TravelTide data segmentation

Introduction

TravelTide, a rising e-booking startup, has been gaining traction in the online travel industry since its launch in April 2021. They're known for their cutting-edge technology that provides the largest travel inventory around.

But, there's a catch – TravelTide has been hyper-focused on inventory and searchability, which has left some customer experience areas lacking, leading to poor retention rates. The company is determined to fix this with a strong marketing strategy.

The Marketing department is renowned for its expertise in customer retention strategies, particularly using rewards programs and they're here to supercharge TravelTide's marketing efforts and keep customers coming back for more.

The goal is to create an amazing personalized rewards program that encourages customers to keep using TravelTide. To make this happen, the marketing department needs to understand the customers first, and for that, they will have to collaborate closely with the data team to gather insights.

The marketing department has identified perks that are most likely to attract customers as part of the reward program. The identified perks are the following

- Free hotel meal
- Free checked bags
- No cancellation fees
- Exclusive discounts
- 1 Night free hotel with flight

The marketing department thinks that to catch customers' interest and boost the chances they'll join the rewards program; we should highlight the perks we think they want the most when we invite them to sign up.

As Data Analysts, we've got a two-part mission. First, we'll see if the data backs up the idea that certain customers are extra interested in the perks as the marketing team suggested. After that, we'll figure out which perk is most likely to be a customer's favourite.

Let's think about this business challenge and dive into the TravelTide database:

- What travel behaviors suggest a customer's preference for each perk? For instance, who might really want a free checked bag?
- Which parts of the database hold data about these behaviors?
- How should we structure the data (like filtering or aggregating) to make sure our segmentation analysis makes sense and isn't flawed?

TravelTide Data Processing

We have a large dataset that includes 4 tables with 1,020,926 users and over 5.4 million sessions ranging from April 2021 to July 2023.

New customers have limited interaction data, which can skew our analysis given our business objective which is to set up a rewards program based on customer behavior as it takes time for behavioral data to accumulate. We need a way to account for the impact of time on the platform. In simple terms, we need to pick a specific group (cohort) for our analysis.

After discussion with the marketing team, it was decided to only include sessions starting after the New Year's holiday (2023-01-04) until the last available date in the database and to only include users with more than 7 sessions during the same period.

Visual representation of our cohort:

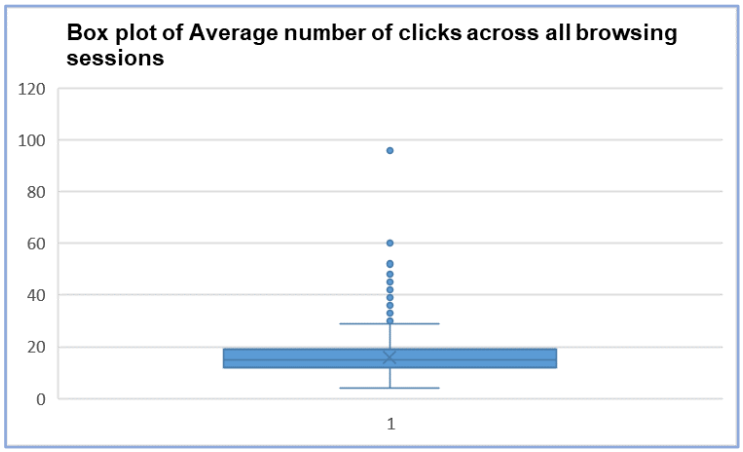
This cohort was extracted using an SQL query (SQL query 1 - Appendix).

Aggregating the data

Several data aggregations were extracted from the database using SQL queries to identify the distribution and be aware of outliers as they can skew the data. Some of the aggregations are:

- 1 - Average number of clicks across all browsing sessions of the cohort

This was extracted using (SQL query 2 - Appendix). The plotted distribution is below. This was done in Excel after downloading the file.



As can be seen from the plot, we may have outliers in this case. However, this was mentioned to emphasize the need to understand the cause of such extreme values (Are they due to data entry errors, measurement errors, or do they represent genuine extreme values in your dataset?) rather than removing them without identifying the cause and then deciding in line with the context and objective of the business. In some cases, outliers may represent genuine extreme values that are important for your analysis. This issue will be decided in every analysis separately as we progress with the data processing and analysis process.

Calculating Metrics

Metrics help measure business performance from different angles. Some of these metrics focus on how business strategy and customer behavior align in an overall view.

Our metric-making efforts have a clear goal: for each perk in the rewards program, we're after data that shows strong interest in that perk. Now, let's take a closer look and analyze each of the 5 perks individually.

➢ Exclusive discounts

The marketing team suggests we're targeting price-sensitive customers for this perk, particularly those who really respond to discount pricing. While everyone appreciates a good deal, we're looking for customers who show significant variability in their bargain-hunting behavior.

Indeed, it's crucial to pinpoint behaviors in the TravelTide database that strongly correlate with pricing and discount preferences for this perk.

Several aggregations that are believed to be highly related to this have been computed. And, all were extracted from the dataset separately and then joined on a spreadsheet using the VLOOKUP function on the user_id.

1 - **The percentage of flight bookings under discount (As “discount_flight_proportion”)** – This was extracted using SQL query 3 – in the appendix. This code is written to include only sessions that ended up with booking a flight and the proportion was calculated by dividing the total number of bookings made with discount by the total number of bookings made. This was done on the user level.

2 - **The average flight discount amount in percentage terms (As "Average_flight_discount") -** This was extracted using SQL query 4 – in the appendix. The average size of the discount was calculated on the user level on all flight bookings.

3 - **The scaled Average Dollars Saved per kilometer (As "scaled_ADSpkm")** amount relative to others. This was extracted using SQL query 5 – in the appendix. This was calculated on the user level and only included booked flights. Cancellation sessions were also excluded as the flight was being canceled and the users were not saving money on them.

The Average Dollars Saved (ADS) calculation has a drawback - it doesn't consider variations in customer travel patterns. Some customers take short domestic flights, while others fly internationally. Consequently, ADS assigns higher values to longer-distance travelers, as longer trips tend to be pricier. To correct this bias, the ADS was divided by the distance traveled (After calculating the haversine distance), giving us dollars saved per kilometer traveled.

Also, to account for the large scale of base_fare_usd (in the thousands of dollars), MinMax transformation was used on the data.

Two approaches were suggested when using MinMax transformation on the data, the average → scale (what we did), and the scale → average. But as we are trying to compare customers based on the average discounts without de-emphasizing large differences in discounts between customers the former approach was used.

Moreover, after discussions with the marketing team other metrics that are related to price sensitivity and responsiveness to discounts were calculated. Those include:

4 – Average Browsing duration per session (As "ABDPS") - This was extracted using SQL query 6 – in the appendix.

5- Average number of clicks per session (As "ACPS") - This was extracted using SQL query 7 – in the appendix.

As we are trying to identify customers who will be more interested in the Exclusive discount perk it is important to combine all these different metrics to give Us a better representation of the customer's interest and affinity towards the perks. However, as the different metrics that were identified here are on different scales, performing scaling on the metrics will be integral before combining the multiple metrics into a single figure to build an index.

Scaling is crucial as it levels the playing field for all variables in your dataset. When variables have different units or magnitudes, they can unfairly sway the segmentation process, potentially leading to less-than-ideal outcomes. Scaling levels the variables, ensuring they have an equal say in the analysis, and preventing those with larger ranges from exerting undue influence.

Scaling was done in SQL and the queries are attached in the appendix.

When building the indexes there are several approaches; multiplication, addition, and Principal Component Analysis (PCA). And in this case, the sum of the identified metrics was used to build the Bargain_hunter_index. This was done to prevent metrics with zero measure from nullifying the overall result (Bargain_hunter_index).

However, there was another important aspect to keep in mind.

When metrics show strong correlation, combining them into a product might not provide substantial extra insights compared to examining them individually. However, when metrics are less correlated, the product could offer more valuable information.

Therefore, the correlations between all the identified metrics that are thought to be related to the Bargain Hunter index were tested in Excel using the Pearson correlation coefficient. The results are as follows. The links to the related files are attached at the end of this document.

