Project - Investigate dataset

This dataset collects information from 100k medical appointments in Brazil and is focused on the question of whether or not patients show up for their appointment. A number of characteristics about the patient are included in each row

Dataset

1. PatienID---Identity of a patient
2. AppointmentID---Identity number of a patient
3. Gender---Shows the gender
4. ScheduledDay---Shows the date of scheduling appointmnet
5. AppointmentDay---Shows the date of the appointment
6. Neighbourhood---Shows the location of the hospital
7. Scholarship ---Shows if the patient receives a scholarship
8. Hipertension--- Shows if the patient has hypertension
9. Diabetes ---Shows if the patient has diabetes
10. Alcoholism ---Shows if the patient is an alcoholic
11. Handcap ---Shows if the patient is handicaped
12. SMS_received ---Shows if message is sent to the patient
13. No-show -- It says 'No' if the patient showed up to their appointment, and 'Yes' if they did not show up

#Importing all the necessary libraries

In [1]:

```
import pandas as pd
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import collections
```

#Reading the dataset

In [2]:

```
df = pd.read_csv("noshow.csv")
df.head()
```

Out[2]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA

#Analyzing the dataset

Check dimensions of the dataframe in terms of rows and columns

In [3]:

```
df.shape
```

Out[3]:

```
(110527, 14)
```

Inference drawn:

- The no.of rows are 211944
- The no.of columns are 26

Checking if the dataset has any duplicate values

In [4]:

```
sum(df.duplicated())
```

Out[4]:

```
0
```

Inference drawn:

- The dataset has no duplicate values

Checking if there are any null or missing values in the dataset

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
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
```

Inference drawn:

- The dataset has no missing values

Displaying the columns in the dataset

In [6]:

```
df.columns
```

Out[6]:

```
Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hiper
tension',
       'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'No-sho
w'],
      dtype='object')
```

Inference drawn:

- Some column names have incorrect spellings and are in the wrong format so they'll be cleaned accordingly

Changing column names which are in incorrect format and have wrong spellings

In [7]:

```
df.rename(columns={"Hipertension": "Hypertension","AppointmentID": "Appointment_id",
```

Checking if datatypes are in correct format

In [8]:

```
df.dtypes
```

Out[8]:

```
Patient_id          float64
Appointment_id        int64
Gender               object
Scheduled_day        object
Appointment_day      object
Age                   int64
Neighbourhood        object
Scholarship           int64
Hypertension          int64
Diabetes              int64
Alcoholism            int64
Handicap              int64
SMS_received          int64
No_show              object
dtype: object
```

Inference drawn:

- Scheduled_day's data type is object but to make it easy to use for the user, we can convert it in datetime format
- Appointment_day's data type is object but to make it easy to use for the user, we can convert it in datetime format

Inference drawn:

- There are no redundant values in the dataset

#Note the redundant variables and drop them

In [9]:

```
df.head()
```

Out[9]:

intment_id	Gender	Scheduled_day	Appointment_day	Age	Neighbourhood	Scholarship	Hyperten
5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	0	
5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	
5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA	0	
5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI	0	
5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	

Inference drawn:

- When we analyze the dataset, we can try can observe that there are no such columns in the dataset which have only 1 unique values in them, and hence we can conclude by stating that there are no redundant variables in the dataset.

#Analysing the variables

Variable 'Patient_id'

In [10]:

```
df.Patient_id.unique()
```

Out[10]:

```
array([2.98724998e+13, 5.58997777e+14, 4.26296230e+12, ...,
       7.26331493e+13, 9.96997666e+14, 1.55766317e+13])
```

Inference:

- The data type of an id should ideally be integer, not float.

In [11]:

```
df['Patient_id'] = df['Patient_id'].astype('int64')
```

Variable 'Gender'

In [12]:

```
df.Gender.unique()
```

Out[12]:

```
array(['F', 'M'], dtype=object)
```

Inference -

- The column has 2 unique values for the genders, male and female in the correct format

Variable 'Scheduled_day'

In [13]:

```
df.Scheduled_day.unique()
```

Out[13]:

