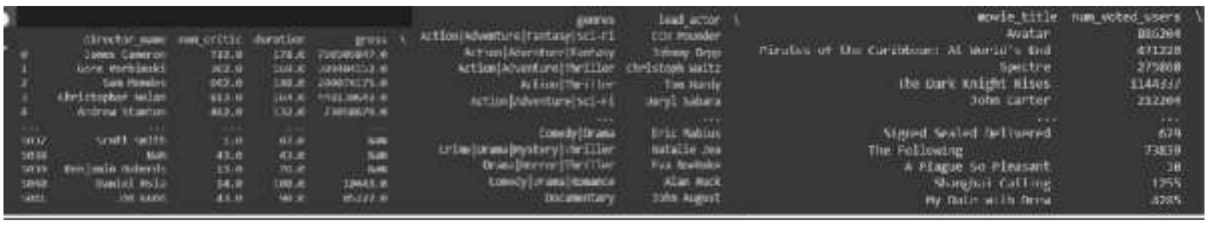
Assignment no.04

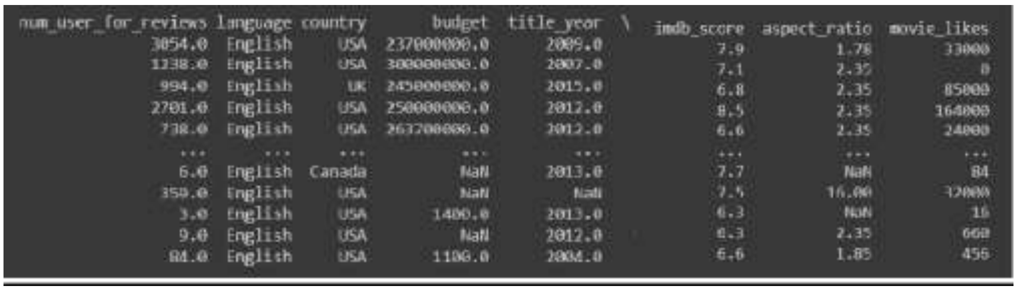
Name :- Nandini Thorat
Div :- C
Roll no:- 384
PRN:- 202201070117

Code:

```
import pandas as pd

df = pd.read_csv("/content/movie_data.csv")
#print all records of dataset
print(df)
```

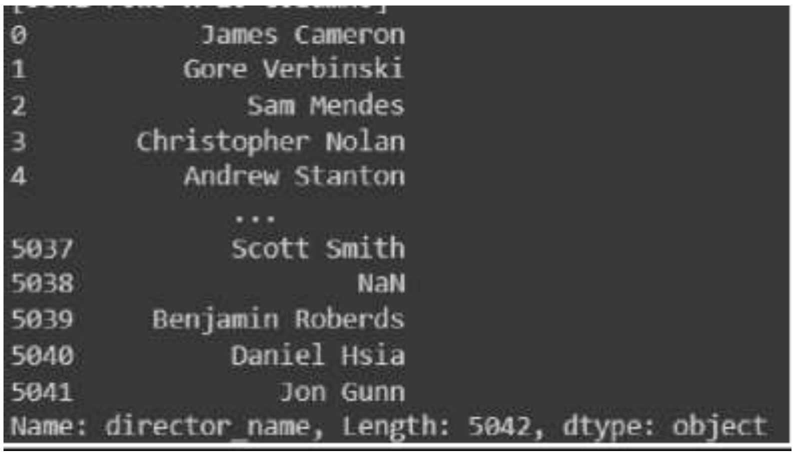


```
           director_name  num_critic  duration        gross  \                          genres         lead_actor  \                          movie_title  num_voted_users  \
0          James Cameron       723.0     178.0  760505847.0   Action|Adventure|Fantasy|Sci-Fi        CCH Pounder                                   Avatar           886204
1         Gore Verbinski       302.0     169.0  309404152.0          Action|Adventure|Fantasy        Johnny Depp  Pirates of the Caribbean: At World's End           471220
2             Sam Mendes       602.0     148.0  200074175.0         Action|Adventure|Thriller    Christoph Waltz                                  Spectre           275868
3      Christopher Nolan       813.0     164.0  448130642.0                   Action|Thriller          Tom Hardy                    The Dark Knight Rises          1144337
4         Andrew Stanton       462.0     132.0   73058679.0           Action|Adventure|Sci-Fi       Daryl Sabara                              John Carter           212204
...                  ...         ...       ...          ...                               ...                ...                                      ...              ...
5037         Scott Smith         1.0      87.0          NaN                      Comedy|Drama        Eric Mabius                  Signed Sealed Delivered              629
5038                 NaN        43.0      43.0          NaN     Crime|Drama|Mystery|Thriller        Natalie Zea                The Following                        73839
5039    Benjamin Roberds        13.0      76.0          NaN            Drama|Horror|Thriller        Eva Boehnke                    A Plague So Pleasant               38
5040         Daniel Hsia        14.0     100.0      10443.0             Comedy|Drama|Romance          Alan Ruck                         Shanghai Calling             1255
5041            Jon Gunn        43.0      90.0      85222.0                       Documentary        John August                      My Date with Drew             4285
```



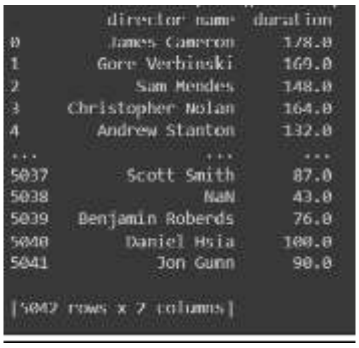
```
num_user_for_reviews language country       budget  title_year  \  imdb_score  aspect_ratio  movie_likes
              3054.0  English     USA  237000000.0      2009.0            7.9          1.78        33000
              1238.0  English     USA  300000000.0      2007.0            7.1          2.35            0
               994.0  English      UK  245000000.0      2015.0            6.8          2.35        85000
              2701.0  English     USA  250000000.0      2012.0            8.5          2.35       164000
               738.0  English     USA  263700000.0      2012.0            6.6          2.35        24000
                 ...      ...     ...          ...         ...            ...           ...          ...
                 6.0  English  Canada          NaN      2013.0            7.7           NaN           84
               359.0  English     USA          NaN         NaN            7.5         16.00        32000
                 3.0  English     USA       1400.0      2013.0            6.3           NaN           16
                 9.0  English     USA          NaN      2012.0            6.3          2.35          660
                84.0  English     USA       1100.0      2004.0            6.6          1.85          456
```

```
#1 print Names of all employees
print(df['director_name'])
```



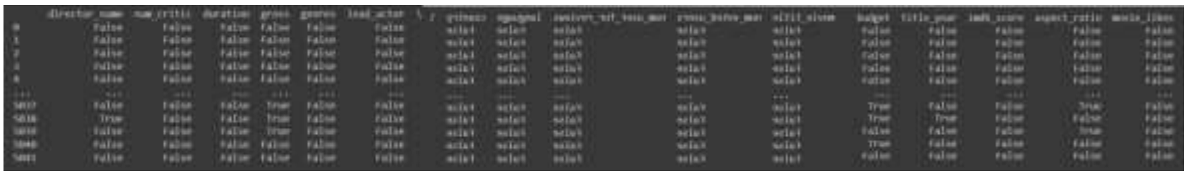
```
0           James Cameron
1          Gore Verbinski
2              Sam Mendes
3       Christopher Nolan
4          Andrew Stanton
               ...
5037          Scott Smith
5038                  NaN
5039     Benjamin Roberds
5040          Daniel Hsia
5041             Jon Gunn
Name: director_name, Length: 5042, dtype: object
```

```
#2 print name and duration
print(df[['director_name','duration']])
```



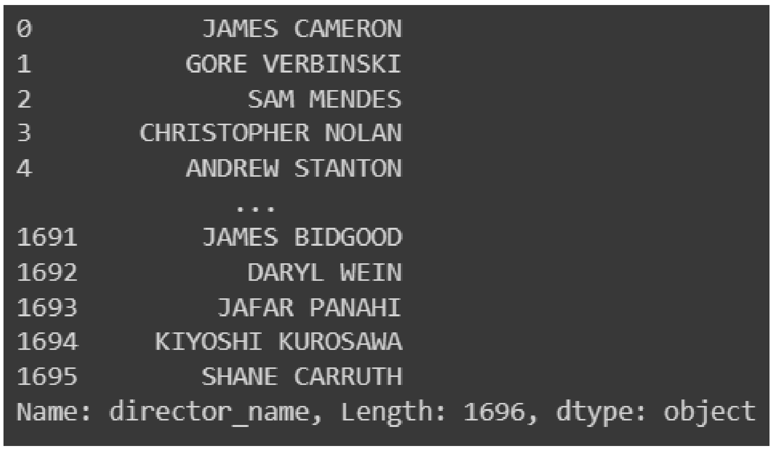
```
         director_name  duration
0        James Cameron     178.0
1       Gore Verbinski     169.0
2           Sam Mendes     148.0
3    Christopher Nolan     164.0
4       Andrew Stanton     132.0
...                ...       ...
5037       Scott Smith      87.0
5038               NaN      43.0
5039  Benjamin Roberds      76.0
5040      Daniel Hsia      100.0
5041          Jon Gunn      90.0

[5042 rows x 2 columns]
```

```
#1 Data cleaning
#check for missing values
print(df.isnull())

# #drop rows with missing values
df.dropna(inplace=True
```



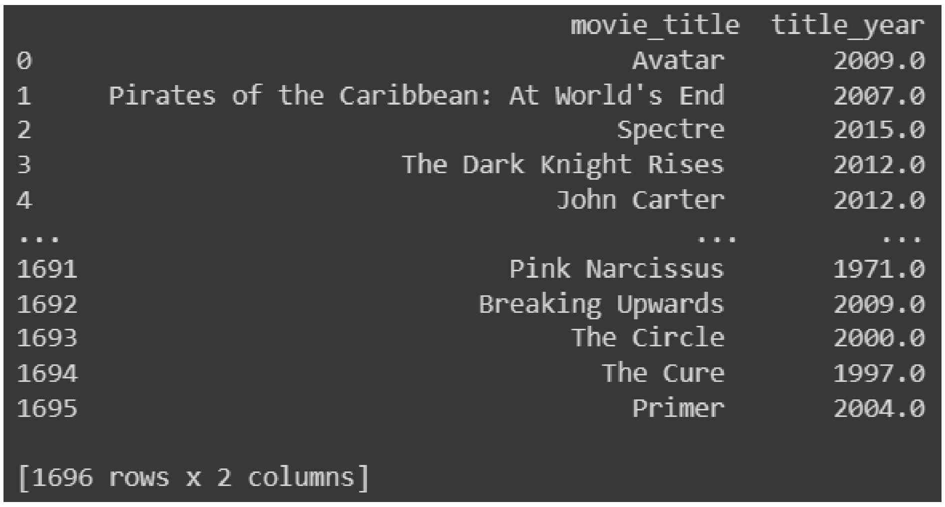
```
      director_name  num_critic  duration  gross  genres  lead_actor  \ [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  budget  title_year  imdb_score  aspect_ratio  movie_likes
0             False       False     False  False   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]   False       False       False         False        False
1             False       False     False  False   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]   False       False       False         False        False
2             False       False     False  False   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]   False       False       False         False        False
3             False       False     False  False   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]   False       False       False         False        False
4             False       False     False  False   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]   False       False       False         False        False
...             ...         ...       ...    ...     ...         ...          ...          ...          ...          ...          ...     ...         ...         ...           ...          ...
5037          False       False     False   True   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]    True       False       False          True        False
5038           True       False     False   True   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]    True        True       False         False        False
5039          False       False     False   True   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]   False       False       False          True        False
5040          False       False     False  False   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]    True       False       False         False        False
5041          False       False     False  False   False       False  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]   False       False       False         False        False
```

```
#2 convert string to upper case
df['director name'].str.upper()
```



```
0           JAMES CAMERON
1          GORE VERBINSKI
2              SAM MENDES
3       CHRISTOPHER NOLAN
4          ANDREW STANTON
              ...
1691        JAMES BIDGOOD
1692           DARYL WEIN
1693         JAFAR PANAHI
1694     KIYOSHI KUROSAWA
1695        SHANE CARRUTH
Name: director_name, Length: 1696, dtype: object
```

```
#3. print movie title along with their year of release
df1 = df[['movie_title','title_year']]
print(df1)
```

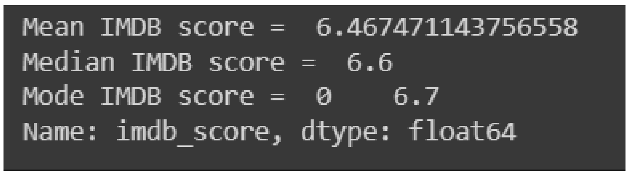


```
                                       movie_title  title_year
0                                           Avatar      2009.0
1     Pirates of the Caribbean: At World's End      2007.0
2                                          Spectre      2015.0
3                            The Dark Knight Rises      2012.0
4                                      John Carter      2012.0
...                                            ...         ...
1691                                Pink Narcissus      1971.0
1692                              Breaking Upwards      2009.0
1693                                    The Circle      2000.0
1694                                      The Cure      1997.0
1695                                        Primer      2004.0

[1696 rows x 2 columns]
```

```
#4 calculate the total budget of all the movies
totalBudget = df['budget'].sum()
print("Total budget of all movies = ", totalBudget)
```

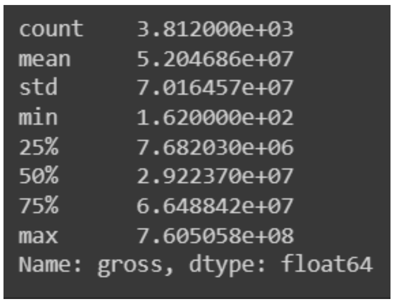
```
Total budget of all movies =  174826107781.0
```

```
#5 calculate mean, median, mode imdb rating
meanImdb = df['imdb_score'].mean()
medianImdb = df['imdb_score'].median()
modeImdb = df['imdb_score'].mode()
print("Mean IMDB score = ", meanImdb)
print("Median IMDB score = ", medianImdb)
print("Mode IMDB score = ", modeImdb)
```



```
Mean IMDB score =  6.467471143756558
Median IMDB score =  6.6
Mode IMDB score =  0    6.7
Name: imdb_score, dtype: float64
```

```
#6 describe gross of all movies
print(df['gross'].describe())
```



```
count    3.812000e+03
mean     5.204686e+07
std      7.016457e+07
min      1.620000e+02
25%      7.682030e+06
50%      2.922370e+07
75%      6.648842e+07
max      7.605058e+08
Name: gross, dtype: float64
```

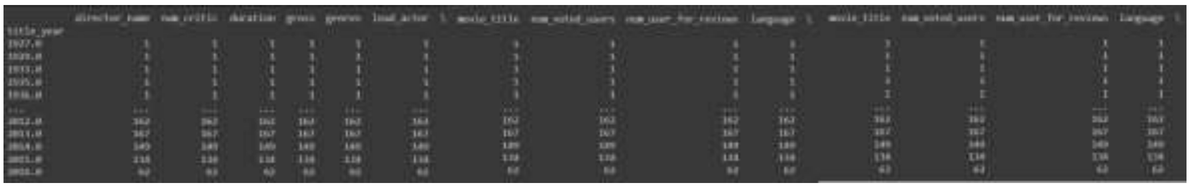
```
#7 minimum and maximum duration movie
minimumDuration = df['duration'].min()
maximumDuration = df['duration'].max()
print("Minimum duration movie: ", minimumDuration)
print("Maximum duration movie: ", maximumDuration)
```

```
Minimum duration movie: 37.0
Maximum duration movie:  330.0
```

```
#8 count number of movies which are released after 2010
released_after_2010 = df[df['title_year'] > 2010]
print("Number of movies released after 2010: ",
released_after_2010['title_year'].count())
```

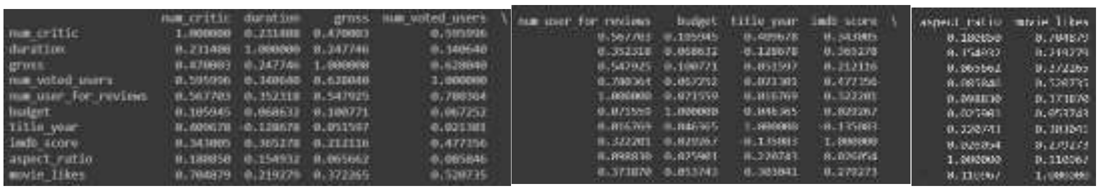
```
Number of movies released after 2010:  430
```

```
#9 print count of movies released in each year
print(df.groupby('title_year').count())
```



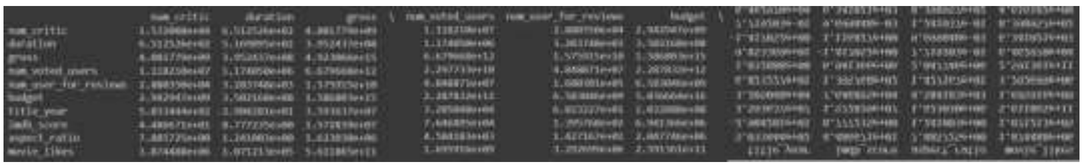
```
            director_name  num_critic  duration  gross  genres  lead_actor  \  movie_title  num_voted_users  num_user_for_reviews  language  \  movie_title  num_voted_users  num_user_for_reviews  language  \
title_year
1927.0                  1           1         1      1       1           1               1                1                     1         1               1                1                     1         1
1929.0                  1           1         1      1       1           1               1                1                     1         1               1                1                     1         1
1933.0                  1           1         1      1       1           1               1                1                     1         1               1                1                     1         1
1935.0                  1           1         1      1       1           1               1                1                     1         1               1                1                     1         1
1936.0                  1           1         1      1       1           1               1                1                     1         1               1                1                     1         1
...                   ...         ...       ...    ...     ...         ...             ...              ...                   ...       ...             ...              ...                   ...       ...
2012.0                162         162       162    162     162         162             162              162                   162       162             162              162                   162       162
2013.0                167         167       167    167     167         167             167              167                   167       167             167              167                   167       167
2014.0                149         149       149    149     149         149             149              149                   149       149             149              149                   149       149
2015.0                116         116       116    116     116         116             116              116                   116       116             116              116                   116       116
2016.0                 62          62        62     62      62          62              62               62                    62        62              62               62                    62        62
```

```
#10 correlation
print(df.corr())
```



```
                      num_critic  duration     gross  num_voted_users  \  num_user_for_reviews    budget  title_year  imdb_score  \  aspect_ratio  movie_likes
num_critic              1.000000  0.231488  0.470083         0.595996  \              0.567703  0.105945    0.409678    0.343005  \      0.180850     0.704879
duration                0.231488  1.000000  0.247746         0.340640  \              0.352318  0.068632   -0.128678    0.365278  \      0.154932     0.219279
gross                   0.470083  0.247746  1.000000         0.628040  \              0.547925  0.100771    0.051597    0.212116  \      0.065662     0.372265
num_voted_users         0.595996  0.340640  0.628040         1.000000  \              0.780364  0.067252    0.021381    0.477156  \      0.085846     0.520735
num_user_for_reviews    0.567703  0.352318  0.547925         0.780364  \              1.000000  0.071559    0.016769    0.322281  \      0.098830     0.371870
budget                  0.105945  0.068632  0.100771         0.067252  \              0.071559  1.000000    0.046365    0.022267  \      0.027981     0.053743
title_year              0.409678 -0.128678  0.051597         0.021381  \              0.016769  0.046365    1.000000   -0.135083  \      0.220741     0.303041
imdb_score              0.343005  0.365278  0.212116         0.477156  \              0.322281  0.022267   -0.135083    1.000000  \      0.026054     0.279273
aspect_ratio            0.180850  0.154932  0.065662         0.085846  \              0.098830  0.027981    0.220741    0.026054  \      1.000000     0.110567
movie_likes             0.704879  0.219279  0.372265         0.520735  \              0.371870  0.053743    0.303041    0.279273  \      0.110567     1.000000
```

```
#11 covariance
print(df.cov())
```

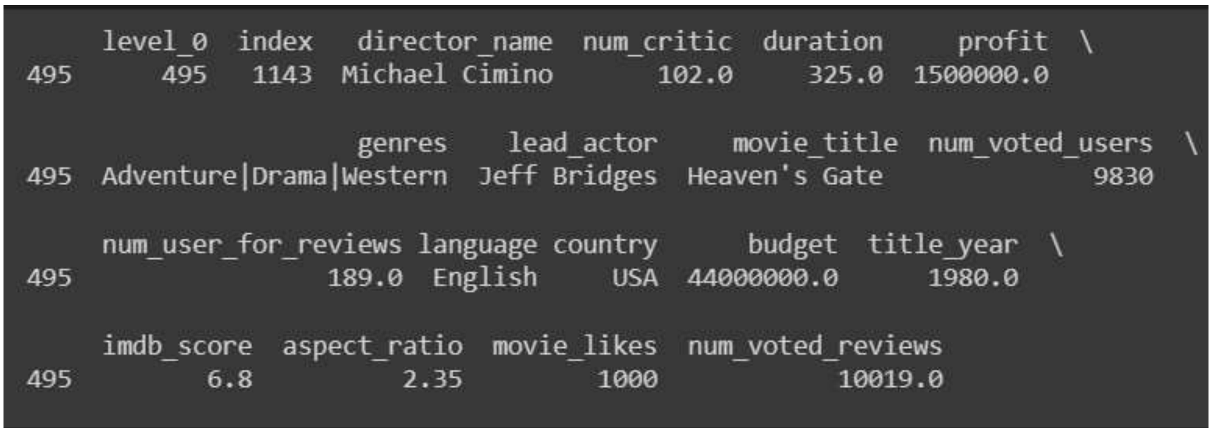


```
                      num_critic      duration         gross  \  num_voted_users  num_user_for_reviews        budget  \  [illegible]
num_critic            [illegible]
duration              [illegible]
gross                 [illegible]
num_voted_users       [illegible]
num_user_for_reviews  [illegible]
budget                [illegible]
title_year            [illegible]
imdb_score            [illegible]
aspect_ratio          [illegible]
movie_likes           [illegible]
```

```
#12 print details of first 10 movies

print(df.head(10))
```

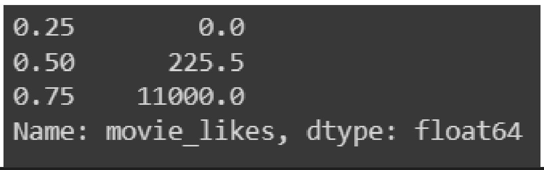
```
[illegible]
```

```python
#13 print details of movies with duration above 300minutes
print(df.loc[df['duration']>300])
```



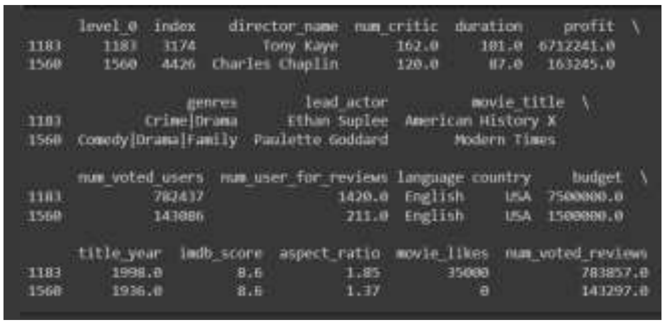
```
     level_0  index   director_name  num_critic  duration     profit  \
495      495   1143  Michael Cimino       102.0     325.0  1500000.0

                       genres    lead_actor     movie_title  num_voted_users  \
495  Adventure|Drama|Western  Jeff Bridges  Heaven's Gate              9830

     num_user_for_reviews language country      budget  title_year  \
495                 189.0  English     USA  44000000.0      1980.0

     imdb_score  aspect_ratio  movie_likes  num_voted_reviews
495         6.8          2.35         1000            10019.0
```

```python
#14 print the quantile of movie likes
print(df['movie likes'].quantile([0.25, 0.5, 0.75]))
```



```
0.25        0.0
0.50      225.5
0.75    11000.0
Name: movie_likes, dtype: float64
```

```python
#15 data preparation

#strip leading and trailing whitespaces if any
df['director_name'].str.strip()

#filter rows based on condition
imdb_above_8 = df[df['imdb_score'] > 8.5]
print(imdb_above_8)

#filter rows based on query
title_year_above_2008 = df.query('title_year > 2008')

#adding a new column
df['num_voted_reviews'] = df['num_voted_users'] +
df['num_user_for_reviews']

#get dummies
dummy countries = pd.get dummies(df['country'])
```



```
      level_0  index    director_name  num_critic  duration     profit  \
1183     1183   3174        Tony Kaye       162.0     101.0  6712241.0
1560     1560   4426  Charles Chaplin       120.0      87.0   163245.0

                   genres         lead_actor          movie_title  \
1183          Crime|Drama       Ethan Suplee  American History X
1560  Comedy|Drama|Family  Paulette Goddard         Modern Times

      num_voted_users  num_user_for_reviews language country     budget  \
1183           782437                1420.0  English     USA  7500000.0
1560           143086                 211.0  English     USA  1500000.0

      title_year  imdb_score  aspect_ratio  movie_likes  num_voted_reviews
1183      1998.0         8.6          1.85        35000           783857.0
1560      1936.0         8.6          1.37            0           143297.0
```

```
#16  data aggregation
#renaming our gross column as profit
df.rename(columns={'gross':'profit'},inplace=True)
df
```

1	1	Gore Verbinski	[illegible]	[illegible]	[illegible]	Action|Adventure|Fantasy	Johnny Depp	Pirates of the Caribbean: At World's End	[illegible]	[illegible]	English	USA	[illegible]	2007.0	7.1	2.35	0	[illegible]
2	2	Sam Mendes	[illegible]	[illegible]	[illegible]	Action|Adventure|Thriller	Christoph Waltz	Spectre	[illegible]	[illegible]	English	UK	[illegible]	2015.0	6.8	2.35	[illegible]	[illegible]
3	3	Christopher Nolan	[illegible]	[illegible]	[illegible]	Action|Thriller	Tom Hardy	The Dark Knight Rises	[illegible]	[illegible]	English	USA	[illegible]	2012.0	8.5	2.35	[illegible]	[illegible]
4	4	Andrew Stanton	[illegible]	[illegible]	[illegible]	Action|Adventure|Sci-Fi	Daryl Sabara	John Carter	[illegible]	[illegible]	English	USA	[illegible]	2012.0	6.6	2.35	[illegible]	[illegible]
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Drama|Fantasy	[illegible]	Pink Narcissus	[illegible]	[illegible]	English	USA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	Daryl Wein	[illegible]	[illegible]	[illegible]	Romance	[illegible]	Breaking Upwards	[illegible]	[illegible]	English	USA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Drama	[illegible]	The Circle	[illegible]	[illegible]	Persian	Iran	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Crime|Horror|Mystery|Thriller	[illegible]	The Cure	[illegible]	[illegible]	Japanese	Japan	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	Shane Carruth	[illegible]	[illegible]	[illegible]	Drama|Sci-Fi|Thriller	Shane Carruth	Primer	[illegible]	[illegible]	English	USA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

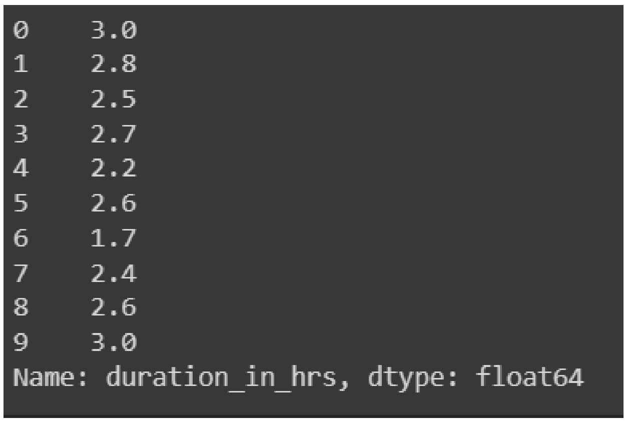
```
#17 Datatype conversion
df['duration'] = df['duration'].astype('float')
print(type(df['duration'][0]))
```

```
<class 'numpy.float64'>
```

```
#18 data wrangling

newdf1 = pd.DataFrame(df[['director_name', 'duration', 'movie_title']])
newdf2 = pd.DataFrame(df[['movie_title', 'title_year', 'imdb_score']])

# merge dataframes
merged_df = pd.merge(newdf1, newdf2)
print(merged_df.head())

#concat dataframes
concatenated_df = pd.concat([newdf1, newdf2], axis=1)
print(concatenated_df.head())
```

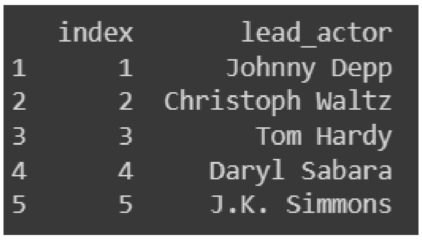
```
      director_name  duration                                 movie_title  \
0     James Cameron     178.0                                     Avatar
1    Gore Verbinski     169.0   Pirates of the Caribbean: At World's End
2        Sam Mendes     148.0                                    Spectre
3 Christopher Nolan     164.0                      The Dark Knight Rises
4    Andrew Stanton     132.0                                John Carter

   title_year  imdb_score
0      2009.0         7.9
1      2007.0         7.1
2      2015.0         6.8
3      2012.0         8.5
4      2012.0         6.6
      director_name  duration                                 movie_title  \
0     James Cameron     178.0                                     Avatar
1    Gore Verbinski     169.0   Pirates of the Caribbean: At World's End
2        Sam Mendes     148.0                                    Spectre
3 Christopher Nolan     164.0                      The Dark Knight Rises
4    Andrew Stanton     132.0                                John Carter

                                 movie_title  title_year  imdb_score
0                                     Avatar      2009.0         7.9
1   Pirates of the Caribbean: At World's End      2007.0         7.1
2                                    Spectre      2015.0         6.8
3                      The Dark Knight Rises      2012.0         8.5
4                                John Carter      2012.0         6.6
```

```python
#19 Data transformation

#convert duration into hours
df['duration_in_hrs'] = round(df['duration']/60, 1)
print(df['duration_in_hrs'].head(10))
```



```
0    3.0
1    2.8
2    2.5
3    2.7
4    2.2
5    2.6
6    1.7
7    2.4
8    2.6
9    3.0
Name: duration_in_hrs, dtype: float64
```

```python
#20 display name of movie and director's name of first 5 movies
selected_Data = df.iloc[[1, 2, 3, 4, 5], [1, 7]]
print(selected_Data)
```



```
   index        lead_actor
1      1       Johnny Depp
2      2  Christoph Waltz
3      3         Tom Hardy
4      4      Daryl Sabara
5      5      J.K. Simmons
```