Correlation	Pearsons's coefficient
Average_flight_discount and proportion of flights booked with discount	0.83
Scaled_ADSpkm and Average_flight_discount	0.80
Scaled_ADSpkm and proportion of flights booked with discount	0.66
Scaled_Average_Browsing_duration_per_session (ABDPS) and scaled_ADSpkm	0.03
Scaled_Average_Browsing_duration_per_session (ABDPS) and scaled_Average_number_of_clicks_per_session (ACPS)	0.99

The correlations were also tested using a scatter plot to observe the type and strength of the relationship.

The Average_flight_discount and proportion of flights booked with discount were highly correlated (0.83). The Scaled_ADSpkm was also clearly correlated with the proportion of flights booked with discount (0.66) and also with Average_flight_discount (0.8). Therefore, fore it was decided to use the Scaled_ADSpkm only as this is a better representation of the discount by taking the length of the flight into consideration.

The Scaled_Average_Browsing_duration_per_session (ABDPS) was also highly correlated (0.99) with the scaled_Average_number _of_clicks_per_session (ACPS). And from these two it was decided to take Scaled_Average_Browsing_duration_per_session (ABDPS) to avoid multicollinearity. The distribution of the selected metrics is below and both of them appears to be right skewed.

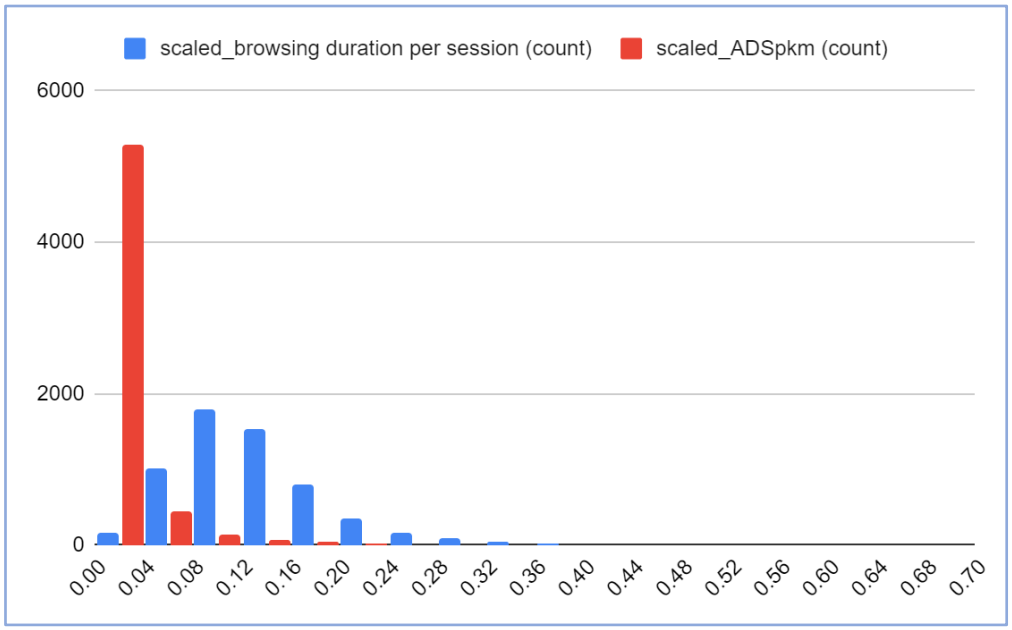
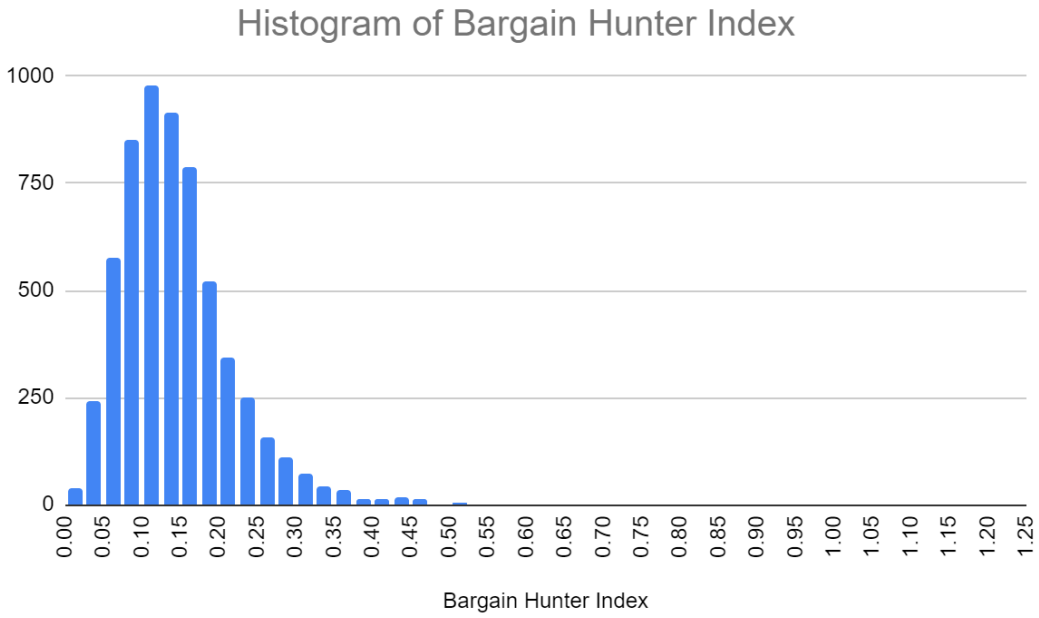


Fig. - Distribution of the metrics

Therefore, the Scaled_Average_Browsing_duration_per_session (ABDPS) and scaled_ADSpkm were used to build the Bargain Hunter Index using the summation method.



Similar methods were used to build affinity indexes for the remaining perks which is below.

➢ 1 Night free hotel with flight

The following metrics were extracted in relation to this perk. All were extracted from the dataset separately and then joined on a spreadsheet using the VLOOKUP function on the user_id.

1 - **The percentage of hotel bookings under discount (As "discount_hotel_proportion")** – This was extracted using SQL query 8 – in the appendix. This code is written to include only sessions

that ended up with booking a hotel room and the proportion was calculated by dividing the total number of hotel bookings made with discount by the total number of bookings made. This was done on the user level.

2 - The average hotel discount amount in percentage terms (As "Average_hotel_discount") - This was extracted using SQL query 9 – in the appendix. The average size of the discount was calculated on the user level on all hotel bookings.

3 - The scaled Average Dollars Saved per booked room (As "ADS_scaled_pnr") amount relative to others. This was extracted using SQL query 10 – in the appendix. This was calculated on the user level and only included sessions that ended up booking a hotel room. Cancellation sessions were also excluded as the flight was being canceled and the users were not saving money on them.

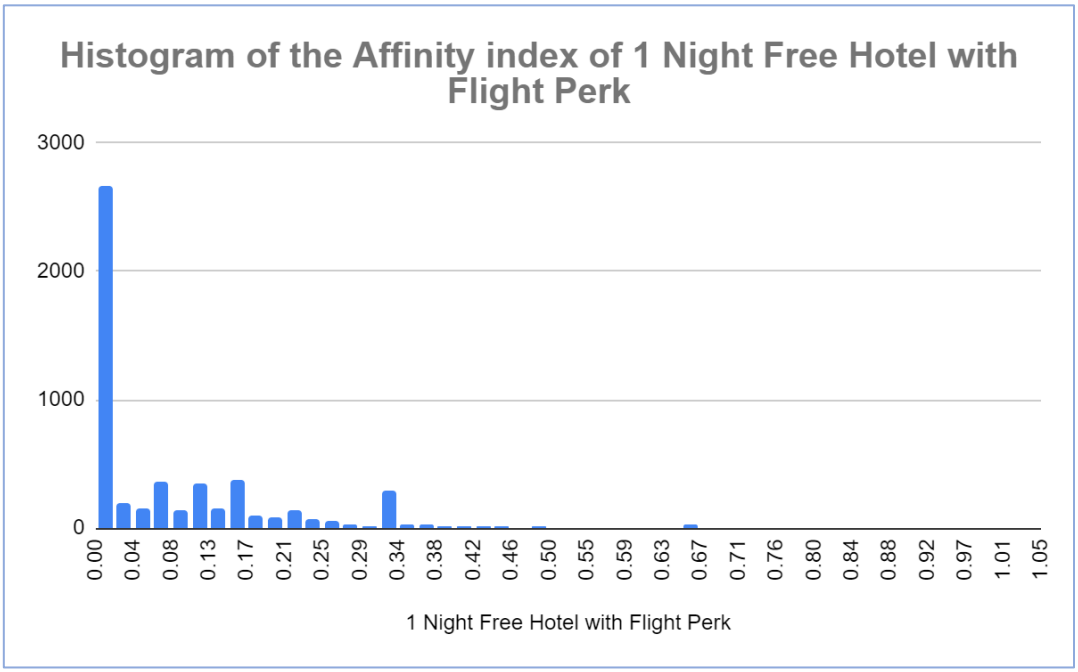
Because ADS assigns higher values to customers who booked more rooms, The scaled Average Dollars Saved per booked room was calculated to correct this bias, the ADS was divided by the average number of rooms booked, giving us dollars saved per room booked.

4 – Average number of rooms booked per flight (AS "scaled_ANRB") - Moreover, based on information from several flight booking websites it was understood that customers with a higher average number of hotel room bookings tend to be attracted to free hotel room reward programs. Therefore, the Scaled form of the average number of rooms booked per flight was extracted using SQL query 11.

CORRELATIONS (Using Pearson's coefficient)	
The scaled Average Dollars Saved per booked room vs discount_hotel_proportion	0.68
The scaled Average Dollars Saved per booked room vs Average_hotel_discount	0.79
The scaled Average Dollars Saved per booked room vs the Average number of rooms booked per flight	-0.054
discount_hotel_proportion vs Average_hotel_discount	0.860
discount_hotel_proportion vs Scaled Average number of rooms booked per flight	0.029

The correlations were also tested using scatter plot to observe the type and strength of the relationship.

Therefore, the scaled Average Dollars Saved per booked room and the scaled Average number of rooms booked per flight only were used to build the Affinity index for the "1 Night free hotel with flight" perk as both of them have no correlation and they give a better representation of the customer's behavior. The remaining metrics were dropped to avoid multicollinearity and as they don't add any additional value to the index.



➢ No cancellation fees

Based on information from several flight booking websites it was understood that a "free cancellation fee loyalty bonus" when booking flights is a perk that can appeal to a wide range of travelers, but considering the limited data we have Travellers with higher rates of flight cancellation and travelers who book flights last minute who often have unpredictable schedules were the standout behaviors that also be calculated with the TravelTide dataset.

Therefore, the **Proportion of cancellation** was extracted using SQL query 12.

Moreover, Average_time_between_booking_and_flight (As “ATBBF”) was also extracted using SQL query 13. But as we need the customers with shorter durations between booking and flight to have higher scores, the obtained data - Average_time_between_booking_and_flight- was Normalized to reflect this (refer to code in appendix). Null values in this regard were converted to Zero after the data was normalized to show that those customer does not have any cancellations.

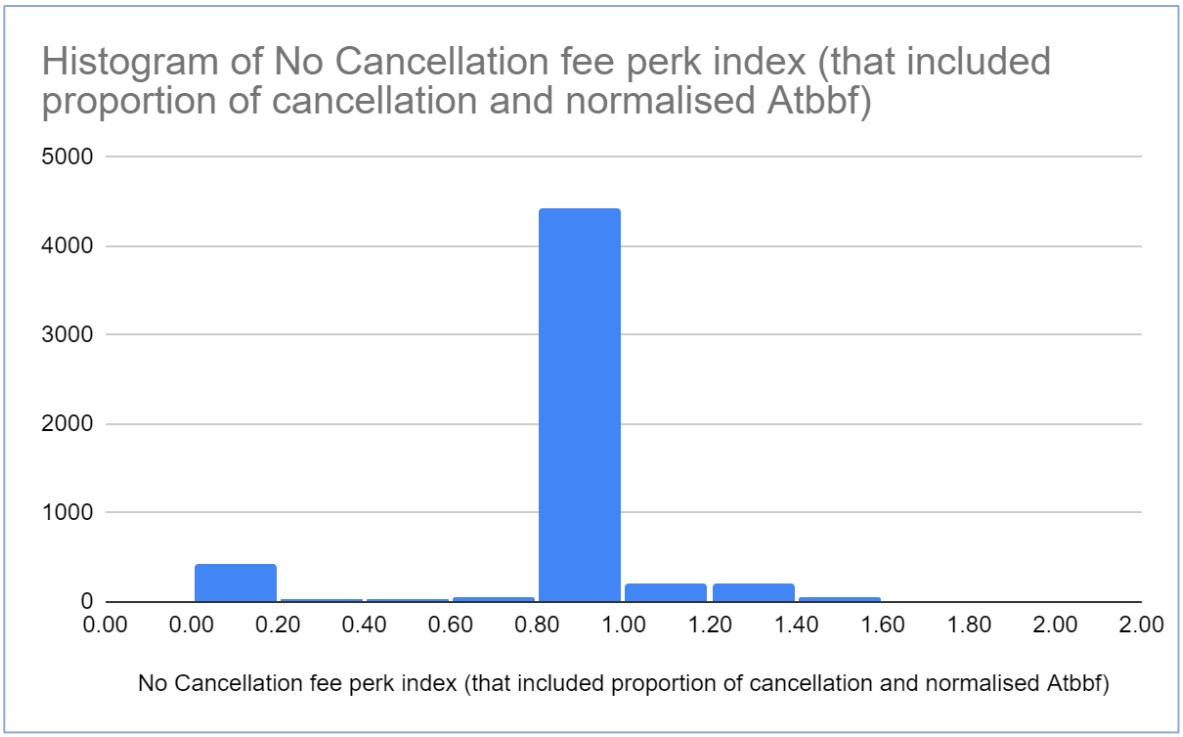
And, both were extracted from the dataset separately and then joined on a spreadsheet using the VLOOKUP function on the user_id.

Then Correlations were tested to avoid multicollinearity using Pearson’s coefficient.

Correlation b/n proportion of cancellation and normalized Average Time Between Booking & Flight (ATBBF)	-0.1717760378

The correlations were also tested using scatter plot to observe the type and strength of the relationship.

There is a slight correlation between the metrics but it was decided to include both metrics as both add value to the overall result.



➢ Free checked bags

Based on information from several flight booking websites three behaviors were common as indicators of interest to Free checked bags reward programs. Customers with longer duration of travel, traveling longer distances, and families traveling with children. It seems very logical to guess those identified types of customers are very likely to have more and larger baggage when traveling. Also, as the TravelTide dataset includes the history of travel specifically the previous history of the Average number of bags checked per flight was identified as another metric that has to be calculated. These Metrics were calculated as follows. And, all were extracted from the dataset separately and then joined on a spreadsheet using the VLOOKUP function on the user_id.

1 - **Average_number_bags_checked (AS "ANBC") –** this was extracted from the data set using the SQL query 14. This Metric was also Minmax scaled.

2 – **Average distance travelled** - this was extracted from the data set using the SQL query 15. This metric was transformed using the Minmax transformation as well.

3 – **Has_children** – This is present in the data set and was extracted for the cohort. However, as this was highly correlated with the Average_number_bags_checked, the has children metric was dropped because the Average_number_bags_checked appears to give a better representation towards the affinity score. Furthermore, the presence of children does not necessarily entail constant travel with them.

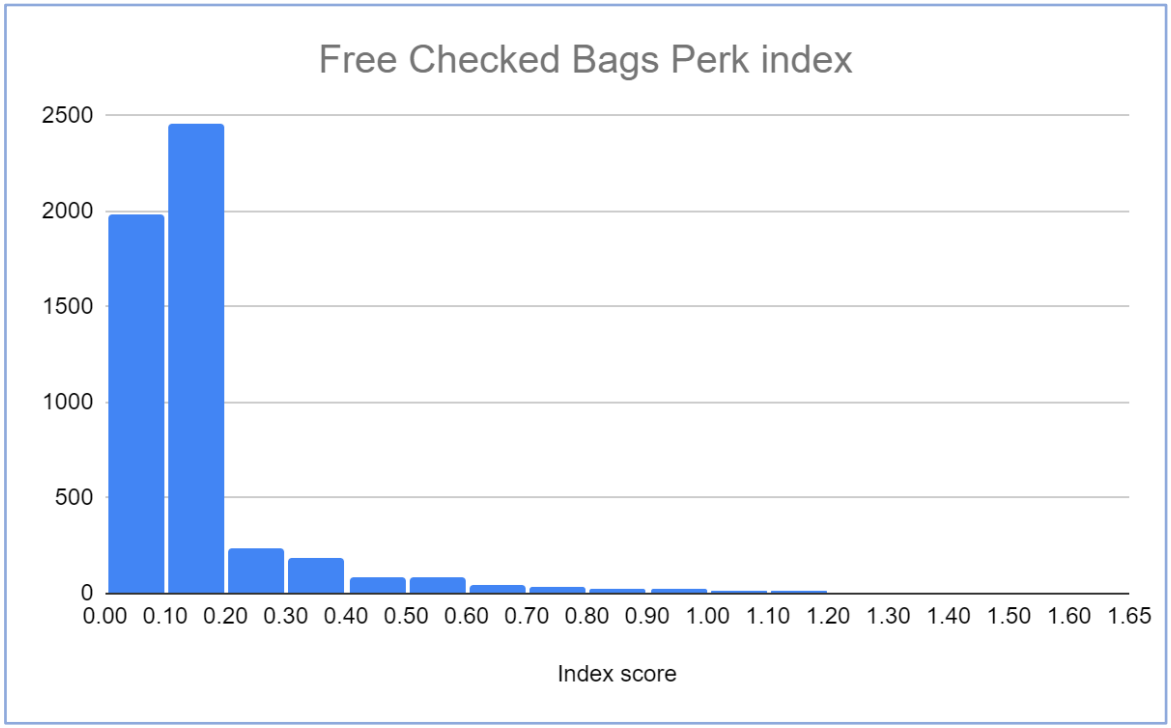
4 – **Length of stay (as "LOS")** - this was extracted from the data set using the SQL query 16. This metric was transformed using the Minmax transformation as well.

Correlations were checked between metrics. Results as follows:

Correlation between length of stay and the average number of bags checked	0.18
Correlation between length of stay and average distance traveled	0.53
Correlation between the average number of checked bags and average distance traveled	0.12

The correlations were also tested using scatter plot to observe the type and strength of the relationship.

The affinity index (Free checked bags perk index) was calculated as a sum of the length of stay and the Average Number of Bags Checked due to the moderate level of correlation found between length of stay and average distance traveled to avoid multicollinearity.



➢ Free hotel meal

Again, based on information from several flight booking websites it was mentioned that older people, families with children, and price-sensitive customers tend to be more interested in the

complimentary hotel meals with flights. Therefore, all three were extracted from the dataset separately and then joined on a spreadsheet using the VLOOKUP function on the user_id.

1 – Age – The age of customers was extracted in years and then scaled using an SQL query 17.

2 – Has_children – this metric is available in the dataset and customers with children were labeled as 1 and those with no children as 0. As the data is binary and only ranges between 0 and 1, this metric was not scaled. SQL query 18 was used to extract this data.

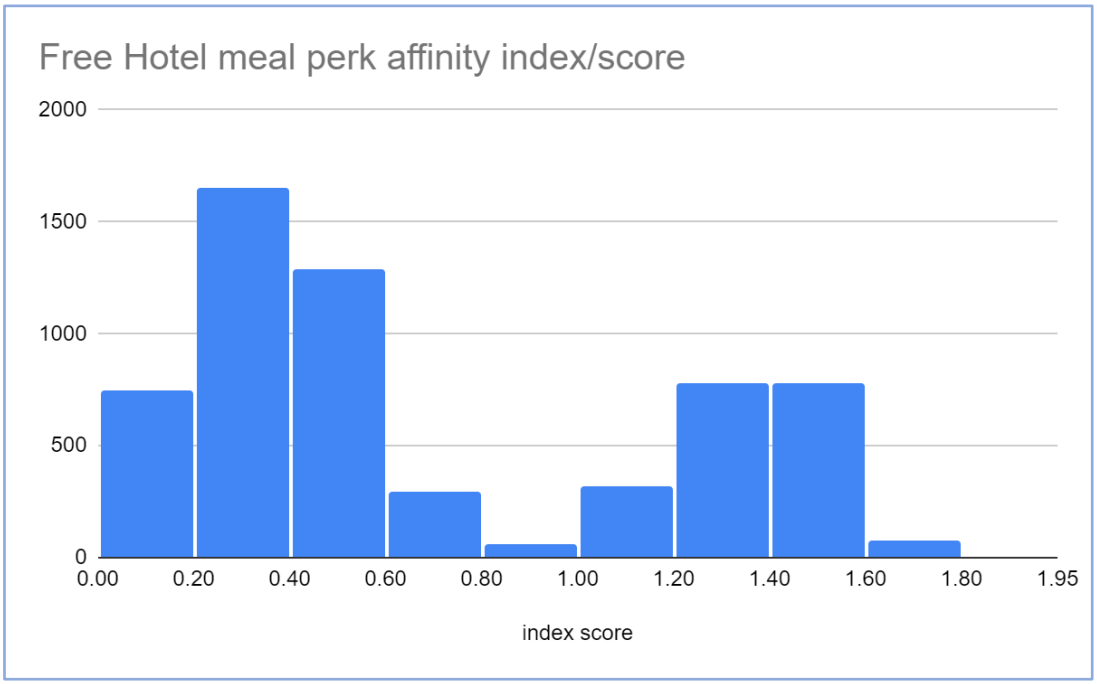
3 – The average flight discount amount in percentage terms (As "Average_flight_discount") - was also extracted to represent price_sensitivity. This was extracted using SQL query 4 – in the appendix. The average size of the discount was calculated on the user level on all flight bookings.

Correlations were checked between metrics. Results as follows:

Correlation between age and having children	-0.0009
Correlation between age and average discount amount	0.01
Correlation between having children and average discount amount	0.003

The correlations were also tested using scatter plot to observe the type and strength of the relationship.

As there was no correlation between all the metrics, all of them were added up to give the Free Hotel meal perk affinity index/score.



The graph is bimodal and shows that we have 2 groups of customers based on this result.

Segmenting Data

Regarding segmenting customers based on the marketing team's perks for the rewards program, it's essential to assign each customer to one specific perk. The idea is to capture customer attention by presenting them with the perk that excites them the most. This means that each customer should exclusively belong to one segment, and this exclusivity is known as mutual exclusivity among segments.

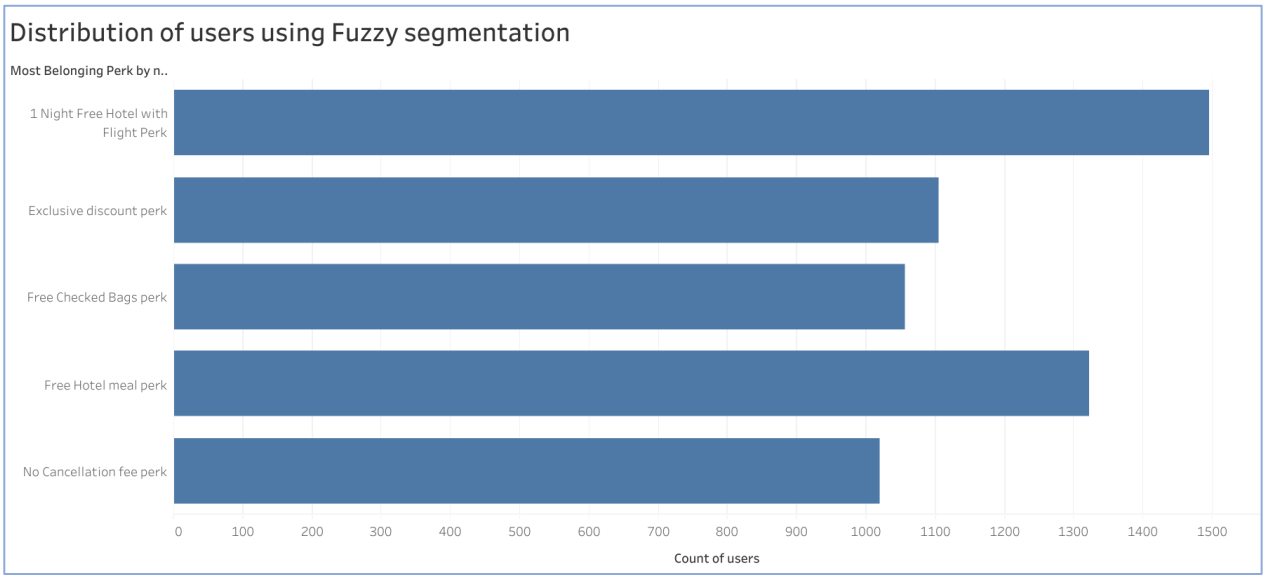
So, to achieve this the solution is a strategy known as Fuzzy Segmentation. In Fuzzy Segmentation, customers can belong to more than one segment, but to varying degrees.

So all the metrics and the affinity scores calculated for every perk/reward program were collected to different worksheets in the spreadsheet. And, a new table was formed using the VLOOKUP function that includes the user id and the user's affinity score for each of the 5 perks. Full details can be found in the appendix and attached files.

Then the customers were ranked based on the affinity index score of every perk separately. And customers with the highest score in each perk were given the lowest rank. This gave every customer a ranking in every affinity index for all the 5 perks.

Then the perk corresponding to this minimum rank is the one the customers most strongly relate to and every user was assigned to this respective perk.

The downside of this method is that a user with less affinity towards a particular perk can be assigned to that perk merely because the minimum rank they got across all of the 5 perks lay on that perk. This can have an impact on the effectiveness of the reward program.



In the next part, a k-means clustering was conducted to compare results.

Advanced task 1 - a Better Haversine Distance Function

This is attached as a file “Haversine.py”.

Advanced task 2 - Make a Tableau Dashboard

In the workbook attached among files. The dashboard was made to show the distribution of 5 affinity scores for all the perks and demographics by customer segment. The affinity scores were selected because they give a better representation of customers interest towards the perks than the individual metrics alone. Also the dashboard include a percentile line for all the distributions with a slider to control the value of the percentile.

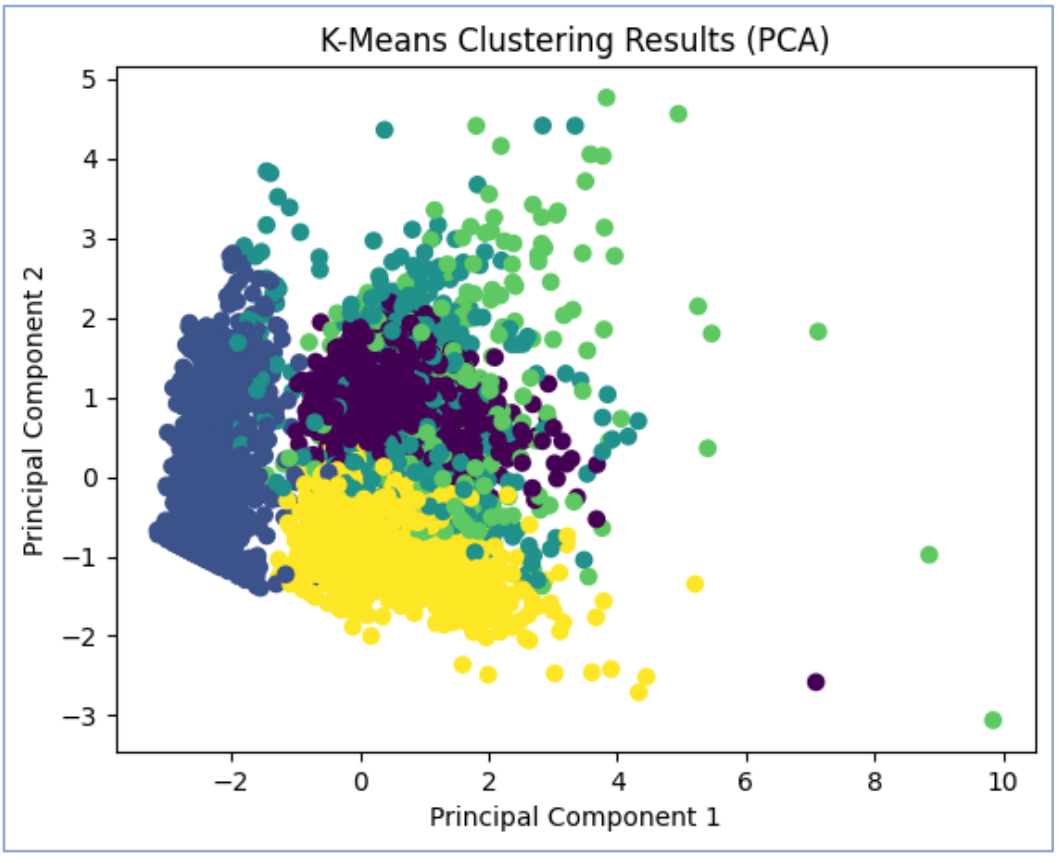
Advanced task 3 - K-means

A k-means clustering was conducted (notebook attached). The following was the visualization obtained.

The following steps were taken to do the Kmeans clustering.

1. Data Preprocessing and K-Means Clustering Visualization:

   - Data is read from a CSV file.
   - Specific columns are selected for scaling.
   - Data is standardized using `StandardScaler`.
   - K-Means clustering with 5 clusters is applied to the scaled data.
   - Dimensionality is reduced using PCA to visualize the clusters in 2D.
   - Clusters are visualized in 2D using a scatter plot.



2. Cluster Centroids:

- Cluster centroids are calculated and printed, representing the average values for each feature within each cluster.

- Cluster labels are assigned to the data based on the K-Means model's predictions.

3. Segment Profiles:

- Mean values for each feature within each cluster are calculated and printed, creating segment profiles.

4. Cluster Evaluation:

- The Silhouette Score and Davies-Bouldin Index are calculated to evaluate the quality of the clusters.

- Clusters are visualized in 2D using PCA with cluster labels displayed.

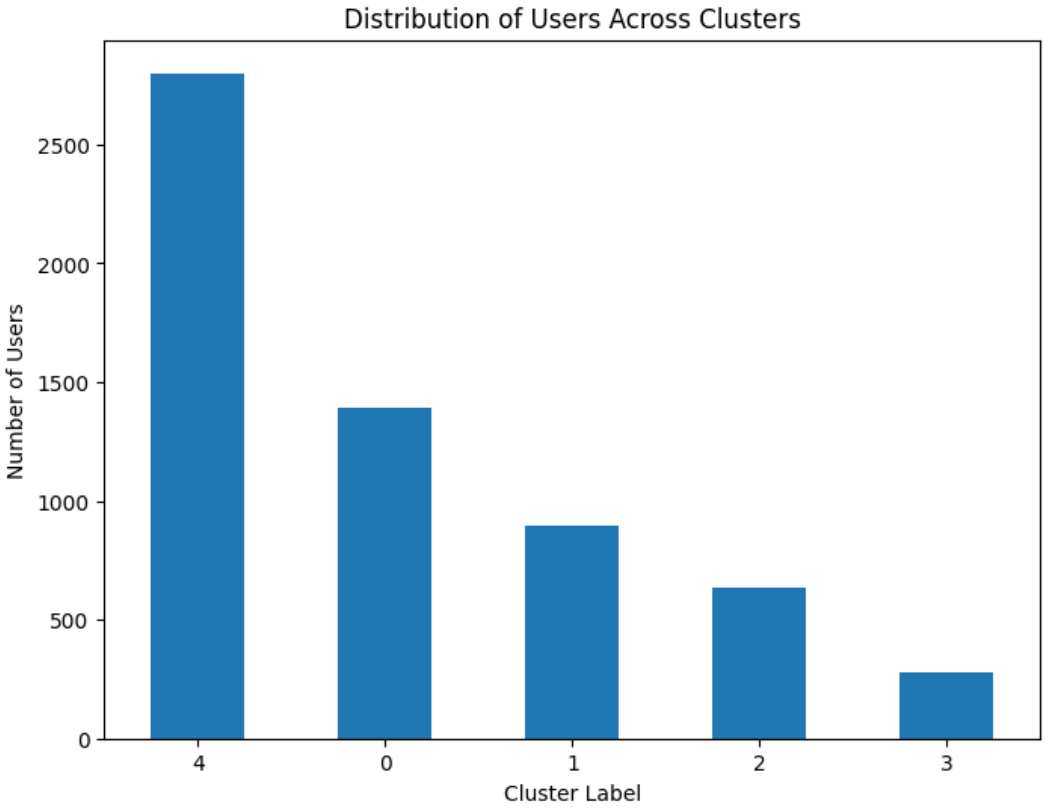
The Silhouette Score and Davies-Bouldin Index are two common metrics used for evaluating the quality of clusters obtained from a clustering algorithm like K-Means. Silhouette Score (0.3669) in this case tells us that the clusters have some degree of separation and cohesion but may not be perfectly distinct. It indicates a reasonable quality of clustering, with room for improvement.

And Davies-Bouldin Index (1.0501) tells us that the clusters are reasonably well-separated but may have some degree of overlap.

So, to wrap it up, based on these scores, it looks like the clusters are somewhat clear, but there's room to make them even better with some fine-tuning.

5. Cluster Counts:

- The number of users in each cluster is counted, and a bar plot is created to visualize the distribution of users across clusters.



6. Displaying Cluster Members:

- Users belonging to each cluster are displayed individually.

7. Saving Results:

- Clustering results are saved to a CSV file.

The saved file that contains the K-means cluster was joined with the Fuzzy segmentation data on a spreadsheet using VLOOKUP and was saved as a CSV file. The new file was then uploaded again to Google Collab and the two clusters were compared based using the adjusted Rand index (ARI) and normalized mutual information (NMI).

8. Printing ARI and NMI:

- The Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) are metrics used to compare the similarity or agreement between two different clusterings. In this project, we are comparing two clustering methods: one based on the affinity score index and the other based on K-Means. The Adjusted Rand Index (ARI) is 0.1722 indicating a relatively low level of

agreement between the two clustering methods. The Normalized Mutual Information (NMI) is 0.2776 and suggests that there is some shared information between the two clusterings, but the agreement is not very strong. The two methods result in clusters that are moderately similar.

Overall, the ARI and NMI scores indicate that there is some level of similarity between the affinity-based clustering and the K-Means clustering, but it's not a very strong agreement.

Conclusion and recommendation

All of the customers have been assigned to a specific group using the Fuzzy segmentation method. However, as mentioned above, the downside of fuzzy segmentation is that a user with less affinity towards a particular perk can be assigned to that perk merely because the minimum rank they got across all of the 5 perks lay on that perk. This can have an impact on the effectiveness of the reward program. Moreover, Small changes in the ranking can result in different segment assignments.

The second method of clustering the k-means clustering indicates that the clusters have some degree of separation and cohesion but may not be perfectly distinct. It indicates a reasonable quality of clustering, with room for improvement.

When both clustering methods were compared there was some level of similarity between the affinity-based clustering and the K-Means clustering, but it's not a very strong agreement.

Therefore, the customer's assignment to different perks using the fizzy segmentation is provided as a file. The clustering using the K-means clustering is not included as it does not add significant value to the segmentation done using the fizzy segmentation. However further experimentation and reiteration with different clustering algorithms, parameters, or preprocessing steps to see if you can achieve a stronger agreement between the two methods.

Furthermore, it will be important to identify behavioral metrics from objective and reliable sources to determine customers' affinity toward the different perks.

Appendix

SQL query 1

```
-- to create session-level data for our filtered cohort
WITH Cohort AS (
    SELECT user_id, COUNT(*) AS session_count
    FROM sessions
    WHERE session_start > '2023-01-04'
    GROUP BY user_id
    HAVING COUNT(*) > 7)
SELECT u.*, s.*, f.*, h.*
FROM sessions s
JOIN cohort ct ON s.user_id = ct.user_id
LEFT JOIN users u ON s.user_id = u.user_id
LEFT JOIN flights f ON s.trip_id= f.trip_id
LEFT JOIN hotels h ON s.trip_id= h.trip_id
WHERE session_start > '2023-01-04';
```

SQL query 2

```
--Average_clicks_per_session(ACPS)
WITH cohort as (SELECT user_id
        FROM sessions
        WHERE session_start > '2023-01-04'
        GROUP BY user_id
        HAVING COUNT(*) > 7)
SELECT s.user_id, AVG(page_clicks)::FLOAT AS ACPS
FROM sessions s
JOIN cohort ct ON s.user_id = ct.user_id
LEFT JOIN users u ON s.user_id = u.user_id
LEFT JOIN flights f ON s.trip_id= f.trip_id
LEFT JOIN hotels h ON s.trip_id= h.trip_id
WHERE session_start > '2023-01-04' AND cancellation IS NOT TRUE
GROUP BY 1;
```

SQL query 3

```
-- Percentage of flight bookings under discount (discount_flight_proportion)
```

```
WITH cohort AS (
   SELECT user_id, COUNT(*) AS session_count
   FROM sessions
   WHERE session_start > '2023-01-04'
   GROUP BY user_id
   HAVING COUNT(*) > 7)
SELECT SUM(CASE WHEN flight_discount THEN 1 ELSE 0 END)  / COUNT(*) AS discount_flight_proportion, s.user_id
FROM sessions s
JOIN cohort ct ON s.user_id = ct.user_id
LEFT JOIN users u ON s.user_id = u.user_id
LEFT JOIN flights f ON s.trip_id= f.trip_id
WHERE session_start > '2023-01-04' AND flight_booked IS TRUE AND cancellation IS NOT TRUE
GROUP BY s.user_id;
```

SQL query 4

```
-- average_flight_discount_amount - average discount size on flight purchases (in percentage terms)
--The CASE WHEN line is included to avoid null values (Flights booked without a discount) being overlooked as they are not included in the average otherwise
WITH cohort AS (
   SELECT user_id, COUNT(*) AS session_count
   FROM sessions
   WHERE session_start > '2023-01-04'
   GROUP BY user_id
   HAVING COUNT(*) > 7)
SELECT s.user_id, AVG(CASE WHEN flight_discount_amount <>0 then flight_discount_amount else 0 end):: FLOAT AS Average_flight_discount
FROM sessions s
JOIN cohort ct ON s.user_id = ct.user_id
LEFT JOIN users u ON s.user_id = u.user_id
LEFT JOIN flights f ON s.trip_id= f.trip_id
LEFT JOIN hotels h ON s.trip_id= h.trip_id
WHERE session_start > '2023-01-04' AND flight_booked IS TRUE AND cancellation IS NOT TRUE
GROUP BY s.user_id;
```

SQL query 5

```
--  Scaled " Average dollars saved per unit distance(KM)" (ADSPKM)
WITH cohort AS (
    SELECT user_id
```

```
    FROM sessions
    WHERE session_start > '2023-01-04'
    GROUP BY user_id
    HAVING COUNT(*) > 7),
    haversine_table as (SELECT*,
                6371 * 2 * ASIN(
                  SQRT(
                    POWER(SIN(RADIANS((destination_airport_lat - home_airport_lat) / 2)), 2) +
                    COS(RADIANS(home_airport_lat)) * COS(RADIANS(destination_airport_lat)) *
                    POWER(SIN(RADIANS((destination_airport_lon - home_airport_lon) / 2)), 2)
                  )
                ) AS haversine_distance
               FROM(
                  SELECT
                    s.user_id, s.session_id, s.trip_id, home_airport_lat,
                    home_airport_lon,
                    destination_airport_lat,
                    destination_airport_lon
                  FROM sessions s
                  LEFT JOIN users u ON s.user_id = u.user_id
                  JOIN flights f ON s.trip_id = f.trip_id
               ) AS subquery),
 metrics AS (
        SELECT
            s.user_id,
    SUM((CASE WHEN flight_discount_amount <>0 then flight_discount_amount else 0 end)*base_fare_usd)/SUM(haversine_distance) AS ADSpkm
        FROM sessions s
        JOIN cohort ct ON s.user_id = ct.user_id
        LEFT JOIN users u ON s.user_id = u.user_id
        LEFT JOIN flights f ON s.trip_id = f.trip_id
        LEFT JOIN hotels h ON s.trip_id = h.trip_id
        LEFT JOIN haversine_table hd ON s.trip_id = hd.trip_id
        WHERE session_start > '2023-01-04'AND flight_booked IS TRUE AND cancellation IS NOT TRUE
        GROUP BY s.user_id
      )
SELECT
```

```
    user_id,
    (ADSpkm - MIN(ADSpkm) OVER ()) / (MAX(ADSpkm) OVER () - MIN(ADSpkm) OVER ()) AS scaled_ADSpkm
FROM metrics;
```

SQL query 6

```
--scaled Average browsing_duration_per_session (BDPS)
WITH cohort AS (
    SELECT user_id
    FROM sessions
    WHERE session_start > '2023-01-04'
    GROUP BY user_id
    HAVING COUNT(*) > 7
),
metrics AS (
    SELECT
        s.user_id,
        SUM(EXTRACT(EPOCH FROM (session_end - session_start)))/ COUNT(*) AS BDPS
    FROM sessions s
    JOIN cohort ct ON s.user_id = ct.user_id
    LEFT JOIN users u ON s.user_id = u.user_id
    LEFT JOIN flights f ON s.trip_id = f.trip_id
    LEFT JOIN hotels h ON s.trip_id = h.trip_id
    WHERE session_start > '2023-01-04' AND cancellation IS NOT TRUE
    GROUP BY 1
)
SELECT
    mt.user_id,
    (BDPS - MIN(BDPS) OVER ()) / (MAX(BDPS) OVER () - MIN(BDPS) OVER ()) AS scaled_BDPS
FROM metrics mt;
```

SQL query 7

```
--Average number of clicks per session (As "ACPS")
WITH cohort as (SELECT user_id
                FROM sessions
                WHERE session_start > '2023-01-04'
```

```
        GROUP BY user_id
        HAVING COUNT(*) > 7)
SELECT s.user_id, AVG(page_clicks)::INT AS ACPS
FROM sessions s
JOIN cohort ct ON s.user_id = ct.user_id
LEFT JOIN users u ON s.user_id = u.user_id
LEFT JOIN flights f ON s.trip_id= f.trip_id
LEFT JOIN hotels h ON s.trip_id= h.trip_id
WHERE session_start > '2023-01-04' AND cancellation IS NOT TRUE
GROUP BY 1;
```

SQL query 8

```
-- to calculate the percentage of hotel bookings under a discount

WITH cohort AS (SELECT user_id, COUNT(*) AS session_count
        FROM sessions
        WHERE session_start > '2023-01-04'
        GROUP BY user_id
        HAVING COUNT(*) > 7)
  SELECT s.user_id, SUM(CASE WHEN hotel_discount THEN 1 ELSE 0 END) :: FLOAT / COUNT(*) AS discount_hotel_proportion
  FROM sessions s
  JOIN cohort ct ON s.user_id = ct.user_id
  LEFT JOIN users u ON s.user_id = u.user_id
  LEFT JOIN flights f ON s.trip_id= f.trip_id
  LEFT JOIN hotels h ON s.trip_id= h.trip_id
  WHERE session_start > '2023-01-04' AND hotel_booked IS TRUE
  GROUP BY s.user_id;
```

SQL query 9

```
--average hotel discount size
WITH cohort AS (
    SELECT user_id, COUNT(*) AS session_count
    FROM sessions
```

```
    WHERE session_start > '2023-01-04'
    GROUP BY user_id
    HAVING COUNT(*) > 7)
SELECT s.user_id,
    AVG(CASE WHEN hotel_discount_amount <>0 then hotel_discount_amount else 0 end)::FLOAT AS Average_hotel_discount
    FROM sessions s
    JOIN cohort ct ON s.user_id = ct.user_id
    LEFT JOIN users u ON s.user_id = u.user_id
    LEFT JOIN flights f ON s.trip_id= f.trip_id
    LEFT JOIN hotels h ON s.trip_id= h.trip_id
    WHERE session_start > '2023-01-04' AND hotel_booked IS TRUE AND cancellation IS NOT TRUE
    GROUP BY s.user_id;
```

SQL query 10

```
--scaled average dollar saved for hotel discount amount PER NUMBER OF ROOMS (As "ADS_scaled_pnr")
WITH cohort AS (SELECT user_id, COUNT(*) AS session_count
            FROM sessions
            WHERE session_start > '2023-01-04'
            GROUP BY user_id
            HAVING COUNT(*) > 7),
        metrics AS (SELECT s.user_id,
                SUM((CASE WHEN hotel_discount_amount <>0 then hotel_discount_amount else 0 end)*hotel_per_room_usd)/SUM(rooms) AS
ADS_hotel_pnr
                FROM sessions s
                JOIN cohort ct ON s.user_id = ct.user_id
                LEFT JOIN users u ON s.user_id = u.user_id
                LEFT JOIN flights f ON s.trip_id = f.trip_id
                LEFT JOIN hotels h ON s.trip_id = h.trip_id
                WHERE session_start > '2023-01-04' AND hotel_booked IS TRUE AND cancellation IS NOT TRUE
                GROUP BY s.user_id)
      SELECT user_id, (ADS_hotel_pnr - MIN(ADS_hotel_pnr) OVER ()) / (MAX(ADS_hotel_pnr) OVER () - MIN(ADS_hotel_pnr) OVER ()) AS
scaled_ADS_hotel_pnr
      FROM metrics;
```

SQL query 11

```
-- Average number of rooms booked (ANRB) Scaled


WITH cohort AS (SELECT user_id, COUNT(*) AS session_count
                FROM sessions
                WHERE session_start > '2023-01-04'
                GROUP BY user_id
                HAVING COUNT(*) > 7),
     metrics AS (SELECT s.user_id, AVG(rooms) AS ANRB
                FROM sessions s
                JOIN cohort ct ON s.user_id = ct.user_id
                LEFT JOIN users u ON s.user_id = u.user_id
                LEFT JOIN flights f ON s.trip_id = f.trip_id
                LEFT JOIN hotels h ON s.trip_id = h.trip_id
                WHERE session_start > '2023-01-04' AND hotel_booked IS TRUE AND cancellation IS NOT TRUE
                GROUP BY s.user_id)
  SELECT
     user_id,
     (ANRB - MIN(ANRB) OVER ()) / (MAX(ANRB) OVER () - MIN(ANRB) OVER ()) AS scaled_ANRB
  FROM metrics;
```

SQL query 12

```
-- proportion_of_cancelation_from_all_booking
WITH cohort AS (SELECT user_id, COUNT(*) AS session_count
                FROM sessions
                WHERE session_start > '2023-01-04'
                GROUP BY user_id
                HAVING COUNT(*) > 7)
    SELECT s.user_id,
    SUM(CASE WHEN cancellation THEN 1 ELSE 0 END) :: FLOAT / COUNT(*) AS proportion_of_cancelation_from_all_booking
    FROM sessions s
```

```
    JOIN cohort ct ON s.user_id = ct.user_id
    LEFT JOIN users u ON s.user_id = u.user_id
    LEFT JOIN flights f ON s.trip_id= f.trip_id
    LEFT JOIN hotels h ON s.trip_id= h.trip_id
    WHERE session_start > '2023-01-04' AND hotel_booked IS TRUE
    GROUP BY s.user_id;
```

SQL query 13

```
--Average_time_between_booking_and_flight (ATBBF) Normalised


WITH cohort AS (SELECT user_id, COUNT(*) AS session_count
                FROM sessions
                WHERE session_start > '2023-01-04'
                GROUP BY user_id
                HAVING COUNT(*) > 7),
  Metrics AS (SELECT s.user_id,
            AVG(EXTRACT(EPOCH FROM (departure_time - session_end))) / 86400.00  AS ATBBF
            FROM sessions s
            JOIN cohort ct ON s.user_id = ct.user_id
            LEFT JOIN users u ON s.user_id = u.user_id
            LEFT JOIN flights f ON s.trip_id= f.trip_id
            LEFT JOIN hotels h ON s.trip_id= h.trip_id
            WHERE session_start > '2023-01-04' AND hotel_booked IS TRUE
            GROUP BY s.user_id)
SELECT
    user_id,
    (1-(ATBBF - MIN(ATBBF) OVER ()) / (MAX(ATBBF) OVER () - MIN(ATBBF) OVER ())) AS Normalised_ATBBF, ATBBF
FROM metrics;
```

SQL query 14

```
--Average_number_bags_checked (ANBC) scaled


WITH cohort AS (
                    SELECT user_id, COUNT(*) AS session_count
                    FROM sessions
```

```
            WHERE session_start > '2023-01-04'
            GROUP BY user_id
            HAVING COUNT(*) > 7,
   Metrics AS (SELECT s.user_id,
        AVG(checked_bags) AS ANBC
        FROM sessions s
        JOIN cohort c ON s.user_id = c.user_id
        LEFT JOIN users u ON s.user_id = u.user_id
        LEFT JOIN flights f ON s.trip_id= f.trip_id
        LEFT JOIN hotels h ON s.trip_id= h.trip_id
        WHERE session_start > '2023-01-04' AND flight_booked IS TRUE
        GROUP BY s.user_id)
SELECT user_id,
(ANBC - MIN(ANBC) OVER ()) / (MAX(ANBC) OVER () - MIN(ANBC) OVER ()) AS scaled_ANBC
FROM Metrics;
```

SQL query 15

```
--Scaled_average distance travelled
WITH cohort AS (SELECT user_id
            FROM sessions
            WHERE session_start > '2023-01-04'
            GROUP BY user_id
            HAVING COUNT(*) > 7),
    haversine_table as (SELECT*,
        6371 * 2 * ASIN(
          SQRT(
            POWER(SIN(RADIANS((destination_airport_lat - home_airport_lat) / 2)), 2) +
            COS(RADIANS(home_airport_lat)) * COS(RADIANS(destination_airport_lat)) *
            POWER(SIN(RADIANS((destination_airport_lon - home_airport_lon) / 2)), 2)
          )
        ) AS haversine_distance
        FROM(
          SELECT
            s.user_id, s.session_id, s.trip_id, home_airport_lat,
            home_airport_lon,
```

```
            destination_airport_lat,
            destination_airport_lon
        FROM sessions s
        LEFT JOIN users u ON s.user_id = u.user_id
        JOIN flights f ON s.trip_id = f.trip_id
    ) AS subquery),
Metrics AS (SELECT s.user_id,
            AVG( haversine_distance) AS average_distance
            FROM sessions s
            JOIN cohort c ON s.user_id = c.user_id
            LEFT JOIN users u ON s.user_id = u.user_id
            LEFT JOIN flights f ON s.trip_id = f.trip_id
            LEFT JOIN hotels h ON s.trip_id = h.trip_id
            LEFT JOIN haversine_table hd ON s.trip_id = hd.trip_id
            WHERE session_start > '2023-01-04' AND cancellation IS NOT TRUE
            GROUP BY s.user_id )
SELECT user_id,
  (average_distance  - MIN(average_distance ) OVER ()) / (MAX(average_distance ) OVER () - MIN(average_distance ) OVER ()) AS
scaled_average_distance
FROM Metrics;
```

SQL query 16

```
--Average Length of stay (LOS) scaled

WITH cohort AS (
        SELECT user_id, COUNT(*) AS session_count
        FROM sessions
        WHERE session_start > '2023-01-04'
        GROUP BY user_id
        HAVING COUNT(*) > 7),
  Metrics AS (SELECT s.user_id,
      AVG(EXTRACT(EPOCH FROM (departure_time - session_end))) / 86400.00 AS LOS
      FROM sessions s
      JOIN cohort c ON s.user_id = c.user_id
      LEFT JOIN users u ON s.user_id = u.user_id
```

```
        LEFT JOIN flights f ON s.trip_id= f.trip_id
        LEFT JOIN hotels h ON s.trip_id= h.trip_id
        WHERE session_start > '2023-01-04' AND flight_booked IS TRUE
        GROUP BY s.user_id)
SELECT user_id,
(LOS - MIN(LOS) OVER ()) / (MAX(LOS) OVER () - MIN(LOS) OVER ()) AS scaled_LOS
FROM Metrics;
```

SQL query 17

```
--age (scaled)
--2023-07-29 last date in of booking in database and was used as a reference

WITH cohort AS (
            SELECT user_id, COUNT(*) AS session_count
            FROM sessions
            WHERE session_start > '2023-01-04'
            GROUP BY user_id
            HAVING COUNT(*) > 7
             ),
Metric as (SELECT u.user_id,
        ROUND(('2023-07-29' - birthdate)/365.25 , 0) AS AgeInYears
        FROM users u
       JOIN cohort ct ON u.user_id = ct.user_id)
 SELECT user_id,
(AgeInYears - MIN(AgeInYears) OVER ()) / (MAX(AgeInYears) OVER () - MIN(AgeInYears) OVER ()) AS scaled_AgeInYears
FROM Metric;
```

SQL query 18

```
--has children
WITH cohort AS (
            SELECT user_id, COUNT(*) AS session_count
            FROM sessions
            WHERE session_start > '2023-01-04'
            GROUP BY user_id
```

```
            HAVING COUNT(*) > 7)
SELECT u.user_id,
  CASE WHEN has_children THEN 1 ELSE 0 END AS has_children
  FROM users u
  JOIN cohort c ON u.user_id = c.user_id ;
```

SQL query 19

Advanced task 1 - a Better Haversine Distance Function

```
CREATE OR REPLACE FUNCTION haversine_distance(lat1 float, lon1 float, lat2 float, lon2 float)
RETURNS float AS $$
DECLARE
  radius float = 6371.0; -- Earth's radius in kilometers

  dlat float = RADIANS(lat2 - lat1);
  dlon float = RADIANS(lon2 - lon1);

  a float = SIN(dlat / 2) * SIN(dlat / 2) +
        COS(RADIANS(lat1)) * COS(RADIANS(lat2)) *
        SIN(dlon / 2) * SIN(dlon / 2);

  c float = 2 * ATAN2(SQRT(a), SQRT(1 - a));

  distance float = radius * c;
BEGIN
  RETURN distance;
END;
$$ LANGUAGE plpgsql;
```

Links to SPREADSHEET FILES USED IN THE PROCESS

All data with perk affinity scores and ranking and assignemt

https://docs.google.com/spreadsheets/d/1FDr1KnQVWqQZUkY78q8vL30buBh_jnZLBSeSwLlWm6Q/edit?usp=sharing

Exclusive discount perk

https://docs.google.com/spreadsheets/d/13f-MUWH0hmo7nX1WYZb5a04otsMtXXIFnEcQpqUgZTA/edit?usp=sharing

Free hotel meal index

https://docs.google.com/spreadsheets/d/1sKEIW5hXS3kpARhouQe9ye4janFzKmqSmTWUYiqwNkg/edit?usp=sharing

free checked bags perk

https://docs.google.com/spreadsheets/d/1Jwmi20seElDJ7qxTv0zrubtMKRZEq4fqd9q2Ogltkuc/edit?usp=sharing

No cancellation fee perk

https://docs.google.com/spreadsheets/d/1TsueC3VfBh2s_g6rnQn6nxDDqOav_hORPXY9Ot_9EQI/edit?usp=sharing

1 night free hotel with flight

https://docs.google.com/spreadsheets/d/1QRacsQLW7DPUQo-kB_malAQZzrG8J1vTIcFwsZekJ3U/edit?usp=sharing

Kmeans data

https://docs.google.com/spreadsheets/d/1cGsmyNIP7ODXceL1m4wOMcHAZkVhr1EMRNazno1ZEkk/edit?usp=sharing

Kmeans clustered data

https://docs.google.com/spreadsheets/d/1ws9i38SHmSVFRxJFNpkS-bUCn1_74ib0QlPqrJrcpy4/edit?usp=sharing

https://docs.google.com/spreadsheets/d/1YbC3EpRJxBCeXNoeCjZkpo_eWCCTxm3yiq6bSbnv9cY/edit?usp=sharing