Initial Data Quality Check

```
SELECT DISTINCT hotel_name FROM hotels LIMIT 10000;
-- We partner with 2798 Hotels

SELECT DISTINCT trip_airline FROM flights LIMIT 1000;
-- We work together with 355 airlines

SELECT DISTINCT origin_airport FROM flights LIMIT 1000;
SELECT DISTINCT destination_airport FROM flights LIMIT 1000;
-- We bring you from 144 places to 219 destinations!
-- (Why more destinations than origins? No direct return flights?)


-- I want to check if there is a return flight for every origin in the data set:
-- There are more destination than returns. Maybe because this is not the full data set or we have cancelled flights etc.
```

Descriptive Analysis Part 1

```
-- Generally speaking only around 6% of our customers are seniors older than 64 years old.
-- For that group we can offer High Quality perks, as the costs for perks would be manageable.

-- The vast majority (588k) of our customers are young adults defined as people between 18 and 44 years old and > 36% of those have children!

-- Therefore, children are a major group of indirect customers! We cannot say how many children exactly our customers have because
-- we do not have data for the number of their kids and there might be duplicates if their mother and there father are both our customers.
-- As an estimate we can use our findings of 0,3671 * 588712 = 211116.
-- (i.e. percentage of young adults with children times group size of young adults).
-- Young adults with children (44 years old, that have at least one kid) add up to 211116.

-- Users Table
SELECT * FROM users LIMIT 1000;

--Birth dates to age
SELECT
    DATE_PART('year', AGE(CURRENT_DATE, birthdate)) AS age
FROM users LIMIT 1000;

-- segmentation of age groups
SELECT
    COUNT(*) FILTER (WHERE DATE_PART('year', AGE(CURRENT_DATE, birthdate)) BETWEEN 0 AND 17) AS underaged,
    COUNT(*) FILTER (WHERE DATE_PART('year', AGE(CURRENT_DATE, birthdate)) BETWEEN 18 AND 44) AS young_adults,
    COUNT(*) FILTER (WHERE DATE_PART('year', AGE(CURRENT_DATE, birthdate)) BETWEEN 45 AND 64) AS older_adults,
    COUNT(*) FILTER (WHERE DATE_PART('year', AGE(CURRENT_DATE, birthdate)) >= 65) AS seniors
FROM users
WHERE birthdate IS NOT NULL;
```

```
-- I want to know the percentage of those customer with children (but only for young_adults and adults)
WITH base AS (
    SELECT
        DATE_PART('year', AGE(CURRENT_DATE, birthdate)) AS age,
        has_children
    FROM users
    WHERE birthdate IS NOT NULL
)
SELECT
    age_group,
    ROUND(
        100.0 * COUNT(*) FILTER (WHERE has_children = true)
        / COUNT(*),
        2
    ) AS percent_with_children
FROM (
    SELECT
        CASE
            WHEN age BETWEEN 18 AND 44 THEN 'young_adults'
            WHEN age BETWEEN 45 AND 64 THEN 'older_adults'
        END AS age_group,
        has_children
    FROM base
    WHERE age BETWEEN 18 AND 64
) t
GROUP BY age_group
ORDER BY age_group;
```

Descriptive Analysis Part 2

```
-- HOTELS:
-- What are the 10 most popular hotels?
-- Include the information about the average duration of stay and average price before the discount.
-- I also considered negative values for nights as 1.

SELECT
    hotel_name,
    SUM(GREATEST(nights, 1)) AS nights_spent,
    ROUND(AVG(GREATEST(nights, 1)), 2) AS avg_stay_duration,
    ROUND(
        SUM(hotel_per_room_usd * GREATEST(nights, 1))
        / SUM(GREATEST(nights, 1)),
        2
    ) AS avg_price_before_discount
FROM hotels
GROUP BY hotel_name
ORDER BY nights_spent DESC
LIMIT 10;
```

```
-- The same for most expensive hotels (top 10)

SELECT
    hotel_name,
    ROUND(
        SUM(hotel_per_room_usd * GREATEST(nights, 1))
        / SUM(GREATEST(nights, 1)),
        2
    ) AS avg_price_per_night,
    SUM(GREATEST(nights, 1)) AS total_nights
FROM hotels
GROUP BY hotel_name
HAVING SUM(GREATEST(nights, 1)) > 0
ORDER BY avg_price_per_night DESC
LIMIT 10;

-- The hotels with the longest stays.
SELECT
    hotel_name,
    ROUND(AVG(GREATEST(nights, 1)), 2) AS avg_stay_duration,
    COUNT(*) AS bookings
FROM hotels
GROUP BY hotel_name
ORDER BY avg_stay_duration DESC
LIMIT 10;


FLIGHTS:
-- What is the most used airline in the last 6 months of recorded data?
-- What is the average number of seats booked on flights via TravelTide?

SELECT trip_airline, COUNT (*) as total_flights_last_6_months,
ROUND(AVG(seats), 2) AS avg_seats_booked
FROM flights
WHERE departure_time >= (SELECT MAX (departure_time) FROM flights) - INTERVAL '6
MONTHS'
GROUP BY trip_airline
ORDER BY total_flights_last_6_months DESC
LIMIT 10
;
```

Session-Based Table

```
-- We create the session based table by connecting all four tables into our session base table.


-- Check for the primary key of our fact table sessions
SELECT COUNT (*) as total_rows,
       COUNT (DISTINCT (session_id)) as unique_session_id,
       COUNT (DISTINCT (user_id)) as unique_user_id,
       COUNT (DISTINCT (trip_id)) as trip_id
FROM sessions
LIMIT 100;
--> session_id is the primary key

-- Joining the tables
-- Elena recommended to only include users that had sessions after Jan 4th 2023 (and have more
than 7 sessions)!
```

```
-- As we do not have the column number_of_sessions, we need to create this with the help of a
CTE.
-- We also need to join the CTE to the main table with the use of a sub-query.
-- As there are duplicates in the columns (like trip_id) we select all we need individually.
-- We corrected for negative nights and used 1 night in these cases.
-- We checked the count and had 49k sessions.
-- Furthermore, we have to consider cancellations (10k).


WITH sessions_start_2023  as (
  SELECT *
  FROM sessions
  WHERE session_start > '2023-01-04'
),

filtered_users as (
  SELECT user_id, COUNT (*)
  FROM sessions_start_2023
  GROUP BY user_id
  HAVING COUNT (*) > 7
),

session_based as (
SELECT s.session_id, s.user_id, s.trip_id, s.session_start, s.session_end, s.flight_discount,
s.hotel_discount, s.flight_discount_amount,
  s.hotel_discount_amount, s.flight_booked, s.hotel_booked, s.page_clicks, s.cancellation,
u.birthdate, u.gender, u.married, u.has_children,
  u.home_country, u.home_city, u.home_airport, u.home_airport_lat, u.home_airport_lon,
u.sign_up_date,
  f.origin_airport, f.destination, f.destination_airport, f.seats, f.return_flight_booked,
f.departure_time, f.return_time, f.checked_bags,
  f.trip_airline, f.destination_airport_lat, f.destination_airport_lon, f.base_fare_usd,
  h.hotel_name, CASE WHEN h.nights < 0 THEN 1 ELSE h.nights END as nights_corrected,
h.nights, h.rooms, h.check_in_time, h.check_out_time, h.hotel_per_room_usd as
hotel_room_price
FROM sessions s
  LEFT JOIN flights f ON s.trip_id = f.trip_id
  LEFT JOIN hotels h ON s.trip_id = h.trip_id
  LEFT JOIN users u ON s.user_id = u.user_id
WHERE s.user_id IN (SELECT user_id FROM filtered_users) -- users with more than 7 nights
AND s.session_id IN (SELECT session_id FROM sessions_start_2023) -- startdate is Jan 4th 2023
  ),
```

Start Building User Based Table —> Continuation based on Sessions table:

```
cancelled_trips AS (
    SELECT DISTINCT trip_id
    FROM session_based
    WHERE cancellation = TRUE
),
session_valid AS (                    -- From here on out we are going to build our user_based
table with the help of aggregations
    SELECT *
    FROM session_based
    WHERE cancellation IS DISTINCT FROM TRUE
-- We want to keep "Dreamers" in. The cohort that has sessions but no bookings.
),
```

```
user_based AS (
    SELECT
        user_id,
        COUNT(DISTINCT session_id) AS total_sessions,
        COUNT(DISTINCT trip_id) FILTER (WHERE trip_id IS NOT NULL) AS total_trips,

        SUM(page_clicks) AS total_page_clicks,
        AVG(page_clicks) AS avg_page_clicks,

        AVG(base_fare_usd) FILTER (WHERE flight_booked) AS avg_flight_fare,

        SUM(COALESCE(base_fare_usd,0) + COALESCE(hotel_room_price,0)) AS total_spent,

        CASE
            WHEN COUNT(DISTINCT trip_id) FILTER (WHERE trip_id IS NOT NULL) = 0 THEN NULL
            ELSE
                SUM(COALESCE(base_fare_usd,0) + COALESCE(hotel_room_price,0))
                / COUNT(DISTINCT trip_id) FILTER (WHERE trip_id IS NOT NULL)
        END AS avg_total_spent,

        MIN(birthdate) AS birthdate,
        MAX(has_children::int) AS has_children,
        AVG(seats) AS avg_seats,
        AVG(checked_bags) AS avg_checked_bags
    FROM session_valid
    GROUP BY user_id
),

  user_segments AS (                          -- We start segmenting the dataset into groups.
SELECT
    u.*,
    DATE_PART('year', AGE(CURRENT_DATE, birthdate)) AS age,
CASE
   WHEN DATE_PART('year', AGE(CURRENT_DATE, birthdate)) >= 60 THEN 'Seniors'

   WHEN total_trips = 0
        AND total_page_clicks >= 75 THEN 'Dreamer'

   WHEN avg_seats < 1.5
        AND avg_flight_fare >= 500
        AND total_trips >= 3 THEN 'Business Traveler'

  WHEN DATE_PART ('year', AGE (CURRENT_DATE, birthdate)) BETWEEN 18 AND 55
      AND COALESCE(has_children,0) = 1
      AND avg_seats >= 2.0 THEN 'Young Family'

  WHEN avg_seats BETWEEN 1.5 AND 2.4 THEN 'Couples Leisure'

  WHEN avg_seats < 1.2
  AND avg_total_spent >= 600
  THEN 'Solo Premium Explorer'

  WHEN avg_seats < 1.2
  AND total_page_clicks > 75
  THEN 'Solo Value Traveler'

  ELSE 'Other'

        END AS user_segment
FROM user_based u)
```

```
SELECT * FROM user_based u                          -- Overview
;

/*SELECT                                            -- We can aggregate the groups.
  user_segment,
  COUNT (*) AS users
FROM user_segments
GROUP BY user_segment
ORDER BY users desc
;*/

 SELECT                                             -- Or look at Customer Value by Group.
    user_segment,
    COUNT(*) AS users,
    ROUND(AVG(total_trips),1) AS avg_trips_per_user,
    ROUND(AVG(avg_total_spent),2) AS avg_spend_per_trip,
    ROUND(SUM(total_spent),2) AS segment_total_spent,
    ROUND(AVG(total_spent),2) AS avg_total_spent_per_user
FROM user_segments
GROUP BY user_segment
ORDER BY avg_spend_per_trip DESC
;
```