About Dataset

Feature	explanation	Values
Diabetes_binary	diabetes	0 = no 1 = yes
HighBP	high BP	0 = no 1 = yes
HighChol		0 = no high cholesterol 1 = high cholesterol
CholCheck	cholesterol check in 5 years	0 = no 1 = yes
BMI	Body Mass Index	
Smoker	Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes]	0 = no 1 = yes
Stroke	(Ever told) you had a stroke.	0 = no 1 = yes
HeartDiseaseorAttack	coronary heart disease (CHD) or myocardial infarction (MI)	0 = no 1 = yes
PhysActivity	physical activity in past 30 days - not including job	0 = no 1 = yes
Fruits	Consume Fruit 1 or more times per day	0 = no 1 = yes
Veggies	Consume Vegetables 1 or more times per day	0 = no 1 = yes
HvyAlcoholConsump	(adult men >=14 drinks per week and adult women>=7 drinks per week)	0 = no 1 = yes
AnyHealthcare	Have any kind of health care coverage, including health insurance, prepaid plans such as HMO, etc.	0 = no 1 = yes

Feature	explanation	Values
NoDocbcCost	Was there a time in the past 12 months when you needed to see a doctor but could not because of cost?	0 = no 1 = yes
GenHlth	Would you say that in general your health is:	1 = excellent 2 = very good 3 = good 4 = fair 5 = poor
MentHlth	days of poor mental health scale 1-30 days	1-30
PhysHlth	physical illness or injury days in past 30 days	1-30
DiffWalk	Do you have serious difficulty walking or climbing stairs?	0 = no 1 = yes
Sex	Gender	0 = female 1 = male
Age	13-level age category (_AGEG5YR see codebook)	1 = 18-24 9 = 60-64 13 = 80 or older
Education	Education level (EDUCA see codebook)	scale 1-6 1 = Never attended school or only kindergarten 2 = elementary etc.
Income	Income scale (INCOME2 see codebook)	scale 1-8 1 = less than $10,0005 = lessthan35,000$ 8 = $75,000 or more

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import os
import warnings
warnings.filterwarnings('ignore')
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
/kaggle/input/diabetes-health-indicators-dataset/diabetes_binary_5050split_health_indicators_BRFSS2015.csv
/kaggle/input/diabetes-health-indicators-dataset/diabetes_binary_health_indicators_BRFSS2015.csv
/kaggle/input/diabetes-health-indicators-dataset/diabetes_012_health_indicators_BRFSS2015.csv
```

Loading the dataset

In [2]:
```
#Loading the dataset
df = pd.read_csv("/kaggle/input/diabetes-health-indicators-datase
df.head()
```

Out[2]:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	He
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	

5 rows × 22 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253680 entries, 0 to 253679
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   Diabetes_binary       253680 non-null  float64
 1   HighBP                253680 non-null  float64
 2   HighChol              253680 non-null  float64
 3   CholCheck             253680 non-null  float64
 4   BMI                   253680 non-null  float64
 5   Smoker                253680 non-null  float64
 6   Stroke                253680 non-null  float64
 7   HeartDiseaseorAttack  253680 non-null  float64
 8   PhysActivity          253680 non-null  float64
 9   Fruits                253680 non-null  float64
 10  Veggies               253680 non-null  float64
 11  HvyAlcoholConsump     253680 non-null  float64
 12  AnyHealthcare         253680 non-null  float64
 13  NoDocbcCost           253680 non-null  float64
 14  GenHlth               253680 non-null  float64
 15  MentHlth              253680 non-null  float64
 16  PhysHlth              253680 non-null  float64
 17  DiffWalk              253680 non-null  float64
 18  Sex                   253680 non-null  float64
 19  Age                   253680 non-null  float64
 20  Education             253680 non-null  float64
 21  Income                253680 non-null  float64
dtypes: float64(22)
memory usage: 42.6 MB
```

In [4]:
```
df.describe()
```

Out[4]:

	Diabetes_binary	HighBP	HighChol	CholCheck	
count	253680.000000	253680.000000	253680.000000	253680.000000	253680.
mean	0.139333	0.429001	0.424121	0.962670	28.
std	0.346294	0.494934	0.494210	0.189571	6.
min	0.000000	0.000000	0.000000	0.000000	12.
25%	0.000000	0.000000	0.000000	1.000000	24.
50%	0.000000	0.000000	0.000000	1.000000	27.
75%	0.000000	1.000000	1.000000	1.000000	31.
max	1.000000	1.000000	1.000000	1.000000	98.

8 rows × 22 columns

Data Preprocessing

In [5]:
```
#Checking for null values
df.isnull().sum().sum()
```

Out[5]:
```
0
```

In [6]:
```
#Checking for duplicated
df.duplicated().sum()
```

Out[6]:
```
24206
```

In [7]:
```
#Drop duplicate rows
df.drop_duplicates(inplace = True)
df.shape
```

Out[7]:
```
(229474, 22)
```

In [8]:
```
df['Diabetes'] = df['Diabetes_binary']
df.drop(columns = 'Diabetes_binary', inplace=True)
```

In [9]:
```
# Value count for each value
for i in df.columns:
```

```
    print(i,'\n',df[i].value_counts())
    print('-'*90)
```

```
HighBP
 0.0    125214
1.0    104260
Name: HighBP, dtype: int64
-------------------------------------------------------------------------
------------------------
HighChol
 0.0    128129
1.0    101345
Name: HighChol, dtype: int64
-------------------------------------------------------------------------
------------------------
CholCheck
 1.0    220176
0.0      9298
Name: CholCheck, dtype: int64
-------------------------------------------------------------------------
------------------------
BMI
 27.0    21514
26.0    17775
24.0    16497
28.0    14914
25.0    14793
        ...
85.0        1
91.0        1
86.0        1
90.0        1
78.0        1
Name: BMI, Length: 84, dtype: int64
-------------------------------------------------------------------------
------------------------
Smoker
 0.0    122585
1.0    106889
Name: Smoker, dtype: int64
-------------------------------------------------------------------------
------------------------
Stroke
 0.0    219190
1.0     10284
Name: Stroke, dtype: int64
-------------------------------------------------------------------------
------------------------
HeartDiseaseorAttack
 0.0    205761
1.0     23713
```

```
Name: HeartDiseaseorAttack, dtype: int64
------------------------------------------------------------------------
------------------------
PhysActivity
 1.0    168214
0.0     61260
Name: PhysActivity, dtype: int64
------------------------------------------------------------------------
------------------------
Fruits
 1.0    140593
0.0     88881
Name: Fruits, dtype: int64
------------------------------------------------------------------------
------------------------
Veggies
 1.0    182337
0.0     47137
Name: Veggies, dtype: int64
------------------------------------------------------------------------
------------------------
HvyAlcoholConsump
 0.0    215524
1.0     13950
Name: HvyAlcoholConsump, dtype: int64
------------------------------------------------------------------------
------------------------
AnyHealthcare
 1.0    217085
0.0     12389
Name: AnyHealthcare, dtype: int64
------------------------------------------------------------------------
------------------------
NoDocbcCost
 0.0    208151
1.0     21323
Name: NoDocbcCost, dtype: int64
------------------------------------------------------------------------
------------------------
GenHlth
 2.0    77365
3.0    73632
1.0    34854
4.0    31545
5.0    12078
Name: GenHlth, dtype: int64
------------------------------------------------------------------------
------------------------
```

```
MentHlth
 0.0     152325
2.0      12692
30.0     12079
5.0       8913
1.0       8307
3.0       7301
10.0      6352
15.0      5501
4.0       3774
20.0      3362
7.0       3090
25.0      1188
14.0      1167
6.0        988
8.0        639
12.0       398
28.0       327
21.0       227
29.0       158
18.0        97
9.0         91
16.0        88
27.0        79
22.0        63
17.0        54
26.0        45
11.0        41
13.0        41
23.0        38
24.0        33
19.0        16
Name: MentHlth, dtype: int64
------------------------------------------------------------------------
-------------------------
PhysHlth
 0.0     136578
30.0     19385
2.0      14491
1.0      11073
3.0       8435
5.0       7595
10.0      5588
15.0      4914
7.0       4531
4.0       4521
20.0      3273
14.0      2584
```

```
25.0      1336
6.0       1328
8.0        809
21.0       663
12.0       578
28.0       522
29.0       215
9.0        179
18.0       152
16.0       112
27.0        99
17.0        96
24.0        72
22.0        70
26.0        69
13.0        68
11.0        60
23.0        56
19.0        22
Name: PhysHlth, dtype: int64
------------------------------------------------------------------------
------------------------
DiffWalk
 0.0    186849
1.0     42625
Name: DiffWalk, dtype: int64
------------------------------------------------------------------------
------------------------
Sex
 0.0    128715
1.0    100759
Name: Sex, dtype: int64
------------------------------------------------------------------------
------------------------
Age
 9.0     29678
10.0    29093
8.0     27272
7.0     23121
11.0    21993
6.0     17280
13.0    16791
12.0    15379
5.0     14040
4.0     12229
3.0     10023
2.0      7064
1.0      5511
```

```
Name: Age, dtype: int64
------------------------------------------------------------------------
------------------------
Education
 6.0    88225
5.0    66444
4.0    61124
3.0     9467
2.0     4040
1.0      174
Name: Education, dtype: int64
------------------------------------------------------------------------
------------------------
Income
 8.0    71640
7.0    40131
6.0    34957
5.0    25326
4.0    19953
3.0    15920
2.0    11756
1.0     9791
Name: Income, dtype: int64
------------------------------------------------------------------------
------------------------
Diabetes
 0.0    194377
1.0     35097
Name: Diabetes, dtype: int64
------------------------------------------------------------------------
------------------------
```

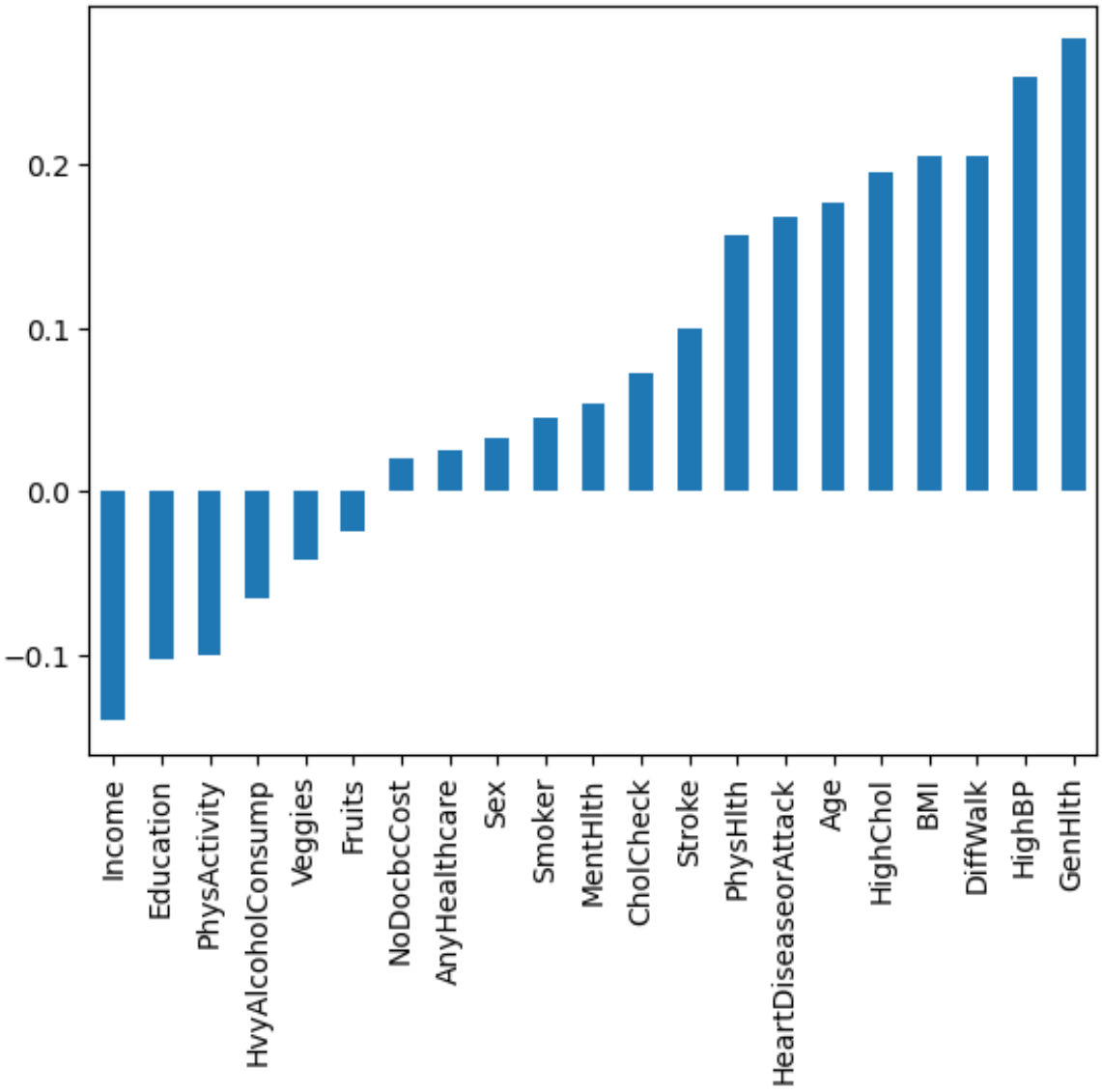
In [10]:
```
#Change dtype to int
df = df.astype('int32')
```

EDA

Find correlation between the variables and the target variable (Diabetes_binary)

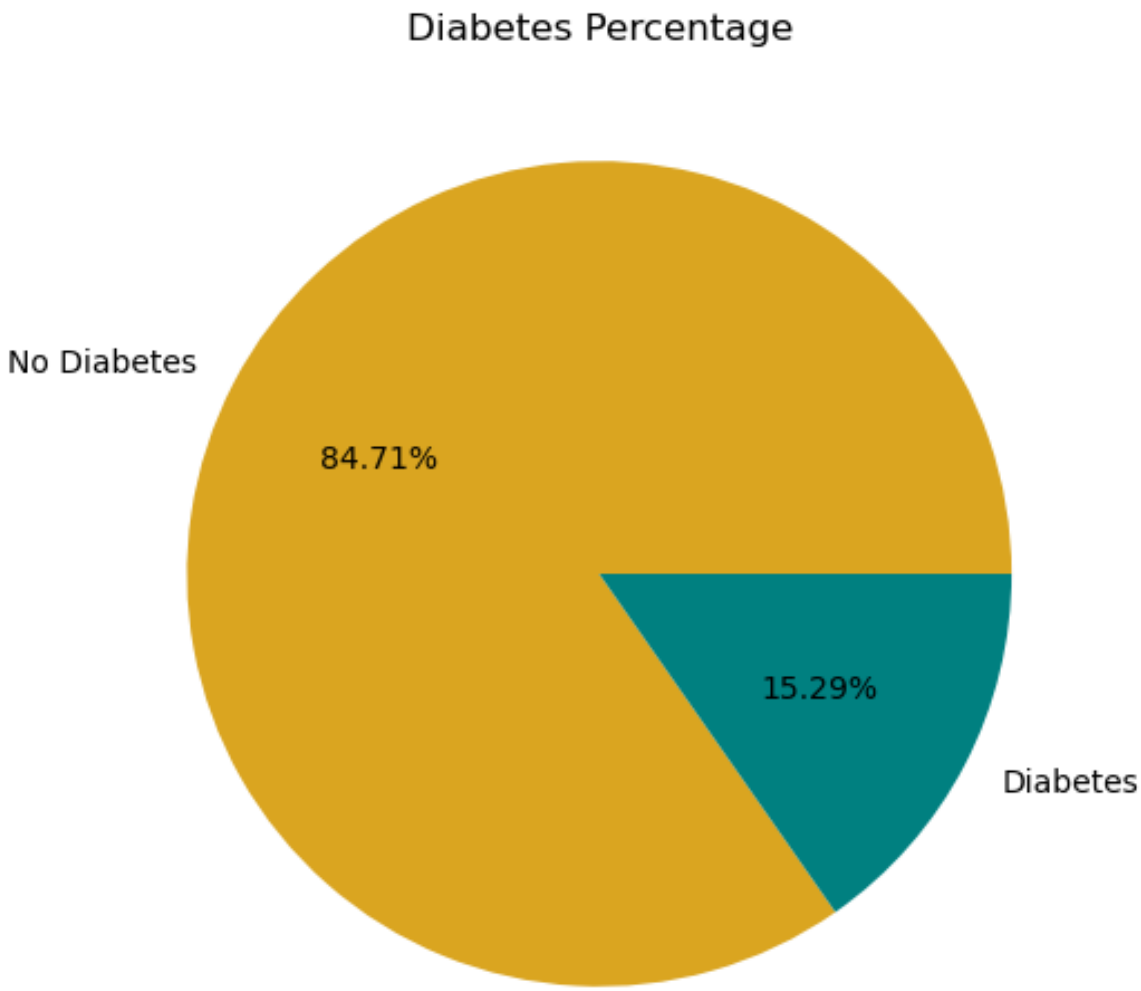
In [11]:
```
#Plot correlation between the variables and Diabetes_binary
df.corr()['Diabetes'][:-1].sort_values().plot(kind='bar')
```

Out[11]:
```
<Axes: >
```



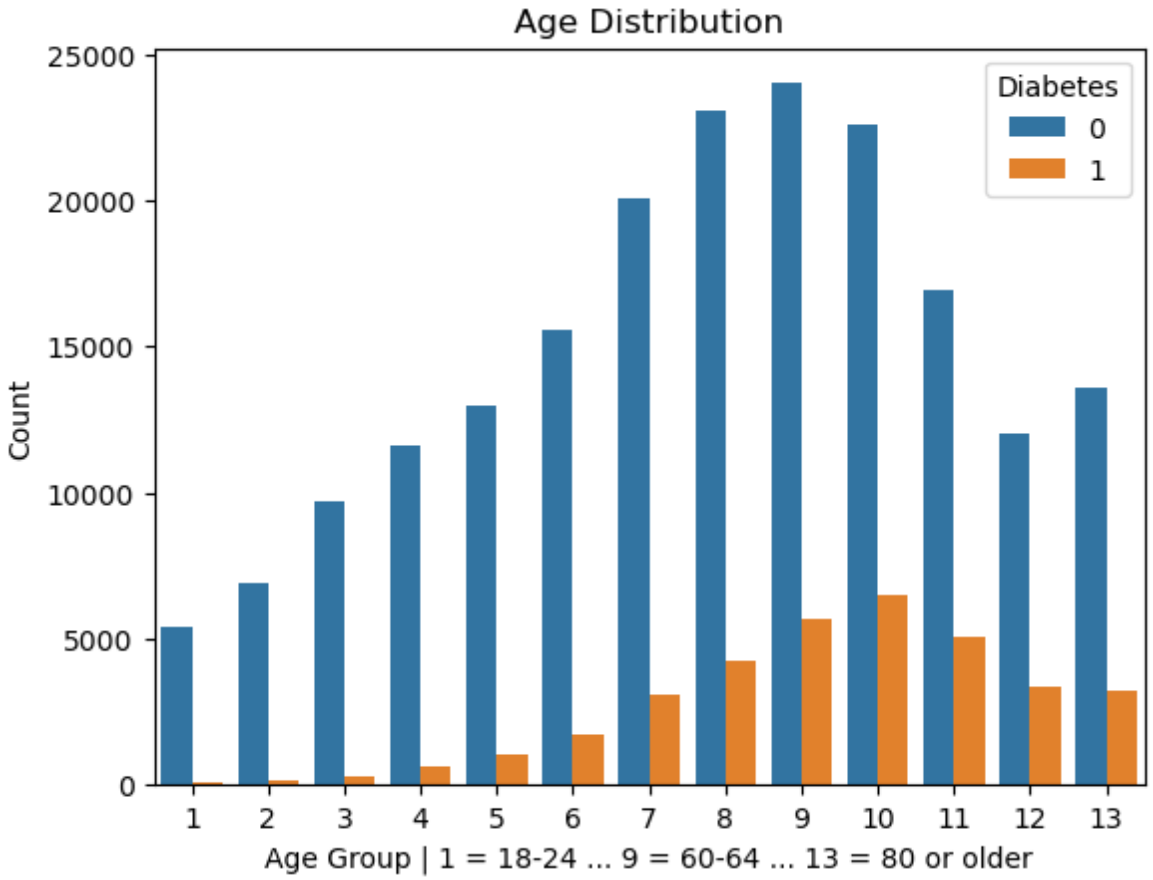
Diabetes Percentage

In [12]:
```
#Plot pie chart to show HeartDisease Percentage
plt.figure(figsize=(10,6))
plt.pie(df['Diabetes'].value_counts(), labels=['No Diabetes', 'Di
plt.title('Diabetes Percentage')
plt.show()
```



Age and Diabetes

In [13]:
```
# Age group distribution
sns.countplot(x='Age', data=df, hue='Diabetes')
plt.title('Age Distribution')
plt.xlabel('Age Group | 1 = 18-24 ... 9 = 60-64 ... 13 = 80 or ol
plt.ylabel('Count')
plt.show()
```

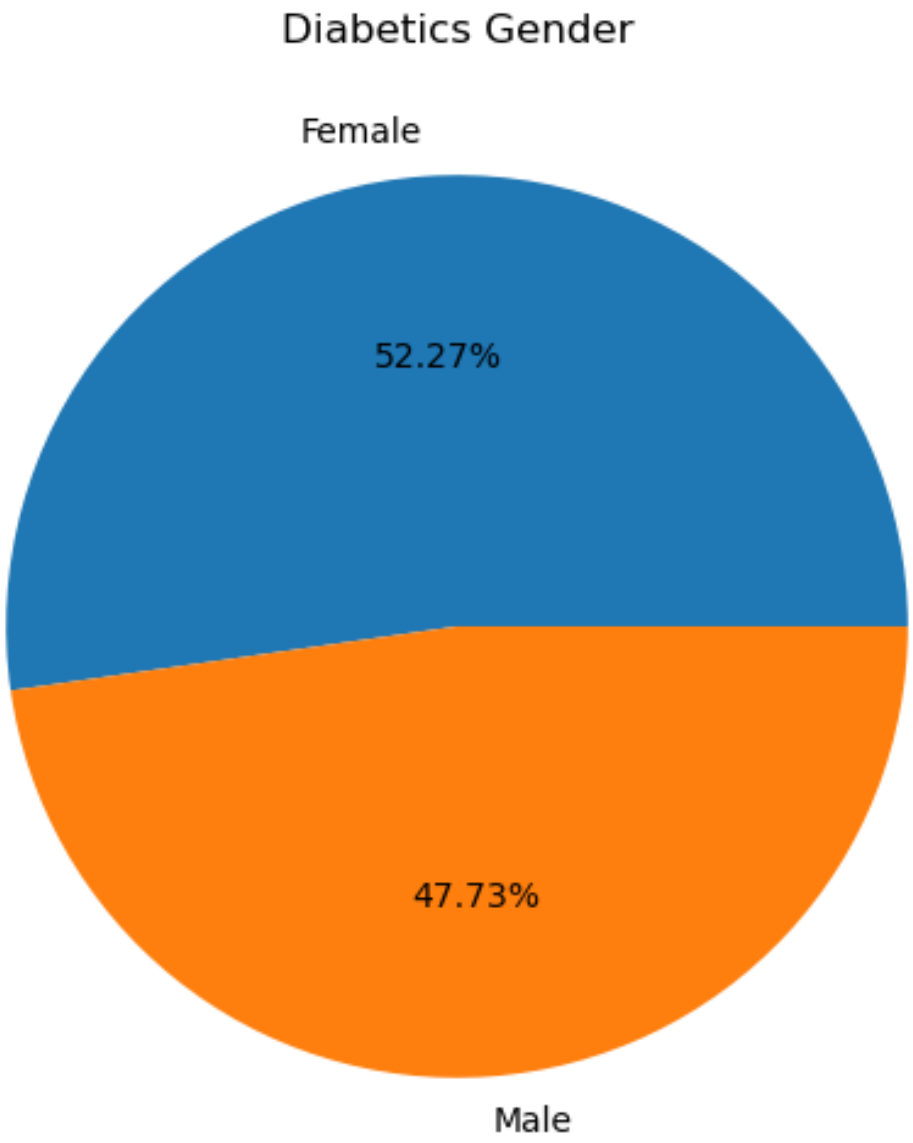


Countplot shows relation between Age and Diabetes.**Elderly people** are at **high risk** of **diabetes**

In [14]:
```
# Split Diabetics
Diabetics = df.where(df.Diabetes == 1)
Diabetics.dropna(inplace=True)
```

Sex and Diabetes

In [15]:
```
# Plot pie chart to show sex distribution of Diabetes pations
plt.figure(figsize=(10,6))
plt.pie(Diabetics['Sex'].value_counts(), labels=['Female','Male']
plt.title('Diabetics Gender')
plt.show()
```

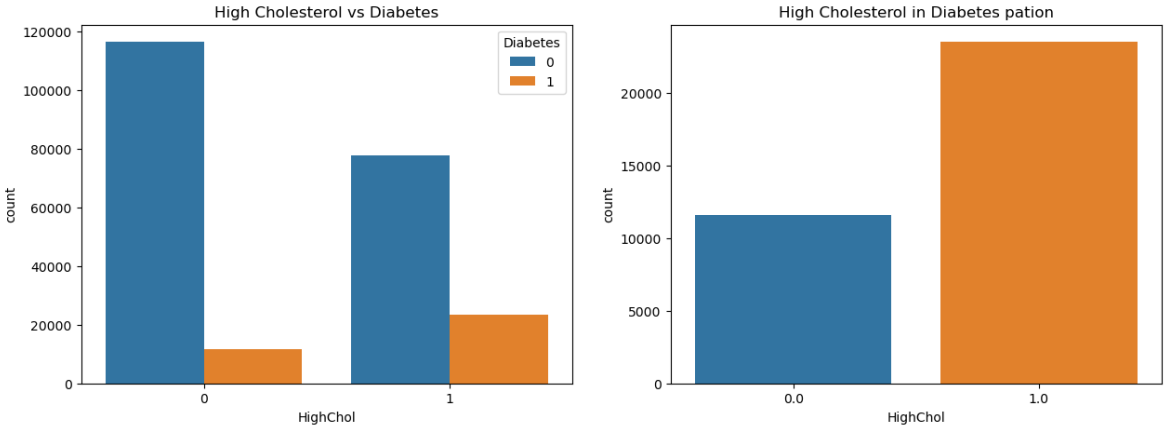


Pie chart shows don't have strong relation between Sex and Diabetes

High Cholesterol and Diabetes

In [16]:
```
# HighChol and Diabetes
fig,ax = plt.subplots(1,2,figsize=(15,5))
sns.countplot(x='HighChol', data=df, hue='Diabetes', ax=ax[0]).se
sns.countplot(x='HighChol', data=Diabetics, ax=ax[1]).set_title('
```

Out[16]:
```
Text(0.5, 1.0, 'High Cholesterol in Diabetes pation')
```

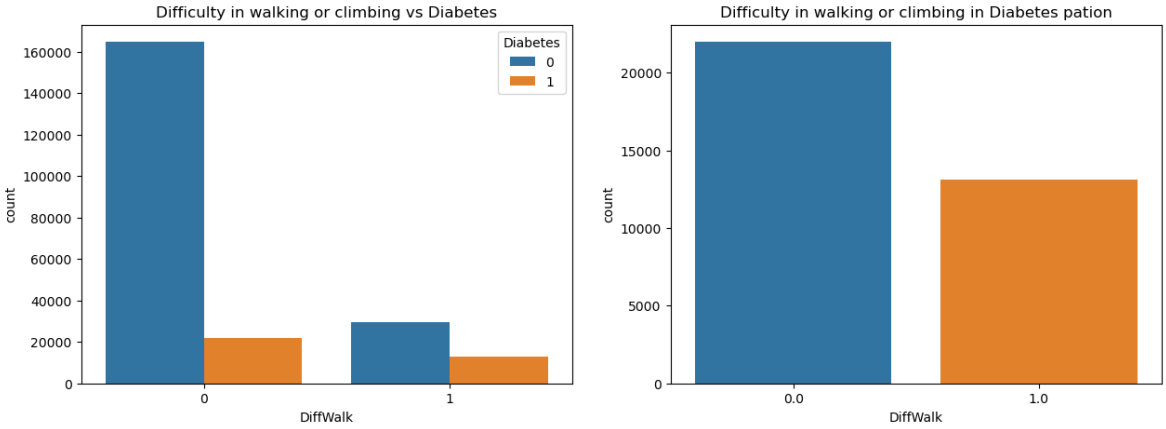


Countplots shows have relation between the High Cholesterol and Diabetes.**Diabetics (with HighChol : 23496 , without HighChol : 11601)**

Difficulty walking or climbing and Diabetes

In [17]:
```
# DiffWalk and Diabetes
fig,ax = plt.subplots(1,2,figsize=(15,5))
sns.countplot(x='DiffWalk', data=df, hue='Diabetes', ax=ax[0]).se
sns.countplot(x='DiffWalk', data=Diabetics, ax=ax[1]).set_title('
```

Out[17]:
```
Text(0.5, 1.0, 'Difficulty in walking or climbing in Diabetes pat
ion')
```



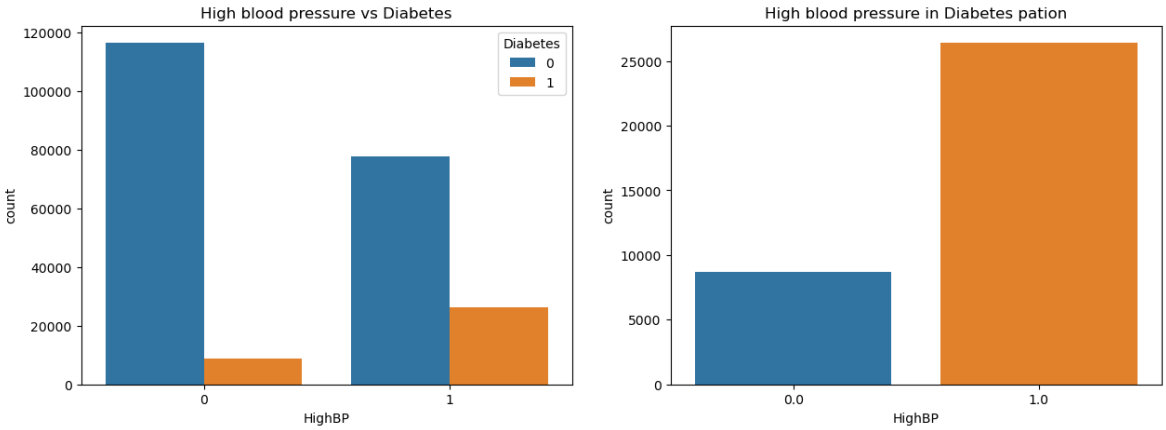
Diabetics (with DiffWalk : 13114 , without DiffWalk : 21983)

High blood pressure and Diabetes

In [18]:
```
# HighBP and Diabetes
fig,ax = plt.subplots(1,2,figsize=(15,5))
```

```
sns.countplot(x='HighBP', data=df, hue='Diabetes', ax=ax[0]).set_
sns.countplot(x='HighBP', data=Diabetics, ax=ax[1]).set_title('Hi
```

Out[18]:
```
Text(0.5, 1.0, 'High blood pressure in Diabetes pation')
```

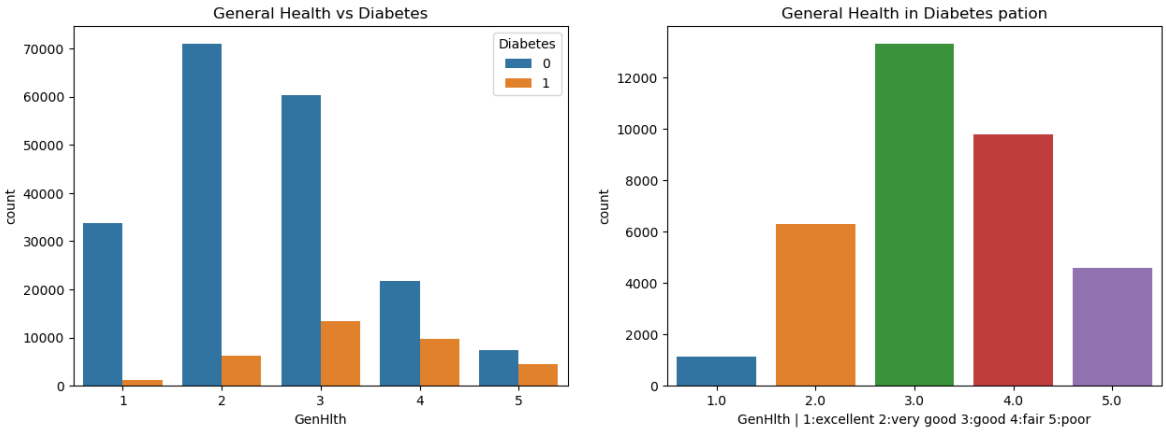


Countplot shows have relation between the High blood pressure and Diabetes. **Diabetics (with HighBP : 26405 , without HighBP : 8692)**

General Health and Diabetes

In [19]:
```
# GenHlth and Diabetes
fig,ax = plt.subplots(1,2,figsize=(15,5))
sns.countplot(x='GenHlth', data=df, hue='Diabetes', ax=ax[0]).set
sns.countplot(x='GenHlth', data=Diabetics, ax=ax[1]).set_title('G
plt.xlabel('GenHlth | 1:excellent 2:very good 3:good 4:fair 5:poo
```

Out[19]:
```
Text(0.5, 0, 'GenHlth | 1:excellent 2:very good 3:good 4:fair 5:p
oor')
```



Both the Countplotsprovides clear understanding of the realtion between the General Health and diabetes. In good, fair, poor health type there is a high risk of developing diabetes

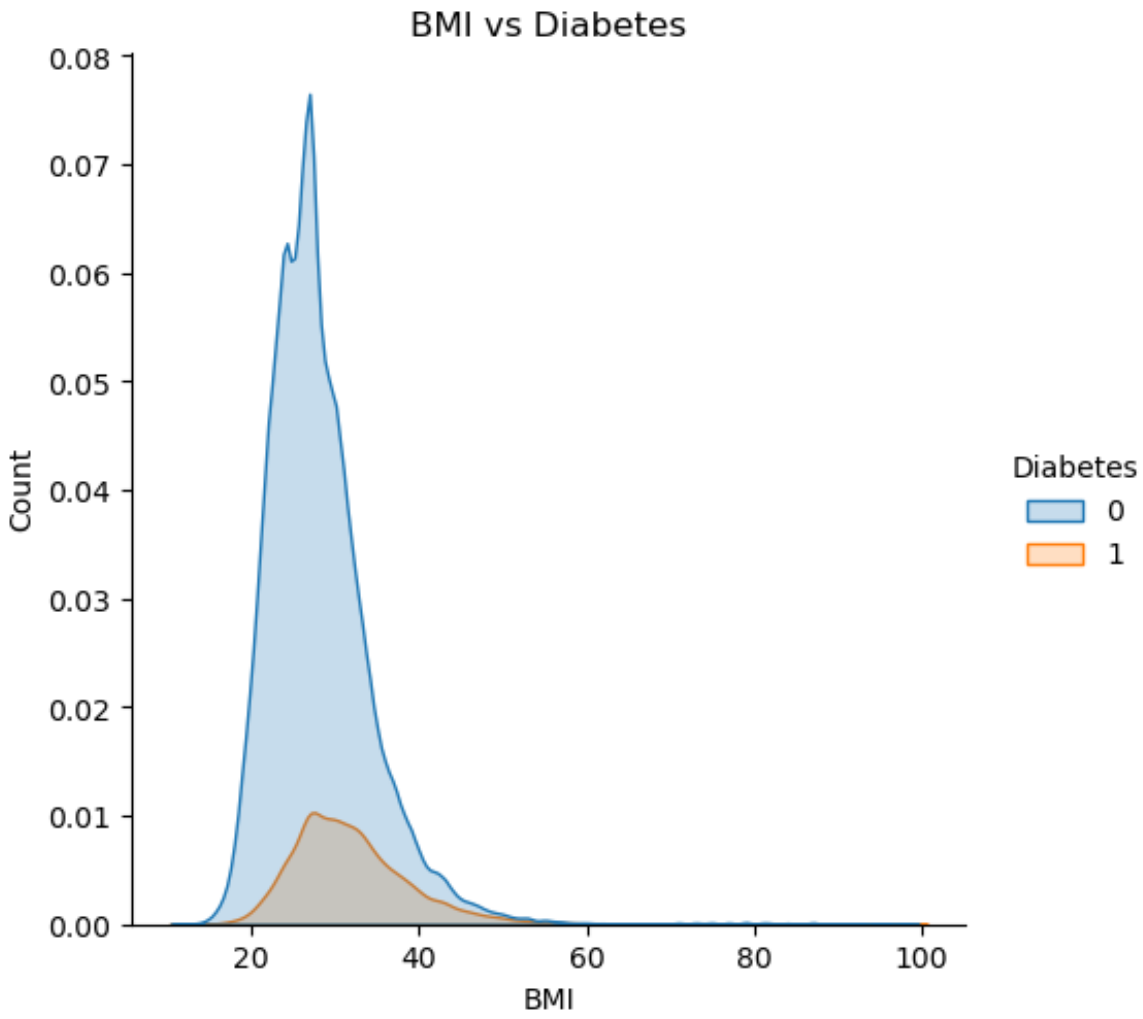
1. good: 13324
2. fair: 9781
3. very good: 6280
4. poor: 4577
5. excellent: 1135

In [20]:
```
Diabetics['GenHlth'].value_counts().sort_values(ascending=False)
```

Out[20]:
```
3.0    13324
4.0     9781
2.0     6280
5.0     4577
1.0     1135
Name: GenHlth, dtype: int64
```

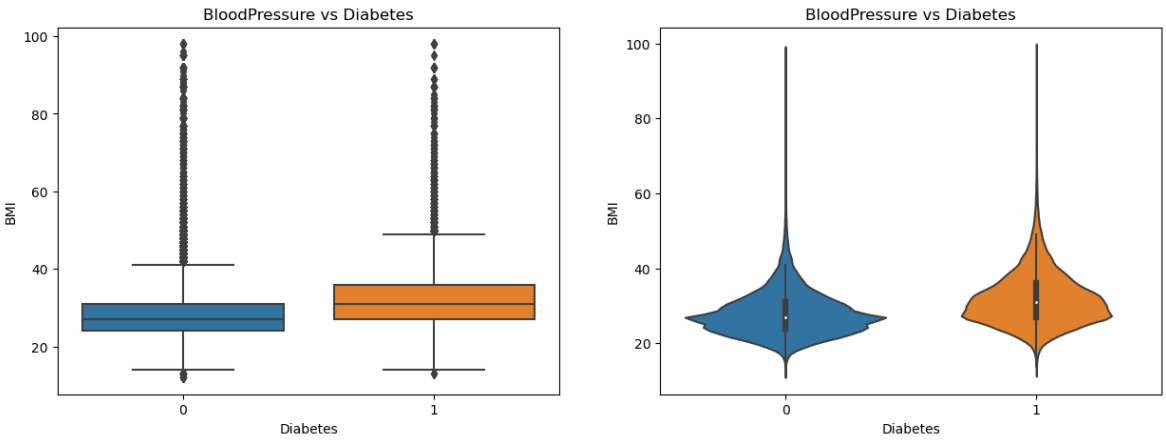
BMI and Diabetes

In [21]:
```
# BMI and Diabetes
sns.displot(df, x="BMI", hue="Diabetes", kind="kde", fill=True)
plt.title('BMI vs Diabetes')
plt.xlabel('BMI')
plt.ylabel('Count')
plt.show()
```



In [22]:
```
fig,ax = plt.subplots(1,2,figsize=(15,5))
sns.boxplot(x='Diabetes', y='BMI', data=df, ax=ax[0]).set_title('
sns.violinplot(x='Diabetes', y='BMI', data=df, ax=ax[1]).set_titl
```

Out[22]:
```
Text(0.5, 1.0, 'BloodPressure vs Diabetes')
```



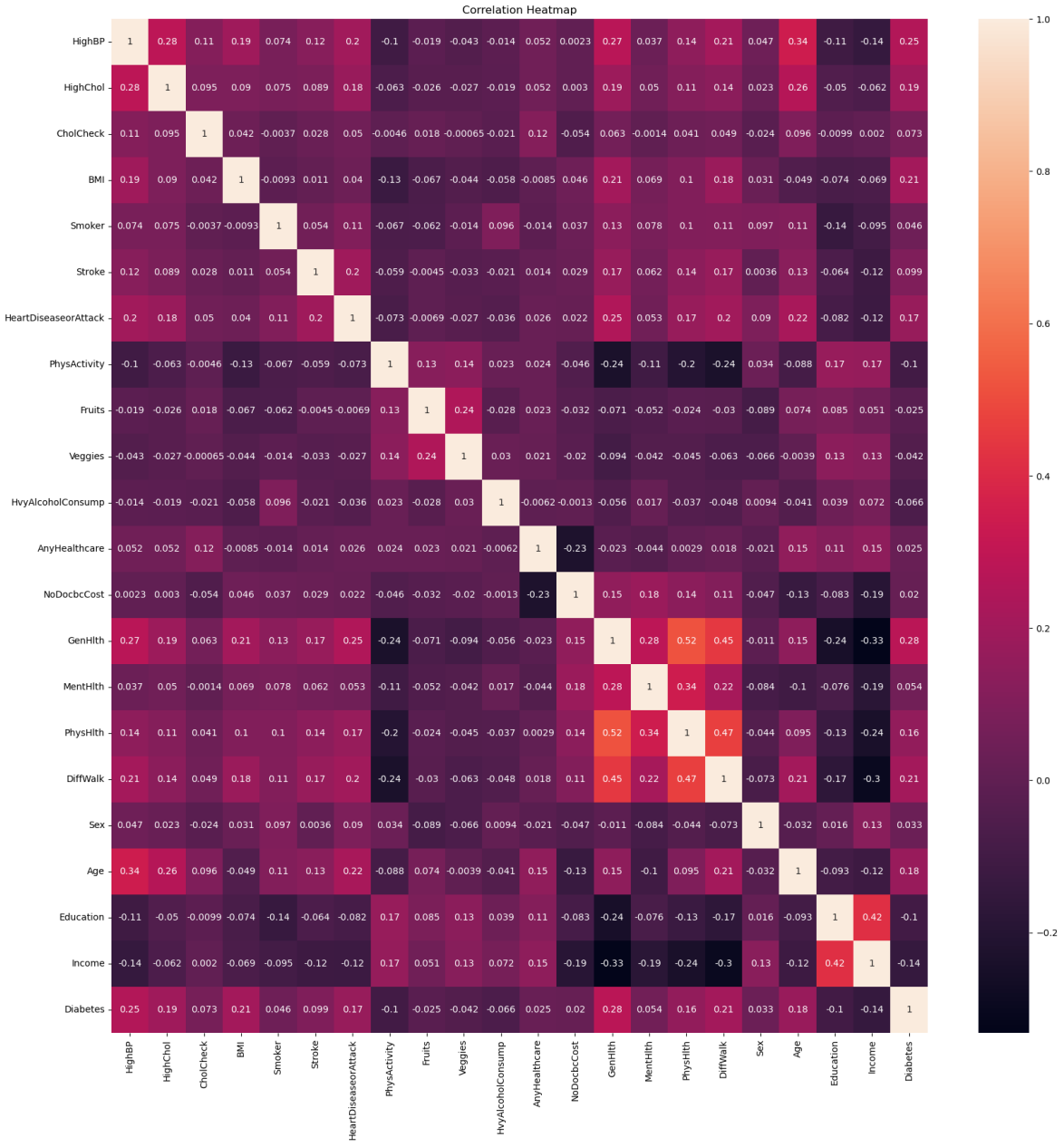
Both graphs highlights the role of BMI in diabetes prediction. Non diabetic patients have a normal BMI within the range of 25-35 whereas

the diabetic patients have a BMI greater .

Correlation heatmap

In [23]:
```
#correlation heatmap
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), annot=True).set_title('Correlation Heatmap
```

Out[23]:
```
Text(0.5, 1.0, 'Correlation Heatmap')
```



Train Test Split

In [24]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df.drop('Diab
```

Model Training

Logistic Regression

In [25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

solver = ['lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', '
best_slover = ''
train_score = np.zeros(6)
for i, n in enumerate(solver):
    lr = LogisticRegression(solver=n).fit(X_train, y_train)
    train_score[i] = lr.score(X_test, y_test)
    if lr.score(X_test, y_test) == train_score.max():
        best_slover = n

lr = LogisticRegression(solver=best_slover)
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
print(f'LogisticRegression Score: {accuracy_score(y_test, lr_pred
```

```
LogisticRegression Score: 0.849373570105676
```

Decision Tree Classifier

In [26]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

dtree = DecisionTreeClassifier(class_weight='balanced')
param_grid = {
    'max_depth': [3, 4, 5, 6, 7, 8],
    'min_samples_split': [2, 3, 4],
    'min_samples_leaf': [1, 2, 3, 4],
    'random_state': [0, 42]
}
grid_search = GridSearchCV(dtree, param_grid, cv=5)
grid_search.fit(X_train, y_train)
Ctree = DecisionTreeClassifier(**grid_search.best_params_, class_
Ctree.fit(X_train, y_train)
```

```
dtc_pred = Ctree.predict(X_test)
print("DecisionTrees's Accuracy: ", accuracy_score(y_test, dtc_pr
```

```
DecisionTrees's Accuracy:  0.7209935722845626
```

Random Forest Classifier

Models Evaluation

Evaluating Logistic Regression Model

In [41]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_erro
print('Accuracy Score :', accuracy_score(y_test, lr_pred))
print('f1 Score :', f1_score(y_test, lr_pred, average="weighted")
print('Mean Absolute Error :',mean_absolute_error(y_test, lr_pred
print('Mean Squared Error : ',mean_squared_error(y_test, lr_pred)
print('R2 Score : ',r2_score(y_test, lr_pred))
```

```
Accuracy Score : 0.849373570105676
f1 Score : 0.8102760048559708
Mean Absolute Error : 0.150626429894324
Mean Squared Error :  0.150626429894324
R2 Score :  -0.1542471114712025
```

Evaluating DecisionTree Model

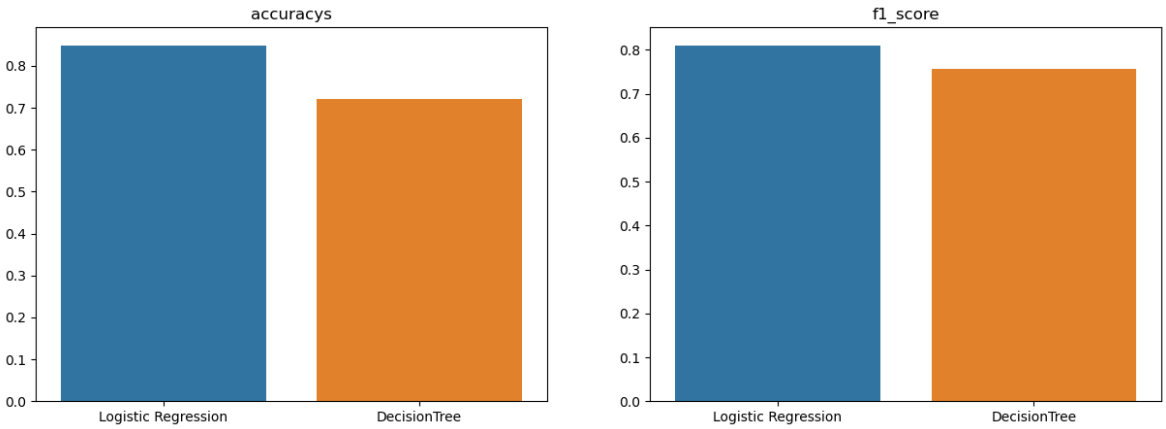
In [42]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_erro
print('Accuracy Score :', accuracy_score(y_test, dtc_pred))
print('f1 Score :', f1_score(y_test, dtc_pred, average="weighted"
print('Mean Absolute Error :',mean_absolute_error(y_test, dtc_pre
print('Mean Squared Error : ',mean_squared_error(y_test, dtc_pred
print('R2 Score : ',r2_score(y_test, dtc_pred))
```

```
Accuracy Score : 0.7209935722845626
f1 Score : 0.7567270217569475
Mean Absolute Error : 0.27900642771543743
Mean Squared Error :  0.27900642771543743
R2 Score :  -1.13802028965554
```

Comparing the models

In [47]:
```
#comparing the accuracy of different models
models = ['Logistic Regression','DecisionTree']
preds = [lr_pred,dtc_pred]
accuracys= []
f1 = []
for i in preds:
    accuracys.append( accuracy_score(y_test, i))
    f1.append(f1_score(y_test, i, average="weighted"))
fig,ax = plt.subplots(1,2,figsize=(15,5))
sns.barplot(x=models, y=accuracys, ax=ax[0]).set_title('accuracys
sns.barplot(x=models, y=f1, ax=ax[1]).set_title('f1_score')
```

Out[47]:
```
Text(0.5, 1.0, 'f1_score')
```



Conclusion

From the exploratory data analysis, I have concluded that the risk of Diabetes depends upon the following factors:

- HighChol
- BMI
- DiffWalk
- HighBP
- GenHlth

The LogisticRegression model performed better than DecisionTrees model with an accuracy of 84.9% .