```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from scipy.sparse import csr_matrix

# Sample data: user_id, movie_id, rating
data = {
    'user_id': [1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 4],
    'movie_id': [101, 102, 103, 101, 104, 102, 103, 106, 101, 103, 104],
    'rating': [5, 4, 3, 5, 2, 3, 5, 4, 4, 2, 3]
}


df = pd.DataFrame(data)


user_item_matrix = df.pivot(index='user_id', columns='movie_id', values='rating').fillna(0)


sparse_matrix = csr_matrix(user_item_matrix)


user_similarity = cosine_similarity(sparse_matrix)


user_similarity_df = pd.DataFrame(user_similarity, index=user_item_matrix.index, columns=user_item_matrix.index)


def get_recommendations(user_id, num_recommendations=5):

    similarity_scores = user_similarity_df[user_id]
    user_ratings = user_item_matrix.loc[user_id]


    unrated_movies = user_ratings[user_ratings == 0]


    predicted_ratings = {}
    for movie in unrated_movies.index:

        weighted_sum = 0
        similarity_sum = 0
        for other_user in user_similarity_df.index:
            if other_user != user_id:
                rating = user_item_matrix.loc[other_user, movie]
                if rating > 0:
                    weighted_sum += similarity_scores[other_user] * rating
                    similarity_sum += similarity_scores[other_user]
        if similarity_sum > 0:
            predicted_ratings[movie] = weighted_sum / similarity_sum


    recommended_movies = sorted(predicted_ratings.items(), key=lambda x: x[1],
reverse=True)[:num_recommendations]

    return recommended_movies


recommendations = get_recommendations(1, num_recommendations=3)
print("Recommended movies for user 1:", recommendations)
```