Instacart Case Study

Online Groceries

Overview

Purpose & Context

Whilst studying with Career Foundry I completed a project that introduced me to Python and using it to perform an exploratory analysis in order to derive insights and suggest strategies. It focused on Instacart, an online grocery service.

Project Goal

Uncover more information about sales patterns. Perform an initial data and exploratory analysis in order to derive insights and suggest strategies for better segmentation based on the provided criteria

Key Objectives

Identify busiest days of the week and hours of the day for ad scheduling
Establish times of the day when people spend the most money
Which departments have the highest frequency of product orders
Uncover customer ordering behaviours based on various features such as region, family status, loyalty, and other demographics

Overview

Datasets

Instacart Data Sets:

- Data Dictionary
- "The Instacart Online Grocery Shopping Dataset 2017", Accessed from www.instacart.com/datasets/grocery-shopping-2017 via Kaggle on 23.6.23

CareerFoundry Data Sets:

- Customers Data Set

Skills

Python
Data wrangling
Data merging
Deriving variables
Grouping data
Aggregating data
Reporting in Excel
Population flows

Tools

python
NumPy
jupyter
ANACONDA
pandas

Steps

Data Preparation

Analysis process

Results & Recommendations

Data Preparation

Overview

● Basic descriptive exploratory tasks
● Changing data types
● Access values and determine their meaning using a data dictionary

● Download data and import into notebook as a pandas dataframe
● Conduct basic descriptive exploratory tasks
● Change data types of identifier variables into more suitable types and rename columns where needed
● Access values and determine their meaning using a data dictionary

The first step was to clean the 3 datasets (orders, products, customers) if you skip this stage it can result in misleading results. It involved checking for missing values, inconsistent data types, and removing unnecessary columns. Python proved to be a great, and fast, tool for this process.

Analysis Process

Overview

● Creating new dataframes
● Data cleaning
● Merging dataframes
● Creating new columns
● Creating flags
● Creating summary columns
● Creating visualisations

● Creating new dataframes based on a certain criteria
● Fixing mixed-type variables, missing values, and removing duplicates
● Merging sets of dataframes and exporting as pickle
● Creating new columns using conditional logic in the form of if-statements, user-defined functions, the loc() function, and for-loops
● Creating flags
● Creating summary columns of descriptive statistics using the groupby() function
● Creating histograms, bar charts, line charts, and scatterplots
● Putting together a report

Merging the dataframes proved to be the most challenging aspect of this process as I was dealing with 32 million rows of data. I encountered a few issues with freezing due to processing power when working on the dataframe, so eventually had to split the data into a training and test set (70/30). I also changed the data types to reduce the size of the dataframe, making it as streamlined as possible. Moving forward I used this smaller set to perform a lot of the analysis. This also prevented crashing issues when creating visualisations. In the future I would have to use a more powerful machine.

Results & Recommendations

Overview

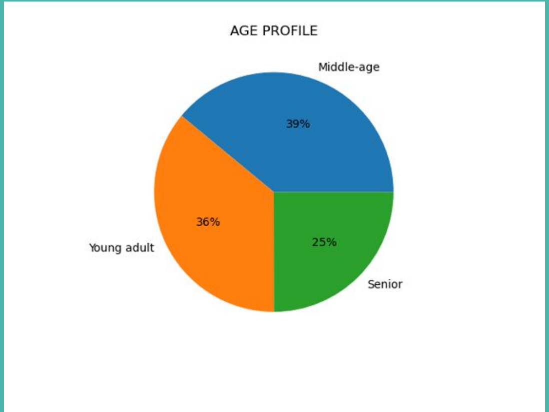
- Schedule advertising
- Grouping products by mark-up
- Promotions
- Incentivise brand loyalty.
- Targeted advertising.

From the analysis process I was able to come up the following recommendations,

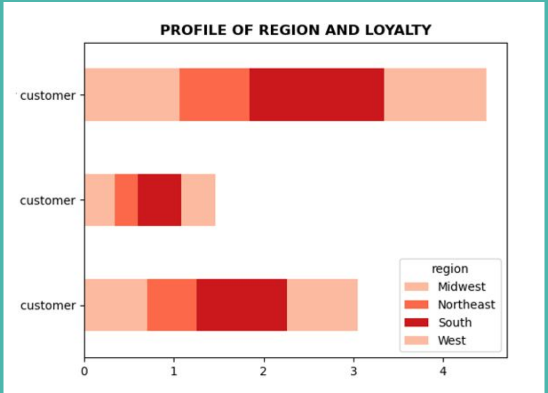
- Focus advertising high-end products during early hours targeting commuters and high-earners
- Groupings are based just on purchase price. They could also group products by mark-up and attribute values to them that way.
- Attach promotions to these best selling items as they seem to be staples, which then encourage customers to buy less well selling items.
- Incentivise brand loyalty by introducing repeat customer bonus schemes, a points system for example.
- Those over 40 have more spending power so targeting advertising to this age group could help boost income.

Visualisations

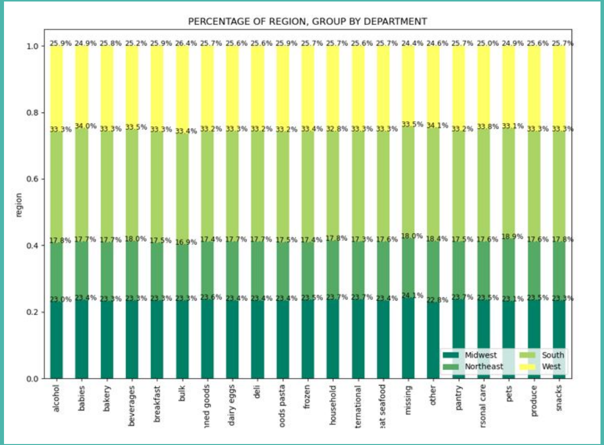
In the following pages are some of the visualisations



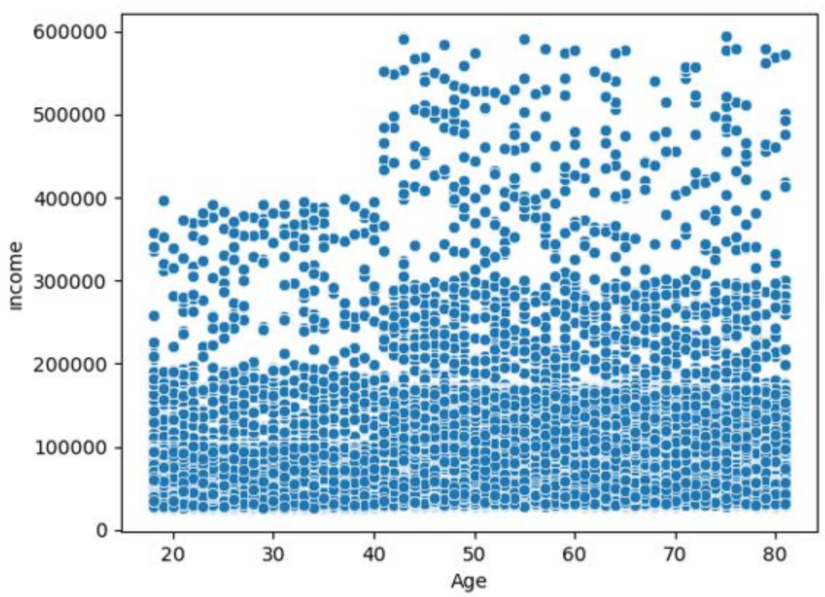
This pie chart shows that the majority of customers are aged under 65.



The most loyal customers are in the south.

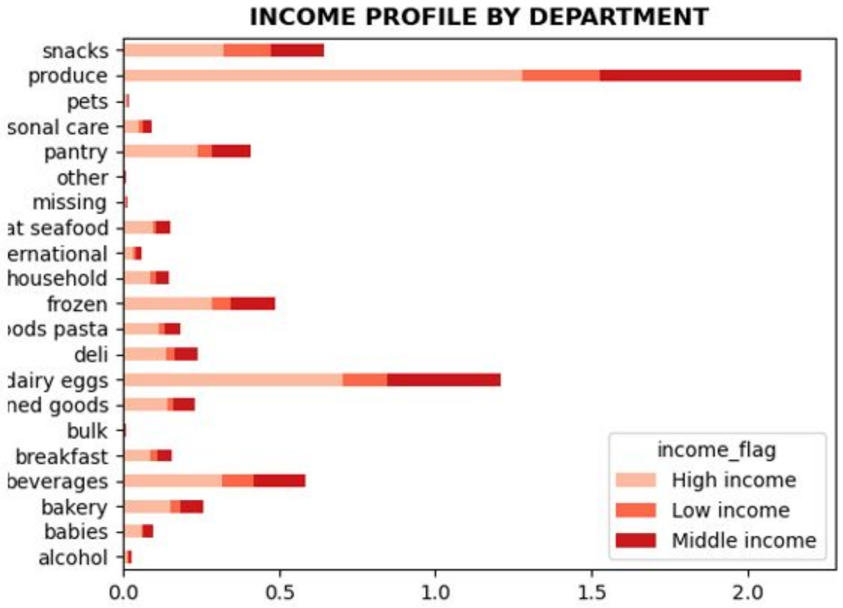


Scatterplot



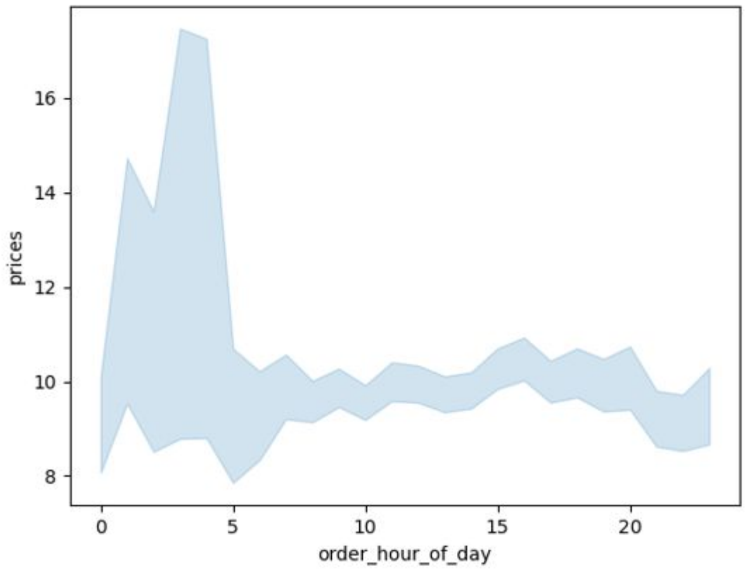
This scatterplot shows the relationship between customer age and income.
It shows that the majority of people earn under 200k. Very few 20-40 year olds earn more than 200k, with none earning more that 400k per year. People older than 40 start making over 200k, with some earning up to 600k.
From this chart we can see that those over 40 have more spending power.

Stacked Bar Chart



This shows that high income customers spend more per department than those earning less by roughly the same proportion as the income profile share (58%). An exception is snacks, where they only make up 50%, suggesting that those on lower incomes by more snacks.

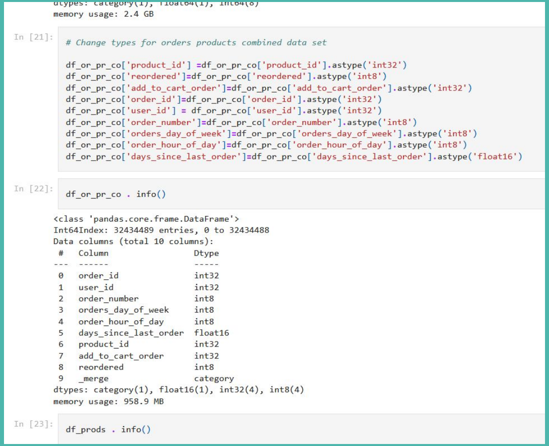
Line Plot Chart



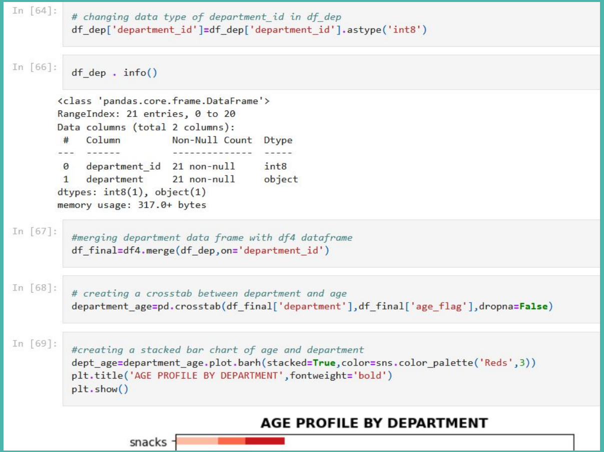
This line plot chart shows the average price of items ordered by order hour of the day.
It seems people spend more between 3am and 5am.

Python Code

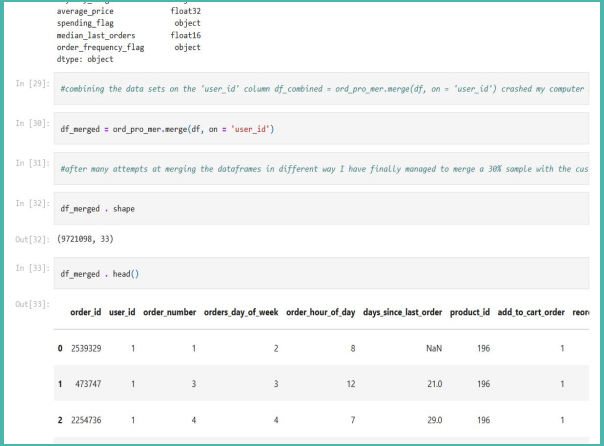
Below are some examples of code used throughout the analysis. Changing data types, creating cross-tabs and merging dataframes.



```
dtypes: category(1), float64(1), int64(8)
memory usage: 2.4 GB

In [21]:  # Change types for orders products combined data set

          df_or_pr_co['product_id'] =df_or_pr_co['product_id'].astype('int32')
          df_or_pr_co['reordered']=df_or_pr_co['reordered'].astype('int8')
          df_or_pr_co['add_to_cart_order']=df_or_pr_co['add_to_cart_order'].astype('int32')
          df_or_pr_co['order_id']=df_or_pr_co['order_id'].astype('int32')
          df_or_pr_co['user_id'] = df_or_pr_co['user_id'].astype('int32')
          df_or_pr_co['order_number']=df_or_pr_co['order_number'].astype('int8')
          df_or_pr_co['orders_day_of_week']=df_or_pr_co['orders_day_of_week'].astype('int8')
          df_or_pr_co['order_hour_of_day']=df_or_pr_co['order_hour_of_day'].astype('int8')
          df_or_pr_co['days_since_last_order']=df_or_pr_co['days_since_last_order'].astype('float16')

In [22]:  df_or_pr_co . info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 32434489 entries, 0 to 32434488
Data columns (total 10 columns):
 #   Column                 Dtype
---  ------                 -----
 0   order_id               int32
 1   user_id                int32
 2   order_number           int8
 3   orders_day_of_week     int8
 4   order_hour_of_day      int8
 5   days_since_last_order  float16
 6   product_id             int32
 7   add_to_cart_order      int32
 8   reordered              int8
 9   _merge                 category
dtypes: category(1), float16(1), int32(4), int8(4)
memory usage: 958.9 MB

In [23]:  df_prods . info()
```



```
In [64]:  # changing data type of department_id in df_dep
          df_dep['department_id']=df_dep['department_id'].astype('int8')

In [66]:  df_dep . info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   department_id  21 non-null     int8
 1   department     21 non-null     object
dtypes: int8(1), object(1)
memory usage: 317.0+ bytes

In [67]:  #merging department data frame with df4 dataframe
          df_final=df4.merge(df_dep,on='department_id')

In [68]:  # creating a crosstab between department and age
          department_age=pd.crosstab(df_final['department'],df_final['age_flag'],dropna=False)

In [69]:  #creating a stacked bar chart of age and department
          dept_age=department_age.plot.barh(stacked=True,color=sns.color_palette('Reds',3))
          plt.title('AGE PROFILE BY DEPARTMENT',fontweight='bold')
          plt.show()
```



```
average_price             float32
spending_flag              object
median_last_orders        float16
order_frequency_flag       object
dtype: object

In [29]:  #combining the data sets on the 'user_id' column df_combined = ord_pro_mer.merge(df, on = 'user_id') crashed my computer

In [30]:  df_merged = ord_pro_mer.merge(df, on = 'user_id')

In [31]:  #after many attempts at merging the dataframes in different way I have finally managed to merge a 30% sample with the cus

In [32]:  df_merged . shape

Out[32]:  (9721098, 33)

In [33]:  df_merged . head()

Out[33]:
```

	order_id	user_id	order_number	orders_day_of_week	order_hour_of_day	days_since_last_order	product_id	add_to_cart_order	reor
0	2539329	1	1	2	8	NaN	196	1	
1	473747	1	3	3	12	21.0	196	1	
2	2254736	1	4	4	7	29.0	196	1	

Thanks

https://github.com/danfradat/Instacartpython/tree/main