3.Working with Pandas Data Frames

Program:

```
#import the pandas library and aliasing as pd
import pandas as pd
#Create empty dataframe
df = pd.DataFrame()
print (df)
#Create dataframe from list of dictionaries
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
#With two column indices, values same as dictionary keys
df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])
#With two column indices with one index with other name
df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])
print (df1)
print (df2)
# Create dataframe from dictionary of Series
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df3= pd.DataFrame(d)
print (df3)
#column selection
print (df3 ['one'])
# Adding a new column to an existing DataFrame object with column label by passing new series
print ("Adding a new column by passing as Series:")
df3['three']=pd.Series([10,20,30],index=['a','b','c'])
print (df3)
print ("Adding a new column using the existing columns in DataFrame:")
```

```
df3['four']=df3['one']+df3['three']
print (df3)
# Using the previous DataFrame delete a column
# using del function
print ("Deleting the first column using DEL function:")
del df3['one']
print (df1)
# using pop function
print ("Deleting another column using POP function:")
df3.pop('two')
print (df3)
#Row selection by lable
print (df3.loc['b'])
#row selection by integer location
print (df3.iloc[2])
#slice rows
print (df3[2:4])
#Addition of rows
df4 = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df5 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])
df4 = df.append(df5)
print (df4)
# Drop rows with label 0
df4 = df4.drop(0)
print (df4)
```

Output:

Empty DataFrame

Columns: []

Index: []

a b

first 1 2

second 5 10

a b1

first 1 NaN

second 5 NaN

one two

a 1.0 1

b 2.0 2

c 3.0 3

d NaN 4

a 1.0

b 2.0

c 3.0

d NaN

Name: one, dtype: float64

Adding a new column by passing as Series:

one two three

a 1.0 1 10.0

b 2.0 2 20.0

c 3.0 3 30.0

d NaN 4 NaN

Adding a new column using the existing columns in DataFrame:

```
   one  two  three  four
a  1.0    1   10.0  11.0
b  2.0    2   20.0  22.0
c  3.0    3   30.0  33.0
d  NaN    4    NaN   NaN
```

Deleting the first column using DEL function:

```
       a   b
first  1   2
second 5  10
```

Deleting another column using POP function:

```
   three  four
a   10.0  11.0
b   20.0  22.0
c   30.0  33.0
d    NaN   NaN
three    20.0
four     22.0
Name: b, dtype: float64
three    30.0
four     33.0
Name: c, dtype: float64
   three  four
c   30.0  33.0
d    NaN   NaN
   a  b
```

4.Reading data from text files, Excel and the web and exploring various commands for doing descriptive analytics on the Iris data set.

Program:

```
import pandas as pd
data = pd.read_csv('iris.csv')
print(data)
data.head()
data.sample(10)
data.columns
data.shape
print(data)
print(data[10:21])
sliced_data=data[10:21]
print(sliced_data)
specific_data=data[["Id","Species"]]
print(specific_data.head(10))
data.iloc[5]
data.loc[data["Species"] == "Iris-setosa"]
data["Species"].value_counts()
sum_data = data["sepallength"].sum()
mean_data = data["sepallength"].mean()
median_data = data["sepallength"].median()
print("Sum:",sum_data, "\nMean:", mean_data, "\nMedian:",median_data)
min_data=data["sepallength"].min()
max_data=data["sepallength"].max()
print("Minimum:",min_data, "\nMaximum:", max_data)
cols = data.columns
print(cols)
```

```
cols = cols[1:5]
data1 = data[cols]
data["total_values"]=data1[cols].sum(axis=1)
newcols={"Id":"id","sepallength":"sepallength","sepalwidth":"sepalwidth"}
data.rename(columns=newcols,inplace=True)
print(data.head())
data.style
data.head(10).style.highlight_max(color='lightgreen', axis=0)
data.head(10).style.highlight_max(color='lightgreen', axis=1)
data.head(10).style.highlight_max(color='lightgreen', axis=None)
data.isnull()
data.isnull().sum()
import seaborn as sns
iris = sns.load_dataset("iris")
sns.heatmap(iris.corr(), linecolor = 'white', linewidths = 1)
sns.heatmap(iris.corr(), linecolor = 'white', linewidths = 1, annot = True )
data.corr(method='pearson')
g = sns.pairplot(data,hue="Species")
```

Output:

	Id	sepallength	sepalwidth	petallength	petalwidth	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
..	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica

146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 6 columns]

	Id	sepallength	sepalwidth	petallength	petalwidth	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
..	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 6 columns]

	Id	sepallength	sepalwidth	petallength	petalwidth	Species
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa

17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa
20	21	5.4	3.4	1.7	0.2	Iris-setosa

	Id	sepallength	sepalwidth	petallength	petalwidth	Species
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa
20	21	5.4	3.4	1.7	0.2	Iris-setosa

	Id	Species
0	1	Iris-setosa
1	2	Iris-setosa
2	3	Iris-setosa
3	4	Iris-setosa
4	5	Iris-setosa
5	6	Iris-setosa
6	7	Iris-setosa
7	8	Iris-setosa
8	9	Iris-setosa
9	10	Iris-setosa

Sum: 876.5

Mean: 5.843333333333335

Median: 5.8

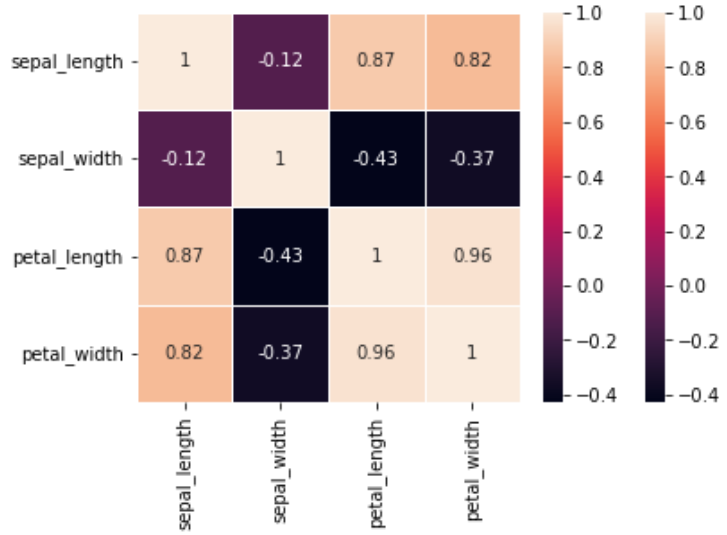
Minimum: 4.3

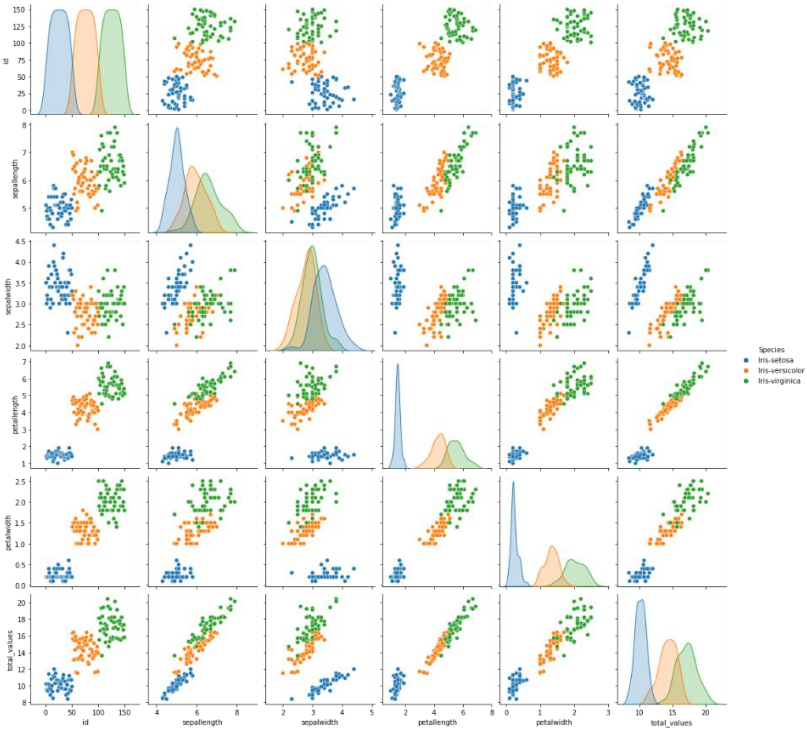
Maximum: 7.9

```
Index(['Id', 'sepallength', 'sepalwidth', 'petallength', 'petalwidth',
       'Species'],
      dtype='object')
```

	id	sepallength	sepalwidth	petallength	petalwidth	Species	\
0	1	5.1	3.5	1.4	0.2	Iris-setosa	
1	2	4.9	3.0	1.4	0.2	Iris-setosa	
2	3	4.7	3.2	1.3	0.2	Iris-setosa	
3	4	4.6	3.1	1.5	0.2	Iris-setosa	
4	5	5.0	3.6	1.4	0.2	Iris-setosa	

	total_values
0	10.2
1	9.5
2	9.4
3	9.4
4	10.2


sepal_length
sepal_width
petal_length
petal_width
1
-0.12
0.87
0.82
-0.12
1
-0.43
-0.37
0.87
-0.43
1
0.96
0.82
-0.37
0.96
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


id
sepallength
sepalwidth
petallength
petalwidth
total_values
Species
Iris-setosa
Iris-versicolor
Iris-virginica

5.1 Standard Deviation, Skewness and Kurtosis of Pima Indians Diabetes Dataset.

Program:

```
import pandas as pd
import statistics
pima = pd.read_csv('diabetes.csv')
print(pima.head())
print(pima.shape)
print(type(pima))
pima_row_idx = pima.index
print(pima_row_idx)
pima_col_idx = pima.columns
print(pima_col_idx)
print(pima.dtypes)
mean=statistics.mean(pima["Insulin"])
mode=statistics.mode(pima["Insulin"])
median=statistics.median(pima["Insulin"])
variance=statistics.variance(pima["Outcome"])
standard_deviation=statistics.stdev(pima["Outcome"])
fre_count=pima["Outcome"].value_counts()
skew=pima.skew(axis=0,skipna=True)
kurt=pima.kurtosis(skipna=True)
print(mean,"\n",mode,"\n",median,"\n",variance,"\n",standard_deviation,"\n",fre_count,"\n",skew
,"\n",kurt)
```

Output:

```
   Pregnancies  Glucose  BloodPressure  ...  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72  ...                     0.627   50        1
1            1       85             66  ...                     0.351   31        0
```

```
2            8      183            64  ...                    0.672   32        1
3            1       89            66  ...                    0.167   21        0
4            0      137            40  ...                    2.288   33        1

[5 rows x 9 columns]
(768, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex(start=0, stop=768, step=1)
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction    float64
Age                         int64
Outcome                     int64
dtype: object
79.79947916666667
 0
 30.5
 0.22748261625380273
 0.47695137724279896
 0    500
1    268
```

Name: Outcome, dtype: int64

```
 Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     2.272251
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
 Pregnancies                 0.159220
Glucose                     0.640780
BloodPressure               5.180157
SkinThickness              -0.520072
Insulin                     7.214260
BMI                         3.290443
DiabetesPedigreeFunction    5.594954
Age                         0.643159
Outcome                    -1.600930
dtype: float64
```

5.2 Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis of UCI Diabetes Dataset.

Program:

```
import pandas as pd
import statistics
uci = pd.read_csv('diabetic_data.csv')
print(uci.head())
print(uci.shape)
print(type(uci))
uci_row_idx = uci.index
print(uci_row_idx)
uci_col_idx = uci.columns
print(uci_col_idx)
print(uci.dtypes)
mean=statistics.mean(uci["num_lab_procedures"])
mode=statistics.mode(uci["num_lab_procedures"])
median=statistics.median(uci["num_lab_procedures"])
variance=statistics.variance(uci["num_lab_procedures"])
standard_deviation=statistics.stdev(uci["num_lab_procedures"])
fre_count=uci["num_lab_procedures"].value_counts()
skew=uci.skew(axis=0,skipna=True)
kurt=uci.kurtosis(skipna=True)
print(mean,"\n",mode,"\n",median,"\n",variance,"\n",standard_deviation,"\n",fre_count,"\n",skew
,"\n",kurt)
```

Output:

```
encounter_id  patient_nbr           race  ... change diabetesMed readmitted
0       2278392      8222157        Caucasian  ...     No         No        NO
1        149190     55629189        Caucasian  ...     Ch        Yes       >30
2         64410     86047875  AfricanAmerican  ...     No        Yes        NO
3        500364     82442376        Caucasian  ...     Ch        Yes        NO
4         16680     42519267        Caucasian  ...     Ch        Yes        NO

[5 rows x 50 columns]
(101766, 50)
<class 'pandas.core.frame.DataFrame'>
RangeIndex(start=0, stop=101766, step=1)
Index(['encounter_id', 'patient_nbr', 'race', 'gender', 'age', 'weight',
       'admission_type_id', 'discharge_disposition_id', 'admission_source_id',
       'time_in_hospital', 'payer_code', 'medical_specialty',
       'num_lab_procedures', 'num_procedures', 'num_medications',
       'number_outpatient', 'number_emergency', 'number_inpatient', 'diag_1',
       'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cresult',
       'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
       'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
       'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',
       'tolazamide', 'examide', 'citoglipton', 'insulin',
       'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-rosiglitazone',
       'metformin-pioglitazone', 'change', 'diabetesMed', 'readmitted'],
      dtype='object')
encounter_id              int64
patient_nbr               int64
```

race	object
gender	object
age	object
weight	object
admission_type_id	int64
discharge_disposition_id	int64
admission_source_id	int64
time_in_hospital	int64
payer_code	object
medical_specialty	object
num_lab_procedures	int64
num_procedures	int64
num_medications	int64
number_outpatient	int64
number_emergency	int64
number_inpatient	int64
diag_1	object
diag_2	object
diag_3	object
number_diagnoses	int64
max_glu_serum	object
A1Cresult	object
metformin	object
repaglinide	object
nateglinide	object
chlorpropamide	object
glimepiride	object
acetohexamide	object

glipizide object
glyburide object
tolbutamide object
pioglitazone object
rosiglitazone object
acarbose object
miglitol object
troglitazone object
tolazamide object
examide object
citoglipton object
insulin object
glyburide-metformin object
glipizide-metformin object
glimepiride-pioglitazone object
metformin-rosiglitazone object
metformin-pioglitazone object
change object
diabetesMed object
readmitted object
dtype: object
43.09564098028811
1
44.0
387.0805299104688
19.674362249142124
1 3208
43 2804

44 2496

45 2376

38 2213

120 1

132 1

121 1

126 1

118 1

Name: num_lab_procedures, Length: 118, dtype: int64

encounter_id 0.699142

patient_nbr 0.471281

admission_type_id 1.591984

discharge_disposition_id 2.563067

admission_source_id 1.029935

time_in_hospital 1.133999

num_lab_procedures -0.236544

num_procedures 1.316415

num_medications 1.326672

number_outpatient 8.832959

number_emergency 22.855582

number_inpatient 3.614139

number_diagnoses -0.876746

dtype: float64

encounter_id -0.102071

patient_nbr -0.347372

admission_type_id 1.942476

discharge_disposition_id 6.003347

admission_source_id	1.744989
time_in_hospital	0.850251
num_lab_procedures	-0.245074
num_procedures	0.857110
num_medications	3.468155
number_outpatient	147.907736
number_emergency	1191.686726
number_inpatient	20.719397
number_diagnoses	-0.079056

dtype: float64

5.3 Bivariate Analysis-Program for linear regression:

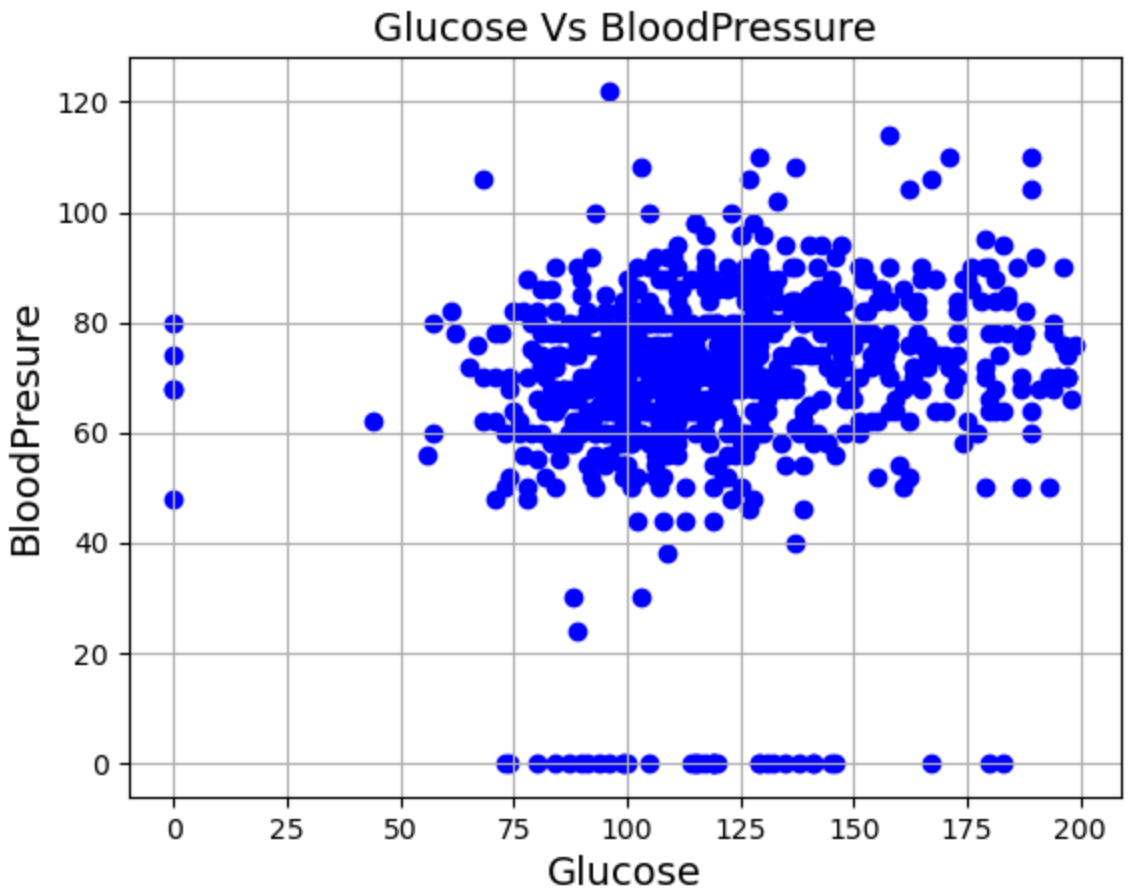
Program:

```
import numpy as np
import pandas as pd
import seaborn as sns
import statistics
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
df = pd.read_csv('diabetes.csv')
head=df.head()
print(head)
cols=["Pregnancies", "Glucose","BloodPressure","SkinThickness","Insulin", "BMI","DiabetesPedigreeFunction", "Age"]
X=df.BMI
Y=df.Age
plt.scatter(df['Glucose'], df['BloodPressure'], color='blue')
plt.title('Glucose Vs BloodPressure', fontsize=14)
plt.xlabel('Glucose', fontsize=14)
plt.ylabel('BloodPresure', fontsize=14)
plt.grid(True)
plt.show()
model = LinearRegression()
model = LinearRegression().fit(X,Y)
```

```
   Pregnancies  Glucose  BloodPressure  ...  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72  ...                     0.627   50        1
1            1       85             66  ...                     0.351   31        0
2            8      183             64  ...                     0.672   32        1
```

3	1	89	66	...	0.167	21	0
4	0	137	40	...	2.288	33	1

[5 rows x 9 columns]



5.4 Bivariate Analysis:Logistic regression

Program:

```
 import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
## Importing stats models for running logistic regression
import statsmodels.api as sm
sns.set(color_codes =True)
%matplotlib inline
df = pd.read_csv('diabetes.csv')
head=df.head()
print(head)
cols=["Pregnancies", "Glucose","BloodPressure","SkinThickness","Insulin",
"BMI","DiabetesPedigreeFunction", "Age"]
X=df[cols]
y=df.Outcome
## Defining the model and assigning Y (Dependent) and X (Independent Variables)
logit_model=sm.Logit(y,X)
## Fitting the model and publishing the results
result=logit_model.fit()
print(result.summary())
cols2=["Pregnancies", "Glucose","BloodPressure","SkinThickness","BMI"]
X=df[cols2]
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
cols3=["Pregnancies", "Glucose","BloodPressure","SkinThickness"]
X=df[cols3]
```

```
logit_model=sm.Logit(y,X)

result=logit_model.fit()

print(result.summary())

cols4=["Pregnancies", "Glucose","BloodPressure"]

X=df[cols4]

logit_model=sm.Logit(y,X)

result=logit_model.fit()

print(result.summary())

## Importing LogisticRegression from Sk.Learn linear model as stats model function cannot give us classification report and confusion matrix

from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

cols4=["Pregnancies", "Glucose","BloodPressure"]

X=df[cols4]

y=df.Outcome

logreg.fit(X,y)

## Defining the y_pred variable for the predicting values. I have taken 392 dia dataset. We can also take a test dataset

y_pred=logreg.predict(X)

## Calculating the precision of the model

from sklearn.metrics import classification_report

print(classification_report(y,y_pred))

from sklearn.metrics import confusion_matrix

## Confusion matrix gives the number of cases where the model is able to accurately predict the outcomes.. both 1 and 0 and how many cases it gives false positive and false negatives

confusion_matrix = confusion_matrix(y, y_pred)

print(confusion_matrix
```

Output:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1

Optimization terminated successfully.

Current function value: 0.608498

Iterations 5

Logit Regression Results

Dep. Variable:	Outcome	No. Observations:	768
Model:	Logit	Df Residuals:	760
Method:	MLE	Df Model:	7
Date:	Wed, 12 Oct 2022	Pseudo R-squ.:	0.05922
Time:	02:32:14	Log-Likelihood:	-467.33
converged:	True	LL-Null:	-496.74
Covariance Type:	nonrobust	LLR p-value:	2.583e-10

	coef	std err	z	P>|z|	[0.025	0.975]

```
---------------------------------------------------------------------------------
Pregnancies                0.1284      0.029      4.484      0.000       0.072       0.185
Glucose                    0.0129      0.003      4.757      0.000       0.008       0.018
BloodPressure             -0.0303      0.005     -6.481      0.000      -0.039      -0.021
SkinThickness              0.0002      0.006      0.032      0.974      -0.012       0.012
Insulin                    0.0007      0.001      0.942      0.346      -0.001       0.002
BMI                       -0.0048      0.011     -0.449      0.653      -0.026       0.016
DiabetesPedigreeFunction   0.3203      0.240      1.335      0.182      -0.150       0.790
Age                       -0.0156      0.008     -1.852      0.064      -0.032       0.001
=================================================================================
Optimization terminated successfully.
         Current function value: 0.612572
         Iterations 5
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.053
Dependent Variable: Outcome          AIC:              950.9107
Date:               2022-10-12 02:32 BIC:              974.1297
No. Observations:   768              Log-Likelihood:   -470.46
Df Model:           4                LL-Null:          -496.74
Df Residuals:       763              LLR p-value:      1.0467e-10
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
-----------------------------------------------------------------
                  Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
-----------------------------------------------------------------
Pregnancies       0.0973   0.0240  4.0565 0.0000  0.0503  0.1444
Glucose           0.0125   0.0024  5.2723 0.0000  0.0079  0.0172
```

BloodPressure -0.0331 0.0046 -7.2700 0.0000 -0.0421 -0.0242

SkinThickness 0.0048 0.0054 0.8871 0.3750 -0.0058 0.0154

BMI -0.0057 0.0106 -0.5365 0.5916 -0.0265 0.0151

==========

Optimization terminated successfully.

Current function value: 0.612759

Iterations 5

Logit Regression Results

==========

Dep. Variable: Outcome No. Observations: 768

Model: Logit Df Residuals: 764

Method: MLE Df Model: 3

Date: Wed, 12 Oct 2022 Pseudo R-squ.: 0.05263

Time: 02:32:14 Log-Likelihood: -470.60

converged: True LL-Null: -496.74

Covariance Type: nonrobust LLR p-value: 2.602e-11

==========

coef std err z P>|z| [0.025 0.975]

Pregnancies 0.0973 0.024 4.051 0.000 0.050 0.144

Glucose 0.0120 0.002 5.661 0.000 0.008 0.016

BloodPressure -0.0343 0.004 -8.536 0.000 -0.042 -0.026

SkinThickness 0.0037 0.005 0.742 0.458 -0.006 0.014

==========

Optimization terminated successfully.

Current function value: 0.613118

Iterations 5

Logit Regression Results

Dep. Variable:	Outcome	No. Observations:	768
Model:	Logit	Df Residuals:	765
Method:	MLE	Df Model:	2
Date:	Wed, 12 Oct 2022	Pseudo R-squ.:	0.05207
Time:	02:32:14	Log-Likelihood:	-470.87
converged:	True	LL-Null:	-496.74
Covariance Type:	nonrobust	LLR p-value:	5.835e-12

	coef	std err	z	P>|z|	[0.025	0.975]
Pregnancies	0.0951	0.024	3.996	0.000	0.048	0.142
Glucose	0.0122	0.002	5.804	0.000	0.008	0.016
BloodPressure	-0.0335	0.004	-8.692	0.000	-0.041	-0.026

	precision	recall	f1-score	support
0	0.77	0.88	0.82	500
1	0.69	0.51	0.59	268
accuracy			0.75	768
macro avg	0.73	0.69	0.70	768
weighted avg	0.74	0.75	0.74	768

5.5 MULTIPLE REGRESSION ANALYSIS

```
import pandas as pd

import seaborn as sns

from pandas.plotting import scatter_matrix

import matplotlib.pyplot as plt

import statsmodels.api as sm

import pylab

df=pd.read_csv('diabetes.csv')

df.head()

print(df)

corr=df.corr()

print(corr)

hm=sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,cmap='RdBu',annot=T
rue)

print(hm)

data=df.Age

sm.qqplot(data,line='s')

pylab.show()

scatter_matrix(df)

#pd.plotting.scatter_matrix(df,alpha=1,figsize(20,20))

plt.show()

data=df[["Pregnancies","Glucose","BloodPressure"]]

print(data)

#pd.plotting.scatter_matrix(data,alpha=1,figsize(30,20))

plt.show()
```

Output:

```
     Pregnancies  Glucose  ...  Age  Outcome
0              6      148  ...   50        1
1              1       85  ...   31        0
2              8      183  ...   32        1
3              1       89  ...   21        0
4              0      137  ...   33        1
..           ...      ...  ...  ...      ...
763           10      101  ...   63        0
764            2      122  ...   27        0
765            5      121  ...   30        0
766            1      126  ...   47        1
767            1       93  ...   23        0

[768 rows x 9 columns]
                          Pregnancies   Glucose  ...       Age   Outcome
Pregnancies                  1.000000  0.129459  ...  0.544341  0.221898
Glucose                      0.129459  1.000000  ...  0.263514  0.466581
BloodPressure                0.141282  0.152590  ...  0.239528  0.065068
SkinThickness               -0.081672  0.057328  ... -0.113970  0.074752
Insulin                     -0.073535  0.331357  ... -0.042163  0.130548
BMI                          0.017683  0.221071  ...  0.036242  0.292695
DiabetesPedigreeFunction    -0.033523  0.137337  ...  0.033561  0.173844
Age                          0.544341  0.263514  ...  1.000000  0.238356
Outcome                      0.221898  0.466581  ...  0.238356  1.000000
```

[9 rows x 9 columns]

AxesSubplot(0.125,0.125;0.62x0.755)

	Pregnancies	Glucose	BloodPressure
0	6	148	72
1	1	85	66
2	8	183	64
3	1	89	66
4	0	137	40
..	...	...	...
763	10	101	76
764	2	122	70
765	5	121	72
766	1	126	60
767	1	93	70

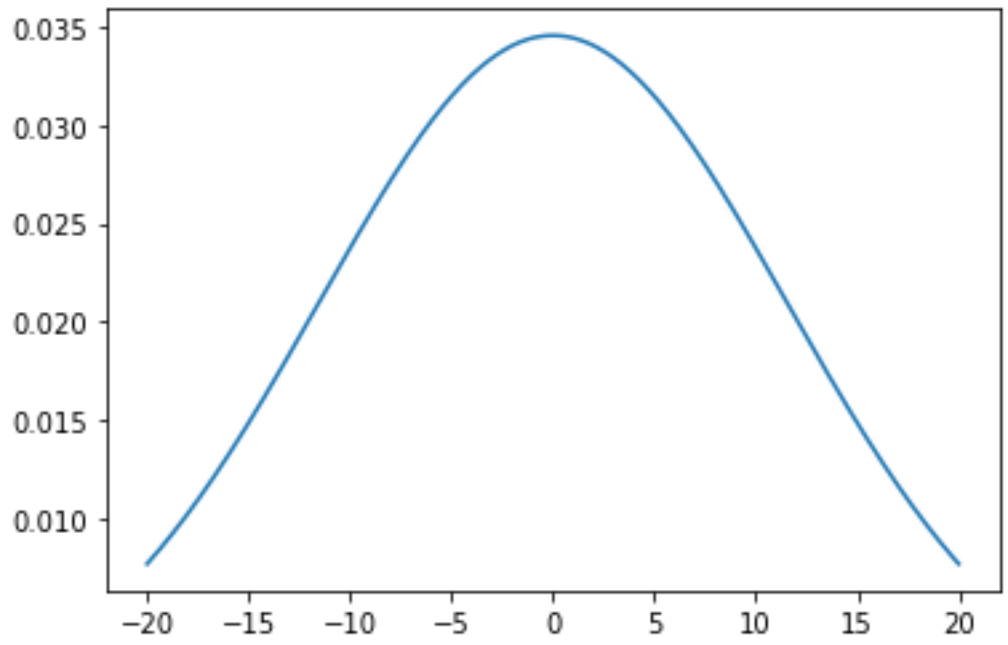
[768 rows x 3 columns]

6. Apply and explore various plotting functions on UCI data sets.

a. Normal curves

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics
# Plot between -10 and 10 with .001 steps.
x_axis = np.arange(-20, 20, 0.01)
# Calculating mean and standard deviation
mean = statistics.mean(x_axis)
sd = statistics.stdev(x_axis)
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.show()
```

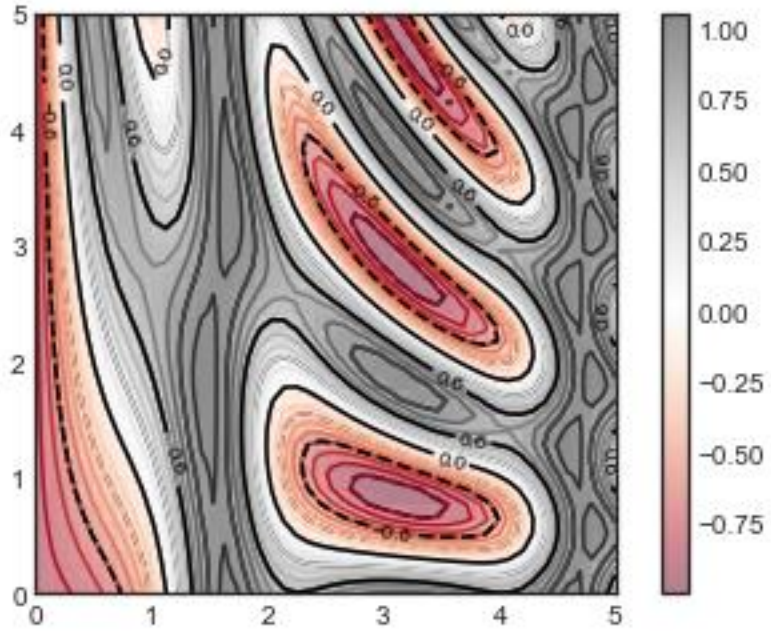
Output:



b. Density and contour plots

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
import numpy as np
def f(x, y):
return np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)
x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
plt.contour(X, Y, Z, colors='black');
plt.contour(X, Y, Z, 20, cmap='RdGy');
contours = plt.contour(X, Y, Z, 3, colors='black')
plt.clabel(contours, inline=True, fontsize=8)
plt.imshow(Z, extent=[0, 5, 0, 5], origin='lower', cmap='RdGy', alpha=0.5)
plt.colorbar();
```

Output:



c. Correlation and scatter plots

```
import pandas as pd
diab=pd.read_csv("diabetes.csv")
print("Diabetes DataFile headers Details")
print(diab.head())
import seaborn as sns
sns.scatterplot(x="BloodPressure", y="BMI", data=diab);
ax = sns.scatterplot(x="BloodPressure", y="BMI", data=diab)
ax.set_title("BloodPressure vs. BMI")
ax.set_xlabel("BloodPressure");
sns.lmplot(x="BloodPressure", y="BMI", data=diab);
sns.lmplot(x="BloodPressure", y="BMI", hue="BloodPressure", data=diab);
from scipy import stats
print("Correaltion coefficient between BloodPressure and BMI")
print(stats.pearsonr(diab['BloodPressure'], diab['BMI']))
cormat = diab.corr()

print("correlation MATRIX")
print(round(cormat,2))
sns.heatmap(cormat);
```

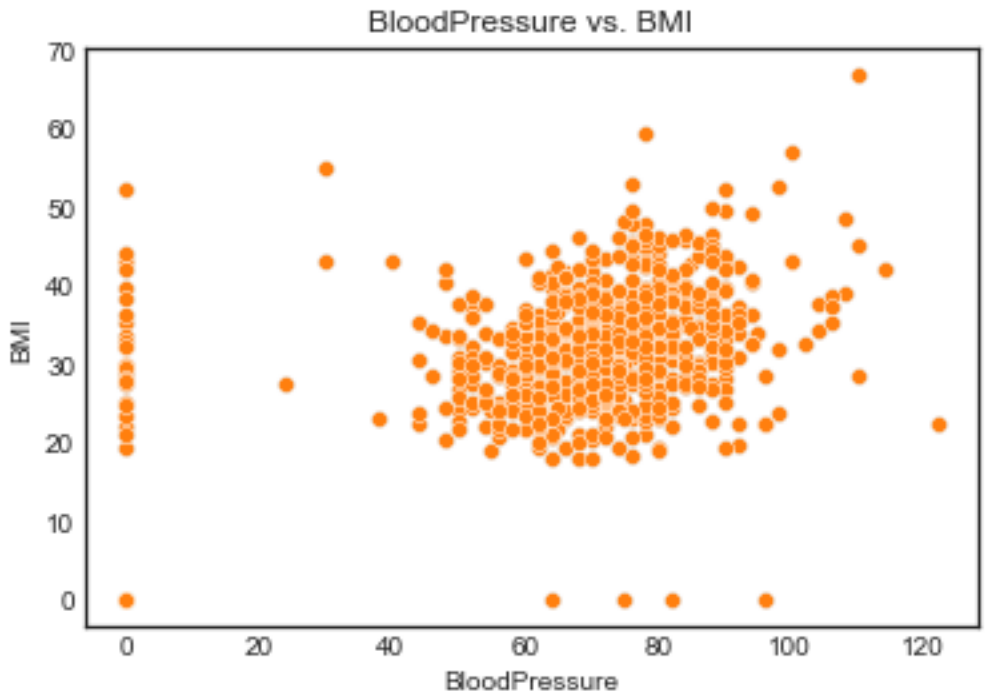
Output:

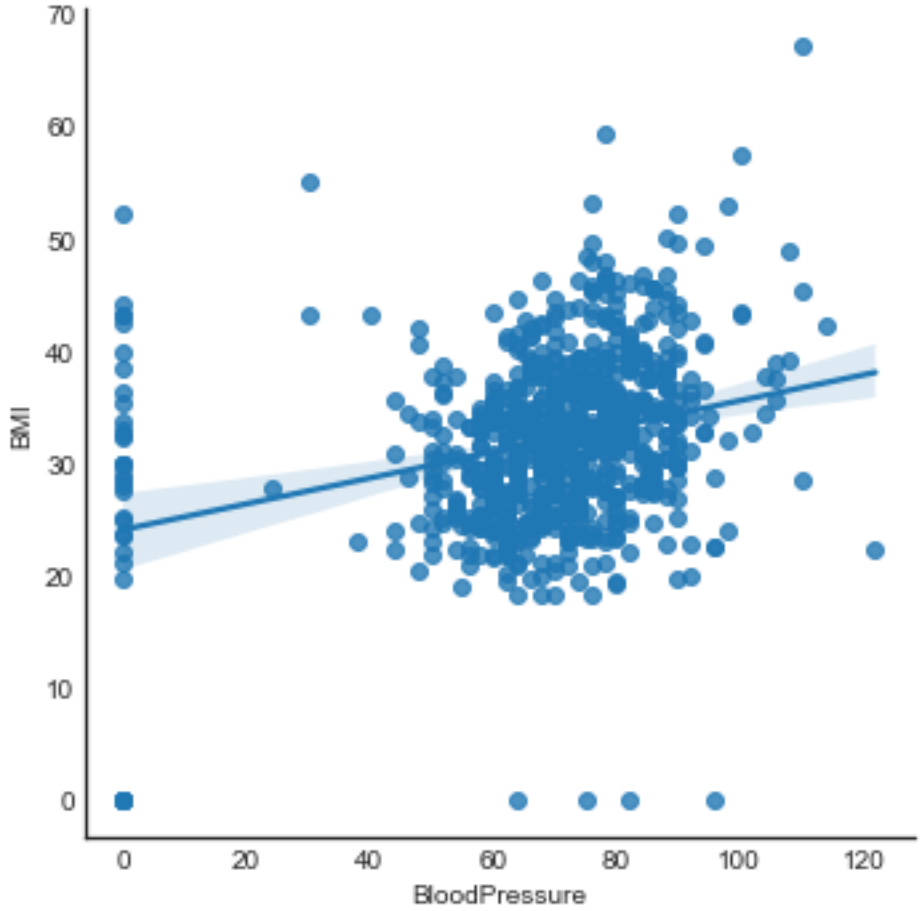
```
Diabetes DataFile headers Details
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
Correaltion coefficient between BloodPressure and BMI
(0.2818052888499105, 1.737888383236698e-15)
correlation MATRIX
                          Pregnancies  Glucose  BloodPressure  SkinThickness  \
Pregnancies                      1.00     0.13           0.14          -0.08
Glucose                          0.13     1.00           0.15           0.06
BloodPressure                    0.14     0.15           1.00           0.21
```

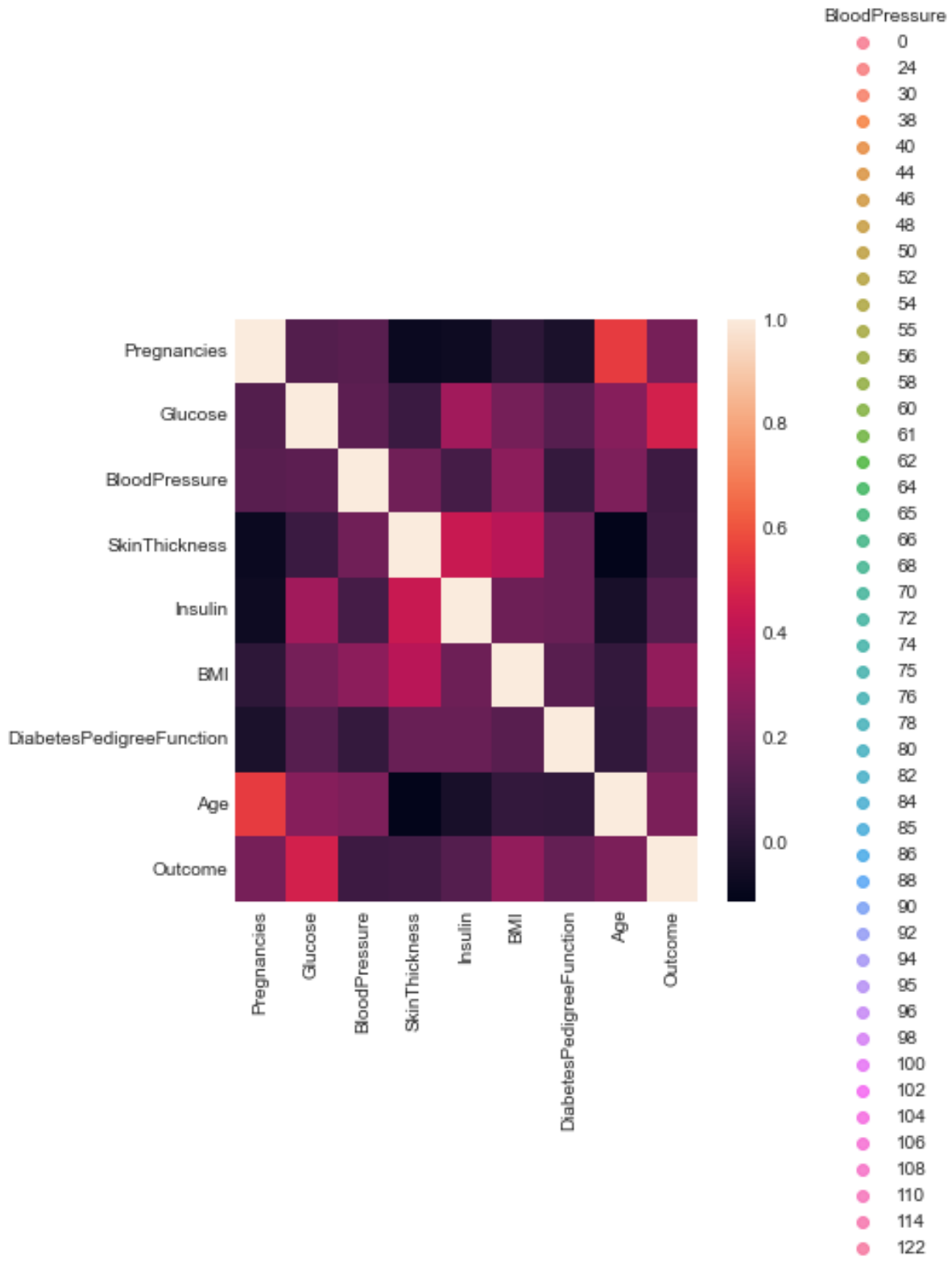
SkinThickness	-0.08	0.06	0.21	1.00
Insulin	-0.07	0.33	0.09	0.44
BMI	0.02	0.22	0.28	0.39
DiabetesPedigreeFunction	-0.03	0.14	0.04	0.18
Age	0.54	0.26	0.24	-0.11
Outcome	0.22	0.47	0.07	0.07

	Insulin	BMI	DiabetesPedigreeFunction	Age	\
Pregnancies	-0.07	0.02	-0.03	0.54	
Glucose	0.33	0.22	0.14	0.26	
BloodPressure	0.09	0.28	0.04	0.24	
SkinThickness	0.44	0.39	0.18	-0.11	
Insulin	1.00	0.20	0.19	-0.04	
BMI	0.20	1.00	0.14	0.04	
DiabetesPedigreeFunction	0.19	0.14	1.00	0.03	
Age	-0.04	0.04	0.03	1.00	
Outcome	0.13	0.29	0.17	0.24	

	Outcome
Pregnancies	0.22
Glucose	0.47
BloodPressure	0.07
SkinThickness	0.07
Insulin	0.13
BMI	0.29
DiabetesPedigreeFunction	0.17
Age	0.24
Outcome	1.00


BloodPressure vs. BMI
BMI
BloodPressure
0
10
20
30
40
50
60
70
0
20
40
60
80
100
120

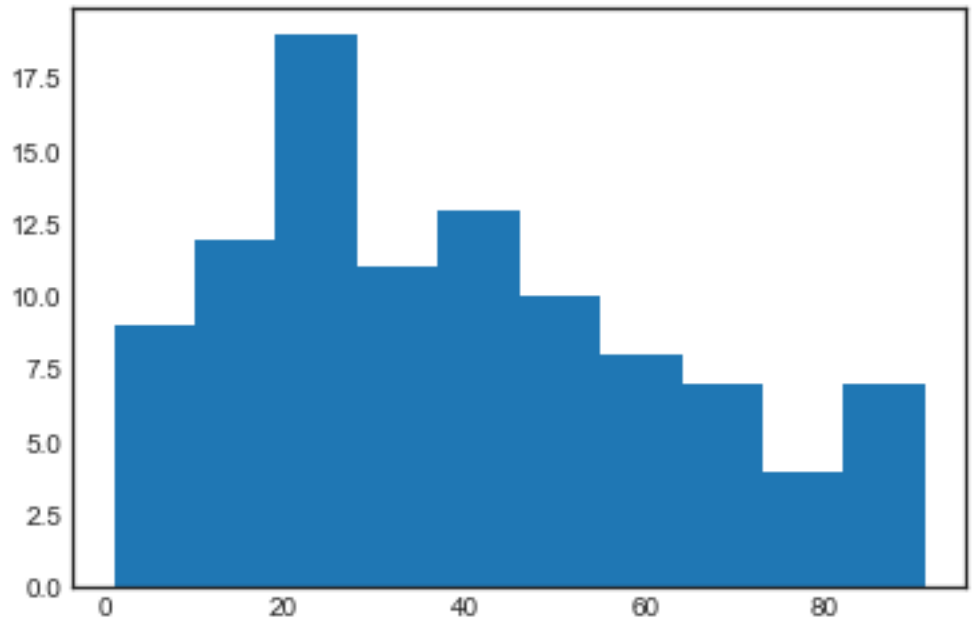

BMI
BloodPressure
0
10
20
30
40
50
60
70
0
20
40
60
80
100
120


BloodPressure
0
24
30
38
40
44
46
48
50
52
54
55
56
58
60
61
62
64
65
66
68
70
72
74
75
76
78
80
82
84
85
86
88
90
92
94
95
96
98
100
102
104
106
108
110
114
122
1.0
0.8
0.6
0.4
0.2
0.0
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome

d. Histograms

A)

```
import matplotlib.pyplot as plt
x = [1,1,2,3,3,5,7,8,9,10,
10,11,11,13,13,15,16,17,18,18,
18,19,20,21,21,23,24,24,25,25,
25,25,26,26,26,27,27,27,27,27,
29,30,30,31,33,34,34,34,35,36,
36,37,37,38,38,39,40,41,41,42,
43,44,45,45,46,47,48,48,49,50,
51,52,53,54,55,55,56,57,58,60,
61,63,64,65,66,68,70,71,72,74,
75,77,81,83,84,87,89,90,90,91
]
plt.hist(x, bins=10)
plt.show()
```



b)

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes =True)
%matplotlib inline
diab=pd.read_csv("diabetes.csv")
print("Diabetes DataFile headers Details")
print(diab.head())
dia1 = diab[diab.Outcome==1]
dia0 = diab[diab.Outcome==0]
sns.countplot(x=diab.Outcome)
```

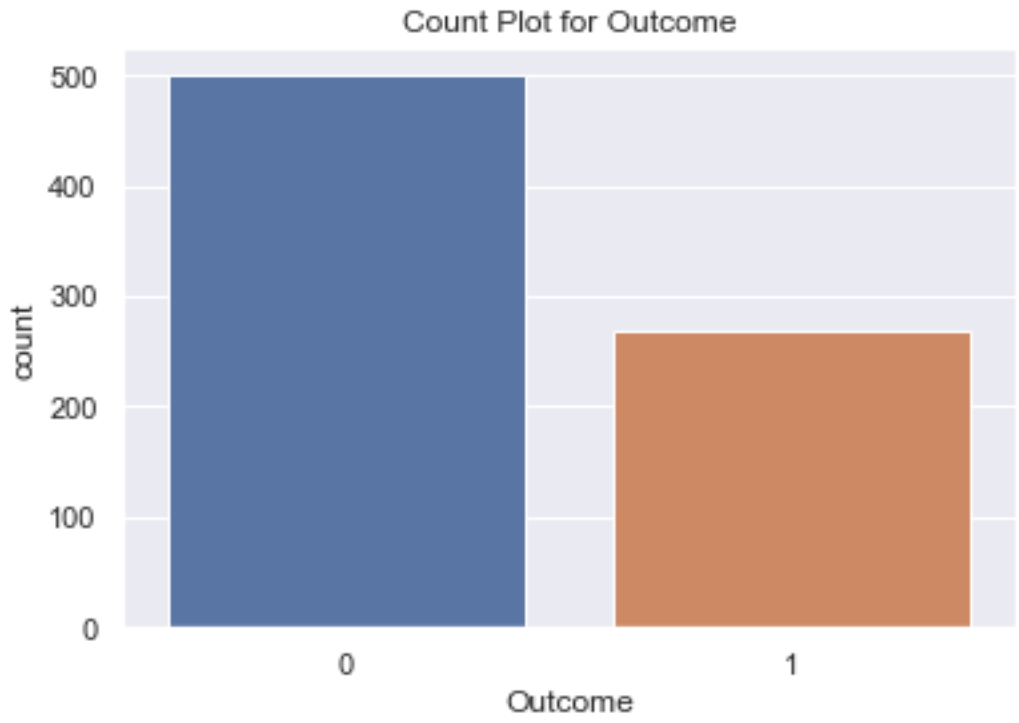
```
plt.title("Count Plot for Outcome")
# Creating 3 subplots - 1st for histogram, 2nd for histogram segmented by Outcome and 3rd for representing same segmentation using boxplot
plt.figure(figsize=(20, 6))
plt.subplot(1,3,1)
sns.set_style("dark")
plt.title("Histogram for BloodPressure")
sns.distplot(diab.BloodPressure,kde=False)
plt.subplot(1,3,2)
sns.distplot(dia0.BloodPressure,kde=False,color="Blue", label="Preg for Outome=0")
sns.distplot(dia1.BloodPressure,kde=False,color = "Gold", label = "Preg for Outcome=1")
plt.title("Histograms for Preg by Outcome")
plt.legend()
plt.subplot(1,3,3)
sns.boxplot(x=diab.Outcome,y=diab.BloodPressure)
plt.title("Boxplot for Preg by Outcome")
```

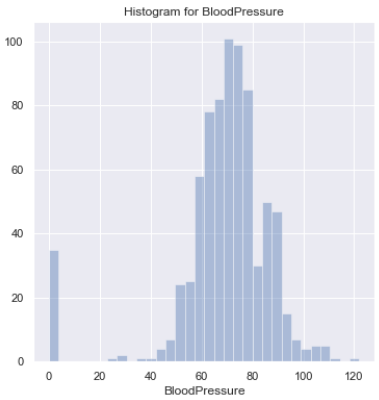
Output:

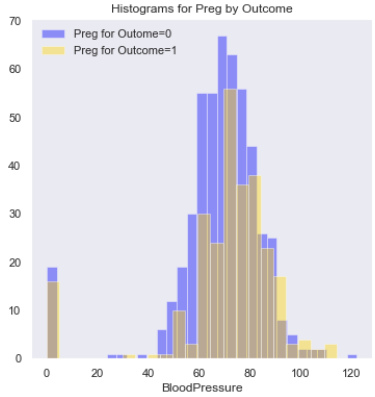
Diabetes DataFile headers Details

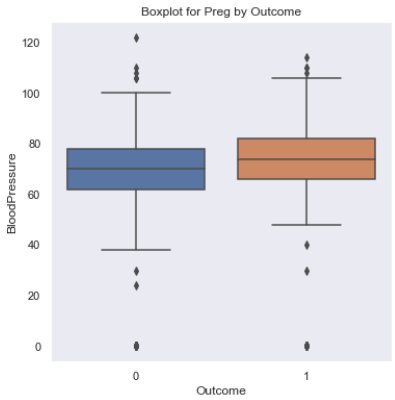
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1


Count Plot for Outcome
count
500
400
300
200
100
0
0
1
Outcome


Histogram for BloodPressure
BloodPressure

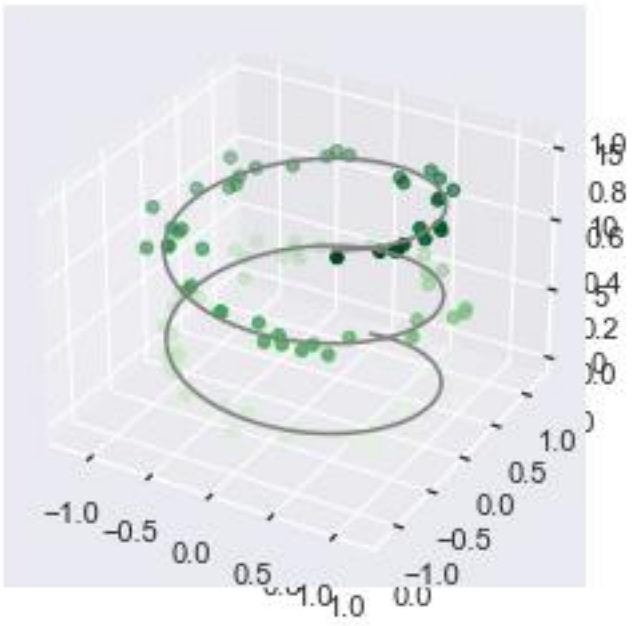

Histograms for Preg by Outcome
Preg for Outome=0
Preg for Outcome=1
BloodPressure


Boxplot for Preg by Outcome
BloodPressure
Outcome

e. Three dimensional plotting

```
from mpl_toolkits import mplot3d
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')
ax = plt.axes(projection='3d')
# Data for a three-dimensional line
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')
# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```

Output:



7.VISUALIZING GEOGRAPHIC DATA WITH BASEMAP

Program:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
fig = plt.figure(figsize=(8, 8))
m = Basemap(projection='lcc', resolution=None,
            width=8E6, height=8E6,
            lat_0=45, lon_0=-100,)
m.etopo(scale=0.5, alpha=0.5)
# Map (long, lat) to (x, y) for plotting
x, y = m(-122.3, 47.6)
plt.plot(x, y, 'ok', markersize=5)
plt.text(x, y, ' Seattle', fontsize=12);
from itertools import chain
def draw_map(m, scale=0.2):
    # draw a shaded-relief image
    m.shadedrelief(scale=scale)
    # lats and longs are returned as a dictionary
    lats = m.drawparallels(np.linspace(-90, 90, 13))
    lons = m.drawmeridians(np.linspace(-180, 180, 13))
    # keys contain the plt.Line2D instances
    lat_lines = chain(*(tup[1][0] for tup in lats.items()))
    lon_lines = chain(*(tup[1][0] for tup in lons.items()))
    all_lines = chain(lat_lines, lon_lines)
    # cycle through these lines and set the desired style
    for line in all_lines:
        line.set(linestyle='-', alpha=0.3, color='w')
```

```
fig = plt.figure(figsize=(8, 6), edgecolor='w')
m = Basemap(projection='cyl', resolution=None,
            llcrnrlat=-90, urcrnrlat=90,
            llcrnrlon=-180, urcrnrlon=180, )
draw_map(m)
fig = plt.figure(figsize=(8, 6), edgecolor='w')
m = Basemap(projection='cyl', resolution=None,
            llcrnrlat=-90, urcrnrlat=90,
            llcrnrlon=-180, urcrnrlon=180, )
draw_map(m)
fig = plt.figure(figsize=(8, 8))
m = Basemap(projection='lcc', resolution=None,
            lon_0=0, lat_0=50, lat_1=45, lat_2=55,
            width=1.6E7, height=1.2E7)
draw_map(m)
```

Output:


Seattle