```
mkdir -p raw_data
```

```
cd raw_data
```

```
!wget -q https://clinicaltrials.gov/AllPublicXML.zip
```

```
!unzip -q AllPublicXML.zip
```

```
import csv
import xml.etree.ElementTree as ET


def pull_tag_content(root_node, tag_path):
    content_list = []
    stack = [(root_node, '')]

    while stack:
        node, prefix = stack.pop()
        path = prefix + '/' + node.tag

        if path == tag_path and node.text:
            content_list.append(node.text)

        for child in node:
            stack.append((child, path))

    return content_list

def extract_criteria(criteria_textblock):
    # Find the positions of "Inclusion Criteria:" and "Exclusion Criteria:"
    c1 = criteria_textblock.find("Inclusion Criteria:",0)
    c2 = criteria_textblock.find("Exclusion Criteria:",0)

    # Extract the Inclusion Criteria and handle missing criteria
    if c1 >= 0:
        if c2 >= 0:
            inclusion_criteria = criteria_textblock[c1 + len("Inclusion Criteria:"):c2].strip()
        else:
            inclusion_criteria = criteria_textblock[c1 + len("Inclusion Criteria:"):].strip()
    else:
        inclusion_criteria = ""

    # Extract the Exclusion Criteria and handle missing criteria
    if c2 >= 0:
        if c1 >= 0:
            exclusion_criteria = criteria_textblock[c2 + len("Exclusion Criteria:"):].strip()
        else:
            exclusion_criteria = criteria_textblock[c2 + len("Exclusion Criteria:"):].strip()
    else:
        exclusion_criteria = ""
    return inclusion_criteria, exclusion_criteria

def read_xml_file(file_path):
    try:
        # Parse the XML file
        tree = ET.parse(file_path)
        root = tree.getroot()

        clinical_data = {
            'nct_id': root.findtext('id_info/nct_id',''),
            'nct_id': root.findtext('id_info/nct_id',''),
            'brief_title': root.findtext('brief_title',''),
            'official_title': root.findtext('official_title',''),
            'agency': root.findtext('sponsors/lead_sponsor/agency',''),
            'agency_class': root.findtext('sponsors/lead_sponsor/agency_class',''),
            'collaborator_agency': root.findtext('sponsors/collaborator/agency',''),
            'brief_summary': root.findtext('brief_summary/textblock',''),
            'detailed_description': root.findtext('detailed_description/textblock',''),
            # 'conditions': root.findtext('condition',''),
            'overall_status': root.findtext('overall_status',''),
            'phase': root.findtext('phase',''),
            'study_type': root.findtext('study_type',''),
            'has_expanded_access': root.findtext('has_expanded_access',''),
            'intervention': root.findtext('intervention',''),
            'intervention_type': root.findtext('intervention/intervention_type',''),
            'intervention_name': root.findtext('intervention/intervention_name',''),
            'lead_sponsor_agency': root.find('sponsors/lead_sponsor/agency',''),
            'primary_completion_date': root.findtext('primary_completion_date',''),
            'start_date': root.findtext('start_date',''),
            'completion_date': root.findtext('completion_date',''),
            'gender': root.findtext('eligibility/gender',''),
            'minimum_age': root.findtext('eligibility/minimum_age',''),
            'maximum_age': root.findtext('eligibility/maximum_age',''),
            'healthy_volunteers': root.findtext('eligibility/healthy_volunteers',''),
            'why_stopped': root.findtext('why_stopped',''),
        }
        # check for multipal marks
        conditions = pull_tag_content(root, '/clinical_study/condition')
        keywords = pull_tag_content(root, '/clinical_study/keyword')
        clinical_data['conditions'] = conditions
        clinical_data['keywords'] = keywords
        # Extract Inclusion and Exclusion Criteria
        criteria_textblock = root.findtext('eligibility/criteria/textblock','')
        inclusion_criteria, exclusion_criteria = extract_criteria(criteria_textblock)
        clinical_data['inclusion_criteria'] = inclusion_criteria
        clinical_data['exclusion_criteria'] = exclusion_criteria
        return clinical_data

    except FileNotFoundError:
        print(f"Error: File '{file_path}' not found.")
        return None
    except ET.ParseError:
        print(f"Error: Invalid XML format in '{file_path}'.")
        return None


def save_to_csv(data_list, csv_file):
    with open(csv_file, 'w', newline='', encoding='utf-8') as f:
        fieldnames = data_list[0].keys()
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(data_list)
```

```
raw_data_dir = '/content/raw_data'
import os
# # Create an empty list to store all the file paths
all_file_paths = []

# # Use os.walk to traverse through all subdirectories in 'raw_data'
for dirpath, _, filenames in os.walk(raw_data_dir):
    # Concatenate the directory path with the filenames to get the full file paths
    file_paths = [os.path.join(dirpath, filename) for filename in filenames]
    # Extend the all_file_paths list with the file_paths list for each subdirectory
    all_file_paths.extend(file_paths)

```

```
data_list = []
for file_path in all_file_paths:
    if file_path.endswith(".xml"):
        clinical_data = read_xml_file(file_path)
        if clinical_data is not None:
            data_list.append(clinical_data)


```

```
mkdir -p data_output
```

```

csv_file= '/content/data_output/combine_output.csv'
if data_list:
        save_to_csv(data_list, csv_file)
        print(f"Data from {len(data_list)} XML files saved to '{csv_file}' successfully.")
else:
    print("No valid data found in XML files.")
```

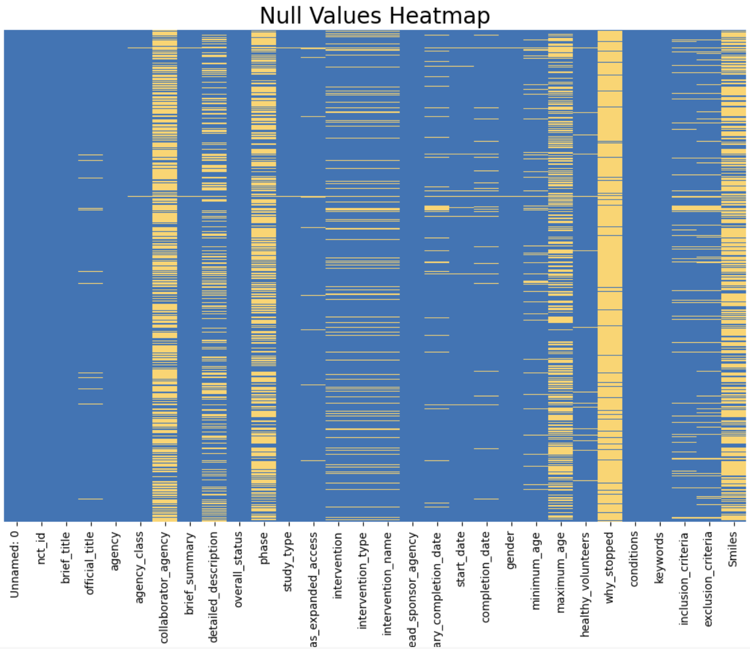
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

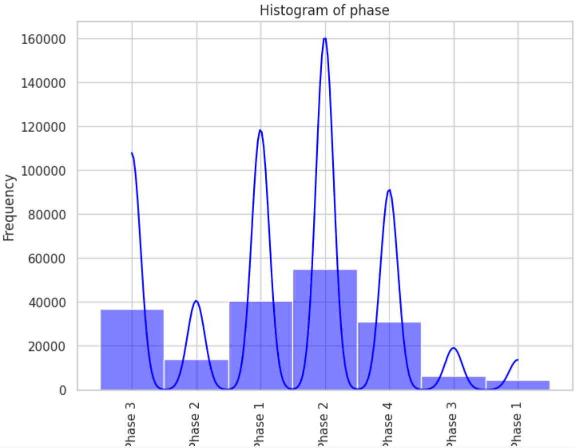
```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/Mtech /Dissertation/data_output/combine_output_with_smiles.csv')
```

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

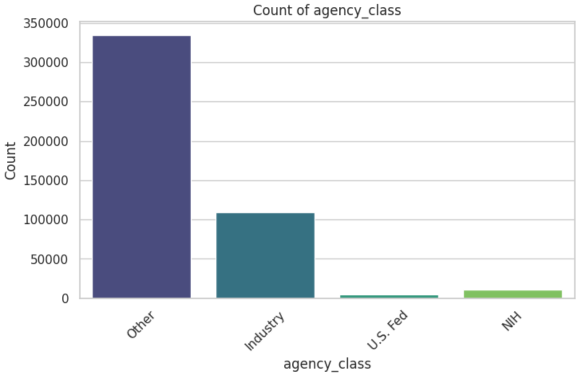
```
# Assuming you have loaded your data into a pandas DataFrame called 'df'
# If not, load your data into 'df' using pandas read_csv or any other method.

# Create the null value DataFrame (True for null, False for non-null)
null_df = df.isnull()

# Set a custom color palette for the heatmap (diverging color map with blue and yellow)
colors = ['#4374B3', '#F9D574']

# Create the heatmap using seaborn
plt.figure(figsize=(12, 8))
sns.heatmap(null_df, cmap=sns.color_palette(colors), cbar=False, yticklabels=False)

# Add a title and labels to the heatmap
plt.title('Null Values Heatmap', fontsize=20)
plt.xlabel('Columns', fontsize=14)

# Show the plot
plt.show()

```

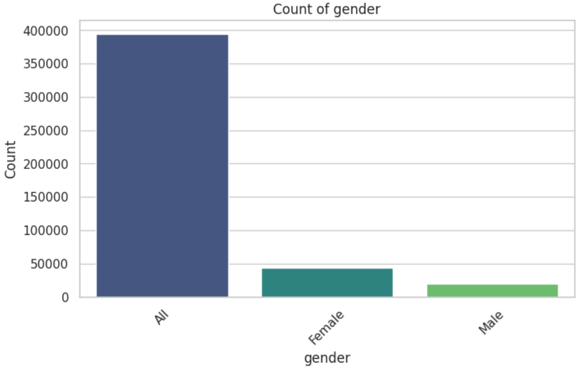


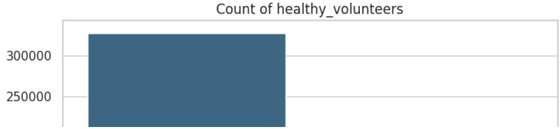
```
sns.set(style="whitegrid")

# Data Visualization
# Histograms for Numerical Columns
numerical_cols = ['phase']#, 'minimum_age', 'maximum_age']
for col in numerical_cols:
    plt.figure(figsize=(8, 6))
    sns.histplot(df[col].dropna(), kde=True, color='blue', bins=30)
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.xticks(rotation=90)
    plt.show()
```

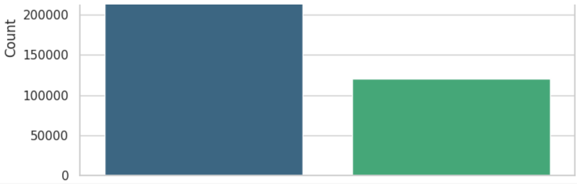


```
categorical_cols = ['agency_class', 'gender', 'healthy_volunteers']

num_plots = len(categorical_cols)
num_rows = num_plots
num_cols = 1

fig, axes = plt.subplots(num_rows, num_cols, figsize=(8, 6*num_rows))

# Plot settings
sns.set_palette("viridis")

for i, col in enumerate(categorical_cols):
    sns.countplot(data=df, x=col, palette='viridis', ax=axes[i])
    axes[i].set_title(f'Count of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')
    axes[i].tick_params(axis='x', rotation=45)

plt.subplots_adjust(hspace=0.5)  # Adjust the vertical spacing between subplots

plt.show()
```


Count of agency_class
Count
350000
300000
250000
200000
150000
100000
50000
0
Other
Industry
U.S. Fed
NIH
agency_class

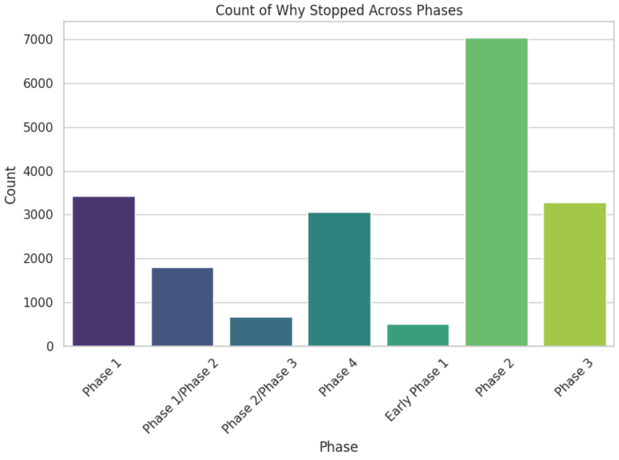

Count of gender
Count
400000
350000
300000
250000
200000
150000
100000
50000
0
All
Female
Male
gender


Count of healthy_volunteers
300000
250000



```python
categorical_cols = ['overall_status', 'study_type']

num_plots = len(categorical_cols)
num_rows = 1
num_cols = num_plots

fig, axes = plt.subplots(num_rows, num_cols, figsize=(12,6))

# Plot settings
sns.set_palette("viridis")

for i, col in enumerate(categorical_cols):
    sns.countplot(data=df, x=col, palette='viridis', ax=axes[i])
    axes[i].set_title(f'Count of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')
    axes[i].tick_params(axis='x', rotation=90)

plt.tight_layout()  # Automatically adjust spacing between subplots

plt.show()
```

```python
plt.figure(figsize=(8, 6))
sns.countplot(data=df[df['why_stopped'].notnull()], x='phase', palette='viridis')
plt.title('Count of Why Stopped Across Phases')
plt.xlabel('Phase')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```python
# Text Data Analysis (if relevant, for text columns)
# Example: Word Cloud for 'brief_summary'
# from wordcloud import WordCloud

# text_column = 'why_stopped'
# text_data = " ".join(text for text in df[text_column].dropna())
# wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text_data)
# plt.figure(figsize=(10, 6))
# plt.imshow(wordcloud, interpolation='bilinear')
# plt.title(f'Word Cloud for {text_column}')
# plt.axis('off')
# plt.show()

```

✓ 1s completed at 4:45 PM ● ✕