In [4]:

```python
# Collaborative Filtering
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
data = {
    'User': ['User1', 'User2', 'User3', 'User4'],
    'Movie A': [5, 4, 0, 0],
    'Movie B': [4, 0, 0, 0],
    'Movie C': [0, 0, 5, 4],
    'Movie D': [0, 3, 4, 5],
    'Movie E': [0, 0, 0, 5]
}
ratings = pd.DataFrame(data)
ratings.set_index('User', inplace=True)
item_similarity = cosine_similarity(ratings.T)
similarity_df = pd.DataFrame(item_similarity, index=ratings.columns, columns=ratings.columns)
def collaborative_filtering(user, ratings, similarity_df):
    user_ratings = ratings.loc[user]
    scores = {}
    for item in ratings.columns:
        if user_ratings[item] == 0:
            sim_items = similarity_df[item]
            rated_items = user_ratings[user_ratings > 0].index
            score = sum(sim_items[rated_item] * user_ratings[rated_item] for rated_item in rated_items)
            scores[item] = score
    return sorted(scores.items(), key=lambda x: x[1], reverse=True)
user_to_recommend = 'User1'
print(f"Collaborative Filtering Recommendations for {user_to_recommend}:")
print(collaborative_filtering(user_to_recommend, ratings, similarity_df))
```

```
Collaborative Filtering Recommendations for User1:
[('Movie D', 1.3251783128981587), ('Movie C', 0.0), ('Movie E', 0.0)]
```

In [9]:

```
# Content based Filtering
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
metadata = {
'Movie A': [1, 0, 1],
'Movie B': [1, 1, 0],
'Movie C': [0, 1, 1],
'Movie D': [0, 0, 1],
'Movie E': [1, 0, 0],
}
metadata_df = pd.DataFrame(metadata, index=['Action', 'Comedy', 'Drama']).T
content_similarity = cosine_similarity(metadata_df)
content_similarity_df = pd.DataFrame(content_similarity, index=metadata_df.index,
columns=metadata_df.index)
data = {
'User': ['User1', 'User2', 'User3', 'User4'],
'Movie A': [5, 4, 0, 0],
'Movie B': [4, 0, 0, 0],
'Movie C': [0, 0, 5, 4],
'Movie D': [0, 3, 4, 5],
'Movie E': [0, 0, 0, 5]
}
ratings = pd.DataFrame(data)
ratings.set_index('User', inplace=True)
def content_based_filtering(user, ratings, content_similarity_df):
    user_ratings = ratings.loc[user]
    scores = {}
    for item in ratings.columns:
        if user_ratings[item] == 0:
            sim_items = content_similarity_df[item]
            rated_items = user_ratings[user_ratings > 0].index
            score = sum(sim_items[rated_item] * user_ratings[rated_item] for rated_item in rated_items)
            scores[item] = score
    return sorted(scores.items(), key=lambda x: x[1], reverse=True)
user_to_recommend = 'User1'
print(f"Content-Based Filtering Recommendations for {user_to_recommend}:")
print(content_based_filtering(user_to_recommend, ratings, content_similarity_df))
```

```
Content-Based Filtering Recommendations for User1:
[('Movie E', 6.363961030678928), ('Movie C', 4.499999999999999), ('Movie D', 3.5355339059327373)]
```

In [21]:

```python
# TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
movies = {
    "Movie": ["Terminator", "Alien", "Predator", "Interstellar", "Gravity"],
    "Description": [
        "A robot is sent to kill a woman in the past.",
        "A crew finds a dangerous alien on their spaceship.",
        "A soldier fights an alien in the jungle.",
        "Astronauts travel through space to find a new home.",
        "Astronauts try to survive after an accident in space."
    ]
}
df = pd.DataFrame(movies)
vectorizer = TfidfVectorizer(stop_words="english")
tfidf_matrix = vectorizer.fit_transform(df['Description'])
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())
print(tfidf_df)
```

```
   accident     alien  astronauts      crew  dangerous    fights     finds  \
0   0.00000  0.000000    0.000000  0.000000   0.000000  0.000000  0.000000   
1   0.00000  0.374105    0.000000  0.463693   0.463693  0.000000  0.463693   
2   0.00000  0.422242    0.000000  0.000000   0.000000  0.523358  0.000000   
3   0.00000  0.000000    0.388988  0.000000   0.000000  0.000000  0.000000   
4   0.48214  0.000000    0.388988  0.000000   0.000000  0.000000  0.000000   

      home    jungle      kill  ...      past     robot      sent   soldier  \
0  0.00000  0.000000  0.447214  ...  0.447214  0.447214  0.447214  0.000000   
1  0.00000  0.000000  0.000000  ...  0.000000  0.000000  0.000000  0.000000   
2  0.00000  0.523358  0.000000  ...  0.000000  0.000000  0.000000  0.523358   
3  0.48214  0.000000  0.000000  ...  0.000000  0.000000  0.000000  0.000000   
4  0.00000  0.000000  0.000000  ...  0.000000  0.000000  0.000000  0.000000   

      space  spaceship  survive   travel      try     woman  
0  0.000000   0.000000  0.00000  0.00000  0.00000  0.447214  
1  0.000000   0.463693  0.00000  0.00000  0.00000  0.000000  
2  0.000000   0.000000  0.00000  0.00000  0.00000  0.000000  
3  0.388988   0.000000  0.00000  0.48214  0.00000  0.000000  
4  0.388988   0.000000  0.48214  0.00000  0.48214  0.000000  

[5 rows x 21 columns]
```

In [23]:

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

# Define user-item ratings matrix
data = {
    'User': ['User1', 'User2', 'User3', 'User4'],
    'Movie A': [5, 4, 0, 0],
    'Movie B': [4, 0, 0, 0],
    'Movie C': [0, 0, 5, 4],
    'Movie D': [0, 3, 4, 5],
    'Movie E': [0, 0, 0, 5]
}
ratings = pd.DataFrame(data).set_index('User')

# Compute item similarity
similarity_df = pd.DataFrame(cosine_similarity(ratings.T),index=ratings.columns,columns=ratings.columns)

def recommend(user, ratings, similarity_df):
    user_ratings = ratings.loc[user]
    scores = {
        item: sum(similarity_df[item][rated] * user_ratings[rated]
                  for rated in user_ratings[user_ratings > 0].index)
        for item in ratings.columns if user_ratings[item] == 0
    }
    return sorted(scores.items(), key=lambda x: x[1], reverse=True)

# Get recommendations
user_to_recommend = 'User1'
print(f"Collaborative Filtering Recommendations for {user_to_recommend}: {recommend(user_to_recommend, ratings, simila
```

```
Collaborative Filtering Recommendations for User1: [('Movie D', 1.3251783128981587), ('Movie C', 0.0), ('Movie E', 0.
0)]
```

In [19]:

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

# Define movie metadata
metadata = {
    'Movie A': [1, 0, 1],
    'Movie B': [1, 1, 0],
    'Movie C': [0, 1, 1],
    'Movie D': [0, 0, 1],
    'Movie E': [1, 0, 0]
}
metadata_df = pd.DataFrame(metadata, index=['Action', 'Comedy', 'Drama']).T

# Compute content similarity
content_similarity_df = pd.DataFrame(cosine_similarity(metadata_df),
                                     index=metadata_df.index, columns=metadata_df.index)

# Define user ratings
data = {
    'User': ['User1', 'User2', 'User3', 'User4'],
    'Movie A': [5, 4, 0, 0],
    'Movie B': [4, 0, 0, 0],
    'Movie C': [0, 0, 5, 4],
    'Movie D': [0, 3, 4, 5],
    'Movie E': [0, 0, 0, 5]
}
ratings = pd.DataFrame(data).set_index('User')

def recommend(user, ratings, similarity_df):
    user_ratings = ratings.loc[user]
    scores = {
        item: sum(similarity_df[item][rated] * user_ratings[rated]
                  for rated in user_ratings[user_ratings > 0].index)
        for item in ratings.columns if user_ratings[item] == 0
    }
    return sorted(scores.items(), key=lambda x: x[1], reverse=True)

# Get recommendations
user_to_recommend = 'User1'
print(f"Content-Based Filtering Recommendations for {user_to_recommend}: {recommend(user_to_recommend, ratings, conten
Content-Based Filtering Recommendations for User1: [('Movie E', 6.363961030678928), ('Movie C', 4.499999999999999),
('Movie D', 3.5355339059327373)]
```