# Calculating the summary
total_confirmed = df['#CONFIRMED'].sum()
total_deaths = df['#DEATHS'].sum()
total_recoveries = df['#RECOVERED'].sum()
avg_death_rate = (df['#DEATHS/100 CASES'].mean())
avg_recovery_rate = (df['#RECOVERED/100 CASES'].mean())

# Finding the country with the highest values
max_confirmed_country = df.loc[df['#CONFIRMED'].idxmax(),
'COUNTRY/REGION']
max_recovery_country = df.loc[df['#RECOVERED/100
CASES'].idxmax(), 'COUNTRY/REGION']

summary = {
    "Total Confirmed": total_confirmed,
```

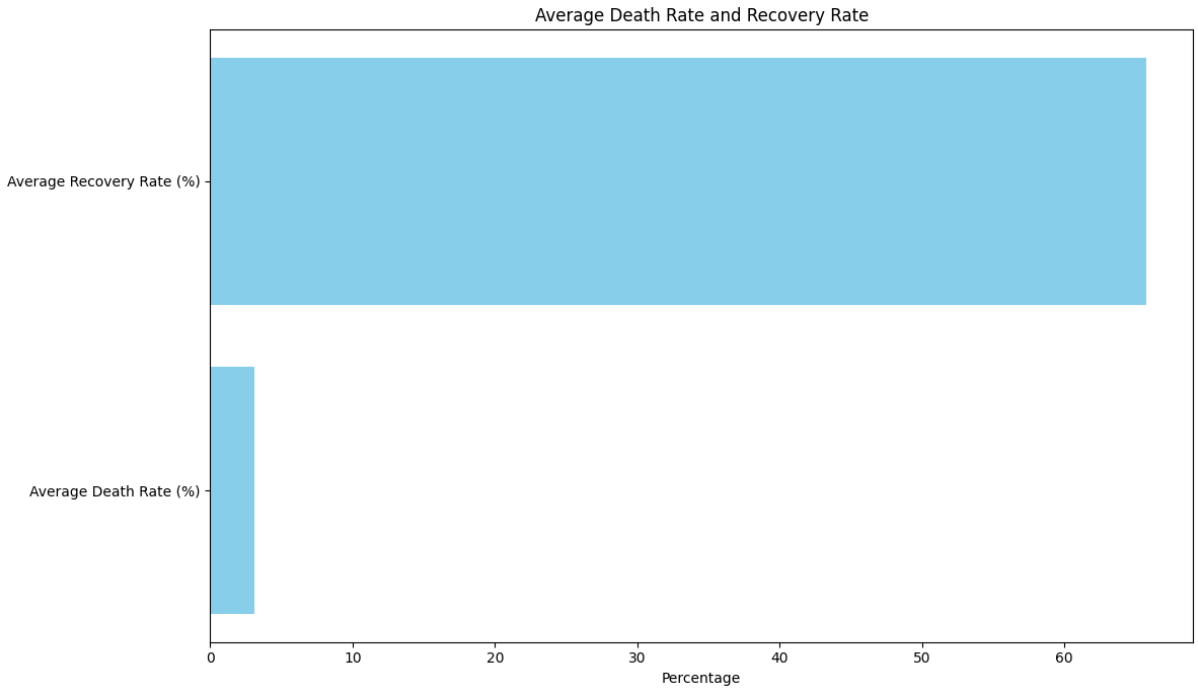
```
    "Total Deaths": total_deaths,
    "Total Recoveries": total_recoveries,
    "Average Death Rate (%)": avg_death_rate,
    "Average Recovery Rate (%)": avg_recovery_rate,
    "Country with Highest Confirmed Cases":
max_confirmed_country,
    "Country with Highest Recovery Rate": max_recovery_country
}

# Plotting the count-based data
fig, ax = plt.subplots(figsize=(12, 7))  # Adjusting figure
size to avoid clutter
count_data = [total_confirmed, total_deaths, total_recoveries]
count_labels = ['Total Confirmed', 'Total Deaths', 'Total
Recoveries']

ax.barh(count_labels, count_data, color='lightcoral')
ax.set_xlabel('Counts')
ax.set_title('Total Counts of COVID-19 Data')
plt.tight_layout()

# Add a super heading across both charts
plt.suptitle('Summary of Key COVID-19 Data Across Countries',
fontsize=16, fontweight='bold', y=1.05)

# Show the first plot
plt.show()

# Plotting the percentage-based data
fig, ax = plt.subplots(figsize=(12, 7))  # Adjusting figure
size for clarity
percentage_data = [avg_death_rate, avg_recovery_rate]
percentage_labels = ['Average Death Rate (%)', 'Average
Recovery Rate (%)']

ax.barh(percentage_labels, percentage_data, color='skyblue')
ax.set_xlabel('Percentage')
ax.set_title('Average Death Rate and Recovery Rate')
plt.tight_layout()

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

❖ ***OUTPUT:***

❖ ***Output-1: Total Counts of COVID-19 Data:***



❖ ***Output-2: Average Death Rate and Recovery Rate:***



5. VISUALIZING DATA: A GLIMPSE INTO THE RAW EXCEL DATASHEET

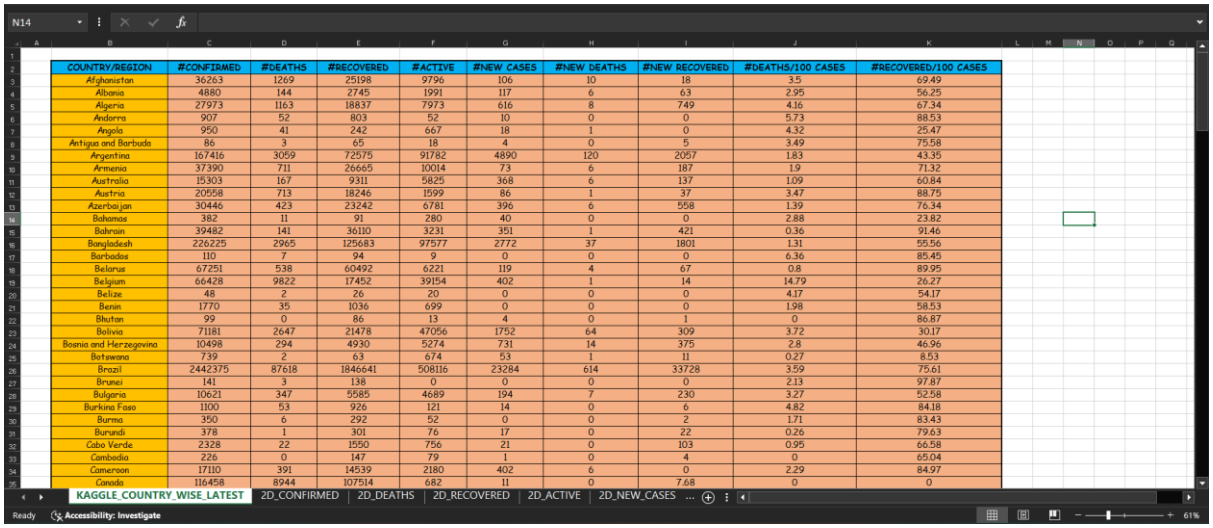
The data utilized for this project is organized into multiple sheets within the Excel file, each providing insights and visualizations for various aspects of the COVID-19 pandemic across different countries and regions. Below is an overview of each sheet and its respective content:

- ***Sheet-1:*** ***KAGGLE_COUNTRY_WISE_LATEST***

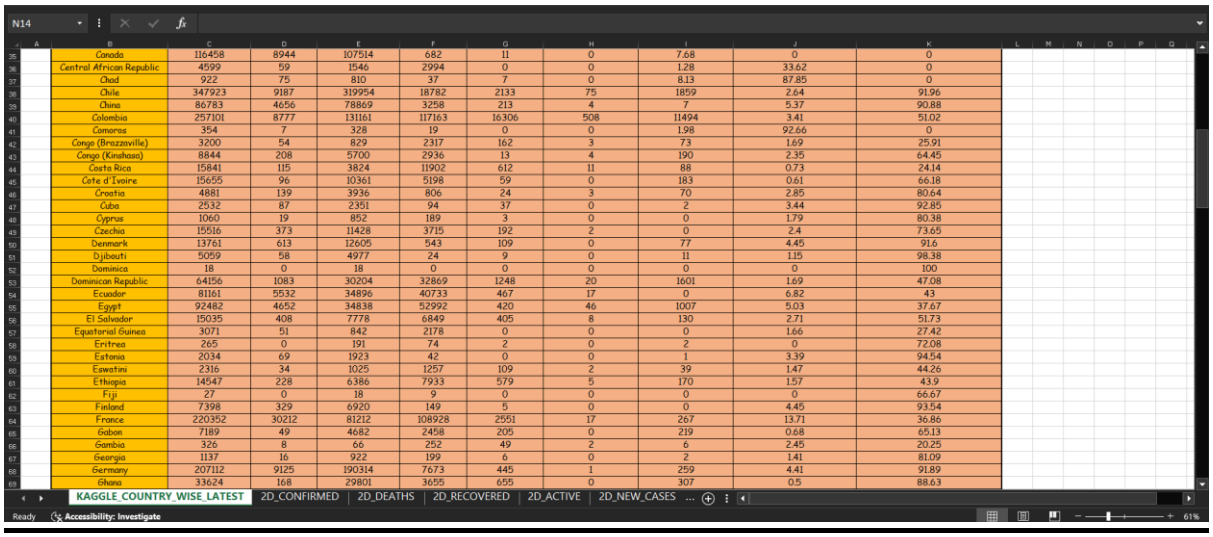
  This sheet contains the most up-to-date and comprehensive dataset of COVID-19 statistics for various countries and regions. It includes key metric such as:

  - Confirmed Cases
  - Deaths
  - Recoveries
  - Active Cases
  - New Cases
  - New Deaths
  - New Recoveries
  - Deaths per 100 Cases
  - Recoveries per 100 Cases

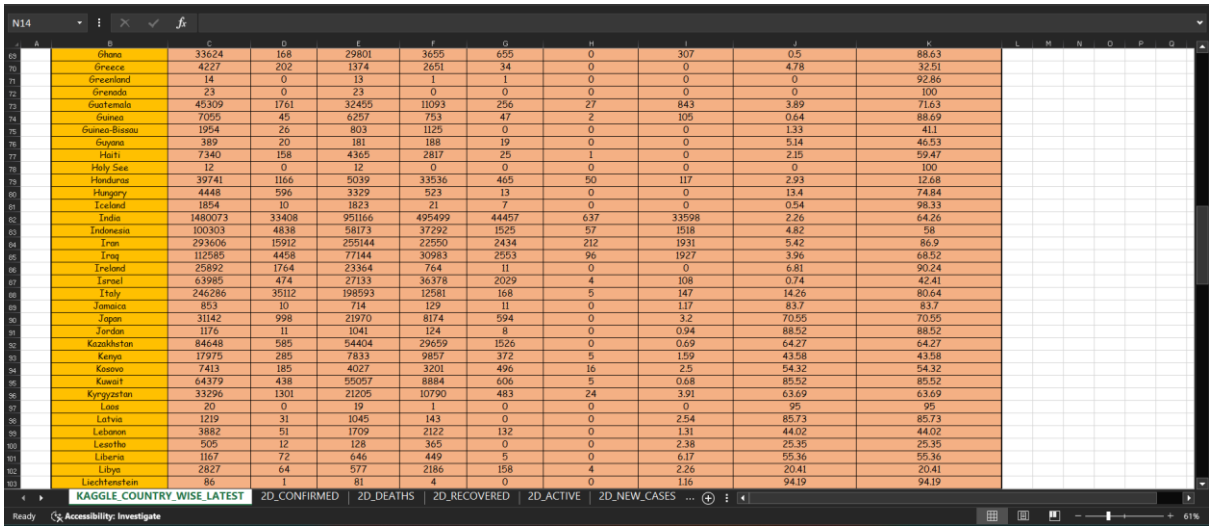
  This data serves as the foundational input for all the subsequent analysis and visualizations in the report.



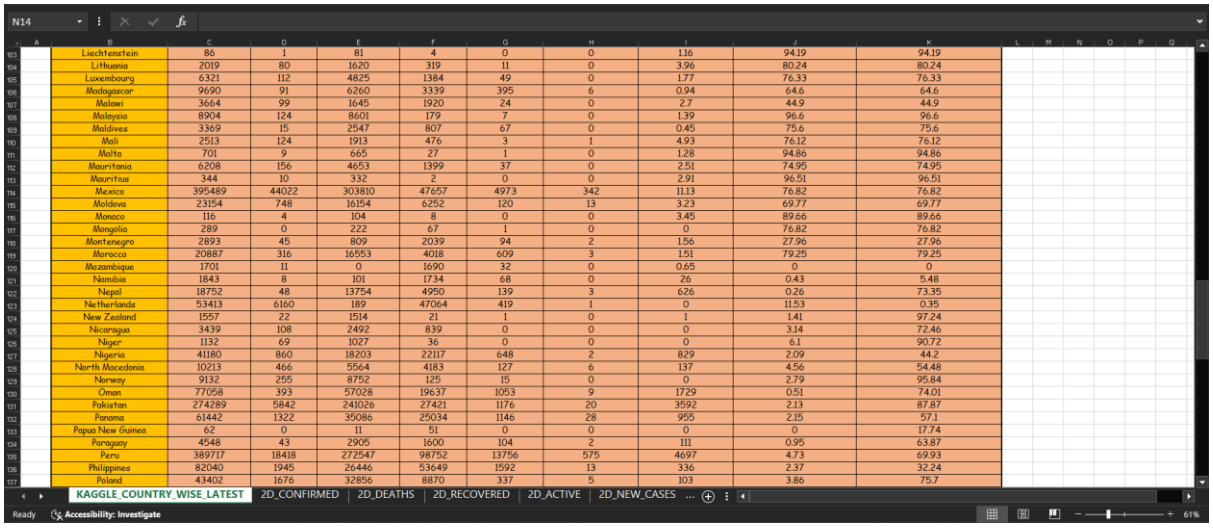
COUNTRY/REGION	#CONFIRMED	#DEATHS	#RECOVERED	#ACTIVE	#NEW CASES	#NEW DEATHS	#NEW RECOVERED	#DEATHS/100 CASES	#RECOVERED/100 CASES
Afghanistan	36263	1269	25198	9796	106	10	18	3.5	69.49
Albania	4880	144	2745	1991	117	6	63	2.95	56.25
Algeria	27973	1163	18837	7973	616	8	749	4.16	67.34
Andorra	907	52	803	52	10	0	0	5.73	88.53
Angola	950	41	242	667	18	1	0	4.32	25.47
Antigua and Barbuda	86	3	65	18	4	0	5	3.49	75.58
Argentina	167416	3059	72575	91782	4890	120	2057	1.83	43.35
Armenia	37390	711	26665	10014	73	6	187	1.9	71.32
Australia	15303	167	9311	5825	368	6	137	1.09	60.84
Austria	20558	713	18246	1599	86	1	37	3.47	88.75
Azerbaijan	30446	423	23242	6781	396	6	558	1.39	76.34
Bahamas	382	11	91	280	40	0	0	2.88	23.82
Bahrain	39482	141	36110	3231	351	1	421	0.36	91.46
Bangladesh	226225	2965	125683	97577	2772	37	1801	1.31	55.56
Barbados	110	7	94	9	0	0	0	6.36	85.45
Belarus	67251	538	60492	6221	119	4	67	0.8	89.95
Belgium	66428	9822	17452	39154	402	1	14	14.79	26.27
Belize	48	2	26	20	0	0	0	4.17	54.17
Benin	1770	35	1036	699	0	0	0	1.98	58.53
Bhutan	99	0	86	13	4	0	1	0	86.87
Bolivia	71181	2647	21478	47056	1752	64	309	3.72	30.17
Bosnia and Herzegovina	10498	294	4930	5274	731	14	375	2.8	46.96
Botswana	739	2	63	674	53	1	11	0.27	8.53
Brazil	2442375	87618	1846641	508116	23284	614	33728	3.59	75.61
Brunei	141	3	138	0	0	0	0	2.13	97.87
Bulgaria	10621	347	5585	4689	194	7	230	3.27	52.58
Burkina Faso	1100	53	926	121	14	0	6	4.82	84.18
Burma	350	6	292	52	0	0	2	1.71	83.43
Burundi	378	1	301	76	17	0	22	0.26	79.63
Cabo Verde	2328	22	1550	756	21	0	103	0.95	66.58
Cambodia	226	0	147	79	1	0	4	0	65.04
Cameroon	17110	391	14539	2180	402	6	0	2.29	84.97
Canada	116458	8944	107514	682	11	0	7.68	0	0



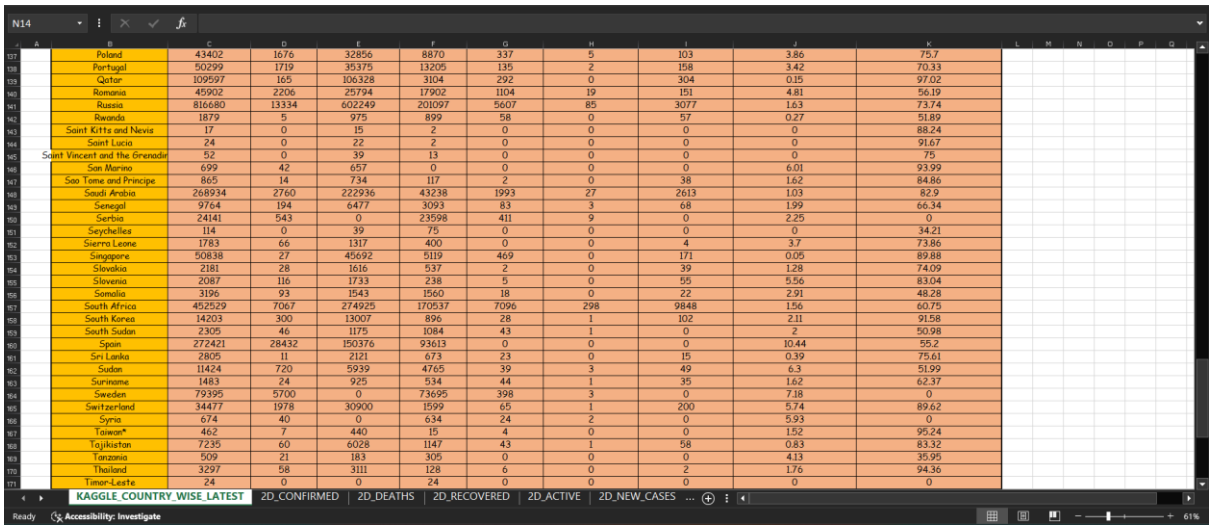
Canada	116458	8944	107514	682	11	0	7.68	0	0
Central African Republic	4599	59	1546	2994	0	0	1.28	33.62	0
Chad	922	75	810	37	7	0	8.13	87.85	0
Chile	347923	9187	319954	18782	2133	75	1859	2.64	91.96
China	86783	4656	78869	3258	213	4	7	5.37	90.88
Colombia	257101	8777	131161	117163	16306	508	11494	3.41	51.02
Comoros	354	7	328	19	0	0	1.98	92.66	0
Congo (Brazzaville)	3200	54	829	2317	162	3	73	1.69	25.91
Congo (Kinshasa)	8844	208	5700	2936	13	4	190	2.35	64.45
Costa Rica	15841	115	3824	11902	612	11	88	0.73	24.14
Cote d'Ivoire	15655	96	10361	5198	59	0	183	0.61	66.18
Croatia	4881	139	3936	806	24	3	70	2.85	80.64
Cuba	2532	87	2351	94	37	0	2	3.44	92.85
Cyprus	1060	19	852	189	3	0	0	1.79	80.38
Czechia	15516	373	11428	3715	192	2	0	2.4	73.65
Denmark	13761	613	12605	543	109	0	77	4.45	91.6
Djibouti	5059	58	4977	24	9	0	11	1.15	98.38
Dominica	18	0	18	0	0	0	0	0	100
Dominican Republic	64156	1083	30204	32869	1248	20	1601	1.69	47.08
Ecuador	81161	5532	34896	40733	467	17	0	6.82	43
Egypt	92482	4652	34838	52992	420	46	1007	5.03	37.67
El Salvador	15035	408	7778	6849	405	8	130	2.71	51.73
Equatorial Guinea	3071	51	842	2178	0	0	0	1.66	27.42
Eritrea	265	0	191	74	2	0	2	0	72.08
Estonia	2034	69	1923	42	0	0	1	3.39	94.54
Eswatini	2316	34	1025	1257	109	2	39	1.47	44.26
Ethiopia	14547	228	6386	7933	579	5	170	1.57	43.9
Fiji	27	0	18	9	0	0	0	0	66.67
Finland	7398	329	6920	149	5	0	0	4.45	93.54
France	220352	30212	81212	108928	2551	17	267	13.71	36.86
Gabon	7189	49	4682	2458	205	0	219	0.68	65.13
Gambia	326	8	66	252	49	2	6	2.45	20.25
Georgia	1137	16	922	199	6	0	2	1.41	81.09
Germany	207112	9125	190314	7673	445	1	259	4.41	91.89
Ghana	33624	168	29801	3655	655	0	307	0.5	88.63



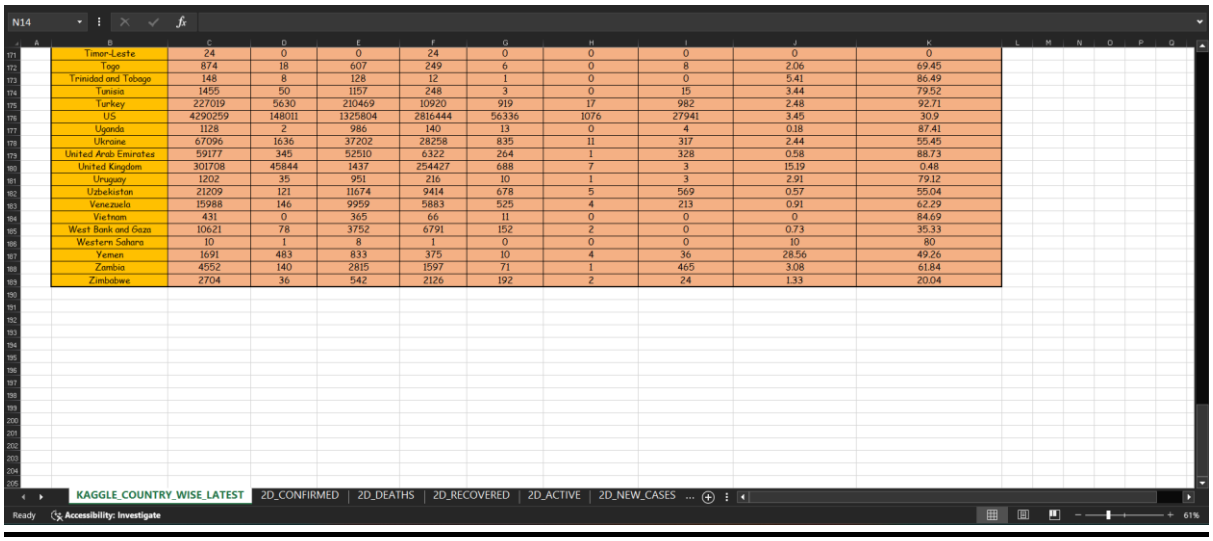
Ghana	33624	168	29801	3655	655	0	307	0.5	88.63
Greece	4227	202	1374	2651	34	0	0	4.78	32.51
Greenland	14	0	13	1	1	0	0	0	92.86
Grenada	23	0	23	0	0	0	0	0	100
Guatemala	45309	1761	32455	11093	256	27	843	3.89	71.63
Guinea	7055	45	6257	753	47	2	105	0.64	88.69
Guinea-Bissau	1954	26	803	1125	0	0	0	1.33	41.1
Guyana	389	20	181	188	19	0	0	5.14	46.53
Haiti	7340	158	4365	2817	25	1	0	2.15	59.47
Holy See	12	0	12	0	0	0	0	0	100
Honduras	39741	1166	5039	33536	465	50	117	2.93	12.68
Hungary	4448	596	3329	523	13	0	0	13.4	74.84
Iceland	1854	10	1823	21	7	0	0	0.54	98.33
India	1480073	33408	951166	495499	44457	637	33598	2.26	64.26
Indonesia	100303	4838	58173	37292	1525	57	1518	4.82	58
Iran	293606	15912	255144	22550	2434	212	1931	5.42	86.9
Iraq	112585	4458	77144	30983	2553	96	1927	3.96	68.52
Ireland	25892	1764	23364	764	11	0	0	6.81	90.24
Israel	63985	474	27133	36378	2029	4	108	0.74	42.41
Italy	246286	35112	198593	12581	168	5	147	14.26	80.64
Jamaica	853	10	714	129	11	0	1.17	83.7	83.7
Japan	31142	998	21970	8174	594	0	3.2	70.55	70.55
Jordan	1176	11	1041	124	8	0	0.94	88.52	88.52
Kazakhstan	84648	585	54404	29659	1526	0	0.69	64.27	64.27
Kenya	17975	285	7833	9857	372	5	1.59	43.58	43.58
Kosovo	7413	185	4027	3201	496	16	2.5	54.32	54.32
Kuwait	64379	438	55057	8884	606	5	0.68	85.52	85.52
Kyrgyzstan	33296	1301	21205	10790	483	24	3.91	63.69	63.69
Laos	20	0	19	1	0	0	0	95	95
Latvia	1219	31	1045	143	0	0	2.54	85.73	85.73
Lebanon	3882	51	1709	2122	132	0	1.31	44.02	44.02
Lesotho	505	12	128	365	0	0	2.38	25.35	25.35
Liberia	1167	72	646	449	5	0	6.17	55.36	55.36
Libya	2827	64	577	2186	158	4	2.26	20.41	20.41
Liechtenstein	86	1	81	4	0	0	1.16	94.19	94.19



Liechtenstein	86	1	81	4	0	0	1.16	94.19	94.19
Lithuania	2019	80	1620	319	11	0	3.96	80.24	80.24
Luxembourg	6321	112	4825	1384	49	0	1.77	76.33	76.33
Madagascar	9690	91	6260	3339	395	6	0.94	64.6	64.6
Malawi	3664	99	1645	1920	24	0	2.7	44.9	44.9
Malaysia	8904	124	8601	179	7	0	1.39	96.6	96.6
Maldives	3369	15	2547	807	67	0	0.45	75.6	75.6
Mali	2513	124	1913	476	3	1	4.93	76.12	76.12
Malta	701	9	665	27	1	0	1.28	94.86	94.86
Mauritania	6208	156	4653	1399	37	0	2.51	74.95	74.95
Mauritius	344	10	332	2	0	0	2.91	96.51	96.51
Mexico	395489	44022	303810	47657	4973	342	11.13	76.82	76.82
Moldova	23154	748	16154	6252	120	13	3.23	69.77	69.77
Monaco	116	4	104	8	0	0	3.45	89.66	89.66
Mongolia	289	0	222	67	1	0	0	76.82	76.82
Montenegro	2893	45	809	2039	94	2	1.56	27.96	27.96
Morocco	20887	316	16553	4018	609	3	1.51	79.25	79.25
Mozambique	1701	11	0	1690	32	0	0.65	0	0
Namibia	1843	8	101	1734	68	0	26	0.43	5.48
Nepal	18752	48	13754	4950	139	3	626	0.26	73.35
Netherlands	53413	6160	189	47064	419	1	0	11.53	0.35
New Zealand	1557	22	1514	21	1	0	1	1.41	97.24
Nicaragua	3439	108	2492	839	0	0	0	3.14	72.46
Niger	1132	69	1027	36	0	0	0	6.1	90.72
Nigeria	41180	860	18203	22117	648	2	829	2.09	44.2
North Macedonia	10213	466	5564	4183	127	6	137	4.56	54.48
Norway	9132	255	8752	125	15	0	0	2.79	95.84
Oman	77058	393	57028	19637	1053	9	1729	0.51	74.01
Pakistan	274289	5842	241026	27421	1176	20	3592	2.13	87.87
Panama	61442	1322	35086	25034	1146	28	955	2.15	57.1
Papua New Guinea	62	0	11	51	0	0	0	0	17.74
Paraguay	4548	43	2905	1600	104	2	111	0.95	63.87
Peru	389717	18418	272547	98752	13756	575	4697	4.73	69.93
Philippines	82040	1945	26446	53649	1592	13	336	2.37	32.24
Poland	43402	1676	32856	8870	337	5	103	3.86	75.7



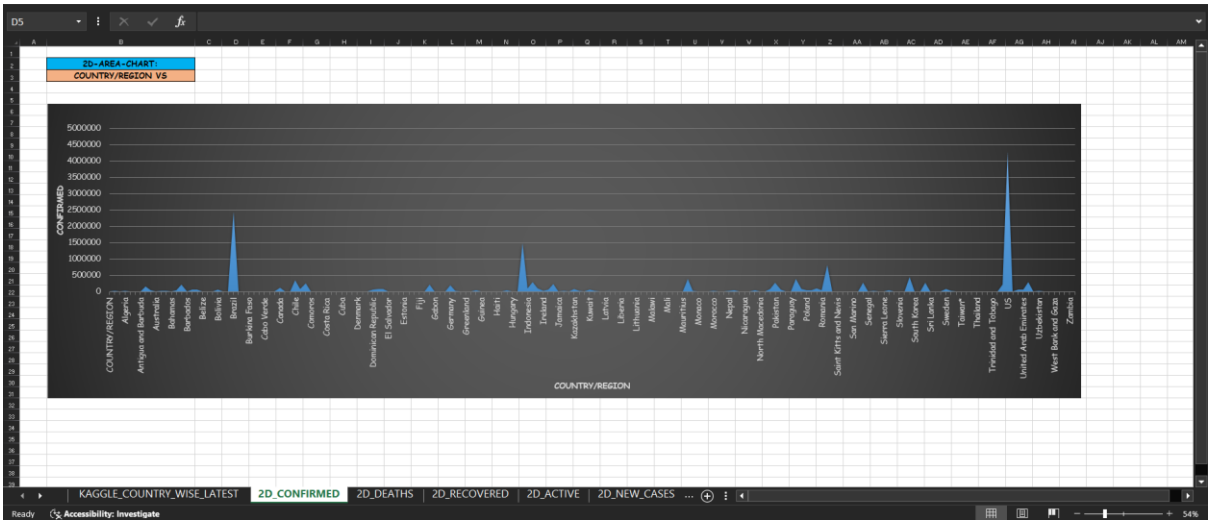
Poland	43402	1676	32856	8870	337	5	103	3.86	75.7
Portugal	50299	1719	35375	13205	135	2	158	3.42	70.33
Qatar	109597	165	106328	3104	292	0	304	0.15	97.02
Romania	45902	2206	25794	17902	1104	19	151	4.81	56.19
Russia	816680	13334	602249	201097	5607	85	3077	1.63	73.74
Rwanda	1879	5	975	899	58	0	57	0.27	51.89
Saint Kitts and Nevis	17	0	15	2	0	0	0	0	88.24
Saint Lucia	24	0	22	2	0	0	0	0	91.67
Saint Vincent and the Grenadir	52	0	39	13	0	0	0	0	75
San Marino	699	42	657	0	0	0	0	6.01	93.99
Sao Tome and Principe	865	14	734	117	2	0	38	1.62	84.86
Saudi Arabia	268934	2760	222936	43238	1993	27	2613	1.03	82.9
Senegal	9764	194	6477	3093	83	3	68	1.99	66.34
Serbia	24141	543	0	23598	411	9	0	2.25	0
Seychelles	114	0	39	75	0	0	0	0	34.21
Sierra Leone	1783	66	1317	400	0	0	4	3.7	73.86
Singapore	50838	27	45692	5119	469	0	171	0.05	89.88
Slovakia	2181	28	1616	537	2	0	39	1.28	74.09
Slovenia	2087	116	1733	238	5	0	55	5.56	83.04
Somalia	3196	93	1543	1560	18	0	22	2.91	48.28
South Africa	452529	7067	274925	170537	7096	298	9848	1.56	60.75
South Korea	14203	300	13007	896	28	1	102	2.11	91.58
South Sudan	2305	46	1175	1084	43	1	0	2	50.98
Spain	272421	28432	150376	93613	0	0	0	10.44	55.2
Sri Lanka	2805	11	2121	673	23	0	15	0.39	75.61
Sudan	11424	720	5939	4765	39	3	49	6.3	51.99
Suriname	1483	24	925	534	44	1	35	1.62	62.37
Sweden	79395	5700	0	73695	398	3	0	7.18	0
Switzerland	34477	1978	30900	1599	65	1	200	5.74	89.62
Syria	674	40	0	634	24	2	0	5.93	0
Taiwan*	462	7	440	15	4	0	0	1.52	95.24
Tajikistan	7235	60	6028	1147	43	1	58	0.83	83.32
Tanzania	509	21	183	305	0	0	0	4.13	35.95
Thailand	3297	58	3111	128	6	0	2	1.76	94.36
Timor-Leste	24	0	0	24	0	0	0	0	0



Timor-Leste	24	0	0	24	0	0	0	0	0
Togo	874	18	607	249	6	0	8	2.06	69.45
Trinidad and Tobago	148	8	128	12	1	0	0	5.41	86.49
Tunisia	1455	50	1157	248	3	0	15	3.44	79.52
Turkey	227019	5630	210469	10920	919	17	982	2.48	92.71
US	4290259	148011	1325804	2816444	56336	1076	27941	3.45	30.9
Uganda	1128	2	986	140	13	0	4	0.18	87.41
Ukraine	67096	1636	37202	28258	835	11	317	2.44	55.45
United Arab Emirates	59177	345	52510	6322	264	1	328	0.58	88.73
United Kingdom	301708	45844	1437	254427	688	7	3	15.19	0.48
Uruguay	1202	35	951	216	10	1	3	2.91	79.12
Uzbekistan	21209	121	11674	9414	678	5	569	0.57	55.04
Venezuela	15988	146	9959	5883	525	4	213	0.91	62.29
Vietnam	431	0	365	66	11	0	0	0	84.69
West Bank and Gaza	10621	78	3752	6791	152	2	0	0.73	35.33
Western Sahara	10	1	8	1	0	0	0	10	80
Yemen	1691	483	833	375	10	4	36	28.56	49.26
Zambia	4552	140	2815	1597	71	1	465	3.08	61.84
Zimbabwe	2704	36	542	2126	192	2	24	1.33	20.04

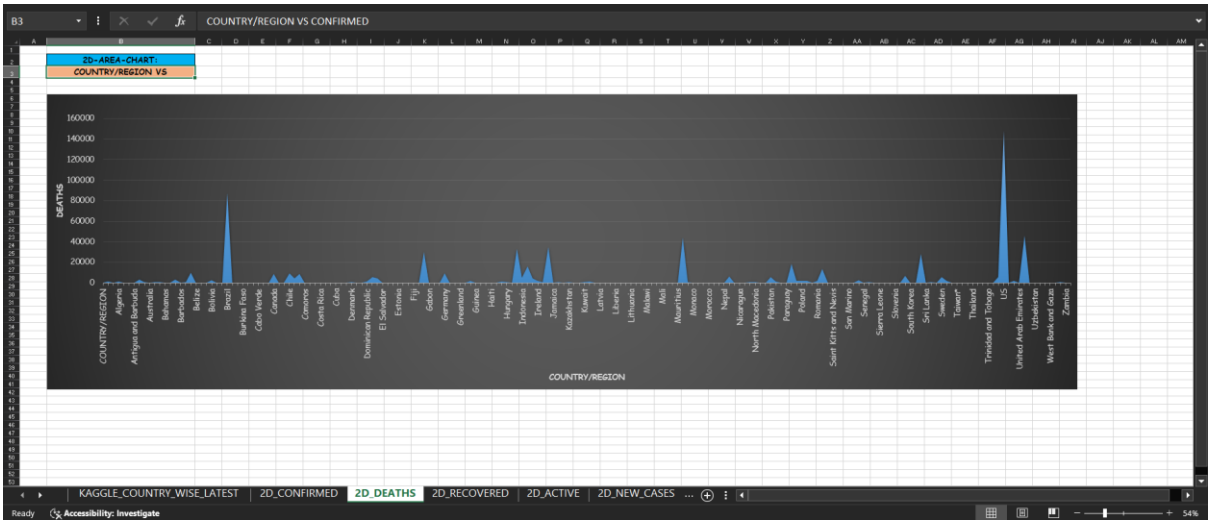
❖ ***Sheet-2: 2D_CONFIRMED***

This sheet contains a 2D Area Chart that represents the relationship between Country/Region and the Confirmed COVID-19 Cases. It helps visualize the distribution of confirmed cases across various regions globally.



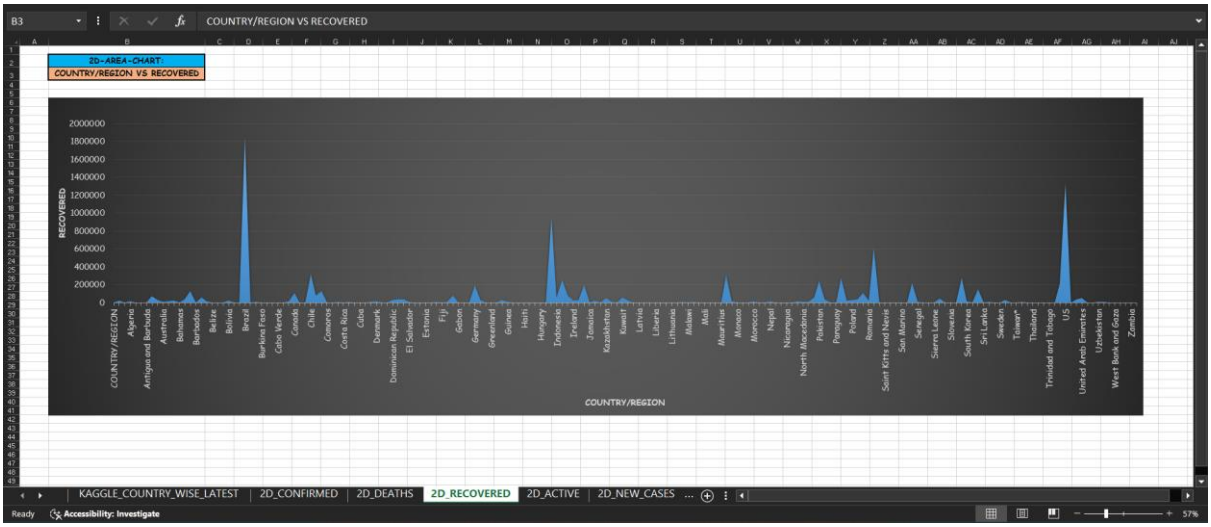
❖ ***Sheet-3: 2D_DEATHS***

The 2D Area Chart in this sheet compares the Country/Region with the Total Deaths due to COVID-19. It highlights the mortality rate in different countries, showcasing regions with the highest death tolls from the pandemic.



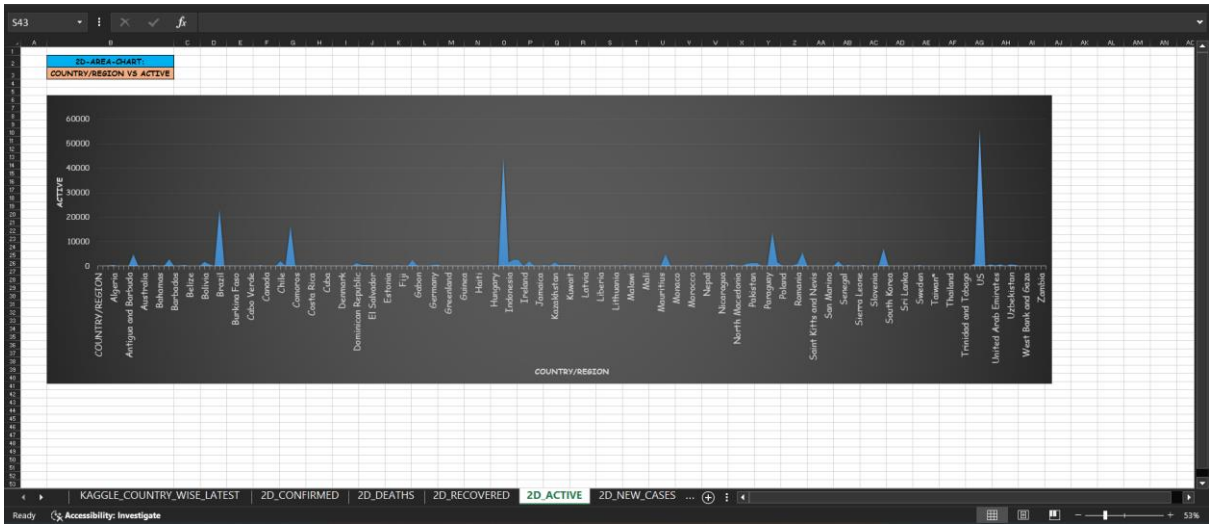
❖ ***Sheet-4: 2D_RECOVERED***

This sheet displays a 2D Area Chart that illustrates the Country/Region versus the Total Recovered COVID-19 Cases. The chart provides a clear view of the recovery trends across the globe, highlighting countries that have shown significant recovery rates.



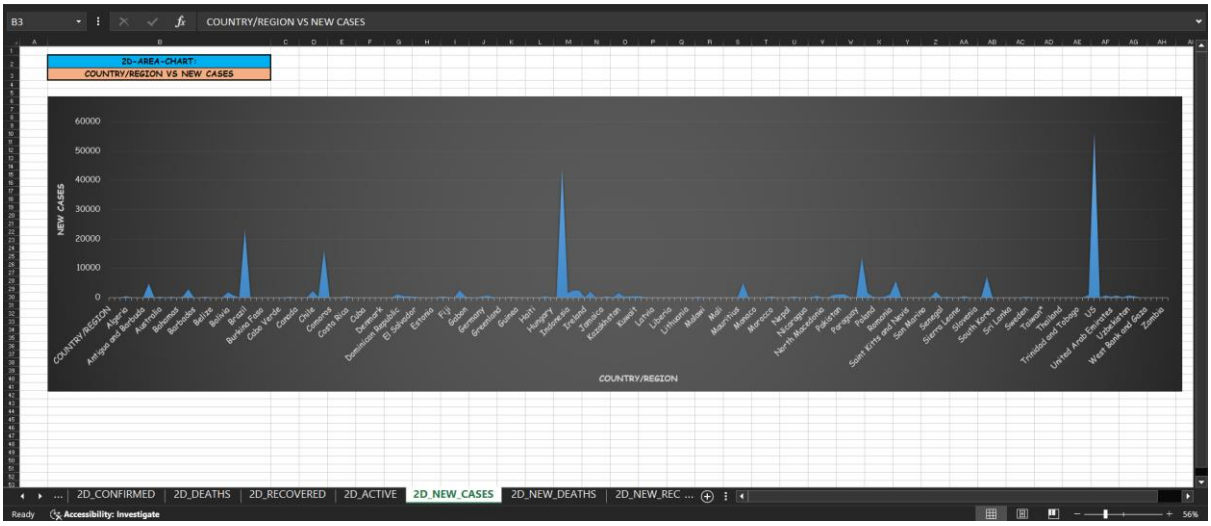
❖ ***Sheet-5: 2D_ACTIVE***

The 2D Area Chart in this sheet tracks the Country/Region in relation to Active COVID-19 Cases. This visualization is crucial for understanding the current load of active infections in different countries.



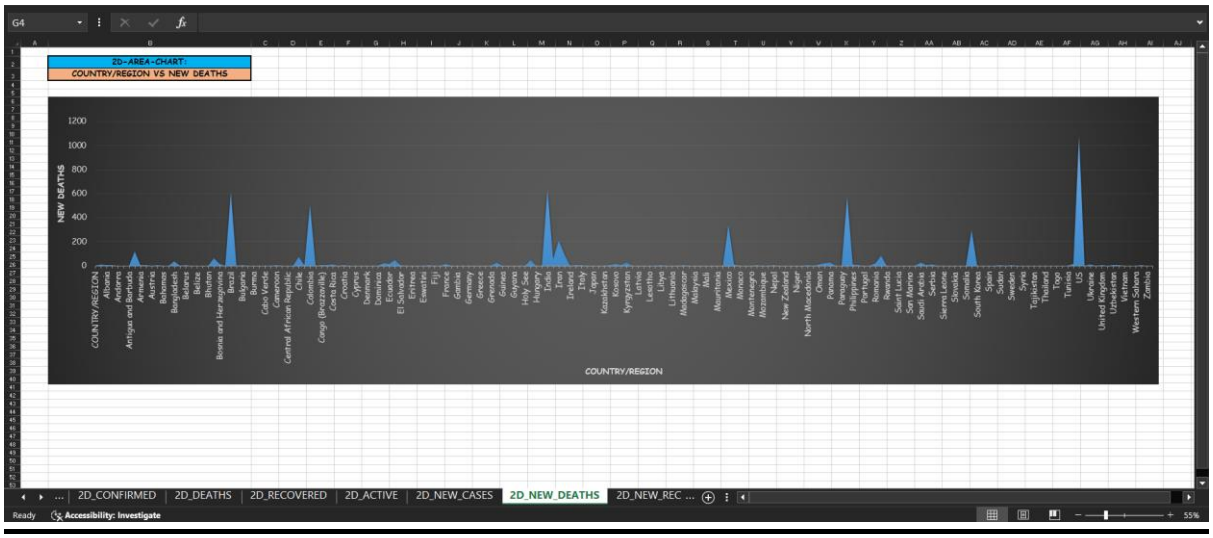
❖ ***Sheet-6: 2D_NEW_CASES***

In this sheet, a 2D Area Chart represents the Country/Region against the New COVID-19 Cases. This chart shows the growth of new infections and helps identify trends in the rate of new cases across countries.



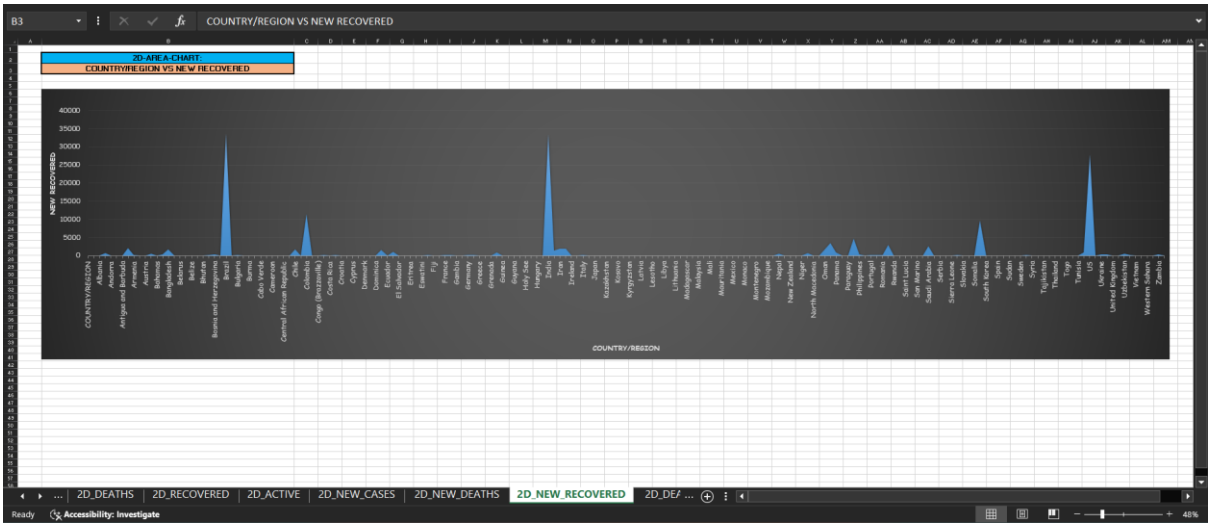
❖ ***Sheet-7: 2D_NEW_DEATHS***

The 2D Area Chart here visualizes the Country/Region in relation to New Deaths due to COVID-19. This sheet provides a focused look at the daily/periodic rise in deaths across the regions.



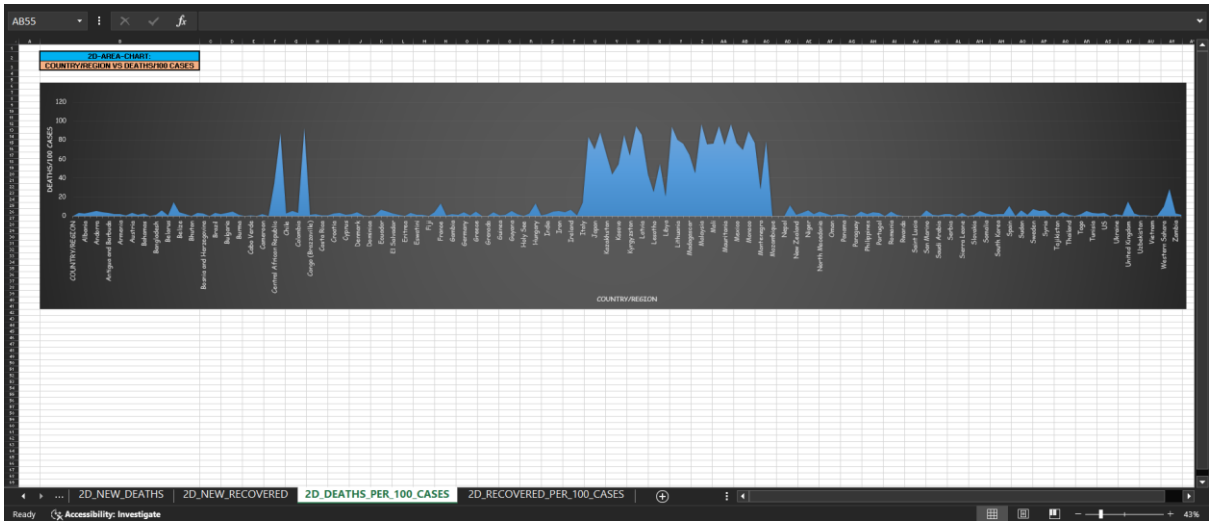
❖ ***Sheet-8: 2D_NEW_RECOVERED***

This sheet presents a 2D Area Chart comparing Country/Region with the New Recoveries after COVID-19 Attack. The chart highlights the pace of recovery in different regions as new cases decrease.



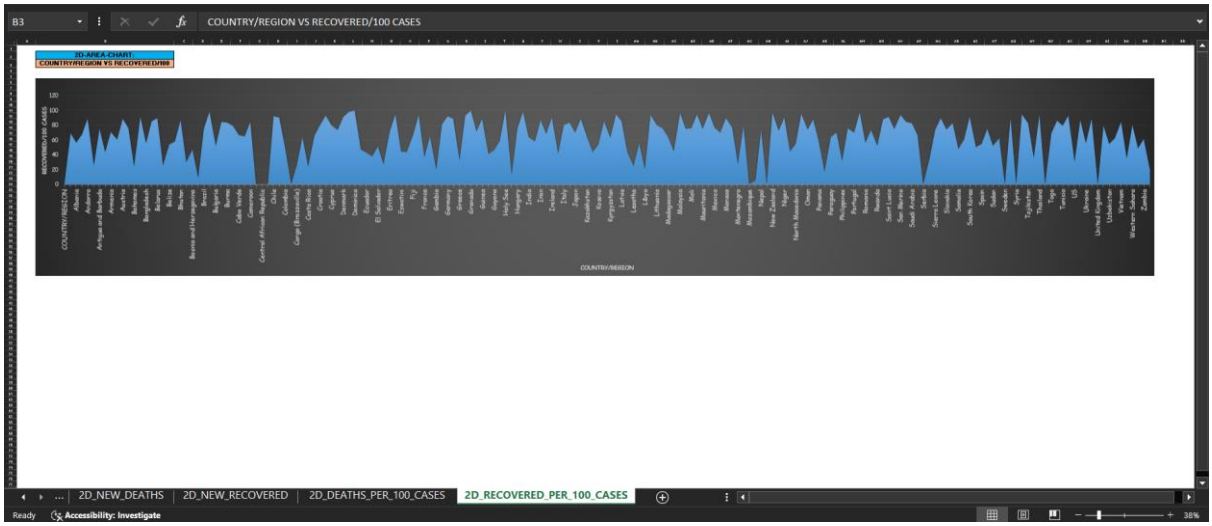
❖ ***Sheet-9: 2D_DEATHS_PER_100_CASES***

The 2D Area Chart in this sheet compared Country/Region with Deaths per 100 COVID-19 Cases. This metric is essential for understanding the mortality rate in proportion to the number of cases in different regions.



❖ ***Sheet-10:*** *2D_RECOVERED_PER_100_CASES*

In this sheet, a 2D Area Chart shows Country/Region versus Recoveries per 100 COVID-19 Cases. It helps analyze the recovery rate as a percentage of confirmed cases, offering insight into how well countries are managing to recover patients.



6. UNLOCKING TRENDS: GAINING INSIGHTS & FORECASTING THE FUTURE OF COVID-19

In this section, we take a closer look at the valuable insights derived from the data, identify emerging trends, and explore potential future predictions regarding the course of the pandemic across various countries. Understanding these patterns can help policymakers, health experts, and the public in better navigating the challenges posed by COVID-19.

- ***Key Insights from the Data:***

  Our analysis has highlighted several critical aspects of the ongoing COVID-19 crisis:

  - ***Total Confirmed Cases:*** We've observed a steady rise in the number of confirmed COVID-19 cases globally, with certain countries showing significant surges. This increase underscores the importance of ongoing testing, tracking, and immediate containment efforts to manage the virus spread.
  - ***Total Deaths and Recovery Rates:*** While recovery rates show a positive trend, the global death rate remains concerning. Countries with higher recovery rates have implemented effective health protocols, and we can see an encouraging improvement in recovery outcomes as the pandemic has progressed.
  - ***Regional Variations:*** There is a notable disparity in the number of cases, deaths, and recoveries across regions, with some countries managing the crisis more

effectively than others. These variations emphasize the importance of regional health policies and the need for international cooperation in vaccine distribution and treatment protocols.

❖ ***Emerging Trends:***

The data reveals several emerging trends that provide a glimpse into the future trajectory of the pandemic.

- ***Vaccination Impact:*** As vaccines are distributed globally, the trend shows a positive impact on case reduction and mortality rates. Countries with higher vaccination rates are seeing decline in active cases and a reduction in the pressure on healthcare systems.
- ***Sensational Variation:*** COVID-19's impact appears to follow certain seasonal patterns, with surges seen during colder months in specific regions. This seasonal variation could inform future preparedness strategies, especially regarding the availability of healthcare resources during high-demand periods.
- ***Shifting New Cases & Recoveries:*** The rates of new cases and recoveries highlight an overall improvement in healthcare management, although challenges remain in certain areas. The consistency of recovery numbers may point toward an increase in the global population's immunity and improved treatment protocols.

❖ ***Future Predictions:***

Based on current trends, the following predictions can be made for the future of the COVID-19 pandemic:

- ***Continued Global Vaccination:*** With the global vaccination drive expanding, it is expected that the rate of infection will continue to decline. This will lead to a decrease in the severity of the disease and a stabilization of case numbers globally.
- ***Localized Waves of Infection:*** Despite the vaccination efforts, localized outbreaks may continue to occur, especially in areas with low vaccination rates or where variants of concern arise. Governments will need to be prepared for rapid response mechanisms to address these surges.
- ***Long-Term Management:*** COVID-19 is expected to become an endemic virus in many regions, much like seasonal flu, with periodic flare-ups and the need for ongoing monitoring and booster vaccines. Societal adaption to these regular health measures will be key in minimizing the pandemic's long-term impact.

❖ ***Conclusion:***

While the pandemic has undoubtedly reshaped the world, the data reveals positive trends in recovery, improved healthcare management, and vaccination progress. As we move forward, the lessons learned from COVID-19 will be invaluable in shaping the future of global health. By continuing to monitor, predict, and adapt, we can work together to ensure a safer, healthier future for all.

7. CONCLUSION: A PATH TOWARDS RECOVERY AND PREPAREDNESS

In conclusion, the COVID-19 pandemic has presented unprecedented challenges across the globe, affecting every aspect of society. Through the data analyzed in this report, we've witnessed the remarkable resilience and adaptability of nations in managing the crisis.

The insights gained from the data showcase not only the human cost of the pandemic but also the strength of global efforts, including vaccination campaigns and healthcare advancements. While some countries have experienced higher case rates and mortality, others have demonstrated successful management strategies, with improved recovery outcome and decreasing infection rates.

Moving forward, we must continue to build on these successes and address the challenged that remain. Vaccination efforts, along with continuous monitoring and healthcare preparedness, will be key to navigating future waves of the pandemic. By focusing on international cooperation, equitable vaccine distribution, and public health measures, the world can prepare for the long-term management of COVID-19 reducing its impact and ensuring the well-being of all.

8. FUTURE DIRECTIONS

The pandemic has reshaped the global healthcare landscape, but this also offers an opportunity to strengthen systems for the future. Here are some critical areas for focus:

- ***Global Health Cooperation:*** The pandemic highlighted the importance of a unified global response to health crises. Strengthening international partnerships and information sharing will be crucial in responding to future pandemics.
- ***Enhanced Data Analytics:*** The continued use of data-drive approaches will enable more accurate predictions, better resource allocation, and quicker response times to emerging health threats.
- ***Sustainability in Healthcare Infrastructure:*** To better handle future health emergencies, there needs to be an investment in scalable healthcare infrastructure that can adapt to high demand, ensuring that no region is left behind during global health crisis.
- ***Long-Term Impact on Public Health:*** COVID-19 has highlighted the importance of mental health and long-term recovery. Future healthcare strategies should also include addressing these facets, offering holistic care for those affected by the pandemic, both physically and mentally.

9. RECOMMENDATIONS

As we reflect on the data presented throughout this report, the following recommendations will help mitigate the impact of COVID-19 and similar future pandemics.

- ***Increase Investment in Healthcare Systems:*** Governments must allocate more resources towards strengthening healthcare systems, especially in underdeveloped regions, to prepare for future health crises.
- ***Foster Public Awareness and Education:*** Public health campaigns should focus on increasing awareness about the importance of vaccination, hygiene, and safety measures to curb future infections.
- ***Expand Global Research Collaboration:*** Governments and institutions should continue to support and fund global research on vaccines, treatments, and diagnostics to ensure faster responses during future pandemics.

10. FINAL REMARKS

While the fight against COVID-19 is far from over, the data and trends presented in this report offer a hopeful outlook. With continued global cooperation, advancements in healthcare, and proactive public health strategies, we can turn the tide against the pandemic and emerge stronger, more prepared, and united.

11. BIBLIOGRAPHY/REFERENCES

The data utilized for this Python-based project has been sourced from the following reference:

- ***Kaggle Dataset:*** Imdevskp's COVID-19 Report: Available at: https://www.kaggle.com/datasets/imdevskp/corona-virus-report?resource=download

For the purpose of this analysis, data from one of the CSV files provided through the aforementioned link has been employed. This resource has been pivotal in gathering and analyzing the required COVID-19 statistics to generate the insights presented in this project.

12. ENDLESS GRATITUDE: A TRIBUTE TO MY PILLARS

Behind every achievement lies the support and sacrifices of those who believe in us. I dedicate this project, ***"2020 COVID-19 GLOBAL IMPACT VISUALIZER"***, to the two most invaluable people in my life – my parents.

Their unwavering encouragement, boundless patience, and unconditional love have been the foundation of my journey in ***Computer Science and Data Science.*** From instilling in me the passion for learning to guiding me through challenges, they have always been my greatest source of strength.

This project reflects not only my efforts but also the dedication and values they have instilled in me. Their constant motivation pushed me to explore, innovate, and persevere through complexities.

No words can truly express my gratitude for their sacrifices, wisdom, and belief in my dreams. This achievement is as much theirs as it is mine.

With all my heart,

Pramodh Narain