Pandas-GroupBy

Monday, October 9, 2023 12:42 PM

The groupby is one of the most frequently used Pandas functions in data analysis. It is used for grouping the data points (i.e. rows) based on the distinct values in the given column or columns.

```
import pandas as pd
import numpy as np

movies = pd.read_csv('/content/imdb-top-1000.csv')
movies.head()

genres = movies.groupby('Genre')

# Applying builtin aggregation fuctions on groupby objects
genres.std()

# find the top 3 genres by total earning
movies.groupby('Genre').sum()['Gross'].sort_values(ascending=False).head(3)
movies.groupby('Genre')['Gross'].sum().sort_values(ascending=False).head(3)

# find the genre with highest avg IMDB rating
movies.groupby('Genre')['IMDB_Rating'].mean().sort_values(ascending=False).head(1)

# find director with most popularity
movies.groupby('Director')['No_of_Votes'].sum().sort_values(ascending=False).head(1)

# find the highest rated movie of each genre
movies.groupby('Genre')['IMDB_Rating'].max()

# find number of movies done by each actor
movies['Star1'].value_counts()

movies.groupby('Star1')['Series_Title'].count().sort_values(ascending=False)

# GroupBy Attributes and Methods
# find total number of groups -> len
# find items in each group -> size
# first()/last() -> nth item
# get_group -> vs filtering
# groups
# describe
# sample
# nunique

len(movies.groupby('Genre'))
movies['Genre'].nunique()
movies.groupby('Genre').size()

genres = movies.groupby('Genre')

# genres.first()
# genres.last()
genres.nth(6)

movies['Genre'].value_counts()
genres.get_group('Fantasy')
movies[movies['Genre'] == 'Fantasy']

genres.groups
genres.describe()
genres.sample(2,replace=True)
genres.nunique()
```

agg method

```
# passing dict
genres.agg(
  {
    'Runtime':'mean',
```

```
    'IMDB_Rating':'mean',
    'No_of_Votes':'sum',
    'Gross':'sum',
    'Metascore':'min'
  }
)

# passing list
genres.agg(['min','max','mean','sum'])

# Adding both the syntax
genres.agg(
  {
    'Runtime':['min','mean'],
    'IMDB_Rating':'mean',
    'No_of_Votes':['sum','max'],
    'Gross':'sum',
    'Metascore':'min'
  }
)

# looping on groups
df = pd.DataFrame(columns=movies.columns)
for group,data in genres:
  df = df.append(data[data['IMDB_Rating'] == data['IMDB_Rating'].max()])
df

# split (apply) combine
# apply -> builtin function

genres.apply(min)

# find number of movies starting with A for each group

def foo(group):
  return group['Series_Title'].str.startswith('A').sum()
  genres.apply(foo)

# find ranking of each movie in the group according to IMDB score

def rank_movie(group):
  group['genre_rank'] = group['IMDB_Rating'].rank(ascending=False)
  return group
genres.apply(rank_movie)

# find normalized IMDB rating group wise

def normal(group):
  group['norm_rating'] = (group['IMDB_Rating'] - group['IMDB_Rating'].min())/(group['IMDB_Rating'].max() - group['IMDB_Rating'].min())
  return group
genres.apply(normal)

# groupby on multiple cols
duo = movies.groupby(['Director','Star1'])
duo
# size
duo.size()
# get_group
duo.get_group(('Aamir Khan','Amole Gupte'))

# find the most earning actor->director combo
duo['Gross'].sum().sort_values(ascending=False).head(1)

# find the best(in-terms of metascore(avg)) actor->genre combo
movies.groupby(['Star1','Genre'])['Metascore'].mean().reset_index().sort_values('Metascore',ascending=False).head(1)

# agg on multiple groupby
duo.agg(['min','max','mean'])
```

Excercise

```
ipl = pd.read_csv('/content/deliveries.csv')
ipl.head()
ipl.shape

# find the top 10 batsman in terms of runs
ipl.groupby('batsman')['batsman_runs'].sum().sort_values(ascending=False).head(10)

# find the batsman with max no of sixes
six = ipl[ipl['batsman_runs'] == 6]

six.groupby('batsman')['batsman'].count().sort_values(ascending=False).head(1).index[0]

# find batsman with most number of 4's and 6's in last 5 overs
temp_df = ipl[ipl['over'] > 15]
temp_df = temp_df[(temp_df['batsman_runs'] == 4) | (temp_df['batsman_runs'] == 6)]
temp_df.groupby('batsman')['batsman'].count().sort_values(ascending=False).head(1).index[0]

# find V Kohli's record against all teams
temp_df = ipl[ipl['batsman'] == 'V Kohli']

temp_df.groupby('bowling_team')['batsman_runs'].sum().reset_index()

# Create a function that can return the highest score of any batsman
temp_df = ipl[ipl['batsman'] == 'V Kohli']
temp_df.groupby('match_id')['batsman_runs'].sum().sort_values(ascending=False).head(1).values[0]

def highest(batsman):
  temp_df = ipl[ipl['batsman'] == batsman]
  return temp_df.groupby('match_id')['batsman_runs'].sum().sort_values(ascending=False).head(1).values[0]


highest('DA Warner')
```