# University of Stirling

## ITNPBD2 Representing and Manipulating Data

## Assignment Autumn 2025

## A Consultancy Job for JC Penney

NOTE: This version of my code is not executed to meet the brief of 15 pages maximum, I have also included a version of this code executed for clarity

### Data provided

The data provided for this assessmement included data on:

- Products.csv - Unique Id, SKU, product name, product description, prive and average score
- Users.csv - Username, DOB, State
- Reviews.csv - Uniqe 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 exmaple 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 prodicts 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 lest 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 [ ]:
```
# Importing libraries needed

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

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

os.getcwd()
```

In [ ]:
```
# 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 [ ]:
```
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
```

## 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 [ ]:
```
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
```

# 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 [ ]:
```
# 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)
```

# jcpenney_products.json initial exploration and cleaning

Imported data in pandas, explored data and perfomed basic cleaning

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

products_json = pd.read_json("jcpenney_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)
```

In [ ]:
```
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)
```

# jcpenney_reviewers.json initial exploration and cleaning

In [ ]:
```
#As above, loading dataframe, looking at size and for missing values

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

display(reviews_json.head())

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

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

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

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

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

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

In [ ]:
```
#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 insghts 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 "averge rating", I created a subset and took an overall average for each product grouped by SKU to represent an individual product.

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

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

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

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

display(subset_sorted)
```

In [ ]:
```
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
```

In [ ]:
```
# 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 = Fal

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

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
```

In [ ]:
```
display(bottom_10)
```

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

plt.subplot(121) #Defining sublot 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 readbility

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

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=1

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

plt.show()
```

This graph would hve 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 [ ]:
```
print(top_reviewed.shape) #To see how many products there are in total
```

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

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

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

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 unneccessary rows and creating a smaller data subset with average rating and total number of ratings

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

In [ ]:
```
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
```

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

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

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

In [ ]:
```
# 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)
```

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

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

In [ ]:
```
brands_sorted = brands_sorted[x] #Updating data frame to reflect change above

display(brands_sorted.head())
```

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

## Plotting the 10 best and worst reviewed brands

# Figure 1.2 in report

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

plt.subplot(211) #Defining sublot 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 readbility

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()
```

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

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

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

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

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

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

print(state_count)
```

## 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 [ ]:
```
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))
```

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

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

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

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

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

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

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

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

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

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": "T
    "Utah": "UT", "Vermont": "VT", "Virginia": "VA", "Washington": "WA",
    "West Virginia": "WV", "Wisconsin": "WI", "Wyoming": "WY",
    "District of Columbia": "DC"}
```

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

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

In [ ]:
```
# 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()
```

In [ ]:
```
#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)
```

# Analysing number of reviews by age group

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

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

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

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

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

## Converting DOB's to age bands

In [ ]:
```
from datetime import datetime #Importing library
```

In [ ]:
```
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())
```

## Visualising age group data

# Figure 2.1 in report

In [ ]:
```
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 perfomed basic feature engineering, for e.g. creating age and age band columns from DOB. Aditionally, 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 distirbutions 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 identiy whether more expensive products recieve 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 recieves the best average product ratings.

In [ ]: