HealthCare Capstone Project

A person makes a doctor appointment, receives all the instructions and no-show. Who to blame? 300k medical appointments and its 15 variables (characteristics) of each. The most important one if the patient show-up or no-show the appointment. Variable names are self-explanatory

Problem Statement :

Predict someone to no-show an appointment.

Importing the Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
pwd
```

Out[2]:

```
'/Users/ds/Desktop/Data_Scientist/Capstone_Project/Data-Science-Caps
tone-Projects-master/Project 2'
```

Import the healthcare dataset

In [3]:

```
df = pd.read_csv('train.csv')
```

Descriptive Statistics

In [4]:

```
df.head()
```

Out[4]:

	Age	Gender	AppointmentRegistration	ApointmentData	DayOfTheWeek	Status	Diabetes
0	38	F	2015-10-20T08:33:56Z	2015-10-23T00:00:00Z	Friday	No-Show	0
1	56	F	2014-02-03T10:05:26Z	2014-02-20T00:00:00Z	Thursday	No-Show	1
2	27	F	2014-04-29T07:57:32Z	2014-05-20T00:00:00Z	Tuesday	Show-Up	0
3	24	M	2014-04-02T13:53:37Z	2014-05-06T00:00:00Z	Tuesday	Show-Up	0
4	48	F	2014-01-07T10:07:17Z	2014-01-30T00:00:00Z	Thursday	Show-Up	0

In [5]:

```
df.columns
```

Out[5]:

```
Index(['Age', 'Gender', 'AppointmentRegistration', 'ApointmentData',
       'DayOfTheWeek', 'Status', 'Diabetes', 'Alcoolism', 'HiperTens
ion',
       'Handcap', 'Smokes', 'Scholarship', 'Tuberculosis', 'Sms_Remi
nder',
       'AwaitingTime'],
      dtype='object')
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210000 entries, 0 to 209999
Data columns (total 15 columns):
Age                       210000 non-null int64
Gender                    210000 non-null object
AppointmentRegistration   210000 non-null object
ApointmentData            210000 non-null object
DayOfTheWeek              210000 non-null object
Status                    210000 non-null object
Diabetes                  210000 non-null int64
Alcoolism                 210000 non-null int64
HiperTension              210000 non-null int64
Handcap                   210000 non-null int64
Smokes                    210000 non-null int64
Scholarship               210000 non-null int64
Tuberculosis              210000 non-null int64
Sms_Reminder              210000 non-null int64
AwaitingTime              210000 non-null int64
dtypes: int64(10), object(5)
memory usage: 24.0+ MB
```

In [7]:

```
df.nunique()
```

Out[7]:

```
Age                         107
Gender                        2
AppointmentRegistration    207711
ApointmentData                533
DayOfTheWeek                  7
Status                        2
Diabetes                      2
Alcoolism                     2
HiperTension                  2
Handcap                       5
Smokes                        2
Scholarship                   2
Tuberculosis                  2
Sms_Reminder                  3
AwaitingTime                205
dtype: int64
```

In [8]:

```
df.describe()
```

Out[8]:

	Age	Diabetes	Alcoolism	HiperTension	Handcap	Sm
count	210000.000000	210000.000000	210000.000000	210000.000000	210000.000000	210000.0(
mean	37.761824	0.077290	0.024676	0.214862	0.020471	0.0
std	22.794334	0.267052	0.155137	0.410727	0.155854	0.22
min	-1.000000	0.000000	0.000000	0.000000	0.000000	0.0(
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.0(
50%	38.000000	0.000000	0.000000	0.000000	0.000000	0.0(
75%	56.000000	0.000000	0.000000	0.000000	0.000000	0.0(
max	113.000000	1.000000	1.000000	1.000000	4.000000	1.0(

In [15]:

```
df.dtypes
```

Out[15]:

```
Age                                    int64
Gender                                object
AppointmentRegistration    datetime64[ns, UTC]
ApointmentData                        object
DayOfTheWeek                          object
Status                                object
Diabetes                               int64
Alcoolism                              int64
HiperTension                           int64
Handcap                                int64
Smokes                                 int64
Scholarship                            int64
Tuberculosis                           int64
Sms_Reminder                           int64
AwaitingTime                           int64
dtype: object
```

Checking for null values in the dataset

In [9]:

```
df.isnull().sum(axis =0)
```

Out[9]:

```
Age                         0
Gender                      0
AppointmentRegistration     0
ApointmentData              0
DayOfTheWeek                0
Status                      0
Diabetes                    0
Alcoolism                   0
HiperTension                0
Handcap                     0
Smokes                      0
Scholarship                 0
Tuberculosis                0
Sms_Reminder                0
AwaitingTime                0
dtype: int64
```

In [10]:

```
df.Handcap.value_counts()
```

Out[10]:

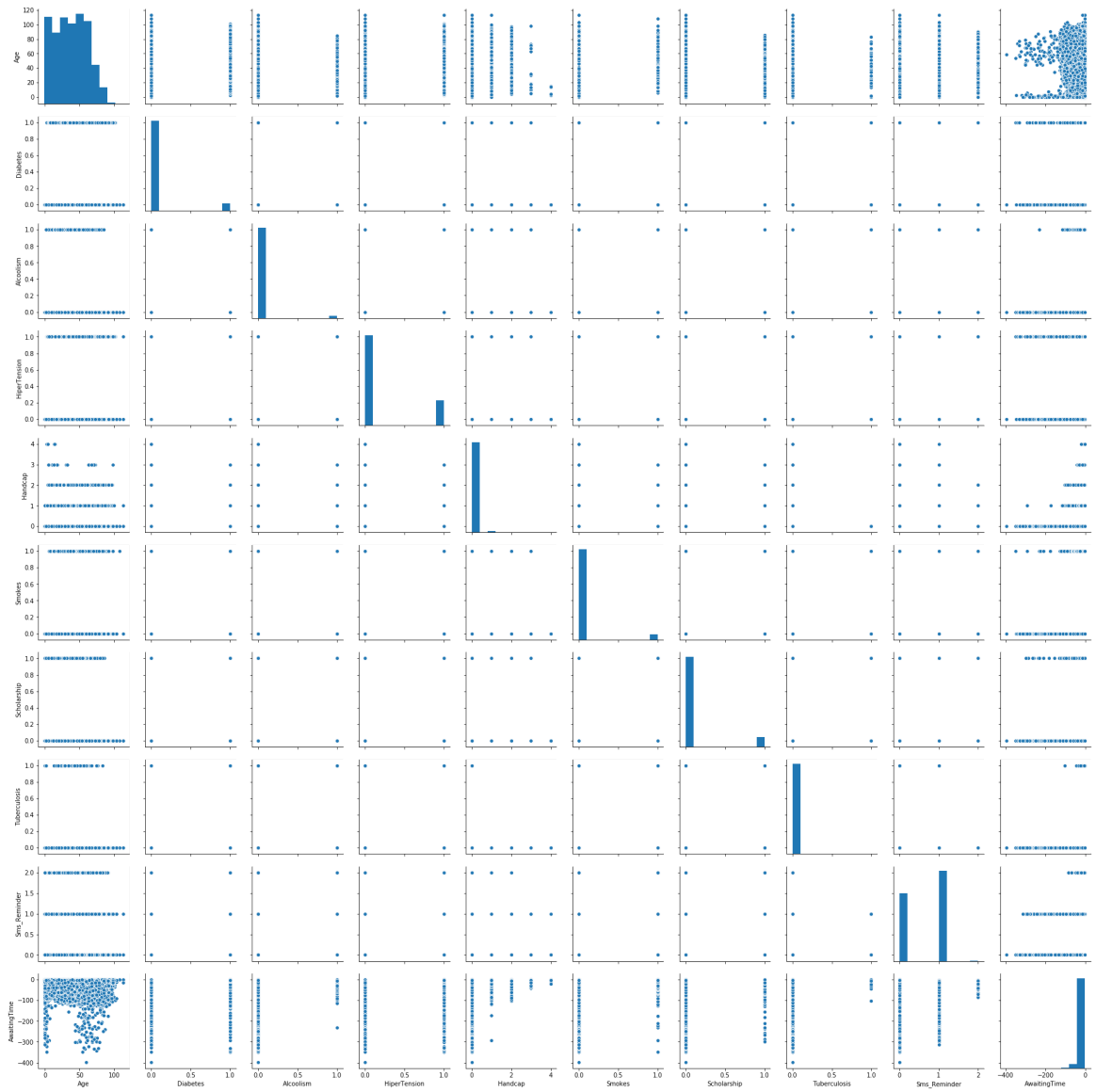
```
0    206096
1      3552
2       316
3        29
4         7
Name: Handcap, dtype: int64
```

In [11]:

```
sns.pairplot(df)
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x1a0c470e48>
```



In [14]:

```
df.head()
```

Out[14]:

	Age	Gender	AppointmentRegistration	ApointmentData	DayOfTheWeek	Status	Diabetes
0	38	F	2015-10-20 08:33:56+00:00	2015-10-23T00:00:00Z	Friday	No-Show	0
1	56	F	2014-02-03 10:05:26+00:00	2014-02-20T00:00:00Z	Thursday	No-Show	1
2	27	F	2014-04-29 07:57:32+00:00	2014-05-20T00:00:00Z	Tuesday	Show-Up	0
3	24	M	2014-04-02 13:53:37+00:00	2014-05-06T00:00:00Z	Tuesday	Show-Up	0
4	48	F	2014-01-07 10:07:17+00:00	2014-01-30T00:00:00Z	Thursday	Show-Up	0

Converting the date variables into datatime datatypes

In [16]:

```
df['AppointmentRegistration'] = pd.to_datetime(df['AppointmentRegistration'])
```

In [17]:

```
df['AppointmentData'] = pd.to_datetime(df['ApointmentData'])
```

In [18]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210000 entries, 0 to 209999
Data columns (total 16 columns):
Age                       210000 non-null int64
Gender                    210000 non-null object
AppointmentRegistration   210000 non-null datetime64[ns, UTC]
ApointmentData            210000 non-null object
DayOfTheWeek              210000 non-null object
Status                    210000 non-null object
Diabetes                  210000 non-null int64
Alcoolism                 210000 non-null int64
HiperTension              210000 non-null int64
Handcap                   210000 non-null int64
Smokes                    210000 non-null int64
Scholarship               210000 non-null int64
Tuberculosis              210000 non-null int64
Sms_Reminder              210000 non-null int64
AwaitingTime              210000 non-null int64
AppointmentData           210000 non-null datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](2), int64(10), object(4)
memory usage: 25.6+ MB
```

In [19]:

```
df['AwaitingTime'] = df['AwaitingTime'].abs()
```

In [20]:

```
df.head()
```

Out[20]:

	Age	Gender	AppointmentRegistration	ApointmentData	DayOfTheWeek	Status	Diabetes
0	38	F	2015-10-20 08:33:56+00:00	2015-10-23T00:00:00Z	Friday	No-Show	0
1	56	F	2014-02-03 10:05:26+00:00	2014-02-20T00:00:00Z	Thursday	No-Show	1
2	27	F	2014-04-29 07:57:32+00:00	2014-05-20T00:00:00Z	Tuesday	Show-Up	0
3	24	M	2014-04-02 13:53:37+00:00	2014-05-06T00:00:00Z	Tuesday	Show-Up	0
4	48	F	2014-01-07 10:07:17+00:00	2014-01-30T00:00:00Z	Thursday	Show-Up	0

In [21]:

```
df.drop(['ApointmentData'], axis = 1, inplace = True)
```

In [22]:

```
df.head()
```

Out[22]:

	Age	Gender	AppointmentRegistration	DayOfTheWeek	Status	Diabetes	Alcoolism	Hiperт
0	38	F	2015-10-20 08:33:56+00:00	Friday	No-Show	0	0	
1	56	F	2014-02-03 10:05:26+00:00	Thursday	No-Show	1	0	
2	27	F	2014-04-29 07:57:32+00:00	Tuesday	Show-Up	0	0	
3	24	M	2014-04-02 13:53:37+00:00	Tuesday	Show-Up	0	0	
4	48	F	2014-01-07 10:07:17+00:00	Thursday	Show-Up	0	0	

In [23]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210000 entries, 0 to 209999
Data columns (total 15 columns):
Age                       210000 non-null int64
Gender                    210000 non-null object
AppointmentRegistration   210000 non-null datetime64[ns, UTC]
DayOfTheWeek              210000 non-null object
Status                    210000 non-null object
Diabetes                  210000 non-null int64
Alcoolism                 210000 non-null int64
HiperTension              210000 non-null int64
Handcap                   210000 non-null int64
Smokes                    210000 non-null int64
Scholarship               210000 non-null int64
Tuberculosis              210000 non-null int64
Sms_Reminder              210000 non-null int64
AwaitingTime              210000 non-null int64
AppointmentData           210000 non-null datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](2), int64(10), object(3)
memory usage: 24.0+ MB
```

In [24]:

```
df['HourOftheDay'] = df['AppointmentRegistration'].dt.strftime('%H')
```

In [25]:

```
df.head()
```

Out[25]:

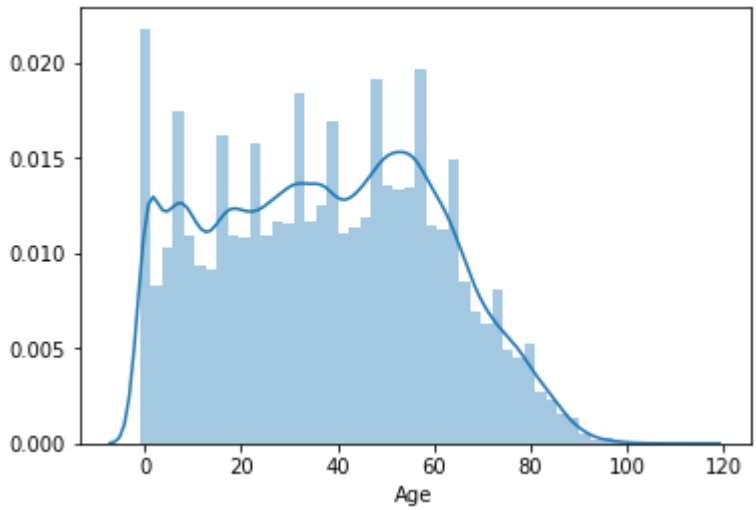
	Age	Gender	AppointmentRegistration	DayOfTheWeek	Status	Diabetes	Alcoolism	Hiper1
0	38	F	2015-10-20 08:33:56+00:00	Friday	No-Show	0	0	
1	56	F	2014-02-03 10:05:26+00:00	Thursday	No-Show	1	0	
2	27	F	2014-04-29 07:57:32+00:00	Tuesday	Show-Up	0	0	
3	24	M	2014-04-02 13:53:37+00:00	Tuesday	Show-Up	0	0	
4	48	F	2014-01-07 10:07:17+00:00	Thursday	Show-Up	0	0	

In [26]:

```
sns.distplot(df['Age'])
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1040eeb8>
```



In [27]:

```
df.Age.value_counts()
```

Out[27]:

```
0      7252
56     3317
51     3275
54     3250
52     3239
55     3209
50     3198
1      3174
49     3171
53     3152
57     3140
37     3067
48     3034
32     2983
31     2977
47     2959
58     2945
7      2905
36     2897
46     2872
30     2856
38     2847
61     2845
33     2838
35     2836
45     2816
27     2783
28     2782
59     2774
8      2761
       ...
77     1074
78     1059
79      931
80      824
81      739
82      701
83      588
84      574
85      519
86      429
87      301
88      241
89      200
90      188
91      120
92      110
93       60
96       50
95       42
94       34
97       25
98       24
99       10
101       9
100       6
102       4
103       3
113       3
```

```
-1          3
 108        1
Name: Age, Length: 107, dtype: int64
```

In [ ]:

```
```

In [28]:

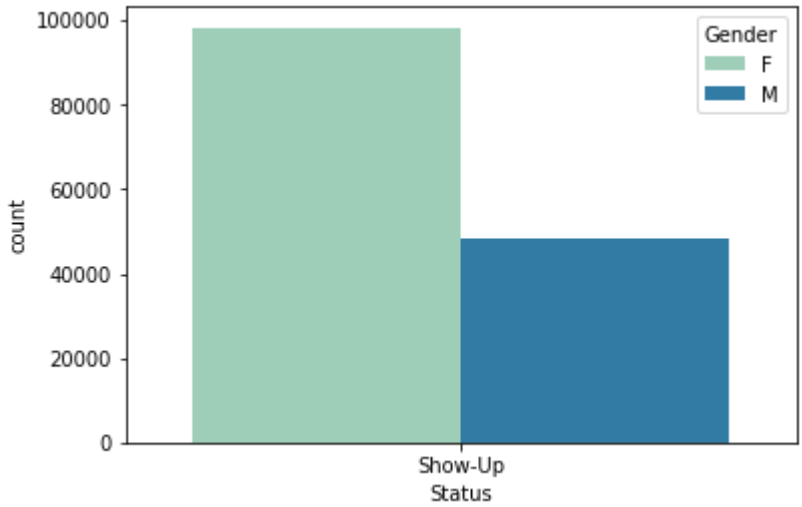
```
bar_df = df[df['Status']=='Show-Up']
```

In [41]:

```
sns.countplot(x='Status',hue ='Gender',data = bar_df,palette = 'YlGnBu')
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a21daff60>
```



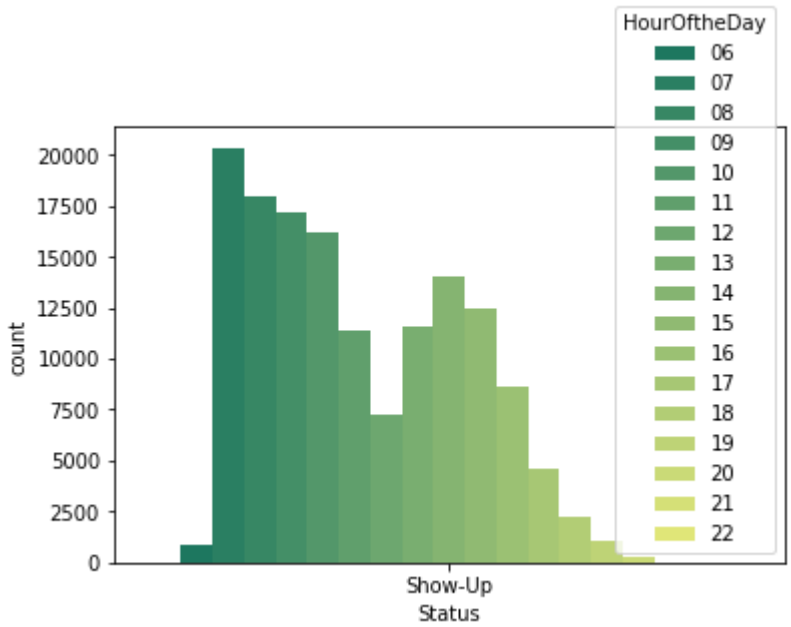
Above graph state that Female patient are more likely show up on appointment date alloted by doctor as compare to the male patient

In [30]:

```
sns.countplot(x='Status',hue ='HourOftheDay',data = bar_df,palette ='summer')
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1e3e5f98>
```



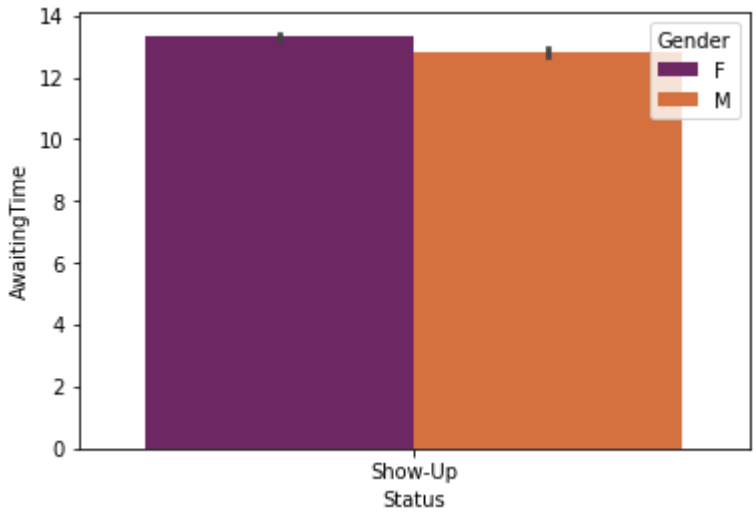
Above graph state that most patients likely to show up in the morning schedule

In [43]:

```
sns.barplot(x='Status',y='AwaitingTime',hue = 'Gender',data=bar_df,palette = 'inferno')
```

Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10f84f60>
```

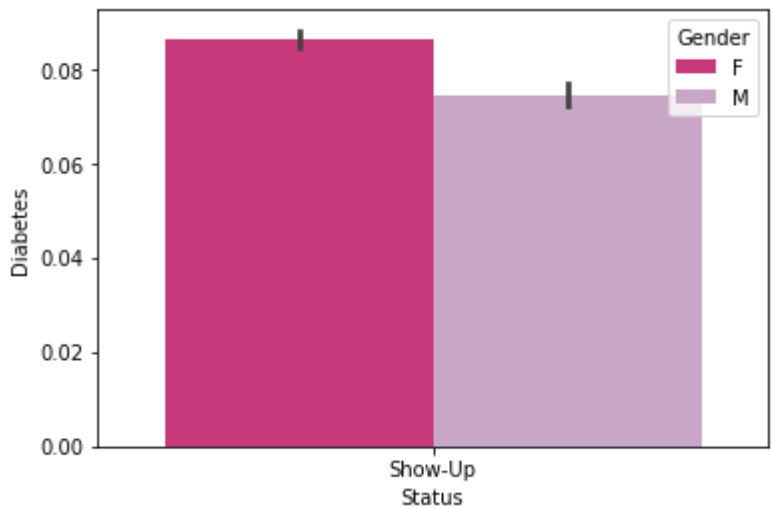


In [49]:

```
sns.barplot(x='Status',y='Diabetes',hue = 'Gender',data = bar_df,palette = 'PuRd_r')
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2ea31f28>
```

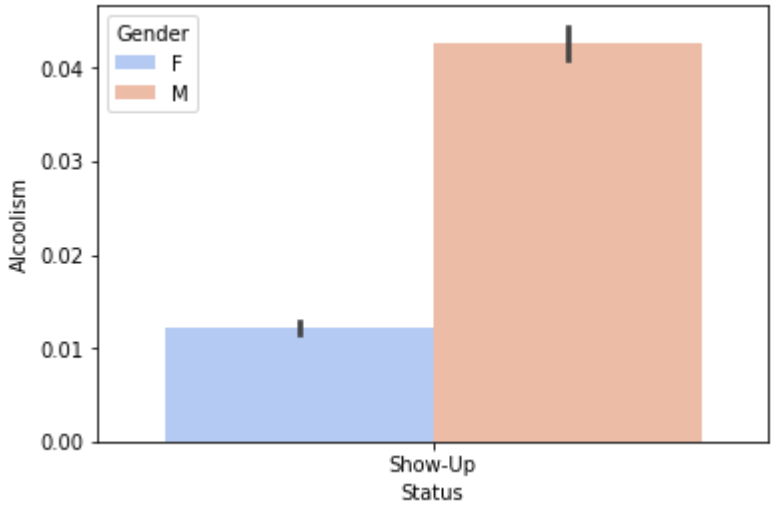


In [52]:

```
sns.barplot(x='Status',y='Alcoolism',hue ='Gender',data = bar_df,palette = 'cool warm')
```

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2ec60588>
```

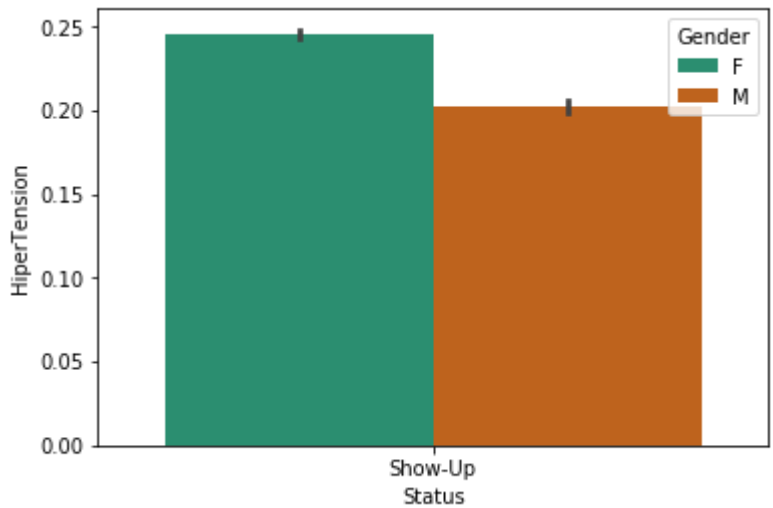


In [57]:

```
sns.barplot(x='Status',y='HiperTension',hue = 'Gender',data = bar_df,palette = 'Dark2')
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2e8a9ac8>
```



In [46]:

```
df.columns
```

Out[46]:

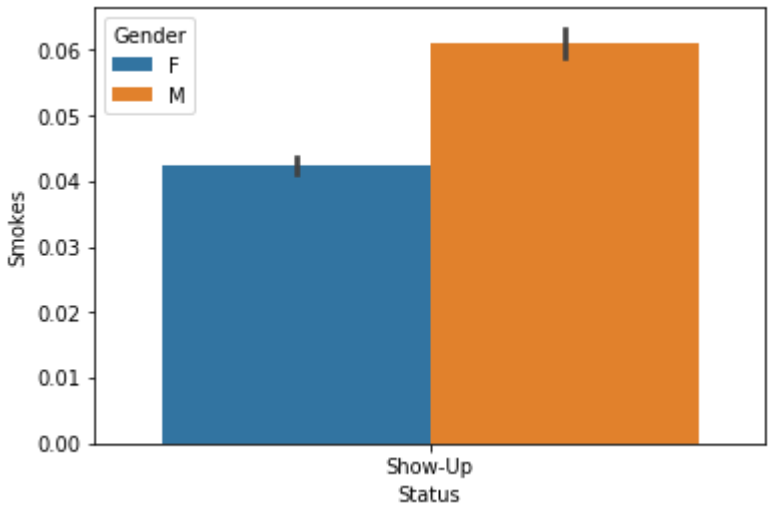
```
Index(['Age', 'Gender', 'AppointmentRegistration', 'DayOfTheWeek',
'Status',
       'Diabetes', 'Alcoolism', 'HiperTension', 'Handcap', 'Smokes',
       'Scholarship', 'Tuberculosis', 'Sms_Reminder', 'AwaitingTim
e',
       'AppointmentData', 'HourOftheDay'],
      dtype='object')
```

In [48]:

```
sns.barplot(x='Status',y='Smokes',hue = 'Gender',data = bar_df)
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1ba62710>
```

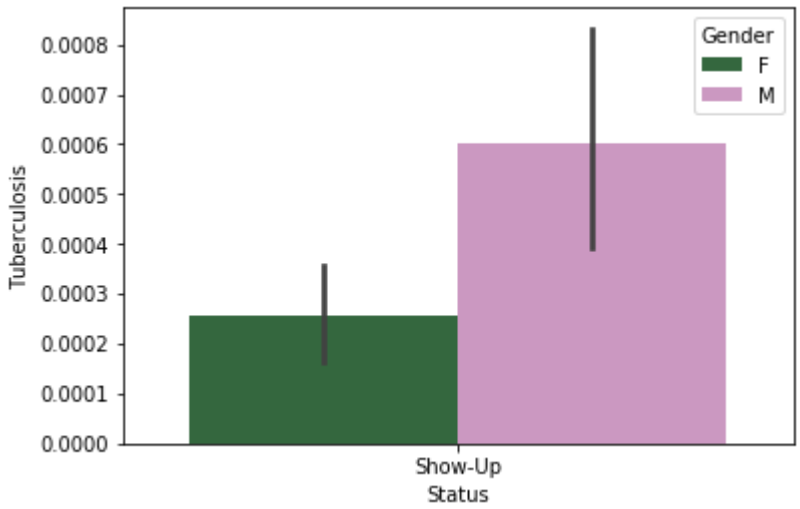


In [59]:

```
sns.barplot(x='Status',y='Tuberculosis',hue = 'Gender',data = bar_df,palette = "cubehelix")
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2eba3400>
```



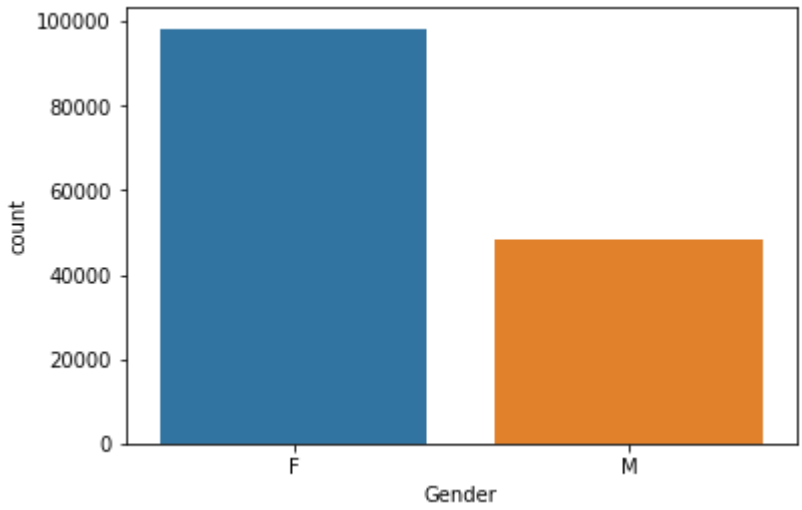
Create separate bar graphs to show the probability of showing up for male and female, day of the week and sms reminder. Describe your interpretation.

In [62]:

```
sns.countplot(x = 'Gender', data = bar_df)
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1c12c7b8>
```



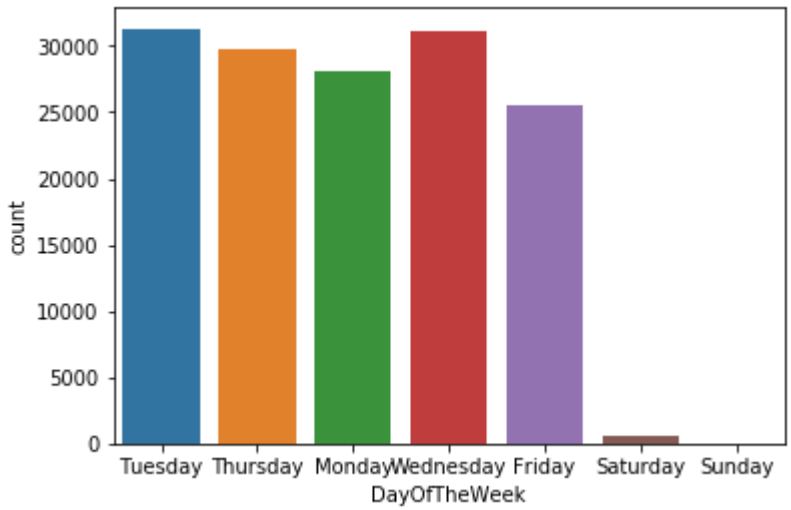
Genderwise comparision show that female patient are more likely to show up on the apointment date as compare to the male patient

In [64]:

```
sns.countplot(x = 'DayOfTheWeek',data = bar_df)
```

Out[64]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11507b6d8>
```



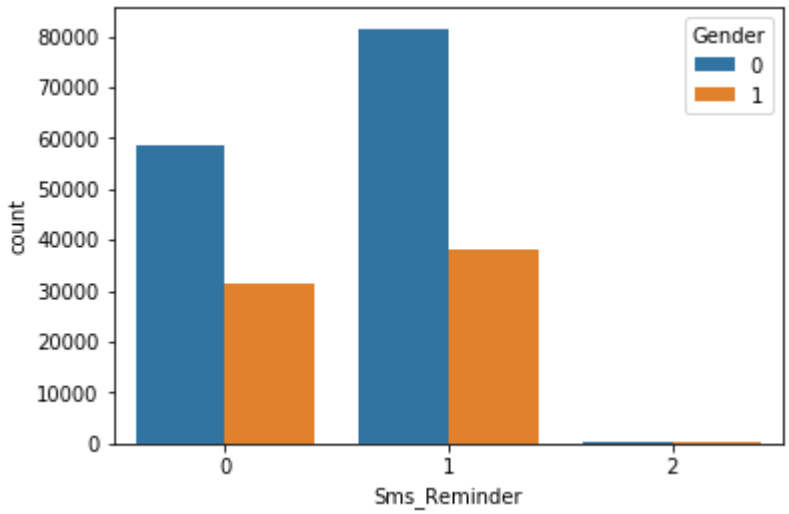
Above DayoftheWeek graph stated that the patient are most likely to show up on given appointment date from Mon to Fri but unlikely on sat-sun most patients didn't show up.

In [127]:

```
sns.countplot(x = 'Sms_Reminder',hue = 'Gender',data = df)
```

Out[127]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2ed0e9e8>
```



Above graphs of Sms Reminder suggest that the reminder sms to female patient has very high probability to show up on the appointment date

In [67]:

```
df.head()
```

Out[67]:

	Age	Gender	AppointmentRegistration	DayOfTheWeek	Status	Diabetes	Alcoolism	Hiperт
0	38	F	2015-10-20 08:33:56+00:00	Friday	No-Show	0	0	
1	56	F	2014-02-03 10:05:26+00:00	Thursday	No-Show	1	0	
2	27	F	2014-04-29 07:57:32+00:00	Tuesday	Show-Up	0	0	
3	24	M	2014-04-02 13:53:37+00:00	Tuesday	Show-Up	0	0	
4	48	F	2014-01-07 10:07:17+00:00	Thursday	Show-Up	0	0	

In [68]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210000 entries, 0 to 209999
Data columns (total 16 columns):
Age                          210000 non-null int64
Gender                       210000 non-null object
AppointmentRegistration      210000 non-null datetime64[ns, UTC]
DayOfTheWeek                 210000 non-null object
Status                       210000 non-null object
Diabetes                     210000 non-null int64
Alcoolism                    210000 non-null int64
HiperTension                 210000 non-null int64
Handcap                      210000 non-null int64
Smokes                       210000 non-null int64
Scholarship                  210000 non-null int64
Tuberculosis                 210000 non-null int64
Sms_Reminder                 210000 non-null int64
AwaitingTime                 210000 non-null int64
AppointmentData              210000 non-null datetime64[ns, UTC]
HourOftheDay                 210000 non-null object
dtypes: datetime64[ns, UTC](2), int64(10), object(4)
memory usage: 25.6+ MB
```

Creating dummies

In [69]:

```
df_dummy = pd.get_dummies(df, columns=["Gender"])
```

In [70]:

```
df_dummy.head()
```

Out[70]:

	Age	AppointmentRegistration	DayOfTheWeek	Status	Diabetes	Alcoolism	HiperTension
0	38	2015-10-20 08:33:56+00:00	Friday	No-Show	0	0	0
1	56	2014-02-03 10:05:26+00:00	Thursday	No-Show	1	0	1
2	27	2014-04-29 07:57:32+00:00	Tuesday	Show-Up	0	0	0
3	24	2014-04-02 13:53:37+00:00	Tuesday	Show-Up	0	0	0
4	48	2014-01-07 10:07:17+00:00	Thursday	Show-Up	0	0	0

Identity the outliers in Age variable

In [60]:

```
def out_std(s, nstd=3.0, return_thresholds=False):

    """
    Return a boolean mask of outliers for a series
    using standard deviation, works column-wise.

    param nstd:
        Set number of standard deviations from the mean to consider an outlier
        :type nstd: ``float``

    param return_thresholds:
        True returns the lower and upper bounds, good for plotting.
        False returns the masked array
        :type return_thresholds: ``bool``

    """
    data_mean, data_std = s.mean(), s.std()

    cut_off = data_std * nstd

    lower, upper = data_mean - cut_off, data_mean + cut_off

#     if return_thresholds:
#         return lower, upper
#     else:
#         return [True if x < lower or x > upper else False for x in s]
    return lower, upper, [True if x < lower or x > upper else False for x in s]
```

In [61]:

```
def out_iqr(s, k=1.5, return_thresholds=False):

    # calculate interquartile range
    q25, q75 = np.percentile(s, 25), np.percentile(s, 75)
    iqr = q75 - q25

    # calculate the outlier cutoff
    cut_off = iqr * k
    lower, upper = q25 - cut_off, q75 + cut_off

    if return_thresholds:
        return lower, upper
    else: # identify outliers
        return [True if x < lower or x > upper else False for x in s]
```

In [62]:

```
# outlier_mask is a boolean list identifies the indices of the outliers
lthresh, uthresh, outlier_mask = out_std(df['Age'], nstd=2.5)

# first 10 elements
print(lthresh, uthresh)
outlier_mask
```

```
-19.224012391019556 94.74766001006718
```

Out[62]:

```
[False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
```

```
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    True,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
    False,
```

```
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 ...]
```

In [63]:

```
# get all the outlier data

#dummydf['feature0'][outlier_mask]
x = df.loc[outlier_mask, ['Age']]
```

In [64]:

```
x.Age.value_counts()
```

Out[64]:

```
96     50
95     42
97     25
98     24
99     10
101     9
100     6
102     4
113     3
103     3
108     1
Name: Age, dtype: int64
```

In [65]:

```
x.Age.count()
```

Out[65]:

```
177
```

In [66]:

```
# python list gets the first index
outlier_mask.index(True)
```

Out[66]:

```
926
```

In [67]:

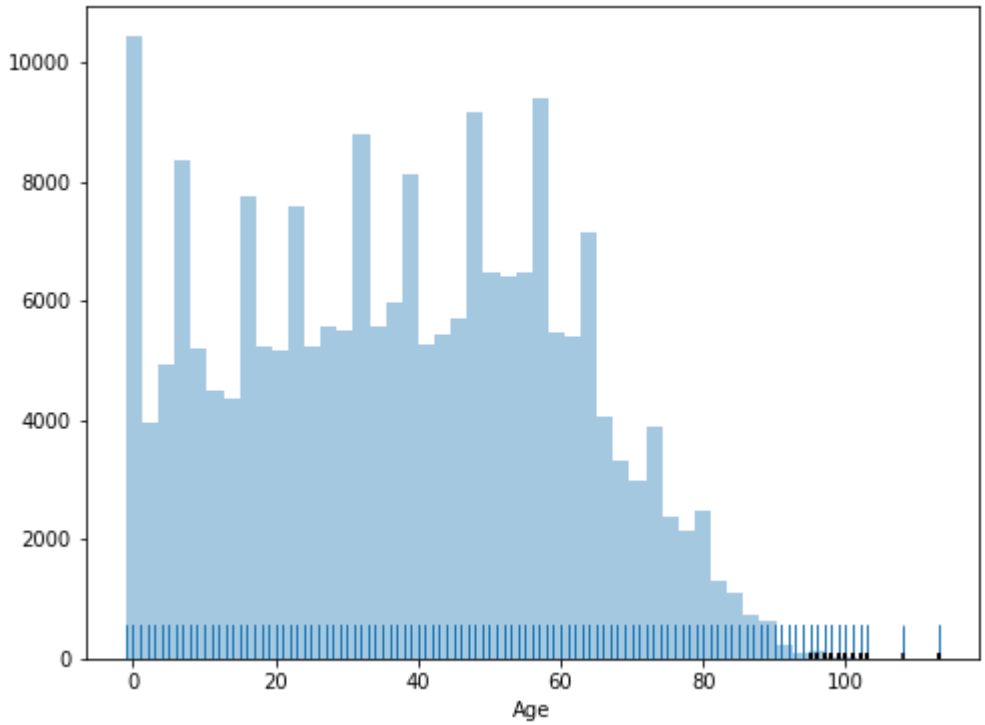
```
# convert the outlier_mask to numpy array
np_outlier_mask = np.array(outlier_mask)

np.where(np_outlier_mask == True)
```

Out[67]:

```
(array([   926,   1063,   1949,   3289,   3819,   8389,   8772,  119
41,
        15084,  17213,  17731,  18500,  18862,  26374,  26830,  269
82,
        27152,  28467,  29131,  32170,  33651,  33788,  33936,  357
53,
        38340,  38759,  39009,  40638,  41271,  43066,  43223,  441
14,
        46787,  48075,  48732,  51599,  55102,  55888,  57295,  590
59,
        59786,  61184,  61290,  64643,  64833,  65301,  66597,  675
80,
        70370,  70768,  71047,  71149,  71826,  72572,  73333,  738
50,
        74026,  74127,  75883,  77409,  78222,  80085,  80146,  807
77,
        82089,  82375,  83316,  83364,  83859,  84953,  86143,  879
05,
        88214,  89183,  89546,  90166,  90323,  93479,  94108,  952
54,
        97135,  97806, 100828, 101649, 102563, 103559, 104722, 1070
38,
       107898, 108387, 110544, 111224, 112306, 114173, 114993, 1167
45,
       116829, 123195, 123870, 124399, 125987, 126764, 127061, 1290
39,
       130384, 134912, 135493, 136537, 136836, 136846, 138319, 1392
02,
       140477, 142991, 143141, 143787, 144392, 144677, 145572, 1465
89,
       147756, 148609, 149388, 150101, 151036, 152416, 156356, 1566
72,
       157549, 157886, 157903, 157973, 160412, 162146, 163035, 1638
57,
       164305, 166292, 169751, 172217, 172702, 172938, 173485, 1748
01,
       179637, 179830, 179954, 180657, 181470, 182658, 183410, 1853
96,
       186165, 187149, 187640, 189155, 189206, 191129, 191284, 1914
88,
       192287, 192562, 193955, 194189, 195637, 196732, 198595, 2000
60,
       200489, 201397, 201566, 202030, 202427, 204907, 206057, 2071
98,
       207854]),)
```

In [70]:

```
plt.figure(figsize=(8,6))
sns.distplot(df['Age'], kde=False,rug = True);
plt.vlines(df['Age'][outlier_mask], ymin=0, ymax=110, linestyles='dashed');
```



In [71]:

```
df.head()
```

Out[71]:

	Age	Gender	AppointmentRegistration	DayOfTheWeek	Status	Diabetes	Alcoolism	Hiperl
0	38	F	2015-10-20 08:33:56+00:00	Friday	No-Show	0	0	
1	56	F	2014-02-03 10:05:26+00:00	Thursday	No-Show	1	0	
2	27	F	2014-04-29 07:57:32+00:00	Tuesday	Show-Up	0	0	
3	24	M	2014-04-02 13:53:37+00:00	Tuesday	Show-Up	0	0	
4	48	F	2014-01-07 10:07:17+00:00	Thursday	Show-Up	0	0	

Here in the dataset Status is the dependent variable and rest of all is indenpendent variables

In [85]:

```
X = df[['Age', 'Gender', 'AppointmentRegistration', 'DayOfTheWeek',
       'Diabetes', 'Alcoolism', 'HiperTension', 'Handcap', 'Smokes',
       'Scholarship', 'Tuberculosis', 'Sms_Reminder', 'AwaitingTime',
       'AppointmentData', 'HourOftheDay']]
y = df['Status']
```

In [86]:

```
df.columns
```

Out[86]:

```
Index(['Age', 'Gender', 'AppointmentRegistration', 'DayOfTheWeek',
'Status',
       'Diabetes', 'Alcoolism', 'HiperTension', 'Handcap', 'Smokes',
       'Scholarship', 'Tuberculosis', 'Sms_Reminder', 'AwaitingTim
e',
       'AppointmentData', 'HourOftheDay'],
      dtype='object')
```

In [87]:

```
X.head()
```

Out[87]:

	Age	Gender	AppointmentRegistration	DayOfTheWeek	Diabetes	Alcoolism	HiperTension
0	38	F	2015-10-20 08:33:56+00:00	Friday	0	0	0
1	56	F	2014-02-03 10:05:26+00:00	Thursday	1	0	1
2	27	F	2014-04-29 07:57:32+00:00	Tuesday	0	0	0
3	24	M	2014-04-02 13:53:37+00:00	Tuesday	0	0	0
4	48	F	2014-01-07 10:07:17+00:00	Thursday	0	0	0

Label Encoder to convert the object data types into int64

In [76]:

```
from sklearn.preprocessing import LabelEncoder
status_encoder = LabelEncoder()
```

In [77]:

```
df["Status"] = status_encoder.fit_transform(df["Status"])
```

In [78]:

```
df.head()
```

Out[78]:

	Age	Gender	AppointmentRegistration	DayOfTheWeek	Status	Diabetes	Alcoolism	HiperT
0	38	F	2015-10-20 08:33:56+00:00	Friday	0	0	0	
1	56	F	2014-02-03 10:05:26+00:00	Thursday	0	1	0	
2	27	F	2014-04-29 07:57:32+00:00	Tuesday	1	0	0	
3	24	M	2014-04-02 13:53:37+00:00	Tuesday	1	0	0	
4	48	F	2014-01-07 10:07:17+00:00	Thursday	1	0	0	

In [79]:

```
df["Gender"] = status_encoder.fit_transform(df["Gender"])
```

In [80]:

```
df["DayOfTheWeek"] = status_encoder.fit_transform(df["DayOfTheWeek"])
```

In [81]:

```
df.head()
```

Out[81]:

	Age	Gender	AppointmentRegistration	DayOfTheWeek	Status	Diabetes	Alcoolism	HiperT
0	38	0	2015-10-20 08:33:56+00:00	0	0	0	0	
1	56	0	2014-02-03 10:05:26+00:00	4	0	1	0	
2	27	0	2014-04-29 07:57:32+00:00	5	1	0	0	
3	24	1	2014-04-02 13:53:37+00:00	5	1	0	0	
4	48	0	2014-01-07 10:07:17+00:00	4	1	0	0	

In [82]:

```
X = df[['Age',
       'Diabetes', 'Alcoolism', 'HiperTension', 'Smokes',
       'Scholarship', 'Tuberculosis']]
y = df['Status']
```

In [83]:

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210000 entries, 0 to 209999
Data columns (total 7 columns):
Age             210000 non-null int64
Diabetes        210000 non-null int64
Alcoolism       210000 non-null int64
HiperTension    210000 non-null int64
Smokes          210000 non-null int64
Scholarship     210000 non-null int64
Tuberculosis    210000 non-null int64
dtypes: int64(7)
memory usage: 11.2 MB
```

Logistic Regression

Splitting the training and testing dataset

In [85]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =0.3,random_state
=1)
```

In [86]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(147000, 7)
(63000, 7)
(147000,)
(63000,)
```

In [87]:

```
%time
# Importing the LR model from scikit learn linear model
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
```

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 6.91 µs
```

In [88]:

```
# Fitting the LR model on training dataset
classifier.fit(X_train,y_train)
```

Out[88]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interce
pt=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='war
n',
          tol=0.0001, verbose=0, warm_start=False)
```

In [89]:

```
# Predicting the values on independent variables testing dataset
y_pred = classifier.predict(X_test)
```

In [90]:

```
# Confusion matrix for evaluation to get the accuracy of the model
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[90]:

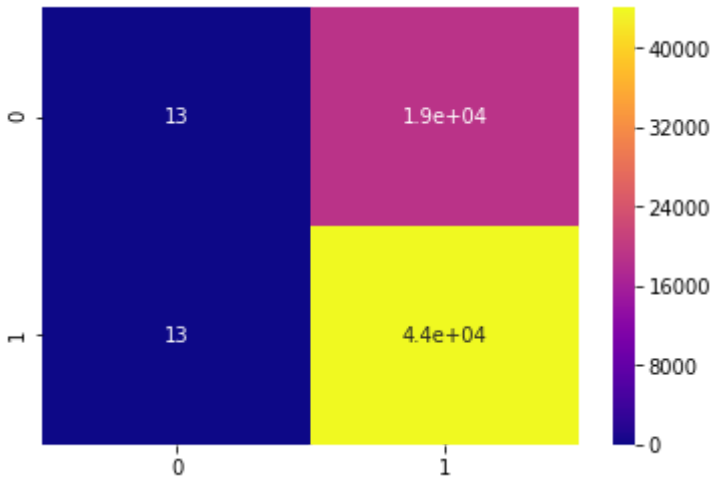
```
array([[   13, 18957],
       [   13, 44017]])
```

In [91]:

```
sns.heatmap(cm,annot = True,cmap ='plasma')
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3c556b70>
```



In [101]:

```
# Accuracy of the LR model is base on Actual values and predicting values by the
model
from sklearn.metrics import accuracy_score
model_accuracy = accuracy_score(y_test,y_pred)
print('Accuracy of the model : ',round(model_accuracy*100,2))
```

```
Accuracy of the model :  69.89
```

In [112]:

```
# Cross validation score of 10 Kfolds
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(estimator = classifier,X=X_train,y=y_train,cv=10)
accuracy
```

Out[112]:

```
array([0.69641521, 0.69627916, 0.69600707, 0.69653061, 0.6962585 ,
       0.6962585 , 0.69605442, 0.69610178, 0.69616981, 0.69623784])
```

In [113]:

```
print("The mean accuracy for  10 Kfolds :  ",accuracy.mean())
```

```
The mean accuracy for  10 Kfolds :   0.6962312912107043
```

In [114]:

```
print("The Std deviation of the model :",accuracy.std())
```

```
The Std deviation of the model : 0.00015144155219598795
```

Decision Tree

In [102]:

```
%time
# Importing the model from scikit learn tree
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
```

```
CPU times: user 6 µs, sys: 2 µs, total: 8 µs
Wall time: 14.1 µs
```

In [103]:

```
# fitting the DT model on training dataset
classifier.fit(X_train,y_train)
```

Out[103]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_dept
h=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_stat
e=None,
            splitter='best')
```

In [104]:

```
# Predicting the values on independent variables testing dataset
y_pred = classifier.predict(X_test)
```

In [105]:

```
# Confusion matrix for evaluation to get the accuracy of the model
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[105]:

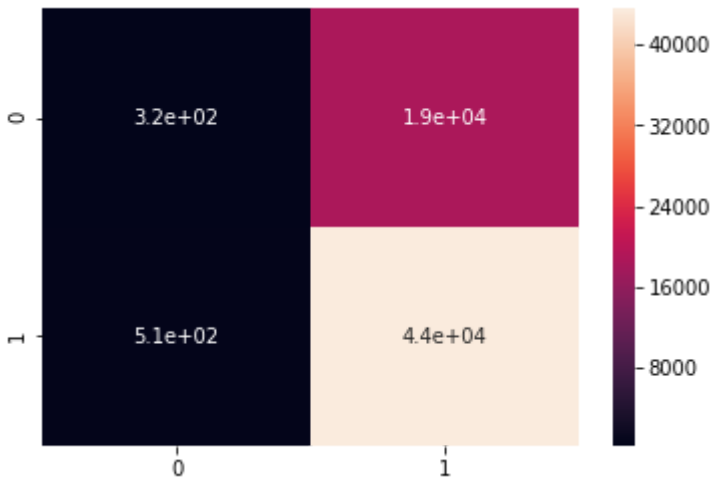
```
array([[  316, 18654],
       [  508, 43522]])
```

In [106]:

```
sns.heatmap(cm,annot = True)
```

Out[106]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a30d54e80>
```



In [107]:

```
# Accuracy of the DT model is base on Actual values and predicting values by the
model
from sklearn.metrics import accuracy_score
model_accuracy = accuracy_score(y_test,y_pred)
print("The accuracy of the DT model :",model_accuracy)
```

```
The accuracy of the DT model : 0.6958412698412698
```

In [108]:

```
# Cross validation score of 10 Kfolds
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(estimator = classifier,X=X_train,y=y_train,cv=10)
accuracy
```

Out[108]:

```
array([0.69349024, 0.6945786 , 0.69355826, 0.69272109, 0.69455782,
       0.69564626, 0.6937415 , 0.6933805 , 0.69412885, 0.69494523])
```

In [109]:

```
print('The mean accuracy for 10 Kfold :',accuracy.mean())
```

```
The mean accuracy for 10 Kfold : 0.6940748355611467
```

In [110]:

```
print("The standard deviation of the DT model :",accuracy.std())
```

```
The standard deviation of the DT model : 0.0008216681284334002
```

Random Forest

In [111]:

```
%time
# Importing the RF model from scikit learn ensemble
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier()
model = classifier.fit(X_train,y_train)
model
```

```
CPU times: user 4 µs, sys: 1 µs, total: 5 µs
Wall time: 11 µs
```

Out[111]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion
='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=Non
e,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=No
ne,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [112]:

```
# Predicting the values on independent variables testing dataset
y_pred = classifier.predict(X_test)
```

In [113]:

```
# Confusion matrix for evaluation to get the accuracy of the RF model
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[113]:

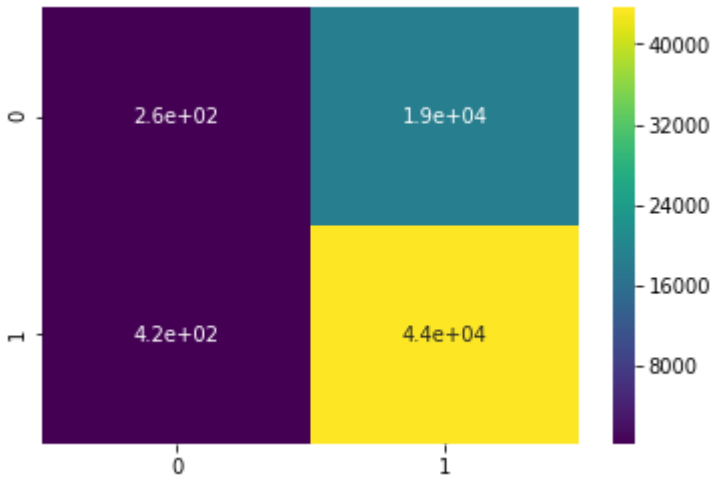
```
array([[  263, 18707],
       [  422, 43608]])
```

In [114]:

```
sns.heatmap(cm,annot = True,cmap = 'viridis')
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a386c78d0>
```



In [115]:

```
# Accuracy of the DT model is base on Actual values and predicting values by the
model
from sklearn.metrics import accuracy_score
model_accuracy = accuracy_score(y_test,y_pred)
print('The accuracy of the RF model : ',model_accuracy*100)
```

```
The accuracy of the RF model :  69.63650793650794
```

In [116]:

```
# Cross validation score of 10 Kfolds
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(estimator = classifier,X=X_train,y=y_train,cv=10)
accuracy
```

Out[116]:

```
array([0.69301408, 0.69505476, 0.69315013, 0.69244898, 0.6944898 ,
       0.6962585 , 0.69394558, 0.69365263, 0.69378869, 0.69501327])
```

In [117]:

```
print('The mean accuracy for 10 Kfold :',accuracy.mean())
```

```
The mean accuracy for 10 Kfold : 0.694081641058667
```

In [118]:

```
print("The standard deviation of the DT model :",accuracy.std())
```

```
The standard deviation of the DT model : 0.0010808133145547333
```

XGBoost Classifier

In [119]:

```
%time
# Importing the XGBoost model from scikit learn ensemble
from xgboost import XGBClassifier
classifier = XGBClassifier(n_estimator =1000)
classifier.fit(X_train,y_train)
```

```
CPU times: user 4 µs, sys: 1 µs, total: 5 µs
Wall time: 8.82 µs
```

Out[119]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_ste
p=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimator=10
00,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
       subsample=1)
```

In [120]:

```
# Predicting the values on independent variables testing dataset
y_pred = classifier.predict(X_test)
```

In [121]:

```
# Confusion matrix for evaluation to get the accuracy of the XGBoost model
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[121]:

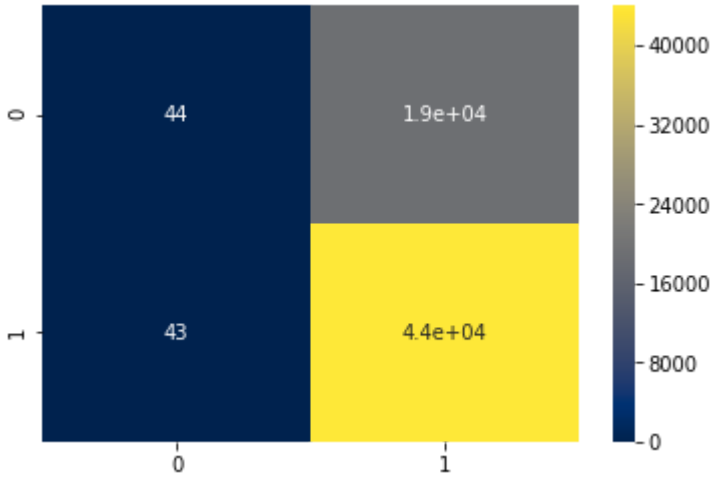
```
array([[   44, 18926],
       [   43, 43987]])
```

In [122]:

```
sns.heatmap(cm,annot = True,cmap="cividis")
```

Out[122]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2ed3c5c0>
```



In [123]:

```
# Accuracy of the XGBoost model is base on Actual values and predicting values b
y the model
from sklearn.metrics import accuracy_score
model_accuracy = accuracy_score(y_test,y_pred)
model_accuracy
```

Out[123]:

```
0.6989047619047619
```

In [124]:

```
# Cross validation score of 10 Kfolds
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(estimator = classifier,X=X_train,y=y_train,cv=10)
accuracy
```

Out[124]:

```
array([0.69641521, 0.69621114, 0.69553092, 0.69632653, 0.69632653,
       0.69639456, 0.69646259, 0.69610178, 0.69610178, 0.6963739 ])
```

In [125]:

```
print('The mean accuracy for 10 Kfold :',accuracy.mean())
```

```
The mean accuracy for 10 Kfold : 0.6962244926543267
```

In [126]:

```
print("The standard deviation of the DT model :",accuracy.std())
```

```
The standard deviation of the DT model : 0.00026009142825188453
```

The Champion model out of all Models is XGBoost

In [ ]:

Tableau Public link

https://public.tableau.com/profile/nikhil8753#!/vizhome/HealthCare_Capstone_P
(https://public.tableau.com/profile/nikhil8753#!/vizhome/HealthCare_Capstone_P

In [ ]: