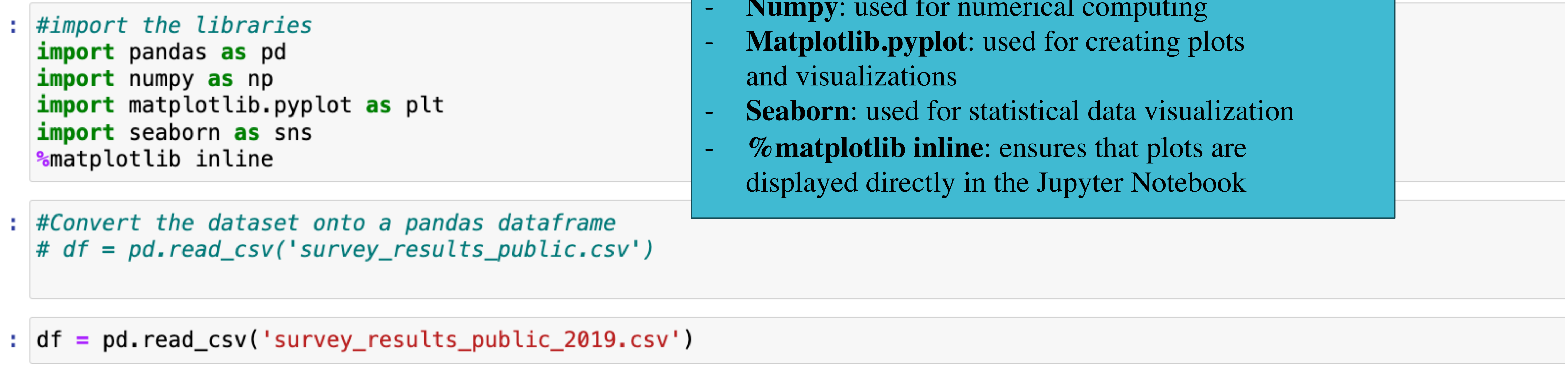
Project 1
Math 014-01 Introduction to Data Science

Seaborn group: Raiyan, Guy, Lexi, and Lily

March 25, 2024

- **Pandas**: used for data manipulation and analysis in Python
- **Numpy**: used for numerical computing
- **Matplotlib.pyplot**: used for creating plots and visualizations
- **Seaborn**: used for statistical data visualization
- **%matplotlib inline**: ensures that plots are displayed directly in the Jupyter Notebook



```
: #import the libraries
  import pandas as pd
  import numpy as np
  import matplotlib.pyplot as plt
  import seaborn as sns
  %matplotlib inline
```

```
: #Convert the dataset onto a pandas dataframe
  # df = pd.read_csv('survey_results_public.csv')
```

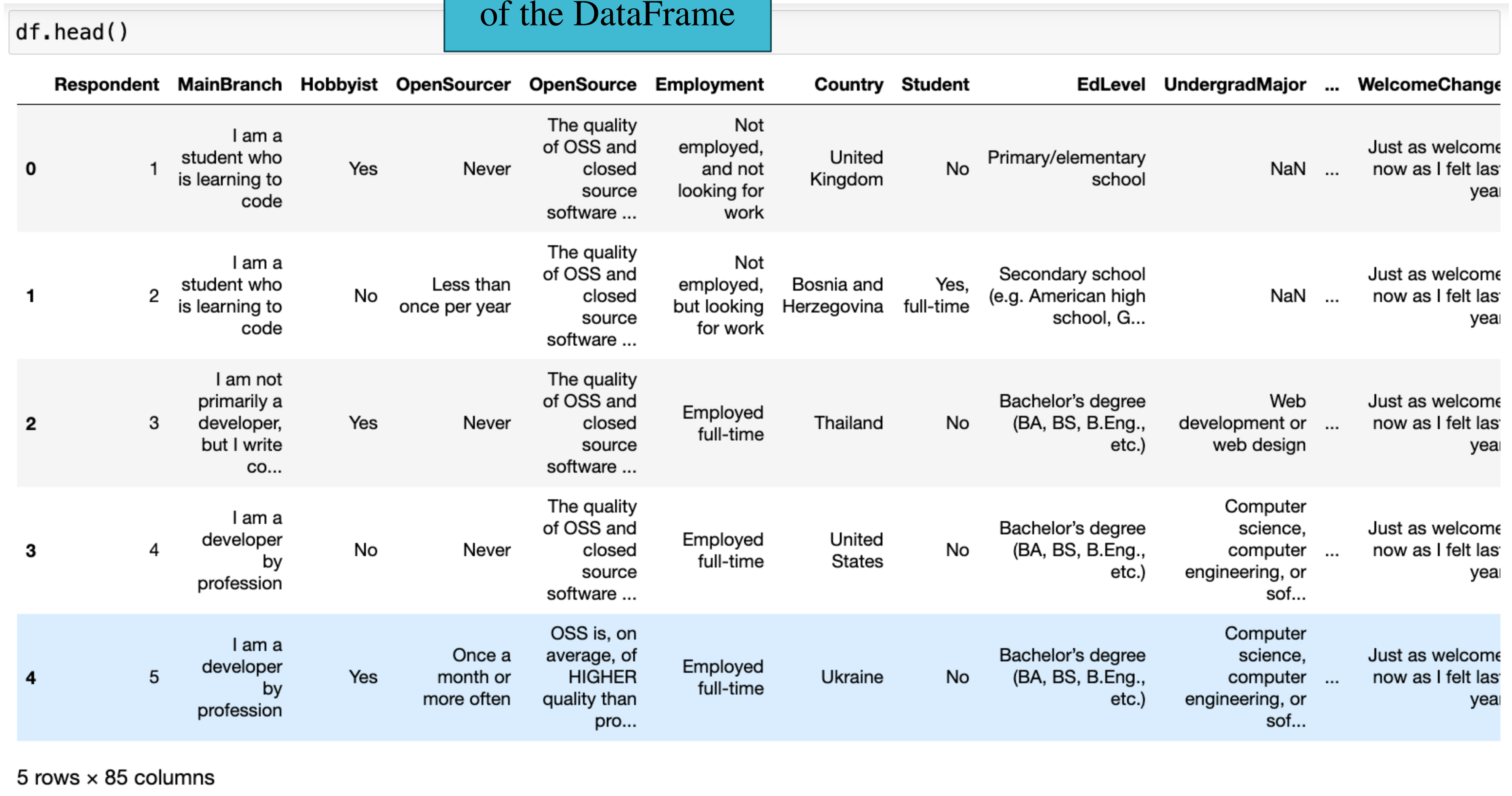
```
: df = pd.read_csv('survey_results_public_2019.csv')
```

df = pd.read_csv('survey_results_public_2019.csv')

- Reads csv into a pandas DataFrame called df
- Step is important for data analysis because it loads the dataset into memory, furthering exploration and manipulation of the data

#Reproduce the following

df.head() : displays the first five rows of the DataFrame



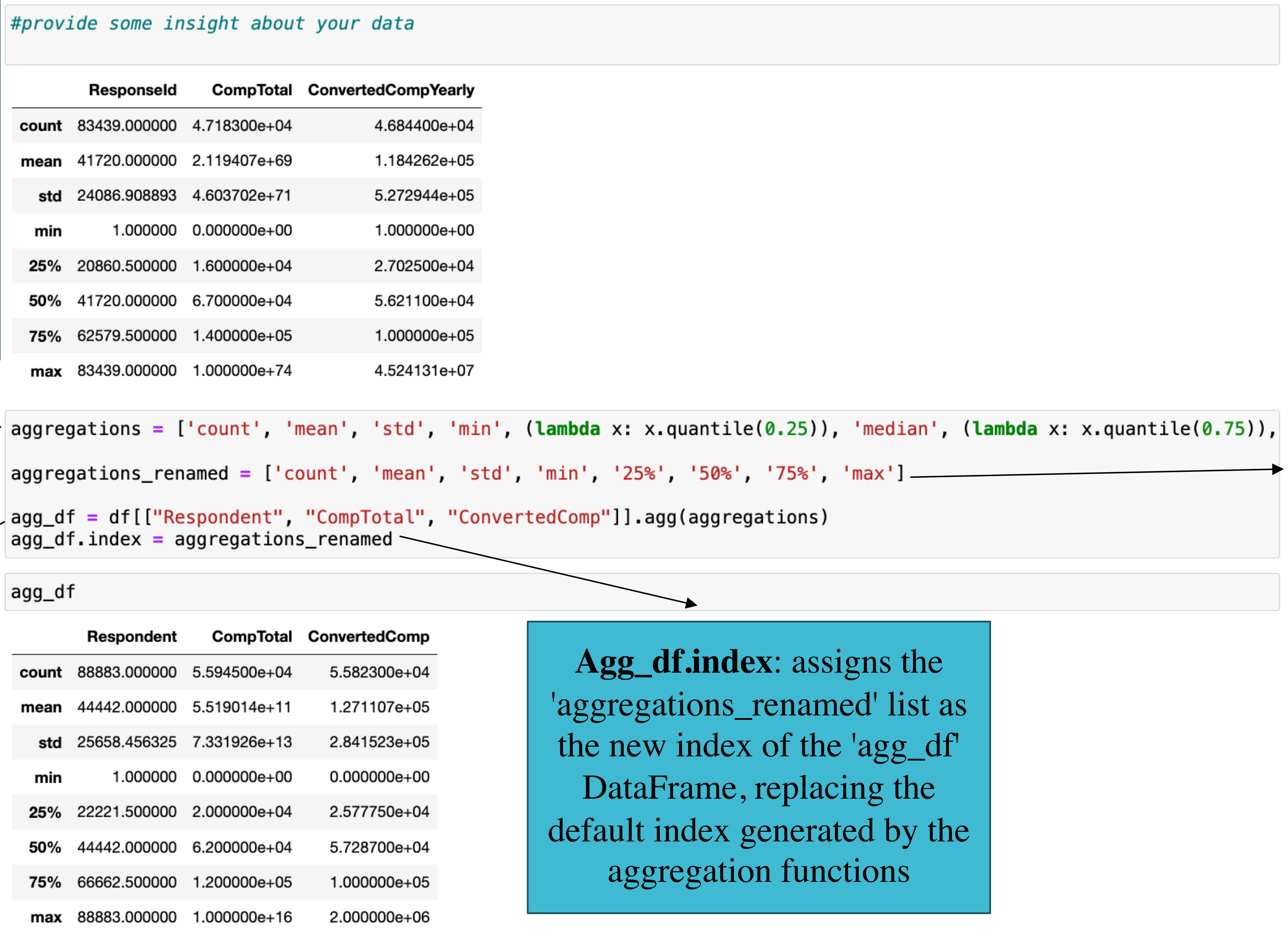
```
df.head()
```

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	Student	EdLevel	UndergradMajor	...	WelcomeChange
0	1	I am a student who is learning to code	Yes	Never	The quality of OSS and closed source software ...	Not employed, and not looking for work	United Kingdom	No	Primary/elementary school	NaN	...	Just as welcome now as I felt las yea
1	2	I am a student who is learning to code	No	Less than once per year	The quality of OSS and closed source software ...	Not employed, but looking for work	Bosnia and Herzegovina	Yes, full-time	Secondary school (e.g. American high school, G...	NaN	...	Just as welcome now as I felt las yea
2	3	I am not primarily a developer, but I write co...	Yes	Never	The quality of OSS and closed source software ...	Employed full-time	Thailand	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Web development or web design	...	Just as welcome now as I felt las yea
3	4	I am a developer by profession	No	Never	The quality of OSS and closed source software ...	Employed full-time	United States	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Computer science, computer engineering, or sof...	...	Just as welcome now as I felt las yea
4	5	I am a developer by profession	Yes	Once a month or more often	OSS is, on average, of HIGHER quality than pro...	Employed full-time	Ukraine	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Computer science, computer engineering, or sof...	...	Just as welcome now as I felt las yea

5 rows × 85 columns

Aggregations: list of aggregation functions to apply to the columns

- Custom lambda functions to calculate the 25th percentile, median (50th percentile), and 75th percentile



```
#provide some insight about your data
```

	ResponseId	CompTotal	ConvertedCompYearly
count	83439.000000	4.718300e+04	4.684400e+04
mean	41720.000000	2.119407e+69	1.184262e+05
std	24086.908893	4.603702e+71	5.272944e+05
min	1.000000	0.000000e+00	1.000000e+00
25%	20860.500000	1.600000e+04	2.702500e+04
50%	41720.000000	6.700000e+04	5.621100e+04
75%	62579.500000	1.400000e+05	1.000000e+05
max	83439.000000	1.000000e+74	4.524131e+07

```
aggregations = ['count', 'mean', 'std', 'min', (lambda x: x.quantile(0.25)), 'median', (lambda x: x.quantile(0.75)),

aggregations_renamed = ['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max']

agg_df = df[["Respondent", "CompTotal", "ConvertedComp"]].agg(aggregations)
agg_df.index = aggregations_renamed
```

```
agg_df
```

	Respondent	CompTotal	ConvertedComp
count	88883.000000	5.594500e+04	5.582300e+04
mean	44442.000000	5.519014e+11	1.271107e+05
std	25658.456325	7.331926e+13	2.841523e+05
min	1.000000	0.000000e+00	0.000000e+00
25%	22221.500000	2.000000e+04	2.577750e+04
50%	44442.000000	6.200000e+04	5.728700e+04
75%	66662.500000	1.200000e+05	1.000000e+05
max	88883.000000	1.000000e+16	2.000000e+06

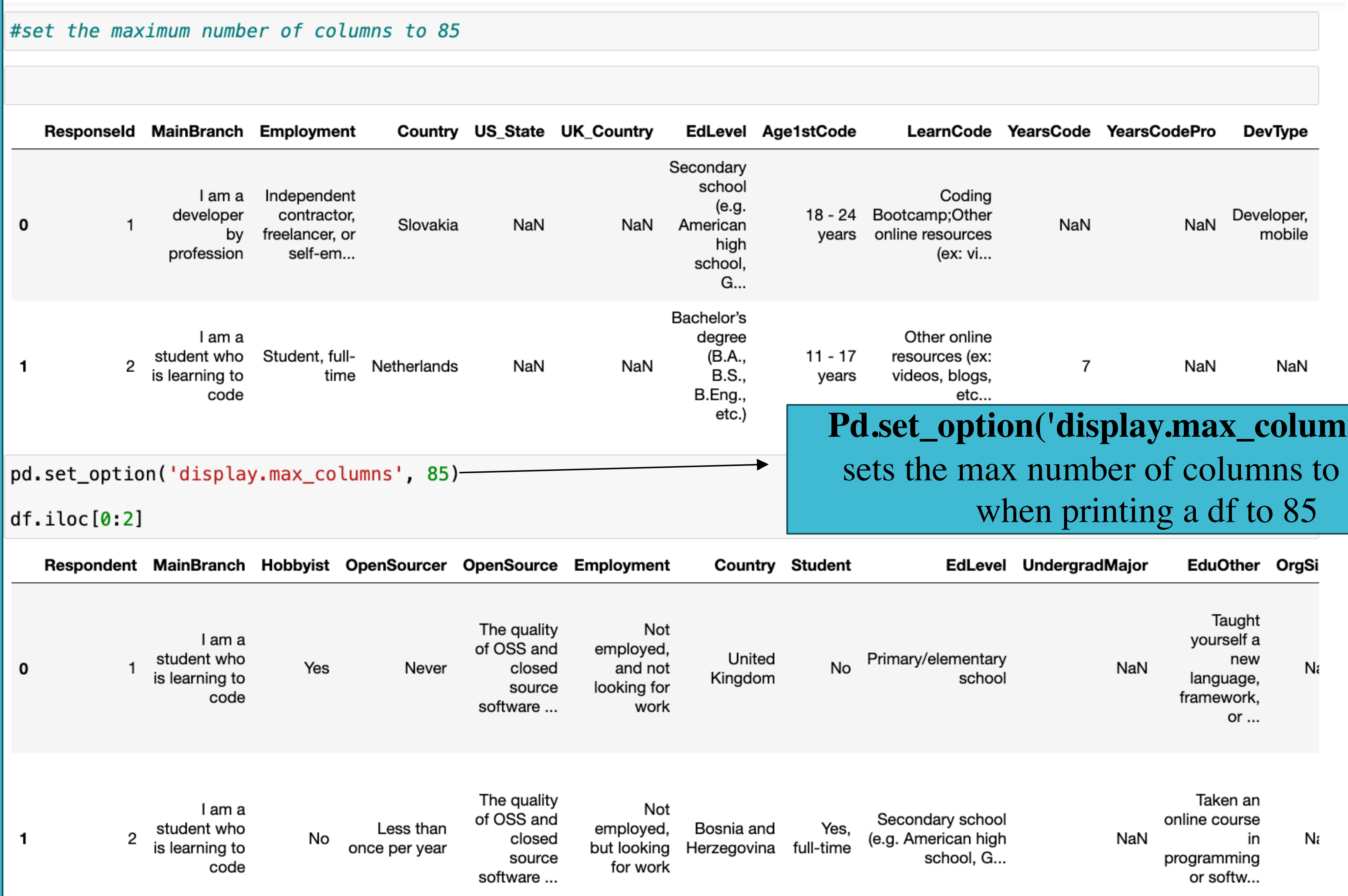
Aggregations_renamed: list of strings used to rename the index of the resulting aggregated DataFrame; names correspond to the aggregation functions for better readability

Agg_df: applies the aggregation functions specified in 'aggregations' to the different columns of the df; results in a new df containing the aggregated values

Agg_df.index: assigns the 'aggregations_renamed' list as the new index of the 'agg_df' DataFrame, replacing the default index generated by the aggregation functions

Df.iloc[0:2]: selects the first two rows of the df using the <u>iloc method</u>

- Iloc method is used for integer-location based indexing
- Returns a new df containing only the selected rows
- Combining this with the previous line ensures that the first 2 rows and all columns are displayed



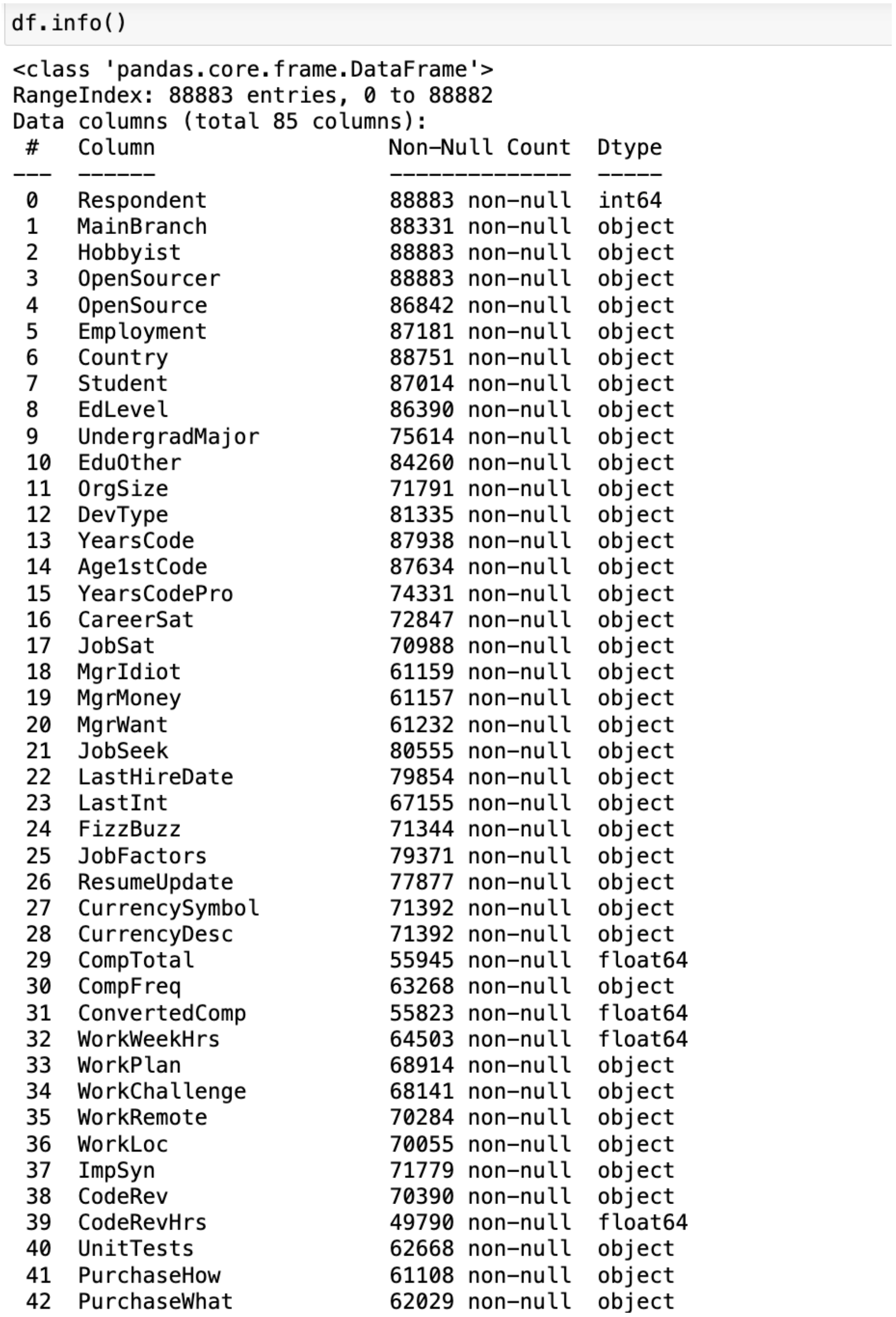
```
#set the maximum number of columns to 85
```

	ResponseId	MainBranch	Employment	Country	US_State	UK_Country	EdLevel	Age1stCode	LearnCode	YearsCode	YearsCodePro	DevType
0	1	I am a developer by profession	Independent contractor, freelancer, or self-em...	Slovakia	NaN	NaN	Secondary school (e.g. American high school, G...	18 - 24 years	Coding Bootcamp;Other online resources (ex: vi...	NaN	NaN	Developer, mobile
1	2	I am a student who is learning to code	Student, full-time	Netherlands	NaN	NaN	Bachelor's degree (B.A., B.S., B.Eng., etc.)	11 - 17 years	Other online resources (ex: videos, blogs, etc...	7	NaN	NaN

```
pd.set_option('display.max_columns', 85)

df.iloc[0:2]
```

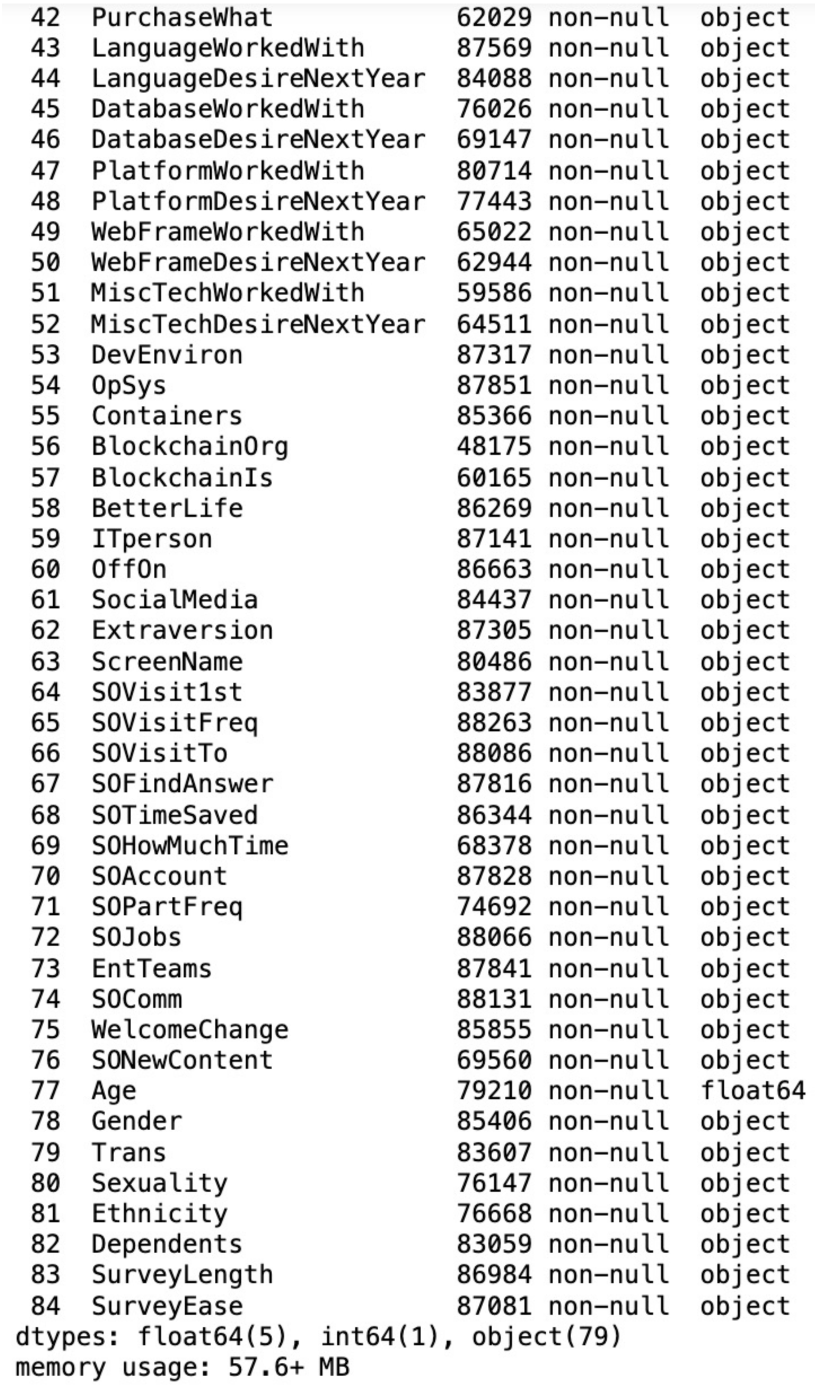
	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	Student	EdLevel	UndergradMajor	EduOther	OrgSi
0	1	I am a student who is learning to code	Yes	Never	The quality of OSS and closed source software ...	Not employed, and not looking for work	United Kingdom	No	Primary/elementary school	NaN	Taught yourself a new language, framework, or ...	Na
1	2	I am a student who is learning to code	No	Less than once per year	The quality of OSS and closed source software ...	Not employed, but looking for work	Bosnia and Herzegovina	Yes, full-time	Secondary school (e.g. American high school, G...	NaN	Taken an online course in programming or softw...	Na

Pd.set_option('display.max_columns', 85): sets the max number of columns to display when printing a df to 85



```
df.info()
```

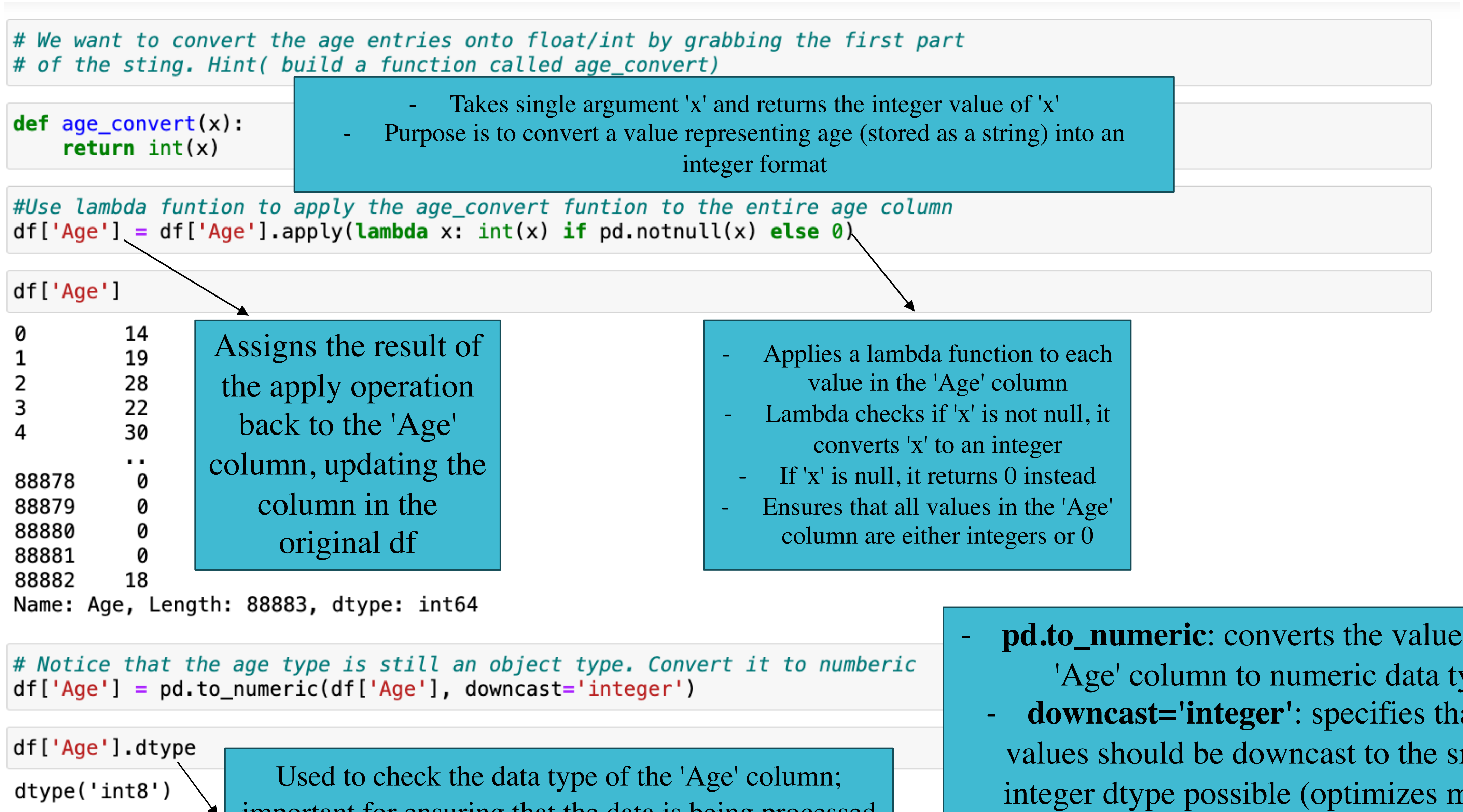
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88883 entries, 0 to 88882
Data columns (total 85 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Respondent              88883 non-null  int64
 1   MainBranch              88331 non-null  object
 2   Hobbyist                88883 non-null  object
 3   OpenSourcer             88883 non-null  object
 4   OpenSource              86842 non-null  object
 5   Employment              87181 non-null  object
 6   Country                 88751 non-null  object
 7   Student                 87014 non-null  object
 8   EdLevel                 86390 non-null  object
 9   UndergradMajor          75614 non-null  object
 10  EduOther                84260 non-null  object
 11  OrgSize                 71791 non-null  object
 12  DevType                 81335 non-null  object
 13  YearsCode               87938 non-null  object
 14  Age1stCode              87634 non-null  object
 15  YearsCodePro            74331 non-null  object
 16  CareerSat               72847 non-null  object
 17  JobSat                  70988 non-null  object
 18  MgrIdiot                61159 non-null  object
 19  MgrMoney                61157 non-null  object
 20  MgrWant                 61232 non-null  object
 21  JobSeek                 80555 non-null  object
 22  LastHireDate            79854 non-null  object
 23  LastInt                 67155 non-null  object
 24  FizzBuzz                71344 non-null  object
 25  JobFactors              79371 non-null  object
 26  ResumeUpdate            77877 non-null  object
 27  CurrencySymbol          71392 non-null  object
 28  CurrencyDesc            71392 non-null  object
 29  CompTotal               55945 non-null  float64
 30  CompFreq                63268 non-null  object
 31  ConvertedComp           55823 non-null  float64
 32  WorkWeekHrs             64503 non-null  float64
 33  WorkPlan                68914 non-null  object
 34  WorkChallenge           68141 non-null  object
 35  WorkRemote              70284 non-null  object
 36  WorkLoc                 70055 non-null  object
 37  ImpSyn                  71779 non-null  object
 38  CodeRev                 70390 non-null  object
 39  CodeRevHrs              49790 non-null  float64
 40  UnitTests               62668 non-null  object
 41  PurchaseHow             61108 non-null  object
 42  PurchaseWhat            62029 non-null  object
```



```
 42  PurchaseWhat            62029 non-null  object
 43  LanguageWorkedWith      87569 non-null  object
 44  LanguageDesireNextYear  84088 non-null  object
 45  DatabaseWorkedWith      76026 non-null  object
 46  DatabaseDesireNextYear  69147 non-null  object
 47  PlatformWorkedWith      80714 non-null  object
 48  PlatformDesireNextYear  77443 non-null  object
 49  WebFrameWorkedWith      65022 non-null  object
 50  WebFrameDesireNextYear  62944 non-null  object
 51  MiscTechWorkedWith      59586 non-null  object
 52  MiscTechDesireNextYear  64511 non-null  object
 53  DevEnviron              87317 non-null  object
 54  OpSys                   87851 non-null  object
 55  Containers              85366 non-null  object
 56  BlockchainOrg           48175 non-null  object
 57  BlockchainIs            60165 non-null  object
 58  BetterLife              86269 non-null  object
 59  ITperson                87141 non-null  object
 60  OffOn                   86663 non-null  object
 61  SocialMedia             84437 non-null  object
 62  Extraversion            87305 non-null  object
 63  ScreenName              80486 non-null  object
 64  SOVisit1st              83877 non-null  object
 65  SOVisitFreq             88263 non-null  object
 66  SOVisitTo               88086 non-null  object
 67  SOFindAnswer            87816 non-null  object
 68  SOTimeSaved             86344 non-null  object
 69  SOHowMuchTime           68378 non-null  object
 70  SOAccount               87828 non-null  object
 71  SOPartFreq              74692 non-null  object
 72  SOJobs                  88066 non-null  object
 73  EntTeams                87841 non-null  object
 74  SOComm                  88131 non-null  object
 75  WelcomeChange           85855 non-null  object
 76  SONewContent            69560 non-null  object
 77  Age                     79210 non-null  float64
 78  Gender                  85406 non-null  object
 79  Trans                   83607 non-null  object
 80  Sexuality               76147 non-null  object
 81  Ethnicity               76668 non-null  object
 82  Dependents              83059 non-null  object
 83  SurveyLength            86984 non-null  object
 84  SurveyEase              87081 non-null  object
dtypes: float64(5), int64(1), object(79)
memory usage: 57.6+ MB
```

df.info(): summarizes the df

- Number of rows and columns
- Column names
- Data types of each column
- Number of non-null values in each column



```
# We want to convert the age entries onto float/int by grabbing the first part
# of the sting. Hint( build a function called age_convert)
```

```
def age_convert(x):
    return int(x)
```

- Takes single argument 'x' and returns the integer value of 'x'
- Purpose is to convert a value representing age (stored as a string) into an integer format

```
#Use lambda funtion to apply the age_convert funtion to the entire age column
df['Age'] = df['Age'].apply(lambda x: int(x) if pd.notnull(x) else 0)
```

```
df['Age']
```

```
0        14
1        19
2        28
3        22
4        30
         ..
88878     0
88879     0
88880     0
88881     0
88882    18
Name: Age, Length: 88883, dtype: int64
```

Assigns the result of the apply operation back to the 'Age' column, updating the column in the original df

- Applies a lambda function to each value in the 'Age' column
- Lambda checks if 'x' is not null, it converts 'x' to an integer
- If 'x' is null, it returns 0 instead
- Ensures that all values in the 'Age' column are either integers or 0

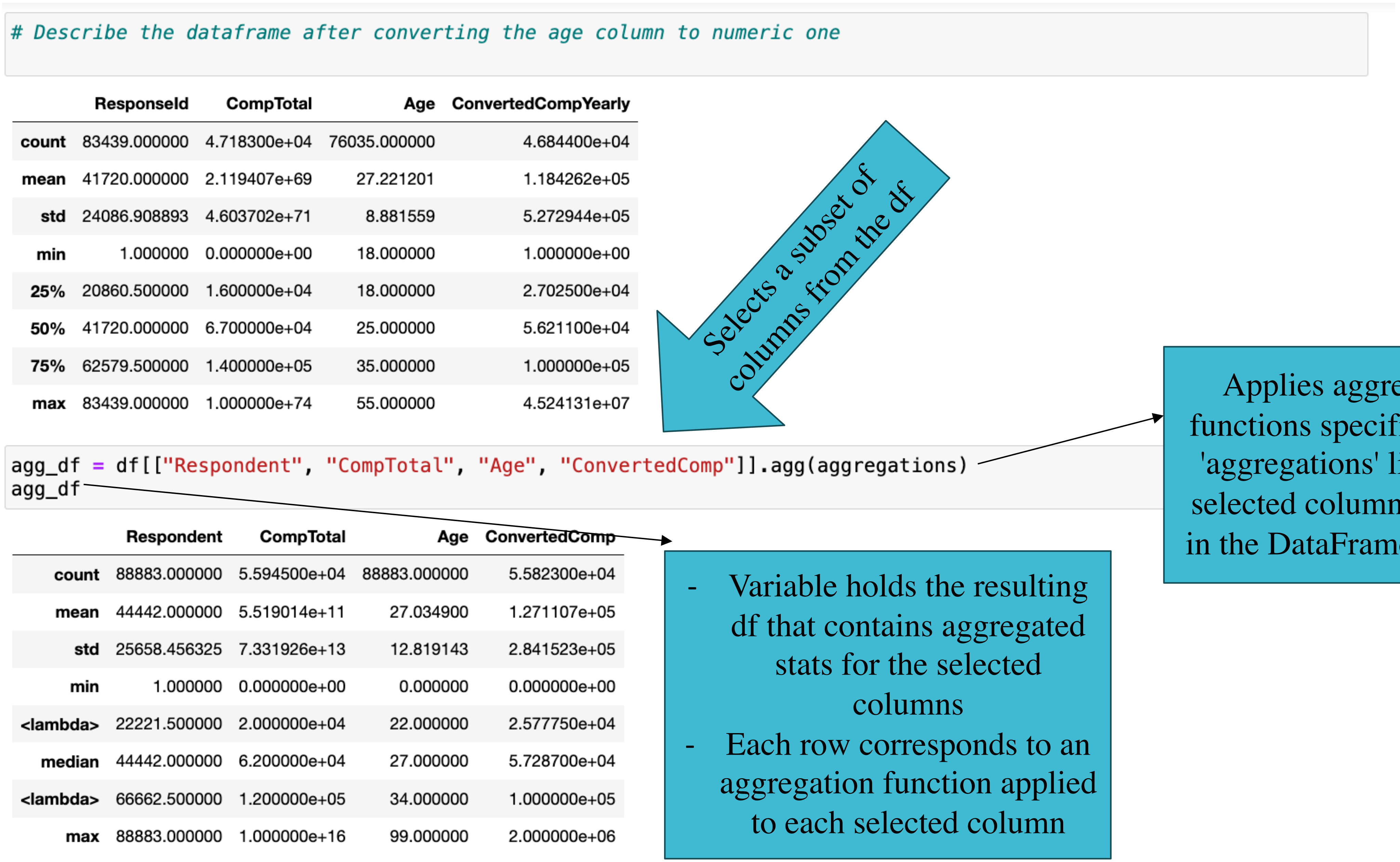
```
# Notice that the age type is still an object type. Convert it to numberic
df['Age'] = pd.to_numeric(df['Age'], downcast='integer')
```

- **pd.to_numeric**: converts the values in the 'Age' column to numeric data type
- **downcast='integer'**: specifies that the values should be downcast to the smallest integer dtype possible (optimizes memory storage)

```
df['Age'].dtype
```

```
dtype('int8')
```

Used to check the data type of the 'Age' column; important for ensuring that the data is being processed correctly and for understanding how the data is stored in the df



```python
# Describe the dataframe after converting the age column to numeric one
```

	ResponseId	CompTotal	Age	ConvertedCompYearly
count	83439.000000	4.718300e+04	76035.000000	4.684400e+04
mean	41720.000000	2.119407e+69	27.221201	1.184262e+05
std	24086.908893	4.603702e+71	8.881559	5.272944e+05
min	1.000000	0.000000e+00	18.000000	1.000000e+00
25%	20860.500000	1.600000e+04	18.000000	2.702500e+04
50%	41720.000000	6.700000e+04	25.000000	5.621100e+04
75%	62579.500000	1.400000e+05	35.000000	1.000000e+05
max	83439.000000	1.000000e+74	55.000000	4.524131e+07

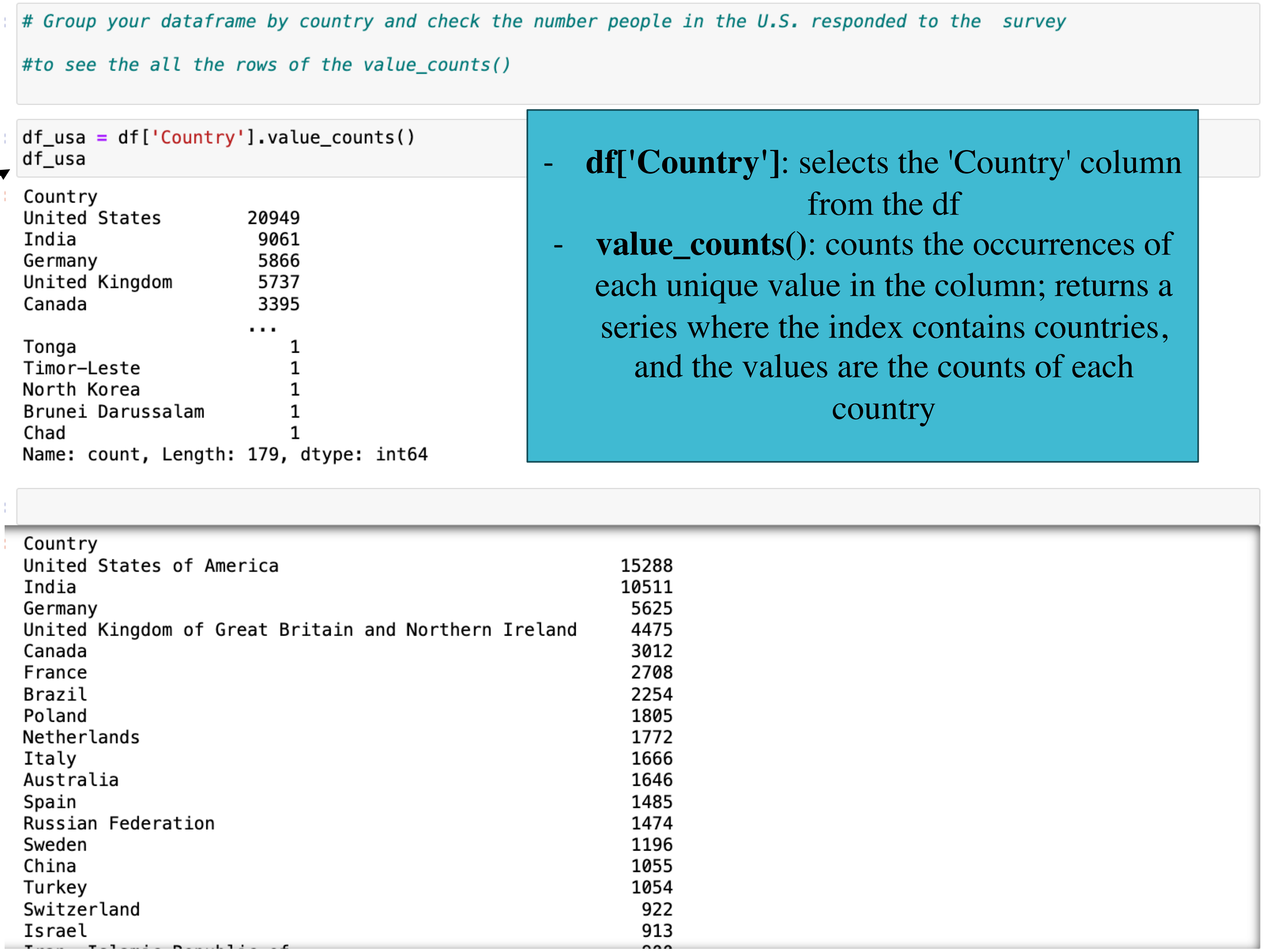
Selects a subset of columns from the df

```python
agg_df = df[["Respondent", "CompTotal", "Age", "ConvertedComp"]].agg(aggregations)
agg_df
```

	Respondent	CompTotal	Age	ConvertedComp
count	88883.000000	5.594500e+04	88883.000000	5.582300e+04
mean	44442.000000	5.519014e+11	27.034900	1.271107e+05
std	25658.456325	7.331926e+13	12.819143	2.841523e+05
min	1.000000	0.000000e+00	0.000000	0.000000e+00
<lambda>	22221.500000	2.000000e+04	22.000000	2.577750e+04
median	44442.000000	6.200000e+04	27.000000	5.728700e+04
<lambda>	66662.500000	1.200000e+05	34.000000	1.000000e+05
max	88883.000000	1.000000e+16	99.000000	2.000000e+06

Applies aggregation functions specified in the 'aggregations' list to the selected columns; results in the DataFrame 'agg_df'

- Variable holds the resulting df that contains aggregated stats for the selected columns
- Each row corresponds to an aggregation function applied to each selected column



```
# Group your dataframe by country and check the number people in the U.S. responded to the  survey

#to see the all the rows of the value_counts()
```

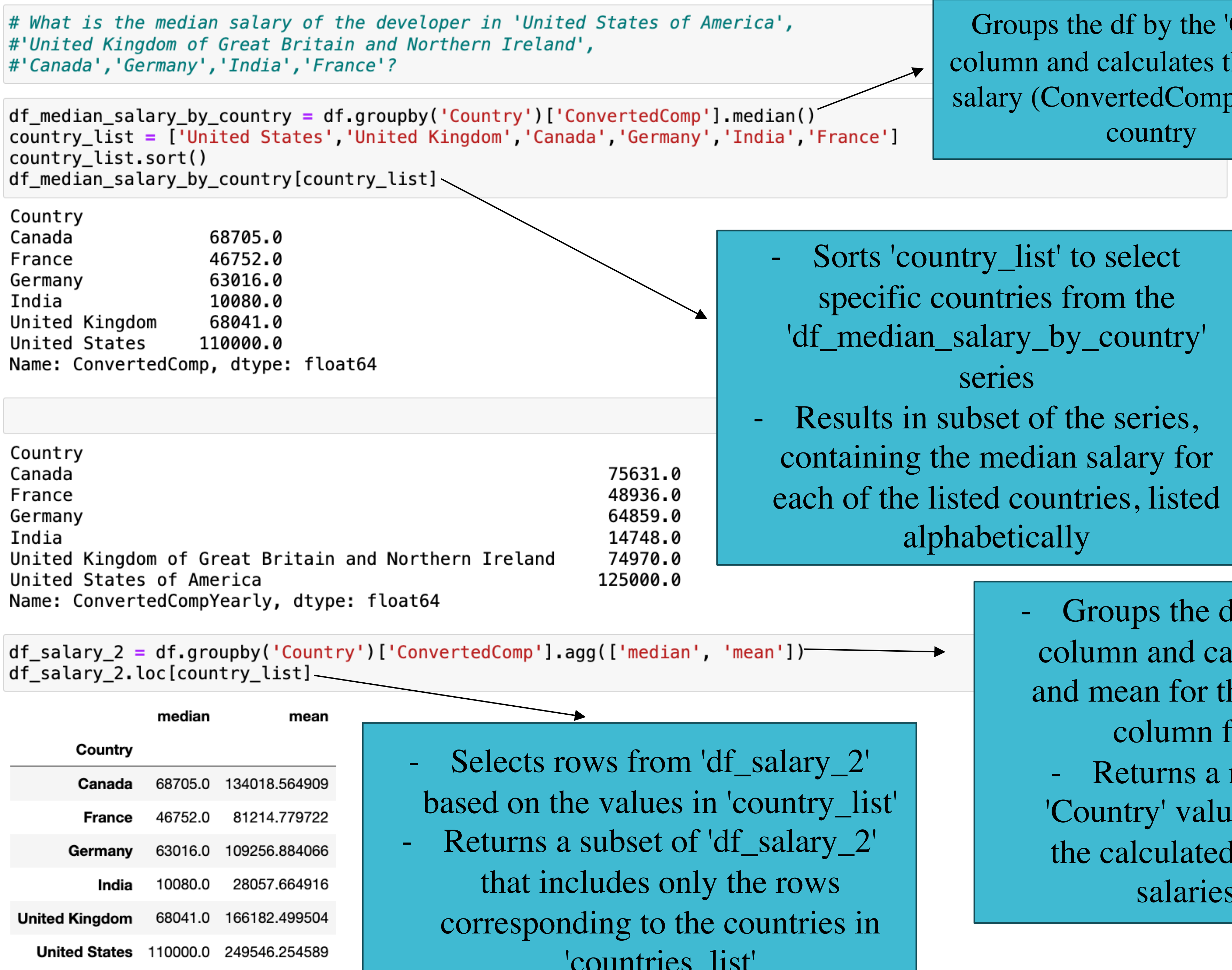
```
df_usa = df['Country'].value_counts()
df_usa
```

```
Country
United States        20949
India                 9061
Germany               5866
United Kingdom        5737
Canada                3395
                      ...
Tonga                    1
Timor-Leste              1
North Korea              1
Brunei Darussalam        1
Chad                     1
Name: count, Length: 179, dtype: int64
```

```
Country
United States of America                                 15288
India                                                    10511
Germany                                                   5625
United Kingdom of Great Britain and Northern Ireland      4475
Canada                                                    3012
France                                                    2708
Brazil                                                    2254
Poland                                                    1805
Netherlands                                               1772
Italy                                                     1666
Australia                                                 1646
Spain                                                     1485
Russian Federation                                        1474
Sweden                                                    1196
China                                                     1055
Turkey                                                    1054
Switzerland                                                922
Israel                                                     913
```

- **df['Country']**: selects the 'Country' column from the df
- **value_counts()**: counts the occurrences of each unique value in the column; returns a series where the index contains countries, and the values are the counts of each country

Variable holds the resulting series; represents the count of respondents from each country



```
# What is the median salary of the developer in 'United States of America',
#'United Kingdom of Great Britain and Northern Ireland',
#'Canada','Germany','India','France'?
```

```
df_median_salary_by_country = df.groupby('Country')['ConvertedComp'].median()
country_list = ['United States','United Kingdom','Canada','Germany','India','France']
country_list.sort()
df_median_salary_by_country[country_list]
```

```
Country
Canada             68705.0
France             46752.0
Germany            63016.0
India              10080.0
United Kingdom     68041.0
United States     110000.0
Name: ConvertedComp, dtype: float64
```

```
Country
Canada                                                   75631.0
France                                                   48936.0
Germany                                                  64859.0
India                                                    14748.0
United Kingdom of Great Britain and Northern Ireland     74970.0
United States of America                                125000.0
Name: ConvertedCompYearly, dtype: float64
```

```
df_salary_2 = df.groupby('Country')['ConvertedComp'].agg(['median', 'mean'])
df_salary_2.loc[country_list]
```

	median	mean
Country		
Canada	68705.0	134018.564909
France	46752.0	81214.779722
Germany	63016.0	109256.884066
India	10080.0	28057.664916
United Kingdom	68041.0	166182.499504
United States	110000.0	249546.254589

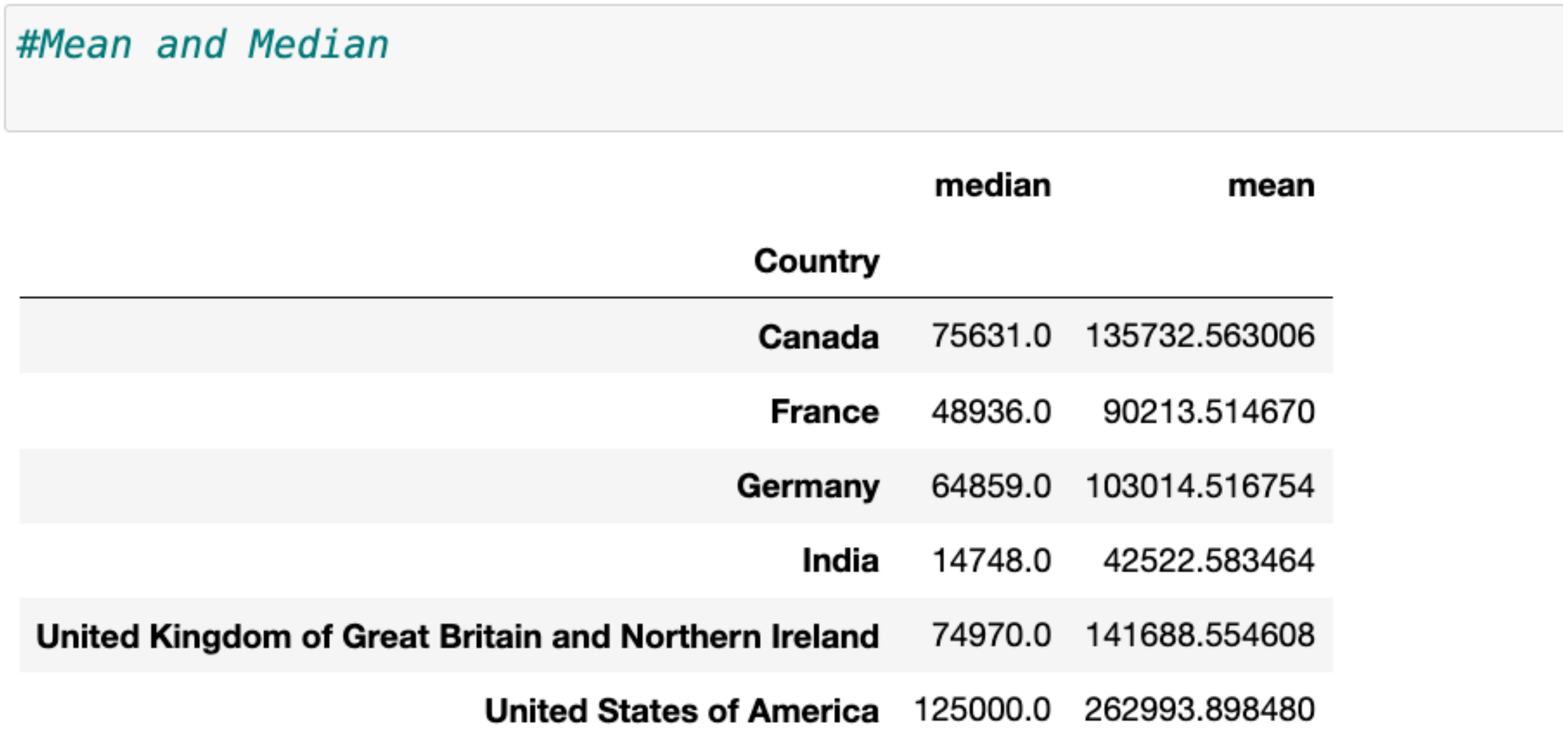
Groups the df by the 'Country' column and calculates the median salary (ConvertedComp) for each country

Sorts list alphabetically

- Sorts 'country_list' to select specific countries from the 'df_median_salary_by_country' series
- Results in subset of the series, containing the median salary for each of the listed countries, listed alphabetically

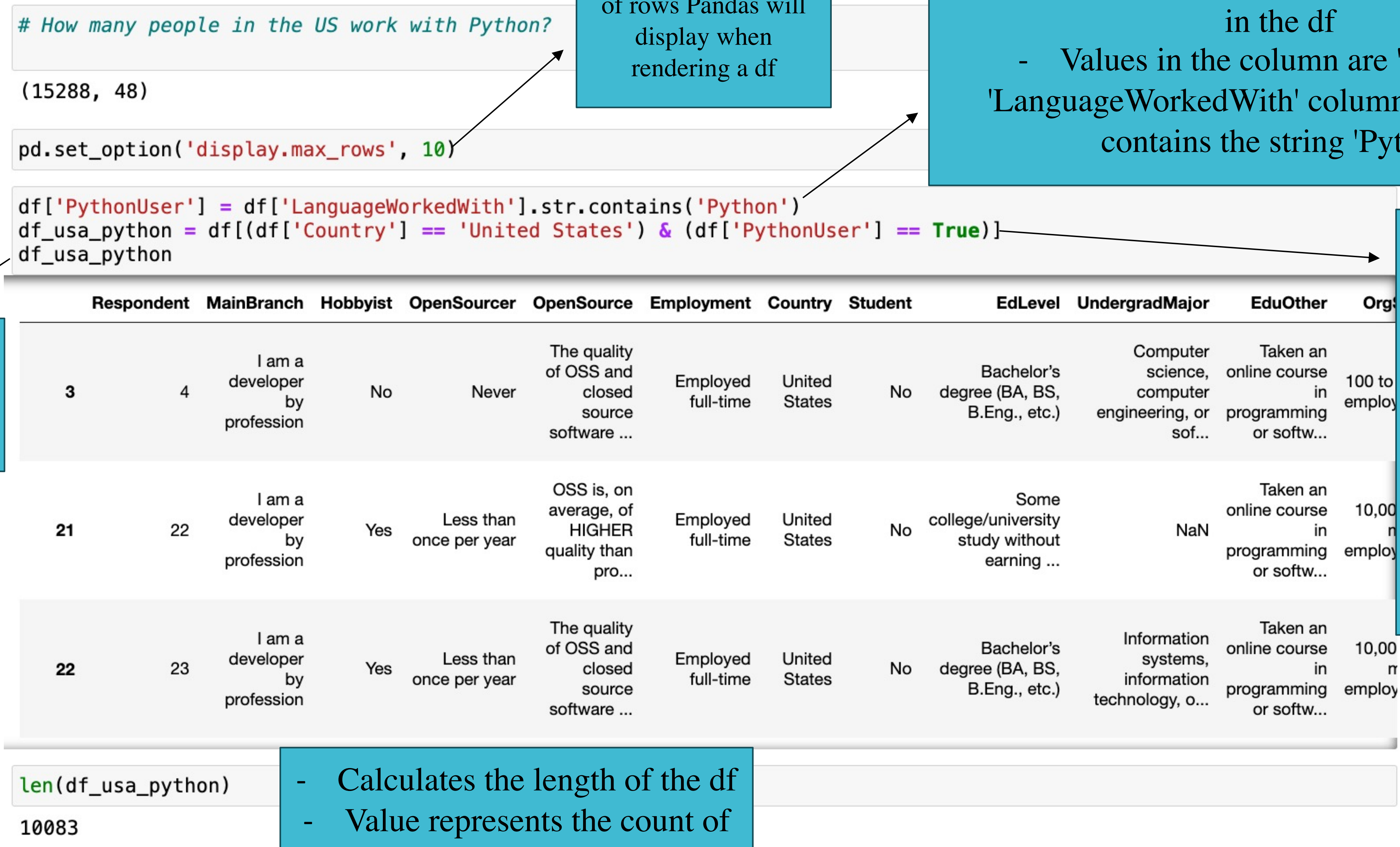
- Groups the df by the 'Country' column and calculates the median and mean for the 'ConvertedComp' column for each group
- Returns a new df with the 'Country' values as the index and the calculated median and mean salaries as columns

- Selects rows from 'df_salary_2' based on the values in 'country_list'
- Returns a subset of 'df_salary_2' that includes only the rows corresponding to the countries in 'countries_list'



```
#Mean and Median
```

Country	median	mean
Canada	75631.0	135732.563006
France	48936.0	90213.514670
Germany	64859.0	103014.516754
India	14748.0	42522.583464
United Kingdom of Great Britain and Northern Ireland	74970.0	141688.554608
United States of America	125000.0	262993.898480



```
# How many people in the US work with Python?
```

```
(15288, 48)
```

```
pd.set_option('display.max_rows', 10)
```

10 is the max number of rows Pandas will display when rendering a df

```
df['PythonUser'] = df['LanguageWorkedWith'].str.contains('Python')
df_usa_python = df[(df['Country'] == 'United States') & (df['PythonUser'] == True)]
df_usa_python
```

- Creates a new column called 'PythonUser' in the df
- Values in the column are 'True' is 'LanguageWorkedWith' column for that row contains the string 'Python'

Filters the df to only include rows where the 'Country' column is 'United States' and the 'PythonUser' column is 'True'

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	Student	EdLevel	UndergradMajor	EduOther	Org
3	4	I am a developer by profession	No	Never	The quality of OSS and closed source software ...	Employed full-time	United States	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Computer science, computer engineering, or sof...	Taken an online course in programming or softw...	100 to employ
21	22	I am a developer by profession	Yes	Less than once per year	OSS is, on average, of HIGHER quality than pro...	Employed full-time	United States	No	Some college/university study without earning ...	NaN	Taken an online course in programming or softw...	10,00 n employ
22	23	I am a developer by profession	Yes	Less than once per year	The quality of OSS and closed source software ...	Employed full-time	United States	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Information systems, information technology, o...	Taken an online course in programming or softw...	10,00 m employ

Displays the filtered df

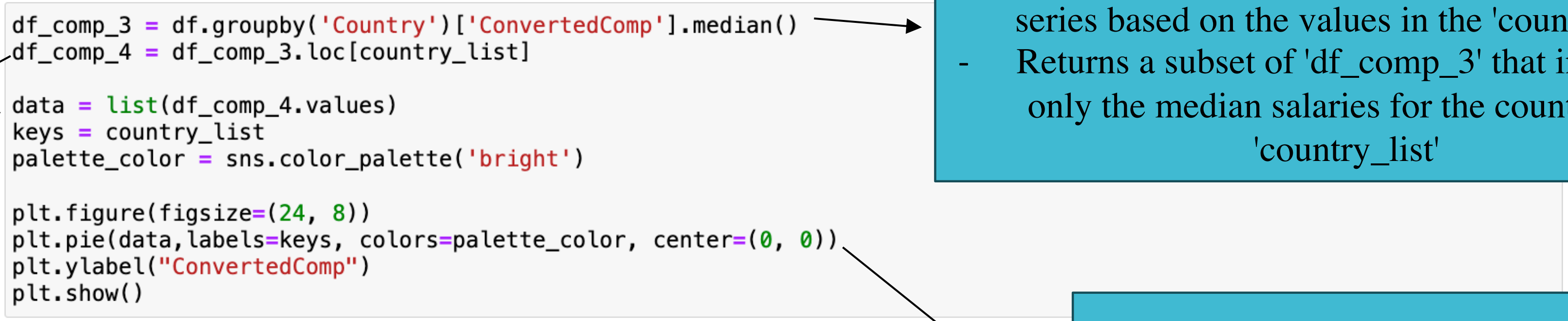
```
len(df_usa_python)
```

```
10083
```

- Calculates the length of the df
- Value represents the count of users from the US who have worked with Python

#Reproduce the following

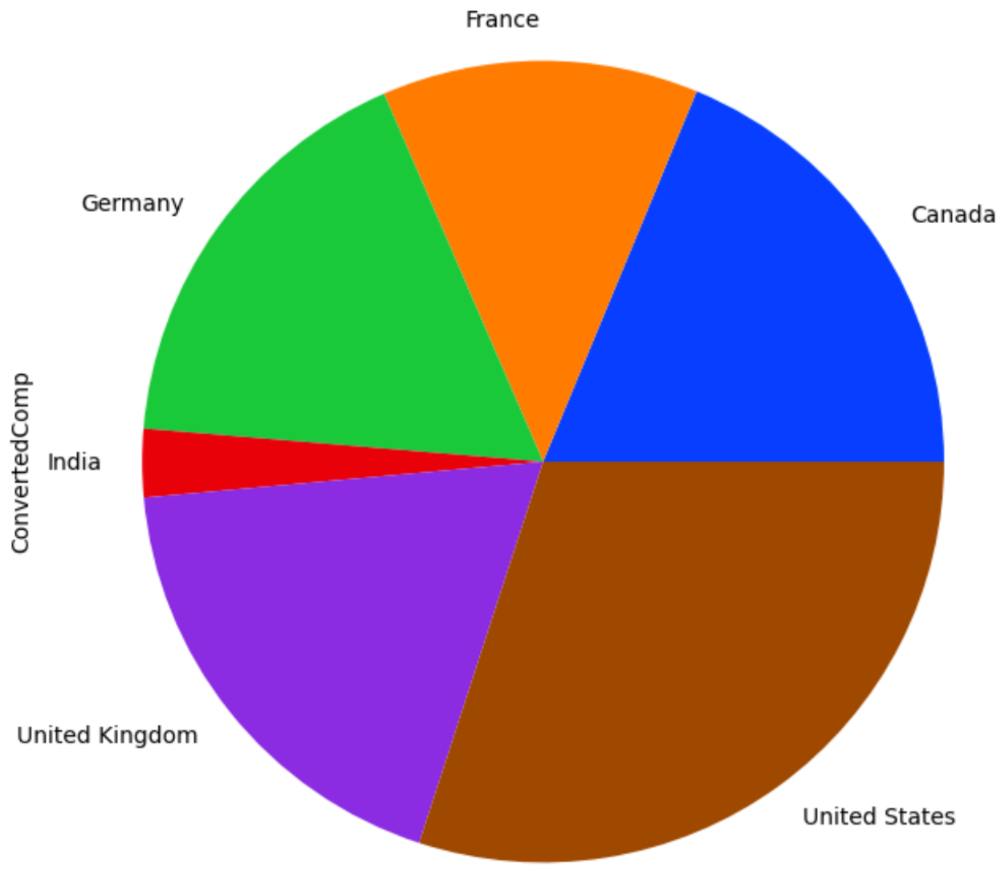
- Groups the df by the 'Country' column and calculates the median of the 'ConvertedComp' column for each group
- Returns a series with the 'Country' values as the index and the calculated median salaries as the values



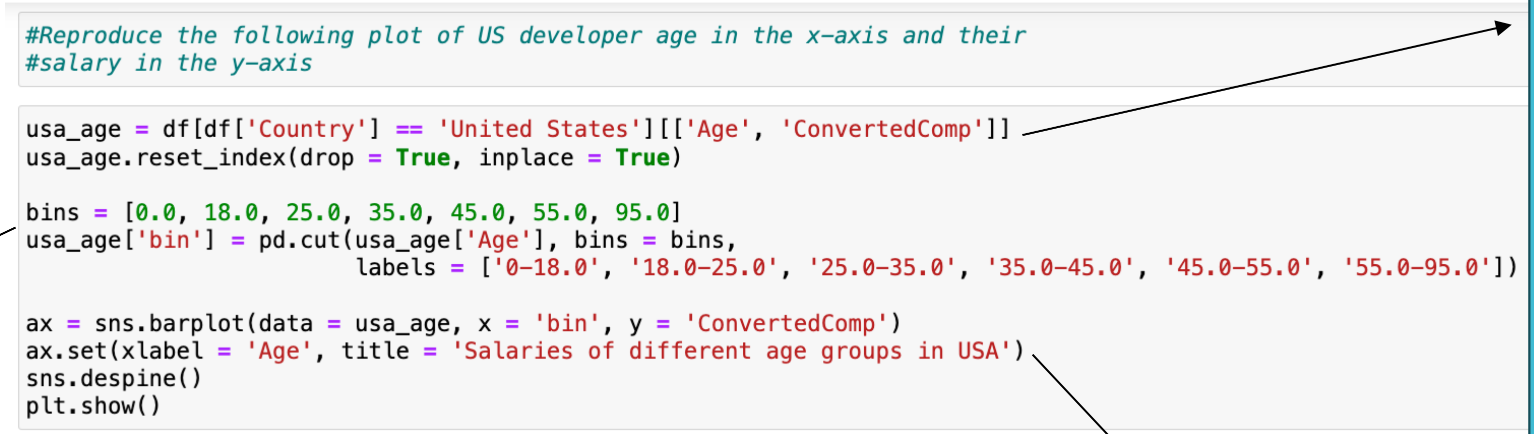
```
df_comp_3 = df.groupby('Country')['ConvertedComp'].median()
df_comp_4 = df_comp_3.loc[country_list]

data = list(df_comp_4.values)
keys = country_list
palette_color = sns.color_palette('bright')

plt.figure(figsize=(24, 8))
plt.pie(data,labels=keys, colors=palette_color, center=(0, 0))
plt.ylabel("ConvertedComp")
plt.show()
```

- Selects specific rows from the 'df_comp_3' series based on the values in the 'country_list'
- Returns a subset of 'df_comp_3' that includes only the median salaries for the countries in 'country_list'

- **Data**: extracts the values (median salaries) from the series 'df_comp_4' and converts them into a list
- **Keys**: assigns the 'country_list' to the variable keys; used as the labels for the slices of the pie chart
- **Palette_color**: uses Seaborn's 'color_palette' function to assign a different color to each slice of the chart

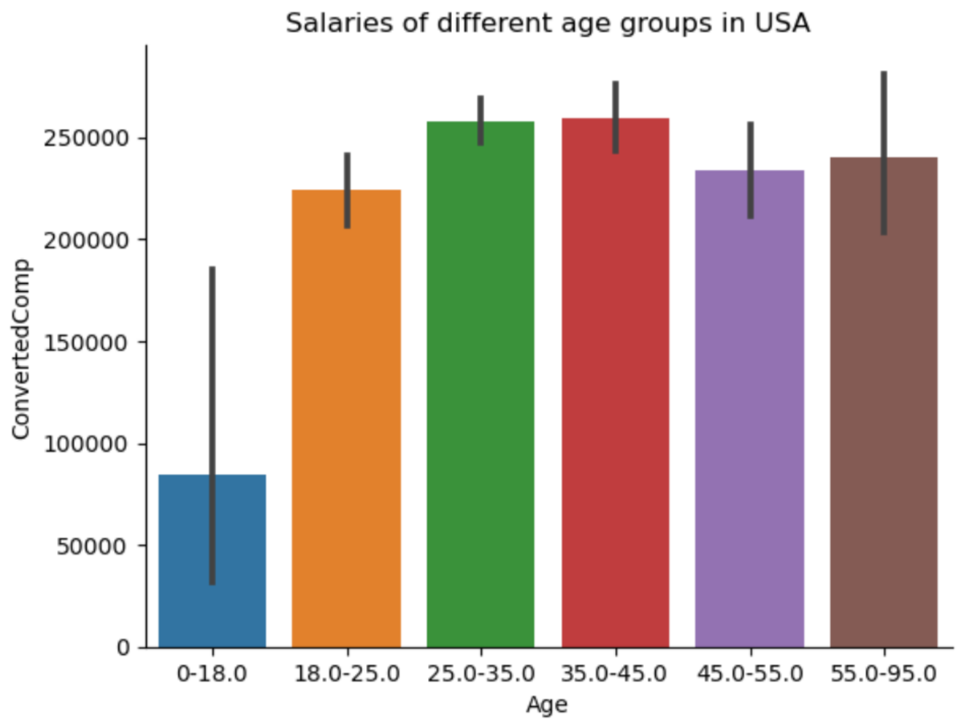


- **Plt.figure...**: creates a new figure for the plot with a specified size
- **Plt.pie...**: creates the pie chart; takes the 'data' list as the data to be plotted, 'keys' as the labels for each slice, 'palette_color' as the colors for each slice, and 'center=(0,0)' to center the pie at the origin (0,0) of the plot
- **Plt.ylabel...**: sets the label for the y-axis as 'ConvertedComp'
- **Plt.show()**: displays the pie chart



```
#Reproduce the following plot of US developer age in the x-axis and their
#salary in the y-axis
```

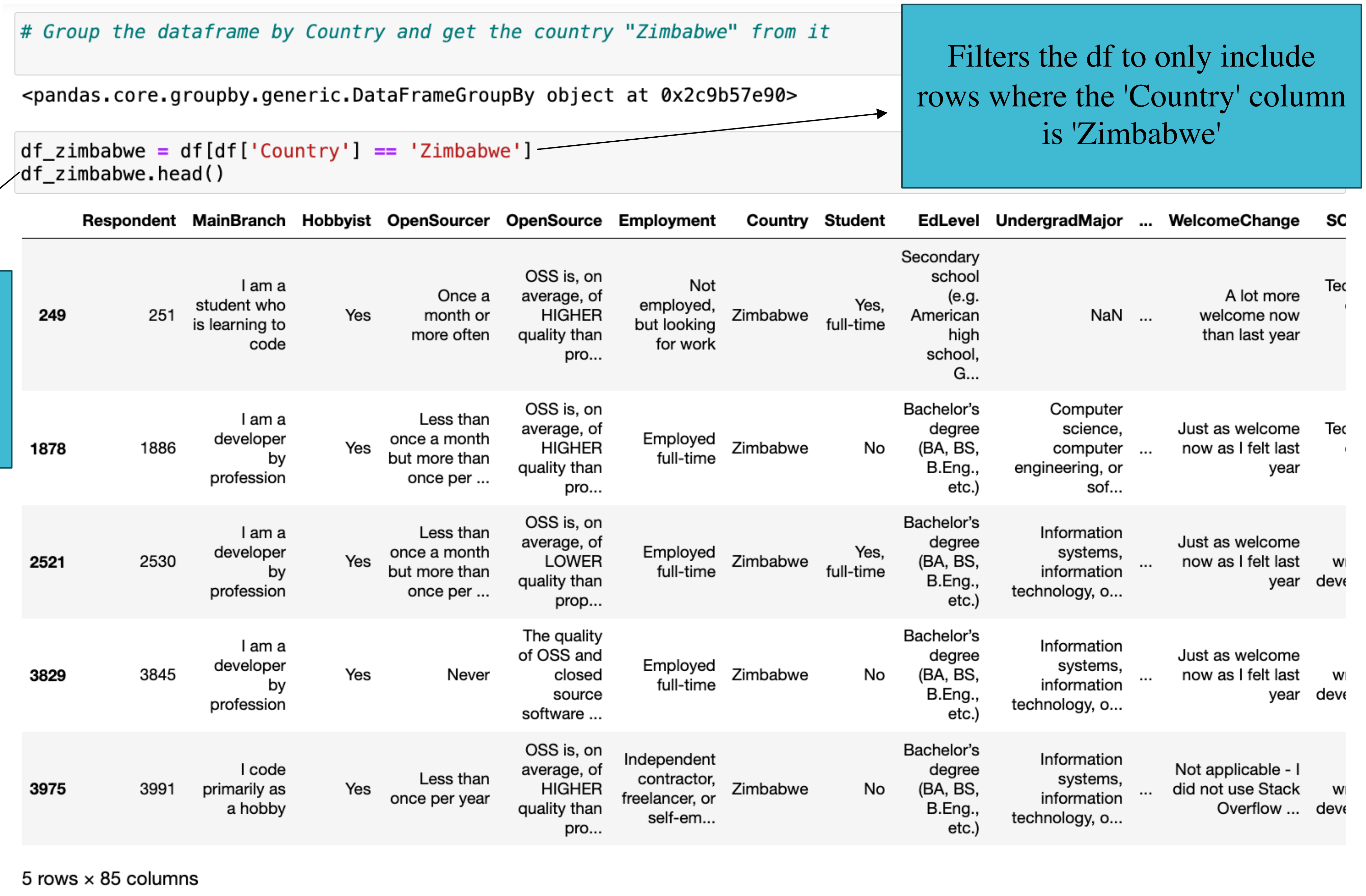
```
usa_age = df[df['Country'] == 'United States'][['Age', 'ConvertedComp']]
usa_age.reset_index(drop = True, inplace = True)

bins = [0.0, 18.0, 25.0, 35.0, 45.0, 55.0, 95.0]
usa_age['bin'] = pd.cut(usa_age['Age'], bins = bins,
                        labels = ['0-18.0', '18.0-25.0', '25.0-35.0', '35.0-45.0', '45.0-55.0', '55.0-95.0'])

ax = sns.barplot(data = usa_age, x = 'bin', y = 'ConvertedComp')
ax.set(xlabel = 'Age', title = 'Salaries of different age groups in USA')
sns.despine()
plt.show()
```



- **Usa_age…**: df that contains only the 'Age' and 'ConvertedComp' columns for users from the US
- **Usa_age.reset…**: resets the index of the df 'usa_age'
  - '**drop=True**': used to drop the previous index
  - '**inplace=True**': used to modify the 'usa_age' df in place

- **Bins**: defines the bin edges for catagorizing ages
- **Usa_age['bin']**: creates a new column called 'bin' in the 'usa_age' df; categorizes the 'Age' column values into specified bins using the 'bins' list, assigning corresponding labels to each bin

- **Ax**: creates a bar plot; specifies the 'usa_age' df as the data source, 'bin' as the x=axis variable, and 'ConvertedComp' as the y-axis variable
- **Ax.set…**: sets the x-axis label and plot title
- **Sns.despine()**: removes the top and right spines from the plot
- **Plt.show()**: displays the plot



```python
# Group the dataframe by Country and get the country "Zimbabwe" from it
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x2c9b57e90>
```

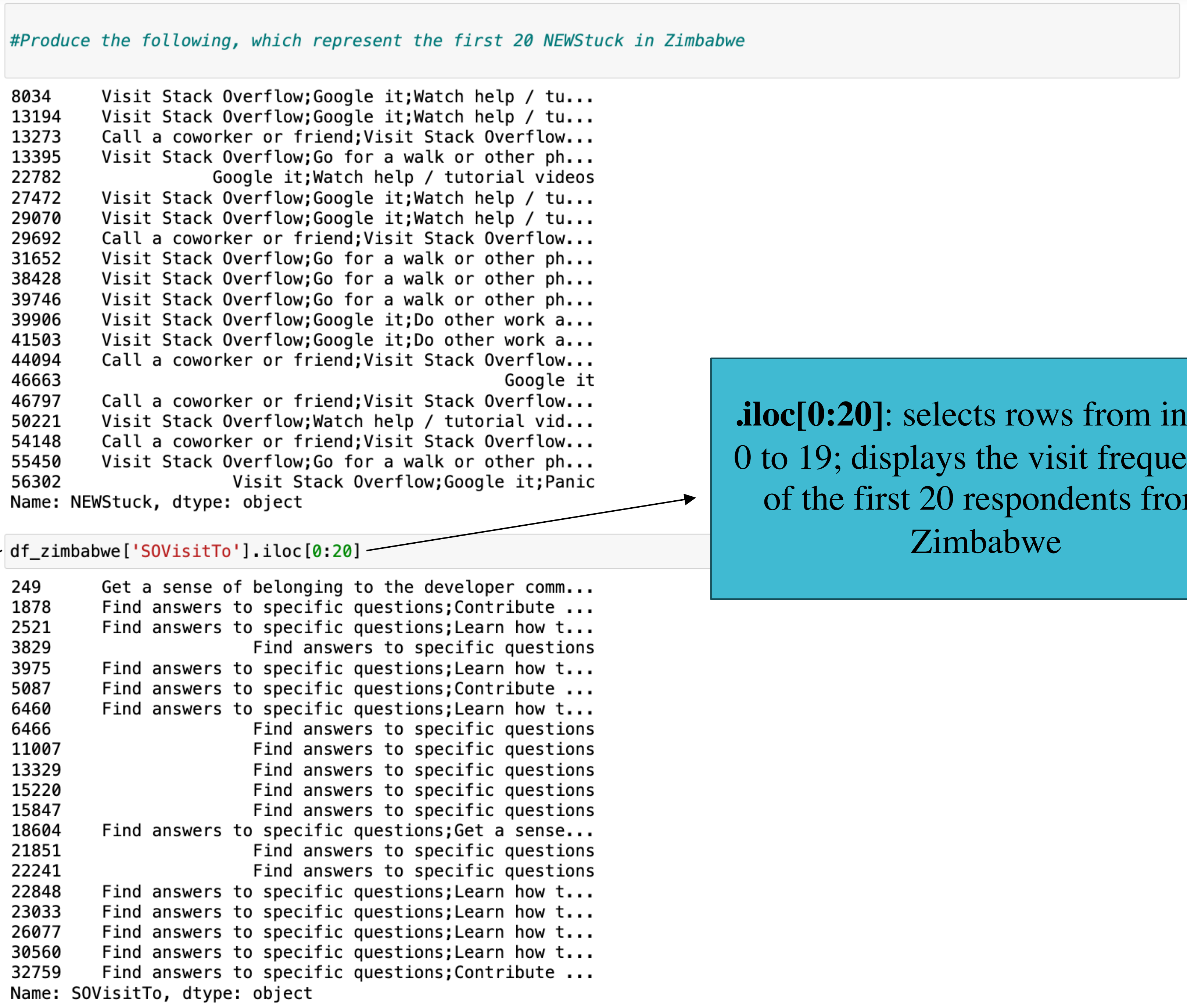
```python
df_zimbabwe = df[df['Country'] == 'Zimbabwe']
df_zimbabwe.head()
```

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	Student	EdLevel	UndergradMajor	...	WelcomeChange	SC
249	251	I am a student who is learning to code	Yes	Once a month or more often	OSS is, on average, of HIGHER quality than pro...	Not employed, but looking for work	Zimbabwe	Yes, full-time	Secondary school (e.g. American high school, G...	NaN	...	A lot more welcome now than last year	Tec
1878	1886	I am a developer by profession	Yes	Less than once a month but more than once per ...	OSS is, on average, of HIGHER quality than pro...	Employed full-time	Zimbabwe	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Computer science, computer engineering, or sof...	...	Just as welcome now as I felt last year	Tec
2521	2530	I am a developer by profession	Yes	Less than once a month but more than once per ...	OSS is, on average, of LOWER quality than prop...	Employed full-time	Zimbabwe	Yes, full-time	Bachelor's degree (BA, BS, B.Eng., etc.)	Information systems, information technology, o...	...	Just as welcome now as I felt last year	wi deve
3829	3845	I am a developer by profession	Yes	Never	The quality of OSS and closed source software ...	Employed full-time	Zimbabwe	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Information systems, information technology, o...	...	Just as welcome now as I felt last year	wi deve
3975	3991	I code primarily as a hobby	Yes	Less than once per year	OSS is, on average, of HIGHER quality than pro...	Independent contractor, freelancer, or self-em...	Zimbabwe	No	Bachelor's degree (BA, BS, B.Eng., etc.)	Information systems, information technology, o...	...	Not applicable - I did not use Stack Overflow ...	wi deve

5 rows × 85 columns

Filters the df to only include rows where the 'Country' column is 'Zimbabwe'

Displays the first 5 rows of the df



```
#Produce the following, which represent the first 20 NEWStuck in Zimbabwe
```

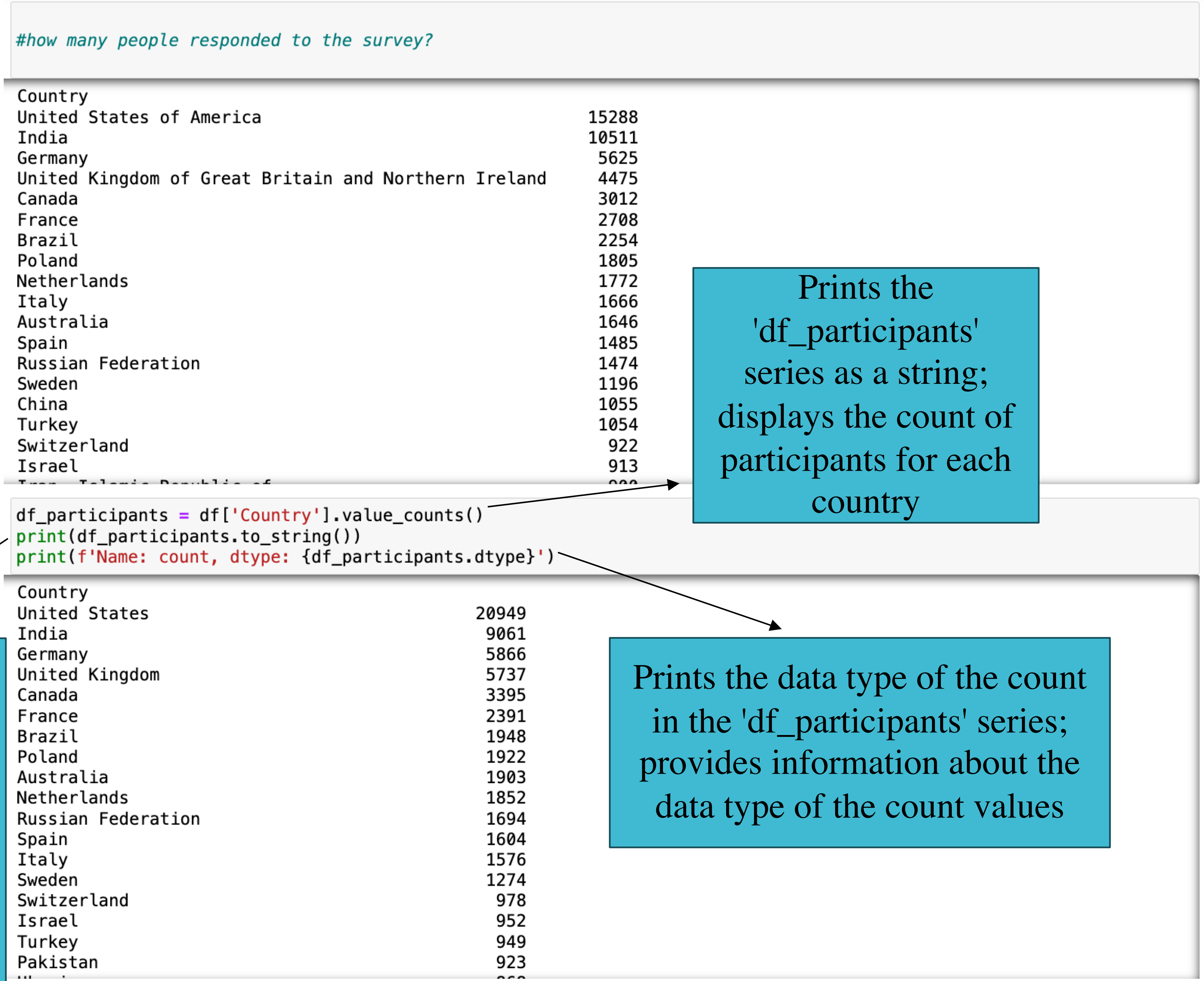
```
8034     Visit Stack Overflow;Google it;Watch help / tu...
13194    Visit Stack Overflow;Google it;Watch help / tu...
13273    Call a coworker or friend;Visit Stack Overflow...
13395    Visit Stack Overflow;Go for a walk or other ph...
22782              Google it;Watch help / tutorial videos
27472    Visit Stack Overflow;Google it;Watch help / tu...
29070    Visit Stack Overflow;Google it;Watch help / tu...
29692    Call a coworker or friend;Visit Stack Overflow...
31652    Visit Stack Overflow;Go for a walk or other ph...
38428    Visit Stack Overflow;Go for a walk or other ph...
39746    Visit Stack Overflow;Go for a walk or other ph...
39906    Visit Stack Overflow;Google it;Do other work a...
41503    Visit Stack Overflow;Google it;Do other work a...
44094    Call a coworker or friend;Visit Stack Overflow...
46663                                           Google it
46797    Call a coworker or friend;Visit Stack Overflow...
50221    Visit Stack Overflow;Watch help / tutorial vid...
54148    Call a coworker or friend;Visit Stack Overflow...
55450    Visit Stack Overflow;Go for a walk or other ph...
56302                Visit Stack Overflow;Google it;Panic
Name: NEWStuck, dtype: object
```

```
df_zimbabwe['SOVisitTo'].iloc[0:20]
```

```
249      Get a sense of belonging to the developer comm...
1878     Find answers to specific questions;Contribute ...
2521     Find answers to specific questions;Learn how t...
3829                   Find answers to specific questions
3975     Find answers to specific questions;Learn how t...
5087     Find answers to specific questions;Contribute ...
6460     Find answers to specific questions;Learn how t...
6466                   Find answers to specific questions
11007                  Find answers to specific questions
13329                  Find answers to specific questions
15220                  Find answers to specific questions
15847                  Find answers to specific questions
18604    Find answers to specific questions;Get a sense...
21851                  Find answers to specific questions
22241                  Find answers to specific questions
22848    Find answers to specific questions;Learn how t...
23033    Find answers to specific questions;Learn how t...
26077    Find answers to specific questions;Learn how t...
30560    Find answers to specific questions;Learn how t...
32759    Find answers to specific questions;Contribute ...
Name: SOVisitTo, dtype: object
```

.iloc[0:20]: selects rows from index 0 to 19; displays the visit frequency of the first 20 respondents from Zimbabwe

Selects the 'SOVisitTo' column in the df, showing how respondents from Zimbabwe visit Stack Overflow



```
#how many people responded to the survey?
```

```
Country
United States of America                                15288
India                                                   10511
Germany                                                  5625
United Kingdom of Great Britain and Northern Ireland     4475
Canada                                                   3012
France                                                   2708
Brazil                                                   2254
Poland                                                   1805
Netherlands                                              1772
Italy                                                    1666
Australia                                                1646
Spain                                                    1485
Russian Federation                                       1474
Sweden                                                   1196
China                                                    1055
Turkey                                                   1054
Switzerland                                               922
Israel                                                    913
```

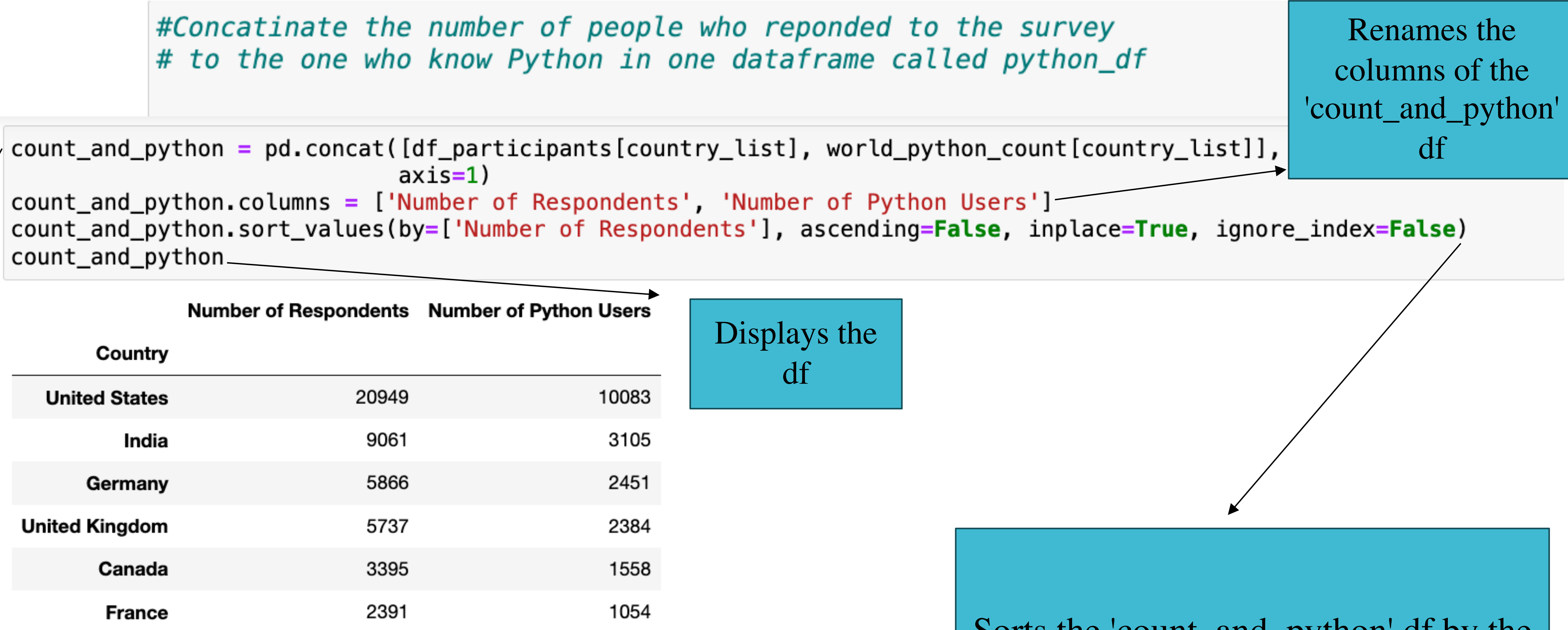
```
df_participants = df['Country'].value_counts()
print(df_participants.to_string())
print(f'Name: count, dtype: {df_participants.dtype}')
```

```
Country
United States                                20949
India                                         9061
Germany                                       5866
United Kingdom                                5737
Canada                                        3395
France                                        2391
Brazil                                        1948
Poland                                        1922
Australia                                     1903
Netherlands                                   1852
Russian Federation                            1694
Spain                                         1604
Italy                                         1576
Sweden                                        1274
Switzerland                                    978
Israel                                         952
Turkey                                         949
Pakistan                                       923
```

Prints the 'df_participants' series as a string; displays the count of participants for each country

Calculates the count of participants from each unique country in the df; stores the result in the 'df_participants' series

Prints the data type of the count in the 'df_participants' series; provides information about the data type of the count values



```python
#Concatinate the number of people who reponded to the survey
# to the one who know Python in one dataframe called python_df
```

```python
count_and_python = pd.concat([df_participants[country_list], world_python_count[country_list]],
                             axis=1)
count_and_python.columns = ['Number of Respondents', 'Number of Python Users']
count_and_python.sort_values(by=['Number of Respondents'], ascending=False, inplace=True, ignore_index=False)
count_and_python
```

	Number of Respondents	Number of Python Users
Country		
United States	20949	10083
India	9061	3105
Germany	5866	2451
United Kingdom	5737	2384
Canada	3395	1558
France	2391	1054

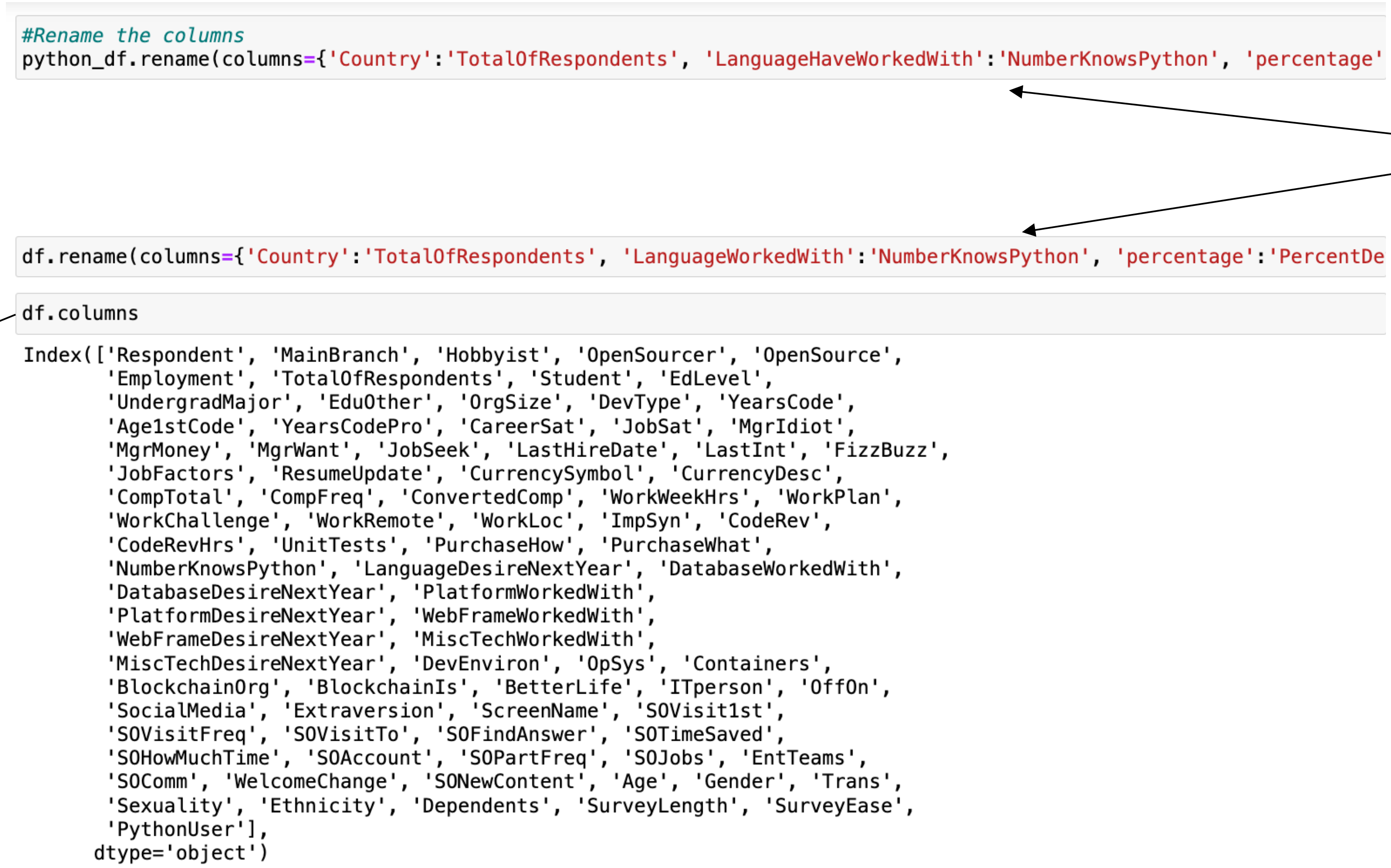
Renames the columns of the 'count_and_python' df

Displays the df

Creates a df 'count_and _python' with two columns: 'Number of Respondent s' and 'Number of Python Users', for each country in the list

Sorts the 'count_and_python' df by the 'Number of Respondents' column in descending order; ''ignore_index=False' ensures that the original index values are retained after sorting

Changes the names of specific columns in the 'python_df' df



```
#Rename the columns
python_df.rename(columns={'Country':'TotalOfRespondents', 'LanguageHaveWorkedWith':'NumberKnowsPython', 'percentage'
```

```
df.rename(columns={'Country':'TotalOfRespondents', 'LanguageWorkedWith':'NumberKnowsPython', 'percentage':'PercentDe
```

```
df.columns
```

```
Index(['Respondent', 'MainBranch', 'Hobbyist', 'OpenSourcer', 'OpenSource',
       'Employment', 'TotalOfRespondents', 'Student', 'EdLevel',
       'UndergradMajor', 'EduOther', 'OrgSize', 'DevType', 'YearsCode',
       'Age1stCode', 'YearsCodePro', 'CareerSat', 'JobSat', 'MgrIdiot',
       'MgrMoney', 'MgrWant', 'JobSeek', 'LastHireDate', 'LastInt', 'FizzBuzz',
       'JobFactors', 'ResumeUpdate', 'CurrencySymbol', 'CurrencyDesc',
       'CompTotal', 'CompFreq', 'ConvertedComp', 'WorkWeekHrs', 'WorkPlan',
       'WorkChallenge', 'WorkRemote', 'WorkLoc', 'ImpSyn', 'CodeRev',
       'CodeRevHrs', 'UnitTests', 'PurchaseHow', 'PurchaseWhat',
       'NumberKnowsPython', 'LanguageDesireNextYear', 'DatabaseWorkedWith',
       'DatabaseDesireNextYear', 'PlatformWorkedWith',
       'PlatformDesireNextYear', 'WebFrameWorkedWith',
       'WebFrameDesireNextYear', 'MiscTechWorkedWith',
       'MiscTechDesireNextYear', 'DevEnviron', 'OpSys', 'Containers',
       'BlockchainOrg', 'BlockchainIs', 'BetterLife', 'ITperson', 'OffOn',
       'SocialMedia', 'Extraversion', 'ScreenName', 'SOVisit1st',
       'SOVisitFreq', 'SOVisitTo', 'SOFindAnswer', 'SOTimeSaved',
       'SOHowMuchTime', 'SOAccount', 'SOPartFreq', 'SOJobs', 'EntTeams',
       'SOComm', 'WelcomeChange', 'SONewContent', 'Age', 'Gender', 'Trans',
       'Sexuality', 'Ethnicity', 'Dependents', 'SurveyLength', 'SurveyEase',
       'PythonUser'],
      dtype='object')
```

Displays the column names of the df

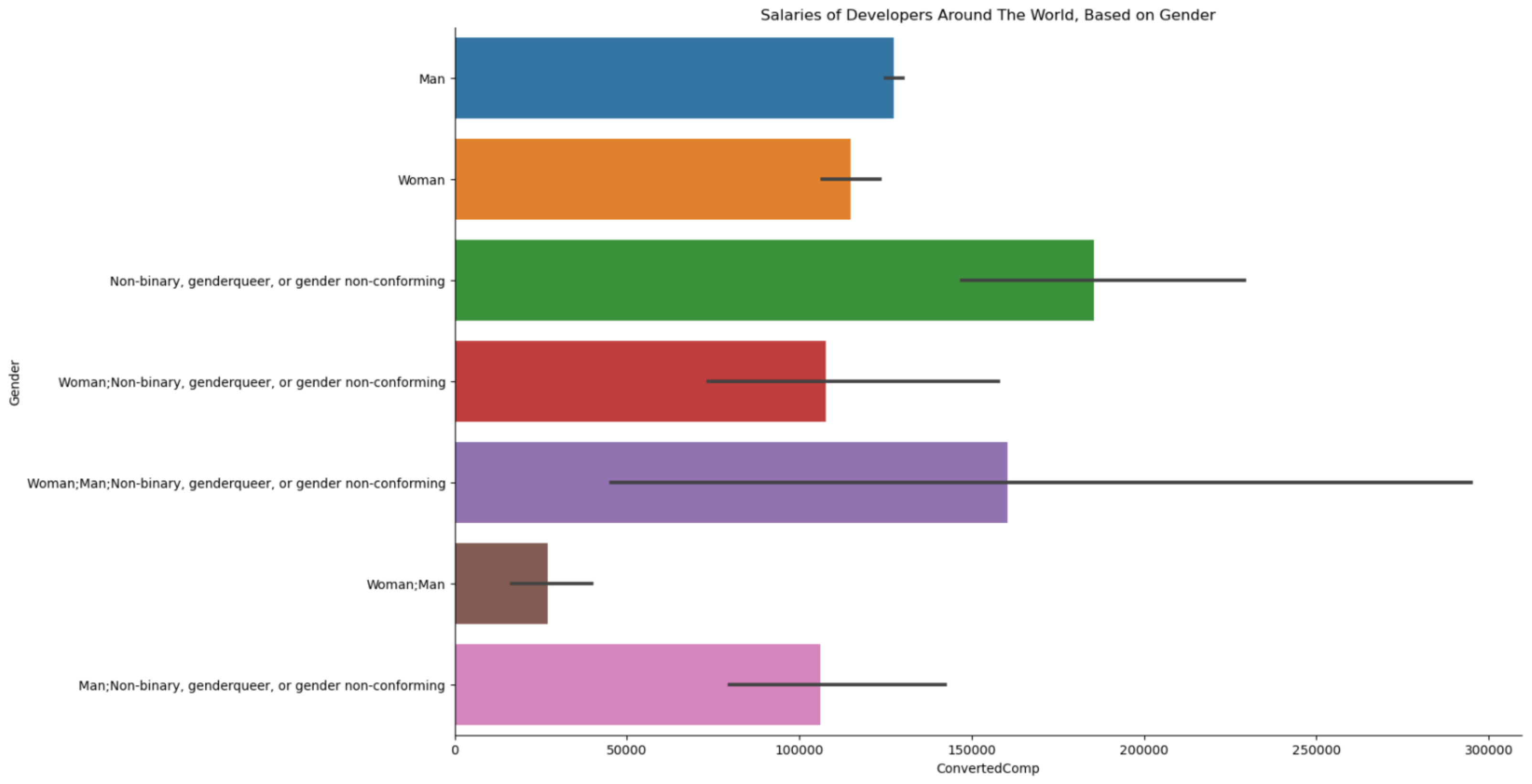
Results in df containing data only for respondents from Greece

```
#Surprise me with some plot off this later dataframe
```

```
df.rename(columns={'TotalOfRespondents':'Country'}, inplace=True)

greece_df = df[df['Country']=='Greece']

plt.rcParams["figure.figsize"] = (15,10)
sns.barplot(x = 'ConvertedComp', y = 'Gender', data = df).set(title='Salaries of Developers Around The World, Based
sns.despine()
plt.show()
plt.savefig('saving-a-seaborn-plot-as-pdf-file.png')
```

Renames the column labeled 'TotalOfRespondents' to 'Country'; <u>'inplace=True'</u> ensures that the change is done directly on the original df, without creating a copy

- **Plt.rcParams...**: sets the size of the plot
- **Sns.barplot...**: creates a barplot with 'ConvertedComp' as the x-axis variable, 'Gender' as the y-axis variable, and the title of the plot
- **Sns.despine()**: removes top and right spines from the plot
- **Plt.show()**: displays the plot
- **Plt.savefig...**: saves the plot as a PNG file



```
<Figure size 1500x1000 with 0 Axes>
```