Heroes Of Pymoli Data Analysis[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Heroes-Of-Pymoli-Data-Analysis)

Of the 1163 active players, the vast majority are male (84%). There also exists, a smaller, but notable proportion of female players (14%).

Our peak age demographic falls between 20-24 (44.8%) with secondary groups falling between 15-19 (18.60%) and 25-29 (13.4%).

Note[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Note)

Instructions have been included for each segment. You do not have to follow them exactly, but they are included to help you think through the steps.

Player Count[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Player-Count)

In [2]:

######################################################################

##### DIVYA TV #############################

## Import the csv file and initialize a dataframe.

## Perform mathematical operations on the columns.

######################################################################

### Import packages needed for the work. Pandas for dataframes.

import pandas as pd

import numpy as py

#Loading CSV file using pandas

file\_to\_load = "Resources/purchase\_data.csv"

#printing the path of the file

print ("path of the file", file\_to\_load)

# Read Purchasing File and store into Pandas data frame

purchase\_data = pd.read\_csv(file\_to\_load)

# Reading the file again to keep the original file to use when needed.

purchase\_data\_original = pd.read\_csv(file\_to\_load)

## Not printing now. Uncomment if you need to see the entire csv file.

#print(purchase\_data\_original)

path of the file Resources/purchase\_data.csv

Display the total number of players

In [5]:

# Display the total number of players. There are duplicate SNs- same people - not counting them.

total\_number\_of\_players=purchase\_data["SN"].nunique()

#print ("total number of players=", total\_number\_of\_players)

totalplayers=pd.DataFrame({'Total Players':[purchase\_data["SN"].nunique()]})

#print(totalplayers)

##Diaplay the total number of players.

totalplayers.style

Out[5]:

|  | **Total Players** |
| --- | --- |
| **0** | 576 |
|  |  |

Purchasing Analysis (Total)[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Purchasing-Analysis-(Total))

Run basic calculations to obtain number of unique items, average price, etc.

Create a summary data frame to hold the results

Optional: give the displayed data cleaner formatting

Display the summary data frame

In [6]:

### Perform mathematical operations.

### 1. Calculate number of unique Items.

### 2. Calcualte avarage price.

NumberofUniqueItems=purchase\_data["Item ID"].nunique()

#print ("total number of unique items=", NumberofUniqueItems)

NumberofPurchases=purchase\_data["Purchase ID"].nunique()

#print (" Number of Purcharses=", NumberofPurchases)

AveragePrice=float(round((purchase\_data["Price"].mean()), 2))

#print ("Average=", AveragePrice)

##Displaying dollars in correct format

AveragePrice1='${:,.2f}'.format(AveragePrice)

## Converting revenue from float to display in dollars.

Revenue= float(purchase\_data["Price"].sum())

Revenue1='${:,.2f}'.format(Revenue)

### Create a dictionary with calculations.

newdf={'Number of Unique Items':[NumberofUniqueItems],

'Average Price':[AveragePrice1],

'Number of Purchases':[NumberofPurchases],

'Total Revenue':[Revenue1]

}

## Initialize the dataframe from the dictionary above.

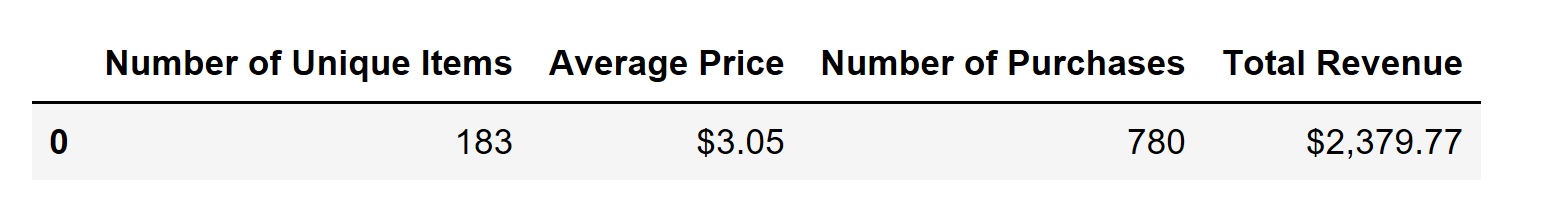
df=pd.DataFrame(newdf)

#print (df)

df.style

Out[6]:

|  | **Number of Unique Items** | **Average Price** | **Number of Purchases** | **Total Revenue** |
| --- | --- | --- | --- | --- |
| **0** | 183 | $3.05 | 780 | $2,379.77 |



Gender Demographics[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Gender-Demographics)

Percentage and Count of Male Players

Percentage and Count of Female Players

Percentage and Count of Other / Non-Disclosed

In [7]:

### \*\*\* Ignore the commented code. They are retained for furure reference. \*\*\* ###

##Genders=purchase\_data["Gender"].unique()

#print(Genders)

#GenderCounts=purchase\_data["Gender"].value\_counts()

#print (GenderCounts)

#print(purchase\_data)

## Create a data frame dropping all the duplicate SN

#newdataframe=purchase\_data.drop\_duplicates(subset='SN', keep='first', inplace=True)

#print(newdataframe)

#DataFrame.drop\_duplicates(subset=None, keep=’first’, inplace=False)

#################################################################################

## Calculate the following:

## 1. Number of Male, female, other players.

## 2. Calculate percentage gender wise

#################################################################################

Gender\_Count=purchase\_data.groupby("Gender")["SN"].nunique()

#print (newlist)

percentage=(purchase\_data.groupby("Gender")["SN"].nunique()/total\_number\_of\_players)\*100

#print (percentage)

## Merging the tables for gender and percentages into one.

merge\_table = pd.merge(Gender\_Count, round(percentage), on="Gender")

#print("merged table=", merge\_table)

## Renaming the columns to match the expected output.

renamed\_df=merge\_table.rename(columns = {'SN\_x':'Total Count','SN\_y':'Percentage of Players'}, inplace = False)

#print(renamed\_df)

#renamed\_df.index

#renamed\_df.reindex(['Male', 'Female', 'Other / Non-Disclosed'])

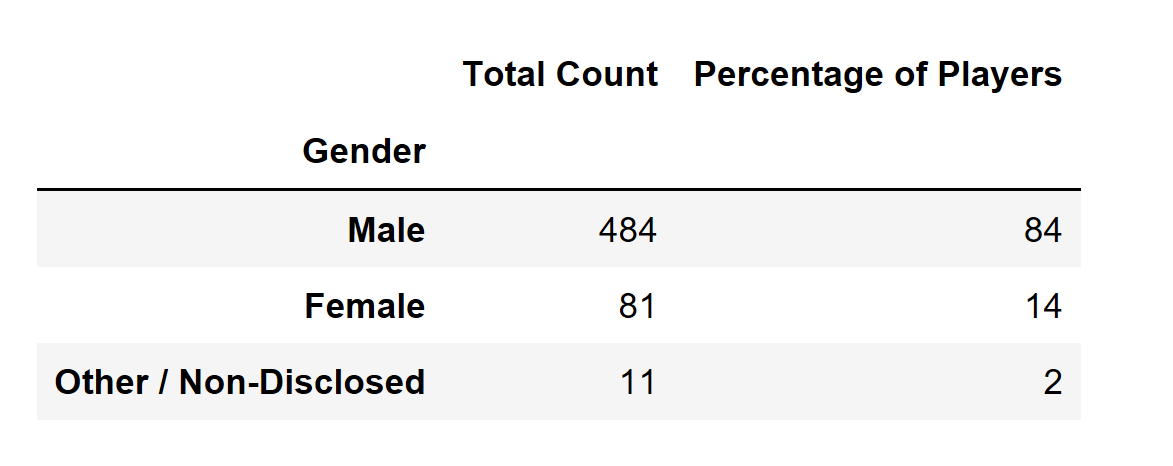
#Sorting the rows based on Total Count.

renamed\_df=renamed\_df.sort\_values(by='Total Count', ascending=0)

renamed\_df.style

Out[7]:

|  | **Total Count** | **Percentage of Players** |
| --- | --- | --- |
| **Gender** |  |  |
| **Male** | 484 | 84 |
| **Female** | 81 | 14 |
| **Other / Non-Disclosed** | 11 | 2 |



Purchasing Analysis (Gender)[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Purchasing-Analysis-(Gender))

Run basic calculations to obtain purchase count, avg. purchase price, avg. purchase total per person etc. by gender

Create a summary data frame to hold the results

Optional: give the displayed data cleaner formatting

Display the summary data frame

In [8]:

#####################################################################################

### Calculate -- BY GENDER

### 1. Purchase count

### 2. Average purchase price

### 3. Average purchase total per person etc. by gender

####################################################################################

NumberofUniqueItems=purchase\_data["Item ID"].nunique()

#print ("total number of unique items=", NumberofUniqueItems)

NumberofPurchases\_groupbyGender=purchase\_data.groupby("Gender")["Purchase ID"].nunique()

#print ("Purchase\_Count=", NumberofPurchases\_gender)

AveragePrice\_groupbyGender=purchase\_data.groupby("Gender")["Price"].mean()

#AveragePrice\_gender\_dollar='${:,.2f}'.format(float(AveragePrice\_gender))

#print(AveragePrice\_groupbyGender)

#print ("Average Purchase Price by gender", AveragePrice\_gender)

TotalPurchase\_groupbyGender=purchase\_data.groupby("Gender")["Price"].sum()

#print ("Total Purchase value=", TotalPurchase\_gender)

## Gender\_Count=purchase\_data.groupby("Gender")["SN"].nunique()

Average\_purchase\_per\_person=(TotalPurchase\_groupbyGender/Gender\_Count)

#Average\_purchase\_pp1='${:,.2f}'.format(Average\_purchase\_pp)

#print(Average\_purchase\_pp)

##'${:,.2f}'.format(AveragePrice)

### Creating a list from the calcualted values.

Summary\_list=[NumberofPurchases\_groupbyGender, round(AveragePrice\_groupbyGender, 2),TotalPurchase\_groupbyGender, round(Average\_purchase\_per\_person, 2)]

#print(Summary\_list)

## Initializing a dataframe from the list created above.

PA\_Summary\_df=pd.DataFrame(Summary\_list)

#print(PA\_Summary\_df)

##Transpose the data - all rows to columns using numpy dot T

Summary\_table\_transposed=PA\_Summary\_df.T

##\*\*\* Igonore the commented code - they were used during first round of work. \*\*\* ###

#df=Summary\_table\_transposed - testing the rename of columns with same names.

#df = df\_column\_uniquify(df) -error- 'Summary\_table\_transposed\_column\_uniquify' is not defined - didnt work

#Summary\_table\_transposed=Summary\_table\_transposed\_column\_uniquify(Summary\_table\_transposed)

#df.rename(columns={ df.columns[1]: "new\_col\_name" }) - didnt work

#Summary\_table\_transposed=Summary\_table\_transposed.rename(columns={Summary\_table\_transposed.columns[1]:"Divya"})

##df.columns.values[index] = "New name" - Works!!

#df.columns.values[1] = "New name"

#df.head()

## Renamig the columns index wise.

Summary\_table\_transposed.columns.values[1]="Average Purchase Price"

Summary\_table\_transposed.columns.values[0]="Purchase Count"

Summary\_table\_transposed.columns.values[2]="Total Purchase Price"

Summary\_table\_transposed.columns.values[3]="Avg total Purchase Per Person"

## Dataframe with same columns names cause problems.

#Summary\_table\_transposed=Summary\_table\_transposed.rename(index=1, columns = {"Purchase ID":"Purchase Count","Price":"Average Purchase Price",

#"Price":"Total Purchase Price", "Unnamed 0":"Avg total Purchase per person"}, inplace = False)

#Summary\_table\_transposed.columns

#print(Summary\_table\_transposed)

#Summary\_table\_transposed.head()

###################################################################################

### \*\*\*\*\*\*\*\* Fuction to display prices in dollars. \*\*\*\*\*\*\*

###################################################################################

def displaydollar(list):

newlist=[]

for value in list:

x=float(value)

dollar= '${:,.2f}'.format(x)

newlist.append(dollar)

#print(newlist)

return(newlist)

#pdToList = list(dataPd['2']) - convert to list and call function displaydollar to display currency in dollars.

AveragePurchasePrice=list(Summary\_table\_transposed['Average Purchase Price'])

TotalPurchasePrice=list(Summary\_table\_transposed['Total Purchase Price'])

AvgtotalPurchasePerPerson=list( Summary\_table\_transposed['Avg total Purchase Per Person'])

## \*\*\* Function calls \*\*\* ##

## Function calls to display currencies in dollars and assign them back to the dataframe

Summary\_table\_transposed['Average Purchase Price']=displaydollar(AveragePurchasePrice)

Summary\_table\_transposed['Total Purchase Price'] =displaydollar(TotalPurchasePrice)

Summary\_table\_transposed['Avg total Purchase Per Person']=displaydollar(AvgtotalPurchasePerPerson)

## Display dataframe

Summary\_table\_transposed

Out[8]:

|  | **Purchase Count** | **Average Purchase Price** | **Total Purchase Price** | **Avg total Purchase Per Person** |
| --- | --- | --- | --- | --- |
| **Gender** |  |  |  |  |
| **Female** | 113.0 | $3.20 | $361.94 | $4.47 |
| **Male** | 652.0 | $3.02 | $1,967.64 | $4.07 |
| **Other / Non-Disclosed** | 15.0 | $3.35 | $50.19 | $4.56 |

Age Demographics[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Age-Demographics)

Establish bins for ages

Categorize the existing players using the age bins. Hint: use pd.cut()

Calculate the numbers and percentages by age group

Create a summary data frame to hold the results

Optional: round the percentage column to two decimal points

Display Age Demographics Table

In [9]:

##################################################################################

###### Calculate age demographics #################

##################################################################################

##Dropping duplicate entries of SN to help with the calculations.

purchase\_data = purchase\_data.drop\_duplicates(subset='SN', keep='first')

# create bins and group for lables for the bins.

bins=[0, 9, 14, 19, 24, 29, 34, 39, 120]

group=['<10', '10-14','15-19', '20-24', '25-29', '30-34', '35-39', '40+' ]

## \*\*\* pd.cut - at Age and assign bins and labels

newdf = pd.cut(purchase\_data["Age"], bins, labels=group)

# Place the data series into a new column inside of the DataFrame

purchase\_data["Agebin"] = pd.cut(purchase\_data["Age"], bins, labels=group)

# Calculate total players and percentages.

sumtable=purchase\_data.groupby("Agebin")["SN"].nunique()

sumtabledf=pd.DataFrame(sumtable)

sumtabledf=sumtabledf.rename(columns={'SN': 'Total Players'})

sumtabledf["Percentage players"]=round((sumtabledf['Total Players']/purchase\_data["SN"].nunique())\*100, 2)

#print(sumtabledf)

### Display the dataframe calculated.

sumtabledf.style

Out[9]:

|  | **Total Players** | **Percentage players** |
| --- | --- | --- |
| **Agebin** |  |  |
| **<10** | 17 | 2.95 |
| **10-14** | 22 | 3.82 |
| **15-19** | 107 | 18.58 |
| **20-24** | 258 | 44.79 |
| **25-29** | 77 | 13.37 |
| **30-34** | 52 | 9.03 |
| **35-39** | 31 | 5.38 |
| **40+** | 12 | 2.08 |

Purchasing Analysis (Age)[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Purchasing-Analysis-(Age))

Bin the purchase\_data data frame by age

Run basic calculations to obtain purchase count, avg. purchase price, avg. purchase total per person etc. in the table below

Create a summary data frame to hold the results

Optional: give the displayed data cleaner formatting

Display the summary data frame

In [10]:

###################################################################################################

#### Purchasing Analysis.

#### Calculate Purchase count, Average purchase price, Average Purchase total per person by age.

###################################################################################################

#Create bins and lables for binning.

purchase\_bin=[0, 9, 14, 19, 24, 29, 34, 39, 120]

purchase\_lables=['<10', '10-14','15-19', '20-24', '25-29', '30-34', '35-39', '40+' ]

#pd.cut at Age to get data sorted into bins.

#pandas.cut(x, bins, right=True, labels=None, retbins=False, precision=3, include\_lowest=False, duplicates='raise')

purchase\_summary\_table=pd.cut(purchase\_data\_original['Age'], bins=purchase\_bin, labels=purchase\_lables)

#Calculate Purchase count.

purchase\_data\_original['age label']=purchase\_summary\_table

purchase\_data\_summary=purchase\_data\_original.groupby('age label')["Purchase ID"].nunique()

## Initialize the dataframe with the calculations.

Purchase\_data\_summarydf=pd.DataFrame(purchase\_data\_summary)

#Rename Purchase ID column to Purchase count.

Purchase\_data\_summarydf=Purchase\_data\_summarydf.rename(columns={"Purchase ID": "Purchase Count" })

## Calculate Average Purchase price

Purchase\_data\_summarydf['Average Purchase Price']=displaydollar(round(purchase\_data\_original.groupby('age label')["Price"].mean(), 2))

## Calculate Total Purchase Value

Purchase\_data\_summarydf['Total Purchase Value']=displaydollar(purchase\_data\_original.groupby('age label')["Price"].sum())

#purchase\_data.groupby("Gender")["Price"].sum()

## Calculate Average Total Purchase per person.

Purchase\_data\_summarydf['Average Total Purchase per Person']=displaydollar(round(purchase\_data\_original.groupby('age label')["Price"].sum()/sumtabledf['Total Players'], 2))

## Move first row to last row to match the expected output.

#print(df.apply(np.roll, shift=1))

#df.reindex(np.roll(df.index, shift=1))

Purchase\_data\_summarydf=Purchase\_data\_summarydf.reindex(py.roll( Purchase\_data\_summarydf.index, shift=-1))

##Display the dataframe.

Purchase\_data\_summarydf

Out[10]:

|  | **Purchase Count** | **Average Purchase Price** | **Total Purchase Value** | **Average Total Purchase per Person** |
| --- | --- | --- | --- | --- |
| **age label** |  |  |  |  |
| **10-14** | 28 | $2.96 | $82.78 | $3.76 |
| **15-19** | 136 | $3.04 | $412.89 | $3.86 |
| **20-24** | 365 | $3.05 | $1,114.06 | $4.32 |
| **25-29** | 101 | $2.90 | $293.00 | $3.81 |
| **30-34** | 73 | $2.93 | $214.00 | $4.12 |
| **35-39** | 41 | $3.60 | $147.67 | $4.76 |
| **40+** | 13 | $2.94 | $38.24 | $3.19 |
| **<10** | 23 | $3.35 | $77.13 | $4.54 |

Top Spenders[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Top-Spenders)

Run basic calculations to obtain the results in the table below

Create a summary data frame to hold the results

Sort the total purchase value column in descending order

Optional: give the displayed data cleaner formatting

Display a preview of the summary data frame

In [11]:

########################################################################################

#### Top Spenders ###############

#### Sorting columns ##############

########################################################################################

## Calcualte top spenders.

top\_spenders=pd.DataFrame(purchase\_data\_original.groupby('SN')['Purchase ID'].nunique())

#Rename the column to match the expected output

top\_spenders.rename(columns={"Purchase ID":"Purchase Count"},inplace = True)

## Calculate Total Purchase Value

top\_spenders['Total Purchase Value']=purchase\_data\_original.groupby('SN')['Price'].sum()

## Calculate Average purchase price

top\_spenders['Average Purchase Price']=displaydollar(round(top\_spenders['Total Purchase Value']/top\_spenders['Purchase Count'], 2))

## Sort Total Purchase Value in descending order

#DataFrame.sort\_values(by, axis=0, ascending=True, inplace=False, kind=’quicksort’, na\_position=’last’)

top\_spenders=top\_spenders.sort\_values('Total Purchase Value', ascending=False, inplace=False)

###\*\*\*\*\* Function call \*\*\*\*\*\* ###

## Calling function to display currency in dollars.

TotalPurchaseValue=list(top\_spenders['Total Purchase Value'])

top\_spenders['Total Purchase Value']=displaydollar(TotalPurchaseValue)

## Re-arranging the columns to make sure the output looks exactly the same as expected.

top\_spenders=top\_spenders[['Purchase Count', 'Average Purchase Price','Total Purchase Value' ]]

## Display portion of the dataframe using head.

top\_spenders.head()

Out[11]:

|  | **Purchase Count** | **Average Purchase Price** | **Total Purchase Value** |
| --- | --- | --- | --- |
| **SN** |  |  |  |
| **Lisosia93** | 5 | $3.79 | $18.96 |
| **Idastidru52** | 4 | $3.86 | $15.45 |
| **Chamjask73** | 3 | $4.61 | $13.83 |
| **Iral74** | 4 | $3.40 | $13.62 |
| **Iskadarya95** | 3 | $4.37 | $13.10 |

Most Popular Items[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Most-Popular-Items)

Retrieve the Item ID, Item Name, and Item Price columns

Group by Item ID and Item Name. Perform calculations to obtain purchase count, item price, and total purchase value

Create a summary data frame to hold the results

Sort the purchase count column in descending order

Optional: give the displayed data cleaner formatting

Display a preview of the summary data frame

In [12]:

#############################################################################

#### Most popular items ##################

#############################################################################

### Ignore commented code \*\* ###

#itemid=purchase\_data['Item ID']

#itemname= purchase\_data.groupby('Item ID')['Item Name'].unique()

#itemprice=purchase\_data.groupby('Item ID')["Price"].unique()

#purchasecount =purchase\_data.groupby('Item ID')['Purchase ID'].nunique()

#totalpurchasevalue=purchase\_data.groupby('Item ID')["Price"].sum()

#most\_popular\_items1={'Item ID':itemid, 'Item Name':itemname,'Item Price':itemprice,'Total Purchase Value':totalpurchasevalue}

#most\_popular\_itemsdf=pd.DataFrame(most\_popular\_items1)

#most\_popular\_itemsdf

###############################################################################################

## Initialise a dataframe

## perform calculations

most\_popular\_items=pd.DataFrame(purchase\_data\_original.groupby('Item ID')['Purchase ID'].nunique())

## Rename the column to match expected output

most\_popular\_items.rename(columns={"Purchase ID":"Purchase Count"},inplace = True)

## Adding colummns to the dataframe with more calculations.

most\_popular\_items['Item Name']= purchase\_data\_original.groupby('Item ID')['Item Name'].unique()

most\_popular\_items['Item Price']=displaydollar(purchase\_data\_original.groupby('Item ID')["Price"].unique())

most\_popular\_items['Total Purchase Value']= displaydollar(purchase\_data\_original.groupby('Item ID')["Price"].sum())

#### Sort the values - Sort the purchase count column in descending order

#DataFrame.sort\_values(by, axis=0, ascending=True, inplace=False, kind=’quicksort’, na\_position=’last’)

most\_popular\_items=most\_popular\_items.sort\_values('Purchase Count', ascending=False)

most\_popular\_items=most\_popular\_items[['Item Name', 'Purchase Count', 'Item Price', 'Total Purchase Value']]

## Display the portion f the dataframe.

most\_popular\_items.head()

Out[12]:

|  | **Item Name** | **Purchase Count** | **Item Price** | **Total Purchase Value** |
| --- | --- | --- | --- | --- |
| **Item ID** |  |  |  |  |
| **178** | [Oathbreaker, Last Hope of the Breaking Storm] | 12 | $4.23 | $50.76 |
| **145** | [Fiery Glass Crusader] | 9 | $4.58 | $41.22 |
| **108** | [Extraction, Quickblade Of Trembling Hands] | 9 | $3.53 | $31.77 |
| **82** | [Nirvana] | 9 | $4.90 | $44.10 |
| **19** | [Pursuit, Cudgel of Necromancy] | 8 | $1.02 | $8.16 |

Most Profitable Items[¶](http://localhost:8890/nbconvert/html/Heroesofpymoli_divya.ipynb?download=false#Most-Profitable-Items)

Sort the above table by total purchase value in descending order

Optional: give the displayed data cleaner formatting

Display a preview of the data frame

In [13]:

#############################################################################

#### Most profitable items ##################

#############################################################################

#### Intialize a dataframe with calculations.

most\_profitable\_items=pd.DataFrame(purchase\_data\_original.groupby('Item ID')['Purchase ID'].nunique())

## Rename the column to match the expected output.

most\_profitable\_items.rename(columns={"Purchase ID":"Purchase Count"},inplace = True)

#most\_profitable\_items.head()

most\_profitable\_items['Item Name']= purchase\_data\_original.groupby('Item ID')['Item Name'].unique()

### \*\*\* Function call \*\*\*\*

##displaydollar - is a function call to display currencies in dollars.

most\_profitable\_items['Item Price']=displaydollar(purchase\_data\_original.groupby('Item ID')["Price"].unique())

most\_profitable\_items['Total Purchase Value']= purchase\_data\_original.groupby('Item ID')["Price"].sum()

##Sort the total purchase value in descending order

#DataFrame.sort\_values(by, axis=0, ascending=True, inplace=False, kind=’quicksort’, na\_position=’last’)

most\_profitable\_items=most\_profitable\_items.sort\_values('Total Purchase Value', ascending=False)

## Re-arrange the columns to match the expected output

most\_profitable\_items=most\_profitable\_items[['Item Name', 'Purchase Count', 'Item Price', 'Total Purchase Value']]

### \*\*\* Function call \*\*\*\*

##Display currencies in dollars calling function 'displaydollar'

TotalPurchaseValueMP=list(most\_profitable\_items['Total Purchase Value'])

most\_profitable\_items['Total Purchase Value']=displaydollar(TotalPurchaseValueMP)

## Display the portion f the dataframe.

most\_profitable\_items.head()

#DataFrame.sort\_values(by, axis=0, ascending=True, inplace=False, kind=’quicksort’, na\_position=’last’)

#most\_profitable\_items=most\_profitable\_items.sort\_values('Total Purchase Value', axis=0, ascending=True, inplace=False)

#most\_profitable\_items.head()

Out[13]:

|  | **Item Name** | **Purchase Count** | **Item Price** | **Total Purchase Value** |
| --- | --- | --- | --- | --- |
| **Item ID** |  |  |  |  |
| **178** | [Oathbreaker, Last Hope of the Breaking Storm] | 12 | $4.23 | $50.76 |
| **82** | [Nirvana] | 9 | $4.90 | $44.10 |
| **145** | [Fiery Glass Crusader] | 9 | $4.58 | $41.22 |
| **92** | [Final Critic] | 8 | $4.88 | $39.04 |
| **103** | [Singed Scalpel] | 8 | $4.35 | $34.80 |

In [ ]:

## Check age and gender connection