Observed Trends

- At just over 84%, males are overwhelmingly the majority of players in Heroes of Pymoli.
- Although males are the majority of the players, females and other/non-disclosed spent almost $0.50 more on average.
- Almost half of the player population (44.79%) is ages 20-24.

In [2]:
```
# Dependencies and Setup
import pandas as pd

# File to load
file_to_load = "Resources/purchase_data.csv"

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

Player Count

- Display the total number of players

In [118]:
```
# Count the number of unique players in the SN column and create a dataframe to store it
total_players = [purchase_data_df['SN'].nunique()]
total_players_df = pd.DataFrame(total_players, columns = ['Total Players'])
```

In [119]:
```
# Display the total number of players
total_players_df
```

Out[119]:

	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 [6]:
```
# Calculate the Number of Unique Items, Average Price, Number of Purchases, and Total Revenue
total_items = purchase_data_df['Item ID'].nunique()
avg_price = purchase_data_df['Price'].mean()
num_purchases = len(purchase_data_df.index)
total_rev = purchase_data_df['Price'].sum()

# Create a dataframe to store the above
purchasing_analysis_df = pd.DataFrame({'Number of Unique Items': total_items,
                                       'Average Price': "${:.2f}".format(avg_price),
                                       'Number of Purchases':num_purchases,
                                       'Total Revenue':"${:,.2f}".format(total_rev)})
```

In [8]:
```
# Display the summary
purchasing_analysis_df
```

Out[8]:

Number of Unique Items	Average Price	Number of Purchases	Total Revenue
0	179	$3.05	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 [124]:
```
# Count the unique number of players for each gender and create a dataFrame
gender_groupby_df = purchase_data_df.groupby('Gender')
gender_count = gender_groupby_df['SN'].nunique()
gender_count_df = pd.DataFrame.from_dict(dict(gender_count), orient='index', columns=['Total Count'])
```

In [123]:
```
# Calculate the percentage of players by gender, and create a dataframe
gender_percentage = (gender_count/total_players)
gender_percentage_df = pd.DataFrame.from_dict(dict(gender_percentage), orient='index', columns=['Percentage of Players'])
```

In [122]:
```
# Merge the two dataFrames and sort by the Total Count column
gender_demographics_df = pd.merge(gender_count_df, gender_percentage_df, left_index=True, right_index=True)
gender_demographics_sorted_df = gender_demographics_df.sort_values(by=['Total Count'], ascending=False)

# Set the format of the Percentage of Players column and displays the dataframe
gender_demographics_sorted_df.style.format({'Percentage of Players': '{:.2%}'})
```

Out[122]:

	Total Count	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 [125]:
```
# Count the total number of purchases for each gender and create a dataframe
purchase_count = purchase_data_df['Gender'].value_counts()
purchase_count_df = pd.DataFrame.from_dict(dict(purchase_count), orient='index', columns=['Purchase Count'])

# Add Gender as the index name and sort by index
purchase_count_df.index.name = 'Gender'
purchase_count_df.sort_index(inplace=True)
```

In [126]:
```
# Calculate the Average Purchase Price by gender and create dataframe
avg_purchase_price_df = purchase_data_df.groupby(['Gender'])['Price'].mean()
avg_purchase_price_df=avg_purchase_price_df.rename(columns={'Gender':'Gender', 'Price': 'Average Purchase Price'}, inplace=True)
```

In [127]:
```
# Merge the Purchase Count and Average Purchase Price dataframes
gender_pur_analysis_pc_app_df = pd.merge(purchase_count_df, avg_purchase_price_df, left_index=True, right_index=True)
```

In [128]:
```
# Calculate the Total Purchase Value by gender and create dataframe
total_purchase_value_df = purchase_data_df.groupby(['Gender'])['Price'].sum()
total_purchase_value_df.rename(columns={'Gender':'Gender', 'Price': 'Total Purchase Value'}, inplace=True)
```

In [129]:
```
# Merge the Total Purchase Value into the merged dataframes
gender_pur_analysis_pc_app_tpv_df = pd.merge(gender_pur_analysis_pc_app_df, total_purchase_value_df, left_index=True, right_index=True)
```

In [130]:
```
# Calculate the Avg Total Purchase per Person by gender and create dataframe
avg_total_pur_per_person = purchase_data_df.groupby(['SN', 'Gender'])['Price'].sum().reset_index().groupby('Gender')['Price'].mean()
avg_total_pur_per_person_df = pd.DataFrame.from_dict(dict(avg_total_pur_per_person), orient='index', columns={'Avg Total Purchase per Person'})

# Add Gender as the index name
avg_total_pur_per_person_df.index.name = 'Gender'
```

In [131]:
```
# Merge in the Avg Total Purchase per Person for the final dataframe
gender_pur_analysis_final_df = pd.merge(gender_pur_analysis_pc_app_tpv_df, avg_total_pur_per_person_df, left_index=True, right_index=True)

# Set the number formats for the columns and display dataframe
gender_pur_analysis_final_df.style.format({'Average Purchase Price': '${:.2f}', 'Total Purchase Value': '${:.2f}', 'Avg Total Purchase per Person': '${:.2f}'})
```

Out[131]:

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	$1,967.64	$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 [116]:
```
# Create bins for ages and create labels
age_bins = [0, 9.90, 14.90, 19.90, 24.90, 29.90, 34.90, 39.90, 99999]
bin_labels = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39", "40+"]

# Create a copy of the dataframe
demo_df = purchase_data_df
```

In [117]:
```
# Splits the data using pd.cut and categorizes the existing players based on the age bins
demo_df['Age Bins'] = pd.cut(demo_df['Age'],age_bins, labels=bin_labels)
```

In [132]:
```
# Grouphy on Age Bins
age_binned_df = demo_df.groupby('Age Bins')
```

In [133]:
```
# Calculate the Total Count and Percentage of Players by age range
unique_players = age_binned_df['SN'].nunique()
percent_of_players = (unique_players/total_players)
```

In [134]:
```
# Creates a DataFrame to hold the above results
age_demo_df = pd.DataFrame({"Total Count":unique_players, "Percentage of Players":percent_of_players})
```

In [135]:
```
# Delete the index name
age_demo_df.index.name = None

# Set the format of the Percentage of Players column and display dataframe
age_demo_df.style.format({'Percentage of Players': '{:.2%}'})
```

Out[135]:

	Total 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 [96]:
```
# Count the total number of purchases for each age bin and create a dataframe
age_purchase_count = age_binned['Item ID'].count()
age_purchase_count_df = pd.DataFrame.from_dict(dict(age_purchase_count), orient='index', columns={'Purchase Count'})
```

In [97]:
```
# Calculate the Average Purchase Price by age and create dataframe
age_avg_purchase_price = age_binned['Price'].mean()
age_avg_purchase_price_df = pd.DataFrame.from_dict(dict(age_avg_purchase_price), orient='index', columns={'Average Purchase Price'})
```

In [98]:
```
# Merge the Purchase Count and Average Purchase Price dataframes
age_pur_analysis_pc_app_df = pd.merge(age_purchase_count_df, age_avg_purchase_price_df, left_index=True, right_index=True)
```

In [99]:
```
# Calculate the Total Purchase Value by age and create dataframe
age_total_purchase_value = age_binned['Price'].sum()
age_total_purchase_value_df = pd.DataFrame.from_dict(dict(age_total_purchase_value), orient='index', columns={'Total Purchase Value'})
```

In [100]:
```
# Merge the Total Purchase Value into the merged dataframes
age_pur_analysis_pc_app_tpv_df = pd.merge(age_pur_analysis_pc_app_df, age_total_purchase_value_df, left_index=True, right_index=True)
```

In [101]:
```
# Calculate the Avg Total Purchase per Person by gender and create dataframe
age_avg_total_pur_per_person = (age_binned['Price'].sum()/unique_players)
age_avg_total_pur_per_person_df = pd.DataFrame.from_dict(dict(age_avg_total_pur_per_person), orient='index', columns={'Avg Total Purchase per Person'})
```

In [102]:
```
# Merge in the Avg Total Purchase per Person for the final dataframe
age_pur_analysis_final_df = pd.merge(age_pur_analysis_pc_app_tpv_df, age_avg_total_pur_per_person_df, left_index=True, right_index=True)
```

In [104]:
```
# Add Age Ranges as the index name
age_purchase_count_df.index.name = 'Age Ranges'

# Set the number formats for the columns
age_pur_analysis_final_df.style.format({'Average Purchase Price': '${:.2f}', 'Total Purchase Value': '${:.2f}', 'Avg Total Purchase per Person': '${:.2f}'})
```

Out[104]:

Age Ranges	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	$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

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 [37]:
```
# Grouphy Users
user_grouped_df = purchase_data_df.groupby('SN')
```

In [95]:
```
# Count the purchases per user and create a dataframe
tn_purchase_count = user_grouped_df['Item ID'].count()
tn_purchase_count_df = pd.DataFrame.from_dict(dict(tn_purchase_count), orient='index', columns={'Purchase Count'})
```

In [94]:
```
# Calculate the average purchase price per user and create a dataframe
tn_avg_purchase = user_grouped_df['Price'].mean()
tn_avg_purchase_df = pd.DataFrame.from_dict(dict(tn_avg_purchase), orient='index', columns={'Average Purchase Price'})
```

In [93]:
```
# Merge the Purchase Count and Average Purchase Price per user dataframe
tn_pc_app_df = pd.merge(tn_purchase_count_df, tn_avg_purchase_df, left_index=True, right_index=True)
```

In [92]:
```
# Calculate the total purchase value per user and create a dataframe
tn_total_purchase_value = user_grouped_df['Price'].sum()
tn_total_purchase_value_df = pd.DataFrame.from_dict(dict(tn_total_purchase_value), orient='index', columns={'Total Purchase Value'})
```

In [91]:
```
# Merge in the Avg Total Purchase per Person for the final dataframe
top_spenders_final_df = pd.merge(tn_pc_app_df, tn_total_purchase_value_df, left_index=True, right_index=True)
```

In [90]:
```
# Sort the total purchase value column in descending order
top_spenders_final_df.sort_values(by='Total Purchase Value', ascending=False, inplace=True)
```

In [89]:
```
# Add SN as the index name
top_spenders_final_df.index.name = 'SN'

# Set the number formats for the columns and display
top_spenders_final_df.head().style.format({'Average Purchase Price': '${:.2f}', 'Total Purchase Value': '${:.2f}'})
```

Out[89]:

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 [82]:
```
# Retrieve the Item ID, Item Name, and Item Price columns
most_popular_items_df = purchase_data_df.loc[:,['Item ID','Item Name','Price']]
```

In [83]:
```
# Group by Item ID and Item Name
most_popular_items_groupby_df = most_popular_items_df.groupby(['Item ID','Item Name'])
```

In [84]:
```
# Calculate the purchase count
most_purchase_count = most_popular_items_groupby_df['Item ID'].count()
```

In [85]:
```
# Calculate the total purchase value
mpi_total_purchase_value = most_popular_items_groupby_df['Price'].sum()
```

In [86]:
```
# Calculate the price from the purchase count and the total purchase value
mpi_item_price = mpi_total_purchase_value/mpi_purchase_count
```

In [87]:
```
# Construct dataframe from variables
mpi_final_df = pd.DataFrame({'Purchase Count':mpi_purchase_count, 'Item Price':mpi_item_price, 'Total Purchase Value':mpi_total_purchase_value})
```

In [88]:
```
# Sort the total purchase value column in descending order:
mpi_final_df.sort_values(by='Purchase Count', ascending=False, inplace=True)
```

In [76]:
```
# Set the number formats for the columns and display
mpi_final_df.head().style.format({'Item Price': '${:.2f}', 'Total Purchase Value': '${:.2f}'})
```

Out[76]:

Item ID	Item Name	Purchase Count	Item Price	Total Purchase Value
92	**Final Critic**	13	$4.61	$59.99
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
103	**Singed Scalpel**	8	$4.35	$34.80

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 [79]:
```
# Sort the above table by total purchase value in descending order
mpi_final_df.sort_values(by='Total Purchase Value', ascending=False, inplace=True)
```

In [78]:
```
# Set the number formats for the columns and display
mpi_final_df.head().style.format({'Item Price': '${:.2f}', 'Total Purchase Value': '${:.2f}'})
```

Out[78]:

Item ID	Item Name	Purchase Count	Item Price	Total Purchase Value
92	**Final Critic**	13	$4.61	$59.99
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
103	**Singed Scalpel**	8	$4.35	$34.80