MARKETING ANALYSIS

Tools: Jupyter Notebook + Libraries

Language: Python

OVERVIEW:
Instacart is an online grocery store that operates through a mobile application in the U.S and Canada. The stakeholders are mostly interested in the variety of customers in their database along with their purchasing behaviors to implement a targeted marketing strategy.

OBJECTIVE:
The purpose of this student project was to dive into analytics with Python, practice the preparation of data analysis, and conduct the most fundamental data engineering tasks using large amount of data.

DATASETS:
The datasets contain open-source data from 2017 made available online by Instacart. The consumer data and the prices of the products were both fabricated for learning purposes.

This study project was part of my data analytics course with CareerFoundry.

SETTING UP THE CODING ENVIROMENT

To start with this project, I set up the Python environment by installing Anaconda, a Python's helper program, wrote and ran scripts in Jupyter notebook.

For the analysis, Pandas and NumPy libraries were used and, later on, also Matplotlib, Seaborn, and SciPy libraries to plot and visualize the results.

UNDERSTANDING THE DATA

The four data sets contain observations about users, orders and products ordered.

Out of four sets analyzed, one was very specific: it contained over 32 million observations, an amount of data impossible to view at a glance without the use of Python and descriptive statistics to explore.

```
df_ORDS_large.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32640698 entries, 0 to 32640697
Data columns (total 9 columns):
 #   Column                  Dtype
---  ------                  -----
 0   order_id                int64
 1   user_id                 int64
 2   order_number            int64
 3   order_day_of_the_week   int64
 4   order_hour_of_day       int64
 5   days_since_prior_order  float64
 6   product_id              float64
 7   add_to_cart_order       float64
 8   reordered               float64
dtypes: float64(4), int64(5)
memory usage: 2.2 GB
```

```
df_CUST.describe()
```

	user_id	age	n_dependants	income
count	206209.000000	206209.000000	206209.000000	206209.000000
mean	103105.000000	49.501646	1.499823	94632.852548
std	59527.555167	18.480962	1.118433	42473.786988
min	1.000000	18.000000	0.000000	25903.000000
25%	51553.000000	33.000000	0.000000	59874.000000
50%	103105.000000	49.000000	1.000000	93547.000000
75%	154657.000000	66.000000	3.000000	124244.000000
max	206209.000000	81.000000	3.000000	593901.000000

WRANGLING THE DATA

While running a consistency check on data, 206,209 missing values in the column 'days_since_prior_order' were discovered. Logically, those missing values indicated all orders that were placed for the first time and I decided not to replace them with any other value.

```
df_ORDS.isnull().sum()
```

```
order_id                  0
user_id                   0
order_number              0
order_day_of_the_week     0
order_hour_of_day         0
days_since_prior_order    206209
dtype: int64
```

```
df_PRODS['product_name'].value_counts(dropna = False)
```

```
NaN                                    16
Adore Forever Body Wash                 2
Black House Coffee Roasty Stout Beer    2
Ranger IPA                              2
Fiber 4g Gummy Dietary Supplement       2
                                       ..
Charcoal Briquettes Mesquite            1
Dentastix® Mini Dog Chews               1
Sugar Free Gum with Xylitol Spearmint   1
Original Thin Sausage Pizza             1
Vinegar & Sea Salt Potato Chips         1
Name: product_name, Length: 49673, dtype: int64
```

In the set with products, 16 observations with missing product names were found. There was no reason to keep them for the analysis as no shopper would buy a product without knowing what it is, so I dropped all the rows with missing product names. The exploration also revealed double entries for the same product.

The summary statistics helped to discover strange values. I found the maximal product price of $99,999 – what an unusual price for a grocery store! In a business setup, this would have to be communicated to stakeholders.

```
df_PRODS_clean.describe()
```

	prices
count	49677.000000
mean	9.993164
std	453.592708
min	1.000000
25%	4.100000
50%	7.100000
75%	11.100000
max	99999.000000

1st MERGER

After dropping redundant columns and renaming others to give them more intuitive names, the two sets were ready to be merged. The joining operation was based on the 'order_id' column common for all sets:

```
# merging the two data sets on column "order_id"
# the order set has  3.421.083 rows
# the prior set has 32.434.489 rows
```

```
df_ORDS_large = df_ORDS.merge(df_ORDS_PRODS_prior, on = 'order_id', how = 'outer', indicator = True)
```

This merging procedure allowed to combine products with orders and users, and created a first foundation for analysis of shoppers' behaviors, as expected by project's stakeholders.

MORE MISSING VALUES

Additional consistency check of the merged data revealed 11 missing values in the orders' columns:

```
df_NaN_order_id
```

order_id	user_id	order_number	order_day_of_the_week	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	product_name
NaN	NaN	NaN	NaN	NaN	NaN	3630.0	NaN	NaN	Protein Granola Apple Crisp
NaN	NaN	NaN	NaN	NaN	NaN	3718.0	NaN	NaN	Wasabi Cheddar Spreadable Cheese
NaN	NaN	NaN	NaN	NaN	NaN	7045.0	NaN	NaN	Unpeeled Apricot Halves in Heavy Syrup
NaN	NaN	NaN	NaN	NaN	NaN	25383.0	NaN	NaN	Chocolate Go Bites

Upon investigation of the missing data, I came to a conclusion that those 11 products have not been ordered yet by any of the users, as they were never assigned to any order number. Quite an important insight for the products department of Instacart.

2nd MERGER

The combination of all four sets turned out to be challenging. With many millions of observations, one small mistake in coding, which I previously overlooked, created memory errors causing my computer to freeze. After learning this a hard way, I changed my approach and performed the mergers once more to make sure all sets are clean from any mistakes.

Keeping this lesson in mind, I additionally resized the new set by creating a sample containing 10% of the merged data with the use of a random number generator:

```
# Since I had so many problems with the dataset size, I am creating a sample containing 10% of records.

dev = np.random.rand(len(df_COP_large)) <= 0.9
```

```
small_df_COP = df_COP_large[~dev]
```

This solution allowed me to apply transformation procedures like deriving new columns using if-statements, loc and for-loops functions, as well as grouping and aggregating methods without interruptions caused by memory errors. With this clean slate, I was able to enrich my data set with new variables to derive insights about Instacart customers:

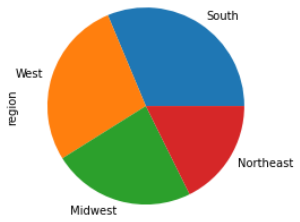
```
# Use "if-statment for-loops" and create and empty list

result = []

for state in df_COP['state']:
    if (state == 'Maine') or (state == 'New Hampshire') or
        result.append('Northeast')
    elif (state == 'Deleware') or (state == 'Maryland') or
        result.append('South')
    elif (state == 'Wisconsin') or (state == 'Michigan') o
        result.append('Midwest')
    else:
        result.append('West')
```

```
# Create the new column "region"

df_COP['region'] = result
```

```
# Check the frequency

df_COP['region'].value_counts(dropna = False)
```

```
South        1021858
West          900824
Midwest       764761
Northeast     576574
Name: region, dtype: int64
```

```
df_reg = df_COP['region'].value_counts().plot.pie()
```

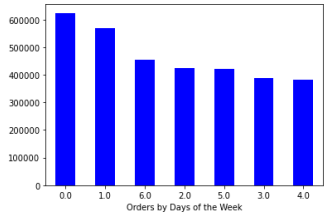


CUSTOMER BEHAVIORS AND THEIR PROFILES

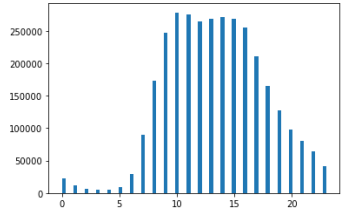
Based on the project brief, the Instacart stakeholders are interested in the variety of customers in their database along with their purchasing behaviors.

At this point, my data set was perfectly clean, so I was ready to explore behaviors of Instacart users, build their profiles, check for relationships, and visualize the results.

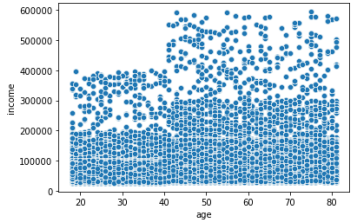
I used histograms, scatterplots, bar and line charts to achieve it and it was the most rewarding part of the entire analysis, when the carefully curated numbers turned into pictures.



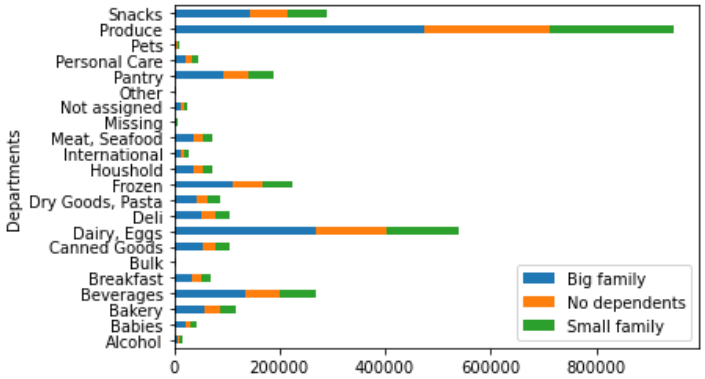
Shoppers are most active on Sundays and Mondays.



The peak of orders is between 9 am – 5 pm.



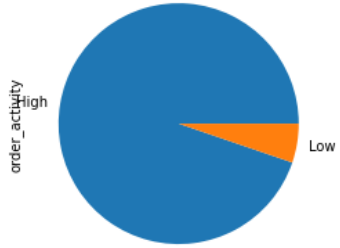
There is no direct relationship between age and spending power of a user, but age group 20-40 has less spending power than the users 40+.



Produce is in the highest demand, followed by Dairy & Eggs, Snacks, Beverages, and Frozen.
Big families with 2+ dependents are the major shoppers throughout all departments.

READY FOR RECOMMENDATIONS

The discovered trends and insights allowed me to address the stakeholders' considerations for applying targeted marketing strategy and suggest a few promotional activities, for example:

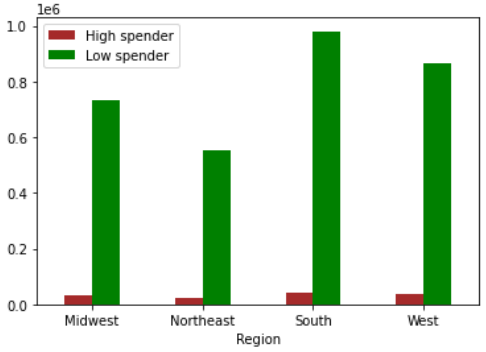


order_activity	days_since_prior_order mean
High	10.810479
Low	19.280465

Shoppers which place 5 and more orders are the majority as shown in the pie chart. Logically, they also shop more often as their average break between orders is around 10 days (shown in the table to the left). A good incentive campaign towards the low active users could make them place more orders.

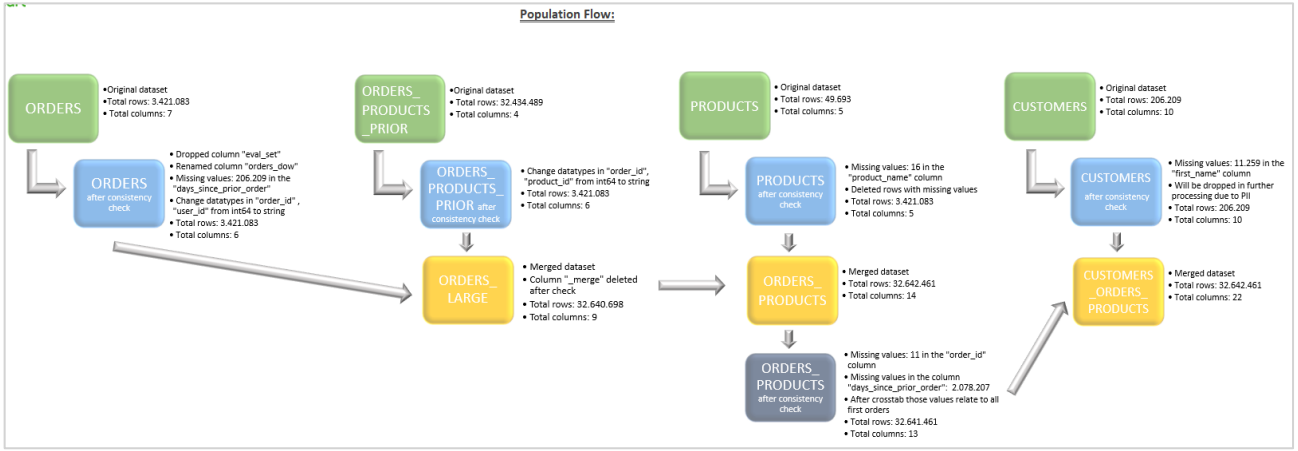
Users across all regions are characterized by similar spending levels: almost all pay less than $10 on average for their products as they are grouped in the 'Low spender' category. They also place more orders on average than 'High spenders'. It would be beneficial to promote activities that would make 'High spenders' place more orders as they buy more expensive products on average.

spending_flag	order_number mean
High spender	9.365759
Low spender	17.464234



FINISHING UP

Finally, I documented the entire analysis process in the form of Excel Reporting that contains the population flow, describes wrangling and deriving operations, shows visualizations of results along with recommendations for the new marketing strategies.



All project's files are stored in my GitHub repository.

CONCLUSIONS

- The large sets with millions of observations required a certain level of abstract thinking about the numbers to recognize what they indicate and to interpret them in the context of the project's scope.

- The merger operations of large data sets turned out to cause memory problems on my computer. This taught me to execute procedures with respect to available hardware resources. Additionally, I learned to look for solutions and to try different approach even if this meant starting all over again.

- While conducting data analysis through steps of data wrangling procedures and the use of Python, I found myself loosing sight of the analysis' objective. By asking myself "Why I did this particular operation?", I managed to refocus and succeeded in uncovering insights about customers' behaviors.

SKILLS LEARNED

- ❑ Python
- ❑ Jupyter Notebook
- ❑ Reporting in Excel
- ❑ Population flow
- ❑ Data wrangling
- ❑ Data merging
- ❑ Deriving variables
- ❑ Grouping data
- ❑ Aggregating data