Unveiling the Power of Social Support in Healthcare

TEXAS

The University of Texas at Austin

Executive Summary: Unveiling the Power of Social Support in Healthcare

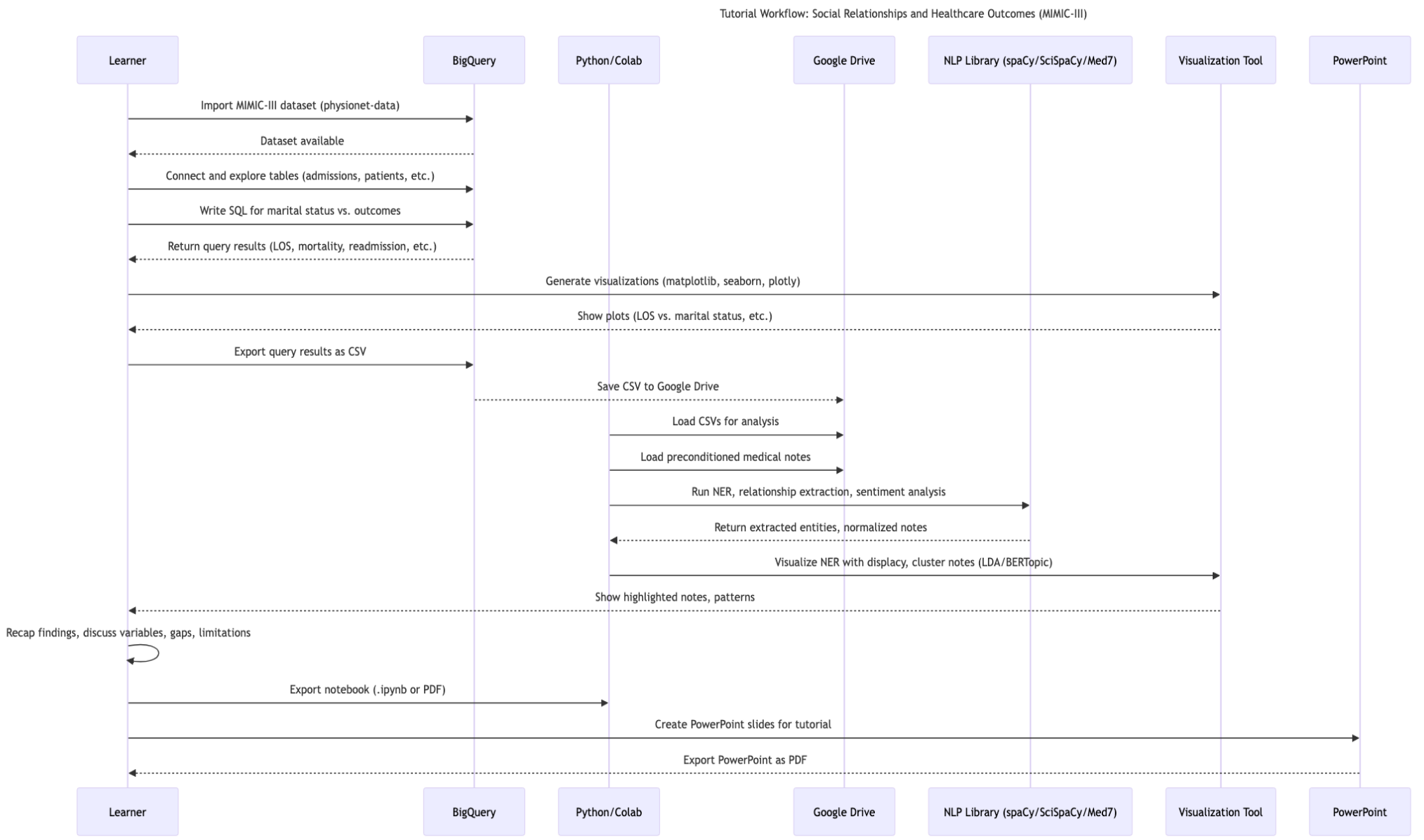
Our upcoming Self-Learning Tutorial (SLT) aims to shed light on the often-underestimated impact of social connections—specifically marriage and companionship—on patient health outcomes. This tutorial will provide a practical, step-by-step guide for anyone interested in exploring this vital link.

We'll start by accessing a vast, real-world medical database (MIMIC-III) stored in Google's cloud. From this extensive data, we'll carefully select and extract key information to build a compelling case for how social support influences health. This extracted data will then be prepared for in-depth analysis and compelling visual presentations, allowing us to easily see and understand important trends.

A significant part of the tutorial will involve using advanced text analysis tools to uncover crucial medical details hidden within large volumes of patient notes. This will help us pinpoint specific mentions related to social support and its context in healthcare records.

Ultimately, the SLT will synthesize all findings into clear conclusions, demonstrating the tangible benefits of social connections on health. The entire tutorial is designed to be highly hands-on, providing explicit instructions and code walk-throughs using various data analysis and text processing tools, ensuring that learners can easily follow along and grasp the concepts. Our goal is to empower individuals to understand and articulate the "unseen" yet critical role of social support in promoting better healthcare outcomes.

Synthesis and Workflow in Self-Learning Tutorial (SLT)



```
sequenceDiagram

title Tutorial Workflow: Social Relationships and Healthcare Outcomes (MIMIC-III)

    participant User as Learner
    participant BigQuery
    participant Python as Python/Colab
    participant Drive as Google Drive
    participant NLP as NLP Library (spaCy/SciSpaCy/Med7)
    participant Viz as Visualization Tool
    participant PPT as PowerPoint

        %% 1. Data Import
        User->>BigQuery: Import MIMIC-III dataset (physionet-data)
        BigQuery-->>User: Dataset available
        %% 2. Data Exploration & SQL Analysis
        User->>BigQuery: Connect and explore tables (admissions, patients, etc.)
        User->>BigQuery: Write SQL for marital status vs. outcomes
        BigQuery-->>User: Return query results (LOS, mortality, readmission, etc.)
        %% 3. Visualization of Findings
        User->>Viz: Generate visualizations (matplotlib, seaborn, plotly)
        Viz-->>User: Show plots (LOS vs. marital status, etc.)
        %% 4. Data Export & Python Analysis
        User->>BigQuery: Export query results as CSV
        BigQuery-->>Drive: Save CSV to Google Drive
        Python->>Drive: Load CSVs for analysis
        %% 5. NLP with Medical Notes
        Python->>Drive: Load preconditioned medical notes
        Python->>NLP: Run NER, relationship extraction, sentiment analysis
        NLP-->>Python: Return extracted entities, normalized notes
        %% 6. Highlight Key Notes
        Python->>Viz: Visualize NER with displacy, cluster notes (LDA/BERTopic)
        Viz-->>User: Show highlighted notes, patterns
        %% 7. Synthesis & Recap
        User->>User: Recap findings, discuss variables, gaps, limitations
        %% 8. Exportable Output
        User->>Python: Export notebook (.ipynb or PDF)
        %% 9. Create PowerPoint & Export as PDF
        User->>PPT: Create PowerPoint slides for tutorial
        PPT-->>User: Export PowerPoint as PDF
```

Mermaid Live Editor

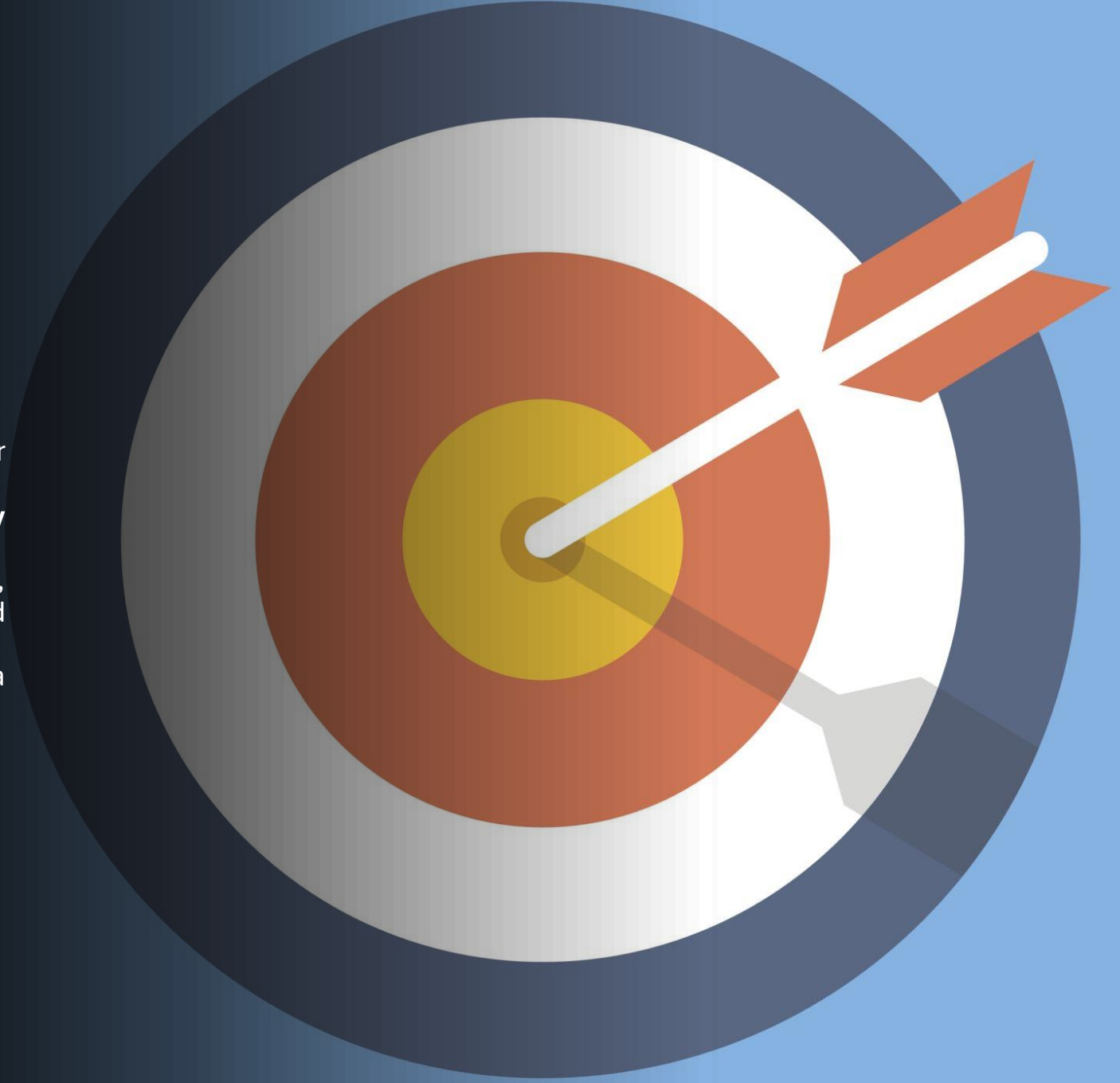
https://mermaid.live/

Synthesis and Workflow in Self-Learning Tutorial (SLT) UML Code

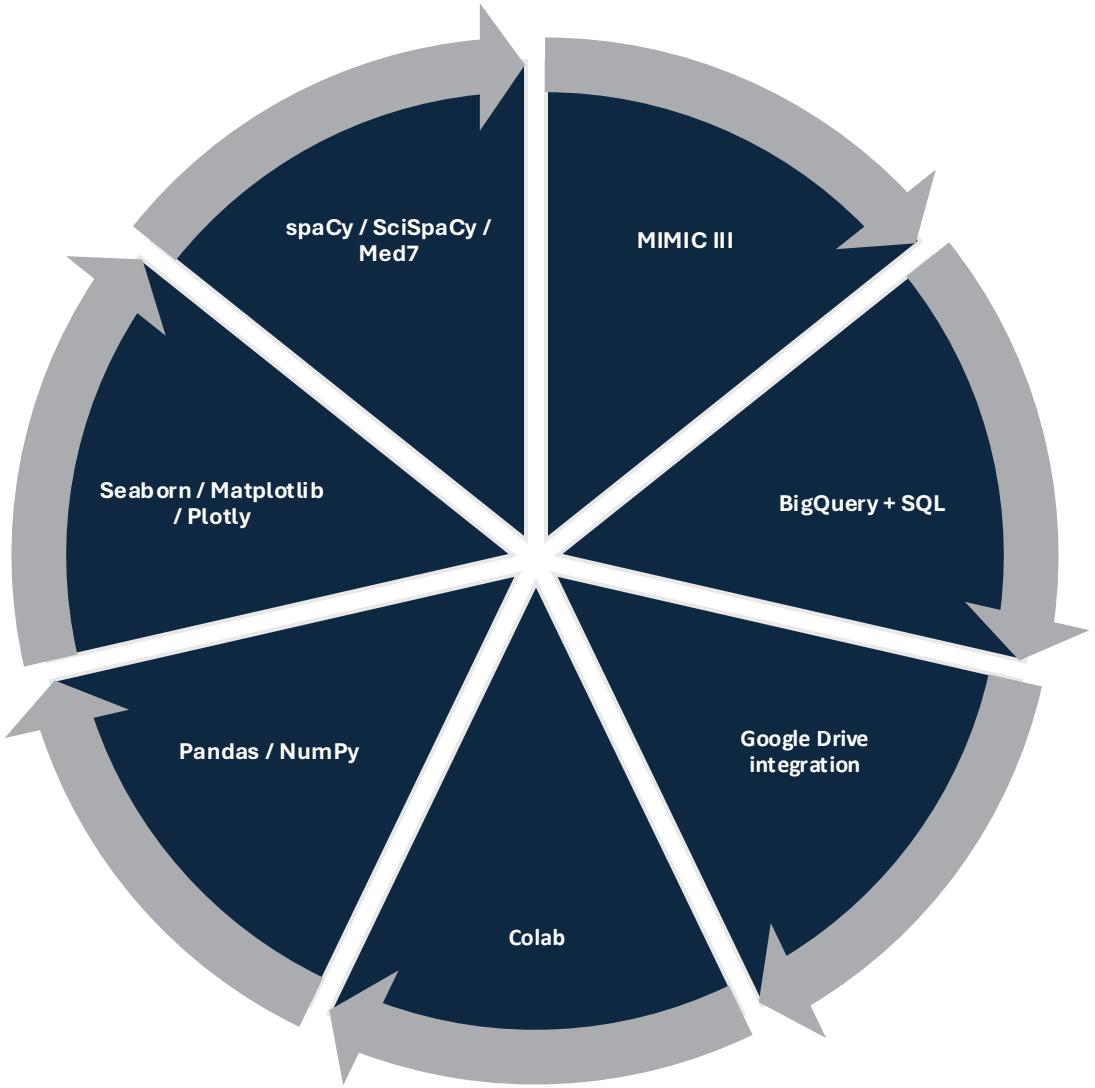
Tutorial Goal

Learners will:

- Understand how social relationships (like marriage or companionship) correlate with healthcare outcomes.
- Gain hands-on experience querying **MIMIC-III** in **BigQuery Learn Clean, Tokenize and Lemmatization, NER**
- Analyze structured data and medical notes using **Python**, **Pandas**, **NLP (spaCy/medSpaCy/SciSpaCy/Med7)**, and **visualization tools**.
- Learn to normalize, visualize, and interpret clinical data with a health policy/public health lens.



Tools/Tech Stack



Collab to Bigquery

BigQuery Integration and Data Prep

- Connect to BigQuery
- Introduce key MIMIC-III tables: `admissions`, `patients`, `diagnoses_icd`, `chartevents`, `noteevents`, etc.
- Write SQLs to explore:
  - Marital status vs. outcomes
  - Length of stay, readmission, mortality, etc.
  - Compare outcomes across married vs. single vs. widowed/divorced patients

```python
# 1. Install Necessary Libraries
# Connecting to BigQuery from Google Colab and pulling data is a straightforward process, primarily involving authentication and using the google-cloud-bigquery library.
# First, we'll need the google-cloud-bigquery library to interact with BigQuery and pandas to work with the data in a DataFrame.
!pip install google-cloud-bigquery pandas
```

```python
# 2. Authenticate Your Colab Environment
# Google Colab provides a simple way to authenticate using your Google account. This is the most common method for interactive use in Colab
from google.colab import auth
auth.authenticate_user()
print('Authenticated')
```

```
Authenticated
```

```python
# 3. Connect Google Cloud Project ID
# We need to use the BigQuery client which Google Cloud Project your data resides in.
project_id = '<<your-project-id>>' # Replace with your actual Google Cloud Project ID
```

```python
# 4. Initialize the BigQuery Client
# Create an instance of the BigQuery client. This object will manage your connection and allow you to send queries.
#  following block is commeneted here because we're conssollidating all imports in one place which will be shown later.
from google.cloud import bigquery
client = bigquery.Client(project=project_id)
print('BigQuery client initialized.')
```

```
BigQuery client initialized.
```

Mortality Counts and Average Age at Death by Marital Status

Mortality Counts and Average Age at Death by Marital Status

Goal:

To determine the number of deceased patients and their average age at death, grouped by marital status.

Implementation:

- Joins patients and admissions tables on subject_id.
- Filters for patients with a non-null date of death (p.dod) and non-null admission time.
- Calculates:
- death_count: Number of deaths per marital status.
- avg_age_at_death: Average age at death, in years, rounded to 1 decimal.
- Results are grouped by marital status and ordered by average age at death (descending).

```python
# 5. Write Your SQL Query
# Craft your SQL query. Remember to specify the full path for your tables using the format resources.dataset_name.table_name.
# This Counts of patients who died within 7 days of hospital admission by marital status and age at death
query = """
SELECT
  a.marital_status,
  COUNT(*) AS death_count,
  ROUND(AVG(DATE_DIFF(p.dod, p.dob, YEAR)), 1) AS avg_age_at_death
FROM
  physionet-data.mimiciii_demo.patients AS p
JOIN
  physionet-data.mimiciii_demo.admissions a
  ON p.subject_id = a.subject_id
WHERE
  p.dod IS NOT NULL
  AND a.admittime IS NOT NULL
  --AND p.dod <= DATE_ADD(a.admittime, INTERVAL 7 DAY)
GROUP BY
  a.marital_status
ORDER BY
  avg_age_at_death DESC;
"""
print('SQL query defined.')
```

```
SQL query defined.
```

Outcome & Interpretation

```python
# import pandas as pd

try:
    df_bigquery_data = client.query(query).to_dataframe()
    print('\nQuery executed successfully! Data loaded into pandas DataFrame.')
    print('First 5 rows of the data:')
    print(df_bigquery_data.head())
except Exception as e:
    print(f"\nAn error occurred: {e}")
    print("Please double-check:")
    print("  - Your `project_id` is correct.")
    print("  - The table names in your SQL query are fully qualified")
    print("  - Your Google account has the necessary BigQuery permissions")
    print("  - The BigQuery API is enabled for your Google Cloud Project.")
```

```
Query executed successfully! Data loaded into pandas DataFrame.
First 5 rows of the data:
      marital_status  death_count  avg_age_at_death
0               None           16             122.4
1            WIDOWED           15              97.7
2            MARRIED           60              81.9
3  UNKNOWN (DEFAULT)            5              79.6
4           DIVORCED            6              76.5
```

Interpretations:

- **Implausible Ages**: The group with `marital_status = None` has an average age at death of 122.4 years, which is biologically implausible and suggests data entry or calculation errors (e.g., missing or incorrect dates of birth/death).
- **Widowed**: High average age at death (97.7), which is plausible given demographic trends.
- **Married**: Largest group (60 deaths), average age at death 81.9.
- **Unknown/Default/None**: These categories indicate incomplete or improperly coded marital status.
- Divorced: Lowest average age at death among these groups (76.5).

Recommendations:

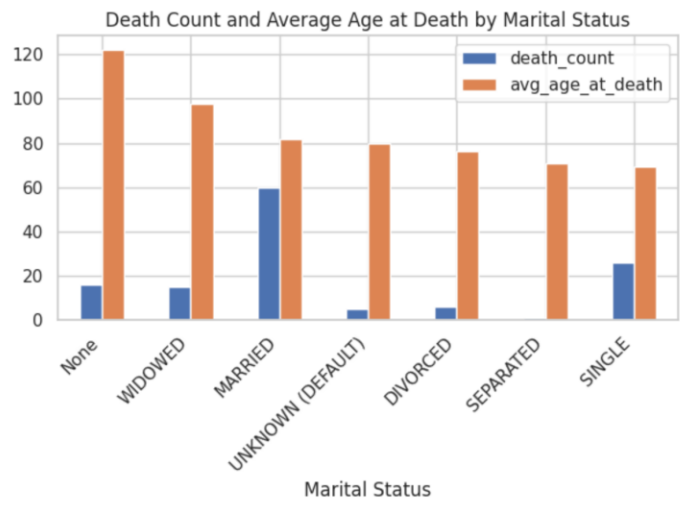
- The results are affected by data quality issues and the lack of a time-to-death filter. Data cleaning is recommended for more accurate insights.
- The presence of implausible ages and ambiguous marital status categories (None, UNKNOWN (DEFAULT)) indicates a need for data cleaning.
- The filter for deaths within 7 days of admission is commented out. As written, the query includes all deaths, not just those soon after admission.
- Clean data by removing implausible ages (e.g., age <0 or >110).
- Standardize or exclude ambiguous marital status.
- We can analyze early mortality, uncomment and use the 7-day filter.

Death Count by Marital Status (Bar Plot)

Goal:

Visualize the query results using a bar chart in pandas and matplotlib, which is ideal for comparing categorical data such as marital status against death counts or average age at death.

```python
# import matplotlib.pyplot as plt

ax = df_bigquery_data.plot.bar(x='marital_status', y=['death_count', 'avg_age_at_death'])
plt.title('Death Count and Average Age at Death by Marital Status')
plt.xlabel('Marital Status')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



- Obtain The Data: We have successfully queried the data from Google BigQuery into a pandas DataFrame named df_bigquery_data.
- Run the Plotting Code: Once you have your df_bigquery_data DataFrame ready, execute the provided Python code in your environment (e.g., a Jupyter Notebook or Google Colab cell).
- Interpret the Plot: After running the code, a bar chart will be displayed. This chart visually represents the death_count and avg_age_at_death for each category of marital_status, allowing for quick comparison and analysis of these metrics across different marital groups.
- Conclusion: This result provides a summary of mortality counts and average age at death by marital status, but the results are affected by data quality issues and the lack of a time-to-death filter. Data cleaning is required for more accurate insights. We can perform this cleaning two ways:
    - 1. with SQL filter or
    - 2. Clean in Pandas after loading

Define BigQuery Helper Function

```python
# Execute the cleaned SQL to retrieve data set
# import pandas as pd

df_bigquery_data = run_bq_query(query)
```

```
Query executed successfully! Data loaded into pandas DataFrame.
First 5 rows of the data:
      marital_status  death_count  avg_age_at_death
0               None           16             122.4
1            WIDOWED           15              97.7
2            MARRIED           60              81.9
3  UNKNOWN (DEFAULT)            5              79.6
4           DIVORCED            6              76.5
```

```python
# Execute the SQL to retrieve data set
# import pandas as pd
def run_bq_query(query):
  try:
      df_bigquery_data = client.query(query).to_dataframe()
      print('\nQuery executed successfully! Data loaded into pandas DataFrame.')
      print('First 5 rows of the data:')
      print(df_bigquery_data.head())
      return df_bigquery_data
  except Exception as e:
      print(f"\nAn error occurred: {e}")
      print("Please double-check:")
      print("  - Your `project_id` is correct.")
      print("  - The table names in your SQL query are fully qualified")
      print("  - Your Google account has the necessary BigQuery permissions")
      print("  - The BigQuery API is enabled for your Google Cloud Project.")
```

Goal:

Now we take the opportunity to improvise our client query to avoid writing redundant code and convert it to a Helper function.

Data Cleansing with SQL:

This SQL query is incorporating data cleaning steps for analysis

```
# SQL query for your analysis, incorporating data cleaning steps
query_cleaned = """
SELECT
  CASE
    WHEN a.marital_status IS NULL
         OR a.marital_status IN ('UNKNOWN (DEFAULT)', 'None', '')
      THEN 'Unknown'
    ELSE a.marital_status
  END AS marital_status_cleaned,
  COUNT(*) AS death_count,
  ROUND(AVG(DATE_DIFF(p.dod, p.dob, YEAR)), 1) AS avg_age_at_death
FROM
  physionet-data.mimiciii_demo.patients AS p
JOIN
  physionet-data.mimiciii_demo.admissions AS a
  ON p.subject_id = a.subject_id
WHERE
  p.dod IS NOT NULL
  AND a.admittime IS NOT NULL
  AND DATE_DIFF(p.dod, p.dob, YEAR) BETWEEN 0 AND 110
  -- Uncomment the next line to restrict to deaths within 7 days of admission
  -- AND p.dod <= DATE_ADD(a.admittime, INTERVAL 7 DAY)
GROUP BY
  marital_status_cleaned
ORDER BY
  avg_age_at_death DESC;
"""
print('SQL query defined.')
```

```
SQL query defined.
```

Cleaned Query Results & Interpretations:

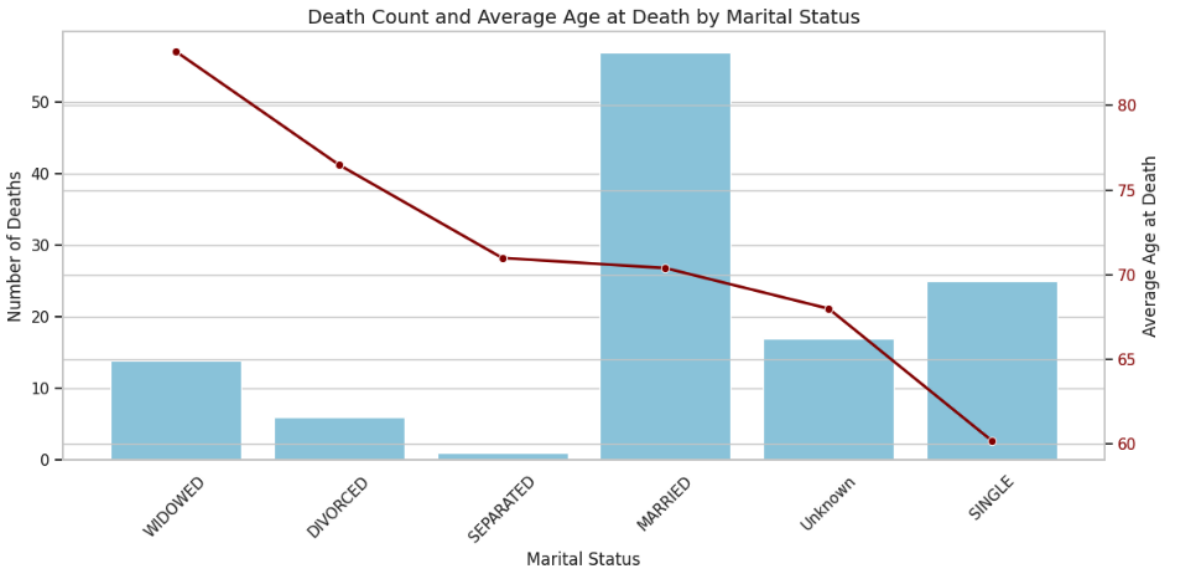
```python
# Execute the cleaned SQL to retrieve data set
# import pandas as pd

marital_counts_cleaned = run_bq_query(query_cleaned)
```

```
Query executed successfully! Data loaded into pandas DataFrame.
First 5 rows of the data:
  marital_status_cleaned  death_count  avg_age_at_death
0                WIDOWED           14              83.2
1               DIVORCED            6              76.5
2              SEPARATED            1              71.0
3                MARRIED           57              70.4
4                Unknown           17              68.0
```

- Married individuals have the highest death count, but they also tend to die at a younger average age (~70) compared to widowed and divorced individuals. This could be due to a higher representation of married individuals in the overall dataset, and not necessarily poorer health outcomes.
- Widowed individuals show a lower death count but the highest average age at death (~83). This indicates longer life expectancy, potentially due to prior long-term companionship or spousal care before widowhood.
- Divorced and separated individuals fall in between - fewer deaths but also lower average age at death than the widowed. May suggest reduced social support after separation, affecting long-term health outcomes.
- Single individuals (from the extended version of the dataset you plotted) show lower life expectancy (~60) with a relatively high number of deaths, supporting the hypothesis that lack of companionship may contribute to poorer outcomes.
- Unknown marital status also skews younger, but is hard to interpret definitively due to lack of context.

Combined Visualization of **Death Count and Average Age at Death by Marital Status**

```python
# Combined Bar + Line Plot: Death Count and Average Age at Death
fig, ax1 = plt.subplots(figsize=(12, 6))

# Bar plot for death count
sns.barplot(
    x='marital_status_cleaned',
    y='death_count',
    data=marital_counts_cleaned,
    ax=ax1,
    color='skyblue'
)
ax1.set_ylabel('Number of Deaths', fontsize=12)
ax1.set_xlabel('Marital Status', fontsize=12)
ax1.set_title('Death Count and Average Age at Death by Marital Status', fontsize=14)
ax1.tick_params(axis='y')
plt.setp(ax1.get_xticklabels(), rotation=45)

# Line plot for average age at death on secondary y-axis
ax2 = ax1.twinx()
sns.lineplot(
    x='marital_status_cleaned',
    y='avg_age_at_death',
    data=marital_counts_cleaned,
    ax=ax2,
    color='darkred',
    marker='o',
    linewidth=2
)
ax2.set_ylabel('Average Age at Death', fontsize=12)
ax2.tick_params(axis='y', labelcolor='darkred')

# Show plot
plt.tight_layout()
plt.show()
```



- **Create a figure and primary axis:** Set the canvas size to ensure clear display.
- **Draw the bar plot :** Use the primary axis to plot the number of deaths per marital status. Apply a consistent color for clarity (Sky blue). Add axis labels and a title. Rotate x-axis labels for readability.
- **Create a secondary y-axis:** Mirror the x-axis but allow for a different y-scale. This axis s used to plot average age at death.
- **Overlay the line plot:** Use the secondary axis to draw a line graph of average age at death. Add data markers for each marital status. Use a distinct color (dark red) to differentiate from the bar plot.
- **Label the secondary axis:** Add appropriate y-axis label and match it to the line color.
- **Finalize the layout :** Apply tight layout to avoid overlaps. Display the combined chart.

Restrict to deaths within 7 days of admission to check early mortality after admission

Mortality Counts and Average Age at Death by Marital Status within 7 days of admission

Goal:

To determine the number of deceased patients and their average age at death, grouped by marital status.

Implementation:

- Joins patients and admissions tables on subject_id.
- Filters for patients with a non-null date of death (p.dod) and non-null admission time.
- Calculates:
- death_count: Number of deaths per marital status.
- avg_age_at_death: Average age at death, in years, rounded to 1 decimal.
- This time Restrict to deaths within 7 days of admission to check early mortality after admission
- Results are grouped by marital status and ordered by average age at death (descending).

```
# Restrict to deaths within 7 days of admission to check early mortality after admission

query_early_mortality = """
SELECT
  CASE
    WHEN a.marital_status IS NULL
         OR a.marital_status IN ('UNKNOWN (DEFAULT)', 'None', '')
      THEN 'Unknown'
    ELSE a.marital_status
  END AS marital_status_cleaned,
  COUNT(*) AS death_count,
  ROUND(AVG(DATE_DIFF(p.dod, p.dob, YEAR)), 1) AS avg_age_at_death
FROM
  physionet-data.mimiciii_demo.patients AS p
JOIN
  physionet-data.mimiciii_demo.admissions AS a
  ON p.subject_id = a.subject_id
WHERE
  p.dod IS NOT NULL
  AND a.admittime IS NOT NULL
  AND DATE_DIFF(p.dod, p.dob, YEAR) BETWEEN 0 AND 110
  AND p.dod <= DATE_ADD(a.admittime, INTERVAL 7 DAY)
GROUP BY
  marital_status_cleaned
ORDER BY
  avg_age_at_death DESC;
"""
print('SQL query defined.')
```

```
SQL query defined.
```

Outcome & Interpretation

```python
# Execute the cleaned SQL to retrieve data set
# import pandas as pd

early_mortality_df = run_bq_query(query_early_mortality)
```

```
Query executed successfully! Data loaded into pandas DataFrame.
First 5 rows of the data:
  marital_status_cleaned  death_count  avg_age_at_death
0                WIDOWED            2              85.0
1                MARRIED            7              79.1
2               DIVORCED            3              72.3
3                Unknown            7              64.4
4                 SINGLE            4              51.8
```

Interpretations:

- **WIDOWED** patients, although few in count, had the **highest average age at early death**, potentially due to **limited social support** post-loss of spouse.
- **MARRIED** patients had relatively **more early deaths** (7) but at a **higher average age (79.1)**, suggesting **supportive companionship may extend life** even in acute cases.
- **SINGLE** individuals had **lower average age at death (51.8)**—a striking indicator that **lack of support correlates with younger mortality**.
- **Unknown** marital status shows a comparable death count to married (7) but **much lower age at death**, which may indicate either missing documentation or neglected social support tracking.

Recommendations:

- Explore Further Stratification by Breaking down these stats by ICU type, gender, or comorbidity index.
- NLP Follow-Up to use unstructured clinical notes to detect loneliness, depression, or caregiver presence.

Early Mortality with in 7 days of admission (Bubble Plot)

Goal:

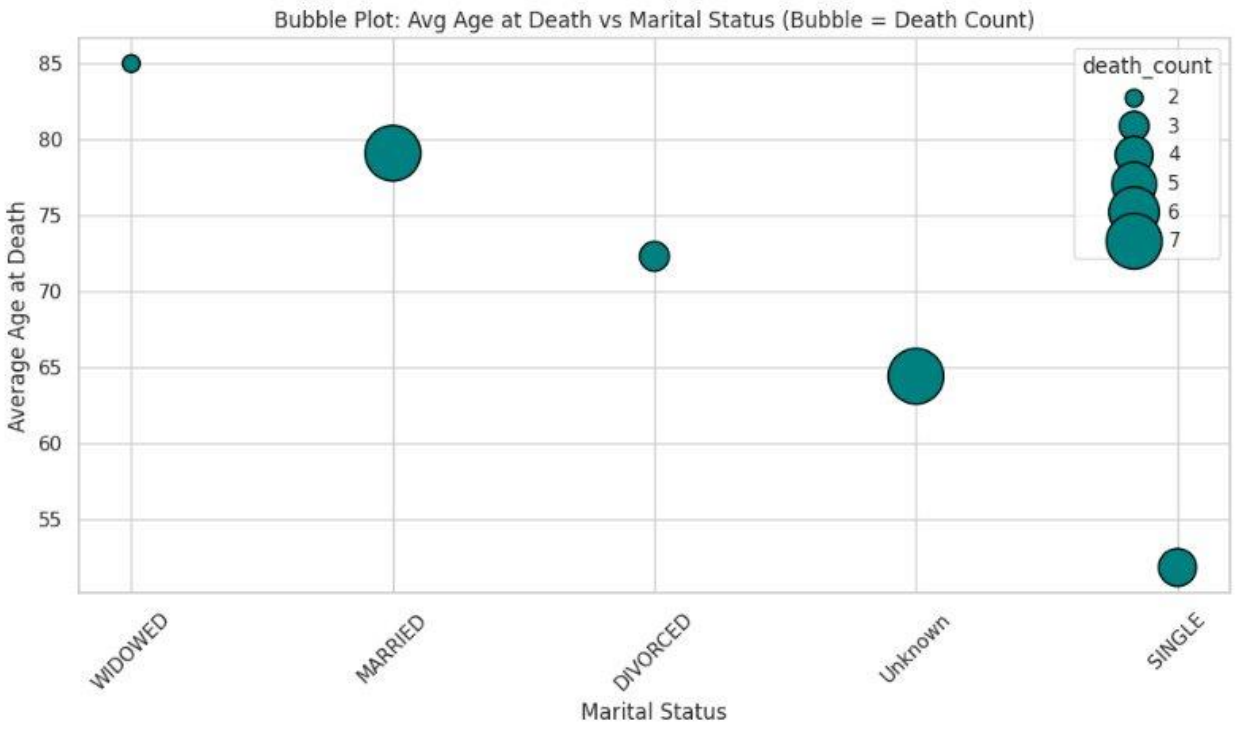
Visualize early mortality within 7 days of admission by marital status using a bubble plot, where the x-axis represents marital status, the y-axis shows average age at death, and bubble size encodes the number of deaths. This enables clear, multi-dimensional comparison of early mortality patterns across social relationship groups.

```python
# Early Mortality with in 7 days of admission (Bubble Plot)

# Convert DataFrame to dictionary with lists as values
dict_stats = early_mortality_df.to_dict(orient='list')

# Bubble Plot: Death count (size), average age (y), marital status (x)
plt.figure(figsize=(10,6))
sns.scatterplot(
    data=death_stats,
    x='marital_status_cleaned',
    y='avg_age_at_death',
    size='death_count',
    legend='brief',
    sizes=(100, 1000),
    color='teal',
    edgecolor='black'
)
plt.title('Bubble Plot: Avg Age at Death vs Marital Status (Bubble = Death Count)')
plt.xlabel('Marital Status')
plt.ylabel('Average Age at Death')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Code Implementation:

- Convert DataFrame early_mortality_df to a dictionary with columns as keys and lists of column values as values, using the to_dict method with the 'list' orientation.
- Set up your plot area with a suitable figure size.
- Use Seaborn's scatterplot function to create a bubble plot:
- Set the x-axis to marital status.
- Set the y-axis to average age at death.
- Set the bubble size proportional to the death count.
- Customize the appearance with color, edge color, and legend.
- Add a title and axis labels, rotate the x-axis labels for readability, and use tight layout to prevent overlap.
- Display the plot.

This workflow will visually encode three variables—marital status, average age at death, and death count—into a single, interpretable bubble plot.



Visualization:

- **Conclusions**:

Married individuals likely benefit from stronger social and emotional support, which may delay mortality even during early critical illness. Those listed as single or divorced/separated often showed earlier average age of death—possibly due to loneliness, stress, or poor health adherence. Widowed Group: Though older (expected), their mortality can reflect loss of companionship impact.

Load files form Google Drive

`df_4019_radiology_notes.csv` is a pre-filtered CSV file containing records with the ICD-9 code **4019** (Hypertension), specifically filtered for **Radiology** events. The source of this data is the **NOTEEVENTS** table from the **MIMIC-III database**.

```python
# Steps to Load CSV from Google Drive in Google Colab
# df_4019_radiology_notes.csv' is a pre conditined csv files with ICD9 code 4019, which is "Hypertension"
# and we have filtered for "Radiology" events
from google.colab import auth, drive, files

drive.mount('/content/drive')
auth.authenticate_user()

df_copy = pd.read_csv('/content/drive/My Drive/df_4019_radiology_notes.csv')
print(df_copy.head(3))
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_
   ROW_ID  SUBJECT_ID   HADM_ID   CHARTDATE            CHARTTIME  STORETIME  \
0  738405       93207  104110.0  2166-10-09  2166-10-09 20:24:00        NaN
1  738661       24552       NaN  2153-01-12  2153-01-12 22:03:00        NaN
2  738662       10390       NaN  2145-03-13  2145-03-13 02:58:00        NaN

    CATEGORY                  DESCRIPTION  CGID  ISERROR  \
0  Radiology  DISTINCT PROCEDURAL SERVICE   NaN      NaN
1  Radiology          CHEST (PORTABLE AP)   NaN      NaN
2  Radiology         CT PELVIS W/CONTRAST   NaN      NaN

                                                TEXT
0  [**Last Name (LF) **],[**First Name3 (LF) **] ...
1  [**2153-1-12**] 10:03 PM\n CHEST (PORTABLE AP)...
2  [**2145-3-13**] 2:58 AM\n CT ABDOMEN W/CONTRAS...
```

Steps to Load a CSV from Google Drive in Google Colab

Import necessary libraries for Google Drive access and authentication.

Mount your Google Drive to the Colab environment so files can be accessed like a local filesystem.

Authenticate access when prompted by following the authorization URL and entering the code.

Locate the full path to the CSV file in your Google Drive (e.g., inside "My Drive").

Read the CSV file into a pandas DataFrame using the full file path.

Verify the data by displaying the first few rows of the DataFrame.

Clean and Tokenize

Clean and Tokenize Texts

```python
1 # cleans and tokenize
2 # import spacy
3 # import re
4 # import string
5 # from collections import Counter
6
7 # import pandas as pd
8 # import numpy as np
9
10 # from spacy.pipeline import EntityRuler
11 # from spacy.tokens import DocBin
12
13 nlp = spacy.load("en_core_web_sm")  # Use medSpaCy model here if installed
14 stop_words = nlp.Defaults.stop_words
15
16 def clean_and_tokenize(text,
17                        lowercase=True,
18                        remove_stopwords=True,
19                        remove_punctuation=True,
20                        min_token_len=2):
21
22     if lowercase:
23         text = text.lower()
24
25     doc = nlp(text)
26
27     tokens = []
28     for token in doc:
29         if remove_stopwords and token.text in stop_words:
30             continue
31         if remove_punctuation and token.text in string.punctuation:
32             continue
33         if len(token.text) < min_token_len:
34             continue
35         if token.is_space or token.is_digit:
36             continue
37         tokens.append(token.lemma_.strip())
38
39     return " ".join(tokens)
```

```python
1 df_copy['cleaned_tokens'] = df_copy["TEXT"].apply(lambda text: clean_and_tokenize(text))
```

Library Imports (Lines 1–13)
Imports essential NLP and data processing libraries like spacy, re, string, and Counter. pandas and numpy are included for dataframe operations and numerical tasks. Some optional SpaCy components (EntityRuler, DocBin) are commented out but can be used for advanced customization.

Loading SpaCy Language Model (Line 13)
Loads the en_core_web_sm model for English NLP tasks.
You can replace this with a more medically tuned model like medSpaCy if available.

Stop Words Setup (Line 14)
Retrieves default English stop words (e.g., “the,” “is,” “in”) from SpaCy for later filtering.

Token Cleaning Function: clean_and_tokenize This custom function processes raw text through several steps

Lowercase (Line 22) – Converts all text to lowercase (if enabled).

SpaCy Tokenization (Line 24) Breaks text into SpaCy Token objects.
Filtering Loop (Lines 27–36) Removes stop words, punctuation, very short tokens (e.g., length < 2), digits, and spaces. Only keeps the **lemma** (base form) of each remaining token, with whitespace stripped.

Join Clean Tokens (Line 39) – Converts the cleaned list of tokens back to a single string.

Apply Cleaning to DataFrame Column (Line 41)
Applies the clean_and_tokenize() function to each row in the TEXT column of df_copy.
Stores the output in a new column named 'cleaned_tokens'

Benefits of This Approach

Tokenization with SpaCy
This uses a powerful natural language processing engine that understands the structure of language. It breaks down text into tokens, identifies their lemmas (base forms), and can recognize parts of speech and named entities.

Lowercasing & Stopword Removal
Converting text to lowercase ensures consistency, and removing common stop words (like “the,” “is,” “and”) reduces noise in the data. This helps in improving the clarity of models and visualizations.

Lemmatization
Lemmatizing words reduces them to their root forms—for example, “running” becomes “run.” This helps in grouping similar words together and provides more meaningful insights during analysis.

Punctuation & Digit Removal
Removing punctuation and numbers strips out irrelevant characters that don’t contribute to the semantic meaning of the text.

Minimum Token Length Filtering
This step helps eliminate short, often meaningless fragments (like “a” or “x”) which are not useful in text analysis.

Result: Cleaned Text
The final output is a cleaned, structured version of the original text. This dramatically improves the quality and accuracy of downstream tasks such as keyword extraction, topic modeling, and word embeddings.

Named Entity Recognition (NER)

NER helps you quickly inspect in recognizing and labeling entities in your notes, and visually highlights them for manual review or further analysis

```python
# from spacy import displacy

nlp = spacy.load("en_core_web_sm")
# Loop through first 3 processed notes and render entities
for i in range(10):
    text = df_copy['cleaned_tokens'][i]
    doc = nlp(text)
    displacy.render(doc, style="ent", jupyter=True)
    print("**********************")
```

If name3 ORG lf eu 8:24 pm TIME cta head w&w recon cta neck w&w oc recon clip clip number radiology ct brain perfusion -59 distinct procedural service reason eval ich cva ORG contrast optiray amt hospital medical condition year old woman new onset dysarthria know reason examination eval ich cva contraindication iv contrast pfi report evidence acute infarct intracranial hemorrhage brain evidence focal flow limit stenosis occlusion aneurysm great mm artery anterior posterior circulation head calcify plaque left distal common carotid artery cause approximately stenosis

****

10:03 pm TIME chest portable ap clip clip number radiology reason scv cvl PERSON check line ptx hospital medical condition year old woman gangrene rt foot PERSON rt fem tib reason examination scv cvl check line ptx final report clinical indication central venous catheter placement comparison previous study day early interval placement left subclavian NORP central venous catheter terminate junction left brachiocephalic vein PERSON superior vena cava pneumothorax identify cardiac silhouette remain enlarge bilateral pleural effusion adjacent area increase lung opacification low lobe note previous mediansternotomy coronary bypass surgery impression central venous catheter terminate confluence left brachiocephalic vein superior vena cava pneumothorax persistent bilateral pleural effusion adjacent lung opacity likely reflect atelectasis infectious process exclude proper clinical setting

****

2:58 TIME ct abdomen contrast ct pelvis contrast clip clip number radiology ct coronal sagital obl reconstruction ct 150cc nonionic contrast LOC reason evaluate diverticulitis PRODUCT contrast optiray amt hospital medical condition year old woman diverticuli ORG recent admit diverticulitis PRODUCT low gi bleed -- ct show thickening stranding paracolic fluid left descending prox sigmoid colon worrisome diverticulitis vs ischemic bowel vs hemorrhage bowel wall present recurrent llq pain diarrhea evaluate diverticulitis perforation reason examination evaluate diverticulitis final report history diverticulitis technique helically acquire ct image lung basis pubic symphysis contrast ct abdomen contrast comparison prior ct scan date demonstrate lung basis mm PERSON calcify granuloma left base lung clear pleural effusion liver spleen pancrea adrenal kidney unremarkable see large cm gallbladder stone gallbladder appear unremarkable bowel upper abdomen appear unremarkable free fluid see ct pelvis contrast mark interval improvement thicken sigmoid colon see prior study numerous diverticula minimal stranding sigmoid colon likely clear residual disease free fluid see kidney demonstrate bilateral symmetrical contrast excretion ureter see course bladder filling defect bladder appear unremarkable free fluid present focal fluid collection identify bone window bone demonstrate suspicious lytic blastic lesion impression interval improvement patient sigmoid disease mild stranding sigmoid colon likely relate residua prior disease rule early diverticulitis

****

11:46 CARDINAL ct head w/o contrast clip clip number radiology reason cerebral edema central pontine myelinolysis mass hospital medical condition year old man liver transplant w/ persistent decrease mental status reason examination cerebral edema central pontine myelinolysis mass final report history year old DATE man status post liver transplant persistent decrease mental status rule cerebral edema central pontine myelinolysis technique contiguous axial image head obtain iv contrast skull base vertex ct head w/o contrast sulci ventricle appropriate patient age intra extraaxial fluid collection masse shift midline structure observe doctor -white matter differentiation unremarkable impression evidence cerebral edema

****

1:06 pm TIME chest portable ap clip clip number radiology reason chf hospital medical condition year old woman niddm cad aicd pace acute sob leg swelling reason examination chf final report indication year old DATE woman diabete cad present follow icd placement acute shortness breath leg swelling rule chf comparison upright ap chest radiograph interval placement left side dual chamber icd lead locate right atrium right ventricle persistent cardiomegaly slight decrease extent upper zone redistribution pulmonary vascularity increase size large right side pleural effusion associate right middle low lobe collapse consolidation linear opacity left cp angle likely represent subsegmental atelectasis interval decrease extent left pleural effusion soft tissue osseous structure unchanged impression new left side dual chamber icd lead appropriate position pneumothorax persistent cardiomegaly GPE interval decrease chf interval increase large right pleural effusion small left pleural effusion leave basilar subsegmental atelectasis right middle low lobe collapse consolidation

****

2:21 pm TIME chest portable ap clip clip number radiology reason acute sob hospital medical condition year old man pvd ischemic lle cad PERSON heparinwithacute sob low bp reason examination acute sob final report indication shortness breath low blood pressure comparison previous study early day DATE left picc line remain place terminate distal superior vena cava patient mediansternotomy coronary bypass surgery heart enlarge stable size previously note mild congestive heart failure pattern demonstrate significantly change allow difference technique confluent area consolidation see lung pleural effusion evident left cp sulcus exclude impression significant change recent study early day DATE persistent mild congestive heart failure pattern

****

4:00 pm TIME chest pa lat clip clip number radiology reason follow cxr pt pneumomia rml hospital medical condition year old man sob chest leave le dvt reason examination follow cxr pt pneumomia rml final report indication right middle lobe pneumonia chest view consolidation collapse right middle lobe anterior segment right low lobe significantly change minimal linear atelectasis left lung base small right pleural effusion present left costophrenic angle clear pulmonary vascularity normal cardiomediastinal silhouette normal impression consolidation collapse right middle lobe anterior segment right low lobe unchanged delayed follow time course week recommend evaluate long term change

****

8:33 TIME ct chest w/o contrast clip clip number radiology reason new pulmonary nodule right upper lobe hospital medical condition year old ORG woman copd exacerbation incidental finding solitary pulmonary nodule cxr complain occasional tightness area chest pack year hx smoking reason examination new pulmonary nodule right upper lobe final report indication year old DATE female chronic obstructive pulmonary disease exacerbation new nodule right upper lobe chest radiograph positive smoking history helical ct thorax perform intravenous oral contrast administration mm PERSON collimation mm PERSON reconstruction interval image lung reveal evidence right upper lobe pulmonary nodule apparent nodule chest radiograph show healing right- sided rib fracture evidence emphysema PERSON severe upper lobe mild severity remain portion lung left low lobe streaky peripheral opacity predominantly linear ORG configuration associate ground glass opacification scatter linear opacity see lingula periphery right upper lobe probably reflec scar central airways appear patent segmental level bilaterally minimal small aiway disease posteriorly right low lobe leave low lobe manifest small branching y- shape structure review osseous structure thorax reveal healing subacute fracture right fifth ORDINAL posterior rib assessment soft tissue structure thorax reveal lymph node precarinal area ORG slightly cm short axis

Code walkthrough and Utility of NER

- This code uses spaCy and its visualization tool displaCy to process and visually inspect named entities in clinical or textual notes. Here's a step-by-step explanation:
- Model Loading:
  nlp = spacy.load("en_core_web_sm")
  Loads the small English language model, which includes a pre-trained named entity recognizer (NER).
- Iterate Through Notes:
  The for loop goes through the first 10 entries (i in range(10)) in the DataFrame column cleaned_tokens, which is assumed to contain preprocessed text notes.
- Process Text:
  For each note, doc = nlp(text) processes the text with the spaCy pipeline, including tokenization, tagging, and NER.
- Visualize Entities:
  displacy.render(doc, style="ent", jupyter=True)
  Uses displaCy to highlight and label any named entities detected in the text (such as PERSON, DATE, ORG, etc.) directly in a Jupyter notebook cell.
  - The style="ent" argument specifies that named entities should be visualized.
  - The jupyter=True flag ensures the visualization appears inline in Jupyter notebooks.
- Separator:
  print("****")
  Prints a line of asterisks to visually separate the output for each note.

Benefits of NER (Named Entity Recognition):

- Extracts key info from unstructured text (e.g., names, dates, places).
- Saves time by automating data tagging and classification.
- Enables analysis like trend detection, summarization, and search.
- Improves decision-making with structured insights from raw text.
- Supports automation in healthcare, finance, legal, and more.

Most Common Diagnoses by Marital Status in MIMIC-III Admissions

[98]

```python
# Top most frequent diagnoses (by short title) grouped by marital status.

query_early_critical = """
SELECT
  CASE
    WHEN adm.marital_status IS NULL
         OR adm.marital_status IN ('UNKNOWN (DEFAULT)', 'None', '')
      THEN 'Unknown'
    ELSE adm.marital_status
  END AS marital_status_cleaned,
  dd.short_title AS diagnosis_name,
  COUNT(*) AS diagnosis_count
FROM
  `physionet-data.mimiciii_demo.admissions` AS adm
JOIN
  `physionet-data.mimiciii_demo.diagnoses_icd` AS d
  ON adm.hadm_id = d.hadm_id
JOIN
  `physionet-data.mimiciii_demo.d_icd_diagnoses` AS dd
  ON d.icd9_code = dd.icd9_code
WHERE
  dd.short_title IS NOT NULL
GROUP BY
  marital_status_cleaned, diagnosis_name
ORDER BY
  marital_status_cleaned, diagnosis_count DESC
LIMIT 10000;
"""
```

```python
query_early_critical_df = run_bq_query(query_early_critical)
```

```
Query executed successfully! Data loaded into pandas DataFrame.
First 5 rows of the data:
  marital_status_cleaned           diagnosis_name  diagnosis_count
0               DIVORCED         Hypertension NOS                4
1               DIVORCED      Atrial fibrillation                3
2               DIVORCED                  CHF NOS                2
3               DIVORCED       Hypothyroidism NOS                2
4               DIVORCED  Ac posthemorrhag anemia                2
```

This process identifies the most frequently occurring medical diagnoses across different marital status categories in the MIMIC-III dataset. It helps examine potential health trends linked to social support. We get a detailed profile of which illnesses are most common for married, divorced, widowed, single, and unknown-status patients - providing insight into how social factors might influence disease burden.

Standardize Marital Status: Marital statuses like "UNKNOWN", "None", or empty values are grouped under a single label, "Unknown", to ensure consistent categorization.

Gather Diagnoses: Diagnoses for each patient are extracted using short, human-readable labels that describe the condition (like: “Hypertension”).

Join Relevant Tables: Combines data from admissions, diagnosis events, and diagnosis dictionary to map each hospital stay to a diagnosis and corresponding marital status.

Count Frequencies: Calculates how many times each diagnosis occurs within each marital status group.

Group and Rank: Organizes the data by marital status and ranks diagnoses from most to least frequent.

Filter Out Invalid Data: Any diagnosis without a proper title is excluded to maintain clean and interpretable results.

Limit Output:

Keeps the results manageable by returning only the top 10,000 records.

From Raw Counts to Risk Scores: Ranking Diagnoses Smarter

```python
# Assign Critical Rating

# Assign critical rating based on frequency rank within each marital group
query_early_critical_df['critical_rating'] = query_early_critical_df.groupby('marital_status_cleaned')['diagnosis_count'] \
                                             .transform(lambda x: pd.qcut(x.rank(method='first'), 5, labels=[1, 2, 3, 4, 5])

# Preview
query_early_critical_df.head(1000)
```

Group by

- Group by Marital Status: The data is first separated into groups based on each patient's marital status (e.g., Married, Divorced, Widowed, etc.).

Rank

- Rank Diagnoses by Frequency: Within each marital group, diagnoses are ranked based on how frequent they appear. The most common diagnosis in that group gets the highest rank.

Divide

- Divide into Buckets (Quantiles): These ranked diagnoses are split into five equal-sized groups (quintiles). Each group represents a different level of criticality based on how common the diagnosis is.

Assign

- Assign a Critical Rating (1 to 5): A numeric rating is assigned:
- 5 for most critical/common diagnoses (top 20%),
- 1 for least frequent/critical (bottom 20%).

Attach

- Attach Rating to Each Record:
  - The new rating is added as a column to the dataset, so each diagnosis now carries a "criticality score" its marital status group.

Identifies Group-Specific Risks: Flags the most urgent diagnoses within eachmarital status group.

lags the most urgent diagnoses within eachmarital status group.

Simplifies Visualization: Enables easy color-coding or ranking in charts anddashboards.

Aids Prioritization: Highlights top-rated (5) conditions for focused clinical attention.

Ensures Fair Comparison: Normalizes across groups, avoiding bias from group sizeor volume.

Supports Risk Modeling: Useful as input for predictive analytics and health riskscoring.

Assign critical rating based on frequency rank within each marital group

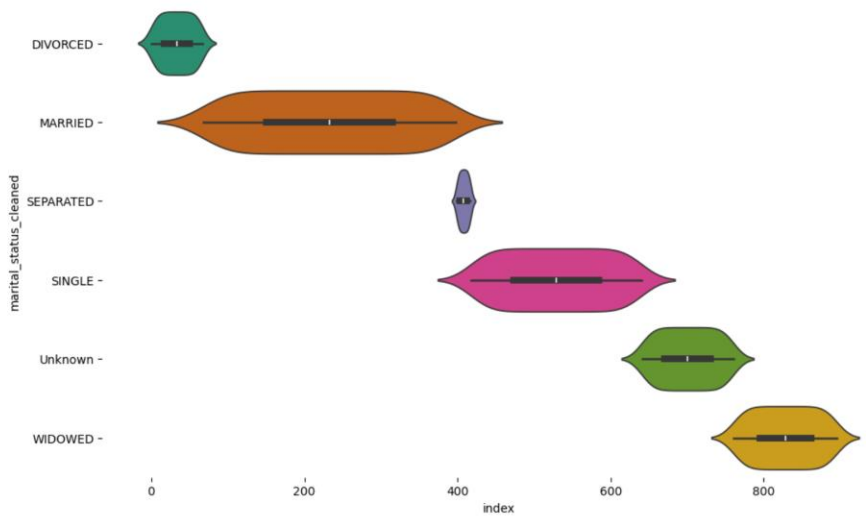
Search by all fields:

index	marital_status_cleaned	diagnosis_name	diagnosis_count	critical_rating
0	DIVORCED	Hypertension NOS	4	5
1	DIVORCED	Atrial fibrillation	3	5
2	DIVORCED	CHF NOS	2	5
3	DIVORCED	Hypothyroidism NOS	2	5
4	DIVORCED	Ac posthemorrhag anemia	2	5
5	DIVORCED	DMII wo cmp nt st uncntr	2	5
6	DIVORCED	Delirium dlt other cond	2	5
7	DIVORCED	Severe sepsis	2	5
8	DIVORCED	Acute respiratry failure	2	5
9	DIVORCED	Cholangitis	1	1
10	DIVORCED	Shock w/o trauma NEC	1	1
11	DIVORCED	Emphysema NEC	1	1
12	DIVORCED	Anoxic brain damage	1	1
13	DIVORCED	Urin tract infection NOS	1	1
14	DIVORCED	Traum arthropathy-pelvis	1	1
15	DIVORCED	Hyperlipidemia NEC/NOS	1	1
16	DIVORCED	Alcoh dep NEC/NOS-unspec	1	1
17	DIVORCED	Septic shock	1	1
18	DIVORCED	Ac kidny fail, tubr necr	1	1
19	DIVORCED	Fx malar/maxillary-close	1	1
20	DIVORCED	Ac alcoholic hepatitis	1	1
21	DIVORCED	Adv eff opiates	1	1
22	DIVORCED	Hypopotassemia	1	1
23	DIVORCED	Idiopathic scoliosis	1	2
24	DIVORCED	Hx-prostatic malignancy	1	2
25	DIVORCED	Mitral valve disorder	1	2
26	DIVORCED	Dvrtclo colon w/o hmrhg	1	2
27	DIVORCED	Persistent vegtv state	1	2
28	DIVORCED	Paralysis agitans	1	2
29	DIVORCED	Acute pancreatitis	1	2
30	DIVORCED	Pneumonia, organism NOS	1	2
31	DIVORCED	Comp-ren dialys dev/grft	1	2
32	DIVORCED	Blood in stool	1	2
33	DIVORCED	Hx-circulatory dis NEC	1	2
34	DIVORCED	Septicemia NOS	1	2
35	DIVORCED	Hyposmolality	1	2
36	DIVORCED	Hyp kid NOS w cr kid V	1	3
37	DIVORCED	Ac systolic hrt failure	1	3
38	DIVORCED	Ventricular fibrillation	1	3
39	DIVORCED	Subendo infarct, initial	1	3
40	DIVORCED	End stage renal disease	1	3
41	DIVORCED	Hematoma complic proc	1	3
42	DIVORCED	Cardiogenic shock	1	3
43	DIVORCED	Unc behav neo bone	1	3
44	DIVORCED	Cerv spondyl w myelopath	1	3
45	DIVORCED	Personal history of fall	1	3
46	DIVORCED	Status autm crd dfbrltr	1	3
47	DIVORCED	Iatrogen pulm emb/infarc	1	3
48	DIVORCED	Other pulmonary insuff	1	3
49	DIVORCED	Iatrogenc hypotnsion NEC	1	3

Uncovering Hidden Health Patterns with Criticality Scores

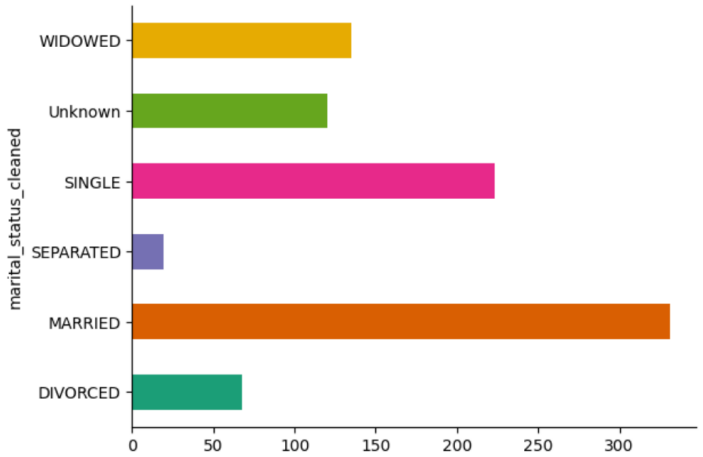
```python
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_17['marital_status_cleaned'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_17, x='index', y='marital_status_cleaned', inner='box', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
/tmp/ipython-input-108-72477812.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set
```



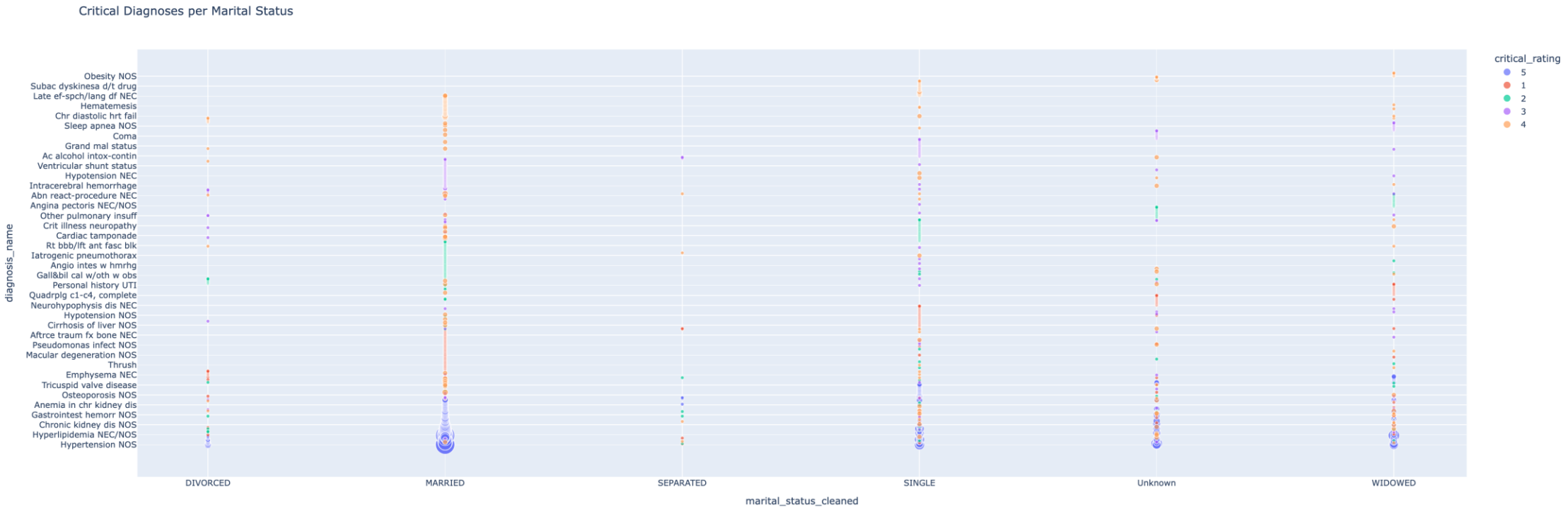
[107]

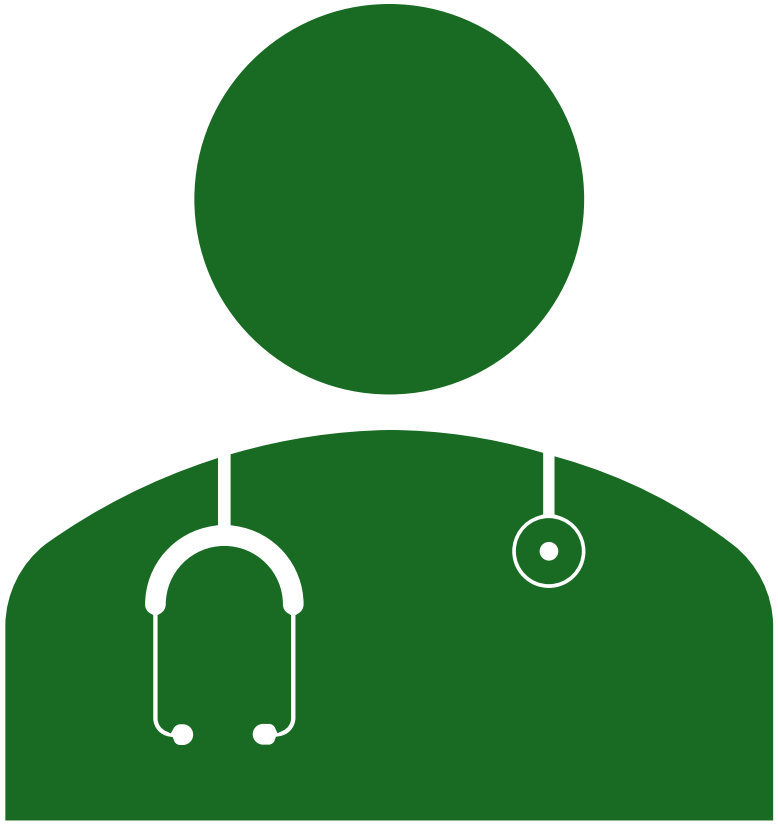
```python
from matplotlib import pyplot as plt
import seaborn as sns
_df_11.groupby('marital_status_cleaned').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



Intuitive, Interactive Insight into Critical Diagnoses Across Marital Status Groups

```python
import plotly.express as px

fig = px.scatter(
    query_early_critical_df,
    x='marital_status_cleaned',
    y='diagnosis_name',
    size='diagnosis_count',
    color='critical_rating',
    color_continuous_scale='Reds',
    title='Critical Diagnoses per Marital Status',
    hover_data=['diagnosis_name', 'diagnosis_count', 'critical_rating']
)
fig.update_layout(height=800)
fig.show()
```





Benefits of This Visualization Approach

Highlights Group-Specific Health Risks: You can easily compare which diagnoses are most common and critical among different marital statuses.

Size and Color Encode Multiple Data Dimensions: Using both bubble size and color allows quick understanding of two variables (count and criticality) simultaneously without clutter.

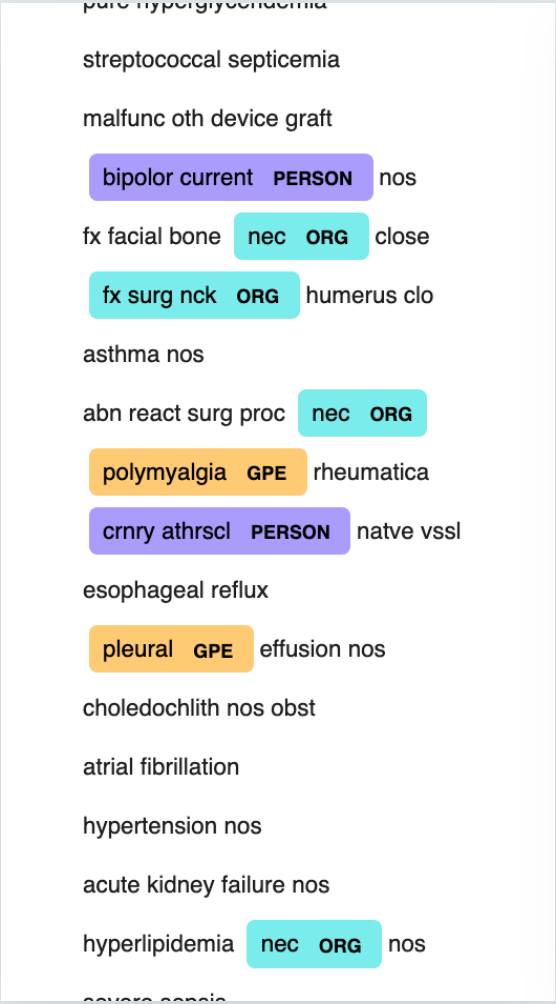
Interactive Exploration: Hover details provide granular data without overwhelming the visual, making it suitable for presentations or clinical reviews.

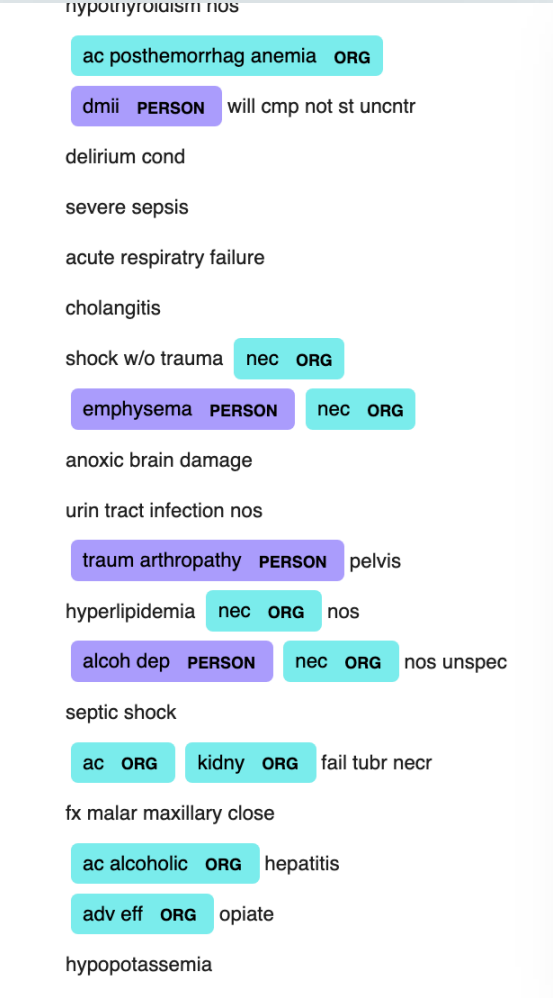
Facilitates Prioritization: Identifying which diagnoses have the highest critical rating and frequency can help prioritize healthcare interventions or further analysis.

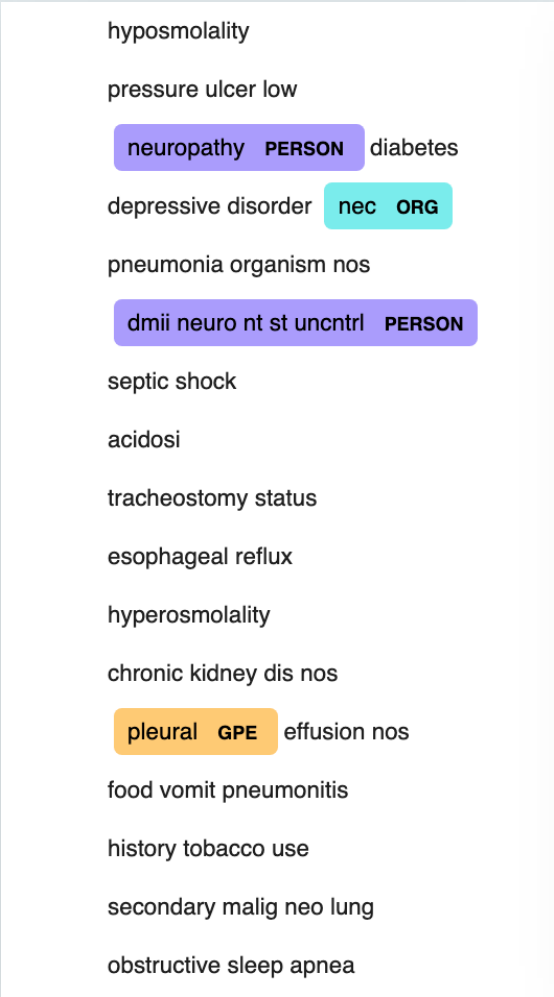
Clear Visual Segmentation: The categorical x-axis groups patients logically, making patterns and disparities easy to spot.

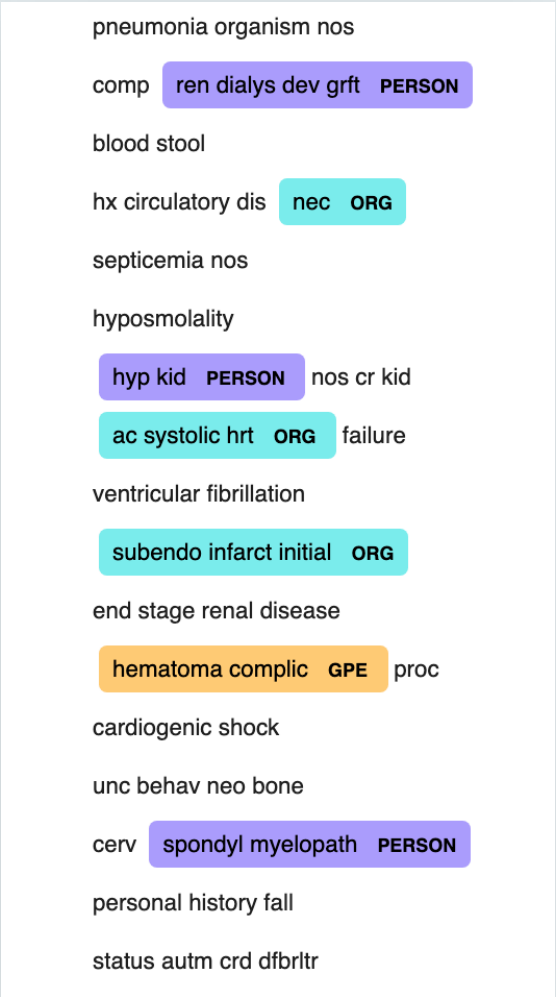
Customizable and Scalable: Plotly's interactive features allow zooming, panning, and easy modification, adapting well for dashboards or reports.

Leveraging NLP to Extract and Visualize Medical Entities from Diagnoses









Named Entity Recognition (NER) on Clinical Diagnoses Using spaCy

```python

query_early_critical_df['cleaned_tokens'] = query_early_critical_df["diagnosis_name"].apply(lambda text: clean_and_tokenize(text))

nlp = spacy.load("en_core_web_sm")
# Loop through first 3 processed notes and render entities
for i in range(100):
    text = query_early_critical_df['cleaned_tokens'][i]
    doc = nlp(text)
    displacy.render(doc, style="ent", jupyter=True)
    # print("************************")
```

Code Walkthrough:

Text Cleaning and Tokenization:
The code applies a custom `clean_and_tokenize` function to the `"diagnosis_name"` column of the DataFrame, creating a new column `"cleaned_tokens"` containing the processed text.

Load spaCy Model:
Loads the pre-trained English language model (`en_core_web_sm`) to enable NLP tasks including Named Entity Recognition (NER).

Entity Extraction and Visualization:
Iterates over the first 100 cleaned diagnosis texts. For each, it:

- Processes the text with spaCy to generate a `Doc` object containing linguistic annotations.
- Uses `displacy.render` to visually highlight and label any named entities (such as diseases, drugs, or other biomedical terms) found in the text, directly in a Jupyter notebook cell for easy inspection.

Benefits of spaCy Modeling Approach

Transforms Unstructured Text into Structured Data: Converts raw diagnosis text into tokenized and labeled components that computers can work with efficiently.

Highlights Clinically Relevant Information: Automatically identifies and visualizes important medical entities, facilitating better understanding of diagnoses.

Supports Data Exploration and Quality Checks: Visualizing entities helps spot inconsistencies or errors in the data early on.

Enables Downstream Analytics: Structured entities can be used in predictive models, clustering, or further natural language processing tasks.

Improves Interpretability: Makes complex clinical notes accessible for clinicians, researchers, or data scientists through clear visualization.

Scalable to Large Datasets: Automates entity recognition on many records, saving manual review time and effort.

The Unseen Role of Social Support: Marital Status and Healthcare Outcomes

While clinical factors are vital in healthcare outcomes, analysis of the MIMIC III dataset reveals that social determinants, especially marital status, play a crucial yet often overlooked role in patient health trajectories.

Research shows that married individuals tend to have better outcomes. This advantage is linked to stronger support systems that promote treatment adherence, provide emotional and practical help, and encourage earlier medical intervention. Such support can improve recovery rates and reduce mortality risk.

Our MIMIC III analysis highlights the vulnerability of single or isolated patients. Without immediate social support, these individuals may delay seeking care, face emotional challenges during health crises, and encounter higher risks of complications or death, particularly after major events like surgery.

Marital status acts as a proxy for broader social support networks in health research. When combined with age at death, it offers deeper insights. For example, a younger average age at death among single patients suggests an urgent need for targeted support and intervention. Conversely, older age at death may reflect natural disease progression, but social support remains a critical factor.

This study emphasizes that clinical data alone cannot capture the full picture. Ignoring social factors like marital status risks missing key influences on patient outcomes. Future research should integrate social determinants more systematically, improve data quality by addressing missing or null marital statuses, and adjust for variables like age, gender, illness severity, and comorbidities. Only then can we fully understand the complex interplay of clinical and social factors shaping health outcomes in datasets like MIMIC III.

https://mermaid.live/

https://colab.research.google.com/drive/1l_tOX2wXK8LuZqxmPWaWfBJmlc6wmZdE?usp=sharing

SLT- Social support in HealthCare.pptx