quirky 2013-01-23 23:29:38 **465561** 138446 55999 **465562** 138446 55999 sad 2013-01-23 23:29:32 rise to power 2007-11-02 21:12:47 **465563** 138472 465564 rows × 4 columns In [15]: tags.shape (465564, 4)Out[15]: In [16]: tags.head() Out[16]: userld movield timestamp tag 4141 Mark Waters 2009-04-24 18:19:40 18 dark hero 2013-05-10 01:41:18 dark hero 2013-05-10 01:41:19 noir thriller 2013-05-10 01:39:43 dark hero 2013-05-10 01:41:18 ratings.head() userld movield rating Out[9]: timestamp 3.5 2005-04-02 23:53:47 3.5 2005-04-02 23:31:16 3.5 2005-04-02 23:33:39 1 3.5 2005-04-02 23:32:07 3.5 2005-04-02 23:29:40 In [17]: del ratings['timestamp'] ratings del tags['timestamp'] tags Out[17]: userld movield tag 18 4141 Mark Waters 208 dark hero 2 353 65 dark hero 65 521 noir thriller 65 592 dark hero **465559** 138446 55999 dragged **465560** 138446 55999 Jason Bateman **465561** 138446 55999 quirky **465562** 138446 55999 **465563** 138472 rise to power 465564 rows × 3 columns Data Structures: Series-Typically refers to a sequence of data elements, often used in data analysis and manipulation tasks. # The .iloc indexer in Python, specifically in the context of the pandas library, is used to select #elements from a DataFrame or Series based on their integer positions (indexes). value_0=ratings.iloc[0] type(value_0) pandas.core.series.Series In [20]: value_0 userId 1.0 Out[20]: movieId rating Name: 0, dtype: float64 In [21]: value_0.index Index(['userId', 'movieId', 'rating'], dtype='object') Out[21]: value_0['userId'] Out[22]: In [23]: **#Conditions** 'rating' in value_0 Out[23]: In [24]: 'tag' in value_0 False In [25]: value_0.name Out[25]: value_0=value_0.rename('0throw') value_0.name 'Othrow' **Data Frames** ratings.head() Out[28]: userld movield rating In [30]: ratings.index RangeIndex(start=0, stop=20000263, step=1) Out[30]: In [31]: ratings.columns Index(['userId', 'movieId', 'rating'], dtype='object') ratings.iloc[[1,10,100,1000]] userld movield rating Out[34]: 1 29 3.5 4.0 10 293 100 1 3037 3.5 1000 11 527 4.5 python a **Descriptive Statistics** with Pandas in Python #.describe() is used to generate descriptive statistics of a DataFrame or Series ratings['rating'].describe() count 2.000026e+07 Out[36]: 3.525529e+00 std 1.051989e+00 min 5.000000e-01 3.000000e+00 50% 3.500000e+00 75% 4.000000e+00 5.000000e+00 max Name: rating, dtype: float64 In [37]: ratings.describe() Out[37]: userld movield rating count 2.000026e+07 2.000026e+07 2.000026e+07 mean 6.904587e+04 9.041567e+03 3.525529e+00 std 4.003863e+04 1.978948e+04 1.051989e+00 min 1.000000e+00 1.000000e+00 5.000000e-01 **25**% 3.439500e+04 9.020000e+02 3.000000e+00 **50%** 6.914100e+04 2.167000e+03 3.500000e+00 **75**% 1.036370e+05 4.770000e+03 4.000000e+00 max 1.384930e+05 1.312620e+05 5.000000e+00 In [41]: ratings.mean() 69045.872583 userId Out[41]: movieId 9041.567330 rating 3.525529 dtype: float64 In [42]: ratings.min() userId 1.0 Out[42]: movieId 1.0 rating 0.5 dtype: float64 ratings.max() 138493.0 userId Out[45]: movieId 131262.0 rating dtype: float64 In [43]: ratings.count() userId 20000263 Out[43]: 20000263 movieId 20000263 rating dtype: int64 In [46]: ratings.std() userId 40038.626653 Out[46]: movieId 19789.477445 1.051989 rating dtype: float64 In [47]: # First quartile ratings.quantile(0.25) userId 34395.0 Out[47]: movieId 902.0 rating 3.0 Name: 0.25, dtype: float64 In [48]: #Second quartile ratings.quantile(0.50) 69141.0 userId Out[48]: 2167.0 movieId rating 3.5 Name: 0.5, dtype: float64 In [49]: # Third quartile ratings.quantile(0.75) userId 103637.0 movieId 4770.0 rating 4.0 Name: 0.75, dtype: float64 In [50]: # The mode is the value that appears most frequently in the data. If there are multiple values with the #same highest frequency, all of them will be returned as the mode. ratings.mode() userId movieId rating Out[50]: **0** 118205 296 4.0 In [51]: # corr() :This function is used to calculate the correlation between columns of a DataFrame. ratings.corr() rating Out[51]: userld movield **userId** 1.000000 -0.000850 0.001175 movield -0.000850 1.000000 0.002606 rating 0.001175 0.002606 1.000000 In [54]: # Filtering Data f1=ratings['rating'] > 20 print(f1) f1.any() 0 False 1 False False False False . . . 20000258 False 20000259 False 20000260 20000261 False 20000262 False Name: rating, Length: 20000263, dtype: bool False Out[54]: In [55]: f2=ratings['rating'] > 0

MDB Movie

Final Project-1

movies=pd.read_csv(r"D:\DS\Practice\Raw Data\archive\movie.csv" , sep=',')

Adventure|Animation|Children|Comedy|Fantasy

Adventure|Children|Fantasy

Comedy|Drama|Romance

Comedy|Romance

genres

Comedy

Comedy

Comedy

Adventure

(no genres listed)

Adventure|Fantasy|Horror

title

Toy Story (1995)

Jumanji (1995)

Grumpier Old Men (1995)

Waiting to Exhale (1995)

Kein Bund für's Leben (2007)

Feuer, Eis & Dosenbier (2002)

The Pirates (2014)

Innocence (2014)

ratings=pd.read_csv(r"D:\DS\Practice\Raw Data\archive\rating.csv")

timestamp

3.5 2005-04-02 23:53:47

3.5 2005-04-02 23:31:16 3.5 2005-04-02 23:33:39

3.5 2005-04-02 23:32:07 3.5 2005-04-02 23:29:40

4.5 2009-11-13 15:42:00

4.5 2009-12-03 18:31:48 3.0 2009-12-07 18:10:57

5.0 2009-11-13 15:42:24

2.5 2009-10-17 20:25:36

tags=pd.read_csv(r"D:\DS\Practice\Raw Data\archive\tag.csv")

Mark Waters 2009-04-24 18:19:40

dark hero 2013-05-10 01:41:18

dark hero 2013-05-10 01:41:19

noir thriller 2013-05-10 01:39:43

dark hero 2013-05-10 01:41:18

dragged 2013-01-23 23:29:32

Jason Bateman 2013-01-23 23:29:38

timestamp

tag

Rentun Ruusu (2001)

5 Father of the Bride Part II (1995)

Import Libraries

import pandas as pd

Read Datasets

In [4]:

In [5]:

Out[5]:

movies

4

27273

27274

27275

27276

27277

movield

131254

131256

131258

131260

131262

27278 rows × 3 columns

userld movield rating

47

68954

69526

69644

70286

71619

1

movies.shape

(27278, 3)

ratings

20000258 138493

20000259 138493

20000260 138493

20000261 138493

20000262 138493

ratings.shape

(20000263, 4)

tags

4

465559 138446

465560 138446

20000263 rows × 4 columns

userId movieId

4141

208

353

521

592

55999

55999

18

65

65

65

65

Out[21]:

Out[6]:

In [24]:

Out[24]:

In [14]:

Out[14]:

3

True Out[55]: True True True True 20000258 20000259 True 20000260 True 20000261 True 20000262 True Name: rating, Length: 20000263, dtype: bool Data Cleaning: Handling Missing Data movies.shape (27278, 3)Out[57]: In [58]: #.isnull().any().any() will return True if there is at least one missing value anywhere in the movies #DataFrame, and False otherwise. It's a quick way to check if any missing values are present in the #DataFrame. movies.isnull().any().any() False Out[58]: In [61]: ratings.shape (20000263, 3) Out[61]: In [59]: ratings.isnull().any().any() Out[59]: In [62]: tags.shape (465564, 3) Out[62]: In [60]: tags.isnull().any().any()# Tags Ds has Some Missed Values in it True Out[60]: In [63]: #.dropna(): This method is used to drop rows containing missing values (NaN) from the DataFrame. #When you call tags.dropna(), a new DataFrame is returned where all rows with any NaN values are removed. tags=tags.dropna() In [64]: tags.shape (465548, 3) Out[64] In [65]: tags.isnull().any().any()# Now there were no null values in tags D/S False Out[65]: Data Visualization in Python using matpl & tlib

In [69]:

In [86]:

import matplotlib.pyplot as plt

plt.title('Histogram')

Text(0.5, 1.0, 'Histogram')

1e6

5

4

3

2

1

3

2

Out[74]:

In [75]:

Out[75]:

Out[76]:

In [81]:

Out[81]:

In [89]:

Out[89]:

Out[87]:

plt.title('Box Plot')

Text(0.5, 1.0, 'Box Plot')

Slicing Columns

Mark Waters

dark hero dark hero noir thriller dark hero Name: tag, dtype: object

Jumanji (1995)

movies[10:]# Slicing from 10th row

American President, The (1995)

Nixon (1995)

Cutthroat Island (1995)

The Pirates (2014)

Innocence (2014)

Rentun Ruusu (2001)

tc[:20].plot(kind='bar', color=colors, figsize=(10,5))

Kein Bund für's Leben (2007)

Feuer, Eis & Dosenbier (2002)

12 Dracula: Dead and Loving It (1995)

Grumpier Old Men (1995)

Waiting to Exhale (1995)

4 Father of the Bride Part II (1995)

movield

13

14

15

131254

131258

131260

27268 rows × 3 columns

tc=tags['tag'].value_counts()

genetically modified monsters

Name: tag, dtype: int64

plt.title('Bar Plot')

Text(0.5, 1.0, 'Bar Plot')

27274 131256

27277 131262

tc[-10:]

Ron Moore Citizen Kane

mullet biker gang Paul Adelstein

the wig killer fish

3500

3000

2500

2000

1500

1000

500

In [92]:

Out[92]:

198

201

208

215

220

221 231

232

233

235

239

242

244

245

246

247

248

252

253

254

In [94]:

Out[94]:

based on a

#Neumarical Data.

ratings[hr][20:40]

2

2

2

2

2

2

2

2

3

3

3

3

3

3

userld movield rating

1544

1748

1974

2948

3450

3513

3927

3928

3930

3959

50

175

223

260

316

318

329

457

480

490

For Categorical Data

movies[comedy][10:30]

movield

21

38

39

45

52

54

64

65

68

69

70

72

75

82

84

87

88

93

20

37

38 44

51 53

57

62

63

64

67

68

69

71

74

81

83

86

87

91

Filters for Selecting Rows

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

5.0

#pattern exists within a Series of strings.

comedy=movies['genres'].str.contains('Comedy')

63 Don't Be a Menace to South Central While Drink...

hr = ratings['rating'] >= 5.0

missing child

topless scene

12

13

14

27273

27275

27276

ratings.hist(column='rating', figsize=(10,5)) # Creating A Histogram by using Matplotlib

Histogram

ratings.boxplot(column='rating', figsize=(10,5))# Creating a Box Plot By using Matplotlib

Box Plot

rating

tags['tag'].head()# To get the top 5 rows of a particular attribute

Toy Story (1995) Adventure|Animation|Children|Comedy|Fantasy

movies[['title', 'genres']].head()# to get top 5 rows of the selected attributes

Adventure|Children|Fantasy

Comedy|Drama|Romance

Comedy|Drama|Romance

Action|Adventure|Romance

Balto (1995) Adventure|Animation|Children

Comedy|Romance

Comedy

genres

Drama

Comedy

Comedy

Adventure

(no genres listed)

#value_counts() function calculates the frequency count of each unique value in the series.

colors=['red','orange','yellow','green','blue','indigo','violet','purple','pink','brown']

twist ending

Bar Plot

Here we applied 2 conditions to get values 1.is the rating must be >= 5.0 and 2 is rows in between 20-40

#.str.contains() function is a method provided by pandas that is used to check if a substring or

genres

Comedy|Crime|Thriller

Children|Comedy

Comedy|Romance

Children|Comedy

Comedy|Crime

Comedy

Comedy

Comedy|Romance

Comedy|Romance

Comedy|Drama

Comedy|Drama

Comedy|Drama

Comedy|Drama

Children|Comedy

Comedy|Horror|Romance

Comedy

Comedy|Drama|Thriller

Comedy|Drama|Romance

Comedy|Drama|Romance

title

Get Shorty (1995)

It Takes Two (1995)

Clueless (1995)

To Die For (1995)

Mighty Aphrodite (1995)

Big Green, The (1995)

Two if by Sea (1996)

Bio-Dome (1996)

Friday (1995)

Big Bully (1996)

From Dusk Till Dawn (1996) Action|Comedy|Horror|Thriller

Postman, The (Postino, II) (1994)

French Twist (Gazon maudit) (1995)

Kicking and Screaming (1995)

Antonia's Line (Antonia) (1995)

Dunston Checks In (1996)

Vampire in Brooklyn (1995)

Black Sheep (1996)

Last Summer in the Hamptons (1995)

travel

psychology

visually appealing

Adventure|Fantasy|Horror

Comedy|Horror