Seaborn: Ecommerce Purchases

JaDay Garel-Acosta, Pratyush Ghimire, Aniyah Gomez, Adebola Babatunde-Lawal, Brian Paul

From a (fake) set of data from amazon, we were able to determine the answers to the assigned questions.

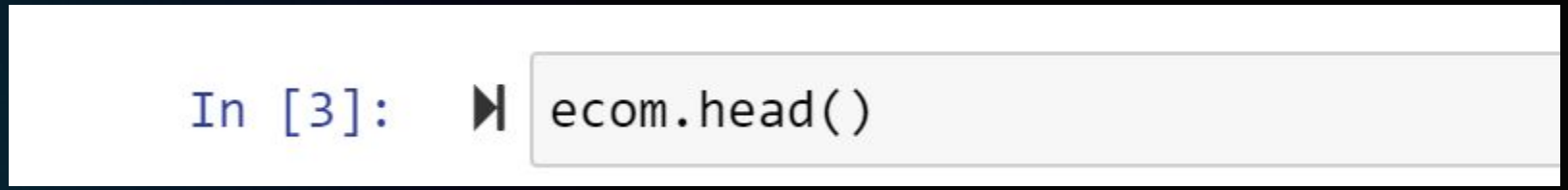
Data Frames

```
In [1]: import numpy as np
        import pandas as pd
```

Reading the CSV file

```
In [2]: ecom = pd.read_csv('Ecommerce Purchases')
```

Checking the head of the data frame



Output of the data frame head

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	CC Provider
0	16629 Pace Camp Apt. 448\nAlexisborough, NE 77...	46 in	PM	Opera/9.56.(X11; Linux x86_64; sl-SI) Presto/2...	Martinez-Herman	6011929061123406	02/20	900	JCB 16 digit
1	9374 Jasmine Spurs Suite 508\nSouth John, TN 8...	28 rn	PM	Opera/8.93.(Windows 98; Win 9x 4.90; en-US) Pr...	Fletcher, Richards and Whitaker	3337758169645356	11/18	561	Mastercard
2	Unit 0065 Box 5052\nDPO AP 27450	94 vE	PM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Simpson, Williams and Pham	675957666125	08/19	699	JCB 16 digit
3	7780 Julia Fords\nNew Stacy, WA 45798	36 vm	PM	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0 ...	Williams, Marshall and Buchanan	6011578504430710	02/24	384	Discover
4	23012 Munoz Drive Suite 337\nNew Cynthia, TX 5...	20 IE	AM	Opera/9.58.(X11; Linux x86_64; it-IT) Presto/2...	Brown, Watson and Andrews	6011456623207998	10/25	678	Diners Club / Carte Blanche

Email	Job	IP Address	Language	Purchase Price
pdunlap@yahoo.com	Scientist, product/process development	149.146.147.205	el	98.14
anthony41@reed.com	Drilling engineer	15.160.41.51	fr	70.73
amymiller@morales-harrison.com	Customer service manager	132.207.160.22	de	0.95
brent16@olson-robinson.info	Drilling engineer	30.250.74.19	es	78.04
christopherwright@gmail.com	Fine artist	24.140.33.94	es	77.82

Adding rows & columns

In [4]:

```
row_col = ecom.shape
print("Number of rows=",row_col[0])
print("Number of Column=",row_col[1])
```

```
Number of rows= 10000
Number of Column= 14
```

Average purchase price

In [5]:

```
pur_avg = ecom['Purchase Price'].mean()
print("Average Purchase Price is",pur_avg)
```

```
Average Purchase Price is 50.34730200000025
```

Highest purchase price


```
pur_max = ecom['Purchase Price'].max()
print("Highest purchase prices=",pur_max)
```

```
Highest purchase prices= 99.99
```

Lowest purchase price

```
pur_min = ecom['Purchase Price'].min()
print("Lowest purchase prices=",pur_min)
```

```
Lowest purchase prices= 0.0
```

Language of choice is English

** How many people have English 'en' as their Language of choice on the website? **

```
]:  #Finding the number of people who speak any language using value_cou
    lan = ecom['Language'].value_counts()
    #using get() function to find the number of people who speak english
    en_count = lan.get(key = 'en')
    print("People who speak English Language:",en_count)
```

```
People who speak English Language: 1098
```

Lawyers

** How many people have the job title of "Lawyer" ? **

```
]:  #Getting number of people with different job titles using value_counts() method and storing the result to a variable
    job = ecom['Job'].value_counts()
    lawyer_count = job.get(key = 'Lawyer')
    #using get() function to find the number of people who are 'Lawyers'
    print("Number of people with 'Lawyer' as their job title:",lawyer_count)
```

```
Number of people with 'Lawyer' as their job title: 30
```

Change in purchase according to time

** How many people made the purchase during the AM and how many people made the purchase during PM ? **

*(Hint: Check out value_counts()) *

:
```
#Getting the number of people who purchase during AM or PM using value_counts() method and storing the result to a variable
am_pm = ecom['AM or PM'].value_counts()
#using get() function to find the number of people who made a purchase during 'AM'
am = am_pm.get(key='AM')
#using get() function to find the number of people who made a purchase during 'PM'
pm = am_pm.get(key='PM')
print("Purchase during AM:",am,"\nPurchase during PM:",pm)
```

```
Purchase during AM: 4932
Purchase during PM: 5068
```

Common job titles

** What are the 5 most common Job Titles? **

```
#Printing the top 5 job titles
top_job = dict(ecom['Job'].value_counts()[0:5])
print(list(top_job.keys()))
```

```
['Interior and spatial designer', 'Lawyer', 'Social researcher', 'Purchasing manager', 'Designer, jewellery']
```

** Someone made a purchase that came from Lot: "90 WT" , what was the Purchase Price for this transaction? **

```
:  #Getting the Purchase price for the transaction from lot='90 WT'
   ecom[ecom['Lot']=="90 WT"]['Purchase Price']
```

```
[12]: 513    75.1
      Name: Purchase Price, dtype: float64
```

** What is the email of the person with the following Credit Card Number: 4926535242672853 **

```
#Getting the name of the person who had Credit Card Number: 4926535242672853, and printing their email
ecom[ecom["Credit Card"]==4926535242672853]['Email']
```

```
1234    bondellen@williams-garza.com
Name: Email, dtype: object
```

American Express

* *How many people have American Express as their Credit Card Provider *and* made a purchase above $95 ?**

```
:  #Getting the number of people who had American Express card and purchaseed more than $95
   ecom[(ecom["CC Provider"]=="American Express") & (ecom["Purchase Price"]>95)].count()
```

```
[14]: Address             39
      Lot                 39
      AM or PM            39
      Browser Info        39
      Company             39
      Credit Card         39
      CC Exp Date         39
      CC Security Code    39
      CC Provider         39
      Email               39
      Job                 39
      IP Address          39
      Language            39
      Purchase Price      39
      dtype: int64
```

Expiration in 2025

** Hard: How many people have a credit card that expires in 2025? **

```
#using a lamda function to get the 'CC Exp Date' which ends with '25' and getting a sum value
exp_2025 = sum(ecom["CC Exp Date"].apply(lambda x: x[3:])=='25')
print("Card that expires in 2025:",exp_2025)
```

```
Card that expires in 2025: 1033
```

Popular email providers

** Hard: What are the top 5 most popular email providers/hosts (e.g. gmail.com, yahoo.com, etc...) **

```
#Printing the top 5 most popular email provider using value counts method
top_email = ecom["Email"].str.split('@').str[1].value_counts()[0:5]
print(list(top_email.keys()))
```

```
['hotmail.com', 'yahoo.com', 'gmail.com', 'smith.com', 'williams.com']
```


THANK YOU