In [1]:
```
#import all useful libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter("ignore")
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
#read the dataset
df=pd.read_csv("Downloads/Suicides in India 2001-2012.csv.zip")
tdf=pd.read_csv("Downloads/Suicides in India 2001-2012.csv.zip")
```

In [3]:
```
df.rename(columns={'Type_code':'Issues'},inplace=True)
tdf.rename(columns={'Type_code':'Issues'},inplace=True)
```

In [4]:
```
tdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237519 entries, 0 to 237518
Data columns (total 7 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   State      237519 non-null  object
 1   Year       237519 non-null  int64 
 2   Issues     237519 non-null  object
 3   Type       237519 non-null  object
 4   Gender     237519 non-null  object
 5   Age_group  237519 non-null  object
 6   Total      237519 non-null  int64 
dtypes: int64(2), object(5)
memory usage: 12.7+ MB
```

In [5]:
```
#statistical analysis
tdf.describe()
```

Out[5]:

	Year	Total
count	237519.000000	237519.000000
mean	2006.500448	55.034477
std	3.452240	792.749038
min	2001.000000	0.000000
25%	2004.000000	0.000000
50%	2007.000000	0.000000
75%	2010.000000	6.000000
max	2012.000000	63343.000000

In [6]:
```
tdf.head(10)
```

Out[6]:

	State	Year	Issues	Type	Gender	Age_group	Total
0	A & N Islands	2001	Causes	Illness (Aids/STD)	Female	0-14	0

	State	Year	Issues	Type	Gender	Age_group	Total
1	A & N Islands	2001	Causes	Bankruptcy or Sudden change in Economic	Female	0-14	0
2	A & N Islands	2001	Causes	Cancellation/Non-Settlement of Marriage	Female	0-14	0
3	A & N Islands	2001	Causes	Physical Abuse (Rape/Incest Etc.)	Female	0-14	0
4	A & N Islands	2001	Causes	Dowry Dispute	Female	0-14	0
5	A & N Islands	2001	Causes	Family Problems	Female	0-14	0
6	A & N Islands	2001	Causes	Ideological Causes/Hero Worshipping	Female	0-14	0
7	A & N Islands	2001	Causes	Other Prolonged Illness	Female	0-14	0
8	A & N Islands	2001	Causes	Property Dispute	Female	0-14	0
9	A & N Islands	2001	Causes	Fall in Social Reputation	Female	0-14	0

In [7]:
```
tdf.tail(10)
```

Out[7]:

	State	Year	Issues	Type	Gender	Age_group	Total
237509	West Bengal	2012	Social_Status	Seperated	Female	0-100+	200
237510	West Bengal	2012	Social_Status	Married	Female	0-100+	3927
237511	West Bengal	2012	Social_Status	Divorcee	Female	0-100+	182
237512	West Bengal	2012	Social_Status	Widowed/Widower	Female	0-100+	455
237513	West Bengal	2012	Social_Status	Never Married	Female	0-100+	1513
237514	West Bengal	2012	Social_Status	Seperated	Male	0-100+	149
237515	West Bengal	2012	Social_Status	Widowed/Widower	Male	0-100+	233
237516	West Bengal	2012	Social_Status	Married	Male	0-100+	5451
237517	West Bengal	2012	Social_Status	Divorcee	Male	0-100+	189
237518	West Bengal	2012	Social_Status	Never Married	Male	0-100+	2658

In [8]:
```
tdf['Gender']=tdf['Gender'].map({'Female':2,'Male':1})
tdf
```

Out[8]:

	State	Year	Issues	Type	Gender	Age_group	Total
0	A & N Islands	2001	Causes	Illness (Aids/STD)	2	0-14	0
1	A & N Islands	2001	Causes	Bankruptcy or Sudden change in Economic	2	0-14	0
2	A & N Islands	2001	Causes	Cancellation/Non-Settlement of Marriage	2	0-14	0

	State	Year	Issues	Type	Gender	Age_group	Total
3	A & N Islands	2001	Causes	Physical Abuse (Rape/Incest Etc.)	2	0-14	0
4	A & N Islands	2001	Causes	Dowry Dispute	2	0-14	0
...	...	...	...	...	...	...	...
237514	West Bengal	2012	Social_Status	Seperated	1	0-100+	149
237515	West Bengal	2012	Social_Status	Widowed/Widower	1	0-100+	233
237516	West Bengal	2012	Social_Status	Married	1	0-100+	5451
237517	West Bengal	2012	Social_Status	Divorcee	1	0-100+	189
237518	West Bengal	2012	Social_Status	Never Married	1	0-100+	2658

237519 rows × 7 columns

In [9]:
```
tdf.isnull()
```

Out[9]:

	State	Year	Issues	Type	Gender	Age_group	Total
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
237514	False	False	False	False	False	False	False
237515	False	False	False	False	False	False	False
237516	False	False	False	False	False	False	False
237517	False	False	False	False	False	False	False
237518	False	False	False	False	False	False	False

237519 rows × 7 columns

In [10]:
```
tdf.notnull()
```

Out[10]:

	State	Year	Issues	Type	Gender	Age_group	Total
0	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True

	State	Year	Issues	Type	Gender	Age_group	Total
4	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...
237514	True	True	True	True	True	True	True
237515	True	True	True	True	True	True	True
237516	True	True	True	True	True	True	True
237517	True	True	True	True	True	True	True
237518	True	True	True	True	True	True	True

237519 rows × 7 columns

In [11]:
```
tdf.isnull().sum()
```

Out[11]:
```
State        0
Year         0
Issues       0
Type         0
Gender       0
Age_group    0
Total        0
dtype: int64
```

In [12]:
```
tdf.notnull().sum()
```

Out[12]:
```
State        237519
Year         237519
Issues       237519
Type         237519
Gender       237519
Age_group    237519
Total        237519
dtype: int64
```

In [13]:
```
tdf.shape
```

Out[13]:
```
(237519, 7)
```

In [14]:
```
tdf['Total'].value_counts()
```

Out[14]:
```
0        135481
1         16047
2          9942
3          6704
4          5121
          ...
17614         1
1606          1
1734          1
8133          1
13369         1
Name: Total, Length: 2180, dtype: int64
```

In [15]:
```
# changing issues into numerics
tdf['Issues']=tdf['Issues'].map({'Causes':1,'Means_adopted':2,'Professional_Profile'
tdf
```

Out[15]:

	State	Year	Issues	Type	Gender	Age_group	Total

	State	Year	Issues	Type	Gender	Age_group	Total
0	A & N Islands	2001	1	Illness (Aids/STD)	2	0-14	0
1	A & N Islands	2001	1	Bankruptcy or Sudden change in Economic	2	0-14	0
2	A & N Islands	2001	1	Cancellation/Non-Settlement of Marriage	2	0-14	0
3	A & N Islands	2001	1	Physical Abuse (Rape/Incest Etc.)	2	0-14	0
4	A & N Islands	2001	1	Dowry Dispute	2	0-14	0
...	...	...	...	...	...	...	...
237514	West Bengal	2012	5	Seperated	1	0-100+	149
237515	West Bengal	2012	5	Widowed/Widower	1	0-100+	233
237516	West Bengal	2012	5	Married	1	0-100+	5451
237517	West Bengal	2012	5	Divorcee	1	0-100+	189
237518	West Bengal	2012	5	Never Married	1	0-100+	2658

237519 rows × 7 columns

In [16]:
```
tdf=tdf.dropna()
tdf
```

Out[16]:

	State	Year	Issues	Type	Gender	Age_group	Total
0	A & N Islands	2001	1	Illness (Aids/STD)	2	0-14	0
1	A & N Islands	2001	1	Bankruptcy or Sudden change in Economic	2	0-14	0
2	A & N Islands	2001	1	Cancellation/Non-Settlement of Marriage	2	0-14	0
3	A & N Islands	2001	1	Physical Abuse (Rape/Incest Etc.)	2	0-14	0
4	A & N Islands	2001	1	Dowry Dispute	2	0-14	0
...	...	...	...	...	...	...	...
237514	West Bengal	2012	5	Seperated	1	0-100+	149
237515	West Bengal	2012	5	Widowed/Widower	1	0-100+	233
237516	West Bengal	2012	5	Married	1	0-100+	5451

	State	Year	Issues	Type	Gender	Age_group	Total
237517	West Bengal	2012	5	Divorcee	1	0-100+	189
237518	West Bengal	2012	5	Never Married	1	0-100+	2658

237519 rows × 7 columns

In [17]:
```
df['State'].unique()
```

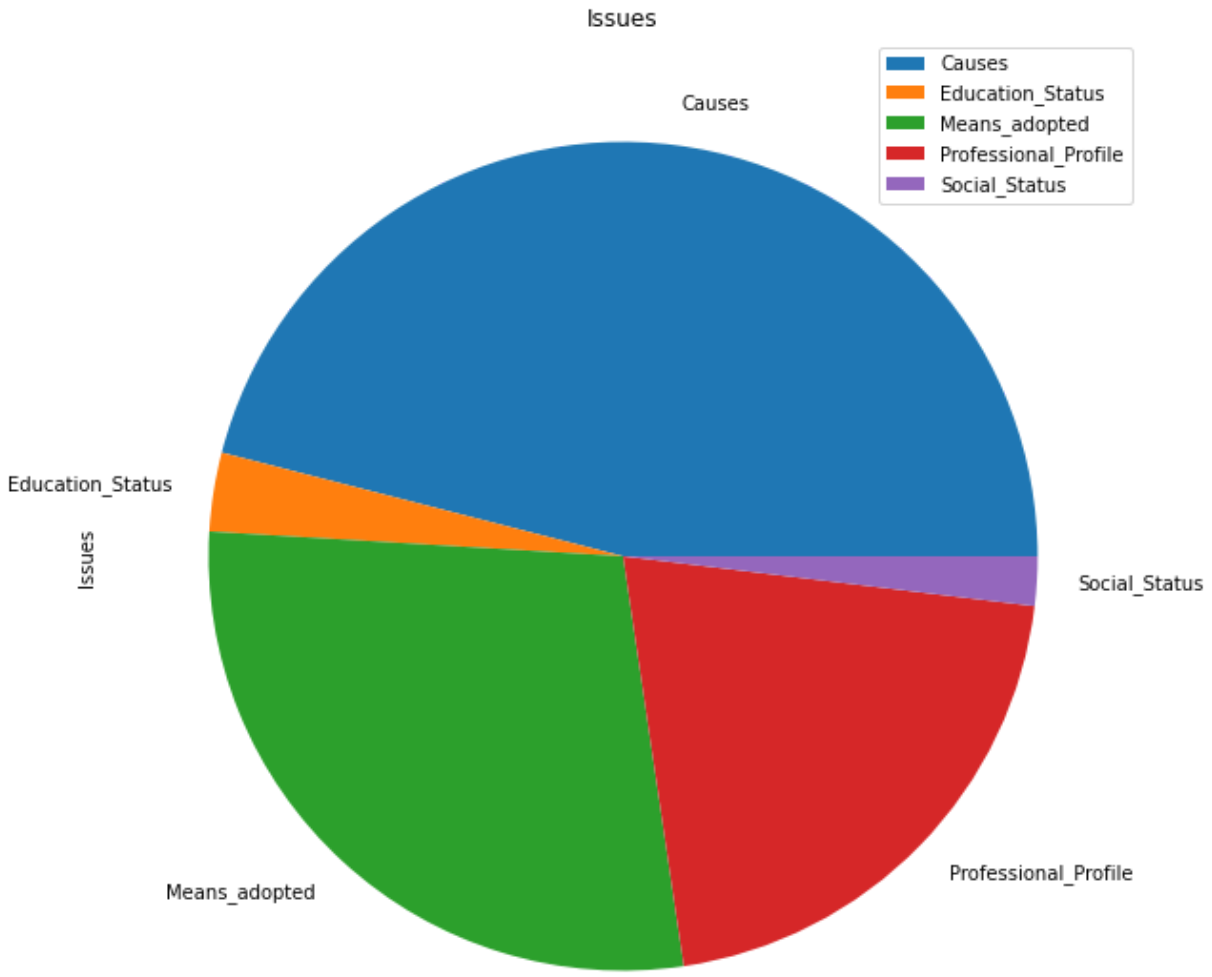
Out[17]:
```
array(['A & N Islands', 'Andhra Pradesh', 'Arunachal Pradesh', 'Assam',
       'Bihar', 'Chandigarh', 'Chhattisgarh', 'D & N Haveli',
       'Daman & Diu', 'Delhi (Ut)', 'Goa', 'Gujarat', 'Haryana',
       'Himachal Pradesh', 'Jammu & Kashmir', 'Jharkhand', 'Karnataka',
       'Kerala', 'Lakshadweep', 'Madhya Pradesh', 'Maharashtra',
       'Manipur', 'Meghalaya', 'Mizoram', 'Nagaland', 'Odisha',
       'Puducherry', 'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu',
       'Total (All India)', 'Total (States)', 'Total (Uts)', 'Tripura',
       'Uttar Pradesh', 'Uttarakhand', 'West Bengal'], dtype=object)
```

In [18]:
```
df['Age_group'].unique()
```

Out[18]:
```
array(['0-14', '15-29', '30-44', '45-59', '60+', '0-100+'], dtype=object)
```

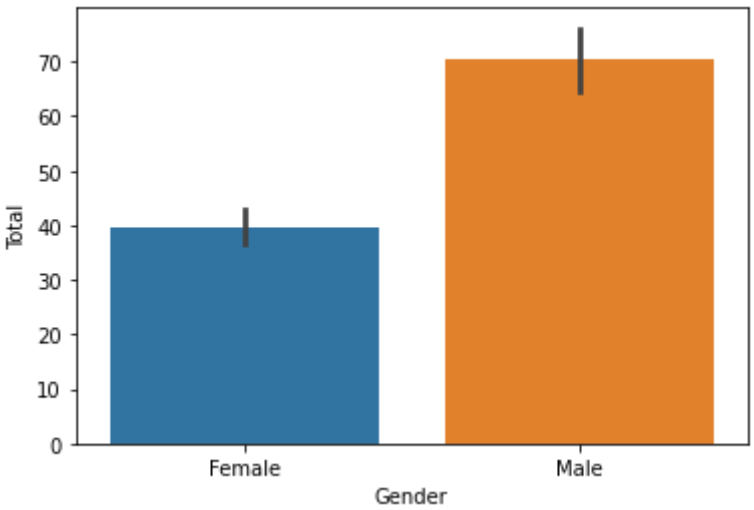
In [19]:
```
counts = df['Issues'].value_counts().sort_index()
print(counts)
# Plot a pie chart
counts.plot(kind='pie', title='Issues',figsize=(20,10))

plt.legend()
plt.show()
```

```
Causes                  109200
Education_Status          7296
Means_adopted            67200
Professional_Profile     49263
Social_Status             4560
Name: Issues, dtype: int64
```



In [20]:
```
sns.barplot(x='Gender', y='Total',data=df )
```

Out[20]: <AxesSubplot:xlabel='Gender', ylabel='Total'>



In [21]:
```
bystate = pd.DataFrame(tdf['Total'].groupby(tdf['State']).sum())
bystate = bystate.reset_index().sort_values('Total',ascending =False)  # Grouping by
bystate
```

Out[21]:

	State	Total
31	Total (All India)	2911862

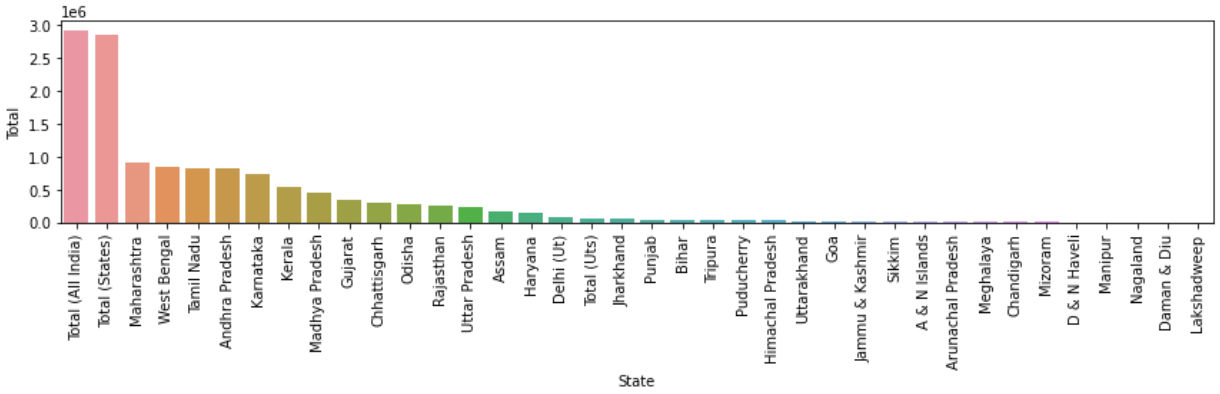
	State	Total
32	Total (States)	2858026
20	Maharashtra	901945
37	West Bengal	849936
30	Tamil Nadu	818691
1	Andhra Pradesh	814059
16	Karnataka	734825
17	Kerala	538946
19	Madhya Pradesh	451535
11	Gujarat	330858
6	Chhattisgarh	302354
25	Odisha	267234
28	Rajasthan	255134
35	Uttar Pradesh	233352
3	Assam	172276
12	Haryana	147176
9	Delhi (Ut)	84272
33	Total (Uts)	53836
15	Jharkhand	49720
27	Punjab	46350
4	Bihar	46214
34	Tripura	45965
26	Puducherry	32144
13	Himachal Pradesh	26562
36	Uttarakhand	18496
10	Goa	17363
14	Jammu & Kashmir	14821
29	Sikkim	9606
0	A & N Islands	8109
2	Arunachal Pradesh	6633
22	Meghalaya	5415
5	Chandigarh	5164
23	Mizoram	4154
7	D & N Haveli	3430
21	Manipur	2102
24	Nagaland	1728
8	Daman & Diu	1391

	State	Total
18	Lakshadweep	50

In [22]:
```
bystate['Total'].sum()  # total suicide count is 4336148(total states count) to keep
```

Out[22]:
```
13071734
```

In [23]:
```
plt.figure(figsize=(12,4))
sns.barplot(x='State',y='Total', data=bystate)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

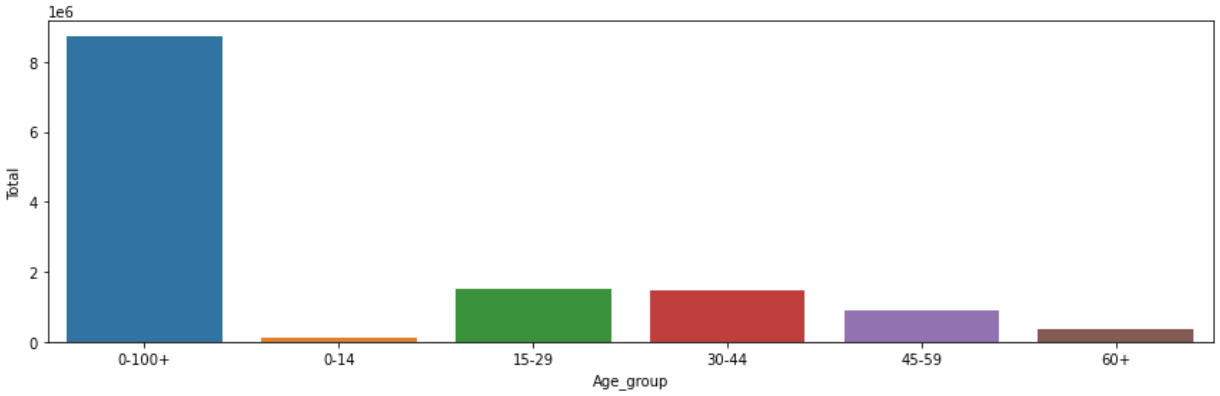


In [24]:
```
byage = pd.DataFrame(df.groupby('Age_group')['Total'].sum())
byage = byage.reset_index()                # Grouping by Age_group
byage
```

Out[24]:

	Age_group	Total
0	0-100+	8735586
1	0-14	98410
2	15-29	1534037
3	30-44	1471599
4	45-59	885177
5	60+	346925

In [25]:
```
plt.figure(figsize=(12,4))
sns.barplot(x='Age_group',y='Total', data=byage)
plt.tight_layout()
plt.show()
```

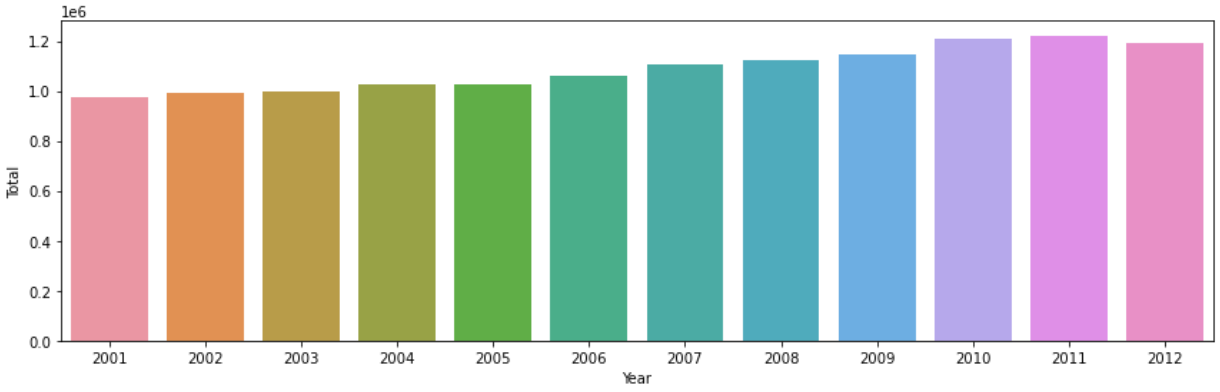


In [26]:
```
byyear = pd.DataFrame(df.groupby('Year')['Total'].sum())
byyear = byyear.reset_index()                # Grouping by year
byyear
```

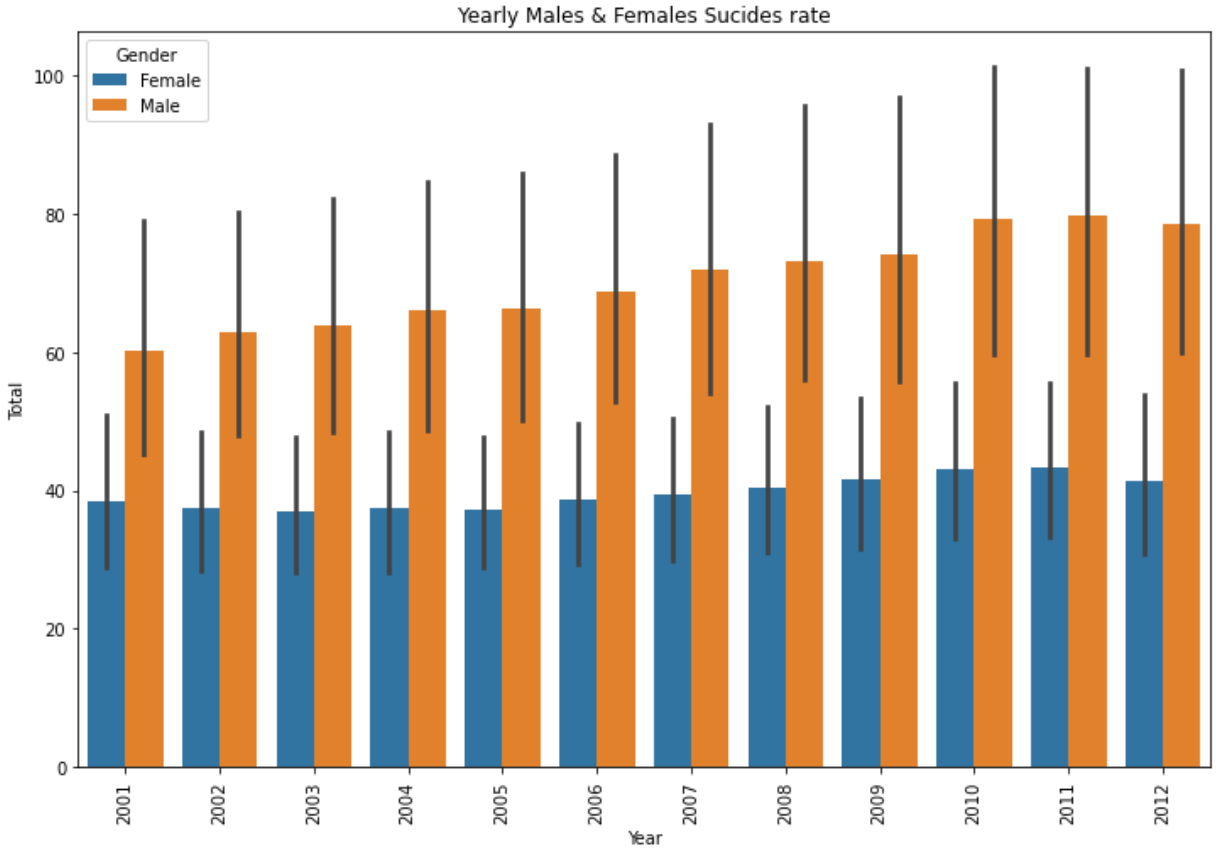
Out[26]:

	Year	Total
0	2001	976464
1	2002	993648
2	2003	997622
3	2004	1023137
4	2005	1025201
5	2006	1062991
6	2007	1103667
7	2008	1125082
8	2009	1144033
9	2010	1211322
10	2011	1219499
11	2012	1189068

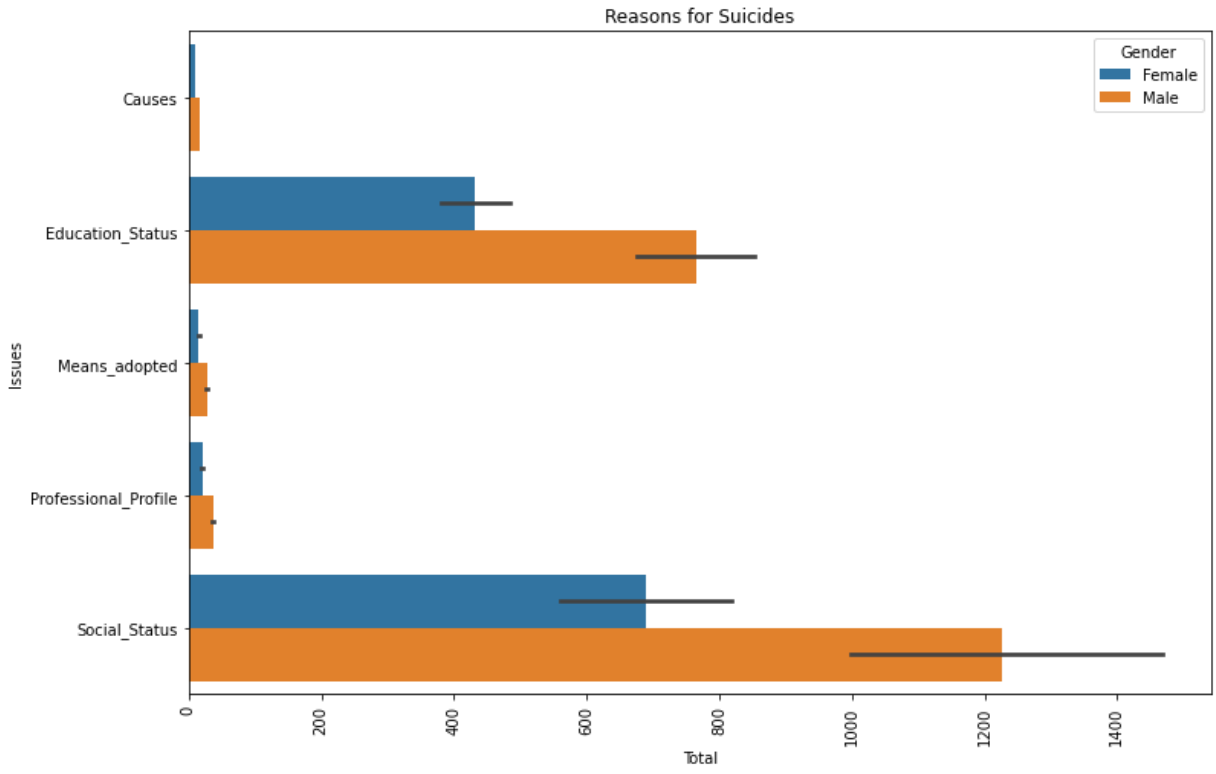
In [27]:
```
plt.figure(figsize=(12,4))
sns.barplot(x='Year',y='Total', data=byyear)
plt.tight_layout()
plt.show()
```



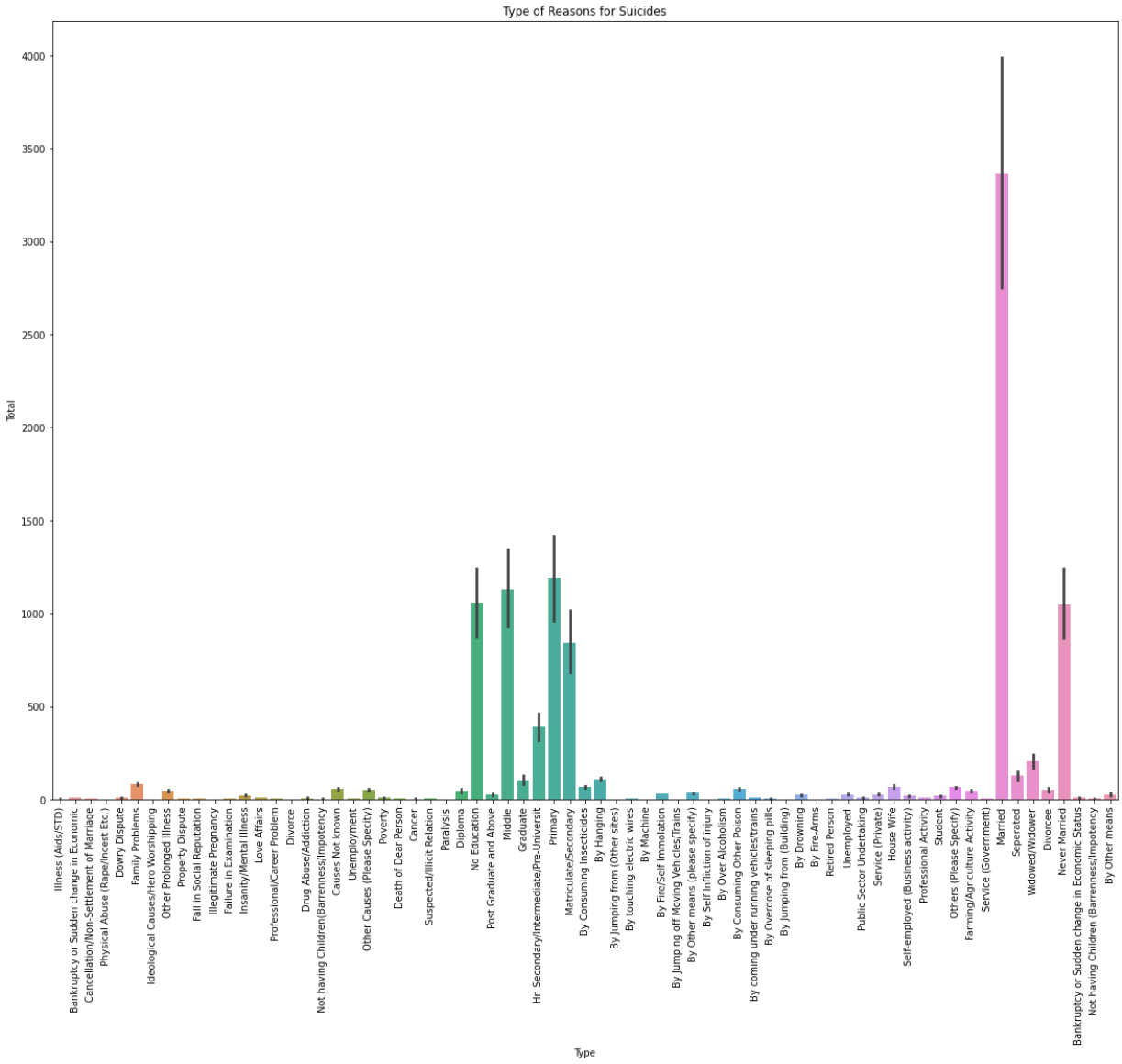
In [28]:
```
plt.figure(figsize=(12,8))
plt.title('Yearly Males & Females Sucides rate')
ax = sns.barplot(x = 'Year', y = 'Total', hue = 'Gender', data = df)
plt.xticks(rotation=90)
plt.show()
```



In [29]:
```
plt.figure(figsize=(12,8))
plt.title('Reasons for Suicides')
ax = sns.barplot(y = 'Issues', x = 'Total', hue = 'Gender', data = df)
plt.xticks(rotation=90)
plt.show()
```

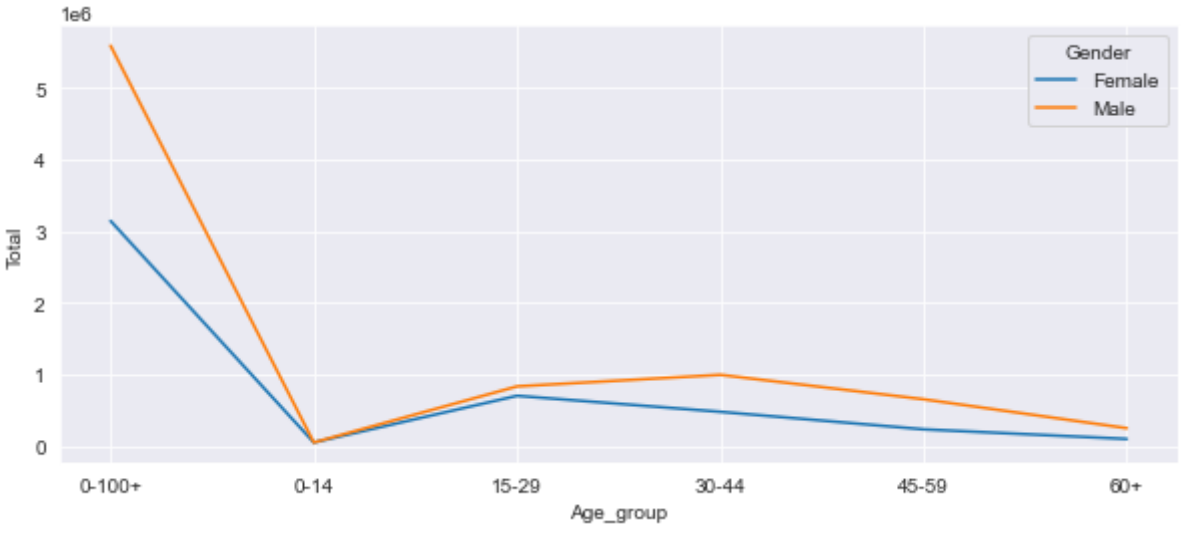


In [30]:
```
plt.figure(figsize=(20,15))
plt.title('Type of Reasons for Suicides')
ax = sns.barplot(x = 'Type', y = 'Total', data = tdf)
plt.xticks(rotation=90)
plt.show()
```



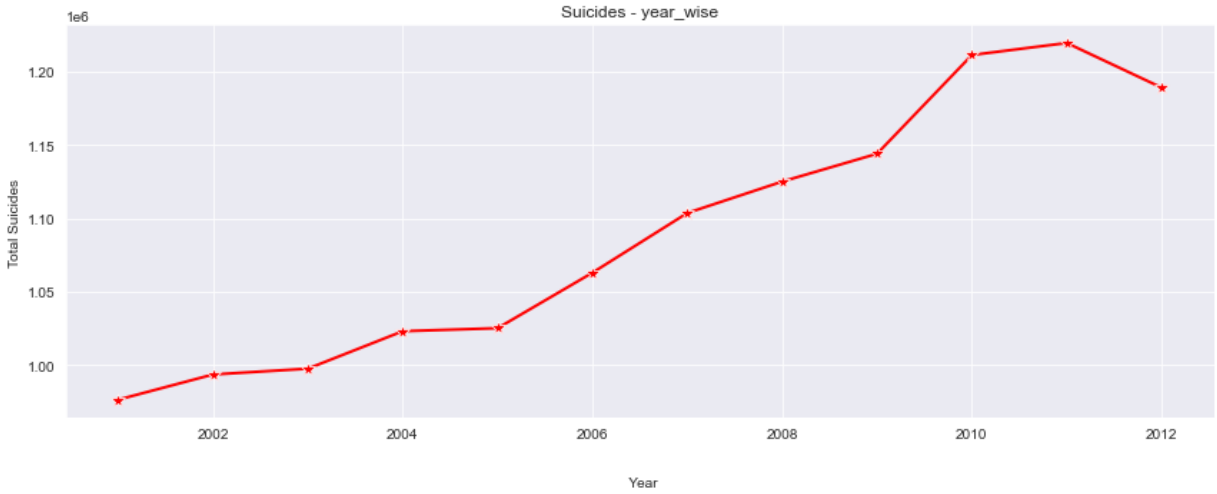
In [31]:
```
x = pd.DataFrame(df.groupby(['Age_group','Gender'])['Total'].sum())
plt.figure(figsize=(10,4))
sns.set_style('darkgrid')
sns.lineplot(x='Age_group', y='Total' ,data=x, hue = 'Gender')
```

Out[31]: <AxesSubplot:xlabel='Age_group', ylabel='Total'>



In [32]:
```
plt.figure(figsize=(12,5))
sns.lineplot(x='Year', y='Total', data=byyear, lw=2, marker="*", markersize=10, colo
plt.title('Suicides - year_wise')
```

```
plt.xlabel('\n \n  Year')
plt.ylabel('Total Suicides')
plt.tight_layout()
plt.show()
```



In [33]:
```
# Building the dataset
X= tdf.drop(["Issues","State","Type","Age_group"],axis=1)
X.head()
```

Out[33]:

	Year	Gender	Total
0	2001	2	0
1	2001	2	0
2	2001	2	0
3	2001	2	0
4	2001	2	0

In [34]:
```
#Select the target variable
Y=tdf["Issues"]
Y.head()
```

Out[34]:
```
0    1
1    1
2    1
3    1
4    1
Name: Issues, dtype: int64
```

In [35]:
```
#Train and split the dataset
from sklearn.model_selection import train_test_split
# Break off validation set from training data
X_train, X_valid, Y_train, Y_valid = train_test_split(X, Y, train_size=0.8, test_siz
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_valid=scaler.transform(X_valid)
```

In [36]:
```
len(X_train),len(Y_train)
```

Out[36]:
```
(190015, 190015)
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
# Define model. Specify a number for random_state to ensure same results each run
model = LogisticRegression()
```

```
# Fit model
model.fit(X_train,Y_train)
```

Out[37]:
```
LogisticRegression()
```

In [38]:
```
#view the accuracy
LogisticRegression_score=model.score(X_valid,Y_valid)
print("Accuracy obtained by LogisticRegression_model:",LogisticRegression_score*100)
```

```
Accuracy obtained by LogisticRegression_model: 46.52660828561805
```

In [39]:
```
X= tdf.drop(["Total","State","Type","Age_group"],axis=1)
X.head()
```

Out[39]:

	Year	Issues	Gender
0	2001	1	2
1	2001	1	2
2	2001	1	2
3	2001	1	2
4	2001	1	2

In [40]:
```
Y=tdf["Total"]
Y.head()
```

Out[40]:
```
0    0
1    0
2    0
3    0
4    0
Name: Total, dtype: int64
```

In [41]:
```
from sklearn.model_selection import train_test_split
# Break off validation set from training data
X_train, X_valid, Y_train, Y_valid = train_test_split(X, Y, train_size=0.8, test_siz
from sklearn.ensemble import RandomForestClassifier
# Define model. Specify a number for random_state to ensure same results each run
model = RandomForestClassifier()
# Fit model
model.fit(X_train,Y_train)
```

Out[41]:
```
RandomForestClassifier()
```

In [42]:
```
RandomForestClassifier_score=model.score(X_valid,Y_valid)
print("Accuracy obtained by RandomForestClassifier_model:",RandomForestClassifier_sc
```

```
Accuracy obtained by RandomForestClassifier_model: 57.11098012798922
```

In [43]:
```
X= tdf.drop(["State","Type","Age_group"],axis=1)
X.head()
```

Out[43]:

	Year	Issues	Gender	Total
0	2001	1	2	0
1	2001	1	2	0
2	2001	1	2	0
3	2001	1	2	0
4	2001	1	2	0

In [44]:
```
Y=tdf["Age_group"]
Y.head()
```

Out[44]:
```
0    0-14
1    0-14
2    0-14
3    0-14
4    0-14
Name: Age_group, dtype: object
```

In [45]:
```
from sklearn.model_selection import train_test_split
# Break off validation set from training data
X_train, X_valid, Y_train, Y_valid = train_test_split(X, Y, train_size=0.8, test_siz
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_valid=scaler.transform(X_valid)
```

In [46]:
```
from sklearn.ensemble import RandomForestClassifier
# Define model. Specify a number for random_state to ensure same results each run
model = RandomForestClassifier()
# Fit model
model.fit(X_train,Y_train)
```

Out[46]:
```
RandomForestClassifier()
```

In [47]:
```
RandomForestClassifier_score=model.score(X_valid,Y_valid)
print("Accuracy obtained by RandomForestClassifier_model:",RandomForestClassifier_sc
```

```
Accuracy obtained by RandomForestClassifier_model: 28.974402155607947
```

In [48]:
```
from sklearn.linear_model import LogisticRegression
# Define model. Specify a number for random_state to ensure same results each run
model = LogisticRegression()
# Fit model
model.fit(X_train,Y_train)
```

Out[48]:
```
LogisticRegression()
```

In [49]:
```
LogisticRegression_score=model.score(X_valid,Y_valid)
print("Accuracy obtained by LogisticRegression_model:",LogisticRegression_score*100)
```

```
Accuracy obtained by LogisticRegression_model: 27.71977096665544
```