NE_inmates_demographics_1

March 8, 2021

0.1 Dataset

The data being used was from inmate databases aquired from the Nebraska Department of Corrections Public Records. https://dcs-inmatesearch.ne.gov/Corrections/COR_download.htm

The objective is to combine the active and complete databases from the Nebraska Department of Corrections, delete unnesscary rows, and prepare the new dataset for analysis

0.2 1. Load the Data

Load the data using Pandas.

Pandas 'ExcelFile' will load the data as a Pandas DataFrame object.

[1]:
```
import numpy as np
import pandas as pd

xls = pd.ExcelFile('inmateDB_updated.xlsx')
df = pd.read_excel(xls, 'Record Type 1')
```

0.3 2. Check the First Few Rows of Data

The DataFrame's first five rows can be viewed using the .head() method

[2]:
```
df.head()
```

[2]:

	ID NUMBER	COMMITTED LAST NAME	FIRST NAME	MIDDLE NAME	NAME EXTENSION	\
0	1702	CLIFFORD	BRADLEY	NaN		
1	6145	KANE	THOMAS	NaN		
2	6452	ATKINS	LARRY	NaN		
3	12444	SHANEYFELT	CHARLEY	NaN		
4	15379	BEADES	JOE	NaN		

	LEGAL LAST NAME	FIRST NAME2	MIDDLE NAME3	NAME EXTENSION4	DATE OF BIRTH	…	\
0	NaN	NaN	NaN	NaN	NaT	…	
1	NaN	NaN	NaN	NaN	1928-12-21	…	
2	NaN	NaN	NaN	NaN	1929-07-26	…	
3	NaN	NaN	NaN	NaN	1905-04-10	…	
4	NaN	NaN	NaN	NaN	1924-10-12	…	

```
  PAROLE ELIGIBILITY DATE EARLIEST POSSIBLE RELEASE DATE  \
0                     NaN                             NaN
1     1952-06-20 00:00:00                             NaN
2                     NaN                             NaN
3                     NaN                             NaN
4     1955-05-02 00:00:00                             LFE

                    GOOD TIME LAW INST RELEASE DATE  \
0                                       1986-01-06
1  2926                                 1952-08-31
2                                       1955-07-20
3                                       1987-12-24
4  2926                                 1989-07-19

                INST RELEASE TYPE PAROLE BOARD NEXT REVIEW DATE(MONTH&YEAR)  \
0  MANDATORY DISCHARGE                                                   NaN
1  ESCAPE                                                                NaN
2  DISCRETIONARY PAROLE                                                  NaN
3  MANDATORY DISCHARGE                                                   NaN
4  DISCRETIONARY PAROLE                                                  NaN

  PAROLE BOARD FINAL HEARING DATE(MONTH&YEAR)             PAROLE BOARD STATUS  \
0                                         NaT                             NaN
1                                         NaT                             NaN
2                                         NaT  PAROLED
3                                         NaT                             NaN
4                                         NaT  PAROLED

   PAROLE DATE           PAROLE DISCHARGE DESC
0          NaT                             NaN
1          NaT
2   1980-12-09  EARLY DISCHARGE BY PAROLE BRD
3          NaT
4   1993-01-17  EARLY DISCHARGE BY PAROLE BRD

[5 rows x 32 columns]
```

0.4 3. Description of Data

The DataFrame info() method is used to see helpful descriptions of the data, such as the column name and number of rows. The 'Non-Null Count' is the number of rows that have a value for that particular column. The 'Dtype' is the data type found within each column. An int64 is an integer, an object type is usually written text, and datetime64 is a date time value.

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72954 entries, 0 to 72953
```

```
Data columns (total 32 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   ID NUMBER                                     72954 non-null  int64
 1   COMMITTED LAST NAME                           72953 non-null  object
 2   FIRST NAME                                    72953 non-null  object
 3   MIDDLE NAME                                   54905 non-null  object
 4   NAME EXTENSION                                72954 non-null  object
 5   LEGAL LAST NAME                               1045 non-null   object
 6   FIRST NAME2                                   1045 non-null   object
 7   MIDDLE NAME3                                  1045 non-null   object
 8   NAME EXTENSION4                               1045 non-null   object
 9   DATE OF BIRTH                                 72940 non-null  datetime64[ns]
 10  RACE DESC                                     72954 non-null  object
 11  GENDER                                        72954 non-null  object
 12  FACILITY                                      14080 non-null  object
 13  CURRENT SENTENCE PARDONED OR COMMUTED DATE    72954 non-null  object
 14  GUN CLAUSE                                    256 non-null    object
 15  SENTENCE BEGIN DATE                           71475 non-null  datetime64[ns]
 16  MIN TERM/YEAR                                 72954 non-null  object
 17  MIN MONTH                                     72588 non-null  float64
 18  MIN DAY                                       72588 non-null  float64
 19  MAX TERM/YEAR                                 72954 non-null  object
 20  MAX MONTH                                     72383 non-null  float64
 21  MAX DAY                                       72383 non-null  float64
 22  PAROLE ELIGIBILITY DATE                       66996 non-null  object
 23  EARLIEST POSSIBLE RELEASE DATE                72306 non-null  object
 24  GOOD TIME LAW                                 72954 non-null  object
 25  INST RELEASE DATE                             67994 non-null  datetime64[ns]
 26  INST RELEASE TYPE                             67994 non-null  object
 27  PAROLE BOARD NEXT REVIEW DATE(MONTH&YEAR)     41546 non-null  object
 28  PAROLE BOARD FINAL HEARING DATE(MONTH&YEAR)   1973 non-null   datetime64[ns]
 29  PAROLE BOARD STATUS                           69868 non-null  object
 30  PAROLE DATE                                   25380 non-null  datetime64[ns]
 31  PAROLE DISCHARGE DESC                         34862 non-null  object
dtypes: datetime64[ns](5), float64(4), int64(1), object(22)
memory usage: 17.8+ MB
```

0.5 4. Creating a New Active Prisoner Distinction Column

A new column will be created that will reflect if the inmates are actively incarcerated or are no longer in prison. This will be created by assigning a new column that includes inmates in both the full and active inmate databases

0.5.1 4.1 Examining the Active Prisoner Database

The Active prisoner database will be loaded and examined in the same manner as with the full inmate database

[4]:
```
xls = pd.ExcelFile('inmateDownloadActive.xlsx')
df_Active = pd.read_excel(xls, 'Record Type 1')
```

Some of the Active prisoner database columns are slightly different than the full database, but it contains the same basic information as the full database.

[5]:
```
df_Active.head()
```

[5]:

```
   ID NUMBER COMMITTED LAST NAME FIRST NAME MIDDLE NAME NAME EXTENSION  \
0       6145                KANE     THOMAS         NaN
1      20841              ARNOLD    WILLIAM           L
2      25324              WALKER    RICHARD           T
3      25565             ALVAREZ     THOMAS           A
4      26103               ADAMS      BRIAN           J

  LEGAL LAST NAME FIRST NAME.1 MIDDLE NAME.1 NAME EXTENSION.1 DATE OF BIRTH  \
0             NaN          NaN           NaN              NaN    1928-12-21
1             NaN          NaN           NaN              NaN    1942-08-28
2             NaN          NaN           NaN              NaN    1946-12-24
3             NaN          NaN           NaN              NaN    1947-10-24
4             NaN          NaN           NaN              NaN    1949-04-20

   … PAROLE ELIGIBILITY DATE EARLIEST POSSIBLE RELEASE DATE  \
0  …     1952-06-20 00:00:00                            NaN
1  …     1959-06-02 00:00:00                            LFE
2  …     1972-12-15 00:00:00                            LFE
3  …                     NaN                            NaN
4  …                     LFE                            LFE

                   GOOD TIME LAW INST RELEASE DATE  \
0  2926                                 1952-08-31
1  2926                                 1967-07-15
2  2926                                 2008-11-25
3  2926                                        NaT
4  2926                                        NaT

                INST RELEASE TYPE PAROLE BOARD NEXT REVIEW DATE(MONTH&YEAR)  \
0  ESCAPE                                                              NaN
1  ESCAPE                                          1959-12-01 00:00:00
2  DISCRETIONARY PAROLE                                                NaN
3                             NaN                  2023-05-01 00:00:00
4                             NaN                  2024-03-01 00:00:00

  PAROLE BOARD FINAL HEARING DATE(MONTH&YEAR)            PAROLE BOARD STATUS  \
0                                         NaT                            NaN
1                                         NaT  INITIAL REVIEW
2                                         NaT  CONTINUED ON PAROLE
```

```
3                                          NaT  DEFERRED
4                                          NaT  DEFERRED

   Unnamed: 30                     Unnamed: 31
0          NaN
1          NaN
2          NaN
3          NaN                             NaN
4          NaN                             NaN

[5 rows x 32 columns]
```

[6]:
```
df_Active.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7453 entries, 0 to 7452
Data columns (total 32 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   ID NUMBER                               7453 non-null   int64
 1   COMMITTED LAST NAME                     7453 non-null   object
 2   FIRST NAME                              7453 non-null   object
 3   MIDDLE NAME                             5464 non-null   object
 4   NAME EXTENSION                          7453 non-null   object
 5   LEGAL LAST NAME                         102 non-null    object
 6   FIRST NAME.1                            102 non-null    object
 7   MIDDLE NAME.1                           102 non-null    object
 8   NAME EXTENSION.1                        102 non-null    object
 9   DATE OF BIRTH                           7453 non-null   datetime64[ns]
 10  RACE DESC                               7453 non-null   object
 11  GENDER                                  7453 non-null   object
 12  FACILITY                                7401 non-null   object
 13  CURRENT SENTENCE PARDONED OR COMMUTED DATE  7453 non-null   object
 14  GUN CLAUSE                              0 non-null      float64
 15  SENTENCE BEGIN DATE                     7450 non-null   datetime64[ns]
 16  MIN TERM/YEAR                           7453 non-null   object
 17  MIN MONTH                               7176 non-null   float64
 18  MIN DAY                                 7176 non-null   float64
 19  MAX TERM/YEAR                           7453 non-null   object
 20  MAX MONTH                               7041 non-null   float64
 21  MAX DAY                                 7041 non-null   float64
 22  PAROLE ELIGIBILITY DATE                 5444 non-null   object
 23  EARLIEST POSSIBLE RELEASE DATE          7047 non-null   object
 24  GOOD TIME LAW                           7453 non-null   object
 25  INST RELEASE DATE                       2594 non-null   datetime64[ns]
 26  INST RELEASE TYPE                       2594 non-null   object
 27  PAROLE BOARD NEXT REVIEW DATE(MONTH&YEAR)   4920 non-null   object
```

```
 28  PAROLE BOARD FINAL HEARING DATE(MONTH&YEAR)  1339 non-null   datetime64[ns]
 29  PAROLE BOARD STATUS                          7449 non-null   object
 30  Unnamed: 30                                  0 non-null      float64
 31  Unnamed: 31                                  1545 non-null   object
dtypes: datetime64[ns](4), float64(6), int64(1), object(21)
memory usage: 1.8+ MB
```

0.5.2 4.2 Assigning a new 'Active' Inmate Column

The pandas DataFrame assign() method returns a new column, 'ACTIVE', that reflects if a prisoner is Active or no longer incarcerated. It uses the DataFrame .isin method to check if the full database's inmates' ID NUMBER is in the Active inmate database. The DataFrame .astype method changes the column to the integer datatype. A new dataframe called 'df1' was also created to store this all the previous columns as well as the new Active Column.

[7]:
```
df1 = df.assign(ACTIVE=df['ID NUMBER']
                .isin(df_Active['ID NUMBER'])
                .astype(int))
```

The new column 'ACTIVE' is seen at the as last column. If a row has a 1, that means the prisoner is active, and a 0 means they are no longer incarcerated.

[8]:
```
df1.head()
```

[8]:

	ID NUMBER	COMMITTED LAST NAME	FIRST NAME	MIDDLE NAME	NAME EXTENSION	…
0	1702	CLIFFORD	BRADLEY	NaN		
1	6145	KANE	THOMAS	NaN		
2	6452	ATKINS	LARRY	NaN		
3	12444	SHANEYFELT	CHARLEY	NaN		
4	15379	BEADES	JOE	NaN		

	LEGAL LAST NAME	FIRST NAME2	MIDDLE NAME3	NAME EXTENSION4	DATE OF BIRTH	…
0	NaN	NaN	NaN	NaN	NaT	…
1	NaN	NaN	NaN	NaN	1928-12-21	…
2	NaN	NaN	NaN	NaN	1929-07-26	…
3	NaN	NaN	NaN	NaN	1905-04-10	…
4	NaN	NaN	NaN	NaN	1924-10-12	…

	EARLIEST POSSIBLE RELEASE DATE	GOOD TIME LAW
0	NaN	
1	NaN	2926
2	NaN	
3	NaN	
4	LFE	2926

	INST RELEASE DATE	INST RELEASE TYPE
0	1986-01-06	MANDATORY DISCHARGE
1	1952-08-31	ESCAPE

```
2        1955-07-20  DISCRETIONARY PAROLE
3        1987-12-24  MANDATORY DISCHARGE
4        1989-07-19  DISCRETIONARY PAROLE

  PAROLE BOARD NEXT REVIEW DATE(MONTH&YEAR)  \
0                                       NaN
1                                       NaN
2                                       NaN
3                                       NaN
4                                       NaN

  PAROLE BOARD FINAL HEARING DATE(MONTH&YEAR)             PAROLE BOARD STATUS  \
0                                         NaT                              NaN
1                                         NaT                              NaN
2                                         NaT  PAROLED
3                                         NaT                              NaN
4                                         NaT  PAROLED

   PAROLE DATE           PAROLE DISCHARGE DESC ACTIVE
0          NaT                             NaN      0
1          NaT                                      1
2   1980-12-09  EARLY DISCHARGE BY PAROLE BRD       0
3          NaT                                      0
4   1993-01-17  EARLY DISCHARGE BY PAROLE BRD       0

[5 rows x 33 columns]
```

0.6 5. Initial Data Cleaning

Datasets are almost always imperfect and this can hinder future analysis.

0.6.1 5.1 Dropping unneeded columns

The columns that were removed contained unnessescary personal information about the inmates or their sentences that were not of use in this research's more macro-based lense. Parole information was also referenced by the NDCS to be fairly incomplete and too difficult to research.

Uneeded columns can be deleted using the DataFrame drop method.

[9]:
```
df1 = df1.drop(['FIRST NAME',
          'FIRST NAME',
          'MIDDLE NAME',
         'MIDDLE NAME',
         'NAME EXTENSION',
         'COMMITTED LAST NAME',
         'LEGAL LAST NAME',
         'NAME EXTENSION',
         'FIRST NAME2',
```

```
        'MIDDLE NAME3',
        'NAME EXTENSION4',
        'GUN CLAUSE',
        'MIN MONTH',
        'CURRENT SENTENCE PARDONED OR COMMUTED DATE',
        'MIN DAY',
        'MAX MONTH',
        'MAX DAY',
        'PAROLE ELIGIBILITY DATE',
        'GOOD TIME LAW',
        'EARLIEST POSSIBLE RELEASE DATE',
        'INST RELEASE TYPE',
        'PAROLE BOARD NEXT REVIEW DATE(MONTH&YEAR)',
        'PAROLE BOARD FINAL HEARING DATE(MONTH&YEAR)',
        'PAROLE BOARD STATUS',
        'PAROLE DISCHARGE DESC',
        'PAROLE DATE'],axis=1)
```

The remaining rows can now be seen.

[10]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72954 entries, 0 to 72953
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID NUMBER            72954 non-null  int64
 1   DATE OF BIRTH        72940 non-null  datetime64[ns]
 2   RACE DESC            72954 non-null  object
 3   GENDER               72954 non-null  object
 4   FACILITY             14080 non-null  object
 5   SENTENCE BEGIN DATE  71475 non-null  datetime64[ns]
 6   MIN TERM/YEAR        72954 non-null  object
 7   MAX TERM/YEAR        72954 non-null  object
 8   INST RELEASE DATE    67994 non-null  datetime64[ns]
 9   ACTIVE               72954 non-null  int64
dtypes: datetime64[ns](3), int64(2), object(5)
memory usage: 5.6+ MB
```

[11]:
```
df1.head()
```

[11]:

	ID NUMBER	DATE OF BIRTH	RACE DESC	GENDER	\
0	1702	NaT		MALE	
1	6145	1928-12-21	WHITE	MALE	
2	6452	1929-07-26	WHITE	MALE	
3	12444	1905-04-10	WHITE	MALE	

```
4      15379    1924-10-12  WHITE                              MALE

                          FACILITY SENTENCE BEGIN DATE MIN TERM/YEAR  \
0                               NaN                NaT             0
1  NEBRASKA STATE PENITENTIARY             1952-06-20             1
2  NEBRASKA STATE PENITENTIARY             1953-11-25             2
3                               NaN          1935-10-15             1
4                               NaN          1945-05-02            10

  MAX TERM/YEAR INST RELEASE DATE  ACTIVE
0             0        1986-01-06       0
1             3        1952-08-31       1
2            10        1955-07-20       0
3             9        1987-12-24       0
4           LFE        1989-07-19       0
```

0.6.2 5.2 Dropping All "NA" Missing Features

We can check if any columns have missing data values and count them by using the isnull() method and sum() method.

[12]:
```
df1['SENTENCE BEGIN DATE'].isnull().sum()
```

[12]: 1479

Missing data values or (NA) is removed from the data in certain columns. The DataFrame dropna() method can be used to do this.

We drop all inmates who have an unknown sentence begin date.

[13]:
```
df1 = df1.dropna(axis=0, how="any", subset=['SENTENCE BEGIN DATE'])
```

We can check to see if this function worked.

[14]:
```
df1['SENTENCE BEGIN DATE'].isnull().sum()
```

[14]: 0

We drop all inmates who have an unknown date of birth.

[15]:
```
df1['DATE OF BIRTH'].isnull().sum()
```

[15]: 6

[16]:
```
df1 = df1.dropna(axis=0, how="any", subset=['DATE OF BIRTH'])
```

0.6.3 5.3 Checking for Duplicate Rows

Sometimes data contains duplicate rows of information that may skew future analysis. These can be found by using the DataFrame duplicated() method.

[17]:
```
duplicate = df1[df1.duplicated()]
duplicate
```

[17]:
```
Empty DataFrame
Columns: [ID NUMBER, DATE OF BIRTH, RACE DESC, GENDER, FACILITY, SENTENCE BEGIN
DATE, MIN TERM/YEAR, MAX TERM/YEAR, INST RELEASE DATE, ACTIVE]
Index: []
```

The duplicate rows are removed by using DataFrame drop_duplicates().

[18]:
```
df1 = df1.drop_duplicates()
```

The overall number of entries will change.

[19]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 71469 entries, 1 to 72953
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID NUMBER            71469 non-null  int64
 1   DATE OF BIRTH        71469 non-null  datetime64[ns]
 2   RACE DESC            71469 non-null  object
 3   GENDER               71469 non-null  object
 4   FACILITY             13589 non-null  object
 5   SENTENCE BEGIN DATE  71469 non-null  datetime64[ns]
 6   MIN TERM/YEAR        71469 non-null  object
 7   MAX TERM/YEAR        71469 non-null  object
 8   INST RELEASE DATE    66510 non-null  datetime64[ns]
 9   ACTIVE               71469 non-null  int64
dtypes: datetime64[ns](3), int64(2), object(5)
memory usage: 6.0+ MB
```

0.7 6. Adding in Inmate Age Columns for Further Analysis

Some columns will be added using information from the original columns to help expand the analysis.

0.7.1 6.1 Changing Column Data Types to Datetime

Datetime is a data type that is used for dates and times. The Pandas to_datetime method changes an object data type to a datetime format.

[20]:
```
df1['SENTENCE BEGIN DATE DT'] = pd.to_datetime(df1['SENTENCE BEGIN DATE'])
df1['DATE OF BIRTH DT'] = pd.to_datetime(df1['DATE OF BIRTH'])
```

0.7.2 6.2 Finding Age of Inmates at Time of Incarceration

This age will be helpful to see at what age inmates were incarcerated and how that may have changed over time

Datetime also is a Python module that can manipulate the datetime datatype. Timedelta is a function from datetime that is used to calcuate the difference between dates. The Sentence Begin Age can be found through finding the number of days between the sentence begin date and the inmate's date of birth. This number is then divided by 365 days to get their age in years.

[21]:
```
import datetime as dt
from datetime import timedelta

df1['SENTENCE BEGIN AGE DAYS'] = df1['SENTENCE BEGIN DATE DT'] - df1['DATE OF␣
 ↪BIRTH DT']
df1['SENTENCE BEGIN AGE'] = df1['SENTENCE BEGIN AGE DAYS'] / timedelta(days=365)
```

0.7.3 6.3 Finding Current Age of Inmates

This age will be helpful to see how old inmates are now, especially for active ones.

The 'date' is set to the current local date and time using the Pandas datetime.now() method. .strftime is the date in a string format that is given in Year/Month/Day. The Current Age can be found through finding the number of days between the current date and the inmate's date of birth. This number is then divided by 365 days to get their age in years.

[22]:
```
date = pd.datetime.now().strftime('%Y%m%d')
date = pd.to_datetime(date,format='%Y%m%d')
df1['CURRENT AGE DAYS'] = date - df1['DATE OF BIRTH DT']
df1['CURRENT AGE'] = df1['CURRENT AGE DAYS'] / timedelta(days=365)
```

```
<ipython-input-22-17aa42885836>:1: FutureWarning: The pandas.datetime class is
deprecated and will be removed from pandas in a future version. Import from
datetime module instead.
  date = pd.datetime.now().strftime('%Y%m%d')
```

We can drop these columns since they were just made temporarily to get the ages in years.

[23]:
```
df1 = df1.drop([
              'CURRENT AGE DAYS',
              'SENTENCE BEGIN AGE DAYS'],axis=1)
```

0.7.4 6.4 Making a New Column for the Sentence Begin Year

This column will be helpful to later separate inmates into different decade cohorts for analysis.

The datetime (dt) .year function can be used to find just the year portion of the full date.

[24]:
```
df1['SENTENCE BEGIN YEAR'] = df1['SENTENCE BEGIN DATE DT'].dt.year
```

0.8 7.0 Altering Life Sentencing Values

Some inmates recieve life sentences in their MIN TERM/YEAR or MAX TERM/YEAR, and are given the value 'LFE' instead of an integer. This object value limits evaluation on average MIN/MAX TERM of the inmates. These values are changed to numerical representations of how much life left the inmates have.

0.8.1 7.1 Declaring Average Lifespan Values and Finding Years Between Lifespan and Inmate Sentence Begin Age

lfeM represents the average lifespan of a male Nebraskan, and lfeF represents the average lifespan of a female Nebraskan. This values were obtained from (LINK). The columns LFE SENTENCE M and LFE SENTENCE F are created from taking the respective average lifespans subtracted by the inmates' age at the start of their incarceration.

[25]:
```
lfeM = 77.7
lfeF = 81.89

df1['LFE SENTENCE M'] = lfeM - df1['SENTENCE BEGIN AGE']
df1['LFE SENTENCE F'] = lfeF - df1['SENTENCE BEGIN AGE']
```

0.8.2 7.2 Assigning Found Values to a Life Sentence Column

The lfe_sentence() function looks to see if the row has the a male or female inmate, and returns the corresponding LFE SENTENCE column. A new column, LFE SENTENCE MIN/MAX is created from applying the lfe_sentence() function to our data. The values in LFE SENTENCE MIN/MAX are then rounded using .round().

[26]:
```
def lfe_sentence(df1):
    if df1['GENDER'] == 'MALE':
        return df1['LFE SENTENCE M']
    else:
        return df1['LFE SENTENCE F']

df1['LFE SENTENCE MIN/MAX'] = df1.apply(lfe_sentence,axis=1)
df1['LFE SENTENCE MIN/MAX'] = df1['LFE SENTENCE MIN/MAX'].round()
```

0.8.3 7.3 Replace MIN or MAX TERM/YEAR Values with Life Sentence Age Values

Now the original MIN or MAX TERM/YEAR value of 'LFE' is replaced with the value in LFE SENTEnCE MIN/MAX by using a numpy where method. The where method lets you find and replace values.

[27]:
```
df1['MIN TERM/YEAR'] = np.where(df1['MIN TERM/YEAR'] == 'LFE',
                                df1['LFE SENTENCE MIN/MAX'], df1['MIN TERM/
↪YEAR'])
```

```
df1['MAX TERM/YEAR'] = np.where(df1['MAX TERM/YEAR'] == 'LFE',
                                df1['LFE SENTENCE MIN/MAX'], df1['MAX TERM/
 ↪YEAR'])
```

0.8.4 7.4 Dropping Old Columns

The columns used to create these life sentence replacement values can now be deleted.

[28]:
```
df1=df1.drop([
             'LFE SENTENCE M',
             'LFE SENTENCE F',
'LFE SENTENCE MIN/MAX'],axis=1)
```

[29]:
```
df1.head()
```

[29]:

	ID NUMBER	DATE OF BIRTH		RACE DESC	GENDER	\
1	6145	1928-12-21	WHITE		MALE	
2	6452	1929-07-26	WHITE		MALE	
3	12444	1905-04-10	WHITE		MALE	
4	15379	1924-10-12	WHITE		MALE	
6	16657	1929-01-10	WHITE		MALE	

	FACILITY	SENTENCE BEGIN DATE	MIN TERM/YEAR	\
1	NEBRASKA STATE PENITENTIARY	1952-06-20	1	
2	NEBRASKA STATE PENITENTIARY	1953-11-25	2	
3	NaN	1935-10-15	1	
4	NaN	1945-05-02	10	
6	NaN	1948-12-22	58	

	MAX TERM/YEAR	INST RELEASE DATE	ACTIVE	SENTENCE BEGIN DATE DT	\
1	3	1952-08-31	1	1952-06-20	
2	10	1955-07-20	0	1953-11-25	
3	9	1987-12-24	0	1935-10-15	
4	57	1989-07-19	0	1945-05-02	
6	58	2002-12-27	0	1948-12-22	

	DATE OF BIRTH DT	SENTENCE BEGIN AGE	CURRENT AGE	SENTENCE BEGIN YEAR
1	1928-12-21	23.512329	92.273973	1952
2	1929-07-26	24.350685	91.679452	1953
3	1905-04-10	30.534247	115.989041	1935
4	1924-10-12	20.567123	96.468493	1945
6	1929-01-10	19.961644	92.219178	1948

0.9 8.0 Deleting Death and Independent Values

Very few of the MIN or MAX TERM/YEAR values are labelled with DTH or IND. These stand for Death and Independent sentences, and only hinder analysis on average sentencing. They are

removed to simplify the data, as they account for fewer than 30 inmates. These few inmates act as major outliers compared to the rest of the database.

0.9.1 8.1 Checking to See if IND or DTH Values Exist

The DataFrame method str.contains() finds if the given value is in the DataFrame being analyzed. .any() returns a true or false statement if the given value is found in the DataFrame.

[30]:
```
df1['MAX TERM/YEAR'].str.contains('IND').any()
```

[30]: True

[31]:
```
df1['MAX TERM/YEAR'].str.contains('DTH').any()
```

[31]: True

0.9.2 8.2 Drop rows where values equal IND or DTH

The drop() method is used again to delete any columns where the MIN or MAX TERM/YEAR value is equal to DTH or IND.

[32]:
```
df1.drop(df1.loc[df['MIN TERM/YEAR']=='DTH'].index, inplace=True)
df1.drop(df1.loc[df['MIN TERM/YEAR']=='IND'].index, inplace=True)

df1.drop(df1.loc[df['MAX TERM/YEAR']=='DTH'].index, inplace=True)
df1.drop(df1.loc[df['MAX TERM/YEAR']=='IND'].index, inplace=True)
```

0.9.3 8.3 Check Work

[33]:
```
df1['MAX TERM/YEAR'].str.contains('IND').any()
```

[33]: False

[34]:
```
df1['MAX TERM/YEAR'].str.contains('DTH').any()
```

[34]: False

0.9.4 8.4 Changing the MIN and MAX TERM/YEAR to numeric values

The pandas method to_numeric() is used to change the MIN and MAX TERM/YEAR string values into float values (integers with decimal points). This will make finding future inmate sentencing averages easier.

[35]:
```
df1['MIN TERM/YEAR'] = pd.to_numeric(df1['MIN TERM/YEAR'])
df1['MAX TERM/YEAR'] = pd.to_numeric(df1['MAX TERM/YEAR'])
```

Now the value type change can be seen.

[36]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 71440 entries, 1 to 72953
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ID NUMBER               71440 non-null  int64
 1   DATE OF BIRTH           71440 non-null  datetime64[ns]
 2   RACE DESC               71440 non-null  object
 3   GENDER                  71440 non-null  object
 4   FACILITY                13570 non-null  object
 5   SENTENCE BEGIN DATE     71440 non-null  datetime64[ns]
 6   MIN TERM/YEAR           71440 non-null  float64
 7   MAX TERM/YEAR           71440 non-null  float64
 8   INST RELEASE DATE       66493 non-null  datetime64[ns]
 9   ACTIVE                  71440 non-null  int64
 10  SENTENCE BEGIN DATE DT  71440 non-null  datetime64[ns]
 11  DATE OF BIRTH DT        71440 non-null  datetime64[ns]
 12  SENTENCE BEGIN AGE      71440 non-null  float64
 13  CURRENT AGE             71440 non-null  float64
 14  SENTENCE BEGIN YEAR     71440 non-null  int64
dtypes: datetime64[ns](5), float64(4), int64(3), object(3)
memory usage: 11.2+ MB
```

0.10 9.0 Reducing the Database to only contain inmates incarcerated after 1979.

When looking at broad trends over the studied period, this is the range that needs to be researched. All previous incarcerations are outlier data that had incomplete digital documentation.

The DataFrame can be altered to only keep rows where inmates were incarcerated after 1979.

[37]:
```
df1 = df1[df1['SENTENCE BEGIN YEAR'] > 1979]
```

The DataFrame lost around 2857 rows of inmates.

[38]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 68583 entries, 142 to 72953
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ID NUMBER               68583 non-null  int64
 1   DATE OF BIRTH           68583 non-null  datetime64[ns]
 2   RACE DESC               68583 non-null  object
 3   GENDER                  68583 non-null  object
 4   FACILITY                11586 non-null  object
 5   SENTENCE BEGIN DATE     68583 non-null  datetime64[ns]
```

```
 6   MIN TERM/YEAR           68583 non-null  float64
 7   MAX TERM/YEAR           68583 non-null  float64
 8   INST RELEASE DATE       63663 non-null  datetime64[ns]
 9   ACTIVE                  68583 non-null  int64
 10  SENTENCE BEGIN DATE DT  68583 non-null  datetime64[ns]
 11  DATE OF BIRTH DT        68583 non-null  datetime64[ns]
 12  SENTENCE BEGIN AGE      68583 non-null  float64
 13  CURRENT AGE             68583 non-null  float64
 14  SENTENCE BEGIN YEAR     68583 non-null  int64
dtypes: datetime64[ns](5), float64(4), int64(3), object(3)
memory usage: 8.4+ MB
```

0.11 10.0 Converting DataFrame to CSV for Future Use

The DataFrame method to_csv converts a DataFrame to a CSV file for easier storage and sharing.

[39]:
```
df1.to_csv('inmate_updatedClean_demographics.csv', encoding='utf-8',␣
↪index=False)
```

[ ]:
```

```