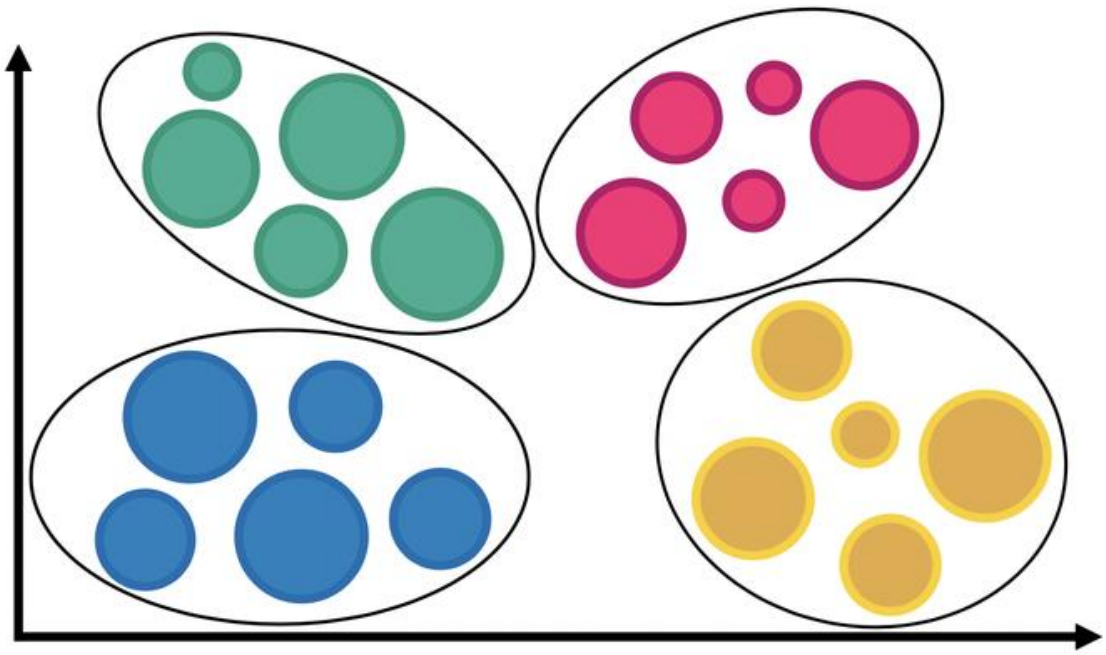
eCommerce Canvas Clusters:

Customer Segmentation / clustering Analysis

GitHub: https://github.com/Milimia/e-commerce-canvas-cluster-Customer-Segmentation-clustering-Analysis-



1. Clustering:

- Clustering is a Machine Learning Technique used for grouping similar objects or data points together into distinct clusters.
- It is a method of Unsupervised Learning in Machine Learning.
- It aims to discover hidden structures, relationships, or natural grouping within datasets without prior labelling of the data.

2. Primary Objective of Clustering:

- Is to uncover natural structures or patterns within the dataset.
- It assists in exploring the dataset and gaining a deeper understanding of its structure.

Common clustering algorithms include K-Means, Hierarchical Clustering, GMM etc.

3. Introduction:

In the dynamic realm of e-commerce, deciphering customer behaviour holds the key to thriving in a competitive landscape. Customer Segmentation emerges as a crucial compass, directing businesses to comprehend and cater to the multifaceted preferences of online shoppers. It stands as an indispensable strategy, facilitating a deeper understanding of diverse consumer needs. By delineating distinct customer segments based on behavior and preferences, this project aims to empower e- commerce enterprises with actionable insights. These insights will serve as guiding lights, enabling tailored strategies and personalized engagement to foster enhanced customer satisfaction and loyalty within the digital marketplace.

4. Objectives and Goals

The objectives and goals of the project on Customer Segmentation in E- commerce are as follows:

- Employ data- driven methodologies such as clustering algorithms, to identify distinct customer segments within the e- commerce dataset.
- Derive actionable insights from the segmented data to enable targeted marketing strategies and enhance customer experiences.

5. Data Description

Dataset Source: https://github.com/Milimia/e-commerce-canvas-cluster-Customer-Segmentation-clustering-Analysis

The Dataset encompasses various attributes capturing essential aspects of customer behaviour and engagement within an e-commerce ecosystem.

6. Methodology

❖ Importing Necessary libraries:

```
#importing necessary libraries
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer, make_column_selector
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.pipeline import FeatureUnion
from math import pi
import matplotlib.pyplot as plt
import seaborn as sns
```

- Pandas is imported as pd and Numpy as np for data manipulation and numerical operations.
- Scikit- learn Libraries:
  - From Sklearn library we import specific module called cluster and within that module we import 'KMeans' function specifically. This allows us to use KMeans algorithm for clustering task.

- We import StandardScaler and OneHotEncoder from Scikit- learns's preprocessing module. StandardScaler is used for scaling numerical features by removing the mean and scaling to unit variance. It transforms the data so that mean of each feature become zero and the standard deviation becomes 1.

- While **OneHotEncoder** is used for handling categorical features. It transforms categorical variables into a binary. It creates binary columns for each category in the original feature where only one column contains a 1 indicating the presence of that category and the others contain 0s.
- From **SkLearn** Library we import **ColumnTransformer** and **Make_column_selector** these libraries are used for applying transformation to different columns or subset of columns in a dataset.
- From **SkLearn** Library we import Pipeline's **make_pipeline** function. It allows us to combine multiple steps (such as preprocessing, feature selection or modelling) into single, sequential process.

- From **SkLearn** Library import **FeatureUnion's** function from pipeline module. It is used to concatenate results from different transformer objects applied to the same input's data.

- From Maths import Pi: this line imports the mathematical constant 'pi' from the python 'math' module.
- Import **matplotlib.pyplot** : this statement imports the '**pyplot**' module from the 'matplotlib' library a popular data visualization library in python.
- Import seaborn: this statement provides additional high-level functions for creating aesthetically pleasing statistical graphics.

```
df = pd.read_csv("C:/Users/kaurs/Downloads/E-commerce Customer Behavior - Sheet1.csv")
```

❖ This statement is used for reading data from a CSV (Comma Separated Values) file into a Pandas **DataFrame** in python.

```
print(df.head())
```

```
   Customer ID  Gender  Age           City Membership Type  Total Spend  \
0          101  Female   29       New York            Gold      1120.20
1          102    Male   34    Los Angeles          Silver       780.50
2          103  Female   43        Chicago          Bronze       510.75
3          104    Male   30  San Francisco            Gold      1480.30
4          105    Male   27          Miami          Silver       720.40

   Items Purchased  Average Rating  Discount Applied  \
0               14             4.6              True
1               11             4.1             False
2                9             3.4              True
3               19             4.7             False
4               13             4.0              True

   Days Since Last Purchase Satisfaction Level
0                        25          Satisfied
1                        18            Neutral
2                        42        Unsatisfied
3                        12          Satisfied
4                        55        Unsatisfied
```

❖ This function is used to display first few rows of data within the Pandas **DataFrame**.

In [137]:
```
numerical_features = df.select_dtypes(include=['int64','float64']).columns
```

In [138]:
```
categorical_features =df.select_dtypes(include =['object']).columns
```

In [139]:
```
numerical_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown ='ignore'))
])
```

In [140]:
```
preprocessor = ColumnTransformer(
    transformers = [
        ('num' , numerical_transformer, numerical_features),
        ('cat' ,categorical_transformer, categorical_features)
    ])
```

In [141]:
```
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
```

❖ Data Preparation:
- Identification of Features:
  - Numerical Features: Identified and extracted numerical columns.
  - Categorical Features: Identified and extracted categorical columns.

❖ Preprocessing Pipeline Setup:
- Numerical Transformer:
  - Utilized a **pipeline** from **Scikit-learn** to create a transformation process for numerical features. It includes scaling using '**StandardScaler()**' to standardize numerical data.
- Categorical Transformer:
  - Utilized another function of **pipeline** for categorical features. This incorporates '**OneHotEncoder()**' to encode categorical data, handling unknown categories with 'ignore' parameter.

❖ Column Transformer:
- Combined both transformers ('**numerical_transformer'** and '**categorical_transformer'**) into single '**ColumnTransformer'** ('**preprocessor**'). This transformer will apply specific preprocessing steps to the respective numerical and categorical features separately.

❖ Clustering Model Setup:

- Configured a K-Means clustering model with:
  - '**n_clusters**=3': Specifies the number of clusters to be formed by the K-Means algorithm.
  - '**random_state**=42': Sets the random seed to ensure reproducibility.
  - '**n_init**=10': Determines the number of times the K-Means algorithm will run with different centroid seeds.

7. Cluster Profiling

```
In [142]: final_pipeline = Pipeline(steps=[
              ('preprocessor', preprocessor),
              ('kmeans', kmeans)
          ])

In [143]: labels = final_pipeline.fit_predict(df)

In [144]: cluster_centers = final_pipeline.named_steps['kmeans'].cluster_centers_
```

❖ Pipeline Configuration:

- Utilized a '**Pipeline**' from **Scikit- Learn**, combining data processing (**'preprocessor'**) and K-Means clustering (**'kmeans'**) into a sequential workflow.

❖ Clustering Execution: Executed the constructed '**final_pipeline'** on the dataset (**'df'**) to perform clustering via the '**fit_predict'** method, generating cluster labels for each data point.

❖ Cluster Centers: Extracted cluster centers to acquire the centroids representing the mean feature values for each cluster.

8. Clustering Result

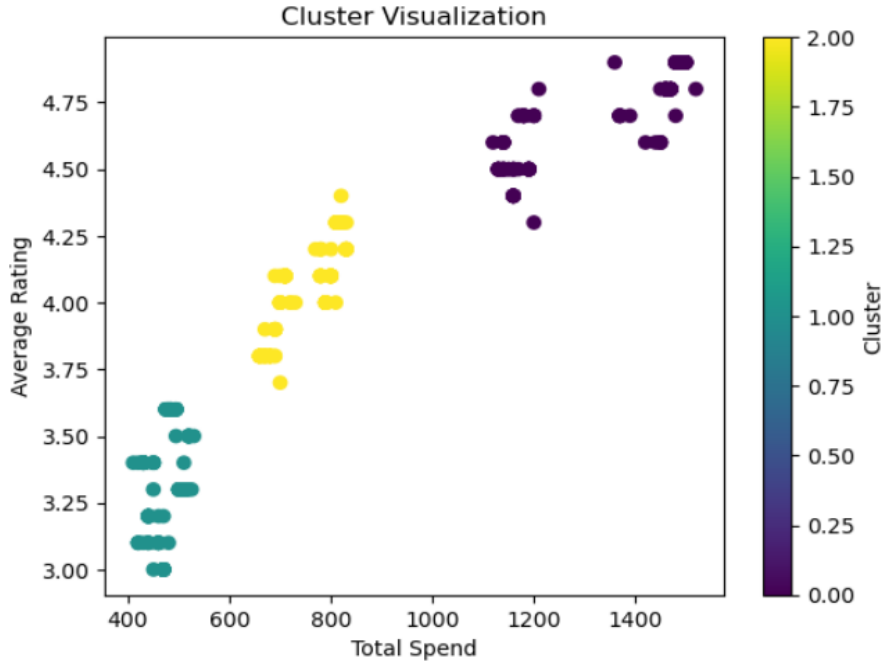
```
In [146]: cluster_stats = df.groupby(labels).agg({'Total Spend': ['mean', 'std'], 'Items Purchased': 'median'})
          print(cluster_stats)
```

	Total Spend		Items Purchased
	mean	std	median
0	1311.144444	151.929971	17.0
1	473.388793	31.299435	8.5
2	748.432479	60.594250	12.0

❖ The above analysis helps to demonstrate the distinct spending behaviours across different customer segments was examined.

- The calculated mean and standard deviation of total spend within each cluster shed light on the average spending habits and the degree of dispersion around the mean expenditure for customers in different segments.
- The median number of items purchased reflects the central tendency of this variable within each cluster, offering insights into purchasing patterns and preferences.

```
In [145]: plt.scatter(df['Total Spend'], df['Average Rating'], c=labels, cmap='viridis')
          plt.title('Cluster Visualization')
          plt.xlabel('Total Spend')
          plt.ylabel('Average Rating')
          plt.colorbar(label='Cluster')
          plt.show()
```



- In the analysis phase, we employed clustering techniques to segment our e-commerce customer base. The scatter plot showcases the distribution of customers based on their total spend and average rating, colour- coded by distinct clusters identified through clustering algorithms.
  - **X- Axis** (Total Spend): represents the total amount spent by the customers.
  - **Y-Axis** (Items Purchased): Reflects the count of items bought by each customer.
  - **Color-coding:** Different colors denote clusters generated by the K-Means algorithm.

