Survery analysis

step 1: import data & preprocessing

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
raw_data = pd.read_csv('INST600_surveyData_December 1.csv')
raw_data
```

Out[5]:

	StartDate	EndDate	Status	IPAddress	Prog
0	Start Date	End Date	Response Type	IP Address	Prog
1	{"ImportId":"startDate","timeZone":"America/De...	{"ImportId":"endDate","timeZone":"America/Denv...	{"ImportId":"status"}	{"ImportId":"ipAddress"}	{"ImportId":"progr
2	2024-11-18 14:49:23	2024-11-18 14:55:45	0	38.27.115.10	
3	2024-11-18 15:00:22	2024-11-18 15:05:26	0	38.27.115.13	
4	2024-11-18 15:27:53	2024-11-18 15:31:52	0	211.234.180.36	
...	...	...	...	...	
79	2024-11-19 06:53:47	2024-11-19 06:54:02	0	219.241.140.37	
80	2024-11-19 08:25:53	2024-11-19 08:26:17	0	218.38.60.225	
81	2024-11-21 05:54:56	2024-11-21 05:57:00	0	59.1.143.132	
82	2024-11-21 15:52:50	2024-11-21 15:53:07	0	100.15.135.73	
83	2024-11-23 08:43:22	2024-11-23 08:44:04	0	39.127.137.44	

84 rows × 115 columns

In [6]:
```
pd.set_option('display.max_columns', None)
```

Simply pre-processing

In [7]:
```
droped_df = raw_data.drop(index=1).drop(columns=['RecipientLastName', 'RecipientFirstName', 'RecipientEmail', 'ExternalReference',
droped_df
```

Out[7]:

	StartDate	EndDate	Progress	Duration (in seconds)	Finished	RecordedDate	ResponseId	UserLanguage	Q2	Q4	Q5	Q5_3_TEXT	
0	Start Date	End Date	Progress	Duration (in seconds)	Finished	Recorded Date	Response ID	User Language	Welcome to the research study! \n\n \n\nWe ar...	How old are you ? (Please enter numbers only.)	Where do you currently live? - Selected Choice	Where do you currently live? - Others - Text	W nation - Sel Cl
2	2024-11-18 14:49:23	2024-11-18 14:55:45	100	381	1	2024-11-18 14:55:45	R_5KoVTplMMg9fj05	EN	1	24	1	NaN	
3	2024-11-18 15:00:22	2024-11-18 15:05:26	100	303	1	2024-11-18 15:05:26	R_3jUTY1nOkJggB0T	KO	1	23	1	NaN	
4	2024-11-18 15:27:53	2024-11-18 15:31:52	100	238	1	2024-11-18 15:31:52	R_4xukSI6zhH9GGyd	KO	1	25	2	NaN	
5	2024-11-18 16:40:12	2024-11-18 16:45:29	100	317	1	2024-11-18 16:45:30	R_3ghcu3OtKty8n2w	KO	1	30	1	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	
79	2024-11-19 06:53:47	2024-11-19 06:54:02	3	15	0	2024-11-26 06:54:05	R_4TzUahtHibgnObT	KO	NaN	NaN	NaN	NaN	
80	2024-11-19 08:25:53	2024-11-19 08:26:17	6	23	0	2024-11-26 08:26:21	R_4veIRWRwoyocHFn	KO	1	NaN	NaN	NaN	
81	2024-11-21 05:54:56	2024-11-21 05:57:00	36	124	0	2024-11-28 05:57:04	R_4p5s9xxJlSDBxY5	KO	1	24	2	NaN	
82	2024-11-21 15:52:50	2024-11-21 15:53:07	3	16	0	2024-11-28 15:53:13	R_137VCif5ENICMia	EN	NaN	NaN	NaN	NaN	
83	2024-11-23 08:43:22	2024-11-23 08:44:04	6	41	0	2024-11-30 08:44:06	R_4GVwLb6grh4gXCH	KO	1	NaN	NaN	NaN	

83 rows × 105 columns

- Selecting **completed** survey reseponses

In [8]:
```
Finished_data = droped_df[droped_df['Finished'] == '1'].reset_index(drop=True)
Finished_data.head(3)
```

Out[8]:

	StartDate	EndDate	Progress	Duration (in seconds)	Finished	RecordedDate	ResponseId	UserLanguage	Q2	Q4	Q5	Q5_3_TEXT	Q6	Q6_3_TEXT	Q7	Q
0	2024-11-18 14:49:23	2024-11-18 14:55:45	100	381	1	2024-11-18 14:55:45	R_5KoVTplMMg9fj05	EN	1	24	1	NaN	2	NaN	2	
1	2024-11-18 15:00:22	2024-11-18 15:05:26	100	303	1	2024-11-18 15:05:26	R_3jUTY1nOkJggB0T	KO	1	23	1	NaN	2	NaN	2	
2	2024-11-18 15:27:53	2024-11-18 15:31:52	100	238	1	2024-11-18 15:31:52	R_4xukSI6zhH9GGyd	KO	1	25	2	NaN	2	NaN	1	

In [9]:
```
print(Finished_data.shape)
```

```
(53, 105)
```

A total of 82 responses have been recorded, of which 53 surveys have been completed.

step2: Basic analysis

- Demographic information

Filter only those who have agreed in the consent form

In [10]:
```
survey_df = Finished_data[Finished_data['Q2'] == '1']
print(survey_df.shape)
```

```
(52, 105)
```

-> A total of 52 responses were analyzed.

- age

In [11]:
```
survey_df.loc[:, "Q4"] = pd.to_numeric(survey_df["Q4"], errors="coerce")
survey_df["Q4"].describe()
```

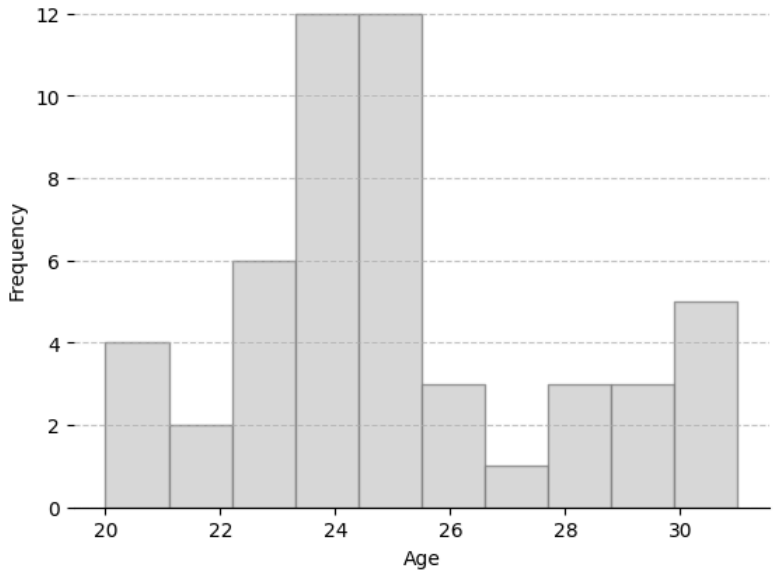
```
C:\Users\Samsung\AppData\Local\Temp\ipykernel_22288\1379385661.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  survey_df.loc[:, "Q4"] = pd.to_numeric(survey_df["Q4"], errors="coerce")
C:\Users\Samsung\AppData\Local\Temp\ipykernel_22288\1379385661.py:1: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To retain the old behavior, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  survey_df.loc[:, "Q4"] = pd.to_numeric(survey_df["Q4"], errors="coerce")
```

Out[11]:
```
count    51.000000
mean     25.098039
std       2.773120
min      20.000000
25%      24.000000
50%      25.000000
75%      26.000000
max      31.000000
Name: Q4, dtype: float64
```

In [12]:
```
print(survey_df["Q4"].mode()[0])  # 최빈값
print(survey_df["Q4"].median())  # 중앙값
```

```
24.0
25.0
```

In [13]:
```
# Histogram
plt.hist(survey_df["Q4"].dropna(), bins=10, edgecolor="Gray", color="lightgray", alpha=0.8)

plt.title("")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.grid(axis="y", linestyle="--", alpha=0.7)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



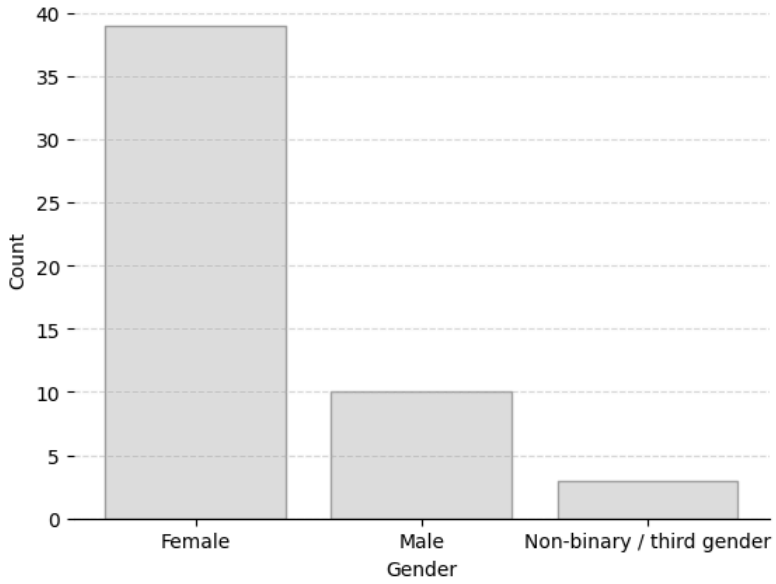
- sex

In [14]:
```
gender_map = {"1": "Male", "2": "Female", "3": "Non-binary / third gender", "4":"Prefer not to say", "5":'prefer to self-describe'
survey_df["Q7_category"] = survey_df["Q7"].map(gender_map)
survey_df["Q7_category"].value_counts()
```

```
C:\Users\Samsung\AppData\Local\Temp\ipykernel_22288\3425605542.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  survey_df["Q7_category"] = survey_df["Q7"].map(gender_map)
```

Out[14]:
```
Female                      39
Male                        10
Non-binary / third gender    3
Name: Q7_category, dtype: int64
```

In [15]:
```
gender_counts = {"Female": 39, "Male": 10, "Non-binary / third gender": 3}
plt.bar(gender_counts.keys(), gender_counts.values(), color='lightgray', edgecolor="gray", alpha=0.7)

plt.title("")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



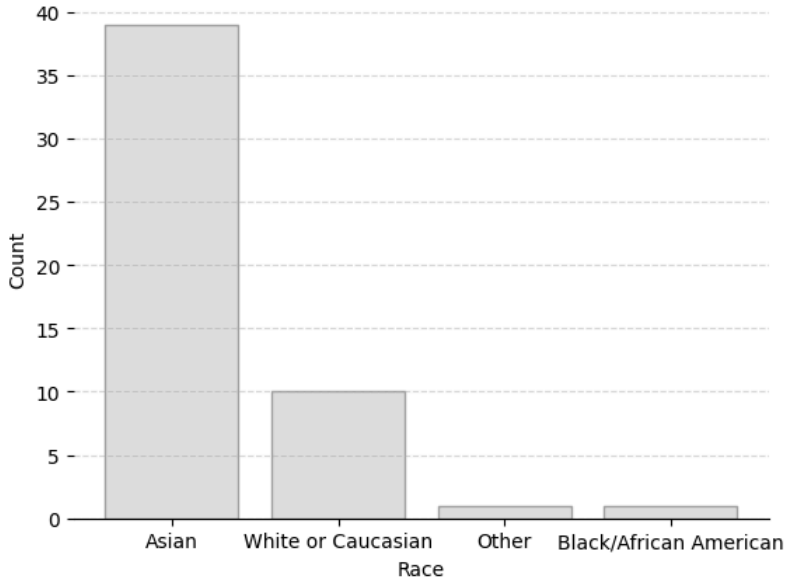
- Race

In [16]:
```
race_map = { "1": "White or Caucasian", "2": "Black or African American", "3": "American Indian/Native American or Alaska Native",
survey_df["Mapped_Race"] = survey_df["Q8"].map(race_map)
survey_df["Mapped_Race"].value_counts()
```

```
C:\Users\Samsung\AppData\Local\Temp\ipykernel_22288\1511279891.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  survey_df["Mapped_Race"] = survey_df["Q8"].map(race_map)
```

Out[16]:
```
Asian                          39
White or Caucasian             10
Other                           1
Black or African American       1
Name: Mapped_Race, dtype: int64
```

In [17]:
```
race_counts = {"Asian": 39, "White or Caucasian": 10, "Other": 1, 'Black/African American': 1}
plt.bar(race_counts.keys(), race_counts.values(), color='lightgray', edgecolor="gray", alpha=0.7)

plt.title("")
plt.xlabel("Race")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



- Education

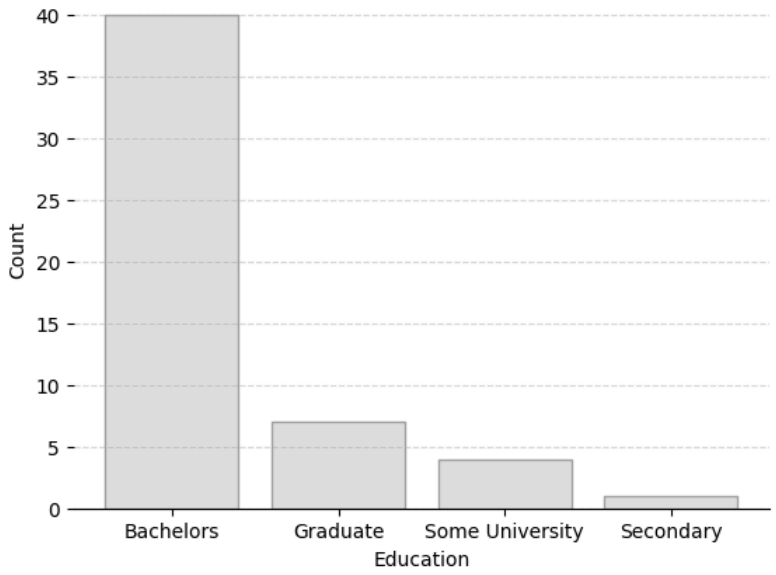
In [18]:
```
Education_map = { "16": "University - Bachelors Degree", "17": "Graduate or professional degree", "15": "Some University but no de
survey_df["Mapped_edu"] = survey_df["Q9"].map(Education_map)
survey_df["Mapped_edu"].value_counts()
```

```
C:\Users\Samsung\AppData\Local\Temp\ipykernel_22288\2006767361.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  survey_df["Mapped_edu"] = survey_df["Q9"].map(Education_map)
```

Out[18]:
```
University - Bachelors Degree      40
Graduate or professional degree     7
Some University but no degree       4
Secondary                           1
Name: Mapped_edu, dtype: int64
```

In [19]:
```
race_counts = {"Bachelors": 40, "Graduate": 7, "Some University": 4, 'Secondary': 1}
plt.bar(race_counts.keys(), race_counts.values(), color='lightgray', edgecolor="gray", alpha=0.7)

plt.title("")
plt.xlabel("Education")
```

```
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



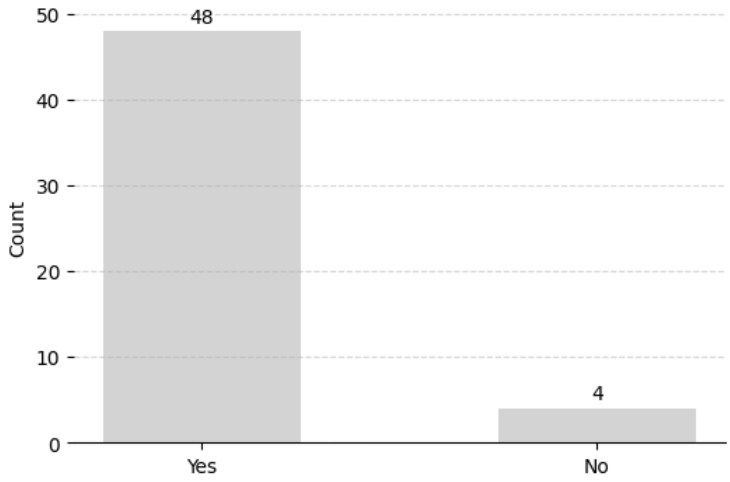
Step3: RQ1_anlysis

- 1) Have you ever used social media (Instagram, Facebook, X, Reddit, Youtube, etc.)?

In [20]:
```
mapping = { "1": "No", "2": "Yes"}
RQ1_1= survey_df["Q11"].map(mapping)
RQ1_1.value_counts()
```

Out[20]:
```
Yes    48
No      4
Name: Q11, dtype: int64
```

In [21]:
```
plt.figure(figsize=(6, 4))
plt.bar(RQ1_1.value_counts().index, RQ1_1.value_counts().values, color='lightgray',width=0.5)
plt.xticks(rotation=0)
plt.title("")
plt.xlabel("")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
for i, value in enumerate(RQ1_1.value_counts().values):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



- 1. How many hours per day do you spend on social media? (Please enter numbers)

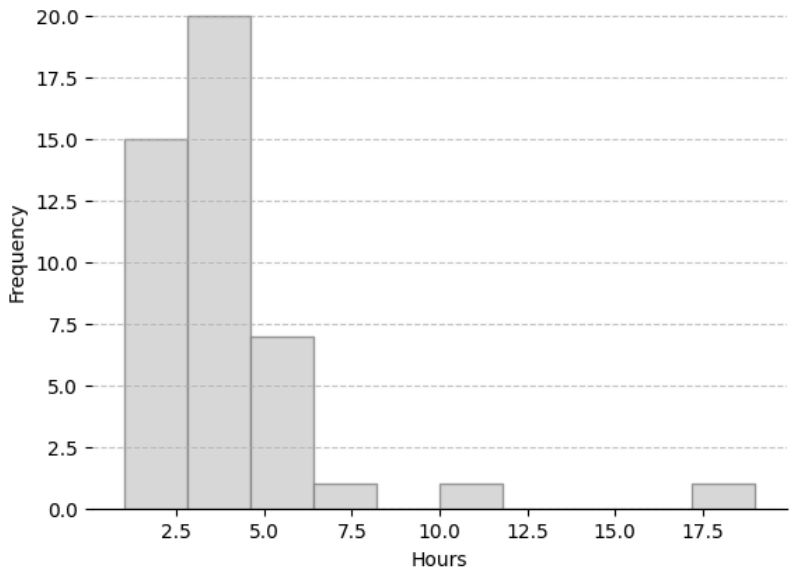
In [22]:
```
survey_df["Q12"].dropna().astype('int').describe()
```

Out[22]:
```
count    45.000000
mean      3.800000
std       2.880972
min       1.000000
25%       2.000000
50%       3.000000
75%       4.000000
max      19.000000
Name: Q12, dtype: float64
```

In [116…
```
60*0.8
```

Out[116]:
```
48.0
```

In [23]:
```
# Histogram
plt.hist(survey_df["Q12"].dropna().astype('int'), bins=10, edgecolor="Gray", color="lightgray", alpha=0.8)

plt.title("")
plt.xlabel("Hours")
plt.ylabel("Frequency")
plt.grid(axis="y", linestyle="--", alpha=0.7)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



- 1. Please select all the social media platforms you have used. (Please check all that apply.)

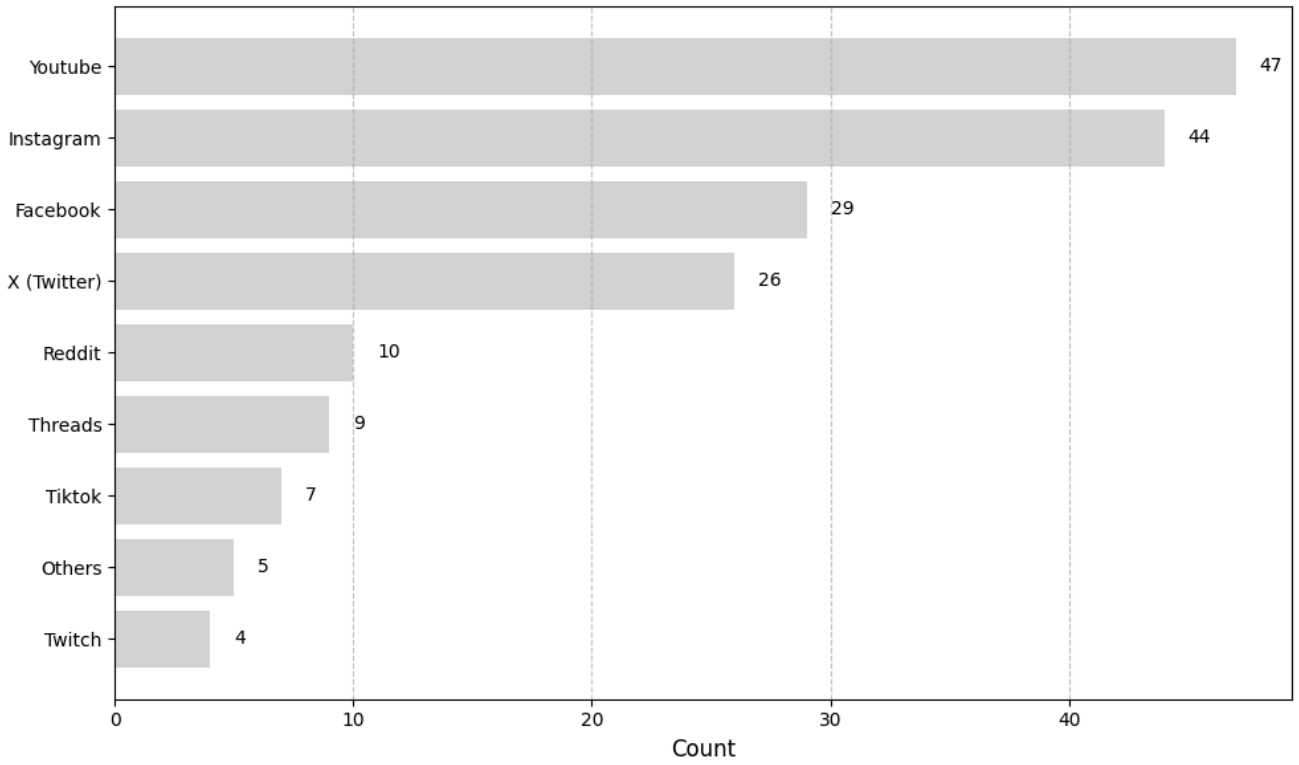
In [24]:
```
sns = { "1": "Youtube", "2": "Instagram", "3": "Facebook", "4": "Reddit", "5": "X (Twitter)", "6": "Threads", "7": "Tiktok", "8":
from collections import Counter
sns_counts = Counter([item for sublist in survey_df["Q13"].dropna().str.split(',') for item in sublist])
sns_mapping_counts = {sns[key]: sns_counts[key] for key in sns_counts}
sns_mapping_counts
```

Out[24]:
```
{'Youtube': 47,
 'Instagram': 44,
 'Facebook': 29,
 'Reddit': 10,
 'X (Twitter)': 26,
 'Threads': 9,
 'Tiktok': 7,
 'Twitch': 4,
 'Others': 5}
```

In [25]:
```
survey_df["Q13_9_TEXT"].dropna()
```

Out[25]:
```
10          아프리카TV
11    마스토돈(분산형SNS)
44           Tumblr
45           Tumblr
Name: Q13_9_TEXT, dtype: object
```

In [26]:
```
sorted_data = dict(sorted(sns_mapping_counts.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(10, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('Count', fontsize=12)
plt.ylabel('', fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
```



- 1. Please rank the three social media platforms you selected in order of most frequently used.

In [27]:
```
Rank_sns = survey_df[['Q14_1','Q14_2','Q14_3','Q14_4','Q14_5','Q14_6','Q14_7','Q14_8','Q14_9']]
Rank_sns.head(5)
```

Out[27]:

	Q14_1	Q14_2	Q14_3	Q14_4	Q14_5	Q14_6	Q14_7	Q14_8	Q14_9
0	2	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2	1	5	4	7	6	3	NaN	NaN
2	1	2	3	NaN	NaN	NaN	NaN	NaN	NaN
3	1	2	NaN	3	NaN	NaN	NaN	NaN	NaN
4	1	2	3	NaN	4	NaN	NaN	NaN	NaN

In [28]:
```
def reorder_ranks_with_nan(row):
    ordered_values = sorted([(i, val) for i, val in enumerate(row) if not pd.isna(val)], key=lambda x: x[1], reverse=True)
    reordered = [None] * len(row)
    for rank, (original_index, _) in enumerate(ordered_values, start=1):
        reordered[original_index] = rank
    return reordered

reordered_df = Rank_sns.apply(reorder_ranks_with_nan, axis=1, result_type='expand')
reordered_df.columns = Rank_sns.columns
reordered_df.head(5)
```

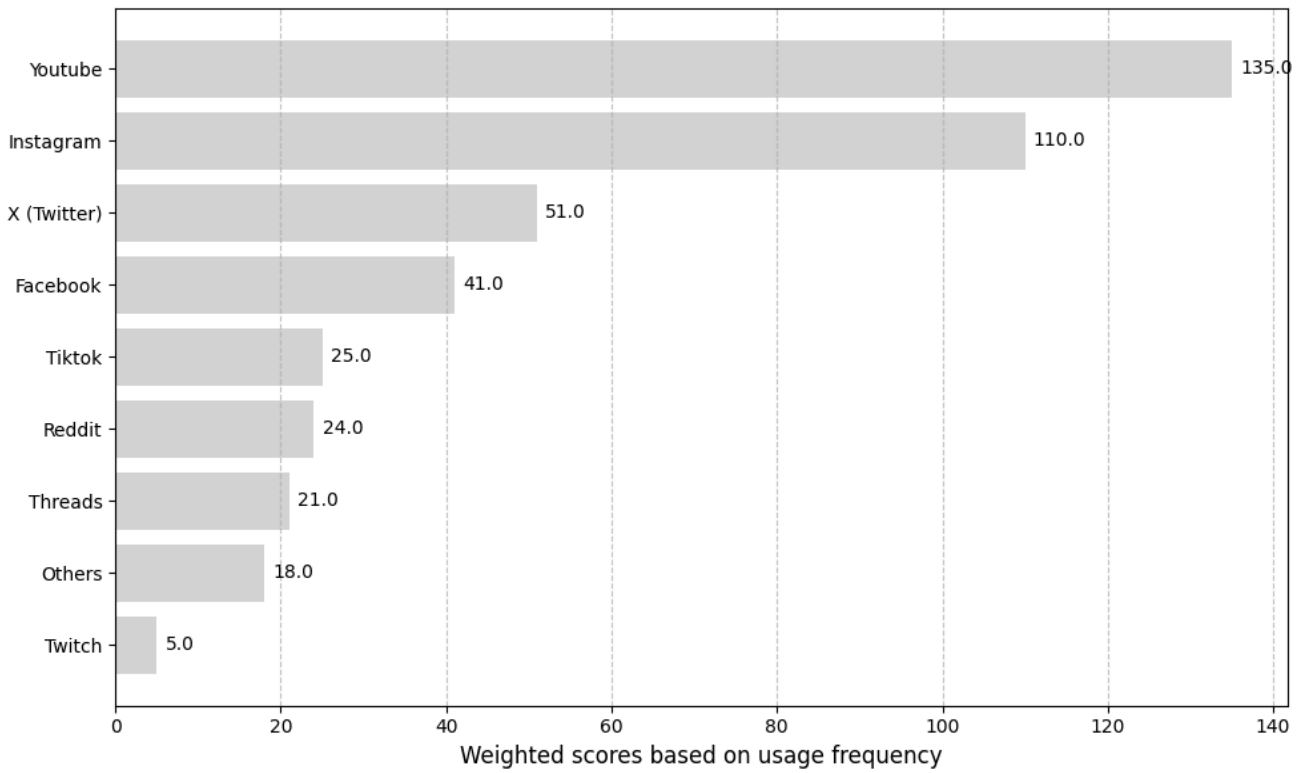
Out[28]:

	Q14_1	Q14_2	Q14_3	Q14_4	Q14_5	Q14_6	Q14_7	Q14_8	Q14_9
0	1.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	6.0	7.0	3.0	4.0	1.0	2.0	5.0	NaN	NaN
2	3.0	2.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN
3	3.0	2.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN
4	4.0	3.0	2.0	NaN	1.0	NaN	NaN	NaN	NaN

In [29]:
```
sns_2 = {"Q14_1": "Youtube", "Q14_2": "Instagram", "Q14_3": "Facebook", "Q14_4": "Reddit", "Q14_5": "X (Twitter)", "Q14_6": "Threa
sns_rankscore = reordered_df.sum(axis=0).rename(index=sns_2)
sns_rankscore
```

Out[29]:
```
Youtube        135.0
Instagram      110.0
Facebook        41.0
Reddit          24.0
X (Twitter)     51.0
Threads         21.0
Tiktok          25.0
Twitch           5.0
Others          18.0
dtype: float64
```

In [30]:
```
sorted_data = dict(sorted(sns_rankscore.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(10, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('Weighted scores based on usage frequency', fontsize=12)
plt.ylabel('', fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
```

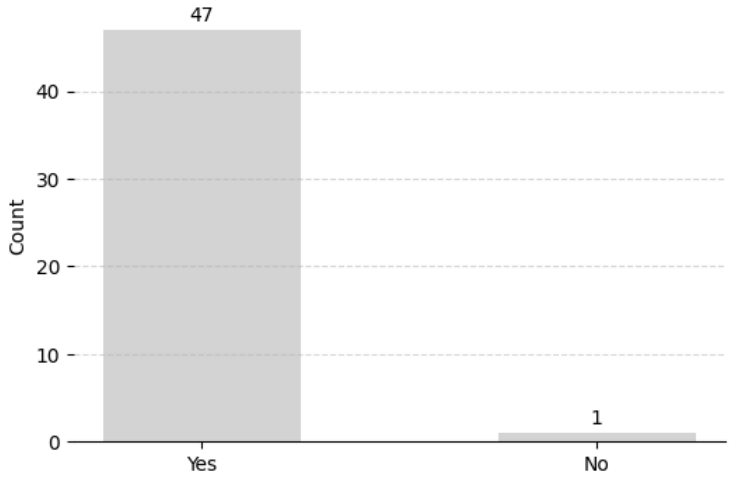


- 1. Have you ever come across posts, photos, or videos about other people's daily lives on social media?

In [31]:
```
RQ1_5= survey_df["Q15"].dropna().map(mapping)
RQ1_5.value_counts()
```

Out[31]:
```
Yes    47
No      1
Name: Q15, dtype: int64
```

In [32]:
```
plt.figure(figsize=(6, 4))
plt.bar(RQ1_5.value_counts().index, RQ1_5.value_counts().values, color='lightgray',width=0.5)
plt.xticks(rotation=0)
plt.title("")
plt.xlabel("")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
for i, value in enumerate(RQ1_5.value_counts().values):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```

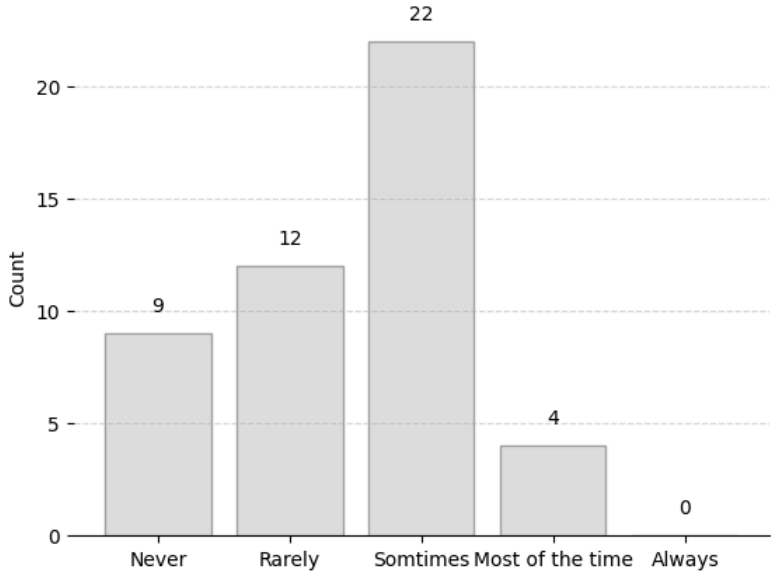


- 1. After encountering posts, photos, or videos about other people's daily lives on social media, have you ever experienced negative emotions (e.g., sadness, regret, loneliness, guilt, anger, etc.)?

In [33]:
```
freq_mapping = {"1": "Never", "2": "Rarely", "3": "Somtimes", "4": "Most of the time", "5": "Always"}
RQ1_6 = survey_df["Q16"].dropna().map(freq_mapping)
RQ1_6.value_counts()
```

Out[33]:
```
Somtimes            22
Rarely              12
Never                9
Most of the time     4
Name: Q16, dtype: int64
```

In [114…
```
RQ1_6_counts = {"Never": 9, "Rarely": 12, "Somtimes": 22, 'Most of the time': 4, 'Always': 0}
plt.bar(RQ1_6_counts.keys(), RQ1_6_counts.values(), color='lightgray', edgecolor="gray", alpha=0.7)
plt.title("")
plt.xlabel("")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
for i, value in enumerate(RQ1_6_counts.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```

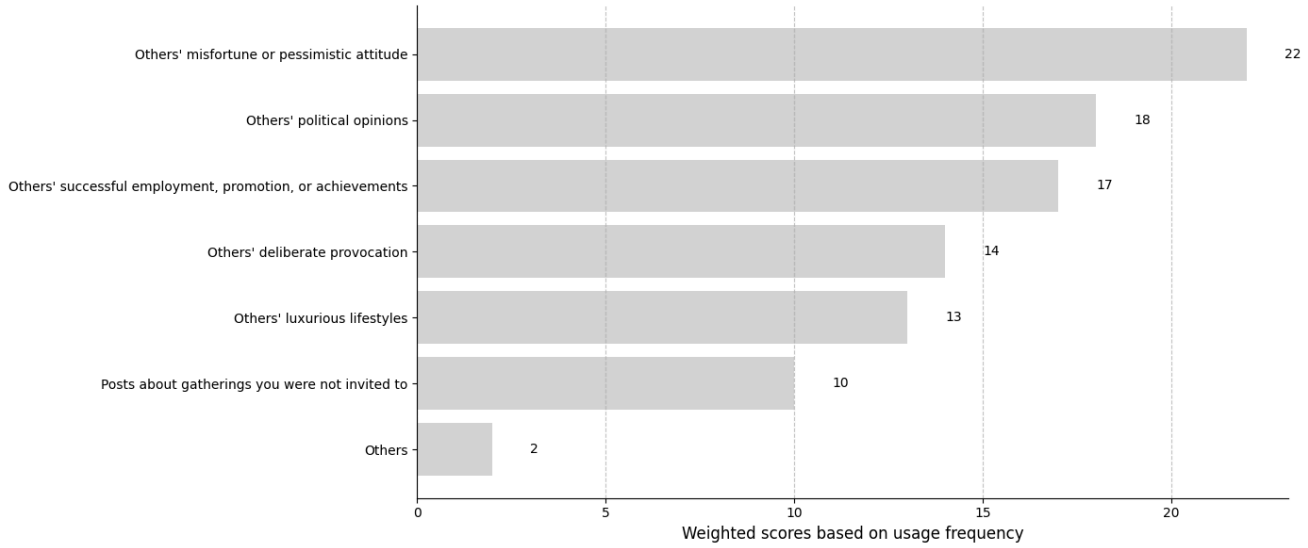


- 1. What was the content about? (Please check all that apply.)

In [35]:
```
Content_map = { "1": "Others' successful employment, promotion, or achievements", "2": "Others' luxurious lifestyles", "3": "Other
from collections import Counter
content_counts = Counter([item for sublist in survey_df["Q17"].dropna().str.split(',') for item in sublist])
content_mapping_counts = {Content_map[key]: content_counts[key] for key in content_counts}
content_mapping_counts
```

Out[35]:
```
{"Others' successful employment, promotion, or achievements": 17,
 "Others' luxurious lifestyles": 13,
 "Others' political opinions": 18,
 "Others' deliberate provocation": 14,
 "Others' misfortune or pessimistic attitude": 22,
 'Posts about gatherings you were not invited to': 10,
 'Others': 2}
```

In [36]:
```
sorted_data = dict(sorted(content_mapping_counts.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(14, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('Weighted scores based on usage frequency', fontsize=12)
plt.ylabel('', fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```

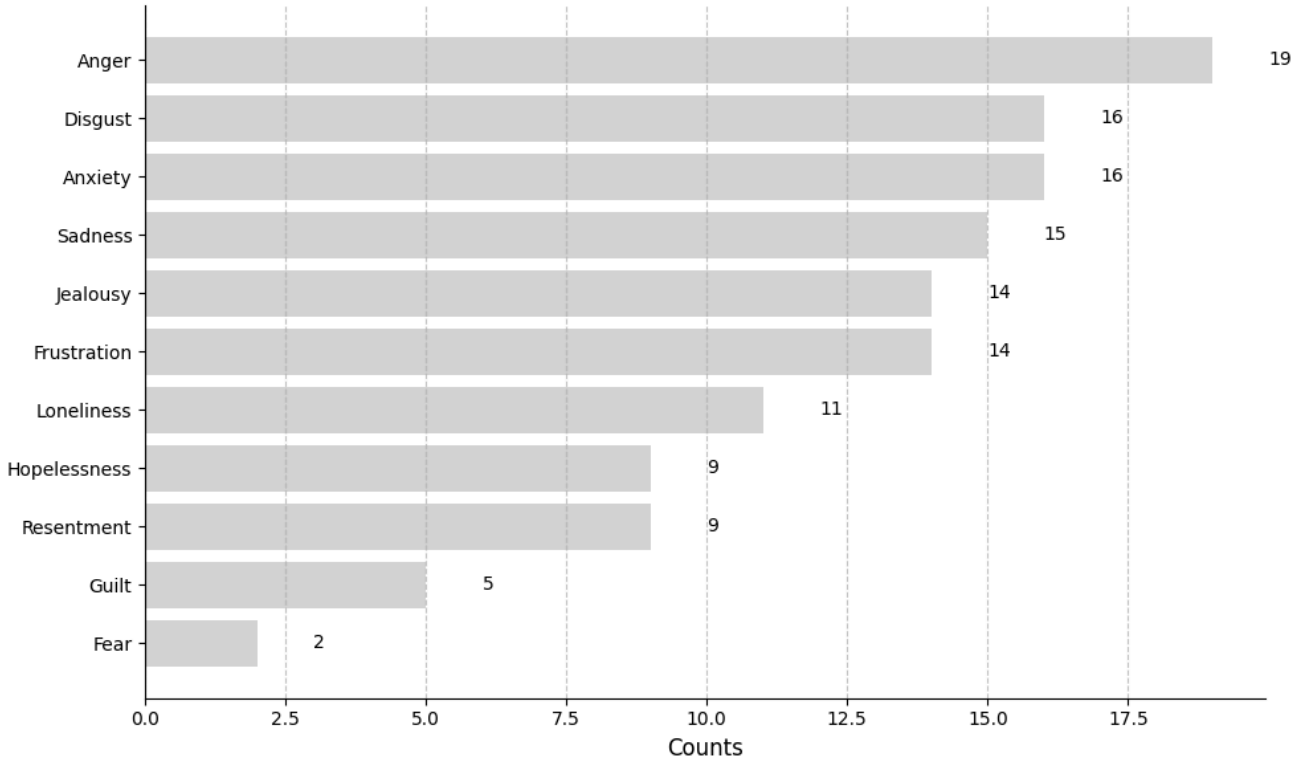


- 1. What emotion did you feel when encountering the unpleasant contents you selected? (Please check all that apply.)

In [37]:
```
emotion_mapping = { "1": "Anger", "2": "Sadness", "3": "Fear", "4": "Frustration", "5": "Disgust", "6": "Anxiety", "7": "Lonelines
from collections import Counter
emotion_counts = Counter([item for sublist in survey_df["Q18"].dropna().str.split(',') for item in sublist])
emotion_mapping_counts = {emotion_mapping[key]: emotion_counts[key] for key in emotion_counts}
emotion_mapping_counts
```

Out[37]:
```
{'Anger': 19,
 'Frustration': 14,
 'Anxiety': 16,
 'Loneliness': 11,
 'Sadness': 15,
 'Disgust': 16,
 'Jealousy': 14,
 'Resentment': 9,
 'Guilt': 5,
 'Hopelessness': 9,
 'Fear': 2}
```

In [38]:
```
sorted_data = dict(sorted(emotion_mapping_counts.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(10, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('Counts', fontsize=12)
plt.ylabel('', fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```



- 1. Please rank the emotions you selected in order of most frequently felt.

In [39]:
```
Rank_emotion = survey_df[['Q19_1','Q19_2','Q19_3','Q19_4','Q19_5','Q19_6','Q19_7','Q19_8','Q19_9', 'Q19_10','Q19_11']]
reordered_df2 = Rank_emotion.apply(reorder_ranks_with_nan, axis=1, result_type='expand')
reordered_df2.columns = Rank_emotion.columns
reordered_df2.head(5)
```

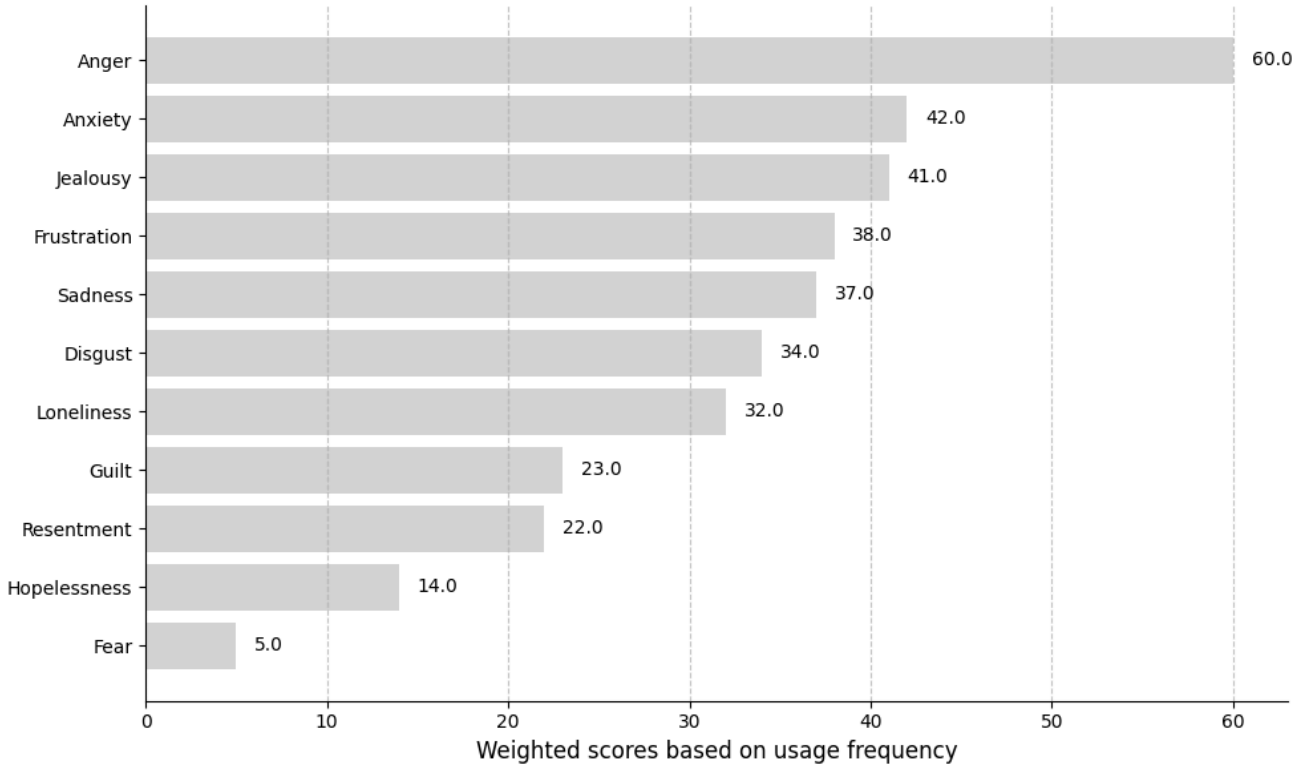
Out[39]:

	Q19_1	Q19_2	Q19_3	Q19_4	Q19_5	Q19_6	Q19_7	Q19_8	Q19_9	Q19_10	Q19_11
0	2.0	NaN	NaN	3.0	NaN	1.0	4.0	NaN	NaN	NaN	NaN
1	2.0	NaN	NaN	1.0	NaN	4.0	3.0	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	2.0	NaN	NaN	1.0	3.0	NaN	NaN	NaN	NaN

In [40]:
```
emotion_2 = { "Q19_1": "Anger", "Q19_2": "Sadness", "Q19_3": "Fear", "Q19_4": "Frustration", "Q19_5": "Disgust", "Q19_6": "Anxiety
emotion_rankscore = reordered_df2.sum(axis=0).rename(index=emotion_2)
emotion_rankscore
```

Out[40]:
```
Anger           60.0
Sadness         37.0
Fear             5.0
Frustration     38.0
Disgust         34.0
Anxiety         42.0
Loneliness      32.0
Hopelessness    14.0
Jealousy        41.0
Resentment      22.0
Guilt           23.0
dtype: float64
```

In [41]:
```
sorted_data = dict(sorted(emotion_rankscore.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(10, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('Weighted scores based on usage frequency', fontsize=12)
plt.ylabel('', fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```



- 1. How much did the contents affect your negative emotions?

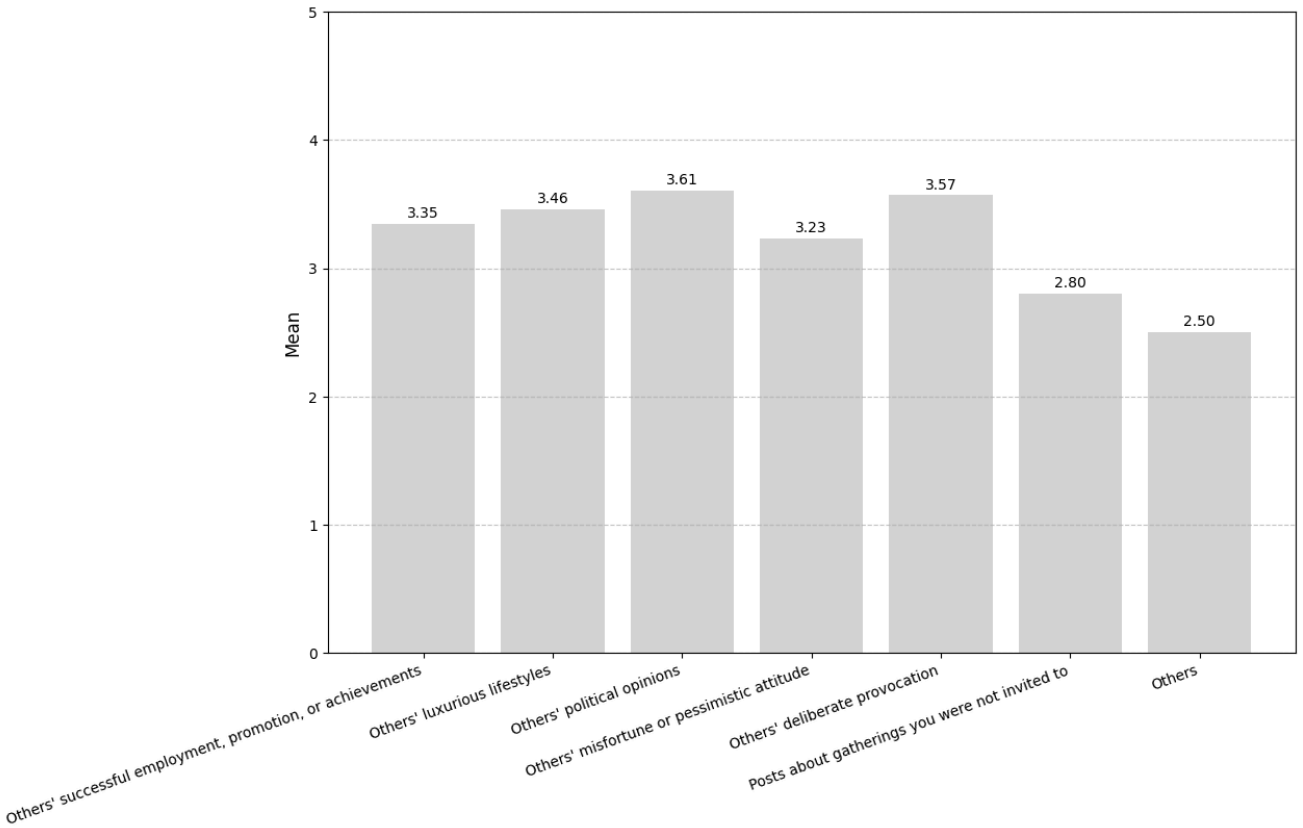
In [78]:
```
df10 = survey_df[['Q20_1','Q20_2','Q20_3','Q20_4','Q20_5','Q20_6','Q20_7']]
mean_dict = {}
```

```
for column in df10.columns:
    mean_value = df10[column].dropna().astype(int).mean(axis=0).round(2)
    mean_dict[column] = mean_value
Q20mapping = { "Q20_1": "Others' successful employment, promotion, or achievements", "Q20_2": "Others' luxurious lifestyles", "Q20
mapped_mean_dict = {Q20mapping[key]: value for key, value in mean_dict.items()}
```

In [79]:
```
mapped_mean_dict
```

Out[79]:
```
{"Others' successful employment, promotion, or achievements": 3.35,
 "Others' luxurious lifestyles": 3.46,
 "Others' political opinions": 3.61,
 "Others' misfortune or pessimistic attitude": 3.23,
 "Others' deliberate provocation": 3.57,
 'Posts about gatherings you were not invited to': 2.8,
 'Others': 2.5}
```

In [94]:
```
categories = list(mapped_mean_dict.keys())
values = list(mapped_mean_dict.values())
plt.figure(figsize=(12, 8))
plt.bar(categories, values, color='lightgray')
plt.title('')
plt.ylabel('Mean', fontsize=12)
plt.xlabel('', fontsize=12)
plt.xticks(rotation=20, ha='right', fontsize=10)  # 레이블 각도와 정렬
plt.yticks(fontsize=10)
plt.ylim(0, 5)
plt.grid(axis="y", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(i, value + 0.05, f"{value:.2f}", ha='center', fontsize=10)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```



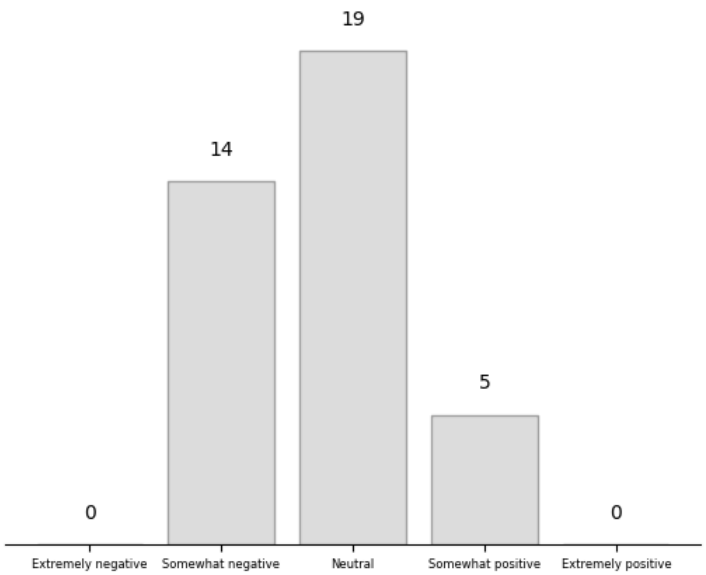
- 1. How do the emotions you experience from social media content affect your daily life?

In [96]:
```
pos_neg_mapping = { "1": "Extremely negative", "2": "Somewhat negative", "3": "Neutral", "4": "Somewhat positive", "5": "Extremely
survey_df["Q21"].map(pos_neg_mapping).value_counts()
```

Out[96]:
```
Neutral             19
Somewhat negative    14
Somewhat positive     5
Name: Q21, dtype: int64
```

In [113…
```
RQ1_11_counts = {"Extremely negative": 0, "Somewhat negative": 14, "Neutral": 19, 'Somewhat positive': 5, 'Extremely positive': 0}
plt.bar(RQ1_11_counts.keys(), RQ1_6_counts.values(), color='lightgray', edgecolor="gray", alpha=0.7)
plt.title("")
plt.xlabel("")
plt.ylabel("")
```

```
plt.grid(axis='y', linestyle="--", alpha=0.5)
plt.xticks(fontsize=6)
plt.yticks([])
for i, value in enumerate(RQ1_11_counts.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



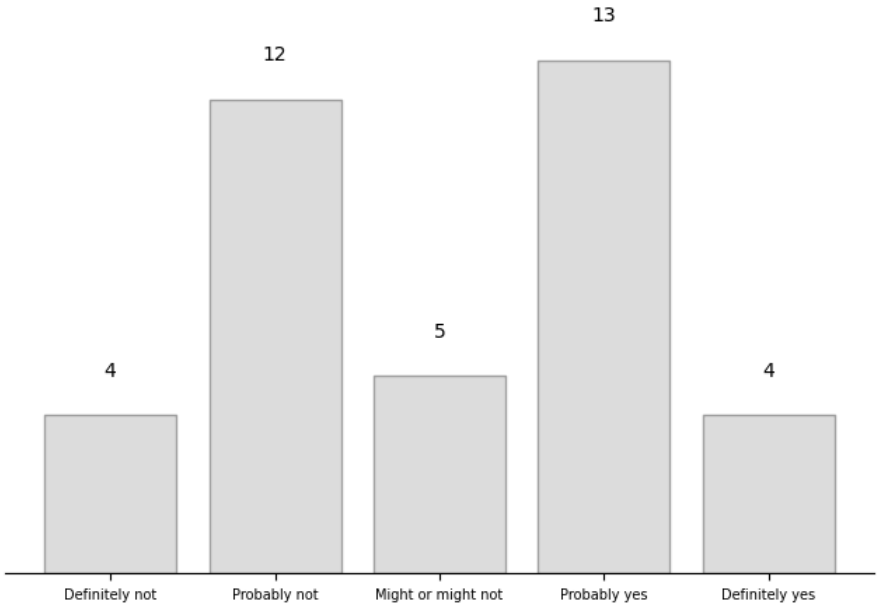
- evaluation

Has this influenced whether or not you use social media?

In [121…
```
yes_no_mapping = { "1": "Definitely not", "2": "Probably not", "3": "Might or might not", "4": "Probably yes", "5": "Definitely ye
eval_1 = survey_df["Q22"].map(yes_no_mapping).value_counts()
eval_1
```

Out[121]:
```
Probably yes          13
Probably not          12
Might or might not     5
Definitely yes         4
Definitely not         4
Name: Q22, dtype: int64
```

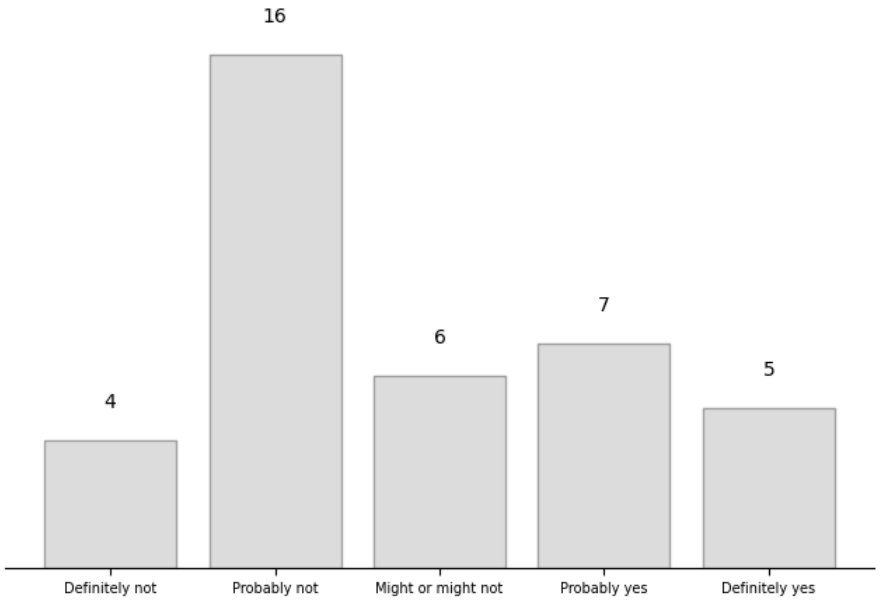
In [136…
```
ordered_labels = ["Definitely not", "Probably not", "Might or might not", "Probably yes", "Definitely yes"]
ordered_data = {label: eval_1[label] for label in ordered_labels}
categories = list(ordered_data.keys())
values = list(ordered_data.values())
plt.figure(figsize=(8, 5))  # 가로 8, 세로 5 크기
plt.bar(categories, values,  color='lightgray', edgecolor="gray", alpha=0.7)

plt.xlabel("")
plt.ylabel("")
plt.grid(axis='y', linestyle="--", alpha=0.5)
plt.xticks(fontsize=7)
plt.yticks([])
for i, value in enumerate(ordered_data.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



In [137…
```
eval_2 = survey_df["Q23"].map(yes_no_mapping).value_counts()
eval_2
```

Out[137]:
```
Probably not          16
Probably yes           7
Might or might not     6
Definitely yes         5
Definitely not         4
Name: Q23, dtype: int64
```

In [138…
```
ordered_labels = ["Definitely not", "Probably not", "Might or might not", "Probably yes", "Definitely yes"]
ordered_data = {label: eval_2[label] for label in ordered_labels}
categories = list(ordered_data.keys())
values = list(ordered_data.values())
plt.figure(figsize=(8, 5))  # 가로 8, 세로 5 크기
plt.bar(categories, values,  color='lightgray', edgecolor="gray", alpha=0.7)

plt.xlabel("")
plt.ylabel("")
plt.grid(axis='y', linestyle="--", alpha=0.5)
plt.xticks(fontsize=7)
plt.yticks([])
for i, value in enumerate(ordered_data.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```

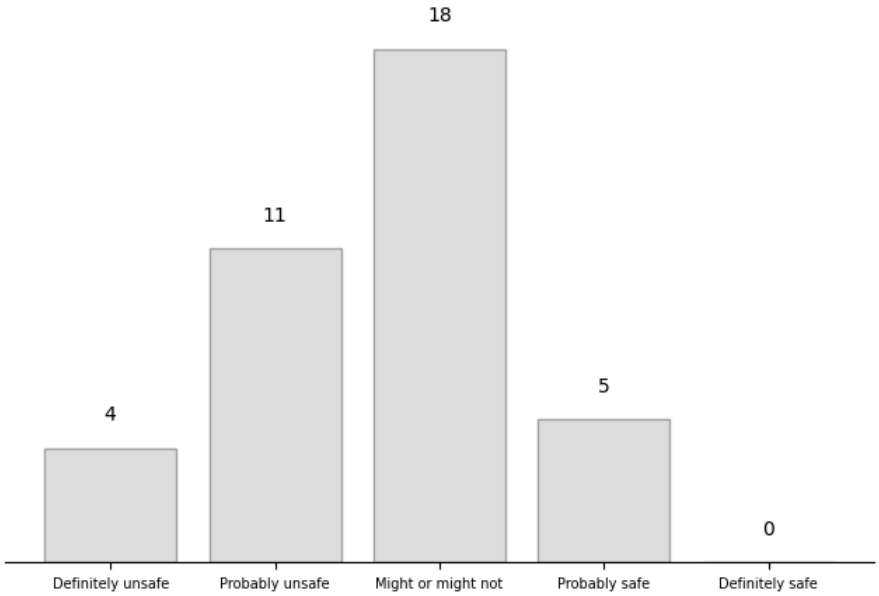


In [110…
```
safe_unsafe_mapping = { "1": "Definitely unsafe", "2": "Probably unsafe", "3": "Might or might not", "4": "Probably safe", "5": "D
```

In [141…
```
eval_3 = survey_df["Q24"].map(safe_unsafe_mapping).value_counts()
eval_3['Definitely safe'] = 0
eval_3
```

Out[141]:
```
Might or might not    18
Probably unsafe       11
Probably safe          5
Definitely unsafe      4
Definitely safe        0
Name: Q24, dtype: int64
```

In [142…
```
ordered_labels = ["Definitely unsafe", "Probably unsafe", "Might or might not", "Probably safe", "Definitely safe"]
ordered_data = {label: eval_3[label] for label in ordered_labels}
categories = list(ordered_data.keys())
values = list(ordered_data.values())
plt.figure(figsize=(8, 5))  # 가로 8, 세로 5 크기
plt.bar(categories, values,  color='lightgray', edgecolor="gray", alpha=0.7)

plt.xlabel("")
plt.ylabel("")
plt.grid(axis='y', linestyle="--", alpha=0.5)
plt.xticks(fontsize=7)
plt.yticks([])
for i, value in enumerate(ordered_data.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



In [ ]:
```
```

Step 4: RQ2

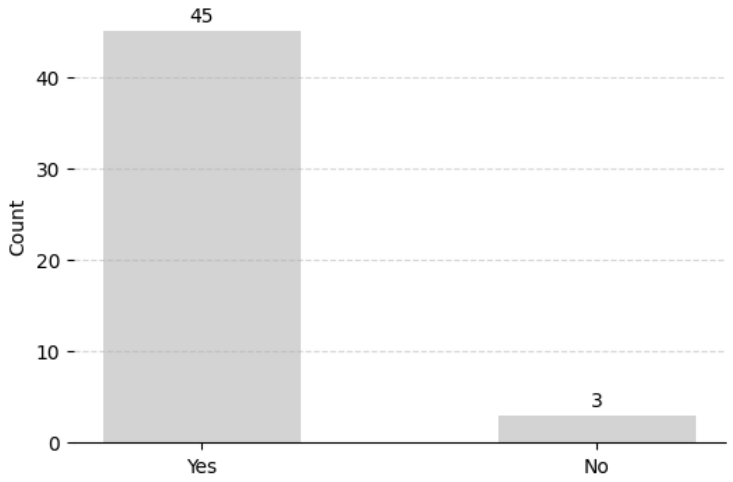
- 1. Have you ever encountered content recommended by algorithms (e.g., YouTube Shorts, Instagram Reels)?

In [143…
```
RQ2_1= survey_df["Q25"].map(mapping).value_counts()
RQ2_1
```

Out[143]:
```
Yes    45
No      3
Name: Q25, dtype: int64
```

In [146…
```
plt.figure(figsize=(6, 4))
plt.bar(RQ2_1.index, RQ2_1.values, color='lightgray',width=0.5)
plt.xticks(rotation=0)
plt.title("")
plt.xlabel("")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
for i, value in enumerate(RQ2_1.values):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
```

```
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```

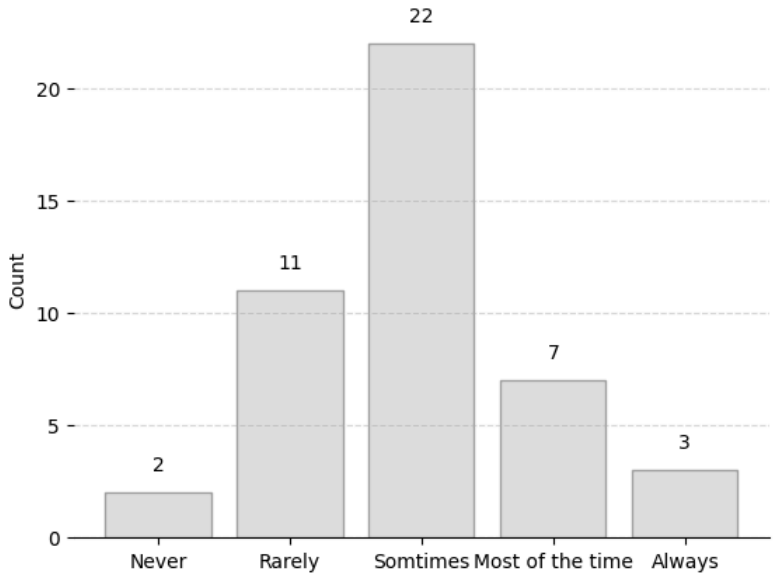


- 1. Have you incidentally encountered unpleasant content recommended by an algorithm?

In [154…
```
freq_mapping = {"1": "Never", "2": "Rarely", "3": "Somtimes", "4": "Most of the time", "5": "Always"}
RQ2_2= survey_df["Q26"].map(freq_mapping).value_counts()
RQ2_2
```

Out[154]:
```
Somtimes            22
Rarely              11
Most of the time     7
Always               3
Never                2
Name: Q26, dtype: int64
```

In [156…
```
RQ1_6_counts = {"Never": 2, "Rarely": 11, "Somtimes": 22, 'Most of the time': 7, 'Always': 3}
plt.bar(RQ1_6_counts.keys(), RQ1_6_counts.values(), color='lightgray', edgecolor="gray", alpha=0.7)
plt.title("")
plt.xlabel("")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
for i, value in enumerate(RQ1_6_counts.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



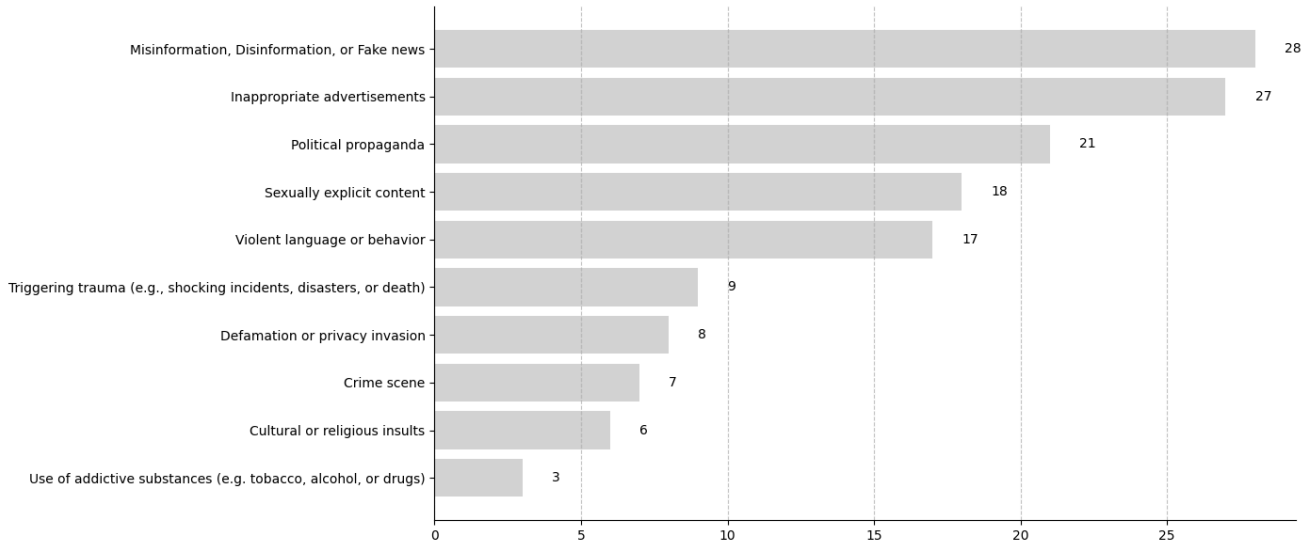
- 1. What was the content about? (Please check all that apply.)

In [165…
```
Content_map = mapping = { "1": "Violent language or behavior", "2": "Sexually explicit content", "3": "Inappropriate advertisement
from collections import Counter
```

```
content_counts = Counter([item for sublist in survey_df["Q27"].dropna().str.split(',') for item in sublist])
content_mapping_counts = {Content_map[key]: content_counts[key] for key in content_counts}
content_mapping_counts
```

Out[165]:
```
{'Political propaganda': 21,
 'Misinformation, Disinformation, or Fake news': 28,
 'Violent language or behavior': 17,
 'Inappropriate advertisements': 27,
 'Triggering trauma (e.g., shocking incidents, disasters, or death)': 9,
 'Crime scene': 7,
 'Defamation or privacy invasion': 8,
 'Sexually explicit content': 18,
 'Cultural or religious insults': 6,
 'Use of addictive substances (e.g. tobacco, alcohol, or drugs)': 3}
```

In [166…
```
sorted_data = dict(sorted(content_mapping_counts.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(14, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('', fontsize=12)
plt.ylabel('', fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```



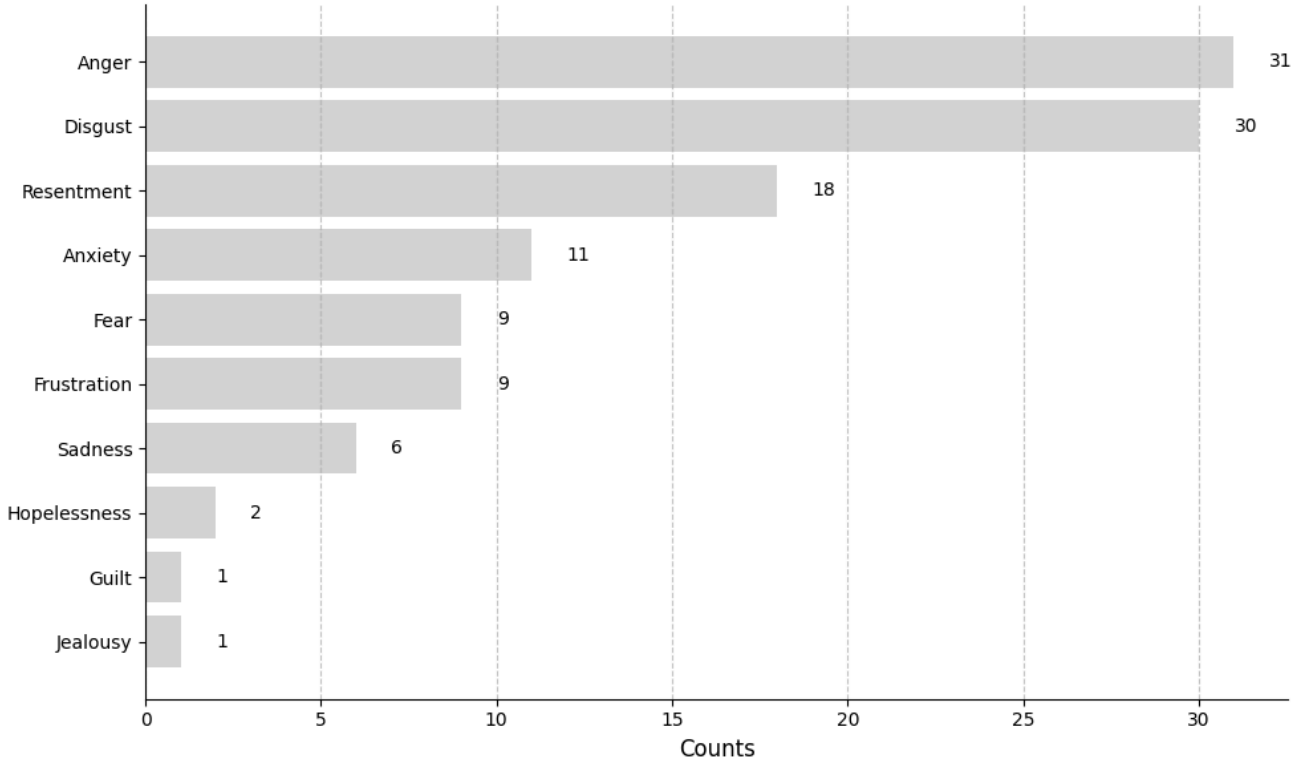
- 1. What emotion did you feel when encountering the unpleasant contents you selected? (Please check all that apply.)

In [168…
```
emotion_mapping = { "1": "Anger", "2": "Sadness", "3": "Fear", "4": "Frustration", "5": "Disgust", "6": "Anxiety", "7": "Lonelines
from collections import Counter
emotion_counts = Counter([item for sublist in survey_df["Q28"].dropna().str.split(',') for item in sublist])
emotion_mapping_counts = {emotion_mapping[key]: emotion_counts[key] for key in emotion_counts}
emotion_mapping_counts
```

Out[168]:
```
{'Anger': 31,
 'Frustration': 9,
 'Fear': 9,
 'Disgust': 30,
 'Resentment': 18,
 'Sadness': 6,
 'Anxiety': 11,
 'Jealousy': 1,
 'Guilt': 1,
 'Hopelessness': 2}
```

In [169…
```
sorted_data = dict(sorted(emotion_mapping_counts.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(10, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('Counts', fontsize=12)
plt.ylabel('', fontsize=12)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
```

```
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```



- 1. Please rank the emotions you selected in order of most frequently felt.

In [171…
```
Rank_emotion = survey_df[['Q29_1','Q29_2','Q29_3','Q29_4','Q29_5','Q29_6','Q29_7','Q29_8','Q29_9','Q29_10','Q29_11']]
reordered_df4 = Rank_emotion.apply(reorder_ranks_with_nan, axis=1, result_type='expand')
reordered_df4.columns = Rank_emotion.columns
reordered_df4.head(5)
```

Out[171]:

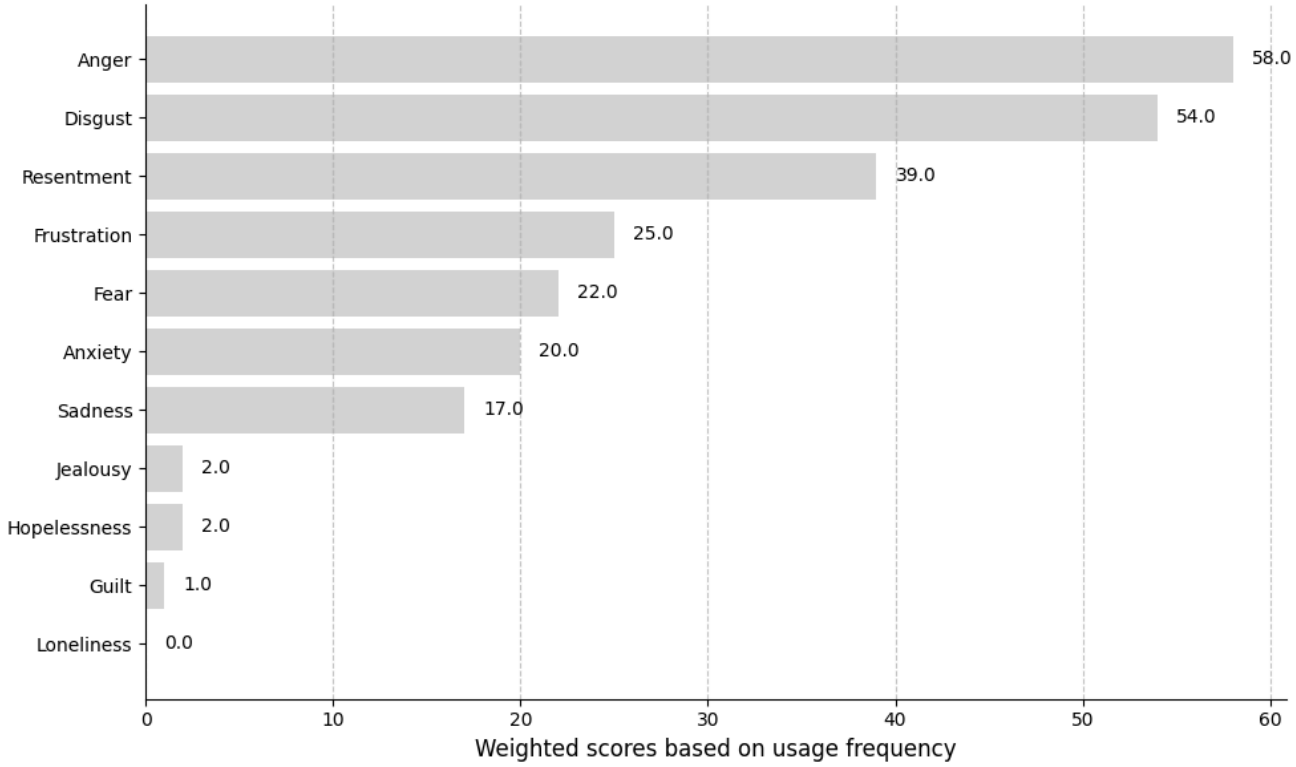
	Q29_1	Q29_2	Q29_3	Q29_4	Q29_5	Q29_6	Q29_7	Q29_8	Q29_9	Q29_10	Q29_11
0	1.0	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1.0	NaN	3.0	NaN	2.0	NaN	NaN	NaN	NaN	4.0	NaN
2	2.0	NaN	1.0	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN
3	2.0	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN
4	3.0	4.0	NaN	NaN	5.0	1.0	NaN	NaN	NaN	2.0	NaN

In [173…
```
emotion_2 = { "Q29_1": "Anger", "Q29_2": "Sadness", "Q29_3": "Fear", "Q29_4": "Frustration", "Q29_5": "Disgust", "Q29_6": "Anxiety
emotion_rankscore = reordered_df4.sum(axis=0).rename(index=emotion_2)
emotion_rankscore
```

Out[173]:
```
Anger           58.0
Sadness         17.0
Fear            22.0
Frustration     25.0
Disgust         54.0
Anxiety         20.0
Loneliness       0.0
Hopelessness     2.0
Jealousy         2.0
Resentment      39.0
Guilt            1.0
dtype: float64
```

In [174…
```
sorted_data = dict(sorted(emotion_rankscore.items(), key=lambda x: x[1], reverse=False))
platforms = list(sorted_data.keys())  # X
values = list(sorted_data.values())  # Y
plt.figure(figsize=(10, 6))
plt.barh(platforms, values, color='lightgray')  # barh
plt.title('')
plt.xlabel('Weighted scores based on usage frequency', fontsize=12)
plt.ylabel('', fontsize=12)
```

```
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 1, i, str(value), va='center', fontsize=10)
plt.tight_layout()  # 레이아웃 조정
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```



- 6.How much did the contents affect your negative emotions?

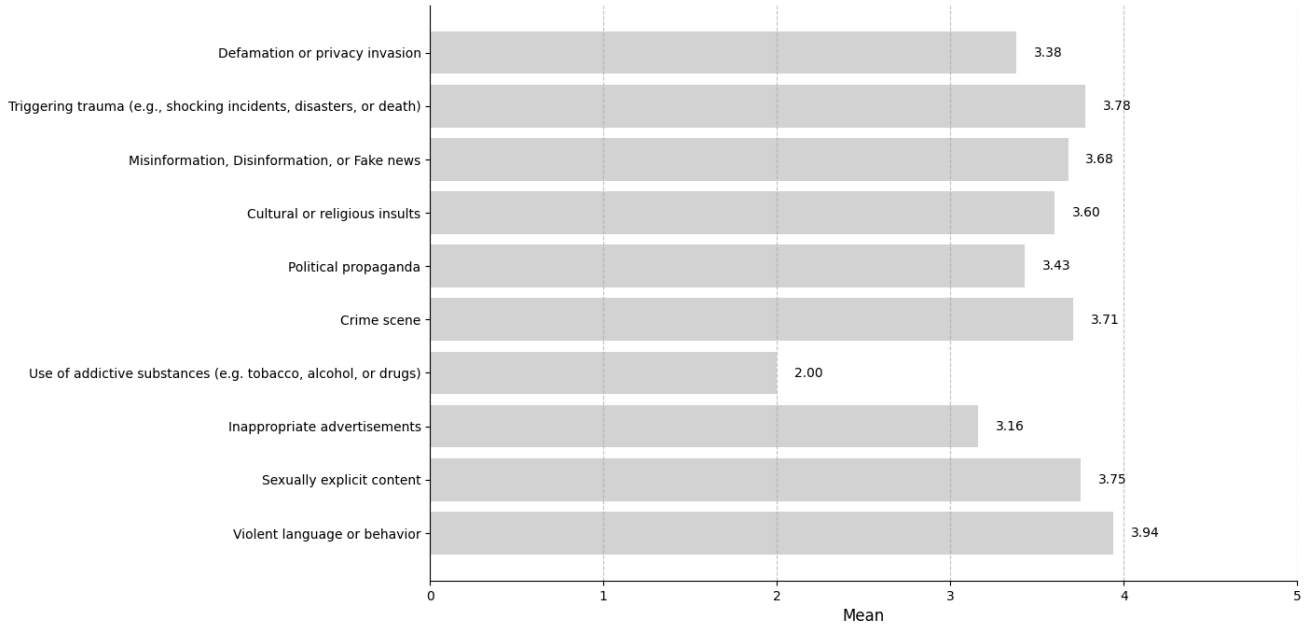
In [254…
```
df30 = survey_df[['Q30_1','Q30_2','Q30_3','Q30_4','Q30_5','Q30_6','Q30_7','Q30_8','Q30_9','Q30_10','Q30_11']]
mean_dict = {}
for column in df30.columns:
    mean_value = df30[column].dropna().astype(float).mean(axis=0)
    mean_dict[column] = mean_value
Q30mapping = Content_map = { "Q30_1": "Violent language or behavior", "Q30_2": "Sexually explicit content", "Q30_3": "Inappropriat
mapped_mean_dict = {Q30mapping[key]: value for key, value in mean_dict.items()}

import math
rounded_dict = {key: round(value, 2) for key, value in mapped_mean_dict.items() if not math.isnan(value)}
```

In [255…
```
rounded_dict
```

Out[255]:
```
{'Violent language or behavior': 3.94,
 'Sexually explicit content': 3.75,
 'Inappropriate advertisements': 3.16,
 'Use of addictive substances (e.g. tobacco, alcohol, or drugs)': 2.0,
 'Crime scene': 3.71,
 'Political propaganda': 3.43,
 'Cultural or religious insults': 3.6,
 'Misinformation, Disinformation, or Fake news': 3.68,
 'Triggering trauma (e.g., shocking incidents, disasters, or death)': 3.78,
 'Defamation or privacy invasion': 3.38}
```

In [258…
```
categories = list(rounded_dict.keys())
values = list(rounded_dict.values())
plt.figure(figsize=(12, 8))
plt.barh(categories, values, color='lightgray')

plt.title('', fontsize=16)
plt.xlabel('Mean', fontsize=12)  # X축 레이블
plt.ylabel('', fontsize=12)  # Y축 레이블은 빈 상태로 둠

plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.xlim(0, 5)
plt.grid(axis="x", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(value + 0.1, i, f"{value:.2f}", va='center', fontsize=10)  # 막대 오른쪽에 값 표시
```

```
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```

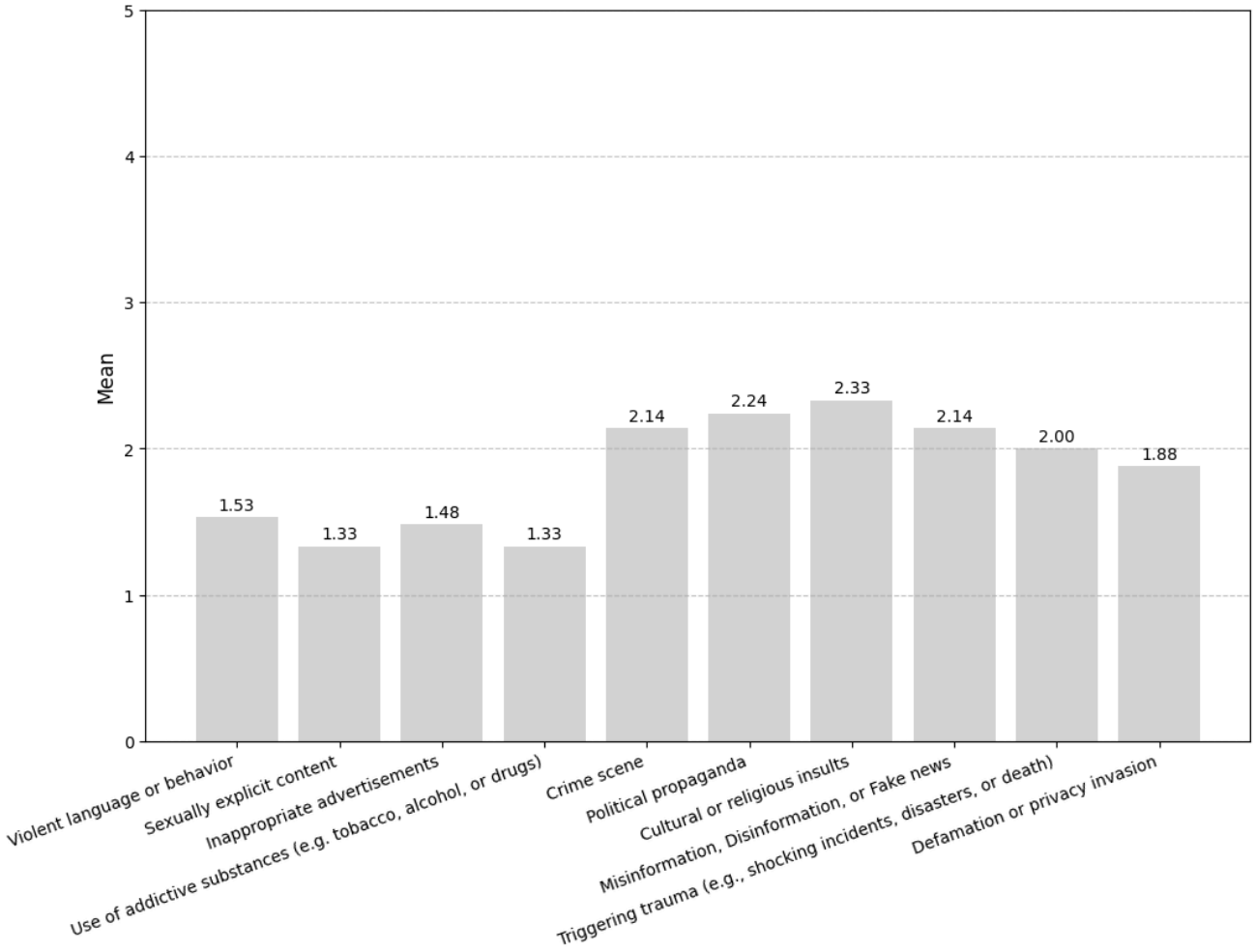


- 1. How do social media contents that provoke unpleasant emotions influence you—prompting you to ignore the information, or encouraging you to seek it further?

In [216…
```
df31 = survey_df[['Q31_1','Q31_2','Q31_3','Q31_4','Q31_5','Q31_6','Q31_7','Q31_8','Q31_9','Q31_10','Q31_11']]
mean_dict = {}
for column in df31.columns:
    mean_value = df30[column].dropna().astype(float).mean(axis=0)
    mean_dict[column] = mean_value
Q31mapping = Content_map = { "Q31_1": "Violent language or behavior", "Q31_2": "Sexually explicit content", "Q31_3": "Inappropriat
mapped_mean_dict = {Q31mapping[key]: value for key, value in mean_dict.items()}
rounded_dict = {key: round(value, 2) for key, value in mapped_mean_dict.items() if not math.isnan(value)}
rounded_dict
```

Out[216]:
```
{'Violent language or behavior': 1.53,
 'Sexually explicit content': 1.33,
 'Inappropriate advertisements': 1.48,
 'Use of addictive substances (e.g. tobacco, alcohol, or drugs)': 1.33,
 'Crime scene': 2.14,
 'Political propaganda': 2.24,
 'Cultural or religious insults': 2.33,
 'Misinformation, Disinformation, or Fake news': 2.14,
 'Triggering trauma (e.g., shocking incidents, disasters, or death)': 2.0,
 'Defamation or privacy invasion': 1.88}
```

In [217…
```
categories = list(rounded_dict.keys())
values = list(rounded_dict.values())
plt.figure(figsize=(12, 8))
plt.bar(categories, values, color='lightgray')
plt.title('')
plt.ylabel('Mean', fontsize=12)
plt.xlabel('', fontsize=12)
plt.xticks(rotation=20, ha='right', fontsize=10)  # 레이블 각도와 정렬
plt.yticks(fontsize=10)
plt.ylim(0, 5)
plt.grid(axis="y", linestyle="--", alpha=0.7)
for i, value in enumerate(values):
    plt.text(i, value + 0.05, f"{value:.2f}", ha='center', fontsize=10)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```

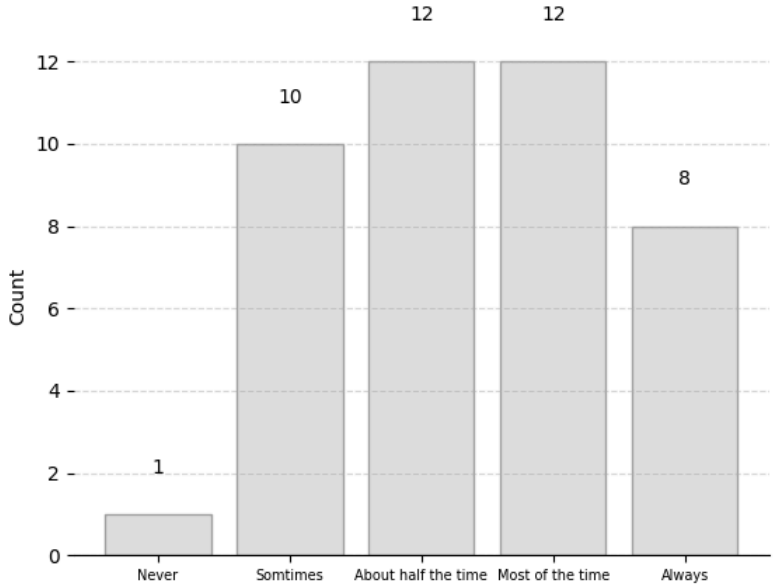


- 1. When you encounter unpleasant contents, do you usually report to the algorithm as "Do not recommend this type of contents"?

In [224…
```
freq_mapping = {"8": "Never", "9": "Somtimes", "10": "About half the time", "11": "Most of the time", "12": "Always"}
RQ2_8 = survey_df["Q32"].dropna().map(freq_mapping)
RQ2_8.value_counts()
```

Out[224]:
```
Most of the time       12
About half the time    12
Somtimes               10
Always                  8
Never                   1
Name: Q32, dtype: int64
```

In [227…
```
counts = {"Never": 1,  "Somtimes": 10, "About half the time": 12,'Most of the time': 12, 'Always': 8}
plt.bar(counts.keys(), counts.values(), color='lightgray', edgecolor="gray", alpha=0.7)
plt.title("")
plt.xlabel("")
plt.ylabel("Count")
plt.xticks(fontsize=7)
plt.grid(axis='y', linestyle="--", alpha=0.5)
for i, value in enumerate(counts.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```

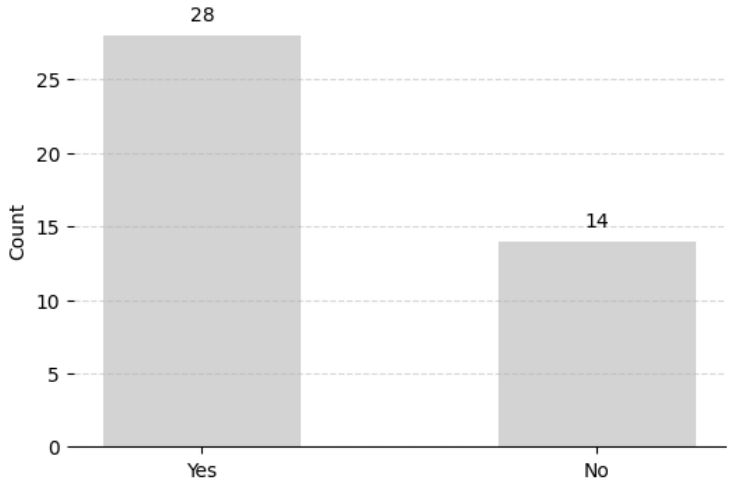


- 1. Have you ever experienced an algorithm repeatedly recommending the unpleasant contents despite reporting it as "Not recommend this type of content"?

In [231…
```
mapping = { "1": "No", "2": "Yes"}
RQ2_9= survey_df["Q33"].map(mapping)
RQ2_9.value_counts()
```

Out[231]:
```
Yes    28
No     14
Name: Q33, dtype: int64
```

In [232…
```
plt.figure(figsize=(6, 4))
plt.bar(RQ2_9.value_counts().index, RQ2_9.value_counts().values, color='lightgray',width=0.5)
plt.xticks(rotation=0)
plt.title("")
plt.xlabel("")
plt.ylabel("Count")
plt.grid(axis='y', linestyle="--", alpha=0.5)
for i, value in enumerate(RQ1_1.value_counts().values):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



Evaluation

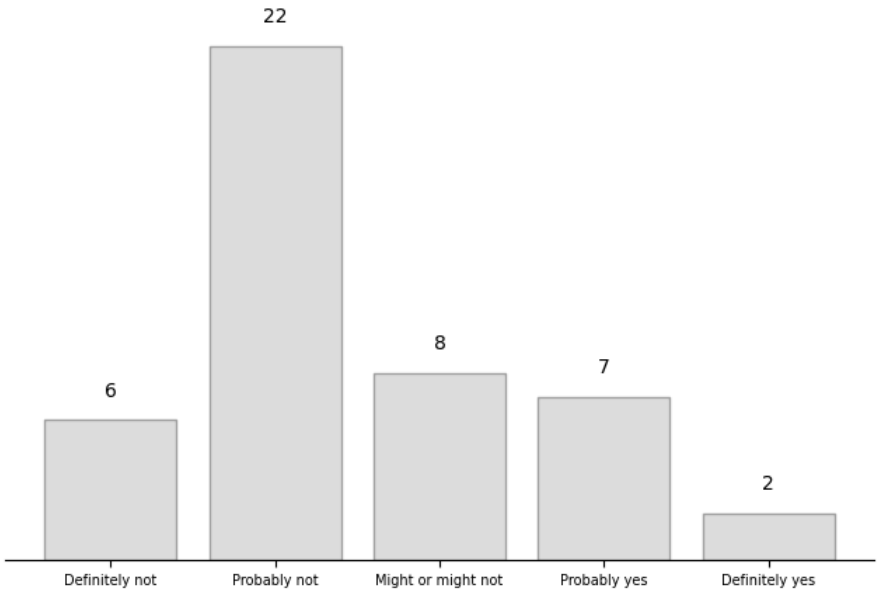
- Has this influenced whether or not you use social media?

In [240…
```
yes_no_mapping = { "1": "Definitely not", "2": "Probably not", "3": "Might or might not", "4": "Probably yes", "5": "Definitely ye
eval_1 = survey_df["Q34"].map(yes_no_mapping).value_counts()
```

```
eval_1
```

Out[240]:
```
Probably not          22
Might or might not     8
Probably yes           7
Definitely not         6
Definitely yes         2
Name: Q34, dtype: int64
```

In [241…
```
ordered_labels = ["Definitely not", "Probably not", "Might or might not", "Probably yes", "Definitely yes"]
ordered_data = {label: eval_1[label] for label in ordered_labels}
categories = list(ordered_data.keys())
values = list(ordered_data.values())
plt.figure(figsize=(8, 5))  # 가로 8, 세로 5 크기
plt.bar(categories, values,  color='lightgray', edgecolor="gray", alpha=0.7)

plt.xlabel("")
plt.ylabel("")
plt.grid(axis='y', linestyle="--", alpha=0.5)
plt.xticks(fontsize=7)
plt.yticks([])
for i, value in enumerate(ordered_data.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



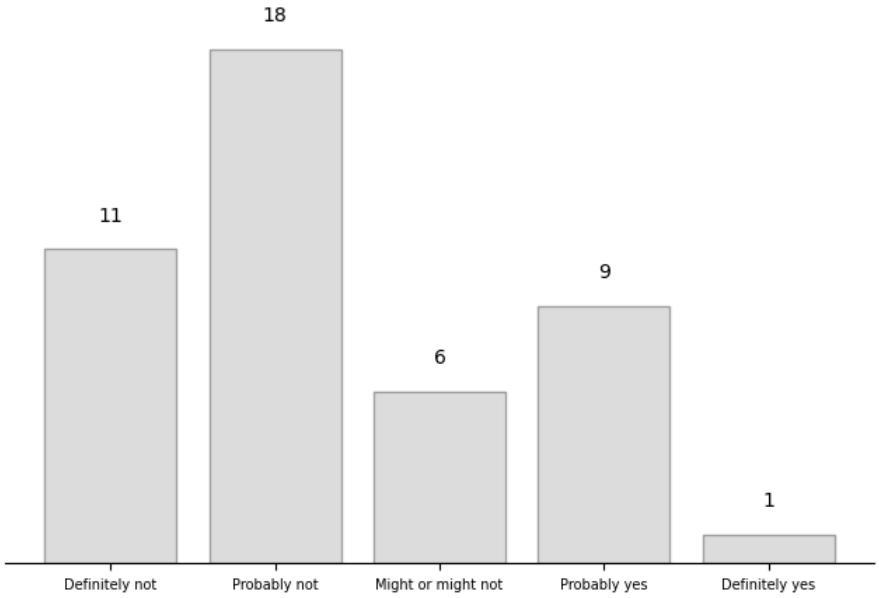
- Has this influenced the amount of time you spend on social media?

In [242…
```
eval_2 = survey_df["Q35"].map(yes_no_mapping).value_counts()
eval_2
```

Out[242]:
```
Probably not          18
Definitely not        11
Probably yes           9
Might or might not     6
Definitely yes         1
Name: Q35, dtype: int64
```

In [243…
```
ordered_labels = ["Definitely not", "Probably not", "Might or might not", "Probably yes", "Definitely yes"]
ordered_data = {label: eval_2[label] for label in ordered_labels}
categories = list(ordered_data.keys())
values = list(ordered_data.values())
plt.figure(figsize=(8, 5))  # 가로 8, 세로 5 크기
plt.bar(categories, values,  color='lightgray', edgecolor="gray", alpha=0.7)

plt.xlabel("")
plt.ylabel("")
plt.grid(axis='y', linestyle="--", alpha=0.5)
plt.xticks(fontsize=7)
plt.yticks([])
for i, value in enumerate(ordered_data.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
```

```
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```



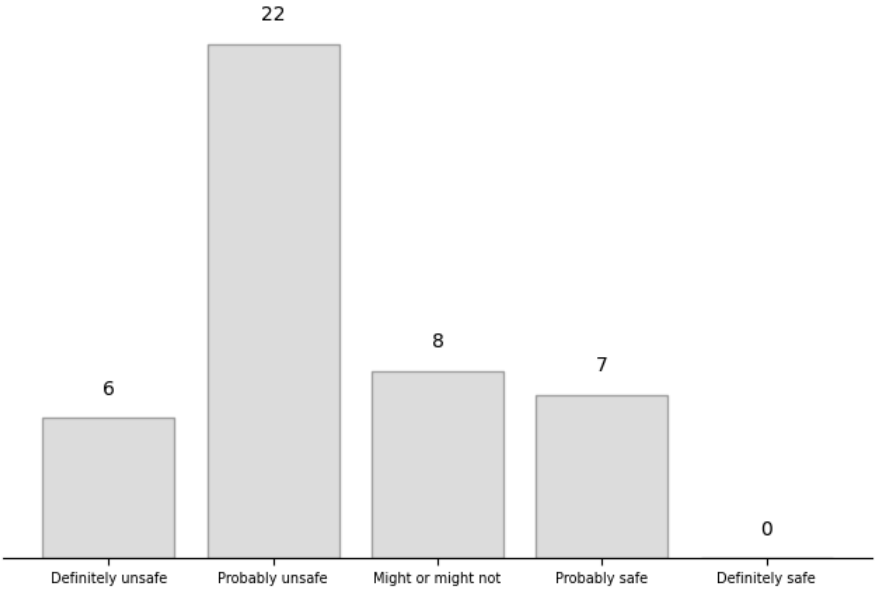
- Has this made you consider social media as a certain type of information source?

In [ ]:
```
safe_unsafe_mapping = { "1": "Definitely unsafe", "2": "Probably unsafe", "3": "Might or might not", "4": "Probably safe", "5": "D
```

In [259…
```
eval_3 = survey_df["Q34"].map(safe_unsafe_mapping).value_counts()
eval_3['Definitely safe'] = 0
eval_3
```

Out[259]:
```
Probably unsafe       22
Might or might not     8
Probably safe          7
Definitely unsafe      6
Definitely safe        0
Name: Q34, dtype: int64
```

In [260…
```
ordered_labels = ["Definitely unsafe", "Probably unsafe", "Might or might not", "Probably safe", "Definitely safe"]
ordered_data = {label: eval_3[label] for label in ordered_labels}
categories = list(ordered_data.keys())
values = list(ordered_data.values())
plt.figure(figsize=(8, 5))  # 가로 8, 세로 5 크기
plt.bar(categories, values,  color='lightgray', edgecolor="gray", alpha=0.7)

plt.xlabel("")
plt.ylabel("")
plt.grid(axis='y', linestyle="--", alpha=0.5)
plt.xticks(fontsize=7)
plt.yticks([])
for i, value in enumerate(ordered_data.values()):
    plt.text(i, value + 1, str(value), ha='center', fontsize=10, color='black')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
```


22
6
8
7
0
Definitely unsafe
Probably unsafe
Might or might not
Probably safe
Definitely safe

In [ ]: