1 newpage

2 Data Science: Bridging Principles and Practice

2.1 Part 3: DataFrames [SOLUTIONS]

2.2 3. DataFrames

2.3 3b. Explore the data: attributes and methods

2.3.1 Attributes

EXERCISE: One extremely useful attribute is `shape`, which returns the number of rows and columns in the DataFrame, separated by commas. In the next cell, get the `shape` attribute from the `ads` DataFrame.

```
In [2]: # get the shape of the DataFrame
        data_shape = ads.shape
        data_shape

Out[2]: (588101, 6)
```

2.3.2 Methods

EXERCISE: the `describe` method is incredibly useful for learning about your data. `describe` returns summary statistics about the numerical data in your DataFrame: things like the count of non-empty items in each column, the average, the minimum, and the maximum.

In the next cell, call the `describe` method on your DataFrame. This call will look very similar to the example where the `head` method was called; only the name of the method changes.

```
In [3]: # use dot notation to call "describe" on the ads table in place of the ellipses
        ads_description = ads.describe()
        ads_description

Out[3]:             user id      total ads   most ads hour
        count  5.881010e+05  588101.000000   588101.000000
        mean   1.310692e+06      24.820876       14.469061
        ... Omitting 2 lines ...
        50%    1.313725e+06      13.000000       14.000000
        75%    1.484088e+06      27.000000       18.000000
        max    1.654483e+06    2065.000000       23.000000
```

2.4 3c. Selecting columns

EXERCISE: Use square brackets to index the "converted" column.

```
In [4]: # index the "converted" column
        converted_col = ads["converted"]
        converted_col

Out[4]: 0         False
        1         False
        2         False
        ... Omitting 55 lines ...
        588099    False
        588100    False
        Name: converted, Length: 588101, dtype: bool
```

2.5 3d. Filtering rows

EXERCISE: Oftentimes, we want to calculate statistics separately for the control and experimental groups. Create two tables, one containing only rows where the user was in the `"ad"` group (the experimental group) and one with only rows where the user was in the `"psa"` group (the control group).

Hint: We've given you the code to create this for the experiment group. Fill in the appropriate value to select users who did NOT see ads.

```
In [5]: # users in the experiment group
        experiment = ads[ads["test group"] == "ad"]
        experiment.head()
```

```
Out[5]:    user id test group  converted  total ads most ads day  most ads hour
        0  1069124         ad      False        130       Monday             20
        1  1119715         ad      False         93      Tuesday             22
        2  1144181         ad      False         21      Tuesday             18
        3  1435133         ad      False        355      Tuesday             10
        4  1015700         ad      False        276       Friday             14
```

```
In [6]: # users in the control group
        # fill in the ellipses with the correct text to select users who saw psas
        control = ads[ads["test group"] == "psa"]
        control.head()
```

```
Out[6]:      user id test group  converted  total ads most ads day  most ads hour
        18    900681        psa      False        248     Saturday             19
        38    905704        psa      False         27     Thursday              8
        68    904595        psa      False         13      Tuesday             19
        140   901904        psa      False         32    Wednesday             19
        157   902234        psa      False        105      Tuesday             19
```