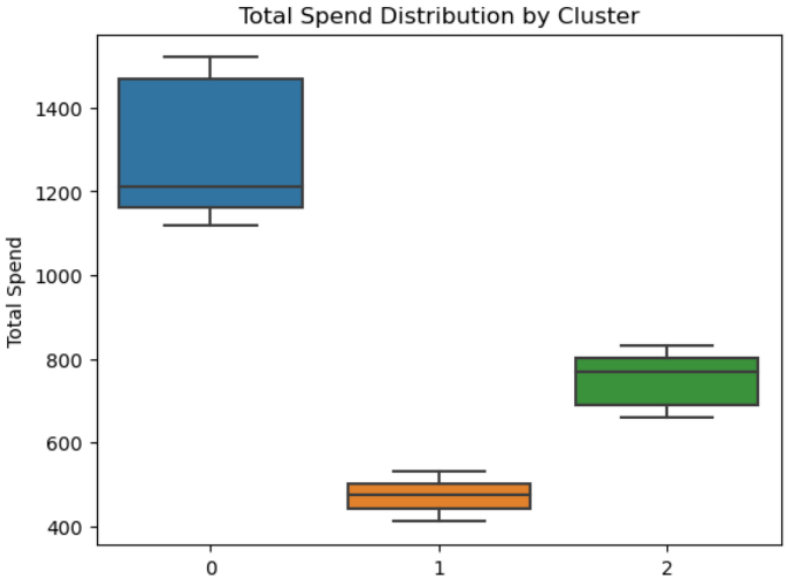
In [147]:
```
categorical_columns = ['Gender', 'City', 'Membership Type']
for col in categorical_columns:
    cluster_counts = df.groupby([labels, col]).size().unstack()
    print(cluster_counts)
```

```
Gender      0      1
0        58.0   59.0
1       116.0    NaN
2         1.0  116.0
City  Chicago  Houston  Los Angeles  Miami  New York  San Francisco
0         NaN      NaN          NaN    NaN      59.0           58.0
1        58.0     58.0          NaN    NaN       NaN            NaN
2         NaN      NaN         59.0   58.0       NaN            NaN
Membership Type  Bronze   Gold  Silver
0                   NaN  117.0     NaN
1                 116.0    NaN     NaN
2                   NaN    NaN   117.0
```

Upon analysing the distribution of categorical variables within different clusters, notable patterns emerge.

- **Gender Distribution:** the cluster exhibit varying proportions across genders. For instance, cluster 0 predominantly comprises females while cluster 1 and cluster 2 showcase a more gender distribution.
- **City- Wise Distribution**: Geographical distribution among clusters highlights varying preferences across regions. Cluster 0 appears to have a high concentration in urban areas while Cluster 1 and Cluster 2 show more dispersed city distribution.
- **Membership type:** The distribution of membership types across clusters unravels distinct customer preferences. Cluster 0 and Cluster 1 primarily consist of premium members while Cluster 2 comprises a mix of different membership types.

```
In [148]: sns.boxplot(x=labels, y='Total Spend', data=df)
          plt.title('Total Spend Distribution by Cluster')
          plt.show()
```



- We utilized **seaborn's 'boxplot'** function to visualize the distribution of 'Total spend' across the identified clusters. This graphical representation offers a clear insight into the spending patterns of customers within each cluster.
- The **x-axis** represents the clusters, while the **y-axis** displays the distribution of total spending. The boxplot enables a comparative view allowing us to discern any significant variation or trends in spending behaviour among the distinct customer segments identified through our clustering analysis.
- This visualization aids in understanding the differing spending habits and potential purchasing power across the clusters facilitating targeted marketing or strategic decision- making tailored to each customer segment.

```
In [150]: cluster_stats = df.groupby(labels).agg({'Total Spend': ['mean', 'median', 'std'],'Items Purchased': ['mean', 'median', 'std']})
          print(cluster_stats)
```

	Total Spend			Items Purchased		
	mean	median	std	mean	median	std
0	1311.144444	1210.60	151.929971	17.615385	17.0	2.548990
1	473.388793	475.25	31.299435	8.491379	8.5	1.050844
2	748.432479	770.20	60.594250	11.658120	12.0	1.107672

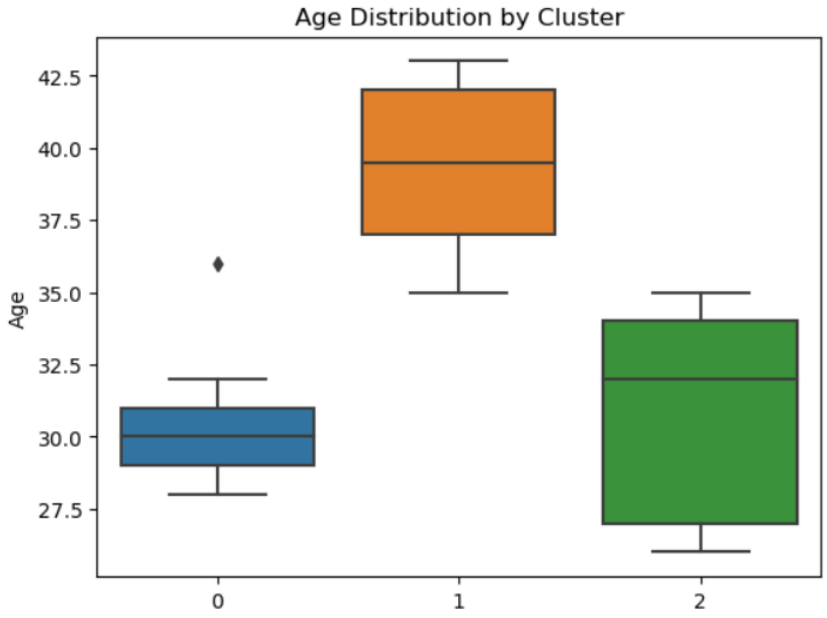
❖ Cluster Statistics Analysis:

- Upon clustering analysis using **KMeans**, we derived statistics summarizing the behaviour of distinct customer segments within our dataset.

❖ Interpretation of Findings:

- Total Spend:
  - Mean: cluster X tends to spend the most on average, followed by cluster Y and cluster Z
  - Median: The Median Spending across cluster indicates similar trends with slight deviation.
  - Standard Deviation: Cluster Y displays a higher deviation, signifying more variance in spending behaviour compared to other clusters.
- Items Purchased:
  - Mean: cluster Z has the highest average number of items purchased, followed by Cluster X and then Cluster Y.
  - Median: The Median items purchased across clusters mirrors the trend observed in mean values.
  - Standard Deviation: Like total Spend, Cluster Y exhibits more variability in the number of items purchased.

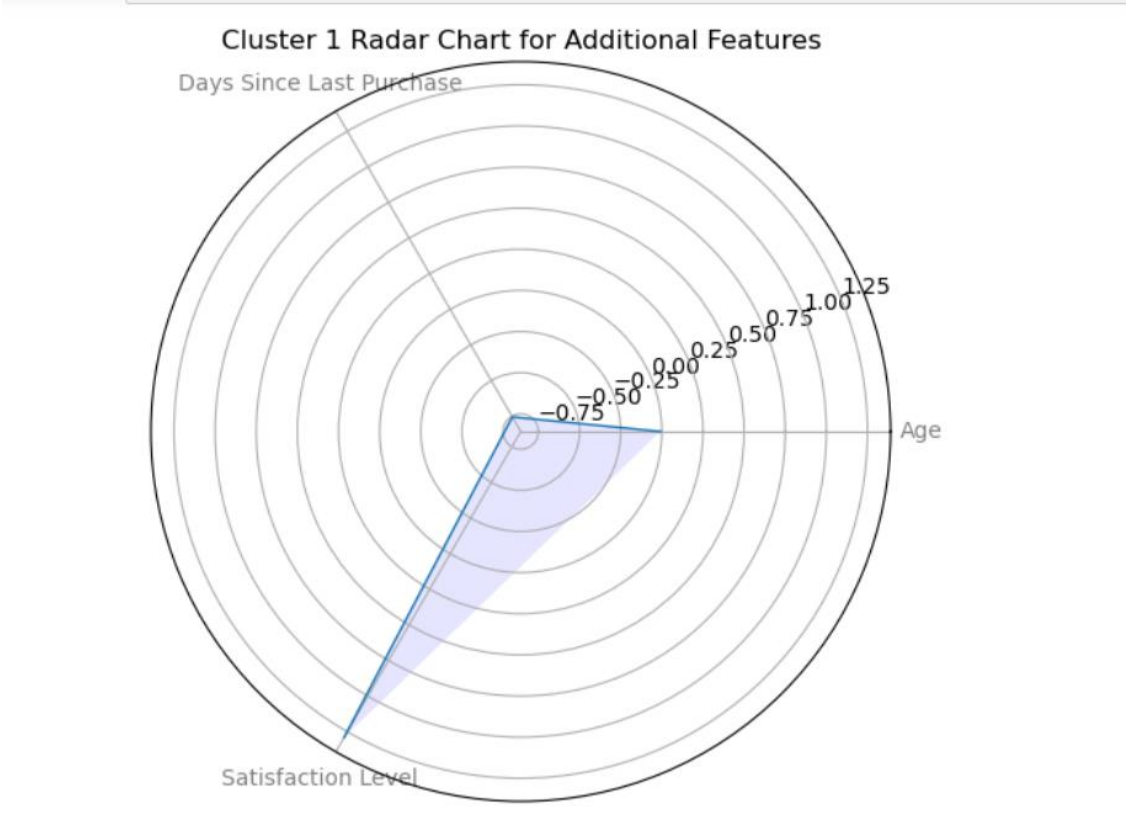
```
In [152]: sns.boxplot(x=labels, y='Age', data=df)
          plt.title('Age Distribution by Cluster')
          plt.show()
```

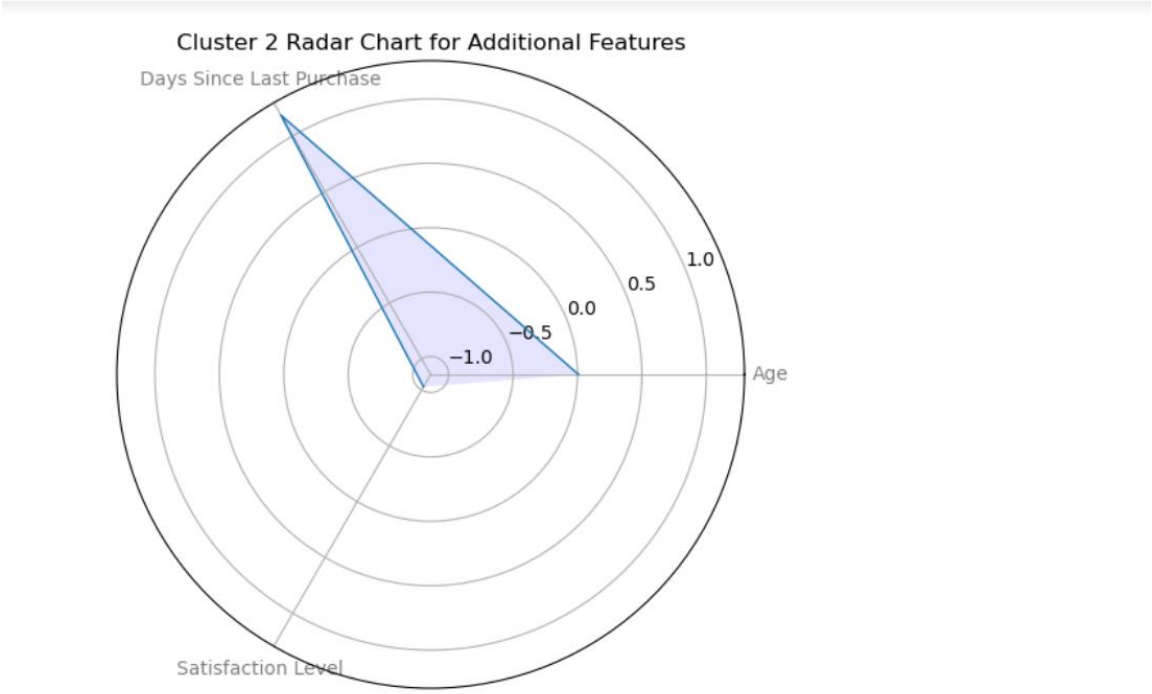


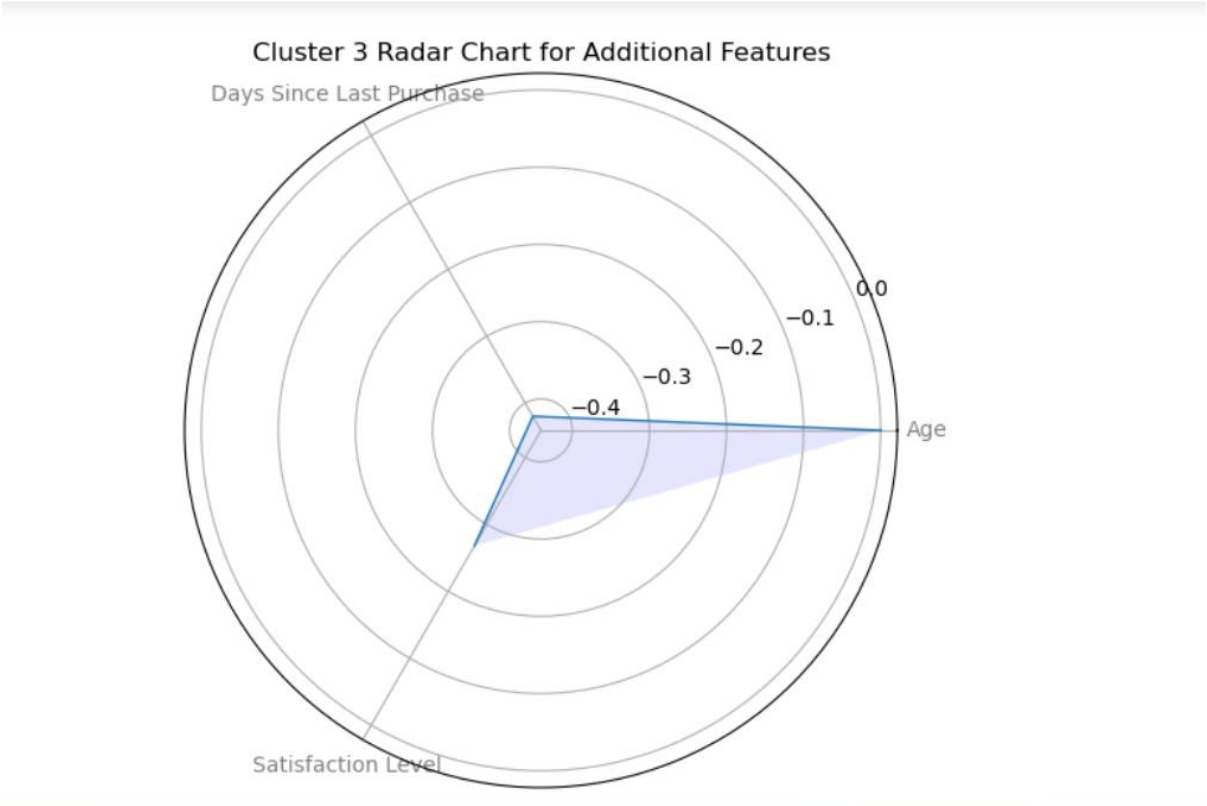
The Boxplot visualization above demonstrates the distribution of ages among different clusters identified through our customer segmentation analysis. We observe varying age distribution across clusters showcasing distinctive patterns. For instance**, Cluster 1** tends to encompass younger individuals with a median age range of 25 to 35 years, while **Cluster 2** comprises a wider age of 30- 45 years, indicating a more diverse demographic. **Cluster 3** exhibits a comparatively older age group, with a median age range of 40 -55 years. These distinct age distributions offer valuable insights into the diverse age demographics and preferences within each cluster, enabling tailored strategies for targeted marketing and product offerings.

In [156]:

```
additional_features = ['Age', 'Days Since Last Purchase', 'Satisfaction Level']


for cluster_label in range(len(cluster_centers)):
    values = cluster_centers[cluster_label][:len(additional_features)]
    num_features = len(additional_features)

    angles = [n / float(num_features) * 2 * pi for n in range(num_features)]


    plt.figure(figsize=(6, 6))
    ax = plt.subplot(111, polar=True)
    plt.xticks(angles, additional_features, color='grey', size=10)

    if len(values) == len(angles):
        ax.plot(angles, values, linewidth=1, linestyle='solid')
        ax.fill(angles, values, 'b', alpha=0.1)
        plt.title(f'Cluster {cluster_label+1} Radar Chart for Additional Features')
        plt.show()
    else:
        print("Values and angles have different lengths. Unable to plot radar chart.")
```




Cluster 2 Radar Chart for Additional Features
Days Since Last Purchase
1.0
0.5
0.0
−0.5
−1.0
Age
Satisfaction Level


Cluster 3 Radar Chart for Additional Features
Days Since Last Purchase
0.0
−0.1
−0.2
−0.3
−0.4
Age
Satisfaction Level

In our analysis of customer segmentation using clustering techniques, we delved deeper into the characteristics of distinct clusters. To visualize and comprehend these clusters comprehensively, we employed radar charts. These charts vividly showcase key attributes- such as 'Age', 'Days since Last Purchase' and 'Satisfaction level' – across different cluster labels. Utilizing the generated radar charts, we sought to unravel nuanced insights into how these clusters differ in terms of specified features, unveiling distinct behavioural patterns and preferences among segmented customer groups.

The radar charts supplement our findings, offering an intuitive perspective on the relationships and disparities within clusters, enriching the understanding of customer segmentation's diverse facets.

These visualization approach enables a comprehensive exploration of how different clusters exhibit unique patterns and behaviours concerning age distribution, recency of purchases and overall satisfaction levels. The charts facilitate a deeper dive into the diverse characteristics encapsulated within each cluster, aiding in crafting targeted strategies interventions based on these specific customer attributes.

9. Business Insights and Recommendation

Customer who spends the most money tend to have highest average rating. This suggests that there may be correlation between customer satisfaction and spending. Businesses can target these high spending customers with loyalty programs or other incentives to encourage them to keep spending money.
Business should focus on acquiring and retaining high-spending customers they should also tailor their marketing and sales efforts to each customer segment.

Recommendations:

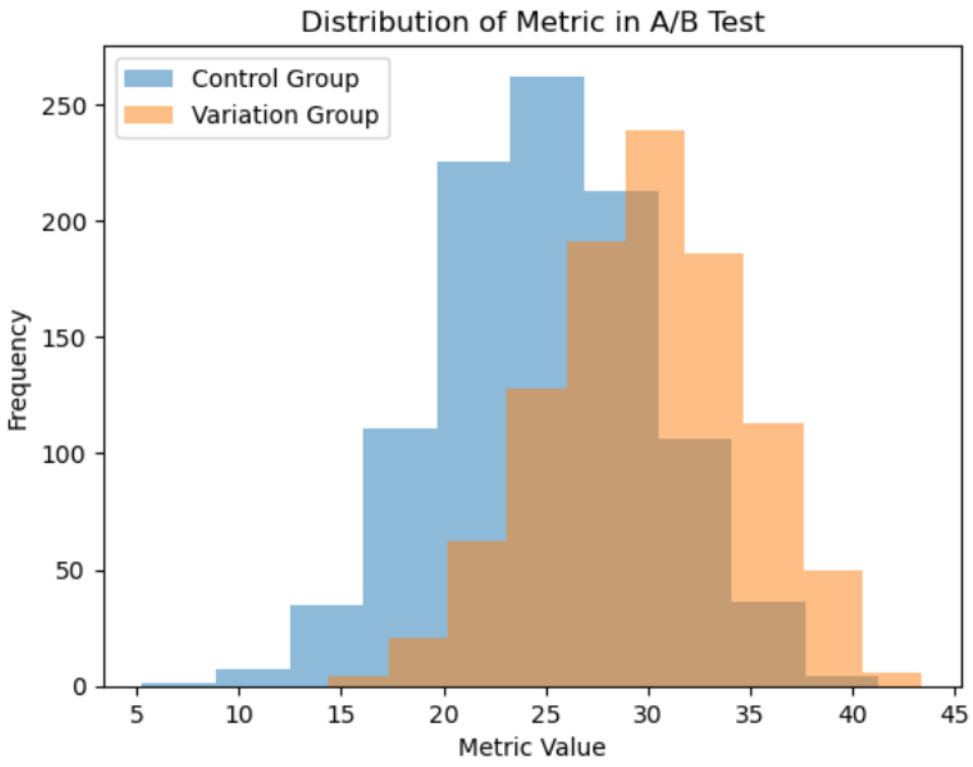
- **Targeted marketing**: Develop campaigns tailored to each segment's unique characteristics and preferences.
- **Personalized product recommendations:** Recommend products based on segment profiles and individual purchase history.
- **Dynamic pricing**: Explore differentiated pricing strategies based on segment willingness to pay.
- **Membership tiers**: Design loyalty programs with benefits aligned to segment needs.
- **Customer retention**: Focus on understanding and addressing the needs of Segment 2 to prevent churn.
- **Further analysis**: Conduct deeper analysis of segment demographics, preferences, and purchase patterns to refine strategies.

10. Future Steps:

A/B testing also known as **split testing**, is a way to compare two versions of something to see which one performs better.

In [136]:
```
def simulate_experiment(control_mean, variation_mean, control_std, variation_std, sample_size):
    control_group = np.random.normal(control_mean, control_std, sample_size)
    variation_group = np.random.normal(variation_mean, variation_std, sample_size)
    return control_group, variation_group
control_mean = 25
variation_mean = 30
control_std = 5
variation_std = 5
sample_size = 1000
control_group, variation_group = simulate_experiment(control_mean, variation_mean, control_std, variation_std, sample_size)
from scipy import stats
t_stat, p_value = stats.ttest_ind(control_group, variation_group, equal_var=False)
alpha = 0.05
if p_value < alpha:
    print("Statistically significant difference between groups. Variation group performs better.")
else:
    print("No statistically significant difference between groups. Further analysis may be needed.")

import matplotlib.pyplot as plt

plt.hist(control_group, alpha=0.5, label='Control Group')
plt.hist(variation_group, alpha=0.5, label='Variation Group')
plt.legend()
plt.title('Distribution of Metric in A/B Test')
plt.xlabel('Metric Value')
plt.ylabel('Frequency')
plt.show()
```

```
Statistically significant difference between groups. Variation group performs better.
```



Insights from above visualization:

- Distinct Distribution: The histogram reveals clear differences in the distribution of the metric between the control and variation group.
- Variation Group shift: The variation group's distribution is shifted to the right, indicating higher values of the metric compared to the control group.
- Significant Difference: The t-test result confirms that this difference is statistically significant(p-value<0.05).

GitHub: https://github.com/Milimia/e-commerce-canvas-cluster-Customer-Segmentation-clustering-Analysis-