Instacart EDA 3_2 Assignment [ANSWER]

In this assignment you will create variables that will describe each customer.

First load the requested packages and data files for this assignment:

In [1]:
```
#load packages
import pandas as pd                    # for data manipulation
import matplotlib.pyplot as plt        # for plotting
import seaborn as sns                  # an extension of matplotlib for statistical gr
aphics

#load data
orders = pd.read_csv('../input/orders.csv')
products = pd.read_csv('../input/products.csv')
order_products_prior = pd.read_csv('../input/order_products__prior.csv')
```

Create variables that describe each customer

Question 1: Create a DataFrame that has the orders and the products purchased

1. Create a DataFrame that contains information for both the orders & order_products_prior DataFrame. Use a inner join and save it as **prd** DataFrame.

Question 2: Get the average days since a prior order for each user.

1. Get the average days since a prior order for each user. You will need to use the prd DataFrame , .groupby( ) method & the appropriate aggregation function. Save the results as **avg_days** and name the column as "**days_order_mean**"
2. Create a histrogram for "**days_order_mean**" of **avg_days**. Use arguments: bins=100.

Question 3: How often does a customer reorder products (reorder ratio of a customer)

1. Get the total purchased products of each customer. Save the results as a DataFrame with name **reorders** and name the column as "**total_bought**"
2. For each user, get the distinct number of its purchased products (no reordered products - only products purchased for first time). Save the column as "**total_unique_bought"** on **reorders** DataFrame.

   Hint: Use prd[prd.reordered==0] to perform a .groupby( ) on

3. Create a ratio of **total_unique_bought**/**total_bought**. Save the column as "**reorder_ratio**" on **reorders** DataFrame.
4. Create a histogram for "**reorder_ratio**" of **reorders**. Use arguments: bins=100
5. Sort the rows of **reorders** in order to get first the customers with the lowest "**reorder_ratio**". Store the results on **reorders** DataFrame.
6. Save the index of the customer with the lowest ratio on a new variable. Name it as **low_ratio_customer**. You will use this index (user_id) on the next step.
7. Now use the **low_ratio_customer** to get from **prd** all of its orders and products purchased. Store the results on **user_low**.
8. Perform an appropriate join of **user_low** with **products** so you can get the name of products. Save the results on **user_low** .
9. Get the name of the products that the user has bought. Select the column "**product_name**" of **user_low** DataFrame and use an appropriate method to get the distinct values of the column.

Create variables that describe each customer

Question 1: Create a DataFrame that has the orders and the products purchased

1. Create a DataFrame that contains information for both the orders & order_products_prior DataFrame. Use a inner join and save it as **prd** DataFrame.

In [2]:
```
# Question 0
prd = pd.merge(orders, order_products_prior, how='inner')
prd.head()
```

Out[2]:

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id
0	2539329	1	prior	1	2	8	NaN	196
1	2539329	1	prior	1	2	8	NaN	14084
2	2539329	1	prior	1	2	8	NaN	12427
3	2539329	1	prior	1	2	8	NaN	26088
4	2539329	1	prior	1	2	8	NaN	26405

Question 2: Get the average days since a prior order for each user.

1. Get the average days since a prior order for each user. You will need to use the prd DataFrame , .groupby( ) method & the appropriate aggregation function. Save the results as **avg_days** and name the column as "**days_order_mean**"

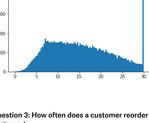
In [3]:
```
# Question 2 - tasks 1
avg_days = prd.groupby('user_id')['days_since_prior_order'].mean()
avg_days.columns = ['days_order_mean']
avg_days.head()
```

Out[3]:

	days_order_mean
user_id	
1	20.259259
2	15.967033
3	11.487179
4	15.357143
5	14.500000

1. Create a histrogram for "**days_order_mean**" of **avg_days**. Use arguments: bins=100.

In [4]:
```
# Question 2 - task 2
plt.hist(avg_days.days_order_mean, bins=100)
plt.show()
```



Question 3: How often does a customer reorder products (reorder ratio of a customer)

1. Get the total purchased products of each customer. Save the results as a DataFrame with name **reorders** and name the column as "**total_bought**"

In [5]:
```
# Question 3 - task 1
reorders = prd.groupby('user_id')['product_id'].count()
reorders.columns = ['total_bought']
```

1. For each user, get the distinct number of its purchased products (no reordered products - only products purchased for first time). Save the column as "**total_unique_bought"** on **reorders** DataFrame.

   Hint: Use prd[prd.reordered==0] to perform a .groupby( ) on

In [6]:
```
# Question 3 - task 2
reorders['total_unique_bought'] = prd[prd.reordered==0].groupby('user_id')['pro
duct_id'].count()
```

1. Create a ratio of **total_unique_bought**/**total_bought**. Save the column as "**reorder_ratio**" on **reorders** DataFrame.

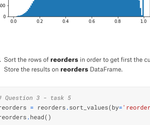
In [7]:
```
# Question 3 - task 3
reorders['reorder_ratio']  =  reorders['total_unique_bought'] / reorders['total_
bought']
reorders.head()
```

Out[7]:

	total_bought	total_unique_bought	reorder_ratio
user_id			
1	59	18	0.305085
2	195	102	0.523077
3	88	33	0.375000
4	18	17	0.944444
5	37	23	0.621622

1. Create a histrogram for "**reorder_ratio**" of **reorders**. Use arguments: bins=100

In [8]:
```
# Question 3 - task 4
plt.hist(reorders.reorder_ratio, bins=100)
plt.show()
```



1. Sort the rows of **reorders** in order to get first the customers with the lowest "**reorder_ratio**". Store the results on **reorders** DataFrame.

In [9]:
```
# Question 3 - task 5
reorders = reorders.sort_values(by='reorder_ratio')
reorders.head()
```

Out[9]:

	total_bought	total_unique_bought	reorder_ratio
user_id			
99753	191	2	0.010471
82414	428	8	0.018692
107528	104	2	0.019231
17997	435	9	0.020690
5588	2223	47	0.021143

1. Save the index of the customer with the lowest ratio on a new variable. Name it as **low_ratio_customer**. You will use this index (user_id) on the next step.

In [10]:
```
# Question 3 - task 6
low_ratio_customer= reorders.index[0]
```

1. Now use the **low_ratio_customer** to get from **prd** all of its orders and products purchased. Store the results on **user_low**.

In [11]:
```
# Question 3 - task 7
user_low = prd[prd.user_id == low_ratio_customer]
```

1. Perform an appropriate join of **user_low** with **products** so you can get the name of products. Save the results on **user_low** .

In [12]:
```
# Question 3 - task 8
user_low = pd.merge(user_low, products, how='left')
```

1. Get the name of the products that the user has bought. Select the column "**product_name**" of **user_low** DataFrame and use an appropriate method to get the distinct values of the column.

In [13]:
```
# Question 3 - task 9
user_low.product_name.drop_duplicates()
```

Out[13]:
```
0    Organic Reduced Fat Milk
1          Organic Whole Milk
Name: product_name, dtype: object
```