In [1]:
```
import pandas as pd
import numpy as np
import datetime
from time import strftime
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

In [2]:
```
# Reading the dataset

base_data = pd.read_csv('Data.csv')
```

In [3]:
```
base_data
```

Out[3]:

AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipe
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	
...	...	...	...	...	...	...	
5651768	F	2016-05-03T09:15:35Z	2016-06-07T00:00:00Z	56	MARIA ORTIZ	0	
5650093	F	2016-05-03T07:27:33Z	2016-06-07T00:00:00Z	51	MARIA ORTIZ	0	
5630692	F	2016-04-27T16:03:52Z	2016-06-07T00:00:00Z	21	MARIA ORTIZ	0	
5630323	F	2016-04-27T15:09:23Z	2016-06-07T00:00:00Z	38	MARIA ORTIZ	0	
5629448	F	2016-04-27T13:30:56Z	2016-06-07T00:00:00Z	54	MARIA ORTIZ	0	

ıns

In [4]:
```
base_data.shape
```

Out[4]:
```
(110527, 14)
```

In [5]:
```
base_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64  
 2   Gender          110527 non-null  object 
 3   ScheduledDay    110527 non-null  object 
 4   AppointmentDay  110527 non-null  object 
 5   Age             110527 non-null  int64  
 6   Neighbourhood   110527 non-null  object 
 7   Scholarship     110527 non-null  int64  
 8   Hipertension    110527 non-null  int64  
 9   Diabetes        110527 non-null  int64  
 10  Alcoholism      110527 non-null  int64  
 11  Handcap         110527 non-null  int64  
 12  SMS_received    110527 non-null  int64  
 13  No-show         110527 non-null  object 
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

In [6]:
```
#modifying the date and time into standard form
base_data['ScheduledDay'] = pd.to_datetime(base_data['ScheduledDay']).
base_data['AppointmentDay'] = pd.to_datetime(base_data['AppointmentDay
```

In [7]:
```
base_data.head(5)
```

Out[7]:

AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipe
5642903	F	2016-04-29	2016-04-29	62	JARDIM DA PENHA	0	
5642503	M	2016-04-29	2016-04-29	56	JARDIM DA PENHA	0	
5642549	F	2016-04-29	2016-04-29	62	MATA DA PRAIA	0	
5642828	F	2016-04-29	2016-04-29	8	PONTAL DE CAMBURI	0	
5642494	F	2016-04-29	2016-04-29	56	JARDIM DA PENHA	0	

for the schedule day and appointment day storing the weekdays only into a variable
I added two new columns to my dataset, sch_weekday and app_weekday, which store the day of the week for ScheduledDay and AppointmentDay respectively. In pandas, dt.dayofweek returns the day of the week as an integer, where Monday is 0 and Sunday is 6.

In [8]:
```
# 5 is Saturday, 6 is Sunday

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

In [9]:
```
base_data['app_weekday'] = base_data['AppointmentDay'].dt.dayofweek
```

In [10]:
```
base_data['sch_weekday'].value_counts()
```

Out[10]:
```
1    26168
2    24262
0    23085
4    18915
3    18073
5       24
Name: sch_weekday, dtype: int64
```

I counted how many appointments were scheduled on each day of the week and printed the results

In [11]:
```
base_data['app_weekday'].value_counts()
```

Out[11]:
```
2    25867
1    25640
0    22715
4    19019
3    17247
5       39
Name: app_weekday, dtype: int64
```

Similarly app_weekdays

In [12]:
```
base_data.columns
```

Out[12]:
```
Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hipe
rtension',
       'Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'No-sho
w',
       'sch_weekday', 'app_weekday'],
      dtype='object')
```

In [13]:
```
#changing the name of some cloumns
base_data= base_data.rename(columns={'Hipertension': 'Hypertension', '
```

In [14]:
```
base_data.columns
```

Out[14]:
```
Index(['PatientId', 'AppointmentID', 'Gender', 'ScheduledDay',
       'AppointmentDay', 'Age', 'Neighbourhood', 'Scholarship', 'Hype
rtension',
       'Diabetes', 'Alcoholism', 'Handicap', 'SMSReceived', 'NoShow',
       'sch_weekday', 'app_weekday'],
      dtype='object')
```

In [57]:
```
base_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 16 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  datetime64[ns]
 4   AppointmentDay  110527 non-null  datetime64[ns]
 5   Age             110527 non-null  int64
 6   Neighbourhood   110527 non-null  object
 7   Scholarship     110527 non-null  int64
 8   Hypertension    110527 non-null  int64
 9   Diabetes        110527 non-null  int64
 10  Alcoholism      110527 non-null  int64
 11  Handicap        110527 non-null  int64
 12  SMSReceived     110527 non-null  int64
 13  NoShow          110527 non-null  object
 14  sch_weekday     110527 non-null  int64
 15  app_weekday     110527 non-null  int64
dtypes: datetime64[ns](2), float64(1), int64(10), object(3)
memory usage: 13.5+ MB
```

In [15]:
```
# dropping some columns which have no significance
base_data.drop(['PatientId', 'AppointmentID', 'Neighbourhood'], axis=1
```

In [16]:
```
base_data
```

Out[16]:

ender	ScheduledDay	AppointmentDay	Age	Scholarship	Hypertension	Diabetes	Alcoholism	I
F	2016-04-29	2016-04-29	62	0	1	0	0	
M	2016-04-29	2016-04-29	56	0	0	0	0	
F	2016-04-29	2016-04-29	62	0	0	0	0	
F	2016-04-29	2016-04-29	8	0	0	0	0	
F	2016-04-29	2016-04-29	56	0	1	1	0	
...	...	...	...	...	...	...	...	
F	2016-05-03	2016-06-07	56	0	0	0	0	
F	2016-05-03	2016-06-07	51	0	0	0	0	
F	2016-04-27	2016-06-07	21	0	0	0	0	
F	2016-04-27	2016-06-07	38	0	0	0	0	
F	2016-04-27	2016-06-07	54	0	0	0	0	

ws × 13 columns

In [17]:
```
base_data.info()
```

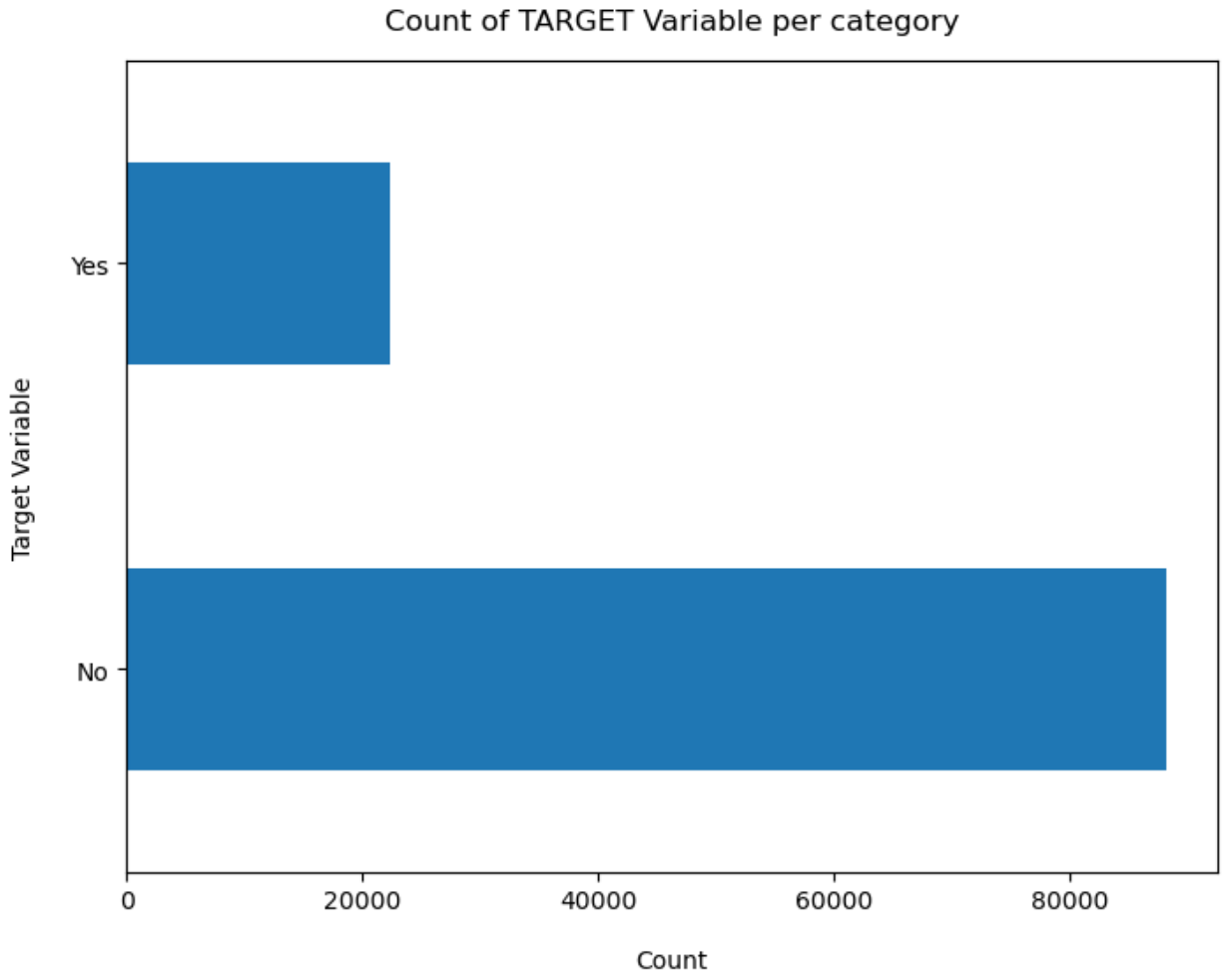
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Gender          110527 non-null  object
 1   ScheduledDay    110527 non-null  datetime64[ns]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

In [18]:
```
base_data.describe()
```

Out[18]:

	Age	Scholarship	Hypertension	Diabetes	Alcoholism	Handi
count	110527.000000	110527.000000	110527.000000	110527.000000	110527.000000	110527.000
mean	37.088874	0.098266	0.197246	0.071865	0.030400	0.022
std	23.110205	0.297675	0.397921	0.258265	0.171686	0.161
min	-1.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	18.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	37.000000	0.000000	0.000000	0.000000	0.000000	0.000
75%	55.000000	0.000000	0.000000	0.000000	0.000000	0.000
max	115.000000	1.000000	1.000000	1.000000	1.000000	4.000

In [19]:
```
base_data['NoShow'].value_counts().plot(kind='barh', figsize=(8, 6))
plt.xlabel("Count", labelpad=14)
plt.ylabel("Target Variable", labelpad=14)
plt.title("Count of TARGET Variable per category", y=1.02);
```



The plot shows two bars: one for 'Yes' (patients who did not show up) and one for 'No' (patients who did show up). The length of each bar represents the count of each category. In this plot, it is evident that a larger number of patients did show up ('No') compared to those who did not show up ('Yes').

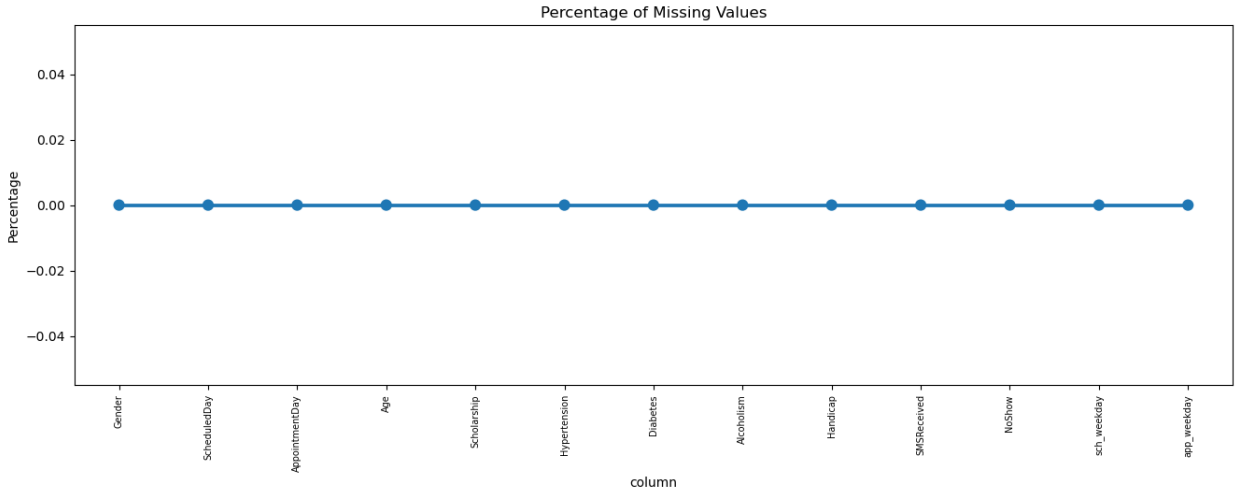
In [20]:
```
# calculating the % of appointments or not
100*base_data['NoShow'].value_counts()/len(base_data['NoShow'])
```

Out[20]:
```
No     79.806744
Yes    20.193256
Name: NoShow, dtype: float64
```

In [21]:
```
base_data['NoShow'].value_counts()
```

Out[21]:
```
No     88208
Yes    22319
Name: NoShow, dtype: int64
```

In [23]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate the percentage of missing values for each column
missing = pd.DataFrame((base_data.isnull().sum() * 100) / base_data.sh
missing.columns = ['column', 'percentage']

# Plot the percentage of missing values
plt.figure(figsize=(16, 5))
ax = sns.pointplot(x='column', y='percentage', data=missing)
plt.xticks(rotation=90, fontsize=7)
plt.title("Percentage of Missing Values")
plt.ylabel("Percentage")
plt.show()
```



The plot indicates that there are no missing values in the dataset, as all columns have a 0% missing value rate.

Missing Data - Initial Intuition

- Here, we don't have any missing data.

General Thumb Rules:

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

Data Cleaning

1. Create a copy of base data for manupulation & processing

In [24]:
```
new_data = base_data.copy()
```

In [25]:
```
new_data.info()
```

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

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

In [26]:
```
# Get the max tenure
print(base_data['Age'].max()) #72
```

```
115
```

In [27]:
```
# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 20) for i in range(1, 118, 20)]

base_data['Age_group'] = pd.cut(base_data.Age, range(1, 130, 20), righ
```

In [28]:
```
base_data.drop(['Age'], axis=1, inplace=True)
```

####Created a Copy of the Data:

Ensured the original dataset remains unchanged by creating a copy for manipulation and processing. Checked the Maximum Age:

Verified that the maximum age in the dataset is 115. Binned the Age Column:

Grouped the Age column into bins of 20 years each and created a new column Age_group. Dropped the Original Age Column:

Removed the original Age column after creating the age groups.

The dataset now includes an Age_group column for better categorical analysis and visualization.

Data Exploration

In [38]:
```
list(base_data.columns)
```

Out[38]:
```
['Gender',
 'ScheduledDay',
 'AppointmentDay',
 'Scholarship',
 'Hypertension',
 'Diabetes',
 'Alcoholism',
 'Handicap',
 'SMSReceived',
 'NoShow',
 'sch_weekday',
 'app_weekday',
 'Age_group']
```

In [39]:
```
#having a loook into the values of count of each columns and there cou
for i, predictor in enumerate(base_data.drop(columns=['NoShow'])):
    print('-'*10,predictor,'-'*10)
    print(base_data[predictor].value_counts())
    plt.figure(i)
    sns.countplot(data=base_data, x=predictor, hue='NoShow')
```

In [40]:
```
base_data['NoShow'] = np.where(base_data.NoShow == 'Yes',1,0)
```

In [41]:
```
base_data.NoShow.value_counts()
```

Out[41]:
```
0    110527
Name: NoShow, dtype: int64
```

In [ ]:
```

```

In [44]:
```
import numpy as np

# Step 1: Print first few rows of the dataset
print(base_data.head())

# Step 2: Check the unique values in the NoShow column
print(base_data['NoShow'].unique())

# Step 3: Convert 'Yes' to 1 and 'No' to 0 using np.where
base_data['NoShow'] = np.where(base_data['NoShow'] == 'Yes', 1, 0)

# Step 4: Print first few rows of the dataset to verify conversion
print(base_data.head())

# Check the value counts after conversion
print(base_data['NoShow'].value_counts())
```

```
  Gender ScheduledDay AppointmentDay  Scholarship  Hypertension  Diab
etes  \
0      F   2016-04-29     2016-04-29            0             1
0
1      M   2016-04-29     2016-04-29            0             0
0
2      F   2016-04-29     2016-04-29            0             0
0
3      F   2016-04-29     2016-04-29            0             0
0
4      F   2016-04-29     2016-04-29            0             1
1

   Alcoholism  Handicap  SMSReceived  NoShow  sch_weekday  app_weekda
y  \
0           0         0            0       0            4
4
1           0         0            0       0            4
4
2           0         0            0       0            4
4
3           0         0            0       0            4
4
4           0         0            0       0            4
```

```
4

  Age_group
0   61 - 81
1   41 - 61
2   61 - 81
3    1 - 21
4   41 - 61
[0]
  Gender ScheduledDay AppointmentDay  Scholarship  Hypertension  Diab
etes  \
0      F   2016-04-29     2016-04-29            0             1
0
1      M   2016-04-29     2016-04-29            0             0
0
2      F   2016-04-29     2016-04-29            0             0
0
3      F   2016-04-29     2016-04-29            0             0
0
4      F   2016-04-29     2016-04-29            0             1
1

   Alcoholism  Handicap  SMSReceived  NoShow  sch_weekday  app_weekda
y  \
0           0         0            0       0            4
4
1           0         0            0       0            4
4
2           0         0            0       0            4
4
3           0         0            0       0            4
4
4           0         0            0       0            4
4

  Age_group
0   61 - 81
1   41 - 61
2   61 - 81
3    1 - 21
4   41 - 61
0    110527
Name: NoShow, dtype: int64
```

In [45]:
```
import numpy as np

# Step 1: Check the unique values in the NoShow column
print(base_data['NoShow'].unique())

# If necessary, manually reset a few values for testing purposes
base_data.loc[base_data.index[:5], 'NoShow'] = ['Yes', 'No', 'Yes', 'N

# Step 2: Convert 'Yes' to 1 and 'No' to 0 using np.where
base_data['NoShow'] = np.where(base_data['NoShow'] == 'Yes', 1, 0)

# Step 3: Verify the conversion by checking the value counts
print(base_data['NoShow'].value_counts())
```

```
[0]
0    110524
1         3
Name: NoShow, dtype: int64
```

Convert all the categorical variables into dummy variables

In [47]:
```
base_data_dummies = pd.get_dummies(base_data)
base_data_dummies.head()
```

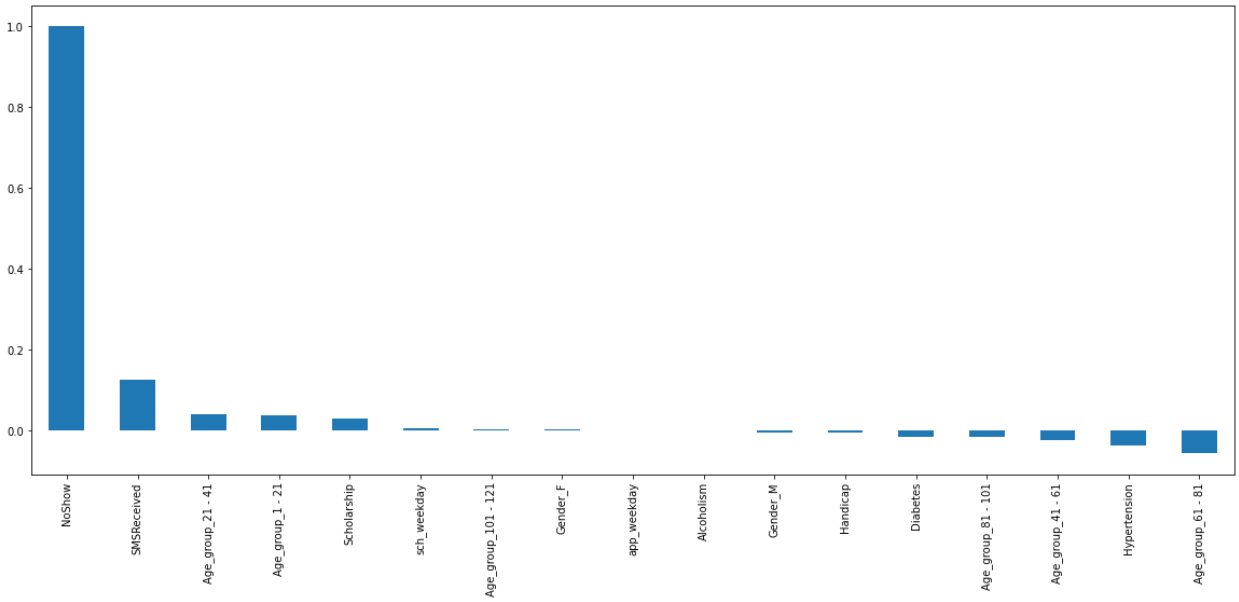
Out[47]:

ceived	NoShow	sch_weekday	app_weekday	Gender_F	Gender_M	Age_group_1 - 21	Age_group_21 - 41
0	1	4	4	1	0	0	0
0	0	4	4	0	1	0	0
0	1	4	4	1	0	0	0
0	0	4	4	1	0	1	0
0	1	4	4	1	0	0	0

Build a corelation of all predictors with 'NoShow'

In [34]:
```
plt.figure(figsize=(20,8))
base_data_dummies.corr()['NoShow'].sort_values(ascending = False).plot
```

Out[34]: <AxesSubplot:>

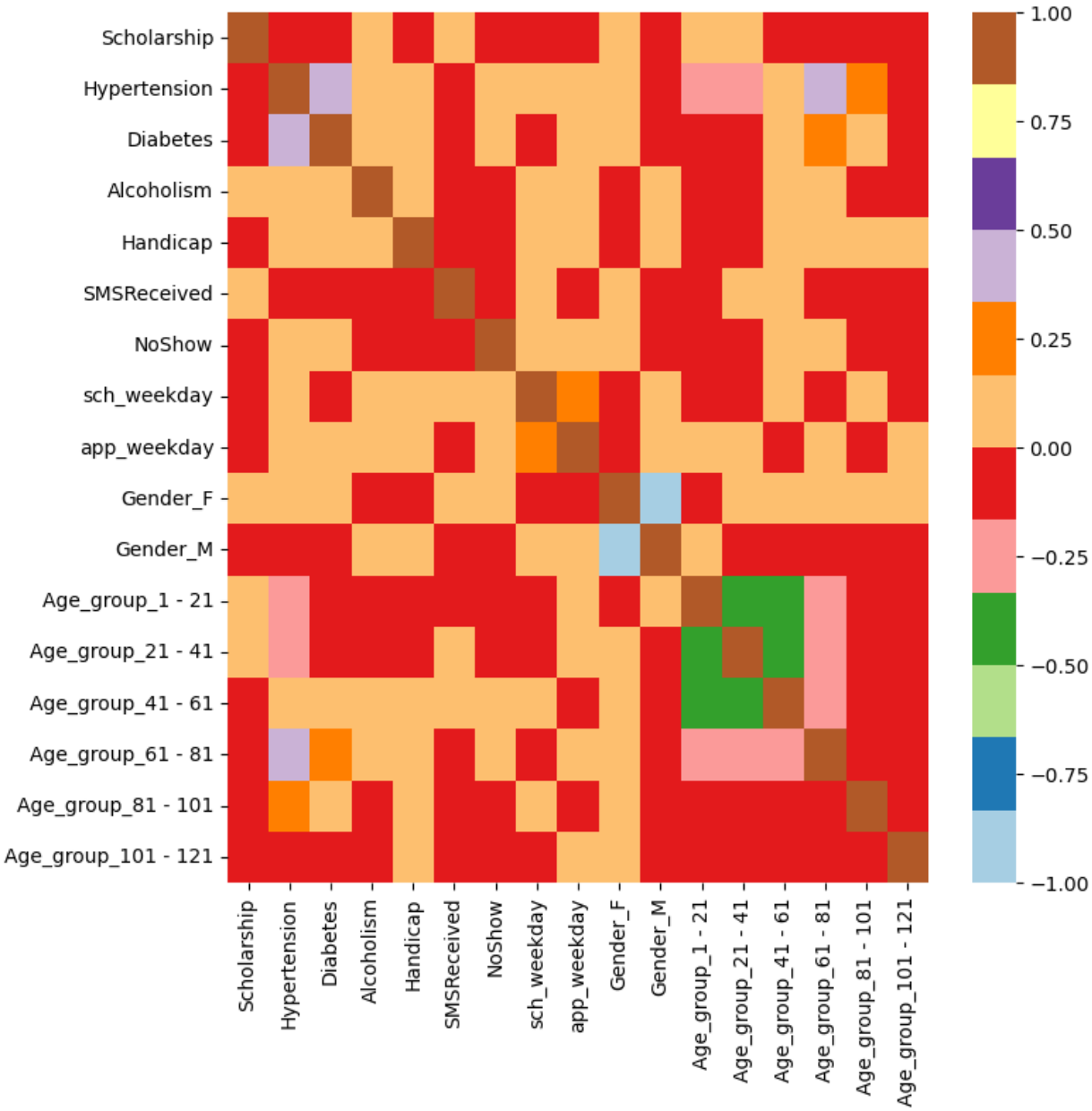


he correlation analysis indicates that certain features such as SMSReceived and Scholarship might have a stronger relationship with the likelihood of missing an appointmen

In [50]:
```
plt.figure(figsize=(8,8))
sns.heatmap(base_data_dummies.corr(), cmap="Paired")
```

```
/var/folders/xh/mq11ygyx017bms8cxp75gy_m0000gn/T/ipykernel_1612/3897525021.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(base_data_dummies.corr(), cmap="Paired")
```

Out[50]: <Axes: >



High Positive Correlations:

Features that show a strong positive correlation with NoShow can indicate factors that might increase the likelihood of missing appointments. High Negative Correlations:

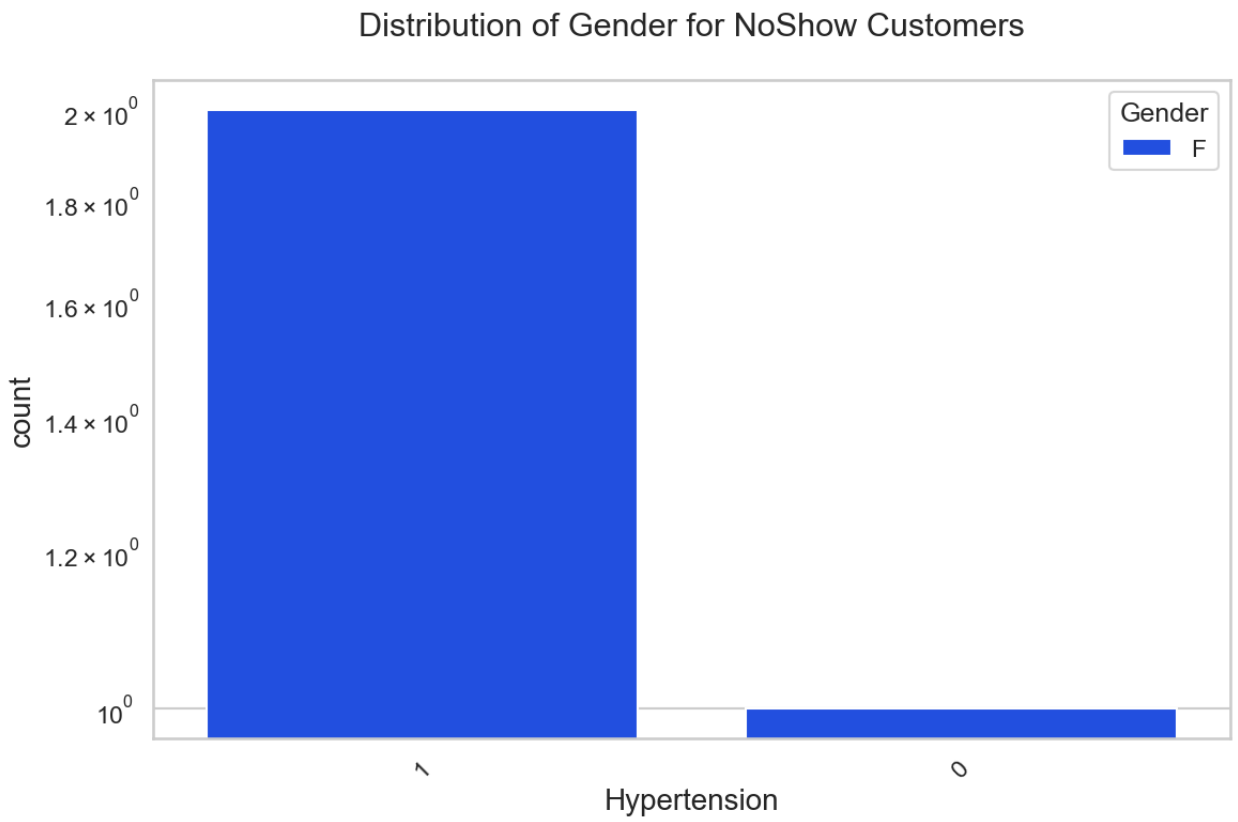
Features with strong negative correlations with NoShow can indicate factors that might decrease the likelihood of missing appointments.

Bivariate Analysis

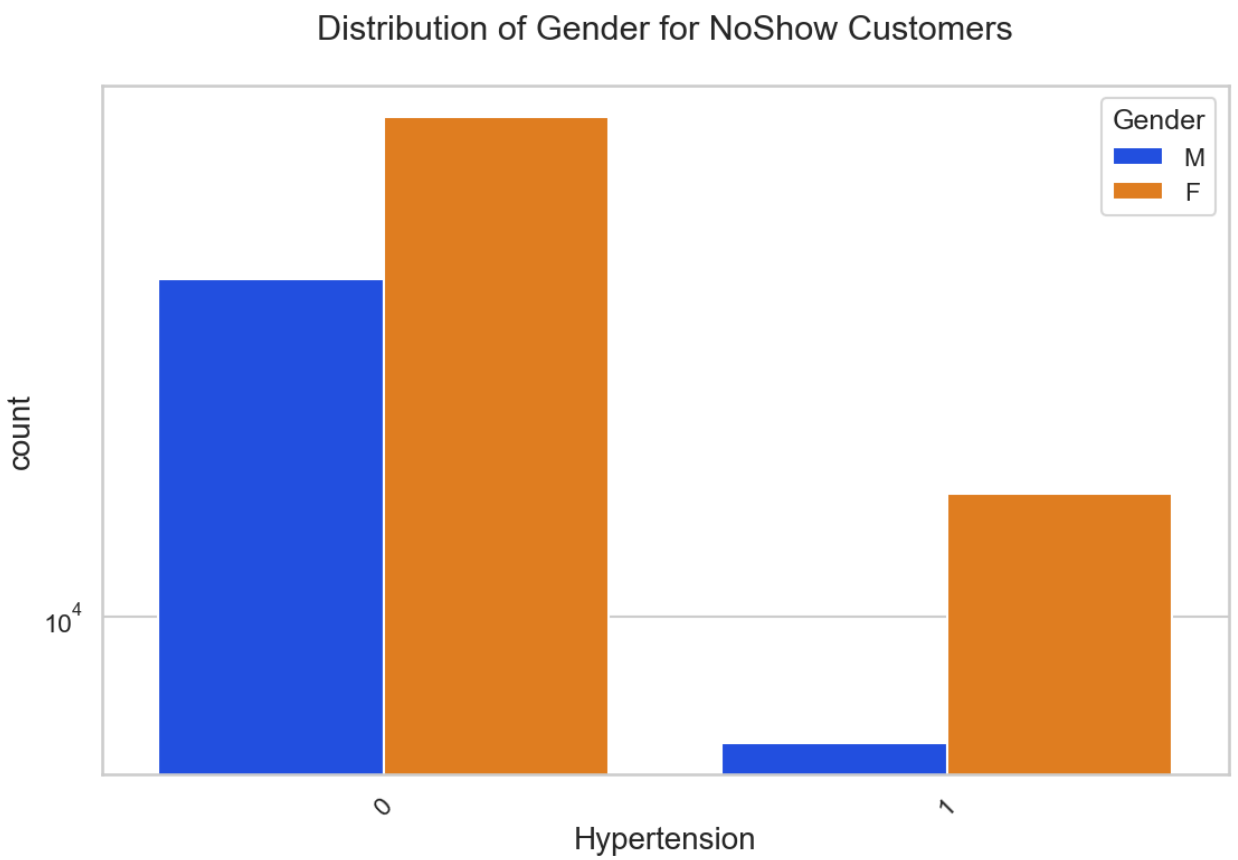
In [51]:
```
new_df1_target0=base_data.loc[base_data["NoShow"]==0]
new_df1_target1=base_data.loc[base_data["NoShow"]==1]
```

In [56]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Filter DataFrames based on NoShow values
new_df1_target0 = base_data.loc[base_data["NoShow"] == 0]
new_df1_target1 = base_data.loc[base_data["NoShow"] == 1]

# Define the uniplot function
def uniplot(df, col, title, hue=None):
    sns.set_style('whitegrid')
    sns.set_context('talk')
    plt.rcParams["axes.labelsize"] = 20
    plt.rcParams['axes.titlesize'] = 22
    plt.rcParams['axes.titlepad'] = 30

    temp = pd.Series(data=hue)
    fig, ax = plt.subplots()
    width = len(df[col].unique()) + 7 + 4 * len(temp.unique())
    fig.set_size_inches(width, 8)
    plt.xticks(rotation=45)
    plt.yscale('log')
    plt.title(title)
    ax = sns.countplot(data=df, x=col, order=df[col].value_counts().in
    plt.show()
```

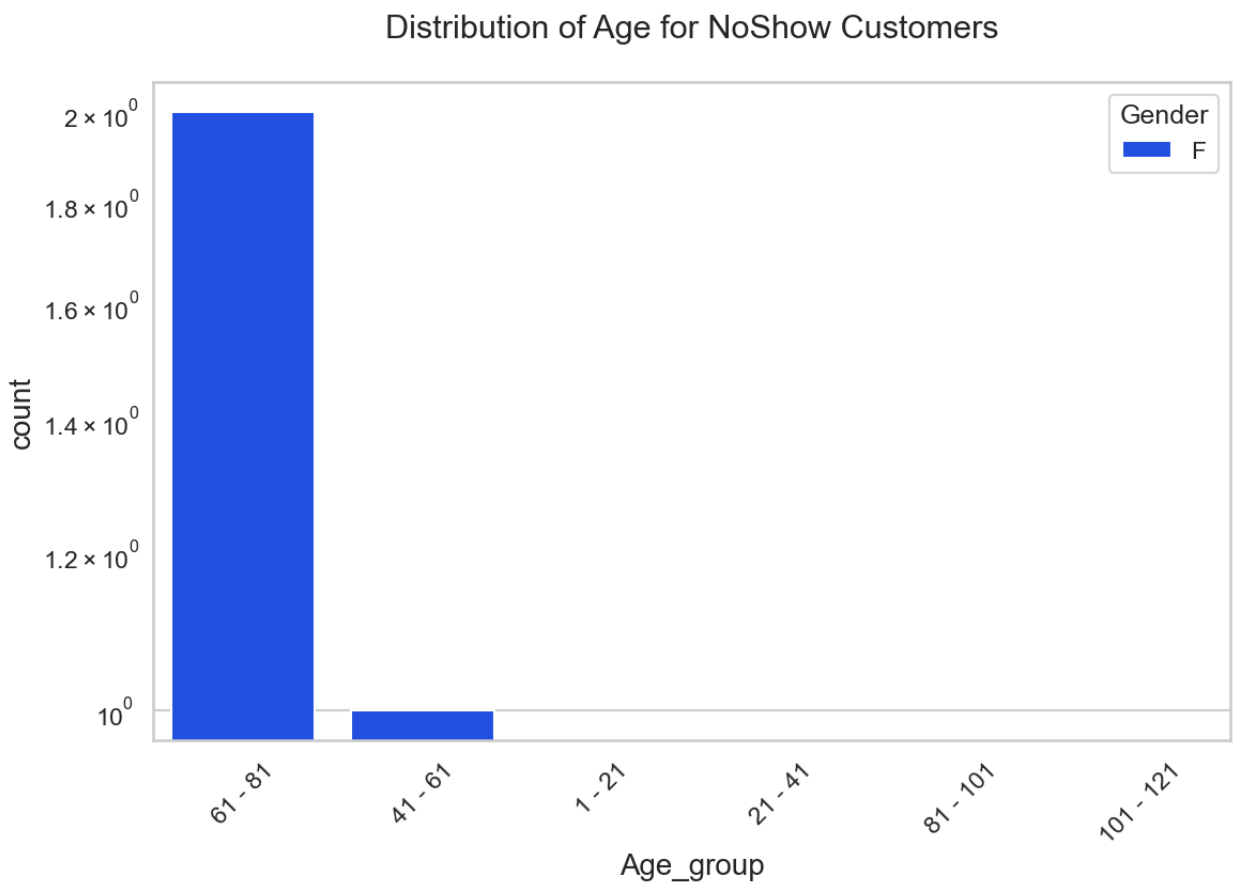
In [57]:
```
uniplot(new_df1_target1, col='Hypertension', title='Distribution of Ge
```



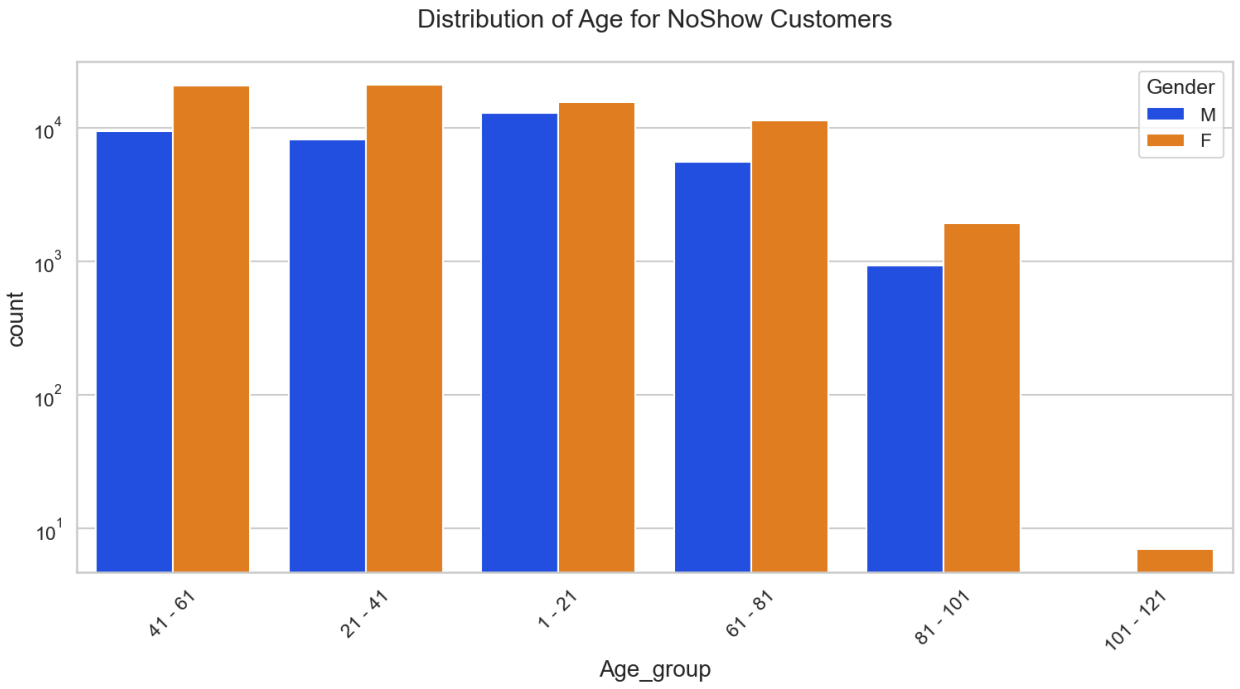
In [58]:
```
uniplot(new_df1_target0,col='Hypertension',title='Distribution of Gend
```



In [59]:
```
uniplot(new_df1_target1,col='Age_group',title='Distribution of Age for
```



In [60]:
```
uniplot(new_df1_target0,col='Age_group',title='Distribution of Age for
```



EDA Findings

1. Female patients have taken more appointments then male patients
2. Ratio of Nohow and Show is almost equal for age group except Age 0 and Age 1 with 80% show rate for each age group
3. Each Neighbourhood have almost 80% show rate
4. There are 99666 patients without Scholarship and out of them around 80% have come for the visit and out of the 21801 patients with Scholarship around 75% of them have come for the visit.
5. there are around 88,726 patients without Hypertension and out of them around 78% have come for the visit and Out of the 21801 patients with Hypertension around 85% of them have come for the visit.
6. there are around 102,584 patients without Diabetes and out of them around 80% have come for the visit and Out of the 7,943 patients with Diabetes around 83% of them have come for the visit.
7. there are around 75,045 patients who have not received SMS and out of them around 84% have come for the visit and out of the 35,482 patients who have received SMS around 72% of them have come for the visit.
8. there is no appointments on sunday and on saturday appointments are very less in comparision to other week days

In [ ]: