In [1]:

```
# Dependencies
import pandas as pd
import numpy as np
```

In [2]:

```
# Create a reference to data and import it into Pandas Dataframe
data_path = "purchase_data.json"
heroes = pd.read_json(data_path)
heroes.head()
```

Out[2]:

	Age	Gender	Item ID	Item Name	Price	SN
0	38	Male	165	Bone Crushing Silver Skewer	3.37	Aelalis34
1	21	Male	119	Stormbringer, Dark Blade of Ending Misery	2.32	Eolo46
2	34	Male	174	Primitive Blade	2.46	Assastnya25
3	21	Male	92	Final Critic	1.36	Pheusrical25
4	23	Male	63	Stormfury Mace	1.27	Aela59

In [3]:

```
heroes.columns
```

Out[3]:

```
Index(['Age', 'Gender', 'Item ID', 'Item Name', 'Price', 'SN'], dtyp
e='object')
```

In [4]:

```
# PLAYER COUNT
# Total number of Players
players_count = heroes["SN"].count()
pd.DataFrame([players_count], columns = ["Total Players"])
```

Out[4]:

	Total Players
0	780

In [5]:

```
# PURCHASING ANALYSIS (TOTAL)

# the number of unique items
unique_items = len(heroes["Item ID"].value_counts())
unique_items
```

Out[5]:

```
183
```

In [6]:

```
# the average of the purchase 'price'
average_price = int(round(heroes["Price"].mean()))
average_price
```

Out[6]:

```
3
```

In [7]:

```
# the total number of purchases
total_purchases = heroes["Price"].count()
total_purchases
```

Out[7]:

```
780
```

In [8]:

```
# total revenue
total_revenue = int(round(heroes["Price"].sum()))
total_revenue
```

Out[8]:

```
2286
```

In [9]:

```
# Purchasing Analysis Summary Table
p_analysis_df = pd.DataFrame({"Number of Unique Items": [unique_items], "Average
Purchase Price": [average_price], "Total Number of Purchases": [total_purchases]
, "Total Revenue": [total_revenue]}, columns = ["Number of Unique Items", "Avera
ge Purchase Price", "Total Number of Purchases", "Total Revenue"])
p_analysis_df.style.format({"Average Purchase Price": "${:.2f}", "Total Revenue"
: "${:.2f}"})
```

Out[9]:

	Number of Unique Items	Average Purchase Price	Total Number of Purchases	Total Revenue
0	183	$3.00	780	$2286.00

In [10]:

```
# GENDER DEMOGRAPHICS

# Group data by gender and check and remove duplicates

gender_groups = heroes.groupby("SN")
gender_groups["Gender"].count()
```

Out[10]:

```
SN
Adairialis76        1
Aduephos78          3
Aeduera68           3
Aela49              1
Aela59              1
Aelalis34           2
Aelin32             1
Aeliriam77          2
Aeliriarin93        1
Aeliru63            2
Aellyria80          1
Aellyrialis39       1
Aellysup38          1
Aelollo59           1
Aenarap34           1
Aenasu69            1
Aeral43             1
Aeral85             1
Aeral97             1
Aeri84              2
Aerillorin70        1
Aerithllora36       3
Aerithnucal56       2
```

```
Aerithnuphos61     1

Aerithriaphos45    1
Aesty51            1
Aesur96            1
Aethe80            1
Aethedru70         1
Aidain51           2
                  ..
Undjaskla97        1
Undjasksya56       1
Undotesta33        1
Wailin72           1
Whaestysu86        1
Yadacal26          1
Yadaisuir65        2
Yadanun74          3
Yalaeria91         1
Yaliru88           1
Yalo71             1
Yalostiphos68      1
Yaralnura48        2
Yararmol43         1
Yarirarn35         1
Yaristi64          1
Yarithllodeu72     1
Yarithphos28       1
Yarithsurgue62     2
Yarmol79           1
Yarolwen77         2
Yasriphos60        3
Yasrisu92          1
Yasur35            1
Yasur85            1
Yasurra52          1
Yathecal72         2
Yathecal82         1
Zhisrisu83         2
Zontibe81          1
Name: Gender, Length: 573, dtype: int64
```

In [11]:

```
gender = heroes[["SN", "Gender"]]
modify_gender = gender.drop_duplicates()
modify_gender.head()
```

Out[11]:

	SN	Gender
0	Aelalis34	Male
1	Eolo46	Male
2	Assastnya25	Male
3	Pheusrical25	Male
4	Aela59	Male

In [12]:

```
total = modify_gender["SN"].count()
total
```

Out[12]:

```
573
```

In [13]:

```
# count and percentage of male, female and other players
males = modify_gender[modify_gender['Gender']=="Male"]["SN"].nunique()
females = modify_gender[modify_gender['Gender']=="Female"]["SN"].nunique()
other = modify_gender[modify_gender["Gender"]=='Other / Non-Disclosed']["SN"].nu
nique()

# percentage of male, female and other players
malepercent = ((males/total)*100)
femalepercent = ((females/total)*100)
otherpercent = ((other/total)*100)

# summary of gender demographics
gender_demo_df = pd.DataFrame({"Gender": ["Male", "Female", "Other / Non-Disclos
ed"], "Percentage of Players": [malepercent, femalepercent, otherpercent],
                                        "Total Count": [males, females, other]},
columns =
                                        ["Gender", "Percentage of Players", "Tot
al Count"])
gender_demo_df.style.format({"Percentage of Players": "{:.2f}%"})
```

Out[13]:

	Gender	Percentage of Players	Total Count
0	Male	81.15%	465
1	Female	17.45%	100
2	Other / Non-Disclosed	1.40%	8

In [14]:

```
# PURCHASING ANALYSIS (GENDER)
gen_purchase = heroes[["SN", "Gender", "Price"]]
mod_gender = gen_purchase.drop_duplicates()
```

In [15]:

```
# purchase count by gender
malepurch = mod_gender[mod_gender["Gender"] == "Male"]["Price"].count()
femalepurch = mod_gender[mod_gender["Gender"] == "Female"]["Price"].count()
otherpurch = mod_gender[mod_gender["Gender"] == "Other / Non-Disclosed"]["Price"
].count()

# average purchase price by gender
mpriceav = mod_gender[mod_gender["Gender"] == "Male"]["Price"].mean()
fpriceav = mod_gender[mod_gender["Gender"] == "Female"]["Price"].mean()
opriceav = mod_gender[mod_gender["Gender"] == "Other / Non-Disclosed"]["Price"].
mean()

# total purchase value by gender
mpricetotal = mod_gender[mod_gender["Gender"] == "Male"]['Price'].sum()
fpricetotal = mod_gender[mod_gender["Gender"] == "Female"]['Price'].sum()
opricetotal = mod_gender[mod_gender["Gender"] == "Other / Non-Disclosed"]['Price
'].sum()

# normalized totals
male_norm = mpricetotal/males
female_norm = fpricetotal/females
other_norm = opricetotal/other

# summary of purchasing analysis (gender)
gender_purchasing_df = pd.DataFrame ({"Gender": ["Male", "Female", "Other / Non-
Disclosed"], "Purchase Count": [malepurch, femalepurch, otherpurch],
                                        "Average Purchase Price": [mpriceav, fpr
iceav, opriceav], "Total Purchase Value": [mpricetotal, fpricetotal, opricetotal
],
                                "Normalized Totals": [male_norm, female_norm, ot
her_norm]}, columns =
                                        ["Gender", "Purchase Count", "Average Pu
rchase Price", "Total Purchase Value", "Normalized Totals"])

gender_purchasing_df.style.format({"Average Purchase Price": "${:.2f}", "Total P
urchase Value": "${:.2f}", "Normalized Totals": "${:.2f}"})
```

Out[15]:

	Gender	Purchase Count	Average Purchase Price	Total Purchase Value	Normalized Totals
0	Male	631	$2.95	$1862.03	$4.00
1	Female	135	$2.83	$381.55	$3.82
2	Other / Non-Disclosed	11	$3.25	$35.74	$4.47

In [16]:

```
# Finding out the maximum and minimum age
print(heroes["Age"].max())
print (heroes["Age"].min())
```

```
45
7
```

In [17]:

```
# AGE DEMOGRAPHICS (Age group bins)
ad = heroes[["SN", "Age"]]
modify_ad = ad.drop_duplicates()

# Create age group counts for players
ten = modify_ad[modify_ad["Age"] < 10].count()[0]
ten_more = modify_ad[(modify_ad["Age"] >= 10) & (modify_ad["Age"] <= 14)].count(
)[0]
teens = modify_ad[(modify_ad["Age"] >= 15) & (modify_ad["Age"] <= 19)].count()[0
]
twenty = modify_ad[(modify_ad["Age"] >= 20) & (modify_ad["Age"] <= 24)].count()[
0]
twenty_more = modify_ad[(modify_ad["Age"] >= 25) & (modify_ad["Age"] <= 29)].cou
nt()[0]
thirty = modify_ad[(modify_ad["Age"] >= 30) & (modify_ad["Age"] <= 34)].count()[
0]
thirty_more = modify_ad[(modify_ad["Age"] >= 35) & (modify_ad["Age"] <= 39)].cou
nt()[0]
forty = modify_ad[modify_ad["Age"] >= 40].count()[0]
ages = [ten, ten_more, teens, twenty, twenty_more, thirty, thirty_more, forty]

# Create age group precent for players
percent_ten = round((ten/players_count)*100)
percent_teen1 = round((ten_more/players_count)*100)
percent_teen2 = round((teens/players_count)*100)
percent_twenty = round((twenty/players_count)*100)
percent_twenty2 = round((twenty_more/players_count)*100)
percent_thirty = round((thirty/players_count)*100)
percent_thirty2 = round((thirty_more/players_count)*100)
percent_forty = round((forty/players_count)*100)
percents_a = [percent_ten, percent_teen1, percent_teen2, percent_twenty, percent
_twenty2, percent_thirty, percent_thirty2, percent_forty]

# Create dataframe for age demography summary
age_demograph = {
        "Percent of Players": percents_a,
        "Total Count": ages
    }
age_demo_df = pd.DataFrame(age_demograph)
age_demo_df.index = (["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39
", "40+"])
age_demo_df
```

Out[17]:

	Percent of Players	Total Count
<10	2.0	19
10-14	3.0	23
15-19	13.0	100
20-24	33.0	259
25-29	11.0	87
30-34	6.0	47
35-39	3.0	27
40+	1.0	11

In [28]:

```
ad2 = heroes[["Age", "Price"]]
mod_ad2 = ad2.drop_duplicates()

# Purchase Count
purchase_10 = mod_ad2[mod_ad2["Age"] < 10].count()[0]
purchase_14 = mod_ad2[(mod_ad2["Age"] >= 10) & (mod_ad2["Age"] <= 14)].count()[0
]
purchase_19 = mod_ad2[(mod_ad2["Age"] >= 15) & (mod_ad2["Age"] <= 19)].count()[0
]
purchase_24 = mod_ad2[(mod_ad2["Age"] >= 20) & (mod_ad2["Age"] <= 24)].count()[0
]
purchase_29 = mod_ad2[(mod_ad2["Age"] >= 25) & (mod_ad2["Age"] <= 29)].count()[0
]
purchase_34 = mod_ad2[(mod_ad2["Age"] >= 30) & (mod_ad2["Age"] <= 34)].count()[0
]
purchase_39 = mod_ad2[(mod_ad2["Age"] >= 35) & (mod_ad2["Age"] <= 39)].count()[0
]
purchase_40 = mod_ad2[mod_ad2["Age"] >= 40].count()[0]
purchases_a = [purchase_10, purchase_14, purchase_19, purchase_24, purchase_29,
purchase_34, purchase_39, purchase_40]

# Total Purchase Value
total_10 = mod_ad2.loc[mod_ad2['Age'] < 10, 'Price'].sum()
total_14 = mod_ad2.loc[(mod_ad2['Age'] >= 10) & (mod_ad2['Age'] <=14), 'Price'].
sum()
total_19 = mod_ad2.loc[(mod_ad2['Age'] >= 15) & (mod_ad2['Age'] <=19), 'Price'].
sum()
total_24 = mod_ad2.loc[(mod_ad2['Age'] >= 20) & (mod_ad2['Age'] <=24), 'Price'].
sum()
total_29 = mod_ad2.loc[(mod_ad2['Age'] >= 25) & (mod_ad2['Age'] <=29), 'Price'].
sum()
total_34 = mod_ad2.loc[(mod_ad2['Age'] >= 30) & (mod_ad2['Age'] <=34), 'Price'].
sum()
sum()
total_39 = mod_ad2.loc[(mod_ad2['Age'] >= 35) & (mod_ad2['Age'] <=39), 'Price'].
sum()
total_40 = mod_ad2.loc[mod_ad2['Age'] >= 40, 'Price'].sum()
totals_a = [total_10, total_14, total_19, total_24, total_29, total_34, total_39
, total_40]

# Average Purchase Price
avg_price_a = [total_10/purchase_10, total_14/purchase_14, total_19/purchase_19,
total_24/purchase_24, total_29/purchase_29,
              total_34/purchase_34, total_39/purchase_39, total_40/purchase_40]

# Normalized Totals
norms_a = [total_10/ten, total_14/ten_more, total_19/teens, total_24/twenty, tot
al_29/twenty_more, total_34/thirty,
           total_39/thirty_more, total_40/forty]

# Creating dictionary
puchase_analysis_a = {
    "Purchase Count": purchases_a,
    "Average Purchase Price": avg_price_a,
    "Total Purchase Value": totals_a,
    "Normalized Totals": norms_a
}

# Creating DataFrame & setting index
purchase_analysis_a_df = pd.DataFrame(puchase_analysis_a)
purchase_analysis_a_df = purchase_analysis_a_df[['Purchase Count', 'Average Purc
hase Price', 'Total Purchase Value',
                                                 'Normalized Totals']]
purchase_analysis_a_df.index = (["<10", "10-14","15-19","20-24","25-29","30-34",
"34-39","40+"])

# Formatting Prices
purchase_analysis_a_df.style.format({"Average Purchase Price": "${:.2f}", "Norma
lized Totals": "${:.2f}",
                                     "Total Purchase Value":"${:.2f}"})
```

```
Out[28]:
```

	Purchase Count	Average Purchase Price	Total Purchase Value	Normalized Totals
<10	24	$2.94	$70.45	$3.71
10-14	34	$2.82	$95.71	$4.16
15-19	123	$2.87	$352.86	$3.53
20-24	265	$2.91	$771.23	$2.98
25-29	106	$3.01	$318.63	$3.66
30-34	63	$3.07	$193.63	$4.12
34-39	40	$2.82	$112.80	$4.18
40+	17	$3.16	$53.75	$4.89

In [38]:

```
# TOP SPENDERS

sn_total_purchase = heroes.groupby("SN")["Price"].sum().to_frame()
sn_purchase_count = heroes.groupby("SN")["Price"].count().to_frame()
sn_purchase_avg = heroes.groupby("SN")["Price"].mean().to_frame()

sn_total_purchase.columns=["Total Purchase Value"]
join_one = sn_total_purchase.join(sn_purchase_count, how="left")
join_one.columns=["Total Purchase Value", "Purchase Count"]

join_two = join_one.join(sn_purchase_avg, how="inner")
join_two.columns=["Total Purchase Value", "Purchase Count", "Average Purchase Price"]

top_spenders_df = join_two[["Purchase Count", "Average Purchase Price", "Total Purchase Value"]]
top_spenders_final = top_spenders_df.sort_values('Total Purchase Value', ascending=False).head()
top_spenders_final.style.format({"Average Purchase Price": "${:.2f}", "Total Purchase Value": "${:.2f}"})
```

Out[38]:

	Purchase Count	Average Purchase Price	Total Purchase Value
SN			
Undirrala66	5	$3.41	$17.06
Saedue76	4	$3.39	$13.56
Mindimnya67	4	$3.18	$12.74
Haellysu29	3	$4.24	$12.73
Eoda93	3	$3.86	$11.58

In [42]:

```
# MOST POPULAR ITEMS

# Merge dataframe to find purchase count, total purchase value for items
premerge1 = heroes.groupby("Item Name").sum().reset_index()
premerge2 = heroes.groupby("Item ID").sum().reset_index()
premerge3 = heroes.groupby("Item Name").count().reset_index()

# Merge dataframes
merge1 = pd.merge(premerge1, premerge2, on="Price")
merge2 = pd.merge(premerge3, merge1, on="Item Name")

# Create final dataframe by manipulating data
merge2["Gender"] = (merge2["Price_y"]/merge2["Item ID"]).round(2)

merge2_renamed = merge2.rename(columns={"Age": "Purchase Count", "Gender": "Item
Price", "Item ID": "null", "Price_y": "Total Purchase Value", "Item ID_y": "Item
ID"})

# Columns needed to look into for top 5 results
clean_df = merge2_renamed[["Item ID", "Item Name", 'Purchase Count', "Item Price
", "Total Purchase Value"]]

prefinal_df = clean_df.set_index(['Item Name', 'Item ID'])
popular_items_final = prefinal_df.sort_values("Purchase Count", ascending=False)
.head(5)
popular_items_final.style.format({"Item Price": "${:.2f}", "Total Purchase Value
": "${:.2f}"})
```

Out[42]:

		Purchase Count	Item Price	Total Purchase Value
Item Name	**Item ID**			
Arcane Gem	**84**	11	$2.23	$24.53
Betrayal, Whisper of Grieving Widows	**39**	11	$2.35	$25.85
Trickster	**31**	9	$2.07	$18.63
Woeful Adamantite Claymore	**175**	9	$1.24	$11.16
Serenity	**13**	9	$1.49	$13.41

In [43]:

```
# MOST PROFITABLE ITEMS

# Use prefinal dataframe to generate information on most profitable items
profit_items_final = prefinal_df.sort_values('Total Purchase Value', ascending=False).head()
profit_items_final.style.format({"Item Price": "${:.2f}", "Total Purchase Value": "${:.2f}"})
```

Out[43]:

		Purchase Count	Item Price	Total Purchase Value
Item Name	**Item ID**			
Retribution Axe	**34**	9	$4.14	$37.26
Spectral Diamond Doomblade	**115**	7	$4.25	$29.75
Orenmir	**32**	6	$4.95	$29.70
Singed Scalpel	**103**	6	$4.87	$29.22
Splitter, Foe Of Subtlety	**107**	8	$3.61	$28.88