In [62]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("combined_data.csv",low_memory=False)
# To display the top 5 rows
df.head(5)
```

Out[62]:

	Q1_HH	Q1_MM	Q2	Q3	Q4	Q5	Q6	Q6A	Q6A_OT	Q6B	...	Q1314	Q1315	Q1400	Q1401	Q1402	Q1403	Q1404	Q1405	Q1406	Gende
0	14	40	23	1	1.0	NaN	3.0	2	NaN	13	...	2	2	1	2	2	2	NaN	NaN	NO	Femal
1	13	24	16	1	1.0	NaN	3.0	2	NaN	13	...	2	2	1	2	2	2	NaN	NaN	NO	Femal
2	16	2	13	1	1.0	NaN	3.0	2	NaN	13	...	2	2	1	2	2	2	NaN	NaN	NOTHING	Femal
3	13	8	18	1	1.0	NaN	3.0	2	NaN	13	...	2	2	1	2	1	2	NaN	NaN	NOTHING	Femal
4	12	26	24	1	2.0	3.0	NaN	2	NaN	13	...	2	2	1	2	2	2	NaN	NaN	GOVERNMENT SHOULD MAKE HEALTH PROVISIONS FOR US	Femal

5 rows × 611 columns

In [ ]:

In [ ]:

In [63]:

```
import pandas as pd

# Rename the columns
df = df.rename(columns={
    'Q2': 'Age',
    'Gender': 'Gender',
    'Q6B': 'Ethnicity',

    'Q39': 'PV_Witnessing_at_home',
    'Q43': 'PV_Witnessing_outside_home',
    'Q100A': 'PV1_punched_kicked_whipped_beat',
    'Q100B': 'PV1_choked_suffocated_drown_burned',
    'Q100C': 'PV1_threatened_knife_gun_otherweapon',
    'Q116A': 'PV2_PEER_VIOLENCE_punched_kicked_whipped_beat',
    'Q116B': 'PV2_choked_suffocated_drown_burned',
    'Q116C': 'PV2_threatened_knife_gun_otherweapon',
    'Q120': 'PV2_Relationshiptoyou',
    'Q128A': 'PV3_PARENTS_ADULTCAREGIVERS_OTHER_ADULT_RELATIVES_punched_kicked_whipped_beat',
    'Q128B': 'PV3_choked_suffocated_drown_burned',
    'Q128C': 'PV3_threatened_knife_gun_otherweapon',
    'Q138': 'PV3_Relationshiptoyou',
    'Q142A': 'PV4_ADULTS_IN_THE_NEIGHBORHOOD_punched_kicked_whipped_beat',
    'Q142B': 'PV4_choked_suffocated_drown_burned',
    'Q142C': 'PV4_threatened_knife_gun_otherweapon',
    'Q151': 'PV4_Relationshiptoyou',
    'Q155': 'Knowledge_of_Support_Resources_PV',
    'Q156': 'Tried_Seeking_Help_PV',
    'Q157': 'Main_Reason_for_Not_Seeking_Help_PV',
    'Q158': 'Did_you_receive_help_PV',
    'Q300A': 'EV1_EMOTIONAL_VIOLENCE_told_were_not_loved',
    'Q300B': 'EV1_said_wished_never_been_born_or_were_dead',
    'Q300C': 'EV1_ridiculed_you_or_put_you_down',

    'Q700': 'Touching_sexual_way_without_permission',

    'Q900': 'SV_Physically_forced_and_succeed',
    'Q920': 'SV_Relationshiptoyou',
    'Q1000': 'SV_Pressured_through_harassment_threats_tricks',
    'Q1101': 'Knowledge_of_Support_Resources_SV',
    'Q1102': 'Tried_Seeking_Help_SV',
    'Q1103': 'Main_Reason_for_Not_Seeking_Help_SV',
    'Q1104': 'Did_you_receive_help_SV',
    'Q1304': 'Intentionally_hurt_yourself',
    'Q1305': 'Thought_about_killing_yourself',
    'Q1306': 'Tried_to_kill_yourself'
})

renamed_columns = [
    'Age', 'Gender', 'Ethnicity','PV_Witnessing_at_home', 'PV_Witnessing_outside_home',
    'PV1_punched_kicked_whipped_beat', 'PV1_choked_suffocated_drown_burned', 'PV1_threatened_knife_gun_otherweapon',
    'PV2_PEER_VIOLENCE_punched_kicked_whipped_beat',
    'PV2_choked_suffocated_drown_burned', 'PV2_threatened_knife_gun_otherweapon',
    'PV2_Relationshiptoyou', 'PV3_PARENTS_ADULTCAREGIVERS_OTHER_ADULT_RELATIVES_punched_kicked_whipped_beat',
    'PV3_choked_suffocated_drown_burned', 'PV3_threatened_knife_gun_otherweapon','PV3_Relationshiptoyou',
    'PV4_ADULTS_IN_THE_NEIGHBORHOOD_punched_kicked_whipped_beat', 'PV4_choked_suffocated_drown_burned',
    'PV4_threatened_knife_gun_otherweapon','PV4_Relationshiptoyou',
    'Knowledge_of_Support_Resources_PV', 'Tried_Seeking_Help_PV', 'Main_Reason_for_Not_Seeking_Help_PV', 'Did_you_receive_hel
    'EV1_EMOTIONAL_VIOLENCE_told_were_not_loved', 'EV1_said_wished_never_been_born_or_were_dead', 'EV1_ridiculed_you_or_put_y
    'Touching_sexual_way_without_permission',
    'SV_Physically_forced_and_succeed','SV_Relationshiptoyou','SV_Pressured_through_harassment_threats_tricks',
    'Knowledge_of_Support_Resources_SV', 'Tried_Seeking_Help_SV', 'Main_Reason_for_Not_Seeking_Help_SV', 'Did_you_receive_hel
    'Intentionally_hurt_yourself', 'Thought_about_killing_yourself', 'Tried_to_kill_yourself'
]

# Drop columns that are not in the list of renamed columns
df = df.drop(columns=[col for col in df.columns if col not in renamed_columns])
# Save cleaned and renamed dataset to a new file
df
```

Out[63]:

	Age	Ethnicity	PV_Witnessing_at_home	PV_Witnessing_outside_home	PV1_punched_kicked_whipped_beat	PV1_choked_suffocated_drown_burned	PV
0	23	13	2	1	1.0	2.0	
1	16	13	1	2	NaN	NaN	
2	13	13	1	1	NaN	NaN	
3	18	13	1	4	2.0	2.0	
4	24	13	1	4	2.0	2.0	
...	...	...	...	...	...	...	
4198	22	10	1	2	NaN	NaN	
4199	20	10	1	1	2.0	2.0	
4200	19	10	1	1	NaN	NaN	
4201	24	10	1	1	NaN	NaN	
4202	24	10	1	1	NaN	NaN	

4203 rows × 38 columns

In [64]:
```
df.dtypes
```

Out[64]:
```
Age                                                                  int64
Ethnicity                                                            int64
PV_Witnessing_at_home                                                int64
PV_Witnessing_outside_home                                           int64
PV1_punched_kicked_whipped_beat                                    float64
PV1_choked_suffocated_drown_burned                                 float64
PV1_threatened_knife_gun_otherweapon                               float64
PV2_PEER_VIOLENCE_punched_kicked_whipped_beat                        int64
PV2_choked_suffocated_drown_burned                                   int64
PV2_threatened_knife_gun_otherweapon                                 int64
PV2_Relationshiptoyou                                              float64
PV3_PARENTS_ADULTCAREGIVERS_OTHER_ADULT_RELATIVES_punched_kicked_whipped_beat    int64
PV3_choked_suffocated_drown_burned                                   int64
PV3_threatened_knife_gun_otherweapon                                 int64
PV3_Relationshiptoyou                                              float64
PV4_ADULTS_IN_THE_NEIGHBORHOOD_punched_kicked_whipped_beat            int64
PV4_choked_suffocated_drown_burned                                   int64
PV4_threatened_knife_gun_otherweapon                                 int64
PV4_Relationshiptoyou                                              float64
Knowledge_of_Support_Resources_PV                                  float64
Tried_Seeking_Help_PV                                              float64
Main_Reason_for_Not_Seeking_Help_PV                                float64
Did_you_receive_help_PV                                            float64
EV1_EMOTIONAL_VIOLENCE_told_were_not_loved                           int64
EV1_said_wished_never_been_born_or_were_dead                         int64
EV1_ridiculed_you_or_put_you_down                                    int64
Touching_sexual_way_without_permission                               int64
SV_Physically_forced_and_succeed                                     int64
SV_Pressured_through_harassment_threats_tricks                       int64
SV_Relationshiptoyou                                               float64
Knowledge_of_Support_Resources_SV                                  float64
Tried_Seeking_Help_SV                                              float64
Main_Reason_for_Not_Seeking_Help_SV                                float64
Did_you_receive_help_SV                                            float64
Intentionally_hurt_yourself                                          int64
Thought_about_killing_yourself                                       int64
Tried_to_kill_yourself                                             float64
Gender                                                              object
dtype: object
```

In [65]:
```
# null counts
print("Remaining Null Counts for each column:")
print(df.isnull().sum())
```

```
Remaining Null Counts for each column:
Age                                                                    0
Ethnicity                                                              0
PV_Witnessing_at_home                                                  0
PV_Witnessing_outside_home                                             0
PV1_punched_kicked_whipped_beat                                     1884
PV1_choked_suffocated_drown_burned                                  1884
PV1_threatened_knife_gun_otherweapon                                1884
PV2_PEER_VIOLENCE_punched_kicked_whipped_beat                          0
PV2_choked_suffocated_drown_burned                                     0
PV2_threatened_knife_gun_otherweapon                                   0
PV2_Relationshiptoyou                                               3432
PV3_PARENTS_ADULTCAREGIVERS_OTHER_ADULT_RELATIVES_punched_kicked_whipped_beat    0
PV3_choked_suffocated_drown_burned                                     0
PV3_threatened_knife_gun_otherweapon                                   0
PV3_Relationshiptoyou                                               2483
PV4_ADULTS_IN_THE_NEIGHBORHOOD_punched_kicked_whipped_beat              0
PV4_choked_suffocated_drown_burned                                     0
PV4_threatened_knife_gun_otherweapon                                   0
PV4_Relationshiptoyou                                               2846
Knowledge_of_Support_Resources_PV                                   1666
Tried_Seeking_Help_PV                                               3592
Main_Reason_for_Not_Seeking_Help_PV                                 3691
Did_you_receive_help_PV                                             4104
EV1_EMOTIONAL_VIOLENCE_told_were_not_loved                             0
EV1_said_wished_never_been_born_or_were_dead                           0
EV1_ridiculed_you_or_put_you_down                                      0
Touching_sexual_way_without_permission                                 0
SV_Physically_forced_and_succeed                                       0
SV_Pressured_through_harassment_threats_tricks                         0
SV_Relationshiptoyou                                                3943
Knowledge_of_Support_Resources_SV                                   3110
Tried_Seeking_Help_SV                                               3961
Main_Reason_for_Not_Seeking_Help_SV                                 3999
Did_you_receive_help_SV                                             4165
Intentionally_hurt_yourself                                            0
Thought_about_killing_yourself                                         0
Tried_to_kill_yourself                                              4022
Gender                                                                 0
dtype: int64
```

In [66]:
```
#since there are many null values want to reduce and only remain with those of less null values
# Set a threshold for missing values
threshold = 0.4
# Drop rows with more than 20% missing values

# Calculate the percentage of missing values in each row
missing_percentages = df.isnull().mean(axis=1)

# Get the rows to drop
rows_to_drop = missing_percentages[missing_percentages > threshold].index

# Drop rows with high missing values
df = df.drop(index=rows_to_drop)

# Display remaining null counts for each column
print("Remaining Null Counts for each column:")
print(df.isnull().sum())
```

```
Remaining Null Counts for each column:
Age                                                                    0
Ethnicity                                                              0
PV_Witnessing_at_home                                                  0
PV_Witnessing_outside_home                                             0
PV1_punched_kicked_whipped_beat                                     1228
PV1_choked_suffocated_drown_burned                                  1228
PV1_threatened_knife_gun_otherweapon                                1228
PV2_PEER_VIOLENCE_punched_kicked_whipped_beat                          0
PV2_choked_suffocated_drown_burned                                     0
PV2_threatened_knife_gun_otherweapon                                   0
PV2_Relationshiptoyou                                               2776
PV3_PARENTS_ADULTCAREGIVERS_OTHER_ADULT_RELATIVES_punched_kicked_whipped_beat       0
PV3_choked_suffocated_drown_burned                                     0
PV3_threatened_knife_gun_otherweapon                                   0
PV3_Relationshiptoyou                                               1827
PV4_ADULTS_IN_THE_NEIGHBORHOOD_punched_kicked_whipped_beat              0
PV4_choked_suffocated_drown_burned                                     0
PV4_threatened_knife_gun_otherweapon                                   0
PV4_Relationshiptoyou                                               2190
Knowledge_of_Support_Resources_PV                                   1010
Tried_Seeking_Help_PV                                               2936
Main_Reason_for_Not_Seeking_Help_PV                                 3035
Did_you_receive_help_PV                                             3448
EV1_EMOTIONAL_VIOLENCE_told_were_not_loved                             0
EV1_said_wished_never_been_born_or_were_dead                           0
EV1_ridiculed_you_or_put_you_down                                      0
Touching_sexual_way_without_permission                                 0
SV_Physically_forced_and_succeed                                       0
SV_Pressured_through_harassment_threats_tricks                         0
SV_Relationshiptoyou                                                3287
Knowledge_of_Support_Resources_SV                                   2454
Tried_Seeking_Help_SV                                               3305
Main_Reason_for_Not_Seeking_Help_SV                                 3343
Did_you_receive_help_SV                                             3509
Intentionally_hurt_yourself                                            0
Thought_about_killing_yourself                                         0
Tried_to_kill_yourself                                              3366
Gender                                                                 0
dtype: int64
```

In [67]:
```
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
#getting the number of duplicate rows
# Dropping duplicate rows
df = df.drop_duplicates()

# Print the number of duplicate rows after dropping
print("Number of duplicate rows:", df.duplicated().sum())
```

```
number of duplicate rows:  (302, 38)
Number of duplicate rows: 0
```

In [68]:
```
# Replace missing values with 99 since 99
#in the questionarre meant dont know or declined instead of replcing with 0, or bfill ,ffill that would bring biased results
df.fillna(99, inplace=True)

# Display remaining null counts for each column
print("Remaining Null Counts for each column:")
print(df.isnull().sum())
```

```
Remaining Null Counts for each column:
Age                                                                  0
Ethnicity                                                            0
PV_Witnessing_at_home                                                0
PV_Witnessing_outside_home                                           0
PV1_punched_kicked_whipped_beat                                      0
PV1_choked_suffocated_drown_burned                                   0
PV1_threatened_knife_gun_otherweapon                                 0
PV2_PEER_VIOLENCE_punched_kicked_whipped_beat                        0
PV2_choked_suffocated_drown_burned                                   0
PV2_threatened_knife_gun_otherweapon                                 0
PV2_Relationshiptoyou                                                0
PV3_PARENTS_ADULTCAREGIVERS_OTHER_ADULT_RELATIVES_punched_kicked_whipped_beat    0
PV3_choked_suffocated_drown_burned                                   0
PV3_threatened_knife_gun_otherweapon                                 0
PV3_Relationshiptoyou                                                0
PV4_ADULTS_IN_THE_NEIGHBORHOOD_punched_kicked_whipped_beat            0
PV4_choked_suffocated_drown_burned                                   0
PV4_threatened_knife_gun_otherweapon                                 0
PV4_Relationshiptoyou                                                0
Knowledge_of_Support_Resources_PV                                    0
Tried_Seeking_Help_PV                                                0
Main_Reason_for_Not_Seeking_Help_PV                                  0
Did_you_receive_help_PV                                              0
EV1_EMOTIONAL_VIOLENCE_told_were_not_loved                           0
EV1_said_wished_never_been_born_or_were_dead                         0
EV1_ridiculed_you_or_put_you_down                                    0
Touching_sexual_way_without_permission                               0
SV_Physically_forced_and_succeed                                     0
SV_Pressured_through_harassment_threats_tricks                       0
SV_Relationshiptoyou                                                 0
Knowledge_of_Support_Resources_SV                                    0
Tried_Seeking_Help_SV                                                0
Main_Reason_for_Not_Seeking_Help_SV                                  0
Did_you_receive_help_SV                                              0
Intentionally_hurt_yourself                                          0
Thought_about_killing_yourself                                       0
Tried_to_kill_yourself                                               0
Gender                                                               0
dtype: int64
```

In [ ]:

In [ ]:

In [69]:
```
df
```

Out[69]:

	Age	Ethnicity	PV_Witnessing_at_home	PV_Witnessing_outside_home	PV1_punched_kicked_whipped_beat	PV1_choked_suffocated_drown_burned	PV
0	23	13	2	1	1.0	2.0	
1	16	13	1	2	99.0	99.0	
3	18	13	1	4	2.0	2.0	
4	24	13	1	4	2.0	2.0	
6	14	13	1	3	2.0	2.0	
...	...	...	...	...	...	...	...
4195	20	10	1	4	2.0	2.0	
4196	21	8	1	1	99.0	99.0	
4197	15	10	1	1	99.0	99.0	
4198	22	10	1	2	99.0	99.0	
4200	19	10	1	1	99.0	99.0	

3245 rows × 38 columns

In [70]:
```
# Change the data type of specific columns to integer
df['PV1_punched_kicked_whipped_beat'] = df['PV1_punched_kicked_whipped_beat'].astype(int)
df['PV2_choked_suffocated_drown_burned'] = df['PV2_choked_suffocated_drown_burned'].astype(int)
df['PV2_threatened_knife_gun_otherweapon'] = df['PV2_threatened_knife_gun_otherweapon'].astype(int)

# Print the DataFrame to verify the changes
print(df)
```

```
      Age  Ethnicity  PV_Witnessing_at_home  PV_Witnessing_outside_home  \
0      23         13                      2                           1
1      16         13                      1                           2
3      18         13                      1                           4
4      24         13                      1                           4
6      14         13                      1                           3
...   ...        ...                    ...                         ...
4195   20         10                      1                           4
4196   21          8                      1                           1
4197   15         10                      1                           1
4198   22         10                      1                           2
4200   19         10                      1                           1

      PV1_punched_kicked_whipped_beat  PV1_choked_suffocated_drown_burned  \
0                                   1                                 2.0
1                                  99                                99.0
3                                   2                                 2.0
4                                   2                                 2.0
6                                   2                                 2.0
```

In [71]:
```
#1 mean Yes
#2 means No
# Create columns for physical violence
# Create columns for physical violence by intimate partner
df['Physical_Violence_by_Intimate_Partner'] = df.apply(lambda row: 1 if any(row[col] == 1 for col in ['PV1_punched_kicked_whi

# Add a new column for physical violence by peers
df['Physical_Violence_by_peer'] = df.apply(lambda row: 1 if any(row[col] == 1 for col in ['PV2_PEER_VIOLENCE_punched_kicked_w

# Add a new column for physical violence by parents, adult caregivers, or other adult relatives
df['Physical_Violence_by_Parents_Caregivers_Relatives'] = df.apply(lambda row: 1 if any(row[col] == 1 for col in ['PV3_PARENT

# Add a new column for physical violence by adults in the neighborhood
df['Physical_Violence_by_Adults_in_Neighborhood'] = df.apply(lambda row: 1 if any(row[col] == 1 for col in ['PV4_ADULTS_IN_TH

# Create a new column indicating presence of any type of physical violence
df['Physical_Violence'] = df.apply(lambda row: 1 if any(row[col] == 1 for col in ['Physical_Violence_by_Intimate_Partner', 'P

# Create columns for emotional violence
df['Emotional Violence'] = df.apply(lambda row: 1 if any(row[col] == 1 for col in ['EV1_EMOTIONAL_VIOLENCE_told_were_not_love

# Create columns for sexual violence
df['Sexual Violence'] = df.apply(lambda row: 1 if any(row[col] == 1 for col in ['Touching_sexual_way_without_permission','SV_
```

In [72]:
```
df
```

Out[72]:

	Age	Ethnicity	PV_Witnessing_at_home	PV_Witnessing_outside_home	PV1_punched_kicked_whipped_beat	PV1_choked_suffocated_drown_burned	PV
0	23	13	2	1	1	2.0	
1	16	13	1	2	99	99.0	
3	18	13	1	4	2	2.0	
4	24	13	1	4	2	2.0	
6	14	13	1	3	2	2.0	
...	...	...	...	...	...	...	
4195	20	10	1	4	2	2.0	
4196	21	8	1	1	99	99.0	
4197	15	10	1	1	99	99.0	
4198	22	10	1	2	99	99.0	
4200	19	10	1	1	99	99.0	

3245 rows × 45 columns

In [73]:
```
#here 1.0 means Yes ,2.0 means No
# Add a new column named 'Knowledge_of_Support_Resources' based on the condition
df['Knowledge_of_Support_Resources'] = [1.0 if any(row[col] == 1.0 for col in ['Knowledge_of_Support_Resources_PV', 'Knowledg

# Add a new column named 'Tried_Seeking_Help' based on the condition
df['Tried_Seeking_Help'] = [1.0 if any(row[col] == 1.0 for col in ['Tried_Seeking_Help_PV', 'Tried_Seeking_Help_SV']) else 2.

# Add a new column named 'Did_you_receive_help' based on the condition
df['Did_you_receive_help'] = [1.0 if any(row[col] == 1.0 for col in ['Did_you_receive_help_PV', 'Did_you_receive_help_SV']) e
```

In [74]:
```
df.head(90)
```

Out[74]:

	Age	Ethnicity	PV_Witnessing_at_home	PV_Witnessing_outside_home	PV1_punched_kicked_whipped_beat	PV1_choked_suffocated_drown_burned	PV1_
0	23	13	2	1	1	2.0	
1	16	13	1	2	99	99.0	
3	18	13	1	4	2	2.0	
4	24	13	1	4	2	2.0	
6	14	13	1	3	2	2.0	
...	...	...	...	...	...	...	
90	18	2	1	4	2	2.0	
91	20	6	1	1	1	2.0	
92	20	6	3	1	2	2.0	
93	24	2	1	99	2	2.0	
94	24	2	1	99	1	2.0	

90 rows × 48 columns

In [ ]:

In [ ]:

In [75]:
```
# Define a function to map the values for witnessing violence at home
def map_home_witnessing_value(value):
    if value in [2, 3, 4]:  # Witnessed violence at home
        return 1
    elif value == 1 or value == 99:  # Never witnessed or Don't know
        return 0
    else:
        return None  # Handle other cases if necessary

# Define a function to map the values for witnessing violence in the neighborhood
def map_neighborhood_witnessing_value(value):
    if value in [2, 3, 4]:  # Witnessed violence in the neighborhood
        return 1
    elif value == 1 or value == 99:  # Never witnessed or Don't know
        return 0
    else:
        return None  # Handle other cases if necessary

# Create new columns by applying the mapping functions row-wise
df['Witnessed_Violence_At_Home'] = [map_home_witnessing_value(row['PV_Witnessing_at_home']) for _, row in df.iterrows()]
df['Witnessed_Violence_In_Neighborhood'] = [map_neighborhood_witnessing_value(row['PV_Witnessing_outside_home']) for _, row i


df.head(14)
```

Out[75]:

	Age	Ethnicity	PV_Witnessing_at_home	PV_Witnessing_outside_home	PV1_punched_kicked_whipped_beat	PV1_choked_suffocated_drown_burned	PV1_
0	23	13	2	1	1	2.0	
1	16	13	1	2	99	99.0	
3	18	13	1	4	2	2.0	
4	24	13	1	4	2	2.0	
6	14	13	1	3	2	2.0	
7	24	13	1	1	2	2.0	
8	21	13	1	1	2	2.0	
9	24	13	1	99	2	2.0	
10	19	13	1	1	99	99.0	
11	24	13	1	1	1	2.0	
12	18	2	1	1	2	2.0	
13	17	13	3	3	99	99.0	
14	17	13	1	3	1	1.0	
15	17	13	3	4	99	99.0	

14 rows × 50 columns

In [76]:
```


#ideas on the eda documataion
#Percent of females and males  who reported experiencing types of sexual abuse
#Touching_sexual_way_without_permission
#SV_Physically_forced_and_succeed',#
#'SV_Pressured_through_harassment_threats_tricks'


#grapgh showing  percent distribution of different pyhsaicl violence across
#Physical_Violence_by_Intimate_Partner', 'Physical_Violence_by_peer',
#'Physical_Violence_by_Parents_Caregivers_Relatives','Physical_Violence_by_Adults_in_Neighborhood'

#gapgh showing distribution btw sexual violence,emotional,and pysical
# it coould also be filltered by female/male ,like a clustered grapgh

#Percent of females and males with grapgh  having Knowledge_of_Support_Resources,Tried_Seeking_Help,Did_you_receive_help colu
```

In [77]:
```
df.head(5)
```

Out[77]:

olence_by_Adults_in_Neighborhood	Physical_Violence	Emotional Violence	Sexual Violence	Knowledge_of_Support_Resources	Tried_Seeking_Help	Did_you_receive_help	Wit
1	1	0	1	2.0	2.0	2.0	
1	1	0	0	2.0	2.0	2.0	
0	1	0	1	2.0	2.0	2.0	
0	0	1	1	1.0	2.0	2.0	
0	1	1	1	1.0	1.0	1.0	

In [ ]:
```
```

In [78]:
```
df["Touching_sexual_way_without_permission"].value_counts()
```

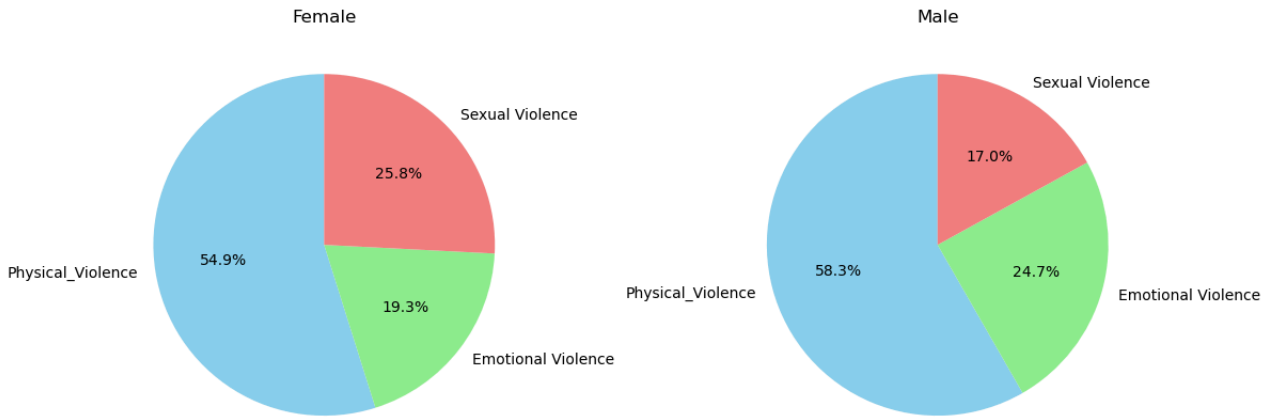
Out[78]:
```
Touching_sexual_way_without_permission
2     2478
1      715
99      52
Name: count, dtype: int64
```

In [79]:
```
df["Sexual Violence"].value_counts()
```

Out[79]:
```
Sexual Violence
0    2326
1     919
Name: count, dtype: int64
```

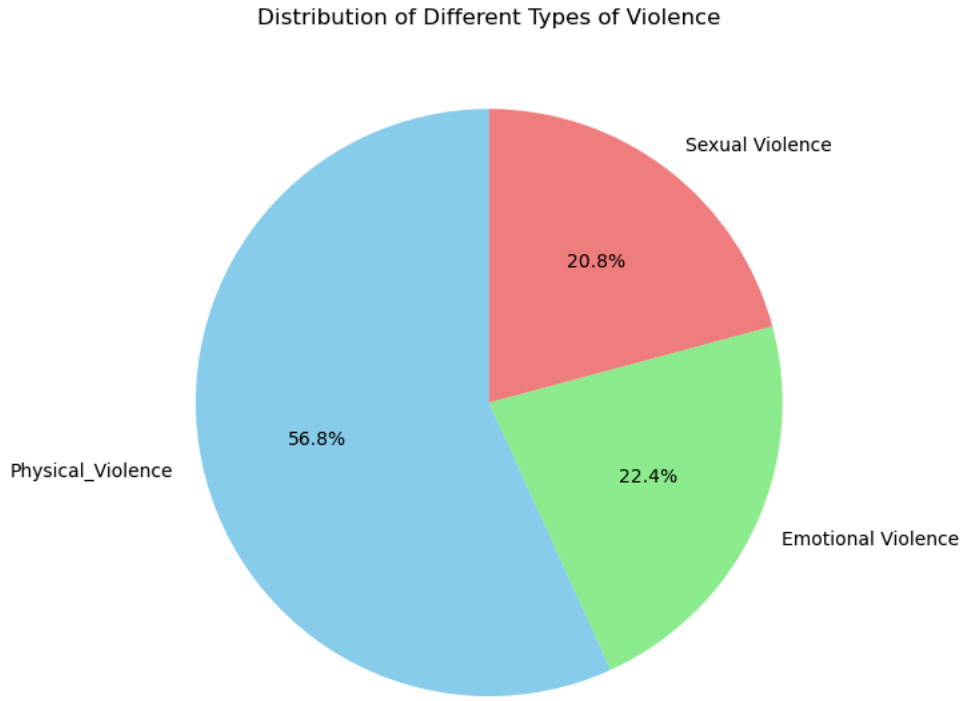
In [80]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Filter data for females and males
female_df = df[df['Gender'] == 'Female']
male_df = df[df['Gender'] == 'Male']

# Define the types of violence
violence_types = ["Physical_Violence", "Emotional Violence", "Sexual Violence"]

# Calculate total counts for females and males
female_counts = [female_df[violation_type].value_counts().loc[1] for violation_type in violence_types]
male_counts = [male_df[violation_type].value_counts().loc[1] for violation_type in violence_types]

# Create subplots for pie charts
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Plot pie chart for females
axs[0].pie(female_counts, labels=violence_types, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightgreen', 'lightcor
axs[0].set_title('Female')

# Plot pie chart for males
axs[1].pie(male_counts, labels=violence_types, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightgreen', 'lightcoral
axs[1].set_title('Male')

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```



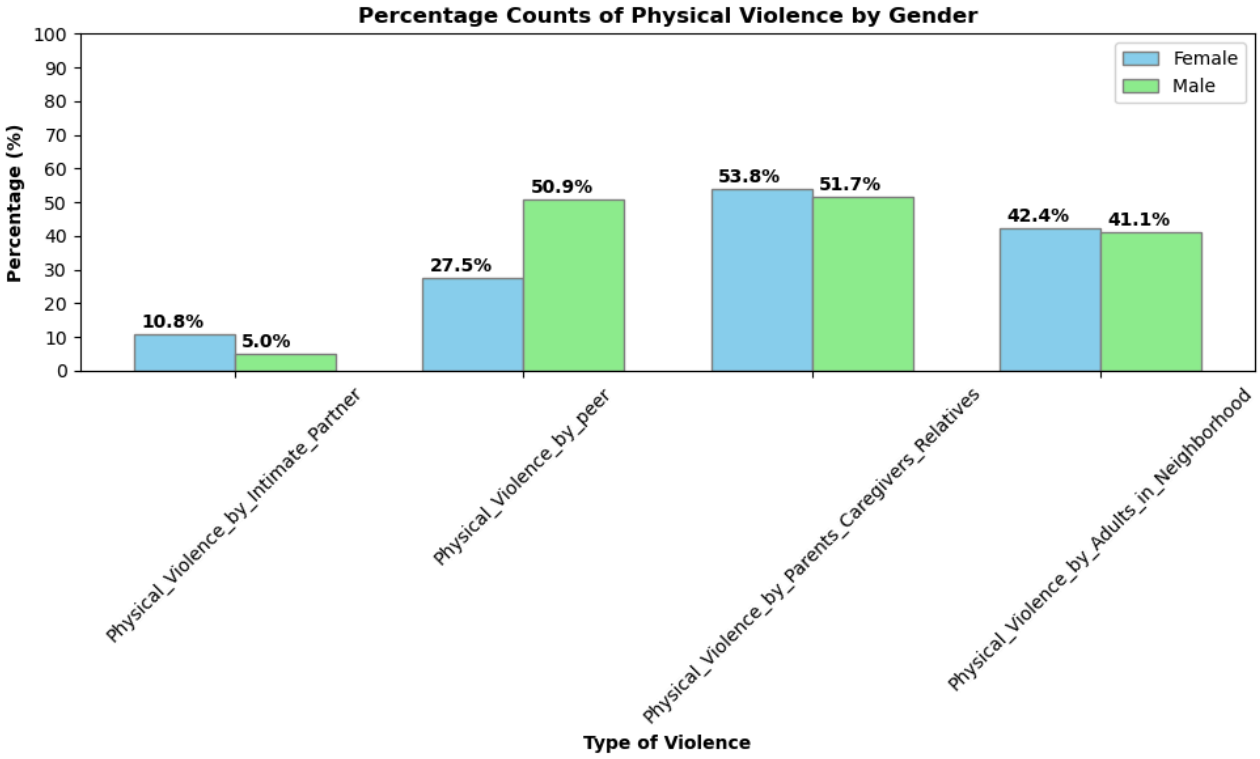
In [81]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Define the types of violence
violence_types = ["Physical_Violence", "Emotional Violence", "Sexual Violence"]

# Calculate total counts for all genders
total_counts = [df[violation_type].value_counts().loc[1] for violation_type in violence_types]

# Create a pie chart
plt.figure(figsize=(12, 6))

# Plot pie chart
plt.pie(total_counts, labels=violence_types, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightgreen', 'lightcoral',

# Add title
plt.title('Distribution of Different Types of Violence')

# Show plot
plt.tight_layout()
plt.show()
```



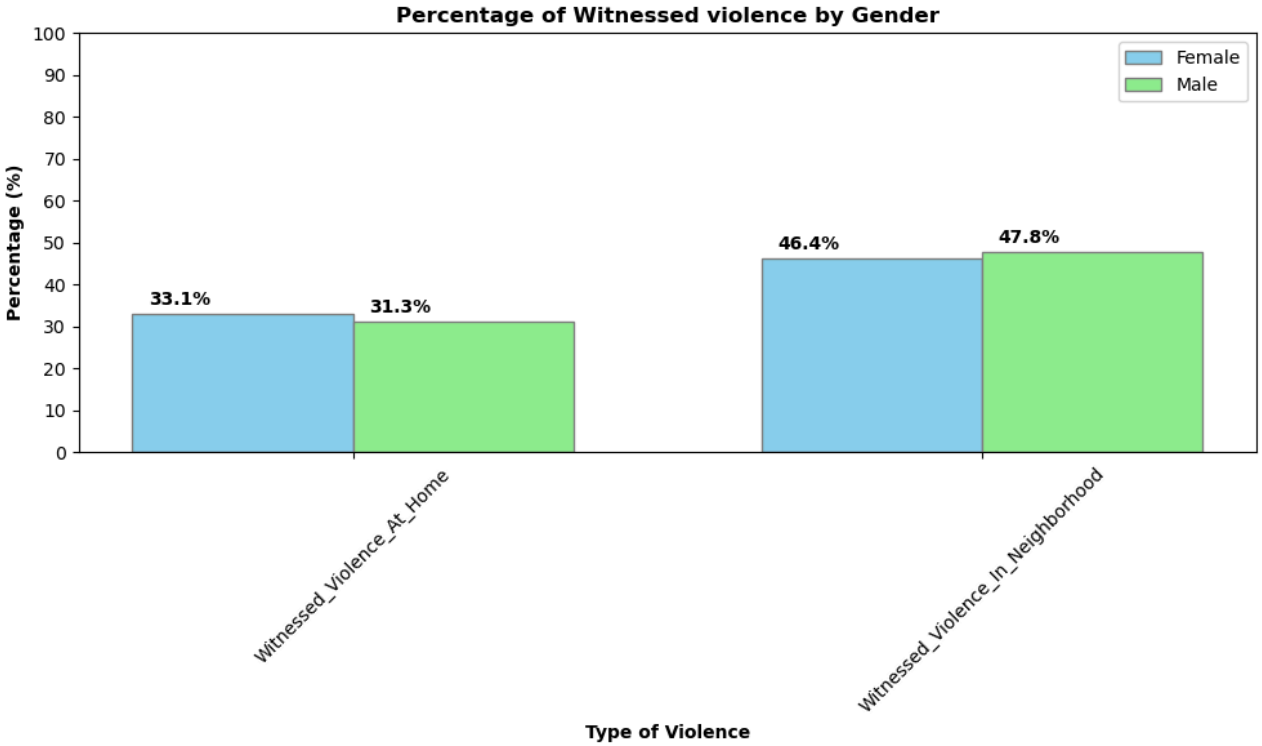
In [82]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Filter data for females and males
female_df = df[df['Gender'] == 'Female']
male_df = df[df['Gender'] == 'Male']

# Define the types of violence
violence_types = ["Physical_Violence_by_Intimate_Partner", "Physical_Violence_by_peer",
                  "Physical_Violence_by_Parents_Caregivers_Relatives", "Physical_Violence_by_Adults_in_Neighborhood"]

# Set the width of the bars
bar_width = 0.35

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Define the positions of the bars on the x-axis
x = range(len(violence_types))

# Calculate total counts
total_females = len(female_df)
total_males = len(male_df)

# Plot bars for females with percentages
female_counts = [female_df[violation_type].value_counts().loc[1] for violation_type in violence_types]
female_percentages = [round((count / total_females) * 100, 1) for count in female_counts]
ax.bar(x, female_percentages, bar_width, label='Female', color='skyblue', edgecolor='grey')

# Plot bars for males with percentages on top (clustered)
male_counts = [male_df[violation_type].value_counts().loc[1] for violation_type in violence_types]
male_percentages = [round((count / total_males) * 100, 1) for count in male_counts]
ax.bar([p + bar_width for p in x], male_percentages, bar_width, label='Male ', color='lightgreen', edgecolor='grey')


for i, perc in enumerate(female_percentages):
    plt.text(i - 0.15, perc + 2, str(perc) + '%', color='black', fontweight='bold')

for i, perc in enumerate(male_percentages):
    plt.text(i + 0.2, perc + 2, str(perc) + '%', color='black', fontweight='bold')

# Customize the plot
ax.set_xlabel('Type of Violence', fontweight='bold')
ax.set_ylabel('Percentage (%)', fontweight='bold')  # Updated label for percentages
ax.set_title('Percentage Counts of Physical Violence by Gender', fontweight='bold')
ax.set_xticks([p + (bar_width / 2) for p in x])  # Center the x-axis labels between bars
ax.set_xticklabels(violence_types, rotation=45)
ax.legend()

# Set y-axis limits and ticks (adjust based on percentage values)
plt.ylim(0, 100)  # Assuming percentages are all within 0-100 range
plt.yticks(range(0, 101, 10))

# Show plot
plt.tight_layout()
plt.show()
```


Percentage Counts of Physical Violence by Gender
Female
Male
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
10.8%
5.0%
27.5%
50.9%
53.8%
51.7%
42.4%
41.1%
Physical_Violence_by_Intimate_Partner
Physical_Violence_by_peer
Physical_Violence_by_Parents_Caregivers_Relatives
Physical_Violence_by_Adults_in_Neighborhood
Type of Violence

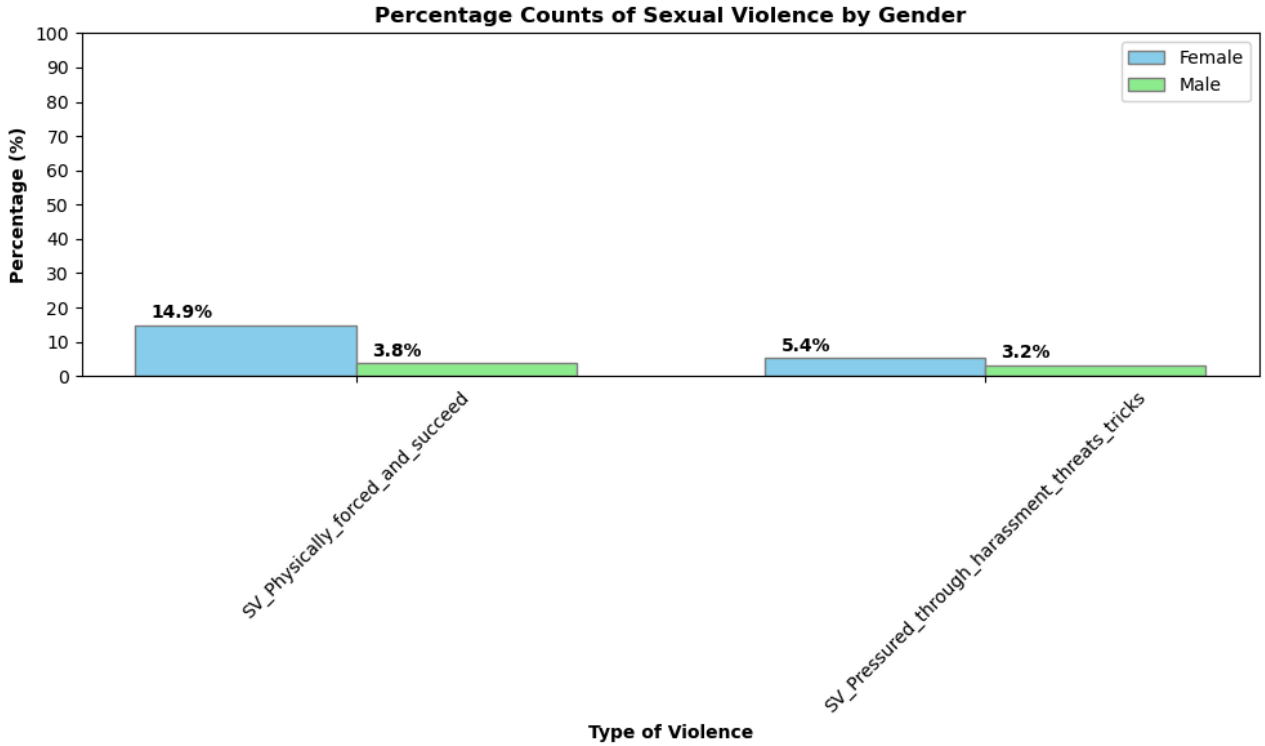
In [83]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Filter data for females and males
female_df = df[df['Gender'] == 'Female']
male_df = df[df['Gender'] == 'Male']

# Define the types of violence
violence_types = ["Witnessed_Violence_At_Home", "Witnessed_Violence_In_Neighborhood"]

# Set the width of the bars
bar_width = 0.35

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Define the positions of the bars on the x-axis
x = range(len(violence_types))

# Calculate total counts
total_females = len(female_df)
total_males = len(male_df)

# Plot bars for females with percentages
female_counts = [female_df[violation_type].value_counts().loc[1] for violation_type in violence_types]
female_percentages = [round((count / total_females) * 100, 1) for count in female_counts]
ax.bar(x, female_percentages, bar_width, label='Female', color='skyblue', edgecolor='grey')

# Plot bars for males with percentages on top (clustered)
male_counts = [male_df[violation_type].value_counts().loc[1] for violation_type in violence_types]
male_percentages = [round((count / total_males) * 100, 1) for count in male_counts]
ax.bar([p + bar_width for p in x], male_percentages, bar_width, label='Male', color='lightgreen', edgecolor='grey')


for i, perc in enumerate(female_percentages):
    plt.text(i - 0.15, perc + 2, str(perc) + '%', color='black', fontweight='bold')

for i, perc in enumerate(male_percentages):
    plt.text(i + 0.2, perc + 2, str(perc) + '%', color='black', fontweight='bold')

# Customize the plot
ax.set_xlabel('Type of Violence', fontweight='bold')
ax.set_ylabel('Percentage (%)', fontweight='bold')  # Updated label for percentages
ax.set_title('Percentage of Witnessed violence by Gender', fontweight='bold')
ax.set_xticks([p + (bar_width / 2) for p in x])  # Center the x-axis labels between bars
ax.set_xticklabels(violence_types, rotation=45)
ax.legend()

# Set y-axis limits and ticks (adjust based on percentage values)
plt.ylim(0, 100)  # Assuming percentages are all within 0-100 range
plt.yticks(range(0, 101, 10))

# Show plot
plt.tight_layout()
plt.show()
```


Percentage of Witnessed violence by Gender
Female
Male
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
33.1%
31.3%
46.4%
47.8%
Witnessed_Violence_At_Home
Witnessed_Violence_In_Neighborhood
Type of Violence

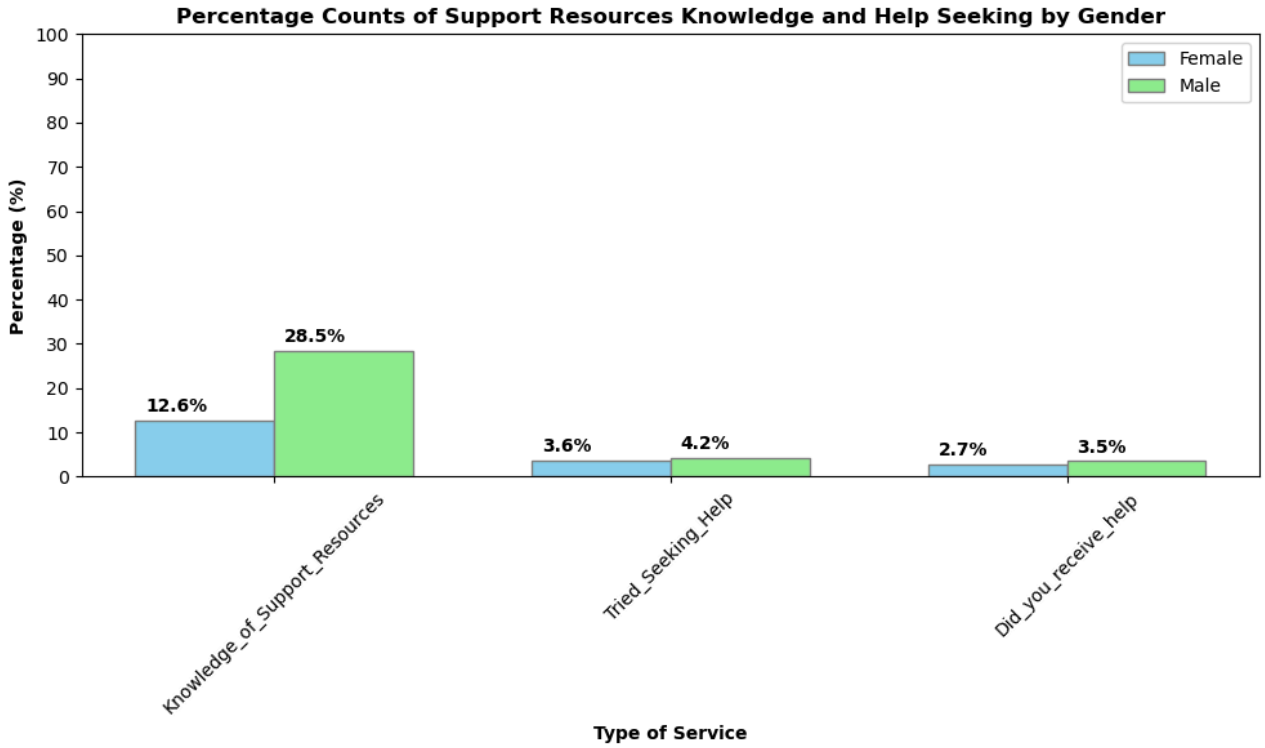
In [84]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Filter data for females and males
female_df = df[df['Gender'] == 'Female']
male_df = df[df['Gender'] == 'Male']

# Define the types of violence
violence_types = ["SV_Physically_forced_and_succeed", "SV_Pressured_through_harassment_threats_tricks"]

# Set the width of the bars
bar_width = 0.35

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Define the positions of the bars on the x-axis
x = range(len(violence_types))

# Calculate total counts
total_females = len(female_df)
total_males = len(male_df)

# Plot bars for females with percentages
female_counts = [female_df[violation_type].value_counts().loc[1] for violation_type in violence_types]
female_percentages = [round((count / total_females) * 100, 1) for count in female_counts]
ax.bar(x, female_percentages, bar_width, label='Female', color='skyblue', edgecolor='grey')

# Plot bars for males with percentages on top (clustered)
male_counts = [male_df[violation_type].value_counts().loc[1] for violation_type in violence_types]
male_percentages = [round((count / total_males) * 100, 1) for count in male_counts]
ax.bar([p + bar_width for p in x], male_percentages, bar_width, label='Male', color='lightgreen', edgecolor='grey')


for i, perc in enumerate(female_percentages):
    plt.text(i - 0.15, perc + 2, str(perc) + '%', color='black', fontweight='bold')

for i, perc in enumerate(male_percentages):
    plt.text(i + 0.2, perc + 2, str(perc) + '%', color='black', fontweight='bold')

# Customize the plot
ax.set_xlabel('Type of Violence', fontweight='bold')
ax.set_ylabel('Percentage (%)', fontweight='bold')  # Updated label for percentages
ax.set_title('Percentage Counts of Sexual Violence by Gender', fontweight='bold')
ax.set_xticks([p + (bar_width / 2) for p in x])  # Center the x-axis labels between bars
ax.set_xticklabels(violence_types, rotation=45)
ax.legend()

# Set y-axis limits and ticks (adjust based on percentage values)
plt.ylim(0, 100)  # Assuming percentages are all within 0-100 range
plt.yticks(range(0, 101, 10))

# Show plot
plt.tight_layout()
plt.show()
```


Percentage Counts of Sexual Violence by Gender
Female
Male
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
14.9%
3.8%
5.4%
3.2%
SV_Physically_forced_and_succeed
SV_Pressured_through_harassment_threats_tricks
Type of Violence

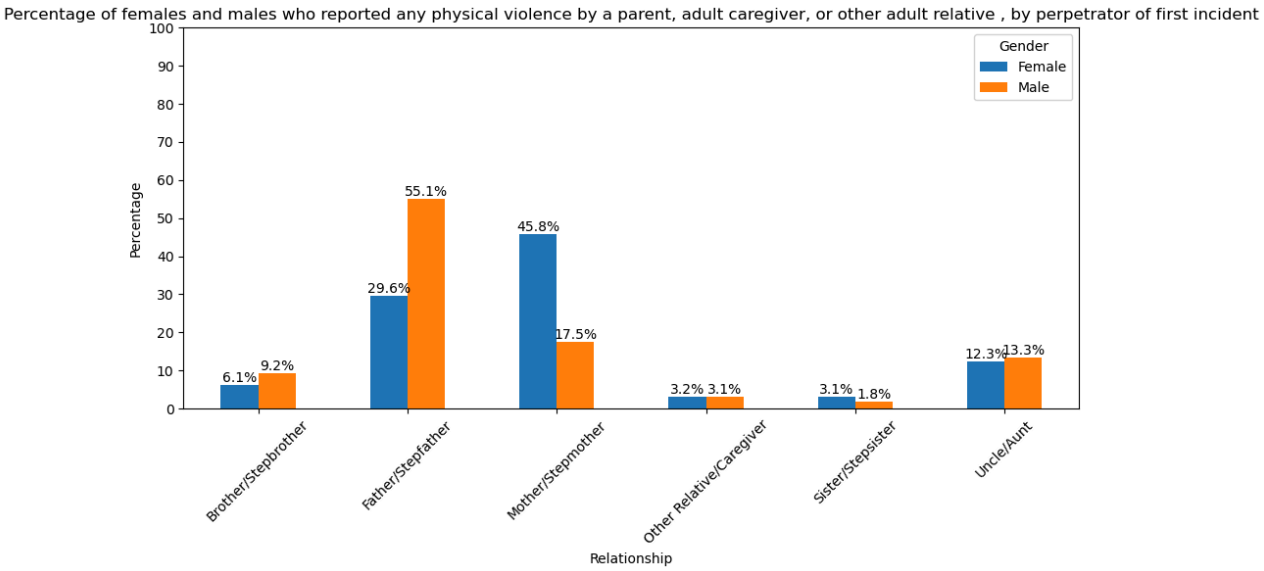
In [85]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Filter data for females and males
female_df = df[df['Gender'] == 'Female']
male_df = df[df['Gender'] == 'Male']

# Define the types of services
services = ["Knowledge_of_Support_Resources", "Tried_Seeking_Help", "Did_you_receive_help"]

# Set the width of the bars
bar_width = 0.35

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Define the positions of the bars on the x-axis
x = range(len(services))

# Calculate total counts
total_females = len(female_df)
total_males = len(male_df)

# Plot bars for females with percentages
female_counts = [female_df[service].value_counts().loc[1] for service in services]
female_percentages = [round((count / total_females) * 100, 1) for count in female_counts]
ax.bar(x, female_percentages, bar_width, label='Female', color='skyblue', edgecolor='grey')

# Plot bars for males with percentages on top (clustered)
male_counts = [male_df[service].value_counts().loc[1] for service in services]
male_percentages = [round((count / total_males) * 100, 1) for count in male_counts]
ax.bar([p + bar_width for p in x], male_percentages, bar_width, label='Male', color='lightgreen', edgecolor='grey')

# Annotate each bar with its corresponding percentage
for i, perc in enumerate(female_percentages):
    plt.text(i - 0.15, perc + 2, str(perc) + '%', color='black', fontweight='bold')

for i, perc in enumerate(male_percentages):
    plt.text(i + 0.2, perc + 2, str(perc) + '%', color='black', fontweight='bold')

# Customize the plot
ax.set_xlabel('Type of Service', fontweight='bold')
ax.set_ylabel('Percentage (%)', fontweight='bold')  # Updated label for percentages
ax.set_title('Percentage Counts of Support Resources Knowledge and Help Seeking by Gender', fontweight='bold')
ax.set_xticks([p + (bar_width / 2) for p in x])  # Center the x-axis labels between bars
ax.set_xticklabels(services, rotation=45)
ax.legend()

# Set y-axis limits and ticks (adjust based on percentage values)
plt.ylim(0, 100)  # Assuming percentages are all within 0-100 range
plt.yticks(range(0, 101, 10))

# Show plot
plt.tight_layout()
plt.show()
```


Percentage Counts of Support Resources Knowledge and Help Seeking by Gender
Female
Male
Percentage (%)
100
90
80
70
60
50
40
30
20
10
0
12.6%
28.5%
3.6%
4.2%
2.7%
3.5%
Knowledge_of_Support_Resources
Tried_Seeking_Help
Did_you_receive_help
Type of Service

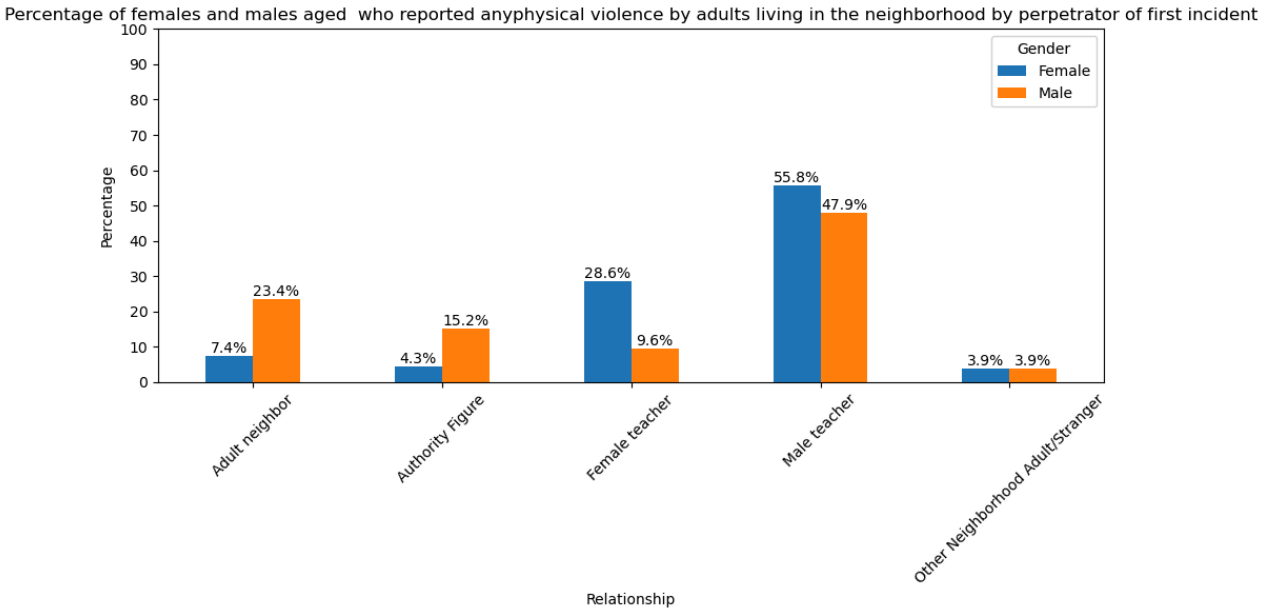
In [96]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Mapping dictionary for relationships
relationship_mapping = {
    1: 'Father/Stepfather',
    2: 'Father/Stepfather',
    3: 'Brother/Stepbrother',
    4: 'Brother/Stepbrother',
    5: 'Uncle/Aunt',
    6: 'Mother/Stepmother',
    7: 'Mother/Stepmother',
    8: 'Sister/Stepsister',
    9: 'Sister/Stepsister',
    10: 'Uncle/Aunt',
    77: 'Other Relative/Caregiver',
    88: 'Other Relative/Caregiver'
}

# Assuming df is your DataFrame

# Filter data for females and males excluding the value 99.0
female_pv_counts = df[(df['Gender'] == 'Female') & (df['PV3_Relationshiptoyou'] != 99.0)]['PV3_Relationshiptoyou'].map(relati
male_pv_counts = df[(df['Gender'] == 'Male') & (df['PV3_Relationshiptoyou'] != 99.0)]['PV3_Relationshiptoyou'].map(relationsh

# Combine counts for related categories
combined_female_counts = female_pv_counts.groupby(level=0).sum()
combined_male_counts = male_pv_counts.groupby(level=0).sum()

# Calculate total counts for females and males
total_female_count = combined_female_counts.sum()
total_male_count = combined_male_counts.sum()

# Calculate percentages for females and males, rounded to one decimal place
female_pv_percentages = (combined_female_counts / total_female_count * 100).round(1)
male_pv_percentages = (combined_male_counts / total_male_count * 100).round(1)

# Combine percentages into a single DataFrame
combined_percentages = pd.concat([female_pv_percentages, male_pv_percentages], axis=1)
combined_percentages.columns = ['Female', 'Male']

# Plot clustered bar graph
ax = combined_percentages.plot(kind='bar', figsize=(10, 6))

# Annotate each bar with its corresponding percentage
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()),
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha='center', va='bottom')

# Customize the plot
plt.xlabel('Relationship')
plt.ylabel('Percentage')
plt.title('Percentage of females and males who reported any physical violence by a parent, adult caregiver, or other adult re
plt.legend(title='Gender')

# Set y-axis limits and ticks
plt.ylim(0, 100)  # Setting y-axis limits from 0 to 100
plt.yticks(range(0, 101, 10))  # Setting y-axis ticks every 10 units

# Show plot
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Percentage of females and males who reported any physical violence by a parent, adult caregiver, or other adult relative , by perpetrator of first incident
Gender
Female
Male
Percentage
Relationship
6.1%
9.2%
29.6%
55.1%
45.8%
17.5%
3.2%
3.1%
3.1%
1.8%
12.3%
13.3%
Brother/Stepbrother
Father/Stepfather
Mother/Stepmother
Other Relative/Caregiver
Sister/Stepsister
Uncle/Aunt

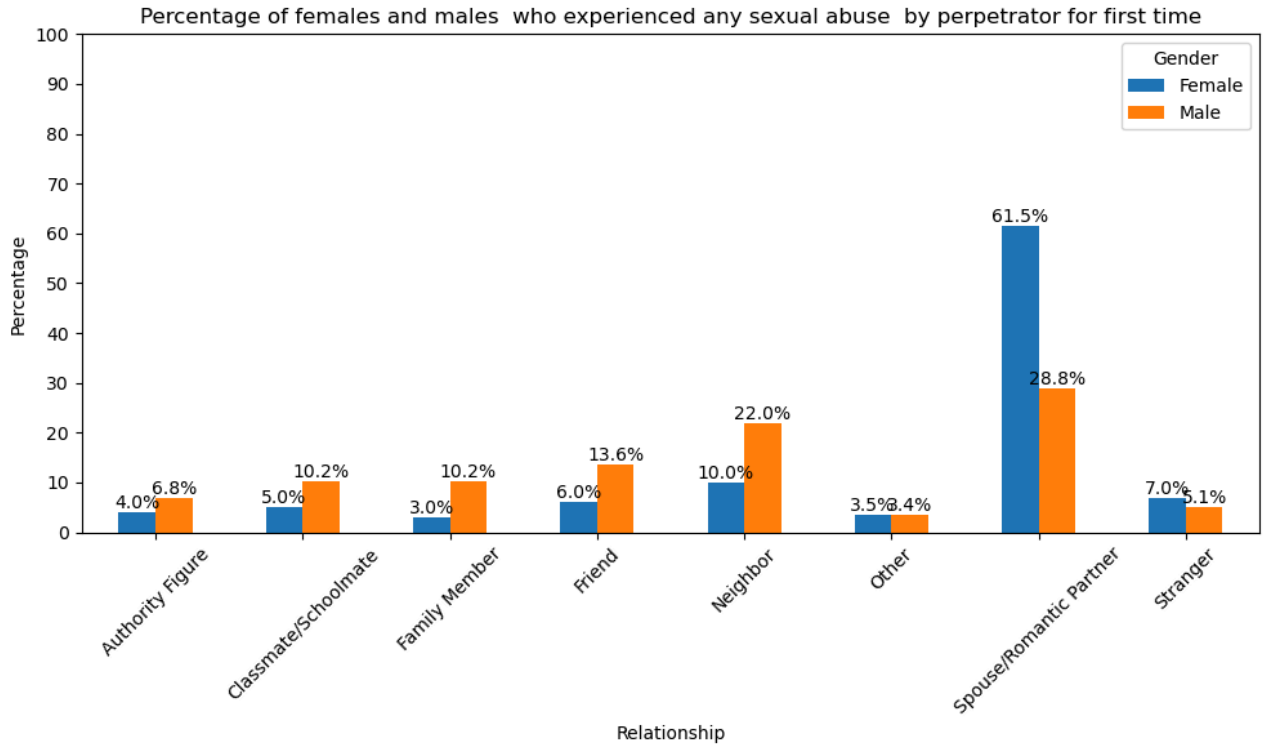
In [97]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Mapping dictionary for relationships
relationship_mapping = {
    1: 'Male teacher',
    2: 'Authority Figure',
    3: 'Authority Figure',
    4: 'Authority Figure',
    5: 'Authority Figure',
    6: 'Adult neighbor',
    7: 'Female teacher',
    8: 'Authority Figure',
    9: 'Authority Figure',
    10: 'Authority Figure',
    11: 'Authority Figure',
    12: 'Adult neighbor',
    77: 'Other Neighborhood Adult/Stranger',
    88: 'Other Neighborhood Adult/Stranger'
}


# Assuming df is your DataFrame

# Filter data for females and males excluding the value 99.0
female_pv_counts = df[(df['Gender'] == 'Female') & (df['PV4_Relationshiptoyou'] != 99.0)]['PV4_Relationshiptoyou'].map(relati
male_pv_counts = df[(df['Gender'] == 'Male') & (df['PV4_Relationshiptoyou'] != 99.0)]['PV4_Relationshiptoyou'].map(relationsh

# Combine counts for related categories
combined_female_counts = female_pv_counts.groupby(level=0).sum()
combined_male_counts = male_pv_counts.groupby(level=0).sum()

# Calculate total counts for females and males
total_female_count = combined_female_counts.sum()
total_male_count = combined_male_counts.sum()

# Calculate percentages for females and males, rounded to one decimal place
female_pv_percentages = (combined_female_counts / total_female_count * 100).round(1)
male_pv_percentages = (combined_male_counts / total_male_count * 100).round(1)

# Combine percentages into a single DataFrame
combined_percentages = pd.concat([female_pv_percentages, male_pv_percentages], axis=1)
combined_percentages.columns = ['Female', 'Male']

# Plot clustered bar graph
ax = combined_percentages.plot(kind='bar', figsize=(10, 6))

# Annotate each bar with its corresponding percentage
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()),
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha='center', va='bottom')

# Customize the plot
plt.xlabel('Relationship')
plt.ylabel('Percentage')
plt.title('Percentage of females and males aged  who reported anyphysical violence by adults living in the neighborhood by pe
plt.legend(title='Gender')

# Set y-axis limits and ticks
plt.ylim(0, 100)  # Setting y-axis limits from 0 to 100
plt.yticks(range(0, 101, 10))  # Setting y-axis ticks every 10 units

# Show plot
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Percentage of females and males aged who reported anyphysical violence by adults living in the neighborhood by perpetrator of first incident
Gender
Female
Male
Percentage
100
90
80
70
60
50
40
30
20
10
0
7.4%
23.4%
4.3%
15.2%
28.6%
9.6%
55.8%
47.9%
3.9%
3.9%
Adult neighbor
Authority Figure
Female teacher
Male teacher
Other Neighborhood Adult/Stranger
Relationship

In [94]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Mapping dictionary for relationships
relationship_mapping = {

    1: 'Spouse/Romantic Partner',
    2: 'Spouse/Romantic Partner',
    3: 'Family Member',
    4: 'Family Member',
    5: 'Family Member',
    6: 'Family Member',
    7: 'Classmate/Schoolmate',
    8: 'Authority Figure',
    9: 'Authority Figure',
    10: 'Authority Figure',
    11: 'Neighbor',
    12: 'Authority Figure',
    13: 'Friend',
    14: 'Stranger',
    15: 'Spouse/Romantic Partner',
    16: 'Spouse/Romantic Partner',
    17: 'Family Member',
    18: 'Family Member',
    19: 'Family Member',
    20: 'Family Member',
    21: 'Classmate/Schoolmate',
    22: 'Authority Figure',
    23: 'Authority Figure',
    24: 'Authority Figure',
    25: 'Neighbor',
    26: 'Authority Figure',
    27: 'Friend',
    28: 'Stranger',
    77: 'Other',
    88: 'Other',

}



# Assuming df is your DataFrame

# Filter data for females and males excluding the value 99.0
female_pv_counts = df[(df['Gender'] == 'Female') & (df['SV_Relationshiptoyou'] != 99.0)]['SV_Relationshiptoyou'].map(relation
male_pv_counts = df[(df['Gender'] == 'Male') & (df['SV_Relationshiptoyou'] != 99.0)]['SV_Relationshiptoyou'].map(relationship

# Combine counts for related categories
combined_female_counts = female_pv_counts.groupby(level=0).sum()
combined_male_counts = male_pv_counts.groupby(level=0).sum()

# Calculate total counts for females and males
total_female_count = combined_female_counts.sum()
total_male_count = combined_male_counts.sum()

# Calculate percentages for females and males, rounded to one decimal place
female_pv_percentages = (combined_female_counts / total_female_count * 100).round(1)
male_pv_percentages = (combined_male_counts / total_male_count * 100).round(1)

# Combine percentages into a single DataFrame
combined_percentages = pd.concat([female_pv_percentages, male_pv_percentages], axis=1)
combined_percentages.columns = ['Female', 'Male']

# Plot clustered bar graph
ax = combined_percentages.plot(kind='bar', figsize=(10, 6))

# Annotate each bar with its corresponding percentage
for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height()),
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha='center', va='bottom')

# Customize the plot
plt.xlabel('Relationship')
plt.ylabel('Percentage')
plt.title()
plt.legend(title='Gender')

# Set y-axis limits and ticks
plt.ylim(0, 100)  # Setting y-axis limits from 0 to 100
plt.yticks(range(0, 101, 10))  # Setting y-axis ticks every 10 units

# Show plot
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Percentage of females and males who experienced any sexual abuse by perpetrator for first time
Gender
Female
Male
Percentage
Relationship
Authority Figure
Classmate/Schoolmate
Family Member
Friend
Neighbor
Other
Spouse/Romantic Partner
Stranger
4.0%
6.8%
5.0%
10.2%
3.0%
10.2%
6.0%
13.6%
10.0%
22.0%
3.5%
3.4%
61.5%
28.8%
7.0%
5.1%

In [ ]:

In [ ]: