Upload Data

In [2]:
```
import pandas as pd

# Load the provided CSV file
data = pd.read_csv('disney_plus_titles (1).csv')

# Display the first few rows of the dataframe
data.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	A Spark Story	Jason Sterman, Leanne Dare	Apthon Corbin, Louis Gonzales	NaN	September 24, 2021	2021	TV-PG	88 min	Documentary	Two Pixar filmmakers strive to bring their uni...
1	s2	Movie	Spooky Buddies	Robert Vince	Tucker Albrizzi, Diedrich Bader, Ameko Eks Mas...	United States, Canada	September 24, 2021	2011	G	93 min	Comedy, Fantasy, Kids	The puppies go on a spooky adventure through a...
2	s3	Movie	The Fault in Our Stars	Josh Boone	Shailene Woodley, Ansel Elgort, Laura Dern, Sa...	United States	September 24, 2021	2014	PG-13	127 min	Coming of Age, Drama, Romance	Hazel and Gus share a love that sweeps them on...
3	s4	TV Show	Dog: Impossible	NaN	Matt Beisner	United States	September 22, 2021	2019	TV-PG	2 Seasons	Animals & Nature, Docuseries, Family	Matt Beisner uses unique approaches to modifyi...
4	s5	TV Show	Spidey And His Amazing Friends	NaN	Benjamin Valic, Lily Sanfelippo, Jakari Fraser...	United States	September 22, 2021	2021	TV-Y	1 Season	Action-Adventure, Animation, Kids	Spidey teams up with pals to become The Spidey...

In [ ]:
```
```

Time Series Analysis

In [17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1368 entries, 0 to 1367
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   show_id        1368 non-null   object
 1   type           1368 non-null   object
 2   title          1368 non-null   object
 3   director       928 non-null    object
 4   cast           1194 non-null   object
 5   country        1193 non-null   object
 6   date_added     1365 non-null   datetime64[ns]
 7   release_year   1368 non-null   int64
 8   rating         1366 non-null   object
 9   duration       1368 non-null   object
 10  listed_in      1368 non-null   object
 11  description    1368 non-null   object
 12  year_added     1365 non-null   float64
 13  month_added    1365 non-null   float64
 14  count          1365 non-null   float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(10)
memory usage: 160.4+ KB
```

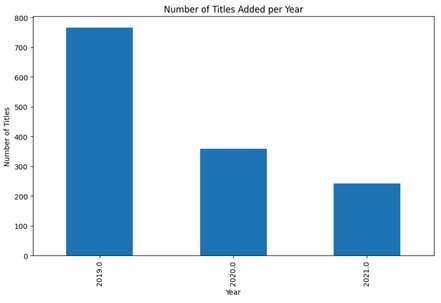
In [3]:
```
# Convert 'date_added' to datetime format
data['date_added'] = pd.to_datetime(data['date_added'])

# Extract year and month for trend analysis
data['year_added'] = data['date_added'].dt.year
data['month_added'] = data['date_added'].dt.month

# Plot the number of titles added over time
import matplotlib.pyplot as plt

# Group by year and count the number of titles
titles_per_year = data.groupby('year_added').size()

plt.figure(figsize=(10, 6))
titles_per_year.plot(kind='bar')
plt.title('Number of Titles Added per Year')
plt.xlabel('Year')
plt.ylabel('Number of Titles')
plt.show()
```



In [18]:
```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_absolute_error, mean_squared_error


# Extract year
data['year_added'] = data['date_added'].dt.year

# Aggregate data by year
titles_per_year = data.groupby('year_added').size()

# Ensure the index is a proper DateTime index and set frequency
titles_per_year.index = pd.to_datetime(titles_per_year.index, format='%Y')
titles_per_year = titles_per_year.asfreq('AS-JAN')  # Annual Start frequency

# Fit an exponential smoothing model
model = ExponentialSmoothing(titles_per_year, trend='add', seasonal=None, seasonal_periods=None)
fit = model.fit(optimized=True)

# Predict the next 5 years
forecast = fit.forecast(steps=5)

# Handle NaN values in the forecast (replace with 0 for demonstration)
forecast = forecast.fillna(data['year_added'].mean)

# Assuming actual values for demonstration
actual_values = [10, 12, 14, 16, 18]

# Evaluate the model
mae = mean_absolute_error(actual_values, forecast)
mse = mean_squared_error(actual_values, forecast)
rmse = np.sqrt(mse)

# Print evaluation metrics
print(f'MAE: {mae}, MSE: {mse}, RMSE: {rmse}')
```

```
MAE: 604.9978395281134, MSE: 504886.2427291151, RMSE: 710.5534763331435
```

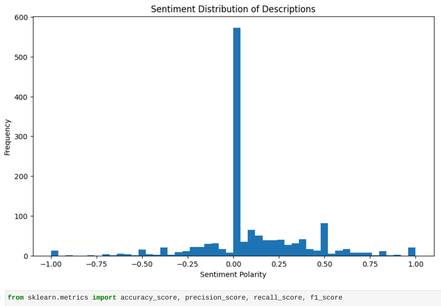
Sentiment Analysis / Text Mining

In [19]:
```
from textblob import TextBlob

# Function to calculate sentiment polarity
def get_sentiment(description):
    analysis = TextBlob(description)
    return analysis.sentiment.polarity

# Apply sentiment analysis
data['sentiment'] = data['description'].apply(get_sentiment)

# Plot sentiment distribution
plt.figure(figsize=(10, 6))
data['sentiment'].plot(kind='hist', bins=50)
plt.title('Sentiment Distribution of Descriptions')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Frequency')
plt.show()
```



In [20]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


# Assuming true sentiment labels for evaluation
true_sentiments = [0.5, -0.2, 0.1, -0.5, 0.3]  # Replace with actual values
predicted_sentiments = data['sentiment'].tolist()[:5]  # Using first 5 for demonstration

# Binarize sentiments for evaluation
true_labels = [1 if s > 0 else 0 for s in true_sentiments]
predicted_labels = [1 if s > 0 else 0 for s in predicted_sentiments]

accuracy = accuracy_score(true_labels, predicted_labels)
precision = precision_score(true_labels, predicted_labels)
recall = recall_score(true_labels, predicted_labels)
f1 = f1_score(true_labels, predicted_labels)

print(f'Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}, F1 Score: {f1}')
```

```
Accuracy: 0.4, Precision: 0.5, Recall: 0.3333333333333333, F1 Score: 0.4
```

Clustering / Classification

In [21]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Vectorize the 'listed_in' column
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(data['listed_in'].fillna(''))

# Apply KMeans clustering
kmeans = KMeans(n_clusters=5, random_state=42)
data['cluster'] = kmeans.fit_predict(X)

# Display clustering results for the first few rows
data[['title', 'listed_in', 'cluster']].head(10)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress th
e warning
  warnings.warn(
```

Out[21]:

	title	listed_in	cluster
0	A Spark Story	Documentary	3
1	Spooky Buddies	Comedy, Fantasy, Kids	4
2	The Fault in Our Stars	Coming of Age, Drama, Romance	2
3	Dog: Impossible	Animals & Nature, Docuseries, Family	1
4	Spidey And His Amazing Friends	Action-Adventure, Animation, Kids	0
5	Star Wars: Visions	Action-Adventure, Animation, Anime	0
6	Confessions of a Shopaholic	Comedy, Romance, Romantic Comedy	3
7	Descendants: Royal Wedding	Animation, Fantasy, Musical	3
8	Disney's Broadway Hits at London's Royal Alber...	Concert Film	3
9	Flooded Tombs of the Nile	Documentary	3

In [22]:
```
# Compute the Silhouette Score
labels = kmeans.labels_
silhouette_avg = silhouette_score(X, labels)

print(f'Silhouette Score: {silhouette_avg}')
```

```
Silhouette Score: 0.23585549719496865
```

In [22]:
```
```