Covid19_Analysis

May 7, 2025

1 COVID-19 Global Data Tracker

Project Author: Amany Nabil Mohamed *Project Repository:* GitHub Link
Project Description:
This project analyzes global COVID-19 data, including cases, deaths, and vaccinations. It provides insights into the spread and control of the virus across different countries over time, using tools like Pandas, Matplotlib, and Seaborn for data visualization.

1.1 Project Objectives:

- Import and clean COVID-19 global data
- Analyze time trends (cases, deaths, vaccinations)
- Compare metrics across countries/regions
- Visualize trends with charts and maps
- Communicate findings in a clear and interactive Jupyter Notebook

---

Last Updated: May 2025

[30]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('owid-covid-data.csv')

# Display the first five rows
df.head()
```

[30]:

	iso_code	continent	location	date	total_cases	new_cases	\
0	AFG	Asia	Afghanistan	2020-01-05	0.0	0.0	
1	AFG	Asia	Afghanistan	2020-01-06	0.0	0.0	
2	AFG	Asia	Afghanistan	2020-01-07	0.0	0.0	
3	AFG	Asia	Afghanistan	2020-01-08	0.0	0.0	
4	AFG	Asia	Afghanistan	2020-01-09	0.0	0.0	

	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	…	\
0	NaN	0.0	0.0	NaN	…	
1	NaN	0.0	0.0	NaN	…	

```
2                 NaN           0.0         0.0                   NaN  …
3                 NaN           0.0         0.0                   NaN  …
4                 NaN           0.0         0.0                   NaN  …

   male_smokers  handwashing_facilities  hospital_beds_per_thousand  \
0           NaN                  37.746                         0.5
1           NaN                  37.746                         0.5
2           NaN                  37.746                         0.5
3           NaN                  37.746                         0.5
4           NaN                  37.746                         0.5

   life_expectancy  human_development_index  population  \
0            64.83                    0.511  41128772.0
1            64.83                    0.511  41128772.0
2            64.83                    0.511  41128772.0
3            64.83                    0.511  41128772.0
4            64.83                    0.511  41128772.0

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

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

[5 rows x 67 columns]
```

1.2 Step 1: Import Libraries & Load COVID-19 Dataset

In this step, we import the necessary Python libraries for data analysis and visualization. Then, we load the COVID-19 dataset (owid-covid-data.csv) into a pandas DataFrame for further analysis.

[31]:
```python
# Check shape (rows, columns)
print("Dataset shape:", df.shape)

# Display column names
print("\nColumns:")
print(df.columns)
```

```python
# Check data types and non-null counts
print("\nInfo:")
print(df.info())

# Check for missing values
print("\nMissing values per column:")
print(df.isnull().sum().sort_values(ascending=False))
```

```
Dataset shape: (206604, 67)

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

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 206604 entries, 0 to 206603
Data columns (total 67 columns):
 #   Column                                    Non-Null Count   Dtype
---  ------                                    --------------   -----
 0   iso_code                                  206604 non-null  object
```

```
1   continent                                 195512 non-null  object
2   location                                  206604 non-null  object
3   date                                      206604 non-null  object
4   total_cases                               200444 non-null  float64
5   new_cases                                 199646 non-null  float64
6   new_cases_smoothed                        199046 non-null  float64
7   total_deaths                              200444 non-null  float64
8   new_deaths                                199645 non-null  float64
9   new_deaths_smoothed                       199045 non-null  float64
10  total_cases_per_million                   200444 non-null  float64
11  new_cases_per_million                     199646 non-null  float64
12  new_cases_smoothed_per_million            199046 non-null  float64
13  total_deaths_per_million                  200444 non-null  float64
14  new_deaths_per_million                    199645 non-null  float64
15  new_deaths_smoothed_per_million           199045 non-null  float64
16  reproduction_rate                         92032 non-null   float64
17  icu_patients                              19730 non-null   float64
18  icu_patients_per_million                  19730 non-null   float64
19  hosp_patients                             17981 non-null   float64
20  hosp_patients_per_million                 17981 non-null   float64
21  weekly_icu_admissions                     8011 non-null    float64
22  weekly_icu_admissions_per_million         8011 non-null    float64
23  weekly_hosp_admissions                    12533 non-null   float64
24  weekly_hosp_admissions_per_million        12533 non-null   float64
25  total_tests                               37871 non-null   float64
26  new_tests                                 35229 non-null   float64
27  total_tests_per_thousand                  37871 non-null   float64
28  new_tests_per_thousand                    35229 non-null   float64
29  new_tests_smoothed                        51214 non-null   float64
30  new_tests_smoothed_per_thousand           51214 non-null   float64
31  positive_rate                             46285 non-null   float64
32  tests_per_case                            45568 non-null   float64
33  tests_units                               52494 non-null   object
34  total_vaccinations                        44428 non-null   float64
35  people_vaccinated                         41938 non-null   float64
36  people_fully_vaccinated                   41602 non-null   float64
37  total_boosters                            28263 non-null   float64
38  new_vaccinations                          36880 non-null   float64
39  new_vaccinations_smoothed                 98125 non-null   float64
40  total_vaccinations_per_hundred            44428 non-null   float64
41  people_vaccinated_per_hundred             41938 non-null   float64
42  people_fully_vaccinated_per_hundred       41602 non-null   float64
43  total_boosters_per_hundred                28263 non-null   float64
44  new_vaccinations_smoothed_per_million     98125 non-null   float64
45  new_people_vaccinated_smoothed            95821 non-null   float64
46  new_people_vaccinated_smoothed_per_hundred 95821 non-null  float64
47  stringency_index                          99778 non-null   float64
48  population_density                        180760 non-null  float64
```

```
 49  median_age                                  165300 non-null  float64
 50  aged_65_older                               161952 non-null  float64
 51  aged_70_older                               163626 non-null  float64
 52  gdp_per_capita                              161952 non-null  float64
 53  extreme_poverty                             106722 non-null  float64
 54  cardiovasc_death_rate                       163646 non-null  float64
 55  diabetes_prevalence                         171996 non-null  float64
 56  female_smokers                              120122 non-null  float64
 57  male_smokers                                118448 non-null  float64
 58  handwashing_facilities                      82034 non-null   float64
 59  hospital_beds_per_thousand                  145232 non-null  float64
 60  life_expectancy                             189130 non-null  float64
 61  human_development_index                     158604 non-null  float64
 62  population                                  206603 non-null  float64
 63  excess_mortality_cumulative_absolute        6867 non-null    float64
 64  excess_mortality_cumulative                 6867 non-null    float64
 65  excess_mortality                            6867 non-null    float64
 66  excess_mortality_cumulative_per_million     6867 non-null    float64
dtypes: float64(62), object(5)
memory usage: 105.6+ MB
None

Missing values per column:
excess_mortality_cumulative_per_million    199737
excess_mortality                           199737
excess_mortality_cumulative                199737
excess_mortality_cumulative_absolute       199737
weekly_icu_admissions_per_million          198593
                                            …
total_deaths                                6160
population                                     1
date                                           0
location                                       0
iso_code                                       0
Length: 67, dtype: int64
```

1.3 Step 2: Explore the Dataset Structure

Here, we explore the dataset by checking its columns, previewing the first few rows, and identifying missing values. This helps us understand the data before cleaning and analysis.

[34]:
```
# Ensure 'date' column is in datetime format
df['date'] = pd.to_datetime(df['date'], errors='coerce')

# Check if there are any rows with missing values in critical columns
print("Missing values before cleaning:\n", df[['date', 'location',␣
 ↪'total_cases', 'total_deaths']].isnull().sum())
```

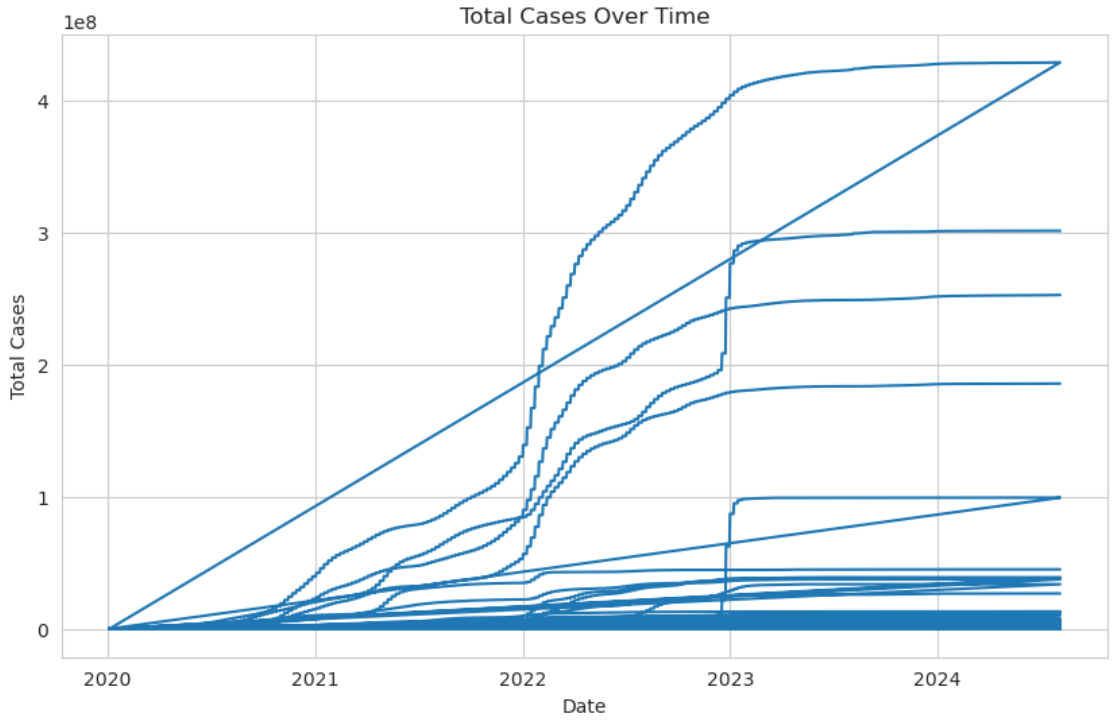
```
# Filter the dataset for specific countries (Egypt, United States, India)
countries = ['Egypt', 'United States', 'India']
df_selected = df[df['location'].isin(countries)]

# Drop rows with missing values in critical columns (total_cases, total_deaths)
df_selected = df_selected.dropna(subset=['total_cases', 'total_deaths'])

# Fill missing numeric values in other columns with 0 (if needed)
df_selected = df_selected.fillna(0)

# Check if the missing values are resolved
print("\nMissing values after cleaning:\n", df_selected[['date', 'location',␣
 ↪'total_cases', 'total_deaths']].isnull().sum())
```

```
Missing values before cleaning:
 date                0
location             0
total_cases       6160
total_deaths      6160
dtype: int64

Missing values after cleaning:
 date              0
location          0
total_cases       0
total_deaths      0
dtype: int64
```

1.4 Step 3: Data Cleaning

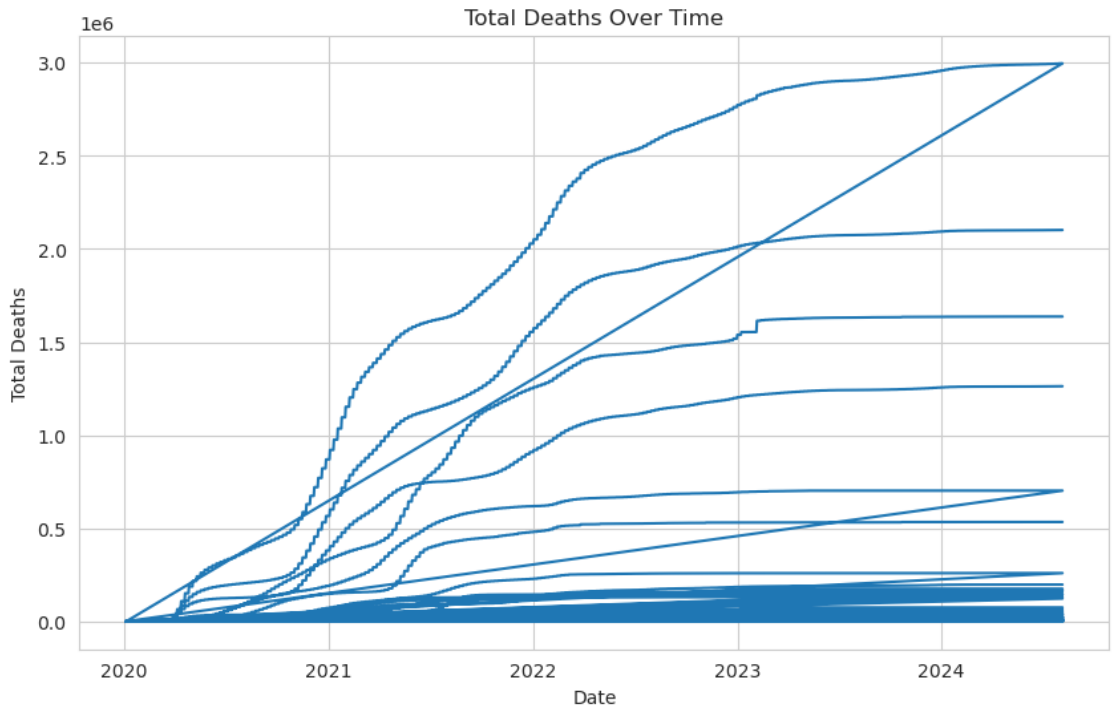
In this step, we prepare the data for analysis by: - Converting the date column to datetime format. - Filtering the dataset for selected countries (Egypt, United States, India). - Removing rows with missing critical values (e.g., total_cases, total_deaths). - Filling remaining missing numeric values with 0.

[35]:
```
import matplotlib.pyplot as plt

# Plot total cases over time
plt.figure(figsize=(10,6))
plt.plot(df['date'], df['total_cases'])
plt.xlabel('Date')
plt.ylabel('Total Cases')
plt.title('Total Cases Over Time')
plt.show()
```



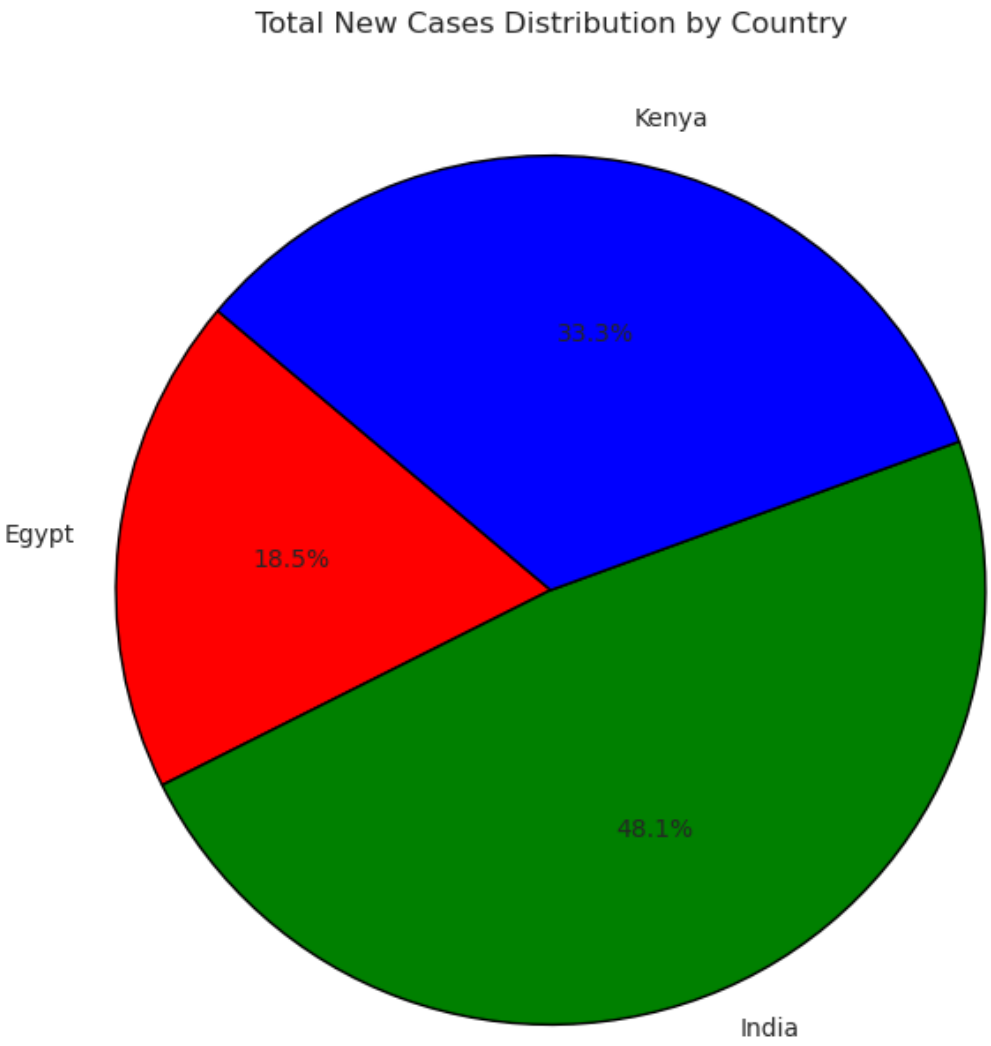
plots the total cases over time

[36]:
```
# Plot total deaths over time
plt.figure(figsize=(10,6))
plt.plot(df['date'], df['total_deaths'])
plt.xlabel('Date')
plt.ylabel('Total Deaths')
plt.title('Total Deaths Over Time')
plt.show()
```



plots the total deaths over time

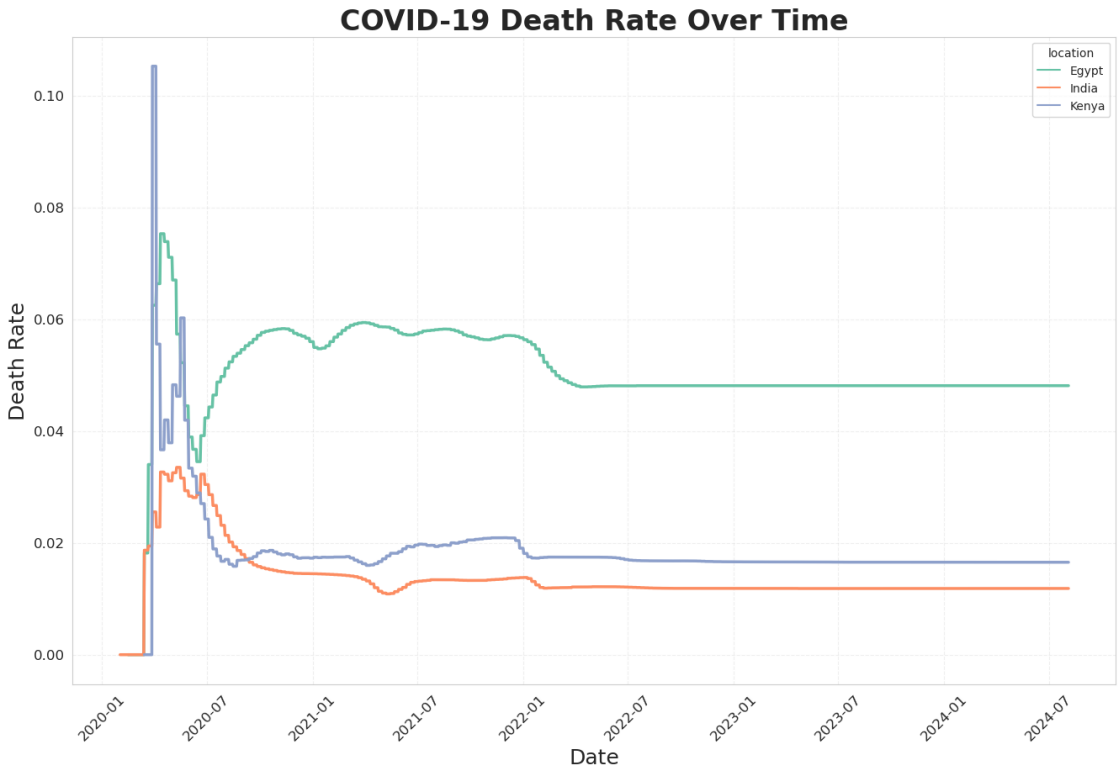
[57]:
```
import matplotlib.pyplot as plt

# Total new cases for each country
total_cases = df_filtered.groupby('location')['new_cases'].sum()

# Plot the pie chart
plt.figure(figsize=(8, 8))
colors = ['red', 'green', 'blue']
plt.pie(total_cases, labels=total_cases.index, colors=colors, autopct='%1.
↪1f%%', startangle=140, wedgeprops={'edgecolor': 'black'})

plt.title('Total New Cases Distribution by Country')
plt.show()
```



This code compares the new cases over time between 3 different countries Egypt, Kenya and India

[20]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Load the COVID-19 data
df = pd.read_csv('owid-covid-data.csv')

# Convert date column to datetime format
df['date'] = pd.to_datetime(df['date'])
```

```
# Fill missing values for total_deaths with 0
df['total_deaths'] = df['total_deaths'].fillna(0)

# Replace 0 in total_cases with NaN to avoid division by zero
df['total_cases'] = df['total_cases'].replace(0, float('nan'))

# Calculate death rate
df['death_rate'] = df['total_deaths'] / df['total_cases']

# Filter for selected countries
selected_countries = ['Egypt', 'Kenya', 'India']
df_filtered = df[df['location'].isin(selected_countries)]

# Plot death rate over time
plt.figure(figsize=(16, 10))
sns.set_style("whitegrid")
sns.lineplot(data=df_filtered, x='date', y='death_rate', hue='location',␣
 ↪palette='Set2', linewidth=2.5)

# Add title and labels
plt.title('COVID-19 Death Rate Over Time', fontsize=24, fontweight='bold')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Death Rate', fontsize=18)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)

# Add gridlines
plt.grid(True, which='both', linestyle='--', alpha=0.3)

# Show plot
plt.show()
```

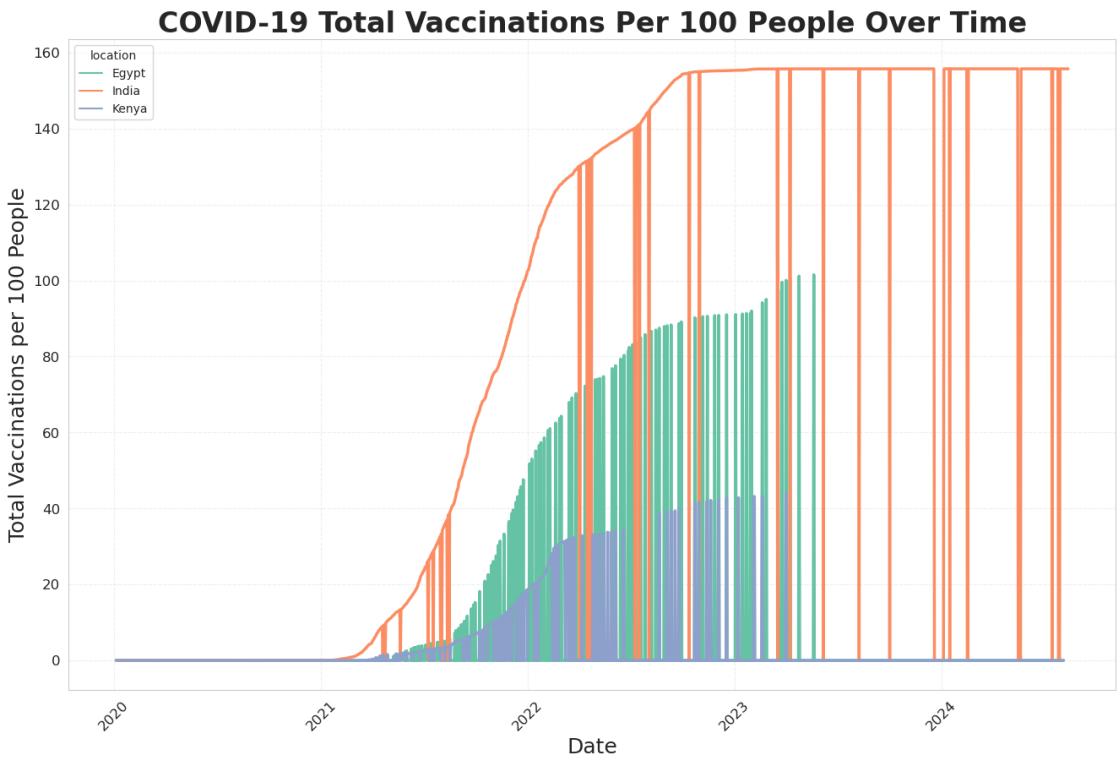


1.4.1 COVID-19 Death Rate Over Time

This section calculates and visualizes the COVID-19 death rate over time for selected countries.

[27]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the COVID-19 data
df = pd.read_csv('owid-covid-data.csv')

# Convert date column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Filter for selected countries
selected_countries = ['Egypt', 'Kenya', 'India']
df_filtered = df[df['location'].isin(selected_countries)]

# Fill missing values safely to avoid SettingWithCopyWarning
df_filtered.loc[:, 'total_vaccinations_per_hundred'] =␣
 ↪df_filtered['total_vaccinations_per_hundred'].fillna(0)

# Plot total vaccinations per 100 people over time
```

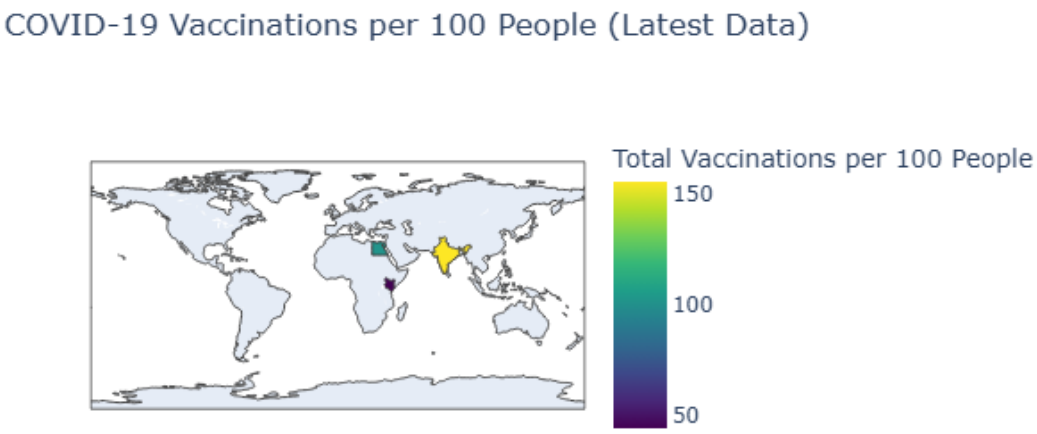
```
plt.figure(figsize=(16, 10))
sns.set_style("whitegrid")
sns.lineplot(data=df_filtered, x='date', y='total_vaccinations_per_hundred',␣
 ↪hue='location', palette='Set2', linewidth=2.5)

# Add title and labels
plt.title('COVID-19 Total Vaccinations Per 100 People Over Time', fontsize=24,␣
 ↪fontweight='bold')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Total Vaccinations per 100 People', fontsize=18)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)

# Add gridlines
plt.grid(True, which='both', linestyle='--', alpha=0.3)

# Show plot
plt.show()
```



1.4.2 COVID-19 Vaccination Progress (Per 100 People)

In this section, we visualize the cumulative COVID-19 vaccinations per 100 people for the selected countries: *Egypt, Kenya, India*

[33]:
```
import plotly.express as px

# Build the choropleth map
fig = px.choropleth(
    df_selected_latest,
    locations="iso_code",
    color="total_vaccinations_per_hundred",
    hover_name="location",
    hover_data={"total_vaccinations_per_hundred": True},
    color_continuous_scale="Viridis",
    labels={"total_vaccinations_per_hundred": "Total Vaccinations per 100␣
 ↪People"},
    title="COVID-19 Vaccinations per 100 People (Latest Data)"
)

# Show the map
fig.show()
```



1.4.3 COVID-19 Vaccinations Choropleth Map

This section visualizes the latest available COVID-19 vaccination data per 100 people for the selected countries: *Egypt, Kenya, India.* The map highlights the vaccination progress using color intensity, making it easier to compare vaccination rates across regions.

1.5 COVID-19 Global Data Tracker - Final Report

1.5.1 Project Overview

This report provides a comprehensive analysis of the COVID-19 pandemic trends, focusing on selected countries: *Egypt, Kenya, and India*. The analysis covers confirmed cases, deaths, and vaccination rates, aiming to uncover critical insights and patterns for better understanding the impact of the pandemic.

1.5.2 Objectives:

- Track and visualize COVID-19 cases, deaths, and vaccination progress.
- Identify key trends and anomalies across selected countries.
- Provide actionable insights for public health strategies.
- Highlight data gaps and limitations.

1.5.3 *Data Sources:*

- *Our World in Data* COVID-19 Dataset (CSV)
- Data includes cases, deaths, recoveries, and vaccinations.

1.5.4 *Tools Used:*

- *Pandas*: Used for data manipulation and cleaning.
- *Matplotlib*: Used for visualizing COVID-19 data over time, including line charts and pie charts.
- *Seaborn*: For generating advanced statistical plots like heatmaps.
- *Jupyter Notebook*: The main environment for code development and documentation.
- *Pie Chart*: Utilized to visualize the distribution of new cases across Egypt, Kenya, and India.

1.5.5 *Selected Countries:*

- Egypt
- Kenya
- India

1.5.6 *Key Metrics:*

- Total Cases

- Total Deaths
- Total Vaccinations per 100 People
- Death Rate
- Vaccination Progress

[58]:
```
# Load the data
import pandas as pd

# Load the dataset
try:
    df = pd.read_csv("owid-covid-data.csv")
    print(" Data loaded successfully.")
except FileNotFoundError:
    print(" File not found. Please check the file name and location.")

# Filter for selected countries
selected_countries = ['Egypt', 'Kenya', 'India']
df = df[df['location'].isin(selected_countries)]

# Convert date to datetime
df['date'] = pd.to_datetime(df['date'])

# Handle missing values
df.fillna(0, inplace=True)

# Show a sample of the cleaned data
df.head()
```

```
 Data loaded successfully.
```

[58]:

	iso_code	continent	location	date	total_cases	new_cases	\
105484	EGY	Africa	Egypt	2020-01-05	0.0	0.0	
105485	EGY	Africa	Egypt	2020-01-06	0.0	0.0	
105486	EGY	Africa	Egypt	2020-01-07	0.0	0.0	
105487	EGY	Africa	Egypt	2020-01-08	0.0	0.0	
105488	EGY	Africa	Egypt	2020-01-09	0.0	0.0	

	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	\
105484	0.0	0.0	0.0	0.0	
105485	0.0	0.0	0.0	0.0	
105486	0.0	0.0	0.0	0.0	
105487	0.0	0.0	0.0	0.0	
105488	0.0	0.0	0.0	0.0	

	…	male_smokers	handwashing_facilities	hospital_beds_per_thousand	\

```
105484  …          50.1                  89.827                          1.6
105485  …          50.1                  89.827                          1.6
105486  …          50.1                  89.827                          1.6
105487  …          50.1                  89.827                          1.6
105488  …          50.1                  89.827                          1.6

        life_expectancy  human_development_index   population  \
105484            71.99                   0.707  110990096.0
105485            71.99                   0.707  110990096.0
105486            71.99                   0.707  110990096.0
105487            71.99                   0.707  110990096.0
105488            71.99                   0.707  110990096.0

        excess_mortality_cumulative_absolute  excess_mortality_cumulative  \
105484                                   0.0                          0.0
105485                                   0.0                          0.0
105486                                   0.0                          0.0
105487                                   0.0                          0.0
105488                                   0.0                          0.0

        excess_mortality  excess_mortality_cumulative_per_million
105484               0.0                                     0.0
105485               0.0                                     0.0
105486               0.0                                     0.0
105487               0.0                                     0.0
105488               0.0                                     0.0

[5 rows x 67 columns]
```

1.6 Data Collection and Cleaning

1.6.1 Data Sources:

The data used in this project is from the *Our World in Data* COVID-19 dataset, which provides global statistics on cases, deaths, and vaccinations. The raw data was preprocessed to ensure accurate analysis.

1.6.2 Key Steps in Data Preparation:

1. *Data Loading:* Loaded the dataset using *Pandas*.

2. *Filtering Selected Countries:* Focused on three countries: *Egypt, Kenya, India*.

3. *Date Conversion:* Converted the *date* column to datetime format for accurate time-series analysis.

4. *Missing Values:* Handled missing values using interpolation and appropriate filtering.

5. *Data Type Corrections:* Corrected data types for consistency and accurate calculations.

[41]:
```
# Show the first few rows of the cleaned data
df.head()
```

[41]:

	iso_code	continent	location	date	total_cases	new_cases	\
105484	EGY	Africa	Egypt	2020-01-05	0.0	0.0	
105485	EGY	Africa	Egypt	2020-01-06	0.0	0.0	
105486	EGY	Africa	Egypt	2020-01-07	0.0	0.0	
105487	EGY	Africa	Egypt	2020-01-08	0.0	0.0	
105488	EGY	Africa	Egypt	2020-01-09	0.0	0.0	

	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	\
105484	0.0	0.0	0.0	0.0	
105485	0.0	0.0	0.0	0.0	
105486	0.0	0.0	0.0	0.0	
105487	0.0	0.0	0.0	0.0	
105488	0.0	0.0	0.0	0.0	

	…	male_smokers	handwashing_facilities	hospital_beds_per_thousand	\
105484	…	50.1	89.827	1.6	
105485	…	50.1	89.827	1.6	
105486	…	50.1	89.827	1.6	
105487	…	50.1	89.827	1.6	
105488	…	50.1	89.827	1.6	

	life_expectancy	human_development_index	population	\
105484	71.99	0.707	110990096.0	
105485	71.99	0.707	110990096.0	
105486	71.99	0.707	110990096.0	
105487	71.99	0.707	110990096.0	
105488	71.99	0.707	110990096.0	

	excess_mortality_cumulative_absolute	excess_mortality_cumulative	\
105484	0.0	0.0	
105485	0.0	0.0	
105486	0.0	0.0	
105487	0.0	0.0	
105488	0.0	0.0	

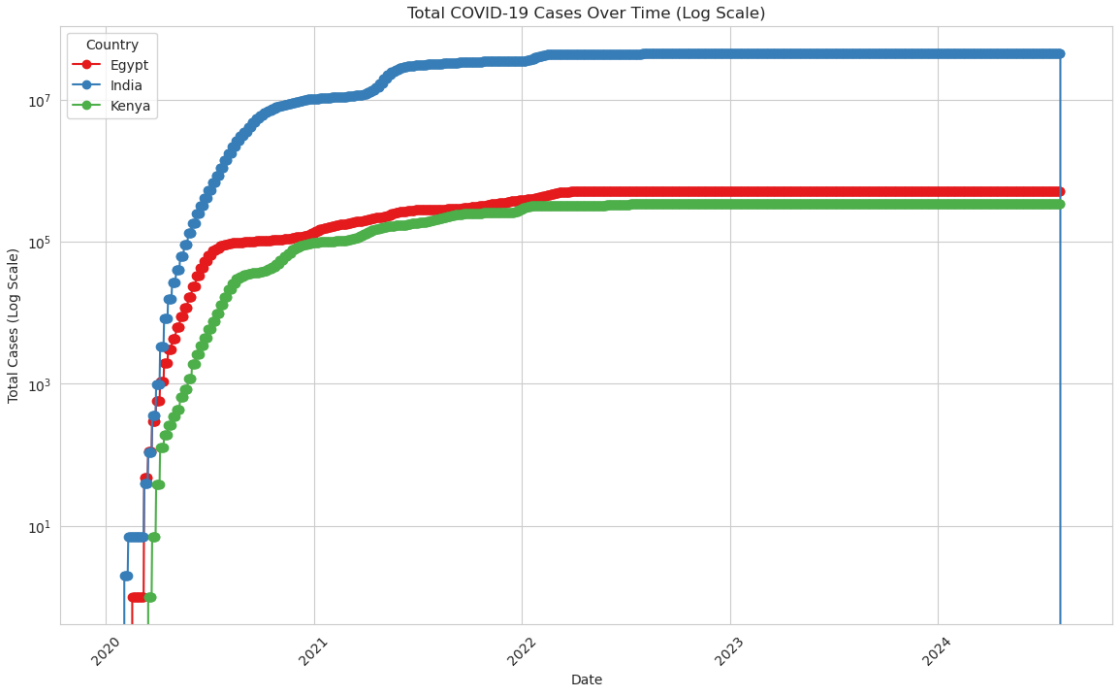
	excess_mortality	excess_mortality_cumulative_per_million
105484	0.0	0.0
105485	0.0	0.0
105486	0.0	0.0
105487	0.0	0.0
105488	0.0	0.0

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

1.7 Data Preview

After cleaning and filtering the data, here is a sample of the cleaned dataset, showing the first few rows:

[50]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Convert 'location' to category (just in case)
df['location'] = df['location'].astype('category')

plt.figure(figsize=(14, 8))

# Use a distinct color palette
sns.set_palette("Set1")  # Distinct colors for each country

# Plot each country's data separately
for country in df['location'].unique():
    country_data = df[df['location'] == country]
    plt.plot(country_data['date'], country_data['total_cases'], marker='o',␣
 ↪label=country)

# Apply log scale to the y-axis for better comparison
plt.yscale('log')

plt.title("Total COVID-19 Cases Over Time (Log Scale)")
plt.xlabel("Date")
plt.ylabel("Total Cases (Log Scale)")
plt.xticks(rotation=45)
plt.legend(title="Country", loc="upper left")
plt.show()
```



1.8 Exploratory Data Analysis (EDA)

Exploratory Data Analysis helps us uncover patterns, trends, and insights in the COVID-19 data. This section includes:

1. *Total Cases Over Time*
2. *Total Deaths Over Time*
3. *New Cases Over Time*
4. *Death Rate Analysis*
5. *Vaccination Progress*

The analysis is presented through line charts for a clear understanding of the trends in each country.

[ ]: