```
pip install wordcloud matplotlib numpy pandas nltk plotly seaborn
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-packages (1.9.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (5.24.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.12/dist-packages (from wordcloud) (11.3.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.59.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-packages (from plotly) (8.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matpl
```

```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.12/dist-packages (from wordcloud) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.12/dist-packages (from wordcloud) (11.3.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordclo
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcloud)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcl
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordcl
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordclou
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wordclo
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib->wor
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matpl
```

```
from datasets import load_dataset
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import nltk, re
from nltk.corpus import stopwords
from collections import Counter
import numpy as np
```

```
ds = load_dataset("keivalya/MedQuad-MedicalQnADataset")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning: 
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/toke
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
```

```
README.md: 100%                                          233/233 [00:00<00:00, 21.8kB/s]

medDataset_processed.csv: 100%                                     22.5M/22.5M [00:00<00:00, 74.4MB/s]

Generating train split: 100%                                   16407/16407 [00:00<00:00, 25017.11 examples/s]
```

```
ds
```

```
DatasetDict({
    train: Dataset({
        features: ['qtype', 'Question', 'Answer'],
        num_rows: 16407
    })
})
```

```
train_data = ds["train"]

# Convert to DataFrame for EDA
df = train_data.to_pandas()

# Show schema and sample rows
print(df.head())
print("\nColumns:", df.columns)
print("\nNumber of samples:", len(df))
```

```
             qtype                                           Question  \
0   susceptibility  Who is at risk for Lymphocytic Choriomeningiti...   
1         symptoms  What are the symptoms of Lymphocytic Choriomen...   
2   susceptibility  Who is at risk for Lymphocytic Choriomeningiti...   
3  exams and tests  How to diagnose Lymphocytic Choriomeningitis (...   
4        treatment  What are the treatments for Lymphocytic Chorio...   

                                              Answer  
0  LCMV infections can occur after exposure to fr...  
1  LCMV is most commonly recognized as causing ne...  
2  Individuals of all ages who come into contact ...  
3  During the first phase of the disease, the mos...  
4  Aseptic meningitis, encephalitis, or meningoen...  

Columns: Index(['qtype', 'Question', 'Answer'], dtype='object')

Number of samples: 16407
```

```
print("Number of samples:", len(df))
print("Columns:", df.columns.tolist())

# Peek at a few examples
for i in range(3):
    print(f"\nQTYPE: {df['qtype'][i]}")
    print(f"QUESTION: {df['Question'][i]}")
    print(f"ANSWER: {df['Answer'][i][:200]}...")  # preview 200 chars
```

```
Number of samples: 16407
Columns: ['qtype', 'Question', 'Answer']

QTYPE: susceptibility
QUESTION: Who is at risk for Lymphocytic Choriomeningitis (LCM)? ?
ANSWER: LCMV infections can occur after exposure to fresh urine, droppings, saliva, or nesting materials from infect
```

```
QTYPE: symptoms
QUESTION: What are the symptoms of Lymphocytic Choriomeningitis (LCM) ?
ANSWER: LCMV is most commonly recognized as causing neurological disease, as its name implies, though infection with
           ...

QTYPE: susceptibility
QUESTION: Who is at risk for Lymphocytic Choriomeningitis (LCM)? ?
ANSWER: Individuals of all ages who come into contact with urine, feces, saliva, or blood of wild mice are potential
```

```
df["q_len"] = df["Question"].apply(lambda x: len(str(x).split()))
df["a_len"] = df["Answer"].apply(lambda x: len(str(x).split()))

print("Avg question length:", df["q_len"].mean())
print("Avg answer length:", df["a_len"].mean())
```

```
Avg question length: 8.212165539099164
Avg answer length: 201.35436094349973
```

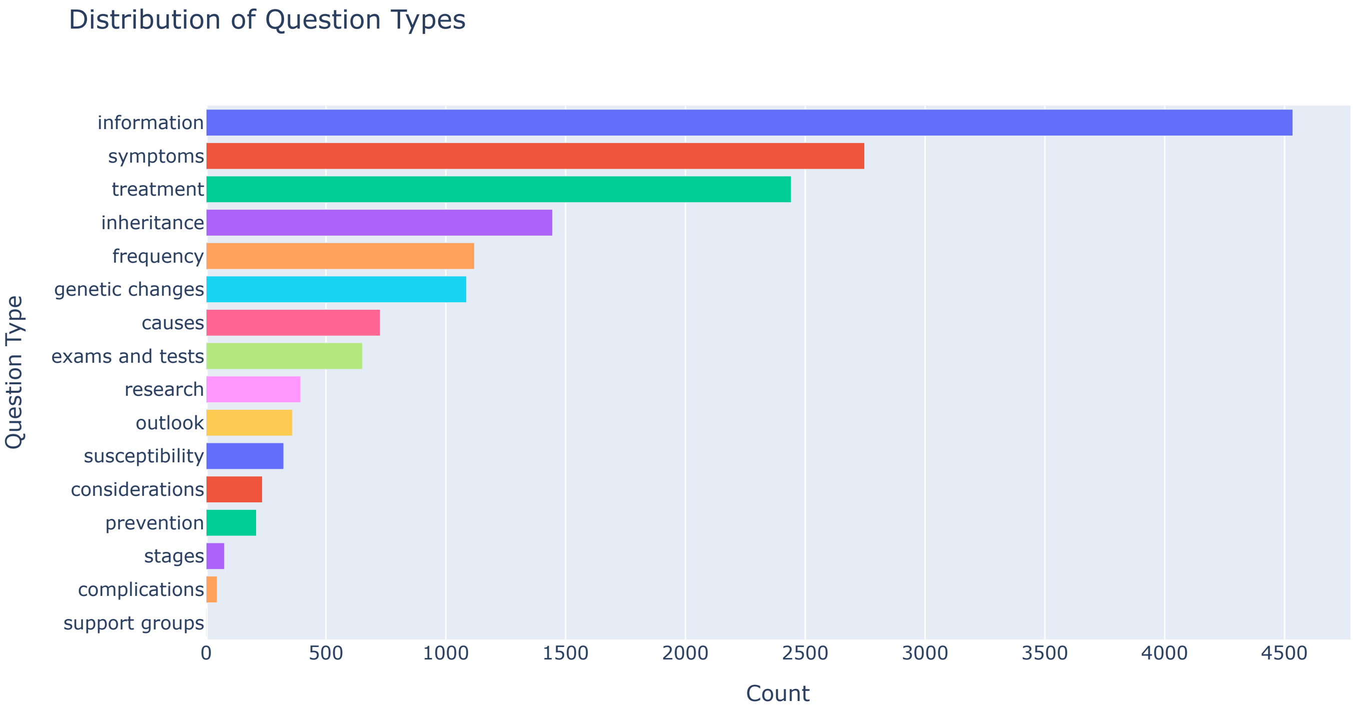
USING PLOTLY FOR INTERACTIVE VISUALIZATIONS

```
import plotly.express as px
import plotly.graph_objects as go
```

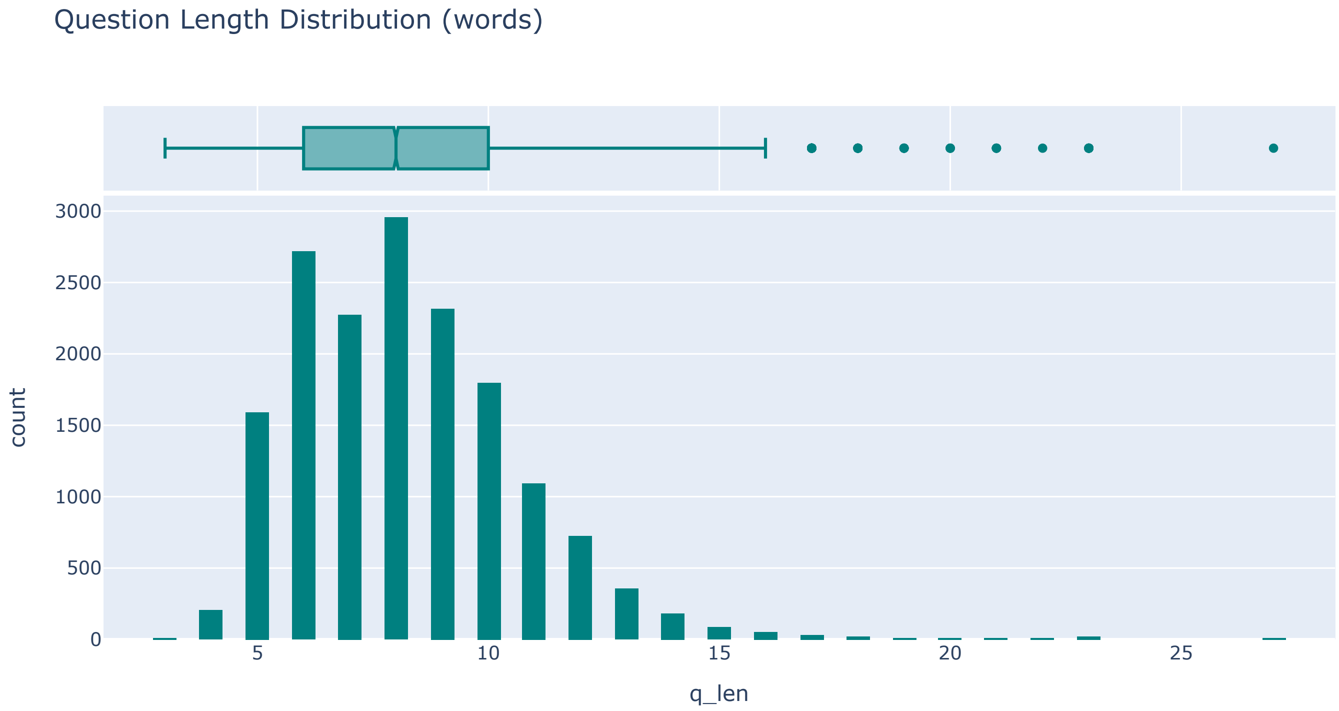
```
pip install --upgrade nbformat
```

```
Requirement already satisfied: nbformat in /usr/local/lib/python3.12/dist-packages (5.10.4)
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.12/dist-packages (from nbformat) (2.21
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.12/dist-packages (from nbformat) (4.25.1)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /usr/local/lib/python3.12/dist-packages (from nbformat)
Requirement already satisfied: traitlets>=5.1 in /usr/local/lib/python3.12/dist-packages (from nbformat) (5.7.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6->nbfor
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6->nbfo
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.12/dist-packages (from jupyter-core!=5.0.
Requirement already satisfied: typing-extensions>=4.4.0 in /usr/local/lib/python3.12/dist-packages (from referencing
```

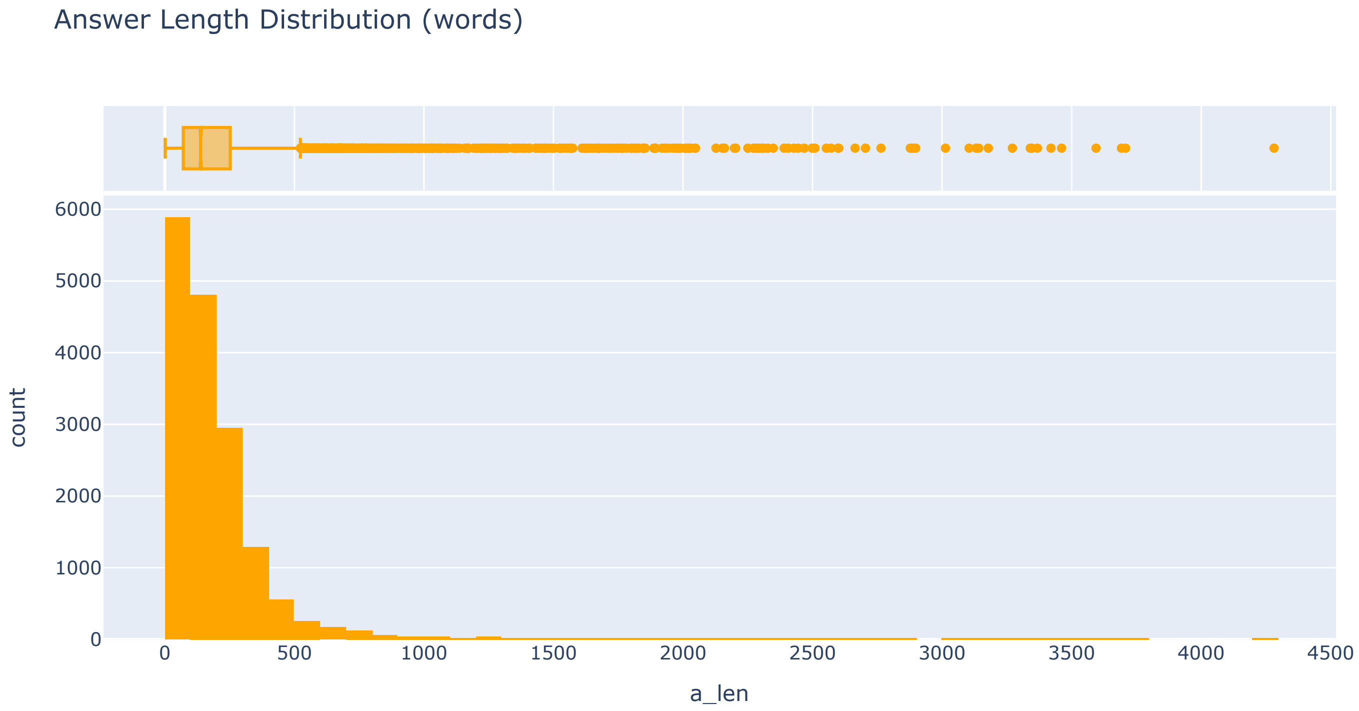
```
fig = px.bar(
    df["qtype"].value_counts().reset_index(),
    x="count",
    y="qtype",
    orientation="h",
    color="qtype",
    title="Distribution of Question Types",
    labels={"qtype":"Question Type", "count":"Count"}
)
fig.update_layout(showlegend=False)
fig.show()
```



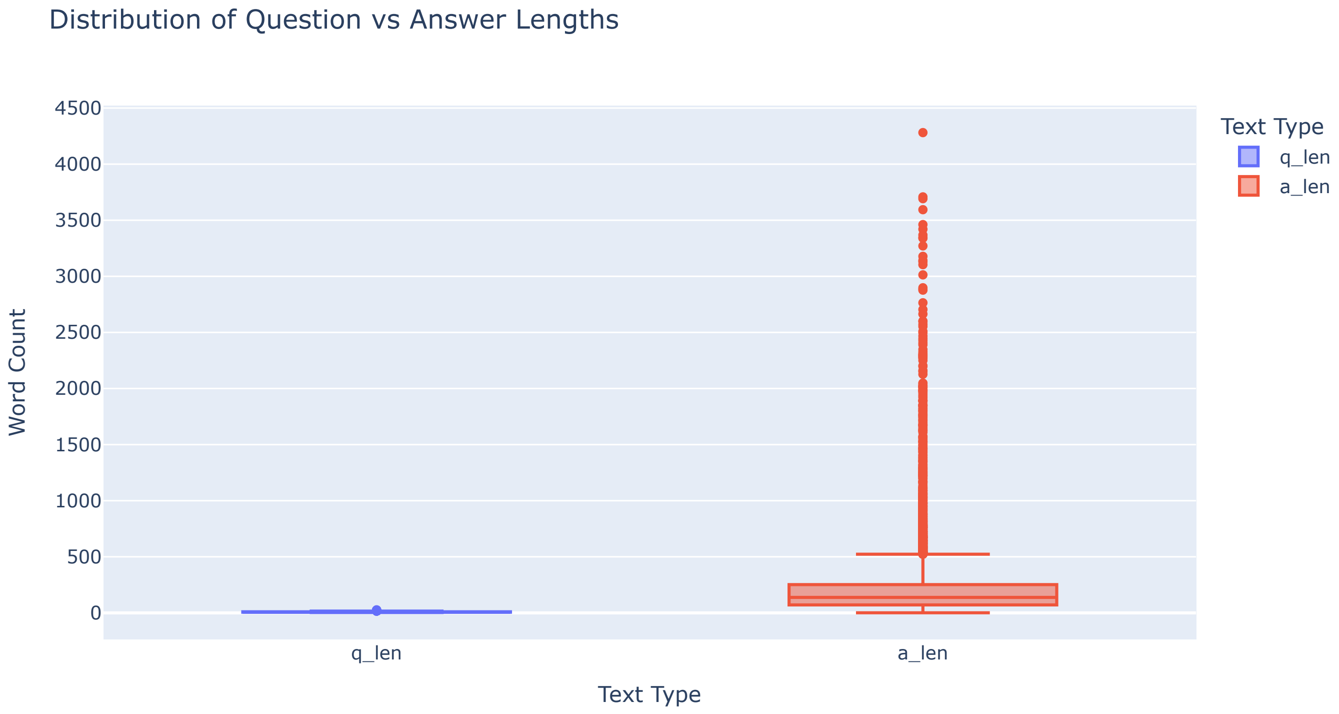
```
fig = px.histogram(
    df,
    x="q_len",
    nbins=50,
    title="Question Length Distribution (words)",
    marginal="box", # adds a boxplot on top
    color_discrete_sequence=["teal"]
)
fig.show()
```



```
fig = px.histogram(
    df,
    x="a_len",
    nbins=50,
    title="Answer Length Distribution (words)",
    marginal="box",
    color_discrete_sequence=["orange"]
)
fig.show()
```

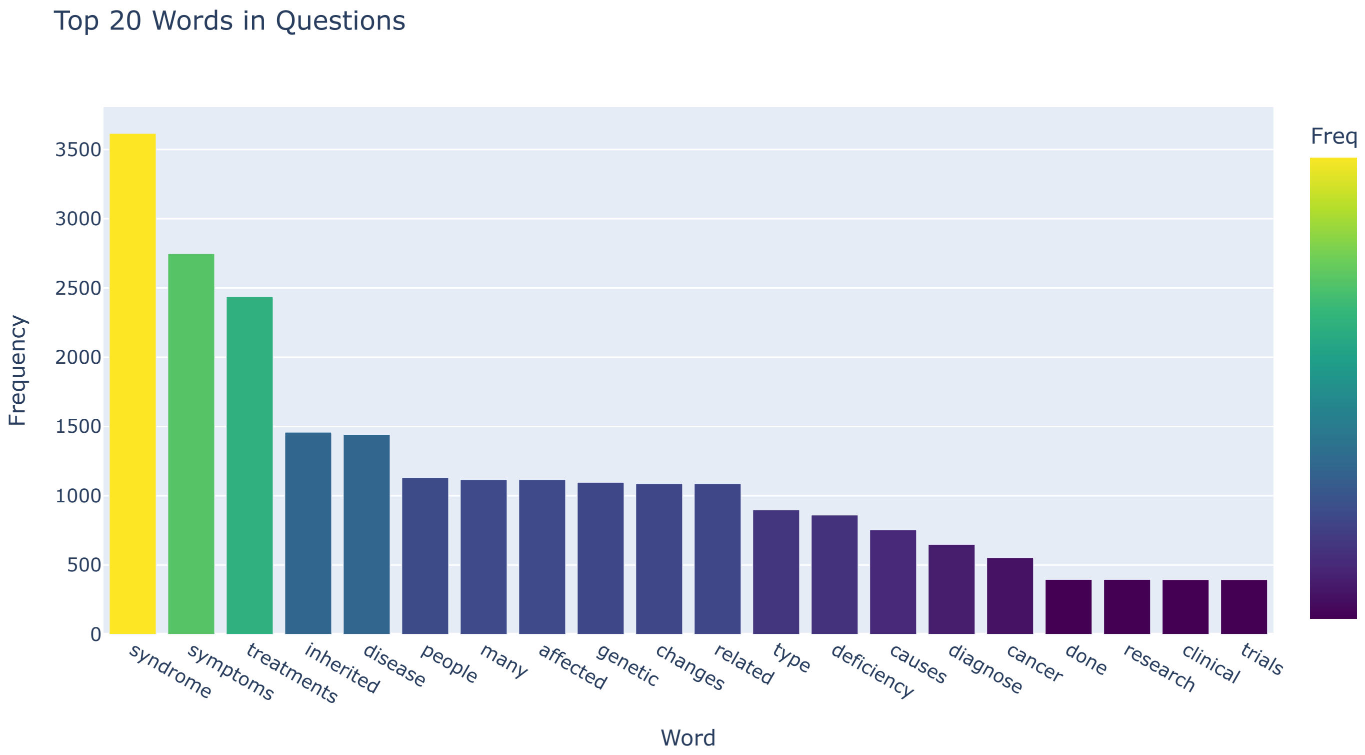


```
melted = df.melt(value_vars=["q_len","a_len"], var_name="Text Type", value_name="Word Count")

fig = px.box(
    melted,
    x="Text Type",
    y="Word Count",
    color="Text Type",
    title="Distribution of Question vs Answer Lengths"
)
fig.show()
```

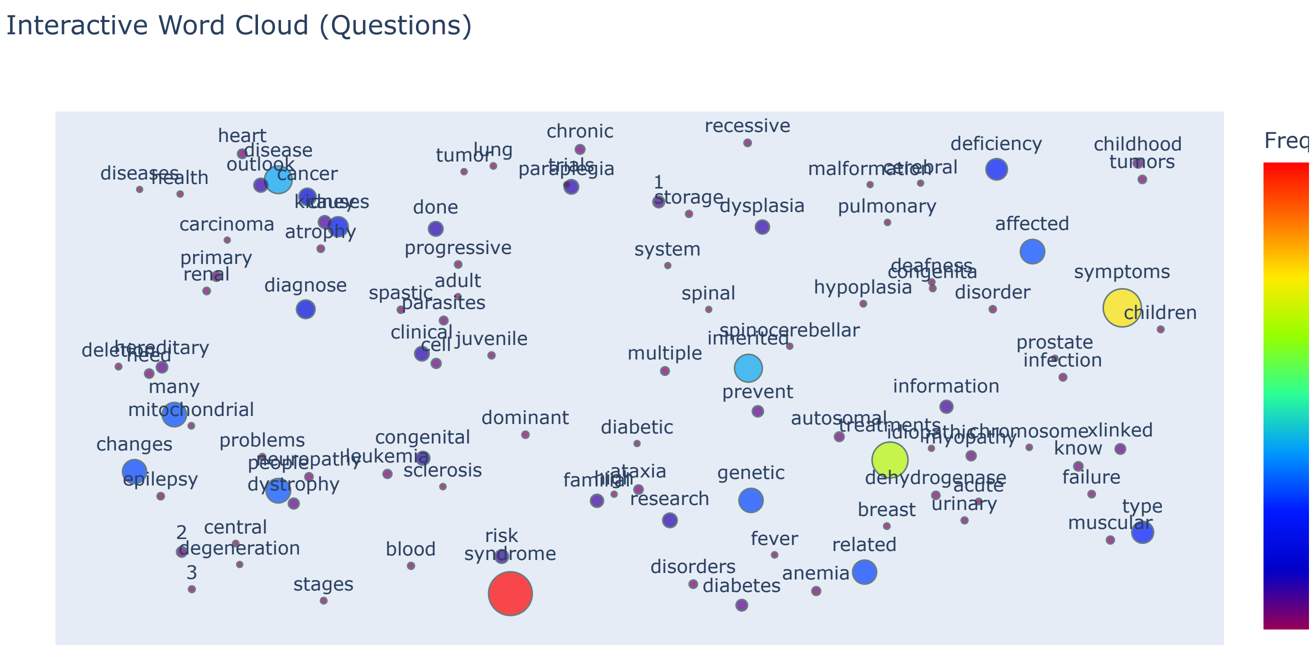


```
nltk.download('stopwords')
stop_words = set(stopwords.words("english"))

def clean_text(text):
    return re.sub(r"[^a-zA-Z0-9\s]", "", text.lower())

all_q_words = " ".join(df["Question"].apply(clean_text)).split()
filtered_q_words = [w for w in all_q_words if w not in stop_words]
top_q = Counter(filtered_q_words).most_common(20)

q_df = pd.DataFrame(top_q, columns=["Word","Frequency"])

fig = px.bar(
    q_df,
    x="Word",
    y="Frequency",
    title="Top 20 Words in Questions",
    color="Frequency",
    color_continuous_scale="viridis"
)
fig.show()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```


Top 20 Words in Questions

```
all_q_words = " ".join(df["Question"].apply(clean_text)).split()
filtered_q_words = [w for w in all_q_words if w not in stop_words]

# Frequency count
q_counter = Counter(filtered_q_words)
q_df = pd.DataFrame(q_counter.items(), columns=["Word", "Frequency"]).sort_values("Frequency", ascending=False)




# take top 100 words for visibility
top_q_df = q_df.head(100).copy()

# random positions for words
np.random.seed(42)
top_q_df["x"] = np.random.rand(len(top_q_df))
top_q_df["y"] = np.random.rand(len(top_q_df))

# scale word sizes by frequency
top_q_df["size"] = top_q_df["Frequency"] / top_q_df["Frequency"].max() * 50

fig = px.scatter(
    top_q_df,
    x="x",
    y="y",
    text="Word",
    size="size",
    color="Frequency",
    color_continuous_scale="rainbow",
    title="Interactive Word Cloud (Questions)"
)

fig.update_traces(textposition="top center", marker=dict(opacity=0.7, line=dict(width=1, color="DarkSlateGrey")))
fig.update_xaxes(visible=False)
fig.update_yaxes(visible=False)
fig.show()
fig.show()
```



Interactive Word Cloud (Questions)

```
# --- ANSWERS WORD CLOUD ---

# Combine all answers
all_a_words = " ".join(df["Answer"].apply(clean_text)).split()
filtered_a_words = [w for w in all_a_words if w not in stop_words]

# Frequency count
a_counter = Counter(filtered_a_words)
a_df = pd.DataFrame(a_counter.items(), columns=["Word", "Frequency"]).sort_values("Frequency", ascending=False)

# Take top 100 for visibility
top_a_df = a_df.head(100).copy()

# Random positions for cloud
np.random.seed(99)
top_a_df["x"] = np.random.rand(len(top_a_df))
top_a_df["y"] = np.random.rand(len(top_a_df))

# Scale sizes
top_a_df["size"] = top_a_df["Frequency"] / top_a_df["Frequency"].max() * 50

fig = px.scatter(
    top_a_df,
    x="x",
    y="y",
    text="Word",
    size="size",
    color="Frequency",
    color_continuous_scale="plasma",
    title="Interactive Word Cloud (Answers)"
```