MIMIC Visualization

Ankita Savaliya

AI In Healthcare Course at UT Austin

Coding environment and packages

- Google Colab
- MIMIC-III Demo Dataset
- Used Google Drive for saving and loading MIMIC CSV data
- Libraries Used: matplotlib, seaborn, plotly, WordCloud, altair
- GitHub and Google Colab Links:

https://github.com/AnkitaSavaliya/AIH/blob/main/MIMIC_Visualization.ipynb

https://colab.research.google.com/github/AnkitaSavaliya/AIH/blob/main/MIMIC_Visualization.ipynb

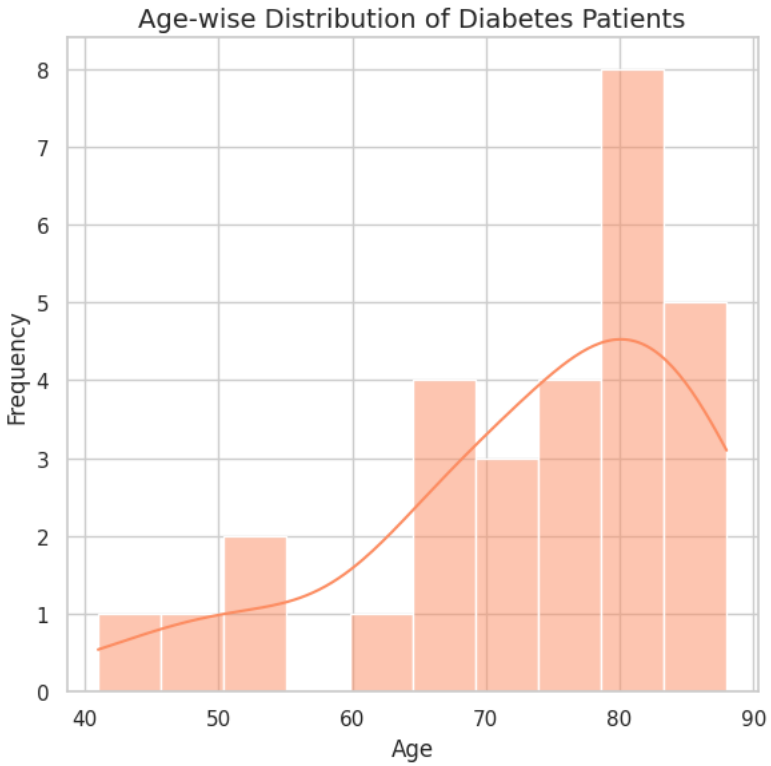
https://github.com/AnkitaSavaliya/AIH/blob/main/MIMIC%20Visualization.pptx

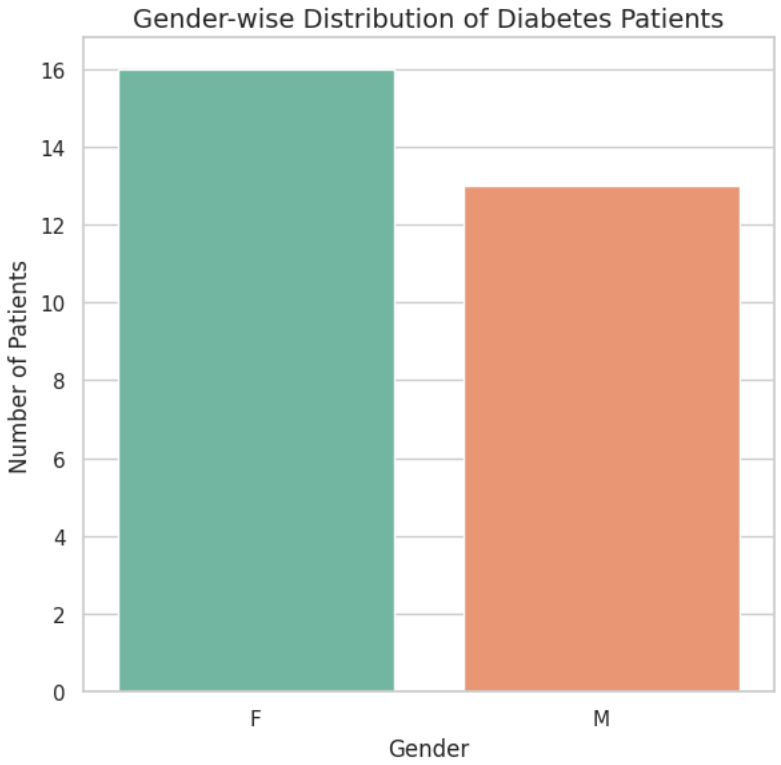
Data Load and Initial Setup

- Used PATIENTS .CSV, ADMISSIONS.CSV , DIAGNOSES_ICD.CSV, D_ICD_DIAGNOSES.CSV, ICUSTAYS.CSV, PRESCRIPTIONS.CSV etc.
- Created the initial dataframe using PATIENTS, ADMISSIONS, and DIAGNOSES_ICD.
- Merged the dataframe with other CSV files as needed for visualization requirements.

Visual 1- Age and Gender-wise Distribution of Diabetes Patients

This visualization represents diabetes patients, showcasing their age and gender distribution. It uses Seaborn to create a histogram for age-wise distribution and a barplot for gender-wise distribution.





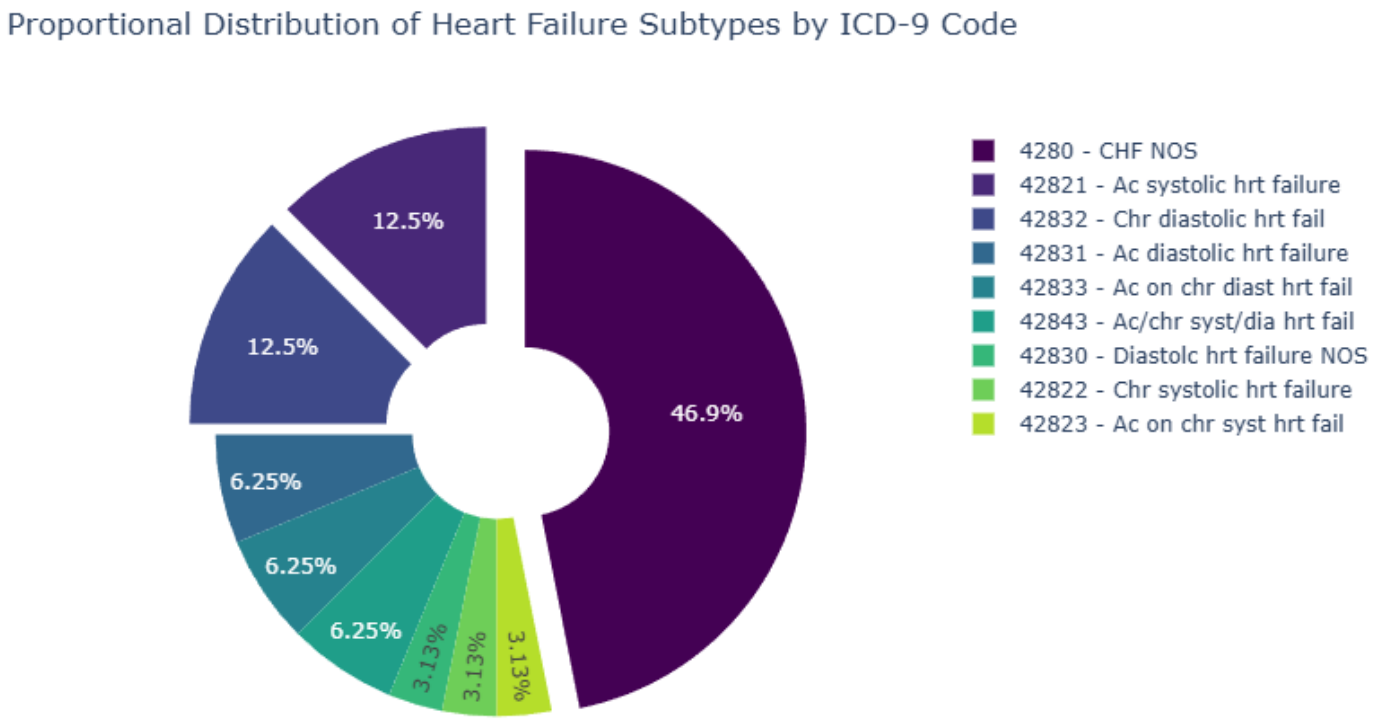
Visual 1 - How to recreate

```
# Filter for diabetes patients based on ICD-9 codes (250.x series)
diabetes_patients = merged_data_diagnosis[merged_data_diagnosis['icd9_code'].str.startswith('250')]

# Drop duplicate subject IDs to avoid double-counting patients
unique_diabetes_patients = diabetes_patients.drop_duplicates(subset='subject_id')

# Count gender distribution
gender_counts = unique_diabetes_patients['gender'].value_counts()

#set seaborn theme
sns.set_theme(style="whitegrid")
plt.figure(figsize=(12, 6))

# Age-wise distribution using histplot
plt.subplot(1, 2, 1)
sns.histplot(data=unique_diabetes_patients, x='age', bins=10, kde=True, color=sns.color_palette("Set2")[1])
plt.title("Age-wise Distribution of Diabetes Patients", fontsize=14)
plt.xlabel("Age", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(rotation=0)

# Gender-wise distribution using barplot
plt.subplot(1, 2, 2)
sns.barplot(x=gender_counts.index, y=gender_counts.values, hue=gender_counts.index, palette="Set2", legend=False)
plt.title('Gender-wise Distribution of Diabetes Patients', fontsize=14)
plt.xlabel('Gender', fontsize=12)
plt.ylabel('Number of Patients', fontsize=12)
plt.xticks(rotation=0)

# Adjust layout to avoid overlap
plt.tight_layout()
plt.show()
```

- Filters patient records based on ICD-9 codes (250.x series)
- Eliminates duplicate patients by their unique IDs.
- Counts gender distribution among unique patients.
- Plot histogram and barchart

Visual 2- Proportional Distribution of Heart Failure Subtypes by ICD-9 Code

This visualization is an interactive pie chart displaying the proportional distribution of heart failure subtypes.



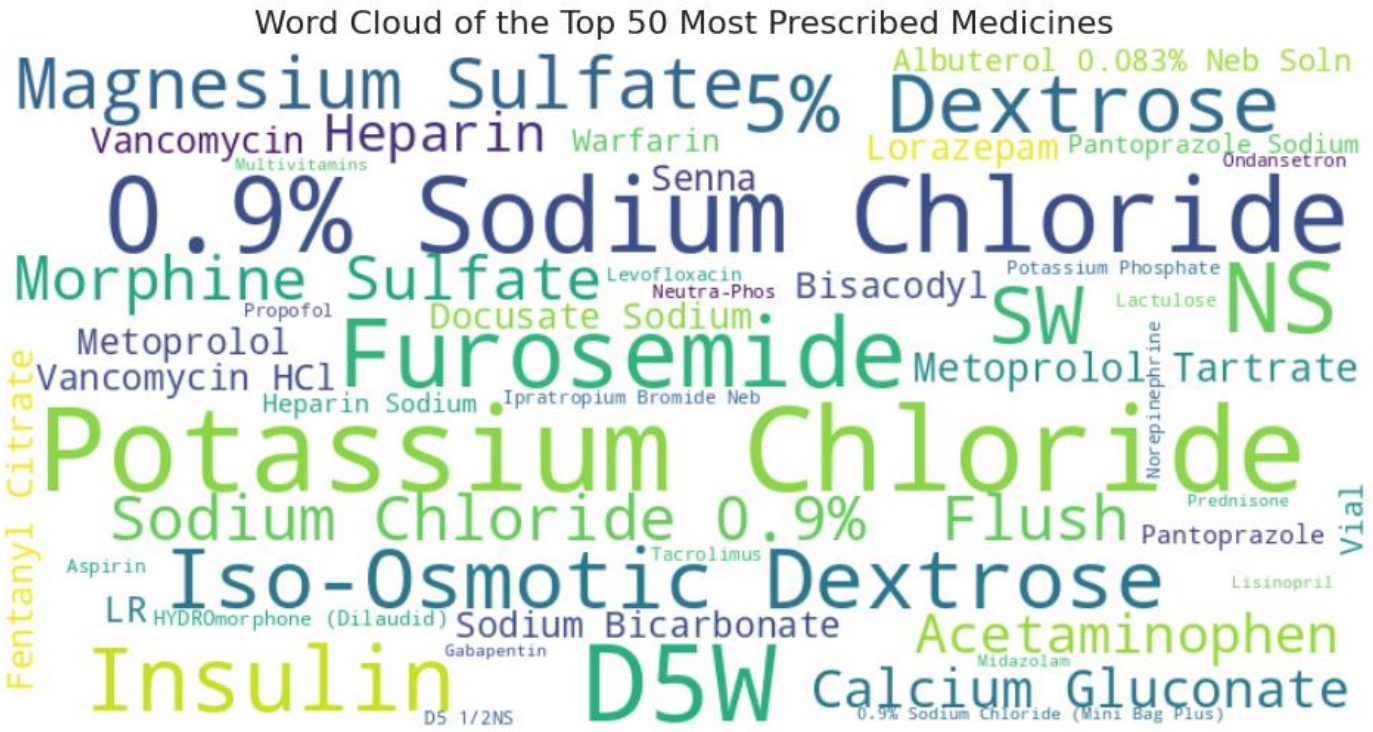
Visual 2 - How to recreate

```python
import plotly.express as px

# Filter for Heart patients based on ICD-9 codes (428.x series)
heart_patients = merged_data_diagnosis[merged_data_diagnosis['icd9_code'].str.startswith('428')]

# Drop duplicate subject IDs to avoid double-counting patients
unique_heart_patients = heart_patients.drop_duplicates(subset='subject_id')

# Prepare icd code - short description labels
icd9_counts = unique_heart_patients['icd9_code'].value_counts()
icd9_labels = [f"{code} - {desc}" for code, desc in zip(icd9_counts.index, icd9_counts.index.map(icd9_dict))]
icd9_data = pd.DataFrame({'ICD-9 Code': icd9_counts.index, 'Description': icd9_labels, 'Count': icd9_counts.values})
#print(icd9_data)

# Set pull value for slice effect
pull_values = [0.1 if i < 3 else 0 for i in range(len(icd9_data))]

# Create a Pie chart using Plotly Express with a modern color palette
fig = px.pie(icd9_data, names='Description', values='Count',
             title='Proportional Distribution of Heart Failure Subtypes by ICD-9 Code',
             color='Description',
             color_discrete_sequence=px.colors.sequential.Viridis)  # Modern Plasma color scheme

# Adding the slice effect
fig.update_traces(hole=0.3, pull=pull_values)
# Align title to center
fig.update_layout(title_x=0.4)
# Show the chart
fig.show()
```

- Extracts patient records with ICD-9 codes (428.x series for heart failure).
- Counts occurrences of each ICD-9 code and prepares labels.
- Creates an interactive pie chart using Plotly Express.

Visual 3 - Word Cloud of the Top 50 Most Prescribed Medicines

This visualization is a word cloud showcasing the top 50 most prescribed medications.



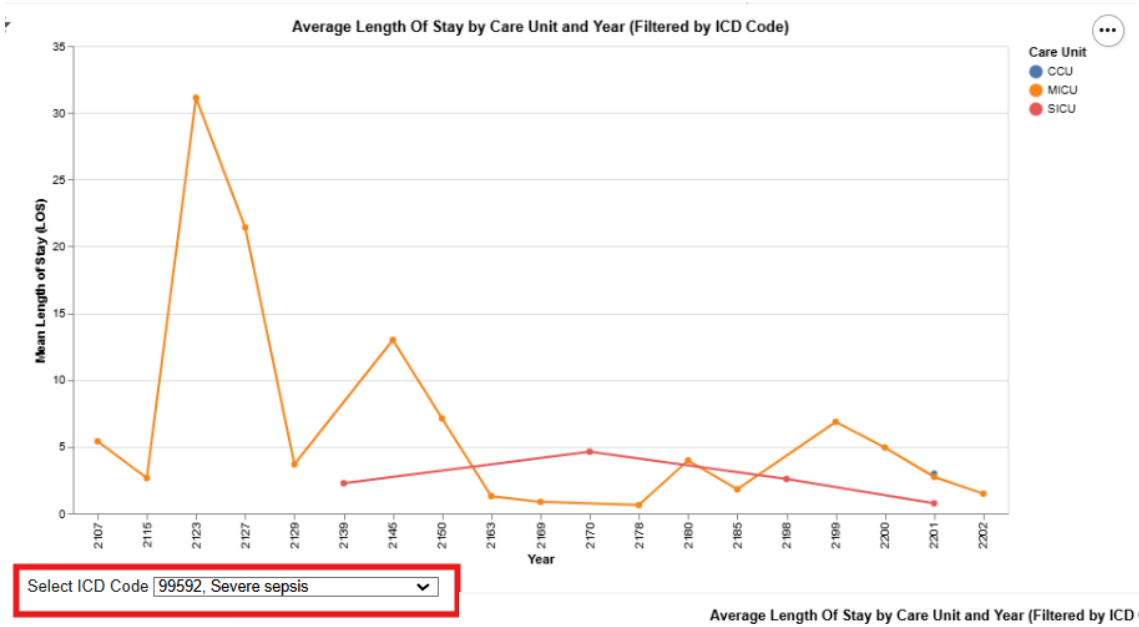
Visual 3 - How to recreate

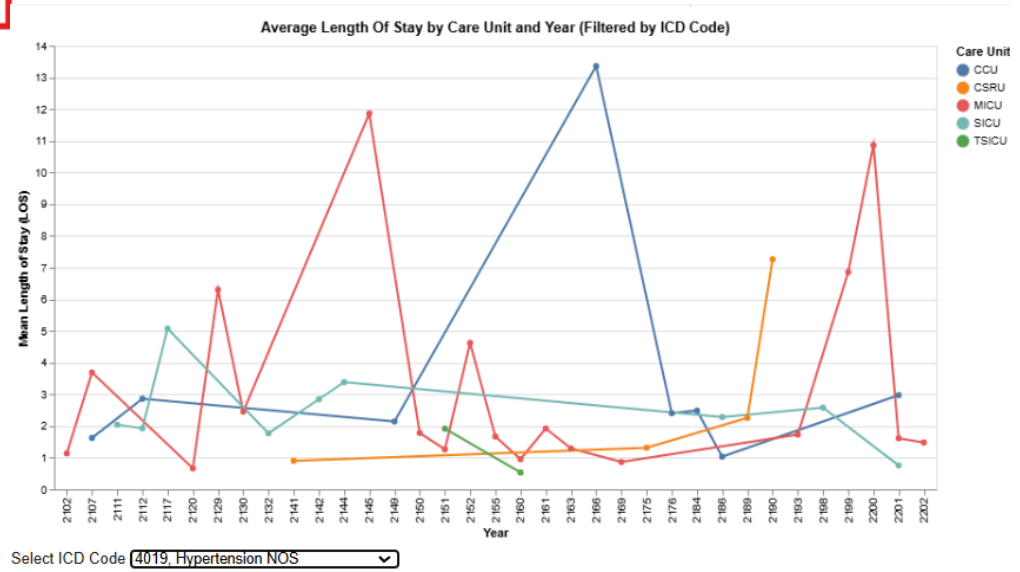
```
from wordcloud import WordCloud

# Extract the top 50 medication names and counts
medication_counts = prescriptions['drug'].value_counts(ascending=False)
top_prescribed_meds = medication_counts.head(50)
#print(medication_counts.head(5))

# Generate the word cloud using the top prescribed medications directly
wordcloud = WordCloud(width=800, height=400, background_color="white").generate_from_frequencies(top_prescribed_meds)

# Plot the word cloud
plt.figure(figsize=(12, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of the Top 50 Most Prescribed Medicines', fontsize=16)
plt.show()
```

- Extracts the top 50 most prescribed medication names and their counts from the dataset.
- Creates and display a word cloud using the frequency of these medications, with larger font sizes representing higher prescription counts.

Visual 4 - Average Length Of Stay by Care Unit and Year (Filtered by ICD Code)

This visualization is an interactive line chart created with Altair, analyzing the average length of stay (LOS) by care unit and year, dynamically filtered by ICD-9 codes.





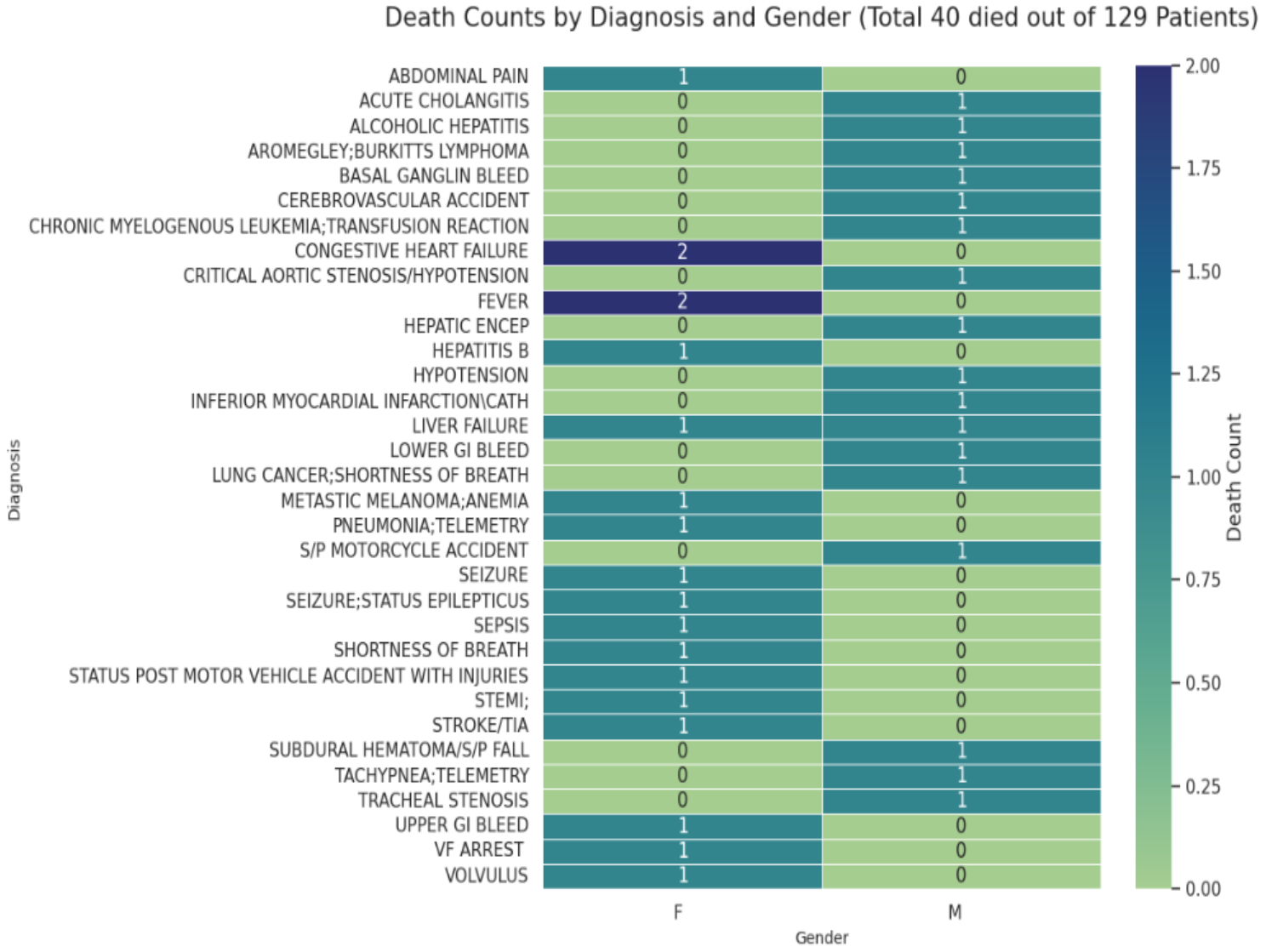
Visual 4 - How to recreate

```
import altair as alt

# Average Length of Stay by Care Unit and Year (Can be Filtered by ICD Code)

# Convert 'intime' and 'outtime' to datetime
icustays['intime'] = pd.to_datetime(icustays['intime'])
icustays['outtime'] = pd.to_datetime(icustays['outtime'])

# Extract the year from 'intime'
icustays['intime_year'] = icustays['intime'].dt.year

# Merge icustays with ICD-9 diagnosis codes
icustays_with_icd = icustays.merge(diagnoses, on='hadm_id', how='left')

# Calculate the mean LOS for each care unit and year
los_stats = icustays_with_icd.groupby(['first_careunit', 'intime_year', 'icd9_code'])['los'].mean().reset_index()

# Create a list of 'ICD9 code - short description' for each unique ICD9 code
icd_label_options = [(code, f" {icd9_dict.get(code, 'Unknown description')}") for code in los_stats['icd9_code'].unique()]

# Create a selection widget for ICD-9 codes
icd_selection = alt.selection_point(
    fields=['icd9_code'],
    bind=alt.binding_select(options=icd_label_options, name="Select ICD Code ")
)

# Create the Altair chart
chart = alt.Chart(los_stats).mark_line(point=True).encode(
    x=alt.X('intime_year:O', title='Year'),
    y=alt.Y('los:Q', title='Mean Length of Stay (LOS)'),
    color=alt.Color('first_careunit:N', title='Care Unit'),
    tooltip=['first_careunit', 'intime_year', 'los', 'icd9_code']  # Add tooltips for interactivity
).add_params(
    icd_selection
).transform_filter(
    icd_selection

).properties(
    title='Average Length Of Stay by Care Unit and Year (Filtered by ICD Code)',
    width=800,
    height=400
)

# Display the chart
chart
```

- Converts ICU stay times to datetime format and extracts the year and merges ICU stay data with ICD-9 diagnosis codes.
- Calculates the mean LOS for each care unit and year.
- Creates and display a line chart showing trends in LOS by care unit and year with selection box for ICD-9 code.

Visual 5 - Analysis of Death Counts by Diagnosis and Gender

This visualization represents deceased patients categorized by their diagnosis and gender.



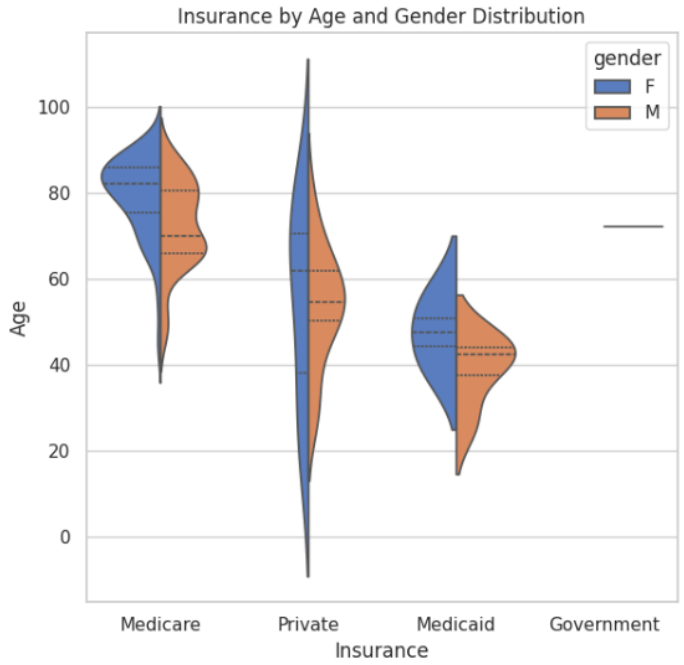
Visual 5 - How to recreate

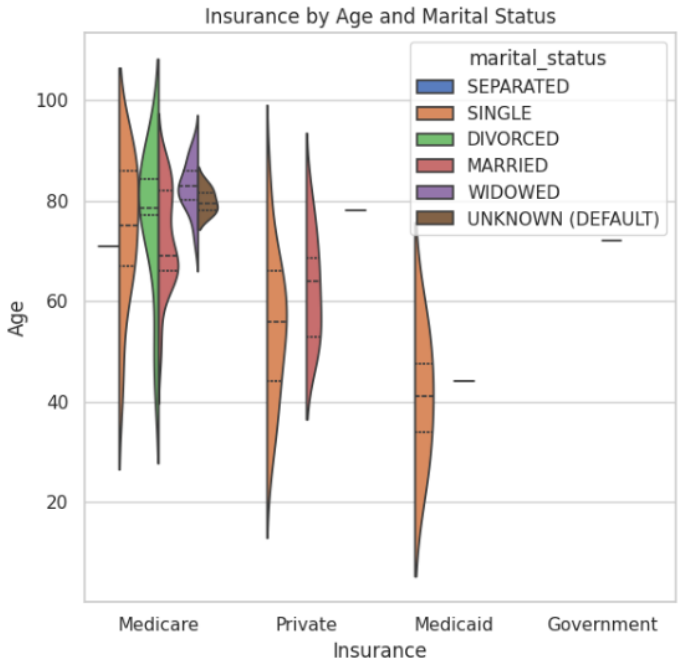
```python
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Filter data for deceased patients
deceased = merged_data[merged_data['discharge_location'] == 'DEAD/EXPIRED']

# Calculate total patients and deceased patients
total_patients = len(admissions)
total_deceased = len(admissions[admissions['discharge_location'] == 'DEAD/EXPIRED'])

# Group deceased data by diagnosis and gender
deaths_by_diagnosis = deceased.groupby(['diagnosis', 'gender']).size().reset_index(name='death_count')
deaths_by_diagnosis = deaths_by_diagnosis.sort_values(by='death_count', ascending=False)

# Prepare heatmap data
heatmap_data = deaths_by_diagnosis.pivot_table(
    index='diagnosis',
    columns='gender',
    values='death_count',
    aggfunc='sum',
    fill_value=0
)

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap="crest", cbar_kws={'label': 'Death Count'}, linewidths=0.5)

# Add title and labels
plt.title(f'Death Counts by Diagnosis and Gender (Total {total_deceased} died out of {total_patients} Patients)\n', fontsize=16)
plt.xlabel('Gender', fontsize=10)
plt.ylabel('Diagnosis', fontsize=10)
plt.xticks(rotation=0, ha='right')
plt.tight_layout()
plt.show()
```

- Filter the data for deceased patients.
- Aggregate the death counts by diagnosis and gender.
- Create and plot a heatmap to visually compare death counts across diagnoses and genders.

Visual 6 - Insurance Analysis based on Gender and Marital Status

This visualization represents the distribution of age across insurance types, segmented by gender and marital status. It uses Seaborn's violin plots for detailed representation of the data.





Visual 6 - How to build/recreate

```
# Set Seaborn theme
sns.set_theme(style="whitegrid")

# Create a figure and set its size
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.violinplot(data=merged_data, x='insurance', y='age', hue='gender', split=True, inner='quart', palette="muted")
plt.title("Insurance by Age and Gender Distribution")
plt.xlabel("Insurance")
plt.ylabel("Age")

plt.subplot(1, 2, 2)
sns.violinplot(data=merged_data, x='insurance', y='age', hue='marital_status', split=True, inner='quart', palette="muted")
plt.title("Insurance by Age and Marital Status")
plt.xlabel("Insurance")
plt.ylabel("Age")

# Adjust layout to avoid overlap
plt.tight_layout()
plt.show()
```

- Create and display violinplot showing age distribution by insurance type, split by gender.
- Create and display violinplot showing age distribution by insurance type, split by marital status