In [1]:
```
import pandas as pd

# Load the dataset (CSV file)
file_path = '/Users/norahmo/Desktop/بيانات_النتائج_المعدلة.csv'
data = pd.read_csv(file_path)

print("Dataset loaded successfully.")

# Check for duplicates before cleaning
duplicates_before = data.duplicated().sum()
print(f"Total duplicates before cleaning: {duplicates_before}")

# Check for missing (null) values before cleaning
missing_values_before = data.isnull().sum().sum()
print(f"Missing values before cleaning: {missing_values_before}")

# Remove duplicate rows
data_cleaned = data.drop_duplicates()
print(f"Duplicates removed. Current shape: {data_cleaned.shape}")

# Drop rows with any missing values (if any)
data_cleaned = data_cleaned.dropna()
print(f"Missing values removed. Current shape: {data_cleaned.shape}")

# Check for duplicates and missing values after cleaning
duplicates_after = data_cleaned.duplicated().sum()
missing_values_after = data_cleaned.isnull().sum().sum()
print(f"Total duplicates after cleaning: {duplicates_after}")
print(f"Missing values after cleaning: {missing_values_after}")

# Method to handle outliers using the Interquartile Range (IQR) method
def remove_outliers_iqr(df, column_name):
    Q1 = df[column_name].quantile(0.25)  # First quartile (25%)
    Q3 = df[column_name].quantile(0.75)  # Third quartile (75%)
    IQR = Q3 - Q1  # Interquartile range
    lower_bound = Q1 - 1.5 * IQR  # Lower bound
    upper_bound = Q3 + 1.5 * IQR  # Upper bound
    filtered_df = df[(df[column_name] >= lower_bound) & (df[column_name]
    print(f"Outliers removed from column '{column_name}'. Rows removed: {
    return filtered_df

# Apply the outlier removal method for numerical columns
numerical_columns = data_cleaned.select_dtypes(include=['float64', 'int64
print(f"Numerical columns detected for outlier removal: {list(numerical_c

for column in numerical_columns:
    data_cleaned = remove_outliers_iqr(data_cleaned, column)

# Check dataset statistics after outlier removal
print(f"Shape of dataset after outlier removal: {data_cleaned.shape}")

# Verify that the dataset is clean
```

```
if duplicates_after == 0 and missing_values_after == 0:
    print("Dataset is clean. No duplicates or missing values remain.")
else:
    print("Some issues remain after cleaning.")

# Save the cleaned dataset (to a CSV file)
cleaned_file_path = '/Users/norahmo/Desktop/cleaned_بيانات_النتائج_المعدلة
data_cleaned.to_csv(cleaned_file_path, index=False)

# Print the cleaned file path
print(f"Cleaned dataset saved at: {cleaned_file_path}")
```

```
Dataset loaded successfully.
Total duplicates before cleaning: 0
Missing values before cleaning: 0
Duplicates removed. Current shape: (8790, 9)
Missing values removed. Current shape: (8790, 9)
Total duplicates after cleaning: 0
Missing values after cleaning: 0
Numerical columns detected for outlier removal: ['release_year']
Outliers removed from column 'release_year'. Rows removed: 717
Shape of dataset after outlier removal: (8073, 9)
Dataset is clean. No duplicates or missing values remain.
Cleaned dataset saved at: /Users/norahmo/Desktop/cleaned_بيانات_النتائج_ال
معدلة.csv
```

In [3]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
import numpy as np
import warnings

# Suppress warnings
warnings.filterwarnings("ignore")

# Load dataset
file_path = '/Users/norahmo/Desktop/cleaned_بيانات_النتائج_المعدلة.csv'
try:
    data = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"File not found at {file_path}")

# Quick inspection
print("Dataset Head:\n", data.head())
print("\nDataset Info:\n")
print(data.info())

# Handling missing or inconsistent values
```

```
data['duration'] = data['duration'].fillna("0 min")  # Fill missing durat
data['duration_minutes'] = data['duration'].str.extract(r'(\d+)', expand=
data['rating'] = data['rating'].fillna("Unknown")  # Fill missing ratings
data = data.dropna(subset=['type', 'release_year', 'listed_in'])  # Drop

# Model 1: Predict 'type' (Movie or TV Show)
features_model1 = ['release_year', 'rating', 'duration_minutes', 'listed_
target_model1 = 'type'

# Model 2: Predict 'rating'
features_model2 = ['release_year', 'duration_minutes', 'listed_in']
target_model2 = 'rating'

# Define categorical features for encoding
categorical_features_model1 = ['rating', 'listed_in']
categorical_features_model2 = ['listed_in']

# Preprocessing pipeline for Model 1
preprocessor_model1 = ColumnTransformer(
    transformers=[
        ('num', SimpleImputer(strategy='median'), ['release_year', 'durat
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_featu
    ]
)

# Preprocessing pipeline for Model 2
preprocessor_model2 = ColumnTransformer(
    transformers=[
        ('num', SimpleImputer(strategy='median'), ['release_year', 'durat
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_featu
    ]
)

# Splitting data for Model 1
X_model1 = data[features_model1]
y_model1 = LabelEncoder().fit_transform(data[target_model1])  # Encode 'M
X_train_m1, X_test_m1, y_train_m1, y_test_m1 = train_test_split(X_model1,

# Splitting data for Model 2
X_model2 = data[features_model2]
y_model2 = LabelEncoder().fit_transform(data[target_model2])  # Encode 'r
X_train_m2, X_test_m2, y_train_m2, y_test_m2 = train_test_split(X_model2,

# Model 1: Logistic Regression Pipeline
pipeline_model1 = Pipeline([
    ('preprocessor', preprocessor_model1),
    ('classifier', LogisticRegression(max_iter=1000))
])

# Model 2: Random Forest Pipeline
pipeline_model2 = Pipeline([
    ('preprocessor', preprocessor_model2),
    ('classifier', RandomForestClassifier(random_state=42))
```

```
])

# Train both models
pipeline_model1.fit(X_train_m1, y_train_m1)
pipeline_model2.fit(X_train_m2, y_train_m2)

# Predictions and evaluation
y_pred_m1 = pipeline_model1.predict(X_test_m1)
y_pred_m2 = pipeline_model2.predict(X_test_m2)

# Classification reports
report_model1 = classification_report(y_test_m1, y_pred_m1)
report_model2 = classification_report(y_test_m2, y_pred_m2)

# Print results
print("Model 1: Logistic Regression (Movie vs. TV Show)")
print(report_model1)
print(f"Model 1 Accuracy: {accuracy_score(y_test_m1, y_pred_m1)}")

print("\nModel 2: Random Forest Classifier (Predict Show Rating)")
print(report_model2)
print(f"Model 2 Accuracy: {accuracy_score(y_test_m2, y_pred_m2)}")
```

```
Dataset Head:
   show_id     type                  title          date_added  release_ye
ar  \
0      s1    Movie   Dick Johnson Is Dead  September 25, 2021          202
0
1      s2  TV Show          Blood & Water  September 24, 2021          202
1
2      s3  TV Show              Ganglands  September 24, 2021          202
1
3      s4  TV Show  Jailbirds New Orleans  September 24, 2021          202
1
4      s5  TV Show           Kota Factory  September 24, 2021          202
1

  rating   duration                                          listed_in  \
0  PG-13     90 min                                       Documentaries
1  TV-MA  2 Seasons    International TV Shows, TV Dramas, TV Mysteries
2  TV-MA   1 Season  Crime TV Shows, International TV Shows, TV Act...
3  TV-MA   1 Season                          Docuseries, Reality TV
4  TV-MA  2 Seasons  International TV Shows, Romantic TV Shows, TV ...

                                         description
0  As her father nears the end of his life, filmm...
1  After crossing paths at a party, a Cape Town t...
2  To protect his family from a powerful drug lor...
3  Feuds, flirtations and toilet talk go down amo...
4  In a city of coaching centers known to train I...

Dataset Info:

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

```
RangeIndex: 8073 entries, 0 to 8072
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8073 non-null   object
 1   type          8073 non-null   object
 2   title         8073 non-null   object
 3   date_added    8073 non-null   object
 4   release_year  8073 non-null   int64 
 5   rating        8073 non-null   object
 6   duration      8073 non-null   object
 7   listed_in     8073 non-null   object
 8   description   8073 non-null   object
dtypes: int64(1), object(8)
memory usage: 567.8+ KB
None
Model 1: Logistic Regression (Movie vs. TV Show)
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1093
           1       1.00      1.00      1.00       522

    accuracy                           1.00      1615
   macro avg       1.00      1.00      1.00      1615
weighted avg       1.00      1.00      1.00      1615

Model 1 Accuracy: 0.9987616099071207

Model 2: Random Forest Classifier (Predict Show Rating)
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           2       0.20      0.17      0.18        12
           3       0.48      0.29      0.36        45
           4       0.37      0.33      0.35        79
           5       0.46      0.43      0.45       132
           6       0.41      0.43      0.42       410
           7       0.09      0.05      0.06        42
           8       0.56      0.62      0.59       623
           9       0.23      0.20      0.21       147
          10       0.63      0.64      0.64        56
          11       0.56      0.47      0.51        66
          12       0.00      0.00      0.00         0
          13       0.00      0.00      0.00         0

    accuracy                           0.47      1615
   macro avg       0.31      0.28      0.29      1615
weighted avg       0.46      0.47      0.46      1615

Model 2 Accuracy: 0.46873065015479876
```

In [5]:
```
from sklearn.metrics import recall_score

# Model 1: Predictions
```

```
y_pred_m1 = pipeline_model1.predict(X_test_m1)

# Calculate accuracy, precision, and recall
accuracy_m1 = accuracy_score(y_test_m1, y_pred_m1)
precision_m1 = precision_score(y_test_m1, y_pred_m1, average='weighted')
recall_m1 = recall_score(y_test_m1, y_pred_m1, average='weighted')  # wei

# Print the results
print("Model 1 Evaluation Metrics:")
print(f"Accuracy: {accuracy_m1:.4f}")
print(f"Precision: {precision_m1:.4f}")
print(f"Recall: {recall_m1:.4f}")
```

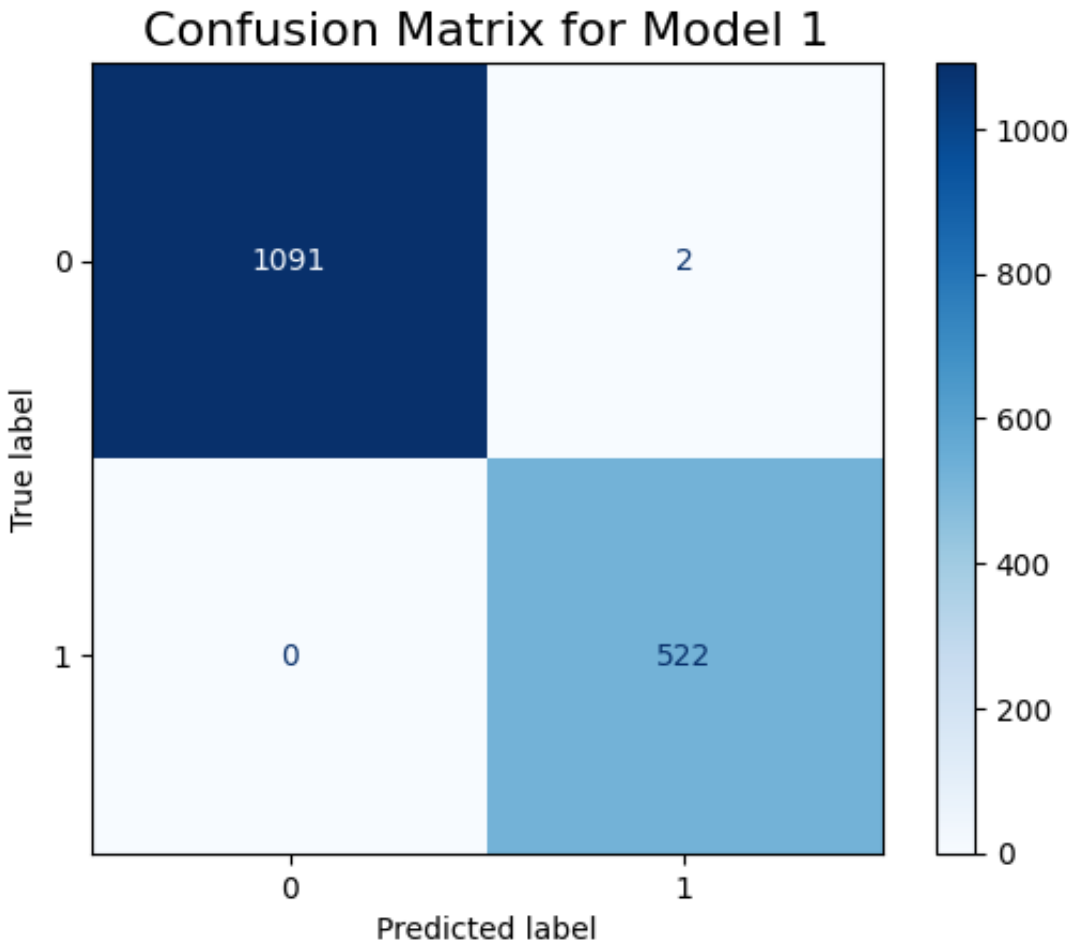
```
Model 1 Evaluation Metrics:
Accuracy: 0.9988
Precision: 0.9988
Recall: 0.9988
```

In [7]:
```
pip install matplotlib
```

```
Collecting matplotlib
  Downloading matplotlib-3.9.2-cp313-cp313-macosx_11_0_arm64.whl.metadata
(11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.1-cp313-cp313-macosx_11_0_arm64.whl.metadata (
5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.55.0-cp313-cp313-macosx_10_13_universal2.whl.met
adata (164 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.7-cp313-cp313-macosx_11_0_arm64.whl.metadata
(6.3 kB)
Requirement already satisfied: numpy>=1.23 in /Library/Frameworks/Python.f
ramework/Versions/3.13/lib/python3.13/site-packages (from matplotlib) (2.
1.3)
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Pyth
on.framework/Versions/3.13/lib/python3.13/site-packages (from matplotlib)
(24.2)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-11.0.0-cp313-cp313-macosx_11_0_arm64.whl.metadata (9.
1 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.0-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in /Library/Framework
s/Python.framework/Versions/3.13/lib/python3.13/site-packages (from matplo
tlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.fram
ework/Versions/3.13/lib/python3.13/site-packages (from python-dateutil>=2.
7->matplotlib) (1.16.0)
Downloading matplotlib-3.9.2-cp313-cp313-macosx_11_0_arm64.whl (7.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 167.4 kB/s eta 0:0
0:0000:0200:04
Downloading contourpy-1.3.1-cp313-cp313-macosx_11_0_arm64.whl (255 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.55.0-cp313-cp313-macosx_10_13_universal2.whl (2.8
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 58.0 kB/s eta 0:00:
00a 0:00:02m
Downloading kiwisolver-1.4.7-cp313-cp313-macosx_11_0_arm64.whl (63 kB)
Downloading pillow-11.0.0-cp313-cp313-macosx_11_0_arm64.whl (3.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 65.2 kB/s eta 0:00:
00a 0:00:02
Downloading pyparsing-3.2.0-py3-none-any.whl (106 kB)
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, c
ycler, contourpy, matplotlib
Successfully installed contourpy-1.3.1 cycler-0.12.1 fonttools-4.55.0 kiwi
solver-1.4.7 matplotlib-3.9.2 pillow-11.0.0 pyparsing-3.2.0

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: pip3 install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

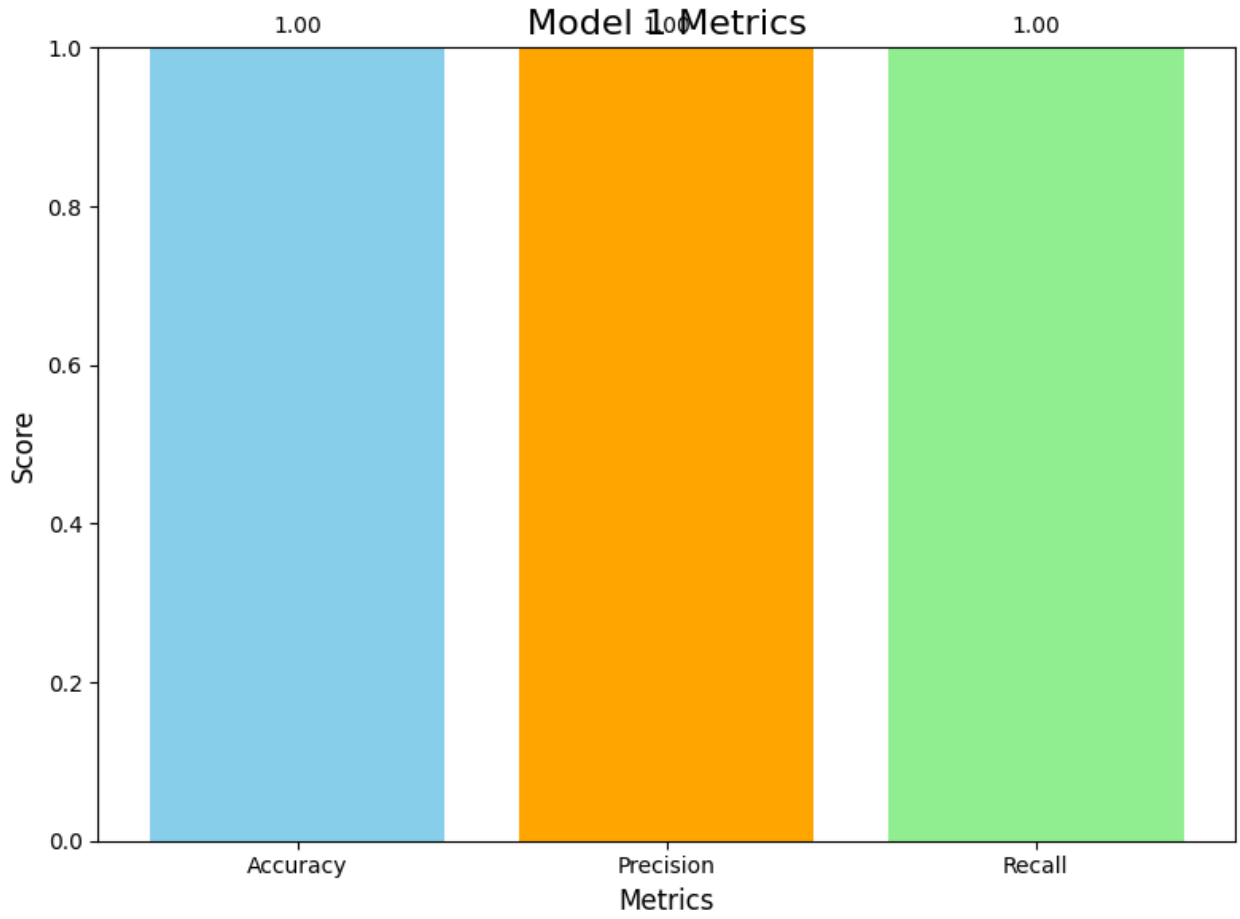
In [10]:
```
%matplotlib inline
```

In [13]:
```
 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Calculate confusion matrix
conf_matrix = confusion_matrix(y_test_m1, y_pred_m1)

# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_label
disp.plot(cmap='Blues', values_format='d')

# Add title
plt.title('Confusion Matrix for Model 1', fontsize=16)
plt.show()
```



In [15]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, accuracy_score

# Calculate metrics for Model 1
accuracy_m1 = accuracy_score(y_test_m1, y_pred_m1)
precision_m1 = precision_score(y_test_m1, y_pred_m1, average='weighted')
recall_m1 = recall_score(y_test_m1, y_pred_m1, average='weighted')

# Metrics and values
metrics = ['Accuracy', 'Precision', 'Recall']
values = [accuracy_m1, precision_m1, recall_m1]
```

```
# Plot
plt.figure(figsize=(8, 6))
plt.bar(metrics, values, color=['skyblue', 'orange', 'lightgreen'])
plt.ylim(0, 1)
plt.title('Model 1 Metrics', fontsize=16)
plt.ylabel('Score', fontsize=12)
plt.xlabel('Metrics', fontsize=12)

# Annotate the bars
for i, v in enumerate(values):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontsize=10)

plt.tight_layout()
plt.show()
```

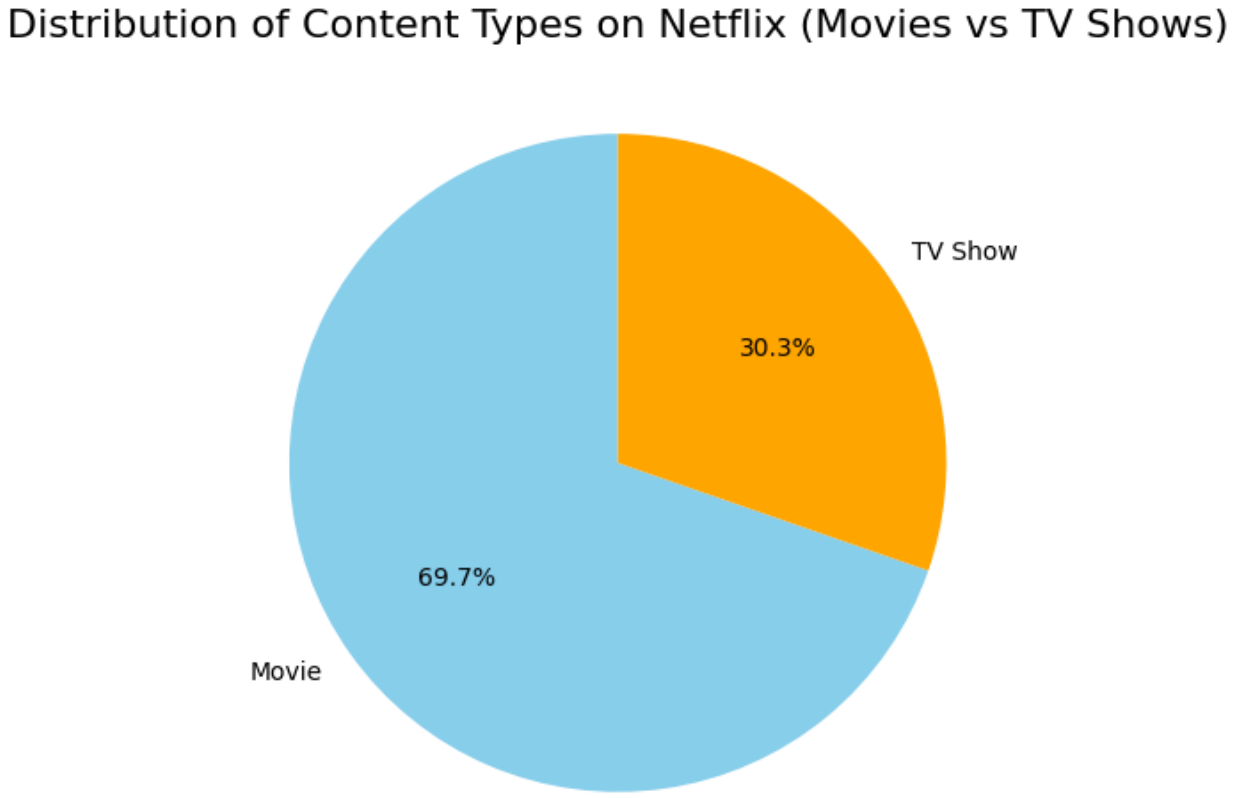


In [2]:
```
import pandas as pd

# Define the file path (ensure it is a string with correct quotes)
file_path = '/Users/norahmo/Desktop/بيانات_النتائج_المعدلة.csv'

# Read the data from the CSV file
data = pd.read_csv(file_path)

# Display the first few rows of the dataset for inspection
print(data.head())
```

```
  show_id     type                  title         date_added  release_yea
r  \
0      s1    Movie   Dick Johnson Is Dead  September 25, 2021          202
0
1      s2  TV Show          Blood & Water  September 24, 2021          202
1
2      s3  TV Show              Ganglands  September 24, 2021          202
1
3      s4  TV Show  Jailbirds New Orleans  September 24, 2021          202
1
4      s5  TV Show           Kota Factory  September 24, 2021          202
1

  rating   duration                                           listed_in  \
0  PG-13     90 min                                       Documentaries
1  TV-MA  2 Seasons    International TV Shows, TV Dramas, TV Mysteries
2  TV-MA   1 Season  Crime TV Shows, International TV Shows, TV Act...
3  TV-MA   1 Season                          Docuseries, Reality TV
4  TV-MA  2 Seasons  International TV Shows, Romantic TV Shows, TV ...

                                          description
0  As her father nears the end of his life, filmm...
1  After crossing paths at a party, a Cape Town t...
2  To protect his family from a powerful drug lor...
3  Feuds, flirtations and toilet talk go down amo...
4  In a city of coaching centers known to train I...
```

In [5]:

```python
import matplotlib.pyplot as plt

# حساب توزيع الأنواع (Movies vs TV Shows)
type_distribution = data['type'].value_counts()

# رسم الدائرة
plt.figure(figsize=(8, 6))
type_distribution.plot(kind='pie', autopct='%1.1f%%', startangle=90, colo

# إضافة عنوان للرسم
plt.title('Distribution of Content Types on Netflix (Movies vs TV Shows)'

# إزالة تسمية المحور الافتراضية
plt.ylabel('')

# حفظ الصورة
plt.savefig("netflix_content_distribution.png")

# عرض الرسم
plt.show()
```


Distribution of Content Types on Netflix (Movies vs TV Shows)
TV Show
30.3%
69.7%
Movie

In [ ]:

In [ ]: