# PROGRAM NO: 4.1

**AIM :** Programs to handle data using pandas.

**DATE:** 2.09.2022

# SOURCE CODE :

1. import pandas as pd

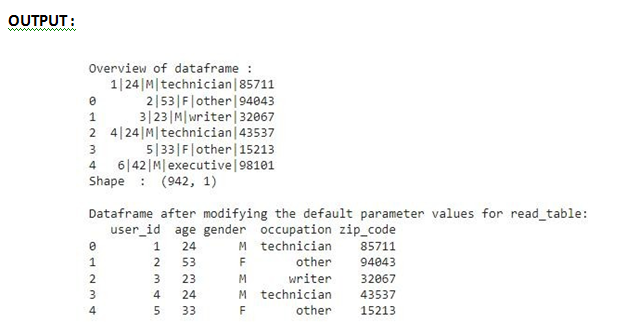
orders = pd.read\_table('[http://bit.ly/movieusers')](http://bit.ly/movieusers%27)) print("Overview of dataframe : ") print(orders.head())

print("Shape : ",orders.shape) print()

user\_cols = ['user\_id', 'age', 'gender', 'occupation', 'zip\_code']

users = pd.read\_table('[http://bit.ly/movieusers',](http://bit.ly/movieusers%27) sep='|', header=None, names=user\_cols)

print("Dataframe after modifying the default parameter values for read\_table: ") print(users.head())

****

1. import pandas as pd **#read a csv file**

ufo = pd.read\_csv('[http://bit.ly/uforeports')](http://bit.ly/uforeports%27)) print("Overview of UFO data reports: ") print(ufo.head())

print()

**#series**

**print("Cityseries(sorted):")** print(ufo.City.sort\_values()) print()

**ufo['Location'] = ufo.City + ', ' + ufo.State** print("After creating a new 'Location' Series : ") print(ufo.head())

print()

print("Calculate summary statistics : ") **print(ufo.describe())**

print()

print("Column names of ufo dataframe : ",ufo.columns) print()

# rename two of the columns by using the 'rename' method **ufo.rename(columns={'Colors Reported':'Colors\_Reported', 'Shape Reported':'Shape\_Reported'},inplace=True)**

print("Column name of ufo dataframe after renaming two column names : ",ufo.columns)

print()

**# remove multiple columns at once** ufo.drop(['City', 'State'], axis=1, inplace=True)

print("Column name of ufo dataframe after removing two columns(city,state) : ",ufo.columns)

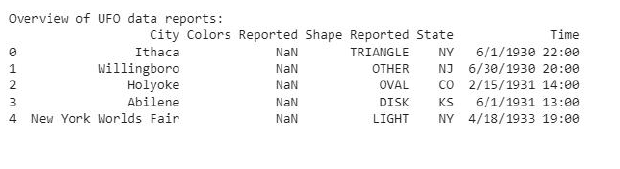
print()

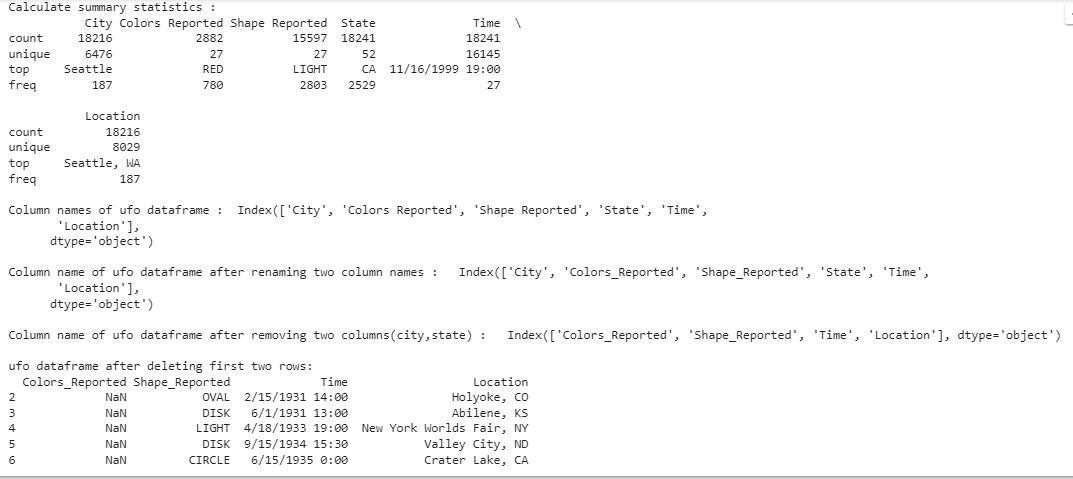
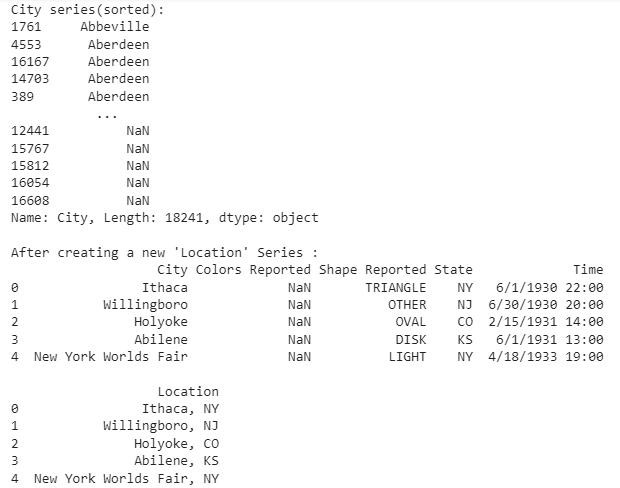
**# remove multiple rows at once (axis=0 refers to rows)**

ufo.drop([0, 1], axis=0, inplace=True)

print("ufo dataframe after deleting first two rows: ") print(ufo.head())

# OUTPUT :





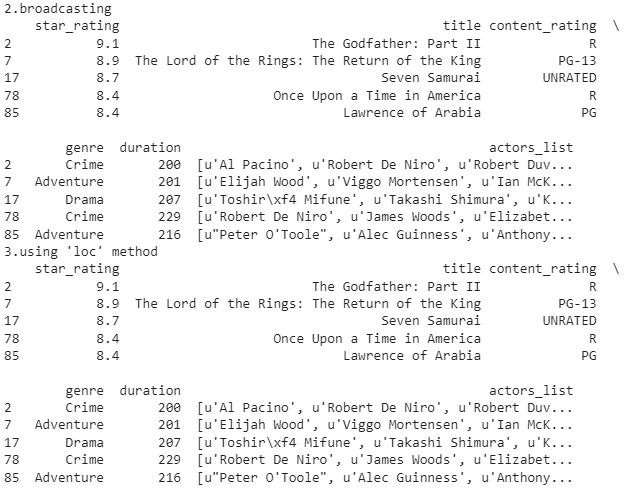
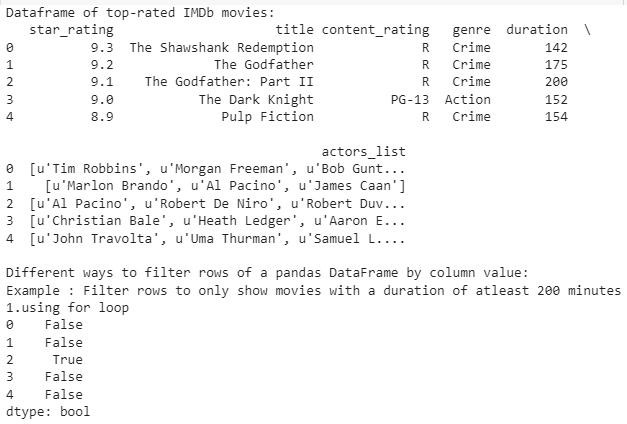
1. import pandas as pd

# read a dataset of top-rated IMDb movies into a DataFrame movies = pd.read\_csv('[http://bit.ly/imdbratings')](http://bit.ly/imdbratings%27)) print("Dataframe of top-rated IMDb movies: ") print(movies.head())

print()

print("**Different ways to filter rows of a pandas DataFrame by column value**: ") print("Example : Filter rows to only show movies with a duration of atleast 200 minutes") print("1.using for loop")

**# create a list in which each element refers to a DataFrame row: True if the row satisfies the condition,False otherwise**



booleans = []

for length in movies.duration:

if length >= 200:

**booleans.append(True)**

else:

**booleans.append(False)**

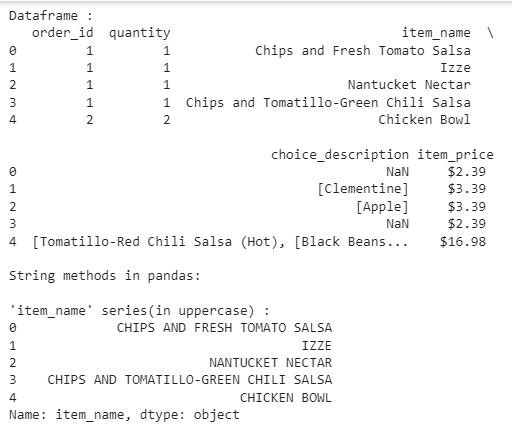
is\_long = pd.Series(booleans) print(is\_long.head())

print() print("2.broadcasting")

print(movies[movies.duration >= 200]) print()

print("3.using 'loc' method") print(movies.loc[movies.duration >= 200])

