DATA1050

December 11, 2022

```
[116]: import pandas as pd
```

```
[117]: pd.set_option('max_colwidth', None)
       pd.set_option('display.max_rows', None)
       pd.set_option('display.max_columns', None)
       pd.set_option('display.width', None)
       pd.set_option('display.max_colwidth', -1)
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/4028272233.py:5:
FutureWarning: Passing a negative integer is deprecated in version 1.0 and will
not be supported in future version. Instead, use None to not limit the column
width.
  pd.set_option('display.max_colwidth', -1)
```

```
[118]: log = pd.read_csv("log2.csv", header=None) # load in the log2.csv file
       log.rename(columns = {0: "Sentiment",
                             1: "Publication_URL",
                             2: "product_URL",
                             3: "clickORnot",
                             4: "gender",
                             5: "age_group"}, inplace = True) # give each column a␣
         ↪column name
       log.head()
```

```
[118]:   Sentiment                 Publication_URL                     product_URL  \
       0  positive  https://www.foxnews.com/        https://lees.com/jeans
       1  neutral   https://www.mirror.co.uk/news/  https://coach.com/purses
       2  negative  https://www.nbcnews.com/        https://covergirl.co/lipsticks
       3  positive  https://www.examiner.com/       https://covergirl.co/makeup
       4  negative  https://www.nj.com              https://dell.com/computers

          clickORnot  gender   age_group
       0  0           female  juvenile
       1  0           male    young
       2  0           male    middle-age
       3  0           male    juvenile
       4  1           female  young
```

```
[119]: products = pd.read_csv("products.csv") # load in the products.csv file
       category = pd.read_csv("product_categories.csv") # load in the␣
        ↪product_categories.csv file
       category = category.rename(columns={"product": "product_type"}) # rename column␣
        ↪product to product_type
       for col in products.columns:
           products[col] = products[col].str.strip() # delete all the spaces for each␣
        ↪data entry in dataframe products
       for col in category.columns:
           category[col] = category[col].str.strip() # delete all the spaces for each␣
        ↪data entry in dataframe category
```

```
[120]: category.head()
```

```
[120]:      product_type                   category
       0   blender          small kitchen appliances
       1   pressure cooker  small kitchen appliances
       2   computer         consumer electronics
       3   coffee           packaged food
       4   vitamin          health
```

```
[121]: products.drop_duplicates(inplace=True) # drop duplicate row in dataframe␣
        ↪products
```

```
[122]: products = products.reset_index().iloc[:,[1,2,3]] # reset index for dataframe␣
        ↪products
       products.head()
```

```
[122]:                       product                       product_URL     product_type
       0  Vitamix blender             https://vitamix.com/blenders     blender
       1  Lenova laptop               https://lenova.com/laptops       computer
       2  InstantPot pressure cooker  https://InstantPot.com/cookers  pressure cooker
       3  NemoK blender               http://nemoK.co/blenders         blender
       4  Hamilton Beach blender      https://HamiltonBeach/blenders  blender
```

```
[123]: products_url = products.iloc[:,1]
       len(products_url.unique()) # there are total 50 different URLs
```

```
[123]: 50
```

1 Task 1

Some of the Product_URLs in the log file might have been corrupted. Write a Python (or PySpark) procedure to determine which Product_URLs are corrupted. Let us assume that if a Product_url in the log file doesn't occur in the products table, it is regarded as corrupted. Using this procedure identify and list the corrupted URLs. (10)

2 Answer 1

The way I detect corrupted URLs is to loop through all the URLs in the log dataframe and match it with the unique product URLs in the products dataframe. Below list contains all the indices for corrupted URLs in the dataframe log.

[124]:

```
yes_indices = []
no_indices = []
for i, url in enumerate(log.iloc[:,2]):
    if url in set(products_url): # if the URL in log is also in products, then␣
 ↪this URL is not corrupted
        yes_indices.append(i)
    else: # otherwise, it is a corrupted URL
        no_indices.append(i)
# print(yes_indices)
print(no_indices)
len(no_indices) # there are total 216 corrupted URLs
```

```
[83, 109, 123, 171, 203, 212, 245, 273, 339, 434, 436, 497, 562, 618, 738, 779,
790, 798, 830, 857, 870, 913, 945, 1025, 1057, 1113, 1138, 1239, 1249, 1338,
1405, 1452, 1609, 1695, 1761, 1767, 1801, 1861, 1900, 1910, 1913, 1995, 1996,
2014, 2057, 2138, 2143, 2161, 2173, 2242, 2252, 2269, 2292, 2294, 2371, 2427,
2433, 2529, 2629, 2669, 2678, 2713, 2769, 2939, 3014, 3020, 3060, 3067, 3076,
3139, 3234, 3248, 3320, 3321, 3353, 3385, 3433, 3444, 3500, 3521, 3537, 3557,
3567, 3577, 3615, 3849, 3863, 3876, 4044, 4088, 4163, 4274, 4570, 4694, 4710,
4735, 4740, 4757, 4864, 4929, 5020, 5092, 5116, 5236, 5274, 5308, 5329, 5372,
5416, 5458, 5482, 5516, 5578, 5596, 5616, 5664, 5713, 5741, 5756, 5782, 5838,
5871, 5875, 5876, 6012, 6034, 6107, 6181, 6272, 6291, 6338, 6365, 6411, 6413,
6465, 6467, 6483, 6484, 6541, 6615, 6620, 6713, 6784, 6787, 6858, 6861, 6911,
6952, 6957, 7013, 7045, 7046, 7047, 7119, 7134, 7191, 7204, 7207, 7231, 7260,
7263, 7287, 7388, 7420, 7456, 7487, 7490, 7532, 7669, 7688, 7728, 7891, 8052,
8188, 8255, 8281, 8420, 8453, 8456, 8517, 8552, 8594, 8735, 8749, 8871, 8881,
8934, 9039, 9051, 9090, 9103, 9125, 9131, 9158, 9159, 9210, 9230, 9231, 9276,
9316, 9359, 9360, 9395, 9411, 9514, 9537, 9585, 9591, 9594, 9632, 9680, 9729,
9773, 9845, 9881, 9951]
```

[124]:

```
216
```

3 Task 2

For each corrupted URL what will you do with it? Don't assume that for each corrupted URL the correct approach is to delete that log entry. What if the URL contained '.cam' instead of '.com' but otherwise corresponded with a URL in the 'products' table? In that case the proper approach would be to correct the URL. In other cases, the URL might be so corrupted that the best approach would be to delete that log entry (the entire row). Describe your approach to dealing with corrupted URLs. That is, describe your approach to determining that a URL is too corrupted to be rescued. It must describe a) a procedure for determining the degree to which the URL is corrupted, b) a threshold for determining in terms of this degree of corruption whether it can be corrected, and c) for those which can be corrected, identifying its corrected form. For extra credit implement this in a Python (or PySpark) program. (25 + 20 points for extra-credit)

4 Answer 2

The way I use to evaluate how corrupted the URL is is using a function called SequenceMatcher from difflib package. This function calculates a similarity score between two URLs (a corrupted one and a correct one) and if the score passes a threshold, then these two URLs are considered similar (the corrupted URL is not too corrupted). When the threshold is 0.95, there are three URLs don't pass the threshold and are considered too corrupted. The indices of them are 790, 5876, and 7263. Actually, we can see that those URLs are not too corrupted; they are only off by one letter, so I decide to decrease the threshold to 0.9.

After changing the threshold to 0.9, there are no super corrupted URLs; all of them have been corrected in the dataframe log.

[125]:
```
url_log = log.iloc[:,2]
unique_log = set(url_log)
unique_log = list(unique_log) # unique_log contains all the unique URLs in␣
 ↪dataframe log
```

[126]:
```
from difflib import SequenceMatcher

def similar(a, b):
    return SequenceMatcher(None, a, b).ratio() # we are using SequenceMatcher␣
 ↪to compare how similar two URLs is
```

[127]:
```
temp = []
for i in range(len(unique_log)):
    for l in range(len(products_url)):
        score = similar(str(unique_log[i]), str(products_url[l]))
        if score > 0.95: # if the score calculated between two URLs is above␣
 ↪threshold 0.95, then these two URLs are similar and can be corrected
            temp.append([unique_log[i],products_url[l]])
```

[128]:
```
temp.sort()
temp[0:5] # temp contains pairs of the corrupted URL and the correct URL
```

[128]:
```
[['http://maybellije.com/lipstick', 'http://maybelline.com/lipstick'],
 ['http://maybelline.com/lipstick', 'http://maybelline.com/lipstick'],
 ['http://maybelline.com/lipstuck', 'http://maybelline.com/lipstick'],
 ['http://maybelline.com/xipstick', 'http://maybelline.com/lipstick'],
 ['http://nejoK.co/blenders', 'http://nemoK.co/blenders']]
```

[129]:
```
for i in range(len(temp)):
    for l in range(len(log)):
        if temp[i][0] == log.iloc[l,2]:
            log.iloc[l,2] = temp[i][1] # correcting the corrupted URLs in the␣
 ↪dataframe log
```

[130]:
```
yes_indices = []
no_indices = []
for i, url in enumerate(log.iloc[:,2]):
    if url in set(products_url):
        yes_indices.append(i)
    else:
        no_indices.append(i)
print(no_indices) # there are only three corrupted URLs after we detecting and␣
 ↪correcting others
```

```
[790, 5876, 7263]
```

[131]:
```
print(log.iloc[790,2])
print(log.iloc[5876,2])
print(log.iloc[7263,2])
```

```
https://besla.com
https://tesla.rom
https://lg.comxtvs
```

[132]:
```
temp = []
for i in range(len(unique_log)):
    for l in range(len(products_url)):
        score = similar(str(unique_log[i]), str(products_url[l]))
        if score > 0.90: # if the score calculated between two URLs is above␣
 ↪threshold 0.9, then these two URLs are similar and can be corrected
            temp.append([unique_log[i],products_url[l]])
```

[133]:
```
for i in range(len(temp)):
    for l in range(len(log)):
        if temp[i][0] == log.iloc[l,2]:
            log.iloc[l,2] = temp[i][1] # correcting the corrupted URLs in the␣
 ↪dataframe log
```

[134]:
```
yes_indices = []
no_indices = []
for i, url in enumerate(log.iloc[:,2]):
    if url in set(products_url):
        yes_indices.append(i)
    else:
        no_indices.append(i)
print(no_indices) # there are no corrupted URLs anymore after we detecting and␣
 ↪correcting them
```

```
[]
```

5 Task 3

For each product, compute all the Publication_URLs containing an ad for that product. (Don't just give the results. Show all the work by which you got those results. This applies to all the questions below.) (10)

6 Answer 3

In order to find all the unique Publication_URLs containing an ad for that product, I first merge dataframes log and products to create a new dataframe called new. Then, I groupby dataframe new ny column "Publication_URL" and display column "product". For each product, I extract only unique publication URLs and store them into a list. Then, I convert all the entries of that list from list to string and create a dataframe callend url_product containing two columns: product and URL_list_unique.

[135]:

```
new = pd.merge(log, products, on = "product_URL")
new.head() # I created a new dataframe that merges dataframes log and products
```

[135]:

```
  Sentiment                  Publication_URL            product_URL  \
0  positive  https://www.foxnews.com/        https://lees.com/jeans
1  negative  https://www.nytimes.com/        https://lees.com/jeans
2  positive  https://www.cnn.com/            https://lees.com/jeans
3  negative  https://www.chicagotribune.com/ https://lees.com/jeans
4  positive  https://www.salon.com/          https://lees.com/jeans

   clickORnot  gender   age_group    product product_type
0  0           female  juvenile    Lee jeans  jeans
1  0           female  middle-age  Lee jeans  jeans
2  1           female  middle-age  Lee jeans  jeans
3  0           male    young       Lee jeans  jeans
4  1           female  middle-age  Lee jeans  jeans
```

[136]:

```
number_product = new.groupby("product")["Publication_URL"] # I groupby the␣
 ↪dataframe new by column "Publication_URL" and display column "product"
```

[137]:

```
temp_list = [] # this list contains all the unique Publication_URLs containing␣
 ↪an ad for one product
for key, item in number_product:
    # print(key); each key represents a product
    temp_list.append(number_product.get_group(key).unique())
```

[138]:

```
for i in range(len(temp_list)):
    temp_list[i] = temp_list[i].tolist() # we convert the result to lists
```

[139]:

```
# create a new dataframe that stores two columns: product and all the␣
 ↪Publication_URLs containing an ad for that product (URL_list_unique)
number_product = new.groupby("product")["Publication_URL"].count()
```

```
url_product = number_product.reset_index()
url_product = url_product.rename(columns={"Publication_URL": "URL_count"})
url_product["URL_list_unique"] = temp_list
```

[140]:

```
# convert all the entries of "URL_list_unique" from list to string
# create a new dataframe called url_product that stores information of product␣
 ↪and unique publication URLs
for i in range(len(url_product)):
    url_product.iloc[i,2] = ', '.join(url_product.iloc[i,2])
url_product = url_product.iloc[:,[0,2]]
url_product
```

[140]:

```
                       product  \
0   Apple computer
1   Apple iPad
2   Apple laptop
3   BasilBasel perfume
4   Broyhill recliner
5   Centrum MultiVitamins
6   Clinique moisturizer
7   Coach purse
8   Cougar jeans
9   Covergirl makeup
10  Dell computer
11  Dell laptop
12  Docker pants
13  Ford  sedan
14  Gillette shaver
15  Giorgio perfume
16  Givenchy perfume
17  Guess perfume
18  Haier refrigerator
19  Hamilton Beach blender
20  Ikea sofa
21  InstantPot pressure cooker
22  Jaguar perfume
23  Kaai handbags
24  LG TV
25  LG dryer
26  LG washer
27  Lavazza Coffee
28  Lee jeans
29  Lenova laptop
30  Maybelline lipstick
31  Maytag dryer
32  Maytag refrigerator
33  Maytag washer
```

```
34  NemoK blender
35  NordicTrack elliptical
36  NordicTrack rower
37  NordicTrack treadmill
38  Remington shaver
39  Samsung TV
40  Samsung dryer
41  Samsung washer
42  Sony TV
43  Soundwave speakers
44  Starbucks Coffee
45  Tesla
46  Vitamix blender
47  bose speakers
48  covergirl lipstick

                                                   URL_list_unique
0   https://www.nydailynews.com/, https://www.mirror.co.uk/news/,
https://www.cbsnews.com/, https://www.engadget.com/, https://www.usatoday.com/,
https://www.cnet.com/, https://nypost.com/, https://www.nj.com,
https://www.dallasnews.com/, https://abcnews.go.com/,
https://www.bostonglobe.com/, https://www.boston.com, https://www.cnn.com/,
https://www.upworthy.com/
1   https://mashable.com/, https://www.boston.com, https://www.nj.com,
https://www.sfgate.com/, https://www.mirror.co.uk/news/,
https://www.businessinsider.com/, https://www.cnn.com/,
https://www.vice.com/en_us, https://www.npr.org/, https://www.slate.com/,
https://www.chicagotribune.com/, https://nypost.com/, https://www.latimes.com/,
https://www.theguardian.com/us, https://www.examiner.com/,
https://www.telegraph.co.uk/, https://www.al.com/
2   https://www.vox.com/, https://www.bbc.com/, https://abcnews.go.com/,
https://www.telegraph.co.uk/, https://nypost.com/, https://www.buzzfeed.com/,
https://www.usatoday.com/, https://www.nytimes.com/,
https://www.thedailybeast.com/, https://www.businessinsider.com/
3   https://www.salon.com/, https://www.independent.co.uk/,
https://www.chicagotribune.com/, https://www.buzzfeed.com/,
https://techcrunch.com/, https://www.examiner.com/, https://www.engadget.com/,
https://www.nydailynews.com/, https://www.businessinsider.com/,
https://www.msn.com/en-us/news, https://www.vice.com/en_us,
https://www.nytimes.com/, https://www.cnn.com/, https://www.washingtonpost.com/,
https://www.boston.com
4   https://www.theatlantic.com/, https://www.upworthy.com/,
https://www.buzzfeed.com/, https://www.cnn.com/,
https://www.huffingtonpost.com/, https://time.com/, https://www.npr.org/,
https://techcrunch.com/, https://www.vice.com/en_us, https://www.vox.com/,
https://www.nydailynews.com/, https://www.usatoday.com/, https://www.al.com/,
https://www.nbcnews.com/, https://www.boston.com, https://www.salon.com/
```

5 https://mashable.com/, https://time.com/, https://www.nytimes.com/, https://www.independent.co.uk/, https://www.vox.com/, https://www.latimes.com/, https://www.washingtonpost.com/, https://www.al.com/, https://www.mirror.co.uk/news/, https://www.cnn.com/, https://www.upworthy.com/, https://www.engadget.com/, https://www.boston.com, https://techcrunch.com/, https://www.thedailybeast.com/, https://www.dallasnews.com/, https://www.usatoday.com/, https://www.nydailynews.com/, https://www.buzzfeed.com/

6 https://www.chicagotribune.com/, https://www.latimes.com/, https://www.huffingtonpost.com/, https://www.salon.com/, https://www.theguardian.com/us, https://www.bostonglobe.com/, https://www.examiner.com/, https://www.bbc.com/, https://www.usnews.com/, https://www.msn.com/en-us/news, https://www.nbcnews.com/, https://techcrunch.com/, https://www.nytimes.com/, https://www.cnn.com/, https://nypost.com/, https://www.washingtonpost.com/

7 https://www.mirror.co.uk/news/, https://www.foxnews.com/, https://www.engadget.com/, https://abcnews.go.com/, https://www.salon.com/, https://www.chicagotribune.com/, https://www.upworthy.com/, https://www.washingtonpost.com/, https://www.cnn.com/, https://techcrunch.com/, https://www.nbcnews.com/, https://www.businessinsider.com/, https://www.al.com/, https://www.vox.com/, https://www.nytimes.com/, https://www.nj.com, https://www.boston.com, https://www.nydailynews.com/

8 https://www.sfgate.com/, https://www.theguardian.com/us, https://www.slate.com/, https://mashable.com/, https://time.com/, https://www.buzzfeed.com/, https://www.cbsnews.com/, https://techcrunch.com/, https://www.boston.com, https://www.dallasnews.com/, https://www.nytimes.com/, https://www.al.com/, https://www.msn.com/en-us/news, https://www.salon.com/, https://www.bostonglobe.com/, https://www.nydailynews.com/, https://www.telegraph.co.uk/

9 https://www.examiner.com/, https://www.vox.com/, https://www.bostonglobe.com/, https://www.cnn.com/, https://www.nj.com, https://time.com/, https://www.theatlantic.com/, https://www.usatoday.com/, https://www.buzzfeed.com/, https://www.engadget.com/, https://www.nydailynews.com/, https://www.bbc.com/, https://abcnews.go.com/, https://www.dallasnews.com/, https://nypost.com/, https://www.upworthy.com/

10 https://www.nj.com, https://www.vox.com/, https://www.engadget.com/, https://www.latimes.com/, https://www.cnn.com/, https://www.cbsnews.com/, https://www.dallasnews.com/, https://www.usnews.com/, https://www.boston.com, https://abcnews.go.com/, https://www.salon.com/, https://www.thedailybeast.com/, https://www.upworthy.com/, https://www.al.com/, https://nypost.com/, https://www.theatlantic.com/

11 https://www.msn.com/en-us/news, https://www.businessinsider.com/, https://www.examiner.com/, https://www.huffingtonpost.com/, https://www.nydailynews.com/, https://www.usnews.com/, https://techcrunch.com/, https://www.buzzfeed.com/, https://www.bostonglobe.com/, https://www.dailymail.co.uk/, https://www.dallasnews.com/, https://www.thedailybeast.com/, https://www.telegraph.co.uk/,

https://www.boston.com, https://www.nj.com, https://www.vice.com/en_us,
https://www.nbcnews.com/, https://www.chicagotribune.com/,
https://www.usatoday.com/, https://www.cnn.com/, https://www.upworthy.com/,
https://www.washingtonpost.com/

12 https://www.bostonglobe.com/, https://www.thedailybeast.com/,
https://www.chicagotribune.com/, https://www.nydailynews.com/,
https://www.businessinsider.com/, https://www.nbcnews.com/, https://www.al.com/,
https://www.usnews.com/, https://nypost.com/, https://www.washingtonpost.com/,
https://time.com/, https://www.upworthy.com/, https://www.msn.com/en-us/news

13 https://www.chicagotribune.com/, https://www.npr.org/,
https://www.engadget.com/, https://www.usnews.com/, https://www.boston.com,
https://techcrunch.com/, https://www.nj.com, https://www.upworthy.com/,
https://www.nytimes.com/, https://www.foxnews.com/, https://www.vox.com/,
https://www.cnet.com/, https://www.dallasnews.com/, https://www.salon.com/,
https://www.independent.co.uk/

14 https://www.bbc.com/, https://www.cnet.com/, https://www.vice.com/en_us,
https://www.huffingtonpost.com/, https://www.thedailybeast.com/,
https://www.nytimes.com/, https://www.vox.com/, https://www.cnn.com/,
https://www.theguardian.com/us, https://www.boston.com,
https://www.washingtonpost.com/

15 https://www.usnews.com/, https://www.latimes.com/,
https://www.telegraph.co.uk/, https://www.sfgate.com/, https://www.bbc.com/,
https://www.businessinsider.com/, https://www.engadget.com/,
https://www.independent.co.uk/, https://www.nj.com, https://www.buzzfeed.com/,
https://mashable.com/, https://www.theguardian.com/us,
https://www.examiner.com/, https://www.cnn.com/

16 https://www.buzzfeed.com/, https://www.usnews.com/,
https://www.dailymail.co.uk/, https://www.bostonglobe.com/, https://nypost.com/,
https://www.washingtonpost.com/, https://www.businessinsider.com/,
https://www.upworthy.com/, https://www.nydailynews.com/,
https://www.usatoday.com/, https://www.vice.com/en_us, https://www.slate.com/,
https://www.nytimes.com/, https://techcrunch.com/,
https://www.mirror.co.uk/news/, https://abcnews.go.com/,
https://www.examiner.com/

17 https://www.foxnews.com/, https://www.washingtonpost.com/,
https://www.vox.com/, https://www.telegraph.co.uk/, https://www.boston.com,
https://techcrunch.com/, https://time.com/, https://www.dailymail.co.uk/,
https://www.upworthy.com/, https://www.latimes.com/,
https://www.dallasnews.com/, https://www.bostonglobe.com/, https://www.cnn.com/

18 https://www.cbsnews.com/, https://www.foxnews.com/, https://www.bbc.com/,
https://abcnews.go.com/, https://www.businessinsider.com/,
https://www.usatoday.com/, https://nypost.com/, https://www.examiner.com/,
https://www.msn.com/en-us/news, https://www.buzzfeed.com/

19 https://nypost.com/, https://www.foxnews.com/, https://www.nbcnews.com/,
https://techcrunch.com/, https://www.telegraph.co.uk/,
https://www.theatlantic.com/, https://www.nydailynews.com/,
https://www.huffingtonpost.com/, https://www.nytimes.com/,

https://www.businessinsider.com/, https://www.upworthy.com/,
https://www.vox.com/, https://www.usnews.com/, https://time.com/,
https://www.boston.com, https://www.usatoday.com/, https://www.dallasnews.com/

20 https://www.cnet.com/, https://www.nbcnews.com/, https://www.buzzfeed.com/,
https://www.al.com/, https://www.slate.com/, https://www.businessinsider.com/,
https://www.nytimes.com/, https://www.nj.com, https://www.huffingtonpost.com/,
https://www.cbsnews.com/, https://nypost.com/

21 https://www.buzzfeed.com/, https://www.latimes.com/,
https://abcnews.go.com/, https://www.dallasnews.com/, https://www.nytimes.com/,
https://www.bbc.com/, https://mashable.com/, https://www.al.com/,
https://www.dailymail.co.uk/, https://www.upworthy.com/,
https://www.theguardian.com/us, https://www.examiner.com/, https://www.nj.com,
https://www.cnn.com/

22 https://www.vox.com/, https://www.nytimes.com/,
https://www.huffingtonpost.com/, https://www.salon.com/,
https://www.washingtonpost.com/, https://www.cnn.com/,
https://www.dallasnews.com/, https://www.independent.co.uk/,
https://www.upworthy.com/, https://techcrunch.com/, https://www.vice.com/en_us,
https://www.businessinsider.com/

23 https://www.cnn.com/, https://www.telegraph.co.uk/,
https://www.usatoday.com/, https://www.dallasnews.com/, https://www.nj.com,
https://www.nbcnews.com/, https://www.mirror.co.uk/news/, https://www.al.com/,
https://www.businessinsider.com/, https://www.boston.com,
https://www.buzzfeed.com/, https://www.slate.com/, https://abcnews.go.com/,
https://www.salon.com/, https://nypost.com/

24 https://www.nbcnews.com/, https://www.salon.com/,
https://www.vice.com/en_us, https://www.businessinsider.com/,
https://www.telegraph.co.uk/, https://www.nytimes.com/,
https://www.buzzfeed.com/, https://nypost.com/, https://www.foxnews.com/,
https://time.com/, https://www.al.com/, https://www.theguardian.com/us,
https://www.nydailynews.com/

25 https://www.nytimes.com/, https://nypost.com/, https://www.bbc.com/,
https://www.businessinsider.com/, https://www.msn.com/en-us/news,
https://www.chicagotribune.com/, https://www.examiner.com/,
https://techcrunch.com/, https://time.com/, https://www.buzzfeed.com/,
https://www.nydailynews.com/, https://www.usatoday.com/

26 https://www.thedailybeast.com/, https://www.vice.com/en_us,
https://www.bostonglobe.com/, https://techcrunch.com/, https://www.npr.org/,
https://www.mirror.co.uk/news/, https://www.nydailynews.com/,
https://www.independent.co.uk/, https://www.businessinsider.com/,
https://www.upworthy.com/, https://www.chicagotribune.com/,
https://www.washingtonpost.com/, https://www.cnn.com/, https://nypost.com/

27 https://www.boston.com, https://www.dallasnews.com/,
https://www.dailymail.co.uk/, https://www.examiner.com/, https://nypost.com/,
https://www.cnn.com/, https://www.thedailybeast.com/, https://www.nytimes.com/,
https://www.vice.com/en_us

28 https://www.foxnews.com/, https://www.nytimes.com/, https://www.cnn.com/,

https://www.chicagotribune.com/, https://www.salon.com/,
https://www.engadget.com/, https://www.nydailynews.com/, https://mashable.com/,
https://www.theguardian.com/us, https://www.nbcnews.com/,
https://www.washingtonpost.com/, https://www.usatoday.com/,
https://www.usnews.com/, https://www.theatlantic.com/, https://www.al.com/,
https://www.buzzfeed.com/, https://www.nj.com, https://www.latimes.com/,
https://www.msn.com/en-us/news, https://www.businessinsider.com/,
https://www.vox.com/, https://nypost.com/
29 https://www.nbcnews.com/, https://www.telegraph.co.uk/,
https://www.cnn.com/, https://www.engadget.com/, https://www.nj.com,
https://www.boston.com, https://www.nytimes.com/, https://www.al.com/,
https://abcnews.go.com/, https://www.theatlantic.com/, https://techcrunch.com/,
https://www.usnews.com/, https://www.washingtonpost.com/,
https://www.businessinsider.com/, https://www.usatoday.com/,
https://www.examiner.com/, https://www.buzzfeed.com/
30 https://www.slate.com/, https://www.theatlantic.com/, https://www.cnn.com/,
https://www.dallasnews.com/, https://www.telegraph.co.uk/,
https://www.independent.co.uk/, https://www.msn.com/en-us/news,
https://www.nydailynews.com/, https://www.huffingtonpost.com/,
https://www.nj.com, https://www.engadget.com/, https://www.businessinsider.com/,
https://nypost.com/, https://www.vice.com/en_us, https://www.buzzfeed.com/
31 https://www.chicagotribune.com/, https://www.vice.com/en_us,
https://www.salon.com/, https://www.nj.com, https://www.cnn.com/,
https://www.businessinsider.com/, https://www.slate.com/,
https://www.nydailynews.com/, https://www.upworthy.com/,
https://www.engadget.com/, https://time.com/, https://www.thedailybeast.com/,
https://www.vox.com/, https://www.boston.com
32 https://www.bostonglobe.com/, https://techcrunch.com/, https://time.com/,
https://www.washingtonpost.com/, https://www.nbcnews.com/, https://www.bbc.com/,
https://www.huffingtonpost.com/, https://www.usnews.com/, https://nypost.com/
33 https://www.latimes.com/, https://www.independent.co.uk/,
https://www.nytimes.com/, https://www.sfgate.com/,
https://www.chicagotribune.com/, https://www.bostonglobe.com/,
https://www.vox.com/, https://www.theguardian.com/us, https://www.examiner.com/,
https://www.mirror.co.uk/news/, https://techcrunch.com/,
https://www.upworthy.com/, https://www.npr.org/, https://www.msn.com/en-us/news,
https://www.thedailybeast.com/, https://www.dailymail.co.uk/,
https://www.boston.com, https://time.com/, https://www.nbcnews.com/,
https://www.nj.com, https://www.washingtonpost.com/
34 https://www.sfgate.com/, https://www.dailymail.co.uk/,
https://mashable.com/, https://www.npr.org/, https://www.cbsnews.com/,
https://www.usnews.com/, https://www.cnn.com/, https://www.boston.com,
https://www.examiner.com/, https://abcnews.go.com/, https://www.usatoday.com/,
https://nypost.com/, https://www.nj.com, https://www.nytimes.com/,
https://www.bostonglobe.com/, https://www.theguardian.com/us,
https://techcrunch.com/
35 https://www.mirror.co.uk/news/, https://www.bbc.com/,

https://www.theguardian.com/us, https://abcnews.go.com/,
https://www.nbcnews.com/, https://www.bostonglobe.com/, https://www.usnews.com/,
https://www.boston.com, https://www.nydailynews.com/, https://www.latimes.com/,
https://www.dailymail.co.uk/, https://www.dallasnews.com/, https://www.vox.com/,
https://www.salon.com/

36 https://www.nbcnews.com/, https://www.usatoday.com/, https://www.nj.com,
https://www.boston.com, https://nypost.com/, https://www.cbsnews.com/,
https://www.vox.com/, https://www.nydailynews.com/, https://www.bbc.com/,
https://mashable.com/, https://abcnews.go.com/, https://www.upworthy.com/,
https://www.bostonglobe.com/, https://www.cnn.com/, https://www.dallasnews.com/,
https://www.examiner.com/, https://techcrunch.com/

37 https://www.nytimes.com/, https://www.boston.com,
https://www.washingtonpost.com/, https://www.vice.com/en_us,
https://www.msn.com/en-us/news, https://www.independent.co.uk/,
https://www.upworthy.com/, https://www.nbcnews.com/,
https://www.bostonglobe.com/, https://www.businessinsider.com/,
https://mashable.com/, https://time.com/, https://www.usatoday.com/,
https://www.nydailynews.com/, https://www.latimes.com/, https://www.salon.com/,
https://www.usnews.com/, https://techcrunch.com/, https://www.buzzfeed.com/

38 https://www.chicagotribune.com/, https://www.businessinsider.com/,
https://www.theguardian.com/us, https://www.vox.com/,
https://www.dailymail.co.uk/, https://www.mirror.co.uk/news/,
https://www.nydailynews.com/, https://www.usnews.com/,
https://www.washingtonpost.com/, https://www.boston.com, https://techcrunch.com/

39 https://mashable.com/, https://www.usnews.com/,
https://www.independent.co.uk/, https://www.nydailynews.com/,
https://www.vice.com/en_us, https://abcnews.go.com/,
https://www.theguardian.com/us, https://www.al.com/,
https://www.huffingtonpost.com/, https://techcrunch.com/,
https://www.nbcnews.com/, https://www.upworthy.com/

40 https://www.huffingtonpost.com/, https://www.mirror.co.uk/news/,
https://www.al.com/, https://www.usnews.com/, https://www.businessinsider.com/,
https://www.thedailybeast.com/, https://www.latimes.com/,
https://www.upworthy.com/, https://www.nj.com, https://www.chicagotribune.com/,
https://nypost.com/, https://techcrunch.com/, https://www.buzzfeed.com/

41 https://www.salon.com/, https://mashable.com/, https://www.buzzfeed.com/,
https://www.theatlantic.com/, https://www.boston.com,
https://www.nydailynews.com/, https://www.slate.com/, https://time.com/,
https://www.washingtonpost.com/, https://www.cbsnews.com/,
https://www.telegraph.co.uk/, https://www.dailymail.co.uk/,
https://www.vice.com/en_us, https://www.dallasnews.com/,
https://www.upworthy.com/, https://www.cnn.com/, https://www.nytimes.com/,
https://www.msn.com/en-us/news, https://nypost.com/

42 https://www.independent.co.uk/, https://www.examiner.com/,
https://techcrunch.com/, https://mashable.com/, https://www.washingtonpost.com/,
https://www.usnews.com/, https://time.com/, https://www.theguardian.com/us,
https://www.upworthy.com/, https://www.cnet.com/,

```
https://www.businessinsider.com/, https://www.usatoday.com/, https://nypost.com/
43  https://www.bbc.com/, https://www.dailymail.co.uk/,
https://www.engadget.com/, https://www.chicagotribune.com/, https://www.al.com/,
https://www.mirror.co.uk/news/, https://www.msn.com/en-us/news,
https://www.dallasnews.com/, https://www.nj.com, https://www.boston.com,
https://www.bostonglobe.com/, https://www.washingtonpost.com/,
https://www.buzzfeed.com/, https://www.nydailynews.com/
44  https://www.vice.com/en_us, https://www.cnet.com/, https://www.npr.org/,
https://mashable.com/, https://techcrunch.com/, https://www.slate.com/,
https://www.washingtonpost.com/, https://www.vox.com/,
https://www.examiner.com/, https://www.usatoday.com/,
https://www.thedailybeast.com/, https://www.usnews.com/, https://www.bbc.com/,
https://www.telegraph.co.uk/, https://www.independent.co.uk/,
https://www.dallasnews.com/, https://nypost.com/, https://www.nj.com,
https://www.buzzfeed.com/
45  https://www.cnet.com/, https://www.dallasnews.com/,
https://www.businessinsider.com/, https://www.usatoday.com/,
https://www.vice.com/en_us, https://www.salon.com/, https://www.nytimes.com/,
https://www.mirror.co.uk/news/, https://www.washingtonpost.com/,
https://techcrunch.com/, https://mashable.com/, https://www.upworthy.com/,
https://www.msn.com/en-us/news, https://www.buzzfeed.com/,
https://www.dailymail.co.uk/, https://www.independent.co.uk/,
https://www.latimes.com/
46  https://www.bostonglobe.com/, https://abcnews.go.com/,
https://www.latimes.com/, https://www.al.com/, https://www.dallasnews.com/,
https://time.com/, https://www.salon.com/, https://www.vox.com/,
https://techcrunch.com/, https://www.independent.co.uk/,
https://www.nydailynews.com/, https://www.usnews.com/, https://nypost.com/,
https://www.usatoday.com/, https://www.nytimes.com/, https://www.cnn.com/
47  https://www.theatlantic.com/, https://www.boston.com,
https://www.bostonglobe.com/, https://www.dallasnews.com/,
https://www.upworthy.com/, https://www.mirror.co.uk/news/,
https://www.cbsnews.com/, https://www.nytimes.com/, https://www.cnn.com/,
https://www.vox.com/, https://www.huffingtonpost.com/, https://techcrunch.com/,
https://www.businessinsider.com/, https://www.buzzfeed.com/, https://www.al.com/
48  https://www.nbcnews.com/, https://www.nytimes.com/,
https://www.usatoday.com/, https://www.boston.com,
https://www.independent.co.uk/, https://www.washingtonpost.com/,
https://www.telegraph.co.uk/, https://www.nydailynews.com/,
https://www.latimes.com/, https://www.cnn.com/, https://www.buzzfeed.com/
```

7 Task 4

For each product type, compute all the Publication_URLs containing an ad for that product type. Your solution must be scalable. That is, it should work well even if there are hundreds of products in each product_type and there are hundreds of product_types. (Hint: To make it scalable you should consider using a Python or PySpark script instead of a SQL query.) (20)

8 Answer 4

In order to find all the unique Publication_URLs containing an ad for that product type, I groupby dataframe new ny column "Publication_URL" and display column "product_type". For each product, I extract only unique publication URLs and store them into a list. Then, I convert all the entries of that list from list to string and create a dataframe callend url_product_type containing two columns: product_type and URL_list_unique.

[141]:
```
# I groupby the dataframe new by column "Publication_URL" and display column␣
  ↪"product_type"
number_product_type = new.groupby("product_type")["Publication_URL"]
```

[142]:
```
temp_list = [] # this list contains all the unique Publication_URLs containing␣
  ↪an ad for one product_type
for key, item in number_product_type:
    temp_list.append(number_product_type.get_group(key).unique())
```

[143]:
```
for i in range(len(temp_list)):
    temp_list[i] = temp_list[i].tolist()
```

[144]:
```
# create a new dataframe that stores two columns: product and all the␣
  ↪Publication_URLs containing an ad for that product_type (URL_list_unique)
number_product_type = new.groupby("product_type")["Publication_URL"].count()
url_product_type = number_product_type.reset_index()
url_product_type = url_product_type.rename(columns={"Publication_URL":␣
  ↪"URL_count"})
url_product_type["URL_list_unique"] = temp_list
```

[145]:
```
# convert all the entries of "URL_list_unique" from list to string
# create a new dataframe called url_product_type that stores information of␣
  ↪product_type and unique publication URLs
for i in range(len(url_product_type)):
    url_product_type.iloc[i,2] = ', '.join(url_product_type.iloc[i,2])
url_product_type = url_product_type.iloc[:,[0,2]]
url_product_type
```

[145]:
```
          product_type  \
0   blender
1   car
2   coffee
3   computer
4   dryer
5   elliptical trainer
6   face cream
7   furniture
8   jeans
9   lipstick
```

```
10  makeup
11  pants
12  perfume
13  pressure cooker
14  refrigerator
15  rowing machine
16  shaver
17  speakers
18  tablet
19  television
20  treadmill
21  vitamin
22  washer
23  women's purse

URL_list_unique
0   https://www.bostonglobe.com/, https://abcnews.go.com/,
https://www.latimes.com/, https://www.al.com/, https://www.dallasnews.com/,
https://time.com/, https://www.salon.com/, https://www.vox.com/,
https://techcrunch.com/, https://www.independent.co.uk/,
https://www.nydailynews.com/, https://www.usnews.com/, https://nypost.com/,
https://www.usatoday.com/, https://www.nytimes.com/, https://www.cnn.com/,
https://www.sfgate.com/, https://www.dailymail.co.uk/, https://mashable.com/,
https://www.npr.org/, https://www.cbsnews.com/, https://www.boston.com,
https://www.examiner.com/, https://www.nj.com, https://www.theguardian.com/us,
https://www.foxnews.com/, https://www.nbcnews.com/,
https://www.telegraph.co.uk/, https://www.theatlantic.com/,
https://www.huffingtonpost.com/, https://www.businessinsider.com/,
https://www.upworthy.com/
1   https://www.cnet.com/, https://www.dallasnews.com/,
https://www.businessinsider.com/, https://www.usatoday.com/,
https://www.vice.com/en_us, https://www.salon.com/, https://www.nytimes.com/,
https://www.mirror.co.uk/news/, https://www.washingtonpost.com/,
https://techcrunch.com/, https://mashable.com/, https://www.upworthy.com/,
https://www.msn.com/en-us/news, https://www.buzzfeed.com/,
https://www.dailymail.co.uk/, https://www.independent.co.uk/,
https://www.latimes.com/, https://www.chicagotribune.com/, https://www.npr.org/,
https://www.engadget.com/, https://www.usnews.com/, https://www.boston.com,
https://www.nj.com, https://www.foxnews.com/, https://www.vox.com/
2   https://www.boston.com, https://www.dallasnews.com/,
https://www.dailymail.co.uk/, https://www.examiner.com/, https://nypost.com/,
https://www.cnn.com/, https://www.thedailybeast.com/, https://www.nytimes.com/,
https://www.vice.com/en_us, https://www.cnet.com/, https://www.npr.org/,
https://mashable.com/, https://techcrunch.com/, https://www.slate.com/,
https://www.washingtonpost.com/, https://www.vox.com/,
https://www.usatoday.com/, https://www.usnews.com/, https://www.bbc.com/,
https://www.telegraph.co.uk/, https://www.independent.co.uk/,
```

https://www.nj.com, https://www.buzzfeed.com/
3 https://www.nj.com, https://www.vox.com/, https://www.engadget.com/,
https://www.latimes.com/, https://www.cnn.com/, https://www.cbsnews.com/,
https://www.dallasnews.com/, https://www.usnews.com/, https://www.boston.com,
https://abcnews.go.com/, https://www.salon.com/, https://www.thedailybeast.com/,
https://www.upworthy.com/, https://www.al.com/, https://nypost.com/,
https://www.theatlantic.com/, https://www.msn.com/en-us/news,
https://www.businessinsider.com/, https://www.examiner.com/,
https://www.huffingtonpost.com/, https://www.nydailynews.com/,
https://techcrunch.com/, https://www.buzzfeed.com/,
https://www.bostonglobe.com/, https://www.dailymail.co.uk/,
https://www.telegraph.co.uk/, https://www.vice.com/en_us,
https://www.nbcnews.com/, https://www.chicagotribune.com/,
https://www.usatoday.com/, https://www.washingtonpost.com/,
https://www.nytimes.com/, https://www.mirror.co.uk/news/, https://www.cnet.com/,
https://www.bbc.com/
4 https://www.chicagotribune.com/, https://www.vice.com/en_us,
https://www.salon.com/, https://www.nj.com, https://www.cnn.com/,
https://www.businessinsider.com/, https://www.slate.com/,
https://www.nydailynews.com/, https://www.upworthy.com/,
https://www.engadget.com/, https://time.com/, https://www.thedailybeast.com/,
https://www.vox.com/, https://www.boston.com, https://www.huffingtonpost.com/,
https://www.mirror.co.uk/news/, https://www.al.com/, https://www.usnews.com/,
https://www.latimes.com/, https://nypost.com/, https://techcrunch.com/,
https://www.buzzfeed.com/, https://www.nytimes.com/, https://www.bbc.com/,
https://www.msn.com/en-us/news, https://www.examiner.com/,
https://www.usatoday.com/
5 https://www.mirror.co.uk/news/, https://www.bbc.com/,
https://www.theguardian.com/us, https://abcnews.go.com/,
https://www.nbcnews.com/, https://www.bostonglobe.com/, https://www.usnews.com/,
https://www.boston.com, https://www.nydailynews.com/, https://www.latimes.com/,
https://www.dailymail.co.uk/, https://www.dallasnews.com/, https://www.vox.com/,
https://www.salon.com/
6 https://www.chicagotribune.com/, https://www.latimes.com/,
https://www.huffingtonpost.com/, https://www.salon.com/,
https://www.theguardian.com/us, https://www.bostonglobe.com/,
https://www.examiner.com/, https://www.bbc.com/, https://www.usnews.com/,
https://www.msn.com/en-us/news, https://www.nbcnews.com/,
https://techcrunch.com/, https://www.nytimes.com/, https://www.cnn.com/,
https://nypost.com/, https://www.washingtonpost.com/
7 https://www.cnet.com/, https://www.nbcnews.com/, https://www.buzzfeed.com/,
https://www.al.com/, https://www.slate.com/, https://www.businessinsider.com/,
https://www.nytimes.com/, https://www.nj.com, https://www.huffingtonpost.com/,
https://www.cbsnews.com/, https://nypost.com/, https://www.theatlantic.com/,
https://www.upworthy.com/, https://www.cnn.com/, https://time.com/,
https://www.npr.org/, https://techcrunch.com/, https://www.vice.com/en_us,
https://www.vox.com/, https://www.nydailynews.com/, https://www.usatoday.com/,

https://www.boston.com, https://www.salon.com/

8 https://www.foxnews.com/, https://www.nytimes.com/, https://www.cnn.com/, https://www.chicagotribune.com/, https://www.salon.com/, https://www.engadget.com/, https://www.nydailynews.com/, https://mashable.com/, https://www.theguardian.com/us, https://www.nbcnews.com/, https://www.washingtonpost.com/, https://www.usatoday.com/, https://www.usnews.com/, https://www.theatlantic.com/, https://www.al.com/, https://www.buzzfeed.com/, https://www.nj.com, https://www.latimes.com/, https://www.msn.com/en-us/news, https://www.businessinsider.com/, https://www.vox.com/, https://nypost.com/, https://www.sfgate.com/, https://www.slate.com/, https://time.com/, https://www.cbsnews.com/, https://techcrunch.com/, https://www.boston.com, https://www.dallasnews.com/, https://www.bostonglobe.com/, https://www.telegraph.co.uk/

9 https://www.nbcnews.com/, https://www.nytimes.com/, https://www.usatoday.com/, https://www.boston.com, https://www.independent.co.uk/, https://www.washingtonpost.com/, https://www.telegraph.co.uk/, https://www.nydailynews.com/, https://www.latimes.com/, https://www.cnn.com/, https://www.buzzfeed.com/, https://www.slate.com/, https://www.theatlantic.com/, https://www.dallasnews.com/, https://www.msn.com/en-us/news, https://www.huffingtonpost.com/, https://www.nj.com, https://www.engadget.com/, https://www.businessinsider.com/, https://nypost.com/, https://www.vice.com/en_us

10 https://www.examiner.com/, https://www.vox.com/, https://www.bostonglobe.com/, https://www.cnn.com/, https://www.nj.com, https://time.com/, https://www.theatlantic.com/, https://www.usatoday.com/, https://www.buzzfeed.com/, https://www.engadget.com/, https://www.nydailynews.com/, https://www.bbc.com/, https://abcnews.go.com/, https://www.dallasnews.com/, https://nypost.com/, https://www.upworthy.com/

11 https://www.bostonglobe.com/, https://www.thedailybeast.com/, https://www.chicagotribune.com/, https://www.nydailynews.com/, https://www.businessinsider.com/, https://www.nbcnews.com/, https://www.al.com/, https://www.usnews.com/, https://nypost.com/, https://www.washingtonpost.com/, https://time.com/, https://www.upworthy.com/, https://www.msn.com/en-us/news

12 https://www.salon.com/, https://www.independent.co.uk/, https://www.chicagotribune.com/, https://www.buzzfeed.com/, https://techcrunch.com/, https://www.examiner.com/, https://www.engadget.com/, https://www.nydailynews.com/, https://www.businessinsider.com/, https://www.msn.com/en-us/news, https://www.vice.com/en_us, https://www.nytimes.com/, https://www.cnn.com/, https://www.washingtonpost.com/, https://www.boston.com, https://www.vox.com/, https://www.huffingtonpost.com/, https://www.dallasnews.com/, https://www.upworthy.com/, https://www.usnews.com/, https://www.latimes.com/, https://www.telegraph.co.uk/, https://www.sfgate.com/, https://www.bbc.com/, https://www.nj.com, https://mashable.com/, https://www.theguardian.com/us, https://www.dailymail.co.uk/, https://www.bostonglobe.com/, https://nypost.com/, https://www.usatoday.com/, https://www.slate.com/, https://www.mirror.co.uk/news/, https://abcnews.go.com/,

https://www.foxnews.com/, https://time.com/

13 https://www.buzzfeed.com/, https://www.latimes.com/, https://abcnews.go.com/, https://www.dallasnews.com/, https://www.nytimes.com/, https://www.bbc.com/, https://mashable.com/, https://www.al.com/, https://www.dailymail.co.uk/, https://www.upworthy.com/, https://www.theguardian.com/us, https://www.examiner.com/, https://www.nj.com, https://www.cnn.com/

14 https://www.cbsnews.com/, https://www.foxnews.com/, https://www.bbc.com/, https://abcnews.go.com/, https://www.businessinsider.com/, https://www.usatoday.com/, https://nypost.com/, https://www.examiner.com/, https://www.msn.com/en-us/news, https://www.buzzfeed.com/, https://www.bostonglobe.com/, https://techcrunch.com/, https://time.com/, https://www.washingtonpost.com/, https://www.nbcnews.com/, https://www.huffingtonpost.com/, https://www.usnews.com/

15 https://www.nbcnews.com/, https://www.usatoday.com/, https://www.nj.com, https://www.boston.com, https://nypost.com/, https://www.cbsnews.com/, https://www.vox.com/, https://www.nydailynews.com/, https://www.bbc.com/, https://mashable.com/, https://abcnews.go.com/, https://www.upworthy.com/, https://www.bostonglobe.com/, https://www.cnn.com/, https://www.dallasnews.com/, https://www.examiner.com/, https://techcrunch.com/

16 https://www.chicagotribune.com/, https://www.businessinsider.com/, https://www.theguardian.com/us, https://www.vox.com/, https://www.dailymail.co.uk/, https://www.mirror.co.uk/news/, https://www.nydailynews.com/, https://www.usnews.com/, https://www.washingtonpost.com/, https://www.boston.com, https://techcrunch.com/, https://www.bbc.com/, https://www.cnet.com/, https://www.vice.com/en_us, https://www.huffingtonpost.com/, https://www.thedailybeast.com/, https://www.nytimes.com/, https://www.cnn.com/

17 https://www.theatlantic.com/, https://www.boston.com, https://www.bostonglobe.com/, https://www.dallasnews.com/, https://www.upworthy.com/, https://www.mirror.co.uk/news/, https://www.cbsnews.com/, https://www.nytimes.com/, https://www.cnn.com/, https://www.vox.com/, https://www.huffingtonpost.com/, https://techcrunch.com/, https://www.businessinsider.com/, https://www.buzzfeed.com/, https://www.al.com/, https://www.bbc.com/, https://www.dailymail.co.uk/, https://www.engadget.com/, https://www.chicagotribune.com/, https://www.msn.com/en-us/news, https://www.nj.com, https://www.washingtonpost.com/, https://www.nydailynews.com/

18 https://mashable.com/, https://www.boston.com, https://www.nj.com, https://www.sfgate.com/, https://www.mirror.co.uk/news/, https://www.businessinsider.com/, https://www.cnn.com/, https://www.vice.com/en_us, https://www.npr.org/, https://www.slate.com/, https://www.chicagotribune.com/, https://nypost.com/, https://www.latimes.com/, https://www.theguardian.com/us, https://www.examiner.com/, https://www.telegraph.co.uk/, https://www.al.com/

19 https://www.independent.co.uk/, https://www.examiner.com/, https://techcrunch.com/, https://mashable.com/, https://www.washingtonpost.com/,

https://www.usnews.com/, https://time.com/, https://www.theguardian.com/us,
https://www.upworthy.com/, https://www.cnet.com/,
https://www.businessinsider.com/, https://www.usatoday.com/,
https://nypost.com/, https://www.nbcnews.com/, https://www.salon.com/,
https://www.vice.com/en_us, https://www.telegraph.co.uk/,
https://www.nytimes.com/, https://www.buzzfeed.com/, https://www.foxnews.com/,
https://www.al.com/, https://www.nydailynews.com/, https://abcnews.go.com/,
https://www.huffingtonpost.com/

20 https://www.nytimes.com/, https://www.boston.com,
https://www.washingtonpost.com/, https://www.vice.com/en_us,
https://www.msn.com/en-us/news, https://www.independent.co.uk/,
https://www.upworthy.com/, https://www.nbcnews.com/,
https://www.bostonglobe.com/, https://www.businessinsider.com/,
https://mashable.com/, https://time.com/, https://www.usatoday.com/,
https://www.nydailynews.com/, https://www.latimes.com/, https://www.salon.com/,
https://www.usnews.com/, https://techcrunch.com/, https://www.buzzfeed.com/

21 https://mashable.com/, https://time.com/, https://www.nytimes.com/,
https://www.independent.co.uk/, https://www.vox.com/, https://www.latimes.com/,
https://www.washingtonpost.com/, https://www.al.com/,
https://www.mirror.co.uk/news/, https://www.cnn.com/, https://www.upworthy.com/,
https://www.engadget.com/, https://www.boston.com, https://techcrunch.com/,
https://www.thedailybeast.com/, https://www.dallasnews.com/,
https://www.usatoday.com/, https://www.nydailynews.com/,
https://www.buzzfeed.com/

22 https://www.salon.com/, https://mashable.com/, https://www.buzzfeed.com/,
https://www.theatlantic.com/, https://www.boston.com,
https://www.nydailynews.com/, https://www.slate.com/, https://time.com/,
https://www.washingtonpost.com/, https://www.cbsnews.com/,
https://www.telegraph.co.uk/, https://www.dailymail.co.uk/,
https://www.vice.com/en_us, https://www.dallasnews.com/,
https://www.upworthy.com/, https://www.cnn.com/, https://www.nytimes.com/,
https://www.msn.com/en-us/news, https://nypost.com/, https://www.latimes.com/,
https://www.independent.co.uk/, https://www.sfgate.com/,
https://www.chicagotribune.com/, https://www.bostonglobe.com/,
https://www.vox.com/, https://www.theguardian.com/us, https://www.examiner.com/,
https://www.mirror.co.uk/news/, https://techcrunch.com/, https://www.npr.org/,
https://www.thedailybeast.com/, https://www.nbcnews.com/, https://www.nj.com,
https://www.businessinsider.com/

23 https://www.mirror.co.uk/news/, https://www.foxnews.com/,
https://www.engadget.com/, https://abcnews.go.com/, https://www.salon.com/,
https://www.chicagotribune.com/, https://www.upworthy.com/,
https://www.washingtonpost.com/, https://www.cnn.com/, https://techcrunch.com/,
https://www.nbcnews.com/, https://www.businessinsider.com/, https://www.al.com/,
https://www.vox.com/, https://www.nytimes.com/, https://www.nj.com,
https://www.boston.com, https://www.nydailynews.com/,
https://www.telegraph.co.uk/, https://www.usatoday.com/,
https://www.dallasnews.com/, https://www.buzzfeed.com/, https://www.slate.com/,

```
https://nypost.com/
```

9 Task 5

Save this information in the database. Should you save it in the products table or the product_categories table or should you create a new table, product_type_pubURLs, and save this information in this table? If you create a new table, make sure to set up all the appropriate foreign key constraints. On the other hand, if you use one of the existing tables, explain how you will avoid redundancy in your data. In either case, justify your decision. (10)

10 Answer 5

In this case, we should save the information URL_list_unique in the existing product_categories (category) table. The reason for this is to avoid redundancy caused by creating new table product_type_pubURLs. Moreover, since product_categories dataframe and url_product_type dataframe have the same column “product_type”, it is easy to match and append the unique publication URLs information.

To implement, I connect to the DATA1050FP database and create two tables: products and product_categories. The primary key for the product_categories table is product_type and there are no foreign key constraint. The primary key for the products table is product and there is one foriegn key constraint: a foreign key constraint from products to product_categories on product_type field.

[146]:
```
category = category.merge(url_product_type, on = "product_type")
```

[147]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Zhiruil1023!",    # REPLACE THIS WITH THE PASSWORD YOU SET
  database = "DATA1050FP" # connecting to database
)

print(mydb)

if mydb.is_connected():
    print("CONNECTION SUCCESSFUL")
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x137b2fa90>
CONNECTION SUCCESSFUL
```

[148]:
```
mycursor = mydb.cursor()
```

[149]:
```
mycursor.execute("DROP TABLE IF EXISTS product_type_sentiment_clickrate")
mycursor.execute("DROP TABLE IF EXISTS products")
mycursor.execute("DROP TABLE IF EXISTS product_categories")
```

```
# create table product_categories in DATA1050FP database

mycursor.execute("CREATE TABLE DATA1050FP.product_categories \
    (product_type VARCHAR(100) NOT NULL, \
     category VARCHAR(100), \
     URL_list_unique text, \
     PRIMARY KEY (product_type)) ")
mycursor.execute("SHOW TABLES")
for x in mycursor:
  print(x)
```

```
('product_categories',)
```

[150]:

```
# insert rows in product_categories table
for i, row in category.iterrows():
    mycursor.execute("INSERT INTO product_categories VALUES (%s, %s, %s)",␣
 ↪tuple(row))
    mydb.commit()
```

[151]:

```
# print rows in product_categories table
mycursor.execute("SELECT * FROM product_categories")
result = mycursor.fetchall()
for row in result:
    print(row)
    print("\n")
```

```
('blender', 'small kitchen appliances', 'https://www.bostonglobe.com/,
https://abcnews.go.com/, https://www.latimes.com/, https://www.al.com/,
https://www.dallasnews.com/, https://time.com/, https://www.salon.com/,
https://www.vox.com/, https://techcrunch.com/, https://www.independent.co.uk/,
https://www.nydailynews.com/, https://www.usnews.com/, https://nypost.com/,
https://www.usatoday.com/, https://www.nytimes.com/, https://www.cnn.com/,
https://www.sfgate.com/, https://www.dailymail.co.uk/, https://mashable.com/,
https://www.npr.org/, https://www.cbsnews.com/, https://www.boston.com,
https://www.examiner.com/, https://www.nj.com, https://www.theguardian.com/us,
https://www.foxnews.com/, https://www.nbcnews.com/,
https://www.telegraph.co.uk/, https://www.theatlantic.com/,
https://www.huffingtonpost.com/, https://www.businessinsider.com/,
https://www.upworthy.com/')


('car', 'transportation', 'https://www.cnet.com/, https://www.dallasnews.com/,
https://www.businessinsider.com/, https://www.usatoday.com/,
https://www.vice.com/en_us, https://www.salon.com/, https://www.nytimes.com/,
https://www.mirror.co.uk/news/, https://www.washingtonpost.com/,
https://techcrunch.com/, https://mashable.com/, https://www.upworthy.com/,
https://www.msn.com/en-us/news, https://www.buzzfeed.com/,
https://www.dailymail.co.uk/, https://www.independent.co.uk/,
https://www.latimes.com/, https://www.chicagotribune.com/, https://www.npr.org/,
https://www.engadget.com/, https://www.usnews.com/, https://www.boston.com,
https://www.nj.com, https://www.foxnews.com/, https://www.vox.com/')
```

```
('coffee', 'packaged food', 'https://www.boston.com,
https://www.dallasnews.com/, https://www.dailymail.co.uk/,
https://www.examiner.com/, https://nypost.com/, https://www.cnn.com/,
https://www.thedailybeast.com/, https://www.nytimes.com/,
https://www.vice.com/en_us, https://www.cnet.com/, https://www.npr.org/,
https://mashable.com/, https://techcrunch.com/, https://www.slate.com/,
https://www.washingtonpost.com/, https://www.vox.com/,
https://www.usatoday.com/, https://www.usnews.com/, https://www.bbc.com/,
https://www.telegraph.co.uk/, https://www.independent.co.uk/,
https://www.nj.com, https://www.buzzfeed.com/')
```

```
('computer', 'consumer electronics', 'https://www.nj.com, https://www.vox.com/,
https://www.engadget.com/, https://www.latimes.com/, https://www.cnn.com/,
https://www.cbsnews.com/, https://www.dallasnews.com/, https://www.usnews.com/,
https://www.boston.com, https://abcnews.go.com/, https://www.salon.com/,
https://www.thedailybeast.com/, https://www.upworthy.com/, https://www.al.com/,
https://nypost.com/, https://www.theatlantic.com/, https://www.msn.com/en-
us/news, https://www.businessinsider.com/, https://www.examiner.com/,
https://www.huffingtonpost.com/, https://www.nydailynews.com/,
https://techcrunch.com/, https://www.buzzfeed.com/,
https://www.bostonglobe.com/, https://www.dailymail.co.uk/,
https://www.telegraph.co.uk/, https://www.vice.com/en_us,
https://www.nbcnews.com/, https://www.chicagotribune.com/,
https://www.usatoday.com/, https://www.washingtonpost.com/,
https://www.nytimes.com/, https://www.mirror.co.uk/news/, https://www.cnet.com/,
https://www.bbc.com/')
```

```
('dryer', 'large kitchen appliances', 'https://www.chicagotribune.com/,
https://www.vice.com/en_us, https://www.salon.com/, https://www.nj.com,
https://www.cnn.com/, https://www.businessinsider.com/, https://www.slate.com/,
https://www.nydailynews.com/, https://www.upworthy.com/,
https://www.engadget.com/, https://time.com/, https://www.thedailybeast.com/,
https://www.vox.com/, https://www.boston.com, https://www.huffingtonpost.com/,
https://www.mirror.co.uk/news/, https://www.al.com/, https://www.usnews.com/,
https://www.latimes.com/, https://nypost.com/, https://techcrunch.com/,
https://www.buzzfeed.com/, https://www.nytimes.com/, https://www.bbc.com/,
https://www.msn.com/en-us/news, https://www.examiner.com/,
https://www.usatoday.com/')
```

```
('elliptical trainer', 'fitness equipment', 'https://www.mirror.co.uk/news/,
https://www.bbc.com/, https://www.theguardian.com/us, https://abcnews.go.com/,
https://www.nbcnews.com/, https://www.bostonglobe.com/, https://www.usnews.com/,
https://www.boston.com, https://www.nydailynews.com/, https://www.latimes.com/,
https://www.dailymail.co.uk/, https://www.dallasnews.com/, https://www.vox.com/,
https://www.salon.com/')


('face cream', 'beauty products', 'https://www.chicagotribune.com/,
https://www.latimes.com/, https://www.huffingtonpost.com/,
https://www.salon.com/, https://www.theguardian.com/us,
https://www.bostonglobe.com/, https://www.examiner.com/, https://www.bbc.com/,
https://www.usnews.com/, https://www.msn.com/en-us/news,
https://www.nbcnews.com/, https://techcrunch.com/, https://www.nytimes.com/,
https://www.cnn.com/, https://nypost.com/, https://www.washingtonpost.com/')


('furniture', 'household durables', 'https://www.cnet.com/,
https://www.nbcnews.com/, https://www.buzzfeed.com/, https://www.al.com/,
https://www.slate.com/, https://www.businessinsider.com/,
https://www.nytimes.com/, https://www.nj.com, https://www.huffingtonpost.com/,
https://www.cbsnews.com/, https://nypost.com/, https://www.theatlantic.com/,
https://www.upworthy.com/, https://www.cnn.com/, https://time.com/,
https://www.npr.org/, https://techcrunch.com/, https://www.vice.com/en_us,
https://www.vox.com/, https://www.nydailynews.com/, https://www.usatoday.com/,
https://www.boston.com, https://www.salon.com/')


('jeans', 'apparel', 'https://www.foxnews.com/, https://www.nytimes.com/,
https://www.cnn.com/, https://www.chicagotribune.com/, https://www.salon.com/,
https://www.engadget.com/, https://www.nydailynews.com/, https://mashable.com/,
https://www.theguardian.com/us, https://www.nbcnews.com/,
https://www.washingtonpost.com/, https://www.usatoday.com/,
https://www.usnews.com/, https://www.theatlantic.com/, https://www.al.com/,
https://www.buzzfeed.com/, https://www.nj.com, https://www.latimes.com/,
https://www.msn.com/en-us/news, https://www.businessinsider.com/,
https://www.vox.com/, https://nypost.com/, https://www.sfgate.com/,
https://www.slate.com/, https://time.com/, https://www.cbsnews.com/,
https://techcrunch.com/, https://www.boston.com, https://www.dallasnews.com/,
https://www.bostonglobe.com/, https://www.telegraph.co.uk/')


('lipstick', 'beauty products', 'https://www.nbcnews.com/,
https://www.nytimes.com/, https://www.usatoday.com/, https://www.boston.com,
https://www.independent.co.uk/, https://www.washingtonpost.com/,
https://www.telegraph.co.uk/, https://www.nydailynews.com/,
https://www.latimes.com/, https://www.cnn.com/, https://www.buzzfeed.com/,
https://www.slate.com/, https://www.theatlantic.com/,
https://www.dallasnews.com/, https://www.msn.com/en-us/news,
https://www.huffingtonpost.com/, https://www.nj.com, https://www.engadget.com/,
https://www.businessinsider.com/, https://nypost.com/,
https://www.vice.com/en_us')


('makeup', 'beauty products', 'https://www.examiner.com/, https://www.vox.com/,
https://www.bostonglobe.com/, https://www.cnn.com/, https://www.nj.com,
https://time.com/, https://www.theatlantic.com/, https://www.usatoday.com/,
https://www.buzzfeed.com/, https://www.engadget.com/,
https://www.nydailynews.com/, https://www.bbc.com/, https://abcnews.go.com/,
https://www.dallasnews.com/, https://nypost.com/, https://www.upworthy.com/')


('pants', 'apparel', 'https://www.bostonglobe.com/,
https://www.thedailybeast.com/, https://www.chicagotribune.com/,
https://www.nydailynews.com/, https://www.businessinsider.com/,
https://www.nbcnews.com/, https://www.al.com/, https://www.usnews.com/,
https://nypost.com/, https://www.washingtonpost.com/, https://time.com/,
https://www.upworthy.com/, https://www.msn.com/en-us/news')


('perfume', 'beauty products', 'https://www.salon.com/,
https://www.independent.co.uk/, https://www.chicagotribune.com/,
https://www.buzzfeed.com/, https://techcrunch.com/, https://www.examiner.com/,
https://www.engadget.com/, https://www.nydailynews.com/,
https://www.businessinsider.com/, https://www.msn.com/en-us/news,
https://www.vice.com/en_us, https://www.nytimes.com/, https://www.cnn.com/,
https://www.washingtonpost.com/, https://www.boston.com, https://www.vox.com/,
https://www.huffingtonpost.com/, https://www.dallasnews.com/,
https://www.upworthy.com/, https://www.usnews.com/, https://www.latimes.com/,
https://www.telegraph.co.uk/, https://www.sfgate.com/, https://www.bbc.com/,
https://www.nj.com, https://mashable.com/, https://www.theguardian.com/us,
https://www.dailymail.co.uk/, https://www.bostonglobe.com/, https://nypost.com/,
https://www.usatoday.com/, https://www.slate.com/,
https://www.mirror.co.uk/news/, https://abcnews.go.com/,
https://www.foxnews.com/, https://time.com/')


('pressure cooker', 'small kitchen appliances', 'https://www.buzzfeed.com/,
https://www.latimes.com/, https://abcnews.go.com/, https://www.dallasnews.com/,
https://www.nytimes.com/, https://www.bbc.com/, https://mashable.com/,
https://www.al.com/, https://www.dailymail.co.uk/, https://www.upworthy.com/,
https://www.theguardian.com/us, https://www.examiner.com/, https://www.nj.com,
https://www.cnn.com/')


('refrigerator', 'large kitchen appliances', 'https://www.cbsnews.com/,
https://www.foxnews.com/, https://www.bbc.com/, https://abcnews.go.com/,
https://www.businessinsider.com/, https://www.usatoday.com/,
https://nypost.com/, https://www.examiner.com/, https://www.msn.com/en-us/news,
https://www.buzzfeed.com/, https://www.bostonglobe.com/,
https://techcrunch.com/, https://time.com/, https://www.washingtonpost.com/,
https://www.nbcnews.com/, https://www.huffingtonpost.com/,
https://www.usnews.com/')


('rowing machine', 'fitness equipment', 'https://www.nbcnews.com/,
https://www.usatoday.com/, https://www.nj.com, https://www.boston.com,
https://nypost.com/, https://www.cbsnews.com/, https://www.vox.com/,
https://www.nydailynews.com/, https://www.bbc.com/, https://mashable.com/,
https://abcnews.go.com/, https://www.upworthy.com/,
https://www.bostonglobe.com/, https://www.cnn.com/, https://www.dallasnews.com/,
https://www.examiner.com/, https://techcrunch.com/')


('shaver', 'consumer electronics', 'https://www.chicagotribune.com/,
https://www.businessinsider.com/, https://www.theguardian.com/us,
https://www.vox.com/, https://www.dailymail.co.uk/,
https://www.mirror.co.uk/news/, https://www.nydailynews.com/,
https://www.usnews.com/, https://www.washingtonpost.com/,
https://www.boston.com, https://techcrunch.com/, https://www.bbc.com/,
https://www.cnet.com/, https://www.vice.com/en_us,
https://www.huffingtonpost.com/, https://www.thedailybeast.com/,
https://www.nytimes.com/, https://www.cnn.com/')


('speakers', 'consumer electronics', 'https://www.theatlantic.com/,
https://www.boston.com, https://www.bostonglobe.com/,
https://www.dallasnews.com/, https://www.upworthy.com/,
https://www.mirror.co.uk/news/, https://www.cbsnews.com/,
https://www.nytimes.com/, https://www.cnn.com/, https://www.vox.com/,
https://www.huffingtonpost.com/, https://techcrunch.com/,
https://www.businessinsider.com/, https://www.buzzfeed.com/,
https://www.al.com/, https://www.bbc.com/, https://www.dailymail.co.uk/,
https://www.engadget.com/, https://www.chicagotribune.com/,
https://www.msn.com/en-us/news, https://www.nj.com,
https://www.washingtonpost.com/, https://www.nydailynews.com/')


('tablet', 'consumer electronics', 'https://mashable.com/,
https://www.boston.com, https://www.nj.com, https://www.sfgate.com/,
https://www.mirror.co.uk/news/, https://www.businessinsider.com/,
https://www.cnn.com/, https://www.vice.com/en_us, https://www.npr.org/,
https://www.slate.com/, https://www.chicagotribune.com/, https://nypost.com/,
https://www.latimes.com/, https://www.theguardian.com/us,
https://www.examiner.com/, https://www.telegraph.co.uk/, https://www.al.com/')


('television', 'consumer electronics', 'https://www.independent.co.uk/,
https://www.examiner.com/, https://techcrunch.com/, https://mashable.com/,
https://www.washingtonpost.com/, https://www.usnews.com/, https://time.com/,
https://www.theguardian.com/us, https://www.upworthy.com/,
https://www.cnet.com/, https://www.businessinsider.com/,
https://www.usatoday.com/, https://nypost.com/, https://www.nbcnews.com/,
https://www.salon.com/, https://www.vice.com/en_us,
https://www.telegraph.co.uk/, https://www.nytimes.com/,
https://www.buzzfeed.com/, https://www.foxnews.com/, https://www.al.com/,
https://www.nydailynews.com/, https://abcnews.go.com/,
https://www.huffingtonpost.com/')


('treadmill', 'fitness equipment', 'https://www.nytimes.com/,
https://www.boston.com, https://www.washingtonpost.com/,
https://www.vice.com/en_us, https://www.msn.com/en-us/news,
https://www.independent.co.uk/, https://www.upworthy.com/,
https://www.nbcnews.com/, https://www.bostonglobe.com/,
https://www.businessinsider.com/, https://mashable.com/, https://time.com/,
https://www.usatoday.com/, https://www.nydailynews.com/,
https://www.latimes.com/, https://www.salon.com/, https://www.usnews.com/,
https://techcrunch.com/, https://www.buzzfeed.com/')


('vitamin', 'health', 'https://mashable.com/, https://time.com/,
https://www.nytimes.com/, https://www.independent.co.uk/, https://www.vox.com/,
https://www.latimes.com/, https://www.washingtonpost.com/, https://www.al.com/,
https://www.mirror.co.uk/news/, https://www.cnn.com/, https://www.upworthy.com/,
https://www.engadget.com/, https://www.boston.com, https://techcrunch.com/,
https://www.thedailybeast.com/, https://www.dallasnews.com/,
https://www.usatoday.com/, https://www.nydailynews.com/,
https://www.buzzfeed.com/')


('washer', 'large kitchen appliances', 'https://www.salon.com/,
https://mashable.com/, https://www.buzzfeed.com/, https://www.theatlantic.com/,
https://www.boston.com, https://www.nydailynews.com/, https://www.slate.com/,
https://time.com/, https://www.washingtonpost.com/, https://www.cbsnews.com/,
https://www.telegraph.co.uk/, https://www.dailymail.co.uk/,
https://www.vice.com/en_us, https://www.dallasnews.com/,
https://www.upworthy.com/, https://www.cnn.com/, https://www.nytimes.com/,
https://www.msn.com/en-us/news, https://nypost.com/, https://www.latimes.com/,
https://www.independent.co.uk/, https://www.sfgate.com/,
https://www.chicagotribune.com/, https://www.bostonglobe.com/,
https://www.vox.com/, https://www.theguardian.com/us, https://www.examiner.com/,
https://www.mirror.co.uk/news/, https://techcrunch.com/, https://www.npr.org/,
https://www.thedailybeast.com/, https://www.nbcnews.com/, https://www.nj.com,
https://www.businessinsider.com/')


("women's purse", 'accessories', 'https://www.mirror.co.uk/news/,
https://www.foxnews.com/, https://www.engadget.com/, https://abcnews.go.com/,
https://www.salon.com/, https://www.chicagotribune.com/,
https://www.upworthy.com/, https://www.washingtonpost.com/,
https://www.cnn.com/, https://techcrunch.com/, https://www.nbcnews.com/,
https://www.businessinsider.com/, https://www.al.com/, https://www.vox.com/,
https://www.nytimes.com/, https://www.nj.com, https://www.boston.com,
https://www.nydailynews.com/, https://www.telegraph.co.uk/,
https://www.usatoday.com/, https://www.dallasnews.com/,
https://www.buzzfeed.com/, https://www.slate.com/, https://nypost.com/')
```

[152]:

```
# create table product_categories in DATA1050FP database
mycursor.execute("CREATE TABLE DATA1050FP.products \
    (product VARCHAR(100) NOT NULL, \
     product_URL VARCHAR(100), \
     product_type VARCHAR(100), \
     PRIMARY KEY (product), \
     FOREIGN KEY (product_type) REFERENCES product_categories(product_type)) ")
mycursor.execute("SHOW TABLES")
for x in mycursor:
  print(x)
```

```
('product_categories',)
('products',)
```

[153]:

```
# insert rows in product_categories table
for i, row in products.iterrows():
    mycursor.execute("INSERT INTO products VALUES (%s, %s, %s)", tuple(row))
    mydb.commit()
```

[154]:

```
# print rows in product_categories table
mycursor.execute("SELECT * FROM products")
result = mycursor.fetchall()
for row in result:
    print(row)
    print("\n")
```

```
('Apple computer', 'https://apple.com/computers', 'computer')


('Apple iPad', 'https://apple.com/ipads', 'tablet')
```

```
('Apple laptop', 'https://apple.com/laptops', 'computer')

('BasilBasel perfume', 'https://basilbasel.io/perfumes', 'perfume')

('bose speakers', 'https://bose.com/speakers', 'speakers')

('Broyhill recliner', 'https://broyhill.com/recliners', 'furniture')

('Centrum MultiVitamins', 'https://centrum.com/vitamins', 'vitamin')

('Clinique moisturizer', 'https://clinique.com/moisturizers', 'face cream')

('Coach purse', 'https://coach.com/purses', "women's purse")

('Cougar jeans', 'https://cougar.co/jeans', 'jeans')

('covergirl lipstick', 'https://covergirl.co/lipsticks', 'lipstick')

('Covergirl makeup', 'https://covergirl.co/makeup', 'makeup')

('Dell computer', 'https://dell.com/computers', 'computer')

('Dell laptop', 'https://dell.com/laptops', 'computer')

('Docker pants', 'https://docker.com/pants', 'pants')

('Ford  sedan', 'https://ford.com/sedans', 'car')

('Gillette shaver', 'https://gillette.com/shavers', 'shaver')

('Giorgio perfume', 'https://giorgio.com/perfumes', 'perfume')
```

```
('Givenchy perfume', 'https://givenchy.com/perfumes', 'perfume')

('Guess perfume', 'https://guess.com/perfumes', 'perfume')

('Haier refrigerator', 'https://haier.com/refrigerators', 'refrigerator')

('Hamilton Beach blender', 'https://HamiltonBeach/blenders', 'blender')

('Ikea sofa', 'https://Ikea.com/sofas', 'furniture')

('InstantPot pressure cooker', 'https://InstantPot.com/cookers', 'pressure
cooker')

('Jaguar perfume', 'https://jaguar.co/perfumes', 'perfume')

('Kaai handbags', 'https://kaai.com/handbags', "women's purse")

('Lavazza Coffee', 'https://Lavazza.com/coffee', 'coffee')

('Lee jeans', 'https://lees.com/jeans', 'jeans')

('Lenova laptop', 'https://lenova.com/laptops', 'computer')

('Levis Jeans', 'https://levis.com/jeans', 'jeans')

('LG dryer', 'https://lg.com/dryers', 'dryer')

('LG TV', 'https://lg.com/tvs', 'television')

('LG washer', 'https://lg.com/washers', 'washer')

('Maybelline lipstick', 'http://maybelline.com/lipstick', 'lipstick')
```

```
('Maytag dryer', 'https://maytag.com/dryers', 'dryer')

('Maytag refrigerator', 'https://maytag.com/refrigerators', 'refrigerator')

('Maytag washer', 'https://maytag.com/washers', 'washer')

('NemoK blender', 'http://nemoK.co/blenders', 'blender')

('NordicTrack elliptical', 'https://NordicTrack/elliptical', 'elliptical
trainer')

('NordicTrack rower', 'https://NordicTrack.com/rowers', 'rowing machine')

('NordicTrack treadmill', 'https://NordicTrack.com/treadmills', 'treadmill')

('Remington shaver', 'https://remington.com/shavers', 'shaver')

('Samsung dryer', 'https://samsung.com/dryers', 'dryer')

('Samsung TV', 'https://samsung.com/televisions', 'television')

('Samsung washer', 'https://samsung.com/washers', 'washer')

('Sony TV', 'https://sony.com/televisions', 'television')

('Soundwave speakers', 'https://soundwave.ai/speakers', 'speakers')

('Starbucks Coffee', 'https://Starbucks.com/coffee', 'coffee')

('Tesla', 'https://tesla.com', 'car')
```

```
('Vitamix blender', 'https://vitamix.com/blenders', 'blender')
```

11 Task 6

For each product, compute the click rate for it. (Click rate is the number of times a display of an ad was clicked on (by any user) divided by the number of times it was displayed (to any user). That is, the click rate is not specific to each user.) (10)

12 Answer 6

In order to compute the click rate for each product, I first merge two dataframes log and products to create a new dataframe called new. Then I groupby dataframe new by column "clickORnot" and display two columns "product" and "clickORnot". Then I convert the result to a dataframe and compute click rate for each product by the formula: $\frac{total\ number\ of\ an\ ad\ is\ clicked\ (clickORnot\ =\ 0)}{total\ number\ of\ an\ ad\ is\ displayed\ (clickORnot\ =\ 0\ and\ 1)}$

```
[155]: # merge dataframes log and products
       new = pd.merge(log, products, on = "product_URL")
```

```
[156]: # group by "clickORnot"
       click_rate = new.groupby(["product", "clickORnot"])["clickORnot"].count()
```

```
[157]: # create a new dataframe called click_rate
       click_rate = pd.DataFrame(click_rate)
       click_rate = click_rate.rename(columns={"clickORnot": "count"})
       click_rate = click_rate.reset_index()
       click_rate.head()
```

```
[157]:           product  clickORnot  count
       0  Apple computer  0           42
       1  Apple computer  1           161
       2  Apple iPad      0           131
       3  Apple iPad      1           133
       4  Apple laptop    0           54
```

```
[158]: # calculating the click rate for each product
       click_rate_product = []
       for i in range(0,len(click_rate)-1,2):
           ans = click_rate.iloc[(i+1),2] / (click_rate.iloc[i,2] + click_rate.
        ↪iloc[(i+1),2])
           click_rate_product.append([click_rate.iloc[i,0], ans])
```

```
[159]: # below table shows the first five rows of click rate for each product
       click_rate_product = pd.DataFrame(click_rate_product)
       click_rate_product = click_rate_product.rename(columns={0:"product", 1:
        ↪"click_rate"})
```

```
click_rate_product
```

[159]:

```
                       product  click_rate
0   Apple computer              0.793103
1   Apple iPad                  0.503788
2   Apple laptop                0.564516
3   BasilBasel perfume          0.649351
4   Broyhill recliner           0.539216
5   Centrum MultiVitamins       0.626556
6   Clinique moisturizer        0.805556
7   Coach purse                 0.388646
8   Cougar jeans                0.260073
9   Covergirl makeup            0.252475
10  Dell computer               0.651584
11  Dell laptop                 0.315186
12  Docker pants                0.685897
13  Ford  sedan                 0.136564
14  Gillette shaver             0.713376
15  Giorgio perfume             0.799065
16  Givenchy perfume            0.459716
17  Guess perfume               0.406977
18  Haier refrigerator          0.207547
19  Hamilton Beach blender      0.407767
20  Ikea sofa                   0.573034
21  InstantPot pressure cooker  0.500000
22  Jaguar perfume              0.473282
23  Kaai handbags               0.660000
24  LG TV                       0.480769
25  LG dryer                    0.630435
26  LG washer                   0.495327
27  Lavazza Coffee              0.564103
28  Lee jeans                   0.570776
29  Lenova laptop               0.637255
30  Maybelline lipstick         0.560976
31  Maytag dryer                0.344828
32  Maytag refrigerator         0.396552
33  Maytag washer               0.506944
34  NemoK blender               0.569672
35  NordicTrack elliptical      0.528409
36  NordicTrack rower           0.223404
37  NordicTrack treadmill       0.489712
38  Remington shaver            0.349650
39  Samsung TV                  0.731250
40  Samsung dryer               0.435897
41  Samsung washer              0.550943
42  Sony TV                     0.392857
43  Soundwave speakers          0.545894
```

```
44   Starbucks Coffee            0.275974
45   Tesla                       0.591667
46   Vitamix blender             0.507317
47   bose speakers               0.525510
48   covergirl lipstick          0.820144
```

13 Task 7

For each product, compute the click rate for each sentiment type. (10)

14 Answer 7

In order to compute the click rate for each product based on different sentiments, I groupby dataframe new by column "clickORnot" and display three columns "Sentiment", "product" and "clickORnot". Then I convert the result to a dataframe and compute click rate for each product based on different sentiments by the formula:

$$\frac{total\ number\ of\ an\ ad\ is\ clicked\ for\ a\ particular\ product\ and\ a\ sentiment\ (clickORnot\ =\ 0)}{total\ number\ of\ an\ ad\ is\ displayed\ for\ a\ particular\ product\ and\ a\ sentiment\ (clickORnot\ =\ 0\ and\ 1)}$$

One thing I notice is that for each product, there are three different sentiments and clickORnot can be either 0 or 1, so there are 6 different combinations for each product. I wrote a for loop to check if every product has 6 rows to make the following computations easier. If some products don't have six rows, I will append new rows with count equal to 0.

```
[160]: click_rate_sentiment = new.groupby(["Sentiment", "product",␣
         ↪"clickORnot"])["clickORnot"].count()
```

```
[161]: click_rate_sentiment = pd.DataFrame(click_rate_sentiment)
       click_rate_sentiment = click_rate_sentiment.rename(columns={"clickORnot":␣
         ↪"count"})
       click_rate_sentiment = click_rate_sentiment.reset_index()
       click_rate_sentiment = click_rate_sentiment.sort_values(["product",␣
         ↪"Sentiment"])
```

```
[162]: click_rate_sentiment.iloc[0:10,:]
```

```
[162]:     Sentiment         product  clickORnot  count
       0    negative  Apple computer  0           21
       1    negative  Apple computer  1           49
       96   neutral   Apple computer  0           5
       97   neutral   Apple computer  1           60
       193  positive  Apple computer  0           16
       194  positive  Apple computer  1           52
       2    negative  Apple iPad      0           56
       3    negative  Apple iPad      1           36
       98   neutral   Apple iPad      0           32
       99   neutral   Apple iPad      1           54
```

[163]:

```
# checking whether each product has six records in the dataframe␣
 ↪click_rate_sentiment
for item in (click_rate_sentiment["product"].unique()):
    if len(click_rate_sentiment.loc[click_rate_sentiment.loc[:,"product"] ==␣
 ↪item]) != 6:
        print(item)
```

```
InstantPot pressure cooker
Samsung washer
covergirl lipstick
```

Products InstantPot pressure cooker, Samsung washer, and Covergirl lipstick don't have six rows, so I will append missing rows to them.

[164]:

```
# appending missing rows
new_row1 = {"Sentiment":"negative", "product":"InstantPot pressure cooker",␣
 ↪"clickORnot":1, "count":0}
new_row2 = {"Sentiment":"negative", "product":"Samsung washer", "clickORnot":1,␣
 ↪"count":0}
new_row3 = {"Sentiment":"neutral", "product":"covergirl lipstick", "clickORnot":
 ↪0, "count":0}
new_row4 = {"Sentiment":"positive", "product":"covergirl lipstick",␣
 ↪"clickORnot":0, "count":0}
click_rate_sentiment = click_rate_sentiment.append(new_row1, ignore_index=True)
click_rate_sentiment = click_rate_sentiment.append(new_row2, ignore_index=True)
click_rate_sentiment = click_rate_sentiment.append(new_row3, ignore_index=True)
click_rate_sentiment = click_rate_sentiment.append(new_row4, ignore_index=True)
click_rate_sentiment = click_rate_sentiment.sort_values(["product",␣
 ↪"Sentiment"])
click_rate_sentiment.shape
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/378942190.py:6:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  click_rate_sentiment = click_rate_sentiment.append(new_row1,
ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/378942190.py:7:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  click_rate_sentiment = click_rate_sentiment.append(new_row2,
ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/378942190.py:8:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  click_rate_sentiment = click_rate_sentiment.append(new_row3,
ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/378942190.py:9:
FutureWarning: The frame.append method is deprecated and will be removed from
```

```
pandas in a future version. Use pandas.concat instead.
  click_rate_sentiment = click_rate_sentiment.append(new_row4,
ignore_index=True)
```

[164]:
```
(294, 4)
```

[165]:
```
# check again to make sure all the products have six rows
for item in (click_rate_sentiment["product"].unique()):
    if len(click_rate_sentiment.loc[click_rate_sentiment.loc[:,"product"] ==␣
 ↪item]) != 6:
        print(item)
```

[166]:
```
click_rate_sentiment = click_rate_sentiment.reset_index()
click_rate_sentiment = click_rate_sentiment.drop(["index"], axis=1)
```

[167]:
```
# calculating click rate for each product based on different sentiments
rate_sentiment_product = []
for i in range(0,len(click_rate_sentiment)-1,6):
    negative = click_rate_sentiment.iloc[(i+1),3] / (click_rate_sentiment.
 ↪iloc[i,3] + click_rate_sentiment.iloc[(i+1),3])
    neutral = click_rate_sentiment.iloc[(i+3),3] /  (click_rate_sentiment.
 ↪iloc[(i+2),3] + click_rate_sentiment.iloc[(i+3),3])
    positive = click_rate_sentiment.iloc[(i+5),3] / (click_rate_sentiment.
 ↪iloc[(i+4),3] + click_rate_sentiment.iloc[(i+5),3])
    rate_sentiment_product.append([click_rate_sentiment.iloc[i,1], negative,␣
 ↪neutral, positive])
rate_sentiment_product
```

[167]:
```
[['Apple computer', 0.7, 0.9230769230769231, 0.7647058823529411],
 ['Apple iPad', 0.391304347826087, 0.627906976744186, 0.5],
 ['Apple laptop', 0.15384615384615385, 0.7317073170731707, 0.7727272727272727],
 ['BasilBasel perfume',
  0.7714285714285715,
  0.864406779661017,
  0.36666666666666664],
 ['Broyhill recliner', 0.8142857142857143, 0.5625, 0.24285714285714285],
 ['Centrum MultiVitamins',
  0.8352941176470589,
  0.8148148148148148,
  0.18666666666666668],
 ['Clinique moisturizer',
  0.9027777777777778,
  0.9305555555555556,
  0.5833333333333334],
 ['Coach purse', 0.31645569620253167, 0.4533333333333333, 0.4],
 ['Cougar jeans', 0.08333333333333333, 0.3116883116883117, 0.39],
 ['Covergirl makeup',
  0.11267605633802817,
  0.38461538461538464,
  0.24528301886792453],
 ['Dell computer',
  0.8309859154929577,
  0.38235294117647056,
  0.7195121951219512],
 ['Dell laptop', 0.1452991452991453, 0.35714285714285715, 0.4528301886792453],
 ['Docker pants', 0.835820895522388, 0.8181818181818182, 0.3333333333333333],
 ['Ford  sedan', 0.013157894736842105, 0.125, 0.26582278481012656],
 ['Gillette shaver',
  0.9047619047619048,
  0.9607843137254902,
  0.13953488372093023],
 ['Giorgio perfume',
  0.6385542168674698,
  0.9696969696969697,
  0.8307692307692308],
 ['Givenchy perfume',
  0.7575757575757576,
  0.3026315789473684,
  0.34782608695652173],
 ['Guess perfume', 0.1724137931034483, 0.3064516129032258, 0.7884615384615384],
 ['Haier refrigerator',
  0.17307692307692307,
  0.16666666666666666,
  0.2830188679245283],
 ['Hamilton Beach blender',
  0.6716417910447762,
  0.35384615384615387,
  0.21621621621621623],
 ['Ikea sofa', 0.84, 0.5942028985507246, 0.3220338983050847],
 ['InstantPot pressure cooker', 0.0, 0.864406779661017, 0.7142857142857143],
 ['Jaguar perfume', 0.43478260869565216, 0.7, 0.3111111111111111],
 ['Kaai handbags',
  0.49056603773584906,
  0.9565217391304348,
  0.5128205128205128],
 ['LG TV', 0.29850746268656714, 0.5909090909090909, 0.6444444444444445],
 ['LG dryer', 0.6808510638297872, 0.7916666666666666, 0.3953488372093023],
 ['LG washer', 0.15384615384615385, 0.5166666666666667, 0.8289473684210527],
 ['Lavazza Coffee',
  0.8409090909090909,
  0.4473684210526316,
  0.34285714285714286],
 ['Lee jeans', 0.37579617834394907, 0.7890625, 0.5882352941176471],
 ['Lenova laptop', 0.75, 0.7142857142857143, 0.41935483870967744],
 ['Maybelline lipstick',
  0.8852459016393442,
  0.20987654320987653,
  0.6984126984126984],
 ['Maytag dryer',
  0.11764705882352941,
  0.7049180327868853,
  0.25675675675675674],
 ['Maytag refrigerator',
  0.05263157894736842,
  0.7222222222222222,
  0.42857142857142855],
 ['Maytag washer',
  0.9222222222222223,
  0.3617021276595745,
  0.27884615384615385],
 ['NemoK blender',
  0.9782608695652174,
  0.3258426966292135,
  0.31746031746031744],
 ['NordicTrack elliptical',
  0.49122807017543857,
  0.7377049180327869,
  0.3448275862068966],
 ['NordicTrack rower', 0.2, 0.14285714285714285, 0.2972972972972973],
 ['NordicTrack treadmill', 0.31645569620253167, 0.5842696629213483, 0.56],
 ['Remington shaver',
  0.14516129032258066,
  0.23684210526315788,
  0.7441860465116279],
 ['Samsung TV', 0.6666666666666666, 0.8235294117647058, 0.703125],
 ['Samsung dryer',
  0.27586206896551724,
  0.7169811320754716,
  0.3111111111111111],
 ['Samsung washer', 0.0, 0.6987951807228916, 0.8979591836734694],
 ['Sony TV', 0.05555555555555555, 0.6785714285714286, 0.43103448275862066],
 ['Soundwave speakers',
  0.11940298507462686,
  0.9552238805970149,
  0.5616438356164384],
 ['Starbucks Coffee', 0.3, 0.3235294117647059, 0.20754716981132076],
 ['Tesla', 0.7916666666666666, 0.9876543209876543, 0.05747126436781609],
 ['Vitamix blender',
  0.5365853658536586,
  0.3387096774193548,
  0.639344262295082],
 ['bose speakers', 0.5967741935483871, 0.463768115942029, 0.5230769230769231],
 ['covergirl lipstick', 0.4186046511627907, 0.0, 0.0]]
```

[168]:
```
# creating a dataframe containing different click rates for products based on␣
↪different sentiments
rate_sentiment_df = pd.DataFrame(rate_sentiment_product)
rate_sentiment_df = rate_sentiment_df.rename(columns={0:"product", 1:
↪"negative", 2:"neutral", 3:"positive"})
rate_sentiment_df
```

[168]:
```
                       product  negative   neutral  positive
0   Apple computer              0.700000  0.923077  0.764706
1   Apple iPad                  0.391304  0.627907  0.500000
2   Apple laptop                0.153846  0.731707  0.772727
3   BasilBasel perfume          0.771429  0.864407  0.366667
4   Broyhill recliner           0.814286  0.562500  0.242857
5   Centrum MultiVitamins       0.835294  0.814815  0.186667
6   Clinique moisturizer        0.902778  0.930556  0.583333
7   Coach purse                 0.316456  0.453333  0.400000
8   Cougar jeans                0.083333  0.311688  0.390000
9   Covergirl makeup            0.112676  0.384615  0.245283
10  Dell computer               0.830986  0.382353  0.719512
11  Dell laptop                 0.145299  0.357143  0.452830
12  Docker pants                0.835821  0.818182  0.333333
13  Ford  sedan                 0.013158  0.125000  0.265823
14  Gillette shaver             0.904762  0.960784  0.139535
15  Giorgio perfume             0.638554  0.969697  0.830769
16  Givenchy perfume            0.757576  0.302632  0.347826
17  Guess perfume               0.172414  0.306452  0.788462
18  Haier refrigerator          0.173077  0.166667  0.283019
19  Hamilton Beach blender      0.671642  0.353846  0.216216
20  Ikea sofa                   0.840000  0.594203  0.322034
21  InstantPot pressure cooker  0.000000  0.864407  0.714286
22  Jaguar perfume              0.434783  0.700000  0.311111
23  Kaai handbags               0.490566  0.956522  0.512821
24  LG TV                       0.298507  0.590909  0.644444
25  LG dryer                    0.680851  0.791667  0.395349
26  LG washer                   0.153846  0.516667  0.828947
27  Lavazza Coffee              0.840909  0.447368  0.342857
28  Lee jeans                   0.375796  0.789062  0.588235
29  Lenova laptop               0.750000  0.714286  0.419355
30  Maybelline lipstick         0.885246  0.209877  0.698413
31  Maytag dryer                0.117647  0.704918  0.256757
32  Maytag refrigerator         0.052632  0.722222  0.428571
33  Maytag washer               0.922222  0.361702  0.278846
34  NemoK blender               0.978261  0.325843  0.317460
35  NordicTrack elliptical      0.491228  0.737705  0.344828
```

```
36  NordicTrack rower        0.200000  0.142857  0.297297
37  NordicTrack treadmill    0.316456  0.584270  0.560000
38  Remington shaver         0.145161  0.236842  0.744186
39  Samsung TV               0.666667  0.823529  0.703125
40  Samsung dryer            0.275862  0.716981  0.311111
41  Samsung washer           0.000000  0.698795  0.897959
42  Sony TV                  0.055556  0.678571  0.431034
43  Soundwave speakers       0.119403  0.955224  0.561644
44  Starbucks Coffee         0.300000  0.323529  0.207547
45  Tesla                    0.791667  0.987654  0.057471
46  Vitamix blender          0.536585  0.338710  0.639344
47  bose speakers            0.596774  0.463768  0.523077
48  covergirl lipstick       0.418605  0.000000  0.000000
```

15 Task 8

For each product type, compute the click rate for it. (10)

16 Answer 8

In order to compute the click rate for each product type, I groupby dataframe new by column “clickORnot” and display two columns “product_type” and “clickORnot”. Then I convert the result to a dataframe and compute click rate for each product_type by the formula: $\frac{total\ number\ of\ an\ ad\ is\ clicked\ (clickORnot\ =\ 0)}{total\ number\ of\ an\ ad\ is\ displayed\ (clickORnot\ =\ 0\ and\ 1)}$

[169]:

```python
# group by "clickORnot"
click_rate = new.groupby(["product_type", "clickORnot"])["clickORnot"].count()
```

[170]:

```python
click_rate = pd.DataFrame(click_rate)
click_rate = click_rate.rename(columns={"clickORnot": "count"})
click_rate = click_rate.reset_index()
click_rate.head()
```

[170]:

```
  product_type  clickORnot  count
0  blender      0           328
1  blender      1           327
2  car          0           294
3  car          1           173
4  coffee       0           274
```

[171]:

```python
# check whether every product_type has two rows
for item in (click_rate["product_type"].unique()):
    if len(click_rate.loc[click_rate.loc[:,"product_type"] == item]) != 2:
        print(item)
```

[172]:
```
# calculating the click rate for each product_type
click_rate_product_type = []
for i in range(0,len(click_rate)-1,2):
    ans = click_rate.iloc[(i+1),2] / (click_rate.iloc[i,2] + click_rate.
 ↪iloc[(i+1),2])
    click_rate_product_type.append([click_rate.iloc[i,0], ans])
```

[173]:
```
# click_rate_product_type_df contains click rate for different product types
click_rate_product_type_df = pd.DataFrame(click_rate_product_type)
click_rate_product_type_df = click_rate_product_type_df.rename(columns={0:
 ↪"product_type", 1:"click_rate"})
click_rate_product_type_df
```

[173]:
```
          product_type  click_rate
0   blender             0.499237
1   car                 0.370450
2   coffee              0.355294
3   computer            0.558583
4   dryer               0.452716
5   elliptical trainer  0.528409
6   face cream          0.805556
7   furniture           0.554974
8   jeans               0.451477
9   lipstick            0.665698
10  makeup              0.252475
11  pants               0.685897
12  perfume             0.566893
13  pressure cooker     0.500000
14  refrigerator        0.287273
15  rowing machine      0.223404
16  shaver              0.540000
17  speakers            0.535980
18  tablet              0.503788
19  television          0.533058
20  treadmill           0.489712
21  vitamin             0.626556
22  washer              0.518905
23  women's purse       0.515152
```

17 Task 9

For each product type compute the click rate for each sentiment type. (10)

18 Answer 9

In order to compute the click rate for each product type based on different sentiments, I groupby dataframe new by column “clickORnot” and display three columns “Sentiment”, “product_type” and “clickORnot”. Then I convert the result to a dataframe and compute click rate for each product type based on different sentiments by the formula: $\frac{total\ number\ of\ an\ ad\ is\ clicked\ for\ a\ particular\ product\ type\ and\ a\ sentiment\ (clickORnot\ =\ 0)}{total\ number\ of\ an\ ad\ is\ displayed\ for\ a\ particular\ product\ type\ and\ a\ sentiment\ (clickORnot\ =\ 0\ and\ 1)}$

One thing I notice is that for each product type, there are three different sentiments and clickORnot can be either 0 or 1, so there are 6 different combinations for each product type. I wrote a for loop to check if every product type has 6 rows to make the following computations easier. If some product types don’t have six rows, I will append new rows with count equal to 0.

[174]:
```
click_rate_sentiment2 = new.groupby(["Sentiment", "product_type",␣
 ↪"clickORnot"])["clickORnot"].count()
```

[175]:
```
click_rate_sentiment2 = pd.DataFrame(click_rate_sentiment2)
click_rate_sentiment2 = click_rate_sentiment2.rename(columns={"clickORnot":␣
 ↪"count"})
click_rate_sentiment2 = click_rate_sentiment2.reset_index()
click_rate_sentiment2 = click_rate_sentiment2.sort_values(["product_type",␣
 ↪"Sentiment"])
```

[176]:
```
# checking whether each product type has six records in the dataframe␣
 ↪click_rate_sentiment2
for item in (click_rate_sentiment2["product_type"].unique()):
    if len(click_rate_sentiment2.loc[click_rate_sentiment2.loc[:
 ↪,"product_type"] == item]) != 6:
        print(item)
```

```
pressure cooker
```

[177]:
```
# appending missing row
new_row = {"Sentiment":"negative", "product_type":"pressure cooker",␣
 ↪"clickORnot":1, "count":0}
click_rate_sentiment2 = click_rate_sentiment2.append(new_row, ignore_index=True)
click_rate_sentiment2  = click_rate_sentiment2.sort_values(["product_type",␣
 ↪"Sentiment"])
click_rate_sentiment2.shape
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/1588479562.py:3:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  click_rate_sentiment2 = click_rate_sentiment2.append(new_row,
ignore_index=True)
```

[177]:
```
(144, 4)
```

[178]:
```
# check again to make sure all the product types have six rows
for item in (click_rate_sentiment2["product_type"].unique()):
    if len(click_rate_sentiment2.loc[click_rate_sentiment2.loc[:
 ↪,"product_type"] == item]) != 6:
        print(item)
```

[179]:

```
click_rate_sentiment2 = click_rate_sentiment2.reset_index()
click_rate_sentiment2 = click_rate_sentiment2.drop(["index"], axis=1)
```

[180]:

```
# calculating click rate for each product type based on different sentiments
rate_sentiment_ptype = []
for i in range(0,len(click_rate_sentiment2)-1,6):
    negative = click_rate_sentiment2.iloc[(i+1),3] / (click_rate_sentiment2.
  ↪iloc[i,3] + click_rate_sentiment2.iloc[(i+1),3])
    neutral = click_rate_sentiment2.iloc[(i+3),3] / (click_rate_sentiment2.
  ↪iloc[(i+2),3] + click_rate_sentiment2.iloc[(i+3),3])
    positive = click_rate_sentiment2.iloc[(i+5),3] / (click_rate_sentiment2.
  ↪iloc[(i+4),3] + click_rate_sentiment2.iloc[(i+5),3])
    rate_sentiment_ptype.append([click_rate_sentiment2.iloc[i,1], negative,␣
  ↪neutral, positive])
rate_sentiment_ptype
```

[180]:

```
[['blender', 0.7427385892116183, 0.33796296296296297, 0.3787878787878788],
 ['car', 0.3918918918918919, 0.5816993464052288, 0.1566265060240964],
 ['coffee', 0.4652777777777778, 0.35714285714285715, 0.24113475177304963],
 ['computer', 0.5013550135501355, 0.5702702702702702, 0.6049723756906077],
 ['dryer', 0.3236994219653179, 0.7345679012345679, 0.30864197530864196],
 ['elliptical trainer',
  0.49122807017543857,
  0.7377049180327869,
  0.3448275862068966],
 ['face cream', 0.9027777777777778, 0.9305555555555556, 0.5833333333333334],
 ['furniture', 0.825, 0.5789473684210527, 0.27906976744186046],
 ['jeans', 0.2648221343873518, 0.6097560975609756, 0.5098814229249012],
 ['lipstick', 0.6923076923076923, 0.5114503816793893, 0.8256880733944955],
 ['makeup', 0.11267605633802817, 0.38461538461538464, 0.24528301886792453],
 ['pants', 0.835820895522388, 0.8181818181818182, 0.3333333333333333],
 ['perfume', 0.5555555555555556, 0.6105610561056105, 0.5326460481099656],
 ['pressure cooker', 0.0, 0.864406779661017, 0.7142857142857143],
 ['refrigerator', 0.12222222222222222, 0.3888888888888889, 0.3473684210526316],
 ['rowing machine', 0.2, 0.14285714285714285, 0.2972972972972973],
 ['shaver', 0.528, 0.651685393258427, 0.4418604651162791],
 ['speakers', 0.3488372093023256, 0.7058823529411765, 0.5434782608695652],
 ['tablet', 0.391304347826087, 0.627906976744186, 0.5],
 ['television', 0.3192771084337349, 0.7019867549668874, 0.592814371257485],
 ['treadmill', 0.31645569620253167, 0.5842696629213483, 0.56],
 ['vitamin', 0.8352941176470589, 0.8148148148148148, 0.18666666666666668],
 ['washer', 0.376984126984127, 0.5189873417721519, 0.6474820143884892],
 ["women's purse",
  0.38636363636363635,
  0.6944444444444444,
  0.45751633986928103]]
```

[181]:

```
# creating a dataframe containing different click rates for product types based␣
 ↪on different sentiments
rate_sentiment_df_ptype = pd.DataFrame(rate_sentiment_ptype)
rate_sentiment_df_ptype = rate_sentiment_df_ptype.rename(columns={0:
 ↪"product_type", 1:"negative_click_rate", 2:"neutral_click_rate", 3:
 ↪"positive_click_rate"})
rate_sentiment_df_ptype
```

[181]:

```
          product_type  negative_click_rate  neutral_click_rate  \
0   blender             0.742739             0.337963
1   car                 0.391892             0.581699
2   coffee              0.465278             0.357143
3   computer            0.501355             0.570270
4   dryer               0.323699             0.734568
5   elliptical trainer  0.491228             0.737705
6   face cream          0.902778             0.930556
7   furniture           0.825000             0.578947
8   jeans               0.264822             0.609756
9   lipstick            0.692308             0.511450
10  makeup              0.112676             0.384615
11  pants               0.835821             0.818182
12  perfume             0.555556             0.610561
13  pressure cooker     0.000000             0.864407
14  refrigerator        0.122222             0.388889
15  rowing machine      0.200000             0.142857
16  shaver              0.528000             0.651685
17  speakers            0.348837             0.705882
18  tablet              0.391304             0.627907
19  television          0.319277             0.701987
20  treadmill           0.316456             0.584270
21  vitamin             0.835294             0.814815
22  washer              0.376984             0.518987
23  women's purse       0.386364             0.694444

    positive_click_rate
0   0.378788
1   0.156627
2   0.241135
3   0.604972
4   0.308642
5   0.344828
6   0.583333
7   0.279070
8   0.509881
9   0.825688
10  0.245283
11  0.333333
```

```
12   0.532646
13   0.714286
14   0.347368
15   0.297297
16   0.441860
17   0.543478
18   0.500000
19   0.592814
20   0.560000
21   0.186667
22   0.647482
23   0.457516
```

19 Task 10

Save this information you computed in 9 above in a database table. Should you save it in the products table or the product_categories table or the product_type_pubURLs table, or should you create a new table product_type_sentiment_clickrate, and save this information in this table? If you create a new table, make sure to set up all the appropriate foreign key constraints. On the other hand, if you use one of the existing tables, explain how you will avoid redundancy in your data. In either case, justify your decision. (10)

20 Answer 10

This time, I will create a new table called product_type_sentiment_clickrate to save the result from task 9. After creaing it in the database DATA1050FP, I set the primary to be product_type and there is one foriegn key constraint: a foreign key constraint from product_type_sentiment_clickrate to product_categories on product_type field.

[182]:

```
'''
for i in range(len(category)):
    category.iloc[i,3] = "{:.2%}".format(category.iloc[i,3])
    category.iloc[i,4] = "{:.2%}".format(category.iloc[i,4])
    category.iloc[i,5] = "{:.2%}".format(category.iloc[i,5])
'''
```

[182]:

```
'\nfor i in range(len(category)):\n    category.iloc[i,3] =
"{:.2%}".format(category.iloc[i,3])\n    category.iloc[i,4] =
"{:.2%}".format(category.iloc[i,4])\n    category.iloc[i,5] =
"{:.2%}".format(category.iloc[i,5])\n'
```

[183]:

```
product_type_sentiment_clickrate = rate_sentiment_df_ptype
```

[184]:

```
# creating a new table called product_type_sentiment_clickrate in the database␣
 ↪DATA1050FP
mycursor.execute("CREATE TABLE DATA1050FP.product_type_sentiment_clickrate \
    (product_type VARCHAR(100) NOT NULL, \
        negative_click_rate DECIMAL(7,6), \
        neutral_click_rate DECIMAL(7,6), \
        positive_click_rate DECIMAL(7,6), \
        PRIMARY KEY (product_type), \
        FOREIGN KEY (product_type) REFERENCES product_categories (product_type)) ")
mycursor.execute("SHOW TABLES")
for x in mycursor:
  print(x)
```

```
('product_categories',)
('product_type_sentiment_clickrate',)
('products',)
```

[185]:

```
# insert rows to the new table product_type_sentiment_clickrate
for i, row in product_type_sentiment_clickrate.iterrows():
    mycursor.execute("INSERT INTO product_type_sentiment_clickrate VALUES (%s,␣
 ↪%s, %s, %s)", tuple(row))
    mydb.commit()
```

[186]:

```
# print all rows from table product_type_sentiment_clickrate
mycursor.execute("SELECT * FROM product_type_sentiment_clickrate")
result = mycursor.fetchall()
for row in result:
    print(row)
    print("\n")
```

```
('blender', Decimal('0.742739'), Decimal('0.337963'), Decimal('0.378788'))


('car', Decimal('0.391892'), Decimal('0.581699'), Decimal('0.156627'))


('coffee', Decimal('0.465278'), Decimal('0.357143'), Decimal('0.241135'))


('computer', Decimal('0.501355'), Decimal('0.570270'), Decimal('0.604972'))


('dryer', Decimal('0.323699'), Decimal('0.734568'), Decimal('0.308642'))


('elliptical trainer', Decimal('0.491228'), Decimal('0.737705'),
Decimal('0.344828'))


('face cream', Decimal('0.902778'), Decimal('0.930556'), Decimal('0.583333'))
```

```
('furniture', Decimal('0.825000'), Decimal('0.578947'), Decimal('0.279070'))

('jeans', Decimal('0.264822'), Decimal('0.609756'), Decimal('0.509881'))

('lipstick', Decimal('0.692308'), Decimal('0.511450'), Decimal('0.825688'))

('makeup', Decimal('0.112676'), Decimal('0.384615'), Decimal('0.245283'))

('pants', Decimal('0.835821'), Decimal('0.818182'), Decimal('0.333333'))

('perfume', Decimal('0.555556'), Decimal('0.610561'), Decimal('0.532646'))

('pressure cooker', Decimal('0.000000'), Decimal('0.864407'),
Decimal('0.714286'))

('refrigerator', Decimal('0.122222'), Decimal('0.388889'), Decimal('0.347368'))

('rowing machine', Decimal('0.200000'), Decimal('0.142857'),
Decimal('0.297297'))

('shaver', Decimal('0.528000'), Decimal('0.651685'), Decimal('0.441860'))

('speakers', Decimal('0.348837'), Decimal('0.705882'), Decimal('0.543478'))

('tablet', Decimal('0.391304'), Decimal('0.627907'), Decimal('0.500000'))

('television', Decimal('0.319277'), Decimal('0.701987'), Decimal('0.592814'))

('treadmill', Decimal('0.316456'), Decimal('0.584270'), Decimal('0.560000'))

('vitamin', Decimal('0.835294'), Decimal('0.814815'), Decimal('0.186667'))

('washer', Decimal('0.376984'), Decimal('0.518987'), Decimal('0.647482'))
```

```
("women's purse", Decimal('0.386364'), Decimal('0.694444'), Decimal('0.457516'))
```

21 Task 11

Determine if the gender of the person viewing ads make a difference with regard to the click rate of ads shown in different sentiment context. That is, determine if there are any ‘significant’ differences in the correlation between the sentiment type of the ad context and clicking on the product type conditioned on gender. You can decide if any difference counts as ‘significant’. (This is not a yes or no question. Compute the different correlations.) (10)

22 Answer 11

In order to determine if there are any significant differences in the correlation between the sentiment type of the ad context and clicking on the product type conditioned on gender, I created a dataframe called question 11, which contains columns “product_type”, “Sentiment”, “male_clickrate”, “female_clickrate”, “average_clickrate”, “correlation”, and “significant”. I first create a new dataframe called new2 which are merged from dataframes new and product_type_sentiment_clickrate. Then, I groupby new2 by column “clickORnot” and display columns “Sentiment”, “product_type”, “clickORnot”,and “gender”. The formula I used to calculate “male_clickrate” and “female_clickrate” is: $\frac{total\ number\ of\ an\ ad\ is\ clicked\ for\ a\ particular\ product\ and\ a\ sentiment\ (clickORnot\ =\ 0)}{total\ number\ of\ an\ ad\ is\ displayed\ for\ a\ particular\ product\ and\ a\ sentiment\ (clickORnot\ =\ 0\ and\ 1)}$

The “average_clickrate” is calculated using formula: $\frac{male\_clickrate\ +\ female\_clickrate}{2}$

The “correlation” is calculated using a method called percentage difference and the formula is: $\frac{abs(male\_clickrate\ -\ female\_clickrate)}{average\_clickrate}$

The column “significant” is determined by comparing the “correlation” column with a threshold. The threshold is determined using a histgram based on “correlation” column and in this case, I choose 0.2. If the correlation is greater than 0.2, this row is considered significant.

[187]:
```
new2 = new.merge(product_type_sentiment_clickrate, on="product_type")
```

[188]:
```
gender_sentiment = pd.DataFrame(new2.groupby(["Sentiment", "product_type",␣
 ↪"clickORnot", "gender"])["clickORnot"].count())
gender_sentiment = gender_sentiment.rename(columns={"clickORnot": "count"})
gender_sentiment = gender_sentiment.reset_index()
```

[189]:
```
# checking whether each product type has 12 records in the dataframe␣
 ↪gender_sentiment
# we have 3 different kinds of sentiments, 2 kinds of gender, and 0 or 1 for␣
 ↪clickORnot (3*2*2 = 12)
# only product_type pressure cooker misses some rows
for item in (gender_sentiment["product_type"].unique()):
    if len(gender_sentiment.loc[gender_sentiment.loc[:,"product_type"] ==␣
 ↪item]) != 12:
        print(item)
```

```
pressure cooker
```

[190]:

```
new_row1 = {"Sentiment":"negative", "product_type":"pressure cooker",␣
 ↪"clickORnot":1, "gender": "male", "count":0}
new_row2 = {"Sentiment":"negative", "product_type":"pressure cooker",␣
 ↪"clickORnot":1, "gender": "female", "count":0}
```

[191]:

```
# append missing rows in the dataframe gender_sentiment
gender_sentiment = gender_sentiment.append(new_row1, ignore_index=True)
gender_sentiment = gender_sentiment.append(new_row2, ignore_index=True)
gender_sentiment = gender_sentiment.sort_values(["product_type", "Sentiment",␣
 ↪"gender"])
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/1629086596.py:2:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  gender_sentiment = gender_sentiment.append(new_row1, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/1629086596.py:3:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  gender_sentiment = gender_sentiment.append(new_row2, ignore_index=True)
```

[192]:

```
# check again to make sure all the product_types have 12 rows
for item in (gender_sentiment["product_type"].unique()):
    if len(gender_sentiment.loc[gender_sentiment.loc[:,"product_type"] ==␣
 ↪item]) != 12:
        print(item)
```

[193]:

```
all_sentiment = gender_sentiment.iloc[:,0].unique()
all_product_type = gender_sentiment.iloc[:,1].unique()
```

[194]:

```
question11 = gender_sentiment.iloc[:,[1,0]]
question11["male_clickrate"] = 0
question11["female_clickrate"] = 0
question11["average_clickrate"] = 0
question11["correlation"] = 0
question11["significant"] = 0
question11.shape
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/300259693.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  question11["male_clickrate"] = 0
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/300259693.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  question11["female_clickrate"] = 0
```

[194]:

```
(288, 7)
```

[195]:

```python
question11 = question11.drop_duplicates()
question11.shape
```

[195]:

```
(72, 7)
```

[196]:

```python
# calculating male and female clickrate for each product type and sentiment
for s in all_sentiment:
    for p in all_product_type:
            male_temp = gender_sentiment.loc[(gender_sentiment.loc[:
↪,"Sentiment"] == s) & (gender_sentiment.loc[:,"product_type"] == p) &␣
↪(gender_sentiment.loc[:,"gender"] == "male")]
            female_temp = gender_sentiment.loc[(gender_sentiment.loc[:
↪,"Sentiment"] == s) & (gender_sentiment.loc[:,"product_type"] == p) &␣
↪(gender_sentiment.loc[:,"gender"] == "female")]
            male_avg = male_temp.iloc[1,4] / (male_temp.iloc[0,4] + male_temp.
↪iloc[1,4])
            female_avg = female_temp.iloc[1,4] / (female_temp.iloc[0,4] +␣
↪female_temp.iloc[1,4])
            question11.loc[(question11.loc[:,"product_type"] == p) &␣
↪(question11.loc[:,"Sentiment"] == s), "male_clickrate"] = male_avg
            question11.loc[(question11.loc[:,"product_type"] == p) &␣
↪(question11.loc[:,"Sentiment"] == s), "female_clickrate"] = female_avg
```

[197]:

```python
'''
# import statistics
for i in range(0, len(question11) , 3):
    # average_click_rate = statistics.mean(question11.iloc[i,2:4])
    average_click_rate = product_type_sentiment_clickrate.
↪loc[product_type_sentiment_clickrate.loc[:,"product_type"] == question11.
↪iloc[i,0]]
    question11.iloc[i,4] = average_click_rate.iloc[0,1]
    question11.iloc[i+1,4] = average_click_rate.iloc[0,2]
    question11.iloc[i+2,4] = average_click_rate.iloc[0,3]
question11.head()
```

```
'''
```

[197]:
```
'\n# import statistics\nfor i in range(0, len(question11) , 3):\n    #
average_click_rate = statistics.mean(question11.iloc[i,2:4])\n
average_click_rate = product_type_sentiment_clickrate.loc[product_type_sentiment
_clickrate.loc[:,"product_type"] == question11.iloc[i,0]]\n
question11.iloc[i,4] = average_click_rate.iloc[0,1]\n    question11.iloc[i+1,4]
= average_click_rate.iloc[0,2]\n    question11.iloc[i+2,4] =
average_click_rate.iloc[0,3]\nquestion11.head()\n'
```

[198]:
```
# calculating average click rate and correlation
for i in range(len(question11)):
    top_v = abs(question11.iloc[i, 2] - question11.iloc[i, 3])
    bot_v = (question11.iloc[i, 2] + question11.iloc[i, 3])/2
    corr = top_v/bot_v
    question11.iloc[i, 4] = bot_v
    question11.iloc[i, 5] = corr
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/3560756371.py:5:
RuntimeWarning: invalid value encountered in double_scalars
  corr = top_v/bot_v
```

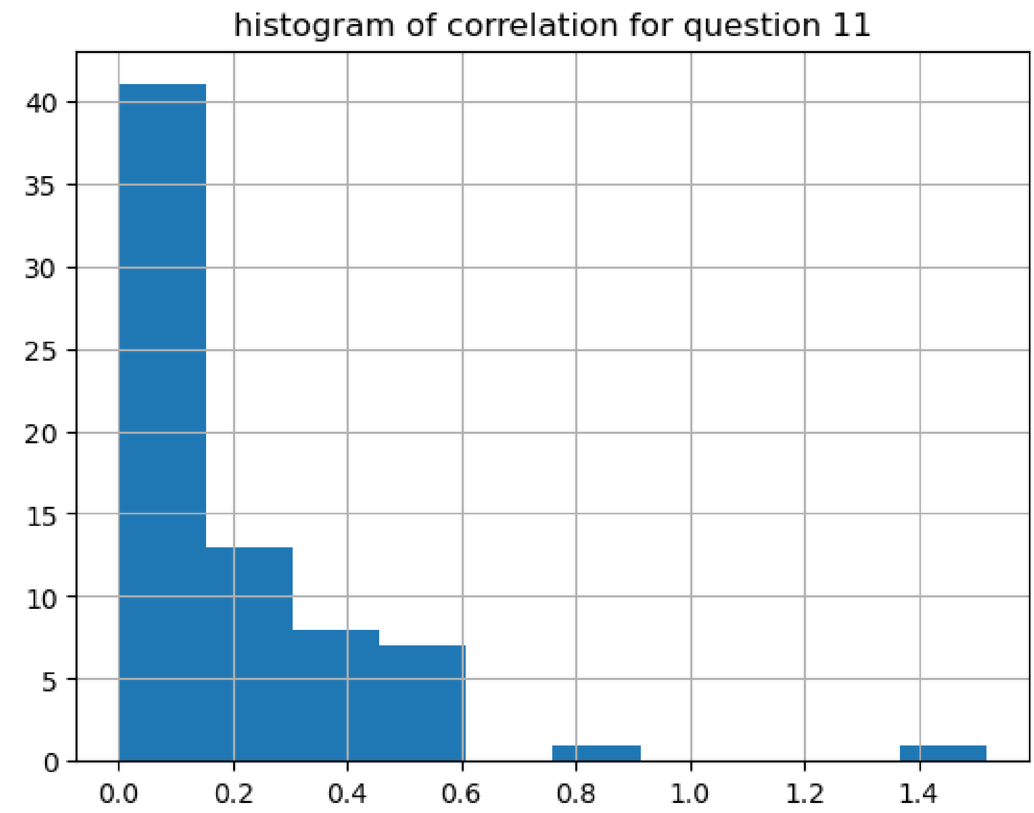
[199]:
```
"""
for i in range(len(question11)):
    temp = question11.iloc[i, 2:5]
    for l in range(2):
        if abs(temp[l] - temp[2]) > 0.05:
            question11.iloc[i, 5] = 1
"""
```

[199]:
```
'\nfor i in range(len(question11)):\n    temp = question11.iloc[i, 2:5]\n    for
l in range(2):\n        if abs(temp[l] - temp[2]) > 0.05:\n
question11.iloc[i, 5] = 1\n'
```

[200]:
```
import matplotlib.pyplot as plt

# histogram of correlation, we pick 0.2 as the threshold to determine␣
 ↪significance
question11.iloc[:,5].hist()
plt.title("histogram of correlation for question 11")
```

[200]:
```
Text(0.5, 1.0, 'histogram of correlation for question 11')
```



[201]:

```
# determine whether each row is significant
for i in range(len(question11)):
    if question11.iloc[i,5] > 0.2:
        question11.iloc[i,6] = 1
display(question11)
```

```
           product_type Sentiment  male_clickrate  female_clickrate  \
0    blender             negative  0.785714        0.695652
94   blender             neutral   0.291667        0.375000
190  blender             positive  0.380435        0.377358
4    car                 negative  0.404762        0.375000
98   car                 neutral   0.636364        0.526316
194  car                 positive  0.183908        0.126582
8    coffee              negative  0.435897        0.500000
102  coffee              neutral   0.328358        0.383562
198  coffee              positive  0.217949        0.269841
12   computer            negative  0.500000        0.502618
106  computer            neutral   0.584270        0.557292
202  computer            positive  0.614035        0.596859
16   dryer               negative  0.321839        0.325581
110  dryer               neutral   0.730769        0.738095
```

206	dryer	positive	0.372093	0.236842
20	elliptical trainer	negative	0.535714	0.448276
114	elliptical trainer	neutral	0.758621	0.718750
210	elliptical trainer	positive	0.424242	0.240000
24	face cream	negative	0.928571	0.886364
118	face cream	neutral	0.944444	0.916667
214	face cream	positive	0.625000	0.550000
28	furniture	negative	0.800000	0.846154
122	furniture	neutral	0.515625	0.637681
218	furniture	positive	0.298507	0.258065
32	jeans	negative	0.271930	0.258993
126	jeans	neutral	0.596154	0.623762
222	jeans	positive	0.539130	0.485507
36	lipstick	negative	0.725490	0.660377
130	lipstick	neutral	0.516129	0.507246
226	lipstick	positive	0.866667	0.775510
40	makeup	negative	0.103448	0.119048
134	makeup	neutral	0.487805	0.270270
230	makeup	positive	0.205882	0.315789
44	pants	negative	0.810811	0.866667
138	pants	neutral	0.800000	0.842105
234	pants	positive	0.238095	0.416667
48	perfume	negative	0.554795	0.556338
142	perfume	neutral	0.574194	0.648649
238	perfume	positive	0.544218	0.520833
52	pressure cooker	negative	0.000000	0.000000
146	pressure cooker	neutral	0.793103	0.933333
242	pressure cooker	positive	0.678571	0.742857
54	refrigerator	negative	0.026316	0.192308
150	refrigerator	neutral	0.304348	0.477273
246	refrigerator	positive	0.250000	0.406780
58	rowing machine	negative	0.151515	0.250000
154	rowing machine	neutral	0.173913	0.115385
250	rowing machine	positive	0.218750	0.357143
62	shaver	negative	0.461538	0.600000
158	shaver	neutral	0.609756	0.687500
254	shaver	positive	0.518519	0.312500
66	speakers	negative	0.402778	0.280702
162	speakers	neutral	0.696970	0.714286
258	speakers	positive	0.476190	0.600000
70	tablet	negative	0.377778	0.404255
166	tablet	neutral	0.727273	0.566038
262	tablet	positive	0.428571	0.568182
74	television	negative	0.272727	0.371795
170	television	neutral	0.653333	0.750000
266	television	positive	0.584416	0.600000
78	treadmill	negative	0.263158	0.365854
174	treadmill	neutral	0.590909	0.577778

```
270  treadmill           positive  0.625000        0.485714
82   vitamin             negative  0.800000        0.875000
178  vitamin             neutral   0.735294        0.872340
274  vitamin             positive  0.108108        0.263158
86   washer              negative  0.378788        0.375000
182  washer              neutral   0.528455        0.508772
278  washer              positive  0.653846        0.641892
90   women's purse       negative  0.402985        0.369231
186  women's purse       neutral   0.740741        0.634921
282  women's purse       positive  0.452055        0.462500

     average_clickrate  correlation  significant
0    0.740683           0.121593     0
94   0.333333           0.250000     1
190  0.378897           0.008119     0
4    0.389881           0.076336     0
98   0.581340           0.189300     0
194  0.155245           0.369260     1
8    0.467949           0.136986     0
102  0.355960           0.155083     0
198  0.243895           0.212766     1
12   0.501309           0.005222     0
106  0.570781           0.047265     0
202  0.605447           0.028370     0
16   0.323710           0.011561     0
110  0.734432           0.009975     0
206  0.304468           0.444221     1
20   0.491995           0.177722     0
114  0.738685           0.053975     0
210  0.332121           0.554745     1
24   0.907468           0.046512     0
118  0.930556           0.029851     0
214  0.587500           0.127660     0
28   0.823077           0.056075     0
122  0.576653           0.211663     1
218  0.278286           0.145329     0
32   0.265461           0.048734     0
126  0.609958           0.045263     0
222  0.512319           0.104668     0
36   0.692934           0.093967     0
130  0.511688           0.017360     0
226  0.821088           0.111019     0
40   0.111248           0.140221     0
134  0.379038           0.573913     1
230  0.260836           0.421365     1
44   0.838739           0.066595     0
138  0.821053           0.051282     0
234  0.327381           0.545455     1
```

```
48   0.555566           0.002778     0
142  0.611421           0.121774     0
238  0.532526           0.043912     0
52   0.000000          NaN           0
146  0.863218           0.162450     0
242  0.710714           0.090452     0
54   0.109312           1.518519     1
150  0.390810           0.442478     1
246  0.328390           0.477419     1
58   0.200758           0.490566     1
154  0.144649           0.404624     1
250  0.287946           0.480620     1
62   0.530769           0.260870     1
158  0.648628           0.119859     0
254  0.415509           0.495822     1
66   0.341740           0.357219     1
162  0.705628           0.024540     0
258  0.538095           0.230088     1
70   0.391017           0.067715     0
166  0.646655           0.249337     1
262  0.498377           0.280130     1
74   0.322261           0.307414     1
170  0.701667           0.137767     0
266  0.592208           0.026316     0
78   0.314506           0.326531     1
174  0.584343           0.022472     0
270  0.555357           0.250804     1
82   0.837500           0.089552     0
178  0.803817           0.170494     0
274  0.185633           0.835249     1
86   0.376894           0.010050     0
182  0.518614           0.037954     0
278  0.647869           0.018452     0
90   0.386108           0.087422     0
186  0.687831           0.153846     0
282  0.457277           0.022842     0
```

[202]:

```
# below table shows all rows that have been considered as significant
question11.loc[question11.loc[:,"significant"] == 1]
```

[202]:

```
           product_type Sentiment  male_clickrate  female_clickrate  \
94   blender              neutral   0.291667        0.375000
194  car                  positive  0.183908        0.126582
198  coffee               positive  0.217949        0.269841
206  dryer                positive  0.372093        0.236842
210  elliptical trainer   positive  0.424242        0.240000
122  furniture            neutral   0.515625        0.637681
```

```
134  makeup             neutral   0.487805        0.270270
230  makeup             positive  0.205882        0.315789
234  pants              positive  0.238095        0.416667
54   refrigerator       negative  0.026316        0.192308
150  refrigerator       neutral   0.304348        0.477273
246  refrigerator       positive  0.250000        0.406780
58   rowing machine     negative  0.151515        0.250000
154  rowing machine     neutral   0.173913        0.115385
250  rowing machine     positive  0.218750        0.357143
62   shaver             negative  0.461538        0.600000
254  shaver             positive  0.518519        0.312500
66   speakers           negative  0.402778        0.280702
258  speakers           positive  0.476190        0.600000
166  tablet             neutral   0.727273        0.566038
262  tablet             positive  0.428571        0.568182
74   television         negative  0.272727        0.371795
78   treadmill          negative  0.263158        0.365854
270  treadmill          positive  0.625000        0.485714
274  vitamin            positive  0.108108        0.263158

     average_clickrate  correlation  significant
94   0.333333           0.250000     1
194  0.155245           0.369260     1
198  0.243895           0.212766     1
206  0.304468           0.444221     1
210  0.332121           0.554745     1
122  0.576653           0.211663     1
134  0.379038           0.573913     1
230  0.260836           0.421365     1
234  0.327381           0.545455     1
54   0.109312           1.518519     1
150  0.390810           0.442478     1
246  0.328390           0.477419     1
58   0.200758           0.490566     1
154  0.144649           0.404624     1
250  0.287946           0.480620     1
62   0.530769           0.260870     1
254  0.415509           0.495822     1
66   0.341740           0.357219     1
258  0.538095           0.230088     1
166  0.646655           0.249337     1
262  0.498377           0.280130     1
74   0.322261           0.307414     1
78   0.314506           0.326531     1
270  0.555357           0.250804     1
274  0.185633           0.835249     1
```

23 Task 12

The same question as 11 above but replace gender with age-group. (10)

24 Answer 12

In order to determine if there are any significant differences in the correlation between the sentiment type of the ad context and clicking on the product type conditioned on age-group, I created a dataframe called question12, which contains columns "product_type", "Sentiment", "juvenile_clickrate", "middle-age_clickrate", "senior_clickrate", "young_clickrate", "average_clickrate", "correlation", and "significant".

I groupby new2 by column "clickORnot" and display columns "Sentiment", "product_type", "clickORnot",and "age_group". The formula I used to calculate "juvenile_clickrate", "middle-age_clickrate", "senior_clickrate", and "young_clickrate" is: $\frac{total\ number\ of\ an\ ad\ is\ clicked\ for\ a\ particular\ product\ and\ a\ sentiment\ (clickORnot = 0)}{total\ number\ of\ an\ ad\ is\ displayed\ for\ a\ particular\ product\ and\ a\ sentiment\ (clickORnot = 0\ and\ 1)}$

The "average_clickrate" is calculated using formula: $\frac{juvenile\_clickrate + middle\ age\_clickrate + senior\_clickrate + young\_}{4}$

The "correlation" is calculated using a method called percentage difference and the formula is slightly different since we have more than two age groups: $\frac{abs(max(juvenile\_clickrate, middle-age\_clickrate, senior\_clickrate, young\_clickrate) - min(juvenile\_clickrate, middle-age\_clickra}{average\_clickrate}$

The column "significant" is determined by comparing the "correlation" column with a threshold. The threshold is determined using a histgram based on "correlation" column and in this case, I choose 0.5. If the correlation is greater than 0.5, this row is considered significant.

```
[203]: age_sentiment = pd.DataFrame(new2.groupby(["Sentiment", "product_type",␣
        ↪"clickORnot", "age_group"])["clickORnot"].count())
       age_sentiment = age_sentiment.rename(columns={"clickORnot": "count"})
       age_sentiment = age_sentiment.reset_index()
```

```
[204]: # checking whether each product type has 24 records in the dataframe␣
        ↪age_sentiment
       # we have 3 different kinds of sentiments, 4 kinds of age_group, and 0 or 1 for␣
        ↪clickORnot (3*4*2 = 24)
       # product_types: face cream, pants, pressure cooker, rowing machine, and␣
        ↪vitamin misses some rows
       for item in (age_sentiment["product_type"].unique()):
           if len(age_sentiment.loc[age_sentiment.loc[:,"product_type"] == item]) !=␣
        ↪24:
               print(item)
```

```
face cream
pants
pressure cooker
rowing machine
vitamin
```

[205]:

```
new_row1 = {"Sentiment":"negative", "product_type":"pressure cooker",␣
 ↪"age_group": "juvenile", "clickORnot":1, "count":0}
new_row2 = {"Sentiment":"negative", "product_type":"pressure cooker",␣
 ↪"age_group": "middle-age", "clickORnot":1, "count":0}
new_row3 = {"Sentiment":"negative", "product_type":"pressure cooker",␣
 ↪"age_group": "senior", "clickORnot":1, "count":0}
new_row4 = {"Sentiment":"negative", "product_type":"pressure cooker",␣
 ↪"age_group": "young", "clickORnot":1, "count":0}
new_row5 = {"Sentiment":"neutral", "product_type":"face cream", "age_group":␣
 ↪"young", "clickORnot":0, "count":0}
new_row6 = {"Sentiment":"neutral", "product_type":"pants", "age_group":␣
 ↪"middle-age", "clickORnot":0, "count":0}
new_row7 = {"Sentiment":"neutral", "product_type":"rowing machine", "age_group":
 ↪ "juvenile", "clickORnot":1, "count":0}
new_row8 = {"Sentiment":"positive", "product_type":"vitamin", "age_group":␣
 ↪"juvenile", "clickORnot":1, "count":0}
```

[206]:

```
# append missing rows in the dataframe age_sentiment
age_sentiment = age_sentiment.append(new_row1, ignore_index=True)
age_sentiment = age_sentiment.append(new_row2, ignore_index=True)
age_sentiment = age_sentiment.append(new_row3, ignore_index=True)
age_sentiment = age_sentiment.append(new_row4, ignore_index=True)
age_sentiment = age_sentiment.append(new_row5, ignore_index=True)
age_sentiment = age_sentiment.append(new_row6, ignore_index=True)
age_sentiment = age_sentiment.append(new_row7, ignore_index=True)
age_sentiment = age_sentiment.append(new_row8, ignore_index=True)
age_sentiment = age_sentiment.sort_values(["product_type", "Sentiment"])
print(age_sentiment.shape)
```

```
(576, 5)
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:2:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row1, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:3:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row2, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:4:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row3, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:5:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row4, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:6:
```

```
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row5, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:7:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row6, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:8:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row7, ignore_index=True)
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/692372128.py:9:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  age_sentiment = age_sentiment.append(new_row8, ignore_index=True)
```

[207]:

```python
# check again to make sure all the product_types have 24 rows
for item in (age_sentiment["product_type"].unique()):
    if len(age_sentiment.loc[age_sentiment.loc[:,"product_type"] == item]) !=␣
 ↪24:
        print(item)
```

[208]:

```python
question12 = age_sentiment.iloc[:,[1,0]]
question12["juvenile_clickrate"] = 0
question12["middle-age_clickrate"] = 0
question12["senior_clickrate"] = 0
question12["young_clickrate"] = 0
question12["average_clickrate"] = 0
question12["correlation"] = 0
question12["significant"] = 0
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/905170972.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  question12["juvenile_clickrate"] = 0
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/905170972.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  question12["middle-age_clickrate"] = 0
```

[209]:
```
question12 = question12.drop_duplicates()
print(question12.shape)
```

```
(72, 9)
```

[210]:
```
# calculating juvenile, middle-age, senior, and young clickrate for each␣
 ↪product type and sentiment
for s in all_sentiment:
    for p in all_product_type:
            juvenile_temp = age_sentiment.loc[(age_sentiment.loc[:,"Sentiment"]␣
 ↪== s) & (age_sentiment.loc[:,"product_type"] == p) & (age_sentiment.loc[:
 ↪,"age_group"] == "juvenile")]
            middle_age_temp = age_sentiment.loc[(age_sentiment.loc[:
 ↪,"Sentiment"] == s) & (age_sentiment.loc[:,"product_type"] == p) &␣
 ↪(age_sentiment.loc[:,"age_group"] == "middle-age")]
            senior_temp = age_sentiment.loc[(age_sentiment.loc[:,"Sentiment"]␣
 ↪== s) & (age_sentiment.loc[:,"product_type"] == p) & (age_sentiment.loc[:
 ↪,"age_group"] == "senior")]
            young_temp = age_sentiment.loc[(age_sentiment.loc[:,"Sentiment"] ==␣
 ↪s) & (age_sentiment.loc[:,"product_type"] == p) & (age_sentiment.loc[:
 ↪,"age_group"] == "young")]
            juvenile_avg = juvenile_temp.iloc[1,4] / (juvenile_temp.iloc[0,4] +␣
 ↪juvenile_temp.iloc[1,4])
            middle_age_avg = middle_age_temp.iloc[1,4] / (middle_age_temp.
 ↪iloc[0,4] + middle_age_temp.iloc[1,4])
            senior_avg = senior_temp.iloc[1,4] / (senior_temp.iloc[0,4] +␣
 ↪senior_temp.iloc[1,4])
            young_avg = young_temp.iloc[1,4] / (young_temp.iloc[0,4] +␣
 ↪young_temp.iloc[1,4])
            question12.loc[(question12.loc[:,"product_type"] == p) &␣
 ↪(question12.loc[:,"Sentiment"] == s), "juvenile_clickrate"] = juvenile_avg
            question12.loc[(question12.loc[:,"product_type"] == p) &␣
 ↪(question12.loc[:,"Sentiment"] == s), "middle-age_clickrate"] =␣
 ↪middle_age_avg
            question12.loc[(question12.loc[:,"product_type"] == p) &␣
 ↪(question12.loc[:,"Sentiment"] == s), "senior_clickrate"] = senior_avg
            question12.loc[(question12.loc[:,"product_type"] == p) &␣
 ↪(question12.loc[:,"Sentiment"] == s), "young_clickrate"] = young_avg
```

[211]:
```
'''
for i in range(0, len(question12) , 3):
    # average_click_rate = statistics.mean(question12.iloc[i,2:6])
    average_click_rate = product_type_sentiment_clickrate.
 ↪loc[product_type_sentiment_clickrate.loc[:,"product_type"] == question12.
 ↪iloc[i,0]]
    question12.iloc[i,6] = average_click_rate.iloc[0,1]
    question12.iloc[i+1,6] = average_click_rate.iloc[0,2]
    question12.iloc[i+2,6] = average_click_rate.iloc[0,3]
question12.head()
'''
```

[211]:
```
'\nfor i in range(0, len(question12) , 3):\n    # average_click_rate =
statistics.mean(question12.iloc[i,2:6])\n    average_click_rate = product_type_s
entiment_clickrate.loc[product_type_sentiment_clickrate.loc[:,"product_type"] ==
question12.iloc[i,0]]\n    question12.iloc[i,6] = average_click_rate.iloc[0,1]\n
question12.iloc[i+1,6] = average_click_rate.iloc[0,2]\n
question12.iloc[i+2,6] = average_click_rate.iloc[0,3]\nquestion12.head()\n'
```

[212]:
```
# calculating average click rate and correlation
import statistics
for i in range(len(question12)):
    temp_row = question12.iloc[i,2:6]
    max_row = max(temp_row)
    min_row = min(temp_row)
    top_v = abs(max_row - min_row)
    bot_v = statistics.mean(temp_row)
    corr = top_v/bot_v
    question12.iloc[i, 6] = bot_v
    question12.iloc[i, 7] = corr
```

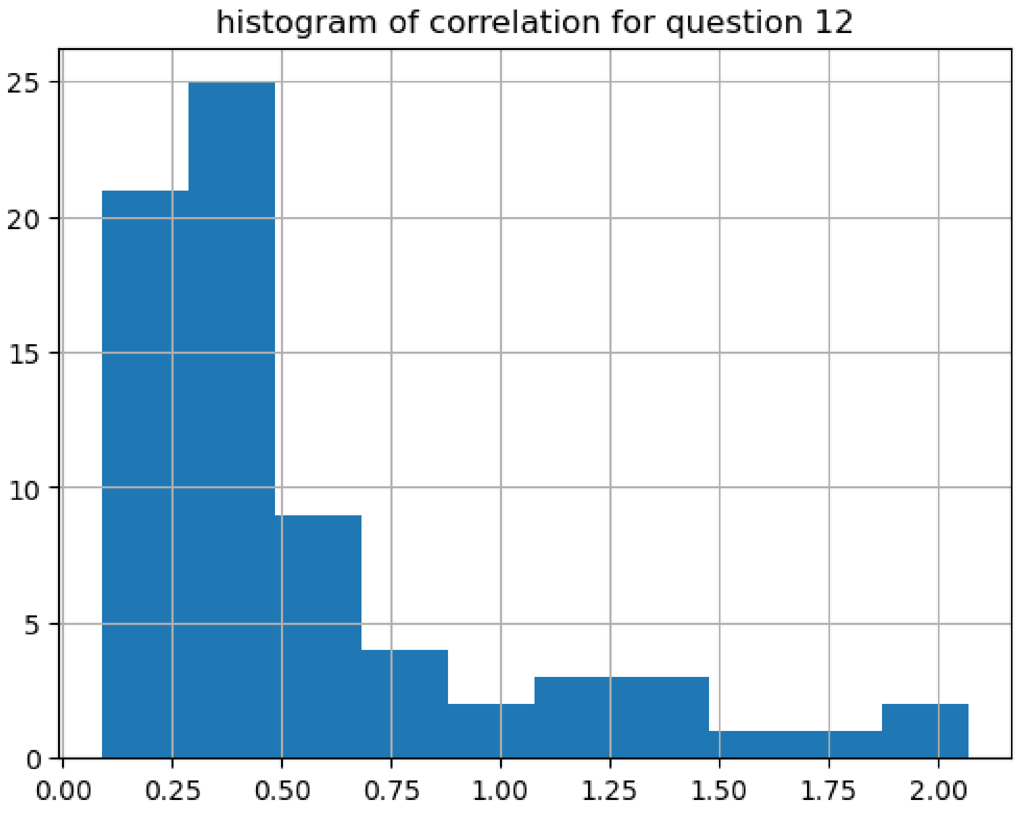
```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/2340058410.py:9:
RuntimeWarning: invalid value encountered in double_scalars
  corr = top_v/bot_v
```

[213]:
```
"""
for i in range(len(question12)):
    temp = question12.iloc[i, 2:7]
    for l in range(4):
        if abs(temp[l] - temp[4]) > 0.1:
            question12.iloc[i, 7] = 1
"""
```

[213]:
```
'\nfor i in range(len(question12)):\n    temp = question12.iloc[i, 2:7]\n    for
l in range(4):\n        if abs(temp[l] - temp[4]) > 0.1:\n
question12.iloc[i, 7] = 1\n'
```

[214]:
```
# histogram of correlation, we pick 0.5 as the threshold to determine␣
 ↪significance
question12.iloc[:,7].hist()
plt.title("histogram of correlation for question 12")
```

[214]:
```
Text(0.5, 1.0, 'histogram of correlation for question 12')
```



[215]:

```
# determine whether each row is significant
for i in range(len(question12)):
    if question12.iloc[i,7] > 0.5:
        question12.iloc[i,8] = 1
display(question12)
```

```
          product_type Sentiment  juvenile_clickrate  middle-age_clickrate  \
0    blender            negative  0.750000            0.738462
188  blender            neutral   0.313725            0.300000
377  blender            positive  0.320000            0.297872
8    car                negative  0.382353            0.404762
196  car                neutral   0.571429            0.484848
385  car                positive  0.050000            0.106383
16   coffee             negative  0.428571            0.580645
204  coffee             neutral   0.357143            0.323529
393  coffee             positive  0.048780            0.361111
24   computer           negative  0.504673            0.559524
212  computer           neutral   0.531915            0.565217
401  computer           positive  0.517647            0.557895
32   dryer              negative  0.289474            0.266667
220  dryer              neutral   0.745098            0.690476
```

```
409  dryer               positive  0.350000           0.179487
40   elliptical trainer  negative  0.500000           0.437500
228  elliptical trainer  neutral   0.842105           0.866667
417  elliptical trainer  positive  0.411765           0.277778
48   face cream          negative  0.947368           0.937500
236  face cream          neutral   0.941176           0.857143
425  face cream          positive  0.500000           0.687500
56   furniture           negative  0.809524           0.870968
243  furniture           neutral   0.444444           0.513514
433  furniture           positive  0.121212           0.343750
64   jeans               negative  0.333333           0.184615
251  jeans               neutral   0.580645           0.577778
441  jeans               positive  0.428571           0.441176
72   lipstick            negative  0.695652           0.608696
259  lipstick            neutral   0.425000           0.441176
449  lipstick            positive  0.692308           0.807692
80   makeup              negative  0.100000           0.200000
267  makeup              neutral   0.285714           0.227273
457  makeup              positive  0.090909           0.133333
88   pants               negative  0.692308           0.904762
275  pants               neutral   0.636364           0.000000
465  pants               positive  0.166667           0.307692
96   perfume             negative  0.525641           0.564706
282  perfume             neutral   0.556818           0.540984
473  perfume             positive  0.459016           0.512500
104  pressure cooker     negative  0.000000           0.000000
290  pressure cooker     neutral   0.900000           0.909091
481  pressure cooker     positive  0.750000           0.565217
108  refrigerator        negative  0.166667           0.117647
298  refrigerator        neutral   0.300000           0.272727
489  refrigerator        positive  0.392857           0.333333
116  rowing machine      negative  0.190476           0.050000
306  rowing machine      neutral   0.000000           0.176471
497  rowing machine      positive  0.050000           0.409091
124  shaver              negative  0.560976           0.551724
313  shaver              neutral   0.695652           0.535714
505  shaver              positive  0.294118           0.368421
132  speakers            negative  0.333333           0.466667
321  speakers            neutral   0.692308           0.666667
513  speakers            positive  0.388889           0.370370
140  tablet              negative  0.333333           0.347826
329  tablet              neutral   0.523810           0.750000
521  tablet              positive  0.473684           0.400000
148  television          negative  0.405405           0.297297
337  television          neutral   0.656250           0.763158
529  television          positive  0.472222           0.488372
156  treadmill           negative  0.388889           0.291667
345  treadmill           neutral   0.520000           0.650000
```

```
537  treadmill           positive  0.411765            0.722222
164  vitamin             negative  0.900000            0.750000
353  vitamin             neutral   0.857143            0.777778
545  vitamin             positive  0.000000            0.125000
172  washer              negative  0.430769            0.269841
361  washer              neutral   0.466667            0.431034
552  washer              positive  0.626667            0.549296
180  women's purse       negative  0.571429            0.424242
369  women's purse       neutral   0.714286            0.483871
560  women's purse       positive  0.500000            0.358974

     senior_clickrate  young_clickrate  average_clickrate  correlation  \
0    0.672727          0.803279         0.741117           0.176155
188  0.418182          0.316667         0.337143           0.350539
377  0.428571          0.461538         0.376996           0.434133
8    0.416667          0.361111         0.391223           0.142005
196  0.720930          0.523810         0.575254           0.410396
385  0.322581          0.187500         0.166616           1.635982
16   0.428571          0.448276         0.471516           0.322521
204  0.428571          0.333333         0.360644           0.291262
393  0.303030          0.290323         0.250811           1.245282
24   0.476190          0.468085         0.502118           0.182106
212  0.602410          0.584158         0.570925           0.123475
401  0.693182          0.648936         0.604415           0.290421
32   0.282609          0.454545         0.323324           0.581086
220  0.750000          0.757576         0.735787           0.091194
409  0.350000          0.348837         0.307081           0.555270
40   0.545455          0.500000         0.495739           0.217765
228  0.466667          0.750000         0.731360           0.546927
417  0.333333          0.363636         0.346628           0.386544
48   0.823529          0.900000         0.902099           0.137279
236  0.928571          0.000000         0.681723           1.380586
425  0.588235          0.571429         0.586791           0.319535
56   0.820513          0.793103         0.823527           0.094550
243  0.666667          0.666667         0.572823           0.387942
433  0.222222          0.405405         0.273147           1.040439
64   0.285714          0.262295         0.266490           0.558063
251  0.632653          0.653061         0.611034           0.123207
441  0.606557          0.558824         0.508782           0.349827
72   0.807692          0.656250         0.692073           0.287537
259  0.678571          0.551724         0.524118           0.483806
449  0.923077          0.870968         0.823511           0.280226
80   0.105263          0.058824         0.116022           1.216811
267  0.545455          0.450000         0.377110           0.843737
457  0.307692          0.428571         0.240127           1.406185
88   0.812500          0.882353         0.822981           0.258152
275  0.818182          0.777778         0.558081           1.466063
465  0.428571          0.333333         0.309066           0.847407
```

```
96   0.615385          0.516667         0.555600           0.177678
282  0.646341          0.694444         0.609647           0.251721
473  0.621951          0.514706         0.527043           0.309149
104  0.000000          0.000000         0.000000          NaN
290  0.833333          0.812500         0.863731           0.111830
481  0.764706          0.909091         0.747254           0.460183
108  0.166667          0.040000         0.122745           1.031949
298  0.473684          0.482759         0.382293           0.549400
489  0.176471          0.434783         0.334361           0.772554
116  0.333333          0.333333         0.226786           1.249344
306  0.250000          0.142857         0.142332           1.756458
497  0.111111          0.714286         0.321122           2.068640
124  0.370370          0.607143         0.522553           0.453107
313  0.684211          0.736842         0.663105           0.303312
505  0.560000          0.480000         0.425635           0.624673
132  0.360000          0.263158         0.355789           0.571992
321  0.680000          0.755556         0.698632           0.127233
513  0.684211          0.675676         0.529786           0.592390
140  0.433333          0.428571         0.385766           0.259224
329  0.578947          0.653846         0.626651           0.360951
521  0.571429          0.500000         0.486278           0.352532
148  0.311111          0.276596         0.322602           0.399283
337  0.711111          0.666667         0.699296           0.152879
529  0.720930          0.666667         0.587048           0.423659
156  0.272727          0.333333         0.321654           0.361138
345  0.600000          0.578947         0.587237           0.221376
537  0.550000          0.550000         0.558497           0.555881
164  0.833333          0.857143         0.835119           0.179615
353  0.789474          0.826087         0.812620           0.097666
545  0.210526          0.347826         0.170838           2.035998
172  0.388060          0.421053         0.377431           0.426377
361  0.610169          0.566667         0.518634           0.345398
552  0.617647          0.812500         0.651527           0.403980
180  0.264706          0.324324         0.396175           0.774210
369  0.782609          0.750000         0.682691           0.437588
560  0.583333          0.423077         0.466346           0.481100

     significant
0    0
188  0
377  0
8    0
196  0
385  1
16   0
204  0
393  1
24   0
```

```
212  0
401  0
32   1
220  0
409  1
40   0
228  1
417  0
48   0
236  1
425  0
56   0
243  0
433  1
64   1
251  0
441  0
72   0
259  0
449  0
80   1
267  1
457  1
88   0
275  1
465  1
96   0
282  0
473  0
104  0
290  0
481  0
108  1
298  1
489  1
116  1
306  1
497  1
124  0
313  0
505  1
132  1
321  0
513  1
140  0
329  0
521  0
148  0
```

```
337    0
529    0
156    0
345    0
537    1
164    0
353    0
545    1
172    0
361    0
552    0
180    1
369    0
560    0
```

[216]:
```
# below table shows all rows that have been considered as significant
question12.loc[question12.loc[:,"significant"] == 1]
```

[216]:
```
           product_type Sentiment  juvenile_clickrate  middle-age_clickrate  \
385                 car  positive            0.050000              0.106383
393              coffee  positive            0.048780              0.361111
32                dryer  negative            0.289474              0.266667
409               dryer  positive            0.350000              0.179487
228  elliptical trainer   neutral            0.842105              0.866667
236          face cream   neutral            0.941176              0.857143
433           furniture  positive            0.121212              0.343750
64                jeans  negative            0.333333              0.184615
80               makeup  negative            0.100000              0.200000
267              makeup   neutral            0.285714              0.227273
457              makeup  positive            0.090909              0.133333
275               pants   neutral            0.636364              0.000000
465               pants  positive            0.166667              0.307692
108        refrigerator  negative            0.166667              0.117647
298        refrigerator   neutral            0.300000              0.272727
489        refrigerator  positive            0.392857              0.333333
116       rowing machine  negative           0.190476              0.050000
306       rowing machine   neutral           0.000000              0.176471
497       rowing machine  positive           0.050000              0.409091
505              shaver  positive            0.294118              0.368421
132            speakers  negative            0.333333              0.466667
513            speakers  positive            0.388889              0.370370
537           treadmill  positive            0.411765              0.722222
545             vitamin  positive            0.000000              0.125000
180        women's purse  negative           0.571429              0.424242

     senior_clickrate  young_clickrate  average_clickrate  correlation  \
385          0.322581         0.187500           0.166616     1.635982
```

```
393  0.303030          0.290323         0.250811           1.245282
32   0.282609          0.454545         0.323324           0.581086
409  0.350000          0.348837         0.307081           0.555270
228  0.466667          0.750000         0.731360           0.546927
236  0.928571          0.000000         0.681723           1.380586
433  0.222222          0.405405         0.273147           1.040439
64   0.285714          0.262295         0.266490           0.558063
80   0.105263          0.058824         0.116022           1.216811
267  0.545455          0.450000         0.377110           0.843737
457  0.307692          0.428571         0.240127           1.406185
275  0.818182          0.777778         0.558081           1.466063
465  0.428571          0.333333         0.309066           0.847407
108  0.166667          0.040000         0.122745           1.031949
298  0.473684          0.482759         0.382293           0.549400
489  0.176471          0.434783         0.334361           0.772554
116  0.333333          0.333333         0.226786           1.249344
306  0.250000          0.142857         0.142332           1.756458
497  0.111111          0.714286         0.321122           2.068640
505  0.560000          0.480000         0.425635           0.624673
132  0.360000          0.263158         0.355789           0.571992
513  0.684211          0.675676         0.529786           0.592390
537  0.550000          0.550000         0.558497           0.555881
545  0.210526          0.347826         0.170838           2.035998
180  0.264706          0.324324         0.396175           0.774210

     significant
385  1
393  1
32   1
409  1
228  1
236  1
433  1
64   1
80   1
267  1
457  1
275  1
465  1
108  1
298  1
489  1
116  1
306  1
497  1
505  1
132  1
```

```
513   1
537   1
545   1
180   1
```

25 Task 13

Based on your results make your recommendations. These should be in the form: a. Based on our analysis (give details of your analysis), ads for such and such product are most likely to produce clicks in such and sentiment context (or state that we see no correlation between click rate of an ad for a product and the sentiment context of the ad) b. Based on our analysis (with details), ads for such and such product are most likely to produce clicks in such and sentiment context by viewers of such and such gender (or state that we see no correlation between click rate of an ad for a product and the sentiment context of the ad and the gender of the viewer). c. Based on our analysis (with details), ads for such and such product are most likely to produce clicks in such and sentiment context by viewers of such and such age-group (or state that we see no correlation between click rate of an ad for a product and the sentiment context of the ad and the age-group of the viewer). (15)

26 Answer 13

[217]:

```
# determining top 10 products/product_types to recommend (13a)
# .max(axis=1) to find the row-wise max
# .argsort() to return the integer indices that would sort the Series values
# .loc to arrange the rows in the desired sequence
product_type_sentiment_clickrate_max = product_type_sentiment_clickrate.
  ↪iloc[product_type_sentiment_clickrate.max(axis=1).argsort()[::-1],:]
product_type_sentiment_clickrate_max.iloc[0:10]
```

```
/var/folders/4h/dwdjsw5n1ln0ngs7nflp8q2h0000gn/T/ipykernel_7452/943101890.py:5:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  product_type_sentiment_clickrate_max = product_type_sentiment_clickrate.iloc[p
roduct_type_sentiment_clickrate.max(axis=1).argsort()[::-1],:]
```

[217]:

```
          product_type  negative_click_rate  neutral_click_rate  \
6   face cream          0.902778             0.930556
13  pressure cooker     0.000000             0.864407
11  pants               0.835821             0.818182
21  vitamin             0.835294             0.814815
9   lipstick            0.692308             0.511450
7   furniture           0.825000             0.578947
0   blender             0.742739             0.337963
5   elliptical trainer  0.491228             0.737705
4   dryer               0.323699             0.734568
17  speakers            0.348837             0.705882
```

```
    positive_click_rate
6   0.583333
13  0.714286
11  0.333333
21  0.186667
9   0.825688
7   0.279070
0   0.378788
5   0.344828
4   0.308642
17  0.543478
```

[218]:

```python
# finding products for each product_type
print(products.loc[products.loc[:,"product_type"] == "face cream"])
print(products.loc[products.loc[:,"product_type"] == "pressure cooker"])
print(products.loc[products.loc[:,"product_type"] == "pants"])
print(products.loc[products.loc[:,"product_type"] == "vitamin"])
print(products.loc[products.loc[:,"product_type"] == "lipstick"])
print(products.loc[products.loc[:,"product_type"] == "furniture"])
print(products.loc[products.loc[:,"product_type"] == "blender"])
print(products.loc[products.loc[:,"product_type"] == "elliptical trainer"])
print(products.loc[products.loc[:,"product_type"] == "dryer"])
print(products.loc[products.loc[:,"product_type"] == "speakers"])
```

```
                 product                       product_URL product_type
10  Clinique moisturizer  https://clinique.com/moisturizers  face cream
                      product                     product_URL     product_type
2  InstantPot pressure cooker  https://InstantPot.com/cookers  pressure cooker
         product               product_URL product_type
17  Docker pants  https://docker.com/pants  pants
                 product                 product_URL product_type
7  Centrum MultiVitamins  https://centrum.com/vitamins  vitamin
                product                     product_URL product_type
20  Maybelline lipstick  http://maybelline.com/lipstick  lipstick
37  covergirl lipstick   https://covergirl.co/lipsticks  lipstick
              product                     product_URL product_type
23  Ikea sofa          https://Ikea.com/sofas          furniture
24  Broyhill recliner  https://broyhill.com/recliners  furniture
                  product                     product_URL product_type
0  Vitamix blender         https://vitamix.com/blenders    blender
3  NemoK blender           http://nemoK.co/blenders        blender
4  Hamilton Beach blender  https://HamiltonBeach/blenders  blender
                   product                     product_URL        product_type
11  NordicTrack elliptical  https://NordicTrack/elliptical  elliptical trainer
          product                product_URL product_type
27  Maytag dryer   https://maytag.com/dryers   dryer
30  LG dryer       https://lg.com/dryers       dryer
```

```
35  Samsung dryer  https://samsung.com/dryers  dryer
               product                    product_URL product_type
41  Soundwave speakers  https://soundwave.ai/speakers  speakers
42  bose speakers       https://bose.com/speakers      speakers
```

Based on my analysis, ads for Clinique moisturizer (face cream) are most likely to produce clicks in neutral textual context in which an ad was displayed to a viewer with a click rate of 93.06%.

Ads for InstantPot pressure cooker (pressure cooker) are most likely to produce clicks in neutral textual context in which an ad was displayed to a viewer with a click rate of 86.44%.

Ads for Docker pants (pants) are most likely to produce clicks in negative textual context in which an ad was displayed to a viewer with a click rate of 83.58%.

Ads for Centrum MultiVitamins (vitamins) are most likely to produce clicks in negative textual context in which an ad was displayed to a viewer with a click rate of 83.53%.

Ads for Maybelline lipstick (lipstick) are most likely to produce clicks in positive textual context in which an ad was displayed to a viewer with a click rate of 82.57%.

Ads for Ikea sofa (furniture) are most likely to produce clicks in negative textual context in which an ad was displayed to a viewer with a click rate of 82.5%.

Ads for Vitamix blender (blender) are most likely to produce clicks in negative textual context in which an ad was displayed to a viewer with a click rate of 74.27%.

Ads for NordicTrack elliptical (elliptical trainer) are most likely to produce clicks in neutral textual context in which an ad was displayed to a viewer with a click rate of 73.77%.

Ads for Maytag dryer (dryer) are most likely to produce clicks in neutral textual context in which an ad was displayed to a viewer with a click rate of 73.46%.

Ads for Soundwave speakers (speakers) are most likely to produce clicks in neutral textual context in which an ad was displayed to a viewer with a click rate of 70.59%.

[219]:
```
# determining top 10 products/product_types to recommend (13b)
question11_max = question11.iloc[question11.iloc[:,2:4].max(axis=1).argsort()[::
 ↪-1],:]
question11_max.iloc[0:20]
```

[219]:
```
           product_type Sentiment  male_clickrate  female_clickrate  \
118  face cream          neutral   0.944444        0.916667
146  pressure cooker     neutral   0.793103        0.933333
24   face cream          negative  0.928571        0.886364
82   vitamin             negative  0.800000        0.875000
178  vitamin             neutral   0.735294        0.872340
226  lipstick            positive  0.866667        0.775510
44   pants               negative  0.810811        0.866667
28   furniture           negative  0.800000        0.846154
138  pants               neutral   0.800000        0.842105
0    blender             negative  0.785714        0.695652
114  elliptical trainer  neutral   0.758621        0.718750
```

```
170  television          neutral   0.653333        0.750000
242  pressure cooker     positive  0.678571        0.742857
186  women's purse       neutral   0.740741        0.634921
110  dryer               neutral   0.730769        0.738095
166  tablet              neutral   0.727273        0.566038
36   lipstick            negative  0.725490        0.660377
162  speakers            neutral   0.696970        0.714286
158  shaver              neutral   0.609756        0.687500
278  washer              positive  0.653846        0.641892

     average_clickrate  correlation  significant
118  0.930556           0.029851     0
146  0.863218           0.162450     0
24   0.907468           0.046512     0
82   0.837500           0.089552     0
178  0.803817           0.170494     0
226  0.821088           0.111019     0
44   0.838739           0.066595     0
28   0.823077           0.056075     0
138  0.821053           0.051282     0
0    0.740683           0.121593     0
114  0.738685           0.053975     0
170  0.701667           0.137767     0
242  0.710714           0.090452     0
186  0.687831           0.153846     0
110  0.734432           0.009975     0
166  0.646655           0.249337     1
36   0.692934           0.093967     0
162  0.705628           0.024540     0
158  0.648628           0.119859     0
278  0.647869           0.018452     0
```

[220]:

```python
# finding products for each product_type
print(products.loc[products.loc[:,"product_type"] == "face cream"])
print(products.loc[products.loc[:,"product_type"] == "pressure cooker"])
print(products.loc[products.loc[:,"product_type"] == "vitamin"])
print(products.loc[products.loc[:,"product_type"] == "lipstick"])
print(products.loc[products.loc[:,"product_type"] == "pants"])
print(products.loc[products.loc[:,"product_type"] == "furniture"])
print(products.loc[products.loc[:,"product_type"] == "blender"])
print(products.loc[products.loc[:,"product_type"] == "elliptical trainer"])
print(products.loc[products.loc[:,"product_type"] == "television"])
print(products.loc[products.loc[:,"product_type"] == "women's purse"])
```

```
                 product                        product_URL product_type
10  Clinique moisturizer  https://clinique.com/moisturizers   face cream
                      product                    product_URL     product_type
2  InstantPot pressure cooker  https://InstantPot.com/cookers  pressure cooker
```

```
                 product                 product_URL product_type
7  Centrum MultiVitamins  https://centrum.com/vitamins  vitamin
                product                     product_URL product_type
20  Maybelline lipstick  http://maybelline.com/lipstick  lipstick
37  covergirl lipstick   https://covergirl.co/lipsticks  lipstick
         product               product_URL product_type
17  Docker pants  https://docker.com/pants  pants
              product                     product_URL product_type
23  Ikea sofa          https://Ikea.com/sofas          furniture
24  Broyhill recliner  https://broyhill.com/recliners  furniture
                  product                     product_URL product_type
0  Vitamix blender         https://vitamix.com/blenders    blender
3  NemoK blender           http://nemoK.co/blenders        blender
4  Hamilton Beach blender  https://HamiltonBeach/blenders  blender
                   product                     product_URL        product_type
11  NordicTrack elliptical  https://NordicTrack/elliptical  elliptical trainer
       product                      product_URL product_type
19  LG TV       https://lg.com/tvs               television
32  Sony TV     https://sony.com/televisions     television
33  Samsung TV  https://samsung.com/televisions  television
          product                product_URL   product_type
18  Coach purse    https://coach.com/purses   women's purse
25  Kaai handbags  https://kaai.com/handbags  women's purse
```

Based on my analysis, ads for Clinique moisturizer (face cream) are most likely to produce clicks in neutral textual context by male with a click rate of 94.44%.

Ads for InstantPot pressure cooker (pressure cooker) are most likely to produce clicks in neutral textual context by female with a click rate of 93.33%.

Ads for Centrum MultiVitamins (vitamin) are most likely to produce clicks in negative textual context by female with a click rate of 87.5%.

Ads for Maybelline lipstick (lipstick) are most likely to produce clicks in positive textual context by male with a click rate of 86.67%.

Ads for Docker pants (pants) are most likely to produce clicks in negative textual context by female with a click rate of 84.62%.

Ads for Ikea sofa (furniture) are most likely to produce clicks in negative textual context by female with a click rate of 86.67%.

Ads for Vitamix blender (blender) are most likely to produce clicks in negative textual context by male with a click rate of 78.57%.

Ads for NordicTrack elliptical (elliptical trainer) are most likely to produce clicks in neutral textual context by male with a click rate of 75.86%.

Ads for LG TV (television) are most likely to produce clicks in neutral textual context by female with a click rate of 75.00%.

Ads for Coach purse (women's purse) are most likely to produce clicks in neutral textual context by male with a click rate of 74.07%.

[221]:
```
# determining top 10 products/product_types to recommend (13c)
question12_max = question12.iloc[question12.iloc[:,2:6].max(axis=1).argsort()[::
  ↪-1],:]
question12_max.iloc[0:20]
```

[221]:
```
          product_type Sentiment  juvenile_clickrate  middle-age_clickrate  \
48   face cream         negative  0.947368            0.937500
236  face cream         neutral   0.941176            0.857143
449  lipstick           positive  0.692308            0.807692
290  pressure cooker    neutral   0.900000            0.909091
481  pressure cooker    positive  0.750000            0.565217
88   pants              negative  0.692308            0.904762
164  vitamin            negative  0.900000            0.750000
56   furniture          negative  0.809524            0.870968
228  elliptical trainer neutral   0.842105            0.866667
353  vitamin            neutral   0.857143            0.777778
275  pants              neutral   0.636364            0.000000
552  washer             positive  0.626667            0.549296
72   lipstick           negative  0.695652            0.608696
0    blender            negative  0.750000            0.738462
369  women's purse      neutral   0.714286            0.483871
337  television         neutral   0.656250            0.763158
220  dryer              neutral   0.745098            0.690476
321  speakers           neutral   0.692308            0.666667
329  tablet             neutral   0.523810            0.750000
313  shaver             neutral   0.695652            0.535714

     senior_clickrate  young_clickrate  average_clickrate  correlation  \
48   0.823529          0.900000         0.902099           0.137279
236  0.928571          0.000000         0.681723           1.380586
449  0.923077          0.870968         0.823511           0.280226
290  0.833333          0.812500         0.863731           0.111830
481  0.764706          0.909091         0.747254           0.460183
88   0.812500          0.882353         0.822981           0.258152
164  0.833333          0.857143         0.835119           0.179615
56   0.820513          0.793103         0.823527           0.094550
228  0.466667          0.750000         0.731360           0.546927
353  0.789474          0.826087         0.812620           0.097666
275  0.818182          0.777778         0.558081           1.466063
552  0.617647          0.812500         0.651527           0.403980
72   0.807692          0.656250         0.692073           0.287537
0    0.672727          0.803279         0.741117           0.176155
369  0.782609          0.750000         0.682691           0.437588
337  0.711111          0.666667         0.699296           0.152879
220  0.750000          0.757576         0.735787           0.091194
321  0.680000          0.755556         0.698632           0.127233
329  0.578947          0.653846         0.626651           0.360951
```

```
313  0.684211          0.736842         0.663105           0.303312

     significant
48   0
236  1
449  0
290  0
481  0
88   0
164  0
56   0
228  1
353  0
275  1
552  0
72   0
0    0
369  0
337  0
220  0
321  0
329  0
313  0
```

[222]:

```
# finding products for each product_type
print(products.loc[products.loc[:,"product_type"] == "face cream"])
print(products.loc[products.loc[:,"product_type"] == "lipstick"])
print(products.loc[products.loc[:,"product_type"] == "pressure cooker"])
print(products.loc[products.loc[:,"product_type"] == "pants"])
print(products.loc[products.loc[:,"product_type"] == "vitamin"])
print(products.loc[products.loc[:,"product_type"] == "furniture"])
print(products.loc[products.loc[:,"product_type"] == "elliptical trainer"])
print(products.loc[products.loc[:,"product_type"] == "washer"])
print(products.loc[products.loc[:,"product_type"] == "blender"])
print(products.loc[products.loc[:,"product_type"] == "women's purse"])
```

```
                 product                        product_URL product_type
10  Clinique moisturizer  https://clinique.com/moisturizers  face cream
                product                    product_URL product_type
20  Maybelline lipstick  http://maybelline.com/lipstick  lipstick
37  covergirl lipstick   https://covergirl.co/lipsticks  lipstick
                      product                    product_URL     product_type
2  InstantPot pressure cooker  https://InstantPot.com/cookers  pressure cooker
          product               product_URL product_type
17  Docker pants  https://docker.com/pants  pants
                 product                       product_URL product_type
7  Centrum MultiVitamins  https://centrum.com/vitamins  vitamin
               product                        product_URL product_type
```

```
23  Ikea sofa          https://Ikea.com/sofas          furniture
24  Broyhill recliner  https://broyhill.com/recliners  furniture
                   product                     product_URL        product_type
11  NordicTrack elliptical  https://NordicTrack/elliptical  elliptical trainer
           product                 product_URL product_type
26  Maytag washer   https://maytag.com/washers   washer
29  LG washer       https://lg.com/washers       washer
34  Samsung washer  https://samsung.com/washers  washer
                  product                     product_URL product_type
0  Vitamix blender         https://vitamix.com/blenders    blender
3  NemoK blender           http://nemoK.co/blenders        blender
4  Hamilton Beach blender  https://HamiltonBeach/blenders  blender
          product                product_URL   product_type
18  Coach purse    https://coach.com/purses   women's purse
25  Kaai handbags  https://kaai.com/handbags  women's purse
```

Based on my analysis, ads for Clinique moisturizer (face cream) are most likely to produce clicks in negative textual context by juvenile with a click rate of 94.74%.

Ads for Maybelline lipstick (lipstick) are most likely to produce clicks in positive textual context by senior with a click rate of 92.31%.

Ads for InstantPot pressure cooker (pressure cooker) are most likely to produce clicks in neutral textual context by middle-age persons with a click rate of 90.91%.

Ads for Docker pants (pants) are most likely to produce clicks in negative textual context by middle-age persons with a click rate of 90.48%.

Ads for Centrum MultiVitamins (vitamin) are most likely to produce clicks in negative textual context by juvenile with a click rate of 90.00%.

Ads for Ikea sofa (furniture) are most likely to produce clicks in negative textual context by middle-age persons with a click rate of 87.10%.

Ads for NordicTrack elliptical (elliptical trainer) are most likely to produce clicks in neutral textual context by middle-age persons with a click rate of 86.67%.

Ads for Maytag washer (washer) are most likely to produce clicks in positive textual context by young people with a click rate of 81.25%.

Ads for Vitamix blender (blender) are most likely to produce clicks in negative textual context by young people with a click rate of 80.33%.

Ads for Coach purse (women's purse) are most likely to produce clicks in neutral textual context by senior with a click rate of 78.26%.

27 Conclusion

When making recommendations regarding ads for such and such product are most likely to produce clicks in such and sentiment context, ads for such and such product are most likely to produce clicks in such and sentiment context by viewers of such and such gender, and ads for such and such product are most likely to produce clicks in such and sentiment context by viewers of such and such age-group, we assume that the analysis of click rate regarding product type can be direcyly apply to individual product. Moreover, if there are multiple products in that product type, we randomly choose one from them. In order to make my methodology scalable, I use python script instead of SQL query to calculate different kinds of click rates. Then, I create tables in the database, making sure all primary keys and foreign key constraints are correct. One main limitation I have within the study is that when choosing the threshold to determine whether each row (in task 11 and 12) is significant, I don't perform any statistical testing. I choose the threshold based on histogram of the correlation column. If one wants to be rigorous, they need to perform statistical testings to determine if the record is significant. Moreover, in the log file, I only have 10000 rows of data for 50 products, can potentially gather more information to conduct a more rigorous analysis. Lastly, the whole study doesn't consider interaction effects when making recommendations, in the future, one should measure how gender and age group together affect the click rate for different products based on different sentiments.