Statistics for Data Science (UE19CS203)

Project Report Format

- **Title :-** Dataset for Diabetes patients in US Hospitals (Impact of HbA1c Measurement on Hospital Re-admission Rates: Analysis of 30,000 Clinical Database Patient Records).
- **Team Details :-**

Name	SRN	Section
Harshika Agrawal	PES2UG19CS145	C
Hrithik HM	PES2UG19CS151	C
Jayanth Kamath	PES2UG19CS161	C
K Pavan Kumar	PES2UG19CS171	C

- **Abstract :-** This data has been prepared to analyse factors related to readmission as well as other outcomes pertaining to patients with diabetes. So we are going to apply methods in order to get meaningful insights from it. First step is to clean the data set which is nothing but data cleaning. After that by plotting the graphs we will come to know how many patients suffering from diabetes and the re-admission rates in hospital. Values from primary diagnoses will help to know how many encounters suffering from disease of circulatory system, disease of respiratory system, disease of digestive system, diabetes mellitus etc.

- **Introduction:-** It is increasingly recognized that the management of hyperglycemia in the hospitalized patient has a significant bearing on outcome, in terms of both morbidity and mortality. This recognition has led to the development of formalized protocols in the intensive care unit (ICU) setting with rigorous glucose targets in many institutions. However, the same cannot be said for most non-ICU inpatient admissions. Rather, anecdotal evidence suggests that inpatient management is arbitrary and often leads to either no treatment at all or wide fluctuations in glucose when traditional management strategies are employed. Although data are few, recent controlled trials have demonstrated that protocoldriven inpatient strategies can be both effective and safe. As such, implementation of protocols in the hospital setting is now recommended [6, 7]. However, there are few national assessments of diabetes care in the hospitalized patient which could serve as a baseline for change.
- The present analysis of large clinical database was undertaken to examine historical patterns of diabetes care in patients with diabetes admitted to a US hospital and to inform future directions which might lead to improvements in patient safety. In particular, we examined the use of HbA1c as a marker of attention to diabetes care in a large number of individuals identified as having a diagnosis of diabetes mellitus. We hypothesize that measurement of HbA1c is associated with a reduction in readmission rates in individuals admitted to the hospital.

Dataset :-

- The dataset represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 15 features representing patient and hospital

outcomes. Information was extracted from the database for encounters that satisfied the following criteria.

- Source - https://archive.ics.uci.edu/ml/datasets/diabetes+130us+hospitals+for+years+1999-2008#
- (1) It is an inpatient encounter (a hospital admission).
  (2) It is a diabetic encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis.
  (3) The length of stay was at least 1 day and at most 14 days.
  (4) Laboratory tests were performed during the encounter.
  (5) Medications were administered during the encounter.
  The data contains such attributes as patient number, race, gender, age, admission type, time in hospital, medical specialty of admitting physician, number of lab test performed, HbA1c test result, diagnosis, number of medication, diabetic medications, number of outpatient, inpatient, and emergency visits in the year before the hospitalization, etc.
- Our data set consists of 30000 records of patients or encounters and it has 15 attributes or Columns. The attributes are encounter id, gender, age, time in hospital, medical specialty, number of lab procedures, number of medications, primary diagnosis, number of diagnosis, insulin, diabetes medication, weight, admission type id, readmitted, and change.

- **Preprocessing or Data Cleaning :-**

Importance of data cleaning in our dataset:

The original database contains incomplete, redundant, and noisy information as expected in any real-world data. There were several features that could not be treated directly since they had a high percentage of missing values. The variables chosen to control for patient demographic and illness severity were gender, age, race, admission source, discharge disposition, primary diagnosis ,medical specialty of the admitting physician, and time spent in hospital. Having clean data will ultimately increase overall productivity and allow for the highest quality information in our decision-making. Benefits include removal of errors when multiple sources of data are at play. T helps us in monitoring errors and better reporting to see where errors are coming from, making it easier to fix incorrect or corrupt data for future applications. Using tools for data cleaning will make for quicker decision-making and collecting meaningful insights.

Techniques required for data cleaning:

1. Removing duplicates.
2. Removing outliers.
3. Medical specialty attribute was maintained, adding the value “missing” in order to account for missing values.
4. Checking for inconsistent data and replacing it with the mean or median of that particular feature.

Part of Original Dataset before data cleaning:

encounter_id	gender	age	time_in_hospital	medical_specialty	num_lab_procedures	num_medications	diag_1	number_diagnoses	insulin	diabetesMed	Weight	admission_type_id	readmitted	change
2278392	Female	[0-10)	1	Pediatrics-Endocrir	41	1	250.83	?	No	No	241.8936	6	NO	No
149190	Female	[10-20)	3	?	59	18	276	9	Up	Yes	162.3105	1	>30	Ch
64410	Female	[20-30)	2	?	11	13	648	6	No	Yes	212.7409	1	NO	No
500364	Male	[30-40)	2	?	44	16	8	7	Up	Yes	220.0425	1	NO	Ch
16680	Male	[40-50)	1	?	51	8	197	5	Steady	Yes	206.3498	1	NO	Ch
35754	Male	[50-60)	3	?	31	16	414	9	Steady	Yes	152.2122	2	>30	No
55842	Male	[60-70)	4	?	70	21	414	7	Steady	Yes	183.9279	3	NO	Ch
63768	Male	[70-80)	5	?	73	12	428	8	No	Yes	167.9711	1	>30	No
12522	Female	[80-90)	?	?	68	28	398	8	Steady	Yes	175.9294	2	NO	Ch
15738	Female	[90-100)	?	InternalMedicine	33	18	434	8	Steady	Yes	156.3997	3	NO	Ch
28236	Female	[40-50)	9	?	47	17	250.7	9	Steady	Yes	186.6049	1	>30	No
36900	Male	[60-70)	7	?	62	11	157	7	Steady	Yes	213.7412	2	<30	Ch
40926	Female	[40-50)	7	Family/GeneralPra	60	15	428	8	Down	Yes	167.1275	1	<30	Ch
42570	Male	[80-90)	?	Family/GeneralPra	55	31	428	8	Steady	Yes	189.4462	1	NO	No
62256	Female	[60-70)	1	?	49	2	518	8	Steady	Yes	186.4342	3	>30	No
73578	Male	[60-70)	?	?	75	13	999	9	Up	Yes	172.1869	1	NO	Ch
77076	Male	[50-60)	4	?	45	17	410	8	Steady	Yes	196.0285	1	<30	Ch
84222	Female	[50-60)	3	Cardiology	29	11	682	3	No	Yes	172.8835	1	NO	No
89682	Male	[70-80)	5	?	35	23	402	9	Steady	Yes	185.984	1	>30	No
148530	Male	[70-80)	6	?	42	23	737	8	Steady	Yes	182.4266	3	NO	Ch
150006	Female	[50-60)	2	?	66	19	410	7	Down	Yes	174.1159	2	NO	Ch
150048	Male	[60-70)	2	?	36	11	572	6	Steady	Yes	197.7314	2	NO	Ch
182796	Female	[70-80)	2	?	47	12	410	8	No	No	149.1736	2	NO	No
183930	Female	[80-90)	?	?	42	19	V57	8	No	No	228.7618	2	>30	No
216156	Female	[70-80)	3	?	19	18	189	6	Steady	Yes	162.0067	3	NO	Ch
221634	Female	[50-60)	1	?	33	7	786	3	No	Yes	192.344	1	NO	No
236316	Male	[80-90)	6	Cardiology	64	18	427	7	No	Yes	184.4352	1	NO	Ch

Part of dataset after data cleaning is done:

	encounter_id	gender	age	time_in_hospital	medical_specialty	num_lab_procedures	num_medications	diag_1	number_diagnoses	insulin	diabetesMed	Weight	admission_type_id	readmitted	change
0	149190	Female	[10-20)	3	other	59	18	276	9	Up	Yes	162.3105	1	>30	Ch
1	64410	Female	[20-30)	2	other	19	13	648	6	No	Yes	203.5967	1	NO	No
2	500364	Male	[30-40)	2	other	44	16	8	7	Up	Yes	203.5967	1	NO	Ch
3	16680	Male	[40-50)	1	other	51	8	197	5	Steady	Yes	203.5967	1	NO	Ch
4	35754	Male	[50-60)	3	other	31	16	414	9	Steady	Yes	152.2122	2	>30	No
5	55842	Male	[60-70)	4	other	68	21	414	7	Steady	Yes	183.9279	3	NO	Ch
6	63768	Male	[70-80)	5	other	68	12	428	8	No	Yes	167.9711	1	>30	No
7	12522	Female	[80-90)	4	other	68	25	398	8	Steady	Yes	175.9294	2	NO	Ch
8	15738	Female	[90-100)	4	InternalMedicine	33	18	434	8	Steady	Yes	156.3997	3	NO	Ch
9	28236	Female	[40-50)	9	other	47	17	250.7	9	Steady	Yes	186.6049	1	>30	No
10	36900	Male	[60-70)	7	other	62	11	157	7	Steady	Yes	203.5967	2	<30	Ch
11	40926	Female	[40-50)	7	Family/GeneralPractice	60	15	428	8	Down	Yes	167.1275	1	<30	Ch
12	42570	Male	[80-90)	4	Family/GeneralPractice	55	25	428	8	Steady	Yes	189.4462	1	NO	No
13	62256	Female	[60-70)	1	other	49	7	518	8	Steady	Yes	186.4342	3	>30	No
14	73578	Male	[60-70)	4	other	68	13	999	9	Up	Yes	172.1869	1	NO	Ch
15	77076	Male	[50-60)	4	other	45	17	410	8	Steady	Yes	196.0285	1	<30	Ch
16	84222	Female	[50-60)	3	Cardiology	29	11	682	3	No	Yes	172.8835	1	NO	No
17	89682	Male	[70-80)	5	other	35	23	402	9	Steady	Yes	185.984	1	>30	No
18	148530	Male	[70-80)	6	other	42	23	737	8	Steady	Yes	182.4266	3	NO	Ch
19	150006	Female	[50-60)	2	other	66	19	410	7	Down	Yes	174.1159	2	NO	Ch
20	150048	Male	[60-70)	2	other	36	11	572	6	Steady	Yes	197.7314	2	NO	Ch
21	182796	Female	[70-80)	2	other	47	12	410	8	No	No	149.1736	2	NO	No
22	183930	Female	[80-90)	4	other	42	19	V57	8	No	No	203.5967	2	>30	No
23	216156	Female	[70-80)	3	other	19	18	189	6	Steady	Yes	162.0067	3	NO	Ch
24	221634	Female	[50-60)	1	other	33	7	786	3	No	Yes	192.344	1	NO	No
25	236316	Male	[80-90)	6	Cardiology	64	18	427	7	No	Yes	184.4352	1	NO	Ch
26	248916	Female	[50-60)	2	Surgery-General	25	11	996	3	Steady	Yes	203.5967	1	>30	No

Code and Output Screenshots of Data Cleaning

In [36]:
```
# importing pandas library and numpy
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/diabetic_dataset.csv')
missing_values = ["?","n/a","na","--","N"]
df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/diabetic_dataset.csv', na_values = missing_value
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
df
```

Out[36]:

	encounter_id	gender	age	time_in_hospital	medical_specialty	num_lab_procedures	num_medications	diag_1	number_diagnoses	insulin	diabete
0	2278392	Female	[0-10)	1.0	Pediatrics-Endocrinology	41	1	250.83	NaN	No	
1	149190	Female	[10-20)	3.0	NaN	59	18	276	9.0	Up	
2	64410	Female	[20-30)	2.0	NaN	11	13	648	6.0	No	
3	500364	Male	[30-40)	2.0	NaN	44	16	8	7.0	Up	
4	16680	Male	[40-50)	1.0	NaN	51	8	197	5.0	Steady	
...	...	...	...	...	...	...	...	...	...	...	
29994	97461414	Male	[50-60)	6.0	Surgery-Cardiovascular/Thoracic	51	55	410	9.0	No	
29995	97461552	Male	[70-80)	6.0	Family/GeneralPractice	52	7	434	5.0	Steady	
29996	97463718	Female	[80-90)	6.0	InternalMedicine	73	22	410	9.0	Down	
29997	97464618	Male	[60-70)	NaN	InternalMedicine	38	13	434	6.0	No	
29998	97466244	Female	[50-60)	1.0	InternalMedicine	61	12	250.8	5.0	No	

In [37]:
```
print(df.isnull().sum())
```

```
encounter_id              0
gender                    0
age                       0
time_in_hospital       2886
medical_specialty     11474
num_lab_procedures        0
num_medications           0
diag_1                    8
number_diagnoses        142
insulin                   0
diabetesMed               0
Weight                    0
admission_type_id         0
readmitted                0
change                    0
dtype: int64
```

In [39]:
```
# checking for any inconsistent data in gender column
df.gender = df.gender.astype(str)
list_gender = ["Male","male","Female","female"]
count_01 = 0
for i in df['gender']:
    if(i in list_gender):
        df.loc[count_01,'gender'] = i
    else:
        df.loc[count_01,'gender'] = "unkown"
    count_01+=1
print(df.isnull().sum())
```

```
encounter_id              0
gender                    0
age                       0
time_in_hospital       2886
medical_specialty     11474
num_lab_procedures        0
num_medications           0
diag_1                    8
number_diagnoses        142
insulin                   0
diabetesMed               0
Weight                    0
admission_type_id         0
readmitted                0
change                    0
dtype: int64
```

In [41]:
```
# cleaning the time_in_hospital column (replacing missing values in this column by median of that feature)
median = df['time_in_hospital'].median()
df['time_in_hospital'].fillna(median,inplace=True)

# cleaning the medical_specialty column (adding "missing")
df['medical_specialty'].fillna("other",inplace=True)
print(df.isnull().sum())
```

```
encounter_id            0
gender                  0
age                     0
time_in_hospital        0
medical_specialty       0
num_lab_procedures      0
num_medications         0
diag_1                  8
number_diagnoses      142
insulin                 0
diabetesMed             0
Weight                  0
admission_type_id       0
readmitted              0
change                  0
dtype: int64
```

In [42]:
```
# checking for inconsistent data in number of lab procedures column
count_02 = 0
for j in df['num_lab_procedures']:
    if(j<0):
        df.loc[count_02,'num_lab_procedures'] = np.NaN
    count_02+=1
count_03 = 0
for k in df['num_medications']:
    if(k<0):
        df.loc[count_03,'num_medications'] = np.NaN
    count_03+=1
print(df.isnull().sum())
```

```
encounter_id            0
gender                  0
age                     0
time_in_hospital        0
medical_specialty       0
num_lab_procedures      0
num_medications         0
diag_1                  8
number_diagnoses      142
insulin                 0
diabetesMed             0
Weight                  0
admission_type_id       0
readmitted              0
change                  0
dtype: int64
```

In [43]:
```
# checking for inconsistent data in insulin column
list_insulin = ["No","Up","Down","Steady"]
count_06 = 0
for l in df['insulin']:
    if (l in list_insulin):
        df.loc[count_06,'insulin'] = l
    else:
        df.loc[count_06,'insulin'] = np.NaN
    count_06+=1
print(df.isnull().sum())
```

```
encounter_id            0
gender                  0
age                     0
time_in_hospital        0
medical_specialty       0
num_lab_procedures      0
num_medications         0
diag_1                  8
number_diagnoses      142
insulin                 0
diabetesMed             0
Weight                  0
admission_type_id       0
readmitted              0
change                  0
dtype: int64
```

In [45]:
```
# checking for inconsistent data in diabetes medication column
df['diabetesMed'] = df.diabetesMed.astype(str)
list_diabetesMed = ["No","Yes"]
count_07 = 0
for i in df['diabetesMed']:
    if (i in list_diabetesMed):
        df.loc[count_07,'diabetesMed'] = i
    else:
        df.loc[count_07,'diabetesMed'] = np.NaN
    count_07+=1

# checking for inconsistent data in weight column
count_08 = 0
for i in df['Weight']:
    if(i<0.0):
        df.loc[count_08,'Weight'] = np.NaN
print(df.isnull().sum())
```

```
encounter_id            0
gender                  0
age                     0
time_in_hospital        0
medical_specialty       0
num_lab_procedures      0
num_medications         0
diag_1                  8
number_diagnoses      142
insulin                 0
diabetesMed             0
Weight                  0
admission_type_id       0
readmitted              0
change                  0
dtype: int64
```

In [46]:
```
# checking for inconsistent data in admission type id column
count_09 = 0
for i in df['admission_type_id']:
    if((i>=1) and (i<=8)):
        df.loc[count_09,'admission_type_id'] = i
    else:
        df.loc[count_09,'admission_type_id'] = np.NaN
    count_09+=1

# checking for inconsistent data in readmitted column
df['readmitted'] = df.readmitted.astype(str)
list_readmitted = ["NO",">30","<30"]
count_10 = 0
for i in df['readmitted']:
    if (i in list_readmitted):
        df.loc[count_10,'readmitted'] = i
    else:
        df.loc[count_10,'readmitted'] = np.NaN
    count_10+=1
print(df.isnull().sum())
```

```
encounter_id            0
gender                  0
age                     0
time_in_hospital        0
medical_specialty       0
num_lab_procedures      0
num_medications         0
diag_1                  8
number_diagnoses      142
insulin                 0
diabetesMed             0
Weight                  0
admission_type_id       0
readmitted              0
change                  0
dtype: int64
```

In [47]:
```
# checking for inconsistent data in change column
df['change'] = df.change.astype(str)
list_change = ["No","Ch"]
count_11 = 0
for i in df['change']:
    if (i in list_change):
        df.loc[count_11,'change'] = i
    else:
        df.loc[count_11,'change'] = np.NaN
    count_11+=1

# dropping rows which has nan values in diag_1 column
df.dropna(subset=['diag_1'],inplace=True)

# dropping rows which has nan values in number_diagnoses column
df.dropna(subset=['number_diagnoses'],inplace=True)

print(df.isnull().sum())  # all columns has 0 nan or missing values
```

```
encounter_id          0
gender                0
age                   0
time_in_hospital      0
medical_specialty     0
num_lab_procedures    0
num_medications       0
diag_1                0
number_diagnoses      0
insulin               0
diabetesMed           0
Weight                0
admission_type_id     0
readmitted            0
change                0
dtype: int64
```

In [49]:
```
df # this is the final data frame without missing values
```

Out[49]:

	encounter_id	gender	age	time_in_hospital	medical_specialty	num_lab_procedures	num_medications	diag_1	number_diagnoses	insulin	diabete
1	149190	Female	[10-20)	3.0	other	59	18	276	9.0	Up	
2	64410	Female	[20-30)	2.0	other	11	13	648	6.0	No	
3	500364	Male	[30-40)	2.0	other	44	16	8	7.0	Up	
4	16680	Male	[40-50)	1.0	other	51	8	197	5.0	Steady	
5	35754	Male	[50-60)	3.0	other	31	16	414	9.0	Steady	
...	...	...	...	...	...	...	...	...	...	...	
29994	97461414	Male	[50-60)	6.0	Surgery-Cardiovascular/Thoracic	51	55	410	9.0	No	
29995	97461552	Male	[70-80)	6.0	Family/GeneralPractice	52	7	434	5.0	Steady	
29996	97463718	Female	[80-90)	6.0	InternalMedicine	73	22	410	9.0	Down	
29997	97464618	Male	[60-70)	4.0	InternalMedicine	38	13	434	6.0	No	
29998	97466244	Female	[50-60)	1.0	InternalMedicine	61	12	250.8	5.0	No	

29849 rows × 15 columns

2) Code :-

```
# importing pandas library and numpy
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/diabetic_dataset.csv'
)
missing_values = ["?","n/a","na","--","N"]
df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/diabetic_dat
aset.csv', na_values = missing_values)
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
df

# checking for any inconsistent data in gender column
df.gender = df.gender.astype(str)
list_gender = ["Male","male","Female","female"]
count_01 = 0
for i in df['gender']:
    if(i in list_gender):
        df.loc[count_01,'gender'] = i
    else:
        df.loc[count_01,'gender'] = "unkown"
    count_01+=1

# cleaning the time_in_hospital column (replacing missing values in this column by median of that
feature)
```

```
median = df['time_in_hospital'].median()
df['time_in_hospital'].fillna(median,inplace=True)

# cleaning the medical_specialty column (adding "missing")
df['medical_specialty'].fillna("other",inplace=True)

# checking for inconsistent data in number of lab procedures column
count_02 = 0
for j in df['num_lab_procedures']:
    if(j<0):
        df.loc[count_02,'num_lab_procedures'] = np.NaN
    count_02+=1
count_03 = 0
for k in df['num_medications']:
    if(k<0):
        df.loc[count_03,'num_medications'] = np.NaN
    count_03+=1

# checking for inconsistent data in insulin column
list_insulin = ["No","Up","Down","Steady"]
count_06 = 0
for l in df['insulin']:
    if (l in list_insulin):
        df.loc[count_06,'insulin'] = l
    else:
        df.loc[count_06,'insulin'] = np.NaN
    count_06+=1

# checking for inconsistent data in diabetes medication column
df['diabetesMed'] = df.diabetesMed.astype(str)
list_diabetesMed = ["No","Yes"]
count_07 = 0
for i in df['diabetesMed']:
    if (i in list_diabetesMed):
        df.loc[count_07,'diabetesMed'] = i
    else:
        df.loc[count_07,'diabetesMed'] = np.NaN
    count_07+=1

# checking for inconsistent data in weight column
count_08 = 0
for i in df['Weight']:
```

```
    if(i<0.0):
        df.loc[count_08,'Weight'] = np.NaN

# checking for inconsistent data in admission type id column
count_09 = 0
for i in df['admission_type_id']:
    if((i>=1) and (i<=8)):
        df.loc[count_09,'admission_type_id'] = i
    else:
        df.loc[count_09,'admission_type_id'] = np.NaN
    count_09+=1

# checking for inconsistent data in readmitted column
df['readmitted'] = df.readmitted.astype(str)
list_readmitted = ["NO",">30","<30"]
count_10 = 0
for i in df['readmitted']:
    if (i in list_readmitted):
        df.loc[count_10,'readmitted'] = i
    else:
        df.loc[count_10,'readmitted'] = np.NaN
    count_10+=1

# checking for inconsistent data in change column
df['change'] = df.change.astype(str)
list_change = ["No","Ch"]
count_11 = 0
for i in df['change']:
    if (i in list_change):
        df.loc[count_11,'change'] = i
    else:
        df.loc[count_11,'change'] = np.NaN
    count_11+=1

# dropping rows which has nan values in diag_1 column
df.dropna(subset=['diag_1'],inplace=True)

# dropping rows which has nan values in number_diagnoses column
df.dropna(subset=['number_diagnoses'],inplace=True)

print(df.isnull().sum())  # all columns has 0 nan or missing values
print(df)
```

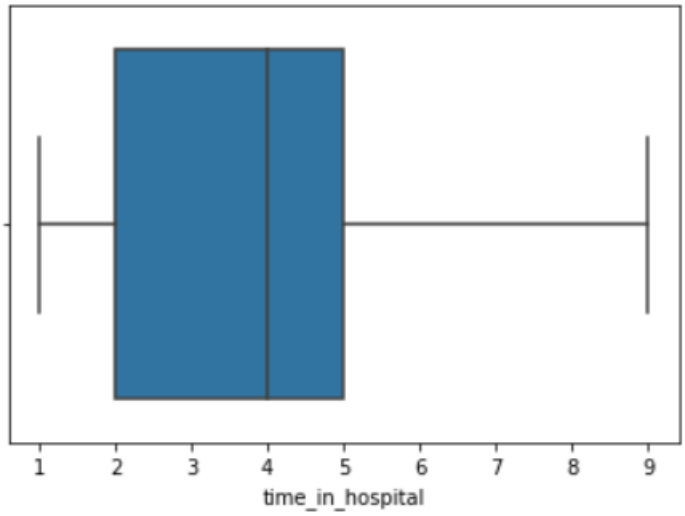
```
# copying the cleaned dataset (df) into a csv file
df.to_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(proj).csv')

# END OF DATA CLEANING
```

Exploratory data analysis:

Graph visualization

1) checking for outliers in time_in_hospital column

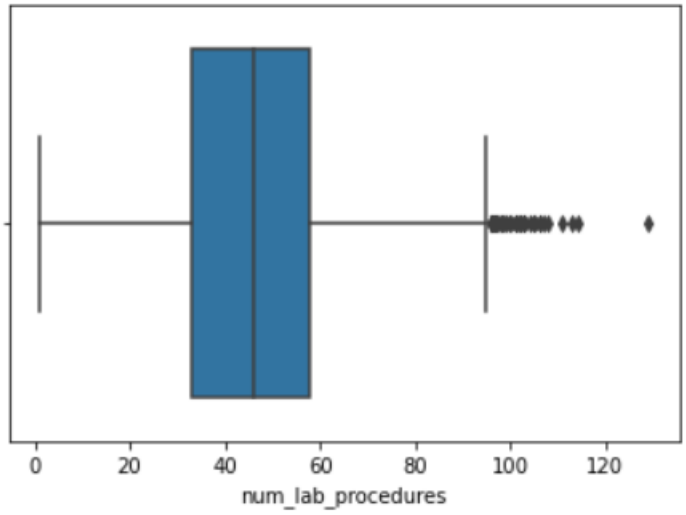
```
In [3]: sns.boxplot(x=df['time_in_hospital'])
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1a4249d4310>
```



Time in hospital column does not have any outliers

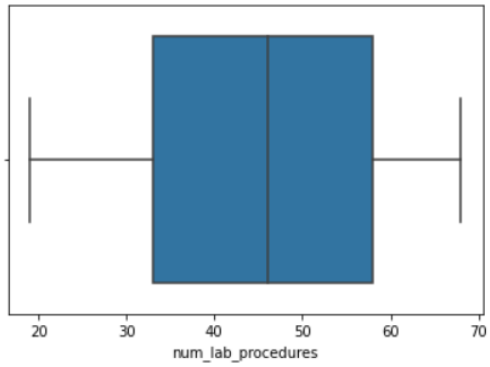
2) checking for outliers in num_lab_procedures column

```
In [4]: sns.boxplot(x=df['num_lab_procedures'])
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x1a424a2b0d0>
```



It has outliers, after filtering those outliers we get

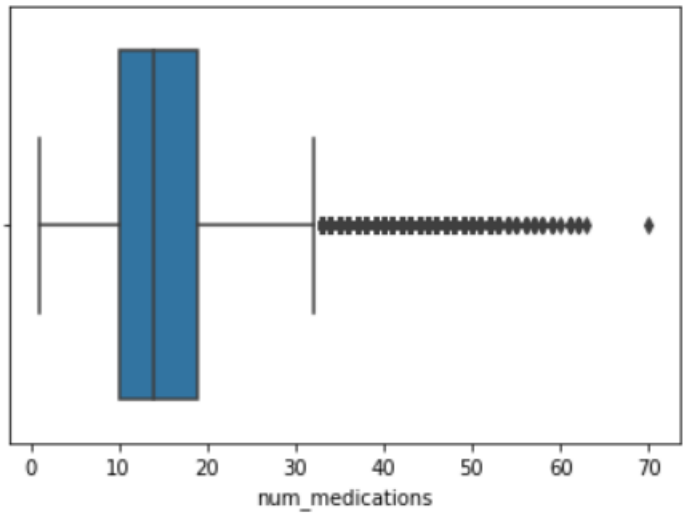
```
In [20]: print(df['num_lab_procedures'].skew())
         df['num_lab_procedures'].describe()
         print(df['num_lab_procedures'].quantile(0.10))
         print(df['num_lab_procedures'].quantile(0.90))
         df["num_lab_procedures"] = np.where(df["num_lab_procedures"] <19.0, 19.0,df['num_lab_procedures'])
         df["num_lab_procedures"] = np.where(df["num_lab_procedures"] >68.0, 68.0,df['num_lab_procedures'])
         df['num_lab_procedures'].skew()
         df['num_lab_procedures'].describe()
         sns.boxplot(x=df['num_lab_procedures'])

         -0.15323157425718087
         19.0
         68.0

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1a424c30460>
```



