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

- 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.

In [1]:

```
# Dependencies and Setup
import pandas as pd
import numpy as np

# File to Load (Remember to Change These)
purchase_data_file = "Resources/purchase_data.csv"

# Read Purchasing File and store into Pandas data frame
purchase_data = pd.read_csv(purchase_data_file)
purchase_data.head()
```

Out[1]:

	Purchase ID	SN	Age	Gender	Item ID	Item Name	Price
0	0	Lisim78	20	Male	108	Extraction, Quickblade Of Trembling Hands	3.53
1	1	Lisovynya38	40	Male	143	Frenzied Scimitar	1.56
2	2	Ithergue48	24	Male	92	Final Critic	4.88
3	3	Chamassasya86	24	Male	100	Blindscythe	3.27
4	4	Iskosia90	23	Male	131	Fury	1.44

Player Count

- Display the total number of players

```
In [2]: # Use nunique on SN to get Total Players:
        players = pd.DataFrame({'Total Players': [purchase_data["SN"].nunique()]})
        players
```

Out[2]:

	Total Players
0	576

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 [4]:

```
#purch_analysis with nunique for items, mean, nunique for # purchases & total revenue as sum
purch_analysis = pd.DataFrame({'Number of Unique Items': [purchase_data["Item ID"].nunique()],
                               'Average Price': [purchase_data["Price"].mean()],
                               'Number of Purchases': [purchase_data["Purchase ID"].nunique()],
                               'Total Revenue': [purchase_data["Price"].sum()] })
# apply formatting
format_dict = {'Average Price':'${0:,.2f}', 'Total Revenue': '${0:,.2f}'}
purch_analysis.style.format(format_dict)
```

Out[4]:

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

Gender Demographics

- Percentage and Count of Male Players
- Percentage and Count of Female Players
- Percentage and Count of Other / Non-Disclosed

In [6]:

```
# dfgd data frame from SN & Gender, with aggregation on gender and unique SN to determine # of unique players
dfgd = purchase_data[["SN", "Gender"]]
dfgd2 = dfgd.groupby(by = 'Gender', as_index=False).agg({'SN': lambda x: x.nunique()})
# Calculate the percentage
dfgd2['SN2'] = (dfgd2.SN/dfgd2.SN.sum()).map("{0:.2%}".format)
# Sort by # players
dfgd2.sort_values(by='SN2', inplace = True, ascending=False)
# this is to format as per HW image: set index with gender & rename columns
dfgd2.set_index('Gender', inplace=True)
dfgd2.rename_axis(None, inplace=True)
dfgd2.rename(columns ={'SN': 'Total Players', 'SN2' : 'Percentage of Players'} , inplace = True)
dfgd2
```

Out[6]:

	Total Players	Percentage of Players
Male	484	84.03%
Female	81	14.06%
Other / Non-Disclosed	11	1.91%

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 [7]:

```
# dfpa is a dataframe to aggregate on SN, Gender, Item ID & Price
dfpa =  purchase_data[["SN", "Gender", "Item ID", "Price"]]
# Create the aggregates, with SN with nunique to get unique count of players
dfpa2 = dfpa.groupby(by = 'Gender', as_index=False).aggregate({'Item ID': "count",
                                                               'Price': ['mean', 'sum'], 'SN': lambda x: x.nu
nique()})
# Calculate average total price per person
dfpa2['AvgTotPerson'] = (dfpa2[dfpa2.columns[3]]/dfpa2[dfpa2.columns[4]]).map("${0:.2f}".format)
#Format column names as the aggregates resulted in 2-level col names
dfpa2.rename(columns ={'count': 'Purchase Count',
                     'mean' : 'Average Purchase Price',
                       'sum' : 'Total Purchase Value',
                          } , level = 1, inplace = True)
# For HW image formatting, use Gender as index
dfpa2.set_index('Gender', inplace=True)
# Clean up col names by dropping a level
dfpa2.columns = dfpa2.columns.droplevel(0)
# Rename cols as per HW
dfpa2.columns = ['Purchase Count', 'Average Purchase Price', 'Total Purchase Value', 'Unique','Average Total
 Price per Person']
# drop the unique as it is not required in the final output
dfpa2 = dfpa2.drop(dfpa2.columns[[3]], axis=1)
# apply dollar formatting
dfpa2['Average Purchase Price'] = dfpa2['Average Purchase Price'].apply(lambda x: "${:.2f}".format((x)))
dfpa2['Total Purchase Value'] = dfpa2['Total Purchase Value'].apply(lambda x: "${:.2f}".format((x)))
dfpa2
```

Out[7]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Average Total Price per Person
Gender				
Female	113	$3.20	$361.94	$4.47
Male	652	$3.02	$1967.64	$4.07
Other / Non-Disclosed	15	$3.35	$50.19	$4.56

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 [12]:

```
# dfage is a dataframe to aggregate on SN, Age
dfage =  purchase_data[["SN", "Age"]]
dfage = dfage.groupby(by = 'Age', as_index=False).aggregate({'SN': lambda x: x.nunique()})
# Create the bins in which Data will be held
bins = [0, 9, 14, 19, 24, 29, 34, 39, 100]
# Create the names for the four bins
group_names = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39", "40+"]
# Apply the bins to the dataframe
dfage["Bin"] = pd.cut(dfage["Age"], bins, labels = group_names)
dfage = dfage.groupby(by="Bin", as_index=False).aggregate({'SN': "sum"})

# Calculate percentage
dfage['Percentage of Players'] = (dfage.SN/dfage.SN.sum()).map("{0:.2%}".format)
# Move Bin as index and remove the col heading
dfage.set_index('Bin', inplace=True)
dfage.rename_axis(None, inplace=True)
dfage.rename(columns = {'SN':'Total Players'}, inplace = True)
dfage
```

Out[12]:

```
       Total Players  Percentage of Players
  <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)

- 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 [15]:

```
# dfpap is a dataframe to aggregate on SN, Age, Item ID & Price
dfpap =  purchase_data[["SN", "Age", "Item ID", "Price"]]

# Create the bins in which Data will be held
bins = [0, 9, 14, 19, 24, 29, 34, 39, 100]
# Create the names for the four bins
group_names = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39", "40+"]
#Apply the bins to the data frame
dfpap["Bin"] = pd.cut(dfpap["Age"], bins, labels = group_names)

# Create the aggregates, with SN with nunique to get unique count of players
dfpap = dfpap.groupby(by = 'Bin', as_index=False).aggregate({'Item ID': "count",
                                                             'Price': ['mean', 'sum'], 'SN': lambda x: x.nuni
que()})
# Calculate average total price per person
dfpap['AvgTotPerson'] = (dfpap[dfpap.columns[3]]/dfpap[dfpap.columns[4]]).map("${0:.2f}".format)
dfpap
dfpap.rename(columns ={'count': 'Purchase Count',
                     'mean' : 'Average Purchase Price',
                       'sum' : 'Total Purchase Value',
                          } , level = 1, inplace = True)

# For HW image formatting, use Gender as index
dfpap.set_index('Bin', inplace=True)
# Clean up col names by dropping a level
dfpap.columns = dfpap.columns.droplevel(0)
# Rename cols as per HW
dfpap.columns = ['Purchase Count', 'Average Purchase Price', 'Total Purchase Value', 'Unique','Average Total
 Price per Person']
# drop the unique as it is not required in the final output
dfpap = dfpap.drop(dfpap.columns[[3]], axis=1)
dfpap['Average Purchase Price'] = dfpap['Average Purchase Price'].apply(lambda x: "${:.2f}".format((x)))
dfpap['Total Purchase Value'] = dfpap['Total Purchase Value'].apply(lambda x: "${:.2f}".format((x)))
# move the 1st row to bottom
dfpap.reindex(index=np.roll(dfpap.index,-1))
```

```
C:\Users\mcala\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy
  if __name__ == '__main__':
```

Out[15]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Average Total Price per Person
Bin				
10-14	28	$2.96	$82.78	$3.76
15-19	136	$3.04	$412.89	$3.86
20-24	365	$3.05	$1114.06	$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

- 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 [28]:

```python
# dfsp is a dataframe to aggregate on SN, Purchase ID, Item ID & Price
dfsp =  purchase_data[["SN", "Purchase ID", "Item ID", "Price"]]
# Create the aggregates, with SN, Item ID count and mean & sum of Price
dfsp = dfsp.groupby(by = 'SN', as_index=False).aggregate({'Item ID': "count", 'Price': ['mean', 'sum']})
#Format column names as the aggregates resulted in 2-level col names
dfsp.rename(columns ={'count': 'Purchase Count',
                     'mean' : 'Average Purchase Price',
                       'sum' : 'Total Purchase Value',
                          } , level = 1, inplace = True)

# For HW image formatting, use SN as index
dfsp.set_index('SN', inplace=True)
# Clean up col names by dropping a level
# Top 5 by Total Purchase Value
dfsp.columns = dfsp.columns.droplevel(0)
dfsp2 = dfsp.nlargest(5, columns=['Total Purchase Value'])
dfsp2['Average Purchase Price'] = dfsp2['Average Purchase Price'].map('${:,.2f}'.format)
dfsp2['Total Purchase Value'] = dfsp2['Total Purchase Value'].map('${:,.2f}'.format)
dfsp2
```

Out[28]:

	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

- 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 [111]:

```
# dfpop is a dataframe to aggregate on Item ID, Purchase ID, Item Name & Price
dfpop =  purchase_data[[ "Item ID", "Item Name", "Price"]]

# Create the aggregates by Item ID and Item Name
dfpop = dfpop.groupby(['Item ID', 'Item Name']).aggregate({'Item ID': "count", 'Price': ['max', 'sum']})
#Format column names as the aggregates resulted in 2-level col names
dfpop.rename(columns ={'count': 'Purchase Count',
                     'max' : 'Item Price',
                       'sum' : 'Total Purchase Value',
                          } , level = 1, inplace = True)
# Clean up col names by dropping a level
dfpop.columns = dfpop.columns.droplevel(0)
dfpop['Item Price'] = dfpop['Item Price'].map('${:,.2f}'.format)
dfpop['Total Purchase Value'] = dfpop['Total Purchase Value'].map('${:,.2f}'.format)
dfpop.sort_values(by= 'Purchase Count', ascending=False).head(5)
```

Out[111]:

		Purchase Count	Item Price	Total Purchase Value
Item ID	**Item Name**			
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

- 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 [112]:

```
#Change the $ Purchase Value to Float
dfpop['Total Purchase Value'] = dfpop['Total Purchase Value'].apply(lambda x : float(x.replace('$', '')))
# Sort by most profitable into a new DF
dfpop2 = dfpop.sort_values(by= 'Total Purchase Value', ascending=False)
# Apply formatting
dfpop2['Total Purchase Value'] = dfpop['Total Purchase Value'].map('${:,.2f}'.format)
dfpop2.head(5)
```

Out[112]:

		Purchase Count	Item Price	Total Purchase Value
Item ID	**Item Name**			
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