DATA EXPLORATION AND MINING

FEB 2021/ GROUP ASSESSMENT 1

Exploration of a large medical dataset

Question 1

Have identified 3 variables: Gender (Male/Female), Change (Ch ange/No change) and DiabetesMed (Yes/No). Since they are binary in nature, these 3 variables can be mapped int o 0 and 1.

Original data type

```
# read in diabetic patient dataset
df = pd.read_csv('D1.csv')

print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51766 entries, 0 to 51765
Data columns (total 37 columns):
encounter_id                51766 non-null int64
patient_nbr                 51766 non-null int64
race                        51766 non-null object
gender                      51766 non-null object
age                         51766 non-null object
weight                      51766 non-null object
admission_type_id           51766 non-null int64
discharge_disposition_id    51766 non-null int64
admission_source_id         51766 non-null int64
time_in_hospital            51766 non-null int64
payer_code                  51766 non-null object
medical_specialty           51766 non-null object
num_lab_procedures          51766 non-null int64
num_procedures              51766 non-null int64
num_medications             51766 non-null int64
number_outpatient           51766 non-null int64
number_emergency            51766 non-null int64
number_inpatient            51766 non-null int64
diag_1                      51766 non-null object
diag_2                      51766 non-null object
diag_3                      51766 non-null object
number_diagnoses            51766 non-null int64
max_glu_serum               51766 non-null object
A1Cresult                   51766 non-null object
metformin                   51766 non-null object
repaglinide                 51766 non-null object
nateglinide                 51766 non-null object
chlorpropamide              51766 non-null object
glimepiride                 51766 non-null object
acetohexamide               51766 non-null object
glipizide                   51766 non-null object
glyburide                   51766 non-null object
tolbutamide                 51766 non-null object
insulin                     51766 non-null object
change                      51766 non-null object
diabetesMed                 51766 non-null object
readmitted                  51766 non-null object
dtypes: int64(13), object(24)
memory usage: 14.6+ MB
None
```

binary

binary

binary

Map gender to binary (int64)

```
# map gender into binary variable

gender_binary = {'Female':0, 'Male':1, 'Unknown/Invalid':np.NaN}
df['gender'] = df['gender'].map(gender_binary)
```

```
df['gender'].value_counts()
```

```
0.0    27718
1.0    24044
Name: gender, dtype: int64
```

Map change to binary (int64)

```
# map change into binary variable

change_binary = {'No':0, 'Ch':1}
df['change'] = df['change'].map(change_binary)
```

```
df['change'].value_counts()
```

```
1    26663
0    25103
Name: change, dtype: int64
```

Map diabetesMed to binary (int64)

```
# map diabetesMed into binary variable

diabetesMed_binary = {'No':0, 'Yes':1}
df['diabetesMed'] = df['diabetesMed'].map(diabetesMed_binary)
```

```
df['diabetesMed'].value_counts()
```

```
1    40984
0    10782
Name: diabetesMed, dtype: int64
```

Question 2

Using suitable statistical measures and functions:

1) Identify and report the skewness present in the variables.

As question 3 will use visualization, for this question we f ocus on descriptive statistics to view the numeric data.

in_hospital	num_lab_procedures	num_procedures	num_medications	number_outpatient	number_emergency	number_inpatient	number_diagnoses
766.000000	51766.000000	51766.000000	51766.000000	51766.000000	51766.000000	51766.000000	51766.000000
4.237337	43.853205	1.308214	16.735850	0.512846	0.269173	0.686879	7.928486
2.870999	19.945471	1.721694	8.063522	1.544866	1.146923	1.320968	1.693707
1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000
2.000000	33.000000	0.000000	11.000000	0.000000	0.000000	0.000000	7.000000
3.000000	45.000000	1.000000	16.000000	0.000000	0.000000	0.000000	9.000000
6.000000	58.000000	2.000000	21.000000	0.000000	0.000000	1.000000	9.000000
14.000000	132.000000	6.000000	75.000000	42.000000	76.000000	19.000000	16.000000

Below are the numeric variables:

- time_in_hospital: mean = 4.237 337 , median = 3 ==> mean > median ==> positive skewed
- num_lab_procedures: mean = 43.853 205 , median = 45 ==> mean < median ==> negative skewed
- num_procedures: mean = 1.308214 median = 1 ==> mean > median ==> posit ive skewed
- num_medications: mean = 16.7359, median = 16 ==> mean > median ==> po sitive skewed

- number_outpatient: mean = 0.512846, median = 0 ==> mean > median ==> posi tive skewed
- number_emergency: mean = 0.269173, median = 0 ==> mean > median ==> neg ative skewed
- number_inpatient: mean = 0.686879, median = 0 ==> mean > median ==> posi tive skewed
- number_diagnoses: mean = 7.928486, median = 9 ==> mean < median ==> neg ative skewed

Below are the nominal variables. Hence descriptive statistics are n ot applicable.

- encounter_id
- patient_nbr
- gender
- admission_type_id
- discharge_disposition_id
- admission_source_id
- change
- diabetesMed

2) There may be inconsistencies or errors in the data. List the erro rs identified and detail how you have identified them.

List out and review the unique values in each variable to i dentify if there are any unusual values.

```
# Looking at unique values in the dataset and see if there is any unusal values

for col in list(df):
    print(col)
    print(df[col].unique())
```

```
encounter_id
[150645834 150646782 150659562 ... 443854148 443857166 443867222]
patient_nbr
[ 66173058 104923647  30917367 ... 140199494 120975314 175429310]
race
['Caucasian' 'AfricanAmerican' 'Hispanic' '?' 'Other' 'Asian']
gender
[ 0.  1. nan]
age
['[60-70)' '[80-90)' '[70-80)' '[40-50)' '[50-60)' '[90-100)' '[30-40)'
 '[20-30)' '[10-20)' '[0-10)' '?']
weight
['?' '[100-125)' '[50-75)' '[75-100)' '[0-25)' '[125-150)' '[25-50)'
 '[150-175)' '[175-200)' '>200']
admission_type_id
[2 3 1 5 6 8 7 4]
discharge_disposition_id
[ 1  2  6  3 22 11  7 23 13  4 15  9 28  5 19  8 25 14 24 27 18]
admission_source_id
[ 1  7  5 17  4  2  3  6  9 10 22 20  8 14 11 25 13]
time_in_hospital
[ 4  8  2  7  3  6  5  1 13 12 11  9 10 14]
payer_code
['?' 'MD' 'BC' 'MC' 'HM' 'CP' 'SP' 'OG' 'UN' 'DM' 'CM' 'PO' 'SI' 'WC' 'CH'
 'OT' 'MP' 'FR']
```

Have identified the following var iables with “?” as values:

- race
- a ge
- weight
- chlorpropamide
- payer_code
- medical_specialty
- diag_2

Then check to see how many "?" in each variable:

```
# view the number of '?' data

num_missing = (df[['race', 'age', 'weight', 'chlorpropamide', 'payer_code', 'medical_specialty',
                   'diag_2']] == "?").sum()
print(num_missing)
```

```
race                 1016
age                    10
weight              50431
chlorpropamide          9
payer_code           7601
medical_specialty   32203
diag_2                 92
dtype: int64
```

3a) What is the average time stay in hospital of a female pa tient who was readmitted in less than 30 days?

4.524752 days

```
print(df.groupby(['gender','readmitted'], as_index=False)['time_in_hospital'].mean())
# Q1 mapped 0: Female
```

	gender	readmitted	time_in_hospital
0	0.0	<30	4.524752
1	0.0	>30	4.465249
2	0.0	NO	4.207442
3	1.0	<30	4.482972
4	1.0	>30	4.192097
5	1.0	NO	4.012216

3b) Which age group has the highest risk of being readmitted wit hin 30 days? 70-80 has the highest risk

```
df_readmitted_age_grouped = df.groupby(['age'])['readmitted'].value_counts()
df_readmitted_age_grouped
```

```
[40-50)   NO     2611
          >30    1537
          <30     471
[50-60)   NO     4789
          >30    2905
          <30     788
[60-70)   NO     6270
          >30    4074
          <30    1274
[70-80)   NO     6770
          >30    4767
          <30    1435
[80-90)   NO     4930
          >30    3611
          <30    1130
[90-100)  NO      940
          >30     483
          <30     193
Name: readmitted, dtype: int64
```

3c) How many age groups have more than 3000 cases of being rea dmitted? 4 age groups

```
filter_readmitted = df[(df['readmitted']=='>30') | (df['readmitted']=='<30')]
readmitted_by_age = filter_readmitted['age'].value_counts()
readmitted_by_age
```

```
[70-80)     6202
[60-70)     5348
[80-90)     4741
[50-60)     3693
[40-50)     2008
[30-40)      709
[90-100)     676
[20-30)      351
[10-20)       91
[0-10)         6
?              3
Name: age, dtype: int64
```

3d) Which are the top-three race categories according to the number of readmission cases?

1st - Caucasian, 2nd - AfricanAmerican, 3rd - Other (NOTE: excluding '?')

```
readmitted_by_race = filter_readmitted['race'].value_counts()
readmitted_by_race
```

```
Caucasian          19120
AfricanAmerican     3530
?                    367
Other                347
Hispanic             344
Asian                120
Name: race, dtype: int64
```

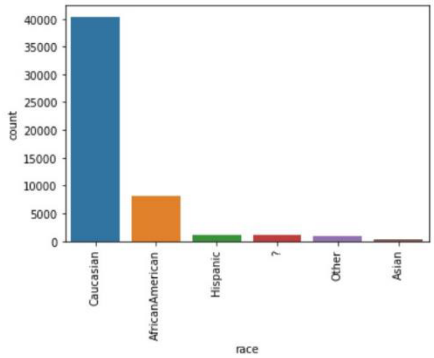
Question 3

Using suitable visualisation plots:

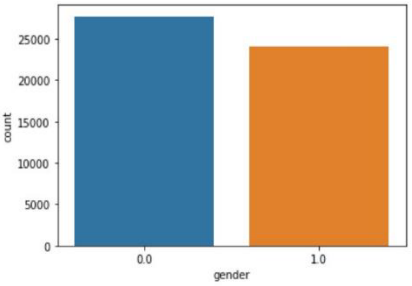
1) Understand the distribution of variables and identify data quality problems.

Findings are summarized in # lines on top of each plot.

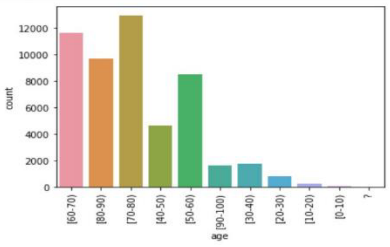
```
# race
race_plot = sns.countplot(data=df, x='race')
race_plot.set_xticklabels(race_plot.get_xticklabels(),rotation=90)
plt.show()

# missing data with '?', Caucasian is the majority
```



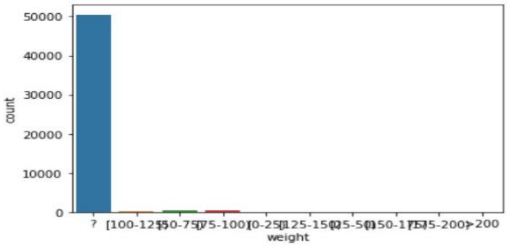

```
# gender (Female = 0, Male = 1)
gender_plot = sns.countplot(data=df, x='gender')
plt.show()

# Not include missing data with 'nan' in plot, about equal shares in gender
```



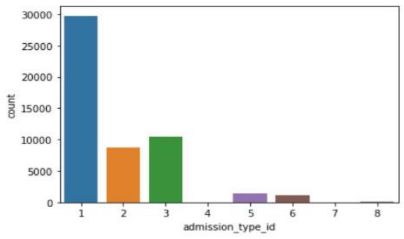

```
# age
age_plot = sns.countplot(data=df, x='age')
age_plot.set_xticklabels(age_plot.get_xticklabels(),rotation=90)
plt.show()

# missing data with "?", +ive skewed, 40 and above are the majority age group
```



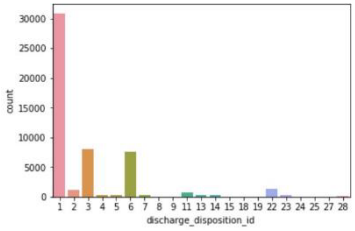

```
# weight
weight_plot = sns.countplot(data=df, x='weight')
plt.show()

# 50431 records are '?' (97.42% of data)
```



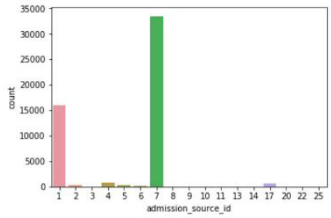

```
# admission type
admit_type_plot = sns.countplot(data=df, x='admission_type_id')
plt.show()

# Majority admitted under Emergency
```



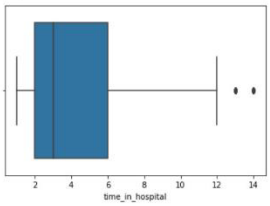

```
# discharge disposition
discharge_plot = sns.countplot(data=df, x='discharge_disposition_id')
plt.show()

# majority is discharged to home. followed by discharged/transferred to SNF and
# discharged/transferred to home with home health service
```



```
# admission source
admit_source_plot = sns.countplot(data=df, x='admission_source_id')
plt.show()

# 1: Physician Referral, 7: Emergency Room
# majority are admitted via emergency room, which co-incide with admission type
```

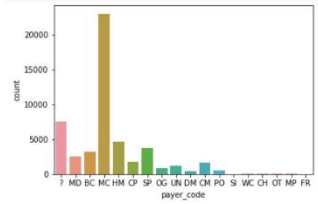



```
# time in hospital
time_plot = sns.boxplot(df['time_in_hospital'])
plt.show()

# positive skewed. 75% majority is below 6 days. median of around 3 days.
#  with outliers of 13 & 14 days

# 977 records are >3sd (1.89% of data)
time_in_hospital_outliers = (df['time_in_hospital'] > (4.237+2.870999*3)).value_counts()
time_in_hospital_outliers
```



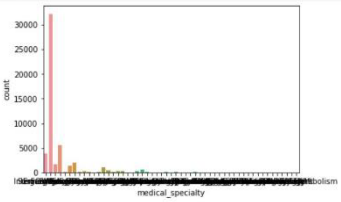

```
False    50789
True       977
Name: time_in_hospital, dtype: int64
```

```
# payer_code (Unique identifier assigned to each insurance company)

payer_code_plot = sns.countplot(data=df, x='payer_code')
plt.show()

df['payer_code'].value_counts()

# missing data with "?"  7601 (about 15% of total data!)
# majority is 'MC'
```



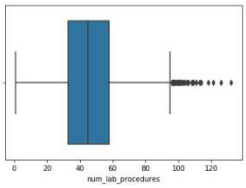

```
MC     23046
?       7601
HM      4702
```

```
# medical_specialty (Indicates specialty of the admitting physician,for example, cardiology, surgeon, etc.)

medical_specialty_plot = sns.countplot(data=df, x='medical_specialty')
plt.show()

df['medical_specialty'].value_counts()

# missing data with "?"  32203 (about 62% of total data!)
# majority is 'InternalMedicine'
```



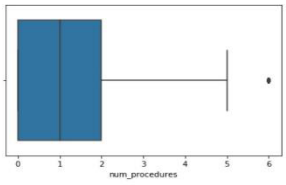

```
?                          32203
Emergency/Trauma            5616
InternalMedicine            3854
```

```
# number of lab procedures
lab_procedures_plot = sns.boxplot(df['num_lab_procedures'])
plt.show()

# Per statistic median>mean, it is  ive skewed . With outliers.
# 20 records are >3sd
lab_procedures_outliers = (df['num_lab_procedures'] > (43.853205+19.945471*3)).value_counts()
lab_procedures_outliers
```




```
False    51746
True        20
Name: num_lab_procedures, dtype: int64
```

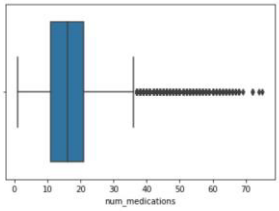
```
# number of procedures (other than lab procedures)
no_procedures_plot = sns.boxplot(df['num_procedures'])
plt.show()

# positive skewed with outlier. Median is 1.
# 2873 records are >3sd (5.55%)
num_procedures_outliers = (df['num_procedures'] == 6).value_counts()
num_procedures_outliers
```

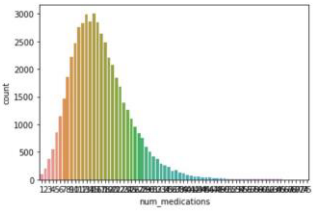



```
False    48893
True      2873
Name: num_procedures, dtype: int64
```

[49]:

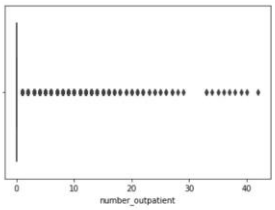
```
# number of medications
no_medications_plot = sns.boxplot(df['num_medications'])
plt.show()

no_medications_countplot = sns.countplot(data=df, x='num_medications')
plt.show()

# positive skewed. Median of 15
# 626 records are >3sd (1.21% of data)
no_medications_outliers = (df['num_medications'] > (16.735850+8.063522*3)).value_counts()
no_medications_outliers
```

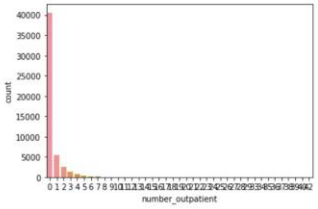




Out[49]:
```
False    51140
True       626
Name: num_medications, dtype: int64
```

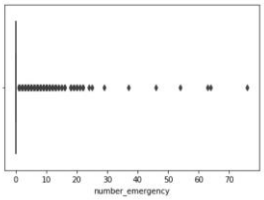
```
# Number of outpatient visits of the patient in the year preceding the encounter
no_outpatient_plot = sns.boxplot(df['number_outpatient'])
plt.show()

no_outpatient_count = sns.countplot(data=df, x='number_outpatient')
plt.show()

# positive skewed. Majority is 0. With outliers
# 737 records are >3sd (1.42% of data)
number_outpatient_outliers = (df['number_outpatient'] > (0.512846+1.544866*3)).value_counts()
number_outpatient_outliers
```

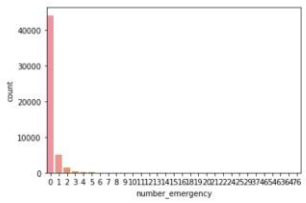





```
False    51029
True       737
Name: number_outpatient, dtype: int64
```

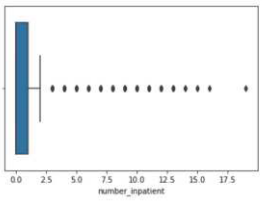
```
# Number of emergency visits of the patient in the year preceding the encounter
no_outpatient_plot = sns.boxplot(df['number_emergency'])
plt.show()

no_outpatient_count = sns.countplot(data=df, x='number_emergency')
plt.show()

# positive skewed. Majority is 0 with outliers
# 667 records are >3sd (1.29% of data)
number_emergency_outliers = (df['number_emergency'] > (0.269173+1.146923*3)).value_counts()
number_emergency_outliers
```

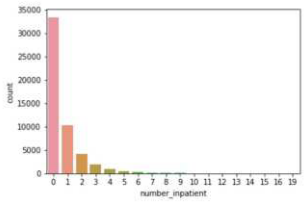





```
False    51099
True       667
Name: number_emergency, dtype: int64
```

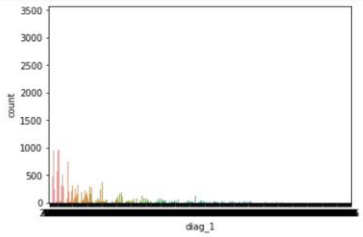
```
# Number of inpatient visits of the patient in the year preceding the encounter
no_outpatient_plot = sns.boxplot(df['number_inpatient'])
plt.show()

no_outpatient_count = sns.countplot(data=df, x='number_inpatient')
plt.show()

# positive skewed. Majority is 0 with outliers
# 1146 records are >3sd (2.21% of data)
number_inpatient_outliers = (df['number_inpatient'] > (0.686879+1.320968*3)).value_counts()
number_inpatient_outliers
```





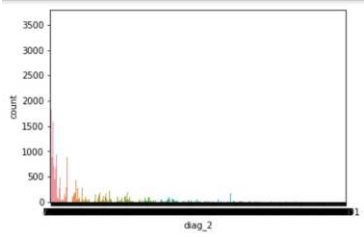

```
False    50620
True      1146
Name: number_inpatient, dtype: int64
```

```
# diag_1 (primary diagnosis), in accordance with first three digits of ICD9
diag_1_count = sns.countplot(data=df, x='diag_1')
plt.show()

# majority is '428' (Congestive heart failure, unspecified) and
#  '414' (Other forms of chronic ischemic heart disease)
#  there are 649 types of diagnosis
df['diag_1'].value_counts()
```



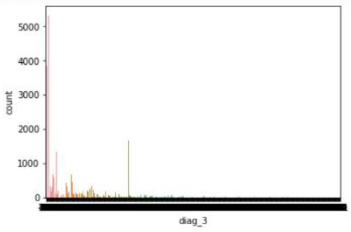

```
428      3395
414      2746
486      1808
```

```
# diag_2 (secondary diagnosis), in accordance with first three digits of ICD9
diag_2_count = sns.countplot(data=df, x='diag_2')
plt.show()

# majority is '276' (Hyperosmolality and/or hypernatremia) and
#  '428' (Congestive heart failure, unspecified)
#   there are 683 types of diagnosis
df['diag_2'].value_counts()
```



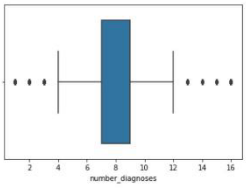

```
276      3604
428      3177
250      2685
```

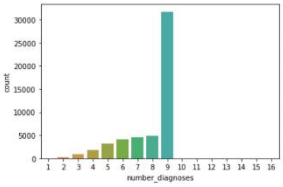
```
# diag_3 (additional secondary)), in accordance with first three digits of ICD9

diag_3_count = sns.countplot(data=df, x='diag_3')
plt.show()

# majority is '250' (Diabetes), '401' (Malignant essential hypertension)
# there are 726 types of diagnosis
df['diag_3'].value_counts()
```



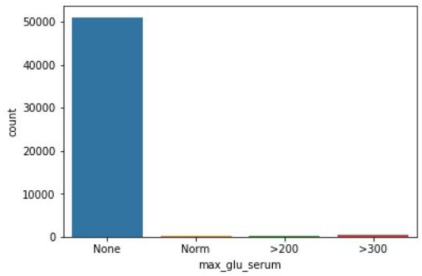

```
250      5323
401      3861
276      2968
```

```
# Number of diagnoses entered to the system
no_diagnoses_plot = sns.boxplot(df['number_diagnoses'])
plt.show()

no_diagnoses_count = sns.countplot(data=df, x='number_diagnoses')
plt.show()

# mean = 7.928485878762122, median = 0, median > mean ==> negative skewed
# outliers on both sides

# 62 records are >3sd  and 304 records are <-3sd (0.71% of data)
number_diagnoses_outliers = (df['number_diagnoses'] > (7.928485878762122+1.693707*3)).value_counts()
number_diagnoses_outliers

number_diagnoses_outliers = (df['number_diagnoses'] < (7.928485878762122-1.693707*3)).value_counts()
number_diagnoses_outliers
```

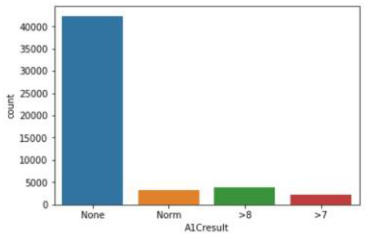




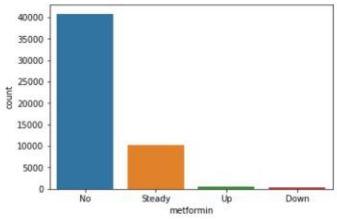
```
# Glucose serum test result.
#  Indicates the range of the result or if the test was not taken("none")
max_glu_serum_count = sns.countplot(data=df, x='max_glu_serum')
plt.show()

# majority are None
```



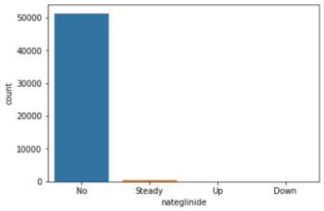

```
# A1c test result.

# Values include:'>8' if the result was greater than 8%,
# ">7" if the result was greater than 7% but less than 8%,
# "normal" if the result was less than 7%, and
# "none" if not measured

A1Cresult_count = sns.countplot(data=df, x='A1Cresult')
plt.show()

# Majority are None.
```



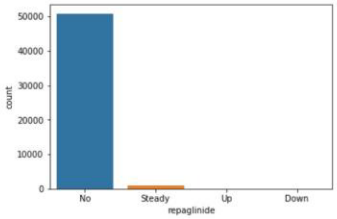

```
# metformin

# A diabetes medications
# The values of these variables indicate whether the drug was pres
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

metformin_count = sns.countplot(data=df, x='metformin')
plt.show()

# majority are not prescribed
```



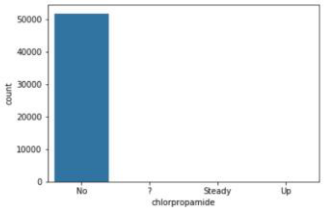

```
# nateglinide

# A diabetes medications
# The values of these variables indicate whether the drug was prescr
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

nateglinide_count = sns.countplot(data=df, x='nateglinide')
plt.show()

# majority are not prescribed
```



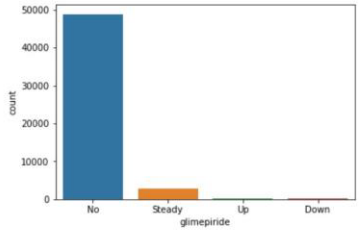

```
# repaglinide

# A diabetes medications
# The values of these variables indicate whether the drug was pres
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

repaglinide_count = sns.countplot(data=df, x='repaglinide')
plt.show()

# majority are not prescribed
```



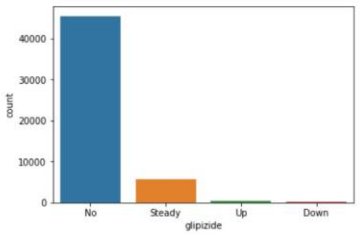

```
# chlorpropamide

# A diabetes medications
# The values of these variables indicate whether the drug was prescr
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

chlorpropamide_count = sns.countplot(data=df, x='chlorpropamide')
plt.show()

# there is '?' values
# majority are not prescribed
```



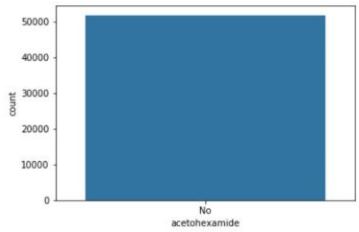
```
# glimepiride

# A diabetes medications
# The values of these variables indicate whether the drug was prescribe
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

glimepiride_count = sns.countplot(data=df, x='glimepiride')
plt.show()

# majority are not prescribed
```



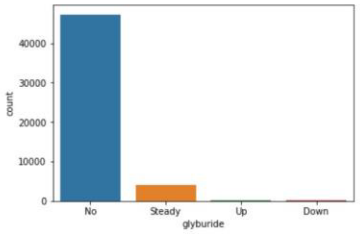

```
# glipizide

# A diabetes medications
# The values of these variables indicate whether the drug was prescribe
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

glipizide_count = sns.countplot(data=df, x='glipizide')
plt.show()

# majority are not prescribed
```



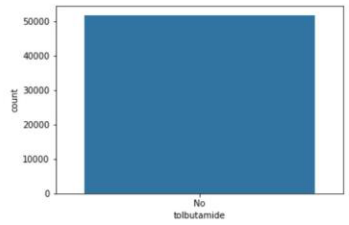

```
# acetohexamide

# A diabetes medications
# The values of these variables indicate whether the drug was prescribe
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

acetohexamide_count = sns.countplot(data=df, x='acetohexamide')
plt.show()

# NO patient is prescribed to it
```



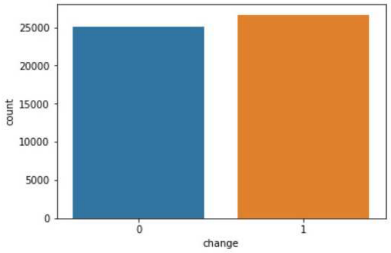

```
# glyburide

# A diabetes medications
# The values of these variables indicate whether the drug was prescribe
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

glyburide_count = sns.countplot(data=df, x='glyburide')
plt.show()

# majority are not prescribed
```



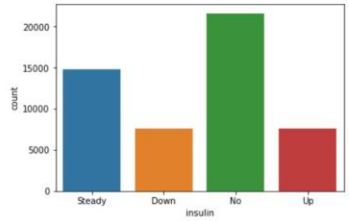

```
# tolbutamide

# A diabetes medications
# The values of these variables indicate whether the drug was prescribed
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

tolbutamide_count = sns.countplot(data=df, x='tolbutamide')
plt.show()

# NO patient is prescribed to it
```



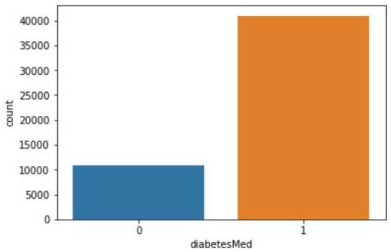

```
# Change of medications
# Indicates if there was a change in diabetic medications (either dosage or generic name)
# 1: "change"and 0: "no change"

change_count = sns.countplot(data=df, x='change')
plt.show()

# roughly similiar
```



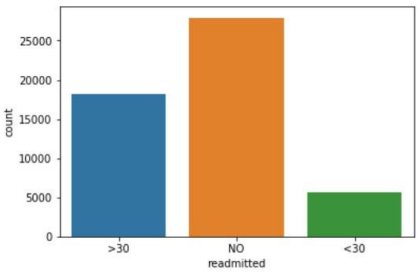

```
# insulin

# A diabetes medications
# The values of these variables indicate whether the drug was prescribed
# "up" if the dosage was increased during the encounter,
# "down" if the dosage was decreased,
# "steady" if the dosage did not change, and
# "no" if the drug was not prescribed.

insulin_count = sns.countplot(data=df, x='insulin')
plt.show()

# Good number of patients are on insulin
```




```
# Diabetes medications
# Indicates if there was any diabetic medication prescribed
# 1: "yes" and 0: "no"

diabetesMed_count = sns.countplot(data=df, x='diabetesMed')
plt.show()

# majority are not on diabetes medications
```



```
# Days to inpatient readmission.
# "<30" if the patient was readmitted in less than 30 days,
# ">30" if the patient was readmitted in more than 30 days, and
# "No" for no record of readmission.

readmitted_count = sns.countplot(data=df, x='readmitted')
plt.show()

# many are readmitted, majority of which are readmitted >30 days
```



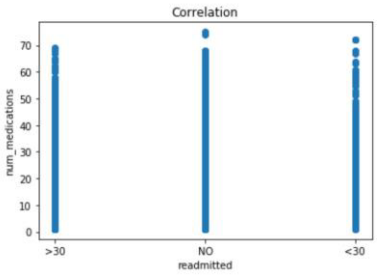
2) Determine if there is any relationship between the variabl es num_medications and readmitted? How would you handle these two variables in the data modelling if a rel ationship exists?

Low correlation of 0.067557 existed between num_medications and readmitted. If high correlat ion do exist between features, it could cause multicollinearity and affect the mod el. Therefore, removing correlated features is recommended.

```
# Q3(2) Relationship between num_medications (numeric) and readmitted (categorical)
# Very low correlation of 0.067557

cat_map={">30":2, "<30":1, "NO":0}
df2=df
df2['readmitted']=df_new['readmitted'].map(cat_map)
df1=pd.DataFrame(df2, columns=['readmitted', 'num_medications'])
df1.corr(method='spearman')
```

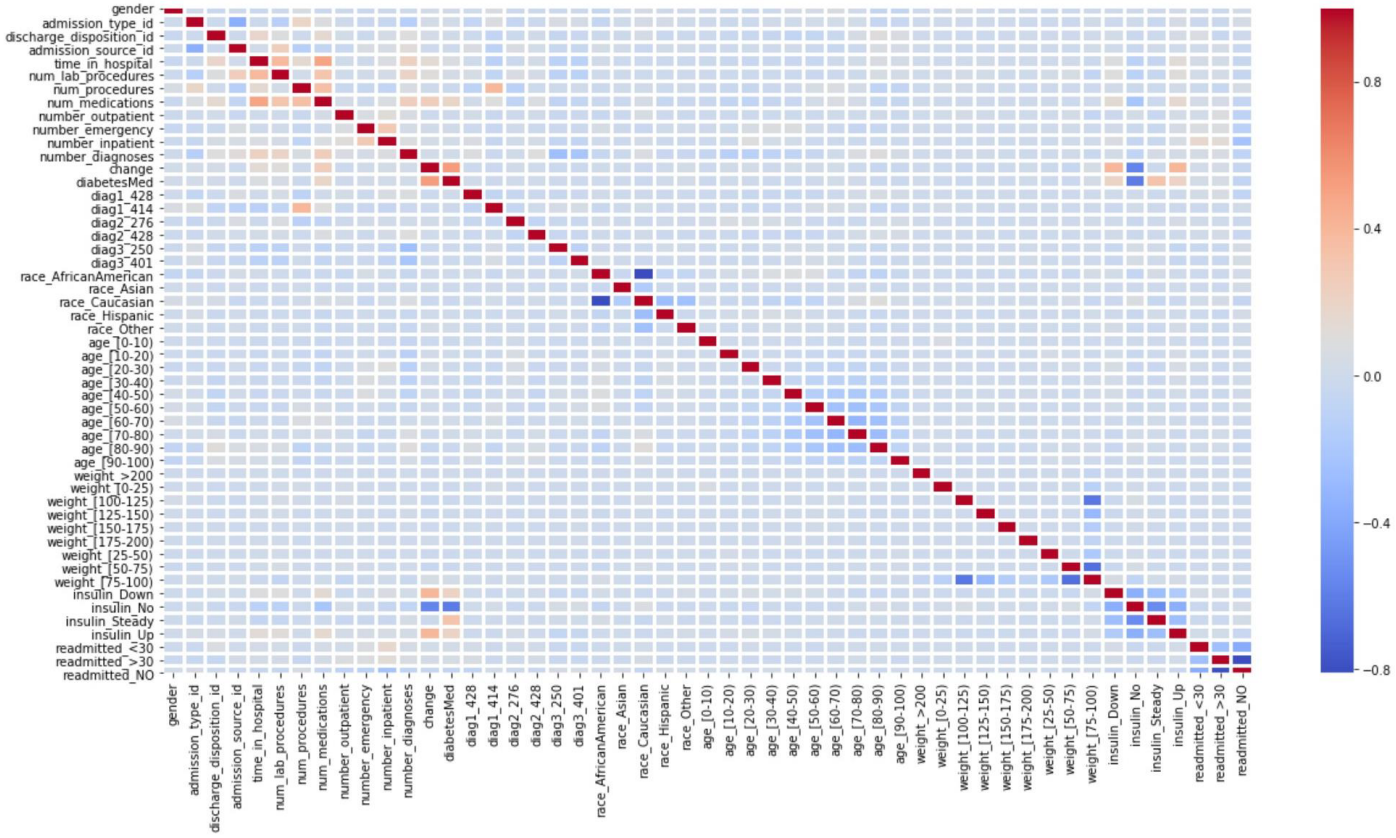
	readmitted	num_medications
readmitted	1.000000	0.067557
num_medications	0.067557	1.000000

```
# Q3(2) Relationship between num_medications and readmitted
plt.scatter(df['readmitted'], df['num_medications'])
plt.title('Correlation')
plt.xlabel('readmitted')
plt.ylabel('num_medications')
plt.show()
```



3) Identify the highly correlated variable pairs and elaborate on how to treat these variables in the mining process in such a case.

A heatmap is used to see the overall correlation between variables:



Taking a deeper look at the variable with high correlation (the dark blue):

```
# What is the correlation b/w race_AfricanAmerican and race_Caucasian
race_corr=pd.DataFrame(df_dummy, columns=['race_AfricanAmerican', 'race_Caucasian'])
race_corr.corr(method='spearman')
```

	race_AfricanAmerican	race_Caucasian
race_AfricanAmerican	1.000000	-0.810038
race_Caucasian	-0.810038	1.000000

```
# What is the correlation b/w readmitted_>30 and readmitted_NO
readmitted_corr=pd.DataFrame(df_dummy, columns=['readmitted_>30', 'readmitted_NO'])
readmitted_corr.corr(method='spearman')
```

	readmitted_>30	readmitted_NO
readmitted_>30	1.000000	-0.797729
readmitted_NO	-0.797729	1.000000

1) race_AfricanAmerican is highly negatively correlated with race_Caucasian (-81%) ==> came from same variable "race"

2) readmitted_30 is highly negatively correlated with readmitted_NO (-79.77%) ==> came from same variable "readmitted"

Will use race and readmitted as variables as a whole and not it's sub-category.

Question 4
Summarization and data preparation:
1) Summarize your findings based on data exploration.

Error in data (values with '?')

- race - 1016 records (1% of data)
- age - 10 records
- weight - 50431 records (about 97% of total data)
- chlorpropamide - 9 records
- payer_code - 7601 records (about 15% of total data)
- medical_specialty - 32203 records (about 62% of total data)
- diag_2 - 92 records (0.18% of total data)

NaN values

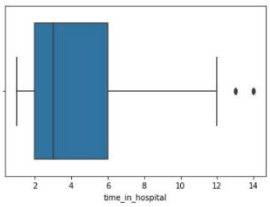
- gender 4 records - mapped from 'Unknown/Invalid':np.NaN

Outliers

They are not noise. They are anomalies. Do not drop.

- time_in_hospital - 977 records are >3sd - (1.89% of data) (positive skewed. 75% majority is below 6 days) (mean = 4.237, sd = 2.87, median = 3, min = 1, 1stQ = 2, 3rdQ = 6, max = 14)

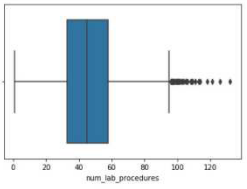
```
# time in hospital
time_plot = sns.boxplot(df['time_in_hospital'])
plt.show()

# positive skewed. 75% majority is below 6 days. median of around 3 days.
#  with outliers of 13 & 14 days

# 977 records are >3sd (1.89% of data)
time_in_hospital_outliers = (df['time_in_hospital'] > (4.237+2.870999*3)).value_counts()
time_in_hospital_outliers
```




```
False    50789
True       977
Name: time_in_hospital, dtype: int64
```

- num_lab_procedures - 20 records are >3sd - (median>mean, it is -ive skewed.) (mean = 43.853205, sd = 19.945471, median = 45, min = 1, 1stQ = 33, 3rdQ = 58, max = 132)

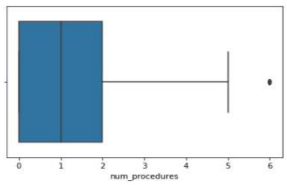
```
# number of lab procedures
lab_procedures_plot = sns.boxplot(df['num_lab_procedures'])
plt.show()

# Per statistic median>mean, it is -ive skewed . With outliers.
# 20 records are >3sd
lab_procedures_outliers = (df['num_lab_procedures'] > (43.853205+19.945471*3)).value_counts()
lab_procedures_outliers
```




```
False    51746
True        20
Name: num_lab_procedures, dtype: int64
```

- num_procedures - 2873 records are >3sd (5.55%) - (positive skewed with outliers Median is 1 procedure) (mean = 1.308214, sd = 1.721694, median = 1, min = 0, 1stQ = 0, 3rdQ = 2, max = 6)

```
# number of procedures (other than lab procedures)
no_procedures_plot = sns.boxplot(df['num_procedures'])
plt.show()

# positive skewed with outlier. Median is 1.
# 2873 records are >3sd (5.55%)
num_procedures_outliers = (df['num_procedures'] == 6).value_counts()
num_procedures_outliers
```

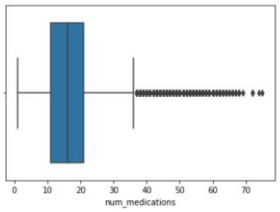


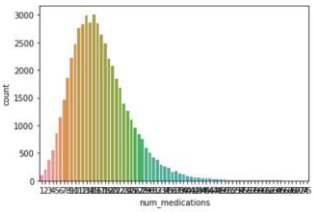

```
False    48893
True      2873
Name: num_procedures, dtype: int64
```

- num_medications - 626 records are >3sd (1.21% of data) - (positive skewed. with outliers Median of 15) (mean = 16.735850, sd = 8.063522, median = 16, min = 1, 1stQ = 11, 3rdQ = 21, max = 75)

[49]:

```
# number of medications
no_medications_plot = sns.boxplot(df['num_medications'])
plt.show()

no_medications_countplot = sns.countplot(data=df, x='num_medications')
plt.show()

# positive skewed. Median of 15
# 626 records are >3sd (1,21% of data)
no_medications_outliers = (df['num_medications'] > (16.735850+8.063522*3)).value_counts()
no_medications_outliers
```

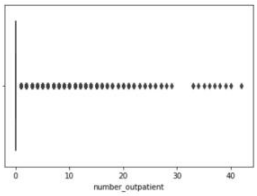


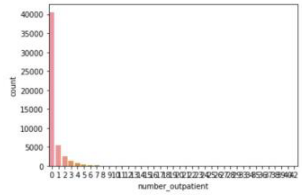


Out[49]:
```
False    51140
True       626
Name: num_medications, dtype: int64
```

- number_outpatient - 737 records are >3sd (1.42% of data) - (positive skewed. Majority is 0) (mean = 0.512846, sd = 1.544866, median = 0, min = 0, 1stQ = 0, 3rdQ = 0, max = 42)

```
# Number of outpatient visits of the patient in the year preceding the encounter
no_outpatient_plot = sns.boxplot(df['number_outpatient'])
plt.show()

no_outpatient_count = sns.countplot(data=df, x='number_outpatient')
plt.show()

# positive skewed. Majority is 0. With outliers
# 737 records are >3sd (1.42% of data)
number_outpatient_outliers = (df['number_outpatient'] > (0.512846+1.544866*3)).value_counts()
number_outpatient_outliers
```

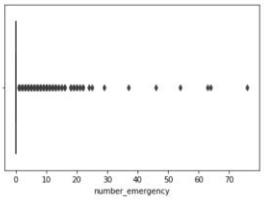


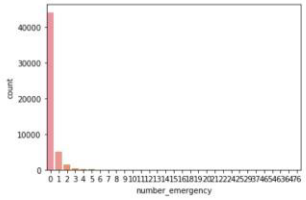



```
False    51029
True       737
Name: number_outpatient, dtype: int64
```

- number_emergency - 667 records are >3sd (1.29% of data) - (positive skewed. Majority is 0) (mean = 0.269173, sd = 1.146923, median = 0, min = 0, 1stQ = 0, 3rdQ = 0, max = 76)

```
# Number of emergency visits of the patient in the year preceding the encounter
no_outpatient_plot = sns.boxplot(df['number_emergency'])
plt.show()

no_outpatient_count = sns.countplot(data=df, x='number_emergency')
plt.show()

# positive skewed. Majority is 0 with outliers
# 667 records are >3sd (1.29% of data)
number_emergency_outliers = (df['number_emergency'] > (0.269173+1.146923*3)).value_counts()
number_emergency_outliers
```

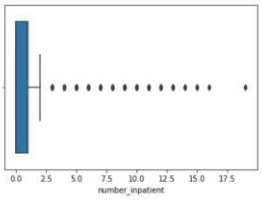


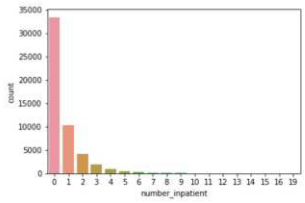



```
False    51099
True       667
Name: number_emergency, dtype: int64
```

- number_inpatient - 1146 records are >3sd (2.21% of data) - (positive skewed. Majority is 0) (mean = 0.686879, sd = 1.320968, median = 0, min = 0, 1stQ = 0, 3rdQ = 1, max = 19

```
# Number of inpatient visits of the patient in the year preceding the encounter
no_outpatient_plot = sns.boxplot(df['number_inpatient'])
plt.show()

no_outpatient_count = sns.countplot(data=df, x='number_inpatient')
plt.show()

# positive skewed. Majority is 0 with outliers
# 1146 records are >3sd (2.21% of data)
number_inpatient_outliers = (df['number_inpatient'] > (0.686879+1.320968*3)).value_counts()
number_inpatient_outliers
```

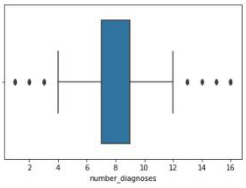


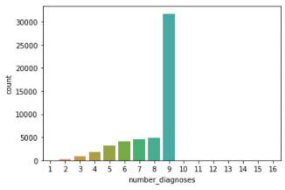



```
False    50620
True      1146
Name: number_inpatient, dtype: int64
```

- number_diagnoses - 62 records are >3sd and 304 records are <-3sd (0.71% of dat a) (median > mean ==> negative skewed) (mean = 7.928486, sd = 1.693707, median = 9, min = 1, 1stQ = 7, 3rdQ = , max = 16)

```
# Number of diagnoses entered to the system
no_diagnoses_plot = sns.boxplot(df['number_diagnoses'])
plt.show()

no_diagnoses_count = sns.countplot(data=df, x='number_diagnoses')
plt.show()

# mean = 7.928485878762122, median = 0, median > mean ==> negative skewed
# outliers on both sides

# 62 records are >3sd  and 304 records are <-3sd (0.71% of data)
number_diagnoses_outliers = (df['number_diagnoses'] > (7.928485878762122+1.693707*3)).value_counts()
number_diagnoses_outliers

number_diagnoses_outliers = (df['number_diagnoses'] < (7.928485878762122-1.693707*3)).value_counts()
number_diagnoses_outliers
```





2) Elaborate on the data preparation steps required (by correcting the data types of some variables, data cleaning and transformation) to address the data quality problems that yo u encountered during data exploration.

In general, the following can be done when errors are foun d:

- ignore
- treat errors as separate value
- impute with mean/median (numeric) or mode (categorical)
- other techniques - eg. regression, interpolation
- flagging

In addition, there are other considerations should be taken into account on whether variables should be dropped. They include:

- Unary variable - A variable that takes only one unique value. Since all the va lues are same, it will not change the outcome of any data mining model.
- Redundant variables - Same information represented in two or more different ways. All of them except one should be removed as no new information is provided by such variab les.
- Derived variable - A variable that is derived from another variab le. In most cases, it is not necessary to use the derived variable in the analysis.

Below the analysis on data preparations steps:

<u>Error in data (values with '?')</u>

- race - 1016 records - difficult to approximate race. 1016 reco rds is about 1% of data. Will keep them.

  Map '?' to np.NaN

<u>Weight</u>

- weight – although there are 50431 records with '?' representing about 97% of total data. Nevertheless, it is an important indicator in diabetes patien ts. Hence, will impute the missing values using mode.

Drop column

- encounter_id – nominal - not useful for analysis
- patient_nbr – nominal - not useful for analysis
- payer_code - 7601 records - (about 15% of total data) – not useful for the analysis
- medical_specialty - 32203 records - (about 62% of total dat a) – not useful for analysis
- max_glu_serum (most value is "none" i.e. test not taken)
- A1Cresult (most value is "none" i.e. not measured)

Drop row

- age with '?' - 10 records – few records
- chlorpropamide with '?' - 9 records – few records

Diabetes Medicine

As majority of the below diabetes medicines are not prescribed pl us it already has a variable on diabetesMed indicating if taking any diabetes medication. Hence, the f ollowing are drop due to redundancies and to reduce dimensionality.

- metformin (diabetes medicine)
- repaglinide (diabetes medicine)
- nateglinide (diabetes medicine)
- chlorpropamide (diabetes medicine)
- glimepiride (diabetes medicine)
- acetohexamide (diabetes medicine)
- glipizide (diabetes medicine)
- glyburide (diabetes medicine)
- tolbutamide (diabetes medicine)

diag_1, diag_2,and diag_3

```
# check to see how many unique values in diag_1
diag1_unique = len(pd.unique(df_process['diag_1']))
diag1_unique
```

649

```
# check to see how many unique values in diag_2
diag2_unique = len(pd.unique(df_process['diag_2']))
diag2_unique
```

683

```
# check to see how many unique values in diag_3
diag3_unique = len(pd.unique(df_process['diag_3']))
diag3_unique
```

726

- diag_1 – there are 649 different diagnoses in diag_1. Need to red uce dimension in the analysis. As a start, will create derived variables on the top 2 diagnosis. '428' diagnosi s (Congestive heart failure, unspecified) and '414' diagnosis (Other forms of chronic ischemic heart disease) to see if these add predictive power of the model. Other diagnoses are drop.
- diag_2 - there are 683 different diagnoses in diag_2. Need to reduce dimension i n the analysis. As a start, will create derived variables on the top 2 diagnosis. '276' diagnosis (Hypero smolality and/or hypernatremia) and '428' diagnosis (Congestive heart failure, unspecified) to see if these add p redictive power of the model. Other diagnoses are drop.
- diag_3 - there are 726 different diagnoses in diag_2. Need to re duce dimension in the analysis. As a start, will create derived variables on the top 2 diagnosis. '250' diagnosis (Diab etes) and '401' diagnosis (Malignant essential hypertension) to see if these add predictive power of the mode l. Other diagnoses are drop.

3) Demonstrate the data preparation by including a screenshot(s) of the Python code and its outputs that show the steps on how you had corrected all the identified data quality problems in this dataset.

```
# create derived variables for diag_1, diag_2, diag_3 for their top two diagnoses

# For diag_1
# create new column for diag_1 with '428' diagnosis (Congestive heart failure, unspecified)
df_process['diag1_428'] = df_process['diag_1'].apply(lambda x: 1 if x == '428' else 0)

# create new column for diag_1 with '414' diagnosis(Other forms of chronic ischemic heart disease)
df_process['diag1_414'] = df_process['diag_1'].apply(lambda x: 1 if x == '414' else 0)



# For diag_2
# create new column for diag_2 with '276' diagnosis(Hyperosmolality and/or hypernatremia)
df_process['diag2_276'] = df_process['diag_2'].apply(lambda x: 1 if x == '276' else 0)

# create new column for diag_2 with '428' diagnosis(Congestive heart failure, unspecified)
df_process['diag2_428'] = df_process['diag_2'].apply(lambda x: 1 if x == '428' else 0)



# For diag_3
# create new column for diag_3 with '250' diagnosis(Diabetes)
df_process['diag3_250'] = df_process['diag_3'].apply(lambda x: 1 if x == '250' else 0)

# create new column for diag_3 with '401' diagnosis(Malignant essential hypertension)
df_process['diag3_401'] = df_process['diag_3'].apply(lambda x: 1 if x == '401' else 0)

```

```
### Pre-process data


# Drop columns : encounter_id, patient_nbr, payer_code, medical_specialty
df_process = df_process.drop(['encounter_id', 'patient_nbr', 'payer_code',
                              'medical_specialty'], axis=1)

# Drop rows : age with ? (10 records)
df_process = df_process[df_process.age != '?']

# Drop rows : chlorpropamide (9 records)
df_process = df_process[df_process.chlorpropamide != '?']

# Drop rows : gender (4 records)  (NaN)
df_process = df_process.dropna(how='any', subset=['gender'])

# Race - replace "?" to nan
df_process['race'] = df_process['race'].replace(['?'],np.NaN)

# Impute missing weight values using mode
df_process['weight'] = df_process['weight'].replace(['?'],np.NaN)
stats.mode(df_process['weight'])
# replace NaN weight with mode '[75-100)'
df_process['weight'] = df_process['weight'].replace(np.NaN, '[75-100)')

# Dropping other variables in df_process
df_process = df_process.drop(['diag_1', 'diag_2', 'diag_3',
                              'max_glu_serum', 'A1Cresult',
                              'metformin', 'repaglinide', 'nateglinide',
                              'chlorpropamide', 'glimepiride', 'acetohexamide', 'glipizide',
                              'glyburide', 'tolbutamide'], axis=1)

```

	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	num_procedures	...	insulin	change	diabetesMed	readmitted	diag1_428	diag1_414	diag2_276	diag2_428	diag3_250	diag3_401
0	Caucasian	0.0	[60-70)	[75-100)	2	1	1	4	43	0	...	Steady	0	1	>30	1	0	0	0	0	0
1	Caucasian	0.0	[80-90)	[75-100)	3	2	1	8	48	6	...	Down	1	1	>30	0	1	0	0	0	0
2	Caucasian	0.0	[80-90)	[75-100)	2	1	1	2	39	0	...	No	0	0	NO	0	0	0	0	0	1
3	Caucasian	0.0	[60-70)	[75-100)	3	1	1	2	54	0	...	No	0	0	NO	0	0	0	0	0	0
4	Caucasian	0.0	[60-70)	[75-100)	3	2	1	7	70	1	...	Down	1	1	>30	0	0	0	0	0	0
5	Caucasian	1.0	[70-80)	[75-100)	2	1	7	3	1	4	...	Steady	0	1	NO	0	0	0	0	0	0
6	Caucasian	1.0	[60-70)	[75-100)	2	1	1	6	28	1	...	Up	1	1	NO	0	0	0	0	0	1
7	AfricanAmerican	0.0	[40-50)	[75-100)	1	1	7	3	48	0	...	Down	1	1	NO	0	0	0	0	1	0
8	AfricanAmerican	1.0	[70-80)	[75-100)	1	1	7	5	42	2	...	No	0	1	<30	0	0	0	0	0	0
9	AfricanAmerican	0.0	[70-80)	[75-100)	1	1	7	1	43	1	...	Up	1	1	>30	0	0	0	0	0	0
10	Caucasian	0.0	[70-80)	[75-100)	3	1	1	3	27	6	...	Steady	1	1	NO	0	1	0	0	0	0

Question 5
Selection of data mining task and feature selection:

1) Identify the most suitable data mining task (i.e. classifica tion, clustering or association mining) that can be performed on this dataset. Justify your choice.

Assuming we would like to predict patients who will be readmitted to the hospital in less than 30 days. The most suitable technique for this data mining task is classification, as it is based on historical labelled data and can model to make prediction on the target variable (ie. readmitted<30 days). As target class is already known hence training data is already labelled with answers.

Classification data mining requires sufficient size of records to be able to train the model, in this case, the criteria can be met with about 51700 records. The dataset required to be divided into training, validation and testing sub-sets in order to create the classification model.

2) What variables will you include in this data mining task and why? Describe here if you will create any derived variables. Identify the roles (i.e., input or target) of each variable.

With the assumption that we would like to predict patients who will be readmitted to the hospital within 30 days.

Target : readmission

Input :

1. gender
2. admission_type_id
3. discharge_disposition_id
4. admission_source_id
5. time_in_hospital
6. num_lab_procedures
7. num_medications
8. number_outpatient
9. number_diagnoses
10. change
11. diabetesMed
12. diag1_428
13. diag1_414
14. diag2_276
15. diag2_428
16. diag3_250
17. diag3_401
18. race
19. age
20. weight
21. insulin

Will use the above variables to build an initial model, then by process of elimination using the features selection techniques (eg. Univariate filter method, multivariate filter method, wrapper methods etc..) to select features to reduce overfitting.

Normally, derived variables are not necessary as it is highly correlated with the 'parent' variables. In this case, we will use the derived variables of the top two diagnoses in diag_1, diag_2 and diag_3 for the model to see if it helps with prediction. At the same time, due to dimensionality reduction, diag_1, diag_2 and diag_3 variables are dropped.