In [5]:

```
import numpy as np
import pandas as pd
```

In [7]:

```
transactions = pd.read_csv("D:/Analyst/Python/NumPy_&_Python/Pandas_Course_Res
                           dtype= {"DAY": "Int16",
                                   "QUANTITY": "Int32",
                                   "STORE_ID": "Int32",
                                   "WEEK_NO": "Int8"}
                           )
```

In [8]:

```
transactions.head()
```

Out[8]:

	household_key	BASKET_ID	DAY	PRODUCT_ID	QUANTITY	SALES_VALUE	ST
0	1364	26984896261	1	842930	1	2.19	
1	1364	26984896261	1	897044	1	2.99	
2	1364	26984896261	1	920955	1	3.09	
3	1364	26984896261	1	937406	1	2.50	
4	1364	26984896261	1	981760	1	0.60	

In [11]:

```
transactions.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2146311 entries, 0 to 2146310
Data columns (total 11 columns):
 #   Column             Dtype  
---  ------             -----  
 0   household_key      int64  
 1   BASKET_ID          int64  
 2   DAY                Int16  
 3   PRODUCT_ID         int64  
 4   QUANTITY           Int32  
 5   SALES_VALUE        float64
 6   STORE_ID           Int32  
 7   RETAIL_DISC        float64
 8   WEEK_NO            Int8   
 9   COUPON_DISC        float64
 10  COUPON_MATCH_DISC  float64
dtypes: Int16(1), Int32(2), Int8(1), float64(4), int64(3)
memory usage: 145.3 MB
```

In [15]:

```
transactions.describe().round()
```

Out[15]:

	household_key	BASKET_ID	DAY	PRODUCT_ID	QUANTITY	SALES
count	2146311.0	2.146311e+06	2146311.0	2146311.0	2146311.0	21
mean	1056.0	3.404897e+10	390.0	2884715.0	101.0	
std	605.0	4.723748e+09	190.0	3831949.0	1152.0	
min	1.0	2.698490e+10	1.0	25671.0	0.0	
25%	548.0	3.040798e+10	229.0	917231.0	1.0	
50%	1042.0	3.281176e+10	392.0	1027960.0	1.0	
75%	1581.0	4.012804e+10	555.0	1132771.0	1.0	
max	2099.0	4.230536e+10	711.0	18316298.0	89638.0	

In [19]:

```
transactions.isna().sum()
```

Out[19]:

```
household_key        0
BASKET_ID            0
DAY                  0
PRODUCT_ID           0
QUANTITY             0
SALES_VALUE          0
STORE_ID             0
RETAIL_DISC          0
WEEK_NO              0
COUPON_DISC          0
COUPON_MATCH_DISC    0
dtype: int64
```

In [33]:

```
transactions["household_key"].nunique()
```

Out[33]:

```
2099
```

In [35]:

```
transactions["PRODUCT_ID"].nunique()
```

Out[35]:

```
84138
```

In [37]:

```
# Create a discount sum column and a percentage discount column

transactions = (
    transactions
    .assign(total_discount = transactions["RETAIL_DISC"] + transactions["COUPO
           percentage_discount = (lambda x: (x["total_discount"] / x["SALES_VA
    .drop(["RETAIL_DISC", "COUPON_DISC", "COUPON_MATCH_DISC"], axis=1)
)

transactions["percentage_discount"] = (transactions["percentage_discount"]
                                        .where(transactions["percentage_discour
                                        .where(transactions["percentage_discour
                                       )
```

```
transactions.head()
```

Out[37]:

	household_key	BASKET_ID	DAY	PRODUCT_ID	QUANTITY	SALES_VALUE	ST
0	1364	26984896261	1	842930	1	2.19	
1	1364	26984896261	1	897044	1	2.99	
2	1364	26984896261	1	920955	1	3.09	
3	1364	26984896261	1	937406	1	2.50	
4	1364	26984896261	1	981760	1	0.60	

Overall Statistics

```
In [40]: transactions["SALES_VALUE"].sum()

Out[40]: 6666243.499999999
```

```
In [44]: transactions["total_discount"].sum()

Out[44]: -1178658.0799999998
```

```
In [56]: transactions["total_discount"].sum() / transactions['SALES_VALUE'].sum()

Out[56]: -0.1768099350106248
```

```
In [62]: transactions['percentage_discount'].mean()

Out[62]: 0.2073244407398103
```

```
In [58]: transactions['QUANTITY'].sum()

Out[58]: 216713611
```

```
In [60]: transactions['QUANTITY'].max()

Out[60]: 89638
```

```
In [66]: # Use to grab row with value - discount rate is lower than average

         transactions.loc[transactions['QUANTITY'].argmax()]
```

```
Out[66]: household_key                    630.0
         BASKET_ID              34749153595.0
         DAY                              503.0
         PRODUCT_ID                   6534178.0
         QUANTITY                       89638.0
         SALES_VALUE                      250.0
         STORE_ID                         384.0
         WEEK_NO                           73.0
         total_discount                  -13.45
         percentage_discount             0.0538
         Name: 1442095, dtype: Float64
```

```
In [70]: transactions['BASKET_ID'].nunique()
```

```
Out[70]: 232939
```

```
In [68]: # Sales value per transaction/basket

         transactions['SALES_VALUE'].sum() / transactions['BASKET_ID'].nunique()
```

```
Out[68]: 28.61797938516092
```

```
In [72]: transactions['household_key'].nunique()
```

```
Out[72]: 2099
```

```
In [74]: # Sales value per household

         transactions['SALES_VALUE'].sum() / transactions['household_key'].nunique()
```

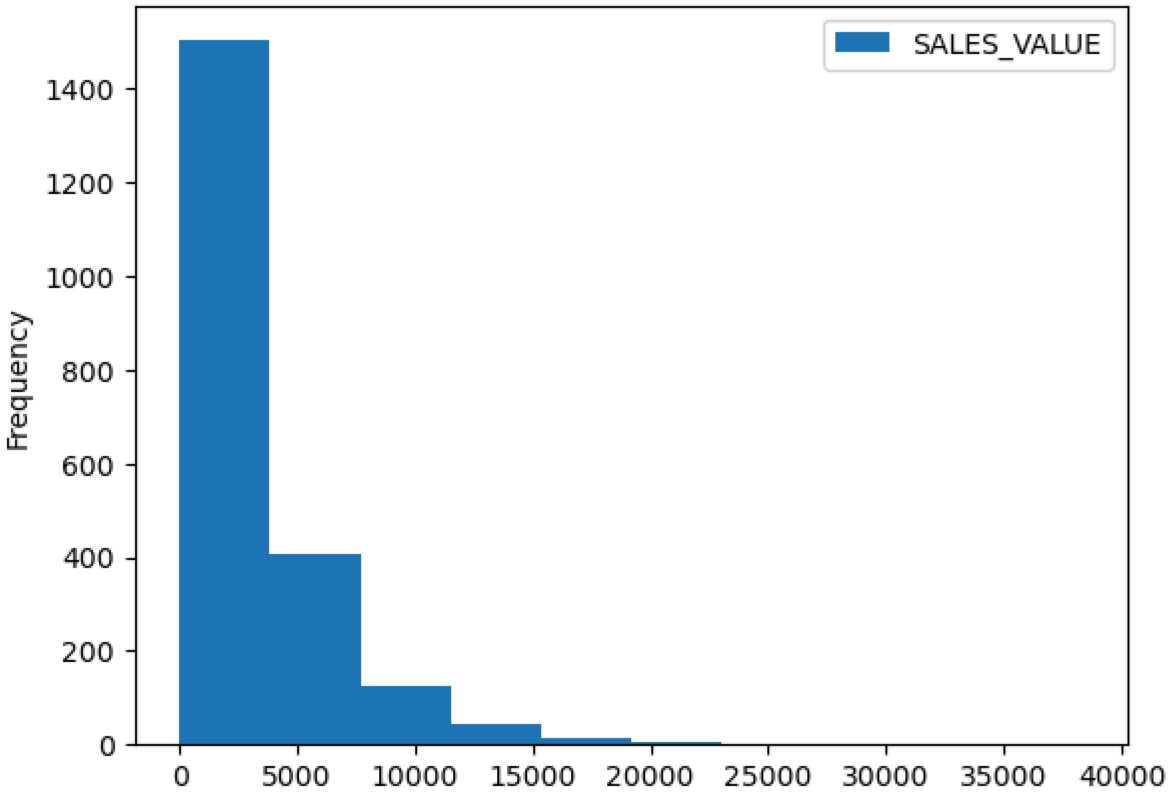
```
Out[74]: 3175.9140066698424
```

Household Analysis

```
In [94]: # plot distribution of households by total sales value
         # First groupby household and calculate sum of sales
         # then plot with a histogram

         (transactions
          .groupby("household_key")
          .agg({'SALES_VALUE':'sum'})
          # .sort_values(by='SALES_VALUE', ascending=False)
          # .head()
          .plot.hist()
         )
```

```
Out[94]: <Axes: ylabel='Frequency'>
```



In [104…

```
# store top 10 households by total value and quantity
# groupby household_key, calculate sum of relevant columns by household
# sort both by relevant metric in descending order, and grab top 10

top10_value = (transactions
               .groupby('household_key')
               .agg({'SALES_VALUE':'sum'})
               .sort_values('SALES_VALUE', ascending = False)
               .iloc[:10]
              )
top10_quant = (transactions
               .groupby('household_key')
               .agg({'QUANTITY': 'sum'})
               .sort_values('QUANTITY', ascending = False)
               .iloc[:10]
              )
```

In [108…

```
top10_value
```

Out[108…

	SALES_VALUE
household_key	
1023	38319.79
1609	27859.68
1453	21661.29
1430	20352.99
718	19299.86
707	19194.42
1653	19153.75
1111	18894.72
982	18790.34
400	18494.14

In [110…

```
top10_quant
```

Out[110…

	QUANTITY
household_key	
1023	4479917
755	3141769
1609	2146715
13	1863829
1430	1741892
1527	1734632
1762	1669880
707	1640193
1029	1496204
1314	1492863

In [114…

```
# Use multiple aggregation to create both in a single table an option
# this here is just to use to compare to chart

(transactions
 .groupby('household_key')
 .agg({'SALES_VALUE': 'sum', 'QUANTITY':'sum'})
 .sort_values('SALES_VALUE', ascending = False)
 .loc[:, "SALES_VALUE"]
 .describe()
)
```

Out[114…

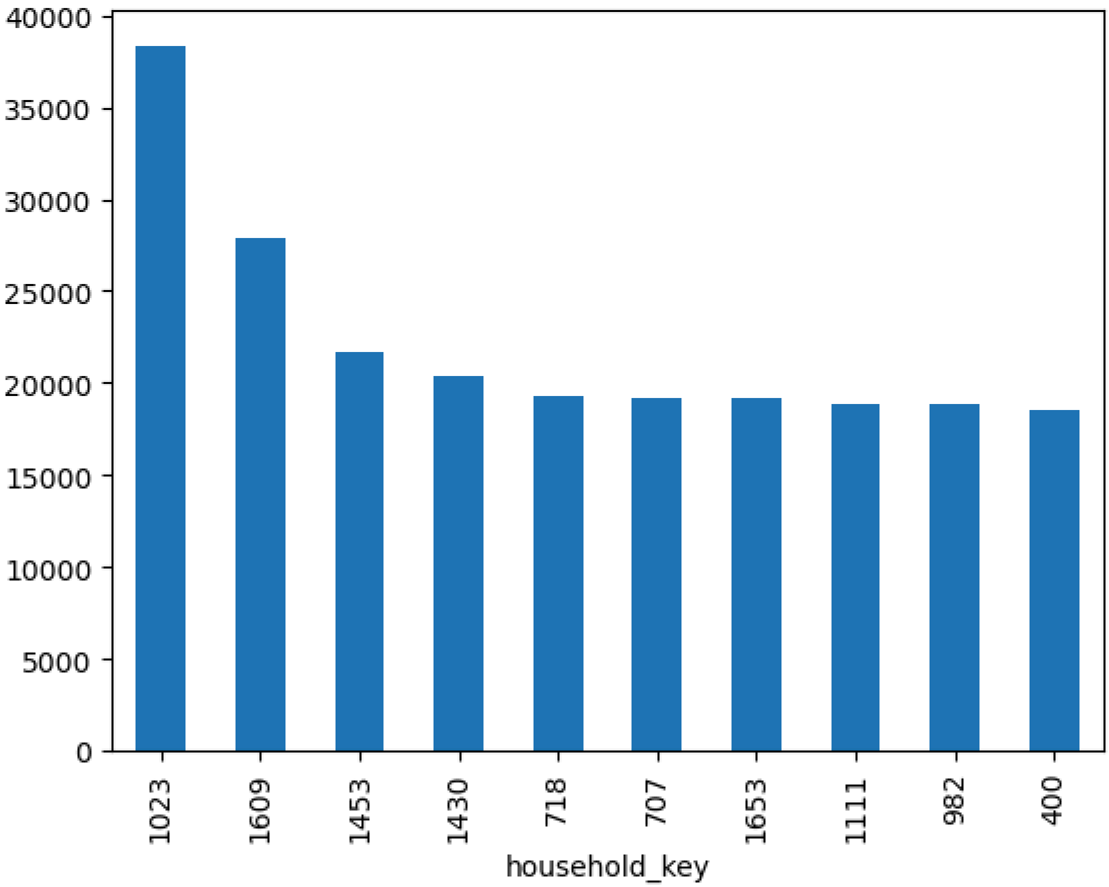
```
count     2099.000000
mean      3175.914007
std       3287.043772
min          8.170000
25%        971.035000
50%       2145.710000
75%       4295.395000
max      38319.790000
Name: SALES_VALUE, dtype: float64
```

In [118…

```
top10_value['SALES_VALUE'].plot.bar()
```

Out[118…

```
<Axes: xlabel='household_key'>
```



Product Analysis

In [125…

```
# Create top 10 products by sales df
# group by PRODUCT_ID and sum sales value by product
# Sort in descending order and grab top 10 rows

top10_products = (transactions
                  .groupby(["PRODUCT_ID"])
```

```
                    .agg({'SALES_VALUE': 'sum'})
                    .sort_values("SALES_VALUE", ascending = False)
                    .iloc[:10]
)
```

In [127…

```
top10_products
```

Out[127…

	SALES_VALUE
PRODUCT_ID	
6534178	420154.13
6533889	42339.31
1029743	33894.75
1082185	24149.79
6533765	23831.14
6534166	23755.70
1106523	22931.01
916122	22749.02
995242	21229.72
5569230	20051.95

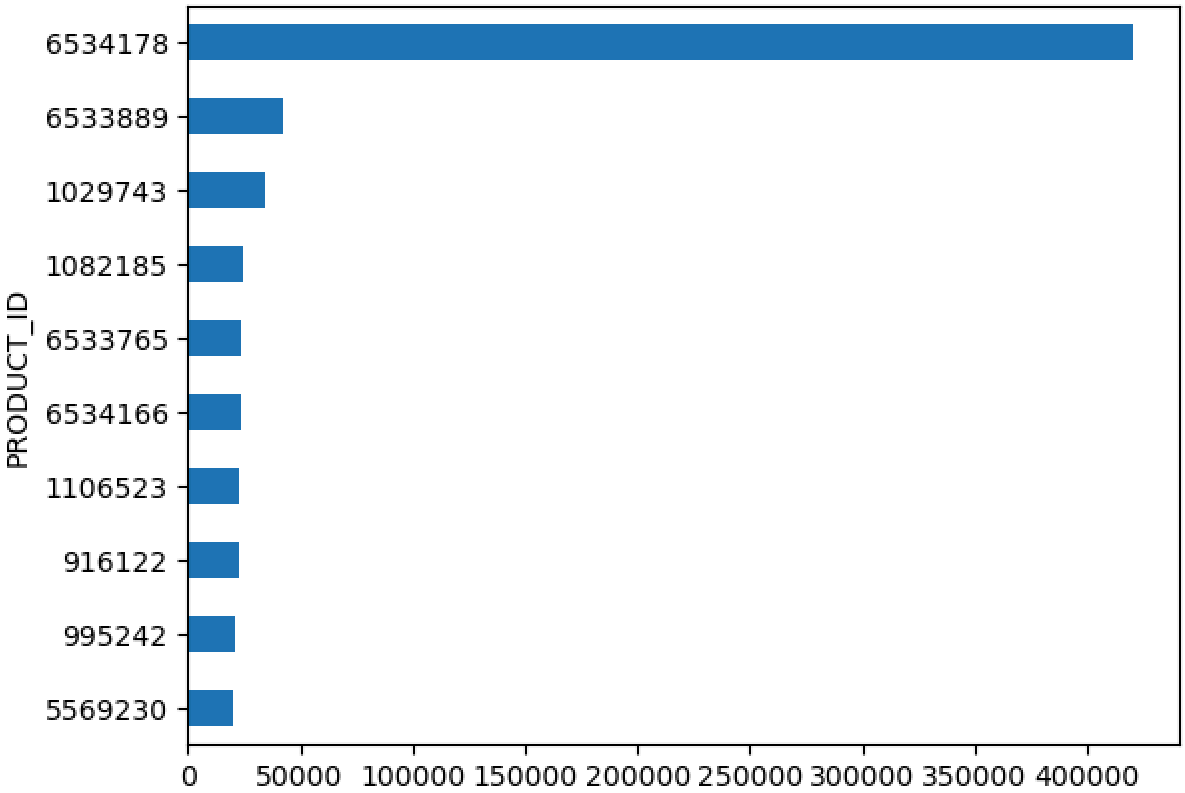
In [181…

```
# Plot top 10 products by sale values

top10_products["SALES_VALUE"].sort_values().plot.barh()
```

Out[181…

```
<Axes: ylabel='PRODUCT_ID'>
```



In [183…

```
# Calculate the total discount for top 10 products
# Divide that by sales value for top 10 products

((transactions
 .query("PRODUCT_ID in @top10_products.index")
 .loc[:, "total_discount"]
 .sum()
)
/
(transactions
 .query("PRODUCT_ID in @top10_products.index")
 .loc[:, "SALES_VALUE"]
 .sum()
))
```

Out[183…

```
-0.10331267387397927
```

In [185…

```
# read in oriducts data

products = pd.read_csv("D:/Analyst/Python/NumPy_&_Python/Pandas_Course_Resourc

products.head()
```

Out[185…

	PRODUCT_ID	MANUFACTURER	DEPARTMENT	BRAND	COMMODITY_DESC	SU
0	25671	2	GROCERY	National	FRZN ICE	
1	26081	2	MISC. TRANS.	National	NO COMMODITY DESCRIPTION	
2	26093	69	PASTRY	Private	BREAD	E
3	26190	69	GROCERY	Private	FRUIT - SHELF STABLE	
4	26355	69	GROCERY	Private	COOKIES/CONES	

In [199…

```
top10_value
```

Out[199…

	SALES_VALUE
household_key	
1023	38319.79
1609	27859.68
1453	21661.29
1430	20352.99
718	19299.86
707	19194.42
1653	19153.75
1111	18894.72
982	18790.34
400	18494.14

In [219…

```
# Look up top 10 products for households in top10_value table
# Use query tpp reference index of top10_value to filter to relecant householc
# Use value counts to get counts by product_id (this will be order in descendi
# Then grab the top 10 pproducts with iloc and extract the index to get produc

top_hh_products = (transactions
                  .query("household_key in @top10_value.index")
                   .loc[:, "PRODUCT_ID"]
                   .value_counts()
                   .iloc[:10]
                   .index
                  )
```

In [223…

```
top_hh_products
```

Out[223…

```
Index([1082185, 1029743, 6534178, 6533889, 1127831,  951590,  860776, 110652
3,
        981760, 9677202],
      dtype='int64', name='PRODUCT_ID')
```

In [227…

```
# Filter product table to products from prior cell

products.query("PRODUCT_ID in @top_hh_products")
```

Out[227…

	PRODUCT_ID	MANUFACTURER	DEPARTMENT	BRAND	COMMODITY_DESC
10630	860776	2	PRODUCE	National	VEGETABLES - ALL OTHERS
20973	951590	910	GROCERY	National	BAKED BREAD/ BUNS/ROLLS
24250	981760	69	GROCERY	Private	EGGS
29657	1029743	69	GROCERY	Private	FLUID MILK PRODUCTS
35576	1082185	2	PRODUCE	National	TROPICAL FRUIT
38262	1106523	69	GROCERY	Private	FLUID MILK PRODUCTS
40600	1127831	5937	PRODUCE	National	BERRIES
57181	6533889	69	MISC SALES TRAN	Private	COUPON/MISC ITEMS
57221	6534178	69	KIOSK-GAS	Private	COUPON/MISC ITEMS
68952	9677202	69	GROCERY	Private	PAPER TOWELS

In [229…

```
# Product with highest quantity in a single row

products.query("PRODUCT_ID == 6534178")
```

Out[229…

	PRODUCT_ID	MANUFACTURER	DEPARTMENT	BRAND	COMMODITY_DESC
57221	6534178	69	KIOSK-GAS	Private	COUPON/MISC ITEMS

In [231…

```
# Look up 10 product names for all customers (from first cell)

products.query("PRODUCT_ID in @top10_products.index")
```

Out[231…

	PRODUCT_ID	MANUFACTURER	DEPARTMENT	BRAND	COMMODITY_DESC
16863	916122	4314	MEAT	National	CHICKEN
25754	995242	69	GROCERY	Private	FLUID MILK PRODUCTS
29657	1029743	69	GROCERY	Private	FLUID MILK PRODUCTS
35576	1082185	2	PRODUCE	National	TROPICAL FRUIT
38262	1106523	69	GROCERY	Private	FLUID MILK PRODUCTS
53097	5569230	1208	GROCERY	National	SOFT DRINKS
57171	6533765	69	KIOSK-GAS	Private	FUEL
57181	6533889	69	MISC SALES TRAN	Private	COUPON/MISC ITEMS
57216	6534166	69	MISC SALES TRAN	Private	COUPON/MISC ITEMS
57221	6534178	69	KIOSK-GAS	Private	COUPON/MISC ITEMS

In [243…

```
top10_products.index
```

Out[243…

```
Index([6534178, 6533889, 1029743, 1082185, 6533765, 6534166, 1106523,  91612
2,
        995242, 5569230],
      dtype='int64', name='PRODUCT_ID')
```