BANA 200A Group Project Report

Introduction

In this case, our objective is to identify useful customer segments in the reservations data by using Python, and to visualize the segments in a way that could lead to actionable marketing insights.

Step 1

We need to identify customer segments in the data by using the K-Means Clustering method. We assume there are 5 centroids.

In [ ]:
```
# Import Module

import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the data into DataFrame
df = pd.read_csv('Clustering Data.csv')

df.head()
```

Out[ ]:

	uid	PNRLocatorID	avg_amt	round_trip
0	5045544552244696420493F7C2067657420746869732072...	AADMLF	0.019524	0
1	46495853454E44696420493F7C20676574207468697320...	AAFBOM	0.081774	1
2	534355545444696420493F7C2067657420746869732072...	AAFILI	0.026650	0
3	534355545444696420493F7C2067657420746869732072...	AAFILI	0.026650	0
4	44554D4D414E4E44696420493F7C206765742074686973...	AAFRQI	0.000000	1

5 rows × 90 columns

Now we can perform the K-means clustering on the data and we assume 5 centroids.

In [ ]:
```
k = 5 # Number of clusters

# Create an instance of the KMeans class and assign it to variable kmeans
kmeans = KMeans(n_clusters = k)

# Applying the fit method to create a KMeans model using the data
df2 = df.drop(['uid','PNRLocatorID'], axis=1)
kmeans.fit(df2)
```

```
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/
sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of `n_init`
will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to
suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/
threadpoolctl.py:1019: RuntimeWarning: libc not found. The ctypes module in Py
thon 3.9 is maybe too old for this OS.
  warnings.warn(
```

Out[ ]:

```
▾ KMeans
KMeans(n_clusters=5)
```

In [ ]:

```
labels = kmeans.labels_

df['Cluster'] = labels
```

Step 2

Now we extract the cluster "Assignments" from the cluster results data set and add it to the reservations data.

In [ ]:

```
# Load the reservation data into a DataFrame
customer_data = pd.read_csv('sample_data_transformed.csv')
clustered_data = df[['uid','Cluster']].copy()
final_dataframe = customer_data.merge(clustered_data[['uid', 'Cluster']], on='u
```

```
/var/folders/fb/lrdyfv897kxbzkt4pt59mkmw0000gn/T/ipykernel_28391/3819511702.p
y:2: DtypeWarning: Columns (13) have mixed types. Specify dtype option on impo
rt or set low_memory=False.
  customer_data = pd.read_csv('sample_data_transformed.csv')
```

Step 3

Now we are trying to visualize the segment data by using Python.

In [ ]:

```
final_dataframe.head()
```

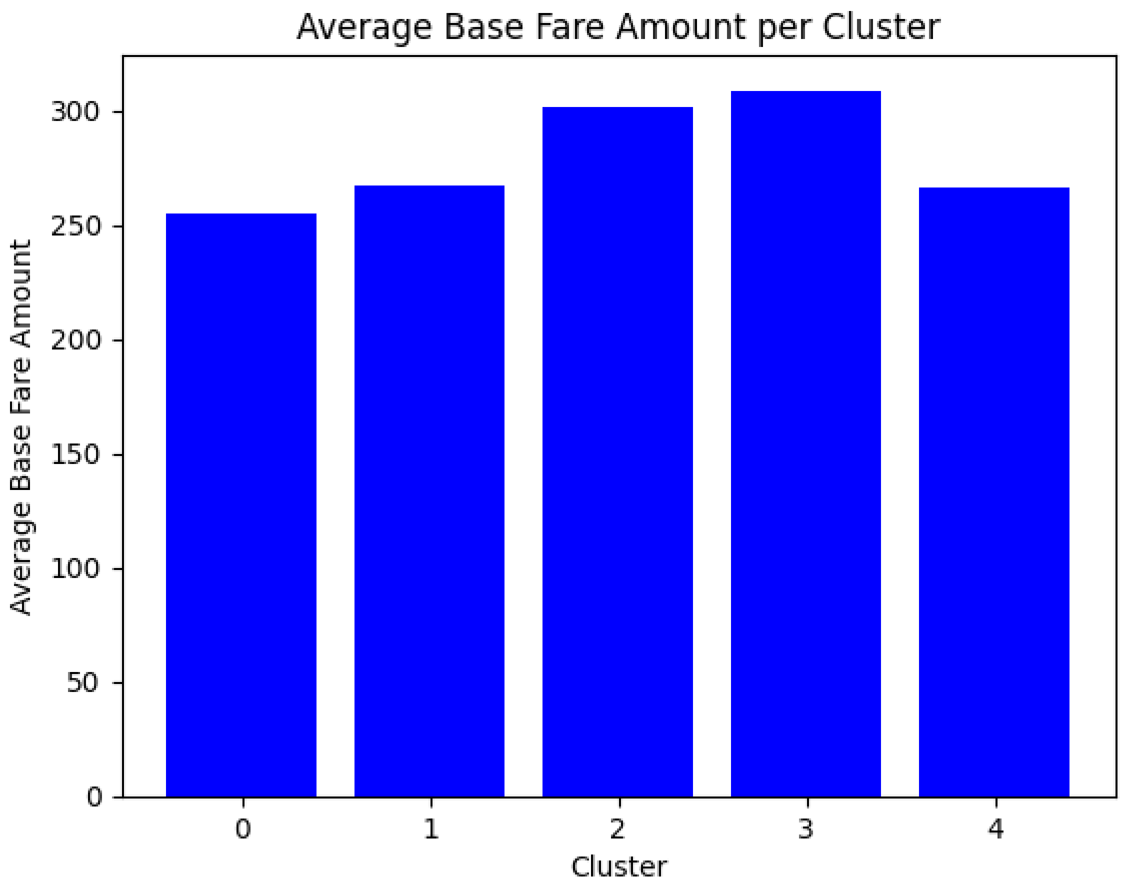
Out[ ]:

	Unnamed: 0	PNRLocatorID	PaxName	TicketNum	CouponSeqNbr	ServiceStartCity	Service
0	1	AADMLF	PETEJO	3.377490e+12	1	MSP	
1	2	AAFBOM	FIXSMO	3.372110e+12	2	JFK	
2	3	AAFBOM	FIXSMO	3.372110e+12	1	MSP	
3	4	AAFILI	SCUTKA	3.372110e+12	2	MSP	
4	5	AAFILI	SCUTKA	3.372110e+12	1	LAN	

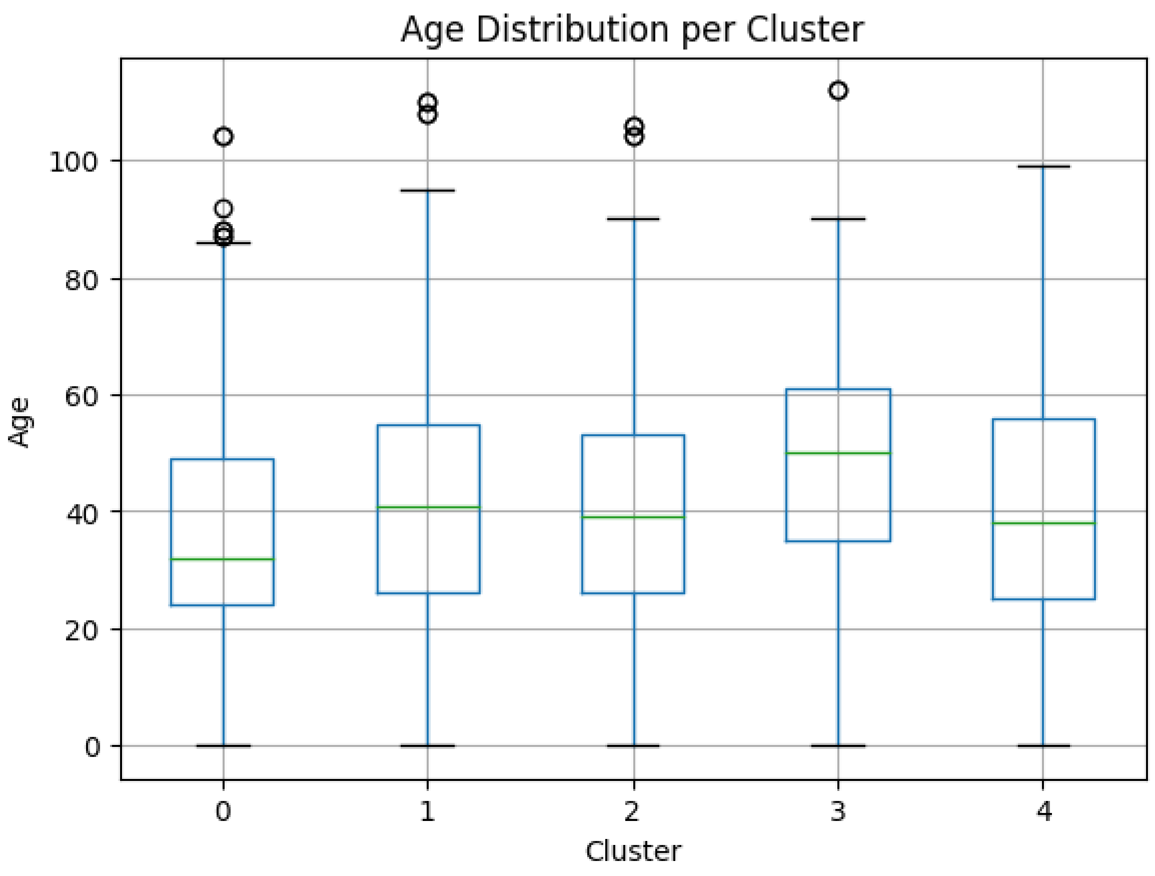
5 rows × 38 columns

In [ ]:
```
# Calculate average base fare for each cluster
avg_base_fare = final_dataframe.groupby('Cluster')['BaseFareAmt'].mean()

# Create bar chart
plt.bar(avg_base_fare.index, avg_base_fare.values, color='blue')
plt.xlabel('Cluster')
plt.ylabel('Average Base Fare Amount')
plt.title('Average Base Fare Amount per Cluster')
plt.show()
```

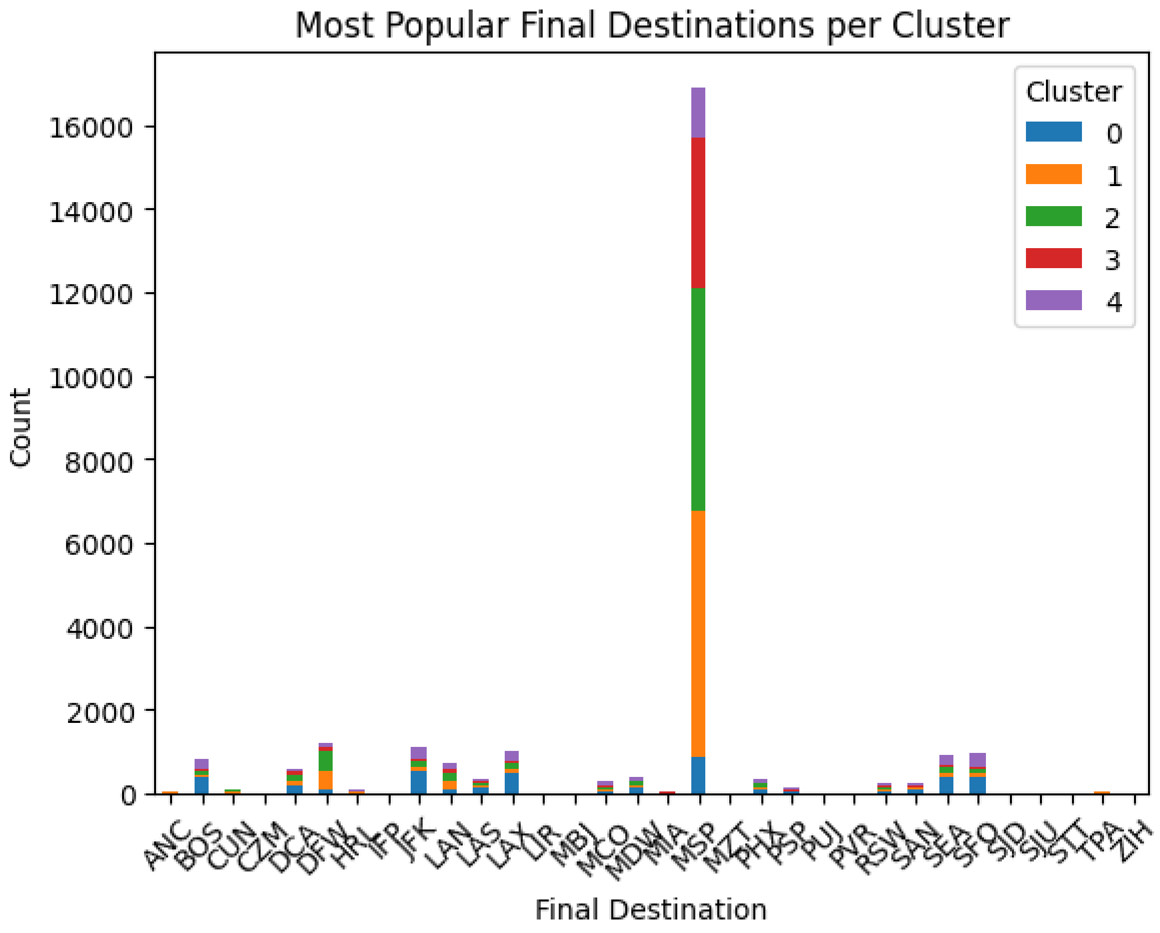


In [ ]:
```
# Create box plot for Age distribution per cluster
final_dataframe.boxplot(column='Age', by='Cluster')
plt.xlabel('Cluster')
plt.ylabel('Age')
plt.title('Age Distribution per Cluster')
plt.suptitle('')  # Suppress the automatic title
plt.show()
```



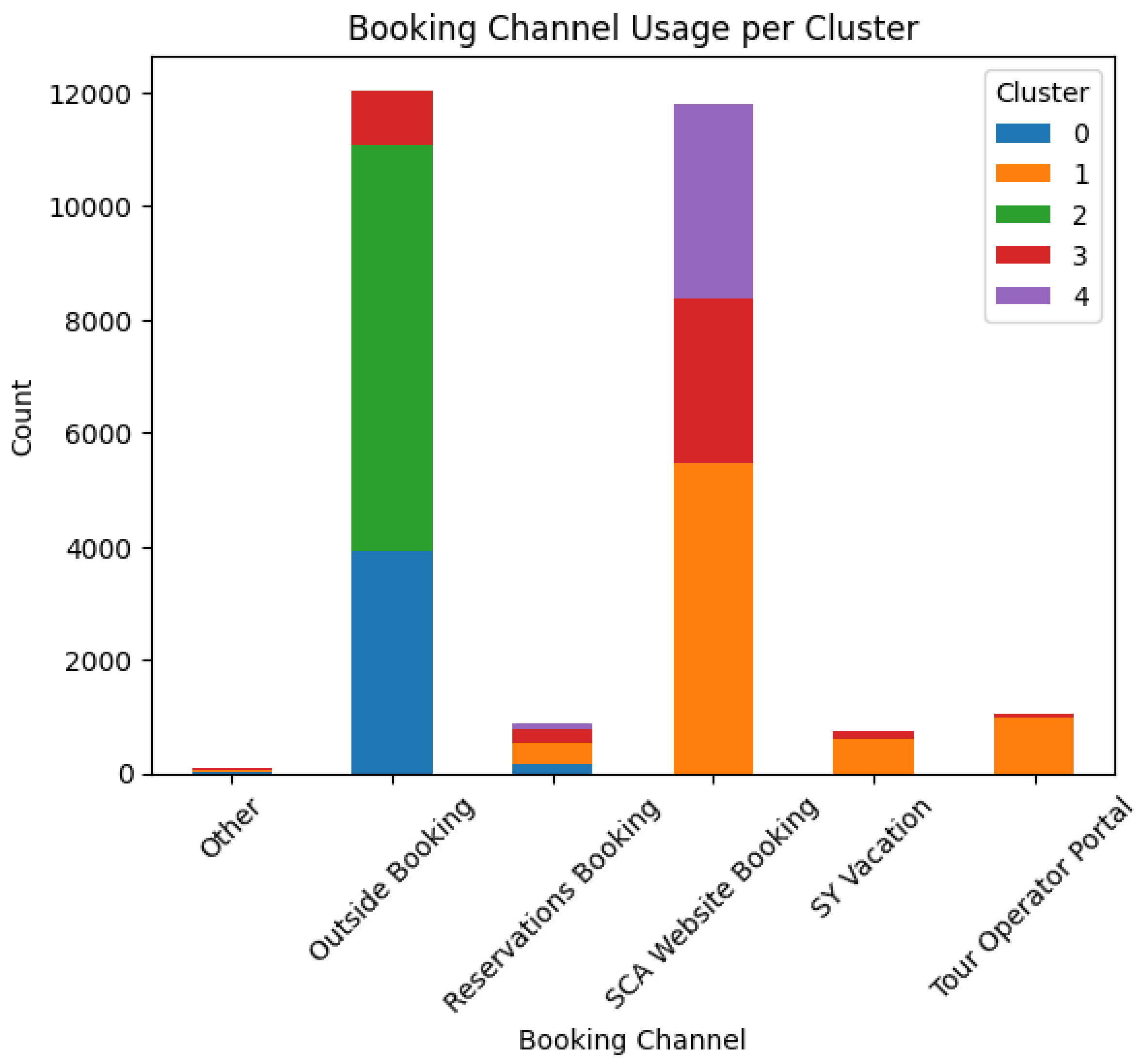
In [ ]:
```
# Count the number of occurrences for each final destination and cluster
destination_count = final_dataframe.groupby(['final_destination', 'Cluster']).s

# Create bar chart
destination_count.plot(kind='bar', stacked=True)
plt.xlabel('Final Destination')
plt.ylabel('Count')
plt.title('Most Popular Final Destinations per Cluster')
plt.xticks(rotation=45)
plt.show()
```



```
In [ ]: # Count the number of occurrences for each booking channel and cluster
        channel_count = final_dataframe.groupby(['BookingChannel', 'Cluster']).size().u

        # Create bar chart
        channel_count.plot(kind='bar', stacked=True)
        plt.xlabel('Booking Channel')
        plt.ylabel('Count')
        plt.title('Booking Channel Usage per Cluster')
        plt.xticks(rotation=45)
        plt.show()
```


Booking Channel Usage per Cluster
Count
0
2000
4000
6000
8000
10000
12000
Booking Channel
Other
Outside Booking
Reservations Booking
SCA Website Booking
SY Vacation
Tour Operator Portal
Cluster
0
1
2
3
4