Processes followed in executing this task

Data understanding - the key to success on any data project is to understand the data in detail. So we took the time to understand the data model and domain of your task.

Data extraction - after understanding your Task, we then architected what an ideal dataset should look like for this problem and extracted it from the relevant data sources

Data Wrangling(Gather, Assess, Clean) - After extracting the raw data, we needed to process and model this data into a dataset that can precisely answer the questions and produce analytics.

Auralin and Novodra Trials

We will be looking at the phase two clinical trial data of 350 patients for a new innovative oral insulin called Auralin - a proprietary capsule that can solve this stomach lining problem.

Phase two trials are intended to:

- Test the efficacy and the dose response of a drug
- Identify adverse reactions

In this trial, half of the patients are being treated with Auralin, and the other 175 being treated with a popular injectable insulin called Novodra. By comparing key metrics between these two drugs, we can determine if Auralin is effective.

Gather

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
patients = pd.read_csv('patients.csv')
treatments = pd.read_csv('treatments.csv')
adverse_reactions = pd.read_csv('adverse_reactions.csv')
```

Assess

In [3]:

```
# assessing the datasets visually
patients
```

Out[3]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	country
0	1	female	Zoe	Wellish	576 Brown Bear Drive	Rancho California	California	92390.0	United States
1	2	female	Pamela	Hill	2370 University Hill Road	Armstrong	Illinois	61812.0	United States
2	3	male	Jae	Debord	1493 Poling Farm Road	York	Nebraska	68467.0	United States
3	4	male	Liêm	Phan	2335 Webster Street	Woodbridge	NJ	7095.0	United States
					1428				Unitec

In [4]:

```
treatments
```

Out[4]:

	given_name	surname	auralin	novodra	hba1c_start	hba1c_end	hba1c_change
0	veronika	jindrová	41u - 48u	-	7.63	7.20	NaN
1	elliot	richardson	-	40u - 45u	7.56	7.09	0.97
2	yukitaka	takenaka	-	39u - 36u	7.68	7.25	NaN
3	skye	gormanston	33u - 36u	-	7.97	7.62	0.35
4	alissa	montez	-	33u - 29u	7.78	7.46	0.32
...	...	...	...	...	...	...	...
275	albina	zetticci	45u - 51u	-	7.93	7.73	0.20

In [5]:

```
adverse_reactions
```

Out[5]:

	given_name	surname	adverse_reaction
0	berta	napolitani	injection site discomfort
1	lena	baer	hypoglycemia
2	joseph	day	hypoglycemia
3	flavia	fiorentino	cough
4	manouck	wubbels	throat irritation
5	jasmine	sykes	hypoglycemia
6	louise	johnson	hypoglycemia
7	albinca	komavec	hypoglycemia
8	noe	aranda	hypoglycemia
9	sofia	hermansen	injection site discomfort

In [6]:

```
# Assessing our datasets programmatically
patients.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 503 entries, 0 to 502
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   patient_id    503 non-null    int64  
 1   assigned_sex  503 non-null    object 
 2   given_name    503 non-null    object 
 3   surname       503 non-null    object 
 4   address       491 non-null    object 
 5   city          491 non-null    object 
 6   state         491 non-null    object 
 7   zip_code      491 non-null    float64
 8   country       491 non-null    object 
 9   contact       491 non-null    object 
 10  birthdate     503 non-null    object 
 11  weight        503 non-null    float64
 12  height        503 non-null    int64  
 13  bmi           503 non-null    float64
dtypes: float64(3), int64(2), object(9)
memory usage: 55.1+ KB
```

In [7]:

```
treatments.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 280 entries, 0 to 279
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   given_name    280 non-null    object 
 1   surname       280 non-null    object 
 2   auralin       280 non-null    object 
 3   novodra       280 non-null    object 
 4   hba1c_start   280 non-null    float64
 5   hba1c_end     280 non-null    float64
 6   hba1c_change  171 non-null    float64
dtypes: float64(3), object(4)
memory usage: 15.4+ KB
```

In [8]:

```
adverse_reactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34 entries, 0 to 33
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   given_name        34 non-null     object
 1   surname           34 non-null     object
 2   adverse_reaction  34 non-null     object
dtypes: object(3)
memory usage: 944.0+ bytes
```

In [9]:

```
entire_column = pd.Series(list(patients) + list(treatments) + list(adverse_reactions))
entire_column[entire_column.duplicated()]
```

Out[9]:

```
14    given_name
15       surname
21    given_name
22       surname
dtype: object
```

In [10]:

```
patients[patients['address'].isnull()]
```

Out[10]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	countr
209	210	female	Lalita	Eldarkhanov	NaN	NaN	NaN	NaN	Na
219	220	male	Mỹ	Quynh	NaN	NaN	NaN	NaN	Na
230	231	female	Elisabeth	Knudsen	NaN	NaN	NaN	NaN	Na
234	235	female	Martina	Tománková	NaN	NaN	NaN	NaN	Na
242	243	male	John	O'Brian	NaN	NaN	NaN	NaN	Na
249	250	male	Benjamin	Mehler	NaN	NaN	NaN	NaN	Na
257	258	male	Jin	Kung	NaN	NaN	NaN	NaN	Na
264	265	female	Wafiyyah	Asfour	NaN	NaN	NaN	NaN	Na
269	270	female	Flavia	Fiorentino	NaN	NaN	NaN	NaN	Na
278	279	female	Generosa	Cabán	NaN	NaN	NaN	NaN	Na
286	287	male	Lewis	Webb	NaN	NaN	NaN	NaN	Na
296	297	female	Chỉ	Lâm	NaN	NaN	NaN	NaN	Na

In [11]:

```
patients.describe()
```

Out[11]:

	patient_id	zip_code	weight	height	bmi
count	503.000000	491.000000	503.000000	503.000000	503.000000
mean	252.000000	49084.118126	173.434990	66.634195	27.483897
std	145.347859	30265.807442	33.916741	4.411297	5.276438
min	1.000000	1002.000000	48.800000	27.000000	17.100000
25%	126.500000	21920.500000	149.300000	63.000000	23.300000
50%	252.000000	48057.000000	175.300000	67.000000	27.200000
75%	377.500000	75679.000000	199.500000	70.000000	31.750000
max	503.000000	99701.000000	255.900000	79.000000	37.700000

In [12]:

```
treatments.describe()
```

Out[12]:

	hba1c_start	hba1c_end	hba1c_change
count	280.000000	280.000000	171.000000
mean	7.985929	7.589286	0.546023
std	0.568638	0.569672	0.279555
min	7.500000	7.010000	0.200000
25%	7.660000	7.270000	0.340000
50%	7.800000	7.420000	0.380000
75%	7.970000	7.570000	0.920000
max	9.950000	9.580000	0.990000

In [13]:

```
patients.sample(5)
```

Out[13]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	c
98	99	male	Jan	Baum	1733 Blackwell Street	Fairbanks	AK	99701.0	
303	304	female	Joe	Edwards	1526 Tully Street	Detroit	MI	48219.0	
240	241	female	Marphisa	Compagnon	3391 Marcus Street	Huntsville	AL	35806.0	
129	130	female	Rebecca	Jephcott	989 Wayback Lane	New York	NY	10004.0	
398	399	male	Ilija	Horvat	4380 Riverside Drive	Cave Spring	GA	30124.0	

In [14]:

```
patients.surname.value_counts()
```

Out[14]:

```
Doe            6
Jakobsen       3
Taylor         3
Ogochukwu      2
Tucker         2
              ..
Casárez        1
Mata           1
Pospíšil       1
Rukavina       1
Onyekaozulu    1
Name: surname, Length: 466, dtype: int64
```

In [15]:

```
patients.address.value_counts()
```

Out[15]:

```
123 Main Street             6
2778 North Avenue           2
2476 Fulton Street          2
648 Old Dear Lane           2
3094 Oral Lake Road         1
                           ..
1066 Goosetown Drive        1
4291 Patton Lane            1
4643 Reeves Street          1
174 Lost Creek Road         1
3652 Boone Crockett Lane    1
Name: address, Length: 483, dtype: int64
```

In [16]:

```
patients[patients['address'].duplicated()]
```

Out[16]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	country	
29	30	male	Jake	Jakobsen	648 Old Dear Lane	Port Jervis	New York	12771.0	United States	Jak
219	220	male	Mỹ	Quynh	NaN	NaN	NaN	NaN	NaN	
229	230	male	John	Doe	123 Main Street	New York	NY	12345.0	United States	john
230	231	female	Elisabeth	Knudsen	NaN	NaN	NaN	NaN	NaN	
234	235	female	Martina	Tománková	NaN	NaN	NaN	NaN	NaN	
237	238	male	John	Doe	123 Main Street	New York	NY	12345.0	United States	john

In [17]:

```
patients.weight.sort_values()
```

Out[17]:

```
210     48.8
459    102.1
335    102.7
74     103.2
317    106.0
       ...
144    244.9
61     244.9
283    245.5
118    254.5
485    255.9
Name: weight, Length: 503, dtype: float64
```

In [18]:

```
weight_lbs = patients[patients.surname == 'Zaitseva'].weight * 2.20462
height_in = patients[patients.surname == 'Zaitseva'].height
bmi_check = 703 * weight_lbs / (height_in * height_in)
bmi_check
```

Out[18]:

```
210    19.055827
dtype: float64
```

In [19]:

```
patients[patients.surname == 'Zaitseva'].bmi
```

Out[19]:

```
210    19.1
Name: bmi, dtype: float64
```

In [20]:

```
sum(treatments.auralin.isnull())
```

Out[20]:

```
0
```

In [21]:

```
sum(treatments.novodra.isnull())
```

Out[21]:

```
0
```

After assessing the datasets visually and programmatically it became obvious that our dataset is of low quality and messy and this are the following things discovered

Quality

patients table

- Zip code is a float not a string
- Zip code has four digits sometimes
- Tim Neudorf height is 27 in instead of 72 in
- Full state names sometimes, abbreviations other times
- Dsvid Gustafsson
- Missing demographic information (address - contact columns) ***(can't clean)***
- Erroneous datatypes (assigned sex, state, zip_code, and birthdate columns)
- Multiple phone number formats
- Default John Doe data
- Multiple records for Jakobsen, Gersten, Taylor
- kgs instead of lbs for Zaitseva weight

treatments table

- Missing HbA1c changes
- The letter 'u' in starting and ending doses for Auralin and Novodra
- Lowercase given names and surnames
- Missing records (280 instead of 350)
- Erroneous datatypes (auralin and novodra columns)
- Inaccurate HbA1c changes (leading 4s mistaken as 9s)
- Nulls represented as dashes (-) in auralin and novodra columns

adverse_reactions table

- Lowercase given names and surnames

Tidiness

- Contact column in `patients` table should be split into phone number and email
- Three variables in two columns in `treatments` table (treatment, start dose and end dose)
- Adverse reaction should be part of the `treatments` table
- Given name and surname columns in `patients` table duplicated in `treatments` and `adverse_reactions` tables

Clean

In [22]:

```
#Making a copy of our datasets before the cleaning commence
patients_clean = patients.copy()
treatments_clean = treatments.copy()
adverse_reactions_clean = adverse_reactions.copy()
```

Missing Data

Define

- treatments: Missing records (280 instead of 350)
- the missing treatments records are stored in a file named treatments_cut.csv which we would be joining to the real treatments table by using concat function

Code

In [23]:

```
#using the concat function to join the missing data to the treatments dataset to 350 record
treatments_cut = pd.read_csv('treatments_cut.csv')
treatments_clean = pd.concat([treatments_clean, treatments_cut],
                             ignore_index = True)
```

Test

In [24]:

```
#checking the dataset to see if our code was correctly used
treatments_clean.head()
```

Out[24]:

	given_name	surname	auralin	novodra	hba1c_start	hba1c_end	hba1c_change
0	veronika	jindrová	41u - 48u	-	7.63	7.20	NaN
1	elliot	richardson	-	40u - 45u	7.56	7.09	0.97
2	yukitaka	takenaka	-	39u - 36u	7.68	7.25	NaN
3	skye	gormanston	33u - 36u	-	7.97	7.62	0.35
4	alissa	montez	-	33u - 29u	7.78	7.46	0.32

In [25]:

```
treatments_clean.tail()
```

Out[25]:

	given_name	surname	auralin	novodra	hba1c_start	hba1c_end	hba1c_change
345	rovzan	kishiev	32u - 37u	-	7.75	7.41	0.34
346	jakob	jakobsen	-	28u - 26u	7.96	7.51	0.95
347	bernd	schneider	48u - 56u	-	7.74	7.44	0.30
348	berta	napolitani	-	42u - 44u	7.68	7.21	NaN
349	armina	sauvé	36u - 46u	-	7.86	7.40	NaN

treatments: Missing HbA1c changes and Inaccurate HbA1c changes (leading 4s mistaken as 9s)

Define

- Recalculate the hba1c_change column: hba1c_start minus hba1c_end

Code

In [26]:

```
# Subtracting the hba1c_end from hba1c_start to get the correct calculation of hba1c_change
treatments_clean.hba1c_change = (treatments_clean.hba1c_start - treatments_clean.hba1c_end)
```

Test

In [27]:

```
#checking if it has been recalculated
treatments_clean.head()
```

Out[27]:

	given_name	surname	auralin	novodra	hba1c_start	hba1c_end	hba1c_change
0	veronika	jindrová	41u - 48u	-	7.63	7.20	0.43
1	elliot	richardson	-	40u - 45u	7.56	7.09	0.47
2	yukitaka	takenaka	-	39u - 36u	7.68	7.25	0.43
3	skye	gormanston	33u - 36u	-	7.97	7.62	0.35
4	alissa	montez	-	33u - 29u	7.78	7.46	0.32

In [28]:

```
treatments_clean.tail()
```

Out[28]:

	given_name	surname	auralin	novodra	hba1c_start	hba1c_end	hba1c_change
345	rovzan	kishiev	32u - 37u	-	7.75	7.41	0.34
346	jakob	jakobsen	-	28u - 26u	7.96	7.51	0.45
347	bernd	schneider	48u - 56u	-	7.74	7.44	0.30
348	berta	napolitani	-	42u - 44u	7.68	7.21	0.47
349	armina	sauvé	36u - 46u	-	7.86	7.40	0.46

Tidiness

- Contact column in patients table contains two variables: phone number and email which needs to be in a separate column

Define

- We have to extract the phone number and email variables from the contact column using regular expressions and pandas' 'str.extract method'. and then drop the contact column when done.

Code

In [29]:

```
patients_clean['phone_number'] = patients_clean.contact.str.extract('((?:\+\d{1,2}\s)?\(?\d

# [a-zA-Z] to signify emails in this dataset all start and end with letters
patients_clean['email'] = patients_clean.contact.str.extract('([a-zA-Z][a-zA-Z0-9_.+-]+@[a-

# Dropping the contact column, and axis=1 denotes that we are referring to a column, not a
patients_clean = patients_clean.drop('contact', axis=1)
```

Test

In [30]:

```
#Checking if our code was properly executed
patients_clean.head()
```

Out[30]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code
0	1	female	Zoe	Wellish	576 Brown Bear Drive	Rancho California	California	92390.0
1	2	female	Pamela	Hill	2370 University Hill Road	Armstrong	Illinois	61812.0
2	3	male	Jae	Debord	1493 Poling Farm Road	York	Nebraska	68467.0
3	4	male	Liêm	Phan	2335 Webster Street	Woodbridge	NJ	7095.0
4	5	male	Tim	Neudorf	1428 Turkey Pen Lane	Dothan	AL	36303.0

In [31]:

```
#Checking if Phone numbers were properly extracted
patients_clean.phone_number.sample(10)
```

Out[31]:

```
111         661-291-1812
150         863-438-6922
220    +1 (707) 896-9250
343         203-251-3573
249                  NaN
228         916-379-7480
130         843-494-0313
364         818-372-7106
93          786-234-0038
137         954-784-6658
Name: phone_number, dtype: object
```

In [32]:

```
#Checking if Emails were properly extracted
patients_clean.email.sort_values().head()
```

Out[32]:

```
404               AaliyahRice@dayrep.com
11          Abdul-NurMummarIsa@rhyta.com
332               AbelEfrem@fleckens.hu
258             AbelYonatan@teleworm.us
305    AddoloratаLombardi@jourrapide.com
Name: email, dtype: object
```

Tidiness

- Three variables in two columns in treatments table (treatment, start dose and end dose)

Define

- We would be using the melt function to melt the auralin and novodra columns to a treatment and a dose column (dose will still contain both start and end dose at this point). Then split the dose column on ' - ' to obtain start_dose and end_dose columns. and then drop the intermediate dose column.

Code

In [33]:

```
treatments_clean = pd.melt(treatments_clean, id_vars = ['given_name', 'surname', 'hba1c_sta
                           var_name = 'treatment', value_name = 'dose')
treatments_clean = treatments_clean[treatments_clean.dose != '-']
treatments_clean['dose_start'], treatments_clean['dose_end'] = treatments_clean['dose'].str
treatments_clean = treatments_clean.drop('dose', axis = 1)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3188\3471446428.py:4: FutureWarni
ng: Columnar iteration over characters will be deprecated in future release
s.
  treatments_clean['dose_start'], treatments_clean['dose_end'] = treatments_
clean['dose'].str.split(' - ', 1).str
```

Test

In [34]:

```
treatments_clean.head()
```

Out[34]:

	given_name	surname	hba1c_start	hba1c_end	hba1c_change	treatment	dose_start	dos
0	veronika	jindrová	7.63	7.20	0.43	auralin	41u	
3	skye	gormanston	7.97	7.62	0.35	auralin	33u	
6	sophia	haugen	7.65	7.27	0.38	auralin	37u	
7	eddie	archer	7.89	7.55	0.34	auralin	31u	
9	asia	woźniak	7.76	7.37	0.39	auralin	30u	

Tidiness

- Adverse reaction should be part of the treatments table

Define

- We would merge the adverse_reaction column to the treatments table, joining on given name and surname.

Code

In [35]:

```
treatments_clean = pd.merge(treatments_clean, adverse_reactions_clean,
                            on = ['given_name', 'surname'], how = 'left')
```

Test

In [36]:

```
treatments_clean.sample(5)
```

Out[36]:

	given_name	surname	hba1c_start	hba1c_end	hba1c_change	treatment	dose_start	do
157	alvin	jackson	7.62	7.23	0.39	auralin	38u	
266	fraser	hunter	7.70	7.42	0.28	novodra	36u	
232	finlay	sheppard	7.51	7.17	0.34	novodra	31u	
337	daimy	tromp	9.41	8.94	0.47	novodra	40u	
175	elliot	richardson	7.56	7.09	0.47	novodra	40u	

Tidiness

- Given name and surname columns in patients table are duplicated in treatments and adverse_reactions tables, and the given names and surnames in patients table needs to be changed to lowercase for proper joining with treatments table before dropping it

Define

- Adverse reactions table is no longer needed so we ignore that part. Isolate the patient ID and names in the patients table, then convert these names to lower case to join with treatments. Then drop the given name and surname columns in the treatments table (so these being lowercase isn't an issue anymore).

Code

In [37]:

```
id_names = patients_clean[['patient_id', 'given_name', 'surname']] # Creating a table named
id_names.given_name = id_names.given_name.str.lower() # Converting to lowercase
id_names.surname = id_names.surname.str.lower()
# Merging the id_names table from patients_clean table with the treatments_clean table
treatments_clean = pd.merge(treatments_clean, id_names,
                            on = ['given_name', 'surname'])
treatments_clean = treatments_clean.drop(['given_name', 'surname'], axis = 1)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3188\3692973624.py:2: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  id_names.given_name = id_names.given_name.str.lower() # Converting to lowe
rcase
C:\Users\user\AppData\Local\Temp\ipykernel_3188\3692973624.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  id_names.surname = id_names.surname.str.lower()
```

Test

In [38]:

```
# Confirming if the merge was executed correctly
treatments_clean.sample(8)
```

Out[38]:

	hba1c_start	hba1c_end	hba1c_change	treatment	dose_start	dose_end	adverse_reaction
77	7.55	7.12	0.43	auralin	44u	54u	NaN
340	7.92	7.52	0.40	novodra	26u	25u	NaN
157	7.62	7.23	0.39	auralin	38u	43u	NaN
282	7.98	7.51	0.47	novodra	30u	32u	NaN
206	7.67	7.29	0.38	novodra	19u	27u	NaN
179	8.08	7.70	0.38	novodra	54u	54u	NaN
347	7.96	7.51	0.45	novodra	28u	26u	hypoglycemia
121	7.84	7.41	0.43	auralin	24u	36u	NaN

In [39]:

```
#Only the patient_id should be duplicated
all_column = pd.Series(list(patients_clean) + list(treatments_clean))
all_column[all_column.duplicated()]
```

Out[39]:

```
22    patient_id
dtype: object
```

Quality

- Zip code is a float not a string and Zip code has four digits sometimes

Define

- Convert the zip code column's data type from a float to a string using astype, remove the '.0' using string slicing, and pad four digit zip codes with a leading 0.

Code

In [40]:

```
patients_clean.zip_code = patients_clean.zip_code.astype(str).str[:-2].str.pad(5, fillchar=
patients_clean.zip_code = patients_clean.zip_code.replace('0000n', np.nan)
```

Test

In [41]:

```
#Checking our code for proper execution
patients_clean.zip_code.head()
```

Out[41]:

```
0    92390
1    61812
2    68467
3    07095
4    36303
Name: zip_code, dtype: object
```

Tim Neudorf height is 27 in instead of 72 in

Define

- Replace height for rows in the patients table that have a height of 27 in (there is only one) with 72 in.

Code

In [42]:

```
patients_clean.height = patients_clean.height.replace(27, 72)
```

Test

In [43]:

```
# This shows our 27 has been replaced to 72
patients_clean[patients_clean.height == 27]
```

Out[43]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	country	birt

In [44]:

```
#Code properly executed
patients_clean[patients_clean.surname == 'Neudorf']
```

Out[44]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	country
4	5	male	Tim	Neudorf	1428 Turkey Pen Lane	Dothan	AL	36303	United States

Full state names sometimes, abbreviations other times

Define

- Apply a function that converts full state name to state abbreviation for California, New York, Illinois, Florida, and Nebraska.

Code

In [45]:

```
state_abbrev = {
    'California': 'CA',
    'New York': 'NY',
    'Illinois': 'IL',
    'Florida': 'FL',
    'Nebraska': 'NE'}
#Function to apply
def abbreviate_state (patient):
    if patient['state'] in state_abbrev.keys():
        abbrev = state_abbrev[patient['state']]
        return abbrev
    else:
        return patient['state']

patients_clean['state'] = patients_clean.apply(abbreviate_state, axis=1)
```

Test

In [46]:

```
patients_clean.state.value_counts()
```

Out[46]:

```
CA    60
NY    47
TX    32
IL    24
FL    22
MA    22
PA    18
GA    15
OH    14
MI    13
OK    13
LA    13
NJ    12
VA    11
WI    10
MS    10
AL     9
```

Dsvid Gustafsson

Define

- Replace given name for rows in the patients table that have a given name of 'Dsvid' with 'David'.

Code

In [47]:

```
patients_clean.given_name = patients_clean.given_name.replace('Dsvid', 'David')
```

Test

In [48]:

```
patients_clean[patients_clean.surname == 'Gustafsson']
```

Out[48]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	country
8	9	male	David	Gustafsson	1790 Nutter Street	Kansas City	MO	64105	United States

Erroneous datatypes (assigned sex, state, zip_code, and birthdate columns) and Erroneous datatypes (auralin and novodra columns) and The letter 'u' in starting and ending doses for Auralin and Novodra

Define

- Convert assigned sex and state to categorical data types. Zip code data type was already addressed above. Convert birthdate to datetime data type. Strip the letter 'u' in start dose and end dose and convert those columns to data type integer.

Code

In [49]:

```
#Changing assigned sex and state to categorical datatypes
patients_clean.assigned_sex = patients_clean.assigned_sex.astype('category')
patients_clean.assigned_sex = patients_clean.state.astype('category')

#Changing Birthdate to Datetime Datatype
patients_clean.birthdate = pd.to_datetime(patients_clean.birthdate)

#Strip the u in the dose_start and dose_end column and changing their datatype to integer i
treatments_clean.dose_start = treatments_clean.dose_start.str.strip('u').astype(int)
treatments_clean.dose_end = treatments_clean.dose_end.str.strip('u').astype(int)
```

Test

In [50]:

```
#Checking the patients table
patients_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 503 entries, 0 to 502
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   patient_id    503 non-null    int64         
 1   assigned_sex  491 non-null    category      
 2   given_name    503 non-null    object        
 3   surname       503 non-null    object        
 4   address       491 non-null    object        
 5   city          491 non-null    object        
 6   state         491 non-null    object        
 7   zip_code      491 non-null    object        
 8   country       491 non-null    object        
 9   birthdate     503 non-null    datetime64[ns]
 10  weight        503 non-null    float64       
 11  height        503 non-null    int64         
 12  bmi           503 non-null    float64       
 13  phone_number  491 non-null    object        
 14  email         491 non-null    object        
dtypes: category(1), datetime64[ns](1), float64(2), int64(2), object(9)
memory usage: 57.1+ KB
```

In [51]:

```
treatments_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 349 entries, 0 to 348
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   hba1c_start       349 non-null    float64
 1   hba1c_end         349 non-null    float64
 2   hba1c_change      349 non-null    float64
 3   treatment         349 non-null    object 
 4   dose_start        349 non-null    int32  
 5   dose_end          349 non-null    int32  
 6   adverse_reaction  35 non-null     object 
 7   patient_id        349 non-null    int64  
dtypes: float64(3), int32(2), int64(1), object(2)
memory usage: 21.8+ KB
```

In [52]:

```
# Checking if the u have been stripped
treatments_clean.sample(5)
```

Out[52]:

	hba1c_start	hba1c_end	hba1c_change	treatment	dose_start	dose_end	adverse_reaction
236	7.59	7.13	0.46	novodra	26	25	NaN
327	7.71	7.30	0.41	novodra	33	33	NaN
29	7.74	7.32	0.42	auralin	61	67	NaN
220	7.76	7.35	0.41	novodra	34	32	NaN
148	7.95	7.60	0.35	auralin	46	57	cough

Multiple phone number formats

Define

- Strip all " ", "-", "(", ")", and "+" and store each number without any formatting. Pad the phone number with a 1 if the length of the number is 10 digits (we want country code).

Code

In [53]:

```
patients_clean.phone_number = patients_clean.phone_number.str.replace(r'\D+', '').str.pad(1
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_3188\3922059896.py:1: FutureWarni
ng: The default value of regex will change from True to False in a future ve
rsion.
  patients_clean.phone_number = patients_clean.phone_number.str.replace(r'\D
+', '').str.pad(11, fillchar='1')
```

Test

In [54]:

```
patients_clean.phone_number.head()
```

Out[54]:

```
0    19517199170
1    12175693204
2    14023636804
3    17326368246
4    13345157487
Name: phone_number, dtype: object
```

Default John Doe data

Define

- Remove the non-recoverable John Doe records from the patients table.

Code

In [55]:

```
patients_clean = patients_clean[patients_clean.surname != 'Doe']
```

Test

In [56]:

```
# There should be no Doe records
patients_clean.surname.value_counts()
```

Out[56]:

```
Jakobsen       3
Taylor         3
Aranda         2
Tucker         2
Souza          2
              ..
Casárez        1
Mata           1
Pospíšil       1
Rukavina       1
Onyekaozulu    1
Name: surname, Length: 465, dtype: int64
```

In [57]:

```
#There should be no 123 Main Street records
patients_clean.address.value_counts()
```

Out[57]:

```
2778 North Avenue           2
2476 Fulton Street          2
648 Old Dear Lane           2
576 Brown Bear Drive        1
2272 Williams Avenue        1
                          ..
1066 Goosetown Drive        1
4291 Patton Lane            1
4643 Reeves Street          1
174 Lost Creek Road         1
3652 Boone Crockett Lane    1
Name: address, Length: 482, dtype: int64
```

Multiple records for Jakobsen, Gersten, Taylor

Define

- Remove the Jake Jakobsen, Pat Gersten, and Sandy Taylor rows from the patients table. These are the nicknames, which happen to also not be in the treatments table (removing the wrong name would create a consistency issue between the patients and treatments table). These are all the second occurrence of the duplicate. These are also the only occurences of non-null duplicate addresses.

Code

In [58]:

```
# tilde means not
patients_clean = patients_clean[~((patients_clean.address.duplicated()) & patients_clean.ad
```

Test

In [59]:

```
patients_clean[patients_clean.surname == 'Jakobsen']
```

Out[59]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	count
24	25	NY	Jakob	Jakobsen	648 Old Dear Lane	Port Jervis	NY	12771	Unit Stat
432	433	TX	Karen	Jakobsen	1690 Fannie Street	Houston	TX	77020	Unit Stat

In [60]:

```
patients_clean[patients_clean.surname == 'Gersten']
```

Out[60]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	country	b
97	98	NE	Patrick	Gersten	2778 North Avenue	Burr	NE	68324	United States	

In [61]:

```
patients_clean[patients_clean.surname == 'Taylor']
```

Out[61]:

	patient_id	assigned_sex	given_name	surname	address	city	state	zip_code	count
131	132	WV	Sandra	Taylor	2476 Fulton Street	Rainelle	WV	25962	Unite Stat
426	427	FL	Rogelio	Taylor	4064 Marigold Lane	Miami	FL	33179	Unite Stat

kgs instead of lbs for Zaitseva weight

Define

- Use advanced indexing to isolate the row where the surname is Zaitseva and convert the entry in its weight field from kg to lbs.

Code

In [62]:

```
weight_kg = patients_clean.weight.min()
mask = patients_clean.surname == 'Zaitseva'
column_name = 'weight'
patients_clean.loc[mask, column_name] = weight_kg * 2.20462
```

Test

In [63]:

```
# 48.8 shouldn't be the lowest anymore
patients_clean.weight.sort_values()
```

Out[63]:

```
459    102.1
335    102.7
74     103.2
317    106.0
171    106.5
       ...
144    244.9
61     244.9
283    245.5
118    254.5
485    255.9
Name: weight, Length: 494, dtype: float64
```

In [ ]:

In [ ]: