Aiman Alam

Instructor - prof. Jan Lorenz

Data Visualizations and Data Story Telling

Rapes in India - Visualizations and Anaylsis

Importing Libraries

In [1]:
```
#Importing important libraries used for upcoming anaylsis

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
import seaborn as sns
```

Accessing Dataset

Victims of Rape - 2016 to 2021

In [2]:
```
# Read the datasets

D16 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\Victims of
D17 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\Victims of
D18 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\Victims of
D19 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\Victims of
D20 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\Victims of
D21 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\Victims of
```

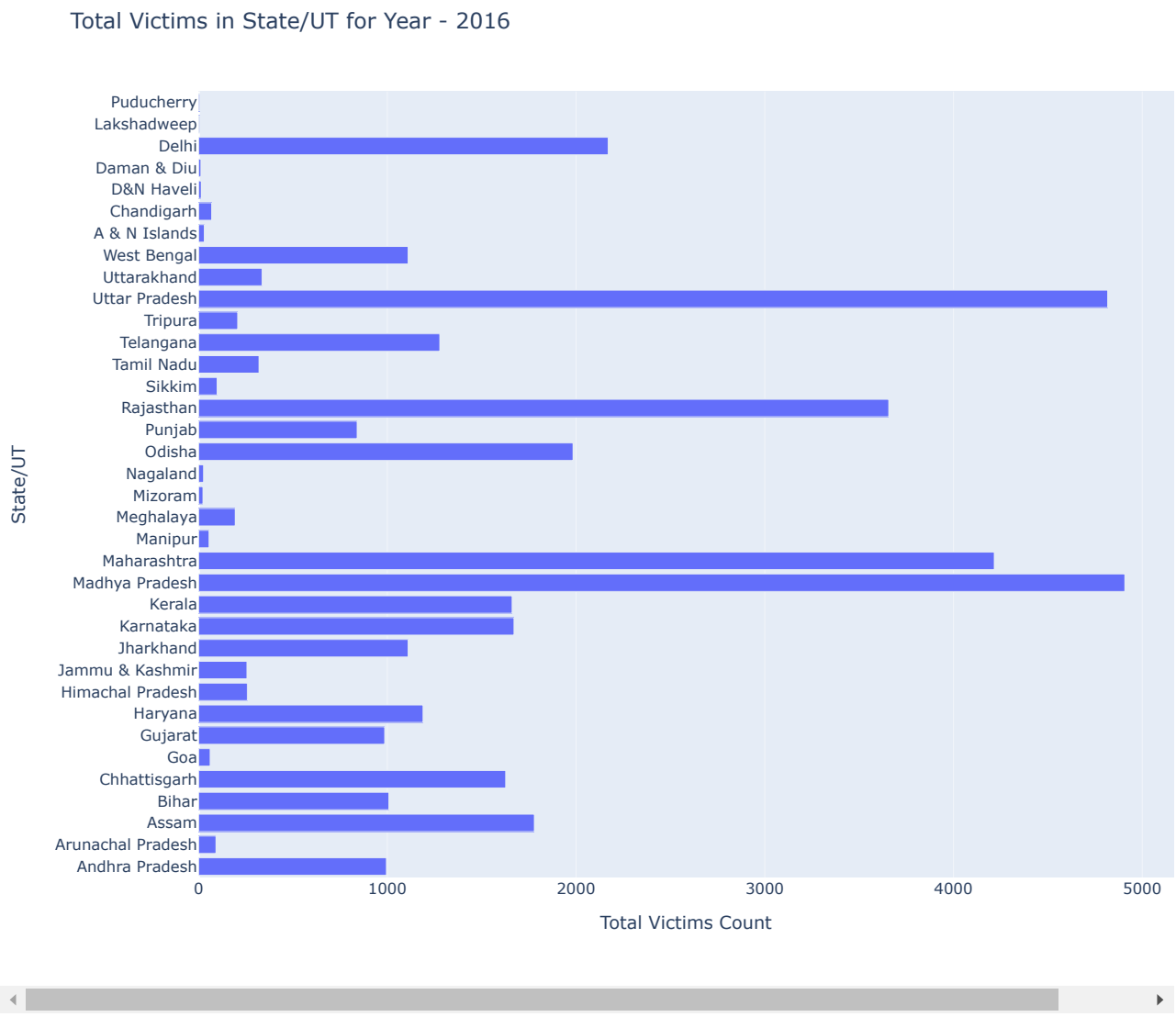
Analysis of Year - 2016

In [3]:
```
D16.head()
```

Out[3]:

	State/UT	Cases Reported	Victims below 6 years - Girls	Victims 6 - 12 years - Girls	Victims 12 - 16 Years - Girls	Victims 16 - 18 years - Girls	Total Victims - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Victims above 60 years - Women	Total Victims - Women	Total Victims
0	Andhra Pradesh	994	17	43	142	261	463	436	85	9	2	532	995
1	Arunachal Pradesh	92	3	12	25	10	50	32	7	2	1	42	92
2	Assam	1779	0	52	147	265	464	938	373	4	0	1315	1779
3	Bihar	1008	0	7	41	121	169	684	146	9	0	839	1008
4	Chhattisgarh	1626	33	79	340	532	984	475	150	13	5	643	1627

In [4]:
```
#Lets visualize the above state list

# D18 is the DataFrame with 'State/UT' and 'Total Victims' columns

fig = px.bar(D16, y='State/UT', x='Total Victims', title='Total Victims in State/UT for Year - 2016',
             labels={'Total Victims': 'Total Victims Count', 'State/UT': 'State/UT'})

# Increase the size of the graph
fig.update_layout(height=800, width=1000)

# Show the bar chart
fig.show()
```



Most targeted age group of rape - 2016

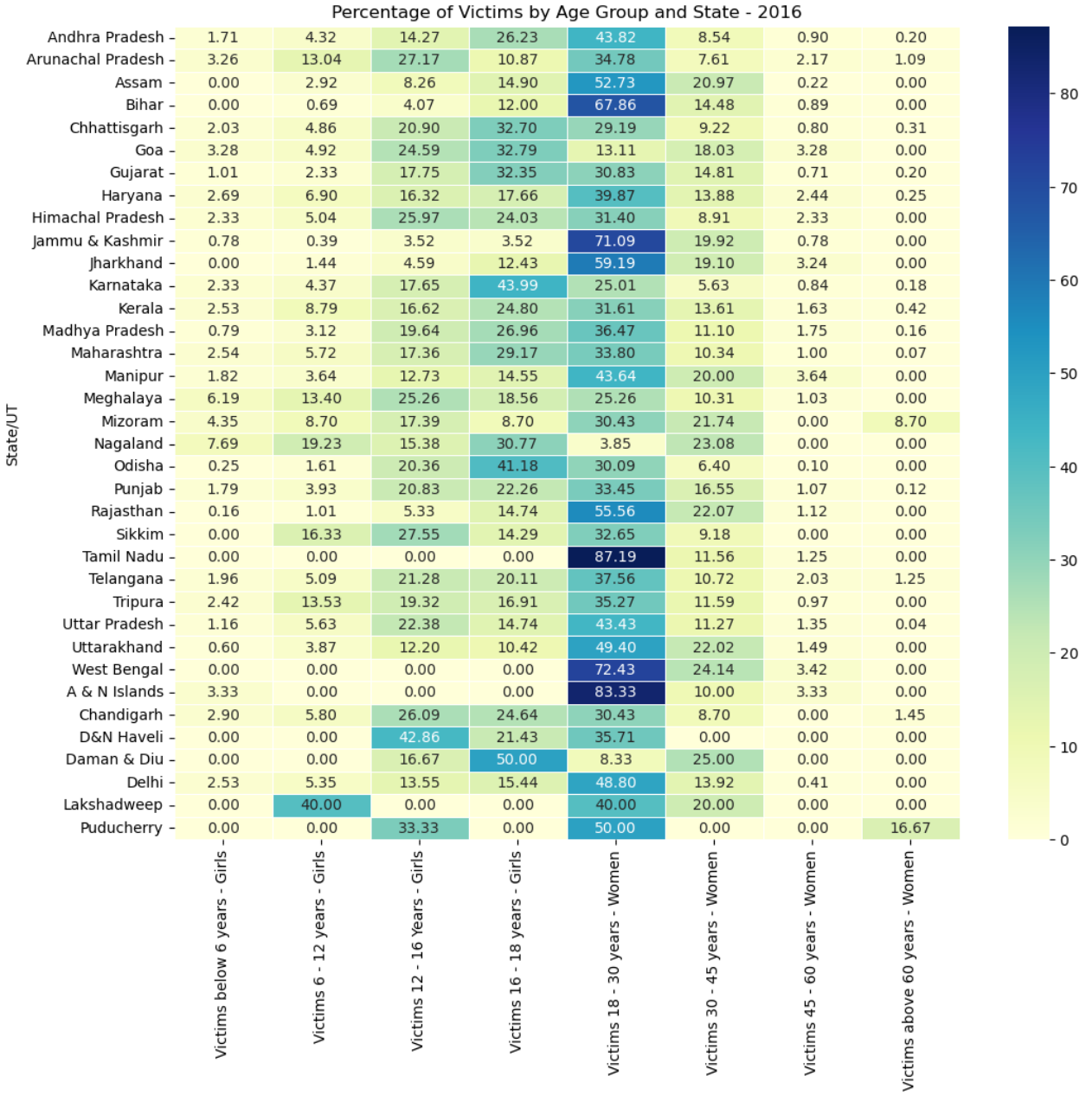
In [5]:
```
# Create a new dataset for percentages
percentage_data = D16.copy()

# Calculate percentages for each age group
age_groups = ['Victims below 6 years - Girls', 'Victims 6 - 12 years - Girls', 'Victims 12 - 16 Years - Girls',
              'Victims 16 - 18 years - Girls', 'Victims 18 - 30 years - Women', 'Victims 30 - 45 years - Women',
              'Victims 45 - 60 years - Women', 'Victims above 60 years - Women']

for age_group in age_groups:
    percentage_data[age_group] = ((D16[age_group] / D16['Total Victims']) * 100).round(2)

# Drop unnecessary columns (if needed)
percentage_data = percentage_data.drop(columns=['Total Victims - Girls', 'Total Victims - Women', 'Total Victims'])

# Display the resulting dataset
percentage_data.head()
```

Out[5]:

	State/UT	Cases Reported	Victims below 6 years - Girls	Victims 6 - 12 years - Girls	Victims 12 - 16 Years - Girls	Victims 16 - 18 years - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Victims above 60 years - Women
0	Andhra Pradesh	994	1.71	4.32	14.27	26.23	43.82	8.54	0.90	0.20
1	Arunachal Pradesh	92	3.26	13.04	27.17	10.87	34.78	7.61	2.17	1.09
2	Assam	1779	0.00	2.92	8.26	14.90	52.73	20.97	0.22	0.00
3	Bihar	1008	0.00	0.69	4.07	12.00	67.86	14.48	0.89	0.00
4	Chhattisgarh	1626	2.03	4.86	20.90	32.70	29.19	9.22	0.80	0.31

In [6]:
```
# lets see which age group have highest victims via heatmap

# for that we need to remove the total count columns from dataset
#because obviously they will have max value and will be marked highest in the heat map

columns_to_drop = ['Cases Reported']

# new DataFrame with specified columns dropped

percentage_data = percentage_data.drop(columns=columns_to_drop)
```

In [7]:
```
# Set the state column as the index for better visualization
percentage_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(percentage_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Victims by Age Group and State - 2016')
plt.show()
```



Analysis of Year - 2017

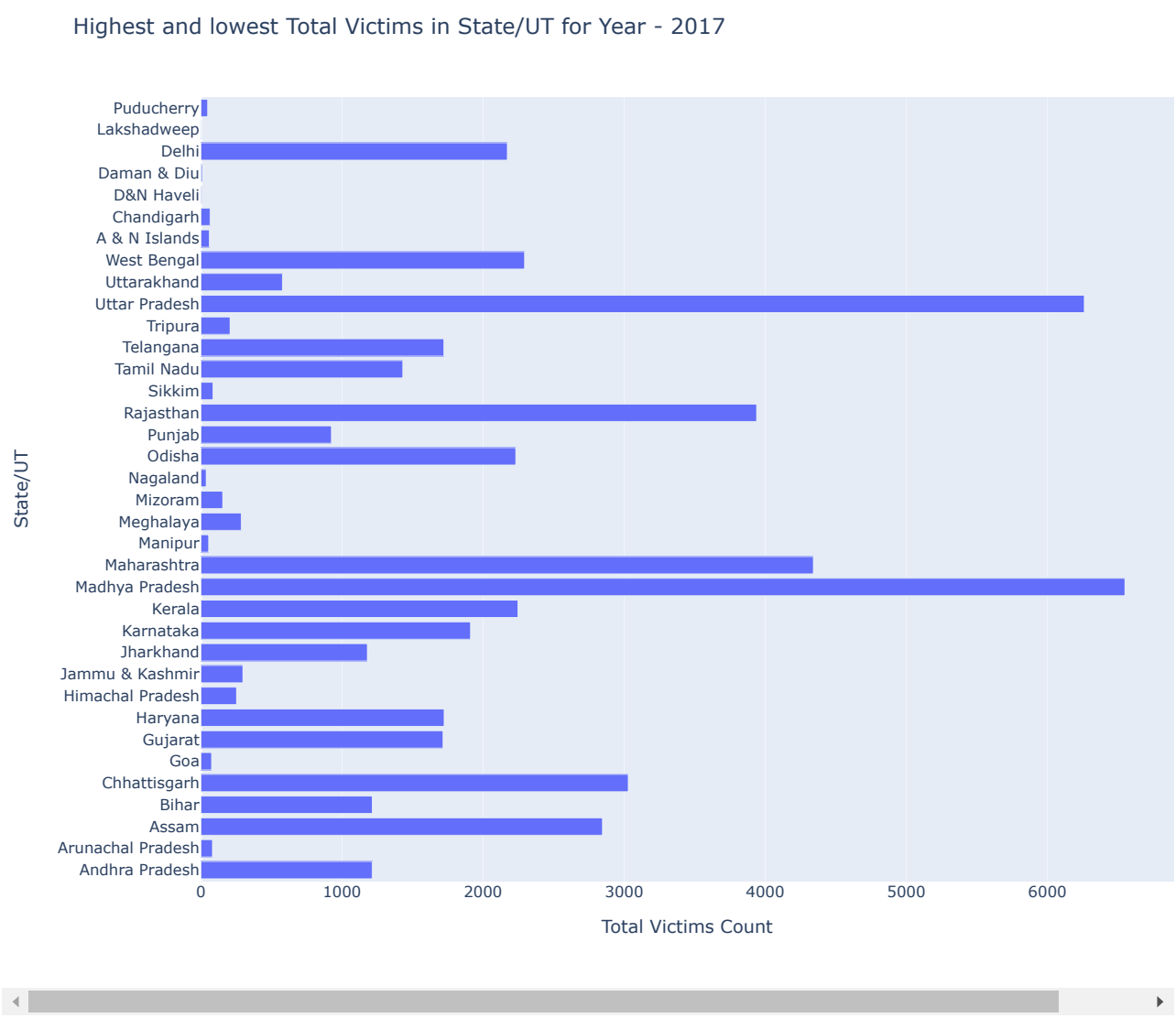
In [8]:
```
D17.head()
```

Out[8]:

	State/UT	Cases Reported	Victims Below 6 years- Girls	Victims 6- 12 years - Girls	Victims 12 to 16 years - Girls	Victims 16 to 18 years - Girls	Total Victims - Girls	Victims 18 - 30 years- women	Victims 30 - 45 years - women	Victims 45 - 60 Years - Women	Victims above 60 years - Women	Total Victims - Women	Total Victims
0	Andhra Pradesh	988	17	68	248	417	750	333	121	6	4	464	1214
1	Arunachal Pradesh	59	1	15	23	14	53	24	5	0	0	29	82
2	Assam	1772	9	201	337	351	898	1291	582	75	0	1948	2846
3	Bihar	605	2	44	126	427	599	546	68	2	0	616	1215
4	Chhattisgarh	1908	96	166	809	1166	2237	525	245	14	8	792	3029

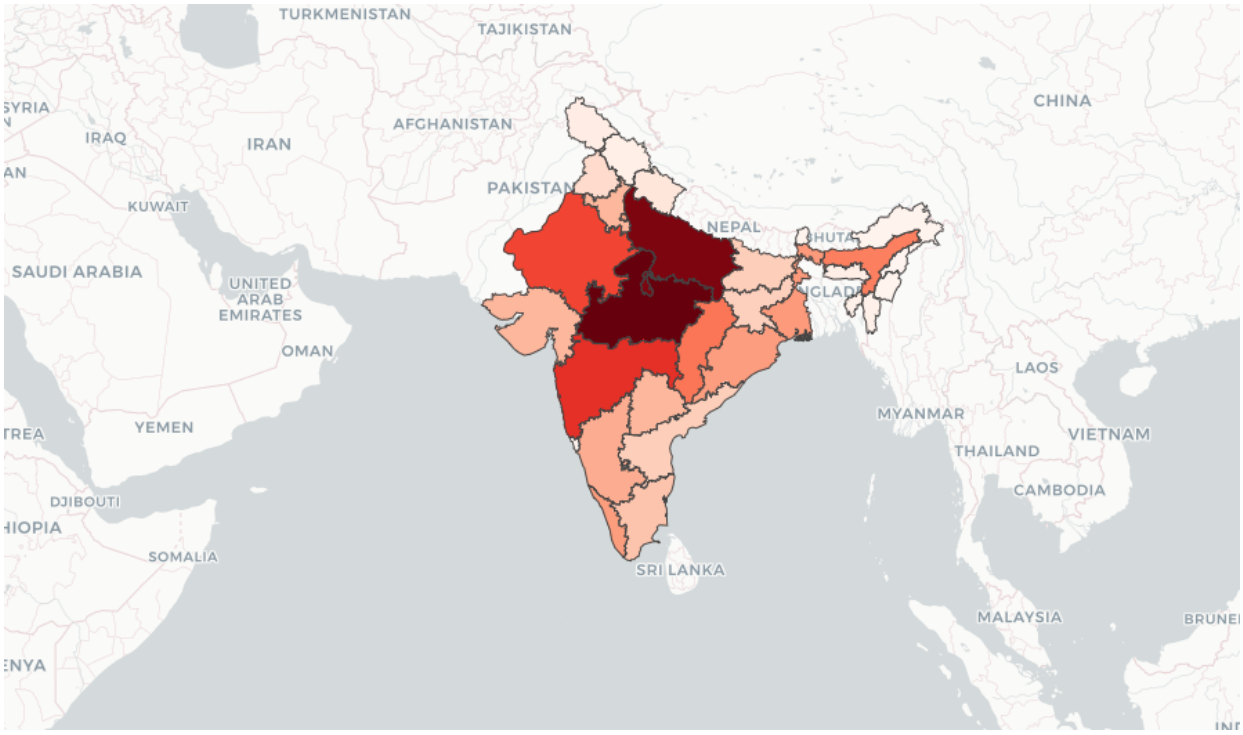
In [9]:

```
# D17 is the DataFrame with 'State/UT' and 'Total Victims' columns

fig = px.bar(D17, y='State/UT', x='Total Victims', title='Highest and lowest Total Victims in State/UT for Year - 2017',
             labels={'Total Victims': 'Total Victims Count', 'State/UT': 'State/UT'})
# Increase the size of the graph
fig.update_layout(height=800, width=1000)
# Show the bar chart
fig.show()
```



Geogriphical Heat Map for Total victims of Rape - 2017

In [10]:
```
import plotly.graph_objects as go



fig = go.Figure(go.Choroplethmapbox(
    geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42
    featureidkey='properties.ST_NM',
    locations=D17['State/UT'],
    z=D17['Total Victims'],
    colorscale='Reds',
    colorbar=dict(thickness=15, ticklen=3),
))

fig.update_layout(
    mapbox=dict(
        center=dict(lat=20, lon=77),
        style="carto-positron",
        zoom=3,
    ),
    margin=dict(l=0, r=0, t=0, b=0),
)

fig.show()
```



Data for Rate of Rape - 2017 - Calculated using population data from Aadhaar card registrations

In [11]:
```
P17 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population 2017.xlsx")
P17.head(3)
```

Out[11]:

	State/UT	Population
0	Andhra Pradesh	52375124
1	Arunachal Pradesh	1506749
2	Assam	34068394

In [12]:
```
# Set the display format for float values
pd.options.display.float_format = '{:.0f}'.format

P17['Female Population'] = 0.48 * P17['Population']


# Display the resulting dataset
P17.head(3)
```

Out[12]:

	State/UT	Population	Female Population
0	Andhra Pradesh	52375124	25140060
1	Arunachal Pradesh	1506749	723240
2	Assam	34068394	16352829

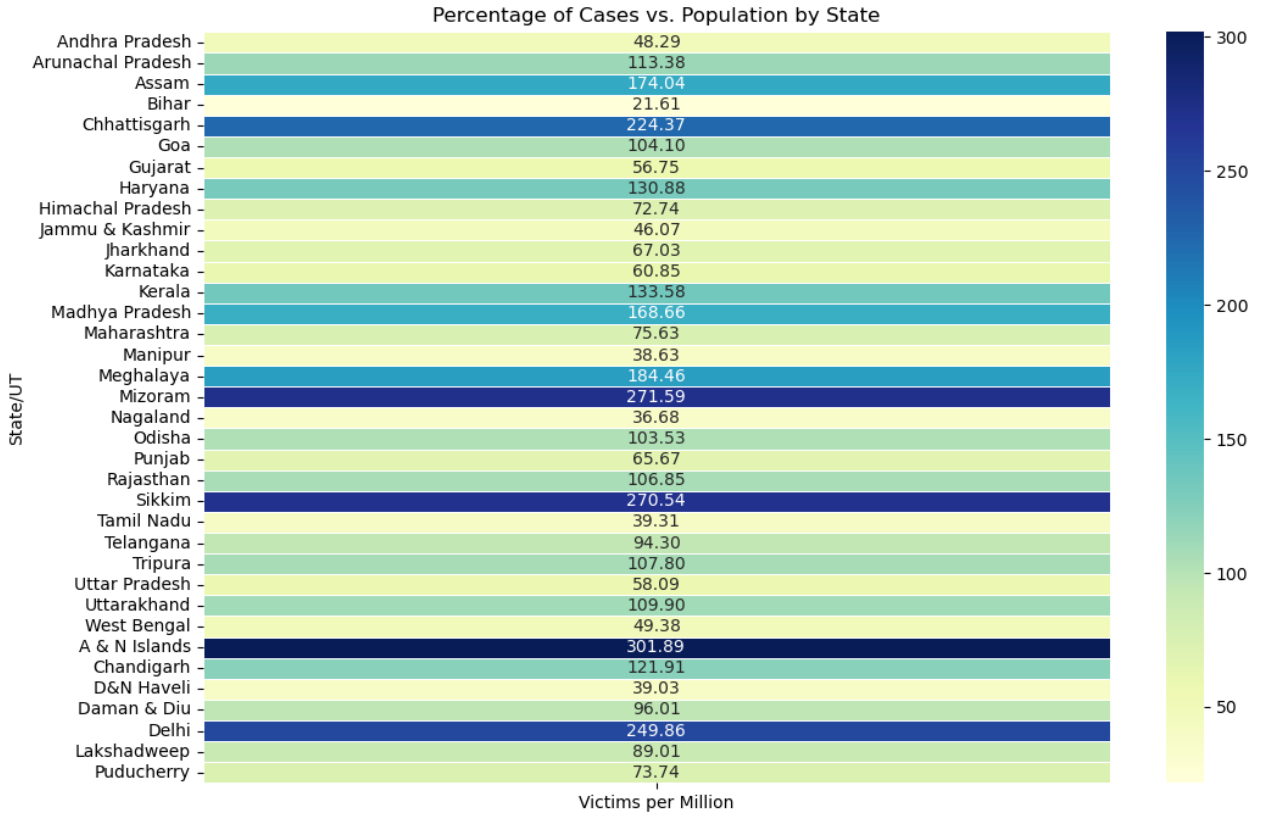
In [13]:
```
# Drop the 'Population' column from 'P17'
P17 = P17.drop(columns=['Population'])

# Merge 'Total Victims' column from 'D17' to 'P17'
P17 = pd.merge(P17, D17[['State/UT', 'Total Victims']], on='State/UT', how='left')
```

In [14]:
```
# Create a new column 'Victims per Million'
P17['Victims per Million'] = (P17['Total Victims'] / P17['Female Population']) * 1000000
P17.head(3)
```

Out[14]:

	State/UT	Female Population	Total Victims	Victims per Million
0	Andhra Pradesh	25140060	1214	48
1	Arunachal Pradesh	723240	82	113
2	Assam	16352829	2846	174

In [15]:
```
# Create a new dataset with relevant columns
heatmap_data = P17[['State/UT', 'Victims per Million']]

# Set the 'State/UT' column as the index for the heatmap data
heatmap_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Cases vs. Population by State')
plt.show()
```

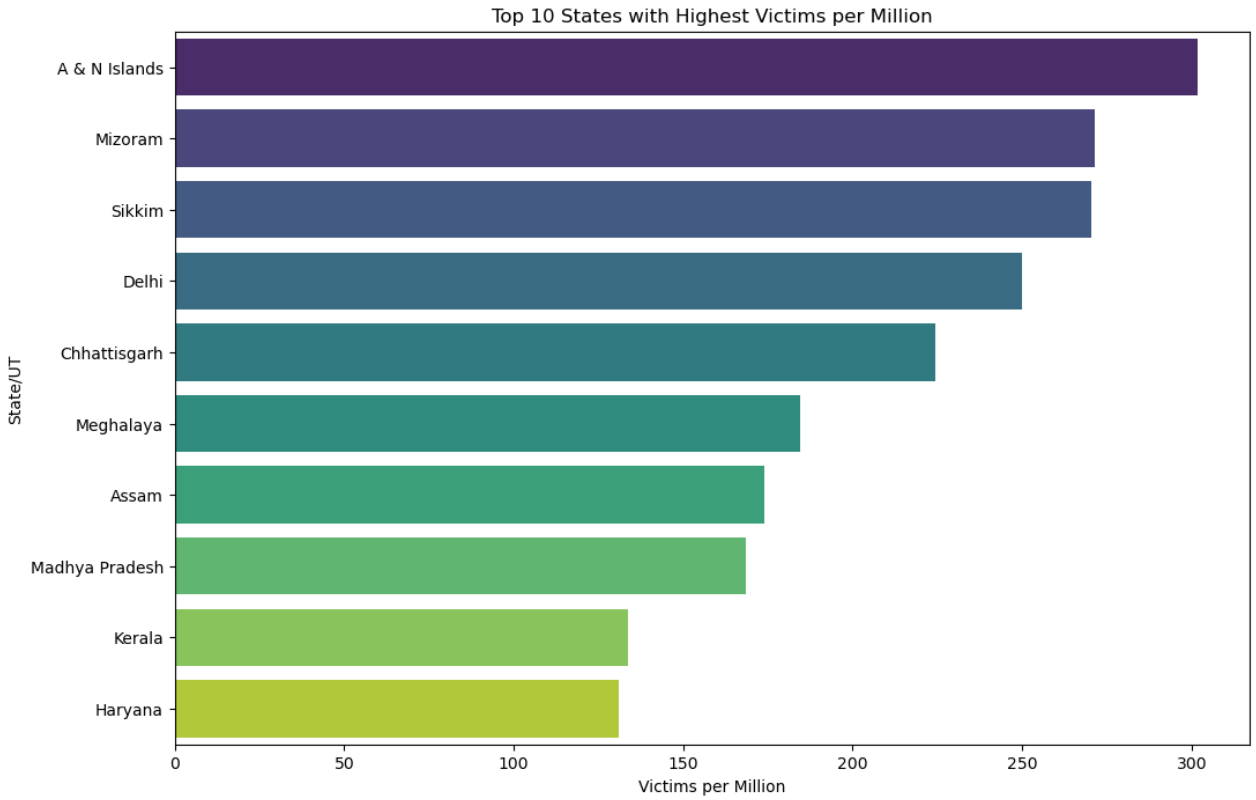


In [16]:
```
# Sort the dataset based on 'Victims per Million' column in descending order
P17_sorted = P17.sort_values(by='Victims per Million', ascending=False)
P17_sorted.head(3)
```

Out[16]:

	State/UT	Female Population	Total Victims	Victims per Million
29	A & N Islands	198747	60	302
17	Mizoram	570706	155	272
22	Sikkim	317880	86	271

In [17]:
```
# Select the top 10 states
top_10_states = P17_sorted.head(10)

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='Victims per Million', y='State/UT', data=top_10_states, palette='viridis')
plt.title('Top 10 States with Highest Victims per Million')
plt.xlabel('Victims per Million')
plt.ylabel('State/UT')
plt.show()
```



Most targeted age group - 2017

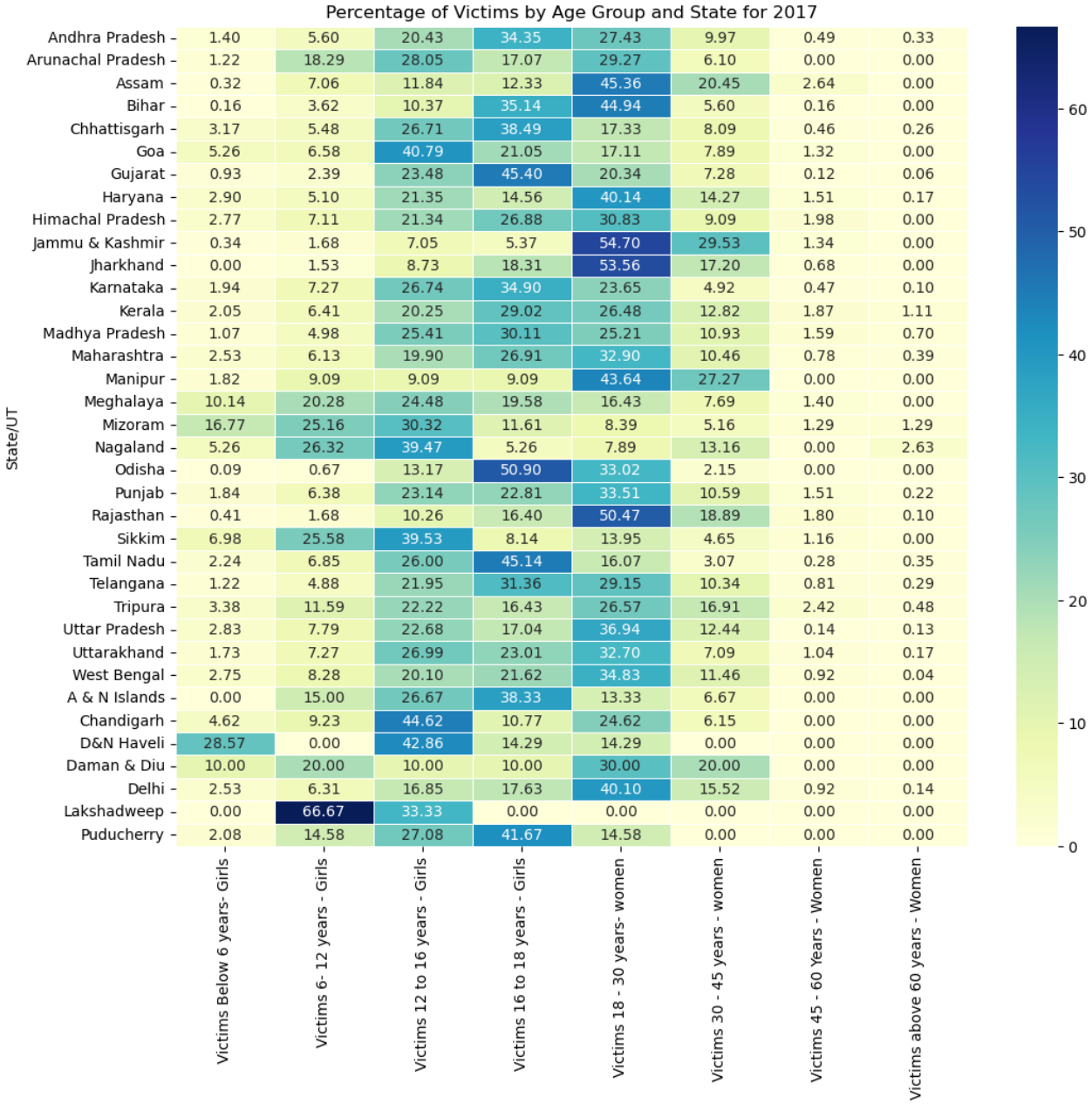
In [18]:
```
# Create a new dataset for percentages
percentage_data = D17.copy()

# Calculate percentages for each age group
age_groups = ['Victims Below 6 years- Girls', 'Victims 6- 12 years - Girls', 'Victims 12 to 16 years - Girls',
              'Victims 16 to 18 years - Girls', 'Total Victims - Girls', 'Victims 18 - 30 years- women',
              'Victims 30 - 45 years - women', 'Victims 45 - 60 Years - Women', 'Victims above 60 years - Women']

for age_group in age_groups:
    percentage_data[age_group] = ((D17[age_group] / D17['Total Victims']) * 100).round(2)

# Drop unnecessary columns (if needed)
percentage_data = percentage_data.drop(columns=['Total Victims - Girls', 'Total Victims - Women', 'Total Victims'])

# Display the resulting dataset
percentage_data.head(2)
```

Out[18]:

	State/UT	Cases Reported	Victims Below 6 years- Girls	Victims 6- 12 years - Girls	Victims 12 to 16 years - Girls	Victims 16 to 18 years - Girls	Victims 18 - 30 years- women	Victims 30 - 45 years - women	Victims 45 - 60 Years - Women	Victims above 60 years - Women
0	Andhra Pradesh	988	1	6	20	34	27	10	0	0
1	Arunachal Pradesh	59	1	18	28	17	29	6	0	0

In [19]:
```
# lets see which age group have highest victims via heatmap

# for that we need to remove the total count columns from dataset
#because obviously they will have max value and will be marked highest in the heat map

columns_to_drop = ['Cases Reported ']

# new DataFrame with specified columns dropped

percentage_data = percentage_data.drop(columns=columns_to_drop)
```

In [20]:
```
# Set the state column as the index for better visualization
percentage_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(percentage_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Victims by Age Group and State for 2017')
plt.show()
```



Analysis of Year - 2018

In [21]:
```
D18.head()
```

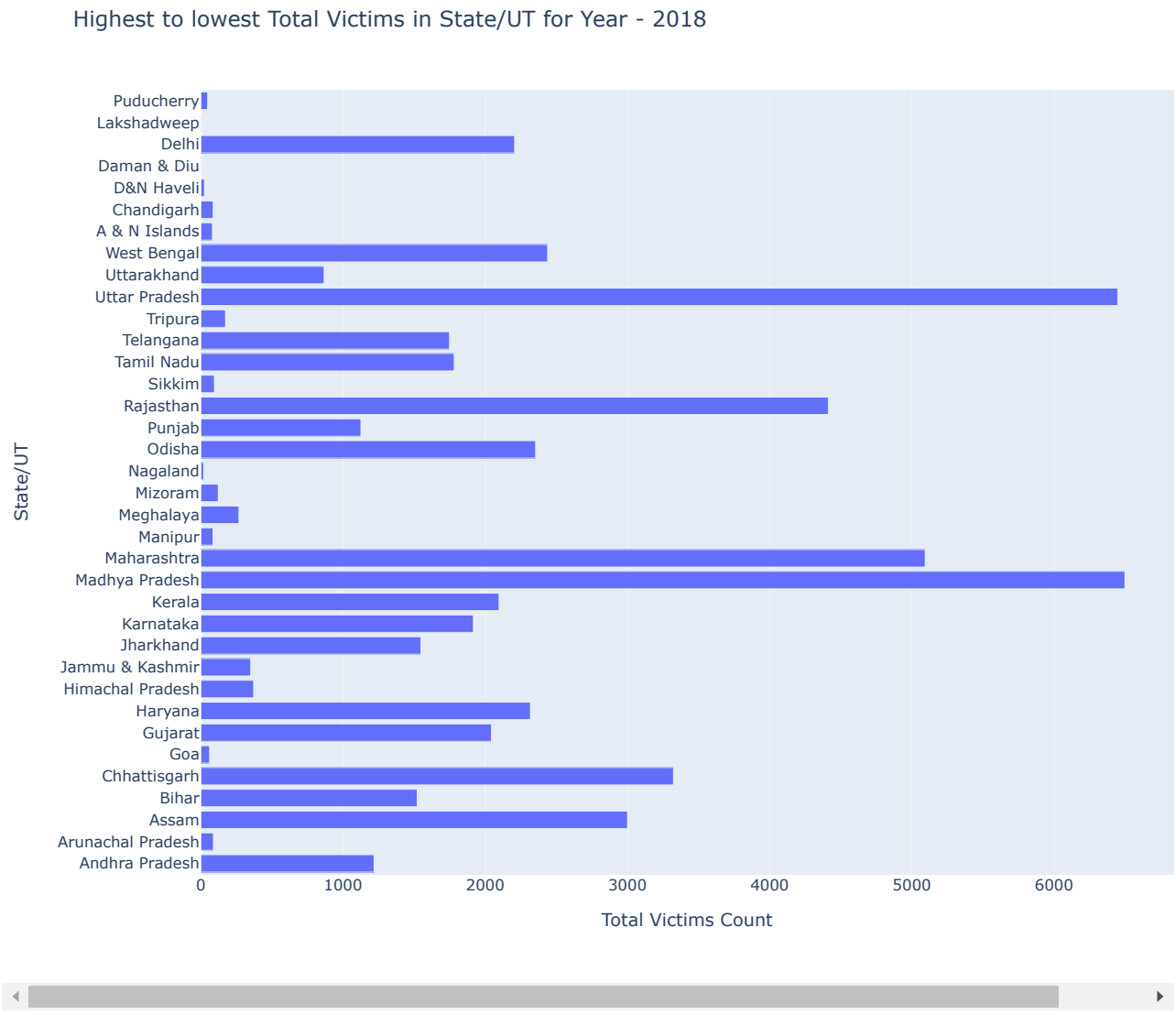
Out[21]:

	State/UT	Victims Below 6 Years - Girls	Victims 6 to 12 years- Girls	Victims 12 - 16 years - Girls	Victims 16 - 18 years - Girls	Total Victims - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Victims above 60 years - Women	Total Victims - Women	Total Victims
0	Andhra Pradesh	28	85	289	348	750	373	76	14	5	468	1218
1	Arunachal Pradesh	5	10	22	13	50	29	9	0	0	38	88
2	Assam	44	278	471	529	1322	1043	523	107	5	1678	3000
3	Bihar	4	67	396	407	874	520	111	16	0	647	1521
4	Chhattisgarh	81	166	1138	1056	2441	644	190	42	6	882	3323

In [22]:
```
D18.info()
```

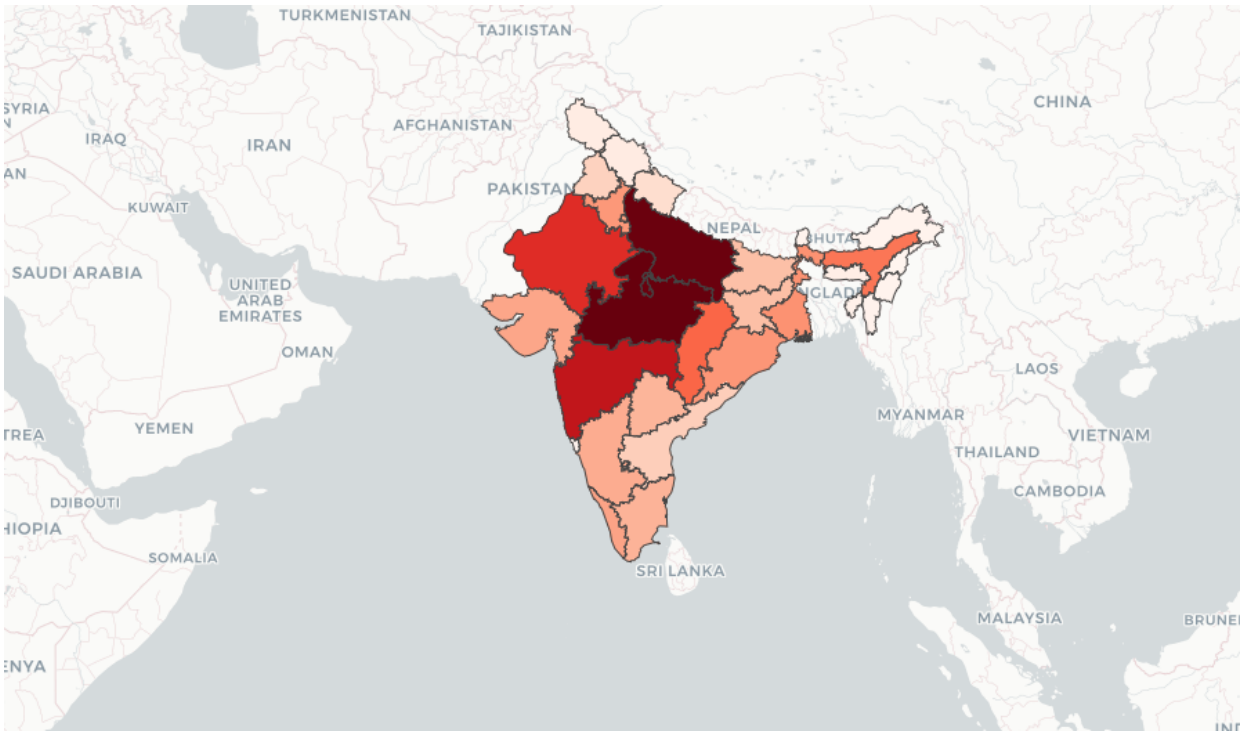
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   State/UT                        36 non-null     object
 1   Victims Below 6 Years - Girls   36 non-null     int64 
 2   Victims 6 to 12 years- Girls    36 non-null     int64 
 3   Victims 12 - 16 years - Girls   36 non-null     int64 
 4   Victims 16 - 18 years - Girls   36 non-null     int64 
 5   Total Victims - Girls           36 non-null     int64 
 6   Victims 18 - 30 years - Women   36 non-null     int64 
 7   Victims 30 - 45  years - Women  36 non-null     int64 
 8   Victims 45 - 60  years - Women  36 non-null     int64 
 9   Victims above 60  years - Women 36 non-null     int64 
 10  Total Victims - Women           36 non-null     int64 
 11  Total Victims                   36 non-null     int64 
dtypes: int64(11), object(1)
memory usage: 3.5+ KB
```

In [23]:
```
# Create the bar chart
fig = px.bar(D18, y='State/UT', x='Total Victims', title='Highest to lowest Total Victims in State/UT for Year - 2018',
             labels={'Total Victims': 'Total Victims Count', 'State/UT': 'State/UT'})

# Increase the size of the graph
fig.update_layout(height=800, width=1000)

# Show the bar chart
fig.show()
```



Geogriphical Heat Map for Total victims of Rape - 2018

In [24]:
```
import plotly.graph_objects as go


fig = go.Figure(go.Choroplethmapbox(
    geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42
    featureidkey='properties.ST_NM',
    locations=D18['State/UT'],
    z=D18['Total Victims'],
    colorscale='Reds',
    colorbar=dict(thickness=15, ticklen=3),
))

fig.update_layout(
    mapbox=dict(
        center=dict(lat=20, lon=77),
        style="carto-positron",
        zoom=3,
    ),
    margin=dict(l=0, r=0, t=0, b=0),
)

fig.show()
```



Accessing Population Data - 2018

In [25]:
```
P18 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population 2018.xlsx")
P18.head(2)
```

Out[25]:

	State/UT	Population
0	Andhra Pradesh	52883163
1	Arunachal Pradesh	1528296

In [26]:
```
# Remove any leading or trailing spaces in column names
P18.columns = P18.columns.str.strip()

# Set the display format for float values
pd.options.display.float_format = '{:.0f}'.format

# Assuming 'Population' is the correct column name
P18['Female Population'] = 0.48 * P18['Population']

# Display the resulting dataset
P18.head(3)
```

Out[26]:

	State/UT	Population	Female Population
0	Andhra Pradesh	52883163	25383918
1	Arunachal Pradesh	1528296	733582
2	Assam	34586234	16601392

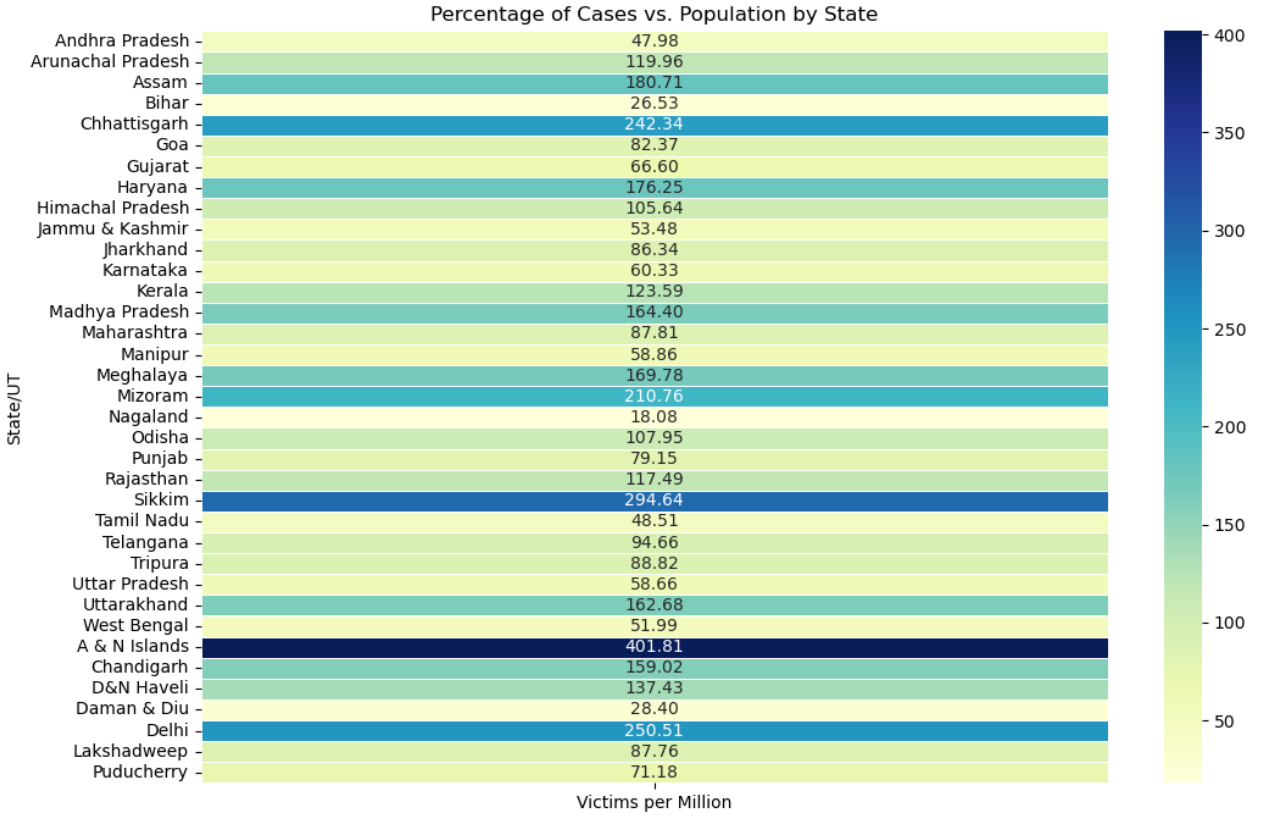
In [27]:
```
# Clean the 'State/UT' column in both DataFrames
P18['State/UT'] = P18['State/UT'].str.strip()
D18['State/UT'] = D18['State/UT'].str.strip()

# Drop the 'Population' column from 'P17'
P18 = P18.drop(columns=['Population'])

# Merge 'Total Victims' column from 'D17' to 'P17'
P18 = pd.merge(P18, D18[['State/UT', 'Total Victims']], on='State/UT', how='left')
```

In [28]:
```
# Create a new column 'Victims per Million'
P18['Victims per Million'] = (P18['Total Victims'] / P18['Female Population']) * 1000000
P18.head(3)
```

Out[28]:

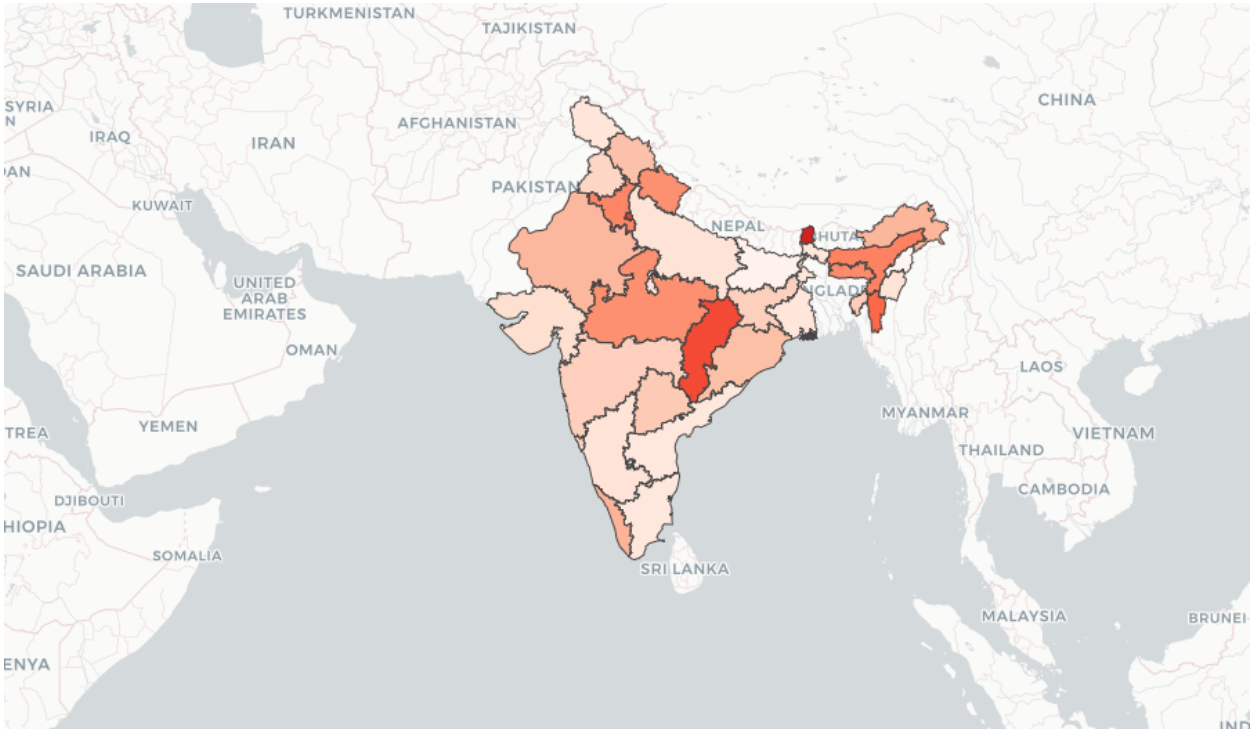
	State/UT	Female Population	Total Victims	Victims per Million
0	Andhra Pradesh	25383918	1218	48
1	Arunachal Pradesh	733582	88	120
2	Assam	16601392	3000	181

In [29]:
```
# Create a new dataset with relevant columns
heatmap_data = P18[['State/UT', 'Victims per Million']]

# Set the 'State/UT' column as the index for the heatmap data
heatmap_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Cases vs. Population by State')
plt.show()
```



Geogriphical Heat Map for Rate of Rape - 2018

In [30]:
```
import plotly.graph_objects as go


fig = go.Figure(go.Choroplethmapbox(
    geojson="https://gist.githubusercontent.com/jbrobst/56c13bbbf9d97d187fea01ca62ea5112/raw/e388c4cae20aa53cb5090210a42
    featureidkey='properties.ST_NM',
    locations=P18['State/UT'],
    z=P18['Victims per Million'],
    colorscale='Reds',
    colorbar=dict(thickness=15, ticklen=3),
))

fig.update_layout(
    mapbox=dict(
        center=dict(lat=20, lon=77),
        style="carto-positron",
        zoom=3,
    ),
    margin=dict(l=0, r=0, t=0, b=0),
)

fig.show()
```

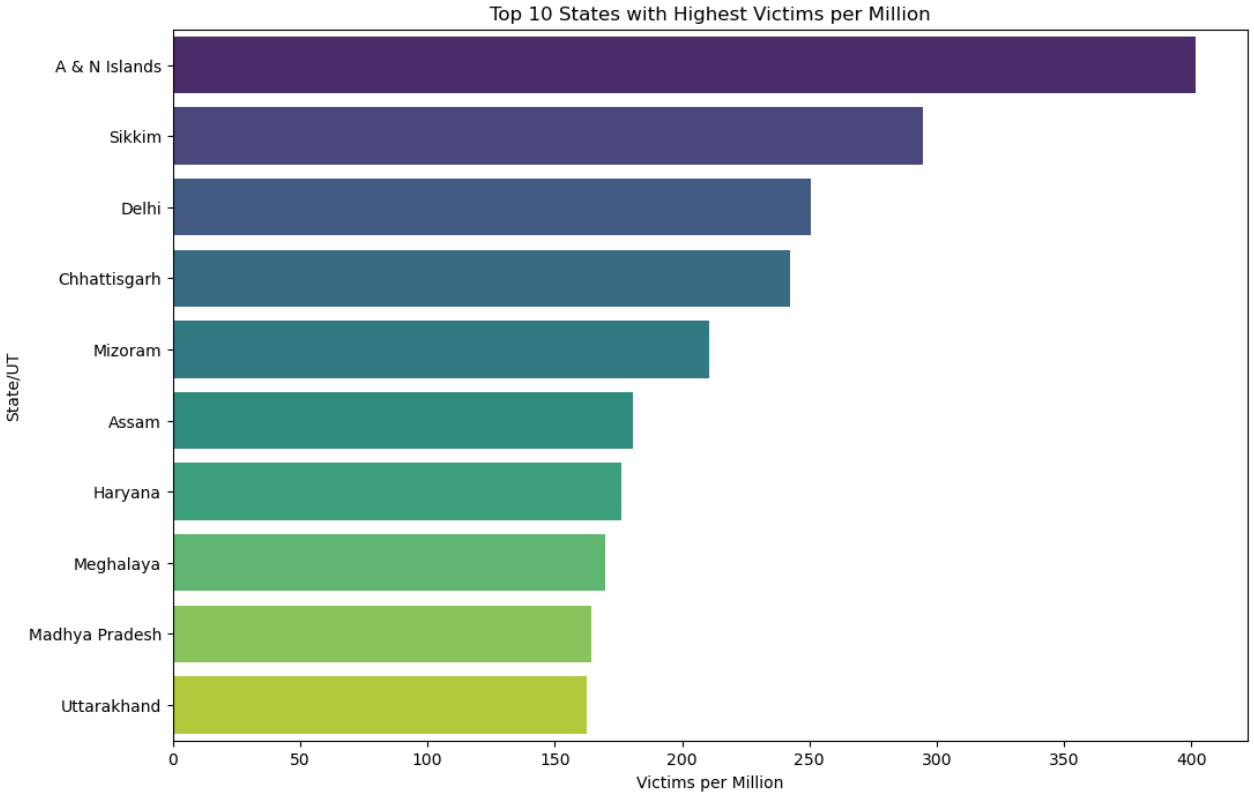


In [31]:
```
# Sort the dataset based on 'Victims per Million' column in descending order
P18_sorted = P18.sort_values(by='Victims per Million', ascending=False)
P18_sorted.head(2)
```

Out[31]:

	State/UT	Female Population	Total Victims	Victims per Million
29	A & N Islands	201589	81	402
22	Sikkim	322426	95	295

In [32]:
```
# Select the top 10 states
top_10_states = P18_sorted.head(10)

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='Victims per Million', y='State/UT', data=top_10_states, palette='viridis')
plt.title('Top 10 States with Highest Victims per Million')
plt.xlabel('Victims per Million')
plt.ylabel('State/UT')
plt.show()
```



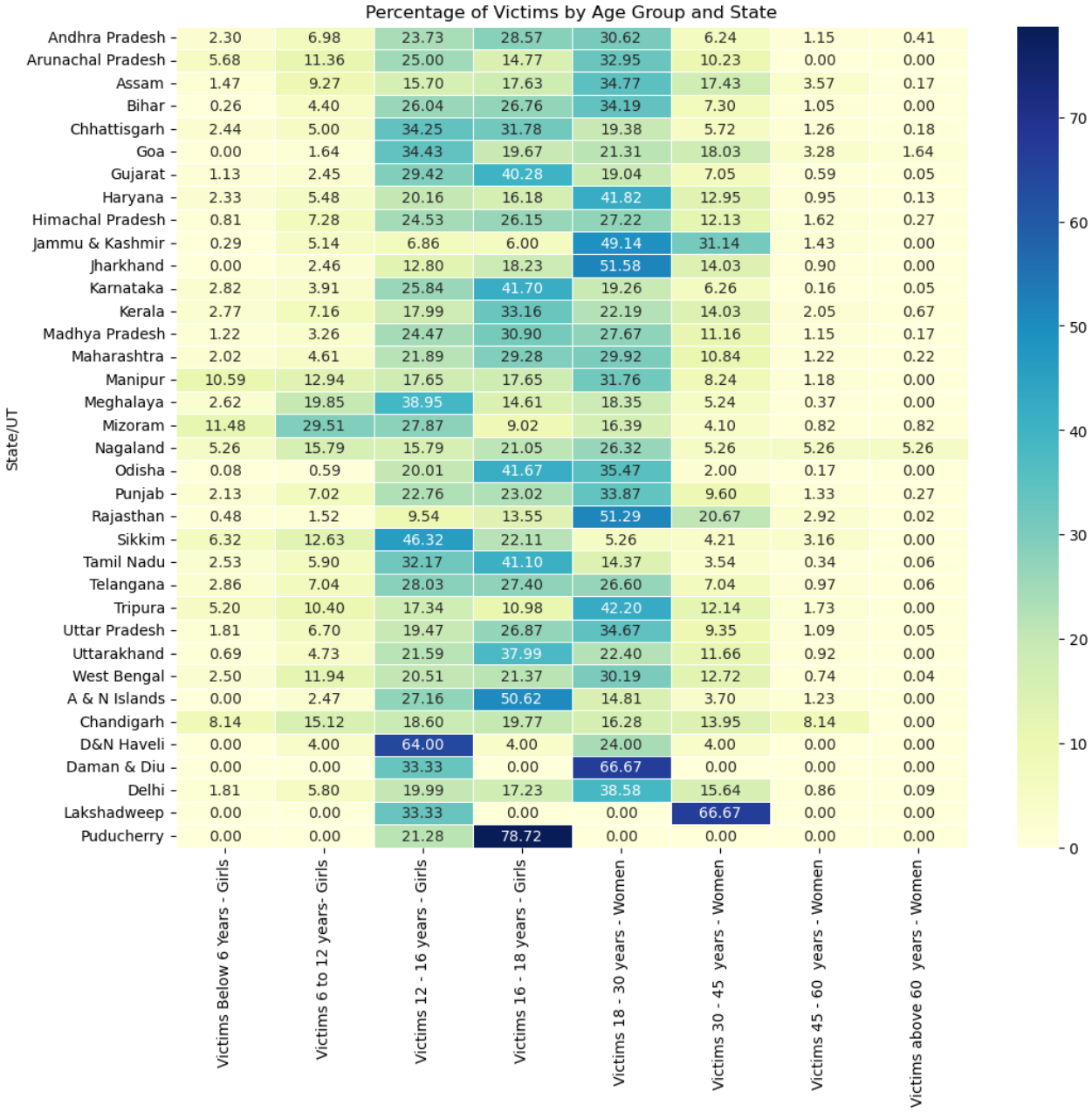
Most targeted age group - 2018

In [33]:
```
# Create a new dataset for percentages
percentage_data = D18.copy()

# Correct column names with spaces
age_groups = ['Victims Below 6 Years - Girls', 'Victims 6 to 12 years- Girls', 'Victims 12 - 16 years - Girls',
              'Victims 16 - 18 years - Girls', 'Total Victims - Girls', 'Victims 18 - 30 years - Women',
              'Victims 30 - 45  years - Women', 'Victims 45 - 60  years - Women', 'Victims above 60  years - Women']

# Loop to calculate percentages for each age group
for age_group in age_groups:
    percentage_data[age_group] = ((D18[age_group] / D18['Total Victims']) * 100).round(2)

# Drop unnecessary columns (if needed)
percentage_data = percentage_data.drop(columns=['Total Victims - Girls', 'Total Victims - Women', 'Total Victims'])

# Display the resulting dataset
percentage_data.head(2)
```

Out[33]:

	State/UT	Victims Below 6 Years - Girls	Victims 6 to 12 years- Girls	Victims 12 - 16 years - Girls	Victims 16 - 18 years - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Victims above 60 years - Women
0	Andhra Pradesh	2	7	24	29	31	6	1	0
1	Arunachal Pradesh	6	11	25	15	33	10	0	0

In [34]:
```
# Set the state column as the index for better visualization
percentage_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(percentage_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Victims by Age Group and State')
plt.show()
```



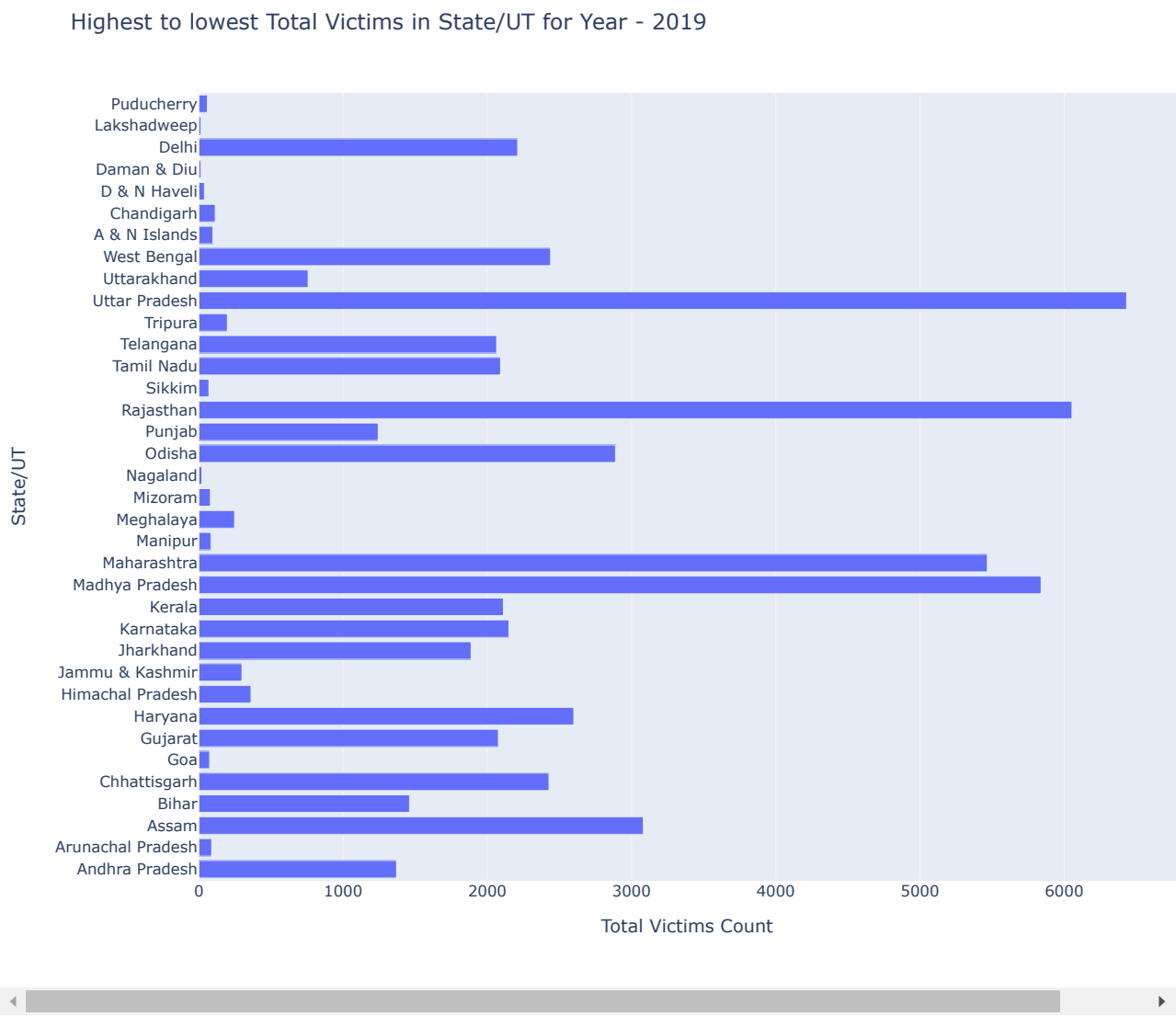
Analysis of Year - 2019

In [35]:
```
D19.head(2)
```

Out[35]:

	State/UT	Victims - below 6 years - Girls	Victims - 6- 12 years - Girls	Victims - 12 - 16 years - Girls	Victims - 16 - 18 years - Girls	Total Victims Girls	Victims 18 - 30 Years - Women	Victims 30 - 45 Years - Women	Victims 45 - 60 Years - Women	Victims above 60 Years - Women	Total Victims - Women	Total Victims
0	Andhra Pradesh	30	65	262	470	827	431	89	21	2	543	1370
1	Arunachal Pradesh	1	6	23	19	49	26	12	1	0	39	88

In [36]:
```
# Create the bar chart
fig = px.bar(D19, y='State/UT', x='Total Victims', title='Highest to lowest Total Victims in State/UT for Year - 2019',
             labels={'Total Victims': 'Total Victims Count', 'State/UT': 'State/UT'})

# Increase the size of the graph
fig.update_layout(height=800, width=1000)

# Show the bar chart
fig.show()
```



Accessing Population Data - 2019

In [37]:
```
P19 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population 2019.xlsx")
P19.head(2)
```

Out[37]:

	State/UT	Population
0	Andhra Pradesh	53390841
1	Arunachal Pradesh	1548776

In [38]:
```
# Remove any leading or trailing spaces in column names
P19.columns = P19.columns.str.strip()

# Set the display format for float values
pd.options.display.float_format = '{:.0f}'.format

# Assuming 'Population' is the correct column name
P19['Female Population'] = 0.48 * P19['Population']
```

Tip: FROM 2019, THE TWO SEPARATE STATES - 'DAMAN AND DIU' AND 'D AND N HVELI' WERE MERGED. THEREFORE THEIR VICTIMS WERE ADDED AND STORED IN A FEW EXCEL FILE, SINCE MERGING WILL NOT WORK

In [39]:
```
#accessing the new dataset with the two states merged

PV19 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population vs total victims
PV19
```

Out[39]:

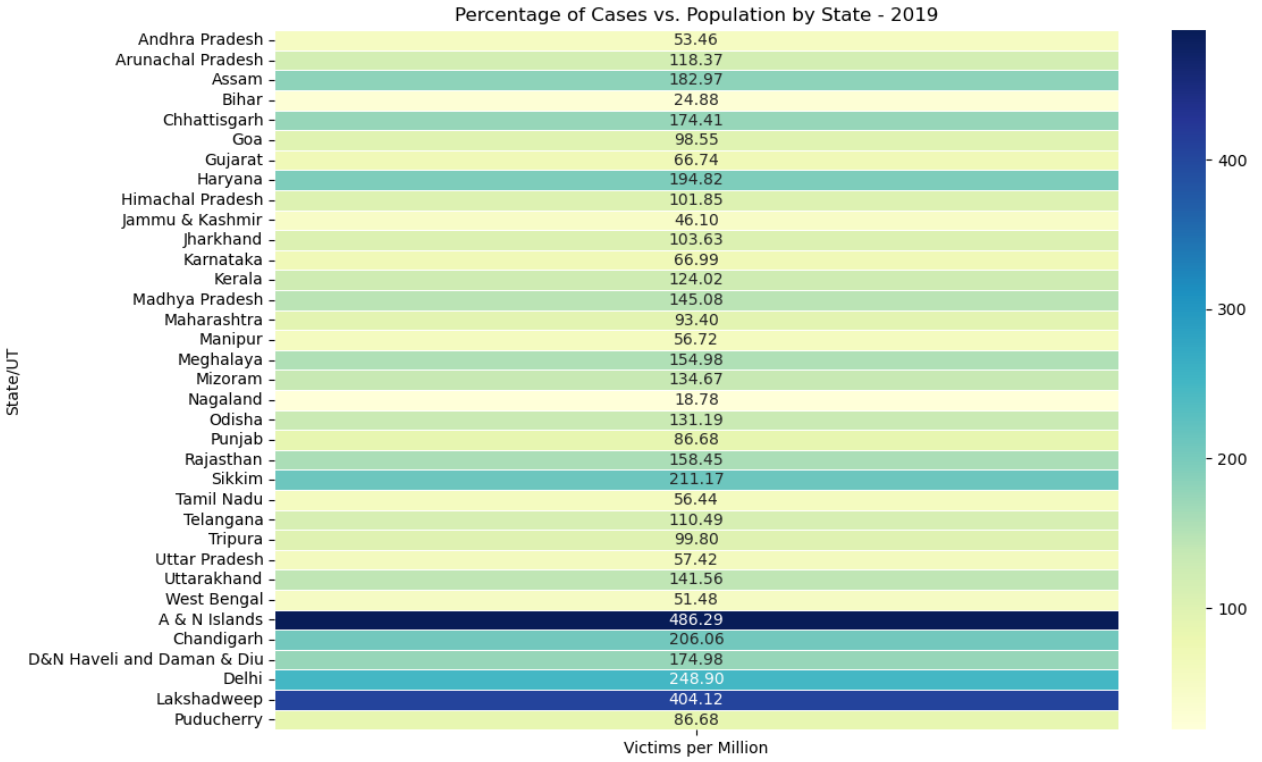
	State/UT	Female Population	Total Victims
0	Andhra Pradesh	25627604	1370
1	Arunachal Pradesh	743412	88
2	Assam	16838797	3081
3	Bihar	58683351	1460
4	Chhattisgarh	13915099	2427
5	Goa	750888	74
6	Gujarat	31104912	2076
7	Haryana	13340808	2599
8	Himachal Pradesh	3544331	361
9	Jammu & Kashmir	6464790	298
10	Jharkhand	18208271	1887
11	Karnataka	32080413	2149
12	Kerala	17021688	2111
13	Madhya Pradesh	40247842	5839
14	Maharashtra	58523987	5466
15	Manipur	1463453	83
16	Meghalaya	1593708	247
17	Mizoram	586624	79
18	Nagaland	1064944	20
19	Odisha	22013297	2888
20	Punjab	14340231	1243
21	Rajasthan	38200442	6053
22	Sikkim	326746	69
23	Tamil Nadu	37045219	2091
24	Telangana	18681146	2064
25	Tripura	1973867	197
26	Uttar Pradesh	112021689	6432
27	Uttarakhand	5347472	757
28	West Bengal	47357830	2438
29	A & N Islands	197413	96
30	Chandigarh	548390	113
31	D&N Haveli and Daman & Diu	291467	51
32	Delhi	8879132	2210
33	Lakshadweep	34643	14
34	Puducherry	669132	58

In [40]:
```
# Create a new column 'Victims per Million'
PV19['Victims per Million'] = (PV19['Total Victims'] / PV19['Female Population']) * 1000000
PV19.head(3)
```

Out[40]:

	State/UT	Female Population	Total Victims	Victims per Million
0	Andhra Pradesh	25627604	1370	53
1	Arunachal Pradesh	743412	88	118
2	Assam	16838797	3081	183

In [41]:
```
# Create a new dataset with relevant columns
heatmap_data = PV19[['State/UT', 'Victims per Million']]

# Set the 'State/UT' column as the index for the heatmap data
heatmap_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Cases vs. Population by State - 2019')
plt.show()
```

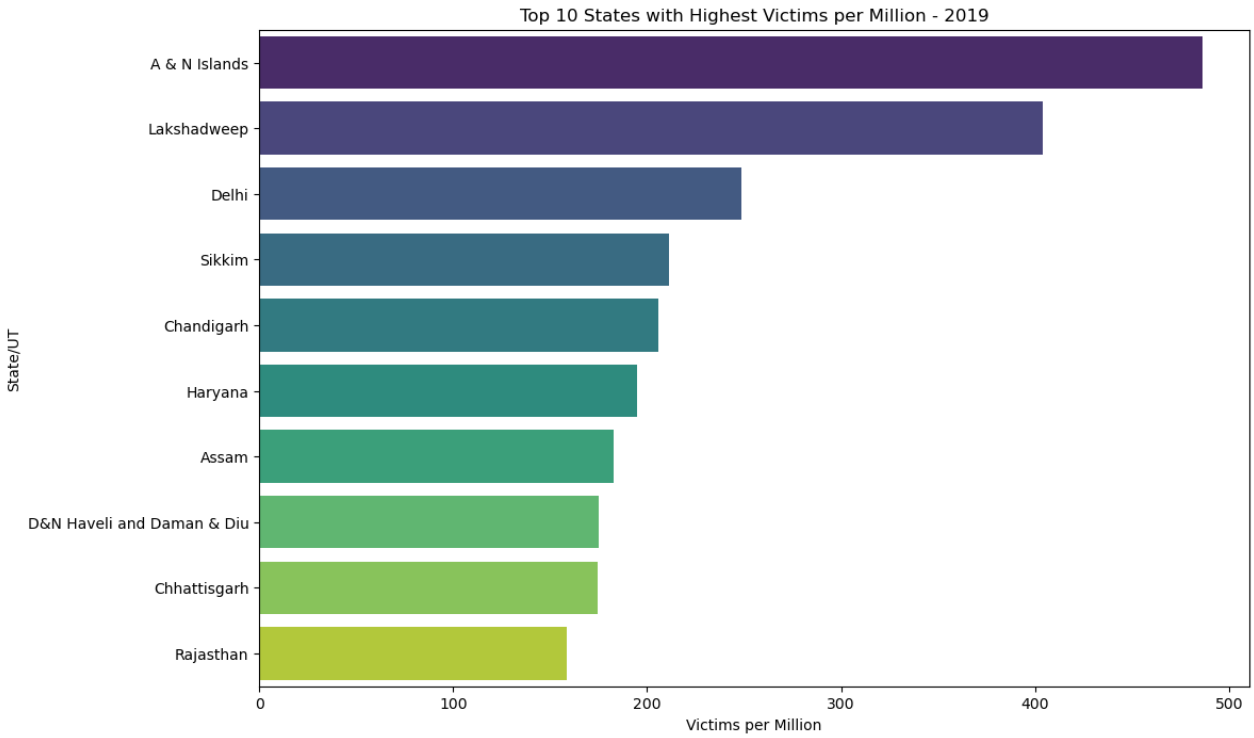


In [42]:
```
# Sort the dataset based on 'Victims per Million' column in descending order
PV19_sorted = PV19.sort_values(by='Victims per Million', ascending=False)
PV19_sorted.head(2)
```

Out[42]:

	State/UT	Female Population	Total Victims	Victims per Million
29	A & N Islands	197413	96	486
33	Lakshadweep	34643	14	404

In [43]:
```
# Select the top 10 states
top_10_states = PV19_sorted.head(10)

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='Victims per Million', y='State/UT', data=top_10_states, palette='viridis')
plt.title('Top 10 States with Highest Victims per Million - 2019')
plt.xlabel('Victims per Million')
plt.ylabel('State/UT')
plt.show()
```



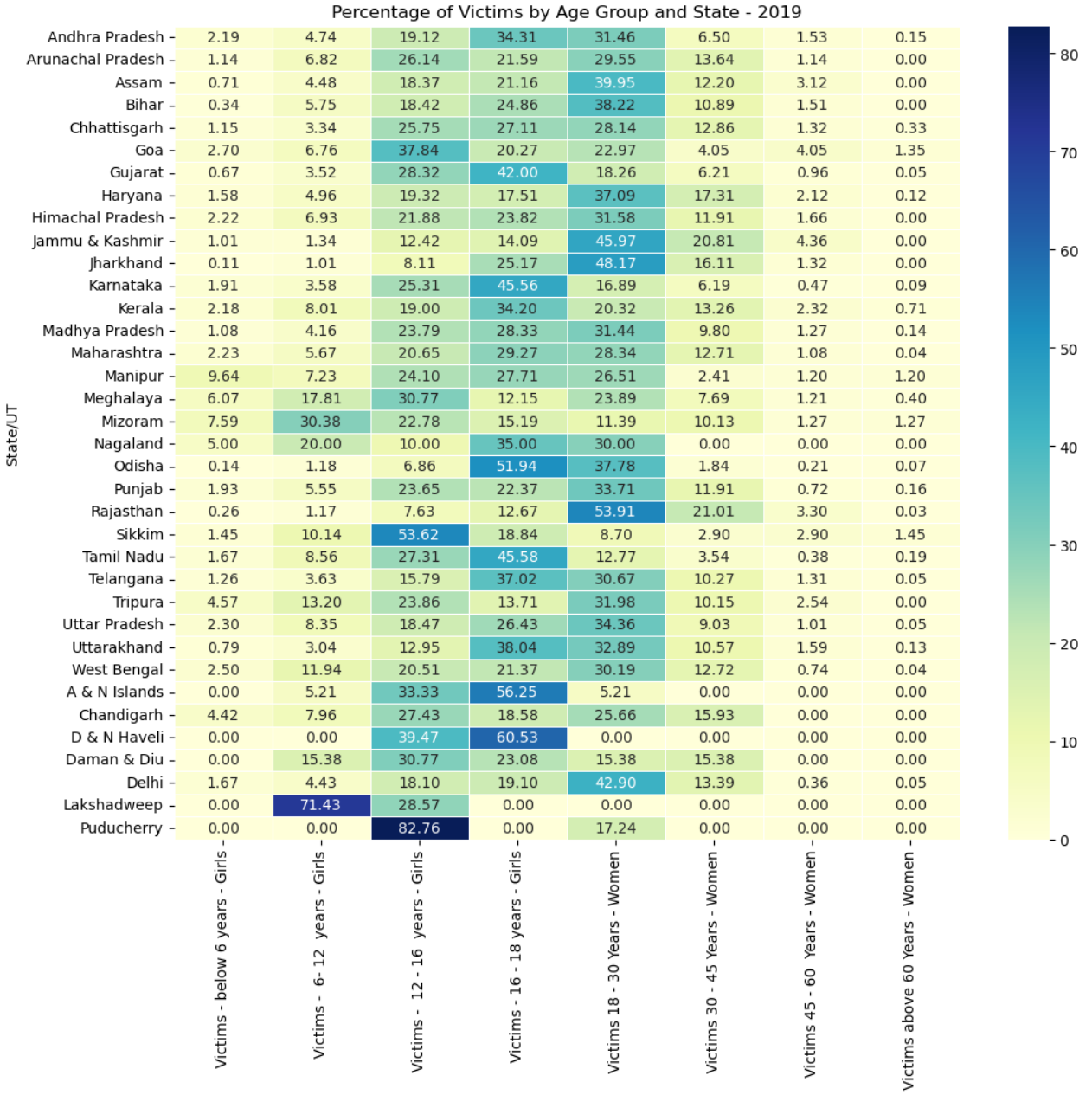
Most targeted age group - 2019

In [44]:
```
# Create a new dataset for percentages
percentage_data = D19.copy()

# Correct column names with spaces
age_groups = ['Victims - below 6 years - Girls', 'Victims -  6- 12  years - Girls',
              'Victims -  12 - 16  years - Girls', 'Victims - 16 - 18 years - Girls',
              'Total Victims Girls', 'Victims 18 - 30 Years - Women',
              'Victims 30 - 45 Years - Women', 'Victims 45 - 60  Years - Women',
              'Victims above 60 Years - Women']

# Loop to calculate percentages for each age group
for age_group in age_groups:
    percentage_data[age_group] = ((D19[age_group] / D19['Total Victims']) * 100).round(2)

# Drop unnecessary columns (if needed)
percentage_data = percentage_data.drop(columns=['Total Victims Girls', 'Total Victims - Women', 'Total Victims'])

# Display the resulting dataset
percentage_data.head(2)
```

Out[44]:

	State/UT	Victims - below 6 years - Girls	Victims - 6- 12 years - Girls	Victims - 12 - 16 years - Girls	Victims - 16 - 18 years - Girls	Victims 18 - 30 Years - Women	Victims 30 - 45 Years - Women	Victims 45 - 60 Years - Women	Victims above 60 Years - Women
0	Andhra Pradesh	2	5	19	34	31	6	2	0
1	Arunachal Pradesh	1	7	26	22	30	14	1	0

In [45]:
```
# Set the state column as the index for better visualization
percentage_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(percentage_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Victims by Age Group and State - 2019')
plt.show()
```



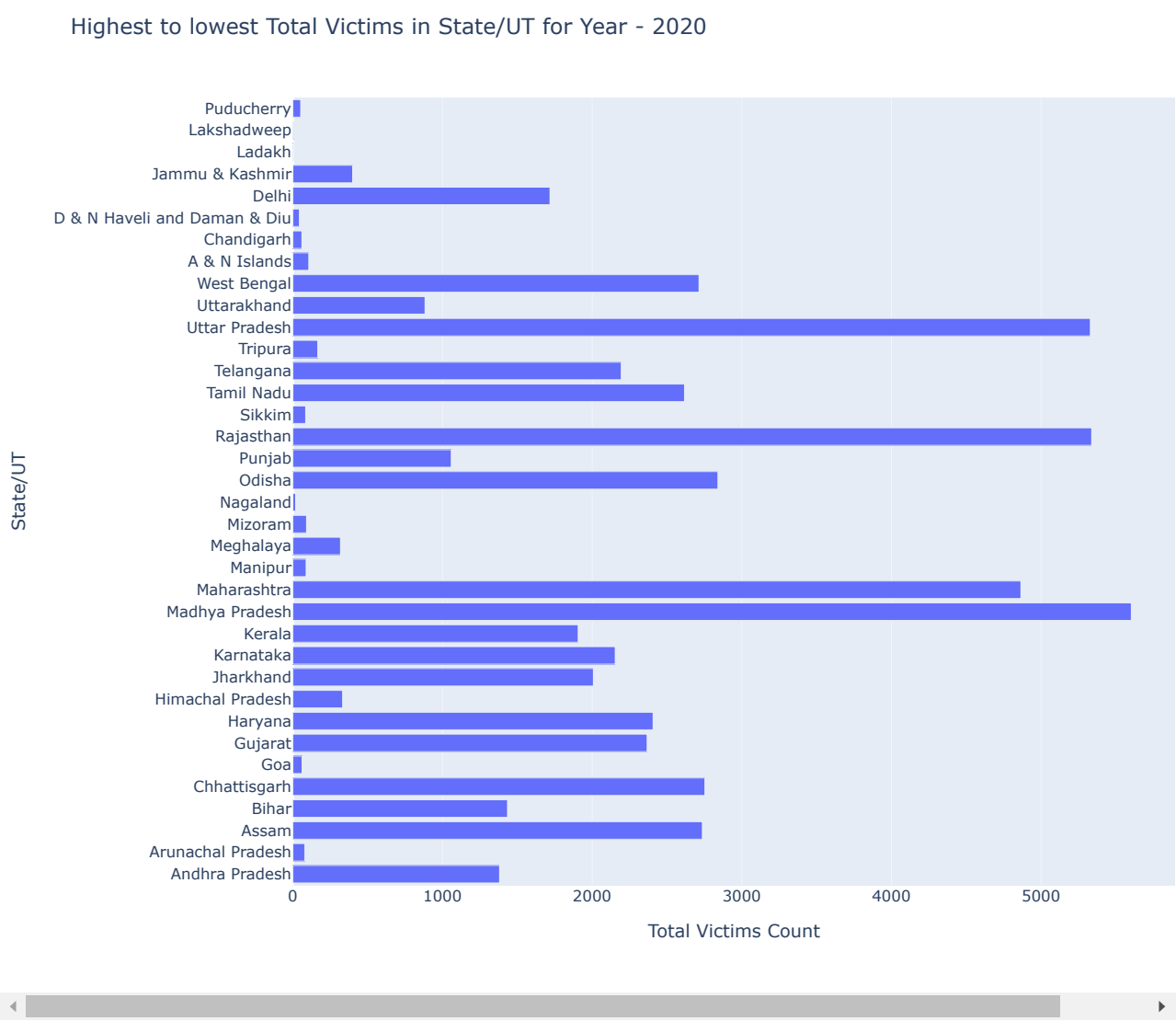
Analysis of Year - 2020

In [46]:
```
D20.head()
```

Out[46]:

	State/UT	Victims below 6 years - Girls	Victims 6 - 12 Years - Girls	Victims 12- 16 years - Girls	Victims 16 - 18 years - Girls	Total Victims - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Above 60 years - Women	Total Victims - Women	Total Victims
0	Andhra Pradesh	30	93	262	476	861	411	87	20	2	520	1381
1	Arunachal Pradesh	1	7	17	20	45	20	14	0	0	34	79
2	Assam	10	91	474	521	1096	1006	584	50	0	1640	2736
3	Bihar	12	59	288	270	629	631	156	18	0	805	1434
4	Chhattisgarh	34	85	578	857	1554	734	398	62	5	1199	2753

In [47]:
```
# Create the bar chart
fig = px.bar(D20, y='State/UT', x='Total Victims', title='Highest to lowest Total Victims in State/UT for Year - 2020',
             labels={'Total Victims': 'Total Victims Count', 'State/UT': 'State/UT'})

# Increase the size of the graph
fig.update_layout(height=800, width=1000)

# Show the bar chart
fig.show()
```



Accessing Population Data - 2020

In [48]:
```
P20 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population 2020.xlsx")
P20.head(3)
```

Out[48]:

	State/UT	Population
0	Andhra Pradesh	53903393
1	Arunachal Pradesh	1570458
2	Assam	35607039

In [49]:
```
# Remove any leading or trailing spaces in column names
P20.columns = P20.columns.str.strip()

# Set the display format for float values
pd.options.display.float_format = '{:.0f}'.format

# Assuming 'Population' is the correct column name
P20['Female Population'] = 0.48 * P20['Population']

# Display the resulting dataset
P20.head(3)
```

Out[49]:

	State/UT	Population	Female Population
0	Andhra Pradesh	53903393	25873629
1	Arunachal Pradesh	1570458	753820
2	Assam	35607039	17091379

In [50]:
```
# Clean the 'State/UT' column in both DataFrames
P20['State/UT'] = P20['State/UT'].str.strip()
D20['State/UT'] = D20['State/UT'].str.strip()

# Drop the 'Population' column from 'P17'
P20 = P20.drop(columns=['Population'])

# Merge 'Total Victims' column from 'D17' to 'P17'
P20 = pd.merge(P20, D20[['State/UT', 'Total Victims']], on='State/UT', how='left')
P20.head(2)
```

Out[50]:

	State/UT	Female Population	Total Victims
0	Andhra Pradesh	25873629	1381
1	Arunachal Pradesh	753820	79

In [51]:
```
#accessing data after merging victims for the two states mentioned above in note

PV20 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population vs total victims
PV20.head(2)
```

Out[51]:

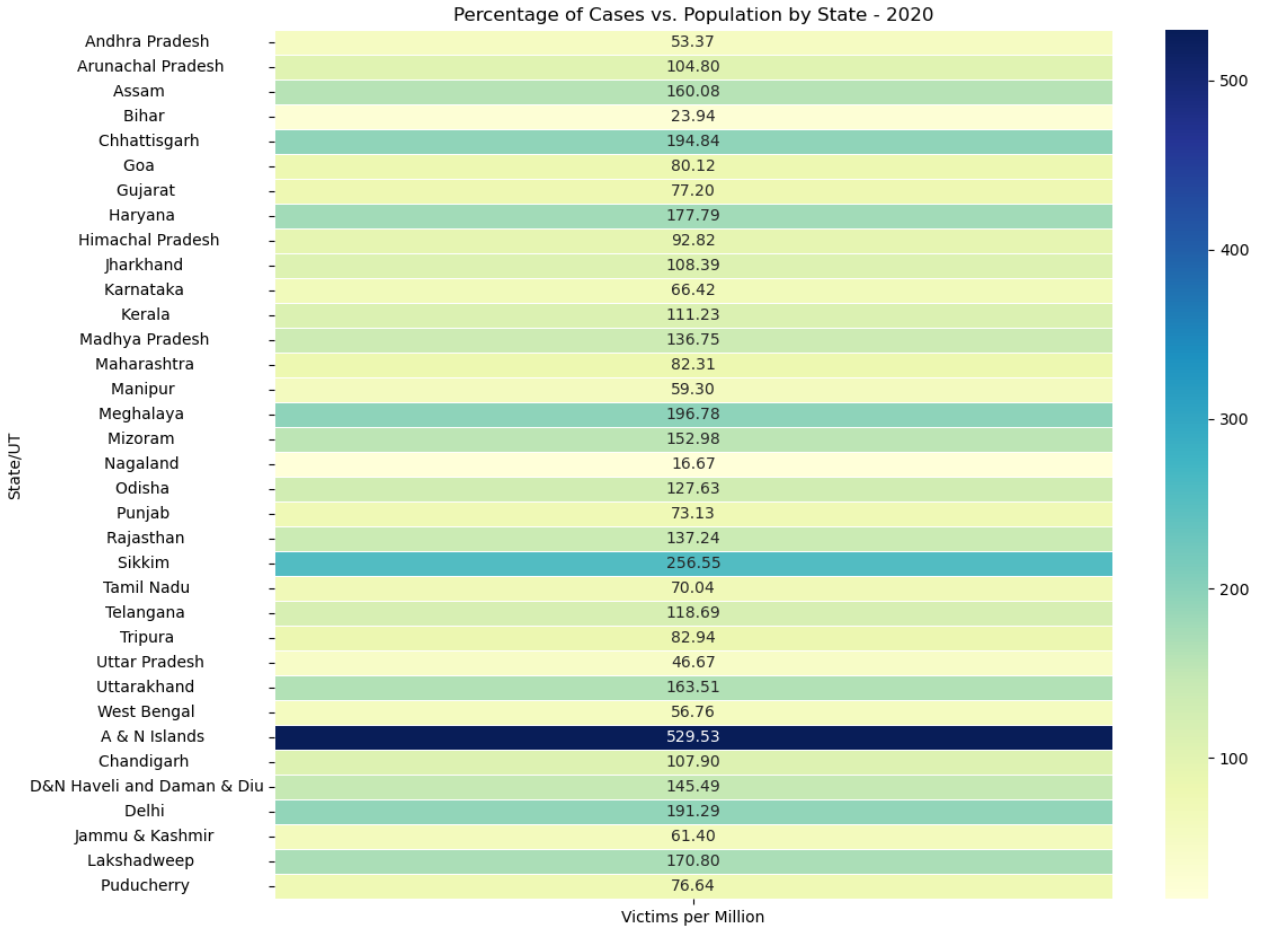
	State/UT	Female Population	Total Victims
0	Andhra Pradesh	25873629	1381
1	Arunachal Pradesh	753820	79

In [52]:
```
# Create a new column 'Victims per Million'
PV20['Victims per Million'] = (PV20['Total Victims'] / PV20['Female Population']) * 1000000
PV20.head(2)
```

Out[52]:

	State/UT	Female Population	Total Victims	Victims per Million
0	Andhra Pradesh	25873629	1381	53
1	Arunachal Pradesh	753820	79	105

In [53]:
```
# Create a new dataset with relevant columns
heatmap_data = PV20[['State/UT', 'Victims per Million']]

# Set the 'State/UT' column as the index for the heatmap data
heatmap_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Cases vs. Population by State - 2020')
plt.show()
```

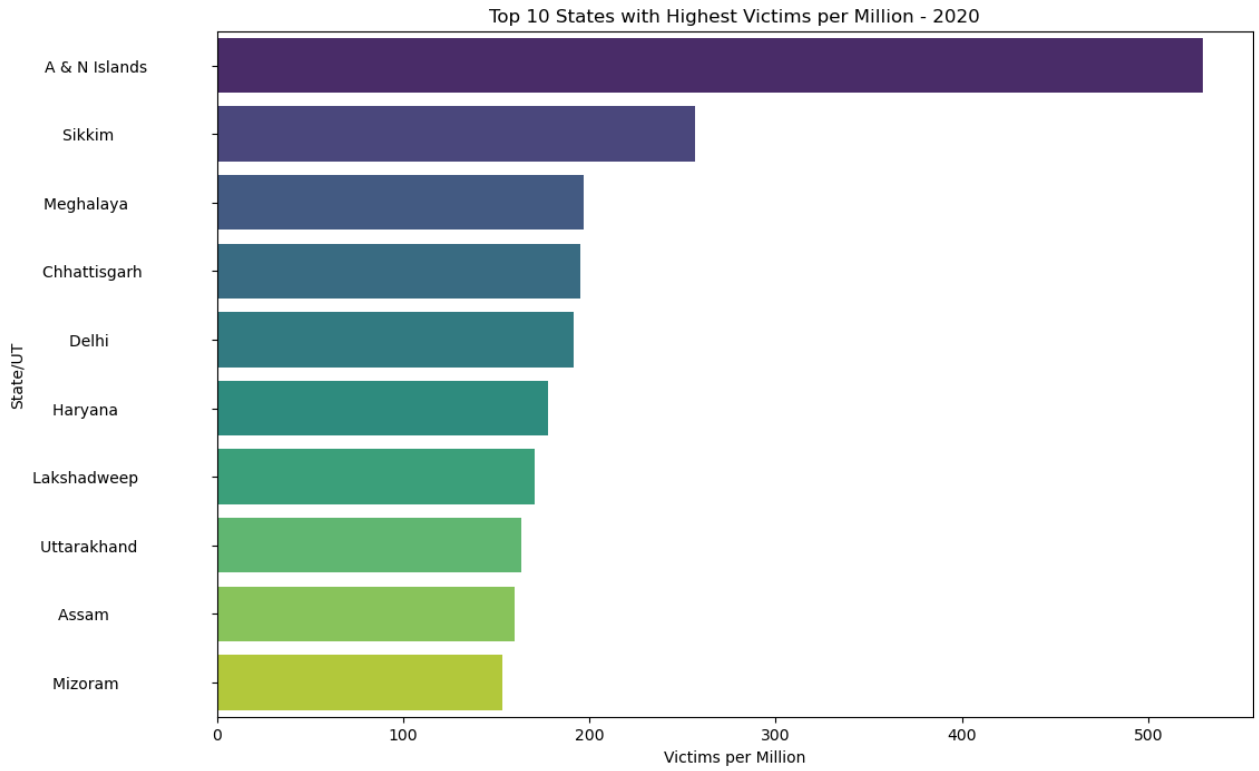


In [54]:
```
# Sort the dataset based on 'Victims per Million' column in descending order
PV20_sorted = PV20.sort_values(by='Victims per Million', ascending=False)
PV20_sorted.head(3)
```

Out[54]:

	State/UT	Female Population	Total Victims	Victims per Million
28	A & N Islands	200177	106	530
21	Sikkim	331320	85	257
15	Meghalaya	1616021	318	197

In [55]:
```
# Select the top 10 states
top_10_states = PV20_sorted.head(10)

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='Victims per Million', y='State/UT', data=top_10_states, palette='viridis')
plt.title('Top 10 States with Highest Victims per Million - 2020')
plt.xlabel('Victims per Million')
plt.ylabel('State/UT')
plt.show()
```



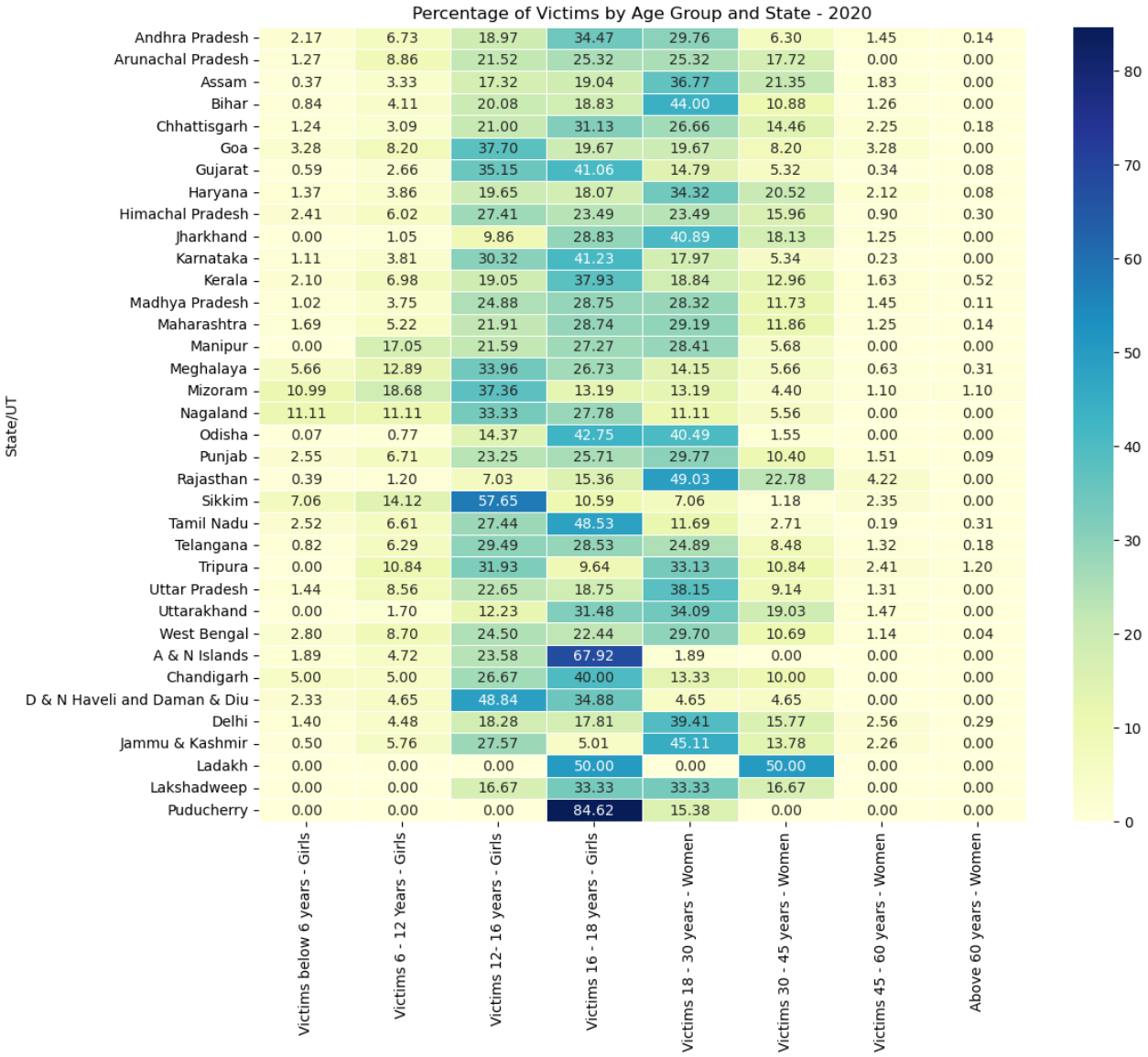
Most targeted age group - 2020

In [56]:
```
# Create a new dataset for percentages
percentage_data = D20.copy()

# Correct column names with spaces
age_groups = ['Victims below 6 years - Girls', 'Victims 6 - 12 Years - Girls', 'Victims 12- 16 years - Girls',
              'Victims 16 - 18 years - Girls', 'Total Victims - Girls', 'Victims 18 - 30 years - Women',
              'Victims 30 - 45 years - Women', 'Victims 45 - 60 years - Women', 'Above 60 years - Women',
              'Total Victims - Women ', 'Total Victims']

# Check if columns exist before performing operations
for age_group in age_groups:
    if age_group in D20.columns and 'Total Victims' in D20.columns:
        percentage_data[age_group] = ((D20[age_group] / D20['Total Victims']) * 100).round(2)

# Drop unnecessary columns (if needed)
# Note: Adjust the column names based on your dataset
columns_to_drop = ['Total Victims - Girls', 'Total Victims - Women ', 'Total Victims']
percentage_data = percentage_data.drop(columns=columns_to_drop, errors='ignore')

# Display the resulting dataset
percentage_data.head(3)
```

Out[56]:

	State/UT	Victims below 6 years - Girls	Victims 6 - 12 Years - Girls	Victims 12- 16 years - Girls	Victims 16 - 18 years - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Above 60 years - Women
0	Andhra Pradesh	2	7	19	34	30	6	1	0
1	Arunachal Pradesh	1	9	22	25	25	18	0	0
2	Assam	0	3	17	19	37	21	2	0

In [57]:
```
# Set the state column as the index for better visualization
percentage_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(percentage_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Victims by Age Group and State - 2020')
plt.show()
```



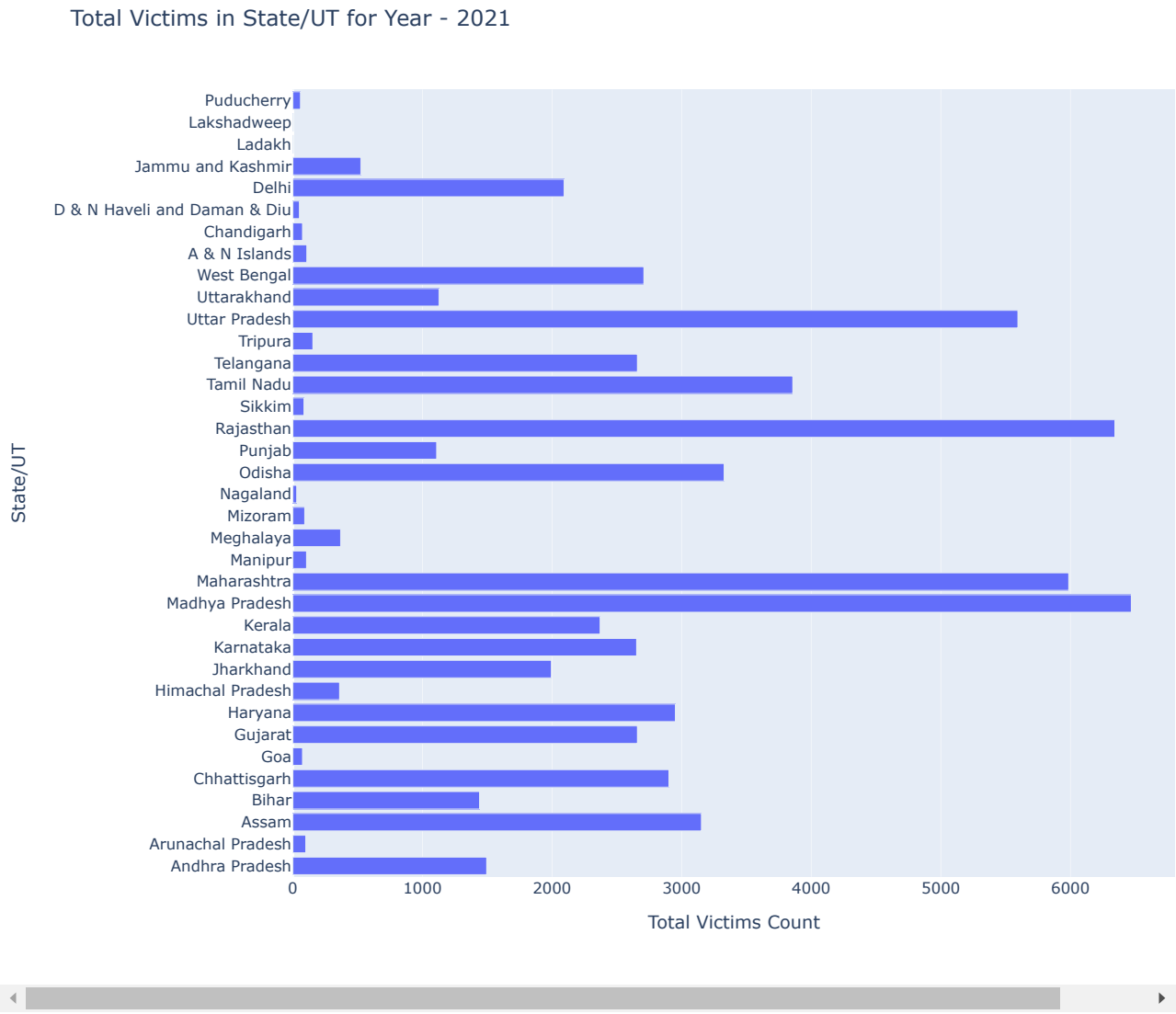
Analysis of Year - 2021

In [58]:
```
D21.head()
```

Out[58]:

	State/UT	Victims - Below 6 years - Girls	Victims 6 - 12 years - Girls	Victims 12 - 16 Years - Girls	Victims 16 - 18 years - Girls	Total Victims - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Victims above 60 years - Women	Total Victims - Women	Total Victims
0	Andhra Pradesh	24	78	390	413	905	463	108	14	5	590	1495
1	Arunachal Pradesh	4	9	22	35	70	13	5	11	0	29	99
2	Assam	33	323	471	493	1320	1306	470	56	0	1832	3152
3	Bihar	2	106	301	245	654	592	176	18	0	786	1440
4	Chhattisgarh	19	82	674	1033	1808	769	278	40	6	1093	2901

In [59]:
```
#Lets visualize the above state list

# D21 is the DataFrame with 'State/UT' and 'Total Victims' columns

fig = px.bar(D21, y='State/UT', x='Total Victims', title='Total Victims in State/UT for Year - 2021',
             labels={'Total Victims': 'Total Victims Count', 'State/UT': 'State/UT'})

# Set the figure size
fig.update_layout(width=1000, height=800)

# Show the bar chart
fig.show()
```



Accessing Population Data

In [60]:
```
P21 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population 2021.xlsx")
P21.head(3)
```

Out[60]:

	State/UT	Population
0	Andhra Pradesh	52972000
1	Arunachal Pradesh	1548000
2	Assam	35378000

In [61]:
```
# Remove any leading or trailing spaces in column names
P21.columns = P21.columns.str.strip()

# Set the display format for float values
pd.options.display.float_format = '{:.0f}'.format

# Assuming 'Population' is the correct column name
P21['Female Population'] = 0.48 * P21['Population']

# Display the resulting dataset
P21.head(3)
```

Out[61]:

	State/UT	Population	Female Population
0	Andhra Pradesh	52972000	25426560
1	Arunachal Pradesh	1548000	743040
2	Assam	35378000	16981440

In [62]:
```
#accessing data after merging victims for the two states mentioned above in note

PV21 = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\Population Data\\population vs total victims
PV21.head(3)
```

Out[62]:

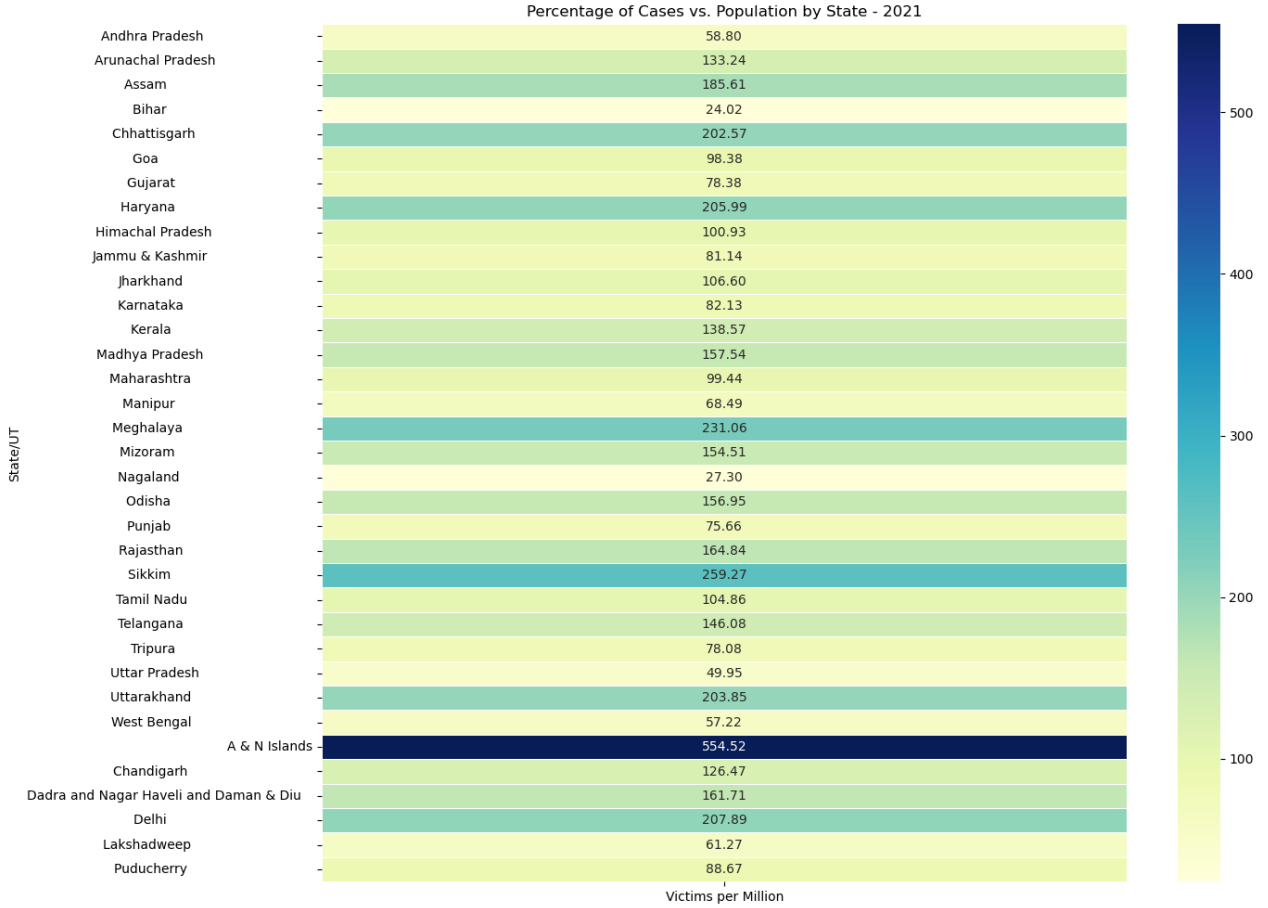
	State/UT	Female Population	Total Victims
0	Andhra Pradesh	25426560	1495
1	Arunachal Pradesh	743040	99
2	Assam	16981440	3152

In [63]:
```
# Create a new column 'Victims per Million'
PV21['Victims per Million'] = (PV21['Total Victims'] / PV21['Female Population']) * 1000000
PV21.head(3)
```

Out[63]:

	State/UT	Female Population	Total Victims	Victims per Million
0	Andhra Pradesh	25426560	1495	59
1	Arunachal Pradesh	743040	99	133
2	Assam	16981440	3152	186

In [64]:
```
# Create a new dataset with relevant columns
heatmap_data = PV21[['State/UT', 'Victims per Million']]

# Set the 'State/UT' column as the index for the heatmap data
heatmap_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(14, 12))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Cases vs. Population by State - 2021')
plt.show()
```

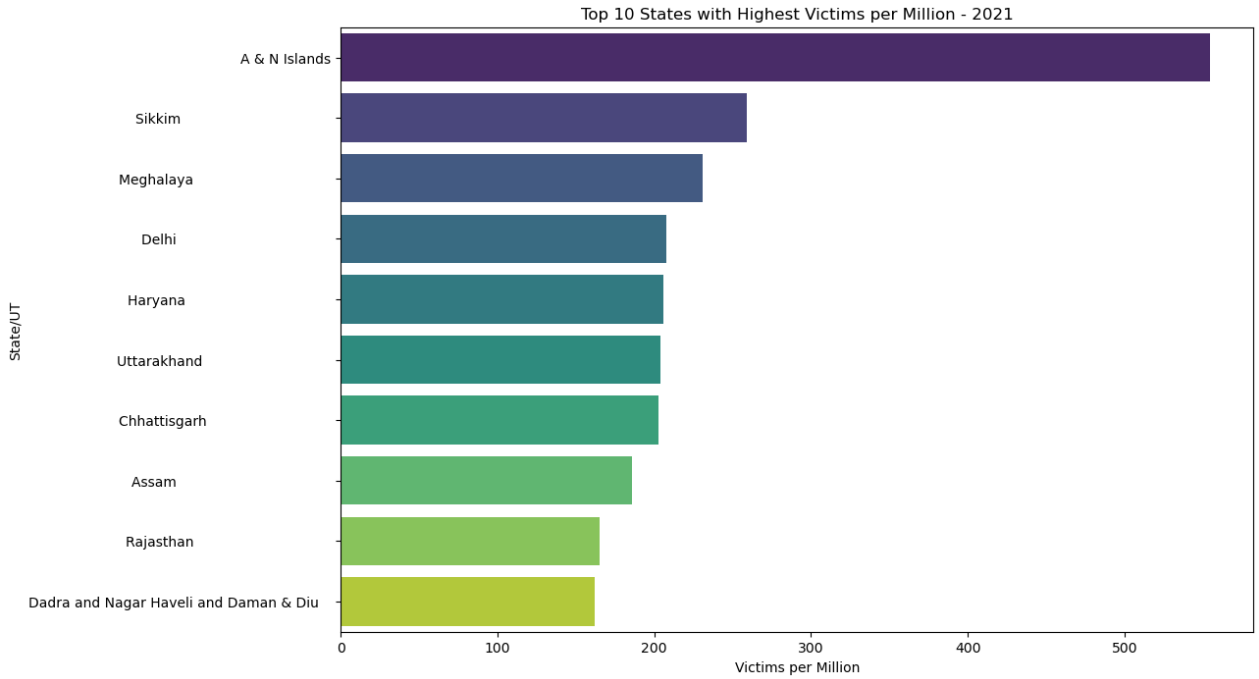


In [65]:
```
# Sort the dataset based on 'Victims per Million' column in descending order
PV21_sorted = PV21.sort_values(by='Victims per Million', ascending=False)
PV21_sorted.head(3)
```

Out[65]:

	State/UT	Female Population	Total Victims	Victims per Million
29	A & N Islands	192960	107	555
22	Sikkim	327840	85	259
16	Meghalaya	1592640	368	231

In [66]:
```
# Select the top 10 states
top_10_states = PV21_sorted.head(10)

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x='Victims per Million', y='State/UT', data=top_10_states, palette='viridis')
plt.title('Top 10 States with Highest Victims per Million - 2021')
plt.xlabel('Victims per Million')
plt.ylabel('State/UT')
plt.show()
```



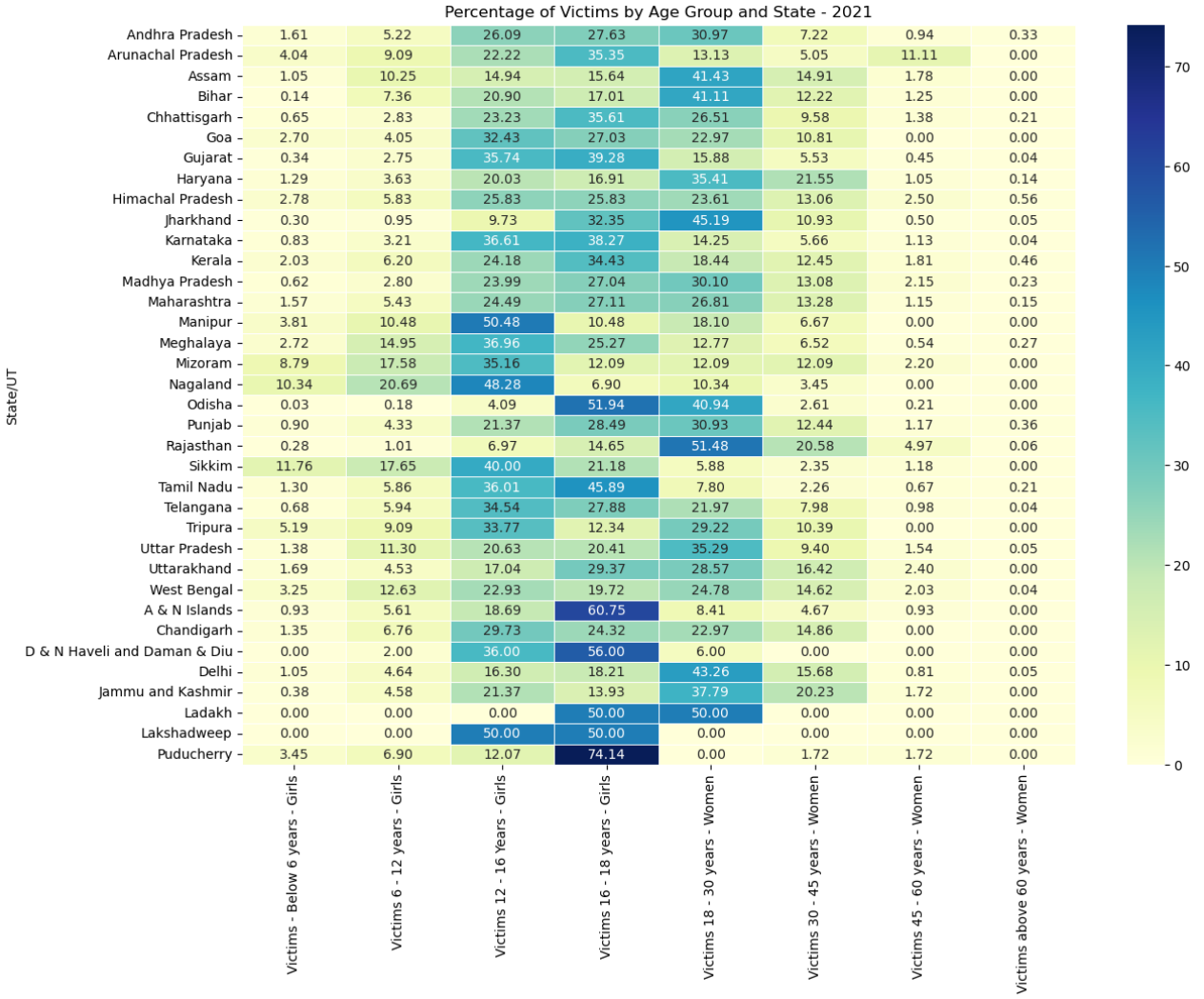
Most targeted age group - 2021

In [67]:
```
# Create a new dataset for percentages
percentage_data = D21.copy()

# Correct column names with spaces
age_groups = [' Victims - Below 6 years - Girls', ' Victims 6 - 12 years - Girls',
              'Victims 12 - 16 Years - Girls', 'Victims 16 - 18 years - Girls',
              'Total Victims - Girls ', 'Victims 18 - 30 years - Women',
              'Victims 30 - 45 years - Women', 'Victims 45 - 60 years - Women',
              'Victims above 60 years - Women', 'Total Victims - Women',
              'Total Victims']

# Check if columns exist before performing operations
for age_group in age_groups:
    if age_group in D21.columns and 'Total Victims' in D21.columns:
        percentage_data[age_group] = ((D21[age_group] / D21['Total Victims']) * 100).round(2)

# Drop unnecessary columns (if needed)
# Note: Adjust the column names based on your dataset
columns_to_drop = ['Total Victims - Girls ', 'Total Victims - Women', 'Total Victims']
percentage_data = percentage_data.drop(columns=columns_to_drop, errors='ignore')

# Display the resulting dataset
percentage_data.head(3)
```

Out[67]:

	State/UT	Victims - Below 6 years - Girls	Victims 6 - 12 years - Girls	Victims 12 - 16 Years - Girls	Victims 16 - 18 years - Girls	Victims 18 - 30 years - Women	Victims 30 - 45 years - Women	Victims 45 - 60 years - Women	Victims above 60 years - Women
0	Andhra Pradesh	2	5	26	28	31	7	1	0
1	Arunachal Pradesh	4	9	22	35	13	5	11	0
2	Assam	1	10	15	16	41	15	2	0

In [68]:
```
# Set the state column as the index for better visualization
percentage_data.set_index('State/UT', inplace=True)

# Create a heatmap
plt.figure(figsize=(14, 10))
sns.heatmap(percentage_data, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=.5)
plt.title('Percentage of Victims by Age Group and State - 2021')
plt.show()
```



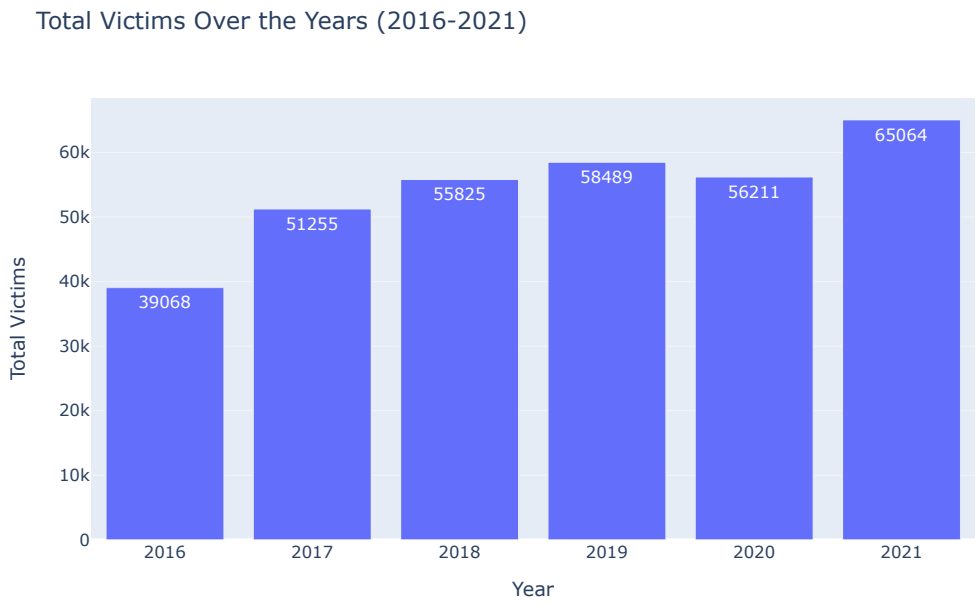
Overall Analysis for all Years

In [69]:
```
#data of total victims of India for all years


Total = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\Overall a
Total
```

Out[69]:

	Year	Total Victims
0	2016	39068
1	2017	51255
2	2018	55825
3	2019	58489
4	2020	56211
5	2021	65064

In [70]:
```
import plotly.express as px

# Assuming 'Year' is a column in your DataFrame representing the years
# Replace 'Total' with the actual name of your DataFrame if it's different
years = Total['Year']

# Assuming 'Total Victims' is the column you want to plot
total_victims = Total['Total Victims']

# Creating an interactive bar graph with Plotly
fig = px.bar(x=years, y=total_victims, text=total_victims,
             labels={'x': 'Year', 'y': 'Total Victims'},
             title='Total Victims Over the Years (2016-2021)',
             height=500)

# Adjusting the size of the plot
fig.update_layout(width=800)

# Show the interactive plot
fig.show()
```



Let's explore the Top and Bottom states for various measures

Top 9 States - Total Victims (2016 - 2021)

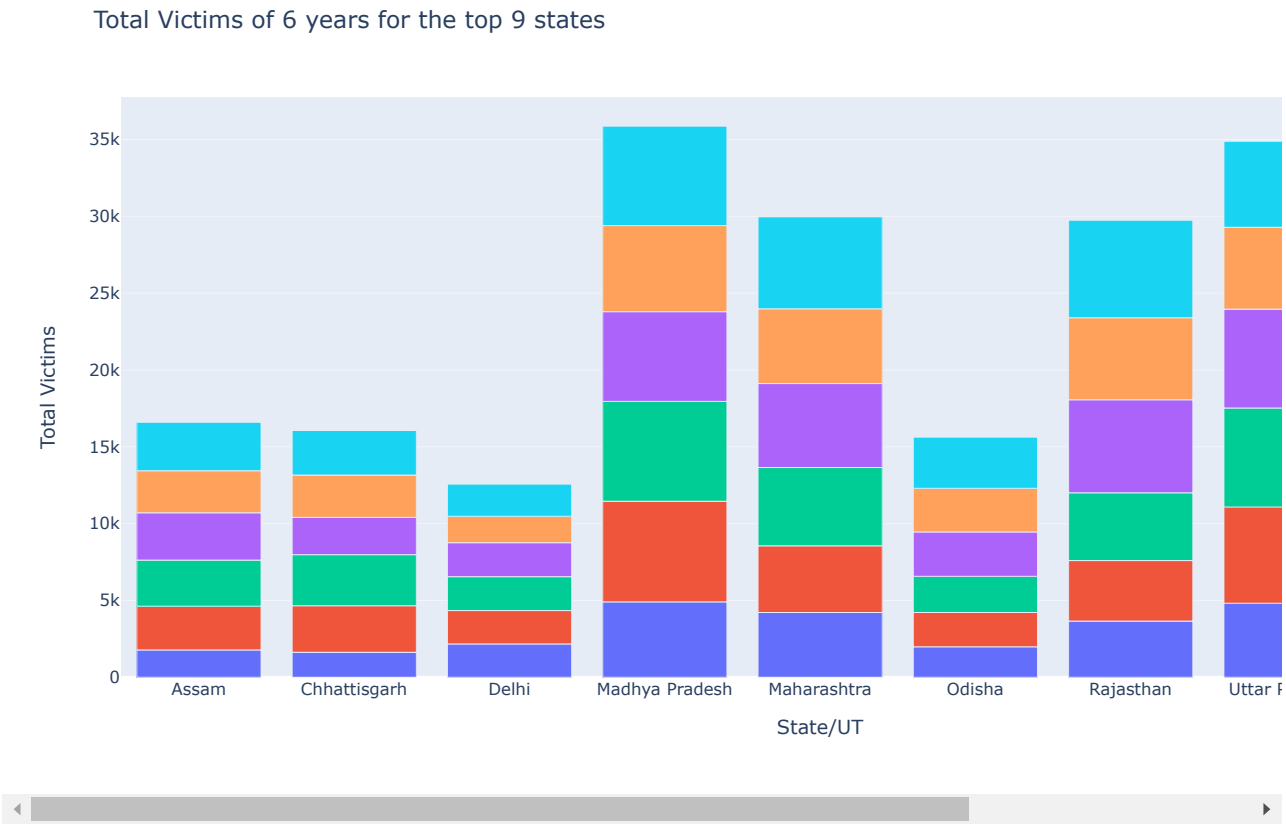
In [71]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Datasets are named D16, D17, D18, D19, D20, D21
datasets = [D16, D17, D18, D19, D20, D21]
states_to_compare = ['Madhya Pradesh', 'Rajasthan', 'Uttar Pradesh', 'Chhattisgarh', 'Maharashtra', 'Assam', 'Odisha', '

# Create an empty DataFrame to store the count of total victims for each state and year
comparison_data = pd.DataFrame(columns=['Year', 'State/UT', 'Total Victims'])

# Loop through each dataset and extract the count for the specified states
for year, dataset in zip(range(2016, 2022), datasets):
    total_victims_by_state = dataset.groupby('State/UT')['Total Victims'].sum().reset_index()
    total_victims_by_state = total_victims_by_state[total_victims_by_state['State/UT'].isin(states_to_compare)]

    # Add the year column
    total_victims_by_state['Year'] = year

    # Append the data to the comparison DataFrame
    comparison_data = pd.concat([comparison_data, total_victims_by_state], ignore_index=True)

# Create traces for each state
traces = []
for state in states_to_compare:
    trace = go.Scatter(
        x=comparison_data[comparison_data['State/UT'] == state]['Year'],
        y=comparison_data[comparison_data['State/UT'] == state]['Total Victims'],
        mode='lines',
        name=state
    )
    traces.append(trace)

# Create a subplot with a dropdown menu
fig = make_subplots(rows=1, cols=1, specs=[[{'type': 'scatter'}]])
dropdown_buttons = [
    {
        'method': 'update',
        'label': 'Show All',
        'args': [{'visible': [True] * len(traces)}]
    }
] + [
    {
        'method': 'update',
        'label': state,
        'args': [{'visible': [state == trace.name for trace in traces]}]
    }
    for state in states_to_compare
]
fig.update_layout(
    updatemenus=[{
        'active': 0,
        'buttons': dropdown_buttons,
        'showactive': True,
    }],
    height=600, width=1000,
    title='Comparison of Total Victims for Selected States (2016-2021)',
    xaxis_title='Year',
    yaxis_title='Total Victims'
)

# Add traces to the subplot
for trace in traces:
    fig.add_trace(trace)

# Show the interactive plot
fig.show()
```


Comparison of Total Victims for Selected States (2016-2021)
Show All
Total Victims
6000
5000
4000
3000
2000
1000
2016
2017
2018
2019
2020
2021
Year
Madhy
Rajast
Uttar
Chhat
Mahar
Assam
Odisha
Delhi
West

Count of Total Victims for all 6 Years

In [72]:

```
# Datasets are named D16, D17, D18, D19, D20, D21
datasets = [D16, D17, D18, D19, D20, D21]
states_to_compare = ['Madhya Pradesh', 'Rajasthan', 'Uttar Pradesh', 'Chhattisgarh', 'Maharashtra', 'Assam', 'Odisha', '

# Create an empty DataFrame to store the total count of victims for each state
total_victims_data = pd.DataFrame(columns=['State/UT', 'Total Victims', 'Year'])

# Loop through each dataset and extract the count for the specified states
for i, dataset in enumerate(datasets):
    total_victims_by_state = dataset.groupby('State/UT')['Total Victims'].sum().reset_index()
    total_victims_by_state = total_victims_by_state[total_victims_by_state['State/UT'].isin(states_to_compare)]

    # Add the year information to the DataFrame
    total_victims_by_state['Year'] = f'20{i + 16}'

    # Append the data to the total victims DataFrame
    total_victims_data = pd.concat([total_victims_data, total_victims_by_state], ignore_index=True)

# Create a bar chart for total victims across all years
fig = px.bar(total_victims_data, x='State/UT', y='Total Victims', color='Year',
             labels={'Total Victims': 'Total Victims'},
             title='Total Victims of 6 years for the top 9 states',
             height=600, width=1200,
             hover_data={'Year': True, 'Total Victims': ':.0f'})  # Specify the hover_data

# Customize the layout as needed
fig.update_layout(xaxis_title='State/UT', yaxis_title='Total Victims')

# Show the interactive plot
fig.show()
```

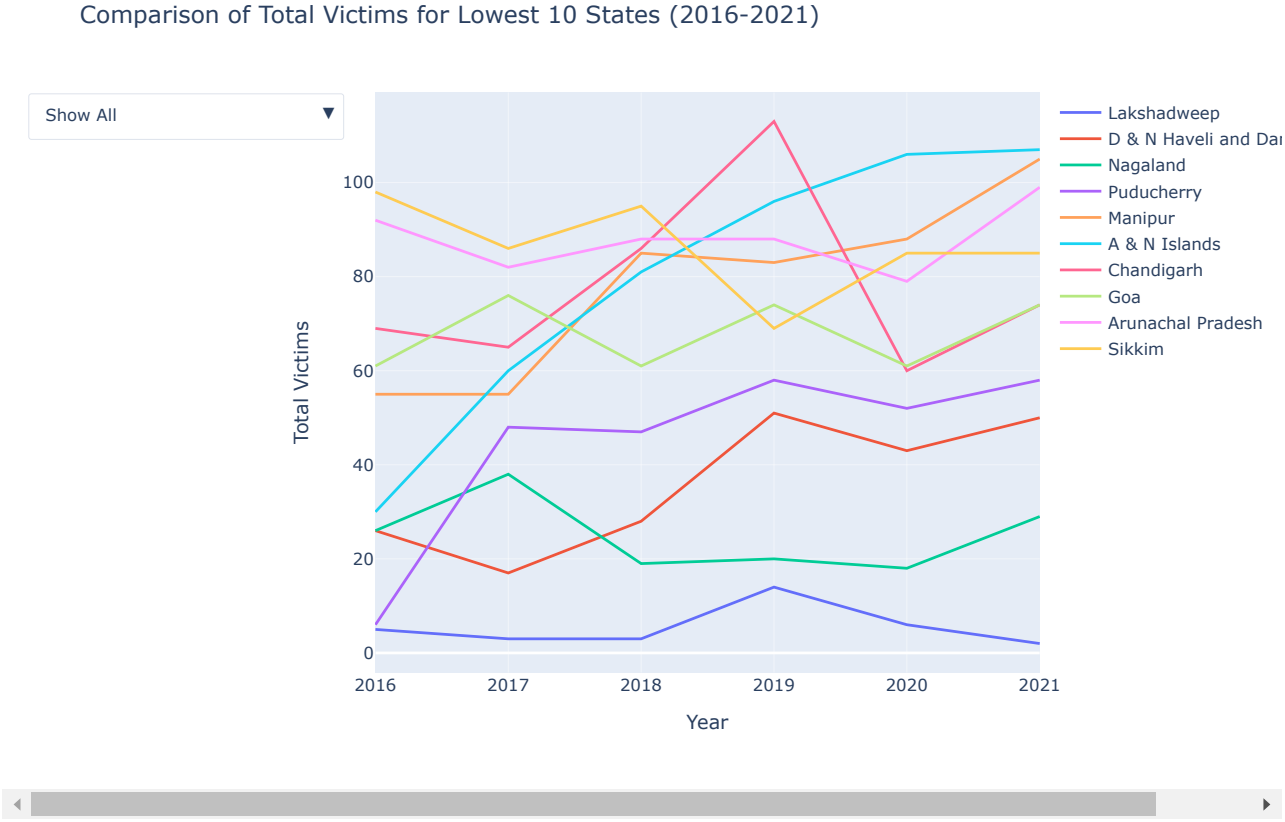


Extracting the datasets from 2016 to 2019 , where D&N Haveli and Daman & Diu has been added, so that we can explore the lowest count states

In [73]:
```
# Read the datasets

D16c = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\DAMAN comb
D17c = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\DAMAN comb
D18c = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\DAMAN comb
D19c = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\VICTIMS OF RAPE - GIRLS AND WOMEN\\DAMAN comb
```

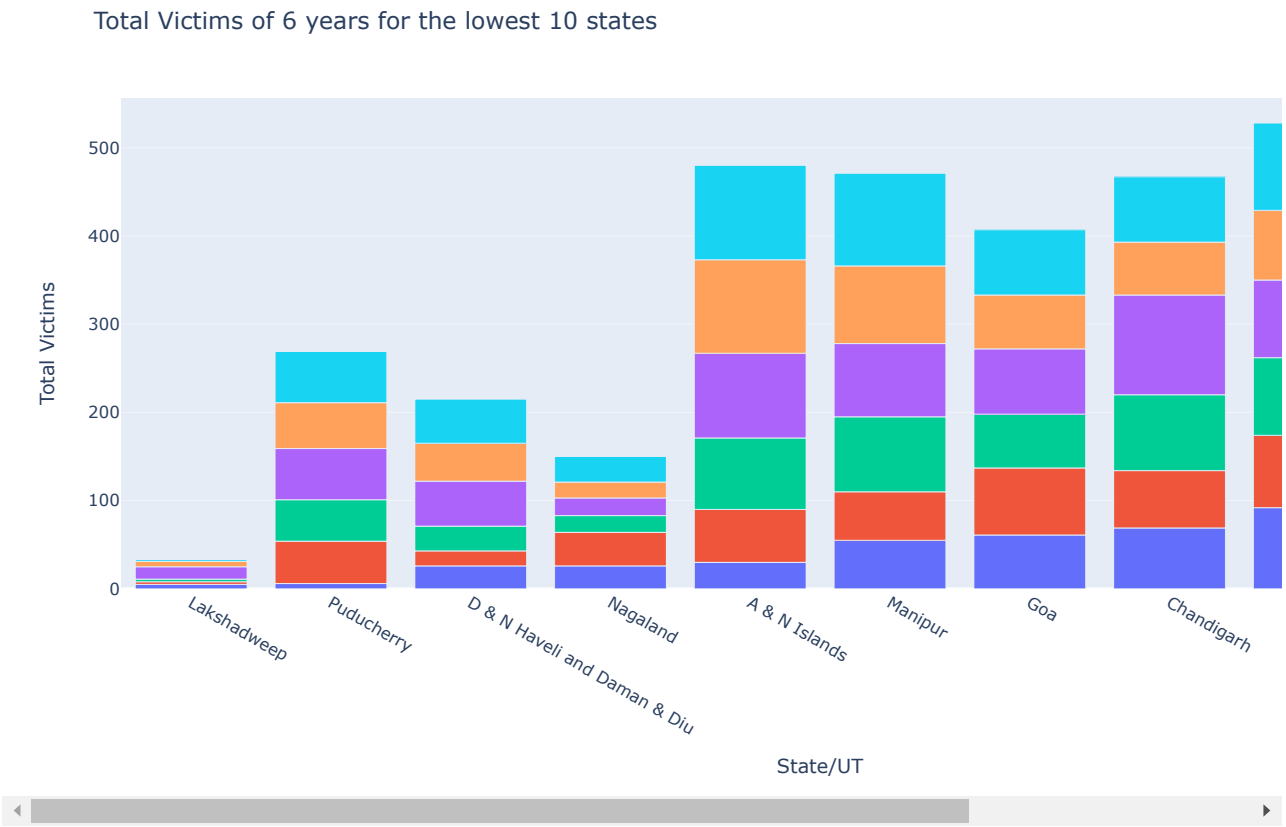
10 States with lowest count

In [74]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# datasets are named D16c, D17c, D18c, D19c, D20, D21
datasets = [D16c, D17c, D18c, D19c, D20, D21]

# List of states to compare (Lowest 10)
states_to_compare_lowest = [
    'Lakshadweep',
    'D & N Haveli and Daman & Diu',
    'Nagaland',
    'Puducherry',
    'Manipur',
    'A & N Islands',
    'Chandigarh',
    'Goa',
    'Arunachal Pradesh',
    'Sikkim'
]

# Create an empty DataFrame to store the count of total victims for each state and year
comparison_data_lowest = pd.DataFrame(columns=['Year', 'State/UT', 'Total Victims'])

# Loop through each dataset and extract the count for the specified states
for year, dataset in zip(range(2016, 2022), datasets):
    total_victims_by_state_lowest = dataset.groupby('State/UT')['Total Victims'].sum().reset_index()
    total_victims_by_state_lowest = total_victims_by_state_lowest[total_victims_by_state_lowest['State/UT'].isin(states_

    # Add the year column
    total_victims_by_state_lowest['Year'] = year

    # Append the data to the comparison DataFrame
    comparison_data_lowest = pd.concat([comparison_data_lowest, total_victims_by_state_lowest], ignore_index=True)

# Create traces for each state (lowest 10)
traces_lowest = []
for state_lowest in states_to_compare_lowest:
    trace_lowest = go.Scatter(
        x=comparison_data_lowest[comparison_data_lowest['State/UT'] == state_lowest]['Year'],
        y=comparison_data_lowest[comparison_data_lowest['State/UT'] == state_lowest]['Total Victims'],
        mode='lines',
        name=state_lowest
    )
    traces_lowest.append(trace_lowest)

# Create a subplot with a dropdown menu
fig_lowest = make_subplots(rows=1, cols=1, specs=[[{'type': 'scatter'}]])
dropdown_buttons_lowest = [
    {
        'method': 'update',
        'label': 'Show All',
        'args': [{'visible': [True] * len(traces_lowest)}]
    }
] + [
    {
        'method': 'update',
        'label': state_lowest,
        'args': [{'visible': [state_lowest == trace_lowest.name for trace_lowest in traces_lowest]}]
    }
    for state_lowest in states_to_compare_lowest
]
fig_lowest.update_layout(
    updatemenus=[{
        'active': 0,
        'buttons': dropdown_buttons_lowest,
        'showactive': True,
    }],
    height=600, width=1000,
    title='Comparison of Total Victims for Lowest 10 States (2016-2021)',
    xaxis_title='Year',
    yaxis_title='Total Victims'
)

# Add traces to the subplot
for trace_lowest in traces_lowest:
    fig_lowest.add_trace(trace_lowest)

# Show the interactive plot
fig_lowest.show()
```


Comparison of Total Victims for Lowest 10 States (2016-2021)
Show All
Total Victims
0
20
40
60
80
100
2016
2017
2018
2019
2020
2021
Year
Lakshadweep
D & N Haveli and Dai
Nagaland
Puducherry
Manipur
A & N Islands
Chandigarh
Goa
Arunachal Pradesh
Sikkim

In [75]:

```
# datasets are named D16c, D17c, D18c, D19c, D20, D21
datasets_lowest = [D16c, D17c, D18c, D19c, D20, D21]
states_to_compare_lowest = ['Lakshadweep', 'D & N Haveli and Daman & Diu', 'Nagaland', 'Puducherry', 'Manipur', 'A & N I

# Create an empty DataFrame to store the total count of victims for each state
total_victims_data_lowest = pd.DataFrame(columns=['State/UT', 'Total Victims', 'Year'])

# Loop through each dataset and extract the count for the specified states
for i, dataset in enumerate(datasets):
    total_victims_by_state_lowest = dataset.groupby('State/UT')['Total Victims'].sum().reset_index()
    total_victims_by_state_lowest = total_victims_by_state_lowest[total_victims_by_state_lowest['State/UT'].isin(states_

    # Add the year information to the DataFrame
    total_victims_by_state_lowest['Year'] = f'20{i + 16}'

    # Sort by total victims in ascending order and select the bottom 10
    total_victims_by_state_lowest = total_victims_by_state_lowest.sort_values(by='Total Victims').head(10)

    # Append the data to the total victims DataFrame
    total_victims_data_lowest = pd.concat([total_victims_data_lowest, total_victims_by_state_lowest], ignore_index=True)

# Create a bar chart for total victims across all years for the lowest 10 states
fig_lowest = px.bar(total_victims_data_lowest, x='State/UT', y='Total Victims', color='Year',
                    labels={'Total Victims': 'Total Victims'},
                    title='Total Victims of 6 years for the lowest 10 states ',
                    height=600, width=1200,
                    hover_data={'Year': True, 'Total Victims': ':.0f'})  # Specify the hover_data

# Customize the layout as needed
fig_lowest.update_layout(xaxis_title='State/UT', yaxis_title='Total Victims')

# Show the interactive plot
fig_lowest.show()
```



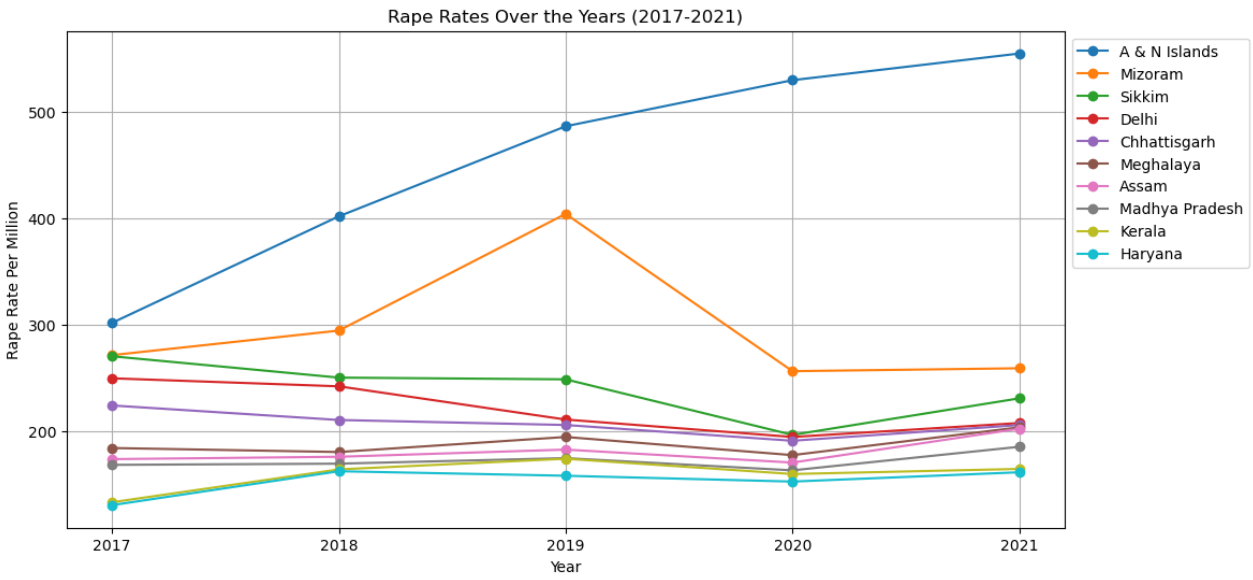
I created a seperate dataset of Rate of Rape - All years for Top states

In [76]:
```
# Define the data
data = {
    'State/UT': ['A & N Islands', 'Mizoram', 'Sikkim', 'Delhi', 'Chhattisgarh', 'Meghalaya', 'Assam', 'Madhya Pradesh',
    '2017': [301.89, 271.59, 270.54, 249.86, 224.37, 184.46, 174.04, 168.66, 133.58, 130.88],
    '2018': [401.81, 294.64, 250.51, 242.34, 210.76, 180.71, 176.25, 169.78, 164.40, 162.68],
    '2019': [486.29, 404.12, 248.90, 211.17, 206.06, 194.82, 182.97, 174.98, 174.41, 158.45],
    '2020': [529.53, 256.55, 196.78, 194.84, 191.29, 177.79, 170.80, 163.51, 160.08, 152.98],
    '2021': [554.52, 259.27, 231.06, 207.89, 205.99, 203.85, 202.57, 185.61, 164.84, 161.71]
}

# Create a DataFrame
rape_rate_table = pd.DataFrame(data)

# Display the table
rape_rate_table
```

Out[76]:

	State/UT	2017	2018	2019	2020	2021
0	A & N Islands	302	402	486	530	555
1	Mizoram	272	295	404	257	259
2	Sikkim	271	251	249	197	231
3	Delhi	250	242	211	195	208
4	Chhattisgarh	224	211	206	191	206
5	Meghalaya	184	181	195	178	204
6	Assam	174	176	183	171	203
7	Madhya Pradesh	169	170	175	164	186
8	Kerala	134	164	174	160	165
9	Haryana	131	163	158	153	162

In [77]:
```
# Set the State/UT column as the index
rape_rate_table.set_index('State/UT', inplace=True)

# Plot the data
plt.figure(figsize=(12, 6))
for state in rape_rate_table.index:
    plt.plot(rape_rate_table.columns, rape_rate_table.loc[state], label=state, marker='o')

plt.title('Rape Rates Over the Years (2017-2021)')
plt.xlabel('Year')
plt.ylabel('Rape Rate Per Million')
plt.legend(loc='upper left', bbox_to_anchor=(1,1))
plt.grid(True)
plt.show()
```



Let's explore what is the Rural and Urban distribution in India, to find if we can find any relation between them.

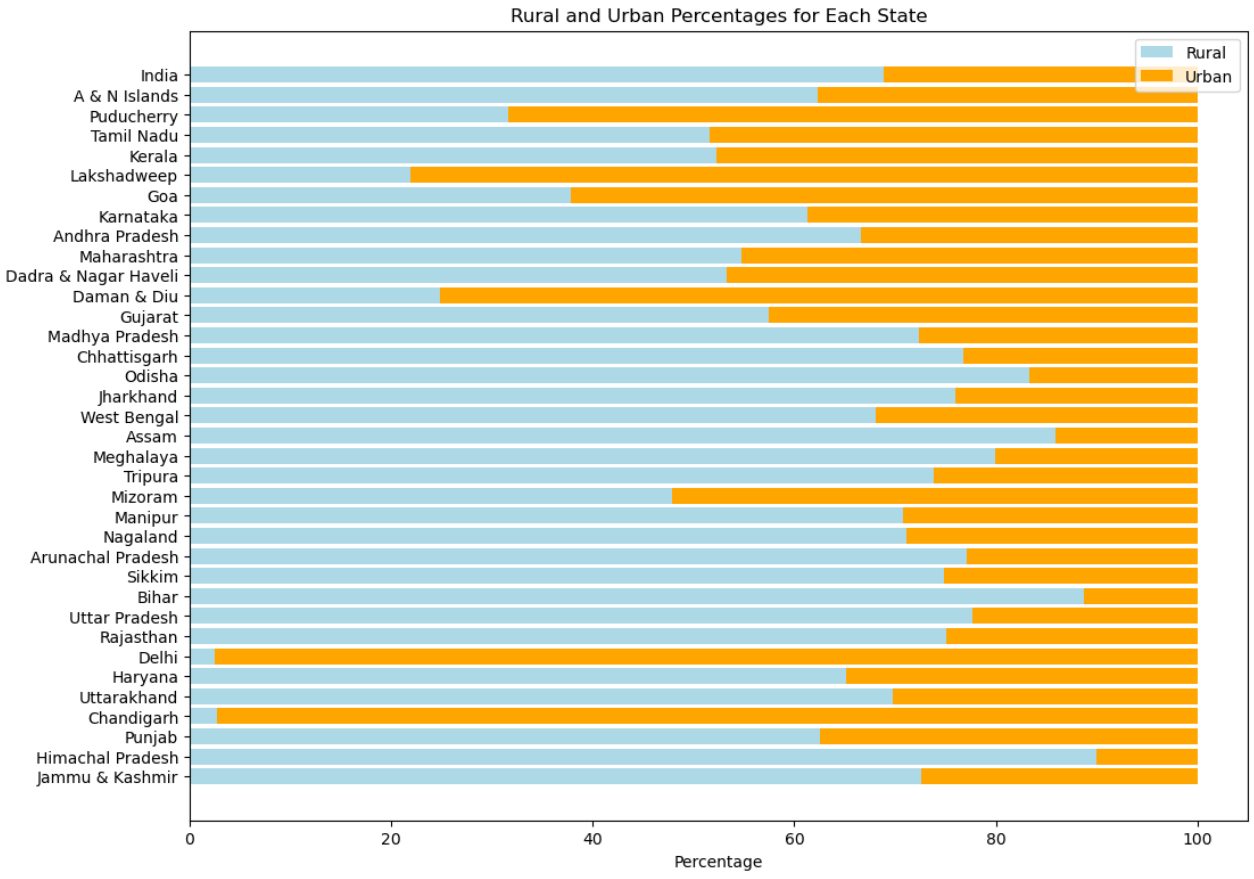
The analysis is explained in report

In [78]:
```
# accessing data extracted from NCRB

R_U_c = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\rural and urban - Copy.xlsx")

R_U_c.head()
```

Out[78]:

	State	Rural Percent	Urban Percent
0	Jammu & Kashmir	73	27
1	Himachal Pradesh	90	10
2	Punjab	63	37
3	Chandigarh	3	97
4	Uttarakhand	70	30

In [79]:
```
# Plotting
plt.figure(figsize=(12, 9))
plt.barh(R_U_c['State'], R_U_c['Rural Percent'], label='Rural', color='lightblue')
plt.barh(R_U_c['State'], R_U_c['Urban Percent'], left=R_U_c['Rural Percent'], label='Urban', color='orange')

plt.xlabel('Percentage')
plt.title('Rural and Urban Percentages for Each State')
plt.legend()
plt.show()
```



Let's explore and do some analysis on 5 states which have higher count and rate trend in all years - 'Madhya Pradesh', 'Uttar Pradesh', 'Rajasthan', 'Delhi', 'Maharashtra'

I separately stored these 5 states data in a separate file for easy analysis

In [80]:
```
#accessing the dataset

States_to_compare = pd.read_excel("C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\states to compare\\states.xlsx")
States_to_compare
```

Out[80]:

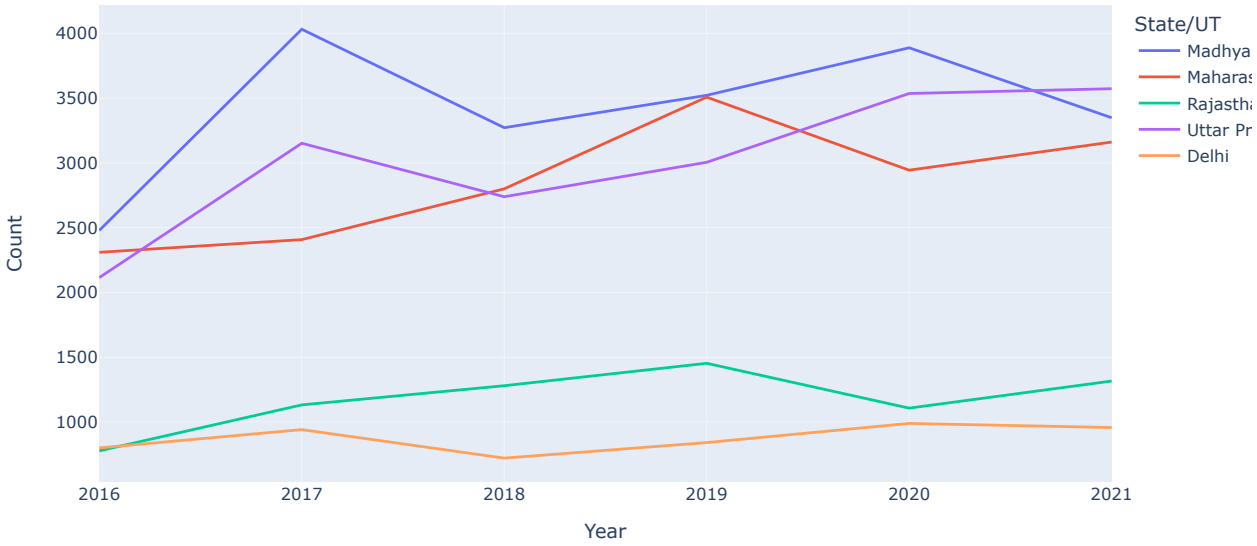
	State/UT	Total Girls	Total Women	Total Victims
0	Madhya Pradesh	2479	2429	4908
1	Maharashtra	2310	1906	4216
2	Rajasthan	777	2880	3657
3	Uttar Pradesh	2115	2702	4817
4	Delhi	800	1370	2170

Total Victims in all years for these 5 states

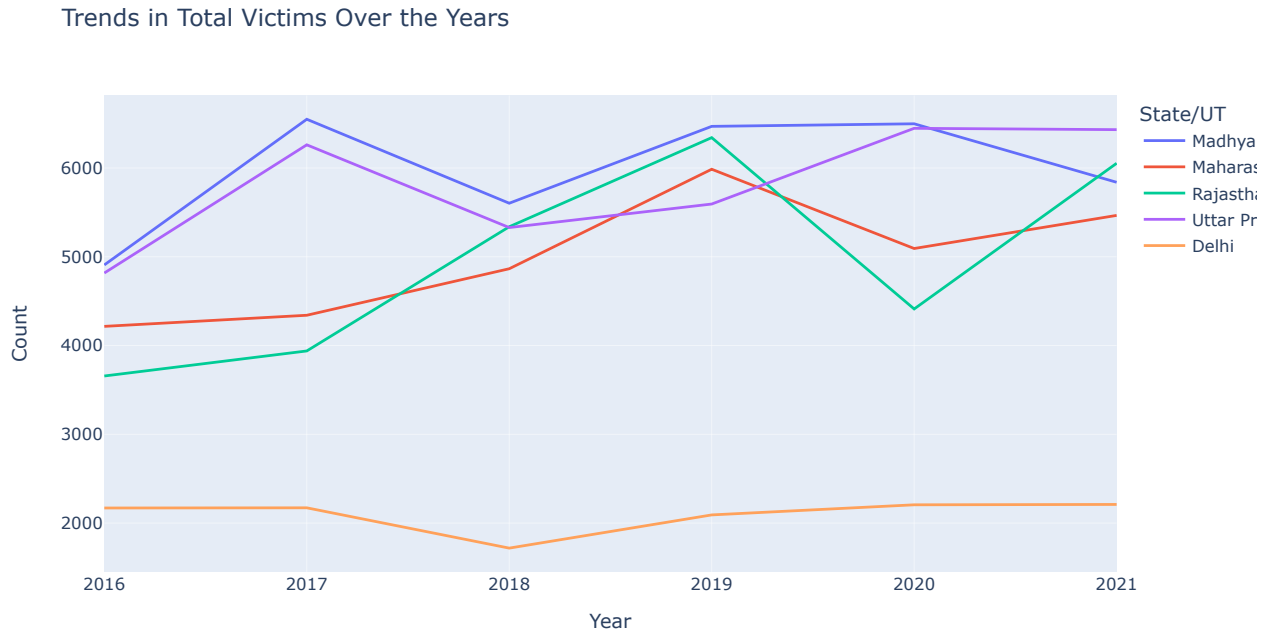
In [81]:
```
#accessing data

file_path = "C:\\Users\\aiman\\OneDrive\\Desktop\\Rapes in India\\states to compare\\states.xlsx"

# Create an empty DataFrame to store all data
all_data = pd.DataFrame()

# Read data from each sheet and append it to the DataFrame
for year in range(2016, 2022):
    sheet_name = str(year)
    data = pd.read_excel(file_path, sheet_name=sheet_name)
    data['Year'] = year  # Add a column for the year
    all_data = all_data.append(data)

# Create line plots for each column
fig_girls = px.line(all_data, x='Year', y='Total Girls', color='State/UT',
                    labels={'Total Girls': 'Count'}, title='Trends in Total Girls Over the Years',
                    line_shape='linear', render_mode='svg')

fig_women = px.line(all_data, x='Year', y='Total Women', color='State/UT',
                    labels={'Total Women': 'Count'}, title='Trends in Total Women Over the Years',
                    line_shape='linear', render_mode='svg')

fig_victims = px.line(all_data, x='Year', y='Total Victims', color='State/UT',
                      labels={'Total Victims': 'Count'}, title='Trends in Total Victims Over the Years',
                      line_shape='linear', render_mode='svg')

# Show the interactive plots
fig_girls.show()
fig_women.show()
fig_victims.show()
```

```
C:\Users\aiman\AppData\Local\Temp\ipykernel_25676\2534298078.py:13: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\aiman\AppData\Local\Temp\ipykernel_25676\2534298078.py:13: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\aiman\AppData\Local\Temp\ipykernel_25676\2534298078.py:13: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\aiman\AppData\Local\Temp\ipykernel_25676\2534298078.py:13: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\aiman\AppData\Local\Temp\ipykernel_25676\2534298078.py:13: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\aiman\AppData\Local\Temp\ipykernel_25676\2534298078.py:13: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
```

Trends in Total Girls Over the Years

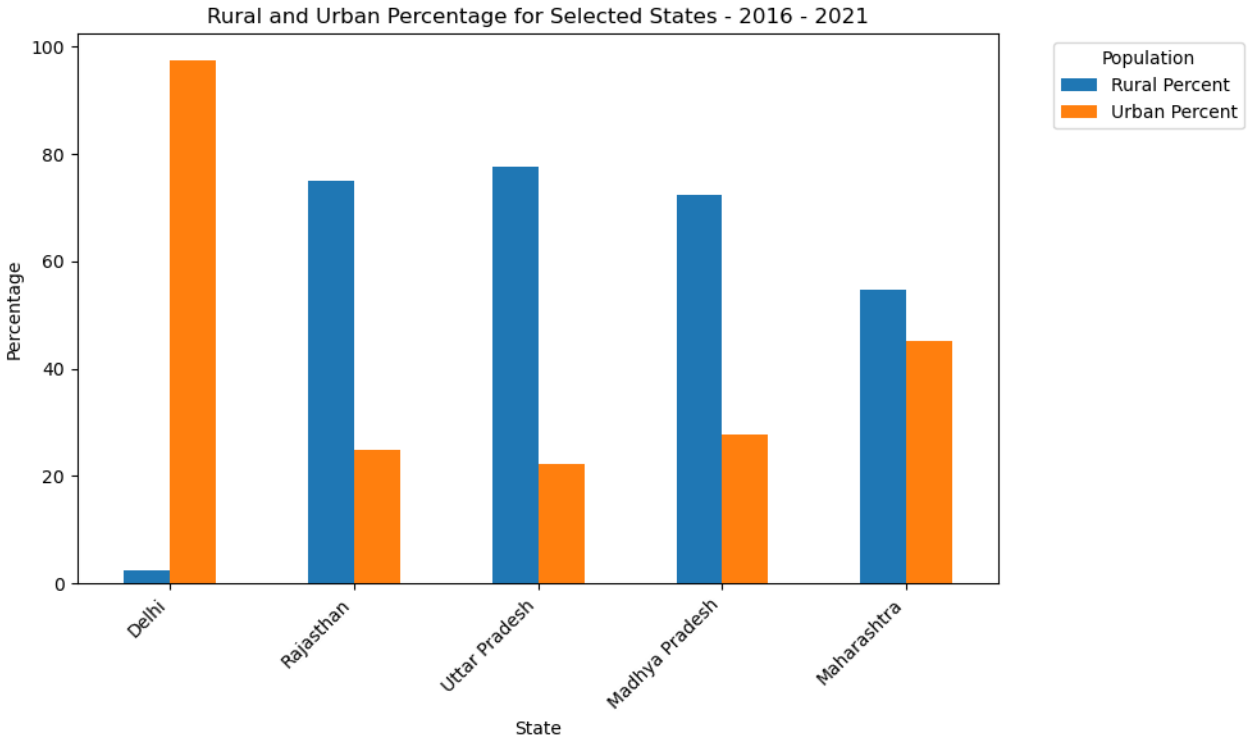


Trends in Total Women Over the Years



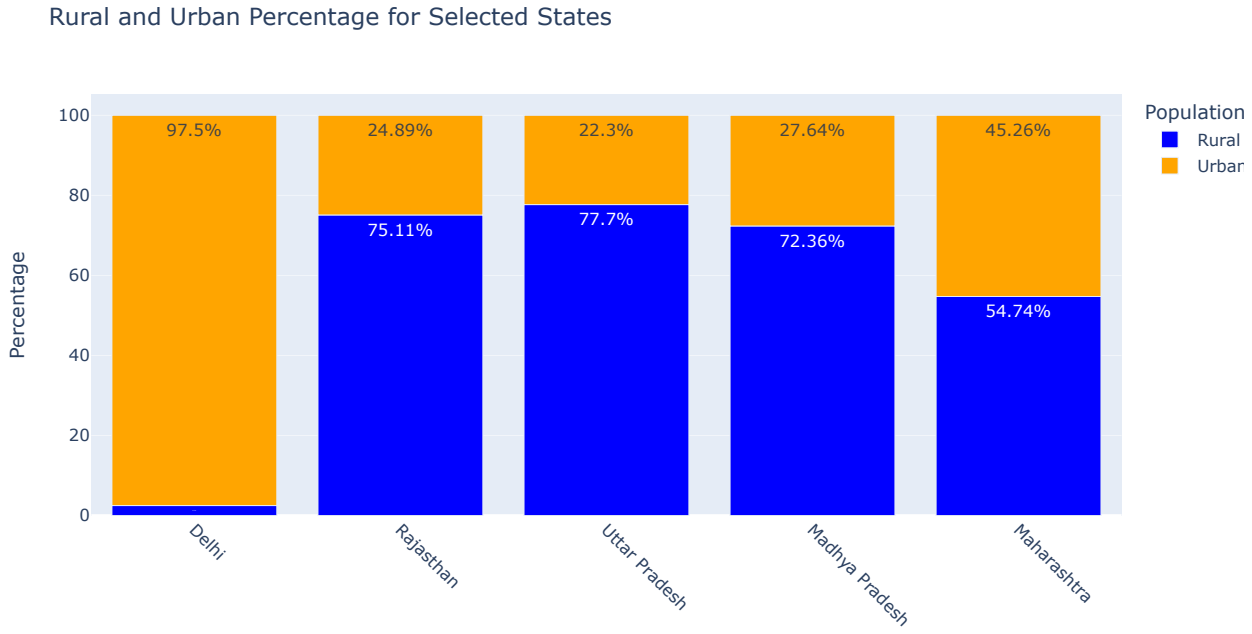
Comparison between Rural and Urban distribution

In [83]:
```
# Filtering data for specific states
selected_states = ['Madhya Pradesh', 'Uttar Pradesh', 'Rajasthan', 'Delhi', 'Maharashtra']
selected_data = R_U_c[R_U_c['State'].isin(selected_states)]

# Plotting the 'Rural Percent' and 'Urban Percent' columns for the selected states
fig, ax = plt.subplots(figsize=(10, 6))
selected_data.plot(kind='bar', x='State', y=['Rural Percent', 'Urban Percent'], ax=ax)
ax.set_ylabel('Percentage')
ax.set_title('Rural and Urban Percentage for Selected States - 2016 - 2021')
ax.legend(title='Population', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



In [84]:

```
# Filtering data for specific states
selected_states = ['Madhya Pradesh', 'Uttar Pradesh', 'Rajasthan', 'Delhi', 'Maharashtra']
selected_data = R_U_c[R_U_c['State'].isin(selected_states)]

# Creating a bar plot using Plotly with percentage values on the bars
fig = px.bar(selected_data, x='State', y=['Rural Percent', 'Urban Percent'],
             title='Rural and Urban Percentage for Selected States',
             labels={'value': 'Percentage', 'variable': 'Population'},
             color_discrete_map={'Rural Percent': 'blue', 'Urban Percent': 'orange'})

# Adding text annotations for percentage values on the bars
for trace in fig.data:
    trace.text = ['{}%'.format(round(val, 2)) for val in trace.y]

# Rotating x-axis labels for better visibility
fig.update_xaxes(tickangle=45, tickmode='array')

# Show the plot
fig.show()
```



Conclusion

In conclusion, the comprehensive analysis of rape cases in India underscores the multifaceted nature of this pervasive issue. The study delved into national and state-level data, comparing high and low victim count states, exploring age group vulnerabilities, and assessing the rate of reported cases per million.

The findings revealed stark variations across states, emphasizing the need for targeted interventions considering regional dynamics. Madhya Pradesh, Maharashtra, Rajasthan, and Uttar Pradesh emerged as states with significant challenges, demanding focused attention and tailored strategies.

Understanding age group vulnerabilities showcased the importance of addressing the unique needs of victims across different life stages. The analysis of rates per million highlighted the disparities in reporting mechanisms and the urgent need for a more consistent and victim-centric approach nationwide.

Despite the gravity of the situation, it is crucial to acknowledge the limitations of reported data, as dark data remains a significant concern. Factors contributing to underreporting, including societal norms, legal barriers, and fear, necessitate a holistic approach for improvement.

Future work should explore diverse data sources, leverage technology, and engage in community empowerment programs to dismantle stigmas and encourage reporting. Legal reforms, international collaboration, and sustained public awareness campaigns are pivotal components of a comprehensive strategy to address this pervasive issue.

In moving forward, continuous evaluation of initiatives, integration of crisis intervention, and counselling services into the reporting process will contribute to a more responsive and supportive system. By prioritizing victim protections, fostering public trust in law enforcement, and dismantling barriers to reporting, India can strive towards a safer and more inclusive society. The analysis serves as a foundation for informed policy decisions, encouraging collaborative efforts to combat sexual violence and champion the rights and well-being of survivors.