```
-- Final Query for TravelTide Mastery Project
-- Goal: Aggregate session, flight, and hotel data to user level for behavioral clustering
-- Filters applied: session_start > '2023-01-04', only users with more than 7 sessions

-- Step 1: Session-level feature engineering with joins and calculated fields
WITH session_features AS (
    SELECT
        s.user_id,
        s.trip_id,
        u.gender,
        u.married::int, -- Convert to integer (0 or 1)
        u.has_children::int, -- Convert to integer (0 or 1)
        DATE_PART('year', AGE(s.session_start, u.birthdate)) AS age,

        s.session_id,
        s.page_clicks,
        EXTRACT(EPOCH FROM s.session_end - s.session_start) AS session_duration_sec,
        s.flight_booked,
        s.hotel_booked,
        s.cancellation,
        f.return_flight_booked,
        f.checked_bags,
        f.seats,
        f.base_fare_usd,
        s.flight_discount_amount,
        h.hotel_name,
        h.rooms,
        h.hotel_per_room_usd,
        h.check_in_time,
        h.check_out_time,
        s.session_end,
        s.flight_discount,
        s.hotel_discount,
        f.departure_time,

        -- Calculate trip distance using the Haversine formula
        6371 * 2 * ASIN(
            SQRT(
                POWER(SIN(RADIANS((f.destination_airport_lat - u.home_airport_lat) / 2)), 2) +
                COS(RADIANS(u.home_airport_lat)) * COS(RADIANS(f.destination_airport_lat)) *
                POWER(SIN(RADIANS((f.destination_airport_lon - u.home_airport_lon) / 2)), 2)
            )
        )::numeric AS distance_km,

        -- Days between booking and trip start (either flight or hotel)
        CASE
            WHEN f.departure_time IS NOT NULL THEN (f.departure_time::date - s.session_end::date)
            WHEN h.check_in_time IS NOT NULL THEN (h.check_in_time::date - s.session_end::date)
            ELSE NULL
        END AS days_booking_to_trip,

        -- Money spent on flight bookings (accounting for discounts and roundtrips)
        CASE
            WHEN s.flight_booked AND NOT s.cancellation THEN
                f.base_fare_usd * f.seats * (CASE WHEN f.return_flight_booked THEN 2 ELSE 1 END) * (1 - C
            ELSE 0
        END AS money_spent_flight,

        -- Money spent on hotel bookings (accounting for discounts and duration)
        CASE
            WHEN s.hotel_booked = TRUE AND s.cancellation = FALSE THEN
                h.hotel_per_room_usd * h.rooms *
                (GREATEST(h.check_out_time::date, h.check_in_time::date) -
                 LEAST(h.check_out_time::date, h.check_in_time::date)) *
                (1 - COALESCE(s.hotel_discount_amount, 0))
            ELSE 0
        END AS money_spent_hotel,

        -- Trip duration in days
        CASE
            WHEN f.departure_time IS NOT NULL AND f.return_time IS NOT NULL THEN
```

```
              (f.return_time::date - f.departure_time::date)
            WHEN f.departure_time IS NULL AND h.check_in_time IS NOT NULL AND h.check_out_time IS NOT NUL
              (GREATEST(h.check_out_time::date, h.check_in_time::date) -
               LEAST(h.check_out_time::date, h.check_in_time::date))
            ELSE NULL
        END AS trip_duration_days

-- Step 1: Sessions only > 2023-01-04
    FROM sessions s
    LEFT JOIN flights f USING (trip_id)
    LEFT JOIN hotels h USING (trip_id)
    LEFT JOIN users u USING (user_id)
    WHERE s.session_start > '2023-01-04'
),

-- Step 2: Keep only users with >7 sessions
filtered_users AS (
    SELECT user_id
    FROM session_features
    GROUP BY user_id
    HAVING COUNT(session_id) > 7
),

-- Step 3: Define completed trips (booked and not cancelled)
completed_trips AS (
    SELECT DISTINCT s1.user_id, s1.trip_id
    FROM sessions s1
    WHERE s1.cancellation = FALSE
      AND NOT EXISTS (
          SELECT 1 FROM sessions s2
          WHERE s2.trip_id = s1.trip_id AND s2.cancellation = TRUE
      )
),

-- Step 4: Aggregate all engineered features at the user level
final_features AS (
    SELECT
        sf.user_id,
        ROUND(AVG(age)) AS age,
        MIN(sf.married) AS married,
        MIN(sf.has_children) AS has_children,

        COUNT(*) AS total_sessions,
        SUM(sf.page_clicks) AS total_clicks,
        ROUND(AVG(sf.session_duration_sec)) AS avg_session_duration_sec,

        COUNT(DISTINCT ct.trip_id) AS total_completed_trips,
        ROUND(COUNT(DISTINCT sf.trip_id)::numeric / COUNT(*), 2) AS booking_conversion_rate,

        SUM(CASE WHEN ct.trip_id IS NOT NULL THEN sf.checked_bags ELSE 0 END) AS total_checked_bags,
        ROUND(SUM(CASE WHEN ct.trip_id IS NOT NULL THEN sf.distance_km ELSE 0 END)) AS total_distance_km,

        ROUND(SUM(CASE WHEN ct.trip_id IS NOT NULL THEN sf.money_spent_flight ELSE 0 END), 1) AS money_sp
        ROUND(SUM(CASE WHEN ct.trip_id IS NOT NULL THEN sf.money_spent_hotel ELSE 0 END), 1) AS money_spe

        -- Hotel loyalty score: inverse of number of distinct hotel brands
        ROUND(
            CASE
                WHEN COUNT(DISTINCT CASE
                      WHEN sf.hotel_booked AND ct.trip_id IS NOT NULL
                      THEN SPLIT_PART(sf.hotel_name, ' - ', 1)
                      ELSE NULL
                END) > 0
                THEN 1.0 / COUNT(DISTINCT CASE
                      WHEN sf.hotel_booked AND ct.trip_id IS NOT NULL
                      THEN SPLIT_PART(sf.hotel_name, ' - ', 1)
                      ELSE NULL
                END)
                ELSE 0
            END, 3
        ) AS hotel_loyalty_score,
```

```
        ROUND(AVG(sf.days_booking_to_trip) FILTER (WHERE ct.trip_id IS NOT NULL)) AS avg_days_booking_to_

        -- Trip duration
        ROUND(AVG(sf.trip_duration_days) FILTER (WHERE ct.trip_id IS NOT NULL)) AS avg_trip_duration_days

        -- Booking breakdown: flight-only, hotel-only, both
        ROUND(
            SUM(CASE
                    WHEN sf.flight_booked AND NOT sf.hotel_booked AND NOT sf.cancellation
                    THEN 1 ELSE 0
                END)::numeric
            / NULLIF(COUNT(DISTINCT sf.trip_id), 0),
        2
        ) AS flight_only_rate,

        ROUND(
            SUM(CASE
                    WHEN sf.hotel_booked AND NOT sf.flight_booked AND NOT sf.cancellation
                    THEN 1 ELSE 0
                END)::numeric
            / NULLIF(COUNT(DISTINCT sf.trip_id), 0),
        2
        ) AS hotel_only_rate,

        ROUND(
            SUM(CASE
                    WHEN sf.hotel_booked AND sf.flight_booked AND NOT sf.cancellation
                    THEN 1 ELSE 0
                END)::numeric
            / NULLIF(COUNT(DISTINCT sf.trip_id), 0),
        2
        ) AS both_booked_rate,

        -- Perk-related: discount usage and cancellations
        ROUND(
            SUM(CASE
                    WHEN (sf.flight_discount OR sf.hotel_discount)
                         AND (sf.hotel_booked OR sf.flight_booked)
                         AND NOT sf.cancellation
                    THEN 1 ELSE 0
                END)::numeric
            / NULLIF(COUNT(DISTINCT sf.trip_id), 0),
        2
        ) AS discount_usage_rate,

        ROUND(
            SUM(CASE WHEN sf.cancellation THEN 1 ELSE 0 END)::numeric
            / NULLIF(COUNT(DISTINCT sf.trip_id), 0),
          2
          ) AS cancellation_per_booking_rate

    FROM session_features sf
    JOIN filtered_users fu ON fu.user_id = sf.user_id
    LEFT JOIN completed_trips ct ON ct.user_id = sf.user_id AND ct.trip_id = sf.trip_id
    GROUP BY sf.user_id
)

-- Final Output: User-level dataset with behavioral metrics for clustering
SELECT * FROM final_features
ORDER BY user_id;
```