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%).

In [1]:

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

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

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

Player Count

In [2]:

```
# Calculate the Number of Unique Players
player_demographics = purchase_data.loc[:, ["Gender", "SN", "Age"]]
num_players = player_demographics["SN"].nunique()
# Display the total number of players
pd.DataFrame({"Total Players": [num_players]})
```

Out[2]:

	Total Players
0	576

Purchasing Analysis (Total)

In [3]:

```
# Run basic calculations
average_item_price = purchase_data["Price"].mean()
total_purchase_value = purchase_data["Price"].sum()
purchase_count = purchase_data["Price"].count()
item_count = len(purchase_data["Item ID"].unique())

# Create a DataFrame to hold results
summary_table = pd.DataFrame({"Number of Unique Items": item_count,
                              "Total Revenue": [total_purchase_value],
                              "Number of Purchases": [purchase_count],
                              "Average Price": [average_item_price]})

# Minor Data Munging
# summary_table = summary_table.round(2)
# summary_table ["Average Price"] = summary_table["Average Price"].map("$
# summary_table ["Number of Purchases"] = summary_table["Number of Purcha
# summary_table ["Total Revenue"] = summary_table["Total Revenue"].map("$
# summary_table = summary_table.loc[:,["Number of Unique Items", "Average

# Display the summary_table
summary_table
```

Out[3]:

	Average Price	Number of Purchases	Number of Unique Items	Total Revenue
0	3.050987	780	183	2379.77

Gender Demographics

In [4]:

```
# Calculate the Number and Percentage by Gender
gender_demographics_totals = player_demographics["Gender"].value_counts()
gender_demographics_percents = gender_demographics_totals / num_players *
gender_demographics = pd.DataFrame({"Total Count": gender_demographics_to

# Minor Data Munging
gender_demographics = gender_demographics.round(2)

gender_demographics
```

Out[4]:

	Percentage of Players	Total Count
Male	113.19	652
Female	19.62	113
Other / Non-Disclosed	2.60	15

Purchasing Analysis (Gender)

In [5]:

```
# Run basic calculations
gender_purchase_total = purchase_data.groupby(["Gender"]).sum()["Price"].
gender_average = purchase_data.groupby(["Gender"]).mean()["Price"].rename
gender_counts = purchase_data.groupby(["Gender"]).count()["Price"].rename

# Calculate Normalized Purchasing (Average Total Purchase per Person)
normalized_total = gender_purchase_total / gender_demographics["Total Cou

# Convert to DataFrame
gender_data = pd.DataFrame({"Purchase Count": gender_counts, "Average Pur

# Minor Data Munging
gender_data["Average Purchase Price"] = gender_data["Average Purchase Pri
gender_data["Total Purchase Value"] = gender_data["Total Purchase Value"]
gender_data ["Purchase Count"] = gender_data["Purchase Count"].map("{:,}"
gender_data["Avg Total Purchase per Person"] = gender_data["Normalized To
gender_data = gender_data.loc[:, ["Purchase Count", "Average Purchase Pri

# Display the Gender Table
gender_data
```

Out[5]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Avg Total Purchase per Person
Gender				
Female	113	$3.20	$361.94	$3.20
Male	652	$3.02	$1,967.64	$3.02
Other / Non-Disclosed	15	$3.35	$50.19	$3.35

Age Demographics

In [6]:

```python
# Establish the bins
age_bins = [0, 9.90, 14.90, 19.90, 24.90, 29.90, 34.90, 39.90, 99999]
group_names = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39

# Categorize the existing players using the age bins
player_demographics["Age Ranges"] = pd.cut(player_demographics["Age"], ag

# Calculate the Numbers and Percentages by Age Group
age_demographics_totals = player_demographics["Age Ranges"].value_counts(
age_demographics_percents = age_demographics_totals / num_players * 100
age_demographics = pd.DataFrame({"Total Count": age_demographics_totals,

# Minor Data Munging
age_demographics = age_demographics.round(2)

# Display Age Demographics Table
age_demographics.sort_index()
```

Out[6]:

```
       Percentage of Players  Total Count
  <10                   3.99           23
10-14                   4.86           28
15-19                  23.61          136
20-24                  63.37          365
25-29                  17.53          101
30-34                  12.67           73
35-39                   7.12           41
  40+                   2.26           13
```

Purchasing Analysis (Age)

In [7]:

```
# Bin the Purchasing Data
purchase_data["Age Ranges"] = pd.cut(purchase_data["Age"], age_bins, labe

# Run basic calculations
age_purchase_total = purchase_data.groupby(["Age Ranges"]).sum()["Price"]
age_average = purchase_data.groupby(["Age Ranges"]).mean()["Price"].renam
age_counts = purchase_data.groupby(["Age Ranges"]).count()["Price"].renam

# Calculate Normalized Purchasing (Average Purchase Total per Person)
normalized_total = age_purchase_total / age_demographics["Total Count"]

# Convert to DataFrame
age_data = pd.DataFrame({"Purchase Count": age_counts, "Average Purchase

# Minor Data Munging
age_data["Average Purchase Price"] = age_data["Average Purchase Price"].m
age_data["Total Purchase Value"] = age_data["Total Purchase Value"].map("
age_data ["Purchase Count"] = age_data["Purchase Count"].map("{:,}".forma
age_data["Avg Total Purchase per Person"] = age_data["Normalized Totals"]
age_data = age_data.loc[:, ["Purchase Count", "Average Purchase Price", "

# Display the Age Table
age_data
```

Out[7]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Avg Total Purchase per Person
10-14	28	$2.96	$82.78	$2.96
15-19	136	$3.04	$412.89	$3.04
20-24	365	$3.05	$1,114.06	$3.05
25-29	101	$2.90	$293.00	$2.90
30-34	73	$2.93	$214.00	$2.93
35-39	41	$3.60	$147.67	$3.60
40+	13	$2.94	$38.24	$2.94
<10	23	$3.35	$77.13	$3.35

Top Spenders

In [8]:

```
# Basic Calculations
user_total = purchase_data.groupby(["SN"]).sum()["Price"].rename("Total P
user_average = purchase_data.groupby(["SN"]).mean()["Price"].rename("Aver
user_count = purchase_data.groupby(["SN"]).count()["Price"].rename("Purch

# Convert to DataFrame
user_data = pd.DataFrame({"Total Purchase Value": user_total, "Average Pu

# Display Table
user_sorted = user_data.sort_values("Total Purchase Value", ascending=Fal

# Minor Data Munging
user_sorted["Average Purchase Price"] = user_sorted["Average Purchase Pri
user_sorted["Total Purchase Value"] = user_sorted["Total Purchase Value"]
user_sorted = user_sorted.loc[:,["Purchase Count", "Average Purchase Pric

# Display DataFrame
user_sorted.head(5)
```

Out[8]:

	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

```
In [9]: # Extract item Data
item_data = purchase_data.loc[:,["Item ID", "Item Name", "Price"]]

# Perform basic calculations
total_item_purchase = item_data.groupby(["Item ID", "Item Name"]).sum()["
average_item_purchase = item_data.groupby(["Item ID", "Item Name"]).mean(
item_count = item_data.groupby(["Item ID", "Item Name"]).count()["Price"]

# Create new DataFrame
item_data_pd = pd.DataFrame({"Total Purchase Value": total_item_purchase,

# Sort Values
item_data_count_sorted = item_data_pd.sort_values("Purchase Count", ascen

# Minor Data Munging
item_data_count_sorted["Item Price"] = item_data_count_sorted["Item Price
item_data_count_sorted["Purchase Count"] = item_data_count_sorted["Purcha
item_data_count_sorted["Total Purchase Value"] = item_data_count_sorted["
item_popularity = item_data_count_sorted.loc[:,["Purchase Count", "Item P

item_popularity.head(5)
```

Out[9]:

		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

In [10]:

```
# Item Table (Sorted by Total Purchase Value)
item_total_purchase = item_data_pd.sort_values("Total Purchase Value", as

# Minor Data Munging
item_total_purchase["Item Price"] = item_total_purchase["Item Price"].map
item_total_purchase["Purchase Count"] = item_total_purchase["Purchase Cou
item_total_purchase["Total Purchase Value"] = item_total_purchase["Total 
item_profitable = item_total_purchase.loc[:,["Purchase Count", "Item Pric

item_profitable.head(5)
```

Out[10]:

		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

In [ ]: