Instacart EDA 4 Assignment [ANSWER]

In this assignment you will answer a four individual questions (Questions 0-2,3). You can complete any question in any order. **The produced results from one question will not be used on another.**

When you complete this assignment keep this kernel as private, make a successful commit and add as collaborator the REAB.

Note that you will be assessed based on the final version submitted before the deadline.

To answer the questions of this assignment you may create as many code blocks as you wish. In addition, you can use (read) or any other method to display your results. **You can use comments to describe the rationale of your solution.** You can also use comments when you use new methods (that we haven't used in our notebooks), new arguments or alternative approaches for the same problem.

Do not use a single block takes more than 3 minutes to execute, hit the stop button on the left bottom corner and check for any mistakes on your code or your solution in general.

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

In [1]:
```
# Cloud 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

# Local 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')
aisles = pd.read_csv('./input/aisles.csv')
```

Question 0

Q0.0 Create a plot which shows how many orders were placed in each hour of the day.

- use groupby() method

In [2]:
```
orders.groupby('order_hour_of_day')['order_id'].count().plot.bar()
```

Out[2]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa00ec7bf10>
```

Q0.1 What is the percentage allocation of the orders placed from 9 to 5 (09:00-17:00 inclusive) compared to all orders?

- store the final percentage on a variable with name 'pct'
- print the result with the appropriate text

In [3]:
```
orders_hour = orders.groupby('order_hour_of_day')['order_id'].count()
orders_hour.columns = ['volume']
orders_hour = orders_hour.reset_index()
pct = orders_hour[(orders_hour.order_hour_of_day >= 9) & (orders_hour.order_hour
_of_day <= 17)].volume.sum() / orders_hour.volume.sum() * 100
print('The orders placed from 9:00 to 17:00 are ', + str(round(pct,2)) + '% of th
e total orders')
```

Out[3]:
```
The orders placed from 9:00 to 17:00 are 71.45% of the total orders
```

Question 1

Q1.0 Create a DataFrame that keeps all the prior orders and the products that have been purchased

In [4]:
```
prd = pd.merge(orders, order_products_prior, on='order_id', how='inner')
```

Q1.1 For each product find its average position in the cart (in the orders of all customers).

In [5]:
```
item_avg = prd.groupby('product_id')['add_to_cart_order'].mean()
item_avg.head()
```

Out[5]:
```
product_id
1    5.801035
2    9.666667
3    6.410162
4    5.157709
5    4.450847
Name: add_to_cart_order, dtype: float64
```

Q1.2 For a given customer, find the average position of a product on its orders

In [6]:
```
position = prd.groupby(['user_id', 'product_id'])['add_to_cart_order'].mean()
position.columns = ['mean_pos_cart']
position.head()
```

Out[6]:
```
user_id  product_id
1        196         1.400000
         10258       2.333333
         10326       5.666667
         12427       3.200000
         13032       6.333333
Name: add_to_cart_order, dtype: float64
```

Q1.3 Select from user with id==35 the product with id==4942. Is the average position of this product for this user greater than the average position from all users?

In [7]:
```
position = position.reset_index()
user_pos = position[(position.user_id==35) & (position.product_id==4942)]
user_pos.head()
```

Out[7]:

	user_id	product_id	mean_pos_cart
3544	35	4942	12.625

In [8]:
```
user_pos_avg = user_pos.mean_pos_cart.values
item_avg
```

Out[8]:
```
array([12.625])
```

In [9]:
```
avg_all = item_avg[item_avg.index==4942].add_to_cart_order.values
avg_all
```

Out[9]:
```
array([8.27692642])
```

In [10]:
```
user_pos_avg > avg_all
```

Out[10]:
```
array([ True])
```

Question 2

Q2.0 Create a DataFrame that keeps only the prior orders

- You need to keep only the orders (from orders.csv) and not the products that have been purchased in each order.
- Save the DataFrame as 'orders_prior'

In [11]:
```
orders_prior = orders[orders.eval_set== 'prior']
orders_prior.head(10)
```

Out[11]:

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
0	2539329	1	prior	1	2	8	NaN
1	2398795	1	prior	2	3	7	15.0
2	473747	1	prior	3	3	12	21.0
3	2254736	1	prior	4	4	7	29.0
4	431534	1	prior	5	4	15	28.0
5	3367565	1	prior	6	2	7	19.0
6	550135	1	prior	7	1	9	20.0
7	3108588	1	prior	8	1	14	14.0
8	2295261	1	prior	9	1	16	8.0
9	2549362	1	prior	10	4	8	30.0
10	2168274	2	prior	1	2	11	NaN
11	1822842	2	prior	2	5	10	10.0
12	1821847	2	prior	3	5	10	3.0
13	735281	2	prior	4	2	10	8.0
14	847553	2	prior	5	2	11	8.0
15	1199808	2	prior	6	2	10	13.0
16	1506162	2	prior	7	2	12	14.0
17	780316	2	prior	8	1	15	27.0
18	1718025	2	prior	9	2	9	6.0
19	1447447	2	prior	10	1	11	6.0

Q2.1 Keep the orders from the customers that have made at least 25 orders

- Use orders_prior DataFrame that you have created on the previous step
- Save the DataFrame as 'orders_25'

In [12]:
```
orders_25 = orders_prior.groupby('user_id').filter(lambda x: x.order_number.max(
) >= 25)
orders_25.head()
```

Out[12]:

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
492	1727750	8	prior	1	4	10	NaN
493	1881451	17	prior	2	5	10	3.0
494	2080214	17	prior	3	3	10	5.0
495	1420925	17	prior	4	5	11	5.0
496	2037467	17	prior	5	6	17	5.0

In [13]:
```
# Easily check on user_id 28
orders_25[orders_25.user_id==28]
```

Out[13]:

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order

Q2.2 How many days on average pass for these customers to place an order?

In [14]:
```
orders_25.days_since_prior_order.mean()
```

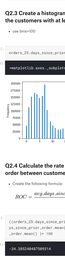
Out[14]:
```
7.8255327215360962
```

Q2.3 Create a histogram for the days that pass since a prior order for the customers with at least 25 orders

- use bins=100

In [15]:
```
orders_25.days_since_prior_order.plot.hist(bins=100)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa00d3ce490>
```



Q2.4 Calculate the rate of change for the average days since prior order between customers with at least 25 orders & all customers

- Create the following formula:

$$ROC = \frac{avg.days.since.prior[\geq 25] - avg.days.since.prior[ALL]}{avg.days.since.prior[ALL]} * 100$$

In [16]:
```
(orders_25.days_since_prior_order.mean() - orders[orders.eval_set== 'prior'].da
ys_since_prior_order.mean()) / orders[orders.eval_set== 'prior'].days_since_prior
_order.mean() * 100
```

Out[16]:
```
-34.101246467585514
```

Question 3

Q3.0 Create a DataFrame that keeps all the prior orders and the products that have been purchased

In [17]:
```
prd = pd.merge(orders, order_products_prior, on='order_id', how='inner')
```

Q3.1 Create a new column which keeps the order_number in reverse order

In [18]:
```
prd['order_number_back'] = prd.groupby('user_id')['order_number'].transform(max)
- prd.order_number +1
```

Q3.2 Keep only the last 10 orders from each user

In [19]:
```
prd10 = prd[prd.order_number_back <= 10]
prd10.head()
```

Out[19]:

	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

Q3.3 Exclude all the customers who have less than 10 orders in total

In [20]:
```
prd10 = prd10.groupby('user_id').filter(lambda x: x.order_id.nunique() >= 10)
```

In [21]:
```
# Easily test
prd10[prd10.user_id==75]
```

Out[21]:

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id

Q3.4 For the last 10 orders, find the average basket size of each user.

In [22]:
```
basket_size= prd10.groupby(['user_id', 'order_id'])['product_id'].count()
basket_size.columns = ['volume']
basket_size.head()
```

Out[22]:
```
                  volume
user_id order_id
1       2539329        8
        2398795        7
        473747         3
        2254736        4
        431534         5
        3367565        6
        550135         7
        3108588        8
        2295261        9
        2549362        8
Name: product_id, dtype: int64
```

In [23]:
```
basket_size = basket_size.reset_index()
basket_size.user = basket_size.groupby('user_id')['volume'].mean()
basket_size.user.head()
```

Out[23]:
```
user_id
1    5.0
2    7.0
3    6.6
4    5.4
5    7.0
Name: user, dtype: float64
```

Q3.5 How many customers have average basket size = 1 for their last 10 orders ?

In [24]:
```
basket_size.user[basket_size.user' volume==1].count()
```

Out[24]:
```
volume    125
dtype: int64
```