EDA on COVID-19 Clinical Trials

1. Introduction

The COVID-19 pandemic sparked a global surge in clinical research, with thousands of trials launched to study prevention, treatment, and diagnostic strategies. This dataset from ClinicalTrials.gov provides metadata on 5,783 COVID-related trials.

This analysis seeks to:

- Explore demographic and status distributions
- Examine study phases and funding sources
- Visualize country-wise trial distribution
- Perform correlation analysis between categorical and numerical variables

2. Dataset Overview

Column Name	Description
NCT Number	Unique identifier for each clinical trial
Status	Current status of the trial (e.g., Completed, Recruiting, Not yet recruiting)
Conditions	Disease or condition being studied (e.g., COVID-19)
Interventions	Drugs, procedures, or treatments being tested
Study Type	Type of study (e.g., Interventional, Observational)
Phases	Trial phase (I–IV or Not Applicable)
Enrollment	Number of participants in the study
Gender	Genders eligible to participate (Male, Female, All)
Age	Age range or minimum/maximum age of participants
Locations	Geographic location(s) of the trial
Funded Bys	Who funded the study (NIH, Industry)
Study Results	Whether study results have been published or not
Start Date	When the trial started
Completion Date	When the trial ended
Study Documents	Links to protocols, results

3. Code, Explanation

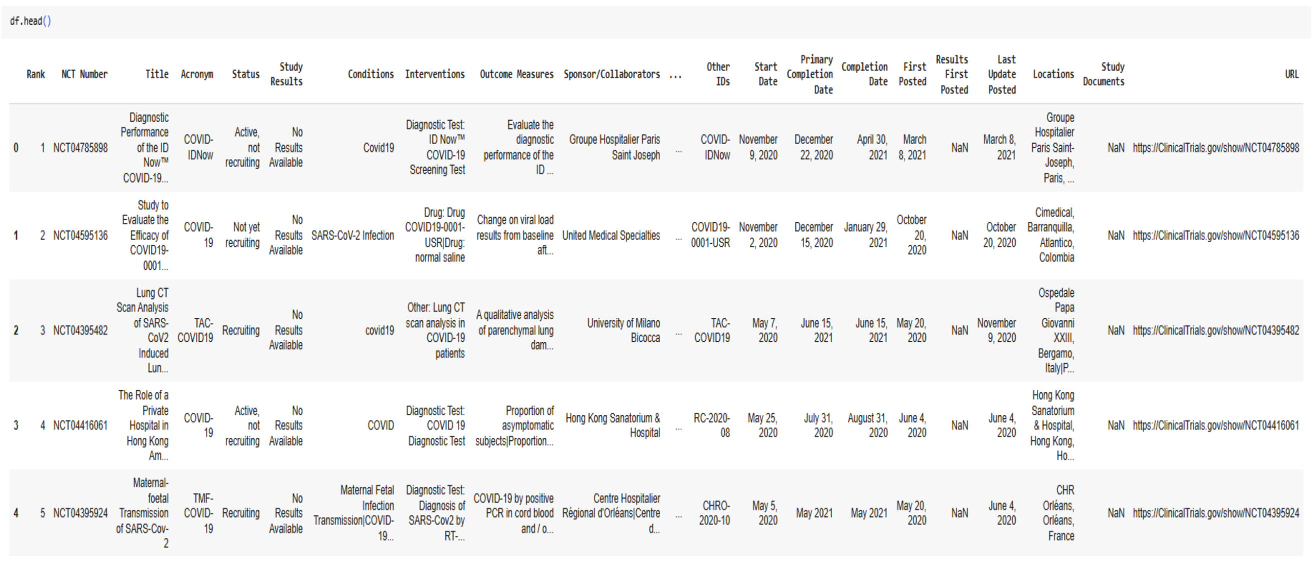
3.1 Import Visualization Libraries

```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
import seaborn as sns
from plotly.subplots import make_subplots
import plotly.graph_objects as go
```

- **Pandas**: to load and manipulate structured data (e.g., Excel, CSV) in dataframes.
- **NumPy:** is used for fast mathematical operations and handling numeric arrays efficiently.
- **Matplotlib:** is the foundational library for plotting in Python.
- **Seaborn:** builds on Matplotlib to create more visually appealing statistical plots (e.g., boxplots, heatmaps, distribution plots).
- **%matplotlib inline:** ensures that plots show up within a Jupyter Notebook cell output, which is helpful for interactive data exploration.
- **Plotly.subplots.make_subplots** and **plotly.graph_objects (go):** bring in Plotly, a library for creating interactive, web-based plots**.**

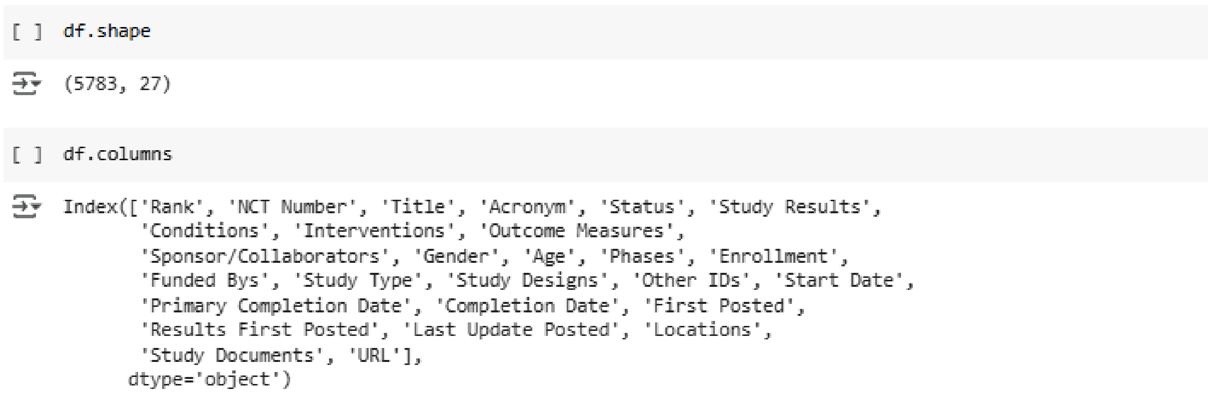
3.2 Load Dataset

```python
df = pd.read_csv('/content/COVID clinical trials.csv')
```



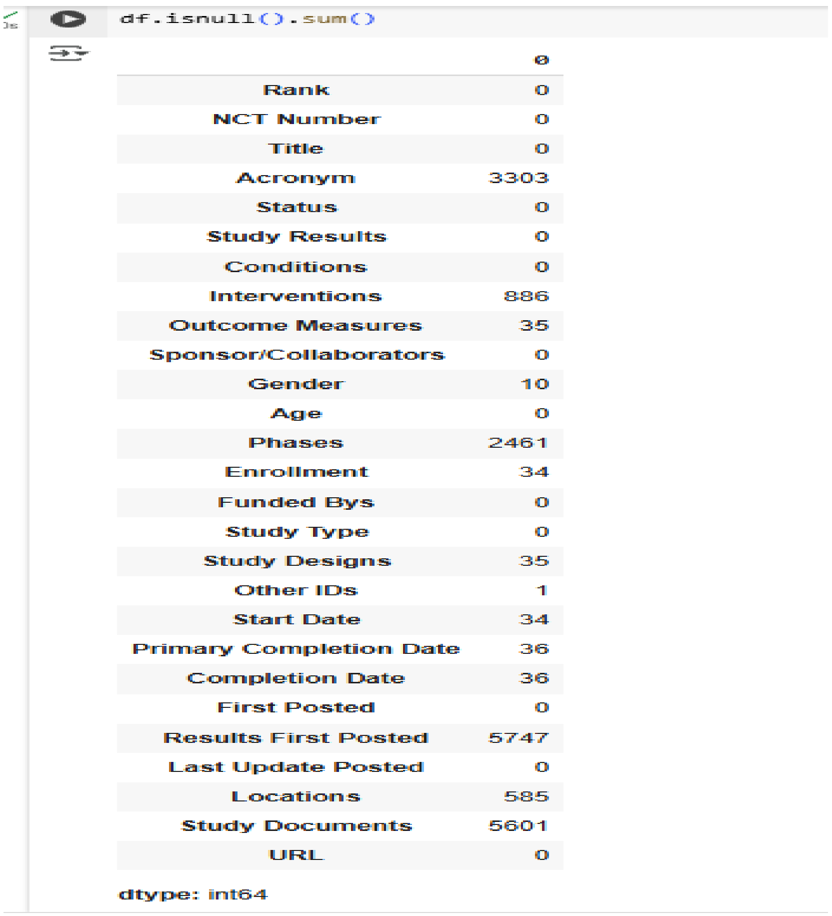
```python
df.head()
```

	Rank	NCT Number	Title	Acronym	Status	Study Results	Conditions	Interventions	Outcome Measures	Sponsor/Collaborators	...	Other IDs	Start Date	Primary Completion Date	Completion Date	First Posted	Results First Posted	Last Update Posted	Locations	Study Documents	URL
0	1	NCT04785898	Diagnostic Performance of the ID Now™ COVID-19...	COVID-IDNow	Active, not recruiting	No Results Available	Covid19	Diagnostic Test: ID Now™ COVID-19 Screening Test	Evaluate the diagnostic performance of the ID ...	Groupe Hospitalier Paris Saint Joseph	...	COVID-IDNow	November 9, 2020	December 22, 2020	April 30, 2021	March 8, 2021	NaN	March 8, 2021	Groupe Hospitalier Paris Saint-Joseph, Paris, ...	NaN	https://ClinicalTrials.gov/show/NCT04785898
1	2	NCT04595136	Study to Evaluate the Efficacy of COVID19-0001...	COVID-19	Not yet recruiting	No Results Available	SARS-CoV-2 Infection	Drug: Drug COVID19-0001-USR|Drug: normal saline	Change on viral load results from baseline aft...	United Medical Specialties	...	COVID19-0001-USR	November 2, 2020	December 15, 2020	January 29, 2021	October 20, 2020	NaN	October 20, 2020	Cimedical, Barranquilla, Atlantico, Colombia	NaN	https://ClinicalTrials.gov/show/NCT04595136
2	3	NCT04395482	Lung CT Scan Analysis of SARS-CoV2 Induced Lun...	TAC-COVID19	Recruiting	No Results Available	covid19	Other: Lung CT scan analysis in COVID-19 patients	A qualitative analysis of parenchymal lung dam...	University of Milano Bicocca	...	TAC-COVID19	May 7, 2020	June 15, 2021	June 15, 2021	May 20, 2020	NaN	November 9, 2020	Ospedale Papa Giovanni XXIII, Bergamo, Italy|P...	NaN	https://ClinicalTrials.gov/show/NCT04395482
3	4	NCT04416061	The Role of a Private Hospital in Hong Kong Am...	COVID-19	Active, not recruiting	No Results Available	COVID	Diagnostic Test: COVID 19 Diagnostic Test	Proportion of asymptomatic subjects|Proportion...	Hong Kong Sanatorium & Hospital	...	RC-2020-08	May 25, 2020	July 31, 2020	August 31, 2020	June 4, 2020	NaN	June 4, 2020	Hong Kong Sanatorium & Hospital, Hong Kong, Ho...	NaN	https://ClinicalTrials.gov/show/NCT04416061
4	5	NCT04395924	Maternal-foetal Transmission of SARS-Cov-2	TMF-COVID-19	Recruiting	No Results Available	Maternal Fetal Infection Transmission|COVID-19...	Diagnostic Test: Diagnosis of SARS-Cov2 by RT-...	COVID-19 by positive PCR in cord blood and / o...	Centre Hospitalier Régional d'Orléans|Centre d...	...	CHRO-2020-10	May 5, 2020	May 2021	May 2021	May 20, 2020	NaN	June 4, 2020	CHR Orléans, Orléans, France	NaN	https://ClinicalTrials.gov/show/NCT04395924



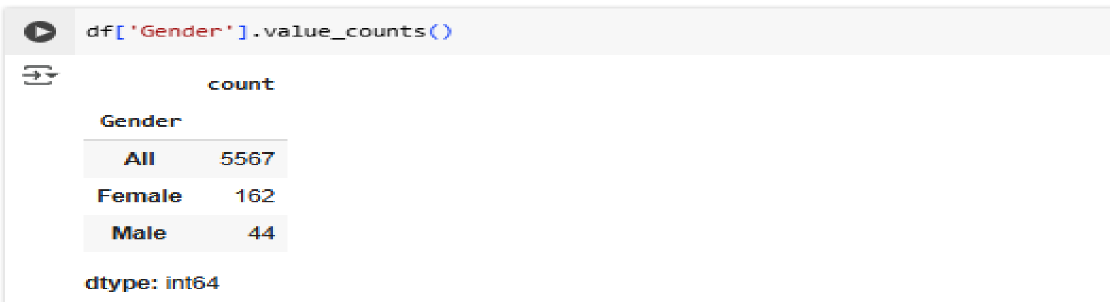
```
[ ] df.shape

(5783, 27)

[ ] df.columns

Index(['Rank', 'NCT Number', 'Title', 'Acronym', 'Status', 'Study Results',
       'Conditions', 'Interventions', 'Outcome Measures',
       'Sponsor/Collaborators', 'Gender', 'Age', 'Phases', 'Enrollment',
       'Funded Bys', 'Study Type', 'Study Designs', 'Other IDs', 'Start Date',
       'Primary Completion Date', 'Completion Date', 'First Posted',
       'Results First Posted', 'Last Update Posted', 'Locations',
       'Study Documents', 'URL'],
      dtype='object')
```

3.3 Explore the missing values



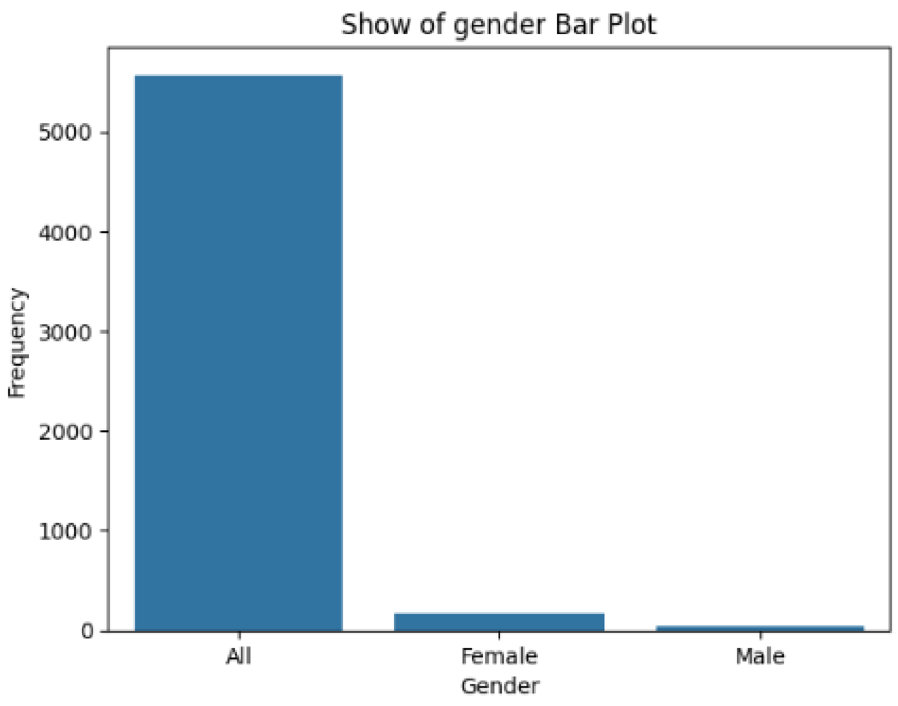
```
df.isnull().sum()

                                 0
                   Rank          0
              NCT Number         0
                  Title          0
                 Acronym      3303
                 Status          0
             Study Results       0
               Conditions        0
             Interventions     886
           Outcome Measures     35
        Sponsor/Collaborators    0
                 Gender         10
                   Age           0
                 Phases       2461
               Enrollment       34
               Funded Bys        0
               Study Type        0
              Study Designs     35
                Other IDs        1
               Start Date       34
       Primary Completion Date  36
             Completion Date    36
               First Posted      0
          Results First Posted 5747
           Last Update Posted    0
                Locations      585
             Study Documents  5601
                   URL           0

dtype: int64
```

3.4 Explore the Gender distribution in the studies



```
df['Gender'].value_counts()

              count
Gender
  All          5567
 Female         162
  Male           44

dtype: int64
```

```
sns.barplot(x=df['Gender'].value_counts().index,
              y=df['Gender'].value_counts().values)
plt.xlabel('Gender')
plt.ylabel('Frequency')
plt.title('Show of gender Bar Plot')
plt.show()
```



Insights:

- Shows how many trials are open to All, Male, or Female participants
- Most trials likely include all genders, which reflects inclusive design

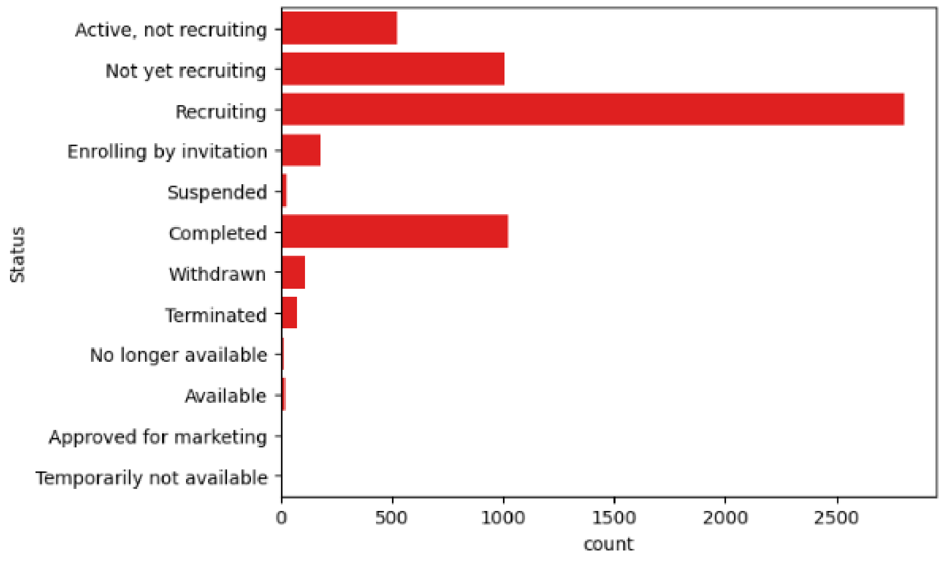
3.5 Exploring Study Status Distribution

```
df['Status'].value_counts()
```

```
                                count
              Status
          Recruiting             2805
          Completed              1025
      Not yet recruiting         1004
    Active, not recruiting        526
   Enrolling by invitation        181
          Withdrawn               107
          Terminated               74
          Suspended                27
          Available                19
     No longer available           12
   Approved for marketing           2
  Temporarily not available         1

dtype: int64
```

```
sns.countplot(y="Status", data=df, color="red")
```

```
<Axes: xlabel='count', ylabel='Status'>
```



Insights:

- Reveals whether studies are Recruiting, Completed, Not yet recruiting, etc.
- Recruiting is the most dominant status, with over 2,500 trials actively enrolling participants.
- Indicates ongoing research efforts in COVID-19 treatment or prevention

3.6 Cleaning Age Column

```
df.Age.unique()
```

```
array(['18 Years and older \xa0 (Adult, Older Adult)',
       'Child, Adult, Older Adult', '18 Years to 48 Years \xa0 (Adult)',
       '18 Years to 75 Years \xa0 (Adult, Older Adult)',
       '18 Years to 45 Years \xa0 (Adult)',
       '18 Years to 99 Years \xa0 (Adult, Older Adult)',
       '18 Years to 55 Years \xa0 (Adult)',
       '15 Years and older \xa0 (Child, Adult, Older Adult)',
       '18 Years to 80 Years \xa0 (Adult, Older Adult)',
       '45 Years and older \xa0 (Adult, Older Adult)',
       '20 Years to 100 Years \xa0 (Adult, Older Adult)',
       '8 Years to 88 Years \xa0 (Child, Adult, Older Adult)',
       '5 Years to 65 Years \xa0 (Child, Adult, Older Adult)',
       'up to 99 Years \xa0 (Child, Adult, Older Adult)',
       '18 Years to 85 Years \xa0 (Adult, Older Adult)',
       '18 Years to 65 Years \xa0 (Adult, Older Adult)',
       'up to 29 Days \xa0 (Child)',
       '18 Years to 70 Years \xa0 (Adult, Older Adult)',
       '18 Years to 59 Years \xa0 (Adult)',
       'up to 100 Years \xa0 (Child, Adult, Older Adult)',
       '20 Years to 60 Years \xa0 (Adult)',
       '40 Years to 80 Years \xa0 (Adult, Older Adult)',
       '23 Years and older \xa0 (Adult, Older Adult)',
       '18 Years to 120 Years \xa0 (Adult, Older Adult)',
       '16 Years and older \xa0 (Child, Adult, Older Adult)',
       '5 Years to 90 Years \xa0 (Child, Adult, Older Adult)',
       '18 Years to 90 Years \xa0 (Adult, Older Adult)',
       'up to 18 Years \xa0 (Child, Adult)',
       '2 Years and older \xa0 (Child, Adult, Older Adult)',
       '70 Years and older \xa0 (Older Adult)',
       '18 Years to 26 Years \xa0 (Adult)',
       '18 Years to 95 Years \xa0 (Adult, Older Adult)',
       '12 Years and older \xa0 (Child, Adult, Older Adult)',
       '16 Years to 55 Years \xa0 (Child, Adult)',
       '30 Years to 70 Years \xa0 (Adult, Older Adult)',
       '35 Years to 65 Years \xa0 (Adult, Older Adult)',
       '18 Years to 40 Years \xa0 (Adult)',
       '18 Years to 60 Years \xa0 (Adult)',
       '18 Years to 100 Years \xa0 (Adult, Older Adult)',
       '6 Years and older \xa0 (Child, Adult, Older Adult)',
       'up to 17 Years \xa0 (Child)',
       '22 Years to 72 Years \xa0 (Adult, Older Adult)',
       '16 Years to 100 Years \xa0 (Child, Adult, Older Adult)',
       '6 Months and older \xa0 (Child, Adult, Older Adult)',
       '20 Years to 65 Years \xa0 (Adult, Older Adult)',
       '14 Years to 75 Years \xa0 (Child, Adult, Older Adult)',
       '5 Years and older \xa0 (Child, Adult, Older Adult)',
       '1 Year to 100 Years \xa0 (Child, Adult, Older Adult)',
```

```
from string import digits

def remove_digits(text):
    return text.translate(str.maketrans('', '', digits))

df["Age"] = df["Age"].apply(lambda text: remove_digits(text))
df[['Age']].head()
```

Explain this code:

```
from string import digits
```

Imports the digits constant, which contains all digits ('0123456789').

```
def remove_digits(text):
    return text.translate(str.maketrans('', '', digits))
```

This function remove_digits takes a string input (text) and removes all digits from it.

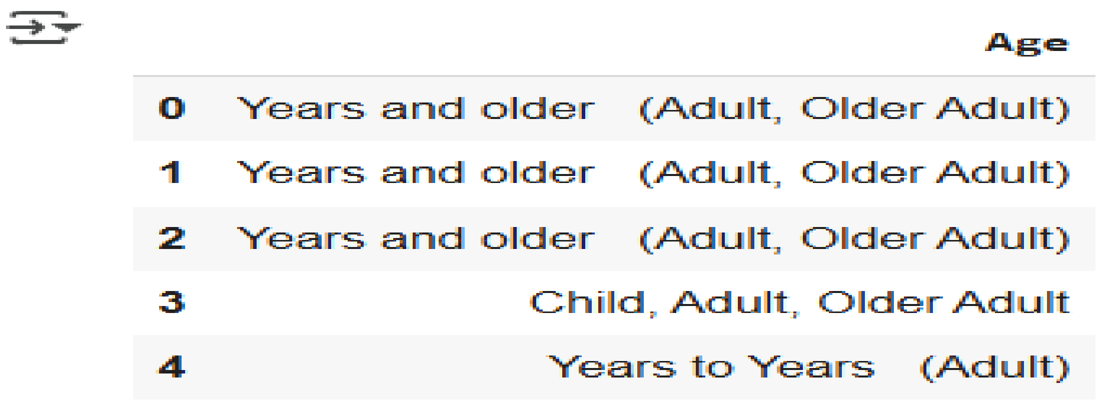
- **str.maketrans('', '', digits)** creates a translation table that deletes any characters found in digits (0–9).
- **text.translate(...)** uses that table to return the string without any numbers.

```
df["Age"] = df["Age"].apply(lambda text: remove_digits(text))
df[['Age']].head()
```

This line applies the remove_digits function to every value in the Age column.

- For example, if an age value is "18 Years", it becomes " Years".
- If it's "6 Months", it becomes " Months".

Output:

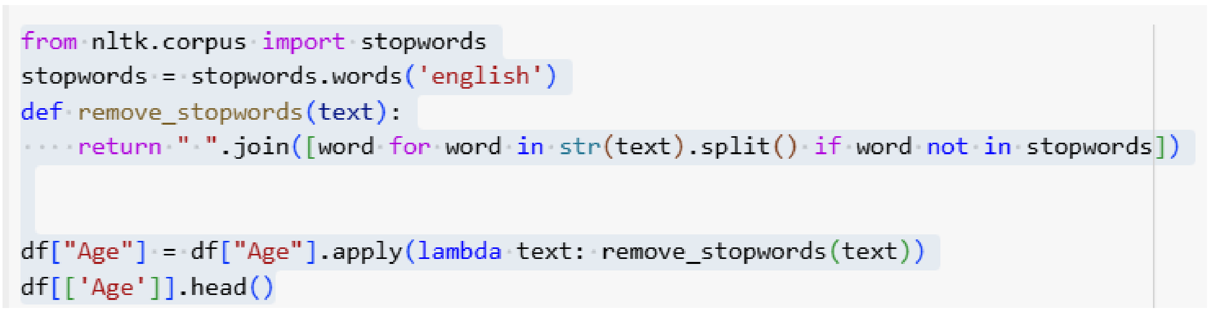


	Age
0	Years and older (Adult, Older Adult)
1	Years and older (Adult, Older Adult)
2	Years and older (Adult, Older Adult)
3	Child, Adult, Older Adult
4	Years to Years (Adult)

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

- **nltk (Natural Language Toolkit**) is a Python library used for working with human language data.
- **stopwords** are commonly used words in a language that are often removed during text preprocessing (like: "is", "the", "in", etc.).

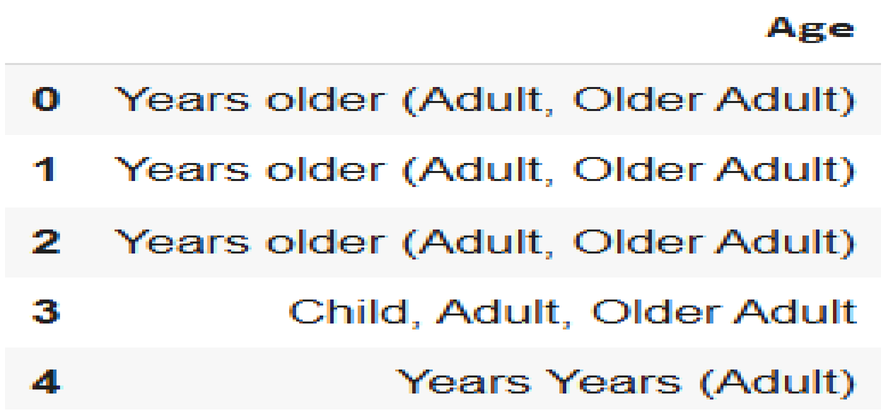


```python
from nltk.corpus import stopwords
stopwords = stopwords.words('english')
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in stopwords])

df["Age"] = df["Age"].apply(lambda text: remove_stopwords(text))
df[['Age']].head()
```

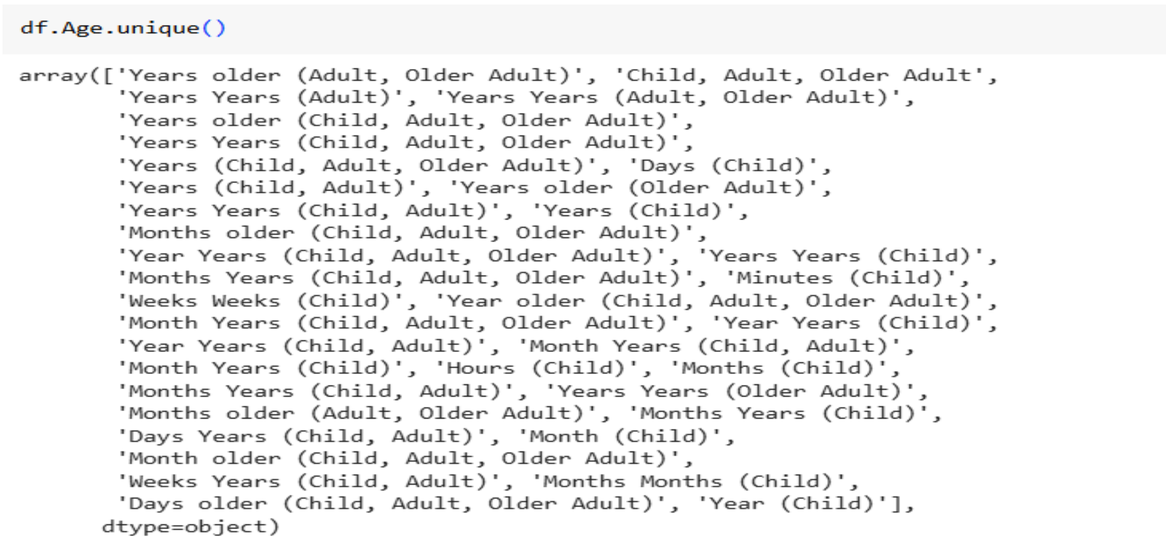
Explain this code:

- **from nltk.corpus import stopwords**: Imports common English stopwords (like "the", "is", "and") from NLTK.
- **remove_stopwords(text)**: Defines a function that removes these stopwords from a given text by keeping only the words not in the stopwords list.
- **df["Age"].apply(lambda text: remove_stopwords(text))**: Applies the remove_stopwords function to each entry in the "Age" column.

Output:



	Age
0	Years older (Adult, Older Adult)
1	Years older (Adult, Older Adult)
2	Years older (Adult, Older Adult)
3	Child, Adult, Older Adult
4	Years Years (Adult)



```
df.Age.unique()
```

```
array(['Years older (Adult, Older Adult)', 'Child, Adult, Older Adult',
       'Years Years (Adult)', 'Years Years (Adult, Older Adult)',
       'Years older (Child, Adult, Older Adult)',
       'Years Years (Child, Adult, Older Adult)',
       'Years (Child, Adult, Older Adult)', 'Days (Child)',
       'Years (Child, Adult)', 'Years older (Older Adult)',
       'Years Years (Child, Adult)', 'Years (Child)',
       'Months older (Child, Adult, Older Adult)',
       'Year Years (Child, Adult, Older Adult)', 'Years Years (Child)',
       'Months Years (Child, Adult, Older Adult)', 'Minutes (Child)',
       'Weeks Weeks (Child)', 'Year older (Child, Adult, Older Adult)',
       'Month Years (Child, Adult, Older Adult)', 'Year Years (Child)',
       'Year Years (Child, Adult)', 'Month Years (Child, Adult)',
       'Month Years (Child)', 'Hours (Child)', 'Months (Child)',
       'Months Years (Child, Adult)', 'Years Years (Older Adult)',
       'Months older (Adult, Older Adult)', 'Months Years (Child)',
       'Days Years (Child, Adult)', 'Month (Child)',
       'Month older (Child, Adult, Older Adult)',
       'Weeks Years (Child, Adult)', 'Months Months (Child)',
       'Days older (Child, Adult, Older Adult)', 'Year (Child)'],
      dtype=object)
```

```
df["Age"]=df["Age"].apply(lambda x:str(x).replace('Years','')if 'Years' in str(x) else str(x))
df["Age"]=df["Age"].apply(lambda x:str(x).replace('Year','')if 'Year' in str(x) else str(x))
```

This line looks at each value x in the Age column.

- If "Years" or “Year” is found in the string, it replaces it with an empty string (''), effectively removing it.
- If "Years" or “Year” is not in the string, it just returns the string unchanged.

Output:

```
df.Age.unique()
```

```
array([' older (Adult, Older Adult)', 'Child, Adult, Older Adult',
       '  (Adult)', '  (Adult, Older Adult)',
       ' older (Child, Adult, Older Adult)',
       '  (Child, Adult, Older Adult)', ' (Child, Adult, Older Adult)',
       'Days (Child)', ' (Child, Adult)', ' older (Older Adult)',
       '  (Child, Adult)', ' (Child)',
       'Months older (Child, Adult, Older Adult)', '  (Child)',
       'Months  (Child, Adult, Older Adult)', 'Minutes (Child)',
       'Weeks Weeks (Child)', 'Month  (Child, Adult, Older Adult)',
       'Month  (Child, Adult)', 'Month  (Child)', 'Hours (Child)',
       'Months (Child)', 'Months  (Child, Adult)', '  (Older Adult)',
       'Months older (Adult, Older Adult)', 'Months  (Child)',
       'Days  (Child, Adult)', 'Month (Child)',
       'Month older (Child, Adult, Older Adult)', 'Weeks  (Child, Adult)',
       'Months Months (Child)', 'Days older (Child, Adult, Older Adult)'],
      dtype=object)
```

```
df["Age"]=df["Age"].apply(lambda x:str(x).replace('Months','')if 'Months' in str(x) else str(x))
df["Age"]=df["Age"].apply(lambda x:str(x).replace('Month','')if 'Month' in str(x) else str(x))
df["Age"]=df["Age"].apply(lambda x:str(x).replace('Days','')if 'Days' in str(x) else str(x))
df["Age"]=df["Age"].apply(lambda x:str(x).replace('Weeks','')if 'Weeks' in str(x) else str(x))
df["Age"]=df["Age"].apply(lambda x:str(x).replace('Hours','')if 'Hours' in str(x) else str(x))
```

Explain This Code:

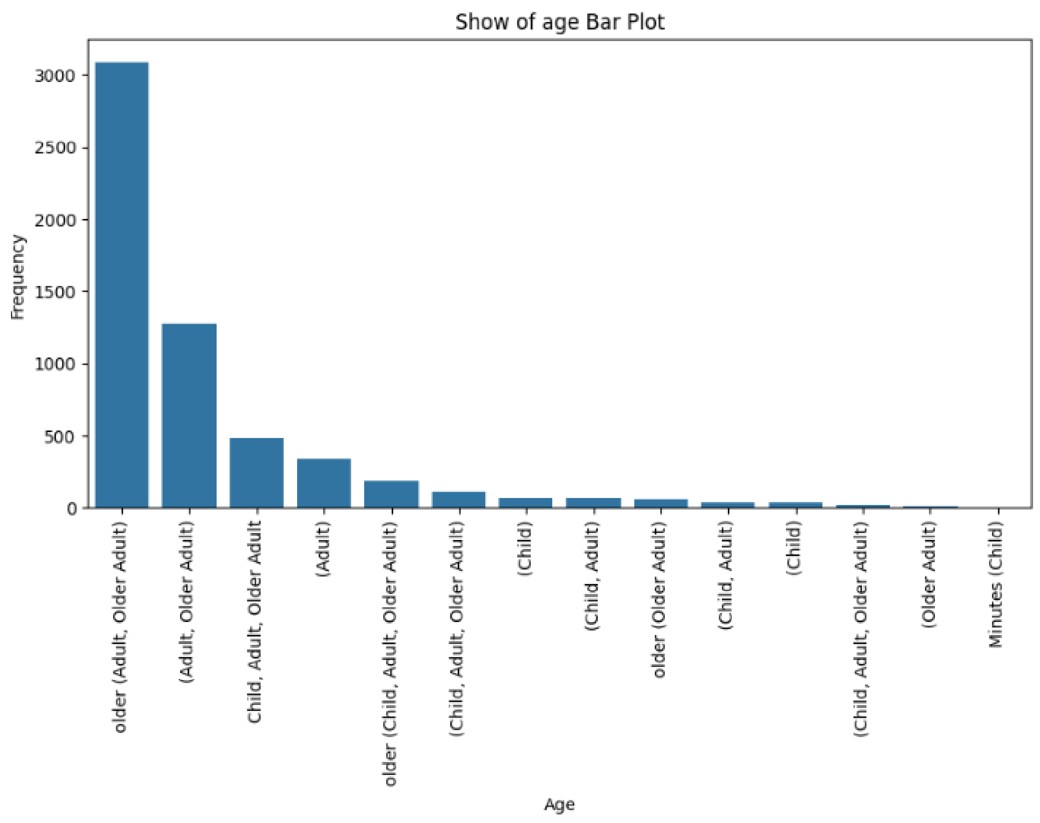
- This code removes the words "Months", "Month", "Days", "Weeks", and "Hours" from each value in the "Age" column.
- For each word, it checks if it exists in the text and replaces it with an empty string.

Output:

```
df.Age.unique()
```

```
array([' older (Adult, Older Adult)', 'Child, Adult, Older Adult',
       '  (Adult)', '  (Adult, Older Adult)',
       ' older (Child, Adult, Older Adult)',
       '  (Child, Adult, Older Adult)', ' (Child, Adult, Older Adult)',
       ' (Child)', ' (Child, Adult)', ' older (Older Adult)',
       '  (Child, Adult)', '  (Child)', 'Minutes (Child)',
       '  (Older Adult)'], dtype=object)
```

```python
plt.figure(figsize=(10,5))
sns.barplot(x=df['Age'].value_counts().index,
            y=df['Age'].value_counts().values)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Show of age Bar Plot')
plt.xticks(rotation=90)
plt.show()
```

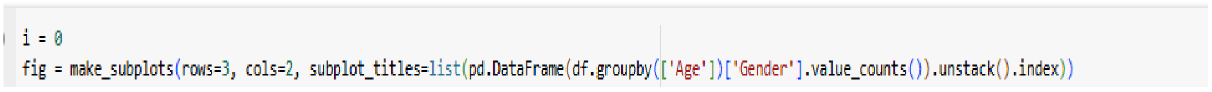


Insights:

- The most frequent age category is "older (Adult, Older Adult)", showing over 3000 trials.
- This reflects the higher vulnerability of older age groups to COVID-19.

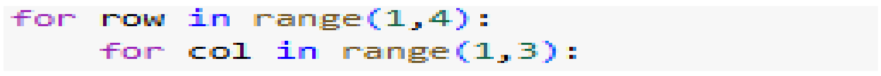
```python
i = 0
fig = make_subplots(rows=3, cols=2, subplot_titles=list(pd.DataFrame(df.groupby(['Age'])['Gender'].value_counts()).unstack().index))
for row in range(1,4):
    for col in range(1,3):
        dt = pd.DataFrame(df.groupby(['Age'])['Gender'].value_counts()).unstack().iloc[i]
        # Check if dt is a Series and convert it to DataFrame if necessary
        if isinstance(dt, pd.Series):
            dt = dt.to_frame(name='Gender')
            #This converts it to a DataFrame with 'Gender' as column name.
        fig.add_trace(go.Bar(x=dt.index, y=dt.Gender.values), row=row, col=col) #Use dt.index instead of dt.Gender.index
        i+=1
fig.show()
```

Explain this code:



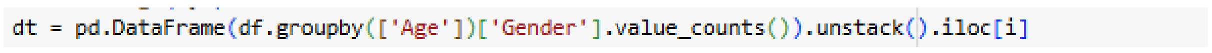
```
i = 0
fig = make_subplots(rows=3, cols=2, subplot_titles=list(pd.DataFrame(df.groupby(['Age'])['Gender'].value_counts()).unstack().index))
```

- **make_subplots(rows=3, cols=2):** Creating a grid of subplots — 3 rows × 2 columns = 6 plots total.
- **subplot_titles=...:**Titles are set using the age groups (from **df.groupby(['Age'])['Gender'].value_counts()**).
    - .unstack().index gives the list of unique Age categories.



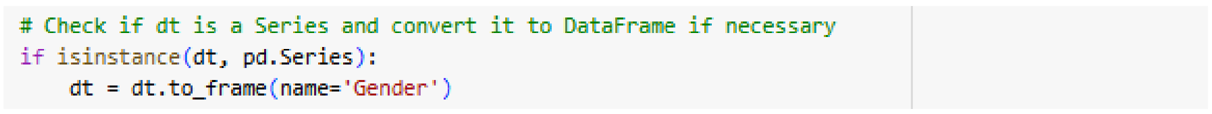
```
for row in range(1,4):
    for col in range(1,3):
```

- This nested loop iterates through all subplot positions (row 1–3 and column 1–2).
- **i** is used to index into the Age group data for each subplot.



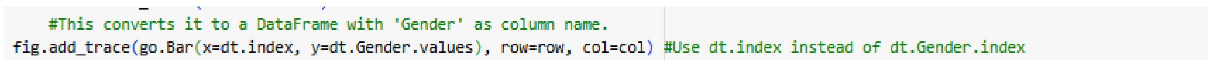
```
dt = pd.DataFrame(df.groupby(['Age'])['Gender'].value_counts()).unstack().iloc[i]
```

- This line fetches the i-th Age group and its associated Gender distribution.
- The **.unstack()** converts the multi-index result of groupby into a table format.
- **iloc[i]** selects the i-th row (Age group).



```
# Check if dt is a Series and convert it to DataFrame if necessary
if isinstance(dt, pd.Series):
    dt = dt.to_frame(name='Gender')
```

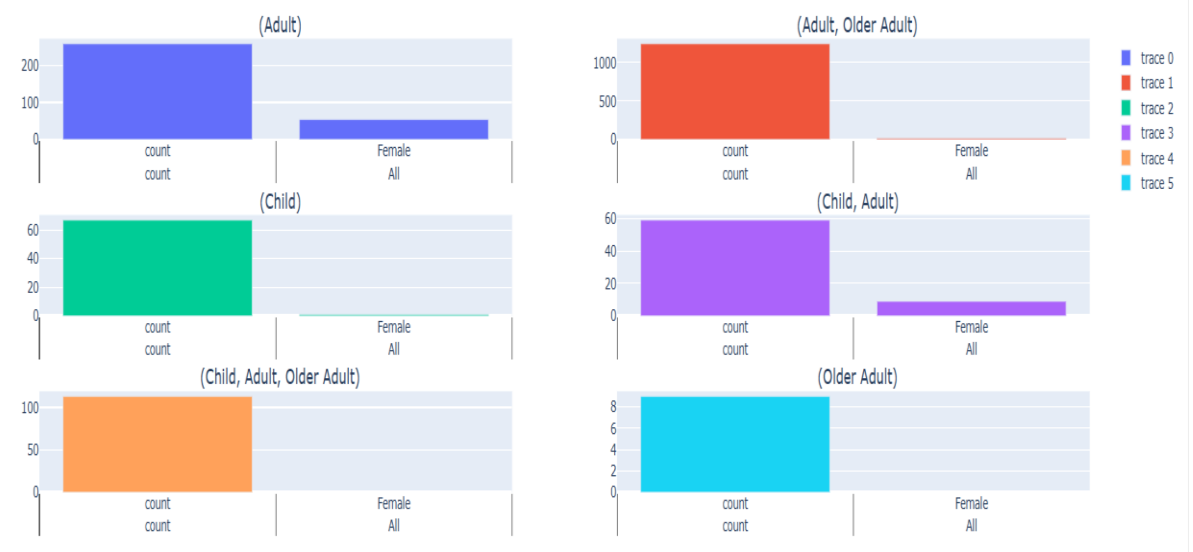
- If the selected dt is a Series, it's converted into a DataFrame with a column name Gender.
- This ensures consistent formatting for plotting.



```
    #This converts it to a DataFrame with 'Gender' as column name.
fig.add_trace(go.Bar(x=dt.index, y=dt.Gender.values), row=row, col=col) #Use dt.index instead of dt.Gender.index
```

- **go.Bar(...)** creates a bar chart:
- **x=dt.index:** Gender categories (Male, Female, All).
- **y=dt.Gender.values:** Number of studies for each gender.
- It's placed at the position (row, col) on the subplot grid.

Output:



Insights:

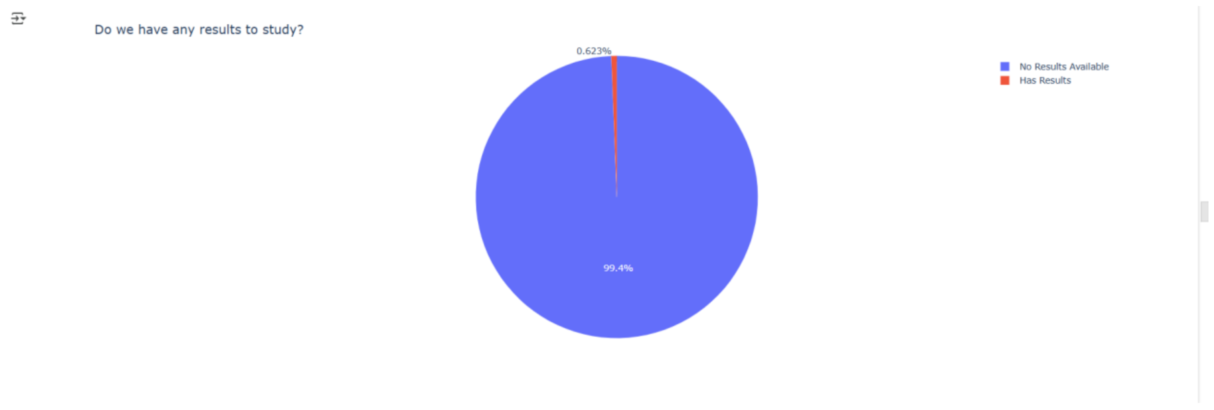
- This shows the 6 subplots, each representing gender distribution for one age group**.**
- Most studies have taken data from All Genders;
- In (Adult) and (Child, Adult) Category there is significant number of Female patients considered for the studies

3.7 Exploring study results

```
import plotly.express as px
fig = px.pie(df,'Study Results')
fig.update_layout(title='Do we have any results to study?')
fig.show()
```

Identifies how many trials have published results vs. not

Output:



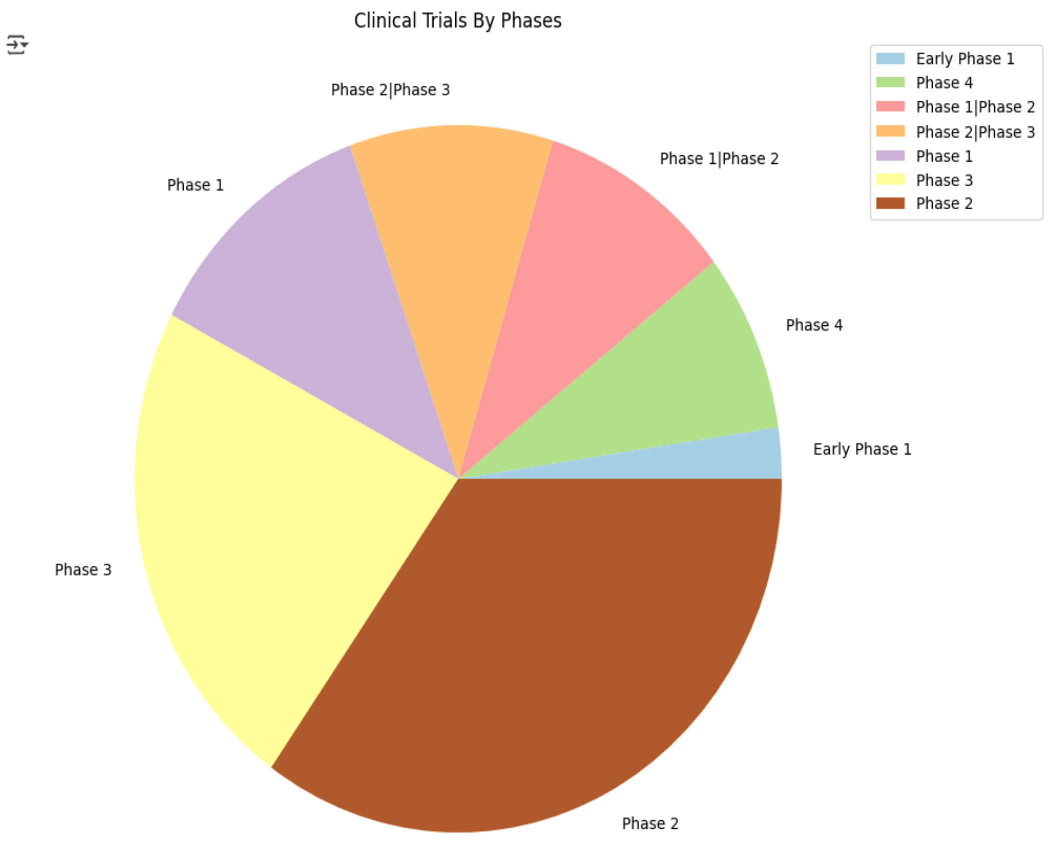
Insights:

- "No Results Available"~99.4% of the studies.
- "Has Results" is only ~0.6%.

3.8 Exploring Study Phases

```
# Pie chart of clinical trial phases (excluding 'Not Applicable')
df[df['Phases'] != 'Not Applicable']['Phases'].value_counts().sort_values().plot(
    kind='pie', figsize=(20, 10), colormap='Paired', title='Clinical Trials By Phases', legend=True
)
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.ylabel('')
plt.show()
```

- **value_counts():** counts how many trials are in each phase.
- **sort_values():** arranges them in ascending order (so smaller segments appear first in the pie).
- **kind='pie':** creates a pie chart.
- **figsize=(20, 10):** large figure size for clarity.
- **colormap='Paired':** uses a colorful palette.
- **title='Clinical Trials By Phases':** chart title.
- **legend=True:** adds a legend.
- Places the legend outside top-right
- Removes the default y-axis label



- **Phase 1** (safety)
- **Phase 2** (efficacy and side effects)
- **Phase 3** (confirmation on larger population)
- **Phase 4** (post-marketing, long-term use)

Insights:

The majority of clinical trials are in Phase 2, indicating a strong focus on evaluating the effectiveness of treatments. Very few trials are in Phase 4, which could mean that few interventions have reached the post-marketing stage.

3.9 Interventional Studies

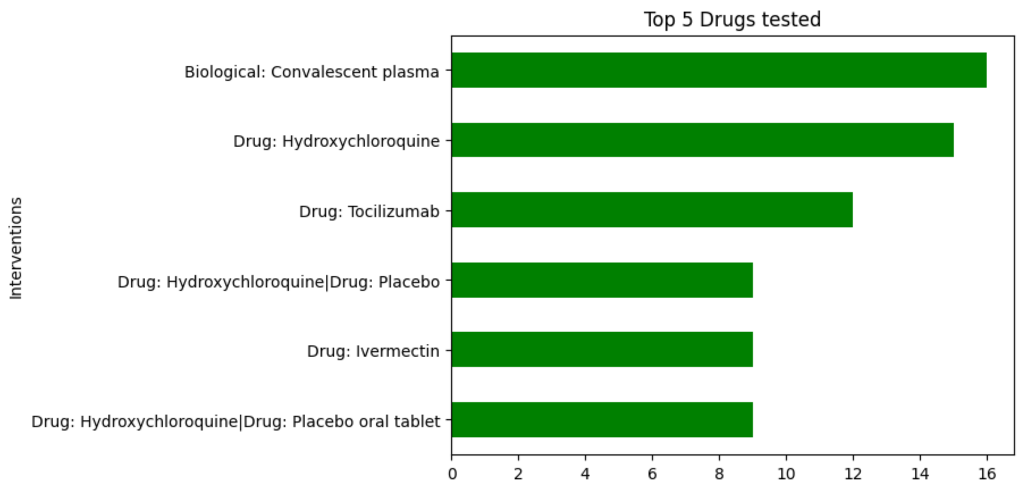
```
df.Interventions.unique()
```

```
array(['Diagnostic Test: ID Now™ COVID-19 Screening Test',
       'Drug: Drug COVID19-0001-USR|Drug: normal saline',
       'Other: Lung CT scan analysis in COVID-19 patients', ...,
       'Other: Antibiotic treatment|Other: No antibiotic treatment',
       'Behavioral: Yoga',
       'Behavioral: Brief Cognitive Behavioral Therapy for Chronic Pain (BCBT-CP)|Other: BCBT-CP Booster'],
      dtype=object)
```

```
] interventions = df[df['Study Type']=='Interventional']
  interventions['Interventions'].value_counts().head(6).sort_values().plot(kind='barh', color='g', title='Top 5 Drugs tested')
```

Output:

```
<Axes: title={'center': 'Top 5 Drugs tested'}, ylabel='Interventions'>
```



Insights:

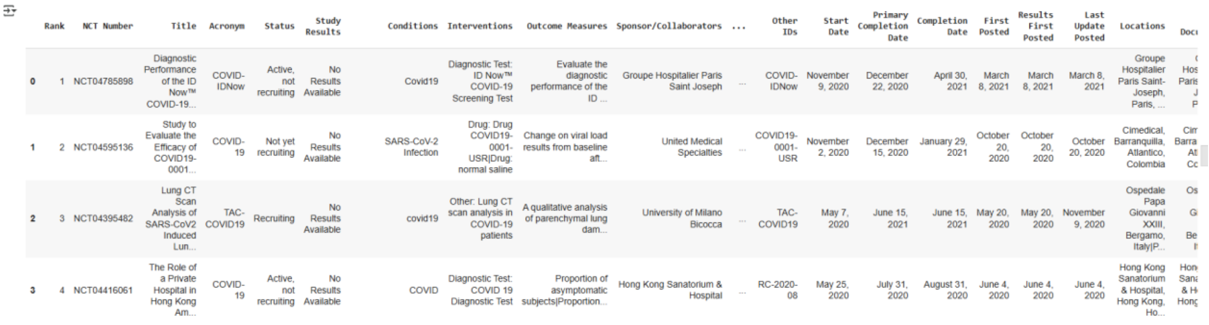
- **Most Tested Drug:**
    - Biological: Convalescent plasma was the most tested, indicating researchers were highly interested in passive immunity (using plasma from recovered patients).
- **Hydroxychloroquine** appears in multiple entries:
    - Drug: Hydroxychloroquine
    - With Placebo
    - With Placebo oral tablet
    - This shows hydroxychloroquine was heavily tested in various trial setups, reflecting early interest and debate around its effectiveness.

```python
df = df.ffill(axis = 1)
```

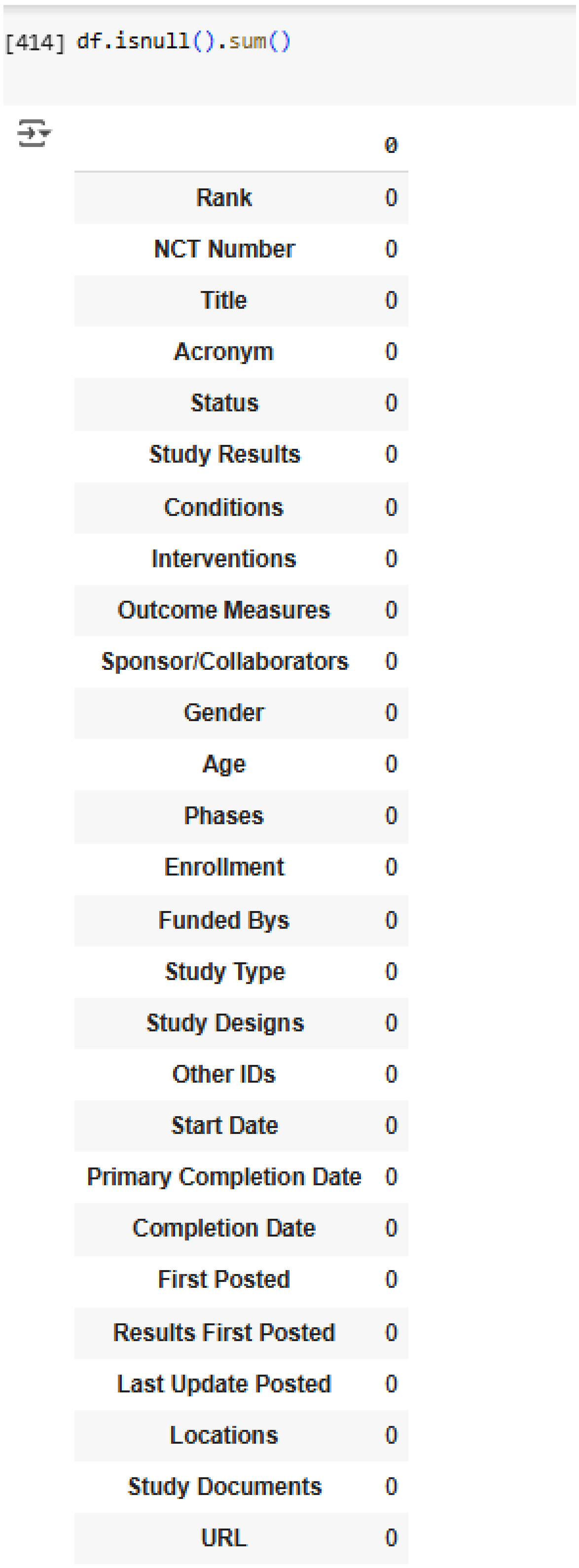
Forward Filling Missing Data: Used df.ffill(axis=1) to fill missing data horizontally (row-wise), which is unusual and should be verified depending on context.

```python
df.head()
```

Output:



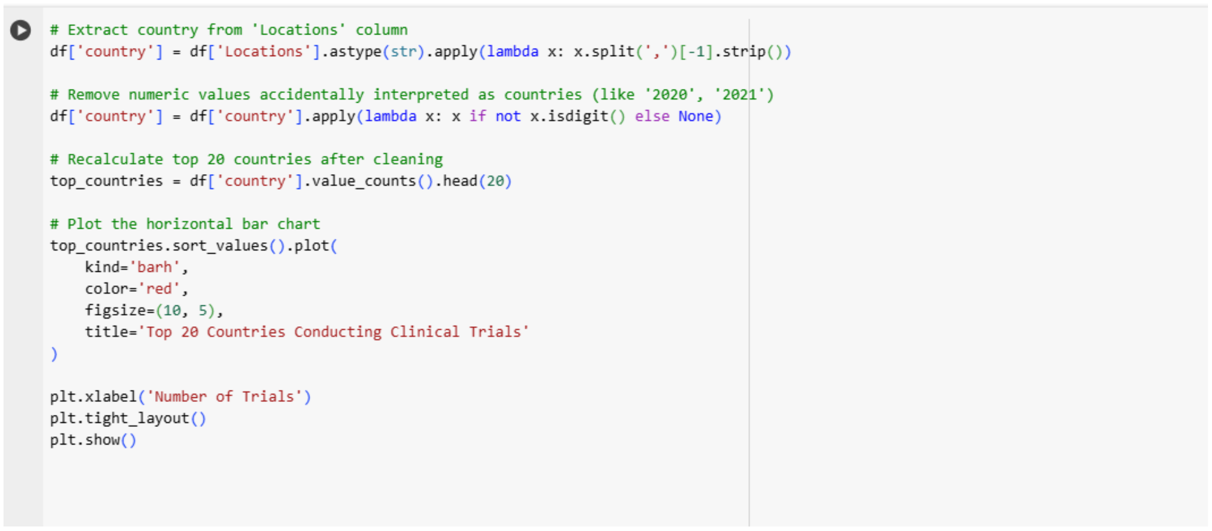
	Rank	NCT Number	Title	Acronym	Status	Study Results	Conditions	Interventions	Outcome Measures	Sponsor/Collaborators	...	Other IDs	Start Date	Primary Completion Date	Completion Date	First Posted	Results First Posted	Last Update Posted	Locations	Doc
0	1	NCT04785898	Diagnostic Performance of the ID Now™ COVID-19...	COVID-IDNow	Active, not recruiting	No Results Available	Covid19	Diagnostic Test: ID Now™ COVID-19 Screening Test	Evaluate the diagnostic performance of the ID ...	Groupe Hospitalier Paris Saint Joseph	...	COVID-IDNow	November 9, 2020	December 22, 2020	April 30, 2021	March 8, 2021	March 8, 2021	March 8, 2021	Groupe Hospitalier Paris Saint-Joseph, Paris, ...	Hos Paris J P
1	2	NCT04595136	Study to Evaluate the Efficacy of COVID19-0001...	COVID-19	Not yet recruiting	No Results Available	SARS-CoV-2 Infection	Drug: Drug COVID19-0001-USR|Drug: normal saline	Change on viral load results from baseline aft...	United Medical Specialties	...	COVID19-0001-USR	November 2, 2020	December 15, 2020	January 29, 2021	October 20, 2020	October 20, 2020	October 20, 2020	Cimedical, Barranquilla, Atlantico, Colombia	Cim Barra At Cc
2	3	NCT04395482	Lung CT Scan Analysis of SARS-CoV2 Induced Lun...	TAC-COVID19	Recruiting	No Results Available	covid19	Other: Lung CT scan analysis in COVID-19 patients	A qualitative analysis of parenchymal lung dam...	University of Milano Bicocca	...	TAC-COVID19	May 7, 2020	June 15, 2021	June 15, 2021	May 20, 2020	May 20, 2020	November 9, 2020	Ospedale Papa Giovanni XXIII, Bergamo, Italy|P...	Os Gi Be I
3	4	NCT04416061	The Role of a Private Hospital in Hong Kong Am...	COVID-19	Active, not recruiting	No Results Available	COVID	Diagnostic Test: COVID 19 Diagnostic Test	Proportion of asymptomatic subjects|Proportion...	Hong Kong Sanatorium & Hospital	...	RC-2020-08	May 25, 2020	July 31, 2020	August 31, 2020	June 4, 2020	June 4, 2020	June 4, 2020	Hong Kong Sanatorium & Hospital, Hong Kong, Ho...	Hon Sana & H Hong



```
[414] df.isnull().sum()
```

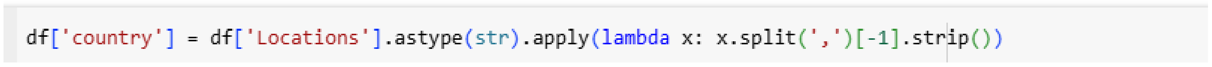
```
                            0
                 Rank       0
           NCT Number       0
                Title       0
              Acronym       0
               Status       0
        Study Results       0
           Conditions       0
        Interventions       0
     Outcome Measures       0
Sponsor/Collaborators       0
               Gender       0
                  Age       0
               Phases       0
           Enrollment       0
          Funded Bys        0
           Study Type       0
        Study Designs       0
            Other IDs       0
           Start Date       0
Primary Completion Date     0
      Completion Date       0
         First Posted       0
 Results First Posted       0
   Last Update Posted       0
            Locations       0
      Study Documents       0
                  URL       0
```

3.10 Clinical Trials Distribution By Country



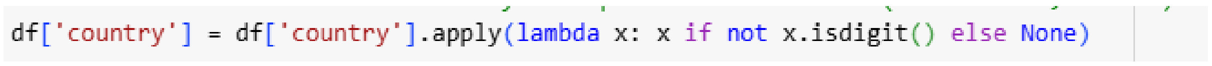
```python
# Extract country from 'Locations' column
df['country'] = df['Locations'].astype(str).apply(lambda x: x.split(',')[-1].strip())

# Remove numeric values accidentally interpreted as countries (like '2020', '2021')
df['country'] = df['country'].apply(lambda x: x if not x.isdigit() else None)

# Recalculate top 20 countries after cleaning
top_countries = df['country'].value_counts().head(20)

# Plot the horizontal bar chart
top_countries.sort_values().plot(
    kind='barh',
    color='red',
    figsize=(10, 5),
    title='Top 20 Countries Conducting Clinical Trials'
)

plt.xlabel('Number of Trials')
plt.tight_layout()
plt.show()
```

Explain Code:



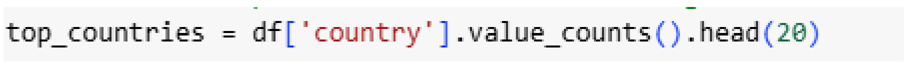
```python
df['country'] = df['Locations'].astype(str).apply(lambda x: x.split(',')[-1].strip())
```

- This extracts the country name from the Locations column.
- The Locations field typically contains values like: "New York, United States"
- This code splits by comma and takes the last part : "United States".



```python
df['country'] = df['country'].apply(lambda x: x if not x.isdigit() else None)
```

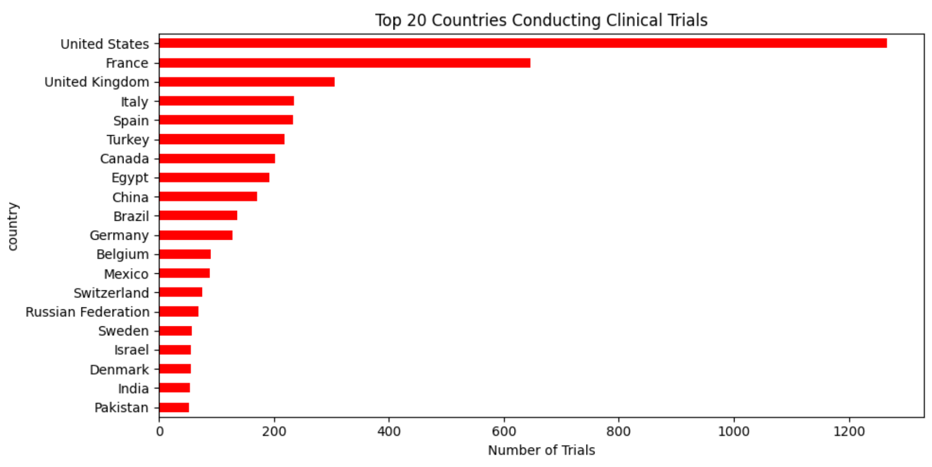
- Some entries end with a number (e.g., a zip code or a year like '2020', '2021') instead of a valid country name.
- This line replaces any such numeric "countries" with None.



```python
top_countries = df['country'].value_counts().head(20)
```

- Counts how many times each country appears in the cleaned 'country' column.
- Selects only the top 20 most frequent countries.

Output:



Insights:

- The U.S. dominates global clinical trial efforts possibly due to:
  - Large research infrastructure.
  - High number of biotech and pharma companies.
  - Government support (e.g., NIH, FDA).
- European countries are also major players.

```python
# Choropleth Map
country_data = df['country'].value_counts().reset_index()
country_data.columns = ['Country', 'Trial_Count']
country_data['Country'] = country_data['Country'].str.strip()

fig = px.choropleth(
    country_data,
    locations="Country",
    locationmode="country names",
    color="Trial_Count",
    hover_name="Country",
    color_continuous_scale=px.colors.sequential.Plasma,
    title="Clinical Trial Distribution by Country"
)

fig.update_layout(
    title_x=0.5,
    geo=dict(showframe=False, showcoastlines=False),
    coloraxis_colorbar=dict(
        title="Trial Count",
        ticks="outside",
        tickformat=","
    )
)

fig.show()
```

Explain Code:

```python
country_data = df['country'].value_counts().reset_index()
country_data.columns = ['Country', 'Trial_Count']
country_data['Country'] = country_data['Country'].str.strip()
```

- **value_counts()** counts how many times each country appears in the df['country'] column.

- **reset_index()** converts the result into a DataFrame (from a Series), so you can use it like a table.
- Renames the columns to:
  - 'Country': the country name
  - 'Trial_Count': how many clinical trials were conducted there
- **str.strip()** removes any extra whitespace from country names to avoid matching issues in the map.

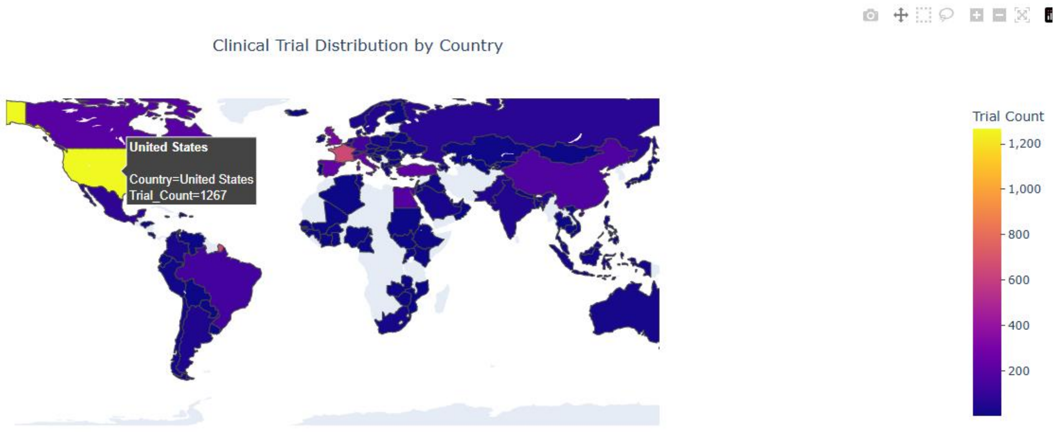
```
fig = px.choropleth(
    country_data,
    locations="Country",
    locationmode="country names",
    color="Trial_Count",
    hover_name="Country",
    color_continuous_scale=px.colors.sequential.Plasma,
    title="Clinical Trial Distribution by Country"
)
```

- This draws a world map where each country is shaded based on the number of trials.
- Darker or more intense colors mean more trials.

```
fig.update_layout(
    title_x=0.5,
    geo=dict(showframe=False, showcoastlines=False),
    coloraxis_colorbar=dict(
        title="Trial Count",
        ticks="outside",
        tickformat=","
    )
)
```

- **title_x=0.5:** Centers the title.
- **showframe=False, showcoastlines=False**: Makes the map cleaner by hiding borders and coastlines.
- Customizes the color legend (color bar on the side):
  - Title is "Trial Count".
  - Ticks are outside the bar.
  - **tickformat=","** adds comma separators (e.g., 1,000).

Output:



Insights:

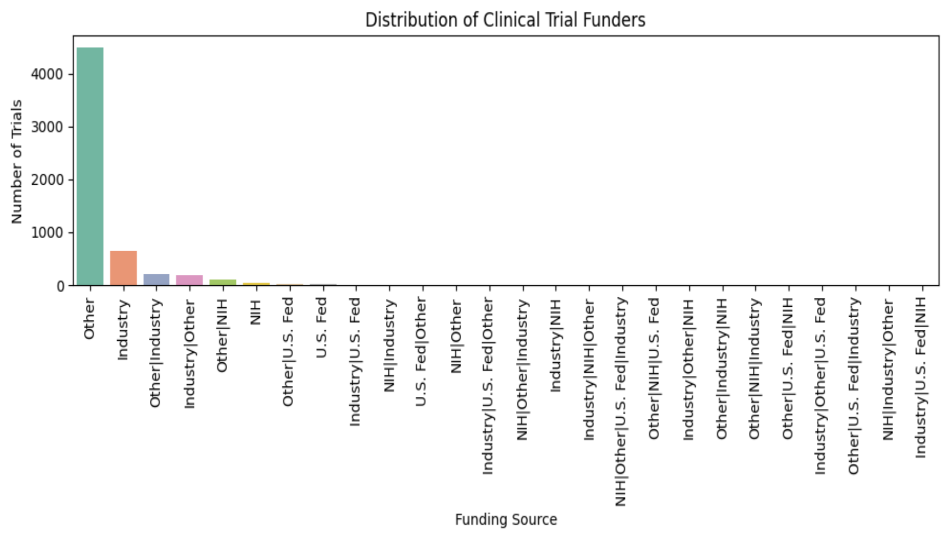
- **United States** has the highest number of trials (1,267), marked in bright yellow.
- Other countries like China, Canada, UK, Germany, and Japan are also highlighted but in cooler tones (indicating fewer trials than the US).

3.11 Distribution of Clinical Trial Funders

```
# Bar plot of all funding sources using Seaborn
plt.figure(figsize=(10, 5))
sns.barplot(
    x=df['Funded Bys'].value_counts().index,
    y=df['Funded Bys'].value_counts().values,
    palette='Set2'
)
plt.xlabel('Funding Source')
plt.ylabel('Number of Trials')
plt.title('Distribution of Clinical Trial Funders')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

- Sets the figure size to 10 inches wide and 5 inches tall.
- Uses **sns.barplot()** from Seaborn to draw the bar chart.
- x: Funding source names (automatically sorted by count using value_counts().index.
- y: Number of clinical trials for each funding source.
- **palette='Set2':** Applies a pleasant Seaborn color palette for distinct bar colors.
- Adds descriptive axis labels and a chart title.
- Rotates x-axis labels vertically (90 degrees) to prevent overlapping, especially useful when there are many categories.
- **tight_layout()** ensures labels/titles fit nicely in the figure area.
- **plt.show()** renders the plot in the notebook or script output.

Output:



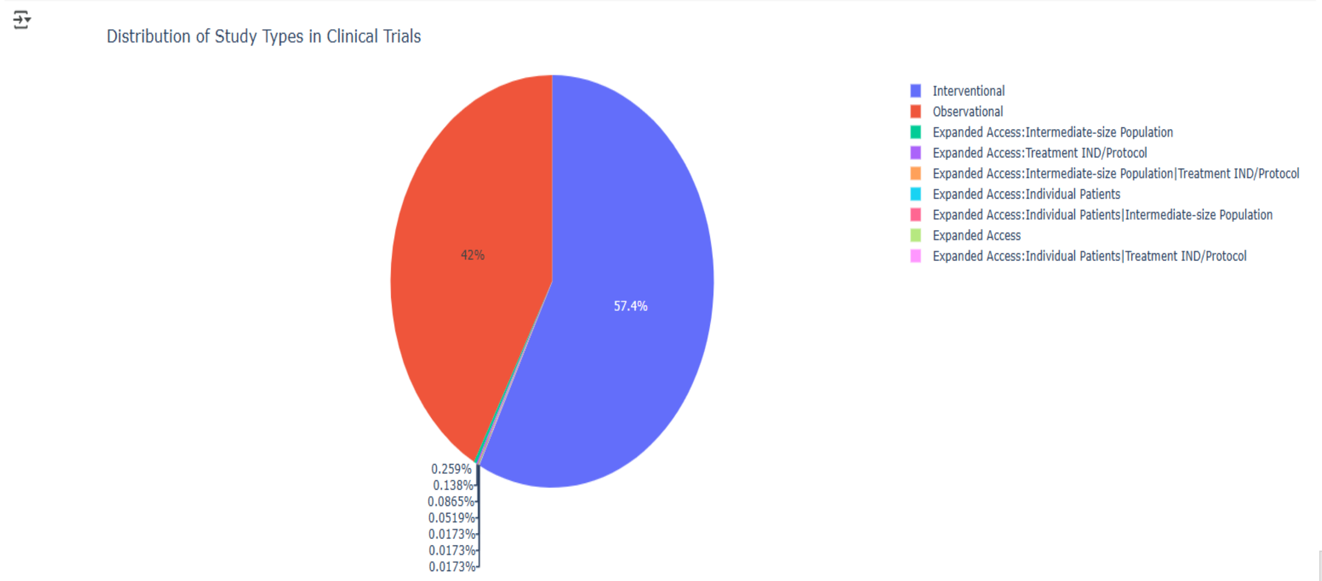
Insights:

- "Other" is the dominant funder, with over 4,000 trials, which significantly overshadows all other funders.
- Industry comes next, funding around 700–800 trials.
- Other combinations (like "Other|Industry", "Industry|Other", "Other|NIH", etc.) contribute much less, each with fewer than 500 trials.

3.12 Distribution of Study Types in COVID-19 Clinical Trials

```
import plotly.express as px
fig = px.pie(df,'Study Type')
fig.update_layout(title='Distribution of Study Types in Clinical Trials')
fig.show()
```

Output:



Insights:

- **57.4%** of all clinical trials are Interventional**.**
- **42%** of the studies are Observational**.**
- All Expanded Access categories combined make up less than 1% of total trials.

3.13 Correlation Heatmap

```
# First, convert categorical to numeric
df_corr = df.copy()

# Example: Encode 'Gender', 'Status', 'Study Type' numerically
df_corr['Gender'] = df_corr['Gender'].astype('category').cat.codes
df_corr['Status'] = df_corr['Status'].astype('category').cat.codes
df_corr['Study Type'] = df_corr['Study Type'].astype('category').cat.codes
df_corr['Phases'] = df_corr['Phases'].astype('category').cat.codes
df_corr['Funded Bys'] = df_corr['Funded Bys'].astype('category').cat.codes

# Convert Enrollment to numeric, coercing errors to NaN
df_corr['Enrollment'] = pd.to_numeric(df_corr['Enrollment'], errors='coerce')

# Compute correlation
corr = df_corr[['Gender', 'Status', 'Study Type', 'Phases', 'Funded Bys', 'Enrollment']].corr()

# Plot
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

Explain Code:

```
# Example: Encode 'Gender', 'Status', 'Study Type' numerically
df_corr['Gender'] = df_corr['Gender'].astype('category').cat.codes
df_corr['Status'] = df_corr['Status'].astype('category').cat.codes
df_corr['Study Type'] = df_corr['Study Type'].astype('category').cat.codes
df_corr['Phases'] = df_corr['Phases'].astype('category').cat.codes
df_corr['Funded Bys'] = df_corr['Funded Bys'].astype('category').cat.codes
```

- Each categorical column (Gender, Status, Study Type, etc.) is converted to numeric codes.
- .astype('category'): Converts the column to a pandas category type.
- .cat.codes: Assigns a numeric code to each unique category.

```
# Convert Enrollment to numeric, coercing errors to NaN
df_corr['Enrollment'] = pd.to_numeric(df_corr['Enrollment'], errors='coerce')
```

- Ensures the Enrollment column is numeric.
- errors='coerce': Converts invalid entries to NaN (null values).

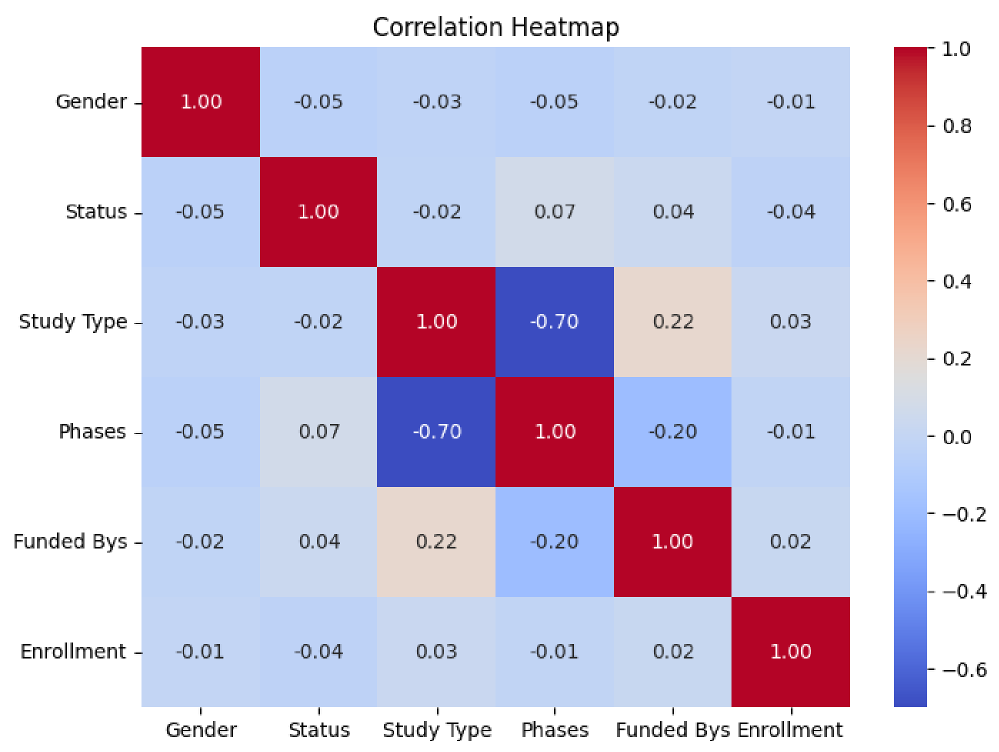
```
# Compute correlation
corr = df_corr[['Gender', 'Status', 'Study Type', 'Phases', 'Funded Bys', 'Enrollment']].corr()
```

- Selects the columns of interest and calculates the pairwise correlation between them using Pearson correlation (default in .corr()).

```
# Plot
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

- **annot=True:** Displays the correlation coefficients.
- **cmap='coolwarm':** Color palette, where red/blue indicates positive/negative correlation.
- **fmt=".2f":** Formats correlation values to 2 decimal places.

Output:



Insights:

Variables	Correlation	Interpretation
Study Type & Phases	**−0.70**	Strong negative correlation — as the study type changes, it is likely associated with a different (often opposite) phase.
Study Type & Funded Bys	+0.22	Weak positive correlation — some study types are more likely to be funded by specific sources.
Phases & Funded Bys	−0.20	Weak negative correlation — funding type may differ slightly across phases.
Gender, Status, Enrollment	~0.00	No significant correlation with other variables — these are largely independent.

4. Summary & Key Findings

- **Top Countries for Clinical Trials**
  - The United States (States) conducted the maximum number of COVID-19 clinical trials.
  - France, United Kingdom, Italy, and Spain were also major contributors.
  - This shows that developed countries were leading clinical research during the pandemic.
- **Funding Sources**

    - Most clinical trials were funded under the category "Other".
    - The second largest funding source was "Industry" (pharmaceutical companies, biotech firms).
    - Combination funding like Industry|Other also appeared but was much less frequent.
    - **Conclusion:** Clinical trials were majorly privately or independently funded rather than solely by governments or official health bodies.
- **Geographic Distribution of Clinical Trials**
    - A choropleth map reveals that the United States leads globally, with 1,267 trials, far exceeding other countries.
    - Other countries with notable trial activity include China**,** Canada**,** Germany, and the United Kingdom**.**
- **Clinical Trial Phases:**
    - Most trials were in Phase 2 and Phase 3, indicating a strong focus on testing effectiveness and safety in larger populations.
    - A smaller portion were in Phase 1 (early safety) and Phase 4 (post-marketing surveillance).
- **Study Types & Results:**
    - The majority of clinical studies were interventional in nature, focusing on testing treatments or preventive measures.
    - However, over 99% of studies had no available results, highlighting a significant gap between study initiation and result reporting.
- **Drug Interventions:**
    - Hydroxychloroquine, Ivermectin, Convalescent Plasma, and Tocilizumab were among the most frequently tested drugs.
    - Combinations involving placebos and existing drugs show efforts to compare effectiveness rigorously.

5. Why analyze the COVID-19 Clinical Trials?

- **Understand Global Research Response**
    - The pandemic triggered an unprecedented wave of medical research**.**
    - By analyzing clinical trials, we can understand how different countries, organizations, and researchers responded to the crisis.
- **Identify Most Tested Treatments**
    - It helps identify which drugs or therapies were most frequently tested (e.g., Hydroxychloroquine, Ivermectin).
    - This can reveal trends, priorities, and hypotheses researchers believed had the most potential.
- **Spot Gaps in Data Availability**
    - The analysis shows that most studies lacked published results, which highlights a gap between research and reporting**.**
    - Knowing this helps policymakers and funding bodies improve future transparency and accountability.
- **Assess Global Collaboration**
    - The dataset reflects which countries were most active in clinical research.

    - This provides insight into international cooperation, funding availability, and healthcare research capacity**.**
- **Understand Research Focus by Phase**
    - Examining trial phases helps understand how far treatments progressed.
    - For example, many studies were in Phase 2 or 3, showing a focus on evaluating effectiveness at scale.
- **Guide Future Pandemics Preparedness**
    - Learning from COVID-19 helps prepare for future pandemics**.**
    - Understanding what worked, where efforts lagged, and how trials were distributed is critical for better response planning.