In [18]:

```
import pandas as pd
import numpy as np
```

In [19]:

```
data=pd.read_csv('movie.csv')
```

DATA VISUALIZATION ¶

In [20]:

```
data.shape
```

Out[20]:

```
(1000, 12)
```

In [21]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Rank                1000 non-null   int64  
 1   Title               1000 non-null   object 
 2   Genre               1000 non-null   object 
 3   Description         1000 non-null   object 
 4   Director            1000 non-null   object 
 5   Actors              1000 non-null   object 
 6   Year                1000 non-null   int64  
 7   Runtime (Minutes)   1000 non-null   int64  
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64  
 10  Revenue (Millions)  872 non-null    float64
 11  Metascore           936 non-null    float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

TOP 10 Movies

In [22]:

```
data.head(10)
```

Out[22]:

	Rank	Title	Genre	Description	Director	Actors	Ye
0	1	Guardians of the Galaxy	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	20
1	2	Prometheus	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	20
2	3	Split	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	20
3	4	Sing	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	20
4	5	Suicide Squad	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	20
5	6	The Great Wall	Action,Adventure,Fantasy	European mercenaries searching for black powde...	Yimou Zhang	Matt Damon, Tian Jing, Willem Dafoe, Andy Lau	20
6	7	La La Land	Comedy,Drama,Music	A jazz pianist falls for an aspiring actress i...	Damien Chazelle	Ryan Gosling, Emma Stone, Rosemarie DeWitt, J....	20
7	8	Mindhorn	Comedy	A has-been actor best known for playing the ti...	Sean Foley	Essie Davis, Andrea Riseborough, Julian Barrat...	20
8	9	The Lost City of Z	Action,Adventure,Biography	A true-life drama, centering on British explor...	James Gray	Charlie Hunnam, Robert Pattinson, Sienna Mille...	20
9	10	Passengers	Adventure,Drama,Romance	A spacecraft traveling to a distant colony pla...	Morten Tyldum	Jennifer Lawrence, Chris Pratt, Michael Sheen,...	20

Bottom 10 Movies

In [23]:

```
data.tail()
```

Out[23]:

	Rank	Title	Genre	Description	Director	Actors	Year	Run (Minu
995	996	Secret in Their Eyes	Crime,Drama,Mystery	A tight-knit team of rising investigators, alo...	Billy Ray	Chiwetel Ejiofor, Nicole Kidman, Julia Roberts...	2015	
996	997	Hostel: Part II	Horror	Three American college students studying abroa...	Eli Roth	Lauren German, Heather Matarazzo, Bijou Philli...	2007	
997	998	Step Up 2: The Streets	Drama,Music,Romance	Romantic sparks occur between two dance studen...	Jon M. Chu	Robert Hoffman, Briana Evigan, Cassie Ventura,...	2008	
998	999	Search Party	Adventure,Comedy	A pair of friends embark on a mission to reuni...	Scot Armstrong	Adam Pally, T.J. Miller, Thomas Middleditch,Sh...	2014	
999	1000	Nine Lives	Comedy,Family,Fantasy	A stuffy businessman finds himself trapped ins...	Barry Sonnenfeld	Kevin Spacey, Jennifer Garner, Robbie Amell,Ch...	2016	

In [24]:

```
data.sample(10)
```

Out[24]:

	Rank	Title	Genre	Description	Director	Actors	Yea
389	390	John Carter	Action,Adventure,Sci-Fi	Transported to Barsoom, a Civil War vet discov...	Andrew Stanton	Taylor Kitsch, Lynn Collins, Willem Dafoe,Sama...	201
295	296	The Choice	Drama,Romance	Travis and Gabby first meet as neighbors in a ...	Ross Katz	Benjamin Walker, Teresa Palmer, Alexandra Dadd...	201
999	1000	Nine Lives	Comedy,Family,Fantasy	A stuffy businessman finds himself trapped ins...	Barry Sonnenfeld	Kevin Spacey, Jennifer Garner, Robbie Amell,Ch...	201
846	847	Home	Animation,Adventure,Comedy	An alien on the run from his own people makes ...	Tim Johnson	Jim Parsons, Rihanna, Steve Martin, Jennifer L...	201
65	66	Kingsman: The Secret Service	Action,Adventure,Comedy	A spy organization recruits an unrefined, but ...	Matthew Vaughn	Colin Firth, Taron Egerton, Samuel L. Jackson,...	201
242	243	Rock Dog	Animation,Adventure,Comedy	When a radio falls from the sky into the hands...	Ash Brannon	Luke Wilson, Eddie Izzard, J.K. Simmons, Lewis...	201
419	420	Shame	Drama	A man's carefully cultivated private life is d...	Steve McQueen	Michael Fassbender, Carey Mulligan, James Badg...	201
836	837	Bacalaureat	Crime,Drama	A film about compromises and the implications ...	Cristian Mungiu	Adrian Titieni, Maria-Victoria Dragus, Lia Bug...	201
250	251	Bonjour Anne	Comedy,Drama,Romance	Anne is at a crossroads in her life. Long marr...	Eleanor Coppola	Diane Lane, Alec Baldwin, Arnaud Viard, Linda ...	201
593	594	She's the Man	Comedy,Romance,Sport	When her brother decides to ditch for a couple...	Andy Fickman	Amanda Bynes, Laura Ramsey, Channing Tatum,Vin...	200

In [25]:

```
data.describe()
```

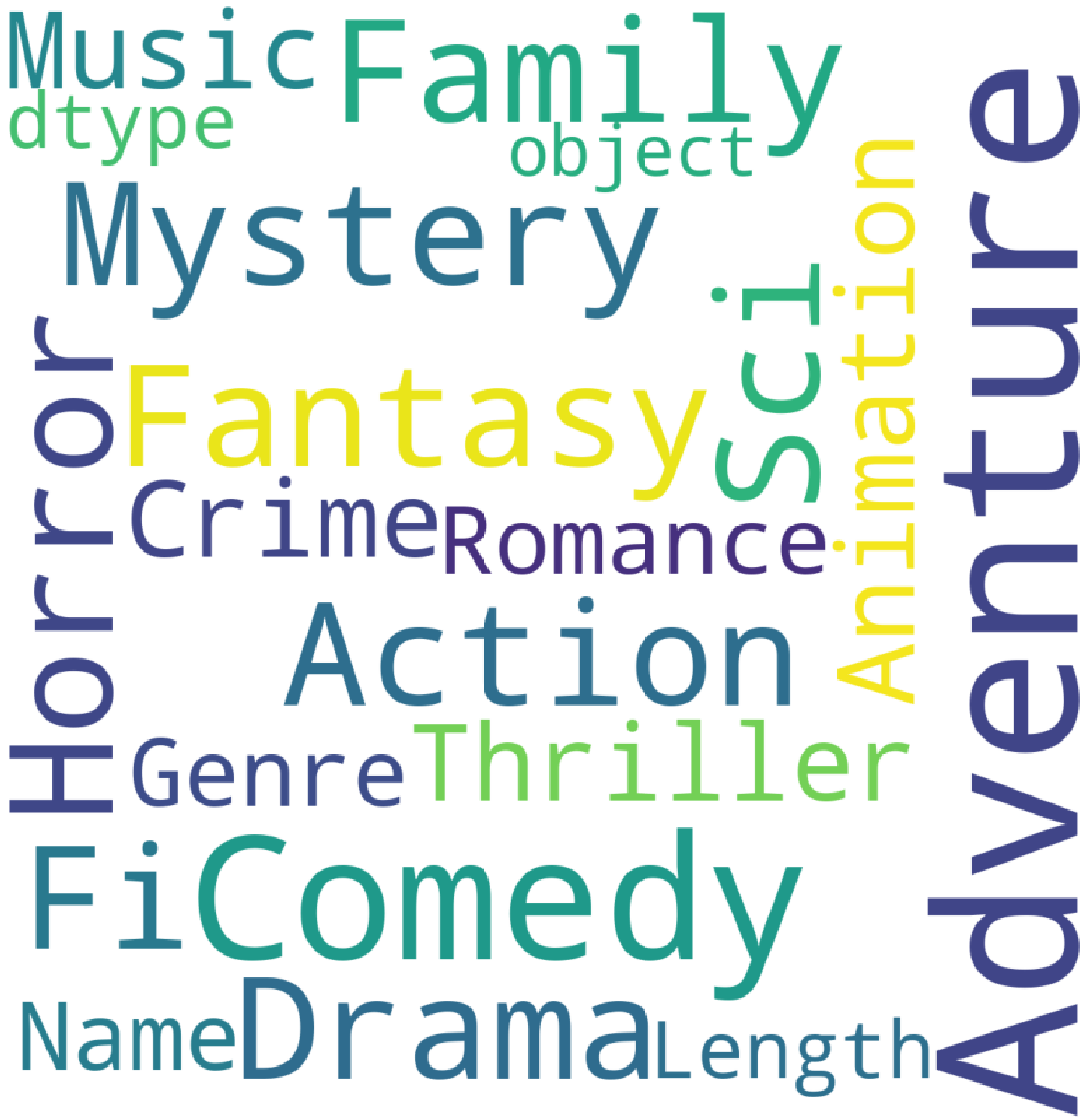
Out[25]:

	Rank	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metasco
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	872.000000	936.0000
mean	500.500000	2012.783000	113.172000	6.723200	1.698083e+05	82.956376	58.9850
std	288.819436	3.205962	18.810908	0.945429	1.887626e+05	103.253540	17.1947
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.0000
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	13.270000	47.0000
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	47.985000	59.5000
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	113.715000	72.0000
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.0000

In [26]:

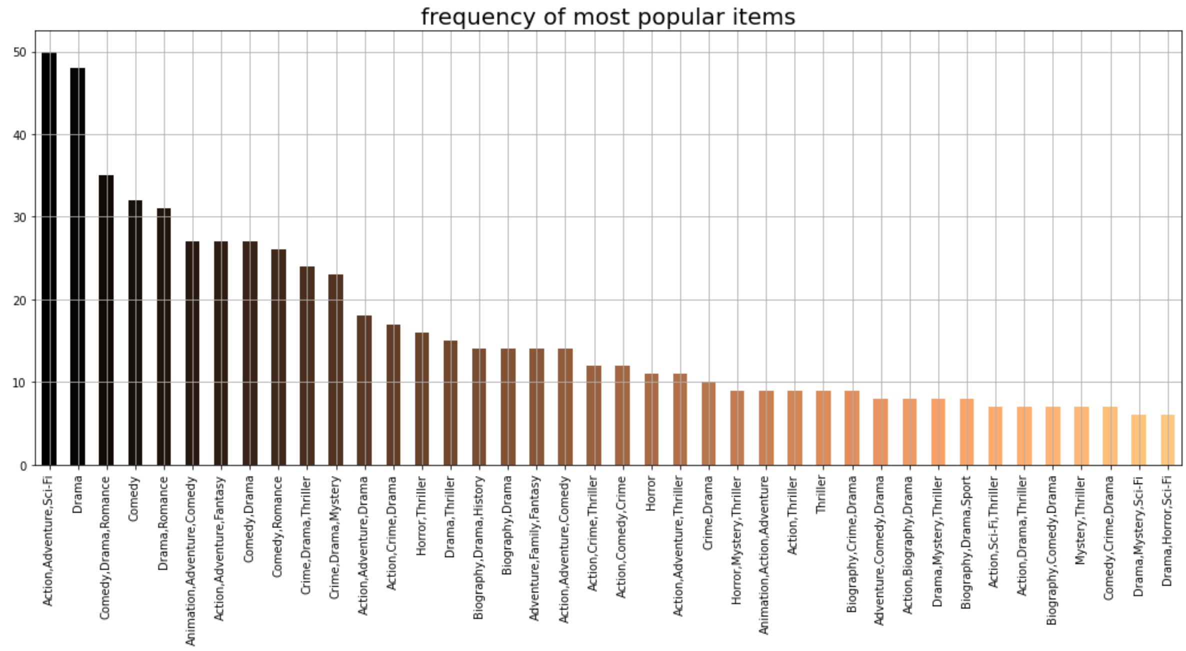
```
import matplotlib.pyplot as plt
import seaborn as sns

from wordcloud import WordCloud

plt.rcParams['figure.figsize'] = (15, 15)
wordcloud = WordCloud(background_color = 'white', width = 1200,  height = 1200, max_words =
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Most Popular Items',fontsize = 20)
plt.show()
```

Most Popular Items



In [27]:

```
plt.rcParams['figure.figsize'] = (18, 7)
color = plt.cm.copper(np.linspace(0, 1, 40))
data['Genre'].value_counts().head(40).plot.bar(color = color)
plt.title('frequency of most popular items', fontsize = 20)
plt.xticks(rotation = 90 )
plt.grid()
plt.show()
```



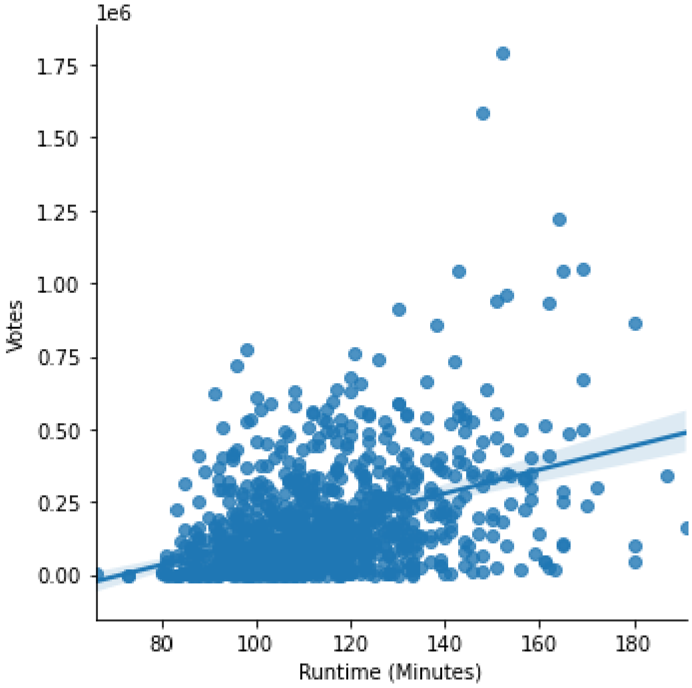
Runtime vs Votes

In [139]:

```
sns.lmplot(x='Runtime (Minutes)',y='Votes',data=data)
```

Out[139]:

```
<seaborn.axisgrid.FacetGrid at 0x2238fe37f40>
```



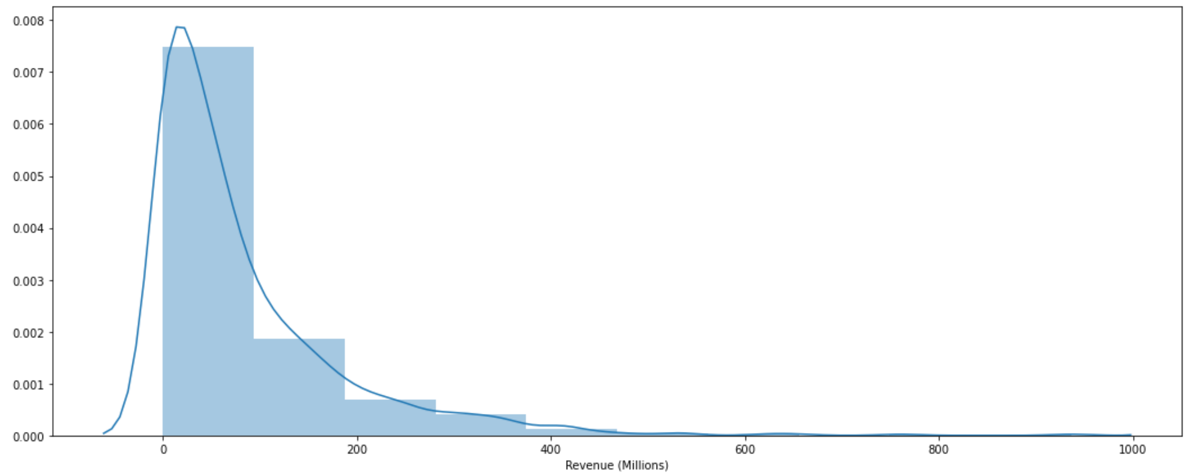
Variation in Revenue

In [145]:

```
sns.distplot(data['Revenue (Millions)'],bins=10)
```

Out[145]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2238e8e7280>
```



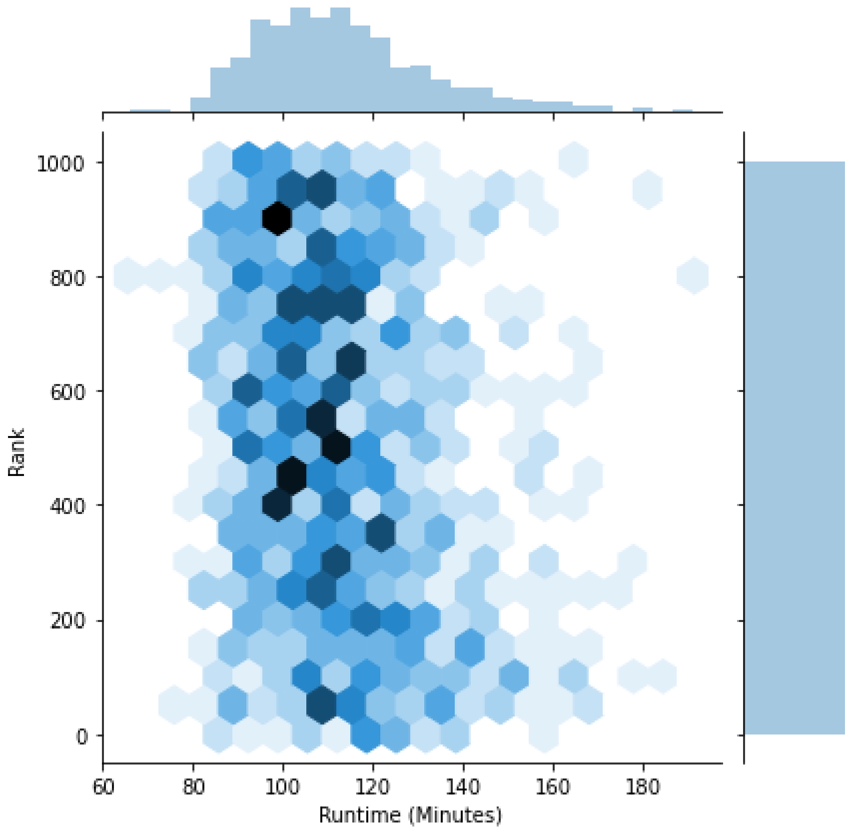
Runtime vs Ranks

In [147]:

```
sns.jointplot(x='Runtime (Minutes)',y='Rank',data=data,kind='hex')
```

Out[147]:

```
<seaborn.axisgrid.JointGrid at 0x2238f9f0d90>
```



APPLYING APRIORI ALGORITHM

In [29]:

```
###movies_gen = data.drop(['Rank','genres','title'],1).join(data.genres.str.get_dummies())
#pd.options.display.max_columns=100
#movies_genero.head()###
```

In [30]:

```
stat2 = data.drop(['Rank','Description','Director','Metascore','Actors','Year','Rating','Vo
```

In [31]:

```
stat2.head(10)
```

Out[31]:

	Genre	Genres Count
0	Action,Adventure,Sci-Fi	3
1	Adventure,Mystery,Sci-Fi	3
2	Horror,Thriller	2
3	Animation,Comedy,Family	3
4	Action,Adventure,Fantasy	3
5	Action,Adventure,Fantasy	3
6	Comedy,Drama,Music	3
7	Comedy	1
8	Action,Adventure,Biography	3
9	Adventure,Drama,Romance	3

In [32]:

```
dummy=data["Genre"].str.get_dummies(",")
```

In [33]:

```
dummy.head(20)
```

Out[33]:

	Action	Adventure	Animation	Biography	Comedy	Crime	Drama	Family	Fantasy	Histor
0	1	1	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	1	0	1	0	0	1	0	
4	1	1	0	0	0	0	0	0	1	
5	1	1	0	0	0	0	0	0	1	
6	0	0	0	0	1	0	1	0	0	
7	0	0	0	0	1	0	0	0	0	
8	1	1	0	1	0	0	0	0	0	
9	0	1	0	0	0	0	1	0	0	
10	0	1	0	0	0	0	0	1	1	
11	0	0	0	1	0	0	1	0	0	
12	1	1	0	0	0	0	0	0	0	
13	0	1	1	0	1	0	0	0	0	
14	1	0	0	0	1	0	1	0	0	
15	0	1	1	0	1	0	0	0	0	
16	0	0	0	1	0	0	1	0	0	
17	1	0	0	0	0	0	0	0	0	
18	0	0	0	1	0	0	1	0	0	
19	0	0	0	0	0	0	1	0	0	

Lift
how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is
The lift of a rule is defined as:

lift (X ---> Y)=sup (X U Y) / sup (X) x sup (Y)

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

In [189]:

```
from mlxtend.frequent_patterns import association_rules
lt=association_rules(frequent_itemsets, metric="lift" ,  min_threshold=0.4 )
lt= lt[['antecedents', 'consequents', 'lift']]
lt
#Here if the min_threshold parameter is removed the default min_threshold value is taken to
#As our min_threshold increases from 0.1 to 0,the count of values decreases
```

```
antecedents    True
consequents    True
lift           True
dtype: bool
```

In [186]:

```
lt.head(10)
```

...

In [187]:

```
lt.tail(10)
```

Out[187]:

	antecedents	consequents	lift
88	(Mystery, Drama)	(Crime)	2.948718
89	(Crime)	(Mystery, Drama)	2.948718
90	(Mystery)	(Crime, Drama)	2.236919
91	(Drama)	(Crime, Mystery)	1.546011
92	(Thriller, Drama)	(Crime)	2.000000
93	(Thriller, Crime)	(Drama)	1.063264
94	(Crime, Drama)	(Thriller)	1.268834
95	(Thriller)	(Crime, Drama)	1.268834
96	(Drama)	(Thriller, Crime)	1.063264
97	(Crime)	(Thriller, Drama)	2.000000

In [176]:

```
lt.max()
```

Out[176]:

```
antecedents                (Crime)
consequents      (Thriller, Drama)
lift                       9.33933
dtype: object
```

Confidence

Confidence is an indication of how often the rule has been found to be true.

The confidence value of a rule, X ---> Y , with respect to a set of transactions T, is the proportion of the transactions that contains X which also contains Y.

Confidence is defined as:

conf(X ---> Y)=supp(X U Y)/supp (X)

In [180]:

```
from mlxtend.frequent_patterns import association_rules
cn=association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
cn = cn[['antecedents', 'consequents', 'confidence']]
cn
#Here if the min_threshold parameter is removed the default min_threshold value is taken to
#As our min_threshold increases from 0.1 to 0,the count of values decreases
```

Out[180]:

	antecedents	consequents	confidence
0	(Animation)	(Adventure)	0.775510
1	(Biography)	(Drama)	0.913580
2	(History)	(Drama)	0.965517
3	(Adventure, Sci-Fi)	(Action)	0.847458
4	(Adventure, Animation)	(Comedy)	0.710526
5	(Comedy, Animation)	(Adventure)	0.794118
6	(Crime, Mystery)	(Drama)	0.793103

In [179]:

```
cn.head(10)
```

Out[179]:

	antecedents	consequents	confidence
0	(Animation)	(Adventure)	0.775510
1	(Biography)	(Drama)	0.913580
2	(History)	(Drama)	0.965517
3	(Adventure, Sci-Fi)	(Action)	0.847458
4	(Adventure, Animation)	(Comedy)	0.710526
5	(Comedy, Animation)	(Adventure)	0.794118
6	(Crime, Mystery)	(Drama)	0.793103

In [177]:

```
cn.max()
```

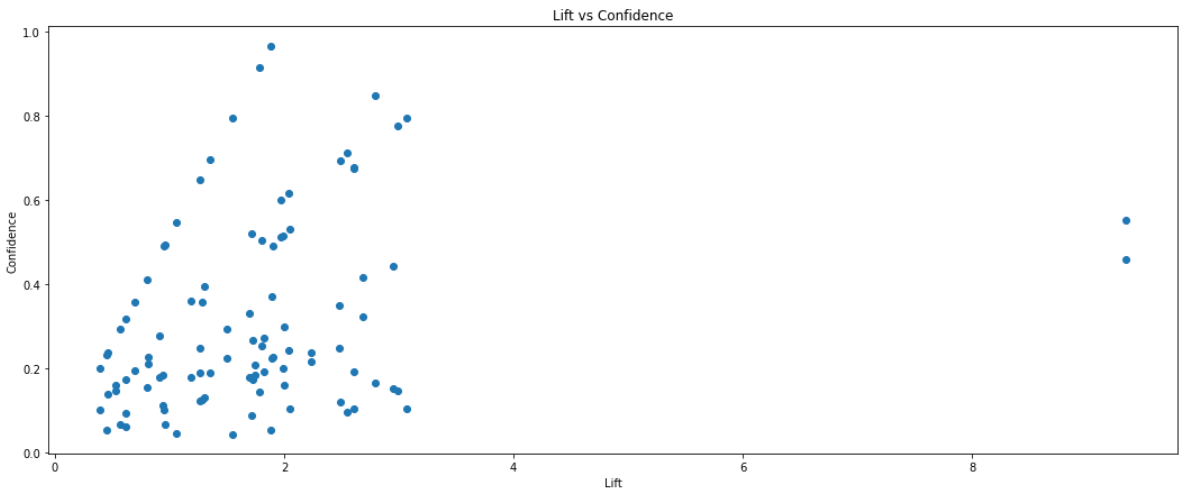
Out[177]:

```
antecedents    (Crime, Mystery)
consequents             (Drama)
confidence             0.965517
dtype: object
```

LIFT vs CONFIDENCE (with metric = lift)

In [128]:

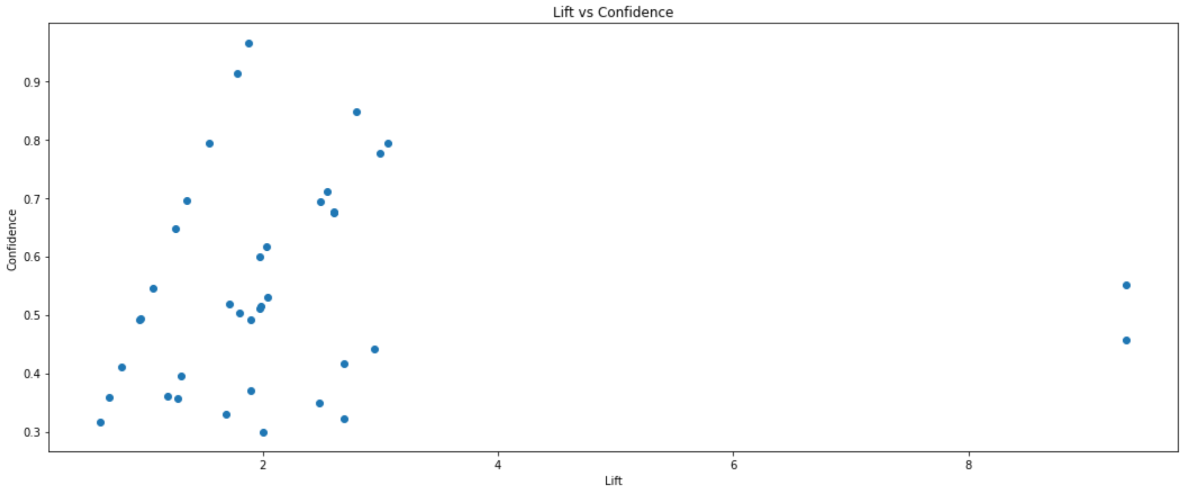
```
rules=association_rules(frequent_itemsets, metric="lift", min_threshold=0.3)
plt.scatter(rules['lift'],rules['confidence'])
plt.xlabel('Lift')
plt.ylabel('Confidence')
plt.title('Lift vs Confidence')
plt.show()
```



LIFT vs CONFIDENCE (with metric = confidence)

In [115]:

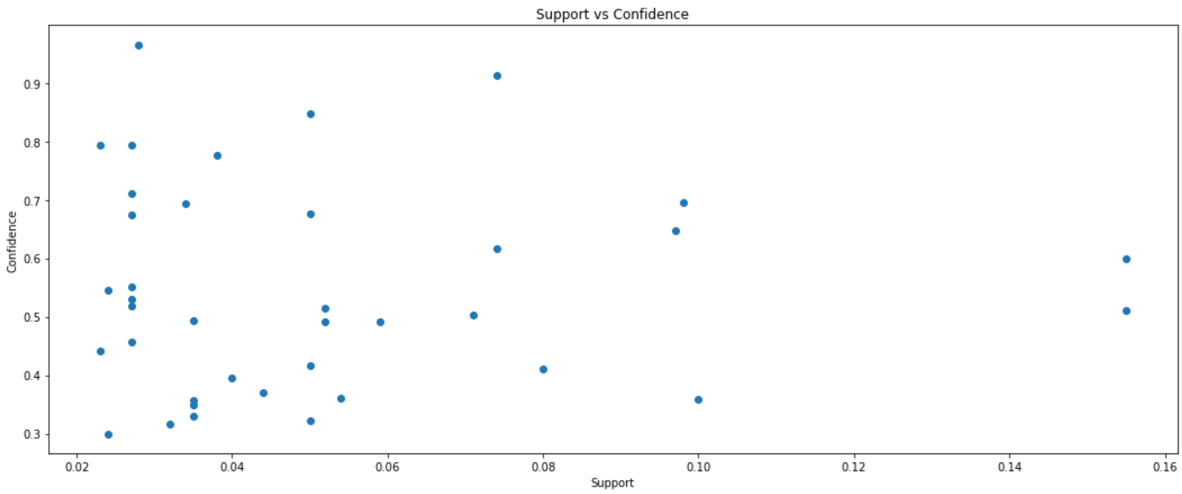
```
rules=association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3)
plt.scatter(rules['lift'],rules['confidence'])
plt.xlabel('Lift')
plt.ylabel('Confidence')
plt.title('Lift vs Confidence')
plt.show()
```



SUPPORT vs CONFIDENCE (with metric = confidence)

In [130]:

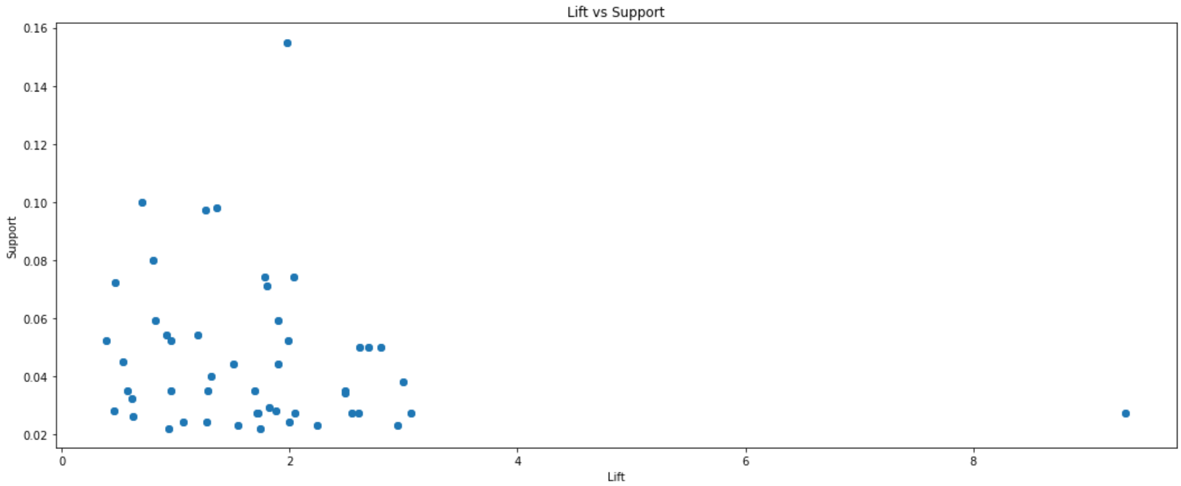
```
rules=association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3)
plt.scatter(rules['support'],rules['confidence'])
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Support vs Confidence')
plt.show()
```



LIFT vs SUPPORT (with metric = lift)

In [131]:

```
rules=association_rules(frequent_itemsets, metric="lift", min_threshold=0.3)
plt.scatter(rules['lift'],rules['support'])
plt.xlabel('Lift')
plt.ylabel('Support')
plt.title('Lift vs Support')
plt.show()
```



Support
Support is an indication of how frequently the itemset appears in the dataset.

The support of X with respect to T is defined as the proportion of transactions t in the dataset which contains the itemset X.

In [35]:

```
from mlxtend.frequent_patterns import apriori

frequent_itemsets = apriori(dummy, min_support = 0.02, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

Out[35]:

	support	itemsets	length
0	0.303	(Action)	1
1	0.259	(Adventure)	1
2	0.049	(Animation)	1
3	0.081	(Biography)	1
4	0.279	(Comedy)	1
5	0.150	(Crime)	1
6	0.513	(Drama)	1
7	0.051	(Family)	1
8	0.101	(Fantasy)	1
9	0.029	(History)	1
10	0.119	(Horror)	1
11	0.106	(Mystery)	1
12	0.141	(Romance)	1
13	0.120	(Sci-Fi)	1
14	0.195	(Thriller)	1
15	0.155	(Adventure, Action)	2
16	0.045	(Comedy, Action)	2
17	0.054	(Crime, Action)	2
18	0.072	(Action, Drama)	2
19	0.040	(Fantasy, Action)	2
20	0.074	(Sci-Fi, Action)	2
21	0.054	(Thriller, Action)	2
22	0.038	(Adventure, Animation)	2
23	0.059	(Adventure, Comedy)	2
24	0.052	(Adventure, Drama)	2
25	0.027	(Adventure, Family)	2
26	0.052	(Fantasy, Adventure)	2
27	0.059	(Adventure, Sci-Fi)	2
28	0.034	(Comedy, Animation)	2
29	0.074	(Biography, Drama)	2
30	0.026	(Comedy, Crime)	2
31	0.100	(Comedy, Drama)	2

	support	itemsets	length
32	0.071	(Comedy, Romance)	2
33	0.097	(Crime, Drama)	2
34	0.029	(Crime, Mystery)	2
35	0.044	(Thriller, Crime)	2
36	0.032	(Fantasy, Drama)	2
37	0.028	(History, Drama)	2
38	0.035	(Horror, Drama)	2
39	0.052	(Mystery, Drama)	2
40	0.098	(Drama, Romance)	2
41	0.028	(Sci-Fi, Drama)	2
42	0.080	(Thriller, Drama)	2
43	0.022	(Horror, Mystery)	2
44	0.044	(Horror, Thriller)	2
45	0.035	(Thriller, Mystery)	2
46	0.022	(Thriller, Sci-Fi)	2
47	0.027	(Adventure, Fantasy, Action)	3
48	0.050	(Adventure, Sci-Fi, Action)	3
49	0.027	(Adventure, Comedy, Animation)	3
50	0.035	(Comedy, Drama, Romance)	3
51	0.023	(Crime, Mystery, Drama)	3
52	0.024	(Thriller, Drama, Crime)	3

In [36]:

```
## getting the item sets with length = 2 and support more han 4%

frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.04) ]
```

Out[36]:

	support	itemsets	length
15	0.155	(Adventure, Action)	2
16	0.045	(Comedy, Action)	2
17	0.054	(Crime, Action)	2
18	0.072	(Action, Drama)	2
19	0.040	(Fantasy, Action)	2
20	0.074	(Sci-Fi, Action)	2
21	0.054	(Thriller, Action)	2
23	0.059	(Adventure, Comedy)	2
24	0.052	(Adventure, Drama)	2
26	0.052	(Fantasy, Adventure)	2
27	0.059	(Adventure, Sci-Fi)	2
29	0.074	(Biography, Drama)	2
31	0.100	(Comedy, Drama)	2
32	0.071	(Comedy, Romance)	2
33	0.097	(Crime, Drama)	2
35	0.044	(Thriller, Crime)	2
39	0.052	(Mystery, Drama)	2
40	0.098	(Drama, Romance)	2
42	0.080	(Thriller, Drama)	2
44	0.044	(Horror, Thriller)	2

In [ ]:

```
```

In [37]:

```
#getting the item sets with length = 1 and support more han 5%

frequent_itemsets[ (frequent_itemsets['length'] == 1) &
                   (frequent_itemsets['support'] >= 0.05) ]
```

Out[37]:

	support	itemsets	length
0	0.303	(Action)	1
1	0.259	(Adventure)	1
3	0.081	(Biography)	1
4	0.279	(Comedy)	1
5	0.150	(Crime)	1
6	0.513	(Drama)	1
7	0.051	(Family)	1
8	0.101	(Fantasy)	1
10	0.119	(Horror)	1
11	0.106	(Mystery)	1
12	0.141	(Romance)	1
13	0.120	(Sci-Fi)	1
14	0.195	(Thriller)	1

In [155]:

```
frequent_itemsets[ (frequent_itemsets['length'] >=3) &
                   (frequent_itemsets['support'] >= 0.01) ]
```

Out[155]:

	support	itemsets	length
47	0.027	(Adventure, Fantasy, Action)	3
48	0.050	(Adventure, Sci-Fi, Action)	3
49	0.027	(Adventure, Comedy, Animation)	3
50	0.035	(Comedy, Drama, Romance)	3
51	0.023	(Crime, Mystery, Drama)	3
52	0.024	(Thriller, Drama, Crime)	3

```
## Association Mining
```

In [95]:

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'Comedy', 'Romance'} ]
```

Out[95]:

	support	itemsets	length
32	0.071	(Comedy, Romance)	2

In [96]:

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'Comedy', 'Drama'} ]
```

Out[96]:

	support	itemsets	length
31	0.1	(Comedy, Drama)	2

In [105]:

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'Adventure', 'Action'} ]
```

Out[105]:

	support	itemsets	length
15	0.155	(Adventure, Action)	2

The association between certain genres has good support

For movies of the Comedy genre, it is usually related to the Romance genre and the Drama genre, with a support of 0.071 and 0.1 respectively.

For adventure genre films the support is 0.035 that they are also action genre, otherwise the support value

CONCLUSION

We apply a APRIORI ALGORITHM to a top rated movies dataset. The technique does not provide a recommendation in a fine-grained user level, but it does enable us to investigate an underlying relationship within the movies. We can utilize such findings to construct a new marketing campaign, research customer's behavior, or make a product suggestion.