MIMIC v. 2.2 Database Dynamic Exploratory Data Analysis

Veronica Ramirez-Lopera

Objectives:

Main objective:

-To assess whether the MIMIC-IV database contains sufficient sequential ECGs to support a study on the dynamic aging of the heart in heart failure patients.

Secondary objectives:

1. To analyze the timing of ECGs during each hospitalization.
2. To visualize patient distribution by ECG frequency and hospitalization count.
3. To describe a plan for the research protocol modifications according to the ECG data.

Table of contents

Introduction

MIMIC-IV is a freely available electronic health record (EHR) dataset encompassing a decade of patient information (2008-2019) from Beth Israel Deaconess Medical Center [1]. It surpasses its predecessor, MIMIC-III, with a better structure and additional patient information [2, 3].

The dataset draws upon two primary sources: a comprehensive hospital-wide EHR system and an ICU-specific clinical information system [1]. Rigorous de-identification procedures ensure patient privacy while preserving the data's scientific integrity.

It includes vital signs, diagnoses, medications, procedures, and de-identified clinical notes [1]. Researchers leverage MIMIC-IV to investigate disease progression, train machine learning models, and develop clinical decision support tools [1, 2].

As a result, null dates of death indicate the patient was alive at least up to that time point. Inferences regarding patient death beyond one year cannot be made using MIMIC-IV (as per the PhyioNet website) [2]. The majority of patient death information is acquired from hospital records when the individual dies within the BIDMC or an affiliated institute.

MIMIC-IV also offers a dedicated module: MIMIC-IV ECG. This subset focuses specifically on diagnostic electrocardiograms (ECGs) [2]. It includes approximately 800,000 10-second ECG recordings from nearly 160,000 unique patients. Each ECG utilizes 12 leads and is sampled at 500 Hz [3].

Package requirements

Notes:

1- You need to install gcloud if you haven't already. Alternatively, you can use the provided csv files that contain the data.

In [ ]:
```
from google.cloud import bigquery
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np



# Formatting for my pandas dataframes, ignore:

pd.set_option('display.max_columns', None)  # Show all columns
pd.set_option('display.width', None)

# Construct a BigQuery client
client = bigquery.Client()
```

Data Importing

By using SQL, I will import the required data into the local environment and organize it in a pandas dataset. For the purpose of this analysis, I will need:

- subject_id: Unique identifier for each patient, used to link admissions and ECG records.- hadm_id: Identifier for each hospitalization event, allowing differentiation between multiple admissions for the same patient.
- admittime/dischtime: Admission and discharge timestamps to determine the hospitalization period and align ECG events within the correct timeframe.
- admission_type: Type of admission (e.g., emergency, elective) to categorize hospitalizations and assess their impact on ECG frequency.
- admission_location: The location or source of admission (e.g., ER, clinic).

- study_id: Unique identifier for each ECG study, used to track individual ECG events.
- file_name: Name of the ECG file, useful for accessing raw waveform data if needed for deeper analysis.
- ecg_time: Timestamp of when the ECG was performed, helps with determining its relation to the admission period.
- path: File path to the ECG data, allowing retrieval of the raw ECG waveform files for further analysis if required.

In [ ]:

```
# Data importing

# Fetch admissions data
'''
With the following code i will fetch both the ecg paths AND the admissions table, b
ecg was in the context of an admission this is to ensure that the ecgs were not tak
same visit, and at two different times in the patients life, which can showcase the
changes in the heart that ocur during hospitalizations.

'''


admissions_query = '''
SELECT
  a.subject_id,
  a.hadm_id,
  a.admittime,
  a.dischtime,
  a.admission_type,
  a.admission_location,
  e.study_id,
  e.file_name,
  e.ecg_time,
  e.path
FROM
  `physionet-data.mimiciv_hosp.admissions` AS a
LEFT JOIN
  `physionet-data.mimiciv_ecg.record_list` AS e
ON
  a.subject_id = e.subject_id
  AND e.ecg_time BETWEEN a.admittime AND a.dischtime

'''


# Fetch the admissions data as a DataFrame
admissions = client.query(admissions_query).to_dataframe(create_bqstorage_client=Tr

# Save the admissions data to a CSV file

path = r'C:\Users\Vero Ramirez\Desktop\CODE\CODE\admissions.csv'
admissions.to_csv(path, index=False)

print('\nAdmissions data:')
print(admissions.head())
```

```
c:\Users\Vero Ramirez\AppData\Local\Programs\Python\Python312\Lib\site-packages\goog
le\cloud\bigquery\table.py:1727: UserWarning: BigQuery Storage module not found, fet
ch data with the REST endpoint instead.
  warnings.warn(
Admissions data:
   subject_id   hadm_id           admittime           dischtime  \
0    10106244  26713233 2147-05-09 10:34:00 2147-05-12 13:43:00
1    10106244  26713233 2147-05-09 10:34:00 2147-05-12 13:43:00
2    10106244  26713233 2147-05-09 10:34:00 2147-05-12 13:43:00
3    10106244  26713233 2147-05-09 10:34:00 2147-05-12 13:43:00
4    13700703  20448599 2172-09-25 01:01:00 2172-10-03 13:25:00

      admission_type  admission_location  study_id file_name  \
0       DIRECT EMER.  PHYSICIAN REFERRAL  49164244  49164244
1       DIRECT EMER.  PHYSICIAN REFERRAL  44859244  44859244
2       DIRECT EMER.  PHYSICIAN REFERRAL  40600970  40600970
3       DIRECT EMER.  PHYSICIAN REFERRAL  48644999  48644999
4  OBSERVATION ADMIT      EMERGENCY ROOM  45997419  45997419

             ecg_time                                     path
0 2147-05-09 11:06:00  files/p1010/p10106244/s49164244/49164244
1 2147-05-09 16:53:00  files/p1010/p10106244/s44859244/44859244
2 2147-05-10 08:17:00  files/p1010/p10106244/s40600970/40600970
3 2147-05-11 07:27:00  files/p1010/p10106244/s48644999/48644999
4 2172-09-26 11:00:00  files/p1370/p13700703/s45997419/45997419
```

In [ ]:

```
# Load the admissions data from the CSV file,

'''
Since running the query is time-consuming, I will save the data to a CSV file and l
This approach will also help avoid additional costs and allow for more efficient wo
'''
path = r'C:\Users\Vero Ramirez\Desktop\CODE\CODE\admissions.csv'
admissions = pd.read_csv(path)
```

```
C:\Users\Vero Ramirez\AppData\Local\Temp\ipykernel_10308\1045624533.py:8: DtypeWarni
ng: Columns (8,9) have mixed types. Specify dtype option on import or set low_memory
=False.
  admissions = pd.read_csv(path)
```

In [ ]:

```
# First, lets drop any hospitalization that does not have an ecg
'''
We will drop any hospitalization that does not have an ECG, as the purpose of this
analyze whether there is enough data to analyze the dynamic changes in the heart du
hospitalizations

'''


admissions = admissions.dropna(subset=['path'])
```

Data filtering and visualization

At this point, we have a dataset containing all patients who were hospitalized at least once and received an ECG during their stay, regardless of their diagnosis. Our goal now is to determine the number of hospitalizations for each patient and the number of ECGs recorded during each hospitalization.

In [ ]:
```
# Lets drop the patients that only had one admission (the combination of subject_id
# is unique)

admission_counts = admissions.groupby('subject_id')['hadm_id'].nunique()
admission_counts = admission_counts[admission_counts > 1]
admissions = admissions[admissions['subject_id'].isin(admission_counts.index)]

admission_counts = admission_counts.reset_index()
admission_counts.columns = ['subject_id', 'Unique admissions']

print(admission_counts.head())
print(admissions.head())
```

```
   subject_id  Unique admissions
0    10000935                  3
1    10000980                  5
2    10001401                  2
3    10001877                  2
4    10001884                 12
   subject_id   hadm_id            admittime            dischtime  \
0    10106244  26713233  2147-05-09 10:34:00  2147-05-12 13:43:00
1    10106244  26713233  2147-05-09 10:34:00  2147-05-12 13:43:00
2    10106244  26713233  2147-05-09 10:34:00  2147-05-12 13:43:00
3    10106244  26713233  2147-05-09 10:34:00  2147-05-12 13:43:00
4    13700703  20448599  2172-09-25 01:01:00  2172-10-03 13:25:00

      admission_type  admission_location    study_id   file_name  \
0       DIRECT EMER.  PHYSICIAN REFERRAL  49164244.0  49164244.0
1       DIRECT EMER.  PHYSICIAN REFERRAL  44859244.0  44859244.0
2       DIRECT EMER.  PHYSICIAN REFERRAL  40600970.0  40600970.0
3       DIRECT EMER.  PHYSICIAN REFERRAL  48644999.0  48644999.0
4  OBSERVATION ADMIT      EMERGENCY ROOM  45997419.0  45997419.0

              ecg_time                                        path
0  2147-05-09 11:06:00  files/p1010/p10106244/s49164244/49164244
1  2147-05-09 16:53:00  files/p1010/p10106244/s44859244/44859244
2  2147-05-10 08:17:00  files/p1010/p10106244/s40600970/40600970
3  2147-05-11 07:27:00  files/p1010/p10106244/s48644999/48644999
4  2172-09-26 11:00:00  files/p1370/p13700703/s45997419/45997419
```

In [ ]:
```
# Lets see how many unique patients we have left

unique_patients = admissions['subject_id'].nunique()

print( f'After dropping the patients that only had one hospitalization, there are \
{unique_patients} unique patients in the dataset with more than one admission, that
one ECG during their hospitalization.')
```

```
After dropping the patients that only had one hospitalization, there are 26769 uniqu
e patients in the dataset with more than one admission, that have at least one ECG d
uring their hospitalization.
```

In [ ]:

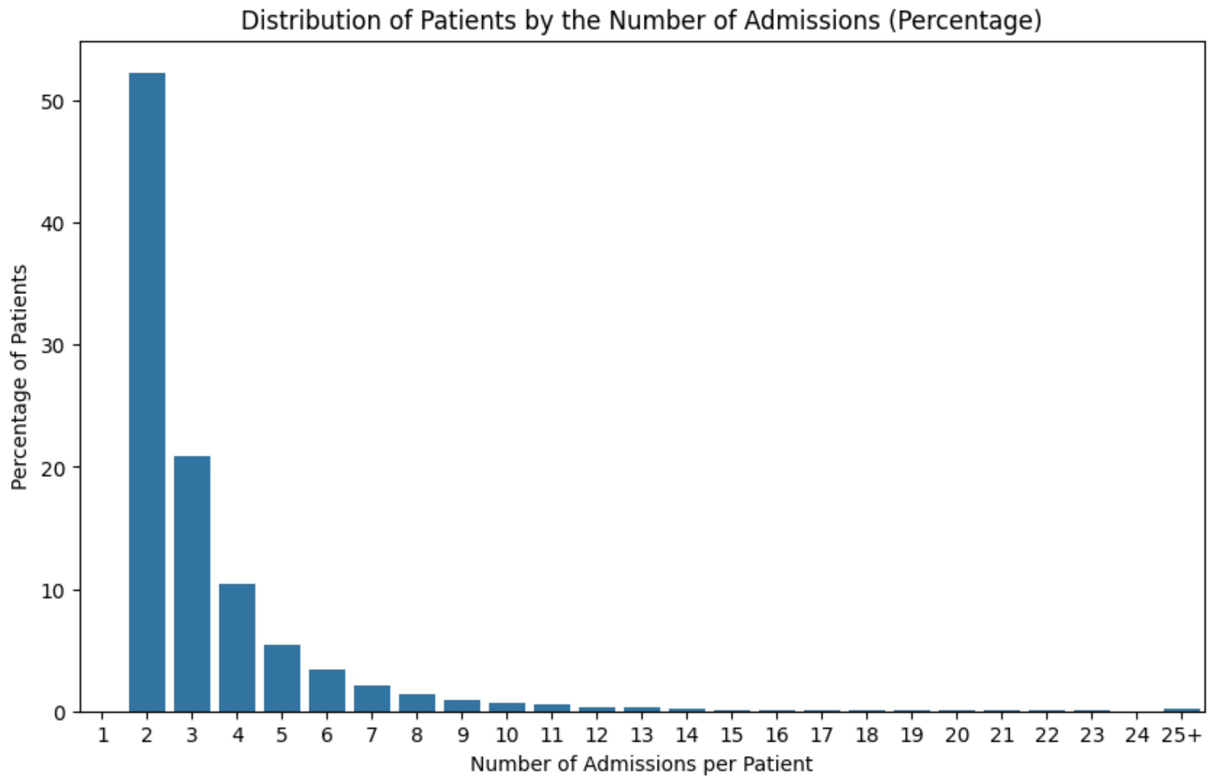
```
# Lets summarize how many patients have more than one admission and how many admiss
'''
It is important to see the distribution of patients by the number of admissions the
will allow us to plan ahead what to do with our research

'''
admission_counts = admissions.groupby('subject_id')['hadm_id'].nunique()
patients_per_admission = admission_counts.value_counts().sort_index()
patients_per_admission = patients_per_admission.reset_index()
patients_per_admission.columns = ['Number of admissions per patient',
                                   'Number of patients']

# Display the results
print(patients_per_admission.head())
print(admission_counts.describe())
```

```
   Number of admissions per patient  Number of patients
0                                 2               14003
1                                 3                5580
2                                 4                2796
3                                 5                1470
4                                 6                 907
count    26769.000000
mean         3.355224
std          2.700377
min          2.000000
25%          2.000000
50%          2.000000
75%          4.000000
max         60.000000
Name: hadm_id, dtype: float64
```

In [ ]:

```
# Lets plot the distribution of patients by the number of admissions
'''
There are multiple single patients that have more than 25 unique stays,
Lets combine categories with more than 24 admissions to simplify the visualization.
'''
patients_per_admission['Number of admissions per patient'] = \
    patients_per_admission['Number of admissions per patient'].apply(
        lambda x: '25+' if x > 24 else str(x))
patients_per_admission = \
        patients_per_admission.groupby('Number of admissions per patient').sum().re

# Calculate the percentage of patients for each category
total_patients = patients_per_admission['Number of patients'].sum()
patients_per_admission['Percentage of patients'] = \
    (patients_per_admission['Number of patients'] / total_patients) * 100
patients_per_admission['Number of admissions per patient'] = pd.Categorical(
    patients_per_admission['Number of admissions per patient'],
    categories=[str(i) for i in range(1, 25)] + ['25+'],
    ordered=True)

# Plot the distribution
plt.figure(figsize=(10, 6))
sns.barplot(x='Number of admissions per patient', y='Percentage of patients',
            data=patients_per_admission)
```

```
plt.title('Distribution of Patients by the Number of Admissions (Percentage)')
plt.ylabel('Percentage of Patients')
plt.xlabel('Number of Admissions per Patient')
plt.show()
```



The data shows that most patients have two admissions, with the highest number of admissions per individual reaching 60. On average, each patient has 3.3 admissions with associated ECGs, and the standard deviation is 2.7. To better reflect the aging process, I am interested in analyzing only the first and last available ECGs. Therefore, we will filter out all intermediate ECGs.

In [ ]:
```
# Lets analyze the timing of ECGs during each hospitalization
'''
We will keep the first ECG for the first hospitalization and the last ECG for the l
hospitalization of each patient, as this will allow us to analyze the dynamic chang
'''
admissions['ecg_time'] = pd.to_datetime(admissions['ecg_time'])
admissions = admissions.sort_values(by=['subject_id', 'ecg_time'])
first_and_last_ecgs = admissions.groupby('subject_id').agg({'ecg_time': ['first', '
first_and_last_ecgs.columns = ['first_ecg_time', 'last_ecg_time']
first_and_last_ecgs = first_and_last_ecgs.reset_index()
print(first_and_last_ecgs)
```

```
       subject_id      first_ecg_time       last_ecg_time
0        10000935 2187-07-11 21:51:00 2187-10-10 20:39:00
1        10000980 2188-01-03 19:56:00 2193-08-15 16:41:00
2        10001401 2131-10-01 12:39:00 2131-11-14 03:16:00
3        10001877 2149-05-25 02:44:00 2150-11-21 23:10:00
4        10001884 2125-10-27 09:17:00 2131-01-13 08:25:00
...           ...                 ...                 ...
26764    19998497 2139-07-02 07:53:00 2145-07-30 19:21:00
26765    19998562 2166-03-16 19:13:00 2166-04-16 10:27:00
26766    19998770 2179-04-27 08:37:00 2182-08-20 12:43:00
26767    19999287 2191-12-30 03:53:00 2197-08-05 02:06:00
26768    19999840 2164-07-27 08:37:00 2164-09-17 11:31:00

[26769 rows x 3 columns]
```

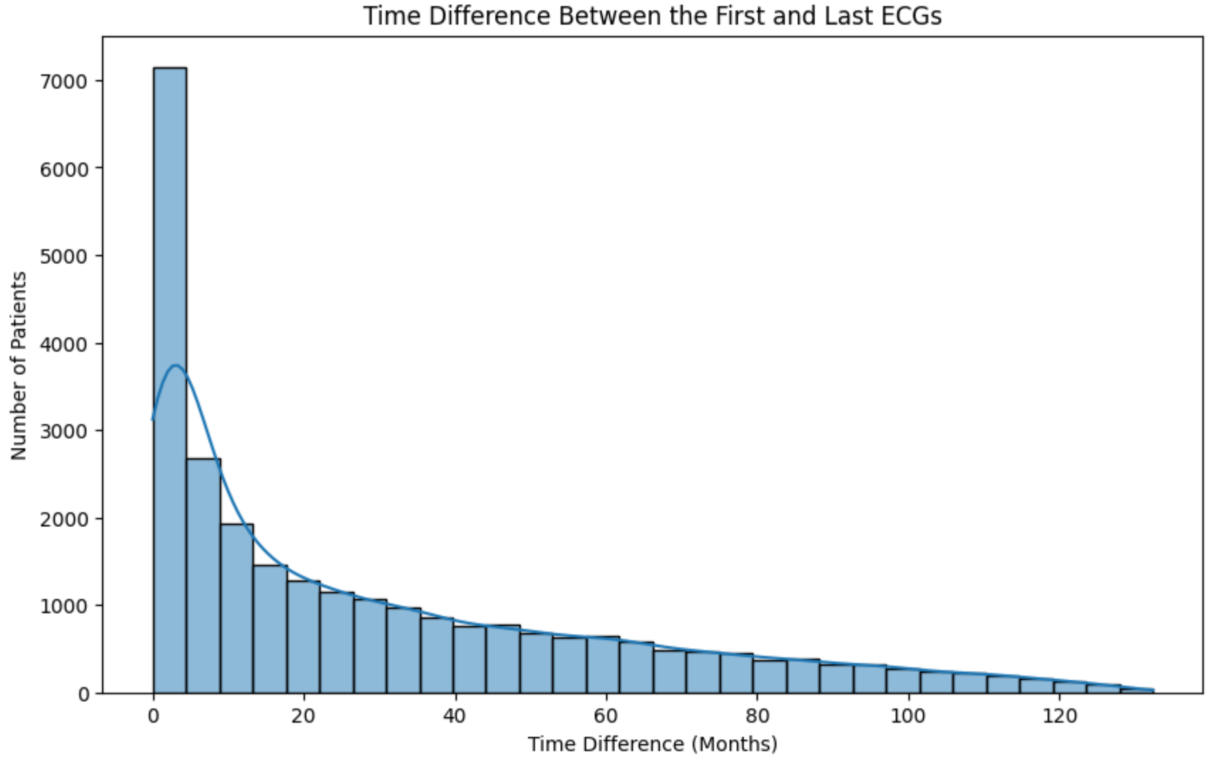
In [ ]:

```
# Lets calculate the time between the first and last ECG for each patient
'''
We will calculate the time between the first and last ECG for each patient, as this
set a time frame for our analysis
'''

# Calculate the time difference in time between the first and last ECGs
first_and_last_ecgs['time_difference'] = \
(first_and_last_ecgs['last_ecg_time'] - first_and_last_ecgs['first_ecg_time']).dt.t

# Convert the time difference to months
first_and_last_ecgs['time_difference'] = first_and_last_ecgs['time_difference'] / (
print(first_and_last_ecgs.head(), first_and_last_ecgs['time_difference'].describe()
```

```
   subject_id      first_ecg_time       last_ecg_time  time_difference
0    10000935 2187-07-11 21:51:00 2187-10-10 20:39:00         3.031667
1    10000980 2188-01-03 19:56:00 2193-08-15 16:41:00        68.362153
2    10001401 2131-10-01 12:39:00 2131-11-14 03:16:00         1.453634
3    10001877 2149-05-25 02:44:00 2150-11-21 23:10:00        18.195046
4    10001884 2125-10-27 09:17:00 2131-01-13 08:25:00        63.465463 count    2676
9.000000
mean        30.151750
std         31.673684
min          0.000000
25%          3.854606
50%         18.186042
75%         48.539190
max        132.387731
Name: time_difference, dtype: float64
```

In [ ]:

```
# Lets plot the distribution of the time difference between the first and last ECGs

plt.figure(figsize=(10, 6))
sns.histplot(first_and_last_ecgs['time_difference'], bins=30, kde=True)
plt.title('Time Difference Between the First and Last ECGs')
plt.xlabel('Time Difference (Months)')
plt.ylabel('Number of Patients')
plt.show()
```



The average time between the first and last ECGs is 30 months, with a standard deviation of 31 months. The minimum time is 0 months, and the maximum is 132 months, equivalent to 11 years.

With multiple patients having ECGs recorded across more than one hospitalization, we are well-positioned to analyze dynamic changes in the heart over time. Our next step is to examine the heart failure diagnoses to determine if we have sufficient information to develop a profile for these patients.

In [ ]:
```
# Lets create a dataframe with the icd_codes and the icd guide
'''
We will fetch the ICD guide and the ICD data from the hosp module
'''
icd_query_guide = '''
SELECT
icd_code,
icd_version,
long_title
FROM
    `physionet-data.mimiciv_hosp.d_icd_diagnoses`
'''

icd_query = '''
SELECT
subject_id,
hadm_id,
icd_code,
icd_version,
seq_num
FROM
```

```
    `physionet-data.mimiciv_hosp.diagnoses_icd`

'''
#Fetch the data as a DataFrame
icd_guide = client.query(icd_query_guide).to_dataframe(create_bqstorage_client=True
icd = client.query(icd_query).to_dataframe(create_bqstorage_client=True)

# Display the first few rows of the DataFrames
print('\nICD guide data:')
print(icd_guide.head())

print('\nICD data:')
print(icd.head())
```

```
c:\Users\Vero Ramirez\AppData\Local\Programs\Python\Python312\Lib\site-packages\goog
le\cloud\bigquery\table.py:1727: UserWarning: BigQuery Storage module not found, fet
ch data with the REST endpoint instead.
  warnings.warn(
ICD guide data:
  icd_code  icd_version                              long_title
0     0010            9         Cholera due to vibrio cholerae
1     0011            9  Cholera due to vibrio cholerae el tor
2     0019            9                Cholera, unspecified
3     0020            9                       Typhoid fever
4     0021            9                  Paratyphoid fever A

ICD data:
   subject_id   hadm_id icd_code  icd_version  seq_num
0    10000980  25242409    72992            9       30
1    10000980  25242409    E8497            9       34
2    10002428  28662225    78097            9       26
3    10003400  26467376    V8532            9       29
4    10004401  29988601    04111            9       28
```

In [ ]:
```
# Lets merge the icd data with the icd guide data
'''
This will allow us to decipher the icd codes and see the different diagnosis for th
more than one admission
'''
icd_merged = pd.merge(icd, icd_guide, on=['icd_code', 'icd_version'], how='inner')

# Save the merged ICD data to a CSV file so we can load it later without running th
path = r'C:\Users\Vero Ramirez\Desktop\CODE\CODE\icd_merged.csv'
icd_merged.to_csv(path, index=False)
```

In [ ]:
```
# Load the merged ICD data from the CSV file
path = r'C:\Users\Vero Ramirez\Desktop\CODE\CODE\icd_merged.csv'
icd_merged = pd.read_csv(path)
```

In [ ]:
```
# Lets filter the icd data to only include heart failure diagnosis

'''
We will filter the icd data to only include heart failure diagnosis. The codes for
were taken from Yang et al. (2019) [4]
'''
heart_failure_icd9 = [
```

```
    '428', '4280', '4281', '4282', '42820', '42821', '42822', '42823',
    '4283', '42830', '42831', '42832', '42833', '4284', '42840', '42841',
    '42842', '42843', '4289', '40201', '40211', '40291', '40401',
    '40403', '40411', '40413', '40491', '40493']

heart_failure_icd10 = [
    'I50', 'I501', 'I502', 'I5020', 'I5021', 'I5022', 'I5023',
    'I503', 'I5030', 'I5031', 'I5032', 'I5033', 'I504', 'I5040',
    'I5041', 'I5042', 'I5043', 'I508', 'I5081', 'I50810', 'I50811',
    'I50812', 'I50813', 'I50814', 'I5082', 'I5083', 'I5084', 'I5089',
    'I509']

heart_failure_codes = heart_failure_icd9 + heart_failure_icd10
heart_failure_data = icd_merged[icd_merged['icd_code'].isin(heart_failure_codes)]

heart_failure_data
```

Out[ ]:

	subject_id	hadm_id	icd_code	icd_version	seq_num	long_title
162	10258162	21744088	4280	9	29	Congestive heart failure, unspecified
276	10494089	25917748	4280	9	27	Congestive heart failure, unspecified
456	10800546	25150796	4280	9	27	Congestive heart failure, unspecified
539	10928511	24107609	4280	9	29	Congestive heart failure, unspecified
707	11165060	29991539	4280	9	27	Congestive heart failure, unspecified
...	...	...	...	...	...	...
4754912	11132535	20379929	I50810	10	25	Right heart failure, unspecified
4755628	15618712	26857232	I509	10	25	Heart failure, unspecified
4755658	15788406	22089377	I509	10	25	Heart failure, unspecified
4756053	18157237	28285611	I50810	10	25	Right heart failure, unspecified
4756289	19770161	21099154	I509	10	25	Heart failure, unspecified

91805 rows × 6 columns

In [ ]:

```
# Lets merge the heart failure data with the admissions data

'''
We will only include the patients that have a heart failure diagnosis in the admiss
patients,but just for the first and last hospitalization corresponding to the first
of each patient.
'''
```

```python
#First, lets drop any hospitalization that is not the first or last hospitalization
admissions_merged = pd.merge(admissions, first_and_last_ecgs, on='subject_id', how=

# Then, lets merge the heart failure data with the admissions data
heart_failure_admissions = \
    pd.merge(admissions_merged, heart_failure_data, on=['subject_id', 'hadm_id'], h

#Now, lets merge the heart failure data with the admissions data
heart_failure_admissions = \
    pd.merge(admissions_merged, heart_failure_data, on=['subject_id', 'hadm_id'], h
print(heart_failure_admissions.head() )
```

```
   subject_id   hadm_id           admittime           dischtime  \
0    10000980  29654838 2188-01-03 17:41:00 2188-01-05 17:30:00
1    10000980  29654838 2188-01-03 17:41:00 2188-01-05 17:30:00
2    10000980  29654838 2188-01-03 17:41:00 2188-01-05 17:30:00
3    10000980  29654838 2188-01-03 17:41:00 2188-01-05 17:30:00
4    10000980  26913865 2189-06-27 07:38:00 2189-07-03 03:00:00

  admission_type admission_location    study_id   file_name  \
0       EW EMER.     EMERGENCY ROOM  47004256.0  47004256.0
1       EW EMER.     EMERGENCY ROOM  47004256.0  47004256.0
2       EW EMER.     EMERGENCY ROOM  49560547.0  49560547.0
3       EW EMER.     EMERGENCY ROOM  49560547.0  49560547.0
4       EW EMER.     EMERGENCY ROOM  49245181.0  49245181.0

             ecg_time                                     path  \
0 2188-01-03 19:56:00  files/p1000/p10000980/s47004256/47004256
1 2188-01-03 19:56:00  files/p1000/p10000980/s47004256/47004256
2 2188-01-04 16:43:00  files/p1000/p10000980/s49560547/49560547
3 2188-01-04 16:43:00  files/p1000/p10000980/s49560547/49560547
4 2189-06-27 07:58:00  files/p1000/p10000980/s49245181/49245181

       first_ecg_time       last_ecg_time  time_difference icd_code  \
0 2188-01-03 19:56:00 2193-08-15 16:41:00        68.362153     4280
1 2188-01-03 19:56:00 2193-08-15 16:41:00        68.362153    42833
2 2188-01-03 19:56:00 2193-08-15 16:41:00        68.362153     4280
3 2188-01-03 19:56:00 2193-08-15 16:41:00        68.362153    42833
4 2188-01-03 19:56:00 2193-08-15 16:41:00        68.362153     4280

   icd_version  seq_num                                long_title
0            9       10      Congestive heart failure, unspecified
1            9        1  Acute on chronic diastolic heart failure
2            9       10      Congestive heart failure, unspecified
3            9        1  Acute on chronic diastolic heart failure
4            9        6      Congestive heart failure, unspecified
```

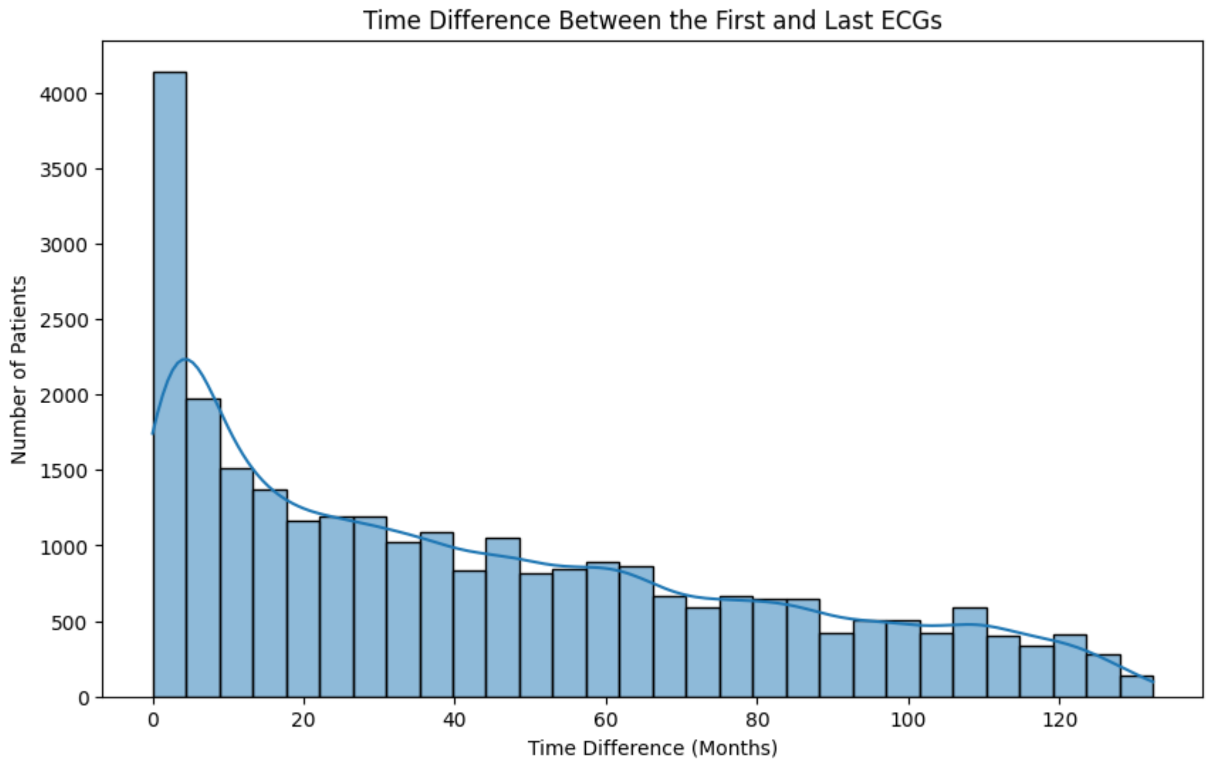
In [ ]:

```python
# Now, lets summarize the time difference between the first and last ECGs for patie
# failure

heart_failure_admissions['time_difference'].describe()
```

```
Out[ ]: count    114291.000000
        mean         40.435676
        std          35.290423
        min           0.018032
        25%           8.476065
        50%          32.257384
        75%          64.031412
        max         132.387731
        Name: time_difference, dtype: float64
```

- The time difference between the first and last ECGs for patients with heart failure is similar to the overall time difference for all patients.
- The mean time difference is approximately 40 months, with a standard deviation of 35 months.
- The minimum time difference is less than 1 month, and the maximum time difference is 132 months.

```
In [ ]: # Lets plot the distribution of the time difference between the first and last ECGs

        # Only keep one register per admission
        heart_failure_admissions_plot = \
        heart_failure_admissions.drop_duplicates(subset=['subject_id', 'hadm_id'])


        plt.figure(figsize=(10, 6))
        sns.histplot(heart_failure_admissions_plot['time_difference'], bins=30, kde=True)
        plt.title('Time Difference Between the First and Last ECGs')
        plt.xlabel('Time Difference (Months)')
        plt.ylabel('Number of Patients')
        plt.show()
```



In [ ]:
```
# How many patients have a time difference thats longer than 1 year?

heart_failure_admissions_long = \
heart_failure_admissions[heart_failure_admissions['time_difference'] > 12]
number_of_patients = heart_failure_admissions_long['subject_id'].nunique()
print(f'There are {number_of_patients} patients with a time difference between the
ECGs longer than 1 year.')
```

```
There are 6446 patients with a time difference between the first and last ECGs longe
r than 1 year.
```

Similar to the previous plot, the distribution of time differences between the first and last ECGs is right-skewed. Most patients have a time difference of less than 10 months, yet there are thousands of patients with differences exceeding 12 months. This indicates a substantial dataset available for analyzing the dynamic changes in the heart over time.

In [ ]:
```
# lets see how many unique patients we have left

'''
These patients have the following characteristics:
- They have more than one admission distributed over time
- They have at least one ECG during their hospitalization
- They have a heart failure diagnosis during each of their hospitalizations
'''

unique_patients = heart_failure_admissions['subject_id'].nunique()
print(f'There are {unique_patients} unique patients in the dataset with more than o
that have at least one ECG during their hospitalization and a heart failure diagnos
hospitalization.')
```

```
There are 10158 unique patients in the dataset with more than one admission, that ha
ve at least one ECG during their hospitalization and a heart failure diagnosis durin
g each hospitalization.
```

Conclusions and plan

- The MIMIC database contains a significant number of sequential ECGs recorded at various times of an individuals life.
- There are 6,400 patients with ECGs showing a time gap of more than one year between recordings.
- Aging profiles could be analyzed for patients, especially those diagnosed with heart failure. While this analysis won't establish causation between the chronic disease and aging, it can provide insights into how heart failure impacts biological age, as estimated using deep learning techniques.

Possible Plan Modification:

- We could analyze the heart profile changes over a one-year period for patients diagnosed with heart failure and compare these changes to a three-year mortality rate. Although we cannot establish causality due to the lack of information on other exposures between hospitalizations, this approach could offer insights into how heart aging progresses in this population. Additionally, it might reveal whether a greater difference in aging profiles is associated with mortality among hospitalized patients.

References

1. Johnson, Alistair, et al. "MIMIC-IV: A freely accessible electronic health record dataset." Schnpj (2021): 36596836. PubMed MIMIC-IV v2.2. PhysioNet
2. Johnson, A., Bulgarelli, L., Pollard, T., Horng, S., Celi, L. A., & Mark, R. (2023). MIMIC-IV (version 2.2). PhysioNet. https://doi.org/10.13026/6mm1-ek67.
3. Gow, B., Pollard, T., Nathanson, L. A., Johnson, A., Moody, B., Fernandes, C., Greenbaum, N., Waks, J. W., Eslami, P., Carbonati, T., Chaudhari, A., Herbst, E., Moukheiber, D., Berkowitz, S., Mark, R., & Horng, S. (2023). MIMIC-IV-ECG: Diagnostic Electrocardiogram Matched Subset (version 1.0). PhysioNet. https://doi.org/10.13026/4nqg-sb35.
4. Fan J, Li X, Yu X, Liu Z, Jiang Y, Fang Y, et al. Global Burden, Risk Factor Analysis, and Prediction Study of Ischemic Stroke, 1990–2030. Neurology [Internet]. 2023 Jul 11 [cited 2024 Jun 30];101(2). Available from: https://www.neurology.org/doi/10.1212/WNL.0000000000207387
5. Yang X, Gong Y, Waheed N, March K, Bian J, Hogan WR, et al. Identifying Cancer Patients at Risk for Heart Failure Using Machine Learning Methods. AMIA Annu Symp Proc. 2019;2019:933-41.