**Data Preparation**

We have prepared a Google Spreadsheet containing the raw customer segmentation data.

[TravelTide Customer Segmentation Data](https://docs.google.com/spreadsheets/d/1ZWExDp5UCbj0OdNgTnj6sNgMGfY6rvMUseS_EchrjNY/edit#gid=1954874297)

**Methodology**

We used a combination of SQL and Python (with the scikit-learn library) to analyze and cluster our customer data.

**SQL Query**

Used for filtering user data, calculating key metrics, removing outliers, scaling metrics, and segmenting customers.

**Python Code**

K-means clustering applied on scaled customer metrics.

Visualizations

1. Executive Summary Visualizations

Cluster sizes

Focus on 'Thrifty Travelers'

2. Presentation Slides

[Google Slide link](https://docs.google.com/presentation/d/15vVUjovQi_Cxjun3LHSifWbpZbXqyzp4sodr5X9EwSo/edit#slide=id.p)

[Zoom Link for Recorded Slide Presentation](https://us05web.zoom.us/clips/share/BHVzMDQgkNi5yZjSnkmpf2peF_NVbQFagoCyRMJj8u8UwITBS1I)

**Python Code for Visualizations on Executive Summary**

import matplotlib.pyplot as plt

# Assuming 'cluster\_sizes' is a Pandas Series with the number of members in each cluster

cluster\_labels = ['Thrifty Travelers', 'Luggage Leaders', 'Luxe Lodgers', 'Diverse Jetsetters', 'High-Rolling Experimenters']

cluster\_counts = cluster\_sizes.sort\_index().values # Sort index to align with labels

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

plt.barh(cluster\_labels, cluster\_counts, color='skyblue')

plt.xlabel('Number of Members')

plt.ylabel('Clusters')

plt.title('Cluster Sizes')

plt.show()

**Python Code for Visualizations on Slidedeck**

import matplotlib.pyplot as plt

# Focusing only on 'Thrifty Travelers' here for simplification

cluster\_label = ['Thrifty Travelers']

cluster\_count = [1769]

plt.figure(figsize=(8, 4))

plt.barh(cluster\_label, cluster\_count, color='skyblue')

plt.xlabel('Number of Members')

plt.title('The Thrifty Travelers: Our Largest Customer Group')

plt.show()

**SQL Query for Data Filtering and Metric Calculation**

WITH filtered\_users AS (

SELECT user\_id

FROM users

WHERE sign\_up\_date > '2023-01-04'

),

-- Key metrics calculation

key\_metrics AS (

SELECT

fu.user\_id,

COUNT(DISTINCT f.trip\_id) AS total\_flights\_booked,

COUNT(DISTINCT h.trip\_id) AS total\_hotel\_stays,

SUM(h.rooms) AS total\_rooms\_booked,

AVG(f.base\_fare\_usd) AS avg\_flight\_fare,

AVG(h.hotel\_per\_room\_usd) AS avg\_hotel\_fare,

SUM(CASE WHEN s.cancellation = TRUE THEN 1 ELSE 0 END) AS total\_cancellations,

AVG(f.checked\_bags) AS avg\_checked\_bags,

AVG(h.rooms) AS avg\_rooms\_per\_stay

FROM filtered\_users fu

LEFT JOIN sessions s ON fu.user\_id = s.user\_id AND s.session\_start >= '2023-01-04'

LEFT JOIN flights f ON s.trip\_id = f.trip\_id

LEFT JOIN hotels h ON s.trip\_id = h.trip\_id

GROUP BY fu.user\_id

HAVING COUNT(DISTINCT s.session\_id) > 7

),

-- Percentiles Calculation for Outlier Filtering

percentiles AS (

SELECT

PERCENTILE\_CONT(0.05) WITHIN GROUP (ORDER BY total\_flights\_booked) AS p5\_flights,

PERCENTILE\_CONT(0.95) WITHIN GROUP (ORDER BY total\_flights\_booked) AS p95\_flights

FROM key\_metrics

),

-- Remove outliers based on percentiles

key\_metrics\_filtered AS (

SELECT km.\*

FROM key\_metrics km, percentiles p

WHERE km.total\_flights\_booked BETWEEN p.p5\_flights AND p.p95\_flights

),

-- Min and Max Values for Scaling

minmax\_values AS (

SELECT

MIN(total\_flights\_booked) AS min\_flights, MAX(total\_flights\_booked) AS max\_flights,

MIN(total\_hotel\_stays) AS min\_hotels, MAX(total\_hotel\_stays) AS max\_hotels,

MIN(avg\_flight\_fare) AS min\_flight\_fare, MAX(avg\_flight\_fare) AS max\_flight\_fare,

MIN(avg\_hotel\_fare) AS min\_hotel\_fare, MAX(avg\_hotel\_fare) AS max\_hotel\_fare,

MIN(total\_cancellations) AS min\_cancellations, MAX(total\_cancellations) AS max\_cancellations,

MIN(avg\_checked\_bags) AS min\_checked\_bags, MAX(avg\_checked\_bags) AS max\_checked\_bags

FROM key\_metrics\_filtered

),

-- Apply Min-Max Scaling

scaled\_metrics AS (

SELECT

kmf.\*,

(kmf.total\_flights\_booked - mm.min\_flights) / (mm.max\_flights - mm.min\_flights) AS scaled\_total\_flights,

(kmf.total\_hotel\_stays - mm.min\_hotels) / (mm.max\_hotels - mm.min\_hotels) AS scaled\_total\_hotels,

(kmf.avg\_flight\_fare - mm.min\_flight\_fare) / (mm.max\_flight\_fare - mm.min\_flight\_fare) AS scaled\_avg\_flight\_fare,

(kmf.avg\_hotel\_fare - mm.min\_hotel\_fare) / (mm.max\_hotel\_fare - mm.min\_hotel\_fare) AS scaled\_avg\_hotel\_fare,

(kmf.total\_cancellations - mm.min\_cancellations) / (mm.max\_cancellations - mm.min\_cancellations) AS scaled\_total\_cancellations,

(kmf.avg\_checked\_bags - mm.min\_checked\_bags) / (mm.max\_checked\_bags - mm.min\_checked\_bags) AS scaled\_avg\_checked\_bags

FROM key\_metrics\_filtered kmf, minmax\_values mm

),

-- Customer Segmentation

customer\_segmentation AS (

SELECT

user\_id,

CASE

WHEN scaled\_avg\_hotel\_fare < 0.2 AND scaled\_total\_hotels < 0.2 THEN 'Free Hotel Meal'

WHEN scaled\_avg\_checked\_bags >= 0.8 THEN 'Free Checked Bag'

WHEN scaled\_total\_cancellations >= 0.8 THEN 'No Cancellation Fees'

WHEN scaled\_avg\_flight\_fare < 0.2 AND scaled\_avg\_hotel\_fare < 0.2 THEN 'Exclusive Discounts'

WHEN scaled\_total\_flights >= 0.8 AND scaled\_total\_hotels >= 0.8 THEN '1 Night Free Hotel with Flight'

ELSE 'General'

END AS predicted\_favorite\_perk,

scaled\_total\_flights,

scaled\_total\_hotels,

scaled\_avg\_flight\_fare,

scaled\_avg\_hotel\_fare,

scaled\_total\_cancellations,

scaled\_avg\_checked\_bags

FROM scaled\_metrics

)

SELECT \* FROM customer\_segmentation;

**Python Code for Clustering**

*Code 1*

import pandas as pd

from sklearn.cluster import KMeans

# Read the CSV file into a pandas DataFrame

df = pd.read\_csv('scaled\_metrics.csv')

# Prepare the data (selecting only the scaled metrics columns)

X = df[['scaled\_total\_flights', 'scaled\_total\_hotels', 'scaled\_avg\_flight\_fare', 'scaled\_avg\_hotel\_fare', 'scaled\_total\_cancellations', 'scaled\_avg\_checked\_bags']].copy()

# Handle NaN values

X.dropna(inplace=True)

# Perform K-Means clustering

kmeans = KMeans(n\_clusters=5, n\_init=10,random\_state=42) # Explicitly set n\_init to 10

X['cluster\_label'] = kmeans.fit\_predict(X)

# Save the DataFrame back to CSV, including the new 'cluster\_label' column

X.to\_csv('scaled\_metrics\_with\_clusters.csv', index=False)

*Code 2*

new\_df = pd.read\_csv('scaled\_metrics\_with\_clusters.csv')

print(new\_df.head())

*Code 3*

cluster\_summary = new\_df.groupby('cluster\_label').mean()

print(cluster\_summary)

*Code 4*

import pandas as pd

from sklearn.cluster import KMeans

# Read the CSV file into a pandas DataFrame

df = pd.read\_csv('scaled\_metrics.csv')

# Prepare the data (selecting only the scaled metrics columns)

X = df[['scaled\_total\_flights', 'scaled\_total\_hotels', 'scaled\_avg\_flight\_fare', 'scaled\_avg\_hotel\_fare', 'scaled\_total\_cancellations', 'scaled\_avg\_checked\_bags']].copy()

# Handle NaN values

X.dropna(inplace=True)

# Perform K-Means clustering

kmeans = KMeans(n\_clusters=5, n\_init=10, random\_state=42) # Explicitly set n\_init to 10

X['cluster\_label'] = kmeans.fit\_predict(X)

# Count the number of data points in each cluster

cluster\_sizes = X['cluster\_label'].value\_counts()

# Print cluster sizes

print("Cluster Sizes:")

print(cluster\_sizes)

# Save the DataFrame back to CSV, including the new 'cluster\_label' column

X.to\_csv('scaled\_metrics\_with\_clusters.csv', index=False)

*Code 5*

import pandas as pd

# Load the scaled metrics with clusters CSV into a DataFrame

scaled\_metrics\_df = pd.read\_csv('scaled\_metrics\_with\_clusters.csv')

# Calculate summary statistics by cluster

summary\_df = scaled\_metrics\_df.groupby('cluster\_label').agg({

'cluster\_label': 'size',

'scaled\_total\_flights': 'mean',

'scaled\_total\_hotels': 'mean',

'scaled\_avg\_flight\_fare': 'mean',

'scaled\_avg\_hotel\_fare': 'mean',

'scaled\_total\_cancellations': 'mean',

'scaled\_avg\_checked\_bags': 'mean'

}).rename(columns={

'cluster\_label': 'Cluster Size',

'scaled\_total\_flights': 'Avg scaled\_total\_flights',

'scaled\_total\_hotels': 'Avg scaled\_total\_hotels',

'scaled\_avg\_flight\_fare': 'Avg scaled\_avg\_flight\_fare',

'scaled\_avg\_hotel\_fare': 'Avg scaled\_avg\_hotel\_fare',

'scaled\_total\_cancellations': 'Avg scaled\_total\_cancellations',

'scaled\_avg\_checked\_bags': 'Avg scaled\_avg\_checked\_bags'

})

# Load the Min-Max values

minmax\_df = pd.read\_csv('minmax\_values.csv')

minmax\_dict = minmax\_df.to\_dict(orient='records')[0]

# Map between Min-Max dictionary keys and DataFrame columns

column\_map = {

'scaled\_total\_flights': 'flights',

'scaled\_total\_hotels': 'hotels',

'scaled\_avg\_flight\_fare': 'flight\_fare',

'scaled\_avg\_hotel\_fare': 'hotel\_fare',

'scaled\_total\_cancellations': 'cancellations',

'scaled\_avg\_checked\_bags': 'checked\_bags'

}

# Un-scale the average scaled values back to their original amounts

for column, original\_metric in column\_map.items():

min\_val = minmax\_dict.get('min\_' + original\_metric)

max\_val = minmax\_dict.get('max\_' + original\_metric)

avg\_column = 'Avg ' + column[7:] # Create the name for the new "Avg" column

if min\_val is not None and max\_val is not None:

summary\_df[avg\_column] = (summary\_df['Avg ' + column] \* (max\_val - min\_val)) + min\_val

# Save summary statistics to a CSV file with original amounts

summary\_df.to\_csv('cluster\_summary\_statistics\_in\_dollars.csv')

# Display summary statistics

print(summary\_df)