## https://www.kaggle.com/datasets/lava18/google-play-store-apps (https://www.kaggle.com/datasets/lava18/google-play-store-apps) ¶

In [2]:

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

In [4]:

```
df=pd.read_csv("clean_gpaydata.csv")
df.head(5)
```

Out[4]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19.0 | 10000 | Free | 0.0 | Everyone | |
| **1** | Coloring book moana | ART_AND_DESIGN | 3.9 | 967 | 14.0 | 500000 | Free | 0.0 | Everyone | De |
| **2** | U Launcher Lite – FREE Live Cool Themes, Hide ... | ART_AND_DESIGN | 4.7 | 87510 | 8.7 | 5000000 | Free | 0.0 | Everyone | |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25.0 | 50000000 | Free | 0.0 | Teen | |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8 | 100000 | Free | 0.0 | Everyone | Des |

In [5]:

```
# check null values
df.isnull().sum()
```

Out[5]:

```
App                  0
Category             0
Rating            1474
Reviews              0
Size              1695
Installs             0
Type                 1
Price                0
Content Rating       0
Genres               0
Last Updated         0
Current Ver          8
Android Ver          2
day                  0
date                 0
month                0
yrar                 0
dtype: int64
```

In [7]:

```
# check total null values
df.isnull().sum().sum()
```

Out[7]:

```
3180
```

In [11]:

```
# sort null values in descending order
df.isnull().sum().sort_values(ascending= False)
```

Out[11]:

```
Size              1695
Rating            1474
Current Ver          8
Android Ver          2
Type                 1
App                  0
Last Updated         0
month                0
date                 0
day                  0
Content Rating       0
Genres               0
Category             0
Price                0
Installs             0
Reviews              0
yrar                 0
dtype: int64
```

## show null values in count plot

In [14]:

```
df.shape
```

Out[14]:

```
(10840, 17)
```

In [16]:

```
# drope duplicate values
df= df.drop_duplicates()
```

In [17]:

```
df.shape
```

Out[17]:

```
(10357, 17)
```

In [18]:

```
len(df)
```

Out[18]:

```
10357
```

## plot bar plot

In [19]:

```
null_counts=df.isnull().sum().sort_values(ascending= False)/len(df)
```

In [20]:

```
null_counts
```

Out[20]:

```
Size              0.147340
Rating            0.141450
Current Ver       0.000772
Android Ver       0.000193
Type              0.000097
App               0.000000
Last Updated      0.000000
month             0.000000
date              0.000000
day               0.000000
Content Rating    0.000000
Genres            0.000000
Category          0.000000
Price             0.000000
Installs          0.000000
Reviews           0.000000
yrar              0.000000
dtype: float64
```

In [24]:

```
len(null_counts)
```

Out[24]:

```
17
```

In [115]:

```
null_counts.index
```

Out[115]:

```
Index(['Size', 'Rating', 'Current Ver', 'Android Ver', 'Type', 'App',
       'Last Updated', 'month', 'date', 'day', 'Content Rating', 'Genres',
       'Category', 'Price', 'Installs', 'Reviews', 'yrar'],
      dtype='object')
```

In [116]:

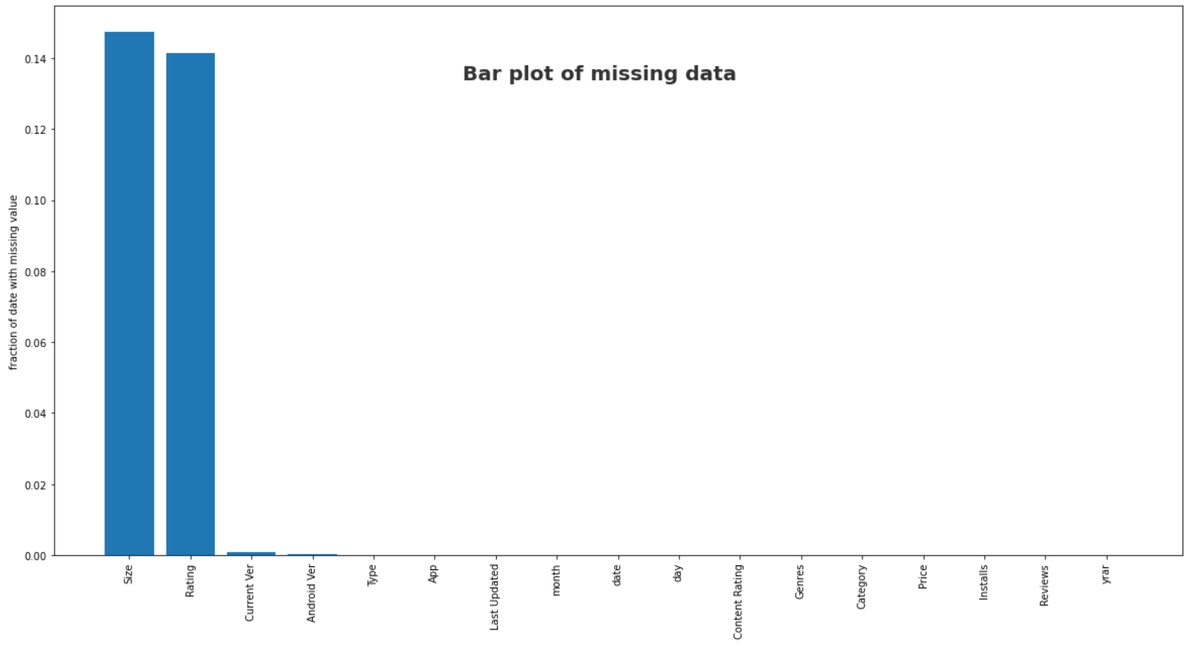
```
np.arange(len(null_counts))
```

Out[116]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

In [114]:

```
plt.figure(figsize=(20,10))
plt.suptitle("Bar plot of missing data" ,fontsize= 20,fontweight= "bold",alpha=0.8,y=0.

# we will get the sum of percentage count by multiplying len(df)

null_counts=df.isnull().sum().sort_values(ascending= False)/len(df)

plt.xticks(np.arange(len(null_counts)),null_counts.index,rotation ="vertical")
plt.ylabel("fraction of date with missing value")

plt.bar(np.arange(len(null_counts)),null_counts)
plt.show()

# Ask in doubt clearning session
```



In [31]:

```
# create a copy of dataa
df_copy= df.copy()
```

In [32]:

```
df_copy.head(3)
```

Out[32]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19.0 | 10000 | Free | 0.0 | Everyone | Ar |
| **1** | Coloring book moana | ART_AND_DESIGN | 3.9 | 967 | 14.0 | 500000 | Free | 0.0 | Everyone | Desig |
| **2** | U Launcher Lite – FREE Live Cool Themes, Hide ... | ART_AND_DESIGN | 4.7 | 87510 | 8.7 | 5000000 | Free | 0.0 | Everyone | Ar |

## get a columns which has null values

In [38]:

```
cols= [fea for fea in df_copy.columns if df_copy[fea].isnull().mean()*100]
```

In [40]:

```
cols
```

Out[40]:

```
['Rating', 'Size', 'Type', 'Current Ver', 'Android Ver']
```

In [42]:

```
# five columns which has null values
len(cols)
```

Out[42]:

```
5
```

In [46]:

```
df_copy["Rating"].isnull().mean()
```

Out[46]:

```
0.14145022689968137
```

In [47]:

```
# to get %
df_copy["Rating"].isnull().mean()*100
```

Out[47]:

```
14.145022689968137
```

In [49]:

```
# coluns having no null value
df_copy["Installs"].isnull().mean()*100
```

Out[49]:

```
0.0
```

in python any numeric value less than 0 or greater than 0 but not equal to zero gives true value

In [50]:

```
if True :
    print("shweta")

```

```
shweta
```

In [52]:

```
if False :
    print("shweta")
```

In [45]:

```
# meaning of above line no 38
# to get all column names
for i in df_copy.columns:
    print(i)
```

```
App
Category
Rating
Reviews
Size
Installs
Type
Price
Content Rating
Genres
Last Updated
Current Ver
Android Ver
day
date
month
yrar
```

In [53]:

```
cols= [var for var in df_copy.columns if df_copy[var].isnull().mean()*100]
cols
```

Out[53]:

```
['Rating', 'Size', 'Type', 'Current Ver', 'Android Ver']
```

In [54]:

```
# data frame only contains null values
df_copy[cols]
```

Out[54]:

| | Rating | Size | Type | Current Ver | Android Ver |
|---|---|---|---|---|---|
| **0** | 4.1 | 19.0 | Free | 1.0.0 | 4.0.3 and up |
| **1** | 3.9 | 14.0 | Free | 2.0.0 | 4.0.3 and up |
| **2** | 4.7 | 8.7 | Free | 1.2.4 | 4.0.3 and up |
| **3** | 4.5 | 25.0 | Free | Varies with device | 4.2 and up |
| **4** | 4.3 | 2.8 | Free | 1.1 | 4.4 and up |
| **...** | ... | ... | ... | ... | ... |
| **10835** | 4.5 | 53.0 | Free | 1.48 | 4.1 and up |
| **10836** | 5.0 | 3.6 | Free | 1.0 | 4.1 and up |
| **10837** | NaN | 9.5 | Free | 1.0 | 2.2 and up |
| **10838** | 4.5 | NaN | Free | Varies with device | Varies with device |
| **10839** | 4.5 | 19.0 | Free | Varies with device | Varies with device |

10357 rows × 5 columns

In [55]:

```
df_copy[cols].dropna()
```

Out[55]:

| | Rating | Size | Type | Current Ver | Android Ver |
|---|---|---|---|---|---|
| **0** | 4.1 | 19.000 | Free | 1.0.0 | 4.0.3 and up |
| **1** | 3.9 | 14.000 | Free | 2.0.0 | 4.0.3 and up |
| **2** | 4.7 | 8.700 | Free | 1.2.4 | 4.0.3 and up |
| **3** | 4.5 | 25.000 | Free | Varies with device | 4.2 and up |
| **4** | 4.3 | 2.800 | Free | 1.1 | 4.4 and up |
| **...** | ... | ... | ... | ... | ... |
| **10832** | 4.8 | 0.619 | Free | 0.8 | 2.2 and up |
| **10833** | 4.0 | 2.600 | Free | 1.0.0 | 4.1 and up |
| **10835** | 4.5 | 53.000 | Free | 1.48 | 4.1 and up |
| **10836** | 5.0 | 3.600 | Free | 1.0 | 4.1 and up |
| **10839** | 4.5 | 19.000 | Free | Varies with device | Varies with device |

7418 rows × 5 columns

In [56]:

```
# assign new variable
drope_df=df_copy[cols].dropna()
```

In [57]:

```
drope_df.shape
```

Out[57]:

```
(7418, 5)
```

In [58]:

```
df_copy.shape
```

Out[58]:

```
(10357, 17)
```

In [ ]:

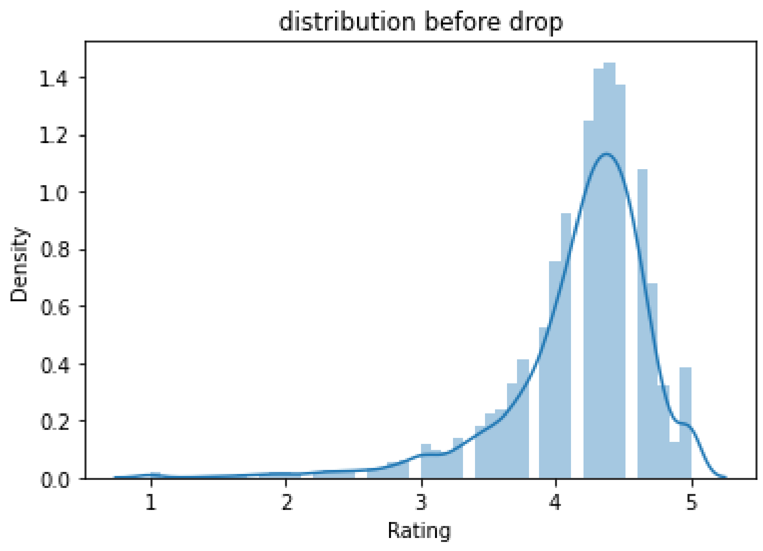
```
# after droping the null value checkbthe distribution of data
```

In [60]:

```
sns.distplot(df_copy["Rating"])
plt.title(" distribution before drop ")
```

Out[60]:

```
Text(0.5, 1.0, ' distribution before drop ')
```

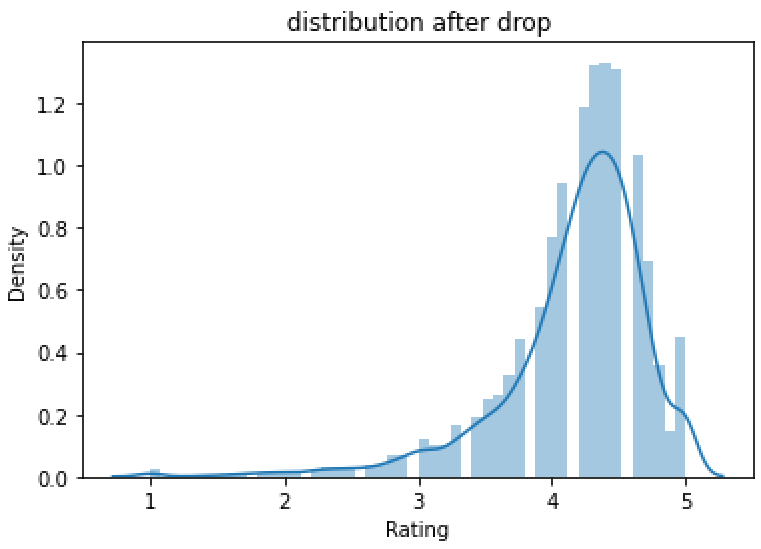


In [62]:

```
sns.distplot(drope_df["Rating"])
plt.title(" distribution after drop ")
```

Out[62]:

```
Text(0.5, 1.0, ' distribution after drop ')
```



# Observations

**Distribution does not change after drope**

# Mean Median and Mode methode

In [67]:

```
# create a copy of dataframe
df_copy_MEANMODE=df.copy()

```

In [69]:

```
df_copy_MEANMODE
```

Out[69]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19.0 | 10000 | Free | 0.0 | E |
| **1** | Coloring book moana | ART_AND_DESIGN | 3.9 | 967 | 14.0 | 500000 | Free | 0.0 | E |
| **2** | U Launcher Lite – FREE Live Cool Themes, Hide ... | ART_AND_DESIGN | 4.7 | 87510 | 8.7 | 5000000 | Free | 0.0 | E |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25.0 | 50000000 | Free | 0.0 | |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8 | 100000 | Free | 0.0 | E |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **10835** | Sya9a Maroc - FR | FAMILY | 4.5 | 38 | 53.0 | 5000 | Free | 0.0 | E |
| **10836** | Fr. Mike Schmitz Audio Teachings | FAMILY | 5.0 | 4 | 3.6 | 100 | Free | 0.0 | E |
| **10837** | Parkinson Exercices FR | MEDICAL | NaN | 3 | 9.5 | 1000 | Free | 0.0 | E |
| **10838** | The SCP Foundation DB fr nn5n | BOOKS_AND_REFERENCE | 4.5 | 114 | NaN | 1000 | Free | 0.0 | |
| **10839** | iHoroscope - 2018 Daily Horoscope & Astrology | LIFESTYLE | 4.5 | 398307 | 19.0 | 10000000 | Free | 0.0 | E |

10357 rows × 17 columns

In [71]:

```
# fill the null values in size column with mean
#df_copy_MEANMODE['Size'].fillna()--------Pass mean inside()
df_copy_MEANMODE['Size'].fillna(df_copy_MEANMODE['Size'].mean())
```

Out[71]:

```
0        19.00000
1        14.00000
2         8.70000
3        25.00000
4         2.80000
           ...
10835    53.00000
10836     3.60000
10837     9.50000
10838    21.28875
10839    19.00000
Name: Size, Length: 10357, dtype: float64
```

In [75]:

```
# check null values
df_copy_MEANMODE['Size'].fillna(df_copy_MEANMODE['Size'].mean()).isnull().sum()
```

Out[75]:

```
0
```

## fill with median

In [76]:

```
df_copy_MEANMODE['Size'].fillna(df_copy_MEANMODE['Size'].median())
```

Out[76]:

```
0        19.0
1        14.0
2         8.7
3        25.0
4         2.8
         ...
10835    53.0
10836     3.6
10837     9.5
10838    13.0
10839    19.0
Name: Size, Length: 10357, dtype: float64
```

In [77]:

```
df_copy_MEANMODE['Size'].fillna(df_copy_MEANMODE['Size'].median()).isna().sum()
```

Out[77]:

```
0
```

In [ ]:

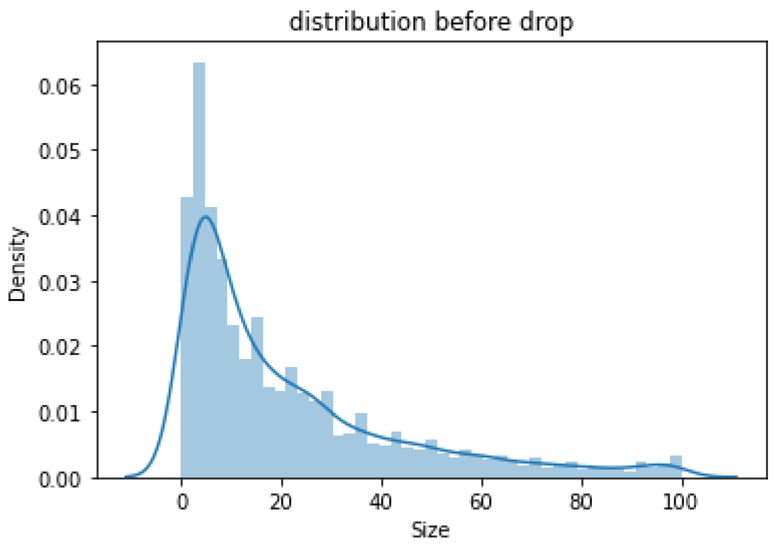
```
# check the distribution

```

In [78]:

```
sns.distplot(df_copy["Size"])
plt.title(" Original distribution before fill ")
```

Out[78]:

```
Text(0.5, 1.0, ' distribution before drop ')
```

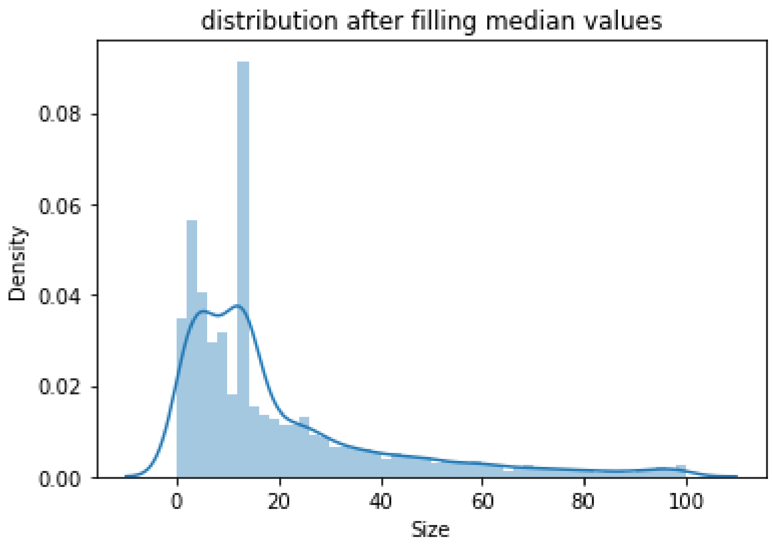


In [80]:

```
sns.distplot(df_copy_MEANMODE['Size'].fillna(df_copy_MEANMODE['Size'].median()))
plt.title(" distribution after filling median values ")
```

Out[80]:

```
Text(0.5, 1.0, ' distribution after filling median values ')
```

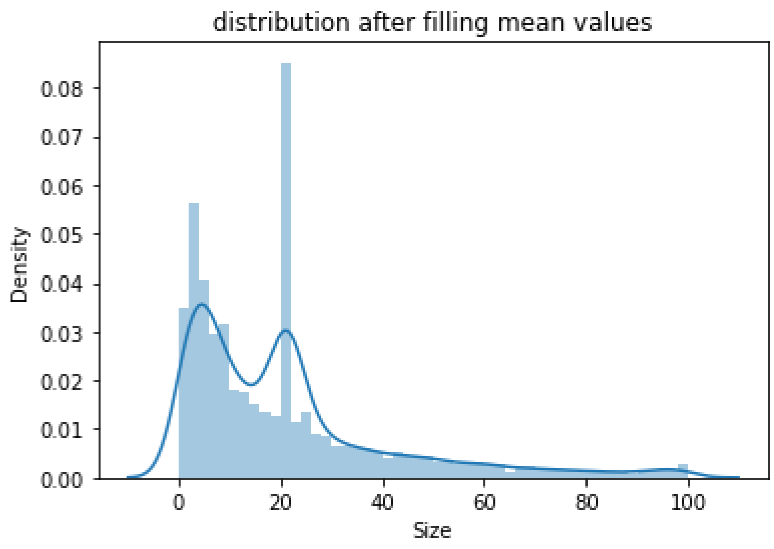


In [81]:

```
sns.distplot(df_copy_MEANMODE['Size'].fillna(df_copy_MEANMODE['Size'].mean()))
plt.title(" distribution after filling mean values ")
```

Out[81]:

```
Text(0.5, 1.0, ' distribution after filling mean values ')
```



## Distribution is going to change

## Handle missing values by random sample imputation

with respect to size

In [99]:

```
# create a copy of dataframe
df_RANDIMPU=df.copy()
df_RANDIMPU.head(3)
```

Out[99]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19.0 | 10000 | Free | 0.0 | Everyone | Ar |
| **1** | Coloring book moana | ART_AND_DESIGN | 3.9 | 967 | 14.0 | 500000 | Free | 0.0 | Everyone | Desiç |
| **2** | U Launcher Lite – FREE Live Cool Themes, Hide ... | ART_AND_DESIGN | 4.7 | 87510 | 8.7 | 5000000 | Free | 0.0 | Everyone | Ar |

In [100]:

```
#
df_RANDIMPU["Size"].isna().sum()
```

Out[100]:

```
1526
```

In [101]:

```
# collect the sample from this data but first drop the null values
df_RANDIMPU["Size"].dropna().sample()  # its giving one sample
```

Out[101]:

```
10470    4.0
Name: Size, dtype: float64
```

In [102]:

```
# we need collect the samples as many as the null values in the data as null values are
df_RANDIMPU["Size"].dropna().sample(1526)
```

Out[102]:

```
8421    27.0
6113    80.0
4057    18.0
3544    11.0
7701     3.3
        ...
8288     8.7
7623    64.0
6196     4.9
460      1.6
4750    16.0
Name: Size, Length: 1526, dtype: float64
```

In [89]:

```
random_samplecollect=df_RANDIMPU["Size"].dropna().sample(1526)
```

In [90]:

```
random_samplecollect
```

Out[90]:

```
7421    92.0
717     21.0
1337    15.0
7594    31.0
8845     2.6
        ...
5520    37.0
8426    85.0
9723    47.0
9169    43.0
6983    44.0
Name: Size, Length: 1526, dtype: float64
```

In [91]:

```
# data frame of null values
df_RANDIMPU[df_RANDIMPU["Size"].isnull()]
```

Out[91]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price |
|---|---|---|---|---|---|---|---|---|
| **37** | Floor Plan Creator | ART_AND_DESIGN | 4.1 | 36639 | NaN | 5000000 | Free | 0.0 |
| **42** | Textgram - write on photos | ART_AND_DESIGN | 4.4 | 295221 | NaN | 10000000 | Free | 0.0 |
| **52** | Used Cars and Trucks for Sale | AUTO_AND_VEHICLES | 4.6 | 17057 | NaN | 1000000 | Free | 0.0 |
| **67** | Ulysse Speedometer | AUTO_AND_VEHICLES | 4.3 | 40211 | NaN | 5000000 | Free | 0.0 |
| **68** | REPUVE | AUTO_AND_VEHICLES | 3.9 | 356 | NaN | 100000 | Free | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **10712** | My Earthquake Alerts - US & Worldwide Earthquakes | WEATHER | 4.4 | 3471 | NaN | 100000 | Free | 0.0 |
| **10724** | Posta App | MAPS_AND_NAVIGATION | 3.6 | 8 | NaN | 1000 | Free | 0.0 |
| **10764** | Chat For Strangers - Video Chat | SOCIAL | 3.4 | 622 | NaN | 100000 | Free | 0.0 |
| **10825** | Frim: get new friends on local chat rooms | SOCIAL | 4.0 | 88486 | NaN | 5000000 | Free | 0.0 |
| **10838** | The SCP Foundation DB fr nn5n | BOOKS_AND_REFERENCE | 4.5 | 114 | NaN | 1000 | Free | 0.0 |

1526 rows × 17 columns

In [93]:

```
# get the indexdes having null values of above frame
df_RANDIMPU[df_RANDIMPU["Size"].isnull()].index
```

Out[93]:

```
Int64Index([   37,    42,    52,    67,    68,    73,    85,    88,    89,
               92,
            ...
            10646, 10678, 10680, 10706, 10711, 10712, 10724, 10764, 10825,
            10838],
           dtype='int64', length=1526)
```

In [103]:

```
# take this inside new variable
random_samplecollect.index =df_RANDIMPU[df_RANDIMPU["Size"].isnull()].index
```

In [ ]:

```
# with the help of location we can  fill the null values
# take a location df_RANDIMPU.loc[]
# pass the particular feature inside it df_RANDIMPU.loc[df_RANDIMPU["Size"]]
#Check wheather it is isnull   df_RANDIMPU.loc[df_RANDIMPU["Size"].isnull()]
# fill the null values with random samples   df_RANDIMPU.loc[df_RANDIMPU["Size"].isnull

```

In [106]:

```
df_RANDIMPU.loc[df_RANDIMPU["Size"].isnull(),"Size"]= random_samplecollect
```

In [109]:

```
df_RANDIMPU.loc[df_RANDIMPU["Size"].isnull(),"Size"]
```

Out[109]:

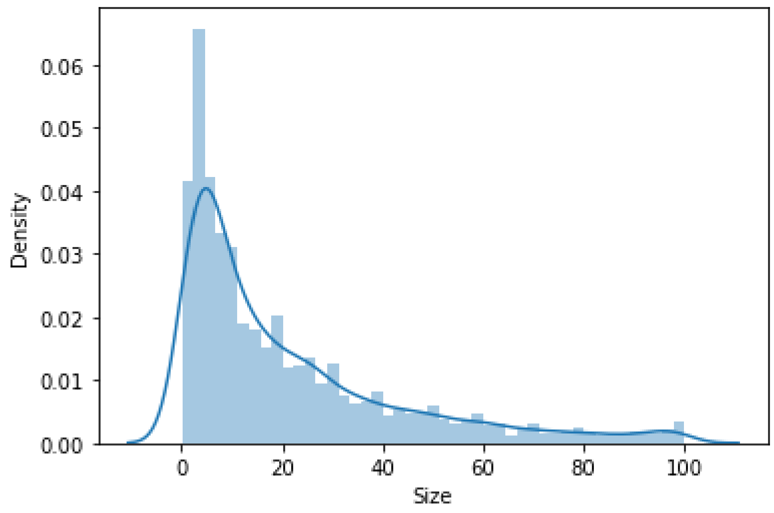
```
Series([], Name: Size, dtype: bool)
```

In [113]:

```
# check the distribution after imputation
sns.distplot(df_RANDIMPU["Size"])
```

Out[113]:

```
<AxesSubplot:xlabel='Size', ylabel='Density'>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: