

Pandas: Heroes Of Pymoli - Analysis Report

- Observed Trend 1: Mourning Blade is the most profitable item for file 2, Retribution Axe is the most profitable item for file 1
- Observed Trend 2: most people only buy 1 item and male is the main buyer for both file 1 & 2
- Observed Trend 3: age group(20-24) consumed the most items for file 1 & 2

In [35]: `import pandas as pd`

In [36]: `# Read the JSON file`

```
fpath = input("Please choice file (1) or file (2)?")

if (fpath == "1"):
    JS = pd.read_json("purchase_data.json")
elif (fpath == "2"):
    JS = pd.read_json("purchase_data2.json")
else:
    print("no such file")
JS.head()

ee =JS.to_excel("test"+fpath+".xlsx", "Sheet1")
```

Please choice file (1) or file (2)?1

Player Count

In [37]: `# Total Number of Players`

```
Player_Count = pd.DataFrame([{"Total Players":JS["SN"].nunique()}])
Player_Count
```

Out[37]:

	Total Players
0	573

Purchasing Analysis (Total)

```
In [38]: #Number of Unique Items
UItem = JS["Item ID"].nunique()
#Average Purchase Price
AvePur = JS["Price"].mean()
#Total Number of Purchases
NPur = JS["Price"].count()
#Total Revenue
TRev = JS["Price"].sum()

PurAnsys = pd.DataFrame([{"Number of Unique Items":UItem, "Average Price":"$"+str
PurAnsys = PurAnsys[["Number of Unique Items","Average Price","Number of Purchase
PurAnsys
```

```
Out[38]:
```

	Number of Unique Items	Average Price	Number of Purchases	Total Revenue
0	183	\$2.93	780	\$2286.33

Gender Demographics

```
In [39]: TotalC = JS["Gender"].count()
MaleC = JS["Gender"].value_counts()
MaleP = round(MaleC / TotalC*100,2)

# Creating a new DataFrame using both duration and count
GenderDemo = pd.DataFrame({"Percentage of players":MaleP,"Total Count":MaleC})
GenderDemo

#Alternative Method
#TotalC = JS["Gender"].count()
#MaleC =pd.DataFrame(JS["Gender"].value_counts())
#MaleP =pd.DataFrame(round(MaleC / TotalC*100,2))
#MaleC = MaleC.reset_index().rename(columns={'index':'Gender','Gender':'Total
#MaleP = MaleP.reset_index().rename(columns={'index':'Gender','Gender':'Perce
#GenderDemo = pd.merge(MaleP,MaleC,how='left',on='Gender')
#GenderDemo
```

```
Out[39]:
```

	Percentage of players	Total Count
Male	81.15	633
Female	17.44	136
Other / Non-Disclosed	1.41	11

Purchasing Analysis (Gender)

The below each broken by gender

```
In [40]: JSMax = JS['Price'].max()
JSMin = JS['Price'].min()
JSMean = JS['Price'].mean()
JSStd = JS['Price'].std()
JSN = (JS['Price'] - JSMin)/(JSMax-JSMin)
JS['JSNor']=JSN
JS.head()
```

```
Out[40]:
```

	Age	Gender	Item ID	Item Name	Price	SN	JSNor
0	38	Male	165	Bone Crushing Silver Skewer	3.37	Aelalis34	0.596939
1	21	Male	119	Stormbringer, Dark Blade of Ending Misery	2.32	Eolo46	0.329082
2	34	Male	174	Primitive Blade	2.46	Assastnya25	0.364796
3	21	Male	92	Final Critic	1.36	Pheusrical25	0.084184
4	23	Male	63	Stormfury Mace	1.27	Aela59	0.061224

```
In [41]: JSGroup = JS.groupby(['Gender'])
#Purchase Count
PCbyG = JSGroup['Price'].count()
#Average Purchase Price
APbyG = round(JSGroup['Price'].mean(),2)
#Total Purchase Value
TPbyG = JSGroup['Price'].sum()
#Normalized Totals
TPbyGN = JSGroup['JSNor'].sum()

PAbyG = pd.DataFrame({"Purchase Count":PCbyG,"Average Purchase Price":APbyG,"Total Purchase Value":TPbyG,"Normalized Totals":TPbyGN})
PAbyG = PAbyG[["Purchase Count","Average Purchase Price","Total Purchase Value","Normalized Totals"]]
PAbyG
```

```
Out[41]:
```

	Purchase Count	Average Purchase Price	Total Purchase Value	Normalized Totals
Gender				
Female	136	2.82	382.91	61.946429
Male	633	2.95	1867.68	310.125000
Other / Non-Disclosed	11	3.25	35.74	6.227041

Age Demographics

The below each broken into bins of 4 years (i.e. <10, 10-14, 15-19, etc.)

```

In [42]: # Create the bins in which Data will be held
JS["Age"]=pd.to_numeric(JS['Age'])
bins = [0,9,14,19,24,29,34,39,40]
group_names = ['<10','10-14','15-19','20-24','25-29','30-34','35-39','40+']
JS["Age Group"] = pd.cut(JS["Age"],bins,labels=group_names)
JSgrAge = JS.groupby(['Age Group'])

TCbyA = JS["Age Group"].count()
VCbyA =JS["Age Group"].value_counts()
PCbyA =round(VCbyA / TCbyA*100,2)

# Creating a new DataFrame using both duration and count
AgeDemo = pd.DataFrame({"Percentage of players":PCbyA,"Total Count":VCbyA})
AgeDemo = AgeDemo.sort_index()
AgeDemo

```

```

Out[42]:

```

	Percentage of players	Total Count
<10	3.60	28
10-14	4.50	35
15-19	17.12	133
20-24	43.24	336
25-29	16.09	125
30-34	8.24	64
35-39	5.41	42
40+	1.80	14

```

In [43]: JSgrAge = JS.groupby(['Age Group'])

#Purchase Count
PCbyA = JSgrAge['Price'].count()
#Average Purchase Price
APbyA = round(JSgrAge['Price'].mean(),2)
#Total Purchase Value
TPbyA = JSgrAge['Price'].sum()
#Normalized Totals
#TPbyAN = JSgrAge['Price'].sum(normalize=True)
MaxbyA = JSgrAge['Price'].max()
MinbyA = JSgrAge['Price'].min()
TPbyAN = round((TPbyA-APbyA) / (MaxbyA-MinbyA),2)

ADbyG = pd.DataFrame({"Purchase Count":PCbyA,"Average Purchase Price":APbyA,"Total Purchase Value":TPbyA,"Normalized Totals":TPbyAN})
ADbyG = ADbyG[["Purchase Count","Average Purchase Price","Total Purchase Value","Normalized Totals"]]
ADbyG

```

Out[43]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Normalized Totals
--	----------------	------------------------	----------------------	-------------------

Age Group				
<10	28	2.98	83.46	20.85
10-14	35	2.77	96.95	26.31
15-19	133	2.91	386.42	97.83
20-24	336	2.91	978.77	248.94
25-29	125	2.96	370.33	93.72
30-34	64	3.08	197.25	49.53
35-39	42	2.84	119.40	32.47
40+	14	3.22	45.11	14.85

Top Spenders

Identify the the top 5 spenders in the game by total purchase value, then list (in a table):

```

In [44]: #SN
GroupbySN = JS.groupby(['SN'])

#Total Purchase Value by SN
TPbyS =GroupbySN['Price'].sum()
#TPbyS = TPbyA.sort_values('Price', ascending=False)
#Purchase Count
PCbyS =GroupbySN['Price'].count()
#Average Purchase Price
APbyS =GroupbySN['Price'].mean()

TopS = pd.DataFrame({"Purchase Count":PCbyS,"Average Purchase Price":round(APbyS,
TopS = TopS[["Purchase Count","Average Purchase Price","Total Purchase Value"]]
TopS = TopS.sort_values('Total Purchase Value',ascending=False)
TopS.head(5)

```

```

Out[44]:

```

	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

Most Popular Items

Identify the 5 most popular items by purchase count, then list (in a table):

```

In [45]: #Item ID
GroupbyID = JS.groupby(['Item ID','Item Name'])

#Total Purchase Value by Item
TPbyI =GroupbyID['Price'].sum()
#TPbyS = TPbyA.sort_values('Price', ascending=False)
#Purchase Count
PCbyI =GroupbyID['Price'].count()
#Average Purchase Price
APbyI =GroupbyID['Price'].mean()
#Item Price
IPbyI =GroupbyID['Price']

TopI = pd.DataFrame({"Purchase Count":PCbyI,"Item Price":APbyI,"Total Purchase Value":TPbyI})
TopI = TopI[["Purchase Count","Item Price","Total Purchase Value"]]
TopI = TopI.sort_values('Purchase Count',ascending=False)
TopI.head()

```

```

Out[45]:

```

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

Most Profitable Items

Identify the 5 most profitable items by total purchase value, then list (in a table):

```

In [46]: #Item ID
GroupbyID = JS.groupby(['Item ID','Item Name'])

#Total Purchase Value by Item
TPbyI =GroupbyID['Price'].sum()
#TPbyS = TPbyA.sort_values('Price', ascending=False)
#Purchase Count
PCbyI =GroupbyID['Price'].count()
#Average Purchase Price
APbyI =GroupbyID['Price'].mean()
#Item Price
IPbyI =GroupbyID['Price']

TopI = pd.DataFrame({"Purchase Count":PCbyI,"Item Price":APbyI,"Total Purchase Value":TPbyI})
TopI = TopI[["Purchase Count","Item Price","Total Purchase Value"]]
TopI = TopI.sort_values('Total Purchase Value',ascending=False)
TopI.head()

```

```

Out[46]:

```

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