Project: NO-SHOW APPOINTMENT DATA ANALYSIS

Table of Contents

Introduction

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# % matplotlib inline
df = pd.read_csv('noshowappointments-kagglev2-may-2016.csv')
```

Data Wrangling

General Properties

In [2]:
```
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	5.589980e+14	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA
2	4.262960e+12	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA
3	8.679510e+11	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI
4	8.841190e+12	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA

In [3]:
```
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
```

There is not missing values.

The type of PatientId column is float64.I am going to convert it from float64 to int64.

Type of Scheduled day and appointment day is String so I am going to convert them from String to datetime64.I am going to create a new column to find out the count of Appointment Day - Scheduled day.(This step includes converting the data types)

Working with 'SMS_received' and 'No-Show' as a value name is hard so I am going to rename them.

In [4]:
```
#cheking duplicated values of data
df.duplicated().sum()
```

Out[4]:
```
0
```

There is not duplicated values.

In [5]:
```
# Cheking the uniques of 'Gender'  column
df.Gender.unique()
```

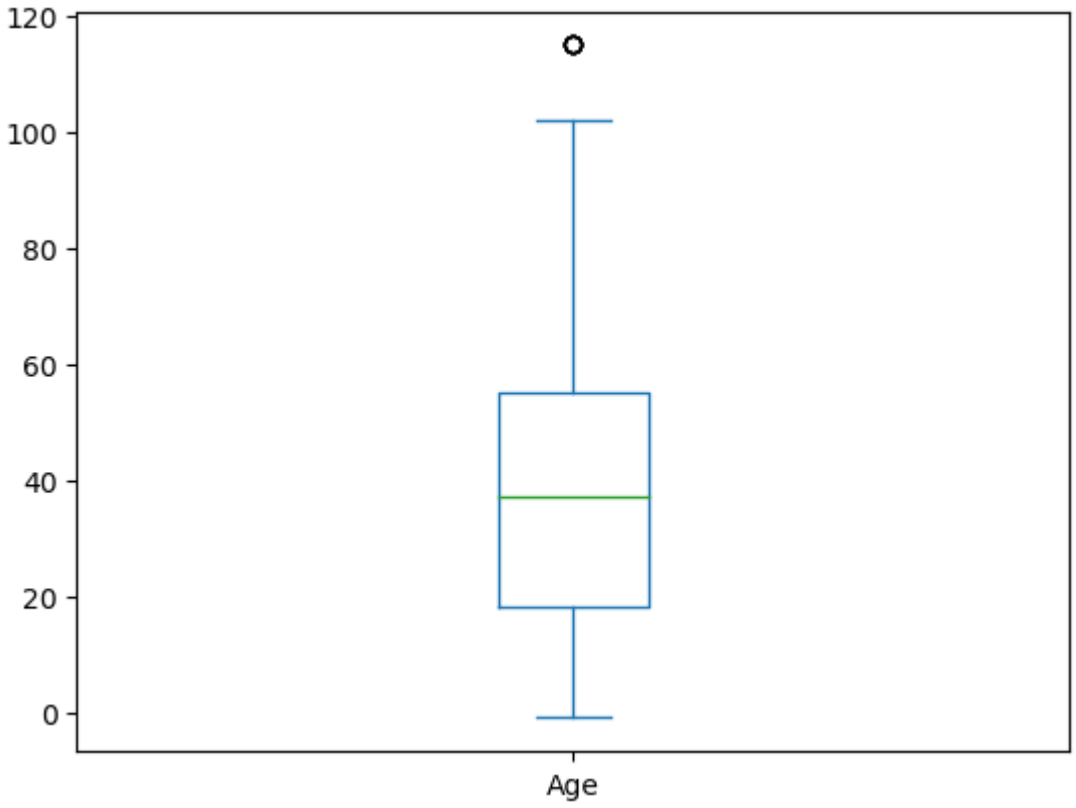
Out[5]:
```
array(['F', 'M'], dtype=object)
```

In [6]:
```
#Cheking the uniques of 'Age' column
df.Age.unique()
```

Out[6]:
```
array([ 62,  56,   8,  76,  23,  39,  21,  19,  30,  29,  22,  28,  54,
        15,  50,  40,  46,   4,  13,  65,  45,  51,  32,  12,  61,  38,
        79,  18,  63,  64,  85,  59,  55,  71,  49,  78,  31,  58,  27,
         6,   2,  11,   7,   0,   3,   1,  69,  68,  60,  67,  36,  10,
        35,  20,  26,  34,  33,  16,  42,   5,  47,  17,  41,  44,  37,
        24,  66,  77,  81,  70,  53,  75,  73,  52,  74,  43,  89,  57,
        14,   9,  48,  83,  72,  25,  80,  87,  88,  84,  82,  90,  94,
        86,  91,  98,  92,  96,  93,  95,  97, 102, 115, 100,  99,  -1],
      dtype=int64)
```

I found some ages are extra big like 112 and 115 and I found -1 number as a unmeaningful age

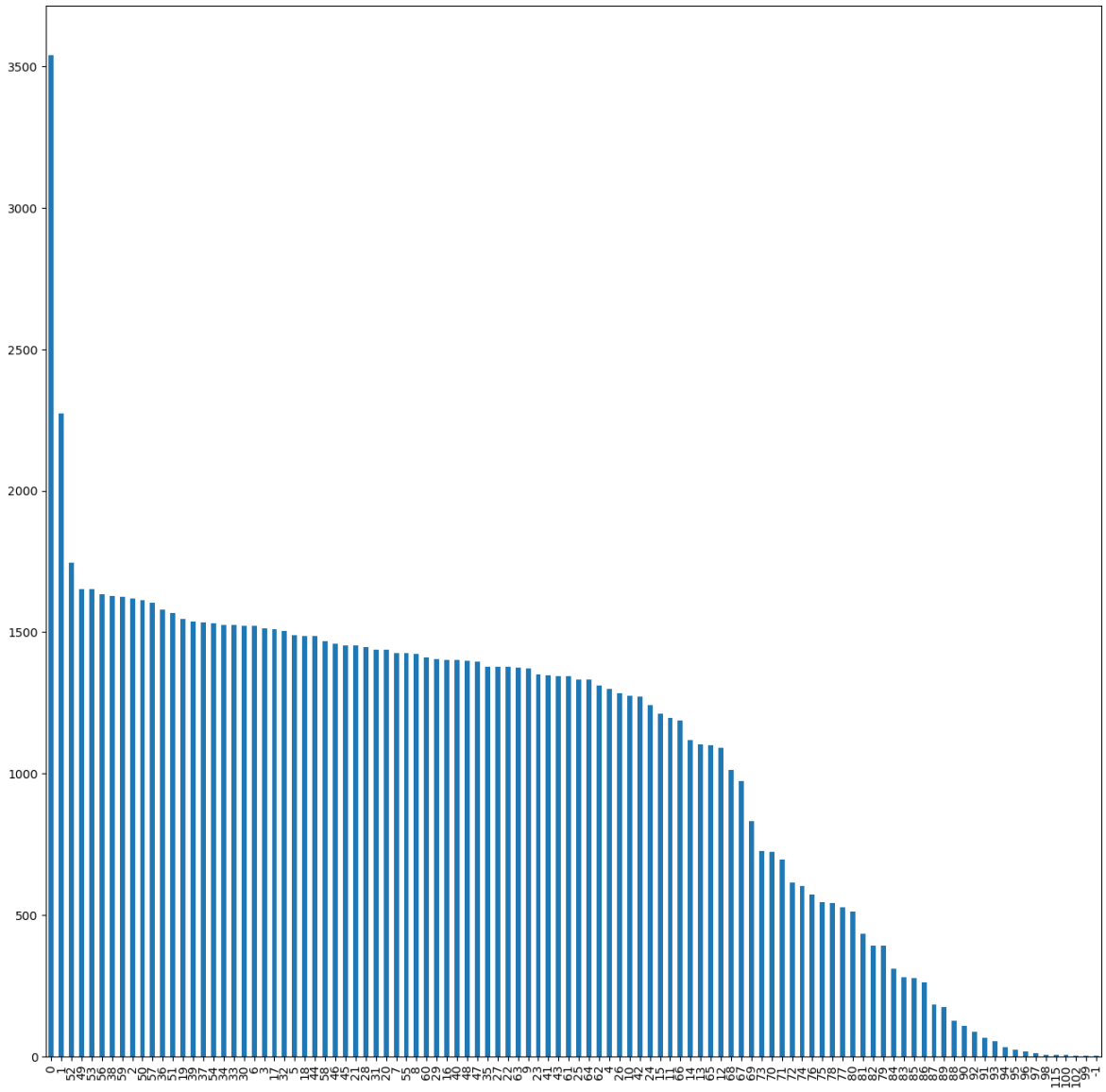
In [7]:
```
# Cheking the outliers of 'Age' column with plot.
df['Age'].plot(kind='box');
```



We can see the big ages as outlier in the plot.

In [8]:
```
#I checked the counts of the ages belons patient.
df.Age.value_counts().plot(kind='bar',figsize=(16,16))
```

Out[8]: <AxesSubplot:>



We can see that most patient count belongs range of age 0-2.

In [9]:
```
#Cheking the uniques of 'Neighbourhood' column
df.Neighbourhood.unique()
```

Out[9]:
```
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 SUÁ',
       'SANTA HELENA', 'ITARARÉ', 'INHANGUETÁ', 'UNIVERSITÁRIO',
       'SÃO JOSÉ', 'REDENÇÃO', 'SANTA CLARA', 'CENTRO', 'PARQUE MOSCOSO',
       'DO MOSCOSO', 'SANTOS DUMONT', 'CARATOÍRA', 'ARIOVALDO FAVALESSA',
       'ILHA DO FRADE', 'GURIGICA', 'JOANA D´ARC', 'CONSOLAÇÃO',
       'PRAIA DO CANTO', 'BOA VISTA', 'MORADA DE CAMBURI', 'SANTA LUÍZA',
       'SANTA LÚCIA', 'BARRO VERMELHO', 'ESTRELINHA', 'FORTE SÃO JOÃO',
       '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 INDUSTRIAL'],
      dtype=object)
```

All names of Neighbourhoods is meaningful.

In [10]:
```
# Cheking uniques of 'Scholarship' coumn for understand  if there is unmeaning
df.Scholarship.unique()
```

Out[10]:
```
array([0, 1], dtype=int64)
```

In [11]:
```
# Cheking uniques of 'Hipertension' coumn for understand  if there is unmeani
df.Hipertension.unique()
```

Out[11]:
```
array([1, 0], dtype=int64)
```

In [12]:
```
# Cheking uniques of 'Diabetes' coumn for understand  if there is unmeaning d
df.Diabetes.unique()
```

Out[12]:
```
array([0, 1], dtype=int64)
```

In [13]:
```
# Cheking uniques of 'Alcolism' coumn for understand  if there is unmeaning d
df.Alcoholism.unique()
```

Out[13]:
```
array([0, 1], dtype=int64)
```

In [14]:
```
# Cheking uniques of 'Handcap' coumn for understand  if there is unmeaning da
df.Handcap.unique()
```

Out[14]:
```
array([0, 1, 2, 3, 4], dtype=int64)
```

In [15]:
```
# Cheking uniques of 'SMS_Received' coumn for understand  if there is unmeani
df.SMS_received.unique()
```

Out[15]:
```
array([0, 1], dtype=int64)
```

In [16]:
```
# Cheking uniques of 'No-Show' coumn for understand  if there is unmeaning da
df['No-show'].unique()
```

Out[16]:
```
array(['No', 'Yes'], dtype=object)
```

In [17]:
```
# check how many rows and columns the data consists of
df.shape
```

Out[17]:
```
(110527, 14)
```

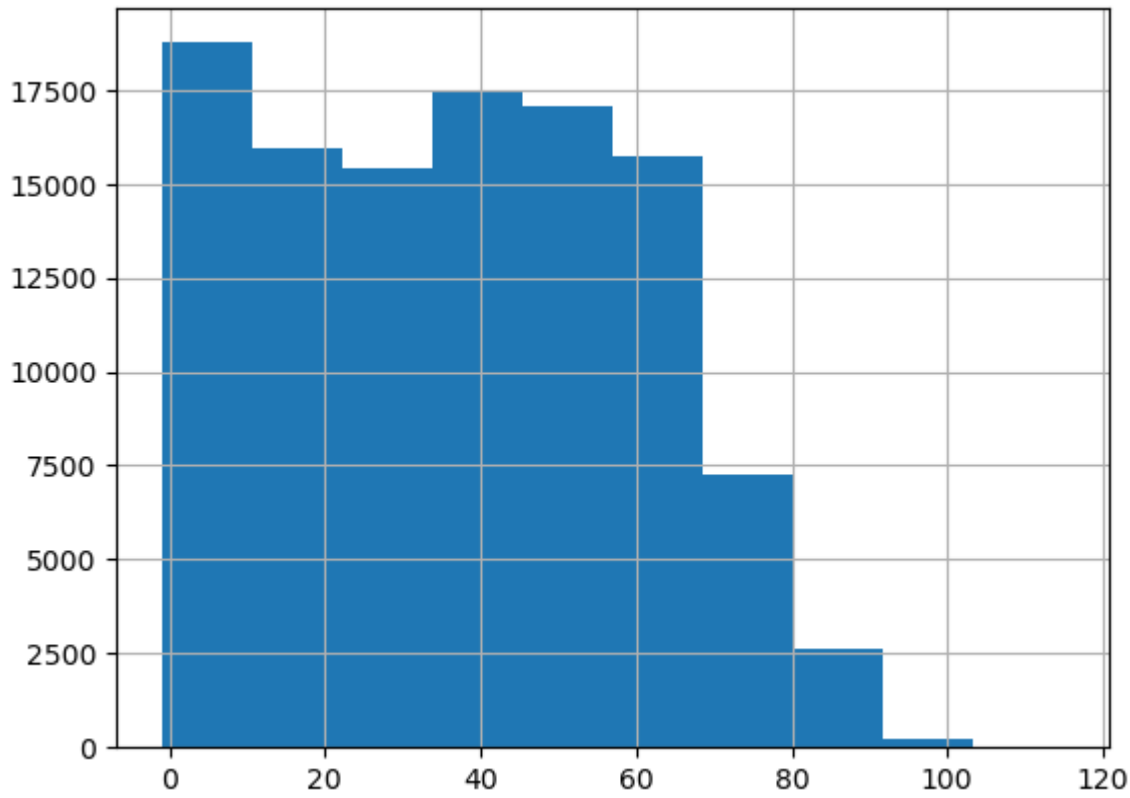
In [18]:
```
# Analyze the data in the Age column statistically
df['Age'].describe()
```

Out[18]:
```
count    110527.000000
mean         37.088874
std          23.110205
min          -1.000000
25%          18.000000
50%          37.000000
75%          55.000000
max         115.000000
Name: Age, dtype: float64
```

I saw that the mean age of the patients is 37. I see std < mean.

In [19]:
```
# Cheking the 'Age column with hist.'
df["Age"].hist()
```

Out[19]: <AxesSubplot:>



I see that it is Right skewed distribution.

Data Cleaning

In [20]:
```
# rename SMS_received to SmsReceived
df.rename(columns={'SMS_received': 'SmsReceived'}, inplace=True)

# rename No-Show to NoShow
df.rename(columns={'No-show': 'NoShow'}, inplace=True)

# confirm changes
df.head(1)
```

Out[20]:

	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

In [21]:
```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

df['AppointmentDay'] = np.array(df['AppointmentDay'].values, dtype='datetime64
df['ScheduledDay'] = np.array(df['ScheduledDay'].values, dtype='datetime64[D]

df['ScheduledDay'] = pd.to_datetime(df['ScheduledDay'])
df['AppointmentDay'] = pd.to_datetime(df['AppointmentDay'])

#confirm changes for 'ScheduledDay' and 'AppointmentDay' column
df.head(1)
```

Out[21]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood
0	2.987250e+13	5642903	F	2016-04-29	2016-04-29	62	JARDIM DA PENHA

In [22]:
```
# confirm changes for data types
df.dtypes
```

Out[22]:
```
PatientId                 float64
AppointmentID               int64
Gender                     object
ScheduledDay       datetime64[ns]
AppointmentDay     datetime64[ns]
Age                         int64
Neighbourhood              object
Scholarship                 int64
Hipertension                int64
Diabetes                    int64
Alcoholism                  int64
Handcap                     int64
SmsReceived                 int64
NoShow                     object
dtype: object
```

In [23]:
```
#Cheking the difference between appointment Day and Scheduled Day
df['RestDaysToApp'] = (df['AppointmentDay'] - df['ScheduledDay']) / np.timede

df['RestDaysToApp'].dtypes
```

Out[23]:
```
dtype('float64')
```

In [24]:
```
# Converting the type of RestDaysToApp from ' float64' to 'int64'.
df['RestDaysToApp'] = df['RestDaysToApp'].astype("int64")
df['RestDaysToApp'].dtypes
```

Out[24]:
```
dtype('int64')
```

In [25]:
```
# Check uniques of 'RestDaysToApp' column to catch the unmeaningful values.
df.RestDaysToApp.unique()
```

Out[25]:
```
array([  0,   2,   3,   1,   4,   9,  29,  10,  23,  11,  18,  17,  14,
        28,  24,  21,  15,  16,  22,  43,  30,  31,  42,  32,  56,  45,
        46,  39,  37,  38,  44,  50,  60,  52,  53,  65,  67,  91,  66,
        84,  78,  87, 115, 109,  63,  70,  72,  57,  58,  51,  59,  41,
        49,  73,  64,  20,  33,  34,   6,  35,  36,  12,  13,  40,  47,
         8,   5,   7,  25,  26,  48,  27,  19,  61,  55,  62, 176,  54,
        77,  69,  83,  76,  89,  81, 103,  79,  68,  75,  85, 112,  -1,
        80,  86,  98,  94, 142, 155, 162, 169, 104, 133, 125,  96,  88,
        90, 151, 126, 127, 111, 119,  74,  71,  82, 108, 110, 102, 122,
       101, 105,  92,  97,  93, 107,  95,  -6, 139, 132, 179, 117, 146,
       123], dtype=int64)
```

I see the unmeaningful results like '-6' and '-1'. The differences can not be the number that is under the '0'.So i have to drop them.

In [26]:
```
#Dropping the negative results
df.drop(df[df['RestDaysToApp']<0].index,inplace=True)
```

In [27]:
```
# Confirm changes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 110522 entries, 0 to 110526
Data columns (total 15 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   PatientId       110522 non-null  float64
 1   AppointmentID   110522 non-null  int64
 2   Gender          110522 non-null  object
 3   ScheduledDay    110522 non-null  datetime64[ns]
 4   AppointmentDay  110522 non-null  datetime64[ns]
 5   Age             110522 non-null  int64
 6   Neighbourhood   110522 non-null  object
 7   Scholarship     110522 non-null  int64
 8   Hipertension    110522 non-null  int64
 9   Diabetes        110522 non-null  int64
 10  Alcoholism      110522 non-null  int64
 11  Handcap         110522 non-null  int64
 12  SmsReceived     110522 non-null  int64
 13  NoShow          110522 non-null  object
 14  RestDaysToApp   110522 non-null  int64
dtypes: datetime64[ns](2), float64(1), int64(9), object(3)
memory usage: 13.5+ MB
```

In [28]:
```
#Check outliers of Age
df_under_zero = df.query('Age < 0')
df_under_zero.head()
```

Out[28]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourh
99832	4.659430e+14	5775010	F	2016-06-06	2016-06-06	-1	ROI

In [29]:
```
#dropping outliers of Age values that is less than "0".
df.drop(df[df['Age']<0].index,inplace=True)


#confirm changes
df_under_zero = df.query('Age < 0')
df_under_zero.head()
```

Out[29]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Sch

In [30]:
```
#Cheking the outliers of Age values that more than '100'.
df_under_zero = df.query('Age > 100')
df_under_zero.head()
```

Out[30]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourh
58014	9.762950e+14	5651757	F	2016-05-03	2016-05-03	102	CONQU
63912	3.196320e+13	5700278	F	2016-05-16	2016-05-19	115	ANDORIN
63915	3.196320e+13	5700279	F	2016-05-16	2016-05-19	115	ANDORIN
68127	3.196320e+13	5562812	F	2016-04-08	2016-05-16	115	ANDORIN
76284	3.196320e+13	5744037	F	2016-05-30	2016-05-30	115	ANDORIN

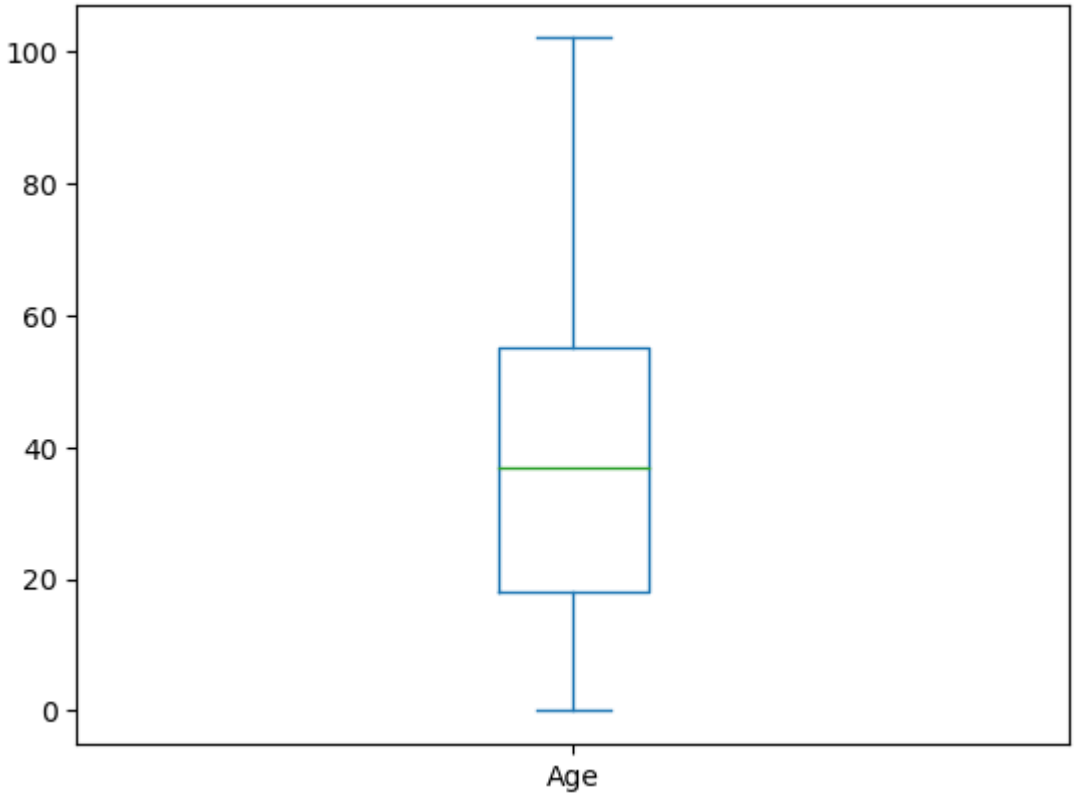
In [31]:
```
#Dropping the outliers of Age  that more than'102'.
df.drop(df[df['Age']>102].index,inplace=True)
```

In [32]:
```
#Confirm changes
df_under_zero = df.query('Age >100')
df_under_zero.head()
```

Out[32]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourh
58014	9.762950e+14	5651757	F	2016-05-03	2016-05-03	102	CONQU
90372	2.342840e+11	5751563	F	2016-05-31	2016-06-02	102	MARIA OI

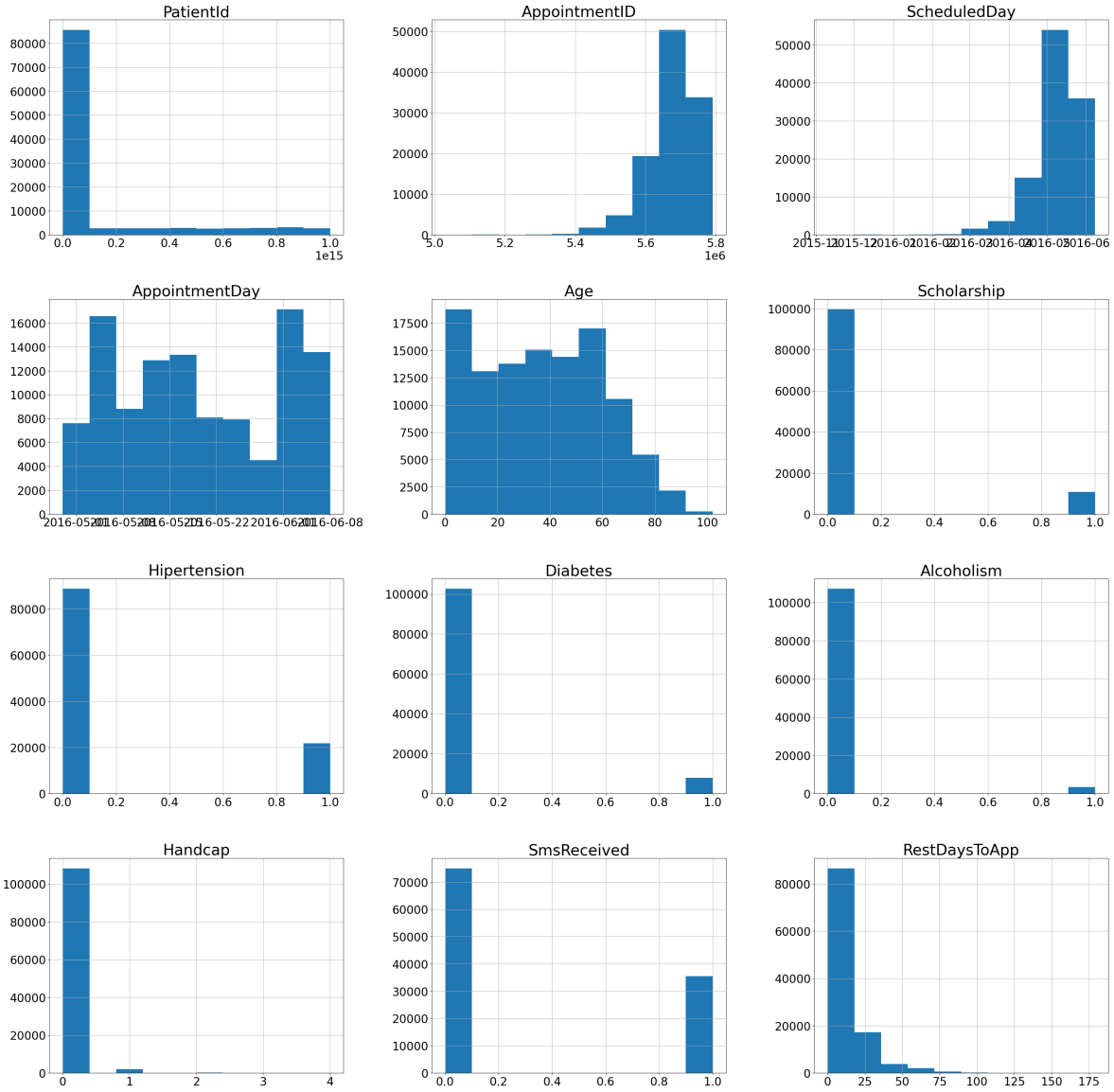
In [33]:
```
# Confirm changes for outliers of 'Age' Column.
df['Age'].plot(kind='box');
```



I can see that the oultliers was dissapeared when I compare with previous plot.

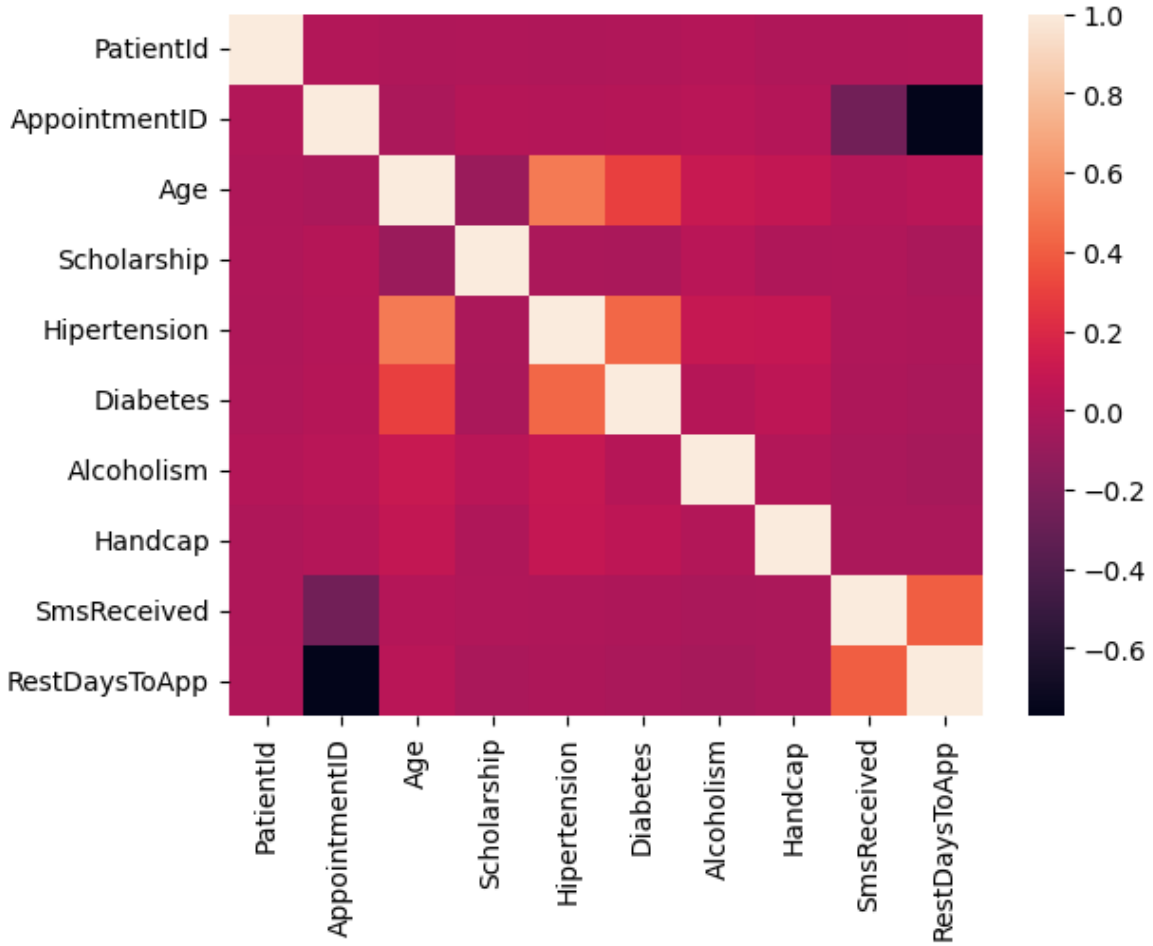
In [34]:
```
# Plot the hist for all columns as a big aspect.
params = {'axes.titlesize':'32',
          'xtick.labelsize':'24',
          'ytick.labelsize':'24'}
plt.rcParams.update(params)
df.hist(figsize=(40, 40))
```

Out[34]:
```
array([[<AxesSubplot:title={'center':'PatientId'}>,
        <AxesSubplot:title={'center':'AppointmentID'}>,
        <AxesSubplot:title={'center':'ScheduledDay'}>],
       [<AxesSubplot:title={'center':'AppointmentDay'}>,
        <AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'Scholarship'}>],
       [<AxesSubplot:title={'center':'Hipertension'}>,
        <AxesSubplot:title={'center':'Diabetes'}>,
        <AxesSubplot:title={'center':'Alcoholism'}>],
       [<AxesSubplot:title={'center':'Handcap'}>,
        <AxesSubplot:title={'center':'SmsReceived'}>,
        <AxesSubplot:title={'center':'RestDaysToApp'}>]], dtype=object)
```



In [35]:
```
# Reset the plot parameters for future plots.
plt.rcParams.update(plt.rcParamsDefault)
```

In [36]:
```
# Check the correlation.
sns.heatmap(df.corr())
plt.show()
```



There is a positive correlation between age and hypertension. There is a positive correlation between age and diabetes. However, this information is not sufficient to reach a clear conclusion.

Explarotary Data Analysis

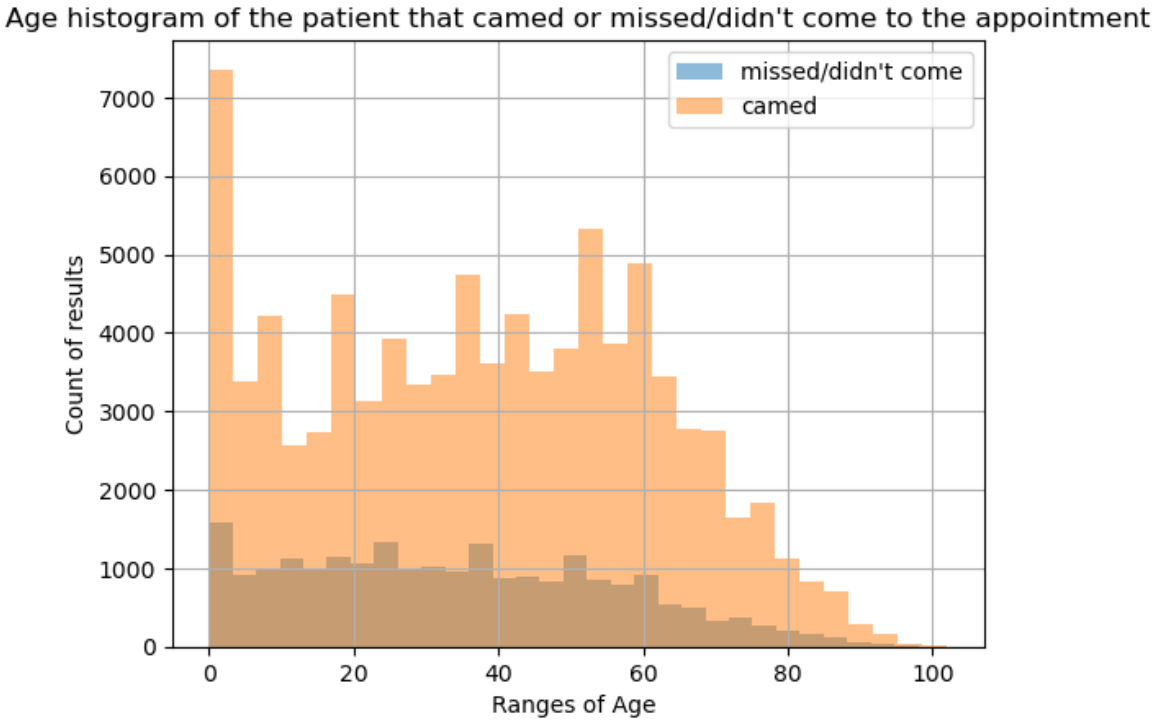
Question 1(What is the relationship between age and no-show?)

In [37]:
```
# Check the average of Ages regarding the No Showed results.
df_age=df.groupby('NoShow')['Age'].mean()
df_age
```

Out[37]:
```
NoShow
No     37.788753
Yes    34.307023
Name: Age, dtype: float64
```

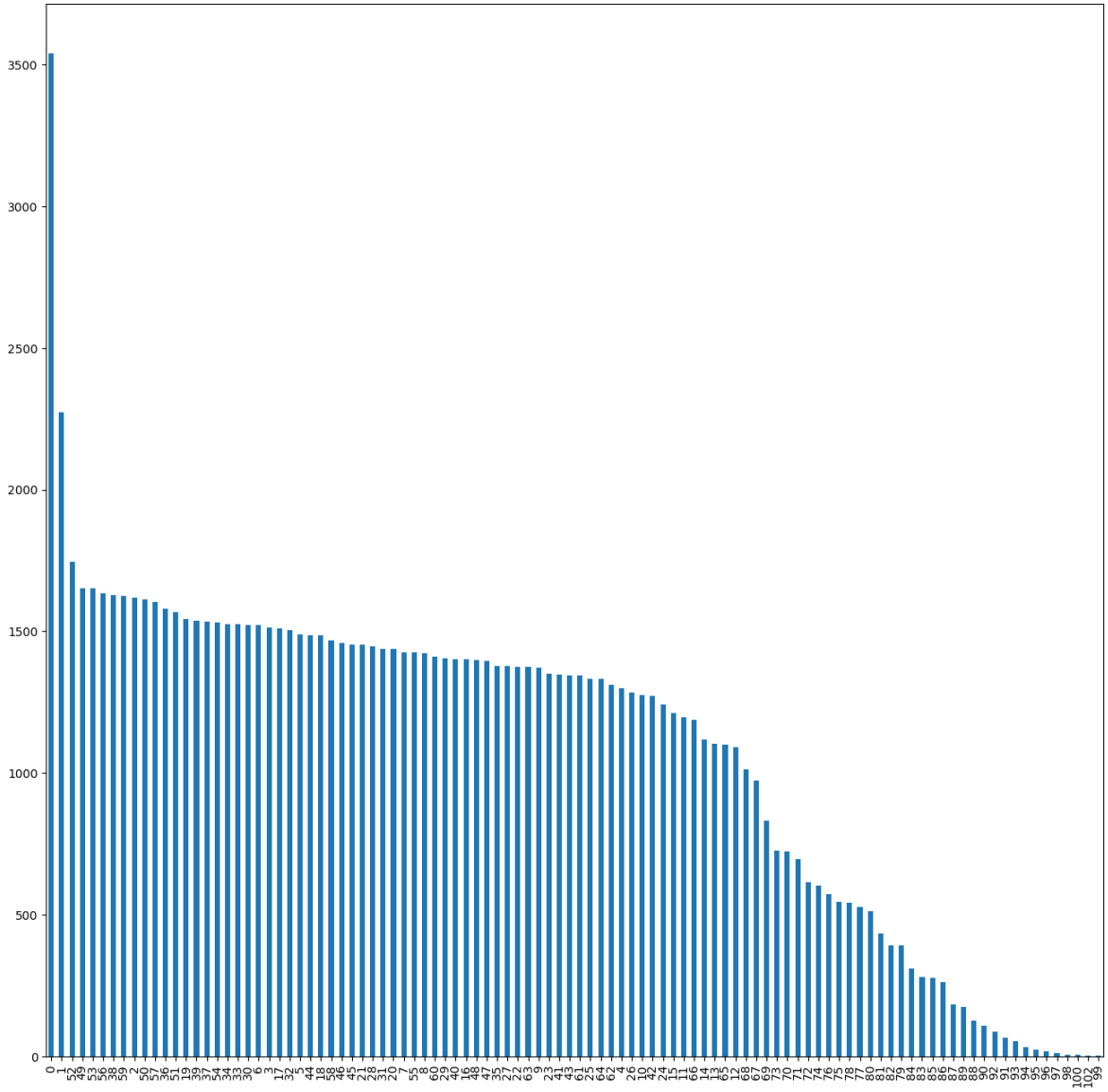
The average age of those who did not attend/miss their appointment is younger than the average age of those who go.

In [38]:
```
#Plot the histogram to understand the distributions of Ages the No-Show situa
df.query('NoShow=="Yes"' ).Age.hist(alpha=0.5, bins=30,label="missed/didn't c
df.query('NoShow=="No"' ).Age.hist(alpha=0.5, bins=30,label="camed")
plt.legend();
plt.title("Age histogram of the patient that camed or missed/didn't come to t
plt.xlabel("Ranges of Age ")
plt.ylabel("Count of results")
plt.show();
```



The number of coming to appointments is higher in individuals over the age of 40. We can say that the number of patients who come to appointments for 0-year-old babies is overwhelmingly superior to those who do not. After the age of 60, the number of missed appointments has decreased.

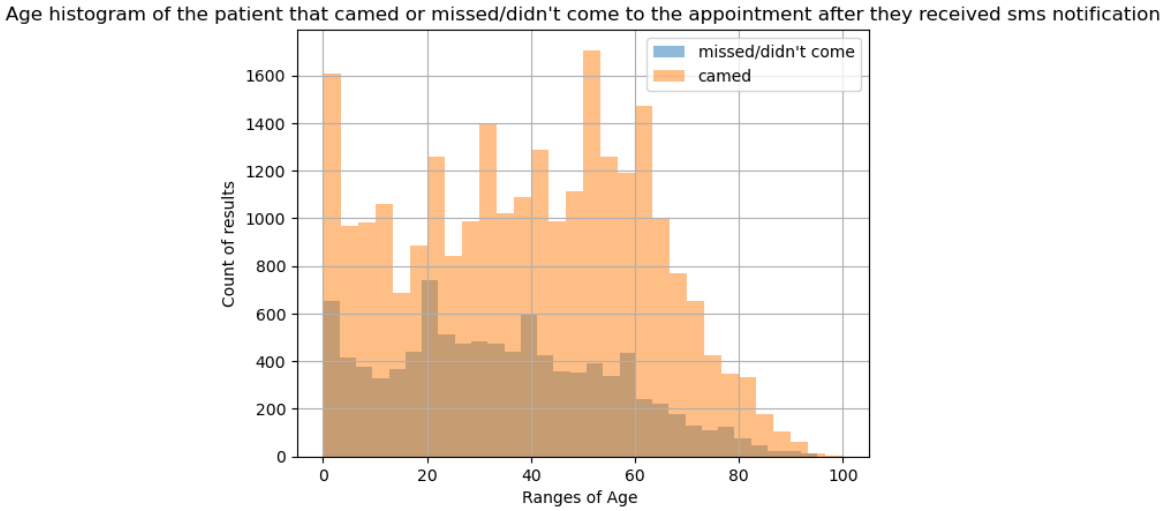
In [39]:
```
#I checked the counts of the ages belons patient.
df.Age.value_counts().plot(kind='bar',figsize=(16,16))
plt.show()
```



The ages with the highest number of patients are 0 and 1 year old babies.

Research Question 2 (What is the effect of reporting the appointment by sms in different age groups on the NoShow value?)

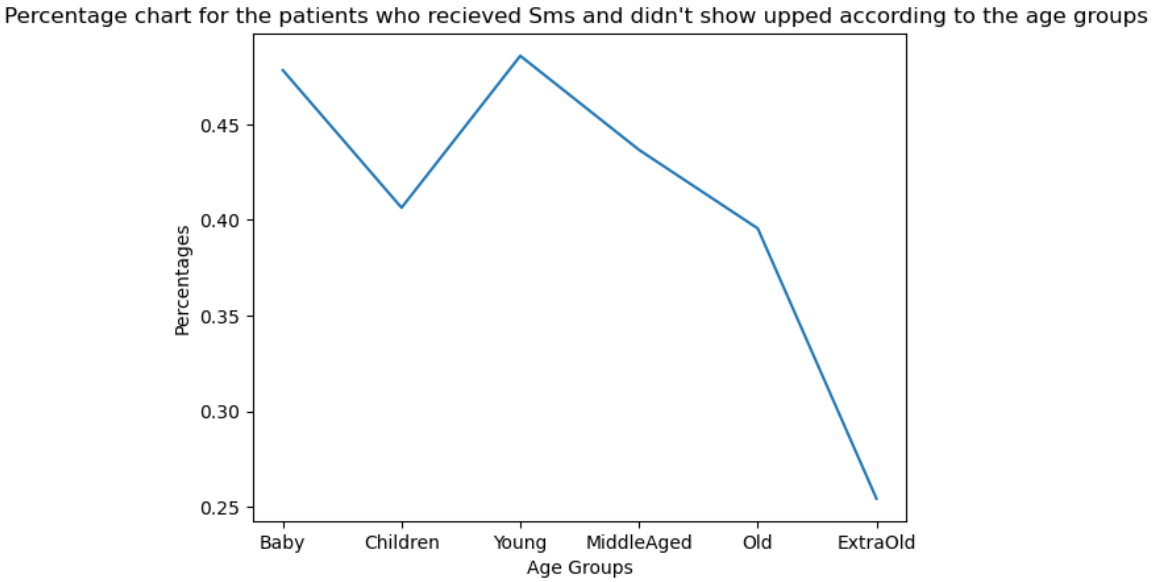
In [40]:
```
#Plot an histogram to understand the differences between camed and didn't come
df.query('NoShow=="Yes" &  SmsReceived==1').Age.hist(alpha=0.5, bins=30,label=
df.query('NoShow=="No"  & SmsReceived==1').Age.hist(alpha=0.5, bins=30,label=
plt.legend();
plt.title("Age histogram of the patient that camed or missed/didn't come to th
plt.xlabel("Ranges of Age ")
plt.ylabel("Count of results")
plt.show();
```



It can be said that receiving an SMS notification is effective in the number of people over the age of age coming to an appointment.However, a more detailed investigation is required.

In [41]:
```
# I adapted the code that i found from a website to make groups from Ages and
#Source:https://stackoverflow.com/questions/62768980/pandas-split-ages-by-grou
def AgeGrouper(df):
    df["AgeGroups"] = pd.cut(x=df['Age'], bins=[0,2,18,40,60,90,120], labels=
    return df

# Call function to group ages and create a new column in No-Show Data
df=AgeGrouper(df)
```

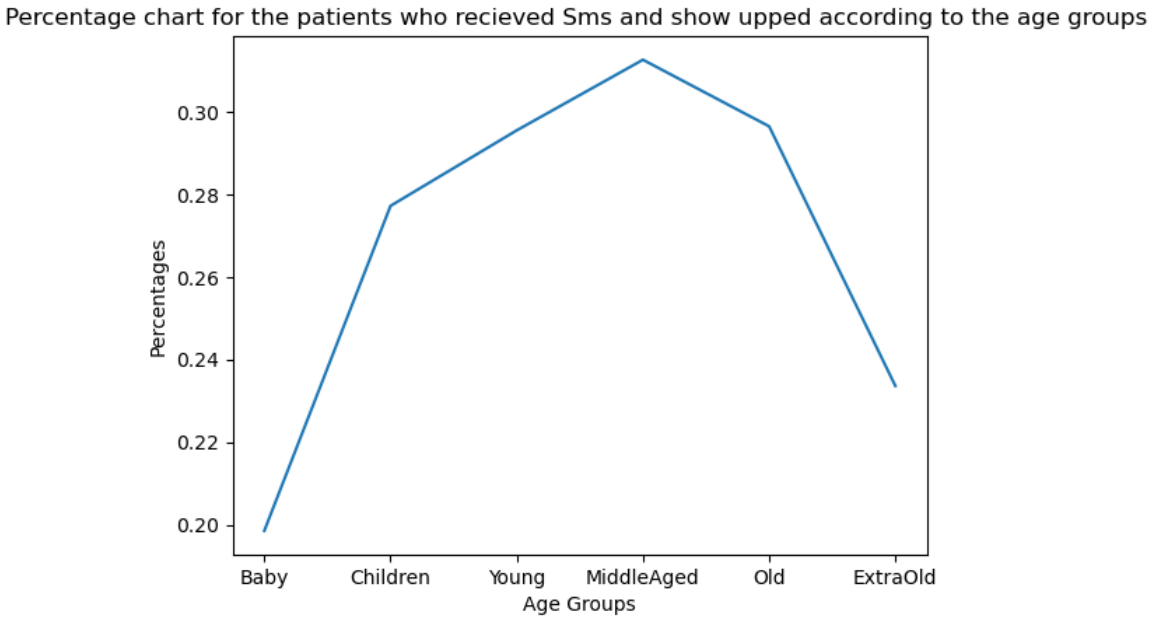
In [42]:
```
#I filtered the data of the people who did not come to the appointment using
df_No_Showed=df.query('NoShow=="Yes"' )

# I grouped the data I filtered according to age groups and looked at the per
# I plotted these percentages to see them better.

df_No_Showed.groupby('AgeGroups')['SmsReceived'].mean().plot()
plt.title("Percentage chart for the patients who recieved Sms and didn't show
plt.xlabel("Age Groups")
plt.ylabel("Percentages")
plt.show()
```



We see that the highest rate of not receiving an SMS notification is in the infant and young age groups.

In [43]:

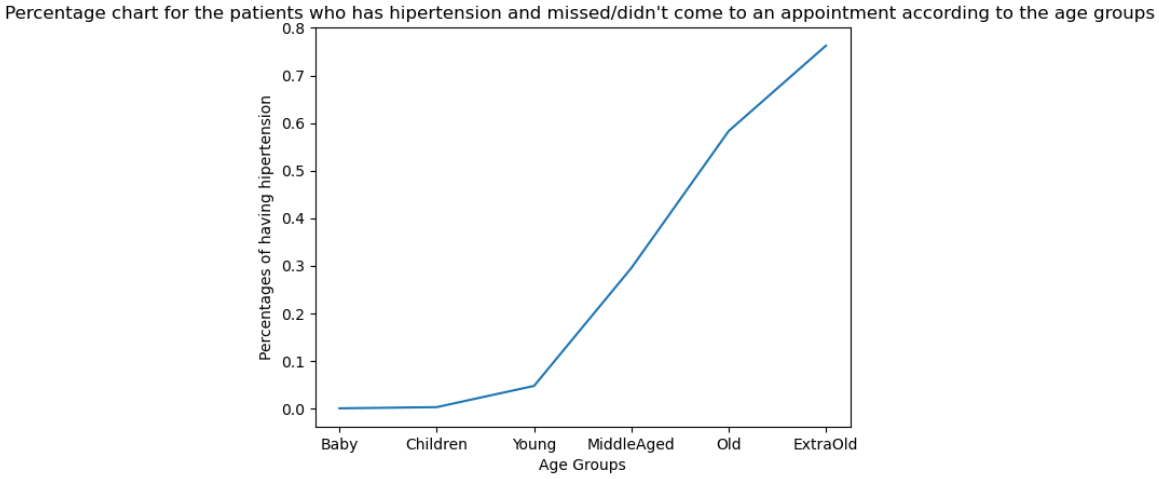
```
# I calculated the percentage of receiving sms notifications in patients from
# I have plotted these percentages to see them better.

df_No_Showed=df.query('NoShow=="No"' )
df_No_Showed.groupby('AgeGroups')['SmsReceived'].mean().plot()
plt.title("Percentage chart for the patients who recieved Sms and show upped a
plt.xlabel("Age Groups")
plt.ylabel("Percentages")
plt.show()
```



The proportion of people who receive SMS notifications is the highest in the middle age group.

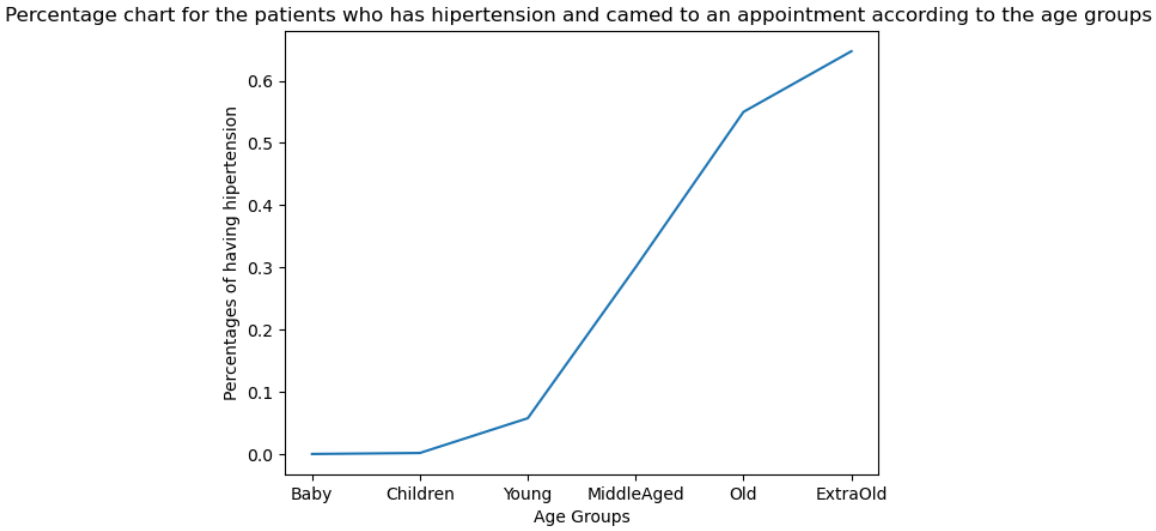
QUESTION 3(What is the effect of hypertension on the absence of appointments from patients who have received SMS notifications according to age groups?)

In [44]:
```
# Plot the status of hypertension according to age groups of patients who do n
df_No_Showed=df.query('NoShow=="Yes"' )
df_No_Showed.groupby('AgeGroups')['Hipertension'].mean().plot()
plt.title("Percentage chart for the patients who has hipertension and missed/d
plt.xlabel("Age Groups")
plt.ylabel("Percentages of having hipertension")
plt.show()
```



The percentage of old and extra old patients who do not come to an appointment to have hypertension disease is over 60 percent.

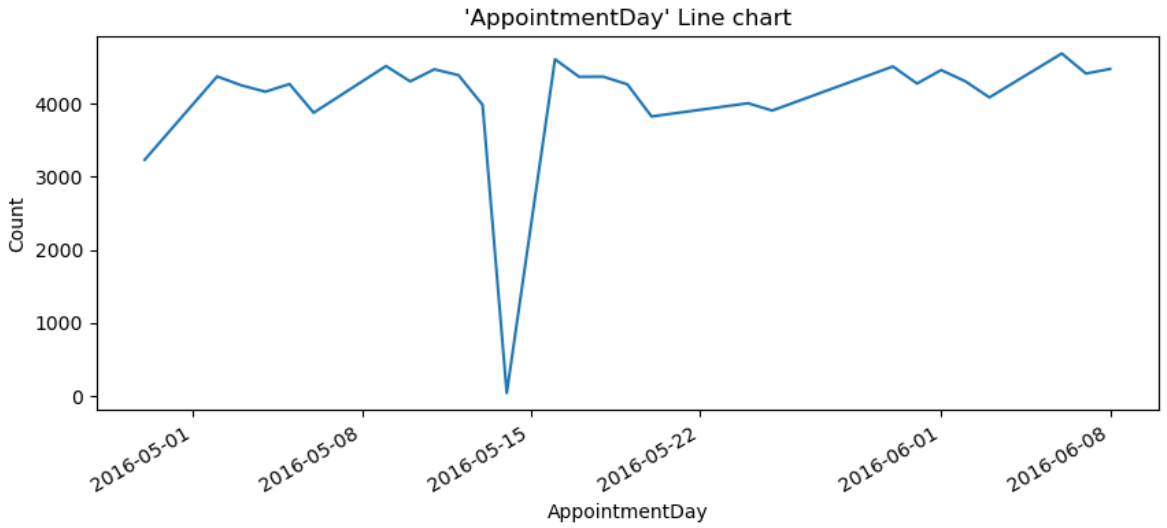
In [45]:
```
df_No_Show=df.query('NoShow=="No"' )
df_No_Show.groupby('AgeGroups')['Hipertension'].mean().plot()
plt.title("Percentage chart for the patients who has hipertension and camed t
plt.xlabel("Age Groups")
plt.ylabel("Percentages of having hipertension")
plt.show()
```



The percentage of old and extra old patients who come to an appointment to have hypertension disease is over 55 percent.

Question 4 (What is the relationship between the appointment dates and the total number of appointments for that day?)

In [46]:
```
# check the counts of Appointment days with plot.
df.AppointmentDay.value_counts().plot(kind="line",figsize=(10,4));
plt.xlabel("AppointmentDay")
plt.ylabel("Count")
plt.title("'AppointmentDay' Line chart")
plt.show()
```



I observe a sharp decrease in the number of appointments made for the date '2016-05-14'

In [47]:
```
# Check the total counts of appointments regarding the dates.
df.AppointmentDay.value_counts()
```

Out[47]:
```
2016-06-06    4691
2016-05-16    4612
2016-05-09    4519
2016-05-30    4513
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    4371
2016-06-02    4310
2016-05-10    4308
2016-05-31    4279
2016-05-05    4272
2016-05-19    4268
2016-05-03    4255
2016-05-04    4167
2016-06-03    4089
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
```

Appointments created for dates generally vary between 3265 and 5691.Date:'2016-05-14' is an exception with 39 appointment number.

Conclusions

1) The middle age group(40+) is the age group in which SMS notification most affects their coming to an appointment.

2) Patients who are in infancy and in the 40+ age group have a higher number of coming to appointments.

3) The average age of people who came to an appointment is 34. The average age of the people missed/didn't come to an appointment is 37. The average age of those who missed/didn't come to an appointment is minder than the average age of those who go.

4) Although hypertension patients over the age of 60 have received SMS notification, the rate of not going to an appointment is higher than those who received SMS notification from the same age group without hypertension.

5)

I observe a sharp decrease in the number of appointments made for the date '2016-05-14' Appointments created for dates generally vary between 3265 and 5691.Date:'2016-05-14' is an exception with 39 appointment number.

The Limitations of No-Show Data Analyze:

I investigated the effect of patients receiving SMS notifications according to their age groups on No-Show Data. And I also checked the status of going or not going to an appointment only according to age distribution. And I also cheked the effect of patients having hypertension on NoShow. However, I do not think that these reviews reveal the exact reason affecting NoShow data. By examining the data in the 'RestDaysToApp' column, I can examine the effect of the number of days left until this appointment date on the 'NoShow'column. In addition, the places where people live may have affected whether or not they go to an appointment. The rates of going to an appointment can be examined here. Finally, these examinations are also insufficient in the final result. I think that having more detailed data is necessary for a definitive result. For example, details such as the distance from the patient's house to the hospital.