Python Project documentation

Jan 27, 2024

Project: Netflix Dataset

Goal: Exploratory Data Analysis and Data Manipulation

This project primarily focuses on exploratory data analysis (EDA).
Key Learnings:

- Creating new columns and DataFrames
- Filtering data (single column and multiple columns)
- Applying filters using AND and OR conditions
- Utilizing the Seaborn library for creating bar graphs and histograms

Handling Duplicate values:

# Finding duplicate values
- data[data.duplicated()]

# REMOVING DUPLICATE VALUES
- data.drop_duplicates(inplace=True)

Finding and handling null values

# Count of total null values in each column
- data.isnull().sum()

# or showing null values using heatmap
- plt.figure(figsize=(5,3))
  sns.heatmap(data.isnull(), yticklabels=False, cmap="magma", cbar=False)

# dropping null values from cast column
- data['Cast'].isnull().value_counts() *# Count of null values*
- data_cast = data['Cast'].dropna() *# dropping all the null values from Cast column*

Converting object data type to datetime/numeric:

# Release_date column is of object type, We'll convert them into datetime type
- data['Release_Date'] = pd.to_datetime(data['Release_Date'], format='mixed', yearfirst=True)
    - By using *format='mixed'* along with *yearfirst=True*, pandas will try to infer the format for each element individually & we won't receive an error for any incorrect date format

# Converting 'Minutes' *object dtype* into *numeric*
- data['Minutes']= pd.to_numeric(data['Minutes'])

Groupby:

# Group all the unqique items of a column and then count them all

- data.groupby('Category').Category.count()

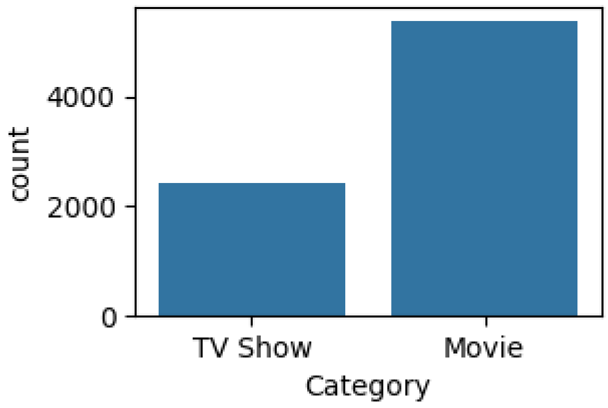
# Counting the movie count by each country

- data_tvshow.groupby('Country')['Title'].count().sort_values(ascending= False)

# Used seaborn lib to plot the categorical count of Movie and TV Shows

- plt.figure(figsize=(3,2))
  sns.countplot(data, x='Category')

# countplot used To show the count of all unique values of any column in the form of bar graph.



Filtering:

1. **.isin()**
   - used to filter rows based on whether a particular column's values are present in a specified list or another DataFrame.
     Eg: Retrieving info about particular element of a column
     # For 'House of cards' what is the Show id and who is the Director of this show?
     - selected_columns=['Title','Show_Id', 'Director']
       data[data["Title"].isin(['House of Cards'])][selected_columns]

2. **.str.contains**
   - versatile method for string matching and filtering
     data[data["Title"].str.contains('House of Cards')][selected_columns]

<u>Note</u>: **.str.contains**
Cannot mask with a non-boolean array containing NA / NaN values. Therefore, we first identify the number of null values in the column and then determine the treatment based on the analysis requirements.

3. **# Filtering based on two conditions**
    - data[(data['Category'] == 'Movie') & (data['Release_Date'].dt.year==2000)]
    - data[(data['Category'] == 'Movie') & (data['Type'] == 'Comedies') | (data['Country'] == 'United Kingdom')]

Counting unique values of a series

# unique() - It shows the all unique values of the series.

- data['Rating'].unique()

# nunique() - It shows the total no. of unique values in the series.

- data['Rating'].nunique()

Splitting columns into two using str.split

# split the Duration column into two ' Minutes and Units

- data[['Minutes', 'Unit']] = data.Duration.str.split(' ', expand=True)
  # expand will return the two different columns

Splitting the coma-separated values of a column and creating new rows :"USA, Poland, India"'

# Split the Country column, Reassigned the 'Country' column with the newly created list and explode it to create new rows

- data_tvshow=data_tvshow.assign(Country=data_tvshow['Country'].str.split(',')).explode('Country')
    - data['Country'].str.split(','): Splits the comma-separated string in the 'Country' column into a list of countries.
    - assign(Country=...): Reassigns the 'Country' column with the newly created list.
    - explode('Country'): Creates a new row for each element in the list, effectively expanding the DataFrame.

Locating records of an element based on some condition for eg min/max

# find the movie with the maximum duration

# Finding the index of the movie with max duration

- max_duration_index = data['Minutes'].idxmax()
  ***idxmax()** method to find the index of the maximum value in the 'Minutes' column*

# Retriving the movie title with max duration

- movie_with_max_duration = data.loc[max_duration_index, 'Title']
  *and then use that index to access the corresponding movie from the 'Title' column using **.loc***