In [1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# load data
data = pd.read_csv(r'C:\Users\YHD\Documents\PycharmProjects\DS Prj\movieReplicationSet.csv')

movie_ratings = data.iloc[:, 0:400]
```

In [2]:
```
# Check if there are any completely empty rows
empty_rows = movie_ratings.index[movie_ratings.isna().all(axis=1)]

# Check if there are any completely empty columns
empty_cols = movie_ratings.columns[movie_ratings.isna().all(axis=0)]

print(f"completely empty rows: {empty_rows}")
print(f"completely empty columns: {empty_cols}")
```

```
completely empty rows: Index([896], dtype='int64')
completely empty columns: Index([], dtype='object')
```

In [3]:
```
# remove completely empty row
movie_ratings = movie_ratings.drop(index=896)
```

In [4]:
```
# Blend-and-fill
# Calculate the average rating of each movie (column average)
movie_means = movie_ratings.mean(axis=0)

# Calculate the average rating for each user (row average)
user_means = movie_ratings.mean(axis=1)

# For each missing value, fill it with the average of the average rating of the corresponding movie and the average rating of the co
for row in movie_ratings.index:
    for col in movie_ratings.columns:
        if pd.isna(movie_ratings.at[row, col]):
            # Use a 50/50 mixed average
            movie_ratings.at[row, col] = (movie_means[col] + user_means[row]) / 2
```

In [5]:
```
# Build regression model
# Get a list of movie names
movie_names = movie_ratings.columns

results_list = []

# Traverse each movie and perform the same analysis process
for target_movie in movie_names:
    model_cod = {}

    # For each other movie, build a model
    for movie in movie_names:
        if movie != target_movie:

            X = movie_ratings[movie].values.reshape(-1, 1)  # Predictor
            y = movie_ratings[target_movie].values          # Response

            # model
            model = LinearRegression()
            model.fit(X, y)

            # Predict and evaluate
            y_pred = model.predict(X)
            cod = r2_score(y, y_pred)
            model_cod[movie] = cod

    # Find the model with the most accurate prediction
    best_predictor = max(model_cod, key=model_cod.get)
    best_cod = model_cod[best_predictor]

    #Add results to list
    results_list.append([target_movie, best_predictor, best_cod])

# Convert the result list to DataFrame
results = pd.DataFrame(results_list, columns=['Target_Movie', 'Best_Predictor', 'COD'])

# Save results to CSV file
results.to_csv(r'C:\Users\YHD\Documents\PycharmProjects\DS Prj\analysis_results.csv', index=False)
```

In [6]:
```
# COD average
average_COD = results['COD'].mean()
print(f"The average COD of the 400 simple linear regression models is: {average_COD}")
```

```
The average COD of the 400 simple linear regression models is: 0.42378171067196035
```

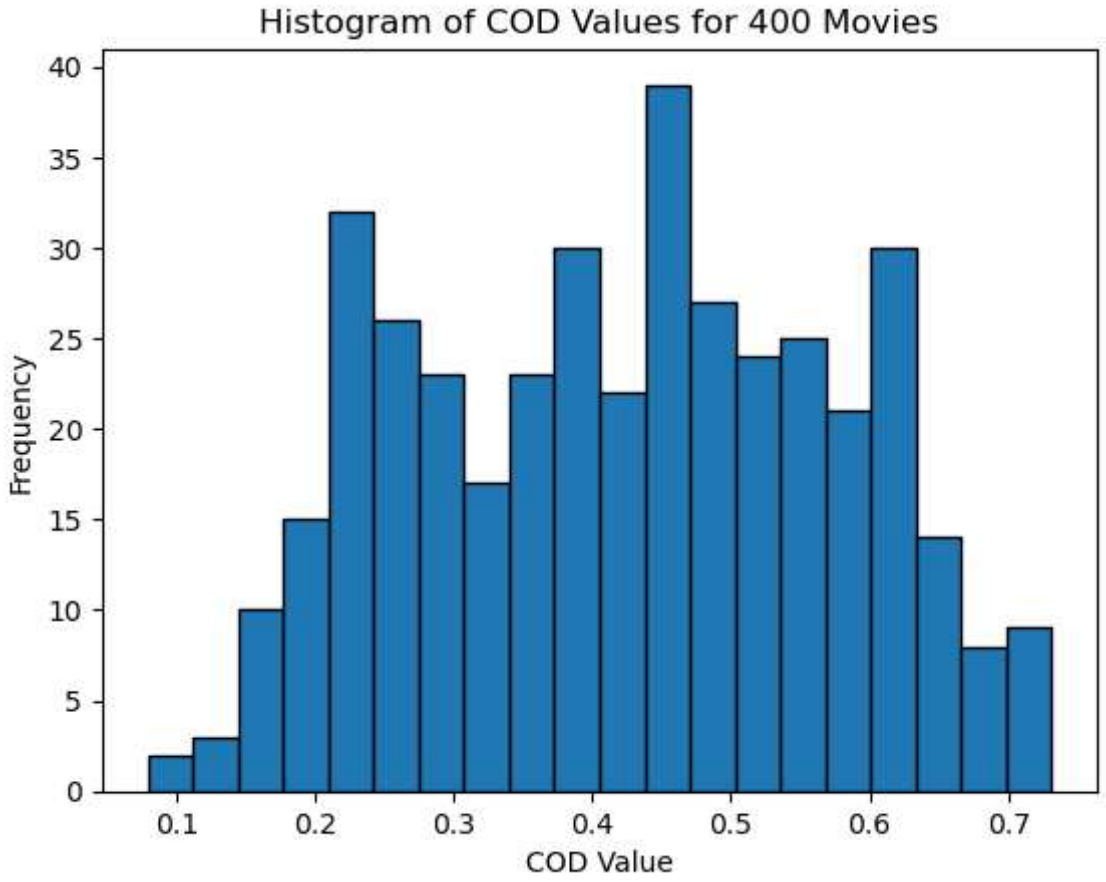
In [7]:
```
# Calculate the number of movies with a COD value lower than 0.5
num_cod_below_0_5 = results[results['COD'] < 0.5].shape[0]

print(f"Number of COD values below 0.5: {num_cod_below_0_5}")
```

```
Number of COD values below 0.5: 265
```

In [8]:
```
# Calculate the number of movies with a COD value lower than 0.6
num_cod_below_0_5 = results[results['COD'] < 0.6].shape[0]

print(f"Number of COD values below 0.6: {num_cod_below_0_5}")
```

```
Number of COD values below 0.6: 339
```

In [9]:
```
import matplotlib.pyplot as plt
results = pd.read_csv(r'C:\Users\YHD\Documents\PycharmProjects\DS Prj\analysis_results.csv')

# Create a histogram of COD values
plt.hist(results['COD'], bins=20, edgecolor='black')
plt.title('Histogram of COD Values for 400 Movies')
plt.xlabel('COD Value')
plt.ylabel('Frequency')
plt.show()
```



In [10]:
```
# Find the 10 movies with the highest and lowest COD values
top_10_predictable = results.nlargest(10, 'COD')
bottom_10_predictable = results.nsmallest(10, 'COD')

print("Top 10 Most Predictable Movies:")
top_10_predictable
```

```
Top 10 Most Predictable Movies:
```

Out[10]:

	Target_Movie	Best_Predictor	COD
203	Erik the Viking (1989)	I.Q. (1994)	0.731507
208	I.Q. (1994)	Erik the Viking (1989)	0.731507
377	The Lookout (2007)	Patton (1970)	0.713554
395	Patton (1970)	The Lookout (2007)	0.713554
240	The Bandit (1996)	Best Laid Plans (1999)	0.711222
249	Best Laid Plans (1999)	The Bandit (1996)	0.711222
282	Congo (1995)	The Straight Story (1999)	0.700569
287	The Straight Story (1999)	Congo (1995)	0.700569
334	The Final Conflict (1981)	The Lookout (2007)	0.700188
300	Ran (1985)	Heavy Traffic (1973)	0.692734

In [11]:
```
print("Top 10 Least Predictable Movies:")
bottom_10_predictable
```

```
Top 10 Least Predictable Movies:
```

Out[11]:

	Target_Movie	Best_Predictor	COD
80	Avatar (2009)	Bad Boys (1995)	0.079485
95	Interstellar (2014)	Torque (2004)	0.111343
9	Black Swan (2010)	Sorority Boys (2002)	0.117080
55	Clueless (1995)	Escape from LA (1996)	0.141426
190	The Cabin in the Woods (2012)	The Evil Dead (1981)	0.143887
319	La La Land (2016)	The Lookout (2007)	0.148514
292	Titanic (1997)	Cocktail (1988)	0.154136
41	13 Going on 30 (2004)	Can't Hardly Wait (1998)	0.160164
14	The Fast and the Furious (2001)	Terminator 3: Rise of the Machines (2003)	0.168991
248	Grown Ups 2 (2013)	The Core (2003)	0.171119

In [12]:
```
# ----------------------------end of question 1, start of question 2----------------------------
```

In [13]:
```
# load result form question 1
selected_movies = top_10_predictable['Target_Movie'].tolist() + bottom_10_predictable['Target_Movie'].tolist()
```

In [14]:
```
additional_predictors = data.iloc[:, 474:477]
```

In [15]:
```
# combine data
combined_data = pd.concat([movie_ratings, additional_predictors], axis=1)

# Now delete any rows containing missing values
combined_data_cleaned = combined_data.dropna(subset=['Gender identity (1 = female; 2 = male; 3 = self-described)'])

# Initialize an empty list to store the multiple regression analysis results of each movie
multi_regress_results = []

# Then loop through the movie names to build a multiple regression model for each movie
for target_movie in selected_movies:
    # Get the rating of the target movie
    y = combined_data_cleaned[target_movie].values

    # Get the rating of the best predictor found in the first question
    best_predictor = results.loc[results['Target_Movie'] == target_movie, 'Best_Predictor'].iloc[0]
    X_best = combined_data_cleaned[best_predictor].values.reshape(-1, 1)

    # Combine the best predictor's score with additional predictors
    X = np.hstack((X_best, combined_data_cleaned[['Gender identity (1 = female; 2 = male; 3 = self-described)', 'Are you an only chi

    # Build and fit multiple regression models
    model = LinearRegression()
    model.fit(X, y)

    # Make predictions and estimates
    y_pred = model.predict(X)
    r2 = r2_score(y, y_pred)


    print(f"Movie '{target_movie}''s multiple regression model R^2 value: {r2}")
    multi_regress_results.append({'Target_Movie': target_movie, 'R2_Score': r2})


multi_regress_results_df = pd.DataFrame(multi_regress_results)

# save result
output_file = r'C:\Users\YHD\Documents\PycharmProjects\DS Prj\multi_regress_results.csv'
```

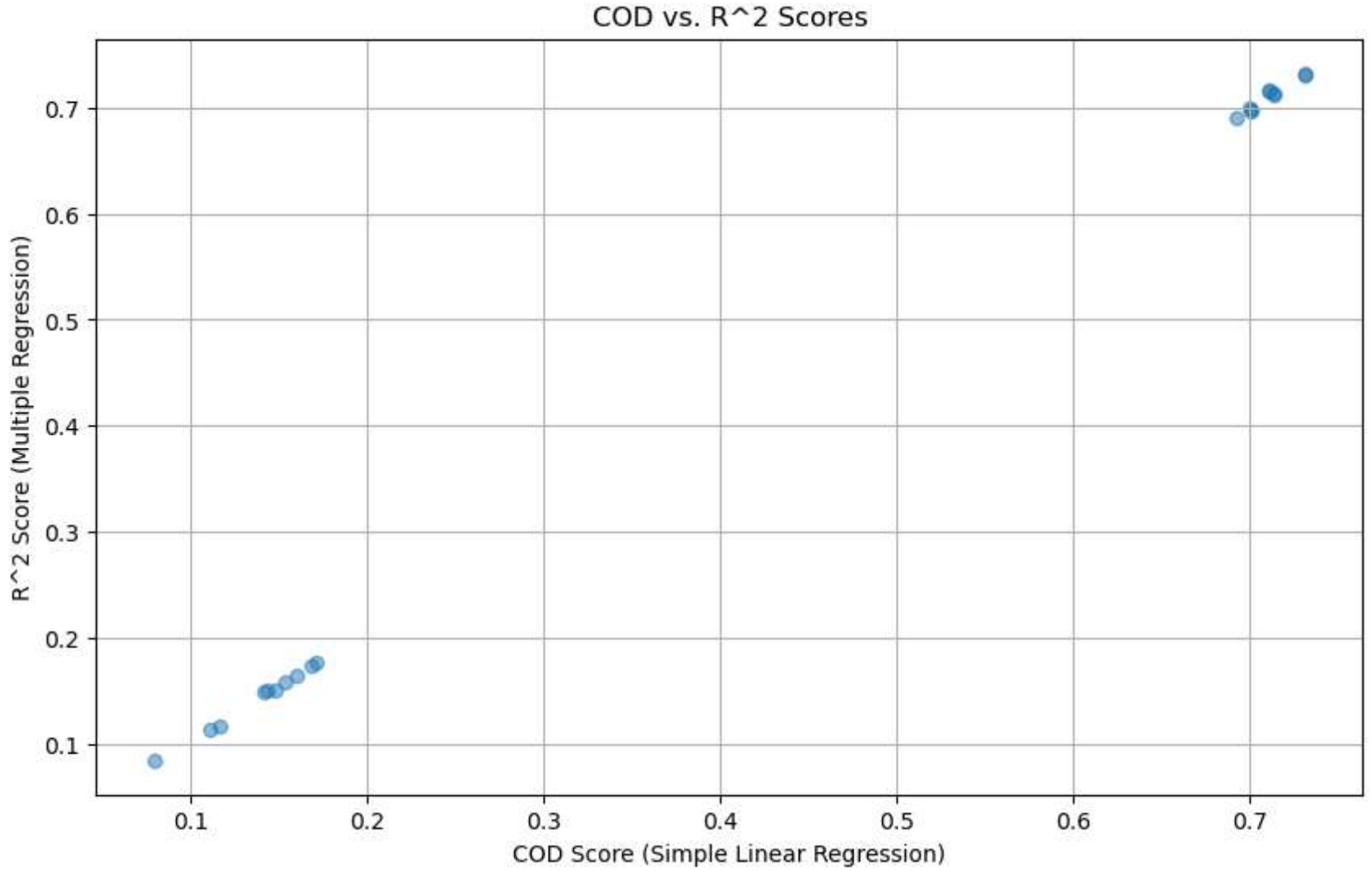
```
multi_regress_results_df.to_csv(output_file, index=False)
```

```
Movie 'Erik the Viking (1989)''s multiple regression model R^2 value: 0.7326943232004455
Movie 'I.Q. (1994)''s multiple regression model R^2 value: 0.7314260181489751
Movie 'The Lookout (2007)''s multiple regression model R^2 value: 0.7131613780628867
Movie 'Patton (1970)''s multiple regression model R^2 value: 0.7122717365528024
Movie 'The Bandit (1996)''s multiple regression model R^2 value: 0.7162937068176809
Movie 'Best Laid Plans (1999)''s multiple regression model R^2 value: 0.7160500038070143
Movie 'Congo (1995)''s multiple regression model R^2 value: 0.6971875747410903
Movie 'The Straight Story (1999)''s multiple regression model R^2 value: 0.6982031084083548
Movie 'The Final Conflict (1981)''s multiple regression model R^2 value: 0.6995767928586167
Movie 'Ran (1985)''s multiple regression model R^2 value: 0.6908654275584807
Movie 'Avatar (2009)''s multiple regression model R^2 value: 0.08435772761869276
Movie 'Interstellar (2014)''s multiple regression model R^2 value: 0.11281910087325442
Movie 'Black Swan (2010)''s multiple regression model R^2 value: 0.11667995262952413
Movie 'Clueless (1995)''s multiple regression model R^2 value: 0.1492205093550658
Movie 'The Cabin in the Woods (2012)''s multiple regression model R^2 value: 0.1503588747197835
Movie 'La La Land (2016)''s multiple regression model R^2 value: 0.15080367118463534
Movie 'Titanic (1997)''s multiple regression model R^2 value: 0.15794061919112778
Movie '13 Going on 30 (2004)''s multiple regression model R^2 value: 0.16442728893811032
Movie 'The Fast and the Furious (2001)''s multiple regression model R^2 value: 0.17296204925286351
Movie 'Grown Ups 2 (2013)''s multiple regression model R^2 value: 0.1758115612564587
```

In [16]:
```
path_to_first_results = r'C:\Users\YHD\Documents\PycharmProjects\DS Prj\analysis_results.csv'
path_to_second_results = r'C:\Users\YHD\Documents\PycharmProjects\DS Prj\multi_regress_results.csv'

# load result from 1
first_results_df = pd.read_csv(path_to_first_results)

# load result from 2
second_results_df = pd.read_csv(path_to_second_results)

# Combine data
combined_results_df = pd.merge(first_results_df, second_results_df, on='Target_Movie', how='inner')

# compute difference
combined_results_df['R2_COD_Difference'] = combined_results_df['R2_Score'] - combined_results_df['COD']

# Display
combined_results_df.head()
# save result
combined_results_path = r'C:\Users\YHD\Documents\PycharmProjects\DS Prj\combined_analysis_results.csv'
combined_results_df.to_csv(combined_results_path, index=False)

combined_results_path
```

Out[16]:
```
'C:\\Users\\YHD\\Documents\\PycharmProjects\\DS Prj\\combined_analysis_results.csv'
```

In [17]:
```
# scatter plot
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(combined_results_df['COD'], combined_results_df['R2_Score'], alpha=0.5)
plt.title('COD vs. R^2 Scores')
plt.xlabel('COD Score (Simple Linear Regression)')
plt.ylabel('R^2 Score (Multiple Regression)')
plt.grid(True)
plt.show()
```


COD vs. R^2 Scores
R^2 Score (Multiple Regression)
COD Score (Simple Linear Regression)
0.1
0.2
0.3
0.4
0.5
0.6
0.7

In [17]: