Heroes Of Pymoli Data Analysis

*Male players generate more revenue overall, due to their sheer numbers, but female players outspend males per head, and players who claim "other/non-disclosed" gender are the highest per-player spenders.

*Per-Player spending drops sharply as players reach Age 39 and Up

*The overwhelming majority of players are males between ages 15-26.

*The top 5 revenue generators bring in roughly 10% of the overall revenue.

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 [18]:
```
# Dependencies and Setup
import pandas as pd
import numpy as np

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

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

Out[18]:

	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 [19]:
```
playerdata['SN'].nunique()  #nunique counts distinct values.
```

Out[19]:
```
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 [7]:
```
TotalOrders = playerdata["Item ID"].count()
TotalRevenue = playerdata["Price"].sum()
AverageOrder = playerdata["Price"].mean()
UniqueItems = playerdata["Item Name"].nunique()
```

In [8]:
```
# Creating a summary DataFrame using above values
purchases_df = pd.DataFrame({"Total Order Count":[TotalOrders],
                             "Total Unique Items": [UniqueItems],
                             "Average Sale": [AverageOrder],
                           "Total Revenue": [TotalRevenue]})

purchases_df["Total Revenue"] = purchases_df["Total Revenue"].map("${:.2f}".form
purchases_df["Average Sale"] = purchases_df["Average Sale"].map("${:.2f}".format
```

In [20]:
```
purchases_df
```

Out[20]:

	Total Order Count	Total Unique Items	Average Sale	Total Revenue
0	780	179	$3.05	$2379.77

Gender Demographics

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

In [10]:
```
#Define the Counters
playercount = playerdata['Gender'].count()
malescount = (playerdata['Gender']=='Male').sum()
femalescount =(playerdata['Gender']=='Female').sum()
otherscount = (playerdata['Gender']=='Other / Non-Disclosed').sum()
pctmale = (malescount / playercount) * 100
pctfemale = (femalescount / playercount)* 100
pctothers = (otherscount/playercount) * 100
```

In [11]:
```
# Creating a summary DataFrame using above values
demos_df = pd.DataFrame({"Total Player Info":[playercount],
                        "Male Players":[malescount],
                        "Female Players":[femalescount],
                        "Other/Non-Disclosed":[otherscount],
                        "Percent by Gender(Male)":[pctmale],
                        "Percent by Gender(Female)":[pctfemale],
                         "Percent by Gender(NonDisclosed)":[pctothers]})

demos_df.round(2) #this rounds all the numbers in the dataframe to 2 places past
```

Out[11]:

	Total Player Info	Male Players	Female Players	Other/Non-Disclosed	Percent by Gender(Male)	Percent by Gender(Female)	Percent by Gender(NonDisclosed)
0	780	652	113	15	83.59	14.49	1.92

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 [12]:

```
#Group by Gender
gengroup = playerdata.groupby("Gender")

# Purchase Count
gn_count = playerdata.groupby(["Gender"]).count()["Price"]
gn_count

# Average Purchase Value
gnaverage_price = playerdata.groupby(["Gender"]).mean()["Price"]
gnaverage_price

# Total Purchase Value
gntotal_purch_v = playerdata.groupby(["Gender"]).sum()["Price"]
gntotal_purch_v


genderspend = pd.DataFrame({"Purchase Count": gn_count,
                      "Average Purchase Value": gnaverage_price,
                            "Total Purchase Value": gntotal_purch_v
                           })

genderspend["Average Purchase Value"] = genderspend["Average Purchase Value"].ma
genderspend["Total Purchase Value"] = genderspend["Total Purchase Value"].map("$

genderspend
```

Out[12]:

	Purchase Count	Average Purchase Value	Total Purchase Value
Gender			
Female	113	$3.20	$361.94
Male	652	$3.02	$1967.64
Other / Non-Disclosed	15	$3.35	$50.19

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 [13]:
```
# Create the bins in which Data will be held
#Need to clean up the formatting

bins = [0, 10, 15, 19, 23, 27, 31, 35, 39, 100]

# Create the names for the four bins
Age_Group_Names = ["Under 10", "10-14", "15-18", "19-22", "23-26","27-30", "31-3

#insert the data into the bins
#then group it

agedata = pd.cut(playerdata["Age"], bins, labels = Age_Group_Names).head()

playerdata["Age Range"] = pd.cut(playerdata["Age"],bins,labels = Age_Group_Names
playerdata.head()

#Group by Age Range
group = playerdata.groupby("Age Range")



age_ranges_df = playerdata.groupby(["Age Range"])
age_ranges_df


playeragecount = age_ranges_df["SN"].count() #count the screen names within the 
playerpercent = (age_ranges_df["SN"].count() / playercount) * 100
playeragecount
playerpercent

agedemo_df = pd.DataFrame({"Percentage of Players": playerpercent,
                                    "Total in this Group": playeragecount
                                    })




agedemo_df.round(2)
```

Out[13]:

	Percentage of Players	Total in this Group
Age Range		
Under 10	4.10	32
10-14	6.92	54
15-18	12.95	101
19-22	38.21	298
23-26	19.23	150
27-30	7.69	60
31-34	5.77	45

	Percentage of Players	Total in this Group
Age Range		
35-38	3.46	27
39 and Up	1.67	13

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 [14]:
```
pd.cut(playerdata["Age"], bins, labels = Age_Group_Names).head()

playerdata["Age Range"] = pd.cut(playerdata["Age"],bins,labels = Age_Group_Names
playerdata.head()

#Group by Age Range
group = playerdata.groupby("Age Range")

# Purchase Count
sn_count = playerdata.groupby(["Age Range"]).count()["Age"]
sn_count

# Average Purchase Value
average_price = playerdata.groupby(["Age Range"]).mean()["Price"]
average_price

# Total Purchase Value
total_purch_v = playerdata.groupby(["Age Range"]).sum()["Price"]
total_purch_v


agemetrics = pd.DataFrame({"Purchase Count": sn_count,
                      "Total Purchase Value": total_purch_v,
                      "Average Purchase Value": average_price
                     })

agemetrics["Average Purchase Value"] = agemetrics["Average Purchase Value"].map(
agemetrics["Total Purchase Value"] = agemetrics["Total Purchase Value"].map("${:


agemetrics
```

Out[14]:

Age Range	Purchase Count	Total Purchase Value	Average Purchase Value
Under 10	32	$108.96	$3.40
10-14	54	$156.60	$2.90
15-18	101	$307.24	$3.04
19-22	298	$903.84	$3.03
23-26	150	$459.54	$3.06
27-30	60	$178.05	$2.97
31-34	45	$131.66	$2.93
35-38	27	$95.64	$3.54
39 and Up	13	$38.24	$2.94

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 [15]:

```
#Group by Screen Name
sngroup = playerdata.groupby("SN")

# Purchase Count
sn_count = playerdata.groupby(["SN"]).count()["Price"]
sn_count

# Average Purchase Value
average_price = playerdata.groupby(["SN"]).mean()["Price"]
average_price

# Total Purchase Value
total_purch_v = playerdata.groupby(["SN"]).sum()["Price"]
total_purch_v


spenders = pd.DataFrame({"Purchase Count": sn_count,
                      "Average Purchase Value": average_price,
                        "Total Purchase Value": total_purch_v
                     })

spenders["Average Purchase Value"] = spenders["Average Purchase Value"].map("${:
spenders["Total Purchase Value"] = spenders["Total Purchase Value"].map("${:.2f}
spenders = spenders.sort_values(by="Total Purchase Value", ascending =False)

spenders
spenders.head(5)
```

Out[15]:

SN	Purchase Count	Average Purchase Value	Total Purchase Value
Haillyrgue51	3	$3.17	$9.50
Phistym51	2	$4.75	$9.50
Lamil79	2	$4.64	$9.29
Aina42	3	$3.07	$9.22
Saesrideu94	2	$4.59	$9.18

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 [16]:

```
# Find the biggest sellers.

itemcount = playerdata.groupby(["Item ID", "Item Name"]).count()["Price"].rename
avgprice =playerdata.groupby(["Item ID", "Item Name"]).mean()["Price"].rename("A
totalprice = playerdata.groupby(["Item ID", "Item Name"]).sum()["Price"].rename(

# Convert to DataFrame

sellers_df = pd.DataFrame({"Item Count":itemcount,
                                   "Average Price":avgprice,
                                   "Total Purchase Price":totalprice,})
sellers_df = sellers_df.sort_values(by='Item Count',ascending=False)

sellers_df.head()
```

Out[16]:

		Item Count	Average Price	Total Purchase Price
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 [17]:
```
# Find the biggest moneymakers.
# This is basically the same as the exercise above, but change the ordering

itemcount = playerdata.groupby(["Item ID", "Item Name"]).count()["Price"].rename
avgprice =playerdata.groupby(["Item ID", "Item Name"]).mean()["Price"].rename("A
totalprice = playerdata.groupby(["Item ID", "Item Name"]).sum()["Price"].rename(

# Convert to DataFrame

toprev_df = pd.DataFrame({"Item Count":itemcount,
                                   "Average Price":avgprice,
                                   "Total Purchase Price":totalprice,})
toprev_df = toprev_df.sort_values(by='Total Purchase Price',ascending=False)
toprev_df["Average Price"] = toprev_df["Average Price"].map("${:.2f}".format)
toprev_df["Total Purchase Price"] = toprev_df["Total Purchase Price"].map("${:.2


toprev_df.head()
```

Out[17]:

		Item Count	Average Price	Total Purchase Price
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 [ ]:

In [ ]: