In [18]:

```
# Dependencies and Setup
import pandas as pd

# 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 [25]:

```
# Define total players, use length of list of screen names "SN", count "SN" valu
es in string
total_players = len(purchase_data["SN"].value_counts())

# Create data frame in order to return values (output/return is formatted)
player_count = pd.DataFrame({"Total Players":[total_players]})
player_count
```

Out[25]:

	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 [54]:

```
# Capture unique items count (unique,) average price (mean,) purchase count (cou
nt,) and total revenue (sum)
unique_items = len((purchase_data["Item ID"]).unique())
average_price = (purchase_data["Price"]).mean()
purchase_count = (purchase_data["Purchase ID"]).count()
total_revenue = (purchase_data["Price"]).sum()

# Create data frame in order to return values (output/return is formatted)
summary_df = pd.DataFrame({"Number of Unique Items":[unique_items],
                           "Average Price":[average_price],
                           "Number of Purchases": [purchase_count],
                           "Total Revenue": [total_revenue]})

# Format average price & total revenue with $ currency and two significant figur
es
summary_df.style.format({'Average Price':"${:,.2f}",
                         'Total Revenue': '${:,.2f}'})
```

Out[54]:

	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 [55]:

```
# Group purchase_data by gender (gouping)
gender_grouping = purchase_data.groupby("Gender")

# Count total screen names "SN" by gender (nunique)
count_by_gender = gender_grouping.nunique()["SN"]

# Calculate percentage by gender, divide count by gender by total players
percentage_by_gender = count_by_gender / total_players * 100

# Create data frame in order to return values (output/return is formatted)
gender_output = pd.DataFrame({"Total Count": count_by_gender, "Percentage of Pla
yers": percentage_by_gender})

# Format table by removing index name (0)
gender_output.index.name = None

# Sort by total count in descending order, add percentage with two significant f
igures
gender_output.sort_values(["Total Count"], ascending = False).style.format({"Per
centage of Players":"{:.2f}%"})
```

Out[55]:

	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 [63]:

```
# Count total purchases by gender (count)
purchase_count = gender_stats["Purchase ID"].count()

# Calcuate average purchase price by gender (mean)
avg_purchase_price = gender_stats["Price"].mean()

# Calculate total purchase value by gender (sum)
total_purchase_value = gender_stats["Price"].sum()

# Calculate average total purchase per gender (total purchase value divided by c
ount by gender)
avg_purchase_per_gender = total_purchase_value/count_by_gender

# Create data frame in order to return values (output/return is formatted)
gender_output = pd.DataFrame({"Purchase Count": purchase_count,
                              "Average Purchase Price": avg_purchase_price
,
                              "Total Purchase Value":total_purchase_value,
                              "Avg Total Purchase per Person": avg_purchas
e_per_gender})

# Label top left index as "Gender" (replace 0 as index label)
gender_output.index.name = "Gender"

# Format average purchase price, total purchase value & avg total purchase/perso
n with $ currency and two significant figures
gender_output.style.format({"Average Purchase Price":"${:,.2f}",
                            "Total Purchase Value":"${:,.2f}",
                            "Avg Total Purchase per Person":"${:,.2f}"})
```

Out[63]:

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 [64]:

```
# Define bins for age segments and assign names
age_bins = [0, 9.99, 14.99, 19.99, 24.99, 29.99, 34.99, 39.99, 99999]
group_names = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39", "40+
"]

# Segment age values into corresponding bins
purchase_data["Age Group"] = pd.cut(purchase_data["Age"],age_bins, labels=group_
names)
purchase_data

# Create data frame in order to return values (output/return is formatted)
# Group purchase_data by age ("Age Group") (gouping)
age_grouped = purchase_data.groupby("Age Group")

# Count total players by age (nunique) #Same as total screen names "SN" by gende
r
count_by_age = age_grouped["SN"].nunique()

# Calculate percentage by age category #Same as percentage by gender
percentage_by_age = (count_by_age/total_players) * 100

# Create data frame in order to return values (output/return is formatted)
age_output = pd.DataFrame({"Total Count": count_by_age, "Percentage of Players":
percentage_by_age})

# Format the data frame with no index name in the corner
age_output.index.name = None

# Format percentage with % and two decimal places
age_output.style.format({"Percentage of Players":"{:,.2f}%"})
```

Out[64]:

	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 [66]:

```
# Count total purchases by age group (count)
purchase_count_age = age_grouped["Purchase ID"].count()

# Calculate average purchase price by age group (mean)
avg_purchase_price_age = age_grouped["Price"].mean()

# Calculate total purchase value by age group (sum)
total_purchase_value = age_grouped["Price"].sum()

# Calculate average total purchase per person in the age group (total purchase v
alue divided by total count age)
avg_purchase_by_age = total_purchase_value/total_count_age

# Create data frame in order to return values (output/return is formatted)
age_output = pd.DataFrame({"Purchase Count": purchase_count_age,
                           "Average Purchase Price": avg_purchase_price_ag
e,
                           "Total Purchase Value":total_purchase_value,
                           "Avg Total Purchase per Person": avg_purchase_b
y_age})

# Format the data frame with no index name in the corner
age_output.index.name = None
```

```
# Format with currency style
age_output.style.format({"Average Purchase Price":"${:,.2f}",
                         "Total Purchase Value":"${:,.2f}",
                         "Avg Total Purchase per Person":"${:,.2f}"})
```

Out[66]:

	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 [50]:

```
# Identify top 5 spenders by purchase count, avg purchase price, total purchase
value
# Replicate the same code as was done for gender and age groups

# Group purchase_data by screen names (gouping)
spender_stats = purchase_data.groupby("SN")

# Count total purchases by spender (count)
purchase_count_by_spender = spender_stats["Purchase ID"].count()

# Calcuate average purchase price by spender (mean)
avg_purchase_price_spender = spender_stats["Price"].mean()

# Calculate total purchase value by spender (sum)
purchase_total_spender = spender_stats["Price"].sum()

# Create data frame in order to return values (output/return is formatted)
top_spenders = pd.DataFrame({"Purchase Count": purchase_count_by_spender,
                             "Average Purchase Price": avg_purchase_price_spende
r,
                             "Total Purchase Value":purchase_total_spender})

# Sort in descending order to obtain top 5 spender names, print head
formatted_spenders = top_spenders.sort_values(["Total Purchase Value"], ascendin
g=False).head()

# Format table by assigning "SN" as index name
formatted_spenders.index.name = "SN"

# Format average purchase price & total purchase value with $ currency and two s
ignificant figures
formatted_spenders.style.format({"Average Purchase Price":"${:,.2f}",
                                 "Total Purchase Value":"${:,.2f}"})
```

Out[50]:

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 [52]:

```
# Identify top 5 items by purchase count, sorting by Item ID, item name, item pr
ice, and total purchase value
# Replicate the same code as was done for gender, age groups, and top spenders

# Create new data frame capturing relevent sort categories
items = purchase_data[["Item ID", "Item Name", "Price"]]

# Group the item data by item id and item name
item_stats = items.groupby(["Item ID","Item Name"])

# Count total purchases by item, grouped 1:1 by both item id and item name (coun
t)
purchase_count_item = item_stats["Price"].count()

# Calculate total purchase value by item (sum)
purchase_value_item = (item_stats["Price"].sum())

# Calculate individual item price (total purchase value divided by purchase coun
t)
item_price = purchase_value_item/purchase_count_item

# Create data frame in order to return values (output/return is formatted)
most_popular_items = pd.DataFrame({"Purchase Count": purchase_count_item,
                                   "Item Price": item_price,
                                   "Total Purchase Value":purchase_value_item})

# Sort in descending order top 5 item names by purchase count, print head
popular_formatted = most_popular_items.sort_values(["Purchase Count"], ascending
=False).head()

# Format item price and total purchase value with $ currency and two significant
figures
popular_formatted.style.format({"Item Price":"${:,.2f}",
                                "Total Purchase Value":"${:,.2f}"})
```

Out[52]:

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 [53]:

```
# Identify top 5 items by total purchase value, sorting by Item ID, item name, p
urchase count, and item price
# Replicate the same code as was done for top 5 items by purchase count

# Sort in descending order top 5 item names by total purchase value, print head
popular_formatted = most_popular_items.sort_values(["Total Purchase Value"],
                                                   ascending=False).head()
# Format item price and total purchase value with $ currency and two significant
figures
popular_formatted.style.format({"Item Price":"${:,.2f}",
                                "Total Purchase Value":"${:,.2f}"})
```

Out[53]:

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

In [ ]: