In [52]:

```
import pandas as pd
import numpy as np
```

In [53]:

```
#name of csv file
file_to_load = 'purchase_data.csv'
```

In [54]:

```
#read csv file
purchase_data_df = pd.read_csv(file_to_load)
purchase_data_df.head()
```

Out[54]:

	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

In [55]:

```
```

Out[55]:

```
Purchase ID    780
SN             780
Age            780
Gender         780
Item ID        780
Item Name      780
Price          780
dtype: int64
```

```
#remove rows with missing data
clean_purchase_data_df = purchase_data_df.dropna(how="any")
clean_purchase_data_df.count()
```

In [49]:

```
#display the total number of players
players_count = pd.DataFrame.count('SN')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call
last)
<ipython-input-49-830e612c58e8> in <module>()
      1 #display the total number of players
----> 2 players_count = pd.DataFrame.count('SN')

/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in count(s
elf, axis, level, numeric_only)
   6776         Myla      1
   6777         """
-> 6778         axis = self._get_axis_number(axis)
   6779         if level is not None:
   6780             return self._count_level(level, axis=axis,

AttributeError: 'str' object has no attribute '_get_axis_number'
```

In [206]:

```
#clean the data
player_info = purchase_data_df.loc[:,["Gender","SN"]]
player_info = player_info.drop_duplicates()
number_of_players = player_info.count()
```

In [207]:

```
#total player count
pd.DataFrame({"Total Players": [number_of_players]})
```

Out[207]:

	Total Players
0	Gender 576 SN 576 dtype: int64

In [208]:

```
print("The total number of players is: 576")
```

```
The total number of players is: 576
```

In [209]:

```
#Purchasing Analysis
average_item_price = purchase_data["Price"].mean()
total_purchase_value = purchase_data["Price"].sum()
purchase_count = purchase_data["Price"].count()
item_count = len(purchase_data["Item ID"].unique())

# Create a DataFrame to hold results
summary_table = pd.DataFrame({"Number of Unique Items": item_count,
                              "Total Revenue": [total_purchase_value],
                              "Number of Purchases": [purchase_count],
                              "Average Price": [average_item_price]})

# Minor Data Munging
summary_table = summary_table.round(2)
summary_table ["Average Price"] = summary_table["Average Price"].map("${:,.2f}".forr
summary_table ["Number of Purchases"] = summary_table["Number of Purchases"].map("{
summary_table ["Total Revenue"] = summary_table["Total Revenue"].map("${:,.2f}".forr
summary_table = summary_table.loc[:,["Number of Unique Items", "Average Price", "Nur

# Display the summary_table
summary_table
```

Out[209]:

	Number of Unique Items	Average Price	Number of Purchases	Total Revenue
0	183	$3.05	780	$2,379.77

In [223]:

```
#Gender Demographics
# Calculate the Number and Percentage by Gender
gender_demographics_totals_df = player_info["Gender"].value_counts()
gender_demographics_percents_df = gender_demographics_totals_df / number_of_players
gender_demographics_df = pd.DataFrame({"Total Count": gender_demographics_totals_df,

gender_demographics_df
```

Out[223]:

	Total Count	Percentage of Players
Female	81.0	NaN
Gender	NaN	NaN
Male	484.0	NaN
Other / Non-Disclosed	11.0	NaN
SN	NaN	NaN

In [212]:

```
#Purchasing Analysis - total purchase value

purchase_total_by_gender = pd.DataFrame(purchase_data_df.groupby(["Gender"]).sum()['
purchase_total_by_gender
```

Out[212]:

Gender	Total Purchase Value
Female	361.94
Male	1967.64
Other / Non-Disclosed	50.19

In [174]:

```
#Average Purchase Price by Gender
avg_purchase_gender = pd.DataFrame(purchase_data_df.groupby(["Gender"]).mean()["Pric
avg_purchase_gender.round(2)
```

Out[174]:

	Average Purchase Price
Gender	
Female	3.20
Male	3.02
Other / Non-Disclosed	3.35

In [173]:

```
#Purchase Count by Gender
gender_count= pd.DataFrame(purchase_data_df.groupby(["Gender"]).count()["Price"].ren
gender_count
```

Out[173]:

	Purchase Count
Gender	
Female	113
Male	652
Other / Non-Disclosed	15

In [227]:

```
gender_demographics_totals_df = player_info["Gender"].value_counts()
gender_demographics_percents_df = gender_demographics_totals_df / number_of_players
gender_demographics_df = pd.DataFrame({"Total Count": gender_demographics_totals_df,

purchase_total_by_gender = pd.DataFrame(purchase_data_df.groupby(["Gender"]).sum()["

avg_purchase_gender = pd.DataFrame(purchase_data_df.groupby(["Gender"]).mean()["Pric
avg_purchase_gender.round(2)

gender_count= pd.DataFrame(purchase_data_df.groupby(["Gender"]).count()["Price"].ren
```

```
avg_total_purchase_person = purchase_total_by_gender / gender_demographics_df["Total

gender_data = pd.DataFrame({"Purchase Count": gender_count, "Average Purchase Price"
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-227-23d9fc1dfffc> in <module>()
     12 avg_total_purchase_person = purchase_total_by_gender /
gender_demographics_df["Total Count"]
     13
---> 14 gender_data = pd.DataFrame({"Purchase Count": gender_count, "A
verage Purchase Price": avg_purchase_gender, "Total Purchase Value":
purchase_total_by_gender, "Average Total Purchase per Person":avg_tota
l_purchase_person})

/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in __init_
_(self, data, index, columns, dtype, copy)
    346                                 dtype=dtype, copy=copy)
    347         elif isinstance(data, dict):
--> 348             mgr = self._init_dict(data, index, columns, dtype=
dtype)
    349         elif isinstance(data, ma.MaskedArray):
    350             import numpy.ma.mrecords as mrecords

/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in _init_d
ict(self, data, index, columns, dtype)
    457             arrays = [data[k] for k in keys]
    458
--> 459         return _arrays_to_mgr(arrays, data_names, index,
columns, dtype=dtype)
    460
    461     def _init_ndarray(self, values, index, columns, dtype=None
, copy=False):

/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in _arrays
_to_mgr(arrays, arr_names, index, columns, dtype)
   7354     # figure out the index, if necessary
   7355     if index is None:
-> 7356         index = extract_index(arrays)
   7357
   7358     # don't force copy because getting jammed in an ndarray an
yway

/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in extract
_index(data)
   7391
   7392         if not indexes and not raw_lengths:
-> 7393             raise ValueError('If using all scalar values, you
must pass'
   7394                              ' an index')
```

```
   7395

ValueError: If using all scalar values, you must pass an index
```

In [228]:

```
#average total purchase per person
avg_purchase_total= purchase_total_by_gender / gender_demographics_totals["Total Cou
avg_purchase_total
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in
get_value(self, series, key)
   3123             try:
-> 3124                 return libindex.get_value_box(s, key)
   3125             except IndexError:

pandas/_libs/index.pyx in pandas._libs.index.get_value_box()

pandas/_libs/index.pyx in pandas._libs.index.get_value_box()

TypeError: 'str' object cannot be interpreted as an integer

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call
last)
<ipython-input-228-4dd28951f59c> in <module>()
      1 #average total purchase per person
----> 2 avg_purchase_total= purchase_total_by_gender /
gender_demographics_totals["Total Count"] * 100
      3 avg_purchase_total

/anaconda3/lib/python3.7/site-packages/pandas/core/series.py in __geti
tem__(self, key)
    765             key = com._apply_if_callable(key, self)
    766             try:
--> 767                 result = self.index.get_value(self, key)
    768
    769                 if not is_scalar(result):

/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in
get_value(self, series, key)
   3130                     raise InvalidIndexError(key)
   3131                 else:
-> 3132                     raise e1
   3133             except Exception:  # pragma: no cover
```

```
   3134                 raise e1

/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in
get_value(self, series, key)
   3116         try:
   3117             return self._engine.get_value(s, k,
-> 3118
tz=getattr(series.dtype, 'tz', None))
   3119         except KeyError as e1:
   3120             if len(self) > 0 and self.inferred_type in ['integ
er', 'boolean']:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyOb
jectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyOb
jectHashTable.get_item()

KeyError: 'Total Count'
```

In [172]:

```

```

Out[172]:

	Purchase Count	Average Purchase Price	Purchase Total
0	Purchase Count Gender ...	Average Purchase Price ...	Total Purchase Value Ge...

In [186]:

```
gender_percentage = gender_count/576
gender_percentage.round(2)
```

Out[186]:

	Purchase Count
Gender	
Female	0.20
Male	1.13
Other / Non-Disclosed	0.03

In [182]:

```
# Establish the bins
age_bins = [0, 9.90, 14.90, 19.90, 24.90, 29.90, 34.90, 39.90, 99999]
group_names = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39", "40+"]

# Categorize players using age range
player_info["Age Range"] = pd.cut(player_info["Age"], age_bins, labels=group_names)

age_demographics_totals = player_info["Age Ranges"].value_counts()
age_demographics_percents = age_demographics_totals / number_of_players * 100
age_demographics = pd.DataFrame({"Total Count": age_demographics_totals, "Percentage
age_demographics.sort_index()
```

Out[182]:

	Total Count	Percentage of Players
10-14	22.0	NaN
15-19	107.0	NaN
20-24	258.0	NaN
25-29	77.0	NaN
30-34	52.0	NaN
35-39	31.0	NaN
40+	12.0	NaN
<10	17.0	NaN
Age	NaN	NaN
Gender	NaN	NaN
SN	NaN	NaN

In [230]:

```
#Age Demographics

#Bin the purchasing data
purchase_data_df["Age Ranges"]=pd.cut(purchase_data_df["Age"], age_bins, labels=grou
age_purchase_total_df = purchase_data_df.groupby(["Age Ranges"]).sum()["Price"].rena
purchase_data_df["Age Range"] = pd.cut(purchase_data_df["Age"], age_bins, labels=gro

# Run basic calculations
age_purchase_total = purchase_data_df.groupby(["Age Ranges"]).sum()["Price"].rename(
age_average = purchase_data_df.groupby(["Age Ranges"]).mean()["Price"].rename("Avera
age_counts = purchase_data_df.groupby(["Age Ranges"]).count()["Price"].rename("Purch
```

```
# Calculate Normalized Purchasing
normalized_total = age_purchase_total / age_demographics["Total Count"]

# Convert to DataFrame
age_data = pd.DataFrame({"Purchase Count": age_counts, "Average Purchase Price": age

# Minor Data Munging
age_data["Average Purchase Price"] = age_data["Average Purchase Price"].map("${:,.2f
age_data["Total Purchase Value"] = age_data["Total Purchase Value"].map("${:,.2f}".f
age_data ["Purchase Count"] = age_data["Purchase Count"].map("{:,}".format)
age_data["Normalized Totals"] = age_data["Normalized Totals"].map("${:,.2f}".format)
age_data = age_data.loc[:, ["Purchase Count", "Average Purchase Price", "Total Purch

# Display the Age Table
age_data.sort_index()
```

Out[230]:

	Purchase Count	Average Purchase Price	Total Purchase Value	Normalized Totals
10-14	28.0	$2.96	$82.78	$3.76
15-19	136.0	$3.04	$412.89	$3.86
20-24	365.0	$3.05	$1,114.06	$4.32
25-29	101.0	$2.90	$293.00	$3.81
30-34	73.0	$2.93	$214.00	$4.12
35-39	41.0	$3.60	$147.67	$4.76
40+	13.0	$2.94	$38.24	$3.19
<10	23.0	$3.35	$77.13	$4.54
Age	nan	$nan	$nan	$nan
Gender	nan	$nan	$nan	$nan
SN	nan	$nan	$nan	$nan

In [241]:

```
#Top Spenders
user_total = purchase_data_df.groupby(["SN"]).sum()["Price"].rename("Total Purchase
user_average = purchase_data_df.groupby(["SN"]).mean()["Price"].rename("Average Purc
user_count = purchase_data_df.groupby(["SN"]).count()["Price"].rename("Purchase Cou
```

In [250]:

```
user_total = pd.DataFrame(purchase_data_df.groupby(["SN"]).sum()["Price"].rename("To
user_total.head(5)
```

Out[250]:

SN	Total Purchase Value
Adairialis76	2.28
Adastirin33	4.48
Aeda94	4.91
Aela59	4.32
Aelaria33	1.79

In [252]:

```
user_average = pd.DataFrame(purchase_data_df.groupby(["SN"]).mean()["Price"].rename(
user_average.head(5)
```

Out[252]:

SN	Average Purchase Price
Adairialis76	2.28
Adastirin33	4.48
Aeda94	4.91
Aela59	4.32
Aelaria33	1.79

In [251]:

```
user_count = pd.DataFrame(purchase_data_df.groupby(["SN"]).count()["Price"].rename('
user_count.head(5)
```

Out[251]:

SN	Purchase Count
Adairialis76	1
Adastirin33	1
Aeda94	1
Aela59	1
Aelaria33	1

In [254]:

```
#most popular items
item_data = purchase_data_df.loc[:,["Item ID", "Item Name", "Price"]]

total_item_purchase = item_data.groupby(["Item ID", "Item Name"]).sum()["Price"].ren
average_item_purchase = item_data.groupby(["Item ID", "Item Name"]).mean()["Price"]
item_count = item_data.groupby(["Item ID", "Item Name"]).count()["Price"].rename("Pu

item_data_pd = pd.DataFrame({"Total Purchase Value": total_item_purchase, "Item Pric

item_data_pd["Item Price"] = item_data_pd["Item Price"].map("${:,.2f}".format)
item_data_pd ["Purchase Count"] = item_data_pd["Purchase Count"].map("{:,}".format)
item_data_pd["Total Purchase Value"] = item_data_pd["Total Purchase Value"].map("${
item_data_pd = item_data_pd.loc[:,["Purchase Count", "Item Price", "Total Purchase V

# Display the Item Table
item_data_pd.sort_values("Purchase Count", ascending=False).head(10)
```

Out[254]:

Item ID	Item Name	Purchase Count	Item Price	Total Purchase Value
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
92	**Final Critic**	8	$4.88	$39.04
75	**Brutality Ivory Warmace**	8	$2.42	$19.36
59	**Lightning, Etcher of the King**	8	$4.23	$33.84
19	**Pursuit, Cudgel of Necromancy**	8	$1.02	$8.16
60	**Wolf**	8	$3.54	$28.32
34	**Retribution Axe**	8	$2.22	$17.76
72	**Winter's Bite**	8	$3.77	$30.16

In [258]:

```
#Most Profitable Items
item_data_pd.sort_values("Total Purchase Value", ascending = False).head()
```

Out[258]:

Item ID	Item Name	Purchase Count	Item Price	Total Purchase Value
63	**Stormfury Mace**	2	$4.99	$9.98
29	**Chaos, Ender of the End**	5	$1.98	$9.90
173	**Stormfury Longsword**	2	$4.93	$9.86
1	**Crucifer**	3	$3.26	$9.78
38	**The Void, Vengeance of Dark Magic**	4	$2.37	$9.48

In [ ]: