Project: Investigate a Dataset (NoshowAppointment Dataset)

Table of Contents

Introduction to the Dataset

This dataset contains medical appointment details for 110,527 Brazilians. The focus of the data is to determine if patients who booked appointment dates showed up for their appointments. There are 14 associated variables (characteristics) in the dataset. These will be explained in much more detail below.

Description of the column variables

01 - PatientId

The unique ID for identifying a patient is defined in this column.

02 - AppointmentID

An appointment's unique ID is defined in this column.

03 - Gender

This pertains to their gender, either Male or Female. The ratio is higher among females because women are more concerned about their health than men.

04 - ScheduledDay

This is the appointment day when they have to visit the doctor.

05 - AppointmentDay

The day someone called or registered the appointment.

06 - Age

The patient's age.

07 - Neighbourhood

Here is where the appointment takes place.

08 - Scholarship

This indicates if the patient has a scholarship. Scholarship in this context refers to a social welfare program the Government of Brazil offers. It provides financial assistance to poor Brazilian families as long as their children attend school and are vaccinated. '1' indicates such a patient has the scholarship, while '0' indicates the patient doesn't have the scholarship. Check here for more details.

09 - Hipertension

Refers to a patient diagnosed with hypertension. '1' signifies yes, while '0' signifies no.

10 - Diabetes

Refers to a patient diagnosed with diabetes. '1' indicates yes, while '0' indicates no.

11 - Alcoholism

Applied to a patient who is an alcoholic. '1' shows that the patient is an alcoholic, while '0' shows that the patient is not an alcoholic.

12 - Handcap

Does the patient have a disability? The values in this column range from 0 to 4.

13 - SMS_received

Does the patient receive one or more messages concerning the appointment? '1' indicates one SMS was sent. '0' indicates SMS was not sent.

14 - No-show

It indicates whether the patients attended their appointments or not, which is one of the most important aspects of the dataset. 'No' signifies that the patient showed up for the appointment, while 'Yes' signifies that the patient didn't show up for their appointment.

Research Questions

1. What is the ratio of people present or absent for the appointment?
2. What gender booked more medical appointments?
3. Among those scheduled for appointments, what is the most common disease?
4. Which age is most affected by the diseases?
5. What gender is most affected by the diseases?

In [1]:
```
# Import all the libraries necessary for investigating the dataset

from datetime import datetime, timedelta

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

%matplotlib inline
```

Data Wrangling

Data wrangling involves the process of cleaning, organizing, visualizing, and transforming the data set.

The steps are as follows:

- **General Properties**
- **Data Cleaning**

General Properties

This involves inspecting in detail the properties of the dataset.

In [2]:
```
# Load the dataset and represent our data frame with appointment_df

appointment_df = pd.read_csv("noshowappointments-kagglev2-may-2016.csv")
appointment_df.head(10)
```

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
5	9.598513e+13	5626772	F	2016-04-27T08:36:51Z	2016-04-29T00:00:00Z	76	REPÚBLICA
6	7.336882e+14	5630279	F	2016-04-27T15:05:12Z	2016-04-29T00:00:00Z	23	GOIABEIRAS
7	3.449833e+12	5630575	F	2016-04-27T15:39:58Z	2016-04-29T00:00:00Z	39	GOIABEIRAS
8	5.639473e+13	5638447	F	2016-04-29T08:02:16Z	2016-04-29T00:00:00Z	21	ANDORINHAS
9	7.812456e+13	5629123	F	2016-04-27T12:48:25Z	2016-04-29T00:00:00Z	19	CONQUISTA

In [3]:
```
# Check the dataset information.
appointment_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
```

In [4]:
```
# Check for missing values
appointment_df.isnull().sum()
```

Out[4]:
```
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
```

In [5]:
```
# Check the numbers of rows and columns respectively
appointment_df.shape
```

Out[5]:
```
(110527, 14)
```

In [6]:
```
# Number of unique PatientId
appointment_df["PatientId"].nunique()
```

Out[6]:
```
62299
```

In [7]:
```
# Number of unique AppointmentID
appointment_df["AppointmentID"].nunique()
```

Out[7]:
```
110527
```

> **The above dataset shows that:**
>
> 1. There are 110,527 observations and 14 column features.
> 2. There are no missing values, indicating each column contains a complete set of values.

Data Cleaning

This process involves removing unnecessary columns, changing column names and values to more meaningful ones. Below are the steps that will be taken:

1. Clean up this data to make it more useful for our analysis by removing columns that are not necessary, such as 'PatientId' and 'AppointmentID'.
2. Change the name of the column 'No-show' to 'present' and its column values to True (if present) and Fulse (if absent).
3. Change 'Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism', and 'Handcap' columns to lowercase and resolve typo errors.
4. Finally, we will change the "ScheduledDate" and "AppointmentDay" datatypes to DateTime.

In [8]:
```
# Drop PatientId and AppointmentID columns

appointment_df.drop(["PatientId", "AppointmentID"], axis=1, inplace=True)
appointment_df.head(10)
```

Out[8]:

	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	D
0	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	0	1	
1	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	0	
2	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA	0	0	
3	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI	0	0	
4	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	1	
5	F	2016-04-27T08:36:51Z	2016-04-29T00:00:00Z	76	REPÚBLICA	0	1	
6	F	2016-04-27T15:05:12Z	2016-04-29T00:00:00Z	23	GOIABEIRAS	0	0	
7	F	2016-04-27T15:39:58Z	2016-04-29T00:00:00Z	39	GOIABEIRAS	0	0	
8	F	2016-04-29T08:02:16Z	2016-04-29T00:00:00Z	21	ANDORINHAS	0	0	
9	F	2016-04-27T12:48:25Z	2016-04-29T00:00:00Z	19	CONQUISTA	0	0	

In [9]:
```
# Replace the column 'No-show' name with present.

appointment_df.rename(columns={"No-show": "present"}, inplace=True)
appointment_df.head(3)
```

Out[9]:

	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	D
0	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	0	1	
1	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	0	
2	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA	0	0	

In [10]:
```
# The row labels for the "met_appointment" column need to be changed to imp
# i.e. Yes to False, and No to True.

appointment_df["present"] = appointment_df["present"].str.replace("No", "Tr
appointment_df["present"].head(10)
```

Out[10]:
```
0     True
1     True
2     True
3     True
4     True
5     True
6    False
7    False
8     True
9     True
Name: present, dtype: object
```

In [11]:
```
# Find the unique values in 'Gender' column
appointment_df["Gender"].unique()
```

Out[11]:
```
array(['F', 'M'], dtype=object)
```

In [12]:
```
# Find the unique values in 'Scholarship' column
appointment_df["Scholarship"].unique()
```

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

In [13]:
```
# Find the unique values in 'Hipertension' column
appointment_df["Hipertension"].unique()
```

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

In [14]:
```
# Find the unique values in 'Alcoholism' column
appointment_df["Alcoholism"].unique()
```

Out[14]:
```
array([0, 1])
```

In [15]:
```
# Find the unique values in 'Handcap' column
appointment_df["Handcap"].unique()
```

Out[15]:
```
array([0, 1, 2, 3, 4])
```

In [16]:
```
# Find the unique values in 'SMS_received' column
appointment_df["SMS_received"].unique()
```

Out[16]:
```
array([0, 1])
```

In [17]:
```
# Find the unique values in 'present' column
appointment_df["present"].unique()
```

Out[17]:
```
array(['True', 'False'], dtype=object)
```

In [18]:
```
# Define a function to count the unique values for each column


def countnum(i):
    print(i.value_counts())
```

In [19]:
```
# Check for 'Neighbourhood'
countnum(appointment_df["Neighbourhood"])
```

```
JARDIM CAMBURI                 7717
MARIA ORTIZ                    5805
RESISTÊNCIA                    4431
JARDIM DA PENHA                3877
ITARARÉ                        3514
                               ...
ILHA DO BOI                      35
ILHA DO FRADE                    10
AEROPORTO                         8
ILHAS OCEÂNICAS DE TRINDADE       2
PARQUE INDUSTRIAL                 1
Name: Neighbourhood, Length: 81, dtype: int64
```

In [20]:
```
# Check for 'present'
countnum(appointment_df["present"])
```

```
True     88208
False    22319
Name: present, dtype: int64
```

In [21]:
```
# Convert the dtypes of the appointment and schedule days to datetime datat

appointment_df["ScheduledDay"] = pd.to_datetime(appointment_df["ScheduledDa
appointment_df["AppointmentDay"] = pd.to_datetime(appointment_df["Appointme
appointment_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Gender          110527 non-null  object
 1   ScheduledDay    110527 non-null  datetime64[ns, UTC]
 2   AppointmentDay  110527 non-null  datetime64[ns, UTC]
 3   Age             110527 non-null  int64
 4   Neighbourhood   110527 non-null  object
 5   Scholarship     110527 non-null  int64
 6   Hipertension    110527 non-null  int64
 7   Diabetes        110527 non-null  int64
 8   Alcoholism      110527 non-null  int64
 9   Handcap         110527 non-null  int64
 10  SMS_received    110527 non-null  int64
 11  present         110527 non-null  object
dtypes: datetime64[ns, UTC](2), int64(7), object(3)
memory usage: 10.1+ MB
```

In [22]:
```
# Change 'Scholarship', 'Hipertension', 'Diabetes', 'Alcoholism',
# and 'Handcap' columns to lowercase and correct spellings.

appointment_df = appointment_df.rename(columns=lambda x: x.lower())

appointment_df.rename(
```

```
    columns={
        "hipertension": "hypertension",
        "scheduledday": "scheduled_day",
        "appointmentday": "appointment_day",
        "handcap": "handicap",
    },
    inplace=True,
)
appointment_df.head(10)
```

Out[22]:

	gender	scheduled_day	appointment_day	age	neighbourhood	scholarship	hypertension	d
0	F	2016-04-29 18:38:08+00:00	2016-04-29 00:00:00+00:00	62	JARDIM DA PENHA	0	1	
1	M	2016-04-29 16:08:27+00:00	2016-04-29 00:00:00+00:00	56	JARDIM DA PENHA	0	0	
2	F	2016-04-29 16:19:04+00:00	2016-04-29 00:00:00+00:00	62	MATA DA PRAIA	0	0	
3	F	2016-04-29 17:29:31+00:00	2016-04-29 00:00:00+00:00	8	PONTAL DE CAMBURI	0	0	
4	F	2016-04-29 16:07:23+00:00	2016-04-29 00:00:00+00:00	56	JARDIM DA PENHA	0	1	
5	F	2016-04-27 08:36:51+00:00	2016-04-29 00:00:00+00:00	76	REPÚBLICA	0	1	
6	F	2016-04-27 15:05:12+00:00	2016-04-29 00:00:00+00:00	23	GOIABEIRAS	0	0	
7	F	2016-04-27 15:39:58+00:00	2016-04-29 00:00:00+00:00	39	GOIABEIRAS	0	0	
8	F	2016-04-29 08:02:16+00:00	2016-04-29 00:00:00+00:00	21	ANDORINHAS	0	0	
9	F	2016-04-27 12:48:25+00:00	2016-04-29 00:00:00+00:00	19	CONQUISTA	0	0	

#

> Now, let's check the age variable to see if there are any quality issues. To accomplish this:
>
> 1. We will use the 'countnum' function we have previously defined.
> 2. Create summary statistics using the describe function.

In [23]:

```
# Check edited information about the Dataset
appointment_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   gender           110527 non-null  object
 1   scheduled_day    110527 non-null  datetime64[ns, UTC]
 2   appointment_day  110527 non-null  datetime64[ns, UTC]
 3   age              110527 non-null  int64
 4   neighbourhood    110527 non-null  object
 5   scholarship      110527 non-null  int64
 6   hypertension     110527 non-null  int64
 7   diabetes         110527 non-null  int64
 8   alcoholism       110527 non-null  int64
 9   handicap         110527 non-null  int64
 10  sms_received     110527 non-null  int64
 11  present          110527 non-null  object
dtypes: datetime64[ns, UTC](2), int64(7), object(3)
memory usage: 10.1+ MB
```

In [24]:
```
# Check unique values in the age column using the countnum function.
countnum(appointment_df["age"])
```

```
 0      3539
 1      2273
 52     1746
 49     1652
 53     1651
        ...
 115       5
 100       4
 102       2
 99        1
-1         1
Name: age, Length: 104, dtype: int64
```

In [25]:
```
# Review the summary statistics of the age column.
appointment_df["age"].describe()
```

Out[25]:
```
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
```

In [26]:
```
# Check the median age
appointment_df["age"].median()
```

Out[26]:
```
37.0
```

In [27]:
```
# replace age -1 with the median age 37
appointment_df["age"].replace({-1: 37}, inplace=True)
```

In [28]:
```
# Review the summary statistics of the age column to confirm changes.
appointment_df["age"].describe()
```

Out[28]:
```
count    110527.000000
mean         37.089218
std          23.109921
min           0.000000
25%          18.000000
50%          37.000000
75%          55.000000
max         115.000000
Name: age, dtype: float64
```

> We can observe that -1 was documented as one of the patient's ages, and 3539 patients were 0 years old. Since only one person had a wrong age of -1, we changed that value to the median age of 37.

Exploratory Data Analysis (EDA)

This process involves statistical analysis and creating visualizations to address the research questions below.

Research Questions

Our EDA would be based on the following questions:

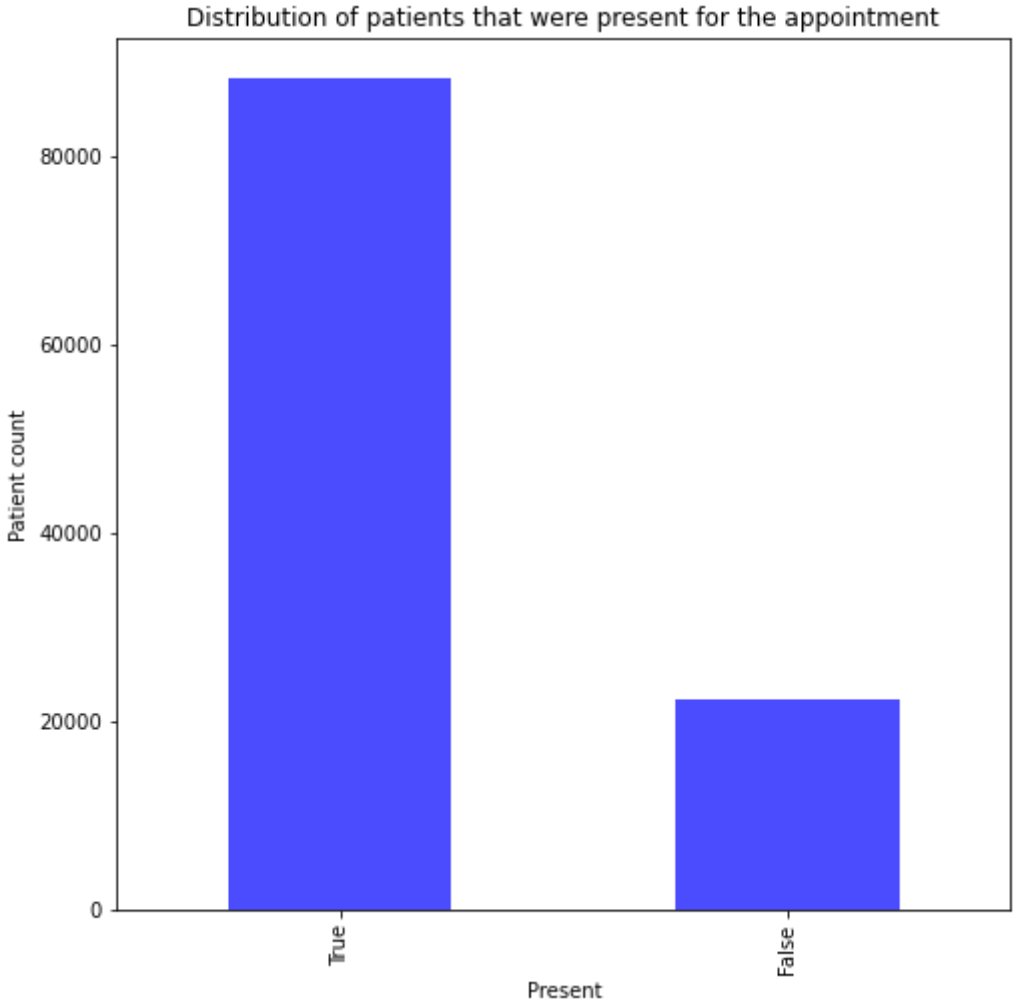
1. What is the ratio of people present or absent for the appointment?
2. What gender booked more medical appointments?
3. Among those scheduled for appointments, what is the most common disease?
4. Which age is most affected by the diseases?
5. What gender is most affected by the diseases?

Note: The diseases are hypertension, diabetes, alcoholism, and handicap.

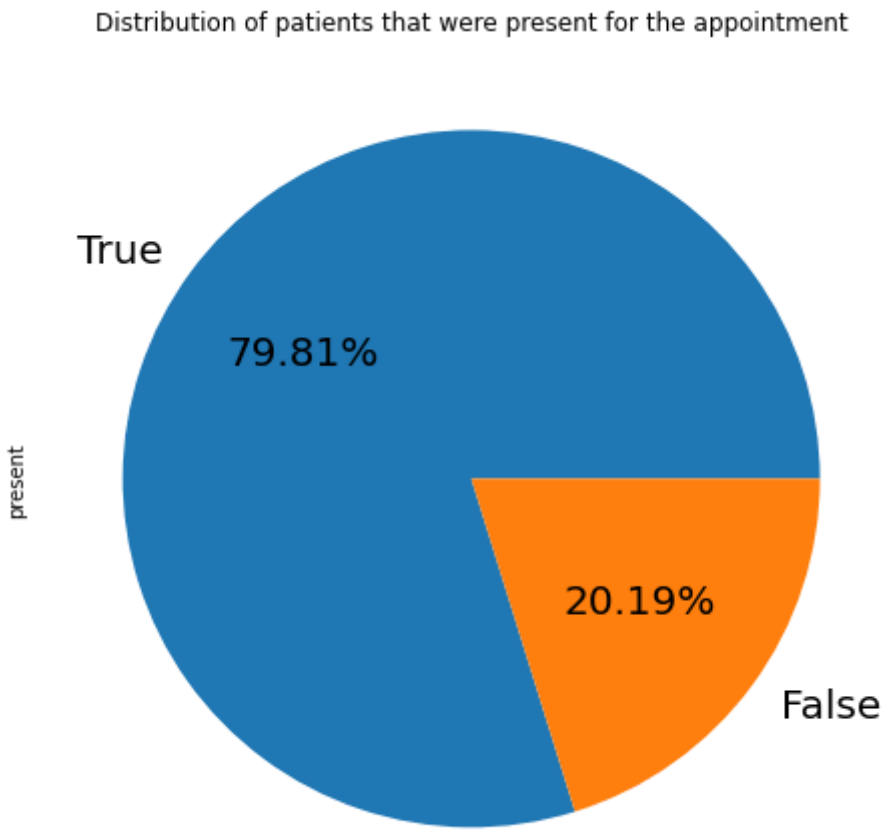
> Now, Let's dive into the question-answering process.

1. What is the ratio of people present or absent for the appointment?

In [29]:
```
# Let's see the visual distributionof those who came for their appointment.

colors = "blue"
appointment_df["present"].value_counts().plot(
    kind="bar",
    figsize=(8, 8),
    color=colors,
    xlabel="Present",
    ylabel="Patient count",
    title="Distribution of patients that were present for the appointment",
    alpha=0.7,
);
```



In [30]:
```
# let also create a pie chart to show the distribution of patients that sho

ind = appointment_df["present"].value_counts().index
appointment_df["present"].value_counts().plot(
    kind="pie",
    figsize=(8, 8),
    autopct="%1.2f%%",
    title="Distribution of patients that were present for the appointment",
    fontsize=20,
);
```



In [31]:
```
# Check appointment attendance (True = Present, False = Absent)
appointment_df["present"].value_counts()
```

Out[31]:
```
True     88208
False    22319
Name: present, dtype: int64
```

> In total, 110527 patients scheduled appointments, of which 88208 showed up, corresponding to approximately 79.8%, and 22319 did not, corresponding to approximately 20.2%.

2. What gender booked more medical appointments?

In [32]:
```
# Group data by gender
appointment_df.groupby("gender").count()
```

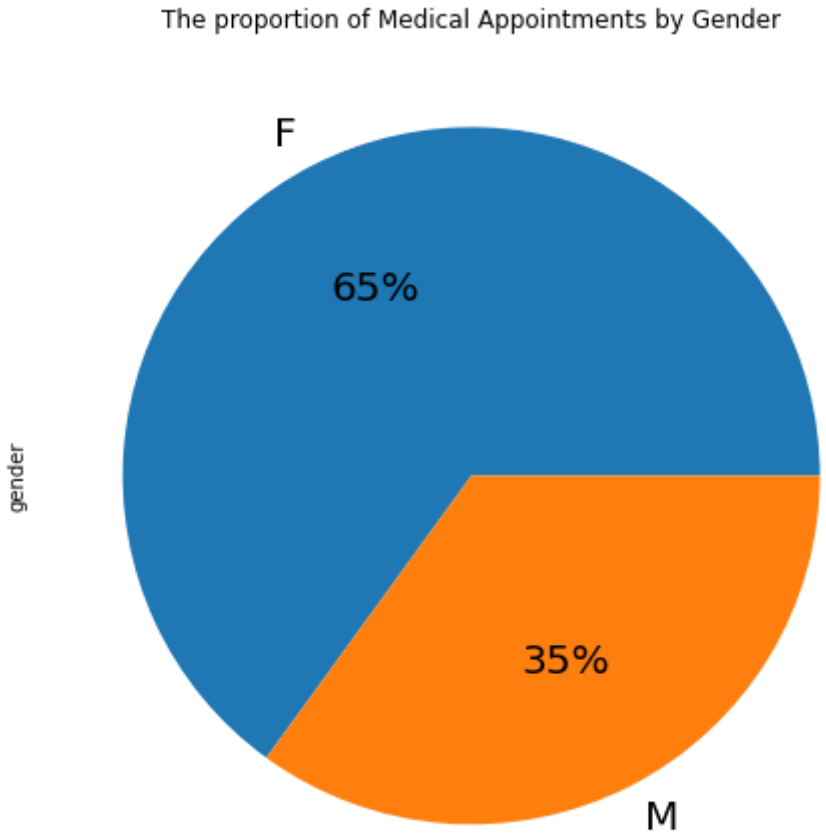
Out[32]:

gender	scheduled_day	appointment_day	age	neighbourhood	scholarship	hypertension	di
F	71840	71840	71840	71840	71840	71840	
M	38687	38687	38687	38687	38687	38687	

In [33]:
```
# The number of males and females that booked an appointment.
appointment_df["gender"].value_counts()
```

Out[33]:
```
F    71840
M    38687
Name: gender, dtype: int64
```

In [34]:
```
# Visualise the proportion of Medical Appointments by Gender with a pie cha
# Note F represents Female while M represents Male.

appointment_df["gender"].value_counts().plot(
    kind="pie",
    title="The proportion of Medical Appointments by Gender",
    figsize=(8, 8),
    autopct="%1.0f%%",
    fontsize=20,
);
```



> According to the above visualisation, it is obvious that more female patients schedule medical appointments than their male counterparts.

3. Among those scheduled for appointments, what is the most common disease?

These diseases include hypertension, diabetes, alcoholism, and handicap.

In [35]:
```
# Sum all values in column "hypertension", "diabetes", "alcoholism" and "ha

columns = ["hypertension", "diabetes", "alcoholism", "handicap"]
columns_total = appointment_df[columns].sum(axis=1).sum()
columns_total
```

Out[35]:
```
35563
```

In [36]:
```
# Define a function to calculate the percentage of specific disease
```

```
def column_percent(i):
    x = i.sum() / columns_total * 100
    return x
```

In [37]:
```
# Proportion of hypertension patient

hypertension_proportion = column_percent(appointment_df.hypertension)
hypertension_proportion
```

Out[37]:
```
61.3024772938166
```

In [38]:
```
# Proportion of diabetes patient

diabetes_proportion = column_percent(appointment_df.diabetes)
diabetes_proportion
```

Out[38]:
```
22.335011107049464
```

In [39]:
```
# Proportion of alcoholism patient

alcoholism_proportion = column_percent(appointment_df.alcoholism)
alcoholism_proportion
```

Out[39]:
```
9.448021820431347
```

In [40]:
```
# Proportion of handicap patient

handicap_proportion = column_percent(appointment_df.handicap)
handicap_proportion
```

Out[40]:
```
6.914489778702585
```

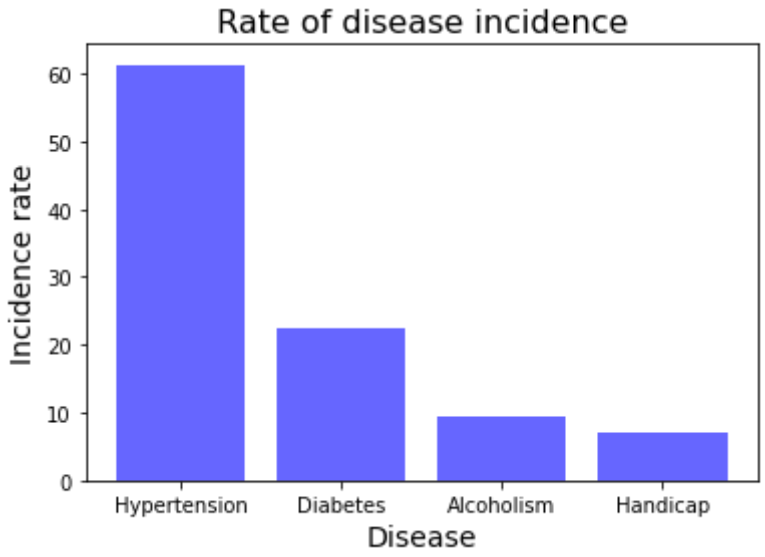
In [41]:
```
# Create a data frame of Diseases and thier respective proportion

diseases_df = pd.DataFrame(
    data={
        "Disease": ["Hypertension", "Diabetes", "Alcoholism", "Handicap"],
        "Proportion": [
            hypertension_proportion,
            diabetes_proportion,
            alcoholism_proportion,
            handicap_proportion,
        ],
    }
)

diseases_df
```

Out[41]:

	Disease	Proportion
0	Hypertension	61.302477
1	Diabetes	22.335011
2	Alcoholism	9.448022
3	Handicap	6.914490

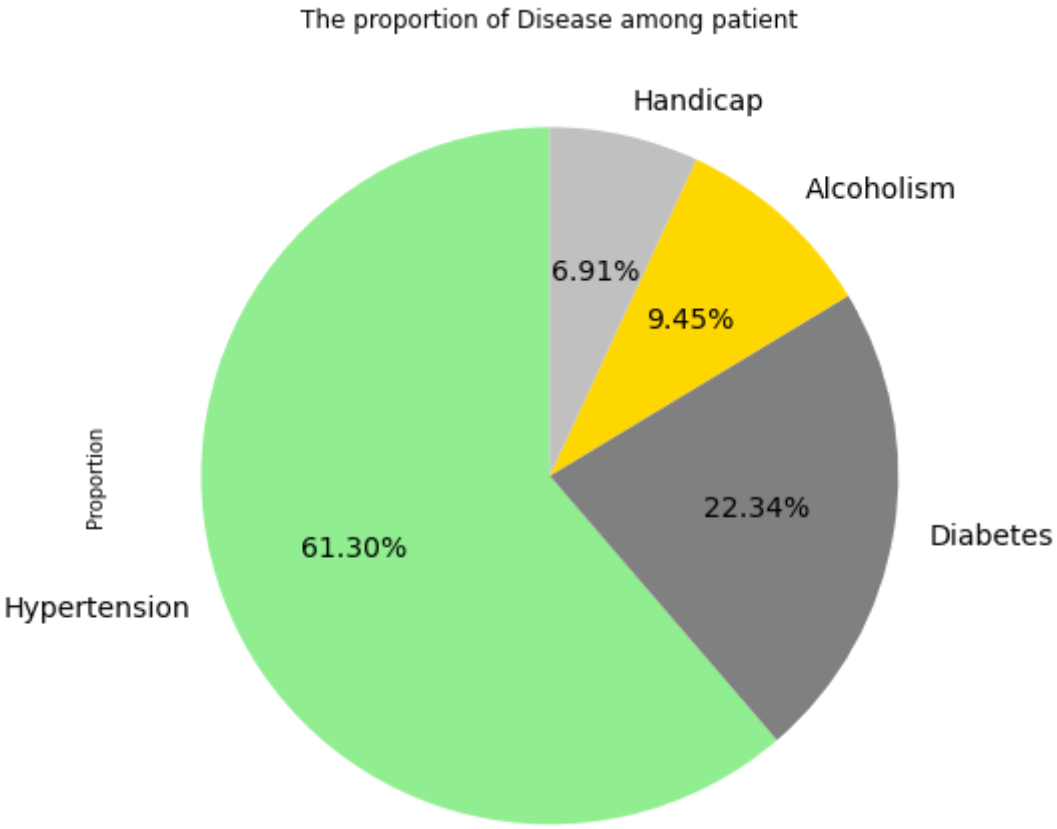
In [42]:
```
# Bar Chat for the rate of disease incidence

locations = [1, 2, 3, 4]
heights = [
```

```
    hypertension_proportion,
    diabetes_proportion,
    alcoholism_proportion,
    handicap_proportion,
]
labels = ["Hypertension", "Diabetes", "Alcoholism", "Handicap"]

plt.bar(locations, heights, color="blue", alpha=0.6)
plt.title("Rate of disease incidence", fontsize=16)
plt.xlabel("Disease", fontsize=14)
plt.ylabel("Incidence rate", fontsize=14)
plt.xticks(locations, labels);
```



In [43]:

```
# Plot a pie chat that shows the proportion of each Disease among patient

my_label = ["Hypertension", "Diabetes", "Alcoholism", "Handicap"]
my_color = ["lightgreen", "gray", "gold", "silver"]
diseases_df.plot.pie(
    y="Proportion",
    title="The proportion of Disease among patient",
    figsize=(8, 8),
    fontsize=14,
    labels=my_label,
    colors=my_color,
    legend=False,
    startangle=90,
    autopct="%1.2f%%",
);
```

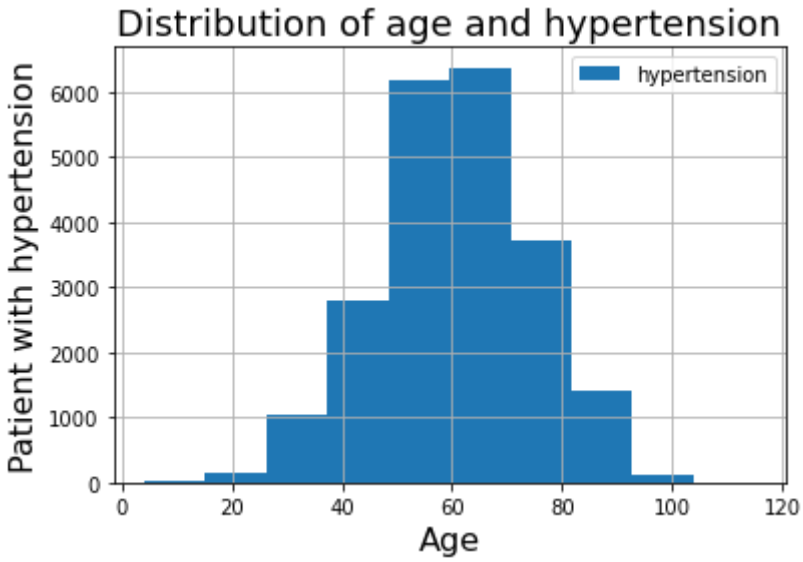


> Based on the investigation and visualization above. It is obvious that hypertension is the most common disease among the people that schedule appointments with an approximate ratio of 61.30%.

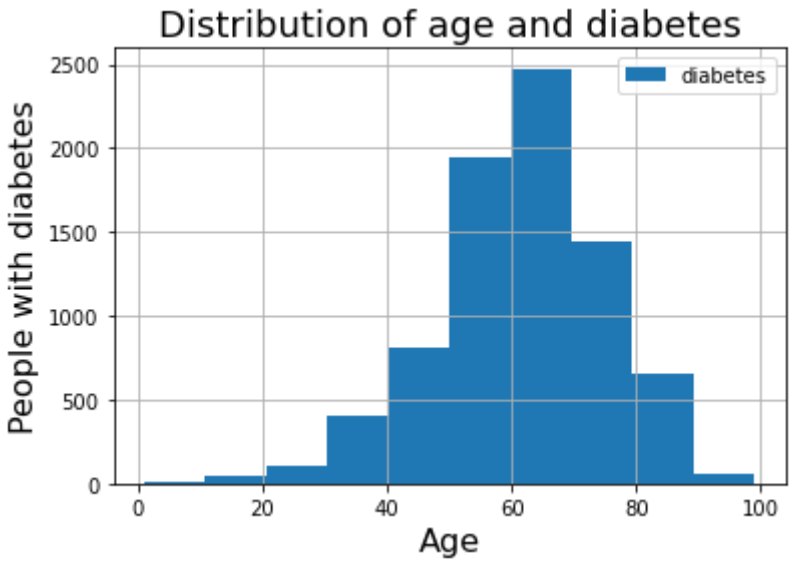
4. Which age is most affected by disease?

These diseases include hypertension, diabetes, alcoholism, and handicap.

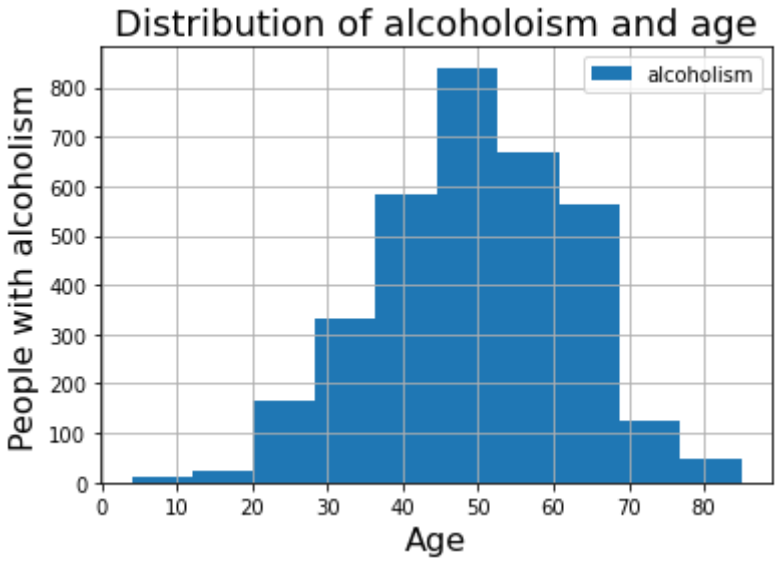
In [44]:
```
# Create an histogram chart for hypertension and age

appointment_df.age[appointment_df.hypertension == 1].hist(bins=10, label="h
plt.title("Distribution of age and hypertension", fontsize=18)
plt.xlabel("Age", fontsize=16)
plt.ylabel("Patient with hypertension", fontsize=16)
plt.legend();
```



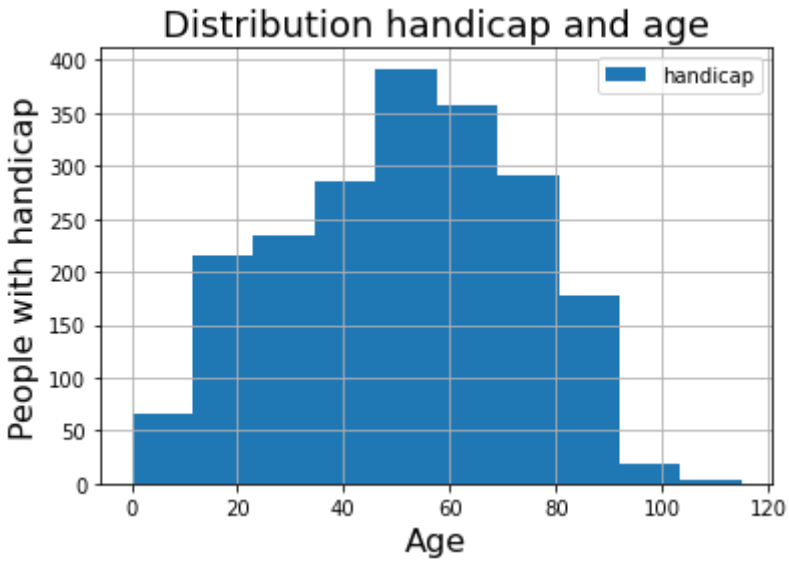
In [45]:
```
# Create an histogram the diabetes and age

appointment_df.age[appointment_df.diabetes == 1].hist(bins=10, label="diabe
plt.title("Distribution of age and diabetes", fontsize=18)
plt.xlabel("Age", fontsize=16)
plt.ylabel("People with diabetes", fontsize=16)
plt.legend();
```



In [46]:
```
# Create an histogram for alcoholism and age

appointment_df.age[appointment_df.alcoholism == 1].hist(bins=10, label="alc
plt.title("Distribution of alcoholoism and age", fontsize=18)
plt.xlabel("Age", fontsize=16)
plt.ylabel("People with alcoholism", fontsize=16)
plt.legend();
```



In [47]:
```
# Create an histogram for handicap and age

appointment_df.age[appointment_df.handicap == 1].hist(bins=10, label="handi
plt.title("Distribution handicap and age", fontsize=18)
plt.xlabel("Age", fontsize=16)
plt.ylabel("People with handicap", fontsize=16)
plt.legend();
```



> The following are observed from the above visualization:
>
> 1. There is a high rate of hypertension between the age of 50 to 70.
> 2. There is a high rate of diabetes between the age of 60 to 70.
> 3. There is a high rate of alcoholism between the age of 40 to 60.
> 4. There is a high rate of handicaps between the age of 40 to 80.

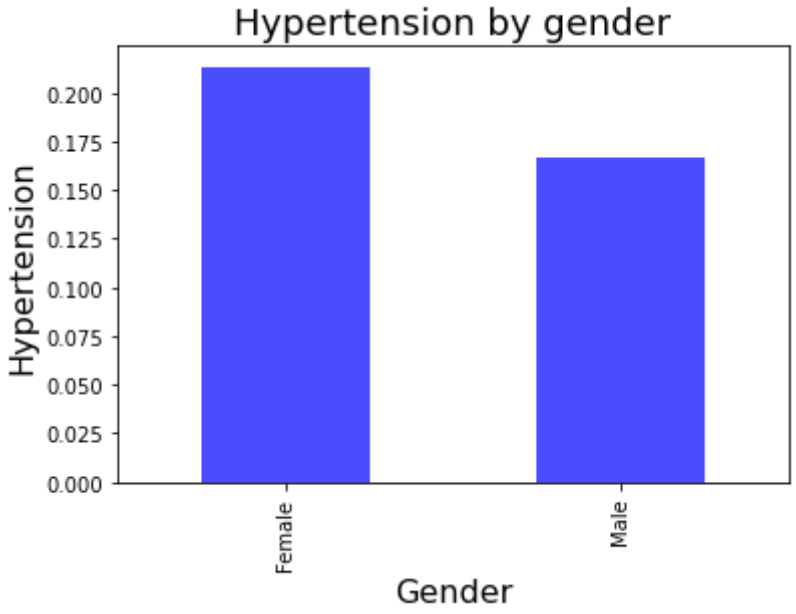
5. What gender is most affected by the diseases?

These diseases include hypertension, diabetes, alcoholism, and handicap.

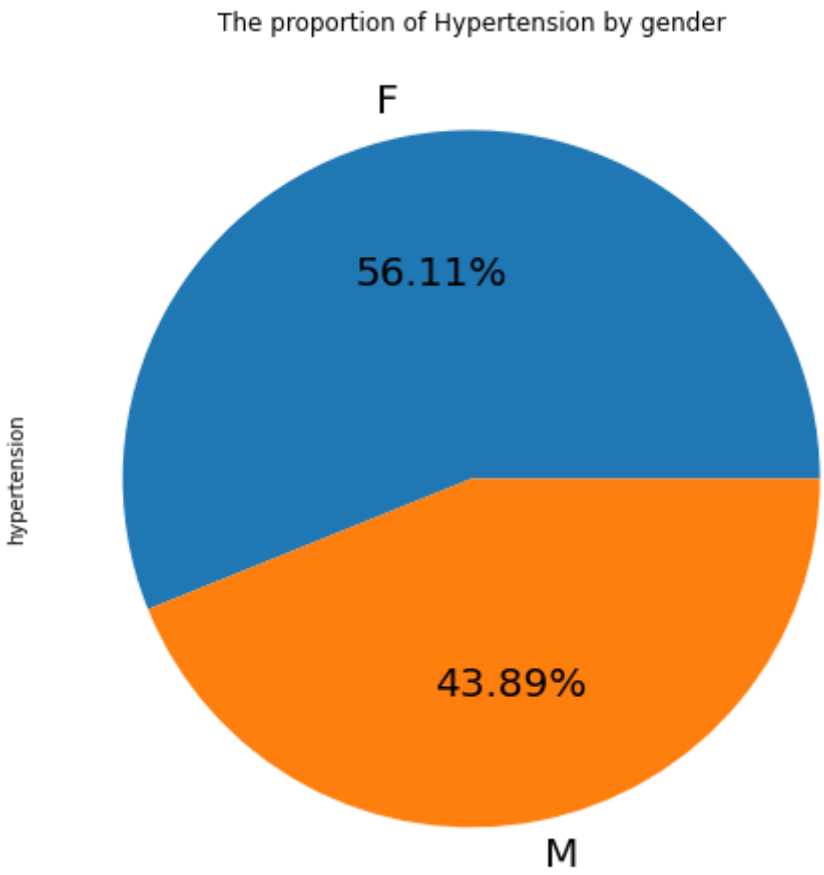
In [48]:
```
# The mean value of hypertension grouped by gender
appointment_df.groupby("gender")["hypertension"].mean()
```

Out[48]:
```
gender
F    0.213516
M    0.167033
Name: hypertension, dtype: float64
```

In [49]:
```
# show  bar chart with hypertension and gender

appointment_df.groupby("gender")["hypertension"].mean().plot(
    kind="bar", alpha=0.7, color="blue"
)

# title and labels
plt.xticks([0, 1], ["Female", "Male"])
plt.title("Hypertension by gender", fontsize=18)
plt.xlabel("Gender", fontsize=16)
plt.ylabel("Hypertension", fontsize=16);
```



In [50]:
```
# Visualise the proportion of Medical Appointments by Gender with a pie cha

appointment_df.groupby("gender")["hypertension"].mean().plot(
    kind="pie",
    title="The proportion of Hypertension by gender",
    figsize=(8, 8),
    fontsize=20,
    autopct="%1.2f%%",
);
```



> It is shown from the above investigation that approximately 56% of females and 44% of males have hypertension. Therefore, this implies that there is more female hypertension patient than men.

Conclusions

The data of medical appointments of patients that attended or did not attend were analyzed to answer several questions.

Our data exploration resulted in an understanding of all the data's features, and we then proceeded to wrangle and clean up the data according to our proposed use.

1. According to the first research question, out of 110527 patients who scheduled appointments, 88208 were present, which amounts to approximately 79.8%, and 22319 were absent, which amounts to 20.2%. As a result, the majority of those who booked appointments were present on the appointment day.
2. As a second step, we investigated the gender with the most medical appointments. Our visualization indicates that more female patients schedule medical appointments, amounting to approximately 65%, than their male counterparts, amounting to approximately 35%.
3. Furthermore, we investigated the most common disease among those scheduled for appointments. Based on the analysis and visualization of our dataset. With an approximate ratio of 61.30%, hypertension constitutes the most common disease among those who set up appointments.

4. Moreover, we examined the age group most affected by the disease. Using our analysis and visualization, we discovered hypertension in patients aged 50 to 70, diabetes in patients aged 60 to 70, alcoholism in patients 40 to 60, and handicaps in patients 40 to 80 are all prevalent. Generally, between the ages of 50 and 60, people are more likely to suffer from these diseases.
5. Lastly, we examined the gender most affected by the diseases. Our examination and visualization of the dataset show that approximately 56% of females and 44% of males suffer from hypertension. Thus, it implies a higher proportion of female patients with hypertension than male patients.

Limitations

Although the data set allowed us to answer five questions, we still encountered challenges and limitations.

A primary limitation of our analysis was insufficient numerical data amongst the column features to enable us to run more numerical analyses. Initially, this was a problem we encountered. However, we were able to wrangle the data set in such a way as to answer our research questions.

Additionally, the dataset contained many errors, making the data cleaning process take much more time than exploratory data analysis.

In addition, certain assumptions had to be formulated to proceed with our analysis since I observed that many patients were aged zero. It is not apparent from the data why a patient would have an age of 0. In this study, zero-age patients constituted a large portion of the total number of patients - 3539. I assumed they were babies under six months old to proceed with our analysis.

In this study, all statistics are descriptive, not inferential, which means that we did not use our data to create hypotheses or control experiences.