```
import pandas as pd
import numpy as np
import datetime
from time import strftime
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

```
base_data = pd.read_csv('/content/Data_HEALTH.csv')
```

```
base_data.shape
```

```
(110527, 14)
```

```
base_data.head()
```

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbou
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARD F
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARD F
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MA
3	8.679512e+11	5642828	F	2016-04-	2016-04-	8	PONT

```
base_data.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
```

```
base_data.isnull().sum()
```

```
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
```

```
#modifying the date and time into standard form
base_data['ScheduledDay'] = pd.to_datetime(base_data['ScheduledDay']).dt.date.astype('datetime6
base_data['AppointmentDay'] = pd.to_datetime(base_data['AppointmentDay']).dt.date.astype('datet
```

```
base_data.head()
```

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	Diabetes	Alcoh
0	2.987250e+13	5642903	F	2016-04-29	2016-04-29	62	JARDIM DA PENHA	0	1	0	
1	5.589978e+14	5642503	M	2016-04-29	2016-04-29	56	JARDIM DA PENHA	0	0	0	
2	4.262962e+12	5642549	F	2016-04-29	2016-04-29	62	MATA DA	0	0	0	

```
#for the schedule day and appointment day storing the weekdays only into a variable

# 5 is Saturday, 6 is Sunday

base_data['sch_weekday'] = base_data['ScheduledDay'].dt.dayofweek


base_data['app_weekday'] = base_data['AppointmentDay'].dt.dayofweek
```

```
base_data['sch_weekday'].value_counts()
```

```
1    26168
2    24262
0    23085
4    18915
3    18073
5       24
Name: sch_weekday, dtype: int64
```

```
base_data['app_weekday'].value_counts()
```

```
2    25867
1    25640
0    22715
4    19019
3    17247
5       39
Name: app_weekday, dtype: int64
```

```
base_data.columns
```

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

```
#changing the name of some cloumns
base_data= base_data.rename(columns={'Hipertension': 'Hypertension', 'Handcap': 'Handicap', 'SM
```

```
# dropping some columns which have no significance
base_data.drop(['PatientId', 'AppointmentID', 'Neighbourhood'], axis=1, inplace=True)
```

```
base_data.shape
```

```
(110527, 13)
```

```
base_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Gender          110527 non-null  object
 1   ScheduledDay    110527 non-null  datetime64[ns]
 2   AppointmentDay  110527 non-null  datetime64[ns]
 3   Age             110527 non-null  int64
 4   Scholarship     110527 non-null  int64
 5   Hypertension    110527 non-null  int64
 6   Diabetes        110527 non-null  int64
 7   Alcoholism      110527 non-null  int64
 8   Handicap        110527 non-null  int64
 9   SMSReceived     110527 non-null  int64
 10  NoShow          110527 non-null  object
 11  sch_weekday     110527 non-null  int64
 12  app_weekday     110527 non-null  int64
dtypes: datetime64[ns](2), int64(9), object(2)
memory usage: 11.0+ MB
```

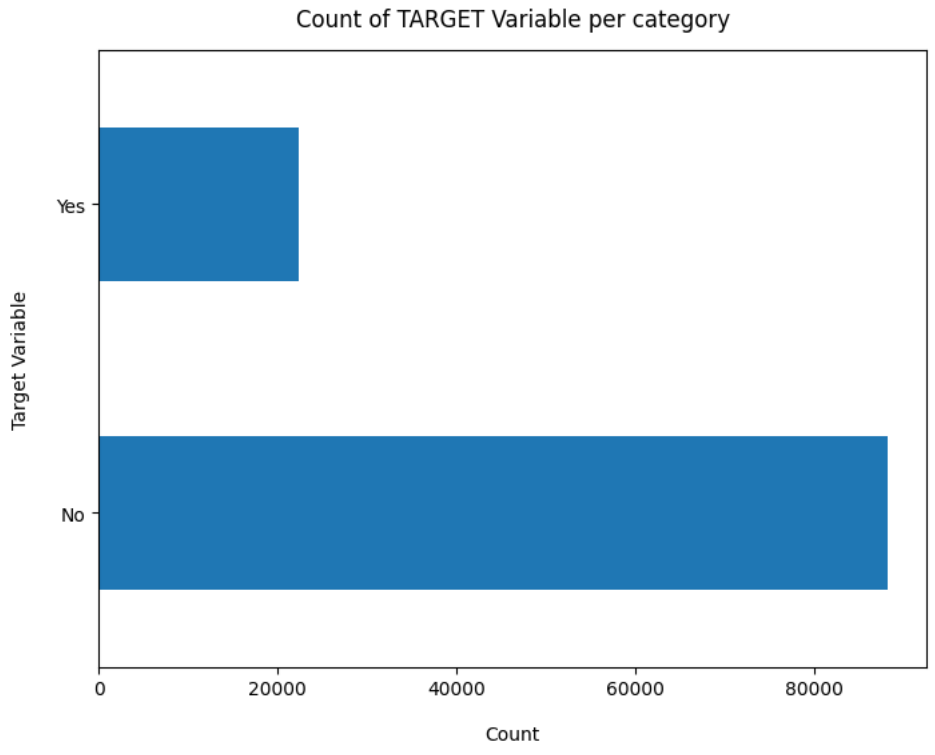
```
base_data.describe()
```

	Age	Scholarship	Hypertension	Diabetes	Alcoholism	Handicap	SMSReceived	sch_weekday	app_wee
count	110527.000000	110527.000000	110527.000000	110527.000000	110527.000000	110527.000000	110527.000000	110527.000000	110527.00
mean	37.088874	0.098266	0.197246	0.071865	0.030400	0.022248	0.321026	1.851955	1.85
std	23.110205	0.297675	0.397921	0.258265	0.171686	0.161543	0.466873	1.378520	1.37
min	-1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	18.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.00
50%	37.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	2.00
75%	55.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	3.000000	3.00
max	115.000000	1.000000	1.000000	1.000000	1.000000	4.000000	1.000000	5.000000	5.00

```
key = 'NoShow'
key = key.strip()  # Remove leading and trailing whitespaces
base_data[key].value_counts().plot(kind='barh', figsize=(8, 6))

plt.xlabel("Count", labelpad=14)
plt.ylabel("Target Variable", labelpad=14)
plt.title("Count of TARGET Variable per category", y=1.02)
```

```
Text(0.5, 1.02, 'Count of TARGET Variable per category')
```



```
print(base_data.columns)
```

```
Index(['Gender', 'ScheduledDay', 'AppointmentDay', 'Age', 'Scholarship',
       'Hypertension', 'Diabetes', 'Alcoholism', 'Handicap', 'SMSReceived',
       'NoShow', 'sch_weekday', 'app_weekday'],
      dtype='object')
```

```
# calculating the % of appointments or not
100*base_data['NoShow'].value_counts()/len(base_data['NoShow'])
```

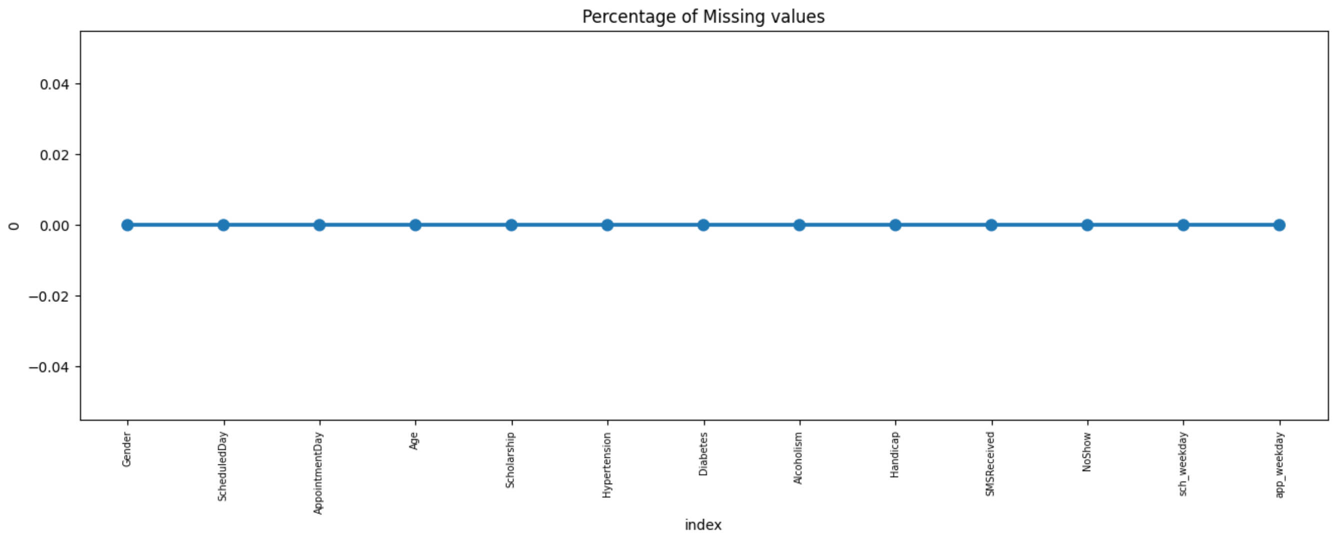
```
No     79.806744
Yes    20.193256
Name: NoShow, dtype: float64
```

```
base_data['NoShow'].value_counts()
```

```
No     88208
Yes    22319
Name: NoShow, dtype: int64
```

```
missing = pd.DataFrame((base_data.isnull().sum()) * 100 / base_data.shape[0]).reset_index()
plt.figure(figsize=(16, 5))
ax = sns.pointplot(x='index', y=0, data=missing)
plt.xticks(rotation=90, fontsize=7)
plt.title("Percentage of Missing values")
```

```
Text(0.5, 1.0, 'Percentage of Missing values')
```



Missing Data - Initial Intuition

Here, we don't have any missing data.

General Thumb Rules:

- For features with less missing values- can use regression to predict the missing values or fill with the mean of the values present, depending on the feature. For features with very high number of missing values- it is better to drop those columns as they give very less insight on analysis.
- As there's no thumb rule on what criteria do we delete the columns with high number of missing values, but generally you can delete the columns, if you have more than 30-40% of missing values.

Data Cleaning

Create a copy of base data for manupulation & processing

```
new_data = base_data.copy()
```

```
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Gender          110527 non-null  object
 1   ScheduledDay    110527 non-null  datetime64[ns]
 2   AppointmentDay  110527 non-null  datetime64[ns]
 3   Age             110527 non-null  int64
 4   Scholarship     110527 non-null  int64
 5   Hypertension    110527 non-null  int64
 6   Diabetes        110527 non-null  int64
 7   Alcoholism      110527 non-null  int64
 8   Handicap        110527 non-null  int64
 9   SMSReceived     110527 non-null  int64
 10  NoShow          110527 non-null  object
 11  sch_weekday     110527 non-null  int64
 12  app_weekday     110527 non-null  int64
dtypes: datetime64[ns](2), int64(9), object(2)
memory usage: 11.0+ MB
```

As we don't have any null records, there's no data cleaning required

```
# Get the max tenure
print(base_data['Age'].max())
```

```
115
```

```
# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 20) for i in range(1, 118, 20)]

base_data['Age_group'] = pd.cut(base_data.Age, range(1, 130, 20), right=False, labels=labels)
base_data.drop(['Age'], axis=1, inplace=True)
```

Data Exploration

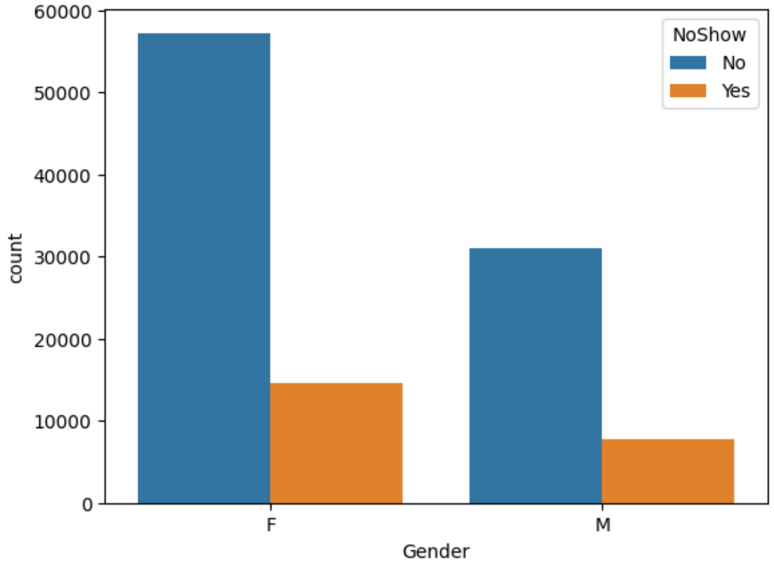
```
list(base_data.columns)
```

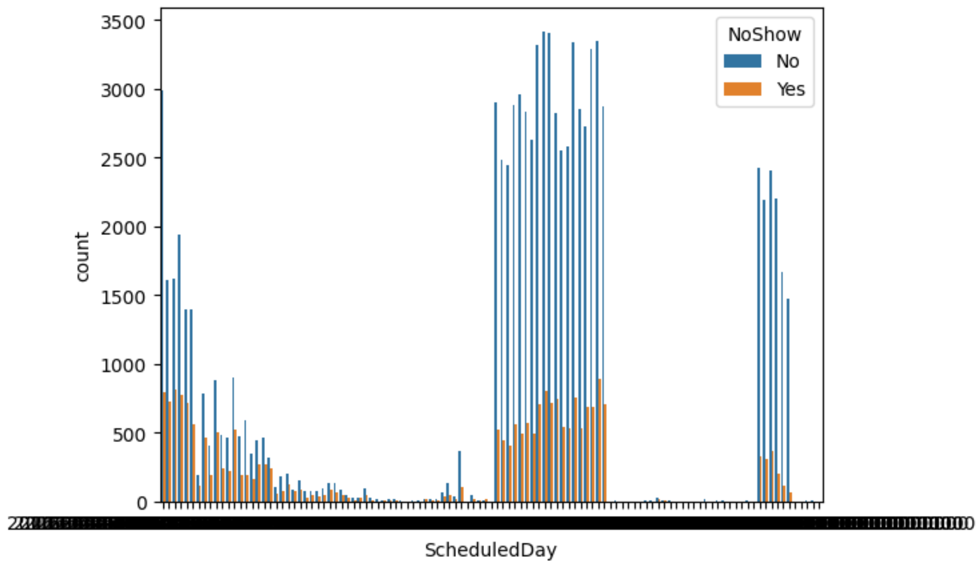
```
['Gender',
 'ScheduledDay',
 'AppointmentDay',
 'Scholarship',
 'Hypertension',
 'Diabetes',
 'Alcoholism',
 'Handicap',
 'SMSReceived',
 'NoShow',
 'sch_weekday',
 'app_weekday',
 'Age_group']
```

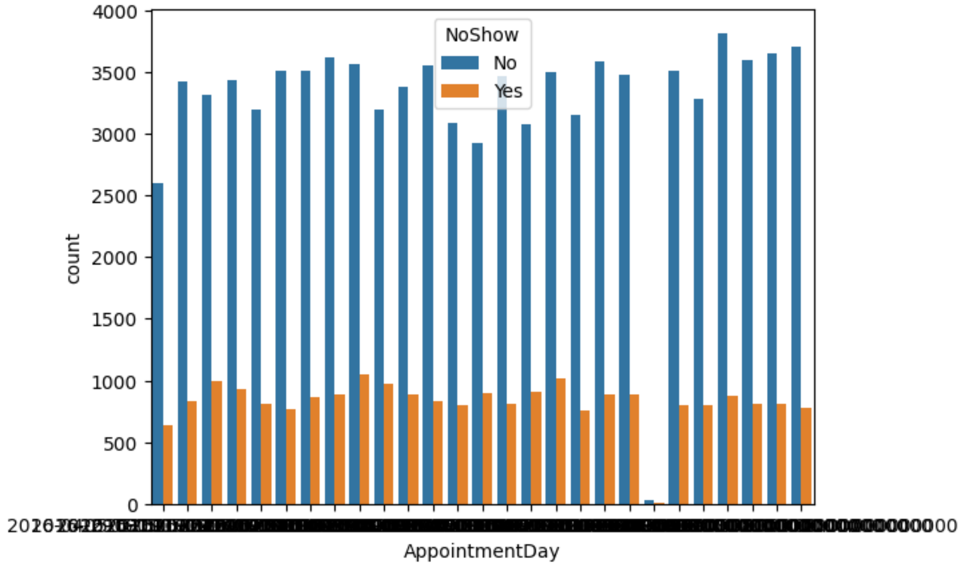
```
#having a loook into the values of count of each columns and there count in respect to NoShow co
for i, predictor in enumerate(base_data.drop(columns=['NoShow'])):
    print('-'*10,predictor,'-'*10)
    print(base_data[predictor].value_counts())
    plt.figure(i)
    sns.countplot(data=base_data, x=predictor, hue='NoShow')
```

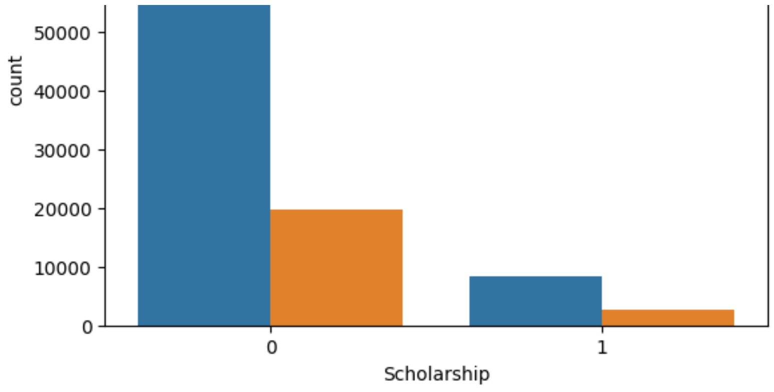
```
---------- Gender ----------
F    71840
M    38687
Name: Gender, dtype: int64
---------- ScheduledDay ----------
2016-05-03    4238
2016-05-02    4216
2016-05-16    4120
2016-05-05    4095
2016-05-10    4024
              ...
2016-04-16       1
2016-01-28       1
2015-11-10       1
2016-03-19       1
2016-03-05       1
Name: ScheduledDay, Length: 111, dtype: int64
---------- AppointmentDay ----------
2016-06-06    4692
2016-05-16    4613
2016-05-09    4520
2016-05-30    4514
2016-06-08    4479
2016-05-11    4474
2016-06-01    4464
2016-06-07    4416
2016-05-12    4394
2016-05-02    4376
2016-05-18    4373
2016-05-17    4372
2016-06-02    4310
2016-05-10    4308
2016-05-31    4279
2016-05-05    4273
2016-05-19    4270
2016-05-03    4256
2016-05-04    4168
2016-06-03    4090
2016-05-24    4009
2016-05-13    3987
2016-05-25    3909
2016-05-06    3879
2016-05-20    3828
2016-04-29    3235
2016-05-14      39
Name: AppointmentDay, dtype: int64
---------- Scholarship ----------
0    99666
1    10861
Name: Scholarship, dtype: int64
---------- Hypertension ----------
0    88726
1    21801
Name: Hypertension, dtype: int64
---------- Diabetes ----------
0    102584
1      7943
Name: Diabetes, dtype: int64
---------- Alcoholism ----------
0    107167
1      3360
Name: Alcoholism, dtype: int64
---------- Handicap ----------
0    108286
1      2042
2       183
3        13
4         3
Name: Handicap, dtype: int64
---------- SMSReceived ----------
0    75045
1    35482
Name: SMSReceived, dtype: int64
---------- sch_weekday ----------
1    26168
2    24262
0    23085
4    18915
3    18073
5       24
Name: sch_weekday, dtype: int64
---------- app_weekday ----------
2    25867
1    25640
0    22715
4    19019
3    17247
5       39
Name: app_weekday, dtype: int64
---------- Age_group ----------
```

```
41 - 61      30081
21 - 41      28835
1 - 21       28309
61 - 81      16910
81 - 101      2845
101 - 121        7
Name: Age_group, dtype: int64
```

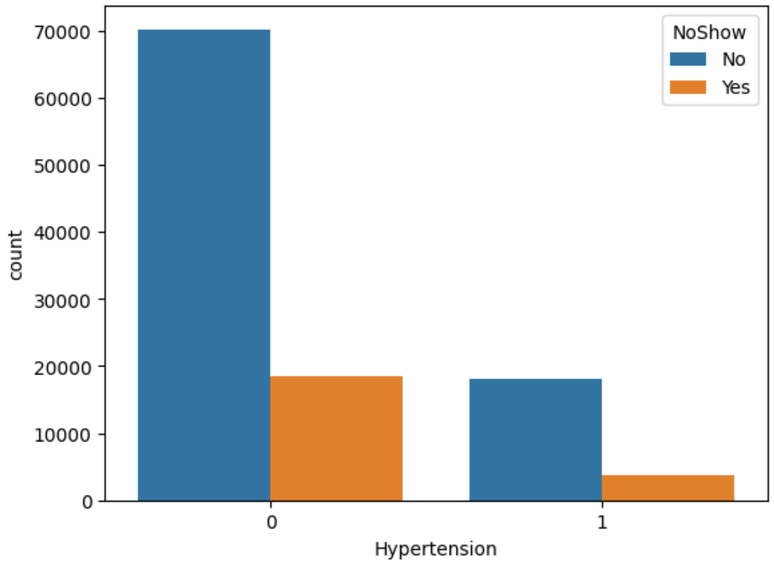




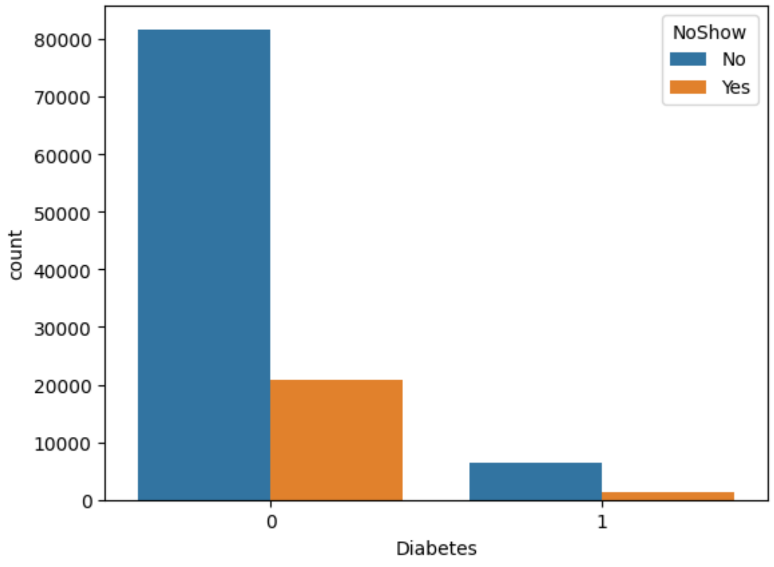




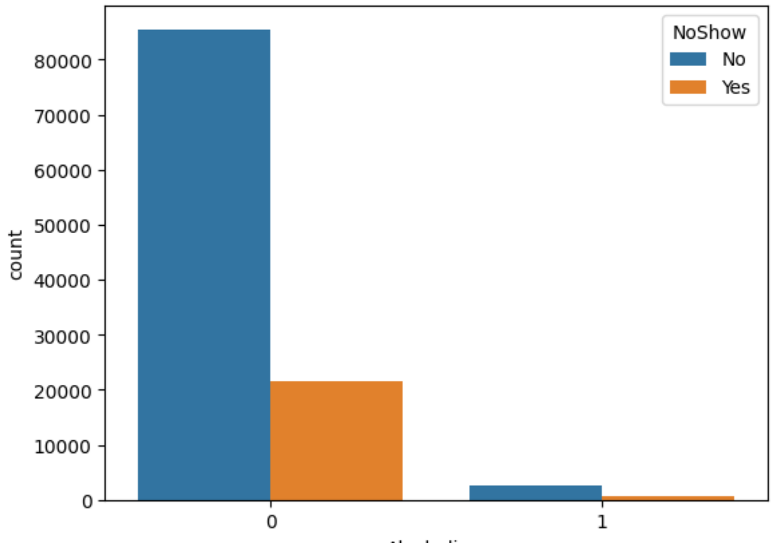
count
50000
40000
30000
20000
10000
0
0
1
Scholarship



NoShow
No
Yes
count
70000
60000
50000
40000
30000
20000
10000
0
0
1
Hypertension

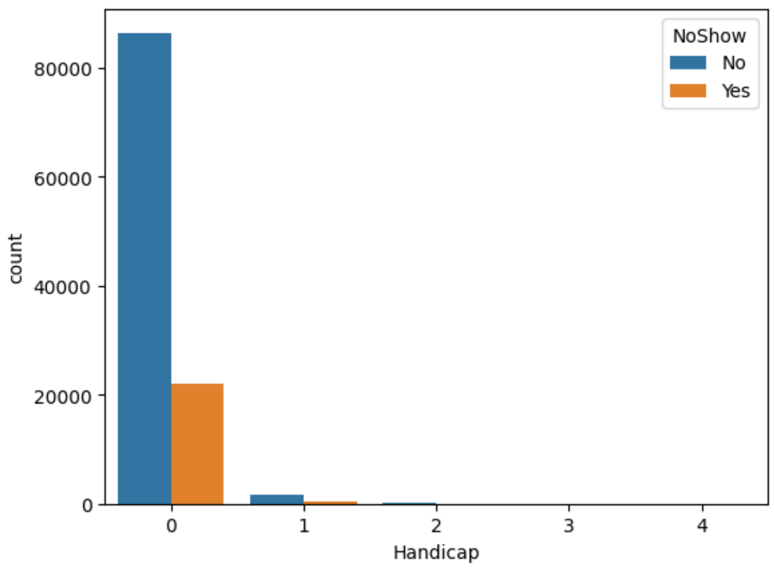


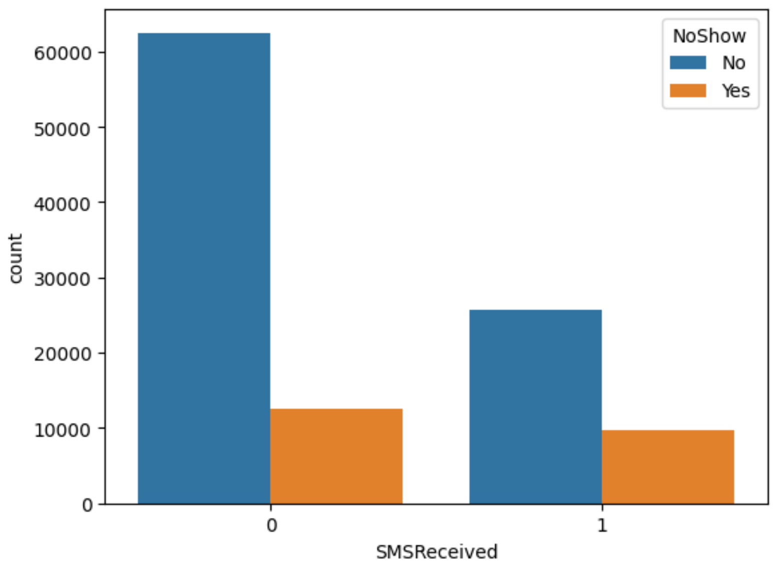
NoShow
No
Yes
count
80000
70000
60000
50000
40000
30000
20000
10000
0
0
1
Diabetes

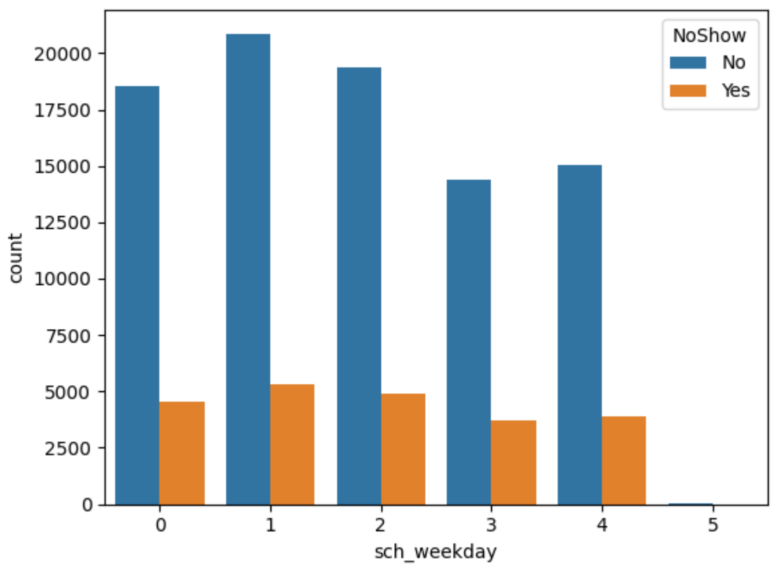


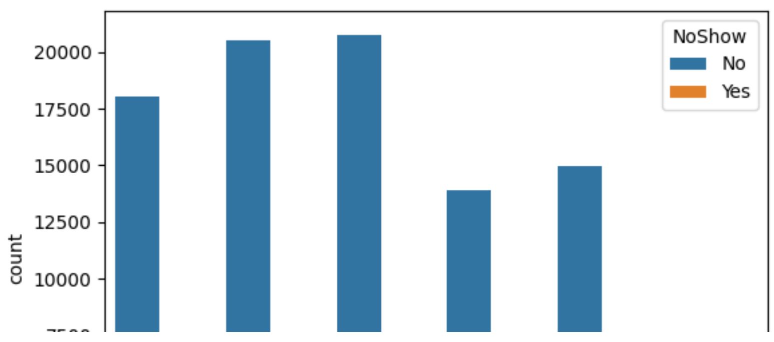
NoShow
No
Yes
count
80000
70000
60000
50000
40000
30000
20000
10000
0
0
1

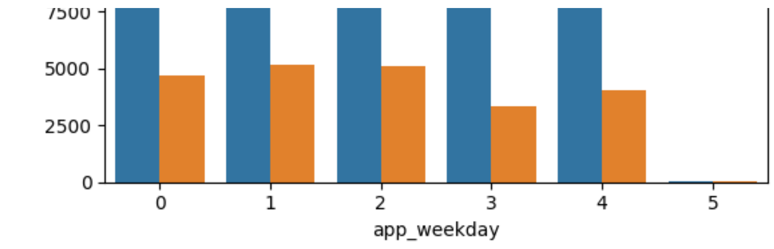
Alcoholism

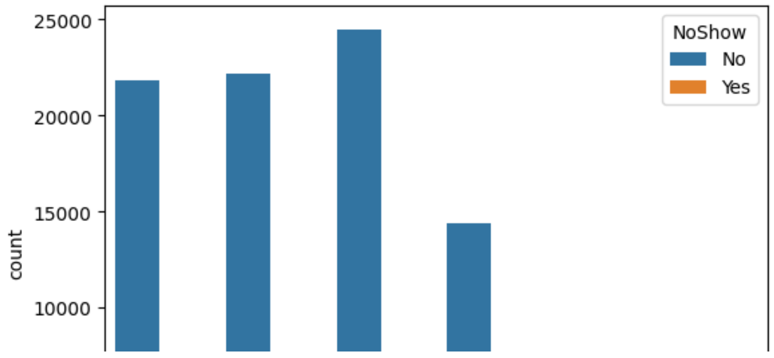

NoShow
No
Yes
count
80000
60000
40000
20000
0
0
1
2
3
4
Handicap


NoShow
No
Yes
count
60000
50000
40000
30000
20000
10000
0
0
1
SMSReceived


NoShow
No
Yes
count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
sch_weekday


NoShow
No
Yes
count
20000
17500
15000
12500
10000





```
base_data['NoShow'] = np.where(base_data.NoShow == 'Yes',1,0)
```

```
base_data.NoShow.value_counts()
```

```
0    88208
1    22319
Name: NoShow, dtype: int64
```

Convert all the categorical variables into dummy variables

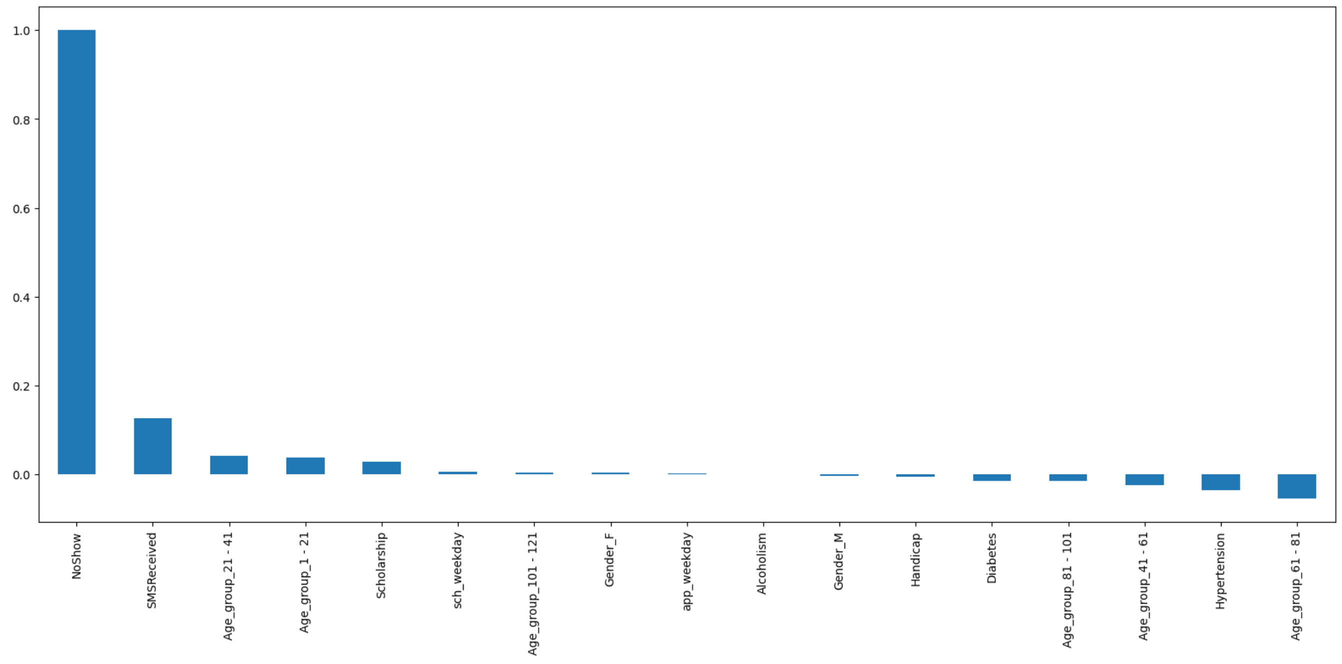
```
base_data_dummies = pd.get_dummies(base_data)
base_data_dummies.head()
```

	ScheduledDay	AppointmentDay	Scholarship	Hypertension	Diabetes	Alcoholism	Handicap	SMSReceived	NoShow	sch_weekday	app_we
0	2016-04-29	2016-04-29	0	1	0	0	0	0	0	4	
1	2016-04-29	2016-04-29	0	0	0	0	0	0	0	4	
2	2016-04-29	2016-04-29	0	0	0	0	0	0	0	4	
3	2016-04-29	2016-04-29	0	0	0	0	0	0	0	4	
4	2016-04-29	2016-04-29	0	1	1	0	0	0	0	4	

Build a corelation of all predictors with 'NoShow'

```
plt.figure(figsize=(20,8))
base_data_dummies.corr()['NoShow'].sort_values(ascending = False).plot(kind='bar')
```

```
<ipython-input-61-2af5ffecb2b2>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  base_data_dummies.corr()['NoShow'].sort_values(ascending = False).plot(kind='bar')
<Axes: >
```



```
plt.figure(figsize=(12,12))
sns.heatmap(base_data_dummies.corr(), cmap="Paired")
```