# OUTPUT :



1. import pandas as pd

# read a dataset of Chipotle orders into a DataFrame orders = pd.read\_table('[http://bit.ly/chiporders')](http://bit.ly/chiporders%27)) print("Dataframe : ")

print(orders.head()) print()

**print("String methods in pandas: ")** print()

print("'item\_name' series(in uppercase) : ") print(orders.item\_name.**str.upper().**head()) print()

print("Checks for a substring 'Chicken' in the given dataframe: ") print(orders[orders.item\_name.**str.contains('Chicken'**)].head()) print()

# many pandas string methods support regular expressions (regex) print(orders.choice\_description.**str.replace('[\[\]]', '').**head()) print()

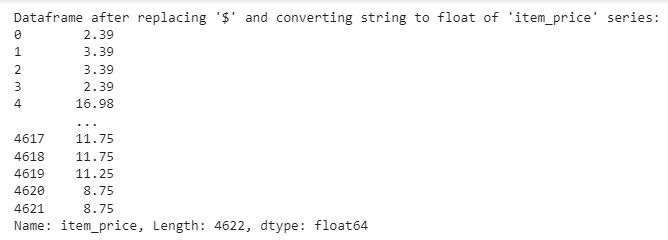
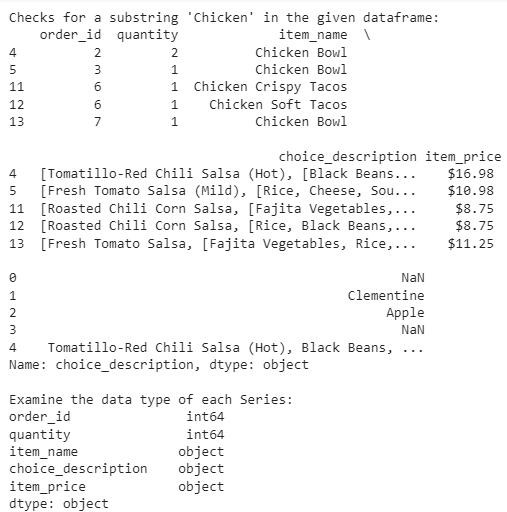
print("Examine the data type of each Series: ") print(**orders.dtypes)**

print()

print("Dataframe after replacing '$' and converting string to float of 'item\_price' series: ") print(orders.item\_price.**str.replace('$', '').astype(float))**

print()

# OUTPUT :



1. import pandas as pd

#read a dataset of alcohol consumption into a DataFrame drinks = pd.read\_csv('[http://bit.ly/drinksbycountry')](http://bit.ly/drinksbycountry%27)) print("Dataframe : ")

print(drinks.head()) print()

print("Mean beer servings across the entire dataset: ",drinks.beer\_servings.**mean())** print("Mean beer servings just for countries in Africa: ",drinks[**drinks.continent=='Africa'].beer\_servings.mean(**))

print()

print("Aggregate functions used with groupby: ") print()

print("Mean beer servings for each continent: ",drinks.groupby('continent').beer\_servings.mean())

print("Maximum beer servings for each continent: ",drinks.groupby('continent').beer\_servings.max())

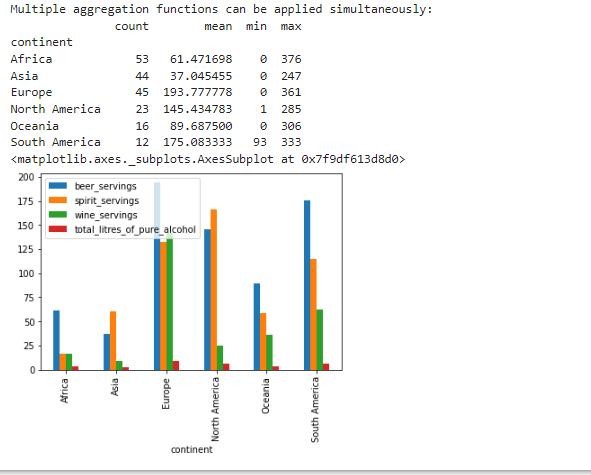
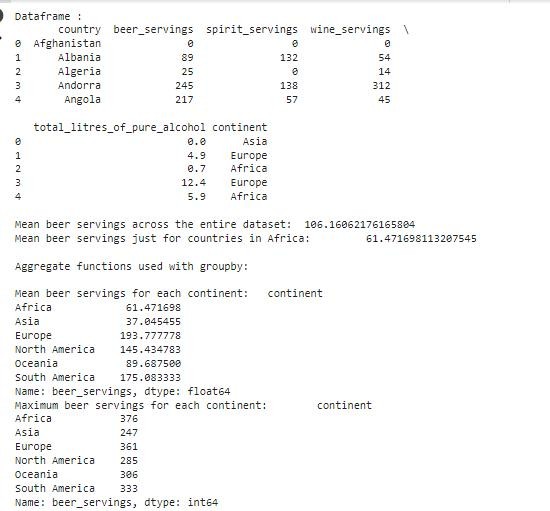
print("Multiple aggregation functions can be applied simultaneously: ") print(drinks.groupby('continent').beer\_servings.agg(['count', 'mean', 'min', 'max']))

# specifying a column to which the aggregation function should be applied is not required

drinks.groupby('continent').mean()

# allow plots to appear in the notebook

%matplotlib inline



# side-by-side bar plot of the DataFrame directly above drinks.groupby('continent').mean().plot(kind='bar')

# OUTPUT :

1. import pandas as pd

ufo = pd.read\_csv('[http://bit.ly/uforeports')](http://bit.ly/uforeports%27)) print(ufo.isnull().tail()) print(ufo.notnull().tail()) print(ufo.isnull().sum())

print(ufo.shape)

# if 'all' values are missing in a row, then drop that row (none are dropped in this case) print(ufo.dropna(how='all').shape)

print(ufo.dropna(subset=['City', 'Shape Reported'], how='any').shape) print(ufo['Shape Reported'].value\_counts().head())

# fill in missing values with a specified value

print(ufo['Shape Reported'].fillna(value='VARIOUS', inplace=True)) # confirm that the missing values were filled in

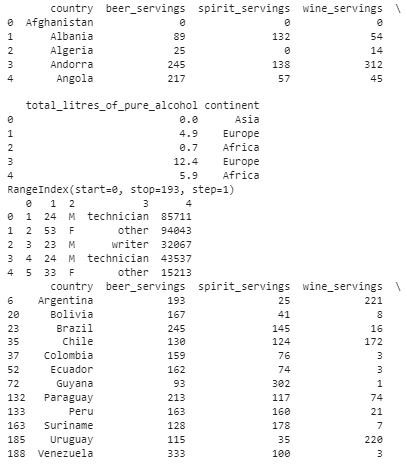
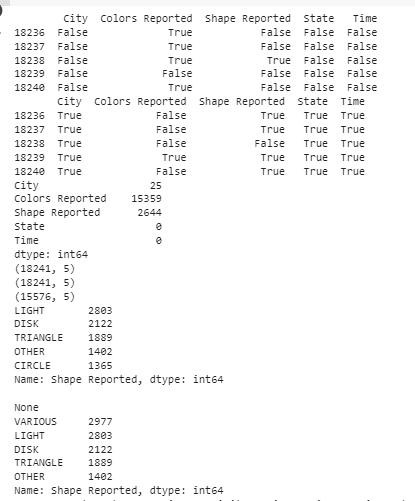
print(ufo['Shape Reported'].value\_counts().head()) drinks = pd.read\_csv('[http://bit.ly/drinksbycountry')](http://bit.ly/drinksbycountry%27)) print(drinks.head())

# every DataFrame has an index (sometimes called the "row labels") print(drinks.index)

# index and columns both default to integers if you don't define them print(pd.read\_table('[http://bit.ly/movieusers',](http://bit.ly/movieusers%27) header=None, sep='|').head()) # identification: index remains with each row when filtering the DataFrame print(drinks[drinks.continent=='South America'])

# selection: select a portion of the DataFrame using the index print(drinks.loc[23, 'beer\_servings'])

# set an existing column as the index print(drinks.set\_index('country', inplace=True)) print(drinks.head())



# you can interact with any DataFrame using its index and columns print(drinks.describe().loc['25%', 'beer\_servings'])

# access the Series index print(drinks.continent.value\_counts().index) # access the Series values print(drinks.continent.value\_counts().values) # any Series can be sorted by its values

print(drinks.continent.value\_counts().sort\_values())

people = pd.Series([3000000, 85000], index=['Albania', 'Andorra'], name='population') # concatenate the 'drinks' DataFrame with the 'population' Series (aligns by the index) print(pd.concat([drinks, people], axis=1).head())

# OUTPUT :

