

🚩 Code Block 1: Importing Libraries This code imports all the necessary Python libraries required for data handling, visualization, machine learning (PCA, KMeans), and model deserialization using pickle.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import pickle
```

```
# Import the relevant libraries
```

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

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.rc("font", size=14)
import seaborn as sns
sns.set()
sns.set(style="whitegrid", color_codes=True)
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

```
import pickle
```

🚩 Code Block 2: Loading the Dataset This block loads the dataset purchase data.csv, displays the first few rows, checks its structure, checks for null values, and identifies duplicate entries

```
# Load the dataset
```

```
purchase_df = pd.read_csv('purchase data.csv', index_col = 0)
```

🚩 Code Block 3: Loading Pre-trained Models This block loads the serialized scaler, PCA, and KMeans models from .pickle files to apply the same transformations used during training for consistent segmentation.

```
# Data quality checks
print("First 5 rows of the dataset:")
print(purchase_df.head())
print("\nDataset Info:")
print(purchase_df.info())
print("\nNumber of duplicated rows:", purchase_df.duplicated().sum())
print("\nMissing values:")
print(purchase_df.isnull().sum())
print("\nSummary Statistics:")
print(purchase_df.describe())
print("\nIndex uniqueness check:", purchase_df.index.is_unique, "(False indicates duplicate IDs)")
```

🔗 First 5 rows of the dataset:

ID	Day	Incidence	Brand	Quantity	Last_Inc_Brand	Last_Inc_Quantity	\
200000001	1	0	0	0	0	0	
200000001	11	0	0	0	0	0	
200000001	12	0	0	0	0	0	
200000001	16	0	0	0	0	0	
200000001	18	0	0	0	0	0	

ID	Price_1	Price_2	Price_3	Price_4	...	Promotion_3	Promotion_4	\
200000001	1.59	1.87	2.01	2.09	...	0	0	
200000001	1.51	1.89	1.99	2.09	...	0	0	
200000001	1.51	1.89	1.99	2.09	...	0	0	
200000001	1.52	1.89	1.98	2.09	...	0	0	
200000001	1.52	1.89	1.99	2.09	...	0	0	

ID	Promotion_5	Sex	Marital status	Age	Education	Income	\
200000001	0	0		0	47	1	110866
200000001	0	0		0	47	1	110866
200000001	0	0		0	47	1	110866
200000001	0	0		0	47	1	110866
200000001	0	0		0	47	1	110866

ID	Occupation	Settlement size
200000001	1	0

```

200000001      1      0
200000001      1      0
200000001      1      0
200000001      1      0

```

[5 rows x 23 columns]

Dataset Info:

```
<class 'pandas.core.frame.DataFrame'>
```

Index: 58693 entries, 200000001 to 200000500

Data columns (total 23 columns):

#	Column	Non-Null Count	Dtype
0	Day	58693 non-null	int64
1	Incidence	58693 non-null	int64
2	Brand	58693 non-null	int64
3	Quantity	58693 non-null	int64
4	Last_Inc_Brand	58693 non-null	int64
5	Last_Inc_Quantity	58693 non-null	int64
6	Price_1	58693 non-null	float64
7	Price_2	58693 non-null	float64
8	Price_3	58693 non-null	float64
9	Price_4	58693 non-null	float64
10	Price_5	58693 non-null	float64
11	Promotion_1	58693 non-null	int64
12	Promotion_2	58693 non-null	int64
13	Promotion_3	58693 non-null	int64
14	Promotion_4	58693 non-null	int64
15	Promotion_5	58693 non-null	int64
16	Sex	58693 non-null	int64

Import Segmentation Model

```

scaler = pickle.load(open('scaler.pickle', 'rb'))
pca = pickle.load(open('pca.pickle', 'rb'))
kmeans_pca = pickle.load(open('kmeans_pca.pickle', 'rb'))

```

Customer Analytics in FMGC Industry (Part 2) by Sooyeon Won Keywords Marketing Mix STP framework Purchase Analytics Descriptive Analysis by Segments Data Visualisations Contents

1. Introduction
2. Data Preparation
3. Data Exploration
4. Data Analysis
  - 4.1. Customer Analytics
    - Proportion of each Segment
    - Revenue Comparison between segments
    - Modeling Brand Choice
  - 4.2. Purchase Analytics
    - i. Descriptive Analyses by Segment
      - i-1. The Proportion of each Segment
      - i-2. Purchase Occasions and Purchase Incidences
      - i-3. Brand Chocie
      - i-4. Revenue Comparison between segments: How much money they spend?
    - ii. Predictive Analyses
      - ii-1. Modeling Purchase Incidence
      - ii-2. Modeling Purchase Quantity
5. Conclusion
6. Data Analysis
7. 2. Purchase Analytics Data Preparation from Part 1
  - i. Descriptive Analysis by Segments
    - i-1. The Proportion of each Segment
    - i-2. Purchase Occasions and Purchase Incidences: How often each segment group go shopping?
    - i-3. Brand Chocie: Which brand of products the customers purchase?
    - i-4. Revenue Comparison between segments: How much money they spend?

Standardisation of Demographic Features

```

# Assign customer segments
demographic_features = ['Sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation', 'Settlement size']
customer_ids = purchase_df.index.unique()
demo_df = purchase_df[demographic_features].drop_duplicates()
features = purchase_df[['Sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation', 'Settlement size']]
purchase_segm_std = scaler.transform(features)

```

PCA Transformation

```
purchase_segm_pca = pca.transform(purchase_segm_std)
```

K-means Clustering with PCA


```

# Predict clusters based on K-means Clustering with PCA
purchase_segm_kmeans_pca= kmeans_pca.predict(purchase_segm_pca)

```

```
# Checkpoint
purchase_predictors = purchase_df.copy()

purchase_predictors['Segment'] = purchase_segm_kmeans_pca
purchase_predictors = purchase_predictors.reset_index()
purchase_predictors.head()
```



	ID	Day	Incidence	Brand	Quantity	Last_Inc_Brand	Last_Inc_Quantity	Price_1	Price_2	Price_3	...	Promotion_
0	200000001	1	0	0	0	0	0	1.59	1.87	2.01	...	
1	200000001	11	0	0	0	0	0	1.51	1.89	1.99	...	
2	200000001	12	0	0	0	0	0	1.51	1.89	1.99	...	
3	200000001	16	0	0	0	0	0	1.52	1.89	1.98	...	
4	200000001	18	0	0	0	0	0	1.52	1.89	1.99	...	

5 rows × 25 columns

i. Descriptive Analysis by Segments In this section, I focus on descriptive analysis of the purchase data by individuals and then by segments to gain insight into customer shopping habits.

i-1. The Proportion of each Segment i-2. Purchase Occasions and Purchase Incidences: How often each segment group go shopping? i-3. Brand Choice: Which brand of products the customers purchase? i-4. Revenue Comparison between segments: How much money they spend? i-1. Segment Proportion Based on the customer data from the company's loyalty program, I firstly compare the size of each segment. To do so, I aggregated the data by Customers then grouped by each segment.


Aggregate the Data by Customers

```
traffic_freq = purchase_predictors[['ID', 'Incidence']].groupby(['ID']).count()
purchase_freq = purchase_predictors[['ID', 'Incidence']].groupby(['ID']).sum()
segm_info = purchase_predictors[['ID', 'Segment']].groupby(['ID']).mean()

traffic_freq = traffic_freq.rename(columns = {'Incidence': 'N_Visits'})
purchase_freq = purchase_freq.rename(columns = {'Incidence': 'N_Purchases'})

# Average number of purchases by customer ID
df_customer = traffic_freq.join(purchase_freq)
df_customer['Avg_N_Purchases'] = df_customer.N_Purchases/ df_customer.N_Visits


purchase_descr = df_customer.join(segm_info)
purchase_descr.head()
```



	N_Visits	N_Purchases	Avg_N_Purchases	Segment
ID				
200000001	101	9	0.089109	0.0
200000002	87	11	0.126437	3.0
200000003	97	10	0.103093	0.0
200000004	85	11	0.129412	1.0
200000005	111	13	0.117117	0.0

Group-by each segment

```
# Calculate the proportions of each segment and set the appropriate column name.
segm_prop = purchase_descr[['N_Visits', 'Segment']].groupby(['Segment']).count() / purchase_descr.shape[0]
segm_prop = segm_prop.rename(columns = {'N_Visits': 'Segment Proportions'}).reset_index()
segm_prop.Segment.replace({0: 'Fewer-Opportunities', 1: 'Career-Focused', 2: "Standard", 3:"Well-Off"}, inplace=True)
segm_prop.head()
```

 <ipython-input-48-09466b293fbb>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c

```
segm_prop.Segment.replace({0: 'Fewer-Opportunities', 1: 'Career-Focused', 2: "Standard", 3:"Well-Off"}, inplace=True)
```

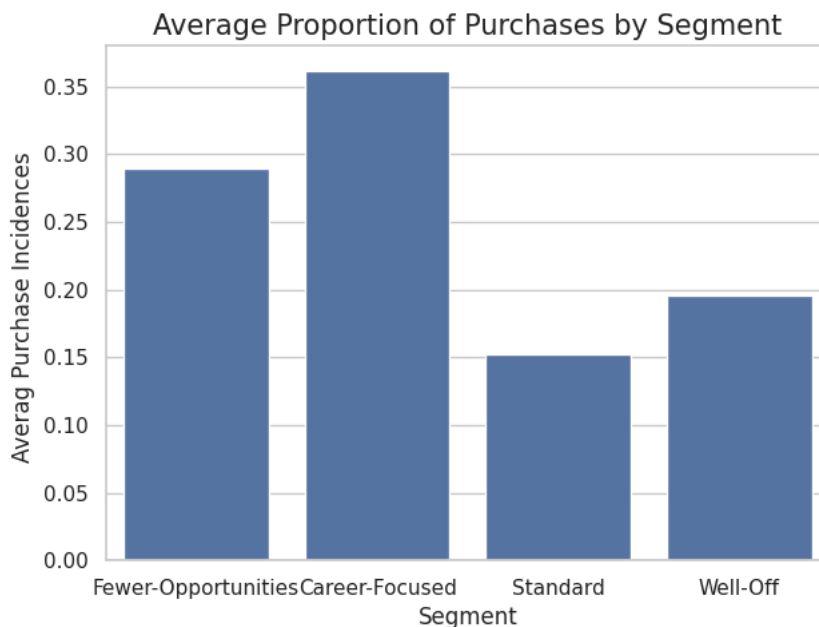
	Segment	Segment Proportions
0	Fewer-Opportunities	0.290
1	Career-Focused	0.362
2	Standard	0.152
3	Well-Off	0.196

There are 58,693 purchase records with 23 features in this dataset. The dataset does not contain missing nor duplicated values. The explanation of each feature is as follows:

ID (numerical): a unique identifier of a customer. Day (numerical) when the customer has visited the store. Incidence (categorical) Purchase Incidence. Did the customer purchase a product? 0 The customer has not purchased an item from the category of interest 1 The customer has purchased an item from the category of interest Brand (categorical) Shows which brand the customer has purchased. 0 No brand was purchased 1,2,3,4,5 Chocolate Product Brand ID Quantity (numerical) Number of items bought by the customer from the product category of interest Last\_Inc\_Brand (categorical) Shows which brand the customer has purchased on their previous store visit 0 No brand was purchased 1,2,3,4,5 Chocolate Product Brand ID Last\_Inc\_Quantity (numerical) Number of items bought by the customer from the product category of interest during their previous store visit. Price\_1,2,3,4,5 (numerical) Price of an item from Brand 1,2,3,4,5 on a particular day Promotion\_1,2,3,4,5 (categorical) Indicator whether Brand 1,2,3,4,5 was on promotion or not on a particular day Rest features (Sex, Marital status, Age, Education, Income, Occupation, Settlement size) are consistent with the features in demographic dataset.


```
plt.figure(figsize=[7,5])
sns.barplot(data = segm_prop, x = 'Segment', y = 'Segment Proportions')
plt.xlabel('Segment', fontsize = 12)
plt.ylabel('Average Purchase Incidences ', fontsize = 12)
plt.title('Average Proportion of Purchases by Segment', fontsize = 15)
```

 Text(0.5, 1.0, 'Average Proportion of Purchases by Segment')



The largest segment is "Fewer-Opportunities". Almost 40 percent of customers belong to this segment. The second largest segment is "career-focused". 22.2 % of customers are in this segment. The proportion of the other groups "Standard" and "Well-Off" are similar to ca. 20%.

```
# Calculate the mean by the four segments to determine the average customer behaviour in each segment.
segments_mean = purchase_descr.groupby(['Segment']).mean().reset_index()
segments_mean.Segment.replace({0: 'Fewer-Opportunities', 1: 'Career-Focused', 2: "Standard", 3:"Well-Off"}, inplace=True)
segments_mean
```

 <ipython-input-50-8afde015c8e5>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c

```
segments_mean.Segment.replace({0: 'Fewer-Opportunities', 1: 'Career-Focused', 2: "Standard", 3:"Well-Off"}, inplace=
```

	Segment	N_Visits	N_Purchases	Avg_N_Purchases
0	Fewer-Opportunities	120.489655	34.965517	0.258081
1	Career-Focused	114.303867	22.823204	0.201760
2	Standard	118.828947	27.171053	0.228956
3	Well-Off	117.367347	34.408163	0.282601

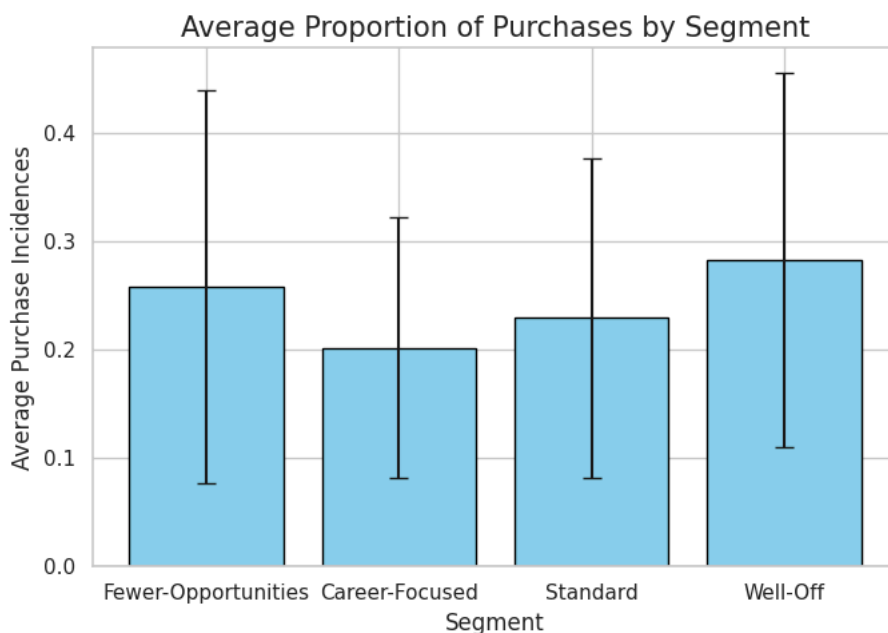
```
# Calculate the standard deviation by segments. It determines how homogeneus each of the segments is.
segments_std = purchase_descr.groupby(['Segment']).std()
```

```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Data
segments = segments_mean['Segment']
means = segments_mean['Avg_N_Purchases']
errors = segments_std['Avg_N_Purchases']
```

```
# Plot
plt.figure(figsize=[7,5])
plt.bar(segments, means, yerr=errors, capsize=5, color='skyblue', edgecolor='black')

plt.xlabel('Segment', fontsize=12)
plt.ylabel('Average Purchase Incidences', fontsize=12)
plt.title('Average Proportion of Purchases by Segment', fontsize=15)
plt.tight_layout()
plt.show()
```



i-3. Brand Choice Now we come to the question: Which brand is the customer going to choose? I focused on the observations only where customers have bought at least one chocolate candy bar.

```
# Standardize demographic features and apply PCA and KMeans
X_demo = scaler.transform(demo_df)
X_pca = pca.transform(X_demo)
cluster_labels = kmeans_pca.predict(X_pca)

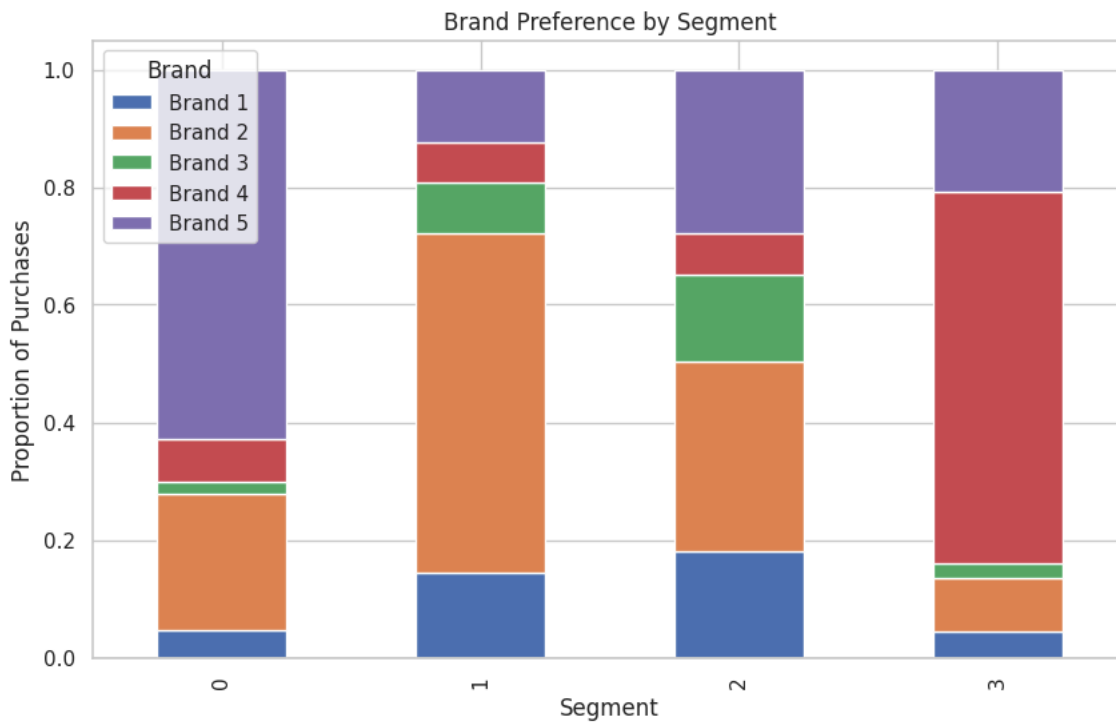
# Create a mapping of customer IDs to cluster labels
id_to_cluster = pd.Series(cluster_labels, index=demo_df.index)
purchase_df['Cluster'] = purchase_df.index.map(id_to_cluster)

# Descriptive Analytics by Segment
```

```
# 1. Purchase Frequency (Incidence Rate)
incidence_by_segment = purchase_df.groupby('Cluster')['Incidence'].mean().reset_index()
incidence_by_segment.columns = ['Cluster', 'Purchase_Incidence_Rate']

# 2. Brand Preference by Segment
brand_by_segment = purchase_df[purchase_df['Incidence'] == 1].groupby(['Cluster', 'Brand']).size().unstack(fill_value=0)
brand_by_segment = brand_by_segment.div(brand_by_segment.sum(axis=1), axis=0) # Normalize to percentages

# Plot Brand Distribution by Segment
brand_by_segment.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Brand Preference by Segment')
plt.xlabel('Segment')
plt.ylabel('Proportion of Purchases')
plt.legend(title='Brand', labels=[f'Brand {i}' for i in brand_by_segment.columns])
plt.show() # Display inline
```



```
# 3. Average Quantity and Spend per Purchase
# Reset index to avoid duplicate label issues
purchase_df_reset = purchase_df.reset_index()

# Initialize Spend column
purchase_df_reset['Spend'] = 0.0

# Verify price columns exist
price_cols = [f'Price_{i}' for i in range(1, 6)]
missing_cols = [col for col in price_cols if col not in purchase_df_reset.columns]
if missing_cols:
    print(f"Error: Missing price columns: {missing_cols}")
    exit()

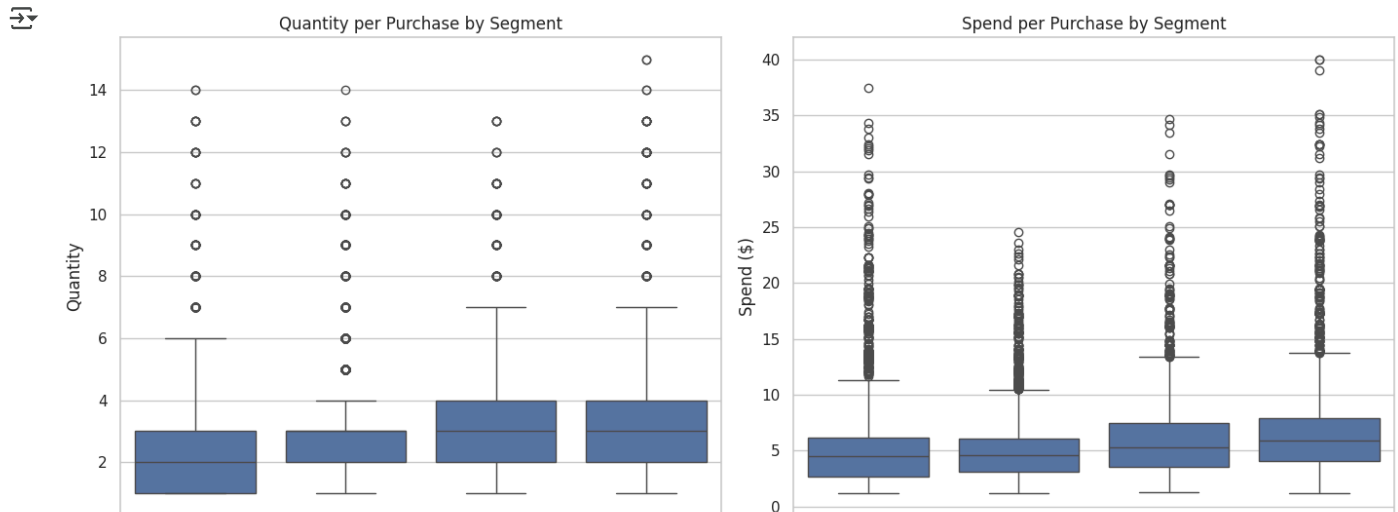
# Calculate Spend (Price * Quantity for purchased brand)
for i in range(1, 6):
    purchase_df_reset.loc[(purchase_df_reset['Incidence'] == 1) & (purchase_df_reset['Brand'] == i), 'Spend'] = (
        purchase_df_reset[f'Price_{i}'] * purchase_df_reset['Quantity']
    )

# Restore ID index
purchase_df = purchase_df_reset.set_index('ID')

# Compute average quantity and spend by segment
quantity_spend_by_segment = purchase_df[purchase_df['Incidence'] == 1].groupby('Cluster')[['Quantity', 'Spend']].mean().reset_index()

# Plot Quantity and Spend by Segment
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
sns.boxplot(x='Cluster', y='Quantity', data=purchase_df[purchase_df['Incidence'] == 1], ax=axes[0])
axes[0].set_title('Quantity per Purchase by Segment')
axes[0].set_xlabel('Segment')
axes[0].set_ylabel('Quantity')
sns.boxplot(x='Cluster', y='Spend', data=purchase_df[purchase_df['Incidence'] == 1], ax=axes[1])
axes[1].set_title('Spend per Purchase by Segment')
axes[1].set_xlabel('Segment')
```

```
axes[1].set_ylabel('Spend ($)')
plt.tight_layout()
plt.show() # Display inline
```



```
# 4. Promotion Effectiveness by Segment
```

```
promotion_cols = ['Promotion_1', 'Promotion_2', 'Promotion_3', 'Promotion_4', 'Promotion_5']
```

```
promotion_by_segment = purchase_df[purchase_df['Incidence'] == 1].groupby('Cluster')[promotion_cols].mean().reset_index()
```

```
# Plot Promotion Usage Heatmap
```

```
plt.figure(figsize=(10, 6))
```

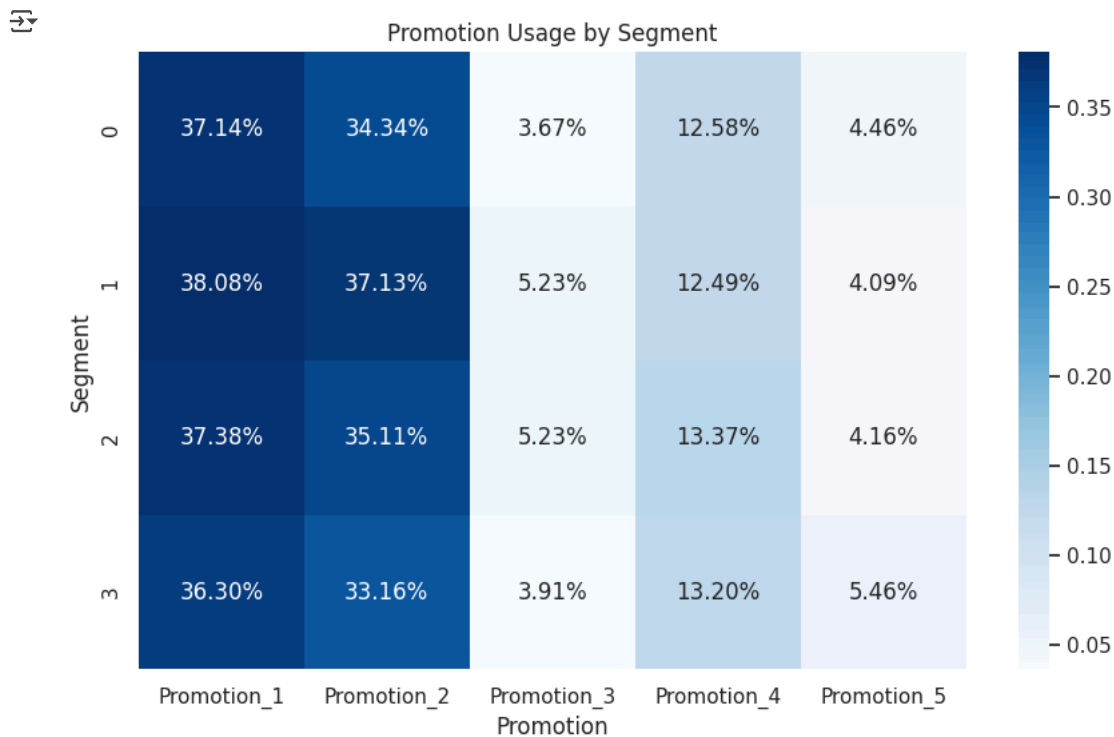
```
sns.heatmap(promotion_by_segment.set_index('Cluster'), annot=True, cmap='Blues', fmt='.2%')
```

```
plt.title('Promotion Usage by Segment')
```

```
plt.xlabel('Promotion')
```

```
plt.ylabel('Segment')
```

```
plt.show() # Display inline
```



```
# Select only rows where incidence is 1. This means that I filtered the times a purchase was made.
```

```
purchase_incidence = purchase_predictors.query('Incidence ==1')
```

```
purchase_incidence.shape
```

```
(14638, 25)
```

```
# Here we make dummies for each of the five brands.
```

```
brand_dummies = pd.get_dummies(purchase_incidence['Brand'], prefix = 'Brand', prefix_sep = '_')
```

```
brand_dummies['Segment'] = purchase_incidence['Segment'], purchase_incidence['ID']
```

```
brand_dummies.head()
```

```

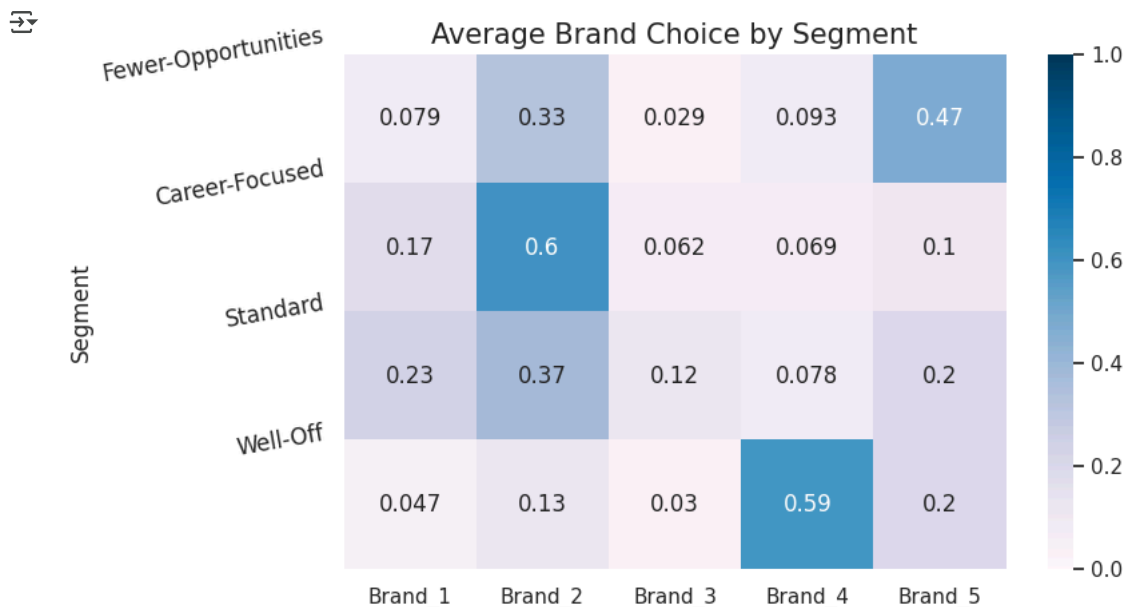
Brand_1 Brand_2 Brand_3 Brand_4 Brand_5 Segment ID
6 False True False False False 0 200000001
11 False False False False True 0 200000001
19 True False False False False 0 200000001
24 False False False True False 0 200000001
29 False True False False False 0 200000001

brand_ID = brand_dummies.groupby(['ID'], as_index = True).mean() # group by each customer

mean_brand_segment = brand_ID.groupby(['Segment'], as_index = True).mean() # group by each segmen

plt.figure(figsize=[8,5])
sns.heatmap(mean_brand_segment, vmin = 0, vmax = 1, cmap = 'PuBu', annot = True)
plt.yticks([0, 1, 2, 3], ['Fewer-Opportunities', 'Career-Focused', 'Standard', 'Well-Off'], rotation = 10, fontsize = 12)
plt.title('Average Brand Choice by Segment', fontsize = 15)
plt.show()

```



Note that each chocolate brands are listed in ascending order of price. This means that brand 1 is the cheapest brand, while brand 5 is the most expensive one.

Fewer-Opportunities segment shows an extremely strong preference for brand\_2. Almost 70 percent of the segment chooses this brand of chocolate. Brand\_2 Chocolate certainly is not the cheapest one. Thus, we can guess that the price of chocolate bars is not really matters to our customers to purchase. Career-Focused segment: ca. 63 percent of the career focus segment buys brand\_5 which is the most expensive brand. It seems that this cluster of young ambitious career focused individuals enjoys this fancy candy bar with no additional information. It can be speculated that the career focus segment is looking for luxury status and this alone may be an opportunity to raise the price of brand 5 even further. The Well-Off segment enjoys one of the most luxurious brands but not the most expensive one. Brand\_4 is the most popularly bought brand followed by brand\_5. Standard segment is the most heterogeneous segment. It seems that people from the standard segment prefer brand\_2 and a weaker preference for brands 1 and 3. It's obvious that this segment don't like buying brand 4. Nevertheless, their preference is scattered all around. One idea could be to try to influence them to try to different brands. i-4. Dissecting the Revenue by Segment In this section, I explore the revenue by segment. The revenue is calculated by multiplying product price with purchased quantity.

```

# Filter the datapoints by each brand
brand_1 = purchase_predictors.query('Brand ==1').copy()
brand_2 = purchase_predictors.query('Brand ==2').copy()
brand_3 = purchase_predictors.query('Brand ==3').copy()
brand_4 = purchase_predictors.query('Brand ==4').copy()
brand_5 = purchase_predictors.query('Brand ==5').copy()

# Compute the revenue of each brand
brand_1['Revenue_Brand_1'] = brand_1['Price_1'] * brand_1['Quantity']
brand_2['Revenue_Brand_2'] = brand_2['Price_2'] * brand_2['Quantity']
brand_3['Revenue_Brand_3'] = brand_3['Price_3'] * brand_3['Quantity']
brand_4['Revenue_Brand_4'] = brand_4['Price_4'] * brand_4['Quantity']
brand_5['Revenue_Brand_5'] = brand_5['Price_5'] * brand_5['Quantity']

# Aggregate the revenues by segment

```



```

brand_1_seg = brand_1.groupby('Segment')['Revenue_Brand_1'].sum()
brand_2_seg = brand_2.groupby('Segment')['Revenue_Brand_2'].sum()
brand_3_seg = brand_3.groupby('Segment')['Revenue_Brand_3'].sum()
brand_4_seg = brand_4.groupby('Segment')['Revenue_Brand_4'].sum()
brand_5_seg = brand_5.groupby('Segment')['Revenue_Brand_5'].sum()

# Create the final DataFrame
segments_brand_revenue = pd.DataFrame({
    'Segment': ['Fewer-Opportunities', 'Career-Focused', 'Standard', 'Well-Off'],
    'Revenue_Brand_1': brand_1_seg.values,
    'Revenue_Brand_2': brand_2_seg.values,
    'Revenue_Brand_3': brand_3_seg.values,
    'Revenue_Brand_4': brand_4_seg.values,
    'Revenue_Brand_5': brand_5_seg.values
})

# Sum only the revenue columns to get the total revenue
revenue_columns = ['Revenue_Brand_1', 'Revenue_Brand_2', 'Revenue_Brand_3', 'Revenue_Brand_4', 'Revenue_Brand_5']
segments_brand_revenue['Total_Rev'] = segments_brand_revenue[revenue_columns].sum(axis=1)

# Add segment proportions (assuming segm_prop is already defined)
segments_brand_revenue['Segment Proportions'] = segm_prop['Segment Proportions']

```

```
segments_brand_revenue
```

	Segment	Revenue_Brand_1	Revenue_Brand_2	Revenue_Brand_3	Revenue_Brand_4	Revenue_Brand_5	Total_Rev	Segment Proportions
0	Fewer-Opportunities	912.12	5222.99	603.79	2268.95	17838.72	26846.57	0.290
1	Career-Focused	2807.51	11742.86	2388.88	1910.89	2638.52	21488.66	0.362
2	Standard	1869.77	3382.47	2303.31	1165.16	3638.75	12359.46	0.152

```

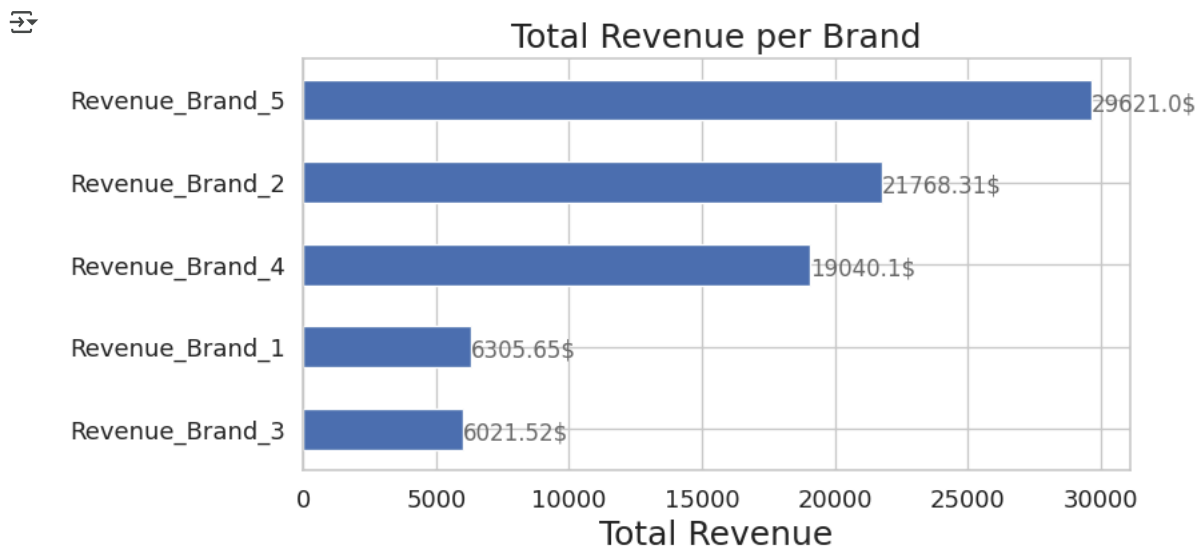
brand_rev = segments_brand_revenue.iloc[:,1:6].sum(axis=0)
ax = brand_rev.sort_values(ascending=False).plot(kind='barh', figsize=(8,4), color="b", fontsize=13);
ax.set_alpha(0.8)
ax.set_title("Total Revenue per Brand", fontsize=18)
ax.set_xlabel("Total Revenue", fontsize=18);

```

```

# create a list to collect the plt.patches data
totals = []
# find the values and append to list
for i in ax.patches:
    totals.append(i.get_width())
# set individual bar labels using above list
total = sum(totals)
# set individual bar labels using above list
for i in ax.patches:
    # get_width pulls left or right; get_y pushes up or down
    ax.text(i.get_width()+.3, i.get_y()+.38, \
            str(round((i.get_width()), 2))+'$', fontsize=12, color='dimgrey')
# invert for largest on top
ax.invert_yaxis()

```



Brand 3 has the lowest revenue compared to the other products. It is the middle brand in terms of price, and its highest contributor is the "Standard" segment. Standard segment would like the first three brands, so they can be influenced to buy more of the 3rd brand. Maybe if brand 3 reduces its price it is likely that the standard segment would pivot towards it.

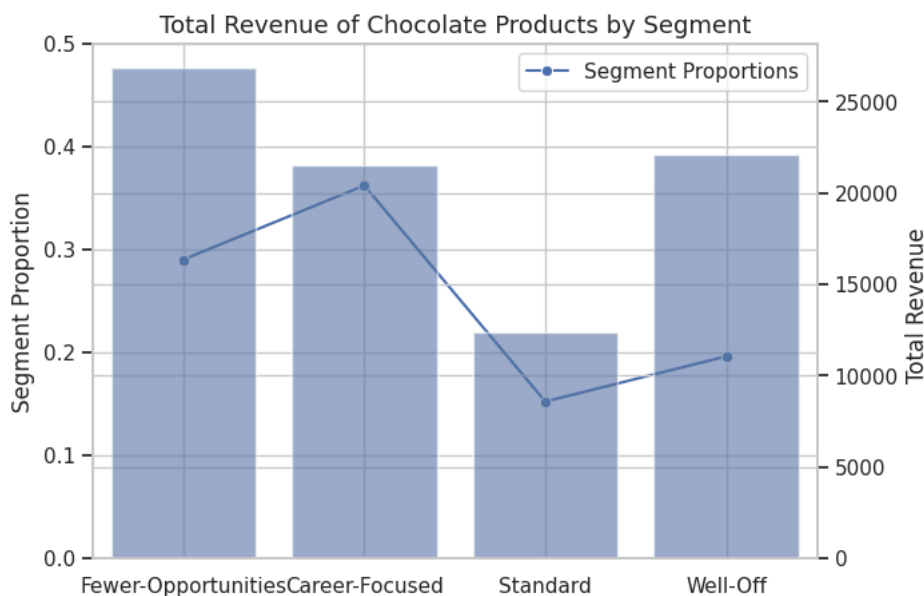
The customers of Brand 4 seem mainly from the "Well-Off" segment. The customers in this segment who did not choose this brand bought an even more expensive alternative: Brand 5. Therefore, they seem to be loyal and not really affected by price. Therefore brand 4 could try cautiously increasing its price. The hypothesis here is they would likely retain most of its customers and increase the revenue per sale.

```
ax1 = sns.set_style()
fig, ax1 = plt.subplots(figsize=[7,5])
sns.lineplot(data = segm_prop, marker='o', sort = False, ax=ax1, color = 'black')
ax1.set_ylabel('Segment Proportion', fontsize = 12)
ax1.set_ylim([0,0.5])
ax2 = ax1.twinx()
sns.barplot(data = segments_brand_revenue, x = 'Segment', y = 'Total_Rev', alpha=0.6)
ax2.set_ylabel('Total Revenue ', fontsize = 12)

plt.xlabel('Segment', fontsize = 12)

plt.title('Total Revenue of Chocolate Products by Segment', fontsize = 13)
```

↗ Text(0.5, 1.0, 'Total Revenue of Chocolate Products by Segment')



In the last plot, I compare the size of each segment with the revenue from each segment. The size of each segment is presented using a line.

"Career-Focused" brings the most revenue followed by "Well-Off" and "Fewer-Opportunities", while the "Standard" segment brings the least. Considering that "Career-Focused" is not the largest segment, it is a surprising finding. In Part 1 of the analysis, we found out that "Career-focused" segment was buying the most expensive brand. Indeed, it seems that they are by far the most prominent segment for the store regarding chocolates. "Standard" segment is almost as big as "Career-focused". However, it brings far less revenue to the company. In fact, "Standard" segment contributes the least of all segments in comparison. "Well-Off" and "Fewer-Opportunities" segments spend similar amount of money on chocolates. However, note that "Fewer-Opportunities" is twice the size of "Well-Off" segment. So far, we examined different measures together. In the next part of analysis, I focused on the point of view and explore the revenue table from a brand's perspective.