

# University of Stirling

## ITNPBD2 Representing and Manipulating Data

### Assignment Autumn 2025

## A Consultancy Job for JC Penney

NOTE: This is the extended version of my code with all cells executed

### Data provided

The data provided for this assessment included data on:

- Products.csv - Unique Id, SKU, product name, product description, price and average score
- Users.csv - Username, DOB, State
- Reviews.csv - Unique Id, Username, Product Score, Review
- JCPenney\_users.json - Username, DOB, State, Reviewed
- JCPenney\_reviews.json - unique ID, sku, description, list\_price, sale\_price, category, category\_tree, average\_product\_rating, product\_url, product\_image\_urls, brand, total\_number\_reviews, Reviews, Bought With

### Data manipulation

To answer the questions posed below I did the following:

- Removed data which did not help to answer these questions such as product URL's and Image URL's
- Joined datasets such as Users.csv and JCPenney\_reviews.json to get customer usernames, DOB's and reviews into one dataset to plot reviews by age group
- Created data subsets which had specific conditions on them to answer questions, for example only including brands with more than 10 reviews.

### Approach

With the data provided I posed the questions:

- Which products and brands are the best reviewed by customers?
- What states do most/ least reviews come from?
- What Age Group do most/least reviews come from?

### Business relevance

With information on the best and worst reviewed products and brands, JCPenney could better understand which products are meeting customer expectations and meeting the quality standards of the products the company wishes to sell. Information on products and brands could inform stock decisions and which brands JCPenney chooses to stock. Using this information to stock better products, they could increase overall customer satisfaction.

The information on demographic and geography helps JCPenney to better understand its client base and form rough conclusions on where in the US most and least of their customers are and how sales are distributed between age groups. This information could inform their marketing and sales strategy, understanding where their most lucrative locations are and targeting the locations with less customers with marketing campaigns.

In [1]: *# Importing libraries needed*

```
import os
import json
import numpy as np
import pandas as pd
import seaborn as sns
```

In [2]: *# Making sure I am in the correct directory to import files*

```
os.getcwd()
```

Out[2]: 'C:\\Users\\Danie\\ITNPBD2-assignment-1'

In [3]: *# Changing to directory where files are stored*

```
os.chdir("C:\\Users\\Danie")
```

## Initial data exploration and cleaning

I started by loading the csv and json files into panda data frames and used general data description functions to begin understanding the datasets.

### Products.csv initial exploration

Here I imported the data set into a pandas dataframe and began to look at the shape and contents of the data

In [4]: `products_csv = pd.read_csv("products.csv")` *# Importing data*

```
display(products_csv.head())
```

```
print(products_csv.shape) # Checking size of data set
print(products_csv.count()) # Using the count function has showed that the SKU,
print(products_csv.isnull().sum()) # Further to the .count() funtion, I am findi
print(products_csv.dtypes) # Checking data types
```

	Uniq_id	SKU	Name	Description	Price	Av_Sc
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on capr...	41.09	2.
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on capr...	41.09	3.
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on capr...	41.09	2.
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on capr...	41.09	3.
4	d969a8542122e1331e304b09f81a83f6	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on capr...	41.09	3.



(7982, 6)

Uniq\_id 7982

SKU 7915

Name 7982

Description 7439

Price 5816

Av\_Score 7982

dtype: int64

Uniq\_id 0

SKU 67

Name 0

Description 543

Price 2166

Av\_Score 0

dtype: int64

Uniq\_id object

SKU object

Name object

Description object

Price float64

Av\_Score float64

dtype: object

## Users.csv initial exploration

Here I imported the data set into a pandas dataframe and began to look at the shape and contents of the data

```
In [5]: users = pd.read_csv("users.csv")

display(users.head())

print(users.shape) # Checking size of data set
print(users.isnull().sum()) #Checking for missing values
print(users.dtypes) #Checking data types
```

	Username	DOB	State
0	bkpn1412	31.07.1983	Oregon
1	gqjs4414	27.07.1998	Massachusetts
2	eehe1434	08.08.1950	Idaho
3	hxkj1334	03.08.1969	Florida
4	jjbd1412	26.07.2001	Georgia

```
(5000, 3)
Username    0
DOB         0
State       0
dtype: int64
Username    object
DOB         object
State       object
dtype: object
```

## Reviews.csv initial exploration

Here I imported the data set into a pandas dataframe and began to look at the shape and contents of the data

```
In [6]: # Same process as above, importing data set, Looking at size of dataframe, check

reviews_csv = pd.read_csv("reviews.csv")

display(reviews_csv.head())

print(reviews_csv.shape)
print(reviews_csv.isnull().sum())
print(reviews_csv.dtypes)
```

	Uniq_id	Username	Score	Review
0	b6c0b6bea69c722939585baeac73c13d	fsdv4141	2	You never have to worry about the fit...Alfred...
1	b6c0b6bea69c722939585baeac73c13d	krpz1113	1	Good quality fabric. Perfect fit. Washed very ...
2	b6c0b6bea69c722939585baeac73c13d	mbmg3241	2	I do not normally wear pants or capris that ha...
3	b6c0b6bea69c722939585baeac73c13d	zeqg1222	0	I love these capris! They fit true to size and...
4	b6c0b6bea69c722939585baeac73c13d	nvfn3212	3	This product is very comfortable and the fabri...

(39063, 4)

Uniq\_id 0

Username 0

Score 0

Review 0

dtype: int64

Uniq\_id object

Username object

Score int64

Review object

dtype: object

## jcpennney\_products.json initial exploration and cleaning

Imported data in pandas, explored data and performed basic cleaning

```
In [7]: # Same process as above, importing data set, looking at size of dataframe, check
products_json = pd.read_json("jcpennney_products.json", lines = True)

display(products_json.head(10))

print(products_json.shape)
print(products_json.count())
print(products_json.isnull().sum())
print(products_json.dtypes)
```

		uniq_id	sku	name_title	description	list_price	
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
4	d969a8542122e1331e304b09f81a83f6	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
5	cf73bb2bd93bbd6e1bdf48d399992270	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
6	8ffd0ef4fcdf1a82fb514aba5d20e05b	pp5006790247		Alfred Dunner® Feels Like Spring 3/4 Sleeve Le...	For easygoing style you'll love, wear our stre...	65.27	
7	4d9337e3c8f974d3c420cdc5c58b3fc3	pp5007090172		Alfred Dunner® Feels Like Spring 3/4-Sleeve Le...	Spring is in the air with our 3/4-sleeve leaf ...		

	uniq_id	sku	name_title	description	list_price	:
8	44f8f8f108c6856acf9630dd1d78516d	pp5007080134	Alfred Dunner® Feels Like Spring 3/4-Sleeve Wa...			
9	8d1d057f5f808c10ce243c222ab0ef6e	pp5007080134	Alfred Dunner® Feels Like Spring 3/4-Sleeve Wa...			

```

(7982, 15)
uniq_id          7982
sku              7982
name_title       7982
description       7982
list_price       7982
sale_price       7982
category         7982
category_tree    7982
average_product_rating 7982
product_url      7982
product_image_urls 7982
brand            7982
total_number_reviews 7982
Reviews          7982
Bought With      7982
dtype: int64
uniq_id          0
sku              0
name_title       0
description       0
list_price       0
sale_price       0
category         0
category_tree    0
average_product_rating 0
product_url      0
product_image_urls 0
brand            0
total_number_reviews 0
Reviews          0
Bought With      0
dtype: int64
uniq_id          object
sku              object
name_title       object
description       object
list_price       object
sale_price       object
category         object
category_tree    object
average_product_rating float64
product_url      object
product_image_urls object
brand            object
total_number_reviews int64
Reviews          object
Bought With      object
dtype: object

```

```

In [8]: products_json.replace("", np.nan, inplace=True) #replacing empty strings with Na
print(products_json.isnull().sum()) #Looking for missing values

print(products_json.dtypes)

```

```

uniq_id          0
sku              67
name_title       0
description      543
list_price      2166
sale_price       18
category        636
category_tree    636
average_product_rating  0
product_url      0
product_image_urls 157
brand           0
total_number_reviews 0
Reviews         0
Bought With     0
dtype: int64
uniq_id          object
sku              object
name_title       object
description      object
list_price       object
sale_price       object
category         object
category_tree    object
average_product_rating float64
product_url      object
product_image_urls object
brand           object
total_number_reviews int64
Reviews         object
Bought With     object
dtype: object

```

## jcpenny\_reviewers.json initial exploration and cleaning

In [9]: *#As above, Loading dataframe, Looking at size and for missing values*

```

reviews_json = pd.read_json("jcpenny_reviewers.json", lines=True)

display(reviews_json.head())

print(reviews_json.shape)
print(reviews_json.count())
print(reviews_json.isnull().sum())

```

	Username	DOB	State	Reviewed
0	bkpn1412	31.07.1983	Oregon	[cea76118f6a9110a893de2b7654319c0]
1	gqjs4414	27.07.1998	Massachusetts	[fa04fe6c0dd5189f54fe600838da43d3]
2	eehe1434	08.08.1950	Idaho	[]
3	hkxj1334	03.08.1969	Florida	[f129b1803f447c2b1ce43508fb822810, 3b0c9bc0be6...]
4	jjbd1412	26.07.2001	Georgia	[]

```
(5000, 4)
Username    5000
DOB         5000
State       5000
Reviewed    5000
dtype: int64
Username    0
DOB         0
State       0
Reviewed    0
dtype: int64
```

```
In [10]: reviews_json.replace("", np.nan, inplace=True) #replacing empty strings with Nan
print(reviews_json.isnull().sum()) #Checking for missing values
```

```
Username    0
DOB         0
State       0
Reviewed    0
dtype: int64
```

## Creating a subset to visualise best and worst reviewed products

Doing some further cleaning on this dataset before creating a subset for the purpose of visualising best and worst reviewed products by average product rating

```
In [11]: #Changing datatype to float
```

```
products_json["list_price"] = products_json["list_price"].astype(float)
```

```
In [12]: #When I initially ran the function to convert object to float, there was an error
#Changing values to numeric data and ranges to Nan values
```

```
products_json["sale_price"] = pd.to_numeric(products_json["sale_price"], errors="coerce")
```

```
In [13]: #Changing datatype to float
```

```
products_json["sale_price"] = products_json["sale_price"].astype(float)
```

## In the first half of my analysis, I focused on drawing insights and creating visualisations based on products

I decided to look at average product ratings and from this, identify the best and worst reviewed products and brands

## Creating subset of Products JSON to show true average value by SKU

As each entry has an individual "average rating", I created a subset and took an overall average for each product grouped by SKU to represent an individual product.

```
In [14]: products_json["sku"] = products_json["sku"].astype(str)
#I was getting an error when running the code in cell 81, converting to a str fi
```

```
In [15]: products_json.head()
```

```
Out[15]:
```

		uniq_id	sku	name_title	description	list_price
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09
4	d969a8542122e1331e304b09f81a83f6	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09

```
In [16]: #Creating a more manageable subset to group the SKU's and get an overall mean fo
subset = products_json[["name_title", "sku", "average_product_rating", "total_numbe
#Removing empty values in sku column
subset = subset[subset["sku"] != "nan"]
```

```
In [17]: #display(subset.head())
print(subset.dtypes)
```

```

name_title      object
sku             object
average_product_rating float64
total_number_reviews int64
dtype: object

```

```

In [18]: # There are muptiple rows for each sku so I am grouping them together and calucl
# "average_product_rating" to get an average rating for each product

```

```

subset = subset.groupby(["name_title", "sku"]).agg(
    total_av = ("average_product_rating", "mean"),
    total_reviews = ("total_number_reviews", "sum"))

```

```

In [19]: #Sorting to see which products have the most/least reviews

```

```

subset_sorted = subset.sort_values(by = "total_reviews", ascending = False) #sor
display(subset_sorted)

```

		total_av	total_reviews
	name_title	sku	
	<b>Stafford® Gunner Mens Cap Toe Leather Boots</b>	<b>pp5004560042</b>	2.777778 144
	<b>Clarks® Leisa Grove Leather Sandals</b>	<b>pp5006210554</b>	3.125000 144
	<b>Xersion™ Quick-Dri Performance Bootcut Pant</b>	<b>pp5005870510</b>	3.057692 104
	<b>Clarks® Leisa Grove Leather Sandals - Wide Width</b>	<b>pp5006210555</b>	3.050595 81
	<b>St. John's Bay® Secretly Slender Straight-Leg Jeans</b>	<b>pp5005840398</b>	3.075000 80
	...	...	...
	<b>nicole by Nicole Miller® Tess Shoulder Bag</b>	<b>pp5006750733</b>	4.000000 1
	<b>philosophy Purity Made Simple</b>	<b>pp5005650382</b>	2.000000 1
	<b>softspots® Caren Leather Loafers - Wide Width</b>	<b>pp5005790633</b>	1.000000 1
	<b>the skinny® Crystal Infinity Ring Set</b>	<b>pp5004040114</b>	3.000000 1
	<b>St. John's Bay® Printed Flat-Front Shorts</b>	<b>pp5006291605</b>	2.000000 1

6159 rows × 2 columns

```

In [20]: print(sum(subset_sorted["total_reviews"] > 10))
# I recognised above that if I were to plot the subset that the data would be sk
# Therefore, I decided to only plot products which have more than 10 reviews
# Here I am checking how many products meet that condition

```

763

```

In [21]: # Creating a new sub set with products which have more than 10 reviews

```

```

x = subset_sorted["total_reviews"] > 10

top_reviewed = subset_sorted[x].reset_index()

```

```
# Sorting by highest average review

top_reviewed_ordered = top_reviewed.sort_values(by = "total_av", ascending = False)

# Creating smaller subsets of the top and bottom 10 products by average review score

top_10 = top_reviewed_ordered.head(10) #Creating subsets top and bottom products

bottom_10 = top_reviewed_ordered.tail(10)

bottom_10.loc[436, "name_title"] = "Zeroxposur® Static Wide-Strap Tankini Swim W
```

C:\Users\Danie\AppData\Local\Temp\ipykernel\_26752\412532790.py:17: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
bottom_10.loc[436, "name_title"] = "Zeroxposur® Static Wide-Strap Tankini Swim Wear" #Renaming long product name which was difficult to read on graph
```

In [22]: `display(bottom_10)`

	name_title	sku	total_av	total_reviews
195	Okie Dokie® Print Leggings - Toddler Girls 2t-5t	pp5005760351	2.184524	23.0
762	Keurig® K250 2.0 Compact Brewer	pp5005140282	2.181818	11.0
629	Total Girl® Printed Capri Leggings - Girls 7-1...	pp5006081442	2.175000	13.0
655	Arizona Long-Sleeve Polo Shirt	pp5002760285	2.166667	12.0
593	Foreo Luna™ For Combination Skin	pp5005960565	2.142857	14.0
534	Zeroxposur® Static Wide-Strap Tankini Swim Top...	pp5006640562	2.125000	16.0
244	Wicker Chair Cushion	pp5005430549	2.125000	19.0
718	Marc Jacobs Beauty Enamored Hi-Shine Nail Polish	pp5006620365	2.090909	11.0
451	St. John's Bay® Short-Sleeve V-Neck T-Shirt	pp5006310918	2.062500	16.0
321	St. John's Bay® Zaber Thong Sandals	pp5006320143	2.062500	16.0
436	Zeroxposur® Static Wide-Strap Tankini Swim Wear	NaN	NaN	NaN

In [23]: `import matplotlib.pyplot as plt`  
`%matplotlib inline`

## Creating a visualisation to look at the top and bottom 10 products by average user rating

# Figure 1.1 in report

```
In [24]: plt.figure(figsize=(12,6)) #Setting figure size

plt.subplot(121) #Defining subplot size and position
plt.title("Best 10 Products")
plt.ylabel("Average Customer Rating (Out of 5)")
plt.bar(top_10["name_title"],top_10["total_av"],
        color="blue",
        edgecolor="black",
        linewidth=0.7) #color, edgecolor, linewidth to improve readability

plt.xticks(ticks=range(len(top_10)),
           labels=top_10["name_title"],
           rotation=45,
           ha="right",fontsize=12) #rotated x labels to improve readability

plt.ylim(0,4.2) #Ensuring scale of Y axis is same for both graphs

plt.subplot(122)
plt.title("Worst 10 Products")
plt.bar(bottom_10["name_title"],bottom_10["total_av"],
        color="orangered",
        edgecolor="black",
        linewidth=0.7)

labels = len(bottom_10) #Defining length of dataframe for ticks function

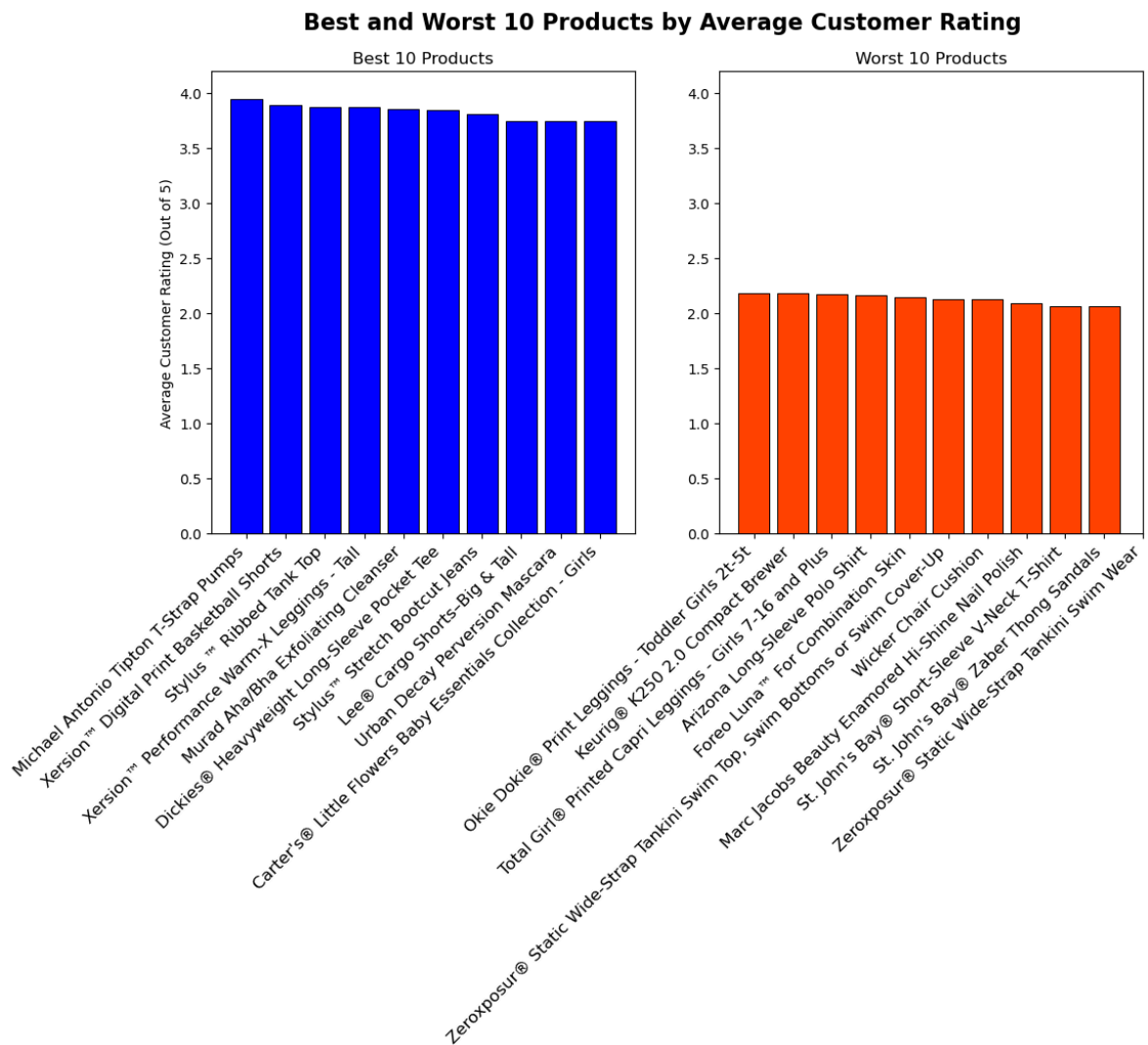
plt.xticks(ticks=range(labels),
           labels=bottom_10["name_title"],
           rotation=45,
           ha="right",fontsize=12) #Defining labels on x axis and rotating 45 deg

plt.ylim(0,4.2)

plt.suptitle("Best and Worst 10 Products by Average Customer Rating", fontsize=12)

plt.savefig("best_products_1.png", dpi=300, bbox_inches='tight') #Saving figure

plt.show()
```



This graph would have been much better displayed as a Plotly graph where the names were shown when hovering over a bar, however, I decided not to use this as the report is a PDF and will not allow for interactive graphs. I also considered a horizontal bar chart but thought the names were too long to fit.

## Creating a visualisation to look at the overall average product score (for products with more than 10 reviews)

### Figure 1 in report

```
In [25]: print(top_reviewed.shape) #To see how many products there are in total
```

```
(763, 4)
```

```
In [26]: product_average = top_reviewed["total_av"].mean() #Calculating a mean average score
```

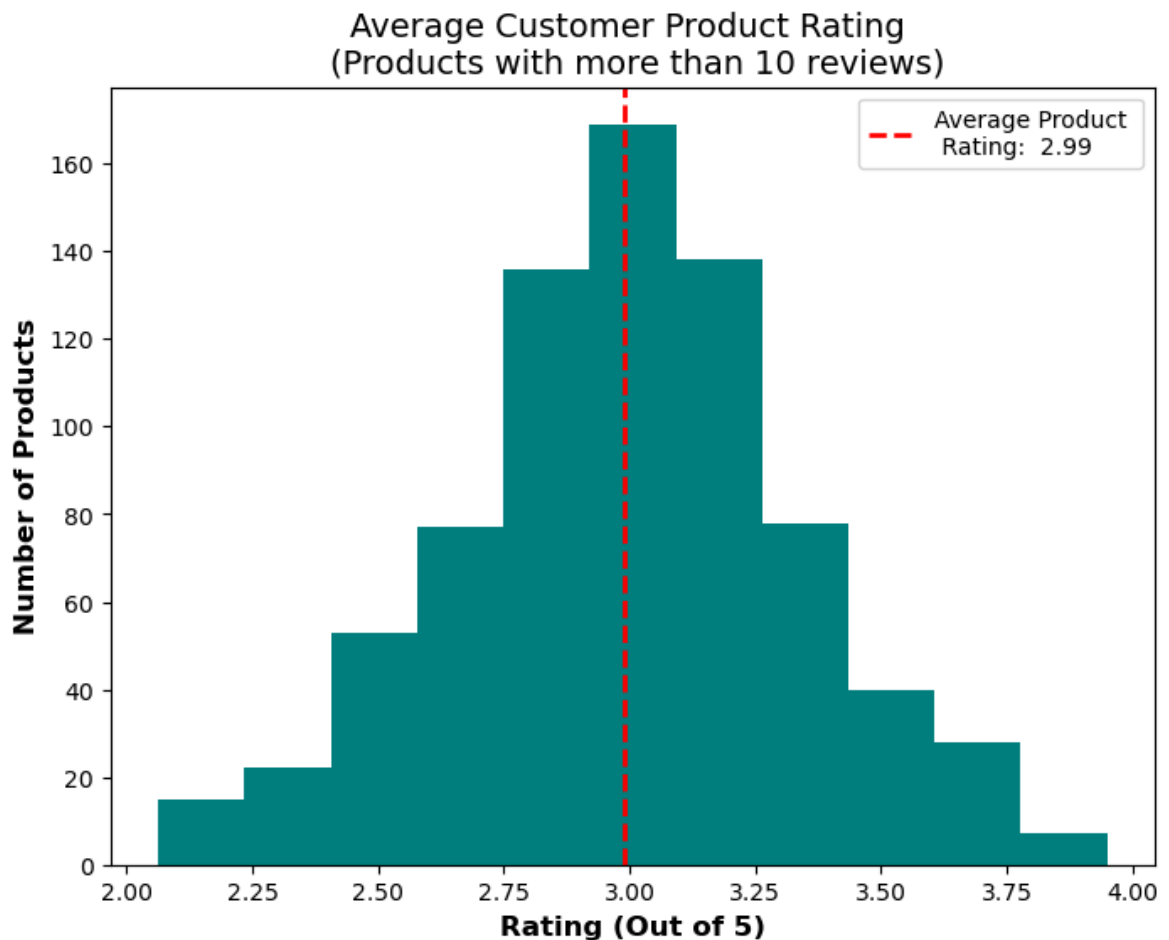
```
In [27]: plt.figure(figsize=(8,6)) #Defining figure size

plt.hist(top_reviewed["total_av"], bins=11,
         color="teal") #defining number of bins for histogram and the data to use
```

```
plt.axvline(x=product_average,
            color="red",
            linestyle="--",
            linewidth=2,
            label=f"Average Product \n Rating: {product_average: .2f}") #improvi

plt.title("Average Customer Product Rating \n (Products with more than 10 review
plt.legend()
plt.xlabel("Rating (Out of 5)", fontsize=12,fontweight="bold")
plt.ylabel("Number of Products",fontsize=12,fontweight="bold")

plt.savefig("av_rating.png", dpi=300, bbox_inches='tight') #Saving figure to exp
plt.show()
```



## Exploring which brands are best and worst reviewed

Here I load the data into pandas, clean by removing unnecessary rows and creating a smaller data subset with average rating and total number of ratings

```
In [28]: df = pd.read_json("jcpenney_products.json", lines=True) #importing data
display(df.head())
```

		uniq_id	sku	name_title	description	list_price	s
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	
4	d969a8542122e1331e304b09f81a83f6	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	

In [29]: `df.drop(columns = ["product_image_urls", "product_url", "Reviews", "Bought With", "display(df.head())`

*# Making number of columns more manageable by using .drop function to remove col*

		uniq_id	sku	name_title	list_price	sale_price	av
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	41.09	24.16	
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	41.09	24.16	
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	41.09	24.16	
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	41.09	24.16	
4	d969a8542122e1331e304b09f81a83f6	pp5006380337		Alfred Dunner® Essential Pull On Capri Pant	41.09	24.16	

```
In [30]: # Reducing size further to make a subset for the brand exploration

brands = df[["brand", "average_product_rating", "total_number_reviews"]]
display(brands.head())
```

	brand	average_product_rating	total_number_reviews
0	Alfred Dunner	2.625	8
1	Alfred Dunner	3.000	8
2	Alfred Dunner	2.625	8
3	Alfred Dunner	3.500	8
4	Alfred Dunner	3.125	8

```
In [31]: unique = brands["brand"].unique() #Finding out how many unique brand names are
display(brands)
print(len(unique))
```

	brand	average_product_rating	total_number_reviews
0	Alfred Dunner	2.625	8
1	Alfred Dunner	3.000	8
2	Alfred Dunner	2.625	8
3	Alfred Dunner	3.500	8
4	Alfred Dunner	3.125	8
...	...	...	...
7977	Hoover	3.875	8
7978	Hoover	3.250	8
7979	Hoover	2.500	8
7980	Asstd National Brand	3.500	8
7981	Asstd National Brand	3.000	8

7982 rows × 3 columns

721

```
In [32]: # I want to find out what brands have the lowest ratings to see if there are tre
# brands which have less than 10 reviews so we have a reasonable sample size of
import pandas as pd

brands_sub = brands.groupby("brand").agg(
    average_rating = ("average_product_rating", "mean"),
    total_reviews = ("total_number_reviews", "sum")) #Each brand has multiple row

display(brands_sub.head())
print(len(brands_sub)) #Checking how many rows there are (how many brands)
```

	average_rating	total_reviews
brand		
1928 Jewelry	4.750000	3
A.N.A	3.032509	1106
A2 BY AEROSOLE	3.288046	118
ACE BAYOU	2.500000	11
ADIDAS	3.075809	351

721

```
In [33]: brands_sorted = brands_sub.sort_values(by = "average_rating", ascending = False)
display(brands_sorted.head())
```

	average_rating	total_reviews
brand		
BIALETTI	5.0	1
SKYWAY	5.0	2
Q-T INTIMATES	5.0	1
Bee Darlin Bee Smart	5.0	1
FIFTH AND PARK	5.0	1

```
In [34]: x = brands_sorted["total_reviews"] > 10 #Filtering to only include brands with m
```

```
In [35]: brands_sorted = brands_sorted[x] #Updating data frame to reflect change above
display(brands_sorted.head())
```

	average_rating	total_reviews
brand		
CREATIVE BATH	4.20	12
Nicole By Nicole Miller	4.18	21
SKYLINE FURNITURE	4.12	14
UMBRA	3.98	12
Nordicware	3.92	20

```
In [36]: best_brands = brands_sorted.head(10).reset_index() #Resetting data table index t
worst_brands = brands_sorted.tail(10).reset_index().sort_values(by = "total_revi
```

```
In [37]: display(worst_brands)
display(best_brands)
```

	brand	average_rating	total_reviews
7	Seventeen	2.07	50
9	Warner's	2.05	33
2	Certified International	2.33	22
0	FOREO	2.38	22
1	John Deere	2.38	14
3	FROZEN	2.29	11
5	Ariya	2.19	11
4	PHILIPS	2.22	11
6	MARC JACOBS BEAUTY	2.09	11
8	SLEEP PHILOSOPHY	2.06	11

	brand	average_rating	total_reviews
0	CREATIVE BATH	4.20	12
1	Nicole By Nicole Miller	4.18	21
2	SKYLINE FURNITURE	4.12	14
3	UMBRA	3.98	12
4	Nordicware	3.92	20
5	MURAD	3.86	14
6	LILIANA	3.73	11
7	Buxton	3.70	14
8	National Presto	3.69	11
9	EVA LONGORIA HOME	3.68	16

## Plotting the 10 best and worst reviewed brands

### Figure 1.2 in report

```
In [38]: plt.figure(figsize=(7,11)) #Setting figure size

plt.subplot(211) #Defining subplot size and position
plt.title("Best 10 Brands", fontweight="bold")
plt.xlabel("Average Customer Rating (Out of 5)")
plt.barh(best_brands["brand"], best_brands["average_rating"],
         color="limegreen",
         edgecolor="black",
         linewidth=0.7) #color, edgecolor, linewidth to improve readability

plt.xlim(0,5) #Ensuring x axis limits are the same for both graphs

plt.subplot(212)
plt.title("Worst 10 Brands", fontweight="bold")
plt.barh(worst_brands["brand"], worst_brands["average_rating"],
         color="orangered",
         edgecolor="black",
         linewidth=0.7)

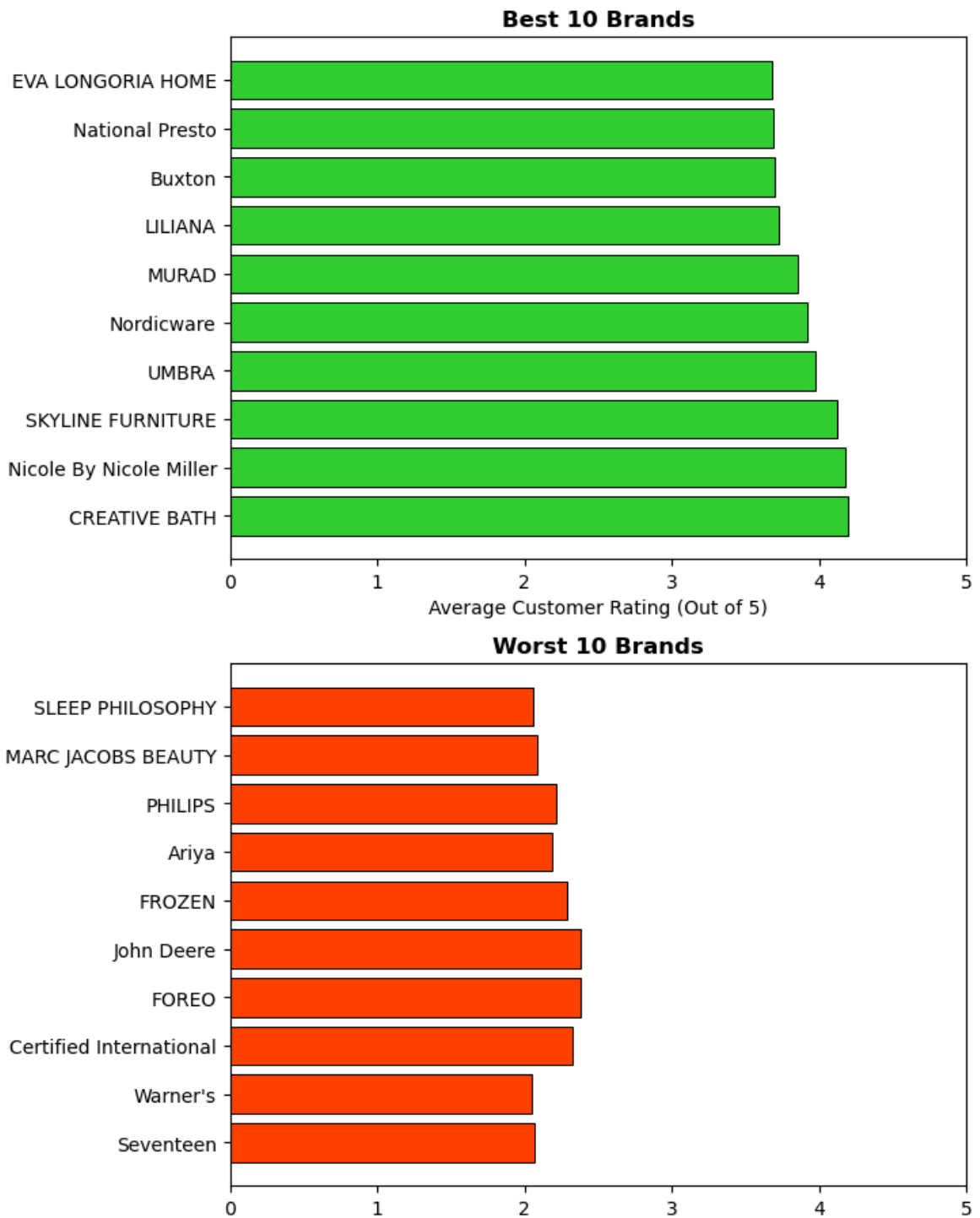
plt.xlim(0,5)

plt.suptitle("Best and Worst 10 Brands by Average Customer Rating", fontsize=14)

plt.savefig("brands.png", dpi=300, bbox_inches='tight') #Saving figure to export

plt.show()
```

## Best and Worst 10 Brands by Average Customer Rating



## Exploring customer data

I am now moving on to analysing and drawing insights from customer data, looking at demographic information - User numbers by US state and reviews by age group

```
In [39]: users_csv = pd.read_csv("users.csv") #importing users data to begin exploration
display(users_csv.head())
print(len(users_csv))
```

	Username	DOB	State
0	bkpn1412	31.07.1983	Oregon
1	gqjs4414	27.07.1998	Massachusetts
2	eehe1434	08.08.1950	Idaho
3	hkxj1334	03.08.1969	Florida
4	jjbd1412	26.07.2001	Georgia

5000

```
In [40]: users_json = pd.read_json("jcpenny_reviewers.json", lines=True) #importing user
display(users_json)
print(len(users_json))
```

	Username	DOB	State	Reviewed
0	bkpn1412	31.07.1983	Oregon	[cea76118f6a9110a893de2b7654319c0]
1	gqjs4414	27.07.1998	Massachusetts	[fa04fe6c0dd5189f54fe600838da43d3]
2	eehe1434	08.08.1950	Idaho	[]
3	hkxj1334	03.08.1969	Florida	[f129b1803f447c2b1ce43508fb822810, 3b0c9bc0be6...]
4	jjbd1412	26.07.2001	Georgia	[]
...	...	...	...	...
4995	mfnn1212	27.07.1997	Delaware	[d6cd506246bd17afa611b6a06236713c]
4996	ejnb3414	01.08.1976	Minnesota	[97de1506cd0bcbe50f2797cd0588eb81]
4997	pdzw1433	28.07.1994	Ohio	[799d62906019d910fa744987da184ae7, b8f5deb7b02...]
4998	npha1342	07.08.1953	Montana	[6250b1d691cd3842f05b87736f2fadbf]
4999	yuoc2324	02.08.1975	South Dakota	[]

5000 rows × 4 columns

5000

```
In [41]: #Finding out how which states are represented in the data

states = users_csv["State"].unique()
display(states)
```

```
array(['Oregon', 'Massachusetts', 'Idaho', 'Florida', 'Georgia',  
      'Montana', 'Pennsylvania', 'Connecticut', 'Arkansas', 'Nebraska',  
      'California', 'New Hampshire', 'District of Columbia',  
      'Washington', 'Minnesota', 'New Mexico', 'Virginia', 'Kansas',  
      'Illinois', 'North Dakota', 'Colorado', 'New York',  
      'Minor Outlying Islands', 'Northern Mariana Islands',  
      'West Virginia', 'Texas', 'South Dakota', 'Maryland', 'Maine',  
      'Ohio', 'Rhode Island', 'Michigan', 'Alaska', 'Iowa', 'Oklahoma',  
      'Mississippi', 'South Carolina', 'Missouri', 'New Jersey',  
      'Tennessee', 'North Carolina', 'Guam', 'Wyoming', 'Delaware',  
      'Vermont', 'Indiana', 'Louisiana', 'Wisconsin', 'Hawaii',  
      'Puerto Rico', 'Alabama', 'Kentucky', 'Arizona', 'Nevada', 'Utah',  
      'American Samoa', 'U.S. Virgin Islands'], dtype=object)
```

In [42]: *#Finding out how many users there are per state*

```
state_count = users_csv["State"].value_counts()  
  
print(state_count)
```

State	
Massachusetts	107
Delaware	106
Vermont	103
Northern Mariana Islands	102
New Jersey	101
Oklahoma	100
California	99
Kentucky	99
Montana	97
Virginia	96
New Mexico	96
Oregon	96
Alabama	95
U.S. Virgin Islands	95
Maine	94
Mississippi	94
Iowa	94
Washington	94
Alaska	94
Rhode Island	93
Arkansas	92
Minor Outlying Islands	92
Nebraska	90
Kansas	90
Nevada	90
Florida	89
Tennessee	89
Hawaii	88
Pennsylvania	86
American Samoa	86
Indiana	86
Wyoming	86
Colorado	85
North Dakota	85
Wisconsin	84
Missouri	84
New Hampshire	83
Texas	83
Puerto Rico	83
District of Columbia	83
New York	83
Connecticut	82
Ohio	81
Louisiana	80
Utah	80
West Virginia	80
South Dakota	79
Georgia	79
Idaho	79
Maryland	77
Minnesota	77
South Carolina	77
Michigan	76
Guam	73
Arizona	71
Illinois	69
North Carolina	68
Name: count, dtype: int64	

## Plotting data to show states which have highest and lowest number of users/customers

I did not use this visualisation in the report as there was not enough space in the 5 pages.

I opted to use a dot plot over a bar chart as it appears much clearer and better shows clusters in the data. I decided to change the x axis limit to better show the variation between states.

```
In [43]: states = state_count.index #extracting states for y axis indexes
counts = state_count.values #extracting numerical data for number of users on x
y = np.arange(len(states)) #creating y co-ordinates
print(type(counts))

<class 'numpy.ndarray'>

In [44]: plt.figure(figsize=(8,12)) #Creating figure

plt.scatter(counts,y,
            color="teal",s=50) #plotting data on scatter plot

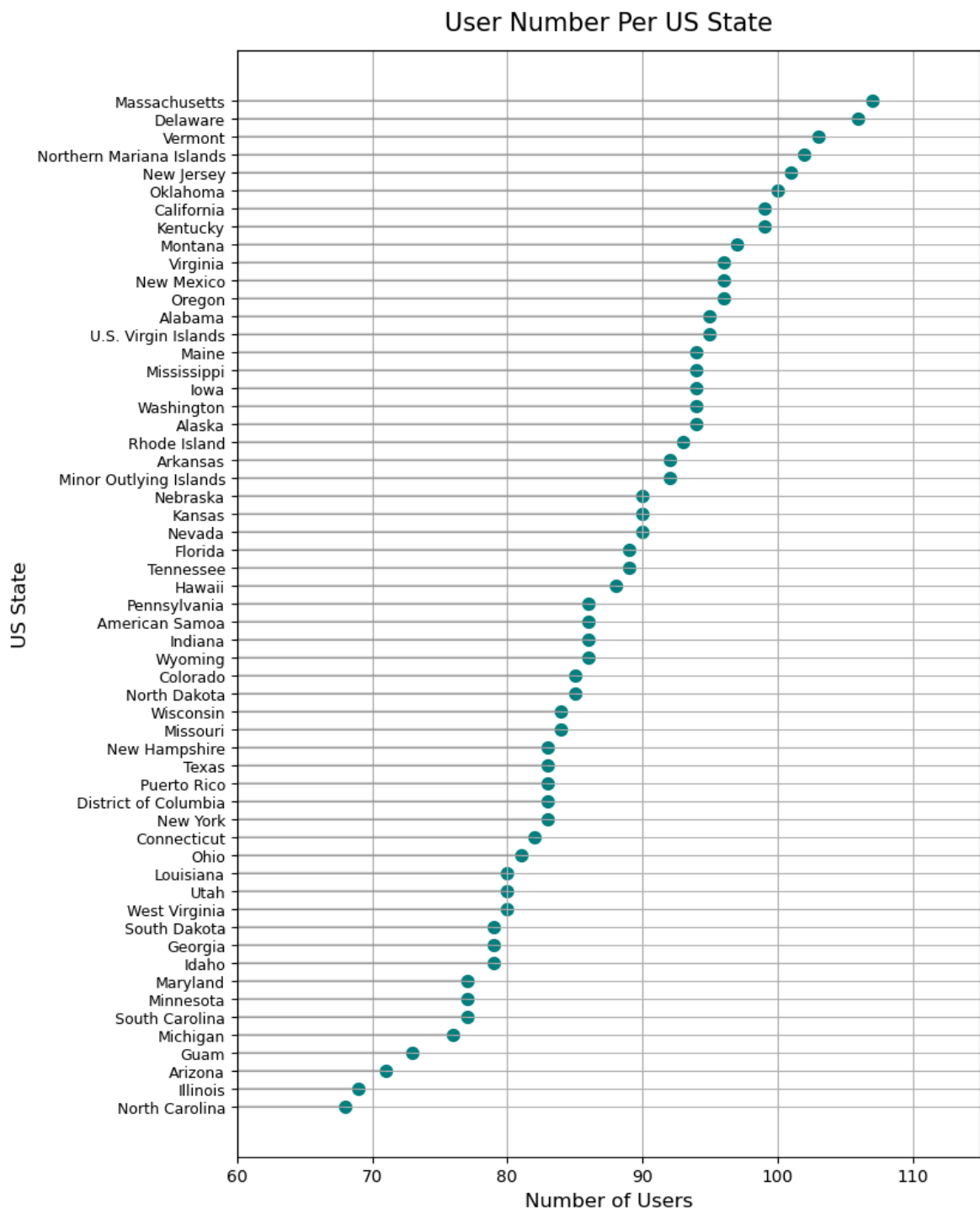
plt.hlines(
    y, xmin=0,
    xmax=counts,
    color="gray",
    alpha=0.4) #Adding horizontal lines for readability

plt.yticks(y, states, fontsize=9) #defining values on y axis
plt.gca().invert_yaxis() #Reversing y axis to show most users at top

plt.title("User Number Per US State", fontsize=15,pad=12)
plt.xlabel("Number of Users", fontsize=12)
plt.ylabel("US State", fontsize=12)
plt.xlim(60,115) #Changed as there were no data points below 60 on x axis. This

plt.tight_layout
plt.grid()

plt.savefig("states.png", dpi=300, bbox_inches='tight') #Saving figure to export
plt.show()
```



## Plotting users per state on US map with Plotly

### Figure 2 in report

If I had more space in the report, I would have displayed this map alongside a full table of the states and their user numbers.

I opted to use this map alongside a small table showing the top and bottom 10 states for clarity.

```
In [45]: import plotly.graph_objects as go #importing library
```

```
In [46]: users = pd.read_csv("users.csv") #importing user data
```

```
In [47]: # Creating a subset which shows users per state

state_counts = users["State"].value_counts().reset_index()
state_counts.columns = ["State", "user_count"]
```

```
In [48]: user_counts = state_counts["user_count"].sum() #Totalling users per state

display(state_counts.head())
print(user_counts)
```

	State	user_count
0	Massachusetts	107
1	Delaware	106
2	Vermont	103
3	Northern Mariana Islands	102
4	New Jersey	101

5000

```
In [49]: states = users_csv["State"].unique() #Looking at what states are in the dataset
display(states)
```

```
array(['Oregon', 'Massachusetts', 'Idaho', 'Florida', 'Georgia',
      'Montana', 'Pennsylvania', 'Connecticut', 'Arkansas', 'Nebraska',
      'California', 'New Hampshire', 'District of Columbia',
      'Washington', 'Minnesota', 'New Mexico', 'Virginia', 'Kansas',
      'Illinois', 'North Dakota', 'Colorado', 'New York',
      'Minor Outlying Islands', 'Northern Mariana Islands',
      'West Virginia', 'Texas', 'South Dakota', 'Maryland', 'Maine',
      'Ohio', 'Rhode Island', 'Michigan', 'Alaska', 'Iowa', 'Oklahoma',
      'Mississippi', 'South Carolina', 'Missouri', 'New Jersey',
      'Tennessee', 'North Carolina', 'Guam', 'Wyoming', 'Delaware',
      'Vermont', 'Indiana', 'Louisiana', 'Wisconsin', 'Hawaii',
      'Puerto Rico', 'Alabama', 'Kentucky', 'Arizona', 'Nevada', 'Utah',
      'American Samoa', 'U.S. Virgin Islands'], dtype=object)
```

```
In [50]: # Removing US territories so that the remaining states align with the US map gra
```

```
remove = ["Minor Outlying Islands", "Northern Mariana Islands",
          "Guam", "Puerto Rico", "American Samoa", "U.S. Virgin Islands",
          "District of Columbia"]
```

```
to_drop = state_counts["State"].isin(remove) #Removing the above states
```

```
# Creating a subset of the filtered states
```

```
states_filtered = state_counts[~to_drop]
states_filtered = states_filtered.reset_index(drop=True)
```

```
In [51]: #display(states_filtered)
print(len(states_filtered))
```

50

```
In [52]: # Converting state names to state codes - required for mapping onto go.Choropleth

state_codes = {"Alabama": "AL", "Alaska": "AK", "Arizona": "AZ", "Arkansas": "AR",
               "California": "CA", "Colorado": "CO", "Connecticut": "CT", "Delaware": "DE",
               "Florida": "FL", "Georgia": "GA", "Hawaii": "HI", "Idaho": "ID",
               "Illinois": "IL", "Indiana": "IN", "Iowa": "IA", "Kansas": "KS",
               "Kentucky": "KY", "Louisiana": "LA", "Maine": "ME", "Maryland": "MD",
               "Massachusetts": "MA", "Michigan": "MI", "Minnesota": "MN",
               "Mississippi": "MS", "Missouri": "MO", "Montana": "MT", "Nebraska": "NE",
               "Nevada": "NV", "New Hampshire": "NH", "New Jersey": "NJ", "New Mexico": "NM",
               "New York": "NY", "North Carolina": "NC", "North Dakota": "ND", "Ohio": "OH",
               "Oklahoma": "OK", "Oregon": "OR", "Pennsylvania": "PA", "Rhode Island": "RI",
               "South Carolina": "SC", "South Dakota": "SD", "Tennessee": "TN", "Texas": "TX",
               "Utah": "UT", "Vermont": "VT", "Virginia": "VA", "Washington": "WA",
               "West Virginia": "WV", "Wisconsin": "WI", "Wyoming": "WY",
               "District of Columbia": "DC"}
```

```
In [53]: # Adding state codes to data frame

states_filtered["state_codes"] = states_filtered["State"].map(state_codes)
#display(states_filtered)
```

```
In [54]: # Plotting information onto map

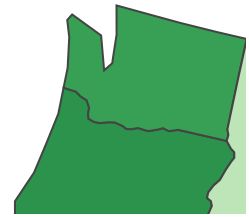
fig = go.Figure(data=go.Choropleth(
    locations=states_filtered["state_codes"],
    z = states_filtered["user_count"].astype(int),
    locationmode = "USA-states",
    colorscale="Greens",
    colorbar_title="Number of Users",)) #imported Choropleth map, defined data v

# Improving map format

fig.update_layout(
    geo=dict(showframe=False, showcoastlines=True,
             projection_type='albers usa'),
    title_text="Users by US State",
    margin=dict(l=0, r=0, t=50, b=0)) #removing frame around map, showing coastl

fig.show()
```

## Users by US State



In [55]: *#Creating dataframes to display alongside the map showing the 10 states with mos*

```
top_10 = state_counts.head(10)
bottom_10 = state_counts.tail(10)

print(top_10)
```

	State	user_count
0	Massachusetts	107
1	Delaware	106
2	Vermont	103
3	Northern Mariana Islands	102
4	New Jersey	101
5	Oklahoma	100
6	California	99
7	Kentucky	99
8	Montana	97
9	Virginia	96

## Analysing number of reviews by age group

```
In [56]: reviews = pd.read_csv("reviews.csv") #importing data
print(reviews.shape)
display(reviews.head(5))
```

(39063, 4)

	Uniq_id	Username	Score	Review
0	b6c0b6bea69c722939585baeac73c13d	fsdv4141	2	You never have to worry about the fit...Alfred...
1	b6c0b6bea69c722939585baeac73c13d	krpz1113	1	Good quality fabric. Perfect fit. Washed very ...
2	b6c0b6bea69c722939585baeac73c13d	mbmg3241	2	I do not normally wear pants or capris that ha...
3	b6c0b6bea69c722939585baeac73c13d	zeqg1222	0	I love these capris! They fit true to size and...
4	b6c0b6bea69c722939585baeac73c13d	nvfn3212	3	This product is very comfortable and the fabri...

```
In [57]: users = pd.read_csv("users.csv") #importing data
print(users.shape)
display(users.head(5))
```

(5000, 3)

	Username	DOB	State
0	bkpn1412	31.07.1983	Oregon
1	gqjs4414	27.07.1998	Massachusetts
2	eehe1434	08.08.1950	Idaho
3	hkxj1334	03.08.1969	Florida
4	jjbd1412	26.07.2001	Georgia

```
In [58]: merged = pd.merge(users, reviews, how="inner", on="Username") #Used inner join in
```

```
In [59]: print(merged.shape) #checking how many rows are in the dataset
```

(39080, 6)

```
In [60]: display(merged.head())
```

	Username	DOB	State	Uniq_id	Score	Review
0	bkpn1412	31.07.1983	Oregon	2565bf02ec05a59cd1fd78955dda108d	0	Love this heated mattress pad! Ordered the que...
1	bkpn1412	31.07.1983	Oregon	e09b42d19cd350f9762f92d59ecf88ba	2	I am 511 and a size 6, with a very straight fi...
2	bkpn1412	31.07.1983	Oregon	cea76118f6a9110a893de2b7654319c0	0	I like the jacket accept it is thin not and th...
3	bkpn1412	31.07.1983	Oregon	bc177e6724df06c676fd526e6d32461d	2	Soft, comfortable, and it fits nice. Whats not...
4	bkpn1412	31.07.1983	Oregon	a0d0f5f772021d6320048dc7b61fb551	5	I bought the quick dri bath rug in the size 21...

## Converting DOB's to age bands

In [61]: `from datetime import datetime #Importing Library`

```
In [62]: merged["DOB"] = pd.to_datetime(merged["DOB"], format="%d.%m.%Y", errors="coerce")

today = datetime.now() #Create a variable with today's date

merged["Age"] = 2025 - merged["DOB"].dt.year #Converting DOB to "age"
merged["Age"] = merged["Age"].astype(int, errors = "ignore") #Making "age" int v

bands = ["20-29", "30-39", "40-49", "50-59", "60-69", "70-79"] #Defining age ban
bins = [20,30,40,50,60,70,80] #Defining bins

merged["Age_band"] = pd.cut(merged["Age"], #Sorting values into bins
                             bins=bins,
                             labels=bands,
                             right=False) #Defining cut off for age bands

ages_counts = merged.groupby("Age_band").size().reset_index(name = "Reviews") #C

display(merged.head())
display(ages_counts.head())
```

C:\Users\Danie\AppData\Local\Temp\ipykernel\_26752\512896493.py:16: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.

	Username	DOB	State	Uniq_id	Score	Review	Age
0	bkpn1412	1983-07-31	Oregon	2565bf02ec05a59cd1fd78955dda108d	0	Love this heated mattress pad! Ordered the que...	4%
1	bkpn1412	1983-07-31	Oregon	e09b42d19cd350f9762f92d59ecf88ba	2	I am 511 and a size 6, with a very straight fi...	4%
2	bkpn1412	1983-07-31	Oregon	cea76118f6a9110a893de2b7654319c0	0	I like the jacket accept it is thin not and th...	4%
3	bkpn1412	1983-07-31	Oregon	bc177e6724df06c676fd526e6d32461d	2	Soft, comfortable, and it fits nice. Whats not...	4%
4	bkpn1412	1983-07-31	Oregon	a0d0f5f772021d6320048dc7b61fb551	5	I bought the quick dri bath rug in the size 21...	4%



	Age_band	Reviews
0	20-29	4430
1	30-39	7382
2	40-49	7261
3	50-59	7553
4	60-69	7910

## Visualising age group data

### Figure 2.1 in report

```
In [63]: plt.figure(figsize=(7,7)) #Setting figure size

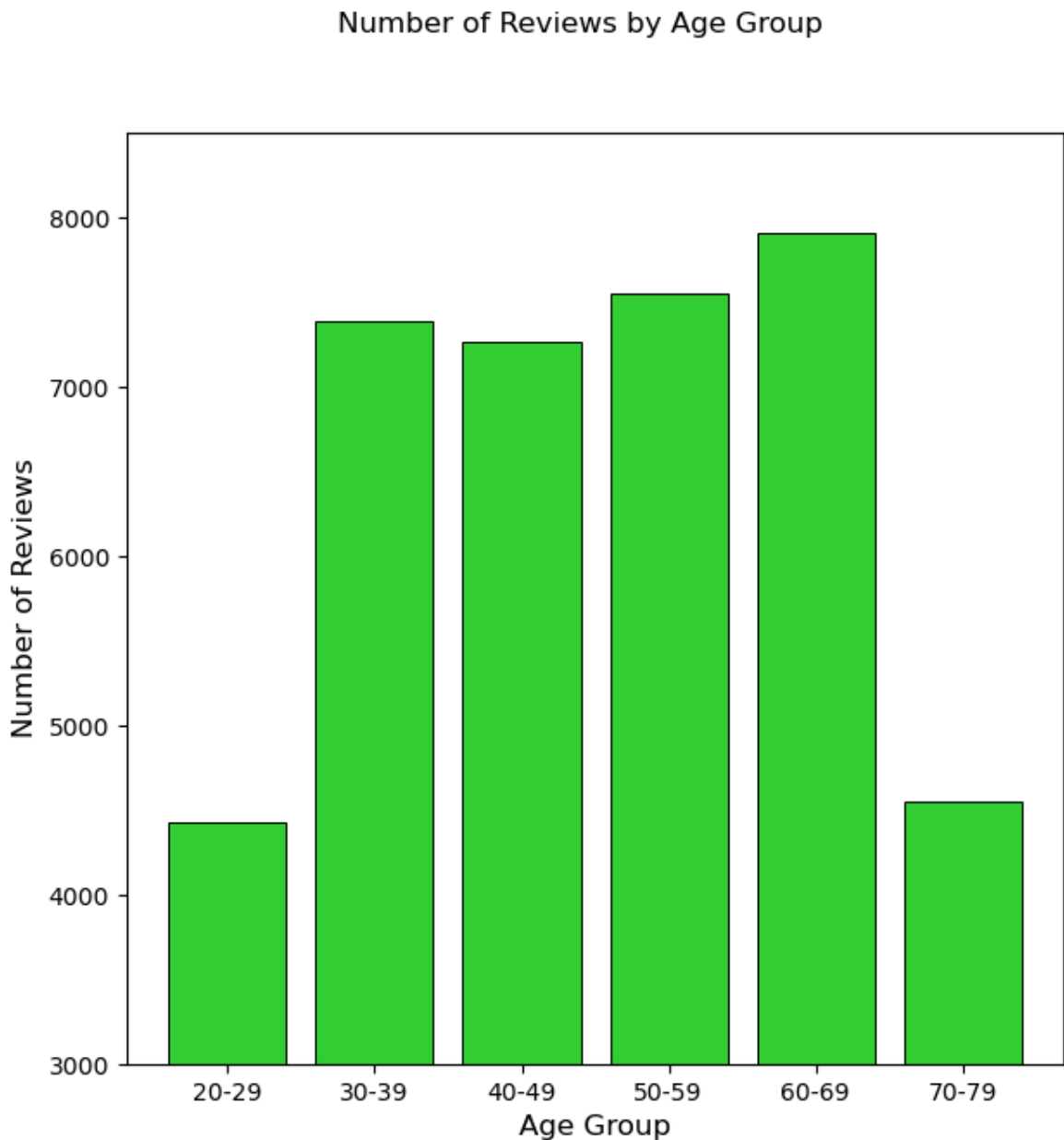
plt.bar(ages_counts["Age_band"],ages_counts["Reviews"], color="limegreen",edgeco
```

```
linewidth=0.7) #Plotting bar chart

plt.suptitle("Number of Reviews by Age Group")
plt.xlabel("Age Group", fontsize=12)
plt.ylabel("Number of Reviews", fontsize=12)
plt.ylim(3000,8500) #Changed y axis to better show differences between groups

plt.savefig("ages.png", dpi=300, bbox_inches='tight') #Saving figure to export t

plt.show()
```



## Considerations and further work

For this assignment, I decided to only use the data provided and try to do the basics well, not overcomplicating things. I performed basic feature engineering, for e.g. creating age and age band columns from DOB. Additionally, only having 5 pages to report meant there were limits to what I could include. To expand on the work I have done here, I would consider the following to be appropriate:

- Find data on population density in each US state to compare this with the Users by US State data, identifying whether the distribution of users is comparative to the population distributions between US states.
- Plot various variables using Linear Regression to look for correlations, for e.g. I would have used this method to explore the correlation between product price and average user rating to identify whether more expensive products receive better reviews.
- Look at sale price to show how much each product had been reduced
- Identify average product score at different price points to identify if there is a price point which receives the best average product ratings.

In [ ]: