Task 1: Data collection and cleaning

(1) - Collect and extract the data from the relevant website;

Firstly, the necessary library, pandas, is imported to provide functionality for data manipulation and analysis. Additionally, the matplotlib.pyplot library is imported to facilitate the creation of visualizations.

Next, the Excel file, 'the_oscar_award.csv', is read using the pd.read_csv() function.

Output:

```
[2]: # (1) - Collect and extract the data from the relevant website;
     ### Your code here ###
     df=pd.read_csv('the_oscar_award.csv')
     df
```

[2]:

	year_film	year_ceremony	ceremony	category	name	film	winner
0	1927	1928	1	ACTOR	Richard Barthelmess	The Noose	False
1	1927	1928	1	ACTOR	Emil Jannings	The Last Command	True
2	1927	1928	1	ACTRESS	Louise Dresser	A Ship Comes In	False
3	1927	1928	1	ACTRESS	Janet Gaynor	7th Heaven	True
4	1927	1928	1	ACTRESS	Gloria Swanson	Sadie Thompson	False
...	...	...	...	...	...	...	...
10884	2023	2024	96	WRITING (Original Screenplay)	Written by Celine Song	Past Lives	False
10885	2023	2024	96	JEAN HERSHOLT HUMANITARIAN AWARD	NaN	NaN	True

4	1927	1928	1	ACTRESS	Gloria Swanson	Sadie Thompson	False
...	...	...	...	...	...	...	...
10884	2023	2024	96	WRITING (Original Screenplay)	Written by Celine Song	Past Lives	False
10885	2023	2024	96	JEAN HERSHOLT HUMANITARIAN AWARD	NaN	NaN	True
10886	2023	2024	96	HONORARY AWARD	To Angela Bassett, who has inspired audiences ...	NaN	True
10887	2023	2024	96	HONORARY AWARD	To Mel Brooks, for his comedic brilliance, pro...	NaN	True
10888	2023	2024	96	HONORARY AWARD	To Carol Littleton, whose commitment to her cr...	NaN	True

10889 rows × 7 columns

(2a) What is the data structure (data type) you have used to store the data (e.g., DataFrame, Series, Array, List)?

The data structure is DataFrame.

```
[44]: # (2a) What is the data structure (data type) you have used to store the data (e.g., DataFrame, Series, Array, List)?
      ### Your code here ###
      print(type(df))
      #The data structure is DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>
```

(2b) How many rows and columns are there in the data?

There are 10889 rows and 7 columns in the data.

```
[4]: # (2b) How many rows and columns are there in the data?
     ### Your code here ###
     print(df.shape[0])
     print(df.shape[1])
     #10889 rows and 7 columns are there in the data
```

```
10889
7
```

(2c) What is the dimension of the data?
The dimension of the data is 2

[5]:
```
# (2c) What is the dimension of the data?
### Your code here ###
print(df.ndim)
```

```
2
```

(2d) What are the data types (e.g., integer, float) of the data?

The data of 'year_film', 'year_ceremony' and 'ceremony' are integer.
The data of 'category', 'name', and 'film' are objects.
The data of 'winner' is bool.

[6]:
```
# (2d) What are the data types (e.g., integer, float) of the data?
### Your code here ###
print(df.dtypes)
```

```
year_film         int64
year_ceremony     int64
ceremony          int64
category         object
name             object
film             object
winner             bool
dtype: object
```

(2e) Are there any null values or non-numeric data in your dataset?
Yes, the dataset contains null values and non-numeric data.

[45]:
```
# (2e) Are there any null values or non-numeric data in your dataset?
### Your code here ###
is_null = df.isnull().values.any()
is_non_numeric = not np.issubdtype(df.values.dtype, np.number)
if is_null:
    print("The dataset contains null values.")
else:
    print("The dataset does not contain null values.")

if is_non_numeric:
    print("The dataset contains non-numeric data.")
else:
    print("The dataset does not contain non-numeric data.")
#Yes, there are null values and non-numeric data in the dataset.
```

```
The dataset does not contain null values.
The dataset contains non-numeric data.
```

The expression is_null = df.isnull().values.any() checks for the presence of any missing values (NaN) in the DataFrame df. It utilizes the isnull() function to create a boolean mask that identifies which elements in the DataFrame are null. The values.any() method is then used to determine if there is at least one True value in the mask. If there are any missing values in the DataFrame, the variable is_null will be assigned a value of True; otherwise, it will be assigned a value of False.

The expression is_non_numeric = not np.issubdtype(df.values.dtype, np.number) verifies whether the values in the DataFrame df are non-numeric. It employs the np.issubdtype() function from the NumPy library to examine if the data type of the values in df is not a subdtype of np.number, which encompasses both floating-point and integer numeric types. If the values in df are non-numeric, the variable is_non_numeric will be assigned a value of True; otherwise, it will be assigned a value of False.

(3) Perform data cleaning. You are suggested to write functions for the data cleaning process.

df.dropna(subset=['film'], inplace=True): This line drops rows from the DataFrame where the value in the 'film' column is missing (NaN). The dropna() function is used to remove rows with missing values, and the subset parameter is set to 'film' to specify that only the 'film' column should be considered. The inplace=True argument ensures that the original DataFrame is modified in-place (i.e., the changes are applied to the DataFrame directly).

[8]:
```
# 3. Perform data cleaning. You are suggested to write functions for the data cleaning process.
# You can get additional cells in this Jupyter Notebook, if necessary
def drop_missing_films(df):
    df.dropna(subset=['film'], inplace=True)
    return df

df= drop_missing_films(df)
df
```

[8]:

	year_film	year_ceremony	ceremony	category	name	film	winner
0	1927	1928	1	ACTOR	Richard Barthelmess	The Noose	False
1	1927	1928	1	ACTOR	Emil Jannings	The Last Command	True
2	1927	1928	1	ACTRESS	Louise Dresser	A Ship Comes In	False
3	1927	1928	1	ACTRESS	Janet Gaynor	7th Heaven	True
4	1927	1928	1	ACTRESS	Gloria Swanson	Sadie Thompson	False
...	...	...	...	...	...	...	...
10880	2023	2024	96	WRITING (Original Screenplay)	Screenplay - Justine Triet and Arthur Harari	Anatomy of a Fall	True
10881	2023	2024	96	WRITING (Original Screenplay)	Written by David Hemingson	The Holdovers	False
10882	2023	2024	96	WRITING (Original Screenplay)	Written by Bradley Cooper & Josh Singer	Maestro	False
10883	2023	2024	96	WRITING (Original Screenplay)	Screenplay by Samy Burch; Story by Samy Burch ...	May December	False
10884	2023	2024	96	WRITING (Original Screenplay)	Written by Celine Song	Past Lives	False

10570 rows × 7 columns

df[column] = df[column].str.replace(';', ''): This line replaces any semicolons (;) present in the values of the specified column (column) in the DataFrame (df) with an empty string. The str.replace() function is used to perform the replacement operation.

[9]:
```
def remove_semicolon(df, column):
    df[column] = df[column].str.replace(';', '')
    return df[column]

df['film']=remove_semicolon(df, 'film')
df
```

[9]:

	year_film	year_ceremony	ceremony	category	name	film	winner
0	1927	1928	1	ACTOR	Richard Barthelmess	The Noose	False
1	1927	1928	1	ACTOR	Emil Jannings	The Last Command	True
2	1927	1928	1	ACTRESS	Louise Dresser	A Ship Comes In	False
3	1927	1928	1	ACTRESS	Janet Gaynor	7th Heaven	True
4	1927	1928	1	ACTRESS	Gloria Swanson	Sadie Thompson	False
...	...	...	...	...	...	...	...
10880	2023	2024	96	WRITING (Original Screenplay)	Screenplay - Justine Triet and Arthur Harari	Anatomy of a Fall	True
10881	2023	2024	96	WRITING (Original Screenplay)	Written by David Hemingson	The Holdovers	False
10882	2023	2024	96	WRITING (Original Screenplay)	Written by Bradley Cooper & Josh Singer	Maestro	False
10883	2023	2024	96	WRITING (Original Screenplay)	Screenplay by Samy Burch; Story by Samy Burch ...	May December	False
10884	2023	2024	96	WRITING (Original Screenplay)	Written by Celine Song	Past Lives	False

10570 rows × 7 columns

df = df[~df[column].str.match(r'.*[^\x00-\xFF]')]: This line filters the DataFrame df to keep

only the rows where the values in the specified column (column) do not contain any Chinese characters. The str.match() function is used to check if each value matches the provided regular expression pattern. In this case, the pattern r'.*[^\x00-\xFF]' matches any string that contains at least one character outside the ASCII range (i.e., any character that is not within the range of \x00 to \xFF, which corresponds to non-Chinese characters).

```
def drop_chinese_rows(df, column):
    df = df[~df[column].str.match(r'.*[^\x00-\xFF]')]
    return df
df=drop_chinese_rows(df,'name')
df=drop_chinese_rows(df,'film')
df
```

	year_film	year_ceremony	ceremony	category	name	film	winner
0	1927	1928	1	ACTOR	Richard Barthelmess	The Noose	False
1	1927	1928	1	ACTOR	Emil Jannings	The Last Command	True
2	1927	1928	1	ACTRESS	Louise Dresser	A Ship Comes In	False
3	1927	1928	1	ACTRESS	Janet Gaynor	7th Heaven	True
4	1927	1928	1	ACTRESS	Gloria Swanson	Sadie Thompson	False
...	...	...	...	...	...	...	...
10880	2023	2024	96	WRITING (Original Screenplay)	Screenplay - Justine Triet and Arthur Harari	Anatomy of a Fall	True
10881	2023	2024	96	WRITING (Original Screenplay)	Written by David Hemingson	The Holdovers	False
10882	2023	2024	96	WRITING (Original Screenplay)	Written by Bradley Cooper & Josh Singer	Maestro	False
10883	2023	2024	96	WRITING (Original Screenplay)	Screenplay by Samy Burch; Story by Samy Burch ...	May December	False
10884	2023	2024	96	WRITING (Original Screenplay)	Written by Celine Song	Past Lives	False

10569 rows × 7 columns

Task 2: Data Processing and Visualization

1. Analyze the number of nominations and wins in Oscar 2024.

‘oscar_2024_data = df[df['year_ceremony'] == 2024]’ Filters the DataFrame df to include only the rows where the 'year_ceremony' column has a value of 2024.

‘film_counts = oscar_2024_data.groupby('film').agg(total_films=('film', 'count'), winner_films=('winner', 'sum'))’ groups the oscar_2024_data DataFrame by the 'film' column and aggregates the data, which calculates the count of films for each title and the sum of the 'winner' column (which likely contains boolean values indicating whether a film won an award or not).

‘film_counts = film_counts.sort_values('total_films')’ Sorts the film_counts DataFrame by the 'total_films' column in ascending order.

‘fig, ax = plt.subplots(figsize=(12, 10))’
‘film_counts.plot(kind='barh', ax=ax)’
which creates a horizontal bar plot using the plot method of the film_counts DataFrame. The figure size of the plot is set to (12, 10) using plt.subplots(figsize=(12, 10)).

plt.xlabel('Count')
plt.ylabel('Film')
plt.title('Total Films vs. Winner Films - Oscar 2024')
plt.legend(['Total Films', 'Winner Films'])
plt.tight_layout()
which adds labels, title, and legend to the plot, and adjusts the layout to ensure the plot is properly displayed.

Therefore, the number of wins is 23 and the number of nominations is 120.

The consideration of Gestalt principles of perception:

Pattern and Continuity: The principle of Continuity plays a role in how we perceive the overall pattern and structure of the bar chart. The alignment and arrangement of bars along the x-axis create a continuous visual pattern that helps viewers associate bars with their respective categories. This principle helps in perceiving trends, comparisons, or patterns in the data.
Completeness and Closure: The principle of Closure affects how we perceive individual bars as complete objects, despite the absence of any physical boundaries. Viewers tend to mentally close the top of each bar, perceiving them as solid shapes. This principle helps in perceiving the magnitude or value represented by each bar, making it easier to compare and interpret the data.
Grouping and Categorization: The principles of Proximity and Similarity influence how we perceive groups and categories within a bar chart. Bars that are close together or share similar visual attributes are perceived as belonging to the same group or category, such as the color

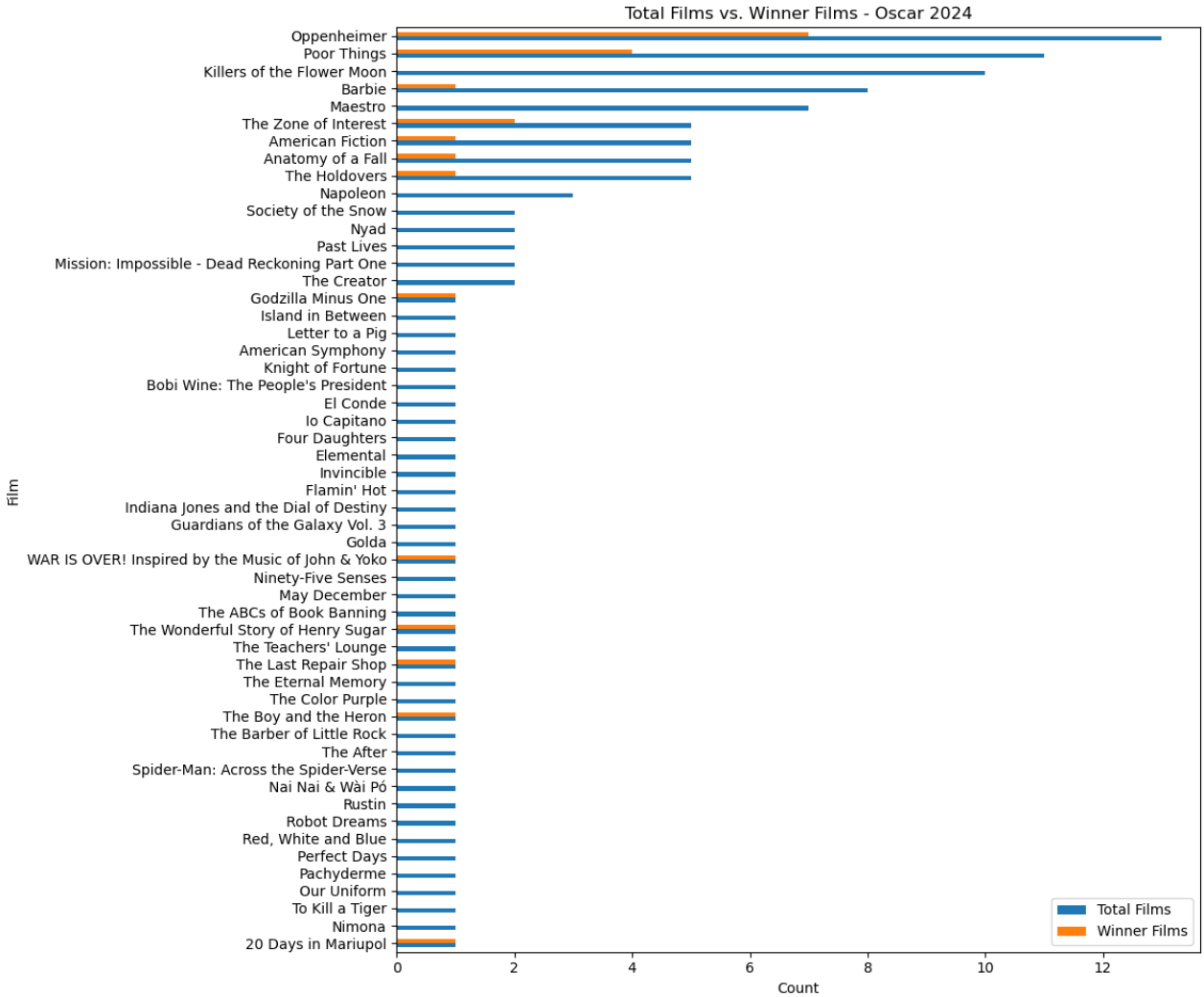
of blue and orange. This can help viewers quickly identify and differentiate different groups or subsets of data in the chart.

```
oscar_2024_data = df[df['year_ceremony'] == 2024]

film_counts = oscar_2024_data.groupby('film').agg(total_films=('film', 'count'), winner_films=('winner', 'sum'))

# Sort the films by total_films in descending order
film_counts = film_counts.sort_values('total_films')

# Plotting the data with adjusted width
fig, ax = plt.subplots(figsize=(12, 10))  # Increase the figure width as needed
film_counts.plot(kind='barh', ax=ax)
plt.xlabel('Count')
plt.ylabel('Film')
plt.title('Total Films vs. Winner Films - Oscar 2024')
plt.legend(['Total Films', 'Winner Films'])
plt.tight_layout()
plt.show()
```



2. Identify top-winning films: Determine the Films which have received the most Wins over the years.

   winning_films = df[df['winner'] == True] filters the original dataset df to include only the rows where the 'winner' column is True, indicating the winning films.

   film_wins = winning_films['film'].value_counts() counts the occurrences of each unique film in the 'film' column.

   most_winning_film = film_wins.idxmax() determines the film with the most wins, which returns the index label of the maximum value in the film_wins Series.

   top_films = film_wins.head(10)
   fig, ax = plt.subplots(figsize=(10, 6))
   plot = top_films.sort_values().plot(kind='barh', ax=ax)
   A horizontal bar chart is created using the plot() method with the 'barh' (horizontal bar) kind. The top 10 films with the most wins are selected using head(10), and the values are sorted for horizontal display using sort_values().

   ```
   for bar in plot.containers[0]:
       if bar.get_width() == top_films.loc[most_winning_film]:
           bar.set_color('orange')
   ```
   To highlight the film with the most wins, each bar in the chart is iterated through using a for loop and plot.containers[0]. The width of each bar is checked using bar.get_width(), and if it matches the count of wins for the most_winning_film, the color of that bar is set to 'orange' using bar.set_color('orange').

   The chart is labeled, titled, and displayed using the plt.xlabel(), plt.ylabel(), plt.title(), and plt.tight_layout() functions. The chart is shown using plt.show()

   Therefore, Titanic has received the most Wins (12 wins) over the years.

The consideration of Gestalt principles of perception:

The Gestalt principles of Proximity and Similarity influence how we group and categorize information in a bar chart. Placing bars close together or using similar visual attributes (like color) helps viewers perceive them as belonging to the same group or category. By leveraging these principles, we can improve the chart's clarity and facilitate quick identification and differentiation of data subsets.

The principle of Continuity affects how we perceive the overall pattern and structure of a bar chart. By aligning and arranging bars along the x-axis in a continuous manner, we create a visual pattern that connects each bar to its corresponding category. This continuity enables viewers to perceive trends, comparisons, and patterns more easily. Ensuring a smooth and

uninterrupted flow of bars along the x-axis enhances the viewer's understanding of category relationships.

The principle of Closure influences how we perceive individual bars as complete objects, even without physical boundaries. When plotting a bar chart, viewers mentally close the top of each bar, perceiving them as solid shapes. This principle assists in understanding the magnitude or value represented by each bar, facilitating comparison and interpretation of the data. Ensuring that bars extend to the baseline or axis line without interruptions leverages the principle of Closure for more accurate data comprehension.

```
:
# Filter the dataset to include only the winning films
winning_films = df[df['winner'] == True]

# Aggregate the data to calculate the total number of wins for each film
film_wins = winning_films['film'].value_counts()

# Get the film with the most wins
most_winning_film = film_wins.idxmax()

# Plotting the data as a horizontal bar chart
top_films = film_wins.head(10)  # Adjust the number of films to display on the plot as needed
fig, ax = plt.subplots(figsize=(10, 6))
plot = top_films.sort_values().plot(kind='barh', ax=ax)  # Sort the values for horizontal display

# Highlight the film with the most wins in orange color
for bar in plot.containers[0]:
    if bar.get_width() == top_films.loc[most_winning_film]:
        bar.set_color('orange')

plt.xlabel('Number of Wins')
plt.ylabel('Film')
plt.title('Top Films with Most Wins')
plt.tight_layout()
plt.show()
```