```
/* TRAVELTIDE COHORT EXTRACTION
*/

-- 1. Filter sessions for Elena's timeframe
WITH sessions_2023 AS (
  SELECT *
  FROM sessions s
  WHERE s.session_start > '2023-01-04'
),

-- 2. Identify high-engagement users (> 7 sessions)
filtered_users AS (
  SELECT user_id
  FROM sessions_2023 s
  GROUP BY user_id
  HAVING COUNT(*) > 7
),

-- 3. Creating the session base (Joining Users, Flights, and Hotels)
session_base AS (
  SELECT
    s.session_id,
    s.user_id,
    s.trip_id,
    s.session_start,
    s.session_end,
    EXTRACT(EPOCH FROM s.session_end - s.session_start) AS
session_duration,
    s.page_clicks,
    s.flight_discount,
    s.flight_discount_amount,
    s.hotel_discount,
    s.hotel_discount_amount,
    s.flight_booked,
    s.hotel_booked,
    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,
    h.rooms,
    h.check_in_time,
    h.check_out_time,
    h.hotel_per_room_usd AS hotel_price_per_room_night_usd
  FROM sessions_2023 s
  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 s.user_id IN (SELECT user_id FROM filtered_users)
),

-- 4. Identifying and isolating canceled trips
canceled_trips AS (
  SELECT DISTINCT trip_id
  FROM session_base
  WHERE cancellation = TRUE
),

-- 5. Focus only on valid, completed trips
not_canceled_trips AS (
  SELECT *
  FROM session_base
  WHERE trip_id IS NOT NULL
    AND trip_id NOT IN (SELECT trip_id FROM canceled_trips)
),

-- 6. Aggregate browsing behavior
user_base_session AS (
  SELECT
    user_id,
    COUNT(DISTINCT session_id) AS num_sessions,
    SUM(page_clicks) AS num_clicks,
    AVG(EXTRACT(EPOCH FROM (session_end - session_start)))::INT AS
session_seconds,
```

```
    COUNT(trip_id) AS total_trips
  FROM session_base
  GROUP BY user_id
),

-- 7. Aggregate travel metrics
user_base_trip AS (
  SELECT
    user_id,
    COUNT(DISTINCT trip_id) AS num_valid_trips,
    SUM(CASE
          WHEN (flight_booked = TRUE) AND (return_flight_booked = TRUE)
THEN 2
          WHEN (flight_booked = TRUE) THEN 1
          ELSE 0
        END) AS num_flights,
    AVG(EXTRACT(DAY FROM departure_time - session_end))::INT AS
days_after_booking,
    AVG(haversine_distance(home_airport_lat, home_airport_lon,
destination_airport_lat, destination_airport_lon))::INT AS avg_km_flown,
    SUM(hotel_price_per_room_night_usd * rooms * nights * (1 -
COALESCE(hotel_discount_amount, 0))) AS revenue_hotel_usd,
    AVG(checked_bags) AS luggage_amount,
    AVG(nights) AS avg_nights
  FROM not_canceled_trips
  GROUP BY user_id
),

-- 8. Building the Final User Table
final_user_table AS (
  SELECT
    u.user_id,
    u.gender,
    u.married,
    u.has_children,
    u.home_country,
    u.home_city,
    EXTRACT(YEAR FROM AGE(u.birthdate)) AS age,
    ubs.num_sessions,
    ubs.num_clicks,
    ubs.session_seconds,
    ubs.total_trips,
    ubt.num_valid_trips,
    ubt.num_flights,
    ubt.days_after_booking,
    ubt.avg_km_flown,
    ubt.avg_nights,
```

```
    ubt.revenue_hotel_usd,
    ROUND(ubt.luggage_amount, 1) AS luggage_amount
  FROM user_base_session ubs
  LEFT JOIN user_base_trip ubt ON ubs.user_id = ubt.user_id
  JOIN users u ON ubs.user_id = u.user_id
),

-- 9. Assign Cohorts
aux AS (
  SELECT
    user_id,
    CASE
      WHEN total_trips = 0 THEN 'Dreamer'
      WHEN age <= 25 THEN 'Young Traveler'
      WHEN age >= 60 THEN 'Senior Traveler'
      WHEN age BETWEEN 25 AND 60
           AND married = TRUE
           AND has_children = TRUE THEN 'Family Traveler'
      WHEN age BETWEEN 25 AND 60
           AND days_after_booking <= 10
           AND avg_nights <= 3 THEN 'Business Traveler'
      ELSE 'Standard Travelers'
    END AS traveler_cohort
  FROM final_user_table
)

-- FINAL OUTPUT: The result table for presentation
SELECT
  traveler_cohort,
  COUNT(DISTINCT user_id) AS user_count
FROM aux
GROUP BY traveler_cohort
ORDER BY user_count DESC;
```