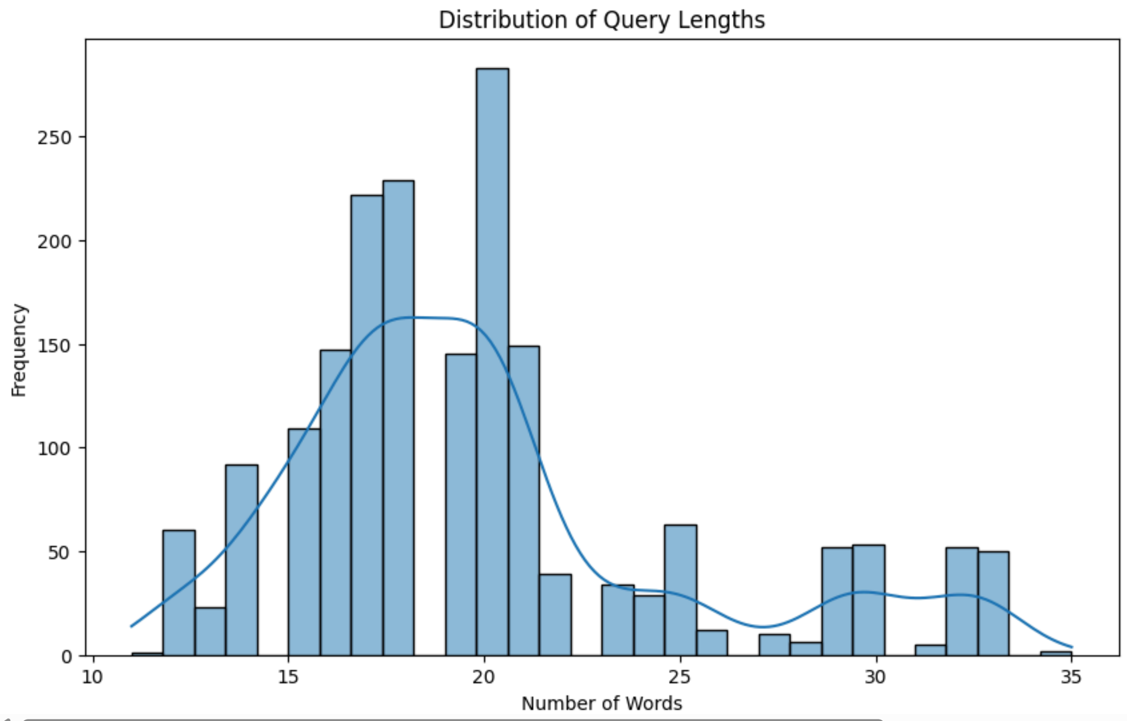
```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Load the CSV file into a DataFrame
file_path = '/content/dataset.csv'
df = pd.read_csv(file_path)

#Examine the structure of the DataFrame
print("Dataset Shape:", df.shape)
print("\nDataset Info:")
print(df.info())

#Check for null values in each column
print("\nNull Values in Each Column:")
print(df.isnull().sum())


#Visualize the distribution of query lengths
# Calculate the number of words in each query
df['query_length'] = df['Output'].apply(lambda x: len(str(x).split()))

plt.figure(figsize=(10, 6))
sns.histplot(df['query_length'], kde=True, bins=30)
plt.title("Distribution of Query Lengths")
plt.xlabel("Number of Words")
plt.ylabel("Frequency")
plt.show()
```

```
Dataset Shape: (1867, 3)

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1867 entries, 0 to 1866
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   User Query  1867 non-null   object
 1   Output      1867 non-null   object
 2   Annotation  1867 non-null   object
dtypes: object(3)
memory usage: 43.9+ KB
None

Null Values in Each Column:
User Query    0
Output        0
Annotation    0
dtype: int64
```



```python
import re
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('punkt_tab')

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split

# Define stopwords and lemmatizer
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Function to clean text
def clean_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove punctuation and special characters
    text = re.sub(r'[^a-z\s]', '', text)
    # Tokenize the text into words
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    # Lemmatize each token
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    # Rejoin tokens to form the cleaned text
    cleaned_text = ' '.join(tokens)
    return cleaned_text

# Apply the cleaning function to the 'User Query' column
df['cleaned_query'] = df['User Query'].apply(clean_text)

# clean the 'Output' column too
df['cleaned_output'] = df['Output'].apply(lambda x: x.lower().strip())

# Data Splitting: 80% for training and 20% for testing
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

# Display basic information about the splits
print("Training set shape:", train_df.shape)
print("Testing set shape:", test_df.shape)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
Training set shape: (1493, 6)
Testing set shape: (374, 6)
```

```
# Install spaCy with transformer support
!pip install -U spacy[transformers]

# Download the transformer-based English model
!python -m spacy download en_core_web_trf
```

```
Downloading curated_transformers-0.1.1-py2.py3-none-any.whl (25 kB)
Installing collected packages: curated-tokenizers, curated-transformers, spacy-curated-transformers, en-core-web-trf
Successfully installed curated-tokenizers-0.0.9 curated-transformers-0.1.1 en-core-web-trf-3.8.0 spacy-curated-transformers-0.3.0
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_trf')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

