HeroesOfPymoli

January 10, 2018

1 Data Analysis for Heroes of Pymoli

2 Observed Trend (1)

- Only 1 of the 6 most popular items appear in the list of the 18 lowest priced items/items prices in the bottom 10% of item prices. (See Popular Items and Analysis(B) - Lowest Priced)
- Males not only the make up over 80% of the players of this game, they are also responsible for over 80% of the revenue. (See Gender Demographics and Analysis(C) - Gender Purcahse Totals)

3 Observed Trend (2)

- Out of the 573 players, each player has spend under 20 dollars on items.
- The 20-24 age bracket spends has generated more revenue than any other bracket, but the 25-29 age group, on average, purchases more expensive items.

4 Observed Trend (3)

- The Retribution Axe not only appears in the most popular item list, it is priced almost 2 dollars more than other popular items on that list and is the item that has generated the most revenue.
- Other than the Retribution Axe, the most popular items list generates items that are priced below the average purcahse price.

In [14]:
```
#Dependencies and file read
import pandas as pd
import numpy as np
import os

file = os.path.join('Resources', 'purchase_data.json')

pur_data = pd.read_json(file)

#view the data
pur_data.head()
```

```
Out[14]:    Age Gender  Item ID                                  Item Name  Price  \
        0   38   Male      165                  Bone Crushing Silver Skewer   3.37
        1   21   Male      119  Stormbringer, Dark Blade of Ending Misery   2.32
        2   34   Male      174                              Primitive Blade   2.46
        3   21   Male       92                                 Final Critic   1.36
        4   23   Male       63                               Stormfury Mace   1.27

                    SN
        0     Aelalis34
        1        Eolo46
        2   Assastnya25
        3  Pheusrical25
        4        Aela59
```

4.1 Player Count

```
In [15]: #Find player count by finding unique screen names and finding the length of that list
         player_count = len(pur_data['SN'].unique())

         # DataFrame creation for player count
         players_df = pd.DataFrame([{'Total Players': player_count}])
         #gets rid of number index and resets to Total Players
         players_df.set_index('Total Players', inplace = True)
         players_df
```

```
Out[15]: Empty DataFrame
         Columns: []
         Index: [573]
```

4.2 Purchasing Analysis (Total)

```
In [16]: #code for inspecting data
         #pur_data['Item ID'].value_counts()
         #unique_items = pd.DataFrame(pur_data['Item ID'].unique())
         #len(unique_items)

         #creates a df but only keeping last occurance of Item ID
         no_dup_items = pur_data.drop_duplicates(['Item ID'], keep = 'last')
         #counts items by unique ID
         total_unique = len(no_dup_items)
         #finds the number of total purchases by counting occurances of price
         total_pur = pur_data['Price'].count()
         #calculates total revenue for table by summing occurance of price and below calc
         total_rev = round(pur_data['Price'].sum(),2)
         #calculates total_rev
         avg_price = round(total_rev/total_pur, 2)

         #creates Purchase Analysis DataFrame
```

```
pur_analysis = pd.DataFrame([{

    "Number of Unique Items": total_unique,
    'Average Purchase Price': avg_price,
    'Total Purchases': total_pur,
    'Total Revenue': total_rev
}])

#format Purchases Analysis Table
pur_analysis.style.format({'Average Purchase Price': '${:.2f}', 'Total Revenue': '${:
```

Out[16]: <pandas.io.formats.style.Styler at 0x10df9f07630>

4.3 Gender Demographics

In [17]:
```
# Gender Demographics

# Percentage and Count of Male Players
# Percentage and Count of Female Players
# Percentage and Count of Other / Non-Disclosed

#creates df of unique player names by only keeping the last occurance
no_dup_players = pur_data.drop_duplicates(['SN'], keep ='last')

#counts gender values from the df with no duplicate screen names
gender_counts = no_dup_players['Gender'].value_counts().reset_index()
#adds column for % of players using player count from first table and gender_count
#column which is a count from line above
gender_counts['% of Players'] = gender_counts['Gender']/player_count * 100
#renames columns
gender_counts.rename(columns = {'index': 'Gender', 'Gender': '# of Players'}, inplace
#sets index as Gender for aesthetics
gender_counts.set_index(['Gender'], inplace = True)
#just checking percents sum to 100%
#gender_counts['% of Players'].sum()
#formats table
gender_counts.style.format({"% of Players": "{:.1f}%"})
```

Out[17]: <pandas.io.formats.style.Styler at 0x10df9f79c88>

4.4 Purchasing Analysis by Gender

In [18]:
```
# Purchasing Analysis (Gender)

# The below each broken by gender
# Purchase Count
# Average Purchase Price
# Total Purchase Value
```

```
# Normalized Totals

# counts purchases by gender
pur_count_by_gen = pd.DataFrame(pur_data.groupby('Gender')['Gender'].count())
# sums price by gender
total_pur_by_gen = pd.DataFrame(pur_data.groupby('Gender')['Price'].sum())
#merges the two data frames from above
pur_analysis_gen = pd.merge(pur_count_by_gen, total_pur_by_gen, left_index = True, ri
#renames columns
pur_analysis_gen.rename(columns = {'Gender': '# of Purchases', 'Price':'Total Purchas
#adds column for average purchase price by gender by dividing total purcahse value by
pur_analysis_gen['Average Purchase Price'] = pur_analysis_gen['Total Purchase Value']/
#merges gender counts from above table (excluding dup SNs) into current df
pur_analysis_gen = pur_analysis_gen.merge(gender_counts, left_index = True, right_inde
# calculates and adds normalized total column by dividing total purchase value by uni
pur_analysis_gen['Normalized Totals'] = pur_analysis_gen['Total Purchase Value']/pur_a
pur_analysis_gen
#deletes columns not needed for table (# of Players was used for normalized totals wh
del pur_analysis_gen['% of Players']
del pur_analysis_gen['# of Players']
# #resets index for aesthetics
# # pur_analysis_gen.set_index('Gender', inplace=True)
# #formats table
pur_analysis_gen.style.format({'Total Purchase Value': '${:.2f}', 'Average Purchase Pr
```

Out[18]: <pandas.io.formats.style.Styler at 0x10df9f077b8>

4.5 Age Demographics

In [19]:
```
# The below each broken into bins of 4 years (i.e. <10, 10-14, 15-19, etc.)
# Purchase Count
# Average Purchase Price
# Total Purchase Value
# Normalized Totals

#creates a column 'age_bin' based on conditional of age range
pur_data.loc[(pur_data['Age'] < 10), 'age_bin'] = "< 10"
pur_data.loc[(pur_data['Age'] >= 10) & (pur_data['Age'] <= 14), 'age_bin'] = "10 - 14
pur_data.loc[(pur_data['Age'] >= 15) & (pur_data['Age'] <= 19), 'age_bin'] = "15 - 19
pur_data.loc[(pur_data['Age'] >= 20) & (pur_data['Age'] <= 24), 'age_bin'] = "20 - 24
pur_data.loc[(pur_data['Age'] >= 25) & (pur_data['Age'] <= 29), 'age_bin'] = "25 - 29
pur_data.loc[(pur_data['Age'] >= 30) & (pur_data['Age'] <= 34), 'age_bin'] = "30 - 34
pur_data.loc[(pur_data['Age'] >= 35) & (pur_data['Age'] <= 39), 'age_bin'] = "35 - 39
pur_data.loc[(pur_data['Age'] >= 40), 'age_bin'] = "> 40"
#double checked count
# pur_data[['age_bin', 'Age']].count()

# counts purchases by age bin by counting screen names (non-unique)
```

```
pur_count_age = pd.DataFrame(pur_data.groupby('age_bin')['SN'].count())
#finds avg price of purchases by age bin
avg_price_age = pd.DataFrame(pur_data.groupby('age_bin')['Price'].mean())
#finds total purchase value by age bin
tot_pur_age = pd.DataFrame(pur_data.groupby('age_bin')['Price'].sum())
#deletes multiple occurances of SN while only keeping last, then counts # of unique
#players by age bin
no_dup_age = pd.DataFrame(pur_data.drop_duplicates('SN', keep = 'last').groupby('age_
#merges all info from above into one df
merge_age = pd.merge(pur_count_age, avg_price_age, left_index = True, right_index = Tr
#renames columns
merge_age.rename(columns = {"SN_x": "# of Purchases", "Price_x": "Average Purchase Pri
#calculates normalized totals
merge_age['Normalized Totals'] = merge_age['Total Purchase Value']/merge_age['# of Pur
#rest index for aesthetics
merge_age.index.rename("Age", inplace = True)
# formats
merge_age.style.format({'Average Purchase Price': '${:.2f}', 'Total Purchase Value':
```

Out[19]: <pandas.io.formats.style.Styler at 0x10df9f88048>

5 Top Spenders

In [20]:

```
# Identify the the top 5 spenders in the game by total purchase value, then list (in
# SN
# Purchase Count
# Average Purchase Price
# Total Purchase Value

#Group by screen name to find, total purchase per person, number of purchases per per
purchase_amt_by_SN = pd.DataFrame(pur_data.groupby('SN')['Price'].sum())
num_purchase_by_SN = pd.DataFrame(pur_data.groupby('SN')['Price'].count())
avg_purchase_by_SN = pd.DataFrame(pur_data.groupby('SN')['Price'].mean())
# merge the above dfs
merged_top5 = pd.merge(purchase_amt_by_SN, num_purchase_by_SN, left_index = True, rig
# rename columns
merged_top5.rename(columns = {'Price_x': 'Total Purchase Value', 'Price_y':'Purchase
# sort from highest purchase value to lowest
merged_top5.sort_values('Total Purchase Value', ascending = False, inplace=True)
# take top 5 only
merged_top5 = merged_top5.head()
# format
merged_top5.style.format({'Total Purchase Value': '${:.2f}', 'Average Purchase Price'
```

Out[20]: <pandas.io.formats.style.Styler at 0x10df9f078d0>

5.1 Most Popular Items

In [21]:
```
# Identify the 5 most popular items by purchase count, then list (in a table):
# Item ID
# Item Name
# Purchase Count
# Item Price
# Total Purchase Value

# gets a count of each item by grouping by Item ID and counting the number of each ID
top5_items_ID = pd.DataFrame(pur_data.groupby('Item ID')['Item ID'].count())
#sort from high to low total purchase count
top5_items_ID.sort_values('Item ID', ascending = False, inplace = True)
#keep the first 6 rows because there is a tie
top5_items_ID = top5_items_ID.iloc[0:6][:]
#find the total purchase value of each item
top5_items_total = pd.DataFrame(pur_data.groupby('Item ID')['Price'].sum())
#merge purcahse count and total purcahse value
top5_items = pd.merge(top5_items_ID, top5_items_total, left_index = True, right_index
#drop duplicate items from original Df
no_dup_items = pur_data.drop_duplicates(['Item ID'], keep = 'last')
# merge to get all other info from the top 6 using the no dup df
top5_merge_ID = pd.merge(top5_items, no_dup_items, left_index = True, right_on = 'Item
#keep only neede columns
top5_merge_ID = top5_merge_ID[['Item ID', 'Item Name', 'Item ID_x', 'Price_y', 'Price_
#reset index as item ID for aesthetics
top5_merge_ID.set_index(['Item ID'], inplace = True)
# rename columns
top5_merge_ID.rename(columns =  {'Item ID_x': 'Purchase Count', 'Price_y': 'Item Price
#format
top5_merge_ID.style.format({'Item Price': '${:.2f}', 'Total Purchase Value': '${:.2f}
```

Out[21]: <pandas.io.formats.style.Styler at 0x10df9f76d68>

5.2 Most Profitable Items

In [22]:
```
# Most Profitable Items

# Identify the 5 most profitable items by total purchase value, then list (in a table
# Item ID
# Item Name
# Purchase Count
# Item Price
# Total Purchase Value

# find total purcahse value and sort by high to low
top5_profit = pd.DataFrame(pur_data.groupby('Item ID')['Price'].sum())
top5_profit.sort_values('Price', ascending = False, inplace = True)
# only keep top 5
```

```
top5_profit = top5_profit.iloc[0:5][:]
#get item purchase count
pur_count_profit = pd.DataFrame(pur_data.groupby('Item ID')['Item ID'].count())

top5_profit = pd.merge(top5_profit, pur_count_profit, left_index = True, right_index =
top5_merge_profit = pd.merge(top5_profit, no_dup_items, left_index = True, right_on =
top5_merge_profit = top5_merge_profit[['Item ID', 'Item Name', 'Item ID_x', 'Price_y'
top5_merge_profit.set_index(['Item ID'], inplace=True)
top5_merge_profit.rename(columns = {'Item ID_x': 'Purchase Count', 'Price_y': 'Item P
top5_merge_profit.style.format({'Item Price': '${:.2f}', 'Total Purchase Value': '${:
```

Out[22]: <pandas.io.formats.style.Styler at 0x10df963c1d0>

5.3 Analysis (A) - Highest Priced Items

In [23]:
```
highest_priced = no_dup_items.sort_values('Price', ascending = False)
highest_priced[['Item ID', 'Item Name', 'Price']].head(18)
```

Out[23]:

	Item ID	Item Name	Price
657	32	Orenmir	4.95
670	177	Winterthorn, Defender of Shifting Worlds	4.89
716	103	Singed Scalpel	4.87
336	173	Stormfury Longsword	4.83
419	42	The Decapitator	4.82
436	131	Fury	4.82
398	96	Blood-Forged Skeletal Spine	4.77
455	137	Aetherius, Boon of the Blessed	4.75
686	46	Hopeless Ebon Dualblade	4.75
743	134	Undead Crusader	4.67
549	135	Warped Diamond Crusader	4.66
737	101	Final Critic	4.62
613	153	Mercenary Sabre	4.57
567	181	Reaper's Toll	4.56
421	150	Deathraze	4.54
300	99	Expiration, Warscythe Of Lost Worlds	4.53
411	7	Thorn, Satchel of Dark Souls	4.51
741	145	Fiery Glass Crusader	4.45

5.4 Analysis (B) - Lowest Priced

In [24]:
```
lowest_priced = no_dup_items.sort_values('Price', ascending = True)
lowest_priced[['Item ID', 'Item Name', 'Price']].head(18)
```

Out[24]:

	Item ID	Item Name	Price
667	15	Soul Infused Crystal	1.03
771	25	Hero Cane	1.03
624	95	Singed Onyx Warscythe	1.03
723	69	Frenzy, Defender of the Harvest	1.06
430	74	Yearning Crusher	1.06

```
720       82                       Nirvana   1.11
774      123              Twilight's Carver   1.14
647      156    Soul-Forged Steel Shortsword   1.16
467       41                         Orbit   1.16
756        6                   Rusty Skull   1.20
767      122               Unending Tyranny   1.21
761      175       Woeful Adamantite Claymore   1.24
656       63                 Stormfury Mace   1.27
750       86              Stormfury Lantern   1.28
712        5                    Putrid Fan   1.32
689       33                    Curved Axe   1.35
776      104               Gladiator's Glaive   1.36
648       92                  Final Critic   1.36
```

6 Analysis (C) - Gender Purchase Total %s

In [25]:
```
pur_analysis_gen.style.format({'Total Purchase Value': '${:.2f}', 'Average Purchase P
```

Out[25]:
```
<pandas.io.formats.style.Styler at 0x10df9f76630>
```

In [26]:
```
percent_total_gen = pur_analysis_gen['Total Purchase Value']/total_rev
percent_total_gen
```

Out[26]:
```
Gender
Female                    0.167478
Male                      0.816890
Other / Non-Disclosed     0.015632
Name: Total Purchase Value, dtype: float64
```