```
array(['2016-04-29T18:38:08Z', '2016-04-29T16:08:27Z',
       '2016-04-29T16:19:04Z', ..., '2016-04-27T16:03:52Z',
       '2016-04-27T15:09:23Z', '2016-04-27T13:30:56Z'], dtype=object)
```

Inference -

- The date type needs to be converted to datetime format

In [14]:

```
df.Scheduled_day = df.Scheduled_day.apply(np.datetime64)
```

Variable 'Appointment_day'

In [15]:

```
df.Appointment_day.unique()
```

Out[15]:

```
array(['2016-04-29T00:00:00Z', '2016-05-03T00:00:00Z',
       '2016-05-10T00:00:00Z', '2016-05-17T00:00:00Z',
       '2016-05-24T00:00:00Z', '2016-05-31T00:00:00Z',
       '2016-05-02T00:00:00Z', '2016-05-30T00:00:00Z',
       '2016-05-16T00:00:00Z', '2016-05-04T00:00:00Z',
       '2016-05-19T00:00:00Z', '2016-05-12T00:00:00Z',
       '2016-05-06T00:00:00Z', '2016-05-20T00:00:00Z',
       '2016-05-05T00:00:00Z', '2016-05-13T00:00:00Z',
       '2016-05-09T00:00:00Z', '2016-05-25T00:00:00Z',
       '2016-05-11T00:00:00Z', '2016-05-18T00:00:00Z',
       '2016-05-14T00:00:00Z', '2016-06-02T00:00:00Z',
       '2016-06-03T00:00:00Z', '2016-06-06T00:00:00Z',
       '2016-06-07T00:00:00Z', '2016-06-01T00:00:00Z',
       '2016-06-08T00:00:00Z'], dtype=object)
```

Inference -

- The date type needs to be converted to datetime format

In [16]:

```
df.Appointment_day = df.Appointment_day.apply(np.datetime64)
```

Variable 'Age'

In [17]:

```
df.Age.unique()
```

Out[17]:

```
array([ 62,  56,   8,  76,  23,  39,  21,  19,  30,  29,  22,  28,  5
4,
        15,  50,  40,  46,   4,  13,  65,  45,  51,  32,  12,  61,  3
8,
        79,  18,  63,  64,  85,  59,  55,  71,  49,  78,  31,  58,  2
7,
         6,   2,  11,   7,   0,   3,   1,  69,  68,  60,  67,  36,  1
0,
        35,  20,  26,  34,  33,  16,  42,   5,  47,  17,  41,  44,  3
7,
        24,  66,  77,  81,  70,  53,  75,  73,  52,  74,  43,  89,  5
7,
        14,   9,  48,  83,  72,  25,  80,  87,  88,  84,  82,  90,  9
4,
        86,  91,  98,  92,  96,  93,  95,  97, 102, 115, 100,  99,  -
1])
```

Inference -

- The age column has negative values which is highly unlikely to happen. So we'll have to filter out the outliers.

In [18]:

```
df = df[(df.Age >= 0)]
```

Variable 'Neighbourhood'

In [19]:

```
df.Neighbourhood.unique()
```

Out[19]:

```
array(['JARDIM DA PENHA', 'MATA DA PRAIA', 'PONTAL DE CAMBURI',
       'REPÚBLICA', 'GOIABEIRAS', 'ANDORINHAS', 'CONQUISTA',
       'NOVA PALESTINA', 'DA PENHA', 'TABUAZEIRO', 'BENTO FERREIRA',
       'SÃO PEDRO', 'SANTA MARTHA', 'SÃO CRISTÓVÃO', 'MARUÍPE',
       'GRANDE VITÓRIA', 'SÃO BENEDITO', 'ILHA DAS CAIEIRAS',
       'SANTO ANDRÉ', 'SOLON BORGES', 'BONFIM', 'JARDIM CAMBURI',
       'MARIA ORTIZ', 'JABOUR', 'ANTÔNIO HONÓRIO', 'RESISTÊNCIA',
       'ILHA DE SANTA MARIA', 'JUCUTUQUARA', 'MONTE BELO',
       'MÁRIO CYPRESTE', 'SANTO ANTÔNIO', 'BELA VISTA', 'PRAIA DO SU
Á',
       'SANTA HELENA', 'ITARARÉ', 'INHANGUETÁ', 'UNIVERSITÁRIO',
       'SÃO JOSÉ', 'REDENÇÃO', 'SANTA CLARA', 'CENTRO', 'PARQUE MOSCOS
O',
       'DO MOSCOSO', 'SANTOS DUMONT', 'CARATOÍRA', 'ARIOVALDO FAVALESS
A',
       'ILHA DO FRADE', 'GURIGICA', 'JOANA D´ARC', 'CONSOLAÇÃO',
       'PRAIA DO CANTO', 'BOA VISTA', 'MORADA DE CAMBURI', 'SANTA LUÍZ
A',
       'SANTA LÚCIA', 'BARRO VERMELHO', 'ESTRELINHA', 'FORTE SÃO JOÃ
O',
       'FONTE GRANDE', 'ENSEADA DO SUÁ', 'SANTOS REIS', 'PIEDADE',
       'JESUS DE NAZARETH', 'SANTA TEREZA', 'CRUZAMENTO',
       'ILHA DO PRÍNCIPE', 'ROMÃO', 'COMDUSA', 'SANTA CECÍLIA',
       'VILA RUBIM', 'DE LOURDES', 'DO QUADRO', 'DO CABRAL', 'HORTO',
       'SEGURANÇA DO LAR', 'ILHA DO BOI', 'FRADINHOS', 'NAZARETH',
       'AEROPORTO', 'ILHAS OCEÂNICAS DE TRINDADE', 'PARQUE INDUSTRIA
L'],
      dtype=object)
```

Inference -

- The variable shows the neighbourhood in which hospital is located

Variable 'Scholarship'

In [20]:

```
df.Scholarship.unique()
```

Out[20]:

```
array([0, 1])
```

Inference -

- The variable has 2 unique values which indicate if patient receives a scholarship or no in the correct data type

Variable 'Hypertension'

In [21]:

```
df.Hypertension.unique()
```

Out[21]:

```
array([1, 0])
```

Inference -

- The variable has 2 unique values which is 1 if patient has hypertension and 0 or else in the correct data type

Variable 'Diabetes'

In [22]:

```
df.Diabetes.unique()
```

Out[22]:

```
array([0, 1])
```

Inference -

- The variable has 2 unique values which is 1 if patient is diabetic and 0 if not in the correct data type

Variable 'Alcoholism'

In [23]:

```
df.Alcoholism.unique()
```

Out[23]:

```
array([0, 1])
```

Inference -

- The variable has 2 unique values which is 1 if patient is alcoholic and 0 if patient is non alcoholic in correct data type

Variable 'Handicap'

In [24]:

```
df.Handicap.unique()
```

Out[24]:

```
array([0, 1, 2, 3, 4])
```

The column has 3 unique values possibly reppresenting the number of disabilities an individual has

Variable 'SMS_received'

In [25]:

```
df.SMS_received.unique()
```

Out[25]:

```
array([0, 1])
```

Inference -

- The variable has 2 unique values which show if patient had received a message or not in the correct data type

Variable 'No_show'

In [26]:

```
df.No_show.unique()
```

Out[26]:

```
array(['No', 'Yes'], dtype=object)
```

Inference -

- The variable has 2 unique values displaying ‘No’ if the patient showed up to their appointment, and ‘Yes’ if they did not show up

Adding a new column displaying the waiting period for a patient

In [27]:

```
df['Wait'] = (df.Appointment_day.dt.date - df.Scheduled_day.dt.date).dt.days
df= df[(df.Wait>=0)]
```

Adding a new column which shows the day of the appointment

In [28]:

```
df['appointment_day'] = df.Scheduled_day.dt.day_name()
```

Understanding the variable 'Appointment_Day'

In [29]:

```
collections.Counter(df.appointment_day)
```

Out[29]:

```
Counter({'Friday': 18915,
         'Wednesday': 24259,
         'Tuesday': 26167,
         'Thursday': 18072,
         'Monday': 23084,
         'Saturday': 24})
```

By observing, very few appointments are made for the weekend, Saturday with majority of appoints being made for the former part of week on days like Monday, Tuesday, Wednesday with the number dropping in the latter part of week for days like Thursday and Friday

In [30]:

```
df.head(5)
```

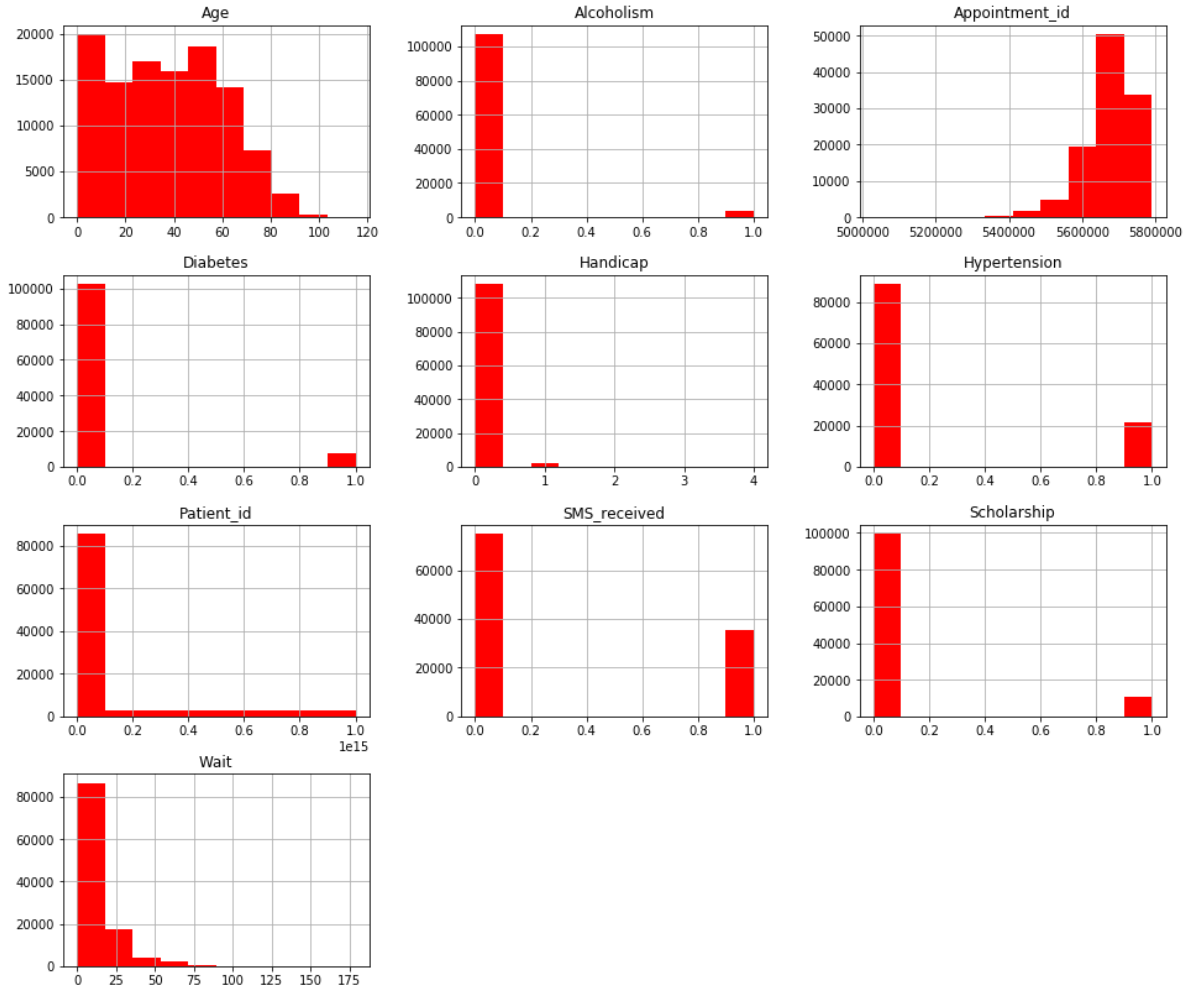
Out[30]:

Scheduled_day	Appointment_day	Age	Neighbourhood	Scholarship	Hypertension	Diabetes	Alc
2016-04-29 18:38:08	2016-04-29	62	JARDIM DA PENHA	0	1	0	
2016-04-29 16:08:27	2016-04-29	56	JARDIM DA PENHA	0	0	0	
2016-04-29 16:19:04	2016-04-29	62	MATA DA PRAIA	0	0	0	
2016-04-29 17:29:31	2016-04-29	8	PONTAL DE CAMBURI	0	0	0	
2016-04-29 16:07:23	2016-04-29	56	JARDIM DA PENHA	0	1	1	

Observations

In [49]:

```
val_main=df.hist(figsize=(16,14),color='red')
```



The observations made from the histograms are:

- Patients are evenly distributed when it comes to their age with majority of patients who are minors make an appointment
- Majority of patients do not have alcoholism. Only a very small amount of patients have alcoholism
- Majority of patients do not have diabetes. Only a very small amount of patients have have diabetes
- Majority of patients are not handicapped. Only a very small amount of patients have some disability
- Around 75% of patients do not have Hypertension while 25% of patients do have Hypertension
- Almost 7k patients did receive a text message whereas almost 3.9k patients did not receive a text message
- Majority of patiients do not receive a scholarship with a small amount of patients receieving a scholarship
- Majority of patients do not have to wait for more than 20 days with a small amount of patients having to wait upto 75 days

Did the gender play any role in the possibilty of a patient missing their appointment?

In [32]:

```
female= df[df['Gender']=='F']
total_females= female.shape[0]
male= df[df['Gender']=='M']
total_males= male.shape[0]
females_who_did_not_attend = (female[["No_show"]]=="Yes").sum()
females_who_attended = (female[["No_show"]]=="No").sum()
males_who_did_not_attend = (male[["No_show"]]=="Yes").sum()
males_who_attended = (male[["No_show"]]=="No").sum()
```

The percentage of females who missed their appointments

In [33]:

```
(females_who_did_not_attend/total_females)*100
```

Out[33]:

```
No_show    20.311543
dtype: float64
```

The percentage of females who attended their appointments

In [34]:

```
(females_who_attended/total_females)*100
```

Out[34]:

```
No_show    79.688457
dtype: float64
```

Percentage of males who missed their appointments

In [35]:

```
(males_who_did_not_attend/total_males)*100
```

Out[35]:

```
No_show    19.96381
dtype: float64
```

Percentage of males who attended their appointments

In [36]:

```
(males_who_attended/total_males)*100
```

Out[36]:

```
No_show    80.03619
dtype: float64
```

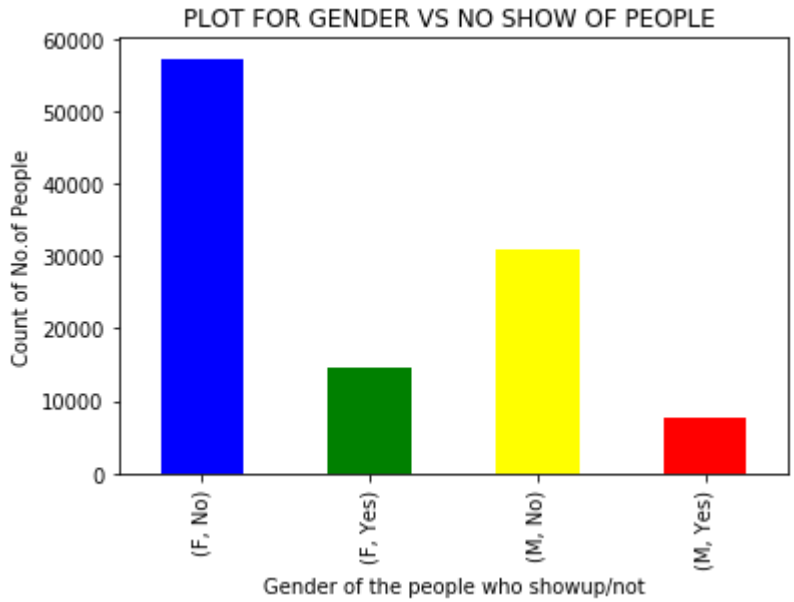
Plotting a graph for better understanding

In [44]:

```
gender =df.groupby('Gender').No_show.value_counts()
color=['blue','green','yellow','red']
p=gender.plot(kind='bar',color=color,stacked=True,title='PLOT FOR GENDER VS NO SHOW
p.set_xlabel("Gender of the people who showup/not")
p.set_ylabel("Count of No.of People")
```

Out[44]:

```
Text(0, 0.5, 'Count of No.of People')
```



Inference

- The percentage of female patients who missed their appointments is approximately equal to the number of male patients who misssed ther appointments
- The percentage of female patients who attended their appointments is approximately equal to the number of male patients who attended ther appointments
- Thus, the gender of a person doesn't play a significant role in causing them to miss theri appointments

Is there a relation of patient not showing up and the number of days a patient has to wait for the appointent?

In [38]:

```
Waiting_df = df[['No_show', 'Wait']].groupby('Wait').count()
```

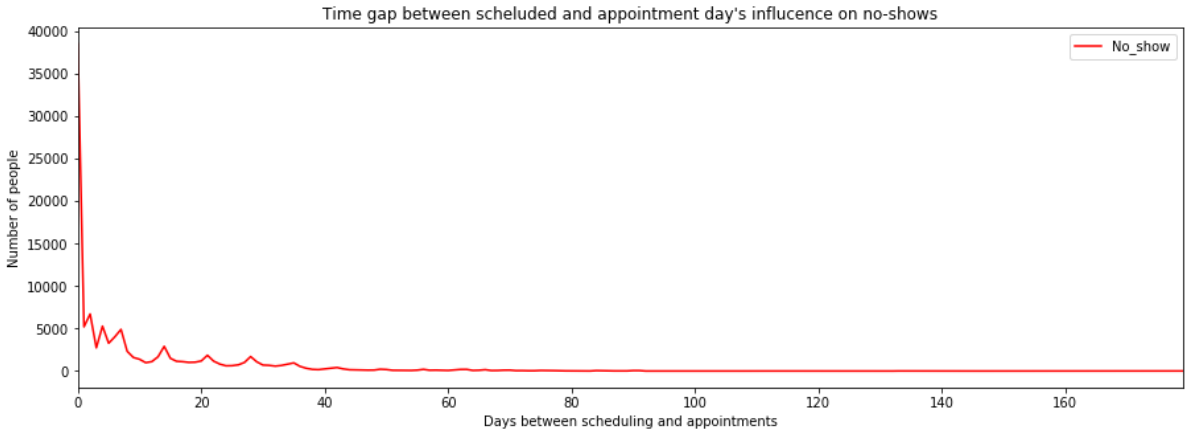
Plotting a graph for better understanding

In [39]:

```
Waiting_df.plot(kind='line', figsize=(15,5),color='red')
plt.title("Time gap between scheluded and appointment day's influcence on no-shows")
plt.xlabel('Days between scheduling and appointments')
plt.ylabel('Number of people')
```

Out[39]:

```
Text(0, 0.5, 'Number of people')
```



Inference:

- Majority of patients attend their appointments if the appointments are scheduled in a small time gap, ideally on the same day

Does the day of the appointment influence the patient's decision to attend or miss the appointent?

In [40]:

```
day = df.groupby('appointment_day').No_show.value_counts()
day
```

Out[40]:

```
appointment_day  No_show
Friday           No         15028
                 Yes         3887
Monday           No         18523
                 Yes         4561
Saturday         No            23
                 Yes            1
Thursday         No         14373
                 Yes         3699
Tuesday          No         20877
                 Yes         5290
Wednesday        No         19383
                 Yes         4876
Name: No_show, dtype: int64
```

Calculating the percentage

In [41]:

```
percent= []
i=0
while i<len(day)-1:
  percent.append( day[i+1] *100 /(day[i]+day[i+1])  )
  i=i+2
percent
```

Out[41]:

```
[20.54982817869416,
 19.758274129267026,
 4.166666666666667,
 20.46812749003984,
 20.21630297703214,
 20.099756791293952]
```

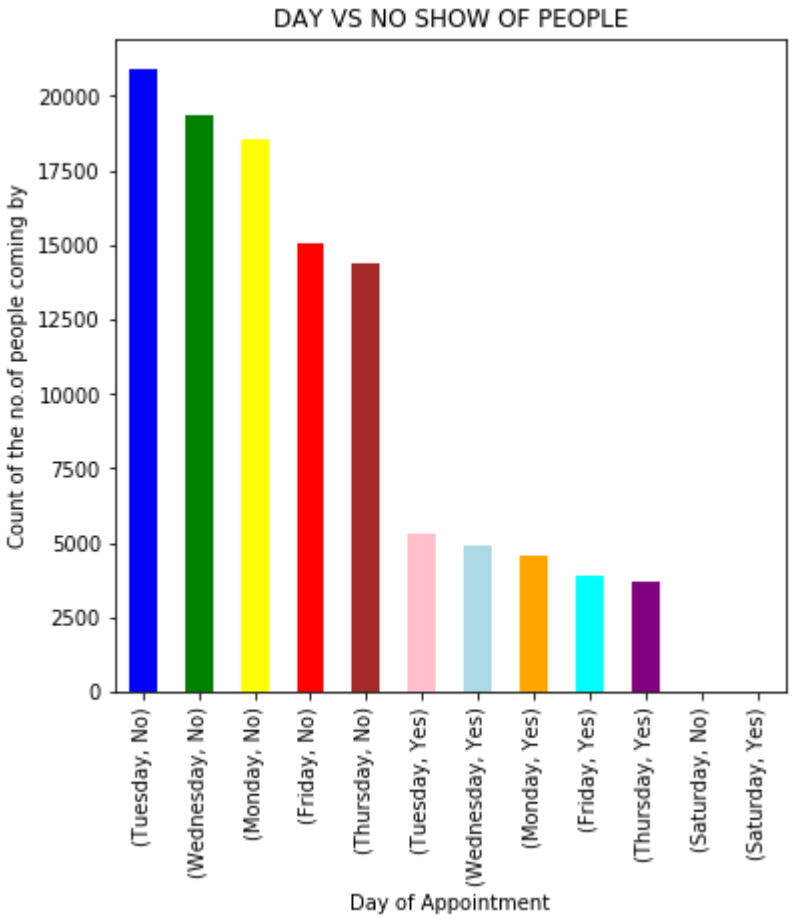
Plotting a graph for better understanding

In [50]:

```
day = day.sort_values(ascending=False)
color=['blue','green','yellow','red','brown','pink','lightblue','orange','cyan','pur
ans_graph=day.plot(kind='bar', figsize=(6,6),color=color,title='DAY VS NO SHOW OF PE
ans_graph.set_xlabel("Day of Appointment")
ans_graph.set_ylabel("Count of the no.of people coming by")
```

Out[50]:

```
Text(0, 0.5, 'Count of the no.of people coming by')
```



Inference:

- The number of appointments scheduled, attended and missed, both are negligible
- The number of appointents , both missed and attended are maximum for Tuesday

- Wednesday comes right after Tuesday for both having the number of appointments attended as well as missed
- It is followed by Monday with a lesser number of patients attening as well as missing the appointent
- The number of patients attending as well as issing the appointment keeps on decreasing for Thursday and Friday
- Thus, the numbers of patients attending as well as missing the appointnets goes hand in hand
- Saturday is the only day when least number of patients, around 4% of those scheduled will miss their appointments
- For all the other days, around 20% of the scheduled appointents will be cancelled

#Conclusion

- In this project, we analyzed the no show database of patients
- We analyzed all the variables of the dataset
- Gender of a patient does not have influence on whether the patient shows up or no
- Whether the patient shows up or not is affected by the amount of time between the patient scheduled his appointment and his appointment
- Patient is more likely to show up if the time between the patient scheduled his appointment and his appointment is less
- The weekday on which the appointment has been scheduled does not affect the patient's behaviour except for on Saturday when percentage of patients not showing is the least

LIMITATIONS-

- Incorrect names of the columns : hypertension, appointment id, scheduled day
- Incorrect data types of columns : scheduled day and appointment day
- Age column having negative values : -1 , age can never be negative
- Data type of ID was flaot : it should have been integer

In [ ]: