In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_excel("diabetes.xlsx")
print(df)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6.0    148.0           72.0           35.0      0.0  33.6
1            1.0     85.0           66.0           29.0      0.0  26.6
2            8.0    183.0           64.0            0.0      0.0  23.3
3            1.0     89.0           66.0           23.0     94.0  28.1
4            0.0    137.0           40.0           35.0    168.0  43.1
..           ...      ...            ...            ...      ...   ...
763         10.0    101.0           76.0           48.0    180.0  32.9
764          2.0    122.0           70.0           27.0      0.0  36.8
765          5.0    121.0           72.0           23.0    112.0  26.2
766          1.0    126.0           60.0            0.0      0.0  30.1
767          1.0     93.0           70.0           31.0      0.0  30.4

     DiabetesPedigreeFunction   Age  Outcome
0                       0.627  50.0      1.0
1                       0.351  31.0      0.0
2                       0.672  32.0      1.0
3                       0.167  21.0      0.0
4                       2.288  33.0      1.0
..                        ...   ...      ...
763                     0.171  63.0      0.0
764                     0.340  27.0      0.0
765                     0.245  30.0      0.0
766                     0.349  47.0      1.0
767                     0.315  23.0      0.0

[768 rows x 9 columns]
```

In [ ]:
```
df.plot(kind ="bar")
df.show()
```

In [5]:
```
#to retrive first 5 rows
df.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0
1	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0
2	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0
4	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0

In [6]:
```
#to retrive last 5 rows
df.tail()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63.
764	2.0	122.0	70.0	27.0	0.0	36.8	0.340	27.
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30.
766	1.0	126.0	60.0	0.0	0.0	30.1	0.349	47.
767	1.0	93.0	70.0	31.0	0.0	30.4	0.315	23.

In [16]:
```
#to find no of rows and columns
df.shape
```

Out[16]:
```
(768, 9)
```

In [8]:
```
#to get informattion about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    float64
 1   Glucose                   768 non-null    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    float64
 8   Outcome                   768 non-null    float64
dtypes: float64(9)
memory usage: 54.1 KB
```

In [9]:
```
#to check null values
df.isnull()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	
763	False	False	False	False	False	False	False	Fa
764	False	False	False	False	False	False	False	Fa
765	False	False	False	False	False	False	False	Fa
766	False	False	False	False	False	False	False	Fa
767	False	False	False	False	False	False	False	Fa

768 rows × 9 columns

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [46]:
```
#check for duplicate data
data_dup=df.duplicated().any()
print(data_dup)
#here there are no dublicates no need to drop
```

```
False
```

In [ ]:

In [14]:
```
#get overall statistics about dataset
df.describe()
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

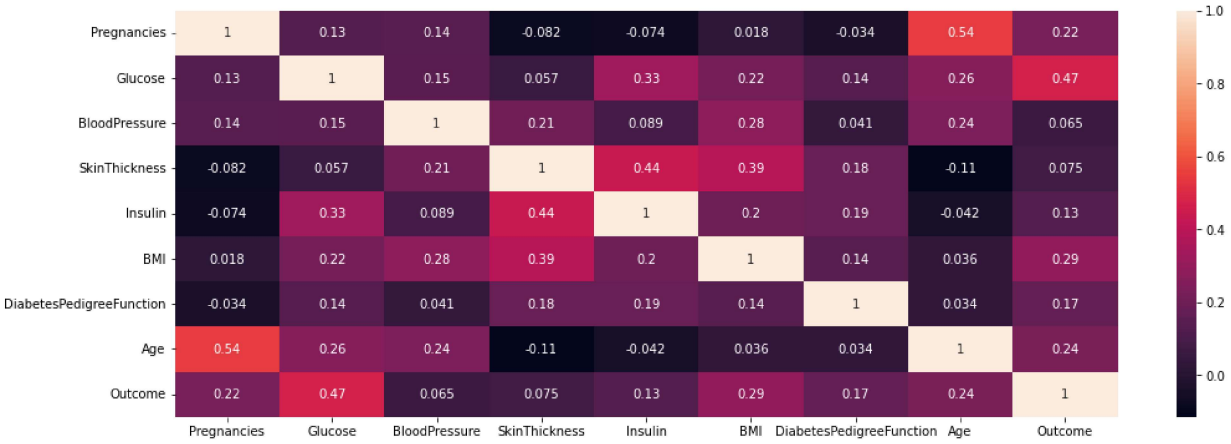
In [17]:
```
#check correaltion b/w varibles in dataset
df.corr()
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	D
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	

In [49]:
```
plt.figure(figsize=(17,6))
sns.heatmap(df.corr(),annot=True)
#here the bright coloured  values are highly correlated to target ,rest are less corre
```

Out[49]:
```
<AxesSubplot:>
```



In [50]:
```
#how many people have disease ,how many don't?
df.columns
```

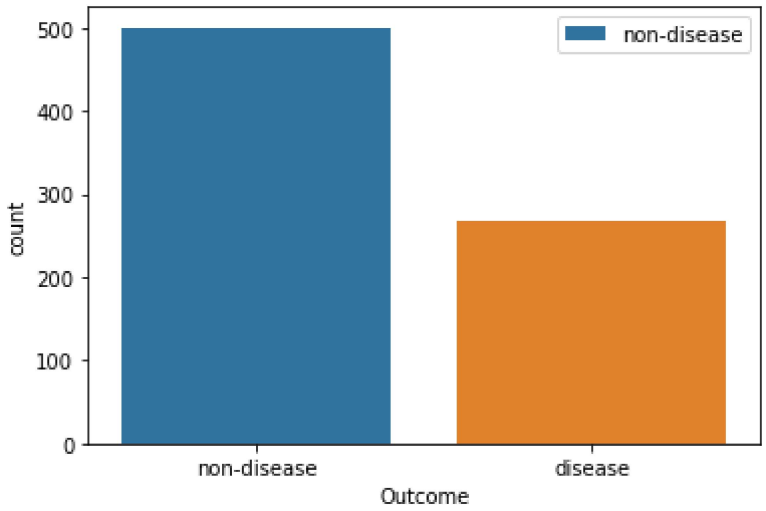
Out[50]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [54]:
```
#here we get count of people with no-disease and disease
df.Outcome.value_counts()
```

Out[54]:
```
0.0    500
1.0    268
Name: Outcome, dtype: int64
```

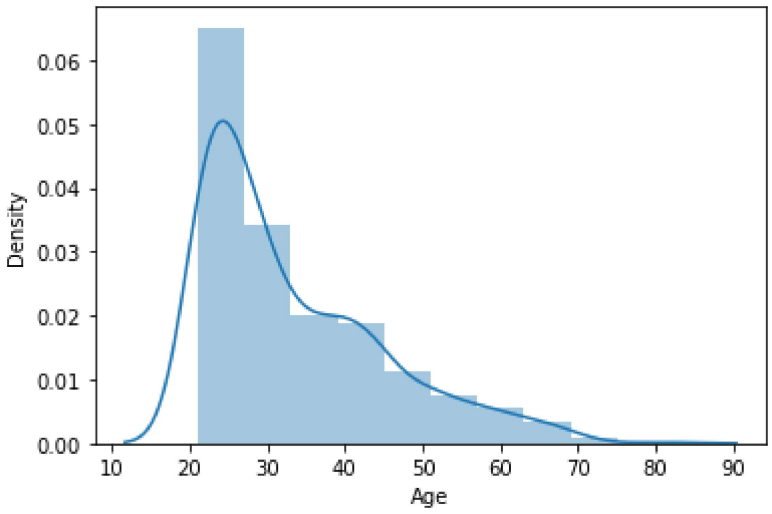
In [68]:
```
sns.countplot (df["Outcome"])
plt.xticks([0,1],["non-disease","disease"])
plt.legend(labels=["non-disease","disease"])
plt.show()
#peolpe with  no diabetics are more than diabetic
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyw
ord will result in an error or misinterpretation.
  warnings.warn(
```



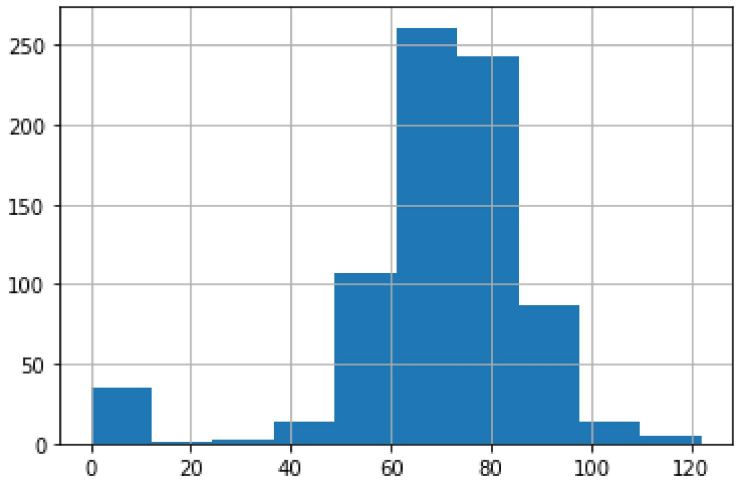
In [71]:
```
sns.distplot(df["Age"],bins=10)
plt.show()
#most of people are having age between 20-30
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Pleas
e adapt your code to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [72]:
```
df["BloodPressure"].hist()
```

Out[72]:
```
<AxesSubplot:>
```

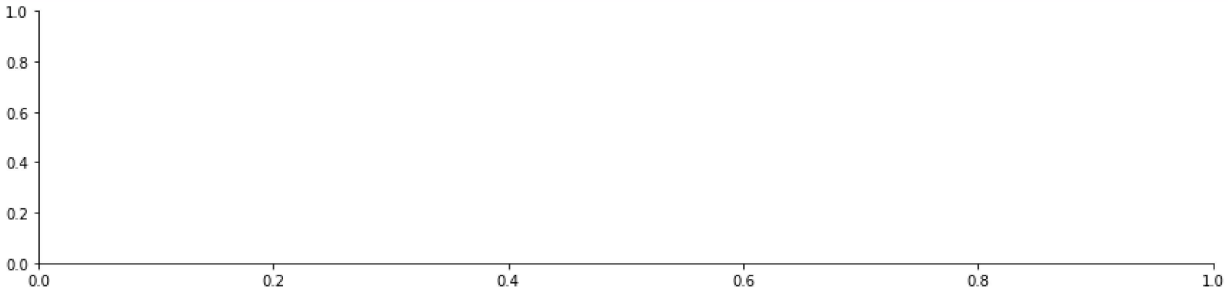


In [81]:
```
g=sns.FacetGrid(df,hue="Outcome",aspect=4)
g.map(sns.kdeplot,"Insulin",shape=True)
plt.legend(label=["non-disease","disease"])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [81], in <cell line: 2>()
      1 g=sns.FacetGrid(df,hue="Outcome",aspect=4)
----> 2 g.map(sns.kdeplot,"Insulin",shape=True)
      3 plt.legend(label=["non-disease","disease"])

File ~\anaconda3\lib\site-packages\seaborn\axisgrid.py:710, in FacetGrid.map(self, fu
nc, *args, **kwargs)
    707         plot_args = [v.values for v in plot_args]
    709     # Draw the plot
--> 710     self._facet_plot(func, ax, plot_args, kwargs)
    712 # Finalize the annotations and layout
    713 self._finalize_grid(args[:2])

File ~\anaconda3\lib\site-packages\seaborn\axisgrid.py:806, in FacetGrid._facet_plot
(self, func, ax, plot_args, plot_kwargs)
    804     plot_args = []
    805     plot_kwargs["ax"] = ax
--> 806 func(*plot_args, **plot_kwargs)
    808 # Sort out the supporting information
    809 self._update_legend_data(ax)

File ~\anaconda3\lib\site-packages\seaborn\_decorators.py:46, in _deprecate_positiona
l_args.<locals>.inner_f(*args, **kwargs)
     36     warnings.warn(
     37         "Pass the following variable{} as {}keyword arg{}: {}. "
     38         "From version 0.12, the only valid positional argument "
   (...)
     43         FutureWarning
     44     )
     45 kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46 return f(**kwargs)

File ~\anaconda3\lib\site-packages\seaborn\distributions.py:1770, in kdeplot(x, y, sh
ade, vertical, kernel, bw, gridsize, cut, clip, legend, cumulative, shade_lowest, cba
r, cbar_ax, cbar_kws, ax, weights, hue, palette, hue_order, hue_norm, multiple, commo
n_norm, common_grid, levels, thresh, bw_method, bw_adjust, log_scale, color, fill, da
ta, data2, warn_singular, **kwargs)
   1767     if color is not None:
   1768         plot_kws["color"] = color
-> 1770     p.plot_univariate_density(
   1771         multiple=multiple,
   1772         common_norm=common_norm,
   1773         common_grid=common_grid,
   1774         fill=fill,
   1775         legend=legend,
   1776         warn_singular=warn_singular,
   1777         estimate_kws=estimate_kws,
   1778         **plot_kws,
   1779     )
   1781 else:
   1783     p.plot_bivariate_density(
   1784         common_norm=common_norm,
   1785         fill=fill,
   (...)
   1795         **kwargs,
   1796     )

File ~\anaconda3\lib\site-packages\seaborn\distributions.py:971, in _DistributionPlot
```

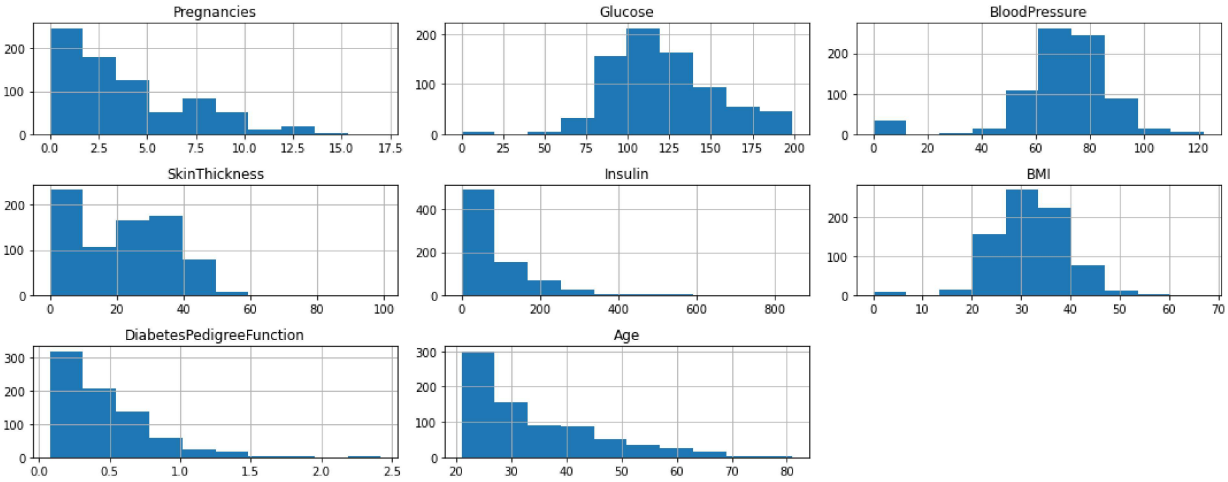
```
ter.plot_univariate_density(self, multiple, common_norm, common_grid, warn_singular,
fill, legend, estimate_kws, **plot_kws)
    969     plot_kws.pop("color", None)
    970 else:
--> 971     scout, = self.ax.plot([], [], **plot_kws)
    972     default_color = scout.get_color()
    973 if scout is not None:

File ~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py:1632, in Axes.plot(self,
scalex, scaley, data, *args, **kwargs)
   1390 """
   1391 Plot y versus x as lines and/or markers.
   1392
   (...)
   1629 (``'green'``) or hex strings (``'#008000'``).
   1630 """
   1631 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1632 lines = [*self._get_lines(*args, data=data, **kwargs)]
   1633 for line in lines:
   1634     self.add_line(line)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:312, in _process_plot_var
_args.__call__(self, data, *args, **kwargs)
    310     this += args[0],
    311     args = args[1:]
--> 312 yield from self._plot_args(this, kwargs)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:538, in _process_plot_var
_args._plot_args(self, tup, kwargs, return_kwargs)
    536     return list(result)
    537 else:
--> 538     return [l[0] for l in result]

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:538, in <listcomp>(.0)
    536     return list(result)
    537 else:
--> 538     return [l[0] for l in result]

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:531, in <genexpr>(.0)
    528 else:
    529     labels = [label] * n_datasets
--> 531 result = (make_artist(x[:, j % ncx], y[:, j % ncy], kw,
    532                       {**kwargs, 'label': label})
    533           for j, label in enumerate(labels))
    535 if return_kwargs:
    536     return list(result)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:351, in _process_plot_var
_args._makeline(self, x, y, kw, kwargs)
    349 default_dict = self._getdefaults(set(), kw)
    350 self._setdefaults(default_dict, kw)
--> 351 seg = mlines.Line2D(x, y, **kw)
    352 return seg, kw

File ~\anaconda3\lib\site-packages\matplotlib\lines.py:393, in Line2D.__init__(self,
xdata, ydata, linewidth, linestyle, color, marker, markersize, markeredgewidth, marke
redgecolor, markerfacecolor, markerfacecoloralt, fillstyle, antialiased, dash_capstyl
e, solid_capstyle, dash_joinstyle, solid_joinstyle, pickradius, drawstyle, markevery,
**kwargs)
    389 self.set_markeredgewidth(markeredgewidth)
```

```
    391 # update kwargs before updating data to give the caller a
    392 # chance to init axes (and hence unit support)
--> 393 self.update(kwargs)
    394 self.pickradius = pickradius
    395 self.ind_offset = 0

File ~\anaconda3\lib\site-packages\matplotlib\artist.py:1064, in Artist.update(self,
props)
   1062             func = getattr(self, f"set_{k}", None)
   1063             if not callable(func):
-> 1064                 raise AttributeError(f"{type(self).__name__!r} object "
   1065                                      f"has no property {k!r}")
   1066             ret.append(func(v))
   1067 if ret:

AttributeError: 'Line2D' object has no property 'shape'
```



In [83]:

```
cat_val=[]
cont_val=[]
for column in df.columns:
    if df[column].nunique()<=10:
        cat_val.append(column)
    else:
        cont_val.append(column)
cat_val
cont_val
df.hist(cont_val,figsize=(15,6))
plt.tight_layout()
plt.show()
```



In [ ]: