```
import pandas as pd
```

```
df = pd.read_excel("/content/Diagnoses_list.xlsx")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 1 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Diagnoses_list  100 non-null    object
dtypes: object(1)
memory usage: 932.0+ bytes
```

```
df.head(10)
```

	Diagnoses_list
0	['Diabetes mellitus without mention of complic...
1	['Long-term (current) use of anticoagulants', ...
2	['Acute respiratory failure', 'Hypopotassemia'...
3	['Antineoplastic and immunosuppressive drugs c...
4	['Personal history of malignant neoplasm of to...
5	['Retention of urine, unspecified', 'Overflow ...
6	['Hyperlipidemia, unspecified', 'Do not resusc...
7	['Chronic airway obstruction, not elsewhere cl...
8	['Other specified bacterial infections in cond...
9	['Chronic kidney disease, stage 3 (moderate)',...

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
import ast

# Flattening and deduplicating all diagnoses
all_diagnoses = []
for row in df['Diagnoses_list']:
    all_diagnoses.extend(ast.literal_eval(row))
unique_diagnoses = sorted(set(all_diagnoses))
```

```
pd.DataFrame({'Diagnosis': unique_diagnoses}).to_csv('unique_diagnoses.csv', index=False)
```

```
diagnosis_df = pd.DataFrame({'Diagnosis': unique_diagnoses})
```

```
import icdcodex
```

```
from icdcodex import hierarchy
```

```
icd_10_cm_hierarchy, icd_10_cm_codes = hierarchy.icd10cm("2024")
```

```
icd10_list = [
    {"code": code, "desc": icd_10_cm_hierarchy.nodes[code].get("title", "")}
    for code in icd_10_cm_codes
]
```

```
icd_df = pd.DataFrame(icd10_list)
```

```
icd_df
```

	code	desc
0	A00.0	
1	A00.1	
2	A00.9	
3	A01.00	
4	A01.01	
...	...	...
74039	Z99.12	
74040	Z99.2	
74041	Z99.3	
74042	Z99.81	
74043	Z99.89	

74044 rows × 2 columns

Next steps: Generate code with icd_df | View recommended plots | New interactive sheet

```
import re
from tqdm import tqdm
from sentence_transformers import util


def preprocess_text(text):
    if pd.isna(text):
        return ""
    text = text.lower()
    text = re.sub(r'[^a-z0-9\s]', '', text)
    return text.strip()


icd_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74044 entries, 0 to 74043
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   code    74044 non-null  object
 1   desc    74044 non-null  object
dtypes: object(2)
memory usage: 1.1+ MB
```

```
diagnosis_df['Processed_Diagnosis'] = diagnosis_df['Diagnosis'].apply(preprocess_text)
icd_df['Processed_Description'] = icd_df['desc'].apply(preprocess_text)


from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np


model = SentenceTransformer('all-MiniLM-L6-v2')
```

```
/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, 38.3kB/s]
config_sentence_transformers.json: 100%                             116/116 [00:00<00:00, 10.3kB/s]
README.md:        10.5k/? [00:00<00:00, 645kB/s]
sentence_bert_config.json: 100%                                     53.0/53.0 [00:00<00:00, 4.99kB/s]
config.json: 100%                                                   612/612 [00:00<00:00, 65.9kB/s]
model.safetensors: 100%                                             90.9M/90.9M [00:01<00:00, 69.1MB/s]
tokenizer_config.json: 100%                                         350/350 [00:00<00:00, 23.5kB/s]
vocab.txt:        232k/? [00:00<00:00, 15.6MB/s]
tokenizer.json:        466k/? [00:00<00:00, 19.1MB/s]
special_tokens_map.json: 100%                                       112/112 [00:00<00:00, 11.0kB/s]
config.json: 100%                                                   190/190 [00:00<00:00, 15.2kB/s]
```

```
icd_descriptions = icd_df['desc'].tolist()
icd_embeddings = model.encode(icd_descriptions, convert_to_tensor=True)


def generate_justification(diagnosis, icd_desc, score):
    if score > 0.8:
        confidence = "high confidence"
    elif score > 0.6:
        confidence = "moderate confidence"
    else:
        confidence = "low confidence"
    return f"The diagnosis '{diagnosis}' was matched to ICD-10 description '{icd_desc}' with {confidence} based on semantic similarity."


def get_icd_matches(diagnosis_text, top_k=3):
    processed_text = preprocess_text(diagnosis_text)
    diag_embedding = model.encode(processed_text, convert_to_tensor=True)
    similarity_scores = util.pytorch_cos_sim(diag_embedding, icd_embeddings)[0]
    top_results = torch.topk(similarity_scores, k=top_k)

    results = []
    for score, idx in zip(top_results[0], top_results[1]):
      idx = int(idx)
      results.append({
          'ICD_Code': icd_df.iloc[idx]['code'],
          'ICD_Desc': icd_df.iloc[idx]['desc'],
          'Score': float(score)
    })
    return results


final_data = []


for idx, row in tqdm(diagnosis_df.iterrows(), total=len(diagnosis_df)):
    diag_text = row['Diagnosis']
    matches = get_icd_matches(diag_text, top_k=3)
    top_match = matches[0]
    justification = generate_justification(diag_text, top_match['ICD_Desc'], top_match['Score'])

    final_data.append({
        'Diagnosis': diag_text,
        'ICD-10 Code': top_match['ICD_Code'],
        'ICD Description': top_match['ICD_Desc'],
        'Similarity Score': top_match['Score'],
        'Justification': justification,
        'Alternative Suggestions': "; ".join([f"{m['ICD_Code']} - {m['ICD_Desc']}" for m in matches[1:]]),
        'Needs Review': top_match['Score'] < 0.6
    })
```

```
100%|██████████| 1388/1388 [00:13<00:00, 105.48it/s]
```

```
Start coding or generate with AI.
```

```
final_df = pd.DataFrame(final_data)
```

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1389 entries, 0 to 1388
Data columns (total 7 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Diagnosis                1389 non-null   object
 1   ICD-10 Code              1389 non-null   object
 2   ICD Description          1389 non-null   object
 3   Similarity Score         1389 non-null   float64
 4   Justification            1389 non-null   object
 5   Alternative Suggestions  1389 non-null   object
 6   Needs Review             1389 non-null   bool
dtypes: bool(1), float64(1), object(5)
memory usage: 66.6+ KB
```

```
final_df.to_csv("icd10_mapped_output.csv", index=False)
```

```
Start coding or generate with AI.
```