```sql
-- Throughout this query we used standard scaling. We created separate
CTEs for scaling in which we called for the average of the metric and
then the STDDEV of the metric.
-- We have done this so in our index CTEs we can use these values in
the standard scaling formula.

WITH cohort AS (SELECT user_id FROM sessions
                        WHERE session_start >= '2023/01/04'
                        GROUP BY user_id
                        HAVING COUNT(session_id) > 7),

---For Exclusive discounts
avg_dollars_saved AS (
        SELECT
          user_id,
          COALESCE(SUM(s.flight_discount_amount *
f.base_fare_usd)/SUM(haversine_distance(u.home_airport_lat,u.home_air
port_lon,f.destination_airport_lat,f.destination_airport_lon)) , 0)
as ADS
          FROM sessions s
          LEFT JOIN flights f ON s.trip_id = f.trip_id
          LEFT JOIN users u USING (user_id)
          WHERE session_start >= '2023/01/04'
          GROUP BY s.user_id
          HAVING COUNT(session_id) > 7
),
-- Standard Scaling the CTE above
-- We do this throughout our query.
ScalingADS AS (
  SELECT
    AVG(ads) AS avg_ads,
    STDDEV(ads) AS stddev_ads
  FROM avg_dollars_saved
),

ads_flight_percentage AS(
              SELECT
    user_id,
              COALESCE( SUM(CASE WHEN flight_discount THEN 1 ELSE 0
END) :: FLOAT / COUNT(*), 0) AS dfp

              FROM sessions
              WHERE session_start >= '2023/01/04'
              GROUP BY user_id
              HAVING COUNT(session_id) > 7
```

```
),

ScalingADS2 AS (
  SELECT
    AVG(dfp) AS avg_dfp,
    STDDEV(dfp) AS stddev_dfp
  FROM ads_flight_percentage
),

avg_f_dis_amt AS (
              SELECT
              COALESCE(AVG(flight_discount_amount),0) AS
average_flight_discount,
              user_id
              FROM sessions
              WHERE session_start >= '2023/01/04'
              GROUP BY user_id
              HAVING COUNT(session_id) > 7
),

Scalingafda3 AS (
  SELECT avg(average_flight_discount) AS avg_afd,
  STDDEV(average_flight_discount) AS stddev_afd
FROM avg_f_dis_amt
  ),

 flight_conversion_rate AS (
              SELECT
              user_id,
              SUM(CASE WHEN flight_booked THEN 1 ELSE 0 END) *1.0 /(
COUNT(session_id) *1.0  ) AS flight_conv_rate
              FROM sessions
              WHERE session_start >= '2023/01/04'
              GROUP BY user_id
              HAVING COUNT(session_id) > 7

),

Scalingfcr4 AS (
  SELECT avg(flight_conv_rate) AS avg_fcr,
  STDDEV(flight_conv_rate) AS stddev_fcr
FROM flight_conversion_rate
  ),
```

```
-- The CTE below is used to scale the values as well as finding the
index giving equal weightage to all of the values.
  RankedEXDisc AS(
SELECT
   ads.user_id,
-- Below we are using the Standard scaling formula to get each metric
scaled before combining them in an index.
COALESCE(( ads.ads -  avg_ads) / stddev_ads , 0) as
flights_ads_scaled,
COALESCE(( afp.dfp -  avg_dfp) / stddev_dfp , 0) as
flights_proportion_Discount_scaled,
COALESCE((afda.average_flight_discount - avg_afd) /  stddev_afd , 0)
AS flights_discount_amount_scaled,
COALESCE((fcr.flight_conv_rate - avg_fcr) /  stddev_fcr , 0) AS
flights_conversion_rate_scaled,
-- Below is the code for the creation of the index for the Exclusive
discount perk.
-- We do this for all the indexes throughout our query.
(.25 * COALESCE(( ads.ads -  avg_ads) / stddev_ads , 0)) +
(.25 * COALESCE(( afp.dfp -  avg_dfp) / stddev_dfp , 0)) +
(.25 * COALESCE((afda.average_flight_discount - avg_afd) /
stddev_afd , 0)) +
(.25 * COALESCE((fcr.flight_conv_rate - avg_fcr) /  stddev_fcr , 0))
AS Index_ExclusiveDiscount


FROM avg_dollars_saved ads
LEFT JOIN ads_flight_percentage afp USING (user_id)
LEFT JOIN avg_f_dis_amt afda USING (user_id)
LEFT JOIN flight_conversion_rate fcr using (user_id)
-- We used CROSS JOINs to call upon the same value for every row of
our dataset.
-- Additionally we use this throughout the rest of perks.
CROSS JOIN ScalingADS
CROSS JOIN Scalingads2
CROSS JOIN Scalingafda3
CROSS JOIN Scalingfcr4
GROUP BY 1,2,3,4,5

),

-- Ranked Free Meal

Above55 AS(
```

```sql
  select u.user_id , Case when EXTRACT(YEAR FROM birthdate) < 1968
then 1 else 0 end as above_55
  from sessions s
  LEFT JOIN users u
  using (user_id)
              WHERE session_start >= '2023/01/04'
              GROUP BY u.user_id
              HAVING COUNT(session_id) > 7 ),

ScalingAbove55 AS (
  SELECT
    AVG(above_55) AS avg_above_55,
    STDDEV(above_55) AS stddev_above_55
  FROM Above55
),

 trip_count AS(

  select u.user_id , count(s.trip_id) as trip_id_count
  from sessions s
  LEFT JOIN users u
  using (user_id)
              WHERE session_start >= '2023/01/04'
              GROUP BY 1
              HAVING COUNT(session_id) > 7
),

ScalingTripCount AS (
  SELECT
    AVG(trip_id_count) AS avg_trip_id_count,
    STDDEV(trip_id_count) AS stddev_trip_id_count
  FROM trip_count
),

kids AS
(select u.user_id , Case when has_children then 1 else 0 end AS
children
  from sessions s
  LEFT JOIN users u
  using (user_id)
              WHERE session_start >= '2023/01/04'
              GROUP BY 1,2
              HAVING COUNT(session_id) > 7
 ),
```

```
ScalingKids AS (
  SELECT avg(children) AS avg_kids,
  STDDEV(children) AS stddev_kids
FROM kids
  ),

Ranked_Free_Meal AS (

SELECT
   a55.user_id,
COALESCE(( a55.above_55 -  avg_above_55) / stddev_above_55 , 0) AS
above_55_scaled,
COALESCE(( tc.trip_id_count -  avg_trip_id_count) /
stddev_trip_id_count , 0) AS trip_count_scaled,
COALESCE((kid.children - avg_kids) /  stddev_kids , 0) AS
has_kids_scaled,

(.33 * COALESCE(( a55.above_55 -  avg_above_55) / stddev_above_55 ,
0)) +
(.33 * COALESCE(( tc.trip_id_count -  avg_trip_id_count) /
stddev_trip_id_count , 0)) +
(.33 * COALESCE((kid.children - avg_kids) /  stddev_kids , 0)) AS
Index_Free_Meal
FROM Above55 a55
LEFT JOIN trip_count tc USING (user_id)
LEFT JOIN kids kid USING (user_id)
CROSS JOIN ScalingAbove55
CROSS JOIN ScalingTripCount
CROSS JOIN ScalingKids
GROUP BY 1,2,3,4
 ),

--- For Free bag

 longdistancetraveller AS(
  select s.user_id ,
COALESCE(avg(haversine_distance(home_airport_lat,home_airport_lon,des
tination_airport_lat,destination_airport_lon)),0) distance
  from sessions s
  LEFT JOIN users u
  using (user_id)
  LEFT JOIN flights f
  using(trip_id)
              WHERE session_start >= '2023/01/04'
              GROUP BY 1
```

```
                HAVING COUNT(session_id) > 7
        ),

Scalingldt AS(
  SELECT
    AVG(distance) AS avg_dist,
    STDDEV(distance ) AS stddev_dist
  FROM longdistancetraveller
),

SumOfBagsNSeats AS (
SELECT u.user_id,
COALESCE(SUM(f.checked_bags),0) AS total_checked_bags ,
COALESCE(SUM(f.seats),0) AS num_of_seats
FROM users u
LEFT JOIN sessions s ON u.user_id = s.user_id
LEFT JOIN flights f ON s.trip_id = f.trip_id
WHERE s.session_start >= '2023-01-04'
GROUP BY u.user_id
HAVING COUNT(s.session_id) > 7
  ),

 Hasmorebagsthanseats AS
 (
   SELECT user_id , CASE WHEN total_checked_bags > num_of_seats THEN
1 ELSE 0 END AS morebags
   FROM SumOfBagsNSeats
   GROUP BY 1,2
   ),

   Scalingbns AS (
  SELECT
    AVG(morebags) AS avg_bags,
    STDDEV(morebags ) AS stddev_bags
  FROM Hasmorebagsthanseats
),

LongTrip AS(
Select user_id , max(COALESCE(EXTRACT(EPOCH FROM (return_time -
departure_time)), 0)) / 86400   as timedifference

from sessions s
  LEFT JOIN flights f
  using(trip_id)
              WHERE session_start >= '2023/01/04'
```

```
                GROUP BY 1
                HAVING COUNT(session_id) > 7
),


 Scalinglt AS (
  SELECT
    AVG(timedifference) AS avg_timed,
    STDDEV(timedifference ) AS stddev_timed
  FROM LongTrip
),

RankFreebag AS (
  SELECT
   ldt.user_id,
COALESCE(( ldt.distance -  avg_dist) / stddev_dist , 0) as
Long_distance_traveller,
COALESCE(( mbns.morebags -  avg_bags) / stddev_bags , 0) as
No_Of_bags,
COALESCE(( lt.timedifference -  avg_timed) / stddev_timed , 0) as
Length_of_trip,

(.33 * COALESCE(( ldt.distance -  avg_dist) / stddev_dist , 0))+
(.33 * COALESCE(( mbns.morebags -  avg_bags) / stddev_bags , 0)) +
(.33 * COALESCE(( lt.timedifference -  avg_timed) / stddev_timed ,
0)) as Index_freebag
FROM longdistancetraveller ldt
LEFT JOIN Hasmorebagsthanseats mbns USING (user_id)
LEFT JOIN LongTrip lt USING (user_id)
CROSS JOIN Scalingldt
CROSS JOIN Scalingbns
CROSS JOIN Scalinglt
GROUP BY 1,2,3,4
  ),


---For No Cancellation Fees

cancellation_percentage AS (
        SELECT
        user_id,
        COALESCE(SUM(CASE WHEN cancellation THEN 1 ELSE 0 END) ::
FLOAT / COUNT(*), 0) AS percentage_of_cancellations_across_sessions,
        COUNT(trip_id) AS count_of_trip,
```

```
        SUM(CASE WHEN cancellation = TRUE THEN 1 ELSE 0 END) AS
cancellation,
        CASE WHEN COUNT(trip_id) > 0 AND SUM(CASE WHEN cancellation =
TRUE THEN 1 ELSE 0 END) > 0 THEN
        COALESCE(SUM(CASE WHEN cancellation THEN 1 ELSE 0 END) ::
FLOAT / COUNT(trip_id), 0) ELSE 0 END AS
percentage_of_trips_cancelled

    FROM sessions
    WHERE session_start >= '2023/01/04'
    AND user_id IN (SELECT user_id FROM cohort)
    GROUP BY user_id
  ),

Scaling_cancellation_percentage AS (
  SELECT
    AVG(percentage_of_trips_cancelled) AS
avg_percentage_of_trips_cancelled,
    STDDEV(percentage_of_trips_cancelled) AS
stddev_percentage_of_trips_cancelled
  FROM cancellation_percentage
),

advanced_flight_booking AS (
   SELECT
    c.user_id,
    AVG(COALESCE(EXTRACT(DAY FROM (departure_time -
session_start)),0)) AS day_difference
FROM
    cohort c
LEFT JOIN sessions USING (user_id)
LEFT JOIN flights USING (trip_id)
GROUP BY 1
),

ScalingAdvancedBooking AS (
  SELECT
    AVG(day_difference) AS avg_day_difference,
    STDDEV(day_difference) AS stddev_day_difference
  FROM advanced_flight_booking
),

conversion_rate AS (
              SELECT
              user_id,
```

```
            COUNT(trip_id) *1.0 /( COUNT(session_id) *1.0  ) AS
conv_rate
            FROM sessions
            WHERE session_start >= '2023/01/04'
            GROUP BY user_id
            HAVING COUNT(session_id) > 7

),

ScalingConvRate AS (
  SELECT avg(conv_rate) AS avg_conv_rate,
  STDDEV(conv_rate) AS stddev_conv_rate
FROM conversion_rate
  ),

Ranked_Free_Cancellation AS (

SELECT
   cp.user_id,
ROUND(COALESCE(( cp.percentage_of_trips_cancelled  -
avg_percentage_of_trips_cancelled)
        / stddev_percentage_of_trips_cancelled , 0) :: Numeric,3) AS
percentage_tips_cancelled_scaled,
ROUND(COALESCE(( afb.day_difference -  avg_day_difference)
        / stddev_day_difference , 0):: Numeric,3) AS
timedifference_scaled,
ROUND(COALESCE((cr.conv_rate - avg_conv_rate)
        /  stddev_conv_rate , 0) :: Numeric,3) AS conv_rate_scaled,

(.33 * COALESCE(( cp.percentage_of_trips_cancelled -
avg_percentage_of_trips_cancelled) /
stddev_percentage_of_trips_cancelled , 0)) +
(.33 * COALESCE(( afb.day_difference -  avg_day_difference) /
stddev_day_difference , 0)) +
(.33 * COALESCE((cr.conv_rate - avg_conv_rate) /  stddev_conv_rate ,
0)) AS Index_Free_Cancellation
FROM cancellation_percentage cp
LEFT JOIN advanced_flight_booking afb USING (user_id)
LEFT JOIN conversion_rate cr USING (user_id)
CROSS JOIN Scaling_cancellation_percentage
CROSS JOIN ScalingAdvancedBooking
CROSS JOIN ScalingConvRate
GROUP BY 1,2,3,4,5
 ),
```

```sql
 --- Free Night

ads_hotel_percentage AS(
              SELECT
              COALESCE( SUM(CASE WHEN hotel_discount THEN 1 ELSE 0
END) :: FLOAT / COUNT(*), 0) AS discount_hotel_proportion,
              user_id
              FROM sessions s
              WHERE session_start >= '2023/01/04'
          GROUP BY s.user_id
          HAVING COUNT(session_id) > 7
),

Scalingdhp AS (
  SELECT
    AVG(discount_hotel_proportion) AS avg_dhp,
    STDDEV(discount_hotel_proportion) AS stddev_dhp
  FROM ads_hotel_percentage
),

avg_h_dis_amt AS (
              SELECT
              COALESCE(AVG(hotel_discount_amount),0) AS
average_hotel_discount,
              user_id
              FROM sessions
              WHERE session_start >= '2023/01/04'
              GROUP BY user_id
              HAVING COUNT(session_id) > 7
),

Scalinghda AS (
  SELECT
    AVG(average_hotel_discount) AS avg_dha,
    STDDEV(average_hotel_discount) AS stddev_dha
  FROM avg_h_dis_amt
),

weekend_traveller AS (
      SELECT
      c.user_id,
      CASE WHEN EXTRACT(dow FROM departure_time) IN (0,5,6) OR
EXTRACT(dow FROM check_in_time) IN (0,5,6)
      THEN 1 ELSE 0 END AS "weekend_traveller?"
      FROM cohort c
```

```
      INNER JOIN sessions s USING (user_id)
      INNER JOIN flights f USING (trip_id)
      INNER JOIN hotels h USING (trip_id)
),
-- We use the table below to figure out whether they have appear more
than twice in the weekend_traveller
-- table. By using SUM we count how many trips they have  taken during
the weekends.
regular_weekend_traveller AS (
      SELECT user_id, CASE WHEN SUM("weekend_traveller?") > 2 THEN 1
      ELSE 0 END AS reg_weekend_traveller
      FROM weekend_traveller
      GROUP BY user_id
  ),

  Scalingwt AS (
  SELECT
    AVG(reg_weekend_traveller) AS avg_wt,
    STDDEV(reg_weekend_traveller) AS stddev_wt
  FROM regular_weekend_traveller
),

  Ranked_freenight AS (
Select ahp.user_id ,
Round(COALESCE(( ahp.discount_hotel_proportion -  avg_dhp) /
stddev_dhp , 0) :: Numeric,  3) AS Hotel_Proportion_Discount,
Round(COALESCE(( ahda.average_hotel_discount -  avg_dha) / stddev_dha
, 0) , 3) AS Hotel_Discount_Amount,
Round(COALESCE(( rwt.reg_weekend_traveller -  avg_wt) / stddev_wt ,
0) , 3) AS Weekend_Traveller,

(.33 * Round(COALESCE(( ahp.discount_hotel_proportion -  avg_dhp) /
stddev_dhp , 0) :: Numeric,  3)) +
(.33 * Round(COALESCE(( ahda.average_hotel_discount -  avg_dha) /
stddev_dha , 0) , 3)) +
(.33 * Round(COALESCE(( rwt.reg_weekend_traveller -  avg_wt) /
stddev_wt , 0) , 3)) AS Index_free_night

FROM ads_hotel_percentage ahp
LEFT JOIN avg_h_dis_amt ahda USING (user_id)
LEFT JOIN regular_weekend_traveller rwt USING (user_id)

CROSS JOIN Scalingdhp
CROSS JOIN Scalinghda
CROSS JOIN Scalingwt
```

```
),


-- This CTE is the give each user a rank for each index/perk. The
lower the value the higher the rank.
-- For example someone ranked 1st in a index should be given that
perk.
Perks_assigned AS (
SELECT DISTINCT c.user_id,
  RANK() OVER(ORDER BY Index_Free_Cancellation DESC) AS
Ranked_Free_Cancellation,
  RANK() OVER(ORDER BY Index_free_night DESC) AS Ranked_Free_Night,
  RANK() OVER(ORDER BY Index_Free_Meal DESC) AS Ranked_Free_Meal,
  RANK() OVER(ORDER BY Index_freebag DESC) AS Ranked_Free_bag,
  RANK() OVER(ORDER BY Index_ExclusiveDiscount DESC) AS
Ranked_exclusive_discount
FROM cohort c
 LEFT JOIN Ranked_freenight USING (user_id)
 LEFT JOIN Ranked_Free_Cancellation USING (user_id)
 LEFT JOIN RankFreebag USING (user_id)
 LEFT JOIN Ranked_Free_Meal USING (user_id)
 LEFT JOIN RankedEXDisc USING (user_id)
GROUP BY
1,Index_Free_Cancellation,Index_free_night,Index_Free_Meal,Index_free
bag,Index_ExclusiveDiscount
)

-- Assignment of perks based on their highest rank across all indexes.
SELECT
    user_id,
    CASE
        WHEN Ranked_Free_Cancellation =
LEAST(Ranked_Free_Cancellation,Ranked_Free_Night,Ranked_Free_Meal,Ran
ked_Free_bag,Ranked_exclusive_discount) THEN 'Free Cancellation'
        WHEN Ranked_Free_Night =
LEAST(Ranked_Free_Cancellation,Ranked_Free_Night,Ranked_Free_Meal,Ran
ked_Free_bag,Ranked_exclusive_discount) THEN 'Free Night'
        WHEN Ranked_Free_Meal =
LEAST(Ranked_Free_Cancellation,Ranked_Free_Night,Ranked_Free_Meal,Ran
ked_Free_bag,Ranked_exclusive_discount) THEN 'Free Meal'
        WHEN Ranked_Free_bag =
LEAST(Ranked_Free_Cancellation,Ranked_Free_Night,Ranked_Free_Meal,Ran
ked_Free_bag,Ranked_exclusive_discount) THEN 'Free Bag'
```

```
        WHEN Ranked_exclusive_discount =
LEAST(Ranked_Free_Cancellation,Ranked_Free_Night,Ranked_Free_Meal,Ran
ked_Free_bag,Ranked_exclusive_discount) THEN 'Exclusive Discount'
        ELSE 'No Perk' -- Handle cases where no perk has the highest
rank
    END AS Perk_assigned
FROM
    Perks_assigned
```