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 [1]:
```
# 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)

# View the purchase_data
purchase_data
```

Out[1]:

	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
...	...	...	...	...	...	...	...
775	775	Aethedru70	21	Female	60	Wolf	3.54
776	776	Iral74	21	Male	164	Exiled Doomblade	1.63
777	777	Yathecal72	20	Male	67	Celeste, Incarnation of the Corrupted	3.46
778	778	Sisur91	7	Male	92	Final Critic	4.19
779	779	Ennrian78	24	Male	50	Dawn	4.60

780 rows × 7 columns

Player Count

- Display the total number of players

In [2]:
```
# Find the total number of players using their SN
player_count = len(purchase_data["SN"].unique())

# Create a dataframe to hold the total number of players
player_count_df = pd.DataFrame({"Total Players": [player_count]})

# Display the dataframe
player_count_df
```

Out[2]:

	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 [3]:
```
# Run basic calculations to obtain number of unique items,
# average price, number of purchases, and the total revenue

Number_uniqueItem = len(purchase_data["Item ID"].unique())
Number_uniqueItem
Average_purchasePrice = round(purchase_data["Price"].mean(), 2)
Average_purchasePrice
Num_Purchases = len(purchase_data["Purchase ID"])
Num_Purchases
Total_revenue = round(purchase_data["Price"].sum(), 2)
Total_revenue

# Create a summary data frame to hold the results
Purchasing_Analysis_df = pd.DataFrame({"Number of Unique Items": [Number_uniqu
eItem], "Average Price": Average_purchasePrice, "Number of Purchases": Num_Pur
chases, "Total Revenue": Total_revenue})

# Give the displayed data cleaner formatting
Purchasing_Analysis_df["Average Price"] = Purchasing_Analysis_df["Average Pric
e"].map("${:.2f}".format)
Purchasing_Analysis_df["Total Revenue"] = Purchasing_Analysis_df["Total Revenu
e"].map("${:,.2f}".format)

# Display the summary dataframe
Purchasing_Analysis_df
```

Out[3]:

	Number of Unique Items	Average Price	Number of Purchases	Total Revenue
0	179	$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 [4]:
```
# Obtain the gender demographics - count and percentage of male, female and ot
her/non-disclosed players
Count = purchase_data.groupby("Gender").nunique()["SN"]
Count
Percent = (Count/player_count) *100
Percent

# Create a dataframe to hold the results
purchase_dataDem_df = pd.DataFrame({
    "Total Count": Count,
    "Percentage of Players": Percent
})

# Format the percentage data
purchase_dataDem_df["Percentage of Players"] = purchase_dataDem_df["Percentage
of Players"].map("{:.2f}%".format)

# Display the dataframe
purchase_dataDem_df
```

Out[4]:

	Total Count	Percentage of Players
Gender		
Female	81	14.06%
Male	484	84.03%
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 [5]:
```
# Run basic calculations to obtain purchase count,
# avg. purchase price, total purchase value and avg. purchase total per perso
n, by gender
purchase_analysis = purchase_data.groupby("Gender").count()["Purchase ID"]
purchase_analysis
Average_price = purchase_data.groupby("Gender").mean()["Price"].map("${:.2f}".
format)
Average_price
Total_price = purchase_data.groupby("Gender").sum()["Price"]
Total_price
Purchase_person = Total_price / Count
Purchase_person

# Create a summary dataframe to hold the results
purchase_analysis_df = pd.DataFrame({
    "Purchase Count": purchase_analysis,
    "Average Purchase Price": Average_price,
    "Total Purchase Value": Total_price,
    "Avg Total Purchase per Person": Purchase_person
})

# Give the displayed data cleaner formatting
purchase_analysis_df["Total Purchase Value"] = purchase_analysis_df["Total Pur
chase Value"].map("${:,.2f}".format)
purchase_analysis_df["Avg Total Purchase per Person"] = purchase_analysis_df[
"Avg Total Purchase per Person"].map("${:.2f}".format)

# Display the summary data frame
purchase_analysis_df
```

Out[5]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Avg Total Purchase per Person
Gender				
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 [6]:
```
# Establish bins for ages
bins = [0, 9.90, 14.90, 19.90, 24.90, 29.90, 34.90, 39.90, 99999]
bins
# Create labels for these bins
group_labels = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34",
                "35-39", "40+"]
group_labels

# Categorize the existing players using the age bins.
purchase_data["Age Group"] = pd.cut(purchase_data["Age"], bins, labels=group_l
abels, right=False)
purchase_data

# Calculate the numbers and percentages by age group
Count_age_group = purchase_data.groupby("Age Group").nunique()["SN"]
Count_age_group

Percent_age_group = (Count_age_group/player_count) *100
Percent_age_group

# Create a summary data frame to hold the results
purchase_dataAG_df = pd.DataFrame({
    "Total Count": Count_age_group,
    "Percentage of Players": Percent_age_group
})

# Round the percentage column to two decimal points
purchase_dataAG_df["Percentage of Players"] = purchase_dataAG_df["Percentage o
f Players"].map("{:.2f}%".format)

# Display Age Demographics Table
purchase_dataAG_df
```

Out[6]:

	Total Count	Percentage of Players
Age Group		
<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 [7]:
```
# Using the age group bin of the purchase dataframe by age (purchase_data) fro
m the previous cell.

# Run basic calculations to obtain purchase count,
# avg. purchase price, total purchase value and avg. purchase total per person
in the table below
purchase_dataAgeGroup = purchase_data.groupby("Age Group")
purchase_dataAgeGroup
purchase_count = purchase_dataAgeGroup["Purchase ID"].count()
avg_purchase_price = purchase_dataAgeGroup["Price"].mean()
total_purchase_value = purchase_dataAgeGroup["Price"].sum()
total_purchase_value
avg_purchase_per_person = total_purchase_value / Count_age_group
avg_purchase_per_person

# Create a summary data frame to hold the results
purchase_dataAge_df = pd.DataFrame({
    "Purchase Count": purchase_count,
    "Average Purchase Price": avg_purchase_price,
    "Total Purchase Value": total_purchase_value,
    "Average Purchase Per Person": avg_purchase_per_person
})

# Give the displayed data cleaner formatting
purchase_dataAge_df["Average Purchase Price"] = purchase_dataAge_df["Average P
urchase Price"].map("${:.2f}".format)
purchase_dataAge_df["Total Purchase Value"] = purchase_dataAge_df["Total Purch
ase Value"].map("${:.2f}".format)
purchase_dataAge_df["Average Purchase Per Person"] = purchase_dataAge_df["Aver
age Purchase Per Person"].map("${:.2f}".format)

# Display the summary data frame
purchase_dataAge_df
```

Out[7]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Average Purchase Per Person
Age Group				
<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 [8]:
```
# Run basic calculations to obtain the top spenders by displaying their purcha
se count,
# average purchase price and total purchase price as shown in the table below
Spenders = purchase_data.groupby("SN").nunique()["Purchase ID"]
Spenders.head()
Purchase_price = purchase_data.groupby("SN").mean()["Price"]
#Purchase_price
Purchase_value = purchase_data.groupby("SN").sum()["Price"]
Purchase_value

# Create a summary data frame to hold the results
Top_Spenders_df= pd.DataFrame ({
    "Purchase Count":Spenders,
    "Average Purchase Price":Purchase_price,
    "Total Purchase Price":Purchase_value

})

# Sort the total purchase value column in descending order
Top_Spenders_df = Top_Spenders_df.sort_values(by=['Total Purchase Price'], asc
ending=False)

# Give the displayed data cleaner formatting
Top_Spenders_df["Average Purchase Price"] = Top_Spenders_df["Average Purchase
 Price"].map("${:.2f}".format)
Top_Spenders_df["Total Purchase Price"] = Top_Spenders_df["Total Purchase Pric
e"].map("${:.2f}".format)

# Display a preview of the summary data frame
Top_Spenders_df.head()
```

Out[8]:

	Purchase Count	Average Purchase Price	Total Purchase Price
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

- Retrieve the Item ID, Item Name, and Item Price columns
- Group by Item ID and Item Name. Perform calculations to obtain purchase count, average 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 [9]:
```
# Retrieve the Item ID, Item Name, and Item Price columns
popular_items = purchase_data[["Item ID", "Item Name", "Price"]]
popular_items.head()

# Group by Item ID and Item Name.
# Perform calculations to obtain purchase count, average item price, and total
purchase value
purchase_count = popular_items.groupby(["Item ID", "Item Name"]).count()["Pric
e"]
purchase_count
total_purchase_value = popular_items.groupby(['Item ID', 'Item Name']).sum()[
"Price"]
total_purchase_value
avg_item_price = total_purchase_value / purchase_count
avg_item_price

# Create a summary data frame to hold the results
most_popular_df= pd.DataFrame({
    "Purchase Count": purchase_count,
    "Item Price": avg_item_price,
    "Total Purchase Value": total_purchase_value
})

# Sort the purchase count column in descending order
most_popularIT_df = most_popular_df.sort_values(by=['Purchase Count'], ascendi
ng = False)

# Give the displayed data cleaner formatting
most_popularIT_df["Item Price"] = most_popular_df["Item Price"].map("${:.2f}".
format)
most_popularIT_df["Total Purchase Value"] = most_popular_df["Total Purchase Va
lue"].map("${:.2f}".format)

# Display a preview of the summary data frame
most_popularIT_df.head()
```

Out[9]:

		Purchase Count	Item Price	Total Purchase Value
Item ID	**Item Name**			
92	**Final Critic**	13	$4.61	$59.99
178	**Oathbreaker, Last Hope of the Breaking Storm**	12	$4.23	$50.76
145	**Fiery Glass Crusader**	9	$4.58	$41.22
132	**Persuasion**	9	$3.22	$28.99
108	**Extraction, Quickblade Of Trembling Hands**	9	$3.53	$31.77

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 [10]:
```
# Sort the above table (listing items by their popularity) by total purchase value in descending order
most_profitable_df = most_popular_df.sort_values(by=['Total Purchase Value'], ascending = False)
most_profitable_df

# Give the displayed data cleaner formatting
most_profitable_df["Item Price"] = most_profitable_df["Item Price"].map("${:.2f}".format)
most_profitable_df["Total Purchase Value"] = most_profitable_df["Total Purchase Value"].map("${:.2f}".format)

# Display a preview of the data frame
most_profitable_df.head()
```

Out[10]:

		Purchase Count	Item Price	Total Purchase Value
Item ID	Item Name			
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

In [ ]: