```
##1. LOADING DATASET
#importing pandas
import pandas as pd

#Reading the clustering data file
clustering_data = pd.read_csv('C:/Users/shyam/Desktop/UCI/Courses/Quarter 1/Python/Project/Clustering Data (1).csv')

#Checking the shape of the imported dataframe
print(f'The dataset has {clustering_data.shape[0]} rows and {clustering_data.shape[1]} columns')
clustering_data.head()
```

```
The dataset has 15144 rows and 90 columns
```

	uid	PNRLocatorID	avg_amt	round_trip	group_size	group	days_pre_booked	BookingCha
0	5045544552446964204 93F7C2067657420746869732072...	AADMLF	0.019524	0	0.000	0	0.029703	
1	46495853454E44696420493F7C20676574207468697320...	AAFBOM	0.081774	1	0.000	0	0.039604	
2	5343555454444696420493F7C2067657420746869732072...	AAFILI	0.026650	0	0.125	1	0.069307	
3	5343555454444696420493F7C2067657420746869732072...	AAFILI	0.026650	0	0.125	1	0.069307	
4	44554D4D414E4E44696420493F7C206765742074686973...	AAFRQI	0.000000	1	0.000	0	0.035361	

5 rows × 90 columns

```
##2. DATA PREPROCESSING
#Preparing the data for clustering by copy the previously created dataset
prep_clustering_data = clustering_data.copy()

# Saving the uid column before dropping it
save_uid_column = prep_clustering_data['uid']

#Dropping the uid and PNRLocaterID fields since they have string values
prep_clustering_data = prep_clustering_data.drop(['PNRLocatorID', 'uid'], axis=1)

#Checking the shape of the prepared dataframe
print(f'The dataset has {prep_clustering_data.shape[0]} rows and {prep_clustering_data.shape[1]} columns')
prep_clustering_data.head(10)
```

```
The dataset has 15144 rows and 88 columns
```

	avg_amt	round_trip	group_size	group	days_pre_booked	BookingChannel_Other	BookingChannel_Outside_Booking	BookingChannel_Reserva
0	0.019524	0	0.000	0	0.029703	0	1	
1	0.081774	1	0.000	0	0.039604	0	0	
2	0.026650	0	0.125	1	0.069307	0	0	
3	0.026650	0	0.125	1	0.069307	0	0	
4	0.000000	1	0.000	0	0.035361	0	1	
5	0.000000	1	0.125	1	0.050919	0	1	
6	0.000000	1	0.125	1	0.050919	0	1	
7	0.074727	1	0.000	0	0.045262	0	0	
8	0.035414	1	0.000	0	0.082037	0	1	
9	0.035414	1	0.125	1	0.018388	0	1	

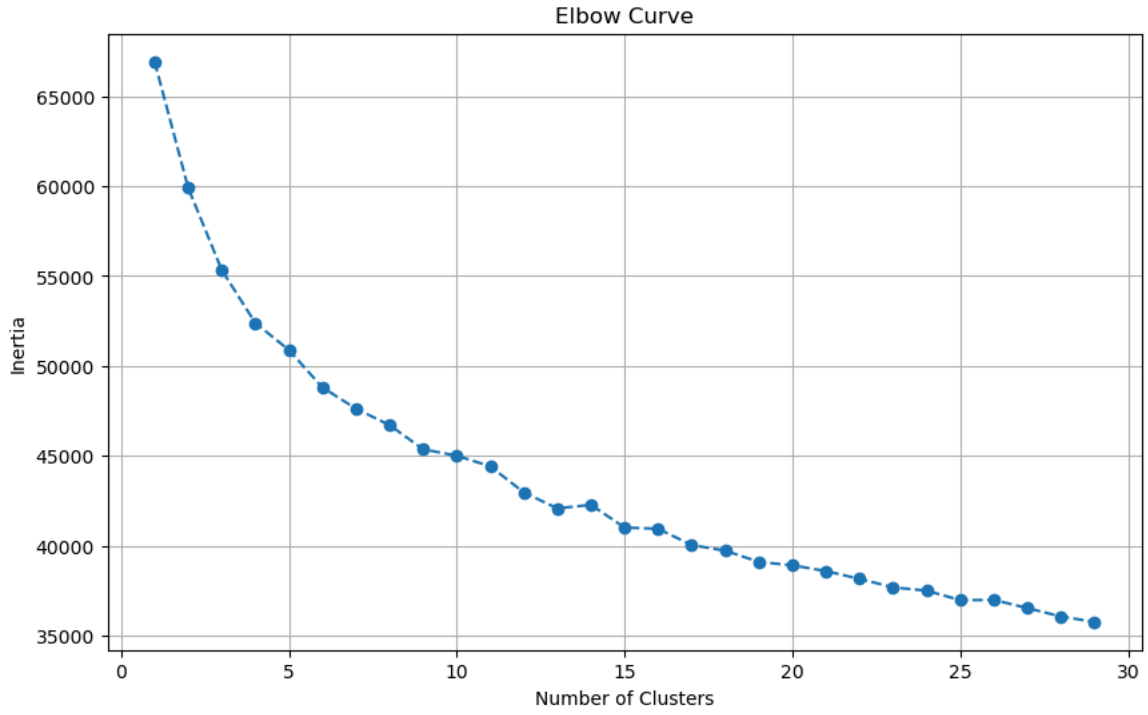
10 rows × 88 columns

```
##3. CLUSTERING
#3.1 Looking at the optimum number of clusters
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

inertia = []                  # sum of squared distances between each data point and its closest cluster centroid (wcss)
cluster_range = range(1, 30)


for cluster_num in cluster_range:
```

```
    print(f'Iteration Number: {cluster_num}')

    kmeans = KMeans(n_clusters=cluster_num, n_init=10)
    kmeans.fit(prep_clustering_data)
    inertia.append(kmeans.inertia_)
```

```
plt.figure(figsize=(10,6))
plt.plot(cluster_range, inertia, marker='o', linestyle='--')
plt.title('Elbow Curve')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.grid(True)
plt.show()
```

```
Iteration Number: 1
Iteration Number: 2
Iteration Number: 3
Iteration Number: 4
Iteration Number: 5
Iteration Number: 6
Iteration Number: 7
Iteration Number: 8
Iteration Number: 9
Iteration Number: 10
Iteration Number: 11
Iteration Number: 12
Iteration Number: 13
Iteration Number: 14
Iteration Number: 15
Iteration Number: 16
Iteration Number: 17
Iteration Number: 18
Iteration Number: 19
Iteration Number: 20
Iteration Number: 21
Iteration Number: 22
Iteration Number: 23
Iteration Number: 24
Iteration Number: 25
Iteration Number: 26
Iteration Number: 27
Iteration Number: 28
Iteration Number: 29
```



```
#3.2 Applying K-Means, and creating 5 centroids as per the Project requirement

kmeans = KMeans(n_clusters=5, n_init=10)
kmeans.fit(prep_clustering_data)

# attach the dropped UID column and cluster
prep_clustering_data = pd.DataFrame({
    'uid': save_uid_column,
    'cluster': kmeans.labels_
})

print(f'The dataset has {prep_clustering_data.shape[0]} rows and {prep_clustering_data.shape[1]} columns')
prep_clustering_data.head()
```

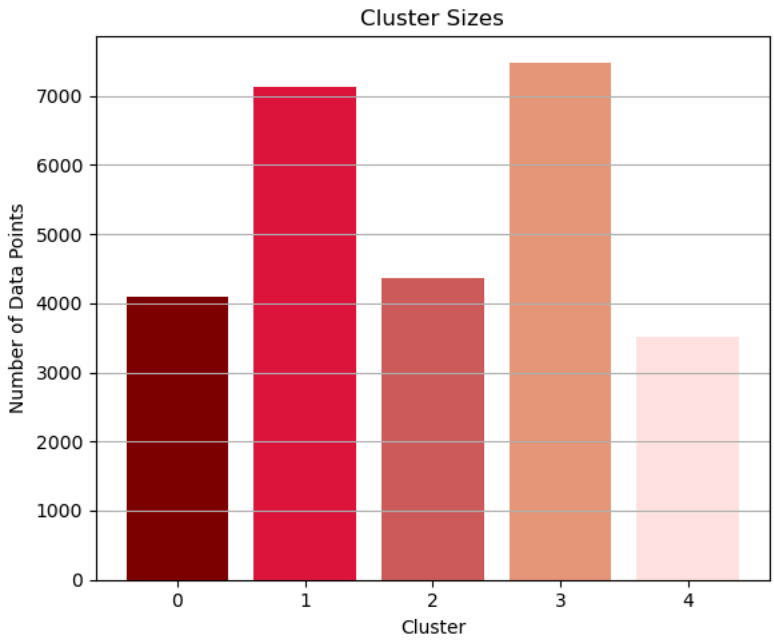
```
The dataset has 15144 rows and 2 columns
```

	uid	cluster
0	5045544552446964204 93F7C2067657420746869732072...	1
1	46495853454E44696420493F7C2067657420746869732 0...	1
2	5343555454446964204 93F7C2067657420746869732072...	4
3	5343555454446964204 93F7C2067657420746869732072...	4
4	44554D4D414E4E44696420493F7C206765742074686973...	2

```
##4. MERGED DATASET FOR VISUALIZATION
# Merging the Clustered dataset with the provided sample_data_transformed.csv
sample_transformed = pd.read_csv('C:/Users/shyam/Desktop/UCI/Courses/Quarter 1/Python/Project/sample_data_transformed.csv')
final_dataframe = sample_transformed.merge(prep_clustering_data[['uid', 'cluster']], on='uid', how='left')

#Checking the shape of the final dataframe and looking at the number of records in each cluster
final_dataframe.head(100)
cluster_sizes = final_dataframe['cluster'].value_counts().sort_index()
print(cluster_sizes)
```

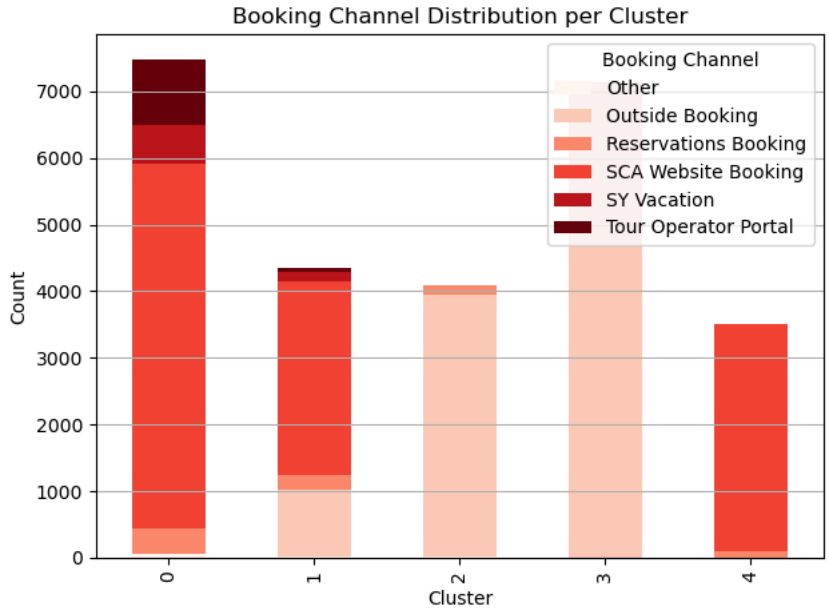
```
cluster
0    7481
1    4348
2    4096
3    7140
4    3510
Name: count, dtype: int64
C:\Users\shyam\AppData\Local\Temp\ipykernel_47360\2175281245.py:3: DtypeWarning: Columns (13) have mixed types. Specify dtype option on
  sample_transformed = pd.read_csv('C:/Users/shyam/Desktop/UCI/Courses/Quarter 1/Python/Project/sample_data_transformed.csv')
```

```
##5. ANALYSIS AND VISUALIZATION
from pandas import *
import matplotlib.pyplot as plt

cluster_colors = {
    0: 'maroon',
    1: 'crimson',
    2: 'indianred',
    3: 'darksalmon',
    4: 'mistyrose'
}

cluster_sizes = final_dataframe['cluster'].value_counts().sort_index()

plt.figure(figsize=(6, 5))


for cluster in range(len(cluster_sizes)):
    plt.bar(cluster, cluster_sizes[cluster], color=cluster_colors[cluster])

# Adding plot details
plt.title('Cluster Sizes')
plt.xlabel('Cluster')
plt.ylabel('Number of Data Points')
plt.xticks(ticks=range(len(cluster_sizes)))
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



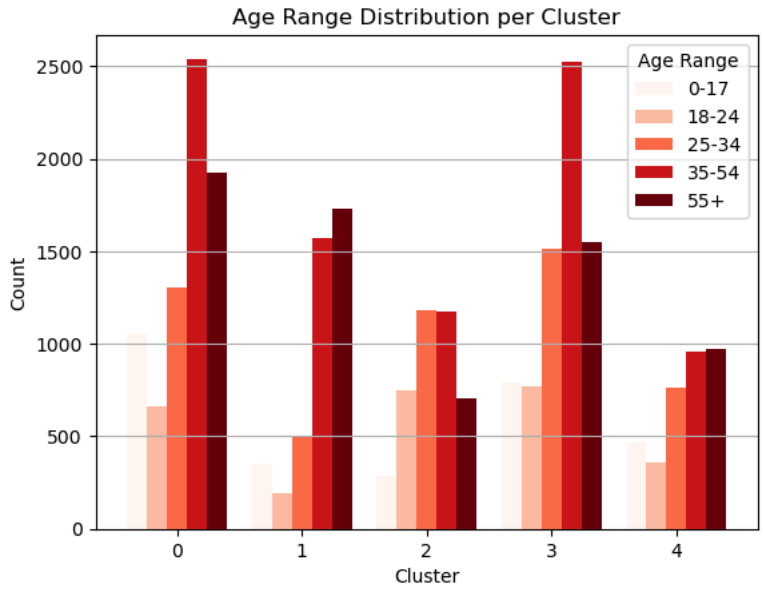

```
#Preparing the sample dataframe to be plotted
booking_channel_counts = final_dataframe.groupby(['cluster', 'BookingChannel']).size().unstack(fill_value=0)

#Printing the sample dataframe
print(booking_channel_counts)

#Creating the Bar plot
booking_channel_counts.plot(kind='bar', stacked=True, colormap='Reds')

# Adding plot details
plt.title('Booking Channel Distribution per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(ticks=range(len(cluster_sizes)))
plt.tight_layout()
plt.grid(axis='y')
plt.legend(title='Booking Channel')

#Displaying the plot
plt.show()
```

```
BookingChannel  Other  Outside Booking  Reservations Booking  \
cluster
0                  50                0                   398
1                  27              985                   228
2                  24             3919                   153
3                   0             7140                     0
4                   3                0                   100

BookingChannel  SCA Website Booking  SY Vacation  Tour Operator Portal
cluster
0                              5465          588                   980
1                              2906          142                    60
2                                 0            0                     0
3                                 0            0                     0
4                              3407            0                     0
```

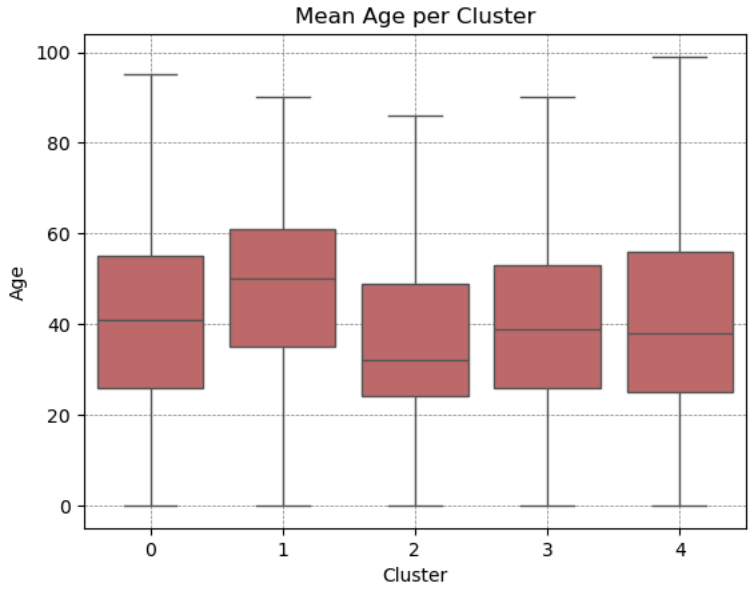



```
#Preparing the sample dataframe to be plotted
age_group_counts = final_dataframe.groupby(['cluster', 'age_group']).size().unstack()

#Printing the sample dataframe
print(age_group_counts)

#Creating the Bar plot
age_group_counts.plot(kind='bar', width=0.8, colormap='Reds')

# Adding plot details
plt.title('Age Range Distribution per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.legend(title='Age Range')

# Displaying the plot
plt.show()
```

```
age_group  0-17  18-24  25-34  35-54   55+
cluster
0          1053    659   1301   2540  1928
1           353    191    503   1572  1729
2           288    749   1184   1171   704
3           791    767   1510   2524  1548
4           466    356    763    956   969
```

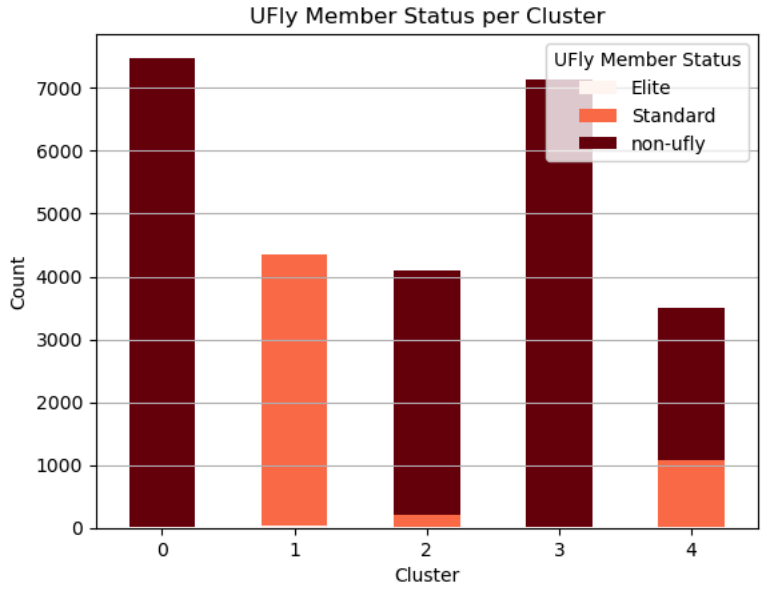



```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the boxplot without showing outliers
sns.boxplot(x='cluster', y='Age', data=final_dataframe, showfliers=False, color='indianred')

# Show grid lines
plt.grid(True)

# Customize grid appearance (optional)
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Adding plot details
plt.title('Mean Age per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Age')

# Displaying the plot
plt.show()
```



```
#Preparing the sample dataframe to be plotted
member_status_counts = final_dataframe.groupby(['cluster', 'UflyMemberStatus']).size().unstack(fill_value=0)

#Printing the sample dataframe
print(member_status_counts)

#Creating the Bar plot
member_status_counts.plot(kind='bar', stacked=True, colormap='Reds')

# Adding plot details
plt.title('UFly Member Status per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.legend(title='UFly Member Status')

# Display the plot
plt.show()
```

```
UflyMemberStatus  Elite  Standard  non-ufly
cluster
0                     6         0      7475
1                    45      4302         1
2                     8       192      3896
3                    18         0      7122
4                    21      1054      2435
```

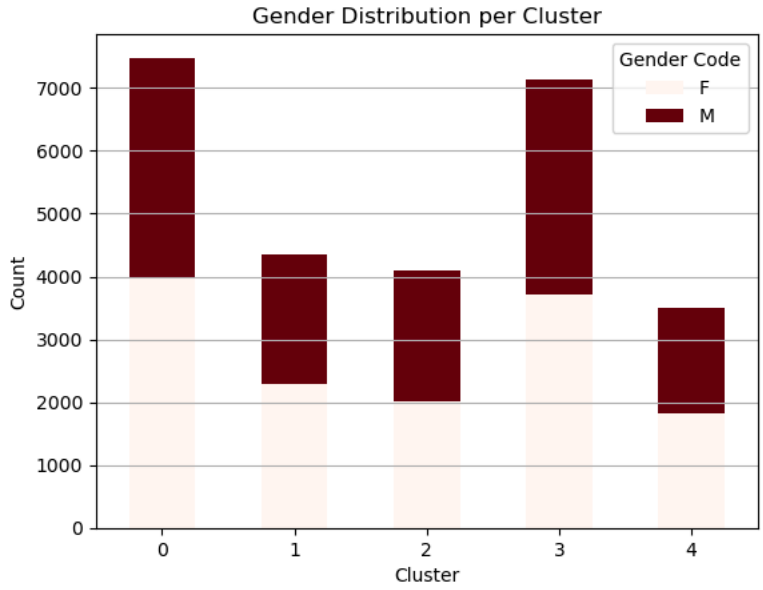



```
#Preparing the sample dataframe to be plotted
booked_class_counts = final_dataframe.groupby(['cluster', 'BkdClassOfService']).size().unstack(fill_value=0)

#Printing the sample dataframe
print(booked_class_counts)

#Creating the Bar plot
booked_class_counts.plot(kind='bar', stacked=True, colormap='Reds')

# Adding plot details
plt.title('Booked Class of Service per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.legend(title='Booked Class of Service')

# Display the plot
plt.show()
```

```
BkdClassOfService  Coach  Discount First Class  First Class
cluster
0                   7298                     0          183
1                   4059                     4          285
2                   4055                     0           41
3                   7101                     0           39
4                   3327                     1          182
```

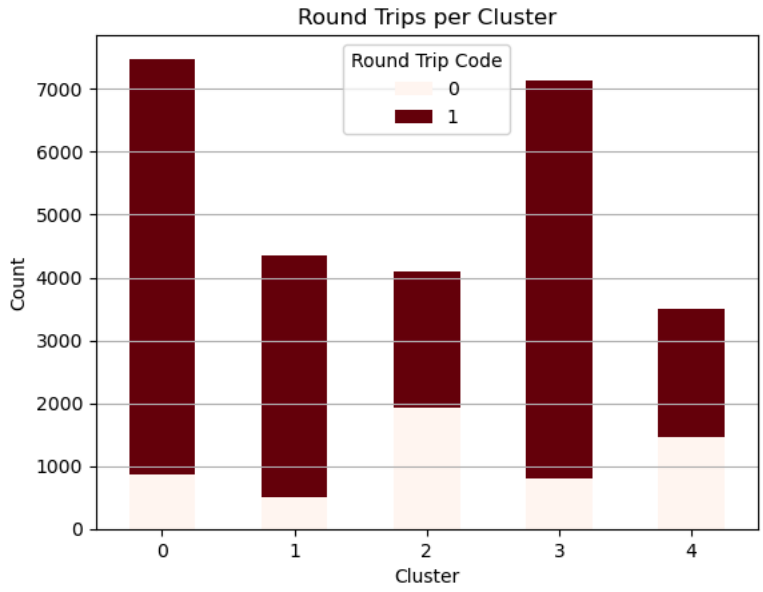

```
#Preparing the sample dataframe to be plotted
gender_code_counts = final_dataframe.groupby(['cluster', 'GenderCode']).size().unstack(fill_value=0)

#Printing the sample dataframe
print(gender_code_counts)

#Creating the Bar plot
gender_code_counts.plot(kind='bar', stacked=True, colormap='Reds')

# Adding plot details
plt.title('Gender Distribution per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.legend(title='Gender Code')

# Displaying the plot
plt.show()
```

```
GenderCode     F     M
cluster
0           3993  3488
1           2279  2069
2           2004  2092
3           3711  3429
4           1823  1687
```

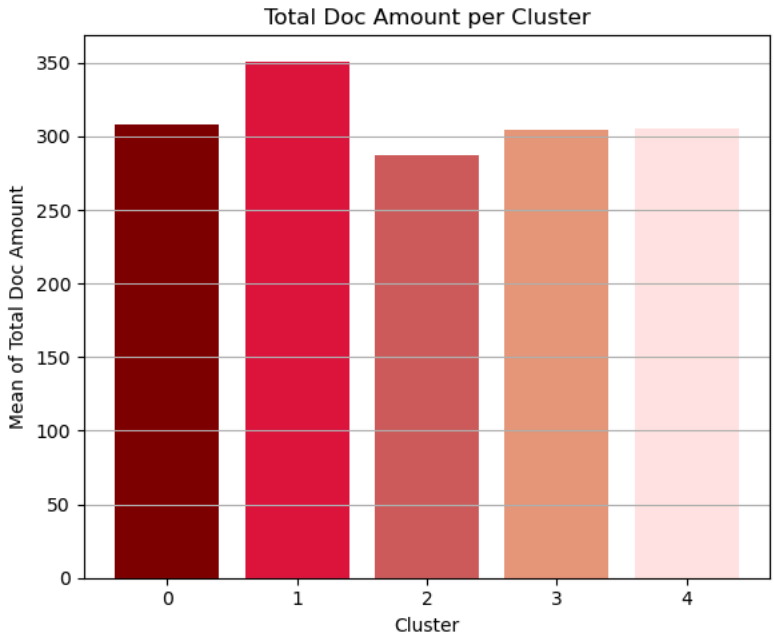



```
#Preparing the sample dataframe to be plotted
round_trip_counts = final_dataframe.groupby(['cluster', 'round_trip']).size().unstack()

#Printing the sample dataframe
print(round_trip_counts)

#Creating the Bar plot
round_trip_counts.plot(kind='bar', stacked=True, colormap='Reds')

# Adding plot details
plt.title('Round Trips per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.legend(title='Round Trip Code')

# Displaying the plot
plt.show()
```

```
round_trip     0     1
cluster
0            865  6616
1            509  3839
2           1935  2161
3            794  6346
4           1464  2046
```

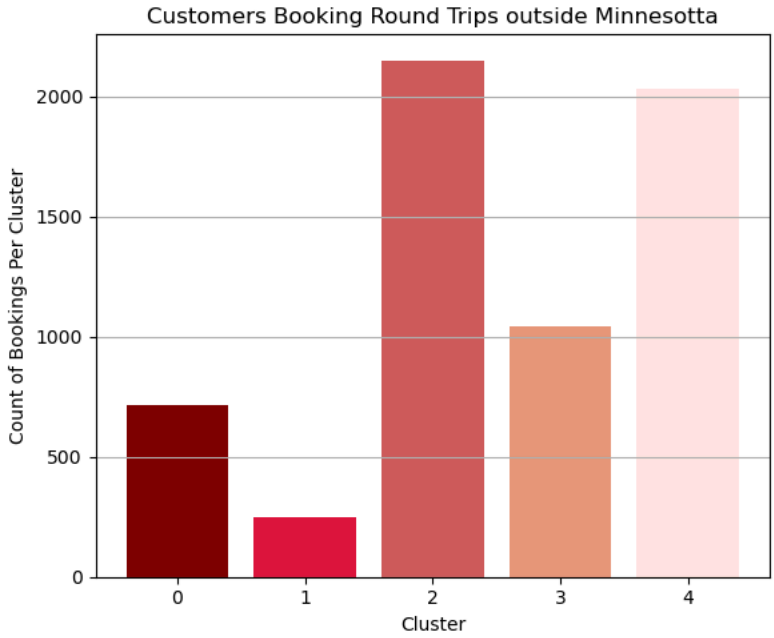



```
cluster_colors = {
    0: 'maroon',
    1: 'crimson',
    2: 'indianred',
    3: 'darksalmon',
    4: 'mistyrose'
}

#Preparing the sample dataframe to be plotted
cluster_revenue = final_dataframe.groupby('cluster')['TotalDocAmt'].mean()

#Creating the Bar plot
plt.figure(figsize=(6, 5))
for cluster in range(len(cluster_revenue)):
    plt.bar(cluster, cluster_revenue[cluster], width = 0.8, color=cluster_colors[cluster])

# Adding plot details
plt.title('Total Doc Amount per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Mean of Total Doc Amount')
plt.xticks(ticks=range(len(cluster_revenue)))
plt.grid(axis='y')
plt.tight_layout()

# Displaying the plot
plt.show()
```



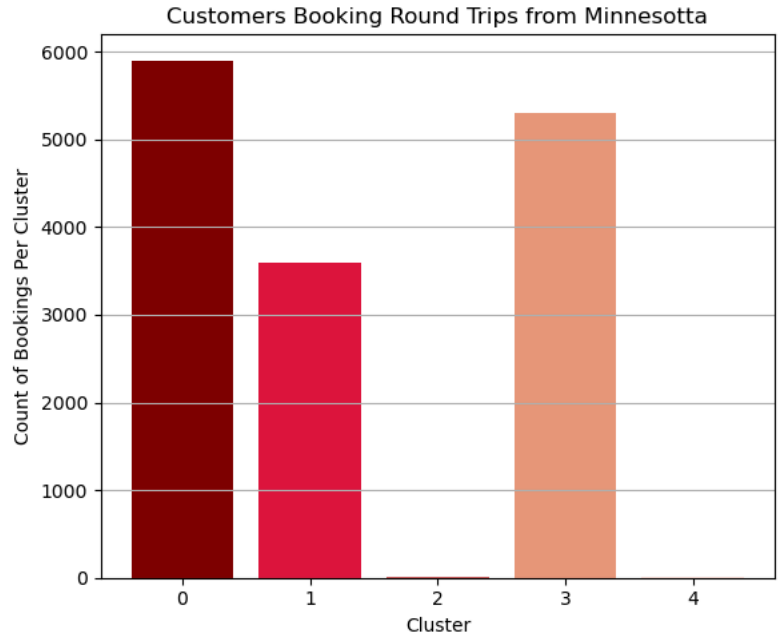

```
#Preparing the sample dataframe to be plotted
msp_origin_dest = final_dataframe.loc[(final_dataframe['round_trip']== 1) & (final_dataframe['true_origins']!= "MSP")]

print(f'The dataset has {msp_origin_dest.shape[0]} rows and {msp_origin_dest.shape[1]} columns')
cluster_sizes = msp_origin_dest['cluster'].value_counts().sort_index()

#Creating the Bar plot
plt.figure(figsize=(6, 5))
for cluster in range(len(cluster_sizes)):
    plt.bar(cluster, cluster_sizes[cluster], color=cluster_colors[cluster])

# Adding plot details
plt.title('Customers Booking Round Trips outside Minnesotta')
plt.xlabel('Cluster')
plt.ylabel('Count of Bookings Per Cluster')
plt.xticks(ticks=range(len(cluster_revenue)))
plt.grid(axis='y')
plt.tight_layout()

# Displaying the plot
plt.show()
```

```
The dataset has 6191 rows and 38 columns
```



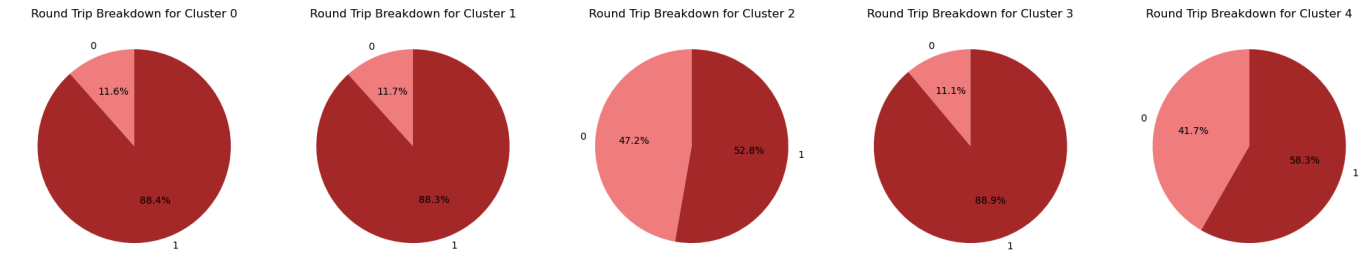
```
#Preparing the sample dataframe to be plotted
msp_origin_dest = final_dataframe.loc[(final_dataframe['round_trip']== 1) & (final_dataframe['true_origins']== "MSP")]

print(f'The dataset has {msp_origin_dest.shape[0]} rows and {msp_origin_dest.shape[1]} columns')
cluster_sizes = msp_origin_dest['cluster'].value_counts().sort_index()

#Creating the Bar plot
plt.figure(figsize=(6, 5))
for cluster in range(len(cluster_sizes)):
    plt.bar(cluster, cluster_sizes[cluster], color=cluster_colors[cluster])

# Adding plot details
plt.title('Customers Booking Round Trips from Minnesotta')
plt.xlabel('Cluster')
plt.ylabel('Count of Bookings Per Cluster')
plt.xticks(ticks=range(len(cluster_revenue)))
plt.grid(axis='y')
plt.tight_layout()

# Displaying the plot
plt.show()
```

```
The dataset has 14817 rows and 38 columns
```



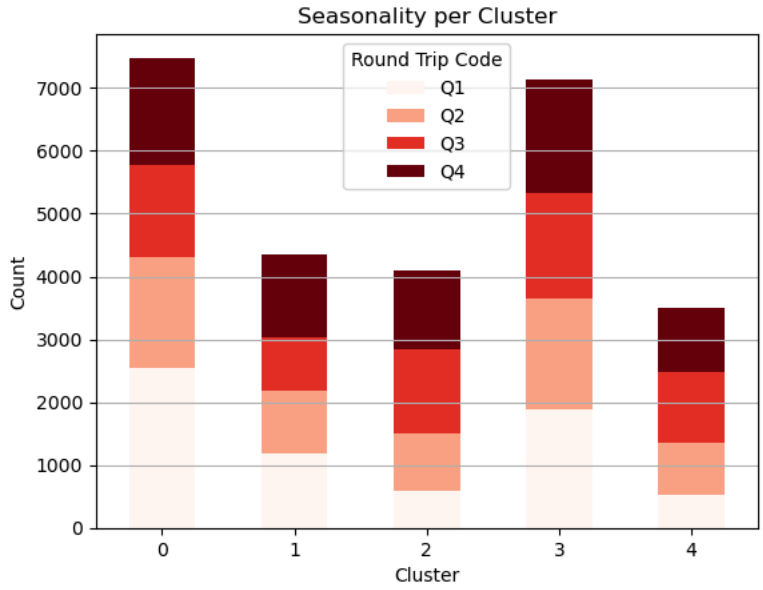

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the boxplot without showing outliers
sns.boxplot(x='cluster', y='days_pre_booked', data=final_dataframe, showfliers=False, color='indianred')

# Show grid lines
plt.grid(True)

# Customize grid appearance (optional)
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Adding plot details
plt.title('Mean of Days PreBooked per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Days PreBooked')

# Displaying the plot
plt.show()
```


```
#Preparing the sample dataframe to be plotted
round_trip_counts = final_dataframe.groupby(['cluster', 'round_trip']).size().unstack()

# Create the pie plot by iterating over each cluster
fig, axes = plt.subplots(1, 5, figsize=(20, 4))
for i, cluster in enumerate(round_trip_counts.index):
    values = round_trip_counts.loc[cluster]
    axes[i].pie(values, labels=values.index, autopct='%1.1f%%', colors=['lightcoral', 'brown'], startangle=90)
    axes[i].axis('equal')
    axes[i].set_title(f'Round Trip Breakdown for Cluster {cluster}')
plt.tight_layout()

# Displaying the plot
plt.show()
```




```
#Preparing the sample dataframe to be plotted
seasonality_counts = final_dataframe.groupby(['cluster', 'seasonality']).size().unstack()

#Printing the sample dataframe
print(seasonality_counts)

#Creating the Bar plot
seasonality_counts.plot(kind='bar',stacked=True, colormap='Reds')

# Adding plot details
plt.title('Seasonality per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.legend(title='Round Trip Code')

# Displaying the plot
plt.show()
```

```
seasonality    Q1    Q2    Q3    Q4
cluster
0            2543  1758  1468  1712
1            1191   987   848  1322
2             586   909  1341  1260
3            1879  1760  1681  1820
4             530   815  1127  1038
```




```
#Preparing the sample dataframe to be plotted
groupsize_counts = final_dataframe.groupby(['cluster', 'group']).size().unstack()

#Printing the sample dataframe
print(round_trip_counts)

#Creating the Bar plot
groupsize_counts.plot(kind='bar', stacked=True, colormap='Reds')

# Adding plot details
plt.title('Group size per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.legend(title='Group size Code')

# Displaying the plot
plt.show()
```

```
group       0     1
cluster
0        2171  5310
1        1797  2551
2        2724  1372
3        2330  4810
4        1730  1780
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the boxplot without showing outliers
sns.boxplot(x='cluster', y='group_size', data=final_dataframe, showfliers=False, color='indianred')

# Show grid lines
plt.grid(True)

# Customize grid appearance (optional)
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Adding plot details
plt.title('Group size per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Group size')

# Displaying the plot
plt.show()
```