Data Wrangling

Data acquisition:

Data has been provided and is taken from Kaggle competition at the below location: https://www.kaggle.com/mehdidag/black-friday.

There is a file 'BlackFriday' of size ~25 Mb containing shopping patterns of a store on black Friday with 537,577 records which is made available for the task at hand.

Data type and null values:

It can be inferred from the dataset that in the total of 12 columns, 2 columns have null values. Namely 'Product_Category_2' and 'Product_Category_3'. Also, it can be noted that these product categories have float64 as type compared to int64 of 'Product_Category_1' column.

```
# Lets see the data types and null values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 12 columns):
User_ID                     537577 non-null int64
Product_ID                  537577 non-null object
Gender                      537577 non-null object
Age                         537577 non-null object
Occupation                  537577 non-null int64
City_Category               537577 non-null object
Stay_In_Current_City_Years  537577 non-null object
Marital_Status              537577 non-null int64
Product_Category_1          537577 non-null int64
Product_Category_2          370591 non-null float64
Product_Category_3          164278 non-null float64
Purchase                    537577 non-null int64
dtypes: float64(2), int64(5), object(5)
memory usage: 49.2+ MB
```

1. First, let's handle the null values. Upon checking the null columns, it is found that none of them have the value 0 as product category. So, a decision to replace the null values to 0 is taken.
2. Next, the columns of Product_Category_* are renamed for ease of understanding to ProdCat* and during the process, Product_Category_2 and Product_Category_3 are converted from float64 to int64 type. The original Product_Category_* columns are dropped.
3. Next we checked if column User_ID represents a unique person by grouping by User_ID and Gender. The count of the unique User_ID was compared to the group by count and it matched. It can therefore be concluded that User_ID is associated with a unique person.
4. We can now check if the Product_ID – User_ID is a unique combination by same means. It can be seen that group by count also matches with the total number or records. It can therefore be concluded that Product_ID – User_ID is a unique key for the dataset.

5. We can now investigate if the product combination is related to a unique product ID. We do a group by of the ProdCat* columns and compare the counts to that of Product_ID. They do not match. Therefore, Product_ID does not seem to be directly related to the combination of categories.
6. The above point can be expanded to the ordering of ProdCat* columns can be permanently captured in a separate called 'ProdCombo'. This order is compared to that of the Product_ID and again there doesn't seem to be any specific relation. It can also be said that the same combo of records could exist with a different Product_ID.
7. Lets check out some rows and confirm this. It can be seen that for User_ID 1001015, the same ProdCombo has 3 different Product_IDs.

Final state of cleaned columns:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 13 columns):
User_ID                     537577 non-null int64
Product_ID                  537577 non-null object
Gender                      537577 non-null object
Age                         537577 non-null object
Occupation                  537577 non-null int64
City_Category               537577 non-null object
Stay_In_Current_City_Years  537577 non-null object
Marital_Status              537577 non-null int64
Purchase                    537577 non-null int64
ProdCat1                    537577 non-null int64
ProdCat2                    537577 non-null int32
ProdCat3                    537577 non-null int32
ProdCombo                   537577 non-null object
dtypes: int32(2), int64(5), object(6)
memory usage: 49.2+ MB
```