Project_04

October 12, 2021

0.1 Survey of the cinema destribution data

Data is provided by ministry of culture.

Based on the provided data it's required to analyze the market of the movies in the cinema theaters and determine the trends.

Additionaly it's required to analyze the movies with goverment support, and answer the questions.

0.1.1 Step 1. Data load and merge of datasets.

[1]:
```
import pandas as pd
import pylab as pl
```

[2]:
```
# data loading
try:
    df_movies = pd.read_csv('mkrf_movies.csv')
    df_shows = pd.read_csv('mkrf_shows.csv')
except:
    df_movies = pd.read_csv('/datasets/mkrf_movies.csv')
    df_shows = pd.read_csv('/datasets/mkrf_shows.csv')
```

[3]:
```
# display of first five rows of datasets
df_movies.head()
```

[3]:
```
                  title   puNumber           show_start_date             type  \
0                221048915  2015-11-27T12:00:00.000Z
1               111013716  2016-09-13T12:00:00.000Z
2                  221038416  2016-10-10T12:00:00.000Z
3                  221026916  2016-06-10T12:00:00.000Z
4                  221030815  2015-07-29T12:00:00.000Z

                                        film_studio production_country  \
0           ,          ,       …
1                                  "    "
2            ,           ,     …
3            ,           ,     …
4            ,           ,     …
```

```
            director                                                producer  \
0                                 ,         ,
1          .                                                        NaN
2                    ,         ,     .   ,
3                    ,         ,     .   ,
4                    ,         ,     .   ,

                 age_restriction  refundable_support  nonrefundable_support  \
0    «18+» -                            NaN                    NaN
1  «6+» -            6                  NaN                    NaN
2    «18+» -                            NaN                    NaN
3    «18+» -                            NaN                    NaN
4    «18+» -                            NaN                    NaN

   budget financing_source ratings                    genres
0     NaN              NaN     7.2        ,   ,
1     NaN              NaN     6.6                ,
2     NaN              NaN     6.8        ,   ,
3     NaN              NaN     6.8        ,   ,
4     NaN              NaN     6.8        ,   ,
```

[4]:
```
df_shows.head()
```

[4]:
```
    puNumber    box_office
0  111000113  2.450000e+03
1  111000115  6.104000e+04
2  111000116  1.530300e+08
3  111000117  1.226096e+07
4  111000118  1.636841e+08
```

[5]:
```
# display short information on datasets
df_shows.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3158 entries, 0 to 3157
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   puNumber    3158 non-null   int64
 1   box_office  3158 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 49.5 KB
```

[6]:
```
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7486 entries, 0 to 7485
Data columns (total 15 columns):
```

```
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   title                  7486 non-null   object
 1   puNumber               7486 non-null   object
 2   show_start_date        7486 non-null   object
 3   type                   7486 non-null   object
 4   film_studio            7468 non-null   object
 5   production_country     7484 non-null   object
 6   director               7477 non-null   object
 7   producer               6918 non-null   object
 8   age_restriction        7486 non-null   object
 9   refundable_support     332 non-null    float64
 10  nonrefundable_support  332 non-null    float64
 11  budget                 332 non-null    float64
 12  financing_source       332 non-null    object
 13  ratings                6519 non-null   object
 14  genres                 6510 non-null   object
dtypes: float64(3), object(12)
memory usage: 877.4+ KB
```

[7]:
```
# changing of datatype of "puNumber" column
df_shows['puNumber'] = df_shows['puNumber'].astype('str')
df_shows.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3158 entries, 0 to 3157
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   puNumber    3158 non-null   object
 1   box_office  3158 non-null   float64
dtypes: float64(1), object(1)
memory usage: 49.5+ KB
```

[8]:
```
# df merge
df_movies = df_movies.merge(df_shows,on ='puNumber',how='left')
```

[9]:
```
# display of the results
df_movies.head(20)
```

[9]:
```
                                                title   puNumber  \
0                                                      221048915
1                                                     111013716
2                                                        221038416
3                                                        221026916
4                                                        221030815
5                                                      111013816
6                                                       111007013
```

```
7                                                 221074614
8                                               .  121011416
9                                                  111019114
10              /                 …  221031416
11                                                   111019014
12              /               . . …  221011415
13                                                121003106
14                /                    /  221008812
15                                                    111014916
16                                                   111007513
17                                                       111027914
18                                                   111003616
19                                                    111003716

             show_start_date            type  \
0   2015-11-27T12:00:00.000Z
1   2016-09-13T12:00:00.000Z
2   2016-10-10T12:00:00.000Z
3   2016-06-10T12:00:00.000Z
4   2015-07-29T12:00:00.000Z
5   2016-09-13T12:00:00.000Z
6   2013-10-18T12:00:00.000Z
7   2014-12-29T12:00:00.000Z
8   2016-05-05T12:00:00.000Z
9   2014-12-01T12:00:00.000Z
10  2016-06-29T12:00:00.000Z
11  2014-12-01T12:00:00.000Z
12  2015-04-03T12:00:00.000Z
13  2013-08-26T12:00:00.000Z
14  2012-01-27T12:00:00.000Z
15  2016-09-13T12:00:00.000Z
16  2013-10-18T12:00:00.000Z
17  2014-12-24T12:00:00.000Z
18  2016-02-12T12:00:00.000Z
19  2016-02-12T12:00:00.000Z

                                          film_studio  \
0             ,          ,       …
1                                          "     "
2              ,           ,     …
3              ,           ,     …
4              ,           ,     …
5                                          "     "
6                                          "     "
7             ,           ,       …
8                               ,  1
9                    "    ",
```

10 / , , …
11 " "
12 , , …
13 , , …
14 , , …
15 " "
16 " "
17 " "
18 " "
19 " "

production_country director \
0
1 .
2
3
4
5 .
6 .
7
8
9 , .
10
11 .
12
13 -
14 -
15 .
16 .
17 .
18 .
19 . , .

producer \
0 , ,
1 NaN
2 , , . ,
3 , , . ,
4 , , . ,
5 NaN
6 NaN
7 ,
8
9 NaN
10 , ,
11 NaN
12 , , . , …

13
14 , ,
15 NaN
16 NaN
17 NaN
18 NaN
19 NaN

	age_restriction	refundable_support	\
0	«18+» -	NaN	
1	«6+» - 6	NaN	
2	«18+» -	NaN	
3	«18+» -	NaN	
4	«18+» -	NaN	
5	«6+» - 6	NaN	
6	«12+» - 12	NaN	
7	«18+» -	NaN	
8	«18+» -	NaN	
9	«12+» - 12	NaN	
10	«16+» - 16	NaN	
11	«12+» - 12	NaN	
12	«16+» - 16	NaN	
13	«16+» - 16	NaN	
14	«18+» -	NaN	
15	«12+» - 12	NaN	
16	«12+» - 12	NaN	
17	«12+» - 12	NaN	
18	«6+» - 6	NaN	
19	«6+» - 6	NaN	

	nonrefundable_support	budget	financing_source	ratings	\
0	NaN	NaN	NaN	7.2	
1	NaN	NaN	NaN	6.6	
2	NaN	NaN	NaN	6.8	
3	NaN	NaN	NaN	6.8	
4	NaN	NaN	NaN	6.8	
5	NaN	NaN	NaN	7.7	
6	NaN	NaN	NaN	8.3	
7	NaN	NaN	NaN	6.6	
8	NaN	NaN	NaN	8.0	
9	NaN	NaN	NaN	7.8	
10	NaN	NaN	NaN	7.7	
11	NaN	NaN	NaN	7.8	
12	NaN	NaN	NaN	8.1	
13	NaN	NaN	NaN	7.1	
14	NaN	NaN	NaN	6.0	
15	NaN	NaN	NaN	8.3	

```
16                    NaN     NaN              NaN     8.0
17                    NaN     NaN              NaN     7.4
18                    NaN     NaN              NaN     8.0
19                    NaN     NaN              NaN     7.7

                       genres  box_office
0           ,   ,              NaN
1                   ,             NaN
2          ,    ,             NaN
3          ,    ,             NaN
4          ,    ,             NaN
5                                    NaN
6                  ,          2700.0
7         ,      ,             NaN
8                  ,             NaN
9                  ,            NaN
10         ,     ,             NaN
11        ,    ,               NaN
12                     ,            NaN
13         ,    ,              NaN
14               ,              NaN
15                                   NaN
16        ,      ,             NaN
17                ,             NaN
18                ,           360.0
19                                420.0
```

[10]:
```
# display of info on merged dataset
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7486 entries, 0 to 7485
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   title                  7486 non-null   object
 1   puNumber               7486 non-null   object
 2   show_start_date        7486 non-null   object
 3   type                   7486 non-null   object
 4   film_studio            7468 non-null   object
 5   production_country     7484 non-null   object
 6   director               7477 non-null   object
 7   producer               6918 non-null   object
 8   age_restriction        7486 non-null   object
 9   refundable_support     332 non-null    float64
 10  nonrefundable_support  332 non-null    float64
 11  budget                 332 non-null    float64
 12  financing_source       332 non-null    object
```

```
 13  ratings                 6519 non-null   object
 14  genres                  6510 non-null   object
 15  box_office              3158 non-null   float64
dtypes: float64(4), object(12)
memory usage: 994.2+ KB
```

Conclusion

- Data sucessfully imported and datasets were merged in one dataset.
- Dataset has 7486 rows and 15 columns with object and float datatypes.

0.1.2 Step 2. Data Preparation

Step 2.1. Data types check

[11]:
```
# deletion of % symbol
df_movies['ratings'] = df_movies['ratings'].str.replace('%','')

# function declaration for reformating of rating column
def devide_ten (df_name):
    if df_name['ratings'] > 10:
        return df_name['ratings']/10
    else:
        return df_name['ratings']

# datatype changing and application of function
df_movies['ratings'] = df_movies['ratings'].astype('float')
df_movies['ratings'] = df_movies.apply(devide_ten, axis=1)

# display of unique results
df_movies['ratings'].sort_values().unique()
```

[11]:
```
array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.1, 2.4, 2.5,
       2.6, 2.7, 2.8, 2.9, 3. , 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8,
       3.9, 4. , 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5. , 5.1,
       5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 6. , 6.1, 6.2, 6.3, 6.4,
       6.5, 6.6, 6.7, 6.8, 6.9, 7. , 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7,
       7.8, 7.9, 8. , 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 9. ,
       9.1, 9.2, 9.4, 9.7, 9.8, 9.9, nan])
```

[12]:
```
# datatype change in colunmn show_start_date
df_movies['show_start_date'] = pd.
 ↪to_datetime(df_movies['show_start_date'],format='%Y-%m-%d')

# display the results
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7486 entries, 0 to 7485
Data columns (total 16 columns):
```

```
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   title                  7486 non-null   object
 1   puNumber               7486 non-null   object
 2   show_start_date        7486 non-null   datetime64[ns, UTC]
 3   type                   7486 non-null   object
 4   film_studio            7468 non-null   object
 5   production_country     7484 non-null   object
 6   director               7477 non-null   object
 7   producer               6918 non-null   object
 8   age_restriction        7486 non-null   object
 9   refundable_support     332 non-null    float64
 10  nonrefundable_support  332 non-null    float64
 11  budget                 332 non-null    float64
 12  financing_source       332 non-null    object
 13  ratings                6519 non-null   float64
 14  genres                 6510 non-null   object
 15  box_office             3158 non-null   float64
dtypes: datetime64[ns, UTC](1), float64(5), object(10)
memory usage: 994.2+ KB
```

[13]:
```
# changing of incorrect value of column "puNumber"
df_movies['puNumber'] = df_movies['puNumber'].str.replace('  ','-1')

# changing of column datatype
df_movies['puNumber'] = df_movies['puNumber'].astype('int')

# display the results
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7486 entries, 0 to 7485
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   title                  7486 non-null   object
 1   puNumber               7486 non-null   int32
 2   show_start_date        7486 non-null   datetime64[ns, UTC]
 3   type                   7486 non-null   object
 4   film_studio            7468 non-null   object
 5   production_country     7484 non-null   object
 6   director               7477 non-null   object
 7   producer               6918 non-null   object
 8   age_restriction        7486 non-null   object
 9   refundable_support     332 non-null    float64
 10  nonrefundable_support  332 non-null    float64
 11  budget                 332 non-null    float64
 12  financing_source       332 non-null    object
```

```
 13  ratings                6519 non-null   float64
 14  genres                 6510 non-null   object
 15  box_office             3158 non-null   float64
dtypes: datetime64[ns, UTC](1), float64(5), int32(1), object(9)
memory usage: 965.0+ KB
```

Step 2.2. Nulls processing

[14]:
```
# selection of columns with categorical values
temp =␣
 ↪['film_studio','director','producer','financing_source','production_country','genres']

# loop for fillup on null value
for n in temp:
    df_movies[n] = df_movies[n].fillna('unknown')

# selection of numeric columns
temp =␣
 ↪['refundable_support','nonrefundable_support','budget','ratings','box_office']

# fillup nulls
for n in temp:
    df_movies[n] = df_movies[n].fillna(0)

# display the results
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7486 entries, 0 to 7485
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   title                  7486 non-null   object
 1   puNumber               7486 non-null   int32
 2   show_start_date        7486 non-null   datetime64[ns, UTC]
 3   type                   7486 non-null   object
 4   film_studio            7486 non-null   object
 5   production_country     7486 non-null   object
 6   director               7486 non-null   object
 7   producer               7486 non-null   object
 8   age_restriction        7486 non-null   object
 9   refundable_support     7486 non-null   float64
 10  nonrefundable_support  7486 non-null   float64
 11  budget                 7486 non-null   float64
 12  financing_source       7486 non-null   object
 13  ratings                7486 non-null   float64
 14  genres                 7486 non-null   object
 15  box_office             7486 non-null   float64
dtypes: datetime64[ns, UTC](1), float64(5), int32(1), object(9)
```

```
memory usage: 965.0+ KB
```

All nulls in categorical coulmns were replaced with value “unknown’, numeric - 0

Step 2.3. Duplicates processing

[15]:
```
# count of duplicates
print(df_movies.duplicated().count())

# count of duplicates in column puNumber
print(df_movies['puNumber'].drop_duplicates().count())

# selection of rows with duplicated value in column puNumber
temp = df_movies[df_movies['puNumber'].duplicated()].puNumber

# display of data with duplicates in column puNumber
for n in temp:
    print(n)
    display(df_movies[df_movies['puNumber'] == n])
```

```
7486
7484
221154310
```

```
                                                  title   puNumber  \
4638                                        221154310
4639      -                 /     …  221154310

              show_start_date            type  \
4638 2010-12-17 12:00:00+00:00
4639 2010-12-17 12:00:00+00:00

                                          film_studio  \
4638      ,         ,     1         …
4639            ,             ,    …

                               production_country        director  \
4638
4639       -         -     -

                                       producer  \
4638                            ,
4639         ,     .    ,

                     age_restriction  refundable_support  \
4638  «16+» -            16                   0.0
4639  «16+» -            16                   0.0

      nonrefundable_support  budget financing_source  ratings  \
4638                   0.0     0.0          unknown      7.0
```

```
4639                    0.0     0.0          unknown      7.6

                       genres  box_office
4638              ,             0.0
4639     ,     ,            0.0
```

```
221054410

                title   puNumber           show_start_date            type  \
5067          !  221054410 2010-05-25 12:00:00+00:00
5068          !  221054410 2010-05-25 12:00:00+00:00

                                           film_studio production_country  \
5067                   ,                     -
5068          ,               ,  - …

            director                          producer  \
5067                      ,
5068                      ,

                     age_restriction  refundable_support  \
5067  «16+» -            16                   0.0
5068  «12+» -            12                   0.0

      nonrefundable_support  budget financing_source  ratings  \
5067                    0.0     0.0          unknown      7.4
5068                    0.0     0.0          unknown      6.8

                genres  box_office
5067      ,             0.0
5068      ,             0.0
```

[16]:
```
# display of uniques values of columns director and producer
print(df_movies.director.sort_values().unique())
print(df_movies.producer.sort_values().unique())
```

```
['          ' '  .   ,  .   ' '  .    ' …
 '                          '
 '                ' '                ']
[' .    ' '  .    '
 '         ,        ,           ,         ,
     ,         '
 …
 '          ,       ,          ,        ,       .      ,
            '
 '        ,         ,          '
 '        ,           ']
```

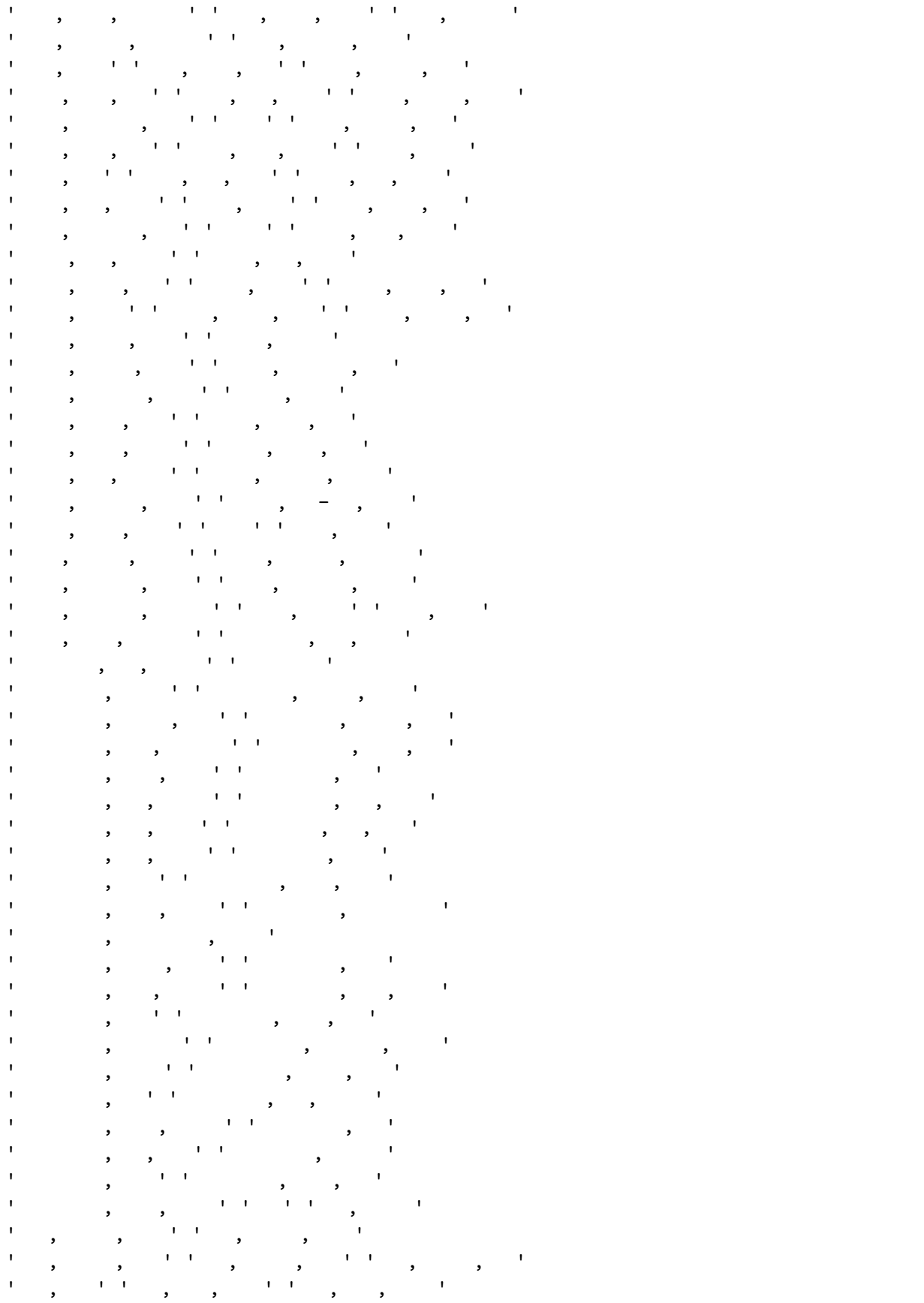
Cocnlusion

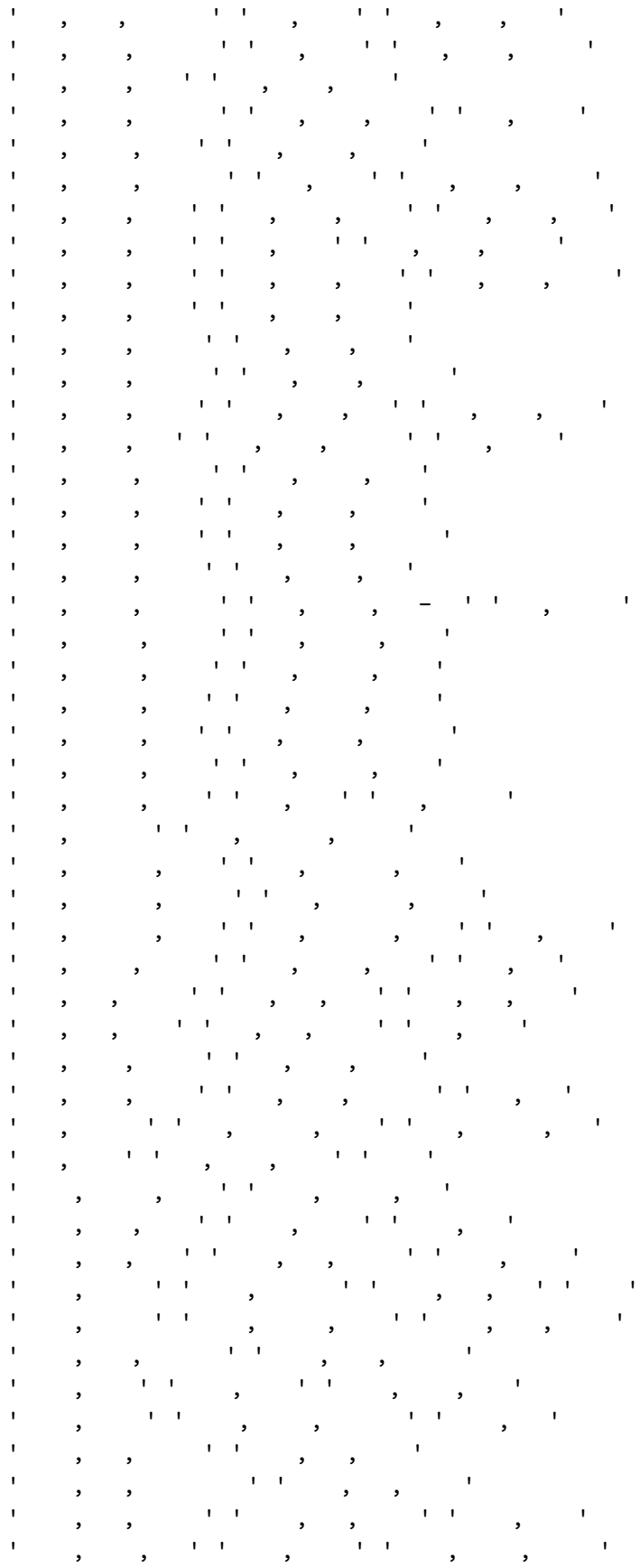
- data set has no obvious duplicates
- there are some duplicates in several columns
- in column puNumber there are few duplicates, hovewer it's not required to delete/merge it. Issue qould be solved by clarification of the data with the person who created the table.
- in columns director and producer there are duplicates - it's normal, due to the fact that more than one movie could be filmed by same director/producer/country. Columns producer and director has impilicit duplicates, but for processing of 7k pcs will take a lot of time, issue also could be solved by the creater of table.

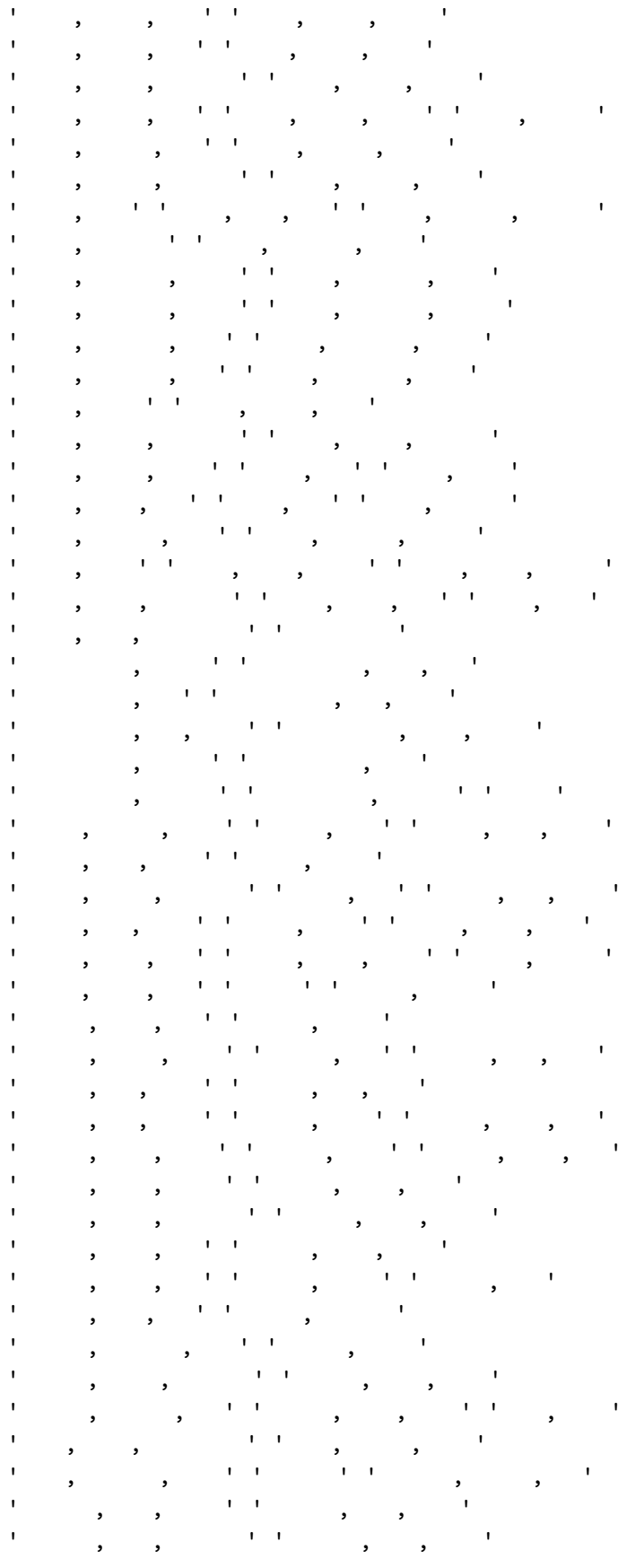
Step 2.4. Categorical columns processing

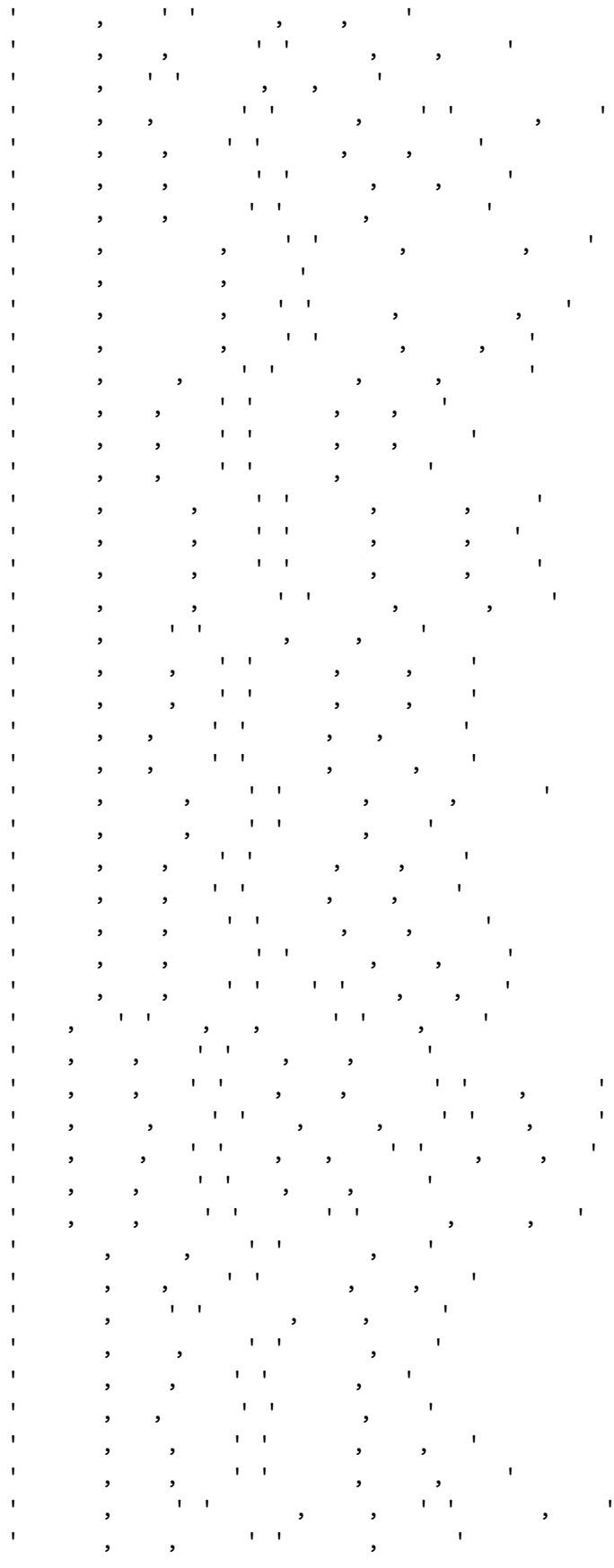
[17]:
```
# diaply of unique values of columns country, director, producer
print(df_movies.genres.sort_values().unique(),'\n\n',
    df_movies.production_country.sort_values().unique(),'\n\n',
    df_movies.age_restriction.sort_values().unique())

# spaces deletion
df_movies.producer = df_movies.producer.str.strip()
df_movies.director = df_movies.director.str.strip()
df_movies.production_country = df_movies.production_country.str.strip()
df_movies.genres = df_movies.genres.str.strip()
```

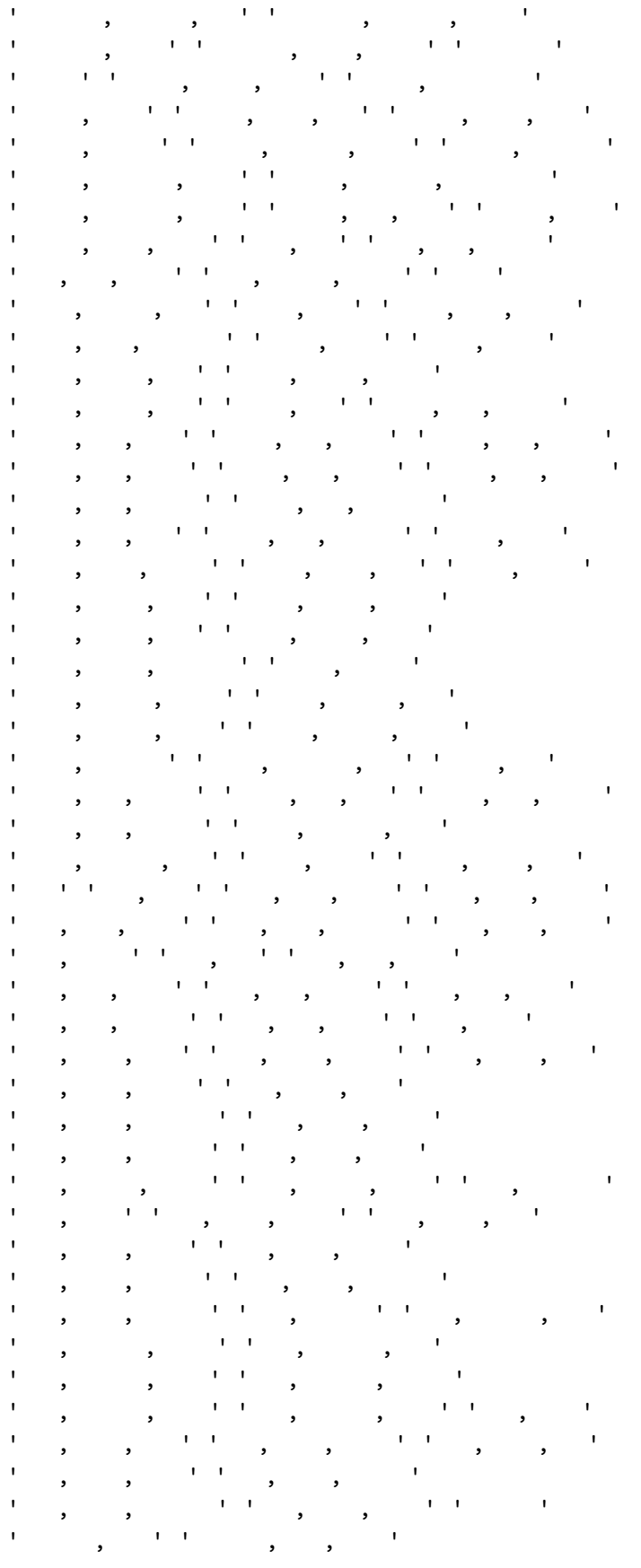
```
['unknown' '   ,      ' '   ,      ,    '
 '   ,      ,     ' '   ,      ,    '
 '   ,      ,     ' '   ,      ,         '
 '   ,      ,      ' '   ,      ,       '
 '   ,      ,    ' '   ,      ,      '
 '   ,      ,     ' '      ' '      ,    ,   '
 '     ,         ' '      ,    ' '     ,   ,     '
 '     ,   ,     ' '      ,    ,     ' '      ,    '
 '     ,    ,    ' '      ,     ,    '
 '     ,     ,    ' '      ,      ,    '
 '     ,    ,    ' '      ,    ,     '
 '     ,   ,     ' '      ,    ,      '
 '     ,   ,     ' '    ' '    ,    ,       '
 '    ,     ' '    ,    ,    ' '    ,    ,    '
 '    ,      ' '    ,    ' '    ,   ,      '
 '    ,   ,     ' '     ,    ,     ' '    ,   ,     '
 '    ,   ,      ' '    ,   ,      ' '    ,   ,      '
 '    ,   ,         ' '     ,   ,       ' '     ,     '
 '    ,    ,    ' '    ,     ' '    ,    ,     '
 '    ,    ,       ' '    ,    ,      ' '    ,     '
 '    ,     ,     ' '    ,     ,    '
 '    ,     ,     ' '     ,          ' '     ,     ,    '
 '    ,        ' '    ,       ,     '
 '    ,       ,     ' '    ,       ,     '
 '    ,       ,     ' '    ,    ,     ' '    ,     '
 '    ,    ,     ' '    ,    ,    ' '    ,    ,    '
 '    ,    ,     ' '    ,     '
```

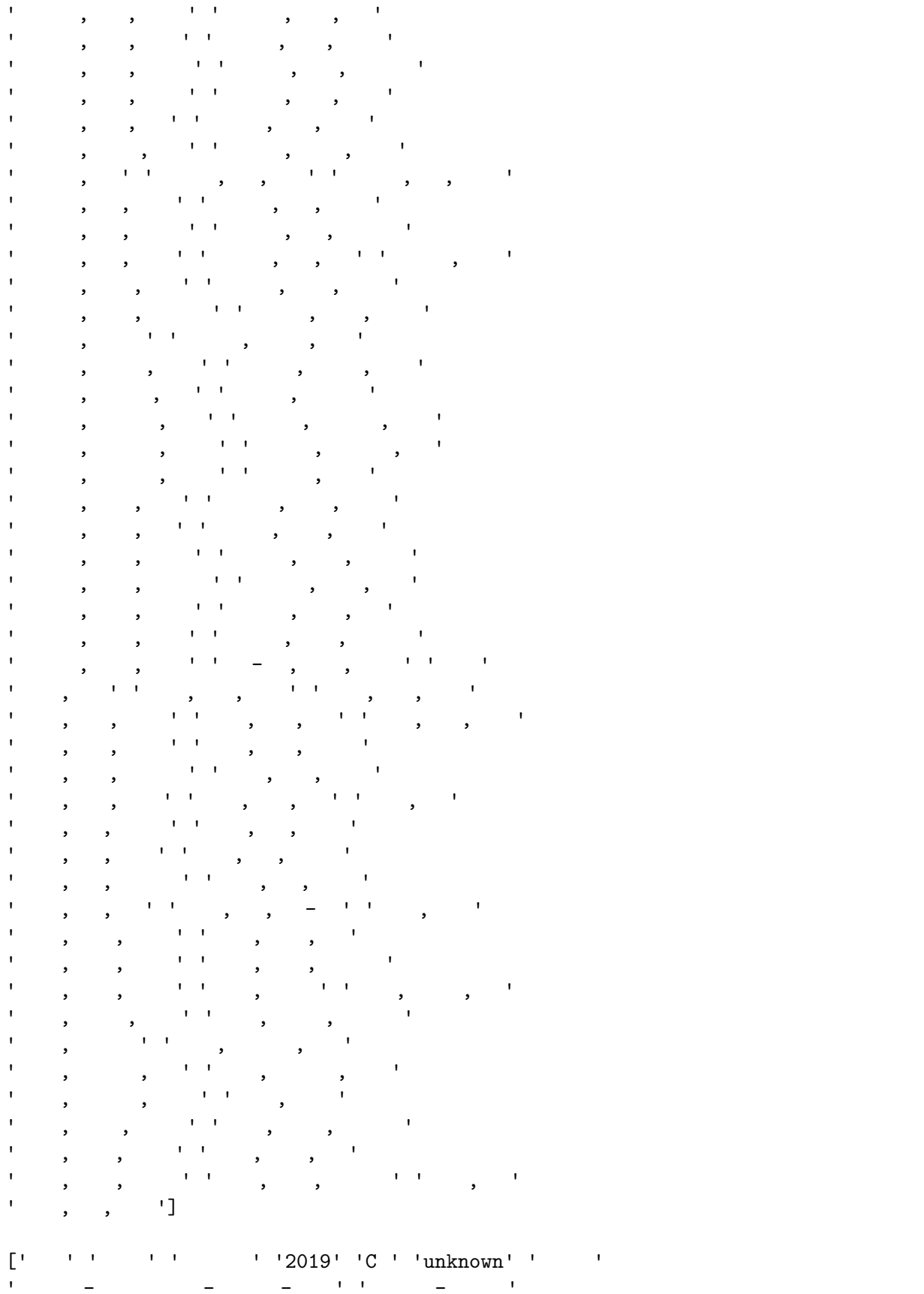


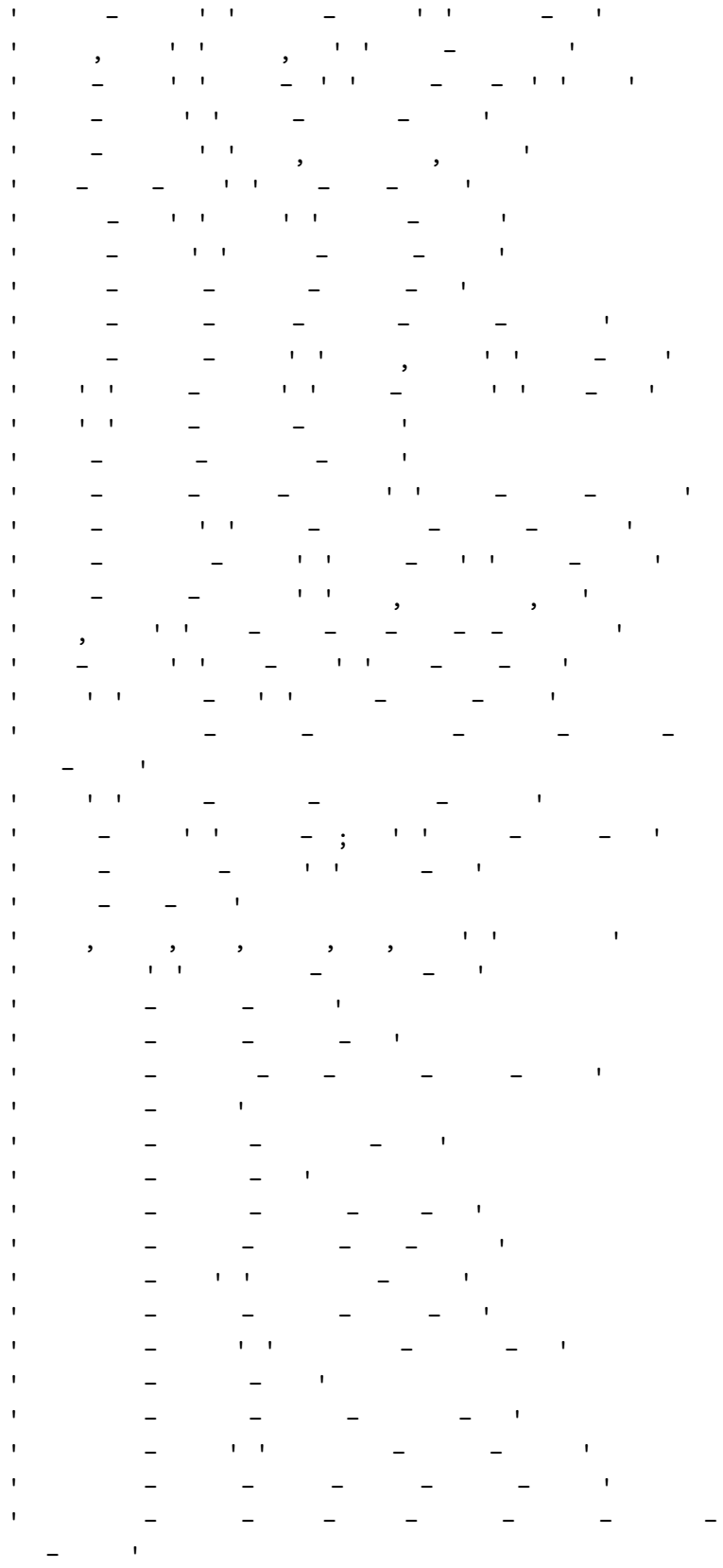


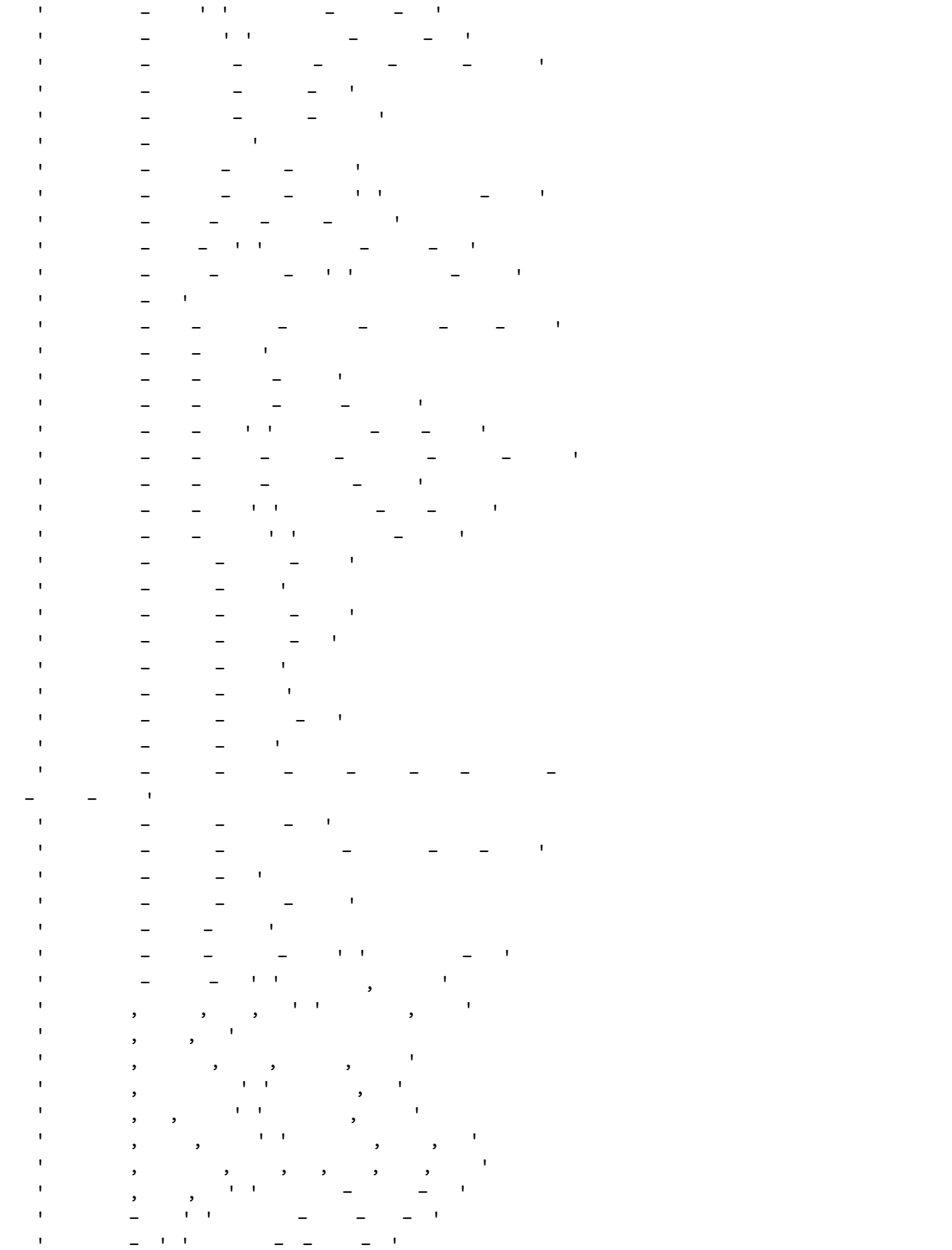


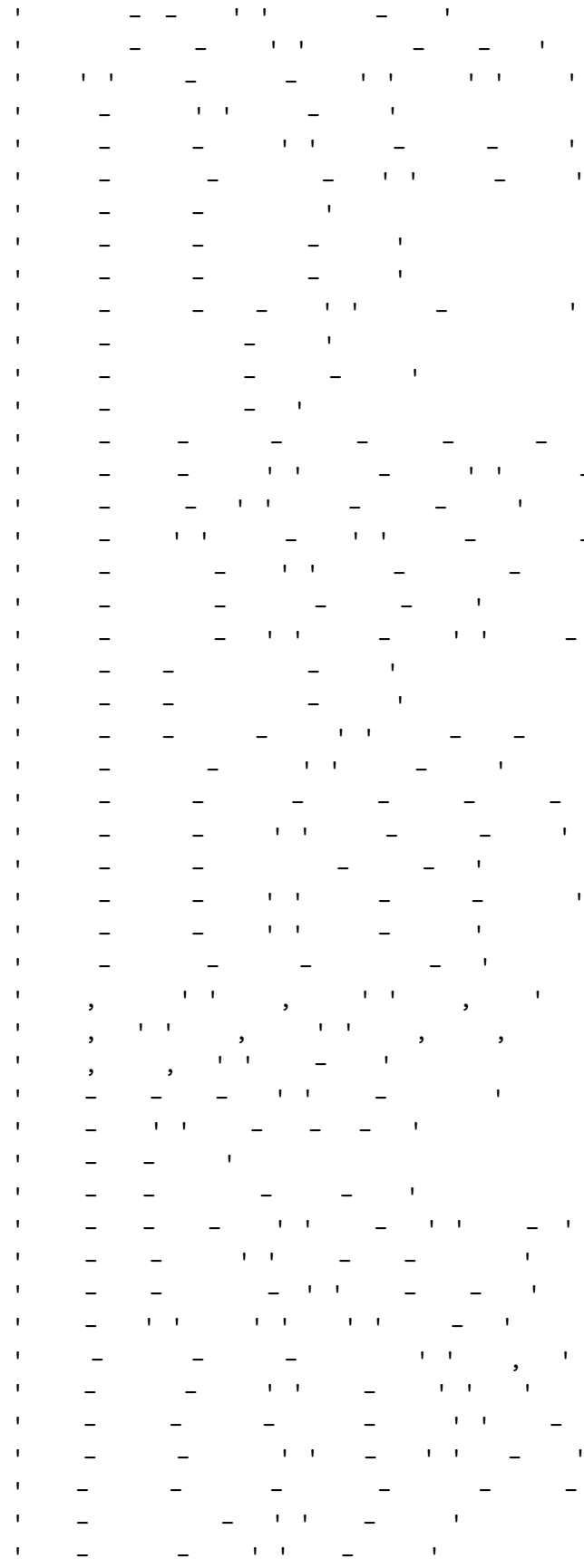


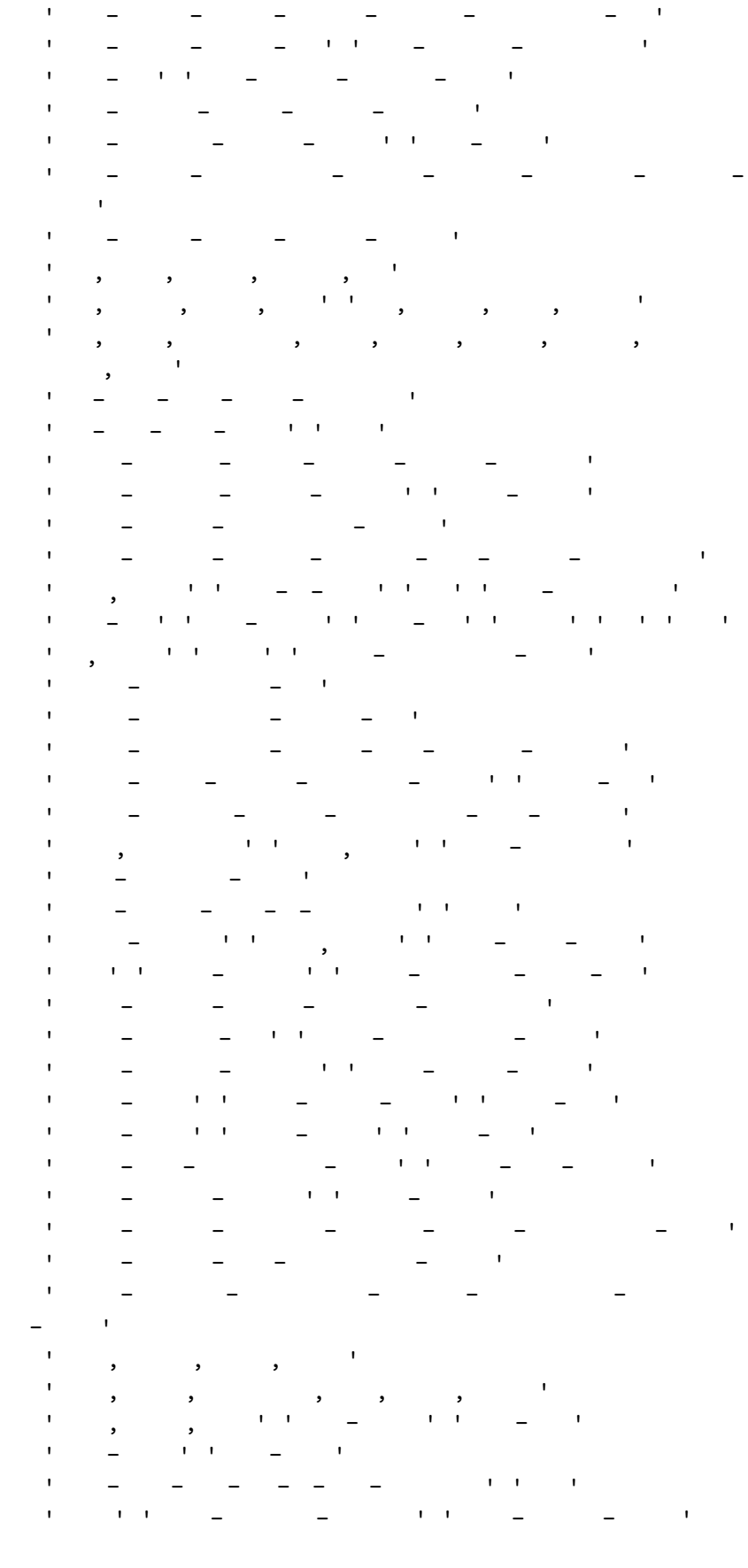


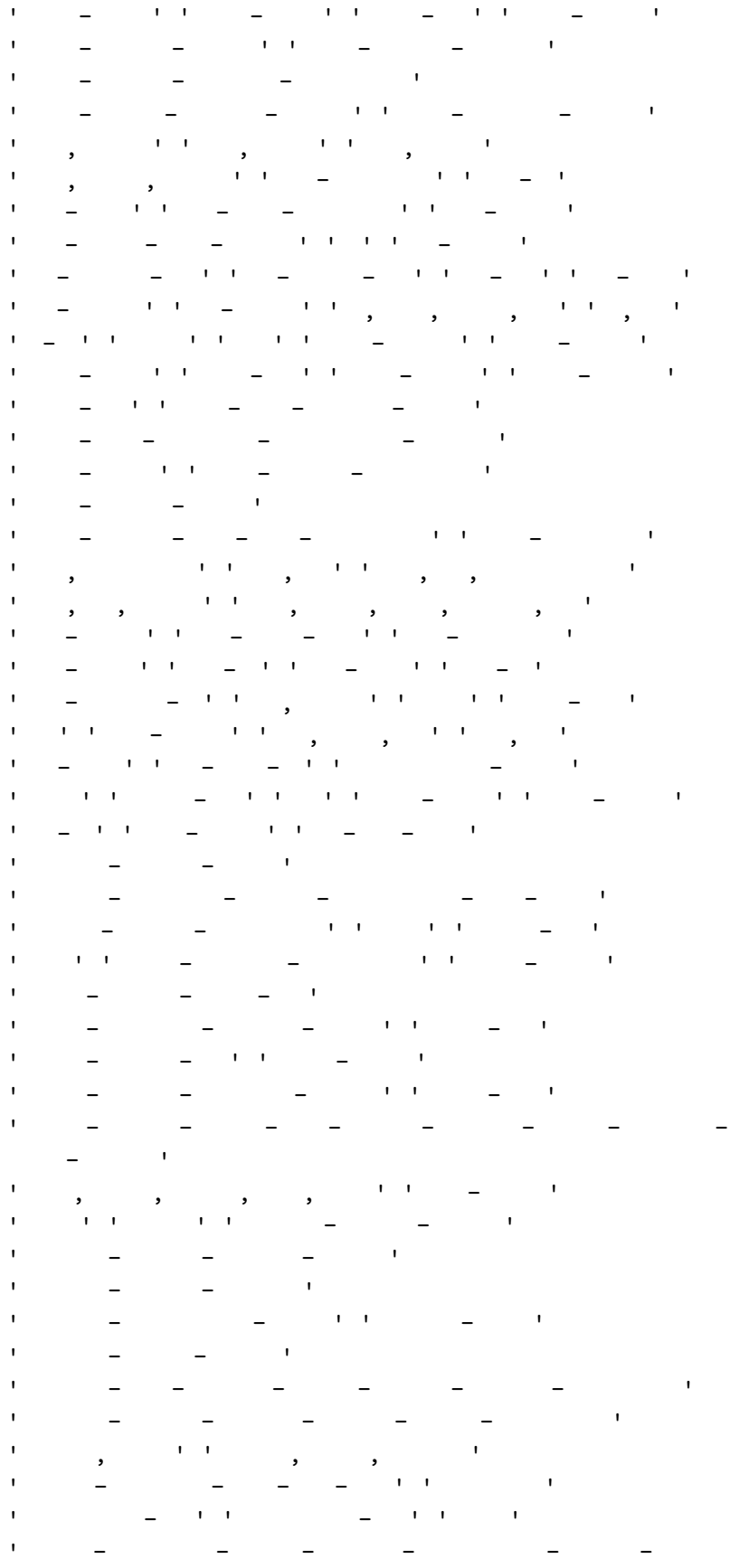


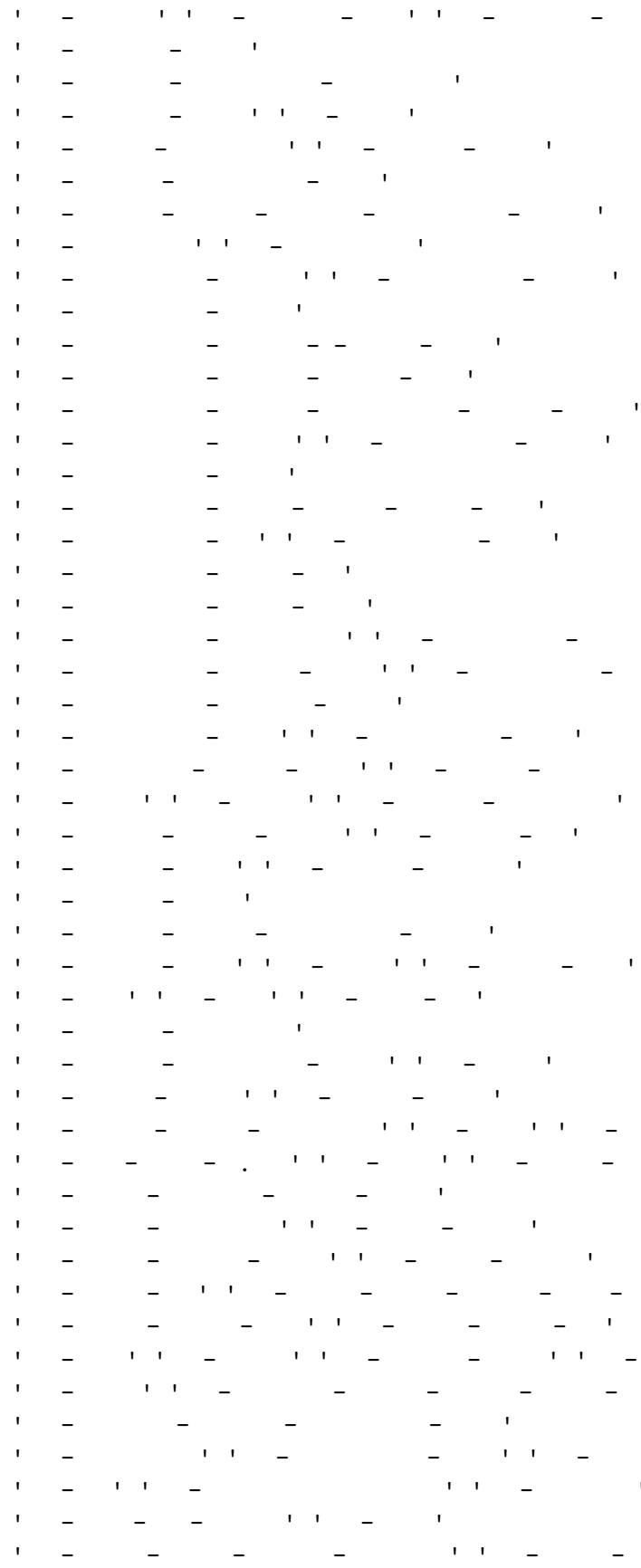


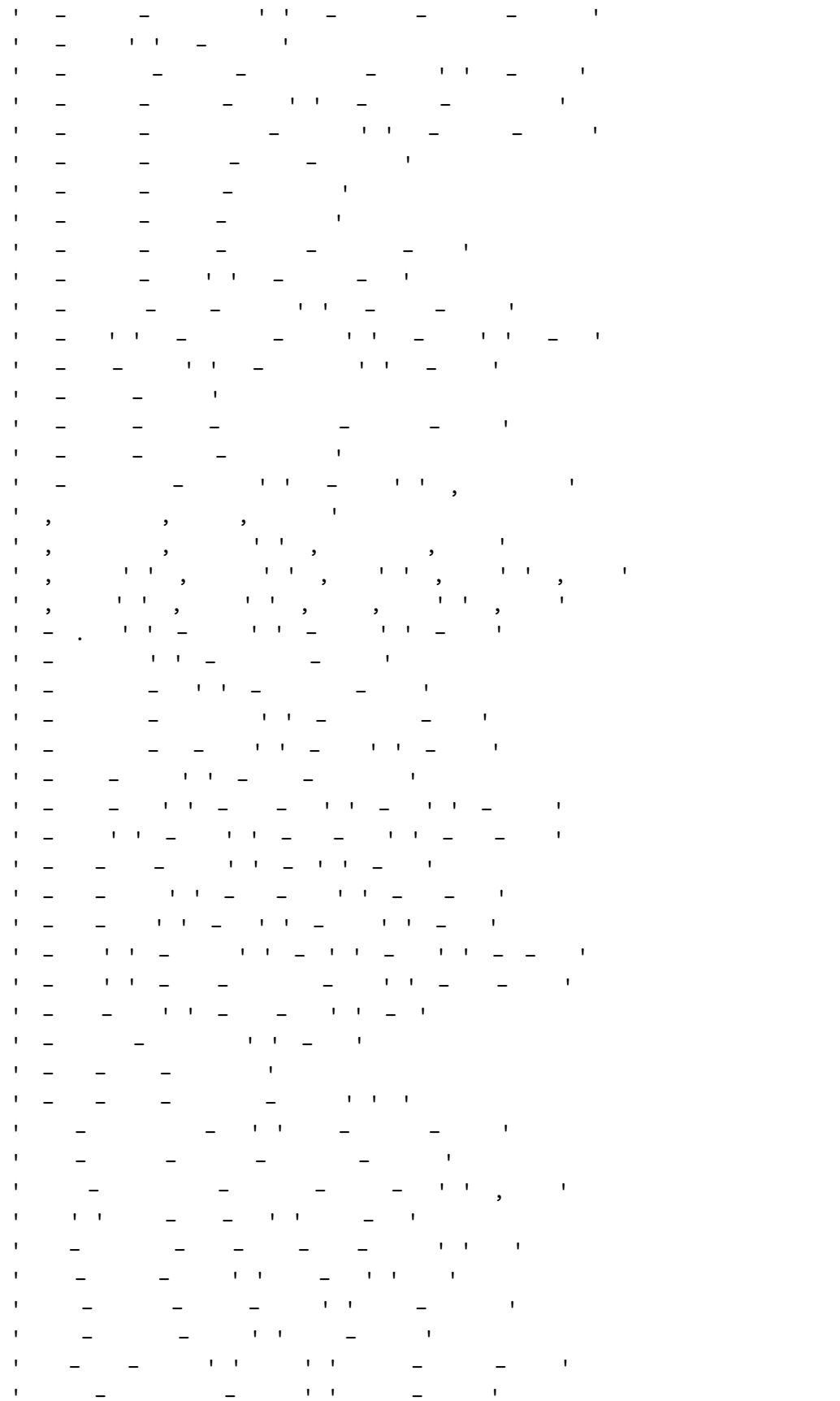


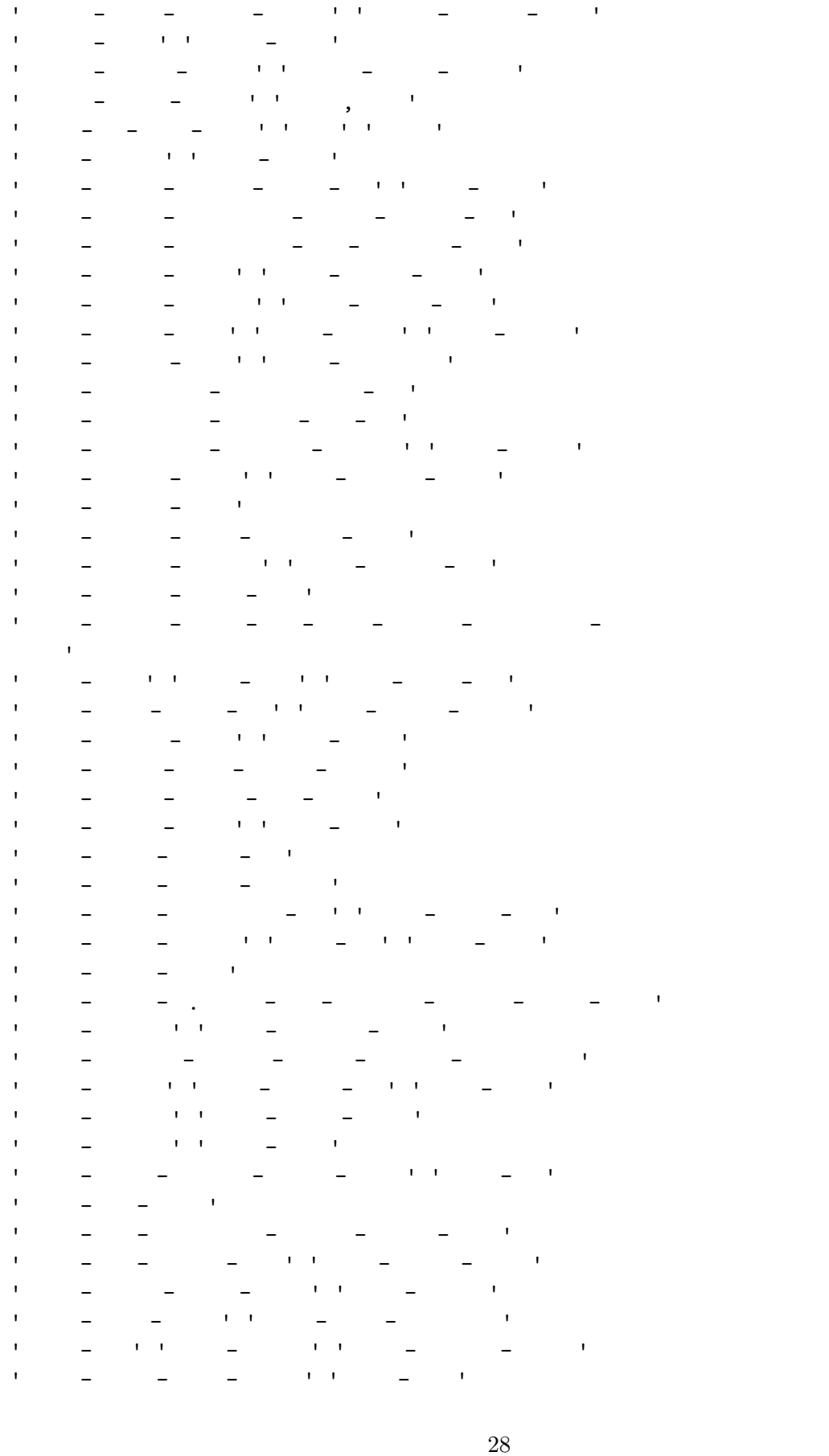


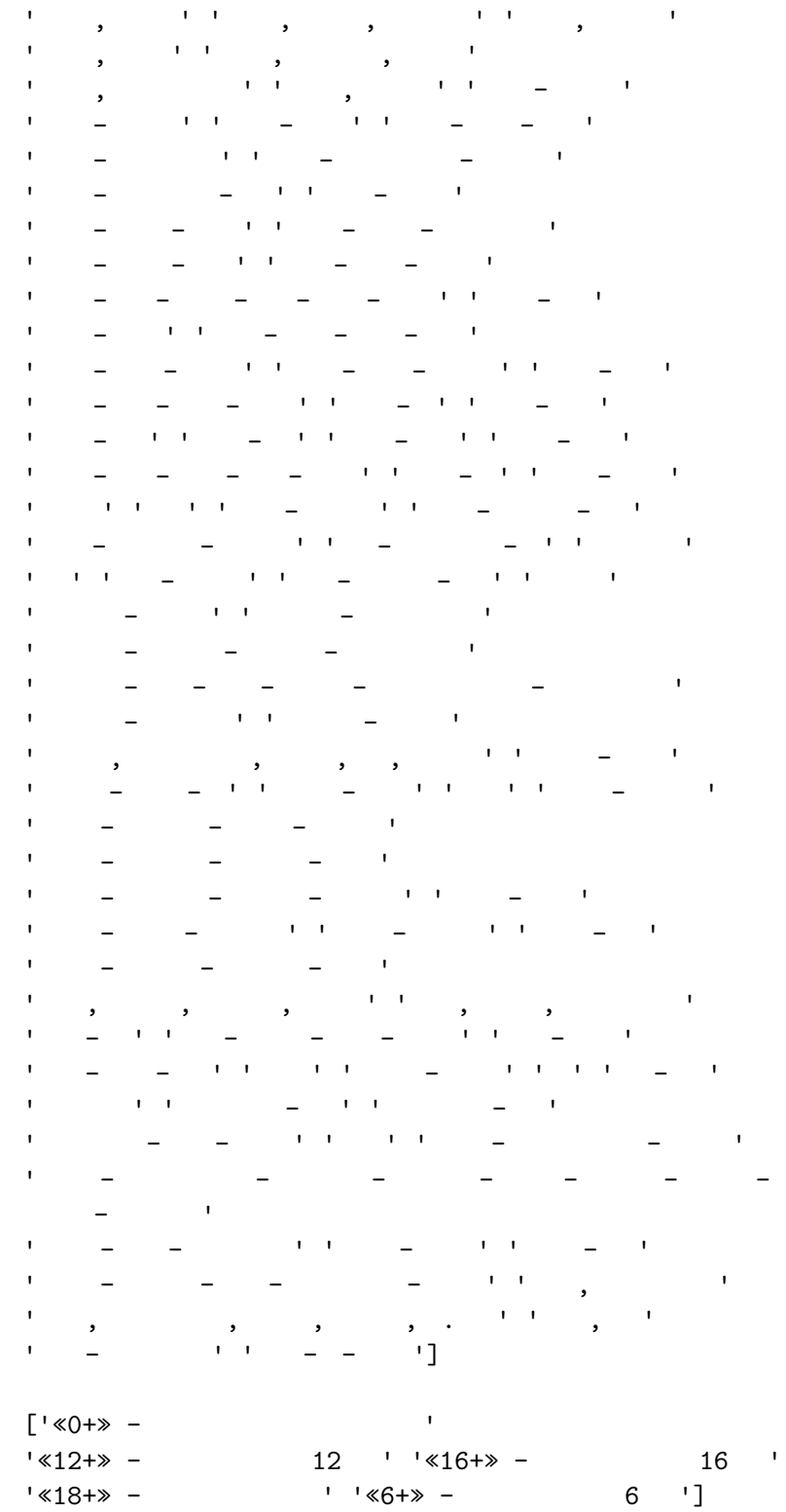












```
['«0+» -                    '
'«12+» -            12   ' '«16+» -            16   '
'«18+» -             ' '«6+» -            6   ']
```

Categorical columns mainly has value of list of values, hovewer, formating, accuracy of data recording and punctiation shall be revised

[18]:
```
# display the unique values of columns movie type
df_movies.type.unique()

# spaces deletion
df_movies.type = df_movies.type.str.strip()

# display of uniaue values
df_movies.type.unique()
```

[18]:
```
array(['        ', '       ', '    ', '        ',
       '   -     ', '     -        '], dtype=object)
```

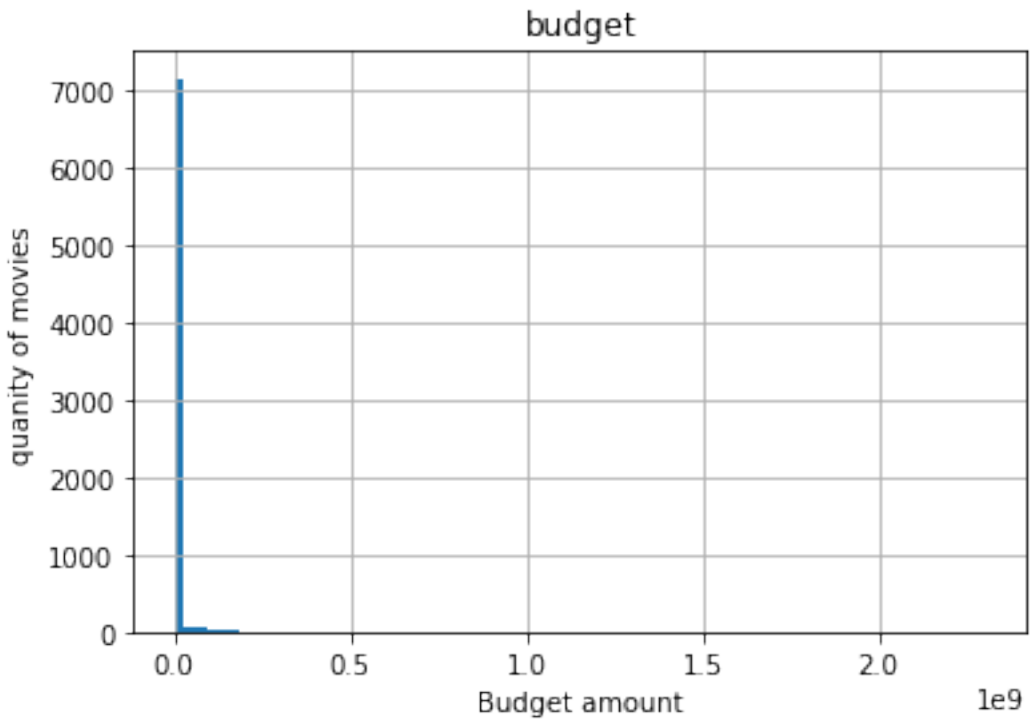
Step 2.5. Numeric columns processing

[19]:
```
# budget calculation function
def budget_check (df_name):
    if (df_name['refundable_support'] + df_name['nonrefundable_support']) >␣
 ↪df_name['budget']:
        return(df_name['refundable_support'] + df_name['nonrefundable_support'])
    else:
        return (df_name['budget'])

df_movies['budget'] = df_movies.apply(budget_check,axis=1)
```

[20]:
```
# diplay info on budget
print(df_movies.query('budget>0')['budget'].describe())

# plotting of hist for budget
df_movies.hist('budget',bins = 100, range =(0,df_movies.budget.max()))
pl.xlabel("Budget amount")
pl.ylabel("quanity of movies")
```

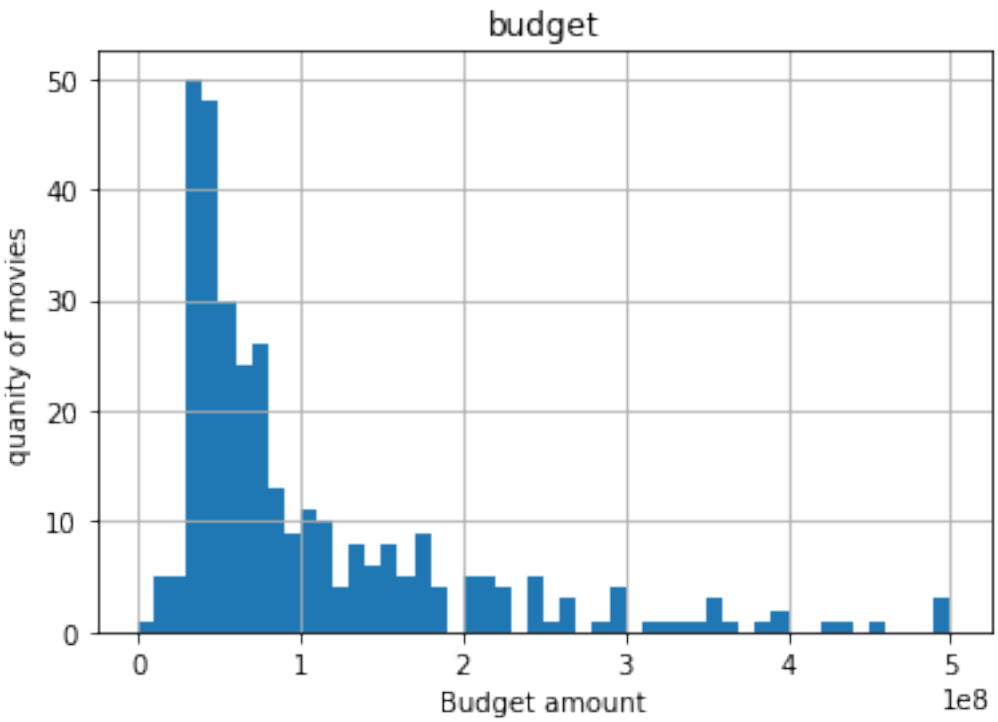
```
count    3.320000e+02
mean     1.314224e+08
std      1.871482e+08
min      6.000000e+06
25%      4.552480e+07
50%      7.119690e+07
75%      1.500000e+08
max      2.305074e+09
Name: budget, dtype: float64
```

[20]:
```
Text(0, 0.5, 'quanity of movies')
```



[21]:
```
# rescaling
df_movies.hist('budget',bins = 50, range =(100,500000000))
pl.xlabel("Budget amount")
pl.ylabel("quanity of movies")
```

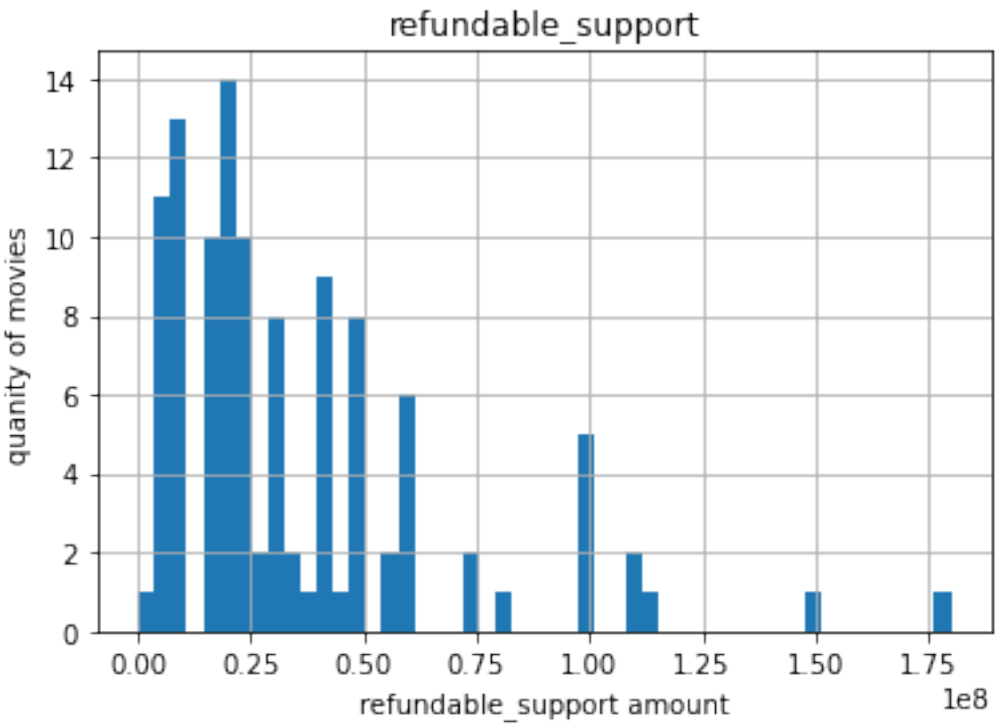
[21]: Text(0, 0.5, 'quanity of movies')



[22]:
```
# display info on column refundable_support
print(df_movies.query('refundable_support>0')['refundable_support'].describe())

# plotting of histogram
df_movies.hist('refundable_support',bins = 50, range =(1,df_movies.
  ↪refundable_support.max()))
pl.xlabel("refundable_support amount")
pl.ylabel("quanity of movies")
```

```
count    1.110000e+02
mean     3.548649e+07
std      3.197288e+07
min      3.500000e+06
25%      1.500000e+07
50%      2.500000e+07
75%      5.000000e+07
max      1.800000e+08
Name: refundable_support, dtype: float64
```

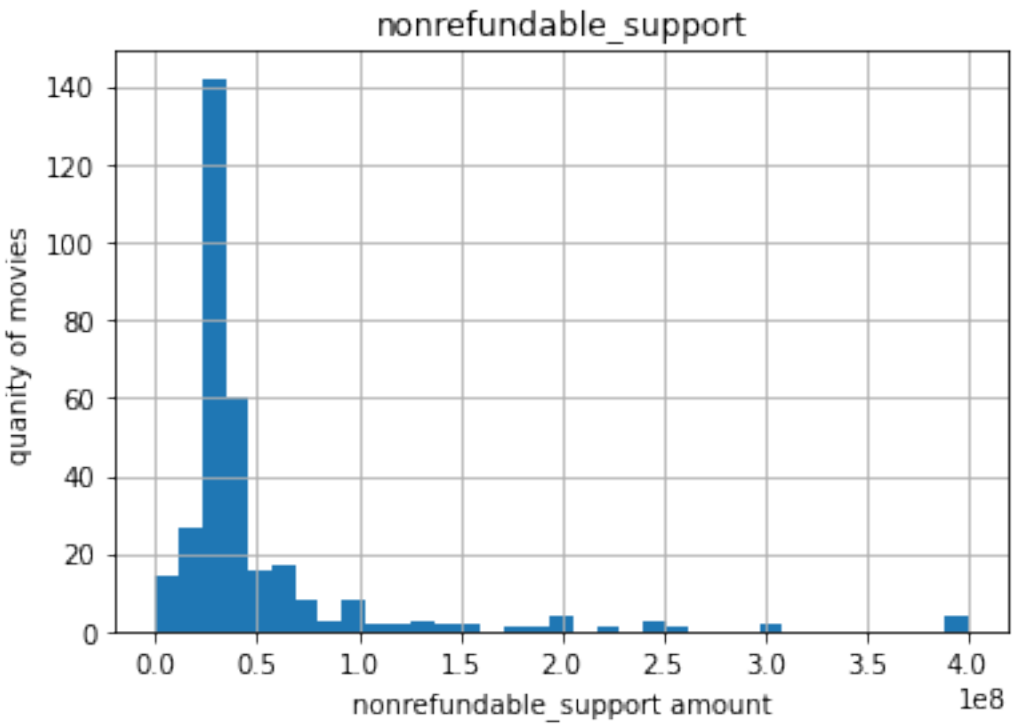
[22]: Text(0, 0.5, 'quanity of movies')



[23]:
```
# display info on column nonrefundable_support
print(df_movies.query('nonrefundable_support>0')['nonrefundable_support'].
 ↪describe())

# # plotting of histogram
df_movies.hist('nonrefundable_support',bins = 35, range =(1,df_movies.
 ↪nonrefundable_support.max()))
pl.xlabel("nonrefundable_support amount")
pl.ylabel("quanity of movies")
```

```
count    3.230000e+02
mean     5.034578e+07
std      6.024321e+07
min      3.000000e+06
25%      2.500000e+07
50%      3.000000e+07
75%      4.500000e+07
max      4.000000e+08
Name: nonrefundable_support, dtype: float64
```

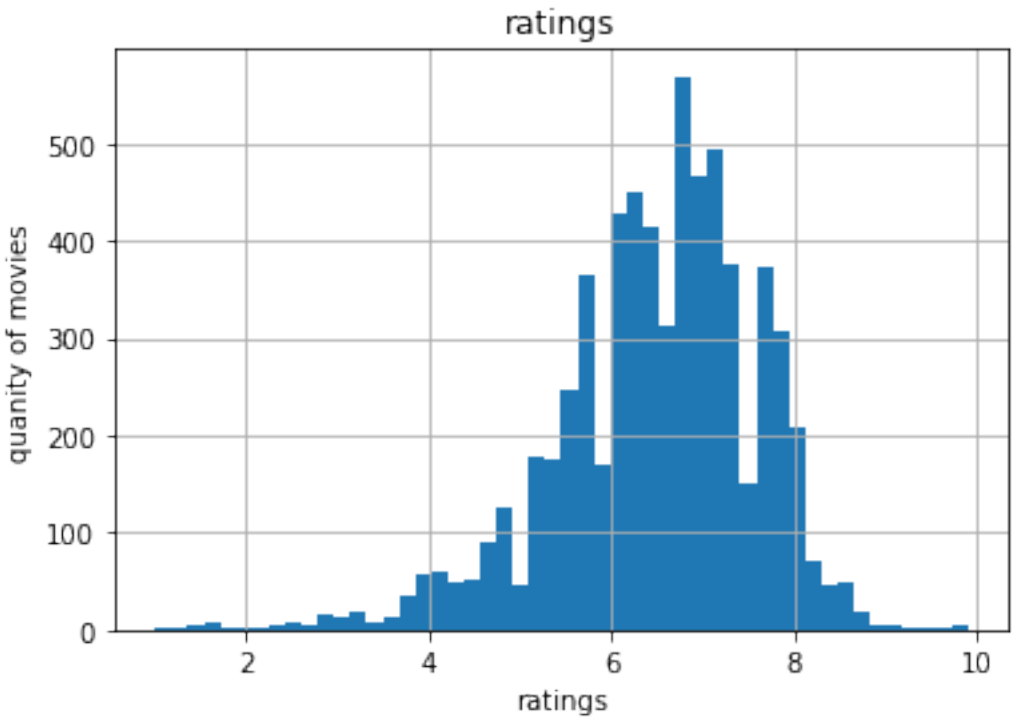
[23]: `Text(0, 0.5, 'quanity of movies')`



[24]:
```
# display info on column ratings
print(df_movies.query('ratings>0')['ratings'].describe())

# histogram plotting
df_movies.hist('ratings',bins = 50, range =(1,df_movies.ratings.max()))
pl.xlabel("ratings")
pl.ylabel("quanity of movies")
```

```
count    6519.000000
mean        6.488173
std         1.114638
min         1.000000
25%         5.900000
50%         6.600000
75%         7.200000
max         9.900000
Name: ratings, dtype: float64
```

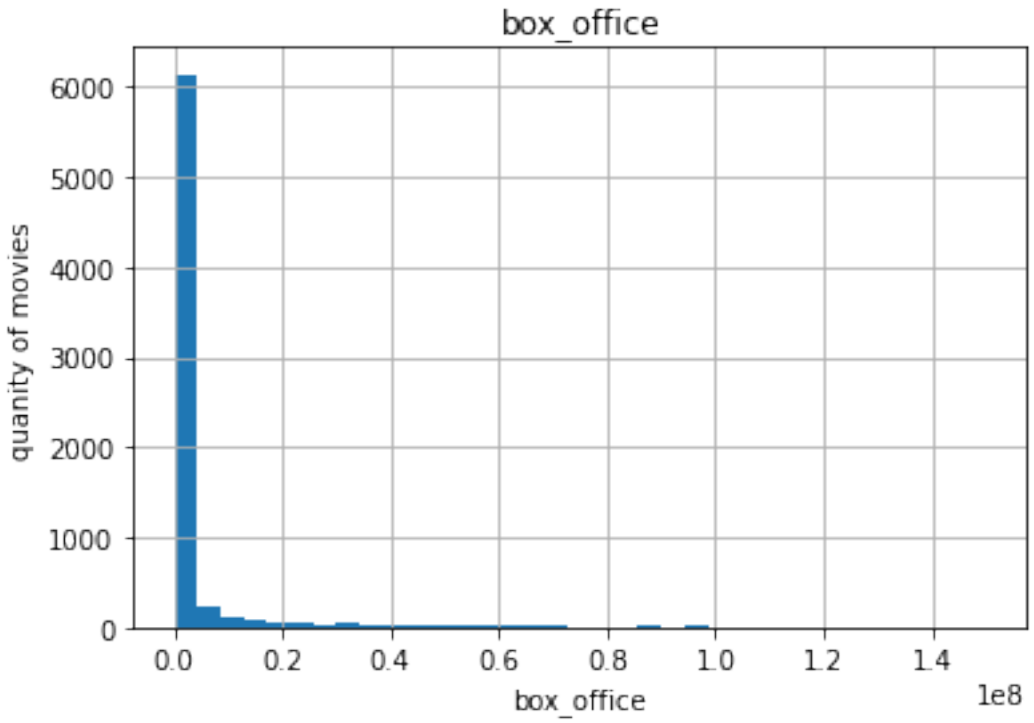
[24]: Text(0, 0.5, 'quanity of movies')



[25]:
```
# display info on column box_office
print(df_movies.query('box_office>0')['box_office'].describe())

# histogram plotting
df_movies.hist('box_office',bins = 35, range =(0,150000000))
pl.xlabel("box_office")
pl.ylabel("quanity of movies")
```

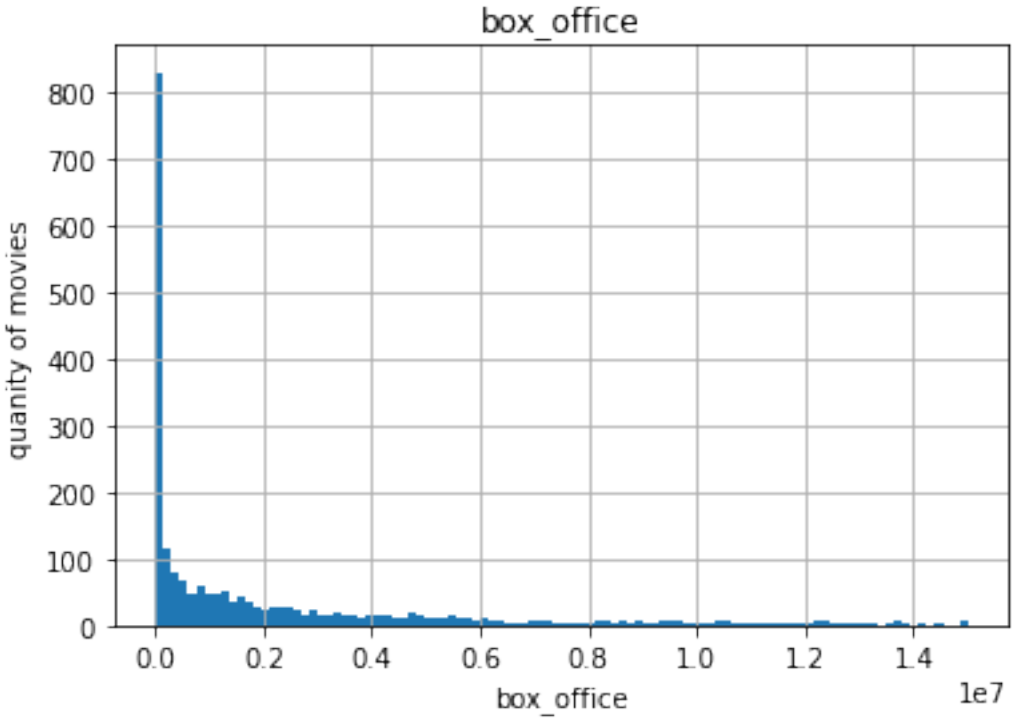
```
count    3.134000e+03
mean     7.706437e+07
std      2.411784e+08
min      4.000000e+01
25%      1.010288e+05
50%      2.409099e+06
75%      2.456979e+07
max      3.073569e+09
Name: box_office, dtype: float64
```

[25]: `Text(0, 0.5, 'quanity of movies')`



[26]:
```
# rescale
df_movies.hist('box_office',bins = 100, range =(100,15000000))
pl.xlabel("box_office")
pl.ylabel("quanity of movies")
```

[26]: Text(0, 0.5, 'quanity of movies')



Conclusions:

- Values in column refundable_support are destributed in accordance with Poisson's law, irregular values are not found.
- Values in column nonrefundable_support are destributed in accordance with Poisson's law, irregular values are not found.
- Values in column budget are destributed in accordance with Poisson's law, irregular values are not found.
- Values in column ratings are destributed in accordance with Poisson's law, but in opposite direcction, irregular values are not found.
- Values in columnbox_office are destributed in accordance with Poisson's law, irregular values are not found.

Step 2.6. Creation of new columns

- Separation of year from issuance date

[27]:
```
# add new column - year
df_movies['year_start_date'] = df_movies['show_start_date'].dt.year

# display head of df
df_movies.head()
```

[27]:
```
                  title   puNumber           show_start_date            type  \
0                221048915 2015-11-27 12:00:00+00:00
1               111013716 2016-09-13 12:00:00+00:00
2                  221038416 2016-10-10 12:00:00+00:00
3                  221026916 2016-06-10 12:00:00+00:00
4                  221030815 2015-07-29 12:00:00+00:00

                                       film_studio production_country  \
0           ,          ,       …
1                                  "    "
2            ,           ,     …
3            ,           ,     …
4            ,           ,     …

            director                                          producer  \
0                            ,         ,
1           .                                             unknown
2                    ,         ,     .   ,
3                    ,         ,     .   ,
4                    ,         ,     .   ,

                 age_restriction  refundable_support  nonrefundable_support  \
0    «18+» -                            0.0                    0.0
1  «6+» -            6                  0.0                    0.0
2    «18+» -                            0.0                    0.0
3    «18+» -                            0.0                    0.0
4    «18+» -                            0.0                    0.0

   budget financing_source  ratings                    genres  box_office  \
0     0.0          unknown      7.2         ,   ,              0.0
1     0.0          unknown      6.6                  ,             0.0
2     0.0          unknown      6.8        ,    ,             0.0
3     0.0          unknown      6.8        ,    ,             0.0
4     0.0          unknown      6.8        ,    ,             0.0

   year_start_date
0             2015
1             2016
2             2016
3             2016
4             2015
```

- Create two columns: - extract name of leading director and leading country. (first values of list).

[28]:
```
# function for selection of first value from the list
def select_first (array):
    return(array.split(',',1)[0])
```

```
# add a new column with leading genre
df_movies['main_genre'] =  df_movies['genres'].apply(select_first)


# function for selection of first value, separated with dash symbol
def select_first_1 (array):
    return(array.split('-',1)[0])

# add a new column with lead production country
df_movies['main_country'] =  df_movies['production_country'].
 ↪apply(select_first_1)

# dicplay the results
df_movies.head()
```

[28]:

	title	puNumber	show_start_date	type	film_studio	production_country	director	producer	age_restriction	refundable_support	nonrefundable_support	budget	financing_source	ratings	genres	box_office
0		221048915	2015-11-27 12:00:00+00:00		, , …				«18+» -	0.0	0.0	0.0	unknown	7.2	, ,	0.0
1		111013716	2016-09-13 12:00:00+00:00			" "	.	unknown	6+» - 6	0.0	0.0	0.0	unknown	6.6	,	0.0
2		221038416	2016-10-10 12:00:00+00:00		, , …		, , . ,		«18+» -	0.0	0.0					
3		221026916	2016-06-10 12:00:00+00:00		, , …		, , . ,		«18+» -	0.0	0.0					
4		221030815	2015-07-29 12:00:00+00:00		, , …		, , . ,		«18+» -	0.0	0.0					

```
2     0.0          unknown      6.8       ,    ,            0.0
3     0.0          unknown      6.8       ,    ,            0.0
4     0.0          unknown      6.8       ,    ,            0.0

   year_start_date  main_genre main_country
0             2015
1             2016
2             2016
3             2016
4             2015
```

- Calculation of percentage of goverment support to total movie budget.

[29]:
```
# add a new column
df_movies['goverment_support'] = round((df_movies['refundable_support']  +␣
 ↪df_movies['nonrefundable_support']) / df_movies['budget']*100,0)

# display the results
df_movies.head()
```

[29]:
```
                 title   puNumber           show_start_date            type  \
0                 221048915 2015-11-27 12:00:00+00:00
1                111013716 2016-09-13 12:00:00+00:00
2                   221038416 2016-10-10 12:00:00+00:00
3                   221026916 2016-06-10 12:00:00+00:00
4                   221030815 2015-07-29 12:00:00+00:00

                                       film_studio production_country  \
0           ,          ,       …
1                                  "    "
2            ,           ,     …
3            ,           ,     …
4            ,           ,     …

            director                                          producer  \
0                          ,        ,
1           .                                             unknown
2                   ,         ,     .   ,
3                   ,         ,     .   ,
4                   ,         ,     .   ,

                age_restriction  refundable_support  nonrefundable_support  \
0    «18+» -                           0.0                    0.0
1  «6+» -            6                 0.0                    0.0
2    «18+» -                           0.0                    0.0
3    «18+» -                           0.0                    0.0
4    «18+» -                           0.0                    0.0
```

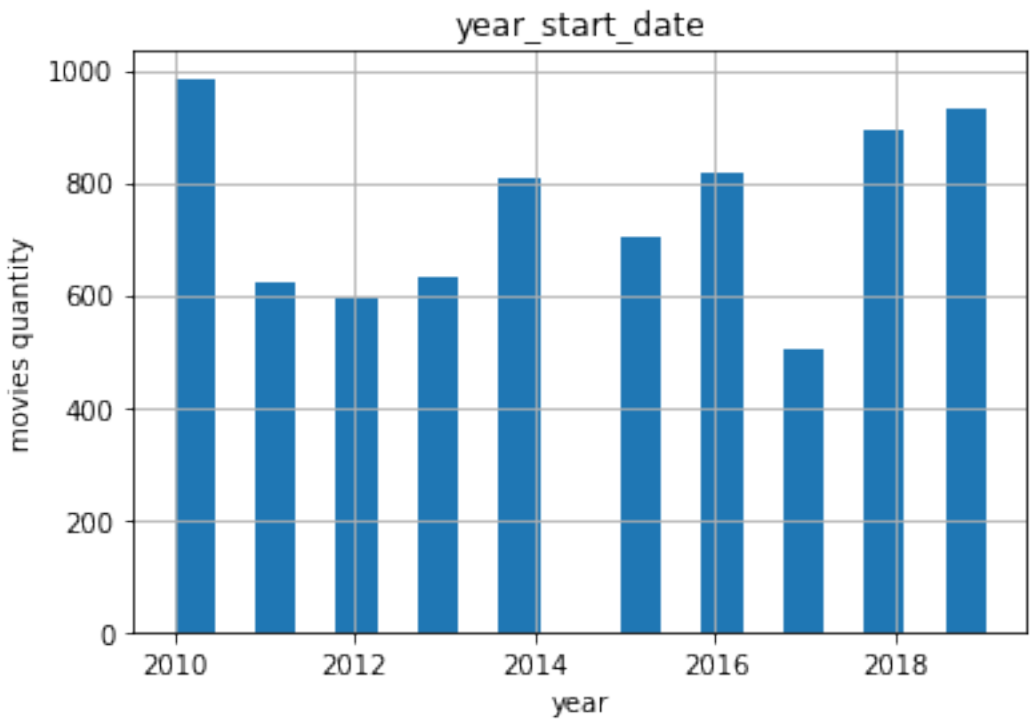
```
   budget financing_source  ratings                    genres  box_office  \
0     0.0          unknown      7.2        ,   ,             0.0
1     0.0          unknown      6.6                  ,            0.0
2     0.0          unknown      6.8        ,   ,            0.0
3     0.0          unknown      6.8        ,   ,            0.0
4     0.0          unknown      6.8        ,   ,            0.0

   year_start_date  main_genre main_country  goverment_support
0             2015                                       NaN
1             2016                                       NaN
2             2016                                     NaN
3             2016                                     NaN
4             2015                                     NaN
```

0.1.3 Step 3. Exploratory data analysis

- Analyze the qauntity of issued movies per year. Notice, not every movie has box office information. Calculate the percentage of movies with specified box office information from total quantity.

[30]:
```
# histogram plotting - quantity of movies per year
df_movies.hist(column = 'year_start_date',grid=True, bins=20 ,range =␣
 ↪(df_movies.year_start_date.min(),df_movies.year_start_date.max()))
pl.xlabel("year")
pl.ylabel("movies quantity")
```

[30]:
```
Text(0, 0.5, 'movies quantity')
```

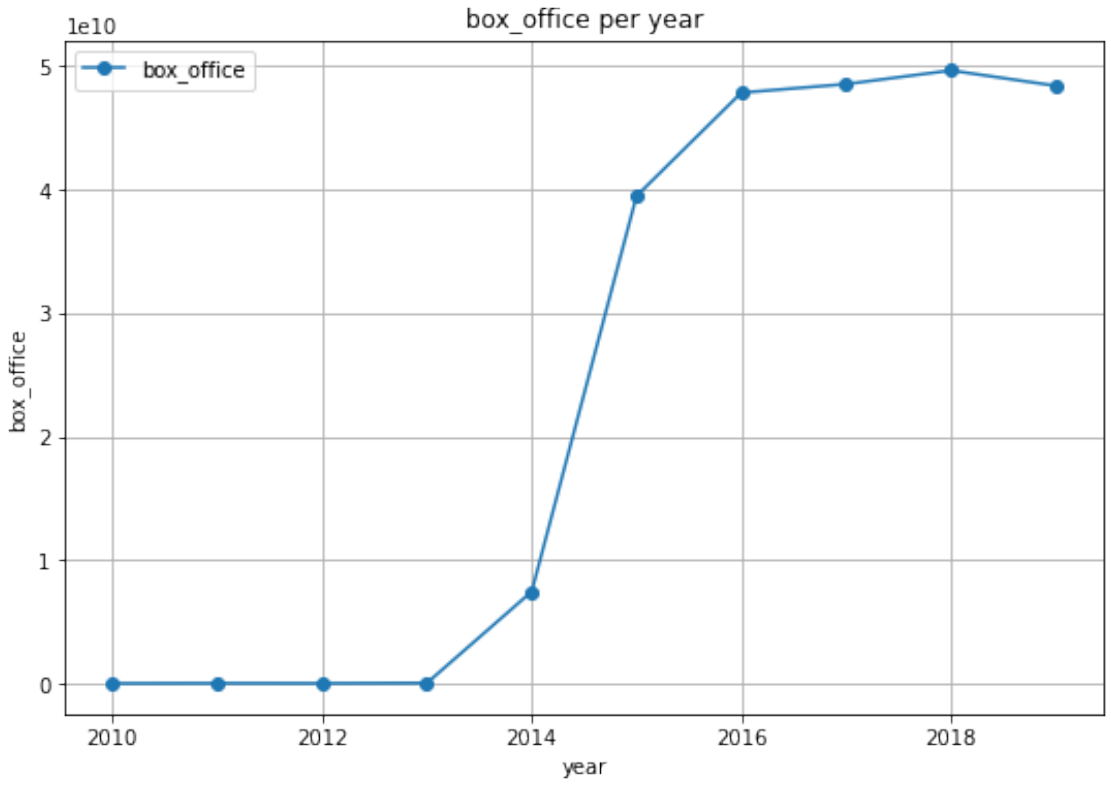


[31]:
```
# Display the percentage of movies with info on box office
print('                    :', round((df_movies.query('box_office > 0').title.
 ↪count()/df_movies.title.count()),2))

# plotting histogram on box office per year
df_box_office = df_movies.query('box_office > 0')
df_box_office.groupby('year_start_date')['box_office'].sum().
 ↪plot(x='year_start_date', y ='box_office', style = 'o-',grid = True, legend␣
 ↪= True,figsize = (9,6))
pl.xlabel("year")
pl.title("box_office per year")
pl.ylabel("box_office")

# display the pivot table with box office per year
df_movies.query('box_office > 0').groupby('year_start_date')['box_office'].
 ↪sum().sort_values()
```

```
                    : 0.42
```

[31]:
```
year_start_date
2010    2.428654e+06
2012    6.955423e+06
2011    1.410276e+07
2013    2.979971e+07
```

```
2014    7.444952e+09
2015    3.949737e+10
2016    4.786630e+10
2019    4.842571e+10
2017    4.856371e+10
2018    4.966840e+10
Name: box_office, dtype: float64
```



- Analyze the dynamic of box office during years. When the amount of box office were lowest and when highest?

Box office dynamics

- statistic on all movies, including the movies w/o info on box office is following:

1) in 2010 almost the 1000 movies were issued
2) in period starting from 2011 to 2013 600 movies were issued per year
3) in 2014 and 2016 the 800 movies were issued
4) in 2015 the quantity of issued movies is near to 700
5) in 2017 year 500 movies were issued
6) in 2018 and 2019 were issued around 900 movies

Box office statistic is following:

- the lowest box office amount was in 2010 and is equeal to 2.428654e+06
- the highest box office amount was in 2018 and is equal to 4.966840e+10

- Using pivot table calculate the averade and median values of box office per year.

[32]:
```
# display the average and median values on box office
display(df_movies.query('box_office > 0').pivot_table(index =␣
 ↪'year_start_date',values = 'box_office',
                                                       aggfunc =␣
 ↪['mean','median']))
```

	mean	median
	box_office	box_office
year_start_date		
2010	2.404608e+04	1710.000
2011	1.293832e+05	3000.000
2012	5.654815e+04	6220.000
2013	1.664788e+05	3580.000
2014	2.727089e+07	20400.000
2015	8.549213e+07	5003450.150
2016	9.117390e+07	3915041.020
2017	1.360328e+08	9968340.000
2018	1.045651e+08	8891102.210
2019	9.136926e+07	4627798.345

The table above represent the statistic on the average and median values of box office pre year

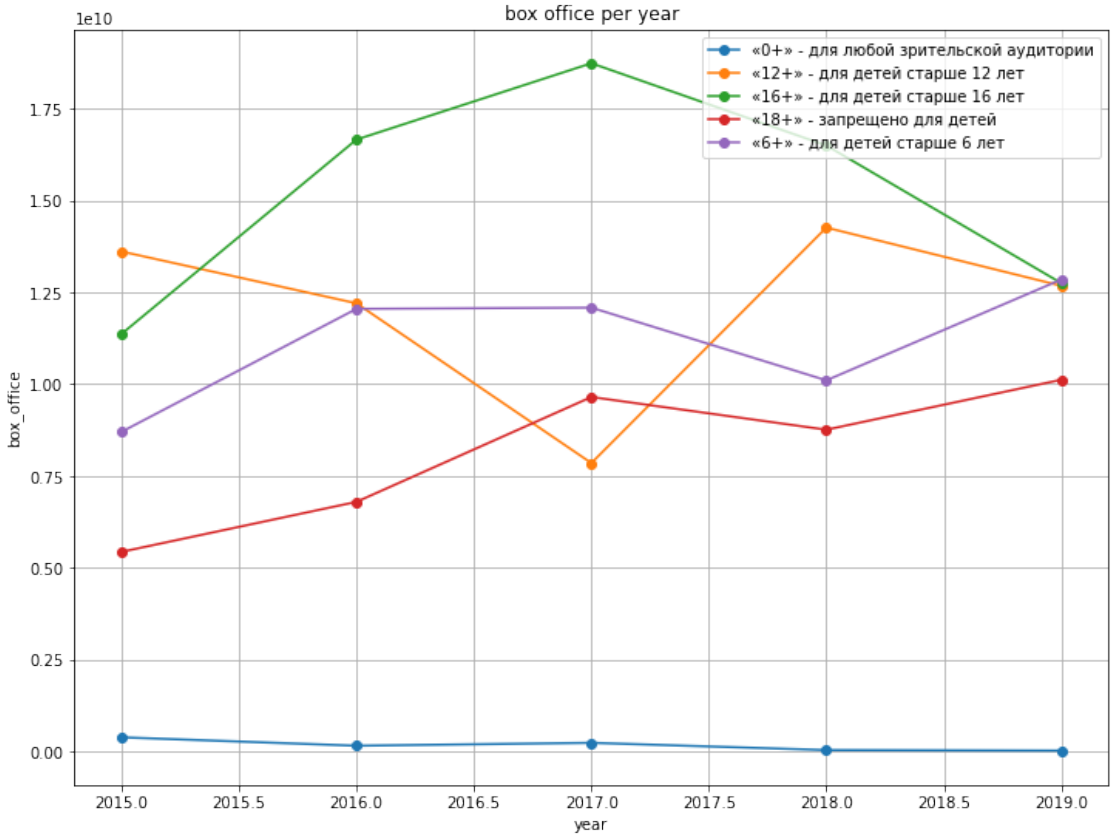
The lowest indecies were in 2010, the highest in 2017

- Define how does the age restriction affect on box office of movies during 2015-2019.
- What type of age restriction has highest box office?
- Does it changing year by year?

[33]:
```
#          2015   2019
years = [2015,2016,2017,2018,2019]

#
ages = df_movies.age_restriction.sort_values().unique()

#
display(df_movies.query('year_start_date in @years').pivot_table(index =␣
 ↪['age_restriction'],values = 'box_office',aggfunc = ['sum']))

#
for age in ages:
    df_temp = df_movies.query('age_restriction == @age and year_start_date in␣
 ↪@years').groupby('year_start_date')['box_office'].sum()
```

```
    df_temp.plot(style='o-',grid='True', figsize=(12,9))
    legend = age
    pl.legend(ages)
    pl.xlabel("year")
    pl.ylabel("box_office")
    pl.title("box office per year")
```

		sum
		box_office
age_restriction		
«0+» -		8.090774e+08
«12+» -	12	6.061945e+10
«16+» -	16	7.603473e+10
«18+» -		4.075962e+10
«6+» -	6	5.579861e+10



Statistic of box office by age restriction

- the lowest popularity has the movies w/o restrictions, the box office has a reduction every year.
- movies with restriction 6+ has overal growth (from 0,5 le10 in 2015 to 1,25 le10 in 2019).

Most likely it is animated films.

- movies with restriction 12+ and 16+ has similar indecies, we can assume the the growth of box office for movies in 2017 year 16+ has affected on reduction of box office of movies 12+. Most likely this categaries is a fantasy movies.
- Box office for movies with restriction 18+ has growth every year.

0.1.4 Step 4. Analyze the movies with goverment support

Conduct the analysis and calculate the amount of goverment support. Chechk the ROI indecies and rating of suchmovies.

[34]:
```
# calculation of total amount of gov support
sum_gov_support = df_movies.refundable_support.sum() + df_movies.
 ↪nonrefundable_support.sum()

# calculatio of total qty of movies qith goverment support
qty_gov_support = df_movies.query('refundable_support >0').refundable_support.
 ↪count() + df_movies.query('nonrefundable_support>0').nonrefundable_support.
 ↪count()

print('movies with refundable support',df_movies.query('refundable_support >0').
 ↪refundable_support.describe(),
      '\n','movies with nonrefundable support',
      df_movies.query('nonrefundable_support >0').nonrefundable_support.
 ↪describe())

# display of the gov support statistic by the year
print('sum = ',sum_gov_support,'\n'
     ,'quantity = ',qty_gov_support,'\n'
     ,'average = ',round(sum_gov_support/qty_gov_support,2),'\n\n'
      ,df_movies.groupby('year_start_date').refundable_support.sum(),'\n\n'
      ,df_movies.groupby('year_start_date').nonrefundable_support.sum())
```

```
movies with refundable support count    1.110000e+02
mean     3.548649e+07
std      3.197288e+07
min      3.500000e+06
25%      1.500000e+07
50%      2.500000e+07
75%      5.000000e+07
max      1.800000e+08
Name: refundable_support, dtype: float64
 movies with nonrefundable support count    3.230000e+02
mean     5.034578e+07
std      6.024321e+07
min      3.000000e+06
25%      2.500000e+07
50%      3.000000e+07
```

```
75%      4.500000e+07
max      4.000000e+08
Name: nonrefundable_support, dtype: float64
sum =  20200688312.0
 quantity =  434
 average =  46545364.77

 year_start_date
2010            0.0
2011            0.0
2012            0.0
2013            0.0
2014     71000000.0
2015    637153119.0
2016    921500000.0
2017    719346881.0
2018    662000000.0
2019    928000000.0
Name: refundable_support, dtype: float64

 year_start_date
2010    0.000000e+00
2011    0.000000e+00
2012    0.000000e+00
2013    1.343479e+08
2014    5.010023e+08
2015    3.019088e+09
2016    3.381655e+09
2017    2.464625e+09
2018    2.784969e+09
2019    3.976000e+09
Name: nonrefundable_support, dtype: float64
```

Conclusion - total qty of movies with goverment support is 434 - total amount of goverment soppurt is 20.200.688.312 - average amount of goverment support is: 46 545 364.77 - minimum and maximum amounts of refundable support is: 1.110000e+02 1.800000e+08 - min and max of unrefoundable goverment support is: 5.034578e+07 4.000000e+08

[35]:
```
# calculation of ROI
df_gov_support = df_movies.query('goverment_support > 0').copy()
print('                    :',int(round(df_gov_support.title.count()/df_movies.
  ↪title.count(),2)*100),'%','\n')
df_gov_support['ROI'] = (df_gov_support['box_office'] /␣
  ↪df_gov_support['budget'])
print(df_gov_support['ROI'].describe())
```

```
                    : 4 %
```

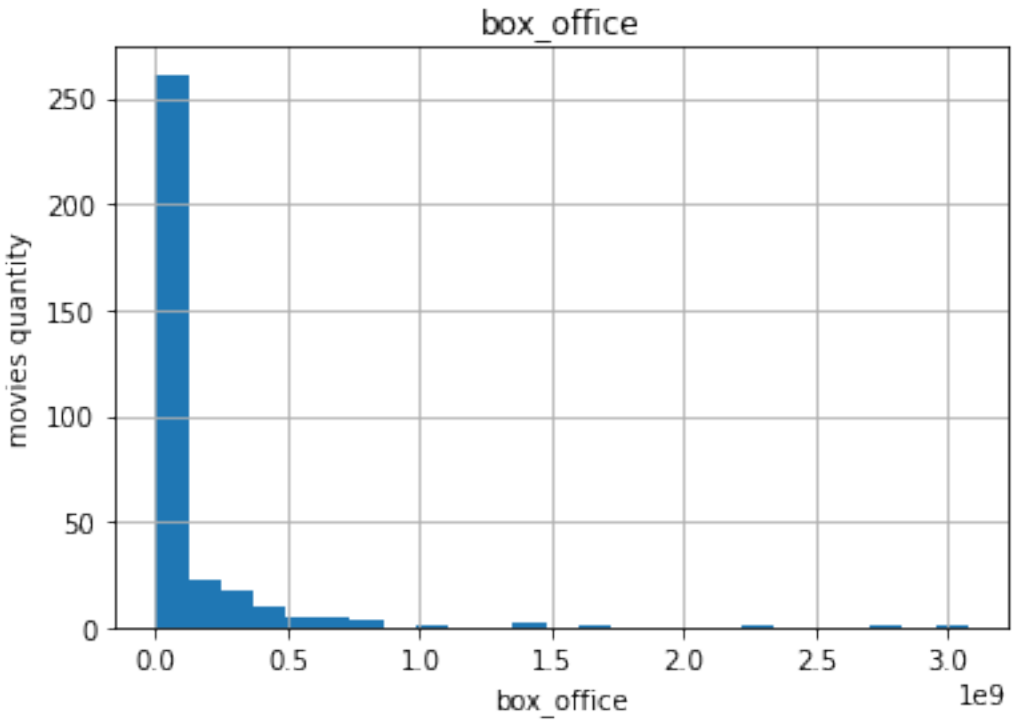
```
count    332.000000
mean       0.791000
std        1.659904
min        0.000000
25%        0.015875
50%        0.148018
75%        0.933302
max       19.209804
Name: ROI, dtype: float64
```

More than 75% of movies has negative ROI index

[36]:
```
# box office analyzis

print(df_movies.query('goverment_support > 0')['box_office'].describe(),'\n')

df_movies.query('goverment_support > 0').hist(column = 'box_office', bins = 25)
pl.ylabel("movies quantity")
pl.xlabel("box_office")
```

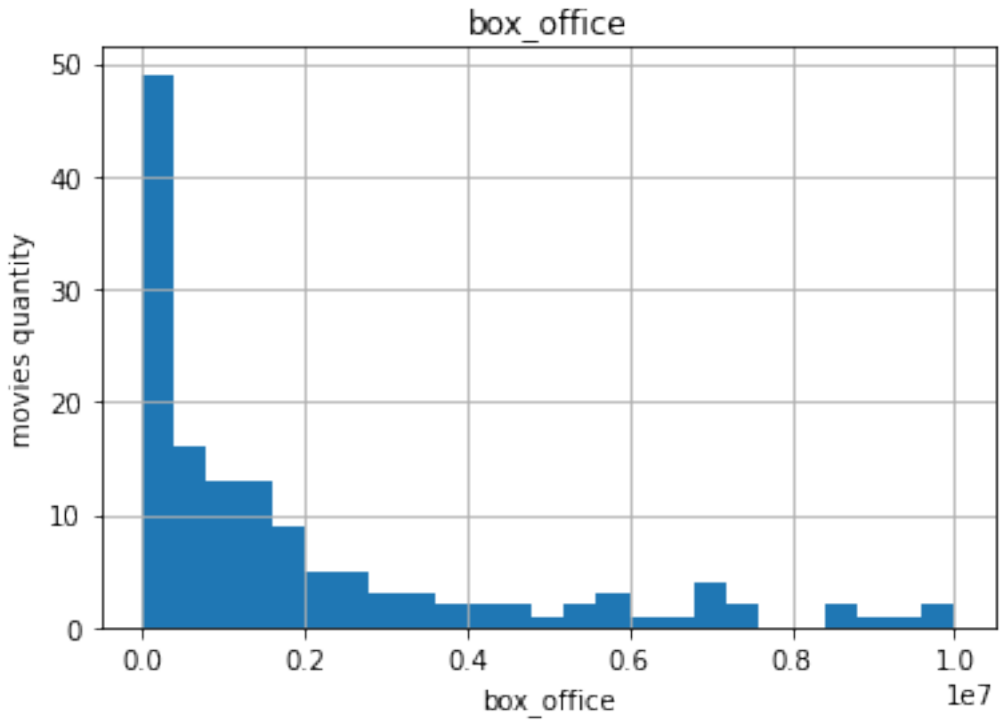
```
count    3.320000e+02
mean     1.268479e+08
std      3.287626e+08
min      0.000000e+00
25%      9.094312e+05
50%      1.214419e+07
75%      1.031074e+08
max      3.073569e+09
Name: box_office, dtype: float64
```

[36]:
```
Text(0.5, 0, 'box_office')
```



[37]:
```
# rescale
df_movies.query('goverment_support > 0').hist(column = 'box_office', bins = 25,␣
 ↪range = (1,10000000))
pl.ylabel("movies quantity")
pl.xlabel("box_office")
```

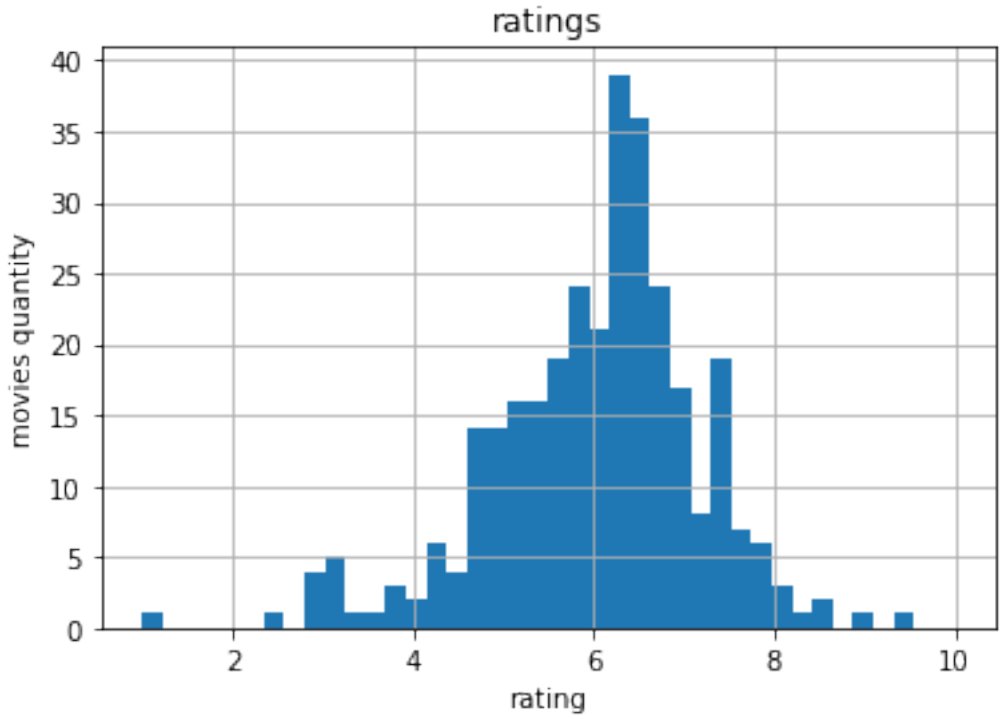
[37]: Text(0.5, 0, 'box_office')



- max amount of box office of movie with goverment support is 3.073569e+09
- min amount is 0
- median value 12 144 193.5

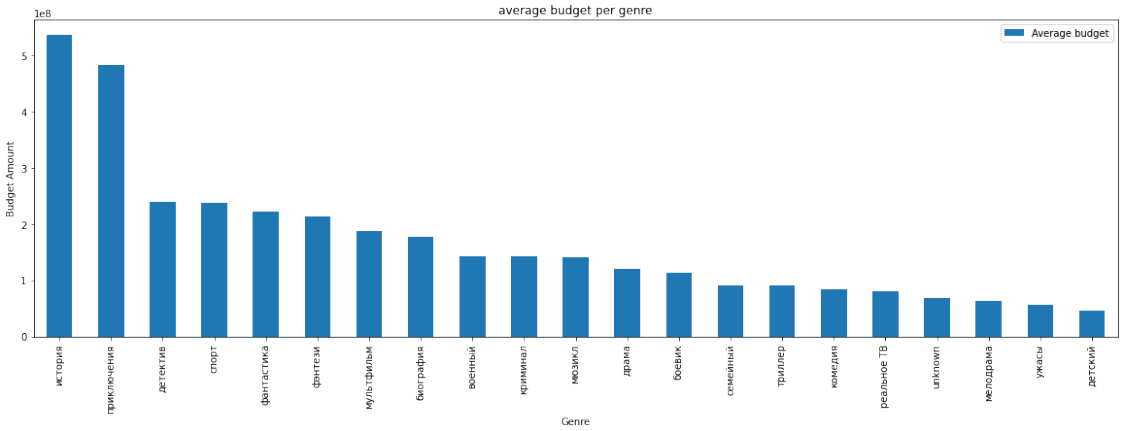
[38]:
```
# plotting of historam with rating of movies with goverment support
df_movies.query('goverment_support > 0').hist(column = 'ratings', bins = 40,␣
 ↪range = (1,10))
pl.ylabel("movies quantity")
pl.xlabel("rating")


# display the rating information
print(df_movies.query('goverment_support > 0')['ratings'].describe())
```

```
count    332.000000
mean       5.730723
std        1.710008
min        0.000000
25%        5.200000
50%        6.100000
75%        6.700000
max        9.400000
Name: ratings, dtype: float64
```



- Maximal rating is 9,4
- Minimal - 1,71
- Median value 6,1

[39]:
```
# histogram plotting of average budget per genre of movie with goverment support

df_genre_budget = df_movies.query('goverment_support > 0').pivot_table(index =␣
 ↪'main_genre',values = 'budget',aggfunc = ['mean'])
df_genre_budget.columns = ['Average budget']
df_genre_budget.sort_values(by = 'Average budget', ascending = False).plot(kind␣
 ↪= 'bar', figsize =(20,6))
pl.ylabel("Budget Amount")
pl.xlabel("Genre")
pl.title('average budget per genre');
```



0.1.5 Step 5. General conclusion

1) The analysis has revealed that box office increased every year, the highest growth was in 2014 and has changed form 0,8 +10 to 4 +10

2) the lowest and highest indecies of box office were in 2010 and 2017 years, and are following:

- 2010 | 2.404608e+04 | 1.710
- 2017 | 1.360328e+08 | 9.968.340

3) The most popular types of movies with age restriction is following categories: 6+ , 12+, 16+. These genres has similar populary and cempete to each other every year, leadin position is always changing. The less popular movies are movies with 18+ age restriction, and the lowest popularity have movies with age restriction +.

4) only 4% movies has goverment support:

- total quantity is 434 movies
- total amount of goverment support is 20.200.688.312
- average amount per movie - 46 545 364.77
- percentage of movies with positive ROI - 25%
- Median value of box office amount is 12 144 193.5, wich 4 times lower than average amoun of goverment support.
- more than a half of movies has rating 6,1 (average and lower)