```python
import spacy

# Load spaCy's transformer-based English model
nlp = spacy.load("en_core_web_trf")

# Process a sample query
sample_query = df['Output'].iloc[0]
doc = nlp(sample_query)

print("Tokenization & POS Tagging:")
for token in doc:
    print(f"{token.text:12s}  {token.pos_:10s}  {token.tag_}")
```

```
Tokenization & POS Tagging:
For           ADP         IN
beginners     NOUN        NNS
,             PUNCT       ,
start         VERB        VB
with          ADP         IN
20-30         NUM         CD
minutes       NOUN        NNS
of            ADP         IN
moderate      ADJ         JJ
cardio        NOUN        NN
like          ADP         IN
brisk         ADJ         JJ
walking       NOUN        NN
,             PUNCT       ,
followed      VERB        VBN
by            ADP         IN
basic         ADJ         JJ
bodyweight    NOUN        NN
exercises     NOUN        NNS
such          ADJ         JJ
as            ADP         IN
squats        NOUN        NNS
,             PUNCT       ,
push          NOUN        NN
-             PUNCT       HYPH
ups           NOUN        NNS
,             PUNCT       ,
and           CCONJ       CC
planks        NOUN        NNS
.             PUNCT       .
```

```python
#Chunks and dependency relation

print("\nNoun Chunks in the query:")
for chunk in doc.noun_chunks:
    print(f" - {chunk.text}")

print("\nDependency Parsing:")
for token in doc:
    print(f"{token.text:12s} --> {token.dep_:10s} --> {token.head.text}")
```

```

Noun Chunks in the query:
 - beginners
 - 20-30 minutes
 - moderate cardio
 - brisk walking
 - basic bodyweight exercises
 - squats
 - push-ups
 - planks

Dependency Parsing:
For          --> prep       --> start
beginners    --> pobj       --> For
```

```
,            --> punct      --> start
start        --> ROOT       --> start
with         --> prep       --> start
20–30        --> nummod     --> minutes
minutes      --> pobj       --> with
of           --> prep       --> minutes
moderate     --> amod       --> cardio
cardio       --> pobj       --> of
like         --> prep       --> cardio
brisk        --> amod       --> walking
walking      --> pobj       --> like
,            --> punct      --> minutes
followed     --> acl        --> minutes
by           --> agent      --> followed
basic        --> amod       --> exercises
bodyweight   --> compound   --> exercises
exercises    --> pobj       --> by
such         --> amod       --> as
as           --> prep       --> exercises
squats       --> pobj       --> as
,            --> punct      --> squats
push         --> compound   --> ups
-            --> punct      --> ups
ups          --> conj       --> squats
,            --> punct      --> ups
and          --> cc         --> ups
planks       --> conj       --> ups
.            --> punct      --> start
```

```python
print("\nNamed Entities Found in the Query:")
if doc.ents:
    for ent in doc.ents:
        print(f"{ent.text:12s} ({ent.label_})")
else:
    print("No entities found in the sample query.")
```

```

Named Entities Found in the Query:
20–30 minutes (TIME)
```

```python
# Install sentence-transformers
!pip install sentence-transformers
```

```
Requirement already satisfied: sentence-transformers in /usr/local/lib/python3.11/dist-packages (3.4.1)
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.49
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (2.6.0+cu124)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.14.1)
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (0.29.3)
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (11.1.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transformers)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-trans
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transf
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transforme
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-transformers)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sent
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->senten
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-tran
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-tra
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-tran
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-t
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-t
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-tr
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transfo
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-trans
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3.2
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (1.1
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch>=1.11.0->sentenc
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->sentence-transf
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->sentence-
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->sent
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->sentence
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->sentence-transformers) (1.4.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->sentence-transformers
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch>=1.11.0->sentence-transfor
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20.0->sentence
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20.0->se
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20.0->se
```

```python
from sentence_transformers import SentenceTransformer

# Load the Sentence-BERT model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Generate an embedding for the sample query
embedding = model.encode(sample_query)
print("\nSentence-BERT Embedding for the sample query:")
print(embedding)
```

```
/usr/local/lib/python3.11/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/tokens), set it as secre
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(
modules.json: 100%                                    349/349 [00:00<00:00, 5.82kB/s]
config_sentence_transformers.json: 100%               116/116 [00:00<00:00, 2.67kB/s]
README.md: 100%                                       10.5k/10.5k [00:00<00:00, 275kB/s]
sentence_bert_config.json: 100%                       53.0/53.0 [00:00<00:00, 1.16kB/s]
config.json: 100%                                     612/612 [00:00<00:00, 13.9kB/s]
model.safetensors: 100%                               90.9M/90.9M [00:00<00:00, 161MB/s]
tokenizer_config.json: 100%                           350/350 [00:00<00:00, 29.4kB/s]
vocab.txt: 100%                                       232k/232k [00:00<00:00, 12.7MB/s]
tokenizer.json: 100%                                  466k/466k [00:00<00:00, 35.0MB/s]
special_tokens_map.json: 100%                         112/112 [00:00<00:00, 8.57kB/s]
config.json: 100%                                     190/190 [00:00<00:00, 12.6kB/s]

Sentence-BERT Embedding for the sample query:
[ 1.84835214e-02 -2.74094529e-02 -1.79916788e-02  3.86158749e-02
 -1.06250755e-01 -4.64252979e-02 -5.44205494e-02 -1.48432972e-02
 -6.06881306e-02 -1.77154876e-02  8.29189643e-03  3.94451171e-02
 -1.73490569e-02  3.26199532e-02  1.39670307e-03  1.48688937e-02
  5.40559925e-02  2.02019494e-02  2.00313479e-02  2.78611761e-02
  1.95406601e-02 -7.26020522e-03  6.89539462e-02 -6.83693122e-03
 -1.27489287e-02 -4.96626785e-03  4.00171690e-02 -2.58707386e-02
  4.02860641e-02  2.22228765e-02  2.96049491e-02 -5.91720231e-02
  8.77704844e-02  3.29544991e-02 -7.88048729e-02  2.60647088e-02
  1.26677185e-01 -5.38186282e-02 -1.01688221e-01  4.09076037e-03
  2.67346203e-02  9.47994832e-03 -3.83434794e-03  6.31621201e-03
  4.16448042e-02  4.54255417e-02  4.76548485e-02 -2.63007004e-02
  4.30551320e-02  3.83962765e-02  3.57637033e-02 -3.29230428e-02
  1.00586126e-02 -1.43079972e-02  3.70694511e-02  2.65719723e-02
 -4.53203321e-02 -3.32907736e-02  8.20846204e-03 -7.44366199e-02
  1.48739768e-02  2.09486168e-02 -1.55283418e-02  7.26637489e-04
 -8.43390301e-02 -6.77855164e-02  2.23249458e-02  4.61693387e-03
  6.84070885e-02  2.59258635e-02 -1.26471724e-02  7.77111331e-04
 -5.07284552e-02 -1.50081078e-02 -9.43713188e-03  1.57554895e-02
 -1.23548955e-02  6.82844147e-02 -2.10242439e-03 -3.30931246e-02
 -4.75708097e-02 -1.12865362e-02  8.23356025e-03  5.62238507e-02
 -3.67332734e-02  6.23321347e-02 -1.84719581e-02  1.53332114e-01
 -1.11013157e-02 -1.45017421e-02  1.28870541e-02  3.83995362e-02
 -1.06975436e-01  9.29350965e-03 -6.47428110e-02 -1.25122108e-02
 -3.37822847e-02 -2.62645446e-02 -9.60252248e-03  1.06358575e-02
  8.15246031e-02 -1.69914644e-02  1.29946902e-01  6.10867739e-02
 -3.58257890e-02 -8.60659406e-02  5.48108108e-02  1.77901201e-02
  1.66356917e-02  9.60460603e-02  3.09957881e-02 -6.00621477e-02
 -1.63449824e-03  1.25699844e-02  5.69183454e-02  7.10281506e-02
 -2.43924186e-02  1.31562147e-02  2.24198843e-03  7.67555237e-02
 -5.19442558e-02 -5.71673252e-02  4.62734792e-03 -7.18659759e-02
 -7.97729790e-02  3.90722938e-02  1.36005329e-02  2.85357315e-33
 -1.62687209e-02  4.86304164e-02  4.12776656e-02  4.61899154e-02
 -1.99135281e-02 -1.19891658e-01  7.27647636e-03 -7.69580379e-02
  6.36012405e-02  2.58996654e-02  5.37260137e-02  1.66357700e-02
  3.65951061e-02  4.26868014e-02  7.67969899e-03 -6.00061081e-02
 -1.33408131e-02 -4.48250249e-02  3.63883711e-02 -7.94426538e-03
  6.09828383e-02 -9.28344205e-02  7.08889565e-04 -2.55090352e-02
  5.43945543e-02 -3.14430743e-02 -3.93154398e-02 -2.83255074e-02
  1.46430638e-02 -1.92074291e-02 -1.74178742e-02  3.07756534e-04
 -5.67141250e-02 -5.92200318e-03  2.30634008e-02  6.63976138e-03
  1.11974232e-01  6.66338727e-02  5.85148148e-02 -3.03268954e-02
  4.63998467e-02  2.57531703e-02  2.81245876e-02 -8.65470693e-02
  1.06390655e-01  6.23061182e-03  5.60944751e-02  7.36116245e-03
 -3.14857624e-03  1.24273226e-02  1.64878666e-02 -6.17097802e-02
  4.67638373e-02 -5.22827432e-02 -1.50504513e-02  5.19814380e-02
 -1.17266709e-02  1.67018287e-02 -7.67394379e-02  4.76369411e-02
  3.50129381e-02  6.38314858e-02 -9.94396303e-03  3.89827602e-02
 -1.08212702e-01  5.39657176e-02 -2.08003428e-02  5.42124771e-02
 -1.48485927e-02  5.82120242e-03 -1.56542659e-02  3.17977965e-02
  3.56274284e-02 -9.18340459e-02  8.40264410e-02 -1.93925165e-02
 -7.62497913e-03 -6.04701638e-02 -1.08786643e-01  1.65495351e-02
  1.28396899e-02  4.03430350e-02 -5.96752726e-02  6.12376630e-03
 -7.98955280e-03  8.94102734e-03 -1.59731209e-02  3.95398587e-02
 -5.23984879e-02 -1.05241109e-02 -6.93212673e-02  6.86114701e-03
 -1.16888463e-04  5.30087203e-02 -7.45742628e-03 -2.81861615e-33
```

```
  9.17975008e-02 -1.09253712e-02  8.62475410e-02 -1.15235476e-02
  3.08466386e-02 -3.33491676e-02  5.02797542e-03  1.02462411e-01
  8.13557282e-02 -4.89437804e-02 -2.84789130e-02 -5.76464646e-02
 -3.54441814e-03 -4.62382957e-02  3.12891565e-02  3.75519395e-02
  1.94101743e-02  3.44083123e-02  3.51605229e-02  1.89154595e-02
  1.23867497e-01 -4.57655601e-02 -1.01837680e-01 -7.40435347e-02
 -2.26007625e-02  3.22277397e-02  2.14767233e-02  9.34859551e-03
  1.89310312e-02  1.38868287e-03  5.05960314e-03 -3.45776863e-02
  6.11590706e-02 -4.96250279e-02 -1.29227161e-01  3.30411717e-02
 -8.57629701e-02  1.99739665e-01  6.56093061e-02  1.56675633e-02
  1.30372923e-02 -3.52377482e-02 -3.52281407e-02 -6.89006317e-03
 -2.76362225e-02 -1.12612091e-01  9.49432887e-03 -5.46875112e-02
 -1.13156974e-01 -9.19804648e-02 -2.01718379e-02 -6.14775270e-02
  7.44989002e-03 -5.13269864e-02  5.61711900e-02 -5.48313037e-02
 -1.86018758e-02 -1.29944170e-02  1.89890750e-02 -7.75679946e-02
 -5.87603077e-02  4.98402789e-02 -7.15531334e-02  3.50076519e-02
 -7.15085911e-03  5.13388813e-02 -5.19783869e-02  3.51954997e-02
 -4.40392084e-02  5.27255908e-02 -1.50628999e-01  6.60738274e-02
  4.64917906e-03  2.10769977e-02 -9.55304410e-03 -5.11525497e-02
  3.06484960e-02 -4.47820574e-02 -1.69125777e-02 -6.17643893e-02
  2.95035262e-02 -5.48991114e-02 -3.73484455e-02 -6.89430535e-02
  4.95837908e-03  6.94564581e-02 -4.60608229e-02 -8.08791742e-02
 -1.21860076e-02 -9.17449407e-03  1.88278891e-02  4.48427796e-02
  9.01843831e-02  6.08272217e-02  1.33690918e-02 -3.04017576e-08
 -3.36875655e-02  2.48009171e-02 -1.62181724e-02  7.97163248e-02
  8.32510740e-03  9.61110070e-02  8.70615337e-03  6.53292686e-02
 -7.57048186e-03 -3.54053043e-02  6.05215020e-02  7.29734227e-02
  1.30295560e-01  5.20841591e-03 -7.73429498e-02  4.79719136e-04
  4.98865033e-03  1.83933089e-03 -6.82095718e-03 -5.96088078e-03
  2.93115899e-02 -6.25014529e-02  6.26943773e-03 -2.71424484e-02
  2.45521311e-02 -1.18923455e-01  4.73078787e-02 -5.08301668e-02
 -1.49220822e-03  2.07754243e-02  3.25640589e-02 -1.06151709e-02
 -7.43527785e-02  8.39679316e-02  2.90378015e-02  2.65081506e-02
  8.80952831e-03  8.76302598e-04 -8.74820873e-02  3.85403447e-02
 -3.91527936e-02  1.57182571e-02  1.29646985e-02 -5.80235533e-02
 -4.70717736e-02 -3.97947915e-02 -9.89454985e-02 -4.00956720e-02
  5.12351841e-02 -1.52886603e-02  9.77758169e-02  7.12869223e-03
 -1.85055267e-02 -4.11243774e-02 -2.39809486e-03  1.00764491e-01
 -3.01155429e-02  8.01074598e-03 -1.96525943e-03  5.37771061e-02
 -1.00924753e-01  1.27574289e-02 -6.42362908e-02  2.84757763e-02]
```

```python
# Encode all responses from the "cleaned_output" column into embedding vectors
responses = df['cleaned_output'].tolist()
response_embeddings = model.encode(responses)

print("Encoded", len(response_embeddings), "response embeddings.")
```

```
Encoded 1867 response embeddings.
```

```python
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

def retrieve_response(query, model, response_embeddings, responses):

    query_embedding = model.encode([query])

    # Compute cosine similarities between query embedding and all response embeddings
    cosine_scores = cosine_similarity(query_embedding, response_embeddings)

    # Get index of the best match
    best_match_idx = np.argmax(cosine_scores)
    best_response = responses[best_match_idx]

    return best_response

# Test the retrieval function with a sample query
sample_query = df['cleaned_query'].iloc[0]
print("Sample Query:", sample_query)
print("Retrieved Response:", retrieve_response(sample_query, model, response_embeddings, responses))
```

```
Sample Query: im new working suggest simple workout routine beginner
Retrieved Response: a beginner plan should focus on low-impact exercises, proper warm-up/cool-down routines, and gradual intensity incre
```

```python
#    Inference function for the chatbot that returns the best matching response for a given user query.
def respond(user_query):
    best_response = retrieve_response(user_query, model, response_embeddings, responses)
    return best_response

# Testing the inference function with a custom query
test_query = "I need a high intensity workout for my legs."
print("Test Query:", test_query)
print("Chatbot Response:", respond(test_query))
```

```
Test Query: I need a high intensity workout for my legs.
Chatbot Response: incorporate exercises like high knees, butt kicks, and leg swings to activate muscles and prepare your body for high-i
```

```
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

def evaluate_retrieval(test_df, model, response_embeddings, responses, k=3):
    mrr_total = 0
    recall_at_k_total = 0
    num_samples = len(test_df)

    # Loop over each test sample
    for idx, row in test_df.iterrows():
        query = row['cleaned_query']
        ground_truth = row['cleaned_output']

        # Compute the embedding for the test query
        query_embedding = model.encode([query])

        # Calculate cosine similarity between the query and all pre-computed response embeddings
        cosine_scores = cosine_similarity(query_embedding, response_embeddings)[0]

        # Rank the responses (indices) by similarity in descending order
        ranked_indices = np.argsort(cosine_scores)[::-1]

        # Find the rank position of the ground truth response
        rank = None
        for i, idx in enumerate(ranked_indices):
            # Simple exact string match after stripping any extra whitespace
            if responses[idx].strip() == ground_truth.strip():
                rank = i + 1  # Rank is 1-indexed
                break

        # If ground truth is not found, we consider the rank as worst-case (length + 1)
        if rank is None:
            rank = len(ranked_indices) + 1

        # Update Mean Reciprocal Rank
        mrr_total += 1 / rank

        # Update Recall@k: if the ground truth is within the top k responses
        if rank <= k:
            recall_at_k_total += 1

    mrr = mrr_total / num_samples
    recall_at_k = recall_at_k_total / num_samples

    return mrr, recall_at_k

# Evaluate the retrieval performance on the test set (with k=3 for Recall@3)
mrr, recall_at_3 = evaluate_retrieval(test_df, model, response_embeddings, responses, k=3)
print("Mean Reciprocal Rank (MRR):", mrr)
print("Recall@3:", recall_at_3)
```

```
Mean Reciprocal Rank (MRR): 0.5070786714669944
Recall@3: 0.5481283422459893
```

rmv

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.23.3-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl.metadata (9.7 kB)
```

```
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.8.0 (from gradio)
  Downloading gradio_client-1.8.0-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.29.3)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.16)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.1.0)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.0)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.2-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.1-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.2)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.0)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.0-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.8.0->gradio) (2025.3.0)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.8.0->gradio)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.1.31)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67.1
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2.8
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (
Requirement already satisfied: pydantic-core==2.33.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio) (2
```

```python
import gradio as gr
import pandas as pd
import datetime

# Updated knowledge base with detailed recommendations
exercise_db = pd.DataFrame({
    "exercise_name": ["Push-ups", "Squats", "Deadlifts", "Plank", "Jump Rope"],
    "target_muscle": ["Chest, Triceps", "Legs, Glutes", "Back, Legs", "Core", "Cardio"],
    "difficulty_level": ["Beginner", "Beginner", "Advanced", "Beginner", "Intermediate"],
    "sets": [3, 3, 4, 3, "Timed"],
    "reps": [12, 15, 6, "Hold 30 sec", "60 sec"],
    "rest_time": ["30 sec", "30 sec", "60 sec", "N/A", "N/A"]
})

# Store conversation history
conversation_history = {}

# Store user queries for future improvements
query_logs = []

# Function to retrieve personalized workouts
def get_exercise_suggestions(fitness_level, equipment):
    filtered_exercises = exercise_db[
        (exercise_db["difficulty_level"] == fitness_level)
    ]
    if "None" in equipment:
        filtered_exercises = filtered_exercises
    exercise_details = filtered_exercises.to_dict(orient="records")
    return exercise_details

# Multi-Turn Memory
def respond(user_id, user_query, mode, fitness_level, age, equipment):
    """
    Handles normal and personalized queries, integrates multi-turn memory, and justifies responses.
    """
    cleaned_query = user_query.strip().lower()

    # Create session for user if not exists
    if user_id not in conversation_history:
        conversation_history[user_id] = []

    if mode == "Personalized Query":
        fitness_level = fitness_level if fitness_level else "Not specified"
        age = str(age) if age else "Not specified"
        equipment_str = ", ".join(equipment) if equipment else "None"

        enriched_query = (
            f"{cleaned_query} | fitness level: {fitness_level.lower()} | age: {age} | equipment: {equipment_str.lower()}"
        )
        final_query = enriched_query
    else:
        final_query = cleaned_query  # Normal query without personalization

    # Retrieve response using retrieval function
    best_response = retrieve_response(final_query, model, response_embeddings, responses)

    # Store conversation history
    conversation_history[user_id].append((user_query, best_response))

    # Log queries for analysis
    query_logs.append({
        "timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "user_id": user_id,
        "query": user_query,
        "response": best_response
    })

    # Suggest personalized exercises
    exercise_suggestions = get_exercise_suggestions(fitness_level, equipment)

    # Justification
    justification = (
        f"Since you are a {fitness_level.lower()} level trainee and selected {', '.join(equipment) or 'no equipment'}, "
        "these workouts are suitable for your level."
    )

    return best_response, conversation_history[user_id], exercise_suggestions, justification

# Feedback Handling Function
def collect_feedback(response, feedback):
    query_logs.append({"response": response, "feedback": feedback})
    return f"Feedback received: {feedback}"

# Toggle input fields based on mode
def toggle_inputs(mode):
    visible = mode == "Personalized Query"
```