In [60]:
```
import pandas as pd
import numpy as np
import re
import math
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

load movie data

In [61]:
```
org_data = pd.read_csv('/content/sample_data/movies_dataset.csv')
```

数据摘要

标称数据： appropriate_for, director, industry, language, storyline, title, id and writer

其中有意义的： appropriate_for, director, industry, language and writer

storyline, title, id 三者几乎是unique 但数据存在问题,存在部分重复样本

数值数据： imdb_rating, downloads, posted_date, release_date, run_time, views

都具有一定意义

In [62]:
```
nominals = ['appropriate_for', 'director',  'industry', 'language', 'writ
numerics = ['imdb_rating', 'downloads', 'posted_date', 'release_date', 'r
# drop first col
data = org_data.drop(columns='Unnamed: 0')
# Convert column names into snake_case.
data.columns = data.columns.str.replace('-', '_').str.lower()

# Make views and dowloads numeric.
for col in 'downloads', 'views':
    data[col] = data[col].str.replace(',','')
    data[col] = data[col].astype('float')
# Make id strings.
data['id'] = data['id'].astype('str')
# Output formte
pd.options.display.float_format = '{:.2f}'.format
def run_time_process(e):
    e = str(e).replace(' ', '')
    if e == 'nan':
        return np.nan
    if 'h' not in e and 'min' not in e:
        return int(e)
    else:
```

```
        hour = 0
        minute = 0
        if 'h' in e:
            hour = int(e.split('h')[0])
        if 'min' in e:
            minute = int(e.split('min')[0].split('h')[-1])
        return int(hour * 60 + minute)
## Run time process
# Convert '1h20min' format to minutes
data['run_time'] = data['run_time'].apply(run_time_process)
# Convert '102' format to minutes
data['run_time'] = pd.to_numeric(data['run_time'], errors='coerce').fillna
# Make dates datetime.
data['old_posted_date'] = data['posted_date']
data['posted_date'] = pd.to_datetime(data['posted_date'])

data['old_release_date'] = data['release_date']
data['release_date'] = pd.to_datetime(data['release_date'])
```

In [63]:
```
data
```

Out[63]:

	imdb_rating	appropriate_for	director	downloads	id	industry	langua
0	4.80	R	John Swab	304.00	372092	Hollywood / English	Eng
1	6.40	TV-PG	Paul Ziller	73.00	372091	Hollywood / English	Eng
2	5.20	R	Ben Wheatley	1427.00	343381	Hollywood / English	English,Hi
3	8.10	NaN	Venky Atluri	1549.00	372090	Tollywood	Hi
4	4.60	NaN	Shaji Kailas	657.00	372089	Tollywood	Hi
...	...	...	...	...	...	...	
20543	NaN	NaN	NaN	1998.00	28957	Bollywood / Indian	Hi
20544	7.70	NaN	Bimal Roy	6080.00	28958	Bollywood / Indian	Hi
20545	8.00	NaN	NaN	3276.00	30459	Bollywood / Indian	Hi
20546	NaN	NaN	NaN	309.00	371669	Wrestling	Eng
20547	NaN	NaN	NaN	2613.00	371816	Wrestling	Eng

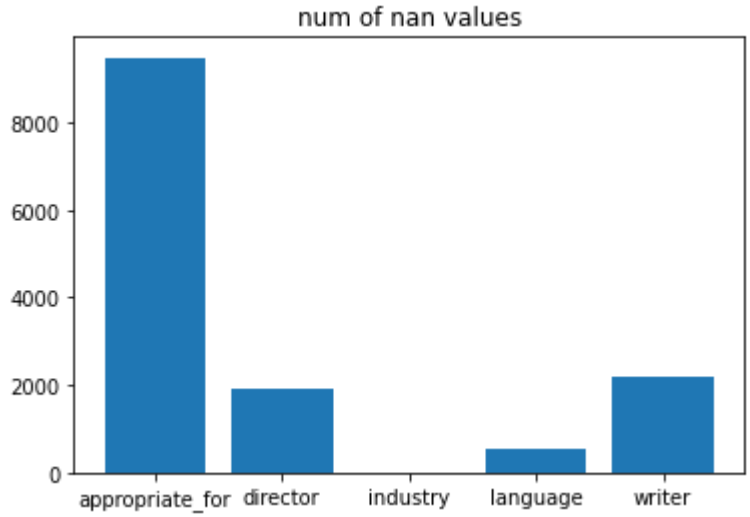
20548 rows × 16 columns

标称属性

标称属性的缺失值的个数

In [64]:
```
ax = nominals
ay = []
for attr in nominals:
    freq = 5
    ay.append(data[attr].isna().sum())
plt.bar(ax, ay)
plt.title('num of nan values')
```

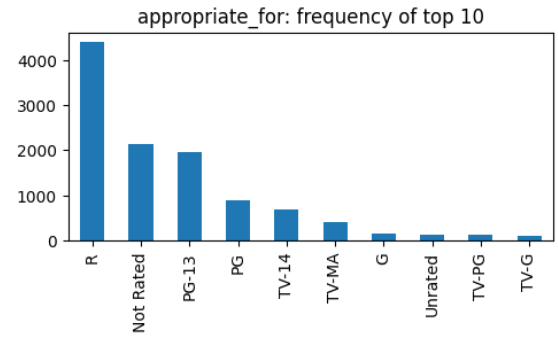
Out[64]:
```
Text(0.5, 1.0, 'num of nan values')
```

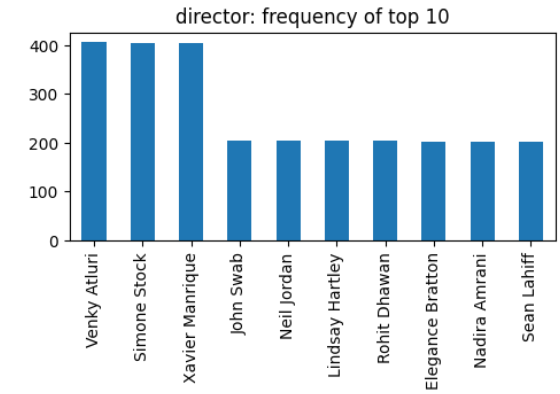


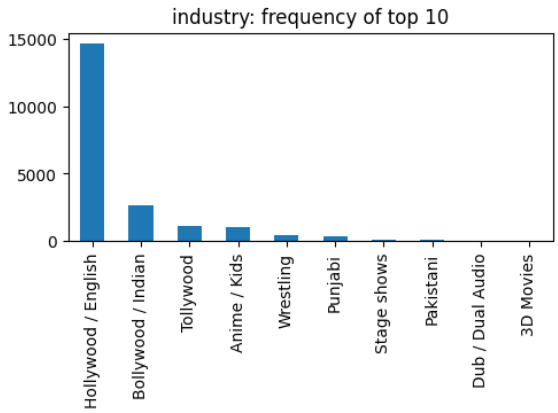
标称属性的每个可能取值的频数

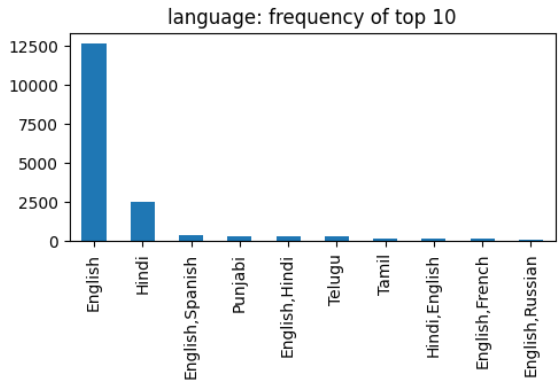
通过.value_counts()取得，这里仅展示频度前五的结果

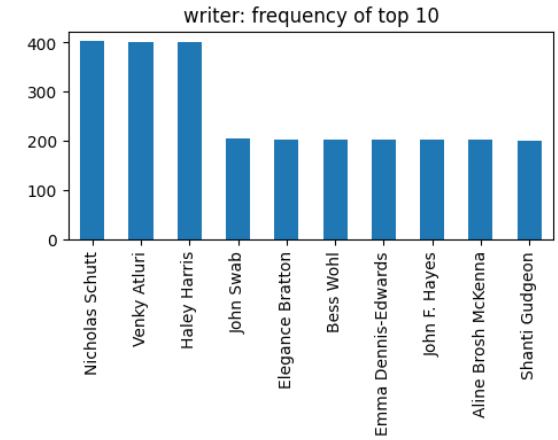
In [65]:
```
index = 1
plt.figure(figsize=(12,12), dpi=100).subplots_adjust(hspace=1)
plt.figure(1)
col = 2
row = int(len(nominals) / col) + 1
for attr in nominals:
    plt.subplot(row, col, index)
    index += 1
    freq = 10
    data[attr].value_counts().head(freq).plot.bar()
    plt.title(f'{attr}: frequency of top {freq}')
```









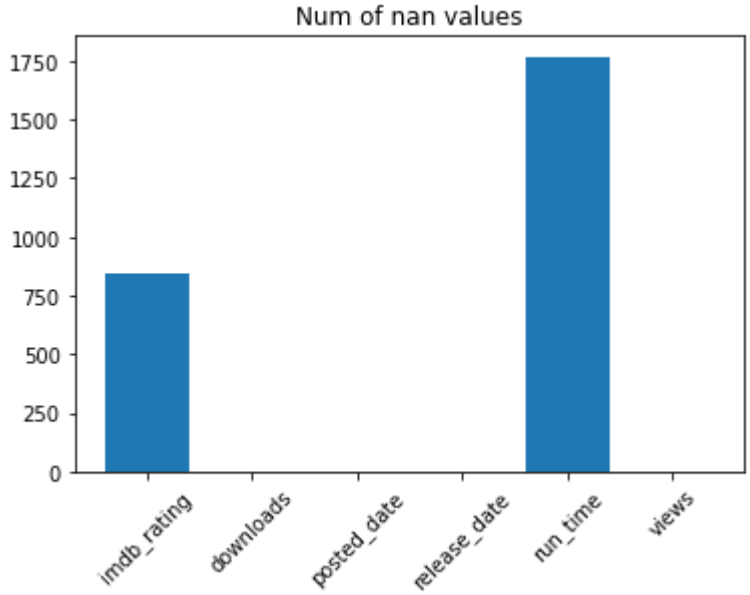


数值属性

数值属性的缺失值个数

In [66]:
```
ax = range(len(numerics))
ay = []
for attr in numerics:
    freq = 5
    ay.append(data[attr].isna().sum())
plt.bar(ax, ay)
plt.xticks(ax, numerics, rotation=45)
plt.title('Num of nan values')
```

Out[66]:
```
Text(0.5, 1.0, 'Num of nan values')
```



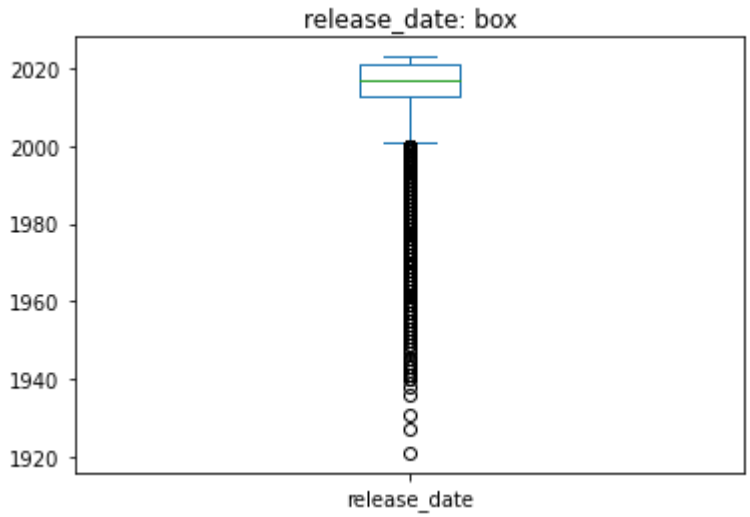
数值属性的五数、盒图

release date

处理时 仅关注年份信息

In [67]:
```
attr = 'release_date'
print(data[attr].dt.year.describe())
visit = pd.DataFrame(data[attr].dt.year)
visit.plot.box()
plt.title(attr + ': box')
plt.show()
```

```
count   20547.00
mean     2013.70
std        12.77
min      1921.00
25%      2013.00
50%      2017.00
75%      2021.00
max      2023.00
Name: release_date, dtype: float64
```

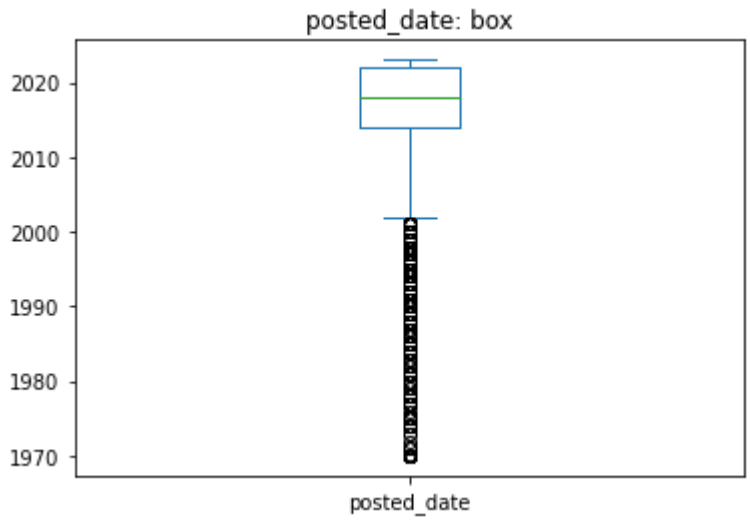


posted date

处理时仅关注年份信息

In [68]:
```
attr = 'posted_date'
print(data[attr].dt.year.describe())
visit = pd.DataFrame(data[attr].dt.year)
visit.plot.box()
plt.title(attr + ': box')
plt.show()
```

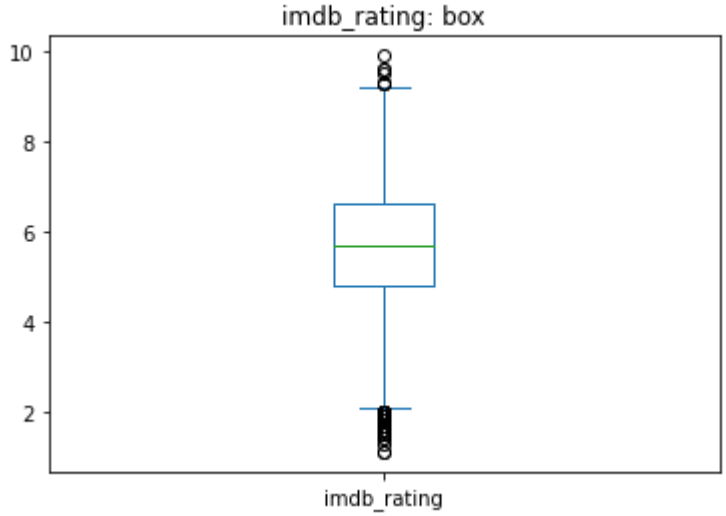
```
count   20547.00
mean     2017.00
std         6.10
min      1970.00
25%      2014.00
50%      2018.00
75%      2022.00
max      2023.00
Name: posted_date, dtype: float64
```



imdb rating

In [69]:
```
attr = 'imdb_rating'
print(data[attr].describe())
visit = pd.DataFrame(data[attr])
visit.plot.box()
plt.title(attr + ': box')
plt.show()
```

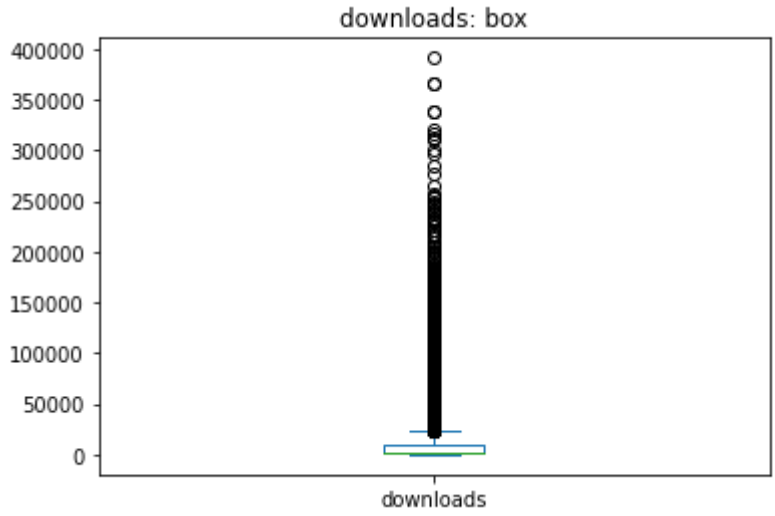
```
count   19707.00
mean        5.76
std         1.37
min         1.10
25%         4.80
50%         5.70
75%         6.60
max         9.90
Name: imdb_rating, dtype: float64
```



downloads

In [70]:
```
attr = 'downloads'
print(data[attr].describe())
visit = pd.DataFrame(data[attr])
visit.plot.box()
plt.title(attr + ': box')
plt.show()
```

```
count    20547.00
mean     10795.24
std      23716.18
min          0.00
25%        855.50
50%       2716.00
75%      10070.00
max     391272.00
Name: downloads, dtype: float64
```



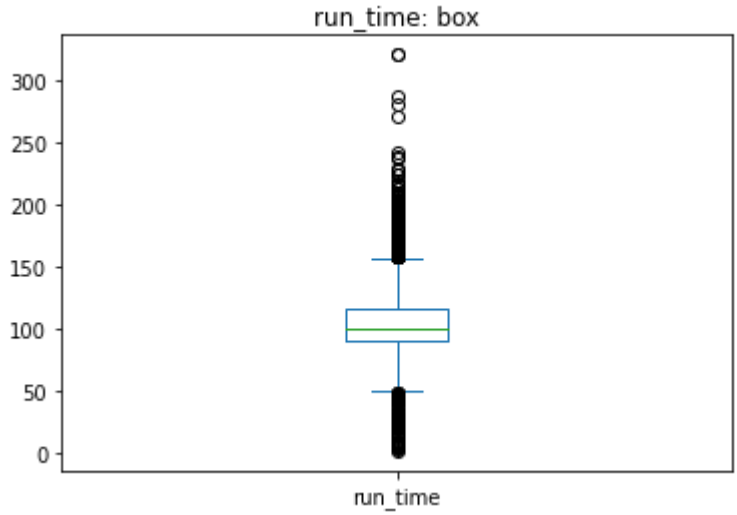
run_time

单位为分钟

In [71]:
```
attr = 'run_time'
print(data[attr].describe())
visit = pd.DataFrame(data[attr])
```

```
visit.plot.box()
plt.title(attr + ': box')
plt.show()
```

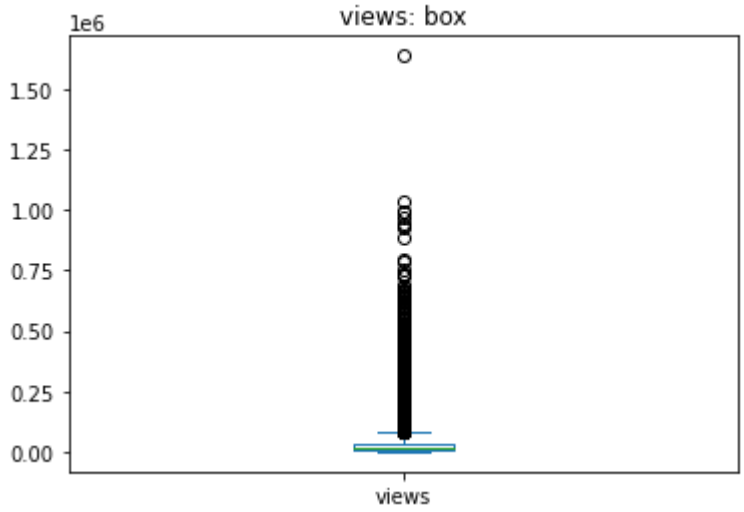
```
count   18780.00
mean      106.20
std        23.64
min         2.00
25%        90.00
50%       100.00
75%       117.00
max       321.00
Name: run_time, dtype: float64
```



views

In [72]:
```
attr = 'views'
print(data[attr].describe())
visit = pd.DataFrame(data[attr])
visit.plot.box()
plt.title(attr + ': box')
plt.show()
```

```
count     20547.00
mean      35595.51
std       62472.42
min         667.00
25%        7571.50
50%       15222.00
75%       36571.00
max     1638533.00
Name: views, dtype: float64
```



缺失值处理

剔除

剔除后仅剩9902条数据，远少于原数据量20548

In [73]:
```
new_data = data.dropna()
```

In [74]:
```
new_data
```

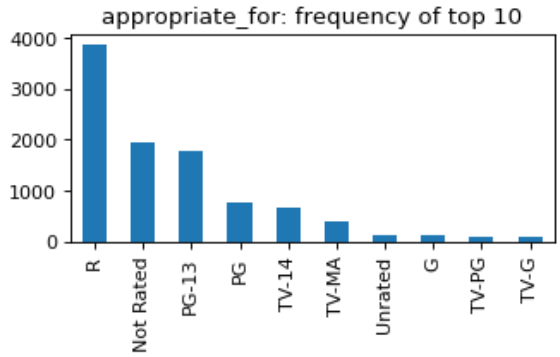
Out[74]:

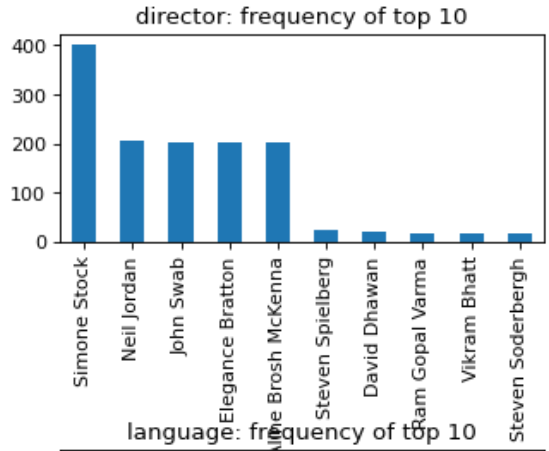
	imdb_rating	appropriate_for	director	downloads	id	industry	
0	4.80	R	John Swab	304.00	372092	Hollywood / English	
1	6.40	TV-PG	Paul Ziller	73.00	372091	Hollywood / English	
2	5.20	R	Ben Wheatley	1427.00	343381	Hollywood / English	
7	6.50	R	Benjamin Caron	1781.00	371751	Hollywood / English	
8	6.90	PG-13	Ravi Kapoor	458.00	372042	Hollywood / English	
...	...	...	...	...	...	...	
20522	7.10	Not Rated	Biren Nag	1932.00	23825	Bollywood / Indian	
20525	7.00	G	Guy Hamilton	2544.00	25548	Hollywood / English	English
20533	5.60	R	Barbara Topsøe-Rothenborg	12284.00	1173	Hollywood / English	Sp
20537	7.10	Not Rated	Biren Nag	1932.00	23825	Bollywood / Indian	
20540	7.00	G	Guy Hamilton	2544.00	25548	Hollywood / English	English

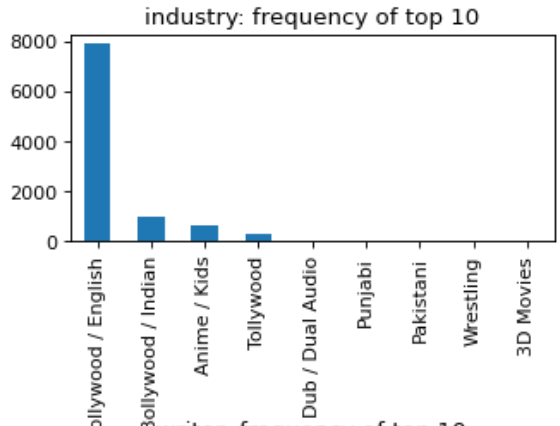
9902 rows × 16 columns

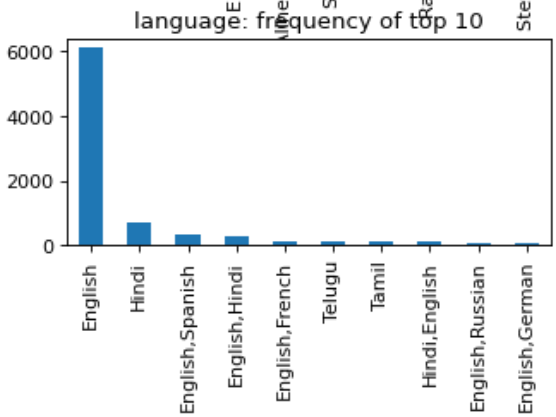
标称属性变化

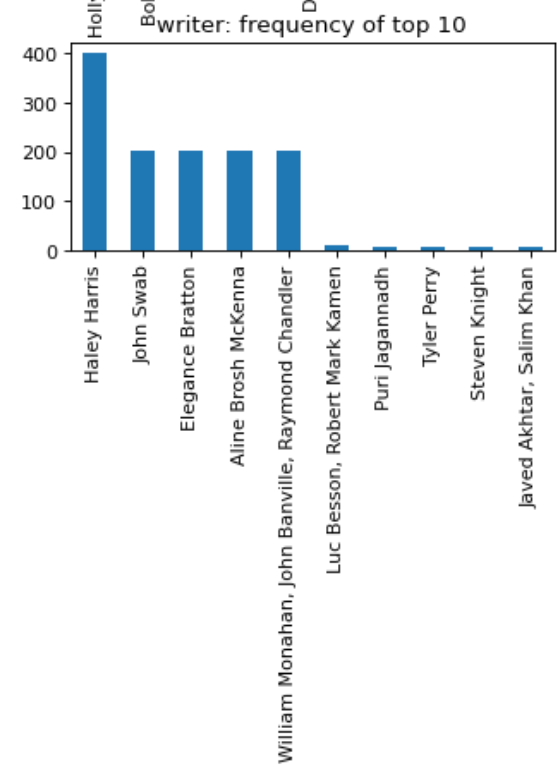
In [75]:
```
index = 1
plt.figure(figsize=(10,10), dpi=80).subplots_adjust(hspace=1)
plt.figure(1)
col = 2
row = int(len(nominals) / col) + 1
for attr in nominals:
    plt.subplot(row, col, index)
    index += 1
    freq = 10
    new_data[attr].value_counts().head(freq).plot.bar()
    plt.title(attr + ': frequency of top {}'.format(freq))
```









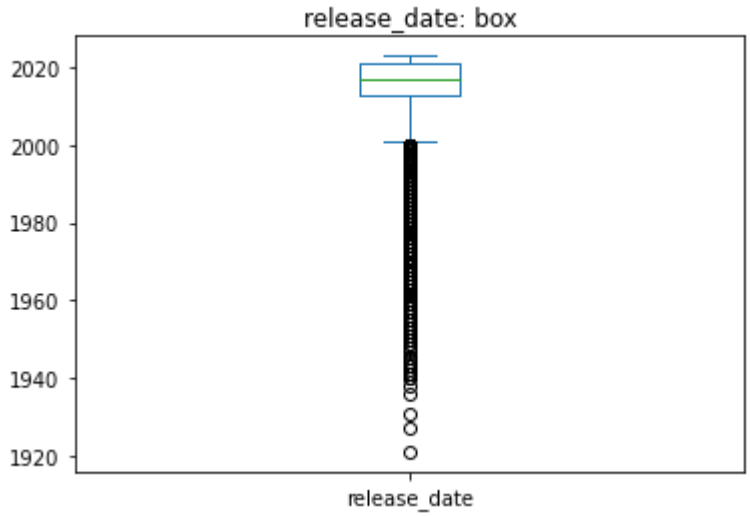


数值属性变化

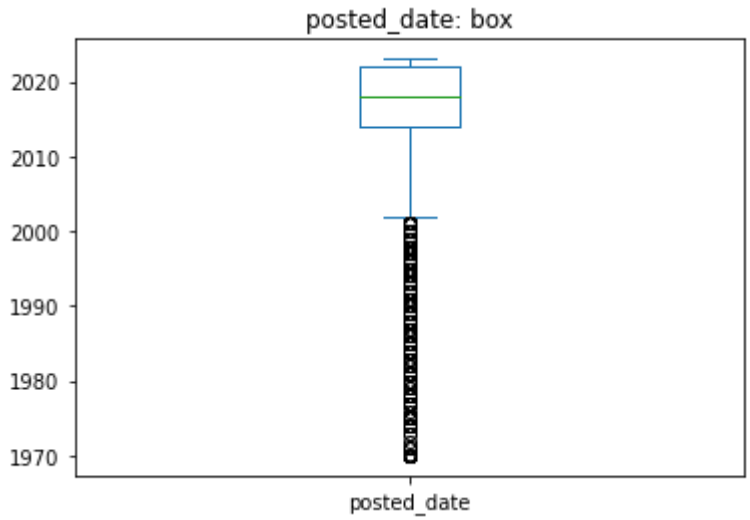
In [76]:
```
attrs = ['release_date', 'posted_date', 'imdb_rating', 'downloads', 'run_
for attr in attrs:
  print(attr)
  try:
    print(new_data[attr].dt.year.describe())
    visit = pd.DataFrame(data[attr].dt.year)
  except:
    print(new_data[attr].describe())
    visit = pd.DataFrame(data[attr])
```

```
  visit.plot.box()
  plt.title(attr + ': box')
  # plt.show()
```

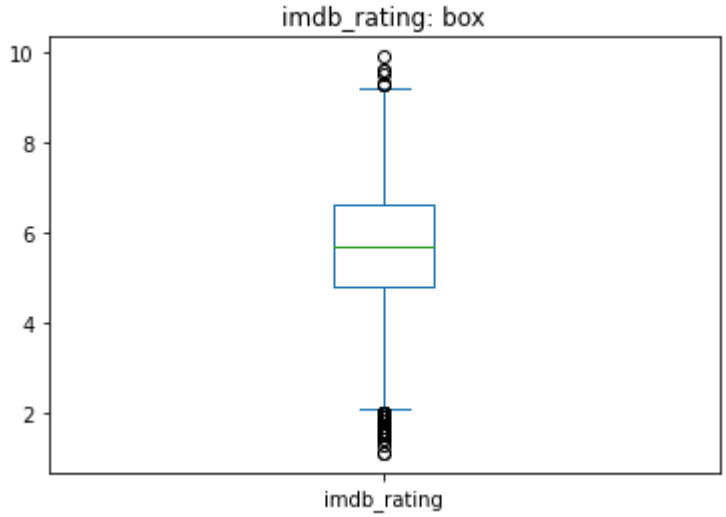
```
release_date
count   9902.00
mean    2013.67
std       10.98
min     1931.00
25%     2012.00
50%     2016.00
75%     2020.00
max     2023.00
Name: release_date, dtype: float64
```



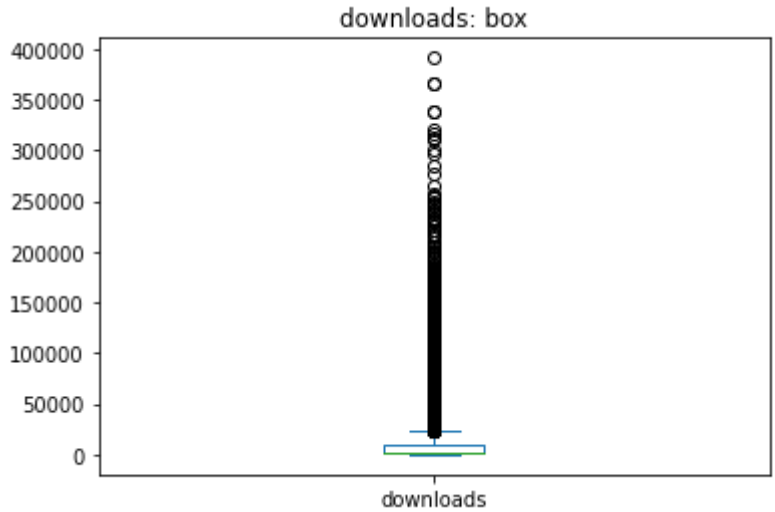

```
posted_date
count   9902.00
mean    2016.27
std        5.87
min     1970.00
25%     2013.00
50%     2017.00
75%     2021.00
max     2023.00
Name: posted_date, dtype: float64
```



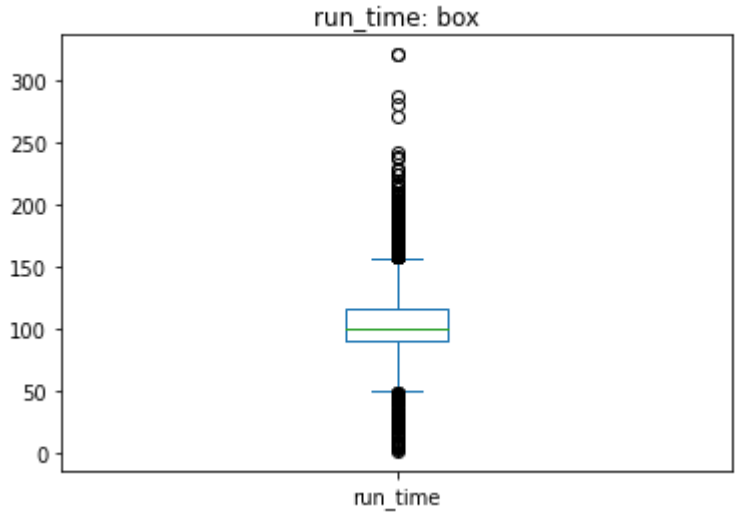
```
imdb_rating
count   9902.00
mean       5.88
std        1.20
min        1.10
25%        5.20
50%        6.00
75%        6.67
max        9.30
Name: imdb_rating, dtype: float64
```



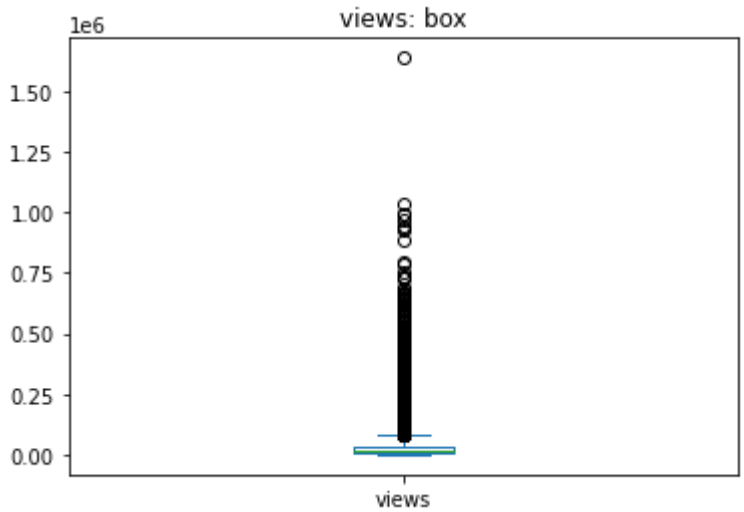

```
downloads
count     9902.00
mean     16154.57
std      31018.63
min          0.00
25%       1340.25
50%       4473.00
75%      16538.50
max     391272.00
Name: downloads, dtype: float64
```



```
run_time
count   9902.00
mean     106.91
std       22.73
min       21.00
25%       91.00
50%      101.00
75%      116.00
max      321.00
Name: run_time, dtype: float64
```




```
views
count      9902.00
mean      49980.30
std       81018.37
min        1002.00
25%       10234.50
50%       22258.00
75%       54012.25
max     1638533.00
Name: views, dtype: float64
```



最高频率值填补

In [77]:
```
attrs = nominals + numerics
new_data = data.copy(deep=True)
for attr in attrs:
    most = data[attr].value_counts().index[0]
```

```
    new_data[attr] = data[attr].fillna(most)
new_data
```

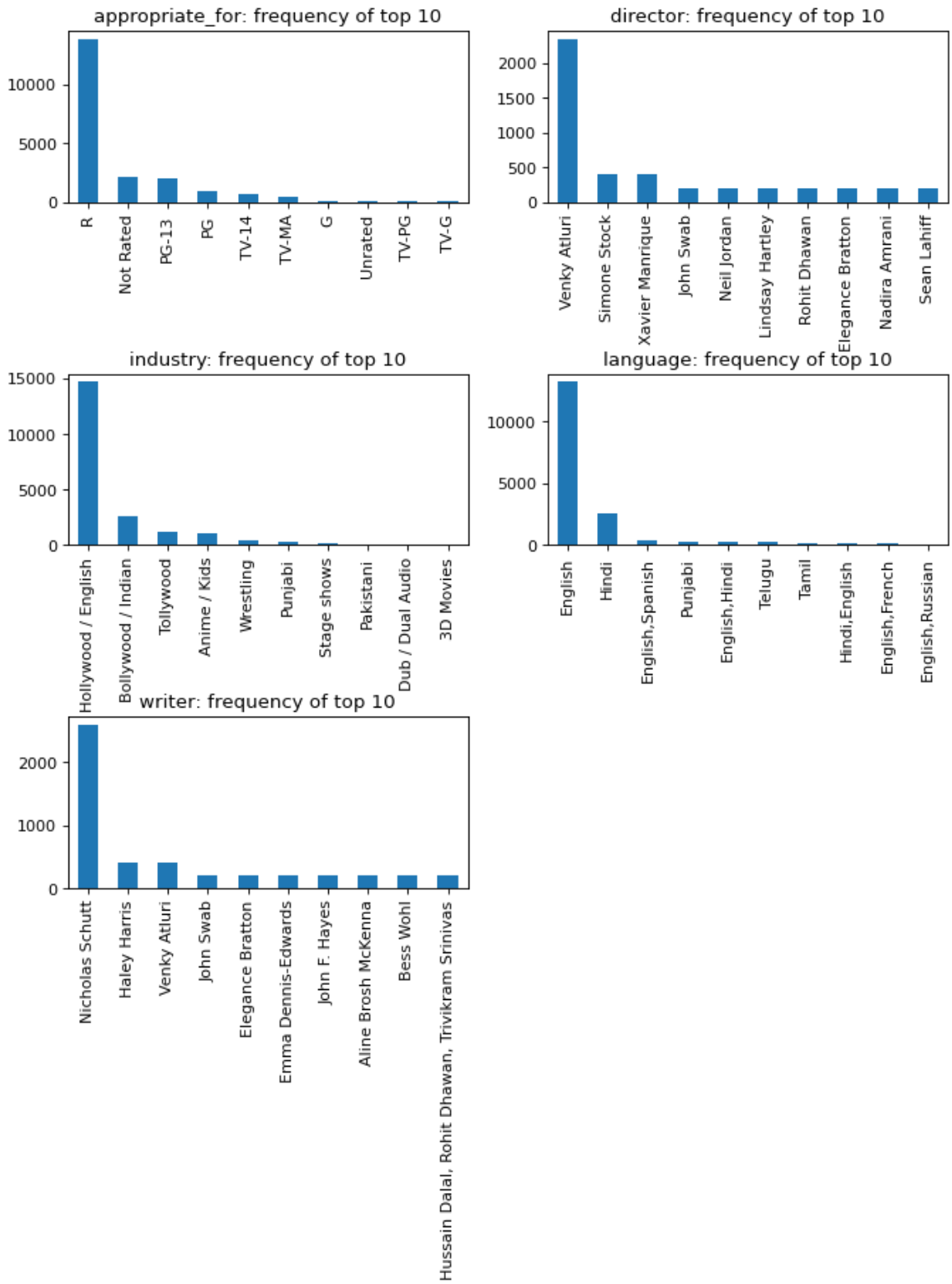
Out[77]:

	imdb_rating	appropriate_for	director	downloads	id	industry	langua
0	4.80	R	John Swab	304.00	372092	Hollywood / English	Eng
1	6.40	TV-PG	Paul Ziller	73.00	372091	Hollywood / English	Eng
2	5.20	R	Ben Wheatley	1427.00	343381	Hollywood / English	English,Hi
3	8.10	R	Venky Atluri	1549.00	372090	Tollywood	Hi
4	4.60	R	Shaji Kailas	657.00	372089	Tollywood	Hi
...	...	...	...	...	...	...	
20543	6.60	R	Venky Atluri	1998.00	28957	Bollywood / Indian	Hi
20544	7.70	R	Bimal Roy	6080.00	28958	Bollywood / Indian	Hi
20545	8.00	R	Venky Atluri	3276.00	30459	Bollywood / Indian	Hi
20546	6.60	R	Venky Atluri	309.00	371669	Wrestling	Eng
20547	6.60	R	Venky Atluri	2613.00	371816	Wrestling	Eng

20548 rows × 16 columns

标称属性变化

In [78]:
```
index = 1
plt.figure(figsize=(10,10), dpi=80).subplots_adjust(hspace=1)
plt.figure(1)
col = 2
row = int(len(nominals) / col) + 1
for attr in nominals:
    plt.subplot(row, col, index)
    index += 1
    freq = 10
    new_data[attr].value_counts().head(freq).plot.bar()
    plt.title(f'{attr}: frequency of top {freq}')
```

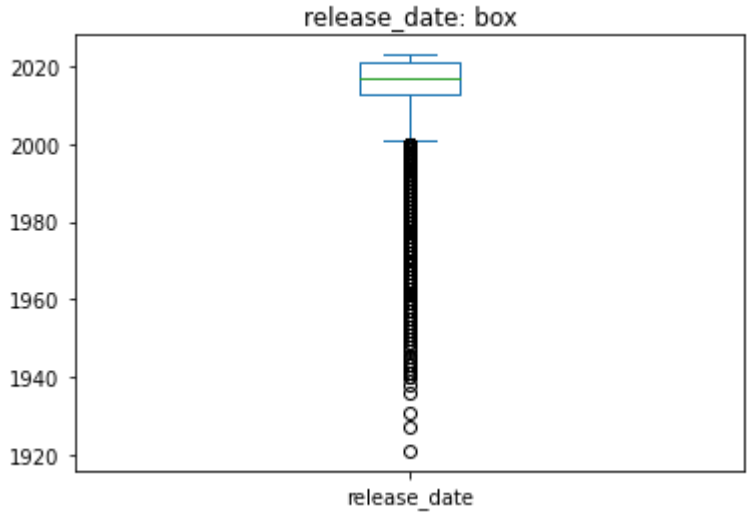


数值属性变化

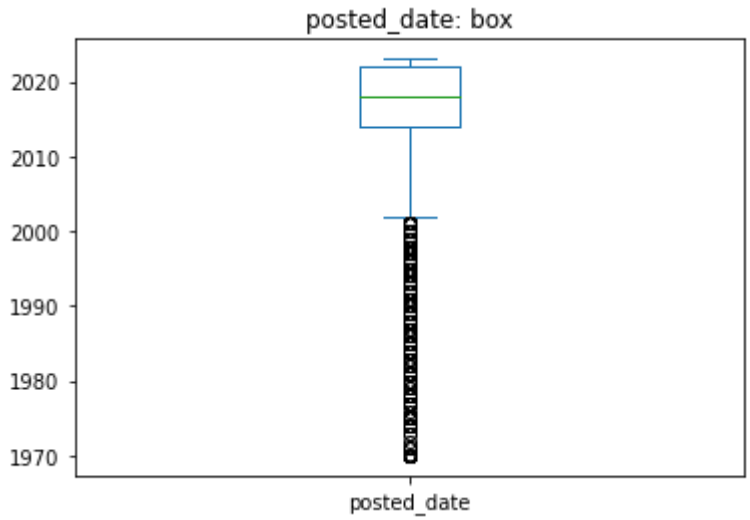
In [79]:
```
attrs = ['release_date', 'posted_date', 'imdb_rating', 'downloads', 'run_
for attr in attrs:
  print(attr)
  try:
    print(new_data[attr].dt.year.describe())
    visit = pd.DataFrame(data[attr].dt.year)
  except:
    print(new_data[attr].describe())
    visit = pd.DataFrame(data[attr])

  visit.plot.box()
```

```
  plt.title(attr + ': box')
  # plt.show()
```

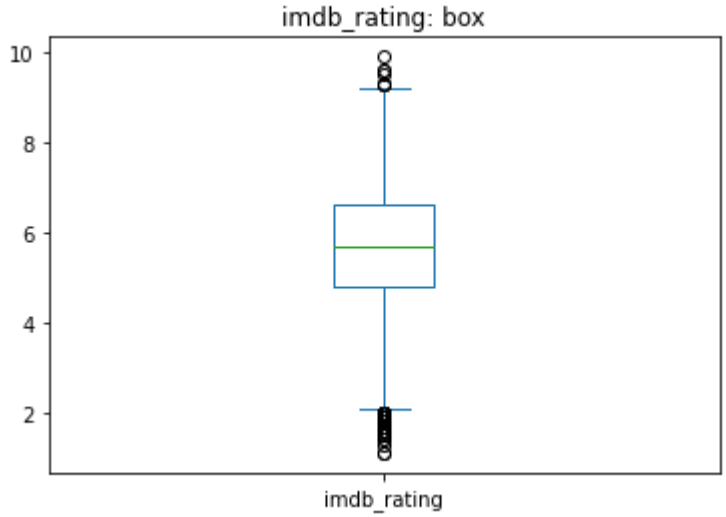
```
release_date
count   20548.00
mean     2013.69
std        12.77
min      1921.00
25%      2013.00
50%      2017.00
75%      2021.00
max      2023.00
Name: release_date, dtype: float64
```



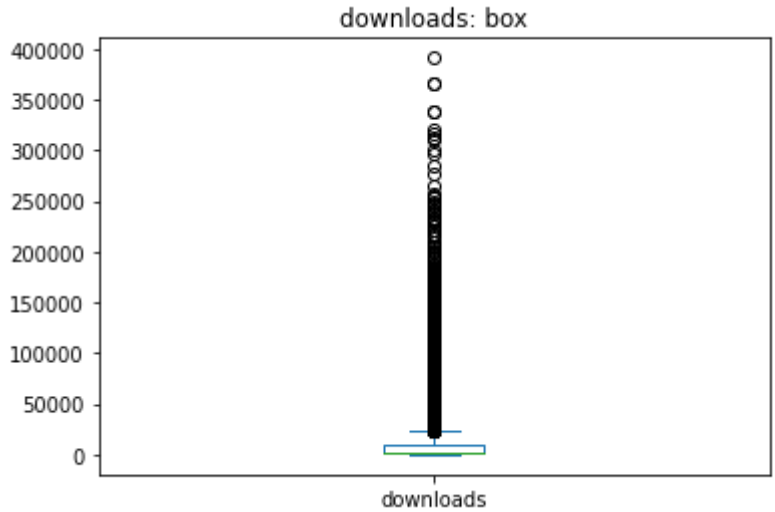

```
posted_date
count   20548.00
mean     2017.00
std         6.10
min      1970.00
25%      2014.00
50%      2018.00
75%      2022.00
max      2023.00
Name: posted_date, dtype: float64
```



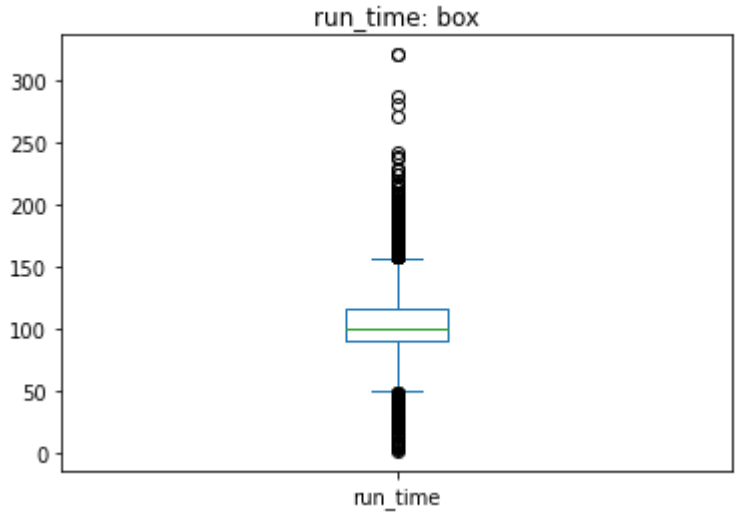
```
imdb_rating
count   20548.00
mean        5.80
std         1.36
min         1.10
25%         4.90
50%         5.80
75%         6.60
max         9.90
Name: imdb_rating, dtype: float64
```



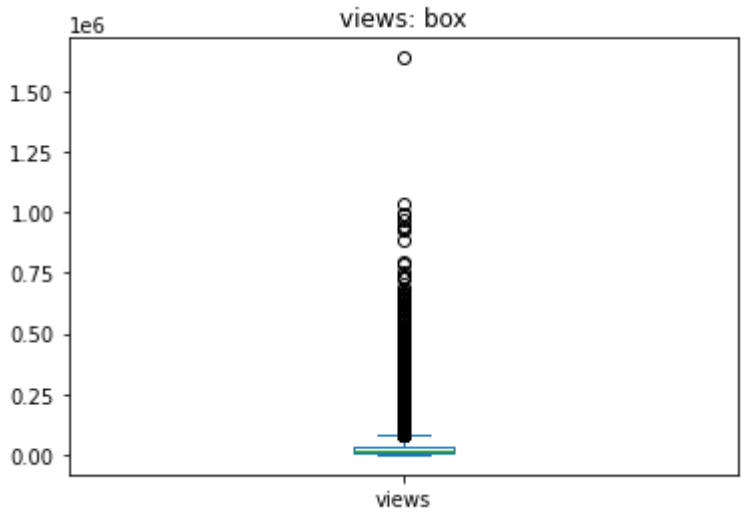

```
downloads
count    20548.00
mean     10794.72
std      23715.72
min          0.00
25%        854.75
50%       2716.00
75%      10069.50
max     391272.00
Name: downloads, dtype: float64
```



```
run_time
count   20548.00
mean      104.80
std        23.05
min         2.00
25%        90.00
50%        98.00
75%       114.00
max       321.00
Name: run_time, dtype: float64
```




```
views
count     20548.00
mean      35594.08
std       62471.24
min         667.00
25%        7571.00
50%       15221.50
75%       36569.50
max     1638533.00
Name: views, dtype: float64
```



相关关系填补

In [80]:
```
new_data = data.copy(deep=True)
corr_matrix = new_data.corr()
corr_matrix
```

Out[80]:

	imdb_rating	downloads	run_time	views
imdb_rating	1.00	0.08	0.33	0.07
downloads	0.08	1.00	0.35	0.95
run_time	0.33	0.35	1.00	0.33
views	0.07	0.95	0.33	1.00

虽然downloads与view呈现高相关性 但数据集中缺失downloads的数据也同时缺失views(仅一条 index 149)

此外imdb_rating的缺失都伴随着run_time download的缺失 同时views数据与其不存在明显的关系 因此无法利用此方法填补缺失值

对于run_time数据缺失使用downloads预测

使用随机森林算法

In [81]:
```
from sklearn.ensemble import RandomForestRegressor
data_map = new_data[['run_time', 'downloads']].dropna()
rfr_1 = RandomForestRegressor(random_state=0, n_estimators=200,  n_jobs=-
matrix = data_map.values
X = matrix[:, 0].reshape(-1,1)
y = matrix[:, 1]
rfr_1.fit(X, y)
data_map = new_data[['run_time', 'downloads']].dropna(subset=['downloads'
X = data_map[data_map.run_time.isnull()].values[:, 1].reshape(-1, 1)
prediction = rfr_1.predict(X)
new_data.loc[(new_data['run_time'].isna() & new_data['downloads'].notna()

new_data
```

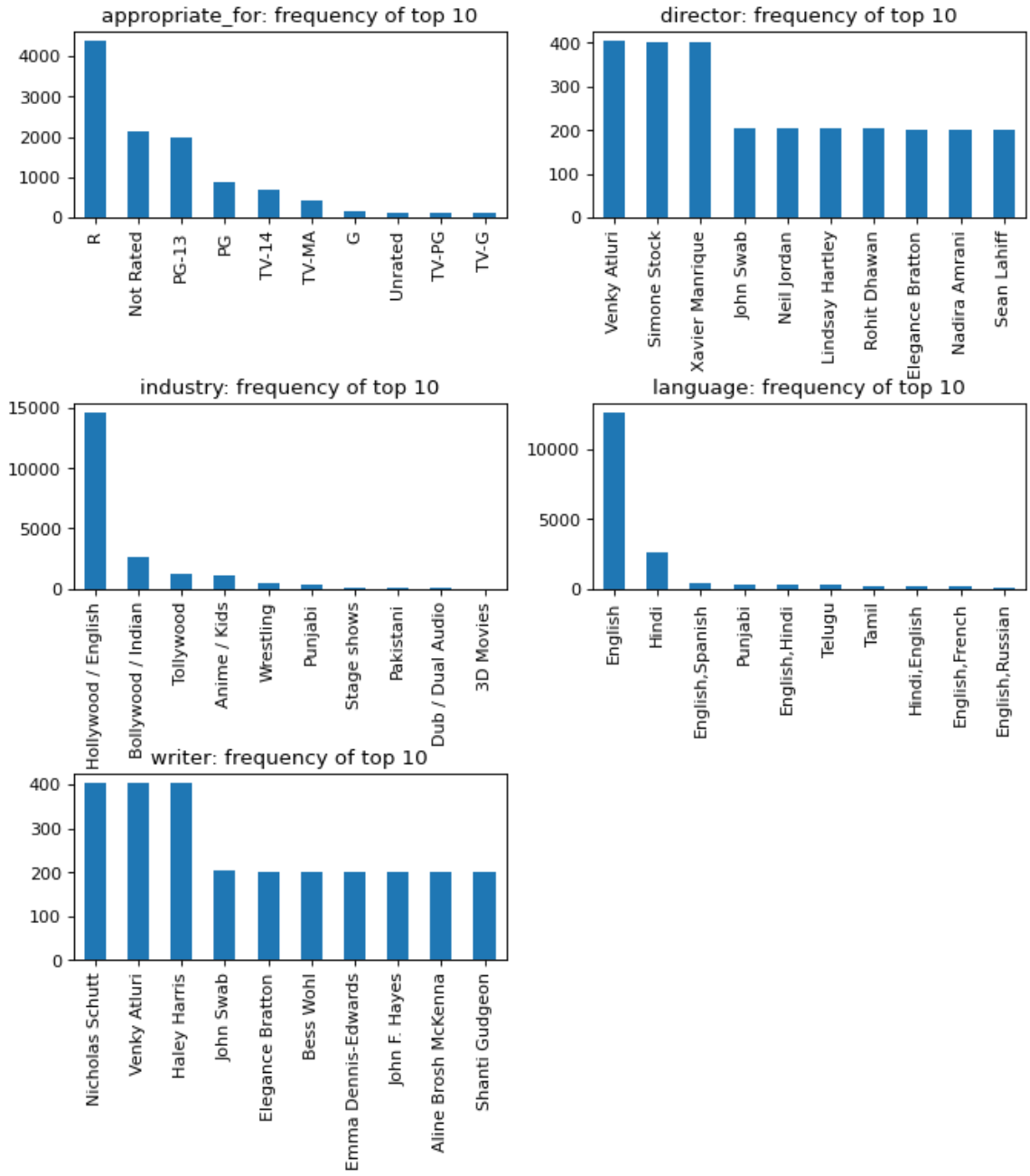
Out[81]:

	imdb_rating	appropriate_for	director	downloads	id	industry	langua
0	4.80	R	John Swab	304.00	372092	Hollywood / English	Eng
1	6.40	TV-PG	Paul Ziller	73.00	372091	Hollywood / English	Eng
2	5.20	R	Ben Wheatley	1427.00	343381	Hollywood / English	English,Hi
3	8.10	NaN	Venky Atluri	1549.00	372090	Tollywood	Hi
4	4.60	NaN	Shaji Kailas	657.00	372089	Tollywood	Hi
...	...	...	...	...	...	...	
20543	NaN	NaN	NaN	1998.00	28957	Bollywood / Indian	Hi
20544	7.70	NaN	Bimal Roy	6080.00	28958	Bollywood / Indian	Hi
20545	8.00	NaN	NaN	3276.00	30459	Bollywood / Indian	Hi
20546	NaN	NaN	NaN	309.00	371669	Wrestling	Eng
20547	NaN	NaN	NaN	2613.00	371816	Wrestling	Eng

20548 rows × 16 columns

标称属性变化

In [82]:
```
index = 1
plt.figure(figsize=(10,10), dpi=80).subplots_adjust(hspace=1)
plt.figure(1)
col = 2
row = int(len(nominals) / col) + 1
for attr in nominals:
    plt.subplot(row, col, index)
    index += 1
    freq = 10
    new_data[attr].value_counts().head(freq).plot.bar()
    plt.title(f'{attr}: frequency of top {freq}')
```

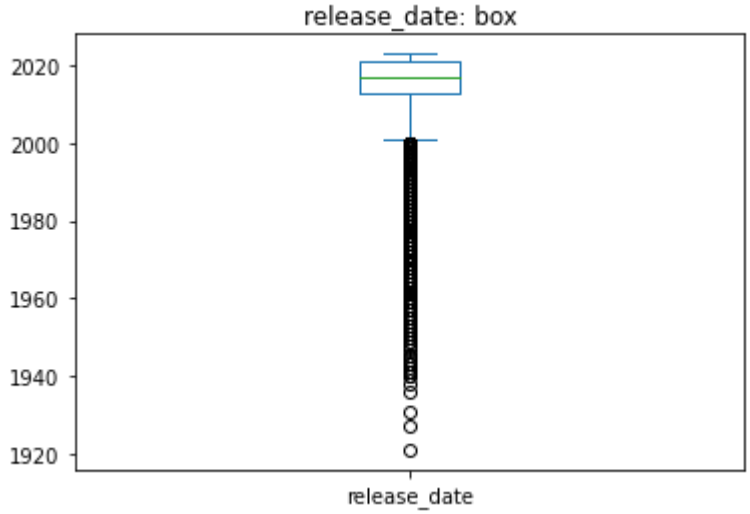


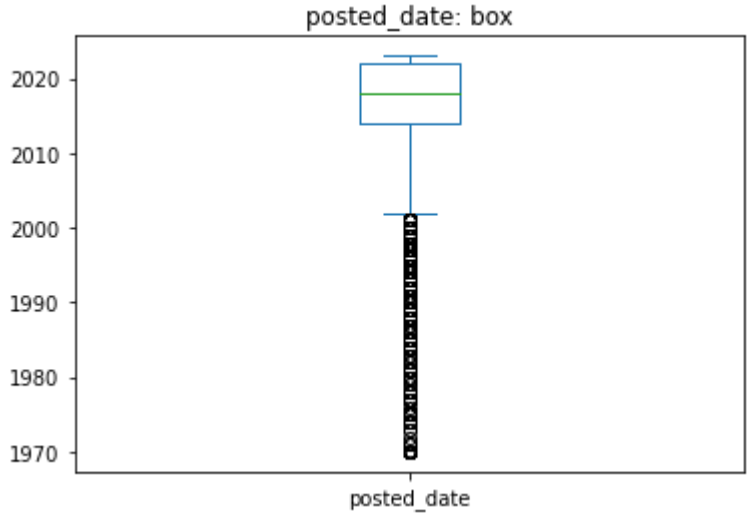
数值属性变化

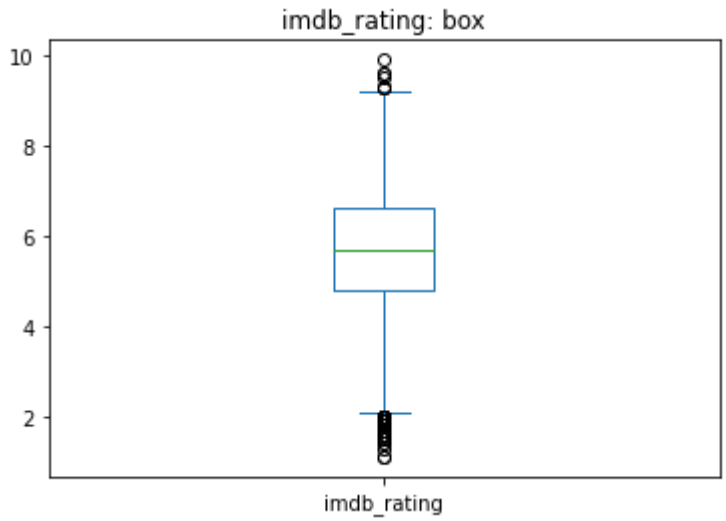
In [89]:
```
attrs = ['release_date', 'posted_date', 'imdb_rating', 'downloads', 'run_
for attr in attrs:
  print(attr)
  try:
    print(new_data[attr].dt.year.describe())
```

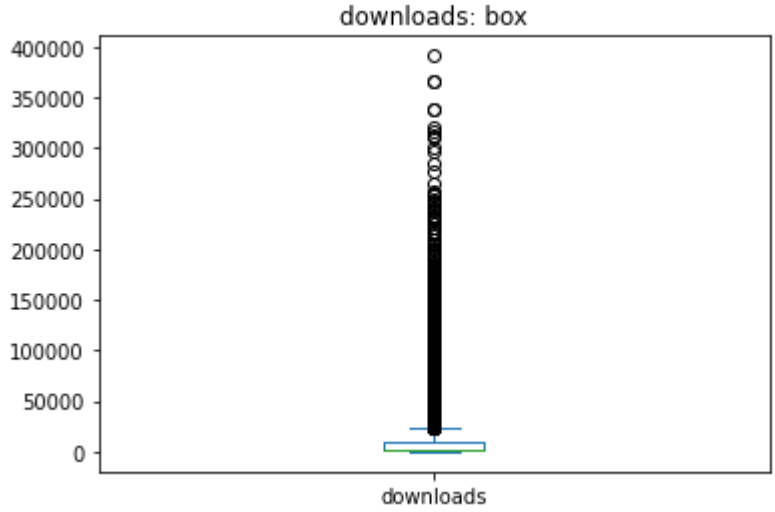
```
    visit = pd.DataFrame(data[attr].dt.year)
  except:
    print(new_data[attr].describe())
    visit = pd.DataFrame(data[attr])

  visit.plot.box()
  plt.title(attr + ': box')
  # plt.show()
```

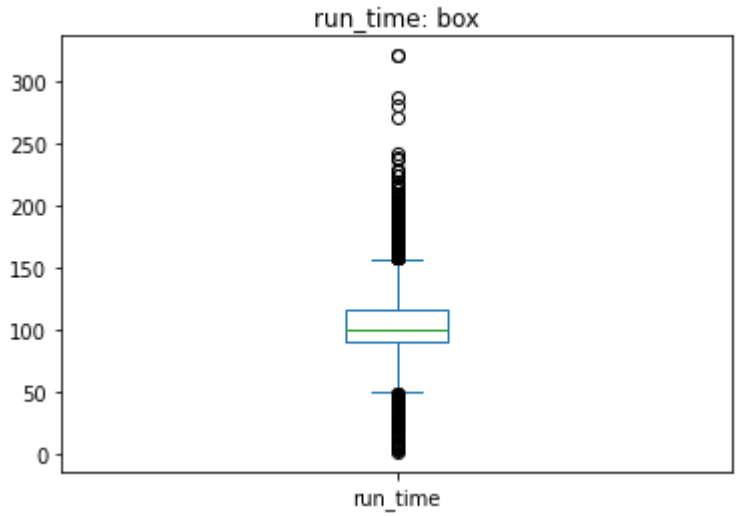
```
release_date
count   20547.00
mean     2013.70
std        12.77
min      1921.00
25%      2013.00
50%      2017.00
75%      2021.00
max      2023.00
Name: release_date, dtype: float64
posted_date
count   20547.00
mean     2017.00
std         6.10
min      1970.00
25%      2014.00
50%      2018.00
75%      2022.00
max      2023.00
Name: posted_date, dtype: float64
imdb_rating
count   20548.00
mean        5.76
std         1.35
min         1.10
25%         4.90
50%         5.76
75%         6.60
max         9.90
Name: imdb_rating, dtype: float64
downloads
count    20548.00
mean     10795.23
std      23715.60
min          0.00
25%        855.75
50%       2716.00
75%      10073.25
max     391272.00
Name: downloads, dtype: float64
run_time
count   20548.00
mean      106.14
std        22.92
min         2.00
25%        90.00
50%       101.00
75%       116.00
max       321.00
Name: run_time, dtype: float64
views
count     20548.00
mean      35595.51
std       62470.90
min         667.00
25%        7571.75
50%       15222.50
75%       36569.50
max     1638533.00
Name: views, dtype: float64
```

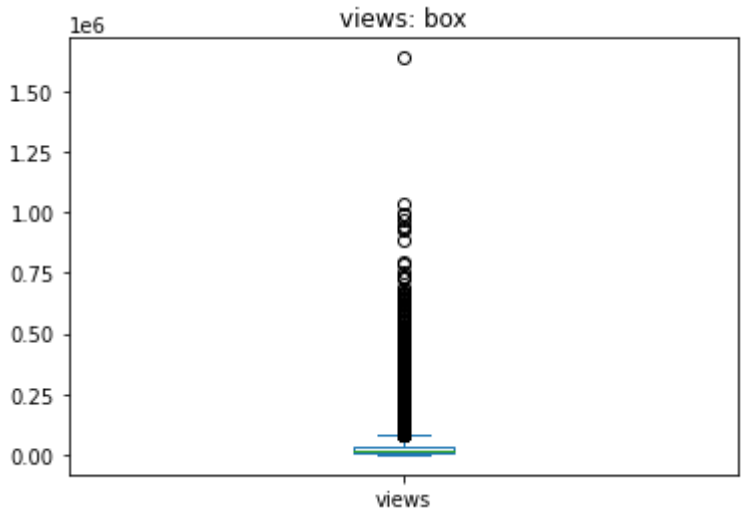

release_date: box
2020
2000
1980
1960
1940
1920
release_date


posted_date: box
2020
2010
2000
1990
1980
1970
posted_date


imdb_rating: box
10
8
6
4
2
imdb_rating







基于相似性

利用impyute工具 对几个数值属性进行填补

In [84]:
```
# pip install impyute
```

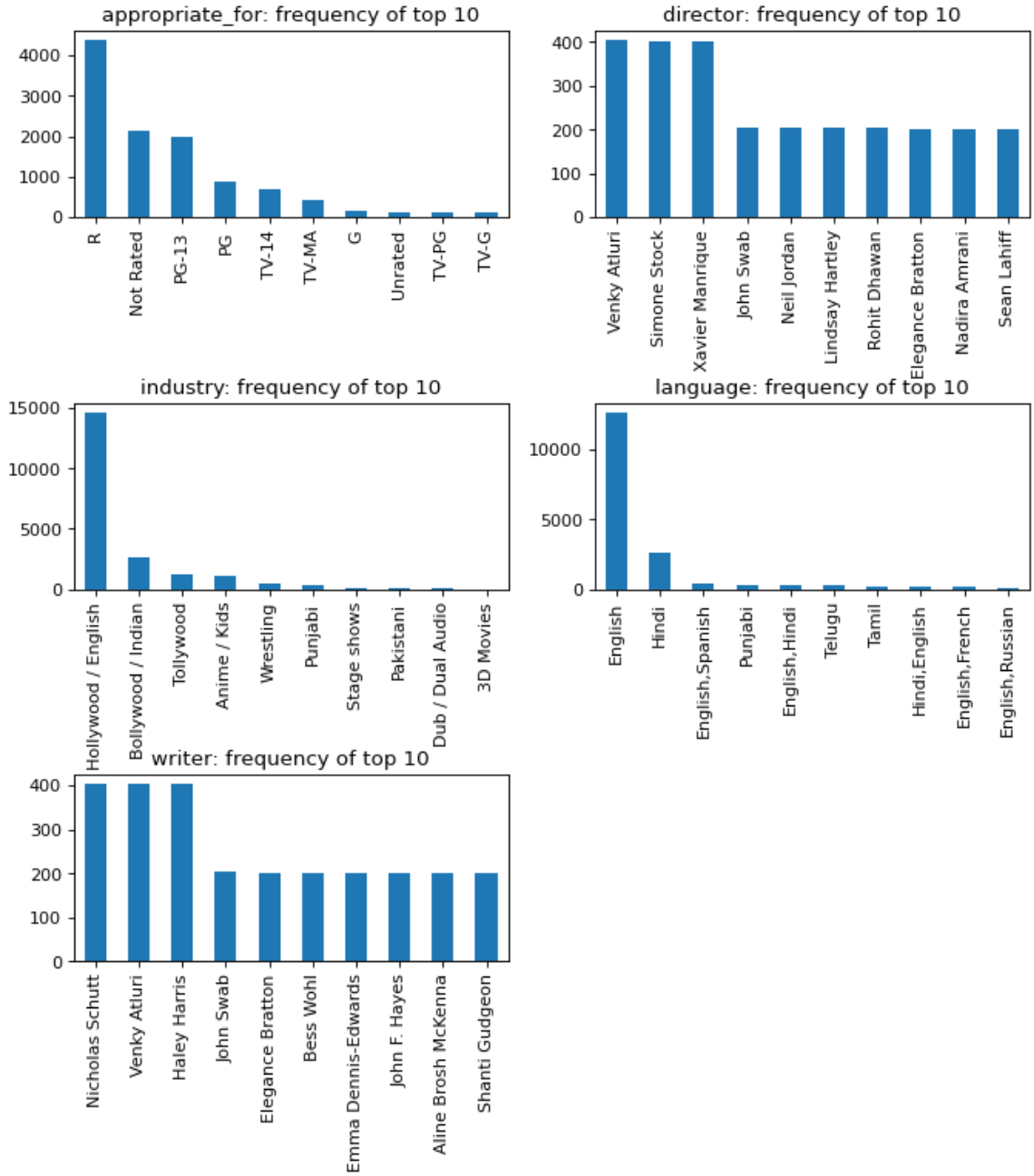
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: impyute in /usr/local/lib/python3.9/dist-packages (0.0.8)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (from impyute) (1.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from impyute) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from impyute) (1.10.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn->impyute) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn->impyute) (1.1.1)
```

In [85]:
```
from impyute import fast_knn
features = ['imdb_rating', 'views', 'downloads', 'run_time']
new_data = data.copy(True)
new_data[features] = pd.DataFrame(fast_knn(np.array(new_data[features]),
new_data.isnull().any()
```

Out[85]:
```
imdb_rating         False
appropriate_for      True
director             True
downloads           False
id                  False
industry             True
language             True
posted_date          True
release_date         True
run_time            False
storyline            True
title                True
views               False
writer               True
old_posted_date      True
old_release_date     True
dtype: bool
```

标称属性变化

In [86]:
```
index = 1
plt.figure(figsize=(10,10), dpi=80).subplots_adjust(hspace=1)
plt.figure(1)
col = 2
row = int(len(nominals) / col) + 1
for attr in nominals:
    plt.subplot(row, col, index)
    index += 1
    freq = 10
    new_data[attr].value_counts().head(freq).plot.bar()
    plt.title(f'{attr}: frequency of top {freq}')
```

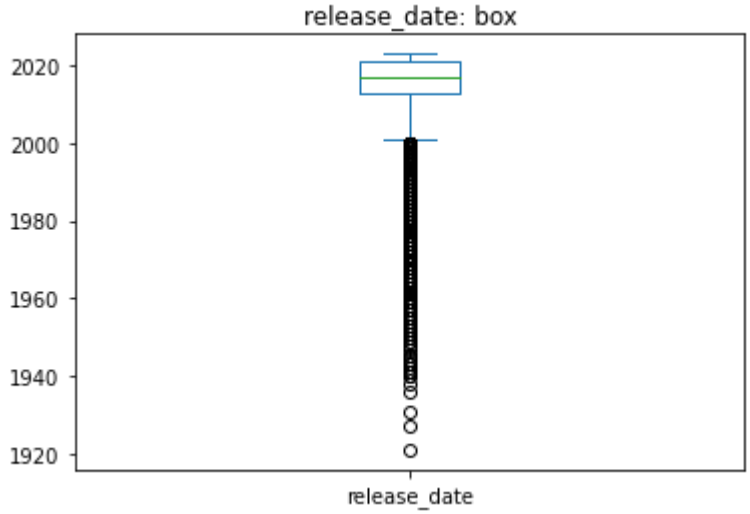


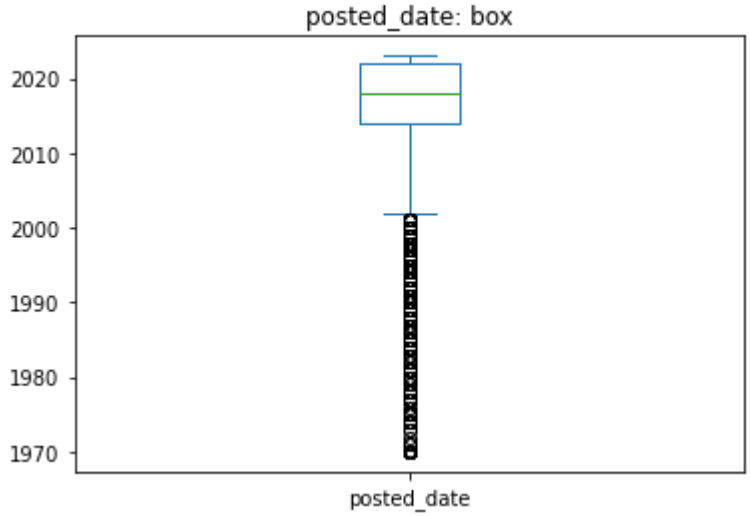
数值属性变化

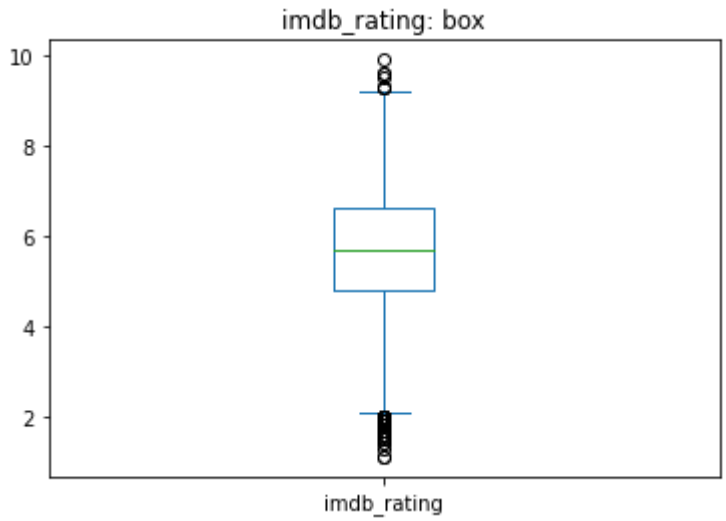
In [88]:

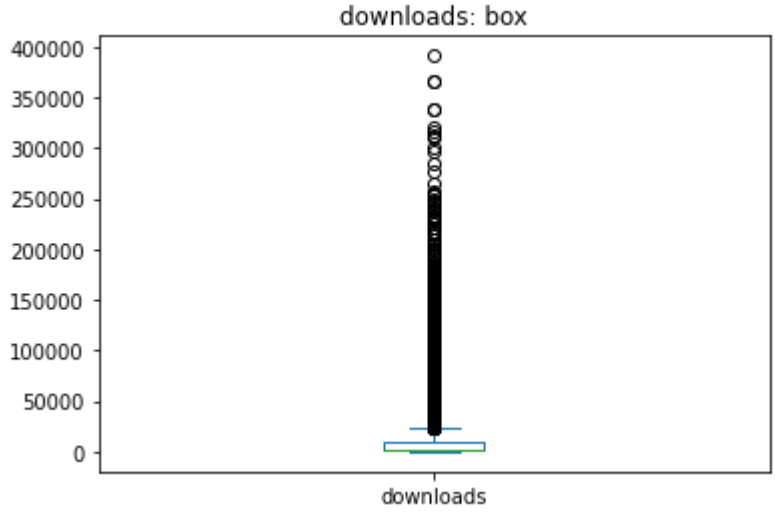
```
attrs = ['release_date', 'posted_date', 'imdb_rating', 'downloads', 'run_
for attr in attrs:
  print(attr)
  try:
    print(new_data[attr].dt.year.describe())
    visit = pd.DataFrame(data[attr].dt.year)
  except:
    print(new_data[attr].describe())
    visit = pd.DataFrame(data[attr])

  visit.plot.box()
  plt.title(attr + ': box')
  # plt.show()
```

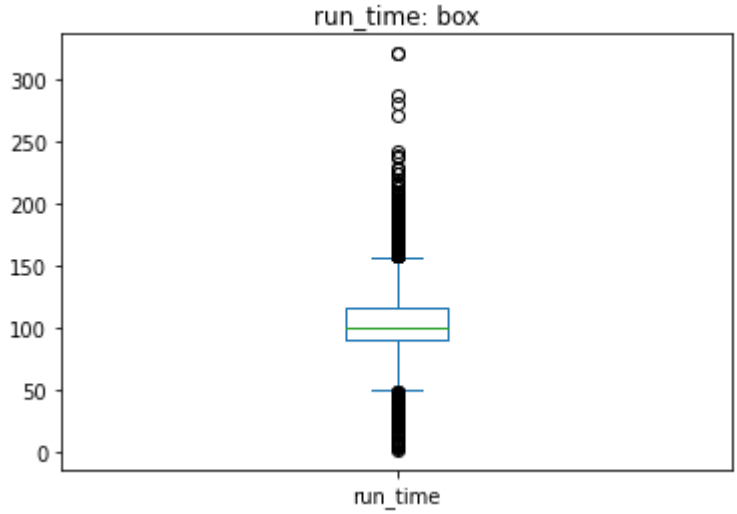
```
release_date
count   20547.00
mean     2013.70
std        12.77
min      1921.00
25%      2013.00
50%      2017.00
75%      2021.00
max      2023.00
Name: release_date, dtype: float64
posted_date
count   20547.00
mean     2017.00
std         6.10
min      1970.00
25%      2014.00
50%      2018.00
75%      2022.00
max      2023.00
Name: posted_date, dtype: float64
imdb_rating
count   20548.00
mean        5.76
std         1.35
min         1.10
25%         4.90
50%         5.76
75%         6.60
max         9.90
Name: imdb_rating, dtype: float64
downloads
count    20548.00
mean     10795.23
std      23715.60
min          0.00
25%        855.75
50%       2716.00
75%      10073.25
max     391272.00
Name: downloads, dtype: float64
run_time
count   20548.00
mean      106.14
std        22.92
min         2.00
25%        90.00
50%       101.00
75%       116.00
max       321.00
Name: run_time, dtype: float64
views
count     20548.00
mean      35595.51
std       62470.90
min         667.00
25%        7571.75
50%       15222.50
75%       36569.50
max     1638533.00
Name: views, dtype: float64
```

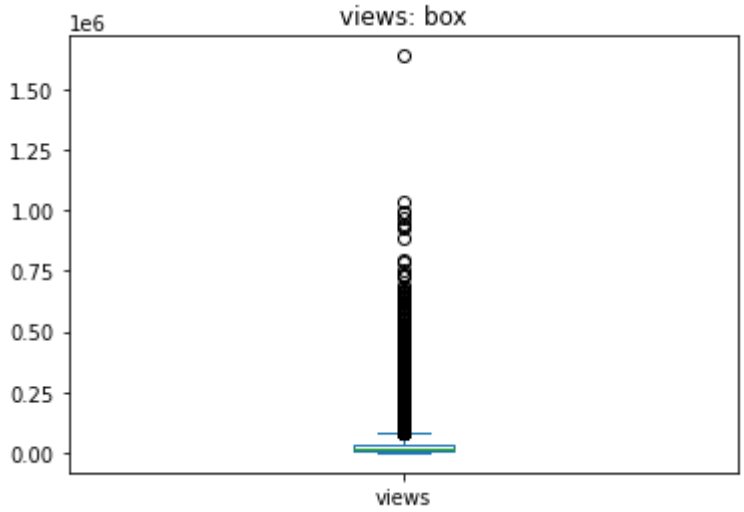

release_date: box
2020
2000
1980
1960
1940
1920
release_date


posted_date: box
2020
2010
2000
1990
1980
1970
posted_date


imdb_rating: box
10
8
6
4
2
imdb_rating


downloads: box
400000
350000
300000
250000
200000
150000
100000
50000
0
downloads


run_time: box
300
250
200
150
100
50
0
run_time


views: box
1e6
1.50
1.25
1.00
0.75
0.50
0.25
0.00
views