Problem Statement:

A film distribution company wants to target audience based on their likes and dislikes, you as a Chief Data Scientist Analyze the data and come up with different rules of movie list so that the business objective is achieved. 3.) my_movies.csv

Objective :-

Target audience based on their likes and dislikes

In [7]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#book = []
#with open("D:\\360Digi\\book.csv") as f:
#    book = f.read()
movies = pd.read_csv("D:\\360Digi\\my_movies.csv")
movies
```

Out[7]:

	V1	V2	V3	V4	V5	Sixth Sense	Gladiator	LOTR1	Harry Potter1	Patriot	LOTR2
0	Sixth Sense	LOTR1	Harry Potter1	Green Mile	LOTR2	1	0	1	1	0	1
1	Gladiator	Patriot	Braveheart	NaN	NaN	0	1	0	0	1	0
2	LOTR1	LOTR2	NaN	NaN	NaN	0	0	1	0	0	1
3	Gladiator	Patriot	Sixth Sense	NaN	NaN	1	1	0	0	1	0
4	Gladiator	Patriot	Sixth Sense	NaN	NaN	1	1	0	0	1	0
5	Gladiator	Patriot	Sixth Sense	NaN	NaN	1	1	0	0	1	0
6	Harry Potter1	Harry Potter2	NaN	NaN	NaN	0	0	0	1	0	0
7	Gladiator	Patriot	NaN	NaN	NaN	0	1	0	0	1	0
8	Gladiator	Patriot	Sixth Sense	NaN	NaN	1	1	0	0	1	0
9	Sixth Sense	LOTR	Gladiator	Green Mile	NaN	1	1	0	0	0	0

In [3]:

```
movies=movies.iloc[:,5:]
movies
```

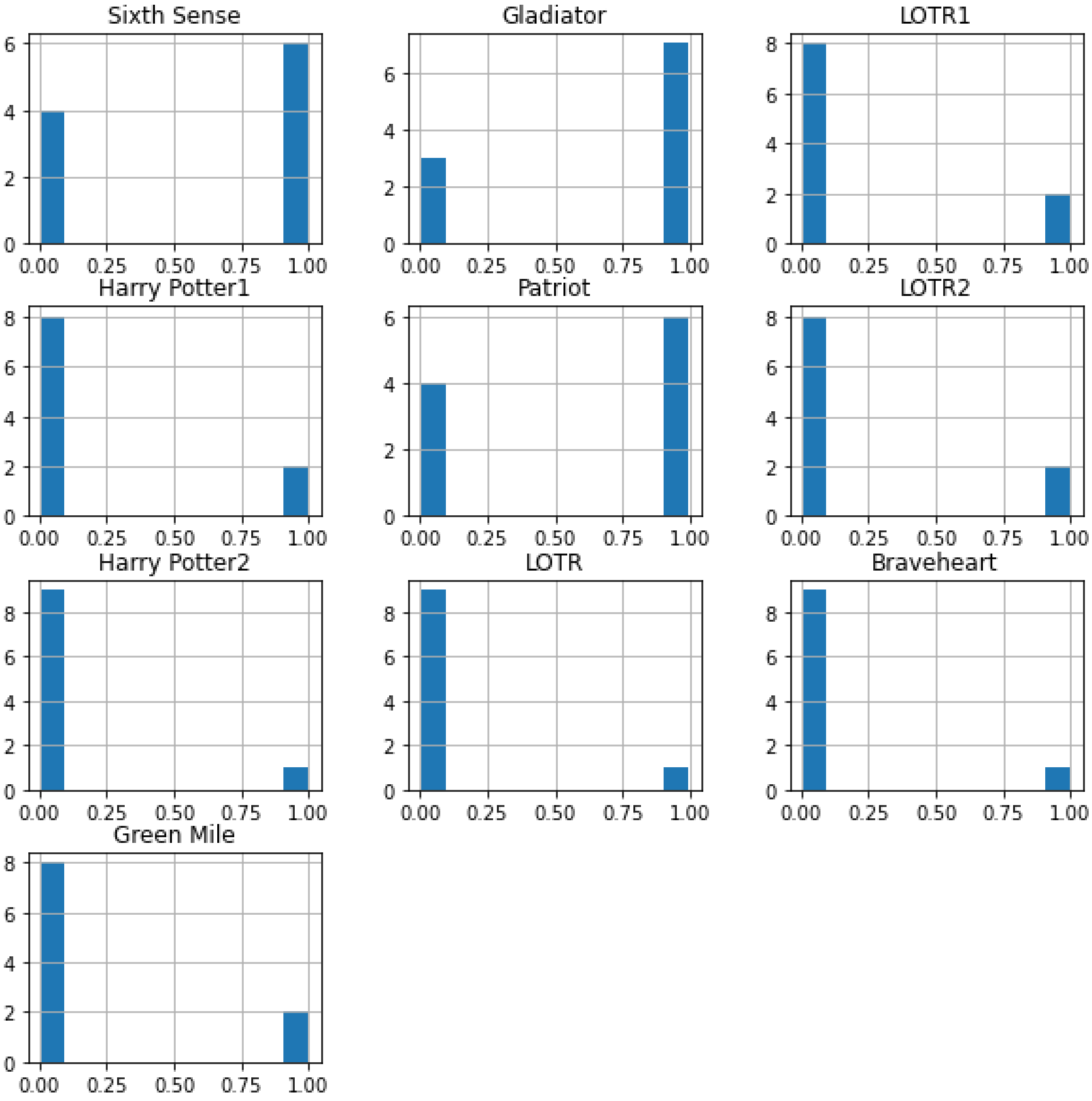
Out[3]:

	Sixth Sense	Gladiator	LOTR1	Harry Potter1	Patriot	LOTR2	Harry Potter2	LOTR	Braveheart	Green Mile
0	1	0	1	1	0	1	0	0	0	1
1	0	1	0	0	1	0	0	0	1	0
2	0	0	1	0	0	1	0	0	0	0
3	1	1	0	0	1	0	0	0	0	0
4	1	1	0	0	1	0	0	0	0	0
5	1	1	0	0	1	0	0	0	0	0
6	0	0	0	1	0	0	1	0	0	0
7	0	1	0	0	1	0	0	0	0	0
8	1	1	0	0	1	0	0	0	0	0
9	1	1	0	0	0	0	0	1	0	1

EDA

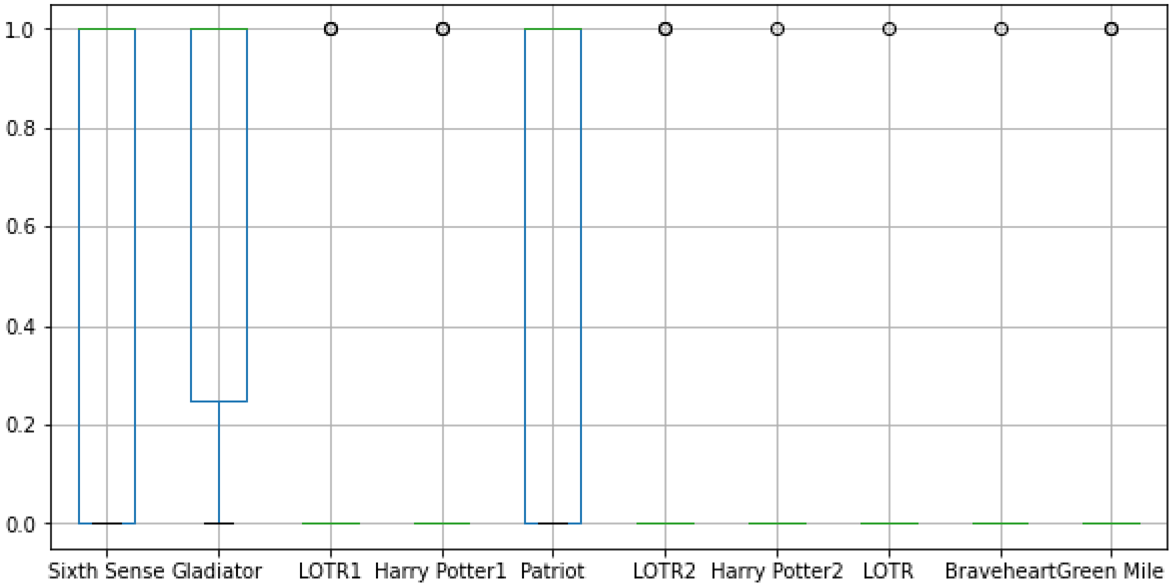
```
In [4]: movies.hist(grid=True, rwidth=0.9, figsize=(10,10))
```

```
Out[4]: array([[<AxesSubplot:title={'center':'Sixth Sense'}>,
                <AxesSubplot:title={'center':'Gladiator'}>,
                <AxesSubplot:title={'center':'LOTR1'}>],
               [<AxesSubplot:title={'center':'Harry Potter1'}>,
                <AxesSubplot:title={'center':'Patriot'}>,
                <AxesSubplot:title={'center':'LOTR2'}>],
               [<AxesSubplot:title={'center':'Harry Potter2'}>,
                <AxesSubplot:title={'center':'LOTR'}>,
                <AxesSubplot:title={'center':'Braveheart'}>],
               [<AxesSubplot:title={'center':'Green Mile'}>, <AxesSubplot:>,
                <AxesSubplot:>]], dtype=object)
```

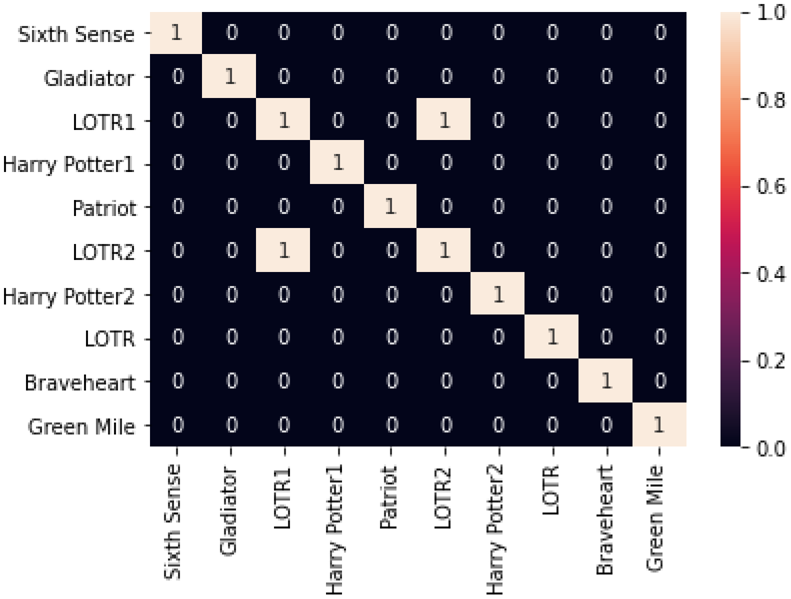


In [5]:
```
movies.boxplot(grid=True,figsize=(10,5))
```

Out[5]:
```
<AxesSubplot:>
```



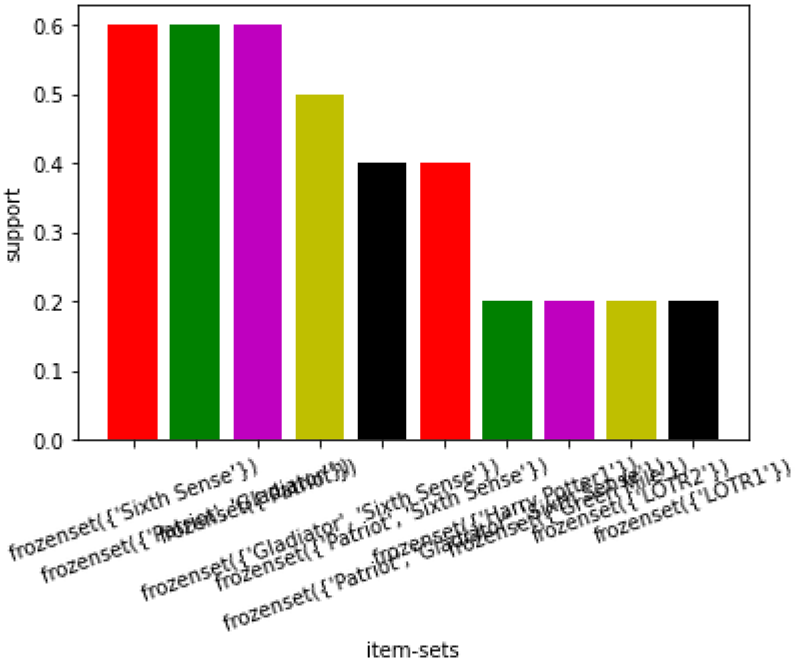
```
In [8]: a = movies.corr(method ='pearson')
        sns.heatmap(a>0.85,annot=True)

Out[8]: <AxesSubplot:>
```



	Sixth Sense	Gladiator	LOTR1	Harry Potter1	Patriot	LOTR2	Harry Potter2	LOTR	Braveheart	Green Mile
Sixth Sense	1	0	0	0	0	0	0	0	0	0
Gladiator	0	1	0	0	0	0	0	0	0	0
LOTR1	0	0	1	0	0	1	0	0	0	0
Harry Potter1	0	0	0	1	0	0	0	0	0	0
Patriot	0	0	0	0	1	0	0	0	0	0
LOTR2	0	0	1	0	0	1	0	0	0	0
Harry Potter2	0	0	0	0	0	0	1	0	0	0
LOTR	0	0	0	0	0	0	0	1	0	0
Braveheart	0	0	0	0	0	0	0	0	1	0
Green Mile	0	0	0	0	0	0	0	0	0	1

In [15]:
```
from mlxtend.frequent_patterns import apriori, association_rules

frequent_itemsets = apriori(movies, min_support = 0.05, max_len = 3, use_colnames

# Most Frequent item sets based on support
frequent_itemsets.sort_values('support', ascending = False, inplace = True)

plt.bar(x = list(range(1, 11)), height = frequent_itemsets.support[1:11], color =
plt.xticks(list(range(1, 11)), frequent_itemsets.itemsets[1:11], rotation=20)
plt.xlabel('item-sets')
plt.ylabel('support')
plt.show()

rules = association_rules(frequent_itemsets, metric = "lift", min_threshold = 1)
rules
```

```
<ipython-input-15-2f99becc6051>:8: MatplotlibDeprecationWarning: Using a string
of single character colors as a color sequence is deprecated since 3.2 and will
be removed two minor releases later. Use an explicit list instead.
  plt.bar(x = list(range(1, 11)), height = frequent_itemsets.support[1:11], col
or ='rgmyk')
```



```
Out[15]:
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(Patriot)	(Gladiator)	0.6	0.7	0.6	1.000000	1.428571	0.18
1	(Gladiator)	(Patriot)	0.7	0.6	0.6	0.857143	1.428571	0.18
2	(Gladiator)	(Sixth Sense)	0.7	0.6	0.5	0.714286	1.190476	0.08
3	(Sixth Sense)	(Gladiator)	0.6	0.7	0.5	0.833333	1.190476	0.08
4	(Patriot)	(Sixth Sense)	0.6	0.6	0.4	0.666667	1.111111	0.04
...	...	...	...	...	...	...	...	...
119	(Green Mile, Harry Potter1)	(LOTR2)	0.1	0.2	0.1	1.000000	5.000000	0.08
120	(LOTR2, Harry Potter1)	(Green Mile)	0.1	0.2	0.1	1.000000	5.000000	0.08
121	(Green Mile)	(LOTR2, Harry Potter1)	0.2	0.1	0.1	0.500000	5.000000	0.08
122	(LOTR2)	(Green Mile, Harry Potter1)	0.2	0.1	0.1	0.500000	5.000000	0.08
123	(Harry Potter1)	(Green Mile, LOTR2)	0.2	0.1	0.1	0.500000	5.000000	0.08

124 rows × 9 columns

In [16]:

```
rules.head(20)
rules.sort_values('lift', ascending = False).head(10)
```

Out[16]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	convi
81	(Green Mile, Gladiator)	(LOTR)	0.1	0.1	0.1	1.0	10.0	0.09	
84	(LOTR)	(Green Mile, Gladiator)	0.1	0.1	0.1	1.0	10.0	0.09	
62	(Harry Potter1, LOTR1)	(LOTR2)	0.1	0.2	0.1	1.0	5.0	0.08	
72	(Green Mile)	(LOTR)	0.2	0.1	0.1	0.5	5.0	0.08	
69	(Green Mile)	(LOTR2, Sixth Sense)	0.2	0.1	0.1	0.5	5.0	0.08	
68	(LOTR2, Sixth Sense)	(Green Mile)	0.1	0.2	0.1	1.0	5.0	0.08	
65	(LOTR1)	(LOTR2, Harry Potter1)	0.2	0.1	0.1	0.5	5.0	0.08	
63	(LOTR2)	(Harry Potter1, LOTR1)	0.2	0.1	0.1	0.5	5.0	0.08	
60	(LOTR2, Harry Potter1)	(LOTR1)	0.1	0.2	0.1	1.0	5.0	0.08	
57	(Green Mile)	(Harry Potter1, Sixth Sense)	0.2	0.1	0.1	0.5	5.0	0.08	

In [17]:

```
######Redudancy is defined as the storing of same data multiple time##

def to_list(i):
    return (sorted(list(i)))

ma_X = rules.antecedents.apply(to_list) + rules.consequents.apply(to_list)

ma_X = ma_X.apply(sorted)

rules_sets = list(ma_X)

unique_rules_sets = [list(m) for m in set(tuple(i) for i in rules_sets)]

index_rules = []

for i in unique_rules_sets:
    index_rules.append(rules_sets.index(i))

# getting rules without any redudancy
rules_no_redudancy = rules.iloc[index_rules, :]

# Sorting them with respect to list and getting top 10 rules
rules_no_redudancy.sort_values('lift', ascending = False).head(10)
```

Out[17]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	convi
38	(LOTR2, Sixth Sense)	(LOTR1)	0.1	0.2	0.1	1.0	5.0	0.08	
114	(Harry Potter2)	(Harry Potter1)	0.1	0.2	0.1	1.0	5.0	0.08	
118	(Green Mile, LOTR2)	(Harry Potter1)	0.1	0.2	0.1	1.0	5.0	0.08	
86	(Green Mile, LOTR2)	(LOTR1)	0.1	0.2	0.1	1.0	5.0	0.08	
12	(LOTR2)	(LOTR1)	0.2	0.2	0.2	1.0	5.0	0.16	
72	(Green Mile)	(LOTR)	0.2	0.1	0.1	0.5	5.0	0.08	
48	(Green Mile, Harry Potter1)	(LOTR1)	0.1	0.2	0.1	1.0	5.0	0.08	
60	(LOTR2, Harry Potter1)	(LOTR1)	0.1	0.2	0.1	1.0	5.0	0.08	
98	(Green Mile)	(LOTR1)	0.2	0.2	0.1	0.5	2.5	0.06	
110	(Harry Potter1)	(LOTR1)	0.2	0.2	0.1	0.5	2.5	0.06	

In [ ]:

Summary:

1- Above the 10 unique Rule that we get by Apply Apriori Algo.

2- Antecedent support variable tells us probability of antecedent product alone.

3- The Support Value is the value of the two Product(Antecedents and Consequents)

4- Confidence is an indication of how often the rule has been found to be True.

5-The ratio of the observed support to that expected if X and Y were independent.

In [ ]: