Instacart EDA 3 Assignment [ANSWER]

In this assignment you will extend the answer of a previous business insight (question 0) and you will create variables that will describe each customer (question 1,2).

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

In [1]:

```python
#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')
```

Extend the answer of a previous business Insight

Question 0: Create the reorder probability and get the names of the products

1. Follow the steps on Chapter 2 of Instacart EDA 2 Notebook to create the reorder probability for each product. Don't forget to use the equivalent filter. Save the results on **reorprob_results** DataFrame
2. Get the name for the products; use the pandas merge function to combine the results from step 1 with the **products** DataFrame. Use the correct matching key and perform the appropriate join.
3. Sort the results so to get the products with the highest ratio first
4. Visualize the 10 products with the highest ratio and include their names. In order to visualize the labels (ticks) of x-axis properly include the following command in your code before plt.show( ):

> plt.xticks(size=12, rotation=90)

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, maximum & minimum order size for each customer.

1. Get for each customer, the size for every of its orders . You will need to use **prd** DataFrame, perform a .groupby( ) on two columns, select the appropiate column and use the correct aggregation function on it. Save the results as **order_size** DataFrame and name the column as **'size'**
2. Get the average order size for each customer by performing a .groupby( ) on **order_size**. Save the outcome as **results** and name the column as **'order_size_avg'**.
3. Get the smallest & biggest order size for each customer. Perform a .groupby( ) on **order_size**, select the **'size'** column and use one of the following aggregation functions:

Aggregation Function	Description
count	Number of non-null observations
sum	Sum of values
mean	Mean of values
mad	Mean absolute deviation
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
prod	Product of values
std	Unbiased standard deviation
var	Unbiased variance
quantile	Sample quantile (value at %)
cumsum	Cumulative sum

Save the outcomes on **results** with two new columns named **'order_size_smallest'** & **'order_size_biggest'** and display the results.

4. Find the 10 users with the highest **'order_size_avg'** of **results** DataFrame, save the results as **top10_order_size_avg**. Diplay the results.
5. Create a histrogram for **'order_size_avg'** of **results**. Use arguments: bins=100.

Extend the answer of a previous business Insight

Question 0: Create the reorder probability and get the names of the products

1. Follow the steps on Chapter 2 of Instacart EDA 2 Notebook to create the reorder probability for each product. Don't forget to use the equivalent filter. Save the results on **reorprob_results** DataFrame.

In [2]:

```python
reorprob = order_products_prior.groupby('product_id').filter(lambda x: x.shape[0
] >40)
reorprob_results  = reorprob.groupby('product_id', as_index=False)['reordered'].
agg('mean')
```

1. Get the name for the products; use the pandas merge function to combine the results from step 1 with the **products** DataFrame. Use the correct matching key and perform the appropriate join.

In [3]:

```python
reorprob_results  = pd.merge(reorprob_results , products, how='left')
```

1. Sort the results so to get the products with the highest ratio first.

In [4]:

```python
reorprob_results  = reorprob_results.sort_values(by='reordered', ascending=False
)
```

In [5]:

```python
reorprob_results.head(20)
```

Out[5]:

	product_id	reordered	product_name	aisle_id	department_id
3734	6433	0.941176	Raw Veggie Wrappers	13	20
1190	2075	0.931034	Serenity Ultimate Extrema Overnight Pads	126	11
16028	27740	0.920792	Chocolate Love Bar	45	19
7952	13875	0.911111	Simply Sleep Nighttime Sleep Aid	6	2
18170	31418	0.900000	Sparking Water	115	7
20581	35604	0.900000	Maca Buttercups	45	19
21109	36543	0.895522	Bars Peanut Butter	88	13
15097	26093	0.893939	Soy Crisps Lightly Salted	107	19
22065	38251	0.891892	Benchbreak Chardonnay	62	5
21257	36801	0.885417	Organic Blueberry B Mega	31	7
19818	34246	0.884615	Beer Can	27	5
22225	38529	0.878049	Very Rare Blended Scotch Whisky	124	5
5894	10236	0.875969	Fragrance Free Clay with Natural Odor Eliminat...	41	8
23674	41046	0.875000	G Series Orange Sports Drink	64	7
11906	20598	0.875000	Thousand Island Salad Snax	50	19
9005	15657	0.863636	Peanut Butter Honey Spread	29	13
20517	35496	0.862528	Real2 Alkalized Water 500 ml	115	7
3151	5457	0.862069	Classic Carbonated Natural Mineral Water	115	7
5377	9292	0.861691	Half And Half Ultra Pasteurized	84	16
26255	45504	0.860233	Whole Organic Omega 3 Milk	84	16

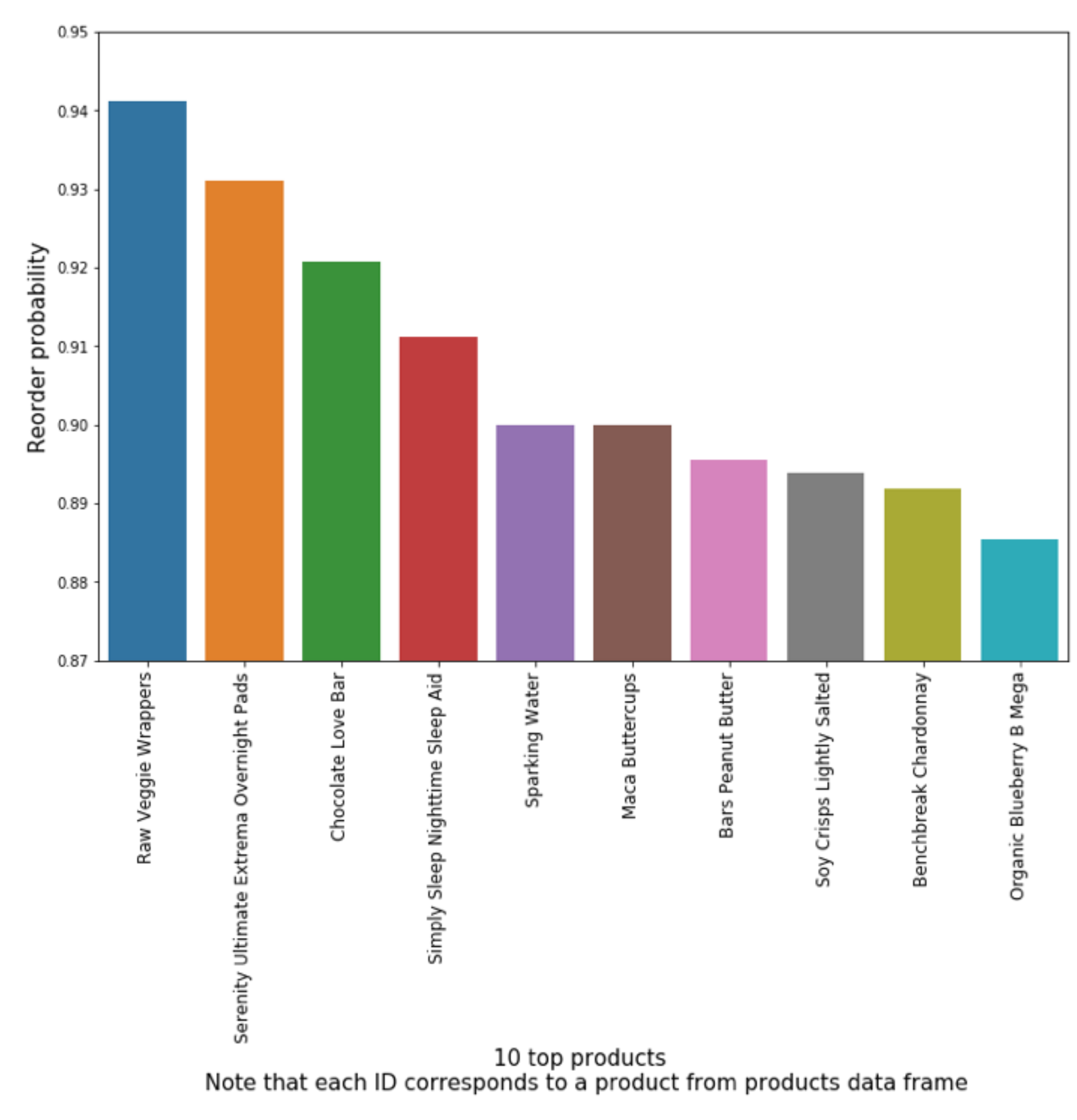
1. Visualize the 10 products with the highest ratio and include their names. In order to visualize the labels (ticks) of x-axis properly include the following command in your code before plt.show( ):

> plt.xticks(size=12, rotation=90)

In [6]:

```python
reorprob_results  = reorprob_results.iloc[0:10]

plt.figure(figsize=(12,8))
sns.barplot(reorprob_results.product_name, reorprob_results.reordered, order=reo
rprob_results.product_name)
plt.xlabel('10 top products \n Note that each ID corresponds to a product from p
roducts data frame', size=15)
plt.ylabel('Reorder probability', size=15)
plt.xticks(size=12, rotation=90)
#we set the range of y-axis to a bit lower from the lowest probability and a bit h
igher from the higest probability
plt.ylim(0.87,0.95)
plt.show()
```



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 [7]:

```python
# Question 1
prd = pd.merge(orders, order_products_prior, how='inner')
prd.head()
```

Out[7]:

	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, maximum & minimum order size for each customer.

1. Get for each customer, the size for every of its orders . You will need to use **prd** DataFrame, perform a .groupby( ) on two columns, select the appropiate column and use the correct aggregation function on it. Save the results as **order_size** DataFrame and name the column as **'size'**

In [8]:

```python
# Question 2 task 1
order_size = prd.groupby(['user_id', 'order_id'])[['product_id']].count()
order_size.columns = ['size']
order_size.head()
```

Out[8]:

		size
user_id	order_id	
1	431534	8
	473747	5
	550135	5
	2254736	5
	2295261	6

1. Get the average order size for each customer by performing a .groupby( ) on **order_size**. Save the outcome as **results** and name the column as **'order_size_avg'**.

In [9]:

```python
# Question 2 task 4
results = order_size.groupby('user_id')[['size']].mean()
results.columns = ['order_size_avg']
results.head()
```

Out[9]:

	order_size_avg
user_id	
1	5.900000
2	13.928571
3	7.333333
4	3.600000
5	9.250000

1. Get the smallest & biggest order size for each customer. Perform a .groupby( ) on **order_size**, select the **'size'** column and use one of the following aggregation functions:

Aggregation Function	Description
count	Number of non-null observations
sum	Sum of values
mean	Mean of values
mad	Mean absolute deviation
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
prod	Product of values
std	Unbiased standard deviation
var	Unbiased variance
quantile	Sample quantile (value at %)
cumsum	Cumulative sum

Save the outcomes on **results** with two new columns named **'order_size_smallest'** & **'order_size_biggest'** and display the results.

In [10]:

```python
# Question 2 task 5
results['order_size_smallest'] = order_size.groupby('user_id')['size'].min()
results['order_size_biggest'] = order_size.groupby('user_id')['size'].max()
results.head()
```

Out[10]:

	order_size_avg	order_size_smallest	order_size_biggest
user_id			
1	5.900000	4	9
2	13.928571	5	26
3	7.333333	5	11
4	3.600000	2	7
5	9.250000	5	12

1. Find the 10 users with the highest **'order_size_avg'** of **results** DataFrame, save the results as **top10_order_size_avg**. Diplay the results.

In [11]:

```python
# Question 2 task 7
top10_order_size_avg = results.sort_values(by='order_size_avg', ascending=False)
.iloc[0:10]
top10_order_size_avg
```

Out[11]:

	order_size_avg	order_size_smallest	order_size_biggest
user_id			
190889	70.250000	34	100
95241	62.000000	45	72
58933	61.000000	40	80
79555	59.000000	48	68
174821	58.500000	31	84
106247	57.250000	40	73
104741	57.000000	49	71
129928	56.843750	1	137
145351	56.666667	51	68
103624	56.166667	18	88

1. Create a histrogram for **'order_size_avg'** of **results**. Use arguments: bins=100.

In [12]:

```python
# Question 2 task 8
plt.hist(results.order_size_avg, bins=100)
plt.show()
```