PACMANN Data Wrangling Project – 2017 Product Analysis

By: Muhammad Alif Aqsha

Introduction

A proper relational database has become crucial for businesses and companies to have in this big data age, especially for e-commerce. These days, with the rapid adaptation of smartphones, the market behavior of customers is increasingly recorded in real time, providing a big volume of valuable data for e-commerces. This data, if organized properly, will allow them to gain valuable insights on how their businesses are performing, analyze their strong and weak points, and optimize their business processes accordingly to gain more profits or avoid more losses. As an example, they could look into which products are popular or usually bought in bundles (more than one quantity in an order), and thus adjust their prices or promos. On the other hand, they could look into which products are extremely unpopular (e.g. nobody has bought them in a period) and decide whether they will continue to display said product on their website.

Objectives

In this project, we are going to identify which categories of products

- bring the largest sales
- are sold the most
- have the largest number of customers
- are usually bought in bundles (more than one per order)
- have the highest review score
- have the highest share of unsold products

in the year 2017.

Data Querying & Description

Data Querying

To get a table of data consisting of the product descriptions (category, product id), sales, and review score, all in the year 2017 (at the time the order was created), we are going to query from the tables

- olist_order_items_dataset

- olist_order_dataset
- olist_order_reviews_dataset
- olist_order_customer_dataset

```
query = '''
WITH agg_products_1 AS(
SELECT
order_id,
product_id,
customer_unique_id,
SUM(price) AS total_price,
COUNT(order_id) AS num_sold,
CASE WHEN COUNT(order_id) > 1 THEN 1
ELSE 0
END more_than_one,
AVG(review_score) AS review_score
FROM olist_order_items_dataset
LEFT JOIN olist_order_dataset USING(order_id)
LEFT JOIN olist_order_reviews_dataset USING(order_id)
LEFT JOIN olist_order_customer_dataset USING(customer_id)
WHERE DATE(order_purchase_timestamp) <= DATE('2017-12-31')
AND DATE(order_purchase_timestamp) >= DATE('2017-01-01')
AND order_status = 'delivered'
GROUP BY order_id, product_id
),
```

```
agg_products_2 AS(
SELECT
product_id,
SUM(total_price) AS sales,
SUM(num_sold) AS total_sold,
COUNT(order_id) AS num_orders,
SUM(more_than_one) AS num_times_more_than_one,
COUNT(DISTINCT customer_unique_id) AS num_uniq_customers,
AVG(review_score) AS mean_review_score
FROM agg_products_1
GROUP BY product_id
)
```

```
SELECT
product_category_name_english,
product_category_name,
product_id,
sales,
total_sold,
num_orders,
num_times_more_than_one,
num_uniq_customers,
mean_review_score
FROM olist_products_dataset
LEFT JOIN agg_products_2 USING(product_id)
LEFT JOIN product_category_name_translation USING(product_category_name)
'''
```

Data Description

Column	Description	Type
product_category_name_english	Product category name in English	Object
product_category_name	Product category name in Portuguese	Object
product_id	Product unique id	Object
sales	Total sales generated by the product in 2017	Float
total_sold	Total number of sold products in 2017	Integer
num_orders	How many times this product were ordered in 2017	Integer
num_times_more_than_one	How many times this product were ordered in bundles (more than 1 in quantity) in 2017	Integer
num_uniq_customers	How many customers ordered this product in 2017	Integer
mean_review_score	Mean review score of this product in 2017	Float

Note: In the original olist_order_reviews_dataset table, there is really no product identification (only order_id identification), so for orders with more than 1 item, there is no column identifying which product is being reviewed. Due to this, we just assume that the reviewer's score for the product is the mean review score out of all scores within the order.

Data Cleaning

After checking the data information, the table has null values in product_category_name_english, product_category_name, sales, total_sold, num_orders, num_times_more_than_one, num_uniq_customers, and mean_review_score columns.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   product_category_name_english  32328 non-null  object
 1   product_category_name          32341 non-null  object
 2   product_id                     32951 non-null  object
 3   sales                          16803 non-null  float64
 4   total_sold                     16803 non-null  float64
 5   num_orders                     16803 non-null  float64
 6   num_times_more_than_one        16803 non-null  float64
 7   num_uniq_customers             16803 non-null  float64
 8   mean_review_score              16716 non-null  float64
dtypes: float64(6), object(3)
memory usage: 2.3+ MB
```

Cleaning sales, total_sold, num_orders, num_times_more_than_one, and num_uniq_customers

For missing values from these columns, we impute them with 0 as NaN value meant that the products were not sold at all in 2017.

Cleaning product_category

First, we check whether there are rows with defined product_category_name but NaN-valued product_category_name_english (or otherwise). It turns out that there is several products with undefined product_category_name_english but defined product_category_name. The Portuguese product_category_names for these row is ‘pc_gamer’ and ‘portateis_cozinha_e_preparadores_de_alimentos’ (kitchen racks) , so we just impute the English category names with ‘pc_gamer’ and ‘kitchen_racks’ respectively. There are no rows with defined English category name but undefined Portuguese category name. After this, we just drop the Portuguese column and rename the English one to ‘product_category’.

There are still rows with NaN-valued product_category, so we just impute these with ‘unknown’ category.

It turns out that there are 74 unique values for product_category, so we need to shrink them. In the end, we managed to shrink these categories into 16 categories.

```
product_2017_copy['product_category'].unique()
```

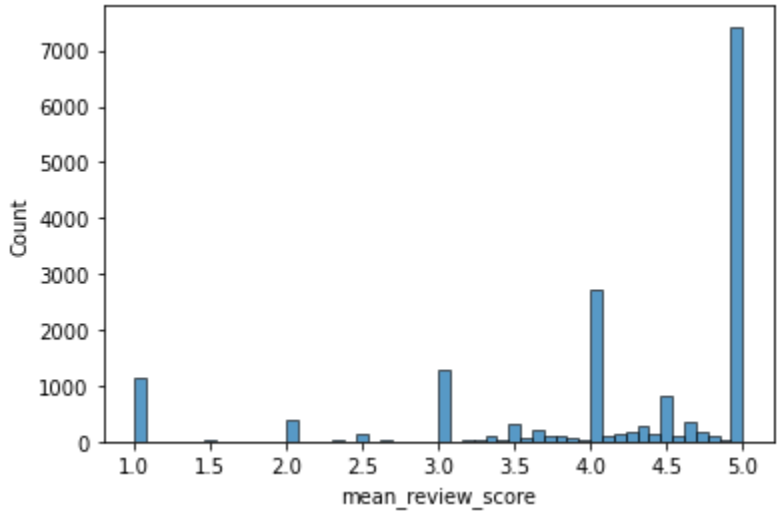
```
array(['cool_stuff', 'home, furniture & appliances',
       'toys, arts, craftmanship, stationery & accessories',
       'health, beauty & hygiene', 'PC, electronics & accessories',
       'auto', 'fashion', 'watches_gifts', 'construction & garden tools',
       'unknown', 'books', 'pet_shop', 'luggage_accessories',
       'industry & commerce', 'food & drink', 'security'], dtype=object)
```

```
product_2017_copy['product_category'].unique().shape
```

```
(16,)
```

Cleaning mean_review_score

First, let's look into the distribution of mean_review_score.



As seen above, there are spikes on the integer values. This is not due to chance, but rather due to the fact that the majority (56,67%) of sold products are only sold once with quantity 1 in 2017. For sold products with null-valued mean_review score, we just impute them with the median. For unsold ones, we just leave them NaN-valued.

Data Manipulation

Now, we are going to aggregate our data table based on their product_category. First, we are going to create a new column 'unsold' with value 1 if the product is unsold, 0 otherwise. Then, the table is aggregated as follows:

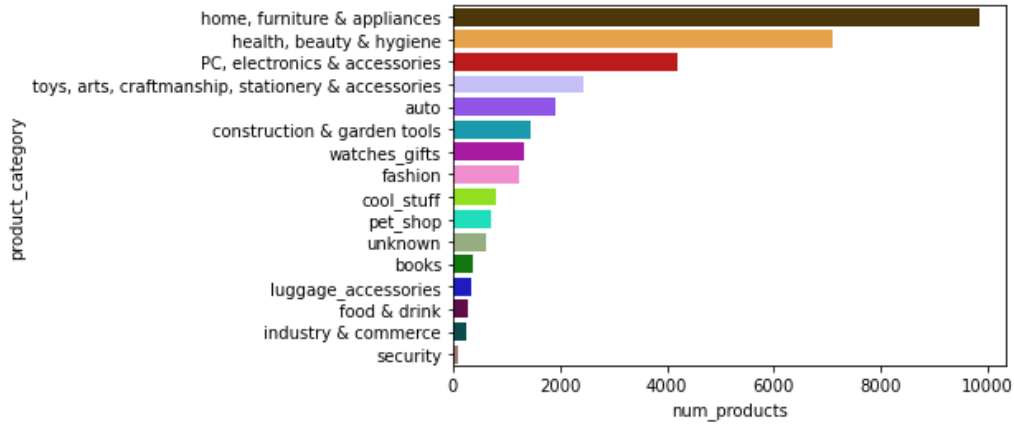
- product_id: count (to produce the number of products in the category)
- unsold: mean (to produce the proportion of unsold products in the category)
- sales: sum (to produce the total sales in the category)
- total_sold: sum (to produce the total of items sold in the category)

- num_orders: sum (to produce the total of orders with products in the category)
- num_times_more_than_one: sum (to produce the total of orders with bundles of products in the category)
- num_uniq_customers: mean (to produce the average of customers for products in the category)
- mean_review_score: median (to produce the median of mean_review_score of products in the category)

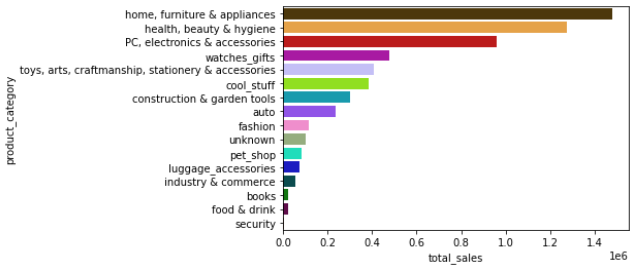
Then we rename the columns to be representative of the aggregating functions used.
Note: we do not use median aggregating function for total_sold, num_orders, num_times_more_than_one, and num_uniq_customers as 44% of products are unsold & 56,67% of SOLD products are only sold once in quantity 1. If we use median, whether on the whole products or only sold products, the values would be the same across (almost) all categories.

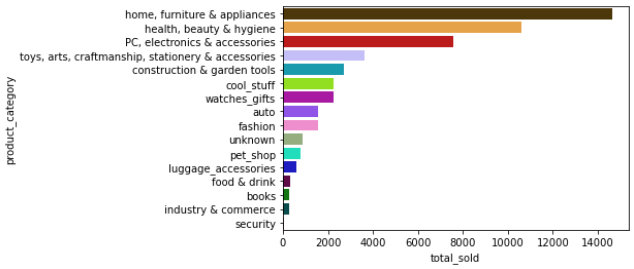
Exhibits

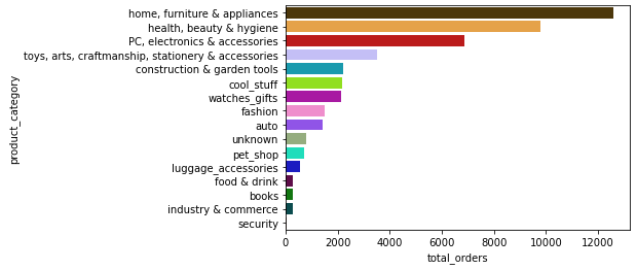
Exhibit 1: The top 3 categories with the largest number of products (and all of the totals) are ‘home, furniture & appliances’, ‘health, beauty & hygiene’ and ‘PC, electronics & accessories’



The relative/proportional distribution of totals (whether total_sales, total_sold, total_orders, or total_times_more_than_one) are almost the same as the distribution of total num_products. This makes sense as categories with large number of products are more likely to have large totals (whether total_sales, total_sold, total_orders, or total_times_more_than_one).







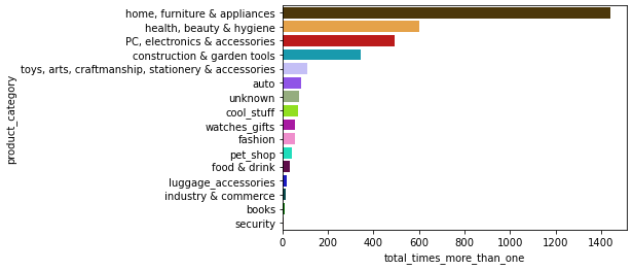
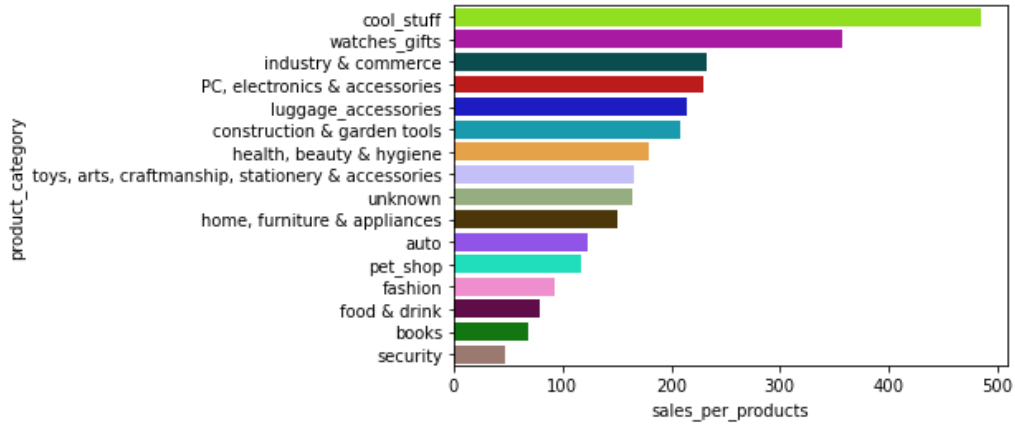


Exhibit 2: The top 3 categories with the largest average sales (whether per number of orders or products) are ‘cool_stuff’, ‘watches_gifts’, and ‘industry & commerce’

For average sales per number of products, the top 3 categories in descending order are ‘cool_stuff’, ‘watches_gifts’, and ‘industry & commerce’, while for average sales per orders, the positions of ‘cool_stuff’ and ‘industry & commerce’ are switched. This is due to the fact that products in these categories are more expensive compared to other categories.



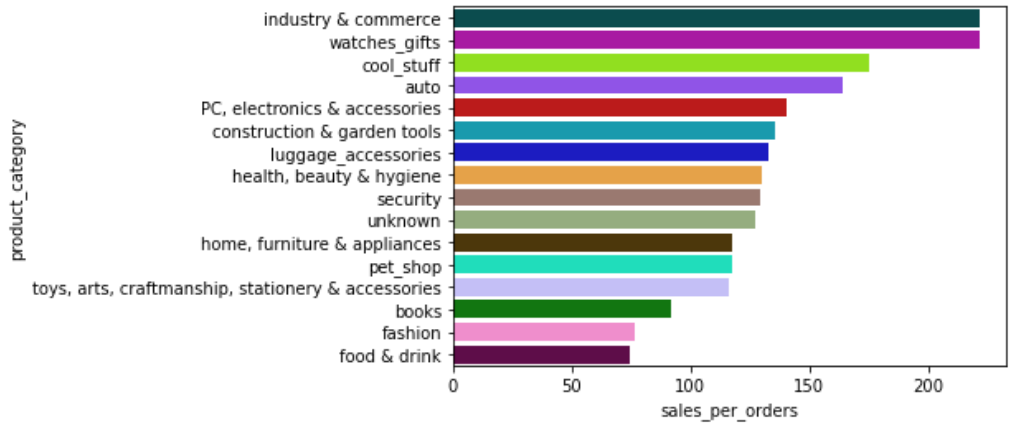


Exhibit 3: The top 3 products ordered in bundles (more than 1 in quantity per order) are ‘construction & garden tools’, ‘security’, and ‘home, furniture & appliances’.

This might be due to the fact that for these categories, customers are more likely to buy in bulk (e.g. it is more likely to buy more than 1 tiling, water pipe, fertilizer, kitchen appliance, or small cabinet in one order).

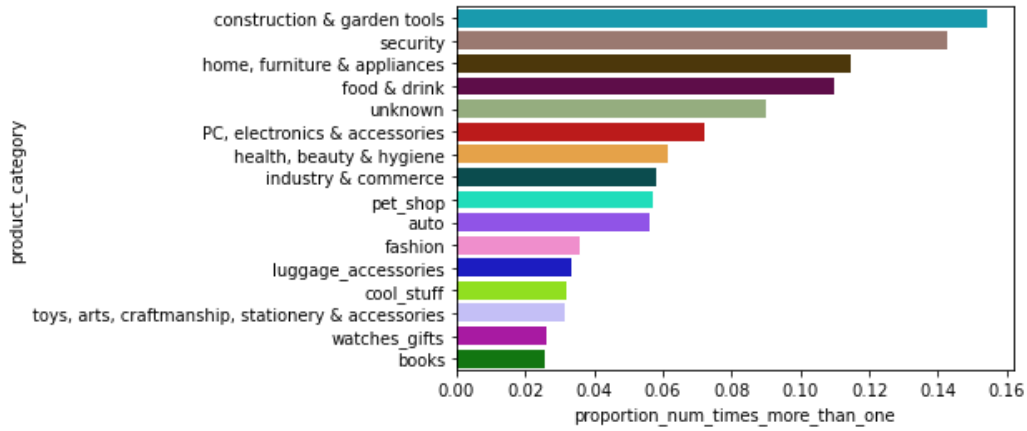


Exhibit 4: The top category with the largest average of customers is ‘cool_stuff’

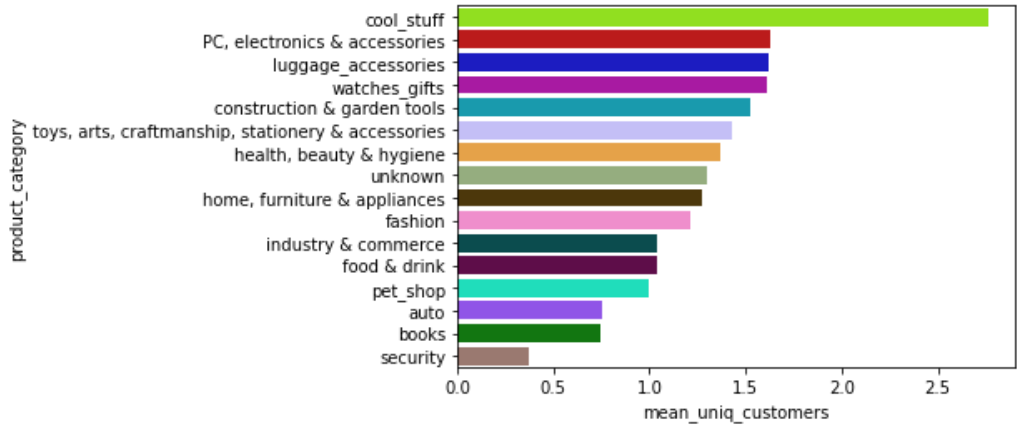


Exhibit 5: The top category with the largest share of unsold products (almost 80%) is ‘security’

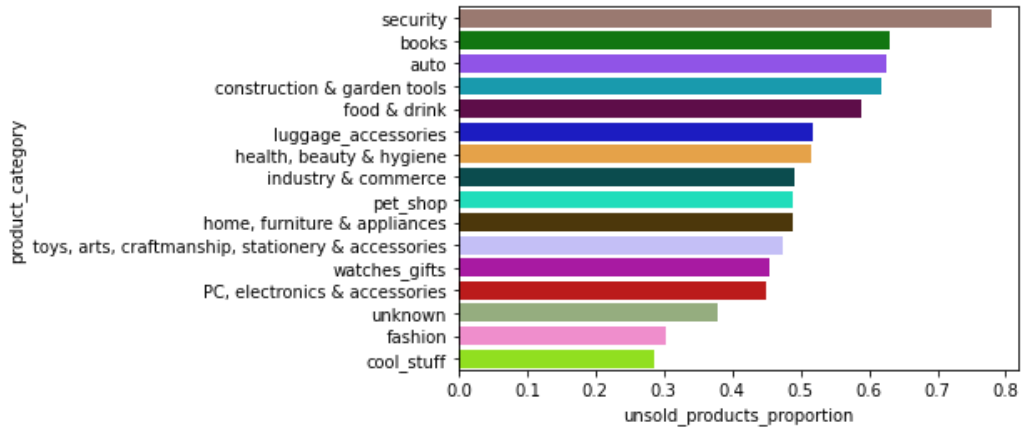
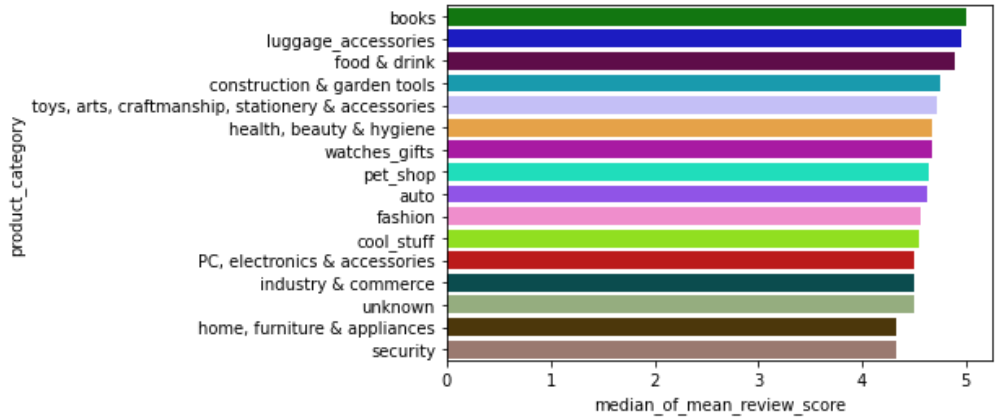


Exhibit 6: The top 3 categories with the highest median of mean_review_score are ‘books’, ‘luggage_accessories’ and ‘food & drink’



Conclusion & Recommendation

- As the top 3 categories with the largest number of products are cool_stuff’, ‘watches_gifts’, and ‘industry & commerce’ (and they brought the largest total of sales), this e-commerce should keep the sales in these categories in the same level (or slightly better).
- As the category ‘cool_stuff’ brought one of the highest sales average (whether per number of products or orders), the largest mean of customers number, and the lowest proportion of unsold products, this e-commerce should try to increase the sales in this category to maximize incomes (whether by advertisements or promos).
- As the categories ‘construction & garden tools’ and ‘home, furniture & appliances’ are in the top 3 for highest proportion of products often ordered in bundle, this e-commerce should encourage more bulk-buying for these two categories (whether by giving promos such as ‘buy 2 get 1’).
- The e-commerce should watch out for the products in the category ‘security’ as almost 80% of the products were unsold in 2017. This might mean that customers only prefer certain kinds of products in this category.

Github Repository

[muhammadalifaqsha/productanalysis-wranglingproject-pacmann: Python notebook code for Pacmann Data Wrangling course project (github.com)]