In [5]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import time
from sklearn.model_selection import train_test_split
from scipy.sparse import coo_matrix, csr_matrix
from scipy.spatial.distance import jaccard, cosine
```

In [6]:
```
MV_users = pd.read_csv('data/users.csv')
MV_movies = pd.read_csv('data/movies.csv')
train = pd.read_csv('data/train.csv')
test = pd.read_csv('data/test.csv')
```

In [7]:
```
from collections import namedtuple
Data = namedtuple('Data', ['users','movies','train','test'])
data = Data(MV_users, MV_movies, train, test)
print(data)
```

```
Data(users=       uID gender  age  accupation    zip
0        1      F    1          10  48067
1        2      M   56          16  70072
2        3      M   25          15  55117
3        4      M   45           7  02460
4        5      M   25          20  55455
...    ...    ...  ...         ...    ...
6035  6036      F   25          15  32603
6036  6037      F   45           1  76006
6037  6038      F   56           1  14706
6038  6039      F   45           0  01060
6039  6040      M   25           6  11106

[6040 rows x 5 columns], movies=       mID                        title  yea
r  Doc  Com  Hor  Adv  Wes  Dra  \
0        1                    Toy Story  1995    0    1    0    0    0    0
1        2                      Jumanji  1995    0    0    0    1    0    0
2        3             Grumpier Old Men  1995    0    1    0    0    0    0
3        4            Waiting to Exhale  1995    0    1    0    0    0    1
4        5  Father of the Bride Part II  1995    0    1    0    0    0    0
...    ...                          ...   ...  ...  ...  ...  ...  ...  ...
3878  3948             Meet the Parents  2000    0    1    0    0    0    0
3879  3949          Requiem for a Dream  2000    0    0    0    0    0    1
3880  3950                    Tigerland  2000    0    0    0    0    0    1
3881  3951             Two Family House  2000    0    0    0    0    0    1
3882  3952               Contender, The  2000    0    0    0    0    0    1

      Ani  ...  Chi  Cri  Thr  Sci  Mys  Rom  Fil  Fan  Act  Mus
0       1  ...    1    0    0    0    0    0    0    0    0    0
1       0  ...    1    0    0    0    0    0    0    1    0    0
2       0  ...    0    0    0    0    0    1    0    0    0    0
3       0  ...    0    0    0    0    0    0    0    0    0    0
4       0  ...    0    0    0    0    0    0    0    0    0    0
...   ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
3878    0  ...    0    0    0    0    0    0    0    0    0    0
3879    0  ...    0    0    0    0    0    0    0    0    0    0
3880    0  ...    0    0    0    0    0    0    0    0    0    0
3881    0  ...    0    0    0    0    0    0    0    0    0    0
3882    0  ...    0    0    1    0    0    0    0    0    0    0

[3883 rows x 21 columns], train=         uID   mID  rating
0         744  1210       5
1        3040  1584       4
2        1451  1293       5
3        5455  3176       2
4        2507  3074       5
...       ...   ...     ...
700141  1184  2916       3
700142   137  1372       5
700143   195  2514       3
700144  1676  2566       3
700145  4611  1888       1

[700146 rows x 3 columns], test=         uID   mID  rating
0        2233   440       4
1        4274   587       5
```

```
2       2498   454       3
3       2868  2336       5
4       1636  2686       5
...      ...   ...     ...
300058   810   247       4
300059  1193  3210       4
300060  6039  2289       4
300061  5397   429       3
300062  1912   117       4

[300063 rows x 3 columns])
```

Train an NMF for user ratings

We will train an NMF on the genre of a movie and the user id to see if we can predict the rating on the test set. From the data above, it is possible that gender, age, occupation, zip, genre, and year of movie may be helpful in predicting rating. Only doing 40k rows to not explode my computer...

In [48]:
```python
from sklearn.decomposition import NMF
from sklearn.preprocessing import LabelEncoder

train_dataset = pd.DataFrame(columns={'uID':[],'mID':[], 'gender':[],'age':[

i = 0
n = 0
for idx in data.train.index[:30000]:
    uID = data.train['uID'][idx]
    mID = data.train['mID'][idx]

    user = data.users[data.users['uID'] == uID]
    gender = user['gender'].values[0]
    if gender == 'M':
        gender = 0
    else:
        gender = 1
    age = user['age'].values[0]
    accupation = user['accupation'].values[0]
    zip = user['zip'].values[0]

    movie = data.movies[data.movies['mID'] == mID]
    year = movie['year'].values[0]
    genre = movie.values[0][3:]
    genre_str = ''
    for item in genre:
        genre_str = genre_str + str(item)

    train_dataset.loc[len(train_dataset)] = [
        uID, mID, gender, age, accupation, zip, genre_str, year
    ]

    if i < 10000:
        i+=1
```

```
    else:
        print(f'{n/(len(data.train.index[:30000])+len(data.test.index[:10000
        i=0
    n+=1

for idx in data.test.index[:10000]:
    uID = data.test['uID'][idx]
    mID = data.test['mID'][idx]

    user = data.users[data.users['uID'] == uID]
    gender = user['gender'].values[0]
    if gender == 'M':
        gender = 0
    else:
        gender = 1
    age = user['age'].values[0]
    accupation = user['accupation'].values[0]
    zip = user['zip'].values[0]

    movie = data.movies[data.movies['mID'] == mID]
    year = movie['year'].values[0]
    genre = movie.values[0][3:]
    genre_str = ''
    for item in genre:
        genre_str = genre_str + str(item)

    train_dataset.loc[len(train_dataset)] = [
        uID, mID, gender, age, accupation, zip, genre_str, year
    ]

    if i < 10000:
        i+=1
    else:
        print(f'{n/(len(data.train[:30000].index)+len(data.test.index[:10000
        i=0
    n+=1

encoder = LabelEncoder()
train_dataset['genre'] = encoder.fit_transform(train_dataset['genre'])

print(train_dataset)
```

```
25.00 percent complete
50.00 percent complete
75.00 percent complete
        uID   mID  gender  age  accupation    zip  genre  year
0       744  1210       0   25          17  77007    161  1983
1      3040  1584       0   25           8  22046     83  1997
2      1451  1293       0   35          20  90012     72  1982
3      5455  3176       1   18          17  55449     91  1999
4      2507  3074       0   25           4  94107    117  1972
...     ...   ...     ...  ...         ...    ...    ...   ...
39995   855  3392       1   18           2  72701    207  1989
39996  3224  1508       1   25          14  93428     72  1997
39997  5267  3409       0   35           7  20191     85  2000
39998   883  2160       1   35          14  92673    193  1968
39999  4471   531       0   25           6  94108    107  1993

[40000 rows x 8 columns]
```

Now that all user and movie data has been organized, an nmf can be used to categorize them from 1-5 and see if the categories match the ratings.

In [49]:
```
from sklearn.decomposition import NMF
import re

train_dataset['zip'] = [re.sub(r'-', '', x) for x in train_dataset.zip.tolis

nmf = NMF(n_components = 5)
rating_df = nmf.fit_transform(train_dataset)
rating_df = pd.DataFrame(rating_df)

print(rating_df)
```

```
              0    1         2    3         4
0      0.007568  0.0  0.000006  0.0  0.016804
1      0.000000  0.0  0.000000  0.0  0.004875
2      0.012315  0.0  0.000012  0.0  0.019533
3      0.000000  0.0  0.000000  0.0  0.012261
4      0.000629  0.0  0.000021  0.0  0.020733
...         ...  ...       ...  ...       ...
39995  0.000000  0.0  0.000000  0.0  0.016076
39996  0.012109  0.0  0.000028  0.0  0.020253
39997  0.000000  0.0  0.000000  0.0  0.004465
39998  0.006926  0.0  0.000008  0.0  0.020282
39999  0.019995  0.0  0.000038  0.0  0.020158

[40000 rows x 5 columns]
```

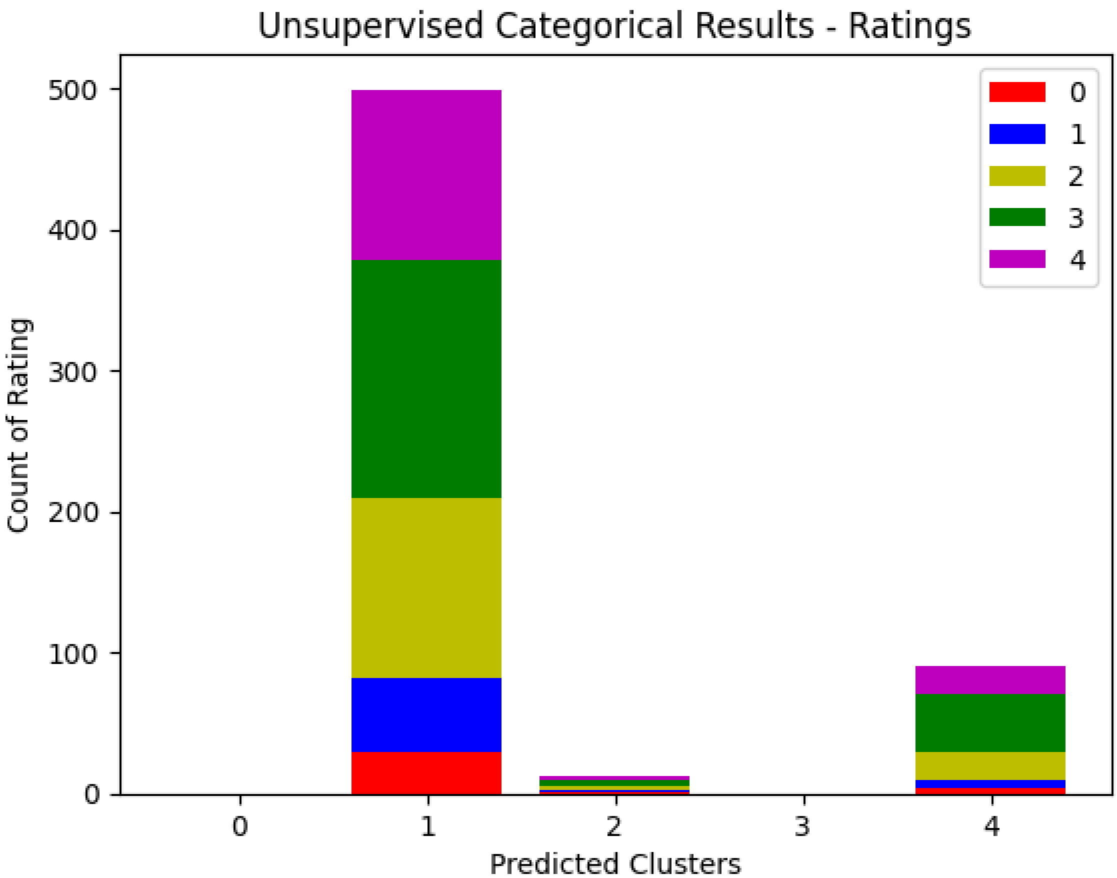
In [50]:
```
test_pred = pd.DataFrame(columns={'uID':[],'mID':[],'Pred':[], 'rating':[]})

for idx in test.index[:10000]:
    uID = test['uID'][idx]
    mID = test['mID'][idx]
    rating = test['rating'][idx]
    pred_list = rating_df.loc[29999+idx].tolist()

    test_pred.loc[len(test_pred)] = [
```

```
        uID,
        mID,
        pred_list.index(max(pred_list))+1,
        rating
    ]

print(test_pred)
```

```
       uID   mID  Pred  rating
0     2233   440     5       4
1     4274   587     5       5
2     2498   454     5       3
3     2868  2336     5       5
4     1636  2686     5       5
...    ...   ...   ...     ...
9995   855  3392     5       2
9996  3224  1508     5       4
9997  5267  3409     5       4
9998   883  2160     5       4
9999  4471   531     5       3

[10000 rows x 4 columns]
```

In [52]:
```
x = [0, 1, 2, 3, 4]

y_cats = {}

for category in test_pred.rating.unique():
    cat_df = test_pred[test_pred['rating'] == category].copy()
    y_cats[category] = []
    for pred in x:
        y_cats[category].append(len(cat_df[cat_df['Pred'] == pred].index))

    y_cats[category] = np.array(y_cats[category])

y_true = []
for idx in test_pred.index:
    category = test_pred['rating'][idx]
    y_true.append(
        list(y_cats[category]).index(max(y_cats[category]))
    )

plt.bar(x, y_cats[1], color = 'r')
plt.bar(x, y_cats[2], bottom = y_cats[1], color='b')
plt.bar(x, y_cats[3], bottom = y_cats[1] + y_cats[2], color='y')
plt.bar(x, y_cats[4], bottom = y_cats[1] + y_cats[2]
        + y_cats[3], color='g')
plt.bar(x, y_cats[5], bottom = y_cats[1] + y_cats[2]
        + y_cats[3] + y_cats[4], color='m')
plt.xlabel("Predicted Clusters")
plt.ylabel("Count of Rating")
plt.legend(["0", "1", "2", "3", "4"])
plt.title("Unsupervised Categorical Results - Ratings")
plt.show()

from sklearn.metrics import accuracy_score
```

```
print(f'Accuracy = {accuracy_score(y_true, test_pred["Pred"]):0.2f}')
print(f'RMSE = {np.sqrt(((y_true-test_pred["Pred"])**2).mean())}')
```




```
Accuracy = 0.05
RMSE = 3.8884315604109583
```