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 [11]:
```
# 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

- Display the total number of players

In [12]:
```
# Total Number of Players
Player_Count = pd.DataFrame([{"Total Players":purchase_data["SN"].nunique()}])
Player_Count
```

Out[12]:

	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 [13]:
```
# Run basic calculations on the purchase_data
num_items = purchase_data['Item ID'].nunique()
avg_price = purchase_data['Price'].mean()
num_purchases = purchase_data['Purchase ID'].count()
total_revenue = purchase_data['Price'].sum()
purchase_analysis = pd.DataFrame({"Number of Unique Items": [num_items],
                                  "Average_Price": [avg_price],
                                  "Number of Purchases": [num_purchases],
                                  "Total_Revenue": total_revenue})
purchase_analysis
```

Out[13]:

	Number of Unique Items	Average_Price	Number of Purchases	Total_Revenue
0	183	3.050987	780	2379.77

Gender Demographics

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

In [14]:
```
# determine the gender demographics of the purchase_data
purchase_data.head(1)
# count and percentage of Unique Screen Names by gender
gender = pd.DataFrame(purchase_data.groupby("Gender")['SN'].nunique())
gender_renamed = gender.rename(columns={"SN": "Total Counts"})
gender_renamed['Percentage of Players'] = gender / total_players * 100

gender_renamed
```

Out[14]:

Gender	Total Counts	Percentage of Players
Female	81	14.062500
Male	484	84.027778
Other / Non-Disclosed	11	1.909722

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 [15]:
```
# determine the gender demographics of the purchase_data
purchase_data.head(1)
# Select Purchase ID, Price,  SN
purchase_analysis = purchase_data[["SN", "Gender", "Price"]]
# Group By Gender
purchase_analysis_group = purchase_analysis.groupby(["Gender"])

# Calculate averages for purchase data by Gender Only
# Purchase Count
purchase_count = purchase_analysis_group['SN'].count()

#Average Purchase Price
average_purchase_price = purchase_analysis_group['Price'].mean().round(2)

#Total Purchase Value By Gender
total_purchase_value = purchase_analysis_group['Price'].sum()

# Average Total Purchase PER PERSON by gender
purchase_analysis_group_person = pd.DataFrame(purchase_analysis.groupby(['Gend
er','SN']).sum())
purchase_analysis_group_person = purchase_analysis_group_person.groupby(['Gend
er'])['Price'].mean().round(2)
purchase_analysis_group_person


#Return dataframe of purchase analysis by gender
purchase_analysis_gender = pd.DataFrame({"Purchase Count" : purchase_count,
                                         "Average Purchase Price" : average_pu
rchase_price,
                                         "Total Purchase Value" : total_purcha
se_value,
                                         "Avg Total Purchase per Person" : pur
chase_analysis_group_person})

#convert column to currency
purchase_analysis_gender["Average Purchase Price"] = purchase_analysis_gender[
"Average Purchase Price"].map("${:.2f}".format)
purchase_analysis_gender["Total Purchase Value"] = purchase_analysis_gender["T
otal Purchase Value"].map("${:.2f}".format)
purchase_analysis_gender["Avg Total Purchase per Person"] = purchase_analysis_
gender["Avg Total Purchase per Person"].map("${:.2f}".format)
purchase_analysis_gender
```

Out[15]:

Gender	Purchase Count	Average Purchase Price	Total Purchase Value	Avg Total Purchase per Person
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 [16]:
```
#original dataframe
purchase_data.head(2)
#new dataframe with no duplicates
age_data = purchase_data.drop_duplicates('SN')
age_data.count()  # This should equal 576


#Create the age bins
# bins are <10, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40+
bins = [-1,9,14,19,24,29,34,39,101]
#Create names for bins  (one less than bins)
bin_names = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39","40+"
]
age_data["Age Demographics"] = pd.cut(age_data["Age"],bins,labels=bin_names)
purchase_data_age = age_data.groupby("Age Demographics")

#Calculate count and percentage for each bin
age_count = purchase_data_age['Age'].count()
percentage_of_players = age_count / total_players
age_demo = pd.DataFrame({"Age Count" : age_count, "Percentage of Players": per
centage_of_players})
age_demo['Percentage of Players'] = age_demo['Percentage of Players'].map("
{:,.2%}".format)

age_demo
```

```
C:\Users\casey\Anaconda3\lib\site-packages\ipykernel_launcher.py:13: SettingW
ithCopyWarning:
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/st
able/indexing.html#indexing-view-versus-copy
  del sys.path[0]
```

Out[16]:

Age Demographics	Age Count	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 [17]:
```
# Do not use the same bins as were use in Age Demographics
#Calculate count and averages
purchase_data["Age Demographics"] = pd.cut(purchase_data["Age"],bins,labels=bi
n_names)
purchase_data_age = purchase_data.groupby("Age Demographics")
purchase_count_age = purchase_data_age['Purchase ID'].count()
avg_purchase_price_age = purchase_data_age['Price'].mean()
total_purchase_value_age = purchase_data_age['Price'].sum()

#Use original dataframe (with duplicate SN to get the avg by SN)
purchase_analysis_age_person = pd.DataFrame(purchase_data.groupby(['Age Demogr
aphics','SN']).sum())
purchase_analysis_age_person = purchase_analysis_age_person.groupby(['Age Demo
graphics'])['Price'].mean().round(2)
purchase_analysis_age_person

#Create a new dateframe
age_purchase_analysis = pd.DataFrame({"Purchase Count" : purchase_count_age,
                                      "Average Purchase Price" : avg_purchase_
price_age,
                                      "Total Purchase Value" : total_purchase_
value_age,
                                     "Avg Total Purchase per Person" : purchas
e_analysis_age_person})


#format columns
age_purchase_analysis["Average Purchase Price"] = age_purchase_analysis["Avera
ge Purchase Price"].map("${:.2f}".format)
age_purchase_analysis["Total Purchase Value"] = age_purchase_analysis["Total P
urchase Value"].map("${:.2f}".format)
age_purchase_analysis["Avg Total Purchase per Person"] = age_purchase_analysis
["Avg Total Purchase per Person"].map("${:.2f}".format)

age_purchase_analysis
```

Out[17]:

Age Demographics	Purchase Count	Average Purchase Price	Total Purchase Value	Avg Total Purchase per Person
<10	23	$3.35	$77.13	$4.54
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

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 [18]:
```
# Group by unique SN, but no bins

purchase_data.head(2)
# Group By SN
purchase_data_SN = purchase_data.groupby(["SN"])
# Calculations
purchase_count_SN = purchase_data_SN['Purchase ID'].count()
avg_purchase_price_SN = purchase_data_SN['Price'].mean()
total_purchase_value_SN = purchase_data_SN['Price'].sum()
#create new dataframe
SN_purchase_analysis = pd.DataFrame({"Purchase Count" : purchase_count_SN,
                                     "Average Purchase Price" : avg_purchase_
price_SN,
                                     "Total Purchase Value" : total_purchase_
value_SN})
SN_purhase_analysis = SN_purchase_analysis.sort_values(['Total Purchase Value'
], ascending = False, inplace = True)
#format columns
SN_purchase_analysis["Average Purchase Price"] = SN_purchase_analysis["Average
 Purchase Price"].map("${:.2f}".format)
SN_purchase_analysis["Total Purchase Value"] = SN_purchase_analysis["Total Pur
chase Value"].map("${:.2f}".format)
SN_purchase_analysis.head()
```

Out[18]:

SN	Purchase Count	Average Purchase Price	Total Purchase Value
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 [19]:
```
#Create df using the original purchase data information, extract Item ID, Item
 Name, and Price
popular_items = purchase_data[['Item ID','Item Name','Price']]
#Groupby Item ID and Item Name
popular_items_group = popular_items.groupby(['Item ID', 'Item Name'])
# Calculations
purchase_count_pop = popular_items_group['Item ID'].count()
item_price_pop = popular_items_group['Price'].mean()
total_purchase_value_pop = popular_items_group['Price'].sum()
#create new dataframe
pop_item_analysis = pd.DataFrame({'Purchase Count' : purchase_count_pop,
                                  'Item Price' : item_price_pop,
                                  'Total Purchase Value' : total_purchase_valu
e_pop})
pop_item_analysis_sorted = pop_item_analysis.sort_values(['Purchase Count'], a
scending = False)
#format columns
pop_item_analysis_sorted["Item Price"] = pop_item_analysis_sorted["Item Price"
].map("${:.2f}".format)
pop_item_analysis_sorted["Total Purchase Value"] = pop_item_analysis_sorted["T
otal Purchase Value"].map("${:.2f}".format)
pop_item_analysis_sorted.head()
```

Out[19]:

Item ID	Item Name	Purchase Count	Item Price	Total Purchase Value
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 [20]:
```
#sort the popular item dataframe by Total purchase value, descending.
profit_item_analysis = pop_item_analysis.sort_values(['Total Purchase Value'],
 ascending = False)
#format columns
profit_item_analysis["Item Price"] = profit_item_analysis["Item Price"].map("$
{:.2f}".format)
profit_item_analysis["Total Purchase Value"] = profit_item_analysis["Total Pur
chase Value"].map("${:.2f}".format)

profit_item_analysis.head()
```

Out[20]:

Item ID	Item Name	Purchase Count	Item Price	Total Purchase Value
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