PROJECT : Investigation of factors to discover reason(s) why patients show up at scheduled appointments or not.

Table of Contents

INTRODUCTION

In this section of thIS report, I will introduce my dataset and what i intend to do with my analysis.

My Data

> I will be analyzing this data which shows some factors that can be used to predict whether patients will show up for their scheduled medical appointments or not.

QUESTIONS

In this section of this report, I will state the problem question, describe the questions that I plan on exploring over the course of this report.

The Problem Question

> **Question** - What factors are important for us to know in order to predict if a patient will show up for their scheduled appointment?

My Explorative Questions

I will be comparing the absence or presence of patients as a dependent variable to some relevants and insightful independent variables.

Independent variables

- Appointment Date convienence
- Gender
- Age
- Hypertension Status
- Neigbhourhood
- Diabetes Status
- Alcoholism Status
- Handicap Status
- SMS reminder

DATA WRANGLING(CLEANING)

In this section of the report, I will load my dataset, assess it and explore it carefully to perform any necessary data cleaning or preparation.

Importing Packages

> I will import the necessary packages that i will be using for my analysis.

In [66]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import collections
from datetime import datetime
%matplotlib inline
```

Loading Data

> I already downloaded my data to my working directory, i will load them using the pandas read_csv function.

In [67]:
```
# I already downloaded the data so I will assess my data by importing the data using p
df = pd.read_csv('noshow.csv')
```

Intuitive Exploration

> I will perform basic exploration to check if my data needs cleaning

Have an Overview

In [68]:
```
# I want to have an overview of my data, so i will build intuition by using some panda
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64
 2   Gender          110527 non-null  object
 3   ScheduledDay    110527 non-null  object
 4   AppointmentDay  110527 non-null  object
 5   Age             110527 non-null  int64
 6   Neighbourhood   110527 non-null  object
 7   Scholarship     110527 non-null  int64
 8   Hipertension    110527 non-null  int64
 9   Diabetes        110527 non-null  int64
 10  Alcoholism      110527 non-null  int64
 11  Handcap         110527 non-null  int64
 12  SMS_received    110527 non-null  int64
 13  No-show         110527 non-null  object
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

- I have a total of 110,527 rows and 13 columns.
- I have 110,527 scheduled appointments.

Check for Null

In [69]:
```
df.isnull().sum()
```

Out[69]:
```
PatientId         0
AppointmentID     0
Gender            0
ScheduledDay      0
AppointmentDay    0
Age               0
Neighbourhood     0
Scholarship       0
Hipertension      0
Diabetes          0
Alcoholism        0
Handcap           0
SMS_received      0
No-show           0
dtype: int64
```

- There are no null values.

Check for duplicates

In [70]:
```
df.duplicated().sum()
```

Out[70]: 0

- There are no duplicates.

Check Unique values per column

In [71]:
```
# I want to explore the total number of unique entries in my columns to have better un
df.nunique()
```

Out[71]:
```
PatientId          62299
AppointmentID     110527
Gender                 2
ScheduledDay      103549
AppointmentDay        27
Age                  104
Neighbourhood         81
Scholarship            2
Hipertension           2
Diabetes               2
Alcoholism             2
Handcap                5
SMS_received           2
No-show                2
dtype: int64
```

- There are a total of 62,299 different patients.
- I have patients from 81 diffrent neighbourhood.

Summary Statistics of each Column

In [72]:
```
df.describe()
```

Out[72]:

	PatientId	AppointmentID	Age	Scholarship	Hipertension	Diabetes	Alcoholi
count	1.105270e+05	1.105270e+05	110527.000000	110527.000000	110527.000000	110527.000000	110527.0000
mean	1.474963e+14	5.675305e+06	37.088874	0.098266	0.197246	0.071865	0.0304
std	2.560949e+14	7.129575e+04	23.110205	0.297675	0.397921	0.258265	0.1716
min	3.921784e+04	5.030230e+06	-1.000000	0.000000	0.000000	0.000000	0.0000
25%	4.172614e+12	5.640286e+06	18.000000	0.000000	0.000000	0.000000	0.0000
50%	3.173184e+13	5.680573e+06	37.000000	0.000000	0.000000	0.000000	0.0000
75%	9.439172e+13	5.725524e+06	55.000000	0.000000	0.000000	0.000000	0.0000
max	9.999816e+14	5.790484e+06	115.000000	1.000000	1.000000	1.000000	1.0000

- Age has the minimum value of minus 1, that is impossible!!!
- I will explore if its just once or repeatedly.

Check for more negative age

In [73]:
```
#Count the total number of negatives in the age column.
collections.Counter(df.Age<0)
```

Out[73]: Counter({False: 110526, True: 1})

- This means it happened just once.
- I assume this is an input error and it will be adjusted in the data cleaning stage

DATA CLEANING

> **1) I have to rename my column titles to be corectly spelt and all in Small letters**
>
> **2) I have to adjust the wrongly inputted age**
>
> **3) The Handicap column has 5 different characters but according to the data source it should be 2(1 or 0).**
>
> **4) Since I don't intend to recognize any patient and there is no duplicate appointment, I can remove both the PatientId and AppointmentID column.**
>
> **5) The ScheduledDay and the Appointment Day can be used to check if there is a relationship between the proximity of appointment date with responsiveness of patients**
>
> **6) I will convert the gender and absence status column to boolean for statistical analysis**
>
> **7) We have 8 independent variables that are of insightful relevance to our analysis**

Renaming Columns

In [74]:
```
#Check column names again
df.columns
```

Out[74]:
```
Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hipertension',
       'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'No-show'],
      dtype='object')
```

In [75]:
```
# I will rename my columns to more user and python friendly names.
def rename_columns(namechange):
    return df.rename(columns= namechange, inplace=True)
rename_columns({'Handcap':'Handicap'})
rename_columns({'No-show':'Absence'})
rename_columns({'Hipertension':'Hypertension'})
#I will convert all to lowercase
df = df.rename(columns=str.lower)
#I want to confirm the new column names
df.columns
```

Out[75]:
```
Index(['patientid', 'appointmentid', 'gender', 'scheduledday',
       'appointmentday', 'age', 'neighbourhood', 'scholarship', 'hypertension',
       'diabetes', 'alcoholism', 'handicap', 'sms_received', 'absence'],
      dtype='object')
```

Fill the wrongly inputted age column with the mean age.

In [76]:
```
#Calculate the mean for the age column
df.age.mean()
```

Out[76]: 37.08887421173107

In [77]:
```
# I will fill the negative age row with 37 since age can't be decimal.
df.age.replace(to_replace=-1, value=37, inplace=True)
collections.Counter(df.age<0)
```

Out[77]: Counter({False: 110527})

Adjust the entry error in the handicap column.

In [78]:
```
#Check the unique values of the handicap column
df.handicap.unique()
```

Out[78]: array([0, 1, 2, 3, 4], dtype=int64)

In [79]:
```
#To be sure they are not higher categories.
#Let me see the frequency of each unique value
collections.Counter(df.handicap)
```

Out[79]: Counter({0: 108286, 1: 2042, 2: 183, 3: 13, 4: 3})

- These values have a relatively low frequency and I will convert them to signify handicap status to be safe.

In [80]:
```
# Replace the negative age with the'1' category
df.handicap.replace(to_replace=[2,3,4], value=1, inplace=True )
collections.Counter(df.handicap)
```

Out[80]: Counter({0: 108286, 1: 2241})

Convert Proximity of ScheduledDay to AppointmentDay to categorical variable

- I am checking if those who had the appointment sceheduled for the day the they booked it had an advantage or not.
- I will convert having it on the same day as 1.
- i will convert it being on another day as 0.

In [81]:
```
# I will be creating two diffrent lists of the Scheduled Day and the Appointment Day
scheduled = list(df.scheduledday)
appointment = list(df.appointmentday)
#I will combine this list by zipping them together.
dates = list(zip(scheduled, appointment))
#Let me see my combined list
dates[:10]
```

Out[81]:
```
[('2016-04-29T18:38:08Z', '2016-04-29T00:00:00Z'),
 ('2016-04-29T16:08:27Z', '2016-04-29T00:00:00Z'),
 ('2016-04-29T16:19:04Z', '2016-04-29T00:00:00Z'),
 ('2016-04-29T17:29:31Z', '2016-04-29T00:00:00Z'),
 ('2016-04-29T16:07:23Z', '2016-04-29T00:00:00Z'),
 ('2016-04-27T08:36:51Z', '2016-04-29T00:00:00Z'),
 ('2016-04-27T15:05:12Z', '2016-04-29T00:00:00Z'),
 ('2016-04-27T15:39:58Z', '2016-04-29T00:00:00Z'),
 ('2016-04-29T08:02:16Z', '2016-04-29T00:00:00Z'),
 ('2016-04-27T12:48:25Z', '2016-04-29T00:00:00Z')]
```

In [82]:
```
# I will now create another list for my category.
same_dates = []
for sday, aday in dates:
    same = sday[5:10] == aday[5:10]
    same_dates.append(same)
#Let me check my new list
same_dates[:10]
```

Out[82]: [True, True, True, True, True, False, False, False, True, False]

In [83]:
```
#I will convert this to another list that will bear boolean values based on the result
same_day = []
for d in same_dates:
    if d==True:
        d=1
    else:
        d=0
    same_day.append(d)
same_day[:10]
```

Out[83]: [1, 1, 1, 1, 1, 0, 0, 0, 1, 0]

In [84]:
```
# I want to add the list as a column on my table.
df['samedate'] = same_day
```

In [85]:
```
#I want to confirm my process.
df.samedate.unique()
```

Out[85]: array([1, 0], dtype=int64)

Drop irrelevant columns

> I will drop the columns that I am not using in my analysis.

- AppointmentID
- PatientID
- SceduledDtae
- AppointmentDate

In [86]:
```
# Check my column names again
df.columns
```

Out[86]:
```
Index(['patientid', 'appointmentid', 'gender', 'scheduledday',
       'appointmentday', 'age', 'neighbourhood', 'scholarship', 'hypertension',
       'diabetes', 'alcoholism', 'handicap', 'sms_received', 'absence',
       'samedate'],
      dtype='object')
```

In [87]:
```
#Drop irrelevant Columns
df.drop(['patientid', 'appointmentid', 'scheduledday', 'appointmentday'], axis=1, inpl
```

In [88]:
```
#Check the first 5 rows of my data
df.head()
```

Out[88]:

	gender	age	neighbourhood	scholarship	hypertension	diabetes	alcoholism	handicap	sms_received	abs
0	F	62	JARDIM DA PENHA	0	1	0	0	0	0	
1	M	56	JARDIM DA PENHA	0	0	0	0	0	0	
2	F	62	MATA DA PRAIA	0	0	0	0	0	0	
3	F	8	PONTAL DE CAMBURI	0	0	0	0	0	0	
4	F	56	JARDIM DA PENHA	0	1	1	0	0	0	

Convert the absence and gender column to boolean for statistical analysis.

In [89]:
```
#Convert the absence and gender columns to boolean
df.gender.replace(to_replace = ['F', 'M'], value = [0, 1], inplace=True)
df.absence.replace(to_replace = ['No', 'Yes'], value = [0, 1], inplace=True)
```

- I changed Female and Male to 0 and 1 respoectively.
- I changed Yes (absent) and No (Present) to 1 and 0 respectively.

In [90]:
```
#Confirm the absence status Column
collections.Counter(df.absence)
```

Out[90]: Counter({0: 88208, 1: 22319})

In [91]:
```
#Confirm the gender column
collections.Counter(df.gender)
```

Out[91]: Counter({0: 71840, 1: 38687})

View my clean dataset

In [92]:
```
#View my clean data
df.head(5)
```

Out[92]:

	gender	age	neighbourhood	scholarship	hypertension	diabetes	alcoholism	handicap	sms_received	abs
0	0	62	JARDIM DA PENHA	0	1	0	0	0	0	
1	1	56	JARDIM DA PENHA	0	0	0	0	0	0	
2	0	62	MATA DA PRAIA	0	0	0	0	0	0	
3	0	8	PONTAL DE CAMBURI	0	0	0	0	0	0	
4	0	56	JARDIM DA PENHA	0	1	1	0	0	0	

> **My Data is clean and ready for analysis**

EXPLORATORY DATA ANALYSIS

> I have cleaned my data, i want to go into deep exploration, compute statistics and create visualizations to address the research questions that i have intriduced earlier.
>
> I will be performing analysis to detect association and not to imply causation.
>
> I will be comparing a single dependent Variable which is the presence or absence of the patients on the appointment date with 9 other independent variables.
>
> These independent variables are in three categories.

Natural Characteristics of Patients

Gender

Age

Disease Status of Patients

Hypertension

Diabetes

Handicap

Alcoholism

External factors on Patients

Scholarship

SMS Recieval

Same Day Appointment

Neighbourhood

GENERAL ANALYSIS

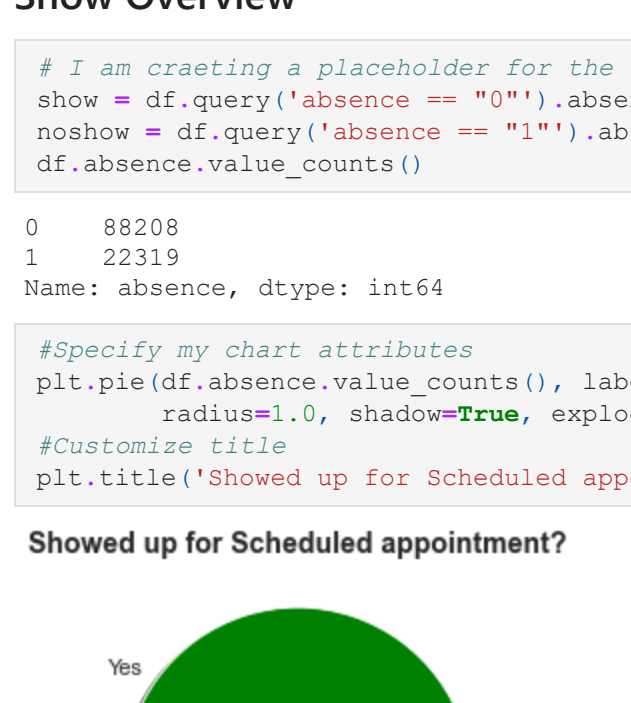
> The Overrall aim of the analysis is to analyze the resposiveness for scheduled medical appointments.
>
> I will do a general overview of those who show up and those who don't
>
> I will also have a general corelation map that relates all our independent variables to our independent variables
>
> I will plot an histogram that just shows al our variables and their distribution.

Show Overview

In [93]:
```
# I am creating a placeholder for the frequency of those that showed up and didnt to p
show = df.query('absence == "0"').absence.count()
noshow = df.query('absence == "1"').absence.count()
df.absence.value_counts()
```

Out[93]:
```
0    88208
1    22319
Name: absence, dtype: int64
```

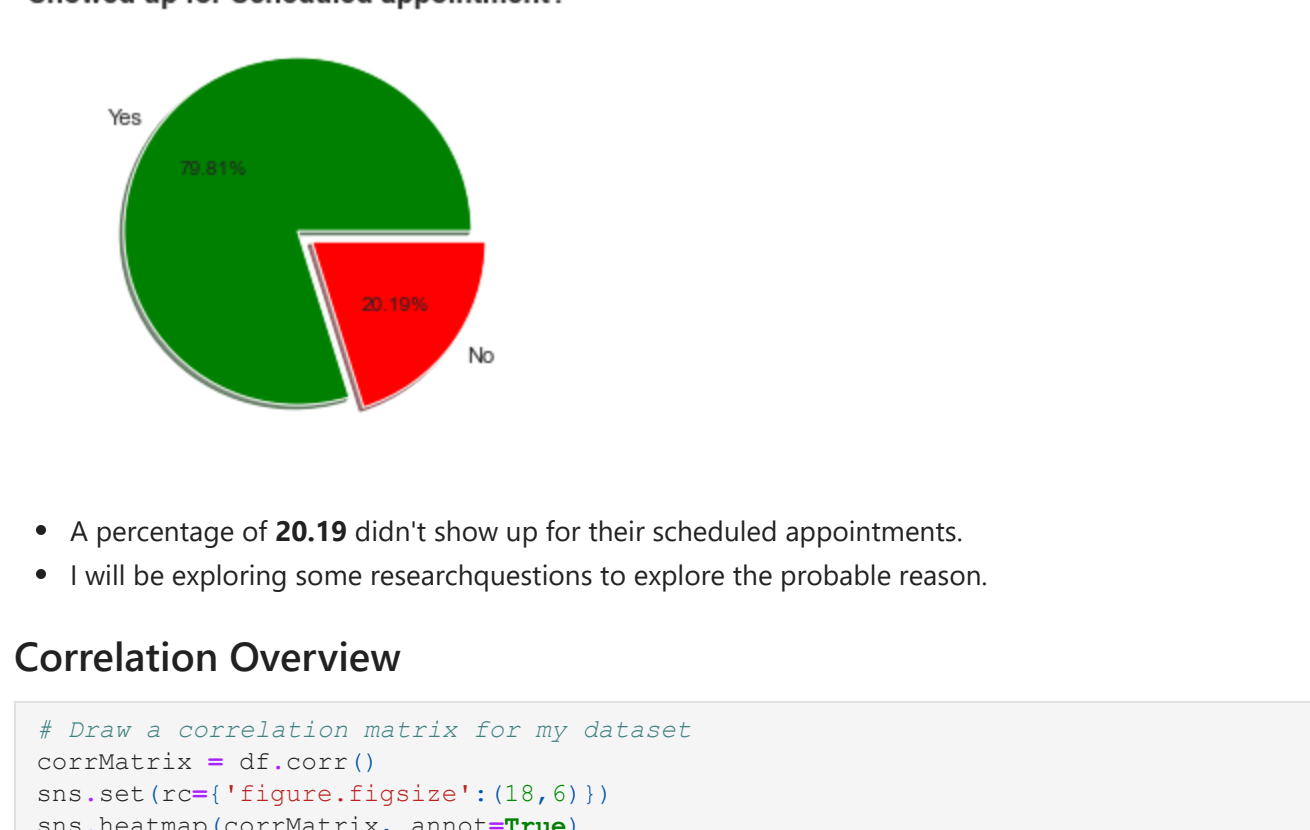
In [94]:
```
#Specify my chart attributes
plt.pie(df.absence.value_counts(), labels=['Yes', 'No'], colors=['Green', 'Red'],
        radius=1.0, shadow=True, explode = (0, 0.1), autopct='%.2f%%')
#Customize title
plt.title('Showed up for Scheduled appointment?', fontsize=14, fontweight='bold');
```



- A percentage of **20.19** didn't show up for their scheduled appointments.
- I will be exploring some researchquestions to explore the probable reason.

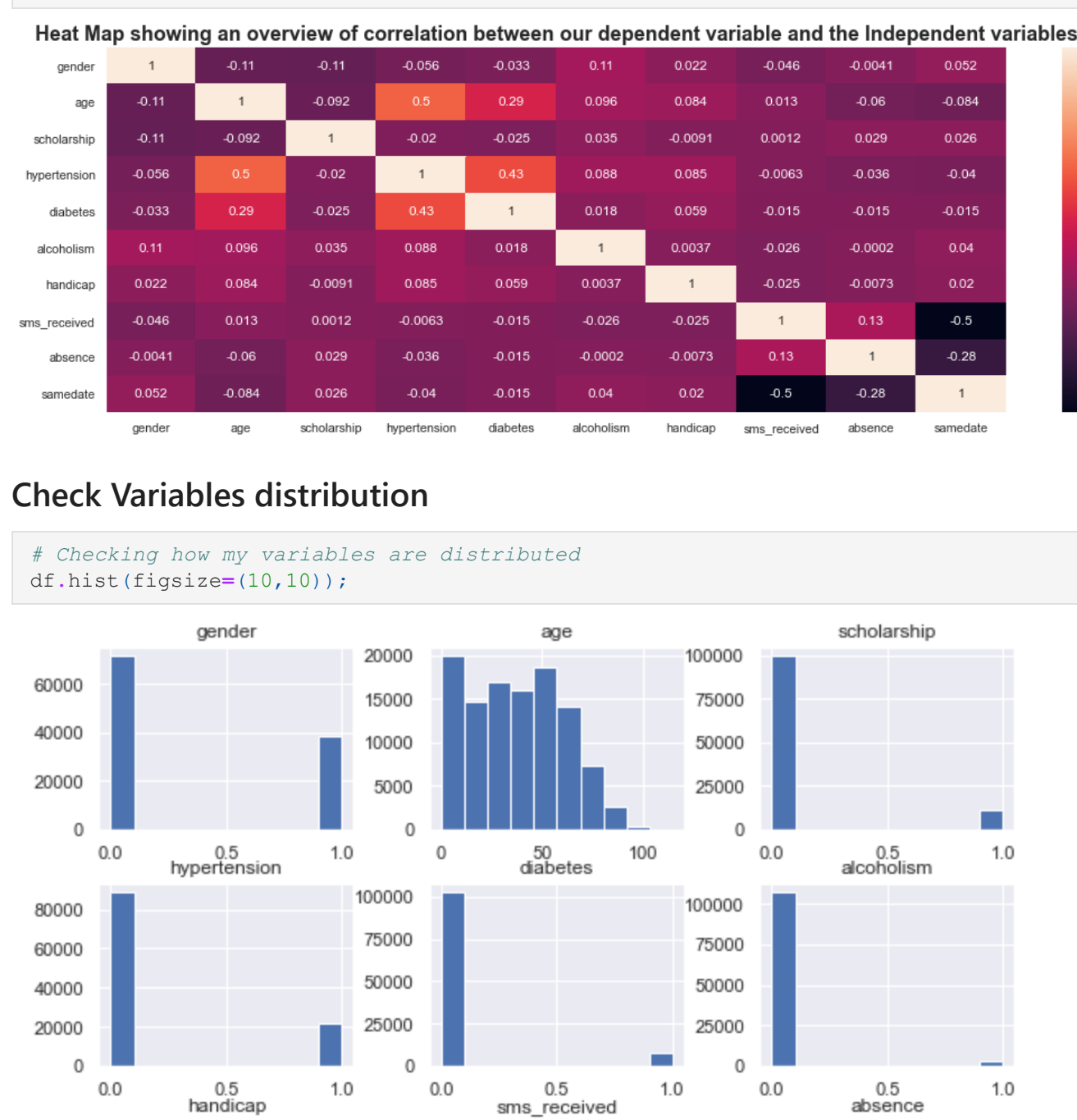
Correlation Overview

In [95]:
```
# Draw a correlation matrix for my dataset
corrMatrix = df.corr()
sns.set(rc={'figure.figsize':(18,6)})
sns.heatmap(corrMatrix, annot=True)
plt.title('Heat Map showing an overview of correlation between our dependent variable
          fontsize=18, fontweight='bold')
plt.show()
```



Check Variables distribution

In [96]:
```
# Checking how my variables are distributed
df.hist(figsize=(10,10));
```



Insights

> **There are some insights that are relevant to our anlysis that can be gotten directly from this heatmap, but we will explore deeper and explain with more visualizations our insights.**
>
> **One important premise i want to lay is that there isn't a very strong correlation between these independent varibles to explore causation but they are good for establishing association on a comaparative basis.**

1) NATURAL CHARACTERISTICS OF PATIENTS

A. GENDER

> I will first have an overview of the the Gender distribution.
>
> i will then comapre their responsiveness to appointments by Gender distribution.

> **Correlation Coefficient**

In [97]:
```
#I will calculate the correlation coefficient.
df['absence'].corr(df['gender'])
```

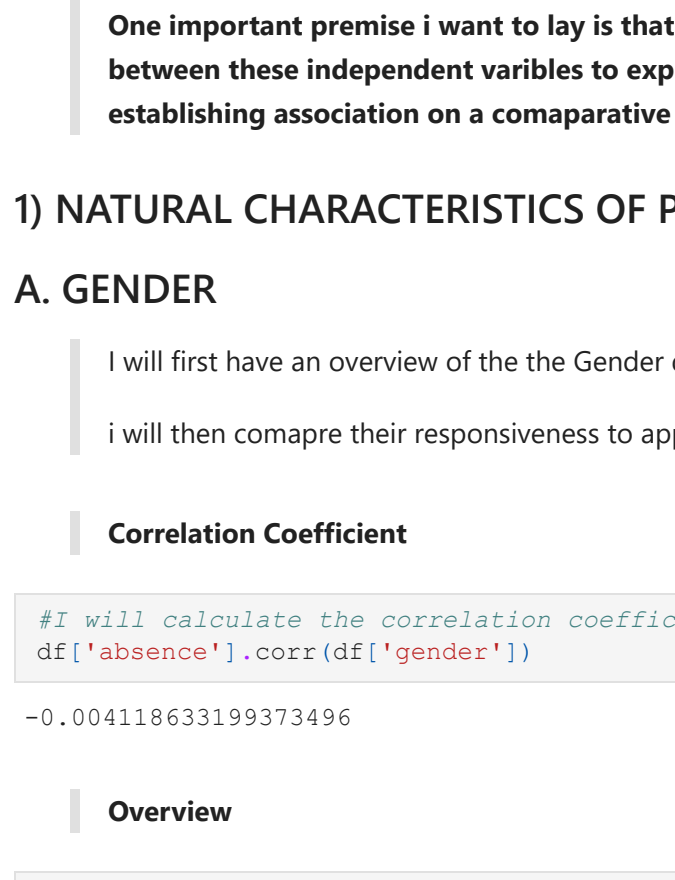
Out[97]: -0.004118633199373496

> **Overview**

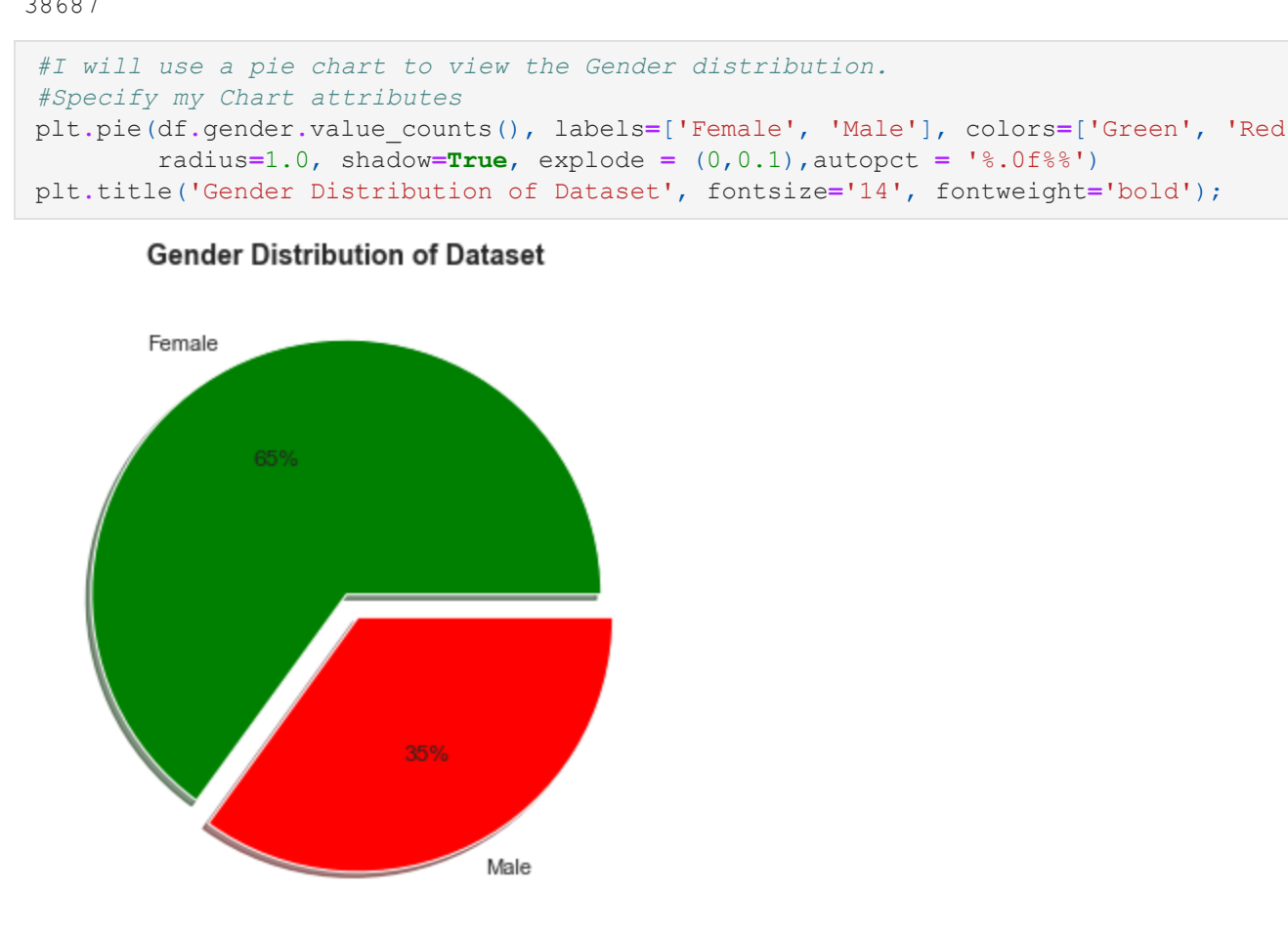
In [98]:
```
#I want to have an idea of my values
print(df.query('gender == "0"').gender.count())
print(df.query('gender == "1"').gender.count())
```

```
71840
38687
```

In [99]:
```
#I will use a pie chart to view the Gender distribution.
#Specify my Chart attributes
plt.pie(df.gender.value_counts(), labels=['Female', 'Male'], colors=['Green', 'Red'],
        radius=1.0, shadow=True, explode = (0,0.1),autopct = '%.0f%%')
plt.title('Gender Distribution of Dataset', fontsize='14', fontweight='bold');
```



In [100]:
```
# I will use a multiple bar chart to examine if there is any visible difference.
ax = sns.countplot(x="gender", hue="absence", data=df)
plt.xticks([0, 1], ['Female', 'Male'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by Gender',  fontsize=18, fontweight='bold')
plt.xlabel('Gender', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```



Observation

> There is no visible variation by gender according to our visualization and the correlation coefficient is very close to zero compared to other variables

B. AGE

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the Age distribution.
>
> I will then compare their responsiveness to appointments by their age.

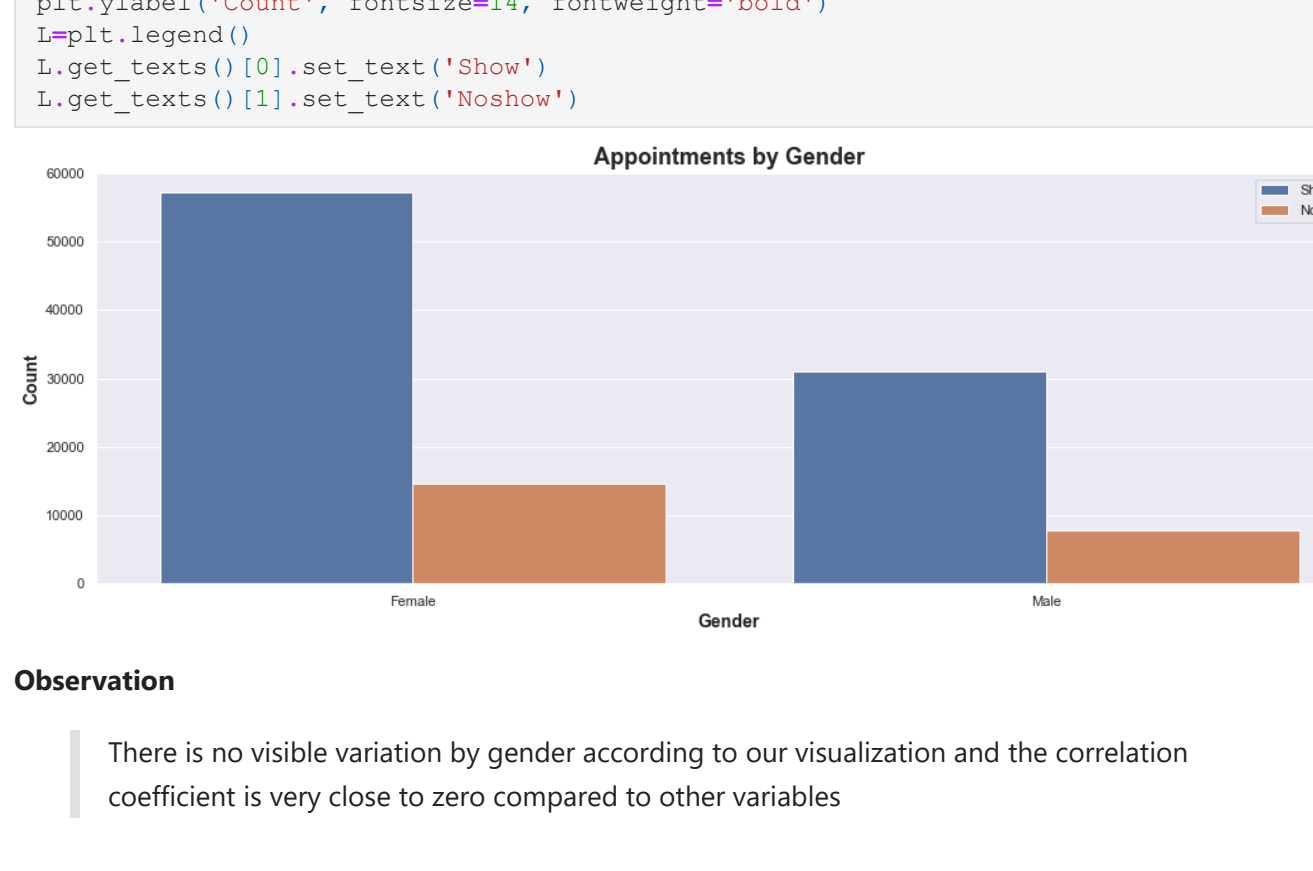
> **Correlation Coefficient**

In [101…
```
#I will calculate the correlation coefficient.
df['absence'].corr(df['age'])
```

Out[101…: -0.06032673609564155

Overview

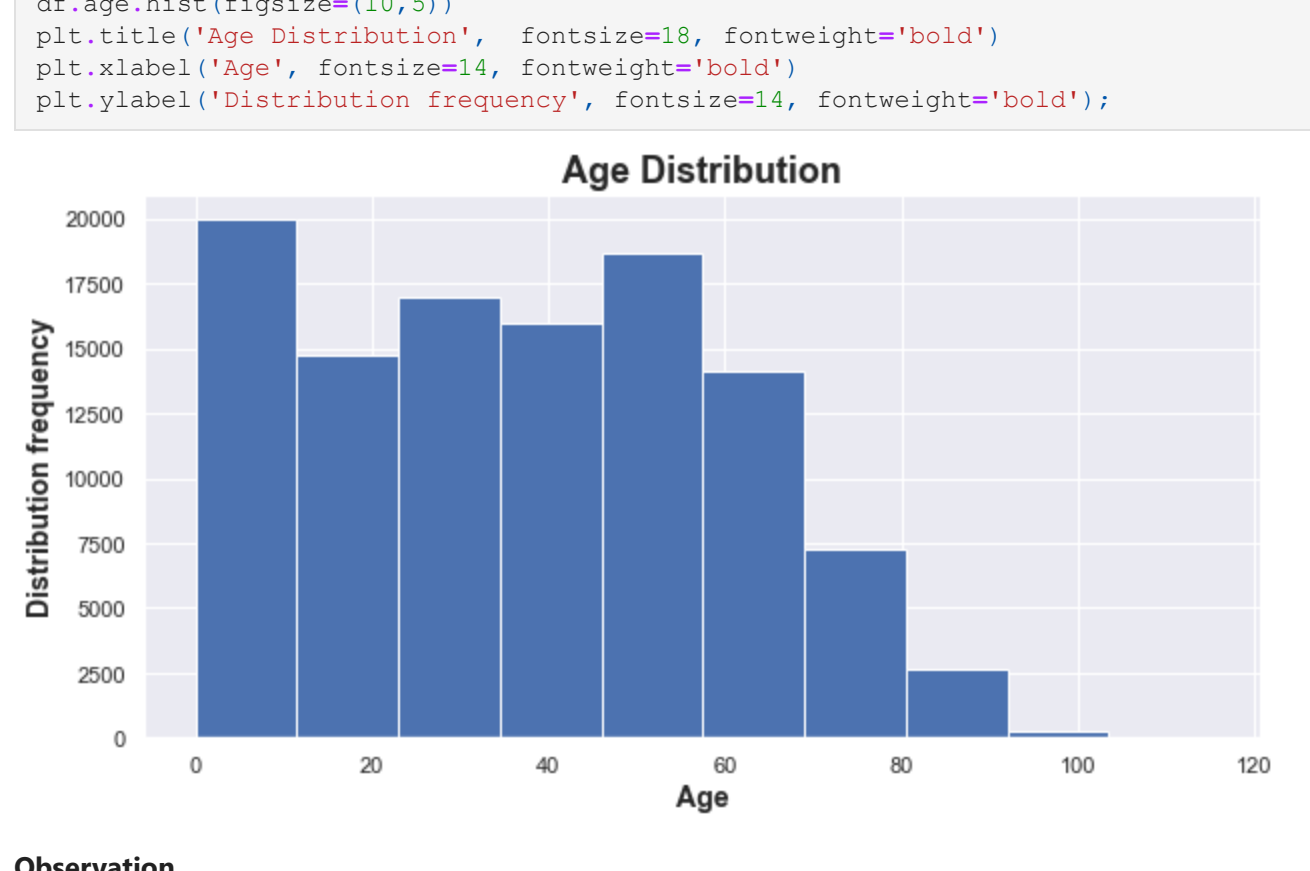
In [102…
```
#Check how my age column is distributed
df.age.hist(figsize=(10,5))
plt.title('Age Distribution',  fontsize=18, fontweight='bold')
plt.xlabel('Age', fontsize=14, fontweight='bold')
plt.ylabel('Distribution frequency', fontsize=14, fontweight='bold');
```



Observation

> The Age distribution is skewed to the left which means we generally have more younger people.

In [103…
```
#i willl use boxplot
ax=sns.catplot(y="age", col="absence", kind="box", data=df)
sns.set(rc={"figure.figsize":(8,4)})
ax.set_axis_labels("Show or No show", "Age")
ax.fig.suptitle('Box Plot Age Comparision');
```



Observation

> There is a relatively lower mean for those that didn't show up.

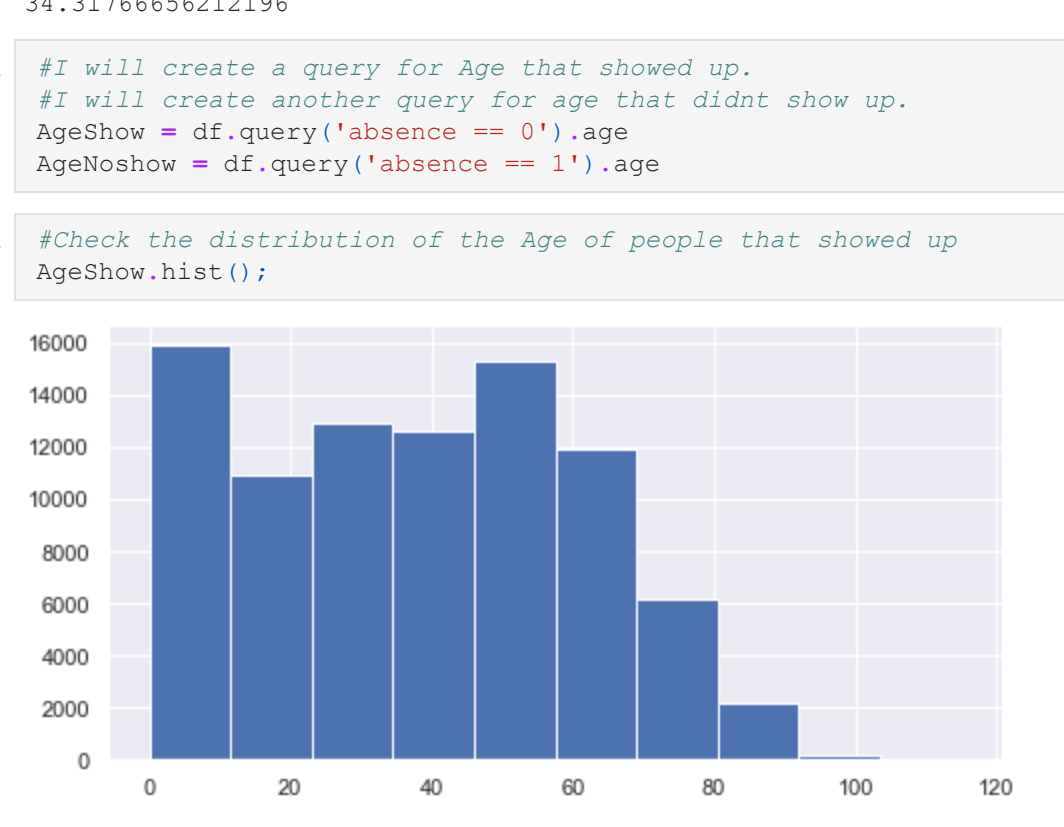
In [104…
```
#Lets confirm by calculating the mean.
print(df.query('absence == 0').age.mean())
print(df.query('absence == 1').age.mean())
```

```
37.79049519317976
34.31766656212196
```

In [105…
```
#I will create a query for Age that showed up.
#I will create another query for age that didnt show up.
AgeShow = df.query('absence == 0').age
AgeNoshow = df.query('absence == 1').age
```

In [106…
```
#Check the distribution of the Age of people that showed up
AgeShow.hist();
```

In [107…
```
# Check the distribution of the Age of people that showed up
AgeNoshow.hist();
```



INSIGHTS

> **Older people show up more for medical appointment compared to teenager and youths.**

Very young people show up more for medical appointment compared to teenager and youths.

2) DISEASE STATUS OF PATIENTS.

A.) HYPERTENSION

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the distribution.
>
> I will then compare their responsiveness to appointments by hypertensive status.

CORELATION COEFICIENT

```
#Calculate the correlation coeficient for absence and hypertension
df['absence'].corr(df['hypertension'])
```

```
-0.03570117734501494
```

Observation

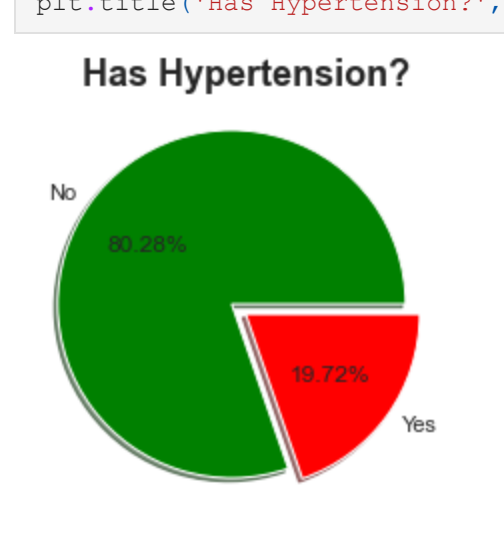
- The negative correlation shows that patients who have hypertension are less likely to miss their appointments

OVERVIEW

```
#Check the the count of both categories of hypertension status of patients
print(df.query('hypertension == 0').hypertension.count())
print(df.query('hypertension == 1').hypertension.count())
```

```
88726
21801
```

```
# Plot a pie chart to show the counts
plt.pie(df.hypertension.value_counts(), labels=['No', 'Yes'], colors =['Green', 'Red']
        explode=(0.1,0), shadow=True, radius=1.0, autopct = '%.2f%%' )
plt.title('Has Hypertension?', fontsize=18, fontweight='bold');
```

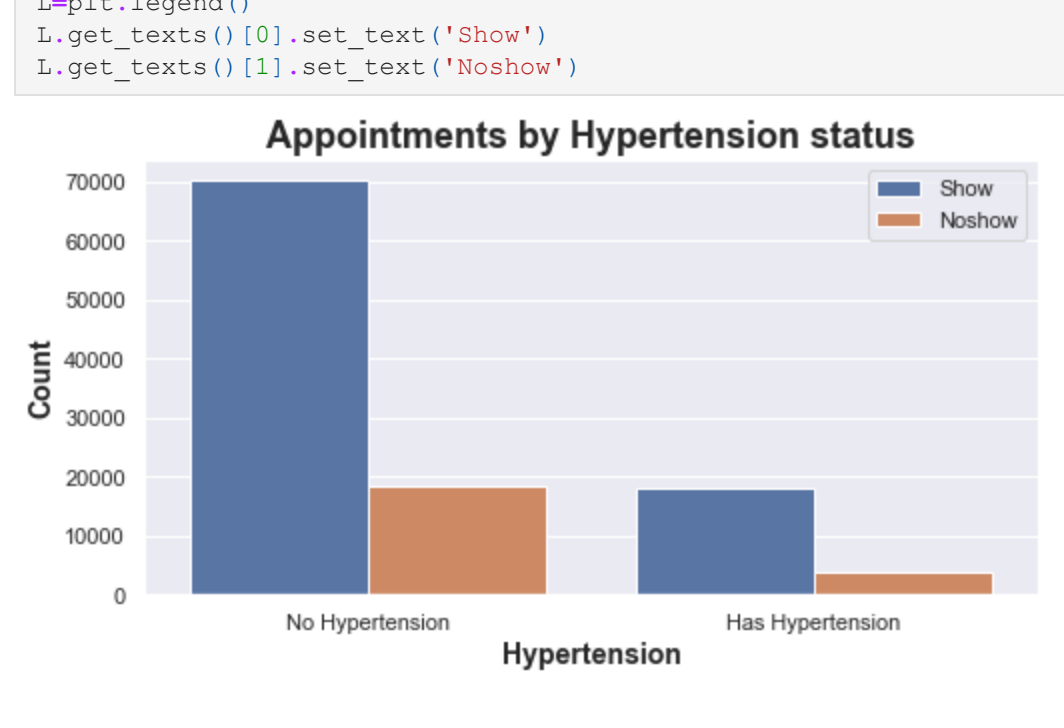


Observation

- Over 80% of the patients don't have hypertension.

COMPARE APPOINTMENTS BY HYPERTENSION

```
#Plot a mutiple column chart to show the comparison betweeen hypertension status and p
ax = sns.countplot(x="hypertension", hue="absence", data=df)
plt.xticks([0, 1], ['No Hypertension', 'Has Hypertension'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by Hypertension status',  fontsize=18, fontweight='bold')
plt.xlabel('Hypertension', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```



INSIGHTS

> **There isn't a strong correlation betwen Appointment and Hypertension Status.**
>
> **Patients who are hypertensive are less likely to miss appointment compared to those who are not**

B.) DIABETES

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the distribution.
>
> I will then compare their responsiveness to appointments by diabetes status.

CORELATION COEFICIENT

```
#Calculate the correlation coeficient for absence and diabetes
df['absence'].corr(df['diabetes'])
```

```
-0.01517994382077371
```

Observation

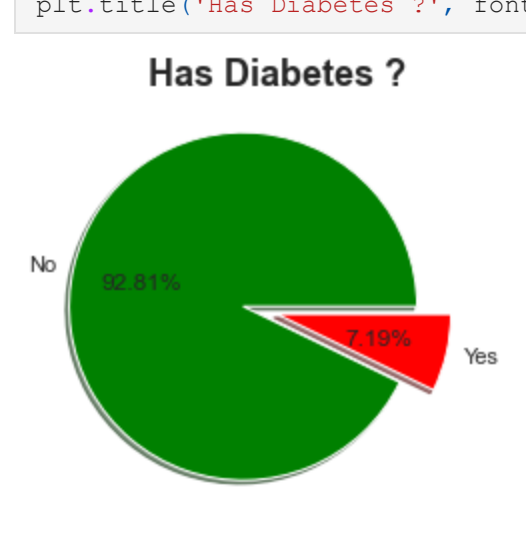
- The negative correlation shows that patients who have diabetes are less likely to miss their appointments compared to those who are not.

OVERVIEW

```
#Check the the count of both categories of diabetic status of patients
print(df.query('diabetes == 0').diabetes.count())
print(df.query('diabetes == 1').diabetes.count())
```

```
102584
7943
```

```
# Plot a pie chart to show the counts
plt.pie(df.diabetes.value_counts(), labels=['No', 'Yes'], colors =['Green', 'Red'],
        explode=(0.2,0), shadow=True, radius=1.0, autopct = '%.2f%%' )
plt.title('Has Diabetes ?', fontsize=18, fontweight='bold');
```

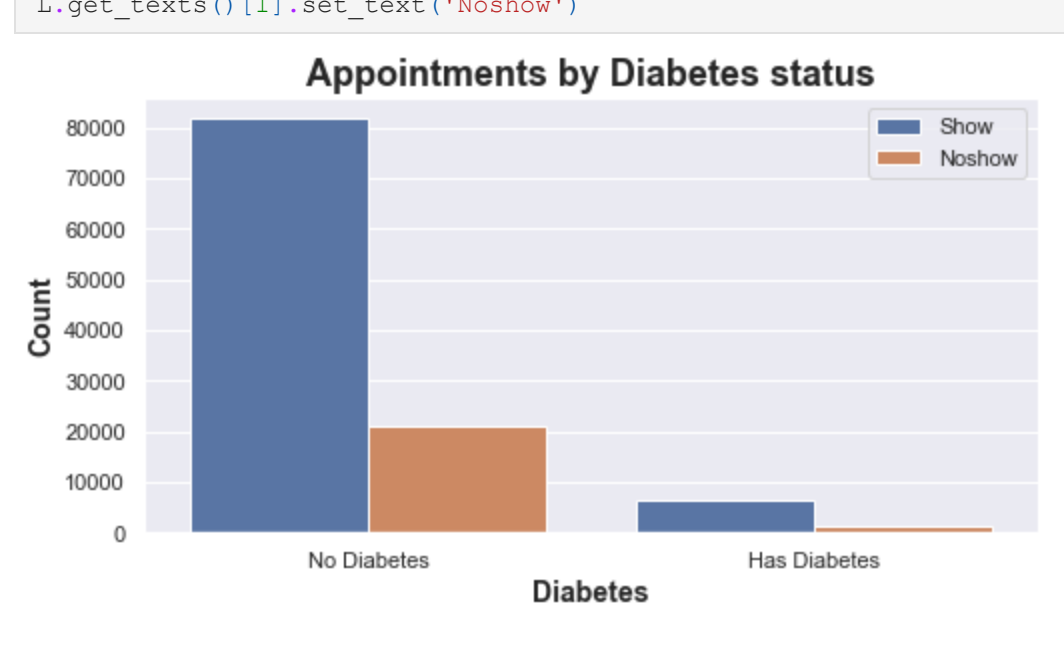


Observation

- Over 92% of the patients don't have diabetes.

COMPARE APPOINTMENT BY DIABETES

```
#Plot a mutiple column chart to show the comparison betweeen diabetic status and prese
ax = sns.countplot(x="diabetes", hue="absence", data=df)
plt.xticks([0, 1], ['No Diabetes', 'Has Diabetes'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by Diabetes status',  fontsize=18, fontweight='bold')
plt.xlabel('Diabetes', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```



INSIGHTS

> **There isn't a strong correlation betwen Appointment and Diabetes Status.**
>
> **Patients who are Diabetes are less likely to miss appointment compared to those who are not**

C.) ALCOHOLISM

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the distribution.
>
> I will then compare their responsiveness to appointments by alcoholic status.

CORELATION COEFICIENT

```
#Calculate the correlation coeficient for absence and alcoholism
df['absence'].corr(df['alcoholism'])
```

```
-0.0001960437236281405
```

Observation

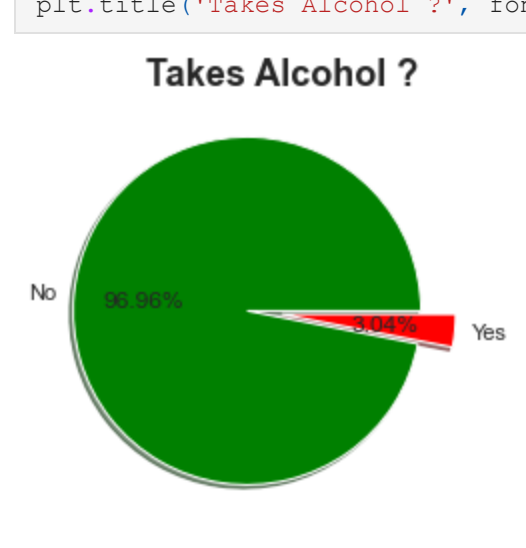
- The negative correlation shows that patients who have diabetes are less likely to miss their appointments compared to those who are not.
- The value is also very close to zero compared to other independent variables, thsimeans there is a very low association.

OVERVIEW

```
#Check the the count of both categories of alcoholic status of patients
print(df.query('alcoholism == 0').alcoholism.count())
print(df.query('alcoholism == 1').alcoholism.count())
```

```
107167
3360
```

```
# Plot a pie chart to show the counts
plt.pie(df.alcoholism.value_counts(), labels=['No', 'Yes'], colors =['Green', 'Red'],
        explode=(0.2,0), shadow=True, radius=1.0, autopct = '%.2f%%' )
plt.title('Takes Alcohol ?', fontsize=18, fontweight='bold');
```

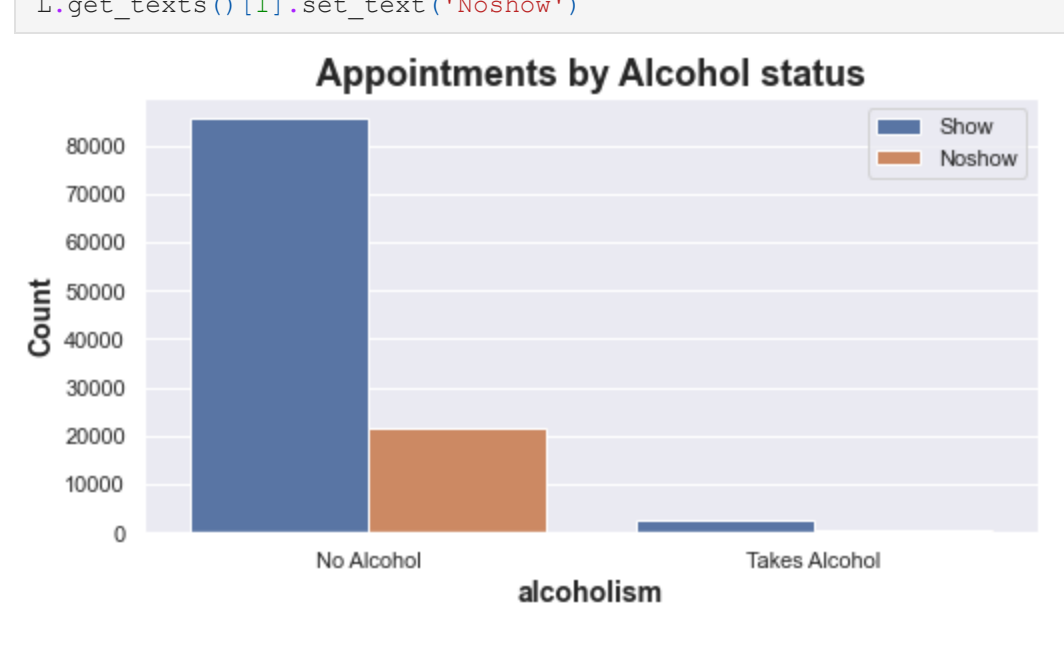


Observation

- About 97% of the patients don't take alcohol.

COMPARE APPOINTMENT BY ALCOHOL STATUS

```
#Plot a mutiple column chart to show the comparison betweeen alcoholic status and pre
ax = sns.countplot(x="alcoholism", hue="absence", data=df)
plt.xticks([0, 1], ['No Alcohol', 'Takes Alcohol'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by Alcohol status',  fontsize=18, fontweight='bold')
plt.xlabel('alcoholism', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```



INSIGHTS

> **There is a very little and negligible correlation betwen Appointment and Alcohol Status.**
>
> **No association can be established between the alcohol status and appointment**

D.) HANDICAP

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the distribution.
>
> I will then compare their responsiveness to appointments by physical disability.

CORELATION COEFICIENT

```
#Calculate the correlation coeficient for absence and disability status
df['absence'].corr(df['handicap'])
```

```
-0.007280745542466062
```

Observation

- The negative correlation shows that patients who are physically challenged are less likely to miss their appointments compared to those who are not.
- This is also the greatest association observed out of all the variables.

OVERVIEW

```
#Check the the count of both categories of disability status of patients
print(df.query('handicap == 0').handicap.count())
print(df.query('handicap == 1').handicap.count())
```

```
108286
2241
```

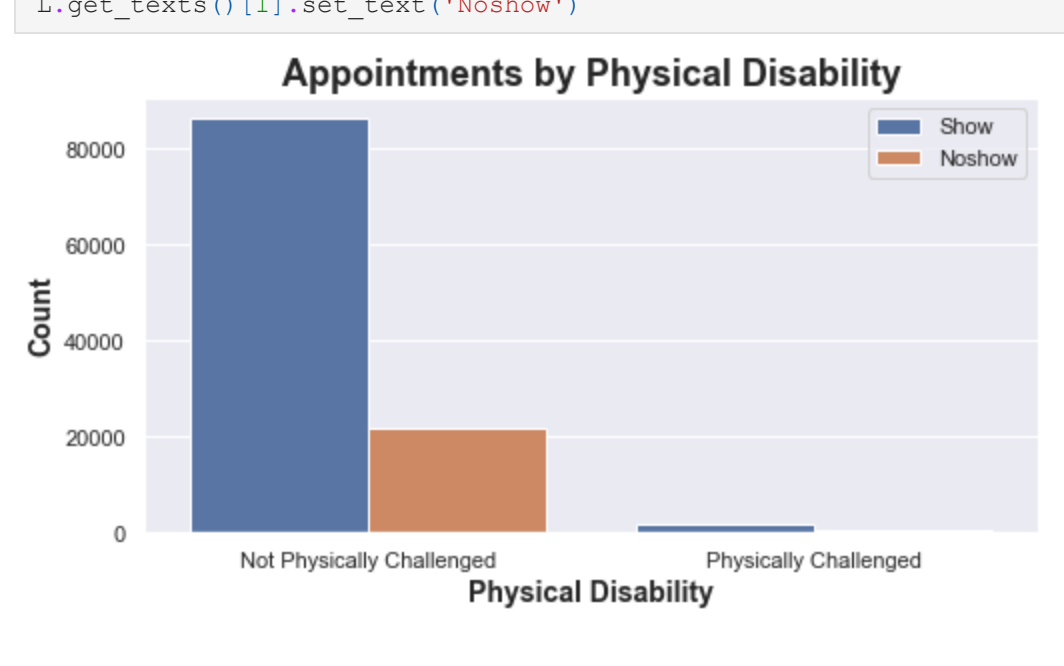
```
# Plot a pie chart to show the counts
plt.pie(df.handicap.value_counts(), labels=['No', 'Yes'], colors =['Green', 'Red'],
        explode=(0.2,0), shadow=True, radius=1.0, autopct = '%.2f%%' )
plt.title('Physically Challenged ?', fontsize=18, fontweight='bold');
```

Observation

- About 98% of the patients don't take alcohol.

COMPARE APPOINTMENT BY PHYSICAL DISABILITY

```
#Plot a mutiple column chart to show the comparison betweeen disability status and pre
ax = sns.countplot(x="handicap", hue="absence", data=df)
plt.xticks([0, 1], ['Not Physically Challenged', 'Physically Challenged'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by Physical Disability',  fontsize=18, fontweight='bold')
plt.xlabel('Physical Disability', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```



INSIGHTS

> **There isn't a very strong correlation betwen Appointment and Physically Disability.**
>
> **Patients who are Physically Challenged are less likely to miss appointment compared to those who are not**

3) EXTERNAL FACTORS ON PATIENTS.

A.) SCHOLARSHIP(Free HealthCare Beneficary)

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the distribution.
>
> I will then compare their responsiveness to appointments by acesss to sponsored healthcare.

CORELATION COEFICIENT

```
#Calculate the correlation coeficient for absence and scholarship
df['absence'].corr(df['scholarship'])
```

```
0.029135031314095103
```

Observation

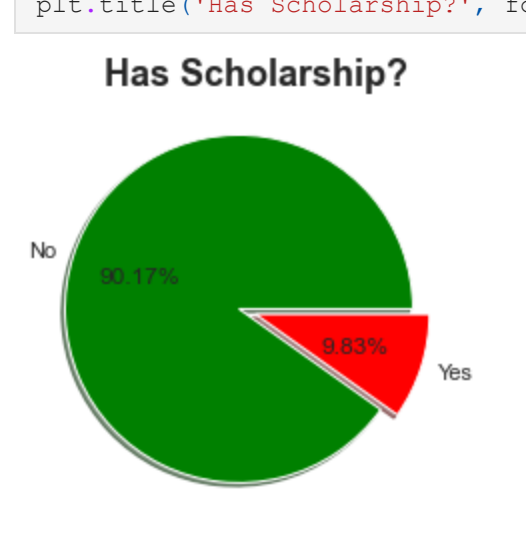
- The positive correlation shows that patients who have free healthcare are more likely to miss their appointments than those who are not.

OVERVIEW

```
#Check the the count of both categories of scholarship status of patients
print(df.query('scholarship == 0').scholarship.count())
print(df.query('scholarship == 1').scholarship.count())
```

```
99666
10861
```

```
# Plot a pie chart to show the counts
plt.pie(df.scholarship.value_counts(), labels=['No', 'Yes'], colors =['Green', 'Red'],
        explode=(0.1,0), shadow=True, radius=1.0, autopct = '%.2f%%' )
plt.title('Has Scholarship?', fontsize=18, fontweight='bold');
```

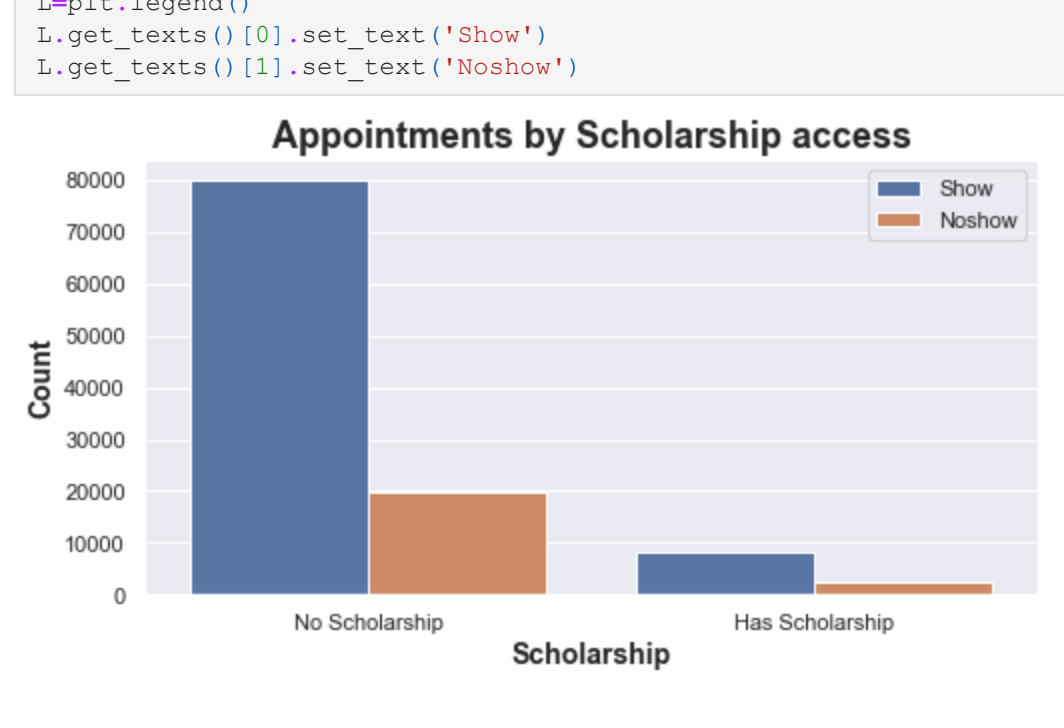


Observation

- Only about 10% of the patients have free healthcare access.

COMPARE APPOINTMENTS BY SCHOLARSHIP

```
#Plot a mutiple column chart to show the comparison betweeen scholarship status and p
ax = sns.countplot(x="scholarship", hue="absence", data=df)
plt.xticks([0, 1], ['No Scholarship', 'Has Scholarship'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by Scholarship access',  fontsize=18, fontweight='bold')
plt.xlabel('Scholarship', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```



INSIGHTS

> **There isn't a strong but there is a positive correlation betwen Appointment and Hypertension Status.**
>
> **Patients who are on scholarships are more likely to miss appointment compared to those who are not**

B.) SMS RECIEVAL

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the distribution.
>
> I will then compare their responsiveness to appointments by recieval of reminder SMS.

CORELATION COEFICIENT

```
#Calculate the correlation coeficient for absence and sms recieval
df['absence'].corr(df['sms_received'])
```

```
0.12643065757314462
```

Observation

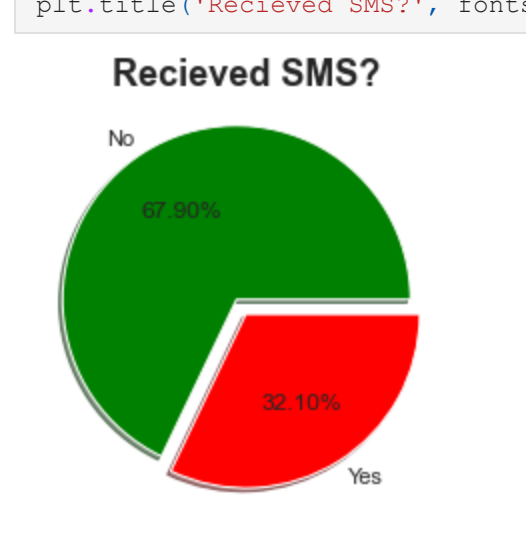
- There is a positive and quite strong correlation that shows that patients who recieve reminder more likely to miss their appointments than those who are not.
- This looks abnormal and I will explore deeper.

OVERVIEW

```
#Check the the count of both categories of sms recieval by patients
print(df.query('sms_received == 0').sms_received.count())
print(df.query('sms_received == 1').sms_received.count())
```

```
75045
35482
```

```
#Plot a pie chart to show the counts
plt.pie(df.sms_received.value_counts(), labels=['No', 'Yes'], colors =['Green', 'Red'
        explode=(0.1,0), shadow=True, radius=1.0, autopct = '%.2f%%' )
plt.title('Recieved SMS?', fontsize=18, fontweight='bold');
```

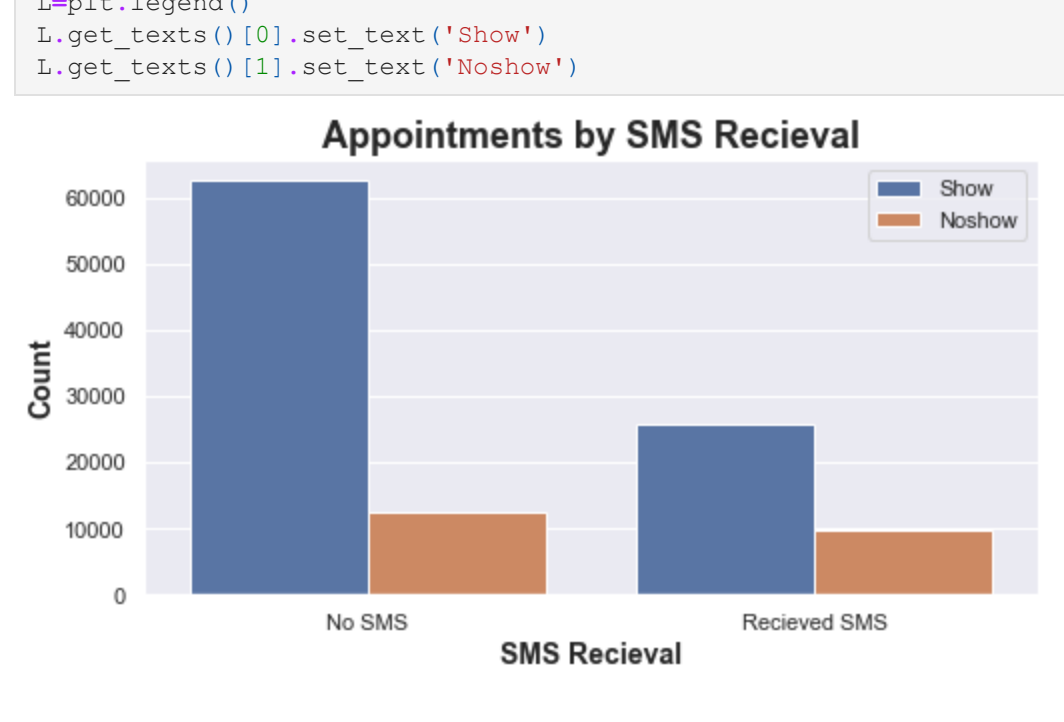


Observation

- About 32% of those who scheduled appointment recieved an SMS.

COMPARE APPOINTMENTS BY SMS RECIEVAL

```
#Plot a mutiple column chart to show the comparison betweeen SMS recieval and presence
ax = sns.countplot(x="sms_received", hue="absence", data=df)
plt.xticks([0, 1], ['No SMS', 'Recieved SMS'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by SMS Recieval',  fontsize=18, fontweight='bold')
plt.xlabel('SMS Recieval', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```



INSIGHTS

> **There is a quite relevant level of strength of positive correlation between Appointment and SMS recieval Status.**
>
> **This shows that patients who recieved SMS are less likely to miss appointment compared to those who do not, this doesn't sound right but there will be further exploration with the samedate variable**

CHECK IF SEPERATE DAY IS RESPONSIBLE FOR SMS SENDING

```
# Check if the counts correlate
df.groupby('sms_received')['samedate'].value_counts()
```

```
sms_received  samedate
0             1           38563
              0           36482
1             0           35482
Name: samedate, dtype: int64
```

> **This shows that SMS are not sent to those who have appointment on the Samedate**
>
> **This is therefore established that SMS sending is not associated with non responsiveness, it is just not a sufficient method to bring patients on another date**

C.) SAMEDATE

> I will first calculate the correlation coefficient.
>
> I will first have an overview of the the distribution.
>
> I will then compare their responsiveness to appointments that are on the sameday.

CORELATION COEFICIENT

```
#Calculate the correlation coeficient for absence and sameday appointment
df['absence'].corr(df['samedate'])
```

```
-0.2834864064340587
```

Observation

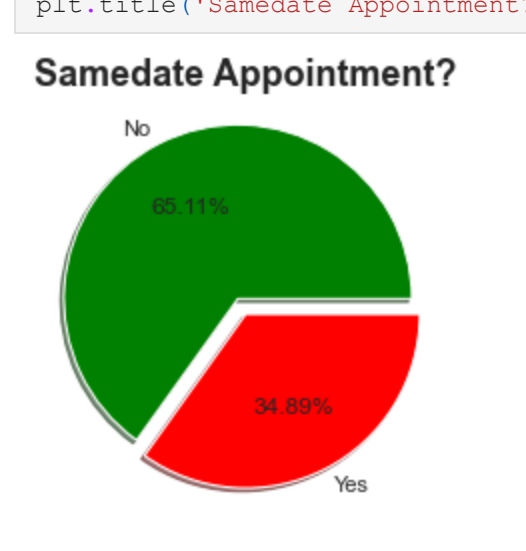
- There is a negative correlation of considerable strength which means that people are less likely to show up for appoointments that are on the same day they scheduled.

OVERVIEW

```
#Check the the counts of patients that are scheduled for same and different days.
print(df.query('samedate == 0').samedate.count())
print(df.query('samedate == 1').samedate.count())
```

```
71964
38563
```

```
#Plot a pie chart to show the counts
plt.pie(df.samedate.value_counts(), labels=['No', 'Yes'], colors =['Green', 'Red'],
        explode=(0.1,0), shadow=True, radius=1.0, autopct = '%.2f%%' )
plt.title('Samedate Appointment?', fontsize=18, fontweight='bold');
```

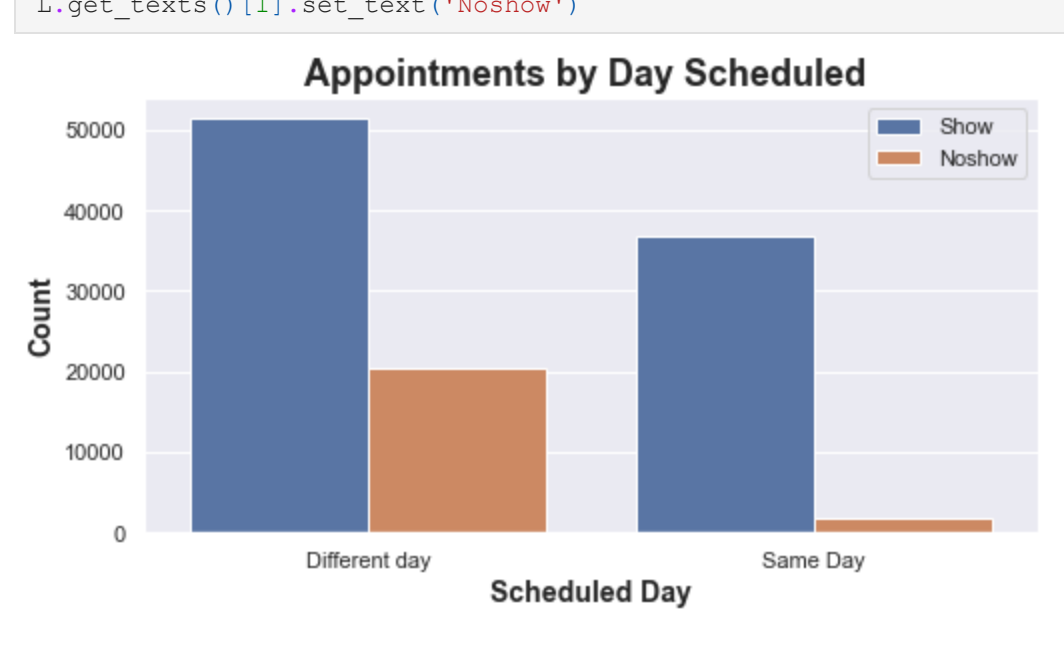


Observation

- About 35% of patients have appointment scheduled for the same day.

COMPARE APPOINTMENTS BY SAMEDATE APPOINTMENT

```
#Plot a mutiple column chart to show the comparison betweeen scheduled date and prese
ax = sns.countplot(x="samedate", hue="absence", data=df)
plt.xticks([0, 1], ['Different day', 'Same Day'])
sns.set(style='white')
sns.set(rc={"figure.figsize":(8,4)})
plt.title('Appointments by Day Scheduled ',  fontsize=18, fontweight='bold')
plt.xlabel('Scheduled Day', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
L=plt.legend()
L.get_texts()[0].set_text('Show')
L.get_texts()[1].set_text('Noshow')
```

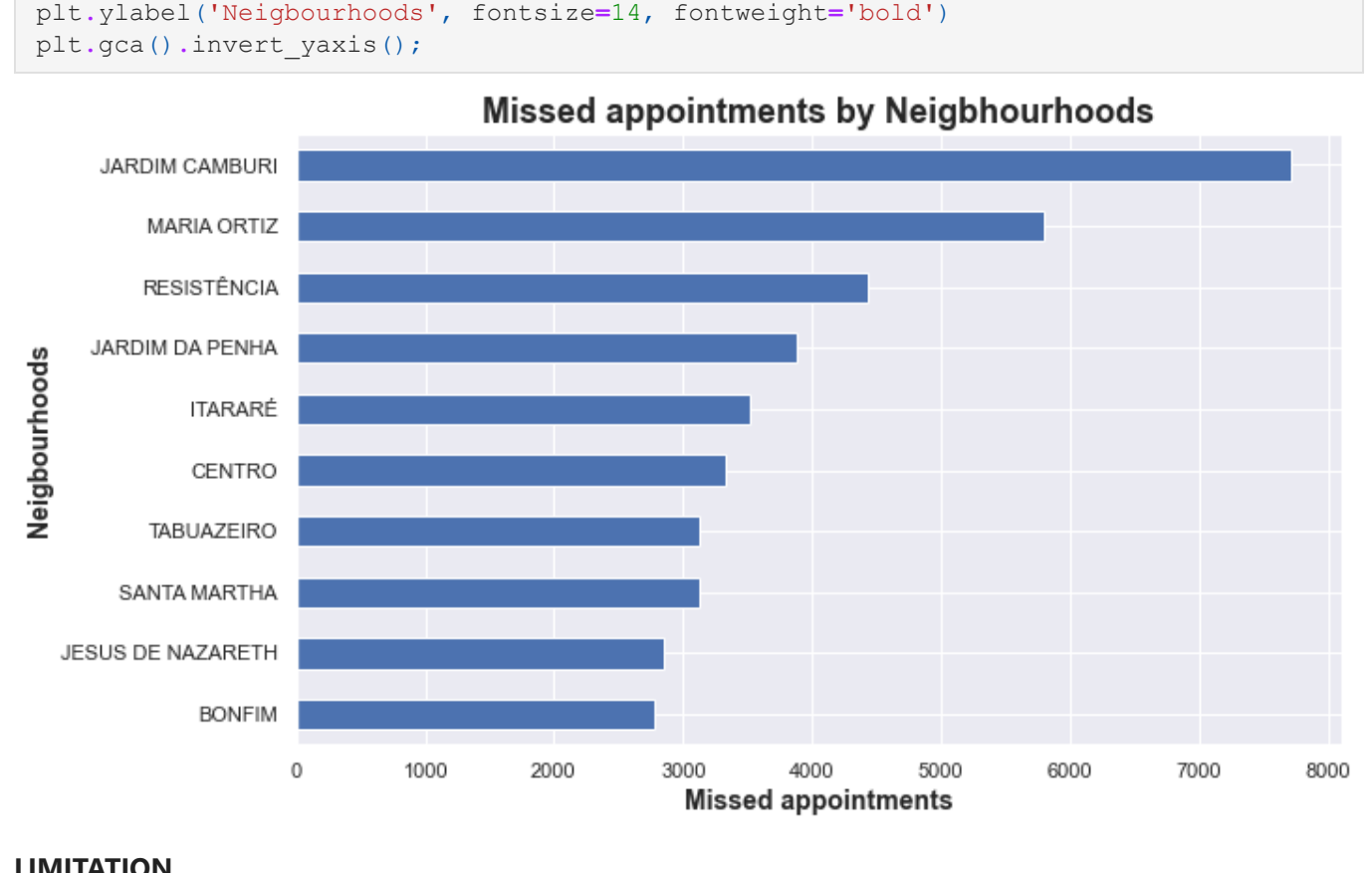


INSIGHTS

> **There is a negative correlation of considerable strength which means that people are less likely to show up for appoointments that are on the same day they scheduled.**

D.) NEIGHBHOURHOOD

```
 #Lets see the top 10 neighbhouirhoold that miss their appointments.
df.neighbourhood.value_counts()[:10].plot(kind='barh', figsize=(10,6))
plt.title('Missed appointments by Neighbhourhoods',  fontsize=18, fontweight='bold')
plt.xlabel('Missed appointments', fontsize=14, fontweight='bold')
plt.ylabel('Neigbourhoods', fontsize=14, fontweight='bold')
plt.gca().invert_yaxis();
```



LIMITATION

- There is no sufficient background knoledge to explore reasons why these neigbhourhood have a lot of missed appointments

CONCLUSIONS

- The analysis was to know What factors are important for us to know in order to predict if a patient will show up for their scheduled appointment.
- Generally Older and very young people are more likely to show up for medical appointments while the gender has negligible association with the absence of the patients at appointments.
- Patients that have hypertension, diabetes and are handicapped are less likely to miss appointment comoared to those who dont have, even though there is a weak correlation but a considerable one enough to suspect an association.
- There is a negligible association between alcohol taking and presence at medical appointment.
- Patients that are on scholarship are more likely to miss appointment compared to those who don't have, even though there is a weak correlation but a considerable one enough to suspect an association.
- Patients that are scheduled on the same date are more likely to miss appointment, it is also important to note that they don't recieve SMS.

LIMITATIONS

- There is not enough background data to explore the neigbhourhoods or distance from medical center.