INSTACART

CASE STUDY

Overview

Instacart, an online grocery store that operates through an app, is looking to uncover more information about their customers background and purchasing behavior to optimize their targeted marketing strategy.

Purpose and Context

This was a personal project I built as part of my Data Analytics course at CareerFoundry to demonstrate my mastery of Python.

Objective

Perform an initial data and exploratory analysis of Instacart's data in order to derive insights, suggest strategies for better segmentation and answer business questions.

Tools

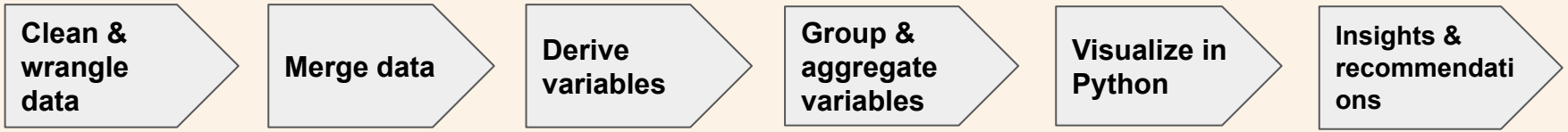
Python, Jupyter Notebook, Anaconda, Excel
Libraries: Pandas, NumPy, Matplotlib, Seaborn, SciPy

Skills

Data wrangling, Data consistency checks, Data merging, Deriving variables, Grouping data, Aggregating data, Visualization, Population flows, Excel reporting.

Data

Instacart data
Customer data



1. Cleaning and wrangling

- Wrangled and cleaned 5 data sets (orders, products, orders_products_prior, customers, departments) in python using Jupyter Notebook.
- Applied below wrangling procedures to every data set:
    - Dropping columns
    - Renaming columns
    - Changing data types
    - Transposing data

```
# renaming order_dow column
df_ords.rename(columns={'order_dow':'orders_day_of_week'}, inplace=True)
```

```
df_ords.head()
```

	order_id	user_id	order_number	orders_day_of_week	order_hour_of_day	days_since_prior_order
0	2539329	1	1	2	8	NaN
1	2398795	1	2	3	7	15.0
2	473747	1	3	3	12	21.0
3	2254736	1	4	4	7	29.0
4	431534	1	5	4	15	28.0

```
# changing the data type of order_id column
df_ords['order_id']=df_ords['order_id'].astype('str')
```

```
df_ords['order_id'].dtype
```

```
dtype('O')
```

- Found and addressed below consistency checks in every data set:
  - Mixed-type data
  - Missing values
  - Duplicates

```
# check for mixed-type data
for col in df_customers.columns.tolist():
  weird = (df_customers[[col]].applymap(type) != df_customers[[col]].iloc[0].apply(type)).any(axis = 1)
  if len (df_customers[weird]) > 0:
    print (col)
```

There are no mixed-type data

```
# check for missing values
df_customers.isnull().sum()
```

```
user_id          0
gender           0
state            0
age              0
n_dependants     0
family_status    0
income           0
dtype: int64
```

There are no missing values

```
# check for duplicates
df_customers[df_customers.duplicated()]
```

user_id	gender	state	age	n_dependants	family_status	income

There are no duplicates

2. Merging

- Data sets were merged into a unified version of 30M+ records.
- Exported the final dataframe to pkl format for faster import and export as well as a high compression rate.

```
# merge df_ords and df_ords_prior using order_id as key, add a merge flag
df_merged_large=df_ords.merge(df_ords_prior, on='order_id', indicator=True)
```

3. Deriving variables

- Derived new variables using if-statements and for-loops by grouping and aggregating data.
- For example, created a loyalty flag for customers using below steps:
  - Apply transform() function, which will create a new column containing the maximum frequency of the number of orders.
  - Apply loc() function which will create a second column containing a flag designating whether a customer is loyal or not.
  - Criteria was defined by the number of orders per customer: over 40, less than 10, or in between.

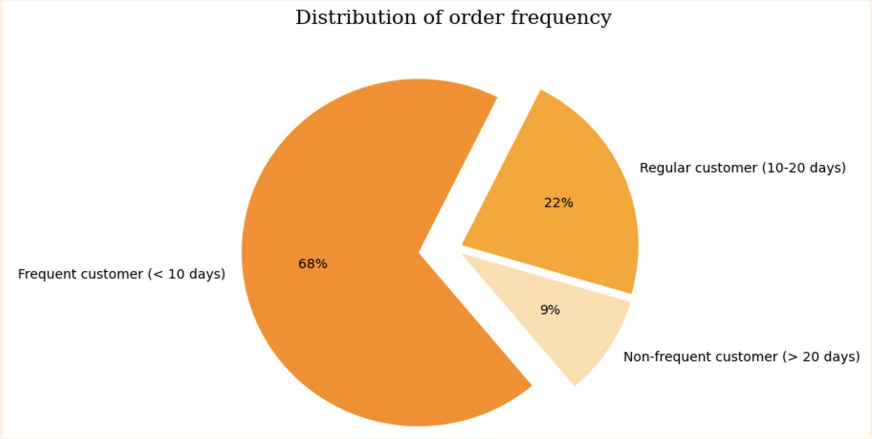
```
# create max_order column by splitting data into user_id groups and applying transform function on order_number
df_ords_prods_merged['max_order']=df_ords_prods_merged.groupby(['user_id'])['order_number'].transform(np.max)
```

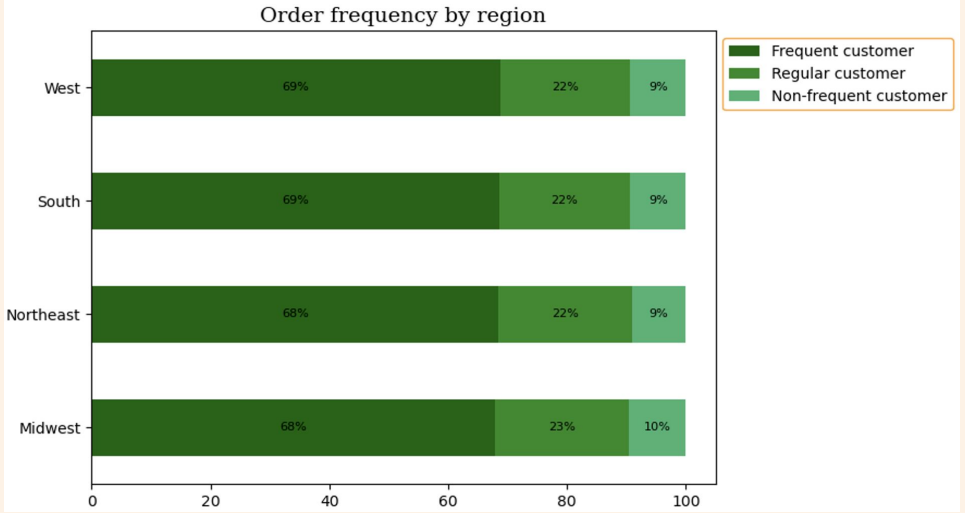
```
# create a loyalty flag for 'Loyal customer' using loc
df_ords_prods_merged.loc[df_ords_prods_merged['max_order']>40, 'loyalty_flag']='Loyal customer'
```

```
# create a loyalty flag for 'Regular customer' using loc
df_ords_prods_merged.loc[(df_ords_prods_merged['max_order'] <= 40) &
                         (df_ords_prods_merged['max_order'] > 10), 'loyalty_flag'] = 'Regular customer'
```

```
# create a loyalty flag for 'New customer' using loc
df_ords_prods_merged.loc[df_ords_prods_merged['max_order']<=10, 'loyalty_flag']='New customer'
```

- Much of the analysis revolved around identifying trends and patterns by:
  - first showing overall distribution of newly derived variables such as loyalty status, ordering frequency, and price range of products
  - and second, comparing and contrasting distribution of these variables with other characteristics of the data set, like region, family status, etc.



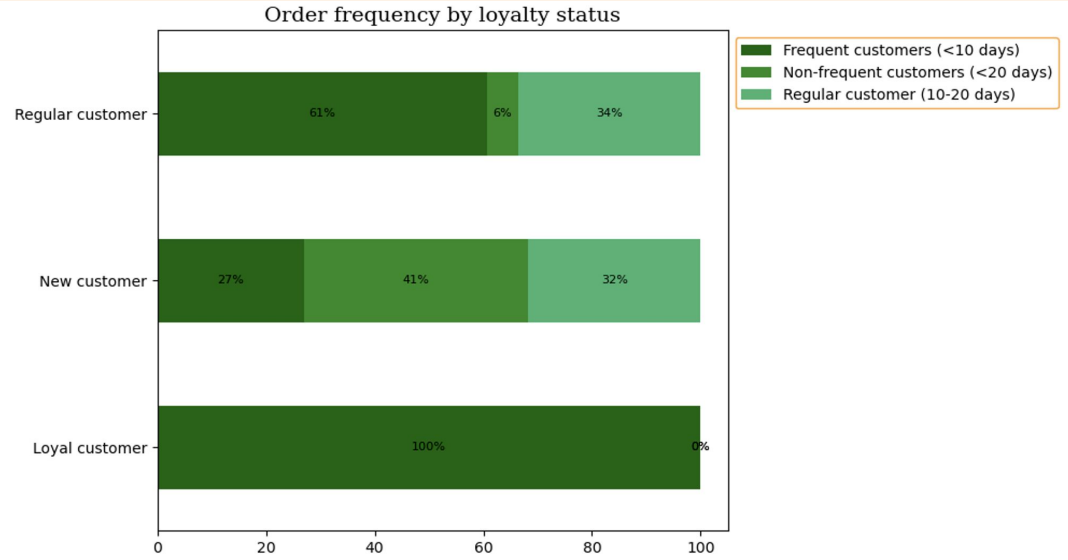


4. Crosstabs

- Created crosstabs to help build customized data frames aiding in analysis.
- For example, created a crosstab between loyalty status and frequency of placing orders.

```
# create a crosstab of order frequency by loyalty status
order_freq_by_loyalty=pd.crosstab(df_final['loyalty_flag'],
                                  df_final['order_frequency_flag']).apply(lambda r: r/r.sum()*100, axis=1)
```

- Analyzed values in % to draw comparative analyses and conclusions excluding the size effect of different groups. Also, since the data being evaluated is extensive, expressing in % would present an accurate value.



- Uncovered an interesting insight when it came to ordering habits based on a customer's region.
- 33% of customer base comes from the southern region, making up the largest proportion of Instacart, whereas Northeast has the lowest proportion of customers at 18%.
- However, as per the **US Census Bureau 2022**, South has the highest resident population in US.

Region	Proportion of population (source: US census bureau)	Proportion of Instacart customers	Affinity
South	**39%**	**33%**	-15%
West	24%	26%	8%
Midwest	21%	24%	14%
Northeast	17%	18%	6%

- Although south has the largest customer base of Instacart, this market is not being utilized to the fullest potential owing to its significantly high population, making it the least successful of all regions.
- Comparing the share of customers with the share of population for each US region revealed this important and interesting observation.

5. Customer segmentation

- Created 9 customer profiles based on age, income, number of dependants, and family status by applying loc() function.

```
# create a list variable to calculate quantile of income
income_list=list(df_final['income'].quantile([.15, .90]))
```

```
# create customer profile for families

df_final.loc[(df_final['age']<=65) & (df_final['n_dependants']>0) &
             (df_final['income']<=income_list[0]), 'customer_profile']='family low-income'

df_final.loc[(df_final['age']<=65) & (df_final['n_dependants']>0) & (df_final['income']>income_list[0]) &
             (df_final['income']<=income_list[1]), 'customer_profile']='family avg-income'

df_final.loc[(df_final['age']<=65) & (df_final['n_dependants']>0) &
             (df_final['income']>income_list[1]), 'customer_profile']='family high-income'
```

Code example to create customer profile for families

Insights & Recommendations

- Weekend (friday, saturday, sunday) are the busiest days and the hours from morning to afternoon (9-15) are the busiest time of the day. Schedule more ads during the middle of the week on days such as tuesday, wednesday and after 5pm up until 8 in the morning to enhance sales at times when there are fewer orders.

- More than 60% of the products are purchased from mid-range category, priced between $5-$15. Sponsor more high-range products during the hours when customers tend to spend most i.e., from late night to early morning.

- Fresh products like organic fruits and vegetables are the most re-ordered items, whereas produce and dairy eggs are the most popular departments. Stock more options and varieties when it comes to organic food items considering its popularity. Recommend popular products to new customers. Aim to optimise aisle positioning that is designed to promote cross-selling and up-selling by placing aisles and products with high popularity and demand near each other.

- More than 60% customers purchase frequently within 10 days, while less than 10% purchase after 20 days.

- Loyal customers on average order every 5 days, where regular customers order every 11 days and new customers every 18 days. Aim to fully convert this segment of regular and new customers into purchasing more frequently. For example, giving promotional coupons to new customers which expire in one week can encourage them to order more often.

- Sales performance in South is relatively poor than the other regions. Prioritize southern market when it comes to marketing and luring customers in order to seize the potential within this customer base. Introduce incentives such as cash back, coupon, promo code, discount offers to draw new customers and impress existing ones.

- The age groups of 18-24 and >75 years have a low customer base. Provide promotional deals and special student discount offers for customers aged 18-24 years. Increase engagement with senior customers by asking appropriate questions, setting up polls, posting relatable and interactive content. Make most of simple opportunities such as maximizing the marketing calendar, for example August 25 is Senior Citizens Day – the perfect time to reach the older audience with flash sales and promotions.

- Customers with low income on average purchase snacks 10% more and beverages 4% more in comparison to other customer profiles. Advertise more products from the snacks and beverages department to low-income groups. Create effective marketing strategy for low income audience in order to attract existing as well as acquire new customers by providing discount vouchers and promo codes, specially for products from produce and dairy eggs departments.

Reflections

This was both my favorite and most challenging project. It allowed me to apply my analytical skills for exploratory analysis in a large dataset consisting of 30M+ records. Quite a lot when it is your first time trying to handle this massive amount of data! Python is supremely good at doing this job and is incredibly fast. I quickly discovered why Python is known as the Swiss Army knife of the coding world. My favorite part was how quick and easy it was to perform complex operations on data, especially compared to conventional analysis tools like Excel. Not only did I develop my coding skills, but also my research skills in the process. Each time I had a doubt or my code broke, I took to helpful blogs online like Stack Overflow, GeeksforGeeks, among others, where I found solutions to my problems and learned a little extra every time. I enjoyed using Jupyter Notebook, especially how I could write my code in one cell and check the output in the next cell. This made locating and fixing errors in my code a lot easier.