FIRST DAY CONTENT VIEWERSHIP PROJECT

In [1]:

```
# Import useful Libraries for the Analysis as below
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import numpy as np
from collections import Counter
from scipy.stats import skew, kurtosis
```

In [2]:

```
# Import csv data supplied by the company
df = pd.read_csv(r"C:\Users\HENRY OKEOMA\Downloads\sttdata.csv")
```

In [3]:

```
# See data Frame to have a feel of the Data for Analysis
df.head()
```

Out[3]:

	visitors	ad_impressions	major_sports_event	genre	dayofweek	season	views_trailer	v
0	1.67	1113.81	0	Horror	Wednesday	Spring	56.70	
1	1.46	1498.41	1	Thriller	Friday	Fall	52.69	
2	1.47	1079.19	1	Thriller	Wednesday	Fall	48.74	
3	1.85	1342.77	1	Sci-Fi	Friday	Fall	49.81	
4	1.46	1498.41	0	Sci-Fi	Sunday	Winter	55.83	

In [4]:

```
# Check the duplicated Values (and The Data is very clean in that aspect with no duplica
df.duplicated().sum()
```

Out[4]:

```
0
```

In [9]:

```
# Data informations
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   visitors            1000 non-null   float64
 1   ad_impressions      1000 non-null   float64
 2   major_sports_event  1000 non-null   int64  
 3   genre               1000 non-null   object 
 4   dayofweek           1000 non-null   object 
 5   season              1000 non-null   object 
 6   views_trailer       1000 non-null   float64
 7   views_content       1000 non-null   float64
dtypes: float64(4), int64(1), object(3)
memory usage: 62.6+ KB
```

All entries are complete, no missing values, DataFrame looks clean so far. All the column have 1000 enteries across the features. Also in this DataFrame, we have one integer, four floats and 3 objects(which are categorical data)

In [7]:

```
# Check the Numericals Stats of the Data set
round(df.describe(),3)
```

Out[7]:

	visitors	ad_impressions	major_sports_event	views_trailer	views_content
count	1000.000	1000.000	1000.00	1000.000	1000.000
mean	1.704	1434.712	0.40	66.916	0.473
std	0.232	289.535	0.49	35.001	0.106
min	1.250	1010.870	0.00	30.080	0.220
25%	1.550	1210.330	0.00	50.948	0.400
50%	1.700	1383.580	0.00	53.960	0.450
75%	1.830	1623.670	1.00	57.755	0.520
max	2.340	2424.200	1.00	199.920	0.890

From here, we can see that the Visitors and Views_content column have a fairly normal distribution figures observing the Mean and the Median (50%). We expect to see some postive skewness in the other columns.

In [8]:

```
# Check the Categorical Data Stats
df.describe(include=["object", "bool"])
```

Out[8]:

	genre	dayofweek	season
count	1000	1000	1000
unique	8	7	4
top	Others	Friday	Winter
freq	255	369	257

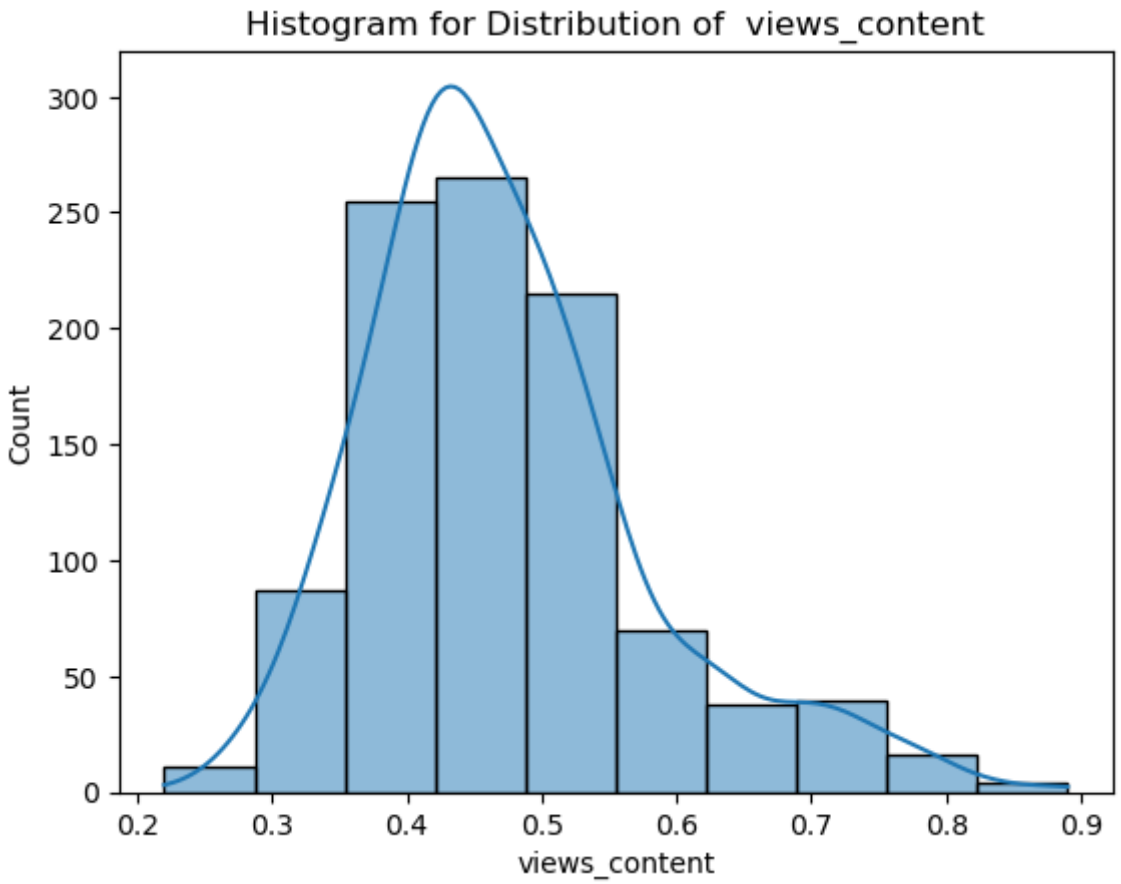
In [32]:

```
# Create a Function that calculates and adds percentage to the graph plots as below
def perc_plot(plot, feature):
    total = len(feature)
    for p in plot.patches:
        perc = '{:.1f}%'.format(100 * p.get_height()/total)
        x = p.get_x() + p.get_width()/2 - 0.5
        y = p.get_y() + p.get_height()
        ax.annotate(perc,(x,y), size=12)
    plt.show()
```

UNIVARIATE ANALYSIS for NUMBERICAL FEATURES

In [58]:

```
# Checking the Viewers Content distribution
plt.title('Histogram for Distribution of  views_content')
sns.histplot(x='views_content', data=df, kde=True, bins=10);
```



In [11]:

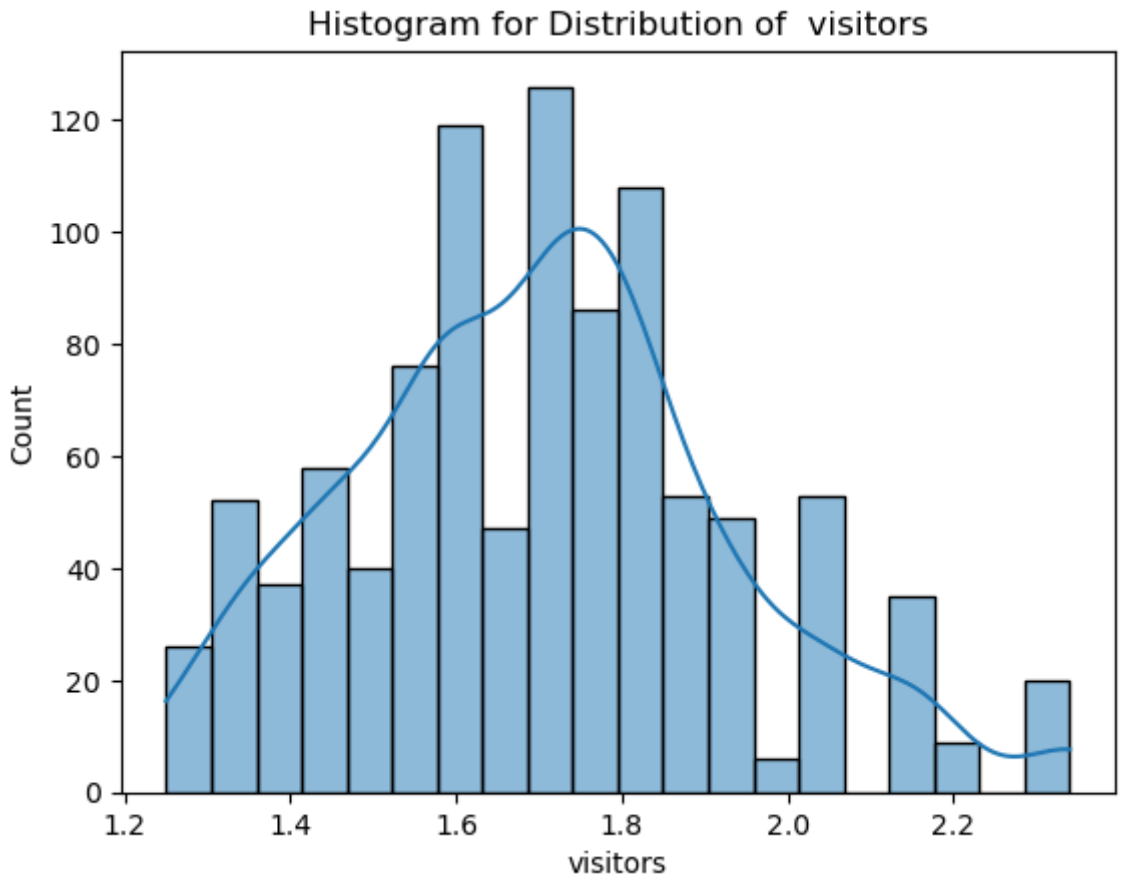
```
print(df['views_content'].skew())
print(df['views_content'].kurtosis())
```

```
0.9428471566302183
1.0781309765492133
```

The Skew = 0.9: which signifies a moderately skewed distribution of the views_content data (Skewed to the right) Kurtosis = 1.07: figure less than 3, which signifies a low kurtosis, not much of tailing to the right (Platykurtic distributions)

In [57]:

```
# Checking the distribution of Visitors to the platform
plt.title('Histogram for Distribution of  visitors')
sns.histplot(x='visitors', data=df, kde=True);
```



In [13]:

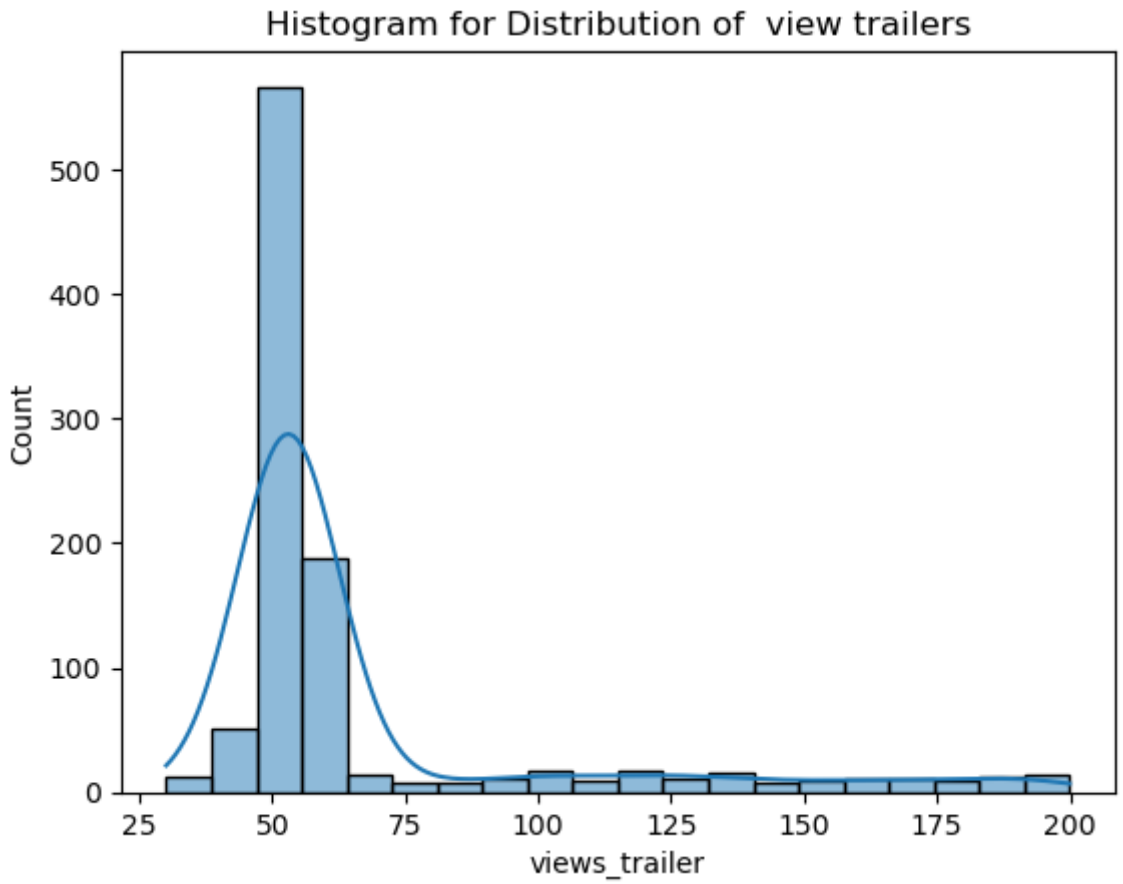
```
print(df['visitors'].skew())
print(df['visitors'].kurtosis())
```

```
0.370597770139047
0.016429566826573705
```

We can observe that both the skewness and Kurtosis are good, also almost normally distributted with moderate skewness

In [56]:

```
# Univariate Distribution of the views_trailer Feature
plt.title('Histogram for Distribution of  view trailers')
sns.histplot(x='views_trailer', data=df, kde=True, bins=20);
```



In [18]:

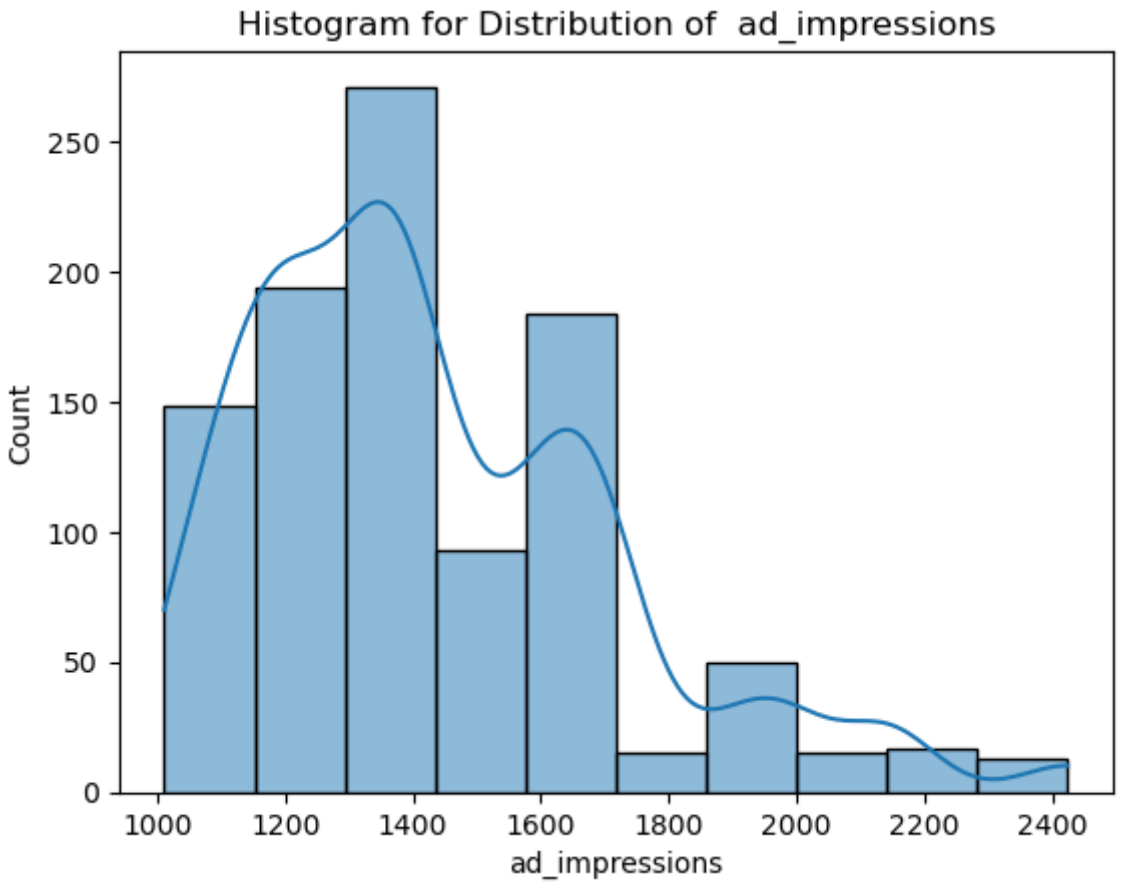
```
print(df['views_trailer'].skew())
print(df['views_trailer'].kurtosis())
```

```
2.37291116913243
4.610648458456113
```

The view_trailer feature is postively skewed and have outliers on both sides of the distribution. Both skew and kurtosis is off the exepectd figures

In [54]:

```
# Univariate Distribution of the ad_impressions Feature
plt.title('Histogram for Distribution of  ad_impressions')
sns.histplot(x='ad_impressions', data=df, kde=True, bins=10);
```



In [17]:

```
print(df['ad_impressions'].skew())
print(df['ad_impressions'].kurtosis())
```

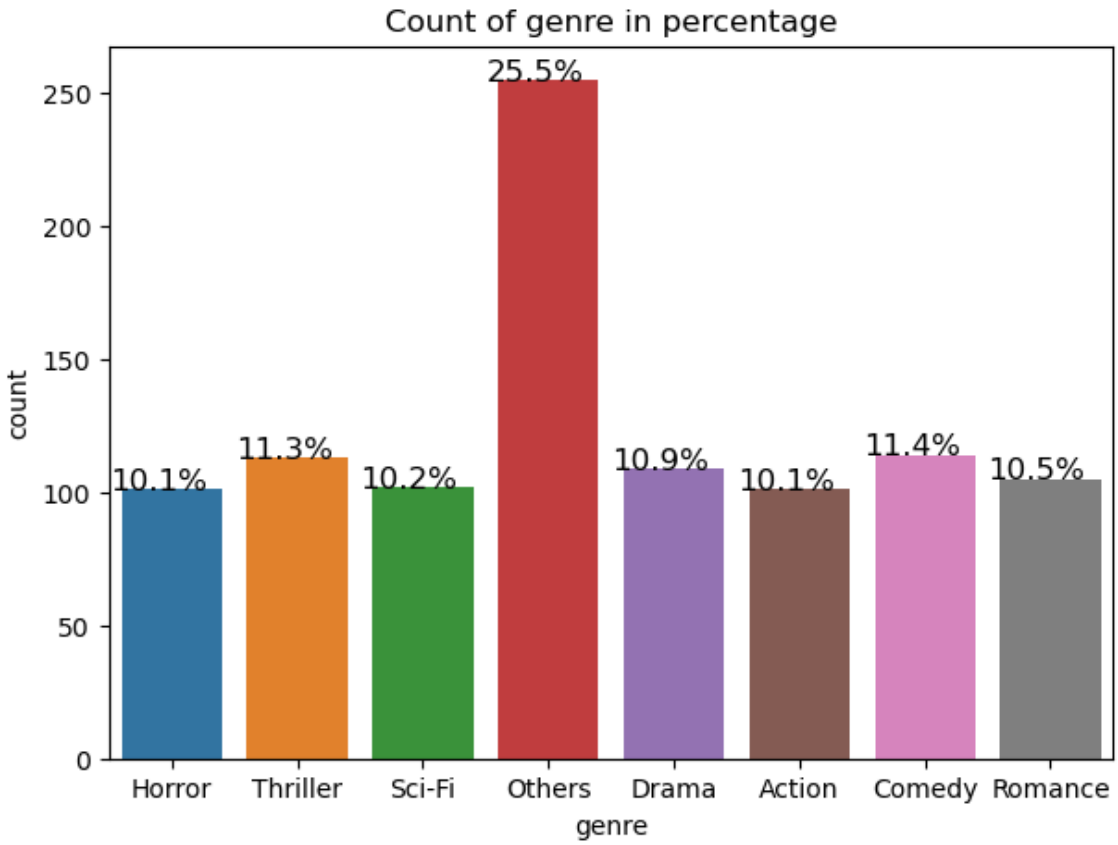
```
1.0335622338187542
0.9777283266214765
```

The ad_impressions have no normal distribution, but have a low Kurtosis, in other words; Platykurtic

UNIVARIATE ANALYSIS for CATEGORICAL FEATURES

In [53]:

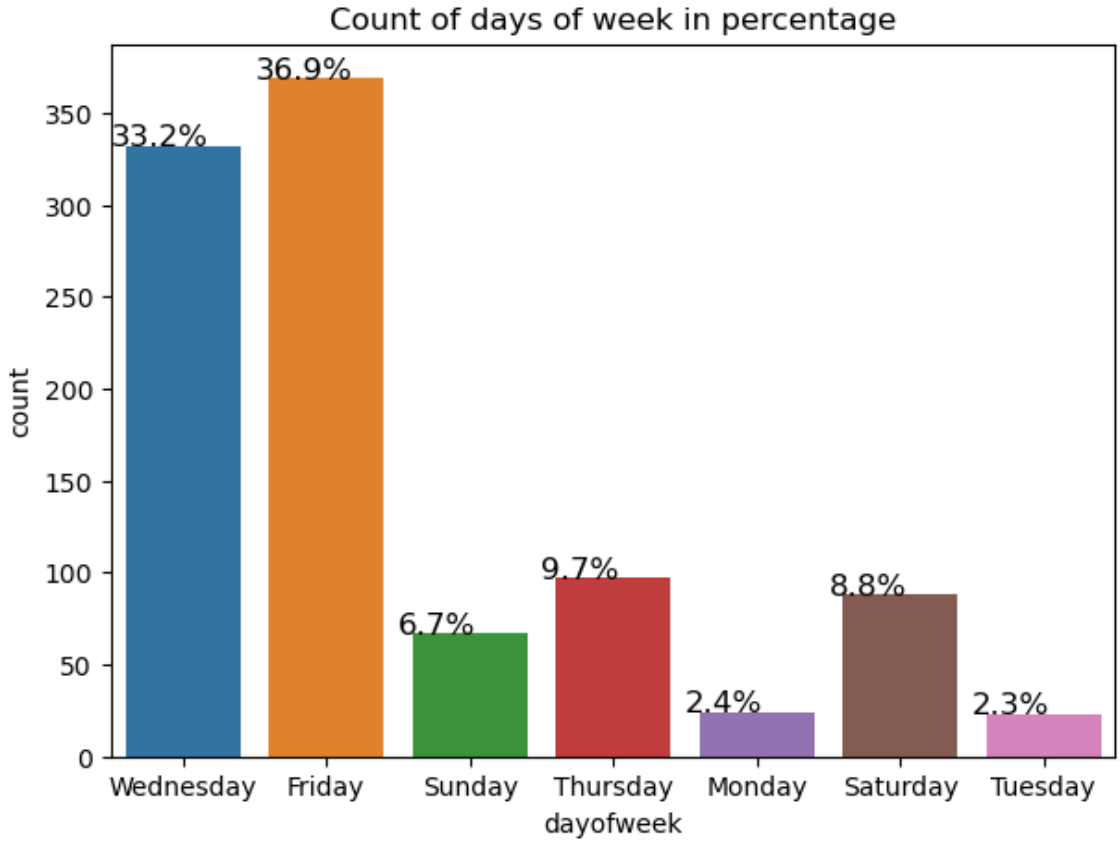
```
# Univariate Plot of the Catergorical Variable 'GENRE'
plt.figure(figsize=(7,5))
plt.title('Count of genre in percentage')
ax = sns.countplot(x='genre', data=df)
perc_plot(ax, df.genre)
```



The univariate analysis of the genre information are almost evenly spread across in percentages with no significant difference between the types; with Thriller and Comedy recording the highest. The 'Others' with 25% is not well defined.

In [38]:

```
# Univariate Plot of the Catergorical Variable 'Dayofweek'
plt.figure(figsize=(7,5))
plt.title('Count of days of week in percentage')
ax = sns.countplot(x='dayofweek', data=df)
perc_plot(ax, df.dayofweek)
```



The univariate analysis of the day of week have friday as the highest, followed by wednesdays. Mondays and Tuesdays are the least.

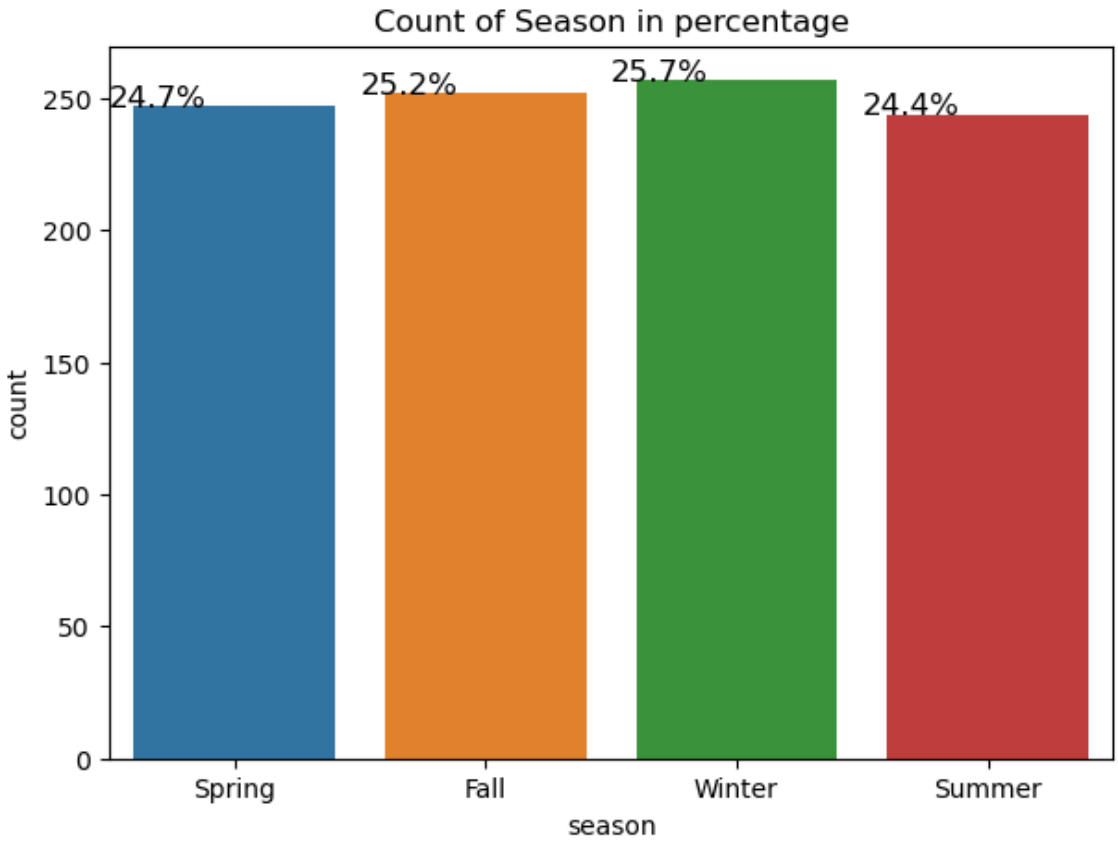
In [ ]:

```
sns.catplot(data=df,
    y="views_content", x="season", kind="boxen")
```

In [39]:

```
# Univariate Plot of the Catergorical Variable 'Season'

plt.figure(figsize=(7,5))
plt.title('Count of Season in percentage')
ax = sns.countplot(x='season', data=df)
perc_plot(ax, df.season)
```

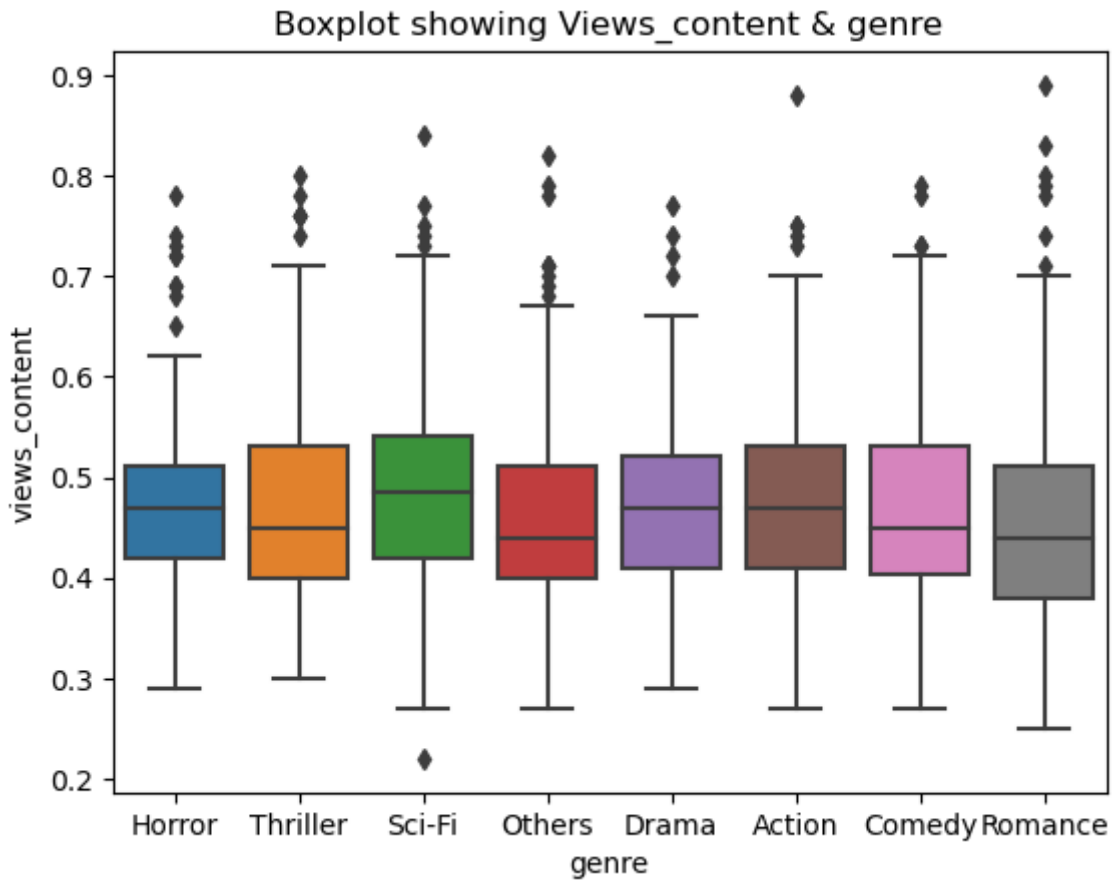


The seasons have almost equal numbers of counts

BIVARIATE ANALYSIS

In [52]:

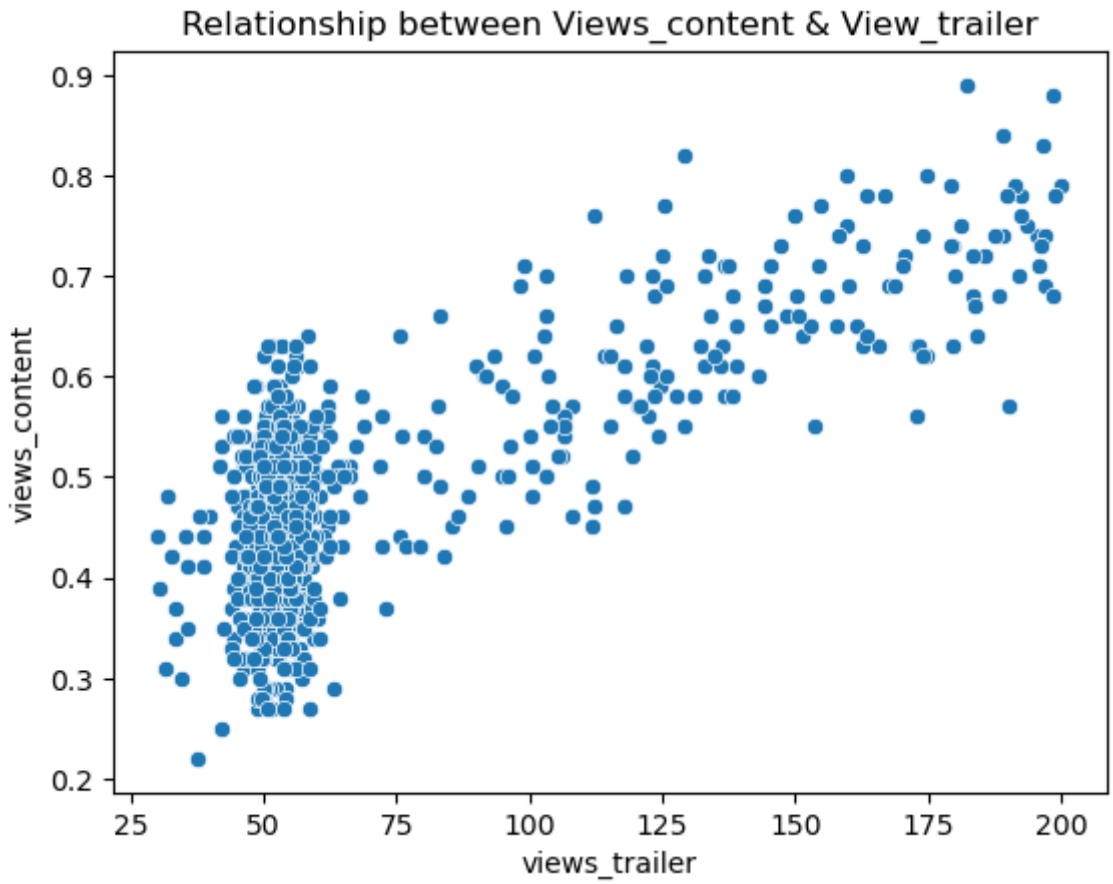
```
#Bivariate distribution examing the relationship between genre and views content
plt.title('Boxplot showing Views_content & genre')
sns.boxplot(x='genre', y='views_content', data=df);
```



From the boxplot above, we can see that the first day content view is higher for scifi and action. Although comedy and thriller also record high numbers. There is not a much signifacnt difference across the genre and thr first day views on the contents

In [101]:

```
# Relationship betweeen the views contents and views trailer
plt.title('Relationship between Views_content & View_trailer')
sns.scatterplot(x='views_trailer', y='views_content', data=df);
```



There is a positive correlation between the view_trailer data and the first day views content. The more people view the trailers, most likely they are to follow the contents to its first release day. Hence the view trailer feature is a key feature for the first day views content.

In [60]:

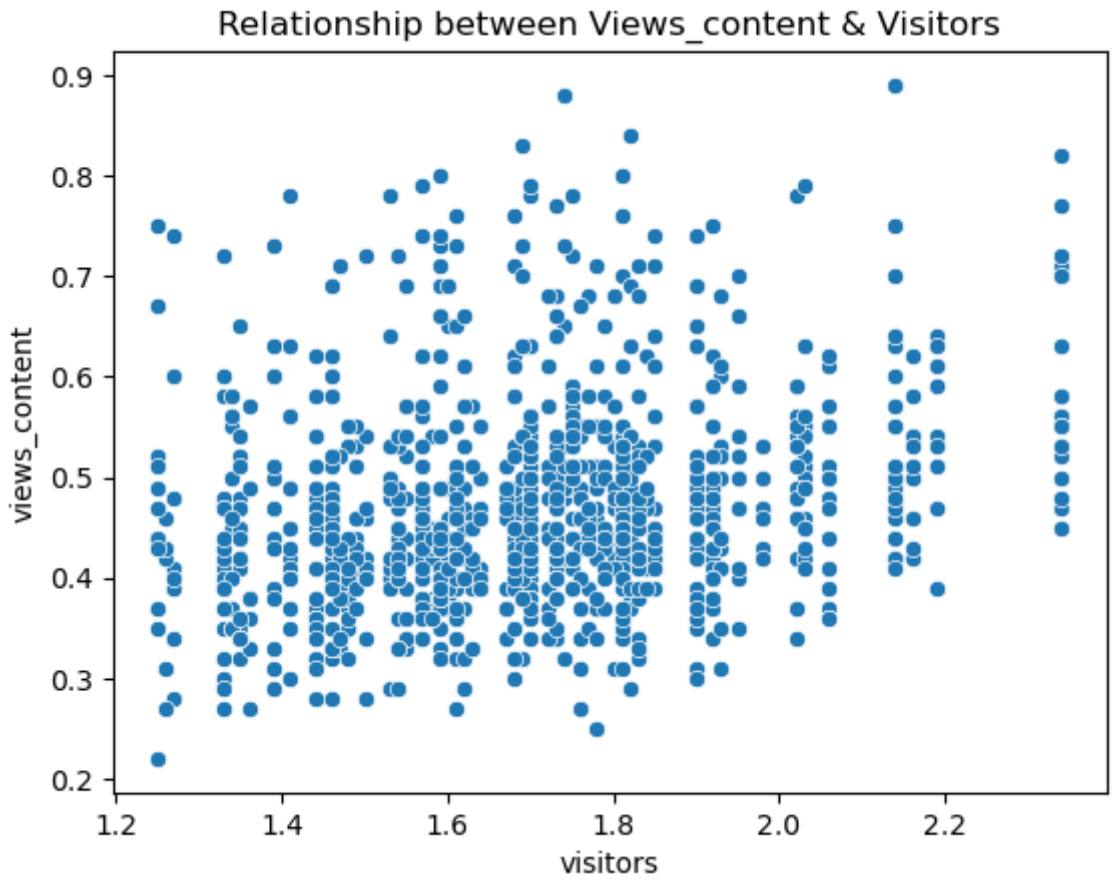
```
df.head(2)
```

Out[60]:

	visitors	ad_impressions	major_sports_event	genre	dayofweek	season	views_trailer	v
0	1.67	1113.81	0	Horror	Wednesday	Spring	56.70	
1	1.46	1498.41	1	Thriller	Friday	Fall	52.69	

In [140]:

```
# Relationship betweeen the views contents and ad impressions
plt.title('Relationship between Views_content & Visitors')
ax = sns.scatterplot(x='visitors', y='views_content', data=df)
plt.show()
```



A scatterplot of the visitors against the first day views have a weak positive correlation.

In [29]:

```
# Days of the week where the first views are more
# First we have to group the days using the group by

dfv2 = df.groupby('dayofweek')['views_content'].sum().reset_index().sort_values(by='view
dfv2
```

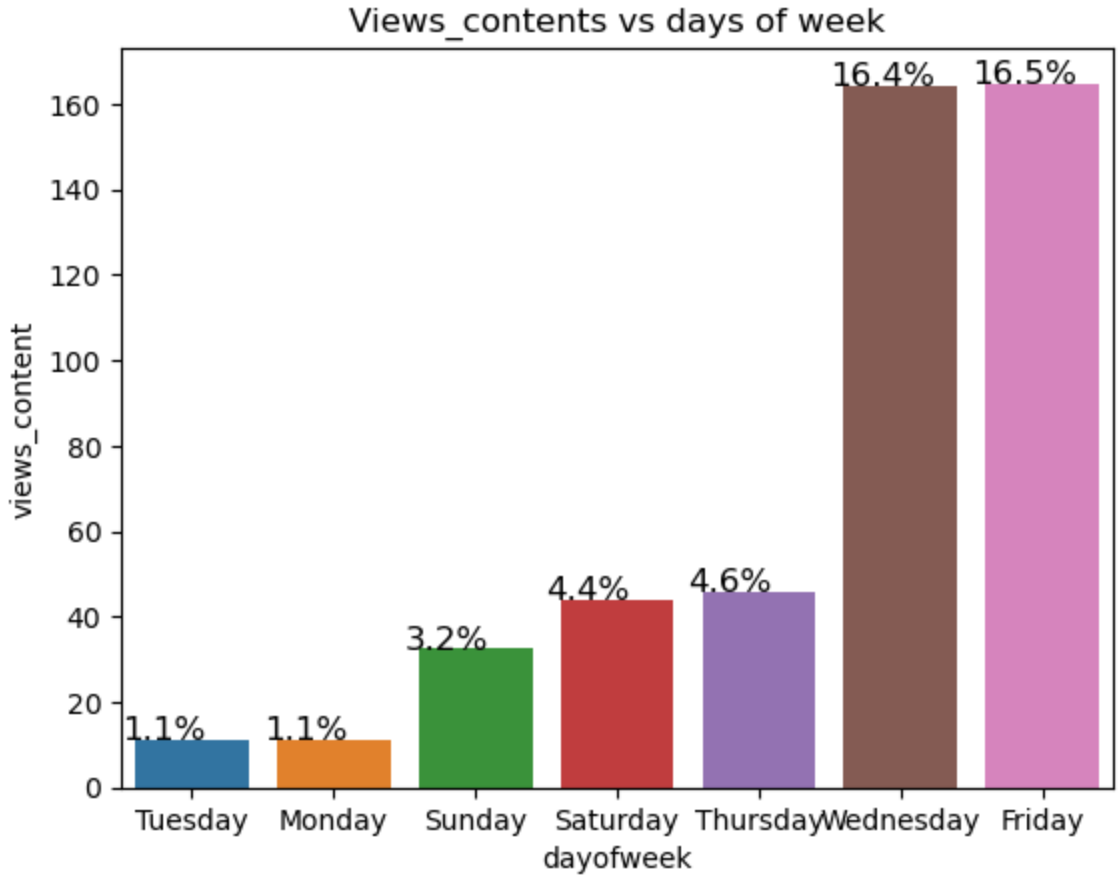
Out[29]:

	dayofweek	views_content
5	Tuesday	11.22
1	Monday	11.23
3	Sunday	32.44
2	Saturday	43.82
4	Thursday	45.65
6	Wednesday	164.21
0	Friday	164.83

We can see that Fridays and Wednesdays are most pronouced for first content views. We shall then use a Barplot to visualise the above data.

In [40]:

```
#Barplot of First day views content against the days of the week
plt.title('Views_contents vs days of week')
ax = sns.barplot(y='views_content', x='dayofweek', data=dfv2)
perc_plot(ax, df.dayofweek)
```



Fridays and Wednesdays are very distinct from the rest of the week for first day content view.

In [42]:

```
# Also we shall like to see the First views contents against the season
# We shall also groupby for the seasons
dfv3 = df.groupby('season')['views_content'].sum().reset_index()
dfv3
```

Out[42]:

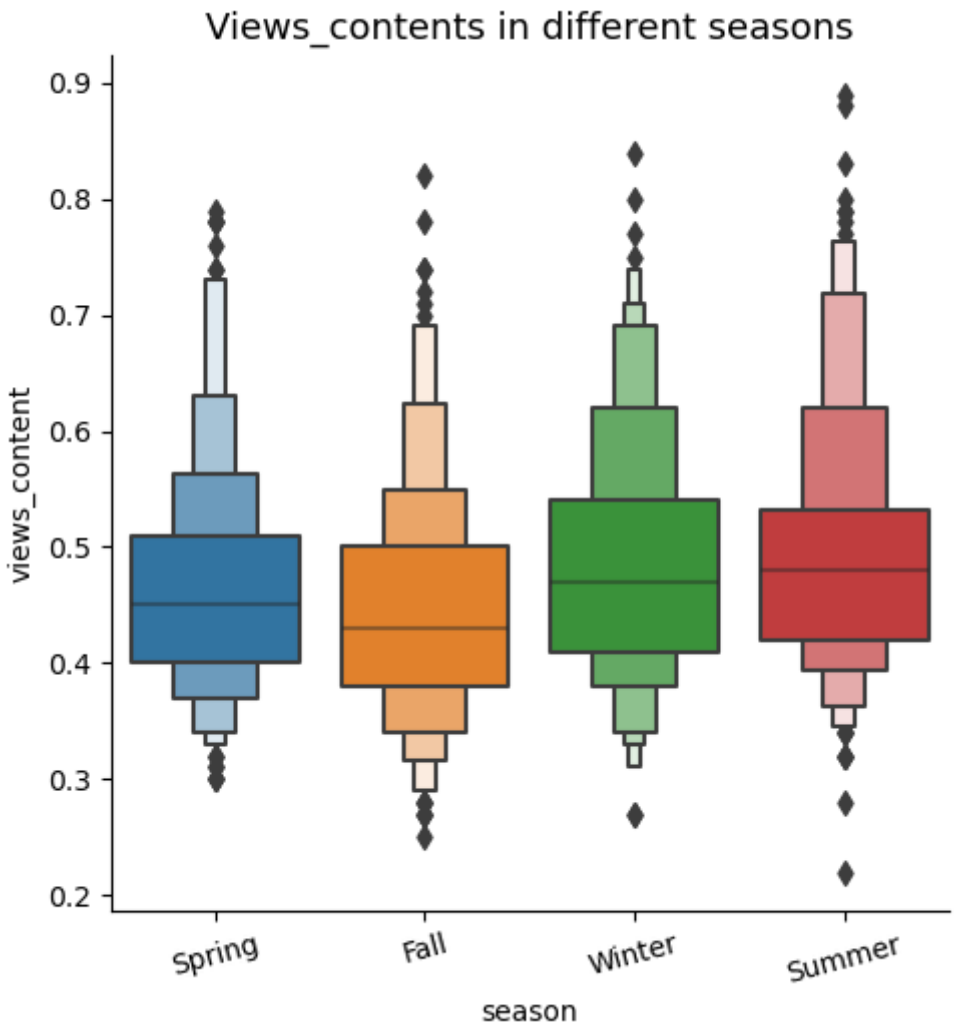
	season	views_content
0	Fall	112.23
1	Spring	115.39
2	Summer	121.22
3	Winter	124.56

Here we can see that the Summer and Winter have the highest numbers for first content views, we shall also use the catplot chart to visualise the above

In [23]:

```
# Plot of Views contents vs the seasons

sns.catplot(y='views_content', x='season', data=df, kind='boxen')
plt.title('Views_contents in different seasons', fontsize=13)
plt.xticks(rotation=15)
plt.show()
```



Although the differences in the seasons are not very significant, but the minimum data still points towards Summer and Winter and may become significant with increased data points.

In [83]:

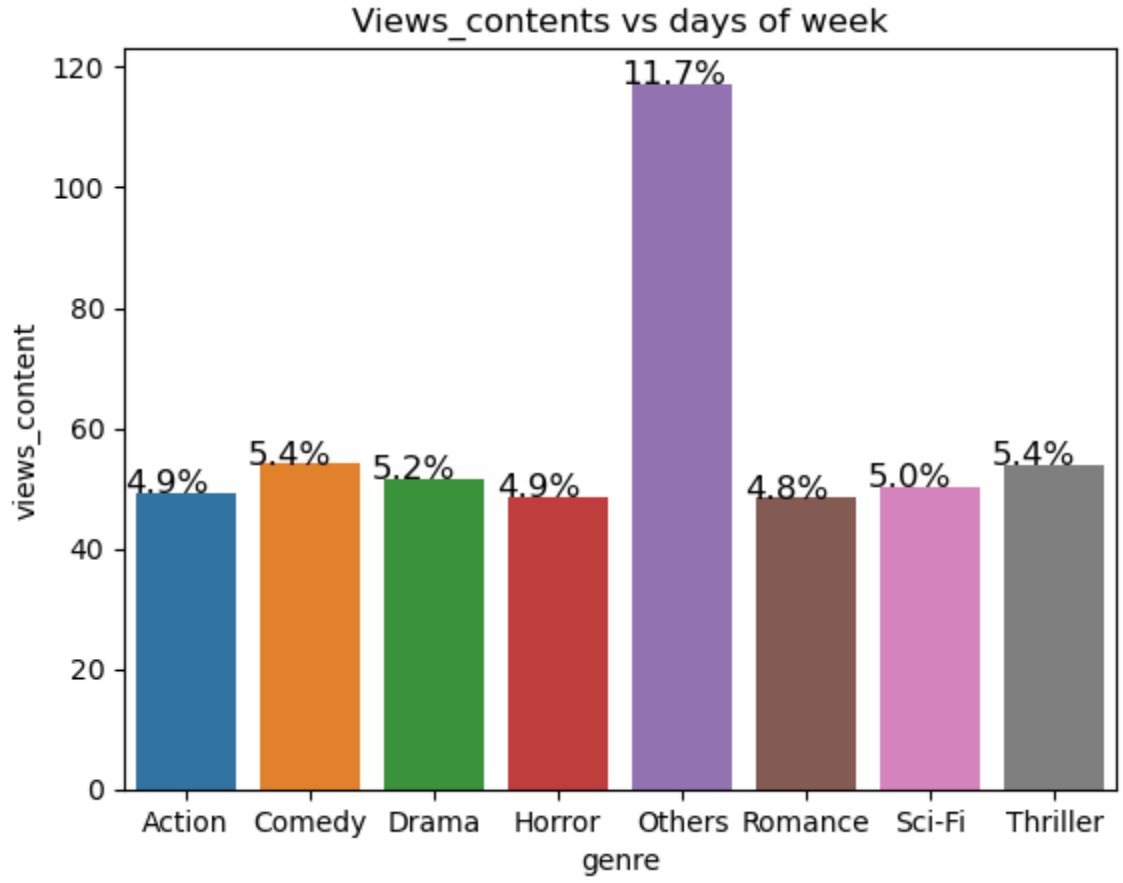
```
# Grouping the genre using the groupby and checking vs the views content
dfv4 = df.groupby('genre')['views_content'].sum().reset_index()
dfv4
```

Out[83]:

	genre	views_content
0	Action	49.22
1	Comedy	54.06
2	Drama	51.66
3	Horror	48.59
4	Others	117.24
5	Romance	48.43
6	Sci-Fi	50.31
7	Thriller	53.89

In [86]:

```
#Plot of Views content vs Genre
plt.title('Views_contents vs genre')
ax = sns.barplot(y='views_content', x='genre', data=dfv4)
perc_plot(ax, df.genre)
```



The above barplot returned similar characteristics as the count of genre above with the Comedy, thriller and

In [119]:

```
df.head(2)
```

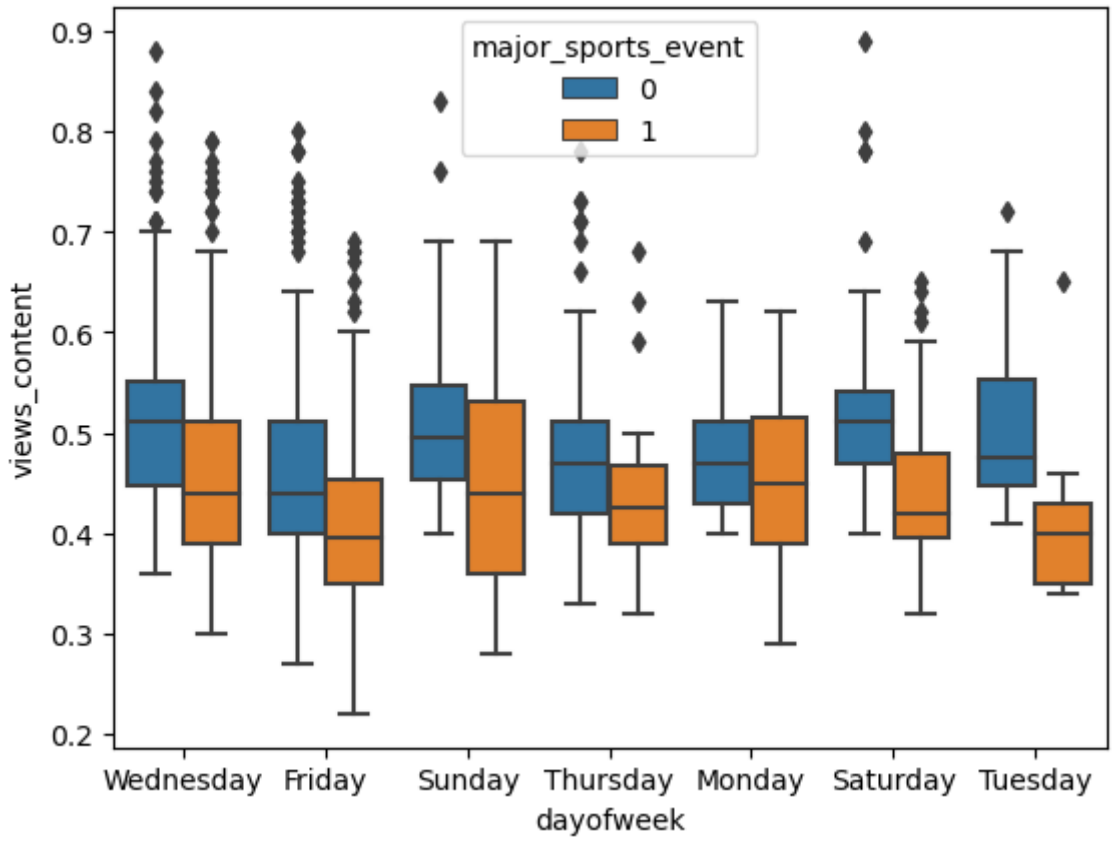
Out[119]:

	visitors	ad_impressions	major_sports_event	genre	dayofweek	season	views_trailer	v
0	1.67	1113.81	0	Horror	Wednesday	Spring	56.70	
1	1.46	1498.41	1	Thriller	Friday	Fall	52.69	

MULTIVARIATE ANALYSIS OF THE DATASET

In [37]:

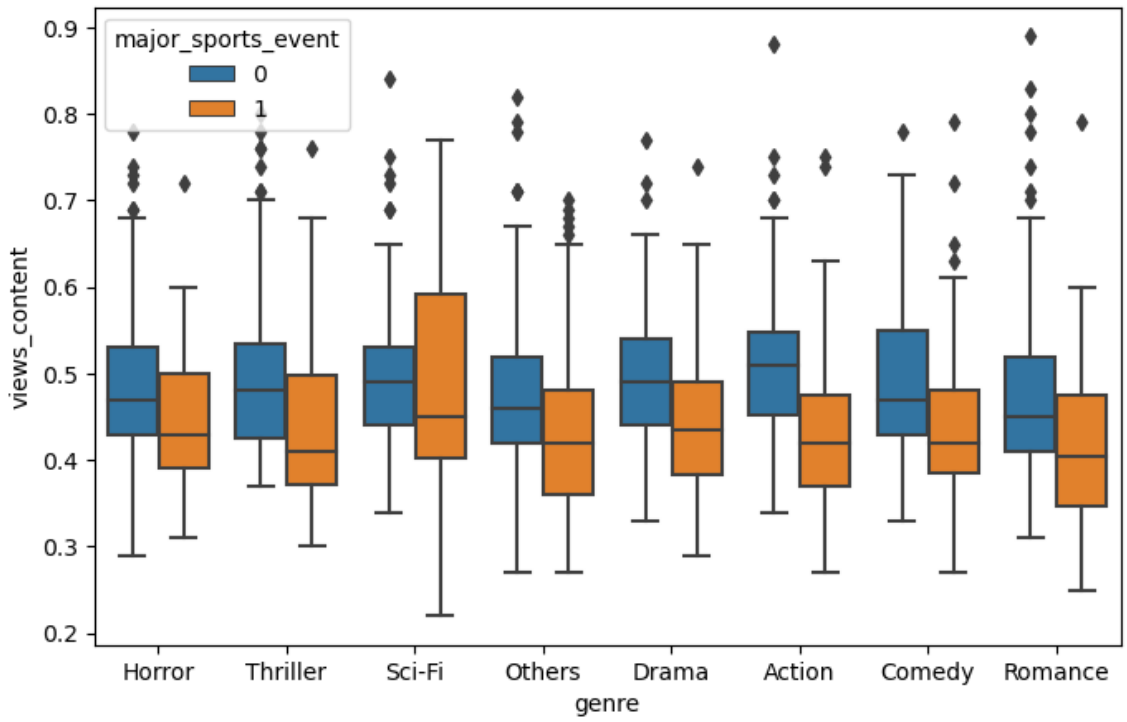
```
# View_content vs  Day of the week on a major sports event
sns.boxplot(x='dayofweek', y='views_content', hue='major_sports_event', data=df);
```



From the Data plot above, on the average, there is less content viewership on the days of major sports event

In [91]:

```
# Views_content vs genre on a major sports event
plt.figure(figsize=(8,5))
sns.boxplot(x='genre', y='views_content', hue='major_sports_event', data=df);
```



Major sports event affects all first day release views except the scifi genre. The scifi genre is higher during major sports event and its unaffected by the major sports event. Also there is more spread of views during the major sports event for the scifi recording no outliers in the spread

In [93]:

```
df.head(2)
```

Out[93]:

	visitors	ad_impressions	major_sports_event	genre	dayofweek	season	views_trailer	v
0	1.67	1113.81	0	Horror	Wednesday	Spring	56.70	
1	1.46	1498.41	1	Thriller	Friday	Fall	52.69	

In [82]:

```
## heatmeap to see the correlation between features.
# Generate a mask for the upper triangle (taken from seaborn example gallery)
mask = np.zeros_like(df.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
sns.set_style('whitegrid')
plt.subplots(figsize = (10,7))
sns.heatmap(df.corr(),
            annot=True,
            mask = mask,
            cmap = 'RdBu',
            linewidths=.9,
            linecolor='white',
            fmt='.2g',
            center = 0,
            square=True)
plt.title("Correlations Among Features", y = 1.03,fontsize = 20, pad = 40);
```

```
C:\Users\HENRY OKEOMA\AppData\Local\Temp\ipykernel_19380\1951444994.py:3:
DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool
`. To silence this warning, use `bool` by itself. Doing this will not modi
fy any behavior and is safe. If you specifically wanted the numpy scalar t
ype, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.or
g/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations)
  mask = np.zeros_like(df.corr(), dtype=np.bool)
```

Correlations Among Features

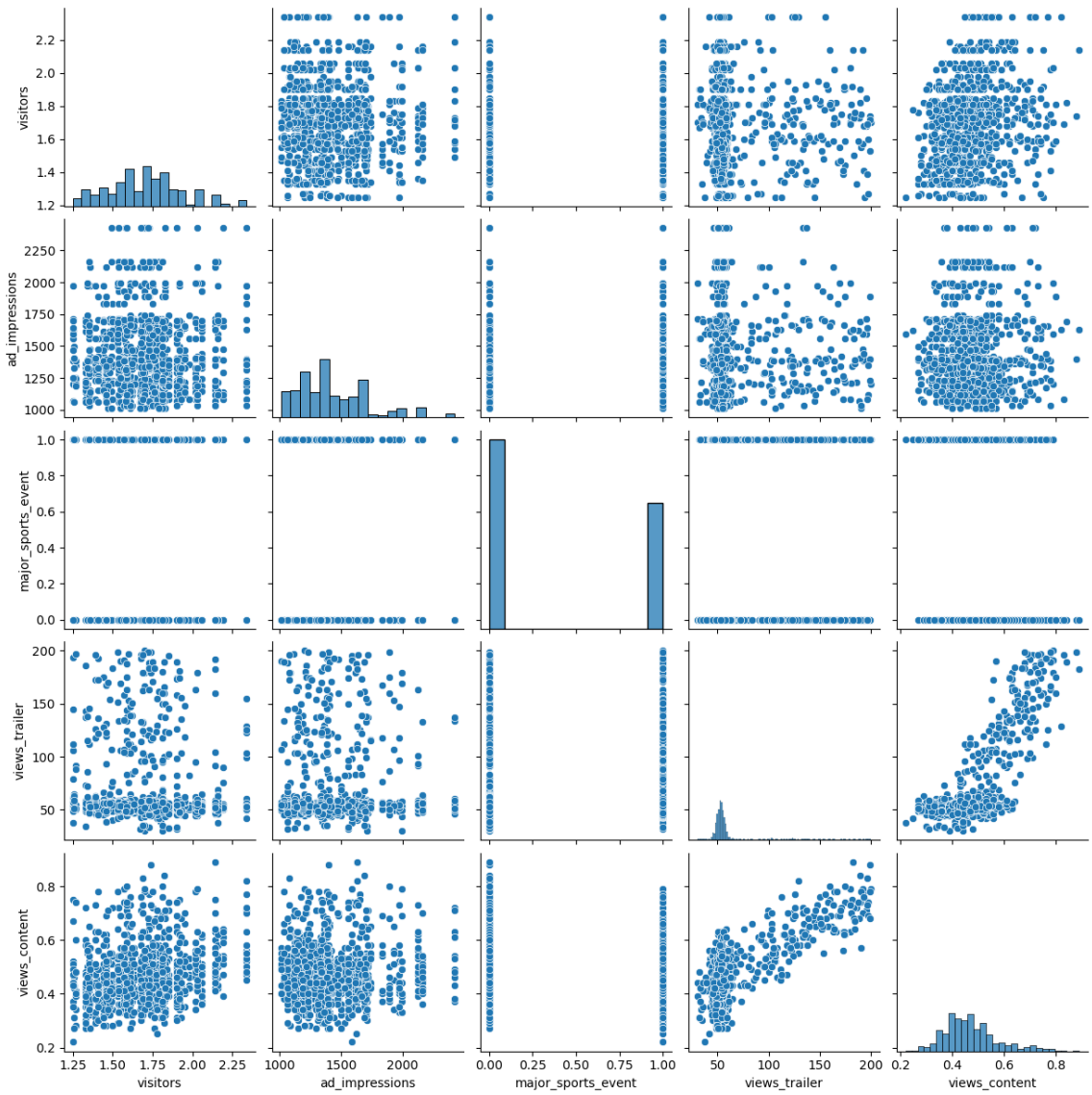
trailers and first
he views

visitors

0.6

```
sns.pairplot(df);
```

ad_impressions 0.03



In [ ]:

```
# ACtionable Insight and Recommendation
Major sports events affects the viewership, hence the company should endevour to mark ou
as it affect the first day views of the contents when released.
Also the company can consider including sports events to their streams to draw more view
```