PROGRAM NO: 4.1
AIM : Programs to handle data using pandas.

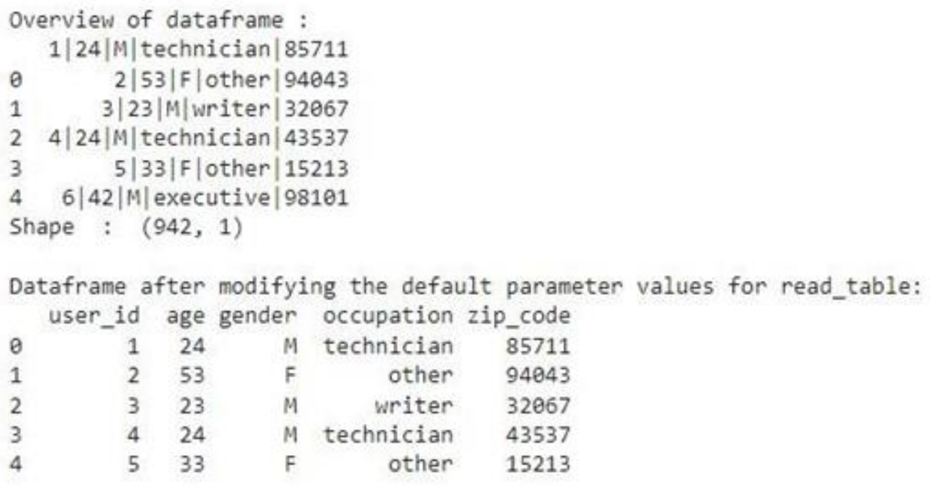
DATE: 2.09.2022

SOURCE CODE :

1.

```
import pandas as pd
orders = pd.read_table('http://bit.ly/movieusers')
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', sep='|', header=None,
names=user_cols)
print("Dataframe after modifying the default parameter values for read_table: ")
print(users.head())
```

OUTPUT :



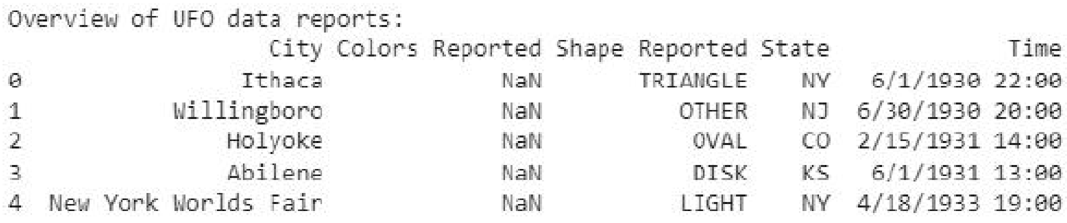
```
Overview of dataframe :
   1|24|M|technician|85711
0       2|53|F|other|94043
1      3|23|M|writer|32067
2  4|24|M|technician|43537
3       5|33|F|other|15213
4   6|42|M|executive|98101
Shape  :  (942, 1)

Dataframe after modifying the default parameter values for read_table:
   user_id  age gender  occupation zip_code
0        1   24      M  technician    85711
1        2   53      F       other    94043
2        3   23      M      writer    32067
3        4   24      M  technician    43537
4        5   33      F       other    15213
```

2.

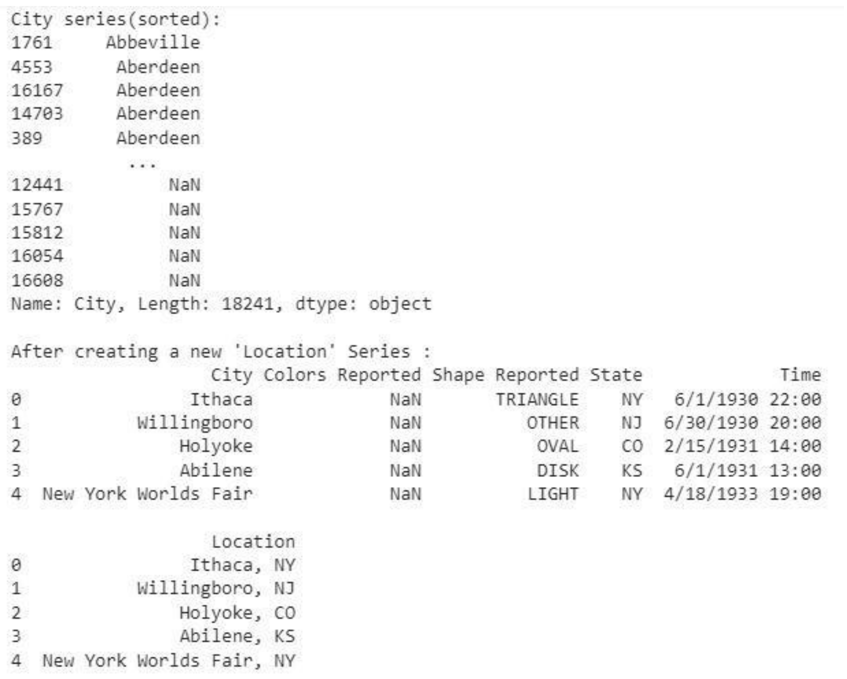
```
import pandas as pd
#read a csv file
ufo = pd.read_csv('http://bit.ly/uforeports')
print("Overview of UFO data reports: ")
print(ufo.head())
print()
```

```
#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',
 'ShapeReported':'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())
```

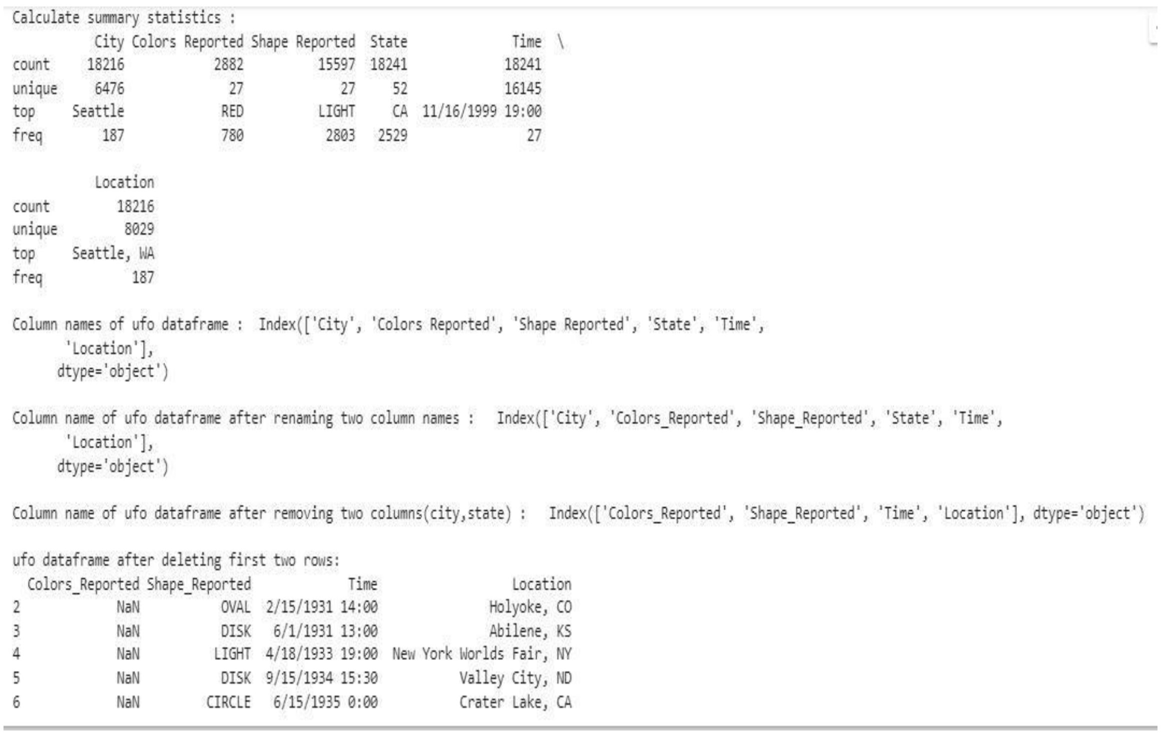
OUTPUT :



```
Overview of UFO data reports:
                 City Colors Reported Shape Reported State             Time
0              Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1         Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2             Holyoke             NaN           OVAL    CO  2/15/1931 14:00
3             Abilene             NaN           DISK    KS   6/1/1931 13:00
4  New York Worlds Fair           NaN          LIGHT    NY  4/18/1933 19:00
```



```
City series(sorted):
1761     Abbeville
4553      Aberdeen
16167     Aberdeen
14703     Aberdeen
389       Aberdeen
            ...
12441          NaN
15767          NaN
15812          NaN
16054          NaN
16608          NaN
Name: City, Length: 18241, dtype: object

After creating a new 'Location' Series :
                   City Colors Reported Shape Reported State             Time
0                Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1           Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2               Holyoke             NaN           OVAL    CO  2/15/1931 14:00
3               Abilene             NaN           DISK    KS   6/1/1931 13:00
4  New York Worlds Fair             NaN          LIGHT    NY  4/18/1933 19:00

                   Location
0                Ithaca, NY
1           Willingboro, NJ
2               Holyoke, CO
3               Abilene, KS
4  New York Worlds Fair, NY
```

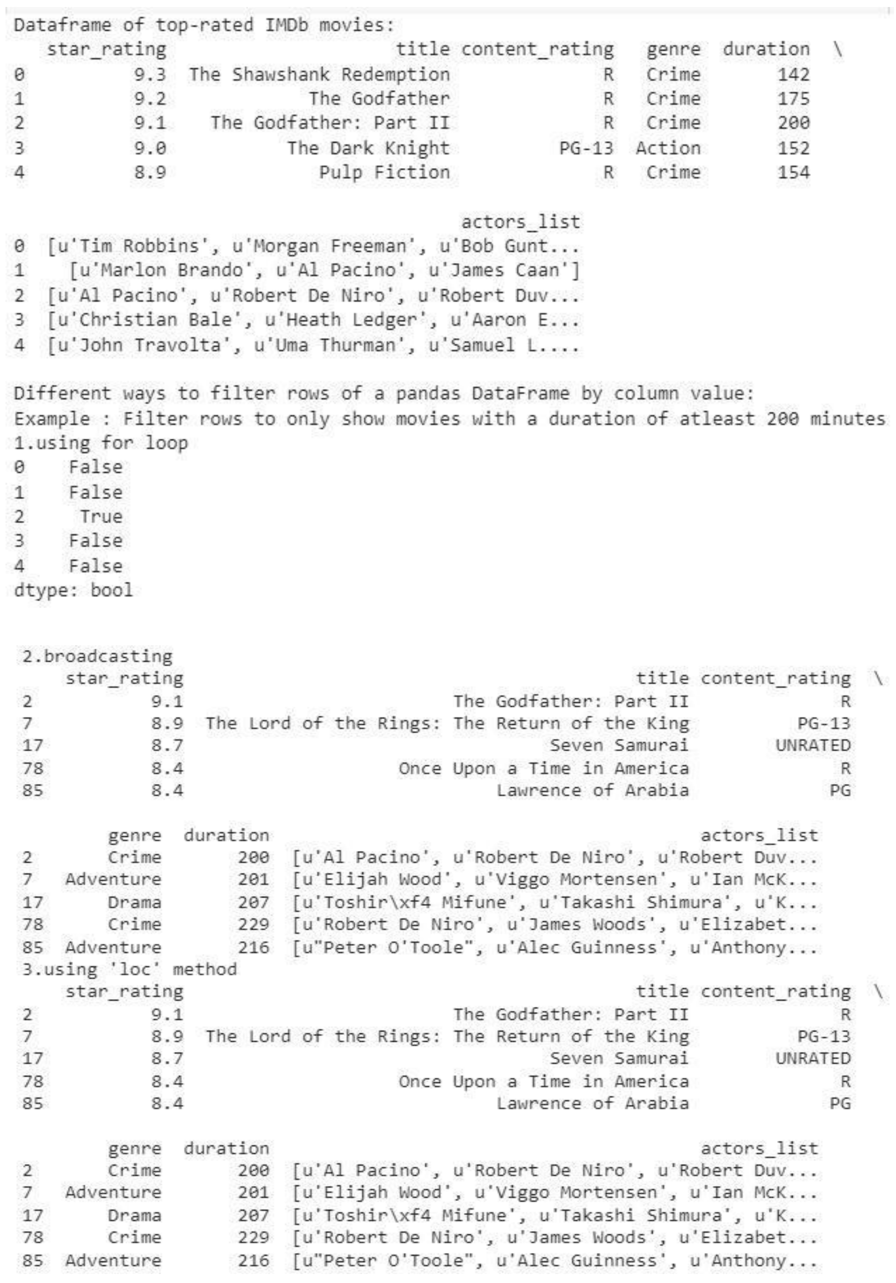


```
Calculate summary statistics :
           City Colors Reported Shape Reported  State             Time  \
count     18216            2882          15597  18241            18241
unique     6476              27             27     52            16145
top     Seattle             RED          LIGHT     CA  11/16/1999 19:00
freq        187             780           2803   2529               27

           Location
count         18216
unique         8029
top     Seattle, WA
freq            187

Column names of ufo dataframe :  Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time',
       'Location'],
      dtype='object')

Column name of ufo dataframe after renaming two column names :   Index(['City', 'Colors_Reported', 'Shape_Reported', 'State', 'Time',
       'Location'],
      dtype='object')

Column name of ufo dataframe after removing two columns(city,state) :   Index(['Colors_Reported', 'Shape_Reported', 'Time', 'Location'], dtype='object')

ufo dataframe after deleting first two rows:
  Colors_Reported Shape_Reported             Time                  Location
2             NaN           OVAL  2/15/1931 14:00               Holyoke, CO
3             NaN           DISK   6/1/1931 13:00               Abilene, KS
4             NaN          LIGHT  4/18/1933 19:00  New York Worlds Fair, NY
5             NaN           DISK  9/15/1934 15:30           Valley City, ND
6             NaN         CIRCLE   6/15/1935 0:00            Crater Lake, CA
```

3. import pandas as pd
# read a dataset of top-rated IMDb movies into a DataFrame
movies = pd.read_csv('http://bit.ly/imdbratings')
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 :

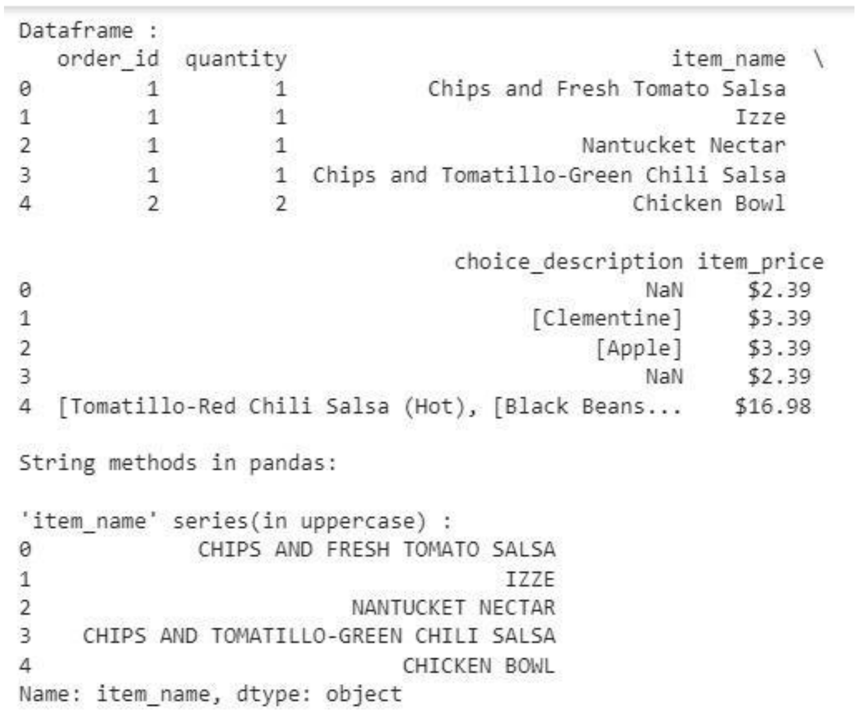


```
Dataframe of top-rated IMDb movies:
   star_rating                     title content_rating   genre  duration  \
0          9.3  The Shawshank Redemption              R   Crime       142
1          9.2             The Godfather              R   Crime       175
2          9.1    The Godfather: Part II              R   Crime       200
3          9.0           The Dark Knight          PG-13  Action       152
4          8.9              Pulp Fiction              R   Crime       154

                                         actors_list
0  [u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1    [u'Marlon Brando', u'Al Pacino', u'James Caan']
2  [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3  [u'Christian Bale', u'Heath Ledger', u'Aaron E...
4  [u'John Travolta', u'Uma Thurman', u'Samuel L....

Different ways to filter rows of a pandas DataFrame by column value:
Example : Filter rows to only show movies with a duration of atleast 200 minutes
1.using for loop
0    False
1    False
2     True
3    False
4    False
dtype: bool


2.broadcasting
    star_rating                                          title content_rating  \
2           9.1                         The Godfather: Part II              R
7           8.9  The Lord of the Rings: The Return of the King          PG-13
17          8.7                                  Seven Samurai        UNRATED
78          8.4                    Once Upon a Time in America              R
85          8.4                             Lawrence of Arabia             PG

        genre  duration                                        actors_list
2       Crime       200  [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7   Adventure       201  [u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17      Drama       207  [u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78      Crime       229  [u'Robert De Niro', u'James Woods', u'Elizabet...
85  Adventure       216  [u"Peter O'Toole", u'Alec Guinness', u'Anthony...
3.using 'loc' method
    star_rating                                          title content_rating  \
2           9.1                         The Godfather: Part II              R
7           8.9  The Lord of the Rings: The Return of the King          PG-13
17          8.7                                  Seven Samurai        UNRATED
78          8.4                    Once Upon a Time in America              R
85          8.4                             Lawrence of Arabia             PG

        genre  duration                                        actors_list
2       Crime       200  [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
7   Adventure       201  [u'Elijah Wood', u'Viggo Mortensen', u'Ian McK...
17      Drama       207  [u'Toshir\xf4 Mifune', u'Takashi Shimura', u'K...
78      Crime       229  [u'Robert De Niro', u'James Woods', u'Elizabet...
85  Adventure       216  [u"Peter O'Toole", u'Alec Guinness', u'Anthony...
```

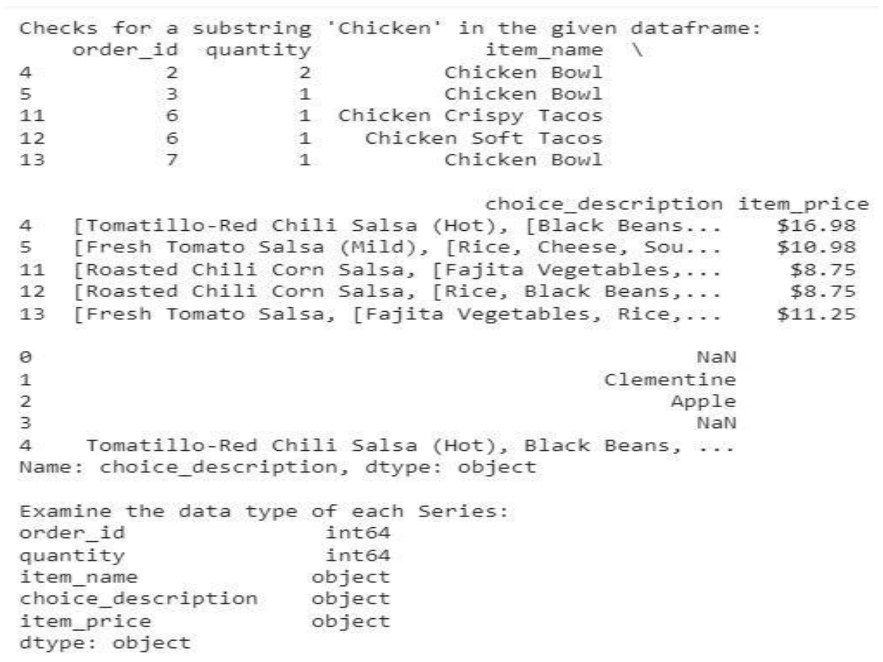
4.

```
import pandas as pd
# read a dataset of Chipotle orders into a DataFrame
orders = pd.read_table('http://bit.ly/chiporders')
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()
```

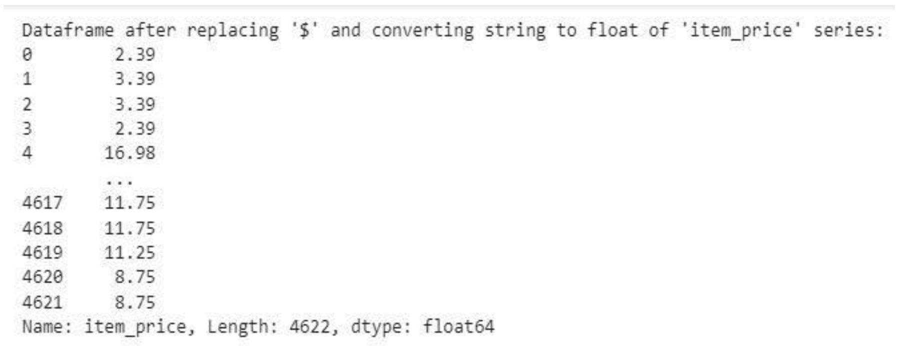
OUTPUT :



```
Dataframe :
   order_id  quantity                              item_name  \
0         1         1           Chips and Fresh Tomato Salsa
1         1         1                                   Izze
2         1         1                       Nantucket Nectar
3         1         1  Chips and Tomatillo-Green Chili Salsa
4         2         2                           Chicken Bowl

                                  choice_description item_price
0                                                NaN     $2.39
1                                       [Clementine]     $3.39
2                                            [Apple]     $3.39
3                                                NaN     $2.39
4  [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98

String methods in pandas:

'item_name' series(in uppercase) :
0             CHIPS AND FRESH TOMATO SALSA
1                                     IZZE
2                         NANTUCKET NECTAR
3    CHIPS AND TOMATILLO-GREEN CHILI SALSA
4                             CHICKEN BOWL
Name: item_name, dtype: object
```



```
Checks for a substring 'Chicken' in the given dataframe:
    order_id  quantity             item_name  \
4          2         2          Chicken Bowl
5          3         1          Chicken Bowl
11         6         1  Chicken Crispy Tacos
12         6         1    Chicken Soft Tacos
13         7         1          Chicken Bowl

                             choice_description item_price
4   [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98
5   [Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...    $10.98
11  [Roasted Chili Corn Salsa, [Fajita Vegetables,...     $8.75
12  [Roasted Chili Corn Salsa, [Rice, Black Beans,...     $8.75
13  [Fresh Tomato Salsa, [Fajita Vegetables, Rice,...    $11.25

0                                                  NaN
1                                           Clementine
2                                                Apple
3                                                  NaN
4    Tomatillo-Red Chili Salsa (Hot), Black Beans, ...
Name: choice_description, dtype: object

Examine the data type of each Series:
order_id               int64
quantity               int64
item_name             object
choice_description    object
item_price            object
dtype: object
```



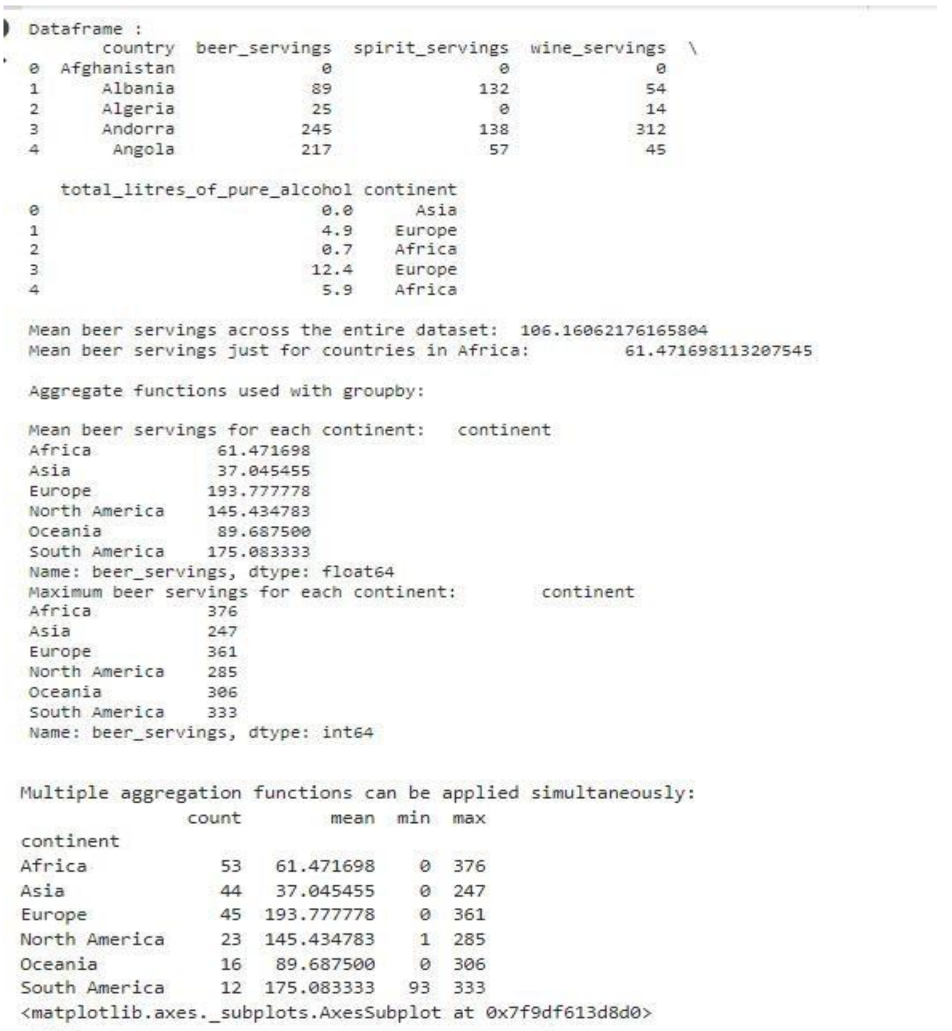
```
Dataframe after replacing '$' and converting string to float of 'item_price' series:
0        2.39
1        3.39
2        3.39
3        2.39
4       16.98
        ...
4617    11.75
4618    11.75
4619    11.25
4620     8.75
4621     8.75
Name: item_price, Length: 4622, dtype: float64
```

5.

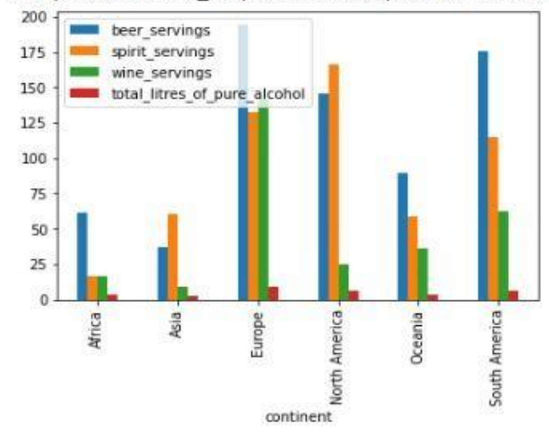
```
import pandas as pd
#read a dataset of alcohol consumption into a DataFrame
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
print("Dataframe : ")
print(drinks.head())
print()
print("Mean beer servings across the entire dataset: ",drinks.beer_servings.mean())
print("Mean beer servings just for 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 :



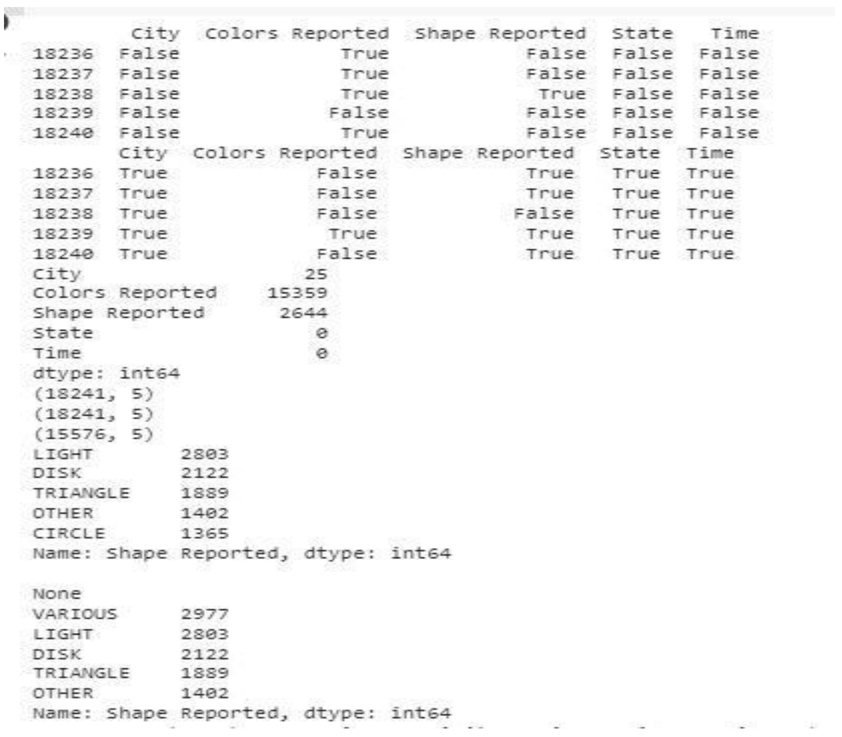
```
Dataframe :
       country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent
0                           0.0      Asia
1                           4.9    Europe
2                           0.7    Africa
3                          12.4    Europe
4                           5.9    Africa

Mean beer servings across the entire dataset:  106.16062176165804
Mean beer servings just for countries in Africa:         61.471698113207545

Aggregate functions used with groupby:

Mean beer servings for each continent:   continent
Africa            61.471698
Asia              37.045455
Europe           193.777778
North America    145.434783
Oceania           89.687500
South America    175.083333
Name: beer_servings, dtype: float64
Maximum beer servings for each continent:        continent
Africa           376
Asia             247
Europe           361
North America    285
Oceania          306
South America    333
Name: beer_servings, dtype: int64


Multiple aggregation functions can be applied simultaneously:
               count        mean  min  max
continent
Africa            53   61.471698    0  376
Asia              44   37.045455    0  247
Europe            45  193.777778    0  361
North America     23  145.434783    1  285
Oceania           16   89.687500    0  306
South America     12  175.083333   93  333
<matplotlib.axes._subplots.AxesSubplot at 0x7f9df613d8d0>
```



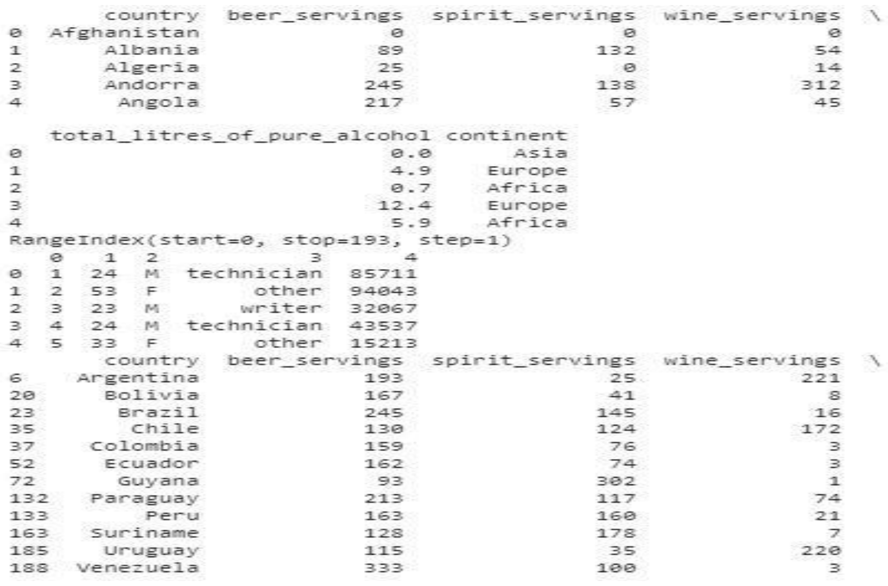
6.
```
import pandas as pd
ufo = pd.read_csv('http://bit.ly/uforeports')
print(ufo.isnull().tail())
print(ufo.notnull().tail())
print(ufo.isnull().sum())
print(ufo.shape)
# if 'all' values are missing in a row, then drop that row (none are dropped in this case)
print(ufo.dropna(how='all').shape)
print(ufo.dropna(subset=['City', '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')
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', 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())
```

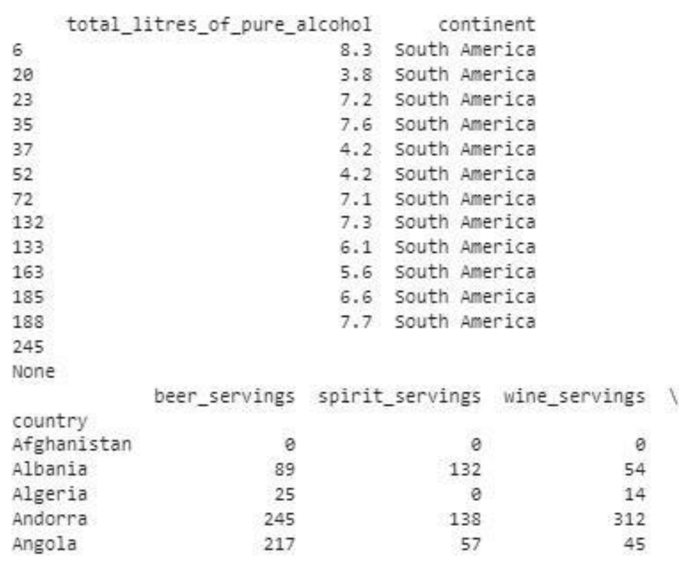
OUTPUT :



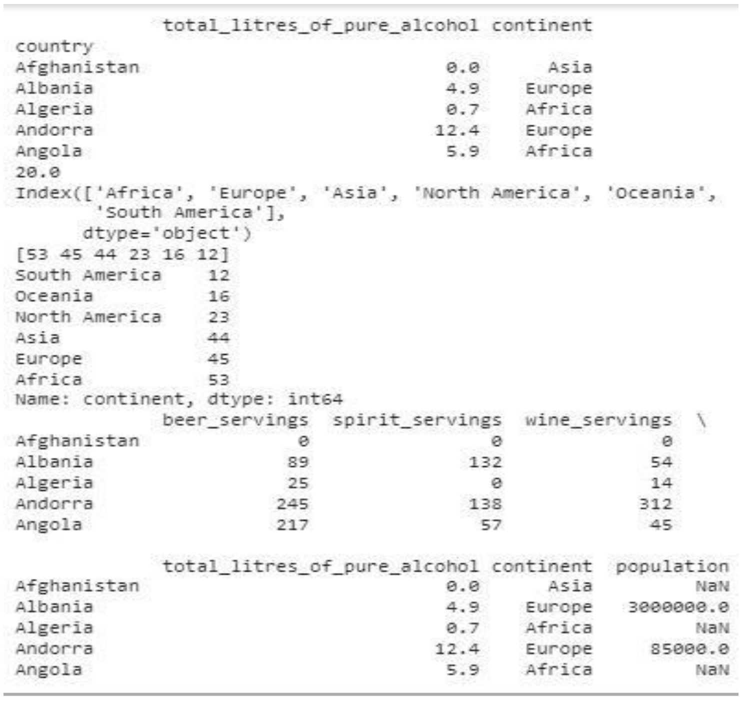
```
        City  Colors Reported  Shape Reported  State   Time
18236  False             True           False  False  False
18237  False             True           False  False  False
18238  False             True            True  False  False
18239  False            False           False  False  False
18240  False             True           False  False  False
       City  Colors Reported  Shape Reported  State  Time
18236  True            False            True   True  True
18237  True            False            True   True  True
18238  True            False           False   True  True
18239  True             True            True   True  True
18240  True            False            True   True  True
City                  25
Colors Reported    15359
Shape Reported      2644
State                  0
Time                   0
dtype: int64
(18241, 5)
(18241, 5)
(15576, 5)
LIGHT       2803
DISK        2122
TRIANGLE    1889
OTHER       1402
CIRCLE      1365
Name: Shape Reported, dtype: int64

None
VARIOUS     2977
LIGHT       2803
DISK        2122
TRIANGLE    1889
OTHER       1402
Name: Shape Reported, dtype: int64
```



```
       country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent
0                           0.0      Asia
1                           4.9    Europe
2                           0.7    Africa
3                          12.4    Europe
4                           5.9    Africa
RangeIndex(start=0, stop=193, step=1)
   0   1  2           3      4
0  1  24  M  technician  85711
1  2  53  F       other  94043
2  3  23  M      writer  32067
3  4  24  M  technician  43537
4  5  33  F       other  15213
       country  beer_servings  spirit_servings  wine_servings  \
6    Argentina            193               25            221
20     Bolivia            167               41              8
23      Brazil            245              145             16
35       Chile            130              124            172
37    Colombia            159               76              3
52     Ecuador            162               74              3
72      Guyana             93              302              1
132   Paraguay            213              117             74
133       Peru            163              160             21
163   Suriname            128              178              7
185    Uruguay            115               35            220
188  Venezuela            333              100              3
```



```
     total_litres_of_pure_alcohol      continent
6                             8.3  South America
20                            3.8  South America
23                            7.2  South America
35                            7.6  South America
37                            4.2  South America
52                            4.2  South America
72                            7.1  South America
132                           7.3  South America
133                           6.1  South America
163                           5.6  South America
185                           6.6  South America
188                           7.7  South America
245
None
             beer_servings  spirit_servings  wine_servings  \
country
Afghanistan              0                0              0
Albania                 89              132             54
Algeria                 25                0             14
Andorra                245              138            312
Angola                 217               57             45
```

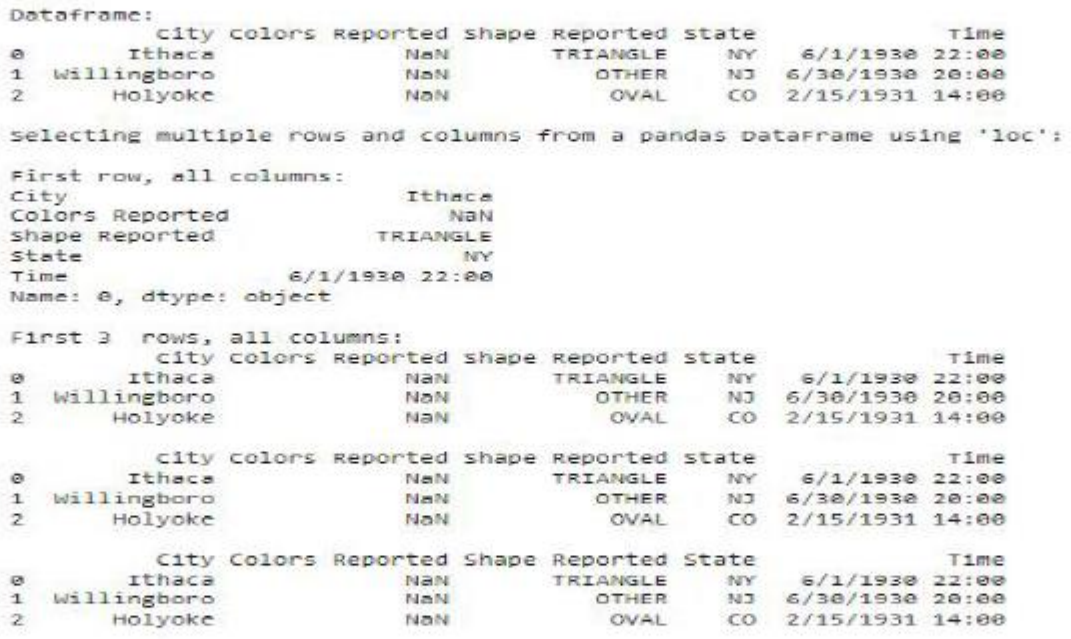


```
             total_litres_of_pure_alcohol continent
country
Afghanistan                           0.0      Asia
Albania                               4.9    Europe
Algeria                               0.7    Africa
Andorra                              12.4    Europe
Angola                                5.9    Africa
20.0
Index(['Africa', 'Europe', 'Asia', 'North America', 'Oceania',
       'South America'],
      dtype='object')
[53 45 44 23 16 12]
South America    12
Oceania          16
North America    23
Asia             44
Europe           45
Africa           53
Name: continent, dtype: int64
             beer_servings  spirit_servings  wine_servings  \
Afghanistan              0                0              0
Albania                 89              132             54
Algeria                 25                0             14
Andorra                245              138            312
Angola                 217               57             45

             total_litres_of_pure_alcohol continent  population
Afghanistan                           0.0      Asia         NaN
Albania                               4.9    Europe   3000000.0
Algeria                               0.7    Africa         NaN
Andorra                              12.4    Europe     85000.0
Angola                                5.9    Africa         NaN
```

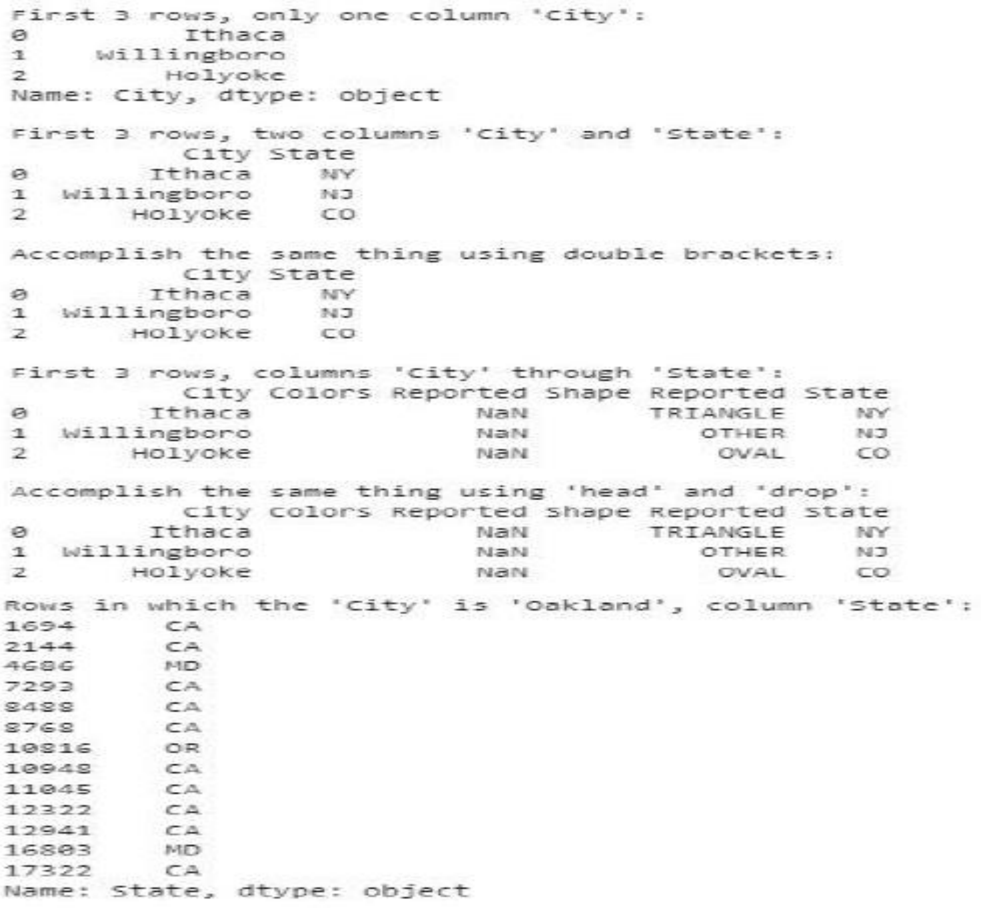
7.

```
import pandas as pd
ufo = pd.read_csv('http://bit.ly/uforeports')
print("Dataframe: ")
print(ufo.head(3))
print()
print("Selecting multiple rows and columns from a pandas DataFrame using 'loc': ")
print()
#loc method is used to select rows and columns by label
print("First row, all columns: ")
print(ufo.loc[0, :])
print()
```

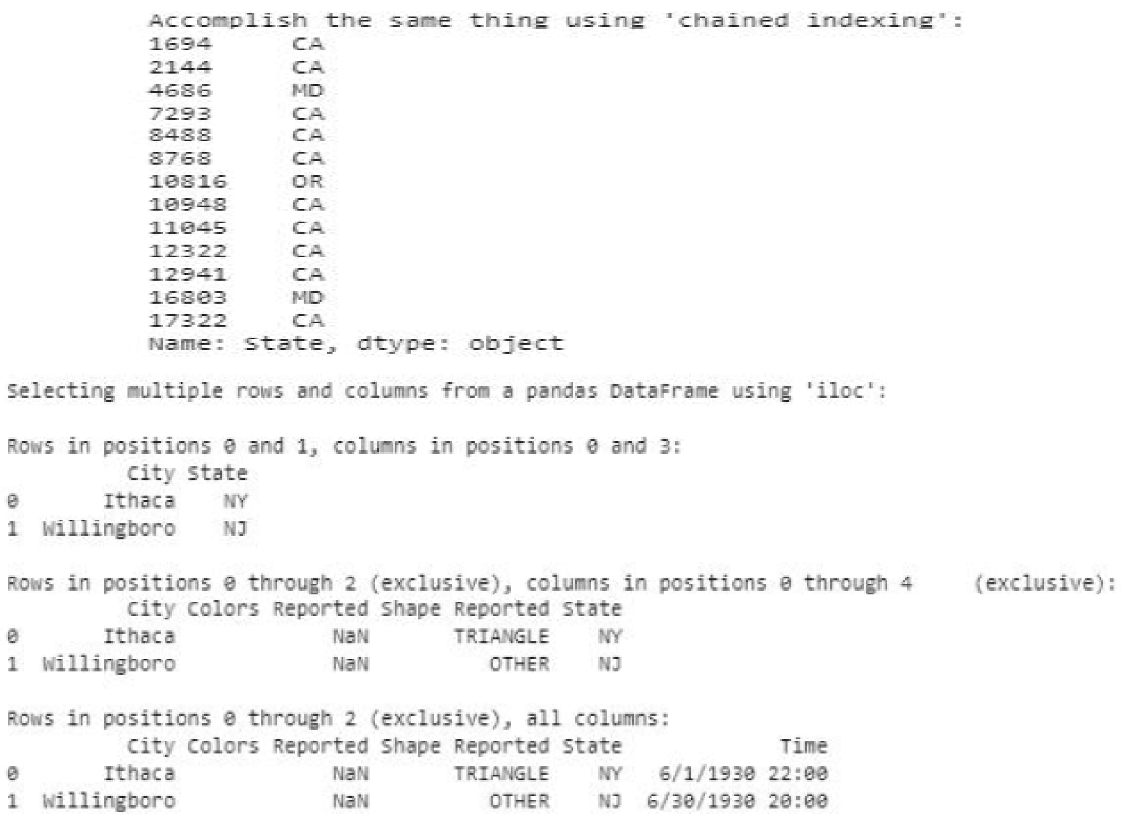
```
  print("First 3 rows, all columns: ")
  print(ufo.loc[[0, 1, 2], :])
  print()
  # rows 0 through 2 (inclusive), all columns
print(ufo.loc[0:2, :])
print()
# this implies "all columns", but explicitly stating "all columns" is better
print(ufo.loc[0:2])
print()
print("First 3 rows, only one column 'City': ")
print(ufo.loc[0:2, 'City'])
print()
print("First 3 rows, two columns 'City' and 'State': ")
print(ufo.loc[0:2, ['City', 'State']])
print()
print("Accomplish the same thing using double brackets: ")
#using 'loc' is preferred since it's more explicit
print(ufo[['City', 'State']].head(3))
print()
print("First 3 rows, columns 'City' through 'State': ")
print(ufo.loc[0:2, 'City':'State'])
print()
print("Accomplish the same thing using 'head' and 'drop': ")
print(ufo.head(3).drop('Time', axis=1))
print()
print("Rows in which the 'City' is 'Oakland', column 'State': ")
print(ufo.loc[ufo.City=='Oakland', 'State'])
print()
print("Accomplish the same thing using 'chained indexing': ")
#using 'loc' is preferred since chained indexing can cause problems
print(ufo[ufo.City=='Oakland'].State)
print()
print("Selecting multiple rows and columns from a pandas DataFrame using 'iloc': ")
print()
print("Rows in positions 0 and 1, columns in positions 0 and 3: ")
print(ufo.iloc[[0, 1], [0, 3]])
print()
print("Rows in positions 0 through 2 (exclusive), columns in positions 0 through 4
(exclusive): ")
print(ufo.iloc[0:2, 0:4])
print()
print("Rows in positions 0 through 2 (exclusive), all columns: ")
print(ufo.iloc[0:2, :])
print()
```



```
Dataframe:
          City Colors Reported Shape Reported State             Time
0       Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1  Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2      Holyoke             NaN           OVAL    CO  2/15/1931 14:00

Selecting multiple rows and columns from a pandas DataFrame using 'loc':

First row, all columns:
City                        Ithaca
Colors Reported                NaN
Shape Reported            TRIANGLE
State                           NY
Time                6/1/1930 22:00
Name: 0, dtype: object

First 3  rows, all columns:
          City Colors Reported Shape Reported State             Time
0       Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1  Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2      Holyoke             NaN           OVAL    CO  2/15/1931 14:00

          City Colors Reported Shape Reported State             Time
0       Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1  Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2      Holyoke             NaN           OVAL    CO  2/15/1931 14:00

          City Colors Reported Shape Reported State             Time
0       Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1  Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2      Holyoke             NaN           OVAL    CO  2/15/1931 14:00
```



```
First 3 rows, only one column 'City':
0         Ithaca
1    Willingboro
2        Holyoke
Name: City, dtype: object

First 3 rows, two columns 'City' and 'State':
          City State
0       Ithaca    NY
1  Willingboro    NJ
2      Holyoke    CO

Accomplish the same thing using double brackets:
          City State
0       Ithaca    NY
1  Willingboro    NJ
2      Holyoke    CO

First 3 rows, columns 'City' through 'State':
          City Colors Reported Shape Reported State
0       Ithaca             NaN       TRIANGLE    NY
1  Willingboro             NaN          OTHER    NJ
2      Holyoke             NaN           OVAL    CO

Accomplish the same thing using 'head' and 'drop':
          City Colors Reported Shape Reported State
0       Ithaca             NaN       TRIANGLE    NY
1  Willingboro             NaN          OTHER    NJ
2      Holyoke             NaN           OVAL    CO
Rows in which the 'City' is 'Oakland', column 'State':
1694     CA
2144     CA
4686     MD
7293     CA
8488     CA
8768     CA
10816    OR
10948    CA
11045    CA
12322    CA
12941    CA
16803    MD
17322    CA
Name: State, dtype: object
```

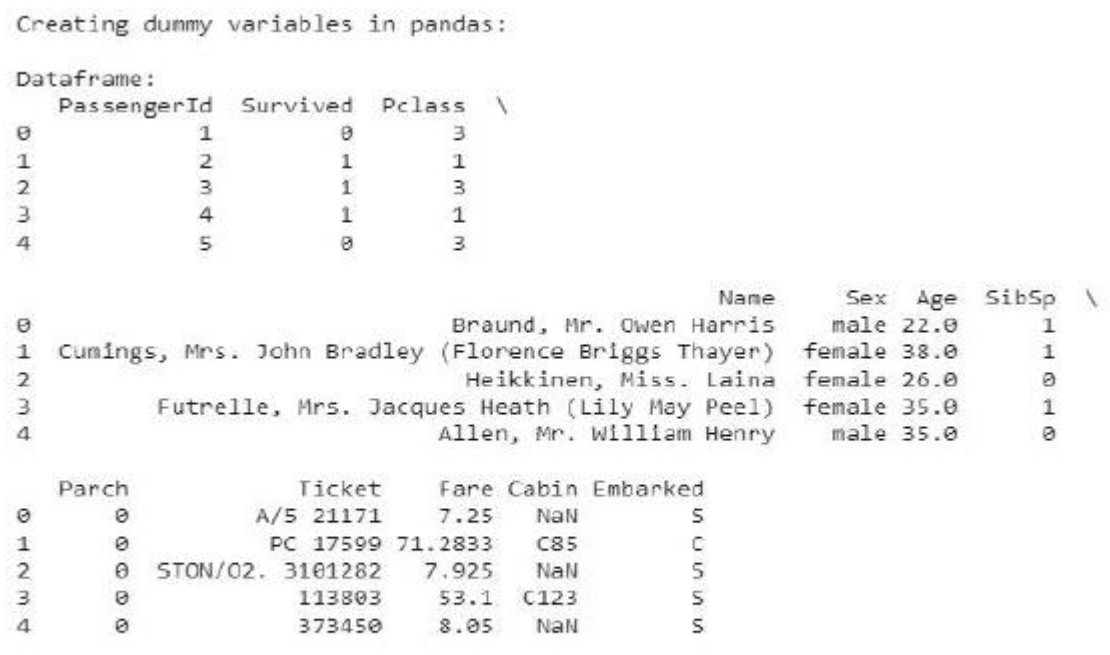


```
Accomplish the same thing using 'chained indexing':
1694     CA
2144     CA
4686     MD
7293     CA
8488     CA
8768     CA
10816    OR
10948    CA
11045    CA
12322    CA
12941    CA
16803    MD
17322    CA
Name: State, dtype: object

Selecting multiple rows and columns from a pandas DataFrame using 'iloc':

Rows in positions 0 and 1, columns in positions 0 and 3:
         City State
0      Ithaca    NY
1  Willingboro   NJ

Rows in positions 0 through 2 (exclusive), columns in positions 0 through 4     (exclusive):
         City Colors Reported Shape Reported State
0      Ithaca             NaN       TRIANGLE    NY
1  Willingboro            NaN          OTHER    NJ

Rows in positions 0 through 2 (exclusive), all columns:
         City Colors Reported Shape Reported State             Time
0      Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1  Willingboro            NaN          OTHER    NJ  6/30/1930 20:00
```

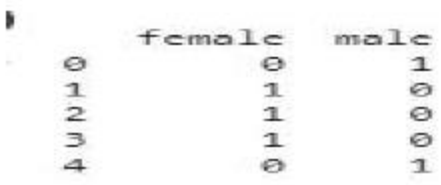
8. import pandas as pd

```
print("Creating dummy variables in pandas: ")
print()
# read the training dataset from Kaggle's Titanic competition
train = pd.read_csv('http://bit.ly/kaggletrain')
print("Dataframe: ")
print(train.head())
print()
#use 'get_dummies' to create one column for every possible value
print(pd.get_dummies(train.Sex).head())
print()
# drop the first dummy variable ('female') using the 'iloc' method
print(pd.get_dummies(train.Sex).iloc[:, 1:].head())
print()
# add a prefix to identify the source of the dummy variables
print(pd.get_dummies(train.Sex, prefix='Sex').iloc[:, 1:].head())
print()
# use 'get_dummies' with a feature that has 3 possible values
print(pd.get_dummies(train.Embarked, prefix='Embarked').head(10))
print()
# drop the first dummy variable ('C')
print(pd.get_dummies(train.Embarked, prefix='Embarked').iloc[:, 1:].head(10))
print()
#0, 0 means C 1, 0 means Q 0, 1 means S
# reset the DataFrame
train = pd.read_csv('http://bit.ly/kaggletrain')
print("Dataframe: ")
print(train.head())
print()
```

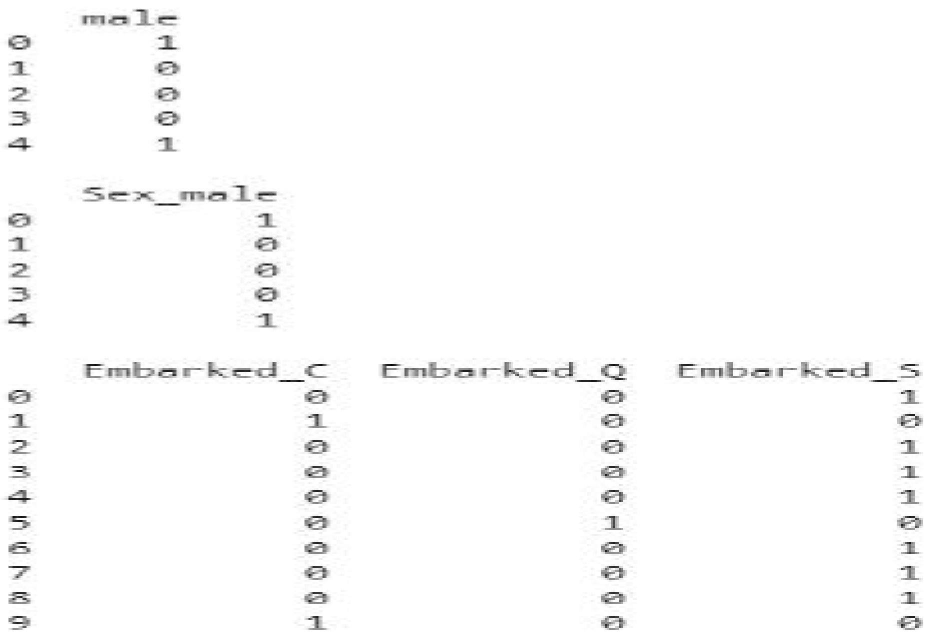
```
# pass the DataFrame to 'get_dummies' and specify which columns to dummy (it drops
#the original columns)
print(pd.get_dummies(train, columns=['Sex', 'Embarked']).head())
print()
# use the 'drop_first' parameter (new in pandas 0.18) to drop the first dummy variable
#for each feature
print(pd.get_dummies(train, columns=['Sex', 'Embarked'], drop_first=True).head())
```



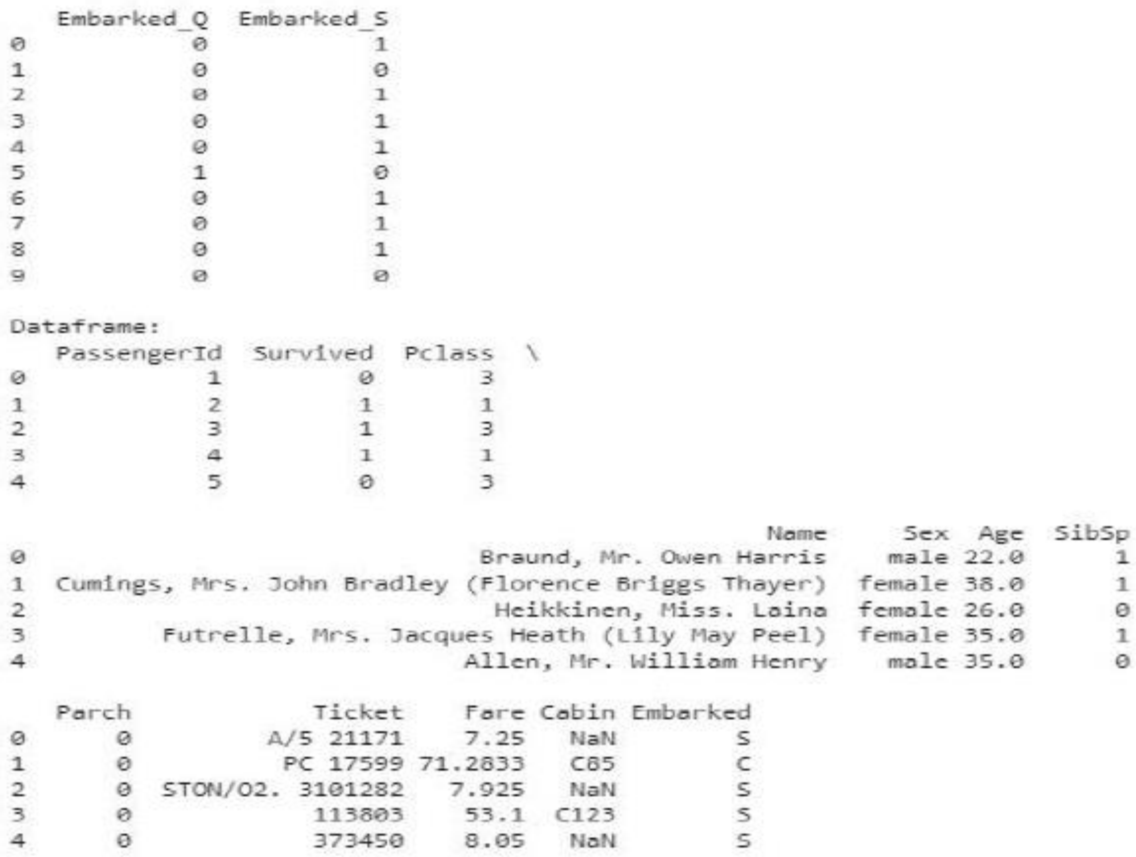
```
Creating dummy variables in pandas:

Dataframe:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Thayer)  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.25   NaN        S
1      0          PC 17599 71.2833   C85        C
2      0  STON/O2. 3101282   7.925   NaN        S
3      0            113803    53.1  C123        S
4      0            373450    8.05   NaN        S
```



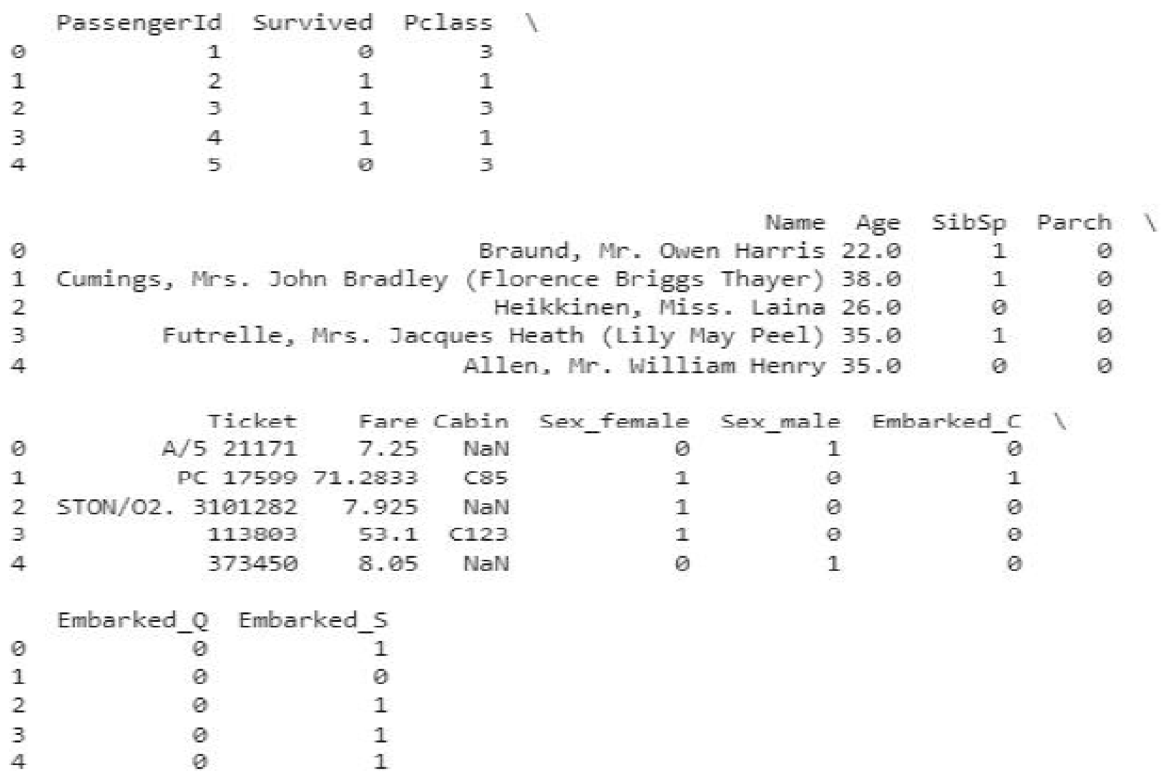
```
   female  male
0       0     1
1       1     0
2       1     0
3       1     0
4       0     1
```



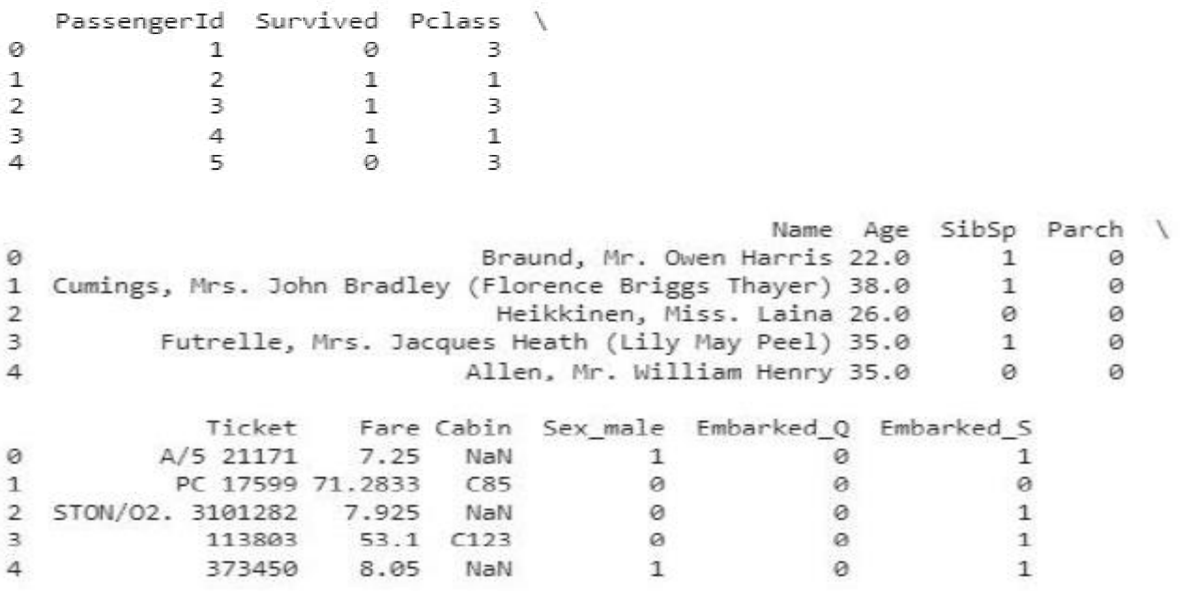
```
   male
0     1
1     0
2     0
3     0
4     1

   Sex_male
0         1
1         0
2         0
3         0
4         1

   Embarked_C  Embarked_Q  Embarked_S
0           0           0           1
1           1           0           0
2           0           0           1
3           0           0           1
4           0           0           1
5           0           1           0
6           0           0           1
7           0           0           1
8           0           0           1
9           1           0           0
```



```
   Embarked_Q  Embarked_S
0           0           1
1           0           0
2           0           1
3           0           1
4           0           1
5           1           0
6           0           1
7           0           1
8           0           1
9           0           0

Dataframe:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Thayer)  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171     7.25   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282    7.925   NaN        S
3      0            113803     53.1  C123        S
4      0            373450     8.05   NaN        S
```



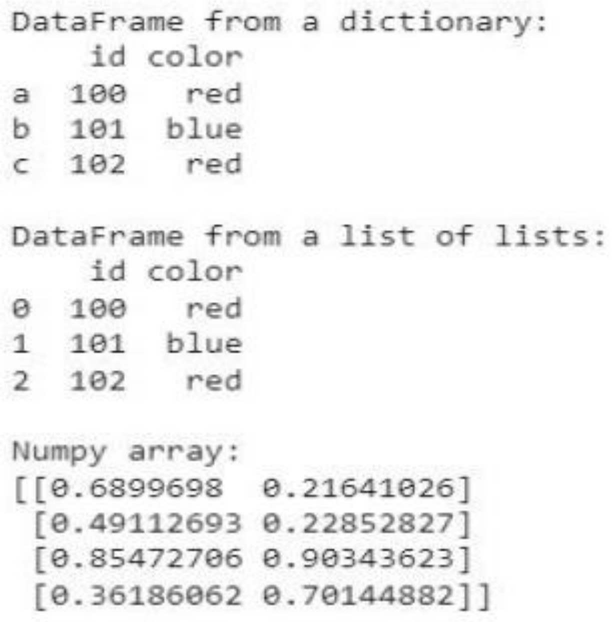
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name   Age  SibSp  Parch  \
0                            Braund, Mr. Owen Harris  22.0      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Thayer)  38.0      1      0
2                             Heikkinen, Miss. Laina  26.0      0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  35.0      1      0
4                           Allen, Mr. William Henry  35.0      0      0

             Ticket     Fare Cabin  Sex_female  Sex_male  Embarked_C  \
0         A/5 21171     7.25   NaN           0         1           0
1          PC 17599  71.2833   C85           1         0           1
2  STON/O2. 3101282    7.925   NaN           1         0           0
3            113803     53.1  C123           1         0           0
4            373450     8.05   NaN           0         1           0

   Embarked_Q  Embarked_S
0           0           1
1           0           0
2           0           1
3           0           1
4           0           1
```



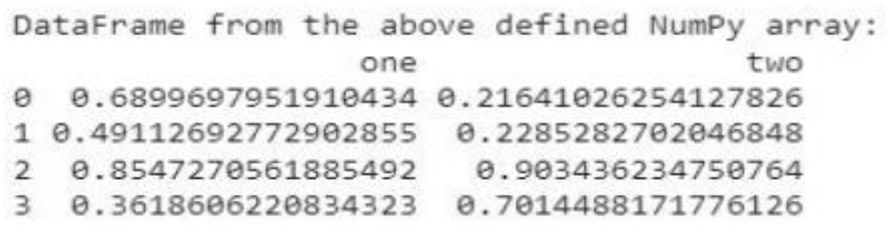
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name   Age  SibSp  Parch  \
0                            Braund, Mr. Owen Harris  22.0      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Thayer)  38.0      1      0
2                             Heikkinen, Miss. Laina  26.0      0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  35.0      1      0
4                           Allen, Mr. William Henry  35.0      0      0

             Ticket     Fare Cabin  Sex_male  Embarked_Q  Embarked_S
0         A/5 21171     7.25   NaN         1           0           1
1          PC 17599  71.2833   C85         0           0           0
2  STON/O2. 3101282    7.925   NaN         0           0           1
3            113803     53.1  C123         0           0           1
4            373450     8.05   NaN         1           0           1
```

9. import pandas as pd
import numpy as np
# create a DataFrame from a dictionary (keys become column names, values become
#data) optionally specify the order of columns and define the index
df = pd.DataFrame({'id':[100, 101, 102], 'color':['red', 'blue', 'red']}, columns=['id', 'color'],
index=['a', 'b', 'c'])
print("DataFrame from a dictionary: ")
print(df)
print()
# create a DataFrame from a list of lists (each inner list becomes a row)

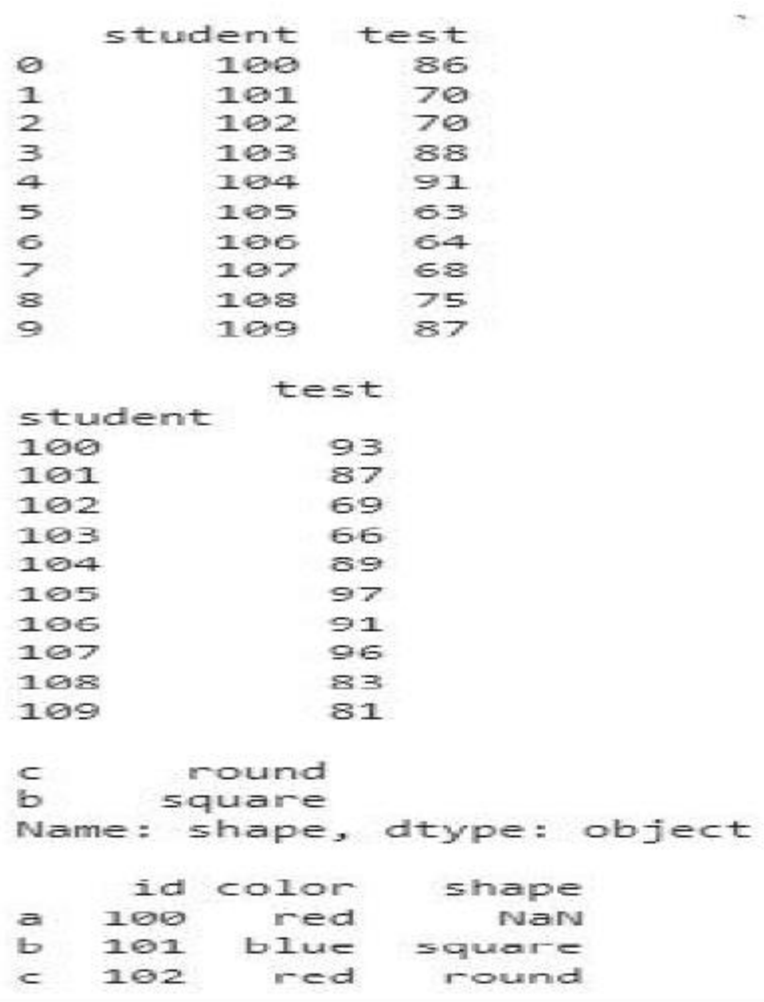
```
print("DataFrame from a list of lists: ")
print(pd.DataFrame([[100, 'red'], [101, 'blue'], [102, 'red']], columns=['id', 'color']))
print()
# create a NumPy array (with shape 4 by 2) and fill it with random numbers between0&1
arr = np.random.rand(4, 2)
print("Numpy array: ")
print(arr)
print()
print("DataFrame from the above defined NumPy array: ")
print(pd.DataFrame(arr, columns=['one', 'two']))
print()
print("DataFrame of student IDs (100 through 109) and test scores (random integers
between 60 and 100: ")
print(pd.DataFrame({'student':np.arange(100, 110, 1), 'test':np.random.randint(60, 101,
10)}))
print()
# 'set_index' can be chained with the DataFrame constructor to select an index
print(pd.DataFrame({'student':np.arange(100, 110, 1), 'test':np.random.randint(60,
101,10)}).set_index('student'))
print()
# create a new Series using the Series constructor
s = pd.Series(['round', 'square'], index=['c', 'b'], name='shape')
print(s)
print()
# concatenate the DataFrame and the Series (use axis=1 to concatenate columns)
print(pd.concat([df, s], axis=1))
```



```
DataFrame from a dictionary:
    id color
a  100   red
b  101  blue
c  102   red

DataFrame from a list of lists:
    id color
0  100   red
1  101  blue
2  102   red

Numpy array:
[[0.6899698  0.21641026]
 [0.49112693 0.22852827]
 [0.85472706 0.90343623]
 [0.36186062 0.70144882]]
```



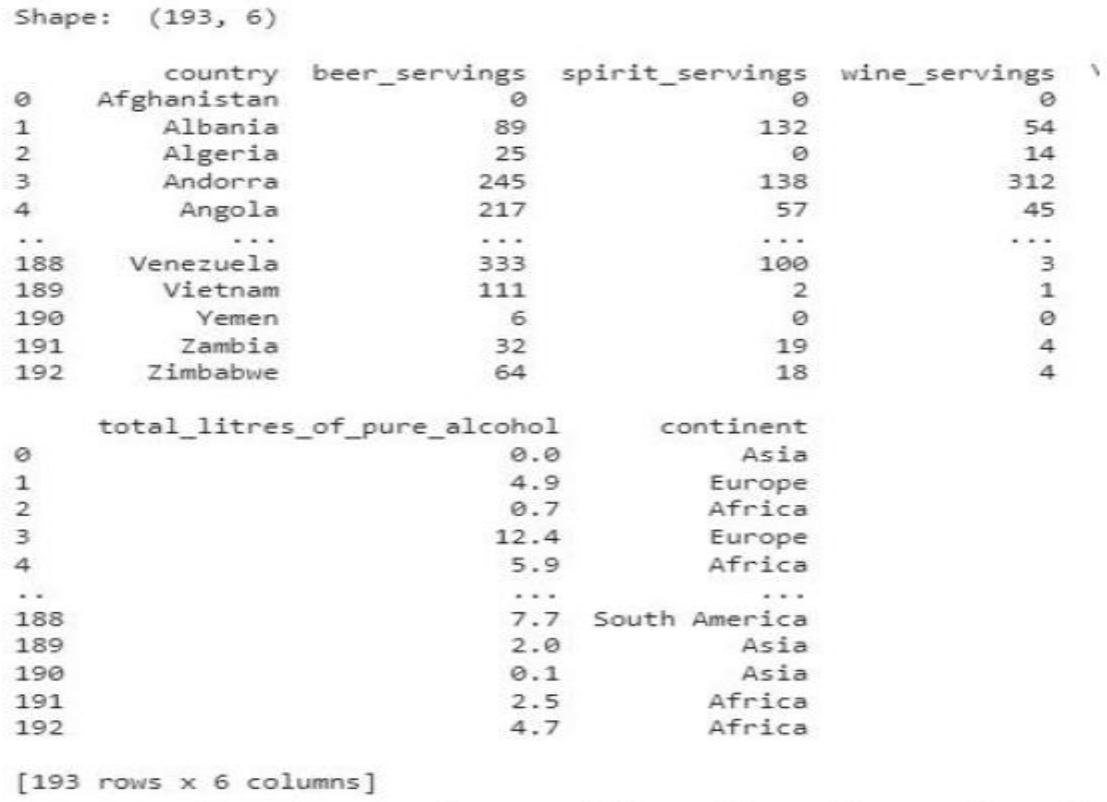
```
DataFrame from the above defined NumPy array:
                  one                 two
0  0.6899697951910434 0.21641026254127826
1 0.49112692772902855  0.2285282702046848
2  0.8547270561885492   0.903436234750764
3  0.3618606220834323  0.7014488171776126
```



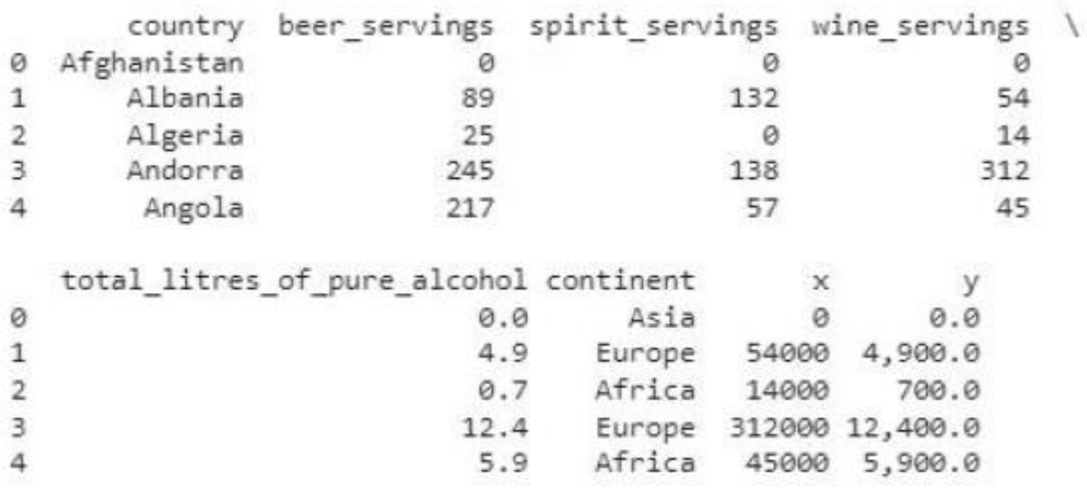
```
   student  test
0      100    86
1      101    70
2      102    70
3      103    88
4      104    91
5      105    63
6      106    64
7      107    68
8      108    75
9      109    87

         test
student
100        93
101        87
102        69
103        66
104        89
105        97
106        91
107        96
108        83
109        81

c     round
b    square
Name: shape, dtype: object

    id color   shape
a  100   red     NaN
b  101  blue  square
c  102   red   round
```

```
10. import pandas as pd
# change display options in pandas
# read a dataset of alcohol consumption into a DataFrame
drinks = pd.read_csv('http://bit.ly/drinksbycountry')
print("Shape: ",drinks.shape)
print()
# check the current setting for the 'max_rows' option
pd.get_option('display.max_rows')
print(drinks)
print()
# overwrite the current setting so that all rows will be displayed
pd.set_option('display.max_rows',2)
print(drinks)
print()
# reset the 'max_rows' option to its default
pd.reset_option('display.max_rows')
print(drinks)
print()
# add two meaningless columns to the drinks DataFrame
drinks['x'] = drinks.wine_servings * 1000
drinks['y'] = drinks.total_litres_of_pure_alcohol * 1000
print(drinks.head())
print()
```

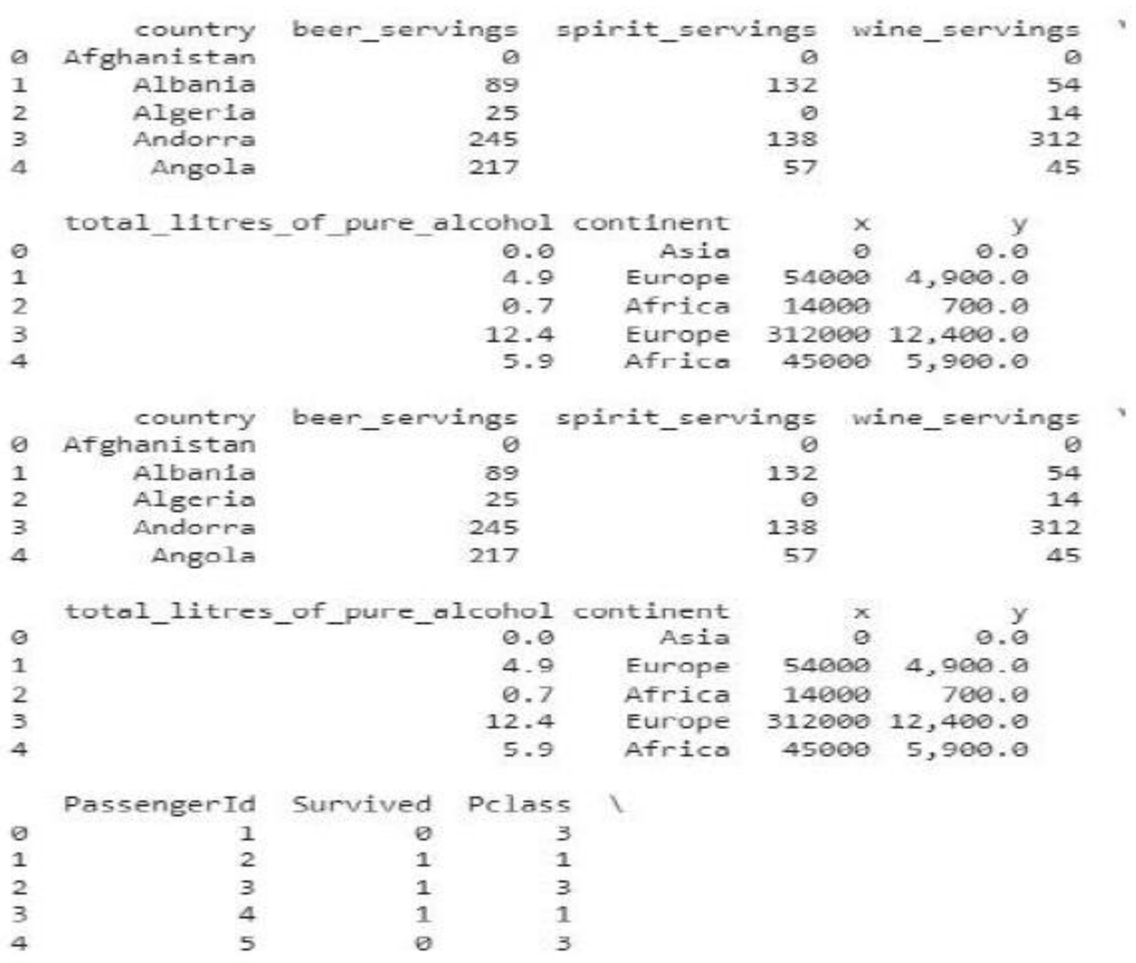
```
# use a Python format string to specify a comma as the thousands separator
pd.set_option('display.float_format', '{:,}'.format)
print(drinks.head())
print()
# read the training dataset from Kaggle's Titanic competition into a DataFrame
train = pd.read_csv('http://bit.ly/kaggletrain')
# an ellipsis is displayed in the 'Name' cell of row 1 because of the 'max_colwidth' option
pd.get_option('display.max_colwidth')
print(train.head())
print()
# overwrite the current setting so that more characters will be displayed
pd.set_option('display.max_colwidth', 1000)
print(train.head())
print()
```



```
Shape:  (193, 6)

         country  beer_servings  spirit_servings  wine_servings  \
0    Afghanistan              0                0              0
1        Albania             89              132             54
2        Algeria             25                0             14
3        Andorra            245              138            312
4         Angola            217               57             45
..           ...            ...              ...            ...
188    Venezuela            333              100              3
189      Vietnam            111                2              1
190        Yemen              6                0              0
191       Zambia             32               19              4
192     Zimbabwe             64               18              4

     total_litres_of_pure_alcohol      continent
0                             0.0           Asia
1                             4.9         Europe
2                             0.7         Africa
3                            12.4         Europe
4                             5.9         Africa
..                            ...            ...
188                           7.7  South America
189                           2.0           Asia
190                           0.1           Asia
191                           2.5         Africa
192                           4.7         Africa

[193 rows x 6 columns]
```



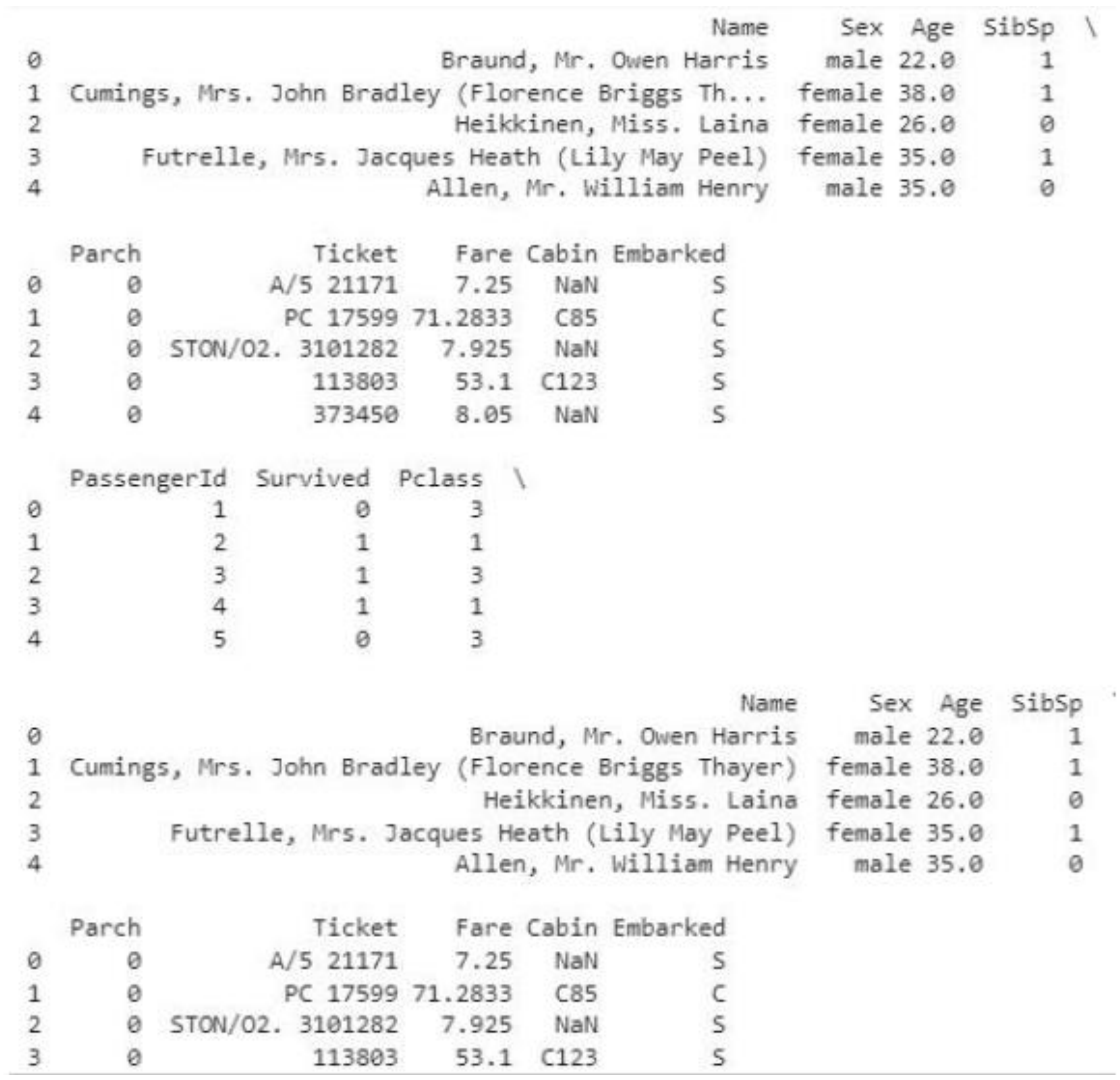
```
       country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent       x        y
0                           0.0      Asia       0      0.0
1                           4.9    Europe   54000  4,900.0
2                           0.7    Africa   14000    700.0
3                          12.4    Europe  312000 12,400.0
4                           5.9    Africa   45000  5,900.0
```



```
       country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent       x          y
0                           0.0      Asia       0        0.0
1                           4.9    Europe   54000    4,900.0
2                           0.7    Africa   14000      700.0
3                          12.4    Europe  312000   12,400.0
4                           5.9    Africa   45000    5,900.0
```

```
       country  beer_servings  spirit_servings  wine_servings  \
0  Afghanistan              0                0              0
1      Albania             89              132             54
2      Algeria             25                0             14
3      Andorra            245              138            312
4       Angola            217               57             45

   total_litres_of_pure_alcohol continent       x          y
0                           0.0      Asia       0        0.0
1                           4.9    Europe   54000    4,900.0
2                           0.7    Africa   14000      700.0
3                          12.4    Europe  312000   12,400.0
4                           5.9    Africa   45000    5,900.0
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
```

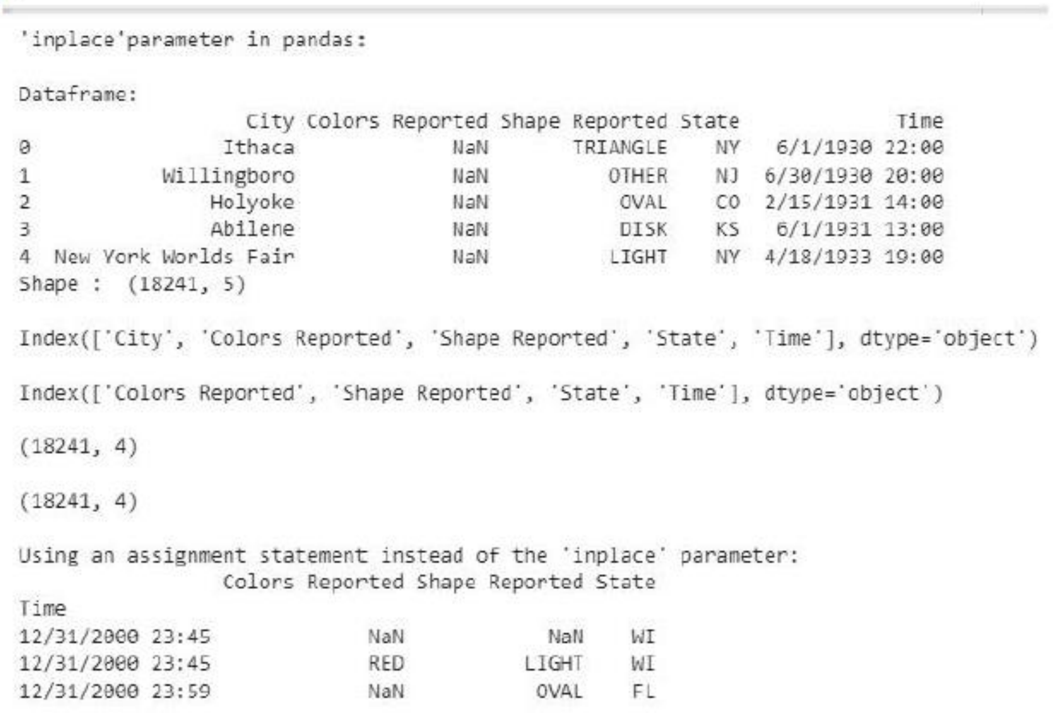


```
                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171     7.25   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282    7.925   NaN        S
3      0            113803     53.1  C123        S
4      0            373450     8.05   NaN        S

   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Thayer)  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171     7.25   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282    7.925   NaN        S
3      0            113803     53.1  C123        S
```

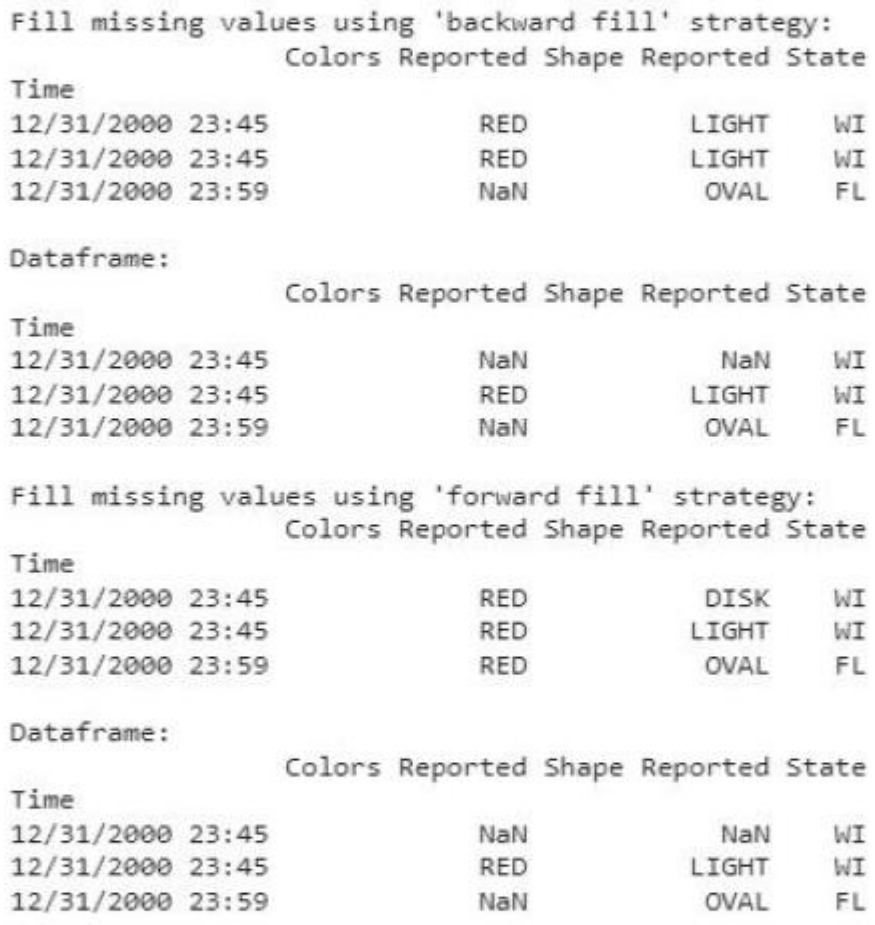
11. import pandas as pd
# read a dataset of UFO reports into a DataFrame
print("'inplace'parameter in pandas: ")
print()
ufo = pd.read_csv('http://bit.ly/uforeports')
print("Dataframe: ")
print(ufo.head())
print("Shape : ",ufo.shape)
print()
# remove the 'City' column (doesn't affect the DataFrame since inplace=False)
ufo.drop('City', axis=1)
# confirm that the 'City' column was not actually removed
print(ufo.columns)
print()
# remove the 'City' column (does affect the DataFrame since inplace=True)
ufo.drop('City', axis=1, inplace=True)
# confirm that the 'City' column was actually removed
print(ufo.columns)
print()
print(ufo.shape)
print()

```
#drop a row if any value is missing from that row (doesn't affect the DataFrame since
#inplace=False)
ufo.dropna(how='any')
# confirm that no rows were actually removed
print(ufo.shape)
print()
print("Using an assignment statement instead of the 'inplace' parameter: ")
ufo = ufo.set_index('Time')
print(ufo.tail(3))
print()
print("Fill missing values using 'backward fill' strategy: ")
# doesn't affect the DataFrame since inplace=False
print(ufo.fillna(method='bfill').tail(3))
print()
print("Dataframe: ")
print(ufo.tail(3))
print()
print("Fill missing values using 'forward fill' strategy: ")
#doesn't affect the DataFrame since inplace=False
print(ufo.fillna(method='ffill').tail(3))
print()
print("Dataframe: ")
print(ufo.tail(3))
```

OUTPUT :



```
'inplace'parameter in pandas:

Dataframe:
                   City Colors Reported Shape Reported State             Time
0                Ithaca             NaN       TRIANGLE    NY   6/1/1930 22:00
1           Willingboro             NaN          OTHER    NJ  6/30/1930 20:00
2               Holyoke             NaN           OVAL    CO  2/15/1931 14:00
3               Abilene             NaN           DISK    KS   6/1/1931 13:00
4  New York Worlds Fair             NaN          LIGHT    NY  4/18/1933 19:00
Shape :  (18241, 5)

Index(['City', 'Colors Reported', 'Shape Reported', 'State', 'Time'], dtype='object')

Index(['Colors Reported', 'Shape Reported', 'State', 'Time'], dtype='object')

(18241, 4)

(18241, 4)

Using an assignment statement instead of the 'inplace' parameter:
                 Colors Reported Shape Reported State
Time
12/31/2000 23:45             NaN            NaN    WI
12/31/2000 23:45             RED          LIGHT    WI
12/31/2000 23:59             NaN           OVAL    FL
```



```
Fill missing values using 'backward fill' strategy:
                Colors Reported Shape Reported State
Time
12/31/2000 23:45             RED          LIGHT    WI
12/31/2000 23:45             RED          LIGHT    WI
12/31/2000 23:59             NaN           OVAL    FL

Dataframe:
                Colors Reported Shape Reported State
Time
12/31/2000 23:45             NaN            NaN    WI
12/31/2000 23:45             RED          LIGHT    WI
12/31/2000 23:59             NaN           OVAL    FL

Fill missing values using 'forward fill' strategy:
                Colors Reported Shape Reported State
Time
12/31/2000 23:45             RED           DISK    WI
12/31/2000 23:45             RED          LIGHT    WI
12/31/2000 23:59             RED           OVAL    FL

Dataframe:
                Colors Reported Shape Reported State
Time
12/31/2000 23:45             NaN            NaN    WI
12/31/2000 23:45             RED          LIGHT    WI
12/31/2000 23:59             NaN           OVAL    FL
```