3) checking for outliers in num_medications column

```
In [7]: sns.boxplot(x=df['num_medications'])
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1a4249d4790>
```

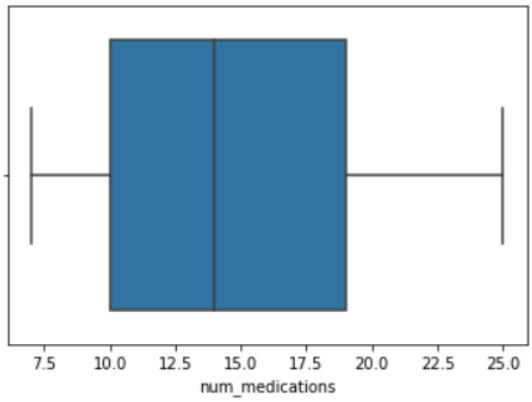


It has outliers,after filtering we get

```
In [21]: print(df['num_medications'].skew())
         df['num_medications'].describe()
         print(df['num_medications'].quantile(0.10))
         print(df['num_medications'].quantile(0.90))
         df["num_medications"] = np.where(df["num_medications"] <7.0, 7.0,df['num_medications'])
         df["num_medications"] = np.where(df["num_medications"] >25.0, 25.0,df['num_medications'])
         df['num_medications'].skew()
         df['num_medications'].describe()
         sns.boxplot(x=df['num_medications'])
```

```
0.39011094054103934
7.0
25.0
```

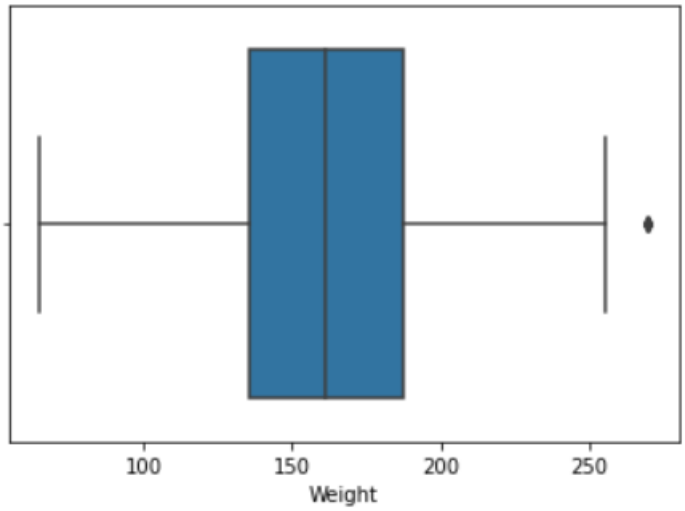
```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1a424c87a00>
```



4) checking for outliers in Weight column

```
In [10]: sns.boxplot(x=df['Weight'])
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1a423dcb790>
```

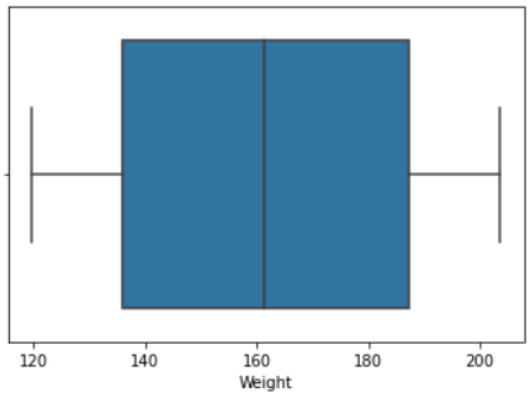


It has one outlier,after filtering we get

```
In [22]: print(df['Weight'].skew())
         df['Weight'].describe()
         print(df['Weight'].quantile(0.10))
         print(df['Weight'].quantile(0.90))
         df["Weight"] = np.where(df["Weight"] <119.63120643999999, 119.63120643999999,df['Weight'])
         df["Weight"] = np.where(df["Weight"] >203.5966954, 203.5966954,df['Weight'])
         df['Weight'].skew()
         df['Weight'].describe()
         sns.boxplot(x=df['Weight'])
```

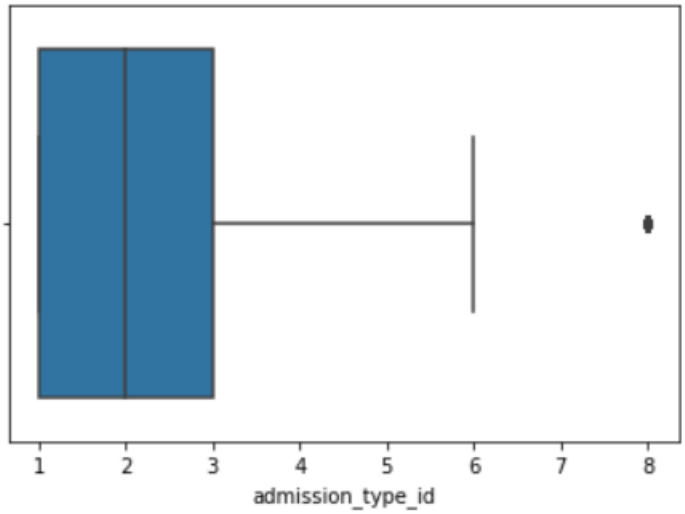
```
0.014448124487307817
119.63186416799999
203.5966954
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1a424ce6ca0>
```



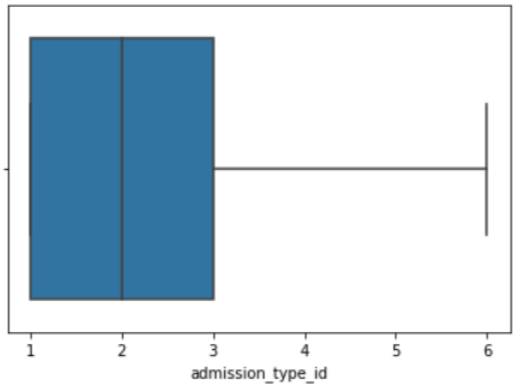
5) checking for outlier in admission_type_id column

```
In [13]: sns.boxplot(x=df['admission_type_id'])
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1a424b563d0>
```



It has one outlier,after filtering we get

```
In [23]: print(df['admission_type_id'].skew())
         df['admission_type_id'].describe()
         print(df['admission_type_id'].quantile(0.10))
         print(df['admission_type_id'].quantile(0.90))
         df['admission_type_id'] = np.where(df['admission_type_id'] <1.0, 1.0,df['admission_type_id'])
         df['admission_type_id'] = np.where(df['admission_type_id'] >6.0, 6.0,df['admission_type_id'])
         df['admission_type_id'].skew()
         df['admission_type_id'].describe()
         sns.boxplot(x=df['admission_type_id'])

         1.1257897066512947
         1.0
         6.0

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1a424d3bca0>
```



After filtering all outlers, copy the dataframe to a new csv file

```
df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(new).csv')
```

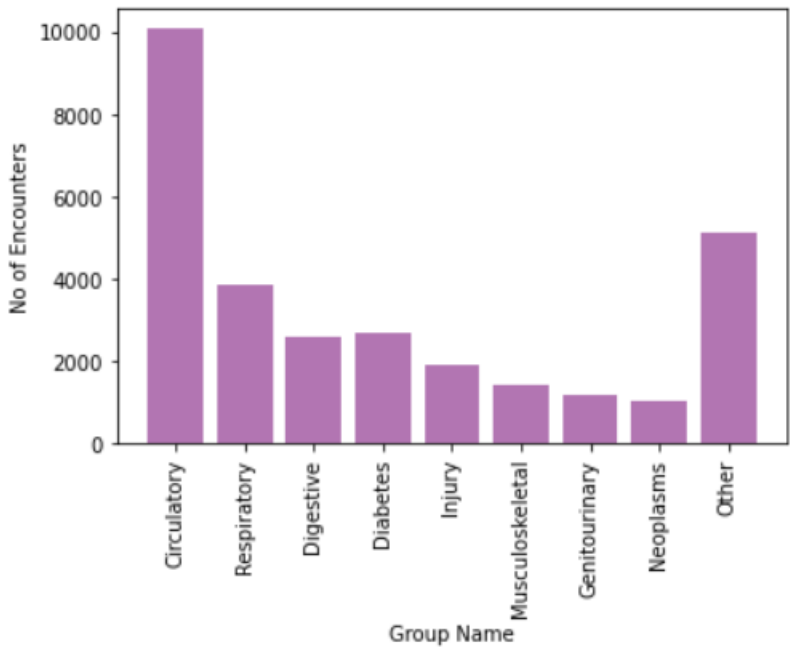
After copying it to csv file,plotting the graph from the modified dataset(without having outliers)

* Code for checking for all outliers :-

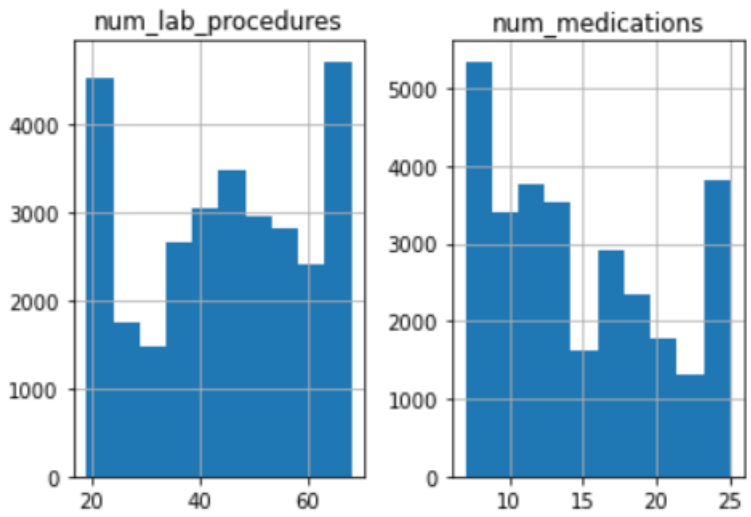
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(proj).csv')

sns.boxplot(x=df['time_in_hospital'])

sns.boxplot(x=df['num_lab_procedures'])
print(df['num_lab_procedures'].skew())
df['num_lab_procedures'].describe()
print(df['num_lab_procedures'].quantile(0.10))
print(df['num_lab_procedures'].quantile(0.90))
df["num_lab_procedures"] = np.where(df["num_lab_procedures"] <19.0, 19.0,df['num_lab_procedures'])
df["num_lab_procedures"] = np.where(df["num_lab_procedures"] >68.0, 68.0,df['num_lab_procedures'])
print(df['num_lab_procedures'].skew())
print(df['num_lab_procedures'].describe())

sns.boxplot(x=df['num_medications'])
print(df['num_medications'].skew())
df['num_medications'].describe()
print(df['num_medications'].quantile(0.10))
print(df['num_medications'].quantile(0.90))
df["num_medications"] = np.where(df["num_medications"] <7.0, 7.0,df['num_medications'])
df["num_medications"] = np.where(df["num_medications"] >25.0, 25.0,df['num_medications'])
print(df['num_medications'].skew())
print(df['num_medications'].describe())

sns.boxplot(x=df['Weight'])
print(df['Weight'].skew())
df['Weight'].describe()
print(df['Weight'].quantile(0.10))
print(df['Weight'].quantile(0.90))
df["Weight"] = np.where(df["Weight"] <119.63120643999999, 119.63120643999999,df['Weight'])
```

```
df["Weight"] = np.where(df["Weight"] >203.5966954, 203.5966954,df['Weight'])
print(df['Weight'].skew())
print(df['Weight'].describe())

sns.boxplot(x=df['admission_type_id'])
print(df['admission_type_id'].skew())
df['admission_type_id'].describe()
print(df['admission_type_id'].quantile(0.10))
print(df['admission_type_id'].quantile(0.90))
df['admission_type_id'] = np.where(df['admission_type_id'] <1.0, 1.0,df['admission_type_id'])
df['admission_type_id'] = np.where(df['admission_type_id'] >6.0, 6.0,df['admission_type_id'])
print(df['admission_type_id'].skew())
print(df['admission_type_id'].describe())

sns.boxplot(x=df['num_lab_procedures'])
sns.boxplot(x=df['num_medications'])
sns.boxplot(x=df['Weight'])
sns.boxplot(x=df['admission_type_id'])

df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(new).csv')
```

1) Bar chart :-

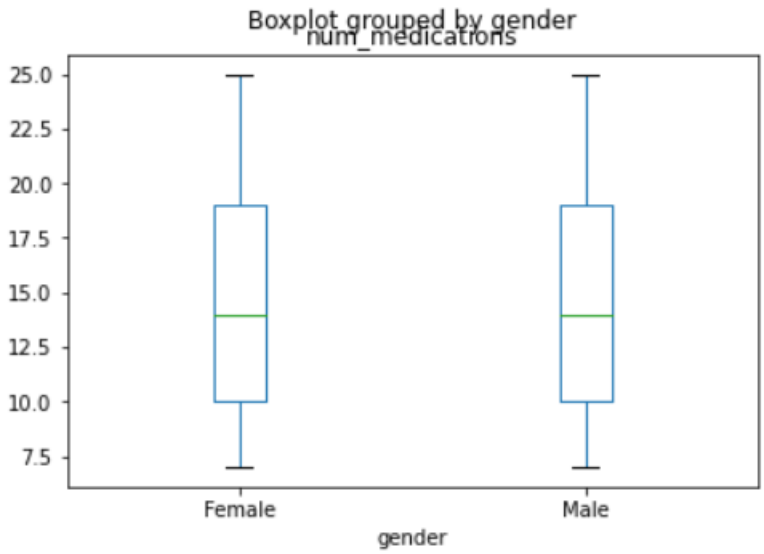


2) Histogram :-



3) Box Plot :-

```
In [3]: df.boxplot(by ='gender', column =['num_medications'], grid = False)
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x18fc816d0a0>
```



* Code for plotting all graphs :-

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

```
df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(new).csv')
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
df
count = 0
circulatory = 0
respiratory = 0
digestive = 0
diabetes = 0
injury = 0
musculoskeletal = 0
genitourinary = 0
noeplasms = 0i
other = 0
number_of_encounters = []
group_name = ["Circulatory","Respiratory","Digestive","Diabetes","Injury","Musculoskeletal","Genitourinary","Neoplasms","Other"]
for i in df['diag_1']:
    if((i.isnumeric())):
        a =  int(i)
        if((a in range(390,460)) or a==785):
            circulatory+=1
        elif ((a in range(460,520)) or a==786):
            respiratory+=1
        elif ((a in range(520,580)) or a==787):
            digestive+=1
        elif ((a in range(800,1000))):
            injury+=1
        elif ((a in range(710,740))):
            musculoskeletal+=1
        elif ((a in range(580,630)) or a==788):
            genitourinary+=1
        elif ((a in range(140,240))):
            noeplasms+=1
        else:
            other+=1
    elif (i[0]=='2' and i[1]=='5' and i[2]=='0'):
        diabetes+=1
    else:
        other+=1
    count+=1
print("circulatory = ",circulatory)
```

```
print("respiratory = ",respiratory)
print("digestive = ",digestive)
print("diabetes = ",diabetes)
print("injury = ",injury)
print("muscuskeletal = ",musculoskeletal)
print("genitu = ",genitourinary)
print("neop = ",noeplasms)
print("other = ",other)

number_of_encounters.append(circulatory)
number_of_encounters.append(respiratory)
number_of_encounters.append(digestive)
number_of_encounters.append(diabetes)
number_of_encounters.append(injury)
number_of_encounters.append(musculoskeletal)
number_of_encounters.append(genitourinary)
number_of_encounters.append(noeplasms)
number_of_encounters.append(other)
print(number_of_encounters)


y_pos = np.arange(len(group_name))
plt.bar(y_pos, number_of_encounters, color = (0.5,0.1,0.5,0.6))
plt.xlabel('Group Name')
plt.ylabel('No of Encounters')
# Create names
plt.xticks(y_pos, group_name,rotation='vertical')

# Show graphic
plt.show()

no_of_lab_procedures = df['num_lab_procedures']
no_of_medications = df['num_medications']
histogram = pd.DataFrame({'num_lab_procedures': no_of_lab_procedures, 'num_medications': no_
of_medications})
histogram
histogram.hist()

df.boxplot(by ='gender', column =['num_medications'], grid = False)
```

Standardization and Normalization:

Mean ,variance for each of the columns and some results using grouping:

SOME STATISTICAL MEASURES FROM THE CLEANED DATASET :

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [110]:
```
import pandas as pd
import numpy as np
data=pd.read_csv("E:\data science project\cleaned_diabetic_dataset.csv")
#Stastical measures for the column time_in_hospital
print("Mean time of all the patients in the hospital: " + str(data['time_in_hospital'].mean()))
print("Median time of all the patients in the hospital: " + str(data['time_in_hospital'].median()))
print("Mode time of all the patients in the hospital: " + str( data['time_in_hospital'].mode()))
print("Measure of standard deviation for time of all the patients in the hospital: " + str(data['time_in_hospital'].std()))
print("Measure of variance of time of all the patients in the hospital: " + str(data['time_in_hospital'].var()))
print("Sum time of all the patients in the hospital: " + str(data['time_in_hospital'].sum()))
print("Maximum time spent among all the patients in the hospital: " + str(data['time_in_hospital'].max()))
print("Minimum time spent among all the patients in the hospital: " + str(data['time_in_hospital'].min()))
print("Count of time of all the patients in the hospital: " + str(data['time_in_hospital'].count()))
```

```
Mean time of all the patients in the hospital: 3.9513216523166603
Median time of all the patients in the hospital: 4.0
Mode time of all the patients in the hospital: 0    4.0
dtype: float64
Measure of standard deviation for time of all the patients in the hospital: 2.1194422200986622
Measure of variance of time of all the patients in the hospital: 4.492035324336746
Sum time of all the patients in the hospital: 117943.0
Maximum time spent among all the patients in the hospital: 9.0
Minimum time spent among all the patients in the hospital: 1.0
Count of time of all the patients in the hospital: 29849
```

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [111]:
```
#Stastical measures for the column num_lab_procedures
print("Mean of number of lab procedures done : " + str(data['num_lab_procedures'].mean()))
print("Median of number of lab procedures done: " + str(data['num_lab_procedures'].median()))
print("Mode of number of lab procedures done: " + str( data['num_lab_procedures'].mode()))
print("Measure of standard deviation for number of lab procedures done: " + str(data['num_lab_procedures'].std()))
print("Measure of variance of number of lab procedures done: " + str(data['num_lab_procedures'].var()))
print("Sum of number of lab procedures done: " + str(data['num_lab_procedures'].sum()))
print("Maximum of number of lab procedures done: " + str(data['num_lab_procedures'].max()))
print("Minimum of number of lab procedures done: " + str(data['num_lab_procedures'].min()))
print("Count of number of lab procedures done: " + str(data['num_lab_procedures'].count()))
```

```
Mean of number of lab procedures done : 44.71399376863547
Median of number of lab procedures done: 46.0
Mode of number of lab procedures done: 0    68.0
dtype: float64
Measure of standard deviation for number of lab procedures done: 15.838435827992386
Measure of variance of number of lab procedures done: 250.85604947743286
Sum of number of lab procedures done: 1334668.0
Maximum of number of lab procedures done: 68.0
Minimum of number of lab procedures done: 19.0
Count of number of lab procedures done: 29849
```

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code

```
Minimum of number of lab procedures done: 19.0
Count of number of lab procedures done: 29849
```

In [112]:
```
#Stastical measures for the column num_medications
print("Mean of number of distinct generic names : " + str(data['num_medications'].mean()))
print("Median of number of distinct generic names: " + str(data['num_medications'].median()))
print("Mode of number of distinct generic names: " + str( data['num_medications'].mode()))
print("Measure of standard deviation for number of distinct generic names: " + str(data['num_medications'].std()))
print("Measure of variance for number of distinct generic names: " + str(data['num_medications'].var()))
print("Sum of number of distinct generic names: " + str(data['num_medications'].sum()))
print("Maximum of number of distinct generic names: " + str(data['num_medications'].max()))
print("Minimum of number of distinct generic names: " + str(data['num_medications'].min()))
print("Count of number of distinct generic names: " + str(data['num_medications'].count()))
```

```
Mean of number of distinct generic names : 14.707427384501994
Median of number of distinct generic names: 14.0
Mode of number of distinct generic names: 0    7.0
dtype: float64
Measure of standard deviation for number of distinct generic names: 5.819109302910811
Measure of variance for number of distinct generic names: 33.862033079223146
Sum of number of distinct generic names: 439002.0
Maximum of number of distinct generic names: 25.0
Minimum of number of distinct generic names: 7.0
Count of number of distinct generic names: 29849
```

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [113]:
```
#Stastical measures for the column Weight
print("Mean of weight of all patients : " + str(data['Weight'].mean()))
print("Median of weight of all patients: " + str(data['Weight'].median()))
print("Mode of weight of all patients: " + str( data['Weight'].mode()))
print("Measure of standard deviation for weight of all patients: " + str(data['Weight'].std()))
print("Measure of variance for weight of all patients: " + str(data['Weight'].var()))
print("Sum of weight of all patients: " + str(data['Weight'].sum()))
print("Maximum of weight among all patients: " + str(data['Weight'].max()))
print("Minimum of weight among all patients: " + str(data['Weight'].min()))
```

```
Mean of weight of all patients : 161.46662643992676
Median of weight of all patients: 161.4222109
Mode of weight of all patients: 0    203.596695
dtype: float64
Measure of standard deviation for weight of all patients: 28.506612164165798
Measure of variance for weight of all patients: 812.6269370781655
Sum of weight of all patients: 4819617.3326059
Maximum of weight among all patients: 203.59669540000002
Minimum of weight among all patients: 119.63120644
```

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [114]:
```
#Stastical measures for the column number_diagnoses
print("Mean of number of diagnoses entered : " + str(data['number_diagnoses'].mean()))
print("Median of number of diagnoses entered: " + str(data['number_diagnoses'].median()))
print("Mode of number of diagnoses entered: " + str( data['number_diagnoses'].mode()))
print("Measure of standard deviation for number of diagnoses entered: " + str(data['number_diagnoses'].std()))
print("Measure of variance for number of diagnoses entered: " + str(data['number_diagnoses'].var()))
print("Sum of number of diagnoses entered: " + str(data['number_diagnoses'].sum()))
print("Maximum of number of diagnoses entered: " + str(data['number_diagnoses'].max()))
print("Minimum of number of diagnoses entered: " + str(data['number_diagnoses'].min()))
print("Count of number of diagnoses entered: " + str(data['number_diagnoses'].count()))
```

```
Mean of number of diagnoses entered : 6.821702569600322
Median of number of diagnoses entered: 7.0
Mode of number of diagnoses entered: 0    9.0
dtype: float64
Measure of standard deviation for number of diagnoses entered: 2.0035072193066052
Measure of variance for number of diagnoses entered: 4.014041177813686
Sum of number of diagnoses entered: 203621.0
Maximum of number of diagnoses entered: 9.0
Minimum of number of diagnoses entered: 2.0
Count of number of diagnoses entered: 29849
```

USING GROUPING :

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [ ]:

In [115]:
```
#Some results generated by grouping the columns
print("Count of numbers,grouped by the Gender:")
print(str(data.groupby('gender')["age"].count()))
print("Count of numbers,grouped by Age:")
print(str(data.groupby('age')["gender"].count()))
```

```
Count of numbers,grouped by the Gender:
gender
Female    15981
Male      13868
Name: age, dtype: int64
Count of numbers,grouped by Age:
age
[0-10)        89
[10-20)      321
[20-30)      481
[30-40)     1328
[40-50)     3187
[50-60)     5400
[60-70)     6450
[70-80)     7938
[80-90)     4069
[90-100)     586
Name: gender, dtype: int64
```

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [116]:
```
print("Count of number of patients witnessing change as well as no change:")
print(str(data.groupby('change')["gender"].count()))
```

```
Count of number of patients witnessing change as well as no change:
change
Ch    11346
No    18503
Name: gender, dtype: int64
```

In [117]:
```
print("Count of number of patients readmitted:")
print(str(data.groupby('readmitted')["gender"].count()))
```

```
Count of number of patients readmitted:
readmitted
<30     3464
>30     9870
NO     16515
Name: gender, dtype: int64
```

jupyter Untitled Last Checkpoint: Yesterday at 7:32 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
Name: gender, dtype: int64
```

In [118]:
```
print("Count of number of patients for each admission type id:")
print(str(data.groupby('admission_type_id')["gender"].count()))
```

```
Count of number of patients for each admission type id:
admission_type_id
1.0    14666
2.0     5079
3.0     4586
4.0        4
5.0     2304
6.0     3210
Name: gender, dtype: int64
```

In [119]:
```
print("Count of number of patients based on insulin:")
print(str(data.groupby('insulin')["gender"].count()))
```

```
Count of number of patients based on insulin:
insulin
Down       2567
No        16353
Steady     9289
Up         1640
Name: gender, dtype: int64
```

2.Normalizing all the numeric columns to make mean 0 and variance 1:

<u>Output Screenshots of Normalisation and Standardisation</u>

1) code :-

```
import pandas as pd
from sklearn import preprocessing
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_dia
betic_dataset(new).csv')
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
from sklearn import preprocessing
import numpy as np
df1 = pd.DataFrame({'time_in_hospital':df['time_in_hospital'],'num_lab_procedures':df['num_lab_p
rocedures'],'num_medications':df['num_medications'],'number_diagnoses':df['number_diagnoses'],'
Weight':df['Weight']})

x = df1.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df1 = pd.DataFrame(x_scaled)
df1
# Get column names first
names = df1.columns
# Create the Scaler object
scaler = preprocessing.StandardScaler()
# Fit your data on the scaler object
scaled_df = scaler.fit_transform(df1)
scaled_df = pd.DataFrame(scaled_df, columns=names)
normalized_df = scaled_df.rename(columns = {0: 'time_in_hospital', 1: 'num_lab_procedures', 2:'nu
m_medications', 3:'number_diagnoses', 4:'Weight'}, inplace = False)
print(round(normalized_df.mean()))
print(round(normalized_df.var()))
print(normalized_df)

plt.hist(normalized_df['time_in_hospital'])
plt.show()

plt.hist(normalized_df['num_lab_procedures'])
plt.show()

plt.hist(normalized_df['num_medications'])
plt.show()

plt.hist(normalized_df['number_diagnoses'])
```

```
plt.show()

plt.hist(normalized_df['Weight'])
plt.show()
```

```
In [68]: import pandas as pd
         from sklearn import preprocessing
         import numpy as np
         df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(new).csv')
         df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
         from sklearn import preprocessing
         import numpy as np
         df1 = pd.DataFrame({'time_in_hospital':df['time_in_hospital'],'num_lab_procedures':df['num_lab_procedures'],'num_medications':df[

         x = df1.values #returns a numpy array
         min_max_scaler = preprocessing.MinMaxScaler()
         x_scaled = min_max_scaler.fit_transform(x)
         df1 = pd.DataFrame(x_scaled)
         df1
         # Get column names first
         names = df1.columns
         # Create the Scaler object
         scaler = preprocessing.StandardScaler()
         # Fit your data on the scaler object
         scaled_df = scaler.fit_transform(df1)
         scaled_df = pd.DataFrame(scaled_df, columns=names)
         normalized_df = scaled_df.rename(columns = {0: 'time_in_hospital', 1: 'num_lab_procedures', 2:'num_medications', 3:'number_diagno
         print(round(normalized_df.mean()))
         print(round(normalized_df.var()))
         normalized_df
```

Output:-

```
time_in_hospital      0.0
num_lab_procedures    0.0
num_medications      -0.0
number_diagnoses      0.0
Weight               -0.0
dtype: float64
time_in_hospital      1.0
num_lab_procedures    1.0
num_medications       1.0
number_diagnoses      1.0
Weight                1.0
dtype: float64
```

Out[68]:

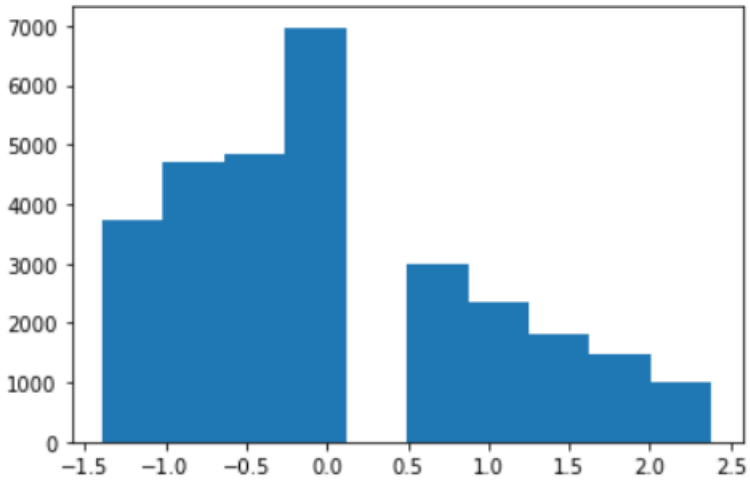
	time_in_hospital	num_lab_procedures	num_medications	number_diagnoses	Weight
0	-0.448862	0.901999	0.565830	1.087260	0.029602
1	-0.920692	-1.623546	-0.293422	-0.410139	1.477930
2	-0.920692	-0.045081	0.222129	0.088994	1.477930
3	-1.392523	0.396890	-1.152675	-0.909272	1.477930
4	-0.448862	-0.865882	0.222129	1.087260	-0.324648
...	...	...	...	...	...
29844	0.966628	0.396890	1.768784	1.087260	-0.650605
29845	0.966628	0.460028	-1.324525	-0.909272	-1.467594
29846	0.966628	1.470246	1.253232	1.087260	-1.341215
29847	0.022968	-0.423912	-0.293422	-0.410139	-1.220335
29848	-1.392523	1.028276	-0.465273	-0.909272	0.580719

29849 rows × 5 columns

2) checking whether the data is normal:-

a) checking in time_in_hospital column :-

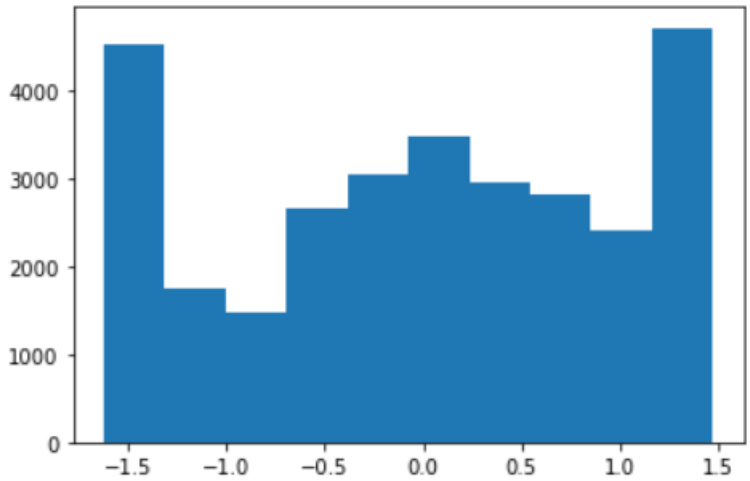
```
In [63]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         plt.hist(normalized_df['time_in_hospital'])
         plt.show()
```



It is not normal

b) checking in num_lab_procedures column

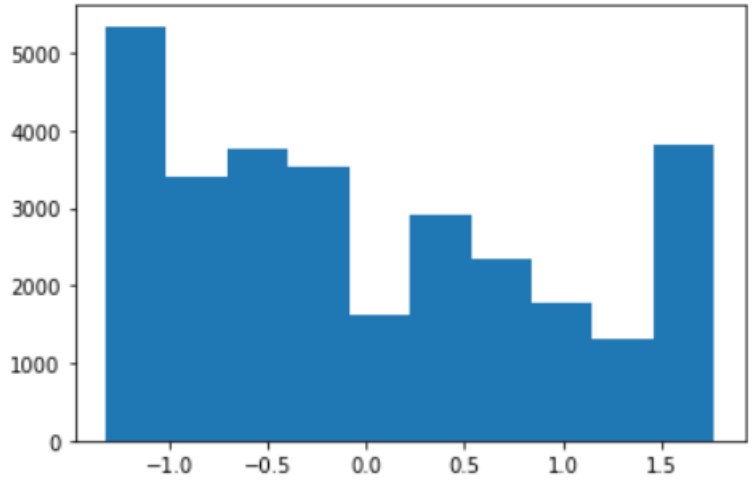
```
In [64]: plt.hist(normalized_df['num_lab_procedures'])
         plt.show()
```



It is not normal

c) checking in num_medications column :-

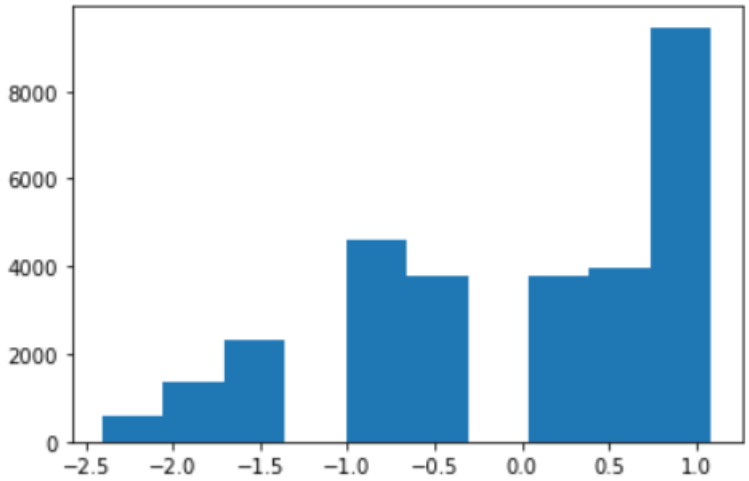
```
In [65]: plt.hist(normalized_df['num_medications'])
         plt.show()
```



It is not normal

d) checking in number_diagnoses column :-

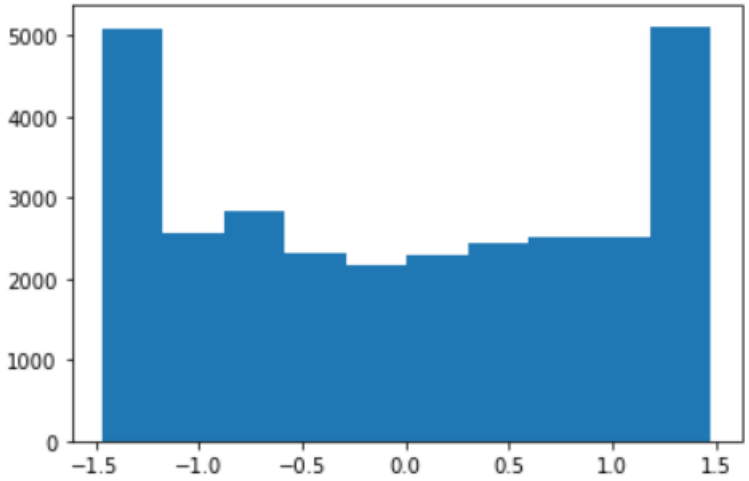
```
In [66]: plt.hist(normalized_df['number_diagnoses'])
         plt.show()
```



It is not normal

e) checking in Weight column :-

```
In [67]: plt.hist(normalized_df['Weight'])
         plt.show()
```



It is not normal

Why is normalization needed? How does it affect dataset?

Normalization in dataset is needed to logically group data together. We want data that relates to each other to be stored together. This will occur in a database which has undergone data normalization. If data is dependent on each other, they should be in close proximity within the data set. It is usually through data normalization that the information within a database can be formatted in such a way that it can be visualized and analyzed. Without it, all the data can ne collected but most of it will simply go unused, taking up space and not benefiting the organization in any meaningful way.

Normalization affects the dataset as it changes the values of numeric columns in the **dataset** to a common scale, without distorting differences in the ranges of values. So we **normalize** the data to bring all the variables to the same range, to make data dependent on each other in such a way that data can be visualized and analyzed.

Hypothesis Testing

Output Screenshots of Hypothesis Testing

```
In [88]: import pandas as pd
         import seaborn as sns
         from scipy import stats as st
         df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(new).csv')

         # performing STATISTICAL TESTS
         # performin one sample t-test (testing whether it is equal to population mean or not)
         df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
         st.ttest_1samp(df['time_in_hospital'],4)  # Ho:mu = 4  H1:mu not equal to 4
         # so p=7.2626e-05 which gives p < 0.05 hence null hypothesis is rejected

Out[88]: Ttest_1sampResult(statistic=-3.968068310095117, pvalue=7.262688187553114e-05)

In [91]: # performing two sample t-test,testing whether the mean in number of medications in the male and female populations were differer
         # To test if this is significant, we do a 2-sample t-test with scipy.stats.ttest_ind():
         female_medications = df[df['gender'] == 'Female']['num_medications']
         male_medications = df[df['gender'] == 'Male']['num_medications']
         st.ttest_ind(female_medications, male_medications)   # Ho:mu(female) not equal to mu(male)   H1:mu(male) = mu(female)
         # so p=0.0225 which gives p < 0.05 hence null hypothesis is rejected

Out[91]: Ttest_indResult(statistic=2.280895199641611, pvalue=0.02256166307773206)
```

```
In [102]: no_of_labProcedures = df['num_lab_procedures']
          mu = no_of_labProcedures.mean()
          sigma = no_of_labProcedures.std(ddof=0)
          print("mu = ",mu,"sigma = ",sigma)

          mu =  44.71399376863547 sigma =  15.838170516449539

In [103]: df['num_lab_procedures'].head().mean()

Out[103]: 40.8

In [104]: z_critical = 1.96
          mean = 40.8
          n = 29849
          standard_deviation = sigma/np.sqrt(n)
          z = (mean - mu)/standard_deviation    # Ho: sample mean = population mean  H1:sample mean not equal to population mean
          print(z)
          # z=-42.69 which gives z < 1.96 hence rejecting null hypotheses

          -42.69529400887596
```

1) In the one sample t-test, we are taking Ho and H1 as (Ho: mu = 4 and H1: mu not equal to 4)

We got p-value as 7.2626e-05,which is less than 0.05 (alpha), hence null hypothesis is rejected

2) In the two sample t-test, we are taking Ho and H1 as (Ho: mean in number of medications of female and mean in number of medications of male are different AND H1: mean in number of medications of female and mean in number of medications of male are same.)

We got p-value as 0.025,which is less than 0.05 (alpha), hence null hypothesis is rejected

3) Testing on population mean, here we are taking Ho and H1 as (Ho: sample mean of number of lab procedure is equal to population mean AND H1: sample mean of number of lab procedure is not equal to population mean)

We got z-value as -42.6952,which is less than 1.96, hence null hypothesis is rejected

code :-

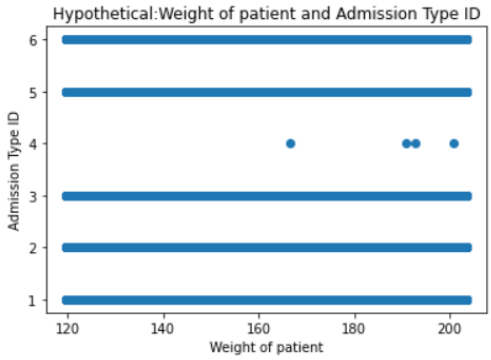
```
import pandas as pd
```

```
import seaborn as sns
from scipy import stats as st
df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_dia
betic_dataset(new).csv')

# performing STATISTICAL TESTS
# performin one sample t-test (testing whether it is equal to population mean or not)
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
st.ttest_1samp(df['time_in_hospital'],4)  # Ho:mu = 4  H1:mu not equal to 4
# so p=7.2626e-05 which gives p < 0.05 hence null hypothesis is rejected

# performing two sample t-
test,testing whether the mean in number of medications in the male and female populations were dif
ferent.
# To test if this is significant, we do a 2-sample t-test with scipy.stats.ttest_ind():
female_medications = df[df['gender'] == 'Female']['num_medications']
male_medications = df[df['gender'] == 'Male']['num_medications']
st.ttest_ind(female_medications, male_medications)   # Ho:mu(female) not equal to mu(male)   H1:
mu(male) = mu(female)
# so p=0.0225 which gives p < 0.05 hence null hypothesis is rejected

no_of_labProcedures = df['num_lab_procedures']
mu = no_of_labProcedures.mean()
sigma = no_of_labProcedures.std(ddof=0)
print("mu = ",mu,"sigma = ",sigma)
print(df['num_lab_procedures'].head().mean())
z_critical = 1.96
mean = 40.8
n = 29849
standard_deviation = sigma/np.sqrt(n)
z = (mean - mu)/standard_deviation    # Ho: sample mean = population mean  H1:sample mean not
equal to population mean
print(z)
# z=-42.69 which gives z < 1.96 hence rejecting null hypotheses
```

Correlation:

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related. An intelligent correlation analysis can lead to a greater understanding of your data. **Correlation works for quantifiable data** in which numbers are meaningful, usually quantities of some sort. It cannot be used for purely categorical data, such as gender, brands purchased, or favourite colour.

In [121]:
```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleaned_diabetic_dataset(new).csv')
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
df
# Draw the scatter plot

plt.scatter(df['Weight'], df['admission_type_id'])
plt.title('Hypothetical:Weight of patient and Admission Type ID')
plt.xlabel('Weight of patient')
plt.ylabel('Admission Type ID')
plt.show()
```




```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('C:/Users/HRITHIK/OneDrive/Desktop/Data Science Project/Project/cleane
d_diabetic_dataset(new).csv')
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
print(df)

# Draw the scatter plot
plt.scatter(df['Weight'], df['admission_type_id'])
plt.title('Hypothetical:Weight of patient and Admission Type ID')
plt.xlabel('Weight of patient')
plt.ylabel('Admission Type ID')
plt.show()
```

In the above code the correlation between admission type id and weight of patient is done and concluded that is neither positive nor negative.

```
In [124]: correlation_df = df.corr()
          print(correlation_df)
```

```
                     encounter_id  time_in_hospital  num_lab_procedures  \
encounter_id             1.000000         -0.018707           -0.094137
time_in_hospital        -0.018707          1.000000            0.252497
num_lab_procedures      -0.094137          0.252497            1.000000
num_medications          0.012307          0.399274            0.154230
number_diagnoses        -0.000653          0.229935            0.147000
Weight                  -0.022599         -0.003355            0.009316
admission_type_id       -0.066545         -0.019031           -0.262293

                     num_medications  number_diagnoses    Weight  \
encounter_id                0.012307         -0.000653 -0.022599
time_in_hospital            0.399274          0.229935 -0.003355
num_lab_procedures          0.154230          0.147000  0.009316
num_medications             1.000000          0.288752 -0.014792
number_diagnoses            0.288752          1.000000 -0.003741
Weight                     -0.014792         -0.003741  1.000000
admission_type_id           0.119365         -0.055160  0.022416

                     admission_type_id
encounter_id                 -0.066545
time_in_hospital             -0.019031
num_lab_procedures           -0.262293
num_medications               0.119365
number_diagnoses             -0.055160
Weight                        0.022416
admission_type_id             1.000000
```

```
In [130]: column = df['time_in_hospital']
          max_value = column.max()
          print(max_value)
          df[df.values == 9.0]
```

```
9.0
```

Out[130]:

	encounter_id	gender	age	time_in_hospital	medical_specialty	num_lab_procedures	num_medications	diag_1	number_diagnoses	insulin	diabete
0	149190	Female	[10-20)	3.0	other	59.0	18.0	276	9.0	Up	
4	35754	Male	[50-60)	3.0	other	31.0	16.0	414	9.0	Steady	
9	28236	Female	[40-50)	9.0	other	47.0	17.0	250.7	9.0	Steady	
9	28236	Female	[40-50)	9.0	other	47.0	17.0	250.7	9.0	Steady	
14	73578	Male	[60-70)	4.0	other	68.0	13.0	999	9.0	Up	
...	...	...	...	...	...	...	...	...	...	...	
29810	97389216	Male	[70-80)	2.0	other	19.0	13.0	276	9.0	Steady	
29828	97428558	Male	[60-70)	8.0	InternalMedicine	36.0	20.0	250.8	9.0	No	
29835	97442082	Male	[70-80)	5.0	InternalMedicine	19.0	16.0	428	9.0	No	
29844	97461414	Male	[50-60)	6.0	Surgery-Cardiovascular/Thoracic	51.0	25.0	410	9.0	No	
29846	97463718	Female	[80-90)	6.0	InternalMedicine	68.0	22.0	410	9.0	Down	

12066 rows × 15 columns

Results and Conclusions :- The decision to obtain a measurement of HbA1c for patients with diabetes mellitus is a useful predictor of readmission rates which may provide valuable in the development of strategies to reduce readmission rates and costs for the care of individuals with

diabetes mellitus. For instance,our analysis showed that the profile of readmission differed significantly in patients where HbA1c was checked in the setting of a primary diabetes diagnosis,when compared to those with a primary circulatory disorder. While readmission rates remained the highest for patients with circulatory diagnoses,readmission rates for patients with diabetes appeared to be associated with the decision to test for HbA1c,rather than the values of the HbA1c result.